百问网 LVGL 中文手册 8.2

www.100ask.net

2024 年 05 月 15 日

Contents

PDF version: 100ASK_LVGL_CN.pdf

CHAPTER 1

文档所有版本

- LVGL V7.11: http://lvgl.100ask.net/7.11
- LVGL V8.1: http://lvgl.100ask.net/8.1
- LVGL V8.2: http://lvgl.100ask.net/8.2
- LVGL V8.3: http://lvgl.100ask.net/8.3
- LVGL V9.0: http://lvgl.100ask.net/9.0
- LVGL master(当前最新版本): http://lvgl.100ask.net/master

CHAPTER 2

百问网 LVGL 视频教程 (韦东山 · 监制)

教程基于 lvgl v8.2 版本，适配多个平台、多款板子。

视频地址：https://www.bilibili.com/video/BV1Ya411r7K2

2.1 Introduction （介绍）

LVGL (Light and Versatile Graphics Library) is a free and open-source graphics library providing everything you need to create embedded GUI with easy-to-use graphical elements, beautiful visual effects and low memory footprint.

LVGL(轻量级和通用图形库) 是一个免费和开源的图形库，它提供了创建嵌入式 GUI 所需的一切，具有易于使用的图形元素，美丽的视觉效果和低内存占用。

```
# 译者添加：

LVGL 的项目作者是来自匈牙利首都布达佩斯的 Gábor Kiss-Vámosi 。Kiss 在 2016 年将其并发布在␣
↪[GitHub](https://github.com/lvgl) 上。

当时叫 LittlevGL 而不是 LVGL，后来作者统一修改为 LVGL 甚至连仓库地址都改了。 像一般的开源项目的那样，它是作为一个人的项目开始的。 从那时起，陆续有近 100 名贡献者参与了项目开发，使得 LVGL 逐渐成为最受欢迎的嵌入式图形库之一。
```

2.1.1 Key features （主要特性）

- Powerful building blocks such as buttons, charts, lists, sliders, images, etc.
- Advanced graphics with animations, anti-aliasing, opacity, smooth scrolling
- Various input devices such as touchpad, mouse, keyboard, encoder, etc.
- Multi-language support with UTF-8 encoding
- Multi-display support, i.e. use multiple TFT, monochrome displays simultaneously
- Fully customizable graphic elements with CSS-like styles
- Hardware independent: use with any microcontroller or display
- Scalable: able to operate with little memory (64 kB Flash, 16 kB RAM)
- OS, external memory and GPU are supported but not required
- Single frame buffer operation even with advanced graphic effects
- Written in C for maximal compatibility (C++ compatible)
- Simulator to start embedded GUI design on a PC without embedded hardware
- Binding to MicroPython
- Tutorials, examples, themes for rapid GUI design
- Documentation is available online and as PDF
- Free and open-source under MIT license
- 丰富且强大的模块化图形组件：按钮 (buttons)、图表 (charts)、列表 (lists)、滑动条 (sliders)、图片 (images) 等
- 高级的图形引擎：动画、抗锯齿、透明度、平滑滚动、图层混合等效果
- 支持多种输入设备：触摸屏、键盘、编码器、按键等
- 支持多显示设备
- 不依赖特定的硬件平台，可以在任何显示屏上运行
- 配置可裁剪（最低资源占用：64 kB Flash，16 kB RAM）
- 基于 UTF-8 的多语种支持，例如中文、日文、韩文、阿拉伯文等
- 可以通过类 CSS的方式来设计、布局图形界面（例如：Flexbox、Grid）
- 支持操作系统、外置内存、以及硬件加速（LVGL 已内建支持 STM32 DMA2D、NXP PXP 和 VGLite）
- 即便仅有单缓冲区 (frame buffer)的情况下，也可保证渲染如丝般顺滑
- 全部由 C 编写完成，并支持 C++ 调用
- 支持 Micropython 编程，参见：LVGL API in Micropython

- 支持模拟器仿真，可以无硬件依托进行开发
- 丰富详实的例程
- 详尽的文档以及 API 参考手册，可线上查阅或可下载为 PDF 格式
- 在 MIT 许可下免费和开源

2.1.2 Requirements（配置要求）

Basically, every modern controller which is able to drive a display is suitable to run LVGL. The minimal requirements are:

基本上，每个能够驱动显示器的现代控制器都适合运行 LVGL。最低要求是：

2.1.3 License（许可证）

The LVGL project (including all repositories) is licensed under MIT license. This means you can use it even in commercial projects.

It's not mandatory but we highly appreciate it if you write a few words about your project in the My projects category of the forum or a private message to lvgl.io.

Although you can get LVGL for free there is a massive amount of work behind it. It's created by a group of volunteers who made it available for you in their free time.

To make the LVGL project sustainable, please consider *contributing* to the project. You can choose from *many different ways of contributing* such as simply writing a tweet about you are using LVGL, fixing bugs, translating the documentation, or even becoming a maintainer.

LVGL 项目（包括所有存储库）在 MIT license 许可下获得许可。这意味着您甚至可以在商业项目中使用它。

这不是强制性的，但如果您在论坛的 My projects 类别或来自 lvgl.io 的私人消息中写下有关您的项目的几句话，我们将不胜感激。

尽管您可以免费获得 LVGL，但它背后的工作量很大。它由一群志愿者创建，他们在空闲时间为您提供。

为了使 LVGL 项目可持续，请考虑为该项目做贡献 。您可以从多种投稿方式 中进行选择，例如简单地写一条关于您正在使用 LVGL 的推文、修复错误、翻译文档，甚至成为维护者。

2.1.4 Repository layout （仓库布局）

All repositories of the LVGL project are hosted n GitHub: https://github.com/lvgl

LVGL 项目的所有代码仓库都托管在 GitHub 上：https://github.com/lvgl

You will find these repositories there:

- lvgl The library itself with many examples.
- lv_demos Demos created with LVGL.
- lv_drivers Display and input device drivers
- blog Source of the blog's site (https://blog.lvgl.io)
- sim Source of the online simulator's site (https://sim.lvgl.io)
- lv_sim_... Simulator projects for various IDEs and platforms
- lv_port_... LVGL ports to development boards
- lv_binding_.. Bindings to other languages
- lv_... Ports to other platforms

您可以从下面的列表获取到 lvgl 所有相关的代码仓库：

- lvgl 库本身有很多 例子。
- lv_demos 使用 LVGL 创建的演示。
- lv_drivers 显示和输入设备驱动程序
- 博客 博客站点的来源 (https://blog.lvgl.io)
- sim 在线模拟器网站的来源 (https://sim.lvgl.io)
- lv_sim_... 各种 IDE 和平台的模拟器项目
- lv_port_... LVGL 端口到开发板
- lv_binding_.. 绑定到其他语言
- lv_... 移植到其他平台

2.1.5 Release policy（发布策略）

The core repositories follow the rules of Semantic versioning:

- Major versions for incompatible API changes. E.g. v5.0.0, v6.0.0
- Minor version for new but backward-compatible functionalities. E.g. v6.1.0, v6.2.0
- Patch version for backward-compatible bug fixes. E.g. v6.1.1, v6.1.2

Tags like `vX.Y.Z` are created for every release.

LVGL 库遵循语义版本管理：

- 不兼容 API 更改的主要版本。比如：v5.0.0, v6.0.0
- 新的但向后兼容的功能的次要版本。比如：v6.1.0, v6.2.0
- 用于向后兼容错误修复的补丁版本。比如：v6.1.1, v6.1.2

为每个版本创建诸如“vX.Y.Z”之类的标签。

Release cycle（发布周期）

- Bug fixes: Released on demand even weekly
- Minor releases: Every 3-4 months
- Major releases: Approximately yearly
- 错误修复：每周按需发布
- 次要版本：每 3-4 个月
- 主要版本：大约每年

Branches（分支）

The core repositories have at least the following branches:

- `master` latest version, patches are merged directly here.
- `release/vX.Y` stable versions of the minor releases
- `fix/some-description` temporary branches for bug fixes
- `feat/some-description` temporary branches for features

代码仓库至少有以下分支：

- `master` 最新版本，补丁在这里直接合并。
- `release/vX.Y` 稳定版本的次要版本
- 用于错误修复的 `fix/some-description` 临时分支
- 开发功能的 `feat/some-description` 临时分支

Changelog（变更日志）

The changes are recorded in CHANGELOG.md.

更改记录在 CHANGELOG.md 中。

Version support(版本支持)

Before v8 every minor release of major releases is supported for 1 year. Starting from v8, every minor release is supported for 1 year.

在 v8 之前，每个主要版本的次要版本都支持 1 年。从 v8 开始，每个次要版本都支持 1 年。

2.1.6 FAQ（常见问题）

Where can I ask questions?（我可以在哪里提问？）

You can ask questions in the forum: https://forum.lvgl.io/.

We use GitHub issues for development related discussion. You should use them only if your question or issue is tightly related to the development of the library.

可以在论坛提问：https://forum.lvgl.io/。

我们使用 GitHub 问题 进行开发相关讨论。仅当您的问题或问题与库的开发密切相关时才应使用它们。

Is my MCU/hardware supported?（LVGL 是否支持我的 MCU/硬件？）

Every MCU which is capable of driving a display via parallel port, SPI, RGB interface or anything else and fulfills the *Requirements* is supported by LVGL.

This includes:

- ”Common” MCUs like STM32F, STM32H, NXP Kinetis, LPC, iMX, dsPIC33, PIC32 etc.
- Bluetooth, GSM, Wi-Fi modules like Nordic NRF and Espressif ESP32
- Linux with frame buffer device such as /dev/fb0. This includes Single-board computers like the Raspberry Pi
- Anything else with a strong enough MCU and a peripheral to drive a display

LVGL 支持每个能够通过并行端口、SPI、RGB 接口或其他任何方式驱动显示器并满足要求 的 MCU。

这包括：

- “通用”MCU，如 STM32F、STM32H、NXP Kinetis、LPC、IMX、dsPIC33、PIC32 等。
- 蓝牙、GSM、Wi-Fi 模块，如 Nordic NRF 和 Espressif ESP32
- 带有帧缓冲设备的 Linux，例如 /dev/fb0。这包括单板计算机，如 Raspberry Pi

- 任何其他具有足够强大 MCU 和外围设备来驱动显示器的设备

Is my display supported?（支持我的显示器吗 ?）

LVGL needs just one simple driver function to copy an array of pixels into a given area of the display. If you can do this with your display then you can use it with LVGL.

Some examples of the supported display types:

- TFTs with 16 or 24 bit color depth
- Monitors with an HDMI port
- Small monochrome displays
- Gray-scale displays
- even LED matrices
- or any other display where you can control the color/state of the pixels

See the *Porting* section to learn more.

LVGL 只需要一个简单的驱动程序函数即可将像素阵列复制到显示器的给定区域。如果您可以在显示器上执行此操作，那么您可以将其与 LVGL 一起使用。

支持的显示类型的一些示例：

- 具有 16 位或 24 位色深的 TFT
- 带有 HDMI 端口的显示器
- 小型单色显示器
- 灰度显示
- 甚至 LED 矩阵
- 或任何其他可以控制像素颜色/状态的显示器

请参阅移植 部分以了解更多信息。

Nothing happens, my display driver is not called. What have I missed?（没有任何反应，我的显示驱动程序没有被调用。我错过了什么 ?）

Be sure you are calling `lv_tick_inc(x)` in an interrupt and `lv_timer_handler()` in your main `while(1)`.

Learn more in the *Tick* and *Task handler* sections.

确保你在中断中调用了 `lv_tick_inc(x)`，在你的主 `while(1)` 中调用了 `lv_timer_handler()`。

在*Tick* 和任务处理程序 部分了解更多信息。

Why the display driver is called only once? Only the upper part of the display is refreshed. （为什么显示驱动程序只调用一次？仅刷新显示的上部。）

Be sure you are calling `lv_disp_flush_ready(drv)` at the end of your "*display flush callback*".

确保在“显示刷新回调”结束时调用 `lv_disp_flush_ready(drv)` 。

Why I see only garbage on the screen?（为什么我在屏幕上只看到垃圾？）

Probably there a bug in your display driver. Try the following code without using LVGL. You should see a square with red-blue gradient

您的显示驱动程序中可能存在错误。在不使用 LVGL 的情况下尝试以下代码。你应该看到一个带有红蓝渐变的正方形

```
#define BUF_W 20
#define BUF_H 10

lv_color_t buf[BUF_W * BUF_H];
lv_color_t * buf_p = buf;
uint16_t x, y;
for(y = 0; y &lt; BUF_H; y++) {
    lv_color_t c = lv_color_mix(LV_COLOR_BLUE, LV_COLOR_RED, (y * 255) / BUF_H);
    for(x = 0; x &lt; BUF_W; x++){
        (*buf_p) =  c;
        buf_p++;
    }
}

lv_area_t a;
a.x1 = 10;
a.y1 = 40;
a.x2 = a.x1 + BUF_W - 1;
a.y2 = a.y1 + BUF_H - 1;
my_flush_cb(NULL, &a, buf);
```

Why I see non-sense colors on the screen? （为什么我在屏幕上看到无意义的颜色？）

Probably LVGL's color format is not compatible with your display's color format. Check `LV_COLOR_DEPTH` in *lv_conf.h*.

If you are using 16-bit colors with SPI (or another byte-oriented interface) you probably need to set `LV_COLOR_16_SWAP  1` in *lv_conf.h*. It swaps the upper and lower bytes of the pixels.

可能 LVGL 的颜色格式与您的显示器的颜色格式不兼容。检查 *lv_conf.h* 中的 `LV_COLOR_DEPTH`。

如果您在 SPI（或其他面向字节的接口）中使用 16 位颜色，您可能需要在 *lv_conf.h* 中设置 “LV_COLOR_16_SWAP 1”。它交换像素的高字节和低字节。

How to speed up my UI?（如何加速我的用户界面？）

- Turn on compiler optimization and enable cache if your MCU has it
- Increase the size of the display buffer
- Use two display buffers and flush the buffer with DMA (or similar peripheral) in the background
- Increase the clock speed of the SPI or parallel port if you use them to drive the display
- If your display has a SPI port consider changing to a model with a parallel interface because it has much higher throughput
- Keep the display buffer in internal RAM (not in external SRAM) because LVGL uses it a lot and it should have a fast access time
- 如果您的 MCU 支持的话，请打开编译器优化并启用缓存
- 增加显示缓冲区的大小
- 使用 2 个显示缓冲区并在后台使用 DMA（或类似外围设备）刷新缓冲区
- 如果您使用 SPI 或并行端口来驱动显示器，请提高它们的时钟速度
- 如果您的显示器具有 SPI 端口，请考虑更改为并行模型，因为它具有更高的吞吐量
- 将显示缓冲区保留在内部 RAM（而不是外部 SRAM）中，因为 LVGL 经常使用它，并且访问时间应该很短

How to reduce flash/ROM usage?（如何减少闪存/ROM 的使用？）

You can disable all the unused features (such as animations, file system, GPU etc.) and object types in *lv_conf.h*.

If you are using GCC you can add `-fdata-sections -ffunction-sections` compiler flags and `--gc-sections` linker flag to remove unused functions and variables from the final binary.

您可以在 *lv_conf.h* 中禁用所有未使用的功能（例如动画、文件系统、GPU 等）和对象类型。

如果您使用 GCC，您可以添加`-fdata-sections -ffunction-sections` 编译器标志和`--gc-sections` 链接器标志，以从最终二进制文件中删除未使用的函数和变量。

How to reduce the RAM usage（如何减少内存使用）

- Lower the size of the *Display buffer*
- Reduce `LV_MEM_SIZE` in *lv_conf.h*. This memory is used when you create objects like buttons, labels, etc.
- To work with lower `LV_MEM_SIZE` you can create objects only when required and delete them when they are not needed anymore
- 降低显示缓冲区的大小
- 减少 *lv_conf.h* 中的 `LV_MEM_SIZE`。当您创建按钮、标签等对象时会使用此内存。
- 要使用较低的“LV_MEM_SIZE”，您可以仅在需要时创建对象并在不再需要时将其删除

How to work with an operating system?（如何使用操作系统？）

To work with an operating system where tasks can interrupt each other (preemptively) you should protect LVGL related function calls with a mutex. See the *Operating system and interrupts* section to learn more.

要使用任务可以相互中断（抢占式）的操作系统，您应该使用互斥锁保护与 LVGL 相关的函数调用。请参阅操作系统和中断 部分以了解更多信息。

2.2 Examples

2.2.1 Get started

2.2.2 Styles

Size styles

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_IMG

/**
 * Using the Size, Position and Padding style properties
 */
void lv_example_style_1(void)
{
     static lv_style_t style;
     lv_style_init(&style);
     lv_style_set_radius(&style, 5);

     /*Make a gradient*/
```

(下页继续)

(续上页)

```
    lv_style_set_width(&style, 150);
    lv_style_set_height(&style, LV_SIZE_CONTENT);

    lv_style_set_pad_ver(&style, 20);
    lv_style_set_pad_left(&style, 5);

    lv_style_set_x(&style, lv_pct(50));
    lv_style_set_y(&style, 80);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_scr_act());
    lv_obj_add_style(obj, &style, 0);

    lv_obj_t * label = lv_label_create(obj);
    lv_label_set_text(label, "Hello");
}

#endif
```

```
#
# Using the Size, Position and Padding style properties
#
style = lv.style_t()
style.init()
style.set_radius(5)

# Make a gradient
style.set_width(150)
style.set_height(lv.SIZE.CONTENT)

style.set_pad_ver(20)
style.set_pad_left(5)

style.set_x(lv.pct(50))
style.set_y(80)

# Create an object with the new style
obj = lv.obj(lv.scr_act())
obj.add_style(style, 0)

label = lv.label(obj)
label.set_text("Hello")
```

Background styles

```c
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

/**
 * Using the background style properties
 */
void lv_example_style_2(void)
{
    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_radius(&style, 5);

    /*Make a gradient*/
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 1));
    lv_style_set_bg_grad_color(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_bg_grad_dir(&style, LV_GRAD_DIR_VER);

    /*Shift the gradient to the bottom*/
    lv_style_set_bg_main_stop(&style, 128);
    lv_style_set_bg_grad_stop(&style, 192);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_scr_act());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_center(obj);
}

#endif
```

```python
#
# Using the background style properties
#
style = lv.style_t()
style.init()
style.set_radius(5)

# Make a gradient
style.set_bg_opa(lv.OPA.COVER)
style.set_bg_color(lv.palette_lighten(lv.PALETTE.GREY, 1))
style.set_bg_grad_color(lv.palette_main(lv.PALETTE.BLUE))
style.set_bg_grad_dir(lv.GRAD_DIR.VER)
```

(下页继续)

(续上页)

```
# Shift the gradient to the bottom
style.set_bg_main_stop(128)
style.set_bg_grad_stop(192)

# Create an object with the new style
obj = lv.obj(lv.scr_act())
obj.add_style(style, 0)
obj.center()
```

Border styles

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

/**
 * Using the border style properties
 */
void lv_example_style_3(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    /*Set a background color and a radius*/
    lv_style_set_radius(&style, 10);
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 1));

    /*Add border to the bottom+right*/
    lv_style_set_border_color(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_border_width(&style, 5);
    lv_style_set_border_opa(&style, LV_OPA_50);
    lv_style_set_border_side(&style, LV_BORDER_SIDE_BOTTOM | LV_BORDER_SIDE_RIGHT);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_scr_act());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_center(obj);
}

#endif
```

```
#
# Using the border style properties
#
style = lv.style_t()
style.init()

# Set a background color and a radius
style.set_radius(10)
style.set_bg_opa(lv.OPA.COVER)
style.set_bg_color(lv.palette_lighten(lv.PALETTE.GREY, 1))

# Add border to the bottom+right
style.set_border_color(lv.palette_main(lv.PALETTE.BLUE))
style.set_border_width(5)
style.set_border_opa(lv.OPA._50)
style.set_border_side(lv.BORDER_SIDE.BOTTOM | lv.BORDER_SIDE.RIGHT)

# Create an object with the new style
obj = lv.obj(lv.scr_act())
obj.add_style(style, 0)
obj.center()
```

Outline styles

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

/**
 * Using the outline style properties
 */
void lv_example_style_4(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    /*Set a background color and a radius*/
    lv_style_set_radius(&style, 5);
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 1));

    /*Add outline*/
    lv_style_set_outline_width(&style, 2);
    lv_style_set_outline_color(&style, lv_palette_main(LV_PALETTE_BLUE));
```

(下页继续)

(续上页)

```
    lv_style_set_outline_pad(&style, 8);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_scr_act());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_center(obj);
}

#endif
```

```
#
# Using the outline style properties
#

style = lv.style_t()
style.init()

# Set a background color and a radius
style.set_radius(5)
style.set_bg_opa(lv.OPA.COVER)
style.set_bg_color(lv.palette_lighten(lv.PALETTE.GREY, 1))

# Add outline
style.set_outline_width(2)
style.set_outline_color(lv.palette_main(lv.PALETTE.BLUE))
style.set_outline_pad(8)

# Create an object with the new style
obj = lv.obj(lv.scr_act())
obj.add_style(style, 0)
obj.center()
```

Shadow styles

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

/**
 * Using the Shadow style properties
 */
void lv_example_style_5(void)
```

(下页继续)

(续上页)

```
{
    static lv_style_t style;
    lv_style_init(&style);

    /*Set a background color and a radius*/
    lv_style_set_radius(&style, 5);
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 1));

    /*Add a shadow*/
    lv_style_set_shadow_width(&style, 55);
    lv_style_set_shadow_color(&style, lv_palette_main(LV_PALETTE_BLUE));
//    lv_style_set_shadow_ofs_x(&style, 10);
//    lv_style_set_shadow_ofs_y(&style, 20);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_scr_act());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_center(obj);
}

#endif
```

```
#
# Using the Shadow style properties
#

style = lv.style_t()
style.init()

# Set a background color and a radius
style.set_radius(5)
style.set_bg_opa(lv.OPA.COVER)
style.set_bg_color(lv.palette_lighten(lv.PALETTE.GREY, 1))

# Add a shadow
style.set_shadow_width(8)
style.set_shadow_color(lv.palette_main(lv.PALETTE.BLUE))
style.set_shadow_ofs_x(10)
style.set_shadow_ofs_y(20)

# Create an object with the new style
obj = lv.obj(lv.scr_act())
```

(下页继续)

(续上页)

```
obj.add_style(style, 0)
obj.center()
```

Image styles

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_IMG

/**
 * Using the Image style properties
 */
void lv_example_style_6(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    /*Set a background color and a radius*/
    lv_style_set_radius(&style, 5);
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 3));
    lv_style_set_border_width(&style, 2);
    lv_style_set_border_color(&style, lv_palette_main(LV_PALETTE_BLUE));

    lv_style_set_img_recolor(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_img_recolor_opa(&style, LV_OPA_50);
    lv_style_set_transform_angle(&style, 300);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_img_create(lv_scr_act());
    lv_obj_add_style(obj, &style, 0);

    LV_IMG_DECLARE(img_cogwheel_argb);
    lv_img_set_src(obj, &img_cogwheel_argb);

    lv_obj_center(obj);
}

#endif
```

```
from imagetools import get_png_info, open_png
# Register PNG image decoder
decoder = lv.img.decoder_create()
```

(下页继续)

(续上页)

```
decoder.info_cb = get_png_info
decoder.open_cb = open_png

# Create an image from the png file
try:
    with open('../assets/img_cogwheel_argb.png', 'rb') as f:
        png_data = f.read()
except:
    print("Could not find img_cogwheel_argb.png")
    sys.exit()

img_cogwheel_argb = lv.img_dsc_t({
  'data_size': len(png_data),
  'data': png_data
})

#
# Using the Image style properties
#
style = lv.style_t()
style.init()

# Set a background color and a radius
style.set_radius(5)
style.set_bg_opa(lv.OPA.COVER)
style.set_bg_color(lv.palette_lighten(lv.PALETTE.GREY, 3))
style.set_border_width(2)
style.set_border_color(lv.palette_main(lv.PALETTE.BLUE))

style.set_img_recolor(lv.palette_main(lv.PALETTE.BLUE))
style.set_img_recolor_opa(lv.OPA._50)
# style.set_transform_angle(300)

# Create an object with the new style
obj = lv.img(lv.scr_act())
obj.add_style(style, 0)

obj.set_src(img_cogwheel_argb)

obj.center()
```

Text styles

```c
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_LABEL

/**
 * Using the text style properties
 */
void lv_example_style_8(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    lv_style_set_radius(&style, 5);
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 2));
    lv_style_set_border_width(&style, 2);
    lv_style_set_border_color(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_pad_all(&style, 10);

    lv_style_set_text_color(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_text_letter_space(&style, 5);
    lv_style_set_text_line_space(&style, 20);
    lv_style_set_text_decor(&style, LV_TEXT_DECOR_UNDERLINE);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_label_create(lv_scr_act());
    lv_obj_add_style(obj, &style, 0);
    lv_label_set_text(obj, "Text of\n"
                           "a label");

    lv_obj_center(obj);
}

#endif
```

```python
#
# Using the text style properties
#

style = lv.style_t()
style.init()

style.set_radius(5)
```

(下页继续)

(续上页)

```
style.set_bg_opa(lv.OPA.COVER)
style.set_bg_color(lv.palette_lighten(lv.PALETTE.GREY, 3))
style.set_border_width(2)
style.set_border_color(lv.palette_main(lv.PALETTE.BLUE))
style.set_pad_all(10)

style.set_text_color(lv.palette_main(lv.PALETTE.BLUE))
style.set_text_letter_space(5)
style.set_text_line_space(20)
style.set_text_decor(lv.TEXT_DECOR.UNDERLINE)

# Create an object with the new style
obj = lv.label(lv.scr_act())
obj.add_style(style, 0)
obj.set_text("Text of\n"
             "a label")

obj.center()
```

Line styles

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_LINE

/**
 * Using the line style properties
 */
void lv_example_style_9(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    lv_style_set_line_color(&style, lv_palette_main(LV_PALETTE_GREY));
    lv_style_set_line_width(&style, 6);
    lv_style_set_line_rounded(&style, true);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_line_create(lv_scr_act());
    lv_obj_add_style(obj, &style, 0);

    static lv_point_t p[] = {{10, 30}, {30, 50}, {100, 0}};
```

(下页继续)

(续上页)

```
    lv_line_set_points(obj, p, 3);

    lv_obj_center(obj);
}

#endif
```

```
#
# Using the line style properties
#

style = lv.style_t()
style.init()

style.set_line_color(lv.palette_main(lv.PALETTE.GREY))
style.set_line_width(6)
style.set_line_rounded(True)

# Create an object with the new style
obj = lv.line(lv.scr_act())
obj.add_style(style, 0)
p =  [ {"x":10, "y":30},
       {"x":30, "y":50},
       {"x":100, "y":0}]

obj.set_points(p, 3)

obj.center()
```

Transition

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_IMG

/**
 * Creating a transition
 */
void lv_example_style_10(void)
{
    static const lv_style_prop_t props[] = {LV_STYLE_BG_COLOR, LV_STYLE_BORDER_COLOR,␣
↪LV_STYLE_BORDER_WIDTH, 0};
```

(下页继续)

(续上页)

```
    /* A default transition
     * Make it fast (100ms) and start with some delay (200 ms)*/
    static lv_style_transition_dsc_t trans_def;
    lv_style_transition_dsc_init(&trans_def, props, lv_anim_path_linear, 100, 200,␣
↪NULL);

    /* A special transition when going to pressed state
     * Make it slow (500 ms) but start  without delay*/
    static lv_style_transition_dsc_t trans_pr;
    lv_style_transition_dsc_init(&trans_pr, props, lv_anim_path_linear, 500, 0, NULL);

    static lv_style_t style_def;
    lv_style_init(&style_def);
    lv_style_set_transition(&style_def, &trans_def);

    static lv_style_t style_pr;
    lv_style_init(&style_pr);
    lv_style_set_bg_color(&style_pr, lv_palette_main(LV_PALETTE_RED));
    lv_style_set_border_width(&style_pr, 6);
    lv_style_set_border_color(&style_pr, lv_palette_darken(LV_PALETTE_RED, 3));
    lv_style_set_transition(&style_pr, &trans_pr);

    /*Create an object with the new style_pr*/
    lv_obj_t * obj = lv_obj_create(lv_scr_act());
    lv_obj_add_style(obj, &style_def, 0);
    lv_obj_add_style(obj, &style_pr, LV_STATE_PRESSED);

    lv_obj_center(obj);
}

#endif
```

```
#
# Creating a transition
#

props = [lv.STYLE.BG_COLOR, lv.STYLE.BORDER_COLOR, lv.STYLE.BORDER_WIDTH, 0]

# A default transition
# Make it fast (100ms) and start with some delay (200 ms)

trans_def =  lv.style_transition_dsc_t()
trans_def.init(props, lv.anim_t.path_linear, 100, 200, None)
```

(下页继续)

(续上页)

```
# A special transition when going to pressed state
# Make it slow (500 ms) but start  without delay

trans_pr = lv.style_transition_dsc_t()
trans_pr.init(props, lv.anim_t.path_linear, 500, 0, None)

style_def = lv.style_t()
style_def.init()
style_def.set_transition(trans_def)

style_pr = lv.style_t()
style_pr.init()
style_pr.set_bg_color(lv.palette_main(lv.PALETTE.RED))
style_pr.set_border_width(6)
style_pr.set_border_color(lv.palette_darken(lv.PALETTE.RED, 3))
style_pr.set_transition(trans_pr)

# Create an object with the new style_pr
obj = lv.obj(lv.scr_act())
obj.add_style(style_def, 0)
obj.add_style(style_pr, lv.STATE.PRESSED)

obj.center()
```

Using multiple styles

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_IMG

/**
 * Using multiple styles
 */
void lv_example_style_11(void)
{
    /*A base style*/
    static lv_style_t style_base;
    lv_style_init(&style_base);
    lv_style_set_bg_color(&style_base, lv_palette_main(LV_PALETTE_LIGHT_BLUE));
    lv_style_set_border_color(&style_base, lv_palette_darken(LV_PALETTE_LIGHT_BLUE,␣
↪3));
    lv_style_set_border_width(&style_base, 2);
```

(下页继续)

(续上页)

```
    lv_style_set_radius(&style_base, 10);
    lv_style_set_shadow_width(&style_base, 10);
    lv_style_set_shadow_ofs_y(&style_base, 5);
    lv_style_set_shadow_opa(&style_base, LV_OPA_50);
    lv_style_set_text_color(&style_base, lv_color_white());
    lv_style_set_width(&style_base, 100);
    lv_style_set_height(&style_base, LV_SIZE_CONTENT);

    /*Set only the properties that should be different*/
    static lv_style_t style_warning;
    lv_style_init(&style_warning);
    lv_style_set_bg_color(&style_warning, lv_palette_main(LV_PALETTE_YELLOW));
    lv_style_set_border_color(&style_warning, lv_palette_darken(LV_PALETTE_YELLOW,␣
↪3));
    lv_style_set_text_color(&style_warning, lv_palette_darken(LV_PALETTE_YELLOW, 4));

    /*Create an object with the base style only*/
    lv_obj_t * obj_base = lv_obj_create(lv_scr_act());
    lv_obj_add_style(obj_base, &style_base, 0);
    lv_obj_align(obj_base, LV_ALIGN_LEFT_MID, 20, 0);

    lv_obj_t * label = lv_label_create(obj_base);
    lv_label_set_text(label, "Base");
    lv_obj_center(label);

    /*Create an other object with the base style and earnings style too*/
    lv_obj_t * obj_warning = lv_obj_create(lv_scr_act());
    lv_obj_add_style(obj_warning, &style_base, 0);
    lv_obj_add_style(obj_warning, &style_warning, 0);
    lv_obj_align(obj_warning, LV_ALIGN_RIGHT_MID, -20, 0);

    label = lv_label_create(obj_warning);
    lv_label_set_text(label, "Warning");
    lv_obj_center(label);
}

#endif
```

```
#
# Using multiple styles
#
# A base style
```

(下页继续)

(续上页)

```
style_base = lv.style_t()
style_base.init()
style_base.set_bg_color(lv.palette_main(lv.PALETTE.LIGHT_BLUE))
style_base.set_border_color(lv.palette_darken(lv.PALETTE.LIGHT_BLUE, 3))
style_base.set_border_width(2)
style_base.set_radius(10)
style_base.set_shadow_width(10)
style_base.set_shadow_ofs_y(5)
style_base.set_shadow_opa(lv.OPA._50)
style_base.set_text_color(lv.color_white())
style_base.set_width(100)
style_base.set_height(lv.SIZE.CONTENT)

# Set only the properties that should be different
style_warning = lv.style_t()
style_warning.init()
style_warning.set_bg_color(lv.palette_main(lv.PALETTE.YELLOW))
style_warning.set_border_color(lv.palette_darken(lv.PALETTE.YELLOW, 3))
style_warning.set_text_color(lv.palette_darken(lv.PALETTE.YELLOW, 4))

# Create an object with the base style only
obj_base = lv.obj(lv.scr_act())
obj_base.add_style(style_base, 0)
obj_base.align(lv.ALIGN.LEFT_MID, 20, 0)

label = lv.label(obj_base)
label.set_text("Base")
label.center()

# Create an other object with the base style and earnings style too
obj_warning = lv.obj(lv.scr_act())
obj_warning.add_style(style_base, 0)
obj_warning.add_style(style_warning, 0)
obj_warning.align(lv.ALIGN.RIGHT_MID, -20, 0)

label = lv.label(obj_warning)
label.set_text("Warning")
label.center()
```

Local styles

```c
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_IMG

/**
 * Local styles
 */
void lv_example_style_12(void)
{
    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_bg_color(&style, lv_palette_main(LV_PALETTE_GREEN));
    lv_style_set_border_color(&style, lv_palette_lighten(LV_PALETTE_GREEN, 3));
    lv_style_set_border_width(&style, 3);

    lv_obj_t * obj = lv_obj_create(lv_scr_act());
    lv_obj_add_style(obj, &style, 0);

    /*Overwrite the background color locally*/
    lv_obj_set_style_bg_color(obj,lv_palette_main(LV_PALETTE_ORANGE), LV_PART_MAIN);

    lv_obj_center(obj);
}

#endif
```

```python
#
# Local styles
#

style = lv.style_t()
style.init()
style.set_bg_color(lv.palette_main(lv.PALETTE.GREEN))
style.set_border_color(lv.palette_lighten(lv.PALETTE.GREEN, 3))
style.set_border_width(3)

obj = lv.obj(lv.scr_act())
obj.add_style(style, 0)

# Overwrite the background color locally
obj.set_style_bg_color(lv.palette_main(lv.PALETTE.ORANGE), lv.PART.MAIN)

obj.center()
```

Add styles to parts and states

```c
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_IMG

/**
 * Add styles to parts and states
 */
void lv_example_style_13(void)
{
    static lv_style_t style_indic;
    lv_style_init(&style_indic);
    lv_style_set_bg_color(&style_indic, lv_palette_lighten(LV_PALETTE_RED, 3));
    lv_style_set_bg_grad_color(&style_indic, lv_palette_main(LV_PALETTE_RED));
    lv_style_set_bg_grad_dir(&style_indic, LV_GRAD_DIR_HOR);

    static lv_style_t style_indic_pr;
    lv_style_init(&style_indic_pr);
    lv_style_set_shadow_color(&style_indic_pr, lv_palette_main(LV_PALETTE_RED));
    lv_style_set_shadow_width(&style_indic_pr, 10);
    lv_style_set_shadow_spread(&style_indic_pr, 3);

    /*Create an object with the new style_pr*/
    lv_obj_t * obj = lv_slider_create(lv_scr_act());
    lv_obj_add_style(obj, &style_indic, LV_PART_INDICATOR);
    lv_obj_add_style(obj, &style_indic_pr, LV_PART_INDICATOR | LV_STATE_PRESSED);
    lv_slider_set_value(obj, 70, LV_ANIM_OFF);
    lv_obj_center(obj);
}

#endif
```

```python
#
# Add styles to parts and states
#

style_indic = lv.style_t()
style_indic.init()
style_indic.set_bg_color(lv.palette_lighten(lv.PALETTE.RED, 3))
style_indic.set_bg_grad_color(lv.palette_main(lv.PALETTE.RED))
style_indic.set_bg_grad_dir(lv.GRAD_DIR.HOR)

style_indic_pr = lv.style_t()
style_indic_pr.init()
```

(下页继续)

(续上页)

```
style_indic_pr.set_shadow_color(lv.palette_main(lv.PALETTE.RED))
style_indic_pr.set_shadow_width(10)
style_indic_pr.set_shadow_spread(3)

# Create an object with the new style_pr
obj = lv.slider(lv.scr_act())
obj.add_style(style_indic, lv.PART.INDICATOR)
obj.add_style(style_indic_pr, lv.PART.INDICATOR | lv.STATE.PRESSED)
obj.set_value(70, lv.ANIM.OFF)
obj.center()
```

Extending the current theme

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_IMG


static lv_style_t style_btn;

/*Will be called when the styles of the base theme are already added
  to add new styles*/
static void new_theme_apply_cb(lv_theme_t * th, lv_obj_t * obj)
{
    LV_UNUSED(th);

    if(lv_obj_check_type(obj, &lv_btn_class)) {
        lv_obj_add_style(obj, &style_btn, 0);
    }
}

static void new_theme_init_and_set(void)
{
    /*Initialize the styles*/
    lv_style_init(&style_btn);
    lv_style_set_bg_color(&style_btn, lv_palette_main(LV_PALETTE_GREEN));
    lv_style_set_border_color(&style_btn, lv_palette_darken(LV_PALETTE_GREEN, 3));
    lv_style_set_border_width(&style_btn, 3);

    /*Initialize the new theme from the current theme*/
    lv_theme_t * th_act = lv_disp_get_theme(NULL);
    static lv_theme_t th_new;
    th_new = *th_act;
```

(下页继续)

(续上页)

```
    /*Set the parent theme and the style apply callback for the new theme*/
    lv_theme_set_parent(&th_new, th_act);
    lv_theme_set_apply_cb(&th_new, new_theme_apply_cb);

    /*Assign the new theme the the current display*/
    lv_disp_set_theme(NULL, &th_new);
}



/**
 * Extending the current theme
 */
void lv_example_style_14(void)
{
    lv_obj_t * btn;
    lv_obj_t * label;

    btn = lv_btn_create(lv_scr_act());
    lv_obj_align(btn, LV_ALIGN_TOP_MID, 0, 20);

    label = lv_label_create(btn);
    lv_label_set_text(label, "Original theme");

    new_theme_init_and_set();

    btn = lv_btn_create(lv_scr_act());
    lv_obj_align(btn, LV_ALIGN_BOTTOM_MID, 0, -20);

    label = lv_label_create(btn);
    lv_label_set_text(label, "New theme");
}

#endif
```

```
# Will be called when the styles of the base theme are already added
# to add new styles


class NewTheme(lv.theme_t):
    def __init__(self):
        super().__init__()
```

(下页继续)

(续上页)

```
        # Initialize the styles
        self.style_btn = lv.style_t()
        self.style_btn.init()
        self.style_btn.set_bg_color(lv.palette_main(lv.PALETTE.GREEN))
        self.style_btn.set_border_color(lv.palette_darken(lv.PALETTE.GREEN, 3))
        self.style_btn.set_border_width(3)

        # This theme is based on active theme
        th_act = lv.theme_get_from_obj(lv.scr_act())
        # This theme will be applied only after base theme is applied
        self.set_parent(th_act)


class ExampleStyle_14:

    def __init__(self):
        #
        # Extending the current theme
        #

        btn = lv.btn(lv.scr_act())
        btn.align(lv.ALIGN.TOP_MID, 0, 20)

        label = lv.label(btn)
        label.set_text("Original theme")

        self.new_theme_init_and_set()

        btn = lv.btn(lv.scr_act())
        btn.align(lv.ALIGN.BOTTOM_MID, 0, -20)

        label = lv.label(btn)
        label.set_text("New theme")

    def new_theme_apply_cb(self, th, obj):
        print(th,obj)
        if obj.get_class() == lv.btn_class:
            obj.add_style(self.th_new.style_btn, 0)

    def new_theme_init_and_set(self):
        print("new_theme_init_and_set")
        # Initialize the new theme from the current theme
        self.th_new = NewTheme()
```

(下页继续)

(续上页)

```
        self.th_new.set_apply_cb(self.new_theme_apply_cb)
        lv.disp_get_default().set_theme(self.th_new)


exampleStyle_14 = ExampleStyle_14()
```

2.2.3 Animations

Start animation on an event

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_SWITCH

static void anim_x_cb(void * var, int32_t v)
{
    lv_obj_set_x(var, v);
}

static void sw_event_cb(lv_event_t * e)
{
    lv_obj_t * sw = lv_event_get_target(e);
    lv_obj_t * label = lv_event_get_user_data(e);

    if(lv_obj_has_state(sw, LV_STATE_CHECKED)) {
        lv_anim_t a;
        lv_anim_init(&a);
        lv_anim_set_var(&a, label);
        lv_anim_set_values(&a, lv_obj_get_x(label), 100);
        lv_anim_set_time(&a, 500);
        lv_anim_set_exec_cb(&a, anim_x_cb);
        lv_anim_set_path_cb(&a, lv_anim_path_overshoot);
        lv_anim_start(&a);
    } else {
        lv_anim_t a;
        lv_anim_init(&a);
        lv_anim_set_var(&a, label);
        lv_anim_set_values(&a, lv_obj_get_x(label), -lv_obj_get_width(label));
        lv_anim_set_time(&a, 500);
        lv_anim_set_exec_cb(&a, anim_x_cb);
        lv_anim_set_path_cb(&a, lv_anim_path_ease_in);
        lv_anim_start(&a);
    }
```

(下页继续)

(续上页)

```
}

/**
 * Start animation on an event
 */
void lv_example_anim_1(void)
{
    lv_obj_t * label = lv_label_create(lv_scr_act());
    lv_label_set_text(label, "Hello animations!");
    lv_obj_set_pos(label, 100, 10);


    lv_obj_t * sw = lv_switch_create(lv_scr_act());
    lv_obj_center(sw);
    lv_obj_add_state(sw, LV_STATE_CHECKED);
    lv_obj_add_event_cb(sw, sw_event_cb, LV_EVENT_VALUE_CHANGED, label);
}

#endif
```

```
def anim_x_cb(label, v):
    label.set_x(v)

def sw_event_cb(e,label):
    sw = e.get_target()

    if sw.has_state(lv.STATE.CHECKED):
        a = lv.anim_t()
        a.init()
        a.set_var(label)
        a.set_values(label.get_x(), 100)
        a.set_time(500)
        a.set_path_cb(lv.anim_t.path_overshoot)
        a.set_custom_exec_cb(lambda a,val: anim_x_cb(label,val))
        lv.anim_t.start(a)
    else:
        a = lv.anim_t()
        a.init()
        a.set_var(label)
        a.set_values(label.get_x(), -label.get_width())
        a.set_time(500)
        a.set_path_cb(lv.anim_t.path_ease_in)
```

(下页继续)

(续上页)

```
        a.set_custom_exec_cb(lambda a,val: anim_x_cb(label,val))
        lv.anim_t.start(a)

#
# Start animation on an event
#

label = lv.label(lv.scr_act())
label.set_text("Hello animations!")
label.set_pos(100, 10)


sw = lv.switch(lv.scr_act())
sw.center()
sw.add_state(lv.STATE.CHECKED)
sw.add_event_cb(lambda e: sw_event_cb(e,label), lv.EVENT.VALUE_CHANGED, None)
```

Playback animation

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_SWITCH


static void anim_x_cb(void * var, int32_t v)
{
    lv_obj_set_x(var, v);
}

static void anim_size_cb(void * var, int32_t v)
{
    lv_obj_set_size(var, v, v);
}

/**
 * Create a playback animation
 */
void lv_example_anim_2(void)
{
```

(下页继续)

(续上页)

```
    lv_obj_t * obj = lv_obj_create(lv_scr_act());
    lv_obj_set_style_bg_color(obj, lv_palette_main(LV_PALETTE_RED), 0);
    lv_obj_set_style_radius(obj, LV_RADIUS_CIRCLE, 0);

    lv_obj_align(obj, LV_ALIGN_LEFT_MID, 10, 0);

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_var(&a, obj);
    lv_anim_set_values(&a, 10, 50);
    lv_anim_set_time(&a, 1000);
    lv_anim_set_playback_delay(&a, 100);
    lv_anim_set_playback_time(&a, 300);
    lv_anim_set_repeat_delay(&a, 500);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
    lv_anim_set_path_cb(&a, lv_anim_path_ease_in_out);

    lv_anim_set_exec_cb(&a, anim_size_cb);
    lv_anim_start(&a);
    lv_anim_set_exec_cb(&a, anim_x_cb);
    lv_anim_set_values(&a, 10, 240);
    lv_anim_start(&a);
}

#endif
```

```
def anim_x_cb(obj, v):
    obj.set_x(v)

def anim_size_cb(obj, v):
    obj.set_size(v, v)



#
# Create a playback animation
#
obj = lv.obj(lv.scr_act())
obj.set_style_bg_color(lv.palette_main(lv.PALETTE.RED), 0)
obj.set_style_radius(lv.RADIUS.CIRCLE, 0)

obj.align(lv.ALIGN.LEFT_MID, 10, 0)

a1 = lv.anim_t()
```

(下页继续)

(续上页)

```
a1.init()
a1.set_var(obj)
a1.set_values(10, 50)
a1.set_time(1000)
a1.set_playback_delay(100)
a1.set_playback_time(300)
a1.set_repeat_delay(500)
a1.set_repeat_count(lv.ANIM_REPEAT.INFINITE)
a1.set_path_cb(lv.anim_t.path_ease_in_out)
a1.set_custom_exec_cb(lambda a1,val: anim_size_cb(obj,val))
lv.anim_t.start(a1)

a2 = lv.anim_t()
a2.init()
a2.set_var(obj)
a2.set_values(10, 240)
a2.set_time(1000)
a2.set_playback_delay(100)
a2.set_playback_time(300)
a2.set_repeat_delay(500)
a2.set_repeat_count(lv.ANIM_REPEAT.INFINITE)
a2.set_path_cb(lv.anim_t.path_ease_in_out)
a2.set_custom_exec_cb(lambda a1,val: anim_x_cb(obj,val))
lv.anim_t.start(a2)
```

Animation timeline

```
#include "../lv_examples.h"
#if LV_USE_FLEX && LV_BUILD_EXAMPLES

static lv_anim_timeline_t * anim_timeline = NULL;

static lv_obj_t * obj1 = NULL;
static lv_obj_t * obj2 = NULL;
static lv_obj_t * obj3 = NULL;

static const lv_coord_t obj_width = 90;
static const lv_coord_t obj_height = 70;

static void set_width(void * var, int32_t v)
{
    lv_obj_set_width((lv_obj_t *)var, v);
```

(下页继续)

(续上页)

```
}

static void set_height(void * var, int32_t v)
{
    lv_obj_set_height((lv_obj_t *)var, v);
}

static void anim_timeline_create(void)
{
    /* obj1 */
    lv_anim_t a1;
    lv_anim_init(&a1);
    lv_anim_set_var(&a1, obj1);
    lv_anim_set_values(&a1, 0, obj_width);
    lv_anim_set_early_apply(&a1, false);
    lv_anim_set_exec_cb(&a1, (lv_anim_exec_xcb_t)set_width);
    lv_anim_set_path_cb(&a1, lv_anim_path_overshoot);
    lv_anim_set_time(&a1, 300);

    lv_anim_t a2;
    lv_anim_init(&a2);
    lv_anim_set_var(&a2, obj1);
    lv_anim_set_values(&a2, 0, obj_height);
    lv_anim_set_early_apply(&a2, false);
    lv_anim_set_exec_cb(&a2, (lv_anim_exec_xcb_t)set_height);
    lv_anim_set_path_cb(&a2, lv_anim_path_ease_out);
    lv_anim_set_time(&a2, 300);

    /* obj2 */
    lv_anim_t a3;
    lv_anim_init(&a3);
    lv_anim_set_var(&a3, obj2);
    lv_anim_set_values(&a3, 0, obj_width);
    lv_anim_set_early_apply(&a3, false);
    lv_anim_set_exec_cb(&a3, (lv_anim_exec_xcb_t)set_width);
    lv_anim_set_path_cb(&a3, lv_anim_path_overshoot);
    lv_anim_set_time(&a3, 300);

    lv_anim_t a4;
    lv_anim_init(&a4);
    lv_anim_set_var(&a4, obj2);
    lv_anim_set_values(&a4, 0, obj_height);
    lv_anim_set_early_apply(&a4, false);
```

(下页继续)

(续上页)

```
    lv_anim_set_exec_cb(&a4, (lv_anim_exec_xcb_t)set_height);
    lv_anim_set_path_cb(&a4, lv_anim_path_ease_out);
    lv_anim_set_time(&a4, 300);

    /* obj3 */
    lv_anim_t a5;
    lv_anim_init(&a5);
    lv_anim_set_var(&a5, obj3);
    lv_anim_set_values(&a5, 0, obj_width);
    lv_anim_set_early_apply(&a5, false);
    lv_anim_set_exec_cb(&a5, (lv_anim_exec_xcb_t)set_width);
    lv_anim_set_path_cb(&a5, lv_anim_path_overshoot);
    lv_anim_set_time(&a5, 300);

    lv_anim_t a6;
    lv_anim_init(&a6);
    lv_anim_set_var(&a6, obj3);
    lv_anim_set_values(&a6, 0, obj_height);
    lv_anim_set_early_apply(&a6, false);
    lv_anim_set_exec_cb(&a6, (lv_anim_exec_xcb_t)set_height);
    lv_anim_set_path_cb(&a6, lv_anim_path_ease_out);
    lv_anim_set_time(&a6, 300);

    /* Create anim timeline */
    anim_timeline = lv_anim_timeline_create();
    lv_anim_timeline_add(anim_timeline, 0, &a1);
    lv_anim_timeline_add(anim_timeline, 0, &a2);
    lv_anim_timeline_add(anim_timeline, 200, &a3);
    lv_anim_timeline_add(anim_timeline, 200, &a4);
    lv_anim_timeline_add(anim_timeline, 400, &a5);
    lv_anim_timeline_add(anim_timeline, 400, &a6);
}

static void btn_start_event_handler(lv_event_t * e)
{
    lv_obj_t * btn = lv_event_get_target(e);

    if (!anim_timeline) {
        anim_timeline_create();
    }

    bool reverse = lv_obj_has_state(btn, LV_STATE_CHECKED);
    lv_anim_timeline_set_reverse(anim_timeline, reverse);
```

(下页继续)

(续上页)

```
    lv_anim_timeline_start(anim_timeline);
}

static void btn_del_event_handler(lv_event_t * e)
{
    LV_UNUSED(e);
    if (anim_timeline) {
        lv_anim_timeline_del(anim_timeline);
        anim_timeline = NULL;
    }
}

static void btn_stop_event_handler(lv_event_t * e)
{
    LV_UNUSED(e);
    if (anim_timeline) {
        lv_anim_timeline_stop(anim_timeline);
    }
}

static void slider_prg_event_handler(lv_event_t * e)
{
    lv_obj_t * slider = lv_event_get_target(e);

    if (!anim_timeline) {
        anim_timeline_create();
    }

    int32_t progress = lv_slider_get_value(slider);
    lv_anim_timeline_set_progress(anim_timeline, progress);
}

/**
 * Create an animation timeline
 */
void lv_example_anim_timeline_1(void)
{
    lv_obj_t * par = lv_scr_act();
    lv_obj_set_flex_flow(par, LV_FLEX_FLOW_ROW);
    lv_obj_set_flex_align(par, LV_FLEX_ALIGN_SPACE_AROUND, LV_FLEX_ALIGN_CENTER, LV_
↪FLEX_ALIGN_CENTER);

    /* create btn_start */
```

(下页继续)

(续上页)

```
    lv_obj_t * btn_start = lv_btn_create(par);
    lv_obj_add_event_cb(btn_start, btn_start_event_handler, LV_EVENT_VALUE_CHANGED,␣
↪NULL);
    lv_obj_add_flag(btn_start, LV_OBJ_FLAG_IGNORE_LAYOUT);
    lv_obj_add_flag(btn_start, LV_OBJ_FLAG_CHECKABLE);
    lv_obj_align(btn_start, LV_ALIGN_TOP_MID, -100, 20);

    lv_obj_t * label_start = lv_label_create(btn_start);
    lv_label_set_text(label_start, "Start");
    lv_obj_center(label_start);

    /* create btn_del */
    lv_obj_t * btn_del = lv_btn_create(par);
    lv_obj_add_event_cb(btn_del, btn_del_event_handler, LV_EVENT_CLICKED, NULL);
    lv_obj_add_flag(btn_del, LV_OBJ_FLAG_IGNORE_LAYOUT);
    lv_obj_align(btn_del, LV_ALIGN_TOP_MID, 0, 20);

    lv_obj_t * label_del = lv_label_create(btn_del);
    lv_label_set_text(label_del, "Delete");
    lv_obj_center(label_del);

    /* create btn_stop */
    lv_obj_t * btn_stop = lv_btn_create(par);
    lv_obj_add_event_cb(btn_stop, btn_stop_event_handler, LV_EVENT_CLICKED, NULL);
    lv_obj_add_flag(btn_stop, LV_OBJ_FLAG_IGNORE_LAYOUT);
    lv_obj_align(btn_stop, LV_ALIGN_TOP_MID, 100, 20);

    lv_obj_t * label_stop = lv_label_create(btn_stop);
    lv_label_set_text(label_stop, "Stop");
    lv_obj_center(label_stop);

    /* create slider_prg */
    lv_obj_t * slider_prg = lv_slider_create(par);
    lv_obj_add_event_cb(slider_prg, slider_prg_event_handler, LV_EVENT_VALUE_CHANGED,␣
↪NULL);
    lv_obj_add_flag(slider_prg, LV_OBJ_FLAG_IGNORE_LAYOUT);
    lv_obj_align(slider_prg, LV_ALIGN_BOTTOM_MID, 0, -20);
    lv_slider_set_range(slider_prg, 0, 65535);

    /* create 3 objects */
    obj1 = lv_obj_create(par);
    lv_obj_set_size(obj1, obj_width, obj_height);
```

(下页继续)

(续上页)

```
    obj2 = lv_obj_create(par);
    lv_obj_set_size(obj2, obj_width, obj_height);

    obj3 = lv_obj_create(par);
    lv_obj_set_size(obj3, obj_width, obj_height);
}

#endif
```

```
class LV_ExampleAnimTimeline_1(object):

    def __init__(self):
        self.obj_width = 120
        self.obj_height = 150
        #
        # Create an animation timeline
        #

        self.par = lv.scr_act()
        self.par.set_flex_flow(lv.FLEX_FLOW.ROW)
        self.par.set_flex_align(lv.FLEX_ALIGN.SPACE_AROUND, lv.FLEX_ALIGN.CENTER, lv.
↪FLEX_ALIGN.CENTER)

        self.btn_run = lv.btn(self.par)
        self.btn_run.add_event_cb(self.btn_run_event_handler, lv.EVENT.VALUE_CHANGED,␣
↪None)
        self.btn_run.add_flag(lv.obj.FLAG.IGNORE_LAYOUT)
        self.btn_run.add_flag(lv.obj.FLAG.CHECKABLE)
        self.btn_run.align(lv.ALIGN.TOP_MID, -50, 20)

        self.label_run = lv.label(self.btn_run)
        self.label_run.set_text("Run")
        self.label_run.center()

        self.btn_del = lv.btn(self.par)
        self.btn_del.add_event_cb(self.btn_del_event_handler, lv.EVENT.CLICKED, None)
        self.btn_del.add_flag(lv.obj.FLAG.IGNORE_LAYOUT)
        self.btn_del.align(lv.ALIGN.TOP_MID, 50, 20)

        self.label_del = lv.label(self.btn_del)
        self.label_del.set_text("Stop")
        self.label_del.center()
```

(下页继续)

(续上页)

```
        self.slider = lv.slider(self.par)
        self.slider.add_event_cb(self.slider_prg_event_handler, lv.EVENT.VALUE_
↪CHANGED, None)
        self.slider.add_flag(lv.obj.FLAG.IGNORE_LAYOUT)
        self.slider.align(lv.ALIGN.BOTTOM_RIGHT, -20, -20)
        self.slider.set_range(0, 65535)

        self.obj1 = lv.obj(self.par)
        self.obj1.set_size(self.obj_width, self.obj_height)

        self.obj2 = lv.obj(self.par)
        self.obj2.set_size(self.obj_width, self.obj_height)

        self.obj3 = lv.obj(self.par)
        self.obj3.set_size(self.obj_width, self.obj_height)

        self.anim_timeline = None

    def set_width(self,obj, v):
        obj.set_width(v)

    def set_height(self,obj, v):
        obj.set_height(v)

    def anim_timeline_create(self):
        # obj1
        self.a1 = lv.anim_t()
        self.a1.init()
        self.a1.set_values(0, self.obj_width)
        self.a1.set_early_apply(False)
        self.a1.set_custom_exec_cb(lambda a,v: self.set_width(self.obj1,v))
        self.a1.set_path_cb(lv.anim_t.path_overshoot)
        self.a1.set_time(300)

        self.a2 = lv.anim_t()
        self.a2.init()
        self.a2.set_values(0, self.obj_height)
        self.a2.set_early_apply(False)
        self.a2.set_custom_exec_cb(lambda a,v: self.set_height(self.obj1,v))
        self.a2.set_path_cb(lv.anim_t.path_ease_out)
        self.a2.set_time(300)

        # obj2
```

(下页继续)

(续上页)

```
        self.a3=lv.anim_t()
        self.a3.init()
        self.a3.set_values(0, self.obj_width)
        self.a3.set_early_apply(False)
        self.a3.set_custom_exec_cb(lambda a,v: self.set_width(self.obj2,v))
        self.a3.set_path_cb(lv.anim_t.path_overshoot)
        self.a3.set_time(300)

        self.a4 = lv.anim_t()
        self.a4.init()
        self.a4.set_values(0, self.obj_height)
        self.a4.set_early_apply(False)
        self.a4.set_custom_exec_cb(lambda a,v: self.set_height(self.obj2,v))
        self.a4.set_path_cb(lv.anim_t.path_ease_out)
        self.a4.set_time(300)

        # obj3
        self.a5 = lv.anim_t()
        self.a5.init()
        self.a5.set_values(0, self.obj_width)
        self.a5.set_early_apply(False)
        self.a5.set_custom_exec_cb(lambda a,v: self.set_width(self.obj3,v))
        self.a5.set_path_cb(lv.anim_t.path_overshoot)
        self.a5.set_time(300)

        self.a6 = lv.anim_t()
        self.a6.init()
        self.a6.set_values(0, self.obj_height)
        self.a6.set_early_apply(False)
        self.a6.set_custom_exec_cb(lambda a,v: self.set_height(self.obj3,v))
        self.a6.set_path_cb(lv.anim_t.path_ease_out)
        self.a6.set_time(300)

        # Create anim timeline
        print("Create new anim_timeline")
        self.anim_timeline = lv.anim_timeline_create()
        lv.anim_timeline_add(self.anim_timeline, 0, self.a1)
        lv.anim_timeline_add(self.anim_timeline, 0, self.a2)
        lv.anim_timeline_add(self.anim_timeline, 200, self.a3)
        lv.anim_timeline_add(self.anim_timeline, 200, self.a4)
        lv.anim_timeline_add(self.anim_timeline, 400, self.a5)
        lv.anim_timeline_add(self.anim_timeline, 400, self.a6)
```

(下页继续)

(续上页)

```
    def slider_prg_event_handler(self,e):
        slider = e.get_target()

        if  not self.anim_timeline:
            self.anim_timeline_create()

        progress = slider.get_value()
        lv.anim_timeline_set_progress(self.anim_timeline, progress)


    def btn_run_event_handler(self,e):
        btn = e.get_target()
        if not self.anim_timeline:
            self.anim_timeline_create()

        reverse = btn.has_state(lv.STATE.CHECKED)
        lv.anim_timeline_set_reverse(self.anim_timeline,reverse)
        lv.anim_timeline_start(self.anim_timeline)

    def btn_del_event_handler(self,e):
        if self.anim_timeline:
            lv.anim_timeline_del(self.anim_timeline)
        self.anim_timeline = None


lv_example_anim_timeline_1 = LV_ExampleAnimTimeline_1()
```

2.2.4 Events

Button click event

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_SWITCH

static void event_cb(lv_event_t * e)
{
    LV_LOG_USER("Clicked");

    static uint32_t cnt = 1;
    lv_obj_t * btn = lv_event_get_target(e);
    lv_obj_t * label = lv_obj_get_child(btn, 0);
    lv_label_set_text_fmt(label, "%"LV_PRIu32, cnt);
```

(下页继续)

(续上页)

```
    cnt++;
}

/**
 * Add click event to a button
 */
void lv_example_event_1(void)
{
    lv_obj_t * btn = lv_btn_create(lv_scr_act());
    lv_obj_set_size(btn, 100, 50);
    lv_obj_center(btn);
    lv_obj_add_event_cb(btn, event_cb, LV_EVENT_CLICKED, NULL);

    lv_obj_t * label = lv_label_create(btn);
    lv_label_set_text(label, "Click me!");
    lv_obj_center(label);
}

#endif
```

```
class Event_1():
    def __init__(self):
        self.cnt = 1
        #
        # Add click event to a button
        #

        btn = lv.btn(lv.scr_act())
        btn.set_size(100, 50)
        btn.center()
        btn.add_event_cb(self.event_cb, lv.EVENT.CLICKED, None)

        label = lv.label(btn)
        label.set_text("Click me!")
        label.center()

    def event_cb(self,e):
        print("Clicked")

        btn = e.get_target()
        label = btn.get_child(0)
        label.set_text(str(self.cnt))
        self.cnt += 1
```

(下页继续)

(续上页)

```
evt1 = Event_1()
```

Handle multiple events

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_SWITCH

static void event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * label = lv_event_get_user_data(e);

    switch(code) {
    case LV_EVENT_PRESSED:
        lv_label_set_text(label, "The last button event:\nLV_EVENT_PRESSED");
        break;
    case LV_EVENT_CLICKED:
        lv_label_set_text(label, "The last button event:\nLV_EVENT_CLICKED");
        break;
    case LV_EVENT_LONG_PRESSED:
        lv_label_set_text(label, "The last button event:\nLV_EVENT_LONG_PRESSED");
        break;
    case LV_EVENT_LONG_PRESSED_REPEAT:
        lv_label_set_text(label, "The last button event:\nLV_EVENT_LONG_PRESSED_REPEAT
↪");
        break;
    default:
        break;
    }
}

/**
 * Handle multiple events
 */
void lv_example_event_2(void)
{
    lv_obj_t * btn = lv_btn_create(lv_scr_act());
    lv_obj_set_size(btn, 100, 50);
    lv_obj_center(btn);

    lv_obj_t * btn_label = lv_label_create(btn);
```

(下页继续)

(续上页)

```
    lv_label_set_text(btn_label, "Click me!");
    lv_obj_center(btn_label);

    lv_obj_t * info_label = lv_label_create(lv_scr_act());
    lv_label_set_text(info_label, "The last button event:\nNone");

    lv_obj_add_event_cb(btn, event_cb, LV_EVENT_ALL, info_label);
}

#endif
```

```
def event_cb(e,label):
    code = e.get_code()
    if code == lv.EVENT.PRESSED:
        label.set_text("The last button event:\nLV_EVENT_PRESSED")
    elif code == lv.EVENT.CLICKED:
        label.set_text("The last button event:\nLV_EVENT_CLICKED")
    elif code ==  lv.EVENT.LONG_PRESSED:
        label.set_text("The last button event:\nLV_EVENT_LONG_PRESSED")
    elif code == lv.EVENT.LONG_PRESSED_REPEAT:
        label.set_text("The last button event:\nLV_EVENT_LONG_PRESSED_REPEAT")
btn = lv.btn(lv.scr_act())
btn.set_size(100, 50)
btn.center()

btn_label = lv.label(btn)
btn_label.set_text("Click me!")
btn_label.center()

info_label = lv.label(lv.scr_act())
info_label.set_text("The last button event:\nNone")

btn.add_event_cb(lambda e: event_cb(e,info_label), lv.EVENT.ALL, None)
```

Event bubbling

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_FLEX

static void event_cb(lv_event_t * e)
{
    /*The original target of the event. Can be the buttons or the container*/
```

(下页继续)

(续上页)

```
    lv_obj_t * target = lv_event_get_target(e);

    /*The current target is always the container as the event is added to it*/
    lv_obj_t * cont = lv_event_get_current_target(e);

    /*If container was clicked do nothing*/
    if(target == cont) return;

    /*Make the clicked buttons red*/
    lv_obj_set_style_bg_color(target, lv_palette_main(LV_PALETTE_RED), 0);
}

/**
 * Demonstrate event bubbling
 */
void lv_example_event_3(void)
{

    lv_obj_t * cont = lv_obj_create(lv_scr_act());
    lv_obj_set_size(cont, 290, 200);
    lv_obj_center(cont);
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_ROW_WRAP);

    uint32_t i;
    for(i = 0; i < 30; i++) {
        lv_obj_t * btn = lv_btn_create(cont);
        lv_obj_set_size(btn, 80, 50);
        lv_obj_add_flag(btn, LV_OBJ_FLAG_EVENT_BUBBLE);

        lv_obj_t * label = lv_label_create(btn);
        lv_label_set_text_fmt(label, "%"LV_PRIu32, i);
        lv_obj_center(label);
    }

    lv_obj_add_event_cb(cont, event_cb, LV_EVENT_CLICKED, NULL);
}

#endif
```

```
def event_cb(e):

    # The original target of the event. Can be the buttons or the container
    target = e.get_target()
```

(下页继续)

(续上页)

```
    # print(type(target))

    # If container was clicked do nothing
    if type(target) != type(lv.btn()):
        return

    # Make the clicked buttons red
    target.set_style_bg_color(lv.palette_main(lv.PALETTE.RED), 0)

#
# Demonstrate event bubbling
#

cont = lv.obj(lv.scr_act())
cont.set_size(320, 200)
cont.center()
cont.set_flex_flow(lv.FLEX_FLOW.ROW_WRAP)

for i in range(30):
    btn = lv.btn(cont)
    btn.set_size(80, 50)
    btn.add_flag(lv.obj.FLAG.EVENT_BUBBLE)

    label = lv.label(btn)
    label.set_text(str(i))
    label.center()
    cont.add_event_cb(event_cb, lv.EVENT.CLICKED, None)
```

2.2.5 Layouts

Flex

A simple row and a column layout with flexbox

```
#include "../../lv_examples.h"
#if LV_USE_FLEX && LV_BUILD_EXAMPLES

/**
 * A simple row and a column layout with flexbox
 */
void lv_example_flex_1(void)
{
```

(下页继续)

(续上页)

```
    /*Create a container with ROW flex direction*/
    lv_obj_t * cont_row = lv_obj_create(lv_scr_act());
    lv_obj_set_size(cont_row, 300, 75);
    lv_obj_align(cont_row, LV_ALIGN_TOP_MID, 0, 5);
    lv_obj_set_flex_flow(cont_row, LV_FLEX_FLOW_ROW);

    /*Create a container with COLUMN flex direction*/
    lv_obj_t * cont_col = lv_obj_create(lv_scr_act());
    lv_obj_set_size(cont_col, 200, 150);
    lv_obj_align_to(cont_col, cont_row, LV_ALIGN_OUT_BOTTOM_MID, 0, 5);
    lv_obj_set_flex_flow(cont_col, LV_FLEX_FLOW_COLUMN);

    uint32_t i;
    for(i = 0; i < 10; i++) {
        lv_obj_t * obj;
        lv_obj_t * label;

        /*Add items to the row*/
        obj= lv_btn_create(cont_row);
        lv_obj_set_size(obj, 100, LV_PCT(100));

        label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "Item: %u", i);
        lv_obj_center(label);

        /*Add items to the column*/
        obj = lv_btn_create(cont_col);
        lv_obj_set_size(obj, LV_PCT(100), LV_SIZE_CONTENT);

        label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "Item: %"LV_PRIu32, i);
        lv_obj_center(label);
    }
}

#endif
```

```
#
# A simple row and a column layout with flexbox
#

# Create a container with ROW flex direction
cont_row = lv.obj(lv.scr_act())
```

(下页继续)

(续上页)

```
cont_row.set_size(300, 75)
cont_row.align(lv.ALIGN.TOP_MID, 0, 5)
cont_row.set_flex_flow(lv.FLEX_FLOW.ROW)

# Create a container with COLUMN flex direction
cont_col = lv.obj(lv.scr_act())
cont_col.set_size(200, 150)
cont_col.align_to(cont_row, lv.ALIGN.OUT_BOTTOM_MID, 0, 5)
cont_col.set_flex_flow(lv.FLEX_FLOW.COLUMN)

for i in range(10):
    # Add items to the row
    obj = lv.btn(cont_row)
    obj.set_size(100, lv.pct(100))

    label = lv.label(obj)
    label.set_text("Item: {:d}".format(i))
    label.center()

    # Add items to the column
    obj = lv.btn(cont_col)
    obj.set_size(lv.pct(100), lv.SIZE.CONTENT)

    label = lv.label(obj)
    label.set_text("Item: {:d}".format(i))
    label.center()
```

Arrange items in rows with wrap and even spacing

```
#include "../../lv_examples.h"
#if LV_USE_FLEX && LV_BUILD_EXAMPLES

/**
 * Arrange items in rows with wrap and place the items to get even space around them.
 */
void lv_example_flex_2(void)
{
    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_flex_flow(&style, LV_FLEX_FLOW_ROW_WRAP);
    lv_style_set_flex_main_place(&style, LV_FLEX_ALIGN_SPACE_EVENLY);
```

(下页继续)

(续上页)

```
    lv_style_set_layout(&style, LV_LAYOUT_FLEX);

    lv_obj_t * cont = lv_obj_create(lv_scr_act());
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_add_style(cont, &style, 0);

    uint32_t i;
    for(i = 0; i < 8; i++) {
        lv_obj_t * obj = lv_obj_create(cont);
        lv_obj_set_size(obj, 70, LV_SIZE_CONTENT);
        lv_obj_add_flag(obj, LV_OBJ_FLAG_CHECKABLE);

        lv_obj_t * label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "%"LV_PRIu32, i);
        lv_obj_center(label);
    }
}

#endif
```

```
#
# Arrange items in rows with wrap and place the items to get even space around them.
#
style = lv.style_t()
style.init()
style.set_flex_flow(lv.FLEX_FLOW.ROW_WRAP)
style.set_flex_main_place(lv.FLEX_ALIGN.SPACE_EVENLY)
style.set_layout(lv.LAYOUT_FLEX.value)

cont = lv.obj(lv.scr_act())
cont.set_size(300, 220)
cont.center()
cont.add_style(style, 0)

for i in range(8):
    obj = lv.obj(cont)
    obj.set_size(70, lv.SIZE.CONTENT)

    label = lv.label(obj)
    label.set_text("{:d}".format(i))
    label.center()
```

Demonstrate flex grow

```c
#include "../../lv_examples.h"
#if LV_USE_FLEX && LV_BUILD_EXAMPLES

/**
 * Demonstrate flex grow.
 */
void lv_example_flex_3(void)
{
    lv_obj_t * cont = lv_obj_create(lv_scr_act());
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_ROW);

    lv_obj_t * obj;
    obj = lv_obj_create(cont);
    lv_obj_set_size(obj, 40, 40);           /*Fix size*/

    obj = lv_obj_create(cont);
    lv_obj_set_height(obj, 40);
    lv_obj_set_flex_grow(obj, 1);           /*1 portion from the free space*/

    obj = lv_obj_create(cont);
    lv_obj_set_height(obj, 40);
    lv_obj_set_flex_grow(obj, 2);           /*2 portion from the free space*/

    obj = lv_obj_create(cont);
    lv_obj_set_size(obj, 40, 40);           /*Fix size. It is flushed to the right by␣
↪the "grow" items*/
}

#endif
```

```python
#
# Demonstrate flex grow.
#

cont = lv.obj(lv.scr_act())
cont.set_size(300, 220)
cont.center()
cont.set_flex_flow(lv.FLEX_FLOW.ROW)

obj = lv.obj(cont)
```

(下页继续)

(续上页)

```
obj.set_size(40, 40)           # Fix size

obj = lv.obj(cont)
obj.set_height(40)
obj.set_flex_grow(1)           # 1 portion from the free space

obj = lv.obj(cont)
obj.set_height(40)
obj.set_flex_grow(2)           # 2 portion from the free space

obj = lv.obj(cont)
obj.set_size(40, 40)           # Fix size. It is flushed to the right by the "grow"␣
↪items
```

Demonstrate flex grow.

```
#include "../../lv_examples.h"
#if LV_USE_FLEX && LV_BUILD_EXAMPLES

/**
 * Reverse the order of flex items
 */
void lv_example_flex_4(void)
{

    lv_obj_t * cont = lv_obj_create(lv_scr_act());
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_COLUMN_REVERSE);

    uint32_t i;
    for(i = 0; i < 6; i++) {
        lv_obj_t * obj = lv_obj_create(cont);
        lv_obj_set_size(obj, 100, 50);

        lv_obj_t * label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "Item: %"LV_PRIu32, i);
        lv_obj_center(label);
    }
}
```

(下页继续)

(续上页)

```
#endif
```

```
#
# Reverse the order of flex items
#
cont = lv.obj(lv.scr_act())
cont.set_size(300, 220)
cont.center()
cont.set_flex_flow(lv.FLEX_FLOW.COLUMN_REVERSE)

for i  in range(6):
    obj = lv.obj(cont)
    obj.set_size(100, 50)

    label = lv.label(obj)
    label.set_text("Item: " + str(i))
    label.center()
```

Demonstrate column and row gap style properties

```
#include "../../lv_examples.h"
#if LV_USE_FLEX && LV_BUILD_EXAMPLES

static void row_gap_anim(void * obj, int32_t v)
{
    lv_obj_set_style_pad_row(obj, v, 0);
}

static void column_gap_anim(void * obj, int32_t v)
{
    lv_obj_set_style_pad_column(obj, v, 0);
}

/**
 * Demonstrate the effect of column and row gap style properties
 */
void lv_example_flex_5(void)
{
    lv_obj_t * cont = lv_obj_create(lv_scr_act());
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
```

(下页继续)

(续上页)

```
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_ROW_WRAP);

    uint32_t i;
    for(i = 0; i < 9; i++) {
        lv_obj_t * obj = lv_obj_create(cont);
        lv_obj_set_size(obj, 70, LV_SIZE_CONTENT);

        lv_obj_t * label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "%"LV_PRIu32, i);
        lv_obj_center(label);
    }

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_var(&a, cont);
    lv_anim_set_values(&a, 0, 10);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);

    lv_anim_set_exec_cb(&a, row_gap_anim);
    lv_anim_set_time(&a, 500);
    lv_anim_set_playback_time(&a, 500);
    lv_anim_start(&a);

    lv_anim_set_exec_cb(&a, column_gap_anim);
    lv_anim_set_time(&a, 3000);
    lv_anim_set_playback_time(&a, 3000);
    lv_anim_start(&a);
}

#endif
```

```
def row_gap_anim(obj, v):
    obj.set_style_pad_row(v, 0)


def column_gap_anim(obj, v):
    obj.set_style_pad_column(v, 0)

#
# Demonstrate the effect of column and row gap style properties
#

cont = lv.obj(lv.scr_act())
```

(下页继续)

(续上页)

```
cont.set_size(300, 220)
cont.center()
cont.set_flex_flow(lv.FLEX_FLOW.ROW_WRAP)

for i in range(9):
    obj = lv.obj(cont)
    obj.set_size(70, lv.SIZE.CONTENT)

    label = lv.label(obj)
    label.set_text(str(i))
    label.center()

a_row = lv.anim_t()
a_row.init()
a_row.set_var(cont)
a_row.set_values(0, 10)
a_row.set_repeat_count(lv.ANIM_REPEAT.INFINITE)

a_row.set_time(500)
a_row.set_playback_time(500)
a_row.set_custom_exec_cb(lambda a,val: row_gap_anim(cont,val))
lv.anim_t.start(a_row)

a_col = lv.anim_t()
a_col.init()
a_col.set_var(cont)
a_col.set_values(0, 10)
a_col.set_repeat_count(lv.ANIM_REPEAT.INFINITE)

a_col.set_time(3000)
a_col.set_playback_time(3000)
a_col.set_custom_exec_cb(lambda a,val: column_gap_anim(cont,val))

lv.anim_t.start(a_col)
```

RTL base direction changes order of the items

```c
#include "../../lv_examples.h"
#if LV_USE_FLEX && LV_BUILD_EXAMPLES

/**
 * RTL base direction changes order of the items.
 * Also demonstrate how horizontal scrolling works with RTL.
 */
void lv_example_flex_6(void)
{
    lv_obj_t * cont = lv_obj_create(lv_scr_act());
    lv_obj_set_style_base_dir(cont, LV_BASE_DIR_RTL, 0);
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_ROW_WRAP);

    uint32_t i;
    for(i = 0; i < 20; i++) {
        lv_obj_t * obj = lv_obj_create(cont);
        lv_obj_set_size(obj, 70, LV_SIZE_CONTENT);

        lv_obj_t * label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "%"LV_PRIu32, i);
        lv_obj_center(label);
    }
}
#endif
```

```python
#
# RTL base direction changes order of the items.
# Also demonstrate how horizontal scrolling works with RTL.
#

cont = lv.obj(lv.scr_act())
cont.set_style_base_dir(lv.BASE_DIR.RTL,0)
cont.set_size(300, 220)
cont.center()
cont.set_flex_flow(lv.FLEX_FLOW.ROW_WRAP)

for i in range(20):
    obj = lv.obj(cont)
    obj.set_size(70, lv.SIZE.CONTENT)
```

(下页继续)

(续上页)

```
    label = lv.label(obj)
    label.set_text(str(i))
    label.center()
```

Grid

A simple grid

```
#include "../../lv_examples.h"
#if LV_USE_GRID && LV_BUILD_EXAMPLES

/**
 * A simple grid
 */
void lv_example_grid_1(void)
{
    static lv_coord_t col_dsc[] = {70, 70, 70, LV_GRID_TEMPLATE_LAST};
    static lv_coord_t row_dsc[] = {50, 50, 50, LV_GRID_TEMPLATE_LAST};

    /*Create a container with grid*/
    lv_obj_t * cont = lv_obj_create(lv_scr_act());
    lv_obj_set_style_grid_column_dsc_array(cont, col_dsc, 0);
    lv_obj_set_style_grid_row_dsc_array(cont, row_dsc, 0);
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set_layout(cont, LV_LAYOUT_GRID);

    lv_obj_t * label;
    lv_obj_t * obj;

    uint32_t i;
    for(i = 0; i < 9; i++) {
        uint8_t col = i % 3;
        uint8_t row = i / 3;

        obj = lv_btn_create(cont);
        /*Stretch the cell horizontally and vertically too
         *Set span to 1 to make the cell 1 column/row sized*/
        lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, col, 1,
                             LV_GRID_ALIGN_STRETCH, row, 1);
```

(下页继续)

(续上页)

```
        label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "c%d, r%d", col, row);
        lv_obj_center(label);
    }
}

#endif
```

```
#
# A simple grid
#

col_dsc = [70, 70, 70, lv.GRID_TEMPLATE.LAST]
row_dsc = [50, 50, 50, lv.GRID_TEMPLATE.LAST]

# Create a container with grid
cont = lv.obj(lv.scr_act())
cont.set_style_grid_column_dsc_array(col_dsc, 0)
cont.set_style_grid_row_dsc_array(row_dsc, 0)
cont.set_size(300, 220)
cont.center()
cont.set_layout(lv.LAYOUT_GRID.value)

for i in range(9):
    col = i % 3
    row = i // 3

    obj = lv.btn(cont)
    # Stretch the cell horizontally and vertically too
    # Set span to 1 to make the cell 1 column/row sized
    obj.set_grid_cell(lv.GRID_ALIGN.STRETCH, col, 1,
                      lv.GRID_ALIGN.STRETCH, row, 1)

    label = lv.label(obj)
    label.set_text("c" +str(col) +  "r" +str(row))
    label.center()
```

Demonstrate cell placement and span

```
#include "../../lv_examples.h"
#if LV_USE_GRID && LV_BUILD_EXAMPLES


/**
 * Demonstrate cell placement and span
 */
void lv_example_grid_2(void)
{
    static lv_coord_t col_dsc[] = {70, 70, 70, LV_GRID_TEMPLATE_LAST};
    static lv_coord_t row_dsc[] = {50, 50, 50, LV_GRID_TEMPLATE_LAST};

    /*Create a container with grid*/
    lv_obj_t * cont = lv_obj_create(lv_scr_act());
    lv_obj_set_grid_dsc_array(cont, col_dsc, row_dsc);
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);

    lv_obj_t * label;
    lv_obj_t * obj;

    /*Cell to 0;0 and align to to the start (left/top) horizontally and vertically␣
↪too*/
    obj = lv_obj_create(cont);
    lv_obj_set_size(obj, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
    lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_START, 0, 1,
                         LV_GRID_ALIGN_START, 0, 1);
    label = lv_label_create(obj);
    lv_label_set_text(label, "c0, r0");

    /*Cell to 1;0 and align to to the start (left) horizontally and center vertically␣
↪too*/
    obj = lv_obj_create(cont);
    lv_obj_set_size(obj, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
    lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_START, 1, 1,
                         LV_GRID_ALIGN_CENTER, 0, 1);
    label = lv_label_create(obj);
    lv_label_set_text(label, "c1, r0");

    /*Cell to 2;0 and align to to the start (left) horizontally and end (bottom)␣
↪vertically too*/
    obj = lv_obj_create(cont);
```

(下页继续)

(续上页)

```
    lv_obj_set_size(obj, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
    lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_START, 2, 1,
                         LV_GRID_ALIGN_END, 0, 1);
    label = lv_label_create(obj);
    lv_label_set_text(label, "c2, r0");

    /*Cell to 1;1 but 2 column wide (span = 2).Set width and height to stretched.*/
    obj = lv_obj_create(cont);
    lv_obj_set_size(obj, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
    lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, 1, 2,
                         LV_GRID_ALIGN_STRETCH, 1, 1);
    label = lv_label_create(obj);
    lv_label_set_text(label, "c1-2, r1");

    /*Cell to 0;1 but 2 rows tall (span = 2).Set width and height to stretched.*/
    obj = lv_obj_create(cont);
    lv_obj_set_size(obj, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
    lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, 0, 1,
                         LV_GRID_ALIGN_STRETCH, 1, 2);
    label = lv_label_create(obj);
    lv_label_set_text(label, "c0\nr1-2");
}

#endif
```

```
#
# Demonstrate cell placement and span
#

col_dsc = [70, 70, 70, lv.GRID_TEMPLATE.LAST]
row_dsc = [50, 50, 50, lv.GRID_TEMPLATE.LAST]

# Create a container with grid
cont = lv.obj(lv.scr_act())
cont.set_grid_dsc_array(col_dsc, row_dsc)
cont.set_size(300, 220)
cont.center()

# Cell to 0;0 and align to the start (left/top) horizontally and vertically too
obj = lv.obj(cont)
obj.set_size(lv.SIZE.CONTENT, lv.SIZE.CONTENT)
obj.set_grid_cell(lv.GRID_ALIGN.START, 0, 1,
                  lv.GRID_ALIGN.START, 0, 1)
```

(下页继续)

(续上页)

```
label = lv.label(obj)
label.set_text("c0, r0")

# Cell to 1;0 and align to the start (left) horizontally and center vertically too
obj = lv.obj(cont)
obj.set_size(lv.SIZE.CONTENT, lv.SIZE.CONTENT)
obj.set_grid_cell(lv.GRID_ALIGN.START, 1, 1,
                  lv.GRID_ALIGN.CENTER, 0, 1)
label = lv.label(obj)
label.set_text("c1, r0")

# Cell to 2;0 and align to the start (left) horizontally and end (bottom) vertically␣
↪too
obj = lv.obj(cont)
obj.set_size(lv.SIZE.CONTENT, lv.SIZE.CONTENT)
obj.set_grid_cell(lv.GRID_ALIGN.START, 2, 1,
                  lv.GRID_ALIGN.END, 0, 1)
label = lv.label(obj)
label.set_text("c2, r0")

# Cell to 1;1 but 2 column wide (span = 2).Set width and height to stretched.
obj = lv.obj(cont)
obj.set_size(lv.SIZE.CONTENT, lv.SIZE.CONTENT)
obj.set_grid_cell(lv.GRID_ALIGN.STRETCH, 1, 2,
                  lv.GRID_ALIGN.STRETCH, 1, 1)
label = lv.label(obj)
label.set_text("c1-2, r1")

# Cell to 0;1 but 2 rows tall (span = 2).Set width and height to stretched.
obj = lv.obj(cont)
obj.set_size(lv.SIZE.CONTENT, lv.SIZE.CONTENT)
obj.set_grid_cell(lv.GRID_ALIGN.STRETCH, 0, 1,
                  lv.GRID_ALIGN.STRETCH, 1, 2)
label = lv.label(obj)
label.set_text("c0\nr1-2")
```

Demonstrate grid's "free unit"

```
#include "../../lv_examples.h"
#if LV_USE_GRID && LV_BUILD_EXAMPLES

/**
 * Demonstrate grid's "free unit"
 */
void lv_example_grid_3(void)
{
    /*Column 1: fix width 60 px
     *Column 2: 1 unit from the remaining free space
     *Column 3: 2 unit from the remaining free space*/
    static lv_coord_t col_dsc[] = {60, LV_GRID_FR(1), LV_GRID_FR(2), LV_GRID_TEMPLATE_
↪LAST};

    /*Row 1: fix width 50 px
     *Row 2: 1 unit from the remaining free space
     *Row 3: fix width 50 px*/
    static lv_coord_t row_dsc[] = {50, LV_GRID_FR(1), 50, LV_GRID_TEMPLATE_LAST};

    /*Create a container with grid*/
    lv_obj_t * cont = lv_obj_create(lv_scr_act());
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set_grid_dsc_array(cont, col_dsc, row_dsc);

    lv_obj_t * label;
    lv_obj_t * obj;
    uint32_t i;
    for(i = 0; i < 9; i++) {
        uint8_t col = i % 3;
        uint8_t row = i / 3;

        obj = lv_obj_create(cont);
        /*Stretch the cell horizontally and vertically too
         *Set span to 1 to make the cell 1 column/row sized*/
        lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, col, 1,
                             LV_GRID_ALIGN_STRETCH, row, 1);

        label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "%d,%d", col, row);
        lv_obj_center(label);
    }
```

(下页继续)

(续上页)

```
}

#endif
```

```
#
# Demonstrate grid's "free unit"
#

# Column 1: fix width 60 px
# Column 2: 1 unit from the remaining free space
# Column 3: 2 unit from the remaining free space

col_dsc = [60, lv.grid_fr(1), lv.grid_fr(2), lv.GRID_TEMPLATE.LAST]

# Row 1: fix width 60 px
# Row 2: 1 unit from the remaining free space
# Row 3: fix width 60 px

row_dsc = [40, lv.grid_fr(1), 40, lv.GRID_TEMPLATE.LAST]

# Create a container with grid
cont = lv.obj(lv.scr_act())
cont.set_size(300, 220)
cont.center()
cont.set_grid_dsc_array(col_dsc, row_dsc)

for i in range(9):
    col = i % 3
    row = i // 3

    obj = lv.obj(cont)
    # Stretch the cell horizontally and vertically too
    # Set span to 1 to make the cell 1 column/row sized
    obj.set_grid_cell(lv.GRID_ALIGN.STRETCH, col, 1,
                      lv.GRID_ALIGN.STRETCH, row, 1)

    label = lv.label(obj)
    label.set_text("%d,%d"%(col, row))
    label.center()
```

Demonstrate track placement

```
#include "../../lv_examples.h"
#if LV_USE_GRID && LV_BUILD_EXAMPLES

/**
 * Demonstrate track placement
 */
void lv_example_grid_4(void)
{
    static lv_coord_t col_dsc[] = {60, 60, 60, LV_GRID_TEMPLATE_LAST};
    static lv_coord_t row_dsc[] = {45, 45, 45, LV_GRID_TEMPLATE_LAST};


    /*Add space between the columns and move the rows to the bottom (end)*/

    /*Create a container with grid*/
    lv_obj_t * cont = lv_obj_create(lv_scr_act());
    lv_obj_set_grid_align(cont, LV_GRID_ALIGN_SPACE_BETWEEN, LV_GRID_ALIGN_END);
    lv_obj_set_grid_dsc_array(cont, col_dsc, row_dsc);
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);

    lv_obj_t * label;
    lv_obj_t * obj;
    uint32_t i;
    for(i = 0; i < 9; i++) {
        uint8_t col = i % 3;
        uint8_t row = i / 3;

        obj = lv_obj_create(cont);
        /*Stretch the cell horizontally and vertically too
         *Set span to 1 to make the cell 1 column/row sized*/
        lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, col, 1,
                             LV_GRID_ALIGN_STRETCH, row, 1);

        label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "%d,%d", col, row);
        lv_obj_center(label);
    }
}

#endif
```

```
#
# Demonstrate track placement
#

col_dsc = [60, 60, 60, lv.GRID_TEMPLATE.LAST]
row_dsc = [40, 40, 40, lv.GRID_TEMPLATE.LAST]


# Add space between the columns and move the rows to the bottom (end)

# Create a container with grid
cont = lv.obj(lv.scr_act())
cont.set_grid_align(lv.GRID_ALIGN.SPACE_BETWEEN, lv.GRID_ALIGN.END)
cont.set_grid_dsc_array(col_dsc, row_dsc)
cont.set_size(300, 220)
cont.center()


for i in range(9):
    col = i % 3
    row = i // 3

    obj = lv.obj(cont)
    # Stretch the cell horizontally and vertically too
    # Set span to 1 to make the cell 1 column/row sized
    obj.set_grid_cell(lv.GRID_ALIGN.STRETCH, col, 1,
                      lv.GRID_ALIGN.STRETCH, row, 1)

    label = lv.label(obj)
    label.set_text("{:d}{:d}".format(col, row))
    label.center()
```

Demonstrate column and row gap

```
#include "../../lv_examples.h"
#if LV_USE_GRID && LV_BUILD_EXAMPLES

static void row_gap_anim(void * obj, int32_t v)
{
    lv_obj_set_style_pad_row(obj, v, 0);
}
```

(下页继续)

(续上页)

```
static void column_gap_anim(void * obj, int32_t v)
{
    lv_obj_set_style_pad_column(obj, v, 0);
}

/**
 * Demonstrate column and row gap
 */
void lv_example_grid_5(void)
{

    /*60x60 cells*/
    static lv_coord_t col_dsc[] = {60, 60, 60, LV_GRID_TEMPLATE_LAST};
    static lv_coord_t row_dsc[] = {45, 45, 45, LV_GRID_TEMPLATE_LAST};

    /*Create a container with grid*/
    lv_obj_t * cont = lv_obj_create(lv_scr_act());
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set_grid_dsc_array(cont, col_dsc, row_dsc);

    lv_obj_t * label;
    lv_obj_t * obj;
    uint32_t i;
    for(i = 0; i < 9; i++) {
        uint8_t col = i % 3;
        uint8_t row = i / 3;

        obj = lv_obj_create(cont);
        lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, col, 1,
                             LV_GRID_ALIGN_STRETCH, row, 1);
        label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "%d,%d", col, row);
        lv_obj_center(label);
    }

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_var(&a, cont);
    lv_anim_set_values(&a, 0, 10);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);

    lv_anim_set_exec_cb(&a, row_gap_anim);
```

(下页继续)

(续上页)

```
    lv_anim_set_time(&a, 500);
    lv_anim_set_playback_time(&a, 500);
    lv_anim_start(&a);

    lv_anim_set_exec_cb(&a, column_gap_anim);
    lv_anim_set_time(&a, 3000);
    lv_anim_set_playback_time(&a, 3000);
    lv_anim_start(&a);
}

#endif
```

```
def row_gap_anim(obj, v):
    obj.set_style_pad_row(v, 0)

def column_gap_anim(obj, v):
    obj.set_style_pad_column(v, 0)

#
# Demonstrate column and row gap
#

# 60x60 cells
col_dsc = [60, 60, 60, lv.GRID_TEMPLATE.LAST]
row_dsc = [40, 40, 40, lv.GRID_TEMPLATE.LAST]

# Create a container with grid
cont = lv.obj(lv.scr_act())
cont.set_size(300, 220)
cont.center()
cont.set_grid_dsc_array(col_dsc, row_dsc)

for i in range(9):
    col = i % 3
    row = i // 3

    obj = lv.obj(cont)
    obj.set_grid_cell(lv.GRID_ALIGN.STRETCH, col, 1,
                      lv.GRID_ALIGN.STRETCH, row, 1)
    label = lv.label(obj)
    label.set_text("{:d},{:d}".format(col, row))
    label.center()
```

(下页继续)

(续上页)

```
    a_row = lv.anim_t()
    a_row.init()
    a_row.set_var(cont)
    a_row.set_values(0, 10)
    a_row.set_repeat_count(lv.ANIM_REPEAT.INFINITE)
    a_row.set_time(500)
    a_row.set_playback_time(500)
    a_row. set_custom_exec_cb(lambda a,val: row_gap_anim(cont,val))
    lv.anim_t.start(a_row)

    a_col = lv.anim_t()
    a_col.init()
    a_col.set_var(cont)
    a_col.set_values(0, 10)
    a_col.set_repeat_count(lv.ANIM_REPEAT.INFINITE)
    a_col.set_time(500)
    a_col.set_playback_time(500)
    a_col. set_custom_exec_cb(lambda a,val: column_gap_anim(cont,val))
    lv.anim_t.start(a_col)
```

Demonstrate RTL direction on grid

```
#include "../../lv_examples.h"
#if LV_USE_GRID && LV_BUILD_EXAMPLES

/**
 * Demonstrate RTL direction on grid
 */
void lv_example_grid_6(void)
{

    static lv_coord_t col_dsc[] = {60, 60, 60, LV_GRID_TEMPLATE_LAST};
    static lv_coord_t row_dsc[] = {45, 45, 45, LV_GRID_TEMPLATE_LAST};

    /*Create a container with grid*/
    lv_obj_t * cont = lv_obj_create(lv_scr_act());
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set_style_base_dir(cont, LV_BASE_DIR_RTL, 0);
```

(下页继续)

(续上页)

```
    lv_obj_set_grid_dsc_array(cont, col_dsc, row_dsc);

    lv_obj_t * label;
    lv_obj_t * obj;
    uint32_t i;
    for(i = 0; i < 9; i++) {
        uint8_t col = i % 3;
        uint8_t row = i / 3;

        obj = lv_obj_create(cont);
        /*Stretch the cell horizontally and vertically too
         *Set span to 1 to make the cell 1 column/row sized*/
        lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, col, 1,
                             LV_GRID_ALIGN_STRETCH, row, 1);

        label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "%d,%d", col, row);
        lv_obj_center(label);
    }
}

#endif
```

```
#
# Demonstrate RTL direction on grid
#
col_dsc = [60, 60, 60, lv.GRID_TEMPLATE.LAST]
row_dsc = [40, 40, 40, lv.GRID_TEMPLATE.LAST]

# Create a container with grid
cont = lv.obj(lv.scr_act())
cont.set_size(300, 220)
cont.center()
cont.set_style_base_dir(lv.BASE_DIR.RTL,0)
cont.set_grid_dsc_array(col_dsc, row_dsc)

for i in range(9):
    col = i % 3
    row = i // 3

    obj = lv.obj(cont)
    # Stretch the cell horizontally and vertically too
    # Set span to 1 to make the cell 1 column/row sized
```

(下页继续)

(续上页)

```
    obj.set_grid_cell(lv.GRID_ALIGN.STRETCH, col, 1,
                      lv.GRID_ALIGN.STRETCH, row, 1)

    label = lv.label(obj)
    label.set_text("{:d},{:d}".format(col, row))
    label.center()
```

2.2.6 Scrolling

Nested scrolling

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

/**
 * Demonstrate how scrolling appears automatically
 */
void lv_example_scroll_1(void)
{
    /*Create an object with the new style*/
    lv_obj_t * panel = lv_obj_create(lv_scr_act());
    lv_obj_set_size(panel, 200, 200);
    lv_obj_center(panel);

    lv_obj_t * child;
    lv_obj_t * label;

    child = lv_obj_create(panel);
    lv_obj_set_pos(child, 0, 0);
    lv_obj_set_size(child, 70, 70);
    label = lv_label_create(child);
    lv_label_set_text(label, "Zero");
    lv_obj_center(label);

    child = lv_obj_create(panel);
    lv_obj_set_pos(child, 160, 80);
    lv_obj_set_size(child, 80, 80);

    lv_obj_t * child2 = lv_btn_create(child);
    lv_obj_set_size(child2, 100, 50);
```

(下页继续)

(续上页)

```
    label = lv_label_create(child2);
    lv_label_set_text(label, "Right");
    lv_obj_center(label);

    child = lv_obj_create(panel);
    lv_obj_set_pos(child, 40, 160);
    lv_obj_set_size(child, 100, 70);
    label = lv_label_create(child);
    lv_label_set_text(label, "Bottom");
    lv_obj_center(label);
}

#endif
```

```
#
# Demonstrate how scrolling appears automatically
#
# Create an object with the new style
panel = lv.obj(lv.scr_act())
panel.set_size(200, 200)
panel.center()

child = lv.obj(panel)
child.set_pos(0, 0)
label = lv.label(child)
label.set_text("Zero")
label.center()

child = lv.obj(panel)
child.set_pos(-40, 100)
label = lv.label(child)
label.set_text("Left")
label.center()

child = lv.obj(panel)
child.set_pos(90, -30)
label = lv.label(child)
label.set_text("Top")
label.center()

child = lv.obj(panel)
child.set_pos(150, 80)
label = lv.label(child)
```

(下页继续)

(续上页)

```
label.set_text("Right")
label.center()

child = lv.obj(panel)
child.set_pos(60, 170)
label = lv.label(child)
label.set_text("Bottom")
label.center()
```

Snapping

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_FLEX

static void sw_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * sw = lv_event_get_target(e);

    if(code == LV_EVENT_VALUE_CHANGED) {
        lv_obj_t * list = lv_event_get_user_data(e);

        if(lv_obj_has_state(sw, LV_STATE_CHECKED)) lv_obj_add_flag(list, LV_OBJ_FLAG_
↪SCROLL_ONE);
        else lv_obj_clear_flag(list, LV_OBJ_FLAG_SCROLL_ONE);
    }
}

/**
 * Show an example to scroll snap
 */
void lv_example_scroll_2(void)
{
    lv_obj_t * panel = lv_obj_create(lv_scr_act());
    lv_obj_set_size(panel, 280, 120);
    lv_obj_set_scroll_snap_x(panel, LV_SCROLL_SNAP_CENTER);
    lv_obj_set_flex_flow(panel, LV_FLEX_FLOW_ROW);
    lv_obj_align(panel, LV_ALIGN_CENTER, 0, 20);

    uint32_t i;
    for(i = 0; i < 10; i++) {
```

(下页继续)

(续上页)

```
        lv_obj_t * btn = lv_btn_create(panel);
        lv_obj_set_size(btn, 150, lv_pct(100));

        lv_obj_t * label = lv_label_create(btn);
        if(i == 3) {
            lv_label_set_text_fmt(label, "Panel %"LV_PRIu32"\nno snap", i);
            lv_obj_clear_flag(btn, LV_OBJ_FLAG_SNAPPABLE);
        } else {
            lv_label_set_text_fmt(label, "Panel %"LV_PRIu32, i);
        }

        lv_obj_center(label);
    }
    lv_obj_update_snap(panel, LV_ANIM_ON);

#if LV_USE_SWITCH
    /*Switch between "One scroll" and "Normal scroll" mode*/
    lv_obj_t * sw = lv_switch_create(lv_scr_act());
    lv_obj_align(sw, LV_ALIGN_TOP_RIGHT, -20, 10);
    lv_obj_add_event_cb(sw, sw_event_cb, LV_EVENT_ALL, panel);
    lv_obj_t * label = lv_label_create(lv_scr_act());
    lv_label_set_text(label, "One scroll");
    lv_obj_align_to(label, sw, LV_ALIGN_OUT_BOTTOM_MID, 0, 5);
#endif
}

#endif
```

```
def sw_event_cb(e,panel):

    code = e.get_code()
    sw = e.get_target()

    if code == lv.EVENT.VALUE_CHANGED:

        if sw.has_state(lv.STATE.CHECKED):
            panel.add_flag(lv.obj.FLAG.SCROLL_ONE)
        else:
            panel.clear_flag(lv.obj.FLAG.SCROLL_ONE)


#
# Show an example to scroll snap
```

(下页继续)

(续上页)

```
#

panel = lv.obj(lv.scr_act())
panel.set_size(280, 150)
panel.set_scroll_snap_x(lv.SCROLL_SNAP.CENTER)
panel.set_flex_flow(lv.FLEX_FLOW.ROW)
panel.center()

for i in range(10):
    btn = lv.btn(panel)
    btn.set_size(150, 100)

    label = lv.label(btn)
    if i == 3:
        label.set_text("Panel {:d}\nno snap".format(i))
        btn.clear_flag(lv.obj.FLAG.SNAPPABLE)
    else:
        label.set_text("Panel {:d}".format(i))
    label.center()

panel.update_snap(lv.ANIM.ON)


# Switch between "One scroll" and "Normal scroll" mode
sw = lv.switch(lv.scr_act())
sw.align(lv.ALIGN.TOP_RIGHT, -20, 10)
sw.add_event_cb(lambda evt:  sw_event_cb(evt,panel), lv.EVENT.ALL, None)
label = lv.label(lv.scr_act())
label.set_text("One scroll")
label.align_to(sw, lv.ALIGN.OUT_BOTTOM_MID, 0, 5)
```

Floating button

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_LIST

static uint32_t btn_cnt = 1;

static void float_btn_event_cb(lv_event_t * e)
{
```

(下页继续)

(续上页)

```
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * float_btn = lv_event_get_target(e);

    if(code == LV_EVENT_CLICKED) {
        lv_obj_t * list = lv_event_get_user_data(e);
        char buf[32];
        lv_snprintf(buf, sizeof(buf), "Track %d", (int)btn_cnt);
        lv_obj_t * list_btn = lv_list_add_btn(list, LV_SYMBOL_AUDIO, buf);
        btn_cnt++;

        lv_obj_move_foreground(float_btn);

        lv_obj_scroll_to_view(list_btn, LV_ANIM_ON);
    }
}

/**
 * Create a list a with a floating button
 */
void lv_example_scroll_3(void)
{
    lv_obj_t * list = lv_list_create(lv_scr_act());
    lv_obj_set_size(list, 280, 220);
    lv_obj_center(list);

    for(btn_cnt = 1; btn_cnt <= 2; btn_cnt++) {
        char buf[32];
        lv_snprintf(buf, sizeof(buf), "Track %d", (int)btn_cnt);
        lv_list_add_btn(list, LV_SYMBOL_AUDIO, buf);
    }

    lv_obj_t * float_btn = lv_btn_create(list);
    lv_obj_set_size(float_btn, 50, 50);
    lv_obj_add_flag(float_btn, LV_OBJ_FLAG_FLOATING);
    lv_obj_align(float_btn, LV_ALIGN_BOTTOM_RIGHT, 0, -lv_obj_get_style_pad_
↪right(list, LV_PART_MAIN));
    lv_obj_add_event_cb(float_btn, float_btn_event_cb, LV_EVENT_ALL, list);
    lv_obj_set_style_radius(float_btn, LV_RADIUS_CIRCLE, 0);
    lv_obj_set_style_bg_img_src(float_btn, LV_SYMBOL_PLUS, 0);
    lv_obj_set_style_text_font(float_btn, lv_theme_get_font_large(float_btn), 0);
}

#endif
```

```
class ScrollExample_3():
    def __init__(self):
        self.btn_cnt = 1
        #
        # Create a list a with a floating button
        #

        list = lv.list(lv.scr_act())
        list.set_size(280, 220)
        list.center()

        for btn_cnt in range(2):
            list.add_btn(lv.SYMBOL.AUDIO,"Track {:d}".format(btn_cnt))

        float_btn = lv.btn(list)
        float_btn.set_size(50, 50)
        float_btn.add_flag(lv.obj.FLAG.FLOATING)
        float_btn.align(lv.ALIGN.BOTTOM_RIGHT, 0, -list.get_style_pad_right(lv.PART.
↪MAIN))
        float_btn.add_event_cb(lambda evt: self.float_btn_event_cb(evt,list), lv.
↪EVENT.ALL, None)
        float_btn.set_style_radius(lv.RADIUS.CIRCLE, 0)
        float_btn.set_style_bg_img_src(lv.SYMBOL.PLUS, 0)
        float_btn.set_style_text_font(lv.theme_get_font_large(float_btn), 0)

    def float_btn_event_cb(self,e,list):
        code = e.get_code()
        float_btn = e.get_target()

        if code == lv.EVENT.CLICKED:
            list_btn = list.add_btn(lv.SYMBOL.AUDIO, "Track {:d}".format(self.btn_
↪cnt))
            self.btn_cnt += 1

            float_btn.move_foreground()

            list_btn.scroll_to_view(lv.ANIM.ON)

scroll_example_3 = ScrollExample_3()
```

Styling the scrollbars

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_LIST


/**
 * Styling the scrollbars
 */
void lv_example_scroll_4(void)
{
    lv_obj_t * obj = lv_obj_create(lv_scr_act());
    lv_obj_set_size(obj, 200, 100);
    lv_obj_center(obj);

    lv_obj_t * label = lv_label_create(obj);
    lv_label_set_text(label,
            "Lorem ipsum dolor sit amet, consectetur adipiscing elit.\n"
            "Etiam dictum, tortor vestibulum lacinia laoreet, mi neque consectetur␣
↪neque, vel mattis odio dolor egestas ligula. \n"
            "Sed vestibulum sapien nulla, id convallis ex porttitor nec. \n"
            "Duis et massa eu libero accumsan faucibus a in arcu. \n"
            "Ut pulvinar odio lorem, vel tempus turpis condimentum quis. Nam␣
↪consectetur condimentum sem in auctor. \n"
            "Sed nisl augue, venenatis in blandit et, gravida ac tortor. \n"
            "Etiam dapibus elementum suscipit. \n"
            "Proin mollis sollicitudin convallis. \n"
            "Integer dapibus tempus arcu nec viverra. \n"
            "Donec molestie nulla enim, eu interdum velit placerat quis. \n"
            "Donec id efficitur risus, at molestie turpis. \n"
            "Suspendisse vestibulum consectetur nunc ut commodo. \n"
            "Fusce molestie rhoncus nisi sit amet tincidunt. \n"
            "Suspendisse a nunc ut magna ornare volutpat.");


    /*Remove the style of scrollbar to have clean start*/
    lv_obj_remove_style(obj, NULL, LV_PART_SCROLLBAR | LV_STATE_ANY);

    /*Create a transition the animate the some properties on state change*/
    static const lv_style_prop_t props[] = {LV_STYLE_BG_OPA, LV_STYLE_WIDTH, 0};
    static lv_style_transition_dsc_t trans;
    lv_style_transition_dsc_init(&trans, props, lv_anim_path_linear, 200, 0, NULL);

    /*Create a style for the scrollbars*/
```

(下页继续)

(续上页)

```
    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_width(&style, 4);      /*Width of the scrollbar*/
    lv_style_set_pad_right(&style, 5);  /*Space from the parallel side*/
    lv_style_set_pad_top(&style, 5);    /*Space from the perpendicular side*/

    lv_style_set_radius(&style, 2);
    lv_style_set_bg_opa(&style, LV_OPA_70);
    lv_style_set_bg_color(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_border_color(&style, lv_palette_darken(LV_PALETTE_BLUE, 3));
    lv_style_set_border_width(&style, 2);
    lv_style_set_shadow_width(&style, 8);
    lv_style_set_shadow_spread(&style, 2);
    lv_style_set_shadow_color(&style, lv_palette_darken(LV_PALETTE_BLUE, 1));

    lv_style_set_transition(&style, &trans);

    /*Make the scrollbars wider and use 100% opacity when scrolled*/
    static lv_style_t style_scrolled;
    lv_style_init(&style_scrolled);
    lv_style_set_width(&style_scrolled, 8);
    lv_style_set_bg_opa(&style_scrolled, LV_OPA_COVER);

    lv_obj_add_style(obj, &style, LV_PART_SCROLLBAR);
    lv_obj_add_style(obj, &style_scrolled, LV_PART_SCROLLBAR | LV_STATE_SCROLLED);
}

#endif
```

```
#
# Styling the scrollbars
#
obj = lv.obj(lv.scr_act())
obj.set_size(200, 100)
obj.center()

label = lv.label(obj)
label.set_text(
"""
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Etiam dictum, tortor vestibulum lacinia laoreet, mi neque consectetur neque, vel␣
↪mattis odio dolor egestas ligula.
Sed vestibulum sapien nulla, id convallis ex porttitor nec.
```

(下页继续)

(续上页)

```
Duis et massa eu libero accumsan faucibus a in arcu.
Ut pulvinar odio lorem, vel tempus turpis condimentum quis. Nam consectetur␣
↪condimentum sem in auctor.
Sed nisl augue, venenatis in blandit et, gravida ac tortor.
Etiam dapibus elementum suscipit.
Proin mollis sollicitudin convallis.
Integer dapibus tempus arcu nec viverra.
Donec molestie nulla enim, eu interdum velit placerat quis.
Donec id efficitur risus, at molestie turpis.
Suspendisse vestibulum consectetur nunc ut commodo.
Fusce molestie rhoncus nisi sit amet tincidunt.
Suspendisse a nunc ut magna ornare volutpat.
""")


# Remove the style of scrollbar to have clean start
obj.remove_style(None, lv.PART.SCROLLBAR | lv.STATE.ANY)

# Create a transition the animate the some properties on state change
props = [lv.STYLE.BG_OPA, lv.STYLE.WIDTH, 0]
trans = lv.style_transition_dsc_t()
trans.init(props, lv.anim_t.path_linear, 200, 0, None)

# Create a style for the scrollbars
style = lv.style_t()
style.init()
style.set_width(4)              # Width of the scrollbar
style.set_pad_right(5)          # Space from the parallel side
style.set_pad_top(5)            # Space from the perpendicular side

style.set_radius(2)
style.set_bg_opa(lv.OPA._70)
style.set_bg_color(lv.palette_main(lv.PALETTE.BLUE))
style.set_border_color(lv.palette_darken(lv.PALETTE.BLUE, 3))
style.set_border_width(2)
style.set_shadow_width(8)
style.set_shadow_spread(2)
style.set_shadow_color(lv.palette_darken(lv.PALETTE.BLUE, 1))

style.set_transition(trans)

# Make the scrollbars wider and use 100% opacity when scrolled
style_scrolled = lv.style_t()
```

(下页继续)

(续上页)

```
style_scrolled.init()
style_scrolled.set_width(8)
style_scrolled.set_bg_opa(lv.OPA.COVER)

obj.add_style(style, lv.PART.SCROLLBAR)
obj.add_style(style_scrolled, lv.PART.SCROLLBAR | lv.STATE.SCROLLED)
```

Right to left scrolling

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_FONT_DEJAVU_16_PERSIAN_HEBREW


/**
 * Scrolling with Right To Left base direction
 */
void lv_example_scroll_5(void)
{
    lv_obj_t * obj = lv_obj_create(lv_scr_act());
    lv_obj_set_style_base_dir(obj, LV_BASE_DIR_RTL, 0);
    lv_obj_set_size(obj, 200, 100);
    lv_obj_center(obj);

    lv_obj_t * label = lv_label_create(obj);
    lv_label_set_text(label,"میکروکُنترولر (به انگلیسی: Microcontroller) گونه‌ای␣
↪ریزپردازنده است که دارای حافظهٔ دسترسی تصادفی (RAM) و حافظهٔ فقط‌خواندنی (ROM)، تایمر،␣
↪پورت‌های ورودی و خروجی (I/O) و درگاه ترتیبی (Serial Port پورت سریال)، درون خود تراشه␣
↪است، و می‌تواند به تنهایی ابزارهای دیگر را کنترل کند. به عبارت دیگر یک میکروکنترلر،␣
↪مدار مجتمع کوچکی است که از یک CPU کوچک و اجزای دیگری مانند تایمر، درگاه‌های ورودی و␣
↪خروجی آنالوگ و دیجیتال و حافظه تشکیل شده‌است.");
    lv_obj_set_width(label, 400);
    lv_obj_set_style_text_font(label, &lv_font_dejavu_16_persian_hebrew, 0);

}

#endif
```

```
#
# Scrolling with Right To Left base direction
#
obj = lv.obj(lv.scr_act())
```

(下页继续)

(续上页)

```
obj.set_style_base_dir(lv.BASE_DIR.RTL, 0)
obj.set_size(200, 100)
obj.center()

label = lv.label(obj)
label.set_text("میکروکُنترولر (به انگلیسی: Microcontroller) گونه‌ای ریزپردازنده است که دارای حافظهٔ دسترسی تصادفی (RAM) و حافظهٔ فقط‌خواندنی (ROM)، تایمر، پورت‌های ورودی و خروجی (I/O) و درگاه ترتیبی (Serial Port پورت سریال)، درون خود تراشه است، و می‌تواند به تنهایی ابزارهای دیگر را کنترل کند. به عبارت دیگر یک میکروکنترلر، مدار مجتمع کوچکی است که از یک CPU کوچک و اجزای دیگری مانند تایمر، درگاه‌های ورودی و خروجی آنالوگ و دیجیتال و حافظه تشکیل شده‌است.")
label.set_width(400)
label.set_style_text_font(lv.font_dejavu_16_persian_hebrew, 0)
```

Translate on scroll

```c
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_FLEX

static void scroll_event_cb(lv_event_t * e)
{
    lv_obj_t * cont = lv_event_get_target(e);

    lv_area_t cont_a;
    lv_obj_get_coords(cont, &cont_a);
    lv_coord_t cont_y_center = cont_a.y1 + lv_area_get_height(&cont_a) / 2;

    lv_coord_t r = lv_obj_get_height(cont) * 7 / 10;
    uint32_t i;
    uint32_t child_cnt = lv_obj_get_child_cnt(cont);
    for(i = 0; i < child_cnt; i++) {
        lv_obj_t * child = lv_obj_get_child(cont, i);
        lv_area_t child_a;
        lv_obj_get_coords(child, &child_a);

        lv_coord_t child_y_center = child_a.y1 + lv_area_get_height(&child_a) / 2;

        lv_coord_t diff_y = child_y_center - cont_y_center;
        diff_y = LV_ABS(diff_y);

        /*Get the x of diff_y on a circle.*/
```

(下页继续)

(续上页)

```
        lv_coord_t x;
        /*If diff_y is out of the circle use the last point of the circle (the␣
↪radius)*/
        if(diff_y >= r) {
            x = r;
        } else {
            /*Use Pythagoras theorem to get x from radius and y*/
            uint32_t x_sqr = r * r - diff_y * diff_y;
            lv_sqrt_res_t res;
            lv_sqrt(x_sqr, &res, 0x8000);   /*Use lvgl's built in sqrt root function*/
            x = r - res.i;
        }

        /*Translate the item by the calculated X coordinate*/
        lv_obj_set_style_translate_x(child, x, 0);

        /*Use some opacity with larger translations*/
        lv_opa_t opa = lv_map(x, 0, r, LV_OPA_TRANSP, LV_OPA_COVER);
        lv_obj_set_style_opa(child, LV_OPA_COVER - opa, 0);
    }
}

/**
 * Translate the object as they scroll
 */
void lv_example_scroll_6(void)
{
    lv_obj_t * cont = lv_obj_create(lv_scr_act());
    lv_obj_set_size(cont, 200, 200);
    lv_obj_center(cont);
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_COLUMN);
    lv_obj_add_event_cb(cont, scroll_event_cb, LV_EVENT_SCROLL, NULL);
    lv_obj_set_style_radius(cont, LV_RADIUS_CIRCLE, 0);
    lv_obj_set_style_clip_corner(cont, true, 0);
    lv_obj_set_scroll_dir(cont, LV_DIR_VER);
    lv_obj_set_scroll_snap_y(cont, LV_SCROLL_SNAP_CENTER);
    lv_obj_set_scrollbar_mode(cont, LV_SCROLLBAR_MODE_OFF);

    uint32_t i;
    for(i = 0; i < 20; i++) {
        lv_obj_t * btn = lv_btn_create(cont);
        lv_obj_set_width(btn, lv_pct(100));
```

(下页继续)

(续上页)

```
        lv_obj_t * label = lv_label_create(btn);
        lv_label_set_text_fmt(label, "Button %"LV_PRIu32, i);
    }

    /*Update the buttons position manually for first*/
    lv_event_send(cont, LV_EVENT_SCROLL, NULL);

    /*Be sure the fist button is in the middle*/
    lv_obj_scroll_to_view(lv_obj_get_child(cont, 0), LV_ANIM_OFF);
}

#endif
```

```
def scroll_event_cb(e):

    cont = e.get_target()

    cont_a = lv.area_t()
    cont.get_coords(cont_a)
    cont_y_center = cont_a.y1 + cont_a.get_height() // 2

    r = cont.get_height() * 7 // 10

    child_cnt = cont.get_child_cnt()
    for i in range(child_cnt):
        child = cont.get_child(i)
        child_a = lv.area_t()
        child.get_coords(child_a)

        child_y_center = child_a.y1 + child_a.get_height() // 2

        diff_y = child_y_center - cont_y_center
        diff_y = abs(diff_y)

        # Get the x of diff_y on a circle.

        # If diff_y is out of the circle use the last point of the circle (the radius)
        if diff_y >= r:
            x = r
        else:
            # Use Pythagoras theorem to get x from radius and y
            x_sqr = r * r - diff_y * diff_y
            res = lv.sqrt_res_t()
```

(下页继续)

(续上页)

```
            lv.sqrt(x_sqr, res, 0x8000)   # Use lvgl's built in sqrt root function
            x = r - res.i

        # Translate the item by the calculated X coordinate
        child.set_style_translate_x(x, 0)

        # Use some opacity with larger translations
        opa = lv.map(x, 0, r, lv.OPA.TRANSP, lv.OPA.COVER)
        child.set_style_opa(lv.OPA.COVER - opa, 0)

#
# Translate the object as they scroll
#

cont = lv.obj(lv.scr_act())
cont.set_size(200, 200)
cont.center()
cont.set_flex_flow(lv.FLEX_FLOW.COLUMN)
cont.add_event_cb(scroll_event_cb, lv.EVENT.SCROLL, None)
cont.set_style_radius(lv.RADIUS.CIRCLE, 0)
cont.set_style_clip_corner(True, 0)
cont.set_scroll_dir(lv.DIR.VER)
cont.set_scroll_snap_y(lv.SCROLL_SNAP.CENTER)
cont.set_scrollbar_mode(lv.SCROLLBAR_MODE.OFF)

for i in range(20):
    btn = lv.btn(cont)
    btn.set_width(lv.pct(100))

    label = lv.label(btn)
    label.set_text("Button " + str(i))

    # Update the buttons position manually for first*
    lv.event_send(cont, lv.EVENT.SCROLL, None)

    # Be sure the fist button is in the middle
    #lv.obj.scroll_to_view(cont.get_child(0), lv.ANIM.OFF)
    cont.get_child(0).scroll_to_view(lv.ANIM.OFF)
```

2.2.7 Widgets

Base object

Base objects with custom styles

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES

void lv_example_obj_1(void)
{
    lv_obj_t * obj1;
    obj1 = lv_obj_create(lv_scr_act());
    lv_obj_set_size(obj1, 100, 50);
    lv_obj_align(obj1, LV_ALIGN_CENTER, -60, -30);

    static lv_style_t style_shadow;
    lv_style_init(&style_shadow);
    lv_style_set_shadow_width(&style_shadow, 10);
    lv_style_set_shadow_spread(&style_shadow, 5);
    lv_style_set_shadow_color(&style_shadow, lv_palette_main(LV_PALETTE_BLUE));

    lv_obj_t * obj2;
    obj2 = lv_obj_create(lv_scr_act());
    lv_obj_add_style(obj2, &style_shadow, 0);
    lv_obj_align(obj2, LV_ALIGN_CENTER, 60, 30);
}
#endif
```

```
obj1 = lv.obj(lv.scr_act())
obj1.set_size(100, 50)
obj1.align(lv.ALIGN.CENTER, -60, -30)

style_shadow = lv.style_t()
style_shadow.init()
style_shadow.set_shadow_width(10)
style_shadow.set_shadow_spread(5)
style_shadow.set_shadow_color(lv.palette_main(lv.PALETTE.BLUE))

obj2 = lv.obj(lv.scr_act())
obj2.add_style(style_shadow, 0)
obj2.align(lv.ALIGN.CENTER, 60, 30)
```

Make an object draggable

```c
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES

static void drag_event_handler(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target(e);

    lv_indev_t * indev = lv_indev_get_act();
    if(indev == NULL)  return;

    lv_point_t vect;
    lv_indev_get_vect(indev, &vect);

    lv_coord_t x = lv_obj_get_x(obj) + vect.x;
    lv_coord_t y = lv_obj_get_y(obj) + vect.y;
    lv_obj_set_pos(obj, x, y);
}


/**
 * Make an object dragable.
 */
void lv_example_obj_2(void)
{
    lv_obj_t * obj;
    obj = lv_obj_create(lv_scr_act());
    lv_obj_set_size(obj, 150, 100);
    lv_obj_add_event_cb(obj, drag_event_handler, LV_EVENT_PRESSING, NULL);

    lv_obj_t * label = lv_label_create(obj);
    lv_label_set_text(label, "Drag me");
    lv_obj_center(label);

}
#endif
```

```python
def drag_event_handler(e):

    obj = e.get_target()

    indev = lv.indev_get_act()

```

(下页继续)

(续上页)

```
    vect = lv.point_t()
    indev.get_vect(vect)
    x = obj.get_x() + vect.x
    y = obj.get_y() + vect.y
    obj.set_pos(x, y)


#
# Make an object dragable.
#

obj = lv.obj(lv.scr_act())
obj.set_size(150, 100)
obj.add_event_cb(drag_event_handler, lv.EVENT.PRESSING, None)

label = lv.label(obj)
label.set_text("Drag me")
label.center()
```

Arc

Simple Arc

```
#include "../../lv_examples.h"

#if LV_USE_ARC && LV_BUILD_EXAMPLES

void lv_example_arc_1(void)
{
  /*Create an Arc*/
  lv_obj_t * arc = lv_arc_create(lv_scr_act());
  lv_obj_set_size(arc, 150, 150);
  lv_arc_set_rotation(arc, 135);
  lv_arc_set_bg_angles(arc, 0, 270);
  lv_arc_set_value(arc, 40);
  lv_obj_center(arc);
}

#endif
```

```
# Create an Arc
arc = lv.arc(lv.scr_act())
arc.set_end_angle(200)
arc.set_size(150, 150)
arc.center()
```

Loader with Arc

```
#include "../../lv_examples.h"

#if LV_USE_ARC && LV_BUILD_EXAMPLES

static void set_angle(void * obj, int32_t v)
{
    lv_arc_set_value(obj, v);
}

/**
 * Create an arc which acts as a loader.
 */
void lv_example_arc_2(void)
{
  /*Create an Arc*/
  lv_obj_t * arc = lv_arc_create(lv_scr_act());
  lv_arc_set_rotation(arc, 270);
  lv_arc_set_bg_angles(arc, 0, 360);
  lv_obj_remove_style(arc, NULL, LV_PART_KNOB);   /*Be sure the knob is not␣
↪displayed*/
  lv_obj_clear_flag(arc, LV_OBJ_FLAG_CLICKABLE);  /*To not allow adjusting by click*/
  lv_obj_center(arc);

  lv_anim_t a;
  lv_anim_init(&a);
  lv_anim_set_var(&a, arc);
  lv_anim_set_exec_cb(&a, set_angle);
  lv_anim_set_time(&a, 1000);
  lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);    /*Just for the demo*/
  lv_anim_set_repeat_delay(&a, 500);
  lv_anim_set_values(&a, 0, 100);
  lv_anim_start(&a);
```

(下页继续)

(续上页)

```
}

#endif
```

```
#
# An `lv_timer` to call periodically to set the angles of the arc
#
class ArcLoader():
    def __init__(self):
        self.a = 270

    def arc_loader_cb(self,tim,arc):
        # print(tim,arc)
        self.a += 5

        arc.set_end_angle(self.a)

        if self.a >= 270 + 360:
            tim._del()


#
# Create an arc which acts as a loader.
#

# Create an Arc
arc = lv.arc(lv.scr_act())
arc.set_bg_angles(0, 360)
arc.set_angles(270, 270)
arc.center()

# create the loader
arc_loader = ArcLoader()

# Create an `lv_timer` to update the arc.

timer = lv.timer_create_basic()
timer.set_period(20)
timer.set_cb(lambda src: arc_loader.arc_loader_cb(timer,arc))
```

(下页继续)

(续上页)

```
```

Bar

Simple Bar

```
#include "../../lv_examples.h"
#if LV_USE_BAR && LV_BUILD_EXAMPLES

void lv_example_bar_1(void)
{
    lv_obj_t * bar1 = lv_bar_create(lv_scr_act());
    lv_obj_set_size(bar1, 200, 20);
    lv_obj_center(bar1);
    lv_bar_set_value(bar1, 70, LV_ANIM_OFF);
}

#endif
```

```
bar1 = lv.bar(lv.scr_act())
bar1.set_size(200, 20)
bar1.center()
bar1.set_value(70, lv.ANIM.OFF)

```

Styling a bar

```
#include "../../lv_examples.h"
#if LV_USE_BAR && LV_BUILD_EXAMPLES

/**
 * Example of styling the bar
 */
void lv_example_bar_2(void)
{
    static lv_style_t style_bg;
    static lv_style_t style_indic;

    lv_style_init(&style_bg);
    lv_style_set_border_color(&style_bg, lv_palette_main(LV_PALETTE_BLUE));
```

(下页继续)

(续上页)

```
    lv_style_set_border_width(&style_bg, 2);
    lv_style_set_pad_all(&style_bg, 6); /*To make the indicator smaller*/
    lv_style_set_radius(&style_bg, 6);
    lv_style_set_anim_time(&style_bg, 1000);

    lv_style_init(&style_indic);
    lv_style_set_bg_opa(&style_indic, LV_OPA_COVER);
    lv_style_set_bg_color(&style_indic, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_radius(&style_indic, 3);

    lv_obj_t * bar = lv_bar_create(lv_scr_act());
    lv_obj_remove_style_all(bar);  /*To have a clean start*/
    lv_obj_add_style(bar, &style_bg, 0);
    lv_obj_add_style(bar, &style_indic, LV_PART_INDICATOR);

    lv_obj_set_size(bar, 200, 20);
    lv_obj_center(bar);
    lv_bar_set_value(bar, 100, LV_ANIM_ON);
}

#endif
```

```
#
# Example of styling the bar
#
style_bg = lv.style_t()
style_indic = lv.style_t()

style_bg.init()
style_bg.set_border_color(lv.palette_main(lv.PALETTE.BLUE))
style_bg.set_border_width(2)
style_bg.set_pad_all(6)            # To make the indicator smaller
style_bg.set_radius(6)
style_bg.set_anim_time(1000)

style_indic.init()
style_indic.set_bg_opa(lv.OPA.COVER)
style_indic.set_bg_color(lv.palette_main(lv.PALETTE.BLUE))
style_indic.set_radius(3)

bar = lv.bar(lv.scr_act())
bar.remove_style_all()   # To have a clean start
bar.add_style(style_bg, 0)
```

(下页继续)

(续上页)

```
bar.add_style(style_indic, lv.PART.INDICATOR)

bar.set_size(200, 20)
bar.center()
bar.set_value(100, lv.ANIM.ON)
```

Temperature meter

```
#include "../../lv_examples.h"
#if LV_USE_BAR && LV_BUILD_EXAMPLES

static void set_temp(void * bar, int32_t temp)
{
    lv_bar_set_value(bar, temp, LV_ANIM_ON);
}

/**
 * A temperature meter example
 */
void lv_example_bar_3(void)
{
    static lv_style_t style_indic;

    lv_style_init(&style_indic);
    lv_style_set_bg_opa(&style_indic, LV_OPA_COVER);
    lv_style_set_bg_color(&style_indic, lv_palette_main(LV_PALETTE_RED));
    lv_style_set_bg_grad_color(&style_indic, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_bg_grad_dir(&style_indic, LV_GRAD_DIR_VER);

    lv_obj_t * bar = lv_bar_create(lv_scr_act());
    lv_obj_add_style(bar, &style_indic, LV_PART_INDICATOR);
    lv_obj_set_size(bar, 20, 200);
    lv_obj_center(bar);
    lv_bar_set_range(bar, -20, 40);

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_exec_cb(&a, set_temp);
    lv_anim_set_time(&a, 3000);
    lv_anim_set_playback_time(&a, 3000);
    lv_anim_set_var(&a, bar);
```

(下页继续)

(续上页)

```
    lv_anim_set_values(&a, -20, 40);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
    lv_anim_start(&a);
}


#endif
```

```
def set_temp(bar, temp):
    bar.set_value(temp, lv.ANIM.ON)

#
# A temperature meter example
#


style_indic = lv.style_t()

style_indic.init()
style_indic.set_bg_opa(lv.OPA.COVER)
style_indic.set_bg_color(lv.palette_main(lv.PALETTE.RED))
style_indic.set_bg_grad_color(lv.palette_main(lv.PALETTE.BLUE))
style_indic.set_bg_grad_dir(lv.GRAD_DIR.VER)

bar = lv.bar(lv.scr_act())
bar.add_style(style_indic, lv.PART.INDICATOR)
bar.set_size(20, 200)
bar.center()
bar.set_range(-20, 40)

a = lv.anim_t()
a.init()
a.set_time(3000)
a.set_playback_time(3000)
a.set_var(bar)
a.set_values(-20, 40)
a.set_repeat_count(lv.ANIM_REPEAT.INFINITE)
a.set_custom_exec_cb(lambda a, val: set_temp(bar,val))
lv.anim_t.start(a)
```

Stripe pattern and range value

```
#include "../../lv_examples.h"
#if LV_USE_BAR && LV_BUILD_EXAMPLES

/**
 * Bar with stripe pattern and ranged value
 */
void lv_example_bar_4(void)
{
    LV_IMG_DECLARE(img_skew_strip);
    static lv_style_t style_indic;

    lv_style_init(&style_indic);
    lv_style_set_bg_img_src(&style_indic, &img_skew_strip);
    lv_style_set_bg_img_tiled(&style_indic, true);
    lv_style_set_bg_img_opa(&style_indic, LV_OPA_30);

    lv_obj_t * bar = lv_bar_create(lv_scr_act());
    lv_obj_add_style(bar, &style_indic, LV_PART_INDICATOR);

    lv_obj_set_size(bar, 260, 20);
    lv_obj_center(bar);
    lv_bar_set_mode(bar, LV_BAR_MODE_RANGE);
    lv_bar_set_value(bar, 90, LV_ANIM_OFF);
    lv_bar_set_start_value(bar, 20, LV_ANIM_OFF);
}

#endif
```

```
#
# get an icon
#
def get_icon(filename,xres,yres):
    try:
        sdl_filename = "../../assets/" + filename + "_" + str(xres) + "x" + str(yres)␣
↪+ "_argb8888.fnt"
        print("file name: ", sdl_filename)
        with open(sdl_filename,'rb') as f:
            icon_data = f.read()
    except:
        print("Could not find image file: " + filename)
        return None
```

(下页继续)

(续上页)

```
    icon_dsc = lv.img_dsc_t(
        {
            "header": {"always_zero": 0, "w": xres, "h": yres, "cf": lv.img.CF.TRUE_
↪COLOR_ALPHA},
            "data": icon_data,
            "data_size": len(icon_data),
        }
    )
    return icon_dsc

#
# Bar with stripe pattern and ranged value
#

img_skew_strip_dsc = get_icon("img_skew_strip",80,20)
style_indic = lv.style_t()

style_indic.init()
style_indic.set_bg_img_src(img_skew_strip_dsc)
style_indic.set_bg_img_tiled(True)
style_indic.set_bg_img_opa(lv.OPA._30)

bar = lv.bar(lv.scr_act())
bar.add_style(style_indic, lv.PART.INDICATOR)

bar.set_size(260, 20)
bar.center()
bar.set_mode(lv.bar.MODE.RANGE)
bar.set_value(90, lv.ANIM.OFF)
bar.set_start_value(20, lv.ANIM.OFF)
```

Bar with LTR and RTL base direction

```
#include "../../lv_examples.h"
#if LV_USE_BAR && LV_BUILD_EXAMPLES

/**
 * Bar with LTR and RTL base direction
 */
```

(下页继续)

(续上页)

```c
void lv_example_bar_5(void)
{
    lv_obj_t * label;


    lv_obj_t * bar_ltr = lv_bar_create(lv_scr_act());
    lv_obj_set_size(bar_ltr, 200, 20);
    lv_bar_set_value(bar_ltr, 70, LV_ANIM_OFF);
    lv_obj_align(bar_ltr, LV_ALIGN_CENTER, 0, -30);

    label = lv_label_create(lv_scr_act());
    lv_label_set_text(label, "Left to Right base direction");
    lv_obj_align_to(label, bar_ltr, LV_ALIGN_OUT_TOP_MID, 0, -5);

    lv_obj_t * bar_rtl = lv_bar_create(lv_scr_act());
    lv_obj_set_style_base_dir(bar_rtl, LV_BASE_DIR_RTL, 0);
    lv_obj_set_size(bar_rtl, 200, 20);
    lv_bar_set_value(bar_rtl, 70, LV_ANIM_OFF);
    lv_obj_align(bar_rtl, LV_ALIGN_CENTER, 0, 30);

    label = lv_label_create(lv_scr_act());
    lv_label_set_text(label, "Right to Left base direction");
    lv_obj_align_to(label, bar_rtl, LV_ALIGN_OUT_TOP_MID, 0, -5);
}

#endif
```

```python
#
# Bar with LTR and RTL base direction
#

bar_ltr = lv.bar(lv.scr_act())
bar_ltr.set_size(200, 20)
bar_ltr.set_value(70, lv.ANIM.OFF)
bar_ltr.align(lv.ALIGN.CENTER, 0, -30)

label = lv.label(lv.scr_act())
label.set_text("Left to Right base direction")
label.align_to(bar_ltr, lv.ALIGN.OUT_TOP_MID, 0, -5)

bar_rtl = lv.bar(lv.scr_act())
bar_rtl.set_style_base_dir(lv.BASE_DIR.RTL,0)
bar_rtl.set_size(200, 20)
```

(下页继续)

(续上页)

```
bar_rtl.set_value(70, lv.ANIM.OFF)
bar_rtl.align(lv.ALIGN.CENTER, 0, 30)

label = lv.label(lv.scr_act())
label.set_text("Right to Left base direction")
label.align_to(bar_rtl, lv.ALIGN.OUT_TOP_MID, 0, -5)
```

Custom drawer to show the current value

```
#include "../../lv_examples.h"
#if LV_USE_BAR && LV_BUILD_EXAMPLES

static void set_value(void *bar, int32_t v)
{
    lv_bar_set_value(bar, v, LV_ANIM_OFF);
}

static void event_cb(lv_event_t * e)
{
    lv_obj_draw_part_dsc_t * dsc = lv_event_get_param(e);
    if(dsc->part != LV_PART_INDICATOR) return;

    lv_obj_t * obj= lv_event_get_target(e);

    lv_draw_label_dsc_t label_dsc;
    lv_draw_label_dsc_init(&label_dsc);
    label_dsc.font = LV_FONT_DEFAULT;

    char buf[8];
    lv_snprintf(buf, sizeof(buf), "%d", (int)lv_bar_get_value(obj));

    lv_point_t txt_size;
    lv_txt_get_size(&txt_size, buf, label_dsc.font, label_dsc.letter_space, label_dsc.
↪line_space, LV_COORD_MAX, label_dsc.flag);

    lv_area_t txt_area;
    /*If the indicator is long enough put the text inside on the right*/
    if(lv_area_get_width(dsc->draw_area) > txt_size.x + 20) {
        txt_area.x2 = dsc->draw_area->x2 - 5;
        txt_area.x1 = txt_area.x2 - txt_size.x + 1;
        label_dsc.color = lv_color_white();
    }
```

(下页继续)

(续上页)

```
    /*If the indicator is still short put the text out of it on the right*/
    else {
        txt_area.x1 = dsc->draw_area->x2 + 5;
        txt_area.x2 = txt_area.x1 + txt_size.x - 1;
        label_dsc.color = lv_color_black();
    }

    txt_area.y1 = dsc->draw_area->y1 + (lv_area_get_height(dsc->draw_area) - txt_size.
↪y) / 2;
    txt_area.y2 = txt_area.y1 + txt_size.y - 1;

    lv_draw_label(dsc->draw_ctx, &label_dsc, &txt_area, buf, NULL);
}

/**
 * Custom drawer on the bar to display the current value
 */
void lv_example_bar_6(void)
{
    lv_obj_t * bar = lv_bar_create(lv_scr_act());
    lv_obj_add_event_cb(bar, event_cb, LV_EVENT_DRAW_PART_END, NULL);
    lv_obj_set_size(bar, 200, 20);
    lv_obj_center(bar);

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_var(&a, bar);
    lv_anim_set_values(&a, 0, 100);
    lv_anim_set_exec_cb(&a, set_value);
    lv_anim_set_time(&a, 2000);
    lv_anim_set_playback_time(&a, 2000);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
    lv_anim_start(&a);

}

#endif
```

```
def set_value(bar, v):
    bar.set_value(v, lv.ANIM.OFF)

def event_cb(e):
    dsc = lv.obj_draw_part_dsc_t.__cast__(e.get_param())
```

(下页继续)

(续上页)

```
    if dsc.part != lv.PART.INDICATOR:
        return

    obj= e.get_target()

    label_dsc = lv.draw_label_dsc_t()
    label_dsc.init()
    # label_dsc.font = LV_FONT_DEFAULT;

    value_txt = str(obj.get_value())
    txt_size = lv.point_t()
    lv.txt_get_size(txt_size, value_txt, label_dsc.font, label_dsc.letter_space,␣
↪label_dsc.line_space, lv.COORD.MAX, label_dsc.flag)

    txt_area = lv.area_t()
    # If the indicator is long enough put the text inside on the right
    if dsc.draw_area.get_width() > txt_size.x + 20:
        txt_area.x2 = dsc.draw_area.x2 - 5
        txt_area.x1 = txt_area.x2 - txt_size.x + 1
        label_dsc.color = lv.color_white()
    # If the indicator is still short put the text out of it on the right*/
    else:
        txt_area.x1 = dsc.draw_area.x2 + 5
        txt_area.x2 = txt_area.x1 + txt_size.x - 1
        label_dsc.color = lv.color_black()

    txt_area.y1 = dsc.draw_area.y1 + (dsc.draw_area.get_height() - txt_size.y) // 2
    txt_area.y2 = txt_area.y1 + txt_size.y - 1

    dsc.draw_ctx.label(label_dsc, txt_area, value_txt, None)

#
# Custom drawer on the bar to display the current value
#

bar = lv.bar(lv.scr_act())
bar.add_event_cb(event_cb, lv.EVENT.DRAW_PART_END, None)
bar.set_size(200, 20)
bar.center()

a = lv.anim_t()
a.init()
a.set_var(bar)
```

(下页继续)

(续上页)

```
a.set_values(0, 100)
a.set_custom_exec_cb(lambda a,val: set_value(bar,val))
a.set_time(2000)
a.set_playback_time(2000)
a.set_repeat_count(lv.ANIM_REPEAT.INFINITE)
lv.anim_t.start(a)
```

Button

Simple Buttons

```
#include "../../lv_examples.h"
#if LV_USE_BTN && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);

    if(code == LV_EVENT_CLICKED) {
        LV_LOG_USER("Clicked");
    }
    else if(code == LV_EVENT_VALUE_CHANGED) {
        LV_LOG_USER("Toggled");
    }
}

void lv_example_btn_1(void)
{
    lv_obj_t * label;

    lv_obj_t * btn1 = lv_btn_create(lv_scr_act());
    lv_obj_add_event_cb(btn1, event_handler, LV_EVENT_ALL, NULL);
    lv_obj_align(btn1, LV_ALIGN_CENTER, 0, -40);

    label = lv_label_create(btn1);
    lv_label_set_text(label, "Button");
    lv_obj_center(label);

    lv_obj_t * btn2 = lv_btn_create(lv_scr_act());
    lv_obj_add_event_cb(btn2, event_handler, LV_EVENT_ALL, NULL);
    lv_obj_align(btn2, LV_ALIGN_CENTER, 0, 40);
```

(下页继续)

(续上页)

```
    lv_obj_add_flag(btn2, LV_OBJ_FLAG_CHECKABLE);
    lv_obj_set_height(btn2, LV_SIZE_CONTENT);

    label = lv_label_create(btn2);
    lv_label_set_text(label, "Toggle");
    lv_obj_center(label);
}
#endif
```

```
def event_handler(evt):
    code = evt.get_code()

    if code == lv.EVENT.CLICKED:
            print("Clicked event seen")
    elif code == lv.EVENT.VALUE_CHANGED:
        print("Value changed seen")

# create a simple button
btn1 = lv.btn(lv.scr_act())

# attach the callback
btn1.add_event_cb(event_handler,lv.EVENT.ALL, None)

btn1.align(lv.ALIGN.CENTER,0,-40)
label=lv.label(btn1)
label.set_text("Button")

# create a toggle button
btn2 = lv.btn(lv.scr_act())

# attach the callback
#btn2.add_event_cb(event_handler,lv.EVENT.VALUE_CHANGED,None)
btn2.add_event_cb(event_handler,lv.EVENT.ALL, None)

btn2.align(lv.ALIGN.CENTER,0,40)
btn2.add_flag(lv.obj.FLAG.CHECKABLE)
btn2.set_height(lv.SIZE.CONTENT)

label=lv.label(btn2)
label.set_text("Toggle")
label.center()
```

Styling buttons

```c
#include "../../lv_examples.h"
#if LV_USE_BTN && LV_BUILD_EXAMPLES

/**
 * Style a button from scratch
 */
void lv_example_btn_2(void)
{
    /*Init the style for the default state*/
    static lv_style_t style;
    lv_style_init(&style);

    lv_style_set_radius(&style, 3);

    lv_style_set_bg_opa(&style, LV_OPA_100);
    lv_style_set_bg_color(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_bg_grad_color(&style, lv_palette_darken(LV_PALETTE_BLUE, 2));
    lv_style_set_bg_grad_dir(&style, LV_GRAD_DIR_VER);

    lv_style_set_border_opa(&style, LV_OPA_40);
    lv_style_set_border_width(&style, 2);
    lv_style_set_border_color(&style, lv_palette_main(LV_PALETTE_GREY));

    lv_style_set_shadow_width(&style, 8);
    lv_style_set_shadow_color(&style, lv_palette_main(LV_PALETTE_GREY));
    lv_style_set_shadow_ofs_y(&style, 8);

    lv_style_set_outline_opa(&style, LV_OPA_COVER);
    lv_style_set_outline_color(&style, lv_palette_main(LV_PALETTE_BLUE));

    lv_style_set_text_color(&style, lv_color_white());
    lv_style_set_pad_all(&style, 10);

    /*Init the pressed style*/
    static lv_style_t style_pr;
    lv_style_init(&style_pr);

    /*Add a large outline when pressed*/
    lv_style_set_outline_width(&style_pr, 30);
    lv_style_set_outline_opa(&style_pr, LV_OPA_TRANSP);

    lv_style_set_translate_y(&style_pr, 5);
```

(下页继续)

(续上页)

```
    lv_style_set_shadow_ofs_y(&style_pr, 3);
    lv_style_set_bg_color(&style_pr, lv_palette_darken(LV_PALETTE_BLUE, 2));
    lv_style_set_bg_grad_color(&style_pr, lv_palette_darken(LV_PALETTE_BLUE, 4));

    /*Add a transition to the outline*/
    static lv_style_transition_dsc_t trans;
    static lv_style_prop_t props[] = {LV_STYLE_OUTLINE_WIDTH, LV_STYLE_OUTLINE_OPA, 0}
↪;
    lv_style_transition_dsc_init(&trans, props, lv_anim_path_linear, 300, 0, NULL);

    lv_style_set_transition(&style_pr, &trans);

    lv_obj_t * btn1 = lv_btn_create(lv_scr_act());
    lv_obj_remove_style_all(btn1);                          /*Remove the style coming␣
↪from the theme*/
    lv_obj_add_style(btn1, &style, 0);
    lv_obj_add_style(btn1, &style_pr, LV_STATE_PRESSED);
    lv_obj_set_size(btn1, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
    lv_obj_center(btn1);

    lv_obj_t * label = lv_label_create(btn1);
    lv_label_set_text(label, "Button");
    lv_obj_center(label);
}
#endif
```

```
#
# Style a button from scratch
#

# Init the style for the default state
style = lv.style_t()
style.init()

style.set_radius(3)

style.set_bg_opa(lv.OPA.COVER)
style.set_bg_color(lv.palette_main(lv.PALETTE.BLUE))
style.set_bg_grad_color(lv.palette_darken(lv.PALETTE.BLUE, 2))
style.set_bg_grad_dir(lv.GRAD_DIR.VER)

style.set_border_opa(lv.OPA._40)
style.set_border_width(2)
```

(下页继续)

(续上页)

```
style.set_border_color(lv.palette_main(lv.PALETTE.GREY))

style.set_shadow_width(8)
style.set_shadow_color(lv.palette_main(lv.PALETTE.GREY))
style.set_shadow_ofs_y(8)

style.set_outline_opa(lv.OPA.COVER)
style.set_outline_color(lv.palette_main(lv.PALETTE.BLUE))

style.set_text_color(lv.color_white())
style.set_pad_all(10)

# Init the pressed style
style_pr = lv.style_t()
style_pr.init()

# Add a large outline when pressed
style_pr.set_outline_width(30)
style_pr.set_outline_opa(lv.OPA.TRANSP)

style_pr.set_translate_y(5)
style_pr.set_shadow_ofs_y(3)
style_pr.set_bg_color(lv.palette_darken(lv.PALETTE.BLUE, 2))
style_pr.set_bg_grad_color(lv.palette_darken(lv.PALETTE.BLUE, 4))

# Add a transition to the outline
trans = lv.style_transition_dsc_t()
props = [lv.STYLE.OUTLINE_WIDTH, lv.STYLE.OUTLINE_OPA, 0]
trans.init(props, lv.anim_t.path_linear, 300, 0, None)

style_pr.set_transition(trans)

btn1 = lv.btn(lv.scr_act())
btn1.remove_style_all()                         # Remove the style coming from the␣
↪theme
btn1.add_style(style, 0)
btn1.add_style(style_pr, lv.STATE.PRESSED)
btn1.set_size(lv.SIZE.CONTENT, lv.SIZE.CONTENT)
btn1.center()

label = lv.label(btn1)
label.set_text("Button")
label.center()
```

Gummy button

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_BTN

/**
 * Create a style transition on a button to act like a gum when clicked
 */
void lv_example_btn_3(void)
{
    /*Properties to transition*/
    static lv_style_prop_t props[] = {
            LV_STYLE_TRANSFORM_WIDTH, LV_STYLE_TRANSFORM_HEIGHT, LV_STYLE_TEXT_LETTER_
↪SPACE, 0
    };

    /*Transition descriptor when going back to the default state.
     *Add some delay to be sure the press transition is visible even if the press was␣
↪very short*/
    static lv_style_transition_dsc_t transition_dsc_def;
    lv_style_transition_dsc_init(&transition_dsc_def, props, lv_anim_path_overshoot,␣
↪250, 100, NULL);

    /*Transition descriptor when going to pressed state.
     *No delay, go to presses state immediately*/
    static lv_style_transition_dsc_t transition_dsc_pr;
    lv_style_transition_dsc_init(&transition_dsc_pr, props, lv_anim_path_ease_in_out,␣
↪250, 0, NULL);

    /*Add only the new transition to he default state*/
    static lv_style_t style_def;
    lv_style_init(&style_def);
    lv_style_set_transition(&style_def, &transition_dsc_def);

    /*Add the transition and some transformation to the presses state.*/
    static lv_style_t style_pr;
    lv_style_init(&style_pr);
    lv_style_set_transform_width(&style_pr, 10);
    lv_style_set_transform_height(&style_pr, -10);
    lv_style_set_text_letter_space(&style_pr, 10);
    lv_style_set_transition(&style_pr, &transition_dsc_pr);

    lv_obj_t * btn1 = lv_btn_create(lv_scr_act());
    lv_obj_align(btn1, LV_ALIGN_CENTER, 0, -80);
```

(下页继续)

(续上页)

```
    lv_obj_add_style(btn1, &style_pr, LV_STATE_PRESSED);
    lv_obj_add_style(btn1, &style_def, 0);

    lv_obj_t * label = lv_label_create(btn1);
    lv_label_set_text(label, "Gum");
}
#endif
```

```
#
# Create a style transition on a button to act like a gum when clicked
#

# Properties to transition
props = [lv.STYLE.TRANSFORM_WIDTH, lv.STYLE.TRANSFORM_HEIGHT, lv.STYLE.TEXT_LETTER_
↪SPACE, 0]

# Transition descriptor when going back to the default state.
# Add some delay to be sure the press transition is visible even if the press was␣
↪very short*/
transition_dsc_def = lv.style_transition_dsc_t()
transition_dsc_def.init(props, lv.anim_t.path_overshoot, 250, 100, None)

# Transition descriptor when going to pressed state.
# No delay, go to pressed state immediately
transition_dsc_pr = lv.style_transition_dsc_t()
transition_dsc_pr.init(props, lv.anim_t.path_ease_in_out, 250, 0, None)

# Add only the new transition to the default state
style_def = lv.style_t()
style_def.init()
style_def.set_transition(transition_dsc_def)

# Add the transition and some transformation to the presses state.
style_pr = lv.style_t()
style_pr.init()
style_pr.set_transform_width(10)
style_pr.set_transform_height(-10)
style_pr.set_text_letter_space(10)
style_pr.set_transition(transition_dsc_pr)

btn1 = lv.btn(lv.scr_act())
btn1.align(lv.ALIGN.CENTER, 0, -80)
btn1.add_style(style_pr, lv.STATE.PRESSED)
```

(下页继续)

(续上页)

```
btn1.add_style(style_def, 0)

label = lv.label(btn1)
label.set_text("Gum")
```

Button matrix

Simple Button matrix

```
#include "../../lv_examples.h"
#if LV_USE_BTNMATRIX && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        uint32_t id = lv_btnmatrix_get_selected_btn(obj);
        const char * txt = lv_btnmatrix_get_btn_text(obj, id);

        LV_LOG_USER("%s was pressed\n", txt);
    }
}


static const char * btnm_map[] = {"1", "2", "3", "4", "5", "\n",
                                  "6", "7", "8", "9", "0", "\n",
                                  "Action1", "Action2", ""};

void lv_example_btnmatrix_1(void)
{
    lv_obj_t * btnm1 = lv_btnmatrix_create(lv_scr_act());
    lv_btnmatrix_set_map(btnm1, btnm_map);
    lv_btnmatrix_set_btn_width(btnm1, 10, 2);        /*Make "Action1" twice as wide␣
↪as "Action2"*/
    lv_btnmatrix_set_btn_ctrl(btnm1, 10, LV_BTNMATRIX_CTRL_CHECKABLE);
    lv_btnmatrix_set_btn_ctrl(btnm1, 11, LV_BTNMATRIX_CTRL_CHECKED);
    lv_obj_align(btnm1, LV_ALIGN_CENTER, 0, 0);
    lv_obj_add_event_cb(btnm1, event_handler, LV_EVENT_ALL, NULL);
}
```

(下页继续)

(续上页)

```
#endif
```

```
def event_handler(evt):
    code = evt.get_code()
    obj  = evt.get_target()

    if code == lv.EVENT.VALUE_CHANGED :
        id = obj.get_selected_btn()
        txt = obj.get_btn_text(id)

        print("%s was pressed"%txt)

btnm_map = ["1", "2", "3", "4", "5", "\n",
            "6", "7", "8", "9", "0", "\n",
            "Action1", "Action2", ""]

btnm1 = lv.btnmatrix(lv.scr_act())
btnm1.set_map(btnm_map)
btnm1.set_btn_width(10, 2)        # Make "Action1" twice as wide as "Action2"
btnm1.set_btn_ctrl(10, lv.btnmatrix.CTRL.CHECKABLE)
btnm1.set_btn_ctrl(11, lv.btnmatrix.CTRL.CHECKED)
btnm1.align(lv.ALIGN.CENTER, 0, 0)
btnm1.add_event_cb(event_handler, lv.EVENT.ALL, None)


#endif
```

Custom buttons

```
#include "../../lv_examples.h"
#if LV_USE_BTNMATRIX && LV_BUILD_EXAMPLES


static void event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target(e);
    if(code == LV_EVENT_DRAW_PART_BEGIN) {
        lv_obj_draw_part_dsc_t * dsc = lv_event_get_param(e);

        /*Change the draw descriptor the 2nd button*/
        if(dsc->id == 1) {
```

(下页继续)

(续上页)

```
            dsc->rect_dsc->radius = 0;
            if(lv_btnmatrix_get_selected_btn(obj) == dsc->id)  dsc->rect_dsc->bg_
↪color = lv_palette_darken(LV_PALETTE_BLUE, 3);
            else dsc->rect_dsc->bg_color = lv_palette_main(LV_PALETTE_BLUE);

            dsc->rect_dsc->shadow_width = 6;
            dsc->rect_dsc->shadow_ofs_x = 3;
            dsc->rect_dsc->shadow_ofs_y = 3;
            dsc->label_dsc->color = lv_color_white();
        }
        /*Change the draw descriptor the 3rd button*/
        else if(dsc->id == 2) {
            dsc->rect_dsc->radius = LV_RADIUS_CIRCLE;
            if(lv_btnmatrix_get_selected_btn(obj) == dsc->id)  dsc->rect_dsc->bg_
↪color = lv_palette_darken(LV_PALETTE_RED, 3);
            else dsc->rect_dsc->bg_color = lv_palette_main(LV_PALETTE_RED);

            dsc->label_dsc->color = lv_color_white();
        }
        else if(dsc->id == 3) {
            dsc->label_dsc->opa = LV_OPA_TRANSP; /*Hide the text if any*/

        }
    }
    if(code == LV_EVENT_DRAW_PART_END) {
        lv_obj_draw_part_dsc_t * dsc = lv_event_get_param(e);

        /*Add custom content to the 4th button when the button itself was drawn*/
        if(dsc->id == 3) {
            LV_IMG_DECLARE(img_star);
            lv_img_header_t header;
            lv_res_t res = lv_img_decoder_get_info(&img_star, &header);
            if(res != LV_RES_OK) return;

            lv_area_t a;
            a.x1 = dsc->draw_area->x1 + (lv_area_get_width(dsc->draw_area) - header.
↪w) / 2;
            a.x2 = a.x1 + header.w - 1;
            a.y1 = dsc->draw_area->y1 + (lv_area_get_height(dsc->draw_area) - header.
↪h) / 2;
            a.y2 = a.y1 + header.h - 1;

            lv_draw_img_dsc_t img_draw_dsc;
```

(下页继续)

(续上页)

```
            lv_draw_img_dsc_init(&img_draw_dsc);
            img_draw_dsc.recolor = lv_color_black();
            if(lv_btnmatrix_get_selected_btn(obj) == dsc->id)  img_draw_dsc.recolor_
↪opa = LV_OPA_30;

            lv_draw_img(dsc->draw_ctx, &img_draw_dsc, &a, &img_star);
        }
    }
}

/**
 * Add custom drawer to the button matrix to customize buttons one by one
 */
void lv_example_btnmatrix_2(void)
{
    lv_obj_t * btnm = lv_btnmatrix_create(lv_scr_act());
    lv_obj_add_event_cb(btnm, event_cb, LV_EVENT_ALL, NULL);
    lv_obj_center(btnm);
}

#endif
```

```
from imagetools import get_png_info, open_png

# Register PNG image decoder
decoder = lv.img.decoder_create()
decoder.info_cb = get_png_info
decoder.open_cb = open_png

# Create an image from the png file
try:
    with open('../../assets/img_star.png','rb') as f:
        png_data = f.read()
except:
    print("Could not find star.png")
    sys.exit()

img_star_argb = lv.img_dsc_t({
  'data_size': len(png_data),
  'data': png_data
})

def event_cb(e):
```

(下页继续)

(续上页)

```
    code = e.get_code()
    obj = e.get_target()
    dsc = lv.obj_draw_part_dsc_t.__cast__(e.get_param())
    if code == lv.EVENT.DRAW_PART_BEGIN:
        # Change the draw descriptor the 2nd button
        if dsc.id == 1:
            dsc.rect_dsc.radius = 0
            if obj.get_selected_btn() == dsc.id:
                dsc.rect_dsc.bg_color = lv.palette_darken(lv.PALETTE.GREY, 3)
            else:
                dsc.rect_dsc.bg_color = lv.palette_main(lv.PALETTE.BLUE)

            dsc.rect_dsc.shadow_width = 6
            dsc.rect_dsc.shadow_ofs_x = 3
            dsc.rect_dsc.shadow_ofs_y = 3
            dsc.label_dsc.color = lv.color_white()

        # Change the draw descriptor the 3rd button

        elif dsc.id == 2:
            dsc.rect_dsc.radius = lv.RADIUS.CIRCLE
            if obj.get_selected_btn() == dsc.id:
                dsc.rect_dsc.bg_color = lv.palette_darken(lv.PALETTE.RED, 3)
            else:
                dsc.rect_dsc.bg_color = lv.palette_main(lv.PALETTE.RED)

                dsc.label_dsc.color = lv.color_white()
        elif dsc.id == 3:
            dsc.label_dsc.opa = lv.OPA.TRANSP  # Hide the text if any

    if code == lv.EVENT.DRAW_PART_END:
        # Add custom content to the 4th button when the button itself was drawn
        if dsc.id == 3:
            # LV_IMG_DECLARE(img_star)
            header = lv.img_header_t()
            res = lv.img.decoder_get_info(img_star_argb, header)
            if res != lv.RES.OK:
                print("error when getting image header")
                return
            else:
                a = lv.area_t()
                a.x1 = dsc.draw_area.x1 + (dsc.draw_area.get_width() - header.w) // 2
                a.x2 = a.x1 + header.w - 1
```

(下页继续)

(续上页)

```
                a.y1 = dsc.draw_area.y1 + (dsc.draw_area.get_height() - header.h) // 2
                a.y2 = a.y1 + header.h - 1
                img_draw_dsc = lv.draw_img_dsc_t()
                img_draw_dsc.init()
                img_draw_dsc.recolor = lv.color_black()
                if obj.get_selected_btn() == dsc.id:
                    img_draw_dsc.recolor_opa = lv.OPA._30

                dsc.draw_ctx.img(img_draw_dsc, a, img_star_argb)

#
# Add custom drawer to the button matrix to c
#
btnm = lv.btnmatrix(lv.scr_act())
btnm.add_event_cb(event_cb, lv.EVENT.ALL, None)
btnm.center()
```

Pagination

```
#include "../../lv_examples.h"
#if LV_USE_BTNMATRIX  && LV_BUILD_EXAMPLES

static void event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target(e);
    uint32_t id = lv_btnmatrix_get_selected_btn(obj);
    bool prev = id == 0 ? true : false;
    bool next = id == 6 ? true : false;
    if(prev || next) {
        /*Find the checked button*/
        uint32_t i;
        for(i = 1; i < 7; i++) {
            if(lv_btnmatrix_has_btn_ctrl(obj, i, LV_BTNMATRIX_CTRL_CHECKED)) break;
        }

        if(prev && i > 1) i--;
        else if(next && i < 5) i++;

        lv_btnmatrix_set_btn_ctrl(obj, i, LV_BTNMATRIX_CTRL_CHECKED);
    }
}
```

(下页继续)

(续上页)

```
/**
 * Make a button group (pagination)
 */
void lv_example_btnmatrix_3(void)
{
    static lv_style_t style_bg;
    lv_style_init(&style_bg);
    lv_style_set_pad_all(&style_bg, 0);
    lv_style_set_pad_gap(&style_bg, 0);
    lv_style_set_clip_corner(&style_bg, true);
    lv_style_set_radius(&style_bg, LV_RADIUS_CIRCLE);
    lv_style_set_border_width(&style_bg, 0);


    static lv_style_t style_btn;
    lv_style_init(&style_btn);
    lv_style_set_radius(&style_btn, 0);
    lv_style_set_border_width(&style_btn, 1);
    lv_style_set_border_opa(&style_btn, LV_OPA_50);
    lv_style_set_border_color(&style_btn, lv_palette_main(LV_PALETTE_GREY));
    lv_style_set_border_side(&style_btn, LV_BORDER_SIDE_INTERNAL);
    lv_style_set_radius(&style_btn, 0);

    static const char * map[] = {LV_SYMBOL_LEFT, "1", "2", "3", "4", "5", LV_SYMBOL_
↪RIGHT, ""};

    lv_obj_t * btnm = lv_btnmatrix_create(lv_scr_act());
    lv_btnmatrix_set_map(btnm, map);
    lv_obj_add_style(btnm, &style_bg, 0);
    lv_obj_add_style(btnm, &style_btn, LV_PART_ITEMS);
    lv_obj_add_event_cb(btnm, event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    lv_obj_set_size(btnm, 225, 35);

    /*Allow selecting on one number at time*/
    lv_btnmatrix_set_btn_ctrl_all(btnm, LV_BTNMATRIX_CTRL_CHECKABLE);
    lv_btnmatrix_clear_btn_ctrl(btnm, 0, LV_BTNMATRIX_CTRL_CHECKABLE);
    lv_btnmatrix_clear_btn_ctrl(btnm, 6, LV_BTNMATRIX_CTRL_CHECKABLE);

    lv_btnmatrix_set_one_checked(btnm, true);
    lv_btnmatrix_set_btn_ctrl(btnm, 1, LV_BTNMATRIX_CTRL_CHECKED);

    lv_obj_center(btnm);
```

(下页继续)

(续上页)

```
}

#endif
```

```
def event_cb(e):
    obj = e.get_target()
    id = obj.get_selected_btn()
    if id == 0:
        prev = True
    else:
        prev = False
    if id == 6:
        next = True
    else:
        next = False
    if prev or next:
        # Find the checked butto
        for i in range(7):
            if obj.has_btn_ctrl(i, lv.btnmatrix.CTRL.CHECKED):
                break
        if prev and i > 1:
            i-=1
        elif next and i < 5:
            i+=1

        obj.set_btn_ctrl(i, lv.btnmatrix.CTRL.CHECKED)

#
# Make a button group
#

style_bg = lv.style_t()
style_bg.init()
style_bg.set_pad_all(0)
style_bg.set_pad_gap(0)
style_bg.set_clip_corner(True)
style_bg.set_radius(lv.RADIUS.CIRCLE)
style_bg.set_border_width(0)


style_btn = lv.style_t()
style_btn.init()
```

(下页继续)

(续上页)

```
style_btn.set_radius(0)
style_btn.set_border_width(1)
style_btn.set_border_opa(lv.OPA._50)
style_btn.set_border_color(lv.palette_main(lv.PALETTE.GREY))
style_btn.set_border_side(lv.BORDER_SIDE.INTERNAL)
style_btn.set_radius(0)

map = [lv.SYMBOL.LEFT,"1","2", "3", "4", "5",lv.SYMBOL.RIGHT, ""]

btnm = lv.btnmatrix(lv.scr_act())
btnm.set_map(map)
btnm.add_style(style_bg, 0)
btnm.add_style(style_btn, lv.PART.ITEMS)
btnm.add_event_cb(event_cb, lv.EVENT.VALUE_CHANGED, None)
btnm.set_size(225, 35)

# Allow selecting on one number at time
btnm.set_btn_ctrl_all(lv.btnmatrix.CTRL.CHECKABLE)
btnm.clear_btn_ctrl(0, lv.btnmatrix.CTRL.CHECKABLE)
btnm.clear_btn_ctrl(6, lv.btnmatrix.CTRL.CHECKABLE)

btnm.set_one_checked(True)
btnm.set_btn_ctrl(1, lv.btnmatrix.CTRL.CHECKED)

btnm.center()
```

Calendar

Calendar with header

```
#include "../../lv_examples.h"
#if LV_USE_CALENDAR && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_current_target(e);

    if(code == LV_EVENT_VALUE_CHANGED) {
        lv_calendar_date_t date;
        if(lv_calendar_get_pressed_date(obj, &date)) {
```

(下页继续)

(续上页)

```
            LV_LOG_USER("Clicked date: %02d.%02d.%d", date.day, date.month, date.
↪year);
        }
    }
}

void lv_example_calendar_1(void)
{
    lv_obj_t  * calendar = lv_calendar_create(lv_scr_act());
    lv_obj_set_size(calendar, 185, 185);
    lv_obj_align(calendar, LV_ALIGN_CENTER, 0, 27);
    lv_obj_add_event_cb(calendar, event_handler, LV_EVENT_ALL, NULL);

    lv_calendar_set_today_date(calendar, 2021, 02, 23);
    lv_calendar_set_showed_date(calendar, 2021, 02);

    /*Highlight a few days*/
    static lv_calendar_date_t highlighted_days[3];       /*Only its pointer will be␣
↪saved so should be static*/
    highlighted_days[0].year = 2021;
    highlighted_days[0].month = 02;
    highlighted_days[0].day = 6;

    highlighted_days[1].year = 2021;
    highlighted_days[1].month = 02;
    highlighted_days[1].day = 11;

    highlighted_days[2].year = 2022;
    highlighted_days[2].month = 02;
    highlighted_days[2].day = 22;

    lv_calendar_set_highlighted_dates(calendar, highlighted_days, 3);

#if LV_USE_CALENDAR_HEADER_DROPDOWN
    lv_calendar_header_dropdown_create(calendar);
#elif LV_USE_CALENDAR_HEADER_ARROW
    lv_calendar_header_arrow_create(calendar);
#endif
    lv_calendar_set_showed_date(calendar, 2021, 10);
}

#endif
```

```
def event_handler(evt):
    code = evt.get_code()

    if code == lv.EVENT.VALUE_CHANGED:
        source = evt.get_current_target()
        date = lv.calendar_date_t()
        if source.get_pressed_date(date) == lv.RES.OK:
            calendar.set_today_date(date.year, date.month, date.day)
            print("Clicked date: %02d.%02d.%02d"%(date.day, date.month, date.year))


calendar = lv.calendar(lv.scr_act())
calendar.set_size(200, 200)
calendar.align(lv.ALIGN.CENTER, 0, 20)
calendar.add_event_cb(event_handler, lv.EVENT.ALL, None)

calendar.set_today_date(2021, 02, 23)
calendar.set_showed_date(2021, 02)

# Highlight a few days
highlighted_days=[
    lv.calendar_date_t({'year':2021, 'month':2, 'day':6}),
    lv.calendar_date_t({'year':2021, 'month':2, 'day':11}),
    lv.calendar_date_t({'year':2021, 'month':2, 'day':22})
]

calendar.set_highlighted_dates(highlighted_days, len(highlighted_days))

lv.calendar_header_dropdown(calendar)
```

Canvas

Drawing on the Canvas and rotate

```
#include "../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES


#define CANVAS_WIDTH  200
#define CANVAS_HEIGHT  150

void lv_example_canvas_1(void)
```

(下页继续)

(续上页)

```
{
    lv_draw_rect_dsc_t rect_dsc;
    lv_draw_rect_dsc_init(&rect_dsc);
    rect_dsc.radius = 10;
    rect_dsc.bg_opa = LV_OPA_COVER;
    rect_dsc.bg_grad.dir = LV_GRAD_DIR_HOR;
    rect_dsc.bg_grad.stops[0].color = lv_palette_main(LV_PALETTE_RED);
    rect_dsc.bg_grad.stops[1].color = lv_palette_main(LV_PALETTE_BLUE);
    rect_dsc.border_width = 2;
    rect_dsc.border_opa = LV_OPA_90;
    rect_dsc.border_color = lv_color_white();
    rect_dsc.shadow_width = 5;
    rect_dsc.shadow_ofs_x = 5;
    rect_dsc.shadow_ofs_y = 5;

    lv_draw_label_dsc_t label_dsc;
    lv_draw_label_dsc_init(&label_dsc);
    label_dsc.color = lv_palette_main(LV_PALETTE_ORANGE);

    static lv_color_t cbuf[LV_CANVAS_BUF_SIZE_TRUE_COLOR(CANVAS_WIDTH, CANVAS_
↪HEIGHT)];

    lv_obj_t * canvas = lv_canvas_create(lv_scr_act());
    lv_canvas_set_buffer(canvas, cbuf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_IMG_CF_TRUE_
↪COLOR);
    lv_obj_center(canvas);
    lv_canvas_fill_bg(canvas, lv_palette_lighten(LV_PALETTE_GREY, 3), LV_OPA_COVER);

    lv_canvas_draw_rect(canvas, 70, 60, 100, 70, &rect_dsc);

    lv_canvas_draw_text(canvas, 40, 20, 100, &label_dsc, "Some text on text canvas");

    /*Test the rotation. It requires another buffer where the original image is␣
↪stored.
     *So copy the current image to buffer and rotate it to the canvas*/
    static lv_color_t cbuf_tmp[CANVAS_WIDTH * CANVAS_HEIGHT];
    memcpy(cbuf_tmp, cbuf, sizeof(cbuf_tmp));
    lv_img_dsc_t img;
    img.data = (void *)cbuf_tmp;
    img.header.cf = LV_IMG_CF_TRUE_COLOR;
    img.header.w = CANVAS_WIDTH;
    img.header.h = CANVAS_HEIGHT;
```

(下页继续)

(续上页)

```
    lv_canvas_fill_bg(canvas, lv_palette_lighten(LV_PALETTE_GREY, 3), LV_OPA_COVER);
    lv_canvas_transform(canvas, &img, 120, LV_IMG_ZOOM_NONE, 0, 0, CANVAS_WIDTH / 2,␣
↪CANVAS_HEIGHT / 2, true);
}

#endif
```

```
_CANVAS_WIDTH = 200
_CANVAS_HEIGHT = 150
LV_IMG_ZOOM_NONE = 256

rect_dsc = lv.draw_rect_dsc_t()
rect_dsc.init()
rect_dsc.radius = 10
rect_dsc.bg_opa = lv.OPA.COVER
rect_dsc.bg_grad.dir = lv.GRAD_DIR.HOR
rect_dsc.bg_grad.stops[0].color = lv.palette_main(lv.PALETTE.RED)
rect_dsc.bg_grad.stops[1].color = lv.palette_main(lv.PALETTE.BLUE)
rect_dsc.border_width = 2
rect_dsc.border_opa = lv.OPA._90
rect_dsc.border_color = lv.color_white()
rect_dsc.shadow_width = 5
rect_dsc.shadow_ofs_x = 5
rect_dsc.shadow_ofs_y = 5

label_dsc = lv.draw_label_dsc_t()
label_dsc.init()
label_dsc.color = lv.palette_main(lv.PALETTE.YELLOW)

cbuf = bytearray(_CANVAS_WIDTH * _CANVAS_HEIGHT * 4)

canvas = lv.canvas(lv.scr_act())
canvas.set_buffer(cbuf, _CANVAS_WIDTH, _CANVAS_HEIGHT, lv.img.CF.TRUE_COLOR)
canvas.center()
canvas.fill_bg(lv.palette_lighten(lv.PALETTE.GREY, 3), lv.OPA.COVER)

canvas.draw_rect(70, 60, 100, 70, rect_dsc)
canvas.draw_text(40, 20, 100, label_dsc, "Some text on text canvas")

# Test the rotation. It requires another buffer where the original image is stored.
# So copy the current image to buffer and rotate it to the canvas

img = lv.img_dsc_t()
```

(下页继续)

(续上页)

```
img.data = cbuf[:]
img.header.cf = lv.img.CF.TRUE_COLOR
img.header.w = _CANVAS_WIDTH
img.header.h = _CANVAS_HEIGHT

canvas.fill_bg(lv.palette_lighten(lv.PALETTE.GREY, 3), lv.OPA.COVER)
canvas.transform(img, 30, LV_IMG_ZOOM_NONE, 0, 0, _CANVAS_WIDTH // 2, _CANVAS_HEIGHT /
↪/ 2, True)
```

Transparent Canvas with chroma keying

```
#include "../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH  50
#define CANVAS_HEIGHT  50

/**
 * Create a transparent canvas with Chroma keying and indexed color format (palette).
 */
void lv_example_canvas_2(void)
{
    /*Create a button to better see the transparency*/
    lv_btn_create(lv_scr_act());

    /*Create a buffer for the canvas*/
    static lv_color_t cbuf[LV_CANVAS_BUF_SIZE_INDEXED_1BIT(CANVAS_WIDTH, CANVAS_
↪HEIGHT)];

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_scr_act());
    lv_canvas_set_buffer(canvas, cbuf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_IMG_CF_INDEXED_
↪1BIT);
    lv_canvas_set_palette(canvas, 0, LV_COLOR_CHROMA_KEY);
    lv_canvas_set_palette(canvas, 1, lv_palette_main(LV_PALETTE_RED));

    /*Create colors with the indices of the palette*/
    lv_color_t c0;
    lv_color_t c1;

    c0.full = 0;
    c1.full = 1;
```

(下页继续)

(续上页)

```
    /*Red background (There is no dedicated alpha channel in indexed images so LV_OPA_
↪COVER is ignored)*/
    lv_canvas_fill_bg(canvas, c1, LV_OPA_COVER);

    /*Create hole on the canvas*/
    uint32_t x;
    uint32_t y;
    for( y = 10; y < 30; y++) {
        for( x = 5; x < 20; x++) {
            lv_canvas_set_px_color(canvas, x, y, c0);
        }
    }

}
#endif
```

```
CANVAS_WIDTH   = 50
CANVAS_HEIGHT  = 50
LV_COLOR_CHROMA_KEY = lv.color_hex(0x00ff00)

def LV_IMG_BUF_SIZE_ALPHA_1BIT(w, h):
    return int(((w / 8) + 1) * h)

def LV_IMG_BUF_SIZE_INDEXED_1BIT(w, h):
    return LV_IMG_BUF_SIZE_ALPHA_1BIT(w, h) + 4 * 2

def LV_CANVAS_BUF_SIZE_INDEXED_1BIT(w, h):
    return LV_IMG_BUF_SIZE_INDEXED_1BIT(w, h)

#
# Create a transparent canvas with Chroma keying and indexed color format (palette).
#

# Create a button to better see the transparency
btn=lv.btn(lv.scr_act())

# Create a buffer for the canvas
cbuf= bytearray(LV_CANVAS_BUF_SIZE_INDEXED_1BIT(CANVAS_WIDTH, CANVAS_HEIGHT))

# Create a canvas and initialize its palette
canvas = lv.canvas(lv.scr_act())
canvas.set_buffer(cbuf, CANVAS_WIDTH, CANVAS_HEIGHT, lv.img.CF.INDEXED_1BIT)
```

(下页继续)

(续上页)

```
canvas.set_palette(0, LV_COLOR_CHROMA_KEY)
canvas.set_palette(1, lv.palette_main(lv.PALETTE.RED))

# Create colors with the indices of the palette
c0 = lv.color_t()
c1 = lv.color_t()

c0.full = 0
c1.full = 1

# Red background (There is no dedicated alpha channel in indexed images so LV_OPA_
↪COVER is ignored)
canvas.fill_bg(c1, lv.OPA.COVER)

# Create hole on the canvas
for y in range(10,30):
    for x in range(5,20):
        canvas.set_px(x, y, c0)
```

Chart

Line Chart

```
#include "../../lv_examples.h"
#if LV_USE_CHART && LV_BUILD_EXAMPLES

void lv_example_chart_1(void)
{
    /*Create a chart*/
    lv_obj_t * chart;
    chart = lv_chart_create(lv_scr_act());
    lv_obj_set_size(chart, 200, 150);
    lv_obj_center(chart);
    lv_chart_set_type(chart, LV_CHART_TYPE_LINE);   /*Show lines and points too*/

    /*Add two data series*/
    lv_chart_series_t * ser1 = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_
↪RED), LV_CHART_AXIS_PRIMARY_Y);
    lv_chart_series_t * ser2 = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_
↪GREEN), LV_CHART_AXIS_SECONDARY_Y);

    /*Set the next points on 'ser1'*/
```

(下页继续)

(续上页)

```
    lv_chart_set_next_value(chart, ser1, 10);
    lv_chart_set_next_value(chart, ser1, 10);
    lv_chart_set_next_value(chart, ser1, 10);
    lv_chart_set_next_value(chart, ser1, 10);
    lv_chart_set_next_value(chart, ser1, 10);
    lv_chart_set_next_value(chart, ser1, 10);
    lv_chart_set_next_value(chart, ser1, 10);
    lv_chart_set_next_value(chart, ser1, 30);
    lv_chart_set_next_value(chart, ser1, 70);
    lv_chart_set_next_value(chart, ser1, 90);

    /*Directly set points on 'ser2'*/
    ser2->y_points[0] = 90;
    ser2->y_points[1] = 70;
    ser2->y_points[2] = 65;
    ser2->y_points[3] = 65;
    ser2->y_points[4] = 65;
    ser2->y_points[5] = 65;
    ser2->y_points[6] = 65;
    ser2->y_points[7] = 65;
    ser2->y_points[8] = 65;
    ser2->y_points[9] = 65;

    lv_chart_refresh(chart); /*Required after direct set*/
}

#endif
```

```
# Create a chart
chart = lv.chart(lv.scr_act())
chart.set_size(200, 150)
chart.center()
chart.set_type(lv.chart.TYPE.LINE)   # Show lines and points too

# Add two data series
ser1 = chart.add_series(lv.palette_main(lv.PALETTE.RED), lv.chart.AXIS.PRIMARY_Y)
ser2 = chart.add_series(lv.palette_main(lv.PALETTE.GREEN), lv.chart.AXIS.SECONDARY_Y)
print(ser2)
# Set next points on ser1
chart.set_next_value(ser1,10)
chart.set_next_value(ser1,10)
chart.set_next_value(ser1,10)
chart.set_next_value(ser1,10)
```

(下页继续)

(续上页)

```
chart.set_next_value(ser1,10)
chart.set_next_value(ser1,10)
chart.set_next_value(ser1,10)
chart.set_next_value(ser1,30)
chart.set_next_value(ser1,70)
chart.set_next_value(ser1,90)

# Directly set points on 'ser2'
ser2.y_points = [90, 70, 65, 65, 65, 65, 65, 65, 65, 65]
chart.refresh()      #  Required after direct set
```

Faded area line chart with custom division lines

```
#include "../../lv_examples.h"
#if LV_USE_CHART && LV_DRAW_COMPLEX && LV_BUILD_EXAMPLES

static lv_obj_t * chart1;
static lv_chart_series_t * ser1;
static lv_chart_series_t * ser2;

static void draw_event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target(e);

    /*Add the faded area before the lines are drawn*/
    lv_obj_draw_part_dsc_t * dsc = lv_event_get_draw_part_dsc(e);
    if(dsc->part == LV_PART_ITEMS) {
        if(!dsc->p1 || !dsc->p2) return;

        /*Add a line mask that keeps the area below the line*/
        lv_draw_mask_line_param_t line_mask_param;
        lv_draw_mask_line_points_init(&line_mask_param, dsc->p1->x, dsc->p1->y, dsc->
↪p2->x, dsc->p2->y, LV_DRAW_MASK_LINE_SIDE_BOTTOM);
        int16_t line_mask_id = lv_draw_mask_add(&line_mask_param, NULL);

        /*Add a fade effect: transparent bottom covering top*/
        lv_coord_t h = lv_obj_get_height(obj);
        lv_draw_mask_fade_param_t fade_mask_param;
        lv_draw_mask_fade_init(&fade_mask_param, &obj->coords, LV_OPA_COVER, obj->
↪coords.y1 + h / 8, LV_OPA_TRANSP,obj->coords.y2);
        int16_t fade_mask_id = lv_draw_mask_add(&fade_mask_param, NULL);
```

(下页继续)

(续上页)

```
        /*Draw a rectangle that will be affected by the mask*/
        lv_draw_rect_dsc_t draw_rect_dsc;
        lv_draw_rect_dsc_init(&draw_rect_dsc);
        draw_rect_dsc.bg_opa = LV_OPA_20;
        draw_rect_dsc.bg_color = dsc->line_dsc->color;

        lv_area_t a;
        a.x1 = dsc->p1->x;
        a.x2 = dsc->p2->x - 1;
        a.y1 = LV_MIN(dsc->p1->y, dsc->p2->y);
        a.y2 = obj->coords.y2;
        lv_draw_rect(dsc->draw_ctx, &draw_rect_dsc, &a);

        /*Remove the masks*/
        lv_draw_mask_free_param(&line_mask_param);
        lv_draw_mask_free_param(&fade_mask_param);
        lv_draw_mask_remove_id(line_mask_id);
        lv_draw_mask_remove_id(fade_mask_id);
    }
    /*Hook the division lines too*/
    else if(dsc->part == LV_PART_MAIN) {
        if(dsc->line_dsc == NULL || dsc->p1 == NULL || dsc->p2 == NULL) return;

        /*Vertical line*/
        if(dsc->p1->x == dsc->p2->x) {
            dsc->line_dsc->color  = lv_palette_lighten(LV_PALETTE_GREY, 1);
            if(dsc->id == 3) {
                dsc->line_dsc->width  = 2;
                dsc->line_dsc->dash_gap  = 0;
                dsc->line_dsc->dash_width  = 0;
            }
            else {
                dsc->line_dsc->width = 1;
                dsc->line_dsc->dash_gap  = 6;
                dsc->line_dsc->dash_width  = 6;
            }
        }
        /*Horizontal line*/
        else {
            if(dsc->id == 2) {
                dsc->line_dsc->width  = 2;
                dsc->line_dsc->dash_gap  = 0;
```

(下页继续)

(续上页)

```
                dsc->line_dsc->dash_width  = 0;
            }
            else {
                dsc->line_dsc->width = 2;
                dsc->line_dsc->dash_gap  = 6;
                dsc->line_dsc->dash_width  = 6;
            }

            if(dsc->id == 1  || dsc->id == 3) {
                dsc->line_dsc->color  = lv_palette_main(LV_PALETTE_GREEN);
            } else {
                dsc->line_dsc->color  = lv_palette_lighten(LV_PALETTE_GREY, 1);
            }
        }
    }
}

static void add_data(lv_timer_t * timer)
{
    LV_UNUSED(timer);
    static uint32_t cnt = 0;
    lv_chart_set_next_value(chart1, ser1, lv_rand(20, 90));

    if(cnt % 4 == 0) lv_chart_set_next_value(chart1, ser2, lv_rand(40, 60));

    cnt++;
}

/**
 * Add a faded area effect to the line chart and make some division lines ticker
 */
void lv_example_chart_2(void)
{
    /*Create a chart1*/
    chart1 = lv_chart_create(lv_scr_act());
    lv_obj_set_size(chart1, 200, 150);
    lv_obj_center(chart1);
    lv_chart_set_type(chart1, LV_CHART_TYPE_LINE);   /*Show lines and points too*/

    lv_chart_set_div_line_count(chart1, 5, 7);

    lv_obj_add_event_cb(chart1, draw_event_cb, LV_EVENT_DRAW_PART_BEGIN, NULL);
    lv_chart_set_update_mode(chart1, LV_CHART_UPDATE_MODE_CIRCULAR);
```

(下页继续)

(续上页)

```
    /*Add two data series*/
    ser1 = lv_chart_add_series(chart1, lv_palette_main(LV_PALETTE_RED), LV_CHART_AXIS_
↪PRIMARY_Y);
    ser2 = lv_chart_add_series(chart1, lv_palette_main(LV_PALETTE_BLUE), LV_CHART_
↪AXIS_SECONDARY_Y);

    uint32_t i;
    for(i = 0; i < 10; i++) {
        lv_chart_set_next_value(chart1, ser1, lv_rand(20, 90));
        lv_chart_set_next_value(chart1, ser2, lv_rand(30, 70));
    }

    lv_timer_create(add_data, 200, NULL);
}

#endif
```

```
def draw_event_cb(e):

    obj = e.get_target()

    # Add the faded area before the lines are drawn
    dsc = lv.obj_draw_part_dsc_t.__cast__(e.get_param())
    if dsc.part != lv.PART.ITEMS:
        return
    if not dsc.p1 or not dsc.p2:
        return

    # Add a line mask that keeps the area below the line
    line_mask_param = lv.draw_mask_line_param_t()
    line_mask_param.points_init(dsc.p1.x, dsc.p1.y, dsc.p2.x, dsc.p2.y, lv.DRAW_MASK_
↪LINE_SIDE.BOTTOM)
    # line_mask_id = line_mask_param.draw_mask_add(None)
    line_mask_id = lv.draw_mask_add(line_mask_param, None)
    # Add a fade effect: transparent bottom covering top
    h = obj.get_height()
    fade_mask_param = lv.draw_mask_fade_param_t()
    coords = lv.area_t()
    obj.get_coords(coords)
    fade_mask_param.init(coords, lv.OPA.COVER, coords.y1 + h // 8, lv.OPA.TRANSP,
↪coords.y2)
    fade_mask_id = lv.draw_mask_add(fade_mask_param,None)
```

(下页继续)

(续上页)

```
    # Draw a rectangle that will be affected by the mask
    draw_rect_dsc = lv.draw_rect_dsc_t()
    draw_rect_dsc.init()
    draw_rect_dsc.bg_opa = lv.OPA._20
    draw_rect_dsc.bg_color = dsc.line_dsc.color

    a = lv.area_t()
    a.x1 = dsc.p1.x
    a.x2 = dsc.p2.x - 1
    a.y1 = min(dsc.p1.y, dsc.p2.y)
    coords = lv.area_t()
    obj.get_coords(coords)
    a.y2 = coords.y2
    dsc.draw_ctx.rect(draw_rect_dsc, a)

    # Remove the masks
    lv.draw_mask_remove_id(line_mask_id)
    lv.draw_mask_remove_id(fade_mask_id)


def add_data(timer):
    # LV_UNUSED(timer);
    cnt = 0
    chart1.set_next_value(ser1, lv.rand(20, 90))

    if cnt % 4 == 0:
        chart1.set_next_value(ser2, lv.rand(40, 60))

    cnt +=1

#
# Add a faded area effect to the line chart
#

# Create a chart1
chart1 = lv.chart(lv.scr_act())
chart1.set_size(200, 150)
chart1.center()
chart1.set_type(lv.chart.TYPE.LINE)    # Show lines and points too

chart1.add_event_cb(draw_event_cb, lv.EVENT.DRAW_PART_BEGIN, None)
chart1.set_update_mode(lv.chart.UPDATE_MODE.CIRCULAR)
```

(下页继续)

(续上页)

```
# Add two data series
ser1 = chart1.add_series(lv.palette_main(lv.PALETTE.RED), lv.chart.AXIS.PRIMARY_Y)
ser2 = chart1.add_series(lv.palette_main(lv.PALETTE.BLUE), lv.chart.AXIS.SECONDARY_Y)

for i in range(10):
    chart1.set_next_value(ser1, lv.rand(20, 90))
    chart1.set_next_value(ser2, lv.rand(30, 70))

timer = lv.timer_create(add_data, 200, None)
```

Axis ticks and labels with scrolling

```
#include "../../lv_examples.h"
#if LV_USE_CHART && LV_BUILD_EXAMPLES

static void draw_event_cb(lv_event_t * e)
{
    lv_obj_draw_part_dsc_t * dsc = lv_event_get_draw_part_dsc(e);
    if(!lv_obj_draw_part_check_type(dsc, &lv_chart_class, LV_CHART_DRAW_PART_TICK_
↪LABEL)) return;

    if(dsc->id == LV_CHART_AXIS_PRIMARY_X && dsc->text) {
        const char * month[] = {"Jan", "Febr", "March", "Apr", "May", "Jun", "July",
↪"Aug", "Sept", "Oct", "Nov", "Dec"};
        lv_snprintf(dsc->text, dsc->text_length, "%s", month[dsc->value]);
    }
}

/**
 * Add ticks and labels to the axis and demonstrate scrolling
 */
void lv_example_chart_3(void)
{
    /*Create a chart*/
    lv_obj_t * chart;
    chart = lv_chart_create(lv_scr_act());
    lv_obj_set_size(chart, 200, 150);
    lv_obj_center(chart);
    lv_chart_set_type(chart, LV_CHART_TYPE_BAR);
    lv_chart_set_range(chart, LV_CHART_AXIS_PRIMARY_Y, 0, 100);
    lv_chart_set_range(chart, LV_CHART_AXIS_SECONDARY_Y, 0, 400);
```

(下页继续)

(续上页)

```
    lv_chart_set_point_count(chart, 12);
    lv_obj_add_event_cb(chart, draw_event_cb, LV_EVENT_DRAW_PART_BEGIN, NULL);

    /*Add ticks and label to every axis*/
    lv_chart_set_axis_tick(chart, LV_CHART_AXIS_PRIMARY_X, 10, 5, 12, 3, true, 40);
    lv_chart_set_axis_tick(chart, LV_CHART_AXIS_PRIMARY_Y, 10, 5, 6, 2, true, 50);
    lv_chart_set_axis_tick(chart, LV_CHART_AXIS_SECONDARY_Y, 10, 5, 3, 4, true, 50);

    /*Zoom in a little in X*/
    lv_chart_set_zoom_x(chart, 800);

    /*Add two data series*/
    lv_chart_series_t * ser1 = lv_chart_add_series(chart, lv_palette_lighten(LV_
↪PALETTE_GREEN, 2), LV_CHART_AXIS_PRIMARY_Y);
    lv_chart_series_t * ser2 = lv_chart_add_series(chart, lv_palette_darken(LV_
↪PALETTE_GREEN, 2), LV_CHART_AXIS_SECONDARY_Y);

    /*Set the next points on 'ser1'*/
    lv_chart_set_next_value(chart, ser1, 31);
    lv_chart_set_next_value(chart, ser1, 66);
    lv_chart_set_next_value(chart, ser1, 10);
    lv_chart_set_next_value(chart, ser1, 89);
    lv_chart_set_next_value(chart, ser1, 63);
    lv_chart_set_next_value(chart, ser1, 56);
    lv_chart_set_next_value(chart, ser1, 32);
    lv_chart_set_next_value(chart, ser1, 35);
    lv_chart_set_next_value(chart, ser1, 57);
    lv_chart_set_next_value(chart, ser1, 85);
    lv_chart_set_next_value(chart, ser1, 22);
    lv_chart_set_next_value(chart, ser1, 58);

    lv_coord_t * ser2_array = lv_chart_get_y_array(chart, ser2);
    /*Directly set points on 'ser2'*/
    ser2_array[0] = 92;
    ser2_array[1] = 71;
    ser2_array[2] = 61;
    ser2_array[3] = 15;
    ser2_array[4] = 21;
    ser2_array[5] = 35;
    ser2_array[6] = 35;
    ser2_array[7] = 58;
    ser2_array[8] = 31;
    ser2_array[9] = 53;
```

(下页继续)

(续上页)

```
    ser2_array[10] = 33;
    ser2_array[11] = 73;

    lv_chart_refresh(chart); /*Required after direct set*/
}

#endif
```

```
def draw_event_cb(e):

    dsc = lv.obj_draw_part_dsc_t.__cast__(e.get_param())
    if dsc.part == lv.PART.TICKS and dsc.id == lv.chart.AXIS.PRIMARY_X:
        month = ["Jan", "Febr", "March", "Apr", "May", "Jun", "July", "Aug", "Sept",
↪"Oct", "Nov", "Dec"]
        # dsc.text is defined char text[16], I must therefore convert the Python␣
↪string to a byte_array
        dsc.text = bytes(month[dsc.value],"ascii")
#
# Add ticks and labels to the axis and demonstrate scrolling
#

# Create a chart
chart = lv.chart(lv.scr_act())
chart.set_size(200, 150)
chart.center()
chart.set_type(lv.chart.TYPE.BAR)
chart.set_range(lv.chart.AXIS.PRIMARY_Y, 0, 100)
chart.set_range(lv.chart.AXIS.SECONDARY_Y, 0, 400)
chart.set_point_count(12)
chart.add_event_cb(draw_event_cb, lv.EVENT.DRAW_PART_BEGIN, None)

# Add ticks and label to every axis
chart.set_axis_tick(lv.chart.AXIS.PRIMARY_X, 10, 5, 12, 3, True, 40)
chart.set_axis_tick(lv.chart.AXIS.PRIMARY_Y, 10, 5, 6, 2, True, 50)
chart.set_axis_tick(lv.chart.AXIS.SECONDARY_Y, 10, 5, 3, 4,True, 50)

# Zoom in a little in X
chart.set_zoom_x(800)

# Add two data series
ser1 = lv.chart.add_series(chart, lv.palette_lighten(lv.PALETTE.GREEN, 2), lv.chart.
↪AXIS.PRIMARY_Y)
ser2 = lv.chart.add_series(chart, lv.palette_darken(lv.PALETTE.GREEN, 2), lv.chart.
↪AXIS.SECONDARY_Y)
```

(下页继续)

(续上页)

```
# Set the next points on 'ser1'
chart.set_next_value(ser1, 31)
chart.set_next_value(ser1, 66)
chart.set_next_value(ser1, 10)
chart.set_next_value(ser1, 89)
chart.set_next_value(ser1, 63)
chart.set_next_value(ser1, 56)
chart.set_next_value(ser1, 32)
chart.set_next_value(ser1, 35)
chart.set_next_value(ser1, 57)
chart.set_next_value(ser1, 85)
chart.set_next_value(ser1, 22)
chart.set_next_value(ser1, 58)

# Directly set points on 'ser2'
ser2.y_points = [92,71,61,15,21,35,35,58,31,53,33,73]

chart.refresh()  # Required after direct set
```

Show the value of the pressed points

```
#include "../../lv_examples.h"
#if LV_USE_CHART && LV_BUILD_EXAMPLES


static void event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * chart = lv_event_get_target(e);

    if(code == LV_EVENT_VALUE_CHANGED) {
        lv_obj_invalidate(chart);
    }
    if(code == LV_EVENT_REFR_EXT_DRAW_SIZE) {
        lv_coord_t * s = lv_event_get_param(e);
        *s = LV_MAX(*s, 20);
    }
    else if(code == LV_EVENT_DRAW_POST_END) {
        int32_t id = lv_chart_get_pressed_point(chart);
        if(id == LV_CHART_POINT_NONE) return;
```

(下页继续)

(续上页)

```
        LV_LOG_USER("Selected point %d", (int)id);

        lv_chart_series_t * ser = lv_chart_get_series_next(chart, NULL);
        while(ser) {
            lv_point_t p;
            lv_chart_get_point_pos_by_id(chart, ser, id, &p);

            lv_coord_t * y_array = lv_chart_get_y_array(chart, ser);
            lv_coord_t value = y_array[id];

            char buf[16];
            lv_snprintf(buf, sizeof(buf), LV_SYMBOL_DUMMY"$%d", value);

            lv_draw_rect_dsc_t draw_rect_dsc;
            lv_draw_rect_dsc_init(&draw_rect_dsc);
            draw_rect_dsc.bg_color = lv_color_black();
            draw_rect_dsc.bg_opa = LV_OPA_50;
            draw_rect_dsc.radius = 3;
            draw_rect_dsc.bg_img_src = buf;
            draw_rect_dsc.bg_img_recolor = lv_color_white();

            lv_area_t a;
            a.x1 = chart->coords.x1 + p.x - 20;
            a.x2 = chart->coords.x1 + p.x + 20;
            a.y1 = chart->coords.y1 + p.y - 30;
            a.y2 = chart->coords.y1 + p.y - 10;

            lv_draw_ctx_t * draw_ctx = lv_event_get_draw_ctx(e);
            lv_draw_rect(draw_ctx, &draw_rect_dsc, &a);

            ser = lv_chart_get_series_next(chart, ser);
        }
    }
    else if(code == LV_EVENT_RELEASED) {
        lv_obj_invalidate(chart);
    }
}

/**
 * Show the value of the pressed points
 */
void lv_example_chart_4(void)
```

(下页继续)

(续上页)

```
{
    /*Create a chart*/
    lv_obj_t * chart;
    chart = lv_chart_create(lv_scr_act());
    lv_obj_set_size(chart, 200, 150);
    lv_obj_center(chart);

    lv_obj_add_event_cb(chart, event_cb, LV_EVENT_ALL, NULL);
    lv_obj_refresh_ext_draw_size(chart);

    /*Zoom in a little in X*/
    lv_chart_set_zoom_x(chart, 800);

    /*Add two data series*/
    lv_chart_series_t * ser1 = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_
↪RED), LV_CHART_AXIS_PRIMARY_Y);
    lv_chart_series_t * ser2 = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_
↪GREEN), LV_CHART_AXIS_PRIMARY_Y);
    uint32_t i;
    for(i = 0; i < 10; i++) {
        lv_chart_set_next_value(chart, ser1, lv_rand(60,90));
        lv_chart_set_next_value(chart, ser2, lv_rand(10,40));
    }
}

#endif
```

```
def event_cb(e):
    code = e.get_code()
    chart = e.get_target()

    if code == lv.EVENT.VALUE_CHANGED:
        chart.invalidate()

    if code == lv.EVENT.REFR_EXT_DRAW_SIZE:
        e.set_ext_draw_size(20)

    elif code == lv.EVENT.DRAW_POST_END:
        id = lv.chart.get_pressed_point(chart)
        if id == lv.CHART_POINT.NONE:
            return
        # print("Selected point ", id)
        for i in range(len(series)):
```

(下页继续)

(续上页)

```
            p = lv.point_t()
            chart.get_point_pos_by_id(series[i], id, p)
            value = series_points[i][id]
            buf = lv.SYMBOL.DUMMY + "$" + str(value)

            draw_rect_dsc = lv.draw_rect_dsc_t()
            draw_rect_dsc.init()
            draw_rect_dsc.bg_color = lv.color_black()
            draw_rect_dsc.bg_opa = lv.OPA._50
            draw_rect_dsc.radius = 3
            draw_rect_dsc.bg_img_src = buf
            draw_rect_dsc.bg_img_recolor = lv.color_white()

            a = lv.area_t()
            coords = lv.area_t()
            chart.get_coords(coords)
            a.x1 = coords.x1 + p.x - 20
            a.x2 = coords.x1 + p.x + 20
            a.y1 = coords.y1 + p.y - 30
            a.y2 = coords.y1 + p.y - 10

            clip_area = lv.area_t.__cast__(e.get_param())
            lv.draw_rect(a, clip_area, draw_rect_dsc)

    elif code == lv.EVENT.RELEASED:
        chart.invalidate()

#
# Add ticks and labels to the axis and demonstrate scrolling
#

# Create a chart
chart = lv.chart(lv.scr_act())
chart.set_size(200, 150)
chart.center()

chart.add_event_cb(event_cb, lv.EVENT.ALL, None)
chart.refresh_ext_draw_size()

# Zoom in a little in X
chart.set_zoom_x(800)

# Add two data series
```

(下页继续)

(续上页)

```
ser1 = chart.add_series(lv.palette_main(lv.PALETTE.RED), lv.chart.AXIS.PRIMARY_Y)
ser2 = chart.add_series(lv.palette_main(lv.PALETTE.GREEN), lv.chart.AXIS.PRIMARY_Y)

ser1_p = []
ser2_p = []
for i in range(10):
    ser1_p.append(lv.rand(60,90))
    ser2_p.append(lv.rand(10,40))
ser1.y_points = ser1_p
ser2.y_points = ser2_p

series = [ser1,ser2]
series_points=[ser1_p,ser2_p]
```

Display 1000 data points with zooming and scrolling

```
#include "../../lv_examples.h"
#if LV_USE_CHART && LV_USE_SLIDER && LV_BUILD_EXAMPLES

static lv_obj_t * chart;
/* Source: https://github.com/ankur219/ECG-Arrhythmia-classification/blob/
↪642230149583adfae1e4bd26c6f0e1fd8af2be0e/sample.csv*/
static const lv_coord_t ecg_sample[] = {
    -2, 2, 0, -15, -39, -63, -71, -68, -67, -69, -84, -95, -104, -107, -108, -107, -
↪107, -107, -107, -114, -118, -117,
    -112, -100, -89, -83, -71, -64, -58, -58, -62, -62, -58, -51, -46, -39, -27, -10,␣
↪4, 7, 1, -3, 0, 14, 24, 30, 25, 19,
    13, 7, 12, 15, 18, 21, 13, 6, 9, 8, 17, 19, 13, 11, 11, 11, 23, 30, 37, 34, 25,␣
↪14, 15, 19, 28, 31, 26, 23, 25, 31,
    39, 37, 37, 34, 30, 32, 22, 29, 31, 33, 37, 23, 13, 7, 2, 4, -2, 2, 11, 22, 33,␣
↪19, -1, -27, -55, -67, -72, -71, -63,
    -49, -18, 35, 113, 230, 369, 525, 651, 722, 730, 667, 563, 454, 357, 305, 288,␣
↪274, 255, 212, 173, 143, 117, 82, 39,
    -13, -53, -78, -91, -101, -113, -124, -131, -131, -131, -129, -128, -129, -125, -
↪123, -123, -129, -139, -148, -153,
    -159, -166, -183, -205, -227, -243, -248, -246, -254, -280, -327, -381, -429, -
↪473, -517, -556, -592, -612, -620,
    -620, -614, -604, -591, -574, -540, -497, -441, -389, -358, -336, -313, -284, -
↪222, -167, -114, -70, -47, -28, -4, 12,
    38, 52, 58, 56, 56, 57, 68, 77, 86, 86, 80, 69, 67, 70, 82, 85, 89, 90, 89, 89,␣
↪88, 91, 96, 97, 91, 83, 78, 82, 88, 95,
```

(下页继续)

(续上页)

```
    96, 105, 106, 110, 102, 100, 96, 98, 97, 101, 98, 99, 100, 107, 113, 119, 115,␣
↪110, 96, 85, 73, 64, 69, 76, 79,
    78, 75, 85, 100, 114, 113, 105, 96, 84, 74, 66, 60, 75, 85, 89, 83, 67, 61, 67,␣
↪73, 79, 74, 63, 57, 56, 58, 61, 55,
    48, 45, 46, 55, 62, 55, 49, 43, 50, 59, 63, 57, 40, 31, 23, 25, 27, 31, 35, 34,␣
↪30, 36, 34, 42, 38, 36, 40, 46, 50,
    47, 32, 30, 32, 52, 67, 73, 71, 63, 54, 53, 45, 41, 28, 13, 3, 1, 4, 4, -8, -23, -
↪32, -31, -19, -5, 3, 9, 13, 19,
    24, 27, 29, 25, 22, 26, 32, 42, 51, 56, 60, 57, 55, 53, 53, 54, 59, 54, 49, 26, -
↪3, -11, -20, -47, -100, -194, -236,
    -212, -123, 8, 103, 142, 147, 120, 105, 98, 93, 81, 61, 40, 26, 28, 30, 30, 27,␣
↪19, 17, 21, 20, 19, 19, 22, 36, 40,
    35, 20, 7, 1, 10, 18, 27, 22, 6, -4, -2, 3, 6, -2, -13, -14, -10, -2, 3, 2, -1, -
↪5, -10, -19, -32, -42, -55, -60,
    -68, -77, -86, -101, -110, -117, -115, -104, -92, -84, -85, -84, -73, -65, -52, -
↪50, -45, -35, -20, -3, 12, 20, 25,
    26, 28, 28, 30, 28, 25, 28, 33, 42, 42, 36, 23, 9, 0, 1, -4, 1, -4, -4, 1, 5, 9,␣
↪9, -3, -1, -18, -50, -108, -190,
    -272, -340, -408, -446, -537, -643, -777, -894, -920, -853, -697, -461, -251, -60,
↪ 58, 103, 129, 139, 155, 170, 173,
    178, 185, 190, 193, 200, 208, 215, 225, 224, 232, 234, 240, 240, 236, 229, 226,␣
↪224, 232, 233, 232, 224, 219, 219,
    223, 231, 226, 223, 219, 218, 223, 223, 223, 233, 245, 268, 286, 296, 295, 283,␣
↪271, 263, 252, 243, 226, 210, 197,
    186, 171, 152, 133, 117, 114, 110, 107, 96, 80, 63, 48, 40, 38, 34, 28, 15, 2, -7,
↪ -11, -14, -18, -29, -37, -44, -50,
    -58, -63, -61, -52, -50, -48, -61, -59, -58, -54, -47, -52, -62, -61, -64, -54, -
↪52, -59, -69, -76, -76, -69, -67,
    -74, -78, -81, -80, -73, -65, -57, -53, -51, -47, -35, -27, -22, -22, -24, -21, -
↪17, -13, -10, -11, -13, -20, -20,
    -12, -2, 7, -1, -12, -16, -13, -2, 2, -4, -5, -2, 9, 19, 19, 14, 11, 13, 19, 21,␣
↪20, 18, 19, 19, 19, 16, 15, 13, 14,
    9, 3, -5, -9, -5, -3, -2, -3, -3, 2, 8, 9, 9, 5, 6, 8, 8, 7, 4, 3, 4, 5, 3, 5, 5,␣
↪13, 13, 12, 10, 10, 15, 22, 17,
    14, 7, 10, 15, 16, 11, 12, 10, 13, 9, -2, -4, -2, 7, 16, 16, 17, 16, 7, -1, -16, -
↪18, -16, -9, -4, -5, -10, -9, -8,
    -3, -4, -10, -19, -20, -16, -9, -9, -23, -40, -48, -43, -33, -19, -21, -26, -31, -
↪33, -19, 0, 17, 24, 9, -17, -47,
    -63, -67, -59, -52, -51, -50, -49, -42, -26, -21, -15, -20, -23, -22, -19, -12, -
↪8, 5, 18, 27, 32, 26, 25, 26, 22,
    23, 17, 14, 17, 21, 25, 2, -45, -121, -196, -226, -200, -118, -9, 73, 126, 131,␣
↪114, 87, 60, 42, 29, 26, 34, 35, 34,
    25, 12, 9, 7, 3, 2, -8, -11, 2, 23, 38, 41, 23, 9, 10, 13, 16, 8, -8, -17, -23, -
↪26, -25, -21, -15, -10, -13, -13,
```

(下页继续)

(续上页)

```
    -19, -22, -29, -40, -48, -48, -54, -55, -66, -82, -85, -90, -92, -98, -114, -119,␣
↪-124, -129, -132, -146, -146, -138,
    -124, -99, -85, -72, -65, -65, -65, -66, -63, -64, -64, -58, -46, -26, -9, 2, 2,␣
↪4, 0, 1, 4, 3, 10, 11, 10, 2, -4,
    0, 10, 18, 20, 6, 2, -9, -7, -3, -3, -2, -7, -12, -5, 5, 24, 36, 31, 25, 6, 3, 7,␣
↪12, 17, 11, 0, -6, -9, -8, -7, -5,
    -6, -2, -2, -6, -2, 2, 14, 24, 22, 15, 8, 4, 6, 7, 12, 16, 25, 20, 7, -16, -41, -
↪60, -67, -65, -54, -35, -11, 30,
    84, 175, 302, 455, 603, 707, 743, 714, 625, 519, 414, 337, 300, 281, 263, 239,␣
↪197, 163, 136, 109, 77, 34, -18, -50,
    -66, -74, -79, -92, -107, -117, -127, -129, -135, -139, -141, -155, -159, -167, -
↪171, -169, -174, -175, -178, -191,
    -202, -223, -235, -243, -237, -240, -256, -298, -345, -393, -432, -475, -518, -
↪565, -596, -619, -623, -623, -614,
    -599, -583, -559, -524, -477, -425, -383, -357, -331, -301, -252, -198, -143, -96,
↪ -57, -29, -8, 10, 31, 45, 60, 65,
    70, 74, 76, 79, 82, 79, 75, 62,
};

static void slider_x_event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target(e);
    int32_t v = lv_slider_get_value(obj);
    lv_chart_set_zoom_x(chart, v);
}

static void slider_y_event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target(e);
    int32_t v = lv_slider_get_value(obj);
    lv_chart_set_zoom_y(chart, v);
}

/**
 * Display 1000 data points with zooming and scrolling.
 * See how the chart changes drawing mode (draw only vertical lines) when
 * the points get too crowded.
 */
void lv_example_chart_5(void)
{
    /*Create a chart*/
    chart = lv_chart_create(lv_scr_act());
    lv_obj_set_size(chart, 200, 150);
```

(下页继续)

(续上页)

```
    lv_obj_align(chart, LV_ALIGN_CENTER, -30, -30);
    lv_chart_set_range(chart, LV_CHART_AXIS_PRIMARY_Y, -1000, 1000);

    /*Do not display points on the data*/
    lv_obj_set_style_size(chart, 0, LV_PART_INDICATOR);

    lv_chart_series_t * ser = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_
↪RED), LV_CHART_AXIS_PRIMARY_Y);

    uint32_t pcnt = sizeof(ecg_sample) / sizeof(ecg_sample[0]);
    lv_chart_set_point_count(chart, pcnt);
    lv_chart_set_ext_y_array(chart, ser, (lv_coord_t *)ecg_sample);

    lv_obj_t * slider;
    slider = lv_slider_create(lv_scr_act());
    lv_slider_set_range(slider, LV_IMG_ZOOM_NONE, LV_IMG_ZOOM_NONE * 10);
    lv_obj_add_event_cb(slider, slider_x_event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    lv_obj_set_size(slider, 200, 10);
    lv_obj_align_to(slider, chart, LV_ALIGN_OUT_BOTTOM_MID, 0, 20);

    slider = lv_slider_create(lv_scr_act());
    lv_slider_set_range(slider, LV_IMG_ZOOM_NONE, LV_IMG_ZOOM_NONE * 10);
    lv_obj_add_event_cb(slider, slider_y_event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    lv_obj_set_size(slider, 10, 150);
    lv_obj_align_to(slider, chart, LV_ALIGN_OUT_RIGHT_MID, 20, 0);
}

#endif
```

```
# Source: https://github.com/ankur219/ECG-Arrhythmia-classification/blob/
↪642230149583adfae1e4bd26c6f0e1fd8af2be0e/sample.csv
ecg_sample = [
    -2, 2, 0, -15, -39, -63, -71, -68, -67, -69, -84, -95, -104, -107, -108, -107, -
↪107, -107, -107, -114, -118, -117,
    -112, -100, -89, -83, -71, -64, -58, -58, -62, -62, -58, -51, -46, -39, -27, -10,␣
↪4, 7, 1, -3, 0, 14, 24, 30, 25, 19,
    13, 7, 12, 15, 18, 21, 13, 6, 9, 8, 17, 19, 13, 11, 11, 11, 23, 30, 37, 34, 25,␣
↪14, 15, 19, 28, 31, 26, 23, 25, 31,
    39, 37, 37, 34, 30, 32, 22, 29, 31, 33, 37, 23, 13, 7, 2, 4, -2, 2, 11, 22, 33,␣
↪19, -1, -27, -55, -67, -72, -71, -63,
    -49, -18, 35, 113, 230, 369, 525, 651, 722, 730, 667, 563, 454, 357, 305, 288,␣
↪274, 255, 212, 173, 143, 117, 82, 39,
    -13, -53, -78, -91, -101, -113, -124, -131, -131, -131, -129, -128, -129, -125, -
↪123, -123, -129, -139, -148, -153,
```

(下页继续)

(续上页)

```
    -159, -166, -183, -205, -227, -243, -248, -246, -254, -280, -327, -381, -429, -
↪473, -517, -556, -592, -612, -620,
    -620, -614, -604, -591, -574, -540, -497, -441, -389, -358, -336, -313, -284, -
↪222, -167, -114, -70, -47, -28, -4, 12,
    38, 52, 58, 56, 56, 57, 68, 77, 86, 86, 80, 69, 67, 70, 82, 85, 89, 90, 89, 89,␣
↪88, 91, 96, 97, 91, 83, 78, 82, 88, 95,
    96, 105, 106, 110, 102, 100, 96, 98, 97, 101, 98, 99, 100, 107, 113, 119, 115,␣
↪110, 96, 85, 73, 64, 69, 76, 79,
    78, 75, 85, 100, 114, 113, 105, 96, 84, 74, 66, 60, 75, 85, 89, 83, 67, 61, 67,␣
↪73, 79, 74, 63, 57, 56, 58, 61, 55,
    48, 45, 46, 55, 62, 55, 49, 43, 50, 59, 63, 57, 40, 31, 23, 25, 27, 31, 35, 34,␣
↪30, 36, 34, 42, 38, 36, 40, 46, 50,
    47, 32, 30, 32, 52, 67, 73, 71, 63, 54, 53, 45, 41, 28, 13, 3, 1, 4, 4, -8, -23, -
↪32, -31, -19, -5, 3, 9, 13, 19,
    24, 27, 29, 25, 22, 26, 32, 42, 51, 56, 60, 57, 55, 53, 53, 54, 59, 54, 49, 26, -
↪3, -11, -20, -47, -100, -194, -236,
    -212, -123, 8, 103, 142, 147, 120, 105, 98, 93, 81, 61, 40, 26, 28, 30, 30, 27,␣
↪19, 17, 21, 20, 19, 19, 22, 36, 40,
    35, 20, 7, 1, 10, 18, 27, 22, 6, -4, -2, 3, 6, -2, -13, -14, -10, -2, 3, 2, -1, -
↪5, -10, -19, -32, -42, -55, -60,
    -68, -77, -86, -101, -110, -117, -115, -104, -92, -84, -85, -84, -73, -65, -52, -
↪50, -45, -35, -20, -3, 12, 20, 25,
    26, 28, 28, 30, 28, 25, 28, 33, 42, 42, 36, 23, 9, 0, 1, -4, 1, -4, -4, 1, 5, 9,␣
↪9, -3, -1, -18, -50, -108, -190,
    -272, -340, -408, -446, -537, -643, -777, -894, -920, -853, -697, -461, -251, -60,
↪ 58, 103, 129, 139, 155, 170, 173,
    178, 185, 190, 193, 200, 208, 215, 225, 224, 232, 234, 240, 240, 236, 229, 226,␣
↪224, 232, 233, 232, 224, 219, 219,
    223, 231, 226, 223, 219, 218, 223, 223, 223, 233, 245, 268, 286, 296, 295, 283,␣
↪271, 263, 252, 243, 226, 210, 197,
    186, 171, 152, 133, 117, 114, 110, 107, 96, 80, 63, 48, 40, 38, 34, 28, 15, 2, -7,
↪ -11, -14, -18, -29, -37, -44, -50,
    -58, -63, -61, -52, -50, -48, -61, -59, -58, -54, -47, -52, -62, -61, -64, -54, -
↪52, -59, -69, -76, -76, -69, -67,
    -74, -78, -81, -80, -73, -65, -57, -53, -51, -47, -35, -27, -22, -22, -24, -21, -
↪17, -13, -10, -11, -13, -20, -20,
    -12, -2, 7, -1, -12, -16, -13, -2, 2, -4, -5, -2, 9, 19, 19, 14, 11, 13, 19, 21,␣
↪20, 18, 19, 19, 19, 16, 15, 13, 14,
    9, 3, -5, -9, -5, -3, -2, -3, -3, 2, 8, 9, 9, 5, 6, 8, 8, 7, 4, 3, 4, 5, 3, 5, 5,␣
↪13, 13, 12, 10, 10, 15, 22, 17,
    14, 7, 10, 15, 16, 11, 12, 10, 13, 9, -2, -4, -2, 7, 16, 16, 17, 16, 7, -1, -16, -
↪18, -16, -9, -4, -5, -10, -9, -8,
    -3, -4, -10, -19, -20, -16, -9, -9, -23, -40, -48, -43, -33, -19, -21, -26, -31, -
↪33, -19, 0, 17, 24, 9, -17, -47,
```

(下页继续)

(续上页)

```
    -63, -67, -59, -52, -51, -50, -49, -42, -26, -21, -15, -20, -23, -22, -19, -12, -
↪8, 5, 18, 27, 32, 26, 25, 26, 22,
    23, 17, 14, 17, 21, 25, 2, -45, -121, -196, -226, -200, -118, -9, 73, 126, 131,␣
↪114, 87, 60, 42, 29, 26, 34, 35, 34,
    25, 12, 9, 7, 3, 2, -8, -11, 2, 23, 38, 41, 23, 9, 10, 13, 16, 8, -8, -17, -23, -
↪26, -25, -21, -15, -10, -13, -13,
    -19, -22, -29, -40, -48, -48, -54, -55, -66, -82, -85, -90, -92, -98, -114, -119,␣
↪-124, -129, -132, -146, -146, -138,
    -124, -99, -85, -72, -65, -65, -65, -66, -63, -64, -64, -58, -46, -26, -9, 2, 2,␣
↪4, 0, 1, 4, 3, 10, 11, 10, 2, -4,
    0, 10, 18, 20, 6, 2, -9, -7, -3, -3, -2, -7, -12, -5, 5, 24, 36, 31, 25, 6, 3, 7,␣
↪12, 17, 11, 0, -6, -9, -8, -7, -5,
    -6, -2, -2, -6, -2, 2, 14, 24, 22, 15, 8, 4, 6, 7, 12, 16, 25, 20, 7, -16, -41, -
↪60, -67, -65, -54, -35, -11, 30,
    84, 175, 302, 455, 603, 707, 743, 714, 625, 519, 414, 337, 300, 281, 263, 239,␣
↪197, 163, 136, 109, 77, 34, -18, -50,
    -66, -74, -79, -92, -107, -117, -127, -129, -135, -139, -141, -155, -159, -167, -
↪171, -169, -174, -175, -178, -191,
    -202, -223, -235, -243, -237, -240, -256, -298, -345, -393, -432, -475, -518, -
↪565, -596, -619, -623, -623, -614,
    -599, -583, -559, -524, -477, -425, -383, -357, -331, -301, -252, -198, -143, -96,
↪ -57, -29, -8, 10, 31, 45, 60, 65,
    70, 74, 76, 79, 82, 79, 75, 62,
]

def slider_x_event_cb(e):

    slider = e.get_target()
    v = slider.get_value()
    chart.set_zoom_x(v)

def slider_y_event_cb(e):

    slider = e.get_target()
    v = slider.get_value()
    chart.set_zoom_y(v)


#
# Display 1000 data points with zooming and scrolling.
# See how the chart changes drawing mode (draw only vertical lines) when
# the points get too crowded.
```

(下页继续)

(续上页)

```
# Create a chart
chart = lv.chart(lv.scr_act())
chart.set_size(200, 150)
chart.align(lv.ALIGN.CENTER, -30, -30)
chart.set_range(lv.chart.AXIS.PRIMARY_Y, -1000, 1000)

# Do not display points on the data
chart.set_style_size(0, lv.PART.INDICATOR)

ser = chart.add_series(lv.palette_main(lv.PALETTE.RED), lv.chart.AXIS.PRIMARY_Y)

pcnt = len(ecg_sample)
chart.set_point_count(pcnt)
chart.set_ext_y_array(ser, ecg_sample)

slider = lv.slider(lv.scr_act())
slider.set_range(lv.IMG_ZOOM.NONE, lv.IMG_ZOOM.NONE * 10)
slider.add_event_cb(slider_x_event_cb, lv.EVENT.VALUE_CHANGED, None)
slider.set_size(200,10)
slider.align_to(chart, lv.ALIGN.OUT_BOTTOM_MID, 0, 20)

slider = lv.slider(lv.scr_act())
slider.set_range(lv.IMG_ZOOM.NONE, lv.IMG_ZOOM.NONE * 10)
slider.add_event_cb(slider_y_event_cb, lv.EVENT.VALUE_CHANGED, None)
slider.set_size(10, 150)
slider.align_to(chart, lv.ALIGN.OUT_RIGHT_MID, 20, 0)
```

Show cursor on the clicked point

```
#include "../../lv_examples.h"
#if LV_USE_CHART && LV_BUILD_EXAMPLES

static lv_obj_t * chart;
static lv_chart_series_t * ser;
static lv_chart_cursor_t * cursor;

static void event_cb(lv_event_t * e)
{
    static int32_t last_id = -1;
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target(e);
```

(下页继续)

(续上页)

```
    if(code == LV_EVENT_VALUE_CHANGED) {
        last_id = lv_chart_get_pressed_point(obj);
        if(last_id != LV_CHART_POINT_NONE) {
            lv_chart_set_cursor_point(obj, cursor, NULL, last_id);
        }
    }
    else if(code == LV_EVENT_DRAW_PART_END) {
        lv_obj_draw_part_dsc_t * dsc = lv_event_get_draw_part_dsc(e);
        if(!lv_obj_draw_part_check_type(dsc, &lv_chart_class, LV_CHART_DRAW_PART_
↪CURSOR)) return;
        if(dsc->p1 == NULL || dsc->p2 == NULL || dsc->p1->y != dsc->p2->y || last_id
↪< 0) return;

        lv_coord_t * data_array = lv_chart_get_y_array(chart, ser);
        lv_coord_t v = data_array[last_id];
        char buf[16];
        lv_snprintf(buf, sizeof(buf), "%d", v);

        lv_point_t size;
        lv_txt_get_size(&size, buf, LV_FONT_DEFAULT, 0, 0, LV_COORD_MAX, LV_TEXT_FLAG_
↪NONE);

        lv_area_t a;
        a.y2 = dsc->p1->y - 5;
        a.y1 = a.y2 - size.y - 10;
        a.x1 = dsc->p1->x + 10;
        a.x2 = a.x1 + size.x + 10;

        lv_draw_rect_dsc_t draw_rect_dsc;
        lv_draw_rect_dsc_init(&draw_rect_dsc);
        draw_rect_dsc.bg_color = lv_palette_main(LV_PALETTE_BLUE);
        draw_rect_dsc.radius = 3;

        lv_draw_rect(dsc->draw_ctx, &draw_rect_dsc, &a);

        lv_draw_label_dsc_t draw_label_dsc;
        lv_draw_label_dsc_init(&draw_label_dsc);
        draw_label_dsc.color = lv_color_white();
        a.x1 += 5;
        a.x2 -= 5;
        a.y1 += 5;
        a.y2 -= 5;
```

(下页继续)

(续上页)

```
        lv_draw_label(dsc->draw_ctx, &draw_label_dsc, &a, buf, NULL);
    }
}

/**
 * Show cursor on the clicked point
 */
void lv_example_chart_6(void)
{
    chart = lv_chart_create(lv_scr_act());
    lv_obj_set_size(chart, 200, 150);
    lv_obj_align(chart, LV_ALIGN_CENTER, 0, -10);

    lv_chart_set_axis_tick(chart, LV_CHART_AXIS_PRIMARY_Y, 10, 5, 6, 5, true, 40);
    lv_chart_set_axis_tick(chart, LV_CHART_AXIS_PRIMARY_X, 10, 5, 10, 1, true, 30);

    lv_obj_add_event_cb(chart, event_cb, LV_EVENT_ALL, NULL);
    lv_obj_refresh_ext_draw_size(chart);

    cursor = lv_chart_add_cursor(chart, lv_palette_main(LV_PALETTE_BLUE), LV_DIR_LEFT␣
↪| LV_DIR_BOTTOM);

    ser = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_RED), LV_CHART_AXIS_
↪PRIMARY_Y);
    uint32_t i;
    for(i = 0; i < 10; i++) {
        lv_chart_set_next_value(chart, ser, lv_rand(10,90));
    }

    lv_chart_set_zoom_x(chart, 500);

    lv_obj_t * label = lv_label_create(lv_scr_act());
    lv_label_set_text(label, "Click on a point");
    lv_obj_align_to(label, chart, LV_ALIGN_OUT_TOP_MID, 0, -5);
}

#endif
```

```
class ExampleChart_6():

    def __init__(self):
        self.last_id = -1
        #
```

(下页继续)

(续上页)

```
        # Show cursor on the clicked point
        #

        chart = lv.chart(lv.scr_act())
        chart.set_size(200, 150)
        chart.align(lv.ALIGN.CENTER, 0, -10)

        chart.set_axis_tick(lv.chart.AXIS.PRIMARY_Y, 10, 5, 6, 5, True, 40)
        chart.set_axis_tick(lv.chart.AXIS.PRIMARY_X, 10, 5, 10, 1, True, 30)

        chart.add_event_cb(self.event_cb, lv.EVENT.ALL, None)
        chart.refresh_ext_draw_size()

        self.cursor = chart.add_cursor(lv.palette_main(lv.PALETTE.BLUE), lv.DIR.LEFT␣
↪| lv.DIR.BOTTOM)

        self.ser = chart.add_series(lv.palette_main(lv.PALETTE.RED), lv.chart.AXIS.
↪PRIMARY_Y)

        self.ser_p = []
        for i in range(10):
            self.ser_p.append(lv.rand(10,90))
        self.ser.y_points = self.ser_p

        newser = chart.get_series_next(None)
        # print("length of data points: ",len(newser.points))
        chart.set_zoom_x(500)

        label = lv.label(lv.scr_act())
        label.set_text("Click on a point")
        label.align_to(chart, lv.ALIGN.OUT_TOP_MID, 0, -5)


    def event_cb(self,e):

        code = e.get_code()
        chart = e.get_target()

        if code == lv.EVENT.VALUE_CHANGED:
            # print("last_id: ",self.last_id)
            self.last_id = chart.get_pressed_point()
            if self.last_id != lv.CHART_POINT.NONE:
                p = lv.point_t()
```

(下页继续)

(续上页)

```
                chart.get_point_pos_by_id(self.ser, self.last_id, p)
                chart.set_cursor_point(self.cursor, None, self.last_id)

        elif code == lv.EVENT.DRAW_PART_END:
            # print("EVENT.DRAW_PART_END")
            dsc = lv.obj_draw_part_dsc_t.__cast__(e.get_param())
            # if dsc.p1 and dsc.p2:
                # print("p1, p2", dsc.p1,dsc.p2)
                # print("p1.y, p2.y", dsc.p1.y, dsc.p2.y)
                # print("last_id: ",self.last_id)
            if dsc.part == lv.PART.CURSOR and dsc.p1 and dsc.p2 and dsc.p1.y == dsc.
↪p2.y and self.last_id >= 0:

                v = self.ser_p[self.last_id]

                # print("value: ",v)
                value_txt = str(v)
                size = lv.point_t()
                lv.txt_get_size(size, value_txt, lv.font_default(), 0, 0, lv.COORD.
↪MAX, lv.TEXT_FLAG.NONE)

                a = lv.area_t()
                a.y2 = dsc.p1.y - 5
                a.y1 = a.y2 - size.y - 10
                a.x1 = dsc.p1.x + 10
                a.x2 = a.x1 + size.x + 10

                draw_rect_dsc = lv.draw_rect_dsc_t()
                draw_rect_dsc.init()
                draw_rect_dsc.bg_color = lv.palette_main(lv.PALETTE.BLUE)
                draw_rect_dsc.radius = 3

                lv.draw_rect(a, dsc.clip_area, draw_rect_dsc)

                draw_label_dsc = lv.draw_label_dsc_t()
                draw_label_dsc.init()
                draw_label_dsc.color = lv.color_white()
                a.x1 += 5
                a.x2 -= 5
                a.y1 += 5
                a.y2 -= 5
                lv.draw_label(a, dsc.clip_area, draw_label_dsc, value_txt, None)

example_chart_6 = ExampleChart_6()
```

Scatter chart

```
#include "../../lv_examples.h"
#if LV_USE_CHART && LV_BUILD_EXAMPLES

static void draw_event_cb(lv_event_t * e)
{
    lv_obj_draw_part_dsc_t * dsc = lv_event_get_draw_part_dsc(e);
    if(dsc->part == LV_PART_ITEMS) {
        lv_obj_t * obj = lv_event_get_target(e);
        lv_chart_series_t * ser = lv_chart_get_series_next(obj, NULL);
        uint32_t cnt = lv_chart_get_point_count(obj);
        /*Make older value more transparent*/
        dsc->rect_dsc->bg_opa = (LV_OPA_COVER *  dsc->id) / (cnt - 1);

        /*Make smaller values blue, higher values red*/
        lv_coord_t * x_array = lv_chart_get_x_array(obj, ser);
        lv_coord_t * y_array = lv_chart_get_y_array(obj, ser);
        /*dsc->id is the tells drawing order, but we need the ID of the point being␣
↪drawn.*/
        uint32_t start_point = lv_chart_get_x_start_point(obj, ser);
        uint32_t p_act = (start_point + dsc->id) % cnt; /*Consider start point to get␣
↪the index of the array*/
        lv_opa_t x_opa = (x_array[p_act] * LV_OPA_50) / 200;
        lv_opa_t y_opa = (y_array[p_act] * LV_OPA_50) / 1000;

        dsc->rect_dsc->bg_color = lv_color_mix(lv_palette_main(LV_PALETTE_RED),
                                               lv_palette_main(LV_PALETTE_BLUE),
                                               x_opa + y_opa);
    }
}

static void add_data(lv_timer_t * timer)
{
    LV_UNUSED(timer);
    lv_obj_t * chart = timer->user_data;
    lv_chart_set_next_value2(chart, lv_chart_get_series_next(chart, NULL), lv_rand(0,
↪200), lv_rand(0,1000));
}

/**
 * A scatter chart
 */
void lv_example_chart_7(void)
```

(下页继续)

(续上页)

```
{
    lv_obj_t * chart = lv_chart_create(lv_scr_act());
    lv_obj_set_size(chart, 200, 150);
    lv_obj_align(chart, LV_ALIGN_CENTER, 0, 0);
    lv_obj_add_event_cb(chart, draw_event_cb, LV_EVENT_DRAW_PART_BEGIN, NULL);
    lv_obj_set_style_line_width(chart, 0, LV_PART_ITEMS);   /*Remove the lines*/

    lv_chart_set_type(chart, LV_CHART_TYPE_SCATTER);

    lv_chart_set_axis_tick(chart, LV_CHART_AXIS_PRIMARY_X, 5, 5, 5, 1, true, 30);
    lv_chart_set_axis_tick(chart, LV_CHART_AXIS_PRIMARY_Y, 10, 5, 6, 5, true, 50);

    lv_chart_set_range(chart, LV_CHART_AXIS_PRIMARY_X, 0, 200);
    lv_chart_set_range(chart, LV_CHART_AXIS_PRIMARY_Y, 0, 1000);

    lv_chart_set_point_count(chart, 50);

    lv_chart_series_t * ser = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_
↪RED), LV_CHART_AXIS_PRIMARY_Y);
    uint32_t i;
    for(i = 0; i < 50; i++) {
        lv_chart_set_next_value2(chart, ser, lv_rand(0, 200), lv_rand(0, 1000));
    }

    lv_timer_create(add_data, 100, chart);
}

#endif
```

```
#!/opt/bin/lv_micropython -i
import utime as time
import lvgl as lv
import display_driver

def draw_event_cb(e):
    dsc = e.get_draw_part_dsc()
    if dsc.part == lv.PART.ITEMS:
        obj = e.get_target()
        ser = obj.get_series_next(None)
        cnt = obj.get_point_count()
        # print("cnt: ",cnt)
        # Make older value more transparent
        dsc.rect_dsc.bg_opa = (lv.OPA.COVER *  dsc.id) // (cnt - 1)
```

(下页继续)

(续上页)

```
        # Make smaller values blue, higher values red
        # x_array = chart.get_x_array(ser)
        # y_array = chart.get_y_array(ser)
        # dsc->id is the tells drawing order, but we need the ID of the point being␣
↪drawn.
        start_point = chart.get_x_start_point(ser)
        # print("start point: ",start_point)
        p_act = (start_point + dsc.id) % cnt # Consider start point to get the index␣
↪of the array
        # print("p_act", p_act)
        x_opa = (x_array[p_act] * lv.OPA._50) // 200
        y_opa = (y_array[p_act] * lv.OPA._50) // 1000

        dsc.rect_dsc.bg_color = lv.palette_main(lv.PALETTE.RED).color_mix(
                                            lv.palette_main(lv.PALETTE.BLUE),
                                            x_opa + y_opa)

def add_data(timer,chart):
    # print("add_data")
    x = lv.rand(0,200)
    y = lv.rand(0,1000)
    chart.set_next_value2(ser, x, y)
    # chart.set_next_value2(chart.gx, y)
    x_array.pop(0)
    x_array.append(x)
    y_array.pop(0)
    y_array.append(y)

#
# A scatter chart
#

chart = lv.chart(lv.scr_act())
chart.set_size(200, 150)
chart.align(lv.ALIGN.CENTER, 0, 0)
chart.add_event_cb(draw_event_cb, lv.EVENT.DRAW_PART_BEGIN, None)
chart.set_style_line_width(0, lv.PART.ITEMS)   # Remove the lines

chart.set_type(lv.chart.TYPE.SCATTER)

chart.set_axis_tick(lv.chart.AXIS.PRIMARY_X, 5, 5, 5, 1, True, 30)
chart.set_axis_tick(lv.chart.AXIS.PRIMARY_Y, 10, 5, 6, 5, True, 50)
```

(下页继续)

(续上页)

```
chart.set_range(lv.chart.AXIS.PRIMARY_X, 0, 200)
chart.set_range(lv.chart.AXIS.PRIMARY_Y, 0, 1000)

chart.set_point_count(50)

ser = chart.add_series(lv.palette_main(lv.PALETTE.RED), lv.chart.AXIS.PRIMARY_Y)

x_array = []
y_array = []
for i in range(50):
    x_array.append(lv.rand(0, 200))
    y_array.append(lv.rand(0, 1000))

ser.x_points = x_array
ser.y_points = y_array

# Create an `lv_timer` to update the chart.

timer = lv.timer_create_basic()
timer.set_period(100)
timer.set_cb(lambda src: add_data(timer,chart))
```

Stacked area chart

```
#include "../../lv_examples.h"
#if LV_USE_CHART && LV_DRAW_COMPLEX && LV_BUILD_EXAMPLES

/*  A struct is used to keep track of the series list because later we need to draw␣
↪to the series in the reverse order to which they were initialised. */
typedef struct
{
    lv_obj_t *obj;
    lv_chart_series_t *series_list[3];
} stacked_area_chart_t;

static stacked_area_chart_t stacked_area_chart;

/**
 * Callback which draws the blocks of colour under the lines
 **/
static void draw_event_cb(lv_event_t *e)
```

(下页继续)

(续上页)

```
{
    lv_obj_t *obj = lv_event_get_target(e);

    /*Add the faded area before the lines are drawn*/
    lv_obj_draw_part_dsc_t *dsc = lv_event_get_draw_part_dsc(e);
    if (dsc->part == LV_PART_ITEMS)
    {
        if (!dsc->p1 || !dsc->p2)
            return;

        /*Add a line mask that keeps the area below the line*/
        lv_draw_mask_line_param_t line_mask_param;
        lv_draw_mask_line_points_init(&line_mask_param, dsc->p1->x, dsc->p1->y, dsc->
↪p2->x, dsc->p2->y, LV_DRAW_MASK_LINE_SIDE_BOTTOM);
        int16_t line_mask_id = lv_draw_mask_add(&line_mask_param, NULL);

        /*Draw a rectangle that will be affected by the mask*/
        lv_draw_rect_dsc_t draw_rect_dsc;
        lv_draw_rect_dsc_init(&draw_rect_dsc);
        draw_rect_dsc.bg_opa = LV_OPA_COVER;
        draw_rect_dsc.bg_color = dsc->line_dsc->color;

        lv_area_t a;
        a.x1 = dsc->p1->x;
        a.x2 = dsc->p2->x;
        a.y1 = LV_MIN(dsc->p1->y, dsc->p2->y);
        a.y2 = obj->coords.y2 - 13; /* -13 cuts off where the rectangle draws over␣
↪the chart margin. Without this an area of 0 doesn't look like 0 */
        lv_draw_rect(dsc->draw_ctx, &draw_rect_dsc, &a);

        /*Remove the mask*/
        lv_draw_mask_free_param(&line_mask_param);
        lv_draw_mask_remove_id(line_mask_id);
    }
}

/**
 * Helper function to round a fixed point number
 **/
static int32_t round_fixed_point(int32_t n, int8_t shift)
{
    /* Create a bitmask to isolates the decimal part of the fixed point number */
    int32_t mask = 1;
```

(下页继续)

(续上页)

```
    for (int32_t bit_pos = 0; bit_pos < shift; bit_pos++)
    {
        mask = (mask << 1) + 1;
    }

    int32_t decimal_part = n & mask;

    /* Get 0.5 as fixed point */
    int32_t rounding_boundary = 1 << (shift - 1);

    /* Return either the integer part of n or the integer part + 1 */
    return (decimal_part < rounding_boundary) ? (n & ~mask) : ((n >> shift) + 1) <<␣
↪shift;
}

/**
 * Stacked area chart
 */
void lv_example_chart_8(void)
{
    /*Create a stacked_area_chart.obj*/
    stacked_area_chart.obj = lv_chart_create(lv_scr_act());
    lv_obj_set_size(stacked_area_chart.obj, 200, 150);
    lv_obj_center(stacked_area_chart.obj);
    lv_chart_set_type(stacked_area_chart.obj, LV_CHART_TYPE_LINE);
    lv_chart_set_div_line_count(stacked_area_chart.obj, 5, 7);
    lv_obj_add_event_cb(stacked_area_chart.obj, draw_event_cb, LV_EVENT_DRAW_PART_
↪BEGIN, NULL);

    /* Set range to 0 to 100 for percentages. Draw ticks */
    lv_chart_set_range(stacked_area_chart.obj,LV_CHART_AXIS_PRIMARY_Y,0,100);
    lv_chart_set_axis_tick(stacked_area_chart.obj, LV_CHART_AXIS_PRIMARY_Y, 3, 0, 5,␣
↪1, true, 30);

    /*Set point size to 0 so the lines are smooth */
    lv_obj_set_style_size(stacked_area_chart.obj, 0, LV_PART_INDICATOR);

    /*Add some data series*/
    stacked_area_chart.series_list[0] = lv_chart_add_series(stacked_area_chart.obj,␣
↪lv_palette_main(LV_PALETTE_RED), LV_CHART_AXIS_PRIMARY_Y);
    stacked_area_chart.series_list[1] = lv_chart_add_series(stacked_area_chart.obj,␣
↪lv_palette_main(LV_PALETTE_BLUE), LV_CHART_AXIS_PRIMARY_Y);
    stacked_area_chart.series_list[2] = lv_chart_add_series(stacked_area_chart.obj,␣
↪lv_palette_main(LV_PALETTE_GREEN), LV_CHART_AXIS_PRIMARY_Y);
```

(下页继续)

(续上页)

```
    for (int point = 0; point < 10; point++)
    {
        /* Make some random data */
        uint32_t vals[3] = {lv_rand(10, 20), lv_rand(20, 30), lv_rand(20, 30)};

        int8_t fixed_point_shift = 5;
        uint32_t total = vals[0] + vals[1] + vals[2];
        uint32_t draw_heights[3];
        uint32_t int_sum = 0;
        uint32_t decimal_sum = 0;

        /* Fixed point cascade rounding ensures percentages add to 100 */
        for (int32_t series_index = 0; series_index < 3; series_index++)
        {
            decimal_sum += (((vals[series_index] * 100) << fixed_point_shift) /␣
↪total);
            int_sum += (vals[series_index] * 100) / total;

            int32_t modifier = (round_fixed_point(decimal_sum, fixed_point_shift) >>␣
↪fixed_point_shift) - int_sum;

            /*  The draw heights are equal to the percentage of the total each value␣
↪is + the cumulative sum of the previous percentages.
                The accumulation is how the values get "stacked" */
            draw_heights[series_index] = int_sum + modifier;

            /*  Draw to the series in the reverse order to which they were␣
↪initialised.
                Without this the higher values will draw on top of the lower ones.
                This is because the Z-height of a series matches the order it was␣
↪initialised */
            lv_chart_set_next_value(stacked_area_chart.obj, stacked_area_chart.series_
↪list[3 - series_index - 1], draw_heights[series_index]);
        }
    }

    lv_chart_refresh(stacked_area_chart.obj);
}

#endif
```

```
import display_driver
```

(下页继续)

(续上页)

```
import lvgl as lv

# A class is used to keep track of the series list because later we
#  need to draw to the series in the reverse order to which they were initialised.
class StackedAreaChart:
    def __init__(self):
        self.obj = None
        self.series_list = [None, None, None]

stacked_area_chart = StackedAreaChart()

#
# Callback which draws the blocks of colour under the lines
#
def draw_event_cb(e):

    obj = e.get_target()
    cont_a = lv.area_t()
    obj.get_coords(cont_a)

    #Add the faded area before the lines are drawn
    dsc = e.get_draw_part_dsc()
    if dsc.part == lv.PART.ITEMS:
        if not dsc.p1 or not dsc.p2:
            return

        # Add a line mask that keeps the area below the line
        line_mask_param = lv.draw_mask_line_param_t()
        line_mask_param.points_init(dsc.p1.x, dsc.p1.y, dsc.p2.x, dsc.p2.y, lv.DRAW_
↪MASK_LINE_SIDE.BOTTOM)
        line_mask_id = lv.draw_mask_add(line_mask_param, None)

        #Draw a rectangle that will be affected by the mask
        draw_rect_dsc = lv.draw_rect_dsc_t()
        draw_rect_dsc.init()
        draw_rect_dsc.bg_opa = lv.OPA.COVER
        draw_rect_dsc.bg_color = dsc.line_dsc.color

        a = lv.area_t()
        a.x1 = dsc.p1.x
        a.x2 = dsc.p2.x
        a.y1 = min(dsc.p1.y, dsc.p2.y)
        a.y2 = cont_a.y2 - 13 # -13 cuts off where the rectangle draws over the chart␣
↪margin. Without this an area of 0 doesn't look like 0
```

(下页继续)

(续上页)

```
        dsc.draw_ctx.rect(draw_rect_dsc, a)

        # Remove the mask
        lv.draw_mask_free_param(line_mask_param)
        lv.draw_mask_remove_id(line_mask_id)


#
# Helper function to round a fixed point number
#
def round_fixed_point(n, shift):
    # Create a bitmask to isolates the decimal part of the fixed point number
    mask = 1
    for bit_pos in range(shift):
        mask = (mask << 1) + 1

    decimal_part = n & mask

    # Get 0.5 as fixed point
    rounding_boundary = 1 << (shift - 1)

    # Return either the integer part of n or the integer part + 1
    if decimal_part < rounding_boundary:
        return (n & ~mask)
    return ((n >> shift) + 1) << shift


#
# Stacked area chart
#
def lv_example_chart_8():

    #Create a stacked_area_chart.obj
    stacked_area_chart.obj = lv.chart(lv.scr_act())
    stacked_area_chart.obj.set_size(200, 150)
    stacked_area_chart.obj.center()
    stacked_area_chart.obj.set_type( lv.chart.TYPE.LINE)
    stacked_area_chart.obj.set_div_line_count(5, 7)
    stacked_area_chart.obj.add_event_cb( draw_event_cb, lv.EVENT.DRAW_PART_BEGIN,␣
↪None)

    # Set range to 0 to 100 for percentages. Draw ticks
    stacked_area_chart.obj.set_range(lv.chart.AXIS.PRIMARY_Y,0,100)
```

(下页继续)

(续上页)

```
    stacked_area_chart.obj.set_axis_tick(lv.chart.AXIS.PRIMARY_Y, 3, 0, 5, 1, True,␣
↪30)

    #Set point size to 0 so the lines are smooth
    stacked_area_chart.obj.set_style_size(0, lv.PART.INDICATOR)

    # Add some data series
    stacked_area_chart.series_list[0] = stacked_area_chart.obj.add_series(lv.palette_
↪main(lv.PALETTE.RED), lv.chart.AXIS.PRIMARY_Y)
    stacked_area_chart.series_list[1] = stacked_area_chart.obj.add_series(lv.palette_
↪main(lv.PALETTE.BLUE), lv.chart.AXIS.PRIMARY_Y)
    stacked_area_chart.series_list[2] = stacked_area_chart.obj.add_series(lv.palette_
↪main(lv.PALETTE.GREEN), lv.chart.AXIS.PRIMARY_Y)

    for point in range(10):
        # Make some random data
        vals = [lv.rand(10, 20), lv.rand(20, 30), lv.rand(20, 30)]

        fixed_point_shift = 5
        total = vals[0] + vals[1] + vals[2]
        draw_heights = [0, 0, 0]
        int_sum = 0
        decimal_sum = 0

        # Fixed point cascade rounding ensures percentages add to 100
        for series_index in range(3):
            decimal_sum += int(((vals[series_index] * 100) << fixed_point_shift) //␣
↪total)
            int_sum += int((vals[series_index] * 100) / total)

            modifier = (round_fixed_point(decimal_sum, fixed_point_shift) >> fixed_
↪point_shift) - int_sum

            #  The draw heights are equal to the percentage of the total each value␣
↪is + the cumulative sum of the previous percentages.
            #   The accumulation is how the values get "stacked"
            draw_heights[series_index] = int(int_sum + modifier)

            #  Draw to the series in the reverse order to which they were initialised.
            #   Without this the higher values will draw on top of the lower ones.
            #   This is because the Z-height of a series matches the order it was␣
↪initialised
            stacked_area_chart.obj.set_next_value( stacked_area_chart.series_list[3 -␣
↪series_index - 1], draw_heights[series_index])
```

(下页继续)

(续上页)

```
    stacked_area_chart.obj.refresh()

lv_example_chart_8()
```

Checkbox

Simple Checkboxes

```
#include "../../lv_examples.h"
#if LV_USE_CHECKBOX && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        const char * txt = lv_checkbox_get_text(obj);
        const char * state = lv_obj_get_state(obj) & LV_STATE_CHECKED ? "Checked" :
↪"Unchecked";
        LV_LOG_USER("%s: %s", txt, state);
    }
}

void lv_example_checkbox_1(void)
{
    lv_obj_set_flex_flow(lv_scr_act(), LV_FLEX_FLOW_COLUMN);
    lv_obj_set_flex_align(lv_scr_act(), LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_START, LV_
↪FLEX_ALIGN_CENTER);

    lv_obj_t * cb;
    cb = lv_checkbox_create(lv_scr_act());
    lv_checkbox_set_text(cb, "Apple");
    lv_obj_add_event_cb(cb, event_handler, LV_EVENT_ALL, NULL);

    cb = lv_checkbox_create(lv_scr_act());
    lv_checkbox_set_text(cb, "Banana");
    lv_obj_add_state(cb, LV_STATE_CHECKED);
    lv_obj_add_event_cb(cb, event_handler, LV_EVENT_ALL, NULL);

    cb = lv_checkbox_create(lv_scr_act());
    lv_checkbox_set_text(cb, "Lemon");
```

(下页继续)

(续上页)

```
    lv_obj_add_state(cb, LV_STATE_DISABLED);
    lv_obj_add_event_cb(cb, event_handler, LV_EVENT_ALL, NULL);

    cb = lv_checkbox_create(lv_scr_act());
    lv_obj_add_state(cb, LV_STATE_CHECKED | LV_STATE_DISABLED);
    lv_checkbox_set_text(cb, "Melon\nand a new line");
    lv_obj_add_event_cb(cb, event_handler, LV_EVENT_ALL, NULL);

    lv_obj_update_layout(cb);
}

#endif
```

```
def event_handler(e):
    code = e.get_code()
    obj = e.get_target()
    if code == lv.EVENT.VALUE_CHANGED:
        txt = obj.get_text()
        if obj.get_state() & lv.STATE.CHECKED:
            state = "Checked"
        else:
            state = "Unchecked"
        print(txt + ":" + state)


lv.scr_act().set_flex_flow(lv.FLEX_FLOW.COLUMN)
lv.scr_act().set_flex_align(lv.FLEX_ALIGN.CENTER, lv.FLEX_ALIGN.START, lv.FLEX_ALIGN.
↪CENTER)

cb = lv.checkbox(lv.scr_act())
cb.set_text("Apple")
cb.add_event_cb(event_handler, lv.EVENT.ALL, None)

cb = lv.checkbox(lv.scr_act())
cb.set_text("Banana")
cb.add_state(lv.STATE.CHECKED)
cb.add_event_cb(event_handler, lv.EVENT.ALL, None)

cb = lv.checkbox(lv.scr_act())
cb.set_text("Lemon")
cb.add_state(lv.STATE.DISABLED)
cb.add_event_cb(event_handler, lv.EVENT.ALL, None)
```

(下页继续)

(续上页)

```
cb = lv.checkbox(lv.scr_act())
cb.add_state(lv.STATE.CHECKED | lv.STATE.DISABLED)
cb.set_text("Melon")
cb.add_event_cb(event_handler, lv.EVENT.ALL, None)

cb.update_layout()
```

Checkboxes as radio buttons

```
#include "../../lv_examples.h"
#if LV_USE_CHECKBOX && LV_BUILD_EXAMPLES

static lv_style_t style_radio;
static lv_style_t style_radio_chk;
static uint32_t active_index_1 = 0;
static uint32_t active_index_2 = 0;

static void radio_event_handler(lv_event_t * e)
{
    uint32_t * active_id = lv_event_get_user_data(e);
    lv_obj_t * cont = lv_event_get_current_target(e);
    lv_obj_t * act_cb = lv_event_get_target(e);
    lv_obj_t * old_cb = lv_obj_get_child(cont, *active_id);

    /*Do nothing if the container was clicked*/
    if(act_cb == cont) return;

    lv_obj_clear_state(old_cb, LV_STATE_CHECKED);   /*Uncheck the previous radio␣
↪button*/
    lv_obj_add_state(act_cb, LV_STATE_CHECKED);     /*Uncheck the current radio␣
↪button*/

    *active_id = lv_obj_get_index(act_cb);

    LV_LOG_USER("Selected radio buttons: %d, %d", (int)active_index_1, (int)active_
↪index_2);
}


static void radiobutton_create(lv_obj_t * parent, const char * txt)
{
```

(下页继续)

(续上页)

```
    lv_obj_t * obj = lv_checkbox_create(parent);
    lv_checkbox_set_text(obj, txt);
    lv_obj_add_flag(obj, LV_OBJ_FLAG_EVENT_BUBBLE);
    lv_obj_add_style(obj, &style_radio, LV_PART_INDICATOR);
    lv_obj_add_style(obj, &style_radio_chk, LV_PART_INDICATOR | LV_STATE_CHECKED);
}

/**
 * Checkboxes as radio buttons
 */
void lv_example_checkbox_2(void)
{
    /* The idea is to enable `LV_OBJ_FLAG_EVENT_BUBBLE` on checkboxes and process the
     * `LV_EVENT_CLICKED` on the container.
     * A variable is passed as event user data where the index of the active
     * radiobutton is saved */


    lv_style_init(&style_radio);
    lv_style_set_radius(&style_radio, LV_RADIUS_CIRCLE);

    lv_style_init(&style_radio_chk);
    lv_style_set_bg_img_src(&style_radio_chk, NULL);

    uint32_t i;
    char buf[32];

    lv_obj_t * cont1 = lv_obj_create(lv_scr_act());
    lv_obj_set_flex_flow(cont1, LV_FLEX_FLOW_COLUMN);
    lv_obj_set_size(cont1, lv_pct(40), lv_pct(80));
    lv_obj_add_event_cb(cont1, radio_event_handler, LV_EVENT_CLICKED, &active_index_
↪1);

    for (i = 0;i < 5;i++) {
        lv_snprintf(buf, sizeof(buf), "A %d", (int)i + 1);
        radiobutton_create(cont1, buf);

    }
    /*Make the first checkbox checked*/
    lv_obj_add_state(lv_obj_get_child(cont1, 0), LV_STATE_CHECKED);

    lv_obj_t * cont2 = lv_obj_create(lv_scr_act());
    lv_obj_set_flex_flow(cont2, LV_FLEX_FLOW_COLUMN);
```

(下页继续)

(续上页)

```
    lv_obj_set_size(cont2, lv_pct(40), lv_pct(80));
    lv_obj_set_x(cont2, lv_pct(50));
    lv_obj_add_event_cb(cont2, radio_event_handler, LV_EVENT_CLICKED, &active_index_
↪2);

    for (i = 0;i < 3;i++) {
        lv_snprintf(buf, sizeof(buf), "B %d", (int)i + 1);
        radiobutton_create(cont2, buf);
    }

    /*Make the first checkbox checked*/
    lv_obj_add_state(lv_obj_get_child(cont2, 0), LV_STATE_CHECKED);
}


#endif
```

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs/100ask_lvgl_
↪docs/examples/widgets/checkbox/lv_example_checkbox_2.py
```

Colorwheel

Simple Colorwheel

```
#include "../../lv_examples.h"
#if LV_USE_COLORWHEEL && LV_BUILD_EXAMPLES

void lv_example_colorwheel_1(void)
{
    lv_obj_t * cw;

    cw = lv_colorwheel_create(lv_scr_act(), true);
    lv_obj_set_size(cw, 200, 200);
    lv_obj_center(cw);
}

#endif
```

```
cw = lv.colorwheel(lv.scr_act(), True)
cw.set_size(200, 200)
cw.center()
```

Dropdown

Simple Drop down list

```c
#include "../../lv_examples.h"
#if LV_USE_DROPDOWN && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        char buf[32];
        lv_dropdown_get_selected_str(obj, buf, sizeof(buf));
        LV_LOG_USER("Option: %s", buf);
    }
}

void lv_example_dropdown_1(void)
{

    /*Create a normal drop down list*/
    lv_obj_t * dd = lv_dropdown_create(lv_scr_act());
    lv_dropdown_set_options(dd, "Apple\n"
                                "Banana\n"
                                "Orange\n"
                                "Cherry\n"
                                "Grape\n"
                                "Raspberry\n"
                                "Melon\n"
                                "Orange\n"
                                "Lemon\n"
                                "Nuts");

    lv_obj_align(dd, LV_ALIGN_TOP_MID, 0, 20);
    lv_obj_add_event_cb(dd, event_handler, LV_EVENT_ALL, NULL);
}

#endif
```

```python
def event_handler(e):
    code = e.get_code()
    obj = e.get_target()
    if code == lv.EVENT.VALUE_CHANGED:
```

(下页继续)

(续上页)

```
        option = " "*10 # should be large enough to store the option
        obj.get_selected_str(option, len(option))
        # .strip() removes trailing spaces
        print("Option: \"%s\"" % option.strip())

# Create a normal drop down list
dd = lv.dropdown(lv.scr_act())
dd.set_options("\n".join([
    "Apple",
    "Banana",
    "Orange",
    "Cherry",
    "Grape",
    "Raspberry",
    "Melon",
    "Orange",
    "Lemon",
    "Nuts"]))

dd.align(lv.ALIGN.TOP_MID, 0, 20)
dd.add_event_cb(event_handler, lv.EVENT.ALL, None)
```

Drop down in four directions

```
#include "../../lv_examples.h"
#if LV_USE_DROPDOWN && LV_BUILD_EXAMPLES


/**
 * Create a drop down, up, left and right menus
 */
void lv_example_dropdown_2(void)
{
    static const char * opts = "Apple\n"
                               "Banana\n"
                               "Orange\n"
                               "Melon";

    lv_obj_t * dd;
    dd = lv_dropdown_create(lv_scr_act());
    lv_dropdown_set_options_static(dd, opts);
```

(下页继续)

(续上页)

```
    lv_obj_align(dd, LV_ALIGN_TOP_MID, 0, 10);

    dd = lv_dropdown_create(lv_scr_act());
    lv_dropdown_set_options_static(dd, opts);
    lv_dropdown_set_dir(dd, LV_DIR_BOTTOM);
    lv_dropdown_set_symbol(dd, LV_SYMBOL_UP);
    lv_obj_align(dd, LV_ALIGN_BOTTOM_MID, 0, -10);

    dd = lv_dropdown_create(lv_scr_act());
    lv_dropdown_set_options_static(dd, opts);
    lv_dropdown_set_dir(dd, LV_DIR_RIGHT);
    lv_dropdown_set_symbol(dd, LV_SYMBOL_RIGHT);
    lv_obj_align(dd, LV_ALIGN_LEFT_MID, 10, 0);

    dd = lv_dropdown_create(lv_scr_act());
    lv_dropdown_set_options_static(dd, opts);
    lv_dropdown_set_dir(dd, LV_DIR_LEFT);
    lv_dropdown_set_symbol(dd, LV_SYMBOL_LEFT);
    lv_obj_align(dd, LV_ALIGN_RIGHT_MID, -10, 0);
}

#endif
```

```
#
# Create a drop down, up, left and right menus
#

opts = "\n".join([
    "Apple",
    "Banana",
    "Orange",
    "Melon",
    "Grape",
    "Raspberry"])

dd = lv.dropdown(lv.scr_act())
dd.set_options_static(opts)
dd.align(lv.ALIGN.TOP_MID, 0, 10)
dd = lv.dropdown(lv.scr_act())
dd.set_options_static(opts)
dd.set_dir(lv.DIR.BOTTOM)
dd.set_symbol(lv.SYMBOL.UP)
dd.align(lv.ALIGN.BOTTOM_MID, 0, -10)
```

(下页继续)

(续上页)

```
dd = lv.dropdown(lv.scr_act())
dd.set_options_static(opts)
dd.set_dir(lv.DIR.RIGHT)
dd.set_symbol(lv.SYMBOL.RIGHT)
dd.align(lv.ALIGN.LEFT_MID, 10, 0)

dd = lv.dropdown(lv.scr_act())
dd.set_options_static(opts)
dd.set_dir(lv.DIR.LEFT)
dd.set_symbol(lv.SYMBOL.LEFT)
dd.align(lv.ALIGN.RIGHT_MID, -10, 0)
```

Menu

```
#include "../../lv_examples.h"
#if LV_USE_DROPDOWN && LV_BUILD_EXAMPLES

static void event_cb(lv_event_t * e)
{
    lv_obj_t * dropdown = lv_event_get_target(e);
    char buf[64];
    lv_dropdown_get_selected_str(dropdown, buf, sizeof(buf));
    LV_LOG_USER("'%s' is selected", buf);
}

/**
 * Create a menu from a drop-down list and show some drop-down list features and␣
↪styling
 */
void lv_example_dropdown_3(void)
{
    /*Create a drop down list*/
    lv_obj_t * dropdown = lv_dropdown_create(lv_scr_act());
    lv_obj_align(dropdown, LV_ALIGN_TOP_LEFT, 10, 10);
    lv_dropdown_set_options(dropdown, "New project\n"
                                      "New file\n"
                                      "Save\n"
                                      "Save as ...\n"
                                      "Open project\n"
```

(下页继续)

(续上页)

```
                                    "Recent projects\n"
                                    "Preferences\n"
                                    "Exit");

    /*Set a fixed text to display on the button of the drop-down list*/
    lv_dropdown_set_text(dropdown, "Menu");

    /*Use a custom image as down icon and flip it when the list is opened*/
    LV_IMG_DECLARE(img_caret_down)
    lv_dropdown_set_symbol(dropdown, &img_caret_down);
    lv_obj_set_style_transform_angle(dropdown, 1800, LV_PART_INDICATOR | LV_STATE_
↪CHECKED);

    /*In a menu we don't need to show the last clicked item*/
    lv_dropdown_set_selected_highlight(dropdown, false);

    lv_obj_add_event_cb(dropdown, event_cb, LV_EVENT_VALUE_CHANGED, NULL);
}

#endif
```

```
from imagetools import get_png_info, open_png

# Register PNG image decoder
decoder = lv.img.decoder_create()
decoder.info_cb = get_png_info
decoder.open_cb = open_png

# Create an image from the png file
try:
    with open('../../assets/img_caret_down.png','rb') as f:
        png_data = f.read()
except:
    print("Could not find img_caret_down.png")
    sys.exit()

img_caret_down_argb = lv.img_dsc_t({
  'data_size': len(png_data),
  'data': png_data
})

def event_cb(e):
    dropdown = e.get_target()
```

(下页继续)

(续上页)

```
    option = " "*64 # should be large enough to store the option
    dropdown.get_selected_str(option, len(option))
    print(option.strip() +" is selected")
#
# Create a menu from a drop-down list and show some drop-down list features and␣
↪styling
#

# Create a drop down list
dropdown = lv.dropdown(lv.scr_act())
dropdown.align(lv.ALIGN.TOP_LEFT, 10, 10)
dropdown.set_options("\n".join([
    "New project",
    "New file",
    "Open project",
    "Recent projects",
    "Preferences",
    "Exit"]))

# Set a fixed text to display on the button of the drop-down list
dropdown.set_text("Menu")

# Use a custom image as down icon and flip it when the list is opened
# LV_IMG_DECLARE(img_caret_down)
dropdown.set_symbol(img_caret_down_argb)
dropdown.set_style_transform_angle(1800, lv.PART.INDICATOR | lv.STATE.CHECKED)

# In a menu we don't need to show the last clicked item
dropdown.set_selected_highlight(False)

dropdown.add_event_cb(event_cb, lv.EVENT.VALUE_CHANGED, None)
```

Image

Image from variable and symbol

```
#include "../../lv_examples.h"
#if LV_USE_IMG && LV_BUILD_EXAMPLES


void lv_example_img_1(void)
```

(下页继续)

(续上页)

```
{
    LV_IMG_DECLARE(img_cogwheel_argb);
    lv_obj_t * img1 = lv_img_create(lv_scr_act());
    lv_img_set_src(img1, &img_cogwheel_argb);
    lv_obj_align(img1, LV_ALIGN_CENTER, 0, -20);
    lv_obj_set_size(img1, 200, 200);

    lv_obj_t * img2 = lv_img_create(lv_scr_act());
    lv_img_set_src(img2, LV_SYMBOL_OK "Accept");
    lv_obj_align_to(img2, img1, LV_ALIGN_OUT_BOTTOM_MID, 0, 20);
}

#endif
```

```
#!/opt/bin/lv_micropython -i
import usys as sys
import lvgl as lv
import display_driver
from imagetools import get_png_info, open_png

# Register PNG image decoder
decoder = lv.img.decoder_create()
decoder.info_cb = get_png_info
decoder.open_cb = open_png

# Create an image from the png file
try:
    with open('../../assets/img_cogwheel_argb.png','rb') as f:
        png_data = f.read()
except:
    print("Could not find img_cogwheel_argb.png")
    sys.exit()

img_cogwheel_argb = lv.img_dsc_t({
  'data_size': len(png_data),
  'data': png_data
})

img1 = lv.img(lv.scr_act())
img1.set_src(img_cogwheel_argb)
img1.align(lv.ALIGN.CENTER, 0, -20)
img1.set_size(200, 200)
```

(下页继续)

(续上页)

```
img2 = lv.img(lv.scr_act())
img2.set_src(lv.SYMBOL.OK + "Accept")
img2.align_to(img1, lv.ALIGN.OUT_BOTTOM_MID, 0, 20)
```

Image recoloring

```
#include "../../lv_examples.h"
#if LV_USE_IMG && LV_USE_SLIDER && LV_BUILD_EXAMPLES

static lv_obj_t * create_slider(lv_color_t color);
static void slider_event_cb(lv_event_t * e);

static lv_obj_t * red_slider, * green_slider, * blue_slider, * intense_slider;
static lv_obj_t * img1;


/**
 * Demonstrate runtime image re-coloring
 */
void lv_example_img_2(void)
{
    /*Create 4 sliders to adjust RGB color and re-color intensity*/
    red_slider = create_slider(lv_palette_main(LV_PALETTE_RED));
    green_slider = create_slider(lv_palette_main(LV_PALETTE_GREEN));
    blue_slider = create_slider(lv_palette_main(LV_PALETTE_BLUE));
    intense_slider = create_slider(lv_palette_main(LV_PALETTE_GREY));

    lv_slider_set_value(red_slider, LV_OPA_20, LV_ANIM_OFF);
    lv_slider_set_value(green_slider, LV_OPA_90, LV_ANIM_OFF);
    lv_slider_set_value(blue_slider, LV_OPA_60, LV_ANIM_OFF);
    lv_slider_set_value(intense_slider, LV_OPA_50, LV_ANIM_OFF);

    lv_obj_align(red_slider, LV_ALIGN_LEFT_MID, 25, 0);
    lv_obj_align_to(green_slider, red_slider, LV_ALIGN_OUT_RIGHT_MID, 25, 0);
    lv_obj_align_to(blue_slider, green_slider, LV_ALIGN_OUT_RIGHT_MID, 25, 0);
    lv_obj_align_to(intense_slider, blue_slider, LV_ALIGN_OUT_RIGHT_MID, 25, 0);

    /*Now create the actual image*/
    LV_IMG_DECLARE(img_cogwheel_argb)
    img1 = lv_img_create(lv_scr_act());
    lv_img_set_src(img1, &img_cogwheel_argb);
    lv_obj_align(img1, LV_ALIGN_RIGHT_MID, -20, 0);
```

(下页继续)

(续上页)

```
    lv_event_send(intense_slider, LV_EVENT_VALUE_CHANGED, NULL);
}

static void slider_event_cb(lv_event_t * e)
{
    LV_UNUSED(e);

    /*Recolor the image based on the sliders' values*/
    lv_color_t color  = lv_color_make(lv_slider_get_value(red_slider), lv_slider_get_
↪value(green_slider), lv_slider_get_value(blue_slider));
    lv_opa_t intense = lv_slider_get_value(intense_slider);
    lv_obj_set_style_img_recolor_opa(img1, intense, 0);
    lv_obj_set_style_img_recolor(img1, color, 0);
}

static lv_obj_t * create_slider(lv_color_t color)
{
    lv_obj_t * slider = lv_slider_create(lv_scr_act());
    lv_slider_set_range(slider, 0, 255);
    lv_obj_set_size(slider, 10, 200);
    lv_obj_set_style_bg_color(slider, color, LV_PART_KNOB);
    lv_obj_set_style_bg_color(slider, lv_color_darken(color, LV_OPA_40), LV_PART_
↪INDICATOR);
    lv_obj_add_event_cb(slider, slider_event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    return slider;
}

#endif
```

```
#!/opt/bin/lv_micropython -i
import usys as sys
import lvgl as lv
import display_driver
from imagetools import get_png_info, open_png

# Register PNG image decoder
decoder = lv.img.decoder_create()
decoder.info_cb = get_png_info
decoder.open_cb = open_png

# Create an image from the png file
try:
```

(下页继续)

(续上页)

```
    with open('../../assets/img_cogwheel_argb.png','rb') as f:
        png_data = f.read()
except:
    print("Could not find img_cogwheel_argb.png")
    sys.exit()

img_cogwheel_argb = lv.img_dsc_t({
  'data_size': len(png_data),
  'data': png_data
})

def create_slider(color):
    slider = lv.slider(lv.scr_act())
    slider.set_range(0, 255)
    slider.set_size(10, 200)
    slider.set_style_bg_color(color, lv.PART.KNOB)
    slider.set_style_bg_color(color.color_darken(lv.OPA._40), lv.PART.INDICATOR)
    slider.add_event_cb(slider_event_cb, lv.EVENT.VALUE_CHANGED, None)
    return slider

def slider_event_cb(e):
    # Recolor the image based on the sliders' values
    color  = lv.color_make(red_slider.get_value(), green_slider.get_value(), blue_
↪slider.get_value())
    intense = intense_slider.get_value()
    img1.set_style_img_recolor_opa(intense, 0)
    img1.set_style_img_recolor(color, 0)

#
# Demonstrate runtime image re-coloring
#
# Create 4 sliders to adjust RGB color and re-color intensity
red_slider = create_slider(lv.palette_main(lv.PALETTE.RED))
green_slider = create_slider(lv.palette_main(lv.PALETTE.GREEN))
blue_slider = create_slider(lv.palette_main(lv.PALETTE.BLUE))
intense_slider = create_slider(lv.palette_main(lv.PALETTE.GREY))

red_slider.set_value(lv.OPA._20, lv.ANIM.OFF)
green_slider.set_value(lv.OPA._90, lv.ANIM.OFF)
blue_slider.set_value(lv.OPA._60, lv.ANIM.OFF)
intense_slider.set_value(lv.OPA._50, lv.ANIM.OFF)

red_slider.align(lv.ALIGN.LEFT_MID, 25, 0)
```

(下页继续)

(续上页)

```
green_slider.align_to(red_slider, lv.ALIGN.OUT_RIGHT_MID, 25, 0)
blue_slider.align_to(green_slider, lv.ALIGN.OUT_RIGHT_MID, 25, 0)
intense_slider.align_to(blue_slider, lv.ALIGN.OUT_RIGHT_MID, 25, 0)

# Now create the actual image
img1 = lv.img(lv.scr_act())
img1.set_src(img_cogwheel_argb)
img1.align(lv.ALIGN.RIGHT_MID, -20, 0)

lv.event_send(intense_slider, lv.EVENT.VALUE_CHANGED, None)
```

Rotate and zoom

```
#include "../../lv_examples.h"
#if LV_USE_IMG && LV_BUILD_EXAMPLES

static void set_angle(void * img, int32_t v)
{
    lv_img_set_angle(img, v);
}

static void set_zoom(void * img, int32_t v)
{
    lv_img_set_zoom(img, v);
}


/**
 * Show transformations (zoom and rotation) using a pivot point.
 */
void lv_example_img_3(void)
{
    LV_IMG_DECLARE(img_cogwheel_argb);

    /*Now create the actual image*/
    lv_obj_t * img = lv_img_create(lv_scr_act());
    lv_img_set_src(img, &img_cogwheel_argb);
```

(下页继续)

(续上页)

```
    lv_obj_align(img, LV_ALIGN_CENTER, 50, 50);
    lv_img_set_pivot(img, 0, 0);    /*Rotate around the top left corner*/

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_var(&a, img);
    lv_anim_set_exec_cb(&a, set_angle);
    lv_anim_set_values(&a, 0, 3600);
    lv_anim_set_time(&a, 5000);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
    lv_anim_start(&a);

    lv_anim_set_exec_cb(&a, set_zoom);
    lv_anim_set_values(&a, 128, 256);
    lv_anim_set_playback_time(&a, 3000);
    lv_anim_start(&a);
}

#endif
```

```
#!/opt/bin/lv_micropython -i
import usys as sys
import lvgl as lv
import display_driver
from imagetools import get_png_info, open_png

# Register PNG image decoder
decoder = lv.img.decoder_create()
decoder.info_cb = get_png_info
decoder.open_cb = open_png

# Create an image from the png file
try:
    with open('../../assets/img_cogwheel_argb.png','rb') as f:
        png_data = f.read()
except:
    print("Could not find img_cogwheel_argb.png")
    sys.exit()

img_cogwheel_argb = lv.img_dsc_t({
  'data_size': len(png_data),
  'data': png_data
})
```

(下页继续)

(续上页)

```
def set_angle(img, v):
    img.set_angle(v)

def set_zoom(img, v):
    img.set_zoom(v)


#
# Show transformations (zoom and rotation) using a pivot point.
#

# Now create the actual image
img = lv.img(lv.scr_act())
img.set_src(img_cogwheel_argb)
img.align(lv.ALIGN.CENTER, 50, 50)
img.set_pivot(0, 0)               # Rotate around the top left corner

a1 = lv.anim_t()
a1.init()
a1.set_var(img)
a1.set_custom_exec_cb(lambda a,val: set_angle(img,val))
a1.set_values(0, 3600)
a1.set_time(5000)
a1.set_repeat_count(lv.ANIM_REPEAT.INFINITE)
lv.anim_t.start(a1)

a2 = lv.anim_t()
a2.init()
a2.set_var(img)
a2.set_custom_exec_cb(lambda a,val: set_zoom(img,val))
a2.set_values(128, 256)
a2.set_time(5000)
a2.set_playback_time(3000)
a2.set_repeat_count(lv.ANIM_REPEAT.INFINITE)
lv.anim_t.start(a2)
```

Image offset and styling

```
#include "../../lv_examples.h"
#if LV_USE_IMG && LV_BUILD_EXAMPLES

static void ofs_y_anim(void * img, int32_t v)
{
    lv_img_set_offset_y(img, v);
}

/**
 * Image styling and offset
 */
void lv_example_img_4(void)
{
    LV_IMG_DECLARE(img_skew_strip);

    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_bg_color(&style, lv_palette_main(LV_PALETTE_YELLOW));
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_img_recolor_opa(&style, LV_OPA_COVER);
    lv_style_set_img_recolor(&style, lv_color_black());

    lv_obj_t * img = lv_img_create(lv_scr_act());
    lv_obj_add_style(img, &style, 0);
    lv_img_set_src(img, &img_skew_strip);
    lv_obj_set_size(img, 150, 100);
    lv_obj_center(img);

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_var(&a, img);
    lv_anim_set_exec_cb(&a, ofs_y_anim);
    lv_anim_set_values(&a, 0, 100);
    lv_anim_set_time(&a, 3000);
    lv_anim_set_playback_time(&a, 500);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
    lv_anim_start(&a);

}

#endif
```

```
from imagetools import get_png_info, open_png

def ofs_y_anim(img, v):
    img.set_offset_y(v)
    # print(img,v)

# Register PNG image decoder
decoder = lv.img.decoder_create()
decoder.info_cb = get_png_info
decoder.open_cb = open_png

# Create an image from the png file
try:
    with open('../../assets/img_skew_strip.png','rb') as f:
        png_data = f.read()
except:
    print("Could not find img_skew_strip.png")
    sys.exit()

img_skew_strip = lv.img_dsc_t({
  'data_size': len(png_data),
  'data': png_data
})

#
# Image styling and offset
#

style = lv.style_t()
style.init()
style.set_bg_color(lv.palette_main(lv.PALETTE.YELLOW))
style.set_bg_opa(lv.OPA.COVER)
style.set_img_recolor_opa(lv.OPA.COVER)
style.set_img_recolor(lv.color_black())

img = lv.img(lv.scr_act())
img.add_style(style, 0)
img.set_src(img_skew_strip)
img.set_size(150, 100)
img.center()

a = lv.anim_t()
a.init()
a.set_var(img)
```

(下页继续)

(续上页)

```
a.set_values(0, 100)
a.set_time(3000)
a.set_playback_time(500)
a.set_repeat_count(lv.ANIM_REPEAT.INFINITE)
a.set_custom_exec_cb(lambda a,val: ofs_y_anim(img,val))
lv.anim_t.start(a)
```

Image button

Simple Image button

```
#include "../../lv_examples.h"
#if LV_USE_IMGBTN && LV_BUILD_EXAMPLES

void lv_example_imgbtn_1(void)
{
    LV_IMG_DECLARE(imgbtn_left);
    LV_IMG_DECLARE(imgbtn_right);
    LV_IMG_DECLARE(imgbtn_mid);

    /*Create a transition animation on width transformation and recolor.*/
    static lv_style_prop_t tr_prop[] = {LV_STYLE_TRANSFORM_WIDTH, LV_STYLE_IMG_
↪RECOLOR_OPA, 0};
    static lv_style_transition_dsc_t tr;
    lv_style_transition_dsc_init(&tr, tr_prop, lv_anim_path_linear, 200, 0, NULL);

    static lv_style_t style_def;
    lv_style_init(&style_def);
    lv_style_set_text_color(&style_def, lv_color_white());
    lv_style_set_transition(&style_def, &tr);

    /*Darken the button when pressed and make it wider*/
    static lv_style_t style_pr;
    lv_style_init(&style_pr);
    lv_style_set_img_recolor_opa(&style_pr, LV_OPA_30);
    lv_style_set_img_recolor(&style_pr, lv_color_black());
    lv_style_set_transform_width(&style_pr, 20);

    /*Create an image button*/
    lv_obj_t * imgbtn1 = lv_imgbtn_create(lv_scr_act());
    lv_imgbtn_set_src(imgbtn1, LV_IMGBTN_STATE_RELEASED, &imgbtn_left, &imgbtn_mid, &
↪imgbtn_right);
```

(下页继续)

(续上页)

```
    lv_obj_add_style(imgbtn1, &style_def, 0);
    lv_obj_add_style(imgbtn1, &style_pr, LV_STATE_PRESSED);

    lv_obj_align(imgbtn1, LV_ALIGN_CENTER, 0, 0);

    /*Create a label on the image button*/
    lv_obj_t * label = lv_label_create(imgbtn1);
    lv_label_set_text(label, "Button");
    lv_obj_align(label, LV_ALIGN_CENTER, 0, -4);
}

#endif
```

```
from imagetools import get_png_info, open_png

# Register PNG image decoder
decoder = lv.img.decoder_create()
decoder.info_cb = get_png_info
decoder.open_cb = open_png

# Create an image from the png file
try:
    with open('../../assets/imgbtn_left.png','rb') as f:
        imgbtn_left_data = f.read()
except:
    print("Could not find imgbtn_left.png")
    sys.exit()

imgbtn_left_dsc = lv.img_dsc_t({
  'data_size': len(imgbtn_left_data),
  'data': imgbtn_left_data
})

try:
    with open('../../assets/imgbtn_mid.png','rb') as f:
        imgbtn_mid_data = f.read()
except:
    print("Could not find imgbtn_mid.png")
    sys.exit()

imgbtn_mid_dsc = lv.img_dsc_t({
  'data_size': len(imgbtn_mid_data),
  'data': imgbtn_mid_data
```

(下页继续)

(续上页)

```
})

try:
    with open('../../assets/imgbtn_right.png','rb') as f:
        imgbtn_right_data = f.read()
except:
    print("Could not find imgbtn_right.png")
    sys.exit()

imgbtn_right_dsc = lv.img_dsc_t({
  'data_size': len(imgbtn_right_data),
  'data': imgbtn_right_data
})

# Create a transition animation on width transformation and recolor.
tr_prop = [lv.STYLE.TRANSFORM_WIDTH, lv.STYLE.IMG_RECOLOR_OPA, 0]
tr = lv.style_transition_dsc_t()
tr.init(tr_prop, lv.anim_t.path_linear, 200, 0, None)

style_def = lv.style_t()
style_def.init()
style_def.set_text_color(lv.color_white())
style_def.set_transition(tr)

# Darken the button when pressed and make it wider
style_pr = lv.style_t()
style_pr.init()
style_pr.set_img_recolor_opa(lv.OPA._30)
style_pr.set_img_recolor(lv.color_black())
style_pr.set_transform_width(20)

# Create an image button
imgbtn1 = lv.imgbtn(lv.scr_act())
imgbtn1.set_src(lv.imgbtn.STATE.RELEASED, imgbtn_left_dsc, imgbtn_mid_dsc, imgbtn_
↪right_dsc)
imgbtn1.add_style(style_def, 0)
imgbtn1.add_style(style_pr, lv.STATE.PRESSED)

imgbtn1.align(lv.ALIGN.CENTER, 0, 0)

# Create a label on the image button
label = lv.label(imgbtn1)
label.set_text("Button")
```

(下页继续)

(续上页)

```
label.align(lv.ALIGN.CENTER, 0, -4)
```

Keyboard

Keyboard with text area

```
#include "../../lv_examples.h"
#if LV_USE_KEYBOARD && LV_BUILD_EXAMPLES

static void ta_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * ta = lv_event_get_target(e);
    lv_obj_t * kb = lv_event_get_user_data(e);
    if(code == LV_EVENT_FOCUSED) {
        lv_keyboard_set_textarea(kb, ta);
        lv_obj_clear_flag(kb, LV_OBJ_FLAG_HIDDEN);
    }

    if(code == LV_EVENT_DEFOCUSED) {
        lv_keyboard_set_textarea(kb, NULL);
        lv_obj_add_flag(kb, LV_OBJ_FLAG_HIDDEN);
    }
}

void lv_example_keyboard_1(void)
{
    /*Create a keyboard to use it with an of the text areas*/
    lv_obj_t *kb = lv_keyboard_create(lv_scr_act());

    /*Create a text area. The keyboard will write here*/
    lv_obj_t * ta;
    ta = lv_textarea_create(lv_scr_act());
    lv_obj_align(ta, LV_ALIGN_TOP_LEFT, 10, 10);
    lv_obj_add_event_cb(ta, ta_event_cb, LV_EVENT_ALL, kb);
    lv_textarea_set_placeholder_text(ta, "Hello");
    lv_obj_set_size(ta, 140, 80);

    ta = lv_textarea_create(lv_scr_act());
    lv_obj_align(ta, LV_ALIGN_TOP_RIGHT, -10, 10);
    lv_obj_add_event_cb(ta, ta_event_cb, LV_EVENT_ALL, kb);
```

(下页继续)

(续上页)

```
    lv_obj_set_size(ta, 140, 80);

    lv_keyboard_set_textarea(kb, ta);
}
#endif
```

```
def ta_event_cb(e,kb):
    code = e.get_code()
    ta = e.get_target()
    if code == lv.EVENT.FOCUSED:
        kb.set_textarea(ta)
        kb.clear_flag(lv.obj.FLAG.HIDDEN)

    if code == lv.EVENT.DEFOCUSED:
        kb.set_textarea(None)
        kb.add_flag(lv.obj.FLAG.HIDDEN)

# Create a keyboard to use it with one of the text areas
kb = lv.keyboard(lv.scr_act())

# Create a text area. The keyboard will write here
ta = lv.textarea(lv.scr_act())
ta.set_width(200)
ta.align(lv.ALIGN.TOP_LEFT, 10, 10)
ta.add_event_cb(lambda e: ta_event_cb(e,kb), lv.EVENT.ALL, None)
ta.set_placeholder_text("Hello")

ta = lv.textarea(lv.scr_act())
ta.set_width(200)
ta.align(lv.ALIGN.TOP_RIGHT, -10, 10)
ta.add_event_cb(lambda e: ta_event_cb(e,kb), lv.EVENT.ALL, None)

kb.set_textarea(ta)
```

Label

Line wrap, recoloring and scrolling

```
#include "../../lv_examples.h"
#if LV_USE_LABEL && LV_BUILD_EXAMPLES

/**
 * Show line wrap, re-color, line align and text scrolling.
 */
void lv_example_label_1(void)
{
    lv_obj_t * label1 = lv_label_create(lv_scr_act());
    lv_label_set_long_mode(label1, LV_LABEL_LONG_WRAP);     /*Break the long lines*/
    lv_label_set_recolor(label1, true);                      /*Enable re-coloring by␣
↪commands in the text*/
    lv_label_set_text(label1, "#0000ff Re-color# #ff00ff words# #ff0000 of a# label,␣
↪align the lines to the center "
                              "and wrap long text automatically.");
    lv_obj_set_width(label1, 150);  /*Set smaller width to make the lines wrap*/
    lv_obj_set_style_text_align(label1, LV_TEXT_ALIGN_CENTER, 0);
    lv_obj_align(label1, LV_ALIGN_CENTER, 0, -40);

    lv_obj_t * label2 = lv_label_create(lv_scr_act());
    lv_label_set_long_mode(label2, LV_LABEL_LONG_SCROLL_CIRCULAR);     /*Circular␣
↪scroll*/
    lv_obj_set_width(label2, 150);
    lv_label_set_text(label2, "It is a circularly scrolling text. ");
    lv_obj_align(label2, LV_ALIGN_CENTER, 0, 40);
}

#endif
```

```
#
# Show line wrap, re-color, line align and text scrolling.
#
label1 = lv.label(lv.scr_act())
label1.set_long_mode(lv.label.LONG.WRAP)      # Break the long lines*/
label1.set_recolor(True)                      # Enable re-coloring by commands in the␣
↪text
label1.set_text("#0000ff Re-color# #ff00ff words# #ff0000 of a# label, align the␣
↪lines to the center"
                              "and  wrap long text automatically.")
label1.set_width(150)                         # Set smaller width to make the lines␣
↪wrap
```

(下页继续)

(续上页)

```
label1.set_style_text_align(lv.ALIGN.CENTER, 0)
label1.align(lv.ALIGN.CENTER, 0, -40)


label2 = lv.label(lv.scr_act())
label2.set_long_mode(lv.label.LONG.SCROLL_CIRCULAR) # Circular scroll
label2.set_width(150)
label2.set_text("It is a circularly scrolling text. ")
label2.align(lv.ALIGN.CENTER, 0, 40)
```

Text shadow

```
#include "../../lv_examples.h"
#if LV_USE_LABEL && LV_BUILD_EXAMPLES

/**
 * Create a fake text shadow
 */
void lv_example_label_2(void)
{
    /*Create a style for the shadow*/
    static lv_style_t style_shadow;
    lv_style_init(&style_shadow);
    lv_style_set_text_opa(&style_shadow, LV_OPA_30);
    lv_style_set_text_color(&style_shadow, lv_color_black());

    /*Create a label for the shadow first (it's in the background)*/
    lv_obj_t * shadow_label = lv_label_create(lv_scr_act());
    lv_obj_add_style(shadow_label, &style_shadow, 0);

    /*Create the main label*/
    lv_obj_t * main_label = lv_label_create(lv_scr_act());
    lv_label_set_text(main_label, "A simple method to create\n"
                                  "shadows on a text.\n"
                                  "It even works with\n\n"
                                  "newlines     and spaces.");

    /*Set the same text for the shadow label*/
    lv_label_set_text(shadow_label, lv_label_get_text(main_label));

    /*Position the main label*/
```

(下页继续)

(续上页)

```
    lv_obj_align(main_label, LV_ALIGN_CENTER, 0, 0);

    /*Shift the second label down and to the right by 2 pixel*/
    lv_obj_align_to(shadow_label, main_label, LV_ALIGN_TOP_LEFT, 2, 2);
}

#endif
```

```
#
# Create a fake text shadow
#

# Create a style for the shadow
style_shadow = lv.style_t()
style_shadow.init()
style_shadow.set_text_opa(lv.OPA._30)
style_shadow.set_text_color(lv.color_black())

# Create a label for the shadow first (it's in the background)
shadow_label = lv.label(lv.scr_act())
shadow_label.add_style(style_shadow, 0)

# Create the main label
main_label = lv.label(lv.scr_act())
main_label.set_text("A simple method to create\n"
                    "shadows on a text.\n"
                    "It even works with\n\n"
                    "newlines     and spaces.")

# Set the same text for the shadow label
shadow_label.set_text(lv.label.get_text(main_label))

# Position the main label
main_label.align(lv.ALIGN.CENTER, 0, 0)

# Shift the second label down and to the right by 2 pixel
shadow_label.align_to(main_label, lv.ALIGN.TOP_LEFT, 2, 2)
```

Show LTR, RTL and Chinese texts

```
#include "../../lv_examples.h"
#if LV_USE_LABEL && LV_BUILD_EXAMPLES && LV_FONT_DEJAVU_16_PERSIAN_HEBREW && LV_FONT_
↪SIMSUN_16_CJK && LV_USE_BIDI

/**
 * Show mixed LTR, RTL and Chinese label
 */
void lv_example_label_3(void)
{
    lv_obj_t * ltr_label = lv_label_create(lv_scr_act());
    lv_label_set_text(ltr_label, "In modern terminology, a microcontroller is similar␣
↪to a system on a chip (SoC).");
    lv_obj_set_style_text_font(ltr_label, &lv_font_montserrat_16, 0);
    lv_obj_set_width(ltr_label, 310);
    lv_obj_align(ltr_label, LV_ALIGN_TOP_LEFT, 5, 5);

    lv_obj_t * rtl_label = lv_label_create(lv_scr_act());
    lv_label_set_text(rtl_label, "□□□□, □□ □□□□ □□□□ □□□□□ □□□□□ □□□□□□ :□□□□□□□) CPU␣
↪- Central Processing Unit).");
    lv_obj_set_style_base_dir(rtl_label, LV_BASE_DIR_RTL, 0);
    lv_obj_set_style_text_font(rtl_label, &lv_font_dejavu_16_persian_hebrew, 0);
    lv_obj_set_width(rtl_label, 310);
    lv_obj_align(rtl_label, LV_ALIGN_LEFT_MID, 5, 0);

    lv_obj_t * cz_label = lv_label_create(lv_scr_act());
    lv_label_set_text(cz_label, "嵌入式系统 (Embedded System), \n是一种嵌入机械或电气系统内部、
具有专一功能和实时计算性能的计算机系统。");
    lv_obj_set_style_text_font(cz_label, &lv_font_simsun_16_cjk, 0);
    lv_obj_set_width(cz_label, 310);
    lv_obj_align(cz_label, LV_ALIGN_BOTTOM_LEFT, 5, -5);
}

#endif
```

```
import fs_driver
#
# Show mixed LTR, RTL and Chinese label
#

ltr_label = lv.label(lv.scr_act())
ltr_label.set_text("In modern terminology, a microcontroller is similar to a system␣
↪on a chip (SoC).")
```

(下页继续)

(续上页)

```
# ltr_label.set_style_text_font(ltr_label, &lv_font_montserrat_16, 0);

fs_drv = lv.fs_drv_t()
fs_driver.fs_register(fs_drv, 'S')

try:
    ltr_label.set_style_text_font(ltr_label, lv.font_montserrat_16, 0)
except:
    font_montserrat_16 = lv.font_load("S:../../assets/font/montserrat-16.fnt")
    ltr_label.set_style_text_font(font_montserrat_16, 0)

ltr_label.set_width(310)
ltr_label.align(lv.ALIGN.TOP_LEFT, 5, 5)

rtl_label = lv.label(lv.scr_act())
rtl_label.set_text("□□□□, □□ □□□□ □□□□ □□□□□ □□□□□ □□□□□□ :□□□□□□□) CPU - Central␣
↪Processing Unit).")
rtl_label.set_style_base_dir(lv.BASE_DIR.RTL, 0)
rtl_label.set_style_text_font(lv.font_dejavu_16_persian_hebrew, 0)
rtl_label.set_width(310)
rtl_label.align(lv.ALIGN.LEFT_MID, 5, 0)

font_simsun_16_cjk = lv.font_load("S:../../assets/font/lv_font_simsun_16_cjk.fnt")

cz_label = lv.label(lv.scr_act())
cz_label.set_style_text_font(font_simsun_16_cjk, 0)
cz_label.set_text("嵌入式系统 (Embedded System), \n是一种嵌入机械或电气系统内部、具有专一功能和实
时计算性能的计算机系统。")
cz_label.set_width(310)
cz_label.align(lv.ALIGN.BOTTOM_LEFT, 5, -5)
```

Draw label with gradient color

```
#include "../../lv_examples.h"
#if LV_USE_LABEL && LV_USE_CANVAS && LV_BUILD_EXAMPLES && LV_DRAW_COMPLEX

#define MASK_WIDTH 100
#define MASK_HEIGHT 45

static void add_mask_event_cb(lv_event_t * e)
{
```

(下页继续)

(续上页)

```
    static lv_draw_mask_map_param_t m;
    static int16_t mask_id;

    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target(e);
    lv_opa_t * mask_map = lv_event_get_user_data(e);
    if(code == LV_EVENT_COVER_CHECK) {
        lv_event_set_cover_res(e, LV_COVER_RES_MASKED);
    }
    else if(code == LV_EVENT_DRAW_MAIN_BEGIN) {
        lv_draw_mask_map_init(&m, &obj->coords, mask_map);
        mask_id = lv_draw_mask_add(&m, NULL);

    }
    else if(code == LV_EVENT_DRAW_MAIN_END) {
        lv_draw_mask_free_param(&m);
        lv_draw_mask_remove_id(mask_id);
    }
}

/**
 * Draw label with gradient color
 */
void lv_example_label_4(void)
{
    /* Create the mask of a text by drawing it to a canvas*/
    static lv_opa_t mask_map[MASK_WIDTH * MASK_HEIGHT];

    /*Create a "8 bit alpha" canvas and clear it*/
    lv_obj_t * canvas = lv_canvas_create(lv_scr_act());
    lv_canvas_set_buffer(canvas, mask_map, MASK_WIDTH, MASK_HEIGHT, LV_IMG_CF_ALPHA_
↪8BIT);
    lv_canvas_fill_bg(canvas, lv_color_black(), LV_OPA_TRANSP);

    /*Draw a label to the canvas. The result "image" will be used as mask*/
    lv_draw_label_dsc_t label_dsc;
    lv_draw_label_dsc_init(&label_dsc);
    label_dsc.color = lv_color_white();
    label_dsc.align = LV_TEXT_ALIGN_CENTER;
    lv_canvas_draw_text(canvas, 5, 5, MASK_WIDTH, &label_dsc, "Text with gradient");

    /*The mask is reads the canvas is not required anymore*/
    lv_obj_del(canvas);
```

(下页继续)

(续上页)

```
    /* Create an object from where the text will be masked out.
     * Now it's a rectangle with a gradient but it could be an image too*/
    lv_obj_t * grad = lv_obj_create(lv_scr_act());
    lv_obj_set_size(grad, MASK_WIDTH, MASK_HEIGHT);
    lv_obj_center(grad);
    lv_obj_set_style_bg_color(grad, lv_color_hex(0xff0000), 0);
    lv_obj_set_style_bg_grad_color(grad, lv_color_hex(0x0000ff), 0);
    lv_obj_set_style_bg_grad_dir(grad, LV_GRAD_DIR_HOR, 0);
    lv_obj_add_event_cb(grad, add_mask_event_cb, LV_EVENT_ALL, mask_map);
}

#endif
```

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs/100ask_lvgl_
↪docs/examples/widgets/label/lv_example_label_4.py
```

LED

LED with custom style

```
#include "../../lv_examples.h"
#if LV_USE_LED && LV_BUILD_EXAMPLES

/**
 * Create LED's with different brightness and color
 */
void lv_example_led_1(void)
{
    /*Create a LED and switch it OFF*/
    lv_obj_t * led1  = lv_led_create(lv_scr_act());
    lv_obj_align(led1, LV_ALIGN_CENTER, -80, 0);
    lv_led_off(led1);

    /*Copy the previous LED and set a brightness*/
    lv_obj_t * led2  = lv_led_create(lv_scr_act());
    lv_obj_align(led2, LV_ALIGN_CENTER, 0, 0);
    lv_led_set_brightness(led2, 150);
    lv_led_set_color(led2, lv_palette_main(LV_PALETTE_RED));

    /*Copy the previous LED and switch it ON*/
    lv_obj_t * led3  = lv_led_create(lv_scr_act());
```

(下页继续)

(续上页)

```
    lv_obj_align(led3, LV_ALIGN_CENTER, 80, 0);
    lv_led_on(led3);
}

#endif
```

```
#
# Create LED's with different brightness and color
#

# Create a LED and switch it OFF
led1  = lv.led(lv.scr_act())
led1.align(lv.ALIGN.CENTER, -80, 0)
led1.off()

# Copy the previous LED and set a brightness
led2  = lv.led(lv.scr_act())
led2.align(lv.ALIGN.CENTER, 0, 0)
led2.set_brightness(150)
led2.set_color(lv.palette_main(lv.PALETTE.RED))

# Copy the previous LED and switch it ON
led3  = lv.led(lv.scr_act())
led3.align(lv.ALIGN.CENTER, 80, 0)
led3.on()
```

Line

Simple Line

```
#include "../../lv_examples.h"
#if LV_USE_LINE && LV_BUILD_EXAMPLES

void lv_example_line_1(void)
{
    /*Create an array for the points of the line*/
    static lv_point_t line_points[] = { {5, 5}, {70, 70}, {120, 10}, {180, 60}, {240,␣
↪10} };

    /*Create style*/
    static lv_style_t style_line;
```

(下页继续)

(续上页)

```
    lv_style_init(&style_line);
    lv_style_set_line_width(&style_line, 8);
    lv_style_set_line_color(&style_line, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_line_rounded(&style_line, true);

    /*Create a line and apply the new style*/
    lv_obj_t * line1;
    line1 = lv_line_create(lv_scr_act());
    lv_line_set_points(line1, line_points, 5);     /*Set the points*/
    lv_obj_add_style(line1, &style_line, 0);
    lv_obj_center(line1);
}

#endif
```

```
# Create an array for the points of the line
line_points = [ {"x":5, "y":5},
                {"x":70, "y":70},
                {"x":120, "y":10},
                {"x":180, "y":60},
                {"x":240, "y":10}]

# Create style
style_line = lv.style_t()
style_line.init()
style_line.set_line_width(8)
style_line.set_line_color(lv.palette_main(lv.PALETTE.BLUE))
style_line.set_line_rounded(True)

# Create a line and apply the new style
line1 = lv.line(lv.scr_act())
line1.set_points(line_points, 5)     # Set the points
line1.add_style(style_line, 0)
line1.center()
```

List

Simple List

```c
#include "../../lv_examples.h"
#if LV_USE_LIST && LV_BUILD_EXAMPLES
static lv_obj_t * list1;

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target(e);
    if(code == LV_EVENT_CLICKED) {
        LV_LOG_USER("Clicked: %s", lv_list_get_btn_text(list1, obj));
    }
}

void lv_example_list_1(void)
{
    /*Create a list*/
    list1 = lv_list_create(lv_scr_act());
    lv_obj_set_size(list1, 180, 220);
    lv_obj_center(list1);

    /*Add buttons to the list*/
    lv_obj_t * btn;

    lv_list_add_text(list1, "File");
    btn = lv_list_add_btn(list1, LV_SYMBOL_FILE, "New");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_btn(list1, LV_SYMBOL_DIRECTORY, "Open");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_btn(list1, LV_SYMBOL_SAVE, "Save");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_btn(list1, LV_SYMBOL_CLOSE, "Delete");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_btn(list1, LV_SYMBOL_EDIT, "Edit");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);

    lv_list_add_text(list1, "Connectivity");
    btn = lv_list_add_btn(list1, LV_SYMBOL_BLUETOOTH, "Bluetooth");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_btn(list1, LV_SYMBOL_GPS, "Navigation");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
```

(下页继续)

(续上页)

```
    btn = lv_list_add_btn(list1, LV_SYMBOL_USB, "USB");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_btn(list1, LV_SYMBOL_BATTERY_FULL, "Battery");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);

    lv_list_add_text(list1, "Exit");
    btn = lv_list_add_btn(list1, LV_SYMBOL_OK, "Apply");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_btn(list1, LV_SYMBOL_CLOSE, "Close");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
}

#endif
```

```
def event_handler(e):
    code = e.get_code()
    obj = e.get_target()
    if code == lv.EVENT.CLICKED:
            print("Clicked: list1." + list1.get_btn_text(obj))

# Create a list
list1 = lv.list(lv.scr_act())
list1.set_size(180, 220)
list1.center()

# Add buttons to the list
list1.add_text("File")
btn_new = list1.add_btn(lv.SYMBOL.FILE, "New")
btn_new.add_event_cb(event_handler,lv.EVENT.ALL, None)
btn_open = list1.add_btn(lv.SYMBOL.DIRECTORY, "Open")
btn_open.add_event_cb(event_handler,lv.EVENT.ALL, None)
btn_save = list1.add_btn(lv.SYMBOL.SAVE, "Save")
btn_save.add_event_cb(event_handler,lv.EVENT.ALL, None)
btn_delete = list1.add_btn(lv.SYMBOL.CLOSE, "Delete")
btn_delete.add_event_cb(event_handler,lv.EVENT.ALL, None)
btn_edit = list1.add_btn(lv.SYMBOL.EDIT, "Edit")
btn_edit.add_event_cb(event_handler,lv.EVENT.ALL, None)

list1.add_text("Connectivity")
btn_bluetooth = list1.add_btn(lv.SYMBOL.BLUETOOTH, "Bluetooth")
btn_bluetooth.add_event_cb(event_handler,lv.EVENT.ALL, None)
btn_navig = list1.add_btn(lv.SYMBOL.GPS, "Navigation")
btn_navig.add_event_cb(event_handler,lv.EVENT.ALL, None)
```

(下页继续)

(续上页)

```
btn_USB = list1.add_btn(lv.SYMBOL.USB, "USB")
btn_USB.add_event_cb(event_handler,lv.EVENT.ALL, None)
btn_battery = list1.add_btn(lv.SYMBOL.BATTERY_FULL, "Battery")
btn_battery.add_event_cb(event_handler,lv.EVENT.ALL, None)

list1.add_text("Exit")
btn_apply = list1.add_btn(lv.SYMBOL.OK, "Apply")
btn_apply.add_event_cb(event_handler,lv.EVENT.ALL, None)
btn_close = list1.add_btn(lv.SYMBOL.CLOSE, "Close")
btn_close.add_event_cb(event_handler,lv.EVENT.ALL, None)
```

Sorting a List using up and down buttons

```
#include <stdlib.h>


#include "../../lv_examples.h"
#if LV_USE_LIST && LV_BUILD_EXAMPLES

static lv_obj_t* list1;
static lv_obj_t* list2;

static lv_obj_t* currentButton = NULL;

static void event_handler(lv_event_t* e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t* obj = lv_event_get_target(e);
    if (code == LV_EVENT_CLICKED)
    {
        LV_LOG_USER("Clicked: %s", lv_list_get_btn_text(list1, obj));

        if (currentButton == obj)
        {
            currentButton = NULL;
        }
        else
        {
            currentButton = obj;
        }
        lv_obj_t* parent = lv_obj_get_parent(obj);
```

(下页继续)

(续上页)

```
        uint32_t i;
        for (i = 0; i < lv_obj_get_child_cnt(parent); i++)
        {
            lv_obj_t* child = lv_obj_get_child(parent, i);
            if (child == currentButton)
            {
                lv_obj_add_state(child, LV_STATE_CHECKED);
            }
            else
            {
                lv_obj_clear_state(child, LV_STATE_CHECKED);
            }
        }
    }
}

static void event_handler_top(lv_event_t* e)
{
    lv_event_code_t code = lv_event_get_code(e);
    if (code == LV_EVENT_CLICKED)
    {
        if (currentButton == NULL) return;
        lv_obj_move_background(currentButton);
        lv_obj_scroll_to_view(currentButton, LV_ANIM_ON);
    }
}

static void event_handler_up(lv_event_t* e)
{
    lv_event_code_t code = lv_event_get_code(e);
    if ((code == LV_EVENT_CLICKED) || (code == LV_EVENT_LONG_PRESSED_REPEAT))
    {
        if (currentButton == NULL) return;
        uint32_t index = lv_obj_get_index(currentButton);
        if (index <= 0) return;
        lv_obj_move_to_index(currentButton, index - 1);
        lv_obj_scroll_to_view(currentButton, LV_ANIM_ON);
    }
}

static void event_handler_center(lv_event_t* e)
{
    const lv_event_code_t code = lv_event_get_code(e);
```

(下页继续)

(续上页)

```
    if ((code == LV_EVENT_CLICKED) || (code == LV_EVENT_LONG_PRESSED_REPEAT))
    {
        if (currentButton == NULL) return;

        lv_obj_t* parent = lv_obj_get_parent(currentButton);
        const uint32_t pos = lv_obj_get_child_cnt(parent) / 2;

        lv_obj_move_to_index(currentButton, pos);

        lv_obj_scroll_to_view(currentButton, LV_ANIM_ON);
    }
}

static void event_handler_dn(lv_event_t* e)
{
    const lv_event_code_t code = lv_event_get_code(e);
    if ((code == LV_EVENT_CLICKED) || (code == LV_EVENT_LONG_PRESSED_REPEAT))
    {
        if (currentButton == NULL) return;
        const uint32_t index = lv_obj_get_index(currentButton);

        lv_obj_move_to_index(currentButton, index + 1);
        lv_obj_scroll_to_view(currentButton, LV_ANIM_ON);
    }
}

static void event_handler_bottom(lv_event_t* e)
{
    const lv_event_code_t code = lv_event_get_code(e);
    if (code == LV_EVENT_CLICKED)
    {
        if (currentButton == NULL) return;
        lv_obj_move_foreground(currentButton);
        lv_obj_scroll_to_view(currentButton, LV_ANIM_ON);
    }
}

static void event_handler_swap(lv_event_t* e)
{
    const lv_event_code_t code = lv_event_get_code(e);
    // lv_obj_t* obj = lv_event_get_target(e);
    if ((code == LV_EVENT_CLICKED) || (code == LV_EVENT_LONG_PRESSED_REPEAT))
    {
```

(下页继续)

(续上页)

```
        uint32_t cnt = lv_obj_get_child_cnt(list1);
        for (int i = 0; i < 100; i++)
            if (cnt > 1)
            {
                lv_obj_t* obj = lv_obj_get_child(list1, rand() % cnt);
                lv_obj_move_to_index(obj, rand() % cnt);
                if (currentButton != NULL)
                {
                    lv_obj_scroll_to_view(currentButton, LV_ANIM_ON);
                }
            }
    }
}

void lv_example_list_2(void)
{
    /*Create a list*/
    list1 = lv_list_create(lv_scr_act());
    lv_obj_set_size(list1, lv_pct(60), lv_pct(100));
    lv_obj_set_style_pad_row(list1, 5, 0);

    /*Add buttons to the list*/
    lv_obj_t* btn;
    int i;
    for (i = 0; i < 15; i++) {
        btn = lv_btn_create(list1);
        lv_obj_set_width(btn, lv_pct(50));
        lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);

        lv_obj_t* lab = lv_label_create(btn);
        lv_label_set_text_fmt(lab, "Item %d", i);
    }

    /*Select the first button by default*/
    currentButton = lv_obj_get_child(list1, 0);
    lv_obj_add_state(currentButton, LV_STATE_CHECKED);

    /*Create a second list with up and down buttons*/
    list2 = lv_list_create(lv_scr_act());
    lv_obj_set_size(list2, lv_pct(40), lv_pct(100));
    lv_obj_align(list2, LV_ALIGN_TOP_RIGHT, 0, 0);
    lv_obj_set_flex_flow(list2, LV_FLEX_FLOW_COLUMN);
```

(下页继续)

(续上页)

```
    btn = lv_list_add_btn(list2, NULL, "Top");
    lv_obj_add_event_cb(btn, event_handler_top, LV_EVENT_ALL, NULL);
    lv_group_remove_obj(btn);

    btn = lv_list_add_btn(list2, LV_SYMBOL_UP, "Up");
    lv_obj_add_event_cb(btn, event_handler_up, LV_EVENT_ALL, NULL);
    lv_group_remove_obj(btn);

    btn = lv_list_add_btn(list2, LV_SYMBOL_LEFT, "Center");
    lv_obj_add_event_cb(btn, event_handler_center, LV_EVENT_ALL, NULL);
    lv_group_remove_obj(btn);

    btn = lv_list_add_btn(list2, LV_SYMBOL_DOWN, "Down");
    lv_obj_add_event_cb(btn, event_handler_dn, LV_EVENT_ALL, NULL);
    lv_group_remove_obj(btn);

    btn = lv_list_add_btn(list2, NULL, "Bottom");
    lv_obj_add_event_cb(btn, event_handler_bottom, LV_EVENT_ALL, NULL);
    lv_group_remove_obj(btn);

    btn = lv_list_add_btn(list2, LV_SYMBOL_SHUFFLE, "Shuffle");
    lv_obj_add_event_cb(btn, event_handler_swap, LV_EVENT_ALL, NULL);
    lv_group_remove_obj(btn);
}

#endif
```

```
import urandom

currentButton = None
list1 = None

def event_handler(evt):
    global currentButton
    code = evt.get_code()
    obj = evt.get_target()
    if code == lv.EVENT.CLICKED:
        if currentButton == obj:
            currentButton = None
        else:
            currentButton = obj
        parent = obj.get_parent()
        for i in range( parent.get_child_cnt()):
```

(下页继续)

(续上页)

```
            child = parent.get_child(i)
            if child == currentButton:
                child.add_state(lv.STATE.CHECKED)
            else:
                child.clear_state(lv.STATE.CHECKED)

def event_handler_top(evt):
    global currentButton
    code = evt.get_code()
    obj = evt.get_target()
    if code == lv.EVENT.CLICKED:
        if currentButton == None:
            return
        currentButton.move_background()
        currentButton.scroll_to_view( lv.ANIM.ON)

def event_handler_up(evt):
    global currentButton
    code = evt.get_code()
    obj = evt.get_target()
    if code == lv.EVENT.CLICKED or code == lv.EVENT.LONG_PRESSED_REPEAT:
        if currentButton == None:
            return
        index = currentButton.get_index()
        if index <= 0:
            return
        currentButton.move_to_index(index - 1)
        currentButton.scroll_to_view(lv.ANIM.ON)

def event_handler_center(evt):
    global currentButton
    code = evt.get_code()
    obj = evt.get_target()
    if code == lv.EVENT.CLICKED or code == lv.EVENT.LONG_PRESSED_REPEAT:
        if currentButton == None:
            return
        parent = currentButton.get_parent()
        pos = parent.get_child_cnt() // 2
        currentButton.move_to_index(pos)
        currentButton.scroll_to_view(lv.ANIM.ON)

def event_handler_dn(evt):
    global currentButton
```

(下页继续)

(续上页)

```
    code = evt.get_code()
    obj = evt.get_target()
    if code == lv.EVENT.CLICKED or code == lv.EVENT.LONG_PRESSED_REPEAT:
        if currentButton == None:
            return
        index = currentButton.get_index()
        currentButton.move_to_index(index + 1)
        currentButton.scroll_to_view(lv.ANIM.ON)

def event_handler_bottom(evt):
    global currentButton
    code = evt.get_code()
    obj = evt.get_target()
    if code == lv.EVENT.CLICKED or code == lv.EVENT.LONG_PRESSED_REPEAT:
        if currentButton == None:
            return
        currentButton.move_foreground()
        currentButton.scroll_to_view(lv.ANIM.ON)

def event_handler_swap(evt):
    global currentButton
    global list1
    code = evt.get_code()
    obj = evt.get_target()
    if code == lv.EVENT.CLICKED:
        cnt = list1.get_child_cnt()
        for i in range(100):
            if cnt > 1:
                obj = list1.get_child(urandom.getrandbits(32) % cnt )
                obj.move_to_index(urandom.getrandbits(32) % cnt)
        if currentButton != None:
            currentButton.scroll_to_view(lv.ANIM.ON)

#Create a list with buttons that can be sorted
list1 = lv.list(lv.scr_act())
list1.set_size(lv.pct(60), lv.pct(100))
list1.set_style_pad_row( 5, 0)

for i in range(15):
    btn = lv.btn(list1)
    btn.set_width(lv.pct(100))
    btn.add_event_cb( event_handler, lv.EVENT.CLICKED, None)
    lab = lv.label(btn)
```

(下页继续)

(续上页)

```
    lab.set_text("Item " + str(i))

#Select the first button by default
currentButton = list1.get_child(0)
currentButton.add_state(lv.STATE.CHECKED)

#Create a second list with up and down buttons
list2 = lv.list(lv.scr_act())
list2.set_size(lv.pct(40), lv.pct(100))
list2.align(lv.ALIGN.TOP_RIGHT, 0, 0)
list2.set_flex_flow(lv.FLEX_FLOW.COLUMN)

btn = list2.add_btn(None, "Top")
btn.add_event_cb(event_handler_top, lv.EVENT.ALL, None)
lv.group_remove_obj(btn)

btn = list2.add_btn(lv.SYMBOL.UP, "Up")
btn.add_event_cb(event_handler_up, lv.EVENT.ALL, None)
lv.group_remove_obj(btn)

btn = list2.add_btn(lv.SYMBOL.LEFT, "Center")
btn.add_event_cb(event_handler_center, lv.EVENT.ALL, None)
lv.group_remove_obj(btn)

btn = list2.add_btn(lv.SYMBOL.DOWN, "Down")
btn.add_event_cb(event_handler_dn, lv.EVENT.ALL, None)
lv.group_remove_obj(btn)

btn = list2.add_btn(None, "Bottom")
btn.add_event_cb(event_handler_bottom, lv.EVENT.ALL, None)
lv.group_remove_obj(btn)

btn = list2.add_btn(lv.SYMBOL.SHUFFLE, "Shuffle")
btn.add_event_cb(event_handler_swap, lv.EVENT.ALL, None)
lv.group_remove_obj(btn)
```

Meter

Simple meter

```
#include "../../lv_examples.h"
#if LV_USE_METER && LV_BUILD_EXAMPLES

static lv_obj_t * meter;

static void set_value(void * indic, int32_t v)
{
    lv_meter_set_indicator_value(meter, indic, v);
}

/**
 * A simple meter
 */
void lv_example_meter_1(void)
{
    meter = lv_meter_create(lv_scr_act());
    lv_obj_center(meter);
    lv_obj_set_size(meter, 200, 200);

    /*Add a scale first*/
    lv_meter_scale_t * scale = lv_meter_add_scale(meter);
    lv_meter_set_scale_ticks(meter, scale, 41, 2, 10, lv_palette_main(LV_PALETTE_
↪GREY));
    lv_meter_set_scale_major_ticks(meter, scale, 8, 4, 15, lv_color_black(), 10);

    lv_meter_indicator_t * indic;

    /*Add a blue arc to the start*/
    indic = lv_meter_add_arc(meter, scale, 3, lv_palette_main(LV_PALETTE_BLUE), 0);
    lv_meter_set_indicator_start_value(meter, indic, 0);
    lv_meter_set_indicator_end_value(meter, indic, 20);

    /*Make the tick lines blue at the start of the scale*/
    indic = lv_meter_add_scale_lines(meter, scale, lv_palette_main(LV_PALETTE_BLUE),␣
↪lv_palette_main(LV_PALETTE_BLUE), false, 0);
    lv_meter_set_indicator_start_value(meter, indic, 0);
    lv_meter_set_indicator_end_value(meter, indic, 20);

    /*Add a red arc to the end*/
    indic = lv_meter_add_arc(meter, scale, 3, lv_palette_main(LV_PALETTE_RED), 0);
```

(下页继续)

(续上页)

```
    lv_meter_set_indicator_start_value(meter, indic, 80);
    lv_meter_set_indicator_end_value(meter, indic, 100);

    /*Make the tick lines red at the end of the scale*/
    indic = lv_meter_add_scale_lines(meter, scale, lv_palette_main(LV_PALETTE_RED),␣
↪lv_palette_main(LV_PALETTE_RED), false, 0);
    lv_meter_set_indicator_start_value(meter, indic, 80);
    lv_meter_set_indicator_end_value(meter, indic, 100);

    /*Add a needle line indicator*/
    indic = lv_meter_add_needle_line(meter, scale, 4, lv_palette_main(LV_PALETTE_
↪GREY), -10);

    /*Create an animation to set the value*/
    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_exec_cb(&a, set_value);
    lv_anim_set_var(&a, indic);
    lv_anim_set_values(&a, 0, 100);
    lv_anim_set_time(&a, 2000);
    lv_anim_set_repeat_delay(&a, 100);
    lv_anim_set_playback_time(&a, 500);
    lv_anim_set_playback_delay(&a, 100);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
    lv_anim_start(&a);
}

#endif
```

```
#!/opt/bin/lv_micropython -i
import utime as time
import lvgl as lv
import display_driver

def set_value(indic, v):
    meter.set_indicator_value(indic, v)

#
# A simple meter
#
meter = lv.meter(lv.scr_act())
meter.center()
meter.set_size(200, 200)
```

(下页继续)

(续上页)

```
# Add a scale first
scale = meter.add_scale()
meter.set_scale_ticks(scale, 51, 2, 10, lv.palette_main(lv.PALETTE.GREY))
meter.set_scale_major_ticks(scale, 10, 4, 15, lv.color_black(), 10)

indic = lv.meter_indicator_t()

# Add a blue arc to the start
indic = meter.add_arc(scale, 3, lv.palette_main(lv.PALETTE.BLUE), 0)
meter.set_indicator_start_value(indic, 0)
meter.set_indicator_end_value(indic, 20)

# Make the tick lines blue at the start of the scale
indic = meter.add_scale_lines(scale, lv.palette_main(lv.PALETTE.BLUE), lv.palette_
↪main(lv.PALETTE.BLUE), False, 0)
meter.set_indicator_start_value(indic, 0)
meter.set_indicator_end_value(indic, 20)

# Add a red arc to the end
indic = meter.add_arc(scale, 3, lv.palette_main(lv.PALETTE.RED), 0)
meter.set_indicator_start_value(indic, 80)
meter.set_indicator_end_value(indic, 100)

# Make the tick lines red at the end of the scale
indic = meter.add_scale_lines(scale, lv.palette_main(lv.PALETTE.RED), lv.palette_
↪main(lv.PALETTE.RED), False, 0)
meter.set_indicator_start_value(indic, 80)
meter.set_indicator_end_value(indic, 100)

# Add a needle line indicator
indic = meter.add_needle_line(scale, 4, lv.palette_main(lv.PALETTE.GREY), -10)

# Create an animation to set the value
a = lv.anim_t()
a.init()
a.set_var(indic)
a.set_values(0, 100)
a.set_time(2000)
a.set_repeat_delay(100)
a.set_playback_time(500)
a.set_playback_delay(100)
a.set_repeat_count(lv.ANIM_REPEAT.INFINITE)
```

(下页继续)

(续上页)

```
a.set_custom_exec_cb(lambda a,val: set_value(indic,val))
lv.anim_t.start(a)
```

A meter with multiple arcs

```
#include "../../lv_examples.h"
#if LV_USE_METER && LV_BUILD_EXAMPLES

static lv_obj_t * meter;

static void set_value(void * indic, int32_t v)
{
    lv_meter_set_indicator_end_value(meter, indic, v);
}


/**
 * A meter with multiple arcs
 */
void lv_example_meter_2(void)
{
    meter = lv_meter_create(lv_scr_act());
    lv_obj_center(meter);
    lv_obj_set_size(meter, 200, 200);

    /*Remove the circle from the middle*/
    lv_obj_remove_style(meter, NULL, LV_PART_INDICATOR);

    /*Add a scale first*/
    lv_meter_scale_t * scale = lv_meter_add_scale(meter);
    lv_meter_set_scale_ticks(meter, scale, 11, 2, 10, lv_palette_main(LV_PALETTE_
↪GREY));
    lv_meter_set_scale_major_ticks(meter, scale, 1, 2, 30, lv_color_hex3(0xeee), 15);
    lv_meter_set_scale_range(meter, scale, 0, 100, 270, 90);

    /*Add a three arc indicator*/
    lv_meter_indicator_t * indic1 = lv_meter_add_arc(meter, scale, 10, lv_palette_
↪main(LV_PALETTE_RED), 0);
    lv_meter_indicator_t * indic2 = lv_meter_add_arc(meter, scale, 10, lv_palette_
↪main(LV_PALETTE_GREEN), -10);
    lv_meter_indicator_t * indic3 = lv_meter_add_arc(meter, scale, 10, lv_palette_
↪main(LV_PALETTE_BLUE), -20);
```

(下页继续)

(续上页)

```
    /*Create an animation to set the value*/
    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_exec_cb(&a, set_value);
    lv_anim_set_values(&a, 0, 100);
    lv_anim_set_repeat_delay(&a, 100);
    lv_anim_set_playback_delay(&a, 100);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);

    lv_anim_set_time(&a, 2000);
    lv_anim_set_playback_time(&a, 500);
    lv_anim_set_var(&a, indic1);
    lv_anim_start(&a);

    lv_anim_set_time(&a, 1000);
    lv_anim_set_playback_time(&a, 1000);
    lv_anim_set_var(&a, indic2);
    lv_anim_start(&a);

    lv_anim_set_time(&a, 1000);
    lv_anim_set_playback_time(&a, 2000);
    lv_anim_set_var(&a, indic3);
    lv_anim_start(&a);
}

#endif
```

```
#!//opt/bin/lv_micropython -i
import utime as time
import lvgl as lv
import display_driver

def set_value(indic,v):
    meter.set_indicator_end_value(indic, v)

#
# A meter with multiple arcs
#

meter = lv.meter(lv.scr_act())
meter.center()
meter.set_size(200, 200)
```

(下页继续)

(续上页)

```
# Remove the circle from the middle
meter.remove_style(None, lv.PART.INDICATOR)

# Add a scale first
scale = meter.add_scale()
meter.set_scale_ticks(scale, 11, 2, 10, lv.palette_main(lv.PALETTE.GREY))
meter.set_scale_major_ticks(scale, 1, 2, 30, lv.color_hex3(0xeee), 10)
meter.set_scale_range(scale, 0, 100, 270, 90)

# Add a three arc indicator
indic1 = meter.add_arc(scale, 10, lv.palette_main(lv.PALETTE.RED), 0)
indic2 = meter.add_arc(scale, 10, lv.palette_main(lv.PALETTE.GREEN), -10)
indic3 = meter.add_arc(scale, 10, lv.palette_main(lv.PALETTE.BLUE), -20)

# Create an animation to set the value
a1 = lv.anim_t()
a1.init()
a1.set_values(0, 100)
a1.set_time(2000)
a1.set_repeat_delay(100)
a1.set_playback_delay(100)
a1.set_playback_time(500)
a1.set_var(indic1)
a1.set_repeat_count(lv.ANIM_REPEAT.INFINITE)
a1.set_custom_exec_cb(lambda a,val: set_value(indic1,val))
lv.anim_t.start(a1)

a2 = lv.anim_t()
a2.init()
a2.set_values(0, 100)
a2.set_time(1000)
a2.set_repeat_delay(100)
a2.set_playback_delay(100)
a2.set_playback_time(1000)
a2.set_var(indic2)
a2.set_repeat_count(lv.ANIM_REPEAT.INFINITE)
a2.set_custom_exec_cb(lambda a,val: set_value(indic2,val))
lv.anim_t.start(a2)

a3 = lv.anim_t()
a3.init()
a3.set_values(0, 100)
```

(下页继续)

(续上页)

```
a3.set_time(1000)
a3.set_repeat_delay(100)
a3.set_playback_delay(100)
a3.set_playback_time(2000)
a3.set_var(indic3)
a3.set_repeat_count(lv.ANIM_REPEAT.INFINITE)
a3.set_custom_exec_cb(lambda a,val: set_value(indic3,val))
lv.anim_t.start(a3)
```

A clock from a meter

```
#include "../../lv_examples.h"
#if LV_USE_METER && LV_BUILD_EXAMPLES

static lv_obj_t * meter;

static void set_value(void * indic, int32_t v)
{
    lv_meter_set_indicator_end_value(meter, indic, v);
}

/**
 * A clock from a meter
 */
void lv_example_meter_3(void)
{
    meter = lv_meter_create(lv_scr_act());
    lv_obj_set_size(meter, 220, 220);
    lv_obj_center(meter);

    /*Create a scale for the minutes*/
    /*61 ticks in a 360 degrees range (the last and the first line overlaps)*/
    lv_meter_scale_t * scale_min = lv_meter_add_scale(meter);
    lv_meter_set_scale_ticks(meter, scale_min, 61, 1, 10, lv_palette_main(LV_PALETTE_
↪GREY));
    lv_meter_set_scale_range(meter, scale_min, 0, 60, 360, 270);

    /*Create another scale for the hours. It's only visual and contains only major␣
↪ticks*/
```

(下页继续)

(续上页)

```
    lv_meter_scale_t * scale_hour = lv_meter_add_scale(meter);
    lv_meter_set_scale_ticks(meter, scale_hour, 12, 0, 0, lv_palette_main(LV_PALETTE_
↪GREY));               /*12 ticks*/
    lv_meter_set_scale_major_ticks(meter, scale_hour, 1, 2, 20, lv_color_black(), 10);
↪    /*Every tick is major*/
    lv_meter_set_scale_range(meter, scale_hour, 1, 12, 330, 300);       /*[1..12]␣
↪values in an almost full circle*/

    LV_IMG_DECLARE(img_hand)

    /*Add a the hands from images*/
    lv_meter_indicator_t * indic_min = lv_meter_add_needle_img(meter, scale_min, &img_
↪hand, 5, 5);
    lv_meter_indicator_t * indic_hour = lv_meter_add_needle_img(meter, scale_min, &
↪img_hand, 5, 5);

    /*Create an animation to set the value*/
    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_exec_cb(&a, set_value);
    lv_anim_set_values(&a, 0, 60);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
    lv_anim_set_time(&a, 2000);     /*2 sec for 1 turn of the minute hand (1 hour)*/
    lv_anim_set_var(&a, indic_min);
    lv_anim_start(&a);

    lv_anim_set_var(&a, indic_hour);
    lv_anim_set_time(&a, 24000);    /*24 sec for 1 turn of the hour hand*/
    lv_anim_set_values(&a, 0, 60);
    lv_anim_start(&a);
}

#endif
```

```
#!//opt/bin/lv_micropython -i
import utime as time
import lvgl as lv
import display_driver
from imagetools import get_png_info, open_png

# Register PNG image decoder
decoder = lv.img.decoder_create()
decoder.info_cb = get_png_info
```

(下页继续)

(续上页)

```
decoder.open_cb = open_png

# Create an image from the png file
try:
    with open('../../assets/img_hand_min.png','rb') as f:
        img_hand_min_data = f.read()
except:
    print("Could not find img_hand_min.png")
    sys.exit()

img_hand_min_dsc = lv.img_dsc_t({
  'data_size': len(img_hand_min_data),
  'data': img_hand_min_data
})

# Create an image from the png file
try:
    with open('../../assets/img_hand_hour.png','rb') as f:
        img_hand_hour_data = f.read()
except:
    print("Could not find img_hand_hour.png")
    sys.exit()

img_hand_hour_dsc = lv.img_dsc_t({
  'data_size': len(img_hand_hour_data),
  'data': img_hand_hour_data
})

def set_value(indic, v):
    meter.set_indicator_value(indic, v)
#
# A clock from a meter
#

meter = lv.meter(lv.scr_act())
meter.set_size(220, 220)
meter.center()

# Create a scale for the minutes
# 61 ticks in a 360 degrees range (the last and the first line overlaps)
scale_min = meter.add_scale()
meter.set_scale_ticks(scale_min, 61, 1, 10, lv.palette_main(lv.PALETTE.GREY))
meter.set_scale_range(scale_min, 0, 60, 360, 270)
```

(下页继续)

(续上页)

```
# Create another scale for the hours. It's only visual and contains only major ticks
scale_hour = meter.add_scale()
meter.set_scale_ticks(scale_hour, 12, 0, 0, lv.palette_main(lv.PALETTE.GREY))  # 12␣
↪ticks
meter.set_scale_major_ticks(scale_hour, 1, 2, 20, lv.color_black(), 10)         #␣
↪Every tick is major
meter.set_scale_range(scale_hour, 1, 12, 330, 300)                              # [1..
↪12] values in an almost full circle

#    LV_IMG_DECLARE(img_hand)

# Add the hands from images
indic_min = meter.add_needle_img(scale_min, img_hand_min_dsc, 5, 5)
indic_hour = meter.add_needle_img(scale_min, img_hand_hour_dsc, 5, 5)

# Create an animation to set the value
a1 = lv.anim_t()
a1.init()
a1.set_values(0, 60)
a1.set_repeat_count(lv.ANIM_REPEAT.INFINITE)
a1.set_time(2000)        # 2 sec for 1 turn of the minute hand (1 hour)
a1.set_var(indic_min)
a1.set_custom_exec_cb(lambda a1,val: set_value(indic_min,val))
lv.anim_t.start(a1)

a2 = lv.anim_t()
a2.init()
a2.set_var(indic_hour)
a2.set_time(24000)       # 24 sec for 1 turn of the hour hand
a2.set_values(0, 60)
a2.set_custom_exec_cb(lambda a2,val: set_value(indic_hour,val))
lv.anim_t.start(a2)
```

Pie chart

```
#include "../../lv_examples.h"
#if LV_USE_METER && LV_BUILD_EXAMPLES

/**
 * Create a pie chart
```

(下页继续)

(续上页)

```
 */
void lv_example_meter_4(void)
{
    lv_obj_t * meter = lv_meter_create(lv_scr_act());

    /*Remove the background and the circle from the middle*/
    lv_obj_remove_style(meter, NULL, LV_PART_MAIN);
    lv_obj_remove_style(meter, NULL, LV_PART_INDICATOR);

    lv_obj_set_size(meter, 200, 200);
    lv_obj_center(meter);

    /*Add a scale first with no ticks.*/
    lv_meter_scale_t * scale = lv_meter_add_scale(meter);
    lv_meter_set_scale_ticks(meter, scale, 0, 0, 0, lv_color_black());
    lv_meter_set_scale_range(meter, scale, 0, 100, 360, 0);

    /*Add a three arc indicator*/
    lv_coord_t indic_w = 100;
    lv_meter_indicator_t * indic1 = lv_meter_add_arc(meter, scale, indic_w,lv_palette_
↪main(LV_PALETTE_ORANGE), 0);
    lv_meter_set_indicator_start_value(meter, indic1, 0);
    lv_meter_set_indicator_end_value(meter, indic1, 40);

    lv_meter_indicator_t * indic2 = lv_meter_add_arc(meter, scale, indic_w, lv_
↪palette_main(LV_PALETTE_YELLOW), 0);
    lv_meter_set_indicator_start_value(meter, indic2, 40);  /*Start from the␣
↪previous*/
    lv_meter_set_indicator_end_value(meter, indic2, 80);

    lv_meter_indicator_t * indic3 = lv_meter_add_arc(meter, scale, indic_w, lv_
↪palette_main(LV_PALETTE_DEEP_ORANGE), 0);
    lv_meter_set_indicator_start_value(meter, indic3, 80);  /*Start from the␣
↪previous*/
    lv_meter_set_indicator_end_value(meter, indic3, 100);
}

#endif
```

```
#
# Create a pie chart
#
```

(下页继续)

(续上页)

```
meter = lv.meter(lv.scr_act())

# Remove the background and the circle from the middle
meter.remove_style(None, lv.PART.MAIN)
meter.remove_style(None, lv.PART.INDICATOR)

meter.set_size(200, 200)
meter.center()

# Add a scale first with no ticks.
scale = meter.add_scale()
meter.set_scale_ticks(scale, 0, 0, 0, lv.color_black())
meter.set_scale_range(scale, 0, 100, 360, 0)

# Add a three arc indicator*
indic_w = 100
indic1 = meter.add_arc(scale, indic_w,lv.palette_main(lv.PALETTE.ORANGE), 0)
meter.set_indicator_start_value(indic1, 0)
meter.set_indicator_end_value(indic1, 40)

indic2 = meter.add_arc(scale, indic_w, lv.palette_main(lv.PALETTE.YELLOW), 0)
meter.set_indicator_start_value(indic2, 40)  # Start from the previous
meter.set_indicator_end_value(indic2, 80)

indic3 = meter.add_arc(scale, indic_w, lv.palette_main(lv.PALETTE.DEEP_ORANGE), 0)
meter.set_indicator_start_value(indic3, 80)  # Start from the previous
meter.set_indicator_end_value(indic3, 100)
```

Message box

Simple Message box

```
#include "../../lv_examples.h"
#if LV_USE_MSGBOX && LV_BUILD_EXAMPLES

static void event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_current_target(e);
    LV_LOG_USER("Button %s clicked", lv_msgbox_get_active_btn_text(obj));
}
```

(下页继续)

(续上页)

```
void lv_example_msgbox_1(void)
{
    static const char * btns[] ={"Apply", "Close", ""};

    lv_obj_t * mbox1 = lv_msgbox_create(NULL, "Hello", "This is a message box with␣
↪two buttons.", btns, true);
    lv_obj_add_event_cb(mbox1, event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    lv_obj_center(mbox1);
}

#endif
```

```
def event_cb(e):
    mbox = e.get_current_target()
    print("Button %s clicked" % mbox.get_active_btn_text())

btns = ["Apply", "Close", ""]

mbox1 = lv.msgbox(lv.scr_act(), "Hello", "This is a message box with two buttons.",␣
↪btns, True)
mbox1.add_event_cb(event_cb, lv.EVENT.VALUE_CHANGED, None)
mbox1.center()
```

Roller

Simple Roller

```
#include "../../lv_examples.h"
#if LV_USE_ROLLER && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        char buf[32];
        lv_roller_get_selected_str(obj, buf, sizeof(buf));
        LV_LOG_USER("Selected month: %s\n", buf);
    }
}
```

(下页继续)

(续上页)

```
/**
 * An infinite roller with the name of the months
 */
void lv_example_roller_1(void)
{
    lv_obj_t *roller1 = lv_roller_create(lv_scr_act());
    lv_roller_set_options(roller1,
                          "January\n"
                          "February\n"
                          "March\n"
                          "April\n"
                          "May\n"
                          "June\n"
                          "July\n"
                          "August\n"
                          "September\n"
                          "October\n"
                          "November\n"
                          "December",
                          LV_ROLLER_MODE_INFINITE);

    lv_roller_set_visible_row_count(roller1, 4);
    lv_obj_center(roller1);
    lv_obj_add_event_cb(roller1, event_handler, LV_EVENT_ALL, NULL);
}

#endif
```

```
def event_handler(e):
    code = e.get_code()
    obj = e.get_target()
    if code == lv.EVENT.VALUE_CHANGED:
        option = " "*10
        obj.get_selected_str(option, len(option))
        print("Selected month: " + option.strip())

#
# An infinite roller with the name of the months
#

roller1 = lv.roller(lv.scr_act())
roller1.set_options("\n".join([
    "January",
```

(下页继续)

(续上页)

```
    "February",
    "March",
    "April",
    "May",
    "June",
    "July",
    "August",
    "September",
    "October",
    "November",
    "December"]),lv.roller.MODE.INFINITE)

roller1.set_visible_row_count(4)
roller1.center()
roller1.add_event_cb(event_handler, lv.EVENT.ALL, None)
```

Styling the roller

```
#include "../../lv_examples.h"
#if LV_USE_ROLLER && LV_FONT_MONTSERRAT_22 && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        char buf[32];
        lv_roller_get_selected_str(obj, buf, sizeof(buf));
        LV_LOG_USER("Selected value: %s", buf);
    }
}

/**
 * Roller with various alignments and larger text in the selected area
 */
void lv_example_roller_2(void)
{
    /*A style to make the selected option larger*/
    static lv_style_t style_sel;
    lv_style_init(&style_sel);
    lv_style_set_text_font(&style_sel, &lv_font_montserrat_22);
```

(下页继续)

(续上页)

```
    const char * opts = "1\n2\n3\n4\n5\n6\n7\n8\n9\n10";
    lv_obj_t *roller;

    /*A roller on the left with left aligned text, and custom width*/
    roller = lv_roller_create(lv_scr_act());
    lv_roller_set_options(roller, opts, LV_ROLLER_MODE_NORMAL);
    lv_roller_set_visible_row_count(roller, 2);
    lv_obj_set_width(roller, 100);
    lv_obj_add_style(roller, &style_sel, LV_PART_SELECTED);
    lv_obj_set_style_text_align(roller, LV_TEXT_ALIGN_LEFT, 0);
    lv_obj_align(roller, LV_ALIGN_LEFT_MID, 10, 0);
    lv_obj_add_event_cb(roller, event_handler, LV_EVENT_ALL, NULL);
    lv_roller_set_selected(roller, 2, LV_ANIM_OFF);

    /*A roller on the middle with center aligned text, and auto (default) width*/
    roller = lv_roller_create(lv_scr_act());
    lv_roller_set_options(roller, opts, LV_ROLLER_MODE_NORMAL);
    lv_roller_set_visible_row_count(roller, 3);
    lv_obj_add_style(roller, &style_sel, LV_PART_SELECTED);
    lv_obj_align(roller, LV_ALIGN_CENTER, 0, 0);
    lv_obj_add_event_cb(roller, event_handler, LV_EVENT_ALL, NULL);
    lv_roller_set_selected(roller, 5, LV_ANIM_OFF);

    /*A roller on the right with right aligned text, and custom width*/
    roller = lv_roller_create(lv_scr_act());
    lv_roller_set_options(roller, opts, LV_ROLLER_MODE_NORMAL);
    lv_roller_set_visible_row_count(roller, 4);
    lv_obj_set_width(roller, 80);
    lv_obj_add_style(roller, &style_sel, LV_PART_SELECTED);
    lv_obj_set_style_text_align(roller, LV_TEXT_ALIGN_RIGHT, 0);
    lv_obj_align(roller, LV_ALIGN_RIGHT_MID, -10, 0);
    lv_obj_add_event_cb(roller, event_handler, LV_EVENT_ALL, NULL);
    lv_roller_set_selected(roller, 8, LV_ANIM_OFF);
}

#endif
```

```
import fs_driver


def event_handler(e):
    code = e.get_code()
```

(下页继续)

(续上页)

```
    obj = e.get_target()
    if code == lv.EVENT.VALUE_CHANGED:
        option = " "*10
        obj.get_selected_str(option, len(option))
        print("Selected value: %s\n" + option.strip())

#
# Roller with various alignments and larger text in the selected area
#

# A style to make the selected option larger
style_sel = lv.style_t()
style_sel.init()

try:
    style_sel.set_text_font(lv.font_montserrat_22)
except:
    fs_drv = lv.fs_drv_t()
    fs_driver.fs_register(fs_drv, 'S')
    print("montserrat-22 not enabled in lv_conf.h, dynamically loading the font")
    font_montserrat_22 = lv.font_load("S:" + "../../assets/font/montserrat-22.fnt")
    style_sel.set_text_font(font_montserrat_22)

opts = "\n".join(["1","2","3","4","5","6","7","8","9","10"])

# A roller on the left with left aligned text, and custom width
roller = lv.roller(lv.scr_act())
roller.set_options(opts, lv.roller.MODE.NORMAL)
roller.set_visible_row_count(2)
roller.set_width(100)
roller.add_style(style_sel, lv.PART.SELECTED)
roller.set_style_text_align(lv.TEXT_ALIGN.LEFT, 0)
roller.align(lv.ALIGN.LEFT_MID, 10, 0)
roller.add_event_cb(event_handler, lv.EVENT.ALL, None)
roller.set_selected(2, lv.ANIM.OFF)

# A roller in the middle with center aligned text, and auto (default) width
roller = lv.roller(lv.scr_act())
roller.set_options(opts, lv.roller.MODE.NORMAL)
roller.set_visible_row_count(3)
roller.add_style(style_sel, lv.PART.SELECTED)
roller.align(lv.ALIGN.CENTER, 0, 0)
roller.add_event_cb(event_handler, lv.EVENT.ALL, None)
```

(下页继续)

(续上页)

```
roller.set_selected(5, lv.ANIM.OFF)

# A roller on the right with right aligned text, and custom width
roller = lv.roller(lv.scr_act())
roller.set_options(opts, lv.roller.MODE.NORMAL)
roller.set_visible_row_count(4)
roller.set_width(80)
roller.add_style(style_sel, lv.PART.SELECTED)
roller.set_style_text_align(lv.TEXT_ALIGN.RIGHT, 0)
roller.align(lv.ALIGN.RIGHT_MID, -10, 0)
roller.add_event_cb(event_handler, lv.EVENT.ALL, None)
roller.set_selected(8, lv.ANIM.OFF)
```

add fade mask to roller

```
#include "../../lv_examples.h"
#if LV_USE_ROLLER && LV_DRAW_COMPLEX && LV_BUILD_EXAMPLES

static void mask_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target(e);

    static int16_t mask_top_id = -1;
    static int16_t mask_bottom_id = -1;

    if (code == LV_EVENT_COVER_CHECK) {
        lv_event_set_cover_res(e, LV_COVER_RES_MASKED);

    } else if (code == LV_EVENT_DRAW_MAIN_BEGIN) {
        /* add mask */
        const lv_font_t * font = lv_obj_get_style_text_font(obj, LV_PART_MAIN);
        lv_coord_t line_space = lv_obj_get_style_text_line_space(obj, LV_PART_MAIN);
        lv_coord_t font_h = lv_font_get_line_height(font);

        lv_area_t roller_coords;
        lv_obj_get_coords(obj, &roller_coords);

        lv_area_t rect_area;
        rect_area.x1 = roller_coords.x1;
        rect_area.x2 = roller_coords.x2;
        rect_area.y1 = roller_coords.y1;
```

(下页继续)

(续上页)

```
        rect_area.y2 = roller_coords.y1 + (lv_obj_get_height(obj) - font_h - line_
↪space) / 2;

        lv_draw_mask_fade_param_t * fade_mask_top = lv_mem_buf_get(sizeof(lv_draw_
↪mask_fade_param_t));
        lv_draw_mask_fade_init(fade_mask_top, &rect_area, LV_OPA_TRANSP, rect_area.y1,
↪ LV_OPA_COVER, rect_area.y2);
        mask_top_id = lv_draw_mask_add(fade_mask_top, NULL);

        rect_area.y1 = rect_area.y2 + font_h + line_space - 1;
        rect_area.y2 = roller_coords.y2;

        lv_draw_mask_fade_param_t * fade_mask_bottom =lv_mem_buf_get(sizeof(lv_draw_
↪mask_fade_param_t));
        lv_draw_mask_fade_init(fade_mask_bottom, &rect_area, LV_OPA_COVER, rect_area.
↪y1, LV_OPA_TRANSP, rect_area.y2);
        mask_bottom_id = lv_draw_mask_add(fade_mask_bottom, NULL);

    } else if (code == LV_EVENT_DRAW_POST_END) {
        lv_draw_mask_fade_param_t * fade_mask_top = lv_draw_mask_remove_id(mask_top_
↪id);
        lv_draw_mask_fade_param_t * fade_mask_bottom = lv_draw_mask_remove_id(mask_
↪bottom_id);
        lv_draw_mask_free_param(fade_mask_top);
        lv_draw_mask_free_param(fade_mask_bottom);
        lv_mem_buf_release(fade_mask_top);
        lv_mem_buf_release(fade_mask_bottom);
        mask_top_id = -1;
        mask_bottom_id = -1;
    }
}

/**
 * Add a fade mask to roller.
 */
void lv_example_roller_3(void)
{
    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_bg_color(&style, lv_color_black());
    lv_style_set_text_color(&style, lv_color_white());
    lv_style_set_border_width(&style, 0);
    lv_style_set_pad_all(&style, 0);
```

(下页继续)

(续上页)

```
    lv_obj_add_style(lv_scr_act(), &style, 0);

    lv_obj_t *roller1 = lv_roller_create(lv_scr_act());
    lv_obj_add_style(roller1, &style, 0);
    lv_obj_set_style_bg_opa(roller1, LV_OPA_TRANSP, LV_PART_SELECTED);

#if LV_FONT_MONTSERRAT_22
    lv_obj_set_style_text_font(roller1, &lv_font_montserrat_22, LV_PART_SELECTED);
#endif

    lv_roller_set_options(roller1,
                          "January\n"
                          "February\n"
                          "March\n"
                          "April\n"
                          "May\n"
                          "June\n"
                          "July\n"
                          "August\n"
                          "September\n"
                          "October\n"
                          "November\n"
                          "December",
                          LV_ROLLER_MODE_NORMAL);

    lv_obj_center(roller1);
    lv_roller_set_visible_row_count(roller1, 3);
    lv_obj_add_event_cb(roller1, mask_event_cb, LV_EVENT_ALL, NULL);
}

#endif
```

```
import fs_driver
import sys

class Lv_Roller_3():

    def __init__(self):
        self.mask_top_id = -1
        self.mask_bottom_id = -1

        #
        # Add a fade mask to roller.
```

(下页继续)

(续上页)

```
        #
        style = lv.style_t()
        style.init()
        style.set_bg_color(lv.color_black())
        style.set_text_color(lv.color_white())

        lv.scr_act().add_style(style, 0)

        roller1 = lv.roller(lv.scr_act())
        roller1.add_style(style, 0)
        roller1.set_style_border_width(0, 0)
        roller1.set_style_pad_all(0, 0)
        roller1.set_style_bg_opa(lv.OPA.TRANSP, lv.PART.SELECTED)

        #if LV_FONT_MONTSERRAT_22
        #    lv_obj_set_style_text_font(roller1, &lv_font_montserrat_22, LV_PART_
↪SELECTED);
        #endif
        try:
            roller1.set_style_text_font(lv.font_montserrat_22,lv.PART.SELECTED)
        except:
            fs_drv = lv.fs_drv_t()
            fs_driver.fs_register(fs_drv, 'S')
            print("montserrat-22 not enabled in lv_conf.h, dynamically loading the␣
↪font")
            font_montserrat_22 = lv.font_load("S:" + "../../assets/font/montserrat-22.
↪fnt")
            roller1.set_style_text_font(font_montserrat_22,lv.PART.SELECTED)

        roller1.set_options("\n".join([
            "January",
            "February",
            "March",
            "April",
            "May",
            "June",
            "July",
            "August",
            "September",
            "October",
            "November",
            "December"]),lv.roller.MODE.NORMAL)
```

(下页继续)

(续上页)

```
        roller1.center()
        roller1.set_visible_row_count(3)
        roller1.add_event_cb(self.mask_event_cb, lv.EVENT.ALL, None)

    def mask_event_cb(self,e):

        code = e.get_code()
        obj = e.get_target()

        if code == lv.EVENT.COVER_CHECK:
            e.set_cover_res(lv.COVER_RES.MASKED)

        elif code == lv.EVENT.DRAW_MAIN_BEGIN:
            # add mask
            font = obj.get_style_text_font(lv.PART.MAIN)
            line_space = obj.get_style_text_line_space(lv.PART.MAIN)
            font_h = font.get_line_height()

            roller_coords = lv.area_t()
            obj.get_coords(roller_coords)

            rect_area = lv.area_t()
            rect_area.x1 = roller_coords.x1
            rect_area.x2 = roller_coords.x2
            rect_area.y1 = roller_coords.y1
            rect_area.y2 = roller_coords.y1 + (obj.get_height() - font_h - line_
↪space) // 2

            fade_mask_top = lv.draw_mask_fade_param_t()
            fade_mask_top.init(rect_area, lv.OPA.TRANSP, rect_area.y1, lv.OPA.COVER,␣
↪rect_area.y2)
            self.mask_top_id = lv.draw_mask_add(fade_mask_top,None)

            rect_area.y1 = rect_area.y2 + font_h + line_space - 1
            rect_area.y2 = roller_coords.y2

            fade_mask_bottom = lv.draw_mask_fade_param_t()
            fade_mask_bottom.init(rect_area, lv.OPA.COVER, rect_area.y1, lv.OPA.
↪TRANSP, rect_area.y2)
            self.mask_bottom_id = lv.draw_mask_add(fade_mask_bottom, None)

        elif code == lv.EVENT.DRAW_POST_END:
            fade_mask_top = lv.draw_mask_remove_id(self.mask_top_id)
```

(下页继续)

(续上页)

```
            fade_mask_bottom = lv.draw_mask_remove_id(self.mask_bottom_id)
            # Remove the masks
            lv.draw_mask_remove_id(self.mask_top_id)
            lv.draw_mask_remove_id(self.mask_bottom_id)
            self.mask_top_id = -1
            self.mask_bottom_id = -1

roller3 = Lv_Roller_3()
```

Slider

Simple Slider

```
#include "../../lv_examples.h"
#if LV_USE_SLIDER && LV_BUILD_EXAMPLES

static void slider_event_cb(lv_event_t * e);
static lv_obj_t * slider_label;

/**
 * A default slider with a label displaying the current value
 */
void lv_example_slider_1(void)
{
    /*Create a slider in the center of the display*/
    lv_obj_t * slider = lv_slider_create(lv_scr_act());
    lv_obj_center(slider);
    lv_obj_add_event_cb(slider, slider_event_cb, LV_EVENT_VALUE_CHANGED, NULL);

    /*Create a label below the slider*/
    slider_label = lv_label_create(lv_scr_act());
    lv_label_set_text(slider_label, "0%");

    lv_obj_align_to(slider_label, slider, LV_ALIGN_OUT_BOTTOM_MID, 0, 10);
}

static void slider_event_cb(lv_event_t * e)
{
    lv_obj_t * slider = lv_event_get_target(e);
    char buf[8];
    lv_snprintf(buf, sizeof(buf), "%d%%", (int)lv_slider_get_value(slider));
    lv_label_set_text(slider_label, buf);
```

(下页继续)

(续上页)

```
    lv_obj_align_to(slider_label, slider, LV_ALIGN_OUT_BOTTOM_MID, 0, 10);
}

#endif
```

```
#
# A default slider with a label displaying the current value
#
def slider_event_cb(e):

    slider = e.get_target()
    slider_label.set_text("{:d}%".format(slider.get_value()))
    slider_label.align_to(slider, lv.ALIGN.OUT_BOTTOM_MID, 0, 10)

# Create a slider in the center of the display
slider = lv.slider(lv.scr_act())
slider.center()
slider.add_event_cb(slider_event_cb, lv.EVENT.VALUE_CHANGED, None)

# Create a label below the slider
slider_label = lv.label(lv.scr_act())
slider_label.set_text("0%")

slider_label.align_to(slider, lv.ALIGN.OUT_BOTTOM_MID, 0, 10)
```

Slider with custom style

```
#include "../../lv_examples.h"
#if LV_USE_SLIDER && LV_BUILD_EXAMPLES



/**
 * Show how to style a slider.
 */
void lv_example_slider_2(void)
{
    /*Create a transition*/
    static const lv_style_prop_t props[] = {LV_STYLE_BG_COLOR, 0};
    static lv_style_transition_dsc_t transition_dsc;
    lv_style_transition_dsc_init(&transition_dsc, props, lv_anim_path_linear, 300, 0,␣
↪NULL);
```

(下页继续)

(续上页)

```
    static lv_style_t style_main;
    static lv_style_t style_indicator;
    static lv_style_t style_knob;
    static lv_style_t style_pressed_color;
    lv_style_init(&style_main);
    lv_style_set_bg_opa(&style_main, LV_OPA_COVER);
    lv_style_set_bg_color(&style_main, lv_color_hex3(0xbbb));
    lv_style_set_radius(&style_main, LV_RADIUS_CIRCLE);
    lv_style_set_pad_ver(&style_main, -2); /*Makes the indicator larger*/

    lv_style_init(&style_indicator);
    lv_style_set_bg_opa(&style_indicator, LV_OPA_COVER);
    lv_style_set_bg_color(&style_indicator, lv_palette_main(LV_PALETTE_CYAN));
    lv_style_set_radius(&style_indicator, LV_RADIUS_CIRCLE);
    lv_style_set_transition(&style_indicator, &transition_dsc);

    lv_style_init(&style_knob);
    lv_style_set_bg_opa(&style_knob, LV_OPA_COVER);
    lv_style_set_bg_color(&style_knob, lv_palette_main(LV_PALETTE_CYAN));
    lv_style_set_border_color(&style_knob, lv_palette_darken(LV_PALETTE_CYAN, 3));
    lv_style_set_border_width(&style_knob, 2);
    lv_style_set_radius(&style_knob, LV_RADIUS_CIRCLE);
    lv_style_set_pad_all(&style_knob, 6); /*Makes the knob larger*/
    lv_style_set_transition(&style_knob, &transition_dsc);

    lv_style_init(&style_pressed_color);
    lv_style_set_bg_color(&style_pressed_color, lv_palette_darken(LV_PALETTE_CYAN,␣
↪2));

    /*Create a slider and add the style*/
    lv_obj_t * slider = lv_slider_create(lv_scr_act());
    lv_obj_remove_style_all(slider);        /*Remove the styles coming from the␣
↪theme*/

    lv_obj_add_style(slider, &style_main, LV_PART_MAIN);
    lv_obj_add_style(slider, &style_indicator, LV_PART_INDICATOR);
    lv_obj_add_style(slider, &style_pressed_color, LV_PART_INDICATOR | LV_STATE_
↪PRESSED);
    lv_obj_add_style(slider, &style_knob, LV_PART_KNOB);
    lv_obj_add_style(slider, &style_pressed_color, LV_PART_KNOB | LV_STATE_PRESSED);

    lv_obj_center(slider);
```

(下页继续)

(续上页)

```
}

#endif
```

```
#
# Show how to style a slider.
#
# Create a transition
props = [lv.STYLE.BG_COLOR, 0]
transition_dsc = lv.style_transition_dsc_t()
transition_dsc.init(props, lv.anim_t.path_linear, 300, 0, None)

style_main = lv.style_t()
style_indicator = lv.style_t()
style_knob = lv.style_t()
style_pressed_color = lv.style_t()
style_main.init()
style_main.set_bg_opa(lv.OPA.COVER)
style_main.set_bg_color(lv.color_hex3(0xbbb))
style_main.set_radius(lv.RADIUS.CIRCLE)
style_main.set_pad_ver(-2)                  # Makes the indicator larger

style_indicator.init()
style_indicator.set_bg_opa(lv.OPA.COVER)
style_indicator.set_bg_color(lv.palette_main(lv.PALETTE.CYAN))
style_indicator.set_radius(lv.RADIUS.CIRCLE)
style_indicator.set_transition(transition_dsc)

style_knob.init()
style_knob.set_bg_opa(lv.OPA.COVER)
style_knob.set_bg_color(lv.palette_main(lv.PALETTE.CYAN))
style_knob.set_border_color(lv.palette_darken(lv.PALETTE.CYAN, 3))
style_knob.set_border_width(2)
style_knob.set_radius(lv.RADIUS.CIRCLE)
style_knob.set_pad_all(6)                    # Makes the knob larger
style_knob.set_transition(transition_dsc)

style_pressed_color.init()
style_pressed_color.set_bg_color(lv.palette_darken(lv.PALETTE.CYAN, 2))

# Create a slider and add the style
slider = lv.slider(lv.scr_act())
slider.remove_style_all()                    # Remove the styles coming from the theme
```

(下页继续)

(续上页)

```
slider.add_style(style_main, lv.PART.MAIN)
slider.add_style(style_indicator, lv.PART.INDICATOR)
slider.add_style(style_pressed_color, lv.PART.INDICATOR | lv.STATE.PRESSED)
slider.add_style(style_knob, lv.PART.KNOB)
slider.add_style(style_pressed_color, lv.PART.KNOB | lv.STATE.PRESSED)

slider.center()
```

Slider with extended drawer

```
#include "../../lv_examples.h"
#if LV_USE_SLIDER && LV_BUILD_EXAMPLES

static void slider_event_cb(lv_event_t * e);

/**
 * Show the current value when the slider is pressed by extending the drawer
 *
 */
void lv_example_slider_3(void)
{
    /*Create a slider in the center of the display*/
    lv_obj_t * slider;
    slider = lv_slider_create(lv_scr_act());
    lv_obj_center(slider);

    lv_slider_set_mode(slider, LV_SLIDER_MODE_RANGE);
    lv_slider_set_value(slider, 70, LV_ANIM_OFF);
    lv_slider_set_left_value(slider, 20, LV_ANIM_OFF);

    lv_obj_add_event_cb(slider, slider_event_cb, LV_EVENT_ALL, NULL);
    lv_obj_refresh_ext_draw_size(slider);
}

static void slider_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target(e);

    /*Provide some extra space for the value*/
```

(下页继续)

(续上页)

```
    if(code == LV_EVENT_REFR_EXT_DRAW_SIZE) {
        lv_coord_t * size = lv_event_get_param(e);
        *size = LV_MAX(*size, 50);
    }
    else if(code == LV_EVENT_DRAW_PART_END) {
        lv_obj_draw_part_dsc_t * dsc = lv_event_get_param(e);
        if(dsc->part == LV_PART_INDICATOR) {
            char buf[16];
            lv_snprintf(buf, sizeof(buf), "%d - %d", (int)lv_slider_get_left_
↪value(obj), (int)lv_slider_get_value(obj));

            lv_point_t label_size;
            lv_txt_get_size(&label_size, buf, LV_FONT_DEFAULT, 0, 0, LV_COORD_MAX, 0);
            lv_area_t label_area;
            label_area.x1 = dsc->draw_area->x1 + lv_area_get_width(dsc->draw_area) /␣
↪2 - label_size.x / 2;
            label_area.x2 = label_area.x1 + label_size.x;
            label_area.y2 = dsc->draw_area->y1 - 10;
            label_area.y1 = label_area.y2 - label_size.y;

            lv_draw_label_dsc_t label_draw_dsc;
            lv_draw_label_dsc_init(&label_draw_dsc);

            lv_draw_label(dsc->draw_ctx, &label_draw_dsc, &label_area, buf, NULL);
        }
    }
}

#endif
```

```
def slider_event_cb(e):
    code = e.get_code()
    obj = e.get_target()

    # Provide some extra space for the value
    if code == lv.EVENT.REFR_EXT_DRAW_SIZE:
        e.set_ext_draw_size(50)

    elif code == lv.EVENT.DRAW_PART_END:
        # print("DRAW_PART_END")
        dsc = lv.obj_draw_part_dsc_t.__cast__(e.get_param())
        # print(dsc)
        if dsc.part == lv.PART.INDICATOR:
```

(下页继续)

(续上页)

```
            label_text = "{:d} - {:d}".format(obj.get_left_value(),slider.get_value())
            label_size = lv.point_t()
            lv.txt_get_size(label_size, label_text, lv.font_default(), 0, 0, lv.COORD.
↪MAX, 0)
            # print(label_size.x,label_size.y)
            label_area = lv.area_t()
            label_area.x1 = dsc.draw_area.x1 + dsc.draw_area.get_width() // 2 - label_
↪size.x // 2
            label_area.x2 = label_area.x1 + label_size.x
            label_area.y2 = dsc.draw_area.y1 - 10
            label_area.y1 = label_area.y2 - label_size.y

            label_draw_dsc = lv.draw_label_dsc_t()
            label_draw_dsc.init()

            dsc.draw_ctx.label(label_draw_dsc, label_area, label_text, None)
#
# Show the current value when the slider if pressed by extending the drawer
#
#
#Create a slider in the center of the display

slider = lv.slider(lv.scr_act())
slider.center()

slider.set_mode(lv.slider.MODE.RANGE)
slider.set_value(70, lv.ANIM.OFF)
slider.set_left_value(20, lv.ANIM.OFF)

slider.add_event_cb(slider_event_cb, lv.EVENT.ALL, None)
slider.refresh_ext_draw_size()
```

Span

Span with custom styles

```
#include "../../lv_examples.h"
#if LV_USE_SPAN && LV_BUILD_EXAMPLES

/**
 * Create span.
```

(下页继续)

(续上页)

```
 */
void lv_example_span_1(void)
{
    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_border_width(&style, 1);
    lv_style_set_border_color(&style, lv_palette_main(LV_PALETTE_ORANGE));
    lv_style_set_pad_all(&style, 2);

    lv_obj_t * spans = lv_spangroup_create(lv_scr_act());
    lv_obj_set_width(spans, 300);
    lv_obj_set_height(spans,300);
    lv_obj_center(spans);
    lv_obj_add_style(spans, &style, 0);

    lv_spangroup_set_align(spans, LV_TEXT_ALIGN_LEFT);
    lv_spangroup_set_overflow(spans, LV_SPAN_OVERFLOW_CLIP);
    lv_spangroup_set_indent(spans, 20);
    lv_spangroup_set_mode(spans, LV_SPAN_MODE_BREAK);

    lv_span_t * span = lv_spangroup_new_span(spans);
    lv_span_set_text(span, "China is a beautiful country.");
    lv_style_set_text_color(&span->style, lv_palette_main(LV_PALETTE_RED));
    lv_style_set_text_decor(&span->style, LV_TEXT_DECOR_STRIKETHROUGH | LV_TEXT_DECOR_
↪UNDERLINE);
    lv_style_set_text_opa(&span->style, LV_OPA_50);

    span = lv_spangroup_new_span(spans);
    lv_span_set_text_static(span, "good good study, day day up.");
#if LV_FONT_MONTSERRAT_24
    lv_style_set_text_font(&span->style,  &lv_font_montserrat_24);
#endif
    lv_style_set_text_color(&span->style, lv_palette_main(LV_PALETTE_GREEN));

    span = lv_spangroup_new_span(spans);
    lv_span_set_text_static(span, "LVGL is an open-source graphics library.");
    lv_style_set_text_color(&span->style, lv_palette_main(LV_PALETTE_BLUE));

    span = lv_spangroup_new_span(spans);
    lv_span_set_text_static(span, "the boy no name.");
    lv_style_set_text_color(&span->style, lv_palette_main(LV_PALETTE_GREEN));
#if LV_FONT_MONTSERRAT_20
    lv_style_set_text_font(&span->style, &lv_font_montserrat_20);
```

(下页继续)

(续上页)

```
#endif
    lv_style_set_text_decor(&span->style, LV_TEXT_DECOR_UNDERLINE);

    span = lv_spangroup_new_span(spans);
    lv_span_set_text(span, "I have a dream that hope to come true.");

    lv_spangroup_refr_mode(spans);
}

#endif
```

```
#
# Create span
#
style = lv.style_t()
style.init()
style.set_border_width(1)
style.set_border_color(lv.palette_main(lv.PALETTE.ORANGE))
style.set_pad_all(2)

spans = lv.spangroup(lv.scr_act())
spans.set_width(300)
spans.set_height(300)
spans.center()
spans.add_style(style, 0)

spans.set_align(lv.TEXT_ALIGN.LEFT)
spans.set_overflow(lv.SPAN_OVERFLOW.CLIP)
spans.set_indent(20)
spans.set_mode(lv.SPAN_MODE.BREAK)

span = spans.new_span()
span.set_text("china is a beautiful country.")
span.style.set_text_color(lv.palette_main(lv.PALETTE.RED))
span.style.set_text_decor(lv.TEXT_DECOR.STRIKETHROUGH | lv.TEXT_DECOR.UNDERLINE)
span.style.set_text_opa(lv.OPA._30)

span = spans.new_span()
span.set_text_static("good good study, day day up.")
#if LV_FONT_MONTSERRAT_24
#    lv_style_set_text_font(&span->style,  &lv_font_montserrat_24);
#endif
span.style.set_text_color(lv.palette_main(lv.PALETTE.GREEN))
```

(下页继续)

(续上页)

```
span = spans.new_span()
span.set_text_static("LVGL is an open-source graphics library.")
span.style.set_text_color(lv.palette_main(lv.PALETTE.BLUE))

span = spans.new_span()
span.set_text_static("the boy no name.")
span.style.set_text_color(lv.palette_main(lv.PALETTE.GREEN))
#if LV_FONT_MONTSERRAT_20
#    lv_style_set_text_font(&span->style, &lv_font_montserrat_20);
#endif
span.style.set_text_decor(lv.TEXT_DECOR.UNDERLINE)

span = spans.new_span()
span.set_text("I have a dream that hope to come true.")

spans.refr_mode()

# lv_span_del(spans, span);
# lv_obj_del(spans);
```

Spinbox

Simple Spinbox

```
#include "../../lv_examples.h"
#if LV_USE_SPINBOX && LV_BUILD_EXAMPLES

static lv_obj_t * spinbox;


static void lv_spinbox_increment_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    if(code == LV_EVENT_SHORT_CLICKED || code  == LV_EVENT_LONG_PRESSED_REPEAT) {
        lv_spinbox_increment(spinbox);
    }
}

static void lv_spinbox_decrement_event_cb(lv_event_t * e)
{
```

(下页继续)

(续上页)

```
    lv_event_code_t code = lv_event_get_code(e);
    if(code == LV_EVENT_SHORT_CLICKED || code == LV_EVENT_LONG_PRESSED_REPEAT) {
        lv_spinbox_decrement(spinbox);
    }
}


void lv_example_spinbox_1(void)
{
    spinbox = lv_spinbox_create(lv_scr_act());
    lv_spinbox_set_range(spinbox, -1000, 25000);
    lv_spinbox_set_digit_format(spinbox, 5, 2);
    lv_spinbox_step_prev(spinbox);
    lv_obj_set_width(spinbox, 100);
    lv_obj_center(spinbox);

    lv_coord_t h = lv_obj_get_height(spinbox);

    lv_obj_t * btn = lv_btn_create(lv_scr_act());
    lv_obj_set_size(btn, h, h);
    lv_obj_align_to(btn, spinbox, LV_ALIGN_OUT_RIGHT_MID, 5, 0);
    lv_obj_set_style_bg_img_src(btn, LV_SYMBOL_PLUS, 0);
    lv_obj_add_event_cb(btn, lv_spinbox_increment_event_cb, LV_EVENT_ALL,  NULL);

    btn = lv_btn_create(lv_scr_act());
    lv_obj_set_size(btn, h, h);
    lv_obj_align_to(btn, spinbox, LV_ALIGN_OUT_LEFT_MID, -5, 0);
    lv_obj_set_style_bg_img_src(btn, LV_SYMBOL_MINUS, 0);
    lv_obj_add_event_cb(btn, lv_spinbox_decrement_event_cb, LV_EVENT_ALL, NULL);
}

#endif
```

```
def increment_event_cb(e):
    code = e.get_code()
    if code == lv.EVENT.SHORT_CLICKED or code  == lv.EVENT.LONG_PRESSED_REPEAT:
        spinbox.increment()

def decrement_event_cb(e):
    code = e.get_code()
    if code == lv.EVENT.SHORT_CLICKED or code == lv.EVENT.LONG_PRESSED_REPEAT:
        spinbox.decrement()
```

(下页继续)

(续上页)

```
spinbox = lv.spinbox(lv.scr_act())
spinbox.set_range(-1000, 25000)
spinbox.set_digit_format(5, 2)
spinbox.step_prev()
spinbox.set_width(100)
spinbox.center()

h = spinbox.get_height()

btn = lv.btn(lv.scr_act())
btn.set_size(h, h)
btn.align_to(spinbox, lv.ALIGN.OUT_RIGHT_MID, 5, 0)
btn.set_style_bg_img_src(lv.SYMBOL.PLUS, 0)
btn.add_event_cb(increment_event_cb, lv.EVENT.ALL,  None)

btn = lv.btn(lv.scr_act())
btn.set_size(h, h)
btn.align_to(spinbox, lv.ALIGN.OUT_LEFT_MID, -5, 0)
btn.set_style_bg_img_src(lv.SYMBOL.MINUS, 0)
btn.add_event_cb(decrement_event_cb, lv.EVENT.ALL, None)
```

Spinner

Simple spinner

```
#include "../../lv_examples.h"
#if LV_USE_SPINNER && LV_BUILD_EXAMPLES

void lv_example_spinner_1(void)
{
    /*Create a spinner*/
    lv_obj_t * spinner = lv_spinner_create(lv_scr_act(), 1000, 60);
    lv_obj_set_size(spinner, 100, 100);
    lv_obj_center(spinner);
}

#endif
```

```
# Create a spinner
spinner = lv.spinner(lv.scr_act(), 1000, 60)
spinner.set_size(100, 100)
spinner.center()
```

(下页继续)

(续上页)

```
```

Switch

Simple Switch

```
#include "../../lv_examples.h"
#if LV_USE_SWITCH && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        LV_LOG_USER("State: %s\n", lv_obj_has_state(obj, LV_STATE_CHECKED) ? "On" :
↪"Off");
    }
}

void lv_example_switch_1(void)
{
    lv_obj_set_flex_flow(lv_scr_act(), LV_FLEX_FLOW_COLUMN);
    lv_obj_set_flex_align(lv_scr_act(), LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER,␣
↪LV_FLEX_ALIGN_CENTER);

    lv_obj_t * sw;

    sw = lv_switch_create(lv_scr_act());
    lv_obj_add_event_cb(sw, event_handler, LV_EVENT_ALL, NULL);

    sw = lv_switch_create(lv_scr_act());
    lv_obj_add_state(sw, LV_STATE_CHECKED);
    lv_obj_add_event_cb(sw, event_handler, LV_EVENT_ALL, NULL);

    sw = lv_switch_create(lv_scr_act());
    lv_obj_add_state(sw, LV_STATE_DISABLED);
    lv_obj_add_event_cb(sw, event_handler, LV_EVENT_ALL, NULL);

    sw = lv_switch_create(lv_scr_act());
    lv_obj_add_state(sw, LV_STATE_CHECKED | LV_STATE_DISABLED);
    lv_obj_add_event_cb(sw, event_handler, LV_EVENT_ALL, NULL);
```

(下页继续)

(续上页)

```
}

#endif
```

```
def event_handler(e):
    code = e.get_code()
    obj = e.get_target()
    if code == lv.EVENT.VALUE_CHANGED:
        if obj.has_state(lv.STATE.CHECKED):
            print("State: on")
        else:
            print("State: off")


lv.scr_act().set_flex_flow(lv.FLEX_FLOW.COLUMN)
lv.scr_act().set_flex_align(lv.FLEX_ALIGN.CENTER, lv.FLEX_ALIGN.CENTER, lv.FLEX_ALIGN.
↪CENTER)

sw = lv.switch(lv.scr_act())
sw.add_event_cb(event_handler,lv.EVENT.ALL, None)

sw = lv.switch(lv.scr_act())
sw.add_state(lv.STATE.CHECKED)
sw.add_event_cb(event_handler, lv.EVENT.ALL, None)

sw = lv.switch(lv.scr_act())
sw.add_state(lv.STATE.DISABLED)
sw.add_event_cb(event_handler, lv.EVENT.ALL, None)

sw = lv.switch(lv.scr_act())
sw.add_state(lv.STATE.CHECKED | lv.STATE.DISABLED)
sw.add_event_cb(event_handler, lv.EVENT.ALL, None)
```

Table

Simple table

```
#include "../../lv_examples.h"
#if LV_USE_TABLE && LV_BUILD_EXAMPLES

static void draw_part_event_cb(lv_event_t * e)
```

(下页继续)

(续上页)

```
{
    lv_obj_t * obj = lv_event_get_target(e);
    lv_obj_draw_part_dsc_t * dsc = lv_event_get_param(e);
    /*If the cells are drawn...*/
    if(dsc->part == LV_PART_ITEMS) {
        uint32_t row = dsc->id /  lv_table_get_col_cnt(obj);
        uint32_t col = dsc->id - row * lv_table_get_col_cnt(obj);

        /*Make the texts in the first cell center aligned*/
        if(row == 0) {
            dsc->label_dsc->align = LV_TEXT_ALIGN_CENTER;
            dsc->rect_dsc->bg_color = lv_color_mix(lv_palette_main(LV_PALETTE_BLUE),␣
↪dsc->rect_dsc->bg_color, LV_OPA_20);
            dsc->rect_dsc->bg_opa = LV_OPA_COVER;
        }
        /*In the first column align the texts to the right*/
        else if(col == 0) {
            dsc->label_dsc->align = LV_TEXT_ALIGN_RIGHT;
        }

        /*MAke every 2nd row grayish*/
        if((row != 0 && row % 2) == 0) {
            dsc->rect_dsc->bg_color = lv_color_mix(lv_palette_main(LV_PALETTE_GREY),␣
↪dsc->rect_dsc->bg_color, LV_OPA_10);
            dsc->rect_dsc->bg_opa = LV_OPA_COVER;
        }
    }
}


void lv_example_table_1(void)
{
    lv_obj_t * table = lv_table_create(lv_scr_act());

    /*Fill the first column*/
    lv_table_set_cell_value(table, 0, 0, "Name");
    lv_table_set_cell_value(table, 1, 0, "Apple");
    lv_table_set_cell_value(table, 2, 0, "Banana");
    lv_table_set_cell_value(table, 3, 0, "Lemon");
    lv_table_set_cell_value(table, 4, 0, "Grape");
    lv_table_set_cell_value(table, 5, 0, "Melon");
    lv_table_set_cell_value(table, 6, 0, "Peach");
    lv_table_set_cell_value(table, 7, 0, "Nuts");
```

(下页继续)

(续上页)

```
    /*Fill the second column*/
    lv_table_set_cell_value(table, 0, 1, "Price");
    lv_table_set_cell_value(table, 1, 1, "$7");
    lv_table_set_cell_value(table, 2, 1, "$4");
    lv_table_set_cell_value(table, 3, 1, "$6");
    lv_table_set_cell_value(table, 4, 1, "$2");
    lv_table_set_cell_value(table, 5, 1, "$5");
    lv_table_set_cell_value(table, 6, 1, "$1");
    lv_table_set_cell_value(table, 7, 1, "$9");

    /*Set a smaller height to the table. It'll make it scrollable*/
    lv_obj_set_height(table, 200);
    lv_obj_center(table);

    /*Add an event callback to to apply some custom drawing*/
    lv_obj_add_event_cb(table, draw_part_event_cb, LV_EVENT_DRAW_PART_BEGIN, NULL);
}

#endif
```

```
def draw_part_event_cb(e):
    obj = e.get_target()
    dsc = lv.obj_draw_part_dsc_t.__cast__(e.get_param())
    # If the cells are drawn../
    if dsc.part == lv.PART.ITEMS:
        row = dsc.id //  obj.get_col_cnt()
        col = dsc.id - row * obj.get_col_cnt()

        # Make the texts in the first cell center aligned
        if row == 0:
            dsc.label_dsc.align = lv.TEXT_ALIGN.CENTER
            dsc.rect_dsc.bg_color = lv.palette_main(lv.PALETTE.BLUE).color_mix(dsc.
↪rect_dsc.bg_color, lv.OPA._20)
            dsc.rect_dsc.bg_opa = lv.OPA.COVER

        # In the first column align the texts to the right
        elif col == 0:
            dsc.label_dsc.flag = lv.TEXT_ALIGN.RIGHT

        # Make every 2nd row grayish
        if row != 0 and (row % 2) == 0:
            dsc.rect_dsc.bg_color = lv.palette_main(lv.PALETTE.GREY).color_mix(dsc.
↪rect_dsc.bg_color, lv.OPA._10)
```

(下页继续)

(续上页)

```
            dsc.rect_dsc.bg_opa = lv.OPA.COVER


table = lv.table(lv.scr_act())

# Fill the first column
table.set_cell_value(0, 0, "Name")
table.set_cell_value(1, 0, "Apple")
table.set_cell_value(2, 0, "Banana")
table.set_cell_value(3, 0, "Lemon")
table.set_cell_value(4, 0, "Grape")
table.set_cell_value(5, 0, "Melon")
table.set_cell_value(6, 0, "Peach")
table.set_cell_value(7, 0, "Nuts")

# Fill the second column
table.set_cell_value(0, 1, "Price")
table.set_cell_value(1, 1, "$7")
table.set_cell_value(2, 1, "$4")
table.set_cell_value(3, 1, "$6")
table.set_cell_value(4, 1, "$2")
table.set_cell_value(5, 1, "$5")
table.set_cell_value(6, 1, "$1")
table.set_cell_value(7, 1, "$9")

# Set a smaller height to the table. It'll make it scrollable
table.set_height(200)
table.center()

# Add an event callback to apply some custom drawing
table.add_event_cb(draw_part_event_cb, lv.EVENT.DRAW_PART_BEGIN, None)
```

Lightweighted list from table

```
#include "../../lv_examples.h"
#if LV_USE_TABLE && LV_BUILD_EXAMPLES

#define ITEM_CNT 200

static void draw_event_cb(lv_event_t * e)
{
```

(下页继续)

(续上页)

```
    lv_obj_t * obj = lv_event_get_target(e);
    lv_obj_draw_part_dsc_t * dsc = lv_event_get_draw_part_dsc(e);
    /*If the cells are drawn...*/
    if(dsc->part == LV_PART_ITEMS) {
        bool chk = lv_table_has_cell_ctrl(obj, dsc->id, 0, LV_TABLE_CELL_CTRL_CUSTOM_
↪1);

        lv_draw_rect_dsc_t rect_dsc;
        lv_draw_rect_dsc_init(&rect_dsc);
        rect_dsc.bg_color = chk ? lv_theme_get_color_primary(obj) : lv_palette_
↪lighten(LV_PALETTE_GREY, 2);
        rect_dsc.radius = LV_RADIUS_CIRCLE;

        lv_area_t sw_area;
        sw_area.x1 = dsc->draw_area->x2 - 50;
        sw_area.x2 = sw_area.x1 + 40;
        sw_area.y1 = dsc->draw_area->y1 + lv_area_get_height(dsc->draw_area) / 2 - 10;
        sw_area.y2 = sw_area.y1 + 20;
        lv_draw_rect(dsc->draw_ctx, &rect_dsc, &sw_area);

        rect_dsc.bg_color = lv_color_white();
        if(chk) {
            sw_area.x2 -= 2;
            sw_area.x1 = sw_area.x2 - 16;
        } else {
            sw_area.x1 += 2;
            sw_area.x2 = sw_area.x1 + 16;
        }
        sw_area.y1 += 2;
        sw_area.y2 -= 2;
        lv_draw_rect(dsc->draw_ctx, &rect_dsc, &sw_area);
    }
}

static void change_event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target(e);
    uint16_t col;
    uint16_t row;
    lv_table_get_selected_cell(obj, &row, &col);
    bool chk = lv_table_has_cell_ctrl(obj, row, 0, LV_TABLE_CELL_CTRL_CUSTOM_1);
    if(chk) lv_table_clear_cell_ctrl(obj, row, 0, LV_TABLE_CELL_CTRL_CUSTOM_1);
    else lv_table_add_cell_ctrl(obj, row, 0, LV_TABLE_CELL_CTRL_CUSTOM_1);
```

(下页继续)

(续上页)

```
}


/**
 * A very light-weighted list created from table
 */
void lv_example_table_2(void)
{
    /*Measure memory usage*/
    lv_mem_monitor_t mon1;
    lv_mem_monitor(&mon1);

    uint32_t t = lv_tick_get();

    lv_obj_t * table = lv_table_create(lv_scr_act());

    /*Set a smaller height to the table. It'll make it scrollable*/
    lv_obj_set_size(table, LV_SIZE_CONTENT, 200);

    lv_table_set_col_width(table, 0, 150);
    lv_table_set_row_cnt(table, ITEM_CNT); /*Not required but avoids a lot of memory␣
↪reallocation lv_table_set_set_value*/
    lv_table_set_col_cnt(table, 1);

    /*Don't make the cell pressed, we will draw something different in the event*/
    lv_obj_remove_style(table, NULL, LV_PART_ITEMS | LV_STATE_PRESSED);

    uint32_t i;
    for(i = 0; i < ITEM_CNT; i++) {
        lv_table_set_cell_value_fmt(table, i, 0, "Item %"LV_PRIu32, i + 1);
    }

    lv_obj_align(table, LV_ALIGN_CENTER, 0, -20);

    /*Add an event callback to to apply some custom drawing*/
    lv_obj_add_event_cb(table, draw_event_cb, LV_EVENT_DRAW_PART_END, NULL);
    lv_obj_add_event_cb(table, change_event_cb, LV_EVENT_VALUE_CHANGED, NULL);

    lv_mem_monitor_t mon2;
    lv_mem_monitor(&mon2);

    uint32_t mem_used = mon1.free_size - mon2.free_size;
```

(下页继续)

(续上页)

```
    uint32_t elaps = lv_tick_elaps(t);

    lv_obj_t * label = lv_label_create(lv_scr_act());
    lv_label_set_text_fmt(label, "%"LV_PRIu32" items were created in %"LV_PRIu32" ms\n
↪"
                                 "using %"LV_PRIu32" bytes of memory",
                                 ITEM_CNT, elaps, mem_used);

    lv_obj_align(label, LV_ALIGN_BOTTOM_MID, 0, -10);

}

#endif
```

```
from utime import ticks_ms
import gc

ITEM_CNT = 200

def draw_event_cb(e):
    obj = e.get_target()
    dsc = lv.obj_draw_part_dsc_t.__cast__(e.get_param())
    # If the cells are drawn...
    if dsc.part == lv.PART.ITEMS:
        chk = obj.has_cell_ctrl(dsc.id, 0, lv.table.CELL_CTRL.CUSTOM_1)

        rect_dsc = lv.draw_rect_dsc_t()
        rect_dsc.init()

        if chk:
            rect_dsc.bg_color = lv.theme_get_color_primary(obj)
        else:
            rect_dsc.bg_color = lv.palette_lighten(lv.PALETTE.GREY, 2)

        rect_dsc.radius = lv.RADIUS.CIRCLE

        sw_area = lv.area_t()
        sw_area.x1 = dsc.draw_area.x2 - 50
        sw_area.x2 = sw_area.x1 + 40
        sw_area.y1 = dsc.draw_area.y1 + dsc.draw_area.get_height() // 2 - 10
        sw_area.y2 = sw_area.y1 + 20
        dsc.draw_ctx.rect(rect_dsc, sw_area)
```

(下页继续)

(续上页)

```
        rect_dsc.bg_color = lv.color_white()

        if chk:
            sw_area.x2 -= 2
            sw_area.x1 = sw_area.x2 - 16
        else:
            sw_area.x1 += 2
            sw_area.x2 = sw_area.x1 + 16
        sw_area.y1 += 2
        sw_area.y2 -= 2
        dsc.draw_ctx.rect(rect_dsc, sw_area)

def change_event_cb(e):
    obj = e.get_target()
    row = lv.C_Pointer()
    col = lv.C_Pointer()
    table.get_selected_cell(row, col)
    # print("row: ",row.uint_val)

    chk = table.has_cell_ctrl(row.uint_val, 0, lv.table.CELL_CTRL.CUSTOM_1)
    if chk:
        table.clear_cell_ctrl(row.uint_val, 0, lv.table.CELL_CTRL.CUSTOM_1)
    else:
        table.add_cell_ctrl(row.uint_val, 0, lv.table.CELL_CTRL.CUSTOM_1)

#
# A very light-weighted list created from table
#

# Measure memory usage
gc.enable()
gc.collect()
mem_free = gc.mem_free()
print("mem_free: ", mem_free)
t = ticks_ms()
print("ticks: ", t)
table = lv.table(lv.scr_act())

# Set a smaller height to the table. It'll make it scrollable
table.set_size(150, 200)

table.set_col_width(0, 150)
table.set_row_cnt(ITEM_CNT)  # Not required but avoids a lot of memory reallocation␣
↪lv_table_set_set_value
```

(下页继续)

(续上页)

```
table.set_col_cnt(1)

# Don't make the cell pressed, we will draw something different in the event
table.remove_style(None, lv.PART.ITEMS | lv.STATE.PRESSED)

for i in range(ITEM_CNT):
    table.set_cell_value(i, 0, "Item " + str(i+1))

table.align(lv.ALIGN.CENTER, 0, -20)

# Add an event callback to apply some custom drawing
table.add_event_cb(draw_event_cb, lv.EVENT.DRAW_PART_END, None)
table.add_event_cb(change_event_cb, lv.EVENT.VALUE_CHANGED, None)

gc.collect()
mem_used = mem_free - gc.mem_free()
elaps = ticks_ms()-t

label = lv.label(lv.scr_act())
label.set_text(str(ITEM_CNT) + " items were created in " + str(elaps) + " ms\n using
↪" + str(mem_used) + " bytes of memory")
#label.set_text(str(ITEM_CNT) + " items were created in " + str(elaps) + " ms")

label.align(lv.ALIGN.BOTTOM_MID, 0, -10)
```

Tabview

Simple Tabview

```
#include "../../lv_examples.h"
#if LV_USE_TABVIEW && LV_BUILD_EXAMPLES

void lv_example_tabview_1(void)
{
    /*Create a Tab view object*/
    lv_obj_t *tabview;
    tabview = lv_tabview_create(lv_scr_act(), LV_DIR_TOP, 50);

    /*Add 3 tabs (the tabs are page (lv_page) and can be scrolled*/
    lv_obj_t *tab1 = lv_tabview_add_tab(tabview, "Tab 1");
    lv_obj_t *tab2 = lv_tabview_add_tab(tabview, "Tab 2");
    lv_obj_t *tab3 = lv_tabview_add_tab(tabview, "Tab 3");
```

(下页继续)

(续上页)

```
    /*Add content to the tabs*/
    lv_obj_t * label = lv_label_create(tab1);
    lv_label_set_text(label, "This the first tab\n\n"
                             "If the content\n"
                             "of a tab\n"
                             "becomes too\n"
                             "longer\n"
                             "than the\n"
                             "container\n"
                             "then it\n"
                             "automatically\n"
                             "becomes\n"
                             "scrollable.\n"
                             "\n"
                             "\n"
                             "\n"
                             "Can you see it?");

    label = lv_label_create(tab2);
    lv_label_set_text(label, "Second tab");

    label = lv_label_create(tab3);
    lv_label_set_text(label, "Third tab");

    lv_obj_scroll_to_view_recursive(label, LV_ANIM_ON);

}
#endif
```

```
# Create a Tab view object
tabview = lv.tabview(lv.scr_act(), lv.DIR.TOP, 50)

# Add 3 tabs (the tabs are page (lv_page) and can be scrolled
tab1 = tabview.add_tab("Tab 1")
tab2 = tabview.add_tab("Tab 2")
tab3 = tabview.add_tab("Tab 3")

# Add content to the tabs
label = lv.label(tab1)
label.set_text("""This the first tab

If the content
```

(下页继续)

(续上页)

```
of a tab
becomes too
longer
than the
container
then it
automatically
becomes
scrollable.



Can you see it?""")

label = lv.label(tab2)
label.set_text("Second tab")

label = lv.label(tab3)
label.set_text("Third tab");

label.scroll_to_view_recursive(lv.ANIM.ON)
```

Tabs on the left, styling and no scrolling

```
#include "../../lv_examples.h"
#if LV_USE_TABVIEW && LV_BUILD_EXAMPLES

static void scroll_begin_event(lv_event_t * e)
{
    /*Disable the scroll animations. Triggered when a tab button is clicked */
    if(lv_event_get_code(e) == LV_EVENT_SCROLL_BEGIN) {
        lv_anim_t * a = lv_event_get_param(e);
        if(a)  a->time = 0;
    }
}

void lv_example_tabview_2(void)
{
    /*Create a Tab view object*/
    lv_obj_t *tabview;
    tabview = lv_tabview_create(lv_scr_act(), LV_DIR_LEFT, 80);
```

(下页继续)

(续上页)

```
    lv_obj_add_event_cb(lv_tabview_get_content(tabview), scroll_begin_event, LV_EVENT_
↪SCROLL_BEGIN, NULL);

    lv_obj_set_style_bg_color(tabview, lv_palette_lighten(LV_PALETTE_RED, 2), 0);

    lv_obj_t * tab_btns = lv_tabview_get_tab_btns(tabview);
    lv_obj_set_style_bg_color(tab_btns, lv_palette_darken(LV_PALETTE_GREY, 3), 0);
    lv_obj_set_style_text_color(tab_btns, lv_palette_lighten(LV_PALETTE_GREY, 5), 0);
    lv_obj_set_style_border_side(tab_btns, LV_BORDER_SIDE_RIGHT, LV_PART_ITEMS | LV_
↪STATE_CHECKED);


    /*Add 3 tabs (the tabs are page (lv_page) and can be scrolled*/
    lv_obj_t *tab1 = lv_tabview_add_tab(tabview, "Tab 1");
    lv_obj_t *tab2 = lv_tabview_add_tab(tabview, "Tab 2");
    lv_obj_t *tab3 = lv_tabview_add_tab(tabview, "Tab 3");
    lv_obj_t *tab4 = lv_tabview_add_tab(tabview, "Tab 4");
    lv_obj_t *tab5 = lv_tabview_add_tab(tabview, "Tab 5");

    lv_obj_set_style_bg_color(tab2, lv_palette_lighten(LV_PALETTE_AMBER, 3), 0);
    lv_obj_set_style_bg_opa(tab2, LV_OPA_COVER, 0);

    /*Add content to the tabs*/
    lv_obj_t * label = lv_label_create(tab1);
    lv_label_set_text(label, "First tab");

    label = lv_label_create(tab2);
    lv_label_set_text(label, "Second tab");

    label = lv_label_create(tab3);
    lv_label_set_text(label, "Third tab");

    label = lv_label_create(tab4);
    lv_label_set_text(label, "Forth tab");

    label = lv_label_create(tab5);
    lv_label_set_text(label, "Fifth tab");

    lv_obj_clear_flag(lv_tabview_get_content(tabview), LV_OBJ_FLAG_SCROLLABLE);
}
#endif
```

```
def scroll_begin_event(e):
```

(下页继续)

(续上页)

```
    #Disable the scroll animations. Triggered when a tab button is clicked */
    if e.get_code() == lv.EVENT.SCROLL_BEGIN:
        a = lv.anim_t.__cast__(e.get_param())
        if a:
            a.time = 0

# Create a Tab view object
tabview = lv.tabview(lv.scr_act(), lv.DIR.LEFT, 80)
tabview.get_content().add_event_cb(scroll_begin_event, lv.EVENT.SCROLL_BEGIN, None)

tabview.set_style_bg_color(lv.palette_lighten(lv.PALETTE.RED, 2), 0)

tab_btns = tabview.get_tab_btns()
tab_btns.set_style_bg_color(lv.palette_darken(lv.PALETTE.GREY, 3), 0)
tab_btns.set_style_text_color(lv.palette_lighten(lv.PALETTE.GREY, 5), 0)
tab_btns.set_style_border_side(lv.BORDER_SIDE.RIGHT, lv.PART.ITEMS | lv.STATE.CHECKED)


# Add 3 tabs (the tabs are page (lv_page) and can be scrolled
tab1 = tabview.add_tab("Tab 1")
tab2 = tabview.add_tab("Tab 2")
tab3 = tabview.add_tab("Tab 3")
tab4 = tabview.add_tab("Tab 4")
tab5 = tabview.add_tab("Tab 5")

tab2.set_style_bg_color(lv.palette_lighten(lv.PALETTE.AMBER, 3), 0)
tab2.set_style_bg_opa(lv.OPA.COVER, 0)

# Add content to the tabs
label = lv.label(tab1)
label.set_text("First tab")

label = lv.label(tab2)
label.set_text("Second tab")

label = lv.label(tab3)
label.set_text("Third tab")

label = lv.label(tab4)
label.set_text("Forth tab")

label = lv.label(tab5)
```

(下页继续)

(续上页)

```
label.set_text("Fifth tab")

tabview.get_content().clear_flag(lv.obj.FLAG.SCROLLABLE)
```

Textarea

Simple Text area

```
#include "../../lv_examples.h"
#if LV_USE_TEXTAREA && LV_BUILD_EXAMPLES

static void textarea_event_handler(lv_event_t * e)
{
    lv_obj_t * ta = lv_event_get_target(e);
    LV_LOG_USER("Enter was pressed. The current text is: %s", lv_textarea_get_
↪text(ta));
}

static void btnm_event_handler(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target(e);
    lv_obj_t * ta = lv_event_get_user_data(e);
    const char * txt = lv_btnmatrix_get_btn_text(obj, lv_btnmatrix_get_selected_
↪btn(obj));

    if(strcmp(txt, LV_SYMBOL_BACKSPACE) == 0) lv_textarea_del_char(ta);
    else if(strcmp(txt, LV_SYMBOL_NEW_LINE) == 0) lv_event_send(ta, LV_EVENT_READY,␣
↪NULL);
    else lv_textarea_add_text(ta, txt);

}

void lv_example_textarea_1(void)
{
    lv_obj_t * ta = lv_textarea_create(lv_scr_act());
    lv_textarea_set_one_line(ta, true);
    lv_obj_align(ta, LV_ALIGN_TOP_MID, 0, 10);
    lv_obj_add_event_cb(ta, textarea_event_handler, LV_EVENT_READY, ta);
    lv_obj_add_state(ta, LV_STATE_FOCUSED); /*To be sure the cursor is visible*/

    static const char * btnm_map[] = {"1", "2", "3", "\n",
```

(下页继续)

(续上页)

```
                                "4", "5", "6", "\n",
                                "7", "8", "9", "\n",
                                LV_SYMBOL_BACKSPACE, "0", LV_SYMBOL_NEW_LINE, ""};

    lv_obj_t * btnm = lv_btnmatrix_create(lv_scr_act());
    lv_obj_set_size(btnm, 200, 150);
    lv_obj_align(btnm, LV_ALIGN_BOTTOM_MID, 0, -10);
    lv_obj_add_event_cb(btnm, btnm_event_handler, LV_EVENT_VALUE_CHANGED, ta);
    lv_obj_clear_flag(btnm, LV_OBJ_FLAG_CLICK_FOCUSABLE); /*To keep the text area␣
↪focused on button clicks*/
    lv_btnmatrix_set_map(btnm, btnm_map);
}

#endif
```

```
def textarea_event_handler(e, ta):
    print("Enter was pressed. The current text is: " + ta.get_text())


def btnm_event_handler(e, ta):
    obj = e.get_target()
    txt = obj.get_btn_text(obj.get_selected_btn())
    if txt == lv.SYMBOL.BACKSPACE:
        ta.del_char()
    elif txt == lv.SYMBOL.NEW_LINE:
        lv.event_send(ta, lv.EVENT.READY, None)
    elif txt:
        ta.add_text(txt)


ta = lv.textarea(lv.scr_act())
ta.set_one_line(True)
ta.align(lv.ALIGN.TOP_MID, 0, 10)
ta.add_event_cb(lambda e: textarea_event_handler(e, ta), lv.EVENT.READY, None)
ta.add_state(lv.STATE.FOCUSED)   # To be sure the cursor is visible

btnm_map = ["1", "2", "3", "\n",
            "4", "5", "6", "\n",
            "7", "8", "9", "\n",
            lv.SYMBOL.BACKSPACE, "0", lv.SYMBOL.NEW_LINE, ""]

btnm = lv.btnmatrix(lv.scr_act())
btnm.set_size(200, 150)
```

(下页继续)

(续上页)

```
btnm.align(lv.ALIGN.BOTTOM_MID, 0, -10)
btnm.add_event_cb(lambda e: btnm_event_handler(e, ta), lv.EVENT.VALUE_CHANGED, None)
btnm.clear_flag(lv.obj.FLAG.CLICK_FOCUSABLE)    # To keep the text area focused on␣
↪button clicks
btnm.set_map(btnm_map)
```

Text area with password field

```
#include "../../lv_examples.h"
#if LV_USE_TEXTAREA && LV_USE_KEYBOARD && LV_BUILD_EXAMPLES

static void ta_event_cb(lv_event_t * e);

static lv_obj_t * kb;

void lv_example_textarea_2(void)
{
    /*Create the password box*/
    lv_obj_t * pwd_ta = lv_textarea_create(lv_scr_act());
    lv_textarea_set_text(pwd_ta, "");
    lv_textarea_set_password_mode(pwd_ta, true);
    lv_textarea_set_one_line(pwd_ta, true);
    lv_obj_set_width(pwd_ta, lv_pct(40));
    lv_obj_set_pos(pwd_ta, 5, 20);
    lv_obj_add_event_cb(pwd_ta, ta_event_cb, LV_EVENT_ALL, NULL);

    /*Create a label and position it above the text box*/
    lv_obj_t * pwd_label = lv_label_create(lv_scr_act());
    lv_label_set_text(pwd_label, "Password:");
    lv_obj_align_to(pwd_label, pwd_ta, LV_ALIGN_OUT_TOP_LEFT, 0, 0);

    /*Create the one-line mode text area*/
    lv_obj_t * text_ta = lv_textarea_create(lv_scr_act());
    lv_textarea_set_one_line(text_ta, true);
    lv_textarea_set_password_mode(text_ta, false);
    lv_obj_set_width(text_ta, lv_pct(40));
    lv_obj_add_event_cb(text_ta, ta_event_cb, LV_EVENT_ALL, NULL);
    lv_obj_align(text_ta, LV_ALIGN_TOP_RIGHT, -5, 20);


    /*Create a label and position it above the text box*/
    lv_obj_t * oneline_label = lv_label_create(lv_scr_act());
```

(下页继续)

(续上页)

```
    lv_label_set_text(oneline_label, "Text:");
    lv_obj_align_to(oneline_label, text_ta, LV_ALIGN_OUT_TOP_LEFT, 0, 0);

    /*Create a keyboard*/
    kb = lv_keyboard_create(lv_scr_act());
    lv_obj_set_size(kb,  LV_HOR_RES, LV_VER_RES / 2);

    lv_keyboard_set_textarea(kb, pwd_ta); /*Focus it on one of the text areas to␣
↪start*/
}

static void ta_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * ta = lv_event_get_target(e);
    if(code == LV_EVENT_CLICKED || code == LV_EVENT_FOCUSED) {
        /*Focus on the clicked text area*/
        if(kb != NULL) lv_keyboard_set_textarea(kb, ta);
    }

    else if(code == LV_EVENT_READY) {
        LV_LOG_USER("Ready, current text: %s", lv_textarea_get_text(ta));
    }
}

#endif
```

```
def ta_event_cb(e):
    code = e.get_code()
    ta = e.get_target()
    if code == lv.EVENT.CLICKED or code == lv.EVENT.FOCUSED:
        # Focus on the clicked text area
        if kb != None:
            kb.set_textarea(ta)

    elif code == lv.EVENT.READY:
        print("Ready, current text: " + ta.get_text())


# Create the password box
LV_HOR_RES = lv.scr_act().get_disp().driver.hor_res
LV_VER_RES = lv.scr_act().get_disp().driver.ver_res
```

(下页继续)

(续上页)

```
pwd_ta = lv.textarea(lv.scr_act())
pwd_ta.set_text("")
pwd_ta.set_password_mode(True)
pwd_ta.set_one_line(True)
pwd_ta.set_width(LV_HOR_RES // 2 - 20)
pwd_ta.set_pos(5, 20)
pwd_ta.add_event_cb(ta_event_cb, lv.EVENT.ALL, None)

# Create a label and position it above the text box
pwd_label = lv.label(lv.scr_act())
pwd_label.set_text("Password:")
pwd_label.align_to(pwd_ta, lv.ALIGN.OUT_TOP_LEFT, 0, 0)

# Create the one-line mode text area
text_ta = lv.textarea(lv.scr_act())
text_ta.set_width(LV_HOR_RES // 2 - 20)
text_ta.set_one_line(True)
text_ta.add_event_cb(ta_event_cb, lv.EVENT.ALL, None)
text_ta.set_password_mode(False)

text_ta.align(lv.ALIGN.TOP_RIGHT, -5, 20)

# Create a label and position it above the text box
oneline_label = lv.label(lv.scr_act())
oneline_label.set_text("Text:")
oneline_label.align_to(text_ta, lv.ALIGN.OUT_TOP_LEFT, 0, 0)

# Create a keyboard
kb = lv.keyboard(lv.scr_act())
kb.set_size(LV_HOR_RES, LV_VER_RES // 2)

kb.set_textarea(pwd_ta)  # Focus it on one of the text areas to start
```

Text auto-formatting

```
#include "../../lv_examples.h"
#if LV_USE_TEXTAREA && LV_USE_KEYBOARD && LV_BUILD_EXAMPLES

static void ta_event_cb(lv_event_t * e);

static lv_obj_t * kb;
```

(下页继续)

(续上页)

```
/**
 * Automatically format text like a clock. E.g. "12:34"
 * Add the ':' automatically.
 */
void lv_example_textarea_3(void)
{
    /*Create the text area*/
    lv_obj_t * ta = lv_textarea_create(lv_scr_act());
    lv_obj_add_event_cb(ta, ta_event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    lv_textarea_set_accepted_chars(ta, "0123456789:");
    lv_textarea_set_max_length(ta, 5);
    lv_textarea_set_one_line(ta, true);
    lv_textarea_set_text(ta, "");

    /*Create a keyboard*/
    kb = lv_keyboard_create(lv_scr_act());
    lv_obj_set_size(kb,  LV_HOR_RES, LV_VER_RES / 2);
    lv_keyboard_set_mode(kb, LV_KEYBOARD_MODE_NUMBER);
    lv_keyboard_set_textarea(kb, ta);
}

static void ta_event_cb(lv_event_t * e)
{
    lv_obj_t * ta = lv_event_get_target(e);
    const char * txt = lv_textarea_get_text(ta);
    if(txt[0] >= '0' && txt[0] <= '9' &&
        txt[1] >= '0' && txt[1] <= '9' &&
        txt[2] != ':')
    {
        lv_textarea_set_cursor_pos(ta, 2);
        lv_textarea_add_char(ta, ':');
    }
}

#endif
```

```
def ta_event_cb(e):
    ta = e.get_target()
    txt = ta.get_text()
    # print(txt)
    pos = ta.get_cursor_pos()
    # print("cursor pos: ",pos)
```

(下页继续)

(续上页)

```
    # find position of ":" in text
    colon_pos= txt.find(":")
    # if there are more than 2 digits before the colon, remove the last one entered
    if colon_pos == 3:
        ta.del_char()
    if colon_pos != -1:
        # if there are more than 3 digits after the ":" remove the last one entered
        rest = txt[colon_pos:]
        if len(rest) > 3:
            ta.del_char()

    if len(txt) < 2:
        return
    if ":" in txt:
        return
    if  txt[0] >= '0' and txt[0] <= '9' and \
        txt[1] >= '0' and txt[1] <= '9':
        if len(txt) == 2 or txt[2] != ':' :
            ta.set_cursor_pos(2)
            ta.add_char(ord(':'))
#
# Automatically format text like a clock. E.g. "12:34"
# Add the ':' automatically
#
# Create the text area

LV_HOR_RES = lv.scr_act().get_disp().driver.hor_res
LV_VER_RES = lv.scr_act().get_disp().driver.ver_res

ta = lv.textarea(lv.scr_act())
ta.add_event_cb(ta_event_cb, lv.EVENT.VALUE_CHANGED, None)
ta.set_accepted_chars("0123456789:")
ta.set_max_length(5)
ta.set_one_line(True)
ta.set_text("")
ta.add_state(lv.STATE.FOCUSED)

# Create a keyboard
kb = lv.keyboard(lv.scr_act())
kb.set_size(LV_HOR_RES, LV_VER_RES // 2)
kb.set_mode(lv.keyboard.MODE.NUMBER)
kb.set_textarea(ta)
```

Tabview

Tileview with content

```
#include "../../lv_examples.h"
#if LV_USE_TILEVIEW && LV_BUILD_EXAMPLES

/**
 * Create a 2x2 tile view and allow scrolling only in an "L" shape.
 * Demonstrate scroll chaining with a long list that
 * scrolls the tile view when it can't be scrolled further.
 */
void lv_example_tileview_1(void)
{
    lv_obj_t *tv = lv_tileview_create(lv_scr_act());

    /*Tile1: just a label*/
    lv_obj_t * tile1 = lv_tileview_add_tile(tv, 0, 0, LV_DIR_BOTTOM);
    lv_obj_t * label = lv_label_create(tile1);
    lv_label_set_text(label, "Scroll down");
    lv_obj_center(label);


    /*Tile2: a button*/
    lv_obj_t * tile2 = lv_tileview_add_tile(tv, 0, 1, LV_DIR_TOP | LV_DIR_RIGHT);

    lv_obj_t * btn = lv_btn_create(tile2);

    label = lv_label_create(btn);
    lv_label_set_text(label, "Scroll up or right");

    lv_obj_set_size(btn, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
    lv_obj_center(btn);

    /*Tile3: a list*/
    lv_obj_t * tile3 = lv_tileview_add_tile(tv, 1, 1, LV_DIR_LEFT);
    lv_obj_t * list = lv_list_create(tile3);
    lv_obj_set_size(list, LV_PCT(100), LV_PCT(100));

    lv_list_add_btn(list, NULL, "One");
    lv_list_add_btn(list, NULL, "Two");
    lv_list_add_btn(list, NULL, "Three");
    lv_list_add_btn(list, NULL, "Four");
    lv_list_add_btn(list, NULL, "Five");
```

(下页继续)

(续上页)

```
    lv_list_add_btn(list, NULL, "Six");
    lv_list_add_btn(list, NULL, "Seven");
    lv_list_add_btn(list, NULL, "Eight");
    lv_list_add_btn(list, NULL, "Nine");
    lv_list_add_btn(list, NULL, "Ten");

}

#endif
```

```
#
# Create a 2x2 tile view and allow scrolling only in an "L" shape.
# Demonstrate scroll chaining with a long list that
# scrolls the tile view when it can't be scrolled further.
#
tv = lv.tileview(lv.scr_act())

# Tile1: just a label
tile1 = tv.add_tile(0, 0, lv.DIR.BOTTOM)
label = lv.label(tile1)
label.set_text("Scroll down")
label.center()

# Tile2: a button
tile2 = tv.add_tile(0, 1, lv.DIR.TOP | lv.DIR.RIGHT)

btn = lv.btn(tile2)

label = lv.label(btn)
label.set_text("Scroll up or right")

btn.set_size(lv.SIZE.CONTENT, lv.SIZE.CONTENT)
btn.center()

# Tile3: a list
tile3 = tv.add_tile(1, 1, lv.DIR.LEFT)
list = lv.list(tile3)
list.set_size(lv.pct(100), lv.pct(100))

list.add_btn(None, "One")
list.add_btn(None, "Two")
list.add_btn(None, "Three")
list.add_btn(None, "Four")
```

(下页继续)

(续上页)

```
list.add_btn(None, "Five")
list.add_btn(None, "Six")
list.add_btn(None, "Seven")
list.add_btn(None, "Eight")
list.add_btn(None, "Nine")
list.add_btn(None, "Ten")
```

Window

Simple window

```
#include "../../lv_examples.h"
#if LV_USE_WIN && LV_BUILD_EXAMPLES


static void event_handler(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target(e);
    LV_LOG_USER("Button %d clicked", (int)lv_obj_get_index(obj));
}

void lv_example_win_1(void)
{
    lv_obj_t * win = lv_win_create(lv_scr_act(), 40);
    lv_obj_t * btn;
    btn = lv_win_add_btn(win, LV_SYMBOL_LEFT, 40);
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);

    lv_win_add_title(win, "A title");

    btn = lv_win_add_btn(win, LV_SYMBOL_RIGHT, 40);
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);

    btn = lv_win_add_btn(win, LV_SYMBOL_CLOSE, 60);
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);

    lv_obj_t * cont = lv_win_get_content(win);  /*Content can be added here*/
    lv_obj_t * label = lv_label_create(cont);
    lv_label_set_text(label, "This is\n"
                             "a pretty\n"
                             "long text\n"
                             "to see how\n"
```

(下页继续)

(续上页)

```
                                "the window\n"
                                "becomes\n"
                                "scrollable.\n"
                                "\n"
                                "\n"
                                "Some more\n"
                                "text to be\n"
                                "sure it\n"
                                "overflows. :)");


}

#endif
```

```
def event_handler(e):
    code = e.get_code()
    obj = e.get_target()
    if code == lv.EVENT.CLICKED:
        print("Button {:d} clicked".format(obj.get_child_id()))


win = lv.win(lv.scr_act(), 60)
btn1 = win.add_btn(lv.SYMBOL.LEFT, 40)
btn1.add_event_cb(event_handler, lv.EVENT.ALL, None)
win.add_title("A title")
btn2=win.add_btn(lv.SYMBOL.RIGHT, 40)
btn2.add_event_cb(event_handler, lv.EVENT.ALL, None)
btn3 = win.add_btn(lv.SYMBOL.CLOSE, 60)
btn3.add_event_cb(event_handler, lv.EVENT.ALL, None)

cont = win.get_content()  # Content can be added here
label = lv.label(cont)
label.set_text("""This is
a pretty
long text
to see how
the window
becomes
scrollable.


We need
```

(下页继续)

(续上页)

```
quite some text
and we will
even put
some more
text to be
sure it
overflows.
""")
```

2.3 Get started（开始）

There are several ways to get your feet wet with LVGL. Here is one recommended order of documents to read and things to play with when you are learning to use LVGL:

1. Check the Online demos to see LVGL in action (3 minutes)
2. Read the Introduction page of the documentation (5 minutes)
3. Read the Quick overview page of the documentation (15 minutes)
4. Set up a Simulator (10 minutes)
5. Try out some Examples
6. Port LVGL to a board. See the Porting guide or check the ready to use Projects
7. Read the Overview page to get a better understanding of the library. (2-3 hours)
8. Check the documentation of the Widgets to see their features and usage
9. If you have questions got to the Forum
10. Read the Contributing guide to see how you can help to improve LVGL (15 minutes)

可以按照如下顺序来学习 LVGL：

1. 使用网页在线例程来体验 LVGL（3 分钟）
2. 阅读文档简介章节来初步了解 LVGL（5 分钟）
3. 再来阅读一下文档快速快速概览章节来了解 LVGL 的基本知识（15 分钟）
4. 学习如何使用模拟器来在电脑上仿真 LVGL（10 分钟）
5. 试着动手实践一些例程
6. 参考移植指南尝试将 LVGL 移植到一块开发板上，LVGL 也已经提供了一些移植好的工程
7. 仔细阅读文档总览章节来更加深入的了解和熟悉 LVGL（2-3 小时）
8. 浏览文档组件 (Widgets)章节来了解如何使用它们

9. 如果你有问题可以到 LVGL论坛提问
10. 阅读文档如何向社区贡献章节来看看你能帮 LVGL 社区做些什么，以促进 LVGL 软件质量的不断提高（15 分钟）

2.3.1 Quick overview （快速概览）

Here you can learn the most important things about LVGL. You should read this first to get a general impression and read the detailed *Porting* and *Overview* sections after that.

在这里您可以了解有关 LVGL 的最重要的事情。您应该先阅读本文以获得大致印象，然后再阅读详细的移植和概述 部分。

Get started in a simulator（从模拟器开始）

Instead of porting LVGL to embedded hardware straight away, it's highly recommended to get started in a simulator first.

强烈建议您先在 lvgl 模拟器上开始学习实验，而不是立即将 LVGL 移植到嵌入式硬件。

LVGL is ported to many IDEs to be sure you will find your favorite one. Go to the *Simulators* section to get ready-to-use projects that can be run on your PC. This way you can save the time of porting for now and get some experience with LVGL immediately.

LVGL 已适配到许多 IDE，以确保您能找到自己喜欢的一种模拟器开发环境。转到模拟器 部分以获取可以在您的 PC 上运行的即用型项目。通过这种方式，您可以暂时节省移植时间并立即获得一些使用 LVGL 的经验。(这是非常有用的！)

Add LVGL into your project（将 LVGL 添加到您的项目中）

If you would rather try LVGL on your own project follow these steps:

如果您更愿意在自己的项目中尝试 LVGL，请按照以下步骤操作：

- Download or clone the library from GitHub with `git clone https://github.com/lvgl/lvgl.git`.
- Copy the `lvgl` folder into your project.
- Copy `lvgl/lv_conf_template.h` as `lv_conf.h` next to the `lvgl` folder, change the first `#if 0` to `1` to enable the file's content and set the `LV_COLOR_DEPTH` defines.
- Include `lvgl/lvgl.h` in files where you need to use LVGL related functions.
- Call `lv_tick_inc(x)` every `x` milliseconds in a Timer or Task (`x` should be between 1 and 10). It is required for the internal timing of LVGL. Alternatively, configure `LV_TICK_CUSTOM` (see `lv_conf.h`) so that LVGL can retrieve the current time directly.
- Call `lv_init()`

- Create a draw buffer: LVGL will render the graphics here first, and send the rendered image to the display. The buffer size can be set freely but 1/10 screen size is a good starting point.
- 使用 git 命令 `git clone https://github.com/lvgl/lvgl.git` 从 GitHub 下载或克隆库。
- 将 `lvgl` 文件夹复制到您的项目中。
- 将 `lvgl/lv_conf_template.h` 作为 `lv_conf.h` 复制到 `lvgl` 文件夹旁边，将其第一个的 `#if 0` 更改为 `1` 以使能文件的内容并修改设置 `LV_COLOR_DEPTH` 宏。
- 在需要使用 LVGL 相关函数的文件中包含 `lvgl/lvgl.h` 。
- 在计时器或任务中每 `x` 毫秒调用一次 `lv_tick_inc(x)`（`x` 应该在 1 到 10 之间）。LVGL 的内部时序需要它。或者，配置 `LV_TICK_CUSTOM`（参见 `lv_conf.h` ），以便 LVGL 可以直接检索当前时间。
- 调用 `lv_init()` (初始化 lvgl 库)
- 创建一个绘制缓冲区：LVGL 将首先在此处渲染图形，并将渲染的图像发送到显示器。缓冲区大小可以自由设置，但 1/10 屏幕大小是一个很好的起点。

```
static lv_disp_draw_buf_t draw_buf;
static lv_color_t buf1[DISP_HOR_RES * DISP_VER_RES / 10];                        /
↪*Declare a buffer for 1/10 screen size*/
lv_disp_draw_buf_init(&draw_buf, buf1, NULL, MY_DISP_HOR_RES * MY_DISP_VER_SER / 10);␣
↪ /*Initialize the display buffer.*/
```

- Implement and register a function which can copy the rendered image to an area of your display:

实现并注册一个函数，该函数可以将渲染图像复制到显示区域：

```
lv_disp_drv_t disp_drv;               /*Descriptor of a display driver*/
lv_disp_drv_init(&disp_drv);          /*Basic initialization*/
disp_drv.flush_cb = my_disp_flush;    /*Set your driver function*/
disp_drv.buffer = &draw_buf;          /*Assign the buffer to the display*/
disp_drv.hor_res = MY_DISP_HOR_RES;   /*Set the horizontal resolution of the display*/
disp_drv.hor_res = MY_DISP_VER_RES;   /*Set the verizontal resolution of the display*/
lv_disp_drv_register(&disp_drv);      /*Finally register the driver*/

void my_disp_flush(lv_disp_drv_t * disp, const lv_area_t * area, lv_color_t * color_p)
{
    int32_t x, y;
    /*It's a very slow but simple implementation.
     *`set_pixel` needs to be written by you to a set pixel on the screen*/
    for(y = area->y1; y <= area->y2; y++) {
        for(x = area->x1; x <= area->x2; x++) {
            set_pixel(x, y, *color_p);
            color_p++;
        }
```

(下页继续)

(续上页)

```
    }

    lv_disp_flush_ready(disp);         /* Indicate you are ready with the flushing*/
}
```

- Implement and register a function which can read an input device. E.g. for a touch pad:
- 实现并注册一个可以读取输入设备的函数。例如。对于触摸板：

```
lv_indev_drv_t indev_drv;                   /*Descriptor of a input device driver*/
lv_indev_drv_init(&indev_drv);              /*Basic initialization*/
indev_drv.type = LV_INDEV_TYPE_POINTER;     /*Touch pad is a pointer-like device*/
indev_drv.read_cb = my_touchpad_read;       /*Set your driver function*/
lv_indev_drv_register(&indev_drv);          /*Finally register the driver*/

bool my_touchpad_read(lv_indev_t * indev, lv_indev_data_t * data)
{
    /*`touchpad_is_pressed` and `touchpad_get_xy` needs to be implemented by you*/
    if(touchpad_is_pressed()) {
      data->state = LV_INDEV_STATE_PRESSED;
      touchpad_get_xy(&data->point.x, &data->point.y);
    } else {
      data->state = LV_INDEV_STATE_RELEASED;
    }

}
```

- Call `lv_timer_handler()` periodically every few milliseconds in the main `while(1)` loop or in an operating system task. It will redraw the screen if required, handle input devices, animation etc.
- 在主 `while(1)` 循环或操作系统任务中每隔几毫秒定期调用 `lv_timer_handler()`。如果需要，它将重绘屏幕，处理输入设备，动画等。

For a more detailed guide go to the *Porting* section.

有关更详细的指南，请转到移植 部分。

Learn the basics（学习基础知识）

Widgets（部件）

The graphical elements like Buttons, Labels, Sliders, Charts etc. are called objects or widgets. Go to *Widgets* to see the full list of available widgets.

按钮、标签、滑块、图表等图形元素称为对象或小部件。转到部件 以查看可用小部件的完整列表。

Every object has a parent object where it is created. For example if a label is created on a button, the button is the parent of label.

每个对象都有一个创建它的父对象。例如，如果在按钮上创建标签，则该按钮是标签的父级。

The child object moves with the parent and if the parent is deleted the children will be deleted too.

子对象与父对象一起移动，如果删除父对象，子对象也将被删除。

Children can be visible only on their parent. It other words, the parts of the children outside of the parent are clipped.

子项只能在其父项上可见。换句话说，父级之外的子级部分被剪掉了。

A Screen is the ”root” parent. You can have any number of screens.

Screen 是“根”父级。您可以拥有任意数量的屏幕。

To get the current screen call `lv_scr_act()`, and to load a screen use `lv_scr_load(scr1)`.

要获取当前屏幕调用 `lv_scr_act()`，并使用 `lv_scr_load(scr1)` 加载屏幕。

You can create a new object with `lv_<type>_create(parent)`. It will return an `lv_obj_t *` variable that can be used as a reference to the object to set its parameters.

您可以使用 `lv_<type>_create(parent)` 创建一个新对象。它将返回一个 `lv_obj_t *` 变量，该变量可用作对象的引用以设置其参数。

For example（例如）:

```
lv_obj_t * slider1 = lv_slider_create(lv_scr_act());
```

To set some basic attributes `lv_obj_set_<parameter_name>(obj, <value>)` functions can be used. For example:

要设置一些基本属性，可以使用 `lv_obj_set_<parameter_name>(obj, <value>)` 函数。例如:

```
lv_obj_set_x(btn1, 30);
lv_obj_set_y(btn1, 10);
lv_obj_set_size(btn1, 200, 50);
```

The widgets have type specific parameters too which can be set by `lv_<widget_type>_set_<parameter_name>(obj, <value>)` functions. For example:

这些小部件也具有类型特定的参数，可以通过 `lv_<widget_type>_set_<parameter_name>(obj, <value>)` 函数设置。例如:

```
lv_slider_set_value(slider1, 70, LV_ANIM_ON);
```

To see the full API visit the documentation of the widgets or the related header file (e.g. lvgl/src/widgets/lv_slider.h).

要查看完整的 API，请访问小部件的文档或相关的头文件（例如 lvgl/src/widgets/lv_slider.h）。

Events（事件）

Events are used to inform the user that something has happened with an object. You can assign one or more callbacks to an object which will be called if the object is clicked, released, dragged, being deleted etc.

A callback is assigned like this:

事件用于通知用户某个对象发生了某些事情。您可以将一个或多个回调分配给一个对象，如果该对象被单击、释放、拖动、删除等将被调用。

一个回调是这样分配的：

```
lv_obj_add_event_cb(btn, btn_event_cb, LV_EVENT_CLICKED, NULL); /*Assign a callback␣
↪to the button*/

...

void btn_event_cb(lv_event_t * e)
{
    printf("Clicked\n");
}
```

Instead of `LV_EVENT_CLICKED` `LV_EVENT_ALL` can be used too to call the callback for any event.

From `lv_event_t * e` the current event code can be get with

代替 `LV_EVENT_CLICKED`` `LV_EVENT_ALL` 也可用于调用任何事件的回调。

从 `lv_event_t * e` 可以得到当前的事件代码

```
lv_event_code_t code = lv_event_get_code(e);
```

The object that triggered the event can be retrieved with

触发事件的对象可以用

```
lv_obj_t * obj = lv_event_get_target(e);
```

To learn all features of the events go to the *Event overview* section.

要了解事件的所有功能，请转到事件概述 部分。

Parts（部分）

Widgets might be built from one or more *parts*. For example a button has only one part called LV_PART_MAIN. However, a *Slider* has LV_PART_MAIN, LV_PART_INDICATOR and LV_PART_KNOB.

部件可能由一个或多个部分构建。例如，一个按钮只有一个名为 LV_PART_MAIN 的部分。但是，滑块 具有 LV_PART_MAIN 、LV_PART_INDICATOR 和 LV_PART_KNOB 。

By using parts you can apply different styles to different parts. (See below)

通过使用零件，您可以将不同的样式应用于不同的零件。（见下文）

To learn which parts are used by which object read the widgets' documentation.

要了解哪个对象使用了哪些部件，请阅读部件 的文档。

States（状态）

The objects can be in a combination of the following states:

对象可以处于以下状态的组合：

- LV_STATE_DEFAULT Normal, released state
- LV_STATE_CHECKED Toggled or checked state
- LV_STATE_FOCUSED Focused via keypad or encoder or clicked via touchpad/mouse
- LV_STATE_FOCUS_KEY Focused via keypad or encoder but not via touchpad/mouse
- LV_STATE_EDITED Edit by an encoder
- LV_STATE_HOVERED Hovered by mouse (not supported now)
- LV_STATE_PRESSED Being pressed
- LV_STATE_SCROLLED Being scrolled
- LV_STATE_DISABLED Disabled
- LV_STATE_DEFAULT 正常，释放状态
- LV_STATE_CHECKED 切换或选中状态
- LV_STATE_FOCUSED 通过键盘或编码器聚焦或通过触摸板/鼠标点击
- LV_STATE_FOCUS_KEY 通过键盘或编码器聚焦，但不通过触摸板/鼠标聚焦
- LV_STATE_EDITED 由编码器编辑
- LV_STATE_HOVERED 鼠标悬停（现在不支持）
- LV_STATE_PRESSED 被按下
- LV_STATE_SCROLLED 正在滚动

- LV_STATE_DISABLED 禁用

For example, if you press an object it will automatically go to LV_STATE_FOCUSED and LV_STATE_PRESSED state and when you release it, the LV_STATE_PRESSED state will be removed.

例如，如果你按下一个对象，它会自动进入 LV_STATE_FOCUSED 和 LV_STATE_PRESSED 状态，当你释放它时，LV_STATE_PRESSED 状态将被移除。

To check if an object is in a given state use lv_obj_has_state(obj, LV_STATE_...). It will return true if the object is in that state at that time.

To manually add or remove states use

要检查对象是否处于给定状态，请使用 lv_obj_has_state(obj, LV_STATE_...)。如果对象当时处于该状态，它将返回 true。

要手动添加或删除状态，请使用下面的函数

```
lv_obj_add_state(obj, LV_STATE_...);
lv_obj_clear_state(obj, LV_STATE_...);
```

Styles（样式）

Styles contains properties such as background color, border width, font, etc to describe the appearance of the objects.

样式包含诸如背景颜色、边框宽度、字体等属性来描述对象的外观。

The styles are lv_style_t variables. Only their pointer is saved in the objects so they need to be static or global.

样式是 lv_style_t 变量。只有它们的指针保存在对象中，因此它们需要是静态的或全局的。

Before using a style it needs to be initialized with lv_style_init(&style1). After that properties can be added. For example:

在使用样式之前，它需要使用 lv_style_init(&style1) 进行初始化。之后可以添加属性。例如：

```
static lv_style_t style1;
lv_style_init(&style1);
lv_style_set_bg_color(&style1, lv_color_hex(0xa03080))
lv_style_set_border_width(&style1, 2))
```

See the full list of properties here.

在 这里 查看完整的属性列表。

The styles are assigned to an object's part and state. For example to *"Use this style on the slider's indicator when the slider is pressed"*:

样式被分配给对象的部分和状态。例如 "按下滑块时在滑块指示器上使用此样式"：

```
lv_obj_add_style(slider1, &style1, LV_PART_INDICATOR | LV_STATE_PRESSED);
```

If the *part* is `LV_PART_MAIN` it can be omitted:

如果 *part* 是 `LV_PART_MAIN` 可以省略：

```
lv_obj_add_style(btn1, &style1, LV_STATE_PRESSED); /*Equal to LV_PART_MAIN | LV_STATE_
↪PRESSED*/
```

Similarly, `LV_STATE_DEFAULT` can be omitted too:

类似地，`LV_STATE_DEFAULT` 也可以省略：

```
lv_obj_add_style(slider1, &style1, LV_PART_INDICATOR); /*Equal to LV_PART_INDICATOR |␣
↪LV_STATE_DEFAULT*/
```

For `LV_STATE_DEFAULT` and `LV_PART_MAIN` simply write `0`:

对于 `LV_STATE_DEFAULT` 和 `LV_PART_MAIN` 只需写下 `0`：

```
lv_obj_add_style(btn1, &style1, 0); /*Equal to LV_PART_MAIN | LV_STATE_DEFAULT*/
```

The styles can be cascaded (similarly to CSS). It means you can add more styles to a part of an object. For example `style_btn` can set a default button appearance, and `style_btn_red` can overwrite the background color to make the button red:

样式可以级联（类似于 CSS）。这意味着您可以为对象的一部分添加更多样式。例如 `style_btn` 可以设置默认按钮外观，`style_btn_red` 可以覆盖背景颜色使按钮变为红色：

```
lv_obj_add_style(btn1, &style_btn, 0);
lv_obj_add_style(btn1, &style1_btn_red, 0);
```

If a property is not set on for the current state the style with `LV_STATE_DEFAULT` will be used. If the property is not defined even in the default state a default value is used.

如果没有为当前状态设置属性，则将使用带有“LV_STATE_DEFAULT”的样式。如果即使在默认状态下也未定义该属性，则使用默认值。

Some properties (typically the text-related ones) can be inherited. It means if a property is not set in an object it will be searched in its parents too. For example, you can set the font once in the screen's style and all text on that screen will inherit it by default.

一些属性（通常是与文本相关的）可以被继承。这意味着如果一个属性没有在一个对象中设置，它也会在它的父级中搜索。例如，您可以在屏幕样式中设置一次字体，该屏幕上的所有文本都会默认继承它。

Local style properties also can be added to the objects. It creates a style which resides inside the object and which is used only by the object:

本地样式属性也可以添加到对象中。它创建了一个位于对象内部并且仅由对象使用的样式：

```
lv_obj_set_style_bg_color(slider1, lv_color_hex(0x2080bb), LV_PART_INDICATOR | LV_
↪STATE_PRESSED);
```

To learn all the features of styles see the *Style overview* section.

要了解样式的所有功能，请参阅样式概述 部分。

Themes

Themes are the default styles of the objects. The styles from the themes are applied automatically when the objects are created.

You can select the theme to use in `lv_conf.h`.

主题是对象的默认样式。创建对象时，将自动应用来自主题的样式。

您可以在 `lv_conf.h` 中选择要使用的主题。

Examples

Micropython

Learn more about *Micropython*.

了解有关*Micropython* 的更多信息。

```
# Create a Button and a Label
scr = lv.obj()
btn = lv.btn(scr)
btn.align(lv.scr_act(), lv.ALIGN.CENTER, 0, 0)
label = lv.label(btn)
label.set_text("Button")

# Load the screen
lv.scr_load(scr)
```

2.3.2 Simulator on PC（PC 上的模拟器）

You can try out LVGL **using only your PC** (i.e. without any development boards). LVGL will run on a simulator environment on the PC where anyone can write and experiment with real LVGL applications.

Using the simulator on a PC has the following advantages:

- Hardware independent - Write code, run it on the PC and see the result on a monitor.
- Cross-platform - Any Windows, Linux or macOS system can run the PC simulator.

- Portability - The written code is portable, which means you can simply copy it when migrating to embedded hardware.
- Easy Validation - The simulator is also very useful to report bugs because it provides a common platform for every user. So it's a good idea to reproduce a bug in the simulator and use that code snippet in the Forum.

您可以仅使用您的 PC ** 试用 LVGL（即没有任何开发板）。LVGL 将在 PC 上的模拟器环境中运行，任何人都可以在其中编写和试验真实的 LVGL 应用程序。

在 PC 上使用模拟器具有以下优点：

- 独立于硬件 - 编写代码，在 PC 上运行并在监视器上查看结果。
- 跨平台 - 任何 Windows、Linux 或 macOS 系统都可以运行 PC 模拟器。
- 可移植性 - 编写的代码是可移植的，这意味着您可以在迁移到嵌入式硬件时简单地复制它。
- 轻松验证 - 模拟器对于报告错误也非常有用，因为它为每个用户提供了一个通用平台。因此，最好在模拟器中重现错误并

可以在 论坛 中使用你在 pc 模拟器写的代码片段。

Select an IDE（选择适合的 IDE）

The simulator is ported to various IDEs (Integrated Development Environments). Choose your favorite IDE, read its README on GitHub, download the project, and load it to the IDE.

- Eclipse with SDL driver: Recommended on Linux and Mac
- CodeBlocks: Recommended on Windows
- VisualStudio with SDL driver: For Windows
- VSCode with SDL driver: Recommended on Linux and Mac
- PlatformIO with SDL driver: Recommended on Linux and Mac

模拟器被移植到各种 IDE（集成开发环境）。选择您最喜欢的 IDE，在 GitHub 上阅读其 README，下载项目，然后将其加载到 IDE。

- Eclipse with SDL driver: Linux 和 Mac
- CodeBlocks: Windows （简单方便推荐使用）
- VisualStudio with SDL driver: Windows
- VSCode with SDL driver: Linux 和 Mac
- PlatformIO with SDL driver: Linux 和 Mac

You can use any IDE for development but, for simplicity, the configuration for Eclipse CDT is what we'll focus on in this tutorial. The following section describes the set-up guide of Eclipse CDT in more detail.

Note: If you are on Windows, it's usually better to use the Visual Studio or CodeBlocks projects instead. They work out of the box without requiring extra steps.

您可以使用任何 IDE 进行开发，但为简单起见，Eclipse CDT 的配置是我们在本教程中重点关注的内容。以下部分更详细地描述了 Eclipse CDT 的设置指南。

注意：如果您使用的是 Windows，通常最好改用 Visual Studio 或 CodeBlocks 项目。它们开箱即用，无需额外步骤。

Set-up Eclipse CDT（使用 Eclipse CDT 开发）

Install Eclipse CDT（安装 Eclipse CDT）

Eclipse CDT is a C/C++ IDE.

Eclipse is a Java based software therefore be sure **Java Runtime Environment** is installed on your system.

On Debian-based distros (e.g. Ubuntu): `sudo apt-get install default-jre`

Note: If you are using other distros, then please refer and install 'Java Runtime Environment' suitable to your distro. Note: If you are using macOS and get a "Failed to create the Java Virtual Machine" error, uninstall any other Java JDK installs and install Java JDK 8u. This should fix the problem.

You can download Eclipse's CDT from: https://www.eclipse.org/cdt/downloads.php. Start the installer and choose *Eclipse CDT* from the list.

Eclipse 是基于 Java 的软件，因此请确保您的系统上安装了 **Java 运行时环境**。

在基于 Debian 的发行版（例如 Ubuntu）上：`sudo apt-get install default-jre`

注意：如果您使用其他发行版，请参考并安装适合您的发行版的 "Java 运行时环境"。注意：如果您使用的是 macOS 并收到 "无法创建 Java 虚拟机" 错误，请卸载任何其他 Java JDK 安装并安装 Java JDK 8u。这应该可以解决问题。

您可以从以下位置下载 Eclipse 的 CDT：https://www.eclipse.org/cdt/downloads.php 。启动安装程序并从列表中选择 Eclipse CDT。

Install SDL 2（安装 SDL 2）

The PC simulator uses the SDL 2 cross platform library to simulate a TFT display and a touch pad.

PC 模拟器使用 SDL 2 跨平台库来模拟 TFT 显示器和触摸板。

Linux

On **Linux** you can easily install SDL2 using a terminal:

1. Find the current version of SDL2: `apt-cache search libsdl2 (e.g. libsdl2-2.0-0)`
2. Install SDL2: `sudo apt-get install libsdl2-2.0-0` (replace with the found version)
3. Install SDL2 development package: `sudo apt-get install libsdl2-dev`
4. If build essentials are not installed yet: `sudo apt-get install build-essential`

在 **Linux** 上，您可以使用终端轻松安装 SDL2：

1. 找到 SDL2 的当前版本：`apt-cache search libsdl2 (e.g. libsdl2-2.0-0)`
2. 安装 SDL2：`sudo apt-get install libsdl2-2.0-0`（替换为找到的版本）
3. 安装 SDL2 开发包：`sudo apt-get install libsdl2-dev`
4. 如果尚未安装 build Essentials：`sudo apt-get install build-essential`

Windows

If you are using **Windows** firstly you need to install MinGW (64 bit version). After installing MinGW, do the following steps to add SDL2:

1. Download the development libraries of SDL.Go to https://www.libsdl.org/download-2.0.php and download *Development Libraries: SDL2-devel-2.0.5-mingw.tar.gz*
2. Decompress the file and go to *x86_64-w64-mingw32* directory (for 64 bit MinGW) or to *i686-w64-mingw32* (for 32 bit MinGW)
3. Copy *_...mingw32/include/SDL2* folder to *C:/MinGW/.../x86_64-w64-mingw32/include*
4. Copy *_...mingw32/lib/* content to *C:/MinGW/.../x86_64-w64-mingw32/lib*
5. Copy *_...mingw32/bin/SDL2.dll* to *{eclipse_worksapce}/pc_simulator/Debug/*. Do it later when Eclipse is installed.

Note: If you are using **Microsoft Visual Studio** instead of Eclipse then you don't have to install MinGW.

如果您使用的是 Windows，则需要安装 MinGW（64 位版本）。安装 MinGW 后，执行以下步骤添加 SDL2：

1. 下载 SDL 的开发库。打开 https://www.libsdl.org/download-2.0.php 并下载开发库：SDL2-devel-2.0.5-mingw.tar.gz
2. 解压文件并进入 x86_64-w64-mingw32 目录（对于 64 位 MinGW）或 i686-w64-mingw32（对于 32 位 MinGW）
3. 将 _...mingw32/include/SDL2 文件夹复制到 C:/MinGW/.../x86_64-w64-mingw32/include
4. 将 _...mingw32/lib/ 内容复制到 C:/MinGW/.../x86_64-w64-mingw32/lib

5. 将 _...mingw32/bin/SDL2.dll 复制到 {eclipse_workapce}/pc_simulator/Debug/。稍后在安装 Eclipse 时执行此操作。

注意：如果您使用 **Microsoft Visual Studio** 而不是 Eclipse，那么您不必安装 MinGW。

OSX

On **OSX** you can easily install SDL2 with brew: `brew install sdl2`

If something is not working, then please refer this tutorial to get started with SDL.

在 **OSX** 上，您可以使用 brew 轻松安装 SDL2：`brew install sdl2`

如果出现问题，请参阅 这个教程 以开始使用 SDL。

Pre-configured project（预配置项目）

A pre-configured graphics library project (based on the latest release) is always available to get started easily. You can find the latest one on GitHub. (Please note that, the project is configured for Eclipse CDT).

预配置的图形库项目（基于最新版本）始终可以轻松上手。你可以在 GitHub 仓库 上找到最新的版本。（请注意，该项目是为 Eclipse CDT 配置的）。

Add the pre-configured project to Eclipse CDT（将预先配置的项目添加到 Eclipse CDT）

Run Eclipse CDT. It will show a dialogue about the **workspace path**. Before accepting the path, check that path and copy (and unzip) the downloaded pre-configured project there. After that, you can accept the workspace path. Of course you can modify this path but, in that case copy the project to the corresponding location.

Close the start up window and go to **File->Import** and choose **General->Existing project into Workspace**. **Browse the root directory** of the project and click **Finish**

On **Windows** you have to do two additional things:

- Copy the **SDL2.dll** into the project's Debug folder
- Right click on the project -> Project properties -> C/C++ Build -> Settings -> Libraries -> Add ... and add *mingw32* above SDLmain and SDL. (The order is important: mingw32, SDLmain, SDL)

运行 Eclipse CDT。它将显示有关 **工作区路径**的对话。在接受路径之前，检查该路径并在那里复制（并解压缩）下载的预配置项目。之后，您可以接受工作区路径。当然，您可以修改此路径，但在这种情况下，将项目复制到相应位置。

关闭启动窗口并转到 **File->Import** 并选择 **General->Existing project into Workspace**。**浏览项目根目录**，点击 **完成**

在 **Windows** 上，您必须做另外两件事：

- 将 **SDL2.dll** 复制到项目的 Debug 文件夹中

- 右键单击 项目-> 项目属性-> C/C++ 构建-> 设置-> 库-> 添加... 并在 SDLmain 和 SDL 上方添加 _mingw32_。（顺序很重要：mingw32、SDLmain、SDL）

Compile and Run（编译并运行）

Now you are ready to run LVGL on your PC. Click on the Hammer Icon on the top menu bar to Build the project. If you have done everything right, then you will not get any errors. Note that on some systems additional steps might be required to ”see” SDL 2 from Eclipse but, in most of cases the configurations in the downloaded project is enough.

After a success build, click on the Play button on the top menu bar to run the project. Now a window should appear in the middle of your screen.

Now you are ready to use LVGL and begin development on your PC.

现在您已准备好在您的 PC 上运行 LVGL。单击顶部菜单栏上的锤子图标以构建项目。如果你做的一切都是正确的，那么你就不会出现任何错误。请注意，在某些系统上，从 Eclipse 中“查看”SDL 2 可能需要额外的步骤，但在大多数情况下，下载项目中的配置就足够了。

成功构建后，单击顶部菜单栏上的“播放”按钮以运行项目。现在，屏幕中间应该会出现一个窗口。

现在您已准备好使用 LVGL 并可以开始在您的 PC 上进行开发了！

2.3.3 STM32

TODO

2.3.4 NXP

NXP has integrated LVGL into the MCUXpresso SDK packages for several of their general purpose and crossover microcontrollers, allowing easy evaluation and migration into your product design. Download an SDK for a supported board today and get started with your next GUI application.

Creating new project with LVGL

Downloading the MCU SDK example project is recommended as a starting point. It comes fully configured with LVGL (and with PXP support if module is present), no additional integration work is required.

Adding HW acceleration for NXP iMX RT platforms using PXP (PiXel Pipeline) engine for existing projects

Several drawing features in LVGL can be offloaded to the PXP engine. The CPU is available for other operations while the PXP is running. An RTOS is required to block the LVGL drawing thread and switch to another task or suspend the CPU for power savings.

Features supported:

- RGB565 color format
- Area fill + optional transparency
- BLIT (BLock Image Transfer) + optional transparency
- Color keying + optional transparency
- Recoloring (color tint) + optional transparency
- RTOS integration layer
- Default FreeRTOS and bare metal code provided

Basic configuration:

- Select NXP PXP engine in lv_conf.h: Set `LV_USE_GPU_NXP_PXP` to 1
- Enable default implementation for interrupt handling, PXP start function and automatic initialization: Set `LV_USE_GPU_NXP_PXP_AUTO_INIT` to 1
- If `FSL_RTOS_FREE_RTOS` symbol is defined, FreeRTOS implementation will be used, otherwise bare metal code will be included

Basic initialization:

- If `LV_USE_GPU_NXP_PXP_AUTO_INIT` is enabled, no user code is required; PXP is initialized automatically in `lv_init()`
- For manual PXP initialization, default configuration structure for callbacks can be used. Initialize PXP before calling `lv_init()`

```
#if LV_USE_GPU_NXP_PXP
  #include "lv_gpu/lv_gpu_nxp_pxp.h"
  #include "lv_gpu/lv_gpu_nxp_pxp_osa.h"
#endif
. . .
```

(下页继续)

(续上页)

```
    #if LV_USE_GPU_NXP_PXP
      if (lv_gpu_nxp_pxp_init(&pxp_default_cfg) != LV_RES_OK) {
          PRINTF("PXP init error. STOP.\n");
          for ( ; ; ) ;
      }
    #endif
```

Project setup:

- Add PXP related files to project:
  - lv_gpu/lv_gpu_nxp.c, lv_gpu/lv_gpu_nxp.h: low level drawing calls for LVGL
  - lv_gpu/lv_gpu_nxp_osa.c, lv_gpu/lv_gpu_osa.h: default implementation of OS-specific functions (bare metal and FreeRTOS only)
    - optional, required only if `LV_USE_GPU_NXP_PXP_AUTO_INIT` is set to 1
- PXP related code depends on two drivers provided by MCU SDK. These drivers need to be added to project:
  - fsl_pxp.c, fsl_pxp.h: PXP driver
  - fsl_cache.c, fsl_cache.h: CPU cache handling functions

Advanced configuration:

- Implementation depends on multiple OS-specific functions. The struct `lv_nxp_pxp_cfg_t` with callback pointers is used as a parameter for the `lv_gpu_nxp_pxp_init()` function. Default implementation for FreeRTOS and baremetal is provided in lv_gpu_nxp_osa.c
  - `pxp_interrupt_init()`: Initialize PXP interrupt (HW setup, OS setup)
  - `pxp_interrupt_deinit()`: Deinitialize PXP interrupt (HW setup, OS setup)
  - `pxp_run()`: Start PXP job. Use OS-specific mechanism to block drawing thread. PXP must finish drawing before leaving this function.
- There are configurable area thresholds which are used to decide whether the area will be processed by CPU, or by PXP. Areas smaller than a defined value will be processed by CPU and those bigger than the threshold will be processed by PXP. These thresholds may be defined as preprocessor variables. Default values are defined lv_gpu/lv_gpu_nxp_pxp.h
  - `GPU_NXP_PXP_BLIT_SIZE_LIMIT`: size threshold for image BLIT, BLIT with color keying, and BLIT with recolor (OPA > LV_OPA_MAX)
  - `GPU_NXP_PXP_BLIT_OPA_SIZE_LIMIT`: size threshold for image BLIT and BLIT with color keying with transparency (OPA < LV_OPA_MAX)

- `GPU_NXP_PXP_FILL_SIZE_LIMIT`: size threshold for fill operation (OPA > LV_OPA_MAX)
- `GPU_NXP_PXP_FILL_OPA_SIZE_LIMIT`: size threshold for fill operation with transparency (OPA < LV_OPA_MAX)

2.3.5 Espressif (ESP32 chip series)

LVGL can be used and configured as a standard ESP-IDF component.

More information about ESP-IDF build system can be found here.

LVGL demo project for ESP32

We've created lv_port_esp32, a project using ESP-IDF and LVGL to show one of the demos from lv_demos. You can configure the project to use one of the many supported display controllers and targets (chips).

See lvgl_esp32_drivers repository for a complete list of supported display and indev (touch) controllers and targets.

Using LVGL in your ESP-IDF project

Prerequisites

- ESP-IDF v4.1 and above
- ESP evaluation board with a display

Obtaining LVGL

Option 1: git submodule

Simply clone LVGL into your `project_root/components` directory and it will be automatically integrated into the project. If the project is a git repository you can include LVGL as a git submodule:

```
git submodule add https://github.com/lvgl/lvgl.git components/lvgl
```

The above command will clone LVGL's main repository into the `components/lvgl` directory. LVGL includes a `CMakeLists.txt` file that sets some configuration options so you can use LVGL right away.

Option 2: IDF Component Manager

LVGL is also distributed through IDF Component Manager. It allows users to seamlessly integrate LVGL component into their project with following command:

```
idf.py add-dependency lvgl/lvgl>=8.*
```

During next project build, LVGL component will be fetched from the component registry and added to project build.

Configuration

When you are ready to configure LVGL, launch the configuration menu with `idf.py menuconfig` in your project root directory, go to `Component config` and then `LVGL configuration`.

Using lvgl_esp32_drivers in ESP-IDF project

You can also add `lvgl_esp32_drivers` as a "component". This component should be located inside a directory named "components" in your project root directory.

When your project is a git repository you can include `lvgl_esp32_drivers` as a git submodule:

```
git submodule add https://github.com/lvgl/lvgl_esp32_drivers.git components/lvgl_
↪esp32_drivers
```

2.3.6 Arduino

The LVGL library is directly available as Arduino libraries.

Note that you need to choose a board powerful enough to run LVGL and your GUI. See the requirements of LVGL.

For example ESP32 is a good candidate to create UI's with LVGL.

Get the LVGL Arduino library

LVGL can be installed via the Arduino IDE Library Manager or as a .ZIP library.

You can Download the latest version of LVGL from GitHub and simply copy it to Arduino's library folder.

Set up drivers

To get started it's recommended to use TFT_eSPI library as a TFT driver to simplify testing. To make it work, setup `TFT_eSPI` according to your TFT display type via editing either

- `User_Setup.h`
- or by selecting a configuration in the `User_Setup_Select.h`

Both files are located in `TFT_eSPI` library's folder.

Configure LVGL

LVGL has its own configuration file called `lv_conf.h`. When LVGL is installed, follow these configuration steps:

1. Go to the directory of the installed Arduino libraries
2. Go to `lvgl` and copy `lv_conf_template.h` as `lv_conf.h` into the Arduino Libraries directory next to the `lvgl` library folder.
3. Open `lv_conf.h` and change the first `#if 0` to `#if 1` to enable the content of the file
4. Set the color depth of you display in `LV_COLOR_DEPTH`
5. Set `LV_TICK_CUSTOM 1`

Finally the layout with `lv_conf.h` should look like this:

```
arduino
 |-libraries
   |-lvgl
   |-other_lib_1
   |-other_lib_2
   |-lv_conf.h
```

Initialize and run LVGL

Take a look at LVGL_Arduino.ino to see how to initialize LVGL. `TFT_eSPI` is used as the display driver.

In the INO file you can see how to register a display and a touchpad for LVGL and call an example.

Use the examples and demos

Note that, there is no dedicated INO file for every example. Instead, you can load an example by calling an `lv_example_...` function. For example `lv_example_btn_1()`.

IMPORTANT Due to some the limitations of Arduino's build system you need to copy `lvgl/examples` to `lvgl/src/examples`. Similarly for the demos `lvgl/demos` to `lvgl/src/demos`.

Debugging and logging

LVGL can display debug information in case of trouble. In the `LVGL_Arduino.ino` example there is a `my_print` method, which sends this debug information to the serial interface. To enable this feature you have to edit the `lv_conf.h` file and enable logging in the section `log settings`:

```
/*Log settings*/
#define USE_LV_LOG      1   /*Enable/disable the log module*/
```

(下页继续)

(续上页)

```
#if LV_USE_LOG
/* How important log should be added:
 * LV_LOG_LEVEL_TRACE       A lot of logs to give detailed information
 * LV_LOG_LEVEL_INFO        Log important events
 * LV_LOG_LEVEL_WARN        Log if something unwanted happened but didn't cause a␣
↪problem
 * LV_LOG_LEVEL_ERROR       Only critical issue, when the system may fail
 * LV_LOG_LEVEL_NONE        Do not log anything
 */
#  define LV_LOG_LEVEL    LV_LOG_LEVEL_WARN
```

After enabling the log module and setting LV_LOG_LEVEL accordingly, the output log is sent to the `Serial` port @ 115200 bps.

2.3.7 Micropython

What is Micropython?

Micropython is Python for microcontrollers.Using Micropython, you can write Python3 code and run it even on a bare metal architecture with limited resources.

Highlights of Micropython

- **Compact** - Fits and runs within just 256k of code space and 16k of RAM. No OS is needed, although you can also run it with an OS, if you want.
- **Compatible** - Strives to be as compatible as possible with normal Python (known as CPython).
- **Versatile** - Supports many architectures (x86, x86-64, ARM, ARM Thumb, Xtensa).
- **Interactive** - No need for the compile-flash-boot cycle. With the REPL (interactive prompt) you can type commands and execute them immediately, run scripts, etc.
- **Popular** - Many platforms are supported. The user base is growing bigger. Notable forks: MicroPython, CircuitPython, MicroPython_ESP32_psRAM_LoBo
- **Embedded Oriented** - Comes with modules specifically for embedded systems, such as the machine module for accessing low-level hardware (I/O pins, ADC, UART, SPI, I2C, RTC, Timers etc.)

Why Micropython + LVGL?

Currently, Micropython does not have a good high-level GUI library by default. LVGL is an Object-Oriented Component Based high-level GUI library, which seems to be a natural candidate to map into a higher level language, such as Python. LVGL is implemented in C and its APIs are in C.

Here are some advantages of using LVGL in Micropython:

- Develop GUI in Python, a very popular high level language. Use paradigms such as Object-Oriented Programming.
- Usually, GUI development requires multiple iterations to get things right. With C, each iteration consists of **`Change code`** > **`Build`** > **`Flash`** > **`Run`**.In Micropython it's just **`Change code`** > **`Run`** ! You can even run commands interactively using the REPL (the interactive prompt)

Micropython + LVGL could be used for:

- Fast prototyping GUI.
- Shortening the cycle of changing and fine-tuning the GUI.
- Modelling the GUI in a more abstract way by defining reusable composite objects, taking advantage of Python's language features such as Inheritance, Closures, List Comprehension, Generators, Exception Handling, Arbitrary Precision Integers and others.
- Make LVGL accessible to a larger audience. No need to know C to create a nice GUI on an embedded system.This goes well with CircuitPython vision. CircuitPython was designed with education in mind, to make it easier for new or inexperienced users to get started with embedded development.
- Creating tools to work with LVGL at a higher level (e.g. drag-and-drop designer).

So what does it look like?

> TL;DR: It's very much like the C API, but Object-Oriented for LVGL components.

Let's dive right into an example!

A simple example

```
import lvgl as lv
lv.init()
scr = lv.obj()
btn = lv.btn(scr)
btn.align(lv.scr_act(), lv.ALIGN.CENTER, 0, 0)
label = lv.label(btn)
label.set_text("Button")
lv.scr_load(scr)
```

How can I use it?

Online Simulator

If you want to experiment with LVGL + Micropython without downloading anything - you can use our online simulator!It's a fully functional LVGL + Micropython that runs entirely in the browser and allows you to edit a python script and run it.

Click here to experiment on the online simulator

Hello World

Note: the online simulator is available for lvgl v6 and v7.

PC Simulator

Micropython is ported to many platforms. One notable port is "unix", which allows you to build and run Micropython (+LVGL) on a Linux machine. (On a Windows machine you might need Virtual Box or WSL or MinGW or Cygwin etc.)

Click here to know more information about building and running the unix port

Embedded platform

In the end, the goal is to run it all on an embedded platform.Both Micropython and LVGL can be used on many embedded architectures, such as stm32, ESP32 etc.You would also need display and input drivers. We have some sample drivers (ESP32+ILI9341, as well as some other examples), but chances are you would want to create your own input/display drivers for your specific hardware. Drivers can be implemented either in C as a Micropython module, or in pure Micropython!

Where can I find more information?

- In this Blog Post
- `lv_micropython` README
- `lv_binding_micropython` README
- The LVGL micropython forum (Feel free to ask anything!)
- At Micropython: docs and forum

2.3.8 Tasmota and berry

What is Tasmota?

Tasmota is a widely used open-source firmware for ESP8266 and EPS32 based devices. It supports a wide variety of devices, sensors and integrations to Home Automation and Cloud services. Tasmota firmware is downloaded more than 200,000 times each month, and has an active and growing community.

Tasmota provides access to hundreds of supported devices, full support of MQTT, HTTP(S), integration with major Home Automation systems, myriad of sensors, IR, RF, Zigbee, Bluetooth, AWS IoT, Azure IoT, Alexa and many more.

What is Berry?

Berry is a ultra-lightweight dynamically typed embedded scripting language. It is designed for lower-performance embedded devices. The interpreter of Berry include a one-pass compiler and register-based VM, all the code is written in ANSI C99. Berry offers a syntax very similar to Python, and is inspired from LUA VM. It is fully integrated in Tasmota

Highlights of Berry

Berry has the following advantages:

- Lightweight: A well-optimized interpreter with very little resources. Ideal for use in microprocessors.
- Fast: optimized one-pass bytecode compiler and register-based virtual machine.
- Powerful: supports imperative programming, object-oriented programming, functional programming.
- Flexible: Berry is a dynamic type script, and it's intended for embedding in applications. It can provide good dynamic scalability for the host system.
- Simple: simple and natural syntax, support garbage collection, and easy to use FFI (foreign function interface).
- RAM saving: With compile-time object construction, most of the constant objects are stored in read-only code data segments, so the RAM usage of the interpreter is very low when it starts.

All features are detailed in the Berry Reference Manual

Why LVGL + Tasmota + Berry?

In 2021, Tasmota added full support of LVGL for ESP32 based devices. It also introduced the Berry scripting language, a small-footprint language similar to Python and fully integrated in Tasmota.

A comprehensive mapping of LVGL in Berry language is now available, similar to the mapping of Micropython. It allows to use +98% of all LVGL features. It is also possible to write custom widgets in Berry.

Versions supported: LVGL v8.0.2, LodePNG v20201017, Freetype 2.10.4

Tasmota + Berry + LVGL could be used for:

- Fast prototyping GUI.
- Shortening the cycle of changing and fine-tuning the GUI.
- Modelling the GUI in a more abstract way by defining reusable composite objects, taking advantage of Berry's language features such as Inheritance, Closures, Exception Handling...
- Make LVGL accessible to a larger audience. No need to know C to create a nice GUI on an embedded system.

A higher level interface compatible with OpenHASP is also under development.

So what does it look like?

> TL;DR: Similar to MicroPython, it's very much like the C API, but Object-Oriented for LVGL components.

Let's dive right into an example!

A simple example

```
lv.start()                  # start LVGL
scr = lv.scr_act()          # get default screen
btn = lv.btn(scr)           # create button
btn.center()
label = lv.label(btn)       # create a label in the button
label.set_text("Button")    # set a label to the button
```

How can I use it?

You can start in less than 10 minutes on a M5Stack or equivalent device in less than 10 minutes in this short tutorial

Where can I find more information?

2.3.9 NuttX RTOS

What is NuttX?

NuttX is a mature and secure real-time operating system (RTOS) with an emphasis on technical standards compliance and small size. It is scalable from 8-bit to 64-bit microcontrollers and microprocessors and compliant with the Portable Operating System Interface (POSIX) and the American National Standards Institute (ANSI) standards and with many Linux-like subsystems. The best way to think about NuttX is to think of it as a small Unix/Linux for microcontrollers.

Highlights of NuttX

- **Small** - Fits and runs in microcontrollers as small as 32 kB Flash and 8 kB of RAM.
- **Compliant** - Strives to be as compatible as possible with POSIX and Linux.
- **Versatile** - Supports many architectures (ARM, ARM Thumb, AVR, MIPS, OpenRISC, RISC-V 32-bit and 64-bit, RX65N, x86-64, Xtensa, Z80/Z180, etc.).
- **Modular** - Its modular design allows developers to select only what really matters and use modules to include new features.
- **Popular** - NuttX is used by many companies around the world. Probably you already used a product with NuttX without knowing it was running NuttX.
- **Predictable** - NuttX is a preemptible Realtime kernel, so you can use it to create predictable applications for realtime control.

Why NuttX + LVGL?

Although NuttX has its own graphic library called NX, LVGL is a good alternative because users could find more eye-candy demos and they can reuse code from previous projects. LVGL is an Object-Oriented Component Based high-level GUI library, that could fit very well for a RTOS with advanced features like NuttX. LVGL is implemented in C and its APIs are in C.

Here are some advantages of using LVGL in NuttX

- Develop GUI in Linux first and when it is done just compile it for NuttX. Nothing more, no wasting of time.
- Usually, GUI development for low level RTOS requires multiple iterations to get things right, where each iteration consists of **Change code > Build > Flash > Run**. Using LVGL, Linux and NuttX you can reduce this process and just test everything on your computer and when it is done, compile it on NuttX and that is it.

NuttX + LVGL could be used for

- GUI demos to demonstrate your board graphics capacities.
- Fast prototyping GUI for MVP (Minimum Viable Product) presentation.
- visualize sensor data directly and easily on the board without using a computer.
- Final products with a GUI without a touchscreen (i.e. 3D Printer Interface using Rotary Encoder to Input data).
- Final products with a touchscreen (and all sorts of bells and whistles).

How to get started with NuttX and LVGL?

There are many boards in the NuttX mainline with support for LVGL. Let's use the STM32F429IDISCOVERY as an example because it is a very popular board.

First you need to install the pre-requisites on your system

Let's use the Windows Subsystem for Linux

```
$ sudo apt-get install automake bison build-essential flex gcc-arm-none-eabi gperf␣
↪git libncurses5-dev libtool libusb-dev libusb-1.0.0-dev pkg-config kconfig-
↪frontends openocd
```

Now let's create a workspace to save our files

```
$ mkdir ~/nuttxspace
$ cd ~/nuttxspace
```

Clone the NuttX and Apps repositories:

```
$ git clone https://github.com/apache/incubator-nuttx nuttx
$ git clone https://github.com/apache/incubator-nuttx-apps apps
```

Configure NuttX to use the stm32f429i-disco board and the LVGL Demo

```
$ ./tools/configure.sh stm32f429i-disco:lvgl
$ make
```

If everything went fine you should have now the file `nuttx.bin` to flash on your board:

```
$ ls -l nuttx.bin
-rwxrwxr-x 1 alan alan 287144 Jun 27 09:26 nuttx.bin
```

Flashing the firmware in the board using OpenOCD:

```
$ sudo openocd -f interface/stlink-v2.cfg -f target/stm32f4x.cfg -c init -c "reset␣
↪halt" -c "flash write_image erase nuttx.bin 0x08000000"
```

Reset the board and using the 'NSH>' terminal start the LVGL demo:

```
nsh> lvgldemo
```

Where can I find more information?

- This blog post: LVGL on LPCXpresso54628
- NuttX mailing list: Apache NuttX Mailing List

2.3.10 CMake

LVGL supports integrating with CMake. It comes with preconfigured targets for:

On top of the preconfigured targets you can also use "plain" CMake to integrate LVGL into any custom C/C++ project.

Prerequisites

- CMake (>= 3.12.4)
- Compatible build tool e.g.

Building LVGL with CMake

There are many ways to include external CMake projects into your own. A modern one also used in this example is the CMake FetchContent module. This module conveniently allows us to download dependencies directly at configure time from e.g. GitHub. Here is an example how we might include LVGL into our own project.

```
cmake_minimum_required(VERSION 3.14)
include(FetchContent)

project(MyProject LANGUAGES C CXX)

# Build an executable called "MyFirmware"
add_executable(MyFirmware src/main.c)

# Specify path to own LVGL config header
set(LV_CONF_PATH
    ${CMAKE_CURRENT_SOURCE_DIR}/src/lv_conf.h
    CACHE STRING "" FORCE)

# Fetch LVGL from GitHub
FetchContent_Declare(lvgl URL https://github.com/lvgl/lvgl.git)
FetchContent_MakeAvailable(lvgl)

# The target "MyFirmware" depends on LVGL
target_link_libraries(MyFirmware PRIVATE lvgl::lvgl)
```

This configuration declares a dependency between the two targets **MyFirmware** and **lvgl**. Upon building the target **MyFirmware** this dependency will be resolved and **lvgl** will be built and linked with it. Since LVGL requires a config header called lv_conf.h to be includable by its sources we also set the option `LV_CONF_PATH` to point to our own copy of it.

Additional CMake options

Besides `LV_CONF_PATH` there are two additional CMake options to specify include paths.

`LV_LVGL_H_INCLUDE_SIMPLE` which specifies whether to `#include "lvgl.h"` absolut or relative

`LV_CONF_INCLUDE_SIMPLE` which specifies whether to `#include "lv_conf.h"` and `"lv_drv_conf.h"` absolut or relative

I do not recommend disabling those options unless your folder layout makes it absolutely necessary.

Building LVGL examples with CMake

LVGL examples have their own CMake target. If you want to build the examples simply add them to your dependencies.

```
# The target "MyFirmware" depends on LVGL and examples
target_link_libraries(MyFirmware PRIVATE lvgl::lvgl lvgl::examples)
```

Building LVGL drivers and demos with CMake

Exactly the same goes for the drivers and the demos.

```
# Specify path to own LVGL demos config header
set(LV_DEMO_CONF_PATH
    ${CMAKE_CURRENT_SOURCE_DIR}/src/lv_demo_conf.h
    CACHE STRING "" FORCE)

FetchContent_Declare(lv_drivers
                     GIT_REPOSITORY https://github.com/lvgl/lv_drivers)
FetchContent_MakeAvailable(lv_drivers)
FetchContent_Declare(lv_demos
                     GIT_REPOSITORY https://github.com/lvgl/lv_demos.git)
FetchContent_MakeAvailable(lv_demos)

# The target "MyFirmware" depends on LVGL, drivers and demos
target_link_libraries(MyFirmware PRIVATE lvgl::lvgl lvgl::drivers lvgl::examples)
```

Just like the lv_conf.h header demos comes with its own config header called lv_demo_conf.h. Analogous to `LV_CONF_PATH` its path can be set by using the option `LV_DEMO_CONF_PATH`.

2.4 Porting（移植）

2.4.1 Set-up a project（设置项目）

Get the library（获取 LVGL 图形库）

LVGL is available on GitHub: https://github.com/lvgl/lvgl.

You can clone it or download the latest version of the library from GitHub.

The graphics library itself is the **lvgl** directory which should be copied into your project.

LVGL 可在 GitHub 上获得：https://github.com/lvgl/lvgl。

您可以克隆它或从 GitHub 下载最新版本的库。

图形库本身是 **lvgl** 目录，应将其复制到您的项目中。

Configuration file（修改配置文件）

There is a configuration header file for LVGL called **lv_conf.h**. In this you can set the library's basic behavior, disable unused modules and features, adjust the size of memory buffers in compile-time, etc.

Copy **lvgl/lv_conf_template.h** next to the *lvgl* directory and rename it to *lv_conf.h*. Open the file and change the `#if 0` at the beginning to `#if 1` to enable its content.

lv_conf.h can be copied to another place as well but then you should add `LV_CONF_INCLUDE_SIMPLE` define to your compiler options (e.g. `-DLV_CONF_INCLUDE_SIMPLE` for gcc compiler) and set the include path manually. In this case LVGL will attempt to include `lv_conf.h` simply with `#include "lv_conf.h"`.

In the config file comments explain the meaning of the options. Be sure to set at least `LV_COLOR_DEPTH` according to your display's color depth.

有一个名为 **lv_conf.h** 的 LVGL 配置头文件。在这里，您可以设置库的基本行为、禁用未使用的模块和功能、在编译时调整内存缓冲区的大小等。

复制 *lvgl* 目录旁边的 **lvgl/lv_conf_template.h** 并将其重命名为 *lv_conf.h*。打开文件并将开头的“#if 0”更改为“#if 1”以启用其内容。

lv_conf.h 也可以复制到另一个地方，但是你应该添加 `LV_CONF_INCLUDE_SIMPLE` 定义到你的编译器选项（例如 `-DLV_CONF_INCLUDE_SIMPLE` 用于 gcc 编译器）并手动设置包含路径。在这种情况下，LVGL 将尝试使用 `#include "lv_conf.h"` 简单地包含 `lv_conf.h`。

在配置文件的注释中解释了选项的含义。请务必根据显示器的颜色深度至少设置“LV_COLOR_DEPTH”。

Initialization（初始化）

To use the graphics library you have to initialize it and the other components too. The order of the initialization is:

1. Call `lv_init()`.
2. Initialize your drivers.
3. Register the display and input devices drivers in LVGL. Lear more about *Display* and *Input device* registration.
4. Call `lv_tick_inc(x)` every `x` milliseconds in an interrupt to tell the elapsed time. *Learn more*.
5. Call `lv_timer_handler()` every few milliseconds to handle LVGL related tasks. *Learn more*.

要使用图形库，您必须初始化它和其他组件。初始化的顺序是：

1. 调用 `lv_init()`。
2. 初始化您的驱动程序。
3. 在 LVGL 中注册显示和输入设备驱动程序。详细了解*Display* 和*Input device* 注册。
4. 在中断中每隔 `x` 毫秒调用 `lv_tick_inc(x)` 以告知经过的时间。*了解更多*。
5. 每隔几毫秒调用 `lv_timer_handler()` 来处理 LVGL 相关的任务。*了解更多*。

2.4.2 Display interface（显示接口）

To register a display for LVGL a `lv_disp_draw_buf_t` and a `lv_disp_drv_t` variable have to be initialized.

- `lv_disp_draw_buf_t` contains internal graphic buffer(s) called draw buffer(s).
- `lv_disp_drv_t` contains callback functions to interact with the display and manipulate drawing related things.

要为 LVGL 注册一个显示器，必须初始化一个 `lv_disp_draw_buf_t` 和一个 `lv_disp_drv_t` 变量。

- `lv_disp_draw_buf_t` 包含称为绘制缓冲区的内部图形缓冲区。
- `lv_disp_drv_t` 包含与显示交互和操作绘图相关事物的回调函数。

Draw buffer（绘制缓冲区）

Draw buffer(s) are simple array(s) that LVGL uses to render the content of the screen. Once rendering is ready the content of the draw buffer is sent to the display using the `flush_cb` function set in the display driver (see below).

A draw draw buffer can be initialized via a `lv_disp_draw_buf_t` variable like this:

绘制缓冲区是 LVGL 用来渲染屏幕内容的简单数组。一旦渲染准备就绪，绘制缓冲区的内容将使用显示驱动程序中设置的 `flush_cb` 函数发送到显示器（见下文）。

绘制绘制缓冲区可以通过“lv_disp_draw_buf_t”变量初始化，如下所示：

```
/*A static or global variable to store the buffers*/
static lv_disp_draw_buf_t disp_buf;

/*Static or global buffer(s). The second buffer is optional*/
static lv_color_t buf_1[MY_DISP_HOR_RES * 10];
static lv_color_t buf_2[MY_DISP_HOR_RES * 10];

/*Initialize `disp_buf` with the buffer(s). With only one buffer use NULL instead buf_
↪2 */
lv_disp_draw_buf_init(&disp_buf, buf_1, buf_2, MY_DISP_HOR_RES*10);
```

Note that `lv_disp_draw_buf_t` needs to be static, global or dynamically allocated and not a local variable destroyed if goes out of the scope.

As you can see the draw buffer can be smaller than the screen. In this case, the larger areas will be redrawn in smaller parts that fit into the draw buffer(s). If only a small area changes (e.g. a button is pressed) then only that area will be refreshed.

A larger buffer results in better performance but above 1/10 screen sized buffer(s) there is no significant performance improvement. Therefore it's recommended to choose the size of the draw buffer(s) to at least 1/10 screen sized.

If only **one buffer** is used LVGL draws the content of the screen into that draw buffer and sends it to the display. This way LVGL needs to wait until the content of the buffer is sent to the display before drawing something new in it.

If **two buffers** are used LVGL can draw into one buffer while the content of the other buffer is sent to display in the background. DMA or other hardware should be used to transfer the data to the display to let the MCU draw meanwhile. This way, the rendering and refreshing of the display become parallel.

请注意，`lv_disp_draw_buf_t` 需要是静态的、全局的或动态分配的，而不是超出范围时销毁的局部变量。

如您所见，绘制缓冲区可以小于屏幕。在这种情况下，较大的区域将被重新绘制为适合绘制缓冲区的较小部分。如果只有一个小区域发生变化（例如按下按钮），则只会刷新该区域。

更大的缓冲区会导致更好的性能，但超过 1/10 屏幕大小的缓冲区没有显着的性能改进。因此，建议选择绘制缓冲区的大小至少为屏幕大小的 1/10。

如果只使用一**个缓冲区**，LVGL 将屏幕内容绘制到该绘制缓冲区中并将其发送到显示器。这样 LVGL 需要等到缓冲区的内容发送到显示器，然后再在其中绘制新内容。

如果使用**两个缓冲区**，LVGL 可以绘制到一个缓冲区中，而另一个缓冲区的内容被发送到后台显示。应使用 DMA 或其他硬件将数据传输到显示器，让 MCU 同时绘制。这样，显示的渲染和刷新变得并行。

In the display driver (`lv_disp_drv_t`) the `full_refresh` bit can be enabled to force LVGL to always redraw the whole screen. This works in both *one buffer* and *two buffers* modes.

If `full_refresh` is enabled and 2 screen sized draw buffers are provided, LVGL's display handling works like "traditional" double buffering. This means in `flush_cb` only the address of the frame buffer needs to be changed to the provided pointer (`color_p` parameter). This configuration should be used if the MCU has LCD controller periphery and not with an external display controller (e.g. ILI9341 or SSD1963).

You can measure the performance of different draw buffer configurations using the benchmark example.

在显示驱动程序（`lv_disp_drv_t`）中，可以启用 `full_refresh` 位以强制 LVGL 始终重绘整个屏幕。这适用于 *one buffer* 和 *two buffers* 模式。

如果启用 `full_refresh` 并提供 2 个屏幕大小的绘制缓冲区，LVGL 的显示处理就像“传统”双缓冲一样工作。

这意味着在 `flush_cb` 中只有帧缓冲区的地址需要更改为提供的指针（`color_p` 参数）。如果 MCU 具有 LCD 控制器外围设备而不是外部显示控制器（例如 ILI9341 或 SSD1963），则应使用此配置。

您可以使用 基准示例 测量不同绘制缓冲区配置的性能。

Display driver（显示驱动程序）

Once the buffer initialization is ready a `lv_disp_drv_t` display driver needs to be

1. initialized with `lv_disp_drv_init(&disp_drv)`
2. its fields need to be set
3. it needs to be registered in LVGL with `lv_disp_drv_register(&disp_drv)`

Note that `lv_disp_drv_t` also needs to be static, global or dynamically allocated and not a local variable destroyed if goes out of the scope.

一旦缓冲区初始化准备好，`lv_disp_drv_t` 显示驱动程序需要

1. 用 `lv_disp_drv_init(&disp_drv)` 初始化
2. 它的字段需要设置
3. 需要在 LVGL 中用 `lv_disp_drv_register(&disp_drv)` 注册

请注意，`lv_disp_drv_t` 也需要是静态的、全局的或动态分配的，而不是超出范围时销毁的局部变量。

Mandatory fields（必须要适配的部分）

In the most simple case only the following fields of `lv_disp_drv_t` need to be set:

- `draw_buf` pointer to an initialized `lv_disp_draw_buf_t` variable.
- `hor_res` horizontal resolution of the display in pixels.
- `ver_res` vertical resolution of the display in pixels.
- `flush_cb` a callback function to copy a buffer's content to a specific area of the display. `lv_disp_flush_ready(&disp_drv)` needs to be called when flushing is ready. LVGL might render the screen in multiple chunks and therefore call `flush_cb` multiple times. To see if the current one is the last chunk of rendering use `lv_disp_flush_is_last(&disp_drv)`.

在最简单的情况下，只需要设置 `lv_disp_drv_t` 的以下字段：

- 指向初始化的 `lv_disp_draw_buf_t` 变量的 `draw_buf` 指针。
- `hor_res` 显示器的水平分辨率（以像素为单位）。
- `ver_res` 显示器的垂直分辨率（以像素为单位）。
- `flush_cb` 一个回调函数，用于将缓冲区的内容复制到显示器的特定区域。

`lv_disp_flush_ready(&disp_drv)` 需要在刷新准备好时调用。LVGL 可能会以多个块呈现屏幕，因此多次调用 `flush_cb`。要查看当前是否是渲染的最后一个块，请使用 `lv_disp_flush_is_last(&disp_drv)`。

Optional fields （可选的部分）

There are some optional data fields:

- `color_chroma_key` A color which will be drawn as transparent on chrome keyed images. Set to `LV_COLOR_CHROMA_KEY` by default from `lv_conf.h`.
- `anti_aliasing` use anti-aliasing (edge smoothing). Enabled by default if `LV_COLOR_DEPTH` is set to at least 16 in `lv_conf.h`.
- `rotated` and `sw_rotate` See the *Rotation* section below.
- `screen_transp` if `1` the screen itself can have transparency as well. `LV_COLOR_SCREEN_TRANSP` needs to enabled in `lv_conf.h` and requires `LV_COLOR_DEPTH 32`.
- `user_data` A custom `void` user data for the driver..

有一些可选的数据字段：

- `color_chroma_key` 将在镀铬键控图像上绘制为透明的颜色。从 `lv_conf.h` 默认设置为 `LV_COLOR_CHROMA_KEY`。
- `anti_aliasing` 使用抗锯齿（边缘平滑）。如果在 lv_conf.h 中将 LV_COLOR_DEPTH 设置为至少 16，则默认启用。
- `rotated` 和 `sw_rotate` 请参阅下面的*Rotation* 部分。
- `screen_transp` 如果 `1` 屏幕本身也可以具有透明度。`LV_COLOR_SCREEN_TRANSP` 需要在 `lv_conf.h` 中启用并且需要 `LV_COLOR_DEPTH 32`。
- `user_data` 驱动程序的自定义 `void` 用户数据..

Some other optional callbacks to make easier and more optimal to work with monochrome, grayscale or other non-standard RGB displays:

- `rounder_cb` Round the coordinates of areas to redraw. E.g. a 2x2 px can be converted to 2x8. It can be used if the display controller can refresh only areas with specific height or width (usually 8 px height with monochrome displays).

- `set_px_cb` a custom function to write the draw buffer. It can be used to store the pixels more compactly in the draw buffer if the display has a special color format. (e.g. 1-bit monochrome, 2-bit grayscale etc.) This way the buffers used in `lv_disp_draw_buf_t` can be smaller to hold only the required number of bits for the given area size. Note that, rendering with `set_px_cb` is slower than normal rendering.
- `monitor_cb` A callback function that tells how many pixels were refreshed in how much time. Called when the last chunk is rendered and sent to the display.
- `clean_dcache_cb` A callback for cleaning any caches related to the display.

一些其他可选的回调，使处理单色、灰度或其他非标准 RGB 显示器更容易、更优化：

- `rounder_cb` 四舍五入要重绘的区域的坐标。例如。2x2 px 可以转换为 2x8。如果显示控制器只能刷新具有特定高度或宽度的区域（单色显示器通常为 8 像素高度），则可以使用它。
- `set_px_cb` 一个自定义函数来写入绘制缓冲区。如果显示器具有特殊的颜色格式，它可用于将像素更紧凑地存储在绘图缓冲区中。（例如 1 位单色、2 位灰度等）这样，`lv_disp_draw_buf_t` 中使用的缓冲区可以更小，以仅容纳给定区域大小所需的位数。请注意，使用 `set_px_cb` 渲染比普通渲染慢。
- `monitor_cb` 一个回调函数，告诉我们在多长时间内刷新了多少像素。当最后一个块被渲染并发送到显示器时调用。
- `clean_dcache_cb` 用于清理与显示相关的任何缓存的回调。

LVGL has built-in support to several GPUs (see `lv_conf.h`) but if something else is required these functions can be used to make LVGL use a GPU:

- `gpu_fill_cb` fill an area in the memory with a color.
- `gpu_wait_cb` if any GPU function returns while the GPU is still working, LVGL will use this function when required to make sure GPU rendering is ready.

LVGL 内置了对多个 GPU 的支持（参见 `lv_conf.h`），但如果需要其他功能，这些函数可用于使 LVGL 使用 GPU：

- `gpu_fill_cb` 用颜色填充内存中的一个区域。
- `gpu_wait_cb` 如果在 GPU 仍在工作时任何 GPU 函数返回，LVGL 将在需要时使用此函数以确保 GPU 渲染准备就绪。

Examples（示例）

All together it looks like this:

放在一起看起来像这样：

```
static lv_disp_drv_t disp_drv;          /*A variable to hold the drivers. Must be␣
↪static or global.*/
lv_disp_drv_init(&disp_drv);            /*Basic initialization*/
disp_drv.draw_buf = &disp_buf;          /*Set an initialized buffer*/
```

(下页继续)

(续上页)

```
disp_drv.flush_cb = my_flush_cb;        /*Set a flush callback to draw to the␣
↪display*/
disp_drv.hor_res = 320;                 /*Set the horizontal resolution in pixels*/
disp_drv.ver_res = 240;                 /*Set the vertical resolution in pixels*/

lv_disp_t * disp;
disp = lv_disp_drv_register(&disp_drv); /*Register the driver and save the created␣
↪display objects*/
```

Here are some simple examples of the callbacks:

以下是回调的一些简单示例：

```
void my_flush_cb(lv_disp_drv_t * disp_drv, const lv_area_t * area, lv_color_t * color_
↪p)
{
    /*The most simple case (but also the slowest) to put all pixels to the screen one-
↪by-one
     *`put_px` is just an example, it needs to implemented by you.*/
    int32_t x, y;
    for(y = area->y1; y <= area->y2; y++) {
        for(x = area->x1; x <= area->x2; x++) {
            put_px(x, y, *color_p)
            color_p++;
        }
    }

    /* IMPORTANT!!!
     * Inform the graphics library that you are ready with the flushing*/
    lv_disp_flush_ready(disp_drv);
}

void my_gpu_fill_cb(lv_disp_drv_t * disp_drv, lv_color_t * dest_buf, const lv_area_t␣
↪* dest_area, const lv_area_t * fill_area, lv_color_t color);
{
    /*It's an example code which should be done by your GPU*/
    uint32_t x, y;
    dest_buf += dest_width * fill_area->y1; /*Go to the first line*/

    for(y = fill_area->y1; y < fill_area->y2; y++) {
        for(x = fill_area->x1; x < fill_area->x2; x++) {
            dest_buf[x] = color;
        }
        dest_buf+=dest_width;    /*Go to the next line*/
```

(下页继续)

(续上页)

```
    }
}


void my_rounder_cb(lv_disp_drv_t * disp_drv, lv_area_t * area)
{
  /* Update the areas as needed.
   * For example it makes the area to start only on 8th rows and have Nx8 pixel␣
↪height.*/
   area->y1 = area->y1 & 0x07;
   area->y2 = (area->y2 & 0x07) + 8;
}

void my_set_px_cb(lv_disp_drv_t * disp_drv, uint8_t * buf, lv_coord_t buf_w, lv_coord_
↪t x, lv_coord_t y, lv_color_t color, lv_opa_t opa)
{
   /* Write to the buffer as required for the display.
    * For example it writes only 1-bit for monochrome displays mapped vertically.*/
   buf += buf_w * (y >> 3) + x;
   if(lv_color_brightness(color) > 128) (*buf) |= (1 << (y % 8));
   else (*buf) &= ~(1 << (y % 8));
}

void my_monitor_cb(lv_disp_drv_t * disp_drv, uint32_t time, uint32_t px)
{
  printf("%d px refreshed in %d ms\n", time, ms);
}

void my_clean_dcache_cb(lv_disp_drv_t * disp_drv, uint32)
{
  /* Example for Cortex-M (CMSIS) */
  SCB_CleanInvalidateDCache();
}
```

Rotation（旋转屏幕）

LVGL supports rotation of the display in 90 degree increments. You can select whether you'd like software rotation or hardware rotation.

If you select software rotation (`sw_rotate` flag set to 1), LVGL will perform the rotation for you. Your driver can and should assume that the screen width and height have not changed. Simply flush pixels to the display as normal. Software rotation requires no additional logic in your `flush_cb` callback.

There is a noticeable amount of overhead to performing rotation in software, which is why hardware rotation is also

available. In this mode, LVGL draws into the buffer as though your screen now has the width and height inverted. You are responsible for rotating the provided pixels yourself.

LVGL 支持以 90 度为增量旋转显示器。您可以选择是要软件轮换还是硬件轮换。

如果您选择软件旋转（`sw_rotate` 标志设置为 1），LVGL 将为您执行旋转。您的驱动程序可以并且应该假设屏幕宽度和高度没有改变。只需像往常一样将像素刷新到显示器即可。软件轮换在您的 `flush_cb` 回调中不需要额外的逻辑。

在软件中执行轮换需要大量的开销，这就是硬件轮换也可用的原因。在这种模式下，LVGL 将绘制到缓冲区中，就好像您的屏幕现在具有反转的宽度和高度一样。您有责任自己旋转提供的像素。

The default rotation of your display when it is initialized can be set using the `rotated` flag. The available options are `LV_DISP_ROT_NONE`, `LV_DISP_ROT_90`, `LV_DISP_ROT_180`, or `LV_DISP_ROT_270`. The rotation values are relative to how you would rotate the physical display in the clockwise direction. Thus, `LV_DISP_ROT_90` means you rotate the hardware 90 degrees clockwise, and the display rotates 90 degrees counterclockwise to compensate.

(Note for users upgrading from 7.10.0 and older: these new rotation enum values match up with the old 0/1 system for rotating 90 degrees, so legacy code should continue to work as expected. Software rotation is also disabled by default for compatibility.)

Display rotation can also be changed at runtime using the `lv_disp_set_rotation(disp, rot)` API.

Support for software rotation is a new feature, so there may be some glitches/bugs depending on your configuration. If you encounter a problem please open an issue on GitHub.

初始化时显示器的默认旋转可以使用 `rotated` 标志设置。可用的选项是“LV_DISP_ROT_NONE”、“LV_DISP_ROT_90”、“LV_DISP_ROT_180”或“LV_DISP_ROT_270”。旋转值与顺时针方向旋转物理显示器的方式有关。因此，`LV_DISP_ROT_90` 表示您将硬件顺时针旋转 90 度，显示器逆时针旋转 90 度以进行补偿。

（请注意从 7.10.0 及更早版本升级的用户：这些新的旋转枚举值与旧的 0/1 系统匹配，用于旋转 90 度，因此遗留代码应继续按预期工作。默认情况下，软件旋转也被禁用以实现兼容性.)

也可以在运行时使用 `lv_disp_set_rotation(disp, rot)` API 更改显示旋转。

支持软件轮换是一项新功能，因此根据您的配置可能存在一些故障/错误。如果您遇到问题，请在 GitHub 上打开一个问题。

Further reading（深入学习）

- lv_port_disp_template.c for a template for your own driver.
- *Drawing* to learn more about how rendering works in LVGL.
- *Display features* to learn more about higher level display features.
- lv_port_disp_template.c 用于您自己的驱动程序的模板。
- *Drawing* 了解更多关于渲染在 LVGL 中是如何工作的。

- 显示功能 以了解有关更高级别显示功能的更多信息。

API

@description Display Driver HAL interface header file

Typedefs

typedef struct *_lv_disp_draw_buf_t* **lv_disp_draw_buf_t**
Structure for holding display buffer information.

typedef struct *_lv_disp_drv_t* **lv_disp_drv_t**
Display Driver structure to be registered by HAL. Only its pointer will be saved in `lv_disp_t` so it should be declared as `static lv_disp_drv_t my_drv` or allocated dynamically.

typedef struct *_lv_disp_t* **lv_disp_t**
Display structure.

注解： `lv_disp_drv_t` should be the first member of the structure.

Enums

enum **lv_disp_rot_t**
Values:

enumerator **LV_DISP_ROT_NONE**

enumerator **LV_DISP_ROT_90**

enumerator **LV_DISP_ROT_180**

enumerator **LV_DISP_ROT_270**

Functions

void **lv_disp_drv_init**(*lv_disp_drv_t* *driver)

Initialize a display driver with default values. It is used to have known values in the fields and not junk in memory. After it you can safely set only the fields you need.

参数 **driver** -- pointer to driver variable to initialize

void **lv_disp_draw_buf_init**(*lv_disp_draw_buf_t* *draw_buf, void *buf1, void *buf2, uint32_t size_in_px_cnt)

Initialize a display buffer

参数

- **draw_buf** -- pointer `lv_disp_draw_buf_t` variable to initialize
- **buf1** -- A buffer to be used by LVGL to draw the image. Always has to specified and can't be NULL. Can be an array allocated by the user. E.g. `static lv_color_t disp_buf1[1024 * 10]` Or a memory address e.g. in external SRAM
- **buf2** -- Optionally specify a second buffer to make image rendering and image flushing (sending to the display) parallel. In the `disp_drv->flush` you should use DMA or similar hardware to send the image to the display in the background. It lets LVGL to render next frame into the other buffer while previous is being sent. Set to `NULL` if unused.
- **size_in_px_cnt** -- size of the `buf1` and `buf2` in pixel count.

lv_disp_t ***lv_disp_drv_register**(*lv_disp_drv_t* *driver)

Register an initialized display driver. Automatically set the first display as active.

参数 **driver** -- pointer to an initialized 'lv_disp_drv_t' variable. Only its pointer is saved!

返回 pointer to the new display or NULL on error

void **lv_disp_drv_update**(*lv_disp_t* *disp, *lv_disp_drv_t* *new_drv)

Update the driver in run time.

参数

- **disp** -- pointer to a display. (return value of `lv_disp_drv_register`)
- **new_drv** -- pointer to the new driver

void **lv_disp_remove**(*lv_disp_t* *disp)

Remove a display

参数 **disp** -- pointer to display

void **lv_disp_set_default**(*lv_disp_t* *disp)

Set a default display. The new screens will be created on it by default.

参数 **disp** -- pointer to a display

lv_disp_t ***lv_disp_get_default**(void)

Get the default display

返回 pointer to the default display

lv_coord_t **lv_disp_get_hor_res**(*lv_disp_t* *disp)

Get the horizontal resolution of a display

参数 **disp** -- pointer to a display (NULL to use the default display)

返回 the horizontal resolution of the display

lv_coord_t **lv_disp_get_ver_res**(*lv_disp_t* *disp)

Get the vertical resolution of a display

参数 **disp** -- pointer to a display (NULL to use the default display)

返回 the vertical resolution of the display

lv_coord_t **lv_disp_get_physical_hor_res**(*lv_disp_t* *disp)

Get the full / physical horizontal resolution of a display

参数 **disp** -- pointer to a display (NULL to use the default display)

返回 the full / physical horizontal resolution of the display

lv_coord_t **lv_disp_get_physical_ver_res**(*lv_disp_t* *disp)

Get the full / physical vertical resolution of a display

参数 **disp** -- pointer to a display (NULL to use the default display)

返回 the full / physical vertical resolution of the display

lv_coord_t **lv_disp_get_offset_x**(*lv_disp_t* *disp)

Get the horizontal offset from the full / physical display

参数 **disp** -- pointer to a display (NULL to use the default display)

返回 the horizontal offset from the full / physical display

lv_coord_t **lv_disp_get_offset_y**(*lv_disp_t* *disp)

Get the vertical offset from the full / physical display

参数 **disp** -- pointer to a display (NULL to use the default display)

返回 the horizontal offset from the full / physical display

bool **lv_disp_get_antialiasing**(*lv_disp_t* *disp)

Get if anti-aliasing is enabled for a display or not

参数 **disp** -- pointer to a display (NULL to use the default display)

返回 true: anti-aliasing is enabled; false: disabled

lv_coord_t **lv_disp_get_dpi**(const *lv_disp_t* *disp)

Get the DPI of the display

参数 **disp** -- pointer to a display (NULL to use the default display)

返回 dpi of the display

void **lv_disp_set_rotation**(*lv_disp_t* *disp, *lv_disp_rot_t* rotation)

Set the rotation of this display.

参数

- **disp** -- pointer to a display (NULL to use the default display)
- **rotation** -- rotation angle

lv_disp_rot_t **lv_disp_get_rotation**(*lv_disp_t* *disp)

Get the current rotation of this display.

参数 **disp** -- pointer to a display (NULL to use the default display)

返回 rotation angle

lv_disp_t ***lv_disp_get_next**(*lv_disp_t* *disp)

Get the next display.

参数 **disp** -- pointer to the current display. NULL to initialize.

返回 the next display or NULL if no more. Give the first display when the parameter is NULL

lv_disp_draw_buf_t ***lv_disp_get_draw_buf**(*lv_disp_t* *disp)

Get the internal buffer of a display

参数 **disp** -- pointer to a display

返回 pointer to the internal buffers

void **lv_disp_drv_use_generic_set_px_cb**(*lv_disp_drv_t* *disp_drv, *lv_img_cf_t* cf)

struct **_lv_disp_draw_buf_t**

#include <lv_hal_disp.h> Structure for holding display buffer information.

Public Members

void ***buf1**

First display buffer.

void ***buf2**

Second display buffer.

void ***buf_act**

uint32_t **size**

int **flushing**

int **flushing_last**

uint32_t **last_area**

uint32_t **last_part**

struct **_lv_disp_drv_t**

#include <lv_hal_disp.h> Display Driver structure to be registered by HAL. Only its pointer will be saved in `lv_disp_t` so it should be declared as `static lv_disp_drv_t my_drv` or allocated dynamically.

Public Members

lv_coord_t **hor_res**
Horizontal resolution.

lv_coord_t **ver_res**
Vertical resolution.

lv_coord_t **physical_hor_res**
Horizontal resolution of the full / physical display. Set to -1 for fullscreen mode.

lv_coord_t **physical_ver_res**
Vertical resolution of the full / physical display. Set to -1 for fullscreen mode.

lv_coord_t **offset_x**
Horizontal offset from the full / physical display. Set to 0 for fullscreen mode.

lv_coord_t **offset_y**
Vertical offset from the full / physical display. Set to 0 for fullscreen mode.

lv_disp_draw_buf_t ***draw_buf**
Pointer to a buffer initialized with *`lv_disp_draw_buf_init()`*. LVGL will use this buffer(s) to draw the screens contents

uint32_t **direct_mode**
1: Use screen-sized buffers and draw to absolute coordinates

uint32_t **full_refresh**
1: Always make the whole screen redrawn

uint32_t **sw_rotate**
1: use software rotation (slower)

uint32_t **antialiasing**
1: anti-aliasing is enabled on this display.

uint32_t **rotated**
1: turn the display by 90 degree.

> **警告：** Does not update coordinates for you!

uint32_t **screen_transp**

uint32_t **dpi**
Handle if the screen doesn't have a solid (opa == LV_OPA_COVER) background. Use only if required because it's slower.

void (***flush_cb**)(struct *_lv_disp_drv_t* *disp_drv, const lv_area_t *area, lv_color_t *color_p)
DPI (dot per inch) of the display. Default value is `LV_DPI_DEF`. MANDATORY: Write the internal buffer (draw_buf) to the display. 'lv_disp_flush_ready()' has to be called when finished

void (***rounder_cb**)(struct *_lv_disp_drv_t* *disp_drv, lv_area_t *area)
OPTIONAL: Extend the invalidated areas to match with the display drivers requirements E.g. round `y` to, 8, 16 ..) on a monochrome display

void (***set_px_cb**)(struct *_lv_disp_drv_t* *disp_drv, uint8_t *buf, lv_coord_t buf_w, lv_coord_t x, lv_coord_t y, lv_color_t color, lv_opa_t opa)
OPTIONAL: Set a pixel in a buffer according to the special requirements of the display Can be used for color format not supported in LittelvGL. E.g. 2 bit -> 4 gray scales

---

注解： Much slower then drawing with supported color formats.

---

void (***clear_cb**)(struct *_lv_disp_drv_t* *disp_drv, uint8_t *buf, uint32_t size)

void (***monitor_cb**)(struct *_lv_disp_drv_t* *disp_drv, uint32_t time, uint32_t px)
OPTIONAL: Called after every refresh cycle to tell the rendering and flushing time + the number of flushed pixels

void (***wait_cb**)(struct *_lv_disp_drv_t* *disp_drv)
OPTIONAL: Called periodically while lvgl waits for operation to be completed. For example flushing or GPU User can execute very simple tasks here or yield the task

void (***clean_dcache_cb**)(struct *_lv_disp_drv_t* *disp_drv)

OPTIONAL: Called when lvgl needs any CPU cache that affects rendering to be cleaned

void (*drv_update_cb)(struct _lv_disp_drv_t *disp_drv)
OPTIONAL: called when driver parameters are updated

lv_color_t color_chroma_key
On CHROMA_KEYED images this color will be transparent. LV_COLOR_CHROMA_KEY by default. (lv_conf.h)

lv_draw_ctx_t *draw_ctx

void (*draw_ctx_init)(struct _lv_disp_drv_t *disp_drv, lv_draw_ctx_t *draw_ctx)

void (*draw_ctx_deinit)(struct _lv_disp_drv_t *disp_drv, lv_draw_ctx_t *draw_ctx)

size_t draw_ctx_size

void *user_data
Custom display driver user data

struct _lv_disp_t
#include <lv_hal_disp.h> Display structure.

注解： lv_disp_drv_t should be the first member of the structure.

Public Members

struct _lv_disp_drv_t *driver
< Driver to the display A timer which periodically checks the dirty areas and refreshes them

lv_timer_t *refr_timer
The theme assigned to the screen

struct _lv_theme_t *theme

struct _lv_obj_t **screens
Screens of the display Array of screen objects.

struct _lv_obj_t *act_scr
Currently active screen on this display

struct _lv_obj_t *prev_scr
Previous screen. Used during screen animations

struct *_lv_obj_t* ***scr_to_load**
The screen prepared to load in lv_scr_load_anim

struct *_lv_obj_t* ***top_layer**

See *lv_disp_get_layer_top*

struct *_lv_obj_t* ***sys_layer**

See *lv_disp_get_layer_sys*

uint32_t **screen_cnt**

uint8_t **del_prev**
1: Automatically delete the previous screen when the screen load animation is ready

lv_opa_t **bg_opa**
Opacity of the background color or wallpaper

lv_color_t **bg_color**
Default display color when screens are transparent

const void ***bg_img**
An image source to display as wallpaper

lv_area_t **inv_areas**[LV_INV_BUF_SIZE]
Invalidated (marked to redraw) areas

uint8_t **inv_area_joined**[LV_INV_BUF_SIZE]

uint16_t **inv_p**

uint32_t **last_activity_time**
Last time when there was activity on this display

2.4.3 Input device interface（输入设备接口）

Types of input devices（输入设备的类型）

To register an input device an `lv_indev_drv_t` variable has to be initialized:

要注册输入设备，必须初始化一个 `lv_indev_drv_t` 变量：

```
lv_indev_drv_t indev_drv;
lv_indev_drv_init(&indev_drv);      /*Basic initialization*/
```

(下页继续)

(续上页)

```
indev_drv.type =...                  /*See below.*/
indev_drv.read_cb =...               /*See below.*/
/*Register the driver in LVGL and save the created input device object*/
lv_indev_t * my_indev = lv_indev_drv_register(&indev_drv);
```

`type` can be

- `LV_INDEV_TYPE_POINTER` touchpad or mouse
- `LV_INDEV_TYPE_KEYPAD` keyboard or keypad
- `LV_INDEV_TYPE_ENCODER` encoder with left/right turn and push options
- `LV_INDEV_TYPE_BUTTON` external buttons virtually pressing the screen

`read_cb` is a function pointer which will be called periodically to report the current state of an input device.

Visit *Input devices* to learn more about input devices in general.

`type` 可以是

- `LV_INDEV_TYPE_POINTER` 触摸板或鼠标
- `LV_INDEV_TYPE_KEYPAD` 键盘或小键盘
- `LV_INDEV_TYPE_ENCODER` 编码器，带有左/右转和推动选项
- `LV_INDEV_TYPE_BUTTON` 外部按钮虚拟按下屏幕

`read_cb` 是一个函数指针，它将被定期调用以报告输入设备的当前状态。

访问输入设备 以了解有关输入设备的更多信息。

Touchpad, mouse or any pointer（触摸板、鼠标或任何指针）

Input devices that can click points on the screen belong to this category.

可以点击屏幕上的点的输入设备属于这一类。

```
indev_drv.type = LV_INDEV_TYPE_POINTER;
indev_drv.read_cb = my_input_read;

...

void my_input_read(lv_indev_drv_t * drv, lv_indev_data_t*data)
{
  if(touchpad_pressed) {
    data->point.x = touchpad_x;
    data->point.y = touchpad_y;
    data->state = LV_INDEV_STATE_PRESSED;
```

(下页继续)

(续上页)

```
  } else {
    data->state = LV_INDEV_STATE_RELEASED;
  }
}
```

To set a mouse cursor use `lv_indev_set_cursor(my_indev, &img_cursor)`. (`my_indev` is the return value of `lv_indev_drv_register`)

要设置鼠标光标，请使用 `lv_indev_set_cursor(my_indev, &img_cursor)`。(`my_indev` 是 `lv_indev_drv_register` 的返回值)

Keypad or keyboard（键盘或键盘）

Full keyboards with all the letters or simple keypads with a few navigation buttons belong here.

To use a keyboard/keypad:

- Register a `read_cb` function with `LV_INDEV_TYPE_KEYPAD` type.
- An object group has to be created: `lv_group_t * g = lv_group_create()` and objects have to be added to it with `lv_group_add_obj(g, obj)`
- The created group has to be assigned to an input device: `lv_indev_set_group(my_indev, g)` (`my_indev` is the return value of `lv_indev_drv_register`)
- Use `LV_KEY_...` to navigate among the objects in the group. See `lv_core/lv_group.h` for the available keys.

带有所有字母的全键盘或带有几个导航按钮的简单键盘都属于这里。

要使用键盘/小键盘：

- 注册一个带有 `LV_INDEV_TYPE_KEYPAD` 类型的 `read_cb` 函数。
- 必须创建一个对象组：`lv_group_t * g = lv_group_create()` 并且对象必须使用 `lv_group_add_obj(g, obj)` 添加到其中
- 创建的组必须分配给输入设备：`lv_indev_set_group(my_indev, g)`（`my_indev` 是 `lv_indev_drv_register` 的返回值）
- 使用 `LV_KEY_...` 在组中的对象之间导航。有关可用密钥，请参阅“lv_core/lv_group.h”。

```
indev_drv.type = LV_INDEV_TYPE_KEYPAD;
indev_drv.read_cb = keyboard_read;

...

void keyboard_read(lv_indev_drv_t * drv, lv_indev_data_t*data){
```

(下页继续)

(续上页)

```
  data->key = last_key();            /*Get the last pressed or released key*/

  if(key_pressed()) data->state = LV_INDEV_STATE_PRESSED;
  else data->state = LV_INDEV_STATE_RELEASED;
}
```

Encoder（编码器）

With an encoder you can do 4 things:

1. Press its button
2. Long-press its button
3. Turn left
4. Turn right

In short, the Encoder input devices work like this:

- By turning the encoder you can focus on the next/previous object.
- When you press the encoder on a simple object (like a button), it will be clicked.
- If you press the encoder on a complex object (like a list, message box, etc.) the object will go to edit mode whereby turning the encoder you can navigate inside the object.
- To leave edit mode press long the button.

To use an *Encoder* (similarly to the *Keypads*) the objects should be added to groups.

使用编码器，您可以做 4 件事：

1. 按下它的按钮
2. 长按它的按钮
3. 左转
4. 右转

简而言之，编码器输入设备的工作方式如下：

- 通过转动编码器，您可以专注于下一个/上一个对象。
- 当您在一个简单的对象（如按钮）上按下编码器时，它将被点击。
- 如果您按下复杂对象（如列表、消息框等）上的编码器，该对象将进入编辑模式，从而转动编码器您可以在对象内部导航。
- 要退出编辑模式，请长按按钮。

要使用 *Encoder*（类似于 *Keypads*），应将对象添加到组中。

```
indev_drv.type = LV_INDEV_TYPE_ENCODER;
indev_drv.read_cb = encoder_read;

...

void encoder_read(lv_indev_drv_t * drv, lv_indev_data_t*data){
  data->enc_diff = enc_get_new_moves();

  if(enc_pressed()) data->state = LV_INDEV_STATE_PRESSED;
  else data->state = LV_INDEV_STATE_RELEASED;
}
```

Using buttons with Encoder logic（使用带有编码器逻辑的按钮）

In addition to standard encoder behavior, you can also utilize its logic to navigate(focus) and edit widgets using buttons. This is especially handy if you have only few buttons available, or you want to use other buttons in addition to encoder wheel.

You need to have 3 buttons available:

- `LV_KEY_ENTER` will simulate press or pushing of the encoder button
- `LV_KEY_LEFT` will simulate turning encoder left
- `LV_KEY_RIGHT` will simulate turning encoder right
- other keys will be passed to the focused widget

If you hold the keys it will simulate encoder click with period specified in `indev_drv.long_press_rep_time`.

除了标准编码器行为之外，您还可以利用其逻辑来使用按钮导航（聚焦）和编辑小部件。如果您只有几个按钮可用，或者您想使用除编码轮之外的其他按钮，这将特别方便。

您需要有 3 个按钮可用：

- `LV_KEY_ENTER` 将模拟按下或按下编码器按钮
- `LV_KEY_LEFT` 将模拟向左转动编码器
- `LV_KEY_RIGHT` 将模拟向右旋转编码器
- 其他键将传递给聚焦的小部件

如果您按住这些键，它将模拟编码器点击，并在 `indev_drv.long_press_rep_time` 中指定周期。

```
indev_drv.type = LV_INDEV_TYPE_ENCODER;
indev_drv.read_cb = encoder_with_keys_read;

...
```

(下页继续)

(续上页)

```
void encoder_with_keys_read(lv_indev_drv_t * drv, lv_indev_data_t*data){
  data->key = last_key();            /*Get the last pressed or released key*/
                                     /* use LV_KEY_ENTER for encoder press */
  if(key_pressed()) data->state = LV_INDEV_STATE_PRESSED;
  else {
      data->state = LV_INDEV_STATE_RELEASED;
      /* Optionally you can also use enc_diff, if you have encoder*/
      data->enc_diff = enc_get_new_moves();
  }
}
```

Button（按钮）

Buttons mean external "hardware" buttons next to the screen which are assigned to specific coordinates of the screen. If a button is pressed it will simulate the pressing on the assigned coordinate. (Similarly to a touchpad)

To assign buttons to coordinates use `lv_indev_set_button_points(my_indev, points_array)`.`points_array` should look like `const lv_point_t points_array[] = { {12, 30},{60,90}, ...}`

按钮表示屏幕旁边的外部“硬件”按钮，这些按钮分配给屏幕的特定坐标。如果按下按钮，它将模拟按下指定坐标。（类似于触摸板）

要将按钮分配给坐标，请使用 `lv_indev_set_button_points(my_indev, points_array)`。`points_array` 应该看起来像 `const lv_point_t points_array[] = { {12,30},{60,90}, ...}`

重要： The points_array can't go out of scope. Either declare it as a global variable or as a static variable inside a function.

```
indev_drv.type = LV_INDEV_TYPE_BUTTON;
indev_drv.read_cb = button_read;

...

void button_read(lv_indev_drv_t * drv, lv_indev_data_t*data){
    static uint32_t last_btn = 0;   /*Store the last pressed button*/
    int btn_pr = my_btn_read();     /*Get the ID (0,1,2...) of the pressed button*/
    if(btn_pr >= 0) {               /*Is there a button press? (E.g. -1 indicated no␣
↪button was pressed)*/
       last_btn = btn_pr;           /*Save the ID of the pressed button*/
       data->state = LV_INDEV_STATE_PRESSED;  /*Set the pressed state*/
```

(下页继续)

(续上页)

```
    } else {
        data->state = LV_INDEV_STATE_RELEASED; /*Set the released state*/
    }

    data->btn = last_btn;           /*Save the last button*/
}
```

Other features（其它功能）

Parameters（参数）

The default value of the following parameters can changed in `lv_indev_drv_t`:

- `scroll_limit` Number of pixels to slide before actually scrolling the object.
- `scroll_throw` Scroll throw (momentum) slow-down in [%]. Greater value means faster slow-down.
- `long_press_time` Press time to send `LV_EVENT_LONG_PRESSED` (in milliseconds)
- `long_press_rep_time` Interval of sending `LV_EVENT_LONG_PRESSED_REPEAT` (in milliseconds)
- `read_timer` pointer to the `lv_timer` which reads the input device. Its parameters can be changed by `lv_timer_...()` functions. `LV_INDEV_DEF_READ_PERIOD` in `lv_conf.h` sets the default read period.

以下参数的默认值可以在 `lv_indev_drv_t` 中更改：

- `scroll_limit` 在实际滚动对象之前要滑动的像素数。
- `scroll_throw` 滚动投掷（动量）减慢 [%]。更大的价值意味着更快的减速。
- `long_press_time` 按下发送 `LV_EVENT_LONG_PRESSED` 的时间（以毫秒为单位）
- `long_press_rep_time` 发送 `LV_EVENT_LONG_PRESSED_REPEAT` 的间隔（以毫秒为单位）
- `read_timer` 指向读取输入设备的 `lv_timer` 的指针。它的参数可以通过 `lv_timer_...()` 函数改变。`lv_conf.h` 中的 `LV_INDEV_DEF_READ_PERIOD` 设置默认读取周期。

Feedback（回调处理）

Besides `read_cb` a `feedback_cb` callback can be also specified in `lv_indev_drv_t`. `feedback_cb` is called when any type of event is sent by the input devices (independently from its type). This allows generating feedback for the user, e.g. to play a sound on `LV_EVENT_CLICKED`.

除了 `read_cb` 一个 `feedback_cb` 回调也可以在 `lv_indev_drv_t` 中指定。`feedback_cb` 在输入设备发送任何类型的事件时被调用（与其类型无关）。这允许为用户生成反馈，例如在“LV_EVENT_CLICKED”上播放声音。

Associating with a display（与显示器关联）

Every input device is associated with a display. By default, a new input device is added to the lastly created or the explicitly selected (using `lv_disp_set_default()`) display. The associated display is stored and can be changed in `disp` field of the driver.

每个输入设备都与一个显示器相关联。默认情况下，一个新的输入设备被添加到最后创建的或明确选择的（使用 `lv_disp_set_default()`）显示。相关的显示被存储并可在驱动程序的“disp”字段中更改。

Buffered reading（缓冲读取）

By default LVGL calls `read_cb` periodically. This way there is a chance that some user gestures are missed.

To solve this you can write an event driven driver for your input device that buffers measured data. In `read_cb` you can set the buffered data instead of reading the input device. You can set the `data->continue_reading` flag to tell that LVGL there is more data to read and it should call the `read_cb` again.

默认情况下，LVGL 会定期调用 `read_cb`。这样就有可能错过一些用户手势。

为了解决这个问题，你可以为你的输入设备编写一个事件驱动的驱动程序来缓冲测量数据。在 read_cb 中，您可以设置缓冲数据而不是读取输入设备。你可以设置 `data->continue_reading` 标志来告诉 LVGL 有更多的数据要读取，它应该再次调用 `read_cb`。

Further reading（深入学习）

- lv_port_indev_template.c for a template for your own driver.
- *INdev features* to learn more about higher level input device features.
- lv_port_indev_template.c 用于您自己的驱动程序的模板。
- *INdev features* 以了解有关更高级别输入设备功能的更多信息。

API

@description Input Device HAL interface layer header file

Typedefs

typedef struct *_lv_indev_drv_t* **`lv_indev_drv_t`**
Initialized by the user and registered by 'lv_indev_add()'

typedef struct *_lv_indev_proc_t* **`_lv_indev_proc_t`**
Run time data of input devices Internally used by the library, you should not need to touch it.

typedef struct *_lv_indev_t* **lv_indev_t**
The main input device descriptor with driver, runtime data ('proc') and some additional information

Enums

enum **lv_indev_type_t**
Possible input device types

Values:

enumerator **LV_INDEV_TYPE_NONE**
Uninitialized state

enumerator **LV_INDEV_TYPE_POINTER**
Touch pad, mouse, external button

enumerator **LV_INDEV_TYPE_KEYPAD**
Keypad or keyboard

enumerator **LV_INDEV_TYPE_BUTTON**
External (hardware button) which is assigned to a specific point of the screen

enumerator **LV_INDEV_TYPE_ENCODER**
Encoder with only Left, Right turn and a Button

enum **lv_indev_state_t**
States for input devices

Values:

enumerator **LV_INDEV_STATE_RELEASED**

enumerator **LV_INDEV_STATE_PRESSED**

Functions

void **lv_indev_drv_init**(struct *_lv_indev_drv_t* *driver)
Initialize an input device driver with default values. It is used to surely have known values in the fields and not memory junk. After it you can set the fields.

参数 **driver** -- pointer to driver variable to initialize

lv_indev_t ***lv_indev_drv_register**(struct *_lv_indev_drv_t* *driver)
Register an initialized input device driver.

参数 **driver** -- pointer to an initialized 'lv_indev_drv_t' variable (can be local variable)

返回 pointer to the new input device or NULL on error

void **lv_indev_drv_update**(*lv_indev_t* *indev, struct *_lv_indev_drv_t* *new_drv)

Update the driver in run time.

参数

- **indev** -- pointer to an input device. (return value of `lv_indev_drv_register`)
- **new_drv** -- pointer to the new driver

void **lv_indev_delete**(*lv_indev_t* *indev)

Remove the provided input device. Make sure not to use the provided input device afterwards anymore.

参数 **indev** -- pointer to delete

lv_indev_t ***lv_indev_get_next**(*lv_indev_t* *indev)

Get the next input device.

参数 **indev** -- pointer to the current input device. NULL to initialize.

返回 the next input device or NULL if there are no more. Provide the first input device when the parameter is NULL

void **_lv_indev_read**(*lv_indev_t* *indev, *lv_indev_data_t* *data)

Read data from an input device.

参数

- **indev** -- pointer to an input device
- **data** -- input device will write its data here

struct **lv_indev_data_t**

#include <lv_hal_indev.h> Data structure passed to an input driver to fill

Public Members

lv_point_t **point**

For LV_INDEV_TYPE_POINTER the currently pressed point

uint32_t **key**

For LV_INDEV_TYPE_KEYPAD the currently pressed key

uint32_t **btn_id**

For LV_INDEV_TYPE_BUTTON the currently pressed button

int16_t **enc_diff**
For LV_INDEV_TYPE_ENCODER number of steps since the previous read

lv_indev_state_t **state**
LV_INDEV_STATE_REL or LV_INDEV_STATE_PR

bool **continue_reading**
If set to true, the read callback is invoked again

struct **_lv_indev_drv_t**
#include <lv_hal_indev.h> Initialized by the user and registered by 'lv_indev_add()'

Public Members

lv_indev_type_t **type**
< Input device type Function pointer to read input device data.

void (***read_cb**)(struct *_lv_indev_drv_t* *indev_drv, *lv_indev_data_t* *data)

void (***feedback_cb**)(struct *_lv_indev_drv_t**, uint8_t)
Called when an action happened on the input device. The second parameter is the event from `lv_event_t`

void ***user_data**

struct *_lv_disp_t* ***disp**
< Pointer to the assigned display Timer to periodically read the input device

lv_timer_t ***read_timer**
Number of pixels to slide before actually drag the object

uint8_t **scroll_limit**
Drag throw slow-down in [%]. Greater value means faster slow-down

uint8_t **scroll_throw**
At least this difference should be between two points to evaluate as gesture

uint8_t **gesture_min_velocity**
At least this difference should be to send a gesture

uint8_t **gesture_limit**
Long press time in milliseconds

uint16_t **long_press_time**
Repeated trigger period in long press [ms]

uint16_t **long_press_repeat_time**

struct **_lv_indev_proc_t**
#include <lv_hal_indev.h> Run time data of input devices Internally used by the library, you should not need to touch it.

Public Members

lv_indev_state_t **state**
Current state of the input device.

uint8_t **long_pr_sent**

uint8_t **reset_query**

uint8_t **disabled**

uint8_t **wait_until_release**

lv_point_t **act_point**
Current point of input device.

lv_point_t **last_point**
Last point of input device.

lv_point_t **last_raw_point**
Last point read from read_cb.

lv_point_t **vect**
Difference between `act_point` and `last_point`.

lv_point_t **scroll_sum**

lv_point_t **scroll_throw_vect**

lv_point_t **scroll_throw_vect_ori**

struct *_lv_obj_t* ***act_obj**

struct *_lv_obj_t* ***last_obj**

struct *_lv_obj_t* ***scroll_obj**

struct *_lv_obj_t* ***last_pressed**

lv_area_t **scroll_area**

lv_point_t **gesture_sum**

lv_dir_t **scroll_dir**

lv_dir_t **gesture_dir**

uint8_t **gesture_sent**

struct *_lv_indev_proc_t*::[anonymous]::[anonymous] **pointer**

lv_indev_state_t **last_state**

uint32_t **last_key**

struct *_lv_indev_proc_t*::[anonymous]::[anonymous] **keypad**

union *_lv_indev_proc_t*::[anonymous] **types**

uint32_t **pr_timestamp**
Pressed time stamp

uint32_t **longpr_rep_timestamp**
Long press repeat time stamp

struct **_lv_indev_t**
#include <lv_hal_indev.h> The main input device descriptor with driver, runtime data ('proc') and some additional information

Public Members

struct *_lv_indev_drv_t* ***driver**

_lv_indev_proc_t **proc**

struct *_lv_obj_t* ***cursor**
Cursor for LV_INPUT_TYPE_POINTER

struct *_lv_group_t* ***group**
Keypad destination group

const lv_point_t ***btn_points**
Array points assigned to the button ()screen will be pressed here by the buttons

2.4.4 Tick interface（心跳接口）

LVGL needs a system tick to know elapsed time for animations and other tasks.

You need to call the `lv_tick_inc(tick_period)` function periodically and provide the call period in milliseconds. For example, `lv_tick_inc(1)` when calling every millisecond.

`lv_tick_inc` should be called in a higher priority routine than `lv_task_handler()` (e.g. in an interrupt) to precisely know the elapsed milliseconds even if the execution of `lv_task_handler` takes more time.

With FreeRTOS `lv_tick_inc` can be called in `vApplicationTickHook`.

On Linux based operating system (e.g. on Raspberry Pi) `lv_tick_inc` can be called in a thread like below:

LVGL 需要一个系统滴答来了解动画和其他任务所用的时间。

您需要定期调用 `lv_tick_inc(tick_period)` 函数并提供以毫秒为单位的调用周期。例如，`lv_tick_inc(1)` 每毫秒调用一次。

`lv_tick_inc` 应该在比 `lv_task_handler()` 更高优先级的例程中调用（例如在中断中），以精确知道经过的毫秒数，即使 `lv_task_handler` 的执行需要更多时间。

使用 FreeRTOS，可以在 `vApplicationTickHook` 中调用 `lv_tick_inc`。

在基于 Linux 的操作系统（例如在 Raspberry Pi 上）可以在如下线程中调用 `lv_tick_inc`：

```
void * tick_thread (void *args)
{
      while(1) {
        usleep(5*1000);   /*Sleep for 5 millisecond*/
        lv_tick_inc(5);      /*Tell LVGL that 5 milliseconds were elapsed*/
    }
}
```

API

Provide access to the system tick with 1 millisecond resolution

Functions

uint32_t **lv_tick_get**(void)

Get the elapsed milliseconds since start up

返回 the elapsed milliseconds

uint32_t **lv_tick_elaps**(uint32_t prev_tick)

Get the elapsed milliseconds since a previous time stamp

参数 **prev_tick** -- a previous time stamp (return value of *lv_tick_get()*)

返回 the elapsed milliseconds since 'prev_tick'

2.4.5 Task Handler（任务处理器）

To handle the tasks of LVGL you need to call `lv_timer_handler()` periodically in one of the following:

- *while(1)* of *main()* function
- timer interrupt periodically (lower priority than `lv_tick_inc()`)
- an OS task periodically

The timing is not critical but it should be about 5 milliseconds to keep the system responsive.

Example:

要处理 LVGL 的任务，您需要以下列方式之一定期调用 `lv_timer_handler()`：

- *main()* 函数的 *while(1)*
- 定时器定期中断（比 `lv_tick_inc()` 优先级低）
- 定期执行操作系统任务

时间并不重要，但它应该是大约 5 毫秒以保持系统响应。

示例：

```
while(1) {
  lv_timer_handler();
  my_delay_ms(5);
}
```

To learn more about timers visit the *Timer* section.

要了解有关计时器的更多信息，请访问*Timer* 部分。

2.4.6 Sleep management（睡眠管理）

The MCU can go to sleep when no user input happens. In this case, the main `while(1)` should look like this:

当没有用户输入发生时，MCU 可以进入睡眠状态。在这种情况下，主要的 `while(1)` 应该是这样的：

```
while(1) {
  /*Normal operation (no sleep) in < 1 sec inactivity*/
  if(lv_disp_get_inactive_time(NULL) < 1000) {
          lv_task_handler();
  }
  /*Sleep after 1 sec inactivity*/
  else {
```

(下页继续)

(续上页)

```
            timer_stop();   /*Stop the timer where lv_tick_inc() is called*/
            sleep();                    /*Sleep the MCU*/
    }
    my_delay_ms(5);
}
```

You should also add the below lines to your input device read function to signal a wake-up (press, touch or click etc.) happened:

您还应该将以下几行添加到您的输入设备读取功能中，以表示发生了唤醒（按下、触摸或点击等）：

```
lv_tick_inc(LV_DISP_DEF_REFR_PERIOD);  /*Force task execution on wake-up*/
timer_start();                         /*Restart the timer where lv_tick_inc() is␣
↪called*/
lv_task_handler();                     /*Call `lv_task_handler()` manually to process␣
↪the wake-up event*/
```

In addition to `lv_disp_get_inactive_time()` you can check `lv_anim_count_running()` to see if all animations have finished.

除了 `lv_disp_get_inactive_time()` 之外，你还可以检查 `lv_anim_count_running()` 以查看是否所有动画都已完成。

2.4.7 Operating system and interrupts（操作系统和中断）

LVGL is **not thread-safe** by default.

However, in the following conditions it's valid to call LVGL related functions:

- In *events*. Learn more in *Events*.
- In *lv_timer*. Learn more in *Timers*.

默认情况下，LVGL **非线程安全**。

但是，在以下情况下调用 LVGL 相关函数是有效的：

- 在事件中。在事件 中了解更多信息。
- 在 *lv_timer* 中。在计时器 中了解更多信息。

Tasks and threads（任务和线程）

If you need to use real tasks or threads, you need a mutex which should be invoked before the call of `lv_timer_handler` and released after it. Also, you have to use the same mutex in other tasks and threads around every LVGL (`lv_...`) related function call and code. This way you can use LVGL in a real multitasking environment. Just make use of a mutex to avoid the concurrent calling of LVGL functions.

如果你需要使用真正的任务或线程，你需要一个互斥锁，它应该在调用 `lv_timer_handler` 之前被调用并在它之后释放。此外，您必须在每个 LVGL (`lv_...`) 相关函数调用和代码周围的其他任务和线程中使用相同的互斥锁。这样你就可以在真正的多任务环境中使用 LVGL。只需使用互斥锁来避免并发调用 LVGL 函数。

Interrupts（中断）

Try to avoid calling LVGL functions from interrupt handlers (except `lv_tick_inc()` and `lv_disp_flush_ready()`). But if you need to do this you have to disable the interrupt which uses LVGL functions while `lv_timer_handler` is running. It's a better approach to set a flag or some value and periodically check it in an `lv_timer`.

尽量避免从中断处理程序调用 LVGL 函数（除了 `lv_tick_inc()` 和 `lv_disp_flush_ready()`）。但是如果你需要这样做，你必须在 `lv_timer_handler` 运行时禁用使用 LVGL 函数的中断。设置一个标志或某个值并在 `lv_timer` 中定期检查它是一种更好的方法。

2.4.8 Logging（日志）

LVGL has built-in *Log* module to inform the user about what is happening in the library.

LVGL 有内置的 *Log* 模块来通知用户库中发生的事情。

Log level（日记等级）

To enable logging, set `LV_USE_LOG 1` in `lv_conf.h` and set `LV_LOG_LEVEL` to one of the following values:

- `LV_LOG_LEVEL_TRACE` A lot of logs to give detailed information
- `LV_LOG_LEVEL_INFO` Log important events
- `LV_LOG_LEVEL_WARN` Log if something unwanted happened but didn't cause a problem
- `LV_LOG_LEVEL_ERROR` Only critical issues, where the system may fail
- `LV_LOG_LEVEL_USER` Only user messages
- `LV_LOG_LEVEL_NONE` Do not log anything

The events which have a higher level than the set log level will be logged too. E.g. if you `LV_LOG_LEVEL_WARN`, errors will be also logged.

要启用日志记录，请在“lv_conf.h”中设置“LV_USE_LOG 1”并将“LV_LOG_LEVEL”设置为以下值之一：

- LV_LOG_LEVEL_TRACE 大量日志提供详细信息
- LV_LOG_LEVEL_INFO 记录重要事件
- LV_LOG_LEVEL_WARN 记录是否发生了不想要的事情但没有引起问题
- LV_LOG_LEVEL_ERROR 只有关键问题，系统可能会失败
- LV_LOG_LEVEL_USER 仅用户消息
- LV_LOG_LEVEL_NONE 不记录任何内容

级别高于设置的日志级别的事件也将被记录。例如。如果你 LV_LOG_LEVEL_WARN，错误也会被记录。

Printing logs（打印日志）

Logging with printf（使用 printf 记录）

If your system supports printf, you just need to enable LV_LOG_PRINTF in lv_conf.h to send the logs with printf.

如果你的系统支持 printf，你只需要在 lv_conf.h 中启用 LV_LOG_PRINTF 就可以发送带有 printf 的日志。

Custom log function（自定义日志功能）

If you can't use printf or want to use a custom function to log, you can register a "logger" callback with lv_log_register_print_cb().

For example:

如果你不能使用 printf 或者想使用自定义函数来记录日志，你可以使用 lv_log_register_print_cb() 注册一个“记录器”回调。

例如：

```
void my_log_cb(const char * buf)
{
  serial_send(buf, strlen(buf));
}

...


lv_log_register_print_cb(my_log_cb);
```

Add logs（添加日志）

You can also use the log module via the `LV_LOG_TRACE/INFO/WARN/ERROR/USER(text)` functions.

您还可以通过 `LV_LOG_TRACE/INFO/WARN/ERROR/USER(text)` 函数使用日志模块。

2.5 Overview（概览）

2.5.1 Objects（对象）

In LVGL the **basic building blocks** of a user interface are the objects, also called *Widgets*. For example a *Button*, *Label*, *Image*, *List*, *Chart* or *Text area*.

You can see all the *Object types* here.

All objects are referenced using an `lv_obj_t` pointer as a handle. This pointer can later be used to set or get the attributes of the object.

在 LVGL 中，用户界面的**基本构建块**是对象，也称为 *Widgets*。例如*Button*、*Label*、*Image*、*List* , 图表 或文本区域。

您可以在此处查看所有对象类型。

所有对象都使用 `lv_obj_t` 指针作为句柄进行引用。此指针稍后可用于设置或获取对象的属性。

Attributes（属性）

Basic attributes（基本属性）

All object types share some basic attributes:

- Position
- Size
- Parent
- Styles
- Event handlers
- Etc

You can set/get these attributes with `lv_obj_set_...` and `lv_obj_get_...` functions. For example:

所有对象类型共享一些基本属性：

- 位置
- 尺寸

- 家长
- 样式
- 事件处理程序
- 等等

您可以使用 `lv_obj_set_...` 和 `lv_obj_get_...` 函数设置/获取这些属性。例如：

```
/*Set basic object attributes*/
lv_obj_set_size(btn1, 100, 50);          /*Set a button's size*/
lv_obj_set_pos(btn1, 20,30);      /*Set a button's position*/
```

To see all the available functions visit the *Base object's documentation*.

要查看所有可用函数，请访问*Base object's documentation*。

Specific attributes（特定属性）

The object types have special attributes too. For example, a slider has

- Minimum and maximum values
- Current value

For these special attributes, every object type may have unique API functions. For example for a slider:

对象类型也有特殊的属性。例如，一个滑块有

- 最小值和最大值
- 当前值

对于这些特殊的属性，每个对象类型都可能有唯一的 API 函数。例如对于滑块：

```
/*Set slider specific attributes*/
lv_slider_set_range(slider1, 0, 100);                                    /*Set␣
↪the min. and max. values*/
lv_slider_set_value(slider1, 40, LV_ANIM_ON);              /*Set the current value␣
↪(position)*/
```

The API of the widgets is described in their *Documentation* but you can also check the respective header files (e.g. *widgets/lv_slider.h*)

小部件的 API 在它们的文档中有描述，但您也可以检查相应的头文件（例如 *widgets/lv_slider.h*）

Working mechanisms（工作机制）

Parent-child structure（父子结构）

A parent object can be considered as the container of its children. Every object has exactly one parent object (except screens), but a parent can have any number of children. There is no limitation for the type of the parent but, there are typical parent (e.g. button) and typical child (e.g. label) objects.

父对象可以被视为其子对象的容器。每个对象只有一个父对象（屏幕除外），但一个父对象可以有任意数量的子对象。父对象的类型没有限制，但是有典型的父对象（例如按钮）和典型的子对象（例如标签）。

Moving together（一起移动）

If the position of the parent changes the children will move with the parent. Therefore all positions are relative to the parent.

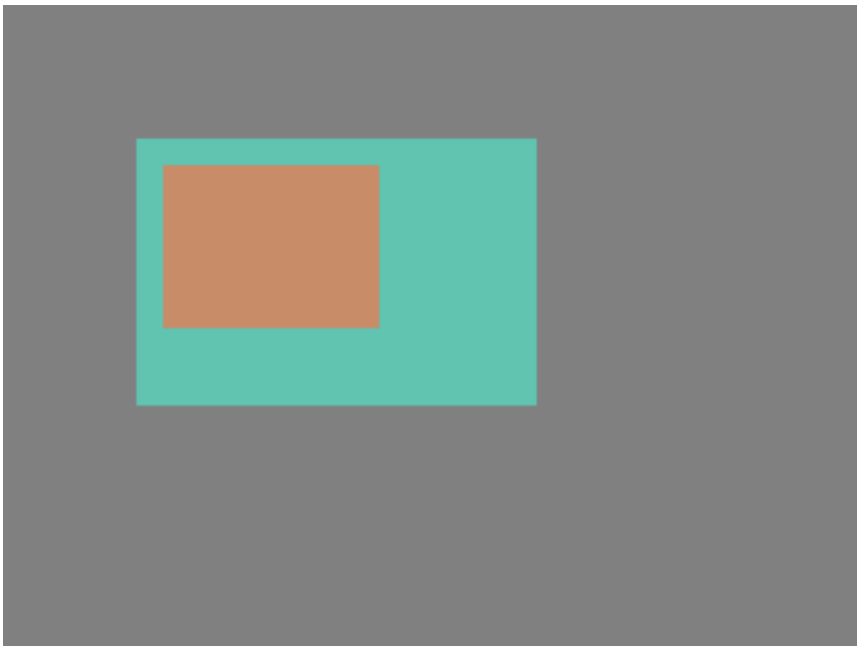
如果父节点的位置发生变化，子节点将与父节点一起移动。因此，所有位置都相对于父级。

```
lv_obj_t * parent = lv_obj_create(lv_scr_act());   /*Create a parent object on the␣
↪current screen*/
lv_obj_set_size(parent, 100, 80);                        /*Set the size of the␣
↪parent*/

lv_obj_t * obj1 = lv_obj_create(parent);                /*Create an object on the␣
↪previously created parent object*/
lv_obj_set_pos(obj1, 10, 10);                            /*Set the position of the␣
↪new object*/
```

Modify the position of the parent:

修改父级的位置:



```
lv_obj_set_pos(parent, 50, 50);         /*Move the parent. The child will move with it.
↪*/
```

(For simplicity the adjusting of colors of the objects is not shown in the example.)

（为简单起见，示例中未显示对象颜色的调整。）

Visibility only on the parent（仅在父对象上可见）

If a child is partially or fully out of its parent then the parts outside will not be visible.

如果孩子部分或完全脱离其父对象，则外部部分将不可见。

```
lv_obj_set_x(obj1, -30);        /*Move the child a little bit off the parent*/
```

Create and delete objects（创建和删除对象）

In LVGL objects can be created and deleted dynamically in run time. It means only the currently created (existing) objects consume RAM.

This allows for the creation of a screen just when a button is clicked to open it, and for deletion of screens when a new screen is loaded.

UIs can be created based on the current environment of the device. For example one can create meters, charts, bars and sliders based on the currently attached sensors.

Every widget has its own **create** function with a prototype like this:

在 LVGL 中，可以在运行时动态创建和删除对象。这意味着只有当前创建的（现有）对象消耗 RAM。

这允许仅在单击按钮打开屏幕时创建屏幕，并在加载新屏幕时删除屏幕。

可以根据设备的当前环境创建 UI。例如，可以根据当前连接的传感器创建仪表、图表、条形图和滑块。

每个小部件都有自己的 **create** 函数，原型如下：

```
lv_obj_t * lv_<widget>_create(lv_obj_t * parent, <other paramaters if any>);
```

In most of the cases the create functions have only a *parent* parameter that tells on which object create the new widget.

The return value is a pointer to the created object with `lv_obj_t *` type.

There is a common **delete** function for all object types. It deletes the object and all of its children.

在大多数情况下，create 函数只有一个 *parent* 参数，它告诉在哪个对象上创建新小部件。

返回值是一个指向创建对象的指针，类型为 `lv_obj_t *`。

所有对象类型都有一个通用的 **delete** 函数。它删除对象及其所有子对象。

```
void lv_obj_del(lv_obj_t * obj);
```

`lv_obj_del` will delete the object immediately. If for any reason you can't delete the object immediately you can use `lv_obj_del_async(obj)` that will perform the deletion on the next call of `lv_timer_handler()`. This is useful e.g. if you want to delete the parent of an object in the child's `LV_EVENT_DELETE` handler.

You can remove all the children of an object (but not the object itself) using `lv_obj_clean(obj)`.

You can use `lv_obj_del_delayed(obj, 1000)` to delete an object after some time. The delay is expressed in millliseconds.

`lv_obj_del` 将立即删除对象。如果由于任何原因你不能立即删除对象，你可以使用 `lv_obj_del_async(obj)` 它将在下一次调用 `lv_timer_handler()` 时执行删除。这很有用，例如如果您想在子对象的 `LV_EVENT_DELETE` 处理程序中删除对象的父对象。您可以使用 `lv_obj_clean(obj)` 删除对象的所有子项（但不是对象本身）。一段时间后，您可以使用 `lv_obj_del_delayed(obj, 1000)` 来删除对象。延迟以毫秒表示。

Screens（屏幕）

Create screens（创建屏幕）

The screens are special objects which have no parent object. So they can be created like:

屏幕是没有父对象的特殊对象。所以它们可以像这样创建：

```
lv_obj_t * scr1 = lv_obj_create(NULL);
```

Screens can be created with any object type. For example, a *Base object* or an image to make a wallpaper.

可以使用任何对象类型创建画面。例如，基础对象 或用于制作壁纸的图像。

Get the active screen（获取活动屏幕）

There is always an active screen on each display. By default, the library creates and loads a "Base object" as a screen for each display.

To get the currently active screen use the `lv_scr_act()` function.

每个显示器上总是有一个活动屏幕。默认情况下，库创建并加载一个“基础对象”作为每个显示的屏幕。

要获取当前活动的屏幕，请使用 `lv_scr_act()` 函数。

Load screens（加载屏幕）

To load a new screen, use `lv_scr_load(scr1)`.

请使用 `lv_scr_load(scr1)` 加载你要加载的屏幕。

Layers（层）

There are two automatically generated layers:

- top layer
- system layer

They are independent of the screens and they will be shown on every screen. The *top layer* is above every object on the screen and the *system layer* is above the *top layer* too. You can add any pop-up windows to the *top layer* freely. But, the *system layer* is restricted to system-level things (e.g. mouse cursor will be placed here in `lv_indev_set_cursor()`).

The `lv_layer_top()` and `lv_layer_sys()` functions return pointers to the top and system layers respectively.

Read the *Layer overview* section to learn more about layers.

有两个自动生成的层：

- 顶层
- 系统层

它们独立于屏幕，将显示在每个屏幕上。*顶层*位于屏幕上的每个对象之上，*系统层*也位于*顶层*之上。您可以自由地将任何弹出窗口添加到*顶层*。但是，*系统层*仅限于系统级事物（例如，鼠标光标将放置在“lv_indev_set_cursor()”中）。

`lv_layer_top()` 和 `lv_layer_sys()` 函数分别返回指向顶层和系统层的指针。

阅读图层概述 部分以了解有关图层的更多信息。

Load screen with animation（用动画加载屏幕）

A new screen can be loaded with animation too using `lv_scr_load_anim(scr, transition_type, time, delay, auto_del)`. The following transition types exist:

- `LV_SCR_LOAD_ANIM_NONE`: switch immediately after `delay` milliseconds
- `LV_SCR_LOAD_ANIM_OVER_LEFT/RIGHT/TOP/BOTTOM` move the new screen over the current towards the given direction
- `LV_SCR_LOAD_ANIM_MOVE_LEFT/RIGHT/TOP/BOTTOM` move both the current and new screens towards the given direction
- `LV_SCR_LOAD_ANIM_FADE_ON` fade the new screen over the old screen

Setting `auto_del` to `true` will automatically delete the old screen when the animation is finished.

The new screen will become active (returned by `lv_scr_act()`) when the animations starts after `delay` time.

新屏幕也可以使用 `lv_scr_load_anim(scr, transition_type, time, delay, auto_del)` 加载动画。存在以下转换类型：

- `LV_SCR_LOAD_ANIM_NONE`：在 `delay` 毫秒后立即切换
- `LV_SCR_LOAD_ANIM_OVER_LEFT/RIGHT/TOP/BOTTOM` 将新屏幕移动到当前的指定方向
- `LV_SCR_LOAD_ANIM_MOVE_LEFT/RIGHT/TOP/BOTTOM` 将当前屏幕和新屏幕都向给定方向移动
- `LV_SCR_LOAD_ANIM_FADE_ON` 在旧屏幕上淡出新屏幕

将 `auto_del` 设置为 `true` 将在动画完成时自动删除旧屏幕。当动画在 `delay` 时间后开始时，新屏幕将变为活动状态（由 `lv_scr_act()` 返回）。

Handling multiple displays（处理多个显示器）

Screens are created on the currently selected *default display*. The *default display* is the last registered display with `lv_disp_drv_register` or you can explicitly select a new default display using `lv_disp_set_default(disp)`.

`lv_scr_act()`, `lv_scr_load()` and `lv_scr_load_anim()` operate on the default screen.

Visit *Multi-display support* to learn more.

屏幕是在当前选择的默认显示上创建的。*default display* 是最后一个用 `lv_disp_drv_register` 注册的显示，或者你可以使用 `lv_disp_set_default(disp)` 明确地选择一个新的默认显示。

`lv_scr_act()`、`lv_scr_load()` 和 `lv_scr_load_anim()` 在默认屏幕上运行。

访问多显示器支持 了解更多信息。

Parts（部分）

The widgets are built from multiple parts. For example a *Base object* uses the main and scrollbar parts but a *Slider* uses the main, the indicator and the knob parts. Parts are similar to *pseudo elements* in CSS.

The following predefined parts exist in LVGL:

- `LV_PART_MAIN` A background like rectangle*/“
- `LV_PART_SCROLLBAR` The scrollbar(s)
- `LV_PART_INDICATOR` Indicator, e.g. for slider, bar, switch, or the tick box of the checkbox
- `LV_PART_KNOB` Like a handle to grab to adjust the value*/
- `LV_PART_SELECTED` Indicate the currently selected option or section
- `LV_PART_ITEMS` Used if the widget has multiple similar elements (e.g. tabel cells)*/

- LV_PART_TICKS Ticks on scales e.g. for a chart or meter
- LV_PART_CURSOR Mark a specific place e.g. text area's or chart's cursor
- LV_PART_CUSTOM_FIRST Custom parts can be added from here.

The main purpose of parts to allow styling the "components" of the widgets. Therefore the parts are described in more detail in the *Style overview* section.

小部件由多个部分构建而成。例如，*Base object* 使用主部件和滚动条部件，而*Slider* 使用主部件、指示器和旋钮部件。部分类似于 CSS 中的*伪元素*。

LVGL 中存在以下预定义部分：

- LV_PART_MAIN 类似矩形的背景 */"
- LV_PART_SCROLLBAR 滚动条
- LV_PART_INDICATOR 指标，例如用于滑块、条、开关或复选框的勾选框
- LV_PART_KNOB 像手柄一样可以抓取调整值 */
- LV_PART_SELECTED 表示当前选择的选项或部分
- LV_PART_ITEMS 如果小部件有多个相似的元素（例如表格单元格）*/
- LV_PART_TICKS 刻度上的刻度，例如对于图表或仪表
- LV_PART_CURSOR 标记一个特定的地方，例如文本区域或图表的光标
- LV_PART_CUSTOM_FIRST 可以从这里添加自定义部件。

部件的主要目的是允许为小部件的“组件”设置样式。因此，在样式概述 部分更详细地描述了这些部分。

States（状态）

The object can be in a combination of the following states:

- LV_STATE_DEFAULT Normal, released state
- LV_STATE_CHECKED Toggled or checked state
- LV_STATE_FOCUSED Focused via keypad or encoder or clicked via touchpad/mouse
- LV_STATE_FOCUS_KEY Focused via keypad or encoder but not via touchpad/mouse
- LV_STATE_EDITED Edit by an encoder
- LV_STATE_HOVERED Hovered by mouse (not supported now)
- LV_STATE_PRESSED Being pressed
- LV_STATE_SCROLLED Being scrolled
- LV_STATE_DISABLED Disabled state
- LV_STATE_USER_1 Custom state

- LV_STATE_USER_2 Custom state
- LV_STATE_USER_3 Custom state
- LV_STATE_USER_4 Custom state

The states are usually automatically changed by the library as the user presses, releases, focuses etc an object. However, the states can be changed manually too. To set or clear given state (but leave the other states untouched) use `lv_obj_add/clear_state(obj, LV_STATE_...)` In both cases ORed state values can be used as well. E.g. `lv_obj_add_state(obj, part, LV_STATE_PRESSED | LV_PRESSED_CHECKED)`.

To learn more about the states read the related section of the *Style overview*.

对象可以处于以下状态的组合：

- LV_STATE_DEFAULT 正常，释放状态
- LV_STATE_CHECKED 切换或选中状态
- LV_STATE_FOCUSED 通过键盘或编码器聚焦或通过触摸板/鼠标点击
- LV_STATE_FOCUS_KEY 通过键盘或编码器聚焦，但不通过触摸板/鼠标聚焦
- LV_STATE_EDITED 由编码器编辑
- LV_STATE_HOVERED 鼠标悬停（现在不支持）
- LV_STATE_PRESSED 被按下
- LV_STATE_SCROLLED 正在滚动
- LV_STATE_DISABLED 禁用状态
- LV_STATE_USER_1 自定义状态
- LV_STATE_USER_2 自定义状态
- LV_STATE_USER_3 自定义状态
- LV_STATE_USER_4 自定义状态

当用户按下、释放、聚焦等对象时，库通常会自动更改状态。但是，状态也可以手动更改。要设置或清除给定状态（但保持其他状态不变），请使用 `lv_obj_add/clear_state(obj, LV_STATE_...)` 在这两种情况下，也可以使用 ORed 状态值。例如。`lv_obj_add_state(obj, part, LV_STATE_PRESSED | LV_PRESSED_CHECKED)`。

要了解有关状态的更多信息，请阅读样式概述 的相关部分。

Snapshot（快照）

A snapshot image could be generated for object together with its children. Check details in *Snapshot*.

可以为对象及其子对象生成快照图像。在快照中查看详细信息。

2.5.2 Positions, sizes, and layouts（位置、大小和布局）

Overview（概述）

Similarly to many other parts of LVGL, the concept of setting the coordinates was inspired by CSS. By no means a complete implementation of the standard but subsets of CSS were implemented (sometimes with minor adjustments). In shorts this means:

- the set coordinates (size, position, layouts, etc) are stored in styles
- support min-width, max-width, min-height, max-height
- have pixel, percentage, and ”content” units
- x=0; y=0 coordinate means the to top-left corner of the parent plus the left/top padding plus border width
- width/height means the full size, the ”content area” is smaller with padding and border width
- a subset of flexbox and grid layouts are supported

与 LVGL 的许多其他部分类似，设置坐标的概念受到 CSS 的启发。绝不是标准的完整实现，而是实现了 CSS 的子集（有时会稍作调整）。简而言之，这意味着：

- 设置的坐标（大小、位置、布局等）存储在样式中
- 支持最小宽度、最大宽度、最小高度、最大高度
- 有像素、百分比和“内容”单位
- x=0; y=0 坐标表示父级的左上角加上左/上填充加上边框宽度
- 宽度/高度表示全尺寸，“内容区域”较小，填充和边框宽度
- 支持 flexbox 和网格布局的子集

Units（单位）

- pixel: Simply a position in pixels. A simple integer always means pixel. E.g. `lv_obj_set_x(btn, 10)`
- percentage: The percentage of the size of the object or its parent (depending on the property). The `lv_pct(value)` converts a value to percentage. E.g. `lv_obj_set_width(btn, lv_pct(50))`
- `LV_SIZE_CONTENT`: Special value to set the width/height of an object to involve all the children. Its similar to `auto` in CSS. E.g. `lv_obj_set_width(btn, LV_SIZE_CONTENT)`.

- 像素：只是一个以像素为单位的位置。一个简单的整数总是意味着像素。例如。`lv_obj_set_x(btn, 10)`
- 百分比：对象或其父对象的大小百分比（取决于属性）。`lv_pct(value)` 将值转换为百分比。例如。`lv_obj_set_width(btn, lv_pct(50))`
- `LV_SIZE_CONTENT`：设置对象的宽度/高度以涉及所有子项的特殊值。它类似于 CSS 中的“auto”。例如。`lv_obj_set_width(btn, LV_SIZE_CONTENT)`。

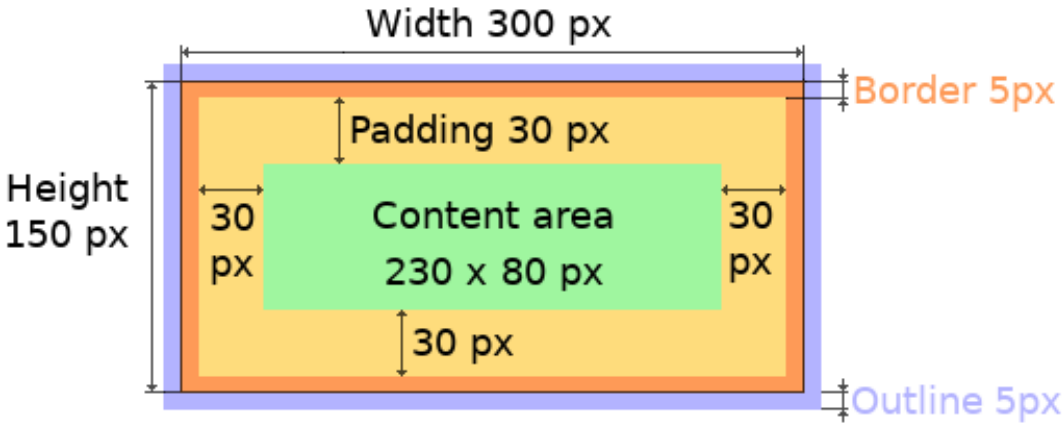
Boxing model（盒子模型）

LVGL follows CSS's border-box model. An object's "box" is built from the following parts:

- bounding box: the width/height of the elements.
- border width: the width of the border.
- padding: space between the sides of the object and its children.
- content: the content area which size if the bounding box reduced by the border width and the size of the paddings.

LVGL 遵循 CSS 的 border-box 模型。对象的“盒子”由以下部分构成：

- 边界 (bounding) 框：元素的宽度/高度围起来的区域。
- 边框 (border) 宽度：边框的宽度。
- 填充 (padding)：对象两侧与其子对象之间的空间。
- 内容 (content)：如果边界框按边框宽度和填充的大小缩小，则显示其大小的内容区域。



The border is drawn inside the bounding box. Inside the border LVGL keeps "padding size" to place the children.

The outline is drawn outside of the bounding box.

边框 (border) 绘制在边界 (bounding) 框内。在边界 (border) 内 LVGL 保持“填充 (padding) 大小”来放置孩子。

轮廓 (outline) 绘制在边界框之外。

Important notes（重要笔记）

This section describes special cases in which LVGL's behavior might be unexpected.

本节描述了 LVGL 的行为可能出乎意料的特殊情况。

Postponed coordinate calculation（坐标会被延迟计算）

LVGL doesn't recalculate all the coordinate changes immediately. This is done to improve performance. Instead, the objects are marked as "dirty" and before redrawing the screen LVGL checks if there are any "dirty" objects. If so it refreshes their position, size and layout.

In other words, if you need to get the any coordinate of an object and it the coordinates were just changed LVGL's needs to be forced to recalculate the coordinates. To do this call `lv_obj_update_layout(obj)`.

The size and position might depend on the parent or layout. Therefore `lv_obj_update_layout` recalculates the coordinates of all objects on the screen of `obj`.

LVGL 不会立即重新计算所有坐标变化。这样做是为了提高性能。相反，对象被标记为 "脏"，并且在重绘屏幕之前 LVGL 检查是否有任何 "脏" 对象。如果是这样，它会刷新它们的位置、大小和布局。

换句话说，如果您需要获取对象的任何坐标并且坐标刚刚更改，则需要强制 LVGL 重新计算坐标。为此调用 `lv_obj_update_layout(obj)`。

大小和位置可能取决于父级或布局。因此 `lv_obj_update_layout` 重新计算 `obj` 屏幕上所有对象的坐标。

Removing styles（删除样式）

As it's described in the *Using styles* section the coordinates can be set via style properties too. To be more precise under the hood every style coordinate related property is stored as style a property. If you use `lv_obj_set_x(obj, 20)` LVGL saves `x=20` in the local style of the object.

It's an internal mechanism and doesn't matter much as you use LVGL. However, there is one case in which you need to aware of that. If the style(s) of an object are removed by

正如使用样式 部分所述，坐标也可以通过样式属性设置。更准确地说，每个与样式坐标相关的属性都存储为样式属性。如果你使用 `lv_obj_set_x(obj, 20)` LVGL 将 `x=20` 保存在对象的本地样式中。

这是一种内部机制，与您使用 LVGL 无关。但是，在一种情况下，您需要了解这一点。如果对象的样式被删除

```
lv_obj_remove_style_all(obj)
```

or

或者

```
lv_obj_remove_style(obj, NULL, LV_PART_MAIN);
```

the earlier set coordinates will be removed as well.

For example:

先前设置的坐标也将被删除。

例如：

```
/*The size of obj1 will be set back to the default in the end*/
lv_obj_set_size(obj1, 200, 100);  /*Now obj1 has 200;100 size*/
lv_obj_remove_style_all(obj1);    /*It removes the set sizes*/


/*obj2 will have 200;100 size in the end */
lv_obj_remove_style_all(obj2);
lv_obj_set_size(obj2, 200, 100);
```

Position（位置）

Simple way（最简单的方法）

To simple set the x and y coordinates of an object use

要简单设置对象的 x 和 y 坐标，请使用

```
lv_obj_set_x(obj, 10);
lv_obj_set_y(obj, 20);
lv_obj_set_pos(obj, 10, 20);         //Or in one function
```

By default the the x and y coordinates are measured from the top left corner of the parent's content area. For example if the parent has 5 pixels padding on every side, the above code will place `obj` at (15, 25) because the content area starts after the padding.

If percentage values are calculated from the parents content area size.

默认情况下，x 和 y 坐标是从父内容区域的左上角开始测量的。例如，如果父级每边有 5 个像素的填充，上面的代码会将 `obj` 放置在 (15, 25) 处，因为内容区域在填充之后开始。

如果百分比值是根据父内容区域大小计算的。

```
lv_obj_set_x(btn, lv_pct(10)); //x = 10 % of parant content area width
```

Align（对齐）

In some cases it's convenient to change the origin of the positioning from the the default top left. If the origin is changed e.g. to bottom-right, the (0,0) position means: align to the bottom-right corner. To change the origin use:

在某些情况下，从默认的左上角更改定位的原点会很方便。如果原点改变，例如到右下角，(0,0) 位置表示：与右下角对齐。要更改原点使用：

```
lv_obj_set_align(obj, align);
```

To change the alignment and set new coordinates:

要更改对齐方式并设置新坐标：

```
lv_obj_align(obj, align, x, y);
```

The following alignment options can be used:

- LV_ALIGN_TOP_LEFT
- LV_ALIGN_TOP_MID
- LV_ALIGN_TOP_RIGHT
- LV_ALIGN_BOTTOM_LEFT
- LV_ALIGN_BOTTOM_MID
- LV_ALIGN_BOTTOM_RIGHT
- LV_ALIGN_LEFT_MID
- LV_ALIGN_RIGHT_MID
- LV_ALIGN_CENTER

It quite common to align a children to the center of its parent, there fore is a dedicated function for it:

可以使用以下对齐选项：

- LV_ALIGN_TOP_LEFT -LV_ALIGN_TOP_MID -LV_ALIGN_TOP_RIGHT -LV_ALIGN_BOTTOM_LEFT -LV_ALIGN_BOTTOM_MID -LV_ALIGN_BOTTOM_RIGHT
- LV_ALIGN_LEFT_MID -LV_ALIGN_RIGHT_MID -LV_ALIGN_CENTER

将孩子对齐到其父母的中心是很常见的，因此有一个专门的功能：

```
lv_obj_center(obj);

//Has the same effect
lv_obj_align(obj, LV_ALIGN_CENTER, 0, 0);
```

If the parent's size changes the set alignment and position of the children is applied again automatically.

The functions introduced above aligns the object to its parent. However it's also possible to align an object to an arbitrary object.

如果父项的大小更改，则会自动再次应用子项的设置对齐方式和位置。

上面介绍的函数将对象与其父对象对齐。但是，也可以将对象与任意对象对齐。

```
lv_obj_align_to(obj_to_align, reference_obj, align, x, y);
```

Besides the alignments options above the following can be used to align the object outside of the reference object:

- LV_ALIGN_OUT_TOP_LEFT
- LV_ALIGN_OUT_TOP_MID
- LV_ALIGN_OUT_TOP_RIGHT
- LV_ALIGN_OUT_BOTTOM_LEFT
- LV_ALIGN_OUT_BOTTOM_MID
- LV_ALIGN_OUT_BOTTOM_RIGHT
- LV_ALIGN_OUT_LEFT_TOP
- LV_ALIGN_OUT_LEFT_MID
- LV_ALIGN_OUT_LEFT_BOTTOM
- LV_ALIGN_OUT_RIGHT_TOP
- LV_ALIGN_OUT_RIGHT_MID
- LV_ALIGN_OUT_RIGHT_BOTTOM

For example to align a label above a button and center the label horizontally:

除了上面的对齐选项之外，以下选项还可用于对齐参考对象之外的对象：

- LV_ALIGN_OUT_TOP_LEFT
- LV_ALIGN_OUT_TOP_MID
- LV_ALIGN_OUT_TOP_RIGHT
- LV_ALIGN_OUT_BOTTOM_LEFT
- LV_ALIGN_OUT_BOTTOM_MID
- LV_ALIGN_OUT_BOTTOM_RIGHT
- LV_ALIGN_OUT_LEFT_TOP
- LV_ALIGN_OUT_LEFT_MID
- LV_ALIGN_OUT_LEFT_BOTTOM

- LV_ALIGN_OUT_RIGHT_TOP
- LV_ALIGN_OUT_RIGHT_MID
- LV_ALIGN_OUT_RIGHT_BOTTOM

例如，在按钮上方对齐标签并使标签水平居中：

```
lv_obj_align_to(label, btn, LV_ALIGN_OUT_TOP_MID, 0, -10);
```

Note that - unlike with `lv_obj_align()` - `lv_obj_align_to()` can not realign the object if its coordinates or the reference object's coordinates changes.

请注意 - 与 `lv_obj_align()` 不同 - `lv_obj_align_to()` 无法重新对齐对象，如果其坐标或参考对象的坐标发生变化。

Size（大小）

Simple way（最简单的方法）

The width and the height of an object can be set easily as well:

对象的宽度和高度也可以轻松设置：

```
lv_obj_set_width(obj, 200);
lv_obj_set_height(obj, 100);
lv_obj_set_size(obj, 200, 100);         //Or in one function
```

Percentage values are calculated based on the parent's content area size. For example to set the object's height to the screen height:

百分比值是根据父内容区域的大小计算的。例如将对象的高度设置为屏幕高度：

```
lv_obj_set_height(obj, lv_pct(100));
```

Size setting supports a value: `LV_SIZE_CONTENT`. It means the object's size in the respective direction will be set to the size of its children. Note that only children on the right and bottom will be considered and children on the top and left remain cropped. This limitation makes the behavior more predictable.

Objects with `LV_OBJ_FLAG_HIDDEN` or `LV_OBJ_FLAG_FLOATING` will be ignored by the `LV_SIZE_CONTENT` calculation.

The above functions set the size of the bounding box of the object but the size of the content area can be set as well. It means the object's bounding box will be larger with the paddings than the set size.

大小设置支持一个值：`LV_SIZE_CONTENT`。这意味着对象在相应方向上的大小将设置为其子对象的大小。请注意，只会考虑右侧和底部的子项，而顶部和左侧的子项仍会被裁剪。此限制使行为更可预测。

带有 `LV_OBJ_FLAG_HIDDEN` 或 `LV_OBJ_FLAG_FLOATING` 的对象将被 `LV_SIZE_CONTENT` 计算忽略。

上述函数设置对象边界框的大小，但也可以设置内容区域的大小。这意味着对象的边界框将比设置的大小更大。

```
lv_obj_set_content_width(obj, 50); //The actual width: padding left + 50 + padding␣
↪right
lv_obj_set_content_height(obj, 30); //The actual width: padding top + 30 + padding␣
↪bottom
```

The size of the bounding box and the content area can be get with the following functions:

可以使用以下函数获取边界框和内容区域的大小：

```
lv_coord_t w = lv_obj_get_width(obj);
lv_coord_t h = lv_obj_get_height(obj);
lv_coord_t content_w = lv_obj_get_content_width(obj);
lv_coord_t content_h = lv_obj_get_content_height(obj);
```

Using styles（使用样式）

Under the hood the position, size and alignment properties are style properties. The above described ”simple functions” hide the style related code for the sake of simplicity and set the position, size, and alignment properties in the local styles of the obejct.

However, using styles as to set the coordinates has some great advantages:

- It makes it easy to set the width/height/etc for several objects together. E.g. make all the sliders 100x10 pixels sized.
- It also makes possible to modify the values in one place.
- The values can be overwritten by other styles. For example `style_btn` makes the object `100x50` by default but adding `style_full_width` overwrites only the width of the object.
- The object can have different position or size in different state. E.g. 100 px wide in `LV_STATE_DEFAULT` but 120 px in `LV_STATE_PRESSED`.
- Style transitions can be used to make the coordinate changes smooth.

在驱动中，位置、大小和对齐属性是样式属性。上面描述的“简单函数”为了简单起见隐藏了样式相关的代码，并在对象的局部样式中设置了位置、大小和对齐属性。

但是，使用样式设置坐标有一些很大的优点：

- 可以轻松地为多个对象设置宽度/高度/等。例如。使所有滑块的大小为 100x10 像素。
- 还可以在一处修改值。
- 这些值可以被其他样式覆盖。例如 `style_btn` 默认使对象 `100x50`，但添加 `style_full_width` 只会覆盖对象的宽度。

- 物体在不同状态下可以有不同的位置或大小。例如。LV_STATE_DEFAULT 中的宽度为 100 像素，而 LV_STATE_PRESSED 中的宽度为 120 像素。
- 样式转换可用于使坐标变化平滑。

Here are some examples to set an object's size using a style:

以下是一些使用样式设置对象大小的示例：

```
static lv_style_t style;
lv_style_init(&style);
lv_style_set_width(&style, 100);

lv_obj_t * btn = lv_btn_create(lv_scr_act());
lv_obj_add_style(btn, &style, LV_PART_MAIN);
```

As you will see below there are some other great features of size and position setting. However, to keep the LVGL's API lean only the most common coordinate setting features have a "simple" version and the more complex features can be used via styles.

正如您将在下面看到的，还有一些其他重要的尺寸和位置设置功能。然而，为了保持 LVGL 的 API 精简，只有最常见的坐标设置功能有一个“简单”版本，更复杂的功能可以通过样式使用。

Translation（风格样式转换）

Let's say the there are 3 buttons next to each other. Their position is set as described above. Now you want to move a buttons up a little when it's pressed.

One way to achieve this is setting a new Y coordinate for pressed state:

假设有 3 个按钮彼此相邻。它们的位置如上所述设置。现在您想在按下按钮时将其向上移动一点。

实现此目的的一种方法是为按下状态设置新的 Y 坐标：

```
static lv_style_t style_normal;
lv_style_init(&style_normal);
lv_style_set_y(&style_normal, 100);

static lv_style_t style_pressed;
lv_style_init(&style_pressed);
lv_style_set_y(&style_pressed, 80);

lv_obj_add_style(btn1, &style_normal, LV_STATE_DEFAULT);
lv_obj_add_style(btn1, &style_pressed, LV_STATE_PRESSED);

lv_obj_add_style(btn2, &style_normal, LV_STATE_DEFAULT);
lv_obj_add_style(btn2, &style_pressed, LV_STATE_PRESSED);
```

(下页继续)

(续上页)

```
lv_obj_add_style(btn3, &style_normal, LV_STATE_DEFAULT);
lv_obj_add_style(btn3, &style_pressed, LV_STATE_PRESSED);
```

It works but it's not really flexible because the pressed coordinate is hard-coded. If the buttons are not at y=100 `style_pressed` won't work as expected. To solve this translations can be used:

它可以工作，但不是很灵活，因为按下的坐标是硬编码的。如果按钮不在 y=100 处，`style_pressed` 将不会按预期工作。要解决这个问题，可以参考使用以下转换：

```
static lv_style_t style_normal;
lv_style_init(&style_normal);
lv_style_set_y(&style_normal, 100);

static lv_style_t style_pressed;
lv_style_init(&style_pressed);
lv_style_set_translate_y(&style_pressed, -20);

lv_obj_add_style(btn1, &style_normal, LV_STATE_DEFAULT);
lv_obj_add_style(btn1, &style_pressed, LV_STATE_PRESSED);

lv_obj_add_style(btn2, &style_normal, LV_STATE_DEFAULT);
lv_obj_add_style(btn2, &style_pressed, LV_STATE_PRESSED);

lv_obj_add_style(btn3, &style_normal, LV_STATE_DEFAULT);
lv_obj_add_style(btn3, &style_pressed, LV_STATE_PRESSED);
```

Translation is applied from the current position of the object.

Percentage values can be used in translations as well. The percentage is relative to the size of the object (and not to the size of the parent). For example `lv_pct(50)` will move the object with half of its width/height.

The translation is applied after the layouts are calculated. Therefore, even the layouted objects' position can be translated.

The translation actually moves the object. It means it makes the scrollbars and `LV_SIZE_CONTENT` sized objects react to the position change.

从对象的当前位置开始应用平移。

百分比值也可用于翻译。百分比是相对于对象的大小（而不是父对象的大小）。例如，`lv_pct(50)` 将移动对象的宽度/高度的一半。

在计算布局后应用翻译。因此，甚至可以平移布局对象的位置。

平移实际上移动了对象。这意味着它使滚动条和 `LV_SIZE_CONTENT` 大小的对象对位置变化做出反应。

Transformation（转换）

Similarly to the position the size can be changed relative to the current size as well. The transformed width and height are added on both sides of the object. This means 10 px transformed width makes the object 2x10 pixel wider.

Unlike position translation, the size transformation doesn't make the object "really" larger. In other words scrollbars, layouts, `LV_SIZE_CONTENT` will not consider the transformed size. Hence size transformation if "only" a visual effect.

This code makes the a button larger when it's pressed:

与位置类似，大小也可以相对于当前大小进行更改。转换后的宽度和高度会添加到对象的两侧。这意味着 10 px 转换宽度使对象更宽 2x10 像素。

与位置平移不同，尺寸变换不会使对象“真正”变大。换句话说，滚动条、布局、`LV_SIZE_CONTENT` 不会考虑转换后的大小。因此，如果“仅”是视觉效果，则尺寸转换。

此代码使下面示例的按钮在按下时变大：

```
static lv_style_t style_pressed;
lv_style_init(&style_pressed);
lv_style_set_transform_width(&style_pressed, 10);
lv_style_set_transform_height(&style_pressed, 10);

lv_obj_add_style(btn, &style_pressed, LV_STATE_PRESSED);
```

Min and Max size（最小和最大尺寸）

Similarly to CSS, LVGL also support `min-width`, `max-width`, `min-height` and `max-height`. These are limits preventing an object's size to be smaller/larger then these values. They are especially useful if the size is set by percentage or `LV_SIZE_CONTENT`.

与 CSS 类似，LVGL 也支持 `min-width`、`max-width`、`min-height` 和 `max-height`。这些是防止对象的大小小于/大于这些值的限制。如果大小按百分比或 `LV_SIZE_CONTENT` 设置，则它们特别有用。

```
static lv_style_t style_max_height;
lv_style_init(&style_max_height);
lv_style_set_y(&style_max_height, 200);

lv_obj_set_height(obj, lv_pct(100));
lv_obj_add_style(obj, &style_max_height, LV_STATE_DEFAULT); //Limit the  height to␣
↪200 px
```

Percentage values can be used as well which are relative to the size of the parent's content area size.

也可以使用与父内容区域大小相关的百分比值。

```
static lv_style_t style_max_height;
lv_style_init(&style_max_height);
lv_style_set_y(&style_max_height, lv_pct(50));

lv_obj_set_height(obj, lv_pct(100));
lv_obj_add_style(obj, &style_max_height, LV_STATE_DEFAULT); //Limit the height to␣
↪half parent height
```

Layout（布局）

Overview（概述）

Layouts can update the position and size of an object's children. They can be used to automatically arrange the children into a line or column, or in much more complicated forms.

The position and size set by the layout overwrites the "normal" x, y, width, and height settings.

There is only one function that is the same for every layout: `lv_obj_set_layout(obj, <LAYOUT_NAME>)` sets the layout on an object.For the further settings of the parent and children see the documentations of the given layout.

布局可以更新对象子项的位置和大小。它们可用于自动将子项排列成一行或一列，或者以更复杂的形式排列。

布局设置的位置和大小会覆盖“正常”的 x、y、宽度和高度设置。

每个布局只有一个相同的函数：`lv_obj_set_layout(obj, <LAYOUT_NAME>)` 在对象上设置布局。有关父级和子级的进一步设置，请参阅给定布局的文档。

Built-in layout

LVGL comes with two very powerful layouts:

- Flexbox
- Grid

Both are heavily inspired by the CSS layouts with the same name.

LVGL 带有两个非常强大的布局：

- Flexbox(弹性伸缩盒)
- Grid(网格)

两者都深受 CSS 布局的启发。

Flags（标志）

There are some flags that can be used on object to affect how they behave with layouts:

- `LV_OBJ_FLAG_HIDDEN` Hidden object are ignored from layout calculations.
- `LV_OBJ_FLAG_IGNORE_LAYOUT` The object is simply ignored by the layouts. Its coordinates can be set as usual.
- `LV_OBJ_FLAG_FLOATING` Same as `LV_OBJ_FLAG_IGNORE_LAYOUT` but the object with `LV_OBJ_FLAG_FLOATING` will be ignored from `LV_SIZE_CONTENT` calculations.

These flags can be added/removed with `lv_obj_add/clear_flag(obj, FLAG);`

有一些标志可用于对象以影响它们在布局中的行为：

- `LV_OBJ_FLAG_HIDDEN` 隐藏对象从布局计算中被忽略。
- `LV_OBJ_FLAG_IGNORE_LAYOUT` 该对象被布局简单地忽略。它的坐标可以照常设置。
- `LV_OBJ_FLAG_FLOATING` 与 `LV_OBJ_FLAG_IGNORE_LAYOUT` 相同，但带有 `LV_OBJ_FLAG_FLOATING` 的对象将在 `LV_SIZE_CONTENT` 计算中被忽略。

可以使用 `lv_obj_add/clear_flag(obj, FLAG);` 添加/删除这些标志

Adding new layouts（添加新布局）

LVGL can be freely extended by a custom layouts like this:

LVGL 可以通过这样的自定义布局自由扩展：

```
uint32_t MY_LAYOUT;

...

MY_LAYOUT = lv_layout_register(my_layout_update, &user_data);

...

void my_layout_update(lv_obj_t * obj, void * user_data)
{
        /*Will be called automatically if required to reposition/resize the children␣
↪of "obj" */
}
```

Custom style properties can be added too that can be get and used in the update callback. For example:

也可以添加可以在更新回调中获取和使用的自定义样式属性。例如：

```
uint32_t MY_PROP;
...

LV_STYLE_MY_PROP = lv_style_register_prop();

...
static inline void lv_style_set_my_prop(lv_style_t * style, uint32_t value)
{
    lv_style_value_t v = {
        .num = (int32_t)value
    };
    lv_style_set_prop(style, LV_STYLE_MY_PROP, v);
}
```

Examples

2.5.3 Styles（风格样式）

Styles are used to set the appearance of objects. Styles in lvgl are heavily inspired by CSS. The concept in a nutshell is as follows:

- A style is an `lv_style_t` variable which can hold properties like border width, text color and so on. It's similar to a `class` in CSS.
- Styles can be assigned to objects to change their appearance. Upon assignment, the target part (*pseudo-element* in CSS) and target state (*pseudo class*) can be specified.For example one can add `style_blue` to the knob of a slider when it's in pressed state.
- The same style can be used by any number of objects.
- Styles can be cascaded which means multiple styles may be assigned to an object and each style can have different properties. Therefore, not all properties have to be specified in a style. LVGL will search for a property until a style defines it or use a default if it's not specified by any of the styles.For example `style_btn` can result in a default gray button and `style_btn_red` can add only a `background-color=red` to overwrite the background color.
- The most recently added style has higher precedence. This means if a property is specified in two styles the newest style in the object will be used.
- Some properties (e.g. text color) can be inherited from a parent(s) if it's not specified in an object.
- Objects can also have local styles with higher precedence than "normal" styles.
- Unlike CSS (where pseudo-classes describe different states, e.g. `:focus`), in LVGL a property is assigned to a given state.
- Transitions can be applied when the object changes state.

Styles 用于设置对象的外观。lvgl 中的样式很大程度上受到 CSS 的启发。简而言之，其概念如下：

- 样式是一个 `lv_style_t` 变量，它可以保存边框宽度、文本颜色等属性。它类似于 CSS 中的“类”。
- 可以将样式分配给对象以更改其外观。在赋值过程中，可以指定目标部分（CSS 中的 *pseudo element*）和目标状态（*pseudo class*）。例如，当滑块处于按下状态时，可以将“style_blue”添加到滑块的旋钮。
- 任何数量的对象都可以使用相同的样式。
- 样式可以级联，这意味着可以将多个样式分配给一个对象，并且每个样式可以具有不同的属性。因此，并非所有属性都必须在样式中指定。LVLG 将寻找一个属性，直到一个样式定义它，或者如果它没有被任何样式指定，则使用默认值。例如，`style_btn` 可以导致默认的灰色按钮，而 `style_btn_red` 只能添加一个 `background-color=red` 来覆盖背景颜色。
- 后来添加的样式具有更高的优先级。这意味着如果在两种样式中指定了一个属性，则将使用稍后添加的样式。
- 如果对象中未指定某些属性（例如文本颜色），则可以从父级继承。
- 对象可以具有比“正常”样式具有更高优先级的本地样式。
- 与 CSS（伪类描述不同的状态，例如`:focus`）不同，在 LVGL 中，属性被分配给给定的状态。
- 当对象改变状态时可以应用转换。

States（状态）

The objects can be in the combination of the following states:

- `LV_STATE_DEFAULT` (0x0000) Normal, released state
- `LV_STATE_CHECKED` (0x0001) Toggled or checked state
- `LV_STATE_FOCUSED` (0x0002) Focused via keypad or encoder or clicked via touchpad/mouse
- `LV_STATE_FOCUS_KEY` (0x0004) Focused via keypad or encoder but not via touchpad/mouse
- `LV_STATE_EDITED` (0x0008) Edit by an encoder
- `LV_STATE_HOVERED` (0x0010) Hovered by mouse (not supported now)
- `LV_STATE_PRESSED` (0x0020) Being pressed
- `LV_STATE_SCROLLED` (0x0040) Being scrolled
- `LV_STATE_DISABLED` (0x0080) Disabled state
- `LV_STATE_USER_1` (0x1000) Custom state
- `LV_STATE_USER_2` (0x2000) Custom state
- `LV_STATE_USER_3` (0x4000) Custom state
- `LV_STATE_USER_4` (0x8000) Custom state

The combination states the object can be focused and pressed at the same time. This is represented as LV_STATE_FOCUSED | LV_STATE_PRESSED.

The style can be added to any state and state combination. For example, setting a different background color for default and pressed state. If a property is not defined in a state the best matching state's property will be used. Typically this means the property with LV_STATE_DEFAULT is used.。If the property is not set even for the default state the default value will be used. (See later)

对象可以处于以下状态的组合：

- LV_STATE_DEFAULT (0x0000) 正常，释放状态
- LV_STATE_CHECKED (0x0001) 切换或检查状态
- LV_STATE_FOCUSED (0x0002) 通过键盘或编码器聚焦或通过触摸板/鼠标点击
- LV_STATE_FOCUS_KEY (0x0004) 通过键盘或编码器聚焦，但不通过触摸板/鼠标聚焦
- LV_STATE_EDITED (0x0008) 由编码器编辑
- LV_STATE_HOVERED (0x0010) 鼠标悬停（现在不支持）
- LV_STATE_PRESSED (0x0020) 被按下
- LV_STATE_SCROLLED (0x0040) 正在滚动
- LV_STATE_DISABLED (0x0080) 禁用状态
- LV_STATE_USER_1 (0x1000) 自定义状态
- LV_STATE_USER_2 (0x2000) 自定义状态
- LV_STATE_USER_3 (0x4000) 自定义状态
- LV_STATE_USER_4 (0x8000) 自定义状态

该组合表示可以同时聚焦和按下对象。这表示为 LV_STATE_FOCUSED | LV_STATE_PRESSED。

样式可以添加到任何状态和状态组合。例如，为默认和按下状态设置不同的背景颜色。如果属性未在状态中定义，则将使用 **最佳匹配状态的属性**。通常这意味着使用带有 LV_STATE_DEFAULT 的属性。如果是默认状态但是没有设置他的属性，那么将会使用默认值。（见后）

But what does the "best matching state's property" really mean? States have a precedence which is shown by their value (see in the above list). A higher value means higher precedence. To determine which state's property to use let's take an example. Imagine the background color is defined like this:

- LV_STATE_DEFAULT: white
- LV_STATE_PRESSED: gray
- LV_STATE_FOCUSED: red

1. By the default the object is in default state, so it's a simple case: the property is perfectly defined in the object's current state as white.

2. When the object is pressed there are 2 related properties: default with white (default is related to every state) and pressed with gray. The pressed state has 0x0020 precedence which is higher than the default state's 0x0000 precedence, so gray color will be used.
3. When the object is focused the same thing happens as in pressed state and red color will be used. (Focused state has higher precedence than default state).
4. When the object is focused and pressed both gray and red would work, but the pressed state has higher precedence than focused so gray color will be used.
5. It's possible to set e.g rose color for AA`LV_STATE_PRESSED | LV_STATE_FOCUSED`. In this case, this combined state has 0x0020 + 0x0002 = 0x0022 precedence, which is higher than the pressed state's precedence so rose color would be used.
6. When the object is in checked state there is no property to set the background color for this state. So for lack of a better option, the object remains white from the default state's property.

但是 **"最佳匹配状态的属性"** 到底是什么意思呢? 状态具有优先级，由它们的值显示（参见上面的列表）。更高的值意味着更高的优先级。为了确定使用哪个状态的属性，让我们举个例子。想象一下，背景颜色是这样定义的：

- `LV_STATE_DEFAULT`：白色
- `LV_STATE_PRESSED`：灰色
- `LV_STATE_FOCUSED`：红色

1. 默认情况下，对象处于默认状态，所以这是一个简单的情况：属性在对象当前状态下完美定义为白色。
2. 当对象被按下时有 2 个相关属性：默认为白色（默认与每个状态相关）和按下为灰色。按下状态的优先级为 0x0020，高于默认状态的 0x0000 优先级，因此将使用灰色。
3. 当物体聚焦时，发生与按下状态相同的事情，将使用红色。（焦点状态比默认状态具有更高的优先级）。
4. 当物体聚焦并按下时，灰色和红色都可以工作，但按下状态的优先级高于聚焦状态，因此将使用灰色。
5. 可以为 `LV_STATE_PRESSED | LV_STATE_FOCUSED` 设置例如玫瑰色。。在这种情况下，此组合状态的优先级为 0x0020 + 0x0002 = 0x0022，高于按下状态的优先级，因此将使用玫瑰色。
6. 当对象处于选中状态时，没有设置此状态的背景颜色的属性。因此，由于缺乏更好的选择，对象从默认状态的属性中保持白色。

Some practical notes:

- The precedence (value) of states is quite intuitive and it's something the user would expect naturally. E.g. if an object is focused the user will still want to see if it's pressed, therefore pressed state has a higher precedence. If the focused state had a higher precedence it would overwrite the pressed color.
- If you want to set a property for all states (e.g. red background color) just set it for the default state. If the object can't find a property for its current state it will fall back to the default state's property.
- Use ORed states to describe the properties for complex cases. (E.g. pressed + checked + focused)

- It might be a good idea to use different style elements for different states. For example, finding background colors for released, pressed, checked + pressed, focused, focused + pressed, focused + pressed + checked, etc states is quite difficult. Instead, for example, use the background color for pressed and checked states and indicate the focused state with a different border color.

一些实用的注意事项：

- 状态的优先级（值）非常直观，这是用户自然期望的。例如。如果一个对象被聚焦，用户仍然希望查看它是否被按下，因此按下状态具有更高的优先级。如果聚焦状态具有更高的优先级，它将覆盖按下的颜色。
- 如果您想为所有状态设置一个属性（例如红色背景色），只需将其设置为默认状态即可。如果对象找不到当前状态的属性，它将回退到默认状态的属性。
- 使用 ORed 状态来描述复杂情况的属性。（例如按下 + 选中 + 聚焦）
- 为不同的状态使用不同的样式元素可能是个好主意。例如，为 released、pressed、checked+pressed、focused、focused+pressed、focused+pressed+checked 等状态寻找背景颜色是相当困难的。相反，例如，对按下和选中状态使用背景颜色，并使用不同的边框颜色指示聚焦状态。

Cascading styles（层叠样式）

It's not required to set all the properties in one style. It's possible to add more styles to an object and let the later added style to modify or extend appearance. For example, create a general gray button style and create a new for red buttons where only the new background color is set.

This is much like in CSS when used classes are listed like `<div class=".btn .btn-red">`.

Styles added later have precedence over ones set earlier. So in the gray/red button example above, the normal button style should be added first and the red style second. However, the precedence coming from states are still taken into account. So let's examine the following case:

- the basic button style defines dark-gray color for default state and light-gray color pressed state
- the red button style defines the background color as red only in the default state

In this case, when the button is released (it's in default state) it will be red because a perfect match is found in the most recently added style (red). When the button is pressed the light-gray color is a better match because it describes the current state perfectly, so the button will be light-gray.

不需要在一种样式中设置所有属性。可以向对象添加更多样式，并让稍后添加的样式修改或扩展外观。例如，创建一个通用的灰色按钮样式并创建一个新的红色按钮，其中只设置了新的背景颜色。

这与 CSS 中使用的类类似，如“<div class=“.btn.btn red”>”。

后来添加的样式优先于之前设置的样式。所以在上面的灰色/红色按钮示例中，应该首先添加普通按钮样式，然后添加红色样式。然而，还要考虑“最佳匹配状态的属性“（优先级）。因此，让我们来研究以下情况：

- 基本按钮样式定义了默认状态的深灰色和浅灰色按下状态
- 红色按钮样式仅在默认状态下将背景颜色定义为红色

在这种情况下，当按钮被释放（处于默认状态）时，它将是红色的，因为在最近添加的样式（红色）中找到了最佳匹配。当按下按钮时，浅灰色更适合，因为它完美地描述了当前状态，所以按钮将是浅灰色（改变了最佳匹配状态的属性）。

Inheritance（继承）

Some properties (typically that are related to texts) can be inherited from the parent object's styles. Inheritance is applied only if the given property is not set in the object's styles (even in default state). In this case, if the property is inheritable, the property's value will be searched in the parents too until an object specifies a value for the property. The parents will use their own state to detemine the value. So if a button is pressed, and the text color comes from here, the pressed text color will be used.

某些属性（通常与文本相关）可以从父对象的样式继承。仅当未在对象的样式中设置给定属性时（即使在默认状态下），才应用继承。在这种情况下，如果该属性是可继承的，则该属性的值也将在父项中搜索，直到一个对象为该属性指定了一个值。父母将使用自己的状态来确定该值。因此，如果按下按钮，并且文本颜色来自此处，则将使用按下的文本颜色。

Parts

Objects can have *parts* which can have their own styles.

The following predefined parts exist in LVGL:

- `LV_PART_MAIN` A background like rectangle*/
- `LV_PART_SCROLLBAR` The scrollbar(s)
- `LV_PART_INDICATOR` Indicator, e.g. for slider, bar, switch, or the tick box of the checkbox
- `LV_PART_KNOB` Like a handle to grab to adjust the value*/
- `LV_PART_SELECTED` Indicate the currently selected option or section
- `LV_PART_ITEMS` Used if the widget has multiple similar elements (e.g. table cells)*/
- `LV_PART_TICKS` Ticks on scales e.g. for a chart or meter
- `LV_PART_CURSOR` Mark a specific place e.g. text area's or chart's cursor
- `LV_PART_CUSTOM_FIRST` Custom parts can be added from here.

For example a *Slider* has three parts:

- Background
- Indiactor
- Knob

It means the all three parts of the slider can have their own styles. See later how to add style styles to objects and parts.

对象可以有 部分 *(parts)* ，它们可以有自己的样式。

LVGL 中存在以下预定义部分：

- LV_PART_MAIN 类似矩形的背景 */
- LV_PART_SCROLLBAR 滚动条
- LV_PART_INDICATOR 指标，例如用于滑块、条、开关或复选框的勾选框
- LV_PART_KNOB 像手柄一样可以抓取调整值 */
- LV_PART_SELECTED 表示当前选择的选项或部分
- LV_PART_ITEMS 如果小部件具有多个相似元素（例如表格单元格）*/
- LV_PART_TICKS 刻度上的刻度，例如对于图表或仪表
- LV_PART_CURSOR 标记一个特定的地方，例如文本区域或图表的光标
- LV_PART_CUSTOM_FIRST 可以从这里添加自定义部件。

例如一个*Slider* 包含三个部分：

- 背景
- 指标
- 旋钮

这意味着滑块的所有三个部分都可以有自己的样式。稍后请参阅如何向对象和部件添加样式样式。

Initialize styles and set/get properties（初始化样式和设置/获取属性）

Styles are stored in `lv_style_t` variables. Style variables should be `static`, global or dynamically allocated. In other words they can not be local variables in functions which are destroyed when the function exists. Before using a style it should be initialized with `lv_style_init(&my_style)`. After initializing the style properties can be set or added to it.

Property set functions looks like this: `lv_style_set_<property_name>(&style, <value>);` For example:

样式存储在 `lv_style_t` 变量中。样式变量应该是 静态、全局或动态分配的。换句话说，它们不能是函数中的局部变量，当函数结束时它们会被销毁。在使用样式之前，它应该用 `lv_style_init(&my_style)` 进行初始化。初始化后，可以设置或添加样式属性。

属性集函数看起来像这样：`lv_style_set_<property_name>(&style, <value>);` 例如：

```
static lv_style_t style_btn;
lv_style_init(&style_btn);
lv_style_set_bg_color(&style_btn, lv_color_grey());
lv_style_set_bg_opa(&style_btn, LV_OPA_50);
lv_style_set_border_width(&style_btn, 2);
lv_style_set_border_color(&style_btn, lv_color_black());
```

(下页继续)

(续上页)

```
static lv_style_t style_btn_red;
lv_style_init(&style_btn_red);
lv_style_set_bg_color(&style_btn_red, lv_color_red());
lv_style_set_bg_opa(&style_btn_red, LV_OPA_COVER);
```

To remove a property use:

要删除属性，请使用：

```
lv_style_remove_prop(&style, LV_STYLE_BG_COLOR);
```

To get a property's value from a style:

从样式中获取属性的值：

```
lv_style_value_t v;
lv_res_t res = lv_style_rget_prop(&style, LV_STYLE_BG_COLOR, &v);
if(res == LV_RES_OK) {        /*Found*/
        do_something(v.color);
}
```

`lv_style_value_t` has 3 fields:

- `num` for integer, boolean and opacity properties
- `color` for color properties
- `ptr` for pointer properties

To reset a style (free all its data) use

`lv_style_value_t` 有 3 个字段：

- `num` 用于整数、布尔值和不透明度属性
- `color` 颜色属性
- `ptr` 指针属性

要重置样式（释放其所有数据），请使用

```
lv_style_reset(&style);
```

Add and remove styles to a widget（向部件添加和删除样式）

A style on its own is not that useful, it needs to be assigned to an object to take effect.

一个样式本身并没有那么有用，它需要分配给一个对象才能生效。

Add styles（添加样式）

To add a style to an object use `lv_obj_add_style(obj, &style, <selector>)`. `<selector>` is an OR-ed value of parts and state to which the style should be added. Some examples:

- `LV_PART_MAIN | LV_STATE_DEFAULT`
- `LV_STATE_PRESSED`: The main part in pressed state. `LV_PART_MAIN` can be omitted
- `LV_PART_SCROLLBAR`: The scrollbar part in the default state. `LV_STATE_DEFAULT` can be omitted.
- `LV_PART_SCROLLBAR | LV_STATE_SCROLLED`: The scrollbar part when the object is being scrolled
- `0` Same as `LV_PART_MAIN | LV_STATE_DEFAULT`.
- `LV_PART_INDICATOR | LV_STATE_PRESSED | LV_STATE_CHECKED` The indicator part when the object is pressed and checked at the same time.

Using `lv_obj_add_style`:

要向对象添加样式，请使用 `lv_obj_add_style(obj, &style, <selector>)`。`<selector>` 是应添加样式的部分和状态的 OR-ed 值，例如：

- `LV_PART_MAIN | LV_STATE_DEFAULT`
- `LV_STATE_PRESSED`：按下状态的主要部分。`LV_PART_MAIN` 可以省略
- `LV_PART_SCROLLBAR`：默认状态下的滚动条部分。`LV_STATE_DEFAULT` 可以省略。
- `LV_PART_SCROLLBAR | LV_STATE_SCROLLED`：对象滚动时的滚动条部分
- `0` 与 `LV_PART_MAIN | 相同 LV_STATE_DEFAULT`。
- `LV_PART_INDICATOR | LV_STATE_PRESSED | LV_STATE_CHECKED` 同时按下和检查对象时的指示器部分。
-

使用 `lv_obj_add_style` 的示例：

```
lv_obj_add_style(btn, &style_btn, 0);                                    /
↪*Default button style*/
lv_obj_add_style(btn, &btn_red, LV_STATE_PRESSED);  /*Overwrite only a some colors to␣
↪red when pressed*/
```

Remove styles（删除样式）

To remove all styles from an object use `lv_obj_remove_style_all(obj)`.

To remove specific styles use `lv_obj_remove_style(obj, style, selector)`. This function will remove `style` only if the `selector` matches with the `selector` used in `lv_obj_add_style`. `style` can be `NULL` to check only the `selector` and remove all matching styles. The `selector` can use the `LV_STATE_ANY` and `LV_PART_ANY` values to remove the style with any state or part.

要从对象中删除所有样式，请使用 `lv_obj_remove_style_all(obj)`。

要删除特定样式，请使用 `lv_obj_remove_style(obj, style, selector)`。

仅当 `selector` 与 `lv_obj_add_style` 中使用的 `selector` 匹配时，此函数才会删除 `style`。

`style` 可以是 `NULL` 只检查 `selector` 并删除所有匹配的样式。`selector` 可以使用 `LV_STATE_ANY` 和 `LV_PART_ANY` 值来删除具有任何状态或部分的样式。

Report style changes（通知样式更改）

If a style which is already assigned to object changes (i.e. a property is added or changed) the objects using that style should be notified. There are 3 options to do this:

1. If you know that the changed properties can be applied by a simple redraw (e.g. color or opacity changes) just call `lv_obj_invalidate(obj)` or `lv_obj_invalideate(lv_scr_act())`.
2. If more complex style properties were changed or added, and you know which object(s) are affected by that style call `lv_obj_refresh_style(obj, part, property)`. To refresh all parts and properties use `lv_obj_refresh_style(obj, LV_PART_ANY, LV_STYLE_PROP_ANY)`.
3. To make LVGL check all objects to see whether they use the style and refresh them when needed call `lv_obj_report_style_change(&style)`. If `style` is `NULL` all objects will be notified about the style change.

如果已分配给对象的样式发生更改（即添加或更改属性），则应通知使用该样式的对象。有 3 个选项可以执行此操作：

1. 如果您知道更改的属性可以通过简单的重绘（例如颜色或不透明度更改）应用，只需调用 `lv_obj_invalidate(obj)` 或 `lv_obj_invalidate(lv_scr_act())`。
2. 如果更改或添加了更复杂的样式属性，并且您知道哪些对象受该样式影响，则调用 `lv_obj_refresh_style(obj, part, property)`。要刷新所有部件和属性，请使用 `lv_obj_refresh_style(obj, LV_PART_ANY, LV_STYLE_PROP_ANY)`。
3. 要让 LVGL 检查所有对象是否使用该样式并在需要时刷新它们，请调用 `lv_obj_report_style_change(&style)`。如果 `style` 为 `NULL`，所有对象都会收到有关样式更改的通知。

Get a property's value on an object（获取对象的属性值）

To get a final value of property - considering cascading, inheritance, local styles and transitions (see below) - get functions like this can be used: `lv_obj_get_style_<property_name>(obj, <part>)`. These functions uses the object's current state and if no better candidate returns a default value. For example:

要获取属性的最终值（考虑级联、继承、本地样式和转换（请参见下文），可以使用如下的 get 函数：

`lv_obj_get_style_<property_name>(obj, <part>)`。

这些函数使用对象的当前状态，如果没有更好的候选对象，则返回默认值。例如：

```
lv_color_t color = lv_obj_get_style_bg_color(btn, LV_PART_MAIN);
```

Local styles（本地样式）

In addition to "normal" styles, objects can also store local styles. This concept is similar to inline styles in CSS (e.g. `<div style="color:red">`) with some modification.

Local styles are like normal styles, but they can't be shared among other objects. If used, local styles are allocated automatically, and freed when the object is deleted. They are useful to add local customization to an object.

Unlike in CSS, LVGL local styles can be assigned to states (*pseudo-classes*) and parts (*pseudo-elements*).

To set a local property use functions like `lv_obj_set_style_<property_name>(obj, <value>, <selector>);` For example:

除了"普通"样式，对象还可以存储本地样式。这个概念类似于 CSS 中的内联样式（例如 `<div style="color:red">`），但做了一些修改。

本地样式与普通样式类似，但不能在其他对象之间共享。本地样式会自动分配空间，并在对象被删除时释放。本地样式在向对象添加本地自定义样式时很有用。

与 CSS 不同，LVGL 本地样式可以分配给状态（*pseudo-classes*）和部件（*pseudo-elements*）。

要设置本地属性，请使用 `lv_obj_set_style_<property_name>(obj, <value>, <selector>);`

例如：

```
lv_obj_set_style_local_bg_color(slider, lv_color_red(), LV_PART_INDICATOR | LV_STATE_
↪FOCUSED);
```

Properties（属性）

For the full list of style properties click *here*.

有关样式属性的完整列表，请单击此处 查看。

Typical background properties（典型的背景属性）

In the documentation of the widgets you will see sentences like ”The widget use the typical background properties”. The ”typical background properties” are the ones related to:

- Background
- Border
- Outline
- Shadow
- Padding
- Width and height transformation
- X and Y translation

在小部件的文档中，您将看到“小部件使用典型的背景属性”这样的句子。“典型背景属性”与以下相关：

- 背景 (Background)
- 边界 (Border)
- 轮廓 (Outline)
- 阴影 (Shadow)
- 填充 (Padding)
- 宽度和高度变换
- X 和 Y 变换

Transitions（过渡特效）

By default, when an object changes state (e.g. it’s pressed) the new properties from the new state are set immediately. However, with transitions it’s possible to play an animation on state change. For example, on pressing a button its background color can be animated to the pressed color over 300 ms.

The parameters of the transitions are stored in the styles. It’s possible to set

- the time of the transition
- the delay before starting the transition
- the animation path (also known as timing or easing function)

- the properties to animate

The transition properties can be defined for each state. For example, setting 500 ms transition time in default state will mean that when the object goes to the default state a 500 ms transition time will be applied. Setting 100 ms transition time in the pressed state will mean a 100 ms transition time when going to pressed state. So this example configuration will result in going to pressed state quickly and then going back to default slowly.

To describe a transition an `lv_transition_dsc_t` variable needs to initialized and added to a style:

默认情况下，当一个对象改变状态（例如它被按下）时，新状态的新属性会立即设置。但是，通过转换，可以在状态更改时播放动画。例如，按下按钮时，其背景颜色可以在 300 毫秒内动画显示为按下的颜色。

过渡的参数存储在样式中。可以设置

- 过渡时期
- 开始过渡前的延迟
- 动画路径（也称为计时或缓动功能）
- 动画的属性

可以为每个状态定义转换属性。例如，在默认状态下设置 500 ms 转换时间意味着当对象进入默认状态时，将应用 500 ms 转换时间。在按下状态设置 100 ms 转换时间将意味着在进入按下状态时有 100 ms 转换时间。因此，此示例配置将导致快速进入按下状态，然后缓慢返回默认状态。

要描述转换，需要初始化 `lv_transition_dsc_t` 变量并将其添加到样式中：

```
/*Only its pointer is saved so must static, global or dynamically allocated */
static const lv_style_prop_t trans_props[] = {
                                                                                LV_
    LV_STYLE_BG_OPA, LV_STYLE_BG_COLOR,
                                                                                0,
    /*End marker*/
};

static lv_style_transition_dsc_t trans1;
lv_style_transition_dsc_init(&trans1, trans_props, lv_anim_path_ease_out, duration_ms,
    delay_ms);

lv_style_set_transition(&style1, &trans1);
```

Color filter（色彩过滤）

TODO

Themes（主题）

Themes are a collection of styles. If there is an active theme LVGL applies it on every created widget. This will give a default appearance to the UI which can then be modified by adding further styles.

Every display can have a different theme. For example you could have a colorful theme on a TFT and monochrome theme on a secondary monochrome display.

To set a theme for a display, 2 steps are required:

1. Initialize a theme
2. Assign the initialized theme to a display.

Theme initialization functions can have different prototype. This example shows how to set the ”default” theme:

主题是风格的集合。如果存在活动主题，LVGL 将其应用于每个创建的部件 (对象)。这将为 UI 提供一个默认外观，然后可以通过添加更多样式对其进行修改。

每个显示器都可以有不同的主题。例如，您可以在 TFT 上使用彩色主题，在辅助单色显示器上使用单色主题。

要为显示设置主题，需要 2 个步骤：

1. 初始化一个主题
2. 将初始化的主题分配给显示器。

主题初始化函数可以有不同的原型。此示例显示如何设置“默认”主题：

```
lv_theme_t * th = lv_theme_default_init(display,  /*Use the DPI, size, etc from this␣
↪display*/
                                        LV_COLOR_PALETTE_BLUE, LV_COLOR_PALETTE_CYAN,␣
↪  /*Primary and secondary palette*/
                                        false,    /*Light or dark mode*/
                                        &lv_font_montserrat_10, &lv_font_montserrat_
↪14, &lv_font_montserrat_18); /*Small, normal, large fonts*/

lv_disp_set_theme(display, th); /*Assign the theme to the display*/
```

The themes can be enabled in `lv_conf.h`. If the default theme is enabled by `LV_USE_THEME_DEFAULT 1` LVGL automatically initializes and sets it when a display is created.

可以在 `lv_conf.h` 中启用主题。如果默认主题由 `LV_USE_THEME_DEFAULT 1` 启用，LVGL 会在创建显示时自动初始化并设置它。

Extending themes（扩展主题）

Built-in themes can be extended. If a custom theme is created a parent theme can be selected. The parent theme's styles will be added before the custom theme's styles. Any number of themes can be chained this way. E.g. default theme -> custom theme -> dark theme.

`lv_theme_set_parent(new_theme, base_theme)` extends the `base_theme` with the `new_theme`.

There is an example for it below.

内置主题可以扩展。如果创建了自定义主题，则可以选择父主题。父主题的样式将添加在自定义主题的样式之前。可以通过这种方式链接任意数量的主题。例如。默认主题 -> 自定义主题 -> 深色主题。

`lv_theme_set_parent(new_theme, base_theme)` 使用 `new_theme` 扩展了 `base_theme`。

下面的是示例：

Examples

Size styles

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_IMG

/**
 * Using the Size, Position and Padding style properties
 */
void lv_example_style_1(void)
{
     static lv_style_t style;
     lv_style_init(&style);
     lv_style_set_radius(&style, 5);

     /*Make a gradient*/
     lv_style_set_width(&style, 150);
     lv_style_set_height(&style, LV_SIZE_CONTENT);

     lv_style_set_pad_ver(&style, 20);
     lv_style_set_pad_left(&style, 5);

     lv_style_set_x(&style, lv_pct(50));
     lv_style_set_y(&style, 80);

     /*Create an object with the new style*/
     lv_obj_t * obj = lv_obj_create(lv_scr_act());
```

(下页继续)

(续上页)

```
    lv_obj_add_style(obj, &style, 0);

    lv_obj_t * label = lv_label_create(obj);
    lv_label_set_text(label, "Hello");
}

#endif
```

```
#
# Using the Size, Position and Padding style properties
#
style = lv.style_t()
style.init()
style.set_radius(5)

# Make a gradient
style.set_width(150)
style.set_height(lv.SIZE.CONTENT)

style.set_pad_ver(20)
style.set_pad_left(5)

style.set_x(lv.pct(50))
style.set_y(80)

# Create an object with the new style
obj = lv.obj(lv.scr_act())
obj.add_style(style, 0)

label = lv.label(obj)
label.set_text("Hello")
```

Background styles

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

/**
 * Using the background style properties
 */
void lv_example_style_2(void)
```

(下页继续)

(续上页)

```
{
    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_radius(&style, 5);

    /*Make a gradient*/
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 1));
    lv_style_set_bg_grad_color(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_bg_grad_dir(&style, LV_GRAD_DIR_VER);

    /*Shift the gradient to the bottom*/
    lv_style_set_bg_main_stop(&style, 128);
    lv_style_set_bg_grad_stop(&style, 192);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_scr_act());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_center(obj);
}

#endif
```

```
#
# Using the background style properties
#
style = lv.style_t()
style.init()
style.set_radius(5)

# Make a gradient
style.set_bg_opa(lv.OPA.COVER)
style.set_bg_color(lv.palette_lighten(lv.PALETTE.GREY, 1))
style.set_bg_grad_color(lv.palette_main(lv.PALETTE.BLUE))
style.set_bg_grad_dir(lv.GRAD_DIR.VER)

# Shift the gradient to the bottom
style.set_bg_main_stop(128)
style.set_bg_grad_stop(192)

# Create an object with the new style
obj = lv.obj(lv.scr_act())
obj.add_style(style, 0)
```

(下页继续)

(续上页)

```
obj.center()
```

Border styles

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

/**
 * Using the border style properties
 */
void lv_example_style_3(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    /*Set a background color and a radius*/
    lv_style_set_radius(&style, 10);
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 1));

    /*Add border to the bottom+right*/
    lv_style_set_border_color(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_border_width(&style, 5);
    lv_style_set_border_opa(&style, LV_OPA_50);
    lv_style_set_border_side(&style, LV_BORDER_SIDE_BOTTOM | LV_BORDER_SIDE_RIGHT);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_scr_act());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_center(obj);
}

#endif
```

```
#
# Using the border style properties
#
style = lv.style_t()
style.init()

# Set a background color and a radius
style.set_radius(10)
```

(下页继续)

(续上页)

```
style.set_bg_opa(lv.OPA.COVER)
style.set_bg_color(lv.palette_lighten(lv.PALETTE.GREY, 1))

# Add border to the bottom+right
style.set_border_color(lv.palette_main(lv.PALETTE.BLUE))
style.set_border_width(5)
style.set_border_opa(lv.OPA._50)
style.set_border_side(lv.BORDER_SIDE.BOTTOM | lv.BORDER_SIDE.RIGHT)

# Create an object with the new style
obj = lv.obj(lv.scr_act())
obj.add_style(style, 0)
obj.center()
```

Outline styles

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

/**
 * Using the outline style properties
 */
void lv_example_style_4(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    /*Set a background color and a radius*/
    lv_style_set_radius(&style, 5);
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 1));

    /*Add outline*/
    lv_style_set_outline_width(&style, 2);
    lv_style_set_outline_color(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_outline_pad(&style, 8);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_scr_act());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_center(obj);
}
```

(下页继续)

(续上页)

```
#endif
```

```
#
# Using the outline style properties
#

style = lv.style_t()
style.init()

# Set a background color and a radius
style.set_radius(5)
style.set_bg_opa(lv.OPA.COVER)
style.set_bg_color(lv.palette_lighten(lv.PALETTE.GREY, 1))

# Add outline
style.set_outline_width(2)
style.set_outline_color(lv.palette_main(lv.PALETTE.BLUE))
style.set_outline_pad(8)

# Create an object with the new style
obj = lv.obj(lv.scr_act())
obj.add_style(style, 0)
obj.center()
```

Shadow styles

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

/**
 * Using the Shadow style properties
 */
void lv_example_style_5(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    /*Set a background color and a radius*/
    lv_style_set_radius(&style, 5);
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
```

(下页继续)

(续上页)

```
    lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 1));

    /*Add a shadow*/
    lv_style_set_shadow_width(&style, 55);
    lv_style_set_shadow_color(&style, lv_palette_main(LV_PALETTE_BLUE));
//    lv_style_set_shadow_ofs_x(&style, 10);
//    lv_style_set_shadow_ofs_y(&style, 20);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_scr_act());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_center(obj);
}

#endif
```

```
#
# Using the Shadow style properties
#

style = lv.style_t()
style.init()

# Set a background color and a radius
style.set_radius(5)
style.set_bg_opa(lv.OPA.COVER)
style.set_bg_color(lv.palette_lighten(lv.PALETTE.GREY, 1))

# Add a shadow
style.set_shadow_width(8)
style.set_shadow_color(lv.palette_main(lv.PALETTE.BLUE))
style.set_shadow_ofs_x(10)
style.set_shadow_ofs_y(20)

# Create an object with the new style
obj = lv.obj(lv.scr_act())
obj.add_style(style, 0)
obj.center()
```

Image styles

```c
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_IMG

/**
 * Using the Image style properties
 */
void lv_example_style_6(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    /*Set a background color and a radius*/
    lv_style_set_radius(&style, 5);
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 3));
    lv_style_set_border_width(&style, 2);
    lv_style_set_border_color(&style, lv_palette_main(LV_PALETTE_BLUE));

    lv_style_set_img_recolor(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_img_recolor_opa(&style, LV_OPA_50);
    lv_style_set_transform_angle(&style, 300);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_img_create(lv_scr_act());
    lv_obj_add_style(obj, &style, 0);

    LV_IMG_DECLARE(img_cogwheel_argb);
    lv_img_set_src(obj, &img_cogwheel_argb);

    lv_obj_center(obj);
}

#endif
```

```python
from imagetools import get_png_info, open_png
# Register PNG image decoder
decoder = lv.img.decoder_create()
decoder.info_cb = get_png_info
decoder.open_cb = open_png

# Create an image from the png file
try:
```

(下页继续)

(续上页)

```
    with open('../assets/img_cogwheel_argb.png', 'rb') as f:
        png_data = f.read()
except:
    print("Could not find img_cogwheel_argb.png")
    sys.exit()

img_cogwheel_argb = lv.img_dsc_t({
  'data_size': len(png_data),
  'data': png_data
})

#
# Using the Image style properties
#
style = lv.style_t()
style.init()

# Set a background color and a radius
style.set_radius(5)
style.set_bg_opa(lv.OPA.COVER)
style.set_bg_color(lv.palette_lighten(lv.PALETTE.GREY, 3))
style.set_border_width(2)
style.set_border_color(lv.palette_main(lv.PALETTE.BLUE))

style.set_img_recolor(lv.palette_main(lv.PALETTE.BLUE))
style.set_img_recolor_opa(lv.OPA._50)
# style.set_transform_angle(300)

# Create an object with the new style
obj = lv.img(lv.scr_act())
obj.add_style(style, 0)

obj.set_src(img_cogwheel_argb)

obj.center()
```

Arc styles

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs/100ask_lvgl_
↪docs/examples/style/lv_example_style_7.c
```

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs/100ask_lvgl_
↪docs/examples/style/lv_example_style_7.py
```

Text styles

```c
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_LABEL

/**
 * Using the text style properties
 */
void lv_example_style_8(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    lv_style_set_radius(&style, 5);
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 2));
    lv_style_set_border_width(&style, 2);
    lv_style_set_border_color(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_pad_all(&style, 10);

    lv_style_set_text_color(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_text_letter_space(&style, 5);
    lv_style_set_text_line_space(&style, 20);
    lv_style_set_text_decor(&style, LV_TEXT_DECOR_UNDERLINE);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_label_create(lv_scr_act());
    lv_obj_add_style(obj, &style, 0);
    lv_label_set_text(obj, "Text of\n"
                           "a label");

    lv_obj_center(obj);
}
```

(下页继续)

(续上页)

```
#endif
```

```
#
# Using the text style properties
#

style = lv.style_t()
style.init()

style.set_radius(5)
style.set_bg_opa(lv.OPA.COVER)
style.set_bg_color(lv.palette_lighten(lv.PALETTE.GREY, 3))
style.set_border_width(2)
style.set_border_color(lv.palette_main(lv.PALETTE.BLUE))
style.set_pad_all(10)

style.set_text_color(lv.palette_main(lv.PALETTE.BLUE))
style.set_text_letter_space(5)
style.set_text_line_space(20)
style.set_text_decor(lv.TEXT_DECOR.UNDERLINE)

# Create an object with the new style
obj = lv.label(lv.scr_act())
obj.add_style(style, 0)
obj.set_text("Text of\n"
             "a label")

obj.center()
```

Line styles

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_LINE

/**
 * Using the line style properties
 */
void lv_example_style_9(void)
{
    static lv_style_t style;
    lv_style_init(&style);
```

(下页继续)

(续上页)

```
    lv_style_set_line_color(&style, lv_palette_main(LV_PALETTE_GREY));
    lv_style_set_line_width(&style, 6);
    lv_style_set_line_rounded(&style, true);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_line_create(lv_scr_act());
    lv_obj_add_style(obj, &style, 0);

    static lv_point_t p[] = {{10, 30}, {30, 50}, {100, 0}};
    lv_line_set_points(obj, p, 3);

    lv_obj_center(obj);
}

#endif
```

```
#
# Using the line style properties
#

style = lv.style_t()
style.init()

style.set_line_color(lv.palette_main(lv.PALETTE.GREY))
style.set_line_width(6)
style.set_line_rounded(True)

# Create an object with the new style
obj = lv.line(lv.scr_act())
obj.add_style(style, 0)
p =  [ {"x":10, "y":30},
       {"x":30, "y":50},
       {"x":100, "y":0}]

obj.set_points(p, 3)

obj.center()
```

Transition

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_IMG

/**
 * Creating a transition
 */
void lv_example_style_10(void)
{
    static const lv_style_prop_t props[] = {LV_STYLE_BG_COLOR, LV_STYLE_BORDER_COLOR,␣
↪LV_STYLE_BORDER_WIDTH, 0};

    /* A default transition
     * Make it fast (100ms) and start with some delay (200 ms)*/
    static lv_style_transition_dsc_t trans_def;
    lv_style_transition_dsc_init(&trans_def, props, lv_anim_path_linear, 100, 200,␣
↪NULL);

    /* A special transition when going to pressed state
     * Make it slow (500 ms) but start  without delay*/
    static lv_style_transition_dsc_t trans_pr;
    lv_style_transition_dsc_init(&trans_pr, props, lv_anim_path_linear, 500, 0, NULL);

    static lv_style_t style_def;
    lv_style_init(&style_def);
    lv_style_set_transition(&style_def, &trans_def);

    static lv_style_t style_pr;
    lv_style_init(&style_pr);
    lv_style_set_bg_color(&style_pr, lv_palette_main(LV_PALETTE_RED));
    lv_style_set_border_width(&style_pr, 6);
    lv_style_set_border_color(&style_pr, lv_palette_darken(LV_PALETTE_RED, 3));
    lv_style_set_transition(&style_pr, &trans_pr);

    /*Create an object with the new style_pr*/
    lv_obj_t * obj = lv_obj_create(lv_scr_act());
    lv_obj_add_style(obj, &style_def, 0);
    lv_obj_add_style(obj, &style_pr, LV_STATE_PRESSED);

    lv_obj_center(obj);
}

#endif
```

```
#
# Creating a transition
#

props = [lv.STYLE.BG_COLOR, lv.STYLE.BORDER_COLOR, lv.STYLE.BORDER_WIDTH, 0]

# A default transition
# Make it fast (100ms) and start with some delay (200 ms)

trans_def =  lv.style_transition_dsc_t()
trans_def.init(props, lv.anim_t.path_linear, 100, 200, None)

# A special transition when going to pressed state
# Make it slow (500 ms) but start  without delay

trans_pr = lv.style_transition_dsc_t()
trans_pr.init(props, lv.anim_t.path_linear, 500, 0, None)

style_def = lv.style_t()
style_def.init()
style_def.set_transition(trans_def)

style_pr = lv.style_t()
style_pr.init()
style_pr.set_bg_color(lv.palette_main(lv.PALETTE.RED))
style_pr.set_border_width(6)
style_pr.set_border_color(lv.palette_darken(lv.PALETTE.RED, 3))
style_pr.set_transition(trans_pr)

# Create an object with the new style_pr
obj = lv.obj(lv.scr_act())
obj.add_style(style_def, 0)
obj.add_style(style_pr, lv.STATE.PRESSED)

obj.center()
```

Using multiple styles

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_IMG

/**
 * Using multiple styles
 */
void lv_example_style_11(void)
{
    /*A base style*/
    static lv_style_t style_base;
    lv_style_init(&style_base);
    lv_style_set_bg_color(&style_base, lv_palette_main(LV_PALETTE_LIGHT_BLUE));
    lv_style_set_border_color(&style_base, lv_palette_darken(LV_PALETTE_LIGHT_BLUE,␣
↪3));
    lv_style_set_border_width(&style_base, 2);
    lv_style_set_radius(&style_base, 10);
    lv_style_set_shadow_width(&style_base, 10);
    lv_style_set_shadow_ofs_y(&style_base, 5);
    lv_style_set_shadow_opa(&style_base, LV_OPA_50);
    lv_style_set_text_color(&style_base, lv_color_white());
    lv_style_set_width(&style_base, 100);
    lv_style_set_height(&style_base, LV_SIZE_CONTENT);

    /*Set only the properties that should be different*/
    static lv_style_t style_warning;
    lv_style_init(&style_warning);
    lv_style_set_bg_color(&style_warning, lv_palette_main(LV_PALETTE_YELLOW));
    lv_style_set_border_color(&style_warning, lv_palette_darken(LV_PALETTE_YELLOW,␣
↪3));
    lv_style_set_text_color(&style_warning, lv_palette_darken(LV_PALETTE_YELLOW, 4));

    /*Create an object with the base style only*/
    lv_obj_t * obj_base = lv_obj_create(lv_scr_act());
    lv_obj_add_style(obj_base, &style_base, 0);
    lv_obj_align(obj_base, LV_ALIGN_LEFT_MID, 20, 0);

    lv_obj_t * label = lv_label_create(obj_base);
    lv_label_set_text(label, "Base");
    lv_obj_center(label);

    /*Create an other object with the base style and earnings style too*/
    lv_obj_t * obj_warning = lv_obj_create(lv_scr_act());
```

(下页继续)

(续上页)

```
    lv_obj_add_style(obj_warning, &style_base, 0);
    lv_obj_add_style(obj_warning, &style_warning, 0);
    lv_obj_align(obj_warning, LV_ALIGN_RIGHT_MID, -20, 0);

    label = lv_label_create(obj_warning);
    lv_label_set_text(label, "Warning");
    lv_obj_center(label);
}

#endif
```

```
#
# Using multiple styles
#
# A base style

style_base = lv.style_t()
style_base.init()
style_base.set_bg_color(lv.palette_main(lv.PALETTE.LIGHT_BLUE))
style_base.set_border_color(lv.palette_darken(lv.PALETTE.LIGHT_BLUE, 3))
style_base.set_border_width(2)
style_base.set_radius(10)
style_base.set_shadow_width(10)
style_base.set_shadow_ofs_y(5)
style_base.set_shadow_opa(lv.OPA._50)
style_base.set_text_color(lv.color_white())
style_base.set_width(100)
style_base.set_height(lv.SIZE.CONTENT)

# Set only the properties that should be different
style_warning = lv.style_t()
style_warning.init()
style_warning.set_bg_color(lv.palette_main(lv.PALETTE.YELLOW))
style_warning.set_border_color(lv.palette_darken(lv.PALETTE.YELLOW, 3))
style_warning.set_text_color(lv.palette_darken(lv.PALETTE.YELLOW, 4))

# Create an object with the base style only
obj_base = lv.obj(lv.scr_act())
obj_base.add_style(style_base, 0)
obj_base.align(lv.ALIGN.LEFT_MID, 20, 0)

label = lv.label(obj_base)
label.set_text("Base")
```

(下页继续)

(续上页)

```
label.center()

# Create an other object with the base style and earnings style too
obj_warning = lv.obj(lv.scr_act())
obj_warning.add_style(style_base, 0)
obj_warning.add_style(style_warning, 0)
obj_warning.align(lv.ALIGN.RIGHT_MID, -20, 0)

label = lv.label(obj_warning)
label.set_text("Warning")
label.center()
```

Local styles

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_IMG

/**
 * Local styles
 */
void lv_example_style_12(void)
{
    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_bg_color(&style, lv_palette_main(LV_PALETTE_GREEN));
    lv_style_set_border_color(&style, lv_palette_lighten(LV_PALETTE_GREEN, 3));
    lv_style_set_border_width(&style, 3);

    lv_obj_t * obj = lv_obj_create(lv_scr_act());
    lv_obj_add_style(obj, &style, 0);

    /*Overwrite the background color locally*/
    lv_obj_set_style_bg_color(obj,lv_palette_main(LV_PALETTE_ORANGE), LV_PART_MAIN);

    lv_obj_center(obj);
}

#endif
```

```
#
# Local styles
#
```

(下页继续)

(续上页)

```
style = lv.style_t()
style.init()
style.set_bg_color(lv.palette_main(lv.PALETTE.GREEN))
style.set_border_color(lv.palette_lighten(lv.PALETTE.GREEN, 3))
style.set_border_width(3)

obj = lv.obj(lv.scr_act())
obj.add_style(style, 0)

# Overwrite the background color locally
obj.set_style_bg_color(lv.palette_main(lv.PALETTE.ORANGE), lv.PART.MAIN)

obj.center()
```

Add styles to parts and states

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_IMG

/**
 * Add styles to parts and states
 */
void lv_example_style_13(void)
{
    static lv_style_t style_indic;
    lv_style_init(&style_indic);
    lv_style_set_bg_color(&style_indic, lv_palette_lighten(LV_PALETTE_RED, 3));
    lv_style_set_bg_grad_color(&style_indic, lv_palette_main(LV_PALETTE_RED));
    lv_style_set_bg_grad_dir(&style_indic, LV_GRAD_DIR_HOR);

    static lv_style_t style_indic_pr;
    lv_style_init(&style_indic_pr);
    lv_style_set_shadow_color(&style_indic_pr, lv_palette_main(LV_PALETTE_RED));
    lv_style_set_shadow_width(&style_indic_pr, 10);
    lv_style_set_shadow_spread(&style_indic_pr, 3);

    /*Create an object with the new style_pr*/
    lv_obj_t * obj = lv_slider_create(lv_scr_act());
    lv_obj_add_style(obj, &style_indic, LV_PART_INDICATOR);
    lv_obj_add_style(obj, &style_indic_pr, LV_PART_INDICATOR | LV_STATE_PRESSED);
    lv_slider_set_value(obj, 70, LV_ANIM_OFF);
```

(下页继续)

(续上页)

```
    lv_obj_center(obj);
}

#endif
```

```
#
# Add styles to parts and states
#

style_indic = lv.style_t()
style_indic.init()
style_indic.set_bg_color(lv.palette_lighten(lv.PALETTE.RED, 3))
style_indic.set_bg_grad_color(lv.palette_main(lv.PALETTE.RED))
style_indic.set_bg_grad_dir(lv.GRAD_DIR.HOR)

style_indic_pr = lv.style_t()
style_indic_pr.init()
style_indic_pr.set_shadow_color(lv.palette_main(lv.PALETTE.RED))
style_indic_pr.set_shadow_width(10)
style_indic_pr.set_shadow_spread(3)

# Create an object with the new style_pr
obj = lv.slider(lv.scr_act())
obj.add_style(style_indic, lv.PART.INDICATOR)
obj.add_style(style_indic_pr, lv.PART.INDICATOR | lv.STATE.PRESSED)
obj.set_value(70, lv.ANIM.OFF)
obj.center()
```

Extending the current theme

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_IMG


static lv_style_t style_btn;

/*Will be called when the styles of the base theme are already added
  to add new styles*/
static void new_theme_apply_cb(lv_theme_t * th, lv_obj_t * obj)
{
    LV_UNUSED(th);
```

(下页继续)

(续上页)

```
    if(lv_obj_check_type(obj, &lv_btn_class)) {
        lv_obj_add_style(obj, &style_btn, 0);
    }
}

static void new_theme_init_and_set(void)
{
    /*Initialize the styles*/
    lv_style_init(&style_btn);
    lv_style_set_bg_color(&style_btn, lv_palette_main(LV_PALETTE_GREEN));
    lv_style_set_border_color(&style_btn, lv_palette_darken(LV_PALETTE_GREEN, 3));
    lv_style_set_border_width(&style_btn, 3);

    /*Initialize the new theme from the current theme*/
    lv_theme_t * th_act = lv_disp_get_theme(NULL);
    static lv_theme_t th_new;
    th_new = *th_act;

    /*Set the parent theme and the style apply callback for the new theme*/
    lv_theme_set_parent(&th_new, th_act);
    lv_theme_set_apply_cb(&th_new, new_theme_apply_cb);

    /*Assign the new theme the the current display*/
    lv_disp_set_theme(NULL, &th_new);
}



/**
 * Extending the current theme
 */
void lv_example_style_14(void)
{
    lv_obj_t * btn;
    lv_obj_t * label;

    btn = lv_btn_create(lv_scr_act());
    lv_obj_align(btn, LV_ALIGN_TOP_MID, 0, 20);

    label = lv_label_create(btn);
    lv_label_set_text(label, "Original theme");

    new_theme_init_and_set();
```

(下页继续)

(续上页)

```
    btn = lv_btn_create(lv_scr_act());
    lv_obj_align(btn, LV_ALIGN_BOTTOM_MID, 0, -20);

    label = lv_label_create(btn);
    lv_label_set_text(label, "New theme");
}

#endif
```

```
# Will be called when the styles of the base theme are already added
# to add new styles


class NewTheme(lv.theme_t):
    def __init__(self):
        super().__init__()
        # Initialize the styles
        self.style_btn = lv.style_t()
        self.style_btn.init()
        self.style_btn.set_bg_color(lv.palette_main(lv.PALETTE.GREEN))
        self.style_btn.set_border_color(lv.palette_darken(lv.PALETTE.GREEN, 3))
        self.style_btn.set_border_width(3)

        # This theme is based on active theme
        th_act = lv.theme_get_from_obj(lv.scr_act())
        # This theme will be applied only after base theme is applied
        self.set_parent(th_act)


class ExampleStyle_14:

    def __init__(self):
        #
        # Extending the current theme
        #

        btn = lv.btn(lv.scr_act())
        btn.align(lv.ALIGN.TOP_MID, 0, 20)

        label = lv.label(btn)
        label.set_text("Original theme")
```

(下页继续)

(续上页)

```
        self.new_theme_init_and_set()

        btn = lv.btn(lv.scr_act())
        btn.align(lv.ALIGN.BOTTOM_MID, 0, -20)

        label = lv.label(btn)
        label.set_text("New theme")

    def new_theme_apply_cb(self, th, obj):
        print(th,obj)
        if obj.get_class() == lv.btn_class:
            obj.add_style(self.th_new.style_btn, 0)

    def new_theme_init_and_set(self):
        print("new_theme_init_and_set")
        # Initialize the new theme from the current theme
        self.th_new = NewTheme()
        self.th_new.set_apply_cb(self.new_theme_apply_cb)
        lv.disp_get_default().set_theme(self.th_new)


exampleStyle_14 = ExampleStyle_14()
```

API

Typedefs

typedef uint8_t **lv_blend_mode_t**

typedef uint8_t **lv_text_decor_t**

typedef uint8_t **lv_border_side_t**

typedef uint8_t **lv_grad_dir_t**

typedef uint8_t **lv_dither_mode_t**

Enums

enum **[anonymous]**
Possible options how to blend opaque drawings

Values:

enumerator **LV_BLEND_MODE_NORMAL**
Simply mix according to the opacity value

enumerator **LV_BLEND_MODE_ADDITIVE**
Add the respective color channels

enumerator **LV_BLEND_MODE_SUBTRACTIVE**
Subtract the foreground from the background

enumerator **LV_BLEND_MODE_MULTIPLY**
Multiply the foreground and background

enumerator **LV_BLEND_MODE_REPLACE**
Replace background with foreground in the area

enum **[anonymous]**
Some options to apply decorations on texts. 'OR'ed values can be used.

Values:

enumerator **LV_TEXT_DECOR_NONE**

enumerator **LV_TEXT_DECOR_UNDERLINE**

enumerator **LV_TEXT_DECOR_STRIKETHROUGH**

enum **[anonymous]**
Selects on which sides border should be drawn 'OR'ed values can be used.

Values:

enumerator **LV_BORDER_SIDE_NONE**

enumerator **LV_BORDER_SIDE_BOTTOM**

enumerator **LV_BORDER_SIDE_TOP**

enumerator **LV_BORDER_SIDE_LEFT**

enumerator **LV_BORDER_SIDE_RIGHT**

enumerator **LV_BORDER_SIDE_FULL**

enumerator **LV_BORDER_SIDE_INTERNAL**
FOR matrix-like objects (e.g. Button matrix)

enum **[anonymous]**
The direction of the gradient.

Values:

enumerator **LV_GRAD_DIR_NONE**
No gradient (the `grad_color` property is ignored)

enumerator **LV_GRAD_DIR_VER**
Vertical (top to bottom) gradient

enumerator **LV_GRAD_DIR_HOR**
Horizontal (left to right) gradient

enum **[anonymous]**
The dithering algorithm for the gradient Depends on LV_DITHER_GRADIENT

Values:

enumerator **LV_DITHER_NONE**
No dithering, colors are just quantized to the output resolution

enumerator **LV_DITHER_ORDERED**
Ordered dithering. Faster to compute and use less memory but lower quality

enumerator **LV_DITHER_ERR_DIFF**
Error diffusion mode. Slower to compute and use more memory but give highest dither quality

enum **lv_style_prop_t**
Enumeration of all built in style properties

Values:

enumerator **LV_STYLE_PROP_INV**

enumerator **LV_STYLE_WIDTH**

enumerator **LV_STYLE_MIN_WIDTH**

enumerator **LV_STYLE_MAX_WIDTH**

enumerator **LV_STYLE_HEIGHT**

enumerator **LV_STYLE_MIN_HEIGHT**

enumerator **LV_STYLE_MAX_HEIGHT**

enumerator **LV_STYLE_X**

enumerator **LV_STYLE_Y**

enumerator **LV_STYLE_ALIGN**

enumerator **LV_STYLE_TRANSFORM_WIDTH**

enumerator **LV_STYLE_TRANSFORM_HEIGHT**

enumerator **LV_STYLE_TRANSLATE_X**

enumerator **LV_STYLE_TRANSLATE_Y**

enumerator **LV_STYLE_TRANSFORM_ZOOM**

enumerator **LV_STYLE_TRANSFORM_ANGLE**

enumerator **LV_STYLE_PAD_TOP**

enumerator **LV_STYLE_PAD_BOTTOM**

enumerator **LV_STYLE_PAD_LEFT**

enumerator **LV_STYLE_PAD_RIGHT**

enumerator **LV_STYLE_PAD_ROW**

enumerator **LV_STYLE_PAD_COLUMN**

enumerator **LV_STYLE_BG_COLOR**

enumerator **LV_STYLE_BG_COLOR_FILTERED**

enumerator **LV_STYLE_BG_OPA**

enumerator **LV_STYLE_BG_GRAD_COLOR**

enumerator **LV_STYLE_BG_GRAD_COLOR_FILTERED**

enumerator **LV_STYLE_BG_GRAD_DIR**

enumerator **LV_STYLE_BG_MAIN_STOP**

enumerator **LV_STYLE_BG_GRAD_STOP**

enumerator **LV_STYLE_BG_GRAD**

enumerator **LV_STYLE_BG_DITHER_MODE**

enumerator **LV_STYLE_BG_IMG_SRC**

enumerator **LV_STYLE_BG_IMG_OPA**

enumerator **LV_STYLE_BG_IMG_RECOLOR**

enumerator **LV_STYLE_BG_IMG_RECOLOR_FILTERED**

enumerator **LV_STYLE_BG_IMG_RECOLOR_OPA**

enumerator **LV_STYLE_BG_IMG_TILED**

enumerator **LV_STYLE_BORDER_COLOR**

enumerator **LV_STYLE_BORDER_COLOR_FILTERED**

enumerator **LV_STYLE_BORDER_OPA**

enumerator **LV_STYLE_BORDER_WIDTH**

enumerator **LV_STYLE_BORDER_SIDE**

enumerator **LV_STYLE_BORDER_POST**

enumerator **LV_STYLE_OUTLINE_WIDTH**

enumerator **LV_STYLE_OUTLINE_COLOR**

enumerator **LV_STYLE_OUTLINE_COLOR_FILTERED**

enumerator **LV_STYLE_OUTLINE_OPA**

enumerator **LV_STYLE_OUTLINE_PAD**

enumerator **LV_STYLE_SHADOW_WIDTH**

enumerator **LV_STYLE_SHADOW_OFS_X**

enumerator **LV_STYLE_SHADOW_OFS_Y**

enumerator **LV_STYLE_SHADOW_SPREAD**

enumerator **LV_STYLE_SHADOW_COLOR**

enumerator **LV_STYLE_SHADOW_COLOR_FILTERED**

enumerator **LV_STYLE_SHADOW_OPA**

enumerator **LV_STYLE_IMG_OPA**

enumerator **LV_STYLE_IMG_RECOLOR**

enumerator **LV_STYLE_IMG_RECOLOR_FILTERED**

enumerator **LV_STYLE_IMG_RECOLOR_OPA**

enumerator **LV_STYLE_LINE_WIDTH**

enumerator **LV_STYLE_LINE_DASH_WIDTH**

enumerator **LV_STYLE_LINE_DASH_GAP**

enumerator **LV_STYLE_LINE_ROUNDED**

enumerator **LV_STYLE_LINE_COLOR**

enumerator **LV_STYLE_LINE_COLOR_FILTERED**

enumerator **LV_STYLE_LINE_OPA**

enumerator **LV_STYLE_ARC_WIDTH**

enumerator **LV_STYLE_ARC_ROUNDED**

enumerator **LV_STYLE_ARC_COLOR**

enumerator **LV_STYLE_ARC_COLOR_FILTERED**

enumerator **LV_STYLE_ARC_OPA**

enumerator **LV_STYLE_ARC_IMG_SRC**

enumerator **LV_STYLE_TEXT_COLOR**

enumerator **LV_STYLE_TEXT_COLOR_FILTERED**

enumerator **LV_STYLE_TEXT_OPA**

enumerator **LV_STYLE_TEXT_FONT**

enumerator **LV_STYLE_TEXT_LETTER_SPACE**

enumerator **LV_STYLE_TEXT_LINE_SPACE**

enumerator **LV_STYLE_TEXT_DECOR**

enumerator **LV_STYLE_TEXT_ALIGN**

enumerator **LV_STYLE_RADIUS**

enumerator **LV_STYLE_CLIP_CORNER**

enumerator **LV_STYLE_OPA**

enumerator **LV_STYLE_COLOR_FILTER_DSC**

enumerator **LV_STYLE_COLOR_FILTER_OPA**

enumerator **LV_STYLE_ANIM_TIME**

enumerator **LV_STYLE_ANIM_SPEED**

enumerator **LV_STYLE_TRANSITION**

enumerator **LV_STYLE_BLEND_MODE**

enumerator **LV_STYLE_LAYOUT**

enumerator **LV_STYLE_BASE_DIR**

enumerator **_LV_STYLE_LAST_BUILT_IN_PROP**

enumerator **LV_STYLE_PROP_ANY**

Functions

LV_EXPORT_CONST_INT(LV_IMG_ZOOM_NONE)

void **lv_style_init**(*lv_style_t* *style)

Initialize a style

注解： Do not call `lv_style_init` on styles that already have some properties because this function won't free the used memory, just sets a default state for the style. In other words be sure to initialize styles only once!

参数 **style** -- pointer to a style to initialize

void **lv_style_reset**(*lv_style_t* *style)

Clear all properties from a style and free all allocated memories.

参数 **style** -- pointer to a style

lv_style_prop_t **lv_style_register_prop**(void)

bool **lv_style_remove_prop**(*lv_style_t* *style, *lv_style_prop_t* prop)

Remove a property from a style

参数

- **style** -- pointer to a style
- **prop** -- a style property ORed with a state.

返回 true: the property was found and removed; false: the property wasn't found

void **lv_style_set_prop**(*lv_style_t* *style, *lv_style_prop_t* prop, *lv_style_value_t* value)

Set the value of property in a style. This function shouldn't be used directly by the user. Instead use `lv_style_set_<prop_name>()`. E.g. `lv_style_set_bg_color()`

参数

- **style** -- pointer to style
- **prop** -- the ID of a property (e.g. `LV_STYLE_BG_COLOR`)
- **value** -- *`lv_style_value_t`* variable in which a field is set according to the type of `prop`

lv_res_t **lv_style_get_prop**(const *lv_style_t* *style, *lv_style_prop_t* prop, *lv_style_value_t* *value)

Get the value of a property

注解： For performance reasons there are no sanity check on `style`

参数

- **style** -- pointer to a style
- **prop** -- the ID of a property
- **value** -- pointer to a *lv_style_value_t* variable to store the value

返回 LV_RES_INV: the property wasn't found in the style (`value` is unchanged) LV_RES_OK: the property was fond, and `value` is set accordingly

static inline lv_res_t **lv_style_get_prop_inlined**(const *lv_style_t* *style, *lv_style_prop_t* prop, *lv_style_value_t* *value)

Get the value of a property

注解： For performance reasons there are no sanity check on `style`

注解： This function is the same as *lv_style_get_prop* but inlined. Use it only on performance critical places

参数

- **style** -- pointer to a style
- **prop** -- the ID of a property
- **value** -- pointer to a *lv_style_value_t* variable to store the value

返回 LV_RES_INV: the property wasn't found in the style (`value` is unchanged) LV_RES_OK: the property was fond, and `value` is set accordingly

void **lv_style_transition_dsc_init**(*lv_style_transition_dsc_t* *tr, const *lv_style_prop_t* props[], *lv_anim_path_cb_t* path_cb, uint32_t time, uint32_t delay, void *user_data)

lv_style_value_t **lv_style_prop_get_default**(*lv_style_prop_t* prop)

Get the default value of a property

参数 **prop** -- the ID of a property

返回 the default value

bool **lv_style_is_empty**(const *lv_style_t* *style)

Checks if a style is empty (has no properties)

参数 **style** -- pointer to a style

返回 true if the style is empty

uint8_t **_lv_style_get_prop_group**(*lv_style_prop_t* prop)

Tell the group of a property. If the a property from a group is set in a style the (1 « group) bit of style->has_group is set. It allows early skipping the style if the property is not exists in the style at all.

参数 **prop** -- a style property

返回 the group [0..7] 7 means all the custom properties with index > 112

static inline void **lv_style_set_size**(*lv_style_t* *style, lv_coord_t value)

static inline void **lv_style_set_pad_all**(*lv_style_t* *style, lv_coord_t value)

static inline void **lv_style_set_pad_hor**(*lv_style_t* *style, lv_coord_t value)

static inline void **lv_style_set_pad_ver**(*lv_style_t* *style, lv_coord_t value)

static inline void **lv_style_set_pad_gap**(*lv_style_t* *style, lv_coord_t value)

struct **lv_gradient_stop_t**

#include <lv_style.h> A gradient stop definition. This matches a color and a position in a virtual 0-255 scale.

Public Members

lv_color_t **color**

The stop color

uint8_t **frac**

The stop position in 1/255 unit

struct **lv_grad_dsc_t**

#include <lv_style.h> A descriptor of a gradient.

Public Members

lv_gradient_stop_t **stops**[LV_GRADIENT_MAX_STOPS]
A gradient stop array

uint8_t **stops_count**
The number of used stops in the array

lv_grad_dir_t **dir**
The gradient direction. Any of LV_GRAD_DIR_HOR, LV_GRAD_DIR_VER, LV_GRAD_DIR_NONE

lv_dither_mode_t **dither**
Whether to dither the gradient or not. Any of LV_DITHER_NONE, LV_DITHER_ORDERED, LV_DITHER_ERR_DIFF

union **lv_style_value_t**
#include <lv_style.h> A common type to handle all the property types in the same way.

Public Members

int32_t **num**
Number integer number (opacity, enums, booleans or "normal" numbers)

const void ***ptr**
Constant pointers (font, cone text, etc)

lv_color_t **color**
Colors

struct **lv_style_transition_dsc_t**
#include <lv_style.h> Descriptor for style transitions

Public Members

const *lv_style_prop_t* ***props**
An array with the properties to animate.

void ***user_data**
A custom user data that will be passed to the animation's user_data

lv_anim_path_cb_t **path_xcb**
A path for the animation.

uint32_t **time**
Duration of the transition in [ms]

uint32_t **delay**
Delay before the transition in [ms]

struct **lv_style_const_prop_t**
#include <lv_style.h> Descriptor of a constant style property.

Public Members

lv_style_prop_t **prop**

lv_style_value_t **value**

struct **lv_style_t**
#include <lv_style.h> Descriptor of a style (a collection of properties and values).

Public Members

uint32_t **sentinel**

lv_style_value_t **value1**

uint8_t ***values_and_props**

const *lv_style_const_prop_t* ***const_props**

union *lv_style_t*::[anonymous] **v_p**

uint16_t **prop1**

uint16_t **is_const**

uint8_t **has_group**

uint8_t **prop_cnt**

Typedefs

typedef void (***lv_theme_apply_cb_t**)(struct *_lv_theme_t**, *lv_obj_t**)

typedef struct *_lv_theme_t* **lv_theme_t**

Functions

lv_theme_t ***lv_theme_get_from_obj**(*lv_obj_t* *obj)

Get the theme assigned to the display of the object

参数 **obj** -- pointer to a theme object

返回 the theme of the object's display (can be NULL)

void **lv_theme_apply**(*lv_obj_t* *obj)

Apply the active theme on an object

参数 **obj** -- pointer to an object

void **lv_theme_set_parent**(*lv_theme_t* *new_theme, *lv_theme_t* *parent)

Set a base theme for a theme. The styles from the base them will be added before the styles of the current theme. Arbitrary long chain of themes can be created by setting base themes.

参数

- **new_theme** -- pointer to theme which base should be set
- **parent** -- pointer to the base theme

void **lv_theme_set_apply_cb**(*lv_theme_t* *theme, *lv_theme_apply_cb_t* apply_cb)

Set an apply callback for a theme. The apply callback is used to add styles to different objects

参数

- **theme** -- pointer to theme which callback should be set
- **apply_cb** -- pointer to the callback

const lv_font_t ***lv_theme_get_font_small**(*lv_obj_t* *obj)

Get the small font of the theme

参数 **obj** -- pointer to an object

返回 pointer to the font

const lv_font_t ***lv_theme_get_font_normal**(*lv_obj_t* *obj)

Get the normal font of the theme

参数 **obj** -- pointer to an object

返回 pointer to the font

const lv_font_t *`lv_theme_get_font_large`(*lv_obj_t* *obj)

Get the subtitle font of the theme

参数 **obj** -- pointer to an object

返回 pointer to the font

lv_color_t `lv_theme_get_color_primary`(*lv_obj_t* *obj)

Get the primary color of the theme

参数 **obj** -- pointer to an object

返回 the color

lv_color_t `lv_theme_get_color_secondary`(*lv_obj_t* *obj)

Get the secondary color of the theme

参数 **obj** -- pointer to an object

返回 the color

struct `_lv_theme_t`

Public Members

lv_theme_apply_cb_t **apply_cb**

struct *_lv_theme_t* ***parent**

Apply the current theme's style on top of this theme.

void ***user_data**

struct *_lv_disp_t* ***disp**

lv_color_t **color_primary**

lv_color_t **color_secondary**

const lv_font_t ***font_small**

const lv_font_t ***font_normal**

const lv_font_t ***font_large**

uint32_t **flags**

Functions

static inline lv_coord_t **lv_obj_get_style_width**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_coord_t **lv_obj_get_style_min_width**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_coord_t **lv_obj_get_style_max_width**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_coord_t **lv_obj_get_style_height**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_coord_t **lv_obj_get_style_min_height**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_coord_t **lv_obj_get_style_max_height**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_coord_t **lv_obj_get_style_x**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_coord_t **lv_obj_get_style_y**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_align_t **lv_obj_get_style_align**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_coord_t **lv_obj_get_style_transform_width**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_coord_t **lv_obj_get_style_transform_height**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_coord_t **lv_obj_get_style_translate_x**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_coord_t **lv_obj_get_style_translate_y**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_coord_t **lv_obj_get_style_transform_zoom**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_coord_t **lv_obj_get_style_transform_angle**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_coord_t **lv_obj_get_style_pad_top**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_coord_t **lv_obj_get_style_pad_bottom**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_coord_t **lv_obj_get_style_pad_left**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_coord_t **lv_obj_get_style_pad_right**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_coord_t **lv_obj_get_style_pad_row**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_coord_t **lv_obj_get_style_pad_column**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_color_t **lv_obj_get_style_bg_color**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_color_t **lv_obj_get_style_bg_color_filtered**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_opa_t **lv_obj_get_style_bg_opa**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_color_t **lv_obj_get_style_bg_grad_color**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_color_t **lv_obj_get_style_bg_grad_color_filtered**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline *lv_grad_dir_t* **lv_obj_get_style_bg_grad_dir**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_coord_t **lv_obj_get_style_bg_main_stop**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_coord_t **lv_obj_get_style_bg_grad_stop**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline const *lv_grad_dsc_t* ***lv_obj_get_style_bg_grad**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline *lv_dither_mode_t* **lv_obj_get_style_bg_dither_mode**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline const void ***lv_obj_get_style_bg_img_src**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_opa_t **lv_obj_get_style_bg_img_opa**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_color_t **lv_obj_get_style_bg_img_recolor**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_color_t **lv_obj_get_style_bg_img_recolor_filtered**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_opa_t **lv_obj_get_style_bg_img_recolor_opa**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline bool **lv_obj_get_style_bg_img_tiled**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_color_t **lv_obj_get_style_border_color**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_color_t **lv_obj_get_style_border_color_filtered**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_opa_t **lv_obj_get_style_border_opa**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_coord_t **lv_obj_get_style_border_width**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline *lv_border_side_t* **lv_obj_get_style_border_side**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline bool **lv_obj_get_style_border_post**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_coord_t **lv_obj_get_style_outline_width**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_color_t **lv_obj_get_style_outline_color**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_color_t **lv_obj_get_style_outline_color_filtered**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_opa_t **lv_obj_get_style_outline_opa**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_coord_t **lv_obj_get_style_outline_pad**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_coord_t **lv_obj_get_style_shadow_width**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_coord_t **lv_obj_get_style_shadow_ofs_x**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_coord_t **lv_obj_get_style_shadow_ofs_y**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_coord_t **lv_obj_get_style_shadow_spread**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_color_t **lv_obj_get_style_shadow_color**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_color_t **lv_obj_get_style_shadow_color_filtered**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_opa_t **lv_obj_get_style_shadow_opa**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_opa_t **lv_obj_get_style_img_opa**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_color_t **lv_obj_get_style_img_recolor**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_color_t **lv_obj_get_style_img_recolor_filtered**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_opa_t **lv_obj_get_style_img_recolor_opa**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_coord_t **lv_obj_get_style_line_width**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_coord_t **lv_obj_get_style_line_dash_width**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_coord_t **lv_obj_get_style_line_dash_gap**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline bool **lv_obj_get_style_line_rounded**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_color_t **lv_obj_get_style_line_color**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_color_t **lv_obj_get_style_line_color_filtered**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_opa_t **lv_obj_get_style_line_opa**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_coord_t **lv_obj_get_style_arc_width**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline bool **lv_obj_get_style_arc_rounded**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_color_t **lv_obj_get_style_arc_color**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_color_t **lv_obj_get_style_arc_color_filtered**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_opa_t **lv_obj_get_style_arc_opa**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline const void ***lv_obj_get_style_arc_img_src**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_color_t **lv_obj_get_style_text_color**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_color_t **lv_obj_get_style_text_color_filtered**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_opa_t **lv_obj_get_style_text_opa**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline const lv_font_t ***lv_obj_get_style_text_font**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_coord_t **lv_obj_get_style_text_letter_space**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_coord_t **lv_obj_get_style_text_line_space**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline *lv_text_decor_t* **lv_obj_get_style_text_decor**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_text_align_t **lv_obj_get_style_text_align**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_coord_t **lv_obj_get_style_radius**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline bool **lv_obj_get_style_clip_corner**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_opa_t **lv_obj_get_style_opa**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline const *lv_color_filter_dsc_t* ***lv_obj_get_style_color_filter_dsc**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_opa_t **lv_obj_get_style_color_filter_opa**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline uint32_t **lv_obj_get_style_anim_time**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline uint32_t **lv_obj_get_style_anim_speed**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline const *lv_style_transition_dsc_t* ***lv_obj_get_style_transition**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline *lv_blend_mode_t* **lv_obj_get_style_blend_mode**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline uint16_t **lv_obj_get_style_layout**(const struct *_lv_obj_t* *obj, uint32_t part)

static inline lv_base_dir_t **lv_obj_get_style_base_dir**(const struct *_lv_obj_t* *obj, uint32_t part)

void **lv_obj_set_style_width**(struct *_lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

void **lv_obj_set_style_min_width**(struct *_lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

void **lv_obj_set_style_max_width**(struct *_lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

void **lv_obj_set_style_height**(struct *_lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

void **lv_obj_set_style_min_height**(struct *_lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

void **lv_obj_set_style_max_height**(struct *_lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

void **lv_obj_set_style_x**(struct *_lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

void **lv_obj_set_style_y**(struct *_lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

void **lv_obj_set_style_align**(struct *_lv_obj_t* *obj, lv_align_t value, lv_style_selector_t selector)

void **lv_obj_set_style_transform_width**(struct *_lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

void **lv_obj_set_style_transform_height**(struct *_lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

void **lv_obj_set_style_translate_x**(struct *_lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

void **lv_obj_set_style_translate_y**(struct *_lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

void **lv_obj_set_style_transform_zoom**(struct *_lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

void **lv_obj_set_style_transform_angle**(struct *_lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

void **lv_obj_set_style_pad_top**(struct *_lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

void **lv_obj_set_style_pad_bottom**(struct *_lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

void **lv_obj_set_style_pad_left**(struct *_lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

void **lv_obj_set_style_pad_right**(struct *_lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

void **lv_obj_set_style_pad_row**(struct *_lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

void **lv_obj_set_style_pad_column**(struct *_lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

void **lv_obj_set_style_bg_color**(struct *_lv_obj_t* *obj, lv_color_t value, lv_style_selector_t selector)

void **lv_obj_set_style_bg_color_filtered**(struct *_lv_obj_t* *obj, lv_color_t value, lv_style_selector_t selector)

void **lv_obj_set_style_bg_opa**(struct *_lv_obj_t* *obj, lv_opa_t value, lv_style_selector_t selector)

void **lv_obj_set_style_bg_grad_color**(struct *_lv_obj_t* *obj, lv_color_t value, lv_style_selector_t selector)

void **lv_obj_set_style_bg_grad_color_filtered**(struct *_lv_obj_t* *obj, lv_color_t value, lv_style_selector_t selector)

void **lv_obj_set_style_bg_grad_dir**(struct *_lv_obj_t* *obj, *lv_grad_dir_t* value, lv_style_selector_t selector)

void **lv_obj_set_style_bg_main_stop**(struct *_lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

void **lv_obj_set_style_bg_grad_stop**(struct *_lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

void **lv_obj_set_style_bg_grad**(struct *_lv_obj_t* *obj, const *lv_grad_dsc_t* *value, lv_style_selector_t selector)

void **lv_obj_set_style_bg_dither_mode**(struct *_lv_obj_t* *obj, *lv_dither_mode_t* value, lv_style_selector_t selector)

void **lv_obj_set_style_bg_img_src**(struct *_lv_obj_t* *obj, const void *value, lv_style_selector_t selector)

void **lv_obj_set_style_bg_img_opa**(struct *_lv_obj_t* *obj, lv_opa_t value, lv_style_selector_t selector)

void **lv_obj_set_style_bg_img_recolor**(struct *_lv_obj_t* *obj, lv_color_t value, lv_style_selector_t selector)

void **lv_obj_set_style_bg_img_recolor_filtered**(struct *_lv_obj_t* *obj, lv_color_t value, lv_style_selector_t selector)

void **lv_obj_set_style_bg_img_recolor_opa**(struct *_lv_obj_t* *obj, lv_opa_t value, lv_style_selector_t selector)

void **lv_obj_set_style_bg_img_tiled**(struct *_lv_obj_t* *obj, bool value, lv_style_selector_t selector)

void **lv_obj_set_style_border_color**(struct *_lv_obj_t* *obj, lv_color_t value, lv_style_selector_t selector)

void **lv_obj_set_style_border_color_filtered**(struct *_lv_obj_t* *obj, lv_color_t value, lv_style_selector_t selector)

void **lv_obj_set_style_border_opa**(struct *_lv_obj_t* *obj, lv_opa_t value, lv_style_selector_t selector)

void **lv_obj_set_style_border_width**(struct *_lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

void **lv_obj_set_style_border_side**(struct *_lv_obj_t* *obj, *lv_border_side_t* value, lv_style_selector_t selector)

void **lv_obj_set_style_border_post**(struct *_lv_obj_t* *obj, bool value, lv_style_selector_t selector)

void **lv_obj_set_style_outline_width**(struct *_lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

void **lv_obj_set_style_outline_color**(struct *_lv_obj_t* *obj, lv_color_t value, lv_style_selector_t selector)

void **lv_obj_set_style_outline_color_filtered**(struct *_lv_obj_t* *obj, lv_color_t value, lv_style_selector_t selector)

void **lv_obj_set_style_outline_opa**(struct *_lv_obj_t* *obj, lv_opa_t value, lv_style_selector_t selector)

void **lv_obj_set_style_outline_pad**(struct *_lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

void **lv_obj_set_style_shadow_width**(struct *_lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

void **lv_obj_set_style_shadow_ofs_x**(struct *_lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

void **lv_obj_set_style_shadow_ofs_y**(struct *_lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

void **lv_obj_set_style_shadow_spread**(struct *_lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

void **lv_obj_set_style_shadow_color**(struct *_lv_obj_t* *obj, lv_color_t value, lv_style_selector_t selector)

void **lv_obj_set_style_shadow_color_filtered**(struct *_lv_obj_t* *obj, lv_color_t value, lv_style_selector_t selector)

void **lv_obj_set_style_shadow_opa**(struct *_lv_obj_t* *obj, lv_opa_t value, lv_style_selector_t selector)

void **lv_obj_set_style_img_opa**(struct *_lv_obj_t* *obj, lv_opa_t value, lv_style_selector_t selector)

void **lv_obj_set_style_img_recolor**(struct *_lv_obj_t* *obj, lv_color_t value, lv_style_selector_t selector)

void **lv_obj_set_style_img_recolor_filtered**(struct *_lv_obj_t* *obj, lv_color_t value, lv_style_selector_t selector)

void **lv_obj_set_style_img_recolor_opa**(struct *_lv_obj_t* *obj, lv_opa_t value, lv_style_selector_t selector)

void **lv_obj_set_style_line_width**(struct *_lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

void **lv_obj_set_style_line_dash_width**(struct *_lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

void **lv_obj_set_style_line_dash_gap**(struct *_lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

void **lv_obj_set_style_line_rounded**(struct *_lv_obj_t* *obj, bool value, lv_style_selector_t selector)

void **lv_obj_set_style_line_color**(struct *_lv_obj_t* *obj, lv_color_t value, lv_style_selector_t selector)

void **lv_obj_set_style_line_color_filtered**(struct *_lv_obj_t* *obj, lv_color_t value, lv_style_selector_t selector)

void **lv_obj_set_style_line_opa**(struct *_lv_obj_t* *obj, lv_opa_t value, lv_style_selector_t selector)

void **lv_obj_set_style_arc_width**(struct *_lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

void **lv_obj_set_style_arc_rounded**(struct *_lv_obj_t* *obj, bool value, lv_style_selector_t selector)

void **lv_obj_set_style_arc_color**(struct *_lv_obj_t* *obj, lv_color_t value, lv_style_selector_t selector)

void **lv_obj_set_style_arc_color_filtered**(struct *_lv_obj_t* *obj, lv_color_t value, lv_style_selector_t selector)

void **lv_obj_set_style_arc_opa**(struct *_lv_obj_t* *obj, lv_opa_t value, lv_style_selector_t selector)

void **lv_obj_set_style_arc_img_src**(struct *_lv_obj_t* *obj, const void *value, lv_style_selector_t selector)

void **lv_obj_set_style_text_color**(struct *_lv_obj_t* *obj, lv_color_t value, lv_style_selector_t selector)

void **lv_obj_set_style_text_color_filtered**(struct *_lv_obj_t* *obj, lv_color_t value, lv_style_selector_t selector)

void **lv_obj_set_style_text_opa**(struct *_lv_obj_t* *obj, lv_opa_t value, lv_style_selector_t selector)

void **lv_obj_set_style_text_font**(struct *_lv_obj_t* *obj, const lv_font_t *value, lv_style_selector_t selector)

void **lv_obj_set_style_text_letter_space**(struct *_lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

void **lv_obj_set_style_text_line_space**(struct *_lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

void **lv_obj_set_style_text_decor**(struct *_lv_obj_t* *obj, *lv_text_decor_t* value, lv_style_selector_t selector)

void **lv_obj_set_style_text_align**(struct *_lv_obj_t* *obj, lv_text_align_t value, lv_style_selector_t selector)

void **lv_obj_set_style_radius**(struct *_lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

void **lv_obj_set_style_clip_corner**(struct *_lv_obj_t* *obj, bool value, lv_style_selector_t selector)

void **lv_obj_set_style_opa**(struct *_lv_obj_t* *obj, lv_opa_t value, lv_style_selector_t selector)

void **lv_obj_set_style_color_filter_dsc**(struct *_lv_obj_t* *obj, const *lv_color_filter_dsc_t* *value, lv_style_selector_t selector)

void **lv_obj_set_style_color_filter_opa**(struct *_lv_obj_t* *obj, lv_opa_t value, lv_style_selector_t selector)

void **lv_obj_set_style_anim_time**(struct *_lv_obj_t* *obj, uint32_t value, lv_style_selector_t selector)

void **lv_obj_set_style_anim_speed**(struct *_lv_obj_t* *obj, uint32_t value, lv_style_selector_t selector)

void **lv_obj_set_style_transition**(struct *_lv_obj_t* *obj, const *lv_style_transition_dsc_t* *value, lv_style_selector_t selector)

void **lv_obj_set_style_blend_mode**(struct *_lv_obj_t* *obj, *lv_blend_mode_t* value, lv_style_selector_t selector)

void **lv_obj_set_style_layout**(struct *_lv_obj_t* *obj, uint16_t value, lv_style_selector_t selector)

void **lv_obj_set_style_base_dir**(struct *_lv_obj_t* *obj, lv_base_dir_t value, lv_style_selector_t selector)

Functions

void **lv_style_set_width**(*lv_style_t* *style, lv_coord_t value)

void **lv_style_set_min_width**(*lv_style_t* *style, lv_coord_t value)

void **lv_style_set_max_width**(*lv_style_t* *style, lv_coord_t value)

void **lv_style_set_height**(*lv_style_t* *style, lv_coord_t value)

void **lv_style_set_min_height**(*lv_style_t* *style, lv_coord_t value)

void **lv_style_set_max_height**(*lv_style_t* *style, lv_coord_t value)

void **lv_style_set_x**(*lv_style_t* *style, lv_coord_t value)

void **lv_style_set_y**(*lv_style_t* *style, lv_coord_t value)

void **lv_style_set_align**(*lv_style_t* *style, lv_align_t value)

void **lv_style_set_transform_width**(*lv_style_t* *style, lv_coord_t value)

void **lv_style_set_transform_height**(*lv_style_t* *style, lv_coord_t value)

void **lv_style_set_translate_x**(*lv_style_t* *style, lv_coord_t value)

void **lv_style_set_translate_y**(*lv_style_t* *style, lv_coord_t value)

void **lv_style_set_transform_zoom**(*lv_style_t* *style, lv_coord_t value)

void **lv_style_set_transform_angle**(*lv_style_t* *style, lv_coord_t value)

void **lv_style_set_pad_top**(*lv_style_t* *style, lv_coord_t value)

void **lv_style_set_pad_bottom**(*lv_style_t* *style, lv_coord_t value)

void **lv_style_set_pad_left**(*lv_style_t* *style, lv_coord_t value)

void **lv_style_set_pad_right**(*lv_style_t* *style, lv_coord_t value)

void **lv_style_set_pad_row**(*lv_style_t* *style, lv_coord_t value)

void **lv_style_set_pad_column**(*lv_style_t* *style, lv_coord_t value)

void **lv_style_set_bg_color**(*lv_style_t* *style, lv_color_t value)

void **lv_style_set_bg_color_filtered**(*lv_style_t* *style, lv_color_t value)

void **lv_style_set_bg_opa**(*lv_style_t* *style, lv_opa_t value)

void **lv_style_set_bg_grad_color**(*lv_style_t* *style, lv_color_t value)

void **lv_style_set_bg_grad_color_filtered**(*lv_style_t* *style, lv_color_t value)

void **lv_style_set_bg_grad_dir**(*lv_style_t* *style, *lv_grad_dir_t* value)

void **lv_style_set_bg_main_stop**(*lv_style_t* *style, lv_coord_t value)

void **lv_style_set_bg_grad_stop**(*lv_style_t* *style, lv_coord_t value)

void **lv_style_set_bg_grad**(*lv_style_t* *style, const *lv_grad_dsc_t* *value)

void **lv_style_set_bg_dither_mode**(*lv_style_t* *style, *lv_dither_mode_t* value)

void **lv_style_set_bg_img_src**(*lv_style_t* *style, const void *value)

void **lv_style_set_bg_img_opa**(*lv_style_t* *style, lv_opa_t value)

void **lv_style_set_bg_img_recolor**(*lv_style_t* *style, lv_color_t value)

void **lv_style_set_bg_img_recolor_filtered**(*lv_style_t* *style, lv_color_t value)

void **lv_style_set_bg_img_recolor_opa**(*lv_style_t* *style, lv_opa_t value)

void **lv_style_set_bg_img_tiled**(*lv_style_t* *style, bool value)

void **lv_style_set_border_color**(*lv_style_t* *style, lv_color_t value)

void **lv_style_set_border_color_filtered**(*lv_style_t* *style, lv_color_t value)

void **lv_style_set_border_opa**(*lv_style_t* *style, lv_opa_t value)

void **lv_style_set_border_width**(*lv_style_t* *style, lv_coord_t value)

void **lv_style_set_border_side**(*lv_style_t* *style, *lv_border_side_t* value)

void **lv_style_set_border_post**(*lv_style_t* *style, bool value)

void **lv_style_set_outline_width**(*lv_style_t* *style, lv_coord_t value)

void **lv_style_set_outline_color**(*lv_style_t* *style, lv_color_t value)

void **lv_style_set_outline_color_filtered**(*lv_style_t* *style, lv_color_t value)

void **lv_style_set_outline_opa**(*lv_style_t* *style, lv_opa_t value)

void **lv_style_set_outline_pad**(*lv_style_t* *style, lv_coord_t value)

void **lv_style_set_shadow_width**(*lv_style_t* *style, lv_coord_t value)

void **lv_style_set_shadow_ofs_x**(*lv_style_t* *style, lv_coord_t value)

void **lv_style_set_shadow_ofs_y**(*lv_style_t* *style, lv_coord_t value)

void **lv_style_set_shadow_spread**(*lv_style_t* *style, lv_coord_t value)

void **lv_style_set_shadow_color**(*lv_style_t* *style, lv_color_t value)

void **lv_style_set_shadow_color_filtered**(*lv_style_t* *style, lv_color_t value)

void **lv_style_set_shadow_opa**(*lv_style_t* *style, lv_opa_t value)

void **lv_style_set_img_opa**(*lv_style_t* *style, lv_opa_t value)

void **lv_style_set_img_recolor**(*lv_style_t* *style, lv_color_t value)

void **lv_style_set_img_recolor_filtered**(*lv_style_t* *style, lv_color_t value)

void **lv_style_set_img_recolor_opa**(*lv_style_t* *style, lv_opa_t value)

void **lv_style_set_line_width**(*lv_style_t* *style, lv_coord_t value)

void **lv_style_set_line_dash_width**(*lv_style_t* *style, lv_coord_t value)

void **lv_style_set_line_dash_gap**(*lv_style_t* *style, lv_coord_t value)

void **lv_style_set_line_rounded**(*lv_style_t* *style, bool value)

void **lv_style_set_line_color**(*lv_style_t* *style, lv_color_t value)

void **lv_style_set_line_color_filtered**(*lv_style_t* *style, lv_color_t value)

void **lv_style_set_line_opa**(*lv_style_t* *style, lv_opa_t value)

void **lv_style_set_arc_width**(*lv_style_t* *style, lv_coord_t value)

void **lv_style_set_arc_rounded**(*lv_style_t* *style, bool value)

void **lv_style_set_arc_color**(*lv_style_t* *style, lv_color_t value)

void **lv_style_set_arc_color_filtered**(*lv_style_t* *style, lv_color_t value)

void **lv_style_set_arc_opa**(*lv_style_t* *style, lv_opa_t value)

void **lv_style_set_arc_img_src**(*lv_style_t* *style, const void *value)

void **lv_style_set_text_color**(*lv_style_t* *style, lv_color_t value)

void **lv_style_set_text_color_filtered**(*lv_style_t* *style, lv_color_t value)

void **lv_style_set_text_opa**(*lv_style_t* *style, lv_opa_t value)

void **lv_style_set_text_font**(*lv_style_t* *style, const lv_font_t *value)

void **lv_style_set_text_letter_space**(*lv_style_t* *style, lv_coord_t value)

void **lv_style_set_text_line_space**(*lv_style_t* *style, lv_coord_t value)

void **lv_style_set_text_decor**(*lv_style_t* *style, *lv_text_decor_t* value)

void **lv_style_set_text_align**(*lv_style_t* *style, lv_text_align_t value)

void **lv_style_set_radius**(*lv_style_t* *style, lv_coord_t value)

void **lv_style_set_clip_corner**(*lv_style_t* *style, bool value)

void **lv_style_set_opa**(*lv_style_t* *style, lv_opa_t value)

void **lv_style_set_color_filter_dsc**(*lv_style_t* *style, const *lv_color_filter_dsc_t* *value)

void **lv_style_set_color_filter_opa**(*lv_style_t* *style, lv_opa_t value)

void **lv_style_set_anim_time**(*lv_style_t* *style, uint32_t value)

void **lv_style_set_anim_speed**(*lv_style_t* *style, uint32_t value)

void **lv_style_set_transition**(*lv_style_t* *style, const *lv_style_transition_dsc_t* *value)

void **lv_style_set_blend_mode**(*lv_style_t* *style, *lv_blend_mode_t* value)

void **lv_style_set_layout**(*lv_style_t* *style, uint16_t value)

void **lv_style_set_base_dir**(*lv_style_t* *style, lv_base_dir_t value)

2.5.4 Style properties

Size and position

TODO

width

Sets the width of object. Pixel, percentage and `LV_SIZE_CONTENT` values can be used. Percentage values are relative to the width of the parent's content area.

min_width

Sets a minimal width. Pixel and percentage values can be used. Percentage values are relative to the width of the parent's content area.

max_width

Sets a maximal width. Pixel and percentage values can be used. Percentage values are relative to the width of the parent's content area.

height

Sets the height of object. Pixel, percentage and `LV_SIZE_CONTENT` can be used. Percentage values are relative to the height of the parent's content area.

min_height

Sets a minimal height. Pixel and percentage values can be used. Percentage values are relative to the width of the parent's content area.

max_height

Sets a maximal height. Pixel and percentage values can be used. Percentage values are relative to the height of the parent's content area.

x

Set the X coordinate of the object considering the set `align`. Pixel and percentage values can be used. Percentage values are relative to the width of the parent's content area.

y

Set the Y coordinate of the object considering the set `align`. Pixel and percentage values can be used. Percentage values are relative to the height of the parent's content area.

align

Set the alignment which determines from which point of the parent the X and Y coordinates should be interpreted. The possible values are: `LV_ALIGN_TOP_LEFT/MID/RIGHT`, `LV_ALIGN_BOTTOM_LEFT/MID/RIGHT`, `LV_ALIGN_LEFT/RIGHT_MID`, `LV_ALIGN_CENTER`

transform_width

Make the object wider on both sides with this value. Pixel and percentage (with `lv_pct(x)`) values can be used. Percentage values are relative to the object's width.

transform_height

Make the object higher on both sides with this value. Pixel and percentage (with `lv_pct(x)`) values can be used. Percentage values are relative to the object's height.

translate_x

Move the object with this value in X direction. Applied after layouts, aligns and other positioning. Pixel and percentage (with `lv_pct(x)`) values can be used. Percentage values are relative to the object's width.

translate_y

Move the object with this value in Y direction. Applied after layouts, aligns and other positioning. Pixel and percentage (with `lv_pct(x)`) values can be used. Percentage values are relative to the object's height.

transform_zoom

Zoom image-like objects. Multiplied with the zoom set on the object. The value 256 (or `LV_IMG_ZOOM_NONE`) means normal size, 128 half size, 512 double size, and so on

transform_angle

Rotate image-like objects. Added to the rotation set on the object. The value is interpreted in 0.1 degree unit. E.g. 45 deg. = 450

Padding

TODO

pad_top

Sets the padding on the top. It makes the content area smaller in this direction.

pad_bottom

Sets the padding on the bottom. It makes the content area smaller in this direction.

pad_left

Sets the padding on the left. It makes the content area smaller in this direction.

pad_right

Sets the padding on the right. It makes the content area smaller in this direction.

pad_row

Sets the padding between the rows. Used by the layouts.

pad_column

Sets the padding between the columns. Used by the layouts.

Miscellaneous

TODO

radius

Set the radius on every corner. The value is interpreted in pixel (>= 0) or `LV_RADIUS_CIRCLE` for max. radius

clip_corner

Enable to clip the overflowed content on the rounded corner. Can be `true` or `false`.

opa

Scale down all opacity values of the object by this factor. Value 0, `LV_OPA_0` or `LV_OPA_TRANSP` means fully transparent, 256, `LV_OPA_100` or `LV_OPA_COVER` means fully covering, other values or LV_OPA_10, LV_OPA_20, etc indicate semi-transparency.

color_filter_dsc

Mix a color to all colors of the object.

color_filter_opa

The intensity of mixing of color filter.

anim_time

The animation time in milliseconds. Its meaning is widget specific. E.g. blink time of the cursor on the text area or scroll time of a roller. See the widgets' documentation to learn more.

anim_speed

The animation speed in pixel/sec. Its meaning is widget specific. E.g. scroll speed of label. See the widgets' documentation to learn more.

transition

An initialized `lv_style_transition_dsc_t` to describe a transition.

blend_mode

Describes how to blend the colors to the background. The possible values are `LV_BLEND_MODE_NORMAL/ADDITIVE/SUBTRACTIVE`

layout

Set the layout if the object. The children will be repositioned and resized according to the policies set for the layout. For the possible values see the documentation of the layouts.

base_dir

Set the base direction of the object. The possible values are `LV_BIDI_DIR_LTR/RTL/AUTO`.

Background

TODO

bg_color

Set the background color of the object.

bg_opa

Set the opacity of the background. Value 0, `LV_OPA_0` or `LV_OPA_TRANSP` means fully transparent, 256, `LV_OPA_100` or `LV_OPA_COVER` means fully covering, other values or LV_OPA_10, LV_OPA_20, etc indicate semi-transparency.

bg_grad_color

Set the gradient color of the background. Used only if `grad_dir` is not `LV_GRAD_DIR_NONE`

bg_grad_dir

Set the direction of the gradient of the background. The possible values are `LV_GRAD_DIR_NONE/HOR/VER`.

bg_main_stop

Set the point from which the background color should start for gradients. 0 means to top/left side, 255 the bottom/right side, 128 the center, and so on

bg_grad_stop

Set the point from which the background's gradient color should start. 0 means to top/left side, 255 the bottom/right side, 128 the center, and so on

bg_img_src

Set a background image. Can be a pointer to `lv_img_dsc_t`, a path to a file or an `LV_SYMBOL_...`

bg_img_opa

Set the opacity of the background image. Value 0, `LV_OPA_0` or `LV_OPA_TRANSP` means fully transparent, 256, `LV_OPA_100` or `LV_OPA_COVER` means fully covering, other values or LV_OPA_10, LV_OPA_20, etc indicate semi-transparency.

bg_img_recolor

Set a color to mix to the background image.

bg_img_recolor_opa

Set the intensity of background image recoloring. Value 0, `LV_OPA_0` or `LV_OPA_TRANSP` means no mixing, 256, `LV_OPA_100` or `LV_OPA_COVER` means full recoloring, other values or LV_OPA_10, LV_OPA_20, etc are interpreted proportionally.

bg_img_tiled

If enabled the background image will be tiled. The possible values are `true` or `false`.

Border

TODO

border_color

Set the color of the border

border_opa

Set the opcitiy of the border. Value 0, `LV_OPA_0` or `LV_OPA_TRANSP` means fully transparent, 256, `LV_OPA_100` or `LV_OPA_COVER` means fully covering, other values or LV_OPA_10, LV_OPA_20, etc indicate semi-transparency.

border_width

Set the width of the border. Only pixel values can be used.

border_side

Set which side(s) the border should be drawn. The possible values are `LV_BORDER_SIDE_NONE/TOP/BOTTOM/LEFT/RIGHT/INTERNAL`. OR-ed calues an be used as well, e.g. `LV_BORDER_SIDE_TOP | LV_BORDER_SIDE_LEFT`.

border_post

Sets wheter the border should be drawn before or after the children ar drawn. `true`: after children, `false`: before children

Text

TODO

text_color

Sets the color of the text.

text_opa

Set the opacity of the text. Value 0, `LV_OPA_0` or `LV_OPA_TRANSP` means fully transparent, 256, `LV_OPA_100` or `LV_OPA_COVER` means fully covering, other values or LV_OPA_10, LV_OPA_20, etc indicate semi-transparency.

text_font

Set the font of the text (a pointer `lv_font_t *`).

text_letter_space

Set the letter space in pixels

text_line_space

Set the line space in pixels.

text_decor

Set decoration for the text. The possible values are `LV_TEXT_DECOR_NONE/UNDERLINE/STRIKETHROUGH`. OR-ed values can be used as well.

text_align

Set how to align the lines of the text. Note that it doesn't align the object itself, only the lines inside the object. The possible values are `LV_TEXT_ALIGN_LEFT/CENTER/RIGHT/AUTO`. `LV_TEXT_ALIGN_AUTO` detect the text base direction and uses left or right alignment accordingly

Image

TODO

img_opa

Set the opacity of an image. Value 0, `LV_OPA_0` or `LV_OPA_TRANSP` means fully transparent, 256, `LV_OPA_100` or `LV_OPA_COVER` means fully covering, other values or LV_OPA_10, LV_OPA_20, etc indicate semi-transparency.

img_recolor

Set color to mix to the image.

img_recolor_opa

Set the intensity of the color mixing. Value 0, `LV_OPA_0` or `LV_OPA_TRANSP` means fully transparent, 256, `LV_OPA_100` or `LV_OPA_COVER` means fully covering, other values or LV_OPA_10, LV_OPA_20, etc indicate semi-transparency.

Outline

TODO

outline_width

Set the width of the outline in pixels.

outline_color

Set the color of the outline.

outline_opa

Set the opacity of the outline. Value 0, `LV_OPA_0` or `LV_OPA_TRANSP` means fully transparent, 256, `LV_OPA_100` or `LV_OPA_COVER` means fully covering, other values or LV_OPA_10, LV_OPA_20, etc indicate semi-transparency.

outline_pad

Set the padding of the outline, i.e. the gap between object and the outline.

Shadow

TODO

shadow_width

Set the width of the shadow in pixels. The value should be >= 0.

shadow_ofs_x

Set an offset on the shadow in pixels in X direction.

shadow_ofs_y

Set an offset on the shadow in pixels in Y direction.

shadow_spread

Make the shadow calculation to use a larger or smaller rectangle as base. The value can be in pixel to make the area larger/smaller

shadow_color

Set the color of the shadow

shadow_opa

Set the opacity of the shadow. Value 0, `LV_OPA_0` or `LV_OPA_TRANSP` means fully transparent, 256, `LV_OPA_100` or `LV_OPA_COVER` means fully covering, other values or LV_OPA_10, LV_OPA_20, etc indicate semi-transparency.

Line

TODO

line_width

Set the width of the lines in pixel.

line_dash_width

Set the width of dashes in pixel. Note that dash works only on horizontal and vertical lines

line_dash_gap

Set the gap between dashes in pixel. Note that dash works only on horizontal and vertical lines

line_rounded

Make the end points of the lines rounded. `true`: rounded, `false`: perpendicular line ending

line_color

Set the color fo the lines.

line_opa

Set the opacity of the lines.

Arc

TODO

arc_width

Set the width (thickness) of the arcs in pixel.

arc_rounded

Make the end points of the arcs rounded. `true`: rounded, `false`: perpendicular line ending

arc_color

Set the color of the arc.

arc_opa

Set the opacity of the arcs.

arc_img_src

Set an image from which the arc will be masked out. It's useful to display complex effects on the arcs. Can be a pointer to `lv_img_dsc_t` or a path to a file

2.5.5 Scroll（滚动）

Overview（概述）

In LVGL scrolling works very intuitively: if an object is out of its parent content area (the size without paddings), the parent becomes scrollable and scrollbar(s) will appear. That's it.

Any object can be scrollable including `lv_obj_t`, `lv_img`, `lv_btn`, `lv_meter`, etc

The object can either be scrolled either horizontally or vertically in one stroke; diagonal scrolling is not possible.

在 LVGL 中，滚动的工作非常直观：如果对象超出其父内容区域（没有填充的大小），则父对象的空间可滚动并且会出现滚动条。仅此而已。

任何对象都可以滚动，包括 `lv_obj_t`、`lv_img`、`lv_btn`、`lv_meter` 等

对象可以一次水平或垂直滚动；对角滚动是不可能的。

Scrollbar（滚动条）

Mode（模式）

The scrollbars are displayed according to the set `mode`. The following `mode`s exist:

- `LV_SCROLLBAR_MODE_OFF` Never show the scrollbars
- `LV_SCROLLBAR_MODE_ON` Always show the scrollbars
- `LV_SCROLLBAR_MODE_ACTIVE` Show scroll bars while object is being scrolled
- `LV_SCROLLBAR_MODE_AUTO` Show scroll bars when the content is large enough to be scrolled

`lv_obj_set_scrollbar_mode(obj, LV_SCROLLBAR_MODE_...)` set the scrollbar mode on an object.

滚动条根据设置的 模式显示。有下面这几种 模式可以选择：

- `LV_SCROLLBAR_MODE_OFF` 从不显示滚动条
- `LV_SCROLLBAR_MODE_ON` 始终显示滚动条
- `LV_SCROLLBAR_MODE_ACTIVE` 在对象滚动时显示滚动条
- `LV_SCROLLBAR_MODE_AUTO` 当内容足够大可以滚动时显示滚动条

`lv_obj_set_scrollbar_mode(obj, LV_SCROLLBAR_MODE_...)` 在对象上设置滚动条模式。

Styling（样式）

The scrollbars have their own dedicated part, called LV_PART_SCROLLBAR. For example a scrollbar can turned to red like this:

滚动条有自己的专用部分，称为 LV_PART_SCROLLBAR 。例如，滚动条可以像这样变成红色：

```
static lv_style_t style_red;
lv_style_init(&style_red);
lv_style_set_bg_color(&style_red, lv_color_red());

...

lv_obj_add_style(obj, &style_red, LV_PART_SCROLLBAR);
```

The object goes to LV_STATE_SCROLLED state while it's being scrolled. It allows adding different style to the scrollbar or the object itself when scrolled. This code makes the scrollbar blue when the object is scrolled:

对象在滚动时进入 LV_STATE_SCROLLED 状态。它允许在滚动时向滚动条或对象本身添加不同的样式。当对象滚动时，此代码使滚动条变为蓝色：

```
static lv_style_t style_blue;
lv_style_init(&style_blue);
lv_style_set_bg_color(&style_red, lv_color_blue());

...

lv_obj_add_style(obj, &style_blue, LV_STATE_SCROLLED | LV_PART_SCROLLBAR);
```

Events(事件)

The following events are related to scrolling:

- LV_EVENT_SCROLL_BEGIN Scrolling begins
- LV_EVENT_SCROLL_END Scrolling ends
- LV_EVENT_SCROLL Scroll happened. Triggered on every position change. Scroll events

以下事件与滚动相关：

- LV_EVENT_SCROLL_BEGIN 开始滚动
- LV_EVENT_SCROLL_END 滚动结束
- LV_EVENT_SCROLL 滚动发生。每次位置变化时触发。

滚动事件

Basic example（基本示例）

TODO

Features of scrolling（滚动的特点）

Besides managing ”normal” scrolling there are many interesting and useful additional features too.

除了管理“正常”滚动之外，还有许多有趣且有用的附加功能。

Scrollable（滚动效果）

It’s possible to make an object non-scrollable with `lv_obj_clear_flag(obj, LV_OBJ_FLAG_SCROLLABLE)`.

Non-scrollable object can still propagate the scrolling (chain) to the parents.

The direction in which scrolling can happen can be controlled by `lv_obj_set_scroll_dir(obj, LV_DIR_...)`. The following values are possible for the direction:

- `LV_DIR_TOP` only scroll up
- `LV_DIR_LEFT` only scroll left
- `LV_DIR_BOTTOM` only scroll down
- `LV_DIR_RIGHT` only scroll right
- `LV_DIR_HOR` only scroll horizontally
- `LV_DIR_VER` only scroll vertically
- `LV_DIR_ALL` scroll any directions

OR-ed values are also possible. E.g. `LV_DIR_TOP | LV_DIR_LEFT`.

可以使用 `lv_obj_clear_flag(obj, LV_OBJ_FLAG_SCROLLABLE)` 使对象不可滚动。

不可滚动对象仍然可以将滚动（链）传播到父对象。

滚动发生的方向可以由 `lv_obj_set_scroll_dir(obj, LV_DIR_...)` 控制。方向可能有以下值：

- `LV_DIR_TOP` 只向上滚动
- `LV_DIR_LEFT` 只向左滚动
- `LV_DIR_BOTTOM` 只向下滚动
- `LV_DIR_RIGHT` 只向右滚动
- `LV_DIR_HOR` 只能水平滚动
- `LV_DIR_VER` 只能垂直滚动

- LV_DIR_ALL 滚动任何方向

可以用同时设置使用多个值（OR-ed）。例如。LV_DIR_TOP | LV_DIR_LEFT。

Scroll chain（滚动条）

If an object can't be scrolled further (e.g. it's content has reached the bottom most position) the scrolling is propagated to it's parent. If the parent an be scrolled in that direction than it will be scrolled instead. It propagets to the grandparent and grand-grandparents too.

The propagation on scrolling is called "scroll chaining" and it can be enabled/disabled with the LV_OBJ_FLAG_SCROLL_CHAIN flag. If chaining is disabled the propagation stops on the object and the parent(s) won't be scrolled.

如果对象无法进一步滚动（例如，它的内容已到达最底部的位置），则滚动将传播到其父级。如果父级在该方向上滚动，则它将改为滚动。它也传播给祖父母和祖父母。

滚动传播称为“滚动链接”，可以使用 LV_OBJ_FLAG_SCROLL_CHAIN 标志启用/禁用。如果禁用链接，则传播将停止在对象上，并且不会滚动父对象。

Scroll momentum（滚动惯性效果）

When the user scrolls an object and releases it, LVGL can emulate a momentum for the scrolling. It's like the object was thrown and scrolling slows down smoothly.

The scroll momentum can be enabled/disabled with the LV_OBJ_FLAG_SCROLL_MOMENTUM flag.

当用户滚动对象并释放它时，LVGL 可以模拟滚动的动量。就好像对象被抛出并且滚动平稳地减慢了速度。

可以使用“LV_OBJ_FLAG_SCROLL_MOMENTUM”标志启用/禁用滚动动量。

Elastic scroll（弹性卷轴效果）

Normally the content can't be scrolled inside the object. That is the top side of the content can't be below the top side of the object.

However, with LV_OBJ_FLAG_SCROLL_ELASTIC a fancy effect can be added when the user "over-scrolls" the content. The scrolling slows down, and the content can be scrolled inside the object. When the object is released the content scrolled in it will be animated back to the valid position.

通常内容不能在对象内滚动。即内容的顶部不能低于对象的顶部。

但是，使用 LV_OBJ_FLAG_SCROLL_ELASTIC 可以在用户“滚动”内容时添加奇特的效果。滚动变慢，内容可以在对象内部滚动。当对象被释放时，滚动到其中的内容将动画回到有效位置。

Snapping（捕捉）

The children of an object can be snapped according to specific rules when scrolling ends. Children can be made snappable individually with the `LV_OBJ_FLAG_SNAPPABLE` flag.

The object can align the snapped children in 4 ways:

- `LV_SCROLL_SNAP_NONE` Snapping is disabled. (default)
- `LV_SCROLL_SNAP_START` Align the children to the left/top side of the scrolled object
- `LV_SCROLL_SNAP_END` Align the children to the right/bottom side of the scrolled object
- `LV_SCROLL_SNAP_CENTER` Align the children to the center of the scrolled object

滚动结束时，可以根据特定规则捕捉对象的子项。可以使用 `LV_OBJ_FLAG_SNAPPABLE` 标志将子对象单独设置为可捕捉。

该对象可以通过 4 种方式对齐对齐的子项：

- `LV_SCROLL_SNAP_NONE` 捕捉被禁用。（默认）
- `LV_SCROLL_SNAP_START` 将子对象与滚动对象的左侧/顶部对齐
- `LV_SCROLL_SNAP_END` 将子对象与滚动对象的右侧/底部对齐
- `LV_SCROLL_SNAP_CENTER` 将子对象与滚动对象的中心对齐

The alignment can be set with `lv_obj_set_scroll_snap_x/y(obj, LV_SCROLL_SNAP_...)`:

Under the hood the following happens:

1. User scrolls an object and releases the screen
2. LVGL calculates where the scroll would end considering scroll momentum
3. LVGL finds the nearest scroll point
4. LVGL scrolls to the snap point with an animation

对齐可以用 `lv_obj_set_scroll_snap_x/y(obj, LV_SCROLL_SNAP_...)` 设置：

底层代码会发生以下情况：

1. 用户滚动对象并释放屏幕
2. LVGL 考虑滚动动量计算滚动结束的位置
3. LVGL 寻找最近的滚动点
4. LVGL 滚动到带动画的捕捉点

Scroll one(只滚动一个)

The ”scroll one” feature tells LVGL to allow scrolling only one snappable child at a time. So this requires to make the children snappable and set a scroll snap alignment different from `LV_SCROLL_SNAP_NONE`.

This feature can be enabled by the `LV_OBJ_FLAG_SCROLL_ONE` flag.

“只滚动一个 (Scroll one)” 功能告诉 LVGL 一次只允许滚动一个可捕捉的孩子。

因此，这需要使子项可捕捉并设置与 `LV_SCROLL_SNAP_NONE` 不同的滚动对齐对齐方式。此功能可以通过 `LV_OBJ_FLAG_SCROLL_ONE` 标志启用。

Scroll on focus（滚动焦点）

Imagine that there a lot of objects in a group that are on scrollable object. Pressing the ”Tab” button focuses the next object but it might be out of the visible area of the scrollable object. If the ”scroll on focus” features is enabled LVGL will automatically scroll to the objects to bring the children into the view. The scrolling happens recursively therefore even nested scrollable object are handled properly. The object will be scrolled to the view even if it’s on a different page of a tabview.

想象一下，一个组中有很多对象位于可滚动对象上。按 “Tab” 按钮聚焦下一个对象，但它可能超出可滚动对象的可见区域。如果启用了 “焦点滚动” 功能，LVGL 将自动滚动到对象以将子项带入视图。

滚动以递归方式发生，因此即使嵌套的可滚 动对象也能得到正确处理。即使对象位于 tabview 的不同页面上，它也会滚动到视图。

Scroll manually

The following API functions allow to manually scroll objects:

- `lv_obj_scroll_by(obj, x, y, LV_ANIM_ON/OFF)` scroll by `x` and `y` values
- `lv_obj_scroll_to(obj, x, y, LV_ANIM_ON/OFF)` scroll to bring the given coordinate to the top left corner
- `lv_obj_scroll_to_x(obj, x, LV_ANIM_ON/OFF)` scroll to bring the given coordinate to the left side
- `lv_obj_scroll_to_y(obj, y, LV_ANIM_ON/OFF)` scroll to bring the given coordinate to the left side

以下 API 函数允许手动滚动对象：

- `lv_obj_scroll_by(obj, x, y, LV_ANIM_ON/OFF)` 按 `x` 和 `y` 值滚动
- `lv_obj_scroll_to(obj, x, y, LV_ANIM_ON/OFF)` 滚动以将给定的坐标带到左上角
- `lv_obj_scroll_to_x(obj, x, LV_ANIM_ON/OFF)` 滚动以将给定的坐标带到左侧
- `lv_obj_scroll_to_y(obj, y, LV_ANIM_ON/OFF)` 滚动以将给定的坐标带到左侧

Self size（自身尺寸）

Self size is a property of an object. Normally, the user shouldn't use this parameter but if a custom widget is created it might be useful.

In short, self size tell the size of the content. To understand it better take the example of a table. Let's say it has 10 rows each with 50 px height. So the total height of the content is 500 px. In other words the "self height" is 500 px. If the user sets only 200 px height for the table LVGL will see that the self size is larger and make the table scrollable.

It means not only the children can make an object scrollable but a larger self size too.

LVGL uses the `LV_EVENT_GET_SELF_SIZE` event to get the self size of an object. Here is an example to see how to handle the event

自身大小是对象的属性。通常，用户不应使用此参数，但如果创建了自定义小、部件，它可能会很有用。

简而言之，自我大小告诉内容的大小。为了更好地理解它，举一个表格的例子。假设它有 10 行，每行 50 像素高度。所以内容的总高度是 500 px。换句话说，“自身高度” 是 500 像素。如果用户只为表格设置 200 像素高度，LVGL 将看到自身尺寸更大并使表格可滚动。

这意味着不仅孩子们可以使对象可滚动，而且还可以使自身尺寸更大。

LVGL 使用 `LV_EVENT_GET_SELF_SIZE` 事件来获取对象的自身大小。下面是一个例子，看看如何处理事件

```
if(event_code == LV_EVENT_GET_SELF_SIZE) {
        lv_point_t * p = lv_event_get_param(e);

  //If x or y < 0 then it doesn't neesd to be calculated now
  if(p->x >= 0) {
    p->x = 200;         //Set or calculate the self width
  }

  if(p->y >= 0) {
    p->y = 50;        //Set or calculate the self height
  }
}
```

Examples

Nested scrolling

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

/**
```

(下页继续)

(续上页)

```
 * Demonstrate how scrolling appears automatically
 */
void lv_example_scroll_1(void)
{
    /*Create an object with the new style*/
    lv_obj_t * panel = lv_obj_create(lv_scr_act());
    lv_obj_set_size(panel, 200, 200);
    lv_obj_center(panel);

    lv_obj_t * child;
    lv_obj_t * label;

    child = lv_obj_create(panel);
    lv_obj_set_pos(child, 0, 0);
    lv_obj_set_size(child, 70, 70);
    label = lv_label_create(child);
    lv_label_set_text(label, "Zero");
    lv_obj_center(label);

    child = lv_obj_create(panel);
    lv_obj_set_pos(child, 160, 80);
    lv_obj_set_size(child, 80, 80);

    lv_obj_t * child2 = lv_btn_create(child);
    lv_obj_set_size(child2, 100, 50);

    label = lv_label_create(child2);
    lv_label_set_text(label, "Right");
    lv_obj_center(label);

    child = lv_obj_create(panel);
    lv_obj_set_pos(child, 40, 160);
    lv_obj_set_size(child, 100, 70);
    label = lv_label_create(child);
    lv_label_set_text(label, "Bottom");
    lv_obj_center(label);
}

#endif
```

```
#
# Demonstrate how scrolling appears automatically
#
```

(下页继续)

(续上页)

```
# Create an object with the new style
panel = lv.obj(lv.scr_act())
panel.set_size(200, 200)
panel.center()

child = lv.obj(panel)
child.set_pos(0, 0)
label = lv.label(child)
label.set_text("Zero")
label.center()

child = lv.obj(panel)
child.set_pos(-40, 100)
label = lv.label(child)
label.set_text("Left")
label.center()

child = lv.obj(panel)
child.set_pos(90, -30)
label = lv.label(child)
label.set_text("Top")
label.center()

child = lv.obj(panel)
child.set_pos(150, 80)
label = lv.label(child)
label.set_text("Right")
label.center()

child = lv.obj(panel)
child.set_pos(60, 170)
label = lv.label(child)
label.set_text("Bottom")
label.center()
```

Snapping

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_FLEX

static void sw_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * sw = lv_event_get_target(e);

    if(code == LV_EVENT_VALUE_CHANGED) {
        lv_obj_t * list = lv_event_get_user_data(e);

        if(lv_obj_has_state(sw, LV_STATE_CHECKED)) lv_obj_add_flag(list, LV_OBJ_FLAG_
↪SCROLL_ONE);
        else lv_obj_clear_flag(list, LV_OBJ_FLAG_SCROLL_ONE);
    }
}

/**
 * Show an example to scroll snap
 */
void lv_example_scroll_2(void)
{
    lv_obj_t * panel = lv_obj_create(lv_scr_act());
    lv_obj_set_size(panel, 280, 120);
    lv_obj_set_scroll_snap_x(panel, LV_SCROLL_SNAP_CENTER);
    lv_obj_set_flex_flow(panel, LV_FLEX_FLOW_ROW);
    lv_obj_align(panel, LV_ALIGN_CENTER, 0, 20);

    uint32_t i;
    for(i = 0; i < 10; i++) {
        lv_obj_t * btn = lv_btn_create(panel);
        lv_obj_set_size(btn, 150, lv_pct(100));

        lv_obj_t * label = lv_label_create(btn);
        if(i == 3) {
            lv_label_set_text_fmt(label, "Panel %"LV_PRIu32"\nno snap", i);
            lv_obj_clear_flag(btn, LV_OBJ_FLAG_SNAPPABLE);
        } else {
            lv_label_set_text_fmt(label, "Panel %"LV_PRIu32, i);
        }

        lv_obj_center(label);
```

(下页继续)

(续上页)

```
    }
    lv_obj_update_snap(panel, LV_ANIM_ON);

#if LV_USE_SWITCH
    /*Switch between "One scroll" and "Normal scroll" mode*/
    lv_obj_t * sw = lv_switch_create(lv_scr_act());
    lv_obj_align(sw, LV_ALIGN_TOP_RIGHT, -20, 10);
    lv_obj_add_event_cb(sw, sw_event_cb, LV_EVENT_ALL, panel);
    lv_obj_t * label = lv_label_create(lv_scr_act());
    lv_label_set_text(label, "One scroll");
    lv_obj_align_to(label, sw, LV_ALIGN_OUT_BOTTOM_MID, 0, 5);
#endif
}

#endif
```

```
def sw_event_cb(e,panel):

    code = e.get_code()
    sw = e.get_target()

    if code == lv.EVENT.VALUE_CHANGED:

        if sw.has_state(lv.STATE.CHECKED):
            panel.add_flag(lv.obj.FLAG.SCROLL_ONE)
        else:
            panel.clear_flag(lv.obj.FLAG.SCROLL_ONE)



#
# Show an example to scroll snap
#

panel = lv.obj(lv.scr_act())
panel.set_size(280, 150)
panel.set_scroll_snap_x(lv.SCROLL_SNAP.CENTER)
panel.set_flex_flow(lv.FLEX_FLOW.ROW)
panel.center()

for i in range(10):
    btn = lv.btn(panel)
    btn.set_size(150, 100)
```

(下页继续)

(续上页)

```
    label = lv.label(btn)
    if i == 3:
        label.set_text("Panel {:d}\nno snap".format(i))
        btn.clear_flag(lv.obj.FLAG.SNAPPABLE)
    else:
        label.set_text("Panel {:d}".format(i))
    label.center()

panel.update_snap(lv.ANIM.ON)


# Switch between "One scroll" and "Normal scroll" mode
sw = lv.switch(lv.scr_act())
sw.align(lv.ALIGN.TOP_RIGHT, -20, 10)
sw.add_event_cb(lambda evt:  sw_event_cb(evt,panel), lv.EVENT.ALL, None)
label = lv.label(lv.scr_act())
label.set_text("One scroll")
label.align_to(sw, lv.ALIGN.OUT_BOTTOM_MID, 0, 5)
```

Floating button

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_LIST

static uint32_t btn_cnt = 1;

static void float_btn_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * float_btn = lv_event_get_target(e);

    if(code == LV_EVENT_CLICKED) {
        lv_obj_t * list = lv_event_get_user_data(e);
        char buf[32];
        lv_snprintf(buf, sizeof(buf), "Track %d", (int)btn_cnt);
        lv_obj_t * list_btn = lv_list_add_btn(list, LV_SYMBOL_AUDIO, buf);
        btn_cnt++;

        lv_obj_move_foreground(float_btn);
```

(下页继续)

(续上页)

```
        lv_obj_scroll_to_view(list_btn, LV_ANIM_ON);
    }
}

/**
 * Create a list a with a floating button
 */
void lv_example_scroll_3(void)
{
    lv_obj_t * list = lv_list_create(lv_scr_act());
    lv_obj_set_size(list, 280, 220);
    lv_obj_center(list);

    for(btn_cnt = 1; btn_cnt <= 2; btn_cnt++) {
        char buf[32];
        lv_snprintf(buf, sizeof(buf), "Track %d", (int)btn_cnt);
        lv_list_add_btn(list, LV_SYMBOL_AUDIO, buf);
    }

    lv_obj_t * float_btn = lv_btn_create(list);
    lv_obj_set_size(float_btn, 50, 50);
    lv_obj_add_flag(float_btn, LV_OBJ_FLAG_FLOATING);
    lv_obj_align(float_btn, LV_ALIGN_BOTTOM_RIGHT, 0, -lv_obj_get_style_pad_
↪right(list, LV_PART_MAIN));
    lv_obj_add_event_cb(float_btn, float_btn_event_cb, LV_EVENT_ALL, list);
    lv_obj_set_style_radius(float_btn, LV_RADIUS_CIRCLE, 0);
    lv_obj_set_style_bg_img_src(float_btn, LV_SYMBOL_PLUS, 0);
    lv_obj_set_style_text_font(float_btn, lv_theme_get_font_large(float_btn), 0);
}

#endif
```

```
class ScrollExample_3():
    def __init__(self):
        self.btn_cnt = 1
        #
        # Create a list a with a floating button
        #

        list = lv.list(lv.scr_act())
        list.set_size(280, 220)
        list.center()
```

(下页继续)

(续上页)

```
        for btn_cnt in range(2):
            list.add_btn(lv.SYMBOL.AUDIO,"Track {:d}".format(btn_cnt))

        float_btn = lv.btn(list)
        float_btn.set_size(50, 50)
        float_btn.add_flag(lv.obj.FLAG.FLOATING)
        float_btn.align(lv.ALIGN.BOTTOM_RIGHT, 0, -list.get_style_pad_right(lv.PART.
↪MAIN))
        float_btn.add_event_cb(lambda evt: self.float_btn_event_cb(evt,list), lv.
↪EVENT.ALL, None)
        float_btn.set_style_radius(lv.RADIUS.CIRCLE, 0)
        float_btn.set_style_bg_img_src(lv.SYMBOL.PLUS, 0)
        float_btn.set_style_text_font(lv.theme_get_font_large(float_btn), 0)

    def float_btn_event_cb(self,e,list):
        code = e.get_code()
        float_btn = e.get_target()

        if code == lv.EVENT.CLICKED:
            list_btn = list.add_btn(lv.SYMBOL.AUDIO, "Track {:d}".format(self.btn_
↪cnt))
            self.btn_cnt += 1

            float_btn.move_foreground()

            list_btn.scroll_to_view(lv.ANIM.ON)

scroll_example_3 = ScrollExample_3()
```

Styling the scrollbars

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_LIST


/**
 * Styling the scrollbars
 */
void lv_example_scroll_4(void)
{
```

(下页继续)

(续上页)

```
    lv_obj_t * obj = lv_obj_create(lv_scr_act());
    lv_obj_set_size(obj, 200, 100);
    lv_obj_center(obj);

    lv_obj_t * label = lv_label_create(obj);
    lv_label_set_text(label,
            "Lorem ipsum dolor sit amet, consectetur adipiscing elit.\n"
            "Etiam dictum, tortor vestibulum lacinia laoreet, mi neque consectetur␣
↪neque, vel mattis odio dolor egestas ligula. \n"
            "Sed vestibulum sapien nulla, id convallis ex porttitor nec. \n"
            "Duis et massa eu libero accumsan faucibus a in arcu. \n"
            "Ut pulvinar odio lorem, vel tempus turpis condimentum quis. Nam␣
↪consectetur condimentum sem in auctor. \n"
            "Sed nisl augue, venenatis in blandit et, gravida ac tortor. \n"
            "Etiam dapibus elementum suscipit. \n"
            "Proin mollis sollicitudin convallis. \n"
            "Integer dapibus tempus arcu nec viverra. \n"
            "Donec molestie nulla enim, eu interdum velit placerat quis. \n"
            "Donec id efficitur risus, at molestie turpis. \n"
            "Suspendisse vestibulum consectetur nunc ut commodo. \n"
            "Fusce molestie rhoncus nisi sit amet tincidunt. \n"
            "Suspendisse a nunc ut magna ornare volutpat.");


    /*Remove the style of scrollbar to have clean start*/
    lv_obj_remove_style(obj, NULL, LV_PART_SCROLLBAR | LV_STATE_ANY);

    /*Create a transition the animate the some properties on state change*/
    static const lv_style_prop_t props[] = {LV_STYLE_BG_OPA, LV_STYLE_WIDTH, 0};
    static lv_style_transition_dsc_t trans;
    lv_style_transition_dsc_init(&trans, props, lv_anim_path_linear, 200, 0, NULL);

    /*Create a style for the scrollbars*/
    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_width(&style, 4);      /*Width of the scrollbar*/
    lv_style_set_pad_right(&style, 5);  /*Space from the parallel side*/
    lv_style_set_pad_top(&style, 5);    /*Space from the perpendicular side*/

    lv_style_set_radius(&style, 2);
    lv_style_set_bg_opa(&style, LV_OPA_70);
    lv_style_set_bg_color(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_border_color(&style, lv_palette_darken(LV_PALETTE_BLUE, 3));
```

(下页继续)

(续上页)

```
    lv_style_set_border_width(&style, 2);
    lv_style_set_shadow_width(&style, 8);
    lv_style_set_shadow_spread(&style, 2);
    lv_style_set_shadow_color(&style, lv_palette_darken(LV_PALETTE_BLUE, 1));

    lv_style_set_transition(&style, &trans);

    /*Make the scrollbars wider and use 100% opacity when scrolled*/
    static lv_style_t style_scrolled;
    lv_style_init(&style_scrolled);
    lv_style_set_width(&style_scrolled, 8);
    lv_style_set_bg_opa(&style_scrolled, LV_OPA_COVER);

    lv_obj_add_style(obj, &style, LV_PART_SCROLLBAR);
    lv_obj_add_style(obj, &style_scrolled, LV_PART_SCROLLBAR | LV_STATE_SCROLLED);
}

#endif
```

```
#
# Styling the scrollbars
#
obj = lv.obj(lv.scr_act())
obj.set_size(200, 100)
obj.center()

label = lv.label(obj)
label.set_text(
"""
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Etiam dictum, tortor vestibulum lacinia laoreet, mi neque consectetur neque, vel␣
↪mattis odio dolor egestas ligula.
Sed vestibulum sapien nulla, id convallis ex porttitor nec.
Duis et massa eu libero accumsan faucibus a in arcu.
Ut pulvinar odio lorem, vel tempus turpis condimentum quis. Nam consectetur␣
↪condimentum sem in auctor.
Sed nisl augue, venenatis in blandit et, gravida ac tortor.
Etiam dapibus elementum suscipit.
Proin mollis sollicitudin convallis.
Integer dapibus tempus arcu nec viverra.
Donec molestie nulla enim, eu interdum velit placerat quis.
Donec id efficitur risus, at molestie turpis.
Suspendisse vestibulum consectetur nunc ut commodo.
```

(下页继续)

(续上页)

```
Fusce molestie rhoncus nisi sit amet tincidunt.
Suspendisse a nunc ut magna ornare volutpat.
""")


# Remove the style of scrollbar to have clean start
obj.remove_style(None, lv.PART.SCROLLBAR | lv.STATE.ANY)

# Create a transition the animate the some properties on state change
props = [lv.STYLE.BG_OPA, lv.STYLE.WIDTH, 0]
trans = lv.style_transition_dsc_t()
trans.init(props, lv.anim_t.path_linear, 200, 0, None)

# Create a style for the scrollbars
style = lv.style_t()
style.init()
style.set_width(4)              # Width of the scrollbar
style.set_pad_right(5)          # Space from the parallel side
style.set_pad_top(5)            # Space from the perpendicular side

style.set_radius(2)
style.set_bg_opa(lv.OPA._70)
style.set_bg_color(lv.palette_main(lv.PALETTE.BLUE))
style.set_border_color(lv.palette_darken(lv.PALETTE.BLUE, 3))
style.set_border_width(2)
style.set_shadow_width(8)
style.set_shadow_spread(2)
style.set_shadow_color(lv.palette_darken(lv.PALETTE.BLUE, 1))

style.set_transition(trans)

# Make the scrollbars wider and use 100% opacity when scrolled
style_scrolled = lv.style_t()
style_scrolled.init()
style_scrolled.set_width(8)
style_scrolled.set_bg_opa(lv.OPA.COVER)

obj.add_style(style, lv.PART.SCROLLBAR)
obj.add_style(style_scrolled, lv.PART.SCROLLBAR | lv.STATE.SCROLLED)
```

Right to left scrolling

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_FONT_DEJAVU_16_PERSIAN_HEBREW


/**
 * Scrolling with Right To Left base direction
 */
void lv_example_scroll_5(void)
{
    lv_obj_t * obj = lv_obj_create(lv_scr_act());
    lv_obj_set_style_base_dir(obj, LV_BASE_DIR_RTL, 0);
    lv_obj_set_size(obj, 200, 100);
    lv_obj_center(obj);

    lv_obj_t * label = lv_label_create(obj);
    lv_label_set_text(label,"رلرتنکروُکرمیه) :یسیلگنا Microcontroller) یاهنوگ␣
↪هدنزادرپزیر تسا هک یاراد ظفاح یسرتسد یفدا (RAM) و ظفاح یندناوختفلق (ROM)، ،رمیات␣
↪یاهروپ یدورو و یجورخ (I/O) و هاگرد یبیترپ (Serial Port تروپ ،(لایرس نورد دوخ هشارت␣
↪،تسا و دنتیم هب یاهنتا یارازبا رگید را لرتنک دنک. هب ترابع رگید ک ،رلرتنکروکرمیم␣
↪رادم عمتجم یکچوک تسا هک زا ک CPU یکچوک و یزاجا یرگید دننام ،رمیات یاهاگرد رد و␣
↪یجورخ گولاونآ و یتاجیدو ظفاح لکشت تساهدش.");
    lv_obj_set_width(label, 400);
    lv_obj_set_style_text_font(label, &lv_font_dejavu_16_persian_hebrew, 0);

}

#endif
```

```
#
# Scrolling with Right To Left base direction
#
obj = lv.obj(lv.scr_act())
obj.set_style_base_dir(lv.BASE_DIR.RTL, 0)
obj.set_size(200, 100)
obj.center()

label = lv.label(obj)
label.set_text("رلرتنکروُکرمیه) :یسیلگنا Microcontroller) یاهنوگ هدنزادرپزیر تسا هک␣
↪یارا ظفاح یسرتسد یفدا (RAM) و ظفاح یندناوختفلق (ROM)، ،رمیات یاهروپ یدورو و␣
↪یجورخ (I/O) و هاگرد یبیترپ (Serial Port تروپ ،(لایرس نورد دوخ هشارت ،تسا و دنتیم␣
↪هب یاهنتا یارازبا رگید را لرتنک دنک. هب ترابع رگید ک ،رلرتنکروکرمیم رادم عمتجم␣
↪یکچوک تسا هک زا ک CPU یکچوک و یزاجا یرگید دننام ،رمیات یاهاگرد رد و یجورخ␣
↪یجولاونآ و یتاجیدو ظفاح لکشت تساهدش.")
```

(下页继续)

(续上页)

```
label.set_width(400)
label.set_style_text_font(lv.font_dejavu_16_persian_hebrew, 0)
```

Translate on scroll

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_FLEX

static void scroll_event_cb(lv_event_t * e)
{
    lv_obj_t * cont = lv_event_get_target(e);

    lv_area_t cont_a;
    lv_obj_get_coords(cont, &cont_a);
    lv_coord_t cont_y_center = cont_a.y1 + lv_area_get_height(&cont_a) / 2;

    lv_coord_t r = lv_obj_get_height(cont) * 7 / 10;
    uint32_t i;
    uint32_t child_cnt = lv_obj_get_child_cnt(cont);
    for(i = 0; i < child_cnt; i++) {
        lv_obj_t * child = lv_obj_get_child(cont, i);
        lv_area_t child_a;
        lv_obj_get_coords(child, &child_a);

        lv_coord_t child_y_center = child_a.y1 + lv_area_get_height(&child_a) / 2;

        lv_coord_t diff_y = child_y_center - cont_y_center;
        diff_y = LV_ABS(diff_y);

        /*Get the x of diff_y on a circle.*/
        lv_coord_t x;
        /*If diff_y is out of the circle use the last point of the circle (the␣
↪radius)*/
        if(diff_y >= r) {
            x = r;
        } else {
            /*Use Pythagoras theorem to get x from radius and y*/
            uint32_t x_sqr = r * r - diff_y * diff_y;
            lv_sqrt_res_t res;
            lv_sqrt(x_sqr, &res, 0x8000);   /*Use lvgl's built in sqrt root function*/
            x = r - res.i;
```

(下页继续)

(续上页)

```
        }

        /*Translate the item by the calculated X coordinate*/
        lv_obj_set_style_translate_x(child, x, 0);

        /*Use some opacity with larger translations*/
        lv_opa_t opa = lv_map(x, 0, r, LV_OPA_TRANSP, LV_OPA_COVER);
        lv_obj_set_style_opa(child, LV_OPA_COVER - opa, 0);
    }
}

/**
 * Translate the object as they scroll
 */
void lv_example_scroll_6(void)
{
    lv_obj_t * cont = lv_obj_create(lv_scr_act());
    lv_obj_set_size(cont, 200, 200);
    lv_obj_center(cont);
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_COLUMN);
    lv_obj_add_event_cb(cont, scroll_event_cb, LV_EVENT_SCROLL, NULL);
    lv_obj_set_style_radius(cont, LV_RADIUS_CIRCLE, 0);
    lv_obj_set_style_clip_corner(cont, true, 0);
    lv_obj_set_scroll_dir(cont, LV_DIR_VER);
    lv_obj_set_scroll_snap_y(cont, LV_SCROLL_SNAP_CENTER);
    lv_obj_set_scrollbar_mode(cont, LV_SCROLLBAR_MODE_OFF);

    uint32_t i;
    for(i = 0; i < 20; i++) {
        lv_obj_t * btn = lv_btn_create(cont);
        lv_obj_set_width(btn, lv_pct(100));

        lv_obj_t * label = lv_label_create(btn);
        lv_label_set_text_fmt(label, "Button %"LV_PRIu32, i);
    }

    /*Update the buttons position manually for first*/
    lv_event_send(cont, LV_EVENT_SCROLL, NULL);

    /*Be sure the fist button is in the middle*/
    lv_obj_scroll_to_view(lv_obj_get_child(cont, 0), LV_ANIM_OFF);
}

#endif
```

```
def scroll_event_cb(e):

    cont = e.get_target()

    cont_a = lv.area_t()
    cont.get_coords(cont_a)
    cont_y_center = cont_a.y1 + cont_a.get_height() // 2

    r = cont.get_height() * 7 // 10

    child_cnt = cont.get_child_cnt()
    for i in range(child_cnt):
        child = cont.get_child(i)
        child_a = lv.area_t()
        child.get_coords(child_a)

        child_y_center = child_a.y1 + child_a.get_height() // 2

        diff_y = child_y_center - cont_y_center
        diff_y = abs(diff_y)

        # Get the x of diff_y on a circle.

        # If diff_y is out of the circle use the last point of the circle (the radius)
        if diff_y >= r:
            x = r
        else:
            # Use Pythagoras theorem to get x from radius and y
            x_sqr = r * r - diff_y * diff_y
            res = lv.sqrt_res_t()
            lv.sqrt(x_sqr, res, 0x8000)   # Use lvgl's built in sqrt root function
            x = r - res.i

        # Translate the item by the calculated X coordinate
        child.set_style_translate_x(x, 0)

        # Use some opacity with larger translations
        opa = lv.map(x, 0, r, lv.OPA.TRANSP, lv.OPA.COVER)
        child.set_style_opa(lv.OPA.COVER - opa, 0)

#
# Translate the object as they scroll
#
```

(下页继续)

(续上页)

```
cont = lv.obj(lv.scr_act())
cont.set_size(200, 200)
cont.center()
cont.set_flex_flow(lv.FLEX_FLOW.COLUMN)
cont.add_event_cb(scroll_event_cb, lv.EVENT.SCROLL, None)
cont.set_style_radius(lv.RADIUS.CIRCLE, 0)
cont.set_style_clip_corner(True, 0)
cont.set_scroll_dir(lv.DIR.VER)
cont.set_scroll_snap_y(lv.SCROLL_SNAP.CENTER)
cont.set_scrollbar_mode(lv.SCROLLBAR_MODE.OFF)

for i in range(20):
    btn = lv.btn(cont)
    btn.set_width(lv.pct(100))

    label = lv.label(btn)
    label.set_text("Button " + str(i))

    # Update the buttons position manually for first*
    lv.event_send(cont, lv.EVENT.SCROLL, None)

    # Be sure the fist button is in the middle
    #lv.obj.scroll_to_view(cont.get_child(0), lv.ANIM.OFF)
    cont.get_child(0).scroll_to_view(lv.ANIM.OFF)
```

2.5.6 Layers（图层）

Order of creation（图层顺序）

By default, LVGL draws new objects on top of old objects.

For example, assume we added a button to a parent object named button1 and then another button named button2. Then button1 (with its child object(s)) will be in the background and can be covered by button2 and its children.

默认情况下，LVGL 在旧对象之上绘制新对象。

例如，假设我们向名为 button1 的父对象添加了一个按钮，然后添加了另一个名为 button2 的按钮。然后 button1（及其子对象）将在背景中并且可以被 button2 及其子对象覆盖。


```
/*Create a screen*/
lv_obj_t * scr = lv_obj_create(NULL, NULL);
lv_scr_load(scr);           /*Load the screen*/

/*Create 2 buttons*/
lv_obj_t * btn1 = lv_btn_create(scr, NULL);         /*Create a button on the screen*/
lv_btn_set_fit(btn1, true, true);                   /*Enable to automatically set the␣
↪size according to the content*/
lv_obj_set_pos(btn1, 60, 40);                         /*Set the position of the␣
↪button*/

lv_obj_t * btn2 = lv_btn_create(scr, btn1);         /*Copy the first button*/
lv_obj_set_pos(btn2, 180, 80);                     /*Set the position of the button*/

/*Add labels to the buttons*/
lv_obj_t * label1 = lv_label_create(btn1, NULL);        /*Create a label on the first␣
↪button*/
lv_label_set_text(label1, "Button 1");                  /*Set the text of the label*/

lv_obj_t * label2 = lv_label_create(btn2, NULL);          /*Create a label on the␣
↪second button*/
lv_label_set_text(label2, "Button 2");                    /*Set the text of the␣
↪label*/

/*Delete the second label*/
lv_obj_del(label2);
```

Bring to the foreground（前台展示）

There are 2 explicit way to bring an object to the foreground:

- Use `lv_obj_move_foreground(obj)` to explicitly tell the library to bring an object to the foreground. Similarly, use `lv_obj_move_background(obj)` to move to the background.
- When `lv_obj_set_parent(obj, new_parent)` is used, `obj` will be on the foreground on the `new_parent`.

有两种显式方法可以将对象置于前台：

- 使用 `lv_obj_move_foreground(obj)` 明确告诉库将一个对象带到前台。同样，使用 `lv_obj_move_background(obj)` 移动到背景。
- 当使用 `lv_obj_set_parent(obj, new_parent)` 时，`obj` 将位于 `new_parent` 的前台。

Top and sys layers（顶层和系统层）

LVGL uses two special layers named as `layer_top` and `layer_sys`. Both are visible and common on all screens of a display. **They are not, however, shared among multiple physical displays.** The `layer_top` is always on top of the default screen (`lv_scr_act()`), and `layer_sys` is on top of `layer_top`.

The `layer_top` can be used by the user to create some content visible everywhere. For example, a menu bar, a pop-up, etc. If the `click` attribute is enabled, then `layer_top` will absorb all user click and acts as a modal.

LVGL 使用名为 “layer_top” 和 “layer_sys” 的两个特殊层。两者在显示器的所有屏幕上都是可见的和通用的。**然而，它们不会在多个物理显示器之间共享**。`layer_top` 始终位于默认屏幕的顶部（`lv_scr_act()`），而 `layer_sys` 位于 `layer_top` 的顶部。

用户可以使用 `layer_top` 来创建一些随处可见的内容。例如，菜单栏、弹出窗口等。如果启用了 `click` 属性，那么 `layer_top` 将吸收所有用户点击并充当模态。

```
lv_obj_set_click(lv_layer_top(), true);
```

The `layer_sys` is also used for similar purposes on LVGL. For example, it places the mouse cursor above all layers to be sure it's always visible.

`layer_sys` 也用于 LVGL 的类似目的。例如，它将鼠标光标放在所有图层上方以确保它始终可见。

2.5.7 Events（事件）

Events are triggered in LVGL when something happens which might be interesting to the user, e.g. when an object

- is clicked
- is scrolled
- has its value changed
- is redrawn, etc.

当发生用户可能感兴趣的事情时，LVGL 中会触发事件，例如当一个对象

- 被点击
- 滚动
- 数值被改变
- 重绘等。

Add events to the object

The user can assign callback functions to an object to see its events. In practice, it looks like this:

用户可以为对象分配回调函数以查看其事件。在实践中，它看起来像这样：

```
lv_obj_t * btn = lv_btn_create(lv_scr_act());
lv_obj_add_event_cb(btn, my_event_cb, LV_EVENT_CLICKED, NULL);   /*Assign an event␣
↪callback*/

...

static void my_event_cb(lv_event_t * event)
{
    printf("Clicked\n");
}
```

In the example `LV_EVENT_CLICKED` means that only the click event will call `my_event_cb`. See the *list of event codes* for all the options. `LV_EVENT_ALL` can be used to receive all the events.

The last parameter of `lv_obj_add_event_cb` is a pointer to any custom data that will be available in the event. It will be described later in more detail.

More events can be added to an object, like this:

在示例中 `LV_EVENT_CLICKED` 意味着只有当对象发生点击事件时，就会触发事件从而进入事件处理回调函数 `my_event_cb` 。有关所有选项，请参阅事件代码列表。`LV_EVENT_ALL` 可用于接收所有事件。

`lv_obj_add_event_cb` 的最后一个参数是指向事件中可用的任何自定义数据的指针。稍后将更详细地描述。

可以向一个对象添加更多事件，如下所示：

```
lv_obj_add_event_cb(obj, my_event_cb_1, LV_EVENT_CLICKED, NULL);
lv_obj_add_event_cb(obj, my_event_cb_2, LV_EVENT_PRESSED, NULL);
lv_obj_add_event_cb(obj, my_event_cb_3, LV_EVENT_ALL, NULL);                /*No␣
↪filtering, receive all events*/
```

Even the same event callback can be used on an object with different `user_data`. For example:

如果传入的用户数据不一样，一个对象可以绑定同一个事件回调函数多次。例如：

```
lv_obj_add_event_cb(obj, increment_on_click, LV_EVENT_CLICKED, &num1);
lv_obj_add_event_cb(obj, increment_on_click, LV_EVENT_CLICKED, &num2);
```

The events will be called in the order as they were added.

More objects can use the same *event callback*.

这些事件将按照添加的顺序被调用。

更多的对象可以使用相同的 事件回调。

Remove event(s) from an object(从对象中删除事件)

Events can be removed from an object with the `lv_obj_remove_event_cb(obj, event_cb)` function or `lv_obj_remove_event_dsc(obj, event_dsc)`. `event_dsc` is a pointer returned by `lv_obj_add_event_cb`.

可以使用 `lv_obj_remove_event_cb(obj, event_cb)` 函数或 `lv_obj_remove_event_dsc(obj, event_dsc)` 从对象中删除事件。`event_dsc` 是一个由 `lv_obj_add_event_cb` 返回的指针。

Event codes

The event codes can be grouped into these categories:

- Input device events
- Drawing events
- Other events
- Special events
- Custom events

All objects (such as Buttons/Labels/Sliders etc.) regardless their type receive the *Input device*, *Drawing* and *Other* events.

However the *Special events* are specific to a particular widget type. See the *widgets' documentation* to learn when they are sent,

Custom events are added by the user and therefore these are never sent by LVGL.

The following event codes exist:

事件代码可以分为以下几类：

- 输入设备事件 (Input device events)
- 绘图事件 (Drawing events)
- 其他事件 (Special events)
- 特殊事件 (Other events)
- 自定义事件 (Custom events)

请查阅源码：lvgl/src/core/lv_event.h (lv_event_code_t)

所有对象（例如按钮/标签/滑块等），无论其类型如何，都会接收 *Input device*、*Drawing* 和 *Other* 事件。

然而，特殊事件特定于特定的小部件类型。查看*widgets' 文档* 了解何时发送，

自定义事件由用户添加，因此这些事件永远不会由 LVGL 发送。

存在以下事件代码：

Input device events（输入设备事件）

- `LV_EVENT_PRESSED` The object has been pressed
- `LV_EVENT_PRESSING` The object is being pressed (called continuously while pressing)
- `LV_EVENT_PRESS_LOST` The object is still being pressed but slid cursor/finger off of the object
- `LV_EVENT_SHORT_CLICKED` The object was pressed for a short period of time, then released it. Not called if scrolled.
- `LV_EVENT_LONG_PRESSED` Object has been pressed for at least the `long_press_time` specified in the input device driver. Not called if scrolled.
- `LV_EVENT_LONG_PRESSED_REPEAT` Called after `long_press_time` in every `long_press_repeat_time` ms. Not called if scrolled.
- `LV_EVENT_CLICKED` Called on release if the object did not scroll (regardless of long press)
- `LV_EVENT_RELEASED` Called in every case when the object has been released
- `LV_EVENT_SCROLL_BEGIN` Scrolling begins. The event paramter is `NULL` or an `lv_anim_t *` with the scroll animation descriptor to modify if required.
- `LV_EVENT_SCROLL_END` Scrolling ends.

- LV_EVENT_SCROLL The object was scrolled
- LV_EVENT_GESTURE A gesture is detected. Get the gesture with `lv_indev_get_gesture_dir(lv_indev_get_act());`
- LV_EVENT_KEY A key is sent to the object. Get the key with `lv_indev_get_key(lv_indev_get_act());`
- LV_EVENT_FOCUSED The object is focused
- LV_EVENT_DEFOCUSED The object is defocused
- LV_EVENT_LEAVE The object is defocused but still selected
- LV_EVENT_HIT_TEST Perform advanced hit-testing. Use `lv_hit_test_info_t * a = lv_event_get_hit_test_info(e)` and check if `a->point` can click the object or not. If not set `a->res = false`
- LV_EVENT_PRESSED 对象已被按下
- LV_EVENT_PRESSING 对象被按下（按下时连续调用）
- LV_EVENT_PRESS_LOST 对象仍被按下，但光标/手指已滑离对象
- LV_EVENT_SHORT_CLICKED 对象被按下一小段时间，然后释放它。如果滚动则不会调用。
- LV_EVENT_LONG_PRESSED 对象已按下输入设备驱动程序中指定的至少 `long_press_time`。如果滚动则不会调用。
- LV_EVENT_LONG_PRESSED_REPEAT 在每个 `long_press_repeat_time` 毫秒的 `long_press_time` 之后调用。如果滚动则不会调用。
- LV_EVENT_CLICKED 如果对象没有滚动，则在释放时调用（无论是否长按）
- LV_EVENT_RELEASED 在对象被释放后的每种情况下调用
- LV_EVENT_SCROLL_BEGIN 开始滚动。事件参数是 NULL 或 `lv_anim_t *`，如果需要，可以修改滚动动画描述符。
- LV_EVENT_SCROLL_END 滚动结束。
- LV_EVENT_SCROLL 对象被滚动
- LV_EVENT_GESTURE 检测到手势。使用 `lv_indev_get_gesture_dir(lv_indev_get_act());` 获取手势
- LV_EVENT_KEY 一个密钥被发送到对象。使用 `lv_indev_get_key(lv_indev_get_act());` 获取密钥
- LV_EVENT_FOCUSED 对象被聚焦
- LV_EVENT_DEFOCUSED 对象散焦
- LV_EVENT_LEAVE 对象散焦但仍被选中

- LV_EVENT_HIT_TEST 执行高级命中测试。使用 lv_hit_test_info_t * a = lv_event_get_hit_test_info(e) 并检查 a->point 是否可以点击对象。如果没有则 a->res = false

Drawing events（绘图事件）

- LV_EVENT_COVER_CHECK Check if the object fully covers an area. The event parameter is lv_cover_check_info_t *.
- LV_EVENT_REFR_EXT_DRAW_SIZE Get the required extra draw area around the object (e.g. for shadow). The event parameter is lv_coord_t * to store the size. Overwrite it only with a larger value.
- LV_EVENT_DRAW_MAIN_BEGIN Starting the main drawing phase.
- LV_EVENT_DRAW_MAIN Perform the main drawing
- LV_EVENT_DRAW_MAIN_END Finishing the main drawing phase
- LV_EVENT_DRAW_POST_BEGIN Starting the post draw phase (when all children are drawn)
- LV_EVENT_DRAW_POST Perform the post draw phase (when all children are drawn)
- LV_EVENT_DRAW_POST_END Finishing the post draw phase (when all children are drawn)
- LV_EVENT_DRAW_PART_BEGIN Starting to draw a part. The event parameter is lv_obj_draw_dsc_t *. Learn more *here*.
- LV_EVENT_DRAW_PART_END Finishing to draw a part. The event parameter is lv_obj_draw_dsc_t *. Learn more *here*.
- LV_EVENT_COVER_CHECK 检查对象是否完全覆盖一个区域。事件参数是 lv_cover_check_info_t *。
- LV_EVENT_REFR_EXT_DRAW_SIZE 获取对象周围所需的额外绘制区域（例如用于阴影）。事件参数是 lv_coord_t * 来存储大小。仅用更大的值覆盖它。
- LV_EVENT_DRAW_MAIN_BEGIN 开始主绘图阶段。
- LV_EVENT_DRAW_MAIN 执行主绘图
- LV_EVENT_DRAW_MAIN_END 完成主绘制阶段
- LV_EVENT_DRAW_POST_BEGIN 开始后期绘制阶段（当所有孩子都被绘制时）
- LV_EVENT_DRAW_POST 执行后期绘制阶段（当所有孩子都被绘制时）
- LV_EVENT_DRAW_POST_END 完成后期绘制阶段（当所有孩子都被绘制时）
- LV_EVENT_DRAW_PART_BEGIN 开始绘制零件。事件参数是 lv_obj_draw_dsc_t *。了解更多此处。
- LV_EVENT_DRAW_PART_END 完成绘制零件。事件参数是 lv_obj_draw_dsc_t *。了解更多此处。

Other events（其他事件）

- `LV_EVENT_DELETE` Object is being deleted
- `LV_EVENT_CHILD_CHANGED` Child was removed/added
- `LV_EVENT_SIZE_CHANGED` Object coordinates/size have changed
- `LV_EVENT_STYLE_CHANGED` Object's style has changed
- `LV_EVENT_BASE_DIR_CHANGED` The base dir has changed
- `LV_EVENT_GET_SELF_SIZE` Get the internal size of a widget
- `LV_EVENT_DELETE` 对象正在被删除
- `LV_EVENT_CHILD_CHANGED` 孩子被移除/添加
- `LV_EVENT_SIZE_CHANGED` 对象坐标/大小已更改
- `LV_EVENT_STYLE_CHANGED` 对象的样式已更改
- `LV_EVENT_BASE_DIR_CHANGED` 基础目录已经改变
- `LV_EVENT_GET_SELF_SIZE` 获取小部件的内部尺寸

Special events（特殊事件）

- `LV_EVENT_VALUE_CHANGED` The object's value has changed (i.e. slider moved)
- `LV_EVENT_INSERT` A text is being inserted to the object. The event data is `char *` being inserted.
- `LV_EVENT_REFRESH` Notify the object to refresh something on it (for the user)
- `LV_EVENT_READY` A process has finished
- `LV_EVENT_CANCEL` A process has been canceled
- `LV_EVENT_VALUE_CHANGED` 对象的值已更改（即滑块移动）
- `LV_EVENT_INSERT` 正在向对象插入文本。事件数据是插入的 `char *`。
- `LV_EVENT_REFRESH` 通知对象刷新其上的某些内容（对于用户）
- `LV_EVENT_READY` 一个过程已经完成
- `LV_EVENT_CANCEL` 一个过程被取消

Custom events（自定义事件）

Any custom event codes can be registered by `uint32_t MY_EVENT_1 = lv_event_register_id();`

And can be sent to any object with `lv_event_send(obj, MY_EVENT_1, &some_data)`

任何自定义事件代码都可以通过 `uint32_t MY_EVENT_1 = lv_event_register_id();` 注册

并且可以使用 `lv_event_send(obj, MY_EVENT_1, &some_data)` 发送到任何对象

Sending events（发送事件）

To manually send events to an object, use `lv_event_send(obj, <EVENT_CODE> &some_data)`.

For example, this can be used to manually close a message box by simulating a button press (although there are simpler ways to do this):

要手动向对象发送事件，请使用 `lv_event_send(obj, <EVENT_CODE> &some_data)`。

例如，这可用于通过模拟按钮按下来手动关闭消息框（尽管有更简单的方法可以做到这一点）：

```
/*Simulate the press of the first button (indexes start from zero)*/
uint32_t btn_id = 0;
lv_event_send(mbox, LV_EVENT_VALUE_CHANGED, &btn_id);
```

Refresh event（刷新事件）

`LV_EVENT_REFRESH` is special event because it's designed to be used by the user to notify an object to refresh itself. Some examples:

- notify a label to refresh its text according to one or more variables (e.g. current time)
- refresh a label when the language changes
- enable a button if some conditions are met (e.g. the correct PIN is entered)
- add/remove styles to/from an object if a limit is exceeded, etc

`LV_EVENT_REFRESH` 是一个特殊事件，因为它被设计为用户使用它来通知对象刷新自身。一些例子：

- 通知标签根据一个或多个变量（例如当前时间）刷新其文本
- 当语言改变时刷新标签
- 如果满足某些条件（例如输入正确的 PIN），则启用按钮
- 如果超出限制，则向/从对象添加/删除样式等

Fields of lv_event_t（lv_event_t 的字段）

`lv_event_t` is the only parameter passed to event callback and it contains all the data about the event. The following values can be gotten from it:

- `lv_event_get_code(e)` get the event code
- `lv_event_get_target(e)` get the object to which the event is sent
- `lv_event_get_original_target(e)` get the object to which the event is sent originally sent (different from `lv_event_get_target` if *event bubbling* is enabled)
- `lv_event_get_user_data(e)` get the pointer passed as the last parameter of `lv_obj_add_event_cb`.
- `lv_event_get_param(e)` get the parameter passed as the last parameter of `lv_event_send`

`lv_event_t` 是传递给事件回调的唯一参数，它包含有关事件的所有数据。可以从中获得以下值：

- `lv_event_get_code(e)` 获取触发的事件代码
- `lv_event_get_target(e)` 获取事件发送到 (关联) 的对象
- `lv_event_get_original_target(e)` 获取事件最初发送到的对象（与 `lv_event_get_target` 不同，如果*event bubbling* 被启用）
- `lv_event_get_user_data(e)` 获取作为 `lv_obj_add_event_cb` 的最后一个参数传递的指针。
- `lv_event_get_param(e)` 获取作为 `lv_event_send` 的最后一个参数传递的参数

Event bubbling（事件冒泡）

If `lv_obj_add_flag(obj, LV_OBJ_FLAG_EVENT_BUBBLE)` is enabled all events will be sent to the object's parent too. If the parent also has `LV_OBJ_FLAG_EVENT_BUBBLE` enabled the event will be sent to its parent too, and so on.

The *target* parameter of the event is always the current target object, not the original object. To get the original target call `lv_event_get_original_target(e)` in the event handler.

如果启用了 `lv_obj_add_flag(obj, LV_OBJ_FLAG_EVENT_BUBBLE)`，所有事件也将发送到对象的父级。如果父级也启用了 `LV_OBJ_FLAG_EVENT_BUBBLE`，则事件也将发送到其父级，依此类推。

事件的 *target* 参数始终是当前目标对象，而不是原始对象。在事件处理程序中获取原始目标调用 `lv_event_get_original_target(e)`。

Examples

Button click event

```c
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_SWITCH

static void event_cb(lv_event_t * e)
{
    LV_LOG_USER("Clicked");

    static uint32_t cnt = 1;
    lv_obj_t * btn = lv_event_get_target(e);
    lv_obj_t * label = lv_obj_get_child(btn, 0);
    lv_label_set_text_fmt(label, "%"LV_PRIu32, cnt);
    cnt++;
}

/**
 * Add click event to a button
 */
void lv_example_event_1(void)
{
    lv_obj_t * btn = lv_btn_create(lv_scr_act());
    lv_obj_set_size(btn, 100, 50);
    lv_obj_center(btn);
    lv_obj_add_event_cb(btn, event_cb, LV_EVENT_CLICKED, NULL);

    lv_obj_t * label = lv_label_create(btn);
    lv_label_set_text(label, "Click me!");
    lv_obj_center(label);
}

#endif
```

```python
class Event_1():
    def __init__(self):
        self.cnt = 1
        #
        # Add click event to a button
        #

        btn = lv.btn(lv.scr_act())
        btn.set_size(100, 50)
```

(下页继续)

(续上页)

```
        btn.center()
        btn.add_event_cb(self.event_cb, lv.EVENT.CLICKED, None)

        label = lv.label(btn)
        label.set_text("Click me!")
        label.center()

    def event_cb(self,e):
        print("Clicked")

        btn = e.get_target()
        label = btn.get_child(0)
        label.set_text(str(self.cnt))
        self.cnt += 1

evt1 = Event_1()
```

Handle multiple events

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_SWITCH

static void event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * label = lv_event_get_user_data(e);

    switch(code) {
    case LV_EVENT_PRESSED:
        lv_label_set_text(label, "The last button event:\nLV_EVENT_PRESSED");
        break;
    case LV_EVENT_CLICKED:
        lv_label_set_text(label, "The last button event:\nLV_EVENT_CLICKED");
        break;
    case LV_EVENT_LONG_PRESSED:
        lv_label_set_text(label, "The last button event:\nLV_EVENT_LONG_PRESSED");
        break;
    case LV_EVENT_LONG_PRESSED_REPEAT:
        lv_label_set_text(label, "The last button event:\nLV_EVENT_LONG_PRESSED_REPEAT
↪");
        break;
    default:
```

(下页继续)

(续上页)

```
        break;
    }
}

/**
 * Handle multiple events
 */
void lv_example_event_2(void)
{
    lv_obj_t * btn = lv_btn_create(lv_scr_act());
    lv_obj_set_size(btn, 100, 50);
    lv_obj_center(btn);

    lv_obj_t * btn_label = lv_label_create(btn);
    lv_label_set_text(btn_label, "Click me!");
    lv_obj_center(btn_label);

    lv_obj_t * info_label = lv_label_create(lv_scr_act());
    lv_label_set_text(info_label, "The last button event:\nNone");

    lv_obj_add_event_cb(btn, event_cb, LV_EVENT_ALL, info_label);
}

#endif
```

```
def event_cb(e,label):
    code = e.get_code()
    if code == lv.EVENT.PRESSED:
        label.set_text("The last button event:\nLV_EVENT_PRESSED")
    elif code == lv.EVENT.CLICKED:
        label.set_text("The last button event:\nLV_EVENT_CLICKED")
    elif code ==  lv.EVENT.LONG_PRESSED:
        label.set_text("The last button event:\nLV_EVENT_LONG_PRESSED")
    elif code == lv.EVENT.LONG_PRESSED_REPEAT:
        label.set_text("The last button event:\nLV_EVENT_LONG_PRESSED_REPEAT")
btn = lv.btn(lv.scr_act())
btn.set_size(100, 50)
btn.center()

btn_label = lv.label(btn)
btn_label.set_text("Click me!")
btn_label.center()
```

(下页继续)

(续上页)

```
info_label = lv.label(lv.scr_act())
info_label.set_text("The last button event:\nNone")

btn.add_event_cb(lambda e: event_cb(e,info_label), lv.EVENT.ALL, None)
```

Event bubbling

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_FLEX

static void event_cb(lv_event_t * e)
{
    /*The original target of the event. Can be the buttons or the container*/
    lv_obj_t * target = lv_event_get_target(e);

    /*The current target is always the container as the event is added to it*/
    lv_obj_t * cont = lv_event_get_current_target(e);

    /*If container was clicked do nothing*/
    if(target == cont) return;

    /*Make the clicked buttons red*/
    lv_obj_set_style_bg_color(target, lv_palette_main(LV_PALETTE_RED), 0);
}

/**
 * Demonstrate event bubbling
 */
void lv_example_event_3(void)
{

    lv_obj_t * cont = lv_obj_create(lv_scr_act());
    lv_obj_set_size(cont, 290, 200);
    lv_obj_center(cont);
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_ROW_WRAP);

    uint32_t i;
    for(i = 0; i < 30; i++) {
        lv_obj_t * btn = lv_btn_create(cont);
        lv_obj_set_size(btn, 80, 50);
        lv_obj_add_flag(btn, LV_OBJ_FLAG_EVENT_BUBBLE);
```

(下页继续)

(续上页)

```
        lv_obj_t * label = lv_label_create(btn);
        lv_label_set_text_fmt(label, "%"LV_PRIu32, i);
        lv_obj_center(label);
    }

    lv_obj_add_event_cb(cont, event_cb, LV_EVENT_CLICKED, NULL);
}

#endif
```

```
def event_cb(e):

    # The original target of the event. Can be the buttons or the container
    target = e.get_target()
    # print(type(target))

    # If container was clicked do nothing
    if type(target) != type(lv.btn()):
        return

    # Make the clicked buttons red
    target.set_style_bg_color(lv.palette_main(lv.PALETTE.RED), 0)

#
# Demonstrate event bubbling
#

cont = lv.obj(lv.scr_act())
cont.set_size(320, 200)
cont.center()
cont.set_flex_flow(lv.FLEX_FLOW.ROW_WRAP)

for i in range(30):
    btn = lv.btn(cont)
    btn.set_size(80, 50)
    btn.add_flag(lv.obj.FLAG.EVENT_BUBBLE)

    label = lv.label(btn)
    label.set_text(str(i))
    label.center()
    cont.add_event_cb(event_cb, lv.EVENT.CLICKED, None)
```

2.5.8 Input devices（输入设备）

An input device usually means:

- Pointer-like input device like touchpad or mouse
- Keypads like a normal keyboard or simple numeric keypad
- Encoders with left/right turn and push options
- External hardware buttons which are assigned to specific points on the screen

一般来说输入设备可以是：

- 类似指针的输入设备，比如：触摸板或鼠标
- 像普通键盘或简单数字键盘那样的键盘
- 带左/右转和推动选项的编码器
- 分配给屏幕外围特定点的外部硬件按钮

重要： Before reading further, please read the [Porting](/porting/indev) section of Input devices

---

重要： 在进一步阅读之前，请阅读输入设备 (Input devices) 的 [Porting](/porting/indev) 部分

Pointers（光标）

Pointer input devices (like a mouse) can have a cursor.

有些输入设备可以有一个光标（如鼠标）。

```
...
lv_indev_t * mouse_indev = lv_indev_drv_register(&indev_drv);

LV_IMG_DECLARE(mouse_cursor_icon);                          /*Declare the image file.
↪*/
lv_obj_t * cursor_obj =  lv_img_create(lv_scr_act(), NULL); /*Create an image object␣
↪for the cursor */
lv_img_set_src(cursor_obj, &mouse_cursor_icon);             /*Set the image source*/
lv_indev_set_cursor(mouse_indev, cursor_obj);               /*Connect the image ␣
↪object to the driver*/
```

Note that the cursor object should have `lv_obj_set_click(cursor_obj, false)`. For images, *clicking* is disabled by default.

请注意，光标对象应该有 `lv_obj_set_click(cursor_obj, false)`。

对于图像，默认情况下禁用 单击。

Gestures (手势)

Pointer input devives can detect basic gestures. By default, most of the widgets send the gestures to its parent, so finally the gestures can be detected on the screen object in a form of an LV_EVENT_GESTURE event. For example:

指针输入设备可以检测基本手势。默认情况下，大多数部件 (对象) 会将手势发送给其父级 (事件冒泡)，因此我们最终可以在屏幕对象 (lv_scr_act()) 上设置回调函数，在 LV_EVENT_GESTURE 的事件类型检测处理手势事件。例如：

```
void my_event(lv_event_t * e)
{
  lv_obj_t * screen = lv_event_get_current_target(e);
  lv_dir_t dir = lv_indev_get_gesture_dir(lv_indev_act());
  switch(dir) {
    case LV_DIR_LEFT:
      ...
      break;
    case LV_DIR_RIGHT:
      ...
      break;
    case LV_DIR_TOP:
      ...
      break;
    case LV_DIR_BOTTOM:
      ...
      break;
  }
}

...

lv_obj_add_event_cb(screen1, my_event, LV_EVENT_GESTURE, NULL);
```

To prevent passing the gesture event to the parent from an obejct use `lv_obj_clear_flag(obj, LV_OBJ_FLAG_GESTURE_BUBBLE)`.

Note that, gestures are not triggered if an object is being scrolled.

如果不想让将手势事件从对象传递给父对象，请使用 `lv_obj_clear_flag(obj, LV_OBJ_FLAG_GESTURE_BUBBLE);` 禁用 事件冒泡。

请注意，如果对象正在滚动，则不会触发手势。

Keypad and encoder（键盘和编码器）

You can fully control the user interface without touchpad or mouse using a keypad or encoder(s). It works similar to the *TAB* key on the PC to select the element in an application or a web page.

您可以使用键盘或编码器在没有触摸板或鼠标的情况下完全控制用户界面。它的工作原理类似于 PC 上的 *TAB* 键，用于选择应用程序或网页中的元素。

Groups（组）

The objects, you want to control with keypad or encoder, needs to be added to a *Group*. In every group, there is exactly one focused object which receives the pressed keys or the encoder actions. For example, if a *Text area* is focused and you press some letter on a keyboard, the keys will be sent and inserted into the text area. Similarly, if a *Slider* is focused and you press the left or right arrows, the slider's value will be changed.

You need to associate an input device with a group. An input device can send the keys to only one group but, a group can receive data from more than one input device too.

To create a group use `lv_group_t * g = lv_group_create()` and to add an object to the group use `lv_group_add_obj(g, obj)`.

To associate a group with an input device use `lv_indev_set_group(indev, g)`, where `indev` is the return value of `lv_indev_drv_register()`

您想用键盘或编码器控制的对象需要将其添加到*组 (Groups)*，一般都会被添加进默认的 *组 (Groups)*。

在每一组中，同时只有一个焦点对象接收按下的键或编码器的动作。

例如，如果*文本区域* 被聚焦并且您在键盘上按下某个字母，则按键将被发送并插入到文本区域中。

类似地，如果*Slider* 获得焦点并且您按下向左或向右箭头，则滑块的值将被更改。

将对象添加到 *组 (Groups)* 还不够，我们还需要将输入设备与组关联。一个输入设备只能将按键发送给一组，但一组也可以从多个输入设备接收数据。

- 要创建一个 *组 (Groups)* 使用 `lv_group_t * g = lv_group_create();`
- 将一个对象添加到 *组 (Groups)* 中使用 `lv_group_add_obj(g, obj);`。
- 要将 *组 (Groups)* 与输入设备相关联，请使用 `lv_indev_set_group(indev, g)`，其中 `indev` 是 `lv_indev_drv_register()` 的返回值

Keys（按键）

There are some predefined keys which have special meaning:

- **LV_KEY_NEXT** Focus on the next object
- **LV_KEY_PREV** Focus on the previous object
- **LV_KEY_ENTER** Triggers `LV_EVENT_PRESSED/CLICKED/LONG_PRESSED` etc. events
- **LV_KEY_UP** Increase value or move upwards
- **LV_KEY_DOWN** Decrease value or move downwards
- **LV_KEY_RIGHT** Increase value or move the the right
- **LV_KEY_LEFT** Decrease value or move the the left
- **LV_KEY_ESC** Close or exit (E.g. close a *Drop down list*)
- **LV_KEY_DEL** Delete (E.g. a character on the right in a *Text area*)
- **LV_KEY_BACKSPACE** Delete a character on the left (E.g. in a *Text area*)
- **LV_KEY_HOME** Go to the beginning/top (E.g. in a *Text area*)
- **LV_KEY_END** Go to the end (E.g. in a *Text area*))

有一些具有特殊含义的预定义键：

- **LV_KEY_NEXT** 聚焦到下一个对象
- **LV_KEY_PREV** 聚焦到上一个对象
- **LV_KEY_ENTER** 触发 `LV_EVENT_PRESSED/CLICKED/LONG_PRESSED` 等事件
- **LV_KEY_UP** 增加值或向上移动
- **LV_KEY_DOWN** 减少值或向下移动
- **LV_KEY_RIGHT** 增加值或向右移动
- **LV_KEY_LEFT** 减少值或向左移动
- **LV_KEY_ESC** 关闭或退出（例如关闭下拉列表）
- **LV_KEY_DEL** 删除（例如文本区域 中右侧的字符）
- **LV_KEY_BACKSPACE** 删除左边的一个字符（例如在文本区域）
- **LV_KEY_HOME** 跳到开头/顶部（例如在文本区域）
- **LV_KEY_END** 跳到最后（例如在文本区域））

The most important special keys are `LV_KEY_NEXT/PREV`, `LV_KEY_ENTER` and `LV_KEY_UP/DOWN/LEFT/RIGHT`. In your `read_cb` function, you should translate some of your keys to these special keys to navigate in the group and interact with the selected object.

Usually, it's enough to use only LV_KEY_LEFT/RIGHT because most of the objects can be fully controlled with them.

With an encoder, you should use only LV_KEY_LEFT, LV_KEY_RIGHT, and LV_KEY_ENTER.

最重要的特殊键是：

- LV_KEY_NEXT/PREV
- LV_KEY_ENTER
- LV_KEY_UP/DOWN/LEFT/RIGHT

在您的 read_cb 函数中，应该优先考虑将一些键转换对应为这些特殊键，以便在组中导航并与所选对象进行交互。

通常，只使用 LV_KEY_LEFT/RIGHT 就足够了，因为大多数对象都可以用它们完全控制。

对于编码器，您应该只使用 LV_KEY_LEFT、LV_KEY_RIGHT 和 LV_KEY_ENTER。

Edit and navigate mode（编辑和导航模式）

Since a keypad has plenty of keys, it's easy to navigate between the objects and edit them using the keypad. But the encoders have a limited number of "keys" and hence it is difficult to navigate using the default options. *Navigate* and *Edit* are created to avoid this problem with the encoders.

In *Navigate* mode, the encoders LV_KEY_LEFT/RIGHT is translated to LV_KEY_NEXT/PREV. Therefore the next or previous object will be selected by turning the encoder. Pressing LV_KEY_ENTER will change to *Edit* mode.

In *Edit* mode, LV_KEY_NEXT/PREV is usually used to edit the object. Depending on the object's type, a short or long press of LV_KEY_ENTER changes back to *Navigate* mode. Usually, an object which can not be pressed (like a *Slider*) leaves *Edit* mode on short click. But with objects where short click has meaning (e.g. *Button*), a long press is required.

由于小键盘有很多键，因此很容易在对象之间导航并使用小键盘对其进行编辑。但是编码器的“键”数量有限，因此很难使用默认选项进行导航。创建 *Navigate* 和 *Edit* 是为了避免编码器出现此问题。

在 *Navigate* 模式下，编码器 LV_KEY_LEFT/RIGHT 被转换为 LV_KEY_NEXT/PREV。因此，将通过转动编码器来选择下一个或上一个对象。

按 LV_KEY_ENTER 将更改为 *Edit* 模式。在 *Edit* 模式下，LV_KEY_NEXT/PREV 通常用于编辑对象。根据对象的类型，短按或长按 LV_KEY_ENTER 会变回 *Navigate* 模式。通常，无法按下的对象（如*Slider*）会在短按时离开 *Edit* 模式。但是对于短按有意义的对象（例如*Button*），则需要长按。

Default group（默认组）

Interactive widgets - such as buttons, checkboxes, sliders, etc - can be automatically added to a default group. Just create a group with `lv_group_t * g = lv_group_create();` and set the default group with `lv_group_set_default(g);`

Don't forget to assign the input device(s) to the default group with `lv_indev_set_group(my_indev, g);`.

交互式小部件 - 例如按钮、复选框、滑块等 - 可以自动添加到默认组。只需使用 `lv_group_t * g = lv_group_create();` 创建一个组并使用 `lv_group_set_default(g);` 设置默认组

不要忘记使用 `lv_indev_set_group(my_indev, g);` 将输入设备分配到默认组。

Styling（风格样式）

If an object is focused either by clicking it via touchpad, or focused via an encoder or keypad it goes to `LV_STATE_FOCUSED`. Hence focused styles will be applied on it.

If the object goes to edit mode it goes to `LV_STATE_FOCUSED | LV_STATE_EDITED` state so these style properties will be shown.

For a more detaild description read the Style section.

如果通过触摸板单击对象或通过编码器或键盘聚焦对象，它会转到 "LV_STATE_FOCUSED"。因此，将在其上应用重点样式。

如果对象进入编辑模式，它将进入 `LV_STATE_FOCUSED | LV_STATE_EDITED` 状态，因此将显示这些样式属性。

有关更详细的说明，请阅读 样式 部分。

API

Input device（输入设备）

Functions

void **lv_indev_read_timer_cb**(*lv_timer_t* *timer)

Called periodically to read the input devices

参数 **timer** -- pointer to a timer to read

void **lv_indev_enable**(*lv_indev_t* *indev, bool en)

lv_indev_t ***lv_indev_get_act**(void)

Get the currently processed input device. Can be used in action functions too.

返回 pointer to the currently processed input device or NULL if no input device processing right now

lv_indev_type_t **lv_indev_get_type**(const *lv_indev_t* *indev)

Get the type of an input device

参数 **indev** -- pointer to an input device

返回 the type of the input device from `lv_hal_indev_type_t` (`LV_INDEV_TYPE_...`)

void **lv_indev_reset**(*lv_indev_t* *indev, *lv_obj_t* *obj)

Reset one or all input devices

参数

- **indev** -- pointer to an input device to reset or NULL to reset all of them
- **obj** -- pointer to an object which triggers the reset.

void **lv_indev_reset_long_press**(*lv_indev_t* *indev)

Reset the long press state of an input device

参数 **indev** -- pointer to an input device

void **lv_indev_set_cursor**(*lv_indev_t* *indev, *lv_obj_t* *cur_obj)

Set a cursor for a pointer input device (for LV_INPUT_TYPE_POINTER and LV_INPUT_TYPE_BUTTON)

参数

- **indev** -- pointer to an input device
- **cur_obj** -- pointer to an object to be used as cursor

void **lv_indev_set_group**(*lv_indev_t* *indev, *lv_group_t* *group)

Set a destination group for a keypad input device (for LV_INDEV_TYPE_KEYPAD)

参数

- **indev** -- pointer to an input device
- **group** -- point to a group

void **lv_indev_set_button_points**(*lv_indev_t* *indev, const lv_point_t points[])

Set the an array of points for LV_INDEV_TYPE_BUTTON. These points will be assigned to the buttons to press a specific point on the screen

参数

- **indev** -- pointer to an input device
- **group** -- point to a group

void **lv_indev_get_point**(const *lv_indev_t* *indev, lv_point_t *point)

Get the last point of an input device (for LV_INDEV_TYPE_POINTER and LV_INDEV_TYPE_BUTTON)

参数

- **indev** -- pointer to an input device

- **point** -- pointer to a point to store the result

lv_dir_t **lv_indev_get_gesture_dir**(const *lv_indev_t* *indev)

Get the current gesture direct

参数 **indev** -- pointer to an input device

返回 current gesture direct

uint32_t **lv_indev_get_key**(const *lv_indev_t* *indev)

Get the last pressed key of an input device (for LV_INDEV_TYPE_KEYPAD)

参数 **indev** -- pointer to an input device

返回 the last pressed key (0 on error)

lv_dir_t **lv_indev_get_scroll_dir**(const *lv_indev_t* *indev)

Check the current scroll direction of an input device (for LV_INDEV_TYPE_POINTER and LV_INDEV_TYPE_BUTTON)

参数 **indev** -- pointer to an input device

返回 LV_DIR_NONE: no scrolling now LV_DIR_HOR/VER

lv_obj_t ***lv_indev_get_scroll_obj**(const *lv_indev_t* *indev)

Get the currently scrolled object (for LV_INDEV_TYPE_POINTER and LV_INDEV_TYPE_BUTTON)

参数 **indev** -- pointer to an input device

返回 pointer to the currently scrolled object or NULL if no scrolling by this indev

void **lv_indev_get_vect**(const *lv_indev_t* *indev, lv_point_t *point)

Get the movement vector of an input device (for LV_INDEV_TYPE_POINTER and LV_INDEV_TYPE_BUTTON)

参数

- **indev** -- pointer to an input device
- **point** -- pointer to a point to store the types.pointer.vector

void **lv_indev_wait_release**(*lv_indev_t* *indev)

Do nothing until the next release

参数 **indev** -- pointer to an input device

lv_obj_t ***lv_indev_get_obj_act**(void)

Gets a pointer to the currently active object in the currently processed input device.

返回 pointer to currently active object or NULL if no active object

lv_timer_t ***lv_indev_get_read_timer**(*lv_disp_t* *indev)

Get a pointer to the indev read timer to modify its parameters with `lv_timer_...` functions.

参数 **indev** -- pointer to an input device

返回 pointer to the indev read refresher timer. (NULL on error)

lv_obj_t ***lv_indev_search_obj**(*lv_obj_t* *obj, lv_point_t *point)

Search the most top, clickable object by a point

参数

- **obj** -- pointer to a start object, typically the screen
- **point** -- pointer to a point for searching the most top child

返回 pointer to the found object or NULL if there was no suitable object

Groups（组）

Typedefs

typedef uint8_t **lv_key_t**

typedef void (***lv_group_focus_cb_t**)(struct *_lv_group_t**)

typedef struct *_lv_group_t* **lv_group_t**

Groups can be used to logically hold objects so that they can be individually focused. They are NOT for laying out objects on a screen (try layouts for that).

Enums

enum **[anonymous]**

Values:

enumerator **LV_KEY_UP**

enumerator **LV_KEY_DOWN**

enumerator **LV_KEY_RIGHT**

enumerator **LV_KEY_LEFT**

enumerator **LV_KEY_ESC**

enumerator **LV_KEY_DEL**

enumerator **LV_KEY_BACKSPACE**

enumerator **LV_KEY_ENTER**

enumerator **LV_KEY_NEXT**

enumerator **LV_KEY_PREV**

enumerator **LV_KEY_HOME**

enumerator **LV_KEY_END**

enum **lv_group_refocus_policy_t**

Values:

enumerator **LV_GROUP_REFOCUS_POLICY_NEXT**

enumerator **LV_GROUP_REFOCUS_POLICY_PREV**

Functions

void **_lv_group_init**(void)

Init. the group module

Remark Internal function, do not call directly.

lv_group_t ***lv_group_create**(void)

Create a new object group

返回 pointer to the new object group

void **lv_group_del**(*lv_group_t* *group)

Delete a group object

参数 **group** -- pointer to a group

void **lv_group_set_default**(*lv_group_t* *group)

Set a default group. New object are added to this group if it's enabled in their class with `add_to_def_group = true`

参数 **group** -- pointer to a group (can be `NULL`)

lv_group_t ***lv_group_get_default**(void)

Get the default group

返回 pointer to the default group

void **lv_group_add_obj**(*lv_group_t* *group, struct *_lv_obj_t* *obj)

Add an object to a group

参数

- **group** -- pointer to a group
- **obj** -- pointer to an object to add

void **lv_group_swap_obj**(struct *_lv_obj_t* *obj1, struct *_lv_obj_t* *obj2)

Swap 2 object in a group. The object must be in the same group

参数

- **obj1** -- pointer to an object

- **obj2** -- pointer to an other object

void **lv_group_remove_obj**(struct *_lv_obj_t* *obj)

Remove an object from its group

参数 **obj** -- pointer to an object to remove

void **lv_group_remove_all_objs**(*lv_group_t* *group)

Remove all objects from a group

参数 **group** -- pointer to a group

void **lv_group_focus_obj**(struct *_lv_obj_t* *obj)

Focus on an object (defocus the current)

参数 **obj** -- pointer to an object to focus on

void **lv_group_focus_next**(*lv_group_t* *group)

Focus the next object in a group (defocus the current)

参数 **group** -- pointer to a group

void **lv_group_focus_prev**(*lv_group_t* *group)

Focus the previous object in a group (defocus the current)

参数 **group** -- pointer to a group

void **lv_group_focus_freeze**(*lv_group_t* *group, bool en)

Do not let to change the focus from the current object

参数

- **group** -- pointer to a group
- **en** -- true: freeze, false: release freezing (normal mode)

lv_res_t **lv_group_send_data**(*lv_group_t* *group, uint32_t c)

Send a control character to the focuses object of a group

参数

- **group** -- pointer to a group
- **c** -- a character (use LV_KEY_.. to navigate)

返回 result of focused object in group.

void **lv_group_set_focus_cb**(*lv_group_t* *group, *lv_group_focus_cb_t* focus_cb)

Set a function for a group which will be called when a new object is focused

参数

- **group** -- pointer to a group
- **focus_cb** -- the call back function or NULL if unused

void **lv_group_set_refocus_policy**(*lv_group_t* *group, *lv_group_refocus_policy_t* policy)

Set whether the next or previous item in a group is focused if the currently focused obj is deleted.

参数

- **group** -- pointer to a group
- **policy** -- new refocus policy enum

void **lv_group_set_editing**(*lv_group_t* *group, bool edit)

Manually set the current mode (edit or navigate).

参数

- **group** -- pointer to group
- **edit** -- true: edit mode; false: navigate mode

void **lv_group_set_wrap**(*lv_group_t* *group, bool en)

Set whether focus next/prev will allow wrapping from first->last or last->first object.

参数

- **group** -- pointer to group
- **en** -- true: wrapping enabled; false: wrapping disabled

struct *_lv_obj_t* ***lv_group_get_focused**(const *lv_group_t* *group)

Get the focused object or NULL if there isn't one

参数 **group** -- pointer to a group

返回 pointer to the focused object

lv_group_focus_cb_t **lv_group_get_focus_cb**(const *lv_group_t* *group)

Get the focus callback function of a group

参数 **group** -- pointer to a group

返回 the call back function or NULL if not set

bool **lv_group_get_editing**(const *lv_group_t* *group)

Get the current mode (edit or navigate).

参数 **group** -- pointer to group

返回 true: edit mode; false: navigate mode

bool **lv_group_get_wrap**(*lv_group_t* *group)

Get whether focus next/prev will allow wrapping from first->last or last->first object.

参数

- **group** -- pointer to group
- **en** -- true: wrapping enabled; false: wrapping disabled

uint32_t **lv_group_get_obj_count**(*lv_group_t* *group)

Get the number of object in the group

参数 **group** -- pointer to a group

返回 number of objects in the group

struct **_lv_group_t**

#include <lv_group.h> Groups can be used to logically hold objects so that they can be individually focused. They are NOT for laying out objects on a screen (try layouts for that).

Public Members

lv_ll_t **obj_ll**

Linked list to store the objects in the group

struct *_lv_obj_t* ****obj_focus**

The object in focus

lv_group_focus_cb_t **focus_cb**

A function to call when a new object is focused (optional)

void ***user_data**

uint8_t **frozen**

1: can't focus to new object

uint8_t **editing**

1: Edit mode, 0: Navigate mode

uint8_t **refocus_policy**

1: Focus prev if focused on deletion. 0: Focus next if focused on deletion.

uint8_t **wrap**

1: Focus next/prev can wrap at end of list. 0: Focus next/prev stops at end of list.

2.5.9 Displays（显示）

重要： The basic concept of *display* in LVGL is explained in the [Porting](/porting/display) section. So before reading further, please read the [Porting](/porting/display) section first.

重要： LVGL 中 *display* 的基本概念在 [Porting](/porting/display) 部分进行了解释。因此，在进一步阅读之前，请先阅读 [移植](/porting/display) 部分。

Multiple display support（多显示器支持）

In LVGL, you can have multiple displays, each with their own driver and objects. The only limitation is that every display needs to be have same color depth (as defined in `LV_COLOR_DEPTH`). If the displays are different in this regard the rendered image can be converted to the correct format in the drivers `flush_cb`.

Creating more displays is easy: just initialize more display buffers and register another driver for every display. When you create the UI, use `lv_disp_set_default(disp)` to tell the library on which display to create objects.

在 LVGL 中，您可以拥有多个显示器，每个显示器都有自己的驱动程序和对象。唯一的限制是每个显示器都需要具有相同的颜色深度（如 `LV_COLOR_DEPTH` 中所定义）。如果在这方面显示不同，渲染图像可以在驱动程序“flush_cb”中转换为正确的格式。

创建更多显示很容易：只需初始化更多显示缓冲区并为每个显示注册另一个驱动程序。创建 UI 时，使用 `lv_disp_set_default(disp)` 告诉库在哪个显示器上创建对象。

Why would you want multi-display support? Here are some examples:

- Have a ”normal” TFT display with local UI and create ”virtual” screens on VNC on demand. (You need to add your VNC driver).
- Have a large TFT display and a small monochrome display.
- Have some smaller and simple displays in a large instrument or technology.
- Have two large TFT displays: one for a customer and one for the shop assistant.

为什么需要多显示器支持？这里有些例子：

- 拥有带有本地 UI 的“正常”TFT 显示，并根据需要在 VNC 上创建“虚拟”屏幕。（您需要添加您的 VNC 驱动程序）。
- 拥有大型 TFT 显示屏和小型单色显示屏。
- 在大型仪器或技术中使用一些更小更简单的显示器。
- 有两个大的 TFT 显示器：一个给顾客用，一个给店员用。

Using only one display（仅使用一个显示器）

Using more displays can be useful but in most cases it's not required. Therefore, the whole concept of multi-display is completely hidden if you register only one display. By default, the lastly created (and only) display is used.

`lv_scr_act()`, `lv_scr_load(scr)`, `lv_layer_top()`, `lv_layer_sys()`, `LV_HOR_RES` and `LV_VER_RES` are always applied on the most recently created (default) screen. If you pass `NULL` as `disp` parameter to display related function, usually the default display will be used. E.g. `lv_disp_trig_activity(NULL)` will trigger a user activity on the default screen. (See below in *Inactivity*).

使用更多显示器可能很有用，但在大多数情况下不是必需的。因此，如果您只注册一个显示器，则完全隐藏多显示器的整个概念。默认情况下，使用最后创建的（也是唯一的）显示。

`lv_scr_act()`、`lv_scr_load(scr)`、`lv_layer_top()`、`lv_layer_sys()`、`LV_HOR_RES` 和 `LV_VER_RES` 始终应用于最近创建的（默认）屏幕。如果你将 `NULL` 作为 `disp` 参数传递给显示相关函数，通常会使用默认显示。例如。`lv_disp_trig_activity(NULL)` 将在默认屏幕上触发用户活动。（请参阅下面的 *Inactivity*）。

Mirror display（镜像显示）

To mirror the image of the display to another display, you don't need to use the multi-display support. Just transfer the buffer received in `drv.flush_cb` to the other display too.

要将显示器的图像镜像到另一个显示器，您不需要使用多显示器支持。只需将在 `drv.flush_cb` 中接收到的缓冲区也传输到另一个显示器。

Split image（分割图像）

You can create a larger display from smaller ones. You can create it as below:

1. Set the resolution of the displays to the large display's resolution.
2. In `drv.flush_cb`, truncate and modify the `area` parameter for each display.
3. Send the buffer's content to each display with the truncated area.

您可以从较小的显示器创建更大的显示器。您可以按如下方式创建它：

1. 将显示器的分辨率设置为大显示器的分辨率。
2. 在 `drv.flush_cb` 中，对每个显示器截断并修改 `area` 参数。
3. 将缓冲区的内容发送到每个截断区域的显示器。

Screens（屏幕）

Every display has each set of Screens and the object on the screens.

Be sure not to confuse displays and screens:

- **Displays** are the physical hardware drawing the pixels.
- **Screens** are the high-level root objects associated with a particular display. One display can have multiple screens associated with it, but not vice versa.

Screens can be considered the highest level containers which have no parent. The screen's size is always equal to its display and size their position is (0;0). Therefore, the screens coordinates can't be changed, i.e. `lv_obj_set_pos()`, `lv_obj_set_size()` or similar functions can't be used on screens.

每个显示器都有每组 屏幕 和屏幕上的对象。

确保不要混淆显示和屏幕：

- **显示器**是绘制像素的物理硬件。
- **屏幕**是与特定显示器关联的高级根对象。一个显示器可以有多个与其关联的屏幕，但反之则不然。

屏幕可以被认为是没有父级的最高级别的容器。屏幕的大小总是等于它的显示和大小，它们的位置是 (0;0)。因此，屏幕坐标不能改变，即 `lv_obj_set_pos()`、`lv_obj_set_size()` 或类似函数不能在屏幕上使用。

A screen can be created from any object type but the two most typical types are the *Base object* and the *Image* (to create a wallpaper).

To create a screen, use `lv_obj_t * scr = lv_<type>_create(NULL, copy)`. `copy` can be an other screen to copy it.

To load a screen, use `lv_scr_load(scr)`. To get the active screen, use `lv_scr_act()`. These functions works on the default display. If you want to to specify which display to work on, use `lv_disp_get_scr_act(disp)` and `lv_disp_load_scr(disp, scr)`. Screen can be loaded with animations too. Read more here.

Screens can be deleted with `lv_obj_del(scr)`, but ensure that you do not delete the currently loaded screen.

可以从任何对象类型创建屏幕，但最典型的两种类型是*Base object* 和*Image*（用于创建壁纸）。

要创建屏幕，请使用 `lv_obj_t * scr = lv_<type>_create(NULL, copy)`。`copy` 可以是另一个屏幕来复制它。

要加载屏幕，请使用 `lv_scr_load(scr)`。要获取活动屏幕，请使用 `lv_scr_act()`。这些功能适用于默认显示。如果你想指定在哪个显示器上工作，使用 `lv_disp_get_scr_act(disp)` 和 `lv_disp_load_scr(disp, scr)`。屏幕也可以加载动画。阅读更多 此处。

可以使用 `lv_obj_del(scr)` 删除屏幕，但请确保不要删除当前加载的屏幕。

Transparent screens（透明屏幕）

Usually, the opacity of the screen is `LV_OPA_COVER` to provide a solid background for its children. If it's not the case (opacity < 100%) the display's background color or image will be visible. See the *Display background* section for more details. If the display's background opacity is also not `LV_OPA_COVER` LVGL has no solid background to draw.

This configuration (transparent screen and display) could be used to create for example OSD menus where a video is played on a lower layer, and a menu is overlayed on an upper layer.

通常，屏幕的不透明度为“LV_OPA_COVER”，为其子项提供纯色背景。如果不是这种情况（不透明度 < 100%），显示器的背景颜色或图像将可见。有关更多详细信息，请参阅显示背景 部分。如果显示器的背景不透明度也不是 `LV_OPA_COVER`，则 LVGL 没有可绘制的纯色背景。

这种配置（透明屏幕和显示）可用于创建例如 OSD 菜单，其中视频在下层播放，菜单覆盖在上层。

To handle transparent displays special (slower) color mixing algorithms need to be used by LVGL so this feature needs to enabled with `LV_COLOR_SCREEN_TRANSP` in `lv_conf.h`. As this mode operates on the Alpha channel of the pixels `LV_COLOR_DEPTH = 32` is also required. The Alpha channel of 32-bit colors will be 0 where there are no objects and 255 where there are solid objects.

In summary, to enable transparent screen and displays to create OSD menu-like UIs:

- Enable `LV_COLOR_SCREEN_TRANSP` in `lv_conf.h`
- Be sure to use `LV_COLOR_DEPTH 32`
- Set the screens opacity to `LV_OPA_TRANSP` e.g. with `lv_obj_set_style_local_bg_opa(lv_scr_act(), LV_OBJMASK_PART_MAIN, LV_STATE_DEFAULT, LV_OPA_TRANSP)`
- Set the display opacity to `LV_OPA_TRANSP` with `lv_disp_set_bg_opa(NULL, LV_OPA_TRANSP);`

为了处理透明显示，LVGL 需要使用特殊的（较慢的）颜色混合算法，因此需要通过 `lv_conf.h` 中的 `LV_COLOR_SCREEN_TRANSP` 启用此功能。由于此模式在像素的 Alpha 通道上运行，因此还需要“LV_COLOR_DEPTH = 32”。32 位颜色的 Alpha 通道在没有对象时为 0，在有实体对象时为 255。

总之，要启用透明屏幕和显示以创建类似 OSD 菜单的 UI：

- 在 `lv_conf.h` 中启用 `LV_COLOR_SCREEN_TRANSP`
- 请务必使用 `LV_COLOR_DEPTH 32`
- 将屏幕不透明度设置为“LV_OPA_TRANSP”，例如使用 `lv_obj_set_style_local_bg_opa(lv_scr_act(), LV_OBJMASK_PART_MAIN, LV_STATE_DEFAULT, LV_OPA_TRANSP)`
- 使用 `lv_disp_set_bg_opa(NULL, LV_OPA_TRANSP);` 将显示不透明度设置为 `LV_OPA_TRANSP`

Features of displays（显示器的特点）

Inactivity（不活动）

The user's inactivity is measured on each display. Every use of an *Input device* (if associated with the display) counts as an activity. To get time elapsed since the last activity, use `lv_disp_get_inactive_time(disp)`. If `NULL` is passed, the overall smallest inactivity time will be returned from all displays (**not the default display**).

You can manually trigger an activity using `lv_disp_trig_activity(disp)`. If `disp` is `NULL`, the default screen will be used (**and not all displays**).

在每个显示器上测量用户的不活动。每次使用输入设备（如果 与显示器相关联 ）都算作一次活动。要获取自上次活动以来经过的时间，请使用 `lv_disp_get_inactive_time(disp)` 。如果传递 `NULL` ，则将从所有显示（**不是默认显示**）返回总体最小不活动时间。

您可以使用 `lv_disp_trig_activity(disp)` 手动触发活动。如果 `disp` 为 `NULL` ，将使用默认屏幕（** 并非所有显示 **）。

Background（背景）

Every display has background color, a background image and background opacity properties. They become visible when the current screen is transparent or not positioned to cover the whole display.

Background color is a simple color to fill the display. It can be adjusted with `lv_disp_set_bg_color(disp, color)`;

Background image is a path to a file or a pointer to an `lv_img_dsc_t` variable (converted image) to be used as wallpaper. It can be set with `lv_disp_set_bg_color(disp, &my_img)`; If the background image is set (not `NULL`) the background won't be filled with `bg_color`.

The opacity of the background color or image can be adjusted with `lv_disp_set_bg_opa(disp, opa)`.

The `disp` parameter of these functions can be `NULL` to refer it to the default display.

每个显示器都有背景颜色、背景图像和背景不透明度属性。当当前屏幕透明或未定位以覆盖整个显示器时，它们变得可见。

背景色是一种简单的颜色来填充显示。可以通过 `lv_disp_set_bg_color(disp, color)` 进行调整；

背景图像是文件的路径或指向用作墙纸的“lv_img_dsc_t”变量（转换后的图像）的指针。可以用 `lv_disp_set_bg_color(disp, &my_img)` 设置；如果设置了背景图像（不是“NULL”），背景将不会被“bg_color”填充。

背景颜色或图像的不透明度可以通过 `lv_disp_set_bg_opa(disp, opa)` 进行调整。

这些函数的 `disp` 参数可以是 `NULL` 以引用默认显示。

API

Enums

enum **lv_scr_load_anim_t**

Values:

enumerator **LV_SCR_LOAD_ANIM_NONE**

enumerator **LV_SCR_LOAD_ANIM_OVER_LEFT**

enumerator **LV_SCR_LOAD_ANIM_OVER_RIGHT**

enumerator **LV_SCR_LOAD_ANIM_OVER_TOP**

enumerator **LV_SCR_LOAD_ANIM_OVER_BOTTOM**

enumerator **LV_SCR_LOAD_ANIM_MOVE_LEFT**

enumerator **LV_SCR_LOAD_ANIM_MOVE_RIGHT**

enumerator **LV_SCR_LOAD_ANIM_MOVE_TOP**

enumerator **LV_SCR_LOAD_ANIM_MOVE_BOTTOM**

enumerator **LV_SCR_LOAD_ANIM_FADE_ON**

Functions

lv_obj_t ***lv_disp_get_scr_act**(*lv_disp_t* *disp)

Return with a pointer to the active screen

参数 **disp** -- pointer to display which active screen should be get. (NULL to use the default screen)

返回 pointer to the active screen object (loaded by 'lv_scr_load()')

lv_obj_t ***lv_disp_get_scr_prev**(*lv_disp_t* *disp)

Return with a pointer to the previous screen. Only used during screen transitions.

参数 **disp** -- pointer to display which previous screen should be get. (NULL to use the default screen)

返回 pointer to the previous screen object or NULL if not used now

void **lv_disp_load_scr**(*lv_obj_t* *scr)

Make a screen active

参数 **scr** -- pointer to a screen

lv_obj_t ***lv_disp_get_layer_top**(*lv_disp_t* *disp)

Return with the top layer. (Same on every screen and it is above the normal screen layer)

参数 **disp** -- pointer to display which top layer should be get. (NULL to use the default screen)

返回 pointer to the top layer object (transparent screen sized lv_obj)

lv_obj_t ***lv_disp_get_layer_sys**(*lv_disp_t* *disp)

Return with the sys. layer. (Same on every screen and it is above the normal screen and the top layer)

参数 **disp** -- pointer to display which sys. layer should be retrieved. (NULL to use the default screen)

返回 pointer to the sys layer object (transparent screen sized lv_obj)

void **lv_disp_set_theme**(*lv_disp_t* *disp, *lv_theme_t* *th)

Set the theme of a display

参数 **disp** -- pointer to a display

lv_theme_t ***lv_disp_get_theme**(*lv_disp_t* *disp)

Get the theme of a display

参数 **disp** -- pointer to a display

返回 the display's theme (can be NULL)

void **lv_disp_set_bg_color**(*lv_disp_t* *disp, lv_color_t color)

Set the background color of a display

参数

- **disp** -- pointer to a display
- **color** -- color of the background

void **lv_disp_set_bg_image**(*lv_disp_t* *disp, const void *img_src)

Set the background image of a display

参数

- **disp** -- pointer to a display
- **img_src** -- path to file or pointer to an *lv_img_dsc_t* variable

void **lv_disp_set_bg_opa**(*lv_disp_t* *disp, lv_opa_t opa)

Set opacity of the background

参数

- **disp** -- pointer to a display
- **opa** -- opacity (0..255)

void **lv_scr_load_anim**(*lv_obj_t* *scr, *lv_scr_load_anim_t* anim_type, uint32_t time, uint32_t delay, bool auto_del)

Switch screen with animation

参数

- **scr** -- pointer to the new screen to load

- **anim_type** -- type of the animation from `lv_scr_load_anim_t`. E.g. `LV_SCR_LOAD_ANIM_MOVE_LEFT`
- **time** -- time of the animation
- **delay** -- delay before the transition
- **auto_del** -- true: automatically delete the old screen

uint32_t **lv_disp_get_inactive_time**(const *lv_disp_t* *disp)

Get elapsed time since last user activity on a display (e.g. click)

参数 **disp** -- pointer to a display (NULL to get the overall smallest inactivity)

返回 elapsed ticks (milliseconds) since the last activity

void **lv_disp_trig_activity**(*lv_disp_t* *disp)

Manually trigger an activity on a display

参数 **disp** -- pointer to a display (NULL to use the default display)

void **lv_disp_clean_dcache**(*lv_disp_t* *disp)

Clean any CPU cache that is related to the display.

参数 **disp** -- pointer to a display (NULL to use the default display)

lv_timer_t ***_lv_disp_get_refr_timer**(*lv_disp_t* *disp)

Get a pointer to the screen refresher timer to modify its parameters with `lv_timer_...` functions.

参数 **disp** -- pointer to a display

返回 pointer to the display refresher timer. (NULL on error)

static inline *lv_obj_t* ***lv_scr_act**(void)

Get the active screen of the default display

返回 pointer to the active screen

static inline *lv_obj_t* ***lv_layer_top**(void)

Get the top layer of the default display

返回 pointer to the top layer

static inline *lv_obj_t* ***lv_layer_sys**(void)

Get the active screen of the default display

返回 pointer to the sys layer

static inline void **lv_scr_load**(*lv_obj_t* *scr)

static inline lv_coord_t **lv_dpx**(lv_coord_t n)

Scale the given number of pixels (a distance or size) relative to a 160 DPI display considering the DPI of the default display. It ensures that e.g. `lv_dpx(100)` will have the same physical size regardless to the DPI of the display.

参数 **n** -- the number of pixels to scale

返回 n x current_dpi/160

static inline lv_coord_t **lv_disp_dpx**(const *lv_disp_t* *disp, lv_coord_t n)

Scale the given number of pixels (a distance or size) relative to a 160 DPI display considering the DPI of the given display. It ensures that e.g. lv_dpx(100) will have the same physical size regardless to the DPI of the display.

参数

- **obj** -- a display whose dpi should be considered
- **n** -- the number of pixels to scale

返回 n x current_dpi/160

2.5.10 Colors（颜色）

The color module handles all color-related functions like changing color depth, creating colors from hex code, converting between color depths, mixing colors, etc.

lv_color_t is used to store a color, its fileds are set according to LV_COLOR_DEPTH in lv_conf.h. (See below)

You may set LV_COLOR_16_SWAP in lv_conf.h to swap the bytes of *RGB565* colors. You may need this to send the 16-bit colors via a byte-oriented interface like SPI. As 16-bit numbers are stored in Little Endian format (lower byte on the lower address), the interface will send the lower byte first. However, displays usually need the higher byte first. A mismatch in the byte order will result in highly distorted colors.

颜色模块处理所有与颜色相关的功能，如更改颜色深度、从十六进制代码创建颜色、颜色深度之间的转换、混合颜色等。

lv_color_t 用于存储颜色，其字段根据 lv_conf.h 中的 LV_COLOR_DEPTH 设置。（见下文）

你可以在 lv_conf.h 中设置 LV_COLOR_16_SWAP 来交换 *RGB565* 颜色的字节。您可能需要它来通过面向字节的接口（如 SPI）发送 16 位颜色。由于 16 位数字以 Little Endian 格式存储（低位字节在低位地址），因此接口将首先发送低位字节。但是，显示器通常首先需要较高的字节。字节顺序不匹配将导致颜色高度失真。

Creating colors（创造色彩）

RGB（三原色）

Create colors from Red, Green and Blue channel values

从红色、绿色和蓝色通道值创建颜色

```
//All channels are 0-255
lv_color_t c = lv_color_make(red, green, blue);
```

(下页继续)

(续上页)

```
//From hex code 0x000000..0xFFFFFF interpreted as RED + GREEN + BLUE
lv_color_t c = lv_color_hex(0x123456);

//From 3 digits. Same as lv_color_hex(0x112233)
lv_color_t c = lv_color_hex3(0x123);
```

HSV（色调饱和值-Hue Saturation Value）

Create colors from Hue, Saturation and Value values

根据色相、饱和度和值创建颜色

```
//h = 0..359, s = 0..100, v = 0..100
lv_color_t c = lv_color_hsv_to_rgb(h, s, v);

//All channels are 0-255
lv_color_hsv_t c_hsv = lv_color_rgb_to_hsv(r, g, b);


//From lv_color_t variable
lv_color_hsv_t c_hsv = lv_color_to_hsv(color);
```

Palette（调色板）

LVGL includes material design's palette. In this all color have a main as well as four darker and five lighter variants.

The names of the colors are as follows:

- LV_PALETTE_RED
- LV_PALETTE_PINK
- LV_PALETTE_PURPLE
- LV_PALETTE_DEEP_PURPLE
- LV_PALETTE_INDIGO
- LV_PALETTE_BLUE
- LV_PALETTE_LIGHT_BLUE
- LV_PALETTE_CYAN
- LV_PALETTE_TEAL
- LV_PALETTE_GREEN

- LV_PALETTE_LIGHT_GREEN
- LV_PALETTE_LIME
- LV_PALETTE_YELLOW
- LV_PALETTE_AMBER
- LV_PALETTE_ORANGE
- LV_PALETTE_DEEP_ORANGE
- LV_PALETTE_BROWN
- LV_PALETTE_BLUE_GREY
- LV_PALETTE_GREY

LVGL 包括材料设计的调色板。在此所有颜色都有一个主要的以及四个较深的变体和五个较浅的变体。

颜色名称如下：

- LV_PALETTE_RED
- LV_PALETTE_PINK
- LV_PALETTE_PURPLE
- LV_PALETTE_DEEP_PURPLE
- LV_PALETTE_INDIGO
- LV_PALETTE_BLUE
- LV_PALETTE_LIGHT_BLUE
- LV_PALETTE_CYAN
- LV_PALETTE_TEAL
- LV_PALETTE_GREEN
- LV_PALETTE_LIGHT_GREEN
- LV_PALETTE_LIME
- LV_PALETTE_YELLOW
- LV_PALETTE_AMBER
- LV_PALETTE_ORANGE
- LV_PALETTE_DEEP_ORANGE
- LV_PALETTE_BROWN
- LV_PALETTE_BLUE_GREY
- LV_PALETTE_GREY

To get the main color use `lv_color_t c = lv_palette_main(LV_PALETTE_...)`.

For the lighter variants of a palette color use `lv_color_t c = lv_palette_lighten(LV_PALETTE_..., v)`. `v` can be 1..5. For the darker variants of a palette color use `lv_color_t c = lv_palette_darken(LV_PALETTE_..., v)`. `v` can be 1..4.

要获得主要颜色，请使用 `lv_color_t c = lv_palette_main(LV_PALETTE_...)`。

对于调色板颜色的较浅变体，请使用 `lv_color_t c = lv_palette_lighten(LV_PALETTE_..., v)`。`v` 可以是 1..5。

对于调色板颜色的较暗变体，请使用 `lv_color_t c = lv_palette_darken(LV_PALETTE_..., v)`。`v` 可以是 1..4。

Modify and mix colors（修改和混合颜色）

The following functions can modify a color:

以下函数可以修改颜色：

```
// Lighten a color. 0: no change, 255: white
lv_color_t c = lv_color_lighten(c, lvl);

// Darken a color. 0: no change, 255: black
lv_color_t c = lv_color_darken(lv_color_t c, lv_opa_t lvl);

// Lighten or darken a color. 0: black, 128: no change 255: black
lv_color_t c = lv_color_change_lightness(lv_color_t c, lv_opa_t lvl);


// Mix 2 colors with a given ratio 0: full c2, 255: full c1, 128: half c1 and half c2
lv_color_t c = lv_color_mix(c1, c2, ratio);
```

Built-in colors（内置颜色）

`lv_color_white()` and `lv_color_black()` return `0xFFFFFF` and `0x000000` respectively.

`lv_color_white()` 和 `lv_color_black()` 分别返回 `0xFFFFFF` 和 `0x000000`。

Opacity（不透明度）

To describe opacity the `lv_opa_t` type is created as a wrapper to `uint8_t`. Some defines are also introduced:

- `LV_OPA_TRANSP` Value: 0, means the opacity makes the color completely transparent
- `LV_OPA_10` Value: 25, means the color covers only a little
- `LV_OPA_20 ... OPA_80` come logically
- `LV_OPA_90` Value: 229, means the color near completely covers
- `LV_OPA_COVER` Value: 255, means the color completely covers

You can also use the `LV_OPA_*` defines in `lv_color_mix()` as a *ratio*.

Color types（颜色类型）

The following variable types are defined by the color module:

- `lv_color1_t` Monochrome color. Also has R, G, B fields for compatibility but they are always the same value (1 byte)
- `lv_color8_t` A structure to store R (3 bit),G (3 bit),B (2 bit) components for 8-bit colors (1 byte)
- `lv_color16_t` A structure to store R (5 bit),G (6 bit),B (5 bit) components for 16-bit colors (2 byte)
- `lv_color32_t` A structure to store R (8 bit),G (8 bit), B (8 bit) components for 24-bit colors (4 byte)
- `lv_color_t` Equal to `lv_color1/8/16/24_t` depending on current color depth setting
- `lv_color_int_t` `uint8_t`, `uint16_t` or `uint32_t` depending on color depth setting. Used to build color arrays from plain numbers.
- `lv_opa_t` A simple `uint8_t` type to describe opacity.

The `lv_color_t`, `lv_color1_t`, `lv_color8_t`, `lv_color16_t` and `lv_color32_t` types have four fields:

- `ch.red` red channel
- `ch.green` green channel
- `ch.blue` blue channel
- `full*` red + green + blue as one number

You can set the current color depth in *lv_conf.h*, by setting the `LV_COLOR_DEPTH` define to 1 (monochrome), 8, 16 or 32.

Convert color（颜色转换）

You can convert a color from the current color depth to another. The converter functions return with a number, so you have to use the `full` field:

```
lv_color_t c;
c.red   = 0x38;
c.green = 0x70;
c.blue  = 0xCC;

lv_color1_t c1;
c1.full = lv_color_to1(c);        /*Return 1 for light colors, 0 for dark colors*/

lv_color8_t c8;
c8.full = lv_color_to8(c);        /*Give a 8 bit number with the converted color*/

lv_color16_t c16;
c16.full = lv_color_to16(c); /*Give a 16 bit number with the converted color*/

lv_color32_t c24;
c32.full = lv_color_to32(c);        /*Give a 32 bit number with the converted color*/
```

API

Typedefs

typedef lv_color_t (***lv_color_filter_cb_t**)(const struct *_lv_color_filter_dsc_t**, lv_color_t, lv_opa_t)

typedef struct *_lv_color_filter_dsc_t* **lv_color_filter_dsc_t**

Enums

enum **[anonymous]**

Opacity percentages.

Values:

enumerator **LV_OPA_TRANSP**

enumerator **LV_OPA_0**

enumerator **LV_OPA_10**

enumerator **LV_OPA_20**

enumerator **LV_OPA_30**

enumerator **LV_OPA_40**

enumerator **LV_OPA_50**

enumerator **LV_OPA_60**

enumerator **LV_OPA_70**

enumerator **LV_OPA_80**

enumerator **LV_OPA_90**

enumerator **LV_OPA_100**

enumerator **LV_OPA_COVER**

enum **lv_palette_t**

Values:

enumerator **LV_PALETTE_RED**

enumerator **LV_PALETTE_PINK**

enumerator **LV_PALETTE_PURPLE**

enumerator **LV_PALETTE_DEEP_PURPLE**

enumerator **LV_PALETTE_INDIGO**

enumerator **LV_PALETTE_BLUE**

enumerator **LV_PALETTE_LIGHT_BLUE**

enumerator **LV_PALETTE_CYAN**

enumerator **LV_PALETTE_TEAL**

enumerator **LV_PALETTE_GREEN**

enumerator **LV_PALETTE_LIGHT_GREEN**

enumerator **LV_PALETTE_LIME**

enumerator **LV_PALETTE_YELLOW**

enumerator **LV_PALETTE_AMBER**

enumerator **LV_PALETTE_ORANGE**

enumerator **LV_PALETTE_DEEP_ORANGE**

enumerator **LV_PALETTE_BROWN**

enumerator **LV_PALETTE_BLUE_GREY**

enumerator **LV_PALETTE_GREY**

enumerator **_LV_PALETTE_LAST**

enumerator **LV_PALETTE_NONE**

Functions

LV_EXPORT_CONST_INT(LV_COLOR_DEPTH)

LV_EXPORT_CONST_INT(LV_COLOR_16_SWAP)

typedef LV_CONCAT3 (uint, LV_COLOR_SIZE, _t) lv_color_int_t

typedef LV_CONCAT3 (lv_color, LV_COLOR_DEPTH, _t) lv_color_t

static inline uint8_t **lv_color_to1**(lv_color_t color)

static inline uint8_t **lv_color_to8**(lv_color_t color)

static inline uint16_t **lv_color_to16**(lv_color_t color)

static inline uint32_t **lv_color_to32**(lv_color_t color)

static inline uint8_t **lv_color_brightness**(lv_color_t color)

Get the brightness of a color

参数 **color** -- a color

返回 the brightness [0..255]

static inline lv_color_t **lv_color_make**(uint8_t r, uint8_t g, uint8_t b)

static inline lv_color_t **lv_color_hex**(uint32_t c)

static inline lv_color_t **lv_color_hex3**(uint32_t c)

static inline void **lv_color_filter_dsc_init**(*lv_color_filter_dsc_t* *dsc, *lv_color_filter_cb_t* cb)

lv_color_t **lv_color_lighten**(lv_color_t c, lv_opa_t lvl)

lv_color_t **lv_color_darken**(lv_color_t c, lv_opa_t lvl)

lv_color_t **lv_color_change_lightness**(lv_color_t c, lv_opa_t lvl)

lv_color_t **lv_color_hsv_to_rgb**(uint16_t h, uint8_t s, uint8_t v)

Convert a HSV color to RGB

参数

- **h** -- hue [0..359]
- **s** -- saturation [0..100]
- **v** -- value [0..100]

返回 the given RGB color in RGB (with LV_COLOR_DEPTH depth)

lv_color_hsv_t **lv_color_rgb_to_hsv**(uint8_t r8, uint8_t g8, uint8_t b8)

Convert a 32-bit RGB color to HSV

参数

- **r8** -- 8-bit red
- **g8** -- 8-bit green
- **b8** -- 8-bit blue

返回 the given RGB color in HSV

lv_color_hsv_t **lv_color_to_hsv**(lv_color_t color)

Convert a color to HSV

参数 **color** -- color

返回 the given color in HSV

static inline lv_color_t **lv_color_chroma_key**(void)

Just a wrapper around LV_COLOR_CHROMA_KEY because it might be more convenient to use a function is some cases

返回 LV_COLOR_CHROMA_KEY

lv_color_t **lv_palette_main**(*lv_palette_t* p)

static inline lv_color_t **lv_color_white**(void)

static inline lv_color_t **lv_color_black**(void)

lv_color_t **lv_palette_lighten**(*lv_palette_t* p, uint8_t lvl)

lv_color_t **lv_palette_darken**(*lv_palette_t* p, uint8_t lvl)

union **lv_color1_t**

Public Members

uint8_t **full**

uint8_t **blue**

uint8_t **green**

uint8_t **red**

union *lv_color1_t*::[anonymous] **ch**

union **lv_color8_t**

Public Members

uint8_t **blue**

uint8_t **green**

uint8_t **red**

struct *lv_color8_t*::[anonymous] **ch**

uint8_t **full**

union **lv_color16_t**

Public Members

uint16_t **blue**

uint16_t **green**

uint16_t **red**

uint16_t **green_h**

uint16_t **green_l**

struct *lv_color16_t*::[anonymous] **ch**

uint16_t **full**

union **lv_color32_t**

Public Members

uint8_t **blue**

uint8_t **green**

uint8_t **red**

uint8_t **alpha**

struct *lv_color32_t*::[anonymous] **ch**

uint32_t **full**

struct **lv_color_hsv_t**

Public Members

uint16_t **h**

uint8_t **s**

uint8_t **v**

struct **_lv_color_filter_dsc_t**

Public Members

lv_color_filter_cb_t **filter_cb**

void ***user_data**

2.5.11 Fonts（字体）

In LVGL fonts are collections of bitmaps and other information required to render the images of the letters (glyph). A font is stored in a `lv_font_t` variable and can be set in a style's *text_font* field. For example:

在 LVGL 中，字体是渲染字母（字形）图像所需的位图和其他信息的集合。字体存储在 `lv_font_t` 变量中，可以在样式的 *text_font* 字段中设置。例如：

```
lv_style_set_text_font(&my_style, LV_STATE_DEFAULT, &lv_font_montserrat_28);  /*Set a␣
↪larger font*/
```

The fonts have a **bpp (bits per pixel)** property. It shows how many bits are used to describe a pixel in the font. The value stored for a pixel determines the pixel's opacity. This way, with higher *bpp*, the edges of the letter can be smoother. The possible *bpp* values are 1, 2, 4 and 8 (higher value means better quality).

The *bpp* also affects the required memory size to store the font. For example, *bpp = 4* makes the font nearly 4 times larger compared to *bpp = 1*.

字体具有 **bpp（每像素位数）**属性。它显示了使用多少位来描述字体中的像素。为像素存储的值决定了像素的不透明度。这样，使用更高的 *bpp*，字母的边缘可以更平滑。可能的 *bpp* 值为 1、2、4 和 8（值越高表示质量越好）。

bpp 还会影响存储字体所需的内存大小。例如，*bpp = 4* 使字体比 *bpp = 1* 大近 4 倍。

Unicode support（支持 Unicode 编码）

LVGL supports **UTF-8** encoded Unicode characters. Your editor needs to be configureed to save your code/text as UTF-8 (usually this the default) and be sure that, `LV_TXT_ENC` is set to `LV_TXT_ENC_UTF8` in *lv_conf.h*. (This is the default value)

To test it try

LVGL 支持 **UTF-8** 编码的 Unicode 字符。您的编辑器需要配置为将您的代码/文本保存为 UTF-8（通常这是默认值），并确保在 *lv_conf.h* 中将 `LV_TXT_ENC` 设置为 `LV_TXT_ENC_UTF8`。（这是默认值）

要测试它，请参考这个用法：

```
lv_obj_t * label1 = lv_label_create(lv_scr_act(), NULL);
lv_label_set_text(label1, LV_SYMBOL_OK);
```

If all works well, a ✓ character should be displayed.

如果一切正常，应显示 ✓ 字符。

Built-in fonts（内置字体）

There are several built-in fonts in different sizes, which can be enabled in `lv_conf.h` by *LV_FONT_...* defines.

有几种不同大小的内置字体，可以通过 *LV_FONT_...* 定义在 `lv_conf.h` 中启用。

Normal fonts（普通字体）

Containing all the ASCII characters, the degree symbol (U+00B0), the bullet symbol (U+2022) and the built-in symbols (see below).

- `LV_FONT_MONTSERRAT_12` 12 px font
- `LV_FONT_MONTSERRAT_14` 14 px font
- `LV_FONT_MONTSERRAT_16` 16 px font
- `LV_FONT_MONTSERRAT_18` 18 px font
- `LV_FONT_MONTSERRAT_20` 20 px font
- `LV_FONT_MONTSERRAT_22` 22 px font
- `LV_FONT_MONTSERRAT_24` 24 px font
- `LV_FONT_MONTSERRAT_26` 26 px font
- `LV_FONT_MONTSERRAT_28` 28 px font
- `LV_FONT_MONTSERRAT_30` 30 px font
- `LV_FONT_MONTSERRAT_32` 32 px font
- `LV_FONT_MONTSERRAT_34` 34 px font
- `LV_FONT_MONTSERRAT_36` 36 px font
- `LV_FONT_MONTSERRAT_38` 38 px font
- `LV_FONT_MONTSERRAT_40` 40 px font
- `LV_FONT_MONTSERRAT_42` 42 px font
- `LV_FONT_MONTSERRAT_44` 44 px font
- `LV_FONT_MONTSERRAT_46` 46 px font
- `LV_FONT_MONTSERRAT_48` 48 px font

包含所有 ASCII 字符、度数符号 (U+00B0)、子弹符号 (U+2022) 和内置符号（见下文）。

- `LV_FONT_MONTSERRAT_12` 12 像素字体
- `LV_FONT_MONTSERRAT_14` 14 像素字体
- `LV_FONT_MONTSERRAT_16` 16 像素字体
- `LV_FONT_MONTSERRAT_18` 18 像素字体
- `LV_FONT_MONTSERRAT_20` 20 像素字体
- `LV_FONT_MONTSERRAT_22` 22 像素字体
- `LV_FONT_MONTSERRAT_24` 24 像素字体

- LV_FONT_MONTSERRAT_26 26 像素字体
- LV_FONT_MONTSERRAT_28 28 像素字体
- LV_FONT_MONTSERRAT_30 30 像素字体
- LV_FONT_MONTSERRAT_32 32 像素字体
- LV_FONT_MONTSERRAT_34 34 像素字体
- LV_FONT_MONTSERRAT_36 36 像素字体
- LV_FONT_MONTSERRAT_38 38 像素字体
- LV_FONT_MONTSERRAT_40 40 像素字体
- LV_FONT_MONTSERRAT_42 42 像素字体
- LV_FONT_MONTSERRAT_44 44 像素字体
- LV_FONT_MONTSERRAT_46 46 像素字体
- LV_FONT_MONTSERRAT_48 48 像素字体

Special fonts（特殊字体）

- LV_FONT_MONTSERRAT_12_SUBPX Same as normal 12 px font but with *subpixel rendering*
- LV_FONT_MONTSERRAT_28_COMPRESSED Same as normal 28 px font but *compressed font* with 3 bpp
- LV_FONT_DEJAVU_16_PERSIAN_HEBREW 16 px font with normal range + Hebrew, Arabic, Persian letters and all their forms
- LV_FONT_SIMSUN_16_CJK16 px font with normal range + 1000 most common CJK radicals
- LV_FONT_UNSCII_8 8 px pixel perfect font with only ASCII characters
- LV_FONT_UNSCII_16 16 px pixel perfect font with only ASCII characters

The built-in fonts are **global variables** with names like `lv_font_montserrat_16` for a 16 px hight font. To use them in a style, just add a pointer to a font variable like shown above.

The built-in fonts with *bpp = 4* contain the ASCII characters and use the Montserrat font.

In addition to the ASCII range, the following symbols are also added to the built-in fonts from the FontAwesome font.

- LV_FONT_MONTSERRAT_12_SUBPX 与普通 12 px 字体相同，但具有子像素渲染
- LV_FONT_MONTSERRAT_28_COMPRESSED 与普通 28 px 字体相同，但压缩字体 为 3 bpp
- LV_FONT_DEJAVU_16_PERSIAN_HEBREW 16 px 字体，正常范围 + 希伯来语、阿拉伯语、波斯语字母及其所有形式
- LV_FONT_SIMSUN_16_CJK16 px 字体，正常范围 + 1000 个最常见的 CJK 部首
- LV_FONT_UNSCII_8 8 px 像素完美字体，只有 ASCII 字符

- `LV_FONT_UNSCII_16` 16 px 像素完美字体，只有 ASCII 字符

内置字体是**全局变量**，其名称类似于 16 像素高字体的 “lv_font_montserrat_16”。要在样式中使用它们，只需添加一个指向字体变量的指针，如上所示。

bpp = 4 的内置字体包含 ASCII 字符并使用 Montserrat 字体。

除了 ASCII 范围外，以下符号还添加到 FontAwesome 字体的内置字体中。

LV_SYMBOL_AUDIO
LV_SYMBOL_VIDEO
LV_SYMBOL_LIST
LV_SYMBOL_OK
LV_SYMBOL_CLOSE
LV_SYMBOL_POWER
LV_SYMBOL_SETTINGS
LV_SYMBOL_TRASH
LV_SYMBOL_HOME
LV_SYMBOL_DOWNLOAD
LV_SYMBOL_DRIVE
LV_SYMBOL_REFRESH
LV_SYMBOL_MUTE
LV_SYMBOL_VOLUME_MID
LV_SYMBOL_VOLUME_MAX
LV_SYMBOL_IMAGE
LV_SYMBOL_EDIT
LV_SYMBOL_PREV
LV_SYMBOL_PLAY
LV_SYMBOL_PAUSE
LV_SYMBOL_STOP
LV_SYMBOL_NEXT
LV_SYMBOL_EJECT
LV_SYMBOL_LEFT
LV_SYMBOL_RIGHT
LV_SYMBOL_PLUS
LV_SYMBOL_MINUS
LV_SYMBOL_EYE_OPEN
LV_SYMBOL_EYE_CLOSE
LV_SYMBOL_WARNING
LV_SYMBOL_SHUFFLE
LV_SYMBOL_UP
LV_SYMBOL_DOWN
LV_SYMBOL_LOOP
LV_SYMBOL_DIRECTORY
LV_SYMBOL_UPLOAD
LV_SYMBOL_CALL
LV_SYMBOL_CUT
LV_SYMBOL_COPY
LV_SYMBOL_SAVE
LV_SYMBOL_CHARGE
LV_SYMBOL_PASTE
LV_SYMBOL_BELL
LV_SYMBOL_KEYBOARD
LV_SYMBOL_GPS
LV_SYMBOL_FILE
LV_SYMBOL_WIFI
LV_SYMBOL_BATTERY_FULL
LV_SYMBOL_BATTERY_3
LV_SYMBOL_BATTERY_2
LV_SYMBOL_BATTERY_1
LV_SYMBOL_BATTERY_EMPTY
LV_SYMBOL_USB
LV_SYMBOL_BLUETOOTH
LV_SYMBOL_BACKSPACE
LV_SYMBOL_SD_CARD
LV_SYMBOL_NEW_LINE

The symbols can be used as:

这些符号可以这样使用：

```
lv_label_set_text(my_label, LV_SYMBOL_OK);
```

Or with together with strings:

或与字符串一起使用：

```
lv_label_set_text(my_label, LV_SYMBOL_OK "Apply");
```

Or more symbols together:

一个或多个符号组合在一起：

```
lv_label_set_text(my_label, LV_SYMBOL_OK LV_SYMBOL_WIFI LV_SYMBOL_PLAY);
```

Special features（特殊功能）

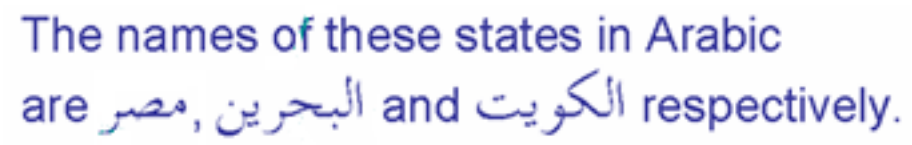
Bidirectional support（双向支持）

Most of the languages use Left-to-Right (LTR for short) writing direction, however some languages (such as Hebrew, Persian or Arabic) uses Right-to-Left (RTL for short) direction.

LVGL not only supports RTL texts but supports mixed (a.k.a. bidirectional, BiDi) text rendering too. Some examples:

大多数语言使用从左到右（简称 LTR）书写方向，但是一些语言（例如希伯来语、波斯语或阿拉伯语）使用从右到左（简称 RTL）方向。

LVGL 不仅支持 RTL 文本，还支持混合（又名双向，BiDi）文本渲染。一些例子：



BiDi support is enabled by `LV_USE_BIDI` in *lv_conf.h*

All texts have a base direction (LTR or RTL) which determines some rendering rules and the default alignment of the text (Left or Right). However, in LVGL, base direction is applied not only for labels. It's a general property which can be set for every object. If unset then it will be inherited from the parent. So it's enough to set the base direction of the screen and every object will inherit it.

The default base direction of screen can be set by `LV_BIDI_BASE_DIR_DEF` in *lv_conf.h* and other objects inherit the base direction from their parent.

BiDi 支持由 *lv_conf.h* 中的 `LV_USE_BIDI` 启用

所有文本都有一个基本方向（LTR 或 RTL），它决定了一些渲染规则和文本的默认对齐方式（左或右）。然而，在 LVGL 中，基本方向不仅适用于标签。这是一个可以为每个对象设置的通用属性。如果未设置，则它将从父级继承。所以设置屏幕的基本方向就足够了，每个对象都会继承它。

屏幕的默认基本方向可以通过 *lv_conf.h* 中的 `LV_BIDI_BASE_DIR_DEF` 设置，其他对象从其父对象继承基本方向。

To set an object's base direction use `lv_obj_set_base_dir(obj, base_dir)`. The possible base direction are:

- `LV_BIDI_DIR_LTR`: Left to Right base direction
- `LV_BIDI_DIR_RTL`: Right to Left base direction
- `LV_BIDI_DIR_AUTO`: Auto detect base direction
- `LV_BIDI_DIR_INHERIT`: Inherit the base direction from the parent (default for non-screen objects)

This list summarizes the effect of RTL base direction on objects:

- Create objects by default on the right
- `lv_tabview`: displays tabs from right to left
- `lv_checkbox`: Show the box on the right
- `lv_btnmatrix`: Show buttons from right to left
- `lv_list`: Show the icon on the right
- `lv_dropdown`: Align the options to the right
- The texts in `lv_table`, `lv_btnmatrix`, `lv_keyboard`, `lv_tabview`, `lv_dropdown`, `lv_roller` are "BiDi processed" to be displayed correctly

要设置对象的基本方向，请使用 `lv_obj_set_base_dir(obj, base_dir)`。可能的基本方向是：

- `LV_BIDI_DIR_LTR`：从左到右的基本方向
- `LV_BIDI_DIR_RTL`：从右到左的基本方向
- `LV_BIDI_DIR_AUTO`：自动检测基本方向
- `LV_BIDI_DIR_INHERIT`：从父级继承基本方向（非屏幕对象的默认值）

此列表总结了 RTL 基本方向对对象的影响：

- 默认在右侧创建对象
- `lv_tabview`：从右到左显示标签
- `lv_checkbox`：显示右侧的框
- `lv_btnmatrix`：从右到左显示按钮
- `lv_list`：在右侧显示图标

- `lv_dropdown`：将选项向右对齐
- `lv_table`、`lv_btnmatrix`、`lv_keyboard`、`lv_tabview`、`lv_dropdown`、`lv_roller` 中的文本是“BiDi 处理”以正确显示

Arabic and Persian support(阿拉伯语和波斯语支持)

There are some special rules to display Arabic and Persian characters: the *form* of the character depends on their position in the text. A different form of the same letter needs to be used if it isolated, start, middle or end position. Besides these some conjunction rules also should be taken into account.

LVGL supports to apply these rules if `LV_USE_ARABIC_PERSIAN_CHARS` is enabled.

显示阿拉伯语和波斯语字符有一些特殊规则：字符的形式取决于它们在文本中的位置。如果同一个字母是孤立的、开始的、中间的或结束的位置，则需要使用不同形式的相同字母。除了这些，还应该考虑一些连词规则。

如果启用了“LV_USE_ARABIC_PERSIAN_CHARS”，LVGL 支持应用这些规则。

However, there some limitations:

- Only displaying texts is supported (e.g. on labels), text inputs (e.g. text area) don't support this feature.
- Static text (i.e. const) is not processed. E.g. texts set by `lv_label_set_text()` will be "Arabic processed" but `lv_lable_set_text_static()` won't.
- Text get functions (e.g. `lv_label_get_text()`) will return the processed text.

但是，有一些限制：

- 仅支持显示文本（例如在标签上），文本输入（例如文本区域）不支持此功能。
- 不处理静态文本（即 const）。例如。`lv_label_set_text()` 设置的文本将是“阿拉伯语处理”，但 `lv_lable_set_text_static()` 不会。
- 文本获取函数（例如 `lv_label_get_text()`）将返回处理后的文本。

Subpixel rendering（亚像素渲染）

Subpixel rendering allows for tripling the horizontal resolution by rendering on Red, Green and Blue channel instead of pixel level. This takes advantage of the position of physical color channels of each pixel, resulting in higher quality letter anti-aliasing. Learn more here.

For subpixel rendering the fonts need to be generated with special settings:

- In the online converter tick the `Subpixel` box
- In the command line tool use `--lcd` flag. Note that the generated font needs about 3 times more memory.

Subpixel rendering works only if the color channels of the pixels have a horizontal layout. That is the R, G, B channels are next each other and not above each other. The order of color channels also needs to match with the library settings. By default LVGL assumes `RGB` order, however this can be swapped by setting `LV_SUBPX_BGR 1` in *lv_conf.h*.

亚像素渲染通过在红色、绿色和蓝色通道（而不是像素级）上渲染，允许将水平分辨率提高三倍。这将利用每个像素的物理颜色通道的位置，从而实现更高质量的字母消除混叠。了解更多此处.

对于亚像素渲染，需要使用特殊设置生成字体：-在在线转换器中，勾选“亚像素”框 -在命令行工具中，使用“---lcd”标志。请注意，生成的字体需要大约 3 倍的内存。

仅当像素的颜色通道具有水平布局时，子像素渲染才起作用。也就是说，R、G、B 通道彼此相邻，而不是彼此上方。颜色通道的顺序也需要与库设置相匹配。默认情况下，LVGL 采用“RGB”顺序，但这可以通过在 *LV_conf.h* 中设置“LV_SUBPX_BGR 1”进行交换。

Compress fonts（压缩字体）

The bitmaps of the fonts can be compressed by

- ticking the `Compressed` check box in the online converter
- not passing `--no-compress` flag to the offline converter (compression is applied by default)

The compression is more effective with larger fonts and higher bpp. However, it's about 30% slower to render the compressed fonts. Therefore it's recommended to compress only the largest fonts of user interface, because

- they need the most memory
- they can be compressed better
- and probably they are used less frequently then the medium sized fonts, so the performance cost is smaller.

字体的位图可以通过以下方式压缩

- 勾选在线转换器中的“压缩”复选框
- 不将 `--no-compress` 标志传递给离线转换器（默认情况下应用压缩）

对于更大的字体和更高的 bpp，压缩更有效。但是，渲染压缩字体的速度要慢 30% 左右。因此建议只压缩用户界面的最大字体，因为

- 他们需要最多的内存
- 它们可以被更好地压缩
- 可能它们的使用频率低于中等字体，因此性能成本更小。

Add new font（添加新字体）

There are several ways to add a new font to your project:

1. The simplest method is to use the Online font converter. Just set the parameters, click the *Convert* button, copy the font to your project and use it. **Be sure to carefully read the steps provided on that site or you will get an error while converting.**
2. Use the Offline font converter. (Requires Node.js to be installed)
3. If you want to create something like the built-in fonts (Roboto font and symbols) but in different size and/or ranges, you can use the `built_in_font_gen.py` script in `lvgl/scripts/built_in_font` folder. (This requires Python and `lv_font_conv` to be installed)

To declare the font in a file, use `LV_FONT_DECLARE(my_font_name)`.

To make the fonts globally available (like the builtin fonts), add them to `LV_FONT_CUSTOM_DECLARE` in *lv_conf.h*.

有几种方法可以将新字体添加到您的项目中：

1. 最简单的方法是使用【在线字体转换器】(https://lvgl.io/tools/fontconverter)。只需设置参数，单击 *Convert* 按钮，将字体复制到您的项目并使用它。**请务必仔细阅读该网站上提供的步骤，否则转换时会出错**。
2. 使用【离线字体转换器】(https://github.com/lvgl/lv_font_conv)。（需要安装 Node.js）
3. 如果您想创建类似内置字体（Roboto 字体和符号）但大小和/或范围不同的内容，您可以使用 `lvgl/scripts/built_in_font` 文件夹中的 `built_in_font_gen.py` 脚本。（这需要安装 Python 和 `lv_font_conv`）

要在文件中声明字体，请使用 `LV_FONT_DECLARE(my_font_name)`。

要使字体全局可用（如内置字体），请将它们添加到 *lv_conf.h* 中的 `LV_FONT_CUSTOM_DECLARE`。

Add new symbols（添加新符号）

The built-in symbols are created from the FontAwesome font.

1. Search symbol on https://fontawesome.com. For example the USB symbol. Copy it's Unicode ID which is `0xf287` in this case.
2. Open the Online font converter. Add Add FontAwesome.woff. .
3. Set the parameters such as Name, Size, BPP. You'll use this name to declare and use the font in your code.
4. Add the Unicode ID of the symbol to the range field. E.g. `0xf287` for the USB symbol. More symbols can be enumerated with `,`.
5. Convert the font and copy it to your project. Make sure to compile the .c file of your font.
6. Declare the font using `extern lv_font_t my_font_name;` or simply `LV_FONT_DECLARE(my_font_name);`.

Using the symbol

内置符号是从 FontAwesome 字体创建的。

1. 在https://fontawesome.com上搜索符号。例如 USB 符号。复制它的 Unicode ID，在本例中为 “0xf287”。
2. 打开【在线字体转换器】(https://lvgl.io/tools/fontconverter)。添加添加 FontAwesome.woff。.
3. 设置 Name、Size、BPP 等参数。您将使用此名称在代码中声明和使用字体。
4. 将符号的 Unicode ID 添加到范围字段中。例如，USB 符号的 `0xf287`。更多的符号可以用，来枚举。
5. 转换字体并将其复制到您的项目中。确保编译字体的.c 文件。
6. 使用 `extern lv_font_t my_font_name;` 或简单的 `LV_FONT_DECLARE(my_font_name);` 声明字体。

使用符号

1. Convert the Unicode value to UTF8, for example on this site. For `0xf287` the *Hex UTF-8 bytes* are `EF 8A 87`.
2. Create a `define` from the UTF8 values: `#define MY_USB_SYMBOL "\xEF\x8A\x87"`
3. Create a label and set the text. Eg. `lv_label_set_text(label, MY_USB_SYMBOL)`

Note - `lv_label_set_text(label, MY_USB_SYMBOL)` searches for this symbol in the font defined in `style.text.font` properties. To use the symbol you may need to change it. Eg `style.text.font = my_font_name`

1. 将 Unicode 值转换为 UTF8，例如在本站。对于 `0xf287`，*Hex UTF-8* 字节是 `EF 8A 87`。
2. 从 UTF8 值创建一个 `define`：`#define MY_USB_SYMBOL "\xEF\x8A\x87"`
3. 创建标签并设置文本。例如。`lv_label_set_text`（标签，`MY_USB_SYMBOL`）

注意 - `lv_label_set_text(label, MY_USB_SYMBOL)` 在 `style.text.font` 属性中定义的字体中搜索此符号。要使用该符号，您可能需要对其进行更改。例如 `style.text.font = my_font_name`

Load font at run-time（在运行时加载字体）

`lv_font_load` can be used to load a font from a file. The font to load needs to have a special binary format. (Not TTF or WOFF). Use lv_font_conv with `--format bin` option to generate an LVGL compatible font file.

Note that to load a font *LVGL's filesystem* needs to be enabled and a driver needs to be added.

Example

`lv_font_load` 可用于从文件加载字体。要加载的字体需要具有特殊的二进制格式。(不是 TTF 或 WOFF)。使用 lv_font_conv 和 `--format bin` 选项来生成 LVGL 兼容字体文件。

请注意，要加载字体*LVGL 的文件系统* 需要启用，并且需要添加驱动程序。

例子

```
lv_font_t * my_font;
my_font = lv_font_load(X/path/to/my_font.bin);

/*Use the font*/

/*Free the font if not required anymore*/
lv_font_free(my_font);
```

Add a new font engine（添加新的字体引擎）

LVGL's font interface is designed to be very flexible. But even so you don't need to use LVGL's internal font engine: you can add your own. For example, use FreeType to real-time render glyphs from TTF fonts or use an external flash to store the font's bitmap and read them when the library needs them.

A ready to use FreeType can be found in lv_freetype repository.

To do this a custom `lv_font_t` variable needs to be created:

LVGL 的字体界面设计得非常灵活。但即便如此，您也不需要使用 LVGL 的内部字体引擎：您可以添加自己的字体引擎。例如，使用 FreeType 从 TTF 字体实时渲染字形或使用外部闪存存储字体的位图并在库需要时读取它们。

可以在 lv_freetype 存储库中找到准备使用的 FreeType。

为此，需要创建一个自定义的 `lv_font_t` 变量：

```
/*Describe the properties of a font*/
lv_font_t my_font;
my_font.get_glyph_dsc = my_get_glyph_dsc_cb;        /*Set a callback to get info␣
↪about gylphs*/
my_font.get_glyph_bitmap = my_get_glyph_bitmap_cb;  /*Set a callback to get bitmap of␣
↪a glyp*/
my_font.line_height = height;                       /*The real line height where any␣
↪text fits*/
my_font.base_line = base_line;                      /*Base line measured from the top␣
↪of line_height*/
my_font.dsc = something_required;                   /*Store any implementation␣
↪specific data here*/
my_font.user_data = user_data;                      /*Optionally some extra user␣
↪data*/

...

/* Get info about glyph of `unicode_letter` in `font` font.
 * Store the result in `dsc_out`.
```

(下页继续)

(续上页)

```
 * The next letter (`unicode_letter_next`) might be used to calculate the width␣
↪required by this glyph (kerning)
 */
bool my_get_glyph_dsc_cb(const lv_font_t * font, lv_font_glyph_dsc_t * dsc_out,␣
↪uint32_t unicode_letter, uint32_t unicode_letter_next)
{
    /*Your code here*/

    /* Store the result.
     * For example ...
     */
    dsc_out->adv_w = 12;        /*Horizontal space required by the glyph in [px]*/
    dsc_out->box_h = 8;         /*Height of the bitmap in [px]*/
    dsc_out->box_w = 6;         /*Width of the bitmap in [px]*/
    dsc_out->ofs_x = 0;         /*X offset of the bitmap in [pf]*/
    dsc_out->ofs_y = 3;         /*Y offset of the bitmap measured from the as line*/
    dsc_out->bpp   = 2;         /*Bits per pixel: 1/2/4/8*/

    return true;                /*true: glyph found; false: glyph was not found*/
}


/* Get the bitmap of `unicode_letter` from `font`. */
const uint8_t * my_get_glyph_bitmap_cb(const lv_font_t * font, uint32_t unicode_
↪letter)
{
    /* Your code here */

    /* The bitmap should be a continuous bitstream where
     * each pixel is represented by `bpp` bits */

    return bitmap;    /*Or NULL if not found*/
}
```

2.5.12 Images（图象）

An image can be a file or variable which stores the bitmap itself and some metadata.

图像可以是存储位图本身和一些元数据的文件或变量。

Store images（存储图像）

You can store images in two places

- as a variable in the internal memory (RAM or ROM)
- as a file

您可以将图像存储在两个地方

- 作为内部存储器（RAM 或 ROM）中的变量
- 作为文件

Variables（变量）

The images stored internally in a variable are composed mainly of an `lv_img_dsc_t` structure with the following fields:

- **header**
  - *cf* Color format. See *below*
  - *w* width in pixels (<= 2048)
  - *h* height in pixels (<= 2048)
  - *always zero* 3 bits which need to be always zero
  - *reserved* reserved for future use
- **data** pointer to an array where the image itself is stored
- **data_size** length of `data` in bytes

These are usually stored within a project as C files. They are linked into the resulting executable like any other constant data.

内部存储在变量中的图像主要由具有以下字段的 `lv_img_dsc_t` 结构组成：

- **标题**
- *cf* 颜色格式。见下面
- *w* 像素宽度 (<= 2048)
- *h* 以像素为单位的高度 (<= 2048)

- 始终为零 3 位，需要始终为零
- 保留保留供将来使用
- **data** 指向存储图像本身的数组的指针
- **data_size** `data` 的长度（以字节为单位）

这些通常作为 C 文件存储在项目中。它们像任何其他常量数据一样链接到生成的可执行文件中。

Files（文件）

To deal with files you need to add a *Drive* to LVGL. In short, a *Drive* is a collection of functions (*open*, *read*, *close*, etc.) registered in LVGL to make file operations. You can add an interface to a standard file system (FAT32 on SD card) or you create your simple file system to read data from an SPI Flash memory. In every case, a *Drive* is just an abstraction to read and/or write data to memory. See the *File system* section to learn more.

Images stored as files are not linked into the resulting executable, and must be read to RAM before being drawn. As a result, they are not as resource-friendly as variable images. However, they are easier to replace without needing to recompile the main program.

要处理文件，您需要将 *Drive* 添加到 LVGL。简而言之，*Drive* 是在 LVGL 中注册的用于进行文件操作的函数（*open*、*read*、*close* 等）的集合。您可以向标准文件系统（SD 卡上的 FAT32）添加接口，或者创建简单的文件系统以从 SPI 闪存读取数据。在任何情况下，*Drive* 都只是读取和/或将数据写入内存的抽象。请参阅文件系统 部分了解更多信息。

存储为文件的图像不会链接到生成的可执行文件中，并且必须在绘制之前读取到 RAM。因此，它们不像可变图像那样资源友好。但是，它们更容易替换而无需重新编译主程序。

Color formats（颜色格式）

Various built-in color formats are supported:

- **LV_IMG_CF_TRUE_COLOR** Simply stores the RGB colors (in whatever color depth LVGL is configured for).
- **LV_IMG_CF_TRUE_COLOR_ALPHA** Like `LV_IMG_CF_TRUE_COLOR` but it also adds an alpha (transparency) byte for every pixel.
- **LV_IMG_CF_TRUE_COLOR_CHROMA_KEYED** Like `LV_IMG_CF_TRUE_COLOR` but if a pixel has `LV_COLOR_TRANSP` (set in *lv_conf.h*) color the pixel will be transparent.
- **LV_IMG_CF_INDEXED_1/2/4/8BIT** Uses a palette with 2, 4, 16 or 256 colors and stores each pixel in 1, 2, 4 or 8 bits.
- **LV_IMG_CF_ALPHA_1/2/4/8BIT Only stores the Alpha value on 1, 2, 4 or 8 bits.** The pixels take the color of `style.image.color` and the set opacity. The source image has to be an alpha channel. This is ideal for bitmaps similar to fonts (where the whole image is one color but you'd like to be able to change it).

The bytes of the LV_IMG_CF_TRUE_COLOR images are stored in the following order.

支持各种内置颜色格式：

- **LV_IMG_CF_TRUE_COLOR** 简单地存储 RGB 颜色（以 LVGL 配置的任何颜色深度）。
- **LV_IMG_CF_TRUE_COLOR_ALPHA** 类似于 LV_IMG_CF_TRUE_COLOR，但它还为每个像素添加了一个 alpha（透明度）字节。
- **LV_IMG_CF_TRUE_COLOR_CHROMA_KEYED** 类似于 LV_IMG_CF_TRUE_COLOR，但如果像素具有 LV_COLOR_TRANSP（在 *lv_conf.h* 中设置）颜色，则像素将是透明的。
- **LV_IMG_CF_INDEXED_1/2/4/8BIT** 使用具有 2、4、16 或 256 种颜色的调色板，并以 1、2、4 或 8 位存储每个像素。
- **LV_IMG_CF_ALPHA_1/2/4/8BIT 仅以 1、2、4 或 8 位存储 Alpha 值**。像素采用 style.image.color 的颜色和设置的不透明度。源图像必须是 alpha 通道。这非常适用于类似于字体的位图（其中整个图像是一种颜色，但您希望能够更改它）。

LV_IMG_CF_TRUE_COLOR 图像的字节按以下顺序存储。

For 32-bit color depth:

- Byte 0: Blue
- Byte 1: Green
- Byte 2: Red
- Byte 3: Alpha

For 16-bit color depth:

- Byte 0: Green 3 lower bit, Blue 5 bit
- Byte 1: Red 5 bit, Green 3 higher bit
- Byte 2: Alpha byte (only with LV_IMG_CF_TRUE_COLOR_ALPHA)

For 8-bit color depth:

- Byte 0: Red 3 bit, Green 3 bit, Blue 2 bit
- Byte 2: Alpha byte (only with LV_IMG_CF_TRUE_COLOR_ALPHA)

对于 32 位色深：

- 字节 0：蓝色
- 字节 1：绿色
- 字节 2：红色
- 字节 3：阿尔法

对于 16 位色深：

- 字节 0：绿色 3 低位，蓝色 5 位

- 字节 1：红色 5 位，绿色 3 高位
- 字节 2：Alpha 字节（仅适用于 LV_IMG_CF_TRUE_COLOR_ALPHA）

对于 8 位色深：

- 字节 0：红色 3 位，绿色 3 位，蓝色 2 位
- 字节 2：Alpha 字节（仅适用于 LV_IMG_CF_TRUE_COLOR_ALPHA）

You can store images in a *Raw* format to indicate that it's not encoded with one of the built-in color formats and an external *Image decoder* needs to be used to decode the image.

- **LV_IMG_CF_RAW** Indicates a basic raw image (e.g. a PNG or JPG image).
- **LV_IMG_CF_RAW_ALPHA** Indicates that the image has alpha and an alpha byte is added for every pixel.
- **LV_IMG_CF_RAW_CHROME_KEYED** Indicates that the image is chroma-keyed as described in `LV_IMG_CF_TRUE_COLOR_CHROMA_KEYED` above.

您可以以 *Raw* 格式存储图像，以表明它没有使用其中一种内置颜色格式进行编码，并且需要使用外部图像解码器 来解码图像。

- **LV_IMG_CF_RAW** 表示基本的原始图像（例如 PNG 或 JPG 图像）。
- **LV_IMG_CF_RAW_ALPHA** 表示图像具有 alpha 并且为每个像素添加一个 alpha 字节。
- **LV_IMG_CF_RAW_CHROME_KEYED** 表示图像是色度键控的，如上面“LV_IMG_CF_TRUE_COLOR_CHROMA_KEYED”中所述。

Add and use images（添加和使用图像）

You can add images to LVGL in two ways:

- using the online converter
- manually create images

您可以通过两种方式将图像添加到 LVGL：

- 使用在线转换器
- 手动创建图像

Online converter（在线转换器）

The online Image converter is available here: https://lvgl.io/tools/imageconverter

Adding an image to LVGL via online converter is easy.

1. You need to select a *BMP*, *PNG* or *JPG* image first.
2. Give the image a name that will be used within LVGL.

3. Select the *Color format*.
4. Select the type of image you want. Choosing a binary will generate a `.bin` file that must be stored separately and read using the *file support*. Choosing a variable will generate a standard C file that can be linked into your project.
5. Hit the *Convert* button. Once the conversion is finished, your browser will automatically download the resulting file.

在线图像转换器可在此处获得：https://lvgl.io/tools/imageconverter

通过在线转换器将图像添加到 LVGL 很容易。

1. 您需要先选择 *BMP*、*PNG* 或 *JPG* 图像。
2. 为图像指定一个将在 LVGL 中使用的名称。
3. 选择颜色格式。
4. 选择您想要的图像类型。选择二进制文件将生成一个 `.bin` 文件，该文件必须单独存储并使用文件支持 读取。选择一个变量将生成一个可以链接到您的项目的标准 C 文件。
5. 点击转换按钮。转换完成后，您的浏览器将自动下载生成的文件。

In the converter C arrays (variables), the bitmaps for all the color depths (1, 8, 16 or 32) are included in the C file, but only the color depth that matches `LV_COLOR_DEPTH` in *lv_conf.h* will actually be linked into the resulting executable.

In case of binary files, you need to specify the color format you want:

- RGB332 for 8-bit color depth
- RGB565 for 16-bit color depth
- RGB565 Swap for 16-bit color depth (two bytes are swapped)
- RGB888 for 32-bit color depth

在转换器 C 数组（变量）中，所有颜色深度（1、8、16 或 32）的位图都包含在 C 文件中，但实际上只有与 *lv_conf.h* 中的 `LV_COLOR_DEPTH` 匹配的颜色深度才会链接到生成的可执行文件中。

对于二进制文件，您需要指定所需的颜色格式：

- RGB332 8 位色深
- RGB565 16 位色深
- RGB565 交换 16 位色深（交换两个字节）
- RGB888 用于 32 位色深

Manually create an image（手动创建图像）

If you are generating an image at run-time, you can craft an image variable to display it using LVGL. For example:

如果您在运行时生成图像，您可以制作一个图像变量以使用 LVGL 显示它。例如：

```
uint8_t my_img_data[] = {0x00, 0x01, 0x02, ...};

static lv_img_dsc_t my_img_dsc = {
    .header.always_zero = 0,
    .header.w = 80,
    .header.h = 60,
    .data_size = 80 * 60 * LV_COLOR_DEPTH / 8,
    .header.cf = LV_IMG_CF_TRUE_COLOR,          /*Set the color format*/
    .data = my_img_data,
};
```

If the color format is `LV_IMG_CF_TRUE_COLOR_ALPHA` you can set `data_size` like `80 * 60 * LV_IMG_PX_SIZE_ALPHA_BYTE`.

Another (possibly simpler) option to create and display an image at run-time is to use the *Canvas* object.

如果颜色格式是 `LV_IMG_CF_TRUE_COLOR_ALPHA`，你可以将 `data_size` 设置为 `80 * 60 * LV_IMG_PX_SIZE_ALPHA_BYTE`。

在运行时创建和显示图像的另一个（可能更简单）选项是使用*Canvas* 对象。

Use images（使用图片）

The simplest way to use an image in LVGL is to display it with an *lv_img* object:

在 LVGL 中使用图像的最简单方法是使用*lv_img* 对象显示它：

```
lv_obj_t * icon = lv_img_create(lv_scr_act(), NULL);

/*From variable*/
lv_img_set_src(icon, &my_icon_dsc);

/*From file*/
lv_img_set_src(icon, "S:my_icon.bin");
```

If the image was converted with the online converter, you should use `LV_IMG_DECLARE(my_icon_dsc)` to declare the image in the file where you want to use it.

如果图像是使用在线转换器转换的，则应使用 `LV_IMG_DECLARE(my_icon_dsc)` 在要使用的文件中声明图像。

Image decoder（图像解码器）

As you can see in the *Color formats* section, LVGL supports several built-in image formats. In many cases, these will be all you need. LVGL doesn't directly support, however, generic image formats like PNG or JPG.

To handle non-built-in image formats, you need to use external libraries and attach them to LVGL via the *Image decoder* interface.

The image decoder consists of 4 callbacks:

- **info** get some basic info about the image (width, height and color format).
- **open** open the image: either store the decoded image or set it to `NULL` to indicate the image can be read line-by-line.
- **read** if *open* didn't fully open the image this function should give some decoded data (max 1 line) from a given position.
- **close** close the opened image, free the allocated resources.

You can add any number of image decoders. When an image needs to be drawn, the library will try all the registered image decoders until it finds one which can open the image, i.e. one which knows that format.

The `LV_IMG_CF_TRUE_COLOR_...`, `LV_IMG_INDEXED_...` and `LV_IMG_ALPHA_...` formats (essentially, all non-`RAW` formats) are understood by the built-in decoder.

正如您在颜色格式 部分中看到的，LVGL 支持多种内置图像格式。在许多情况下，这些将是您所需要的。但是，LVGL 不直接支持 PNG 或 JPG 等通用图像格式。

要处理非内置图像格式，您需要使用外部库并通过图像解码器接口将它们附加到 LVGL。

图像解码器由 4 个回调组成：

- **info** 获取有关图像的一些基本信息（宽度、高度和颜色格式）。
- **open** 打开图像：要么存储解码后的图像，要么将其设置为 `NULL` 以指示可以逐行读取图像。
- **read** 如果 *open* 没有完全打开图像，这个函数应该从给定的位置给出一些解码数据（最多 1 行）。
- **close** 关闭打开的图片，释放分配的资源。

您可以添加任意数量的图像解码器。当需要绘制图像时，库将尝试所有注册的图像解码器，直到找到可以打开图像的解码器，即知道该格式的解码器。

`LV_IMG_CF_TRUE_COLOR_...`、`LV_IMG_INDEXED_...` 和 `LV_IMG_ALPHA_...` 格式（基本上，所有非 `RAW` 格式）被内置解码器理解。

Custom image formats（自定义图像格式）

The easiest way to create a custom image is to use the online image converter and set `Raw`, `Raw with alpha` or `Raw with chroma-keyed` format. It will just take every byte of the binary file you uploaded and write it as the image "bitmap". You then need to attach an image decoder that will parse that bitmap and generate the real, renderable bitmap.

`header.cf` will be `LV_IMG_CF_RAW`, `LV_IMG_CF_RAW_ALPHA` or `LV_IMG_CF_RAW_CHROME_KEYED` accordingly. You should choose the correct format according to your needs: fully opaque image, use alpha channel or use chroma keying.

After decoding, the *raw* formats are considered *True color* by the library. In other words, the image decoder must decode the *Raw* images to *True color* according to the format described in [#color-formats](Color formats) section.

创建自定义图像的最简单方法是使用在线图像转换器并设置“Raw”、“Raw with alpha”或“Raw with chroma-keyed”格式。它只会获取您上传的二进制文件的每个字节并将其写入图像“位图”。然后，您需要附加一个图像解码器，该解码器将解析该位图并生成真实的、可渲染的位图。

`header.cf` 将相应地为 `LV_IMG_CF_RAW`、`LV_IMG_CF_RAW_ALPHA` 或 `LV_IMG_CF_RAW_CHROME_KEYED`。您应该根据需要选择正确的格式：完全不透明的图像、使用 alpha 通道或使用色度键控。

解码后，*raw* 格式被库视为*真彩色*。换句话说，图像解码器必须根据 [#color-formats]（颜色格式）部分中描述的格式将 *Raw* 图像解码为 *True color*。

If you want to create a custom image, you should use `LV_IMG_CF_USER_ENCODED_0..7` color formats. However, the library can draw the images only in *True color* format (or *Raw* but finally it's supposed to be in *True color* format). The `LV_IMG_CF_USER_ENCODED_...` formats are not known by the library and therefore they should be decoded to one of the known formats from [#color-formats](Color formats) section. It's possible to decode the image to a non-true color format first (for example: `LV_IMG_INDEXED_4BITS`) and then call the built-in decoder functions to convert it to *True color*.

With *User encoded* formats, the color format in the open function (`dsc->header.cf`) should be changed according to the new format.

如果要创建自定义图像，则应使用 `LV_IMG_CF_USER_ENCODED_0..7` 颜色格式。但是，该库只能以 *True color* 格式（或 *Raw* 但最终它应该以 *True color* 格式）绘制图像。

lvgl 库不知道 `LV_IMG_CF_USER_ENCODED_...` 格式，因此应该将它们解码为 [#color-formats]（颜色格式）部分中的已知格式之一。可以先将图像解码为非真彩色格式（例如：`LV_IMG_INDEXED_4BITS`），然后调用内置解码器函数将其转换为*真彩色*。

使用*用户编码*格式，打开函数（`dsc->header.cf`）中的颜色格式应根据新格式进行更改。

Register an image decoder（注册图像解码器）

Here's an example of getting LVGL to work with PNG images.

First, you need to create a new image decoder and set some functions to open/close the PNG files. It should looks like this:

这是让 LVGL 处理 PNG 图像的示例。

首先，您需要创建一个新的图像解码器并设置一些功能来打开/关闭 PNG 文件。它应该是这样的：

```
/*Create a new decoder and register functions */
lv_img_decoder_t * dec = lv_img_decoder_create();
lv_img_decoder_set_info_cb(dec, decoder_info);
lv_img_decoder_set_open_cb(dec, decoder_open);
lv_img_decoder_set_close_cb(dec, decoder_close);


/**
 * Get info about a PNG image
 * @param decoder pointer to the decoder where this function belongs
 * @param src can be file name or pointer to a C array
 * @param header store the info here
 * @return LV_RES_OK: no error; LV_RES_INV: can't get the info
 */
static lv_res_t decoder_info(lv_img_decoder_t * decoder, const void * src, lv_img_
↪header_t * header)
{
  /*Check whether the type `src` is known by the decoder*/
  if(is_png(src) == false) return LV_RES_INV;

  /* Read the PNG header and find `width` and `height` */
  ...

  header->cf = LV_IMG_CF_RAW_ALPHA;
  header->w = width;
  header->h = height;
}

/**
 * Open a PNG image and return the decided image
 * @param decoder pointer to the decoder where this function belongs
 * @param dsc pointer to a descriptor which describes this decoding session
 * @return LV_RES_OK: no error; LV_RES_INV: can't get the info
 */
static lv_res_t decoder_open(lv_img_decoder_t * decoder, lv_img_decoder_dsc_t * dsc)
```

(下页继续)

(续上页)

```
{

  /*Check whether the type `src` is known by the decoder*/
  if(is_png(src) == false) return LV_RES_INV;

  /*Decode and store the image. If `dsc->img_data` is `NULL`, the `read_line`␣
↪function will be called to get the image data line-by-line*/
  dsc->img_data = my_png_decoder(src);

  /*Change the color format if required. For PNG usually 'Raw' is fine*/
  dsc->header.cf = LV_IMG_CF_...

  /*Call a built in decoder function if required. It's not required if`my_png_
↪decoder` opened the image in true color format.*/
  lv_res_t res = lv_img_decoder_built_in_open(decoder, dsc);

  return res;
}

/**
 * Decode `len` pixels starting from the given `x`, `y` coordinates and store them in␣
↪`buf`.
 * Required only if the "open" function can't open the whole decoded pixel array.␣
↪(dsc->img_data == NULL)
 * @param decoder pointer to the decoder the function associated with
 * @param dsc pointer to decoder descriptor
 * @param x start x coordinate
 * @param y start y coordinate
 * @param len number of pixels to decode
 * @param buf a buffer to store the decoded pixels
 * @return LV_RES_OK: ok; LV_RES_INV: failed
 */
lv_res_t decoder_built_in_read_line(lv_img_decoder_t * decoder, lv_img_decoder_dsc_t␣
↪* dsc, lv_coord_t x,
                                                lv_coord_t y, lv_coord_t len, uint8_
↪t * buf)
{
   /*With PNG it's usually not required*/

   /*Copy `len` pixels from `x` and `y` coordinates in True color format to `buf` */

}
```

(下页继续)

(续上页)

```
/**
 * Free the allocated resources
 * @param decoder pointer to the decoder where this function belongs
 * @param dsc pointer to a descriptor which describes this decoding session
 */
static void decoder_close(lv_img_decoder_t * decoder, lv_img_decoder_dsc_t * dsc)
{
  /*Free all allocated data*/

  /*Call the built-in close function if the built-in open/read_line was used*/
  lv_img_decoder_built_in_close(decoder, dsc);

}
```

So in summary:

- In `decoder_info`, you should collect some basic information about the image and store it in `header`.
- In `decoder_open`, you should try to open the image source pointed by `dsc->src`. Its type is already in `dsc->src_type == LV_IMG_SRC_FILE/VARIABLE`. If this format/type is not supported by the decoder, return `LV_RES_INV`. However, if you can open the image, a pointer to the decoded *True color* image should be set in `dsc->img_data`. If the format is known but you don't want to decode the entire image (e.g. no memory for it) set `dsc->img_data = NULL` to call `read_line` to get the pixels.
- In `decoder_close` you should free all the allocated resources.
- `decoder_read` is optional. Decoding the whole image requires extra memory and some computational overhead. However, if can decode one line of the image without decoding the whole image, you can save memory and time. To indicate that the *line read* function should be used, set `dsc->img_data = NULL` in the open function.

所以总结一下：

- 在 `decoder_info` 中，你应该收集一些关于图像的基本信息并将其存储在 `header` 中。
- 在 `decoder_open` 中，你应该尝试打开 `dsc->src` 指向的图像源。它的类型已经在 `dsc->src_type == LV_IMG_SRC_FILE/VARIABLE` 中。如果解码器不支持此格式/类型，则返回 "LV_RES_INV"。但是，如果您可以打开图像，则应在 `dsc->img_data` 中设置指向解码的真彩色图像的指针。如果格式已知但您不想解码整个图像（例如没有内存），请设置 `dsc->img_data = NULL` 以调用 `read_line` 来获取像素。
- 在 `decoder_close` 中，你应该释放所有分配的资源。
- `decoder_read` 是可选的。解码整个图像需要额外的内存和一些计算开销。但是，如果可以解码一行图像而不解码整个图像，则可以节省内存和时间。表示 *line read* 函数应该是我们

Manually use an image decoder（手动使用图像解码器）

LVGL will use the registered image decoder automatically if you try and draw a raw image (i.e. using the `lv_img` object) but you can use them manually too. Create a `lv_img_decoder_dsc_t` variable to describe the decoding session and call `lv_img_decoder_open()`.

如果您尝试绘制原始图像（即使用 `lv_img` 对象），LVGL 将自动使用注册的图像解码器，但您也可以手动使用它们。创建一个 `lv_img_decoder_dsc_t` 变量来描述解码会话并调用 `lv_img_decoder_open()`。

```
lv_res_t res;
lv_img_decoder_dsc_t dsc;
res = lv_img_decoder_open(&dsc, &my_img_dsc, LV_COLOR_WHITE);

if(res == LV_RES_OK) {
  /*Do something with `dsc->img_data`*/
  lv_img_decoder_close(&dsc);
}
```

Image caching（图片缓存）

Sometimes it takes a lot of time to open an image. Continuously decoding a PNG image or loading images from a slow external memory would be inefficient and detrimental to the user experience.

Therefore, LVGL caches a given number of images. Caching means some images will be left open, hence LVGL can quickly access them from `dsc->img_data` instead of needing to decode them again.

Of course, caching images is resource-intensive as it uses more RAM (to store the decoded image). LVGL tries to optimize the process as much as possible (see below), but you will still need to evaluate if this would be beneficial for your platform or not. If you have a deeply embedded target which decodes small images from a relatively fast storage medium, image caching may not be worth it.

有时打开图像需要很多时间。连续解码 PNG 图像或从缓慢的外部存储器加载图像将是低效的并且不利于用户体验。

因此，LVGL 缓存给定数量的图像。缓存意味着一些图像将保持打开状态，因此 LVGL 可以从 `dsc->img_data` 快速访问它们，而无需再次解码它们。

当然，缓存图像是资源密集型的，因为它使用更多的 RAM（用于存储解码的图像）。LVGL 尝试尽可能地优化流程（见下文），但您仍需要评估这是否对您的平台有益。如果您有一个深度嵌入的目标，可以从相对较快的存储介质中解码小图像，则图像缓存可能不值得。

Cache size（缓存大小）

The number of cache entries can be defined in `LV_IMG_CACHE_DEF_SIZE` in *lv_conf.h*. The default value is 1 so only the most recently used image will be left open.

The size of the cache can be changed at run-time with `lv_img_cache_set_size(entry_num)`.

缓存条目的数量可以在 *lv_conf.h* 中的 `LV_IMG_CACHE_DEF_SIZE` 中定义。默认值为 1，因此只有最近使用的图像将保持打开状态。

缓存的大小可以在运行时通过 `lv_img_cache_set_size(entry_num)` 改变。

Value of images（图片的价值）

When you use more images than cache entries, LVGL can't cache all of the images. Instead, the library will close one of the cached images (to free space).

To decide which image to close, LVGL uses a measurement it previously made of how long it took to open the image. Cache entries that hold slower-to-open images are considered more valuable and are kept in the cache as long as possible.

If you want or need to override LVGL's measurement, you can manually set the *time to open* value in the decoder open function in `dsc->time_to_open = time_ms` to give a higher or lower value. (Leave it unchanged to let LVGL set it.)

Every cache entry has a *"life"* value. Every time an image opening happens through the cache, the *life* value of all entries is decreased to make them older. When a cached image is used, its *life* value is increased by the *time to open* value to make it more alive.

If there is no more space in the cache, the entry with the smallest life value will be closed.

当您使用的图像多于缓存条目时，LVGL 无法缓存所有图像。相反，库将关闭缓存的图像之一（以释放空间）。

为了决定关闭哪个图像，LVGL 使用它之前对打开图像所花费的时间进行的测量。保存打开速度较慢的图像的缓存条目被认为更有价值，并尽可能长时间地保存在缓存中。

如果您想要或需要覆盖 LVGL 的测量，您可以在 `dsc->time_to_open = time_ms` 中的解码器打开函数中手动设置 *time to open* 值，以给出更高或更低的值。（保持不变，让 LVGL 设置它。）

每个缓存条目都有一个 *"life"* 值。每次通过缓存打开图像时，所有条目的 *life* 值都会减少以使其更旧。当使用缓存图像时，它的 *life* 值会增加 *time to open* 值以使其更加活跃。

如果缓存中没有更多空间，则生命值最小的条目将被关闭。

Memory usage（内存使用情况）

Note that the cached image might continuously consume memory. For example, if 3 PNG images are cached, they will consume memory while they are open.

Therefore, it's the user's responsibility to be sure there is enough RAM to cache even the largest images at the same time.

请注意，缓存的图像可能会持续消耗内存。例如，如果缓存了 3 个 PNG 图片，它们将在打开时消耗内存。

因此，用户有责任确保有足够的 RAM 来同时缓存最大的图像。

Clean the cache（清理缓存）

Let's say you have loaded a PNG image into a `lv_img_dsc_t my_png` variable and use it in an `lv_img` object. If the image is already cached and you then change the underlying PNG file, you need to notify LVGL to cache the image again. Otherwise, there is no easy way of detecting that the underlying file changed and LVGL will still draw the old image.

To do this, use `lv_img_cache_invalidate_src(&my_png)`. If `NULL` is passed as a parameter, the whole cache will be cleaned.

假设您已将 PNG 图像加载到 `lv_img_dsc_t my_png` 变量中，并在 `lv_img` 对象中使用它。如果图像已经缓存，然后您更改了底层 PNG 文件，则需要通知 LVGL 再次缓存图像。否则，没有简单的方法可以检测到底层文件发生了变化，而 LVGL 仍会绘制旧图像。

为此，请使用 `lv_img_cache_invalidate_src(&my_png)`。如果将 `NULL` 作为参数传递，则整个缓存将被清除。

API

Image buffer（图像缓冲区）

Typedefs

typedef uint8_t **`lv_img_cf_t`**

Enums

enum **`[anonymous]`**
Values:

enumerator **`LV_IMG_CF_UNKNOWN`**

enumerator **LV_IMG_CF_RAW**
Contains the file as it is. Needs custom decoder function

enumerator **LV_IMG_CF_RAW_ALPHA**
Contains the file as it is. The image has alpha. Needs custom decoder function

enumerator **LV_IMG_CF_RAW_CHROMA_KEYED**
Contains the file as it is. The image is chroma keyed. Needs custom decoder function

enumerator **LV_IMG_CF_TRUE_COLOR**
Color format and depth should match with LV_COLOR settings

enumerator **LV_IMG_CF_TRUE_COLOR_ALPHA**
Same as `LV_IMG_CF_TRUE_COLOR` but every pixel has an alpha byte

enumerator **LV_IMG_CF_TRUE_COLOR_CHROMA_KEYED**
Same as `LV_IMG_CF_TRUE_COLOR` but LV_COLOR_TRANSP pixels will be transparent

enumerator **LV_IMG_CF_INDEXED_1BIT**
Can have 2 different colors in a palette (always chroma keyed)

enumerator **LV_IMG_CF_INDEXED_2BIT**
Can have 4 different colors in a palette (always chroma keyed)

enumerator **LV_IMG_CF_INDEXED_4BIT**
Can have 16 different colors in a palette (always chroma keyed)

enumerator **LV_IMG_CF_INDEXED_8BIT**
Can have 256 different colors in a palette (always chroma keyed)

enumerator **LV_IMG_CF_ALPHA_1BIT**
Can have one color and it can be drawn or not

enumerator **LV_IMG_CF_ALPHA_2BIT**
Can have one color but 4 different alpha value

enumerator **LV_IMG_CF_ALPHA_4BIT**
Can have one color but 16 different alpha value

enumerator **LV_IMG_CF_ALPHA_8BIT**
Can have one color but 256 different alpha value

enumerator **LV_IMG_CF_RESERVED_15**
Reserved for further use.

enumerator **LV_IMG_CF_RESERVED_16**
Reserved for further use.

enumerator **LV_IMG_CF_RESERVED_17**
Reserved for further use.

enumerator **LV_IMG_CF_RESERVED_18**
Reserved for further use.

enumerator **LV_IMG_CF_RESERVED_19**
Reserved for further use.

enumerator **LV_IMG_CF_RESERVED_20**
Reserved for further use.

enumerator **LV_IMG_CF_RESERVED_21**
Reserved for further use.

enumerator **LV_IMG_CF_RESERVED_22**
Reserved for further use.

enumerator **LV_IMG_CF_RESERVED_23**
Reserved for further use.

enumerator **LV_IMG_CF_USER_ENCODED_0**
User holder encoding format.

enumerator **LV_IMG_CF_USER_ENCODED_1**
User holder encoding format.

enumerator **LV_IMG_CF_USER_ENCODED_2**
User holder encoding format.

enumerator **LV_IMG_CF_USER_ENCODED_3**
User holder encoding format.

enumerator **LV_IMG_CF_USER_ENCODED_4**
User holder encoding format.

enumerator **LV_IMG_CF_USER_ENCODED_5**
User holder encoding format.

enumerator **LV_IMG_CF_USER_ENCODED_6**
User holder encoding format.

enumerator **LV_IMG_CF_USER_ENCODED_7**
User holder encoding format.

Functions

lv_img_dsc_t ***lv_img_buf_alloc**(lv_coord_t w, lv_coord_t h, *lv_img_cf_t* cf)
Allocate an image buffer in RAM

参数

- **w** -- width of image
- **h** -- height of image
- **cf** -- a color format (`LV_IMG_CF_...`)

返回 an allocated image, or NULL on failure

lv_color_t **lv_img_buf_get_px_color**(*lv_img_dsc_t* *dsc, lv_coord_t x, lv_coord_t y, lv_color_t color)
Get the color of an image's pixel

参数

- **dsc** -- an image descriptor
- **x** -- x coordinate of the point to get
- **y** -- x coordinate of the point to get
- **color** -- the color of the image. In case of `LV_IMG_CF_ALPHA_1/2/4/8` this color is used. Not used in other cases.
- **safe** -- true: check out of bounds

返回 color of the point

lv_opa_t **lv_img_buf_get_px_alpha**(*lv_img_dsc_t* *dsc, lv_coord_t x, lv_coord_t y)
Get the alpha value of an image's pixel

参数

- **dsc** -- pointer to an image descriptor
- **x** -- x coordinate of the point to set
- **y** -- x coordinate of the point to set

- **safe** -- true: check out of bounds

返回 alpha value of the point

void **lv_img_buf_set_px_color**(*lv_img_dsc_t* *dsc, lv_coord_t x, lv_coord_t y, lv_color_t c)

Set the color of a pixel of an image. The alpha channel won't be affected.

参数

- **dsc** -- pointer to an image descriptor
- **x** -- x coordinate of the point to set
- **y** -- x coordinate of the point to set
- **c** -- color of the point
- **safe** -- true: check out of bounds

void **lv_img_buf_set_px_alpha**(*lv_img_dsc_t* *dsc, lv_coord_t x, lv_coord_t y, lv_opa_t opa)

Set the alpha value of a pixel of an image. The color won't be affected

参数

- **dsc** -- pointer to an image descriptor
- **x** -- x coordinate of the point to set
- **y** -- x coordinate of the point to set
- **opa** -- the desired opacity
- **safe** -- true: check out of bounds

void **lv_img_buf_set_palette**(*lv_img_dsc_t* *dsc, uint8_t id, lv_color_t c)

Set the palette color of an indexed image. Valid only for `LV_IMG_CF_INDEXED1/2/4/8`

参数

- **dsc** -- pointer to an image descriptor
- **id** -- the palette color to set:
  - for `LV_IMG_CF_INDEXED1`: 0..1
  - for `LV_IMG_CF_INDEXED2`: 0..3
  - for `LV_IMG_CF_INDEXED4`: 0..15
  - for `LV_IMG_CF_INDEXED8`: 0..255
- **c** -- the color to set

void **lv_img_buf_free**(*lv_img_dsc_t* *dsc)

Free an allocated image buffer

参数 **dsc** -- image buffer to free

uint32_t **lv_img_buf_get_img_size**(lv_coord_t w, lv_coord_t h, *lv_img_cf_t* cf)

Get the memory consumption of a raw bitmap, given color format and dimensions.

参数

- **w** -- width
- **h** -- height
- **cf** -- color format

返回 size in bytes

void **_lv_img_buf_transform_init**(*lv_img_transform_dsc_t* *dsc)

Initialize a descriptor to rotate an image

参数 **dsc** -- pointer to an *lv_img_transform_dsc_t* variable whose `cfg` field is initialized

bool **_lv_img_buf_transform_anti_alias**(*lv_img_transform_dsc_t* *dsc)

Continue transformation by taking the neighbors into account

参数 **dsc** -- pointer to the transformation descriptor

bool **_lv_img_buf_transform**(*lv_img_transform_dsc_t* *dsc, lv_coord_t x, lv_coord_t y)

Get which color and opa would come to a pixel if it were rotated

---

注解: the result is written back to `dsc->res_color` and `dsc->res_opa`

参数

- **dsc** -- a descriptor initialized by `lv_img_buf_rotate_init`
- **x** -- the coordinate which color and opa should be get
- **y** -- the coordinate which color and opa should be get

返回 true: there is valid pixel on these x/y coordinates; false: the rotated pixel was out of the image

void **_lv_img_buf_get_transformed_area**(lv_area_t *res, lv_coord_t w, lv_coord_t h, int16_t angle, uint16_t zoom, const lv_point_t *pivot)

Get the area of a rectangle if its rotated and scaled

参数

- **res** -- store the coordinates here
- **w** -- width of the rectangle to transform
- **h** -- height of the rectangle to transform
- **angle** -- angle of rotation
- **zoom** -- zoom, (256 no zoom)

- **pivot** -- x,y pivot coordinates of rotation

struct **lv_img_header_t**

#include <lv_img_buf.h> The first 8 bit is very important to distinguish the different source types. For more info see `lv_img_get_src_type()` in lv_img.c On big endian systems the order is reversed so cf and always_zero must be at the end of the struct.

Public Members

uint32_t **h**

uint32_t **w**

uint32_t **reserved**

uint32_t **always_zero**

uint32_t **cf**

struct **lv_img_dsc_t**

#include <lv_img_buf.h> Image header it is compatible with the result from image converter utility

Public Members

lv_img_header_t **header**

A header describing the basics of the image

uint32_t **data_size**

Size of the image in bytes

const uint8_t ***data**

Pointer to the data of the image

struct **lv_img_transform_dsc_t**

Public Members

const void ***src**

lv_coord_t **src_w**

lv_coord_t **src_h**

lv_coord_t **pivot_x**

lv_coord_t **pivot_y**

int16_t **angle**

uint16_t **zoom**

lv_color_t **color**

lv_img_cf_t **cf**

bool **antialias**

struct *lv_img_transform_dsc_t*::[anonymous] **cfg**

lv_opa_t **opa**

struct *lv_img_transform_dsc_t*::[anonymous] **res**

lv_img_dsc_t **img_dsc**

int32_t **pivot_x_256**

int32_t **pivot_y_256**

int32_t **sinma**

int32_t **cosma**

uint8_t **chroma_keyed**

uint8_t **has_alpha**

uint8_t **native_color**

uint32_t **zoom_inv**

lv_coord_t **xs**

lv_coord_t **ys**

lv_coord_t **xs_int**

lv_coord_t **ys_int**

uint32_t **pxi**

uint8_t **px_size**

struct *lv_img_transform_dsc_t*::[anonymous] **tmp**

2.5.13 File system（文件系统）

LVGL has a 'File system' abstraction module that enables you to attach any type of file system. The file system is identified by a drive letter. For example, if the SD card is associated with the letter `'S'`, a file can be reached like `"S:path/to/file.txt"`.

LVGL 有一个“文件系统”抽象模块，使您能够附加任何类型的文件系统。文件系统由驱动器号标识。例如，如果 SD 卡与字母“S”相关联，则可以访问类似“S:path/to/file.txt”的文件。

Ready to use drivers（准备使用驱动程序）

The lv_fs_if repository contains ready to use drivers using POSIX, standard C and FATFS API. See it's README for the details.

lv_fs_if 存储库包含使用 POSIX、标准 C 和 FATFS 的即用型驱动程序) API。有关详细信息，请参阅 README。

Add a driver（添加驱动程序）

Registering a driver（注册驱动）

To add a driver, `lv_fs_drv_t` needs to be initialized like below. `lv_fs_drv_t` needs to be static, global or dynamically allocated and not a local varaible.

要添加驱动程序，`lv_fs_drv_t` 需要像下面这样初始化。`lv_fs_drv_t` 需要是静态的、全局的或动态分配的，而不是局部变量。

```
static lv_fs_drv_t drv;                         /*Needs to be static or global*/
lv_fs_drv_init(&drv);                           /*Basic initialization*/

drv.letter = 'S';                               /*An uppercase letter to identify the drive␣
↪*/
drv.ready_cb = my_ready_cb;                     /*Callback to tell if the drive is ready to␣
↪use */
drv.open_cb = my_open_cb;                       /*Callback to open a file */
drv.close_cb = my_close_cb;                     /*Callback to close a file */
drv.read_cb = my_read_cb;                       /*Callback to read a file */
drv.write_cb = my_write_cb;                     /*Callback to write a file */
drv.seek_cb = my_seek_cb;                       /*Callback to seek in a file (Move cursor)␣
↪*/
drv.tell_cb = my_tell_cb;                       /*Callback to tell the cursor position  */

drv.dir_open_cb = my_dir_open_cb;               /*Callback to open directory to read its␣
↪content */
drv.dir_read_cb = my_dir_read_cb;               /*Callback to read a directory's content */
```

(下页继续)

(续上页)

```
drv.dir_close_cb = my_dir_close_cb;       /*Callback to close a directory */

drv.user_data = my_user_data;             /*Any custom data if required*/

lv_fs_drv_register(&drv);                 /*Finally register the drive*/
```

Any of the callbacks can be `NULL` to indicate that operation is not supported.

任何回调都可以为“NULL”以指示不支持该操作。

Implementing the callbacks（实现回调）

Open callback（打开回调）

The prototype of `open_cb` looks like this:

`open_cb` 的原型如下所示：

```
void * (*open_cb)(lv_fs_drv_t * drv, const char * path, lv_fs_mode_t mode);
```

`path` is path after the driver letter (e.g. ”S:path/to/file.txt” -> ”path/to/file.txt”). `mode` can be `LV_FS_MODE_WR` or `LV_FS_MODE_RD` to open for write or read.

The return value is a pointer the *file object* the describes the opened file or `NULL` if there were any issues (e.g. the file wasn't found). The returned file object will be passed to to other file system related callbacks. (see below)

`path` 是驱动程序字母后的路径（例如 “S:path/to/file.txt” -> “path/to/file.txt”）。`mode` 可以是 `LV_FS_MODE_WR` 或 `LV_FS_MODE_RD` 来打开写入或读取。

返回值是 *file object* 的指针，它描述了打开的文件，如果有任何问题（例如找不到文件），则返回 “NULL”。返回的文件对象将传递给其他与文件系统相关的回调。（见下文）

Other callbacks（其他回调）

The other callbacks are quite similar. For example `write_cb` looks like this:

其他回调非常相似。例如 `write_cb` 看起来像这样：

```
lv_fs_res_t (*write_cb)(lv_fs_drv_t * drv, void * file_p, const void * buf, uint32_t␣
↪btw, uint32_t * bw);
```

As `file_p` LVGL passes the return value of `open_cb`, `buf` is the data to write, `btw` is the Bytes To Write, `bw` is the actually written bytes.

For a template to the callbacks see lv_fs_template.c.

由于 `file_p` LVGL 传递 `open_cb` 的返回值，`buf` 是要写入的数据，`btw` 是要写入的字节数，`bw` 是实际写入的字节数。

有关回调的模板，请参阅 lv_fs_template.c。

Usage example（使用示例）

The example below shows how to read from a file:

下面的示例演示如何从文件中读取：

```
lv_fs_file_t f;
lv_fs_res_t res;
res = lv_fs_open(&f, "S:folder/file.txt", LV_FS_MODE_RD);
if(res != LV_FS_RES_OK) my_error_handling();

uint32_t read_num;
uint8_t buf[8];
res = lv_fs_read(&f, buf, 8, &read_num);
if(res != LV_FS_RES_OK || read_num != 8) my_error_handling();

lv_fs_close(&f);
```

The mode in `lv_fs_open` can be `LV_FS_MODE_WR` to open for write or `LV_FS_MODE_RD | LV_FS_MODE_WR` for both

This example shows how to read a directory's content. It's up to the driver how to mark the directories, but it can be a good practice to insert a `'/'` in front of the directory name.

`lv_fs_open` 中的模式可以是 `LV_FS_MODE_WR` to open for write 或 `LV_FS_MODE_RD | LV_FS_MODE_WR` 为两者

此示例演示如何读取目录的内容。如何标记目录取决于驱动程序，但在目录名称前插入“/”是一个好习惯。

```
lv_fs_dir_t dir;
lv_fs_res_t res;
res = lv_fs_dir_open(&dir, "S:/folder");
if(res != LV_FS_RES_OK) my_error_handling();

char fn[256];
while(1) {
    res = lv_fs_dir_read(&dir, fn);
    if(res != LV_FS_RES_OK) {
        my_error_handling();
        break;
    }
```

(下页继续)

(续上页)

```
    /*fn is empty, if not more files to read*/
    if(strlen(fn) == 0) {
        break;
    }

    printf("%s\n", fn);
}

lv_fs_dir_close(&dir);
```

Use drivers for images（使用图像驱动程序）

Image objects can be opened from files too (besides variables stored in the flash).

To use files in image widgets the following callbacks are required:

- open
- close
- read
- seek
- tell

Image 对象也可以从文件中打开（除了存储在闪存中的变量）。

要在图像小部件中使用文件，需要以下回调：

- 打开
- 关
- 读
- 寻找
- 告诉

API

Typedefs

typedef uint8_t **lv_fs_res_t**

typedef uint8_t **lv_fs_mode_t**

typedef struct *_lv_fs_drv_t* **lv_fs_drv_t**

Enums

enum **[anonymous]**
Errors in the file system module.

Values:

enumerator **LV_FS_RES_OK**

enumerator **LV_FS_RES_HW_ERR**

enumerator **LV_FS_RES_FS_ERR**

enumerator **LV_FS_RES_NOT_EX**

enumerator **LV_FS_RES_FULL**

enumerator **LV_FS_RES_LOCKED**

enumerator **LV_FS_RES_DENIED**

enumerator **LV_FS_RES_BUSY**

enumerator **LV_FS_RES_TOUT**

enumerator **LV_FS_RES_NOT_IMP**

enumerator **LV_FS_RES_OUT_OF_MEM**

enumerator **LV_FS_RES_INV_PARAM**

enumerator **LV_FS_RES_UNKNOWN**

enum **[anonymous]**
File open mode.

Values:

enumerator **LV_FS_MODE_WR**

enumerator **LV_FS_MODE_RD**

enum **lv_fs_whence_t**
Seek modes.

Values:

enumerator **LV_FS_SEEK_SET**
Set the position from absolutely (from the start of file)

enumerator **LV_FS_SEEK_CUR**
Set the position from the current position

enumerator **LV_FS_SEEK_END**
Set the position from the end of the file

Functions

void **_lv_fs_init**(void)
Initialize the File system interface

void **lv_fs_drv_init**(*lv_fs_drv_t* *drv)
Initialize a file system driver with default values. It is used to surly have known values in the fields ant not memory junk. After it you can set the fields.

参数 **drv** -- pointer to driver variable to initialize

void **lv_fs_drv_register**(*lv_fs_drv_t* *drv)
Add a new drive

参数 **drv** -- pointer to an lv_fs_drv_t structure which is inited with the corresponding function pointers. Only pointer is saved, so the driver should be static or dynamically allocated.

lv_fs_drv_t ***lv_fs_get_drv**(char letter)
Give a pointer to a driver from its letter

参数 **letter** -- the driver letter

返回 pointer to a driver or NULL if not found

bool **lv_fs_is_ready**(char letter)
Test if a drive is ready or not. If the `ready` function was not initialized `true` will be returned.

参数 **letter** -- letter of the drive

返回 true: drive is ready; false: drive is not ready

lv_fs_res_t **lv_fs_open**(*lv_fs_file_t* *file_p, const char *path, *lv_fs_mode_t* mode)
Open a file

参数

- **file_p** -- pointer to a *lv_fs_file_t* variable
- **path** -- path to the file beginning with the driver letter (e.g. S:/folder/file.txt)
- **mode** -- read: FS_MODE_RD, write: FS_MODE_WR, both: FS_MODE_RD | FS_MODE_WR

返回 LV_FS_RES_OK or any error from lv_fs_res_t enum

lv_fs_res_t **lv_fs_close**(*lv_fs_file_t* *file_p)

Close an already opened file

参数 **file_p** -- pointer to a *lv_fs_file_t* variable

返回 LV_FS_RES_OK or any error from lv_fs_res_t enum

lv_fs_res_t **lv_fs_read**(*lv_fs_file_t* *file_p, void *buf, uint32_t btr, uint32_t *br)

Read from a file

参数

- **file_p** -- pointer to a *lv_fs_file_t* variable
- **buf** -- pointer to a buffer where the read bytes are stored
- **btr** -- Bytes To Read
- **br** -- the number of real read bytes (Bytes Read). NULL if unused.

返回 LV_FS_RES_OK or any error from lv_fs_res_t enum

lv_fs_res_t **lv_fs_write**(*lv_fs_file_t* *file_p, const void *buf, uint32_t btw, uint32_t *bw)

Write into a file

参数

- **file_p** -- pointer to a *lv_fs_file_t* variable
- **buf** -- pointer to a buffer with the bytes to write
- **btr** -- Bytes To Write
- **br** -- the number of real written bytes (Bytes Written). NULL if unused.

返回 LV_FS_RES_OK or any error from lv_fs_res_t enum

lv_fs_res_t **lv_fs_seek**(*lv_fs_file_t* *file_p, uint32_t pos, *lv_fs_whence_t* whence)

Set the position of the 'cursor' (read write pointer) in a file

参数

- **file_p** -- pointer to a *lv_fs_file_t* variable
- **pos** -- the new position expressed in bytes index (0: start of file)
- **whence** -- tells from where set the position. See @lv_fs_whence_t

返回 LV_FS_RES_OK or any error from lv_fs_res_t enum

lv_fs_res_t **lv_fs_tell**(*lv_fs_file_t* *file_p, uint32_t *pos)

Give the position of the read write pointer

参数

- **file_p** -- pointer to a *lv_fs_file_t* variable
- **pos_p** -- pointer to store the position of the read write pointer

返回 LV_FS_RES_OK or any error from 'fs_res_t'

lv_fs_res_t **lv_fs_dir_open**(*lv_fs_dir_t* *rddir_p, const char *path)

Initialize a 'fs_dir_t' variable for directory reading

参数

- **rddir_p** -- pointer to a '*lv_fs_dir_t*' variable
- **path** -- path to a directory

返回 LV_FS_RES_OK or any error from lv_fs_res_t enum

lv_fs_res_t **lv_fs_dir_read**(*lv_fs_dir_t* *rddir_p, char *fn)

Read the next filename form a directory. The name of the directories will begin with '/'

参数

- **rddir_p** -- pointer to an initialized 'fs_dir_t' variable
- **fn** -- pointer to a buffer to store the filename

返回 LV_FS_RES_OK or any error from lv_fs_res_t enum

lv_fs_res_t **lv_fs_dir_close**(*lv_fs_dir_t* *rddir_p)

Close the directory reading

参数 **rddir_p** -- pointer to an initialized 'fs_dir_t' variable

返回 LV_FS_RES_OK or any error from lv_fs_res_t enum

char ***lv_fs_get_letters**(char *buf)

Fill a buffer with the letters of existing drivers

参数 **buf** -- buffer to store the letters ('\0' added after the last letter)

返回 the buffer

const char ***lv_fs_get_ext**(const char *fn)

Return with the extension of the filename

参数 **fn** -- string with a filename

返回 pointer to the beginning extension or empty string if no extension

char ***lv_fs_up**(char *path)

Step up one level

参数 **path** -- pointer to a file name

返回 the truncated file name

const char ***lv_fs_get_last**(const char *path)*

Get the last element of a path (e.g. U:/folder/file -> file)

参数 **path** -- pointer to a file name

返回 pointer to the beginning of the last element in the path

struct **_lv_fs_drv_t**

Public Members

char **letter**

uint16_t **cache_size**

bool (***ready_cb**)(struct *_lv_fs_drv_t* *drv)

void *(***open_cb**)(struct *_lv_fs_drv_t* *drv, const char *path, *lv_fs_mode_t* mode)

lv_fs_res_t (***close_cb**)(struct *_lv_fs_drv_t* *drv, void *file_p)

lv_fs_res_t (***read_cb**)(struct *_lv_fs_drv_t* *drv, void *file_p, void *buf, uint32_t btr, uint32_t *br)

lv_fs_res_t (***write_cb**)(struct *_lv_fs_drv_t* *drv, void *file_p, const void *buf, uint32_t btw, uint32_t *bw)

lv_fs_res_t (***seek_cb**)(struct *_lv_fs_drv_t* *drv, void *file_p, uint32_t pos, *lv_fs_whence_t* whence)

lv_fs_res_t (***tell_cb**)(struct *_lv_fs_drv_t* *drv, void *file_p, uint32_t *pos_p)

void *(***dir_open_cb**)(struct *_lv_fs_drv_t* *drv, const char *path)

lv_fs_res_t (***dir_read_cb**)(struct *_lv_fs_drv_t* *drv, void *rddir_p, char *fn)

lv_fs_res_t (***dir_close_cb**)(struct *_lv_fs_drv_t* *drv, void *rddir_p)

void ***user_data**

Custom file user data

struct **lv_fs_file_cache_t**

Public Members

uint32_t **start**

uint32_t **end**

uint32_t **file_position**

void ***buffer**

struct **lv_fs_file_t**

Public Members

void ***file_d**

lv_fs_drv_t ***drv**

lv_fs_file_cache_t ***cache**

struct **lv_fs_dir_t**

Public Members

void ***dir_d**

lv_fs_drv_t ***drv**

2.5.14 Animations（动画）

You can automatically change the value of a variable between a start and an end value using animations. The animation will happen by periodically calling an ”animator” function with the corresponding value parameter.

The *animator* functions have the following prototype:

您可以使用动画在开始值和结束值之间自动更改变量的值。动画将通过使用相应的 value 参数定期调用“animator”函数来发生。

animator 函数具有以下原型：

```
void func(void * var, lv_anim_var_t value);
```

This prototype is compatible with the majority of the *set* functions of LVGL. For example `lv_obj_set_x(obj, value)` or `lv_obj_set_width(obj, value)`

该原型与 LVGL 的大多数 *set* 函数兼容。例如 `lv_obj_set_x(obj, value)` 或 `lv_obj_set_width(obj, value)`

Create an animation（创建动画）

To create an animation an `lv_anim_t` variable has to be initialized and configured with `lv_anim_set_...()` functions.

要创建动画，必须使用 `lv_anim_set_...()` 函数初始化和配置 `lv_anim_t` 变量。

```
/* INITIALIZE AN ANIMATION
 *----------------------*/

lv_anim_t a;
lv_anim_init(&a);

/* MANDATORY SETTINGS
 *------------------*/

/*Set the "animator" function*/
lv_anim_set_exec_cb(&a, (lv_anim_exec_xcb_t) lv_obj_set_x);

/*Set the "animator" function*/
lv_anim_set_var(&a, obj);

/*Length of the animation [ms]*/
lv_anim_set_time(&a, duration);

/*Set start and end values. E.g. 0, 150*/
lv_anim_set_values(&a, start, end);

/* OPTIONAL SETTINGS
 *------------------*/

/*Time to wait before starting the animation [ms]*/
lv_anim_set_delay(&a, delay);

/*Set path (curve). Default is linear*/
lv_anim_set_path(&a, lv_anim_path_ease_in);

/*Set a callback to call when animation is ready.*/
lv_anim_set_ready_cb(&a, ready_cb);

/*Set a callback to call when animation is started (after delay).*/
lv_anim_set_start_cb(&a, start_cb);

/*Play the animation backward too with this duration. Default is 0 (disabled) [ms]*/
```

(下页继续)

(续上页)

```
lv_anim_set_playback_time(&a, wait_time);

/*Delay before playback. Default is 0 (disabled) [ms]*/
lv_anim_set_playback_delay(&a, wait_time);

/*Number of repetitions. Default is 1.  LV_ANIM_REPEAT_INFINIT for infinite␣
↪repetition*/
lv_anim_set_repeat_count(&a, wait_time);

/*Delay before repeat. Default is 0 (disabled) [ms]*/
lv_anim_set_repeat_delay(&a, wait_time);

/*true (default): apply the start vale immediately, false: apply start vale after␣
↪delay when then anim. really starts. */
lv_anim_set_early_apply(&a, true/false);

/* START THE ANIMATION
 *------------------*/
lv_anim_start(&a);                              /*Start the animation*/
```

You can apply multiple different animations on the same variable at the same time. For example, animate the x and y coordinates with `lv_obj_set_x` and `lv_obj_set_y`. However, only one animation can exist with a given variable and function pair. Therefore `lv_anim_start()` will delete the already existing variable-function animations.

您可以同时对同一个变量应用多个不同的动画。例如，使用 `lv_obj_set_x` 和 `lv_obj_set_y` 为 x 和 y 坐标设置动画。但是，对于给定的变量和函数对，只能存在一个动画。因此 `lv_anim_start()` 将删除已经存在的可变函数动画。

Animation path（动画轨迹）

You can determinate the path of animation. The most simple case is linear, meaning the current value between *start* and *end* is changed with fixed steps. A *path* is a function which calculates the next value to set based on the current state of the animation. Currently, there are the following built-in paths functions:

- `lv_anim_path_linear` linear animation
- `lv_anim_path_step` change in one step at the end
- `lv_anim_path_ease_in` slow at the beginning
- `lv_anim_path_ease_out` slow at the end
- `lv_anim_path_ease_in_out` slow at the beginning and at the end
- `lv_anim_path_overshoot` overshoot the end value
- `lv_anim_path_bounce` bounce back a little from the end value (like hitting a wall)

您可以确定动画的路径。最简单的情况是线性的，这意味着 *start* 和 *end* 之间的当前值以固定步长变化。*path* 是一个函数，它根据动画的当前状态计算要设置的下一个值。目前，有以下内置路径函数：

- `lv_anim_path_linear` 线性动画
- `lv_anim_path_step` 最后一步改变
- `lv_anim_path_ease_in` 开始时很慢
- `lv_anim_path_ease_out` 最后慢
- `lv_anim_path_ease_in_out` 开始和结束都很慢
- `lv_anim_path_overshoot` 超过结束值
- `lv_anim_path_bounce` 从最终值反弹一点（比如撞墙）

Speed vs time（速度与时间）

By default, you set the animation time. But in some cases, setting the animation speed is more practical.

The `lv_anim_speed_to_time(speed, start, end)` function calculates the required time in milliseconds to reach the end value from a start value with the given speed. The speed is interpreted in *unit/sec* dimension. For example, `lv_anim_speed_to_time(20,0,100)` will yield 5000 milliseconds. For example, in case of `lv_obj_set_x` *unit* is pixels so *20* means *20 px/sec* speed.

默认情况下，您设置动画时间。但在某些情况下，设置动画速度更实用。

`lv_anim_speed_to_time(speed, start, end)` 函数计算从给定速度的起始值到达结束值所需的时间（以毫秒为单位）。速度以 *unit/sec* 维度解释。例如，`lv_anim_speed_to_time(20,0,100)` 将产生 5000 毫秒。例如，在 `lv_obj_set_x` 的情况下 *unit* 是像素，所以 *20* 意味着 *20 px/sec* 速度。

Delete animations（删除动画）

You can delete an animation with `lv_anim_del(var, func)` if you provide the animated variable and its animator function.

如果您提供动画变量及其动画器函数，您可以使用 `lv_anim_del(var, func)` 删除动画。

Timeline（时间线）

Timeline is a collection of multiple Animations, which makes it easy to create complex composite animations.

Firstly, create the animation element, but don’t call `lv_anim_start()`.

Secondly, create an animation timeline object, by calling `lv_anim_timeline_create()`.

Thirdly, add animation elements to the animation timeline, by calling `lv_anim_timeline_add(at, start_time, &a)`. `start_time` is the start time of the animation on the timeline. Note that `start_time` will override the value of `delay`.

Finally, call `lv_anim_timeline_start(at)` to start the animation timeline.

时间线是多个动画的集合，可以轻松创建复杂的复合动画。

首先，创建动画元素，但不要调用 `lv_anim_start()`。

其次，通过调用 `lv_anim_timeline_create()` 创建一个动画时间线对象。

第三，通过调用 `lv_anim_timeline_add(at, start_time, &a)` 将动画元素添加到动画时间线。`start_time` 是时间线上动画的开始时间。请注意，`start_time` 将覆盖 `delay` 的值。

最后，调用 `lv_anim_timeline_start(at)` 启动动画时间线。

It supports forward and backward playback of the entire animation group, using `lv_anim_timeline_set_reverse(at, reverse)`.

Call the `lv_anim_timeline_set_progress(at, progress)` function to set the state of the object corresponding to the progress of the timeline.

Call the `lv_anim_timeline_get_playtime(at)` function to get the total duration of the entire animation timeline.

Call the `lv_anim_timeline_get_reverse(at)` function to get whether to reverse the animation timeline.

Call the `lv_anim_timeline_del(at)` function to delete the animation timeline.

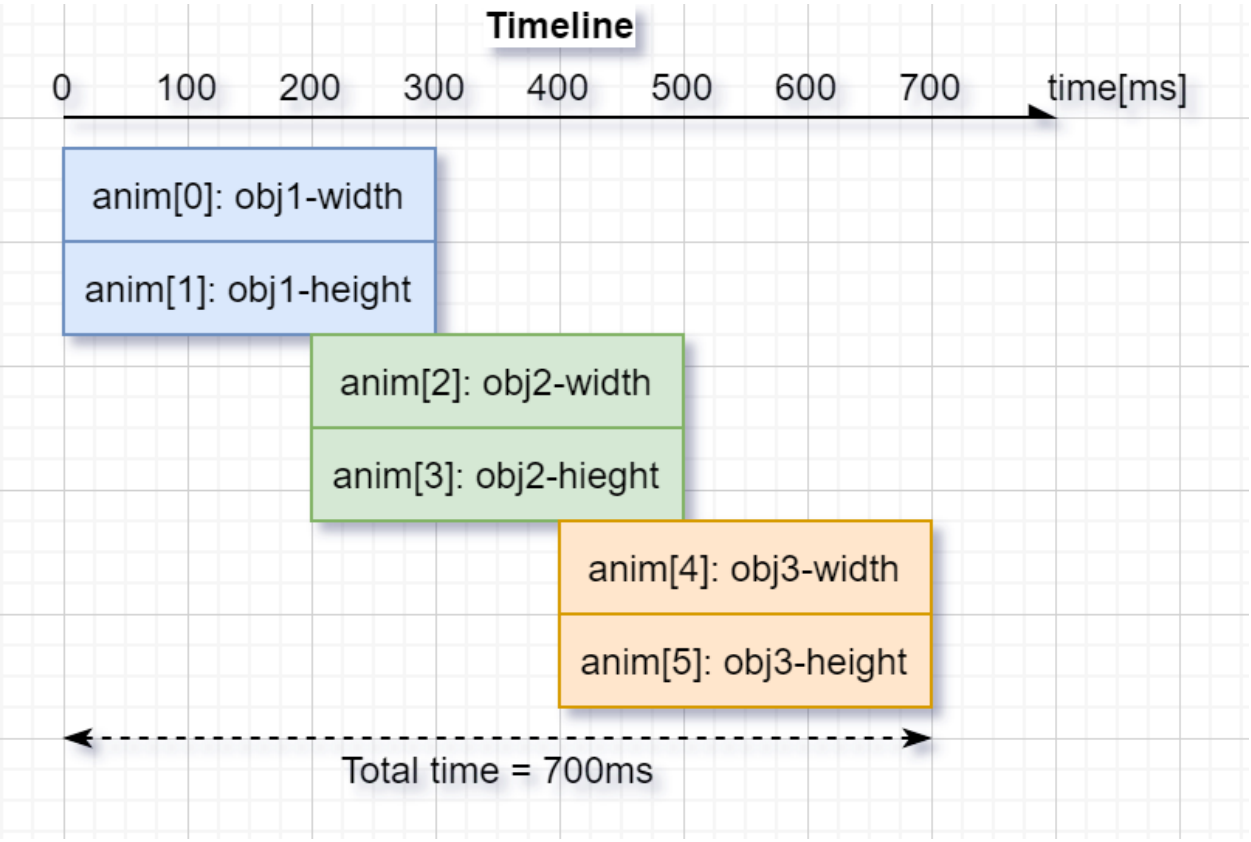
它支持整个动画组的向前和向后播放，使用 `lv_anim_timeline_set_reverse(at, reverse)`。

调用 `lv_anim_timeline_set_progress(at,progress)` 函数设置时间线进度对应的对象状态。

调用 `lv_anim_timeline_get_playtime(at)` 函数获取整个动画时间线的总时长。

调用 `lv_anim_timeline_get_reverse(at)` 函数获取是否反转动画时间线。

调用 `lv_anim_timeline_del(at)` 函数删除动画时间线。



Examples

Start animation on an event

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_SWITCH

static void anim_x_cb(void * var, int32_t v)
{
    lv_obj_set_x(var, v);
}

static void sw_event_cb(lv_event_t * e)
{
    lv_obj_t * sw = lv_event_get_target(e);
    lv_obj_t * label = lv_event_get_user_data(e);

    if(lv_obj_has_state(sw, LV_STATE_CHECKED)) {
        lv_anim_t a;
        lv_anim_init(&a);
```

(下页继续)

(续上页)

```
        lv_anim_set_var(&a, label);
        lv_anim_set_values(&a, lv_obj_get_x(label), 100);
        lv_anim_set_time(&a, 500);
        lv_anim_set_exec_cb(&a, anim_x_cb);
        lv_anim_set_path_cb(&a, lv_anim_path_overshoot);
        lv_anim_start(&a);
    } else {
        lv_anim_t a;
        lv_anim_init(&a);
        lv_anim_set_var(&a, label);
        lv_anim_set_values(&a, lv_obj_get_x(label), -lv_obj_get_width(label));
        lv_anim_set_time(&a, 500);
        lv_anim_set_exec_cb(&a, anim_x_cb);
        lv_anim_set_path_cb(&a, lv_anim_path_ease_in);
        lv_anim_start(&a);
    }

}

/**
 * Start animation on an event
 */
void lv_example_anim_1(void)
{
    lv_obj_t * label = lv_label_create(lv_scr_act());
    lv_label_set_text(label, "Hello animations!");
    lv_obj_set_pos(label, 100, 10);


    lv_obj_t * sw = lv_switch_create(lv_scr_act());
    lv_obj_center(sw);
    lv_obj_add_state(sw, LV_STATE_CHECKED);
    lv_obj_add_event_cb(sw, sw_event_cb, LV_EVENT_VALUE_CHANGED, label);
}

#endif
```

```
def anim_x_cb(label, v):
    label.set_x(v)

def sw_event_cb(e,label):
    sw = e.get_target()
```

(下页继续)

(续上页)

```
    if sw.has_state(lv.STATE.CHECKED):
        a = lv.anim_t()
        a.init()
        a.set_var(label)
        a.set_values(label.get_x(), 100)
        a.set_time(500)
        a.set_path_cb(lv.anim_t.path_overshoot)
        a.set_custom_exec_cb(lambda a,val: anim_x_cb(label,val))
        lv.anim_t.start(a)
    else:
        a = lv.anim_t()
        a.init()
        a.set_var(label)
        a.set_values(label.get_x(), -label.get_width())
        a.set_time(500)
        a.set_path_cb(lv.anim_t.path_ease_in)
        a.set_custom_exec_cb(lambda a,val: anim_x_cb(label,val))
        lv.anim_t.start(a)

#
# Start animation on an event
#

label = lv.label(lv.scr_act())
label.set_text("Hello animations!")
label.set_pos(100, 10)


sw = lv.switch(lv.scr_act())
sw.center()
sw.add_state(lv.STATE.CHECKED)
sw.add_event_cb(lambda e: sw_event_cb(e,label), lv.EVENT.VALUE_CHANGED, None)
```

Playback animation

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_SWITCH


static void anim_x_cb(void * var, int32_t v)
{
    lv_obj_set_x(var, v);
}

static void anim_size_cb(void * var, int32_t v)
{
    lv_obj_set_size(var, v, v);
}

/**
 * Create a playback animation
 */
void lv_example_anim_2(void)
{

    lv_obj_t * obj = lv_obj_create(lv_scr_act());
    lv_obj_set_style_bg_color(obj, lv_palette_main(LV_PALETTE_RED), 0);
    lv_obj_set_style_radius(obj, LV_RADIUS_CIRCLE, 0);

    lv_obj_align(obj, LV_ALIGN_LEFT_MID, 10, 0);

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_var(&a, obj);
    lv_anim_set_values(&a, 10, 50);
    lv_anim_set_time(&a, 1000);
    lv_anim_set_playback_delay(&a, 100);
    lv_anim_set_playback_time(&a, 300);
    lv_anim_set_repeat_delay(&a, 500);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
    lv_anim_set_path_cb(&a, lv_anim_path_ease_in_out);

    lv_anim_set_exec_cb(&a, anim_size_cb);
    lv_anim_start(&a);
    lv_anim_set_exec_cb(&a, anim_x_cb);
    lv_anim_set_values(&a, 10, 240);
    lv_anim_start(&a);
```

(下页继续)

(续上页)

```
}

#endif
```

```
def anim_x_cb(obj, v):
    obj.set_x(v)

def anim_size_cb(obj, v):
    obj.set_size(v, v)



#
# Create a playback animation
#
obj = lv.obj(lv.scr_act())
obj.set_style_bg_color(lv.palette_main(lv.PALETTE.RED), 0)
obj.set_style_radius(lv.RADIUS.CIRCLE, 0)

obj.align(lv.ALIGN.LEFT_MID, 10, 0)

a1 = lv.anim_t()
a1.init()
a1.set_var(obj)
a1.set_values(10, 50)
a1.set_time(1000)
a1.set_playback_delay(100)
a1.set_playback_time(300)
a1.set_repeat_delay(500)
a1.set_repeat_count(lv.ANIM_REPEAT.INFINITE)
a1.set_path_cb(lv.anim_t.path_ease_in_out)
a1.set_custom_exec_cb(lambda a1,val: anim_size_cb(obj,val))
lv.anim_t.start(a1)

a2 = lv.anim_t()
a2.init()
a2.set_var(obj)
a2.set_values(10, 240)
a2.set_time(1000)
a2.set_playback_delay(100)
a2.set_playback_time(300)
a2.set_repeat_delay(500)
a2.set_repeat_count(lv.ANIM_REPEAT.INFINITE)
a2.set_path_cb(lv.anim_t.path_ease_in_out)
```

(下页继续)

(续上页)

```
a2.set_custom_exec_cb(lambda a1,val: anim_x_cb(obj,val))
lv.anim_t.start(a2)
```

Animation timeline

```
#include "../lv_examples.h"
#if LV_USE_FLEX && LV_BUILD_EXAMPLES

static lv_anim_timeline_t * anim_timeline = NULL;

static lv_obj_t * obj1 = NULL;
static lv_obj_t * obj2 = NULL;
static lv_obj_t * obj3 = NULL;

static const lv_coord_t obj_width = 90;
static const lv_coord_t obj_height = 70;

static void set_width(void * var, int32_t v)
{
    lv_obj_set_width((lv_obj_t *)var, v);
}

static void set_height(void * var, int32_t v)
{
    lv_obj_set_height((lv_obj_t *)var, v);
}

static void anim_timeline_create(void)
{
    /* obj1 */
    lv_anim_t a1;
    lv_anim_init(&a1);
    lv_anim_set_var(&a1, obj1);
    lv_anim_set_values(&a1, 0, obj_width);
    lv_anim_set_early_apply(&a1, false);
    lv_anim_set_exec_cb(&a1, (lv_anim_exec_xcb_t)set_width);
    lv_anim_set_path_cb(&a1, lv_anim_path_overshoot);
    lv_anim_set_time(&a1, 300);

    lv_anim_t a2;
    lv_anim_init(&a2);
    lv_anim_set_var(&a2, obj1);
```

(下页继续)

(续上页)

```
    lv_anim_set_values(&a2, 0, obj_height);
    lv_anim_set_early_apply(&a2, false);
    lv_anim_set_exec_cb(&a2, (lv_anim_exec_xcb_t)set_height);
    lv_anim_set_path_cb(&a2, lv_anim_path_ease_out);
    lv_anim_set_time(&a2, 300);

    /* obj2 */
    lv_anim_t a3;
    lv_anim_init(&a3);
    lv_anim_set_var(&a3, obj2);
    lv_anim_set_values(&a3, 0, obj_width);
    lv_anim_set_early_apply(&a3, false);
    lv_anim_set_exec_cb(&a3, (lv_anim_exec_xcb_t)set_width);
    lv_anim_set_path_cb(&a3, lv_anim_path_overshoot);
    lv_anim_set_time(&a3, 300);

    lv_anim_t a4;
    lv_anim_init(&a4);
    lv_anim_set_var(&a4, obj2);
    lv_anim_set_values(&a4, 0, obj_height);
    lv_anim_set_early_apply(&a4, false);
    lv_anim_set_exec_cb(&a4, (lv_anim_exec_xcb_t)set_height);
    lv_anim_set_path_cb(&a4, lv_anim_path_ease_out);
    lv_anim_set_time(&a4, 300);

    /* obj3 */
    lv_anim_t a5;
    lv_anim_init(&a5);
    lv_anim_set_var(&a5, obj3);
    lv_anim_set_values(&a5, 0, obj_width);
    lv_anim_set_early_apply(&a5, false);
    lv_anim_set_exec_cb(&a5, (lv_anim_exec_xcb_t)set_width);
    lv_anim_set_path_cb(&a5, lv_anim_path_overshoot);
    lv_anim_set_time(&a5, 300);

    lv_anim_t a6;
    lv_anim_init(&a6);
    lv_anim_set_var(&a6, obj3);
    lv_anim_set_values(&a6, 0, obj_height);
    lv_anim_set_early_apply(&a6, false);
    lv_anim_set_exec_cb(&a6, (lv_anim_exec_xcb_t)set_height);
    lv_anim_set_path_cb(&a6, lv_anim_path_ease_out);
    lv_anim_set_time(&a6, 300);
```

(下页继续)

(续上页)

```
    /* Create anim timeline */
    anim_timeline = lv_anim_timeline_create();
    lv_anim_timeline_add(anim_timeline, 0, &a1);
    lv_anim_timeline_add(anim_timeline, 0, &a2);
    lv_anim_timeline_add(anim_timeline, 200, &a3);
    lv_anim_timeline_add(anim_timeline, 200, &a4);
    lv_anim_timeline_add(anim_timeline, 400, &a5);
    lv_anim_timeline_add(anim_timeline, 400, &a6);
}

static void btn_start_event_handler(lv_event_t * e)
{
    lv_obj_t * btn = lv_event_get_target(e);

    if (!anim_timeline) {
        anim_timeline_create();
    }

    bool reverse = lv_obj_has_state(btn, LV_STATE_CHECKED);
    lv_anim_timeline_set_reverse(anim_timeline, reverse);
    lv_anim_timeline_start(anim_timeline);
}

static void btn_del_event_handler(lv_event_t * e)
{
    LV_UNUSED(e);
    if (anim_timeline) {
        lv_anim_timeline_del(anim_timeline);
        anim_timeline = NULL;
    }
}

static void btn_stop_event_handler(lv_event_t * e)
{
    LV_UNUSED(e);
    if (anim_timeline) {
        lv_anim_timeline_stop(anim_timeline);
    }
}

static void slider_prg_event_handler(lv_event_t * e)
{
```

(下页继续)

(续上页)

```
    lv_obj_t * slider = lv_event_get_target(e);

    if (!anim_timeline) {
        anim_timeline_create();
    }

    int32_t progress = lv_slider_get_value(slider);
    lv_anim_timeline_set_progress(anim_timeline, progress);
}

/**
 * Create an animation timeline
 */
void lv_example_anim_timeline_1(void)
{
    lv_obj_t * par = lv_scr_act();
    lv_obj_set_flex_flow(par, LV_FLEX_FLOW_ROW);
    lv_obj_set_flex_align(par, LV_FLEX_ALIGN_SPACE_AROUND, LV_FLEX_ALIGN_CENTER, LV_
↪FLEX_ALIGN_CENTER);

    /* create btn_start */
    lv_obj_t * btn_start = lv_btn_create(par);
    lv_obj_add_event_cb(btn_start, btn_start_event_handler, LV_EVENT_VALUE_CHANGED,␣
↪NULL);
    lv_obj_add_flag(btn_start, LV_OBJ_FLAG_IGNORE_LAYOUT);
    lv_obj_add_flag(btn_start, LV_OBJ_FLAG_CHECKABLE);
    lv_obj_align(btn_start, LV_ALIGN_TOP_MID, -100, 20);

    lv_obj_t * label_start = lv_label_create(btn_start);
    lv_label_set_text(label_start, "Start");
    lv_obj_center(label_start);

    /* create btn_del */
    lv_obj_t * btn_del = lv_btn_create(par);
    lv_obj_add_event_cb(btn_del, btn_del_event_handler, LV_EVENT_CLICKED, NULL);
    lv_obj_add_flag(btn_del, LV_OBJ_FLAG_IGNORE_LAYOUT);
    lv_obj_align(btn_del, LV_ALIGN_TOP_MID, 0, 20);

    lv_obj_t * label_del = lv_label_create(btn_del);
    lv_label_set_text(label_del, "Delete");
    lv_obj_center(label_del);

    /* create btn_stop */
```

(下页继续)

(续上页)

```
    lv_obj_t * btn_stop = lv_btn_create(par);
    lv_obj_add_event_cb(btn_stop, btn_stop_event_handler, LV_EVENT_CLICKED, NULL);
    lv_obj_add_flag(btn_stop, LV_OBJ_FLAG_IGNORE_LAYOUT);
    lv_obj_align(btn_stop, LV_ALIGN_TOP_MID, 100, 20);

    lv_obj_t * label_stop = lv_label_create(btn_stop);
    lv_label_set_text(label_stop, "Stop");
    lv_obj_center(label_stop);

    /* create slider_prg */
    lv_obj_t * slider_prg = lv_slider_create(par);
    lv_obj_add_event_cb(slider_prg, slider_prg_event_handler, LV_EVENT_VALUE_CHANGED,␣
↪NULL);
    lv_obj_add_flag(slider_prg, LV_OBJ_FLAG_IGNORE_LAYOUT);
    lv_obj_align(slider_prg, LV_ALIGN_BOTTOM_MID, 0, -20);
    lv_slider_set_range(slider_prg, 0, 65535);

    /* create 3 objects */
    obj1 = lv_obj_create(par);
    lv_obj_set_size(obj1, obj_width, obj_height);

    obj2 = lv_obj_create(par);
    lv_obj_set_size(obj2, obj_width, obj_height);

    obj3 = lv_obj_create(par);
    lv_obj_set_size(obj3, obj_width, obj_height);
}

#endif
```

```
class LV_ExampleAnimTimeline_1(object):

    def __init__(self):
        self.obj_width = 120
        self.obj_height = 150
        #
        # Create an animation timeline
        #

        self.par = lv.scr_act()
        self.par.set_flex_flow(lv.FLEX_FLOW.ROW)
        self.par.set_flex_align(lv.FLEX_ALIGN.SPACE_AROUND, lv.FLEX_ALIGN.CENTER, lv.
↪FLEX_ALIGN.CENTER)
```

(下页继续)

(续上页)

```
        self.btn_run = lv.btn(self.par)
        self.btn_run.add_event_cb(self.btn_run_event_handler, lv.EVENT.VALUE_CHANGED,␣
↪None)
        self.btn_run.add_flag(lv.obj.FLAG.IGNORE_LAYOUT)
        self.btn_run.add_flag(lv.obj.FLAG.CHECKABLE)
        self.btn_run.align(lv.ALIGN.TOP_MID, -50, 20)

        self.label_run = lv.label(self.btn_run)
        self.label_run.set_text("Run")
        self.label_run.center()

        self.btn_del = lv.btn(self.par)
        self.btn_del.add_event_cb(self.btn_del_event_handler, lv.EVENT.CLICKED, None)
        self.btn_del.add_flag(lv.obj.FLAG.IGNORE_LAYOUT)
        self.btn_del.align(lv.ALIGN.TOP_MID, 50, 20)

        self.label_del = lv.label(self.btn_del)
        self.label_del.set_text("Stop")
        self.label_del.center()

        self.slider = lv.slider(self.par)
        self.slider.add_event_cb(self.slider_prg_event_handler, lv.EVENT.VALUE_
↪CHANGED, None)
        self.slider.add_flag(lv.obj.FLAG.IGNORE_LAYOUT)
        self.slider.align(lv.ALIGN.BOTTOM_RIGHT, -20, -20)
        self.slider.set_range(0, 65535)

        self.obj1 = lv.obj(self.par)
        self.obj1.set_size(self.obj_width, self.obj_height)

        self.obj2 = lv.obj(self.par)
        self.obj2.set_size(self.obj_width, self.obj_height)

        self.obj3 = lv.obj(self.par)
        self.obj3.set_size(self.obj_width, self.obj_height)

        self.anim_timeline = None

    def set_width(self,obj, v):
        obj.set_width(v)

    def set_height(self,obj, v):
```

(下页继续)

(续上页)

```
        obj.set_height(v)

    def anim_timeline_create(self):
        # obj1
        self.a1 = lv.anim_t()
        self.a1.init()
        self.a1.set_values(0, self.obj_width)
        self.a1.set_early_apply(False)
        self.a1.set_custom_exec_cb(lambda a,v: self.set_width(self.obj1,v))
        self.a1.set_path_cb(lv.anim_t.path_overshoot)
        self.a1.set_time(300)

        self.a2 = lv.anim_t()
        self.a2.init()
        self.a2.set_values(0, self.obj_height)
        self.a2.set_early_apply(False)
        self.a2.set_custom_exec_cb(lambda a,v: self.set_height(self.obj1,v))
        self.a2.set_path_cb(lv.anim_t.path_ease_out)
        self.a2.set_time(300)

        # obj2
        self.a3=lv.anim_t()
        self.a3.init()
        self.a3.set_values(0, self.obj_width)
        self.a3.set_early_apply(False)
        self.a3.set_custom_exec_cb(lambda a,v: self.set_width(self.obj2,v))
        self.a3.set_path_cb(lv.anim_t.path_overshoot)
        self.a3.set_time(300)

        self.a4 = lv.anim_t()
        self.a4.init()
        self.a4.set_values(0, self.obj_height)
        self.a4.set_early_apply(False)
        self.a4.set_custom_exec_cb(lambda a,v: self.set_height(self.obj2,v))
        self.a4.set_path_cb(lv.anim_t.path_ease_out)
        self.a4.set_time(300)

        # obj3
        self.a5 = lv.anim_t()
        self.a5.init()
        self.a5.set_values(0, self.obj_width)
        self.a5.set_early_apply(False)
        self.a5.set_custom_exec_cb(lambda a,v: self.set_width(self.obj3,v))
```

(下页继续)

(续上页)

```
        self.a5.set_path_cb(lv.anim_t.path_overshoot)
        self.a5.set_time(300)

        self.a6 = lv.anim_t()
        self.a6.init()
        self.a6.set_values(0, self.obj_height)
        self.a6.set_early_apply(False)
        self.a6.set_custom_exec_cb(lambda a,v: self.set_height(self.obj3,v))
        self.a6.set_path_cb(lv.anim_t.path_ease_out)
        self.a6.set_time(300)

        # Create anim timeline
        print("Create new anim_timeline")
        self.anim_timeline = lv.anim_timeline_create()
        lv.anim_timeline_add(self.anim_timeline, 0, self.a1)
        lv.anim_timeline_add(self.anim_timeline, 0, self.a2)
        lv.anim_timeline_add(self.anim_timeline, 200, self.a3)
        lv.anim_timeline_add(self.anim_timeline, 200, self.a4)
        lv.anim_timeline_add(self.anim_timeline, 400, self.a5)
        lv.anim_timeline_add(self.anim_timeline, 400, self.a6)

    def slider_prg_event_handler(self,e):
        slider = e.get_target()

        if  not self.anim_timeline:
            self.anim_timeline_create()

        progress = slider.get_value()
        lv.anim_timeline_set_progress(self.anim_timeline, progress)


    def btn_run_event_handler(self,e):
        btn = e.get_target()
        if not self.anim_timeline:
            self.anim_timeline_create()

        reverse = btn.has_state(lv.STATE.CHECKED)
        lv.anim_timeline_set_reverse(self.anim_timeline,reverse)
        lv.anim_timeline_start(self.anim_timeline)

    def btn_del_event_handler(self,e):
        if self.anim_timeline:
            lv.anim_timeline_del(self.anim_timeline)
```

(下页继续)

(续上页)

```
        self.anim_timeline = None


lv_example_anim_timeline_1 = LV_ExampleAnimTimeline_1()
```

API

Typedefs

typedef int32_t (*`lv_anim_path_cb_t`)(const struct *_lv_anim_t*)
Get the current value during an animation

typedef void (*`lv_anim_exec_xcb_t`)(void*, int32_t)
Generic prototype of "animator" functions. First parameter is the variable to animate. Second parameter is the value to set. Compatible with `lv_xxx_set_yyy(obj, value)` functions The `x` in `_xcb_t` means it's not a fully generic prototype because it doesn't receive `lv_anim_t *` as its first argument

typedef void (*`lv_anim_custom_exec_cb_t`)(struct *_lv_anim_t*, int32_t)
Same as `lv_anim_exec_xcb_t` but receives `lv_anim_t *` as the first parameter. It's more consistent but less convenient. Might be used by binding generator functions.

typedef void (*`lv_anim_ready_cb_t`)(struct *_lv_anim_t*)
Callback to call when the animation is ready

typedef void (*`lv_anim_start_cb_t`)(struct *_lv_anim_t*)
Callback to call when the animation really stars (considering `delay`)

typedef int32_t (*`lv_anim_get_value_cb_t`)(struct *_lv_anim_t*)
Callback used when the animation values are relative to get the current value

typedef struct *_lv_anim_t* `lv_anim_t`
Describes an animation

Enums

enum **lv_anim_enable_t**

Can be used to indicate if animations are enabled or disabled in a case

Values:

enumerator **LV_ANIM_OFF**

enumerator **LV_ANIM_ON**

Functions

LV_EXPORT_CONST_INT(LV_ANIM_REPEAT_INFINITE)

LV_EXPORT_CONST_INT(LV_ANIM_PLAYTIME_INFINITE)

void **_lv_anim_core_init**(void)

Init. the animation module

void **lv_anim_init**(*lv_anim_t* *a)

Initialize an animation variable. E.g.: lv_anim_t a; lv_anim_init(&a); lv_anim_set_...(&a); lv_anim_start(&a);

参数 **a** -- pointer to an `lv_anim_t` variable to initialize

static inline void **lv_anim_set_var**(*lv_anim_t* *a, void *var)

Set a variable to animate

参数

- **a** -- pointer to an initialized `lv_anim_t` variable
- **var** -- pointer to a variable to animate

static inline void **lv_anim_set_exec_cb**(*lv_anim_t* *a, *lv_anim_exec_xcb_t* exec_cb)

Set a function to animate `var`

参数

- **a** -- pointer to an initialized `lv_anim_t` variable
- **exec_cb** -- a function to execute during animation LVGL's built-in functions can be used. E.g. lv_obj_set_x

static inline void **lv_anim_set_time**(*lv_anim_t* *a, uint32_t duration)

Set the duration of an animation

参数

- **a** -- pointer to an initialized `lv_anim_t` variable
- **duration** -- duration of the animation in milliseconds

static inline void **lv_anim_set_delay**(*lv_anim_t* *a, uint32_t delay)

Set a delay before starting the animation

参数

- **a** -- pointer to an initialized `lv_anim_t` variable
- **delay** -- delay before the animation in milliseconds

static inline void **lv_anim_set_values**(*lv_anim_t* *a, int32_t start, int32_t end)

Set the start and end values of an animation

参数

- **a** -- pointer to an initialized `lv_anim_t` variable
- **start** -- the start value
- **end** -- the end value

static inline void **lv_anim_set_custom_exec_cb**(*lv_anim_t* *a, *lv_anim_custom_exec_cb_t* exec_cb)

Similar to `lv_anim_set_exec_cb` but `lv_anim_custom_exec_cb_t` receives `lv_anim_t *` as its first parameter instead of `void *`. This function might be used when LVGL is bound to other languages because it's more consistent to have `lv_anim_t *` as first parameter. The variable to animate can be stored in the animation's `user_data`

参数

- **a** -- pointer to an initialized `lv_anim_t` variable
- **exec_cb** -- a function to execute.

static inline void **lv_anim_set_path_cb**(*lv_anim_t* *a, *lv_anim_path_cb_t* path_cb)

Set the path (curve) of the animation.

参数

- **a** -- pointer to an initialized `lv_anim_t` variable
- **path_cb** -- a function to set the current value of the animation.

static inline void **lv_anim_set_start_cb**(*lv_anim_t* *a, *lv_anim_start_cb_t* start_cb)

Set a function call when the animation really starts (considering `delay`)

参数

- **a** -- pointer to an initialized `lv_anim_t` variable
- **start_cb** -- a function call when the animation starts

static inline void **lv_anim_set_get_value_cb**(*lv_anim_t* *a, *lv_anim_get_value_cb_t* get_value_cb)

Set a function to use the current value of the variable and make start and end value relative to the returned current value.

参数

- **a** -- pointer to an initialized `lv_anim_t` variable
- **get_value_cb** -- a function call when the animation starts

static inline void **lv_anim_set_ready_cb**(*lv_anim_t* *a, *lv_anim_ready_cb_t* ready_cb)

Set a function call when the animation is ready

参数

- **a** -- pointer to an initialized `lv_anim_t` variable
- **ready_cb** -- a function call when the animation is ready

static inline void **lv_anim_set_playback_time**(*lv_anim_t* *a, uint32_t time)

Make the animation to play back to when the forward direction is ready

参数

- **a** -- pointer to an initialized `lv_anim_t` variable
- **time** -- the duration of the playback animation in milliseconds. 0: disable playback

static inline void **lv_anim_set_playback_delay**(*lv_anim_t* *a, uint32_t delay)

Make the animation to play back to when the forward direction is ready

参数

- **a** -- pointer to an initialized `lv_anim_t` variable
- **delay** -- delay in milliseconds before starting the playback animation.

static inline void **lv_anim_set_repeat_count**(*lv_anim_t* *a, uint16_t cnt)

Make the animation repeat itself.

参数

- **a** -- pointer to an initialized `lv_anim_t` variable
- **cnt** -- repeat count or `LV_ANIM_REPEAT_INFINITE` for infinite repetition. 0: to disable repetition.

static inline void **lv_anim_set_repeat_delay**(*lv_anim_t* *a, uint32_t delay)

Set a delay before repeating the animation.

参数

- **a** -- pointer to an initialized `lv_anim_t` variable
- **delay** -- delay in milliseconds before repeating the animation.

static inline void **lv_anim_set_early_apply**(*lv_anim_t* *a, bool en)

Set a whether the animation's should be applied immediately or only when the delay expired.

参数

- **a** -- pointer to an initialized `lv_anim_t` variable
- **en** -- true: apply the start value immediately in `lv_anim_start`; false: apply the start value only when `delay` ms is elapsed and the animations really starts

static inline void **lv_anim_set_user_data**(*lv_anim_t* *a, void *user_data)

Set the custom user data field of the animation.

参数

- **a** -- pointer to an initialized `lv_anim_t` variable
- **user_data** -- pointer to the new user_data.

lv_anim_t ***lv_anim_start**(const *lv_anim_t* *a)

Create an animation

参数 **a** -- an initialized 'anim_t' variable. Not required after call.

返回 pointer to the created animation (different from the `a` parameter)

static inline uint32_t **lv_anim_get_delay**(*lv_anim_t* *a)

Get a delay before starting the animation

参数 **a** -- pointer to an initialized `lv_anim_t` variable

返回 delay before the animation in milliseconds

uint32_t **lv_anim_get_playtime**(*lv_anim_t* *a)

Get the time used to play the animation.

参数 **a** -- pointer to an animation.

返回 the play time in milliseconds.

static inline void ***lv_anim_get_user_data**(*lv_anim_t* *a)

Get the user_data field of the animation

参数 **a** -- pointer to an initialized `lv_anim_t` variable

返回 the pointer to the custom user_data of the animation

bool **lv_anim_del**(void *var, *lv_anim_exec_xcb_t* exec_cb)

Delete an animation of a variable with a given animator function

参数

- **var** -- pointer to variable
- **exec_cb** -- a function pointer which is animating 'var', or NULL to ignore it and delete all the animations of 'var

返回 true: at least 1 animation is deleted, false: no animation is deleted

void **lv_anim_del_all**(void)

Delete all the animations

lv_anim_t ***lv_anim_get**(void *var, *lv_anim_exec_xcb_t* exec_cb)

Get the animation of a variable and its `exec_cb`.

参数

- **var** -- pointer to variable
- **exec_cb** -- a function pointer which is animating 'var', or NULL to return first matching 'var'

返回 pointer to the animation.

static inline bool **lv_anim_custom_del**(*lv_anim_t* *a, *lv_anim_custom_exec_cb_t* exec_cb)

Delete an animation by getting the animated variable from `a`. Only animations with `exec_cb` will be deleted. This function exists because it's logical that all anim. functions receives an `lv_anim_t` as their first parameter. It's not practical in C but might make the API more consequent and makes easier to generate bindings.

参数

- **a** -- pointer to an animation.
- **exec_cb** -- a function pointer which is animating 'var', or NULL to ignore it and delete all the animations of 'var

返回 true: at least 1 animation is deleted, false: no animation is deleted

static inline *lv_anim_t* ***lv_anim_custom_get**(*lv_anim_t* *a, *lv_anim_custom_exec_cb_t* exec_cb)

Get the animation of a variable and its `exec_cb`. This function exists because it's logical that all anim. functions receives an `lv_anim_t` as their first parameter. It's not practical in C but might make the API more consequent and makes easier to generate bindings.

参数

- **a** -- pointer to an animation.
- **exec_cb** -- a function pointer which is animating 'var', or NULL to return first matching 'var'

返回 pointer to the animation.

uint16_t **lv_anim_count_running**(void)

Get the number of currently running animations

返回 the number of running animations

uint32_t **lv_anim_speed_to_time**(uint32_t speed, int32_t start, int32_t end)

Calculate the time of an animation with a given speed and the start and end values

参数

- **speed** -- speed of animation in unit/sec

- **start** -- start value of the animation
- **end** -- end value of the animation

返回 the required time [ms] for the animation with the given parameters

void **lv_anim_refr_now**(void)

Manually refresh the state of the animations. Useful to make the animations running in a blocking process where `lv_timer_handler` can't run for a while. Shouldn't be used directly because it is called in `lv_refr_now()`.

int32_t **lv_anim_path_linear**(const *lv_anim_t* *a)

Calculate the current value of an animation applying linear characteristic

参数 **a** -- pointer to an animation

返回 the current value to set

int32_t **lv_anim_path_ease_in**(const *lv_anim_t* *a)

Calculate the current value of an animation slowing down the start phase

参数 **a** -- pointer to an animation

返回 the current value to set

int32_t **lv_anim_path_ease_out**(const *lv_anim_t* *a)

Calculate the current value of an animation slowing down the end phase

参数 **a** -- pointer to an animation

返回 the current value to set

int32_t **lv_anim_path_ease_in_out**(const *lv_anim_t* *a)

Calculate the current value of an animation applying an "S" characteristic (cosine)

参数 **a** -- pointer to an animation

返回 the current value to set

int32_t **lv_anim_path_overshoot**(const *lv_anim_t* *a)

Calculate the current value of an animation with overshoot at the end

参数 **a** -- pointer to an animation

返回 the current value to set

int32_t **lv_anim_path_bounce**(const *lv_anim_t* *a)

Calculate the current value of an animation with 3 bounces

参数 **a** -- pointer to an animation

返回 the current value to set

int32_t **lv_anim_path_step**(const *lv_anim_t* *a)

Calculate the current value of an animation applying step characteristic. (Set end value on the end of the animation)

参数 **a** -- pointer to an animation

返回 the current value to set

struct **_lv_anim_t**

#include <lv_anim.h> Describes an animation

Public Members

void ***var**

Variable to animate

lv_anim_exec_xcb_t **exec_cb**

Function to execute to animate

lv_anim_start_cb_t **start_cb**

Call it when the animation is starts (considering `delay`)

lv_anim_ready_cb_t **ready_cb**

Call it when the animation is ready

lv_anim_get_value_cb_t **get_value_cb**

Get the current value in relative mode

void ***user_data**

Custom user data

lv_anim_path_cb_t **path_cb**

Describe the path (curve) of animations

int32_t **start_value**

Start value

int32_t **current_value**

Current value

int32_t **end_value**

End value

int32_t **time**

Animation time in ms

int32_t **act_time**

Current time in animation. Set to negative to make delay.

uint32_t **playback_delay**
Wait before play back

uint32_t **playback_time**
Duration of playback animation

uint32_t **repeat_delay**
Wait before repeat

uint16_t **repeat_cnt**
Repeat count for the animation

uint8_t **early_apply**
1: Apply start value immediately even is there is `delay`

uint8_t **playback_now**
Play back is in progress

uint8_t **run_round**
Indicates the animation has run in this round

uint8_t **start_cb_called**
Indicates that the `start_cb` was already called

2.5.15 Timers（定时器）

LVGL has a built-in timer system. You can register a function to have it be called periodically. The timers are handled and called in `lv_timer_handler()`, which needs to be called every few milliseconds. See *Porting* for more information.

The timers are non-preemptive, which means a timer cannot interrupt another timer. Therefore, you can call any LVGL related function in a timer.

LVGL 有一个内置的定时器系统。如果我们有一些需要定期执行的操作，可以往定时器中注册一个函数，这样 lvgl 就会定期调用执行。定时器系统在 `lv_timer_handler()` 中被处理和调用，它需要每隔几毫秒调用一次。有关详细信息，请参阅移植。

定时器是非抢占式的，所以一个定时器不能中断另一个定时器。因此，我们可以在定时器中调用任何与 LVGL 相关的函数。

Create a timer（创建定时器）

To create a new timer, use `lv_timer_create(timer_cb, period_ms, user_data)`. It will create an `lv_timer_t *` variable, which can be used later to modify the parameters of the timer. `lv_timer_create_basic()` can also be used. This allows you to create a new timer without specifying any parameters.

A timer callback should have `void (*lv_timer_cb_t)(lv_timer_t *);` prototype.

For example:

通过 `lv_timer_create(timer_cb, period_ms, user_data)` 创建一个新的定时器，它返回 `lv_timer_t *` 类型的指针，以后我们可以通过它来操作定时器。也可以使用 `lv_timer_create_basic()`。这允许我们在不指定任何参数的情况下创建一个新的定时器。

一个定时器的回调函数 (注册函数) 的原型格式是 `void (*lv_timer_cb_t)(lv_timer_t *);` 。

例如：

```
void my_timer(lv_timer_t * timer)
{
  /*Use the user_data*/
  uint32_t * user_data = timer->user_data;
  printf("my_timer called with user data: %d\n", *user_data);

  /*Do something with LVGL*/
  if(something_happened) {
    something_happened = false;
    lv_btn_create(lv_scr_act(), NULL);
  }
}

...

static uint32_t user_data = 10;
lv_timer_t * timer = lv_timer_create(my_timer, 500,  &user_data);
```

Ready and Reset（准备与重置）

`lv_timer_ready(timer)` makes the timer run on the next call of `lv_timer_handler()`.

`lv_timer_reset(timer)` resets the period of a timer. It will be called again after the defined period of milliseconds has elapsed.

`lv_timer_ready(timer)` 使定时器在下一次调用 `lv_timer_handler()` 时运行 (也就是会马上运行，而不是等过了给定的第一个周期过了之后才运行)。

`lv_timer_reset(timer)` 重置定时器的周期。它将在创建时设置的毫秒时间段过去后再调用。

Set parameters(参数设置)

You can modify some parameters of the timers later:

- `lv_timer_set_cb(timer, new_cb)`
- `lv_timer_set_period(timer, new_period)`

我们可以通过 lv_timer_create 返回的值，修改定时器的一些参数：

- `lv_timer_set_cb(timer, new_cb)`
- `lv_timer_set_period(timer, new_period)`

Repeat count(设置重复次数)

You can make a timer repeat only a given number of times with `lv_timer_set_repeat_count(timer, count)`. The timer will automatically be deleted after being called the defined number of times. Set the count to `-1` to repeat indefinitely.

我们可以使用 `lv_timer_set_repeat_count(timer, count)` 让注册的定时器仅重复给定次数。定时器在执行指定次数后会自动删除。将计数设置为 `-1` 会无限重复 (默认)。

Measure idle time(测量空闲时间)

You can get the idle percentage time of `lv_timer_handler` with `lv_timer_get_idle()`. Note that, it doesn't measure the idle time of the overall system, only `lv_timer_handler`. It can be misleading if you use an operating system and call `lv_timer_handler` in a timer, as it won't actually measure the time the OS spends in an idle thread.

可以通过 `lv_timer_get_idle()` 函数获取 `lv_timer_handler` 的空闲百分比时间。请注意，它不测量整个系统的空闲时间，只测量 `lv_timer_handler` 的空闲时间。如果您使用操作系统 (RTOS) 并在定时器中调用 `lv_timer_handler`，这可能会产生误导，因为它实际上不会测量操作系统 (RTOS) 在空闲线程中花费的时间。

Asynchronous calls(异步调用)

In some cases, you can't do an action immediately. For example, you can't delete an object because something else is still using it or you don't want to block the execution now. For these cases, `lv_async_call(my_function, data_p)` can be used to make `my_function` be called on the next call of `lv_timer_handler`. `data_p` will be passed to function when it's called. Note that, only the pointer of the data is saved so you need to ensure that the variable will be "alive" while the function is called. It can be *static*, global or dynamically allocated data.

For example:

在某些情况下，我们不能立即执行某些操作。例如，不能马上就删除一个对象，因为有其他东西仍在使用它，并且现在不能阻止它继续执行。对于这些情况，可以使用 `lv_async_call(my_function, data_p)` 使 `my_function`(也就是你的定时器回调函数) 在下一次调用 `lv_timer_handler` 时被调用。`data_p` 将在

调用时传递给函数。请注意，这仅保存数据的指针，因此需要确保在调用函数时该变量将“处于活动状态”。它可以是 静态、全局或动态分配的数据。

例如：

```
void my_screen_clean_up(void * scr)
{
  /*Free some resources related to `scr`*/

  /*Finally delete the screen*/
  lv_obj_del(scr);
}

...

/*Do somethings with the object on the current screen*/

/*Delete screen on next call of `lv_timer_handler`, so not now.*/
lv_async_call(my_screen_clean_up, lv_scr_act());

/*The screen is still valid so you can do other things with it*/
```

If you just want to delete an object, and don't need to clean anything up in `my_screen_cleanup`, you could just use `lv_obj_del_async`, which will delete the object on the next call to `lv_timer_handler`.

如果你只是想删除一个对象，而不需要在 `my_screen_cleanup` 中清理任何东西，你可以使用 `lv_obj_del_async`，它会在下次调用 `lv_timer_handler` 时删除该对象。

API

Typedefs

typedef void (*`lv_timer_cb_t`)(struct *_lv_timer_t**)
Timers execute this type of functions.

typedef struct *_lv_timer_t* `lv_timer_t`
Descriptor of a lv_timer

Functions

void **_lv_timer_core_init**(void)

Init the lv_timer module

lv_timer_t ***lv_timer_create_basic**(void)

Create an ”empty” timer. It needs to initialized with at least `lv_timer_set_cb` and `lv_timer_set_period`

返回 pointer to the created timer

lv_timer_t ***lv_timer_create**(*lv_timer_cb_t* timer_xcb, uint32_t period, void *user_data)

Create a new lv_timer

参数

- **timer_xcb** -- a callback to call periodically. (the ’x’ in the argument name indicates that it’s not a fully generic function because it not follows the `func_name(object, callback, ...)` convention)
- **period** -- call period in ms unit
- **user_data** -- custom parameter

返回 pointer to the new timer

void **lv_timer_del**(*lv_timer_t* *timer)

Delete a lv_timer

参数 **timer** -- pointer to an lv_timer

void **lv_timer_pause**(*lv_timer_t* *timer)

Pause/resume a timer.

参数 **timer** -- pointer to an lv_timer

void **lv_timer_resume**(*lv_timer_t* *timer)

void **lv_timer_set_cb**(*lv_timer_t* *timer, *lv_timer_cb_t* timer_cb)

Set the callback the timer (the function to call periodically)

参数

- **timer** -- pointer to a timer
- **timer_cb** -- the function to call periodically

void **lv_timer_set_period**(*lv_timer_t* *timer, uint32_t period)

Set new period for a lv_timer

参数

- **timer** -- pointer to a lv_timer

- **period** -- the new period

void **lv_timer_ready**(*lv_timer_t* *timer)

Make a lv_timer ready. It will not wait its period.

参数 **timer** -- pointer to a lv_timer.

void **lv_timer_set_repeat_count**(*lv_timer_t* *timer, int32_t repeat_count)

Set the number of times a timer will repeat.

参数

- **timer** -- pointer to a lv_timer.
- **repeat_count** -- -1 : infinity; 0 : stop ; n>0: residual times

void **lv_timer_reset**(*lv_timer_t* *timer)

Reset a lv_timer. It will be called the previously set period milliseconds later.

参数 **timer** -- pointer to a lv_timer.

void **lv_timer_enable**(bool en)

Enable or disable the whole lv_timer handling

参数 **en** -- true: lv_timer handling is running, false: lv_timer handling is suspended

uint8_t **lv_timer_get_idle**(void)

Get idle percentage

返回 the lv_timer idle in percentage

lv_timer_t ***lv_timer_get_next**(*lv_timer_t* *timer)

Iterate through the timers

参数 **timer** -- NULL to start iteration or the previous return value to get the next timer

返回 the next timer or NULL if there is no more timer

struct **_lv_timer_t**

#include <lv_timer.h> Descriptor of a lv_timer

Public Members

uint32_t **period**

How often the timer should run

uint32_t **last_run**

Last time the timer ran

lv_timer_cb_t **timer_cb**

Timer function

void ***user_data**
Custom user data

int32_t **repeat_count**
1: One time; -1 : infinity; n>0: residual times

uint32_t **paused**

Typedefs

typedef void (***lv_async_cb_t**)(void*)
Type for async callback.

Functions

lv_res_t **lv_async_call**(*lv_async_cb_t* async_xcb, void *user_data)
Call an asynchronous function the next time lv_timer_handler() is run. This function is likely to return **before** the call actually happens!

参数

- **async_xcb** -- a callback which is the task itself. (the 'x' in the argument name indicates that it's not a fully generic function because it not follows the `func_name(object, callback, ...)` convention)
- **user_data** -- custom parameter

2.5.16 Drawing（绘画）

With LVGL, you don't need to draw anything manually. Just create objects (like buttons, labels, arc, etc), move and change them, and LVGL will refresh and redraw what is required.

However, it might be useful to have a basic understanding of how drawing happens in LVGL to add customization, make it easier to find bugs or just out of curiosity.

The basic concept is to not draw directly to the screen, but draw to an internal draw buffer first. When drawing (rendering) is ready, that buffer is copied to the screen.

The draw buffer can be smaller than the screen's size. LVGL will simply render in "tiles" that fit into the given draw buffer.

使用 LVGL，您无需手动绘制任何内容。只需创建对象（如按钮、标签、圆弧等）、移动和更改它们，LVGL 将刷新并重绘所需的内容。

但是，对 LVGL 中的绘制方式有一个基本的了解以添加自定义、更容易地发现错误或只是出于好奇可能会很有用。

基本概念是不直接绘制到屏幕，而是首先绘制到内部绘制缓冲区。当绘图（渲染）准备好时，该缓冲区被复制到屏幕上。

绘制缓冲区可以小于屏幕的大小。LVGL 将简单地在适合给定绘制缓冲区的“图块”中进行渲染。

This approach has two main advantages compared to directly drawing to the screen:

1. It avoids flickering while the layers of the UI are drawn. For example, if LVGL drawn directly into the display, when drawing a *background* + *button* + *text*, each ”stage” would be visible for a short time .
2. It’s faster to modify a buffer in internal RAM and finally write one pixel only once than reading/writing the display directly on each pixel access. (e.g. via a display controller with SPI interface).

Note that this concept is different from ”traditional” double buffering where there are 2 screen sized frame buffers: one holds the current image to show on the display, and rendering happens to the other (inactive) frame buffer, and they are swapped when the rendering is finished. The main difference is that with LVGL you don’t have to store 2 frame buffers (which usually requires external RAM) but only smaller draw buffer(s) that can easily fit into the internal RAM too.

与直接绘制到屏幕相比，这种方法有两个主要优点：

1. 避免绘制 UI 层时闪烁。例如，如果 LVGL 直接绘制到显示中，那么在绘制 * 背景 + 按钮 + 文本 * 时，每个“阶段”都会在短时间内可见。
2. 修改内部 RAM 中的缓冲区并最终仅写入一个像素一次比在每个像素访问时直接读取/写入显示更快。（例如，通过带有 SPI 接口的显示控制器）。

请注意，此概念与“传统”双缓冲不同，后者有 2 个屏幕大小的帧缓冲区：一个保存当前图像以显示在显示器上，渲染发生在另一个（非活动）帧缓冲区中，渲染完成后它们会被交换。主要区别在于，使用 LVGL，您不必存储 2 个帧缓冲区（通常需要外部 RAM），而只需存储更小的绘图缓冲区，也可以轻松装入内部 RAM。

Mechanism of screen refreshing（屏幕刷新机制）

Be sure to get familiar with the *Buffering modes of LVGL* first.

LVGL refreshes the screen in the following steps:

1. Something happens on the UI which requires redrawing. For example, a button is pressed, a chart is changed, an animation happened, etc.
2. LVGL saves the changed object’s old and new area into a buffer, called an *Invalid area buffer*. For optimization, in some cases, objects are not added to the buffer:
    - Hidden objects are not added.
    - Objects completely out of their parent are not added.
    - Areas partially out of the parent are cropped to the parent’s area.
    - The objects on other screens are not added.

3. In every `LV_DISP_DEF_REFR_PERIOD` (set in `lv_conf.h`) the followings happen:
    - LVGL checks the invalid areas and joins the adjacent or intersecting areas.
    - Takes the first joined area, if it's smaller than the *draw buffer*, then simply render the area's content into the *draw buffer*. If the area doesn't fit into the buffer, draw as many lines as possible to the *draw buffer*.
    - When the area is rendered, call `flush_cb` from the display driver to refresh the display.
    - If the area was larger than the buffer, render the remaining parts too.
    - Do the same with all the joined areas.

一定要先熟悉*LVGL 的缓冲机制*。

LVGL 按以下步骤刷新屏幕：

1. UI 上发生了一些需要重绘的事情。例如，按下按钮、更改图表、发生动画等。
2. LVGL 将改变对象的旧区和新区保存到一个缓冲区中，称为*无效区缓冲区*。为了优化，在某些情况下，不会将对象添加到缓冲区中：
    - 不添加隐藏对象。
    - 不添加完全脱离其父对象的对象。
    - 部分超出父级的区域被裁剪到父级的区域。
    - 不添加其他屏幕上的对象。
3. 在每个 `LV_DISP_DEF_REFR_PERIOD`（在 `lv_conf.h` 中设置）发生以下情况：
    - LVGL 检查无效区域并连接相邻或相交的区域。
    - 获取第一个连接区域，如果它小于 *draw buffer*，则只需将该区域的内容渲染到 *draw buffer* 中。如果该区域不适合缓冲区，则在 *draw buffer* 中绘制尽可能多的线。
    - 当区域被渲染时，从显示驱动程序调用 `flush_cb` 来刷新显示。
    - 如果该区域大于缓冲区，也渲染其余部分。
    - 对所有连接的区域执行相同操作。

When an area is redrawn, the library searches the top most object which covers that area, and starts drawing from that object. For example, if a button's label has changed, the library will see that it's enough to draw the button under the text, and that it's not required to draw the screen under the button too.

The difference between buffering modes regarding the drawing mechanism is the following:

1. **One buffer** - LVGL needs to wait for `lv_disp_flush_ready()` (called from `flush_cb`) before starting to redraw the next part.
2. **Two buffers** - LVGL can immediately draw to the second buffer when the first is sent to `flush_cb` because the flushing should be done by DMA (or similar hardware) in the background.
3. **Double buffering** - `flush_cb` should only swap the address of the frame buffer.

当一个区域被重绘时，库搜索覆盖该区域的最上面的对象，并从该对象开始绘制。例如，如果按钮的标签发生了变化，库将看到在文本下方绘制按钮就足够了，并且不需要在按钮下方绘制屏幕。

关于绘图机制的缓冲模式之间的区别如下：

1. **One buffer** - LVGL 需要等待 `lv_disp_flush_ready()`（从 `flush_cb` 调用），然后才开始重绘下一部分。
2. **两个缓冲区** - 当第一个缓冲区发送到 `flush_cb` 时，LVGL 可以立即绘制到第二个缓冲区，因为刷新应该由 DMA（或类似硬件）在后台完成。
3. **双缓冲** - `flush_cb` 应该只交换帧缓冲区的地址。

Masking（蒙版）

Masking is the basic concept of LVGL's draw engine. To use LVGL it's not required to know about the mechanisms described here, but you might find interesting to know how drawing works under hood. Knowing about masking comes in handy if you want to customize drawing.

To learn masking let's learn the steps of drawing first. LVGL performs the following steps to render any shape, image or text. It can be considered as a drawing pipeline.

1. **Prepare the draw descriptors** Create a draw descriptor from an object's styles (e.g. `lv_draw_rect_dsc_t`). This gives us the parameters for drawing, for example the colors, widths, opacity, fonts, radius, etc.
2. **Call the draw function** Call the draw function with the draw descriptor and some other parameters (e.g. `lv_draw_rect()`). It renders the primitive shape to the current draw buffer.
3. **Create masks** If the shape is very simple and doesn't require masks go to #5. Else create the required masks (e.g. a rounded rectangle mask)
4. **Calculate all the added mask**. It creates 0..255 values into a *mask buffer* with the "shape" of the created masks. E.g. in case of a "line mask" according to the parameters of the mask, keep one side of the buffer as it is (255 by default) and set the rest to 0 to indicate that this side should be removed.
5. **Blend a color or image** During blending masks (make some pixels transparent or opaque), blending modes (additive, subtractive, etc) and opacity are handled.

Masking 是 LVGL 绘图引擎的基本概念。要使用 LVGL，不需要了解这里描述的机制，但您可能会发现了解如何在引擎盖下绘制是很有趣的。如果您想自定义绘图，了解蒙版会派上用场。

要学习蒙版，让我们先学习绘图的步骤。LVGL 执行以下步骤来渲染任何形状、图像或文本。它可以被认为是一个绘图管道。

1. **准备绘制描述符**根据对象的样式（例如 `lv_draw_rect_dsc_t`）创建绘制描述符。这为我们提供了绘图参数，例如颜色、宽度、不透明度、字体、半径等。
2. **调用绘图函数**使用绘图描述符和一些其他参数（例如 `lv_draw_rect()`）调用绘图函数。它将原始形状渲染到当前绘制缓冲区。
3. **创建蒙版**如果形状非常简单并且不需要蒙版，请转到 #5。否则创建所需的蒙版（例如圆角矩形蒙版）

4. **计算所有添加的蒙版**。它使用创建的蒙版的“形状”将 0..255 个值创建到 蒙版缓冲区中。例如。如果根据蒙版的参数出现“线蒙版”，则将缓冲区的一侧保持原样（默认为 255）并将其余部分设置为 0 以指示应删除该侧。
5. **混合颜色或图像**在混合蒙版（使一些像素透明或不透明）期间，处理混合模式（加法、减法等）和不透明度。

LVGL has the following built-in mask types which can be calculated and applied real-time:

- `LV_DRAW_MASK_TYPE_LINE` Removes a side from a line (top, bottom, left or right). `lv_draw_line` uses 4 of it. Essentially, every (skew) line is bounded with 4 line masks by forming a rectangle.
- `LV_DRAW_MASK_TYPE_RADIUS` Removes the inner or outer parts of a rectangle which can have radius. It's also used to create circles by setting the radius to large value (`LV_RADIUS_CIRCLE`)
- `LV_DRAW_MASK_TYPE_ANGLE` Removes a circle sector. It is used by `lv_draw_arc` to remove the "empty" sector.
- `LV_DRAW_MASK_TYPE_FADE` Create a vertical fade (change opacity)
- `LV_DRAW_MASK_TYPE_MAP` The mask is stored in an array and the necessary parts are applied

LVGL 具有以下可以实时计算和应用的内置蒙版类型：

- `LV_DRAW_MASK_TYPE_LINE` 从一条线（顶部、底部、左侧或右侧）中删除一侧。`lv_draw_line` 使用了其中的 4 个。本质上，每条（倾斜）线都通过形成一个矩形以 4 个线掩模为界。
- `LV_DRAW_MASK_TYPE_RADIUS` 移除可以有半径的矩形的内部或外部部分。它还用于通过将半径设置为大值（`LV_RADIUS_CIRCLE`）来创建圆
- `LV_DRAW_MASK_TYPE_ANGLE` 删除一个圆形扇区。`lv_draw_arc` 使用它来删除“空”扇区。
- `LV_DRAW_MASK_TYPE_FADE` 创建垂直淡入淡出（改变不透明度）
- `LV_DRAW_MASK_TYPE_MAP` 蒙版存储在一个数组中，并应用必要的部分

Masks are used the create almost every basic primitives:

- **letters** Create a mask from the letter and draw a rectangle with the letter's color considering the mask.
- **line** Created from 4 "line masks", to mask out the left, right, top and bottom part of the line to get perfectly perpendicular line ending.
- **rounded rectangle** A mask is created real-time to add radius to the corners.
- **clip corner** To clip to overflowing content (usually children) on the rounded corners also a rounded rectangle mask is applied.
- **rectangle border** Same as a rounded rectangle, but inner part is masked out too.
- **arc drawing** A circle border is drawn, but an arc mask is applied too.
- **ARGB images** The alpha channel is separated into a mask and the image is drawn as a normal RGB image.

蒙版用于创建几乎所有基本图元：

- **字母**根据字母创建一个蒙版，并根据蒙版用字母的颜色绘制一个矩形。
- **line** 由 4 个“线蒙版”创建，用于遮住线的左、右、上和下部分，以获得完美垂直的线尾。
- **圆角矩形**实时创建蒙版以向角添加半径。
- **剪辑角落**为了剪辑到圆角上的溢出内容（通常是儿童），还应用了圆角矩形蒙版。
- **矩形边框**与圆角矩形相同，但内部部分也被屏蔽了。
- **圆弧绘制**绘制了圆形边框，但也应用了圆弧蒙版。
- **ARGB 图像** alpha 通道被分离成一个蒙版，图像被绘制为一个普通的 RGB 图像。

Hook drawing（挂钩绘图）

Although widgets can be very well customized by styles there might be cases when something really custom is required. To ensure a great level of flexibility LVGL sends a lot events during drawing with parameters that tell what LVGL is about to draw. Some fields of these parameters can be modified to draw something else or any custom drawing can be added manually.

A good use case for it is the *Button matrix* widget. By default its buttons can be styled in different states but you can't style the buttons one by one. However, an event is sent for every button and you can for example tell LVGL to use different colors on a specific button or to manually draw an image on some buttons.

Below each of these events are described in detail.

尽管小部件可以通过样式很好地自定义，但在某些情况下可能需要真正自定义。为了确保高度的灵活性，LVGL 在绘制过程中发送了很多事件，并带有告诉 LVGL 将要绘制什么的参数。可以修改这些参数的某些字段以绘制其他内容，或者可以手动添加任何自定义绘图。

它的一个很好的用例是*Button matrix* 小部件。默认情况下，它的按钮可以在不同状态下设置样式，但您不能一个一个地设置按钮样式。但是，每个按钮都会发送一个事件，例如，您可以告诉 LVGL 在特定按钮上使用不同的颜色或在某些按钮上手动绘制图像。下面详细描述了这些事件中的每一个。

Main drawing（主图）

These events are related to the actual drawing of the object. E.g. drawing of buttons, texts, etc happens here.

`lv_event_get_clip_area(event)` can be used to get the current clip area. The clip area is required in draw functions to make them draw only on a limited area.

这些事件与对象的实际绘制有关。例如。按钮、文本等的绘制发生在这里。

`lv_event_get_clip_area(event)` 可以用来获取当前的剪辑区域。绘制函数中需要剪辑区域，以使它们仅在有限的区域上绘制。

LV_EVENT_DRAW_MAIN_BEGIN

Sent before starting to draw an object. This is a good place to add masks manually. E.g. add a line mask that ”removes” the right side of an object.

在开始绘制对象之前发送。这是手动添加蒙版的好地方。例如。添加一个“删除”对象右侧的线蒙版。

LV_EVENT_DRAW_MAIN

The actual drawing of the object happens in this event. E.g. a rectangle for a button is drawn here. First, the widgets' internal events are called to perform drawing and after that you can draw anything on top of them. For example you can add a custom text or an image.

对象的实际绘制发生在此事件中。例如。这里绘制了一个按钮的矩形。首先，调用小部件的内部事件来执行绘图，然后您可以在它们之上绘制任何内容。例如，您可以添加自定义文本或图像。

LV_EVENT_DRAW_MAIN_END

Called when the main drawing is finished. You can draw anything here as well and it's also good place to remove the masks created in `LV_EVENT_DRAW_MAIN_BEGIN`.

在主绘图完成时调用。你也可以在这里画任何东西，它也是删除在 `LV_EVENT_DRAW_MAIN_BEGIN` 中创建的蒙版的好地方。

Post drawing（后绘图）

Post drawing events are called when all the children of an object are drawn. For example LVGL use the post drawing phase to draw the scrollbars because they should be above all the children.

`lv_event_get_clip_area(event)` can be used to get the current clip area.

当绘制对象的所有子项时，会调用绘制后事件。例如，LVGL 使用后期绘制阶段来绘制滚动条，因为它们应该位于所有子项之上。`lv_event_get_clip_area(event)` 可以用来获取当前的剪辑区域。

LV_EVENT_DRAW_POST_BEGIN

Sent before starting the post draw phase. Masks can be added here too to mask out the post drawn content.

在开始绘制后阶段之前发送。也可以在此处添加遮罩以遮蔽后期绘制的内容。

LV_EVENT_DRAW_POST

The actual drawing should happen here.

实际绘图应该发生在这里。

LV_EVENT_DRAW_POST_END

Called when post drawing has finished. If the masks were not removed in `LV_EVENT_DRAW_MAIN_END` they should be removed here.

在后期绘制完成时调用。如果在 `LV_EVENT_DRAW_MAIN_END` 中没有移除遮罩，则应在此处移除。

Part drawing（零件绘图）

When LVGL draws a part of an object (e.g. a slider's indicator, a table's cell or a button matrix's button) it sends events before and after drawing that part with some context of the drawing. It allows changing the parts on a very low level with masks, extra drawing, or changing the parameters that LVGL is planning to use for drawing.

In these events an `lv_obj_draw_part_t` structure is used to describe the context of the drawing. Not all fields are set for every part and widget. To see which fields are set for a widget see the widget's documentation.

`lv_obj_draw_part_t` has the following fields:

当 LVGL 绘制对象的一部分（例如滑块的指示器、表格的单元格或按钮矩阵的按钮）时，它会在绘制该部分之前和之后使用绘图的某些上下文发送事件。它允许使用蒙版、额外绘图或更改 LVGL 计划用于绘图的参数在非常低的级别上更改零件。

在这些事件中，`lv_obj_draw_part_t` 结构用于描述绘图的上下文。并非为每个部件和小部件设置所有字段。要查看为小部件设置了哪些字段，请参阅小部件的文档。

`lv_obj_draw_part_t` 具有以下字段：

```
// Always set
const lv_area_t * clip_area;        // The current clip area, required if you need to␣
↪draw something in the event
uint32_t part;                      // The current part for which the event is sent
uint32_t id;                        // The index of the part. E.g. a button's index␣
↪on button matrix or table cell index.

// Draw desciptors, set only if related
lv_draw_rect_dsc_t * rect_dsc;      // A draw descriptor that can be modified to␣
↪changed what LVGL will draw. Set only for rectangle-like parts
lv_draw_label_dsc_t * label_dsc;    // A draw descriptor that can be modified to␣
↪changed what LVGL will draw. Set only for text-like parts
lv_draw_line_dsc_t * line_dsc;      // A draw descriptor that can be modified to␣
↪changed what LVGL will draw. Set only for line-like parts
```

(下页继续)

(续上页)

```
lv_draw_img_dsc_t *  img_dsc;       // A draw descriptor that can be modified to␣
↪changed what LVGL will draw. Set only for image-like parts
lv_draw_arc_dsc_t *  arc_dsc;       // A draw descriptor that can be modified to␣
↪changed what LVGL will draw. Set only for arc-like parts

// Other paramters
lv_area_t * draw_area;              // The area of the part being drawn
const lv_point_t * p1;              // A point calculated during drawing. E.g. a␣
↪point of chart or the center of an arc.
const lv_point_t * p2;              // A point calculated during drawing. E.g. a␣
↪point of chart.
char text[16];                      // A text calculated during drawing. Can be␣
↪modified. E.g. tick labels on a chart axis.
lv_coord_t radius;                  // E.g. the radius of an arc (not the corner␣
↪radius).
int32_t value;                      // A value calculated during drawing. E.g. Chart
↪'s tick line value.
const void * sub_part_ptr;          // A pointer the identifies something in the part.
↪ E.g. chart series.
```

`lv_event_get_draw_part_dsc(event)` can be used to get a pointer to `lv_obj_draw_part_t`.

`lv_event_get_draw_part_dsc(event)` 可用于获取指向 `lv_obj_draw_part_t` 的指针。

LV_EVENT_DRAW_PART_BEGIN

Start the drawing of a part. This is a good place to modify the draw descriptors (e.g. `rect_dsc`), or add masks.

开始绘制零件。这是修改绘图描述符（例如 `rect_dsc`）或添加遮罩的好地方。

LV_EVENT_DRAW_PART_END

Finish the drawing of a part. This is a good place to draw extra content on the part, or remove the masks added in `LV_EVENT_DRAW_PART_BEGIN`.

完成零件的绘制。这是在零件上绘制额外内容的好地方，或者删除在 `LV_EVENT_DRAW_PART_BEGIN` 中添加的蒙版。

Others（其他）

LV_EVENT_COVER_CHECK

This event is used to check whether an object fully covers an area or not.

`lv_event_get_cover_area(event)` returns an pointer to an area to check and `lv_event_set_cover_res(event, res)` can be used to set one of these results:

- `LV_COVER_RES_COVER` the areas is fully covered by the object
- `LV_COVER_RES_NOT_COVER` the areas is not covered by the object
- `LV_COVER_RES_MASKED` there is a mask on the object so it can not cover the area

Here are some reasons why an object would be unable to fully cover an area:

- It's simply not fully in area
- It has a radius
- It has not 100% background opacity
- It's an ARGB or chroma keyed image
- It does not have normal blending mode. In this case LVGL needs to know the colors under the object to do the blending properly
- It's a text, etc

此事件用于检查对象是否完全覆盖一个区域。

`lv_event_get_cover_area(event)` 返回一个指向要检查的区域的指针，`lv_event_set_cover_res(event, res)` 可用于设置以下结果之一：

- `LV_COVER_RES_COVER` 区域被对象完全覆盖
- `LV_COVER_RES_NOT_COVER` 区域未被对象覆盖
- `LV_COVER_RES_MASKED` 对象上有一个遮罩，所以它不能覆盖该区域

以下是物体无法完全覆盖区域的一些原因：

- 它只是不完全在区域内
- 它有一个半径
- 它没有 100% 的背景不透明度
- 这是一个 ARGB 或色度键控图像
- 它没有正常的混合模式。在这种情况下，LVGL 需要知道对象下的颜色才能正确进行混合
- 这是一个文本等

In short if for any reason the area below the object is visible than it doesn't cover that area.

Before sending this event LVGL checks if at least the widget's coordinates fully cover the area or not. If not the event is not called.

You need to check only the drawing you have added. The existing properties known by widget are handled in the widget's internal events. E.g. if a widget has > 0 radius it might not cover an area but you need to handle `radius` only if you will modify it and the widget can't know about it.

简而言之，如果由于某种原因对象下方的区域是可见的，则它不覆盖该区域。

在发送此事件之前，LVGL 检查至少小部件的坐标是否完全覆盖该区域。如果不是，则不会调用该事件。

您只需要检查您添加的图纸。小部件已知的现有属性在小部件的内部事件中处理。例如。如果小部件具有 > 0 半径，它可能不会覆盖一个区域，但只有当您修改它并且小部件无法知道它时，您才需要处理“半径”。

LV_EVENT_REFR_EXT_DRAW_SIZE

If you need to draw outside of a widget LVGL needs to know about it to provide the extra space for drawing. Let's say you create an event the writes the current value of a slider above its knob. In this case LVGL needs to know that the slider's draw area should be larger with the size required for the text.

You can simple set the required draw area with `lv_event_set_ext_draw_size(e, size)`.

如果您需要在小部件之外绘制，LVGL 需要了解它以提供额外的绘制空间。假设您创建了一个事件，将滑块的当前值写入其旋钮上方。在这种情况下，LVGL 需要知道滑块的绘制区域应该与文本所需的大小相同。

您可以使用 `lv_event_set_ext_draw_size(e, size)` 简单设置所需的绘图区域。

2.5.17 New widget

2.6 Widgets（组件）

2.6.1 Base object（基础对象）(lv_obj)

Overview

The 'Base Object' implements the basic properties of widgets on a screen, such as:

- coordinates
- parent object
- children
- contains the styles
- attributes like *Clickable*, *Scrollable*, etc.

In object-oriented thinking, it is the base class from which all other objects in LVGL are inherited.

The functions and functionalities of the Base object can be used with other widgets too. For example `lv_obj_set_width(slider, 100)`

The Base object can be directly used as a simple widget: it's nothing more than a rectangle. In HTML terms, think of it as a `<div>`.

基础对象实现了屏幕上组件的基本属性，例如：

- 坐标
- 父对象
- 基于父对象的后代
- 包含样式
- 诸如 *Clickable*、*Scrollable* 等属性。

在面向对象的思想中，基础对象是 LVGL 中所有其他对象都继承自的基类。

基础对象的功能可以与其他组件一起使用。例如 `lv_obj_set_width(slider, 100);`

基础对象可以直接用作一个简单的组件：它只不过是一个矩形。在 HTML 术语中，将其视为 **<\div>**。

Coordinates（坐标）

Only a small subset of coordinate settings is described here. To see all the features of LVGL (padding, coordinates in styles, layouts, etc) visit the *Coordinates* page.

此处仅描述了一小部分坐标设置。要查看 LVGL 的所有功能（填充、样式中的坐标、布局等），请阅读坐标页面了解。

Size（大小）

The object size can be modified on individual axes with `lv_obj_set_width(obj, new_width);` and `lv_obj_set_height(obj, new_height);`, or both axes can be modified at the same time with `lv_obj_set_size(obj, new_width, new_height);`.

可以使用 `lv_obj_set_width(obj, new_width);` 和 `lv_obj_set_height(obj, new_height);` 设置或修改单个轴上的对象大小，或者可以同时修改两个轴，使用 `lv_obj_set_width(obj,new_set_width);` 设置或修改高度。

Position（位置）

You can set the position relative to the parent with `lv_obj_set_x(obj, new_x);` and `lv_obj_set_y(obj, new_y);`, or both axes at the same time with `lv_obj_set_pos(obj, new_x, new_y);`.

您可以使用 `lv_obj_set_x(obj, new_x);` 和 `lv_obj_set_y(obj, new_y);` 设置相对于父级的位置，或者使用 `lv_obj_set_pos(obj, new_x, new_y);` 来同时设置两个轴的位置。

Alignment（对齐）

You can align the object on its parent with `lv_obj_set_align(obj, LV_ALIGN_...);`. After this every x and y setting will be relative to the set alignment mode. For example, this will shift the object by 10;20 px from the center of its parent:

您可以使用 `lv_obj_set_align(obj, LV_ALIGN_...)` 将对象参照其父对象对齐。此后，每个 x 和 y 设置都将适用于设置对齐模式。例如，这会将对象从其父对象的中心移动 10(x)，20(y) 像素：

```
lv_obj_set_align(obj, LV_ALIGN_CENTER);
lv_obj_set_pos(obj, 10, 20);

//Or in one function
lv_obj_align(obj, LV_ALIGN_CENTER, 10, 20);
```

To align one object to another use `lv_obj_align_to(obj_to_align, obj_referece, LV_ALIGN_..., x, y);`

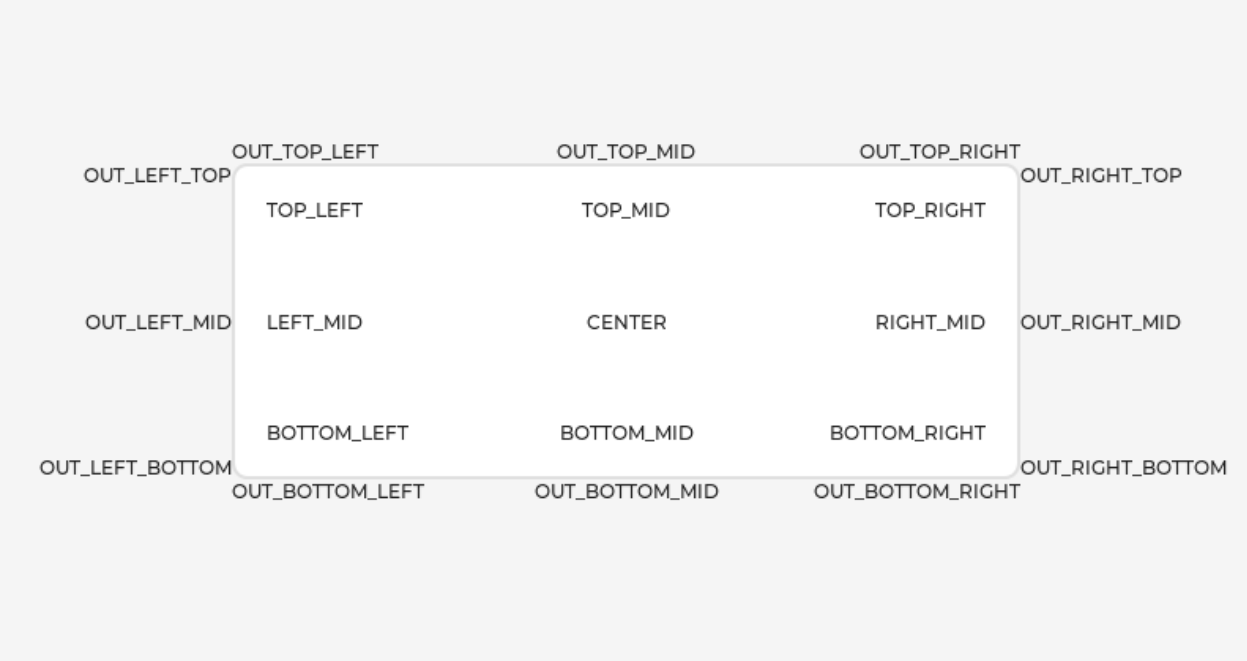
For example, to align a text below an image: `lv_obj_align_to(text, image, LV_ALIGN_OUT_BOTTOM_MID, 0, 10);`.

The following align types exist:

要将一个对象参照另一个对象对齐，请使用 `lv_obj_align_to(obj_to_align, obj_referece, LV_ALIGN_..., x, y);`

例如，让图片下方的文本参照图片对齐：`lv_obj_align_to(text, image, LV_ALIGN_OUT_BOTTOM_MID, 0, 10);`。

存在以下对齐类型：



Parents and children（父母和孩子）

You can set a new parent for an object with `lv_obj_set_parent(obj, new_parent);`. To get the current parent, use `lv_obj_get_parent(obj);`.

To get a specific children of a parent use `lv_obj_get_child(parent, idx);`. Some examples for `idx`:

- `0` get the child created first child
- `1` get the child created second
- `-1` get the child created last

The children can be iterated lke this

您可以使用 `lv_obj_set_parent(obj, new_parent);` 为对象设置新的父级。要获取当前父级，请使用 `lv_obj_get_parent(obj);`。

要获取父母的特定孩子，请使用 `lv_obj_get_child(parent, idx);`。`idx` 的一些示例：

- `0` 获取创建的第一个子项
- `1` 获取创建的第二个子项
- `-1` 获取最后创建的子项

父级的孩子们可以这样迭代

```
uint32_t i;
for(i = 0; i < lv_obj_get_child_cnt(parent); i++) {
  lv_obj_t * child = lv_obj_get_child(paernt, i);
```

(下页继续)

(续上页)

```
  /*Do something with child*/
}
```

`lv_obj_get_index(obj)` returns the index of the object in its parent. It is equivalent to the number of younger children in the parent.

You can bring an object to the foreground or send it to the background with `lv_obj_move_foreground(obj)` and `lv_obj_move_background(obj)`.

You can change the index of an object in its parent using `lv_obj_move_to_index(obj, index)`.

You can swap the position of two objects with `lv_obj_swap(obj1, obj2)`.

`lv_obj_get_index(obj);` 返回对象在其父对象中的索引。它相当于父对象拥有的孩子数量。

您可以使用 `lv_obj_move_foreground(obj);` 和 `lv_obj_move_background(obj);` 将对象带到前台或将其发移到后台。

您可以使用 `lv_obj_move_to_index(obj, index);` 更改对象在其父对象中的索引。

你可以用 `lv_obj_swap(obj1, obj2);` 交换两个对象的位置 (列表框中，它可用于对列表框项目进行排序。)。

Display and Screens(显示和屏幕)

At the highest level of the LVGL object hierarchy is the *display* which represents the driver for a display device (physical display or simulator). A display can have one or more screens associated with it. Each screen contains a hierarchy of objects for graphical widgets representing a layout that covers the entire display.

When you have created a screen like `lv_obj_t * screen = lv_obj_create(NULL)`, you can make it active with `lv_scr_load(screen)`. The `lv_scr_act()` function gives you a pointer to the active screen.

If you have multiple displays, it's important to know that the screen functions operate on the most recently created display or the one explicitly selected with `lv_disp_set_default`.

To get an object's screen use the `lv_obj_get_screen(obj)` function.

在 LVGL 中，对象层次结构的最高级别是 *display*，它代表显示设备（物理显示器或模拟器）的驱动程序。一个显示器可以有一个或多个与其相关联的屏幕。每个屏幕都包含图形组件的对象层次结构，代表覆盖整个显示的布局。

当你创建了一个像 `lv_obj_t * screen = lv_obj_create(NULL);` 这样的屏幕时，你可以用 `lv_scr_load(screen);` 激活它。`lv_scr_act()` 函数为您提供指向活动屏幕的指针。

如果您有多个显示器，重要的是要知道，最后创建的显示器时指定了屏幕，或者是用 `lv_disp_set_default` 明确切换的。

要获取对象的屏幕，请使用 `lv_obj_get_screen(obj);` 函数。

Events（事件）

To set an event callback for an object, use `lv_obj_add_event_cb(obj, event_cb, LV_EVENT_..., user_data)`,

To manually send an event to an object, use `lv_event_send(obj, LV_EVENT_..., param)`

Read the *Event overview* to learn more about events.

要为对象设置事件回调，请使用 `lv_obj_add_event_cb(obj, event_cb, LV_EVENT_..., user_data);`，

要手动向对象发送事件，请使用 `lv_event_send(obj, LV_EVENT_..., param);`

阅读事件概述 页面，以了解有关事件的更多信息。

Styles（样式）

Be sure to read the *Style overview*. Here only the most essential functions are described.

A new style can be added to an object with the `lv_obj_add_style(obj, &new_style, selector)` function. `selector` is an ORed combination of part and state(s). E.g. `LV_PART_SCROLLBAR | LV_STATE_PRESSED`.

The base objects use `LV_PART_MAIN` style properties and `LV_PART_SCROLLBAR` with the typical backgroud style properties.

这里只介绍最基本的功能，请务必阅读样式概述页面详细了解。

可以使用 `lv_obj_add_style(obj, &new_style, selector)` 函数向对象添加新样式。

`selector` 可以组合使用。例如。`LV_PART_SCROLLBAR | LV_STATE_PRESSED`。

基本对象使用 `LV_PART_MAIN` 样式属性和带有典型背景样式属性的 `LV_PART_SCROLLBAR` 。

Flags（宏开关）

There are some attributes which can be enabled/disabled by `lv_obj_add/clear_flag(obj, LV_OBJ_FLAG_...)`:

- `LV_OBJ_FLAG_HIDDEN` Make the object hidden. (Like it wasn't there at all)
- `LV_OBJ_FLAG_CLICKABLE` Make the object clickable by input devices
- `LV_OBJ_FLAG_CLICK_FOCUSABLE` Add focused state to the object when clicked
- `LV_OBJ_FLAG_CHECKABLE` Toggle checked state when the object is clicked
- `LV_OBJ_FLAG_SCROLLABLE` Make the object scrollable
- `LV_OBJ_FLAG_SCROLL_ELASTIC` Allow scrolling inside but with slower speed

- LV_OBJ_FLAG_SCROLL_MOMENTUM Make the object scroll further when "thrown"
- LV_OBJ_FLAG_SCROLL_ONE Allow scrolling only one snappable children
- LV_OBJ_FLAG_SCROLL_CHAIN Allow propagating the scroll to a parent
- LV_OBJ_FLAG_SCROLL_ON_FOCUS Automatically scroll object to make it visible when focused
- LV_OBJ_FLAG_SNAPPABLE If scroll snap is enabled on the parent it can snap to this object
- LV_OBJ_FLAG_PRESS_LOCK Keep the object pressed even if the press slid from the object
- LV_OBJ_FLAG_EVENT_BUBBLE Propagate the events to the parent too
- LV_OBJ_FLAG_GESTURE_BUBBLE Propagate the gestures to the parent
- LV_OBJ_FLAG_ADV_HITTEST Allow performing more accurate hit (click) test. E.g. accounting for rounded corners
- LV_OBJ_FLAG_IGNORE_LAYOUT Make the object positionable by the layouts
- LV_OBJ_FLAG_FLOATING Do not scroll the object when the parent scrolls and ignore layout
- LV_OBJ_FLAG_LAYOUT_1 Custom flag, free to use by layouts
- LV_OBJ_FLAG_LAYOUT_2 Custom flag, free to use by layouts
- LV_OBJ_FLAG_WIDGET_1 Custom flag, free to use by widget
- LV_OBJ_FLAG_WIDGET_2 Custom flag, free to use by widget
- LV_OBJ_FLAG_USER_1 Custom flag, free to use by user
- LV_OBJ_FLAG_USER_2 Custom flag, free to use by user
- LV_OBJ_FLAG_USER_3 Custom flag, free to use by user
- LV_OBJ_FLAG_USER_4 Custom flag, free to use by user

Some examples:

有一些属性可以通过 `lv_obj_add/clear_flag(obj, LV_OBJ_FLAG_...);` 启用/禁用：

- LV_OBJ_FLAG_HIDDEN 隐藏对象。（就像它根本不存在一样）
- LV_OBJ_FLAG_CLICKABLE 使输入设备可点击对象
- LV_OBJ_FLAG_CLICK_FOCUSABLE 单击时将焦点状态添加到对象
- LV_OBJ_FLAG_CHECKABLE 对象被点击时切换选中状态
- LV_OBJ_FLAG_SCROLLABLE 使对象可滚动
- LV_OBJ_FLAG_SCROLL_ELASTIC 允许在内部滚动但速度较慢
- LV_OBJ_FLAG_SCROLL_MOMENTUM 在“抛出”时使对象滚动得更远
- LV_OBJ_FLAG_SCROLL_ONE 只允许滚动一个可捕捉的孩子

- LV_OBJ_FLAG_SCROLL_CHAIN 允许将滚动传播到父级
- LV_OBJ_FLAG_SCROLL_ON_FOCUS 自动滚动对象以使其在聚焦时可见
- LV_OBJ_FLAG_SNAPPABLE 如果在父对象上启用了滚动捕捉，它可以捕捉到这个对象
- LV_OBJ_FLAG_PRESS_LOCK 保持对象被按下，即使按下从对象上滑动
- LV_OBJ_FLAG_EVENT_BUBBLE 也将事件传播给父级
- LV_OBJ_FLAG_GESTURE_BUBBLE 将手势传播给父级
- LV_OBJ_FLAG_ADV_HITTEST 允许执行更准确的命中（点击）测试。例如。考虑圆角。
- LV_OBJ_FLAG_IGNORE_LAYOUT 使对象可以通过布局定位
- LV_OBJ_FLAG_FLOATING 父滚动时不滚动对象，忽略布局
- LV_OBJ_FLAG_LAYOUT_1 自定义标志，可供布局免费使用
- LV_OBJ_FLAG_LAYOUT_2 自定义标志，可供布局免费使用
- LV_OBJ_FLAG_WIDGET_1 自定义标志，组件免费使用
- LV_OBJ_FLAG_WIDGET_2 自定义标志，组件免费使用
- LV_OBJ_FLAG_USER_1 自定义标志，用户免费使用
- LV_OBJ_FLAG_USER_2 自定义标志，用户免费使用
- LV_OBJ_FLAG_USER_3 自定义标志，用户免费使用
- LV_OBJ_FLAG_USER_4 自定义标志，由用户部分免费使用。

示例：

```
/*Hide on object*/
lv_obj_add_flag(obj, LV_OBJ_FLAG_HIDDEN);

/*Make an obejct non-clickable*/
lv_obj_clear_flag(obj, LV_OBJ_FLAG_CLICKABLE);
```

Groups

Read the *Input devices overview* to learn more about *Groups*.

Objects are added to a *group* with `lv_group_add_obj(group, obj)`, and you can use `lv_obj_get_group(obj)` to see which group an object belongs to.

`lv_obj_is_focused(obj)` returns if the object is currently focused on its group or not. If the object is not added to a group, `false` will be returned.

阅读输入设备概述 以了解有关 *Groups* 的更多信息。

使用 `lv_group_add_obj(group, obj);` 将对象添加到组，您可以使用 `lv_obj_get_group(obj);` 查看对象属于哪个组。

`lv_obj_is_focused(obj);` 返回对象当前是否聚焦在其组上。如果对象未添加到组中，则将返回 `false`。

Extended click area（拓展的点击区域）

By default, the objects can be clicked only on their coordinates, however, this area can be extended with `lv_obj_set_ext_click_area(obj, size)`.

默认情况下，只能在对象的坐标上单击对象，但是，可以使用 `lv_obj_set_ext_click_area(obj, size);` 扩展该区域。

Events（事件）

- `LV_EVENT_VALUE_CHANGED` when the `LV_OBJ_FLAG_CHECKABLE` flag is enabled and the object clicked (on transition to/from the checked state)
- `LV_EVENT_DRAW_PART_BEGIN` and `LV_EVENT_DRAW_PART_END` is sent for the following types:
  - `LV_OBJ_DRAW_PART_RECTANGLE` The main rectangle
    - `part`: `LV_PART_MAIN`
    - `rect_dsc`
    - `draw_area`: the area of the rectangle
  - `LV_OBJ_DRAW_PART_BORDER_POST` The border if the `border_post` style property is `true`
    - `part`: `LV_PART_MAIN`
    - `rect_dsc`
    - `draw_area`: the area of the rectangle
  - `LV_OBJ_DRAW_PART_SCROLLBAR` the scrollbars
    - `part`: `LV_PART_SCROLLBAR`
    - `rect_dsc`
    - `draw_area`: the area of the rectangle

Learn more about *Events*.

- `LV_EVENT_VALUE_CHANGED` 当 `LV_OBJ_FLAG_CHECKABLE` 标志被启用并且对象被点击时（转换到/从选中状态）
- 为以下类型发送 `LV_EVENT_DRAW_PART_BEGIN` 和 `LV_EVENT_DRAW_PART_END`:
  - `LV_OBJ_DRAW_PART_RECTANGLE` 主矩形

        * 部分：LV_PART_MAIN-rect_dsc
        * draw_area：矩形的面积
    - LV_OBJ_DRAW_PART_BORDER_POST 如果 border_post 样式属性为 true 的边框
        * 部分：LV_PART_MAIN-rect_dsc
        * draw_area：矩形的面积
    - LV_OBJ_DRAW_PART_SCROLLBAR 滚动条
        * 部分：LV_PART_SCROLLBAR-rect_dsc
        * draw_area：矩形的面积

详细了解事件。

Keys

If LV_OBJ_FLAG_CHECKABLE is enabled, LV_KEY_RIGHT and LV_KEY_UP make the object checked, and LV_KEY_LEFT and LV_KEY_DOWN make it unchecked.

If LV_OBJ_FLAG_SCROLLABLE is enabled, but the object is not editable (as declared by the widget class), the arrow keys (LV_KEY_UP, LV_KEY_DOWN, LV_KEY_LEFT, LV_KEY_RIGHT) scroll the object. If the object can only scroll vertically, LV_KEY_LEFT and LV_KEY_RIGHT will scroll up/down instead, making it compatible with an encoder input device. See *Input devices overview* for more on encoder behaviors and the edit mode.

如果启用了 LV_OBJ_FLAG_CHECKABLE，则 LV_KEY_RIGHT 和 LV_KEY_UP 使对象被选中，而 LV_KEY_LEFT 和 LV_KEY_DOWN 使其取消选中。

如果启用了 “LV_OBJ_FLAG_SCROLLABLE” ，但对象不可编辑（由组件类声明） ，则箭头键（“LV_KEY_UP”、“LV_KEY_DOWN”、“LV_KEY_LEFT”、“LV_KEY_RIGHT”）滚动对象。如果对象只能垂直滚动，LV_KEY_LEFT 和 LV_KEY_RIGHT 将改为向上/向下滚动，使其与编码器输入设备兼容。有关编码器行为和编辑模式的更多信息，请参阅输入设备概述。

了解有关*Keys* 的更多信息。

Example

Base objects with custom styles

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES

void lv_example_obj_1(void)
{
    lv_obj_t * obj1;
```

(下页继续)

(续上页)

```
    obj1 = lv_obj_create(lv_scr_act());
    lv_obj_set_size(obj1, 100, 50);
    lv_obj_align(obj1, LV_ALIGN_CENTER, -60, -30);

    static lv_style_t style_shadow;
    lv_style_init(&style_shadow);
    lv_style_set_shadow_width(&style_shadow, 10);
    lv_style_set_shadow_spread(&style_shadow, 5);
    lv_style_set_shadow_color(&style_shadow, lv_palette_main(LV_PALETTE_BLUE));

    lv_obj_t * obj2;
    obj2 = lv_obj_create(lv_scr_act());
    lv_obj_add_style(obj2, &style_shadow, 0);
    lv_obj_align(obj2, LV_ALIGN_CENTER, 60, 30);
}
#endif
```

```
obj1 = lv.obj(lv.scr_act())
obj1.set_size(100, 50)
obj1.align(lv.ALIGN.CENTER, -60, -30)

style_shadow = lv.style_t()
style_shadow.init()
style_shadow.set_shadow_width(10)
style_shadow.set_shadow_spread(5)
style_shadow.set_shadow_color(lv.palette_main(lv.PALETTE.BLUE))

obj2 = lv.obj(lv.scr_act())
obj2.add_style(style_shadow, 0)
obj2.align(lv.ALIGN.CENTER, 60, 30)
```

Make an object draggable

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES

static void drag_event_handler(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target(e);

    lv_indev_t * indev = lv_indev_get_act();
```

(下页继续)

(续上页)

```
    if(indev == NULL)  return;

    lv_point_t vect;
    lv_indev_get_vect(indev, &vect);

    lv_coord_t x = lv_obj_get_x(obj) + vect.x;
    lv_coord_t y = lv_obj_get_y(obj) + vect.y;
    lv_obj_set_pos(obj, x, y);
}


/**
 * Make an object dragable.
 */
void lv_example_obj_2(void)
{
    lv_obj_t * obj;
    obj = lv_obj_create(lv_scr_act());
    lv_obj_set_size(obj, 150, 100);
    lv_obj_add_event_cb(obj, drag_event_handler, LV_EVENT_PRESSING, NULL);

    lv_obj_t * label = lv_label_create(obj);
    lv_label_set_text(label, "Drag me");
    lv_obj_center(label);

}
#endif
```

```
def drag_event_handler(e):

    obj = e.get_target()

    indev = lv.indev_get_act()

    vect = lv.point_t()
    indev.get_vect(vect)
    x = obj.get_x() + vect.x
    y = obj.get_y() + vect.y
    obj.set_pos(x, y)


#
# Make an object dragable.
```

(下页继续)

(续上页)

```
#

obj = lv.obj(lv.scr_act())
obj.set_size(150, 100)
obj.add_event_cb(drag_event_handler, lv.EVENT.PRESSING, None)

label = lv.label(obj)
label.set_text("Drag me")
label.center()
```

API

Typedefs

typedef uint16_t **lv_state_t**

typedef uint32_t **lv_part_t**

typedef uint32_t **lv_obj_flag_t**

typedef struct *_lv_obj_t* **lv_obj_t**

Enums

enum **[anonymous]**

Possible states of a widget. OR-ed values are possible

Values:

enumerator **LV_STATE_DEFAULT**

enumerator **LV_STATE_CHECKED**

enumerator **LV_STATE_FOCUSED**

enumerator **LV_STATE_FOCUS_KEY**

enumerator **LV_STATE_EDITED**

enumerator **LV_STATE_HOVERED**

enumerator **LV_STATE_PRESSED**

enumerator **LV_STATE_SCROLLED**

enumerator **LV_STATE_DISABLED**

enumerator **LV_STATE_USER_1**

enumerator **LV_STATE_USER_2**

enumerator **LV_STATE_USER_3**

enumerator **LV_STATE_USER_4**

enumerator **LV_STATE_ANY**
Special value can be used in some functions to target all states

enum **[anonymous]**
The possible parts of widgets. The parts can be considered as the internal building block of the widgets. E.g. slider = background + indicator + knob Note every part is used by every widget

Values:

enumerator **LV_PART_MAIN**
A background like rectangle

enumerator **LV_PART_SCROLLBAR**
The scrollbar(s)

enumerator **LV_PART_INDICATOR**
Indicator, e.g. for slider, bar, switch, or the tick box of the checkbox

enumerator **LV_PART_KNOB**
Like handle to grab to adjust the value

enumerator **LV_PART_SELECTED**
Indicate the currently selected option or section

enumerator **LV_PART_ITEMS**
Used if the widget has multiple similar elements (e.g. table cells)

enumerator **LV_PART_TICKS**
Ticks on scale e.g. for a chart or meter

enumerator **LV_PART_CURSOR**
Mark a specific place e.g. for text area's cursor or on a chart

enumerator **LV_PART_CUSTOM_FIRST**
Extension point for custom widgets

enumerator **LV_PART_ANY**
Special value can be used in some functions to target all parts

enum **[anonymous]**
On/Off features controlling the object's behavior. OR-ed values are possible

Values:

enumerator **LV_OBJ_FLAG_HIDDEN**
Make the object hidden. (Like it wasn't there at all)

enumerator **LV_OBJ_FLAG_CLICKABLE**
Make the object clickable by the input devices

enumerator **LV_OBJ_FLAG_CLICK_FOCUSABLE**
Add focused state to the object when clicked

enumerator **LV_OBJ_FLAG_CHECKABLE**
Toggle checked state when the object is clicked

enumerator **LV_OBJ_FLAG_SCROLLABLE**
Make the object scrollable

enumerator **LV_OBJ_FLAG_SCROLL_ELASTIC**
Allow scrolling inside but with slower speed

enumerator **LV_OBJ_FLAG_SCROLL_MOMENTUM**
Make the object scroll further when "thrown"

enumerator **LV_OBJ_FLAG_SCROLL_ONE**
Allow scrolling only one snappable children

enumerator **LV_OBJ_FLAG_SCROLL_CHAIN_HOR**
Allow propagating the horizontal scroll to a parent

enumerator **LV_OBJ_FLAG_SCROLL_CHAIN_VER**
Allow propagating the vertical scroll to a parent

enumerator **LV_OBJ_FLAG_SCROLL_CHAIN**

enumerator **LV_OBJ_FLAG_SCROLL_ON_FOCUS**
Automatically scroll object to make it visible when focused

enumerator **LV_OBJ_FLAG_SCROLL_WITH_ARROW**
Allow scrolling the focused object with arrow keys

enumerator **LV_OBJ_FLAG_SNAPPABLE**
If scroll snap is enabled on the parent it can snap to this object

enumerator **LV_OBJ_FLAG_PRESS_LOCK**
Keep the object pressed even if the press slid from the object

enumerator **LV_OBJ_FLAG_EVENT_BUBBLE**
Propagate the events to the parent too

enumerator **LV_OBJ_FLAG_GESTURE_BUBBLE**
Propagate the gestures to the parent

enumerator **LV_OBJ_FLAG_ADV_HITTEST**
Allow performing more accurate hit (click) test. E.g. consider rounded corners.

enumerator **LV_OBJ_FLAG_IGNORE_LAYOUT**
Make the object position-able by the layouts

enumerator **LV_OBJ_FLAG_FLOATING**
Do not scroll the object when the parent scrolls and ignore layout

enumerator **LV_OBJ_FLAG_OVERFLOW_VISIBLE**
Do not clip the children's content to the parent's boundary

enumerator **LV_OBJ_FLAG_LAYOUT_1**
Custom flag, free to use by layouts

enumerator **LV_OBJ_FLAG_LAYOUT_2**
Custom flag, free to use by layouts

enumerator **LV_OBJ_FLAG_WIDGET_1**
Custom flag, free to use by widget

enumerator **LV_OBJ_FLAG_WIDGET_2**
Custom flag, free to use by widget

enumerator **LV_OBJ_FLAG_USER_1**
Custom flag, free to use by user

enumerator **LV_OBJ_FLAG_USER_2**
Custom flag, free to use by user

enumerator **LV_OBJ_FLAG_USER_3**
Custom flag, free to use by user

enumerator **LV_OBJ_FLAG_USER_4**
Custom flag, free to use by user

enum **lv_obj_draw_part_type_t**
`type` field in `lv_obj_draw_part_dsc_t` if `class_p = lv_obj_class` Used in `LV_EVENT_DRAW_PART_BEGIN` and `LV_EVENT_DRAW_PART_END`

Values:

enumerator **LV_OBJ_DRAW_PART_RECTANGLE**
The main rectangle

enumerator **LV_OBJ_DRAW_PART_BORDER_POST**
The border if style_border_post = true

enumerator **LV_OBJ_DRAW_PART_SCROLLBAR**
The scrollbar

Functions

void **lv_init**(void)
Initialize LVGL library. Should be called before any other LVGL related function.

void **lv_deinit**(void)
Deinit the 'lv' library Currently only implemented when not using custom allocators, or GC is enabled.

bool **lv_is_initialized**(void)
Returns whether the 'lv' library is currently initialized

lv_obj_t ***lv_obj_create**(*lv_obj_t* *parent)
Create a base object (a rectangle)

参数 **parent** -- pointer to a parent object. If NULL then a screen will be created.

返回 pointer to the new object

void **lv_obj_add_flag**(*lv_obj_t* *obj, *lv_obj_flag_t* f)
Set one or more flags

参数

- **obj** -- pointer to an object
- **f** -- R-ed values from `lv_obj_flag_t` to set.

void **lv_obj_clear_flag**(*lv_obj_t* *obj, *lv_obj_flag_t* f)

Clear one or more flags

参数

- **obj** -- pointer to an object
- **f** -- OR-ed values from `lv_obj_flag_t` to set.

void **lv_obj_add_state**(*lv_obj_t* *obj, *lv_state_t* state)

Add one or more states to the object. The other state bits will remain unchanged. If specified in the styles, transition animation will be started from the previous state to the current.

参数

- **obj** -- pointer to an object
- **state** -- the states to add. E.g `LV_STATE_PRESSED | LV_STATE_FOCUSED`

void **lv_obj_clear_state**(*lv_obj_t* *obj, *lv_state_t* state)

Remove one or more states to the object. The other state bits will remain unchanged. If specified in the styles, transition animation will be started from the previous state to the current.

参数

- **obj** -- pointer to an object
- **state** -- the states to add. E.g `LV_STATE_PRESSED | LV_STATE_FOCUSED`

static inline void **lv_obj_set_user_data**(*lv_obj_t* *obj, void *user_data)

Set the user_data field of the object

参数

- **obj** -- pointer to an object
- **user_data** -- pointer to the new user_data.

bool **lv_obj_has_flag**(const *lv_obj_t* *obj, *lv_obj_flag_t* f)

Check if a given flag or all the given flags are set on an object.

参数

- **obj** -- pointer to an object
- **f** -- the flag(s) to check (OR-ed values can be used)

返回 true: all flags are set; false: not all flags are set

bool **lv_obj_has_flag_any**(const *lv_obj_t* *obj, *lv_obj_flag_t* f)

Check if a given flag or any of the flags are set on an object.

参数

- **obj** -- pointer to an object
- **f** -- the flag(s) to check (OR-ed values can be used)

返回 true: at lest one flag flag is set; false: none of the flags are set

lv_state_t **lv_obj_get_state**(const *lv_obj_t* *obj)

Get the state of an object

参数 **obj** -- pointer to an object

返回 the state (OR-ed values from `lv_state_t`)

bool **lv_obj_has_state**(const *lv_obj_t* *obj, *lv_state_t* state)

Check if the object is in a given state or not.

参数

- **obj** -- pointer to an object
- **state** -- a state or combination of states to check

返回 true: `obj` is in `state`; false: `obj` is not in `state`

void ***lv_obj_get_group**(const *lv_obj_t* *obj)

Get the group of the object

参数 **obj** -- pointer to an object

返回 the pointer to group of the object

static inline void ***lv_obj_get_user_data**(*lv_obj_t* *obj)

Get the user_data field of the object

参数 **obj** -- pointer to an object

返回 the pointer to the user_data of the object

void **lv_obj_allocate_spec_attr**(*lv_obj_t* *obj)

Allocate special data for an object if not allocated yet.

参数 **obj** -- pointer to an object

bool **lv_obj_check_type**(const *lv_obj_t* *obj, const lv_obj_class_t *class_p)

Check the type of obj.

参数

- **obj** -- pointer to an object
- **class_p** -- a class to check (e.g. `lv_slider_class`)

返回 true: `class_p` is the `obj` class.

bool **lv_obj_has_class**(const *lv_obj_t* *obj, const lv_obj_class_t *class_p)

Check if any object has a given class (type). It checks the ancestor classes too.

参数

- **obj** -- pointer to an object
- **class_p** -- a class to check (e.g. `lv_slider_class`)

返回 true: `obj` has the given class

const lv_obj_class_t ***lv_obj_get_class**(const *lv_obj_t* *obj)

Get the class (type) of the object

参数 **obj** -- pointer to an object

返回 the class (type) of the object

bool **lv_obj_is_valid**(const *lv_obj_t* *obj)

Check if any object is still "alive".

参数 **obj** -- pointer to an object

返回 true: valid

static inline lv_coord_t **lv_obj_dpx**(const *lv_obj_t* *obj, lv_coord_t n)

Scale the given number of pixels (a distance or size) relative to a 160 DPI display considering the DPI of the `obj`'s display. It ensures that e.g. `lv_dpx(100)` will have the same physical size regardless to the DPI of the display.

参数

- **obj** -- an object whose display's dpi should be considered
- **n** -- the number of pixels to scale

返回 `n x current_dpi/160`

Variables

const lv_obj_class_t **lv_obj_class**

Make the base object's class publicly available.

struct **_lv_obj_spec_attr_t**

#include <lv_obj.h> Special, rarely used attributes. They are allocated automatically if any elements is set.

Public Members

struct *_lv_obj_t* ****children**
Store the pointer of the children in an array.

uint32_t **child_cnt**
Number of children

lv_group_t ***group_p**

struct _lv_event_dsc_t ***event_dsc**
Dynamically allocated event callback and user data array

lv_point_t **scroll**
The current X/Y scroll offset

lv_coord_t **ext_click_pad**
Extra click padding in all direction

lv_coord_t **ext_draw_size**
EXTend the size in every direction for drawing.

lv_scrollbar_mode_t **scrollbar_mode**
How to display scrollbars

lv_scroll_snap_t **scroll_snap_x**
Where to align the snappable children horizontally

lv_scroll_snap_t **scroll_snap_y**
Where to align the snappable children vertically

lv_dir_t **scroll_dir**
The allowed scroll direction(s)

uint8_t **event_dsc_cnt**
Number of event callbacks stored in `event_dsc` array

struct **_lv_obj_t**

Public Members

const lv_obj_class_t ***class_p**

struct *_lv_obj_t* ***parent**

_lv_obj_spec_attr_t ***spec_attr**

_lv_obj_style_t ***styles**

void ***user_data**

lv_area_t **coords**

lv_obj_flag_t **flags**

lv_state_t **state**

uint16_t **layout_inv**

uint16_t **scr_layout_inv**

uint16_t **skip_trans**

uint16_t **style_cnt**

uint16_t **h_layout**

uint16_t **w_layout**

2.6.2 Core widgets（核心组件）

Arc（圆弧）(lv_arc)

Overview（概述）

The Arc consists of a background and a foreground arc. The foregrond (indicator) can be touch-adjusted.

圆弧由背景和前景弧组成。前景（指示器）可以进行触摸调整。

Parts and Styles

- `LV_PART_MAIN` Draws a background using the typical background style properties and an arc using the arc style properties. The arc's size and position will respect the *padding* style properties.
- `LV_PART_INDICATOR` Draws an other arc using the *arc* style properties. Its padding values are interpreted relative to the background arc.

- `LV_PART_KNOB` Draws a handle on the end of the indicator using all background properties and padding values. With zero padding the knob size is the same as the indicator's width. Larger padding makes it larger, smaller padding makes it smaller.
- `LV_PART_MAIN` 圆弧的背景部分。使用典型的背景样式属性绘制背景，使用圆弧样式属性绘制圆弧。圆弧的大小和位置可以通过 *padding* 样式调整。
- `LV_PART_INDICATOR` 圆弧的指示器部分。使用 *arc* 样式属性绘制另一个圆弧。它的填充值是相对于背景弧来设置的。
- `LV_PART_KNOB` 圆弧的旋钮部分。使用所有背景属性和填充值在指标的末尾绘制一个旋钮。如果使用零填充，旋钮大小与指示器的宽度相同。较大的填充使其更大，较小的填充使其更小。

Usage（用法）

Value and range（值和范围）

A new value can be set using `lv_arc_set_value(arc, new_value)`. The value is interpreted in a range (minimum and maximum values) which can be modified with `lv_arc_set_range(arc, min, max)`. The default range is 0..100.

The indicator arc is drawn on the main part's arc. This if the value is set to maximum the indicator arc will cover the entire "background" arc. To set the start and end angle of the background arc use the `lv_arc_set_bg_angles(arc, start_angle, end_angle)` functions or `lv_arc_set_bg_start/end_angle(arc, angle)`.

Zero degrees is at the middle right (3 o'clock) of the object and the degrees are increasing in clockwise direction. The angles should be in the [0;360] range.

可以通过 `lv_arc_set_value(arc, new_value)` 设置一个数值。

可以通过 `lv_arc_set_range(arc, min, max)` 指定数值的范围（最小值和最大值）。如果没有指定数值范围，将使用默认范围 (0-100)。

指示器绘制在 LV_PART_MAIN 部分的弧上面。如果指示器的值设置为最大值，则指示器弧将呈现一个闭合的弧 (也就是圆形)。要设置背景弧的开始和结束角度，请使用 `lv_arc_set_bg_angles(arc, start_angle, end_angle)` 函数或 `lv_arc_set_bg_start/end_angle(arc, angle)`。

零度位于对象的中间右侧（3 点钟方向），并且度数沿顺时针方向增加。角度应在 [0;360] 范围内。

Rotation（旋转）

An offset to the 0 degree position can added with `lv_arc_set_rotation(arc, deg)`.

可以使用 `lv_arc_set_rotation(arc, deg)` 添加到 0 度位置的偏移量。

Mode（模式）

The arc can be one of the following modes:

- `LV_ARC_MODE_NORMAL` The indicator arc is drawn from the minimimum value to the current.
- `LV_ARC_MODE_REVERSE` The indicator arc is drawn counter-clockwise from the maximum value to the current.
- `LV_ARC_MODE_SYMMETRICAL` The indicator arc is drawn from the middle point to the current value.

The mode can be set by `lv_arc_set_mode(arc, LV_ARC_MODE_...)` and used only if the the angle is set by `lv_arc_set_value()` or the arc is adjusted by finger.

弧可以是以下模式之一：

- `LV_ARC_MODE_NORMAL` 普通模式。指示器从最小值绘制到当前值。
- `LV_ARC_MODE_REVERSE` 反向模式。指示器从最大值到当前值逆时针绘制。
- `LV_ARC_MODE_SYMMETRICAL` 对称模式。指示器从中间点绘制到当前值。

可以通过 `lv_arc_set_mode(arc, LV_ARC_MODE_...)` 设置模式，并且仅当角度由 `lv_arc_set_value()` 设置或通过手指调整弧度时使用。

Change rate（变化率）

If the arc is pressed the current value will set with a limited speed according to the set *change rate*. The change rate is defined in degree/second unit and can be set with `lv_arc_set_change_rage(arc, rate)`

如果弧被按下，当前值将根据设置的 变化率以有限的速度设置。变化率以度/秒为单位定义，可以用 `lv_arc_set_change_rage(arc, rate)` 设置

Setting the indicator manually（手动设置指示器）

It also possible to set the angles of the indicator arc directly with `lv_arc_set_angles(arc, start_angle, end_angle)` function or `lv_arc_set_start/end_angle(arc, start_angle)`. In this case the set "value" and "mode" is ignored.

In other words, the angle and value settings are independent. You should exclusively use one or the other. Mixing the two might result in unintended behavior.

To make the arc non-adjustabe remove the style of the knob and make the object non-clickable:

也可以直接使用 `lv_arc_set_angles(arc, start_angle, end_angle)` 函数或 `lv_arc_set_start/end_angle(arc, start_angle)` 设置指示器的角度（零度位于对象的中间右侧（3 点钟方向），并且度数沿顺时针方向增加。）。在这种情况下，设置的“值”和“模式”将被忽略。

换句话说，角度和值的设置是独立的。两者混合可能会导致意外行为。比如：

```
static void arc_event_cb(lv_event_t * e)
{
    lv_obj_t * arc = lv_event_get_target(e);

        lv_arc_set_bg_angles(arc, 0, lv_arc_get_value(arc));
    lv_arc_set_angles(arc, 0, lv_arc_get_value(arc));
}

.....
lv_obj_add_event_cb(arc, arc_event_cb, LV_EVENT_VALUE_CHANGED, NULL);
.....
```

要使圆弧不可调整，请移除旋钮的样式并使对象不可点击：

```
lv_obj_remove_style(arc, NULL, LV_PART_KNOB);
lv_obj_clear_flag(arc, LV_OBJ_FLAG_CLICKABLE);
```

Events（事件）

- `LV_EVENT_VALUE_CHANGED` sent when the arc is pressed/dragged to set a new value.
- `LV_EVENT_DRAW_PART_BEGIN` and `LV_EVENT_DRAW_PART_END` are sent with the following types:
  - `LV_ARC_DRAW_PART_BACKGROUND` The background arc.
    - `part`: `LV_PART_MAIN`
    - `p1`: center of the arc
    - `radius`: radius of the arc
    - `arc_dsc`
  - `LV_ARC_DRAW_PART_FOREGROUND` The foreground arc.
    - `part`: `LV_PART_INDICATOR`
    - `p1`: center of the arc
    - `radius`: radius of the arc
    - `arc_dsc`
  - LV_ARC_DRAW_PART_KNOB The knob

        * part: LV_PART_KNOB
        * draw_area: the arae of the knob
        * rect_dsc:

See the events of the *Base object* too.

Learn more about *Events*.

- 按下或拖动圆弧设置新值时发送会 LV_EVENT_VALUE_CHANGED
- LV_EVENT_DRAW_PART_BEGIN 和 LV_EVENT_DRAW_PART_END 使用以下类型发送：
  - LV_ARC_DRAW_PART_BACKGROUND 背景弧。
    * 部分：LV_PART_MAIN
    * p1：圆弧的中心
    * radius：弧的半径
    * arc_dsc
  - LV_ARC_DRAW_PART_FOREGROUND 前景弧。
    * 部分：LV_PART_INDICATOR
    * p1：圆弧的中心
    * radius：弧的半径
    * arc_dsc
  - LV_ARC_DRAW_PART_KNOB 旋钮
    * 部分：LV_PART_KNOB
    * draw_area：旋钮的面积 -rect_dsc：

参见*Base object* 的事件。

详细了解事件。

Keys（按键）

- LV_KEY_RIGHT/UP Increases the value by one.
- LV_KEY_LEFT/DOWN Decreases the value by one.

Learn more about *Keys*.

- LV_KEY_RIGHT/UP 将值增加一。
- LV_KEY_LEFT/DOWN 将值减一。

了解有关*Keys* 的更多信息。

Example

Simple Arc

```
#include "../../lv_examples.h"

#if LV_USE_ARC && LV_BUILD_EXAMPLES

void lv_example_arc_1(void)
{
  /*Create an Arc*/
  lv_obj_t * arc = lv_arc_create(lv_scr_act());
  lv_obj_set_size(arc, 150, 150);
  lv_arc_set_rotation(arc, 135);
  lv_arc_set_bg_angles(arc, 0, 270);
  lv_arc_set_value(arc, 40);
  lv_obj_center(arc);
}

#endif
```

```
# Create an Arc
arc = lv.arc(lv.scr_act())
arc.set_end_angle(200)
arc.set_size(150, 150)
arc.center()
```

Loader with Arc

```
#include "../../lv_examples.h"

#if LV_USE_ARC && LV_BUILD_EXAMPLES

static void set_angle(void * obj, int32_t v)
{
    lv_arc_set_value(obj, v);
}

/**
```

(下页继续)

(续上页)

```
 * Create an arc which acts as a loader.
 */
void lv_example_arc_2(void)
{
  /*Create an Arc*/
  lv_obj_t * arc = lv_arc_create(lv_scr_act());
  lv_arc_set_rotation(arc, 270);
  lv_arc_set_bg_angles(arc, 0, 360);
  lv_obj_remove_style(arc, NULL, LV_PART_KNOB);   /*Be sure the knob is not␣
↪displayed*/
  lv_obj_clear_flag(arc, LV_OBJ_FLAG_CLICKABLE);  /*To not allow adjusting by click*/
  lv_obj_center(arc);

  lv_anim_t a;
  lv_anim_init(&a);
  lv_anim_set_var(&a, arc);
  lv_anim_set_exec_cb(&a, set_angle);
  lv_anim_set_time(&a, 1000);
  lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);    /*Just for the demo*/
  lv_anim_set_repeat_delay(&a, 500);
  lv_anim_set_values(&a, 0, 100);
  lv_anim_start(&a);



}

#endif
```

```
#
# An `lv_timer` to call periodically to set the angles of the arc
#
class ArcLoader():
    def __init__(self):
        self.a = 270

    def arc_loader_cb(self,tim,arc):
        # print(tim,arc)
        self.a += 5

        arc.set_end_angle(self.a)

        if self.a >= 270 + 360:
```

(下页继续)

(续上页)

```
            tim._del()


#
# Create an arc which acts as a loader.
#

# Create an Arc
arc = lv.arc(lv.scr_act())
arc.set_bg_angles(0, 360)
arc.set_angles(270, 270)
arc.center()

# create the loader
arc_loader = ArcLoader()

# Create an `lv_timer` to update the arc.

timer = lv.timer_create_basic()
timer.set_period(20)
timer.set_cb(lambda src: arc_loader.arc_loader_cb(timer,arc))
```

API

Typedefs

typedef uint8_t **lv_arc_mode_t**

Enums

enum **[anonymous]**

Values:

enumerator **LV_ARC_MODE_NORMAL**

enumerator **LV_ARC_MODE_SYMMETRICAL**

enumerator **LV_ARC_MODE_REVERSE**

enum **lv_arc_draw_part_type_t**

`type` field in `lv_obj_draw_part_dsc_t` if `class_p = lv_arc_class` Used in `LV_EVENT_DRAW_PART_BEGIN` and `LV_EVENT_DRAW_PART_END`

Values:

enumerator **LV_ARC_DRAW_PART_BACKGROUND**

The background arc

enumerator **LV_ARC_DRAW_PART_FOREGROUND**

The foreground arc

enumerator **LV_ARC_DRAW_PART_KNOB**

The knob

Functions

lv_obj_t ***lv_arc_create**(*lv_obj_t* *parent)

Create an arc object

参数 **parent** -- pointer to an object, it will be the parent of the new arc

返回 pointer to the created arc

void **lv_arc_set_start_angle**(*lv_obj_t* *arc, uint16_t start)

Set the start angle of an arc. 0 deg: right, 90 bottom, etc.

参数

- **arc** -- pointer to an arc object
- **start** -- the start angle

void **lv_arc_set_end_angle**(*lv_obj_t* *arc, uint16_t end)

Set the end angle of an arc. 0 deg: right, 90 bottom, etc.

参数

- **arc** -- pointer to an arc object
- **end** -- the end angle

void **lv_arc_set_angles**(*lv_obj_t* *arc, uint16_t start, uint16_t end)

Set the start and end angles

参数

- **arc** -- pointer to an arc object
- **start** -- the start angle

- **end** -- the end angle

void **lv_arc_set_bg_start_angle**(*lv_obj_t* *arc, uint16_t start)

Set the start angle of an arc background. 0 deg: right, 90 bottom, etc.

参数

- **arc** -- pointer to an arc object
- **start** -- the start angle

void **lv_arc_set_bg_end_angle**(*lv_obj_t* *arc, uint16_t end)

Set the start angle of an arc background. 0 deg: right, 90 bottom etc.

参数

- **arc** -- pointer to an arc object
- **end** -- the end angle

void **lv_arc_set_bg_angles**(*lv_obj_t* *arc, uint16_t start, uint16_t end)

Set the start and end angles of the arc background

参数

- **arc** -- pointer to an arc object
- **start** -- the start angle
- **end** -- the end angle

void **lv_arc_set_rotation**(*lv_obj_t* *arc, uint16_t rotation)

Set the rotation for the whole arc

参数

- **arc** -- pointer to an arc object
- **rotation** -- rotation angle

void **lv_arc_set_mode**(*lv_obj_t* *arc, *lv_arc_mode_t* type)

Set the type of arc.

参数

- **arc** -- pointer to arc object
- **mode** -- arc's mode

void **lv_arc_set_value**(*lv_obj_t* *arc, int16_t value)

Set a new value on the arc

参数

- **arc** -- pointer to an arc object
- **value** -- new value

void **lv_arc_set_range**(*lv_obj_t* *arc, int16_t min, int16_t max)

Set minimum and the maximum values of an arc

参数

- **arc** -- pointer to the arc object
- **min** -- minimum value
- **max** -- maximum value

void **lv_arc_set_change_rate**(*lv_obj_t* *arc, uint16_t rate)

Set a change rate to limit the speed how fast the arc should reach the pressed point.

参数

- **arc** -- pointer to an arc object
- **rate** -- the change rate

uint16_t **lv_arc_get_angle_start**(*lv_obj_t* *obj)

Get the start angle of an arc.

参数 **arc** -- pointer to an arc object

返回 the start angle [0..360]

uint16_t **lv_arc_get_angle_end**(*lv_obj_t* *obj)

Get the end angle of an arc.

参数 **arc** -- pointer to an arc object

返回 the end angle [0..360]

uint16_t **lv_arc_get_bg_angle_start**(*lv_obj_t* *obj)

Get the start angle of an arc background.

参数 **arc** -- pointer to an arc object

返回 the start angle [0..360]

uint16_t **lv_arc_get_bg_angle_end**(*lv_obj_t* *obj)

Get the end angle of an arc background.

参数 **arc** -- pointer to an arc object

返回 the end angle [0..360]

int16_t **lv_arc_get_value**(const *lv_obj_t* *obj)

Get the value of an arc

参数 **arc** -- pointer to an arc object

返回 the value of the arc

int16_t **lv_arc_get_min_value**(const *lv_obj_t* *obj)

Get the minimum value of an arc

参数 **arc** -- pointer to an arc object

返回 the minimum value of the arc

int16_t **lv_arc_get_max_value**(const *lv_obj_t* *obj)

Get the maximum value of an arc

参数 **arc** -- pointer to an arc object

返回 the maximum value of the arc

lv_arc_mode_t **lv_arc_get_mode**(const *lv_obj_t* *obj)

Get whether the arc is type or not.

参数 **arc** -- pointer to an arc object

返回 arc's mode

Variables

const lv_obj_class_t **lv_arc_class**

struct **lv_arc_t**

Public Members

lv_obj_t **obj**

uint16_t **rotation**

uint16_t **indic_angle_start**

uint16_t **indic_angle_end**

uint16_t **bg_angle_start**

uint16_t **bg_angle_end**

int16_t **value**

int16_t **min_value**

int16_t **max_value**

uint16_t **dragging**

uint16_t **type**

uint16_t **min_close**

uint16_t **chg_rate**

uint32_t **last_tick**

int16_t **last_angle**

Bar（进度条）(lv_bar)

Overview（概述）

The bar object has a background and an indicator on it. The width of the indicator is set according to the current value of the bar.

Vertical bars can be created if the width of the object is smaller than its height.

Not only the end, but also the start value of the bar can be set, which changes the start position of the indicator.

进度条对象有一个背景和一个指示器。指示器的宽度根据进度条的当前值自动设置。

如果设置进度条的宽度小于其高度，就可以创建出垂直摆放的进度条。

可以通过设置进度条的值，从而改变指标的开始位置。

Parts and Styles（零件和样式）

- `LV_PART_MAIN` The background of the bar and it uses the typical background style properties. Adding padding makes the indicator smaller or larger. The `anim_time` style property sets the animation time if the values set with `LV_ANIM_ON`.
- `LV_PART_INDICATOR` The indicator itself; also also uses all the typical background properties.
- `LV_PART_MAIN` 进度条的背景，它使用典型的背景样式属性。添加填充会让指示器变小或变大。如果值设置为 `LV_ANIM_ON` ，则可以继续通过设置 `anim_time` 样式属性设置动画的时间。
- `LV_PART_INDICATOR` 指示器本身；也是使用所有典型的背景属性。

Usage（用法）

Value and range（值和范围）

A new value can be set by `lv_bar_set_value(bar, new_value, LV_ANIM_ON/OFF)`. The value is interpreted in a range (minimum and maximum values) which can be modified with `lv_bar_set_range(bar, min, max)`. The default range is 0..100.

The new value in `lv_bar_set_value` can be set with or without an animation depending on the last parameter (`LV_ANIM_ON/OFF`).

可以通过 `lv_bar_set_value(bar, new_value, LV_ANIM_ON/OFF)` 设置新值。该值在可以使用 `lv_bar_set_range(bar, min, max)` 修改的范围（最小值和最大值）内进行解释。进度条默认的范围是 0..100。

使用 `lv_bar_set_value` 设置的新值可以通过其最后的一个参数 (`LV_ANIM_ON/OFF`) 设置生效新值时的过渡动画。

Modes（模式）

The bar can be one the following modes:

- `LV_BAR_MODE_NORMAL` A normal bar as described above
- `LV_BAR_MODE_SYMMETRICAL` Draw the indicator from the zero value to current value. Requires a negative minimum range and positive maximum range.
- `LV_BAR_MODE_RANGE` Allows setting the start value too by `lv_bar_set_start_value(bar, new_value, LV_ANIM_ON/OFF)`. The start value always has to be smaller than the end value.

进度条可以是以下模式之一：

- `LV_BAR_MODE_NORMAL` 像上文所说的普通进度条 (默认)
- `LV_BAR_MODE_SYMMETRICAL` 这个模式下可以指定负的最小范围。但是只能从零值到当前值绘制指示器。
- `LV_BAR_MODE_RANGE` 这个模式下也可以指定负的最小范围。这样进度条的起始值可以不是 0，而是你指定的数值 (*参考示例*)，但是这样设置的起始值 (lv_bar_set_start_value) 必须小于结束值 (lv_bar_set_value)。

Events（事件）

- `LV_EVENT_DRAW_PART_BEGIN` and `LV_EVENT_DRAW_PART_END` are sent for the following parts:
  - `LV_BAR_DRAW_PART_INDICATOR` The indicator of the bar
    - `part`: `LV_PART_INDICATOR`
    - `draw_area`: area of the indicator
    - `rect_dsc`

See the events of the *Base object* too.

Learn more about *Events*.

- `LV_EVENT_DRAW_PART_BEGIN` 和 `LV_EVENT_DRAW_PART_END` 会因为下面这些部分而触发：
  - `LV_BAR_DRAW_PART_INDICATOR` 进度条的指示器
    - 部分：`LV_PART_INDICATOR`

* draw_area：指示器的区域
* rect_dsc

也就是说我们可以继续通过 LV_EVENT_DRAW_PART_BEGIN 和 LV_EVENT_DRAW_PART_END 修改进度条的样式。

参见*Base object* 的事件。

详细了解事件。

Keys（按键）

No *Keys* are processed by the object type.

Learn more about *Keys*.

进度条不处理 *Keys*， 也就是不能通过触摸或者按键控制进度条。

了解有关*Keys* 的更多信息。

Example

Simple Bar

```
#include "../../lv_examples.h"
#if LV_USE_BAR && LV_BUILD_EXAMPLES

void lv_example_bar_1(void)
{
    lv_obj_t * bar1 = lv_bar_create(lv_scr_act());
    lv_obj_set_size(bar1, 200, 20);
    lv_obj_center(bar1);
    lv_bar_set_value(bar1, 70, LV_ANIM_OFF);
}

#endif
```

```
bar1 = lv.bar(lv.scr_act())
bar1.set_size(200, 20)
bar1.center()
bar1.set_value(70, lv.ANIM.OFF)
```

Styling a bar

```c
#include "../../lv_examples.h"
#if LV_USE_BAR && LV_BUILD_EXAMPLES

/**
 * Example of styling the bar
 */
void lv_example_bar_2(void)
{
    static lv_style_t style_bg;
    static lv_style_t style_indic;

    lv_style_init(&style_bg);
    lv_style_set_border_color(&style_bg, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_border_width(&style_bg, 2);
    lv_style_set_pad_all(&style_bg, 6); /*To make the indicator smaller*/
    lv_style_set_radius(&style_bg, 6);
    lv_style_set_anim_time(&style_bg, 1000);

    lv_style_init(&style_indic);
    lv_style_set_bg_opa(&style_indic, LV_OPA_COVER);
    lv_style_set_bg_color(&style_indic, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_radius(&style_indic, 3);

    lv_obj_t * bar = lv_bar_create(lv_scr_act());
    lv_obj_remove_style_all(bar);  /*To have a clean start*/
    lv_obj_add_style(bar, &style_bg, 0);
    lv_obj_add_style(bar, &style_indic, LV_PART_INDICATOR);

    lv_obj_set_size(bar, 200, 20);
    lv_obj_center(bar);
    lv_bar_set_value(bar, 100, LV_ANIM_ON);
}

#endif
```

```python
#
# Example of styling the bar
#
style_bg = lv.style_t()
style_indic = lv.style_t()

style_bg.init()
```

(下页继续)

(续上页)

```
style_bg.set_border_color(lv.palette_main(lv.PALETTE.BLUE))
style_bg.set_border_width(2)
style_bg.set_pad_all(6)            # To make the indicator smaller
style_bg.set_radius(6)
style_bg.set_anim_time(1000)

style_indic.init()
style_indic.set_bg_opa(lv.OPA.COVER)
style_indic.set_bg_color(lv.palette_main(lv.PALETTE.BLUE))
style_indic.set_radius(3)

bar = lv.bar(lv.scr_act())
bar.remove_style_all()   # To have a clean start
bar.add_style(style_bg, 0)
bar.add_style(style_indic, lv.PART.INDICATOR)

bar.set_size(200, 20)
bar.center()
bar.set_value(100, lv.ANIM.ON)
```

Temperature meter

```
#include "../../lv_examples.h"
#if LV_USE_BAR && LV_BUILD_EXAMPLES

static void set_temp(void * bar, int32_t temp)
{
    lv_bar_set_value(bar, temp, LV_ANIM_ON);
}

/**
 * A temperature meter example
 */
void lv_example_bar_3(void)
{
    static lv_style_t style_indic;

    lv_style_init(&style_indic);
    lv_style_set_bg_opa(&style_indic, LV_OPA_COVER);
    lv_style_set_bg_color(&style_indic, lv_palette_main(LV_PALETTE_RED));
    lv_style_set_bg_grad_color(&style_indic, lv_palette_main(LV_PALETTE_BLUE));
```

(下页继续)

(续上页)

```
    lv_style_set_bg_grad_dir(&style_indic, LV_GRAD_DIR_VER);

    lv_obj_t * bar = lv_bar_create(lv_scr_act());
    lv_obj_add_style(bar, &style_indic, LV_PART_INDICATOR);
    lv_obj_set_size(bar, 20, 200);
    lv_obj_center(bar);
    lv_bar_set_range(bar, -20, 40);

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_exec_cb(&a, set_temp);
    lv_anim_set_time(&a, 3000);
    lv_anim_set_playback_time(&a, 3000);
    lv_anim_set_var(&a, bar);
    lv_anim_set_values(&a, -20, 40);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
    lv_anim_start(&a);
}


#endif
```

```
def set_temp(bar, temp):
    bar.set_value(temp, lv.ANIM.ON)

#
# A temperature meter example
#


style_indic = lv.style_t()

style_indic.init()
style_indic.set_bg_opa(lv.OPA.COVER)
style_indic.set_bg_color(lv.palette_main(lv.PALETTE.RED))
style_indic.set_bg_grad_color(lv.palette_main(lv.PALETTE.BLUE))
style_indic.set_bg_grad_dir(lv.GRAD_DIR.VER)

bar = lv.bar(lv.scr_act())
bar.add_style(style_indic, lv.PART.INDICATOR)
bar.set_size(20, 200)
bar.center()
bar.set_range(-20, 40)
```

(下页继续)

(续上页)

```
a = lv.anim_t()
a.init()
a.set_time(3000)
a.set_playback_time(3000)
a.set_var(bar)
a.set_values(-20, 40)
a.set_repeat_count(lv.ANIM_REPEAT.INFINITE)
a.set_custom_exec_cb(lambda a, val: set_temp(bar,val))
lv.anim_t.start(a)
```

Stripe pattern and range value

```
#include "../../lv_examples.h"
#if LV_USE_BAR && LV_BUILD_EXAMPLES

/**
 * Bar with stripe pattern and ranged value
 */
void lv_example_bar_4(void)
{
    LV_IMG_DECLARE(img_skew_strip);
    static lv_style_t style_indic;

    lv_style_init(&style_indic);
    lv_style_set_bg_img_src(&style_indic, &img_skew_strip);
    lv_style_set_bg_img_tiled(&style_indic, true);
    lv_style_set_bg_img_opa(&style_indic, LV_OPA_30);

    lv_obj_t * bar = lv_bar_create(lv_scr_act());
    lv_obj_add_style(bar, &style_indic, LV_PART_INDICATOR);

    lv_obj_set_size(bar, 260, 20);
    lv_obj_center(bar);
    lv_bar_set_mode(bar, LV_BAR_MODE_RANGE);
    lv_bar_set_value(bar, 90, LV_ANIM_OFF);
    lv_bar_set_start_value(bar, 20, LV_ANIM_OFF);
}

#endif
```

```
#
# get an icon
#
def get_icon(filename,xres,yres):
    try:
        sdl_filename = "../../assets/" + filename + "_" + str(xres) + "x" + str(yres)␣
↪+ "_argb8888.fnt"
        print("file name: ", sdl_filename)
        with open(sdl_filename,'rb') as f:
            icon_data = f.read()
    except:
        print("Could not find image file: " + filename)
        return None

    icon_dsc = lv.img_dsc_t(
        {
            "header": {"always_zero": 0, "w": xres, "h": yres, "cf": lv.img.CF.TRUE_
↪COLOR_ALPHA},
            "data": icon_data,
            "data_size": len(icon_data),
        }
    )
    return icon_dsc

#
# Bar with stripe pattern and ranged value
#

img_skew_strip_dsc = get_icon("img_skew_strip",80,20)
style_indic = lv.style_t()

style_indic.init()
style_indic.set_bg_img_src(img_skew_strip_dsc)
style_indic.set_bg_img_tiled(True)
style_indic.set_bg_img_opa(lv.OPA._30)

bar = lv.bar(lv.scr_act())
bar.add_style(style_indic, lv.PART.INDICATOR)

bar.set_size(260, 20)
bar.center()
bar.set_mode(lv.bar.MODE.RANGE)
bar.set_value(90, lv.ANIM.OFF)
bar.set_start_value(20, lv.ANIM.OFF)
```

(下页继续)

(续上页)

```
```

Bar with LTR and RTL base direction

```
#include "../../lv_examples.h"
#if LV_USE_BAR && LV_BUILD_EXAMPLES

/**
 * Bar with LTR and RTL base direction
 */
void lv_example_bar_5(void)
{
    lv_obj_t * label;


    lv_obj_t * bar_ltr = lv_bar_create(lv_scr_act());
    lv_obj_set_size(bar_ltr, 200, 20);
    lv_bar_set_value(bar_ltr, 70, LV_ANIM_OFF);
    lv_obj_align(bar_ltr, LV_ALIGN_CENTER, 0, -30);

    label = lv_label_create(lv_scr_act());
    lv_label_set_text(label, "Left to Right base direction");
    lv_obj_align_to(label, bar_ltr, LV_ALIGN_OUT_TOP_MID, 0, -5);

    lv_obj_t * bar_rtl = lv_bar_create(lv_scr_act());
    lv_obj_set_style_base_dir(bar_rtl, LV_BASE_DIR_RTL, 0);
    lv_obj_set_size(bar_rtl, 200, 20);
    lv_bar_set_value(bar_rtl, 70, LV_ANIM_OFF);
    lv_obj_align(bar_rtl, LV_ALIGN_CENTER, 0, 30);

    label = lv_label_create(lv_scr_act());
    lv_label_set_text(label, "Right to Left base direction");
    lv_obj_align_to(label, bar_rtl, LV_ALIGN_OUT_TOP_MID, 0, -5);
}

#endif
```

```
#
# Bar with LTR and RTL base direction
#
```

(下页继续)

(续上页)

```
bar_ltr = lv.bar(lv.scr_act())
bar_ltr.set_size(200, 20)
bar_ltr.set_value(70, lv.ANIM.OFF)
bar_ltr.align(lv.ALIGN.CENTER, 0, -30)

label = lv.label(lv.scr_act())
label.set_text("Left to Right base direction")
label.align_to(bar_ltr, lv.ALIGN.OUT_TOP_MID, 0, -5)

bar_rtl = lv.bar(lv.scr_act())
bar_rtl.set_style_base_dir(lv.BASE_DIR.RTL,0)
bar_rtl.set_size(200, 20)
bar_rtl.set_value(70, lv.ANIM.OFF)
bar_rtl.align(lv.ALIGN.CENTER, 0, 30)

label = lv.label(lv.scr_act())
label.set_text("Right to Left base direction")
label.align_to(bar_rtl, lv.ALIGN.OUT_TOP_MID, 0, -5)
```

Custom drawer to show the current value

```
#include "../../lv_examples.h"
#if LV_USE_BAR && LV_BUILD_EXAMPLES

static void set_value(void *bar, int32_t v)
{
    lv_bar_set_value(bar, v, LV_ANIM_OFF);
}

static void event_cb(lv_event_t * e)
{
    lv_obj_draw_part_dsc_t * dsc = lv_event_get_param(e);
    if(dsc->part != LV_PART_INDICATOR) return;

    lv_obj_t * obj= lv_event_get_target(e);

    lv_draw_label_dsc_t label_dsc;
    lv_draw_label_dsc_init(&label_dsc);
    label_dsc.font = LV_FONT_DEFAULT;

    char buf[8];
```

(下页继续)

(续上页)

```
    lv_snprintf(buf, sizeof(buf), "%d", (int)lv_bar_get_value(obj));

    lv_point_t txt_size;
    lv_txt_get_size(&txt_size, buf, label_dsc.font, label_dsc.letter_space, label_dsc.
↪line_space, LV_COORD_MAX, label_dsc.flag);

    lv_area_t txt_area;
    /*If the indicator is long enough put the text inside on the right*/
    if(lv_area_get_width(dsc->draw_area) > txt_size.x + 20) {
        txt_area.x2 = dsc->draw_area->x2 - 5;
        txt_area.x1 = txt_area.x2 - txt_size.x + 1;
        label_dsc.color = lv_color_white();
    }
    /*If the indicator is still short put the text out of it on the right*/
    else {
        txt_area.x1 = dsc->draw_area->x2 + 5;
        txt_area.x2 = txt_area.x1 + txt_size.x - 1;
        label_dsc.color = lv_color_black();
    }

    txt_area.y1 = dsc->draw_area->y1 + (lv_area_get_height(dsc->draw_area) - txt_size.
↪y) / 2;
    txt_area.y2 = txt_area.y1 + txt_size.y - 1;

    lv_draw_label(dsc->draw_ctx, &label_dsc, &txt_area, buf, NULL);
}

/**
 * Custom drawer on the bar to display the current value
 */
void lv_example_bar_6(void)
{
    lv_obj_t * bar = lv_bar_create(lv_scr_act());
    lv_obj_add_event_cb(bar, event_cb, LV_EVENT_DRAW_PART_END, NULL);
    lv_obj_set_size(bar, 200, 20);
    lv_obj_center(bar);

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_var(&a, bar);
    lv_anim_set_values(&a, 0, 100);
    lv_anim_set_exec_cb(&a, set_value);
    lv_anim_set_time(&a, 2000);
```

(下页继续)

(续上页)

```
    lv_anim_set_playback_time(&a, 2000);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
    lv_anim_start(&a);

}

#endif
```

```
def set_value(bar, v):
    bar.set_value(v, lv.ANIM.OFF)

def event_cb(e):
    dsc = lv.obj_draw_part_dsc_t.__cast__(e.get_param())
    if dsc.part != lv.PART.INDICATOR:
        return

    obj= e.get_target()

    label_dsc = lv.draw_label_dsc_t()
    label_dsc.init()
    # label_dsc.font = LV_FONT_DEFAULT;

    value_txt = str(obj.get_value())
    txt_size = lv.point_t()
    lv.txt_get_size(txt_size, value_txt, label_dsc.font, label_dsc.letter_space,␣
↪label_dsc.line_space, lv.COORD.MAX, label_dsc.flag)

    txt_area = lv.area_t()
    # If the indicator is long enough put the text inside on the right
    if dsc.draw_area.get_width() > txt_size.x + 20:
        txt_area.x2 = dsc.draw_area.x2 - 5
        txt_area.x1 = txt_area.x2 - txt_size.x + 1
        label_dsc.color = lv.color_white()
    # If the indicator is still short put the text out of it on the right*/
    else:
        txt_area.x1 = dsc.draw_area.x2 + 5
        txt_area.x2 = txt_area.x1 + txt_size.x - 1
        label_dsc.color = lv.color_black()

    txt_area.y1 = dsc.draw_area.y1 + (dsc.draw_area.get_height() - txt_size.y) // 2
    txt_area.y2 = txt_area.y1 + txt_size.y - 1

    dsc.draw_ctx.label(label_dsc, txt_area, value_txt, None)
```

(下页继续)

(续上页)

```
#
# Custom drawer on the bar to display the current value
#

bar = lv.bar(lv.scr_act())
bar.add_event_cb(event_cb, lv.EVENT.DRAW_PART_END, None)
bar.set_size(200, 20)
bar.center()

a = lv.anim_t()
a.init()
a.set_var(bar)
a.set_values(0, 100)
a.set_custom_exec_cb(lambda a,val: set_value(bar,val))
a.set_time(2000)
a.set_playback_time(2000)
a.set_repeat_count(lv.ANIM_REPEAT.INFINITE)
lv.anim_t.start(a)
```

API

Typedefs

typedef uint8_t **lv_bar_mode_t**

Enums

enum **[anonymous]**

Values:

enumerator **LV_BAR_MODE_NORMAL**

enumerator **LV_BAR_MODE_SYMMETRICAL**

enumerator **LV_BAR_MODE_RANGE**

enum **lv_bar_draw_part_type_t**

`type` field in `lv_obj_draw_part_dsc_t` if `class_p = lv_bar_class` Used in `LV_EVENT_DRAW_PART_BEGIN` and `LV_EVENT_DRAW_PART_END`

Values:

enumerator **LV_BAR_DRAW_PART_INDICATOR**
The indicator

Functions

lv_obj_t ***lv_bar_create**(*lv_obj_t* *parent)*
Create a bar object

参数 **parent** -- pointer to an object, it will be the parent of the new bar

返回 pointer to the created bar

void **lv_bar_set_value**(*lv_obj_t* *obj, int32_t value, *lv_anim_enable_t* anim)
Set a new value on the bar

参数

- **bar** -- pointer to a bar object
- **value** -- new value
- **anim** -- LV_ANIM_ON: set the value with an animation; LV_ANIM_OFF: change the value immediately

void **lv_bar_set_start_value**(*lv_obj_t* *obj, int32_t start_value, *lv_anim_enable_t* anim)
Set a new start value on the bar

参数

- **obj** -- pointer to a bar object
- **value** -- new start value
- **anim** -- LV_ANIM_ON: set the value with an animation; LV_ANIM_OFF: change the value immediately

void **lv_bar_set_range**(*lv_obj_t* *obj, int32_t min, int32_t max)
Set minimum and the maximum values of a bar

参数

- **obj** -- pointer to the bar object
- **min** -- minimum value
- **max** -- maximum value

void **lv_bar_set_mode**(*lv_obj_t* *obj, *lv_bar_mode_t* mode)
Set the type of bar.

参数

- **obj** -- pointer to bar object

- **mode** -- bar type from ::lv_bar_mode_t

int32_t **lv_bar_get_value**(const *lv_obj_t* *obj)

Get the value of a bar

参数 **obj** -- pointer to a bar object

返回 the value of the bar

int32_t **lv_bar_get_start_value**(const *lv_obj_t* *obj)

Get the start value of a bar

参数 **obj** -- pointer to a bar object

返回 the start value of the bar

int32_t **lv_bar_get_min_value**(const *lv_obj_t* *obj)

Get the minimum value of a bar

参数 **obj** -- pointer to a bar object

返回 the minimum value of the bar

int32_t **lv_bar_get_max_value**(const *lv_obj_t* *obj)

Get the maximum value of a bar

参数 **obj** -- pointer to a bar object

返回 the maximum value of the bar

lv_bar_mode_t **lv_bar_get_mode**(*lv_obj_t* *obj)

Get the type of bar.

参数 **obj** -- pointer to bar object

返回 bar type from ::lv_bar_mode_t

Variables

const lv_obj_class_t **lv_bar_class**

struct **_lv_bar_anim_t**

Public Members

lv_obj_t ***bar**

int32_t **anim_start**

int32_t **anim_end**

int32_t **anim_state**

struct **lv_bar_t**

Public Members

lv_obj_t **obj**

int32_t **cur_value**
Current value of the bar

int32_t **min_value**
Minimum value of the bar

int32_t **max_value**
Maximum value of the bar

int32_t **start_value**
Start value of the bar

lv_area_t **indic_area**
Save the indicator area. Might be used by derived types

_lv_bar_anim_t **cur_value_anim**

_lv_bar_anim_t **start_value_anim**

lv_bar_mode_t **mode**
Type of bar

Button（按钮）(lv_btn)

Overview

Buttons have no new features compared to the *Base object*. They are usuful for semantic purposes and have slightly different default settings.

Buttons, by default, differ from Base object in the following ways:

- Not scrollable
- Added to the default group
- Default height and width set to `LV_SIZE_CONTENT`

与基础对象 相比，按钮没有新功能。它们可用于语义目的，并且默认设置略有不同。

默认情况下，按钮在以下方面与基础对象不同：

- 不可滚动
- 添加到默认组
- 默认高度和宽度设置为 `LV_SIZE_CONTENT`

Parts and Styles（零件和样式）

- `LV_PART_MAIN` The background of the button. Uses the typical background style properties.
- `LV_PART_MAIN` 按钮的背景。使用典型的背景样式属性。

Usage（用法）

There are no new features compared to *Base object*.

与基础对象 相比，没有新功能。

Events（事件）

- `LV_EVENT_VALUE_CHANGED` when the `LV_OBJ_FLAG_CHECKABLE` flag is enabled and the object is clicked. The event happens on transition to/from the checked state.

Learn more about *Events*.

- 如果打开了 `LV_OBJ_FLAG_CHECKABLE` ，当对象被点击时有选中切换 (Toggle) 状态的效果，并且可以在 `LV_EVENT_VALUE_CHANGED` 事件类型中处理事件。

详细了解事件。

Keys（按键）

Note that the state of LV_KEY_ENTER is translated to LV_EVENT_PRESSED/PRESSING/RELEASED etc.

See the events of the *Base object* too.

Learn more about *Keys*.

请注意，LV_KEY_ENTER 的状态被转换为 LV_EVENT_PRESSED/PRESSING/RELEASED 等。

参见基础对象 的事件。

了解有关*Keys* 的更多信息。

Example

Simple Buttons

```
#include "../../lv_examples.h"
#if LV_USE_BTN && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);

    if(code == LV_EVENT_CLICKED) {
        LV_LOG_USER("Clicked");
    }
    else if(code == LV_EVENT_VALUE_CHANGED) {
        LV_LOG_USER("Toggled");
    }
}

void lv_example_btn_1(void)
{
    lv_obj_t * label;

    lv_obj_t * btn1 = lv_btn_create(lv_scr_act());
    lv_obj_add_event_cb(btn1, event_handler, LV_EVENT_ALL, NULL);
    lv_obj_align(btn1, LV_ALIGN_CENTER, 0, -40);

    label = lv_label_create(btn1);
    lv_label_set_text(label, "Button");
    lv_obj_center(label);
```

(下页继续)

(续上页)

```
    lv_obj_t * btn2 = lv_btn_create(lv_scr_act());
    lv_obj_add_event_cb(btn2, event_handler, LV_EVENT_ALL, NULL);
    lv_obj_align(btn2, LV_ALIGN_CENTER, 0, 40);
    lv_obj_add_flag(btn2, LV_OBJ_FLAG_CHECKABLE);
    lv_obj_set_height(btn2, LV_SIZE_CONTENT);

    label = lv_label_create(btn2);
    lv_label_set_text(label, "Toggle");
    lv_obj_center(label);
}
#endif
```

```
def event_handler(evt):
    code = evt.get_code()

    if code == lv.EVENT.CLICKED:
            print("Clicked event seen")
    elif code == lv.EVENT.VALUE_CHANGED:
        print("Value changed seen")

# create a simple button
btn1 = lv.btn(lv.scr_act())

# attach the callback
btn1.add_event_cb(event_handler,lv.EVENT.ALL, None)

btn1.align(lv.ALIGN.CENTER,0,-40)
label=lv.label(btn1)
label.set_text("Button")

# create a toggle button
btn2 = lv.btn(lv.scr_act())

# attach the callback
#btn2.add_event_cb(event_handler,lv.EVENT.VALUE_CHANGED,None)
btn2.add_event_cb(event_handler,lv.EVENT.ALL, None)

btn2.align(lv.ALIGN.CENTER,0,40)
btn2.add_flag(lv.obj.FLAG.CHECKABLE)
btn2.set_height(lv.SIZE.CONTENT)

label=lv.label(btn2)
label.set_text("Toggle")
```

(下页继续)

(续上页)

```
label.center()
```

Styling buttons

```
#include "../../lv_examples.h"
#if LV_USE_BTN && LV_BUILD_EXAMPLES

/**
 * Style a button from scratch
 */
void lv_example_btn_2(void)
{
    /*Init the style for the default state*/
    static lv_style_t style;
    lv_style_init(&style);

    lv_style_set_radius(&style, 3);

    lv_style_set_bg_opa(&style, LV_OPA_100);
    lv_style_set_bg_color(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_bg_grad_color(&style, lv_palette_darken(LV_PALETTE_BLUE, 2));
    lv_style_set_bg_grad_dir(&style, LV_GRAD_DIR_VER);

    lv_style_set_border_opa(&style, LV_OPA_40);
    lv_style_set_border_width(&style, 2);
    lv_style_set_border_color(&style, lv_palette_main(LV_PALETTE_GREY));

    lv_style_set_shadow_width(&style, 8);
    lv_style_set_shadow_color(&style, lv_palette_main(LV_PALETTE_GREY));
    lv_style_set_shadow_ofs_y(&style, 8);

    lv_style_set_outline_opa(&style, LV_OPA_COVER);
    lv_style_set_outline_color(&style, lv_palette_main(LV_PALETTE_BLUE));

    lv_style_set_text_color(&style, lv_color_white());
    lv_style_set_pad_all(&style, 10);

    /*Init the pressed style*/
    static lv_style_t style_pr;
    lv_style_init(&style_pr);

    /*Add a large outline when pressed*/
```

(下页继续)

(续上页)

```
    lv_style_set_outline_width(&style_pr, 30);
    lv_style_set_outline_opa(&style_pr, LV_OPA_TRANSP);

    lv_style_set_translate_y(&style_pr, 5);
    lv_style_set_shadow_ofs_y(&style_pr, 3);
    lv_style_set_bg_color(&style_pr, lv_palette_darken(LV_PALETTE_BLUE, 2));
    lv_style_set_bg_grad_color(&style_pr, lv_palette_darken(LV_PALETTE_BLUE, 4));

    /*Add a transition to the outline*/
    static lv_style_transition_dsc_t trans;
    static lv_style_prop_t props[] = {LV_STYLE_OUTLINE_WIDTH, LV_STYLE_OUTLINE_OPA, 0}
↪;
    lv_style_transition_dsc_init(&trans, props, lv_anim_path_linear, 300, 0, NULL);

    lv_style_set_transition(&style_pr, &trans);

    lv_obj_t * btn1 = lv_btn_create(lv_scr_act());
    lv_obj_remove_style_all(btn1);                          /*Remove the style coming␣
↪from the theme*/
    lv_obj_add_style(btn1, &style, 0);
    lv_obj_add_style(btn1, &style_pr, LV_STATE_PRESSED);
    lv_obj_set_size(btn1, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
    lv_obj_center(btn1);

    lv_obj_t * label = lv_label_create(btn1);
    lv_label_set_text(label, "Button");
    lv_obj_center(label);
}
#endif
```

```
#
# Style a button from scratch
#

# Init the style for the default state
style = lv.style_t()
style.init()

style.set_radius(3)

style.set_bg_opa(lv.OPA.COVER)
style.set_bg_color(lv.palette_main(lv.PALETTE.BLUE))
style.set_bg_grad_color(lv.palette_darken(lv.PALETTE.BLUE, 2))
```

(下页继续)

(续上页)

```
style.set_bg_grad_dir(lv.GRAD_DIR.VER)

style.set_border_opa(lv.OPA._40)
style.set_border_width(2)
style.set_border_color(lv.palette_main(lv.PALETTE.GREY))

style.set_shadow_width(8)
style.set_shadow_color(lv.palette_main(lv.PALETTE.GREY))
style.set_shadow_ofs_y(8)

style.set_outline_opa(lv.OPA.COVER)
style.set_outline_color(lv.palette_main(lv.PALETTE.BLUE))

style.set_text_color(lv.color_white())
style.set_pad_all(10)

# Init the pressed style
style_pr = lv.style_t()
style_pr.init()

# Add a large outline when pressed
style_pr.set_outline_width(30)
style_pr.set_outline_opa(lv.OPA.TRANSP)

style_pr.set_translate_y(5)
style_pr.set_shadow_ofs_y(3)
style_pr.set_bg_color(lv.palette_darken(lv.PALETTE.BLUE, 2))
style_pr.set_bg_grad_color(lv.palette_darken(lv.PALETTE.BLUE, 4))

# Add a transition to the outline
trans = lv.style_transition_dsc_t()
props = [lv.STYLE.OUTLINE_WIDTH, lv.STYLE.OUTLINE_OPA, 0]
trans.init(props, lv.anim_t.path_linear, 300, 0, None)

style_pr.set_transition(trans)

btn1 = lv.btn(lv.scr_act())
btn1.remove_style_all()                         # Remove the style coming from the␣
↪theme
btn1.add_style(style, 0)
btn1.add_style(style_pr, lv.STATE.PRESSED)
btn1.set_size(lv.SIZE.CONTENT, lv.SIZE.CONTENT)
btn1.center()
```

(下页继续)

(续上页)

```
label = lv.label(btn1)
label.set_text("Button")
label.center()
```

Gummy button

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_BTN

/**
 * Create a style transition on a button to act like a gum when clicked
 */
void lv_example_btn_3(void)
{
    /*Properties to transition*/
    static lv_style_prop_t props[] = {
            LV_STYLE_TRANSFORM_WIDTH, LV_STYLE_TRANSFORM_HEIGHT, LV_STYLE_TEXT_LETTER_
↪SPACE, 0
    };

    /*Transition descriptor when going back to the default state.
     *Add some delay to be sure the press transition is visible even if the press was␣
↪very short*/
    static lv_style_transition_dsc_t transition_dsc_def;
    lv_style_transition_dsc_init(&transition_dsc_def, props, lv_anim_path_overshoot,␣
↪250, 100, NULL);

    /*Transition descriptor when going to pressed state.
     *No delay, go to presses state immediately*/
    static lv_style_transition_dsc_t transition_dsc_pr;
    lv_style_transition_dsc_init(&transition_dsc_pr, props, lv_anim_path_ease_in_out,␣
↪250, 0, NULL);

    /*Add only the new transition to he default state*/
    static lv_style_t style_def;
    lv_style_init(&style_def);
    lv_style_set_transition(&style_def, &transition_dsc_def);

    /*Add the transition and some transformation to the presses state.*/
    static lv_style_t style_pr;
```

(下页继续)

(续上页)

```
    lv_style_init(&style_pr);
    lv_style_set_transform_width(&style_pr, 10);
    lv_style_set_transform_height(&style_pr, -10);
    lv_style_set_text_letter_space(&style_pr, 10);
    lv_style_set_transition(&style_pr, &transition_dsc_pr);

    lv_obj_t * btn1 = lv_btn_create(lv_scr_act());
    lv_obj_align(btn1, LV_ALIGN_CENTER, 0, -80);
    lv_obj_add_style(btn1, &style_pr, LV_STATE_PRESSED);
    lv_obj_add_style(btn1, &style_def, 0);

    lv_obj_t * label = lv_label_create(btn1);
    lv_label_set_text(label, "Gum");
}
#endif
```

```
#
# Create a style transition on a button to act like a gum when clicked
#

# Properties to transition
props = [lv.STYLE.TRANSFORM_WIDTH, lv.STYLE.TRANSFORM_HEIGHT, lv.STYLE.TEXT_LETTER_
↪SPACE, 0]

# Transition descriptor when going back to the default state.
# Add some delay to be sure the press transition is visible even if the press was␣
↪very short*/
transition_dsc_def = lv.style_transition_dsc_t()
transition_dsc_def.init(props, lv.anim_t.path_overshoot, 250, 100, None)

# Transition descriptor when going to pressed state.
# No delay, go to pressed state immediately
transition_dsc_pr = lv.style_transition_dsc_t()
transition_dsc_pr.init(props, lv.anim_t.path_ease_in_out, 250, 0, None)

# Add only the new transition to the default state
style_def = lv.style_t()
style_def.init()
style_def.set_transition(transition_dsc_def)

# Add the transition and some transformation to the presses state.
style_pr = lv.style_t()
style_pr.init()
```

(下页继续)

(续上页)

```
style_pr.set_transform_width(10)
style_pr.set_transform_height(-10)
style_pr.set_text_letter_space(10)
style_pr.set_transition(transition_dsc_pr)

btn1 = lv.btn(lv.scr_act())
btn1.align(lv.ALIGN.CENTER, 0, -80)
btn1.add_style(style_pr, lv.STATE.PRESSED)
btn1.add_style(style_def, 0)

label = lv.label(btn1)
label.set_text("Gum")
```

API

Functions

lv_obj_t ***lv_btn_create**(*lv_obj_t* *parent)

Create a button object

参数 **parent** -- pointer to an object, it will be the parent of the new button

返回 pointer to the created button

Variables

const lv_obj_class_t **lv_btn_class**

struct **lv_btn_t**

Public Members

lv_obj_t **obj**

Button matrix（按钮矩阵）(lv_btnmatrix)

Overview（概述）

The Button Matrix object is a lightweight way to display multiple buttons in rows and columns. Lightweight because the buttons are not actually created but just virtually drawn on the fly. This way, one button use only eight extra bytes of memory instead of the ~100-150 bytes a normal *Button* object plus the 100 or so bytes for the the *Label* object.

The Button matrix is added to the default group (if one is set). Besides the Button matrix is an editable object to allow selecting and clicking the buttons with encoder navigation too.

按钮矩阵 (lv_btnmatrix) 组件是一种在行和列中显示多个按钮的轻量级实现方式。按钮不是实际创建出来的，而是实时绘制出来的，所以轻量级，因为这样一个按钮仅使用 8 个字节的内存，而不是普通*Button* 组件那样：~100-150 字节再加上*Label* 组件的内存占用。

按钮矩阵添加到默认组（如果之前已设置了组）。此外，按钮矩阵是一个可编辑的对象，可以通过编码器导航选择和单击按钮。

Parts and Styles（零件和样式）

- `LV_PART_MAIN` The background of the button matrix, uses the typical background style properties. `pad_row` and `pad_column` sets the space between the buttons.
- `LV_PART_ITEMS` The buttons all use the text and typical background style properties except translations and transformations.
- `LV_PART_MAIN` 按钮矩阵的背景，使用所有组件默认都有的典型的背景样式属性。可通过 `pad_row` 和 `pad_column` 设置按钮之间的空间。
- `LV_PART_ITEMS` 除了转变之外，按钮都使用文本和典型的背景样式属性。

Usage（用法）

Button's text（按钮的文字）

There is a text on each button. To specify them a descriptor string array, called *map*, needs to be used. The map can be set with `lv_btnmatrix_set_map(btnm, my_map)`. The declaration of a map should look like `const char * map[] = {"btn1", "btn2", "btn3", NULL}`. Note that the last element has to be either `NULL` or an empty string (`""`)!

Use `"\n"` in the map to insert a **line break**. E.g. `{"btn1", "btn2", "\n", "btn3", ""}`. Each line's buttons have their width calculated automatically. So in the example the first row will have 2 buttons each with 50% width and a second row with 1 button having 100% width.

每个按钮上都可以有文字。要指定按钮的文字，需要使用称为 *map* 的描述符按钮布局的字符串数组。map 可以使用 `lv_btnmatrix_set_map(btnm, my_map)` 接口设置。map 的格式：`const char * map[] = {"btn1", "btn2", "btn3", NULL}`。请注意，map 数组的最后一个元素必须是 `NULL` 或空字符串 (`""`)!

在 map 中使用 `"\n"` 插入**换行符**。例如。`{"btn1", "btn2", "\n", "btn3", ""}`。每行按钮的宽度都会自动计算平均分配 (默认)。因此，在上面的示例中，第一行将有 2 个按钮，每个按钮的宽度为 50%，第二行将有 1 个按钮的宽度为 100%。

Control buttons（控制按钮）

The buttons' width can be set relative to the other button in the same row with `lv_btnmatrix_set_btn_width(btnm, btn_id, width)` E.g. in a line with two buttons: *btnA, width = 1* and *btnB, width = 2*, *btnA* will have 33 % width and *btnB* will have 66 % width. It's similar to how the `flex-grow` property works in CSS. The width must be in the [1..7] range and the default width is 1.

In addition to the width, each button can be customized with the following parameters:

- `LV_BTNMATRIX_CTRL_HIDDEN` Makes a button hidden (hidden buttons still take up space in the layout, they are just not visible or clickable)
- `LV_BTNMATRIX_CTRL_NO_REPEAT` Disable repeating when the button is long pressed
- `LV_BTNMATRIX_CTRL_DISABLED` Makes a button disabled Like `LV_STATE_DISABLED` on normal objects
- `LV_BTNMATRIX_CTRL_CHECKABLE` Enable toggling of a button. I.e. `LV_STATE_CHECHED` will be added/removed as the button is clicked
- `LV_BTNMATRIX_CTRL_CHECKED` MAke the button checked. It will use the `LV_STATE_CHECHKED` styles.
- `LV_BTNMATRIX_CTRL_CLICK_TRIG` Enabled: send LV_EVENT_VALUE_CHANGE on CLICK, Disabled: send LV_EVENT_VALUE_CHANGE on PRESS*/
- `LV_BTNMATRIX_CTRL_RECOLOR` Enable recoloring of button texts with `#`. E.g. `"It's #ff0000 red#"`
- `LV_BTNMATRIX_CTRL_CUSTOM_1` Custom free to use flag
- `LV_BTNMATRIX_CTRL_CUSTOM_2` Custom free to use flag

By default all flags are disabled.

可以使用 `lv_btnmatrix_set_btn_width(btnm, btn_id, width)` 接口设置相对于同一行中的另一个按钮的宽度。例如。在一行中有两个按钮这样设置：*btnA, width = 1* 和 *btnB, width = 2*，这样 *btnA* 将有 33% 的宽度，*btnB* 将有 66% 的宽度。它类似于 `flex-grow` 属性在 CSS 中的工作方式。宽度必须在 [1..7] 范围内，默认宽度为 1。

除了宽度之外，每个按钮还可以使用以下参数进行自定义：

- `LV_BTNMATRIX_CTRL_HIDDEN` 隐藏按钮（隐藏的按钮仍然占用布局中的空间，它们只是不可见或不可点击）
- `LV_BTNMATRIX_CTRL_NO_REPEAT` 长按按钮时禁用重复

- LV_BTNMATRIX_CTRL_DISABLED 使按钮被禁用，就像普通对象上的 LV_STATE_DISABLED
- LV_BTNMATRIX_CTRL_CHECKABLE 启用按钮切换。也就是 LV_STATE_CHECHED 将在按钮被点击时添加/删除
- LV_BTNMATRIX_CTRL_CHECKED 检查按钮。它将使用 LV_STATE_CHECHKED 的样式。
- LV_BTNMATRIX_CTRL_CLICK_TRIG 启用：在点击时发送 **LV_EVENT_VALUE_CHANGE**；禁用：在按下时发送 **LV_EVENT_VALUE_CHANGE**
- LV_BTNMATRIX_CTRL_RECOLOR 使用 # 启用按钮文本的重新着色。例如 "It's #ff0000 red btn#"
- LV_BTNMATRIX_CTRL_CUSTOM_1 可自定义使用的标志
- LV_BTNMATRIX_CTRL_CUSTOM_2 可自定义使用的标志

默认情况下，所有标志都被禁用。

To set or clear a button's control attribute, use lv_btnmatrix_set_btn_ctrl(btnm, btn_id, LV_BTNM_CTRL_...) and lv_btnmatrix_clear_btn_ctrl(btnm, btn_id, LV_BTNMATRIX_CTRL_...) respectively. More LV_BTNM_CTRL_... values can be OR-ed

To set/clear the same control attribute for all buttons of a button matrix, use lv_btnmatrix_set_btn_ctrl_all(btnm, btn_id, LV_BTNM_CTRL_...) and lv_btnmatrix_clear_btn_ctrl_all(btnm, btn_id, LV_BTNMATRIX_CTRL_...).

The set a control map for a button matrix (similarly to the map for the text), use lv_btnmatrix_set_ctrl_map(btnm, ctrl_map). An element of ctrl_map should look like ctrl_map[0] = width | LV_BTNM_CTRL_NO_REPEAT | LV_BTNM_CTRL_CHECHKABLE. The number of elements should be equal to the number of buttons (excluding newlines characters).

要设置或清除按钮的控制属性，请使用 lv_btnmatrix_set_btn_ctrl(btnm, btn_id, LV_BTNM_CTRL_...) 和 lv_btnmatrix_clear_btn_ctrl(btnm, btn_id, LV_BTNMATRIX_CTRL_...) 分别。更多 LV_BTNM_CTRL_... 值可以被 OR-ed

要为按钮矩阵的所有按钮设置/清除相同的控制属性，请使用 lv_btnmatrix_set_btn_ctrl_all(btnm, btn_id, LV_BTNM_CTRL_...) 和 lv_btnmatrix_clear_btn_ctrl_all(btnm, btn_id, LV_BTNMATRIX_CTRL_...)。

我们可以写一个数组来一次单独设置多个或者所有的按钮，这有点像一个控制表，这里称其为 ctrl_map，我们可以使用 lv_btnmatrix_set_ctrl_map(btnm, ctrl_map) 将控制表添加到按钮矩阵中。ctrl_map 中的元素的格式：ctrl_map[0] = width | LV_BTNM_CTRL_NO_REPEAT | LV_BTNM_CTRL_CHECHKABLE，也就是我们可以添加多个属性。元素的数量应该等于 (可以小于，但是不应该超出) 按钮的数量 (不包括换行符)。

One check（一次检查）

The "One check" feature can be enabled with `lv_btnmatrix_set_one_check(btnm, true)` to allow only one button to be checked at a time.

可以使用 `lv_btnmatrix_set_one_check(btnm, true)` 启用“一次检查”功能，这样一次只能检查一个按钮 (我们就能知道最后点击的是哪个按钮)。

Events（事件）

- `LV_EVENT_VALUE_CHANGED` Sent when a button is pressed/released or repeated after long press. The event parameter is set to the ID of the pressed/released button.
- `LV_EVENT_DRAW_PART_BEGIN` and `LV_EVENT_DRAW_PART_END` are sent for the following types:
  - `LV_BTNMATRIX_DRAW_PART_BTN` The individual buttons.
    - `part`: `LV_PART_ITEMS`
    - `id`:index of the button being drawn
    - `draw_area`: the area of teh button
    - `rect_dsc`
- `LV_EVENT_VALUE_CHANGED` 按下/释放按钮或长按时发送。事件参数设置为按下/释放按钮的 ID。
- 为以下类型发送 `LV_EVENT_DRAW_PART_BEGIN` 和 `LV_EVENT_DRAW_PART_END`：
  - `LV_BTNMATRIX_DRAW_PART_BTN` 单个按钮。
    - 部分：`LV_PART_ITEMS`
    - `id`：正在绘制的按钮的索引
    - `draw_area`：按钮的区域
    - `rect_dsc`

See the events of the *Base object* too.

`lv_btnmatrix_get_selected_btn(btnm)` returns the index of the most recently released or focused button or `LV_BTNMATRIX_BTN_NONE` if no such button.

`lv_btnmatrix_get_btn_text(btnm, btn_id)` returns a pointer to the text of `btn_id`th button.

Learn more about *Events*.

参见基础对象 的事件。

`lv_btnmatrix_get_selected_btn(btnm)` 返回最后被释放或聚焦的按钮的索引值，如果没有这样的按钮，则返回 `LV_BTNMATRIX_BTN_NONE`。

`lv_btnmatrix_get_btn_text(btnm, btn_id)` 返回索引为 `btn_id` 的按钮的文本的指针。

详细了解事件。

Keys（按键）

- LV_KEY_RIGHT/UP/LEFT/RIGHT To navigate among the buttons to select one
- LV_KEY_ENTER To press/release the selected button

Learn more about *Keys*.

- LV_KEY_RIGHT/UP/LEFT/RIGHT 在按钮矩阵的按钮之间导航来选中不同的按钮。
- LV_KEY_ENTER 按下/释放所选按钮。

了解有关按键 的更多信息。

Example

Simple Button matrix

```
#include "../../lv_examples.h"
#if LV_USE_BTNMATRIX && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        uint32_t id = lv_btnmatrix_get_selected_btn(obj);
        const char * txt = lv_btnmatrix_get_btn_text(obj, id);

        LV_LOG_USER("%s was pressed\n", txt);
    }
}


static const char * btnm_map[] = {"1", "2", "3", "4", "5", "\n",
                                  "6", "7", "8", "9", "0", "\n",
                                  "Action1", "Action2", ""};

void lv_example_btnmatrix_1(void)
{
    lv_obj_t * btnm1 = lv_btnmatrix_create(lv_scr_act());
    lv_btnmatrix_set_map(btnm1, btnm_map);
    lv_btnmatrix_set_btn_width(btnm1, 10, 2);        /*Make "Action1" twice as wide␣
↪as "Action2"*/
```

(下页继续)

(续上页)

```
    lv_btnmatrix_set_btn_ctrl(btnm1, 10, LV_BTNMATRIX_CTRL_CHECKABLE);
    lv_btnmatrix_set_btn_ctrl(btnm1, 11, LV_BTNMATRIX_CTRL_CHECKED);
    lv_obj_align(btnm1, LV_ALIGN_CENTER, 0, 0);
    lv_obj_add_event_cb(btnm1, event_handler, LV_EVENT_ALL, NULL);
}

#endif
```

```
def event_handler(evt):
    code = evt.get_code()
    obj  = evt.get_target()

    if code == lv.EVENT.VALUE_CHANGED :
        id = obj.get_selected_btn()
        txt = obj.get_btn_text(id)

        print("%s was pressed"%txt)

btnm_map = ["1", "2", "3", "4", "5", "\n",
            "6", "7", "8", "9", "0", "\n",
            "Action1", "Action2", ""]

btnm1 = lv.btnmatrix(lv.scr_act())
btnm1.set_map(btnm_map)
btnm1.set_btn_width(10, 2)        # Make "Action1" twice as wide as "Action2"
btnm1.set_btn_ctrl(10, lv.btnmatrix.CTRL.CHECKABLE)
btnm1.set_btn_ctrl(11, lv.btnmatrix.CTRL.CHECKED)
btnm1.align(lv.ALIGN.CENTER, 0, 0)
btnm1.add_event_cb(event_handler, lv.EVENT.ALL, None)


#endif
```

Custom buttons

```
#include "../../lv_examples.h"
#if LV_USE_BTNMATRIX && LV_BUILD_EXAMPLES


static void event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
```

(下页继续)

(续上页)

```
    lv_obj_t * obj = lv_event_get_target(e);
    if(code == LV_EVENT_DRAW_PART_BEGIN) {
        lv_obj_draw_part_dsc_t * dsc = lv_event_get_param(e);

        /*Change the draw descriptor the 2nd button*/
        if(dsc->id == 1) {
            dsc->rect_dsc->radius = 0;
            if(lv_btnmatrix_get_selected_btn(obj) == dsc->id)  dsc->rect_dsc->bg_
↪color = lv_palette_darken(LV_PALETTE_BLUE, 3);
            else dsc->rect_dsc->bg_color = lv_palette_main(LV_PALETTE_BLUE);

            dsc->rect_dsc->shadow_width = 6;
            dsc->rect_dsc->shadow_ofs_x = 3;
            dsc->rect_dsc->shadow_ofs_y = 3;
            dsc->label_dsc->color = lv_color_white();
        }
        /*Change the draw descriptor the 3rd button*/
        else if(dsc->id == 2) {
            dsc->rect_dsc->radius = LV_RADIUS_CIRCLE;
            if(lv_btnmatrix_get_selected_btn(obj) == dsc->id)  dsc->rect_dsc->bg_
↪color = lv_palette_darken(LV_PALETTE_RED, 3);
            else dsc->rect_dsc->bg_color = lv_palette_main(LV_PALETTE_RED);

            dsc->label_dsc->color = lv_color_white();
        }
        else if(dsc->id == 3) {
            dsc->label_dsc->opa = LV_OPA_TRANSP; /*Hide the text if any*/

        }
    }
    if(code == LV_EVENT_DRAW_PART_END) {
        lv_obj_draw_part_dsc_t * dsc = lv_event_get_param(e);

        /*Add custom content to the 4th button when the button itself was drawn*/
        if(dsc->id == 3) {
            LV_IMG_DECLARE(img_star);
            lv_img_header_t header;
            lv_res_t res = lv_img_decoder_get_info(&img_star, &header);
            if(res != LV_RES_OK) return;

            lv_area_t a;
            a.x1 = dsc->draw_area->x1 + (lv_area_get_width(dsc->draw_area) - header.
↪w) / 2;
```

(下页继续)

(续上页)

```
            a.x2 = a.x1 + header.w - 1;
            a.y1 = dsc->draw_area->y1 + (lv_area_get_height(dsc->draw_area) - header.
↪h) / 2;
            a.y2 = a.y1 + header.h - 1;

            lv_draw_img_dsc_t img_draw_dsc;
            lv_draw_img_dsc_init(&img_draw_dsc);
            img_draw_dsc.recolor = lv_color_black();
            if(lv_btnmatrix_get_selected_btn(obj) == dsc->id)  img_draw_dsc.recolor_
↪opa = LV_OPA_30;

            lv_draw_img(dsc->draw_ctx, &img_draw_dsc, &a, &img_star);
        }
    }
}

/**
 * Add custom drawer to the button matrix to customize buttons one by one
 */
void lv_example_btnmatrix_2(void)
{
    lv_obj_t * btnm = lv_btnmatrix_create(lv_scr_act());
    lv_obj_add_event_cb(btnm, event_cb, LV_EVENT_ALL, NULL);
    lv_obj_center(btnm);
}

#endif
```

```
from imagetools import get_png_info, open_png

# Register PNG image decoder
decoder = lv.img.decoder_create()
decoder.info_cb = get_png_info
decoder.open_cb = open_png

# Create an image from the png file
try:
    with open('../../assets/img_star.png','rb') as f:
        png_data = f.read()
except:
    print("Could not find star.png")
    sys.exit()
```

(下页继续)

(续上页)

```
img_star_argb = lv.img_dsc_t({
  'data_size': len(png_data),
  'data': png_data
})

def event_cb(e):
    code = e.get_code()
    obj = e.get_target()
    dsc = lv.obj_draw_part_dsc_t.__cast__(e.get_param())
    if code == lv.EVENT.DRAW_PART_BEGIN:
        # Change the draw descriptor the 2nd button
        if dsc.id == 1:
            dsc.rect_dsc.radius = 0
            if obj.get_selected_btn() == dsc.id:
                dsc.rect_dsc.bg_color = lv.palette_darken(lv.PALETTE.GREY, 3)
            else:
                dsc.rect_dsc.bg_color = lv.palette_main(lv.PALETTE.BLUE)

            dsc.rect_dsc.shadow_width = 6
            dsc.rect_dsc.shadow_ofs_x = 3
            dsc.rect_dsc.shadow_ofs_y = 3
            dsc.label_dsc.color = lv.color_white()

        # Change the draw descriptor the 3rd button

        elif dsc.id == 2:
            dsc.rect_dsc.radius = lv.RADIUS.CIRCLE
            if obj.get_selected_btn() == dsc.id:
                dsc.rect_dsc.bg_color = lv.palette_darken(lv.PALETTE.RED, 3)
            else:
                dsc.rect_dsc.bg_color = lv.palette_main(lv.PALETTE.RED)

                dsc.label_dsc.color = lv.color_white()
        elif dsc.id == 3:
            dsc.label_dsc.opa = lv.OPA.TRANSP  # Hide the text if any

    if code == lv.EVENT.DRAW_PART_END:
        # Add custom content to the 4th button when the button itself was drawn
        if dsc.id == 3:
            # LV_IMG_DECLARE(img_star)
            header = lv.img_header_t()
            res = lv.img.decoder_get_info(img_star_argb, header)
            if res != lv.RES.OK:
```

(下页继续)

(续上页)

```
                print("error when getting image header")
                return
            else:
                a = lv.area_t()
                a.x1 = dsc.draw_area.x1 + (dsc.draw_area.get_width() - header.w) // 2
                a.x2 = a.x1 + header.w - 1
                a.y1 = dsc.draw_area.y1 + (dsc.draw_area.get_height() - header.h) // 2
                a.y2 = a.y1 + header.h - 1
                img_draw_dsc = lv.draw_img_dsc_t()
                img_draw_dsc.init()
                img_draw_dsc.recolor = lv.color_black()
                if obj.get_selected_btn() == dsc.id:
                    img_draw_dsc.recolor_opa = lv.OPA._30

                dsc.draw_ctx.img(img_draw_dsc, a, img_star_argb)

#
# Add custom drawer to the button matrix to c
#
btnm = lv.btnmatrix(lv.scr_act())
btnm.add_event_cb(event_cb, lv.EVENT.ALL, None)
btnm.center()
```

Pagination

```
#include "../../lv_examples.h"
#if LV_USE_BTNMATRIX  && LV_BUILD_EXAMPLES

static void event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target(e);
    uint32_t id = lv_btnmatrix_get_selected_btn(obj);
    bool prev = id == 0 ? true : false;
    bool next = id == 6 ? true : false;
    if(prev || next) {
        /*Find the checked button*/
        uint32_t i;
        for(i = 1; i < 7; i++) {
            if(lv_btnmatrix_has_btn_ctrl(obj, i, LV_BTNMATRIX_CTRL_CHECKED)) break;
        }
```

(下页继续)

(续上页)

```
        if(prev && i > 1) i--;
        else if(next && i < 5) i++;

        lv_btnmatrix_set_btn_ctrl(obj, i, LV_BTNMATRIX_CTRL_CHECKED);
    }
}

/**
 * Make a button group (pagination)
 */
void lv_example_btnmatrix_3(void)
{
    static lv_style_t style_bg;
    lv_style_init(&style_bg);
    lv_style_set_pad_all(&style_bg, 0);
    lv_style_set_pad_gap(&style_bg, 0);
    lv_style_set_clip_corner(&style_bg, true);
    lv_style_set_radius(&style_bg, LV_RADIUS_CIRCLE);
    lv_style_set_border_width(&style_bg, 0);


    static lv_style_t style_btn;
    lv_style_init(&style_btn);
    lv_style_set_radius(&style_btn, 0);
    lv_style_set_border_width(&style_btn, 1);
    lv_style_set_border_opa(&style_btn, LV_OPA_50);
    lv_style_set_border_color(&style_btn, lv_palette_main(LV_PALETTE_GREY));
    lv_style_set_border_side(&style_btn, LV_BORDER_SIDE_INTERNAL);
    lv_style_set_radius(&style_btn, 0);

    static const char * map[] = {LV_SYMBOL_LEFT, "1", "2", "3", "4", "5", LV_SYMBOL_
↪RIGHT, ""};

    lv_obj_t * btnm = lv_btnmatrix_create(lv_scr_act());
    lv_btnmatrix_set_map(btnm, map);
    lv_obj_add_style(btnm, &style_bg, 0);
    lv_obj_add_style(btnm, &style_btn, LV_PART_ITEMS);
    lv_obj_add_event_cb(btnm, event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    lv_obj_set_size(btnm, 225, 35);

    /*Allow selecting on one number at time*/
    lv_btnmatrix_set_btn_ctrl_all(btnm, LV_BTNMATRIX_CTRL_CHECKABLE);
    lv_btnmatrix_clear_btn_ctrl(btnm, 0, LV_BTNMATRIX_CTRL_CHECKABLE);
```

(下页继续)

(续上页)

```
    lv_btnmatrix_clear_btn_ctrl(btnm, 6, LV_BTNMATRIX_CTRL_CHECKABLE);

    lv_btnmatrix_set_one_checked(btnm, true);
    lv_btnmatrix_set_btn_ctrl(btnm, 1, LV_BTNMATRIX_CTRL_CHECKED);

    lv_obj_center(btnm);

}

#endif
```

```
def event_cb(e):
    obj = e.get_target()
    id = obj.get_selected_btn()
    if id == 0:
        prev = True
    else:
        prev = False
    if id == 6:
        next = True
    else:
        next = False
    if prev or next:
        # Find the checked butto
        for i in range(7):
            if obj.has_btn_ctrl(i, lv.btnmatrix.CTRL.CHECKED):
                break
        if prev and i > 1:
            i-=1
        elif next and i < 5:
            i+=1

        obj.set_btn_ctrl(i, lv.btnmatrix.CTRL.CHECKED)

#
# Make a button group
#

style_bg = lv.style_t()
style_bg.init()
style_bg.set_pad_all(0)
style_bg.set_pad_gap(0)
style_bg.set_clip_corner(True)
```

(下页继续)

(续上页)

```
style_bg.set_radius(lv.RADIUS.CIRCLE)
style_bg.set_border_width(0)


style_btn = lv.style_t()
style_btn.init()
style_btn.set_radius(0)
style_btn.set_border_width(1)
style_btn.set_border_opa(lv.OPA._50)
style_btn.set_border_color(lv.palette_main(lv.PALETTE.GREY))
style_btn.set_border_side(lv.BORDER_SIDE.INTERNAL)
style_btn.set_radius(0)

map = [lv.SYMBOL.LEFT,"1","2", "3", "4", "5",lv.SYMBOL.RIGHT, ""]

btnm = lv.btnmatrix(lv.scr_act())
btnm.set_map(map)
btnm.add_style(style_bg, 0)
btnm.add_style(style_btn, lv.PART.ITEMS)
btnm.add_event_cb(event_cb, lv.EVENT.VALUE_CHANGED, None)
btnm.set_size(225, 35)

# Allow selecting on one number at time
btnm.set_btn_ctrl_all(lv.btnmatrix.CTRL.CHECKABLE)
btnm.clear_btn_ctrl(0, lv.btnmatrix.CTRL.CHECKABLE)
btnm.clear_btn_ctrl(6, lv.btnmatrix.CTRL.CHECKABLE)

btnm.set_one_checked(True)
btnm.set_btn_ctrl(1, lv.btnmatrix.CTRL.CHECKED)

btnm.center()
```

API

Typedefs

typedef uint16_t **lv_btnmatrix_ctrl_t**

typedef bool (***lv_btnmatrix_btn_draw_cb_t**)(*lv_obj_t* *btnm, uint32_t btn_id, const lv_area_t *draw_area, const lv_area_t *clip_area)

Enums

enum **[anonymous]**

Type to store button control bits (disabled, hidden etc.) The first 3 bits are used to store the width

Values:

enumerator **_LV_BTNMATRIX_WIDTH**

Reserved to stire the size units

enumerator **LV_BTNMATRIX_CTRL_HIDDEN**

Button hidden

enumerator **LV_BTNMATRIX_CTRL_NO_REPEAT**

Do not repeat press this button.

enumerator **LV_BTNMATRIX_CTRL_DISABLED**

Disable this button.

enumerator **LV_BTNMATRIX_CTRL_CHECKABLE**

The button can be toggled.

enumerator **LV_BTNMATRIX_CTRL_CHECKED**

Button is currently toggled (e.g. checked).

enumerator **LV_BTNMATRIX_CTRL_CLICK_TRIG**

1: Send LV_EVENT_VALUE_CHANGE on CLICK, 0: Send LV_EVENT_VALUE_CHANGE on PRESS

enumerator **LV_BTNMATRIX_CTRL_POPOVER**

Show a popover when pressing this key

enumerator **LV_BTNMATRIX_CTRL_RECOLOR**

Enable text recoloring with `#color`

enumerator **_LV_BTNMATRIX_CTRL_RESERVED**

Reserved for later use

enumerator **LV_BTNMATRIX_CTRL_CUSTOM_1**

Custom free to use flag

enumerator **LV_BTNMATRIX_CTRL_CUSTOM_2**

Custom free to use flag

enum **lv_btnmatrix_draw_part_type_t**

`type` field in `lv_obj_draw_part_dsc_t` if `class_p = lv_btnmatrix_class` Used in `LV_EVENT_DRAW_PART_BEGIN` and `LV_EVENT_DRAW_PART_END`

Values:

enumerator **LV_BTNMATRIX_DRAW_PART_BTN**

The rectangle and label of buttons

Functions

LV_EXPORT_CONST_INT(LV_BTNMATRIX_BTN_NONE)

lv_obj_t ***lv_btnmatrix_create**(*lv_obj_t* *parent)

Create a button matrix object

参数 **parent** -- pointer to an object, it will be the parent of the new button matrix

返回 pointer to the created button matrix

void **lv_btnmatrix_set_map**(*lv_obj_t* *obj, const char *map[])

Set a new map. Buttons will be created/deleted according to the map. The button matrix keeps a reference to the map and so the string array must not be deallocated during the life of the matrix.

参数

- **obj** -- pointer to a button matrix object
- **map** -- pointer a string array. The last string has to be: "". Use "\n" to make a line break.

void **lv_btnmatrix_set_ctrl_map**(*lv_obj_t* *obj, const *lv_btnmatrix_ctrl_t* ctrl_map[])

Set the button control map (hidden, disabled etc.) for a button matrix. The control map array will be copied and so may be deallocated after this function returns.

参数

- **obj** -- pointer to a button matrix object
- **ctrl_map** -- pointer to an array of `lv_btn_ctrl_t` control bytes. The length of the array and position of the elements must match the number and order of the individual buttons (i.e. excludes newline entries). An element of the map should look like e.g.: `ctrl_map[0] = width | LV_BTNMATRIX_CTRL_NO_REPEAT | LV_BTNMATRIX_CTRL_TGL_ENABLE`

void **lv_btnmatrix_set_selected_btn**(*lv_obj_t* *obj, uint16_t btn_id)

Set the selected buttons

参数

- **obj** -- pointer to button matrix object
- **btn_id** -- 0 based index of the button to modify. (Not counting new lines)

void **lv_btnmatrix_set_btn_ctrl**(*lv_obj_t* *obj, uint16_t btn_id, *lv_btnmatrix_ctrl_t* ctrl)

Set the attributes of a button of the button matrix

参数

- **obj** -- pointer to button matrix object
- **btn_id** -- 0 based index of the button to modify. (Not counting new lines)
- **ctrl** -- OR-ed attributs. E.g. `LV_BTNMATRIX_CTRL_NO_REPEAT | LV_BTNMATRIX_CTRL_CHECKABLE`

void **lv_btnmatrix_clear_btn_ctrl**(*lv_obj_t* *obj, uint16_t btn_id, *lv_btnmatrix_ctrl_t* ctrl)

Clear the attributes of a button of the button matrix

参数

- **obj** -- pointer to button matrix object
- **btn_id** -- 0 based index of the button to modify. (Not counting new lines)
- **ctrl** -- OR-ed attributs. E.g. `LV_BTNMATRIX_CTRL_NO_REPEAT | LV_BTNMATRIX_CTRL_CHECKABLE`

void **lv_btnmatrix_set_btn_ctrl_all**(*lv_obj_t* *obj, *lv_btnmatrix_ctrl_t* ctrl)

Set attributes of all buttons of a button matrix

参数

- **obj** -- pointer to a button matrix object
- **ctrl** -- attribute(s) to set from `lv_btnmatrix_ctrl_t`. Values can be ORed.

void **lv_btnmatrix_clear_btn_ctrl_all**(*lv_obj_t* *obj, *lv_btnmatrix_ctrl_t* ctrl)

Clear the attributes of all buttons of a button matrix

参数

- **obj** -- pointer to a button matrix object
- **ctrl** -- attribute(s) to set from `lv_btnmatrix_ctrl_t`. Values can be ORed.
- **en** -- true: set the attributes; false: clear the attributes

void **lv_btnmatrix_set_btn_width**(*lv_obj_t* *obj, uint16_t btn_id, uint8_t width)

Set a single button's relative width. This method will cause the matrix be regenerated and is a relatively expensive operation. It is recommended that initial width be specified using `lv_btnmatrix_set_ctrl_map` and this method only be used for dynamic changes.

参数

- **obj** -- pointer to button matrix object

- **btn_id** -- 0 based index of the button to modify.
- **width** -- relative width compared to the buttons in the same row. [1..7]

void **lv_btnmatrix_set_one_checked**(*lv_obj_t* *obj, bool en)

Make the button matrix like a selector widget (only one button may be checked at a time). `LV_BTNMATRIX_CTRL_CHECKABLE` must be enabled on the buttons to be selected using `lv_btnmatrix_set_ctrl()` or *lv_btnmatrix_set_btn_ctrl_all()*.

参数

- **obj** -- pointer to a button matrix object
- **en** -- whether "one check" mode is enabled

const char ****lv_btnmatrix_get_map**(const *lv_obj_t* *obj)

Get the current map of a button matrix

参数 **obj** -- pointer to a button matrix object

返回 the current map

uint16_t **lv_btnmatrix_get_selected_btn**(const *lv_obj_t* *obj)

Get the index of the lastly "activated" button by the user (pressed, released, focused etc) Useful in the `event_cb` to get the text of the button, check if hidden etc.

参数 **obj** -- pointer to button matrix object

返回 index of the last released button (LV_BTNMATRIX_BTN_NONE: if unset)

const char ***lv_btnmatrix_get_btn_text**(const *lv_obj_t* *obj, uint16_t btn_id)

Get the button's text

参数

- **obj** -- pointer to button matrix object
- **btn_id** -- the index a button not counting new line characters.

返回 text of btn_index' button

bool **lv_btnmatrix_has_btn_ctrl**(*lv_obj_t* *obj, uint16_t btn_id, *lv_btnmatrix_ctrl_t* ctrl)

Get the whether a control value is enabled or disabled for button of a button matrix

参数

- **obj** -- pointer to a button matrix object
- **btn_id** -- the index of a button not counting new line characters.
- **ctrl** -- control values to check (ORed value can be used)

返回 true: the control attribute is enabled false: disabled

bool **lv_btnmatrix_get_one_checked**(const *lv_obj_t* *obj)

Tell whether "one check" mode is enabled or not.

参数 **obj** -- Button matrix object

返回 true: ”one check” mode is enabled; false: disabled

Variables

const lv_obj_class_t **lv_btnmatrix_class**

struct **lv_btnmatrix_t**

Public Members

lv_obj_t **obj**

const char ****map_p**

lv_area_t ***button_areas**

lv_btnmatrix_ctrl_t ***ctrl_bits**

uint16_t **btn_cnt**

uint16_t **row_cnt**

uint16_t **btn_id_sel**

uint8_t **one_check**

Canvas（画布）(lv_canvas)

Overview（概述）

A Canvas inherits from *Image* where the user can draw anything. Rectangles, texts, images, lines, arcs can be drawn here using lvgl’s drawing engine. Additionally ”effects” can be applied, such as rotation, zoom and blur.

Canvas 继承自*Image*，用户可以在其中绘制任何内容。矩形、文本、图像、线条、圆弧可以在这里使用 lvgl 的绘图引擎绘制。此外，可以应用“效果”，例如旋转、缩放和模糊。

Parts and Styles（零件和风格）

`LV_PART_MAIN` Uses the typical rectangle style properties and image style properties.

`LV_PART_MAIN` 使用典型的矩形样式属性和图像样式属性。

Usage（用法）

Buffer（缓冲区）

The Canvas needs a buffer in which stores the drawn image. To assign a buffer to a Canvas, use `lv_canvas_set_buffer(canvas, buffer, width, height, LV_IMG_CF_...)`. Where `buffer` is a static buffer (not just a local variable) to hold the image of the canvas. For example, `static lv_color_t buffer[LV_CANVAS_BUF_SIZE_TRUE_COLOR(width, height)]`. `LV_CANVAS_BUF_SIZE_...` macros help to determine the size of the buffer with different color formats.

The canvas supports all the built-in color formats like `LV_IMG_CF_TRUE_COLOR` or `LV_IMG_CF_INDEXED_2BIT`. See the full list in the Color formats section.

Canvas 需要一个缓冲区来存储绘制的图像。要为 Canvas 分配缓冲区，请使用 `lv_canvas_set_buffer(canvas, buffer, width, height, LV_IMG_CF_...)`。其中 `buffer` 是一个静态缓冲区（不仅仅是一个局部变量）来保存画布的图像。例如，静态 `lv_color_t` 缓冲区 `[LV_CANVAS_BUF_SIZE_TRUE_COLOR(width, height)]`。`LV_CANVAS_BUF_SIZE_...` 宏有助于确定具有不同颜色格式的缓冲区的大小。

画布支持所有内置颜色格式，如“LV_IMG_CF_TRUE_COLOR”或“LV_IMG_CF_INDEXED_2BIT”。请参阅 颜色格式 部分中的完整列表。

Indexed colors（颜色索引）

For `LV_IMG_CF_INDEXED_1/2/4/8` color formats a palette needs to be initialized with `lv_canvas_set_palette(canvas, 3, LV_COLOR_RED)`. It sets pixels with *index=3* to red.

对于“LV_IMG_CF_INDEXED_1/2/4/8”颜色格式，调色板需要用 `lv_canvas_set_palette(canvas, 3, LV_COLOR_RED)` 初始化。它将 *index=3* 的像素设置为红色。

Drawing（画画）

To set a pixel on the canvas, use `lv_canvas_set_px(canvas, x, y, LV_COLOR_RED)`. With `LV_IMG_CF_INDEXED_...` or `LV_IMG_CF_ALPHA_...`, the index of the color or the alpha value needs to be passed as color. E.g. `lv_color_t c; c.full = 3;`

`lv_canvas_fill_bg(canvas, LV_COLOR_BLUE, LV_OPA_50)` fills the whole canvas to blue with 50% opacity. Note that if the current color format doesn't support colors (e.g. `LV_IMG_CF_ALPHA_2BIT`) the color will be ignored. Similarly, if opacity is not supported (e.g. `LV_IMG_CF_TRUE_COLOR`) it will be ignored.

An array of pixels can be copied to the canvas with `lv_canvas_copy_buf(canvas, buffer_to_copy, x, y, width, height)`. The color format of the buffer and the canvas need to match.

要在画布上设置像素，请使用 `lv_canvas_set_px(canvas, x, y, LV_COLOR_RED)`。使用 `LV_IMG_CF_INDEXED_...` 或 `LV_IMG_CF_ALPHA_...`，颜色的索引或 alpha 值需要作为颜色传递。

例如。`lv_color_t c; c.full = 3;`

`lv_canvas_fill_bg(canvas, LV_COLOR_BLUE, LV_OPA_50)` 将整个画布填充为蓝色，不透明度为50%。请注意，如果当前颜色格式不支持颜色（例如 `LV_IMG_CF_ALPHA_2BIT`），则颜色将被忽略。同样，如果不支持不透明度（例如 `LV_IMG_CF_TRUE_COLOR`），它将被忽略。

可以使用 `lv_canvas_copy_buf(canvas, buffer_to_copy, x, y, width, height)` 将像素数组复制到画布。缓冲区和画布的颜色格式需要匹配。

To draw something to the canvas use

- `lv_canvas_draw_rect(canvas, x, y, width, heigth, &draw_dsc)`
- `lv_canvas_draw_text(canvas, x, y, max_width, &draw_dsc, txt)`
- `lv_canvas_draw_img(canvas, x, y, &img_src, &draw_dsc)`
- `lv_canvas_draw_line(canvas, point_array, point_cnt, &draw_dsc)`
- `lv_canvas_draw_polygon(canvas, points_array, point_cnt, &draw_dsc)`
- `lv_canvas_draw_arc(canvas, x, y, radius, start_angle, end_angle, &draw_dsc)`

`draw_dsc` is a `lv_draw_rect/label/img/line/arc_dsc_t` variable which should be first initialized with one of `lv_draw_rect/label/img/line/arc_dsc_init()` and then modified with the desired colors and other values.

The draw function can draw to any color format. For example, it's possible to draw a text to an `LV_IMG_VF_ALPHA_8BIT` canvas and use the result image as a *draw mask* later.

要在画布上绘制一些东西，请使用

- `lv_canvas_draw_rect(canvas, x, y, width, heigth, &draw_dsc)`
- `lv_canvas_draw_text(canvas, x, y, max_width, &draw_dsc, txt)`
- `lv_canvas_draw_img(canvas, x, y, &img_src, &draw_dsc)`
- `lv_canvas_draw_line(canvas, point_array, point_cnt, &draw_dsc)`
- `lv_canvas_draw_polygon(canvas, points_array, point_cnt, &draw_dsc)`
- `lv_canvas_draw_arc(canvas, x, y, radius, start_angle, end_angle, &draw_dsc)`

`draw_dsc` 是一个 `lv_draw_rect/label/img/line/arc_dsc_t` 变量，它应该首先使用 `lv_draw_rect/label/img/line/arc_dsc_init()` 中的一个进行初始化，然后使用所需的颜色和其他值进行修改。

draw 函数可以绘制成任何颜色格式。例如，可以在 “LV_IMG_VF_ALPHA_8BIT” 画布上绘制文本，然后将结果图像用作绘制蒙版。

Transformations（变换）

`lv_canvas_transform()` can be used to rotate and/or scale the image of an image and store the result on the canvas. The function needs the following parameters:

- `canvas` pointer to a canvas object to store the result of the transformation.
- `img pointer` to an image descriptor to transform. Can be the image descriptor of an other canvas too (`lv_canvas_get_img()`).
- `angle` the angle of rotation (0..3600), 0.1 deg resolution
- `zoom` zoom factor (256: no zoom, 512: double size, 128: half size);
- `offset_x` offset X to tell where to put the result data on destination canvas
- `offset_y` offset X to tell where to put the result data on destination canvas
- `pivot_x` pivot X of rotation. Relative to the source canvas. Set to `source width / 2` to rotate around the center
- `pivot_y` pivot Y of rotation. Relative to the source canvas. Set to `source height / 2` to rotate around the center
- `antialias` true: apply anti-aliasing during the transformation. Looks better but slower.

Note that a canvas can't be rotated on itself. You need a source and destination canvas or image.

`lv_canvas_transform()` 可用于旋转和/或缩放图像的图像并将结果存储在画布上。该函数需要以下参数：

- `canvas` 指向一个画布对象的指针，用于存储转换的结果。
- 指向要转换的图像描述符的 “img 指针”。也可以是其他画布的图像描述符（`lv_canvas_get_img()`）。
- `angle` 旋转角度 (0..3600)，0.1 度分辨率
- `zoom` 缩放系数（256：无缩放，512：双倍尺寸，128：半尺寸）；
- `offset_x` 偏移 X 来告诉将结果数据放在目标画布上的什么位置
- `offset_y` 偏移 X 来告诉将结果数据放在目标画布上的什么位置
- `pivot_x` 旋转的枢轴 X。相对于源画布。设置为 `source width / 2` 以围绕中心旋转
- `pivot_y` 旋转轴 Y。相对于源画布。设置为 `source height / 2` 以围绕中心旋转
- `antialias` true：在转换过程中应用抗锯齿。看起来更好但更慢。

请注意，画布不能自行旋转。您需要一个源和目标画布或图像。

Blur（糊化）

A given area of the canvas can be blurred horizontally with `lv_canvas_blur_hor(canvas, &area, r)` or vertically with `lv_canvas_blur_ver(canvas, &area, r)`. `r` is the radius of the blur (greater value means more intensive burring). `area` is the area where the blur should be applied (interpreted relative to the canvas).

画布的给定区域可以使用 `lv_canvas_blur_hor(canvas, &area, r)` 进行水平模糊处理，或者使用 `lv_canvas_blur_ver(canvas, &area, r)` 进行垂直模糊处理。`r` 是模糊的半径（值越大意味着毛刺越强）。`area` 是应该应用模糊的区域（相对于画布进行解释）。

Events（事件）

No special events are sent by canvas objects. The same events are sent as for the

See the events of the *Images* too.

Learn more about *Events*.

画布对象不会发送特殊事件。

也可以查看*Images* 的事件。

详细了解事件。

Keys（按键）

No *Keys* are processed by the object type.

Learn more about *Keys*.

对象类型不处理 *Keys*。

了解有关*Keys* 的更多信息。

Example

Drawing on the Canvas and rotate

```
#include "../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES


#define CANVAS_WIDTH  200
#define CANVAS_HEIGHT  150

void lv_example_canvas_1(void)
```

(下页继续)

(续上页)

```
{
    lv_draw_rect_dsc_t rect_dsc;
    lv_draw_rect_dsc_init(&rect_dsc);
    rect_dsc.radius = 10;
    rect_dsc.bg_opa = LV_OPA_COVER;
    rect_dsc.bg_grad.dir = LV_GRAD_DIR_HOR;
    rect_dsc.bg_grad.stops[0].color = lv_palette_main(LV_PALETTE_RED);
    rect_dsc.bg_grad.stops[1].color = lv_palette_main(LV_PALETTE_BLUE);
    rect_dsc.border_width = 2;
    rect_dsc.border_opa = LV_OPA_90;
    rect_dsc.border_color = lv_color_white();
    rect_dsc.shadow_width = 5;
    rect_dsc.shadow_ofs_x = 5;
    rect_dsc.shadow_ofs_y = 5;

    lv_draw_label_dsc_t label_dsc;
    lv_draw_label_dsc_init(&label_dsc);
    label_dsc.color = lv_palette_main(LV_PALETTE_ORANGE);

    static lv_color_t cbuf[LV_CANVAS_BUF_SIZE_TRUE_COLOR(CANVAS_WIDTH, CANVAS_
↪HEIGHT)];

    lv_obj_t * canvas = lv_canvas_create(lv_scr_act());
    lv_canvas_set_buffer(canvas, cbuf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_IMG_CF_TRUE_
↪COLOR);
    lv_obj_center(canvas);
    lv_canvas_fill_bg(canvas, lv_palette_lighten(LV_PALETTE_GREY, 3), LV_OPA_COVER);

    lv_canvas_draw_rect(canvas, 70, 60, 100, 70, &rect_dsc);

    lv_canvas_draw_text(canvas, 40, 20, 100, &label_dsc, "Some text on text canvas");

    /*Test the rotation. It requires another buffer where the original image is␣
↪stored.
     *So copy the current image to buffer and rotate it to the canvas*/
    static lv_color_t cbuf_tmp[CANVAS_WIDTH * CANVAS_HEIGHT];
    memcpy(cbuf_tmp, cbuf, sizeof(cbuf_tmp));
    lv_img_dsc_t img;
    img.data = (void *)cbuf_tmp;
    img.header.cf = LV_IMG_CF_TRUE_COLOR;
    img.header.w = CANVAS_WIDTH;
    img.header.h = CANVAS_HEIGHT;
```

(下页继续)

(续上页)

```
    lv_canvas_fill_bg(canvas, lv_palette_lighten(LV_PALETTE_GREY, 3), LV_OPA_COVER);
    lv_canvas_transform(canvas, &img, 120, LV_IMG_ZOOM_NONE, 0, 0, CANVAS_WIDTH / 2,␣
↪CANVAS_HEIGHT / 2, true);
}

#endif
```

```
_CANVAS_WIDTH = 200
_CANVAS_HEIGHT = 150
LV_IMG_ZOOM_NONE = 256

rect_dsc = lv.draw_rect_dsc_t()
rect_dsc.init()
rect_dsc.radius = 10
rect_dsc.bg_opa = lv.OPA.COVER
rect_dsc.bg_grad.dir = lv.GRAD_DIR.HOR
rect_dsc.bg_grad.stops[0].color = lv.palette_main(lv.PALETTE.RED)
rect_dsc.bg_grad.stops[1].color = lv.palette_main(lv.PALETTE.BLUE)
rect_dsc.border_width = 2
rect_dsc.border_opa = lv.OPA._90
rect_dsc.border_color = lv.color_white()
rect_dsc.shadow_width = 5
rect_dsc.shadow_ofs_x = 5
rect_dsc.shadow_ofs_y = 5

label_dsc = lv.draw_label_dsc_t()
label_dsc.init()
label_dsc.color = lv.palette_main(lv.PALETTE.YELLOW)

cbuf = bytearray(_CANVAS_WIDTH * _CANVAS_HEIGHT * 4)

canvas = lv.canvas(lv.scr_act())
canvas.set_buffer(cbuf, _CANVAS_WIDTH, _CANVAS_HEIGHT, lv.img.CF.TRUE_COLOR)
canvas.center()
canvas.fill_bg(lv.palette_lighten(lv.PALETTE.GREY, 3), lv.OPA.COVER)

canvas.draw_rect(70, 60, 100, 70, rect_dsc)
canvas.draw_text(40, 20, 100, label_dsc, "Some text on text canvas")

# Test the rotation. It requires another buffer where the original image is stored.
# So copy the current image to buffer and rotate it to the canvas

img = lv.img_dsc_t()
```

(下页继续)

(续上页)

```
img.data = cbuf[:]
img.header.cf = lv.img.CF.TRUE_COLOR
img.header.w = _CANVAS_WIDTH
img.header.h = _CANVAS_HEIGHT

canvas.fill_bg(lv.palette_lighten(lv.PALETTE.GREY, 3), lv.OPA.COVER)
canvas.transform(img, 30, LV_IMG_ZOOM_NONE, 0, 0, _CANVAS_WIDTH // 2, _CANVAS_HEIGHT /
↪/ 2, True)
```

Transparent Canvas with chroma keying

```
#include "../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH  50
#define CANVAS_HEIGHT  50

/**
 * Create a transparent canvas with Chroma keying and indexed color format (palette).
 */
void lv_example_canvas_2(void)
{
    /*Create a button to better see the transparency*/
    lv_btn_create(lv_scr_act());

    /*Create a buffer for the canvas*/
    static lv_color_t cbuf[LV_CANVAS_BUF_SIZE_INDEXED_1BIT(CANVAS_WIDTH, CANVAS_
↪HEIGHT)];

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_scr_act());
    lv_canvas_set_buffer(canvas, cbuf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_IMG_CF_INDEXED_
↪1BIT);
    lv_canvas_set_palette(canvas, 0, LV_COLOR_CHROMA_KEY);
    lv_canvas_set_palette(canvas, 1, lv_palette_main(LV_PALETTE_RED));

    /*Create colors with the indices of the palette*/
    lv_color_t c0;
    lv_color_t c1;

    c0.full = 0;
    c1.full = 1;
```

(下页继续)

(续上页)

```
    /*Red background (There is no dedicated alpha channel in indexed images so LV_OPA_
↪COVER is ignored)*/
    lv_canvas_fill_bg(canvas, c1, LV_OPA_COVER);

    /*Create hole on the canvas*/
    uint32_t x;
    uint32_t y;
    for( y = 10; y < 30; y++) {
        for( x = 5; x < 20; x++) {
            lv_canvas_set_px_color(canvas, x, y, c0);
        }
    }

}
#endif
```

```
CANVAS_WIDTH   = 50
CANVAS_HEIGHT  = 50
LV_COLOR_CHROMA_KEY = lv.color_hex(0x00ff00)

def LV_IMG_BUF_SIZE_ALPHA_1BIT(w, h):
    return int(((w / 8) + 1) * h)

def LV_IMG_BUF_SIZE_INDEXED_1BIT(w, h):
    return LV_IMG_BUF_SIZE_ALPHA_1BIT(w, h) + 4 * 2

def LV_CANVAS_BUF_SIZE_INDEXED_1BIT(w, h):
    return LV_IMG_BUF_SIZE_INDEXED_1BIT(w, h)

#
# Create a transparent canvas with Chroma keying and indexed color format (palette).
#

# Create a button to better see the transparency
btn=lv.btn(lv.scr_act())

# Create a buffer for the canvas
cbuf= bytearray(LV_CANVAS_BUF_SIZE_INDEXED_1BIT(CANVAS_WIDTH, CANVAS_HEIGHT))

# Create a canvas and initialize its palette
canvas = lv.canvas(lv.scr_act())
canvas.set_buffer(cbuf, CANVAS_WIDTH, CANVAS_HEIGHT, lv.img.CF.INDEXED_1BIT)
```

(下页继续)

(续上页)

```
canvas.set_palette(0, LV_COLOR_CHROMA_KEY)
canvas.set_palette(1, lv.palette_main(lv.PALETTE.RED))

# Create colors with the indices of the palette
c0 = lv.color_t()
c1 = lv.color_t()

c0.full = 0
c1.full = 1

# Red background (There is no dedicated alpha channel in indexed images so LV_OPA_
↪COVER is ignored)
canvas.fill_bg(c1, lv.OPA.COVER)

# Create hole on the canvas
for y in range(10,30):
    for x in range(5,20):
        canvas.set_px(x, y, c0)
```

API

Functions

lv_obj_t ***lv_canvas_create**(*lv_obj_t* *parent)

Create a canvas object

参数 **parent** -- pointer to an object, it will be the parent of the new canvas

返回 pointer to the created canvas

void **lv_canvas_set_buffer**(*lv_obj_t* *canvas, void *buf, lv_coord_t w, lv_coord_t h, *lv_img_cf_t* cf)

Set a buffer for the canvas.

参数

- **buf** -- a buffer where the content of the canvas will be. The required size is (lv_img_color_format_get_px_size(cf) * w) / 8 * h) It can be allocated with `lv_mem_alloc()` or it can be statically allocated array (e.g. static lv_color_t buf[100*50]) or it can be an address in RAM or external SRAM
- **canvas** -- pointer to a canvas object
- **w** -- width of the canvas
- **h** -- height of the canvas
- **cf** -- color format. `LV_IMG_CF_...`

void **lv_canvas_set_px_color**(*lv_obj_t* *canvas, lv_coord_t x, lv_coord_t y, lv_color_t c)

Set the color of a pixel on the canvas

参数

- **canvas** --
- **x** -- x coordinate of the point to set
- **y** -- x coordinate of the point to set
- **c** -- color of the pixel

static inline void **lv_canvas_set_px**(*lv_obj_t* *canvas, lv_coord_t x, lv_coord_t y, lv_color_t c)

DEPRECATED: added only for backward compatibility

void **lv_canvas_set_px_opa**(*lv_obj_t* *canvas, lv_coord_t x, lv_coord_t y, lv_opa_t opa)

Set the opacity of a pixel on the canvas

参数

- **canvas** --
- **x** -- x coordinate of the point to set
- **y** -- x coordinate of the point to set
- **opa** -- opacity of the pixel (0..255)

void **lv_canvas_set_palette**(*lv_obj_t* *canvas, uint8_t id, lv_color_t c)

Set the palette color of a canvas with index format. Valid only for `LV_IMG_CF_INDEXED1/2/4/8`

参数

- **canvas** -- pointer to canvas object
- **id** -- the palette color to set:
  - for `LV_IMG_CF_INDEXED1`: 0..1
  - for `LV_IMG_CF_INDEXED2`: 0..3
  - for `LV_IMG_CF_INDEXED4`: 0..15
  - for `LV_IMG_CF_INDEXED8`: 0..255
- **c** -- the color to set

lv_color_t **lv_canvas_get_px**(*lv_obj_t* *canvas, lv_coord_t x, lv_coord_t y)

Get the color of a pixel on the canvas

参数

- **canvas** --
- **x** -- x coordinate of the point to set

- **y** -- x coordinate of the point to set

返回 color of the point

lv_img_dsc_t ***lv_canvas_get_img**(*lv_obj_t* *canvas)

Get the image of the canvas as a pointer to an *lv_img_dsc_t* variable.

参数 **canvas** -- pointer to a canvas object

返回 pointer to the image descriptor.

void **lv_canvas_copy_buf**(*lv_obj_t* *canvas, const void *to_copy, lv_coord_t x, lv_coord_t y, lv_coord_t w, lv_coord_t h)

Copy a buffer to the canvas

参数

- **canvas** -- pointer to a canvas object
- **to_copy** -- buffer to copy. The color format has to match with the canvas's buffer color format
- **x** -- left side of the destination position
- **y** -- top side of the destination position
- **w** -- width of the buffer to copy
- **h** -- height of the buffer to copy

void **lv_canvas_transform**(*lv_obj_t* *canvas, *lv_img_dsc_t* *img, int16_t angle, uint16_t zoom, lv_coord_t offset_x, lv_coord_t offset_y, int32_t pivot_x, int32_t pivot_y, bool antialias)

Transform and image and store the result on a canvas.

参数

- **canvas** -- pointer to a canvas object to store the result of the transformation.
- **img** -- pointer to an image descriptor to transform. Can be the image descriptor of an other canvas too (*lv_canvas_get_img()*).
- **angle** -- the angle of rotation (0..3600), 0.1 deg resolution
- **zoom** -- zoom factor (256 no zoom);
- **offset_x** -- offset X to tell where to put the result data on destination canvas
- **offset_y** -- offset X to tell where to put the result data on destination canvas
- **pivot_x** -- pivot X of rotation. Relative to the source canvas Set to `source width / 2` to rotate around the center
- **pivot_y** -- pivot Y of rotation. Relative to the source canvas Set to `source height / 2` to rotate around the center
- **antialias** -- apply anti-aliasing during the transformation. Looks better but slower.

void **lv_canvas_blur_hor**(*lv_obj_t* *canvas, const lv_area_t *area, uint16_t r)

Apply horizontal blur on the canvas

参数

- **canvas** -- pointer to a canvas object
- **area** -- the area to blur. If NULL the whole canvas will be blurred.
- **r** -- radius of the blur

void **lv_canvas_blur_ver**(*lv_obj_t* *canvas, const lv_area_t *area, uint16_t r)

Apply vertical blur on the canvas

参数

- **canvas** -- pointer to a canvas object
- **area** -- the area to blur. If NULL the whole canvas will be blurred.
- **r** -- radius of the blur

void **lv_canvas_fill_bg**(*lv_obj_t* *canvas, lv_color_t color, lv_opa_t opa)

Fill the canvas with color

参数

- **canvas** -- pointer to a canvas
- **color** -- the background color
- **opa** -- the desired opacity

void **lv_canvas_draw_rect**(*lv_obj_t* *canvas, lv_coord_t x, lv_coord_t y, lv_coord_t w, lv_coord_t h, const lv_draw_rect_dsc_t *draw_dsc)

Draw a rectangle on the canvas

参数

- **canvas** -- pointer to a canvas object
- **x** -- left coordinate of the rectangle
- **y** -- top coordinate of the rectangle
- **w** -- width of the rectangle
- **h** -- height of the rectangle
- **draw_dsc** -- descriptor of the rectangle

void **lv_canvas_draw_text**(*lv_obj_t* *canvas, lv_coord_t x, lv_coord_t y, lv_coord_t max_w, lv_draw_label_dsc_t *draw_dsc, const char *txt)

Draw a text on the canvas.

参数

- **canvas** -- pointer to a canvas object
- **x** -- left coordinate of the text
- **y** -- top coordinate of the text
- **max_w** -- max width of the text. The text will be wrapped to fit into this size
- **draw_dsc** -- pointer to a valid label descriptor `lv_draw_label_dsc_t`
- **txt** -- text to display

void **lv_canvas_draw_img**(*lv_obj_t* *canvas, lv_coord_t x, lv_coord_t y, const void *src, const lv_draw_img_dsc_t *draw_dsc)

Draw an image on the canvas

参数

- **canvas** -- pointer to a canvas object
- **x** -- left coordinate of the image
- **y** -- top coordinate of the image
- **src** -- image source. Can be a pointer an *lv_img_dsc_t* variable or a path an image.
- **draw_dsc** -- pointer to a valid label descriptor `lv_draw_img_dsc_t`

void **lv_canvas_draw_line**(*lv_obj_t* *canvas, const lv_point_t points[], uint32_t point_cnt, const lv_draw_line_dsc_t *draw_dsc)

Draw a line on the canvas

参数

- **canvas** -- pointer to a canvas object
- **points** -- point of the line
- **point_cnt** -- number of points
- **draw_dsc** -- pointer to an initialized `lv_draw_line_dsc_t` variable

void **lv_canvas_draw_polygon**(*lv_obj_t* *canvas, const lv_point_t points[], uint32_t point_cnt, const lv_draw_rect_dsc_t *draw_dsc)

Draw a polygon on the canvas

参数

- **canvas** -- pointer to a canvas object
- **points** -- point of the polygon
- **point_cnt** -- number of points
- **draw_dsc** -- pointer to an initialized `lv_draw_rect_dsc_t` variable

void **lv_canvas_draw_arc**(*lv_obj_t* *canvas, lv_coord_t x, lv_coord_t y, lv_coord_t r, int32_t start_angle, int32_t end_angle, const lv_draw_arc_dsc_t *draw_dsc)

Draw an arc on the canvas

参数

- **canvas** -- pointer to a canvas object
- **x** -- origo x of the arc
- **y** -- origo y of the arc
- **r** -- radius of the arc
- **start_angle** -- start angle in degrees
- **end_angle** -- end angle in degrees
- **draw_dsc** -- pointer to an initialized `lv_draw_line_dsc_t` variable

Variables

const lv_obj_class_t **lv_canvas_class**

struct **lv_canvas_t**

Public Members

lv_img_t **img**

lv_img_dsc_t **dsc**

Checkbox（复选框）(lv_checkbox)

Overview（概述）

The Checkbox object is created from a "tick box" and a label. When the Chackbox is clicked the tick box is toggled.

复选框 (Checkbox) 对象是从“勾选框”和标签创建的。当 Chackbox 被点击时，勾选框被切换。

Parts and Styles（部分和样式）

- `LV_PART_MAIN` The is the background of the Checkbox and it uses the text and all the typical backround style properties. `pad_column` adjusts the spacing between the tickbox and the label
- `LV_PART_INDICATOR` The "tick box" is a square that uses all the typical backround style properties. By default its size is equal to the height of the main part's font. Padding properties make the tick box larger in the respective directions.

The Checkbox is added to the default group (if it is set).

- `LV_PART_MAIN` 这是复选框的背景，它使用文本和所有典型的背景样式属性。`pad_column` 调整复选框和标签之间的间距
- `LV_PART_INDICATOR` "勾选框"是一个使用所有典型背景样式属性的正方形。默认情况下，它的大小等于主要部分字体的高度。填充属性使复选框在相应方向上变大。

复选框将添加到默认组（如果已设置）。

Usage（用法）

Text () 文本

The text can be modified with the `lv_checkbox_set_text(cb, "New text")` function and will be dynamically allocated.

To set a static text, use `lv_checkbox_set_static_text(cb, txt)`. This way, only a pointer to `txt` will be stored. The text then shouldn't be deallocated while the checkbox exists.

文本可以使用 `lv_checkbox_set_text(cb, "New text")` 函数进行修改，并将被动态分配。

要设置静态文本，使用 `lv_checkbox_set_static_text(cb, txt)`。这样，只会存储一个指向 `txt` 的指针。当复选框存在时，不应取消分配文本。

Check, uncheck, disable（选中，取消选中，禁用）

You can manually check, un-check, and disable the Checkbox by using the common state add/clear function:

您可以使用通用状态添加/清除功能手动选中、取消选中和禁用复选框：

```
lv_obj_add_state(cb, LV_STATE_CHECKED);   /*Make the chekbox checked*/
lv_obj_clear_state(cb, LV_STATE_CHECKED); /*MAke the checkbox unchecked*/
lv_obj_add_state(cb, LV_STATE_CHECKED | LV_STATE_DISABLED); /*Make the checkbox␣
↪checked and disabled*/
```

Events（事件）

- LV_EVENT_VALUE_CHANGED Sent when the checkbox is toggled.
- LV_EVENT_DRAW_PART_BEGIN and LV_EVENT_DRAW_PART_END are sent for the following types:
  - LV_CHECKBOX_DRAW_PART_BOX The tickbox of the checkbox
    - part: LV_PART_INDICATOR
    - draw_area: the area of the tickbox
    - rect_dsc

See the events of the *Base object* too.

Learn more about *Events*.

- LV_EVENT_VALUE_CHANGED 当复选框被切换时发送。
- 为以下类型发送 LV_EVENT_DRAW_PART_BEGIN 和 LV_EVENT_DRAW_PART_END：
  - LV_CHECKBOX_DRAW_PART_BOX 复选框的勾选框
    - 部分：LV_PART_INDICATOR
    - draw_area：勾选框的区域 -rect_dsc

参见*Base object* 的事件。

详细了解事件。

Keys（按键）

The following *Keys* are processed by the 'Buttons':

- LV_KEY_RIGHT/UP Go to toggled state if toggling is enabled
- LV_KEY_LEFT/DOWN Go to non-toggled state if toggling is enabled
- LV_KEY_ENTER Clicks the checkbox and toggles it

Note that, as usual, the state of LV_KEY_ENTER is translated to LV_EVENT_PRESSED/PRESSING/RELEASED etc.

Learn more about *Keys*.

以下键由“按钮”处理：

- LV_KEY_RIGHT/UP 如果启用切换，则转到切换状态
- LV_KEY_LEFT/DOWN 如果启用切换，则转到非切换状态
- LV_KEY_ENTER 单击复选框并切换它

请注意，像往常一样，“LV_KEY_ENTER” 的状态被转换为 “LV_EVENT_PRESSED/PRESSING/RELEASED” 等。

了解有关*Keys* 的更多信息。

Example

Simple Checkboxes

```
#include "../../lv_examples.h"
#if LV_USE_CHECKBOX && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        const char * txt = lv_checkbox_get_text(obj);
        const char * state = lv_obj_get_state(obj) & LV_STATE_CHECKED ? "Checked" :
↪"Unchecked";
        LV_LOG_USER("%s: %s", txt, state);
    }
}

void lv_example_checkbox_1(void)
{
    lv_obj_set_flex_flow(lv_scr_act(), LV_FLEX_FLOW_COLUMN);
    lv_obj_set_flex_align(lv_scr_act(), LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_START, LV_
↪FLEX_ALIGN_CENTER);

    lv_obj_t * cb;
    cb = lv_checkbox_create(lv_scr_act());
    lv_checkbox_set_text(cb, "Apple");
    lv_obj_add_event_cb(cb, event_handler, LV_EVENT_ALL, NULL);

    cb = lv_checkbox_create(lv_scr_act());
    lv_checkbox_set_text(cb, "Banana");
    lv_obj_add_state(cb, LV_STATE_CHECKED);
    lv_obj_add_event_cb(cb, event_handler, LV_EVENT_ALL, NULL);

    cb = lv_checkbox_create(lv_scr_act());
    lv_checkbox_set_text(cb, "Lemon");
    lv_obj_add_state(cb, LV_STATE_DISABLED);
    lv_obj_add_event_cb(cb, event_handler, LV_EVENT_ALL, NULL);
```

(下页继续)

(续上页)

```
    cb = lv_checkbox_create(lv_scr_act());
    lv_obj_add_state(cb, LV_STATE_CHECKED | LV_STATE_DISABLED);
    lv_checkbox_set_text(cb, "Melon\nand a new line");
    lv_obj_add_event_cb(cb, event_handler, LV_EVENT_ALL, NULL);

    lv_obj_update_layout(cb);
}

#endif
```

```
def event_handler(e):
    code = e.get_code()
    obj = e.get_target()
    if code == lv.EVENT.VALUE_CHANGED:
        txt = obj.get_text()
        if obj.get_state() & lv.STATE.CHECKED:
            state = "Checked"
        else:
            state = "Unchecked"
        print(txt + ":" + state)


lv.scr_act().set_flex_flow(lv.FLEX_FLOW.COLUMN)
lv.scr_act().set_flex_align(lv.FLEX_ALIGN.CENTER, lv.FLEX_ALIGN.START, lv.FLEX_ALIGN.
↪CENTER)

cb = lv.checkbox(lv.scr_act())
cb.set_text("Apple")
cb.add_event_cb(event_handler, lv.EVENT.ALL, None)

cb = lv.checkbox(lv.scr_act())
cb.set_text("Banana")
cb.add_state(lv.STATE.CHECKED)
cb.add_event_cb(event_handler, lv.EVENT.ALL, None)

cb = lv.checkbox(lv.scr_act())
cb.set_text("Lemon")
cb.add_state(lv.STATE.DISABLED)
cb.add_event_cb(event_handler, lv.EVENT.ALL, None)

cb = lv.checkbox(lv.scr_act())
cb.add_state(lv.STATE.CHECKED | lv.STATE.DISABLED)
```

(下页继续)

(续上页)

```
cb.set_text("Melon")
cb.add_event_cb(event_handler, lv.EVENT.ALL, None)

cb.update_layout()
```

Checkboxes as radio buttons

```
#include "../../lv_examples.h"
#if LV_USE_CHECKBOX && LV_BUILD_EXAMPLES

static lv_style_t style_radio;
static lv_style_t style_radio_chk;
static uint32_t active_index_1 = 0;
static uint32_t active_index_2 = 0;

static void radio_event_handler(lv_event_t * e)
{
    uint32_t * active_id = lv_event_get_user_data(e);
    lv_obj_t * cont = lv_event_get_current_target(e);
    lv_obj_t * act_cb = lv_event_get_target(e);
    lv_obj_t * old_cb = lv_obj_get_child(cont, *active_id);

    /*Do nothing if the container was clicked*/
    if(act_cb == cont) return;

    lv_obj_clear_state(old_cb, LV_STATE_CHECKED);   /*Uncheck the previous radio␣
↪button*/
    lv_obj_add_state(act_cb, LV_STATE_CHECKED);     /*Uncheck the current radio␣
↪button*/

    *active_id = lv_obj_get_index(act_cb);

    LV_LOG_USER("Selected radio buttons: %d, %d", (int)active_index_1, (int)active_
↪index_2);
}


static void radiobutton_create(lv_obj_t * parent, const char * txt)
{
    lv_obj_t * obj = lv_checkbox_create(parent);
    lv_checkbox_set_text(obj, txt);
```

(下页继续)

(续上页)

```
    lv_obj_add_flag(obj, LV_OBJ_FLAG_EVENT_BUBBLE);
    lv_obj_add_style(obj, &style_radio, LV_PART_INDICATOR);
    lv_obj_add_style(obj, &style_radio_chk, LV_PART_INDICATOR | LV_STATE_CHECKED);
}

/**
 * Checkboxes as radio buttons
 */
void lv_example_checkbox_2(void)
{
    /* The idea is to enable `LV_OBJ_FLAG_EVENT_BUBBLE` on checkboxes and process the
     * `LV_EVENT_CLICKED` on the container.
     * A variable is passed as event user data where the index of the active
     * radiobutton is saved */


    lv_style_init(&style_radio);
    lv_style_set_radius(&style_radio, LV_RADIUS_CIRCLE);

    lv_style_init(&style_radio_chk);
    lv_style_set_bg_img_src(&style_radio_chk, NULL);

    uint32_t i;
    char buf[32];

    lv_obj_t * cont1 = lv_obj_create(lv_scr_act());
    lv_obj_set_flex_flow(cont1, LV_FLEX_FLOW_COLUMN);
    lv_obj_set_size(cont1, lv_pct(40), lv_pct(80));
    lv_obj_add_event_cb(cont1, radio_event_handler, LV_EVENT_CLICKED, &active_index_
↪1);

    for (i = 0;i < 5;i++) {
        lv_snprintf(buf, sizeof(buf), "A %d", (int)i + 1);
        radiobutton_create(cont1, buf);

    }
    /*Make the first checkbox checked*/
    lv_obj_add_state(lv_obj_get_child(cont1, 0), LV_STATE_CHECKED);

    lv_obj_t * cont2 = lv_obj_create(lv_scr_act());
    lv_obj_set_flex_flow(cont2, LV_FLEX_FLOW_COLUMN);
    lv_obj_set_size(cont2, lv_pct(40), lv_pct(80));
    lv_obj_set_x(cont2, lv_pct(50));
```

(下页继续)

(续上页)

```
    lv_obj_add_event_cb(cont2, radio_event_handler, LV_EVENT_CLICKED, &active_index_
↪2);

    for (i = 0;i < 3;i++) {
        lv_snprintf(buf, sizeof(buf), "B %d", (int)i + 1);
        radiobutton_create(cont2, buf);
    }

    /*Make the first checkbox checked*/
    lv_obj_add_state(lv_obj_get_child(cont2, 0), LV_STATE_CHECKED);
}


#endif
```

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs/100ask_lvgl_
↪docs/examples/widgets/checkbox/lv_example_checkbox_2.py
```

API

Enums

enum **lv_checkbox_draw_part_type_t**

`type` field in `lv_obj_draw_part_dsc_t` if `class_p = lv_checkbox_class` Used in `LV_EVENT_DRAW_PART_BEGIN` and `LV_EVENT_DRAW_PART_END`

Values:

enumerator **LV_CHECKBOX_DRAW_PART_BOX**

The tick box

Functions

lv_obj_t ***lv_checkbox_create**(*lv_obj_t* *parent)

Create a check box object

参数 **parent** -- pointer to an object, it will be the parent of the new button

返回 pointer to the created check box

void **lv_checkbox_set_text**(*lv_obj_t* *obj, const char *txt)

Set the text of a check box. `txt` will be copied and may be deallocated after this function returns.

参数

- **cb** -- pointer to a check box
- **txt** -- the text of the check box. NULL to refresh with the current text.

void **lv_checkbox_set_text_static**(*lv_obj_t* *obj, const char *txt)

Set the text of a check box. `txt` must not be deallocated during the life of this checkbox.

参数

- **cb** -- pointer to a check box
- **txt** -- the text of the check box.

const char ***lv_checkbox_get_text**(const *lv_obj_t* *obj)

Get the text of a check box

参数 **cb** -- pointer to check box object

返回 pointer to the text of the check box

Variables

const lv_obj_class_t **lv_checkbox_class**

struct **lv_checkbox_t**

Public Members

lv_obj_t **obj**

char ***txt**

uint32_t **static_txt**

Drop-down list（下拉列表）(lv_dropdown)

Overview（概述）

The drop-down list allows the user to select one value from a list.

The drop-down list is closed by default and displays a single value or a predefined text. When activated (by click on the drop-down list), a list is created from which the user may select one option. When the user selects a new value, the list is deleted again.

The Drop-down list is added to the default group (if it is set). Besides the Drop-down list is an editable object to allow selecting an option with encoder navigation too.

下拉列表允许用户从列表中选择一个值。

下拉列表的选项表默认是关闭的，他可以选项可以是单个值或预定义文本。当单击下拉列表后，其将创建一个列表，用户可以从中选择一个选项。当用户选择了一个值后，该列表将再次被删除。

下拉列表将已经添加到默认组了。此外，下拉列表是一个可编辑的对象，允许通过编码器导航选项。

Parts and Styles（部分和样式）

The Dropdown widget is built from the elements: ”button” and ”list” (both not related to the button and list widgets)

下拉组件由以下元素构建：“按钮”和“列表”（均与按钮和列表组件无关）

Button（按钮）

- `LV_PART_MAIN` The background of the button. Uses the typical background properties and text properties for the text on it.
- `LV_PART_INDICATOR` Typically an arrow symbol that can be an image or a text (`LV_SYMBOL`).

The button goes to `LV_STATE_CHECKED` when its opened.

- `LV_PART_MAIN` 按钮的背景。对其上面的文本使用典型的背景属性和文本属性。
- `LV_PART_INDICATOR` 通常是一个箭头符号，可以是图像或文本（`LV_SYMBOL...`）。

按钮在打开时，会设置为“LV_STATE_CHECKED”状态。

List（列表）

- `LV_PART_MAIN` The list itself. Uses the typical background properties. `max_height` can be used to limit the height of the list.
- `LV_PART_SCROLLBAR` The scrollbar background, border, shadow properties and width (for its own width) and right padding for the spacing on the right.
- `LV_PART_SELECTED` Refers to the currently pressed, checked or pressed+checked option. Also uses the typical background properties.

As list does not exist when the drop-down list is closed it’s not possible to simply add styles to it. Instead the following should be done:

1. Ad an event handler to the button for `LV_EVENT_VALUE_CHANGED` (triggered when the list is opened/closed)
2. Use `lv_obj_t * list = lv_dropdown_get_list(dropdown)`
3. `if(list != NULL) {/*Add the styles to the list*/}`

Alternatively the theme can be extended with the new styles.

- LV_PART_MAIN 列表本身。使用典型的背景属性。可通过设置 max_height 限制列表的高度。
- LV_PART_SCROLLBAR 列表滚动条的背景、边框、阴影属性和宽度（对于它自己的宽度）以及右侧间距的右侧填充。
- LV_PART_SELECTED 指的是当前按下、选中或按下 + 选中的选项。也是使用典型的背景属性。

由于下拉列表关闭时列表不存在，因此无法简单地向其添加样式。我们可以通过下面的方法实现：

1. 当在列表打开/关闭时会触发 LV_EVENT_VALUE_CHANGED 事件类型
2. 在事件处理回调函数中这样就获取到列表的指针 `lv_obj_t * list = lv_dropdown_get_list(dropdown)`
3. 添加样式：`if(list != NULL) {/* 将样式添加到列表中 */}`

或者，可以使用自定义新样式扩展主题。

Usage（用法）

Overview（概述）

Set options（设置选项）

Options are passed to the drop-down list as a string with `lv_dropdown_set_options(dropdown, options)`. Options should be separated by `\n`. For example: `"First\nSecond\nThird"`. This string will be saved in the drop-down list, so it can in a local variable.

The `lv_dropdown_add_option(dropdown, "New option", pos)` function inserts a new option to `pos` index.

To save memory the options can set from a static(constant) string too with `lv_dropdown_set_static_options(dropdown, options)`. In this case the options string should be alive while the drop-down list exists and `lv_dropdown_add_option` can't be used

You can select an option manually with `lv_dropdown_set_selected(dropdown, id)`, where `id` is the index of an option.

可以通过这个函数 `lv_dropdown_set_options(dropdown, options)` 设置列表中的选项。选项之间需要使用 `\n` 分隔开来，例如：`"First\nSecond\nThird"`，该字符串将保存在下拉列表开辟的空间中，因此在设置到列表之前它可以保存在局部变量中。

`lv_dropdown_add_option(dropdown, "New option", pos)` 函数向 `pos` 索引插入一个新选项。

为了节省内存，选项也可以使用 `lv_dropdown_set_static_options(dropdown, options)` 从静态（常量）字符串中设置。在这种情况下，当下拉列表存在时，选项字符串应该处于活动状态，并且不能使用 `lv_dropdown_add_option` 插入新的选项。

可以使用 `lv_dropdown_set_selected(dropdown, id)` 手动选择一个选项，其中 `id` 是一个选项的索引，选项从 `0` 开始索引。

Get selected option（获取选择的选项）

The get the *index* of the selected option, use `lv_dropdown_get_selected(dropdown)`.

`lv_dropdown_get_selected_str(dropdown, buf, buf_size)` copies the *name* of the selected option to `buf`.

要获取所选中的选项的索引 (*index*)，可以使用 `lv_dropdown_get_selected(dropdown);`。

`lv_dropdown_get_selected_str(dropdown, buf, buf_size);` 将所选选项的 *name* 复制到 `buf`。

Direction（方向）

The list can be created on any side. The default `LV_DIR_BOTTOM` can be modified by `lv_dropdown_set_dir(dropdown, LV_DIR_LEFT/RIGHT/UP/BOTTOM)` function.

If the list would be vertically out of the screen, it will be aligned to the edge.

列表可以在任何一侧创建。默认的 `LV_DIR_BOTTOM` 可以通过 `lv_dropdown_set_dir(dropdown, LV_DIR_LEFT/RIGHT/UP/BOTTOM)` 函数进行修改。

如果列表展开会超出屏幕，他会自动进行调整。

Symbol（符号）

A symbol (typically an arrow) can be added to the drop down list with `lv_dropdown_set_symbol(dropdown, LV_SYMBOL_...)`

If the direction of the drop-down list is `LV_DIR_LEFT` the symbol will be shown on the left, otherwise on the right.

可以使用 `lv_dropdown_set_symbol(dropdown, LV_SYMBOL_...)` 将符号（通常是箭头）添加到下拉列表中

如果下拉列表的方向是 `LV_DIR_LEFT`，符号将显示在左侧，否则显示在右侧，上下侧类似。

Show selected（显示选中）

The main part can either show the selected option or a static text. If a static is set with `lv_dropdown_set_text(dropdown, "Some text")` it will be shown regardless to th selected option. If the text is `NULL` the selected option is displayed on the button.

主要部分 (LV_PART_MAIN) 可以显示所选选项或静态文本。如果使用 `lv_dropdown_set_text(dropdown, "Some text");` 设置内容，那么无论选择哪个选项，它都会只会显示你所设置的内容。如果文本为“NULL”，则所当前选选项将显示在按钮上。

Manually open/close（手动打开/关闭）

To manually open or close the drop-down list the `lv_dropdown_open/close(dropdown)` function can be used.

要手动打开或关闭下拉列表，可以使用 `lv_dropdown_open/close(dropdown);` 函数。

Events（事件）

Apart from the Generic events, the following Special events are sent by the drop-down list:

- `LV_EVENT_VALUE_CHANGED` Sent when the new option is selected or the list is opened/closed.

See the events of the *Base object* too.

Learn more about *Events*.

除了通用事件，下拉列表还可以发送以下特殊事件：

- `LV_EVENT_VALUE_CHANGED` 在选择新选项或打开/关闭列表时发送。

可以参考*Base object* 的事件。

详细了解事件。

Keys（按键）

- `LV_KEY_RIGHT/DOWN` Select the next option.
- `LV_KEY_LEFT/UP` Select the previous option.
- `LY_KEY_ENTER` Apply the selected option (Sends `LV_EVENT_VALUE_CHANGED` event and closes the drop-down list).

Learn more about *Keys*.

- `LV_KEY_RIGHT/DOWN` 选择下一个选项。
- `LV_KEY_LEFT/UP` 选择上一个选项。
- `LY_KEY_ENTER` 应用选择的选项（发送 `LV_EVENT_VALUE_CHANGED` 事件并关闭下拉列表）。

了解有关*Keys* 的更多信息。

Example

Simple Drop down list

```
#include "../../lv_examples.h"
#if LV_USE_DROPDOWN && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        char buf[32];
        lv_dropdown_get_selected_str(obj, buf, sizeof(buf));
        LV_LOG_USER("Option: %s", buf);
    }
}

void lv_example_dropdown_1(void)
{

    /*Create a normal drop down list*/
    lv_obj_t * dd = lv_dropdown_create(lv_scr_act());
    lv_dropdown_set_options(dd, "Apple\n"
                                "Banana\n"
                                "Orange\n"
                                "Cherry\n"
                                "Grape\n"
                                "Raspberry\n"
                                "Melon\n"
                                "Orange\n"
                                "Lemon\n"
                                "Nuts");

    lv_obj_align(dd, LV_ALIGN_TOP_MID, 0, 20);
    lv_obj_add_event_cb(dd, event_handler, LV_EVENT_ALL, NULL);
}

#endif
```

```
def event_handler(e):
    code = e.get_code()
    obj = e.get_target()
```

(下页继续)

(续上页)

```
    if code == lv.EVENT.VALUE_CHANGED:
        option = " "*10 # should be large enough to store the option
        obj.get_selected_str(option, len(option))
        # .strip() removes trailing spaces
        print("Option: \"%s\"" % option.strip())

# Create a normal drop down list
dd = lv.dropdown(lv.scr_act())
dd.set_options("\n".join([
    "Apple",
    "Banana",
    "Orange",
    "Cherry",
    "Grape",
    "Raspberry",
    "Melon",
    "Orange",
    "Lemon",
    "Nuts"]))

dd.align(lv.ALIGN.TOP_MID, 0, 20)
dd.add_event_cb(event_handler, lv.EVENT.ALL, None)
```

Drop down in four directions

```
#include "../../lv_examples.h"
#if LV_USE_DROPDOWN && LV_BUILD_EXAMPLES


/**
 * Create a drop down, up, left and right menus
 */
void lv_example_dropdown_2(void)
{
    static const char * opts = "Apple\n"
                               "Banana\n"
                               "Orange\n"
                               "Melon";

    lv_obj_t * dd;
    dd = lv_dropdown_create(lv_scr_act());
```

(下页继续)

(续上页)

```
    lv_dropdown_set_options_static(dd, opts);
    lv_obj_align(dd, LV_ALIGN_TOP_MID, 0, 10);

    dd = lv_dropdown_create(lv_scr_act());
    lv_dropdown_set_options_static(dd, opts);
    lv_dropdown_set_dir(dd, LV_DIR_BOTTOM);
    lv_dropdown_set_symbol(dd, LV_SYMBOL_UP);
    lv_obj_align(dd, LV_ALIGN_BOTTOM_MID, 0, -10);

    dd = lv_dropdown_create(lv_scr_act());
    lv_dropdown_set_options_static(dd, opts);
    lv_dropdown_set_dir(dd, LV_DIR_RIGHT);
    lv_dropdown_set_symbol(dd, LV_SYMBOL_RIGHT);
    lv_obj_align(dd, LV_ALIGN_LEFT_MID, 10, 0);

    dd = lv_dropdown_create(lv_scr_act());
    lv_dropdown_set_options_static(dd, opts);
    lv_dropdown_set_dir(dd, LV_DIR_LEFT);
    lv_dropdown_set_symbol(dd, LV_SYMBOL_LEFT);
    lv_obj_align(dd, LV_ALIGN_RIGHT_MID, -10, 0);
}

#endif
```

```
#
# Create a drop down, up, left and right menus
#

opts = "\n".join([
    "Apple",
    "Banana",
    "Orange",
    "Melon",
    "Grape",
    "Raspberry"])

dd = lv.dropdown(lv.scr_act())
dd.set_options_static(opts)
dd.align(lv.ALIGN.TOP_MID, 0, 10)
dd = lv.dropdown(lv.scr_act())
dd.set_options_static(opts)
dd.set_dir(lv.DIR.BOTTOM)
dd.set_symbol(lv.SYMBOL.UP)
```

(下页继续)

(续上页)

```
dd.align(lv.ALIGN.BOTTOM_MID, 0, -10)

dd = lv.dropdown(lv.scr_act())
dd.set_options_static(opts)
dd.set_dir(lv.DIR.RIGHT)
dd.set_symbol(lv.SYMBOL.RIGHT)
dd.align(lv.ALIGN.LEFT_MID, 10, 0)

dd = lv.dropdown(lv.scr_act())
dd.set_options_static(opts)
dd.set_dir(lv.DIR.LEFT)
dd.set_symbol(lv.SYMBOL.LEFT)
dd.align(lv.ALIGN.RIGHT_MID, -10, 0)
```

Menu

```
#include "../../lv_examples.h"
#if LV_USE_DROPDOWN && LV_BUILD_EXAMPLES

static void event_cb(lv_event_t * e)
{
    lv_obj_t * dropdown = lv_event_get_target(e);
    char buf[64];
    lv_dropdown_get_selected_str(dropdown, buf, sizeof(buf));
    LV_LOG_USER("'%s' is selected", buf);
}

/**
 * Create a menu from a drop-down list and show some drop-down list features and␣
↪styling
 */
void lv_example_dropdown_3(void)
{
    /*Create a drop down list*/
    lv_obj_t * dropdown = lv_dropdown_create(lv_scr_act());
    lv_obj_align(dropdown, LV_ALIGN_TOP_LEFT, 10, 10);
    lv_dropdown_set_options(dropdown, "New project\n"
                                      "New file\n"
                                      "Save\n"
                                      "Save as ...\n"
```

(下页继续)

(续上页)

```
                                    "Open project\n"
                                    "Recent projects\n"
                                    "Preferences\n"
                                    "Exit");

    /*Set a fixed text to display on the button of the drop-down list*/
    lv_dropdown_set_text(dropdown, "Menu");

    /*Use a custom image as down icon and flip it when the list is opened*/
    LV_IMG_DECLARE(img_caret_down)
    lv_dropdown_set_symbol(dropdown, &img_caret_down);
    lv_obj_set_style_transform_angle(dropdown, 1800, LV_PART_INDICATOR | LV_STATE_
↪CHECKED);

    /*In a menu we don't need to show the last clicked item*/
    lv_dropdown_set_selected_highlight(dropdown, false);

    lv_obj_add_event_cb(dropdown, event_cb, LV_EVENT_VALUE_CHANGED, NULL);
}

#endif
```

```
from imagetools import get_png_info, open_png

# Register PNG image decoder
decoder = lv.img.decoder_create()
decoder.info_cb = get_png_info
decoder.open_cb = open_png

# Create an image from the png file
try:
    with open('../../assets/img_caret_down.png','rb') as f:
        png_data = f.read()
except:
    print("Could not find img_caret_down.png")
    sys.exit()

img_caret_down_argb = lv.img_dsc_t({
  'data_size': len(png_data),
  'data': png_data
})

def event_cb(e):
```

(下页继续)

(续上页)

```
    dropdown = e.get_target()
    option = " "*64 # should be large enough to store the option
    dropdown.get_selected_str(option, len(option))
    print(option.strip() +" is selected")
#
# Create a menu from a drop-down list and show some drop-down list features and␣
↪styling
#

# Create a drop down list
dropdown = lv.dropdown(lv.scr_act())
dropdown.align(lv.ALIGN.TOP_LEFT, 10, 10)
dropdown.set_options("\n".join([
    "New project",
    "New file",
    "Open project",
    "Recent projects",
    "Preferences",
    "Exit"]))

# Set a fixed text to display on the button of the drop-down list
dropdown.set_text("Menu")

# Use a custom image as down icon and flip it when the list is opened
# LV_IMG_DECLARE(img_caret_down)
dropdown.set_symbol(img_caret_down_argb)
dropdown.set_style_transform_angle(1800, lv.PART.INDICATOR | lv.STATE.CHECKED)

# In a menu we don't need to show the last clicked item
dropdown.set_selected_highlight(False)

dropdown.add_event_cb(event_cb, lv.EVENT.VALUE_CHANGED, None)
```

API

Functions

LV_EXPORT_CONST_INT(LV_DROPDOWN_POS_LAST)

lv_obj_t ***lv_dropdown_create**(*lv_obj_t* *parent)

Create a drop-down list object

参数 **parent** -- pointer to an object, it will be the parent of the new drop-down list

返回 pointer to the created drop-down list

void **lv_dropdown_set_text**(*lv_obj_t* *obj, const char *txt)

Set text of the drop-down list's button. If set to NULL the selected option's text will be displayed on the button. If set to a specific text then that text will be shown regardless of the selected option.

参数

- **obj** -- pointer to a drop-down list object
- **txt** -- the text as a string (Only its pointer is saved)

void **lv_dropdown_set_options**(*lv_obj_t* *obj, const char *options)

Set the options in a drop-down list from a string. The options will be copied and saved in the object so the options can be destroyed after calling this function

参数

- **obj** -- pointer to drop-down list object
- **options** -- a string with '

  ' separated options. E.g. "One\nTwo\nThree"

void **lv_dropdown_set_options_static**(*lv_obj_t* *obj, const char *options)

Set the options in a drop-down list from a static string (global, static or dynamically allocated). Only the pointer of the option string will be saved.

参数

- **obj** -- pointer to drop-down list object
- **options** -- a static string with '

  ' separated options. E.g. "One\nTwo\nThree"

void **lv_dropdown_add_option**(*lv_obj_t* *obj, const char *option, uint32_t pos)

Add an options to a drop-down list from a string. Only works for non-static options.

参数

- **obj** -- pointer to drop-down list object
- **option** -- a string without '

  '. E.g. "Four"

- **pos** -- the insert position, indexed from 0, LV_DROPDOWN_POS_LAST = end of string

void **lv_dropdown_clear_options**(*lv_obj_t* *obj)

Clear all options in a drop-down list. Works with both static and dynamic options.

参数 **obj** -- pointer to drop-down list object

void **lv_dropdown_set_selected**(*lv_obj_t* *obj, uint16_t sel_opt)

Set the selected option

参数

- **obj** -- pointer to drop-down list object
- **sel_opt** -- id of the selected option (0 ... number of option - 1);

void **lv_dropdown_set_dir**(*lv_obj_t* *obj, lv_dir_t dir)

Set the direction of the a drop-down list

参数

- **obj** -- pointer to a drop-down list object
- **dir** -- LV_DIR_LEFT/RIGHT/TOP/BOTTOM

void **lv_dropdown_set_symbol**(*lv_obj_t* *obj, const void *symbol)

Set an arrow or other symbol to display when on drop-down list's button. Typically a down caret or arrow.

注解: angle and zoom transformation can be applied if the symbol is an image. E.g. when drop down is checked (opened) rotate the symbol by 180 degree

---

参数

- **obj** -- pointer to drop-down list object
- **symbol** -- a text like `LV_SYMBOL_DOWN`, an image (pointer or path) or NULL to not draw symbol icon

void **lv_dropdown_set_selected_highlight**(*lv_obj_t* *obj, bool en)

Set whether the selected option in the list should be highlighted or not

参数

- **obj** -- pointer to drop-down list object
- **en** -- true: highlight enabled; false: disabled

lv_obj_t ***lv_dropdown_get_list**(*lv_obj_t* *obj)

Get the list of a drop-down to allow styling or other modifications

参数 **obj** -- pointer to a drop-down list object

返回 pointer to the list of the drop-down

const char ***lv_dropdown_get_text**(*lv_obj_t* *obj)

Get text of the drop-down list's button.

参数 **obj** -- pointer to a drop-down list object

返回 the text as string, NULL if no text

const char ***lv_dropdown_get_options**(const *lv_obj_t* *obj)

Get the options of a drop-down list

参数 **obj** -- pointer to drop-down list object

返回

the options separated by '

'-s (E.g. "Option1\nOption2\nOption3")

uint16_t **lv_dropdown_get_selected**(const *lv_obj_t* *obj)

Get the index of the selected option

参数 **obj** -- pointer to drop-down list object

返回 index of the selected option (0 ... number of option - 1);

uint16_t **lv_dropdown_get_option_cnt**(const *lv_obj_t* *obj)

Get the total number of options

参数 **obj** -- pointer to drop-down list object

返回 the total number of options in the list

void **lv_dropdown_get_selected_str**(const *lv_obj_t* *obj, char *buf, uint32_t buf_size)

Get the current selected option as a string

参数

- **obj** -- pointer to drop-down object
- **buf** -- pointer to an array to store the string
- **buf_size** -- size of buf in bytes. 0: to ignore it.

const char ***lv_dropdown_get_symbol**(*lv_obj_t* *obj)

Get the symbol on the drop-down list. Typically a down caret or arrow.

参数 **obj** -- pointer to drop-down list object

返回 the symbol or NULL if not enabled

bool **lv_dropdown_get_selected_highlight**(*lv_obj_t* *obj)

Get whether the selected option in the list should be highlighted or not

参数 **obj** -- pointer to drop-down list object

返回 true: highlight enabled; false: disabled

lv_dir_t **lv_dropdown_get_dir**(const *lv_obj_t* *obj)

Get the direction of the drop-down list

参数 **obj** -- pointer to a drop-down list object

返回 LV_DIR_LEF/RIGHT/TOP/BOTTOM

void **lv_dropdown_open**(*lv_obj_t* *dropdown_obj)

Open the drop.down list

参数 **obj** -- pointer to drop-down list object

void **lv_dropdown_close**(*lv_obj_t* *obj)

Close (Collapse) the drop-down list

参数 **obj** -- pointer to drop-down list object

bool **lv_dropdown_is_open**(*lv_obj_t* *obj)

Tells whether the list is opened or not

参数 **obj** -- pointer to a drop-down list object

返回 true if the list os opened

Variables

const lv_obj_class_t **lv_dropdown_class**

const lv_obj_class_t **lv_dropdownlist_class**

struct **lv_dropdown_t**

Public Members

lv_obj_t **obj**

lv_obj_t ***list**

The dropped down list

const char ***text**

Text to display on the dropdown's button

const void ***symbol**

Arrow or other icon when the drop-down list is closed

char ***options**

Options in a '

' separated list

uint16_t **option_cnt**

Number of options

uint16_t **sel_opt_id**
Index of the currently selected option

uint16_t **sel_opt_id_orig**
Store the original index on focus

uint16_t **pr_opt_id**
Index of the currently pressed option

lv_dir_t **dir**
Direction in which the list should open

uint8_t **static_txt**
1: Only a pointer is saved in `options`

uint8_t **selected_highlight**
1: Make the selected option highlighted in the list

struct **lv_dropdown_list_t**

Public Members

lv_obj_t **obj**

lv_obj_t ***dropdown**

Image（图象）(lv_img)

Overview（概述）

Images are the basic object to display images from flash (as arrays) or from files. Images can display symbols (`LV_SYMBOL_...`) too.

Using the Image decoder interface custom image formats can be supported as well.

图像是显示来自闪存（作为数组）或来自文件的图像的基本对象。图像也可以显示符号（`LV_SYMBOL_...`）。

使用图像解码器接口 也可以支持自定义图像格式。

Parts and Styles（部分和风格）

- `LV_PART_MAIN` A background rectangle that uses the typical background style properties and the image itself using the image style properties.
- `LV_PART_MAIN` 使用典型背景样式属性的背景矩形和使用图像样式属性的图像本身。

Usage（用法）

Image source（图片来源）

To provide maximum flexibility, the source of the image can be:

- a variable in code (a C array with the pixels).
- a file stored externally (e.g. on an SD card).
- a text with *Symbols*.

To set the source of an image, use `lv_img_set_src(img, src)`.

To generate a pixel array from a PNG, JPG or BMP image, use the Online image converter tool and set the converted image with its pointer: `lv_img_set_src(img1, &converted_img_var);` To make the variable visible in the C file, you need to declare it with `LV_IMG_DECLARE(converted_img_var)`.

为了提供最大的灵活性，图像的来源可以是：

- 代码中的变量（带有像素的 C 数组）。
- 外部存储的文件（例如在 SD 卡上）。
- 带有*Symbols* 的文本。

要设置图像的来源，请使用 `lv_img_set_src(img, src)`。

要从 PNG、JPG 或 BMP 图像生成像素数组，请使用 在线图像转换工具 并使用其指针设置转换后的图像：`lv_img_set_src(img1, &converted_img_var );` 要使该变量在 C 文件中可见，您需要使用 `LV_IMG_DECLARE(converted_img_var)` 声明它。

To use external files, you also need to convert the image files using the online converter tool but now you should select the binary output format. You also need to use LVGL's file system module and register a driver with some functions for the basic file operation. Go to the *File system* to learn more. To set an image sourced from a file, use `lv_img_set_src(img, "S:folder1/my_img.bin")`.

You can also set a symbol similarly to *Labels*. In this case, the image will be rendered as text according to the *font* specified in the style. It enables to use of light-weight monochrome "letters" instead of real images. You can set symbol like `lv_img_set_src(img1, LV_SYMBOL_OK)`.

要使用外部文件，您还需要使用在线转换器工具转换图像文件，但现在您应该选择二进制输出格式。您还需要使用 LVGL 的文件系统模块，并为基本文件操作注册一个具有一些功能的驱动程序。转到文件系统 了解更

多信息。要设置来自文件的图像，请使用 `lv_img_set_src(img, "S:folder1/my_img.bin")`。

您还可以设置类似于标签 的符号。在这种情况下，图像将根据样式中指定的 *font* 呈现为文本。它允许使用轻量级单色“字母”而不是真实图像。你可以设置像 `lv_img_set_src(img1, LV_SYMBOL_OK)` 这样的符号。

Label as an image（标签作为图象）

Images and labels are sometimes used to convey the same thing. For example, to describe what a button does. Therefore, images and labels are somewhat interchangeable, that is the images can display texts by using `LV_SYMBOL_DUMMY` as the prefix of the text. For example, `lv_img_set_src(img, LV_SYMBOL_DUMMY "Some text")`.

图像和标签有时用于传达相同的内容。例如，描述按钮的作用。因此，图像和标签在某种程度上是可以互换的，即图像可以通过使用“LV_SYMBOL_DUMMY”作为文本的前缀来显示文本。例如，`lv_img_set_src(img, LV_SYMBOL_DUMMY "Some text")`。

Transparency（透明度）

The internal (variable) and external images support 2 transparency handling methods:

- **Chroma-keying** - Pixels with `LV_COLOR_CHROMA_KEY` (*lv_conf.h*) color will be transparent.
- **Alpha byte** - An alpha byte is added to every pixel that contains the pixel's opacity

内部（变量）和外部图像支持 2 种透明度处理方法：

- **色度键控** - 具有 `LV_COLOR_CHROMA_KEY` (*lv_conf.h*) 颜色的像素将是透明的。
- **Alpha 字节** - 向每个包含像素不透明度的像素添加一个 Alpha 字节

Palette and Alpha index（调色板和 Alpha 索引）

Besides the *True color* (RGB) color format, the following formats are supported:

- **Indexed** - Image has a palette.
- **Alpha indexed** - Only alpha values are stored.

These options can be selected in the image converter. To learn more about the color formats, read the *Images* section.

除了 *True color* (RGB) 颜色格式外，还支持以下格式：

- **索引** - 图像有调色板。
- **Alpha 索引** - 仅存储 Alpha 值。

可以在图像转换器中选择这些选项。要了解有关颜色格式的更多信息，请阅读图像 部分。

Recolor（重新着色）

A color can be mixed with every pixel of an image with a given intensity. This can be useful to show different states (checked, inactive, pressed, etc.) of an image without storing more versions of the same image. This feature can be enabled in the style by setting `img_recolor_opa` between `LV_OPA_TRANSP` (no recolor, value: 0) and `LV_OPA_COVER` (full recolor, value: 255). The default value is `LV_OPA_TRANSP` so this feature is disabled.

The color to mix is set by `img_recolor`.

颜色可以与具有给定强度的图像的每个像素混合。这对于显示图像的不同状态（选中、非活动、按下等）非常有用，而无需存储同一图像的更多版本。可以通过在 “LV_OPA_TRANSP” （无重新着色，值：0）和 “LV_OPA_COVER”（完全重新着色，值：255）之间设置 “img_recolor_opa” 在样式中启用此功能。默认值为 “LV_OPA_TRANSP”，因此禁用此功能。

要混合的颜色由 `img_recolor` 设置。

Auto-size（自动大小）

If the width or height of the image object is set to `LV_SIZE_CONTENT` the object's size will be set according to the size of the image source in the respective direction.

如果图像对象的宽度或高度设置为 `LV_SIZE_CONTENT`，则对象的大小将根据图像源在相应方向上的大小设置。

Mosaic（马赛克）

If the object's size is greater than the image size in any directions, then the image will be repeated like a mosaic. This allows creation a large image from only a very narrow source. For example, you can have a *300 x 5* image with a special gradient and set it as a wallpaper using the mosaic feature.

如果对象的大小在任何方向上都大于图像大小，则图像将像马赛克一样重复。这允许仅从非常窄的源创建大图像。例如，您可以使用带有特殊渐变的 *300 x 5* 图像，并使用马赛克功能将其设置为墙纸。

Offset（偏移）

With `lv_img_set_offset_x(img, x_ofs)` and `lv_img_set_offset_y(img, y_ofs)`, you can add some offset to the displayed image. Useful if the object size is smaller than the image source size. Using the offset parameter a Texture atlas or a "running image" effect can be created by *Animating* the x or y offset.

使用 `lv_img_set_offset_x(img, x_ofs)` 和 `lv_img_set_offset_y(img, y_ofs)`，您可以为显示的图像添加一些偏移量。如果对象大小小于图像源大小，则很有用。使用偏移参数 Texture atlas 或 “运行图像” 效果可以通过*Animating* x 或 y 偏移创建。

Transformations（转换）

Using the `lv_img_set_zoom(img, factor)` the images will be zoomed. Set `factor` to `256` or `LV_IMG_ZOOM_NONE` to disable zooming. A larger value enlarges the images (e.g. `512` double size), a smaller value shrinks it (e.g. `128` half size). Fractional scale works as well. E.g. `281` for 10% enlargement.

To rotate the image use `lv_img_set_angle(img, angle)`. Angle has 0.1 degree precision, so for 45.8° set 458.

The `transform_zoom` and `transform_angle` style properties are also used to determine the final zoom and angle.

By default, the pivot point of the rotation is the center of the image. It can be changed with `lv_img_set_pivot(img, pivot_x, pivot_y)`. `0;0` is the top left corner.

使用 `lv_img_set_zoom(img, factor)` 图像将被缩放。将 `factor` 设置为 `256` 或 `LV_IMG_ZOOM_NONE` 以禁用缩放。较大的值会放大图像（例如 “512” 双倍尺寸），较小的值会缩小图像（例如 “128” 半尺寸）。分数尺度也有效。例如。`281` 放大 10%。

要旋转图像，请使用 `lv_img_set_angle(img, angle)`。角度的精度为 0.1 度，因此对于 45.8° 设置 458。

`transform_zoom` 和 `transform_angle` 样式属性也用于确定最终的缩放和角度。

默认情况下，旋转的轴心点是图像的中心。它可以通过 `lv_img_set_pivot(img, pivot_x, pivot_y)` 改变。`0;0` 是左上角。

The quality of the transformation can be adjusted with `lv_img_set_antialias(img, true/false)`. With enabled anti-aliasing the transformations are higher quality but slower.

The transformations require the whole image to be available. Therefore indexed images (`LV_IMG_CF_INDEXED_...`), alpha only images (`LV_IMG_CF_ALPHA_...`) or images from files can not be transformed. In other words transformations work only on true color images stored as C array, or if a custom Image decoder returns the whole image.

Note that the real coordinates of image objects won't change during transformation. That is `lv_obj_get_width/height/x/y()` will return the original, non-zoomed coordinates.

转换的质量可以通过 `lv_img_set_antialias(img, true/false)` 来调整。启用抗锯齿后，转换质量更高但速度更慢。

转换需要整个图像可用。因此，索引图像（`LV_IMG_CF_INDEXED_...`）、仅 alpha 图像（`LV_IMG_CF_ALPHA_...`）或来自文件的图像无法转换。换句话说，转换仅适用于存储为 C 数组的真彩色图像，或者如果自定义 图像解码器 返回整个图像。

请注意，图像对象的真实坐标在转换过程中不会改变。即 `lv_obj_get_width/height/x/y()` 将返回原始的、未缩放的坐标。

Size mode（尺寸模式）

By default if the image is zoom or rotated the real coordinates of the image object are not changed. The larger content simply overflows the object's boundaries. It also means the layouts are not affected the by the transformations.

If you need the object size to be updated to the transformed size set `lv_img_set_size_mode(img, LV_IMG_SIZE_MODE_REAL)`. (The previous mode is the default and called `LV_IMG_SIZE_MODE_VIRTUAL`). In this case if the width/height of the object is set to `LV_SIZE_CONTENT` the object's size will be set to the zoomed and rotated size. If an explicit size is set then the overflowing content will be cropped.

默认情况下，如果图像被缩放或旋转，图像对象的真实坐标不会改变。较大的内容只是溢出对象的边界。这也意味着布局不受转换的影响。

如果您需要将对象大小更新为转换后的大小集 `lv_img_set_size_mode(img, LV_IMG_SIZE_MODE_REAL)`。（之前的模式是默认模式，称为 `LV_IMG_SIZE_MODE_VIRTUAL`）。在这种情况下，如果对象的宽度/高度设置为“LV_SIZE_CONTENT”，则对象的大小将设置为缩放和旋转后的大小。如果设置了明确的大小，那么溢出的内容将被裁剪。

Events（事件）

No special events are sent by image objects.

See the events of the *Base object* too.

Learn more about *Events*.

图像对象不发送特殊事件。

参见*Base object* 的事件。

详细了解事件。

Keys（按键）

No *Keys* are processed by the object type.

Learn more about *Keys*.

对象类型不处理 *Keys*。

了解有关*Keys* 的更多信息。

Example

Image from variable and symbol

```
#include "../../lv_examples.h"
#if LV_USE_IMG && LV_BUILD_EXAMPLES


void lv_example_img_1(void)
{
    LV_IMG_DECLARE(img_cogwheel_argb);
    lv_obj_t * img1 = lv_img_create(lv_scr_act());
    lv_img_set_src(img1, &img_cogwheel_argb);
    lv_obj_align(img1, LV_ALIGN_CENTER, 0, -20);
    lv_obj_set_size(img1, 200, 200);

    lv_obj_t * img2 = lv_img_create(lv_scr_act());
    lv_img_set_src(img2, LV_SYMBOL_OK "Accept");
    lv_obj_align_to(img2, img1, LV_ALIGN_OUT_BOTTOM_MID, 0, 20);
}

#endif
```

```
#!/opt/bin/lv_micropython -i
import usys as sys
import lvgl as lv
import display_driver
from imagetools import get_png_info, open_png

# Register PNG image decoder
decoder = lv.img.decoder_create()
decoder.info_cb = get_png_info
decoder.open_cb = open_png

# Create an image from the png file
try:
    with open('../../assets/img_cogwheel_argb.png','rb') as f:
        png_data = f.read()
except:
    print("Could not find img_cogwheel_argb.png")
    sys.exit()

img_cogwheel_argb = lv.img_dsc_t({
```

(下页继续)

(续上页)

```
  'data_size': len(png_data),
  'data': png_data
})

img1 = lv.img(lv.scr_act())
img1.set_src(img_cogwheel_argb)
img1.align(lv.ALIGN.CENTER, 0, -20)
img1.set_size(200, 200)

img2 = lv.img(lv.scr_act())
img2.set_src(lv.SYMBOL.OK + "Accept")
img2.align_to(img1, lv.ALIGN.OUT_BOTTOM_MID, 0, 20)
```

Image recoloring

```
#include "../../lv_examples.h"
#if LV_USE_IMG && LV_USE_SLIDER && LV_BUILD_EXAMPLES

static lv_obj_t * create_slider(lv_color_t color);
static void slider_event_cb(lv_event_t * e);

static lv_obj_t * red_slider, * green_slider, * blue_slider, * intense_slider;
static lv_obj_t * img1;


/**
 * Demonstrate runtime image re-coloring
 */
void lv_example_img_2(void)
{
    /*Create 4 sliders to adjust RGB color and re-color intensity*/
    red_slider = create_slider(lv_palette_main(LV_PALETTE_RED));
    green_slider = create_slider(lv_palette_main(LV_PALETTE_GREEN));
    blue_slider = create_slider(lv_palette_main(LV_PALETTE_BLUE));
    intense_slider = create_slider(lv_palette_main(LV_PALETTE_GREY));

    lv_slider_set_value(red_slider, LV_OPA_20, LV_ANIM_OFF);
    lv_slider_set_value(green_slider, LV_OPA_90, LV_ANIM_OFF);
    lv_slider_set_value(blue_slider, LV_OPA_60, LV_ANIM_OFF);
    lv_slider_set_value(intense_slider, LV_OPA_50, LV_ANIM_OFF);

    lv_obj_align(red_slider, LV_ALIGN_LEFT_MID, 25, 0);
```

(下页继续)

(续上页)

```
    lv_obj_align_to(green_slider, red_slider, LV_ALIGN_OUT_RIGHT_MID, 25, 0);
    lv_obj_align_to(blue_slider, green_slider, LV_ALIGN_OUT_RIGHT_MID, 25, 0);
    lv_obj_align_to(intense_slider, blue_slider, LV_ALIGN_OUT_RIGHT_MID, 25, 0);

    /*Now create the actual image*/
    LV_IMG_DECLARE(img_cogwheel_argb)
    img1 = lv_img_create(lv_scr_act());
    lv_img_set_src(img1, &img_cogwheel_argb);
    lv_obj_align(img1, LV_ALIGN_RIGHT_MID, -20, 0);

    lv_event_send(intense_slider, LV_EVENT_VALUE_CHANGED, NULL);
}

static void slider_event_cb(lv_event_t * e)
{
    LV_UNUSED(e);

    /*Recolor the image based on the sliders' values*/
    lv_color_t color  = lv_color_make(lv_slider_get_value(red_slider), lv_slider_get_
↪value(green_slider), lv_slider_get_value(blue_slider));
    lv_opa_t intense = lv_slider_get_value(intense_slider);
    lv_obj_set_style_img_recolor_opa(img1, intense, 0);
    lv_obj_set_style_img_recolor(img1, color, 0);
}

static lv_obj_t * create_slider(lv_color_t color)
{
    lv_obj_t * slider = lv_slider_create(lv_scr_act());
    lv_slider_set_range(slider, 0, 255);
    lv_obj_set_size(slider, 10, 200);
    lv_obj_set_style_bg_color(slider, color, LV_PART_KNOB);
    lv_obj_set_style_bg_color(slider, lv_color_darken(color, LV_OPA_40), LV_PART_
↪INDICATOR);
    lv_obj_add_event_cb(slider, slider_event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    return slider;
}

#endif
```

```
#!/opt/bin/lv_micropython -i
import usys as sys
import lvgl as lv
import display_driver
```

(下页继续)

(续上页)

```
from imagetools import get_png_info, open_png

# Register PNG image decoder
decoder = lv.img.decoder_create()
decoder.info_cb = get_png_info
decoder.open_cb = open_png

# Create an image from the png file
try:
    with open('../../assets/img_cogwheel_argb.png','rb') as f:
        png_data = f.read()
except:
    print("Could not find img_cogwheel_argb.png")
    sys.exit()

img_cogwheel_argb = lv.img_dsc_t({
  'data_size': len(png_data),
  'data': png_data
})

def create_slider(color):
    slider = lv.slider(lv.scr_act())
    slider.set_range(0, 255)
    slider.set_size(10, 200)
    slider.set_style_bg_color(color, lv.PART.KNOB)
    slider.set_style_bg_color(color.color_darken(lv.OPA._40), lv.PART.INDICATOR)
    slider.add_event_cb(slider_event_cb, lv.EVENT.VALUE_CHANGED, None)
    return slider

def slider_event_cb(e):
    # Recolor the image based on the sliders' values
    color  = lv.color_make(red_slider.get_value(), green_slider.get_value(), blue_
↪slider.get_value())
    intense = intense_slider.get_value()
    img1.set_style_img_recolor_opa(intense, 0)
    img1.set_style_img_recolor(color, 0)

#
# Demonstrate runtime image re-coloring
#
# Create 4 sliders to adjust RGB color and re-color intensity
red_slider = create_slider(lv.palette_main(lv.PALETTE.RED))
green_slider = create_slider(lv.palette_main(lv.PALETTE.GREEN))
```

(下页继续)

(续上页)

```
blue_slider = create_slider(lv.palette_main(lv.PALETTE.BLUE))
intense_slider = create_slider(lv.palette_main(lv.PALETTE.GREY))

red_slider.set_value(lv.OPA._20, lv.ANIM.OFF)
green_slider.set_value(lv.OPA._90, lv.ANIM.OFF)
blue_slider.set_value(lv.OPA._60, lv.ANIM.OFF)
intense_slider.set_value(lv.OPA._50, lv.ANIM.OFF)

red_slider.align(lv.ALIGN.LEFT_MID, 25, 0)
green_slider.align_to(red_slider, lv.ALIGN.OUT_RIGHT_MID, 25, 0)
blue_slider.align_to(green_slider, lv.ALIGN.OUT_RIGHT_MID, 25, 0)
intense_slider.align_to(blue_slider, lv.ALIGN.OUT_RIGHT_MID, 25, 0)

# Now create the actual image
img1 = lv.img(lv.scr_act())
img1.set_src(img_cogwheel_argb)
img1.align(lv.ALIGN.RIGHT_MID, -20, 0)

lv.event_send(intense_slider, lv.EVENT.VALUE_CHANGED, None)
```

Rotate and zoom

```
#include "../../lv_examples.h"
#if LV_USE_IMG && LV_BUILD_EXAMPLES

static void set_angle(void * img, int32_t v)
{
    lv_img_set_angle(img, v);
}

static void set_zoom(void * img, int32_t v)
{
    lv_img_set_zoom(img, v);
}


/**
```

(下页继续)

(续上页)

```
 * Show transformations (zoom and rotation) using a pivot point.
 */
void lv_example_img_3(void)
{
    LV_IMG_DECLARE(img_cogwheel_argb);

    /*Now create the actual image*/
    lv_obj_t * img = lv_img_create(lv_scr_act());
    lv_img_set_src(img, &img_cogwheel_argb);
    lv_obj_align(img, LV_ALIGN_CENTER, 50, 50);
    lv_img_set_pivot(img, 0, 0);    /*Rotate around the top left corner*/

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_var(&a, img);
    lv_anim_set_exec_cb(&a, set_angle);
    lv_anim_set_values(&a, 0, 3600);
    lv_anim_set_time(&a, 5000);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
    lv_anim_start(&a);

    lv_anim_set_exec_cb(&a, set_zoom);
    lv_anim_set_values(&a, 128, 256);
    lv_anim_set_playback_time(&a, 3000);
    lv_anim_start(&a);
}

#endif
```

```
#!/opt/bin/lv_micropython -i
import usys as sys
import lvgl as lv
import display_driver
from imagetools import get_png_info, open_png

# Register PNG image decoder
decoder = lv.img.decoder_create()
decoder.info_cb = get_png_info
decoder.open_cb = open_png

# Create an image from the png file
try:
    with open('../../assets/img_cogwheel_argb.png','rb') as f:
```

(下页继续)

(续上页)

```
        png_data = f.read()
except:
    print("Could not find img_cogwheel_argb.png")
    sys.exit()

img_cogwheel_argb = lv.img_dsc_t({
  'data_size': len(png_data),
  'data': png_data
})

def set_angle(img, v):
    img.set_angle(v)

def set_zoom(img, v):
    img.set_zoom(v)


#
# Show transformations (zoom and rotation) using a pivot point.
#

# Now create the actual image
img = lv.img(lv.scr_act())
img.set_src(img_cogwheel_argb)
img.align(lv.ALIGN.CENTER, 50, 50)
img.set_pivot(0, 0)               # Rotate around the top left corner

a1 = lv.anim_t()
a1.init()
a1.set_var(img)
a1.set_custom_exec_cb(lambda a,val: set_angle(img,val))
a1.set_values(0, 3600)
a1.set_time(5000)
a1.set_repeat_count(lv.ANIM_REPEAT.INFINITE)
lv.anim_t.start(a1)

a2 = lv.anim_t()
a2.init()
a2.set_var(img)
a2.set_custom_exec_cb(lambda a,val: set_zoom(img,val))
a2.set_values(128, 256)
a2.set_time(5000)
a2.set_playback_time(3000)
```

(下页继续)

(续上页)

```
a2.set_repeat_count(lv.ANIM_REPEAT.INFINITE)
lv.anim_t.start(a2)
```

Image offset and styling

```
#include "../../lv_examples.h"
#if LV_USE_IMG && LV_BUILD_EXAMPLES

static void ofs_y_anim(void * img, int32_t v)
{
    lv_img_set_offset_y(img, v);
}

/**
 * Image styling and offset
 */
void lv_example_img_4(void)
{
    LV_IMG_DECLARE(img_skew_strip);

    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_bg_color(&style, lv_palette_main(LV_PALETTE_YELLOW));
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_img_recolor_opa(&style, LV_OPA_COVER);
    lv_style_set_img_recolor(&style, lv_color_black());

    lv_obj_t * img = lv_img_create(lv_scr_act());
    lv_obj_add_style(img, &style, 0);
    lv_img_set_src(img, &img_skew_strip);
    lv_obj_set_size(img, 150, 100);
    lv_obj_center(img);

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_var(&a, img);
    lv_anim_set_exec_cb(&a, ofs_y_anim);
    lv_anim_set_values(&a, 0, 100);
    lv_anim_set_time(&a, 3000);
    lv_anim_set_playback_time(&a, 500);
```

(下页继续)

(续上页)

```
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
    lv_anim_start(&a);

}

#endif
```

```
from imagetools import get_png_info, open_png

def ofs_y_anim(img, v):
    img.set_offset_y(v)
    # print(img,v)

# Register PNG image decoder
decoder = lv.img.decoder_create()
decoder.info_cb = get_png_info
decoder.open_cb = open_png

# Create an image from the png file
try:
    with open('../../assets/img_skew_strip.png','rb') as f:
        png_data = f.read()
except:
    print("Could not find img_skew_strip.png")
    sys.exit()

img_skew_strip = lv.img_dsc_t({
  'data_size': len(png_data),
  'data': png_data
})

#
# Image styling and offset
#

style = lv.style_t()
style.init()
style.set_bg_color(lv.palette_main(lv.PALETTE.YELLOW))
style.set_bg_opa(lv.OPA.COVER)
style.set_img_recolor_opa(lv.OPA.COVER)
style.set_img_recolor(lv.color_black())

img = lv.img(lv.scr_act())
```

(下页继续)

(续上页)

```
img.add_style(style, 0)
img.set_src(img_skew_strip)
img.set_size(150, 100)
img.center()

a = lv.anim_t()
a.init()
a.set_var(img)
a.set_values(0, 100)
a.set_time(3000)
a.set_playback_time(500)
a.set_repeat_count(lv.ANIM_REPEAT.INFINITE)
a.set_custom_exec_cb(lambda a,val: ofs_y_anim(img,val))
lv.anim_t.start(a)
```

API

Typedefs

typedef uint8_t **lv_img_size_mode_t**

Enums

enum **[anonymous]**

Image size mode, when image size and object size is different

Values:

enumerator **LV_IMG_SIZE_MODE_VIRTUAL**

Zoom doesn't affect the coordinates of the object, however if zoomed in the image is drawn out of the its coordinates. The layout's won't change on zoom

enumerator **LV_IMG_SIZE_MODE_REAL**

If the object size is set to SIZE_CONTENT, then object size equals zoomed image size. It causes layout recalculation. If the object size is set explicitly, the image will be cropped when zoomed in.

Functions

lv_obj_t ***lv_img_create**(*lv_obj_t* *parent)

Create an image object

参数 **parent** -- pointer to an object, it will be the parent of the new image

返回 pointer to the created image

void **lv_img_set_src**(*lv_obj_t* *obj, const void *src)

Set the image data to display on the object

参数

- **obj** -- pointer to an image object
- **src_img** -- 1) pointer to an *lv_img_dsc_t* descriptor (converted by LVGL's image converter) (e.g. &my_img) or 2) path to an image file (e.g. "S:/dir/img.bin")or 3) a SYMBOL (e.g. LV_SYMBOL_OK)

void **lv_img_set_offset_x**(*lv_obj_t* *obj, lv_coord_t x)

Set an offset for the source of an image so the image will be displayed from the new origin.

参数

- **obj** -- pointer to an image
- **x** -- the new offset along x axis.

void **lv_img_set_offset_y**(*lv_obj_t* *obj, lv_coord_t y)

Set an offset for the source of an image. so the image will be displayed from the new origin.

参数

- **obj** -- pointer to an image
- **y** -- the new offset along y axis.

void **lv_img_set_angle**(*lv_obj_t* *obj, int16_t angle)

Set the rotation angle of the image. The image will be rotated around the set pivot set by *lv_img_set_pivot()*

参数

- **obj** -- pointer to an image object
- **angle** -- rotation angle in degree with 0.1 degree resolution (0..3600: clock wise)

void **lv_img_set_pivot**(*lv_obj_t* *obj, lv_coord_t x, lv_coord_t y)

Set the rotation center of the image. The image will be rotated around this point

参数

- **obj** -- pointer to an image object
- **x** -- rotation center x of the image

- **y** -- rotation center y of the image

void **lv_img_set_zoom**(*lv_obj_t* *obj, uint16_t zoom)

void **lv_img_set_antialias**(*lv_obj_t* *obj, bool antialias)

Enable/disable anti-aliasing for the transformations (rotate, zoom) or not. The quality is better with anti-aliasing looks better but slower.

参数

- **obj** -- pointer to an image object
- **antialias** -- true: anti-aliased; false: not anti-aliased

void **lv_img_set_size_mode**(*lv_obj_t* *obj, *lv_img_size_mode_t* mode)

Set the image object size mode.

参数

- **obj** -- pointer to an image object
- **mode** -- the new size mode.

const void ***lv_img_get_src**(*lv_obj_t* *obj)

Get the source of the image

参数 **obj** -- pointer to an image object

返回 the image source (symbol, file name or ::lv-img_dsc_t for C arrays)

lv_coord_t **lv_img_get_offset_x**(*lv_obj_t* *obj)

Get the offset's x attribute of the image object.

参数 **img** -- pointer to an image

返回 offset X value.

lv_coord_t **lv_img_get_offset_y**(*lv_obj_t* *obj)

Get the offset's y attribute of the image object.

参数 **obj** -- pointer to an image

返回 offset Y value.

uint16_t **lv_img_get_angle**(*lv_obj_t* *obj)

Get the rotation angle of the image.

参数 **obj** -- pointer to an image object

返回 rotation angle in 0.1 degrees (0..3600)

void **lv_img_get_pivot**(*lv_obj_t* *obj, lv_point_t *pivot)

Get the pivot (rotation center) of the image.

参数

- **img** -- pointer to an image object
- **pivot** -- store the rotation center here

uint16_t **lv_img_get_zoom**(*lv_obj_t* *obj)

Get the zoom factor of the image.

参数 **obj** -- pointer to an image object

返回 zoom factor (256: no zoom)

bool **lv_img_get_antialias**(*lv_obj_t* *obj)

Get whether the transformations (rotate, zoom) are anti-aliased or not

参数 **obj** -- pointer to an image object

返回 true: anti-aliased; false: not anti-aliased

lv_img_size_mode_t **lv_img_get_size_mode**(*lv_obj_t* *obj)

Get the size mode of the image

参数 **obj** -- pointer to an image object

返回 element of lv_img_size_mode_t

Variables

const lv_obj_class_t **lv_img_class**

struct **lv_img_t**

#include <lv_img.h> Data of image

Public Members

lv_obj_t **obj**

const void ***src**

lv_point_t **offset**

lv_coord_t **w**

lv_coord_t **h**

uint16_t **angle**

lv_point_t **pivot**

uint16_t **zoom**

uint8_t **src_type**

uint8_t **cf**

uint8_t **antialias**

uint8_t **obj_size_mode**

Label（标签）(lv_label)

Overview（概述）

A label is the basic object type that is used to display text.

标签是用来显示文本的基本对象类型。

Parts and Styles（部分和风格）

- `LV_PART_MAIN` Uses all the typical background properties and the text properties. The padding values can be used to add space between the text and the background.
- `LV_PART_SCROLLBAR` The scrollbar that is shown when the text is larger than the widget's size.
- `LV_PART_SELECTED` Tells the style of the *selected text*. Only `text_color` and `bg_color` style properties can be used.
- `LV_PART_MAIN` 使用所有典型的背景属性和文本属性。填充值可用于在文本和背景之间添加空间。
- `LV_PART_SCROLLBAR` 当文本大于组件的大小时显示的滚动条。
- `LV_PART_SELECTED` 告诉所选文本 的样式。只能使用 `text_color` 和 `bg_color` 样式属性。

Usage（用法）

Set text（设置文本）

You can set the text on a label at runtime with `lv_label_set_text(label, "New text")`. This will allocate a buffer dynamically, and the provided string will be copied into that buffer. Therefore, you don't need to keep the text you pass to `lv_label_set_text` in scope after that function returns.

With `lv_label_set_text_fmt(label, "Value: %d", 15)` printf formatting can be used to set the text.

您可以在运行时使用 `lv_label_set_text(label, "New text")` 设置标签上的文本。这将动态分配一个缓冲区，并且提供的字符串将被复制到该缓冲区中。因此，在该函数返回后，您不需要将传递给 `lv_label_set_text` 的文本保留在作用域中。

使用 `lv_label_set_text_fmt(label, "Value: %d", 15)` printf 格式可用于设置文本。

Labels are able to show text from a static character buffer. To do so, use `lv_label_set_text_static(label, "Text")`. In this case, the text is not stored in the dynamic memory and the given buffer is used directly instead. This means that the array can't be a local variable which goes out of scope when the function exits. Constant strings are safe to use with `lv_label_set_text_static` (except when used with `LV_LABEL_LONG_DOT`, as it modifies the buffer in-place), as they are stored in ROM memory, which is always accessible.

标签能够显示来自静态字符缓冲区的文本。为此，请使用 `lv_label_set_text_static(label, "Text")`。在这种情况下，文本不存储在动态内存中，而是直接使用给定的缓冲区。这意味着数组不能是在函数退出时超出范围的局部变量。常量字符串可以安全地与 `lv_label_set_text_static` 一起使用（除非与 `LV_LABEL_LONG_DOT` 一起使用，因为它会就地修改缓冲区），因为它们存储在 ROM 内存中，始终可以访问。

Newline（新行）

Newline characters are handled automatically by the label object. You can use `\n` to make a line break. For example: `"line1\nline2\n\nline4"`

换行符由标签对象自动处理。您可以使用 `\n` 来换行。例如：`"line1\nline2\n\nline4"`

Long modes（长模式）

By default, the width and height of the label is set to `LV_SIZE_CONTENT`. Therefore the size of the label is automatically expanded to the text size. Otherwise, if the width or height are explicitly set (useing e.g.`lv_obj_set_width` or a layout), the lines wider than the label's width can be manipulated according to several long mode policies. Similary, the policies can be applied if the height of the text is greater than the height of the label.

- `LV_LABEL_LONG_WRAP` Wrap too long lines. If the height is `LV_SIZE_CONTENT` the label's height will be expanded, otherwise the text will be clipped. (Default)
- `LV_LABEL_LONG_DOT` Replaces the last 3 characters from bottom right corner of the label with dots (`.`)
- `LV_LABEL_LONG_SCROLL` If the text is wider than the label scroll it horizontally back and forth. If it's higher, scroll vertically. Only one direction is scrolled and horizontal scrolling has higher precedence.
- `LV_LABEL_LONG_SCROLL_CIRCULAR` If the text is wider than the label scroll it horizontally continously. If it's higher, scroll vertically. Only one direction is scrolled and horizontal scrolling has higher precedence.
- `LV_LABEL_LONG_CLIP` Simply clip the parts of the text outside of the label.

默认情况下，标签的宽度和高度设置为 `LV_SIZE_CONTENT`。因此标签的大小会自动扩展到文本大小。否则，如果显式设置宽度或高度（使用例如 `lv_obj_set_width` 或布局），则可以根据几种长模式策略来操作比标签宽度更宽的行。类似地，如果文本的高度大于标签的高度，则可以应用策略。

- `LV_LABEL_LONG_WRAP` 换行太长。如果高度为 `LV_SIZE_CONTENT`，标签的高度将被扩展，否则文本将被剪裁。（默认）
- `LV_LABEL_LONG_DOT` 将标签右下角的最后 3 个字符替换为点 (`.`)

- LV_LABEL_LONG_SCROLL 如果文本比标签宽，则水平来回滚动它。如果它更高，请垂直滚动。只滚动一个方向，水平滚动的优先级更高。
- LV_LABEL_LONG_SCROLL_CIRCULAR 如果文本比标签宽，则水平滚动它。如果它更高，请垂直滚动。只滚动一个方向，水平滚动的优先级更高。
- LV_LABEL_LONG_CLIP 只需剪掉标签外的文本部分。

You can specify the long mode with `lv_label_set_long_mode(label, LV_LABEL_LONG_...)`

Note that `LV_LABEL_LONG_DOT` manipulates the text buffer in-place in order to add/remove the dots.When `lv_label_set_text` or `lv_label_set_array_text` are used, a separate buffer is allocated and this implementation detail is unnoticed. This is not the case with `lv_label_set_text_static`. The buffer you pass to `lv_label_set_text_static` must be writable if you plan to use `LV_LABEL_LONG_DOT`.

您可以使用 `lv_label_set_long_mode(label, LV_LABEL_LONG_...)` 指定长模式

请注意，`LV_LABEL_LONG_DOT` 就地操作文本缓冲区以添加/删除点。当使用 `lv_label_set_text` 或 `lv_label_set_array_text` 时，会分配一个单独的缓冲区，并且不会注意到此实现细节。`lv_label_set_text_static` 不是这种情况。如果你打算使用 `LV_LABEL_LONG_DOT`，你传递给 `lv_label_set_text_static` 的缓冲区必须是可写的。

Text recolor（文本重新着色）

In the text, you can use commands to recolor parts of the text. For example: `"Write a #ff0000 red# word"`. This feature can be enabled individually for each label by `lv_label_set_recolor()` function.

在文本中，您可以使用命令对文本的某些部分重新着色。例如：“写一个 `#ff0000 red#` 字”。可以通过 `lv_label_set_recolor()` 函数为每个标签单独启用此功能。

Text selection（文本选择）

If enabled by `LV_LABEL_TEXT_SELECTION` part of the text can be selected. It's similar when on PC a you use your mouse to select a text. The whole mechanism (click and select the text as you drag your finger/mouse) is implemented in *Text area* and the Label widget only allows manual text selection with `lv_label_get_text_selection_start(label, start_char_index)` and `lv_label_get_text_selection_start(label, end_char_index)`.

如果通过 `LV_LABEL_TEXT_SELECTION` 启用，可以选择部分文本。这与在 PC 上使用鼠标选择文本时类似。整个机制（在拖动手指/鼠标时单击并选择文本）在文本区域 中实现，而标签组件仅允许手动选择文本 `lv_label_get_text_selection_start(label, start_char_index)` 和 `lv_label_get_text_selection_start(label, end_char_index)`。

Very long texts（非常长的文本）

LVGL can efficiently handle very long (e.g. > 40k characters) labels by saving some extra data (~12 bytes) to speed up drawing. To enable this feature, set `LV_LABEL_LONG_TXT_HINT 1` in `lv_conf.h`.

LVGL 可以通过保存一些额外的数据（~12 字节）来有效地处理很长（例如 > 40k 个字符）的标签以加快绘图速度。要启用此功能，请在“lv_conf.h”中设置“LV_LABEL_LONG_TXT_HINT 1”。

Symbols（符号）

The labels can display symbols alongside letters (or on their own). Read the *Font* section to learn more about the symbols.

标签可以在字母旁边显示符号（或单独显示）。阅读字体 部分以了解有关符号的更多信息。

Events（事件）

No special events are sent by the Label.

See the events of the *Base object* too.

Learn more about *Events*.

标签不发送特殊事件。

参见*Base object* 的事件。

详细了解事件。

Keys（按键）

No *Keys* are processed by the object type.

Learn more about *Keys*.

对象类型不处理 *Keys*。

了解有关按键 的更多信息。

Example

Line wrap, recoloring and scrolling

```
#include "../../lv_examples.h"
#if LV_USE_LABEL && LV_BUILD_EXAMPLES

/**
```

(下页继续)

(续上页)

```
 * Show line wrap, re-color, line align and text scrolling.
 */
void lv_example_label_1(void)
{
    lv_obj_t * label1 = lv_label_create(lv_scr_act());
    lv_label_set_long_mode(label1, LV_LABEL_LONG_WRAP);     /*Break the long lines*/
    lv_label_set_recolor(label1, true);                      /*Enable re-coloring by␣
↪commands in the text*/
    lv_label_set_text(label1, "#0000ff Re-color# #ff00ff words# #ff0000 of a# label,␣
↪align the lines to the center "
                              "and wrap long text automatically.");
    lv_obj_set_width(label1, 150);  /*Set smaller width to make the lines wrap*/
    lv_obj_set_style_text_align(label1, LV_TEXT_ALIGN_CENTER, 0);
    lv_obj_align(label1, LV_ALIGN_CENTER, 0, -40);

    lv_obj_t * label2 = lv_label_create(lv_scr_act());
    lv_label_set_long_mode(label2, LV_LABEL_LONG_SCROLL_CIRCULAR);     /*Circular␣
↪scroll*/
    lv_obj_set_width(label2, 150);
    lv_label_set_text(label2, "It is a circularly scrolling text. ");
    lv_obj_align(label2, LV_ALIGN_CENTER, 0, 40);
}

#endif
```

```
#
# Show line wrap, re-color, line align and text scrolling.
#
label1 = lv.label(lv.scr_act())
label1.set_long_mode(lv.label.LONG.WRAP)      # Break the long lines*/
label1.set_recolor(True)                      # Enable re-coloring by commands in the␣
↪text
label1.set_text("#0000ff Re-color# #ff00ff words# #ff0000 of a# label, align the␣
↪lines to the center"
                              "and  wrap long text automatically.")
label1.set_width(150)                         # Set smaller width to make the lines␣
↪wrap
label1.set_style_text_align(lv.ALIGN.CENTER, 0)
label1.align(lv.ALIGN.CENTER, 0, -40)


label2 = lv.label(lv.scr_act())
label2.set_long_mode(lv.label.LONG.SCROLL_CIRCULAR) # Circular scroll
```

(下页继续)

(续上页)

```
label2.set_width(150)
label2.set_text("It is a circularly scrolling text. ")
label2.align(lv.ALIGN.CENTER, 0, 40)
```

Text shadow

```
#include "../../lv_examples.h"
#if LV_USE_LABEL && LV_BUILD_EXAMPLES

/**
 * Create a fake text shadow
 */
void lv_example_label_2(void)
{
    /*Create a style for the shadow*/
    static lv_style_t style_shadow;
    lv_style_init(&style_shadow);
    lv_style_set_text_opa(&style_shadow, LV_OPA_30);
    lv_style_set_text_color(&style_shadow, lv_color_black());

    /*Create a label for the shadow first (it's in the background)*/
    lv_obj_t * shadow_label = lv_label_create(lv_scr_act());
    lv_obj_add_style(shadow_label, &style_shadow, 0);

    /*Create the main label*/
    lv_obj_t * main_label = lv_label_create(lv_scr_act());
    lv_label_set_text(main_label, "A simple method to create\n"
                                  "shadows on a text.\n"
                                  "It even works with\n\n"
                                  "newlines     and spaces.");

    /*Set the same text for the shadow label*/
    lv_label_set_text(shadow_label, lv_label_get_text(main_label));

    /*Position the main label*/
    lv_obj_align(main_label, LV_ALIGN_CENTER, 0, 0);

    /*Shift the second label down and to the right by 2 pixel*/
    lv_obj_align_to(shadow_label, main_label, LV_ALIGN_TOP_LEFT, 2, 2);
}
```

(下页继续)

(续上页)

```
#endif
```

```
#
# Create a fake text shadow
#

# Create a style for the shadow
style_shadow = lv.style_t()
style_shadow.init()
style_shadow.set_text_opa(lv.OPA._30)
style_shadow.set_text_color(lv.color_black())

# Create a label for the shadow first (it's in the background)
shadow_label = lv.label(lv.scr_act())
shadow_label.add_style(style_shadow, 0)

# Create the main label
main_label = lv.label(lv.scr_act())
main_label.set_text("A simple method to create\n"
                    "shadows on a text.\n"
                    "It even works with\n\n"
                    "newlines     and spaces.")

# Set the same text for the shadow label
shadow_label.set_text(lv.label.get_text(main_label))

# Position the main label
main_label.align(lv.ALIGN.CENTER, 0, 0)

# Shift the second label down and to the right by 2 pixel
shadow_label.align_to(main_label, lv.ALIGN.TOP_LEFT, 2, 2)
```

Show LTR, RTL and Chinese texts

```
#include "../../lv_examples.h"
#if LV_USE_LABEL && LV_BUILD_EXAMPLES && LV_FONT_DEJAVU_16_PERSIAN_HEBREW && LV_FONT_
↪SIMSUN_16_CJK && LV_USE_BIDI

/**
 * Show mixed LTR, RTL and Chinese label
 */
```

(下页继续)

(续上页)

```
void lv_example_label_3(void)
{
    lv_obj_t * ltr_label = lv_label_create(lv_scr_act());
    lv_label_set_text(ltr_label, "In modern terminology, a microcontroller is similar␣
↪to a system on a chip (SoC).");
    lv_obj_set_style_text_font(ltr_label, &lv_font_montserrat_16, 0);
    lv_obj_set_width(ltr_label, 310);
    lv_obj_align(ltr_label, LV_ALIGN_TOP_LEFT, 5, 5);

    lv_obj_t * rtl_label = lv_label_create(lv_scr_act());
    lv_label_set_text(rtl_label, "□□□□, □□ □□□□ □□□□ □□□□□ □□□□□ □□□□□□ :□□□□□□□) CPU␣
↪- Central Processing Unit).");
    lv_obj_set_style_base_dir(rtl_label, LV_BASE_DIR_RTL, 0);
    lv_obj_set_style_text_font(rtl_label, &lv_font_dejavu_16_persian_hebrew, 0);
    lv_obj_set_width(rtl_label, 310);
    lv_obj_align(rtl_label, LV_ALIGN_LEFT_MID, 5, 0);

    lv_obj_t * cz_label = lv_label_create(lv_scr_act());
    lv_label_set_text(cz_label, "嵌入式系统 (Embedded System), \n是一种嵌入机械或电气系统内部、
具有专一功能和实时计算性能的计算机系统。");
    lv_obj_set_style_text_font(cz_label, &lv_font_simsun_16_cjk, 0);
    lv_obj_set_width(cz_label, 310);
    lv_obj_align(cz_label, LV_ALIGN_BOTTOM_LEFT, 5, -5);
}

#endif
```

```
import fs_driver
#
# Show mixed LTR, RTL and Chinese label
#

ltr_label = lv.label(lv.scr_act())
ltr_label.set_text("In modern terminology, a microcontroller is similar to a system␣
↪on a chip (SoC).")
# ltr_label.set_style_text_font(ltr_label, &lv_font_montserrat_16, 0);

fs_drv = lv.fs_drv_t()
fs_driver.fs_register(fs_drv, 'S')

try:
    ltr_label.set_style_text_font(ltr_label, lv.font_montserrat_16, 0)
except:
```

(下页继续)

(续上页)

```
    font_montserrat_16 = lv.font_load("S:../../assets/font/montserrat-16.fnt")
    ltr_label.set_style_text_font(font_montserrat_16, 0)

ltr_label.set_width(310)
ltr_label.align(lv.ALIGN.TOP_LEFT, 5, 5)

rtl_label = lv.label(lv.scr_act())
rtl_label.set_text("□□□□, □□ □□□□ □□□□ □□□□□ □□□□□ □□□□□□ :□□□□□□□) CPU - Central␣
↪Processing Unit).")
rtl_label.set_style_base_dir(lv.BASE_DIR.RTL, 0)
rtl_label.set_style_text_font(lv.font_dejavu_16_persian_hebrew, 0)
rtl_label.set_width(310)
rtl_label.align(lv.ALIGN.LEFT_MID, 5, 0)

font_simsun_16_cjk = lv.font_load("S:../../assets/font/lv_font_simsun_16_cjk.fnt")

cz_label = lv.label(lv.scr_act())
cz_label.set_style_text_font(font_simsun_16_cjk, 0)
cz_label.set_text("嵌入式系统 (Embedded System), \n是一种嵌入机械或电气系统内部、具有专一功能和实
时计算性能的计算机系统。")
cz_label.set_width(310)
cz_label.align(lv.ALIGN.BOTTOM_LEFT, 5, -5)
```

Draw label with gradient color

```
#include "../../lv_examples.h"
#if LV_USE_LABEL && LV_USE_CANVAS && LV_BUILD_EXAMPLES && LV_DRAW_COMPLEX

#define MASK_WIDTH 100
#define MASK_HEIGHT 45

static void add_mask_event_cb(lv_event_t * e)
{
    static lv_draw_mask_map_param_t m;
    static int16_t mask_id;

    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target(e);
    lv_opa_t * mask_map = lv_event_get_user_data(e);
    if(code == LV_EVENT_COVER_CHECK) {
        lv_event_set_cover_res(e, LV_COVER_RES_MASKED);
```

(下页继续)

(续上页)

```
    }
    else if(code == LV_EVENT_DRAW_MAIN_BEGIN) {
        lv_draw_mask_map_init(&m, &obj->coords, mask_map);
        mask_id = lv_draw_mask_add(&m, NULL);

    }
    else if(code == LV_EVENT_DRAW_MAIN_END) {
        lv_draw_mask_free_param(&m);
        lv_draw_mask_remove_id(mask_id);
    }
}

/**
 * Draw label with gradient color
 */
void lv_example_label_4(void)
{
    /* Create the mask of a text by drawing it to a canvas*/
    static lv_opa_t mask_map[MASK_WIDTH * MASK_HEIGHT];

    /*Create a "8 bit alpha" canvas and clear it*/
    lv_obj_t * canvas = lv_canvas_create(lv_scr_act());
    lv_canvas_set_buffer(canvas, mask_map, MASK_WIDTH, MASK_HEIGHT, LV_IMG_CF_ALPHA_
↪8BIT);
    lv_canvas_fill_bg(canvas, lv_color_black(), LV_OPA_TRANSP);

    /*Draw a label to the canvas. The result "image" will be used as mask*/
    lv_draw_label_dsc_t label_dsc;
    lv_draw_label_dsc_init(&label_dsc);
    label_dsc.color = lv_color_white();
    label_dsc.align = LV_TEXT_ALIGN_CENTER;
    lv_canvas_draw_text(canvas, 5, 5, MASK_WIDTH, &label_dsc, "Text with gradient");

    /*The mask is reads the canvas is not required anymore*/
    lv_obj_del(canvas);

    /* Create an object from where the text will be masked out.
     * Now it's a rectangle with a gradient but it could be an image too*/
    lv_obj_t * grad = lv_obj_create(lv_scr_act());
    lv_obj_set_size(grad, MASK_WIDTH, MASK_HEIGHT);
    lv_obj_center(grad);
    lv_obj_set_style_bg_color(grad, lv_color_hex(0xff0000), 0);
    lv_obj_set_style_bg_grad_color(grad, lv_color_hex(0x0000ff), 0);
```

(下页继续)

(续上页)

```
    lv_obj_set_style_bg_grad_dir(grad, LV_GRAD_DIR_HOR, 0);
    lv_obj_add_event_cb(grad, add_mask_event_cb, LV_EVENT_ALL, mask_map);
}

#endif
```

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs/100ask_lvgl_
↪docs/examples/widgets/label/lv_example_label_4.py
```

API

Typedefs

typedef uint8_t **lv_label_long_mode_t**

Enums

enum **[anonymous]**

Long mode behaviors. Used in 'lv_label_ext_t'

Values:

enumerator **LV_LABEL_LONG_WRAP**

Keep the object width, wrap the too long lines and expand the object height

enumerator **LV_LABEL_LONG_DOT**

Keep the size and write dots at the end if the text is too long

enumerator **LV_LABEL_LONG_SCROLL**

Keep the size and roll the text back and forth

enumerator **LV_LABEL_LONG_SCROLL_CIRCULAR**

Keep the size and roll the text circularly

enumerator **LV_LABEL_LONG_CLIP**

Keep the size and clip the text out of it

Functions

LV_EXPORT_CONST_INT(LV_LABEL_DOT_NUM)

LV_EXPORT_CONST_INT(LV_LABEL_POS_LAST)

LV_EXPORT_CONST_INT(LV_LABEL_TEXT_SELECTION_OFF)

lv_obj_t ***lv_label_create**(*lv_obj_t* *parent)

Create a label object

参数 **parent** -- pointer to an object, it will be the parent of the new label.

返回 pointer to the created button

void **lv_label_set_text**(*lv_obj_t* *obj, const char *text)

Set a new text for a label. Memory will be allocated to store the text by the label.

参数

- **obj** -- pointer to a label object
- **text** -- '\0' terminated character string. NULL to refresh with the current text.

**void lv_label_set_text_fmt (lv_obj_t *obj, const char *fmt,...
) LV_FORMAT_ATTRIBUTE(2**

void void lv_label_set_text_static (lv_obj_t *obj, const char *text)

Set a static text. It will not be saved by the label so the 'text' variable has to be 'alive' while the label exists.

参数

- **obj** -- pointer to a label object
- **text** -- pointer to a text. NULL to refresh with the current text.

void **lv_label_set_long_mode**(*lv_obj_t* *obj, *lv_label_long_mode_t* long_mode)

Set the behavior of the label with longer text then the object size

参数

- **obj** -- pointer to a label object
- **long_mode** -- the new mode from 'lv_label_long_mode' enum. In LV_LONG_WRAP/DOT/SCROLL/SCROLL_CIRC the size of the label should be set AFTER this function

void **lv_label_set_recolor**(*lv_obj_t* *obj, bool en)

void **lv_label_set_text_sel_start**(*lv_obj_t* *obj, uint32_t index)

Set where text selection should start

参数

- **obj** -- pointer to a label object
- **index** -- character index from where selection should start. LV_LABEL_TEXT_SELECTION_OFF for no selection

void **lv_label_set_text_sel_end**(*lv_obj_t* *obj, uint32_t index)

Set where text selection should end

参数

- **obj** -- pointer to a label object
- **index** -- character index where selection should end. LV_LABEL_TEXT_SELECTION_OFF for no selection

char ***lv_label_get_text**(const *lv_obj_t* *obj)

Get the text of a label

参数 **obj** -- pointer to a label object

返回 the text of the label

lv_label_long_mode_t **lv_label_get_long_mode**(const *lv_obj_t* *obj)

Get the long mode of a label

参数 **obj** -- pointer to a label object

返回 the current long mode

bool **lv_label_get_recolor**(const *lv_obj_t* *obj)

Get the recoloring attribute

参数 **obj** -- pointer to a label object

返回 true: recoloring is enabled, false: disable

void **lv_label_get_letter_pos**(const *lv_obj_t* *obj, uint32_t char_id, lv_point_t *pos)

Get the relative x and y coordinates of a letter

参数

- **obj** -- pointer to a label object
- **index** -- index of the character [0 ... text length - 1]. Expressed in character index, not byte index (different in UTF-8)
- **pos** -- store the result here (E.g. index = 0 gives 0;0 coordinates if the text if aligned to the left)

uint32_t **lv_label_get_letter_on**(const *lv_obj_t* *obj, lv_point_t *pos_in)

Get the index of letter on a relative point of a label.

参数

- **obj** -- pointer to label object
- **pos** -- pointer to point with coordinates on a the label

返回 The index of the letter on the 'pos_p' point (E.g. on 0;0 is the 0. letter if aligned to the left) Expressed in character index and not byte index (different in UTF-8)

bool **lv_label_is_char_under_pos**(const *lv_obj_t* *obj, lv_point_t *pos)

Check if a character is drawn under a point.

参数

- **obj** -- pointer to a label object
- **pos** -- Point to check for character under

返回 whether a character is drawn under the point

uint32_t **lv_label_get_text_selection_start**(const *lv_obj_t* *obj)

Get the selection start index.

参数 **obj** -- pointer to a label object.

返回 selection start index. `LV_LABEL_TEXT_SELECTION_OFF` if nothing is selected.

uint32_t **lv_label_get_text_selection_end**(const *lv_obj_t* *obj)

Get the selection end index.

参数 **obj** -- pointer to a label object.

返回 selection end index. `LV_LABEL_TXT_SEL_OFF` if nothing is selected.

void **lv_label_ins_text**(*lv_obj_t* *obj, uint32_t pos, const char *txt)

Insert a text to a label. The label text can not be static.

参数

- **obj** -- pointer to a label object
- **pos** -- character index to insert. Expressed in character index and not byte index. 0: before first char. LV_LABEL_POS_LAST: after last char.
- **txt** -- pointer to the text to insert

void **lv_label_cut_text**(*lv_obj_t* *obj, uint32_t pos, uint32_t cnt)

Delete characters from a label. The label text can not be static.

参数

- **obj** -- pointer to a label object

- **pos** -- character index from where to cut. Expressed in character index and not byte index. 0: start in from of the first character
- **cnt** -- number of characters to cut

Variables

const lv_obj_class_t **lv_label_class**

struct **lv_label_t**

Public Members

lv_obj_t **obj**

char ***text**

char ***tmp_ptr**

char **tmp**[LV_LABEL_DOT_NUM + 1]

union *lv_label_t*::[anonymous] **dot**

uint32_t **dot_end**

lv_draw_label_hint_t **hint**

uint32_t **sel_start**

uint32_t **sel_end**

lv_point_t **offset**

lv_label_long_mode_t **long_mode**

uint8_t **static_txt**

uint8_t **recolor**

uint8_t **expand**

uint8_t **dot_tmp_alloc**

Line（线条）(lv_line)

Overview（概述）

The Line object is capable of drawing straight lines between a set of points.

线条 (Line) 组件能在给出的一组点之间绘制出相连的直线。

Parts and Styles（零件和样式）

- `LV_PART_MAIN` uses all the typical background properties and line style properties.
- `LV_PART_MAIN` 使用所有典型的背景属性和线条样式属性。

Usage（用法）

Set points(设置点)

The points have to be stored in an `lv_point_t` array and passed to the object by the `lv_line_set_points(lines, point_array, point_cnt)` function.

点必须存储在 `lv_point_t` 类型的数组中，并通过 `lv_line_set_points(lines, point_array, point_cnt)` 函数将数组传递给 line 对象。

Auto-size（自动调整大小）

By default the Line's width and height are set to `LV_SIZE_CONTENT`. This means it will automatically set its size to fit all the points. If the size is set explicitly, parts on the line may not be visible.

默认情况下，线条的宽度和高度被设置为 `LV_SIZE_CONTENT` 。也就是它会根据给出的点自动调整其大小来适应所有点的分布。如果你设置了宽、高，那么线条上的有些部分可能会不可见。

Invert y(反转 y 轴)

By default, the *y == 0* point is in the top of the object. It might be conter-intuitive in some cases so the y coordinates can be inverted with `lv_line_set_y_invert(line, true)`. In this case, *y == 0* will be the bottom of the object. *y invert* is disabled by default.

默认情况下，*y == 0* 点是在 line 对象的顶部最左侧。这在某些情况下可能是不直观的 (LCD 坐标系)，这时候可以使用 `lv_line_set_y_invert(line, true)` 函数来反转 y 坐标。在这种情况下，*y == 0* 将是物体的底部最左侧 (直角坐标系)。

默认不反转 y 轴。

Events（事件）

Only the Generic events are sent by the object type.

See the events of the *Base object* too.

Learn more about *Events*.

线条对象仅发送 通用事件。

参见*Base object* 的事件。

详细了解事件。

Keys（按键）

No *Keys* are processed by the object type.

Learn more about *Keys*.

线条对象不响应处理按键 (以及触摸)。

了解有关键 的更多信息。

Example

Simple Line

```
#include "../../lv_examples.h"
#if LV_USE_LINE && LV_BUILD_EXAMPLES

void lv_example_line_1(void)
{
    /*Create an array for the points of the line*/
    static lv_point_t line_points[] = { {5, 5}, {70, 70}, {120, 10}, {180, 60}, {240,␣
↪10} };

    /*Create style*/
    static lv_style_t style_line;
    lv_style_init(&style_line);
    lv_style_set_line_width(&style_line, 8);
    lv_style_set_line_color(&style_line, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_line_rounded(&style_line, true);

    /*Create a line and apply the new style*/
    lv_obj_t * line1;
```

(下页继续)

(续上页)

```
    line1 = lv_line_create(lv_scr_act());
    lv_line_set_points(line1, line_points, 5);     /*Set the points*/
    lv_obj_add_style(line1, &style_line, 0);
    lv_obj_center(line1);
}

#endif
```

```
# Create an array for the points of the line
line_points = [ {"x":5, "y":5},
                {"x":70, "y":70},
                {"x":120, "y":10},
                {"x":180, "y":60},
                {"x":240, "y":10}]

# Create style
style_line = lv.style_t()
style_line.init()
style_line.set_line_width(8)
style_line.set_line_color(lv.palette_main(lv.PALETTE.BLUE))
style_line.set_line_rounded(True)

# Create a line and apply the new style
line1 = lv.line(lv.scr_act())
line1.set_points(line_points, 5)     # Set the points
line1.add_style(style_line, 0)
line1.center()
```

API

Functions

lv_obj_t ***lv_line_create**(*lv_obj_t* *parent)

Create a line object

参数 **parent** -- pointer to an object, it will be the parent of the new line

返回 pointer to the created line

void **lv_line_set_points**(*lv_obj_t* *obj, const lv_point_t points[], uint16_t point_num)

Set an array of points. The line object will connect these points.

参数

- **obj** -- pointer to a line object
- **points** -- an array of points. Only the address is saved, so the array needs to be alive while the line exists
- **point_num** -- number of points in 'point_a'

void **lv_line_set_y_invert**(*lv_obj_t* *obj, bool en)

Enable (or disable) the y coordinate inversion. If enabled then y will be subtracted from the height of the object, therefore the y = 0 coordinate will be on the bottom.

参数

- **obj** -- pointer to a line object
- **en** -- true: enable the y inversion, false:disable the y inversion

bool **lv_line_get_y_invert**(const *lv_obj_t* *obj)

Get the y inversion attribute

参数 **obj** -- pointer to a line object

返回 true: y inversion is enabled, false: disabled

Variables

const lv_obj_class_t **lv_line_class**

struct **lv_line_t**

Public Members

lv_obj_t **obj**

const lv_point_t ***point_array**

Pointer to an array with the points of the line

uint16_t **point_num**

Number of points in 'point_array'

uint8_t **y_inv**

1: y == 0 will be on the bottom

Roller（滚轮）(lv_roller)

Overview（概述）

Roller allows you to simply select one option from a list by scrolling.

滚轮类似下拉列表，和下拉列表不同的是，滚轮有直接可见的几个选项，我们可以通过滚动滚轮中的列表来浏览并选择选项。

Parts and Styles（零件和样式）

- `LV_PART_MAIN` The background of the roller uses all the typical background properties and text style properties. `style_text_line_space` adjusts the space between the options. When the Roller is scrolled and doesn't stop exactly on an option it will scroll to the nearest valid option automatically in `anim_time` milliseconds as specified in the style.
- `LV_PART_SELECTED` The selected option in the middle. Besides the typical background properties it uses the text style properties to change the appearance of the text in the selected area.
- `LV_PART_MAIN` 滚轮的背景使用了所有典型的背景属性和文本样式属性。可以通过 `style_text_line_space` 调整选项之间的间隔。当滚轮滚动并且没有完全停在一个选项上时，它将按照样式中指定的 `anim_time` 值 (毫秒) 自动滚动到前进方向最近的有效选项。
- `LV_PART_SELECTED` 中间选中的选项。除了典型的背景属性之外，它还使用文本样式属性来更改所选区域中文本的外观。

Usage（用法）

Set options（设置选项）

Options are passed to the Roller as a string with `lv_roller_set_options(roller, options, LV_ROLLER_MODE_NORMAL/INFINITE)`. The options should be separated by `\n`. For example: `"First\nSecond\nThird"`.

`LV_ROLLER_MODE_INFINITE` makes the roller circular.

You can select an option manually with `lv_roller_set_selected(roller, id, LV_ANIM_ON/OFF)`, where *id* is the index of an option.

可以通过这个函数 `lv_roller_set_options(roller, options, LV_ROLLER_MODE_NORMAL/INFINITE)` 设置 Roller 中的选项。选项之间要用 `\n` 分隔。例如：`"First\nSecond\nThird"`。

- 参数 `LV_ROLLER_MODE_NORMAL` 是设置为正常模式（滚轮在选项结束时结束）
- 参数 `LV_ROLLER_MODE_INFINITE` 是设置为无限模式（滚轮可以永远滚动）

可以使用 `lv_roller_set_selected(roller, id, LV_ANIM_ON/OFF)` 手动选中一个选项，其中 `id` 是选项的索引，选项从 0 开始索引。

Get selected option（获取选中的选项）

The get the *index* of the currently selected option use `lv_roller_get_selected(roller)`.

`lv_roller_get_selected_str(roller, buf, buf_size)` will copy the name of the selected option to `buf`.

- 要获取所选中的选项的索引 (index)，可以使用函数 `lv_roller_get_selected(roller)`
- 函数 `lv_roller_get_selected_str(roller, buf, buf_size)` 会将所选选项的内容复制到 `buf`

Visible rows（可见行）

The number of visible rows can be adjusted with `lv_roller_set_visible_row_count(roller, num)`.

This function calculates the height with the current style. If the font, line space, border width, etc of the roller changes this function needs to be called again.

可见行数可以通过 `lv_roller_set_visible_row_count(roller, num)` 进行调整。

此函数会根据滚轮当前样式的高度（字体、行距、边框宽度等）进行调整。**如果滚轮的字体、行距、边框宽度等发生变化，则需要再次调用此函数以重新进行调整**。

Events（事件）

- `LV_EVENT_VALUE_CHANGED` Sent when a new option is selected.

See the events of the *Base object* too.

Learn more about *Events*.

- 在选中选项是会触发 `LV_EVENT_VALUE_CHANGED`（按照逻辑来说只要你滚动了滚轮中的选项都会触发此事件类型）。

可以参考*Base object* 的事件。

详细了解事件。

Keys（按键）

- LV_KEY_RIGHT/DOWN Select the next option
- LV_KEY_LEFT/UP Select the previous option
- LY_KEY_ENTER Apply the selected option (Send LV_EVENT_VALUE_CHANGED event)
- LV_KEY_RIGHT/DOWN 选择下一个选项
- LV_KEY_LEFT/UP 选择上一个选项
- LY_KEY_ENTER 应用选择的选项（发送 LV_EVENT_VALUE_CHANGED 事件）

使用编码器控制时要注意有两种模式：导航模式和编辑模式

Example

Simple Roller

```
#include "../../lv_examples.h"
#if LV_USE_ROLLER && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        char buf[32];
        lv_roller_get_selected_str(obj, buf, sizeof(buf));
        LV_LOG_USER("Selected month: %s\n", buf);
    }
}

/**
 * An infinite roller with the name of the months
 */
void lv_example_roller_1(void)
{
    lv_obj_t *roller1 = lv_roller_create(lv_scr_act());
    lv_roller_set_options(roller1,
                          "January\n"
                          "February\n"
                          "March\n"
                          "April\n"
                          "May\n"
```

(下页继续)

(续上页)

```
                          "June\n"
                          "July\n"
                          "August\n"
                          "September\n"
                          "October\n"
                          "November\n"
                          "December",
                          LV_ROLLER_MODE_INFINITE);

    lv_roller_set_visible_row_count(roller1, 4);
    lv_obj_center(roller1);
    lv_obj_add_event_cb(roller1, event_handler, LV_EVENT_ALL, NULL);
}

#endif
```

```
def event_handler(e):
    code = e.get_code()
    obj = e.get_target()
    if code == lv.EVENT.VALUE_CHANGED:
        option = " "*10
        obj.get_selected_str(option, len(option))
        print("Selected month: " + option.strip())

#
# An infinite roller with the name of the months
#

roller1 = lv.roller(lv.scr_act())
roller1.set_options("\n".join([
    "January",
    "February",
    "March",
    "April",
    "May",
    "June",
    "July",
    "August",
    "September",
    "October",
    "November",
    "December"]),lv.roller.MODE.INFINITE)
```

(下页继续)

(续上页)

```
roller1.set_visible_row_count(4)
roller1.center()
roller1.add_event_cb(event_handler, lv.EVENT.ALL, None)
```

Styling the roller

```
#include "../../lv_examples.h"
#if LV_USE_ROLLER && LV_FONT_MONTSERRAT_22 && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        char buf[32];
        lv_roller_get_selected_str(obj, buf, sizeof(buf));
        LV_LOG_USER("Selected value: %s", buf);
    }
}

/**
 * Roller with various alignments and larger text in the selected area
 */
void lv_example_roller_2(void)
{
    /*A style to make the selected option larger*/
    static lv_style_t style_sel;
    lv_style_init(&style_sel);
    lv_style_set_text_font(&style_sel, &lv_font_montserrat_22);

    const char * opts = "1\n2\n3\n4\n5\n6\n7\n8\n9\n10";
    lv_obj_t *roller;

    /*A roller on the left with left aligned text, and custom width*/
    roller = lv_roller_create(lv_scr_act());
    lv_roller_set_options(roller, opts, LV_ROLLER_MODE_NORMAL);
    lv_roller_set_visible_row_count(roller, 2);
    lv_obj_set_width(roller, 100);
    lv_obj_add_style(roller, &style_sel, LV_PART_SELECTED);
    lv_obj_set_style_text_align(roller, LV_TEXT_ALIGN_LEFT, 0);
    lv_obj_align(roller, LV_ALIGN_LEFT_MID, 10, 0);
```

(下页继续)

(续上页)

```
    lv_obj_add_event_cb(roller, event_handler, LV_EVENT_ALL, NULL);
    lv_roller_set_selected(roller, 2, LV_ANIM_OFF);

    /*A roller on the middle with center aligned text, and auto (default) width*/
    roller = lv_roller_create(lv_scr_act());
    lv_roller_set_options(roller, opts, LV_ROLLER_MODE_NORMAL);
    lv_roller_set_visible_row_count(roller, 3);
    lv_obj_add_style(roller, &style_sel, LV_PART_SELECTED);
    lv_obj_align(roller, LV_ALIGN_CENTER, 0, 0);
    lv_obj_add_event_cb(roller, event_handler, LV_EVENT_ALL, NULL);
    lv_roller_set_selected(roller, 5, LV_ANIM_OFF);

    /*A roller on the right with right aligned text, and custom width*/
    roller = lv_roller_create(lv_scr_act());
    lv_roller_set_options(roller, opts, LV_ROLLER_MODE_NORMAL);
    lv_roller_set_visible_row_count(roller, 4);
    lv_obj_set_width(roller, 80);
    lv_obj_add_style(roller, &style_sel, LV_PART_SELECTED);
    lv_obj_set_style_text_align(roller, LV_TEXT_ALIGN_RIGHT, 0);
    lv_obj_align(roller, LV_ALIGN_RIGHT_MID, -10, 0);
    lv_obj_add_event_cb(roller, event_handler, LV_EVENT_ALL, NULL);
    lv_roller_set_selected(roller, 8, LV_ANIM_OFF);
}

#endif
```

```
import fs_driver


def event_handler(e):
    code = e.get_code()
    obj = e.get_target()
    if code == lv.EVENT.VALUE_CHANGED:
        option = " "*10
        obj.get_selected_str(option, len(option))
        print("Selected value: %s\n" + option.strip())

#
# Roller with various alignments and larger text in the selected area
#

# A style to make the selected option larger
style_sel = lv.style_t()
```

(下页继续)

(续上页)

```
style_sel.init()

try:
    style_sel.set_text_font(lv.font_montserrat_22)
except:
    fs_drv = lv.fs_drv_t()
    fs_driver.fs_register(fs_drv, 'S')
    print("montserrat-22 not enabled in lv_conf.h, dynamically loading the font")
    font_montserrat_22 = lv.font_load("S:" + "../../assets/font/montserrat-22.fnt")
    style_sel.set_text_font(font_montserrat_22)

opts = "\n".join(["1","2","3","4","5","6","7","8","9","10"])

# A roller on the left with left aligned text, and custom width
roller = lv.roller(lv.scr_act())
roller.set_options(opts, lv.roller.MODE.NORMAL)
roller.set_visible_row_count(2)
roller.set_width(100)
roller.add_style(style_sel, lv.PART.SELECTED)
roller.set_style_text_align(lv.TEXT_ALIGN.LEFT, 0)
roller.align(lv.ALIGN.LEFT_MID, 10, 0)
roller.add_event_cb(event_handler, lv.EVENT.ALL, None)
roller.set_selected(2, lv.ANIM.OFF)

# A roller in the middle with center aligned text, and auto (default) width
roller = lv.roller(lv.scr_act())
roller.set_options(opts, lv.roller.MODE.NORMAL)
roller.set_visible_row_count(3)
roller.add_style(style_sel, lv.PART.SELECTED)
roller.align(lv.ALIGN.CENTER, 0, 0)
roller.add_event_cb(event_handler, lv.EVENT.ALL, None)
roller.set_selected(5, lv.ANIM.OFF)

# A roller on the right with right aligned text, and custom width
roller = lv.roller(lv.scr_act())
roller.set_options(opts, lv.roller.MODE.NORMAL)
roller.set_visible_row_count(4)
roller.set_width(80)
roller.add_style(style_sel, lv.PART.SELECTED)
roller.set_style_text_align(lv.TEXT_ALIGN.RIGHT, 0)
roller.align(lv.ALIGN.RIGHT_MID, -10, 0)
roller.add_event_cb(event_handler, lv.EVENT.ALL, None)
roller.set_selected(8, lv.ANIM.OFF)
```

add fade mask to roller

```
#include "../../lv_examples.h"
#if LV_USE_ROLLER && LV_DRAW_COMPLEX && LV_BUILD_EXAMPLES

static void mask_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target(e);

    static int16_t mask_top_id = -1;
    static int16_t mask_bottom_id = -1;

    if (code == LV_EVENT_COVER_CHECK) {
        lv_event_set_cover_res(e, LV_COVER_RES_MASKED);

    } else if (code == LV_EVENT_DRAW_MAIN_BEGIN) {
        /* add mask */
        const lv_font_t * font = lv_obj_get_style_text_font(obj, LV_PART_MAIN);
        lv_coord_t line_space = lv_obj_get_style_text_line_space(obj, LV_PART_MAIN);
        lv_coord_t font_h = lv_font_get_line_height(font);

        lv_area_t roller_coords;
        lv_obj_get_coords(obj, &roller_coords);

        lv_area_t rect_area;
        rect_area.x1 = roller_coords.x1;
        rect_area.x2 = roller_coords.x2;
        rect_area.y1 = roller_coords.y1;
        rect_area.y2 = roller_coords.y1 + (lv_obj_get_height(obj) - font_h - line_
↪space) / 2;

        lv_draw_mask_fade_param_t * fade_mask_top = lv_mem_buf_get(sizeof(lv_draw_
↪mask_fade_param_t));
        lv_draw_mask_fade_init(fade_mask_top, &rect_area, LV_OPA_TRANSP, rect_area.y1,
↪ LV_OPA_COVER, rect_area.y2);
        mask_top_id = lv_draw_mask_add(fade_mask_top, NULL);

        rect_area.y1 = rect_area.y2 + font_h + line_space - 1;
        rect_area.y2 = roller_coords.y2;

        lv_draw_mask_fade_param_t * fade_mask_bottom =lv_mem_buf_get(sizeof(lv_draw_
↪mask_fade_param_t));
        lv_draw_mask_fade_init(fade_mask_bottom, &rect_area, LV_OPA_COVER, rect_area.
↪y1, LV_OPA_TRANSP, rect_area.y2);
```

(下页继续)

(续上页)

```
        mask_bottom_id = lv_draw_mask_add(fade_mask_bottom, NULL);

    } else if (code == LV_EVENT_DRAW_POST_END) {
        lv_draw_mask_fade_param_t * fade_mask_top = lv_draw_mask_remove_id(mask_top_
↪id);
        lv_draw_mask_fade_param_t * fade_mask_bottom = lv_draw_mask_remove_id(mask_
↪bottom_id);
        lv_draw_mask_free_param(fade_mask_top);
        lv_draw_mask_free_param(fade_mask_bottom);
        lv_mem_buf_release(fade_mask_top);
        lv_mem_buf_release(fade_mask_bottom);
        mask_top_id = -1;
        mask_bottom_id = -1;
    }
}

/**
 * Add a fade mask to roller.
 */
void lv_example_roller_3(void)
{
    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_bg_color(&style, lv_color_black());
    lv_style_set_text_color(&style, lv_color_white());
    lv_style_set_border_width(&style, 0);
    lv_style_set_pad_all(&style, 0);
    lv_obj_add_style(lv_scr_act(), &style, 0);

    lv_obj_t *roller1 = lv_roller_create(lv_scr_act());
    lv_obj_add_style(roller1, &style, 0);
    lv_obj_set_style_bg_opa(roller1, LV_OPA_TRANSP, LV_PART_SELECTED);

#if LV_FONT_MONTSERRAT_22
    lv_obj_set_style_text_font(roller1, &lv_font_montserrat_22, LV_PART_SELECTED);
#endif

    lv_roller_set_options(roller1,
                          "January\n"
                          "February\n"
                          "March\n"
                          "April\n"
                          "May\n"
```

(下页继续)

(续上页)

```
                          "June\n"
                          "July\n"
                          "August\n"
                          "September\n"
                          "October\n"
                          "November\n"
                          "December",
                          LV_ROLLER_MODE_NORMAL);

    lv_obj_center(roller1);
    lv_roller_set_visible_row_count(roller1, 3);
    lv_obj_add_event_cb(roller1, mask_event_cb, LV_EVENT_ALL, NULL);
}

#endif
```

```
import fs_driver
import sys

class Lv_Roller_3():

    def __init__(self):
        self.mask_top_id = -1
        self.mask_bottom_id = -1

        #
        # Add a fade mask to roller.
        #
        style = lv.style_t()
        style.init()
        style.set_bg_color(lv.color_black())
        style.set_text_color(lv.color_white())

        lv.scr_act().add_style(style, 0)

        roller1 = lv.roller(lv.scr_act())
        roller1.add_style(style, 0)
        roller1.set_style_border_width(0, 0)
        roller1.set_style_pad_all(0, 0)
        roller1.set_style_bg_opa(lv.OPA.TRANSP, lv.PART.SELECTED)

        #if LV_FONT_MONTSERRAT_22
        #    lv_obj_set_style_text_font(roller1, &lv_font_montserrat_22, LV_PART_
↪SELECTED);
```

(下页继续)

(续上页)

```
        #endif
        try:
            roller1.set_style_text_font(lv.font_montserrat_22,lv.PART.SELECTED)
        except:
            fs_drv = lv.fs_drv_t()
            fs_driver.fs_register(fs_drv, 'S')
            print("montserrat-22 not enabled in lv_conf.h, dynamically loading the␣
↪font")
            font_montserrat_22 = lv.font_load("S:" + "../../assets/font/montserrat-22.
↪fnt")
            roller1.set_style_text_font(font_montserrat_22,lv.PART.SELECTED)

        roller1.set_options("\n".join([
            "January",
            "February",
            "March",
            "April",
            "May",
            "June",
            "July",
            "August",
            "September",
            "October",
            "November",
            "December"]),lv.roller.MODE.NORMAL)

        roller1.center()
        roller1.set_visible_row_count(3)
        roller1.add_event_cb(self.mask_event_cb, lv.EVENT.ALL, None)

    def mask_event_cb(self,e):

        code = e.get_code()
        obj = e.get_target()

        if code == lv.EVENT.COVER_CHECK:
            e.set_cover_res(lv.COVER_RES.MASKED)

        elif code == lv.EVENT.DRAW_MAIN_BEGIN:
            # add mask
            font = obj.get_style_text_font(lv.PART.MAIN)
            line_space = obj.get_style_text_line_space(lv.PART.MAIN)
            font_h = font.get_line_height()
```

(下页继续)

(续上页)

```
            roller_coords = lv.area_t()
            obj.get_coords(roller_coords)

            rect_area = lv.area_t()
            rect_area.x1 = roller_coords.x1
            rect_area.x2 = roller_coords.x2
            rect_area.y1 = roller_coords.y1
            rect_area.y2 = roller_coords.y1 + (obj.get_height() - font_h - line_
↪space) // 2

            fade_mask_top = lv.draw_mask_fade_param_t()
            fade_mask_top.init(rect_area, lv.OPA.TRANSP, rect_area.y1, lv.OPA.COVER,␣
↪rect_area.y2)
            self.mask_top_id = lv.draw_mask_add(fade_mask_top,None)

            rect_area.y1 = rect_area.y2 + font_h + line_space - 1
            rect_area.y2 = roller_coords.y2

            fade_mask_bottom = lv.draw_mask_fade_param_t()
            fade_mask_bottom.init(rect_area, lv.OPA.COVER, rect_area.y1, lv.OPA.
↪TRANSP, rect_area.y2)
            self.mask_bottom_id = lv.draw_mask_add(fade_mask_bottom, None)

        elif code == lv.EVENT.DRAW_POST_END:
            fade_mask_top = lv.draw_mask_remove_id(self.mask_top_id)
            fade_mask_bottom = lv.draw_mask_remove_id(self.mask_bottom_id)
            # Remove the masks
            lv.draw_mask_remove_id(self.mask_top_id)
            lv.draw_mask_remove_id(self.mask_bottom_id)
            self.mask_top_id = -1
            self.mask_bottom_id = -1

roller3 = Lv_Roller_3()
```

API

Typedefs

typedef uint8_t **lv_roller_mode_t**

Enums

enum **[anonymous]**
Roller mode.

Values:

enumerator **LV_ROLLER_MODE_NORMAL**
Normal mode (roller ends at the end of the options).

enumerator **LV_ROLLER_MODE_INFINITE**
Infinite mode (roller can be scrolled forever).

Functions

lv_obj_t ***lv_roller_create**(*lv_obj_t* *parent)
Create a roller object

参数 **parent** -- pointer to an object, it will be the parent of the new roller.

返回 pointer to the created roller

void **lv_roller_set_options**(*lv_obj_t* *obj, const char *options, *lv_roller_mode_t* mode)
Set the options on a roller

参数

- **obj** -- pointer to roller object
- **options** -- a string with '
  ' separated options. E.g. "One\nTwo\nThree"
- **mode** -- LV_ROLLER_MODE_NORMAL or LV_ROLLER_MODE_INFINITE

void **lv_roller_set_selected**(*lv_obj_t* *obj, uint16_t sel_opt, *lv_anim_enable_t* anim)
Set the selected option

参数

- **obj** -- pointer to a roller object
- **sel_opt** -- index of the selected option (0 ... number of option - 1);
- **anim_en** -- LV_ANIM_ON: set with animation; LV_ANOM_OFF set immediately

void **lv_roller_set_visible_row_count**(*lv_obj_t* *obj, uint8_t row_cnt)
Set the height to show the given number of rows (options)

参数

- **obj** -- pointer to a roller object

- **row_cnt** -- number of desired visible rows

uint16_t **lv_roller_get_selected**(const *lv_obj_t* *obj)

Get the index of the selected option

参数 **obj** -- pointer to a roller object

返回 index of the selected option (0 ... number of option - 1);

void **lv_roller_get_selected_str**(const *lv_obj_t* *obj, char *buf, uint32_t buf_size)

Get the current selected option as a string.

参数

- **obj** -- pointer to ddlist object
- **buf** -- pointer to an array to store the string
- **buf_size** -- size of `buf` in bytes. 0: to ignore it.

const char ***lv_roller_get_options**(const *lv_obj_t* *obj)

Get the options of a roller

参数 **obj** -- pointer to roller object

返回

the options separated by '

'-s (E.g. "Option1\nOption2\nOption3")

uint16_t **lv_roller_get_option_cnt**(const *lv_obj_t* *obj)

Get the total number of options

参数 **obj** -- pointer to a roller object

返回 the total number of options

Variables

const lv_obj_class_t **lv_roller_class**

struct **lv_roller_t**

Public Members

lv_obj_t **obj**

uint16_t **option_cnt**
Number of options

uint16_t **sel_opt_id**
Index of the current option

uint16_t **sel_opt_id_ori**
Store the original index on focus

lv_roller_mode_t **mode**

uint32_t **moved**

Slider（滑动条）(lv_slider)

Overview（概述）

The Slider object looks like a *Bar* supplemented with a knob. The knob can be dragged to set a value. Just like Bar, Slider can be vertical or horizontal.

滑动条对象看起来像是在*进度条 (bar)* 增加了一个可以调节的旋钮，使用时可以通过拖动旋钮来设置一个值。就像进度条 (bar) 一样，Slider 可以是垂直的或水平的 (当设置进度条的宽度小于其高度，就可以创建出垂直摆放的滑动条)。

Parts and Styles（部分和样式）

- `LV_PART_MAIN` The background of the slider. Uses all the typical background style properties. `padding` makes the indicator smaller in the respective direction.
- `LV_PART_INDICATOR` The indicator that shows the current state of the slider. Also uses all the typical background style properties.
- `LV_PART_KNOB` A rectangle (or circle) drawn at the current value. Also uses all the typical background properties to describe the knob(s). By default the knob is square (with a optional corner radius) with side length equal to the smaller side of the slider. The knob can be made larger with the `padding` values. Padding values can be asymmetric too.
- `LV_PART_MAIN` 滑动条的背景。使用所有典型的背景样式属性。设置 `padding` 样式会使指标在相应方向上变小。
- `LV_PART_INDICATOR` 显示滑动条当前状态的指示器。也是使用所有典型的背景样式属性。

- LV_PART_KNOB 旋钮 (可以是原形或矩形)。也是使用所有典型的背景属性。默认情况下，旋钮是方形的（带有可选的圆角半径），边长等于滑动条的较小边。可以通过设置 padding 样式调整旋钮的大小。填充值也可以是不对称的。

Usage（用法）

Value and range（值和范围）

To set an initial value use `lv_slider_set_value(slider, new_value, LV_ANIM_ON/OFF)`. The animation time is set by the styles' `anim_time` property.

To specify the range (min, max values), `lv_slider_set_range(slider, min, max)` can be used.

要设置滑动条的初始值，请使用 `lv_slider_set_value(slider, new_value, LV_ANIM_ON/OFF)`。动画时间由样式的 `anim_time` 属性设置。

要指定滑动条的范围（最小值、最大值），可以使用 `lv_slider_set_range(slider, min, max)`。

Modes（模式）

The slider can be one the following modes:

- `LV_SLIDER_MODE_NORMAL` A normal slider as described above
- `LV_SLIDER_MODE_SYMMETRICAL` Draw the indicator form the zero value to current value. Requires negaitve minimum range and positive maximum range.
- `LV_SLIDER_MODE_RANGE` Allows setting the start value too by `lv_bar_set_start_value(bar, new_value, LV_ANIM_ON/OFF)`. The start value has to be always smaller than the end value.

The mode can be changed with `lv_slider_set_mode(slider, LV_SLIDER_MODE_...)`

与 bar 类似，滑动条可以是以下模式之一：

- `LV_SLIDER_MODE_NORMAL` 像上面说的普通情况
- `LV_SLIDER_MODE_SYMMETRICAL` 这个模式下可以指定负的最小范围。但是只能从零值到当前值绘制指示器。
- `LV_SLIDER_MODE_RANGE` 在这个模式下也可以指定负的最小范围。这样滑动条的起始值可以不是 0，可以使用 `lv_slider_set_value` 设置结束值，使用 `lv_slider_set_left_value` 设置起始值，两侧如果不设置将使用默认值 0。要注意设置的起始值必须小于结束值。

可以使用 `lv_slider_set_mode(slider, LV_SLIDER_MODE_...)` 更改模式。

Knob-only mode（仅旋钮模式）

Normally, the slider can be adjusted either by dragging the knob, or by clicking on the slider bar. In the latter case the knob moves to the point clicked and slider value changes accordingly. In some cases it is desirable to set the slider to react on dragging the knob only. This feature is enabled by adding the `LV_OBJ_FLAG_ADV_HITTEST`: `lv_obj_add_flag(slider, LV_OBJ_FLAG_ADV_HITTEST)`.

通常，可以通过拖动旋钮或单击滑动条来调整滑动条的值。在后一种情况下，旋钮会移动到单击的点，指示器也会相应更改。在某些情况下，需要将滑动条设置为仅对拖动旋钮做出反应，可以通过添加 `LV_OBJ_FLAG_ADV_HITTEST` 来启用此功能：`lv_obj_add_flag(slider, LV_OBJ_FLAG_ADV_HITTEST)`。

Events（事件）

- `LV_EVENT_VALUE_CHANGED` Sent while the slider is being dragged or changed with keys. The event is sent continuously while the slider is dragged and once when released. Use `lv_slider_is_dragged` to detemine whether the Slider is still being dragged or has just been released.
- `LV_EVENT_DRAW_PART_BEGIN` and `LV_EVENT_DRAW_PART_END` are sent for the following parts.
  - `LV_SLIDER_DRAW_PART_KNOB` The main (right) knob of the slider
    - `part`: `LV_PART_KNOB`
    - `draw_area`: area of the indicator
    - `rect_dsc`
    - `id`: 0
  - `LV_SLIDER_DRAW_PART_KNOB` The left knob of the slider
    - `part`: `LV_PART_KNOB`
    - `draw_area`: area of the indicator
    - `rect_dsc`
    - `id`: 1

See the events of the *Bar* too.

Learn more about *Events*.

- `LV_EVENT_VALUE_CHANGED` 滑动条的值被改变时发送事件。拖动滑动条时会持续发送事件，最后释放时也会发送一次。使用 `lv_slider_is_dragged` 来确定滑动条是处于被拖动状态还是被释放状态。
- `LV_EVENT_DRAW_PART_BEGIN` 和 `LV_EVENT_DRAW_PART_END` 被发送用于以下部分。
  - `LV_SLIDER_DRAW_PART_KNOB` 滑动条的主（右）旋钮
    - 部分：`LV_PART_KNOB`

        * draw_area：指示器的区域
        * rect_dsc
        * id：0
    - LV_SLIDER_DRAW_PART_KNOB 滑块的左侧旋钮
        * 部分：LV_PART_KNOB
        * draw_area：指标的区域
        * rect_dsc
        * id：1

也可以查看*Bar* 的事件。

详细了解事件。

Keys（按键）

- LV_KEY_UP/RIGHT Increment the slider's value by 1
- LV_KEY_DOWN/LEFT Decrement the slider's value by 1

Learn more about *Keys*.

- LV_KEY_UP/RIGHT 将滑动条的值增加 1
- LV_KEY_DOWN/LEFT 将滑动条的值减 1

了解有关键 的更多信息。

Example

Simple Slider

```
#include "../../lv_examples.h"
#if LV_USE_SLIDER && LV_BUILD_EXAMPLES

static void slider_event_cb(lv_event_t * e);
static lv_obj_t * slider_label;

/**
 * A default slider with a label displaying the current value
 */
void lv_example_slider_1(void)
{
```

(下页继续)

(续上页)

```
    /*Create a slider in the center of the display*/
    lv_obj_t * slider = lv_slider_create(lv_scr_act());
    lv_obj_center(slider);
    lv_obj_add_event_cb(slider, slider_event_cb, LV_EVENT_VALUE_CHANGED, NULL);

    /*Create a label below the slider*/
    slider_label = lv_label_create(lv_scr_act());
    lv_label_set_text(slider_label, "0%");

    lv_obj_align_to(slider_label, slider, LV_ALIGN_OUT_BOTTOM_MID, 0, 10);
}

static void slider_event_cb(lv_event_t * e)
{
    lv_obj_t * slider = lv_event_get_target(e);
    char buf[8];
    lv_snprintf(buf, sizeof(buf), "%d%%", (int)lv_slider_get_value(slider));
    lv_label_set_text(slider_label, buf);
    lv_obj_align_to(slider_label, slider, LV_ALIGN_OUT_BOTTOM_MID, 0, 10);
}

#endif
```

```
#
# A default slider with a label displaying the current value
#
def slider_event_cb(e):

    slider = e.get_target()
    slider_label.set_text("{:d}%".format(slider.get_value()))
    slider_label.align_to(slider, lv.ALIGN.OUT_BOTTOM_MID, 0, 10)

# Create a slider in the center of the display
slider = lv.slider(lv.scr_act())
slider.center()
slider.add_event_cb(slider_event_cb, lv.EVENT.VALUE_CHANGED, None)

# Create a label below the slider
slider_label = lv.label(lv.scr_act())
slider_label.set_text("0%")

slider_label.align_to(slider, lv.ALIGN.OUT_BOTTOM_MID, 0, 10)
```

Slider with custom style

```
#include "../../lv_examples.h"
#if LV_USE_SLIDER && LV_BUILD_EXAMPLES



/**
 * Show how to style a slider.
 */
void lv_example_slider_2(void)
{
    /*Create a transition*/
    static const lv_style_prop_t props[] = {LV_STYLE_BG_COLOR, 0};
    static lv_style_transition_dsc_t transition_dsc;
    lv_style_transition_dsc_init(&transition_dsc, props, lv_anim_path_linear, 300, 0,␣
↪NULL);

    static lv_style_t style_main;
    static lv_style_t style_indicator;
    static lv_style_t style_knob;
    static lv_style_t style_pressed_color;
    lv_style_init(&style_main);
    lv_style_set_bg_opa(&style_main, LV_OPA_COVER);
    lv_style_set_bg_color(&style_main, lv_color_hex3(0xbbb));
    lv_style_set_radius(&style_main, LV_RADIUS_CIRCLE);
    lv_style_set_pad_ver(&style_main, -2); /*Makes the indicator larger*/

    lv_style_init(&style_indicator);
    lv_style_set_bg_opa(&style_indicator, LV_OPA_COVER);
    lv_style_set_bg_color(&style_indicator, lv_palette_main(LV_PALETTE_CYAN));
    lv_style_set_radius(&style_indicator, LV_RADIUS_CIRCLE);
    lv_style_set_transition(&style_indicator, &transition_dsc);

    lv_style_init(&style_knob);
    lv_style_set_bg_opa(&style_knob, LV_OPA_COVER);
    lv_style_set_bg_color(&style_knob, lv_palette_main(LV_PALETTE_CYAN));
    lv_style_set_border_color(&style_knob, lv_palette_darken(LV_PALETTE_CYAN, 3));
    lv_style_set_border_width(&style_knob, 2);
    lv_style_set_radius(&style_knob, LV_RADIUS_CIRCLE);
    lv_style_set_pad_all(&style_knob, 6); /*Makes the knob larger*/
    lv_style_set_transition(&style_knob, &transition_dsc);

    lv_style_init(&style_pressed_color);
```

(下页继续)

(续上页)

```
    lv_style_set_bg_color(&style_pressed_color, lv_palette_darken(LV_PALETTE_CYAN,␣
↪2));

    /*Create a slider and add the style*/
    lv_obj_t * slider = lv_slider_create(lv_scr_act());
    lv_obj_remove_style_all(slider);        /*Remove the styles coming from the␣
↪theme*/

    lv_obj_add_style(slider, &style_main, LV_PART_MAIN);
    lv_obj_add_style(slider, &style_indicator, LV_PART_INDICATOR);
    lv_obj_add_style(slider, &style_pressed_color, LV_PART_INDICATOR | LV_STATE_
↪PRESSED);
    lv_obj_add_style(slider, &style_knob, LV_PART_KNOB);
    lv_obj_add_style(slider, &style_pressed_color, LV_PART_KNOB | LV_STATE_PRESSED);

    lv_obj_center(slider);
}

#endif
```

```
#
# Show how to style a slider.
#
# Create a transition
props = [lv.STYLE.BG_COLOR, 0]
transition_dsc = lv.style_transition_dsc_t()
transition_dsc.init(props, lv.anim_t.path_linear, 300, 0, None)

style_main = lv.style_t()
style_indicator = lv.style_t()
style_knob = lv.style_t()
style_pressed_color = lv.style_t()
style_main.init()
style_main.set_bg_opa(lv.OPA.COVER)
style_main.set_bg_color(lv.color_hex3(0xbbb))
style_main.set_radius(lv.RADIUS.CIRCLE)
style_main.set_pad_ver(-2)                  # Makes the indicator larger

style_indicator.init()
style_indicator.set_bg_opa(lv.OPA.COVER)
style_indicator.set_bg_color(lv.palette_main(lv.PALETTE.CYAN))
style_indicator.set_radius(lv.RADIUS.CIRCLE)
style_indicator.set_transition(transition_dsc)
```

(下页继续)

(续上页)

```
style_knob.init()
style_knob.set_bg_opa(lv.OPA.COVER)
style_knob.set_bg_color(lv.palette_main(lv.PALETTE.CYAN))
style_knob.set_border_color(lv.palette_darken(lv.PALETTE.CYAN, 3))
style_knob.set_border_width(2)
style_knob.set_radius(lv.RADIUS.CIRCLE)
style_knob.set_pad_all(6)                    # Makes the knob larger
style_knob.set_transition(transition_dsc)

style_pressed_color.init()
style_pressed_color.set_bg_color(lv.palette_darken(lv.PALETTE.CYAN, 2))

# Create a slider and add the style
slider = lv.slider(lv.scr_act())
slider.remove_style_all()                    # Remove the styles coming from the theme

slider.add_style(style_main, lv.PART.MAIN)
slider.add_style(style_indicator, lv.PART.INDICATOR)
slider.add_style(style_pressed_color, lv.PART.INDICATOR | lv.STATE.PRESSED)
slider.add_style(style_knob, lv.PART.KNOB)
slider.add_style(style_pressed_color, lv.PART.KNOB | lv.STATE.PRESSED)

slider.center()
```

Slider with extended drawer

```
#include "../../lv_examples.h"
#if LV_USE_SLIDER && LV_BUILD_EXAMPLES

static void slider_event_cb(lv_event_t * e);

/**
 * Show the current value when the slider is pressed by extending the drawer
 *
 */
void lv_example_slider_3(void)
{
    /*Create a slider in the center of the display*/
    lv_obj_t * slider;
    slider = lv_slider_create(lv_scr_act());
```

(下页继续)

(续上页)

```
    lv_obj_center(slider);

    lv_slider_set_mode(slider, LV_SLIDER_MODE_RANGE);
    lv_slider_set_value(slider, 70, LV_ANIM_OFF);
    lv_slider_set_left_value(slider, 20, LV_ANIM_OFF);

    lv_obj_add_event_cb(slider, slider_event_cb, LV_EVENT_ALL, NULL);
    lv_obj_refresh_ext_draw_size(slider);
}

static void slider_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target(e);

    /*Provide some extra space for the value*/
    if(code == LV_EVENT_REFR_EXT_DRAW_SIZE) {
        lv_coord_t * size = lv_event_get_param(e);
        *size = LV_MAX(*size, 50);
    }
    else if(code == LV_EVENT_DRAW_PART_END) {
        lv_obj_draw_part_dsc_t * dsc = lv_event_get_param(e);
        if(dsc->part == LV_PART_INDICATOR) {
            char buf[16];
            lv_snprintf(buf, sizeof(buf), "%d - %d", (int)lv_slider_get_left_
↪value(obj), (int)lv_slider_get_value(obj));

            lv_point_t label_size;
            lv_txt_get_size(&label_size, buf, LV_FONT_DEFAULT, 0, 0, LV_COORD_MAX, 0);
            lv_area_t label_area;
            label_area.x1 = dsc->draw_area->x1 + lv_area_get_width(dsc->draw_area) /␣
↪2 - label_size.x / 2;
            label_area.x2 = label_area.x1 + label_size.x;
            label_area.y2 = dsc->draw_area->y1 - 10;
            label_area.y1 = label_area.y2 - label_size.y;

            lv_draw_label_dsc_t label_draw_dsc;
            lv_draw_label_dsc_init(&label_draw_dsc);

            lv_draw_label(dsc->draw_ctx, &label_draw_dsc, &label_area, buf, NULL);
        }
    }
}
```

(下页继续)

(续上页)

```
#endif
```

```
def slider_event_cb(e):
    code = e.get_code()
    obj = e.get_target()

    # Provide some extra space for the value
    if code == lv.EVENT.REFR_EXT_DRAW_SIZE:
        e.set_ext_draw_size(50)

    elif code == lv.EVENT.DRAW_PART_END:
        # print("DRAW_PART_END")
        dsc = lv.obj_draw_part_dsc_t.__cast__(e.get_param())
        # print(dsc)
        if dsc.part == lv.PART.INDICATOR:
            label_text = "{:d} - {:d}".format(obj.get_left_value(),slider.get_value())
            label_size = lv.point_t()
            lv.txt_get_size(label_size, label_text, lv.font_default(), 0, 0, lv.COORD.
↪MAX, 0)
            # print(label_size.x,label_size.y)
            label_area = lv.area_t()
            label_area.x1 = dsc.draw_area.x1 + dsc.draw_area.get_width() // 2 - label_
↪size.x // 2
            label_area.x2 = label_area.x1 + label_size.x
            label_area.y2 = dsc.draw_area.y1 - 10
            label_area.y1 = label_area.y2 - label_size.y

            label_draw_dsc = lv.draw_label_dsc_t()
            label_draw_dsc.init()

            dsc.draw_ctx.label(label_draw_dsc, label_area, label_text, None)
#
# Show the current value when the slider if pressed by extending the drawer
#
#
#Create a slider in the center of the display

slider = lv.slider(lv.scr_act())
slider.center()

slider.set_mode(lv.slider.MODE.RANGE)
slider.set_value(70, lv.ANIM.OFF)
```

(下页继续)

(续上页)

```
slider.set_left_value(20, lv.ANIM.OFF)

slider.add_event_cb(slider_event_cb, lv.EVENT.ALL, None)
slider.refresh_ext_draw_size()
```

API

Typedefs

typedef uint8_t **lv_slider_mode_t**

Enums

enum **[anonymous]**

Values:

enumerator **LV_SLIDER_MODE_NORMAL**

enumerator **LV_SLIDER_MODE_SYMMETRICAL**

enumerator **LV_SLIDER_MODE_RANGE**

enum **lv_slider_draw_part_type_t**

`type` field in `lv_obj_draw_part_dsc_t` if `class_p = lv_slider_class` Used in `LV_EVENT_DRAW_PART_BEGIN` and `LV_EVENT_DRAW_PART_END`

Values:

enumerator **LV_SLIDER_DRAW_PART_KNOB**

The main (right) knob's rectangle

enumerator **LV_SLIDER_DRAW_PART_KNOB_LEFT**

The left knob's rectangle

Functions

lv_obj_t ***lv_slider_create**(*lv_obj_t* *parent)

Create a slider object

参数 **parent** -- pointer to an object, it will be the parent of the new slider.

返回 pointer to the created slider

static inline void **lv_slider_set_value**(*lv_obj_t* *obj, int32_t value, *lv_anim_enable_t* anim)

Set a new value on the slider

参数

- **obj** -- pointer to a slider object
- **value** -- the new value
- **anim** -- LV_ANIM_ON: set the value with an animation; LV_ANIM_OFF: change the value immediately

static inline void **lv_slider_set_left_value**(*lv_obj_t* *obj, int32_t value, *lv_anim_enable_t* anim)

Set a new value for the left knob of a slider

参数

- **obj** -- pointer to a slider object
- **value** -- new value
- **anim** -- LV_ANIM_ON: set the value with an animation; LV_ANIM_OFF: change the value immediately

static inline void **lv_slider_set_range**(*lv_obj_t* *obj, int32_t min, int32_t max)

Set minimum and the maximum values of a bar

参数

- **obj** -- pointer to the slider object
- **min** -- minimum value
- **max** -- maximum value

static inline void **lv_slider_set_mode**(*lv_obj_t* *obj, *lv_slider_mode_t* mode)

Set the mode of slider.

参数

- **obj** -- pointer to a slider object
- **mode** -- the mode of the slider. See ::lv_slider_mode_t

static inline int32_t **lv_slider_get_value**(const *lv_obj_t* *obj)

Get the value of the main knob of a slider

参数 **obj** -- pointer to a slider object

返回 the value of the main knob of the slider

static inline int32_t **lv_slider_get_left_value**(const *lv_obj_t* *obj)

Get the value of the left knob of a slider

参数 **obj** -- pointer to a slider object

返回 the value of the left knob of the slider

static inline int32_t **lv_slider_get_min_value**(const *lv_obj_t* *obj)

Get the minimum value of a slider

参数 **obj** -- pointer to a slider object

返回 the minimum value of the slider

static inline int32_t **lv_slider_get_max_value**(const *lv_obj_t* *obj)

Get the maximum value of a slider

参数 **obj** -- pointer to a slider object

返回 the maximum value of the slider

bool **lv_slider_is_dragged**(const *lv_obj_t* *obj)

Give the slider is being dragged or not

参数 **obj** -- pointer to a slider object

返回 true: drag in progress false: not dragged

static inline *lv_slider_mode_t* **lv_slider_get_mode**(*lv_obj_t* *slider)

Get the mode of the slider.

参数 **obj** -- pointer to a bar object

返回 see ::lv_slider_mode_t

Variables

const lv_obj_class_t **lv_slider_class**

struct **lv_slider_t**

Public Members

lv_bar_t **bar**

lv_area_t **left_knob_area**

lv_area_t **right_knob_area**

int32_t ***value_to_set**

uint8_t **dragging**

uint8_t **left_knob_focus**

Switch（开关）(lv_switch)

Overview（概述）

The Switch looks like a little slider and can be used to turn something on and off.

开关看起来像一个小滑块，开关的功能类似于按钮，也可以用来打开和关闭某些东西。

Parts and Styles（零件和样式）

- `LV_PART_MAIN` The background of the switch uses all the typical background style properties. `padding` makes the indicator smaller in the respective direction.
- `LV_PART_INDICATOR` The indicator that shows the current state of the switch. Also uses all the typical background style properties.
- `LV_PART_KNOB` A rectangle (or circle) drawn at left or right side of the indicator. Also uses all the typical background properties to describe the knob(s). By default the knob is square (with a optional corner radius) with side length equal to the smaller side of the slider. The knob can be made larger with the `padding` values. Padding values can be asymmetric too.

开关包括以下 3 个零件：

- `LV_PART_MAIN` 背景。修改其 padding 会让下面的 (指示器和旋钮) 在相应方向上的大小发生变化。
- `LV_PART_INDICATOR` 显示开关状态的指示器。
- `LV_PART_KNOB` 在指标左侧或右侧的旋钮。默认情况下，旋钮是圆形的，边长等于滑块的较小边。可以修改 padding 值使旋钮变大，填充值可以是不对称的。

示例：

```
// 修改开关背景部分的颜色
lv_obj_set_style_bg_color(sw, lv_color_hex(0xc43e1c), LV_PART_MAIN);
```

(下页继续)

(续上页)

```
// 修改开关状态指示器部分，关闭状态时的背景颜色
lv_obj_set_style_bg_opa(sw, 100, LV_PART_INDICATOR);
lv_obj_set_style_bg_color(sw, lv_color_hex(0xc43e1c), LV_PART_INDICATOR);

// 修改开关状态指示器部分，打开状态时的背景颜色
lv_obj_set_style_bg_color(sw, lv_color_hex(0x7719aa), LV_PART_INDICATOR | LV_STATE_
↪CHECKED);

// 修改开关旋钮部分的颜色
lv_obj_set_style_bg_color(sw, lv_color_hex(0xc43e1c), LV_PART_KNOB);
```

Usage（用法）

Change state（改变状态）

When the switch is turned on it goes to `LV_STATE_CHECKED`. To get the current satte of the switch use `lv_obj_has_state(switch, LV_STATE_CHECKED)`. To manually turn the switch on/off call `lvobj_add/clear_state(switch, LV_STATE_CHECKED)`.

当开关被打开时，开关的状态会变为 LV_STATE_CHECKED 。我们可以通过下面这个接口获取开关当前的状态：

```
lv_obj_has_state(switch, LV_STATE_CHECKED);   // 返回 bool 类型， 开-1 ； 关-2
```

一般我们通过触摸或按键控制让开关打开/关闭，其实我们还可以通过下面这个接口来主动打开/关闭开关：

```
lv_obj_add_state(switch, LV_STATE_CHECKED);        // 开
lv_obj_clear_state(switch, LV_STATE_CHECKED);        // 关
```

我们可以通过下面的接口让按钮处于不可更改状态：

```
lv_obj_add_state(sw, LV_STATE_DISABLED);                                    // 当前状态是关，
并且不可更改
lv_obj_add_state(sw, LV_STATE_CHECKED | LV_STATE_DISABLED);  // 当前状态是开，并且不可更改
```

要让按钮恢复可以更改的状态，我们只要将 LV_STATE_DISABLED 清除即可：

```
lv_obj_clear_state(switch, LV_STATE_ DISABLED);                             // 清除禁用状态，
按钮可正常使用
```

Events（事件）

- LV_EVENT_VALUE_CHANGED Sent when the switch changes state.

See the events of the *Base object* too.

Learn more about *Events*.

正常情况下，当开关被点击并且状态发生改变时，会触发 LV_EVENT_VALUE_CHANGED 事件类型。也就是说我们可以在 LV_EVENT_VALUE_CHANGED 事件类型中处理事件。

参见*Base object* 的事件。

详细了解事件。

Keys（按键）

- LV_KEY_UP/RIGHT Turns on the slider
- LV_KEY_DOWN/LEFT Turns off the slider
- LV_KEY_ENTER Toggles the switch

Learn more about *Keys*.

- LV_KEY_UP/RIGHT 开
- LV_KEY_DOWN/LEFT 关
- LV_KEY_ENTER 切换开关

了解有关按键 的更多信息。

Example

Simple Switch

```
#include "../../lv_examples.h"
#if LV_USE_SWITCH && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        LV_LOG_USER("State: %s\n", lv_obj_has_state(obj, LV_STATE_CHECKED) ? "On" :
↪"Off");
    }
```

(下页继续)

(续上页)

```
}

void lv_example_switch_1(void)
{
    lv_obj_set_flex_flow(lv_scr_act(), LV_FLEX_FLOW_COLUMN);
    lv_obj_set_flex_align(lv_scr_act(), LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER,␣
↪LV_FLEX_ALIGN_CENTER);

    lv_obj_t * sw;

    sw = lv_switch_create(lv_scr_act());
    lv_obj_add_event_cb(sw, event_handler, LV_EVENT_ALL, NULL);

    sw = lv_switch_create(lv_scr_act());
    lv_obj_add_state(sw, LV_STATE_CHECKED);
    lv_obj_add_event_cb(sw, event_handler, LV_EVENT_ALL, NULL);

    sw = lv_switch_create(lv_scr_act());
    lv_obj_add_state(sw, LV_STATE_DISABLED);
    lv_obj_add_event_cb(sw, event_handler, LV_EVENT_ALL, NULL);

    sw = lv_switch_create(lv_scr_act());
    lv_obj_add_state(sw, LV_STATE_CHECKED | LV_STATE_DISABLED);
    lv_obj_add_event_cb(sw, event_handler, LV_EVENT_ALL, NULL);
}

#endif
```

```
def event_handler(e):
    code = e.get_code()
    obj = e.get_target()
    if code == lv.EVENT.VALUE_CHANGED:
        if obj.has_state(lv.STATE.CHECKED):
            print("State: on")
        else:
            print("State: off")


lv.scr_act().set_flex_flow(lv.FLEX_FLOW.COLUMN)
lv.scr_act().set_flex_align(lv.FLEX_ALIGN.CENTER, lv.FLEX_ALIGN.CENTER, lv.FLEX_ALIGN.
↪CENTER)

sw = lv.switch(lv.scr_act())
```

(下页继续)

(续上页)

```
sw.add_event_cb(event_handler,lv.EVENT.ALL, None)

sw = lv.switch(lv.scr_act())
sw.add_state(lv.STATE.CHECKED)
sw.add_event_cb(event_handler, lv.EVENT.ALL, None)

sw = lv.switch(lv.scr_act())
sw.add_state(lv.STATE.DISABLED)
sw.add_event_cb(event_handler, lv.EVENT.ALL, None)

sw = lv.switch(lv.scr_act())
sw.add_state(lv.STATE.CHECKED | lv.STATE.DISABLED)
sw.add_event_cb(event_handler, lv.EVENT.ALL, None)
```

API

Functions

lv_obj_t ***lv_switch_create**(*lv_obj_t* *parent)

Create a switch object

参数 **parent** -- pointer to an object, it will be the parent of the new switch

返回 pointer to the created switch

Variables

const lv_obj_class_t **lv_switch_class**

struct **lv_switch_t**

Public Members

lv_obj_t **obj**

int32_t **anim_state**

Table（表）(lv_table)

Overview

Tables, as usual, are built from rows, columns, and cells containing texts.

The Table object is very lightweight because only the texts are stored. No real objects are created for cells but they are just drawn on the fly.

表格是由包含文本的行、列和单元格构建的。表格对象非常轻量级，因为仅存储文本。没有为细胞创建真实的对象，但它们只是即时绘制的。

Parts and Styles（部分和样式）

- `LV_PART_MAIN` The background of the table uses all the typical background style properties.
- `LV_PART_ITEMS` The cells of the table also use all the typical background style properties and the text properties.
- `LV_PART_MAIN` 表格的背景使用了所有典型的背景样式属性。
- `LV_PART_ITEMS` 表格的单元格也使用所有典型的背景样式属性和文本属性。

Usage（用法）

Set cell value（设置单元格的值）

The cells can store only text so numbers need to be converted to text before displaying them in a table.

`lv_table_set_cell_value(table, row, col, "Content")`. The text is saved by the table so it can be even a local variable.

Line breaks can be used in the text like `"Value\n60.3"`.

New rows and columns are automatically added is required

单元格只能存储文本，因此在将数字显示在表格中之前，需要将其转换为文本。

`lv_table_set_cell_value(table, row, col, "Content")`。文本由表保存，因此它甚至可以是局部变量。

可以在文本中使用换行符，例如 `"Value\n60.3"`。

需要自动添加新的行和列

Rows and Columns（行和列）

To explicitly set number of rows and columns use `lv_table_set_row_cnt(table, row_cnt)` and `lv_table_set_col_cnt(table, col_cnt)`

要明确设置行数和列数，请使用 `lv_table_set_row_cnt(table, row_cnt)` 和 `lv_table_set_col_cnt(table, col_cnt)`

Width and Height（宽度和高度）

The width of the columns can be set with `lv_table_set_col_width(table, col_id, width)`. The overall width of the Table object will be set to the sum of columns widths.

The height is calculated automatically from the cell styles (font, padding etc) and the number of rows.

列的宽度可以通过 `lv_table_set_col_width(table, col_id, width)` 设置。Table 对象的总宽度将设置为列宽的总和。

高度是根据单元格样式（字体、填充等）和行数自动计算的。

Merge cells（合并单元格）

Cells can be merged horizontally with `lv_table_set_cell_merge_right(table, col, row, true)`. To merge more adjacent cells call this function for each cell.

单元格可以使用 `lv_table_set_cell_merge_right(table, col, row, true)` 进行水平合并。要合并更多相邻单元格，请为每个单元格调用此函数。

Scroll（滚动）

If the label's width or height is set to `LV_SIZE_CONTENT` that size will be used to show the whole table in the respective direction. E.g. `lv_obj_set_size(table, LV_SIZE_CONTENT, LV_SIZE_CONTENT)` automatically sets the table size to show all the columns and rows.

If the width or height is set to a smaller number than the "intrinsic" size then the table becomes scrollable.

如果标签的宽度或高度设置为“LV_SIZE_CONTENT”，则该尺寸将用于在相应方向上显示整个表格。例如。`lv_obj_set_size(table, LV_SIZE_CONTENT, LV_SIZE_CONTENT)` 自动设置表格大小以显示所有列和行。

如果宽度或高度设置为小于“固有”大小的数字，则表格变为可滚动的。

Events（事件）

- `LV_EVENT_DRAW_PART_BEGIN` and `LV_EVENT_DRAW_PART_END` are sent for the following types:
  - `LV_TABLE_DRAW_PART_CELL` The individual cells of the table
    - `part`: `LV_PART_ITEMS`
    - `draw_area`: area of the indicator
    - `rect_dsc`
    - `label_dsc`
    - `id`: current row × col count + current column

See the events of the *Base object* too.

Learn more about *Events*.

- 为以下类型发送 `LV_EVENT_DRAW_PART_BEGIN` 和 `LV_EVENT_DRAW_PART_END`：
  - `LV_TABLE_DRAW_PART_CELL` 表格的各个单元格
    - 部分：`LV_PART_ITEMS`
    - `draw_area`：指标的区域 -`rect_dsc`
    - `label_dsc`
    - `id`：当前行 × 列数 + 当前列

参见*Base object* 的事件。

详细了解事件。

Keys（按键）

No *Keys* are processed by the object type.

Learn more about *Keys*.

对象类型不处理 *Keys*。

了解有关键 的更多信息。

Example

Simple table

```
#include "../../lv_examples.h"
#if LV_USE_TABLE && LV_BUILD_EXAMPLES

static void draw_part_event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target(e);
    lv_obj_draw_part_dsc_t * dsc = lv_event_get_param(e);
    /*If the cells are drawn...*/
    if(dsc->part == LV_PART_ITEMS) {
        uint32_t row = dsc->id /  lv_table_get_col_cnt(obj);
        uint32_t col = dsc->id - row * lv_table_get_col_cnt(obj);

        /*Make the texts in the first cell center aligned*/
        if(row == 0) {
            dsc->label_dsc->align = LV_TEXT_ALIGN_CENTER;
            dsc->rect_dsc->bg_color = lv_color_mix(lv_palette_main(LV_PALETTE_BLUE),␣
↪dsc->rect_dsc->bg_color, LV_OPA_20);
            dsc->rect_dsc->bg_opa = LV_OPA_COVER;
        }
        /*In the first column align the texts to the right*/
        else if(col == 0) {
            dsc->label_dsc->align = LV_TEXT_ALIGN_RIGHT;
        }

        /*MAke every 2nd row grayish*/
        if((row != 0 && row % 2) == 0) {
            dsc->rect_dsc->bg_color = lv_color_mix(lv_palette_main(LV_PALETTE_GREY),␣
↪dsc->rect_dsc->bg_color, LV_OPA_10);
            dsc->rect_dsc->bg_opa = LV_OPA_COVER;
        }
    }
}


void lv_example_table_1(void)
{
    lv_obj_t * table = lv_table_create(lv_scr_act());

    /*Fill the first column*/
    lv_table_set_cell_value(table, 0, 0, "Name");
```

(下页继续)

(续上页)

```
    lv_table_set_cell_value(table, 1, 0, "Apple");
    lv_table_set_cell_value(table, 2, 0, "Banana");
    lv_table_set_cell_value(table, 3, 0, "Lemon");
    lv_table_set_cell_value(table, 4, 0, "Grape");
    lv_table_set_cell_value(table, 5, 0, "Melon");
    lv_table_set_cell_value(table, 6, 0, "Peach");
    lv_table_set_cell_value(table, 7, 0, "Nuts");

    /*Fill the second column*/
    lv_table_set_cell_value(table, 0, 1, "Price");
    lv_table_set_cell_value(table, 1, 1, "$7");
    lv_table_set_cell_value(table, 2, 1, "$4");
    lv_table_set_cell_value(table, 3, 1, "$6");
    lv_table_set_cell_value(table, 4, 1, "$2");
    lv_table_set_cell_value(table, 5, 1, "$5");
    lv_table_set_cell_value(table, 6, 1, "$1");
    lv_table_set_cell_value(table, 7, 1, "$9");

    /*Set a smaller height to the table. It'll make it scrollable*/
    lv_obj_set_height(table, 200);
    lv_obj_center(table);

    /*Add an event callback to to apply some custom drawing*/
    lv_obj_add_event_cb(table, draw_part_event_cb, LV_EVENT_DRAW_PART_BEGIN, NULL);
}

#endif
```

```
def draw_part_event_cb(e):
    obj = e.get_target()
    dsc = lv.obj_draw_part_dsc_t.__cast__(e.get_param())
    # If the cells are drawn../
    if dsc.part == lv.PART.ITEMS:
        row = dsc.id //  obj.get_col_cnt()
        col = dsc.id - row * obj.get_col_cnt()

        # Make the texts in the first cell center aligned
        if row == 0:
            dsc.label_dsc.align = lv.TEXT_ALIGN.CENTER
            dsc.rect_dsc.bg_color = lv.palette_main(lv.PALETTE.BLUE).color_mix(dsc.
↪rect_dsc.bg_color, lv.OPA._20)
            dsc.rect_dsc.bg_opa = lv.OPA.COVER
```

(下页继续)

(续上页)

```
        # In the first column align the texts to the right
        elif col == 0:
            dsc.label_dsc.flag = lv.TEXT_ALIGN.RIGHT

        # Make every 2nd row grayish
        if row != 0 and (row % 2) == 0:
            dsc.rect_dsc.bg_color = lv.palette_main(lv.PALETTE.GREY).color_mix(dsc.
↪rect_dsc.bg_color, lv.OPA._10)
            dsc.rect_dsc.bg_opa = lv.OPA.COVER


table = lv.table(lv.scr_act())

# Fill the first column
table.set_cell_value(0, 0, "Name")
table.set_cell_value(1, 0, "Apple")
table.set_cell_value(2, 0, "Banana")
table.set_cell_value(3, 0, "Lemon")
table.set_cell_value(4, 0, "Grape")
table.set_cell_value(5, 0, "Melon")
table.set_cell_value(6, 0, "Peach")
table.set_cell_value(7, 0, "Nuts")

# Fill the second column
table.set_cell_value(0, 1, "Price")
table.set_cell_value(1, 1, "$7")
table.set_cell_value(2, 1, "$4")
table.set_cell_value(3, 1, "$6")
table.set_cell_value(4, 1, "$2")
table.set_cell_value(5, 1, "$5")
table.set_cell_value(6, 1, "$1")
table.set_cell_value(7, 1, "$9")

# Set a smaller height to the table. It'll make it scrollable
table.set_height(200)
table.center()

# Add an event callback to apply some custom drawing
table.add_event_cb(draw_part_event_cb, lv.EVENT.DRAW_PART_BEGIN, None)
```

Lightweighted list from table

```
#include "../../lv_examples.h"
#if LV_USE_TABLE && LV_BUILD_EXAMPLES

#define ITEM_CNT 200

static void draw_event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target(e);
    lv_obj_draw_part_dsc_t * dsc = lv_event_get_draw_part_dsc(e);
    /*If the cells are drawn...*/
    if(dsc->part == LV_PART_ITEMS) {
        bool chk = lv_table_has_cell_ctrl(obj, dsc->id, 0, LV_TABLE_CELL_CTRL_CUSTOM_
↪1);

        lv_draw_rect_dsc_t rect_dsc;
        lv_draw_rect_dsc_init(&rect_dsc);
        rect_dsc.bg_color = chk ? lv_theme_get_color_primary(obj) : lv_palette_
↪lighten(LV_PALETTE_GREY, 2);
        rect_dsc.radius = LV_RADIUS_CIRCLE;

        lv_area_t sw_area;
        sw_area.x1 = dsc->draw_area->x2 - 50;
        sw_area.x2 = sw_area.x1 + 40;
        sw_area.y1 = dsc->draw_area->y1 + lv_area_get_height(dsc->draw_area) / 2 - 10;
        sw_area.y2 = sw_area.y1 + 20;
        lv_draw_rect(dsc->draw_ctx, &rect_dsc, &sw_area);

        rect_dsc.bg_color = lv_color_white();
        if(chk) {
            sw_area.x2 -= 2;
            sw_area.x1 = sw_area.x2 - 16;
        } else {
            sw_area.x1 += 2;
            sw_area.x2 = sw_area.x1 + 16;
        }
        sw_area.y1 += 2;
        sw_area.y2 -= 2;
        lv_draw_rect(dsc->draw_ctx, &rect_dsc, &sw_area);
    }
}

static void change_event_cb(lv_event_t * e)
```

(下页继续)

(续上页)

```
{
    lv_obj_t * obj = lv_event_get_target(e);
    uint16_t col;
    uint16_t row;
    lv_table_get_selected_cell(obj, &row, &col);
    bool chk = lv_table_has_cell_ctrl(obj, row, 0, LV_TABLE_CELL_CTRL_CUSTOM_1);
    if(chk) lv_table_clear_cell_ctrl(obj, row, 0, LV_TABLE_CELL_CTRL_CUSTOM_1);
    else lv_table_add_cell_ctrl(obj, row, 0, LV_TABLE_CELL_CTRL_CUSTOM_1);
}


/**
 * A very light-weighted list created from table
 */
void lv_example_table_2(void)
{
    /*Measure memory usage*/
    lv_mem_monitor_t mon1;
    lv_mem_monitor(&mon1);

    uint32_t t = lv_tick_get();

    lv_obj_t * table = lv_table_create(lv_scr_act());

    /*Set a smaller height to the table. It'll make it scrollable*/
    lv_obj_set_size(table, LV_SIZE_CONTENT, 200);

    lv_table_set_col_width(table, 0, 150);
    lv_table_set_row_cnt(table, ITEM_CNT); /*Not required but avoids a lot of memory␣
↪reallocation lv_table_set_set_value*/
    lv_table_set_col_cnt(table, 1);

    /*Don't make the cell pressed, we will draw something different in the event*/
    lv_obj_remove_style(table, NULL, LV_PART_ITEMS | LV_STATE_PRESSED);

    uint32_t i;
    for(i = 0; i < ITEM_CNT; i++) {
        lv_table_set_cell_value_fmt(table, i, 0, "Item %"LV_PRIu32, i + 1);
    }

    lv_obj_align(table, LV_ALIGN_CENTER, 0, -20);

    /*Add an event callback to to apply some custom drawing*/
```

(下页继续)

(续上页)

```
    lv_obj_add_event_cb(table, draw_event_cb, LV_EVENT_DRAW_PART_END, NULL);
    lv_obj_add_event_cb(table, change_event_cb, LV_EVENT_VALUE_CHANGED, NULL);

    lv_mem_monitor_t mon2;
    lv_mem_monitor(&mon2);

    uint32_t mem_used = mon1.free_size - mon2.free_size;

    uint32_t elaps = lv_tick_elaps(t);

    lv_obj_t * label = lv_label_create(lv_scr_act());
    lv_label_set_text_fmt(label, "%"LV_PRIu32" items were created in %"LV_PRIu32" ms\n
↪"
                                 "using %"LV_PRIu32" bytes of memory",
                                 ITEM_CNT, elaps, mem_used);

    lv_obj_align(label, LV_ALIGN_BOTTOM_MID, 0, -10);

}

#endif
```

```
from utime import ticks_ms
import gc

ITEM_CNT = 200

def draw_event_cb(e):
    obj = e.get_target()
    dsc = lv.obj_draw_part_dsc_t.__cast__(e.get_param())
    # If the cells are drawn...
    if dsc.part == lv.PART.ITEMS:
        chk = obj.has_cell_ctrl(dsc.id, 0, lv.table.CELL_CTRL.CUSTOM_1)

        rect_dsc = lv.draw_rect_dsc_t()
        rect_dsc.init()

        if chk:
            rect_dsc.bg_color = lv.theme_get_color_primary(obj)
        else:
            rect_dsc.bg_color = lv.palette_lighten(lv.PALETTE.GREY, 2)

        rect_dsc.radius = lv.RADIUS.CIRCLE
```

(下页继续)

(续上页)

```
        sw_area = lv.area_t()
        sw_area.x1 = dsc.draw_area.x2 - 50
        sw_area.x2 = sw_area.x1 + 40
        sw_area.y1 = dsc.draw_area.y1 + dsc.draw_area.get_height() // 2 - 10
        sw_area.y2 = sw_area.y1 + 20
        dsc.draw_ctx.rect(rect_dsc, sw_area)

        rect_dsc.bg_color = lv.color_white()

        if chk:
            sw_area.x2 -= 2
            sw_area.x1 = sw_area.x2 - 16
        else:
            sw_area.x1 += 2
            sw_area.x2 = sw_area.x1 + 16
        sw_area.y1 += 2
        sw_area.y2 -= 2
        dsc.draw_ctx.rect(rect_dsc, sw_area)

def change_event_cb(e):
    obj = e.get_target()
    row = lv.C_Pointer()
    col = lv.C_Pointer()
    table.get_selected_cell(row, col)
    # print("row: ",row.uint_val)

    chk = table.has_cell_ctrl(row.uint_val, 0, lv.table.CELL_CTRL.CUSTOM_1)
    if chk:
        table.clear_cell_ctrl(row.uint_val, 0, lv.table.CELL_CTRL.CUSTOM_1)
    else:
        table.add_cell_ctrl(row.uint_val, 0, lv.table.CELL_CTRL.CUSTOM_1)

#
# A very light-weighted list created from table
#

# Measure memory usage
gc.enable()
gc.collect()
mem_free = gc.mem_free()
print("mem_free: ", mem_free)
t = ticks_ms()
```

(下页继续)

(续上页)

```
print("ticks: ", t)
table = lv.table(lv.scr_act())

# Set a smaller height to the table. It'll make it scrollable
table.set_size(150, 200)

table.set_col_width(0, 150)
table.set_row_cnt(ITEM_CNT)  # Not required but avoids a lot of memory reallocation␣
↪lv_table_set_set_value
table.set_col_cnt(1)

# Don't make the cell pressed, we will draw something different in the event
table.remove_style(None, lv.PART.ITEMS | lv.STATE.PRESSED)

for i in range(ITEM_CNT):
    table.set_cell_value(i, 0, "Item " + str(i+1))

table.align(lv.ALIGN.CENTER, 0, -20)

# Add an event callback to apply some custom drawing
table.add_event_cb(draw_event_cb, lv.EVENT.DRAW_PART_END, None)
table.add_event_cb(change_event_cb, lv.EVENT.VALUE_CHANGED, None)

gc.collect()
mem_used = mem_free - gc.mem_free()
elaps = ticks_ms()-t

label = lv.label(lv.scr_act())
label.set_text(str(ITEM_CNT) + " items were created in " + str(elaps) + " ms\n using
↪" + str(mem_used) + " bytes of memory")
#label.set_text(str(ITEM_CNT) + " items were created in " + str(elaps) + " ms")

label.align(lv.ALIGN.BOTTOM_MID, 0, -10)
```

MicroPython

No examples yet.

API

Typedefs

typedef uint8_t **lv_table_cell_ctrl_t**

Enums

enum **[anonymous]**

Values:

enumerator **LV_TABLE_CELL_CTRL_MERGE_RIGHT**

enumerator **LV_TABLE_CELL_CTRL_TEXT_CROP**

enumerator **LV_TABLE_CELL_CTRL_CUSTOM_1**

enumerator **LV_TABLE_CELL_CTRL_CUSTOM_2**

enumerator **LV_TABLE_CELL_CTRL_CUSTOM_3**

enumerator **LV_TABLE_CELL_CTRL_CUSTOM_4**

enum **lv_table_draw_part_type_t**

`type` field in `lv_obj_draw_part_dsc_t` if `class_p = lv_table_class` Used in `LV_EVENT_DRAW_PART_BEGIN` and `LV_EVENT_DRAW_PART_END`

Values:

enumerator **LV_TABLE_DRAW_PART_CELL**

A cell

Functions

LV_EXPORT_CONST_INT(LV_TABLE_CELL_NONE)

lv_obj_t ***lv_table_create**(*lv_obj_t* *parent)

Create a table object

参数 **parent** -- pointer to an object, it will be the parent of the new table

返回 pointer to the created table

void **lv_table_set_cell_value**(*lv_obj_t* *obj, uint16_t row, uint16_t col, const char *txt)

Set the value of a cell.

注解: New roes/columns are added automatically if required

参数

- **obj** -- pointer to a Table object
- **row** -- id of the row [0 .. row_cnt -1]
- **col** -- id of the column [0 .. col_cnt -1]
- **txt** -- text to display in the cell. It will be copied and saved so this variable is not required after this function call.

void **lv_table_set_cell_value_fmt**(*lv_obj_t* *obj, uint16_t row, uint16_t col, const char *fmt, ...)

Set the value of a cell. Memory will be allocated to store the text by the table.

注解: New roes/columns are added automatically if required

参数

- **obj** -- pointer to a Table object
- **row** -- id of the row [0 .. row_cnt -1]
- **col** -- id of the column [0 .. col_cnt -1]
- **fmt** -- `printf`-like format

void **lv_table_set_row_cnt**(*lv_obj_t* *obj, uint16_t row_cnt)

Set the number of rows

参数

- **obj** -- table pointer to a Table object
- **row_cnt** -- number of rows

void **lv_table_set_col_cnt**(*lv_obj_t* *obj, uint16_t col_cnt)

Set the number of columns

参数

- **obj** -- table pointer to a Table object
- **col_cnt** -- number of columns.

void **lv_table_set_col_width**(*lv_obj_t* *obj, uint16_t col_id, lv_coord_t w)

Set the width of a column

参数

- **obj** -- table pointer to a Table object
- **col_id** -- id of the column [0 .. LV_TABLE_COL_MAX -1]
- **w** -- width of the column

void **lv_table_add_cell_ctrl**(*lv_obj_t* *obj, uint16_t row, uint16_t col, *lv_table_cell_ctrl_t* ctrl)

Add control bits to the cell.

参数

- **obj** -- pointer to a Table object
- **row** -- id of the row [0 .. row_cnt -1]
- **col** -- id of the column [0 .. col_cnt -1]
- **ctrl** -- OR-ed values from ::lv_table_cell_ctrl_t

void **lv_table_clear_cell_ctrl**(*lv_obj_t* *obj, uint16_t row, uint16_t col, *lv_table_cell_ctrl_t* ctrl)

Clear control bits of the cell.

参数

- **obj** -- pointer to a Table object
- **row** -- id of the row [0 .. row_cnt -1]
- **col** -- id of the column [0 .. col_cnt -1]
- **ctrl** -- OR-ed values from ::lv_table_cell_ctrl_t

const char ***lv_table_get_cell_value**(*lv_obj_t* *obj, uint16_t row, uint16_t col)

Get the value of a cell.

参数

- **obj** -- pointer to a Table object
- **row** -- id of the row [0 .. row_cnt -1]
- **col** -- id of the column [0 .. col_cnt -1]

返回 text in the cell

uint16_t **lv_table_get_row_cnt**(*lv_obj_t* *obj)

Get the number of rows.

参数 **obj** -- table pointer to a Table object

返回 number of rows.

uint16_t **lv_table_get_col_cnt**(*lv_obj_t* *obj)

Get the number of columns.

参数 **obj** -- table pointer to a Table object

返回 number of columns.

lv_coord_t **lv_table_get_col_width**(*lv_obj_t* *obj, uint16_t col)

Get the width of a column

参数

- **obj** -- table pointer to a Table object
- **col** -- id of the column [0 .. LV_TABLE_COL_MAX -1]

返回 width of the column

bool **lv_table_has_cell_ctrl**(*lv_obj_t* *obj, uint16_t row, uint16_t col, *lv_table_cell_ctrl_t* ctrl)

Get whether a cell has the control bits

参数

- **obj** -- pointer to a Table object
- **row** -- id of the row [0 .. row_cnt -1]
- **col** -- id of the column [0 .. col_cnt -1]
- **ctrl** -- OR-ed values from ::lv_table_cell_ctrl_t

返回 true: all control bits are set; false: not all control bits are set

void **lv_table_get_selected_cell**(*lv_obj_t* *obj, uint16_t *row, uint16_t *col)

Get the selected cell (pressed and or focused)

参数

- **obj** -- pointer to a table object
- **row** -- pointer to variable to store the selected row (LV_TABLE_CELL_NONE: if no cell selected)
- **col** -- pointer to variable to store the selected column (LV_TABLE_CELL_NONE: if no cell selected)

Variables

const lv_obj_class_t **lv_table_class**

struct **lv_table_t**

Public Members

lv_obj_t **obj**

uint16_t **col_cnt**

uint16_t **row_cnt**

char ****cell_data**

lv_coord_t ***row_h**

lv_coord_t ***col_w**

uint16_t **col_act**

uint16_t **row_act**

Text area（文本框）(lv_textarea)

Overview（概述）

The Text Area is a *Base object* with a *Label* and a cursor on it. Texts or characters can be added to it. Long lines are wrapped and when the text becomes long enough the Text area can be scrolled.

One line mode and password modes are supported.

文本框是一个基础对象，其上面有一个标签 *(Label)* 和一个光标 (cursor)。我们可以向文本框中添加文本或字符。长行会被换行，当文本内容变得足够长时 (文本框可视区域容纳不下时)，可以滚动文本框。

支持单行输入模式和密码输入模式。

Parts and Styles（部分和样式）

- `LV_PART_MAIN` The background of the text area. Uses all the typical backgrond style properties and the text related style properties including `text_align` to align the text to the left, right or center.
- `LV_PART_SCROLLBAR` The scrollbar that is shown when the text is too long.
- `LV_PART_SELECTED` Detemines the style of the selected text. Only `text_color` and `bg_color` style properties can be used. `bg_color` should be set directly on the label of the text area.

- LV_PART_CURSOR Marks the position where the characters are inserted. The cursor's area is always the bounding box of the current character. A block cursor can be created by adding a background color and background opacity to LV_PART_CURSOR's style. The create line cursor leave the cursor transparent and set a left border. The anim_time style property sets the cursor's blink time.
- LV_PART_TEXTAREA_PLACEHOLDER Unique to Text Area, allows styling the placeholder text.
- LV_PART_MAIN 文本框的背景。使用所有组件都有的典型的背景样式属性和与文本相关的样式属性（包括 text_align ）将文本向左、向右或居中对齐。
- LV_PART_SCROLLBAR 文本内容过长时显示的滚动条。
- LV_PART_SELECTED 确定选定文本 (label) 的样式。只能使用 text_color 和 bg_color 样式属性。bg_color 会直接应用在文本框的标签上。
- LV_PART_CURSOR 字符插入位置的光标。光标的区域始终是当前字符的边界框。可以通过 anim_time 样式属性设置光标的闪烁时间。
- LV_PART_TEXTAREA_PLACEHOLDER 文本占位符，文本框独有的部分。可以通过这部分设置文本占位符的样式。

Usage（用法）

Add text（添加文本）

You can insert text or characters to the current cursor's position with:

- lv_textarea_add_char(textarea, 'c')
- lv_textarea_add_text(textarea, "insert this text")

To add wide characters like 'á', 'ß' or CJK characters use lv_textarea_add_text(ta, "á").

lv_textarea_set_text(ta, "New text") changes the whole text.

您可以使用以下命令在光标的当前位置插入文本或字符：

- lv_textarea_add_char(textarea, 'c')
- lv_textarea_add_text(textarea, " 插入此文本")

要添加宽字符，如 'á'、'ß' 或 CJK 字符，请使用 lv_textarea_add_text(ta, "á")。

lv_textarea_set_text(ta, "New text") 会改变 (清空再覆盖) 文本框中的所有内容。

Placeholder（占位符）

A placeholder text can be specified - which is displayed when the Text area is empty - with `lv_textarea_set_placeholder_text(ta, "Placeholder text")`

您可以通过 `lv_textarea_set_placeholder_text(ta, "Placeholder text")` 指定占位符文本，当文本框的内容为空时，所设置的占位符文本将会展示出来。

Delete character（删除字符）

To delete a character from the left of the current cursor position use `lv_textarea_del_char(textarea)`. To delete from the right use `lv_textarea_del_char_forward(textarea)`

要删除光标左侧的字符，请使用 `lv_textarea_del_char(textarea)` 。要从光标右侧删除字符，请使用 `lv_textarea_del_char_forward(textarea)`

Move the cursor（移动光标）

The cursor position can be modified directly like `lv_textarea_set_cursor_pos(textarea, 10)`. The `0` position means "before the first characters", `LV_TA_CURSOR_LAST` means "after the last character"

You can step the cursor with

- `lv_textarea_cursor_right(textarea)`
- `lv_textarea_cursor_left(textarea)`
- `lv_textarea_cursor_up(textarea)`
- `lv_textarea_cursor_down(textarea)`

If `lv_textarea_set_cursor_click_pos(textarea, true)` is applied the cursor will jump to the position where the Text area was clicked.

光标的位置可以使用函数 `lv_textarea_set_cursor_pos(textarea, 10)` 直接修改。第二个参数为你要指定的光标的位置，要注意的是：

- `0` 表示将光标移动到“第一个字符之前”，
- `LV_TEXTAREA_CURSOR_LAST` 表示将光标移动到“最后一个字符之后”

您可以使用下面这些接口函数修改光标位置：

- `lv_textarea_cursor_up(textarea)` 上
- `lv_textarea_cursor_down(textarea)` 下
- `lv_textarea_cursor_left(textarea)` 左
- `lv_textarea_cursor_right(textarea)` 右

如果设置了 `lv_textarea_set_cursor_click_pos(textarea, true)` 那光标会跟随触摸跳转位置(在文本框范围内)

Hide the cursor（隐藏光标）

The cursor is always visible, however it can be a good idea to style it to be visible only in `LV_STATE_FOCUSED` state.

一般情况下光标始终可见，但最好将其样式设置为仅在 `LV_STATE_FOCUSED` 状态下 (聚焦状态) 可见。

One line mode（单行模式）

The Text area can be configured to be on a single line with `lv_textarea_set_one_line(textarea, true)`. In this mode the height is set automatically to show only one line, line break characters are ignored, and word wrap is disabled.

可以使用 `lv_textarea_set_one_line(textarea, true)` 将文本框配置为单行输入模式。在这个模式下，高度自动设置为仅显示一行，忽略换行符，并禁用自动换行。

Password mode（密码模式）

The text area supports password mode which can be enabled with `lv_textarea_set_password_mode(textarea, true)`.

If the • (Bullet, U+2022) character exists in the font, the entered characters are converted to it after some time or when a new character is entered. If • not exists, * will be used.

In password mode `lv_textarea_get_text(textarea)` returns the actual text entered, not the bullet characters.

The visibility time can be adjusted with `LV_TEXTAREA_DEF_PWD_SHOW_TIME)` in `lv_conf.h`.

文本框支持密码模式，可以通过 `lv_textarea_set_password_mode(textarea, true)` 启用该模式。

如果字体中存在 • (Bullet, U+2022) 字符，则将输入的字符会在一定时间后或者输入新字符时自动转换为该字符。如果 • 不存在，* 将被使用。

在密码模式下 `lv_textarea_get_text(textarea)` 返回的是输入的实际文本，而不是 • 字符。

密码模式下，实际输入文本的可见时间可以通过 `lv_conf.h` 中的 `LV_TEXTAREA_DEF_PWD_SHOW_TIME` 进行调整。

Accepted characters（字符白名单）

You can set a list of accepted characters with `lv_textarae_set_accepted_chars(textarea, "0123456789.+-")`. Other characters will be ignored.

您可以使用 `lv_textarea_set_accepted_chars(textarea, "0123456789.+-")` 设置可接受字符列表 (白名单)。输入其他字符将被忽略。

Max text length（设置文本长度）

The maximum number of characters can be limited with `lv_textarea_set_max_length(textarea, max_char_num)`

可以使用 `lv_textarea_set_max_length(textarea, max_char_num)` 设置文本框可容纳的最大字符数

Very long texts（超长文本）

If there is a very long text in the Text area (e. g. > 20k characters), scrolling and drawing might be slow. However, by enabling `LV_LABEL_LONG_TXT_HINT 1` in `lv_conf.h` the performance can be hugely improved. This will save some additional information about the label to speed up its drawing. Using `LV_LABEL_LONG_TXT_HINT` the scrolling and drawing will as fast as with "normal" short texts.

如果文本框中有很长的文本 (例如 > 20k 个字符), 可能会导致滚动和绘制速度很慢。但是, 如果在 `lv_conf.h` 中启用 `LV_LABEL_LONG_TXT_HINT 1`, 那么这个问题可以得到很好的改善。这将会保存有关标签 (label) 的一些附加信息以加快其绘制速度。使用 `LV_LABEL_LONG_TXT_HINT` 后, 滚动和绘制速度将与 "普通" 的短文本一样快。

Select text（选择文本）

Any part of the text can be selected if enabled with `lv_textarea_set_text_selection(textarea, true)`. This works much like when you select text on your PC with your mouse.

如果使用函数 `lv_textarea_set_text_selection(textarea, true)` 启用文本选择功能, 则可以选择文本的任何部分。这和我们使用鼠标在电脑上选择文字非常相似。

Events（事件）

- `LV_EVENT_INSERT` Sent right before a character or text is inserted. The event paramter is the text about to be inserted. `lv_textarea_set_insert_replace(textarea, "New text")` replaces the text to insert. The new text cannot be in a local variable which is destroyed when the event callback exists. `""` means do not insert anything.
- `LV_EVENT_VALUE_CHANGED` Sent when the content of the text area has been changed.
- `LV_EVENT_APPLY` Sent when `LV_KEY_ENTER` is pressed (or(sent) to a one line text area.

See the events of the *Base object* too.

Learn more about *Events*.

- `LV_EVENT_INSERT` 在插入字符或文本之前发送该事件。事件参数是即将插入的文本。可以通过接口 `lv_textarea_set_insert_replace(textarea, "New text")` 将要计划插入的文本替换为其他文本。注意替换的新文本不能时局部变量，也就是不能在回调函数直接创建局部变量保存新文本。`""` 表示不插入任何内容。
- `LV_EVENT_VALUE_CHANGED` 当文本框的内容被改变时发送该事件。
- `LV_EVENT_APPLY` 在按下 `LV_KEY_ENTER` 时发送到单行文本框。

参见*Base object* 的事件。

详细了解事件。

Keys（按键）

- `LV_KEY_UP/DOWN/LEFT/RIGHT` Move the cursor
- `Any character` Add the character to the current cursor position

Learn more about *Keys*.

- `LV_KEY_UP/DOWN/LEFT/RIGHT` 移动光标
- 任意字符将字符添加到当前光标位置

了解有关按键 的更多信息。

Example

Simple Text area

```
#include "../../lv_examples.h"
#if LV_USE_TEXTAREA && LV_BUILD_EXAMPLES

static void textarea_event_handler(lv_event_t * e)
{
    lv_obj_t * ta = lv_event_get_target(e);
    LV_LOG_USER("Enter was pressed. The current text is: %s", lv_textarea_get_
↪text(ta));
}

static void btnm_event_handler(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target(e);
    lv_obj_t * ta = lv_event_get_user_data(e);
    const char * txt = lv_btnmatrix_get_btn_text(obj, lv_btnmatrix_get_selected_
↪btn(obj));

    if(strcmp(txt, LV_SYMBOL_BACKSPACE) == 0) lv_textarea_del_char(ta);
    else if(strcmp(txt, LV_SYMBOL_NEW_LINE) == 0) lv_event_send(ta, LV_EVENT_READY,␣
↪NULL);
    else lv_textarea_add_text(ta, txt);

}

void lv_example_textarea_1(void)
{
    lv_obj_t * ta = lv_textarea_create(lv_scr_act());
    lv_textarea_set_one_line(ta, true);
    lv_obj_align(ta, LV_ALIGN_TOP_MID, 0, 10);
    lv_obj_add_event_cb(ta, textarea_event_handler, LV_EVENT_READY, ta);
    lv_obj_add_state(ta, LV_STATE_FOCUSED); /*To be sure the cursor is visible*/

    static const char * btnm_map[] = {"1", "2", "3", "\n",
                                      "4", "5", "6", "\n",
                                      "7", "8", "9", "\n",
                                      LV_SYMBOL_BACKSPACE, "0", LV_SYMBOL_NEW_LINE, ""};

    lv_obj_t * btnm = lv_btnmatrix_create(lv_scr_act());
    lv_obj_set_size(btnm, 200, 150);
    lv_obj_align(btnm, LV_ALIGN_BOTTOM_MID, 0, -10);
```

(下页继续)

(续上页)

```
    lv_obj_add_event_cb(btnm, btnm_event_handler, LV_EVENT_VALUE_CHANGED, ta);
    lv_obj_clear_flag(btnm, LV_OBJ_FLAG_CLICK_FOCUSABLE); /*To keep the text area␣
↪focused on button clicks*/
    lv_btnmatrix_set_map(btnm, btnm_map);
}

#endif
```

```
def textarea_event_handler(e, ta):
    print("Enter was pressed. The current text is: " + ta.get_text())


def btnm_event_handler(e, ta):
    obj = e.get_target()
    txt = obj.get_btn_text(obj.get_selected_btn())
    if txt == lv.SYMBOL.BACKSPACE:
        ta.del_char()
    elif txt == lv.SYMBOL.NEW_LINE:
        lv.event_send(ta, lv.EVENT.READY, None)
    elif txt:
        ta.add_text(txt)


ta = lv.textarea(lv.scr_act())
ta.set_one_line(True)
ta.align(lv.ALIGN.TOP_MID, 0, 10)
ta.add_event_cb(lambda e: textarea_event_handler(e, ta), lv.EVENT.READY, None)
ta.add_state(lv.STATE.FOCUSED)   # To be sure the cursor is visible

btnm_map = ["1", "2", "3", "\n",
            "4", "5", "6", "\n",
            "7", "8", "9", "\n",
            lv.SYMBOL.BACKSPACE, "0", lv.SYMBOL.NEW_LINE, ""]

btnm = lv.btnmatrix(lv.scr_act())
btnm.set_size(200, 150)
btnm.align(lv.ALIGN.BOTTOM_MID, 0, -10)
btnm.add_event_cb(lambda e: btnm_event_handler(e, ta), lv.EVENT.VALUE_CHANGED, None)
btnm.clear_flag(lv.obj.FLAG.CLICK_FOCUSABLE)    # To keep the text area focused on␣
↪button clicks
btnm.set_map(btnm_map)
```

Text area with password field

```
#include "../../lv_examples.h"
#if LV_USE_TEXTAREA && LV_USE_KEYBOARD && LV_BUILD_EXAMPLES

static void ta_event_cb(lv_event_t * e);

static lv_obj_t * kb;

void lv_example_textarea_2(void)
{
    /*Create the password box*/
    lv_obj_t * pwd_ta = lv_textarea_create(lv_scr_act());
    lv_textarea_set_text(pwd_ta, "");
    lv_textarea_set_password_mode(pwd_ta, true);
    lv_textarea_set_one_line(pwd_ta, true);
    lv_obj_set_width(pwd_ta, lv_pct(40));
    lv_obj_set_pos(pwd_ta, 5, 20);
    lv_obj_add_event_cb(pwd_ta, ta_event_cb, LV_EVENT_ALL, NULL);

    /*Create a label and position it above the text box*/
    lv_obj_t * pwd_label = lv_label_create(lv_scr_act());
    lv_label_set_text(pwd_label, "Password:");
    lv_obj_align_to(pwd_label, pwd_ta, LV_ALIGN_OUT_TOP_LEFT, 0, 0);

    /*Create the one-line mode text area*/
    lv_obj_t * text_ta = lv_textarea_create(lv_scr_act());
    lv_textarea_set_one_line(text_ta, true);
    lv_textarea_set_password_mode(text_ta, false);
    lv_obj_set_width(text_ta, lv_pct(40));
    lv_obj_add_event_cb(text_ta, ta_event_cb, LV_EVENT_ALL, NULL);
    lv_obj_align(text_ta, LV_ALIGN_TOP_RIGHT, -5, 20);


    /*Create a label and position it above the text box*/
    lv_obj_t * oneline_label = lv_label_create(lv_scr_act());
    lv_label_set_text(oneline_label, "Text:");
    lv_obj_align_to(oneline_label, text_ta, LV_ALIGN_OUT_TOP_LEFT, 0, 0);

    /*Create a keyboard*/
    kb = lv_keyboard_create(lv_scr_act());
    lv_obj_set_size(kb,  LV_HOR_RES, LV_VER_RES / 2);

    lv_keyboard_set_textarea(kb, pwd_ta); /*Focus it on one of the text areas to␣
↪start*/
```

(下页继续)

(续上页)

```
}

static void ta_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * ta = lv_event_get_target(e);
    if(code == LV_EVENT_CLICKED || code == LV_EVENT_FOCUSED) {
        /*Focus on the clicked text area*/
        if(kb != NULL) lv_keyboard_set_textarea(kb, ta);
    }

    else if(code == LV_EVENT_READY) {
        LV_LOG_USER("Ready, current text: %s", lv_textarea_get_text(ta));
    }
}

#endif
```

```
def ta_event_cb(e):
    code = e.get_code()
    ta = e.get_target()
    if code == lv.EVENT.CLICKED or code == lv.EVENT.FOCUSED:
        # Focus on the clicked text area
        if kb != None:
            kb.set_textarea(ta)

    elif code == lv.EVENT.READY:
        print("Ready, current text: " + ta.get_text())


# Create the password box
LV_HOR_RES = lv.scr_act().get_disp().driver.hor_res
LV_VER_RES = lv.scr_act().get_disp().driver.ver_res

pwd_ta = lv.textarea(lv.scr_act())
pwd_ta.set_text("")
pwd_ta.set_password_mode(True)
pwd_ta.set_one_line(True)
pwd_ta.set_width(LV_HOR_RES // 2 - 20)
pwd_ta.set_pos(5, 20)
pwd_ta.add_event_cb(ta_event_cb, lv.EVENT.ALL, None)

# Create a label and position it above the text box
```

(下页继续)

(续上页)

```
pwd_label = lv.label(lv.scr_act())
pwd_label.set_text("Password:")
pwd_label.align_to(pwd_ta, lv.ALIGN.OUT_TOP_LEFT, 0, 0)

# Create the one-line mode text area
text_ta = lv.textarea(lv.scr_act())
text_ta.set_width(LV_HOR_RES // 2 - 20)
text_ta.set_one_line(True)
text_ta.add_event_cb(ta_event_cb, lv.EVENT.ALL, None)
text_ta.set_password_mode(False)

text_ta.align(lv.ALIGN.TOP_RIGHT, -5, 20)

# Create a label and position it above the text box
oneline_label = lv.label(lv.scr_act())
oneline_label.set_text("Text:")
oneline_label.align_to(text_ta, lv.ALIGN.OUT_TOP_LEFT, 0, 0)

# Create a keyboard
kb = lv.keyboard(lv.scr_act())
kb.set_size(LV_HOR_RES, LV_VER_RES // 2)

kb.set_textarea(pwd_ta)  # Focus it on one of the text areas to start
```

Text auto-formatting

```
#include "../../lv_examples.h"
#if LV_USE_TEXTAREA && LV_USE_KEYBOARD && LV_BUILD_EXAMPLES

static void ta_event_cb(lv_event_t * e);

static lv_obj_t * kb;

/**
 * Automatically format text like a clock. E.g. "12:34"
 * Add the ':' automatically.
 */
void lv_example_textarea_3(void)
{
    /*Create the text area*/
    lv_obj_t * ta = lv_textarea_create(lv_scr_act());
```

(下页继续)

(续上页)

```
    lv_obj_add_event_cb(ta, ta_event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    lv_textarea_set_accepted_chars(ta, "0123456789:");
    lv_textarea_set_max_length(ta, 5);
    lv_textarea_set_one_line(ta, true);
    lv_textarea_set_text(ta, "");

    /*Create a keyboard*/
    kb = lv_keyboard_create(lv_scr_act());
    lv_obj_set_size(kb,  LV_HOR_RES, LV_VER_RES / 2);
    lv_keyboard_set_mode(kb, LV_KEYBOARD_MODE_NUMBER);
    lv_keyboard_set_textarea(kb, ta);
}

static void ta_event_cb(lv_event_t * e)
{
    lv_obj_t * ta = lv_event_get_target(e);
    const char * txt = lv_textarea_get_text(ta);
    if(txt[0] >= '0' && txt[0] <= '9' &&
        txt[1] >= '0' && txt[1] <= '9' &&
        txt[2] != ':')
    {
        lv_textarea_set_cursor_pos(ta, 2);
        lv_textarea_add_char(ta, ':');
    }
}

#endif
```

```
def ta_event_cb(e):
    ta = e.get_target()
    txt = ta.get_text()
    # print(txt)
    pos = ta.get_cursor_pos()
    # print("cursor pos: ",pos)
    # find position of ":" in text
    colon_pos= txt.find(":")
    # if there are more than 2 digits before the colon, remove the last one entered
    if colon_pos == 3:
        ta.del_char()
    if colon_pos != -1:
        # if there are more than 3 digits after the ":" remove the last one entered
        rest = txt[colon_pos:]
        if len(rest) > 3:
```

(下页继续)

(续上页)

```
            ta.del_char()

    if len(txt) < 2:
        return
    if ":" in txt:
        return
    if  txt[0] >= '0' and txt[0] <= '9' and \
        txt[1] >= '0' and txt[1] <= '9':
        if len(txt) == 2 or txt[2] != ':' :
            ta.set_cursor_pos(2)
            ta.add_char(ord(':'))
#
# Automatically format text like a clock. E.g. "12:34"
# Add the ':' automatically
#
# Create the text area

LV_HOR_RES = lv.scr_act().get_disp().driver.hor_res
LV_VER_RES = lv.scr_act().get_disp().driver.ver_res

ta = lv.textarea(lv.scr_act())
ta.add_event_cb(ta_event_cb, lv.EVENT.VALUE_CHANGED, None)
ta.set_accepted_chars("0123456789:")
ta.set_max_length(5)
ta.set_one_line(True)
ta.set_text("")
ta.add_state(lv.STATE.FOCUSED)

# Create a keyboard
kb = lv.keyboard(lv.scr_act())
kb.set_size(LV_HOR_RES, LV_VER_RES // 2)
kb.set_mode(lv.keyboard.MODE.NUMBER)
kb.set_textarea(ta)
```

API

Enums

enum **[anonymous]**
Values:

enumerator **LV_PART_TEXTAREA_PLACEHOLDER**

Functions

LV_EXPORT_CONST_INT(LV_TEXTAREA_CURSOR_LAST)

lv_obj_t ***lv_textarea_create**(*lv_obj_t* *parent)
Create a text area object

参数 **parent** -- pointer to an object, it will be the parent of the new text area

返回 pointer to the created text area

void **lv_textarea_add_char**(*lv_obj_t* *obj, uint32_t c)
Insert a character to the current cursor position. To add a wide char, e.g. 'Á' use _lv_txt_encoded_conv_wc('Á')‘

参数

- **obj** -- pointer to a text area object
- **c** -- a character (e.g. 'a')

void **lv_textarea_add_text**(*lv_obj_t* *obj, const char *txt)
Insert a text to the current cursor position

参数

- **obj** -- pointer to a text area object
- **txt** -- a '\0' terminated string to insert

void **lv_textarea_del_char**(*lv_obj_t* *obj)
Delete a the left character from the current cursor position

参数 **obj** -- pointer to a text area object

void **lv_textarea_del_char_forward**(*lv_obj_t* *obj)
Delete the right character from the current cursor position

参数 **obj** -- pointer to a text area object

void **lv_textarea_set_text**(*lv_obj_t* *obj, const char *txt)
Set the text of a text area

参数

- **obj** -- pointer to a text area object
- **txt** -- pointer to the text

void **lv_textarea_set_placeholder_text**(*lv_obj_t* *obj, const char *txt)

Set the placeholder text of a text area

参数

- **obj** -- pointer to a text area object
- **txt** -- pointer to the text

void **lv_textarea_set_cursor_pos**(*lv_obj_t* *obj, int32_t pos)

Set the cursor position

参数

- **obj** -- pointer to a text area object
- **pos** -- the new cursor position in character index < 0 : index from the end of the text LV_TEXTAREA_CURSOR_LAST: go after the last character

void **lv_textarea_set_cursor_click_pos**(*lv_obj_t* *obj, bool en)

Enable/Disable the positioning of the cursor by clicking the text on the text area.

参数

- **obj** -- pointer to a text area object
- **en** -- true: enable click positions; false: disable

void **lv_textarea_set_password_mode**(*lv_obj_t* *obj, bool en)

Enable/Disable password mode

参数

- **obj** -- pointer to a text area object
- **en** -- true: enable, false: disable

void **lv_textarea_set_one_line**(*lv_obj_t* *obj, bool en)

Configure the text area to one line or back to normal

参数

- **obj** -- pointer to a text area object
- **en** -- true: one line, false: normal

void **lv_textarea_set_accepted_chars**(*lv_obj_t* *obj, const char *list)

Set a list of characters. Only these characters will be accepted by the text area

参数

- **obj** -- pointer to a text area object
- **list** -- list of characters. Only the pointer is saved. E.g. "+-.,0123456789"

void **lv_textarea_set_max_length**(*lv_obj_t* *obj, uint32_t num)

Set max length of a Text Area.

参数

- **obj** -- pointer to a text area object
- **num** -- the maximal number of characters can be added (`lv_textarea_set_text` ignores it)

void **lv_textarea_set_insert_replace**(*lv_obj_t* *obj, const char *txt)

In `LV_EVENT_INSERT` the text which planned to be inserted can be replaced by an other text. It can be used to add automatic formatting to the text area.

参数

- **obj** -- pointer to a text area object
- **txt** -- pointer to a new string to insert. If `""` no text will be added. The variable must be live after the `event_cb` exists. (Should be `global` or `static`)

void **lv_textarea_set_text_selection**(*lv_obj_t* *obj, bool en)

Enable/disable selection mode.

参数

- **obj** -- pointer to a text area object
- **en** -- true or false to enable/disable selection mode

void **lv_textarea_set_password_show_time**(*lv_obj_t* *obj, uint16_t time)

Set how long show the password before changing it to '*'

参数

- **obj** -- pointer to a text area object
- **time** -- show time in milliseconds. 0: hide immediately.

void **lv_textarea_set_align**(*lv_obj_t* *obj, lv_text_align_t align)

Deprecated: use the normal text_align style property instead Set the label's alignment. It sets where the label is aligned (in one line mode it can be smaller than the text area) and how the lines of the area align in case of multiline text area

参数

- **obj** -- pointer to a text area object
- **align** -- the align mode from ::lv_text_align_t

const char ***lv_textarea_get_text**(const *lv_obj_t* *obj)

Get the text of a text area. In password mode it gives the real text (not '*'s).

参数 **obj** -- pointer to a text area object

返回 pointer to the text

const char ***lv_textarea_get_placeholder_text**(*lv_obj_t* *obj)

Get the placeholder text of a text area

参数 **obj** -- pointer to a text area object

返回 pointer to the text

lv_obj_t ***lv_textarea_get_label**(const *lv_obj_t* *obj)

Get the label of a text area

参数 **obj** -- pointer to a text area object

返回 pointer to the label object

uint32_t **lv_textarea_get_cursor_pos**(const *lv_obj_t* *obj)

Get the current cursor position in character index

参数 **obj** -- pointer to a text area object

返回 the cursor position

bool **lv_textarea_get_cursor_click_pos**(*lv_obj_t* *obj)

Get whether the cursor click positioning is enabled or not.

参数 **obj** -- pointer to a text area object

返回 true: enable click positions; false: disable

bool **lv_textarea_get_password_mode**(const *lv_obj_t* *obj)

Get the password mode attribute

参数 **obj** -- pointer to a text area object

返回 true: password mode is enabled, false: disabled

bool **lv_textarea_get_one_line**(const *lv_obj_t* *obj)

Get the one line configuration attribute

参数 **obj** -- pointer to a text area object

返回 true: one line configuration is enabled, false: disabled

const char ***lv_textarea_get_accepted_chars**(*lv_obj_t* *obj)

Get a list of accepted characters.

参数 **obj** -- pointer to a text area object

返回 list of accented characters.

uint32_t **lv_textarea_get_max_length**(*lv_obj_t* *obj)

Get max length of a Text Area.

参数 **obj** -- pointer to a text area object

返回 the maximal number of characters to be add

bool **lv_textarea_text_is_selected**(const *lv_obj_t* *obj)

Find whether text is selected or not.

参数 **obj** -- pointer to a text area object

返回 whether text is selected or not

bool **lv_textarea_get_text_selection**(*lv_obj_t* *obj)

Find whether selection mode is enabled.

参数 **obj** -- pointer to a text area object

返回 true: selection mode is enabled, false: disabled

uint16_t **lv_textarea_get_password_show_time**(*lv_obj_t* *obj)

Set how long show the password before changing it to '*'

参数 **obj** -- pointer to a text area object

返回 show time in milliseconds. 0: hide immediately.

void **lv_textarea_clear_selection**(*lv_obj_t* *obj)

Clear the selection on the text area.

参数 **obj** -- pointer to a text area object

void **lv_textarea_cursor_right**(*lv_obj_t* *obj)

Move the cursor one character right

参数 **obj** -- pointer to a text area object

void **lv_textarea_cursor_left**(*lv_obj_t* *obj)

Move the cursor one character left

参数 **obj** -- pointer to a text area object

void **lv_textarea_cursor_down**(*lv_obj_t* *obj)

Move the cursor one line down

参数 **obj** -- pointer to a text area object

void **lv_textarea_cursor_up**(*lv_obj_t* *obj)

Move the cursor one line up

参数 **obj** -- pointer to a text area object

Variables

const lv_obj_class_t **lv_textarea_class**

struct **lv_textarea_t**

Public Members

lv_obj_t **obj**

lv_obj_t ***label**

char ***placeholder_txt**

char ***pwd_tmp**

const char ***accepted_chars**

uint32_t **max_length**

uint16_t **pwd_show_time**

lv_coord_t **valid_x**

uint32_t **pos**

lv_area_t **area**

uint32_t **txt_byte_pos**

uint8_t **show**

uint8_t **click_pos**

struct *lv_textarea_t*::[anonymous] **cursor**

uint32_t **sel_start**

uint32_t **sel_end**

uint8_t **text_sel_in_prog**

uint8_t **text_sel_en**

uint8_t **pwd_mode**

uint8_t **one_line**

2.6.3 Extra widgets

Animation Image (lv_animimg)

Overview

The animation image is similar to the normal 'Image' object. The only difference is that instead of one source image, you set an array of multiple source images.

You can specify a duration and repeat count.

Parts and Styles

- `LV_PART_MAIN` A background rectangle that uses the typical background style properties and the image itself using the image style properties.

Usage

Image sources

To set the image in a state, use the `lv_animimg_set_src(imgbtn, dsc[], num)`.

Events

No special events are sent by image objects.

See the events of the Base object too.

Learn more about *Events*.

Keys

No Keys are processed by the object type.

Learn more about *Keys*.

Example

Simple Animation Image

```
#include "../../lv_examples.h"
#if LV_USE_ANIMIMG && LV_BUILD_EXAMPLES
LV_IMG_DECLARE(animimg001)
LV_IMG_DECLARE(animimg002)
LV_IMG_DECLARE(animimg003)

static const lv_img_dsc_t* anim_imgs[3] = {
    &animimg001,
    &animimg002,
    &animimg003,
};

void lv_example_animimg_1(void)
{
    lv_obj_t * animimg0 = lv_animimg_create(lv_scr_act());
    lv_obj_center(animimg0);
    lv_animimg_set_src(animimg0, (lv_img_dsc_t**) anim_imgs, 3);
    lv_animimg_set_duration(animimg0, 1000);
    lv_animimg_set_repeat_count(animimg0, LV_ANIM_REPEAT_INFINITE);
    lv_animimg_start(animimg0);
}

#endif
```

```
from imagetools import get_png_info, open_png

# Register PNG image decoder
decoder = lv.img.decoder_create()
decoder.info_cb = get_png_info
decoder.open_cb = open_png

anim_imgs = [None]*3
# Create an image from the png file
try:
    with open('../../assets/animimg001.png','rb') as f:
        anim001_data = f.read()
except:
    print("Could not find animimg001.png")
    sys.exit()
```

(下页继续)

(续上页)

```
anim_imgs[0] = lv.img_dsc_t({
  'data_size': len(anim001_data),
  'data': anim001_data
})

try:
    with open('../../assets/animimg002.png','rb') as f:
        anim002_data = f.read()
except:
    print("Could not find animimg002.png")
    sys.exit()

anim_imgs[1] = lv.img_dsc_t({
  'data_size': len(anim002_data),
  'data': anim002_data
})

try:
    with open('../../assets/animimg003.png','rb') as f:
        anim003_data = f.read()
except:
    print("Could not find animimg003.png")
    sys.exit()

anim_imgs[2] = lv.img_dsc_t({
  'data_size': len(anim003_data),
  'data': anim003_data
})

animimg0 = lv.animimg(lv.scr_act())
animimg0.center()
animimg0.set_src(anim_imgs, 3)
animimg0.set_duration(1000)
animimg0.set_repeat_count(lv.ANIM_REPEAT.INFINITE)
animimg0.start()
```

API

Typedefs

typedef uint8_t **lv_animimg_part_t**

Enums

enum **[anonymous]**
Values:

enumerator **LV_ANIM_IMG_PART_MAIN**

Functions

lv_obj_t ***lv_animimg_create**(*lv_obj_t* *parent)
Create an animation image objects

参数 **parent** -- pointer to an object, it will be the parent of the new button

返回 pointer to the created animation image object

void **lv_animimg_set_src**(*lv_obj_t* *img, *lv_img_dsc_t* *dsc[], uint8_t num)
Set the image animation images source.

参数

- **img** -- pointer to an animation image object
- **dsc** -- pointer to a series images
- **num** -- images' number

void **lv_animimg_start**(*lv_obj_t* *obj)
Startup the image animation.

参数 **obj** -- pointer to an animation image object

void **lv_animimg_set_duration**(*lv_obj_t* *img, uint32_t duration)
Set the image animation duration time. unit:ms

参数 **img** -- pointer to an animation image object

void **lv_animimg_set_repeat_count**(*lv_obj_t* *img, uint16_t count)
Set the image animation reapeatly play times.

参数

- **img** -- pointer to an animation image object

- **count** -- the number of times to repeat the animation

Variables

const lv_obj_class_t **lv_animimg_class**

struct **lv_animimg_t**

Public Members

lv_img_t **img**

lv_anim_t **anim**

lv_img_dsc_t ****dsc**

int8_t **pic_count**

Calendar (lv_calendar)

Overview

The Calendar object is a classic calendar which can:

- show the days of any month in a 7x7 matrix
- Show the name of the days
- highlight the current day (today)
- highlight any user-defined dates

The Calendar is added to the default group (if it is set). Calendar is an editable object which allow selecting and clicking the dates with encoder navigation too.

To make the Calendar flexible, by default it doesn't show the current year or month. Instead, there are optional "headers" that can be attached to the calendar.

Parts and Styles

The calendar object uses the *Button matrix* object under the hood to arrange the days into a matrix.

- `LV_PART_MAIN` The background of the calendar. Uses all the background related style properties.
- `LV_PART_ITEMS` Refers to the dates and day names. Button matrix control flags are set to differentiate the buttons and a custom drawer event is added modify the properties of the buttons as follows:

- day names have no border, no background and drawn with a gray color
- days of the previous and next month have `LV_BTNMATRIX_CTRL_DISABLED` flag
- today has a thicker border with the theme's primary color
- highlighted days have some opacity with the theme's primary color.

Usage

Some functions use the `lv_calendar_date_t` type which is a structure with `year`, `month` and `day` fields.

Current date

To set the current date (today), use the `lv_calendar_set_today_date(calendar, year, month, day)` function. `month` needs to be in 1..12 range and `day` in 1..31 range.

Shown date

To set the shown date, use `lv_calendar_set_shown_date(calendar, year, month);`

Highlighted days

The list of highlighted dates should be stored in a `lv_calendar_date_t` array loaded by `lv_calendar_set_highlighted_dates(calendar, highlighted_dates, date_num)`. Only the array's pointer will be saved so the array should be a static or global variable.

Name of the days

The name of the days can be adjusted with `lv_calendar_set_day_names(calendar, day_names)` where `day_names` looks like `const char * day_names[7] = {"Su", "Mo", ...};` Only the pointer of the day names is saved so the elements should be static, global or constant variables.

Events

- `LV_EVENT_VALUE_CHANGED` Sent if a date is clicked. `lv_calendar_get_pressed_date(calendar, &date)` set `date` to the date currently being pressed. Returns `LV_RES_OK` if there is a valid pressed date, else `LV_RES_INV`.

Learn more about *Events*.

Keys

- `LV_KEY_RIGHT/UP/LEFT/RIGHT` To navigate among the buttons to dates
- `LV_KEY_ENTER` To press/release the selected date

Learn more about *Keys*.

Headers

From v8.1 the header is added directly into the Calendar widget and the API of the headers has been changed.

Arrow buttons

`lv_calendar_header_arrow_create(calendar)` creates a header that contains a left and right arrow on the sides and a text with the current year and month between them.

Drop-down

`lv_calendar_header_dropdown_create(calendar)` creates a header that contains 2 drop-drown lists: one for the year and another for the month.

Example

Calendar with header

```
#include "../../lv_examples.h"
#if LV_USE_CALENDAR && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_current_target(e);

    if(code == LV_EVENT_VALUE_CHANGED) {
        lv_calendar_date_t date;
        if(lv_calendar_get_pressed_date(obj, &date)) {
            LV_LOG_USER("Clicked date: %02d.%02d.%d", date.day, date.month, date.
↪year);
        }
    }
}
```

(下页继续)

(续上页)

```
void lv_example_calendar_1(void)
{
    lv_obj_t  * calendar = lv_calendar_create(lv_scr_act());
    lv_obj_set_size(calendar, 185, 185);
    lv_obj_align(calendar, LV_ALIGN_CENTER, 0, 27);
    lv_obj_add_event_cb(calendar, event_handler, LV_EVENT_ALL, NULL);

    lv_calendar_set_today_date(calendar, 2021, 02, 23);
    lv_calendar_set_showed_date(calendar, 2021, 02);

    /*Highlight a few days*/
    static lv_calendar_date_t highlighted_days[3];       /*Only its pointer will be␣
↪saved so should be static*/
    highlighted_days[0].year = 2021;
    highlighted_days[0].month = 02;
    highlighted_days[0].day = 6;

    highlighted_days[1].year = 2021;
    highlighted_days[1].month = 02;
    highlighted_days[1].day = 11;

    highlighted_days[2].year = 2022;
    highlighted_days[2].month = 02;
    highlighted_days[2].day = 22;

    lv_calendar_set_highlighted_dates(calendar, highlighted_days, 3);

#if LV_USE_CALENDAR_HEADER_DROPDOWN
    lv_calendar_header_dropdown_create(calendar);
#elif LV_USE_CALENDAR_HEADER_ARROW
    lv_calendar_header_arrow_create(calendar);
#endif
    lv_calendar_set_showed_date(calendar, 2021, 10);
}

#endif
```

```
def event_handler(evt):
    code = evt.get_code()

    if code == lv.EVENT.VALUE_CHANGED:
```

(下页继续)

(续上页)

```
        source = evt.get_current_target()
        date = lv.calendar_date_t()
        if source.get_pressed_date(date) == lv.RES.OK:
            calendar.set_today_date(date.year, date.month, date.day)
            print("Clicked date: %02d.%02d.%02d"%(date.day, date.month, date.year))


calendar = lv.calendar(lv.scr_act())
calendar.set_size(200, 200)
calendar.align(lv.ALIGN.CENTER, 0, 20)
calendar.add_event_cb(event_handler, lv.EVENT.ALL, None)

calendar.set_today_date(2021, 02, 23)
calendar.set_showed_date(2021, 02)

# Highlight a few days
highlighted_days=[
    lv.calendar_date_t({'year':2021, 'month':2, 'day':6}),
    lv.calendar_date_t({'year':2021, 'month':2, 'day':11}),
    lv.calendar_date_t({'year':2021, 'month':2, 'day':22})
]

calendar.set_highlighted_dates(highlighted_days, len(highlighted_days))

lv.calendar_header_dropdown(calendar)
```

API

Functions

lv_obj_t ***lv_calendar_create**(*lv_obj_t* *parent)

void **lv_calendar_set_today_date**(*lv_obj_t* *obj, uint32_t year, uint32_t month, uint32_t day)

Set the today's date

参数

- **obj** -- pointer to a calendar object
- **year** -- today's year
- **month** -- today's month [1..12]
- **day** -- today's day [1..31]

void **lv_calendar_set_showed_date**(*lv_obj_t* *obj, uint32_t year, uint32_t month)

Set the currently showed

参数

- **obj** -- pointer to a calendar object
- **year** -- today's year
- **month** -- today's month [1..12]

void **lv_calendar_set_highlighted_dates**(*lv_obj_t* *obj, *lv_calendar_date_t* highlighted[], uint16_t date_num)

Set the highlighted dates

参数

- **obj** -- pointer to a calendar object
- **highlighted** -- pointer to an *lv_calendar_date_t* array containing the dates. Only the pointer will be saved so this variable can't be local which will be destroyed later.
- **date_num** -- number of dates in the array

void **lv_calendar_set_day_names**(*lv_obj_t* *obj, const char **day_names)

Set the name of the days

参数

- **obj** -- pointer to a calendar object
- **day_names** -- pointer to an array with the names. E.g. `const char * days[7] = {"Sun", "Mon", ...}` Only the pointer will be saved so this variable can't be local which will be destroyed later.

lv_obj_t ***lv_calendar_get_btnmatrix**(const *lv_obj_t* *obj)

Get the button matrix object of the calendar. It shows the dates and day names.

参数 **obj** -- pointer to a calendar object

返回 pointer to a the button matrix

const *lv_calendar_date_t* ***lv_calendar_get_today_date**(const *lv_obj_t* *calendar)

Get the today's date

参数 **calendar** -- pointer to a calendar object

返回 return pointer to an *lv_calendar_date_t* variable containing the date of today.

const *lv_calendar_date_t* ***lv_calendar_get_showed_date**(const *lv_obj_t* *calendar)

Get the currently showed

参数 **calendar** -- pointer to a calendar object

返回 pointer to an *lv_calendar_date_t* variable containing the date is being shown.

lv_calendar_date_t ***lv_calendar_get_highlighted_dates**(const *lv_obj_t* *calendar)

Get the highlighted dates

参数 **calendar** -- pointer to a calendar object

返回 pointer to an *lv_calendar_date_t* array containing the dates.

uint16_t **lv_calendar_get_highlighted_dates_num**(const *lv_obj_t* *calendar)

Get the number of the highlighted dates

参数 **calendar** -- pointer to a calendar object

返回 number of highlighted days

lv_res_t **lv_calendar_get_pressed_date**(const *lv_obj_t* *calendar, *lv_calendar_date_t* *date)

Get the currently pressed day

参数

- **calendar** -- pointer to a calendar object
- **date** -- store the pressed date here

返回 LV_RES_OK: there is a valid pressed date; LV_RES_INV: there is no pressed data

Variables

const lv_obj_class_t **lv_calendar_class**

struct **lv_calendar_date_t**

#include <lv_calendar.h> Represents a date on the calendar object (platform-agnostic).

Public Members

uint16_t **year**

int8_t **month**

int8_t **day**

1..12

struct **lv_calendar_t**

Public Members

lv_obj_t **obj**

lv_obj_t ***btnm**

lv_calendar_date_t **today**

lv_calendar_date_t **showed_date**

lv_calendar_date_t ***highlighted_dates**

uint16_t **highlighted_dates_num**

const char ***map**[8 * 7]

char **nums**[7 * 6][4]

Chart (lv_chart)

Overview

Charts are a basic object to visualize data points. Currently *Line* charts (connect points with lines and/or draw points on them) and *Bar* charts are supported.

Charts can have:

- division lines
- 2 y axis
- axis ticks and texts on ticks
- cursors
- scrolling and zooming

Parts and Styles

- `LV_PART_MAIN` The background of the chart. Uses all the typical background and *line* (for the division lines) related style properties. *Padding* makes the series area smaller.
- `LV_PART_SCROLLBAR` The scrollbar used if the chart is zoomed. See the *Base object*'s documentation for details.
- `LV_PART_ITEMS` Refers to the line or bar series.
  - Line chart: The *line* properties are used by the lines. `width`, `height`, `bg_color` and `radius` is used to set the appearance of points.
  - Bar chart: The typical background properties are used to style the bars.

- `LV_PART_INDICATOR` Refers to the points on line and scatter chart (small circles or squares).
- `LV_PART_CURSOR` *Line* properties are used to style the cursors. `width`, `height`, `bg_color` and `radius` are used to set the appearance of points.
- `LV_PART_TICKS` *Line* and *Text* style properties are used to style the ticks

Usage

Chart type

The following data display types exist:

- `LV_CHART_TYPE_NONE` Do not display any data. Can be used to hide the series.
- `LV_CHART_TYPE_LINE` Draw lines between the data points and/or points (rectangles or circles) on the data points.
- `LV_CHART_TYPE_BAR` - Draw bars.
- `LV_CHART_TYPE_SCATTER` - X/Y chart drawing point's and lines between the points. .

You can specify the display type with `lv_chart_set_type(chart, LV_CHART_TYPE_...)`.

Data series

You can add any number of series to the charts by `lv_chart_add_series(chart, color, axis)`. This allocates an `lv_chart_series_t` structure which contains the chosen `color` and an array for the data points. `axis` can have the following values:

- `LV_CHART_AXIS_PRIMARY_Y` Left axis
- `LV_CHART_AXIS_SECONDARY_Y` Right axis
- `LV_CHART_AXIS_PRIMARY_X` Bottom axis
- `LV_CHART_AXIS_SECONDARY_X` Top axis

`axis` tells which axis's range should be used te scale the values.

`lv_chart_set_ext_y_array(chart, ser, value_array)` makes the chart use an external array for the given series. `value_array` should look like this: `lv_coord_t * value_array[num_points]`. The array size needs to be large enough to hold all the points of that series. The array's pointer will be saved in the chart so it needs to be global, static or dynamically allocated. Note: you should call `lv_chart_refresh(chart)` after the external data source has been updated to update the chart.

The value array of a series can be obtained with `lv_chart_get_y_array(chart, ser)`, which can be used with `ext_array` or *normal array*s.

For `LV_CHART_TYPE_SCATTER` type `lv_chart_set_ext_x_array(chart, ser, value_array)` and `lv_chart_get_x_array(chart, ser)` can be used as well.

Modify the data

You have several options to set the data of series:

1. Set the values manually in the array like `ser1->points[3] = 7` and refresh the chart with `lv_chart_refresh(chart)`.
2. Use `lv_chart_set_value_by_id(chart, ser, id, value)` where `id` is the index of the point you wish to update.
3. Use the `lv_chart_set_next_value(chart, ser, value)`.
4. Initialize all points to a given value with: `lv_chart_set_all_value(chart, ser, value)`.

Use `LV_CHART_POINT_NONE` as value to make the library skip drawing that point, column, or line segment.

For `LV_CHART_TYPE_SCATTER` type `lv_chart_set_value_by_id2(chart, ser, id, value)` and `lv_chart_set_next_value2(chart, ser, x_valuem y_value)` can be used as well.

Update modes

`lv_chart_set_next_value` can behave in two ways depending on *update mode*:

- `LV_CHART_UPDATE_MODE_SHIFT` Shift old data to the left and add the new one to the right.
- `LV_CHART_UPDATE_MODE_CIRCULAR` - Add the new data in circular fashion, like an ECG diagram.

The update mode can be changed with `lv_chart_set_update_mode(chart, LV_CHART_UPDATE_MODE_...)`.

Number of points

The number of points in the series can be modified by `lv_chart_set_point_count(chart, point_num)`. The default value is 10. Note: this also affects the number of points processed when an external buffer is assigned to a series, so you need to be sure the external array is large enough.

Handling large number of points

On line charts, if the number of points is greater than the pixels horizontally, the Chart will draw only vertical lines to make the drawing of large amount of data effective. If there are, let's say, 10 points to a pixel, LVGL searches the smallest and the largest value and draws a vertical lines between them to ensure no peaks are missed.

Vertical range

You can specify the minimum and maximum values in y-direction with `lv_chart_set_range(chart, axis, min, max)`. `axis` can be `LV_CHART_AXIS_PRIMARY` (left axis) or `LV_CHART_AXIS_SECONDARY` (right axis).

The value of the points will be scaled proportionally. The default range is: 0..100.

Division lines

The number of horizontal and vertical division lines can be modified by `lv_chart_set_div_line_count(chart, hdiv_num, vdiv_num)`. The default settings are 3 horizontal and 5 vertical division lines. If there is a visible border on a side and no padding on that side, the division line would be drawn on top of the border and therefore it won't be drawn.

Override default start point for series

If you want a plot to start from a point other than the default which is `point[0]` of the series, you can set an alternative index with the function `lv_chart_set_x_start_point(chart, ser, id)` where `id` is the new index position to start plotting from.

Note that `LV_CHART_UPDATE_MODE_SHIFT` also changes the `start_point`.

Tick marks and labels

Ticks and labels can be added to the axis with `lv_chart_set_axis_tick(chart, axis, major_len, minor_len, major_cnt, minor_cnt, label_en, draw_size)`.

- `axis` can be `LV_CHART_AXIS_X/PRIMARY_Y/SECONDARY_Y`
- `major_len` is the length of major ticks
- `minor_len` is the length of minor ticks
- `major_cnt` is the number of major ticks on the axis
- `minor_cnt` in the number of minor ticks between two major ticks
- `label_en` `true`: enable label drawing on major ticks

- `draw_size` extra size required to draw the tick and labels (start with 20 px and increase if the ticks/labels are clipped)

Zoom

The chart can be zoomed independently in x and y directions with `lv_chart_set_zoom_x(chart, factor)` and `lv_chart_set_zoom_y(chart, factor)`. If `factor` is 256 there is no zoom. 512 means double zoom, etc. Fractional values are also possible but < 256 value is not allowed.

Cursor

A cursor can be added with `lv_chart_cursor_t * c1 = lv_chart_add_cursor(chart, color, dir);`. The possible values of `dir` `LV_DIR_NONE/RIGHT/UP/LEFT/DOWN/HOR/VER/ALL` or their OR-ed values to tell in which direction(s) should the cursor be drawn.

`lv_chart_set_cursor_pos(chart, cursor, &point)` sets the position of the cursor. `pos` is a pointer to an `lv_point_t` variable. E.g. `lv_point_t point = {10, 20};`. If the chart is scrolled the cursor will remain in the same place.

`lv_chart_get_point_pos_by_id(chart, series, id, &point_out)` gets the coordinate of a given point. It's useful to place the cursor at a given point.

`lv_chart_set_cursor_point(chart, cursor, series, point_id)` sticks the cursor at a point. If the point's position changes (new value or scrolling) the cursor will move with the point.

Events

- `LV_EVENT_VALUE_CHANGED` Sent when a new point is clicked pressed. `lv_chart_get_pressed_point(chart)` returns the zero-based index of the pressed point.
- `LV_EVENT_DRAW_PART_BEGIN` and `LV_EVENT_DRAW_PART_END` are sent with the following types:
  - `LV_CHART_DRAW_PART_DIV_LINE_INIT` Used before/after drawn the div lines to add masks to any extra drawings. The following fields are set:
    - `part`: `LV_PART_MAIN`
    - `line_dsc`
  - `LV_CHART_DRAW_PART_DIV_LINE_HOR`, `LV_CHART_DRAW_PART_DIV_LINE_VER` Used for each horizontal and vertical division lines.
    - `part`: `LV_PART_MAIN`
    - `id`: index of the line
    - `p1`, `p2`: points of the line

    * line_dsc

- LV_CHART_DRAW_PART_LINE_AND_POINT Used on line and scatter charts for lines and points.
    * part: LV_PART_ITEMS
    * id: index of the point
    * value: value of idth point
    * p1, p2: points of the line
    * draw_area: area of the point
    * line_dsc
    * rect_dsc
    * sub_part_ptr: pointer to the series
- LV_CHART_DRAW_PART_BAR Used on bar charts for the rectangles.
    * part: LV_PART_ITEMS
    * id: index of the point
    * value: value of idth point
    * draw_area: area of the point
    * rect_dsc:
    * sub_part_ptr: pointer to the series
- LV_CHART_DRAW_PART_CURSOR Used on cursor lines and points.
    * part: LV_PART_CURSOR
    * p1, p2: points of the line
    * line_dsc
    * rect_dsc
    * draw_area: area of the points
- LV_CHART_DRAW_PART_TICK_LABEL Used on tick lines and labels.
    * part: LV_PART_TICKS
    * id: axis
    * value: value of the tick
    * text: value converted to decimal or NULL for minor ticks
    * line_dsc,
    * label_dsc,

See the events of the *Base object* too.

Learn more about *Events*.

Keys

No *Keys* are processed by the object type.

Learn more about *Keys*.

Example

Line Chart

```
#include "../../lv_examples.h"
#if LV_USE_CHART && LV_BUILD_EXAMPLES

void lv_example_chart_1(void)
{
    /*Create a chart*/
    lv_obj_t * chart;
    chart = lv_chart_create(lv_scr_act());
    lv_obj_set_size(chart, 200, 150);
    lv_obj_center(chart);
    lv_chart_set_type(chart, LV_CHART_TYPE_LINE);   /*Show lines and points too*/

    /*Add two data series*/
    lv_chart_series_t * ser1 = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_
↪RED), LV_CHART_AXIS_PRIMARY_Y);
    lv_chart_series_t * ser2 = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_
↪GREEN), LV_CHART_AXIS_SECONDARY_Y);

    /*Set the next points on 'ser1'*/
    lv_chart_set_next_value(chart, ser1, 10);
    lv_chart_set_next_value(chart, ser1, 10);
    lv_chart_set_next_value(chart, ser1, 10);
    lv_chart_set_next_value(chart, ser1, 10);
    lv_chart_set_next_value(chart, ser1, 10);
    lv_chart_set_next_value(chart, ser1, 10);
    lv_chart_set_next_value(chart, ser1, 10);
    lv_chart_set_next_value(chart, ser1, 30);
    lv_chart_set_next_value(chart, ser1, 70);
    lv_chart_set_next_value(chart, ser1, 90);
```

(下页继续)

(续上页)

```
    /*Directly set points on 'ser2'*/
    ser2->y_points[0] = 90;
    ser2->y_points[1] = 70;
    ser2->y_points[2] = 65;
    ser2->y_points[3] = 65;
    ser2->y_points[4] = 65;
    ser2->y_points[5] = 65;
    ser2->y_points[6] = 65;
    ser2->y_points[7] = 65;
    ser2->y_points[8] = 65;
    ser2->y_points[9] = 65;

    lv_chart_refresh(chart); /*Required after direct set*/
}

#endif
```

```
# Create a chart
chart = lv.chart(lv.scr_act())
chart.set_size(200, 150)
chart.center()
chart.set_type(lv.chart.TYPE.LINE)   # Show lines and points too

# Add two data series
ser1 = chart.add_series(lv.palette_main(lv.PALETTE.RED), lv.chart.AXIS.PRIMARY_Y)
ser2 = chart.add_series(lv.palette_main(lv.PALETTE.GREEN), lv.chart.AXIS.SECONDARY_Y)
print(ser2)
# Set next points on ser1
chart.set_next_value(ser1,10)
chart.set_next_value(ser1,10)
chart.set_next_value(ser1,10)
chart.set_next_value(ser1,10)
chart.set_next_value(ser1,10)
chart.set_next_value(ser1,10)
chart.set_next_value(ser1,10)
chart.set_next_value(ser1,30)
chart.set_next_value(ser1,70)
chart.set_next_value(ser1,90)

# Directly set points on 'ser2'
ser2.y_points = [90, 70, 65, 65, 65, 65, 65, 65, 65, 65]
chart.refresh()      #  Required after direct set
```

Faded area line chart with custom division lines

```
#include "../../lv_examples.h"
#if LV_USE_CHART && LV_DRAW_COMPLEX && LV_BUILD_EXAMPLES

static lv_obj_t * chart1;
static lv_chart_series_t * ser1;
static lv_chart_series_t * ser2;

static void draw_event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target(e);

    /*Add the faded area before the lines are drawn*/
    lv_obj_draw_part_dsc_t * dsc = lv_event_get_draw_part_dsc(e);
    if(dsc->part == LV_PART_ITEMS) {
        if(!dsc->p1 || !dsc->p2) return;

        /*Add a line mask that keeps the area below the line*/
        lv_draw_mask_line_param_t line_mask_param;
        lv_draw_mask_line_points_init(&line_mask_param, dsc->p1->x, dsc->p1->y, dsc->
↪p2->x, dsc->p2->y, LV_DRAW_MASK_LINE_SIDE_BOTTOM);
        int16_t line_mask_id = lv_draw_mask_add(&line_mask_param, NULL);

        /*Add a fade effect: transparent bottom covering top*/
        lv_coord_t h = lv_obj_get_height(obj);
        lv_draw_mask_fade_param_t fade_mask_param;
        lv_draw_mask_fade_init(&fade_mask_param, &obj->coords, LV_OPA_COVER, obj->
↪coords.y1 + h / 8, LV_OPA_TRANSP,obj->coords.y2);
        int16_t fade_mask_id = lv_draw_mask_add(&fade_mask_param, NULL);

        /*Draw a rectangle that will be affected by the mask*/
        lv_draw_rect_dsc_t draw_rect_dsc;
        lv_draw_rect_dsc_init(&draw_rect_dsc);
        draw_rect_dsc.bg_opa = LV_OPA_20;
        draw_rect_dsc.bg_color = dsc->line_dsc->color;

        lv_area_t a;
        a.x1 = dsc->p1->x;
        a.x2 = dsc->p2->x - 1;
        a.y1 = LV_MIN(dsc->p1->y, dsc->p2->y);
        a.y2 = obj->coords.y2;
        lv_draw_rect(dsc->draw_ctx, &draw_rect_dsc, &a);
```

(下页继续)

(续上页)

```
        /*Remove the masks*/
        lv_draw_mask_free_param(&line_mask_param);
        lv_draw_mask_free_param(&fade_mask_param);
        lv_draw_mask_remove_id(line_mask_id);
        lv_draw_mask_remove_id(fade_mask_id);
    }
    /*Hook the division lines too*/
    else if(dsc->part == LV_PART_MAIN) {
        if(dsc->line_dsc == NULL || dsc->p1 == NULL || dsc->p2 == NULL) return;

        /*Vertical line*/
        if(dsc->p1->x == dsc->p2->x) {
            dsc->line_dsc->color  = lv_palette_lighten(LV_PALETTE_GREY, 1);
            if(dsc->id == 3) {
                dsc->line_dsc->width  = 2;
                dsc->line_dsc->dash_gap  = 0;
                dsc->line_dsc->dash_width  = 0;
            }
            else {
                dsc->line_dsc->width = 1;
                dsc->line_dsc->dash_gap  = 6;
                dsc->line_dsc->dash_width  = 6;
            }
        }
        /*Horizontal line*/
        else {
            if(dsc->id == 2) {
                dsc->line_dsc->width  = 2;
                dsc->line_dsc->dash_gap  = 0;
                dsc->line_dsc->dash_width  = 0;
            }
            else {
                dsc->line_dsc->width = 2;
                dsc->line_dsc->dash_gap  = 6;
                dsc->line_dsc->dash_width  = 6;
            }

            if(dsc->id == 1  || dsc->id == 3) {
                dsc->line_dsc->color  = lv_palette_main(LV_PALETTE_GREEN);
            } else {
                dsc->line_dsc->color  = lv_palette_lighten(LV_PALETTE_GREY, 1);
            }
        }
```

(下页继续)

(续上页)

```
    }
}

static void add_data(lv_timer_t * timer)
{
    LV_UNUSED(timer);
    static uint32_t cnt = 0;
    lv_chart_set_next_value(chart1, ser1, lv_rand(20, 90));

    if(cnt % 4 == 0) lv_chart_set_next_value(chart1, ser2, lv_rand(40, 60));

    cnt++;
}

/**
 * Add a faded area effect to the line chart and make some division lines ticker
 */
void lv_example_chart_2(void)
{
    /*Create a chart1*/
    chart1 = lv_chart_create(lv_scr_act());
    lv_obj_set_size(chart1, 200, 150);
    lv_obj_center(chart1);
    lv_chart_set_type(chart1, LV_CHART_TYPE_LINE);   /*Show lines and points too*/

    lv_chart_set_div_line_count(chart1, 5, 7);

    lv_obj_add_event_cb(chart1, draw_event_cb, LV_EVENT_DRAW_PART_BEGIN, NULL);
    lv_chart_set_update_mode(chart1, LV_CHART_UPDATE_MODE_CIRCULAR);

    /*Add two data series*/
    ser1 = lv_chart_add_series(chart1, lv_palette_main(LV_PALETTE_RED), LV_CHART_AXIS_
↪PRIMARY_Y);
    ser2 = lv_chart_add_series(chart1, lv_palette_main(LV_PALETTE_BLUE), LV_CHART_
↪AXIS_SECONDARY_Y);

    uint32_t i;
    for(i = 0; i < 10; i++) {
        lv_chart_set_next_value(chart1, ser1, lv_rand(20, 90));
        lv_chart_set_next_value(chart1, ser2, lv_rand(30, 70));
    }

    lv_timer_create(add_data, 200, NULL);
```

(下页继续)

(续上页)

```
}

#endif
```

```
def draw_event_cb(e):

    obj = e.get_target()

    # Add the faded area before the lines are drawn
    dsc = lv.obj_draw_part_dsc_t.__cast__(e.get_param())
    if dsc.part != lv.PART.ITEMS:
        return
    if not dsc.p1 or not dsc.p2:
        return

    # Add a line mask that keeps the area below the line
    line_mask_param = lv.draw_mask_line_param_t()
    line_mask_param.points_init(dsc.p1.x, dsc.p1.y, dsc.p2.x, dsc.p2.y, lv.DRAW_MASK_
↪LINE_SIDE.BOTTOM)
    # line_mask_id = line_mask_param.draw_mask_add(None)
    line_mask_id = lv.draw_mask_add(line_mask_param, None)
    # Add a fade effect: transparent bottom covering top
    h = obj.get_height()
    fade_mask_param = lv.draw_mask_fade_param_t()
    coords = lv.area_t()
    obj.get_coords(coords)
    fade_mask_param.init(coords, lv.OPA.COVER, coords.y1 + h // 8, lv.OPA.TRANSP,
↪coords.y2)
    fade_mask_id = lv.draw_mask_add(fade_mask_param,None)

    # Draw a rectangle that will be affected by the mask
    draw_rect_dsc = lv.draw_rect_dsc_t()
    draw_rect_dsc.init()
    draw_rect_dsc.bg_opa = lv.OPA._20
    draw_rect_dsc.bg_color = dsc.line_dsc.color

    a = lv.area_t()
    a.x1 = dsc.p1.x
    a.x2 = dsc.p2.x - 1
    a.y1 = min(dsc.p1.y, dsc.p2.y)
    coords = lv.area_t()
    obj.get_coords(coords)
    a.y2 = coords.y2
```

(下页继续)

(续上页)

```
    dsc.draw_ctx.rect(draw_rect_dsc, a)

    # Remove the masks
    lv.draw_mask_remove_id(line_mask_id)
    lv.draw_mask_remove_id(fade_mask_id)


def add_data(timer):
    # LV_UNUSED(timer);
    cnt = 0
    chart1.set_next_value(ser1, lv.rand(20, 90))

    if cnt % 4 == 0:
        chart1.set_next_value(ser2, lv.rand(40, 60))

    cnt +=1

#
# Add a faded area effect to the line chart
#

# Create a chart1
chart1 = lv.chart(lv.scr_act())
chart1.set_size(200, 150)
chart1.center()
chart1.set_type(lv.chart.TYPE.LINE)    # Show lines and points too

chart1.add_event_cb(draw_event_cb, lv.EVENT.DRAW_PART_BEGIN, None)
chart1.set_update_mode(lv.chart.UPDATE_MODE.CIRCULAR)

# Add two data series
ser1 = chart1.add_series(lv.palette_main(lv.PALETTE.RED), lv.chart.AXIS.PRIMARY_Y)
ser2 = chart1.add_series(lv.palette_main(lv.PALETTE.BLUE), lv.chart.AXIS.SECONDARY_Y)

for i in range(10):
    chart1.set_next_value(ser1, lv.rand(20, 90))
    chart1.set_next_value(ser2, lv.rand(30, 70))

timer = lv.timer_create(add_data, 200, None)
```

Axis ticks and labels with scrolling

```
#include "../../lv_examples.h"
#if LV_USE_CHART && LV_BUILD_EXAMPLES

static void draw_event_cb(lv_event_t * e)
{
    lv_obj_draw_part_dsc_t * dsc = lv_event_get_draw_part_dsc(e);
    if(!lv_obj_draw_part_check_type(dsc, &lv_chart_class, LV_CHART_DRAW_PART_TICK_
↪LABEL)) return;

    if(dsc->id == LV_CHART_AXIS_PRIMARY_X && dsc->text) {
        const char * month[] = {"Jan", "Febr", "March", "Apr", "May", "Jun", "July",
↪"Aug", "Sept", "Oct", "Nov", "Dec"};
        lv_snprintf(dsc->text, dsc->text_length, "%s", month[dsc->value]);
    }
}

/**
 * Add ticks and labels to the axis and demonstrate scrolling
 */
void lv_example_chart_3(void)
{
    /*Create a chart*/
    lv_obj_t * chart;
    chart = lv_chart_create(lv_scr_act());
    lv_obj_set_size(chart, 200, 150);
    lv_obj_center(chart);
    lv_chart_set_type(chart, LV_CHART_TYPE_BAR);
    lv_chart_set_range(chart, LV_CHART_AXIS_PRIMARY_Y, 0, 100);
    lv_chart_set_range(chart, LV_CHART_AXIS_SECONDARY_Y, 0, 400);
    lv_chart_set_point_count(chart, 12);
    lv_obj_add_event_cb(chart, draw_event_cb, LV_EVENT_DRAW_PART_BEGIN, NULL);

    /*Add ticks and label to every axis*/
    lv_chart_set_axis_tick(chart, LV_CHART_AXIS_PRIMARY_X, 10, 5, 12, 3, true, 40);
    lv_chart_set_axis_tick(chart, LV_CHART_AXIS_PRIMARY_Y, 10, 5, 6, 2, true, 50);
    lv_chart_set_axis_tick(chart, LV_CHART_AXIS_SECONDARY_Y, 10, 5, 3, 4, true, 50);

    /*Zoom in a little in X*/
    lv_chart_set_zoom_x(chart, 800);

    /*Add two data series*/
    lv_chart_series_t * ser1 = lv_chart_add_series(chart, lv_palette_lighten(LV_
↪PALETTE_GREEN, 2), LV_CHART_AXIS_PRIMARY_Y);
```

(下页继续)

(续上页)

```
    lv_chart_series_t * ser2 = lv_chart_add_series(chart, lv_palette_darken(LV_
↪PALETTE_GREEN, 2), LV_CHART_AXIS_SECONDARY_Y);

    /*Set the next points on 'ser1'*/
    lv_chart_set_next_value(chart, ser1, 31);
    lv_chart_set_next_value(chart, ser1, 66);
    lv_chart_set_next_value(chart, ser1, 10);
    lv_chart_set_next_value(chart, ser1, 89);
    lv_chart_set_next_value(chart, ser1, 63);
    lv_chart_set_next_value(chart, ser1, 56);
    lv_chart_set_next_value(chart, ser1, 32);
    lv_chart_set_next_value(chart, ser1, 35);
    lv_chart_set_next_value(chart, ser1, 57);
    lv_chart_set_next_value(chart, ser1, 85);
    lv_chart_set_next_value(chart, ser1, 22);
    lv_chart_set_next_value(chart, ser1, 58);

    lv_coord_t * ser2_array = lv_chart_get_y_array(chart, ser2);
    /*Directly set points on 'ser2'*/
    ser2_array[0] = 92;
    ser2_array[1] = 71;
    ser2_array[2] = 61;
    ser2_array[3] = 15;
    ser2_array[4] = 21;
    ser2_array[5] = 35;
    ser2_array[6] = 35;
    ser2_array[7] = 58;
    ser2_array[8] = 31;
    ser2_array[9] = 53;
    ser2_array[10] = 33;
    ser2_array[11] = 73;

    lv_chart_refresh(chart); /*Required after direct set*/
}

#endif
```

```
def draw_event_cb(e):

    dsc = lv.obj_draw_part_dsc_t.__cast__(e.get_param())
    if dsc.part == lv.PART.TICKS and dsc.id == lv.chart.AXIS.PRIMARY_X:
        month = ["Jan", "Febr", "March", "Apr", "May", "Jun", "July", "Aug", "Sept",
↪"Oct", "Nov", "Dec"]
```

(下页继续)

(续上页)

```
        # dsc.text is defined char text[16], I must therefore convert the Python␣
↪string to a byte_array
        dsc.text = bytes(month[dsc.value],"ascii")
#
# Add ticks and labels to the axis and demonstrate scrolling
#

# Create a chart
chart = lv.chart(lv.scr_act())
chart.set_size(200, 150)
chart.center()
chart.set_type(lv.chart.TYPE.BAR)
chart.set_range(lv.chart.AXIS.PRIMARY_Y, 0, 100)
chart.set_range(lv.chart.AXIS.SECONDARY_Y, 0, 400)
chart.set_point_count(12)
chart.add_event_cb(draw_event_cb, lv.EVENT.DRAW_PART_BEGIN, None)

# Add ticks and label to every axis
chart.set_axis_tick(lv.chart.AXIS.PRIMARY_X, 10, 5, 12, 3, True, 40)
chart.set_axis_tick(lv.chart.AXIS.PRIMARY_Y, 10, 5, 6, 2, True, 50)
chart.set_axis_tick(lv.chart.AXIS.SECONDARY_Y, 10, 5, 3, 4,True, 50)

# Zoom in a little in X
chart.set_zoom_x(800)

# Add two data series
ser1 = lv.chart.add_series(chart, lv.palette_lighten(lv.PALETTE.GREEN, 2), lv.chart.
↪AXIS.PRIMARY_Y)
ser2 = lv.chart.add_series(chart, lv.palette_darken(lv.PALETTE.GREEN, 2), lv.chart.
↪AXIS.SECONDARY_Y)

# Set the next points on 'ser1'
chart.set_next_value(ser1, 31)
chart.set_next_value(ser1, 66)
chart.set_next_value(ser1, 10)
chart.set_next_value(ser1, 89)
chart.set_next_value(ser1, 63)
chart.set_next_value(ser1, 56)
chart.set_next_value(ser1, 32)
chart.set_next_value(ser1, 35)
chart.set_next_value(ser1, 57)
chart.set_next_value(ser1, 85)
chart.set_next_value(ser1, 22)
```

(下页继续)

(续上页)

```
chart.set_next_value(ser1, 58)

# Directly set points on 'ser2'
ser2.y_points = [92,71,61,15,21,35,35,58,31,53,33,73]

chart.refresh()  # Required after direct set
```

Show the value of the pressed points

```
#include "../../lv_examples.h"
#if LV_USE_CHART && LV_BUILD_EXAMPLES


static void event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * chart = lv_event_get_target(e);

    if(code == LV_EVENT_VALUE_CHANGED) {
        lv_obj_invalidate(chart);
    }
    if(code == LV_EVENT_REFR_EXT_DRAW_SIZE) {
        lv_coord_t * s = lv_event_get_param(e);
        *s = LV_MAX(*s, 20);
    }
    else if(code == LV_EVENT_DRAW_POST_END) {
        int32_t id = lv_chart_get_pressed_point(chart);
        if(id == LV_CHART_POINT_NONE) return;

        LV_LOG_USER("Selected point %d", (int)id);

        lv_chart_series_t * ser = lv_chart_get_series_next(chart, NULL);
        while(ser) {
            lv_point_t p;
            lv_chart_get_point_pos_by_id(chart, ser, id, &p);

            lv_coord_t * y_array = lv_chart_get_y_array(chart, ser);
            lv_coord_t value = y_array[id];

            char buf[16];
            lv_snprintf(buf, sizeof(buf), LV_SYMBOL_DUMMY"$%d", value);
```

(下页继续)

(续上页)

```
            lv_draw_rect_dsc_t draw_rect_dsc;
            lv_draw_rect_dsc_init(&draw_rect_dsc);
            draw_rect_dsc.bg_color = lv_color_black();
            draw_rect_dsc.bg_opa = LV_OPA_50;
            draw_rect_dsc.radius = 3;
            draw_rect_dsc.bg_img_src = buf;
            draw_rect_dsc.bg_img_recolor = lv_color_white();

            lv_area_t a;
            a.x1 = chart->coords.x1 + p.x - 20;
            a.x2 = chart->coords.x1 + p.x + 20;
            a.y1 = chart->coords.y1 + p.y - 30;
            a.y2 = chart->coords.y1 + p.y - 10;

            lv_draw_ctx_t * draw_ctx = lv_event_get_draw_ctx(e);
            lv_draw_rect(draw_ctx, &draw_rect_dsc, &a);

            ser = lv_chart_get_series_next(chart, ser);
        }
    }
    else if(code == LV_EVENT_RELEASED) {
        lv_obj_invalidate(chart);
    }
}

/**
 * Show the value of the pressed points
 */
void lv_example_chart_4(void)
{
    /*Create a chart*/
    lv_obj_t * chart;
    chart = lv_chart_create(lv_scr_act());
    lv_obj_set_size(chart, 200, 150);
    lv_obj_center(chart);

    lv_obj_add_event_cb(chart, event_cb, LV_EVENT_ALL, NULL);
    lv_obj_refresh_ext_draw_size(chart);

    /*Zoom in a little in X*/
    lv_chart_set_zoom_x(chart, 800);
```

(下页继续)

(续上页)

```
    /*Add two data series*/
    lv_chart_series_t * ser1 = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_
↪RED), LV_CHART_AXIS_PRIMARY_Y);
    lv_chart_series_t * ser2 = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_
↪GREEN), LV_CHART_AXIS_PRIMARY_Y);
    uint32_t i;
    for(i = 0; i < 10; i++) {
        lv_chart_set_next_value(chart, ser1, lv_rand(60,90));
        lv_chart_set_next_value(chart, ser2, lv_rand(10,40));
    }
}

#endif
```

```
def event_cb(e):
    code = e.get_code()
    chart = e.get_target()

    if code == lv.EVENT.VALUE_CHANGED:
        chart.invalidate()

    if code == lv.EVENT.REFR_EXT_DRAW_SIZE:
        e.set_ext_draw_size(20)

    elif code == lv.EVENT.DRAW_POST_END:
        id = lv.chart.get_pressed_point(chart)
        if id == lv.CHART_POINT.NONE:
            return
        # print("Selected point ", id)
        for i in range(len(series)):
            p = lv.point_t()
            chart.get_point_pos_by_id(series[i], id, p)
            value = series_points[i][id]
            buf = lv.SYMBOL.DUMMY + "$" + str(value)

            draw_rect_dsc = lv.draw_rect_dsc_t()
            draw_rect_dsc.init()
            draw_rect_dsc.bg_color = lv.color_black()
            draw_rect_dsc.bg_opa = lv.OPA._50
            draw_rect_dsc.radius = 3
            draw_rect_dsc.bg_img_src = buf
            draw_rect_dsc.bg_img_recolor = lv.color_white()
```

(下页继续)

(续上页)

```
            a = lv.area_t()
            coords = lv.area_t()
            chart.get_coords(coords)
            a.x1 = coords.x1 + p.x - 20
            a.x2 = coords.x1 + p.x + 20
            a.y1 = coords.y1 + p.y - 30
            a.y2 = coords.y1 + p.y - 10

            clip_area = lv.area_t.__cast__(e.get_param())
            lv.draw_rect(a, clip_area, draw_rect_dsc)

    elif code == lv.EVENT.RELEASED:
        chart.invalidate()

#
# Add ticks and labels to the axis and demonstrate scrolling
#

# Create a chart
chart = lv.chart(lv.scr_act())
chart.set_size(200, 150)
chart.center()

chart.add_event_cb(event_cb, lv.EVENT.ALL, None)
chart.refresh_ext_draw_size()

# Zoom in a little in X
chart.set_zoom_x(800)

# Add two data series
ser1 = chart.add_series(lv.palette_main(lv.PALETTE.RED), lv.chart.AXIS.PRIMARY_Y)
ser2 = chart.add_series(lv.palette_main(lv.PALETTE.GREEN), lv.chart.AXIS.PRIMARY_Y)

ser1_p = []
ser2_p = []
for i in range(10):
    ser1_p.append(lv.rand(60,90))
    ser2_p.append(lv.rand(10,40))
ser1.y_points = ser1_p
ser2.y_points = ser2_p

series = [ser1,ser2]
series_points=[ser1_p,ser2_p]
```

Display 1000 data points with zooming and scrolling

```
#include "../../lv_examples.h"
#if LV_USE_CHART && LV_USE_SLIDER && LV_BUILD_EXAMPLES

static lv_obj_t * chart;
/* Source: https://github.com/ankur219/ECG-Arrhythmia-classification/blob/
↪642230149583adfae1e4bd26c6f0e1fd8af2be0e/sample.csv*/
static const lv_coord_t ecg_sample[] = {
    -2, 2, 0, -15, -39, -63, -71, -68, -67, -69, -84, -95, -104, -107, -108, -107, -
↪107, -107, -107, -114, -118, -117,
    -112, -100, -89, -83, -71, -64, -58, -58, -62, -62, -58, -51, -46, -39, -27, -10,␣
↪4, 7, 1, -3, 0, 14, 24, 30, 25, 19,
    13, 7, 12, 15, 18, 21, 13, 6, 9, 8, 17, 19, 13, 11, 11, 11, 23, 30, 37, 34, 25,␣
↪14, 15, 19, 28, 31, 26, 23, 25, 31,
    39, 37, 37, 34, 30, 32, 22, 29, 31, 33, 37, 23, 13, 7, 2, 4, -2, 2, 11, 22, 33,␣
↪19, -1, -27, -55, -67, -72, -71, -63,
    -49, -18, 35, 113, 230, 369, 525, 651, 722, 730, 667, 563, 454, 357, 305, 288,␣
↪274, 255, 212, 173, 143, 117, 82, 39,
    -13, -53, -78, -91, -101, -113, -124, -131, -131, -131, -129, -128, -129, -125, -
↪123, -123, -129, -139, -148, -153,
    -159, -166, -183, -205, -227, -243, -248, -246, -254, -280, -327, -381, -429, -
↪473, -517, -556, -592, -612, -620,
    -620, -614, -604, -591, -574, -540, -497, -441, -389, -358, -336, -313, -284, -
↪222, -167, -114, -70, -47, -28, -4, 12,
    38, 52, 58, 56, 56, 57, 68, 77, 86, 86, 80, 69, 67, 70, 82, 85, 89, 90, 89, 89,␣
↪88, 91, 96, 97, 91, 83, 78, 82, 88, 95,
    96, 105, 106, 110, 102, 100, 96, 98, 97, 101, 98, 99, 100, 107, 113, 119, 115,␣
↪110, 96, 85, 73, 64, 69, 76, 79,
    78, 75, 85, 100, 114, 113, 105, 96, 84, 74, 66, 60, 75, 85, 89, 83, 67, 61, 67,␣
↪73, 79, 74, 63, 57, 56, 58, 61, 55,
    48, 45, 46, 55, 62, 55, 49, 43, 50, 59, 63, 57, 40, 31, 23, 25, 27, 31, 35, 34,␣
↪30, 36, 34, 42, 38, 36, 40, 46, 50,
    47, 32, 30, 32, 52, 67, 73, 71, 63, 54, 53, 45, 41, 28, 13, 3, 1, 4, 4, -8, -23, -
↪32, -31, -19, -5, 3, 9, 13, 19,
    24, 27, 29, 25, 22, 26, 32, 42, 51, 56, 60, 57, 55, 53, 53, 54, 59, 54, 49, 26, -
↪3, -11, -20, -47, -100, -194, -236,
    -212, -123, 8, 103, 142, 147, 120, 105, 98, 93, 81, 61, 40, 26, 28, 30, 30, 27,␣
↪19, 17, 21, 20, 19, 19, 22, 36, 40,
    35, 20, 7, 1, 10, 18, 27, 22, 6, -4, -2, 3, 6, -2, -13, -14, -10, -2, 3, 2, -1, -
↪5, -10, -19, -32, -42, -55, -60,
    -68, -77, -86, -101, -110, -117, -115, -104, -92, -84, -85, -84, -73, -65, -52, -
↪50, -45, -35, -20, -3, 12, 20, 25,
    26, 28, 28, 30, 28, 25, 28, 33, 42, 42, 36, 23, 9, 0, 1, -4, 1, -4, -4, 1, 5, 9,␣
↪9, -3, -1, -18, -50, -108, -190,
```

(下页继续)

(续上页)

```
    -272, -340, -408, -446, -537, -643, -777, -894, -920, -853, -697, -461, -251, -60,
↪ 58, 103, 129, 139, 155, 170, 173,
    178, 185, 190, 193, 200, 208, 215, 225, 224, 232, 234, 240, 240, 236, 229, 226,␣
↪224, 232, 233, 232, 224, 219, 219,
    223, 231, 226, 223, 219, 218, 223, 223, 223, 233, 245, 268, 286, 296, 295, 283,␣
↪271, 263, 252, 243, 226, 210, 197,
    186, 171, 152, 133, 117, 114, 110, 107, 96, 80, 63, 48, 40, 38, 34, 28, 15, 2, -7,
↪ -11, -14, -18, -29, -37, -44, -50,
    -58, -63, -61, -52, -50, -48, -61, -59, -58, -54, -47, -52, -62, -61, -64, -54, -
↪52, -59, -69, -76, -76, -69, -67,
    -74, -78, -81, -80, -73, -65, -57, -53, -51, -47, -35, -27, -22, -22, -24, -21, -
↪17, -13, -10, -11, -13, -20, -20,
    -12, -2, 7, -1, -12, -16, -13, -2, 2, -4, -5, -2, 9, 19, 19, 14, 11, 13, 19, 21,␣
↪20, 18, 19, 19, 19, 16, 15, 13, 14,
    9, 3, -5, -9, -5, -3, -2, -3, -3, 2, 8, 9, 9, 5, 6, 8, 8, 7, 4, 3, 4, 5, 3, 5, 5,␣
↪13, 13, 12, 10, 10, 15, 22, 17,
    14, 7, 10, 15, 16, 11, 12, 10, 13, 9, -2, -4, -2, 7, 16, 16, 17, 16, 7, -1, -16, -
↪18, -16, -9, -4, -5, -10, -9, -8,
    -3, -4, -10, -19, -20, -16, -9, -9, -23, -40, -48, -43, -33, -19, -21, -26, -31, -
↪33, -19, 0, 17, 24, 9, -17, -47,
    -63, -67, -59, -52, -51, -50, -49, -42, -26, -21, -15, -20, -23, -22, -19, -12, -
↪8, 5, 18, 27, 32, 26, 25, 26, 22,
    23, 17, 14, 17, 21, 25, 2, -45, -121, -196, -226, -200, -118, -9, 73, 126, 131,␣
↪114, 87, 60, 42, 29, 26, 34, 35, 34,
    25, 12, 9, 7, 3, 2, -8, -11, 2, 23, 38, 41, 23, 9, 10, 13, 16, 8, -8, -17, -23, -
↪26, -25, -21, -15, -10, -13, -13,
    -19, -22, -29, -40, -48, -48, -54, -55, -66, -82, -85, -90, -92, -98, -114, -119,␣
↪-124, -129, -132, -146, -146, -138,
    -124, -99, -85, -72, -65, -65, -65, -66, -63, -64, -64, -58, -46, -26, -9, 2, 2,␣
↪4, 0, 1, 4, 3, 10, 11, 10, 2, -4,
    0, 10, 18, 20, 6, 2, -9, -7, -3, -3, -2, -7, -12, -5, 5, 24, 36, 31, 25, 6, 3, 7,␣
↪12, 17, 11, 0, -6, -9, -8, -7, -5,
    -6, -2, -2, -6, -2, 2, 14, 24, 22, 15, 8, 4, 6, 7, 12, 16, 25, 20, 7, -16, -41, -
↪60, -67, -65, -54, -35, -11, 30,
    84, 175, 302, 455, 603, 707, 743, 714, 625, 519, 414, 337, 300, 281, 263, 239,␣
↪197, 163, 136, 109, 77, 34, -18, -50,
    -66, -74, -79, -92, -107, -117, -127, -129, -135, -139, -141, -155, -159, -167, -
↪171, -169, -174, -175, -178, -191,
    -202, -223, -235, -243, -237, -240, -256, -298, -345, -393, -432, -475, -518, -
↪565, -596, -619, -623, -623, -614,
    -599, -583, -559, -524, -477, -425, -383, -357, -331, -301, -252, -198, -143, -96,
↪ -57, -29, -8, 10, 31, 45, 60, 65,
    70, 74, 76, 79, 82, 79, 75, 62,
```

(下页继续)

(续上页)

```
};

static void slider_x_event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target(e);
    int32_t v = lv_slider_get_value(obj);
    lv_chart_set_zoom_x(chart, v);
}

static void slider_y_event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target(e);
    int32_t v = lv_slider_get_value(obj);
    lv_chart_set_zoom_y(chart, v);
}

/**
 * Display 1000 data points with zooming and scrolling.
 * See how the chart changes drawing mode (draw only vertical lines) when
 * the points get too crowded.
 */
void lv_example_chart_5(void)
{
    /*Create a chart*/
    chart = lv_chart_create(lv_scr_act());
    lv_obj_set_size(chart, 200, 150);
    lv_obj_align(chart, LV_ALIGN_CENTER, -30, -30);
    lv_chart_set_range(chart, LV_CHART_AXIS_PRIMARY_Y, -1000, 1000);

    /*Do not display points on the data*/
    lv_obj_set_style_size(chart, 0, LV_PART_INDICATOR);

    lv_chart_series_t * ser = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_
↪RED), LV_CHART_AXIS_PRIMARY_Y);

    uint32_t pcnt = sizeof(ecg_sample) / sizeof(ecg_sample[0]);
    lv_chart_set_point_count(chart, pcnt);
    lv_chart_set_ext_y_array(chart, ser, (lv_coord_t *)ecg_sample);

    lv_obj_t * slider;
    slider = lv_slider_create(lv_scr_act());
    lv_slider_set_range(slider, LV_IMG_ZOOM_NONE, LV_IMG_ZOOM_NONE * 10);
    lv_obj_add_event_cb(slider, slider_x_event_cb, LV_EVENT_VALUE_CHANGED, NULL);
```

(下页继续)

(续上页)

```
    lv_obj_set_size(slider, 200, 10);
    lv_obj_align_to(slider, chart, LV_ALIGN_OUT_BOTTOM_MID, 0, 20);

    slider = lv_slider_create(lv_scr_act());
    lv_slider_set_range(slider, LV_IMG_ZOOM_NONE, LV_IMG_ZOOM_NONE * 10);
    lv_obj_add_event_cb(slider, slider_y_event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    lv_obj_set_size(slider, 10, 150);
    lv_obj_align_to(slider, chart, LV_ALIGN_OUT_RIGHT_MID, 20, 0);
}

#endif
```

```
# Source: https://github.com/ankur219/ECG-Arrhythmia-classification/blob/
↪642230149583adfae1e4bd26c6f0e1fd8af2be0e/sample.csv
ecg_sample = [
    -2, 2, 0, -15, -39, -63, -71, -68, -67, -69, -84, -95, -104, -107, -108, -107, -
↪107, -107, -107, -114, -118, -117,
    -112, -100, -89, -83, -71, -64, -58, -58, -62, -62, -58, -51, -46, -39, -27, -10,␣
↪4, 7, 1, -3, 0, 14, 24, 30, 25, 19,
    13, 7, 12, 15, 18, 21, 13, 6, 9, 8, 17, 19, 13, 11, 11, 11, 23, 30, 37, 34, 25,␣
↪14, 15, 19, 28, 31, 26, 23, 25, 31,
    39, 37, 37, 34, 30, 32, 22, 29, 31, 33, 37, 23, 13, 7, 2, 4, -2, 2, 11, 22, 33,␣
↪19, -1, -27, -55, -67, -72, -71, -63,
    -49, -18, 35, 113, 230, 369, 525, 651, 722, 730, 667, 563, 454, 357, 305, 288,␣
↪274, 255, 212, 173, 143, 117, 82, 39,
    -13, -53, -78, -91, -101, -113, -124, -131, -131, -131, -129, -128, -129, -125, -
↪123, -123, -129, -139, -148, -153,
    -159, -166, -183, -205, -227, -243, -248, -246, -254, -280, -327, -381, -429, -
↪473, -517, -556, -592, -612, -620,
    -620, -614, -604, -591, -574, -540, -497, -441, -389, -358, -336, -313, -284, -
↪222, -167, -114, -70, -47, -28, -4, 12,
    38, 52, 58, 56, 56, 57, 68, 77, 86, 86, 80, 69, 67, 70, 82, 85, 89, 90, 89, 89,␣
↪88, 91, 96, 97, 91, 83, 78, 82, 88, 95,
    96, 105, 106, 110, 102, 100, 96, 98, 97, 101, 98, 99, 100, 107, 113, 119, 115,␣
↪110, 96, 85, 73, 64, 69, 76, 79,
    78, 75, 85, 100, 114, 113, 105, 96, 84, 74, 66, 60, 75, 85, 89, 83, 67, 61, 67,␣
↪73, 79, 74, 63, 57, 56, 58, 61, 55,
    48, 45, 46, 55, 62, 55, 49, 43, 50, 59, 63, 57, 40, 31, 23, 25, 27, 31, 35, 34,␣
↪30, 36, 34, 42, 38, 36, 40, 46, 50,
    47, 32, 30, 32, 52, 67, 73, 71, 63, 54, 53, 45, 41, 28, 13, 3, 1, 4, 4, -8, -23, -
↪32, -31, -19, -5, 3, 9, 13, 19,
    24, 27, 29, 25, 22, 26, 32, 42, 51, 56, 60, 57, 55, 53, 53, 54, 59, 54, 49, 26, -
↪3, -11, -20, -47, -100, -194, -236,
```

(下页继续)

(续上页)

```
    -212, -123, 8, 103, 142, 147, 120, 105, 98, 93, 81, 61, 40, 26, 28, 30, 30, 27,␣
↪19, 17, 21, 20, 19, 19, 22, 36, 40,
    35, 20, 7, 1, 10, 18, 27, 22, 6, -4, -2, 3, 6, -2, -13, -14, -10, -2, 3, 2, -1, -
↪5, -10, -19, -32, -42, -55, -60,
    -68, -77, -86, -101, -110, -117, -115, -104, -92, -84, -85, -84, -73, -65, -52, -
↪50, -45, -35, -20, -3, 12, 20, 25,
    26, 28, 28, 30, 28, 25, 28, 33, 42, 42, 36, 23, 9, 0, 1, -4, 1, -4, -4, 1, 5, 9,␣
↪9, -3, -1, -18, -50, -108, -190,
    -272, -340, -408, -446, -537, -643, -777, -894, -920, -853, -697, -461, -251, -60,
↪ 58, 103, 129, 139, 155, 170, 173,
    178, 185, 190, 193, 200, 208, 215, 225, 224, 232, 234, 240, 240, 236, 229, 226,␣
↪224, 232, 233, 232, 224, 219, 219,
    223, 231, 226, 223, 219, 218, 223, 223, 223, 233, 245, 268, 286, 296, 295, 283,␣
↪271, 263, 252, 243, 226, 210, 197,
    186, 171, 152, 133, 117, 114, 110, 107, 96, 80, 63, 48, 40, 38, 34, 28, 15, 2, -7,
↪ -11, -14, -18, -29, -37, -44, -50,
    -58, -63, -61, -52, -50, -48, -61, -59, -58, -54, -47, -52, -62, -61, -64, -54, -
↪52, -59, -69, -76, -76, -69, -67,
    -74, -78, -81, -80, -73, -65, -57, -53, -51, -47, -35, -27, -22, -22, -24, -21, -
↪17, -13, -10, -11, -13, -20, -20,
    -12, -2, 7, -1, -12, -16, -13, -2, 2, -4, -5, -2, 9, 19, 19, 14, 11, 13, 19, 21,␣
↪20, 18, 19, 19, 19, 16, 15, 13, 14,
    9, 3, -5, -9, -5, -3, -2, -3, -3, 2, 8, 9, 9, 5, 6, 8, 8, 7, 4, 3, 4, 5, 3, 5, 5,␣
↪13, 13, 12, 10, 10, 15, 22, 17,
    14, 7, 10, 15, 16, 11, 12, 10, 13, 9, -2, -4, -2, 7, 16, 16, 17, 16, 7, -1, -16, -
↪18, -16, -9, -4, -5, -10, -9, -8,
    -3, -4, -10, -19, -20, -16, -9, -9, -23, -40, -48, -43, -33, -19, -21, -26, -31, -
↪33, -19, 0, 17, 24, 9, -17, -47,
    -63, -67, -59, -52, -51, -50, -49, -42, -26, -21, -15, -20, -23, -22, -19, -12, -
↪8, 5, 18, 27, 32, 26, 25, 26, 22,
    23, 17, 14, 17, 21, 25, 2, -45, -121, -196, -226, -200, -118, -9, 73, 126, 131,␣
↪114, 87, 60, 42, 29, 26, 34, 35, 34,
    25, 12, 9, 7, 3, 2, -8, -11, 2, 23, 38, 41, 23, 9, 10, 13, 16, 8, -8, -17, -23, -
↪26, -25, -21, -15, -10, -13, -13,
    -19, -22, -29, -40, -48, -48, -54, -55, -66, -82, -85, -90, -92, -98, -114, -119,␣
↪-124, -129, -132, -146, -146, -138,
    -124, -99, -85, -72, -65, -65, -65, -66, -63, -64, -64, -58, -46, -26, -9, 2, 2,␣
↪4, 0, 1, 4, 3, 10, 11, 10, 2, -4,
    0, 10, 18, 20, 6, 2, -9, -7, -3, -3, -2, -7, -12, -5, 5, 24, 36, 31, 25, 6, 3, 7,␣
↪12, 17, 11, 0, -6, -9, -8, -7, -5,
    -6, -2, -2, -6, -2, 2, 14, 24, 22, 15, 8, 4, 6, 7, 12, 16, 25, 20, 7, -16, -41, -
↪60, -67, -65, -54, -35, -11, 30,
    84, 175, 302, 455, 603, 707, 743, 714, 625, 519, 414, 337, 300, 281, 263, 239,␣
↪197, 163, 136, 109, 77, 34, -18, -50,
```

(下页继续)

(续上页)

```
    -66, -74, -79, -92, -107, -117, -127, -129, -135, -139, -141, -155, -159, -167, -
↪171, -169, -174, -175, -178, -191,
    -202, -223, -235, -243, -237, -240, -256, -298, -345, -393, -432, -475, -518, -
↪565, -596, -619, -623, -623, -614,
    -599, -583, -559, -524, -477, -425, -383, -357, -331, -301, -252, -198, -143, -96,
↪ -57, -29, -8, 10, 31, 45, 60, 65,
    70, 74, 76, 79, 82, 79, 75, 62,
]

def slider_x_event_cb(e):

    slider = e.get_target()
    v = slider.get_value()
    chart.set_zoom_x(v)

def slider_y_event_cb(e):

    slider = e.get_target()
    v = slider.get_value()
    chart.set_zoom_y(v)


#
# Display 1000 data points with zooming and scrolling.
# See how the chart changes drawing mode (draw only vertical lines) when
# the points get too crowded.

# Create a chart
chart = lv.chart(lv.scr_act())
chart.set_size(200, 150)
chart.align(lv.ALIGN.CENTER, -30, -30)
chart.set_range(lv.chart.AXIS.PRIMARY_Y, -1000, 1000)

# Do not display points on the data
chart.set_style_size(0, lv.PART.INDICATOR)

ser = chart.add_series(lv.palette_main(lv.PALETTE.RED), lv.chart.AXIS.PRIMARY_Y)

pcnt = len(ecg_sample)
chart.set_point_count(pcnt)
chart.set_ext_y_array(ser, ecg_sample)

slider = lv.slider(lv.scr_act())
```

(下页继续)

(续上页)

```
slider.set_range(lv.IMG_ZOOM.NONE, lv.IMG_ZOOM.NONE * 10)
slider.add_event_cb(slider_x_event_cb, lv.EVENT.VALUE_CHANGED, None)
slider.set_size(200,10)
slider.align_to(chart, lv.ALIGN.OUT_BOTTOM_MID, 0, 20)

slider = lv.slider(lv.scr_act())
slider.set_range(lv.IMG_ZOOM.NONE, lv.IMG_ZOOM.NONE * 10)
slider.add_event_cb(slider_y_event_cb, lv.EVENT.VALUE_CHANGED, None)
slider.set_size(10, 150)
slider.align_to(chart, lv.ALIGN.OUT_RIGHT_MID, 20, 0)
```

Show cursor on the clicked point

```
#include "../../lv_examples.h"
#if LV_USE_CHART && LV_BUILD_EXAMPLES

static lv_obj_t * chart;
static lv_chart_series_t * ser;
static lv_chart_cursor_t * cursor;

static void event_cb(lv_event_t * e)
{
    static int32_t last_id = -1;
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target(e);

    if(code == LV_EVENT_VALUE_CHANGED) {
        last_id = lv_chart_get_pressed_point(obj);
        if(last_id != LV_CHART_POINT_NONE) {
            lv_chart_set_cursor_point(obj, cursor, NULL, last_id);
        }
    }
    else if(code == LV_EVENT_DRAW_PART_END) {
        lv_obj_draw_part_dsc_t * dsc = lv_event_get_draw_part_dsc(e);
        if(!lv_obj_draw_part_check_type(dsc, &lv_chart_class, LV_CHART_DRAW_PART_
↪CURSOR)) return;
        if(dsc->p1 == NULL || dsc->p2 == NULL || dsc->p1->y != dsc->p2->y || last_id
↪< 0) return;

        lv_coord_t * data_array = lv_chart_get_y_array(chart, ser);
        lv_coord_t v = data_array[last_id];
```

(下页继续)

(续上页)

```
        char buf[16];
        lv_snprintf(buf, sizeof(buf), "%d", v);

        lv_point_t size;
        lv_txt_get_size(&size, buf, LV_FONT_DEFAULT, 0, 0, LV_COORD_MAX, LV_TEXT_FLAG_
↪NONE);

        lv_area_t a;
        a.y2 = dsc->p1->y - 5;
        a.y1 = a.y2 - size.y - 10;
        a.x1 = dsc->p1->x + 10;
        a.x2 = a.x1 + size.x + 10;

        lv_draw_rect_dsc_t draw_rect_dsc;
        lv_draw_rect_dsc_init(&draw_rect_dsc);
        draw_rect_dsc.bg_color = lv_palette_main(LV_PALETTE_BLUE);
        draw_rect_dsc.radius = 3;

        lv_draw_rect(dsc->draw_ctx, &draw_rect_dsc, &a);

        lv_draw_label_dsc_t draw_label_dsc;
        lv_draw_label_dsc_init(&draw_label_dsc);
        draw_label_dsc.color = lv_color_white();
        a.x1 += 5;
        a.x2 -= 5;
        a.y1 += 5;
        a.y2 -= 5;
        lv_draw_label(dsc->draw_ctx, &draw_label_dsc, &a, buf, NULL);
    }
}

/**
 * Show cursor on the clicked point
 */
void lv_example_chart_6(void)
{
    chart = lv_chart_create(lv_scr_act());
    lv_obj_set_size(chart, 200, 150);
    lv_obj_align(chart, LV_ALIGN_CENTER, 0, -10);

    lv_chart_set_axis_tick(chart, LV_CHART_AXIS_PRIMARY_Y, 10, 5, 6, 5, true, 40);
    lv_chart_set_axis_tick(chart, LV_CHART_AXIS_PRIMARY_X, 10, 5, 10, 1, true, 30);
```

(下页继续)

(续上页)

```
    lv_obj_add_event_cb(chart, event_cb, LV_EVENT_ALL, NULL);
    lv_obj_refresh_ext_draw_size(chart);

    cursor = lv_chart_add_cursor(chart, lv_palette_main(LV_PALETTE_BLUE), LV_DIR_LEFT␣
↪| LV_DIR_BOTTOM);

    ser = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_RED), LV_CHART_AXIS_
↪PRIMARY_Y);
    uint32_t i;
    for(i = 0; i < 10; i++) {
        lv_chart_set_next_value(chart, ser, lv_rand(10,90));
    }

    lv_chart_set_zoom_x(chart, 500);

    lv_obj_t * label = lv_label_create(lv_scr_act());
    lv_label_set_text(label, "Click on a point");
    lv_obj_align_to(label, chart, LV_ALIGN_OUT_TOP_MID, 0, -5);
}

#endif
```

```
class ExampleChart_6():

    def __init__(self):
        self.last_id = -1
        #
        # Show cursor on the clicked point
        #

        chart = lv.chart(lv.scr_act())
        chart.set_size(200, 150)
        chart.align(lv.ALIGN.CENTER, 0, -10)

        chart.set_axis_tick(lv.chart.AXIS.PRIMARY_Y, 10, 5, 6, 5, True, 40)
        chart.set_axis_tick(lv.chart.AXIS.PRIMARY_X, 10, 5, 10, 1, True, 30)

        chart.add_event_cb(self.event_cb, lv.EVENT.ALL, None)
        chart.refresh_ext_draw_size()

        self.cursor = chart.add_cursor(lv.palette_main(lv.PALETTE.BLUE), lv.DIR.LEFT␣
↪| lv.DIR.BOTTOM)
```

(下页继续)

(续上页)

```
        self.ser = chart.add_series(lv.palette_main(lv.PALETTE.RED), lv.chart.AXIS.
↪PRIMARY_Y)

        self.ser_p = []
        for i in range(10):
            self.ser_p.append(lv.rand(10,90))
        self.ser.y_points = self.ser_p

        newser = chart.get_series_next(None)
        # print("length of data points: ",len(newser.points))
        chart.set_zoom_x(500)

        label = lv.label(lv.scr_act())
        label.set_text("Click on a point")
        label.align_to(chart, lv.ALIGN.OUT_TOP_MID, 0, -5)


    def event_cb(self,e):

        code = e.get_code()
        chart = e.get_target()

        if code == lv.EVENT.VALUE_CHANGED:
            # print("last_id: ",self.last_id)
            self.last_id = chart.get_pressed_point()
            if self.last_id != lv.CHART_POINT.NONE:
                p = lv.point_t()
                chart.get_point_pos_by_id(self.ser, self.last_id, p)
                chart.set_cursor_point(self.cursor, None, self.last_id)

        elif code == lv.EVENT.DRAW_PART_END:
            # print("EVENT.DRAW_PART_END")
            dsc = lv.obj_draw_part_dsc_t.__cast__(e.get_param())
            # if dsc.p1 and dsc.p2:
                # print("p1, p2", dsc.p1,dsc.p2)
                # print("p1.y, p2.y", dsc.p1.y, dsc.p2.y)
                # print("last_id: ",self.last_id)
            if dsc.part == lv.PART.CURSOR and dsc.p1 and dsc.p2 and dsc.p1.y == dsc.
↪p2.y and self.last_id >= 0:

                v = self.ser_p[self.last_id]

                # print("value: ",v)
```

(下页继续)

(续上页)

```
                value_txt = str(v)
                size = lv.point_t()
                lv.txt_get_size(size, value_txt, lv.font_default(), 0, 0, lv.COORD.
↪MAX, lv.TEXT_FLAG.NONE)

                a = lv.area_t()
                a.y2 = dsc.p1.y - 5
                a.y1 = a.y2 - size.y - 10
                a.x1 = dsc.p1.x + 10
                a.x2 = a.x1 + size.x + 10

                draw_rect_dsc = lv.draw_rect_dsc_t()
                draw_rect_dsc.init()
                draw_rect_dsc.bg_color = lv.palette_main(lv.PALETTE.BLUE)
                draw_rect_dsc.radius = 3

                lv.draw_rect(a, dsc.clip_area, draw_rect_dsc)

                draw_label_dsc = lv.draw_label_dsc_t()
                draw_label_dsc.init()
                draw_label_dsc.color = lv.color_white()
                a.x1 += 5
                a.x2 -= 5
                a.y1 += 5
                a.y2 -= 5
                lv.draw_label(a, dsc.clip_area, draw_label_dsc, value_txt, None)

example_chart_6 = ExampleChart_6()
```

Scatter chart

```
#include "../../lv_examples.h"
#if LV_USE_CHART && LV_BUILD_EXAMPLES

static void draw_event_cb(lv_event_t * e)
{
    lv_obj_draw_part_dsc_t * dsc = lv_event_get_draw_part_dsc(e);
    if(dsc->part == LV_PART_ITEMS) {
        lv_obj_t * obj = lv_event_get_target(e);
        lv_chart_series_t * ser = lv_chart_get_series_next(obj, NULL);
        uint32_t cnt = lv_chart_get_point_count(obj);
        /*Make older value more transparent*/
```

(下页继续)

(续上页)

```
        dsc->rect_dsc->bg_opa = (LV_OPA_COVER *  dsc->id) / (cnt - 1);

        /*Make smaller values blue, higher values red*/
        lv_coord_t * x_array = lv_chart_get_x_array(obj, ser);
        lv_coord_t * y_array = lv_chart_get_y_array(obj, ser);
        /*dsc->id is the tells drawing order, but we need the ID of the point being␣
↪drawn.*/
        uint32_t start_point = lv_chart_get_x_start_point(obj, ser);
        uint32_t p_act = (start_point + dsc->id) % cnt; /*Consider start point to get␣
↪the index of the array*/
        lv_opa_t x_opa = (x_array[p_act] * LV_OPA_50) / 200;
        lv_opa_t y_opa = (y_array[p_act] * LV_OPA_50) / 1000;

        dsc->rect_dsc->bg_color = lv_color_mix(lv_palette_main(LV_PALETTE_RED),
                                               lv_palette_main(LV_PALETTE_BLUE),
                                               x_opa + y_opa);
    }
}

static void add_data(lv_timer_t * timer)
{
    LV_UNUSED(timer);
    lv_obj_t * chart = timer->user_data;
    lv_chart_set_next_value2(chart, lv_chart_get_series_next(chart, NULL), lv_rand(0,
↪200), lv_rand(0,1000));
}

/**
 * A scatter chart
 */
void lv_example_chart_7(void)
{
    lv_obj_t * chart = lv_chart_create(lv_scr_act());
    lv_obj_set_size(chart, 200, 150);
    lv_obj_align(chart, LV_ALIGN_CENTER, 0, 0);
    lv_obj_add_event_cb(chart, draw_event_cb, LV_EVENT_DRAW_PART_BEGIN, NULL);
    lv_obj_set_style_line_width(chart, 0, LV_PART_ITEMS);   /*Remove the lines*/

    lv_chart_set_type(chart, LV_CHART_TYPE_SCATTER);

    lv_chart_set_axis_tick(chart, LV_CHART_AXIS_PRIMARY_X, 5, 5, 5, 1, true, 30);
    lv_chart_set_axis_tick(chart, LV_CHART_AXIS_PRIMARY_Y, 10, 5, 6, 5, true, 50);
```

(下页继续)

(续上页)

```
    lv_chart_set_range(chart, LV_CHART_AXIS_PRIMARY_X, 0, 200);
    lv_chart_set_range(chart, LV_CHART_AXIS_PRIMARY_Y, 0, 1000);

    lv_chart_set_point_count(chart, 50);

    lv_chart_series_t * ser = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_
↪RED), LV_CHART_AXIS_PRIMARY_Y);
    uint32_t i;
    for(i = 0; i < 50; i++) {
        lv_chart_set_next_value2(chart, ser, lv_rand(0, 200), lv_rand(0, 1000));
    }

    lv_timer_create(add_data, 100, chart);
}

#endif
```

```
#!/opt/bin/lv_micropython -i
import utime as time
import lvgl as lv
import display_driver

def draw_event_cb(e):
    dsc = e.get_draw_part_dsc()
    if dsc.part == lv.PART.ITEMS:
        obj = e.get_target()
        ser = obj.get_series_next(None)
        cnt = obj.get_point_count()
        # print("cnt: ",cnt)
        # Make older value more transparent
        dsc.rect_dsc.bg_opa = (lv.OPA.COVER *  dsc.id) // (cnt - 1)

        # Make smaller values blue, higher values red
        # x_array = chart.get_x_array(ser)
        # y_array = chart.get_y_array(ser)
        # dsc->id is the tells drawing order, but we need the ID of the point being␣
↪drawn.
        start_point = chart.get_x_start_point(ser)
        # print("start point: ",start_point)
        p_act = (start_point + dsc.id) % cnt # Consider start point to get the index␣
↪of the array
        # print("p_act", p_act)
        x_opa = (x_array[p_act] * lv.OPA._50) // 200
```

(下页继续)

(续上页)

```
        y_opa = (y_array[p_act] * lv.OPA._50) // 1000

        dsc.rect_dsc.bg_color = lv.palette_main(lv.PALETTE.RED).color_mix(
                                            lv.palette_main(lv.PALETTE.BLUE),
                                            x_opa + y_opa)

def add_data(timer,chart):
    # print("add_data")
    x = lv.rand(0,200)
    y = lv.rand(0,1000)
    chart.set_next_value2(ser, x, y)
    # chart.set_next_value2(chart.gx, y)
    x_array.pop(0)
    x_array.append(x)
    y_array.pop(0)
    y_array.append(y)

#
# A scatter chart
#

chart = lv.chart(lv.scr_act())
chart.set_size(200, 150)
chart.align(lv.ALIGN.CENTER, 0, 0)
chart.add_event_cb(draw_event_cb, lv.EVENT.DRAW_PART_BEGIN, None)
chart.set_style_line_width(0, lv.PART.ITEMS)   # Remove the lines

chart.set_type(lv.chart.TYPE.SCATTER)

chart.set_axis_tick(lv.chart.AXIS.PRIMARY_X, 5, 5, 5, 1, True, 30)
chart.set_axis_tick(lv.chart.AXIS.PRIMARY_Y, 10, 5, 6, 5, True, 50)

chart.set_range(lv.chart.AXIS.PRIMARY_X, 0, 200)
chart.set_range(lv.chart.AXIS.PRIMARY_Y, 0, 1000)

chart.set_point_count(50)

ser = chart.add_series(lv.palette_main(lv.PALETTE.RED), lv.chart.AXIS.PRIMARY_Y)

x_array = []
y_array = []
for i in range(50):
    x_array.append(lv.rand(0, 200))
```

(下页继续)

(续上页)

```
    y_array.append(lv.rand(0, 1000))

ser.x_points = x_array
ser.y_points = y_array

# Create an `lv_timer` to update the chart.

timer = lv.timer_create_basic()
timer.set_period(100)
timer.set_cb(lambda src: add_data(timer,chart))
```

Stacked area chart

```
#include "../../lv_examples.h"
#if LV_USE_CHART && LV_DRAW_COMPLEX && LV_BUILD_EXAMPLES

/*  A struct is used to keep track of the series list because later we need to draw␣
↪to the series in the reverse order to which they were initialised. */
typedef struct
{
    lv_obj_t *obj;
    lv_chart_series_t *series_list[3];
} stacked_area_chart_t;

static stacked_area_chart_t stacked_area_chart;

/**
 * Callback which draws the blocks of colour under the lines
 **/
static void draw_event_cb(lv_event_t *e)
{
    lv_obj_t *obj = lv_event_get_target(e);

    /*Add the faded area before the lines are drawn*/
    lv_obj_draw_part_dsc_t *dsc = lv_event_get_draw_part_dsc(e);
    if (dsc->part == LV_PART_ITEMS)
    {
        if (!dsc->p1 || !dsc->p2)
            return;

        /*Add a line mask that keeps the area below the line*/
        lv_draw_mask_line_param_t line_mask_param;
```

(下页继续)

(续上页)

```
        lv_draw_mask_line_points_init(&line_mask_param, dsc->p1->x, dsc->p1->y, dsc->
↪p2->x, dsc->p2->y, LV_DRAW_MASK_LINE_SIDE_BOTTOM);
        int16_t line_mask_id = lv_draw_mask_add(&line_mask_param, NULL);

        /*Draw a rectangle that will be affected by the mask*/
        lv_draw_rect_dsc_t draw_rect_dsc;
        lv_draw_rect_dsc_init(&draw_rect_dsc);
        draw_rect_dsc.bg_opa = LV_OPA_COVER;
        draw_rect_dsc.bg_color = dsc->line_dsc->color;

        lv_area_t a;
        a.x1 = dsc->p1->x;
        a.x2 = dsc->p2->x;
        a.y1 = LV_MIN(dsc->p1->y, dsc->p2->y);
        a.y2 = obj->coords.y2 - 13; /* -13 cuts off where the rectangle draws over␣
↪the chart margin. Without this an area of 0 doesn't look like 0 */
        lv_draw_rect(dsc->draw_ctx, &draw_rect_dsc, &a);

        /*Remove the mask*/
        lv_draw_mask_free_param(&line_mask_param);
        lv_draw_mask_remove_id(line_mask_id);
    }
}

/**
 * Helper function to round a fixed point number
 **/
static int32_t round_fixed_point(int32_t n, int8_t shift)
{
    /* Create a bitmask to isolates the decimal part of the fixed point number */
    int32_t mask = 1;
    for (int32_t bit_pos = 0; bit_pos < shift; bit_pos++)
    {
        mask = (mask << 1) + 1;
    }

    int32_t decimal_part = n & mask;

    /* Get 0.5 as fixed point */
    int32_t rounding_boundary = 1 << (shift - 1);

    /* Return either the integer part of n or the integer part + 1 */
    return (decimal_part < rounding_boundary) ? (n & ~mask) : ((n >> shift) + 1) <<␣
↪shift;
```

(下页继续)

(续上页)

```
}

/**
 * Stacked area chart
 */
void lv_example_chart_8(void)
{
    /*Create a stacked_area_chart.obj*/
    stacked_area_chart.obj = lv_chart_create(lv_scr_act());
    lv_obj_set_size(stacked_area_chart.obj, 200, 150);
    lv_obj_center(stacked_area_chart.obj);
    lv_chart_set_type(stacked_area_chart.obj, LV_CHART_TYPE_LINE);
    lv_chart_set_div_line_count(stacked_area_chart.obj, 5, 7);
    lv_obj_add_event_cb(stacked_area_chart.obj, draw_event_cb, LV_EVENT_DRAW_PART_
↪BEGIN, NULL);

    /* Set range to 0 to 100 for percentages. Draw ticks */
    lv_chart_set_range(stacked_area_chart.obj,LV_CHART_AXIS_PRIMARY_Y,0,100);
    lv_chart_set_axis_tick(stacked_area_chart.obj, LV_CHART_AXIS_PRIMARY_Y, 3, 0, 5,␣
↪1, true, 30);

    /*Set point size to 0 so the lines are smooth */
    lv_obj_set_style_size(stacked_area_chart.obj, 0, LV_PART_INDICATOR);

    /*Add some data series*/
    stacked_area_chart.series_list[0] = lv_chart_add_series(stacked_area_chart.obj,␣
↪lv_palette_main(LV_PALETTE_RED), LV_CHART_AXIS_PRIMARY_Y);
    stacked_area_chart.series_list[1] = lv_chart_add_series(stacked_area_chart.obj,␣
↪lv_palette_main(LV_PALETTE_BLUE), LV_CHART_AXIS_PRIMARY_Y);
    stacked_area_chart.series_list[2] = lv_chart_add_series(stacked_area_chart.obj,␣
↪lv_palette_main(LV_PALETTE_GREEN), LV_CHART_AXIS_PRIMARY_Y);

    for (int point = 0; point < 10; point++)
    {
        /* Make some random data */
        uint32_t vals[3] = {lv_rand(10, 20), lv_rand(20, 30), lv_rand(20, 30)};

        int8_t fixed_point_shift = 5;
        uint32_t total = vals[0] + vals[1] + vals[2];
        uint32_t draw_heights[3];
        uint32_t int_sum = 0;
        uint32_t decimal_sum = 0;
```

(下页继续)

(续上页)

```
        /* Fixed point cascade rounding ensures percentages add to 100 */
        for (int32_t series_index = 0; series_index < 3; series_index++)
        {
            decimal_sum += (((vals[series_index] * 100) << fixed_point_shift) /␣
↪total);
            int_sum += (vals[series_index] * 100) / total;

            int32_t modifier = (round_fixed_point(decimal_sum, fixed_point_shift) >>␣
↪fixed_point_shift) - int_sum;

            /*  The draw heights are equal to the percentage of the total each value␣
↪is + the cumulative sum of the previous percentages.
                The accumulation is how the values get "stacked" */
            draw_heights[series_index] = int_sum + modifier;

            /*  Draw to the series in the reverse order to which they were␣
↪initialised.
                Without this the higher values will draw on top of the lower ones.
                This is because the Z-height of a series matches the order it was␣
↪initialised */
            lv_chart_set_next_value(stacked_area_chart.obj, stacked_area_chart.series_
↪list[3 - series_index - 1], draw_heights[series_index]);
        }
    }

    lv_chart_refresh(stacked_area_chart.obj);
}

#endif
```

```
import display_driver
import lvgl as lv

# A class is used to keep track of the series list because later we
#  need to draw to the series in the reverse order to which they were initialised.
class StackedAreaChart:
    def __init__(self):
        self.obj = None
        self.series_list = [None, None, None]

stacked_area_chart = StackedAreaChart()

#
```

(下页继续)

(续上页)

```
# Callback which draws the blocks of colour under the lines
#
def draw_event_cb(e):

    obj = e.get_target()
    cont_a = lv.area_t()
    obj.get_coords(cont_a)

    #Add the faded area before the lines are drawn
    dsc = e.get_draw_part_dsc()
    if dsc.part == lv.PART.ITEMS:
        if not dsc.p1 or not dsc.p2:
            return

        # Add a line mask that keeps the area below the line
        line_mask_param = lv.draw_mask_line_param_t()
        line_mask_param.points_init(dsc.p1.x, dsc.p1.y, dsc.p2.x, dsc.p2.y, lv.DRAW_
↪MASK_LINE_SIDE.BOTTOM)
        line_mask_id = lv.draw_mask_add(line_mask_param, None)

        #Draw a rectangle that will be affected by the mask
        draw_rect_dsc = lv.draw_rect_dsc_t()
        draw_rect_dsc.init()
        draw_rect_dsc.bg_opa = lv.OPA.COVER
        draw_rect_dsc.bg_color = dsc.line_dsc.color

        a = lv.area_t()
        a.x1 = dsc.p1.x
        a.x2 = dsc.p2.x
        a.y1 = min(dsc.p1.y, dsc.p2.y)
        a.y2 = cont_a.y2 - 13 # -13 cuts off where the rectangle draws over the chart␣
↪margin. Without this an area of 0 doesn't look like 0
        dsc.draw_ctx.rect(draw_rect_dsc, a)

        # Remove the mask
        lv.draw_mask_free_param(line_mask_param)
        lv.draw_mask_remove_id(line_mask_id)


#
# Helper function to round a fixed point number
#
def round_fixed_point(n, shift):
```

(下页继续)

(续上页)

```
    # Create a bitmask to isolates the decimal part of the fixed point number
    mask = 1
    for bit_pos in range(shift):
        mask = (mask << 1) + 1

    decimal_part = n & mask

    # Get 0.5 as fixed point
    rounding_boundary = 1 << (shift - 1)

    # Return either the integer part of n or the integer part + 1
    if decimal_part < rounding_boundary:
        return (n & ~mask)
    return ((n >> shift) + 1) << shift


#
# Stacked area chart
#
def lv_example_chart_8():

    #Create a stacked_area_chart.obj
    stacked_area_chart.obj = lv.chart(lv.scr_act())
    stacked_area_chart.obj.set_size(200, 150)
    stacked_area_chart.obj.center()
    stacked_area_chart.obj.set_type( lv.chart.TYPE.LINE)
    stacked_area_chart.obj.set_div_line_count(5, 7)
    stacked_area_chart.obj.add_event_cb( draw_event_cb, lv.EVENT.DRAW_PART_BEGIN,␣
↪None)

    # Set range to 0 to 100 for percentages. Draw ticks
    stacked_area_chart.obj.set_range(lv.chart.AXIS.PRIMARY_Y,0,100)
    stacked_area_chart.obj.set_axis_tick(lv.chart.AXIS.PRIMARY_Y, 3, 0, 5, 1, True,␣
↪30)

    #Set point size to 0 so the lines are smooth
    stacked_area_chart.obj.set_style_size(0, lv.PART.INDICATOR)

    # Add some data series
    stacked_area_chart.series_list[0] = stacked_area_chart.obj.add_series(lv.palette_
↪main(lv.PALETTE.RED), lv.chart.AXIS.PRIMARY_Y)
    stacked_area_chart.series_list[1] = stacked_area_chart.obj.add_series(lv.palette_
↪main(lv.PALETTE.BLUE), lv.chart.AXIS.PRIMARY_Y)
```

(下页继续)

(续上页)

```
    stacked_area_chart.series_list[2] = stacked_area_chart.obj.add_series(lv.palette_
↪main(lv.PALETTE.GREEN), lv.chart.AXIS.PRIMARY_Y)

    for point in range(10):
        # Make some random data
        vals = [lv.rand(10, 20), lv.rand(20, 30), lv.rand(20, 30)]

        fixed_point_shift = 5
        total = vals[0] + vals[1] + vals[2]
        draw_heights = [0, 0, 0]
        int_sum = 0
        decimal_sum = 0

        # Fixed point cascade rounding ensures percentages add to 100
        for series_index in range(3):
            decimal_sum += int(((vals[series_index] * 100) << fixed_point_shift) //␣
↪total)
            int_sum += int((vals[series_index] * 100) / total)

            modifier = (round_fixed_point(decimal_sum, fixed_point_shift) >> fixed_
↪point_shift) - int_sum

            #  The draw heights are equal to the percentage of the total each value␣
↪is + the cumulative sum of the previous percentages.
            #   The accumulation is how the values get "stacked"
            draw_heights[series_index] = int(int_sum + modifier)

            #  Draw to the series in the reverse order to which they were initialised.
            #   Without this the higher values will draw on top of the lower ones.
            #   This is because the Z-height of a series matches the order it was␣
↪initialised
            stacked_area_chart.obj.set_next_value( stacked_area_chart.series_list[3 -␣
↪series_index - 1], draw_heights[series_index])

    stacked_area_chart.obj.refresh()

lv_example_chart_8()
```

API

Typedefs

typedef uint8_t **lv_chart_type_t**

typedef uint8_t **lv_chart_update_mode_t**

typedef uint8_t **lv_chart_axis_t**

Enums

enum **[anonymous]**
Chart types

Values:

enumerator **LV_CHART_TYPE_NONE**
Don't draw the series

enumerator **LV_CHART_TYPE_LINE**
Connect the points with lines

enumerator **LV_CHART_TYPE_BAR**
Draw columns

enumerator **LV_CHART_TYPE_SCATTER**
Draw points and lines in 2D (x,y coordinates)

enum **[anonymous]**
Chart update mode for `lv_chart_set_next`

Values:

enumerator **LV_CHART_UPDATE_MODE_SHIFT**
Shift old data to the left and add the new one the right

enumerator **LV_CHART_UPDATE_MODE_CIRCULAR**
Add the new data in a circular way

enum **[anonymous]**
Enumeration of the axis'

Values:

enumerator **LV_CHART_AXIS_PRIMARY_Y**

enumerator **LV_CHART_AXIS_SECONDARY_Y**

enumerator **LV_CHART_AXIS_PRIMARY_X**

enumerator **LV_CHART_AXIS_SECONDARY_X**

enumerator **_LV_CHART_AXIS_LAST**

enum **lv_chart_draw_part_type_t**

`type` field in `lv_obj_draw_part_dsc_t` if `class_p = lv_chart_class` Used in `LV_EVENT_DRAW_PART_BEGIN` and `LV_EVENT_DRAW_PART_END`

Values:

enumerator **LV_CHART_DRAW_PART_DIV_LINE_INIT**

Used before/after drawn the div lines

enumerator **LV_CHART_DRAW_PART_DIV_LINE_HOR**

Used for each horizontal division lines

enumerator **LV_CHART_DRAW_PART_DIV_LINE_VER**

Used for each vertical division lines

enumerator **LV_CHART_DRAW_PART_LINE_AND_POINT**

Used on line and scatter charts for lines and points

enumerator **LV_CHART_DRAW_PART_BAR**

Used on bar charts for the rectangles

enumerator **LV_CHART_DRAW_PART_CURSOR**

Used on cursor lines and points

enumerator **LV_CHART_DRAW_PART_TICK_LABEL**

Used on tick lines and labels

Functions

LV_EXPORT_CONST_INT(LV_CHART_POINT_NONE)

lv_obj_t ***lv_chart_create**(*lv_obj_t* *parent)

Create a chart object

参数 parent -- pointer to an object, it will be the parent of the new chart

返回 pointer to the created chart

void **lv_chart_set_type**(*lv_obj_t* *obj, *lv_chart_type_t* type)

Set a new type for a chart

参数

- **obj** -- pointer to a chart object
- **type** -- new type of the chart (from 'lv_chart_type_t' enum)

void **lv_chart_set_point_count**(*lv_obj_t* *obj, uint16_t cnt)

Set the number of points on a data line on a chart

参数

- **obj** -- pointer to a chart object
- **cnt** -- new number of points on the data lines

void **lv_chart_set_range**(*lv_obj_t* *obj, *lv_chart_axis_t* axis, lv_coord_t min, lv_coord_t max)

Set the minimal and maximal y values on an axis

参数

- **obj** -- pointer to a chart object
- **axis** -- `LV_CHART_AXIS_PRIMARY_Y` or `LV_CHART_AXIS_SECONDARY_Y`
- **min** -- minimum value of the y axis
- **max** -- maximum value of the y axis

void **lv_chart_set_update_mode**(*lv_obj_t* *obj, *lv_chart_update_mode_t* update_mode)

Set update mode of the chart object. Affects

参数

- **obj** -- pointer to a chart object
- **mode** -- the update mode

void **lv_chart_set_div_line_count**(*lv_obj_t* *obj, uint8_t hdiv, uint8_t vdiv)

Set the number of horizontal and vertical division lines

参数

- **obj** -- pointer to a chart object
- **hdiv** -- number of horizontal division lines
- **vdiv** -- number of vertical division lines

void **lv_chart_set_zoom_x**(*lv_obj_t* *obj, uint16_t zoom_x)

Zoom into the chart in X direction

参数

- **obj** -- pointer to a chart object
- **zoom_x** -- zoom in x direction. LV_ZOOM_NONE or 256 for no zoom, 512 double zoom

void **lv_chart_set_zoom_y**(*lv_obj_t* *obj, uint16_t zoom_y)

Zoom into the chart in Y direction

参数

- **obj** -- pointer to a chart object
- **zoom_y** -- zoom in y direction. LV_ZOOM_NONE or 256 for no zoom, 512 double zoom

uint16_t **lv_chart_get_zoom_x**(const *lv_obj_t* *obj)

Get X zoom of a chart

参数 **obj** -- pointer to a chart object

返回 the X zoom value

uint16_t **lv_chart_get_zoom_y**(const *lv_obj_t* *obj)

Get Y zoom of a chart

参数 **obj** -- pointer to a chart object

返回 the Y zoom value

void **lv_chart_set_axis_tick**(*lv_obj_t* *obj, *lv_chart_axis_t* axis, lv_coord_t major_len, lv_coord_t minor_len, lv_coord_t major_cnt, lv_coord_t minor_cnt, bool label_en, lv_coord_t draw_size)

Set the number of tick lines on an axis

参数

- **obj** -- pointer to a chart object
- **axis** -- an axis which ticks count should be set
- **major_len** -- length of major ticks
- **minor_len** -- length of minor ticks
- **major_cnt** -- number of major ticks on the axis
- **minor_cnt** -- number of minor ticks between two major ticks
- **label_en** -- true: enable label drawing on major ticks
- **draw_size** -- extra size required to draw the tick and labels (start with 20 px and increase if the ticks/labels are clipped)

lv_chart_type_t **lv_chart_get_type**(const *lv_obj_t* *obj)

Get the type of a chart

参数 **obj** -- pointer to chart object

返回 type of the chart (from '*lv_chart_t*' enum)

uint16_t **lv_chart_get_point_count**(const *lv_obj_t* *obj)

Get the data point number per data line on chart

参数 **chart** -- pointer to chart object

返回 point number on each data line

uint16_t **lv_chart_get_x_start_point**(const *lv_obj_t* *obj, *lv_chart_series_t* *ser)

Get the current index of the x-axis start point in the data array

参数

- **chart** -- pointer to a chart object
- **ser** -- pointer to a data series on 'chart'

返回 the index of the current x start point in the data array

void **lv_chart_get_point_pos_by_id**(*lv_obj_t* *obj, *lv_chart_series_t* *ser, uint16_t id, lv_point_t *p_out)

Get the position of a point to the chart.

参数

- **chart** -- pointer to a chart object
- **ser** -- pointer to series
- **id** -- the index.
- **p_out** -- store the result position here

void **lv_chart_refresh**(*lv_obj_t* *obj)

Refresh a chart if its data line has changed

参数 **chart** -- pointer to chart object

lv_chart_series_t ***lv_chart_add_series**(*lv_obj_t* *obj, lv_color_t color, *lv_chart_axis_t* axis)

Allocate and add a data series to the chart

参数

- **obj** -- pointer to a chart object
- **color** -- color of the data series
- **axis** -- the y axis to which the series should be attached (::LV_CHART_AXIS_PRIMARY_Y or ::LV_CHART_AXIS_SECONDARY_Y)

返回 pointer to the allocated data series

void **lv_chart_remove_series**(*lv_obj_t* *obj, *lv_chart_series_t* *series)

Deallocate and remove a data series from a chart

参数

- **chart** -- pointer to a chart object
- **series** -- pointer to a data series on 'chart'

void **lv_chart_hide_series**(*lv_obj_t* *chart, *lv_chart_series_t* *series, bool hide)

Hide/Unhide a single series of a chart.

参数

- **obj** -- pointer to a chart object.
- **series** -- pointer to a series object
- **hide** -- true: hide the series

void **lv_chart_set_series_color**(*lv_obj_t* *chart, *lv_chart_series_t* *series, lv_color_t color)

Change the color of a series

参数

- **obj** -- pointer to a chart object.
- **series** -- pointer to a series object
- **color** -- the new color of the series

void **lv_chart_set_x_start_point**(*lv_obj_t* *obj, *lv_chart_series_t* *ser, uint16_t id)

Set the index of the x-axis start point in the data array. This point will be considers the first (left) point and the other points will be drawn after it.

参数

- **obj** -- pointer to a chart object
- **ser** -- pointer to a data series on 'chart'
- **id** -- the index of the x point in the data array

lv_chart_series_t ***lv_chart_get_series_next**(const *lv_obj_t* *chart, const *lv_chart_series_t* *ser)

Get the next series.

参数

- **chart** -- pointer to a chart
- **ser** -- the previous series or NULL to get the first

返回 the next series or NULL if there is no more.

lv_chart_cursor_t ***lv_chart_add_cursor**(*lv_obj_t* *obj, lv_color_t color, lv_dir_t dir)

Add a cursor with a given color

参数

- **obj** -- pointer to chart object
- **color** -- color of the cursor

- **dir** -- direction of the cursor. LV_DIR_RIGHT/LEFT/TOP/DOWN/HOR/VER/ALL. OR-ed values are possible

返回 pointer to the created cursor

void **lv_chart_set_cursor_pos**(*lv_obj_t* *chart, *lv_chart_cursor_t* *cursor, lv_point_t *pos)

Set the coordinate of the cursor with respect to the paddings

参数

- **obj** -- pointer to a chart object
- **cursor** -- pointer to the cursor
- **pos** -- the new coordinate of cursor relative to the chart

void **lv_chart_set_cursor_point**(*lv_obj_t* *chart, *lv_chart_cursor_t* *cursor, *lv_chart_series_t* *ser, uint16_t point_id)

Stick the cursor to a point

参数

- **obj** -- pointer to a chart object
- **cursor** -- pointer to the cursor
- **ser** -- pointer to a series
- **point_id** -- the point's index or LV_CHART_POINT_NONE to not assign to any points.

lv_point_t **lv_chart_get_cursor_point**(*lv_obj_t* *chart, *lv_chart_cursor_t* *cursor)

Get the coordinate of the cursor with respect to the paddings

参数

- **obj** -- pointer to a chart object
- **cursor** -- pointer to cursor

返回 coordinate of the cursor as lv_point_t

void **lv_chart_set_all_value**(*lv_obj_t* *obj, *lv_chart_series_t* *ser, lv_coord_t value)

Initialize all data points of a series with a value

参数

- **obj** -- pointer to chart object
- **ser** -- pointer to a data series on 'chart'
- **value** -- the new value for all points. LV_CHART_POINT_NONE can be used to hide the points.

void **lv_chart_set_next_value**(*lv_obj_t* *obj, *lv_chart_series_t* *ser, lv_coord_t value)

Set the next point's Y value according to the update mode policy.

参数

- **obj** -- pointer to chart object
- **ser** -- pointer to a data series on 'chart'
- **value** -- the new value of the next data

void **lv_chart_set_next_value2**(*lv_obj_t* *obj, *lv_chart_series_t* *ser, lv_coord_t x_value, lv_coord_t y_value)

Set the next point's X and Y value according to the update mode policy.

参数

- **obj** -- pointer to chart object
- **ser** -- pointer to a data series on 'chart'
- **x_value** -- the new X value of the next data
- **y_value** -- the new Y value of the next data

void **lv_chart_set_value_by_id**(*lv_obj_t* *obj, *lv_chart_series_t* *ser, uint16_t id, lv_coord_t value)

Set an individual point's y value of a chart's series directly based on its index

参数

- **obj** -- pointer to a chart object
- **ser** -- pointer to a data series on 'chart'
- **id** -- the index of the x point in the array
- **value** -- value to assign to array point

void **lv_chart_set_value_by_id2**(*lv_obj_t* *obj, *lv_chart_series_t* *ser, uint16_t id, lv_coord_t x_value, lv_coord_t y_value)

Set an individual point's x and y value of a chart's series directly based on its index Can be used only with `LV_CHART_TYPE_SCATTER`.

参数

- **obj** -- pointer to chart object
- **ser** -- pointer to a data series on 'chart'
- **id** -- the index of the x point in the array
- **x_value** -- the new X value of the next data
- **y_value** -- the new Y value of the next data

void **lv_chart_set_ext_y_array**(*lv_obj_t* *obj, *lv_chart_series_t* *ser, lv_coord_t array[])

Set an external array for the y data points to use for the chart NOTE: It is the users responsibility to make sure the `point_cnt` matches the external array size.

参数

- **obj** -- pointer to a chart object
- **ser** -- pointer to a data series on 'chart'
- **array** -- external array of points for chart

void **lv_chart_set_ext_x_array**(*lv_obj_t* *obj, *lv_chart_series_t* *ser, lv_coord_t array[])

Set an external array for the x data points to use for the chart NOTE: It is the users responsibility to make sure the `point_cnt` matches the external array size.

参数

- **obj** -- pointer to a chart object
- **ser** -- pointer to a data series on 'chart'
- **array** -- external array of points for chart

lv_coord_t ***lv_chart_get_y_array**(const *lv_obj_t* *obj, *lv_chart_series_t* *ser)

Get the array of y values of a series

参数

- **obj** -- pointer to a chart object
- **ser** -- pointer to a data series on 'chart'

返回 the array of values with 'point_count' elements

lv_coord_t ***lv_chart_get_x_array**(const *lv_obj_t* *obj, *lv_chart_series_t* *ser)

Get the array of x values of a series

参数

- **obj** -- pointer to a chart object
- **ser** -- pointer to a data series on 'chart'

返回 the array of values with 'point_count' elements

uint32_t **lv_chart_get_pressed_point**(const *lv_obj_t* *obj)

Get the index of the currently pressed point. It's the same for every series.

参数 **obj** -- pointer to a chart object

返回 the index of the point [0 .. point count] or LV_CHART_POINT_ID_NONE if no point is being pressed

Variables

const lv_obj_class_t **lv_chart_class**

struct **lv_chart_series_t**

#include <lv_chart.h> Descriptor a chart series

Public Members

lv_coord_t ***x_points**

lv_coord_t ***y_points**

lv_color_t **color**

uint16_t **start_point**

uint8_t **hidden**

uint8_t **x_ext_buf_assigned**

uint8_t **y_ext_buf_assigned**

uint8_t **x_axis_sec**

uint8_t **y_axis_sec**

struct **lv_chart_cursor_t**

Public Members

lv_point_t **pos**

uint16_t **point_id**

lv_color_t **color**

lv_chart_series_t ***ser**

lv_dir_t **dir**

uint8_t **pos_set**

struct **lv_chart_tick_dsc_t**

Public Members

lv_coord_t **major_len**

lv_coord_t **minor_len**

lv_coord_t **draw_size**

uint32_t **minor_cnt**

uint32_t **major_cnt**

uint32_t **label_en**

struct **lv_chart_t**

Public Members

lv_obj_t **obj**

lv_ll_t **series_ll**
Linked list for the series (stores *lv_chart_series_t*)

lv_ll_t **cursor_ll**
Linked list for the cursors (stores *lv_chart_cursor_t*)

lv_chart_tick_dsc_t **tick**[4]

lv_coord_t **ymin**[2]

lv_coord_t **ymax**[2]

lv_coord_t **xmin**[2]

lv_coord_t **xmax**[2]

uint16_t **pressed_point_id**

uint16_t **hdiv_cnt**
Number of horizontal division lines

uint16_t **vdiv_cnt**
Number of vertical division lines

uint16_t **point_cnt**
Point number in a data line

uint16_t **zoom_x**

uint16_t **zoom_y**

lv_chart_type_t **type**
Line or column chart

lv_chart_update_mode_t **update_mode**

Color wheel (lv_colorwheel)

Overview

As its name implies *Color wheel* allows the user to select a color. The Hue, Saturation and Value of the color can be selected separately.

Long pressing the object, the color wheel will change to the next parameter of the color (hue, saturation or value). A double click will reset the current parameter.

Parts and Styles

- `LV_PART_MAIN` Only `arc_width` is used to set the width of the color wheel
- `LV_PART_KNOB` A rectangle (or circle) drawn on the current value. It uses all the rectangle like style properties and padding to make it larger than the width of the arc.

Usage

Create a color wheel

`lv_colorwheel_create(parent, knob_recolor)` creates a new color wheel. With `knob_recolor=true` the knob's background color will be set to the current color.

Set color

The color can be set manually with `lv_colorwheel_set_hue/saturation/value(colorwheel, x)` or all at once with `lv_colorwheel_set_hsv(colorwheel, hsv)` or `lv_colorwheel_set_color(colorwheel, rgb)`

Color mode

The current color mode can be manually selected with `lv_colorwheel_set_mode(colorwheel, LV_COLORWHEEL_MODE_HUE/SATURATION/VALUE)`.

The color mode can be fixed (so as to not change with long press) using `lv_colorwheel_set_mode_fixed(colorwheel, true)`

Events

- `LV_EVENT_VALUE_CHANGED` Sent if a new color is selected.

Learn more about *Events*.

Keys

- `LV_KEY_UP`, `LV_KEY_RIGHT` Increment the current parameter's value by 1
- `LV_KEY_DOWN`, `LV_KEY_LEFT` Decrement the current parameter's value by 1
- `LV_KEY_ENTER` A long press will show the next mode. Double click to reset the current parameter.

Learn more about *Keys*.

Example

Simple Colorwheel

```
#include "../../lv_examples.h"
#if LV_USE_COLORWHEEL && LV_BUILD_EXAMPLES

void lv_example_colorwheel_1(void)
{
    lv_obj_t * cw;

    cw = lv_colorwheel_create(lv_scr_act(), true);
    lv_obj_set_size(cw, 200, 200);
    lv_obj_center(cw);
}

#endif
```

```
cw = lv.colorwheel(lv.scr_act(), True)
cw.set_size(200, 200)
cw.center()
```

API

Typedefs

typedef uint8_t **lv_colorwheel_mode_t**

Enums

enum **[anonymous]**

Values:

enumerator **LV_COLORWHEEL_MODE_HUE**

enumerator **LV_COLORWHEEL_MODE_SATURATION**

enumerator **LV_COLORWHEEL_MODE_VALUE**

Functions

lv_obj_t ***lv_colorwheel_create**(*lv_obj_t* *parent, bool knob_recolor)

Create a color picker object with disc shape

参数

- **parent** -- pointer to an object, it will be the parent of the new color picker
- **knob_recolor** -- true: set the knob's color to the current color

返回 pointer to the created color picker

bool **lv_colorwheel_set_hsv**(*lv_obj_t* *obj, *lv_color_hsv_t* hsv)

Set the current hsv of a color wheel.

参数

- **colorwheel** -- pointer to color wheel object
- **color** -- current selected hsv

返回 true if changed, otherwise false

bool **lv_colorwheel_set_rgb**(*lv_obj_t* *obj, lv_color_t color)

Set the current color of a color wheel.

参数

- **colorwheel** -- pointer to color wheel object
- **color** -- current selected color

返回 true if changed, otherwise false

void **lv_colorwheel_set_mode**(*lv_obj_t* *obj, *lv_colorwheel_mode_t* mode)

Set the current color mode.

参数

- **colorwheel** -- pointer to color wheel object
- **mode** -- color mode (hue/sat/val)

void **lv_colorwheel_set_mode_fixed**(*lv_obj_t* *obj, bool fixed)

Set if the color mode is changed on long press on center

参数

- **colorwheel** -- pointer to color wheel object
- **fixed** -- color mode cannot be changed on long press

lv_color_hsv_t **lv_colorwheel_get_hsv**(*lv_obj_t* *obj)

Get the current selected hsv of a color wheel.

参数 **colorwheel** -- pointer to color wheel object

返回 current selected hsv

lv_color_t **lv_colorwheel_get_rgb**(*lv_obj_t* *obj)

Get the current selected color of a color wheel.

参数 **colorwheel** -- pointer to color wheel object

返回 color current selected color

lv_colorwheel_mode_t **lv_colorwheel_get_color_mode**(*lv_obj_t* *obj)

Get the current color mode.

参数 **colorwheel** -- pointer to color wheel object

返回 color mode (hue/sat/val)

bool **lv_colorwheel_get_color_mode_fixed**(*lv_obj_t* *obj)

Get if the color mode is changed on long press on center

参数 **colorwheel** -- pointer to color wheel object

返回 mode cannot be changed on long press

Variables

const lv_obj_class_t **lv_colorwheel_class**

struct **lv_colorwheel_t**

Public Members

lv_obj_t **obj**

lv_color_hsv_t **hsv**

lv_point_t **pos**

uint8_t **recolor**

struct *lv_colorwheel_t*::[anonymous] **knob**

uint32_t **last_click_time**

uint32_t **last_change_time**

lv_point_t **last_press_point**

lv_colorwheel_mode_t **mode**

uint8_t **mode_fixed**

Image button (lv_imgbtn)

Overview

The Image button is very similar to the simple 'Button' object. The only difference is that it displays user-defined images in each state instead of drawing a rectangle.

You can set a left, right and center image, and the center image will be repeated to match the width of the object.

Parts and Styles

- `LV_PART_MAIN` Refers to the image(s). If background style properties are used, a rectangle will be drawn behind the image button.

Usage

Image sources

To set the image in a state, use the `lv_imgbtn_set_src(imgbtn, LV_IMGBTN_STATE_..., src_left, src_center, src_right)`.

The image sources work the same as described in the *Image object* except that "Symbols" are not supported by the Image button. Any of the sources can `NULL`.

The possible states are:

- `LV_IMGBTN_STATE_RELEASED`
- `LV_IMGBTN_STATE_PRESSED`
- `LV_IMGBTN_STATE_DISABLED`
- `LV_IMGBTN_STATE_CHECKED_RELEASED`
- `LV_IMGBTN_STATE_CHECKED_PRESSED`
- `LV_IMGBTN_STATE_CHECKED_DISABLED`

If you set sources only in `LV_IMGBTN_STATE_RELEASED`, these sources will be used in other states too. If you set e.g. `LV_IMGBTN_STATE_PRESSED` they will be used in pressed state instead of the released images.

States

Instead of the regular `lv_obj_add/clear_state()` functions the `lv_imgbtn_set_state(imgbtn, LV_IMGBTN_STATE_...)` functions should be used to manually set a state.

Events

- `LV_EVENT_VALUE_CHANGED` Sent when the button is toggled.

Learn more about *Events*.

Keys

- `LV_KEY_RIGHT/UP` Go to toggled state if `LV_OBJ_FLAG_CHECKABLE` is enabled.
- `LV_KEY_LEFT/DOWN` Go to non-toggled state if `LV_OBJ_FLAG_CHECKABLE` is enabled.
- `LV_KEY_ENTER` Clicks the button

Learn more about *Keys*.

Example

Simple Image button

```
#include "../../lv_examples.h"
#if LV_USE_IMGBTN && LV_BUILD_EXAMPLES

void lv_example_imgbtn_1(void)
{
    LV_IMG_DECLARE(imgbtn_left);
    LV_IMG_DECLARE(imgbtn_right);
    LV_IMG_DECLARE(imgbtn_mid);

    /*Create a transition animation on width transformation and recolor.*/
    static lv_style_prop_t tr_prop[] = {LV_STYLE_TRANSFORM_WIDTH, LV_STYLE_IMG_
↪RECOLOR_OPA, 0};
    static lv_style_transition_dsc_t tr;
    lv_style_transition_dsc_init(&tr, tr_prop, lv_anim_path_linear, 200, 0, NULL);

    static lv_style_t style_def;
    lv_style_init(&style_def);
    lv_style_set_text_color(&style_def, lv_color_white());
    lv_style_set_transition(&style_def, &tr);

    /*Darken the button when pressed and make it wider*/
    static lv_style_t style_pr;
    lv_style_init(&style_pr);
    lv_style_set_img_recolor_opa(&style_pr, LV_OPA_30);
    lv_style_set_img_recolor(&style_pr, lv_color_black());
    lv_style_set_transform_width(&style_pr, 20);

    /*Create an image button*/
    lv_obj_t * imgbtn1 = lv_imgbtn_create(lv_scr_act());
    lv_imgbtn_set_src(imgbtn1, LV_IMGBTN_STATE_RELEASED, &imgbtn_left, &imgbtn_mid, &
↪imgbtn_right);
    lv_obj_add_style(imgbtn1, &style_def, 0);
    lv_obj_add_style(imgbtn1, &style_pr, LV_STATE_PRESSED);

    lv_obj_align(imgbtn1, LV_ALIGN_CENTER, 0, 0);

    /*Create a label on the image button*/
    lv_obj_t * label = lv_label_create(imgbtn1);
    lv_label_set_text(label, "Button");
    lv_obj_align(label, LV_ALIGN_CENTER, 0, -4);
```

(下页继续)

(续上页)

```
}

#endif
```

```
from imagetools import get_png_info, open_png

# Register PNG image decoder
decoder = lv.img.decoder_create()
decoder.info_cb = get_png_info
decoder.open_cb = open_png

# Create an image from the png file
try:
    with open('../../assets/imgbtn_left.png','rb') as f:
        imgbtn_left_data = f.read()
except:
    print("Could not find imgbtn_left.png")
    sys.exit()

imgbtn_left_dsc = lv.img_dsc_t({
  'data_size': len(imgbtn_left_data),
  'data': imgbtn_left_data
})

try:
    with open('../../assets/imgbtn_mid.png','rb') as f:
        imgbtn_mid_data = f.read()
except:
    print("Could not find imgbtn_mid.png")
    sys.exit()

imgbtn_mid_dsc = lv.img_dsc_t({
  'data_size': len(imgbtn_mid_data),
  'data': imgbtn_mid_data
})

try:
    with open('../../assets/imgbtn_right.png','rb') as f:
        imgbtn_right_data = f.read()
except:
    print("Could not find imgbtn_right.png")
    sys.exit()
```

(下页继续)

(续上页)

```
imgbtn_right_dsc = lv.img_dsc_t({
  'data_size': len(imgbtn_right_data),
  'data': imgbtn_right_data
})

# Create a transition animation on width transformation and recolor.
tr_prop = [lv.STYLE.TRANSFORM_WIDTH, lv.STYLE.IMG_RECOLOR_OPA, 0]
tr = lv.style_transition_dsc_t()
tr.init(tr_prop, lv.anim_t.path_linear, 200, 0, None)

style_def = lv.style_t()
style_def.init()
style_def.set_text_color(lv.color_white())
style_def.set_transition(tr)

# Darken the button when pressed and make it wider
style_pr = lv.style_t()
style_pr.init()
style_pr.set_img_recolor_opa(lv.OPA._30)
style_pr.set_img_recolor(lv.color_black())
style_pr.set_transform_width(20)

# Create an image button
imgbtn1 = lv.imgbtn(lv.scr_act())
imgbtn1.set_src(lv.imgbtn.STATE.RELEASED, imgbtn_left_dsc, imgbtn_mid_dsc, imgbtn_
↪right_dsc)
imgbtn1.add_style(style_def, 0)
imgbtn1.add_style(style_pr, lv.STATE.PRESSED)

imgbtn1.align(lv.ALIGN.CENTER, 0, 0)

# Create a label on the image button
label = lv.label(imgbtn1)
label.set_text("Button")
label.align(lv.ALIGN.CENTER, 0, -4)
```

API

Enums

enum **lv_imgbtn_state_t**
Values:

enumerator **LV_IMGBTN_STATE_RELEASED**

enumerator **LV_IMGBTN_STATE_PRESSED**

enumerator **LV_IMGBTN_STATE_DISABLED**

enumerator **LV_IMGBTN_STATE_CHECKED_RELEASED**

enumerator **LV_IMGBTN_STATE_CHECKED_PRESSED**

enumerator **LV_IMGBTN_STATE_CHECKED_DISABLED**

enumerator **_LV_IMGBTN_STATE_NUM**

Functions

lv_obj_t ***lv_imgbtn_create**(*lv_obj_t* *parent)
Create an image button object

参数 **parent** -- pointer to an object, it will be the parent of the new image button

返回 pointer to the created image button

void **lv_imgbtn_set_src**(*lv_obj_t* *imgbtn, *lv_imgbtn_state_t* state, const void *src_left, const void *src_mid, const void *src_right)
Set images for a state of the image button

参数

- **imgbtn** -- pointer to an image button object
- **state** -- for which state set the new image
- **src_left** -- pointer to an image source for the left side of the button (a C array or path to a file)
- **src_mid** -- pointer to an image source for the middle of the button (ideally 1px wide) (a C array or path to a file)
- **src_right** -- pointer to an image source for the right side of the button (a C array or path to a file)

void **lv_imgbtn_set_state**(*lv_obj_t* *imgbtn, *lv_imgbtn_state_t* state)
Use this function instead of `lv_obj_add/clear_state` to set a state manually

参数

- **imgbtn** -- pointer to an image button object
- **state** -- the new state

const void ***lv_imgbtn_get_src_left**(*lv_obj_t* *imgbtn, *lv_imgbtn_state_t* state)*

Get the left image in a given state

参数

- **imgbtn** -- pointer to an image button object
- **state** -- the state where to get the image (from `lv_btn_state_t`) ‘

返回 pointer to the left image source (a C array or path to a file)

const void ***lv_imgbtn_get_src_middle**(*lv_obj_t* *imgbtn, *lv_imgbtn_state_t* state)*

Get the middle image in a given state

参数

- **imgbtn** -- pointer to an image button object
- **state** -- the state where to get the image (from `lv_btn_state_t`) ‘

返回 pointer to the middle image source (a C array or path to a file)

const void ***lv_imgbtn_get_src_right**(*lv_obj_t* *imgbtn, *lv_imgbtn_state_t* state)*

Get the right image in a given state

参数

- **imgbtn** -- pointer to an image button object
- **state** -- the state where to get the image (from `lv_btn_state_t`) ‘

返回 pointer to the left image source (a C array or path to a file)

Variables

const lv_obj_class_t **lv_imgbtn_class**

struct **lv_imgbtn_t**

Public Members

lv_obj_t **obj**

const void ***img_src_mid**[*_LV_IMGBTN_STATE_NUM*]

const void ***img_src_left**[*_LV_IMGBTN_STATE_NUM*]

const void ***img_src_right**[*_LV_IMGBTN_STATE_NUM*]

lv_img_cf_t **act_cf**

Keyboard（键盘）(lv_keyboard)

Overview（概述）

The Keyboard object is a special *Button matrix* with predefined keymaps and other features to realize a virtual keyboard to write texts into a *Text area*.

键盘 (lv_keyboard) 对象是一个特殊的按钮矩阵 *(lv_btnmatrix)*，自身实现了按键 (map) 映射和其他功能，目的是用于实现一个虚拟键盘将文本写入文本框 *(lv_textarea)*。

Parts and Styles（部分和样式）

Similarly to Button matrices Keyboards consist of 2 part:

- `LV_PART_MAIN` The main part. Uses all the typical background properties
- `LV_PART_ITEMS` The buttons. Also uses all typical background properties as well as the *text* properties.

与按钮矩阵类似，键盘由 2 部分组成：

- `LV_PART_MAIN` 主要部分 (自身背景部分)，使用所有组件默认都有的典型的背景样式属性。
- `LV_PART_ITEMS` 键盘中的按钮。使用所有典型的背景属性以及 *text* 属性。

Usage（用法）

Modes（模式）

The Keyboards have the following modes:

- `LV_KEYBOARD_MODE_TEXT_LOWER` Display lower case letters
- `LV_KEYBOARD_MODE_TEXT_UPPER` Display upper case letters
- `LV_KEYBOARD_MODE_TEXT_SPECIAL` Display special characters
- `LV_KEYBOARD_MODE_NUMBER` Display numbers, +/- sign, and decimal dot

- LV_KEYBOARD_MODE_USER_1 through LV_KEYBOARD_MODE_USER_4 User-defined modes.

The TEXT modes' layout contains buttons to change mode.

To set the mode manually, use lv_keyboard_set_mode(kb, mode). The default mode is LV_KEYBOARD_MODE_TEXT_UPPER.

键盘可以切换下面这几种输入模式：

- LV_KEYBOARD_MODE_TEXT_LOWER 26 键英文小写键盘模式
- LV_KEYBOARD_MODE_TEXT_UPPER 26 键英文大写键盘模式
- LV_KEYBOARD_MODE_TEXT_SPECIAL 特殊字符输入模式
- LV_KEYBOARD_MODE_NUMBER 数字键盘模式。可以输入数字、+/- 符号和小数点
- LV_KEYBOARD_MODE_USER_1 到 LV_KEYBOARD_MODE_USER_4 用户可自定义拓展的模式 lvgl

更改模式会更改键盘的按钮的文字甚至布局。

您可以通过 lv_keyboard_set_mode(kb, mode) 函数手动设置模式。默认的模式是 LV_KEYBOARD_MODE_TEXT_UPPER 。

Assign Text area（指定文本框）

You can assign a *Text area* to the Keyboard to automatically put the clicked characters there. To assign the text area, use lv_keyboard_set_textarea(kb, ta).

你可以将文本框分配给键盘绑定，之后点击键盘上的按钮就能更改文本框中的内容。可以通过 lv_keyboard_set_textarea(kb, ta) 函数将文本框分配给键盘绑定。

Key Popovers（按键弹出提示）

To enable key popovers on press, like on common Android and iOS keyboards, use lv_keyboard_set_popovers(kb, true). The default control maps are preconfigured to only show the popovers on keys that produce a symbol and not on e.g. space. If you use a custom keymap, set the LV_BTNMATRIX_CTRL_POPOVER flag for all keys that you want to show a popover.

Note that popovers for keys in the top row will draw outside the widget boundaries. To account for this, reserve extra free space on top of the keyboard or ensure that the keyboard is added *after* any widgets adjacent to its top boundary so that the popovers can draw over those.

The popovers currently are merely a visual effect and don't allow selecting additional characters such as accents yet.

这个效果就像常见的安卓和 iOS 键盘上的效果：按下按键时会有在相应的按键之上弹出该按键提示当前按下的按钮。调用函数 lv_keyboard_set_popovers(kb, true) 即可得到这样得效果。默认控制 map 默认的配置是只在有 text 的按键上显示弹出提示框，而不会在空格键 (space) 上显示。如果使用自定义的按键 map，请为所有要显示弹出框的按键设置 LV_BTNMATRIX_CTRL_POPOVER 标志。

请注意，顶行中的按键的弹出窗口将被绘制在超过键盘的边界之外。因此，建议在键盘顶部保留额外的可用空间，或确保在其顶部边界附近的任何其他组件 (对象) 之后再添加或者创建键盘，来确保弹出窗口不被这些组件遮挡。

目前，弹出窗口仅仅是一种视觉效果，还不支持使用其他字符，例如重音符号等。

New Keymap（自定义键盘布局）

You can specify a new map (layout) for the keyboard with `lv_keyboard_set_map(kb, map)` and `lv_keyboard_set_ctrl_map(kb, ctrl_map)`. Learn more about the *Button matrix* object. Keep in mind that using following keywords will have the same effect as with the original map:

- `LV_SYMBOL_OK` Apply.
- `LV_SYMBOL_CLOSE` or `LV_SYMBOL_KEYBOARD` Close.
- `LV_SYMBOL_BACKSPACE` Delete on the left.
- `LV_SYMBOL_LEFT` Move the cursor left.
- `LV_SYMBOL_RIGHT` Move the cursor right.
- `LV_SYMBOL_NEW_LINE` New line.
- *"ABC"* Load the uppercase map.
- *"abc"* Load the lower case map.
- *"1#"* Load the lower case map.

您可以使用 `lv_keyboard_set_map(kb, map)` 和 `lv_keyboard_set_ctrl_map(kb, ctrl_map)` 函数为键盘指定新的 map（布局）。

了解有关按钮矩阵 对象的更多信息。

请记住，使用以下关键字可以得到与内置 map(布局) 相同的效果：

- `LV_SYMBOL_OK` 应用
- `LV_SYMBOL_CLOSE` 或者 `LV_SYMBOL_KEYBOARD` 关闭。
- `LV_SYMBOL_BACKSPACE` 删除光标左侧的字符。
- `LV_SYMBOL_LEFT` 向左移动光标。
- `LV_SYMBOL_RIGHT` 向右移动光标。
- `LV_SYMBOL_NEW_LINE` 换行。
- *"ABC"* 切换为 26 键英文大写键盘布局。
- *"abc"* 切换为 26 键英文大写键盘布局。
- *"1#"* 切换为特殊字符键盘布局

Events（事件）

- `LV_EVENT_VALUE_CHANGED` Sent when the button is pressed/released or repeated after long press. The event data is set to the ID of the pressed/released button.
- `LV_EVENT_READY` - The *Ok* button is clicked.
- `LV_EVENT_CANCEL` - The *Close* button is clicked.

The keyboard has a **default event handler** callback called `lv_keyboard_def_event_cb`, which handles the button pressing, map changing, the assigned text area, etc. You can remove it and replace it with a custom event handler if you wish.

注解： In 8.0 and newer, adding an event handler to the keyboard does not remove the default event handler. This behavior differs from v7, where adding an event handler would always replace the previous one.

Learn more about *Events*.

- `LV_EVENT_VALUE_CHANGED` 当按钮被按下/释放或允许长按后重复时发送。事件数据设置为按下/释放的按钮的 ID 。
- `LV_EVENT_READY` - *Ok* (LV_SYMBOL_OK) 按钮被点击
- `LV_EVENT_CANCEL` - *Close* (`LV_SYMBOL_CLOSE`) 按钮被点击

键盘有一个默认的事件处理回调函数：`lv_keyboard_def_event_cb` 。其用于处理按钮的按下、map(布局) 的更改、绑定的文本框等。如果您愿意，可以将其删除并替换为自定义的事件处理回调函数。

注解： 在 8.0 及更高版本中，向键盘添加事件处理回调函数不会删除默认 (之前) 的事件处理回调函数。此行为与 v7 不同，因为在 v7 中，新的事件处理回调函数会替换之前的。

了解有关事件 的更多信息。

Keys（按键）

- `LV_KEY_RIGHT/UP/LEFT/RIGHT` To navigate among the buttons and select one.
- `LV_KEY_ENTER` To press/release the selected button.

Learn more about *Keys*.

- `LV_KEY_RIGHT/UP/LEFT/RIGHT` 通过导航模式选中按钮。
- `LV_KEY_ENTER` 按下/释放所选中按钮 (模拟触摸点击的效果)。

了解关于按键 的更多信息。

Examples（示例）

Keyboard with text area

```
#include "../../lv_examples.h"
#if LV_USE_KEYBOARD && LV_BUILD_EXAMPLES

static void ta_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * ta = lv_event_get_target(e);
    lv_obj_t * kb = lv_event_get_user_data(e);
    if(code == LV_EVENT_FOCUSED) {
        lv_keyboard_set_textarea(kb, ta);
        lv_obj_clear_flag(kb, LV_OBJ_FLAG_HIDDEN);
    }

    if(code == LV_EVENT_DEFOCUSED) {
        lv_keyboard_set_textarea(kb, NULL);
        lv_obj_add_flag(kb, LV_OBJ_FLAG_HIDDEN);
    }
}

void lv_example_keyboard_1(void)
{
    /*Create a keyboard to use it with an of the text areas*/
    lv_obj_t *kb = lv_keyboard_create(lv_scr_act());

    /*Create a text area. The keyboard will write here*/
    lv_obj_t * ta;
    ta = lv_textarea_create(lv_scr_act());
    lv_obj_align(ta, LV_ALIGN_TOP_LEFT, 10, 10);
    lv_obj_add_event_cb(ta, ta_event_cb, LV_EVENT_ALL, kb);
    lv_textarea_set_placeholder_text(ta, "Hello");
    lv_obj_set_size(ta, 140, 80);

    ta = lv_textarea_create(lv_scr_act());
    lv_obj_align(ta, LV_ALIGN_TOP_RIGHT, -10, 10);
    lv_obj_add_event_cb(ta, ta_event_cb, LV_EVENT_ALL, kb);
    lv_obj_set_size(ta, 140, 80);

    lv_keyboard_set_textarea(kb, ta);
}
#endif
```

```
def ta_event_cb(e,kb):
    code = e.get_code()
    ta = e.get_target()
    if code == lv.EVENT.FOCUSED:
        kb.set_textarea(ta)
        kb.clear_flag(lv.obj.FLAG.HIDDEN)

    if code == lv.EVENT.DEFOCUSED:
        kb.set_textarea(None)
        kb.add_flag(lv.obj.FLAG.HIDDEN)

# Create a keyboard to use it with one of the text areas
kb = lv.keyboard(lv.scr_act())

# Create a text area. The keyboard will write here
ta = lv.textarea(lv.scr_act())
ta.set_width(200)
ta.align(lv.ALIGN.TOP_LEFT, 10, 10)
ta.add_event_cb(lambda e: ta_event_cb(e,kb), lv.EVENT.ALL, None)
ta.set_placeholder_text("Hello")

ta = lv.textarea(lv.scr_act())
ta.set_width(200)
ta.align(lv.ALIGN.TOP_RIGHT, -10, 10)
ta.add_event_cb(lambda e: ta_event_cb(e,kb), lv.EVENT.ALL, None)

kb.set_textarea(ta)
```

API

Typedefs

typedef uint8_t **lv_keyboard_mode_t**

Enums

enum **[anonymous]**
Current keyboard mode.

Values:

enumerator **LV_KEYBOARD_MODE_TEXT_LOWER**

enumerator **LV_KEYBOARD_MODE_TEXT_UPPER**

enumerator **LV_KEYBOARD_MODE_SPECIAL**

enumerator **LV_KEYBOARD_MODE_NUMBER**

enumerator **LV_KEYBOARD_MODE_USER_1**

enumerator **LV_KEYBOARD_MODE_USER_2**

enumerator **LV_KEYBOARD_MODE_USER_3**

enumerator **LV_KEYBOARD_MODE_USER_4**

Functions

lv_obj_t ***lv_keyboard_create**(*lv_obj_t* *parent)
Create a Keyboard object

参数 **parent** -- pointer to an object, it will be the parent of the new keyboard

返回 pointer to the created keyboard

void **lv_keyboard_set_textarea**(*lv_obj_t* *kb, *lv_obj_t* *ta)
Assign a Text Area to the Keyboard. The pressed characters will be put there.

参数

- **kb** -- pointer to a Keyboard object
- **ta** -- pointer to a Text Area object to write there

void **lv_keyboard_set_mode**(*lv_obj_t* *kb, *lv_keyboard_mode_t* mode)
Set a new a mode (text or number map)

参数

- **kb** -- pointer to a Keyboard object
- **mode** -- the mode from 'lv_keyboard_mode_t'

void **lv_keyboard_set_popovers**(*lv_obj_t* *kb, bool en)
Show the button title in a popover when pressed.

参数

- **kb** -- pointer to a Keyboard object
- **en** -- whether ”popovers” mode is enabled

void **lv_keyboard_set_map**(*lv_obj_t* *kb, *lv_keyboard_mode_t* mode, const char *map[], const *lv_btnmatrix_ctrl_t* ctrl_map[])

Set a new map for the keyboard

参数

- **kb** -- pointer to a Keyboard object
- **mode** -- keyboard map to alter ’lv_keyboard_mode_t’
- **map** -- pointer to a string array to describe the map. See ’*lv_btnmatrix_set_map()*’ for more info.

lv_obj_t ***lv_keyboard_get_textarea**(const *lv_obj_t* *kb)

Assign a Text Area to the Keyboard. The pressed characters will be put there.

参数 **kb** -- pointer to a Keyboard object

返回 pointer to the assigned Text Area object

lv_keyboard_mode_t **lv_keyboard_get_mode**(const *lv_obj_t* *kb)

Set a new a mode (text or number map)

参数 **kb** -- pointer to a Keyboard object

返回 the current mode from ’lv_keyboard_mode_t’

bool **lv_btnmatrix_get_popovers**(const *lv_obj_t* *obj)

Tell whether ”popovers” mode is enabled or not.

参数 **kb** -- pointer to a Keyboard object

返回 true: ”popovers” mode is enabled; false: disabled

static inline const char ****lv_keyboard_get_map_array**(const *lv_obj_t* *kb)

Get the current map of a keyboard

参数 **kb** -- pointer to a keyboard object

返回 the current map

static inline uint16_t **lv_keyboard_get_selected_btn**(const *lv_obj_t* *obj)

Get the index of the lastly ”activated” button by the user (pressed, released, focused etc) Useful in the `event_cb` to get the text of the button, check if hidden etc.

参数 **obj** -- pointer to button matrix object

返回 index of the last released button (LV_BTNMATRIX_BTN_NONE: if unset)

static inline const char *`lv_keyboard_get_btn_text`(const *lv_obj_t* *obj, uint16_t btn_id)

Get the button's text

参数

- **obj** -- pointer to button matrix object
- **btn_id** -- the index a button not counting new line characters.

返回 text of btn_index' button

void `lv_keyboard_def_event_cb`(lv_event_t *e)

Default keyboard event to add characters to the Text area and change the map. If a custom `event_cb` is added to the keyboard this function can be called from it to handle the button clicks

参数

- **kb** -- pointer to a keyboard
- **event** -- the triggering event

Variables

const lv_obj_class_t **lv_keyboard_class**

struct **lv_keyboard_t**

Public Members

lv_btnmatrix_t **btnm**

lv_obj_t ***ta**

lv_keyboard_mode_t **mode**

uint8_t **popovers**

LED (lv_led)

Overview

The LEDs are rectangle-like (or circle) object whose brightness can be adjusted. With lower brightness the colors of the LED become darker.

Parts and Styles

The LEDs have only one main part, called `LV_LED_PART_MAIN` and it uses all the typical background style properties.

Usage

Color

You can set the color of the LED with `lv_led_set_color(led, lv_color_hex(0xff0080))`. This will be used as background color, border color, and shadow color.

Brightness

You can set their brightness with `lv_led_set_bright(led, bright)`. The brightness should be between 0 (darkest) and 255 (lightest).

Toggle

Use `lv_led_on(led)` and `lv_led_off(led)` to set the brightness to a predefined ON or OFF value. The `lv_led_toggle(led)` toggles between the ON and OFF state.

Events

- `LV_EVENT_DRAW_PART_BEGIN` and `LV_EVENT_DRAW_PART_END` is sent for the following types:
  - `LV_LED_DRAW_PART_RECTANGLE` The main rectangle. `LV_OBJ_DRAW_PART_RECTANGLE` is not sent by the base object.
    * `part`: `LV_PART_MAIN`
    * `rect_dsc`
    * `draw_area`: the area of the rectangle

See the events of the *Base object* too.

Learn more about *Events*.

Keys

No *Keys* are processed by the object type.

Learn more about *Keys*.

Example

LED with custom style

```
#include "../../lv_examples.h"
#if LV_USE_LED && LV_BUILD_EXAMPLES

/**
 * Create LED's with different brightness and color
 */
void lv_example_led_1(void)
{
    /*Create a LED and switch it OFF*/
    lv_obj_t * led1  = lv_led_create(lv_scr_act());
    lv_obj_align(led1, LV_ALIGN_CENTER, -80, 0);
    lv_led_off(led1);

    /*Copy the previous LED and set a brightness*/
    lv_obj_t * led2  = lv_led_create(lv_scr_act());
    lv_obj_align(led2, LV_ALIGN_CENTER, 0, 0);
    lv_led_set_brightness(led2, 150);
    lv_led_set_color(led2, lv_palette_main(LV_PALETTE_RED));

    /*Copy the previous LED and switch it ON*/
    lv_obj_t * led3  = lv_led_create(lv_scr_act());
    lv_obj_align(led3, LV_ALIGN_CENTER, 80, 0);
    lv_led_on(led3);
}

#endif
```

```
#
# Create LED's with different brightness and color
#

# Create a LED and switch it OFF
led1  = lv.led(lv.scr_act())
```

(下页继续)

(续上页)

```
led1.align(lv.ALIGN.CENTER, -80, 0)
led1.off()

# Copy the previous LED and set a brightness
led2  = lv.led(lv.scr_act())
led2.align(lv.ALIGN.CENTER, 0, 0)
led2.set_brightness(150)
led2.set_color(lv.palette_main(lv.PALETTE.RED))

# Copy the previous LED and switch it ON
led3  = lv.led(lv.scr_act())
led3.align(lv.ALIGN.CENTER, 80, 0)
led3.on()
```

API

Enums

enum **lv_led_draw_part_type_t**

`type` field in `lv_obj_draw_part_dsc_t` if `class_p = lv_led_class` Used in `LV_EVENT_DRAW_PART_BEGIN` and `LV_EVENT_DRAW_PART_END`

Values:

enumerator **LV_LED_DRAW_PART_RECTANGLE**

The main rectangle

Functions

lv_obj_t ***lv_led_create**(*lv_obj_t* *parent)

Create a led object

参数 **parent** -- pointer to an object, it will be the parent of the new led

返回 pointer to the created led

void **lv_led_set_color**(*lv_obj_t* *led, lv_color_t color)

Set the color of the LED

参数

- **led** -- pointer to a LED object
- **color** -- the color of the LED

void **lv_led_set_brightness**(*lv_obj_t* *led, uint8_t bright)

Set the brightness of a LED object

参数

- **led** -- pointer to a LED object
- **bright** -- LV_LED_BRIGHT_MIN (max. dark) ... LV_LED_BRIGHT_MAX (max. light)

void **lv_led_on**(*lv_obj_t* *led)

Light on a LED

参数 **led** -- pointer to a LED object

void **lv_led_off**(*lv_obj_t* *led)

Light off a LED

参数 **led** -- pointer to a LED object

void **lv_led_toggle**(*lv_obj_t* *led)

Toggle the state of a LED

参数 **led** -- pointer to a LED object

uint8_t **lv_led_get_brightness**(const *lv_obj_t* *obj)

Get the brightness of a LEd object

参数 **led** -- pointer to LED object

返回 bright 0 (max. dark) ... 255 (max. light)

Variables

const lv_obj_class_t **lv_led_class**

struct **lv_led_t**

Public Members

lv_obj_t **obj**

lv_color_t **color**

uint8_t **bright**

Current brightness of the LED (0..255)

List (lv_list)

Overview

The List is basically a rectangle with vertical layout to which Buttons and Texts can be added

Parts and Styles

Background

- `LV_PART_MAIN` The main part of the list that uses all the typical background properties
- `LV_PART_SCROLLBAR` The scrollbar. See the *Base objects* documentation for details.

Buttons and Texts See the *Button*'s and *Label*'s documentation.

Usage

Buttons

`lv_list_add_btn(list, icon, text)` adds a full-width button with an icon - that can be an image or symbol - and a text.

The text starts to scroll horizontally if it's too long.

Texts

`lv_list_add_text(list, text)` adds a text.

Events

No special events are sent by the List, but sent by the Button as usual.

Learn more about *Events*.

Keys

No *Keys* are processed by the object type.

Learn more about *Keys*.

Example

Simple List

```
#include "../../lv_examples.h"
#if LV_USE_LIST && LV_BUILD_EXAMPLES
static lv_obj_t * list1;

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target(e);
    if(code == LV_EVENT_CLICKED) {
        LV_LOG_USER("Clicked: %s", lv_list_get_btn_text(list1, obj));
    }
}

void lv_example_list_1(void)
{
    /*Create a list*/
    list1 = lv_list_create(lv_scr_act());
    lv_obj_set_size(list1, 180, 220);
    lv_obj_center(list1);

    /*Add buttons to the list*/
    lv_obj_t * btn;

    lv_list_add_text(list1, "File");
    btn = lv_list_add_btn(list1, LV_SYMBOL_FILE, "New");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_btn(list1, LV_SYMBOL_DIRECTORY, "Open");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_btn(list1, LV_SYMBOL_SAVE, "Save");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_btn(list1, LV_SYMBOL_CLOSE, "Delete");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_btn(list1, LV_SYMBOL_EDIT, "Edit");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);

    lv_list_add_text(list1, "Connectivity");
    btn = lv_list_add_btn(list1, LV_SYMBOL_BLUETOOTH, "Bluetooth");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_btn(list1, LV_SYMBOL_GPS, "Navigation");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
```

(下页继续)

(续上页)

```
    btn = lv_list_add_btn(list1, LV_SYMBOL_USB, "USB");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_btn(list1, LV_SYMBOL_BATTERY_FULL, "Battery");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);

    lv_list_add_text(list1, "Exit");
    btn = lv_list_add_btn(list1, LV_SYMBOL_OK, "Apply");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_btn(list1, LV_SYMBOL_CLOSE, "Close");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
}

#endif
```

```
def event_handler(e):
    code = e.get_code()
    obj = e.get_target()
    if code == lv.EVENT.CLICKED:
            print("Clicked: list1." + list1.get_btn_text(obj))

# Create a list
list1 = lv.list(lv.scr_act())
list1.set_size(180, 220)
list1.center()

# Add buttons to the list
list1.add_text("File")
btn_new = list1.add_btn(lv.SYMBOL.FILE, "New")
btn_new.add_event_cb(event_handler,lv.EVENT.ALL, None)
btn_open = list1.add_btn(lv.SYMBOL.DIRECTORY, "Open")
btn_open.add_event_cb(event_handler,lv.EVENT.ALL, None)
btn_save = list1.add_btn(lv.SYMBOL.SAVE, "Save")
btn_save.add_event_cb(event_handler,lv.EVENT.ALL, None)
btn_delete = list1.add_btn(lv.SYMBOL.CLOSE, "Delete")
btn_delete.add_event_cb(event_handler,lv.EVENT.ALL, None)
btn_edit = list1.add_btn(lv.SYMBOL.EDIT, "Edit")
btn_edit.add_event_cb(event_handler,lv.EVENT.ALL, None)

list1.add_text("Connectivity")
btn_bluetooth = list1.add_btn(lv.SYMBOL.BLUETOOTH, "Bluetooth")
btn_bluetooth.add_event_cb(event_handler,lv.EVENT.ALL, None)
btn_navig = list1.add_btn(lv.SYMBOL.GPS, "Navigation")
btn_navig.add_event_cb(event_handler,lv.EVENT.ALL, None)
```

(下页继续)

(续上页)

```
btn_USB = list1.add_btn(lv.SYMBOL.USB, "USB")
btn_USB.add_event_cb(event_handler,lv.EVENT.ALL, None)
btn_battery = list1.add_btn(lv.SYMBOL.BATTERY_FULL, "Battery")
btn_battery.add_event_cb(event_handler,lv.EVENT.ALL, None)

list1.add_text("Exit")
btn_apply = list1.add_btn(lv.SYMBOL.OK, "Apply")
btn_apply.add_event_cb(event_handler,lv.EVENT.ALL, None)
btn_close = list1.add_btn(lv.SYMBOL.CLOSE, "Close")
btn_close.add_event_cb(event_handler,lv.EVENT.ALL, None)
```

Sorting a List using up and down buttons

```
#include <stdlib.h>


#include "../../lv_examples.h"
#if LV_USE_LIST && LV_BUILD_EXAMPLES

static lv_obj_t* list1;
static lv_obj_t* list2;

static lv_obj_t* currentButton = NULL;

static void event_handler(lv_event_t* e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t* obj = lv_event_get_target(e);
    if (code == LV_EVENT_CLICKED)
    {
        LV_LOG_USER("Clicked: %s", lv_list_get_btn_text(list1, obj));

        if (currentButton == obj)
        {
            currentButton = NULL;
        }
        else
        {
            currentButton = obj;
        }
        lv_obj_t* parent = lv_obj_get_parent(obj);
```

(下页继续)

(续上页)

```
        uint32_t i;
        for (i = 0; i < lv_obj_get_child_cnt(parent); i++)
        {
            lv_obj_t* child = lv_obj_get_child(parent, i);
            if (child == currentButton)
            {
                lv_obj_add_state(child, LV_STATE_CHECKED);
            }
            else
            {
                lv_obj_clear_state(child, LV_STATE_CHECKED);
            }
        }
    }
}

static void event_handler_top(lv_event_t* e)
{
    lv_event_code_t code = lv_event_get_code(e);
    if (code == LV_EVENT_CLICKED)
    {
        if (currentButton == NULL) return;
        lv_obj_move_background(currentButton);
        lv_obj_scroll_to_view(currentButton, LV_ANIM_ON);
    }
}

static void event_handler_up(lv_event_t* e)
{
    lv_event_code_t code = lv_event_get_code(e);
    if ((code == LV_EVENT_CLICKED) || (code == LV_EVENT_LONG_PRESSED_REPEAT))
    {
        if (currentButton == NULL) return;
        uint32_t index = lv_obj_get_index(currentButton);
        if (index <= 0) return;
        lv_obj_move_to_index(currentButton, index - 1);
        lv_obj_scroll_to_view(currentButton, LV_ANIM_ON);
    }
}

static void event_handler_center(lv_event_t* e)
{
    const lv_event_code_t code = lv_event_get_code(e);
```

(下页继续)

(续上页)

```
    if ((code == LV_EVENT_CLICKED) || (code == LV_EVENT_LONG_PRESSED_REPEAT))
    {
        if (currentButton == NULL) return;

        lv_obj_t* parent = lv_obj_get_parent(currentButton);
        const uint32_t pos = lv_obj_get_child_cnt(parent) / 2;

        lv_obj_move_to_index(currentButton, pos);

        lv_obj_scroll_to_view(currentButton, LV_ANIM_ON);
    }
}

static void event_handler_dn(lv_event_t* e)
{
    const lv_event_code_t code = lv_event_get_code(e);
    if ((code == LV_EVENT_CLICKED) || (code == LV_EVENT_LONG_PRESSED_REPEAT))
    {
        if (currentButton == NULL) return;
        const uint32_t index = lv_obj_get_index(currentButton);

        lv_obj_move_to_index(currentButton, index + 1);
        lv_obj_scroll_to_view(currentButton, LV_ANIM_ON);
    }
}

static void event_handler_bottom(lv_event_t* e)
{
    const lv_event_code_t code = lv_event_get_code(e);
    if (code == LV_EVENT_CLICKED)
    {
        if (currentButton == NULL) return;
        lv_obj_move_foreground(currentButton);
        lv_obj_scroll_to_view(currentButton, LV_ANIM_ON);
    }
}

static void event_handler_swap(lv_event_t* e)
{
    const lv_event_code_t code = lv_event_get_code(e);
    // lv_obj_t* obj = lv_event_get_target(e);
    if ((code == LV_EVENT_CLICKED) || (code == LV_EVENT_LONG_PRESSED_REPEAT))
    {
```

(下页继续)

(续上页)

```
        uint32_t cnt = lv_obj_get_child_cnt(list1);
        for (int i = 0; i < 100; i++)
            if (cnt > 1)
            {
                lv_obj_t* obj = lv_obj_get_child(list1, rand() % cnt);
                lv_obj_move_to_index(obj, rand() % cnt);
                if (currentButton != NULL)
                {
                    lv_obj_scroll_to_view(currentButton, LV_ANIM_ON);
                }
            }
    }
}

void lv_example_list_2(void)
{
    /*Create a list*/
    list1 = lv_list_create(lv_scr_act());
    lv_obj_set_size(list1, lv_pct(60), lv_pct(100));
    lv_obj_set_style_pad_row(list1, 5, 0);

    /*Add buttons to the list*/
    lv_obj_t* btn;
    int i;
    for (i = 0; i < 15; i++) {
        btn = lv_btn_create(list1);
        lv_obj_set_width(btn, lv_pct(50));
        lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);

        lv_obj_t* lab = lv_label_create(btn);
        lv_label_set_text_fmt(lab, "Item %d", i);
    }

    /*Select the first button by default*/
    currentButton = lv_obj_get_child(list1, 0);
    lv_obj_add_state(currentButton, LV_STATE_CHECKED);

    /*Create a second list with up and down buttons*/
    list2 = lv_list_create(lv_scr_act());
    lv_obj_set_size(list2, lv_pct(40), lv_pct(100));
    lv_obj_align(list2, LV_ALIGN_TOP_RIGHT, 0, 0);
    lv_obj_set_flex_flow(list2, LV_FLEX_FLOW_COLUMN);
```

(下页继续)

(续上页)

```
    btn = lv_list_add_btn(list2, NULL, "Top");
    lv_obj_add_event_cb(btn, event_handler_top, LV_EVENT_ALL, NULL);
    lv_group_remove_obj(btn);

    btn = lv_list_add_btn(list2, LV_SYMBOL_UP, "Up");
    lv_obj_add_event_cb(btn, event_handler_up, LV_EVENT_ALL, NULL);
    lv_group_remove_obj(btn);

    btn = lv_list_add_btn(list2, LV_SYMBOL_LEFT, "Center");
    lv_obj_add_event_cb(btn, event_handler_center, LV_EVENT_ALL, NULL);
    lv_group_remove_obj(btn);

    btn = lv_list_add_btn(list2, LV_SYMBOL_DOWN, "Down");
    lv_obj_add_event_cb(btn, event_handler_dn, LV_EVENT_ALL, NULL);
    lv_group_remove_obj(btn);

    btn = lv_list_add_btn(list2, NULL, "Bottom");
    lv_obj_add_event_cb(btn, event_handler_bottom, LV_EVENT_ALL, NULL);
    lv_group_remove_obj(btn);

    btn = lv_list_add_btn(list2, LV_SYMBOL_SHUFFLE, "Shuffle");
    lv_obj_add_event_cb(btn, event_handler_swap, LV_EVENT_ALL, NULL);
    lv_group_remove_obj(btn);
}

#endif
```

```
import urandom

currentButton = None
list1 = None

def event_handler(evt):
    global currentButton
    code = evt.get_code()
    obj = evt.get_target()
    if code == lv.EVENT.CLICKED:
        if currentButton == obj:
            currentButton = None
        else:
            currentButton = obj
        parent = obj.get_parent()
        for i in range( parent.get_child_cnt()):
```

(下页继续)

(续上页)

```
            child = parent.get_child(i)
            if child == currentButton:
                child.add_state(lv.STATE.CHECKED)
            else:
                child.clear_state(lv.STATE.CHECKED)

def event_handler_top(evt):
    global currentButton
    code = evt.get_code()
    obj = evt.get_target()
    if code == lv.EVENT.CLICKED:
        if currentButton == None:
            return
        currentButton.move_background()
        currentButton.scroll_to_view( lv.ANIM.ON)

def event_handler_up(evt):
    global currentButton
    code = evt.get_code()
    obj = evt.get_target()
    if code == lv.EVENT.CLICKED or code == lv.EVENT.LONG_PRESSED_REPEAT:
        if currentButton == None:
            return
        index = currentButton.get_index()
        if index <= 0:
            return
        currentButton.move_to_index(index - 1)
        currentButton.scroll_to_view(lv.ANIM.ON)

def event_handler_center(evt):
    global currentButton
    code = evt.get_code()
    obj = evt.get_target()
    if code == lv.EVENT.CLICKED or code == lv.EVENT.LONG_PRESSED_REPEAT:
        if currentButton == None:
            return
        parent = currentButton.get_parent()
        pos = parent.get_child_cnt() // 2
        currentButton.move_to_index(pos)
        currentButton.scroll_to_view(lv.ANIM.ON)

def event_handler_dn(evt):
    global currentButton
```

(下页继续)

(续上页)

```
    code = evt.get_code()
    obj = evt.get_target()
    if code == lv.EVENT.CLICKED or code == lv.EVENT.LONG_PRESSED_REPEAT:
        if currentButton == None:
            return
        index = currentButton.get_index()
        currentButton.move_to_index(index + 1)
        currentButton.scroll_to_view(lv.ANIM.ON)

def event_handler_bottom(evt):
    global currentButton
    code = evt.get_code()
    obj = evt.get_target()
    if code == lv.EVENT.CLICKED or code == lv.EVENT.LONG_PRESSED_REPEAT:
        if currentButton == None:
            return
        currentButton.move_foreground()
        currentButton.scroll_to_view(lv.ANIM.ON)

def event_handler_swap(evt):
    global currentButton
    global list1
    code = evt.get_code()
    obj = evt.get_target()
    if code == lv.EVENT.CLICKED:
        cnt = list1.get_child_cnt()
        for i in range(100):
            if cnt > 1:
                obj = list1.get_child(urandom.getrandbits(32) % cnt )
                obj.move_to_index(urandom.getrandbits(32) % cnt)
        if currentButton != None:
            currentButton.scroll_to_view(lv.ANIM.ON)

#Create a list with buttons that can be sorted
list1 = lv.list(lv.scr_act())
list1.set_size(lv.pct(60), lv.pct(100))
list1.set_style_pad_row( 5, 0)

for i in range(15):
    btn = lv.btn(list1)
    btn.set_width(lv.pct(100))
    btn.add_event_cb( event_handler, lv.EVENT.CLICKED, None)
    lab = lv.label(btn)
```

(下页继续)

(续上页)

```
    lab.set_text("Item " + str(i))

#Select the first button by default
currentButton = list1.get_child(0)
currentButton.add_state(lv.STATE.CHECKED)

#Create a second list with up and down buttons
list2 = lv.list(lv.scr_act())
list2.set_size(lv.pct(40), lv.pct(100))
list2.align(lv.ALIGN.TOP_RIGHT, 0, 0)
list2.set_flex_flow(lv.FLEX_FLOW.COLUMN)

btn = list2.add_btn(None, "Top")
btn.add_event_cb(event_handler_top, lv.EVENT.ALL, None)
lv.group_remove_obj(btn)

btn = list2.add_btn(lv.SYMBOL.UP, "Up")
btn.add_event_cb(event_handler_up, lv.EVENT.ALL, None)
lv.group_remove_obj(btn)

btn = list2.add_btn(lv.SYMBOL.LEFT, "Center")
btn.add_event_cb(event_handler_center, lv.EVENT.ALL, None)
lv.group_remove_obj(btn)

btn = list2.add_btn(lv.SYMBOL.DOWN, "Down")
btn.add_event_cb(event_handler_dn, lv.EVENT.ALL, None)
lv.group_remove_obj(btn)

btn = list2.add_btn(None, "Bottom")
btn.add_event_cb(event_handler_bottom, lv.EVENT.ALL, None)
lv.group_remove_obj(btn)

btn = list2.add_btn(lv.SYMBOL.SHUFFLE, "Shuffle")
btn.add_event_cb(event_handler_swap, lv.EVENT.ALL, None)
lv.group_remove_obj(btn)
```

API

Functions

lv_obj_t ***lv_list_create**(*lv_obj_t* *parent)

lv_obj_t ***lv_list_add_text**(*lv_obj_t* *list, const char *txt)

lv_obj_t ***lv_list_add_btn**(*lv_obj_t* *list, const char *icon, const char *txt)

const char ***lv_list_get_btn_text**(*lv_obj_t* *list, *lv_obj_t* *btn)

Variables

const lv_obj_class_t **lv_list_class**

const lv_obj_class_t **lv_list_text_class**

const lv_obj_class_t **lv_list_btn_class**

Menu (lv_menu)

Overview

The menu widget can be used to easily create multi-level menus. It handles the traversal between pages automatically.

Parts and Styles

The menu widget is built from the following objects:

- Main container: lv_menu_main_cont
  - Main header: lv_menu_main_header_cont
    - Back btn: *lv_btn*
      - Back btn icon: *lv_img*
  - Main page: lv_menu_page
- Sidebar container: lv_menu_sidebar_cont
  - Sidebar header: lv_menu_sidebar_header_cont

        * Back btn: *lv_btn*
            - Back btn icon: *lv_img*
    - Sidebar page: lv_menu_page

Usage

Create a menu

`lv_menu_create(parent)` creates a new empty menu.

Header mode

The following header modes exist:

- `LV_MENU_HEADER_TOP_FIXED` Header is positioned at the top.
- `LV_MENU_HEADER_TOP_UNFIXED` Header is positioned at the top and can be scrolled out of view.
- `LV_MENU_HEADER_BOTTOM_FIXED` Header is positioned at the bottom.

You can set header modes with `lv_menu_set_mode_header(menu, LV_MENU_HEADER...)`.

Root back button mode

The following root back button modes exist:

- `LV_MENU_ROOT_BACK_BTN_DISABLED`
- `LV_MENU_ROOT_BACK_BTN_ENABLED`

You can set root back button modes with `lv_menu_set_mode_root_back_btn(menu, LV_MENU_ROOT_BACK_BTN...)`.

Create a menu page

`lv_menu_page_create(menu, title)` creates a new empty menu page. You can add any widgets to the page.

Set a menu page in the main area

Once a menu page has been created, you can set it to the main area with `lv_menu_set_page(menu, page)`. NULL to clear main and clear menu history.

Set a menu page in the sidebar

Once a menu page has been created, you can set it to the sidebar with `lv_menu_set_sidebar_page(menu, page)`. NULL to clear sidebar.

Linking between menu pages

For instance, you have created a btn obj in the main page. When you click the btn obj, you want it to open up a new page, use `lv_menu_set_load_page_event(menu, obj, new page)`.

Create a menu container, section, separator

The following objects can be created so that it is easier to style the menu:

`lv_menu_cont_create(parent page)` creates a new empty container.

`lv_menu_section_create(parent page)` creates a new empty section.

`lv_menu_separator_create(parent page)` creates a separator.

Events

- `LV_EVENT_VALUE_CHANGED` Sent when a page is shown.
  - `lv_menu_get_cur_main_page(menu)` returns a pointer to menu page that is currently displayed in main.
  - `lv_menu_get_cur_sidebar_page(menu)` returns a pointer to menu page that is currently displayed in sidebar.
- `LV_EVENT_CLICKED` Sent when a back btn in a header from either main or sidebar is clicked. `LV_OBJ_FLAG_EVENT_BUBBLE` is enabled on the buttons so you can add events to the menu itself.
  - `lv_menu_back_btn_is_root(menu, btn)` to check if btn is root back btn

See the events of the *Base object* too.

Learn more about *Events*.

Keys

No keys are handled by the menu widget.

Learn more about *Keys*.

Example

Simple Menu

```
#include "../../lv_examples.h"
#if LV_USE_MENU && LV_BUILD_EXAMPLES

void lv_example_menu_1(void)
{
    /*Create a menu object*/
    lv_obj_t * menu = lv_menu_create(lv_scr_act());
    lv_obj_set_size(menu, lv_disp_get_hor_res(NULL), lv_disp_get_ver_res(NULL));
    lv_obj_center(menu);

    lv_obj_t * cont;
    lv_obj_t * label;

    /*Create a sub page*/
    lv_obj_t * sub_page = lv_menu_page_create(menu, NULL);

    cont = lv_menu_cont_create(sub_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Hello, I am hiding here");

    /*Create a main page*/
    lv_obj_t * main_page = lv_menu_page_create(menu, NULL);

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 1");

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 2");

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 3 (Click me!)");
```

(下页继续)

(续上页)

```
    lv_menu_set_load_page_event(menu, cont, sub_page);

    lv_menu_set_page(menu, main_page);
}

#endif
```

```
# Create a menu object
menu = lv.menu(lv.scr_act())
menu.set_size(320, 240)
menu.center()

# Create a sub page
sub_page = lv.menu_page(menu, None)
cont = lv.menu_cont(sub_page)
label = lv.label(cont)
label.set_text("Hello, I am hiding here")

# Create a main page
main_page = lv.menu_page(menu, None)

cont = lv.menu_cont(main_page)
label = lv.label(cont)
label.set_text("Item 1")

cont = lv.menu_cont(main_page)
label = lv.label(cont)
label.set_text("Item 2")

cont = lv.menu_cont(main_page)
label = lv.label(cont)
label.set_text("Item 3 (Click me!)")
menu.set_load_page_event(cont, sub_page)

menu.set_page(main_page)
```

Simple Menu with root btn

```
#include "../../lv_examples.h"
#if LV_USE_MENU && LV_USE_MSGBOX && LV_BUILD_EXAMPLES

static void back_event_handler(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target(e);
    lv_obj_t * menu = lv_event_get_user_data(e);

    if(lv_menu_back_btn_is_root(menu, obj)) {
        lv_obj_t * mbox1 = lv_msgbox_create(NULL, "Hello", "Root back btn click.",␣
↪NULL, true);
        lv_obj_center(mbox1);
    }
}

void lv_example_menu_2(void)
{
    lv_obj_t * menu = lv_menu_create(lv_scr_act());
    lv_menu_set_mode_root_back_btn(menu, LV_MENU_ROOT_BACK_BTN_ENABLED);
    lv_obj_add_event_cb(menu, back_event_handler, LV_EVENT_CLICKED, menu);
    lv_obj_set_size(menu, lv_disp_get_hor_res(NULL), lv_disp_get_ver_res(NULL));
    lv_obj_center(menu);

    lv_obj_t * cont;
    lv_obj_t * label;

    /*Create a sub page*/
    lv_obj_t * sub_page = lv_menu_page_create(menu, NULL);

    cont = lv_menu_cont_create(sub_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Hello, I am hiding here");

    /*Create a main page*/
    lv_obj_t * main_page = lv_menu_page_create(menu, NULL);

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 1");

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
```

(下页继续)

(续上页)

```
    lv_label_set_text(label, "Item 2");

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 3 (Click me!)");
    lv_menu_set_load_page_event(menu, cont, sub_page);

    lv_menu_set_page(menu, main_page);
}

#endif
```

```
def back_event_handler(e):
    obj = e.get_target()
    if menu.back_btn_is_root(obj):
        mbox1 = lv.msgbox(lv.scr_act(), "Hello", "Root back btn click.", None, True)
        mbox1.center()

# Create a menu object
menu = lv.menu(lv.scr_act())
menu.set_mode_root_back_btn(lv.menu.ROOT_BACK_BTN.ENABLED)
menu.add_event_cb(back_event_handler, lv.EVENT.CLICKED, None)
menu.set_size(320, 240)
menu.center()

# Create a sub page
sub_page = lv.menu_page(menu, None)
cont = lv.menu_cont(sub_page)
label = lv.label(cont)
label.set_text("Hello, I am hiding here")

# Create a main page
main_page = lv.menu_page(menu, None)

cont = lv.menu_cont(main_page)
label = lv.label(cont)
label.set_text("Item 1")

cont = lv.menu_cont(main_page)
label = lv.label(cont)
label.set_text("Item 2")

cont = lv.menu_cont(main_page)
```

(下页继续)

(续上页)

```
label = lv.label(cont)
label.set_text("Item 3 (Click me!)")
menu.set_load_page_event(cont, sub_page)

menu.set_page(main_page)
```

Simple Menu with custom header

```
#include "../../lv_examples.h"
#if LV_USE_MENU && LV_USE_USER_DATA && LV_BUILD_EXAMPLES

void lv_example_menu_3(void)
{
    /*Create a menu object*/
    lv_obj_t * menu = lv_menu_create(lv_scr_act());
    lv_obj_set_size(menu, lv_disp_get_hor_res(NULL), lv_disp_get_ver_res(NULL));
    lv_obj_center(menu);

    /*Modify the header*/
    lv_obj_t * back_btn = lv_menu_get_main_header_back_btn(menu);
    lv_obj_t * back_btn_label = lv_label_create(back_btn);
    lv_label_set_text(back_btn_label, "Back");

    lv_obj_t * cont;
    lv_obj_t * label;

    /*Create sub pages*/
    lv_obj_t * sub_1_page = lv_menu_page_create(menu, "Page 1");

    cont = lv_menu_cont_create(sub_1_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Hello, I am hiding here");

    lv_obj_t * sub_2_page = lv_menu_page_create(menu, "Page 2");

    cont = lv_menu_cont_create(sub_2_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Hello, I am hiding here");

    lv_obj_t * sub_3_page = lv_menu_page_create(menu, "Page 3");

    cont = lv_menu_cont_create(sub_3_page);
```

(下页继续)

(续上页)

```
    label = lv_label_create(cont);
    lv_label_set_text(label, "Hello, I am hiding here");

    /*Create a main page*/
    lv_obj_t * main_page = lv_menu_page_create(menu, NULL);

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 1 (Click me!)");
    lv_menu_set_load_page_event(menu, cont, sub_1_page);

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 2 (Click me!)");
    lv_menu_set_load_page_event(menu, cont, sub_2_page);

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 3 (Click me!)");
    lv_menu_set_load_page_event(menu, cont, sub_3_page);

    lv_menu_set_page(menu, main_page);
}

#endif
```

```
# Create a menu object
menu = lv.menu(lv.scr_act())
menu.set_size(320, 240)
menu.center()

# Create sub pages
sub_page_1 = lv.menu_page(menu, "Page 1")

cont = lv.menu_cont(sub_page_1)
label = lv.label(cont)
label.set_text("Hello, I am hiding here")

sub_page_2 = lv.menu_page(menu, "Page 2")

cont = lv.menu_cont(sub_page_2)
label = lv.label(cont)
label.set_text("Hello, I am hiding here")
```

(下页继续)

(续上页)

```
sub_page_3 = lv.menu_page(menu, "Page 3")

cont = lv.menu_cont(sub_page_3)
label = lv.label(cont)
label.set_text("Hello, I am hiding here")

# Create a main page
main_page = lv.menu_page(menu, None)

cont = lv.menu_cont(main_page)
label = lv.label(cont)
label.set_text("Item 1 (Click me!)")
menu.set_load_page_event(cont, sub_page_1)

cont = lv.menu_cont(main_page)
label = lv.label(cont)
label.set_text("Item 2 (Click me!)")
menu.set_load_page_event(cont, sub_page_2)

cont = lv.menu_cont(main_page)
label = lv.label(cont)
label.set_text("Item 3 (Click me!)")
menu.set_load_page_event(cont, sub_page_3)

menu.set_page(main_page)
```

Simple Menu with floating btn to add new menu page

```
#include "../../lv_examples.h"
#if LV_USE_MENU && LV_BUILD_EXAMPLES

static uint32_t btn_cnt = 1;
static lv_obj_t * main_page;
static lv_obj_t * menu;

static void float_btn_event_cb(lv_event_t * e)
{
    LV_UNUSED(e);

    btn_cnt++;
```

(下页继续)

(续上页)

```
    lv_obj_t * cont;
    lv_obj_t * label;

    lv_obj_t * sub_page = lv_menu_page_create(menu, NULL);

    cont = lv_menu_cont_create(sub_page);
    label= lv_label_create(cont);
    lv_label_set_text_fmt(label, "Hello, I am hiding inside %i", btn_cnt);

    cont = lv_menu_cont_create(main_page);
    label= lv_label_create(cont);
    lv_label_set_text_fmt(label, "Item %i", btn_cnt);
    lv_menu_set_load_page_event(menu, cont, sub_page);

    lv_obj_scroll_to_view_recursive(cont, LV_ANIM_ON);
}

void lv_example_menu_4(void)
{
    /*Create a menu object*/
    menu = lv_menu_create(lv_scr_act());
    lv_obj_set_size(menu, lv_disp_get_hor_res(NULL), lv_disp_get_ver_res(NULL));
    lv_obj_center(menu);

    lv_obj_t * cont;
    lv_obj_t * label;

    /*Create a sub page*/
    lv_obj_t * sub_page = lv_menu_page_create(menu, NULL);

    cont = lv_menu_cont_create(sub_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Hello, I am hiding inside the first item");

    /*Create a main page*/
    main_page = lv_menu_page_create(menu, NULL);

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 1");
    lv_menu_set_load_page_event(menu, cont, sub_page);

    lv_menu_set_page(menu, main_page);
```

(下页继续)

(续上页)

```
    /*Create floating btn*/
    lv_obj_t * float_btn = lv_btn_create(lv_scr_act());
    lv_obj_set_size(float_btn, 50, 50);
    lv_obj_add_flag(float_btn, LV_OBJ_FLAG_FLOATING);
    lv_obj_align(float_btn, LV_ALIGN_BOTTOM_RIGHT, -10, -10);
    lv_obj_add_event_cb(float_btn, float_btn_event_cb, LV_EVENT_CLICKED, menu);
    lv_obj_set_style_radius(float_btn, LV_RADIUS_CIRCLE, 0);
    lv_obj_set_style_bg_img_src(float_btn, LV_SYMBOL_PLUS, 0);
    lv_obj_set_style_text_font(float_btn, lv_theme_get_font_large(float_btn), 0);
}

#endif
```

```
btn_cnt = 1

def float_btn_event_cb(e):
    global btn_cnt
    btn_cnt += 1

    sub_page = lv.menu_page(menu, None)

    cont = lv.menu_cont(sub_page)
    label = lv.label(cont)
    label.set_text("Hello, I am hiding inside {:d}".format(btn_cnt))

    cont = lv.menu_cont(main_page)
    label = lv.label(cont)
    label.set_text("Item {:d}".format(btn_cnt))
    menu.set_load_page_event(cont, sub_page)

# Create a menu object
menu = lv.menu(lv.scr_act())
menu.set_size(320, 240)
menu.center()

# Create a sub page
sub_page = lv.menu_page(menu, None)

cont = lv.menu_cont(sub_page)
label = lv.label(cont)
label.set_text("Hello, I am hiding inside the first item")
```

(下页继续)

(续上页)

```
# Create a main page
main_page = lv.menu_page(menu, None)

cont = lv.menu_cont(main_page)
label = lv.label(cont)
label.set_text("Item 1")
menu.set_load_page_event(cont, sub_page)

menu.set_page(main_page)

float_btn = lv.btn(lv.scr_act())
float_btn.set_size(50, 50)
float_btn.add_flag(lv.obj.FLAG.FLOATING)
float_btn.align(lv.ALIGN.BOTTOM_RIGHT, -10, -10)
float_btn.add_event_cb(float_btn_event_cb, lv.EVENT.CLICKED, None)
float_btn.set_style_radius(lv.RADIUS.CIRCLE, 0)
float_btn.set_style_bg_img_src(lv.SYMBOL.PLUS, 0)
float_btn.set_style_text_font(lv.theme_get_font_large(float_btn), 0)
```

Complex Menu

```
#include "../../lv_examples.h"
#if LV_USE_MENU && LV_USE_MSGBOX && LV_BUILD_EXAMPLES

enum {
    LV_MENU_ITEM_BUILDER_VARIANT_1,
    LV_MENU_ITEM_BUILDER_VARIANT_2
};
typedef uint8_t lv_menu_builder_variant_t;

static void back_event_handler(lv_event_t * e);
static void switch_handler(lv_event_t * e);
lv_obj_t * root_page;
static lv_obj_t * create_text(lv_obj_t * parent, const char * icon, const char * txt,
                              lv_menu_builder_variant_t builder_variant);
static lv_obj_t * create_slider(lv_obj_t * parent,
                                const char * icon, const char * txt, int32_t min,␣
↪int32_t max, int32_t val);
static lv_obj_t * create_switch(lv_obj_t * parent,
                                const char * icon, const char * txt, bool chk);
```

(下页继续)

(续上页)

```
void lv_example_menu_5(void)
{
    lv_obj_t * menu = lv_menu_create(lv_scr_act());

    lv_color_t bg_color = lv_obj_get_style_bg_color(menu, 0);
    if(lv_color_brightness(bg_color) > 127) {
        lv_obj_set_style_bg_color(menu, lv_color_darken(lv_obj_get_style_bg_
↪color(menu, 0), 10), 0);
    }else{
        lv_obj_set_style_bg_color(menu, lv_color_darken(lv_obj_get_style_bg_
↪color(menu, 0), 50), 0);
    }
    lv_menu_set_mode_root_back_btn(menu, LV_MENU_ROOT_BACK_BTN_ENABLED);
    lv_obj_add_event_cb(menu, back_event_handler, LV_EVENT_CLICKED, menu);
    lv_obj_set_size(menu, lv_disp_get_hor_res(NULL), lv_disp_get_ver_res(NULL));
    lv_obj_center(menu);

    lv_obj_t * cont;
    lv_obj_t * section;

    /*Create sub pages*/
    lv_obj_t * sub_mechanics_page = lv_menu_page_create(menu, NULL);
    lv_obj_set_style_pad_hor(sub_mechanics_page, lv_obj_get_style_pad_left(lv_menu_
↪get_main_header(menu), 0), 0);
    lv_menu_separator_create(sub_mechanics_page);
    section = lv_menu_section_create(sub_mechanics_page);
    create_slider(section, LV_SYMBOL_SETTINGS, "Velocity", 0, 150, 120);
    create_slider(section, LV_SYMBOL_SETTINGS, "Acceleration", 0, 150, 50);
    create_slider(section, LV_SYMBOL_SETTINGS, "Weight limit", 0, 150, 80);

    lv_obj_t * sub_sound_page = lv_menu_page_create(menu, NULL);
    lv_obj_set_style_pad_hor(sub_sound_page, lv_obj_get_style_pad_left(lv_menu_get_
↪main_header(menu), 0), 0);
    lv_menu_separator_create(sub_sound_page);
    section = lv_menu_section_create(sub_sound_page);
    create_switch(section, LV_SYMBOL_AUDIO, "Sound", false);

    lv_obj_t * sub_display_page = lv_menu_page_create(menu, NULL);
    lv_obj_set_style_pad_hor(sub_display_page, lv_obj_get_style_pad_left(lv_menu_get_
↪main_header(menu), 0), 0);
    lv_menu_separator_create(sub_display_page);
    section = lv_menu_section_create(sub_display_page);
    create_slider(section, LV_SYMBOL_SETTINGS, "Brightness", 0, 150, 100);
```

(下页继续)

(续上页)

```
    lv_obj_t * sub_software_info_page = lv_menu_page_create(menu, NULL);
    lv_obj_set_style_pad_hor(sub_software_info_page, lv_obj_get_style_pad_left(lv_
↪menu_get_main_header(menu), 0), 0);
    section = lv_menu_section_create(sub_software_info_page);
    create_text(section, NULL, "Version 1.0", LV_MENU_ITEM_BUILDER_VARIANT_1);

    lv_obj_t * sub_legal_info_page = lv_menu_page_create(menu, NULL);
    lv_obj_set_style_pad_hor(sub_legal_info_page, lv_obj_get_style_pad_left(lv_menu_
↪get_main_header(menu), 0), 0);
    section = lv_menu_section_create(sub_legal_info_page);
    for(uint32_t i=0; i<15; i++){
        create_text(section, NULL, "This is a long long long long long long long␣
↪long text, if it is long enough it may scroll.", LV_MENU_ITEM_BUILDER_VARIANT_1);
    }

    lv_obj_t * sub_about_page = lv_menu_page_create(menu, NULL);
    lv_obj_set_style_pad_hor(sub_about_page, lv_obj_get_style_pad_left(lv_menu_get_
↪main_header(menu), 0), 0);
    lv_menu_separator_create(sub_about_page);
    section = lv_menu_section_create(sub_about_page);
    cont = create_text(section, NULL, "Software information", LV_MENU_ITEM_BUILDER_
↪VARIANT_1);
    lv_menu_set_load_page_event(menu, cont, sub_software_info_page);
    cont = create_text(section, NULL, "Legal information", LV_MENU_ITEM_BUILDER_
↪VARIANT_1);
    lv_menu_set_load_page_event(menu, cont, sub_legal_info_page);

    lv_obj_t * sub_menu_mode_page = lv_menu_page_create(menu, NULL);
    lv_obj_set_style_pad_hor(sub_menu_mode_page, lv_obj_get_style_pad_left(lv_menu_
↪get_main_header(menu), 0), 0);
    lv_menu_separator_create(sub_menu_mode_page);
    section = lv_menu_section_create(sub_menu_mode_page);
    cont = create_switch(section, LV_SYMBOL_AUDIO, "Sidebar enable", true);
    lv_obj_add_event_cb(lv_obj_get_child(cont, 2), switch_handler, LV_EVENT_VALUE_
↪CHANGED, menu);

    /*Create a root page*/
    root_page = lv_menu_page_create(menu, "Settings");
    lv_obj_set_style_pad_hor(root_page, lv_obj_get_style_pad_left(lv_menu_get_main_
↪header(menu), 0), 0);
    section = lv_menu_section_create(root_page);
    cont = create_text(section, LV_SYMBOL_SETTINGS, "Mechanics", LV_MENU_ITEM_BUILDER_
↪VARIANT_1);
```

(下页继续)

(续上页)

```
    lv_menu_set_load_page_event(menu, cont, sub_mechanics_page);
    cont = create_text(section, LV_SYMBOL_AUDIO, "Sound", LV_MENU_ITEM_BUILDER_
↪VARIANT_1);
    lv_menu_set_load_page_event(menu, cont, sub_sound_page);
    cont = create_text(section, LV_SYMBOL_SETTINGS, "Display", LV_MENU_ITEM_BUILDER_
↪VARIANT_1);
    lv_menu_set_load_page_event(menu, cont, sub_display_page);

    create_text(root_page, NULL, "Others", LV_MENU_ITEM_BUILDER_VARIANT_1);
    section = lv_menu_section_create(root_page);
    cont = create_text(section, NULL, "About", LV_MENU_ITEM_BUILDER_VARIANT_1);
    lv_menu_set_load_page_event(menu, cont, sub_about_page);
    cont = create_text(section, LV_SYMBOL_SETTINGS, "Menu mode", LV_MENU_ITEM_BUILDER_
↪VARIANT_1);
    lv_menu_set_load_page_event(menu, cont, sub_menu_mode_page);

    lv_menu_set_sidebar_page(menu, root_page);

    lv_event_send(lv_obj_get_child(lv_obj_get_child(lv_menu_get_cur_sidebar_
↪page(menu), 0), 0), LV_EVENT_CLICKED, NULL);
}

static void back_event_handler(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target(e);
    lv_obj_t * menu = lv_event_get_user_data(e);

    if(lv_menu_back_btn_is_root(menu, obj)) {
        lv_obj_t * mbox1 = lv_msgbox_create(NULL, "Hello", "Root back btn click.",␣
↪NULL, true);
        lv_obj_center(mbox1);
    }
}

static void switch_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * menu = lv_event_get_user_data(e);
    lv_obj_t * obj = lv_event_get_target(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        if(lv_obj_has_state(obj, LV_STATE_CHECKED)) {
            lv_menu_set_page(menu, NULL);
            lv_menu_set_sidebar_page(menu, root_page);
```

(下页继续)

(续上页)

```
            lv_event_send(lv_obj_get_child(lv_obj_get_child(lv_menu_get_cur_sidebar_
↪page(menu), 0), 0), LV_EVENT_CLICKED, NULL);
        }else {
            lv_menu_set_sidebar_page(menu, NULL);
            lv_menu_clear_history(menu); /* Clear history because we will be showing␣
↪the root page later */
            lv_menu_set_page(menu, root_page);
        }
    }
}

static lv_obj_t * create_text(lv_obj_t * parent, const char * icon, const char * txt,
                                        lv_menu_builder_variant_t builder_variant)
{
    lv_obj_t * obj = lv_menu_cont_create(parent);

    lv_obj_t * img = NULL;
    lv_obj_t * label = NULL;

    if(icon) {
        img = lv_img_create(obj);
        lv_img_set_src(img, icon);
    }

    if(txt) {
        label = lv_label_create(obj);
        lv_label_set_text(label, txt);
        lv_label_set_long_mode(label, LV_LABEL_LONG_SCROLL_CIRCULAR);
        lv_obj_set_flex_grow(label, 1);
    }

    if(builder_variant == LV_MENU_ITEM_BUILDER_VARIANT_2 && icon && txt) {
        lv_obj_add_flag(img, LV_OBJ_FLAG_FLEX_IN_NEW_TRACK);
        lv_obj_swap(img, label);
    }

    return obj;
}

static lv_obj_t * create_slider(lv_obj_t * parent, const char * icon, const char *␣
↪txt, int32_t min, int32_t max, int32_t val)
{
    lv_obj_t * obj = create_text(parent, icon, txt, LV_MENU_ITEM_BUILDER_VARIANT_2);
```

(下页继续)

(续上页)

```
    lv_obj_t * slider = lv_slider_create(obj);
    lv_obj_set_flex_grow(slider, 1);
    lv_slider_set_range(slider, min, max);
    lv_slider_set_value(slider, val, LV_ANIM_OFF);

    if(icon == NULL) {
        lv_obj_add_flag(slider, LV_OBJ_FLAG_FLEX_IN_NEW_TRACK);
    }

    return obj;
}

static lv_obj_t * create_switch(lv_obj_t * parent, const char * icon, const char *␣
↪txt, bool chk)
{
    lv_obj_t * obj = create_text(parent, icon, txt, LV_MENU_ITEM_BUILDER_VARIANT_1);

    lv_obj_t * sw = lv_switch_create(obj);
    lv_obj_add_state(sw, chk ? LV_STATE_CHECKED : 0);

    return obj;
}

#endif
```

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs/100ask_lvgl_
↪docs/examples/widgets/menu/lv_example_menu_5.py
```

API

Typedefs

typedef uint8_t **lv_menu_mode_header_t**

typedef uint8_t **lv_menu_mode_root_back_btn_t**

typedef struct *lv_menu_load_page_event_data_t* **lv_menu_load_page_event_data_t**

Enums

enum **[anonymous]**
Values:

enumerator **LV_MENU_HEADER_TOP_FIXED**

enumerator **LV_MENU_HEADER_TOP_UNFIXED**

enumerator **LV_MENU_HEADER_BOTTOM_FIXED**

enum **[anonymous]**
Values:

enumerator **LV_MENU_ROOT_BACK_BTN_DISABLED**

enumerator **LV_MENU_ROOT_BACK_BTN_ENABLED**

Functions

lv_obj_t ***lv_menu_create**(*lv_obj_t* *parent)
Create a menu object

参数 **parent** -- pointer to an object, it will be the parent of the new menu

返回 pointer to the created menu

lv_obj_t ***lv_menu_page_create**(*lv_obj_t* *parent, char *title)
Create a menu page object

参数

- **parent** -- pointer to menu object
- **title** -- pointer to text for title in header (NULL to not display title)

返回 pointer to the created menu page

lv_obj_t ***lv_menu_cont_create**(*lv_obj_t* *parent)
Create a menu cont object

参数 **parent** -- pointer to an object, it will be the parent of the new menu cont object

返回 pointer to the created menu cont

lv_obj_t ***lv_menu_section_create**(*lv_obj_t* *parent)
Create a menu section object

参数 **parent** -- pointer to an object, it will be the parent of the new menu section object

返回 pointer to the created menu section

lv_obj_t ***lv_menu_separator_create**(*lv_obj_t* *parent)

Create a menu separator object

参数 **parent** -- pointer to an object, it will be the parent of the new menu separator object

返回 pointer to the created menu separator

void **lv_menu_set_page**(*lv_obj_t* *obj, *lv_obj_t* *page)

Set menu page to display in main

参数

- **obj** -- pointer to the menu
- **page** -- pointer to the menu page to set (NULL to clear main and clear menu history)

void **lv_menu_set_sidebar_page**(*lv_obj_t* *obj, *lv_obj_t* *page)

Set menu page to display in sidebar

参数

- **obj** -- pointer to the menu
- **page** -- pointer to the menu page to set (NULL to clear sidebar)

void **lv_menu_set_mode_header**(*lv_obj_t* *obj, *lv_menu_mode_header_t* mode_header)

Set the how the header should behave and its position

参数

- **obj** -- pointer to a menu
- **mode_header** --

void **lv_menu_set_mode_root_back_btn**(*lv_obj_t* *obj, *lv_menu_mode_root_back_btn_t* mode_root_back_btn)

Set whether back button should appear at root

参数

- **obj** -- pointer to a menu
- **mode_root_back_btn** --

void **lv_menu_set_load_page_event**(*lv_obj_t* *menu, *lv_obj_t* *obj, *lv_obj_t* *page)

Add menu to the menu item

参数

- **menu** -- pointer to the menu
- **obj** -- pointer to the obj
- **page** -- pointer to the page to load when obj is clicked

lv_obj_t ***lv_menu_get_cur_main_page**(*lv_obj_t* *obj)

Get a pointer to menu page that is currently displayed in main

参数 **obj** -- pointer to the menu

返回 pointer to current page

lv_obj_t ***lv_menu_get_cur_sidebar_page**(*lv_obj_t* *obj)

Get a pointer to menu page that is currently displayed in sidebar

参数 **obj** -- pointer to the menu

返回 pointer to current page

lv_obj_t ***lv_menu_get_main_header**(*lv_obj_t* *obj)

Get a pointer to main header obj

参数 **obj** -- pointer to the menu

返回 pointer to main header obj

lv_obj_t ***lv_menu_get_main_header_back_btn**(*lv_obj_t* *obj)

Get a pointer to main header back btn obj

参数 **obj** -- pointer to the menu

返回 pointer to main header back btn obj

lv_obj_t ***lv_menu_get_sidebar_header**(*lv_obj_t* *obj)

Get a pointer to sidebar header obj

参数 **obj** -- pointer to the menu

返回 pointer to sidebar header obj

lv_obj_t ***lv_menu_get_sidebar_header_back_btn**(*lv_obj_t* *obj)

Get a pointer to sidebar header obj

参数 **obj** -- pointer to the menu

返回 pointer to sidebar header back btn obj

bool **lv_menu_back_btn_is_root**(*lv_obj_t* *menu, *lv_obj_t* *obj)

Check if an obj is a root back btn

参数 **menu** -- pointer to the menu

返回 true if it is a root back btn

void **lv_menu_clear_history**(*lv_obj_t* *obj)

Clear menu history

参数 **obj** -- pointer to the menu

Variables

const lv_obj_class_t **lv_menu_class**

const lv_obj_class_t **lv_menu_page_class**

const lv_obj_class_t **lv_menu_cont_class**

const lv_obj_class_t **lv_menu_section_class**

const lv_obj_class_t **lv_menu_separator_class**

const lv_obj_class_t **lv_menu_sidebar_cont_class**

const lv_obj_class_t **lv_menu_main_cont_class**

const lv_obj_class_t **lv_menu_sidebar_header_cont_class**

const lv_obj_class_t **lv_menu_main_header_cont_class**

struct **lv_menu_load_page_event_data_t**

Public Members

lv_obj_t ***menu**

lv_obj_t ***page**

struct **lv_menu_history_t**

Public Members

lv_obj_t ***page**

struct **lv_menu_t**

Public Members

lv_obj_t **obj**

lv_obj_t ***storage**

lv_obj_t ***main**

lv_obj_t ***main_page**

lv_obj_t ***main_header**

lv_obj_t ***main_header_back_btn**

lv_obj_t ***main_header_title**

lv_obj_t ***sidebar**

lv_obj_t ***sidebar_page**

lv_obj_t ***sidebar_header**

lv_obj_t ***sidebar_header_back_btn**

lv_obj_t ***sidebar_header_title**

lv_obj_t ***selected_tab**

lv_ll_t **history_ll**

uint8_t **cur_depth**

uint8_t **prev_depth**

uint8_t **sidebar_generated**

lv_menu_mode_header_t **mode_header**

lv_menu_mode_root_back_btn_t **mode_root_back_btn**

struct **lv_menu_page_t**

Public Members

lv_obj_t **obj**

char ***title**

Meter (lv_meter)

Overview

The Meter widget can visualize data in very flexible ways. In can show arcs, needles, ticks lines and labels.

Parts and Styles

- `LV_PART_MAIN` The background of the Meter. Uses the typical background properties.
- `LV_PART_TICK` The tick lines a labels using the *line* and *text* style properties.
- `LV_PART_INDICATOR` The needle line or image using the *line* and *img* style properties, as well as the background properties to draw a square (or circle) on the pivot of the needles. Padding makes the square larger.
- `LV_PART_ITEMS` The arcs using the *arc* properties.

Usage

Add a scale

First a *Scale* needs to be added to the Meter with `lv_meter_scale_t * scale = lv_meter_add_scale(meter)`. The Scale has minor and major ticks and labels on the major ticks. Later indicators (needles, arcs, tick modifiers) can be added to the meter

Any number of scales can be added to Meter.

The minor tick lines can be configured with: `lv_meter_set_scale_ticks(meter, scale, tick_count, line_width, tick_length, ctick_olor)`.

To add major tick lines use `lv_meter_set_scale_major_ticks(meter, scale, nth_major, tick_width, tick_length, tick_color, label_gap)`. `nth_major` to specify how many minor ticks to skip to draw a major tick.

Labels are added automatically on major ticks with `label_gap` distance from the ticks with text proportionally to the values of the tick line.

`lv_meter_set_scale_range(meter, scale, min, max, angle_range, rotation)` sets the value and angle range of the scale.

Add indicators

Indicators need to be added to a Scale and their value is interpreted in the range of the Scale.

All the indicator add functions return `lv_meter_indicator_t *`.

Needle line

`indic = lv_meter_add_needle_line(meter, scale, line_width, line_color, r_mod)` adds a needle line to a Scale. By default, the length of the line is the same as the scale's radius but `r_mod` changes the length.

`lv_meter_set_indicator_value(meter, indic, value)` sets the value of the indicator.

Needle image

`indic = lv_meter_add_needle_img(meter, scale, img_src, pivot_x, pivot_y)` sets an image that will be used as a needle. `img_src` should be a needle pointing to the right like this `-O--->`. `pivot_x` and `pivot_y` sets the pivot point of the rotation relative to the top left corner of the image.

`lv_meter_set_indicator_value(meter, inidicator, value)` sets the value of the indicator.

Arc

`indic = lv_meter_add_arc(meter, scale, arc_width, arc_color, r_mod)` adds and arc indicator. . By default, the radius of the arc is the same as the scale's radius but `r_mod` changes the radius.

`lv_meter_set_indicator_start_value(meter, indic, value)` and `lv_meter_set_indicator_end_value(meter, inidicator, value)` sets the value of the indicator.

Scale lines (ticks)

`indic = lv_meter_add_scale_lines(meter, scale, color_start, color_end, local, width_mod)` adds an indicator that modifies the ticks lines. If `local` is `true` the ticks' color will be faded from `color_start` to `color_end` in the indicator's start and end value range. If `local` is `false` `color_start` and `color_end` will be mapped to the start and end value of the scale and only a "slice" of that color gradient will be visible in the indicator's start and end value range. `width_mod` modifies the width of the tick lines.

`lv_meter_set_indicator_start_value(meter, inidicator, value)` and `lv_meter_set_indicator_end_value(meter, inidicator, value)` sets the value of the indicator.

Events

- `LV_EVENT_DRAW_PART_BEGIN` and `LV_EVENT_DRAW_PART_END` is sent for the following types:
  - `LV_METER_DRAW_PART_ARC` The arc indicator
    - `part`: `LV_PART_ITEMS`
    - `sub_part_ptr`: pointer to the indicator
    - `arc_dsc`
    - `radius`: radius of the arc
    - `p1` center of the arc
  - `LV_METER_DRAW_PART_NEEDLE_LINE` The needle lines
    - `part`: `LV_PART_ITEMS`
    - `p1`, `p2` points of the line
    - `line_dsc`
    - `sub_part_ptr`: pointer to the indicator
  - `LV_METER_DRAW_PART_NEEDLE_IMG` The needle images
    - `part`: `LV_PART_ITEMS`

        * `p1`, `p2` points of the line
        * `img_dsc`
        * `sub_part_ptr`: pointer to the indicator
    - `LV_METER_DRAW_PART_TICK` The tick lines and labels
        * `part`: `LV_PART_TICKS`
        * `value`: the value of the line
        * `text`: `value` converted to decimal or `NULL` on minor lines
        * `label_dsc`: label draw descriptor or `NULL` on minor lines
        * `line_dsc`:
        * `id`: the index of the line

See the events of the *Base object* too.

Learn more about *Events*.

Keys

No keys are handled by the Meter widget.

Learn more about *Keys*.

Example

Simple meter

```
#include "../../lv_examples.h"
#if LV_USE_METER && LV_BUILD_EXAMPLES

static lv_obj_t * meter;

static void set_value(void * indic, int32_t v)
{
    lv_meter_set_indicator_value(meter, indic, v);
}

/**
 * A simple meter
 */
void lv_example_meter_1(void)
```

(下页继续)

(续上页)

```
{
    meter = lv_meter_create(lv_scr_act());
    lv_obj_center(meter);
    lv_obj_set_size(meter, 200, 200);

    /*Add a scale first*/
    lv_meter_scale_t * scale = lv_meter_add_scale(meter);
    lv_meter_set_scale_ticks(meter, scale, 41, 2, 10, lv_palette_main(LV_PALETTE_
↪GREY));
    lv_meter_set_scale_major_ticks(meter, scale, 8, 4, 15, lv_color_black(), 10);

    lv_meter_indicator_t * indic;

    /*Add a blue arc to the start*/
    indic = lv_meter_add_arc(meter, scale, 3, lv_palette_main(LV_PALETTE_BLUE), 0);
    lv_meter_set_indicator_start_value(meter, indic, 0);
    lv_meter_set_indicator_end_value(meter, indic, 20);

    /*Make the tick lines blue at the start of the scale*/
    indic = lv_meter_add_scale_lines(meter, scale, lv_palette_main(LV_PALETTE_BLUE),␣
↪lv_palette_main(LV_PALETTE_BLUE), false, 0);
    lv_meter_set_indicator_start_value(meter, indic, 0);
    lv_meter_set_indicator_end_value(meter, indic, 20);

    /*Add a red arc to the end*/
    indic = lv_meter_add_arc(meter, scale, 3, lv_palette_main(LV_PALETTE_RED), 0);
    lv_meter_set_indicator_start_value(meter, indic, 80);
    lv_meter_set_indicator_end_value(meter, indic, 100);

    /*Make the tick lines red at the end of the scale*/
    indic = lv_meter_add_scale_lines(meter, scale, lv_palette_main(LV_PALETTE_RED),␣
↪lv_palette_main(LV_PALETTE_RED), false, 0);
    lv_meter_set_indicator_start_value(meter, indic, 80);
    lv_meter_set_indicator_end_value(meter, indic, 100);

    /*Add a needle line indicator*/
    indic = lv_meter_add_needle_line(meter, scale, 4, lv_palette_main(LV_PALETTE_
↪GREY), -10);

    /*Create an animation to set the value*/
    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_exec_cb(&a, set_value);
```

(下页继续)

(续上页)

```
    lv_anim_set_var(&a, indic);
    lv_anim_set_values(&a, 0, 100);
    lv_anim_set_time(&a, 2000);
    lv_anim_set_repeat_delay(&a, 100);
    lv_anim_set_playback_time(&a, 500);
    lv_anim_set_playback_delay(&a, 100);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
    lv_anim_start(&a);
}

#endif
```

```
#!//opt/bin/lv_micropython -i
import utime as time
import lvgl as lv
import display_driver

def set_value(indic, v):
    meter.set_indicator_value(indic, v)

#
# A simple meter
#
meter = lv.meter(lv.scr_act())
meter.center()
meter.set_size(200, 200)

# Add a scale first
scale = meter.add_scale()
meter.set_scale_ticks(scale, 51, 2, 10, lv.palette_main(lv.PALETTE.GREY))
meter.set_scale_major_ticks(scale, 10, 4, 15, lv.color_black(), 10)

indic = lv.meter_indicator_t()

# Add a blue arc to the start
indic = meter.add_arc(scale, 3, lv.palette_main(lv.PALETTE.BLUE), 0)
meter.set_indicator_start_value(indic, 0)
meter.set_indicator_end_value(indic, 20)

# Make the tick lines blue at the start of the scale
indic = meter.add_scale_lines(scale, lv.palette_main(lv.PALETTE.BLUE), lv.palette_
↪main(lv.PALETTE.BLUE), False, 0)
meter.set_indicator_start_value(indic, 0)
```

(下页继续)

(续上页)

```
meter.set_indicator_end_value(indic, 20)

# Add a red arc to the end
indic = meter.add_arc(scale, 3, lv.palette_main(lv.PALETTE.RED), 0)
meter.set_indicator_start_value(indic, 80)
meter.set_indicator_end_value(indic, 100)

# Make the tick lines red at the end of the scale
indic = meter.add_scale_lines(scale, lv.palette_main(lv.PALETTE.RED), lv.palette_
↪main(lv.PALETTE.RED), False, 0)
meter.set_indicator_start_value(indic, 80)
meter.set_indicator_end_value(indic, 100)

# Add a needle line indicator
indic = meter.add_needle_line(scale, 4, lv.palette_main(lv.PALETTE.GREY), -10)

# Create an animation to set the value
a = lv.anim_t()
a.init()
a.set_var(indic)
a.set_values(0, 100)
a.set_time(2000)
a.set_repeat_delay(100)
a.set_playback_time(500)
a.set_playback_delay(100)
a.set_repeat_count(lv.ANIM_REPEAT.INFINITE)
a.set_custom_exec_cb(lambda a,val: set_value(indic,val))
lv.anim_t.start(a)
```

A meter with multiple arcs

```
#include "../../lv_examples.h"
#if LV_USE_METER && LV_BUILD_EXAMPLES

static lv_obj_t * meter;

static void set_value(void * indic, int32_t v)
{
    lv_meter_set_indicator_end_value(meter, indic, v);
}
```

(下页继续)

(续上页)

```
/**
 * A meter with multiple arcs
 */
void lv_example_meter_2(void)
{
    meter = lv_meter_create(lv_scr_act());
    lv_obj_center(meter);
    lv_obj_set_size(meter, 200, 200);

    /*Remove the circle from the middle*/
    lv_obj_remove_style(meter, NULL, LV_PART_INDICATOR);

    /*Add a scale first*/
    lv_meter_scale_t * scale = lv_meter_add_scale(meter);
    lv_meter_set_scale_ticks(meter, scale, 11, 2, 10, lv_palette_main(LV_PALETTE_
↪GREY));
    lv_meter_set_scale_major_ticks(meter, scale, 1, 2, 30, lv_color_hex3(0xeee), 15);
    lv_meter_set_scale_range(meter, scale, 0, 100, 270, 90);

    /*Add a three arc indicator*/
    lv_meter_indicator_t * indic1 = lv_meter_add_arc(meter, scale, 10, lv_palette_
↪main(LV_PALETTE_RED), 0);
    lv_meter_indicator_t * indic2 = lv_meter_add_arc(meter, scale, 10, lv_palette_
↪main(LV_PALETTE_GREEN), -10);
    lv_meter_indicator_t * indic3 = lv_meter_add_arc(meter, scale, 10, lv_palette_
↪main(LV_PALETTE_BLUE), -20);

    /*Create an animation to set the value*/
    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_exec_cb(&a, set_value);
    lv_anim_set_values(&a, 0, 100);
    lv_anim_set_repeat_delay(&a, 100);
    lv_anim_set_playback_delay(&a, 100);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);

    lv_anim_set_time(&a, 2000);
    lv_anim_set_playback_time(&a, 500);
    lv_anim_set_var(&a, indic1);
    lv_anim_start(&a);

    lv_anim_set_time(&a, 1000);
```

(下页继续)

(续上页)

```
    lv_anim_set_playback_time(&a, 1000);
    lv_anim_set_var(&a, indic2);
    lv_anim_start(&a);

    lv_anim_set_time(&a, 1000);
    lv_anim_set_playback_time(&a, 2000);
    lv_anim_set_var(&a, indic3);
    lv_anim_start(&a);
}

#endif
```

```
#!//opt/bin/lv_micropython -i
import utime as time
import lvgl as lv
import display_driver

def set_value(indic,v):
    meter.set_indicator_end_value(indic, v)

#
# A meter with multiple arcs
#

meter = lv.meter(lv.scr_act())
meter.center()
meter.set_size(200, 200)

# Remove the circle from the middle
meter.remove_style(None, lv.PART.INDICATOR)

# Add a scale first
scale = meter.add_scale()
meter.set_scale_ticks(scale, 11, 2, 10, lv.palette_main(lv.PALETTE.GREY))
meter.set_scale_major_ticks(scale, 1, 2, 30, lv.color_hex3(0xeee), 10)
meter.set_scale_range(scale, 0, 100, 270, 90)

# Add a three arc indicator
indic1 = meter.add_arc(scale, 10, lv.palette_main(lv.PALETTE.RED), 0)
indic2 = meter.add_arc(scale, 10, lv.palette_main(lv.PALETTE.GREEN), -10)
indic3 = meter.add_arc(scale, 10, lv.palette_main(lv.PALETTE.BLUE), -20)

# Create an animation to set the value
```

(下页继续)

(续上页)

```
a1 = lv.anim_t()
a1.init()
a1.set_values(0, 100)
a1.set_time(2000)
a1.set_repeat_delay(100)
a1.set_playback_delay(100)
a1.set_playback_time(500)
a1.set_var(indic1)
a1.set_repeat_count(lv.ANIM_REPEAT.INFINITE)
a1.set_custom_exec_cb(lambda a,val: set_value(indic1,val))
lv.anim_t.start(a1)

a2 = lv.anim_t()
a2.init()
a2.set_values(0, 100)
a2.set_time(1000)
a2.set_repeat_delay(100)
a2.set_playback_delay(100)
a2.set_playback_time(1000)
a2.set_var(indic2)
a2.set_repeat_count(lv.ANIM_REPEAT.INFINITE)
a2.set_custom_exec_cb(lambda a,val: set_value(indic2,val))
lv.anim_t.start(a2)

a3 = lv.anim_t()
a3.init()
a3.set_values(0, 100)
a3.set_time(1000)
a3.set_repeat_delay(100)
a3.set_playback_delay(100)
a3.set_playback_time(2000)
a3.set_var(indic3)
a3.set_repeat_count(lv.ANIM_REPEAT.INFINITE)
a3.set_custom_exec_cb(lambda a,val: set_value(indic3,val))
lv.anim_t.start(a3)
```

A clock from a meter

```
#include "../../lv_examples.h"
#if LV_USE_METER && LV_BUILD_EXAMPLES

static lv_obj_t * meter;

static void set_value(void * indic, int32_t v)
{
    lv_meter_set_indicator_end_value(meter, indic, v);
}

/**
 * A clock from a meter
 */
void lv_example_meter_3(void)
{
    meter = lv_meter_create(lv_scr_act());
    lv_obj_set_size(meter, 220, 220);
    lv_obj_center(meter);

    /*Create a scale for the minutes*/
    /*61 ticks in a 360 degrees range (the last and the first line overlaps)*/
    lv_meter_scale_t * scale_min = lv_meter_add_scale(meter);
    lv_meter_set_scale_ticks(meter, scale_min, 61, 1, 10, lv_palette_main(LV_PALETTE_
↪GREY));
    lv_meter_set_scale_range(meter, scale_min, 0, 60, 360, 270);

    /*Create another scale for the hours. It's only visual and contains only major␣
↪ticks*/
    lv_meter_scale_t * scale_hour = lv_meter_add_scale(meter);
    lv_meter_set_scale_ticks(meter, scale_hour, 12, 0, 0, lv_palette_main(LV_PALETTE_
↪GREY));               /*12 ticks*/
    lv_meter_set_scale_major_ticks(meter, scale_hour, 1, 2, 20, lv_color_black(), 10);
↪    /*Every tick is major*/
    lv_meter_set_scale_range(meter, scale_hour, 1, 12, 330, 300);       /*[1..12]␣
↪values in an almost full circle*/

    LV_IMG_DECLARE(img_hand)

    /*Add a the hands from images*/
    lv_meter_indicator_t * indic_min = lv_meter_add_needle_img(meter, scale_min, &img_
↪hand, 5, 5);
    lv_meter_indicator_t * indic_hour = lv_meter_add_needle_img(meter, scale_min, &
↪img_hand, 5, 5);
```

(下页继续)

(续上页)

```
    /*Create an animation to set the value*/
    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_exec_cb(&a, set_value);
    lv_anim_set_values(&a, 0, 60);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
    lv_anim_set_time(&a, 2000);     /*2 sec for 1 turn of the minute hand (1 hour)*/
    lv_anim_set_var(&a, indic_min);
    lv_anim_start(&a);

    lv_anim_set_var(&a, indic_hour);
    lv_anim_set_time(&a, 24000);    /*24 sec for 1 turn of the hour hand*/
    lv_anim_set_values(&a, 0, 60);
    lv_anim_start(&a);
}

#endif
```

```
#!/opt/bin/lv_micropython -i
import utime as time
import lvgl as lv
import display_driver
from imagetools import get_png_info, open_png

# Register PNG image decoder
decoder = lv.img.decoder_create()
decoder.info_cb = get_png_info
decoder.open_cb = open_png

# Create an image from the png file
try:
    with open('../../assets/img_hand_min.png','rb') as f:
        img_hand_min_data = f.read()
except:
    print("Could not find img_hand_min.png")
    sys.exit()

img_hand_min_dsc = lv.img_dsc_t({
  'data_size': len(img_hand_min_data),
  'data': img_hand_min_data
})
```

(下页继续)

(续上页)

```
# Create an image from the png file
try:
    with open('../../assets/img_hand_hour.png','rb') as f:
        img_hand_hour_data = f.read()
except:
    print("Could not find img_hand_hour.png")
    sys.exit()

img_hand_hour_dsc = lv.img_dsc_t({
  'data_size': len(img_hand_hour_data),
  'data': img_hand_hour_data
})

def set_value(indic, v):
    meter.set_indicator_value(indic, v)
#
# A clock from a meter
#

meter = lv.meter(lv.scr_act())
meter.set_size(220, 220)
meter.center()

# Create a scale for the minutes
# 61 ticks in a 360 degrees range (the last and the first line overlaps)
scale_min = meter.add_scale()
meter.set_scale_ticks(scale_min, 61, 1, 10, lv.palette_main(lv.PALETTE.GREY))
meter.set_scale_range(scale_min, 0, 60, 360, 270)

# Create another scale for the hours. It's only visual and contains only major ticks
scale_hour = meter.add_scale()
meter.set_scale_ticks(scale_hour, 12, 0, 0, lv.palette_main(lv.PALETTE.GREY))  # 12␣
↪ticks
meter.set_scale_major_ticks(scale_hour, 1, 2, 20, lv.color_black(), 10)         #␣
↪Every tick is major
meter.set_scale_range(scale_hour, 1, 12, 330, 300)                              # [1..
↪12] values in an almost full circle

#    LV_IMG_DECLARE(img_hand)

# Add the hands from images
indic_min = meter.add_needle_img(scale_min, img_hand_min_dsc, 5, 5)
indic_hour = meter.add_needle_img(scale_min, img_hand_hour_dsc, 5, 5)
```

(下页继续)

(续上页)

```
# Create an animation to set the value
a1 = lv.anim_t()
a1.init()
a1.set_values(0, 60)
a1.set_repeat_count(lv.ANIM_REPEAT.INFINITE)
a1.set_time(2000)        # 2 sec for 1 turn of the minute hand (1 hour)
a1.set_var(indic_min)
a1.set_custom_exec_cb(lambda a1,val: set_value(indic_min,val))
lv.anim_t.start(a1)

a2 = lv.anim_t()
a2.init()
a2.set_var(indic_hour)
a2.set_time(24000)       # 24 sec for 1 turn of the hour hand
a2.set_values(0, 60)
a2.set_custom_exec_cb(lambda a2,val: set_value(indic_hour,val))
lv.anim_t.start(a2)
```

Pie chart

```
#include "../../lv_examples.h"
#if LV_USE_METER && LV_BUILD_EXAMPLES

/**
 * Create a pie chart
 */
void lv_example_meter_4(void)
{
    lv_obj_t * meter = lv_meter_create(lv_scr_act());

    /*Remove the background and the circle from the middle*/
    lv_obj_remove_style(meter, NULL, LV_PART_MAIN);
    lv_obj_remove_style(meter, NULL, LV_PART_INDICATOR);

    lv_obj_set_size(meter, 200, 200);
    lv_obj_center(meter);

    /*Add a scale first with no ticks.*/
    lv_meter_scale_t * scale = lv_meter_add_scale(meter);
    lv_meter_set_scale_ticks(meter, scale, 0, 0, 0, lv_color_black());
```

(下页继续)

(续上页)

```
    lv_meter_set_scale_range(meter, scale, 0, 100, 360, 0);

    /*Add a three arc indicator*/
    lv_coord_t indic_w = 100;
    lv_meter_indicator_t * indic1 = lv_meter_add_arc(meter, scale, indic_w,lv_palette_
↪main(LV_PALETTE_ORANGE), 0);
    lv_meter_set_indicator_start_value(meter, indic1, 0);
    lv_meter_set_indicator_end_value(meter, indic1, 40);

    lv_meter_indicator_t * indic2 = lv_meter_add_arc(meter, scale, indic_w, lv_
↪palette_main(LV_PALETTE_YELLOW), 0);
    lv_meter_set_indicator_start_value(meter, indic2, 40);  /*Start from the␣
↪previous*/
    lv_meter_set_indicator_end_value(meter, indic2, 80);

    lv_meter_indicator_t * indic3 = lv_meter_add_arc(meter, scale, indic_w, lv_
↪palette_main(LV_PALETTE_DEEP_ORANGE), 0);
    lv_meter_set_indicator_start_value(meter, indic3, 80);  /*Start from the␣
↪previous*/
    lv_meter_set_indicator_end_value(meter, indic3, 100);
}

#endif
```

```
#
# Create a pie chart
#

meter = lv.meter(lv.scr_act())

# Remove the background and the circle from the middle
meter.remove_style(None, lv.PART.MAIN)
meter.remove_style(None, lv.PART.INDICATOR)

meter.set_size(200, 200)
meter.center()

# Add a scale first with no ticks.
scale = meter.add_scale()
meter.set_scale_ticks(scale, 0, 0, 0, lv.color_black())
meter.set_scale_range(scale, 0, 100, 360, 0)

# Add a three arc indicator*
```

(下页继续)

(续上页)

```
indic_w = 100
indic1 = meter.add_arc(scale, indic_w,lv.palette_main(lv.PALETTE.ORANGE), 0)
meter.set_indicator_start_value(indic1, 0)
meter.set_indicator_end_value(indic1, 40)

indic2 = meter.add_arc(scale, indic_w, lv.palette_main(lv.PALETTE.YELLOW), 0)
meter.set_indicator_start_value(indic2, 40)  # Start from the previous
meter.set_indicator_end_value(indic2, 80)

indic3 = meter.add_arc(scale, indic_w, lv.palette_main(lv.PALETTE.DEEP_ORANGE), 0)
meter.set_indicator_start_value(indic3, 80)  # Start from the previous
meter.set_indicator_end_value(indic3, 100)
```

API

Typedefs

typedef uint8_t **lv_meter_indicator_type_t**

Enums

enum **[anonymous]**

Values:

enumerator **LV_METER_INDICATOR_TYPE_NEEDLE_IMG**

enumerator **LV_METER_INDICATOR_TYPE_NEEDLE_LINE**

enumerator **LV_METER_INDICATOR_TYPE_SCALE_LINES**

enumerator **LV_METER_INDICATOR_TYPE_ARC**

enum **lv_meter_draw_part_type_t**

`type` field in `lv_obj_draw_part_dsc_t` if `class_p = lv_meter_class` Used in `LV_EVENT_DRAW_PART_BEGIN` and `LV_EVENT_DRAW_PART_END`

Values:

enumerator **LV_METER_DRAW_PART_ARC**

The arc indicator

enumerator **LV_METER_DRAW_PART_NEEDLE_LINE**
The needle lines

enumerator **LV_METER_DRAW_PART_NEEDLE_IMG**
The needle images

enumerator **LV_METER_DRAW_PART_TICK**
The tick lines and labels

Functions

lv_obj_t ***lv_meter_create**(*lv_obj_t* *parent)
Create a Meter object

参数 **parent** -- pointer to an object, it will be the parent of the new bar.

返回 pointer to the created meter

lv_meter_scale_t ***lv_meter_add_scale**(*lv_obj_t* *obj)
Add a new scale to the meter.

注解: Indicators can be attached to scales.

参数 **obj** -- pointer to a meter object

返回 the new scale

void **lv_meter_set_scale_ticks**(*lv_obj_t* *obj, *lv_meter_scale_t* *scale, uint16_t cnt, uint16_t width, uint16_t len, lv_color_t color)
Set the properties of the ticks of a scale

参数

- **obj** -- pointer to a meter object
- **scale** -- pointer to scale (added to `meter`)
- **cnt** -- number of tick lines
- **width** -- width of tick lines
- **len** -- length of tick lines
- **color** -- color of tick lines

void **lv_meter_set_scale_major_ticks**(*lv_obj_t* *obj, *lv_meter_scale_t* *scale, uint16_t nth, uint16_t width, uint16_t len, lv_color_t color, int16_t label_gap)

Make some ”normal” ticks major ticks and set their attributes. Texts with the current value are also added to the major ticks.

参数

- **obj** -- pointer to a meter object
- **scale** -- pointer to scale (added to `meter`)
- **nth** -- make every Nth normal tick major tick. (start from the first on the left)
- **width** -- width of the major ticks
- **len** -- length of the major ticks
- **color** -- color of the major ticks
- **label_gap** -- gap between the major ticks and the labels

void **lv_meter_set_scale_range**(*lv_obj_t* *obj, *lv_meter_scale_t* *scale, int32_t min, int32_t max, uint32_t angle_range, uint32_t rotation)

Set the value and angular range of a scale.

参数

- **obj** -- pointer to a meter object
- **scale** -- pointer to scale (added to `meter`)
- **min** -- the minimum value
- **max** -- the maximal value
- **angle_range** -- the angular range of the scale
- **rotation** -- the angular offset from the 3 o’clock position (clock-wise)

lv_meter_indicator_t ***lv_meter_add_needle_line**(*lv_obj_t* *obj, *lv_meter_scale_t* *scale, uint16_t width, lv_color_t color, int16_t r_mod)

Add a needle line indicator the scale

参数

- **obj** -- pointer to a meter object
- **scale** -- pointer to scale (added to `meter`)
- **width** -- width of the line
- **color** -- color of the line
- **r_mod** -- the radius modifier (added to the scale’s radius) to get the lines length

返回 the new indicator

lv_meter_indicator_t ***lv_meter_add_needle_img**(*lv_obj_t* *obj, *lv_meter_scale_t* *scale, const void *src, lv_coord_t pivot_x, lv_coord_t pivot_y)*

Add a needle image indicator the scale

注解： the needle image should point to the right, like -O-->

参数

- **obj** -- pointer to a meter object
- **scale** -- pointer to scale (added to `meter`)
- **src** -- the image source of the indicator. path or pointer to *lv_img_dsc_t*
- **pivot_x** -- the X pivot point of the needle
- **pivot_y** -- the Y pivot point of the needle

返回 the new indicator

lv_meter_indicator_t ***lv_meter_add_arc**(*lv_obj_t* *obj, *lv_meter_scale_t* *scale, uint16_t width, lv_color_t color, int16_t r_mod)*

Add an arc indicator the scale

参数

- **obj** -- pointer to a meter object
- **scale** -- pointer to scale (added to `meter`)
- **width** -- width of the arc
- **color** -- color of the arc
- **r_mod** -- the radius modifier (added to the scale's radius) to get the outer radius of the arc

返回 the new indicator

lv_meter_indicator_t ***lv_meter_add_scale_lines**(*lv_obj_t* *obj, *lv_meter_scale_t* *scale, lv_color_t color_start, lv_color_t color_end, bool local, int16_t width_mod)*

Add a scale line indicator the scale. It will modify the ticks.

参数

- **obj** -- pointer to a meter object
- **scale** -- pointer to scale (added to `meter`)
- **color_start** -- the start color
- **color_end** -- the end color

- **local** -- tell how to map start and end color. true: the indicator's start and end_value; false: the scale's min max value
- **width_mod** -- add this the affected tick's width

返回 the new indicator

void **lv_meter_set_indicator_value**(*lv_obj_t* *obj, *lv_meter_indicator_t* *indic, int32_t value)

Set the value of the indicator. It will set start and and value to the same value

参数

- **obj** -- pointer to a meter object
- **indic** -- pointer to an indicator
- **value** -- the new value

void **lv_meter_set_indicator_start_value**(*lv_obj_t* *obj, *lv_meter_indicator_t* *indic, int32_t value)

Set the start value of the indicator.

参数

- **obj** -- pointer to a meter object
- **indic** -- pointer to an indicator
- **value** -- the new value

void **lv_meter_set_indicator_end_value**(*lv_obj_t* *obj, *lv_meter_indicator_t* *indic, int32_t value)

Set the start value of the indicator.

参数

- **obj** -- pointer to a meter object
- **indic** -- pointer to an indicator
- **value** -- the new value

Variables

const lv_obj_class_t **lv_meter_class**

struct **lv_meter_scale_t**

Public Members

lv_color_t **tick_color**

uint16_t **tick_cnt**

uint16_t **tick_length**

uint16_t **tick_width**

lv_color_t **tick_major_color**

uint16_t **tick_major_nth**

uint16_t **tick_major_length**

uint16_t **tick_major_width**

int16_t **label_gap**

int16_t **label_color**

int32_t **min**

int32_t **max**

int16_t **r_mod**

uint16_t **angle_range**

int16_t **rotation**

struct **lv_meter_indicator_t**

Public Members

lv_meter_scale_t ***scale**

lv_meter_indicator_type_t **type**

lv_opa_t **opa**

int32_t **start_value**

int32_t **end_value**

const void ***src**

lv_point_t **pivot**

struct *lv_meter_indicator_t*::[anonymous]::[anonymous] **needle_img**

uint16_t **width**

int16_t **r_mod**

lv_color_t **color**

struct *lv_meter_indicator_t*::[anonymous]::[anonymous] **needle_line**

struct *lv_meter_indicator_t*::[anonymous]::[anonymous] **arc**

int16_t **width_mod**

lv_color_t **color_start**

lv_color_t **color_end**

uint8_t **local_grad**

struct *lv_meter_indicator_t*::[anonymous]::[anonymous] **scale_lines**

union *lv_meter_indicator_t*::[anonymous] **type_data**

struct **lv_meter_t**

Public Members

lv_obj_t **obj**

lv_ll_t **scale_ll**

lv_ll_t **indicator_ll**

Message box (lv_msgbox)

Overview

The Message boxes act as pop-ups. They are built from a background container, a title, an optional close button, a text and optional buttons.

The text will be broken into multiple lines automatically and the height will be set automatically to include the text and the buttons.

The message box can be modal (blocking clicks on the rest of the screen) or not modal.

Parts and Styles

The message box is built from other widgets, so you can check these widgets' documentation for details.

- Background: *lv_obj*
- Close button: *lv_btn*
- Title and text: *lv_label*
- Buttons: *lv_btnmatrix*

Usage

Create a message box

`lv_msgbox_create(parent, title, txt, btn_txts[], add_close_btn)` creates a message box.

If `parent` is `NULL` the message box will be modal. `title` and `txt` are strings for the title and the text. `btn_txts[]` is an array with the buttons' text. E.g. `const char * btn_txts[] = {"Ok", "Cancel", NULL}`. `add_colse_btn` can be `true` or `false` to add/don't add a close button.

Get the parts

The building blocks of the message box can be obtained using the following functions:

```
lv_obj_t * lv_msgbox_get_title(lv_obj_t * mbox);
lv_obj_t * lv_msgbox_get_close_btn(lv_obj_t * mbox);
lv_obj_t * lv_msgbox_get_text(lv_obj_t * mbox);
lv_obj_t * lv_msgbox_get_btns(lv_obj_t * mbox);
```

Close the message box

`lv_msgbox_close(msgbox)` closes (deletes) the message box.

Events

- `LV_EVENT_VALUE_CHANGED` is sent by the buttons if one of them is clicked. `LV_OBJ_FLAG_EVENT_BUBBLE` is enabled on the buttons so you can add events to the message box itself. In the event handler, `lv_event_get_target(e)` will return the button matrix and `lv_event_get_current_target(e)` will return the message box. `lv_msgbox_get_active_btn(msgbox)` and `lv_msgbox_get_active_btn_text(msgbox)` can be used to get the index and text of the clicked button.

Learn more about *Events*.

Keys

Keys have effect on the close button and button matrix. You can add them manually to a group if required.

Learn more about *Keys*.

Example

Simple Message box

```c
#include "../../lv_examples.h"
#if LV_USE_MSGBOX && LV_BUILD_EXAMPLES

static void event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_current_target(e);
    LV_LOG_USER("Button %s clicked", lv_msgbox_get_active_btn_text(obj));
}

void lv_example_msgbox_1(void)
{
    static const char * btns[] ={"Apply", "Close", ""};

    lv_obj_t * mbox1 = lv_msgbox_create(NULL, "Hello", "This is a message box with␣
↪two buttons.", btns, true);
    lv_obj_add_event_cb(mbox1, event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    lv_obj_center(mbox1);
}

#endif
```

```python
def event_cb(e):
    mbox = e.get_current_target()
    print("Button %s clicked" % mbox.get_active_btn_text())

btns = ["Apply", "Close", ""]

mbox1 = lv.msgbox(lv.scr_act(), "Hello", "This is a message box with two buttons.",␣
↪btns, True)
mbox1.add_event_cb(event_cb, lv.EVENT.VALUE_CHANGED, None)
```

(下页继续)

(续上页)

```
mbox1.center()
```

API

Functions

lv_obj_t ***lv_msgbox_create**(*lv_obj_t* *parent, const char *title, const char *txt, const char *btn_txts[], bool add_close_btn)

Create a message box object

参数

- **parent** -- pointer to parent or NULL to create a full screen modal message box
- **title** -- the title of the message box
- **txt** -- the text of the message box
- **btn_txts** -- the buttons as an array of texts terminated by an "" element. E.g. {"btn1", "btn2", ""}
- **add_close_btn** -- true: add a close button

返回 pointer to the message box object

lv_obj_t ***lv_msgbox_get_title**(*lv_obj_t* *obj)

lv_obj_t ***lv_msgbox_get_close_btn**(*lv_obj_t* *obj)

lv_obj_t ***lv_msgbox_get_text**(*lv_obj_t* *obj)

lv_obj_t ***lv_msgbox_get_content**(*lv_obj_t* *obj)

lv_obj_t ***lv_msgbox_get_btns**(*lv_obj_t* *obj)

uint16_t **lv_msgbox_get_active_btn**(*lv_obj_t* *mbox)

Get the index of the selected button

参数 **mbox** -- message box object

返回 index of the button (LV_BTNMATRIX_BTN_NONE: if unset)

const char ***lv_msgbox_get_active_btn_text**(*lv_obj_t* *mbox)

void **lv_msgbox_close**(*lv_obj_t* *mbox)

void **lv_msgbox_close_async**(*lv_obj_t* *mbox)

Variables

const lv_obj_class_t **lv_msgbox_class**

const lv_obj_class_t **lv_msgbox_content_class**

const lv_obj_class_t **lv_msgbox_backdrop_class**

struct **lv_msgbox_t**

Public Members

lv_obj_t **obj**

lv_obj_t ***title**

lv_obj_t ***close_btn**

lv_obj_t ***content**

lv_obj_t ***text**

lv_obj_t ***btns**

Span (lv_span)

Overview

A spangroup is the object that is used to display rich text. Different from the label object, `spangroup` can render text styled with different fonts, colors, and sizes into the spangroup object.

Parts and Styles

- `LV_PART_MAIN` The spangroup has only one part.

Usage

Set text and style

The spangroup object uses span to describe text and text style. so, first we need to create `span` descriptor using `lv_span_t * span = lv_spangroup_new_span(spangroup)`. Then use `lv_span_set_text(span, "text")` to set text. The style of the span is configured as with a normal style object by using its `style` member, eg:`lv_style_set_text_color(&span->style, lv_palette_main(LV_PALETTE_RED))`.

If spangroup object `mode != LV_SPAN_MODE_FIXED` you must call `lv_spangroup_refr_mode()` after you have modified `span` style(eg:set text, changed the font size, del span).

Retrieving a span child

Spangroups store their children differently from normal objects, so normal functions for getting children won't work.

`lv_spangroup_get_child(spangroup, id)` will return a pointer to the child span at index `id`. In addition, `id` can be negative to index from the end of the spangroup where `-1` is the youngest child, `-2` is second youngest, etc.

e.g. `lv_span_t* span = lv_spangroup_get_child(spangroup, 0)` will return the first child of the spangroup. `lv_span_t* span = lv_spangroup_get_child(spangroup, -1)` will return the last (or most recent) child.

Child Count

Use the function `lv_spangroup_get_child_cnt(spangroup)` to get back the number of spans the group is maintaining.

e.g. `uint32_t size = lv_spangroup_get_child_cnt(spangroup)`

Text align

like label object, the spangroup can be set to one the following modes:

- `LV_TEXT_ALIGN_LEFT` Align text to left.
- `LV_TEXT_ALIGN_CENTER` Align text to center.
- `LV_TEXT_ALIGN_RIGHT` Align text to right.
- `LV_TEXT_ALIGN_AUTO` Align text auto.

use function `lv_spangroup_set_align(spangroup, LV_TEXT_ALIGN_CENTER)` to set text align.

Modes

The spangroup can be set to one the following modes:

- `LV_SPAN_MODE_FIXED` fixes the object size.
- `LV_SPAN_MODE_EXPAND` Expand the object size to the text size but stay on a single line.
- `LV_SPAN_MODE_BREAK` Keep width, break the too long lines and auto expand height.

Use `lv_spangroup_set_mode(spangroup, LV_SPAN_MODE_BREAK)` to set object mode.

Overflow

The spangroup can be set to one the following modes:

- `LV_SPAN_OVERFLOW_CLIP` truncates the text at the limit of the area.
- `LV_SPAN_OVERFLOW_ELLIPSIS` will display an ellipsis(`...`) when text overflows the area.

Use `lv_spangroup_set_overflow(spangroup, LV_SPAN_OVERFLOW_CLIP)` to set object overflow mode.

first line indent

Use `lv_spangroup_set_indent(spangroup, 20)` to set the indent of the first line. all modes support pixel units, in addition to LV_SPAN_MODE_FIXED and LV_SPAN_MODE_BREAK mode supports percentage units too.

Events

No special events are sent by this widget.

Learn more about *Events*.

Keys

No *Keys* are processed by the object type.

Learn more about *Keys*.

Example

Span with custom styles

```
#include "../../lv_examples.h"
#if LV_USE_SPAN && LV_BUILD_EXAMPLES

/**
 * Create span.
 */
void lv_example_span_1(void)
{
    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_border_width(&style, 1);
    lv_style_set_border_color(&style, lv_palette_main(LV_PALETTE_ORANGE));
    lv_style_set_pad_all(&style, 2);

    lv_obj_t * spans = lv_spangroup_create(lv_scr_act());
    lv_obj_set_width(spans, 300);
    lv_obj_set_height(spans,300);
    lv_obj_center(spans);
    lv_obj_add_style(spans, &style, 0);

    lv_spangroup_set_align(spans, LV_TEXT_ALIGN_LEFT);
    lv_spangroup_set_overflow(spans, LV_SPAN_OVERFLOW_CLIP);
    lv_spangroup_set_indent(spans, 20);
    lv_spangroup_set_mode(spans, LV_SPAN_MODE_BREAK);

    lv_span_t * span = lv_spangroup_new_span(spans);
    lv_span_set_text(span, "China is a beautiful country.");
    lv_style_set_text_color(&span->style, lv_palette_main(LV_PALETTE_RED));
    lv_style_set_text_decor(&span->style, LV_TEXT_DECOR_STRIKETHROUGH | LV_TEXT_DECOR_
↪UNDERLINE);
    lv_style_set_text_opa(&span->style, LV_OPA_50);

    span = lv_spangroup_new_span(spans);
    lv_span_set_text_static(span, "good good study, day day up.");
#if LV_FONT_MONTSERRAT_24
    lv_style_set_text_font(&span->style,  &lv_font_montserrat_24);
#endif
    lv_style_set_text_color(&span->style, lv_palette_main(LV_PALETTE_GREEN));

    span = lv_spangroup_new_span(spans);
```

(下页继续)

(续上页)

```
    lv_span_set_text_static(span, "LVGL is an open-source graphics library.");
    lv_style_set_text_color(&span->style, lv_palette_main(LV_PALETTE_BLUE));

    span = lv_spangroup_new_span(spans);
    lv_span_set_text_static(span, "the boy no name.");
    lv_style_set_text_color(&span->style, lv_palette_main(LV_PALETTE_GREEN));
#if LV_FONT_MONTSERRAT_20
    lv_style_set_text_font(&span->style, &lv_font_montserrat_20);
#endif
    lv_style_set_text_decor(&span->style, LV_TEXT_DECOR_UNDERLINE);

    span = lv_spangroup_new_span(spans);
    lv_span_set_text(span, "I have a dream that hope to come true.");

    lv_spangroup_refr_mode(spans);
}

#endif
```

```
#
# Create span
#
style = lv.style_t()
style.init()
style.set_border_width(1)
style.set_border_color(lv.palette_main(lv.PALETTE.ORANGE))
style.set_pad_all(2)

spans = lv.spangroup(lv.scr_act())
spans.set_width(300)
spans.set_height(300)
spans.center()
spans.add_style(style, 0)

spans.set_align(lv.TEXT_ALIGN.LEFT)
spans.set_overflow(lv.SPAN_OVERFLOW.CLIP)
spans.set_indent(20)
spans.set_mode(lv.SPAN_MODE.BREAK)

span = spans.new_span()
span.set_text("china is a beautiful country.")
span.style.set_text_color(lv.palette_main(lv.PALETTE.RED))
span.style.set_text_decor(lv.TEXT_DECOR.STRIKETHROUGH | lv.TEXT_DECOR.UNDERLINE)
```

(下页继续)

(续上页)

```
span.style.set_text_opa(lv.OPA._30)

span = spans.new_span()
span.set_text_static("good good study, day day up.")
#if LV_FONT_MONTSERRAT_24
#    lv_style_set_text_font(&span->style,  &lv_font_montserrat_24);
#endif
span.style.set_text_color(lv.palette_main(lv.PALETTE.GREEN))

span = spans.new_span()
span.set_text_static("LVGL is an open-source graphics library.")
span.style.set_text_color(lv.palette_main(lv.PALETTE.BLUE))

span = spans.new_span()
span.set_text_static("the boy no name.")
span.style.set_text_color(lv.palette_main(lv.PALETTE.GREEN))
#if LV_FONT_MONTSERRAT_20
#    lv_style_set_text_font(&span->style, &lv_font_montserrat_20);
#endif
span.style.set_text_decor(lv.TEXT_DECOR.UNDERLINE)

span = spans.new_span()
span.set_text("I have a dream that hope to come true.")

spans.refr_mode()

# lv_span_del(spans, span);
# lv_obj_del(spans);
```

API

Typedefs

typedef uint8_t **lv_span_overflow_t**

typedef uint8_t **lv_span_mode_t**

Enums

enum **[anonymous]**
Values:

enumerator **LV_SPAN_OVERFLOW_CLIP**

enumerator **LV_SPAN_OVERFLOW_ELLIPSIS**

enum **[anonymous]**
Values:

enumerator **LV_SPAN_MODE_FIXED**
fixed the obj size

enumerator **LV_SPAN_MODE_EXPAND**
Expand the object size to the text size

enumerator **LV_SPAN_MODE_BREAK**
Keep width, break the too long lines and expand height

Functions

lv_obj_t ***lv_spangroup_create**(*lv_obj_t* *par)
Create a spangroup object

参数 **par** -- pointer to an object, it will be the parent of the new spangroup

返回 pointer to the created spangroup

lv_span_t ***lv_spangroup_new_span**(*lv_obj_t* *obj)
Create a span string descriptor and add to spangroup.

参数 **obj** -- pointer to a spangroup object.

返回 pointer to the created span.

void **lv_spangroup_del_span**(*lv_obj_t* *obj, *lv_span_t* *span)
Remove the span from the spangroup and free memory.

参数

- **obj** -- pointer to a spangroup object.
- **span** -- pointer to a span.

void **lv_span_set_text**(*lv_span_t* *span, const char *text)
Set a new text for a span. Memory will be allocated to store the text by the span.

参数

- **span** -- pointer to a span.
- **text** -- pointer to a text.

void **lv_span_set_text_static**(*lv_span_t* *span, const char *text)

Set a static text. It will not be saved by the span so the 'text' variable has to be 'alive' while the span exist.

参数

- **span** -- pointer to a span.
- **text** -- pointer to a text.

void **lv_spangroup_set_align**(*lv_obj_t* *obj, lv_text_align_t align)

Set the align of the spangroup.

参数

- **obj** -- pointer to a spangroup object.
- **align** -- see lv_text_align_t for details.

void **lv_spangroup_set_overflow**(*lv_obj_t* *obj, *lv_span_overflow_t* overflow)

Set the overflow of the spangroup.

参数

- **obj** -- pointer to a spangroup object.
- **overflow** -- see lv_span_overflow_t for details.

void **lv_spangroup_set_indent**(*lv_obj_t* *obj, lv_coord_t indent)

Set the indent of the spangroup.

参数

- **obj** -- pointer to a spangroup object.
- **indent** -- The first line indentation

void **lv_spangroup_set_mode**(*lv_obj_t* *obj, *lv_span_mode_t* mode)

Set the mode of the spangroup.

参数

- **obj** -- pointer to a spangroup object.
- **mode** -- see lv_span_mode_t for details.

lv_span_t ***lv_spangroup_get_child**(const *lv_obj_t* *obj, int32_t id)

Get a spangroup child by its index.

参数

- **obj** -- The spangroup object

- **id** -- the index of the child. 0: the oldest (firstly created) child 1: the second oldest child count-1: the youngest -1: the youngest -2: the second youngest

返回 The child span at index `id`, or NULL if the ID does not exist

uint32_t **lv_spangroup_get_child_cnt**(const *lv_obj_t* *obj)

参数 **obj** -- The spangroup object to get the child count of.

返回 The span count of the spangroup.

lv_text_align_t **lv_spangroup_get_align**(*lv_obj_t* *obj)

get the align of the spangroup.

参数 **obj** -- pointer to a spangroup object.

返回 the align value.

lv_span_overflow_t **lv_spangroup_get_overflow**(*lv_obj_t* *obj)

get the overflow of the spangroup.

参数 **obj** -- pointer to a spangroup object.

返回 the overflow value.

lv_coord_t **lv_spangroup_get_indent**(*lv_obj_t* *obj)

get the indent of the spangroup.

参数 **obj** -- pointer to a spangroup object.

返回 the indent value.

lv_span_mode_t **lv_spangroup_get_mode**(*lv_obj_t* *obj)

get the mode of the spangroup.

参数 **obj** -- pointer to a spangroup object.

lv_coord_t **lv_spangroup_get_max_line_h**(*lv_obj_t* *obj)

get max line height of all span in the spangroup.

参数 **obj** -- pointer to a spangroup object.

uint32_t **lv_spangroup_get_expand_width**(*lv_obj_t* *obj, uint32_t max_width)

get the text content width when all span of spangroup on a line.

参数

- **obj** -- pointer to a spangroup object.
- **max_width** -- if text content width >= max_width, return max_width to reduce computation, if max_width == 0, returns the text content width.

返回 text content width or max_width.

lv_coord_t **lv_spangroup_get_expand_height**(*lv_obj_t* *obj, lv_coord_t width)

get the text content height with width fixed.

参数 obj -- pointer to a spangroup object.

void **lv_spangroup_refr_mode**(*lv_obj_t* *obj)

update the mode of the spangroup.

参数 obj -- pointer to a spangroup object.

Variables

const lv_obj_class_t **lv_spangroup_class**

struct **lv_span_t**

Public Members

char ***txt**

lv_obj_t ***spangroup**

lv_style_t **style**

uint8_t **static_flag**

struct **lv_spangroup_t**

#include <lv_span.h> Data of label

Public Members

lv_obj_t **obj**

lv_coord_t **indent**

lv_coord_t **cache_w**

lv_coord_t **cache_h**

lv_ll_t **child_ll**

uint8_t **mode**

uint8_t **overflow**

uint8_t **refresh**

Spinbox (lv_spinbox)

Overview

The Spinbox contains a number as text which can be increased or decreased by *Keys* or API functions. Under the hood the Spinbox is a modified *Text area*.

Parts and Styles

The parts of the Spinbox are identical to the *Text area*.

Value, range and step

`lv_spinbox_set_value(spinbox, 1234)` sets a new value on the Spinbox.

`lv_spinbox_increment(spinbox)` and `lv_spinbox_decrement(spinbox)` increments/decrements the value of the Spinbox according to the currently selected digit.

`lv_spinbox_set_range(spinbox, -1000, 2500)` sets a range. If the value is changed by `lv_spinbox_set_value`, by *Keys*,`lv_spinbox_increment/decrement` this range will be respected.

`lv_spinbox_set_step(spinbox, 100)` sets which digits to change on increment/decrement. Only multiples of ten can be set, and not for example 3.

`lv_spinbox_set_pos(spinbox, 1)` sets the cursor to a specific digit to change on increment/decrement. For example position '0' sets the cursor to the least significant digit.

If an encoder is used as input device, the selected digit is shifted to the right by default whenever the encoder button is clicked. To change this behaviour to shifting to the left, the `lv_spinbox_set_digit_step_direction(spinbox, LV_DIR_LEFT)` can be used

Format

`lv_spinbox_set_digit_format(spinbox, digit_count, separator_position)` sets the number format. `digit_count` is the number of digits excluding the decimal separator and the sign. `separator_position` is the number of digits before the decimal point. If 0, no decimal point is displayed.

Rollover

`lv_spinbox_set_rollover(spinbox, true/false)` enables/disabled rollover mode. If either the minimum or maximum value is reached with rollover enabled, the value will change to the other limit. If rollover is disabled the value will remain at the minimum or maximum value.

Events

- `LV_EVENT_VALUE_CHANGED` Sent when the value has changed.

See the events of the *Text area* too.

Learn more about *Events*.

Keys

- `LV_KEY_LEFT/RIGHT` With *Keypad* move the cursor left/right. With *Encoder* decrement/increment the selected digit.
- `LV_KEY_UP/DOWN` With *Keypad* and *Encoder* increment/decrement the value.
- `LV_KEY_ENTER` With *Encoder* got the net digit. Jump to the first after the last.

Example

Simple Spinbox

```
#include "../../lv_examples.h"
#if LV_USE_SPINBOX && LV_BUILD_EXAMPLES

static lv_obj_t * spinbox;


static void lv_spinbox_increment_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    if(code == LV_EVENT_SHORT_CLICKED || code  == LV_EVENT_LONG_PRESSED_REPEAT) {
        lv_spinbox_increment(spinbox);
    }
}

static void lv_spinbox_decrement_event_cb(lv_event_t * e)
{
```

(下页继续)

(续上页)

```
    lv_event_code_t code = lv_event_get_code(e);
    if(code == LV_EVENT_SHORT_CLICKED || code == LV_EVENT_LONG_PRESSED_REPEAT) {
        lv_spinbox_decrement(spinbox);
    }
}


void lv_example_spinbox_1(void)
{
    spinbox = lv_spinbox_create(lv_scr_act());
    lv_spinbox_set_range(spinbox, -1000, 25000);
    lv_spinbox_set_digit_format(spinbox, 5, 2);
    lv_spinbox_step_prev(spinbox);
    lv_obj_set_width(spinbox, 100);
    lv_obj_center(spinbox);

    lv_coord_t h = lv_obj_get_height(spinbox);

    lv_obj_t * btn = lv_btn_create(lv_scr_act());
    lv_obj_set_size(btn, h, h);
    lv_obj_align_to(btn, spinbox, LV_ALIGN_OUT_RIGHT_MID, 5, 0);
    lv_obj_set_style_bg_img_src(btn, LV_SYMBOL_PLUS, 0);
    lv_obj_add_event_cb(btn, lv_spinbox_increment_event_cb, LV_EVENT_ALL,  NULL);

    btn = lv_btn_create(lv_scr_act());
    lv_obj_set_size(btn, h, h);
    lv_obj_align_to(btn, spinbox, LV_ALIGN_OUT_LEFT_MID, -5, 0);
    lv_obj_set_style_bg_img_src(btn, LV_SYMBOL_MINUS, 0);
    lv_obj_add_event_cb(btn, lv_spinbox_decrement_event_cb, LV_EVENT_ALL, NULL);
}

#endif
```

```
def increment_event_cb(e):
    code = e.get_code()
    if code == lv.EVENT.SHORT_CLICKED or code  == lv.EVENT.LONG_PRESSED_REPEAT:
        spinbox.increment()

def decrement_event_cb(e):
    code = e.get_code()
    if code == lv.EVENT.SHORT_CLICKED or code == lv.EVENT.LONG_PRESSED_REPEAT:
        spinbox.decrement()
```

(下页继续)

(续上页)

```
spinbox = lv.spinbox(lv.scr_act())
spinbox.set_range(-1000, 25000)
spinbox.set_digit_format(5, 2)
spinbox.step_prev()
spinbox.set_width(100)
spinbox.center()

h = spinbox.get_height()

btn = lv.btn(lv.scr_act())
btn.set_size(h, h)
btn.align_to(spinbox, lv.ALIGN.OUT_RIGHT_MID, 5, 0)
btn.set_style_bg_img_src(lv.SYMBOL.PLUS, 0)
btn.add_event_cb(increment_event_cb, lv.EVENT.ALL,  None)

btn = lv.btn(lv.scr_act())
btn.set_size(h, h)
btn.align_to(spinbox, lv.ALIGN.OUT_LEFT_MID, -5, 0)
btn.set_style_bg_img_src(lv.SYMBOL.MINUS, 0)
btn.add_event_cb(decrement_event_cb, lv.EVENT.ALL, None)
```

API

Functions

lv_obj_t ***lv_spinbox_create**(*lv_obj_t* *parent)

Create a Spinbox object

参数 **parent** -- pointer to an object, it will be the parent of the new spinbox

返回 pointer to the created spinbox

void **lv_spinbox_set_value**(*lv_obj_t* *obj, int32_t i)

Set spinbox value

参数

- **obj** -- pointer to spinbox
- **i** -- value to be set

void **lv_spinbox_set_rollover**(*lv_obj_t* *obj, bool b)

Set spinbox rollover function

参数

- **obj** -- pointer to spinbox

- **b** -- true or false to enable or disable (default)

void **lv_spinbox_set_digit_format**(*lv_obj_t* *obj, uint8_t digit_count, uint8_t separator_position)

Set spinbox digit format (digit count and decimal format)

参数

- **obj** -- pointer to spinbox
- **digit_count** -- number of digit excluding the decimal separator and the sign
- **separator_position** -- number of digit before the decimal point. If 0, decimal point is not shown

void **lv_spinbox_set_step**(*lv_obj_t* *obj, uint32_t step)

Set spinbox step

参数

- **obj** -- pointer to spinbox
- **step** -- steps on increment/decrement. Can be 1, 10, 100, 1000, etc the digit that will change.

void **lv_spinbox_set_range**(*lv_obj_t* *obj, int32_t range_min, int32_t range_max)

Set spinbox value range

参数

- **obj** -- pointer to spinbox
- **range_min** -- maximum value, inclusive
- **range_max** -- minimum value, inclusive

void **lv_spinbox_set_pos**(*lv_obj_t* *obj, uint8_t pos)

Set cursor position to a specific digit for edition

参数

- **obj** -- pointer to spinbox
- **pos** -- selected position in spinbox

void **lv_spinbox_set_digit_step_direction**(*lv_obj_t* *obj, lv_dir_t direction)

Set direction of digit step when clicking an encoder button while in editing mode

参数

- **obj** -- pointer to spinbox
- **direction** -- the direction (LV_DIR_RIGHT or LV_DIR_LEFT)

bool **lv_spinbox_get_rollover**(*lv_obj_t* *obj)

Get spinbox rollover function status

参数 **obj** -- pointer to spinbox

int32_t **lv_spinbox_get_value**(*lv_obj_t* *obj)

Get the spinbox numeral value (user has to convert to float according to its digit format)

参数 obj -- pointer to spinbox

返回 value integer value of the spinbox

int32_t **lv_spinbox_get_step**(*lv_obj_t* *obj)

Get the spinbox step value (user has to convert to float according to its digit format)

参数 obj -- pointer to spinbox

返回 value integer step value of the spinbox

void **lv_spinbox_step_next**(*lv_obj_t* *obj)

Select next lower digit for edition by dividing the step by 10

参数 obj -- pointer to spinbox

void **lv_spinbox_step_prev**(*lv_obj_t* *obj)

Select next higher digit for edition by multiplying the step by 10

参数 obj -- pointer to spinbox

void **lv_spinbox_increment**(*lv_obj_t* *obj)

Increment spinbox value by one step

参数 obj -- pointer to spinbox

void **lv_spinbox_decrement**(*lv_obj_t* *obj)

Decrement spinbox value by one step

参数 obj -- pointer to spinbox

Variables

const lv_obj_class_t **lv_spinbox_class**

struct **lv_spinbox_t**

Public Members

lv_textarea_t **ta**

int32_t **value**

int32_t **range_max**

int32_t **range_min**

int32_t **step**

uint16_t **digit_count**

uint16_t **dec_point_pos**

uint16_t **rollover**

uint16_t **digit_step_dir**

Example

Spinner (lv_spinner)

Overview

The Spinner object is a spinning arc over a ring.

Parts and Styles

The parts are identical to the parts of *lv_arc*.

Usage

Create a spinner

To create a spinner use `lv_spinner_create(parent, spin_time, arc_length)`. `spin time` sets the spin time in milliseconds, `arc_length` sets the length of the spinning arc in degrees.

Events

No special events are sent to the Spinner.

See the events of the *Arc* too.

Learn more about *Events*.

Keys

No *Keys* are processed by the object type.

Learn more about *Keys*.

Example

Simple spinner

```
#include "../../lv_examples.h"
#if LV_USE_SPINNER && LV_BUILD_EXAMPLES

void lv_example_spinner_1(void)
{
    /*Create a spinner*/
    lv_obj_t * spinner = lv_spinner_create(lv_scr_act(), 1000, 60);
    lv_obj_set_size(spinner, 100, 100);
    lv_obj_center(spinner);
}

#endif
```

```
# Create a spinner
spinner = lv.spinner(lv.scr_act(), 1000, 60)
spinner.set_size(100, 100)
spinner.center()
```

API

Functions

lv_obj_t ***lv_spinner_create**(*lv_obj_t* *parent, uint32_t time, uint32_t arc_length)

Variables

const lv_obj_class_t **lv_spinner_class**

Tabview (lv_tabview)

Overview

The Tab view object can be used to organize content in tabs. The Tab view is built from other widgets:

- Main container: *lv_obj*)
  - Tab buttons: *lv_btnmatrix*
  - Container for the tabs: *lv_obj*
    - Content of the tabs: *lv_obj*

The tab buttons can be positioned on the top, bottom, left and right side of the Tab view.

A new tab can be selected either by clicking on a tab button or by sliding horizontally on the content.

Parts and Styles

There are no special parts on the Tab view but the `lv_obj` and `lv_btnnmatrix` widgets are used to create the Tab view.

Usage

Create a Tab view

`lv_tabview_create(parent, tab_pos, tab_size);` creates a new empty Tab view. `tab_pos` can be `LV_DIR_TOP/BOTTOM/LEFT/RIGHT` to position the tab buttons to a side. `tab_size` is the height (in case of `LV_DIR_TOP/BOTTOM`) or width (in case of `LV_DIR_LEFT/RIGHT`) tab buttons.

Add tabs

New tabs can be added with `lv_tabview_add_tab(tabview, "Tab name")`. This will return a pointer to an *lv_obj* object where the tab's content can be created.

Change tab

To select a new tab you can:

- Click on its tab button
- Slide horizontally
- Use `lv_tabview_set_act(tabview, id, LV_ANIM_ON/OFF)` function

Get the parts

`lv_tabview_get_content(tabview)` returns the container for the tabs, `lv_tabview_get_tab_btns(tabview)` returns the Tab buttons object which is a *Button matrix*.

Events

- `LV_EVENT_VALUE_CHANGED` Sent when a new tab is selected by sliding or clicking the tab button. `lv_tabview_get_tab_act(tabview)` returns the zero based index of the current tab.

Learn more about *Events*.

Keys

Keys have effect only on the tab buttons (Button matrix). Add manually to a group if required.

Learn more about *Keys*.

Example

Simple Tabview

```
#include "../../lv_examples.h"
#if LV_USE_TABVIEW && LV_BUILD_EXAMPLES

void lv_example_tabview_1(void)
{
    /*Create a Tab view object*/
    lv_obj_t *tabview;
    tabview = lv_tabview_create(lv_scr_act(), LV_DIR_TOP, 50);

    /*Add 3 tabs (the tabs are page (lv_page) and can be scrolled*/
    lv_obj_t *tab1 = lv_tabview_add_tab(tabview, "Tab 1");
```

(下页继续)

(续上页)

```
    lv_obj_t *tab2 = lv_tabview_add_tab(tabview, "Tab 2");
    lv_obj_t *tab3 = lv_tabview_add_tab(tabview, "Tab 3");

    /*Add content to the tabs*/
    lv_obj_t * label = lv_label_create(tab1);
    lv_label_set_text(label, "This the first tab\n\n"
                             "If the content\n"
                             "of a tab\n"
                             "becomes too\n"
                             "longer\n"
                             "than the\n"
                             "container\n"
                             "then it\n"
                             "automatically\n"
                             "becomes\n"
                             "scrollable.\n"
                             "\n"
                             "\n"
                             "\n"
                             "Can you see it?");

    label = lv_label_create(tab2);
    lv_label_set_text(label, "Second tab");

    label = lv_label_create(tab3);
    lv_label_set_text(label, "Third tab");

    lv_obj_scroll_to_view_recursive(label, LV_ANIM_ON);

}
#endif
```

```
# Create a Tab view object
tabview = lv.tabview(lv.scr_act(), lv.DIR.TOP, 50)

# Add 3 tabs (the tabs are page (lv_page) and can be scrolled
tab1 = tabview.add_tab("Tab 1")
tab2 = tabview.add_tab("Tab 2")
tab3 = tabview.add_tab("Tab 3")

# Add content to the tabs
label = lv.label(tab1)
label.set_text("""This the first tab
```

(下页继续)

(续上页)

```
If the content
of a tab
becomes too
longer
than the
container
then it
automatically
becomes
scrollable.



Can you see it?""")

label = lv.label(tab2)
label.set_text("Second tab")

label = lv.label(tab3)
label.set_text("Third tab");

label.scroll_to_view_recursive(lv.ANIM.ON)
```

Tabs on the left, styling and no scrolling

```
#include "../../lv_examples.h"
#if LV_USE_TABVIEW && LV_BUILD_EXAMPLES

static void scroll_begin_event(lv_event_t * e)
{
    /*Disable the scroll animations. Triggered when a tab button is clicked */
    if(lv_event_get_code(e) == LV_EVENT_SCROLL_BEGIN) {
        lv_anim_t * a = lv_event_get_param(e);
        if(a)  a->time = 0;
    }
}

void lv_example_tabview_2(void)
{
    /*Create a Tab view object*/
```

(下页继续)

(续上页)

```
    lv_obj_t *tabview;
    tabview = lv_tabview_create(lv_scr_act(), LV_DIR_LEFT, 80);
    lv_obj_add_event_cb(lv_tabview_get_content(tabview), scroll_begin_event, LV_EVENT_
↪SCROLL_BEGIN, NULL);

    lv_obj_set_style_bg_color(tabview, lv_palette_lighten(LV_PALETTE_RED, 2), 0);

    lv_obj_t * tab_btns = lv_tabview_get_tab_btns(tabview);
    lv_obj_set_style_bg_color(tab_btns, lv_palette_darken(LV_PALETTE_GREY, 3), 0);
    lv_obj_set_style_text_color(tab_btns, lv_palette_lighten(LV_PALETTE_GREY, 5), 0);
    lv_obj_set_style_border_side(tab_btns, LV_BORDER_SIDE_RIGHT, LV_PART_ITEMS | LV_
↪STATE_CHECKED);


    /*Add 3 tabs (the tabs are page (lv_page) and can be scrolled*/
    lv_obj_t *tab1 = lv_tabview_add_tab(tabview, "Tab 1");
    lv_obj_t *tab2 = lv_tabview_add_tab(tabview, "Tab 2");
    lv_obj_t *tab3 = lv_tabview_add_tab(tabview, "Tab 3");
    lv_obj_t *tab4 = lv_tabview_add_tab(tabview, "Tab 4");
    lv_obj_t *tab5 = lv_tabview_add_tab(tabview, "Tab 5");

    lv_obj_set_style_bg_color(tab2, lv_palette_lighten(LV_PALETTE_AMBER, 3), 0);
    lv_obj_set_style_bg_opa(tab2, LV_OPA_COVER, 0);

    /*Add content to the tabs*/
    lv_obj_t * label = lv_label_create(tab1);
    lv_label_set_text(label, "First tab");

    label = lv_label_create(tab2);
    lv_label_set_text(label, "Second tab");

    label = lv_label_create(tab3);
    lv_label_set_text(label, "Third tab");

    label = lv_label_create(tab4);
    lv_label_set_text(label, "Forth tab");

    label = lv_label_create(tab5);
    lv_label_set_text(label, "Fifth tab");

    lv_obj_clear_flag(lv_tabview_get_content(tabview), LV_OBJ_FLAG_SCROLLABLE);
}
#endif
```

```
def scroll_begin_event(e):

    #Disable the scroll animations. Triggered when a tab button is clicked */
    if e.get_code() == lv.EVENT.SCROLL_BEGIN:
        a = lv.anim_t.__cast__(e.get_param())
        if a:
            a.time = 0

# Create a Tab view object
tabview = lv.tabview(lv.scr_act(), lv.DIR.LEFT, 80)
tabview.get_content().add_event_cb(scroll_begin_event, lv.EVENT.SCROLL_BEGIN, None)

tabview.set_style_bg_color(lv.palette_lighten(lv.PALETTE.RED, 2), 0)

tab_btns = tabview.get_tab_btns()
tab_btns.set_style_bg_color(lv.palette_darken(lv.PALETTE.GREY, 3), 0)
tab_btns.set_style_text_color(lv.palette_lighten(lv.PALETTE.GREY, 5), 0)
tab_btns.set_style_border_side(lv.BORDER_SIDE.RIGHT, lv.PART.ITEMS | lv.STATE.CHECKED)


# Add 3 tabs (the tabs are page (lv_page) and can be scrolled
tab1 = tabview.add_tab("Tab 1")
tab2 = tabview.add_tab("Tab 2")
tab3 = tabview.add_tab("Tab 3")
tab4 = tabview.add_tab("Tab 4")
tab5 = tabview.add_tab("Tab 5")

tab2.set_style_bg_color(lv.palette_lighten(lv.PALETTE.AMBER, 3), 0)
tab2.set_style_bg_opa(lv.OPA.COVER, 0)

# Add content to the tabs
label = lv.label(tab1)
label.set_text("First tab")

label = lv.label(tab2)
label.set_text("Second tab")

label = lv.label(tab3)
label.set_text("Third tab")

label = lv.label(tab4)
label.set_text("Forth tab")

label = lv.label(tab5)
```

(下页继续)

(续上页)

```
label.set_text("Fifth tab")

tabview.get_content().clear_flag(lv.obj.FLAG.SCROLLABLE)
```

API

Functions

lv_obj_t ***lv_tabview_create**(*lv_obj_t* *parent, lv_dir_t tab_pos, lv_coord_t tab_size)

lv_obj_t ***lv_tabview_add_tab**(*lv_obj_t* *tv, const char *name)

lv_obj_t ***lv_tabview_get_content**(*lv_obj_t* *tv)

lv_obj_t ***lv_tabview_get_tab_btns**(*lv_obj_t* *tv)

void **lv_tabview_set_act**(*lv_obj_t* *obj, uint32_t id, *lv_anim_enable_t* anim_en)

uint16_t **lv_tabview_get_tab_act**(*lv_obj_t* *tv)

Variables

const lv_obj_class_t **lv_tabview_class**

struct **lv_tabview_t**

Public Members

lv_obj_t **obj**

char ****map**

uint16_t **tab_cnt**

uint16_t **tab_cur**

lv_dir_t **tab_pos**

Tile view (lv_tileview)

Overview

The Tile view is a container object whose elements (called *tiles*) can be arranged in grid form. A user can navigate between the tiles by swiping. Any direction of swiping can be disabled on the tiles individually to not allow moving from one tile to another.

If the Tile view is screen sized, the user interface resembles what you may have seen on smartwatches.

Parts and Styles

The Tile view is built from an *lv_obj* container and *lv_obj* tiles.

The parts and styles work the same as for *lv_obj*.

Usage

Add a tile

`lv_tileview_add_tile(tileview, row_id, col_id, dir)` creates a new tile on the `row_id`th row and `col_id`th column. `dir` can be `LV_DIR_LEFT/RIGHT/TOP/BOTTOM/HOR/VER/ALL` or OR-ed values to enable moving to the adjacent tiles into the given direction by swiping.

The returned value is an `lv_obj_t *` on which the content of the tab can be created.

Change tile

The Tile view can scroll to a tile with `lv_obj_set_tile(tileview, tile_obj, LV_ANIM_ON/OFF)` or `lv_obj_set_tile_id(tileviewv, col_id, row_id, LV_ANIM_ON/OFF);`

Events

- `LV_EVENT_VALUE_CHANGED` Sent when a new tile loaded by scrolling. `lv_tileview_get_tile_act(tabview)` can be used to get current tile.

Keys

Keys are not handled by the Tile view.

Learn more about *Keys*.

Example

Tileview with content

```
#include "../../lv_examples.h"
#if LV_USE_TILEVIEW && LV_BUILD_EXAMPLES

/**
 * Create a 2x2 tile view and allow scrolling only in an "L" shape.
 * Demonstrate scroll chaining with a long list that
 * scrolls the tile view when it can't be scrolled further.
 */
void lv_example_tileview_1(void)
{
    lv_obj_t *tv = lv_tileview_create(lv_scr_act());

    /*Tile1: just a label*/
    lv_obj_t * tile1 = lv_tileview_add_tile(tv, 0, 0, LV_DIR_BOTTOM);
    lv_obj_t * label = lv_label_create(tile1);
    lv_label_set_text(label, "Scroll down");
    lv_obj_center(label);


    /*Tile2: a button*/
    lv_obj_t * tile2 = lv_tileview_add_tile(tv, 0, 1, LV_DIR_TOP | LV_DIR_RIGHT);

    lv_obj_t * btn = lv_btn_create(tile2);

    label = lv_label_create(btn);
    lv_label_set_text(label, "Scroll up or right");

    lv_obj_set_size(btn, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
    lv_obj_center(btn);

    /*Tile3: a list*/
    lv_obj_t * tile3 = lv_tileview_add_tile(tv, 1, 1, LV_DIR_LEFT);
    lv_obj_t * list = lv_list_create(tile3);
    lv_obj_set_size(list, LV_PCT(100), LV_PCT(100));
```

(下页继续)

(续上页)

```
    lv_list_add_btn(list, NULL, "One");
    lv_list_add_btn(list, NULL, "Two");
    lv_list_add_btn(list, NULL, "Three");
    lv_list_add_btn(list, NULL, "Four");
    lv_list_add_btn(list, NULL, "Five");
    lv_list_add_btn(list, NULL, "Six");
    lv_list_add_btn(list, NULL, "Seven");
    lv_list_add_btn(list, NULL, "Eight");
    lv_list_add_btn(list, NULL, "Nine");
    lv_list_add_btn(list, NULL, "Ten");

}

#endif
```

```
#
# Create a 2x2 tile view and allow scrolling only in an "L" shape.
# Demonstrate scroll chaining with a long list that
# scrolls the tile view when it can't be scrolled further.
#
tv = lv.tileview(lv.scr_act())

# Tile1: just a label
tile1 = tv.add_tile(0, 0, lv.DIR.BOTTOM)
label = lv.label(tile1)
label.set_text("Scroll down")
label.center()

# Tile2: a button
tile2 = tv.add_tile(0, 1, lv.DIR.TOP | lv.DIR.RIGHT)

btn = lv.btn(tile2)

label = lv.label(btn)
label.set_text("Scroll up or right")

btn.set_size(lv.SIZE.CONTENT, lv.SIZE.CONTENT)
btn.center()

# Tile3: a list
tile3 = tv.add_tile(1, 1, lv.DIR.LEFT)
list = lv.list(tile3)
```

(下页继续)

(续上页)

```
list.set_size(lv.pct(100), lv.pct(100))

list.add_btn(None, "One")
list.add_btn(None, "Two")
list.add_btn(None, "Three")
list.add_btn(None, "Four")
list.add_btn(None, "Five")
list.add_btn(None, "Six")
list.add_btn(None, "Seven")
list.add_btn(None, "Eight")
list.add_btn(None, "Nine")
list.add_btn(None, "Ten")
```

API

Functions

lv_obj_t ***lv_tileview_create**(*lv_obj_t* *parent)

Create a Tileview object

参数 **parent** -- pointer to an object, it will be the parent of the new tileview

返回 pointer to the created tileview

lv_obj_t ***lv_tileview_add_tile**(*lv_obj_t* *tv, uint8_t col_id, uint8_t row_id, lv_dir_t dir)

void **lv_obj_set_tile**(*lv_obj_t* *tv, *lv_obj_t* *tile_obj, *lv_anim_enable_t* anim_en)

void **lv_obj_set_tile_id**(*lv_obj_t* *tv, uint32_t col_id, uint32_t row_id, *lv_anim_enable_t* anim_en)

lv_obj_t ***lv_tileview_get_tile_act**(*lv_obj_t* *obj)

Variables

const lv_obj_class_t **lv_tileview_class**

const lv_obj_class_t **lv_tileview_tile_class**

struct **lv_tileview_t**

Public Members

lv_obj_t **obj**

lv_obj_t ***tile_act**

struct **lv_tileview_tile_t**

Public Members

lv_obj_t **obj**

lv_dir_t **dir**

Window (lv_win)

Overview

The Window is container-like object built from a header with title and buttons and a content area.

Parts and Styles

The Window is built from other widgets so you can check their documentation for details:

- Background: *lv_obj*
- Header on the background: *lv_obj*
- Title on the header: *lv_label*
- Buttons on the header: *lv_btn*
- Content area on the background: *lv_obj*

Usage

Create a Window

`lv_win_create(parent, header_height)` creates a Window with an empty header.

Title and buttons

Any number of texts (but typically only one) can be added to the header with `lv_win_add_title(win, "The title")`.

Control buttons can be added to the window's header with `lv_win_add_btn(win, icon, btn_width)`. `icon` can be any image source, and `btn_width` is the width of the button.

The title and the buttons will be added in the order the functions are called. So adding a button, a text and two other buttons will result in a button on the left, a title, and 2 buttons on the right. The width of the title is set to take all the remaining space on the header. In other words: it pushes to the right all the buttons that are added after the title.

Get the parts

`lv_win_get_header(win)` returns a pointer to the header, `lv_win_get_content(win)` returns a pointer to the content container to which the content of the window can be added.

Events

No special events are sent by the windows, however events can be added manually to the return value of `lv_win_add_btn`.

Learn more about *Events*.

Keys

No *Keys* are handled by the window.

Learn more about *Keys*.

Example

Simple window

```
#include "../../lv_examples.h"
#if LV_USE_WIN && LV_BUILD_EXAMPLES


static void event_handler(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target(e);
    LV_LOG_USER("Button %d clicked", (int)lv_obj_get_index(obj));
```

(下页继续)

(续上页)

```
}

void lv_example_win_1(void)
{
    lv_obj_t * win = lv_win_create(lv_scr_act(), 40);
    lv_obj_t * btn;
    btn = lv_win_add_btn(win, LV_SYMBOL_LEFT, 40);
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);

    lv_win_add_title(win, "A title");

    btn = lv_win_add_btn(win, LV_SYMBOL_RIGHT, 40);
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);

    btn = lv_win_add_btn(win, LV_SYMBOL_CLOSE, 60);
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);

    lv_obj_t * cont = lv_win_get_content(win);  /*Content can be added here*/
    lv_obj_t * label = lv_label_create(cont);
    lv_label_set_text(label, "This is\n"
                             "a pretty\n"
                             "long text\n"
                             "to see how\n"
                             "the window\n"
                             "becomes\n"
                             "scrollable.\n"
                             "\n"
                             "\n"
                             "Some more\n"
                             "text to be\n"
                             "sure it\n"
                             "overflows. :)");


}

#endif
```

```
def event_handler(e):
    code = e.get_code()
    obj = e.get_target()
    if code == lv.EVENT.CLICKED:
        print("Button {:d} clicked".format(obj.get_child_id()))
```

(下页继续)

(续上页)

```
win = lv.win(lv.scr_act(), 60)
btn1 = win.add_btn(lv.SYMBOL.LEFT, 40)
btn1.add_event_cb(event_handler, lv.EVENT.ALL, None)
win.add_title("A title")
btn2=win.add_btn(lv.SYMBOL.RIGHT, 40)
btn2.add_event_cb(event_handler, lv.EVENT.ALL, None)
btn3 = win.add_btn(lv.SYMBOL.CLOSE, 60)
btn3.add_event_cb(event_handler, lv.EVENT.ALL, None)

cont = win.get_content()  # Content can be added here
label = lv.label(cont)
label.set_text("""This is
a pretty
long text
to see how
the window
becomes
scrollable.


We need
quite some text
and we will
even put
some more
text to be
sure it
overflows.
""")
```

API

Functions

lv_obj_t ***lv_win_create**(*lv_obj_t* *parent, lv_coord_t header_height)

lv_obj_t ***lv_win_add_title**(*lv_obj_t* *win, const char *txt)

lv_obj_t ***lv_win_add_btn**(*lv_obj_t* *win, const void *icon, lv_coord_t btn_w)

lv_obj_t ***lv_win_get_header**(*lv_obj_t* *win)

lv_obj_t ***lv_win_get_content**(*lv_obj_t* *win)

Variables

const lv_obj_class_t **lv_win_class**

struct **lv_win_t**

Public Members

lv_obj_t **obj**

2.7 Layouts（布局）

2.7.1 Flex（弹性布局）

Overview（概述）

The Flexbox (or Flex for short) is a subset of CSS Flexbox.

It can arrange items into rows or columns (tracks), handle wrapping, adjust the spacing between the items and tracks, handle *grow* to make the item(s) fill the remaining space with respect to min/max width and height.

To make an object flex container call `lv_obj_set_layout(obj, LV_LAYOUT_FLEX)`.

Note that the flex layout feature of LVGL needs to be globally enabled with `LV_USE_FLEX` in `lv_conf.h`.

Flexbox（或简称 Flex）是 CSS Flexbox 的一个子集。

它可以将项目排列成行或列（轨道），处理环绕，调整项目和轨道之间的间距，处理 *grow* 以使项目填充剩余空间的最小/最大宽度和高度。

要使对象 flex 容器调用 `lv_obj_set_layout(obj, LV_LAYOUT_FLEX)`。

请注意，LVGL 的 flex 布局功能需要通过 `lv_conf.h` 中的 `LV_USE_FLEX` 全局启用。

Terms（约定）

- tracks: the rows or columns
- main direction: row or column, the direction in which the items are placed
- cross direction: perpendicular to the main direction
- wrap: if there there is no more space in the track a new track is started
- grow: if set on an item it will grow to fill the remaining space on the track. The available space will be distributed among items respective to the their grow value (larger value means more space)
- gap: the space between the rows and columns or the items on a track
- (tracks) 轨道：行或列
- (main direction) 主要方向：行或列，物品放置的方向
- (cross direction) 横向：垂直于主方向
- (wrap) 环绕：如果曲目中没有更多空间，则开始新曲目
- (grow) 增长：如果设置在一个项目上，它将增长以填充轨道上的剩余空间。可用空间将根据其增长值分配给各个项目（值越大意味着空间越大）
- (gap) 间隙：行和列或轨道上的项目之间的空间

Simple interface（简单接口）

With the following functions you can set a Flex layout on any parent.

使用以下功能，您可以在任何父级上设置 Flex 布局。

Flex flow

`lv_obj_set_flex_flow(obj, flex_flow)`

The possible values for `flex_flow` are:

- `LV_FLEX_FLOW_ROW` Place the children in a row without wrapping
- `LV_FLEX_FLOW_COLUMN` Place the children in a column without wrapping
- `LV_FLEX_FLOW_ROW_WRAP` Place the children in a row with wrapping
- `LV_FLEX_FLOW_COLUMN_WRAP` Place the children in a column with wrapping
- `LV_FLEX_FLOW_ROW_REVERSE` Place the children in a row without wrapping but in reversed order
- `LV_FLEX_FLOW_COLUMN_REVERSE` Place the children in a column without wrapping but in reversed order
- `LV_FLEX_FLOW_ROW_WRAP_REVERSE` Place the children in a row without wrapping but in reversed order

- LV_FLEX_FLOW_COLUMN_WRAP_REVERSE Place the children in a column without wrapping but in reversed order

lv_obj_set_flex_flow(obj, flex_flow)

flex_flow 的可能值是：

- LV_FLEX_FLOW_ROW 将子元素排成一排而不包裹
- LV_FLEX_FLOW_COLUMN 将子项放在一列中而不换行
- LV_FLEX_FLOW_ROW_WRAP 将孩子排成一排并包裹起来
- LV_FLEX_FLOW_COLUMN_WRAP 将子元素放置在带有环绕的列中
- LV_FLEX_FLOW_ROW_REVERSE 将子元素排成一行而不换行，但顺序相反
- LV_FLEX_FLOW_COLUMN_REVERSE 将子项放在一列中，不换行，但顺序相反
- LV_FLEX_FLOW_ROW_WRAP_REVERSE 将子元素排成一行而不换行，但顺序相反
- LV_FLEX_FLOW_COLUMN_WRAP_REVERSE 将子项放在一列中，不换行，但顺序相反

Flex align

To manage the placement of the children use lv_obj_set_flex_align(obj, main_place, cross_place, track_cross_place)

- main_place determines how to distribute the items in their track on the main axis. E.g. flush the items to the right on LV_FLEX_FLOW_ROW_WRAP. (It's called justify-content in CSS)
- cross_place determines how to distribute the items in their track on the cross axis. E.g. if the items have different height place them to the bottom of the track. (It's called align-items in CSS)
- track_cross_place determines how to distribute the tracks (It's called align-content in CSS)

要管理孩子的位置，请使用 lv_obj_set_flex_align(obj, main_place, cross_place, track_cross_place)

- main_place 确定如何在主轴上的轨道中分布项目。例如。将 "LV_FLEX_FLOW_ROW_WRAP" 上的项目向右刷新。（它在 CSS 中称为 justify-content）
- cross_place 确定如何在横轴上的轨道中分布项目。例如。如果项目具有不同的高度，则将它们放置在轨道的底部。（在 CSS 中称为 align-items）
- track_cross_place 确定如何分配轨道（在 CSS 中称为 align-content）

The possible values are:

- LV_FLEX_ALIGN_START means left on a horizontally and top vertically. (default)
- LV_FLEX_ALIGN_END means right on a horizontally and bottom vertically
- LV_FLEX_ALIGN_CENTER simply center

- LV_FLEX_ALIGN_SPACE_EVENLY items are distributed so that the spacing between any two items (and the space to the edges) is equal. Does not apply to track_cross_place.
- LV_FLEX_ALIGN_SPACE_AROUND items are evenly distributed in the track with equal space around them. Note that visually the spaces aren't equal, since all the items have equal space on both sides. The first item will have one unit of space against the container edge, but two units of space between the next item because that next item has its own spacing that applies. Not applies to track_cross_place.
- LV_FLEX_ALIGN_SPACE_BETWEEN items are evenly distributed in the track: first item is on the start line, last item on the end line. Not applies to track_cross_place.

可能的值为：

- LV_FLEX_ALIGN_START 表示在水平方向和垂直方向的顶部左侧。（默认）
- LV_FLEX_ALIGN_END 表示水平和底部垂直
- LV_FLEX_ALIGN_CENTER 只是居中
- LV_FLEX_ALIGN_SPACE_EVENLY 项目是分布的，因此任意两个项目之间的间距（以及到边缘的空间）是相等的。不适用于 track_cross_place。
- LV_FLEX_ALIGN_SPACE_AROUND 项目均匀分布在轨道中，它们周围的空间相等。请注意，视觉上的空间并不相等，因为所有项目的两侧都有相等的空间。第一个项目将与容器边缘有一个单位的空间，但下一个项目之间有两个单位的空间，因为下一个项目有自己的适用间距。不适用于 track_cross_place。
- LV_FLEX_ALIGN_SPACE_BETWEEN 项目均匀分布在轨道中：第一个项目在开始线上，最后一个项目在结束线上。不适用于 track_cross_place。

Flex grow

Flex grow can be used to make one or more children fill the available space on the track. If more children has grow the available space will be distributed proportionally to the grow values. For example let's there is 400 px remaining space and 4 object with grow:

- A with grow = 1
- B with grow = 1
- C with grow = 2

A and B will have 100 px size, and C will have 200 px size.

Flex grow can be set on a child with lv_obj_set_flex_grow(child, value). value needs to be > 1 or 0 to disable grow on the child.

Flex Growth 可用于让一个或多个孩子填充轨道上的可用空间。如果有更多的孩子成长，可用空间将与成长值成比例地分配。例如，让我们有 400 像素的剩余空间和 4 个增长的对象：

- A 增长 = 1

- B 增长 = 1
- C 增长 = 2

A 和 B 的大小为 100 px，而 C 的大小为 200 px。

可以使用 `lv_obj_set_flex_grow(child, value)` 在子节点上设置 Flex 增长。`value` 需要 > 1 或 0 禁用在孩子身上生长。

Style interface（样式接口）

All the Flex-related values are style properties under the hood and you can use them similarly to any other style property. The following flex related style properties exist:

- `FLEX_FLOW`
- `FLEX_MAIN_PLACE`
- `FLEX_CROSS_PLACE`
- `FLEX_TRACK_PLACE`
- `FLEX_GROW`

所有与 Flex 相关的值都是底层的样式属性，您可以像使用任何其他样式属性一样使用它们。存在以下与 flex 相关的样式属性：

- `FLEX_FLOW`
- `FLEX_MAIN_PLACE`
- `FLEX_CROSS_PLACE`
- `FLEX_TRACK_PLACE`
- `FLEX_GROW`

Other features（其它功能）

RTL

If the base direction of the container is set the `LV_BASE_DIR_RTL` the meaning of `LV_FLEX_ALIGN_START` and `LV_FLEX_ALIGN_END` is swapped on `ROW` layouts. I.e. `START` will mean right.

The items on `ROW` layouts, and tracks of `COLUMN` layouts will be placed from right to left.

如果容器的基本方向设置为 `LV_BASE_DIR_RTL`，`LV_FLEX_ALIGN_START` 和 `LV_FLEX_ALIGN_END` 的含义在 `ROW` 布局上交换。IE。`START` 表示正确。

`ROW` 布局上的项目和 `COLUMN` 布局的轨道将从右到左放置。

New track（新轨道）

You can force Flex to put an item into a new line with `lv_obj_add_flag(child, LV_OBJ_FLAG_FLEX_IN_NEW_TRACK)`.

您可以使用 `lv_obj_add_flag(child, LV_OBJ_FLAG_FLEX_IN_NEW_TRACK)` 强制 Flex 将项目放入新行。

Example

A simple row and a column layout with flexbox

```
#include "../../lv_examples.h"
#if LV_USE_FLEX && LV_BUILD_EXAMPLES

/**
 * A simple row and a column layout with flexbox
 */
void lv_example_flex_1(void)
{
    /*Create a container with ROW flex direction*/
    lv_obj_t * cont_row = lv_obj_create(lv_scr_act());
    lv_obj_set_size(cont_row, 300, 75);
    lv_obj_align(cont_row, LV_ALIGN_TOP_MID, 0, 5);
    lv_obj_set_flex_flow(cont_row, LV_FLEX_FLOW_ROW);

    /*Create a container with COLUMN flex direction*/
    lv_obj_t * cont_col = lv_obj_create(lv_scr_act());
    lv_obj_set_size(cont_col, 200, 150);
    lv_obj_align_to(cont_col, cont_row, LV_ALIGN_OUT_BOTTOM_MID, 0, 5);
    lv_obj_set_flex_flow(cont_col, LV_FLEX_FLOW_COLUMN);

    uint32_t i;
    for(i = 0; i < 10; i++) {
        lv_obj_t * obj;
        lv_obj_t * label;

        /*Add items to the row*/
        obj= lv_btn_create(cont_row);
        lv_obj_set_size(obj, 100, LV_PCT(100));

        label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "Item: %u", i);
        lv_obj_center(label);
```

(下页继续)

(续上页)

```
        /*Add items to the column*/
        obj = lv_btn_create(cont_col);
        lv_obj_set_size(obj, LV_PCT(100), LV_SIZE_CONTENT);

        label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "Item: %"LV_PRIu32, i);
        lv_obj_center(label);
    }
}

#endif
```

```
#
# A simple row and a column layout with flexbox
#

# Create a container with ROW flex direction
cont_row = lv.obj(lv.scr_act())
cont_row.set_size(300, 75)
cont_row.align(lv.ALIGN.TOP_MID, 0, 5)
cont_row.set_flex_flow(lv.FLEX_FLOW.ROW)

# Create a container with COLUMN flex direction
cont_col = lv.obj(lv.scr_act())
cont_col.set_size(200, 150)
cont_col.align_to(cont_row, lv.ALIGN.OUT_BOTTOM_MID, 0, 5)
cont_col.set_flex_flow(lv.FLEX_FLOW.COLUMN)

for i in range(10):
    # Add items to the row
    obj = lv.btn(cont_row)
    obj.set_size(100, lv.pct(100))

    label = lv.label(obj)
    label.set_text("Item: {:d}".format(i))
    label.center()

    # Add items to the column
    obj = lv.btn(cont_col)
    obj.set_size(lv.pct(100), lv.SIZE.CONTENT)

    label = lv.label(obj)
```

(下页继续)

(续上页)

```
    label.set_text("Item: {:d}".format(i))
    label.center()
```

Arrange items in rows with wrap and even spacing

```
#include "../../lv_examples.h"
#if LV_USE_FLEX && LV_BUILD_EXAMPLES

/**
 * Arrange items in rows with wrap and place the items to get even space around them.
 */
void lv_example_flex_2(void)
{
    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_flex_flow(&style, LV_FLEX_FLOW_ROW_WRAP);
    lv_style_set_flex_main_place(&style, LV_FLEX_ALIGN_SPACE_EVENLY);
    lv_style_set_layout(&style, LV_LAYOUT_FLEX);

    lv_obj_t * cont = lv_obj_create(lv_scr_act());
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_add_style(cont, &style, 0);

    uint32_t i;
    for(i = 0; i < 8; i++) {
        lv_obj_t * obj = lv_obj_create(cont);
        lv_obj_set_size(obj, 70, LV_SIZE_CONTENT);
        lv_obj_add_flag(obj, LV_OBJ_FLAG_CHECKABLE);

        lv_obj_t * label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "%"LV_PRIu32, i);
        lv_obj_center(label);
    }
}

#endif
```

```
#
# Arrange items in rows with wrap and place the items to get even space around them.
#
```

(下页继续)

(续上页)

```
style = lv.style_t()
style.init()
style.set_flex_flow(lv.FLEX_FLOW.ROW_WRAP)
style.set_flex_main_place(lv.FLEX_ALIGN.SPACE_EVENLY)
style.set_layout(lv.LAYOUT_FLEX.value)

cont = lv.obj(lv.scr_act())
cont.set_size(300, 220)
cont.center()
cont.add_style(style, 0)

for i in range(8):
    obj = lv.obj(cont)
    obj.set_size(70, lv.SIZE.CONTENT)

    label = lv.label(obj)
    label.set_text("{:d}".format(i))
    label.center()
```

Demonstrate flex grow

```
#include "../../lv_examples.h"
#if LV_USE_FLEX && LV_BUILD_EXAMPLES

/**
 * Demonstrate flex grow.
 */
void lv_example_flex_3(void)
{
    lv_obj_t * cont = lv_obj_create(lv_scr_act());
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_ROW);

    lv_obj_t * obj;
    obj = lv_obj_create(cont);
    lv_obj_set_size(obj, 40, 40);           /*Fix size*/

    obj = lv_obj_create(cont);
    lv_obj_set_height(obj, 40);
    lv_obj_set_flex_grow(obj, 1);           /*1 portion from the free space*/
```

(下页继续)

(续上页)

```
    obj = lv_obj_create(cont);
    lv_obj_set_height(obj, 40);
    lv_obj_set_flex_grow(obj, 2);           /*2 portion from the free space*/

    obj = lv_obj_create(cont);
    lv_obj_set_size(obj, 40, 40);           /*Fix size. It is flushed to the right by␣
↪the "grow" items*/
}

#endif
```

```
#
# Demonstrate flex grow.
#

cont = lv.obj(lv.scr_act())
cont.set_size(300, 220)
cont.center()
cont.set_flex_flow(lv.FLEX_FLOW.ROW)

obj = lv.obj(cont)
obj.set_size(40, 40)            # Fix size

obj = lv.obj(cont)
obj.set_height(40)
obj.set_flex_grow(1)            # 1 portion from the free space

obj = lv.obj(cont)
obj.set_height(40)
obj.set_flex_grow(2)            # 2 portion from the free space

obj = lv.obj(cont)
obj.set_size(40, 40)            # Fix size. It is flushed to the right by the "grow"␣
↪items
```

Demonstrate flex grow.

```c
#include "../../lv_examples.h"
#if LV_USE_FLEX && LV_BUILD_EXAMPLES

/**
 * Reverse the order of flex items
 */
void lv_example_flex_4(void)
{

    lv_obj_t * cont = lv_obj_create(lv_scr_act());
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_COLUMN_REVERSE);

    uint32_t i;
    for(i = 0; i < 6; i++) {
        lv_obj_t * obj = lv_obj_create(cont);
        lv_obj_set_size(obj, 100, 50);

        lv_obj_t * label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "Item: %"LV_PRIu32, i);
        lv_obj_center(label);
    }
}

#endif
```

```python
#
# Reverse the order of flex items
#
cont = lv.obj(lv.scr_act())
cont.set_size(300, 220)
cont.center()
cont.set_flex_flow(lv.FLEX_FLOW.COLUMN_REVERSE)

for i  in range(6):
    obj = lv.obj(cont)
    obj.set_size(100, 50)

    label = lv.label(obj)
    label.set_text("Item: " + str(i))
    label.center()
```

(下页继续)

(续上页)

```
```

Demonstrate column and row gap style properties

```
#include "../../lv_examples.h"
#if LV_USE_FLEX && LV_BUILD_EXAMPLES

static void row_gap_anim(void * obj, int32_t v)
{
    lv_obj_set_style_pad_row(obj, v, 0);
}

static void column_gap_anim(void * obj, int32_t v)
{
    lv_obj_set_style_pad_column(obj, v, 0);
}

/**
 * Demonstrate the effect of column and row gap style properties
 */
void lv_example_flex_5(void)
{
    lv_obj_t * cont = lv_obj_create(lv_scr_act());
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_ROW_WRAP);

    uint32_t i;
    for(i = 0; i < 9; i++) {
        lv_obj_t * obj = lv_obj_create(cont);
        lv_obj_set_size(obj, 70, LV_SIZE_CONTENT);

        lv_obj_t * label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "%"LV_PRIu32, i);
        lv_obj_center(label);
    }

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_var(&a, cont);
    lv_anim_set_values(&a, 0, 10);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
```

(下页继续)

(续上页)

```
    lv_anim_set_exec_cb(&a, row_gap_anim);
    lv_anim_set_time(&a, 500);
    lv_anim_set_playback_time(&a, 500);
    lv_anim_start(&a);

    lv_anim_set_exec_cb(&a, column_gap_anim);
    lv_anim_set_time(&a, 3000);
    lv_anim_set_playback_time(&a, 3000);
    lv_anim_start(&a);
}

#endif
```

```
def row_gap_anim(obj, v):
    obj.set_style_pad_row(v, 0)


def column_gap_anim(obj, v):
    obj.set_style_pad_column(v, 0)

#
# Demonstrate the effect of column and row gap style properties
#

cont = lv.obj(lv.scr_act())
cont.set_size(300, 220)
cont.center()
cont.set_flex_flow(lv.FLEX_FLOW.ROW_WRAP)

for i in range(9):
    obj = lv.obj(cont)
    obj.set_size(70, lv.SIZE.CONTENT)

    label = lv.label(obj)
    label.set_text(str(i))
    label.center()

a_row = lv.anim_t()
a_row.init()
a_row.set_var(cont)
a_row.set_values(0, 10)
a_row.set_repeat_count(lv.ANIM_REPEAT.INFINITE)
```

(下页继续)

(续上页)

```
a_row.set_time(500)
a_row.set_playback_time(500)
a_row.set_custom_exec_cb(lambda a,val: row_gap_anim(cont,val))
lv.anim_t.start(a_row)

a_col = lv.anim_t()
a_col.init()
a_col.set_var(cont)
a_col.set_values(0, 10)
a_col.set_repeat_count(lv.ANIM_REPEAT.INFINITE)

a_col.set_time(3000)
a_col.set_playback_time(3000)
a_col.set_custom_exec_cb(lambda a,val: column_gap_anim(cont,val))

lv.anim_t.start(a_col)
```

RTL base direction changes order of the items

```
#include "../../lv_examples.h"
#if LV_USE_FLEX && LV_BUILD_EXAMPLES

/**
 * RTL base direction changes order of the items.
 * Also demonstrate how horizontal scrolling works with RTL.
 */
void lv_example_flex_6(void)
{
    lv_obj_t * cont = lv_obj_create(lv_scr_act());
    lv_obj_set_style_base_dir(cont, LV_BASE_DIR_RTL, 0);
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_ROW_WRAP);

    uint32_t i;
    for(i = 0; i < 20; i++) {
        lv_obj_t * obj = lv_obj_create(cont);
        lv_obj_set_size(obj, 70, LV_SIZE_CONTENT);

        lv_obj_t * label = lv_label_create(obj);
```

(下页继续)

(续上页)

```
        lv_label_set_text_fmt(label, "%"LV_PRIu32, i);
        lv_obj_center(label);
    }
}
#endif
```

```
#
# RTL base direction changes order of the items.
# Also demonstrate how horizontal scrolling works with RTL.
#

cont = lv.obj(lv.scr_act())
cont.set_style_base_dir(lv.BASE_DIR.RTL,0)
cont.set_size(300, 220)
cont.center()
cont.set_flex_flow(lv.FLEX_FLOW.ROW_WRAP)

for i in range(20):
    obj = lv.obj(cont)
    obj.set_size(70, lv.SIZE.CONTENT)

    label = lv.label(obj)
    label.set_text(str(i))
    label.center()
```

API

Enums

enum **lv_flex_align_t**

Values:

enumerator **LV_FLEX_ALIGN_START**

enumerator **LV_FLEX_ALIGN_END**

enumerator **LV_FLEX_ALIGN_CENTER**

enumerator **LV_FLEX_ALIGN_SPACE_EVENLY**

enumerator **LV_FLEX_ALIGN_SPACE_AROUND**

enumerator **LV_FLEX_ALIGN_SPACE_BETWEEN**

enum **lv_flex_flow_t**

Values:

enumerator **LV_FLEX_FLOW_ROW**

enumerator **LV_FLEX_FLOW_COLUMN**

enumerator **LV_FLEX_FLOW_ROW_WRAP**

enumerator **LV_FLEX_FLOW_ROW_REVERSE**

enumerator **LV_FLEX_FLOW_ROW_WRAP_REVERSE**

enumerator **LV_FLEX_FLOW_COLUMN_WRAP**

enumerator **LV_FLEX_FLOW_COLUMN_REVERSE**

enumerator **LV_FLEX_FLOW_COLUMN_WRAP_REVERSE**

Functions

LV_EXPORT_CONST_INT(LV_OBJ_FLAG_FLEX_IN_NEW_TRACK)

void **lv_flex_init**(void)

Initialize a flex layout the default values

参数 **flex** -- pointer to a flex layout descriptor

void **lv_obj_set_flex_flow**(*lv_obj_t* *obj, *lv_flex_flow_t* flow)

Set hot the item should flow

参数

- **flex** -- pointer to a flex layout descriptor
- **flow** -- an element of `lv_flex_flow_t`.

void **lv_obj_set_flex_align**(*lv_obj_t* *obj, *lv_flex_align_t* main_place, *lv_flex_align_t* cross_place, *lv_flex_align_t* track_cross_place)

Set how to place (where to align) the items and tracks

参数

- **flex** -- pointer: to a flex layout descriptor
- **main_place** -- where to place the items on main axis (in their track). Any value of `lv_flex_align_t`.
- **cross_place** -- where to place the item in their track on the cross axis. `LV_FLEX_ALIGN_START/END/CENTER`

- **track_place** -- where to place the tracks in the cross direction. Any value of `lv_flex_align_t`.

void **lv_obj_set_flex_grow**(*lv_obj_t* *obj, uint8_t grow)

Sets the width or height (on main axis) to grow the object in order fill the free space

参数

- **obj** -- pointer to an object. The parent must have flex layout else nothing will happen.
- **grow** -- a value to set how much free space to take proportionally to other growing items.

void **lv_style_set_flex_flow**(*lv_style_t* *style, *lv_flex_flow_t* value)

void **lv_style_set_flex_main_place**(*lv_style_t* *style, *lv_flex_align_t* value)

void **lv_style_set_flex_cross_place**(*lv_style_t* *style, *lv_flex_align_t* value)

void **lv_style_set_flex_track_place**(*lv_style_t* *style, *lv_flex_align_t* value)

void **lv_style_set_flex_grow**(*lv_style_t* *style, uint8_t value)

void **lv_obj_set_style_flex_flow**(*lv_obj_t* *obj, *lv_flex_flow_t* value, lv_style_selector_t selector)

void **lv_obj_set_style_flex_main_place**(*lv_obj_t* *obj, *lv_flex_align_t* value, lv_style_selector_t selector)

void **lv_obj_set_style_flex_cross_place**(*lv_obj_t* *obj, *lv_flex_align_t* value, lv_style_selector_t selector)

void **lv_obj_set_style_flex_track_place**(*lv_obj_t* *obj, *lv_flex_align_t* value, lv_style_selector_t selector)

void **lv_obj_set_style_flex_grow**(*lv_obj_t* *obj, uint8_t value, lv_style_selector_t selector)

static inline *lv_flex_flow_t* **lv_obj_get_style_flex_flow**(const *lv_obj_t* *obj, uint32_t part)

static inline *lv_flex_align_t* **lv_obj_get_style_flex_main_place**(const *lv_obj_t* *obj, uint32_t part)

static inline *lv_flex_align_t* **lv_obj_get_style_flex_cross_place**(const *lv_obj_t* *obj, uint32_t part)

static inline *lv_flex_align_t* **lv_obj_get_style_flex_track_place**(const *lv_obj_t* *obj, uint32_t part)

static inline uint8_t **lv_obj_get_style_flex_grow**(const *lv_obj_t* *obj, uint32_t part)

Variables

uint32_t **LV_LAYOUT_FLEX**

lv_style_prop_t **LV_STYLE_FLEX_FLOW**

lv_style_prop_t **LV_STYLE_FLEX_MAIN_PLACE**

lv_style_prop_t **LV_STYLE_FLEX_CROSS_PLACE**

lv_style_prop_t **LV_STYLE_FLEX_TRACK_PLACE**

lv_style_prop_t **LV_STYLE_FLEX_GROW**

2.7.2 Grid（网格布局）

Overview（概述）

The Grid layout is a subset of CSS Flexbox.

It can arrange items into 2D ”table” that has rows or columns (tracks). The item can span through multiple columns or rows. The track’s size can be set in pixel, to the largest item (`LV_GRID_CONTENT`) or in ”Free unit” (FR) to distribute the free space proportionally.

To make an object a grid container call `lv_obj_set_layout(obj, LV_LAYOUT_GRID)`.

Note that the grid layout feature of LVGL needs to be globally enabled with `LV_USE_GRID` in `lv_conf.h`.

网格布局是 CSS Flexbox 的一个子集。

它可以将项目排列成具有行或列（轨道）的二维“表格”。该项目可以跨越多个列或行。轨道的大小可以设置为像素、最大项目（`LV_GRID_CONTENT`）或“空闲单元”（FR）以按比例分配空闲空间。

要使对象成为网格容器，请调用 `lv_obj_set_layout(obj, LV_LAYOUT_GRID)`。

请注意，LVGL 的网格布局功能需要通过 `lv_conf.h` 中的 `LV_USE_GRID` 全局启用。

Terms（约定）

- tracks: the rows or columns
- free unit (FR): if set on track's size is set in FR it will grow to fill the remaining space on the parent.
- gap: the space between the rows and columns or the items on a track
- 轨道：行或列
- 空闲单元（FR）：如果在 FR 中设置了轨道的大小，它将增长以填充父级上的剩余空间。
- 间隙：行和列或轨道上的项目之间的空间

Simple interface（简单的接口）

With the following functions you can easily set a Grid layout on any parent.

使用以下功能，您可以轻松地在任何父级上设置网格布局。

Grid descriptors

First you need to describe the size of rows and columns. It can be done by declaring 2 arrays and the track sizes in them. The last element must be `LV_GRID_TEMPLATE_LAST`.

For example:

首先，您需要描述行和列的大小。可以通过声明 2 个数组和其中的轨道大小来完成。最后一个元素必须是 `LV_GRID_TEMPLATE_LAST`。

例如：

```
static lv_coord_t column_dsc[] = {100, 400, LV_GRID_TEMPLATE_LAST};   /*2 columns␣
↪with 100 and 400 ps width*/
static lv_coord_t row_dsc[] = {100, 100, 100, LV_GRID_TEMPLATE_LAST}; /*3 100 px tall␣
↪rows*/
```

To set the descriptors on a parent use `lv_obj_set_grid_dsc_array(obj, col_dsc, row_dsc)`.

Besides simple settings the size in pixel you can use two special values:

- `LV_GRID_CONTENT` set the width to the largest children on this track
- `LV_GRID_FR(X)` tell what portion of the remaining space should be used by this track. Larger value means larger space.

要在父级上设置描述符，请使用 `lv_obj_set_grid_dsc_array(obj, col_dsc, row_dsc)`。

除了简单的设置像素大小之外，您还可以使用两个特殊值：

- `LV_GRID_CONTENT` 将宽度设置为这条轨道上最大的孩子

- LV_GRID_FR(X) 告诉这个轨道应该使用剩余空间的哪一部分。更大的值意味着更大的空间。

Grid items（网格项）

By default the children are not added to the grid. They need to be added manually to a cell.

To do this call lv_obj_set_grid_cell(child, column_align, column_pos, column_span, row_align, row_pos, row_span).

column_align and row_align determine how to align the children in its cell. The possible values are:

- LV_GRID_ALIGN_START means left on a horizontally and top vertically. (default)
- LV_GRID_ALIGN_END means right on a horizontally and bottom vertically
- LV_GRID_ALIGN_CENTER simply center

colum_pos and row_pos means the zero based index of the cell into the item should be placed.

colum_span and row_span means how many tracks should the item involve from the start cell. Must be > 1.

默认情况下，子项不会添加到网格中。它们需要手动添加到单元格中。

为此调用 lv_obj_set_grid_cell(child, column_align, column_pos, column_span, row_align, row_pos, row_span)。

column_align 和 row_align 确定如何在其单元格中对齐子项。可能的值为：

- LV_GRID_ALIGN_START 表示在水平方向和垂直方向的顶部左侧。（默认）
- LV_GRID_ALIGN_END 表示水平和底部垂直
- LV_GRID_ALIGN_CENTER 只是居中

colum_pos 和 row_pos 表示应该放置项目中单元格的从零开始的索引。

colum_span 和 row_span 表示该项目应该从起始单元格开始包含多少个轨道。必须 > 1.

Grid align（网格对齐）

If there are some empty space the track can be aligned several ways:

- LV_GRID_ALIGN_START means left on a horizontally and top vertically. (default)
- LV_GRID_ALIGN_END means right on a horizontally and bottom vertically
- LV_GRID_ALIGN_CENTER simply center
- LV_GRID_ALIGN_SPACE_EVENLY items are distributed so that the spacing between any two items (and the space to the edges) is equal. Not applies to track_cross_place.
- LV_GRID_ALIGN_SPACE_AROUND items are evenly distributed in the track with equal space around them. Note that visually the spaces aren’t equal, since all the items have equal space on both sides. The first item will

have one unit of space against the container edge, but two units of space between the next item because that next item has its own spacing that applies. Not applies to `track_cross_place`.

- `LV_GRID_ALIGN_SPACE_BETWEEN` items are evenly distributed in the track: first item is on the start line, last item on the end line. Not applies to `track_cross_place`.

To set the track's alignment use `lv_obj_set_grid_align(obj, column_align, row_align)`.

如果有一些空白空间，轨道可以通过几种方式对齐：

- `LV_GRID_ALIGN_START` 表示在水平方向和垂直方向的顶部左侧。（默认）
- `LV_GRID_ALIGN_END` 表示水平和底部垂直
- `LV_GRID_ALIGN_CENTER` 只是居中
- `LV_GRID_ALIGN_SPACE_EVENLY` 项目是分布的，因此任意两个项目之间的间距（以及到边缘的空间）是相等的。不适用于 `track_cross_place`。
- `LV_GRID_ALIGN_SPACE_AROUND` 项目均匀分布在轨道中，它们周围的空间相等。请注意，视觉上的空间并不相等，因为所有项目的两侧都有相等的空间。第一个项目将与容器边缘有一个单位的空间，但下一个项目之间有两个单位的空间，因为下一个项目有自己的适用间距。不适用于 `track_cross_place`。
- `LV_GRID_ALIGN_SPACE_BETWEEN` 项目均匀分布在轨道中：第一个项目在开始线上，最后一个项目在结束线上。不适用于 `track_cross_place`。

要设置轨道的对齐方式，请使用 `lv_obj_set_grid_align(obj, column_align, row_align)`。

Style interface（样式接口）

All the Grid related values are style properties under the hood and you can use them similarly to any other style properties. The following Grid related style properties exist:

- `GRID_COLUMN_DSC_ARRAY`
- `GRID_ROW_DSC_ARRAY`
- `GRID_COLUMN_ALIGN`
- `GRID_ROW_ALIGN`
- `GRID_CELL_X_ALIGN`
- `GRID_CELL_COLUMN_POS`
- `GRID_CELL_COLUMN_SPAN`
- `GRID_CELL_Y_ALIGN`
- `GRID_CELL_ROW_POS`
- `GRID_CELL_ROW_SPAN`

所有与 Grid 相关的值都是底层的样式属性，您可以像使用任何其他样式属性一样使用它们。存在以下与网格相关的样式属性：

- GRID_COLUMN_DSC_ARRAY
- GRID_ROW_DSC_ARRAY
- GRID_COLUMN_ALIGN
- GRID_ROW_ALIGN
- GRID_CELL_X_ALIGN
- GRID_CELL_COLUMN_POS
- GRID_CELL_COLUMN_SPAN
- GRID_CELL_Y_ALIGN
- GRID_CELL_ROW_POS
- GRID_CELL_ROW_SPAN

Other features （其它功能）

RTL

If the base direction of the container is set to LV_BASE_DIR_RTL, the meaning of LV_GRID_ALIGN_START and LV_GRID_ALIGN_END is swapped. I.e. START will mean right-most.

The columns will be placed from right to left.

如果容器的基本方向设置为 LV_BASE_DIR_RTL，则 LV_GRID_ALIGN_START 和 LV_GRID_ALIGN_END 的含义互换。IE。START 表示最右边。

列将从右到左放置。

Example

A simple grid

```
#include "../../lv_examples.h"
#if LV_USE_GRID && LV_BUILD_EXAMPLES

/**
 * A simple grid
 */
void lv_example_grid_1(void)
{
```

(下页继续)

(续上页)

```
    static lv_coord_t col_dsc[] = {70, 70, 70, LV_GRID_TEMPLATE_LAST};
    static lv_coord_t row_dsc[] = {50, 50, 50, LV_GRID_TEMPLATE_LAST};

    /*Create a container with grid*/
    lv_obj_t * cont = lv_obj_create(lv_scr_act());
    lv_obj_set_style_grid_column_dsc_array(cont, col_dsc, 0);
    lv_obj_set_style_grid_row_dsc_array(cont, row_dsc, 0);
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set_layout(cont, LV_LAYOUT_GRID);

    lv_obj_t * label;
    lv_obj_t * obj;

    uint32_t i;
    for(i = 0; i < 9; i++) {
        uint8_t col = i % 3;
        uint8_t row = i / 3;

        obj = lv_btn_create(cont);
        /*Stretch the cell horizontally and vertically too
         *Set span to 1 to make the cell 1 column/row sized*/
        lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, col, 1,
                             LV_GRID_ALIGN_STRETCH, row, 1);

        label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "c%d, r%d", col, row);
        lv_obj_center(label);
    }
}

#endif
```

```
#
# A simple grid
#

col_dsc = [70, 70, 70, lv.GRID_TEMPLATE.LAST]
row_dsc = [50, 50, 50, lv.GRID_TEMPLATE.LAST]

# Create a container with grid
cont = lv.obj(lv.scr_act())
cont.set_style_grid_column_dsc_array(col_dsc, 0)
```

(下页继续)

(续上页)

```
cont.set_style_grid_row_dsc_array(row_dsc, 0)
cont.set_size(300, 220)
cont.center()
cont.set_layout(lv.LAYOUT_GRID.value)

for i in range(9):
    col = i % 3
    row = i // 3

    obj = lv.btn(cont)
    # Stretch the cell horizontally and vertically too
    # Set span to 1 to make the cell 1 column/row sized
    obj.set_grid_cell(lv.GRID_ALIGN.STRETCH, col, 1,
                      lv.GRID_ALIGN.STRETCH, row, 1)

    label = lv.label(obj)
    label.set_text("c" +str(col) +  "r" +str(row))
    label.center()
```

Demonstrate cell placement and span

```
#include "../../lv_examples.h"
#if LV_USE_GRID && LV_BUILD_EXAMPLES


/**
 * Demonstrate cell placement and span
 */
void lv_example_grid_2(void)
{
    static lv_coord_t col_dsc[] = {70, 70, 70, LV_GRID_TEMPLATE_LAST};
    static lv_coord_t row_dsc[] = {50, 50, 50, LV_GRID_TEMPLATE_LAST};

    /*Create a container with grid*/
    lv_obj_t * cont = lv_obj_create(lv_scr_act());
    lv_obj_set_grid_dsc_array(cont, col_dsc, row_dsc);
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);

    lv_obj_t * label;
    lv_obj_t * obj;
```

(下页继续)

(续上页)

```
    /*Cell to 0;0 and align to to the start (left/top) horizontally and vertically␣
↪too*/
    obj = lv_obj_create(cont);
    lv_obj_set_size(obj, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
    lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_START, 0, 1,
                             LV_GRID_ALIGN_START, 0, 1);
    label = lv_label_create(obj);
    lv_label_set_text(label, "c0, r0");

    /*Cell to 1;0 and align to to the start (left) horizontally and center vertically␣
↪too*/
    obj = lv_obj_create(cont);
    lv_obj_set_size(obj, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
    lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_START, 1, 1,
                             LV_GRID_ALIGN_CENTER, 0, 1);
    label = lv_label_create(obj);
    lv_label_set_text(label, "c1, r0");

    /*Cell to 2;0 and align to to the start (left) horizontally and end (bottom)␣
↪vertically too*/
    obj = lv_obj_create(cont);
    lv_obj_set_size(obj, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
    lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_START, 2, 1,
                             LV_GRID_ALIGN_END, 0, 1);
    label = lv_label_create(obj);
    lv_label_set_text(label, "c2, r0");

    /*Cell to 1;1 but 2 column wide (span = 2).Set width and height to stretched.*/
    obj = lv_obj_create(cont);
    lv_obj_set_size(obj, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
    lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, 1, 2,
                             LV_GRID_ALIGN_STRETCH, 1, 1);
    label = lv_label_create(obj);
    lv_label_set_text(label, "c1-2, r1");

    /*Cell to 0;1 but 2 rows tall (span = 2).Set width and height to stretched.*/
    obj = lv_obj_create(cont);
    lv_obj_set_size(obj, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
    lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, 0, 1,
                             LV_GRID_ALIGN_STRETCH, 1, 2);
    label = lv_label_create(obj);
    lv_label_set_text(label, "c0\nr1-2");
```

(下页继续)

(续上页)

```
}

#endif
```

```
#
# Demonstrate cell placement and span
#

col_dsc = [70, 70, 70, lv.GRID_TEMPLATE.LAST]
row_dsc = [50, 50, 50, lv.GRID_TEMPLATE.LAST]

# Create a container with grid
cont = lv.obj(lv.scr_act())
cont.set_grid_dsc_array(col_dsc, row_dsc)
cont.set_size(300, 220)
cont.center()

# Cell to 0;0 and align to the start (left/top) horizontally and vertically too
obj = lv.obj(cont)
obj.set_size(lv.SIZE.CONTENT, lv.SIZE.CONTENT)
obj.set_grid_cell(lv.GRID_ALIGN.START, 0, 1,
                  lv.GRID_ALIGN.START, 0, 1)
label = lv.label(obj)
label.set_text("c0, r0")

# Cell to 1;0 and align to the start (left) horizontally and center vertically too
obj = lv.obj(cont)
obj.set_size(lv.SIZE.CONTENT, lv.SIZE.CONTENT)
obj.set_grid_cell(lv.GRID_ALIGN.START, 1, 1,
                  lv.GRID_ALIGN.CENTER, 0, 1)
label = lv.label(obj)
label.set_text("c1, r0")

# Cell to 2;0 and align to the start (left) horizontally and end (bottom) vertically␣
↪too
obj = lv.obj(cont)
obj.set_size(lv.SIZE.CONTENT, lv.SIZE.CONTENT)
obj.set_grid_cell(lv.GRID_ALIGN.START, 2, 1,
                  lv.GRID_ALIGN.END, 0, 1)
label = lv.label(obj)
label.set_text("c2, r0")

# Cell to 1;1 but 2 column wide (span = 2).Set width and height to stretched.
```

(下页继续)

(续上页)

```
obj = lv.obj(cont)
obj.set_size(lv.SIZE.CONTENT, lv.SIZE.CONTENT)
obj.set_grid_cell(lv.GRID_ALIGN.STRETCH, 1, 2,
                  lv.GRID_ALIGN.STRETCH, 1, 1)
label = lv.label(obj)
label.set_text("c1-2, r1")

# Cell to 0;1 but 2 rows tall (span = 2).Set width and height to stretched.
obj = lv.obj(cont)
obj.set_size(lv.SIZE.CONTENT, lv.SIZE.CONTENT)
obj.set_grid_cell(lv.GRID_ALIGN.STRETCH, 0, 1,
                  lv.GRID_ALIGN.STRETCH, 1, 2)
label = lv.label(obj)
label.set_text("c0\nr1-2")
```

Demonstrate grid's "free unit"

```
#include "../../lv_examples.h"
#if LV_USE_GRID && LV_BUILD_EXAMPLES

/**
 * Demonstrate grid's "free unit"
 */
void lv_example_grid_3(void)
{
    /*Column 1: fix width 60 px
     *Column 2: 1 unit from the remaining free space
     *Column 3: 2 unit from the remaining free space*/
    static lv_coord_t col_dsc[] = {60, LV_GRID_FR(1), LV_GRID_FR(2), LV_GRID_TEMPLATE_
↪LAST};

    /*Row 1: fix width 50 px
     *Row 2: 1 unit from the remaining free space
     *Row 3: fix width 50 px*/
    static lv_coord_t row_dsc[] = {50, LV_GRID_FR(1), 50, LV_GRID_TEMPLATE_LAST};

    /*Create a container with grid*/
    lv_obj_t * cont = lv_obj_create(lv_scr_act());
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set_grid_dsc_array(cont, col_dsc, row_dsc);
```

(下页继续)

(续上页)

```
    lv_obj_t * label;
    lv_obj_t * obj;
    uint32_t i;
    for(i = 0; i < 9; i++) {
        uint8_t col = i % 3;
        uint8_t row = i / 3;

        obj = lv_obj_create(cont);
        /*Stretch the cell horizontally and vertically too
         *Set span to 1 to make the cell 1 column/row sized*/
        lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, col, 1,
                             LV_GRID_ALIGN_STRETCH, row, 1);

        label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "%d,%d", col, row);
        lv_obj_center(label);
    }
}

#endif
```

```
#
# Demonstrate grid's "free unit"
#

# Column 1: fix width 60 px
# Column 2: 1 unit from the remaining free space
# Column 3: 2 unit from the remaining free space

col_dsc = [60, lv.grid_fr(1), lv.grid_fr(2), lv.GRID_TEMPLATE.LAST]

# Row 1: fix width 60 px
# Row 2: 1 unit from the remaining free space
# Row 3: fix width 60 px

row_dsc = [40, lv.grid_fr(1), 40, lv.GRID_TEMPLATE.LAST]

# Create a container with grid
cont = lv.obj(lv.scr_act())
cont.set_size(300, 220)
cont.center()
cont.set_grid_dsc_array(col_dsc, row_dsc)
```

(下页继续)

(续上页)

```
for i in range(9):
    col = i % 3
    row = i // 3

    obj = lv.obj(cont)
    # Stretch the cell horizontally and vertically too
    # Set span to 1 to make the cell 1 column/row sized
    obj.set_grid_cell(lv.GRID_ALIGN.STRETCH, col, 1,
                      lv.GRID_ALIGN.STRETCH, row, 1)

    label = lv.label(obj)
    label.set_text("%d,%d"%(col, row))
    label.center()
```

Demonstrate track placement

```
#include "../../lv_examples.h"
#if LV_USE_GRID && LV_BUILD_EXAMPLES

/**
 * Demonstrate track placement
 */
void lv_example_grid_4(void)
{
    static lv_coord_t col_dsc[] = {60, 60, 60, LV_GRID_TEMPLATE_LAST};
    static lv_coord_t row_dsc[] = {45, 45, 45, LV_GRID_TEMPLATE_LAST};


    /*Add space between the columns and move the rows to the bottom (end)*/

    /*Create a container with grid*/
    lv_obj_t * cont = lv_obj_create(lv_scr_act());
    lv_obj_set_grid_align(cont, LV_GRID_ALIGN_SPACE_BETWEEN, LV_GRID_ALIGN_END);
    lv_obj_set_grid_dsc_array(cont, col_dsc, row_dsc);
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);

    lv_obj_t * label;
    lv_obj_t * obj;
    uint32_t i;
    for(i = 0; i < 9; i++) {
```

(下页继续)

(续上页)

```
        uint8_t col = i % 3;
        uint8_t row = i / 3;

        obj = lv_obj_create(cont);
        /*Stretch the cell horizontally and vertically too
         *Set span to 1 to make the cell 1 column/row sized*/
        lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, col, 1,
                             LV_GRID_ALIGN_STRETCH, row, 1);

        label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "%d,%d", col, row);
        lv_obj_center(label);
    }
}

#endif
```

```
#
# Demonstrate track placement
#

col_dsc = [60, 60, 60, lv.GRID_TEMPLATE.LAST]
row_dsc = [40, 40, 40, lv.GRID_TEMPLATE.LAST]


# Add space between the columns and move the rows to the bottom (end)

# Create a container with grid
cont = lv.obj(lv.scr_act())
cont.set_grid_align(lv.GRID_ALIGN.SPACE_BETWEEN, lv.GRID_ALIGN.END)
cont.set_grid_dsc_array(col_dsc, row_dsc)
cont.set_size(300, 220)
cont.center()


for i in range(9):
    col = i % 3
    row = i // 3

    obj = lv.obj(cont)
    # Stretch the cell horizontally and vertically too
    # Set span to 1 to make the cell 1 column/row sized
    obj.set_grid_cell(lv.GRID_ALIGN.STRETCH, col, 1,
```

(下页继续)

(续上页)

```
                      lv.GRID_ALIGN.STRETCH, row, 1)

    label = lv.label(obj)
    label.set_text("{:d}{:d}".format(col, row))
    label.center()
```

Demonstrate column and row gap

```
#include "../../lv_examples.h"
#if LV_USE_GRID && LV_BUILD_EXAMPLES

static void row_gap_anim(void * obj, int32_t v)
{
    lv_obj_set_style_pad_row(obj, v, 0);
}

static void column_gap_anim(void * obj, int32_t v)
{
    lv_obj_set_style_pad_column(obj, v, 0);
}

/**
 * Demonstrate column and row gap
 */
void lv_example_grid_5(void)
{

    /*60x60 cells*/
    static lv_coord_t col_dsc[] = {60, 60, 60, LV_GRID_TEMPLATE_LAST};
    static lv_coord_t row_dsc[] = {45, 45, 45, LV_GRID_TEMPLATE_LAST};

    /*Create a container with grid*/
    lv_obj_t * cont = lv_obj_create(lv_scr_act());
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set_grid_dsc_array(cont, col_dsc, row_dsc);

    lv_obj_t * label;
    lv_obj_t * obj;
    uint32_t i;
    for(i = 0; i < 9; i++) {
```

(下页继续)

(续上页)

```
        uint8_t col = i % 3;
        uint8_t row = i / 3;

        obj = lv_obj_create(cont);
        lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, col, 1,
                                  LV_GRID_ALIGN_STRETCH, row, 1);
        label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "%d,%d", col, row);
        lv_obj_center(label);
    }

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_var(&a, cont);
    lv_anim_set_values(&a, 0, 10);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);

    lv_anim_set_exec_cb(&a, row_gap_anim);
    lv_anim_set_time(&a, 500);
    lv_anim_set_playback_time(&a, 500);
    lv_anim_start(&a);

    lv_anim_set_exec_cb(&a, column_gap_anim);
    lv_anim_set_time(&a, 3000);
    lv_anim_set_playback_time(&a, 3000);
    lv_anim_start(&a);
}

#endif
```

```
def row_gap_anim(obj, v):
    obj.set_style_pad_row(v, 0)

def column_gap_anim(obj, v):
    obj.set_style_pad_column(v, 0)

#
# Demonstrate column and row gap
#

# 60x60 cells
col_dsc = [60, 60, 60, lv.GRID_TEMPLATE.LAST]
```

(下页继续)

(续上页)

```
row_dsc = [40, 40, 40, lv.GRID_TEMPLATE.LAST]

# Create a container with grid
cont = lv.obj(lv.scr_act())
cont.set_size(300, 220)
cont.center()
cont.set_grid_dsc_array(col_dsc, row_dsc)

for i in range(9):
    col = i % 3
    row = i // 3

    obj = lv.obj(cont)
    obj.set_grid_cell(lv.GRID_ALIGN.STRETCH, col, 1,
                      lv.GRID_ALIGN.STRETCH, row, 1)
    label = lv.label(obj)
    label.set_text("{:d},{:d}".format(col, row))
    label.center()

    a_row = lv.anim_t()
    a_row.init()
    a_row.set_var(cont)
    a_row.set_values(0, 10)
    a_row.set_repeat_count(lv.ANIM_REPEAT.INFINITE)
    a_row.set_time(500)
    a_row.set_playback_time(500)
    a_row. set_custom_exec_cb(lambda a,val: row_gap_anim(cont,val))
    lv.anim_t.start(a_row)

    a_col = lv.anim_t()
    a_col.init()
    a_col.set_var(cont)
    a_col.set_values(0, 10)
    a_col.set_repeat_count(lv.ANIM_REPEAT.INFINITE)
    a_col.set_time(500)
    a_col.set_playback_time(500)
    a_col. set_custom_exec_cb(lambda a,val: column_gap_anim(cont,val))
    lv.anim_t.start(a_col)
```

Demonstrate RTL direction on grid

```
#include "../../lv_examples.h"
#if LV_USE_GRID && LV_BUILD_EXAMPLES

/**
 * Demonstrate RTL direction on grid
 */
void lv_example_grid_6(void)
{

    static lv_coord_t col_dsc[] = {60, 60, 60, LV_GRID_TEMPLATE_LAST};
    static lv_coord_t row_dsc[] = {45, 45, 45, LV_GRID_TEMPLATE_LAST};

    /*Create a container with grid*/
    lv_obj_t * cont = lv_obj_create(lv_scr_act());
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set_style_base_dir(cont, LV_BASE_DIR_RTL, 0);
    lv_obj_set_grid_dsc_array(cont, col_dsc, row_dsc);

    lv_obj_t * label;
    lv_obj_t * obj;
    uint32_t i;
    for(i = 0; i < 9; i++) {
        uint8_t col = i % 3;
        uint8_t row = i / 3;

        obj = lv_obj_create(cont);
        /*Stretch the cell horizontally and vertically too
         *Set span to 1 to make the cell 1 column/row sized*/
        lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, col, 1,
                             LV_GRID_ALIGN_STRETCH, row, 1);

        label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "%d,%d", col, row);
        lv_obj_center(label);
    }
}

#endif
```

```
#
# Demonstrate RTL direction on grid
```

(下页继续)

(续上页)

```
#
col_dsc = [60, 60, 60, lv.GRID_TEMPLATE.LAST]
row_dsc = [40, 40, 40, lv.GRID_TEMPLATE.LAST]

# Create a container with grid
cont = lv.obj(lv.scr_act())
cont.set_size(300, 220)
cont.center()
cont.set_style_base_dir(lv.BASE_DIR.RTL,0)
cont.set_grid_dsc_array(col_dsc, row_dsc)

for i in range(9):
    col = i % 3
    row = i // 3

    obj = lv.obj(cont)
    # Stretch the cell horizontally and vertically too
    # Set span to 1 to make the cell 1 column/row sized
    obj.set_grid_cell(lv.GRID_ALIGN.STRETCH, col, 1,
                      lv.GRID_ALIGN.STRETCH, row, 1)

    label = lv.label(obj)
    label.set_text("{:d},{:d}".format(col, row))
    label.center()
```

API

Enums

enum **lv_grid_align_t**

Values:

enumerator **LV_GRID_ALIGN_START**

enumerator **LV_GRID_ALIGN_CENTER**

enumerator **LV_GRID_ALIGN_END**

enumerator **LV_GRID_ALIGN_STRETCH**

enumerator **LV_GRID_ALIGN_SPACE_EVENLY**

enumerator **LV_GRID_ALIGN_SPACE_AROUND**

enumerator **LV_GRID_ALIGN_SPACE_BETWEEN**

Functions

LV_EXPORT_CONST_INT(LV_GRID_CONTENT)

LV_EXPORT_CONST_INT(LV_GRID_TEMPLATE_LAST)

void **lv_grid_init**(void)

void **lv_obj_set_grid_dsc_array**(*lv_obj_t* *obj, const lv_coord_t col_dsc[], const lv_coord_t row_dsc[])

void **lv_obj_set_grid_align**(*lv_obj_t* *obj, *lv_grid_align_t* column_align, *lv_grid_align_t* row_align)

void **lv_obj_set_grid_cell**(*lv_obj_t* *obj, *lv_grid_align_t* column_align, uint8_t col_pos, uint8_t col_span, *lv_grid_align_t* row_align, uint8_t row_pos, uint8_t row_span)

Set the cell of an object. The object's parent needs to have grid layout, else nothing will happen

参数

- **obj** -- pointer to an object
- **column_align** -- the vertical alignment in the cell. LV_GRID_START/END/CENTER/STRETCH
- **col_pos** -- column ID
- **col_span** -- number of columns to take (>= 1)
- **row_align** -- the horizontal alignment in the cell. LV_GRID_START/END/CENTER/STRETCH
- **row_pos** -- row ID
- **row_span** -- number of rows to take (>= 1)

static inline lv_coord_t **lv_grid_fr**(uint8_t x)

Just a wrapper to LV_GRID_FR for bindings.

void **lv_style_set_grid_row_dsc_array**(*lv_style_t* *style, const lv_coord_t value[])

void **lv_style_set_grid_column_dsc_array**(*lv_style_t* *style, const lv_coord_t value[])

void **lv_style_set_grid_row_align**(*lv_style_t* *style, *lv_grid_align_t* value)

void **lv_style_set_grid_column_align**(*lv_style_t* *style, *lv_grid_align_t* value)

void **lv_style_set_grid_cell_column_pos**(*lv_style_t* *style, lv_coord_t value)

void **lv_style_set_grid_cell_column_span**(*lv_style_t* *style, lv_coord_t value)

void **lv_style_set_grid_cell_row_pos**(*lv_style_t* *style, lv_coord_t value)

void **lv_style_set_grid_cell_row_span**(*lv_style_t* *style, lv_coord_t value)

void **lv_style_set_grid_cell_x_align**(*lv_style_t* *style, lv_coord_t value)

void **lv_style_set_grid_cell_y_align**(*lv_style_t* *style, lv_coord_t value)

void **lv_obj_set_style_grid_row_dsc_array**(*lv_obj_t* *obj, const lv_coord_t value[], lv_style_selector_t selector)

void **lv_obj_set_style_grid_column_dsc_array**(*lv_obj_t* *obj, const lv_coord_t value[], lv_style_selector_t selector)

void **lv_obj_set_style_grid_row_align**(*lv_obj_t* *obj, *lv_grid_align_t* value, lv_style_selector_t selector)

void **lv_obj_set_style_grid_column_align**(*lv_obj_t* *obj, *lv_grid_align_t* value, lv_style_selector_t selector)

void **lv_obj_set_style_grid_cell_column_pos**(*lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

void **lv_obj_set_style_grid_cell_column_span**(*lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

void **lv_obj_set_style_grid_cell_row_pos**(*lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

void **lv_obj_set_style_grid_cell_row_span**(*lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

void **lv_obj_set_style_grid_cell_x_align**(*lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

void **lv_obj_set_style_grid_cell_y_align**(*lv_obj_t* *obj, lv_coord_t value, lv_style_selector_t selector)

static inline const lv_coord_t ***lv_obj_get_style_grid_row_dsc_array**(const *lv_obj_t* *obj, uint32_t part)

static inline const lv_coord_t ***lv_obj_get_style_grid_column_dsc_array**(const *lv_obj_t* *obj, uint32_t part)

static inline *lv_grid_align_t* **lv_obj_get_style_grid_row_align**(const *lv_obj_t* *obj, uint32_t part)

static inline *lv_grid_align_t* **lv_obj_get_style_grid_column_align**(const *lv_obj_t* *obj, uint32_t part)

static inline lv_coord_t **lv_obj_get_style_grid_cell_column_pos**(const *lv_obj_t* *obj, uint32_t part)

static inline lv_coord_t **lv_obj_get_style_grid_cell_column_span**(const *lv_obj_t* *obj, uint32_t part)

static inline lv_coord_t **lv_obj_get_style_grid_cell_row_pos**(const *lv_obj_t* *obj, uint32_t part)

static inline lv_coord_t **lv_obj_get_style_grid_cell_row_span**(const *lv_obj_t* *obj, uint32_t part)

static inline lv_coord_t **lv_obj_get_style_grid_cell_x_align**(const *lv_obj_t* *obj, uint32_t part)

static inline lv_coord_t **lv_obj_get_style_grid_cell_y_align**(const *lv_obj_t* *obj, uint32_t part)

Variables

uint32_t **LV_LAYOUT_GRID**

lv_style_prop_t **LV_STYLE_GRID_COLUMN_DSC_ARRAY**

lv_style_prop_t **LV_STYLE_GRID_COLUMN_ALIGN**

lv_style_prop_t **LV_STYLE_GRID_ROW_DSC_ARRAY**

lv_style_prop_t **LV_STYLE_GRID_ROW_ALIGN**

lv_style_prop_t **LV_STYLE_GRID_CELL_COLUMN_POS**

lv_style_prop_t **LV_STYLE_GRID_CELL_COLUMN_SPAN**

lv_style_prop_t **LV_STYLE_GRID_CELL_X_ALIGN**

lv_style_prop_t **LV_STYLE_GRID_CELL_ROW_POS**

lv_style_prop_t **LV_STYLE_GRID_CELL_ROW_SPAN**

lv_style_prop_t **LV_STYLE_GRID_CELL_Y_ALIGN**

2.8 3rd party libraries（第 3 方库）

2.8.1 File System Interfaces(文件系统接口)

LVGL has a File system module to provide an abstraction layer for various file system drivers.

LVG has built in support for:

- FATFS
- STDIO (Linux and Windows using C standard function .e.g fopen, fread)
- POSIX (Linux and Windows using POSIX function .e.g open, read)
- WIN32 (Windows using Win32 API function .e.g CreateFileA, ReadFile)

You still need to provide the drivers and libraries, this extension provides only the bridge between FATFS, STDIO, POSIX, WIN32 and LVGL.

LVGL 有一个文件系统模块，可为各种类型的文件系统驱动程序提供一个通用的抽象层接口。

LVG 内置支持以下文件系统：

- FATFS
- STDIO (Linux 和 Windows 都可以使用的 C 标准函数接口，比如：fopen, fread...)
- POSIX (Linux 和 Windows 都可以使用的 POSIX 函数接口，比如：open, read...)
- WIN32 (Windows 使用 Win32 API 函数接口比如：CreateFileA, ReadFile...)

Usage(用法)

In `lv_conf.h` enable `LV_USE_FS_...` and assign an upper cased letter to `LV_FS_..._LETTER` (e.g. `'S'`). After that you can access files using that driver letter. E.g. `"S:path/to/file.txt"`.

The work directory can be set with `LV_FS_..._PATH`. E.g. `"/home/joe/projects/"` The actual file/directory paths will be appended to it.

Cached reading is also supported if `LV_FS_..._CACHE_SIZE` is set to not `0` value. `lv_fs_read` caches this size of data to lower the number of actual reads from the storage.

如果你使用的是上面说到的几种文件系统 (并且本身可以正常工作)，那么可以根据自己的文件系统在 `lv_conf.h` 中打开 `LV_USE_FS_...` 然后在 `LV_FS_..._LETTER` 分配一个盘符 (驱动程序号) (一般是大写字母，比如：`'S'`)

之后，您就可以通过 lvgl 提供的文件系统接口访问指定的盘符中的文件。例如：`"S:path/to/file.txt"` 。

你可以通过设置 `LV_FS_..._PATH` 来指定一个工作目录，比如：`"/home/joe/projects/"` 。实际的文件/目录路径将附加到它上面，比如：访问 `"/home/joe/projects/file.txt"` 时，直接写 `"file.txt"` 即可。

lvgl 的文件系统抽象接口支持设置文件访问缓冲区，可以通过设置 `LV_FS_..._CACHE_SIZE` 的值来指定缓冲区的大小 (默认是 0)，这样的好处是可以减少对从存储设备的实际操作次数，提高效率。

拓展阅读

2.8.2 BMP decoder（BMP 解码器）

This extension allows the use of BMP images in LVGL. This implementation uses bmp-decoder library. The pixels are read on demand (not the whole image is loaded) so using BMP images requires very little RAM.

If enabled in `lv_conf.h` by `LV_USE_BMP` LVGL will register a new image decoder automatically so BMP files can be directly used as image sources. For example:

```
lv_img_set_src(my_img, "S:path/to/picture.bmp");
```

Note that, a file system driver needs to registered to open images from files. Read more about it here or just enable one in `lv_conf.h` with `LV_USE_FS_...`

这个拓展库让我们可以在 LVGL 中使用 BMP 图像。其中使用的是 bmp-decoder 库。bmp 图的像素是按需读取的（不是加载整个图像），因此使用 BMP 图像需要很少的 RAM。

在 lv_conf.h 中打开 LV_USE_BMP 后，LVGL 会在初始化时自动注册一个新的 BMP 图象解码器，之后就可以直接打开 BMP 格式的图像文件。例如：

```
lv_img_set_src(my_img, "S:path/to/picture.bmp");
```

注意，需要注册文件系统驱动程序才能从文件系统中打开 bmp 图像，点击这里阅读了解更多信息，或直接在 `lv_conf.h` 中打开其中一个类似 `LV_USE_FS_...` 的宏。

Limitations（限制）

- Only BMP files are supported and BMP images as C array (`lv_img_dsc_t`) are not. It's because there is no practical differences between how the BMP files and LVGL's image format stores the image data.
- BMP files can be loaded only from file. If you want to store them in flash it's better to convert them to C array with LVGL's image converter.
- The BMP files color format needs to match with `LV_COLOR_DEPTH`. Use GIMP to save the image in the required format. Both RGB888 and ARGB888 works with `LV_COLOR_DEPTH 32`
- Palette is not supported.
- Because not the whole image is read in can not be zoomed or rotated.
- 仅支持 BMP 文件，不支持像 C 数组 (`lv_img_dsc_t`) 这样的 BMP 图像数据。这是因为 BMP 文件和 LVGL 的图像格式存储图像数据的方式没有实际区别。
- BMP 文件只能从文件中加载。如果您想将它们存储在闪存 (FLASH) 中，最好使用 LVGL 的图像转换器将它们转换为 C 数组。
- BMP 文件的颜色格式需要与 `lv_conf.h` 中的 `LV_COLOR_DEPTH` 匹配。使用 GIMP 以所需格式保存图像。RGB888 和 ARGB888 都适用于 `LV_COLOR_DEPTH 32`
- 不支持调色板。
- 因为不是整个图像被读入后不能进行缩放或旋转。

拓展阅读

Example

Open a BMP image from file

```
#include "../../lv_examples.h"
#if LV_USE_BMP && LV_BUILD_EXAMPLES

/**
 * Open a BMP file from a file
 */
```

(下页继续)

(续上页)

```
void lv_example_bmp_1(void)
{
    lv_obj_t * img = lv_img_create(lv_scr_act());
    /* Assuming a File system is attached to letter 'A'
     * E.g. set LV_USE_FS_STDIO 'A' in lv_conf.h */
#if LV_COLOR_DEPTH == 32
    lv_img_set_src(img, "A:lvgl/examples/libs/bmp/example_32bit.bmp");
#elif LV_COLOR_DEPTH == 16
    lv_img_set_src(img, "A:lvgl/examples/libs/bmp/example_16bit.bmp");
#endif
    lv_obj_center(img);

}

#endif
```

```
#!/opt/bin/lv_micropython -i
import lvgl as lv
import display_driver
import fs_driver

fs_drv = lv.fs_drv_t()
fs_driver.fs_register(fs_drv, 'S')

img = lv.img(lv.scr_act())
# The File system is attached to letter 'S'

img.set_src("S:example_32bit.bmp")
img.center()
```

API

Functions

void **lv_bmp_init**(void)

2.8.3 JPG decoder（JPG 解码器）

Allow the use of JPG images in LVGL. Besides that it also allows the use of a custom format, called Split JPG (SJPG), which can be decoded in more optimal way on embedded systems.

JPG 解码器让我们可以在 LVGL 中使用 JPG 图像。除此之外，它还允许使用称为拆分 JPG (SJPG) 的自定义格式，这个格式可以在嵌入式系统上以更优化的方式进行解码。

Overview（概述）

- Supports both normal JPG and the custom SJPG formats.
- Decoding normal JPG consumes RAM with the size fo the whole uncompressed image (recommended only for devices with more RAM)
- SJPG is a custom format based on ”normal” JPG and specially made for LVGL.
- SJPG is ’split-jpeg’ which is a bundle of small jpeg fragments with an sjpg header.
- SJPG size will be almost comparable to the jpg file or might be a slightly larger.
- File read from file and c-array are implemented.
- SJPEG frame fragment cache enables fast fetching of lines if available in cache.
- By default the sjpg image cache will be image width * 2 * 16 bytes (can be modified)
- Currently only 16 bit image format is supported (TODO)
- Only the required partion of the JPG and SJPG images are decoded, therefore they can’t be zoomed or rotated.
- 支持普通 JPG 和自定义 SJPG 格式。
- 解码普通 JPG 会消耗整个未压缩图像大小的 RAM（仅推荐用于具有更多 RAM 的设备）
- SJPG 是一种基于“普通”JPG 的自定义格式，非常适合在 LVGL 上使用。
- SJPG 是 `split-jpeg`，它是一段带有 sjpg 标头的小 jpeg 片段数据。
- SJPG 大小将几乎与 jpg 文件相当，或者可能会稍大一些。
- 实现了从文件和 c-array 读取文件。
- 如果在缓存中可用，SJPEG 帧片段缓存可以快速获取行。
- 默认情况下，sjpg 图像缓存大小为 **图像宽度 * 2 * 16** 字节（可修改）
- 目前仅支持 16 位图像格式 (TODO)
- 仅对 JPG 和 SJP G 图像的所需部分进行解码，所以无法缩放或旋转。

Usage（用法）

If enabled in `lv_conf.h` by `LV_USE_SJPG` LVGL will register a new image decoder automatically so JPG and SJPG files can be directly used as image sources. For example:

```
lv_img_set_src(my_img, "S:path/to/picture.jpg");
```

Note that, a file system driver needs to registered to open images from files. Read more about it here or just enable one in `lv_conf.h` with `LV_USE_FS_...`

如果在 `lv_conf.h` 中启用了 `LV_USE_SJPG`，LVGL 在初始化时会自动注册 jpg 图像解码器，之后 JPG 和 SJPG 文件可以直接用作图像源使用。例如：

```
lv_img_set_src(my_img, "S:path/to/picture.jpg");
```

注意，这需要注册文件系统驱动程序才能从文件中打开图像。点击这里阅读了解更多信息，或直接在 `lv_conf.h` 中打开其中一个类似 `LV_USE_FS_...` 的宏。

Converter（转换器）

Converting JPG to C array（转换 JPG 为 C 数组）

- Use lvgl online tool https://lvgl.io/tools/imageconverter
- Color format = RAW, output format = C Array
- 使用 lvgl 在线工具：https://lvgl.io/tools/imageconverter
- 颜色格式为 RAW，输出格式为 Array

Converting JPG to SJPG （转换 JPG 为 SJPG）

python3 and the PIL library required. (PIL can be installed with `pip3 install pillow`)

To create SJPG from JPG:

- Copy the image to convert into `lvgl/scripts`
- `cd lvgl/scripts`
- `python3 jpg_to_sjpg.py image_to_convert.jpg`. It creates both a C files and an SJPG image.

The expected result is:

```
Conversion started...

Input:
        image_to_convert.jpg
```

(下页继续)

(续上页)

```
        RES = 640 x 480

Output:
        Time taken = 1.66 sec
        bin size = 77.1 KB
        walpaper.sjpg           (bin file)
        walpaper.c              (c array)

All good!
```

需要 python3 和 PIL 库。(PIL 可以用 pip 安装 pip3 install pillow)

通过 jpg 生成 SJPG:

- 复制 jpg 图像文件到 `lvgl/scripts` 目录下
- 进入到 `lvgl/scripts` 目录: cd lvgl/scripts
- 执行脚本转换脚本, 注意, 这会同时创建一个 C 文件和一个 SJPG 图像: `python3 jpg_to_sjpg.py image_to_convert.jpg`

正常的运行结果:

```
Conversion started...

Input:
        image_to_convert.jpg
        RES = 640 x 480

Output:
        Time taken = 1.66 sec
        bin size = 77.1 KB
        walpaper.sjpg           (bin file)
        walpaper.c              (c array)

All good!
```

拓展阅读

Example

Load an SJPG image

```
#include "../../lv_examples.h"
#if LV_USE_SJPG && LV_BUILD_EXAMPLES

/**
 * Load an SJPG image
 */
void lv_example_sjpg_1(void)
{
  lv_obj_t * wp;

  wp = lv_img_create(lv_scr_act());
  /* Assuming a File system is attached to letter 'A'
   * E.g. set LV_USE_FS_STDIO 'A' in lv_conf.h */
  lv_img_set_src(wp, "A:lvgl/examples/libs/sjpg/small_image.sjpg");
}

#endif
```

```
#!/opt/bin/lv_micropython -i
import lvgl as lv
import display_driver
import fs_driver

fs_drv = lv.fs_drv_t()
fs_driver.fs_register(fs_drv, 'S')

wp = lv.img(lv.scr_act())
# The File system is attached to letter 'S'

wp.set_src("S:small_image.sjpg")
wp.center()
```

API

Functions

void **lv_split_jpeg_init**(void)

2.8.4 PNG decoder（PNG 解码器）

Allow the use of PNG images in LVGL. This implementation uses lodepng library.

If enabled in `lv_conf.h` by `LV_USE_PNG` LVGL will register a new image decoder automatically so PNG files can be directly used as any other image sources.

Note that, a file system driver needs to registered to open images from files. Read more about it here or just enable one in `lv_conf.h` with `LV_USE_FS_...`

The whole PNG image is decoded so during decoding RAM equals to `image width x image height x 4` bytes are required.

As it might take significant time to decode PNG images LVGL's images caching feature can be useful.

PNG 解码器让我们可以在 LVGL 中使用 PNG 图像。这个实现中使用到了 lodepng 库。

如果在 `lv_conf.h` 中启用了 `LV_USE_PNG`，LVGL 在初始化时会自动注册 PNG 图像解码器，之后 PNG 文件可以直接用作图像源使用。

请注意，需要注册文件系统驱动程序才能从文件中打开图像。在此处阅读有关它的更多信息，或者仅在 lv_conf.h 中使用 LV_USE_FS_ 启用一个...

请注意，需要注册文件系统驱动程序才能从文件中打开图像。点击这里阅读关于文件系统的更多信息，或直接在 `lv_conf.h` 中打开其中一个类似 `LV_USE_FS_...` 的宏。

PNG 解码器会对整个 PNG 图像解码，所以在解码过程中需要 RAM 为：`图像宽度 x 图像高度 x 4 字节`

由于解码 PNG 图像可能需要大量时间，这时候 LVGL 的图像缓存功能就能派上用场了（`lv_conf.h` 中的 `LV_IMG_CACHE_DEF_SIZE`）。

拓展阅读

Example

Open a PNG image from file and variable

```c
#include "../../lv_examples.h"
#if LV_USE_PNG && LV_USE_IMG && LV_BUILD_EXAMPLES

/**
 * Open a PNG image from a file and a variable
 */
void lv_example_png_1(void)
{
    LV_IMG_DECLARE(img_wink_png);
    lv_obj_t * img;

    img = lv_img_create(lv_scr_act());
    lv_img_set_src(img, &img_wink_png);
    lv_obj_align(img, LV_ALIGN_LEFT_MID, 20, 0);

    img = lv_img_create(lv_scr_act());
    /* Assuming a File system is attached to letter 'A'
     * E.g. set LV_USE_FS_STDIO 'A' in lv_conf.h */
    lv_img_set_src(img, "A:lvgl/examples/libs/png/wink.png");
    lv_obj_align(img, LV_ALIGN_RIGHT_MID, -20, 0);
}

#endif
```

```python
#!/opt/bin/lv_micropython -i
import lvgl as lv
import display_driver
from imagetools import get_png_info, open_png
from img_wink_png import img_wink_png_map
# Register PNG image decoder
decoder = lv.img.decoder_create()
decoder.info_cb = get_png_info
decoder.open_cb = open_png

img_wink_png = lv.img_dsc_t(
    {
        "header": {"always_zero": 0, "w": 50, "h": 50,  "cf": lv.img.CF.RAW_ALPHA},
        "data_size": 5158,
        "data": img_wink_png_map,
    }
)
img1 = lv.img(lv.scr_act())
img1.set_src(img_wink_png)
img1.align(lv.ALIGN.RIGHT_MID, -250, 0)
```

(下页继续)

(续上页)

```
# Create an image from the png file
try:
    with open('wink.png','rb') as f:
        png_data = f.read()
except:
    print("Could not find wink.png")
    sys.exit()

wink_argb = lv.img_dsc_t({
  'data_size': len(png_data),
  'data': png_data
})

img2 = lv.img(lv.scr_act())
img2.set_src(wink_argb)
img2.align(lv.ALIGN.RIGHT_MID, -150, 0)
```

API

Functions

void **lv_png_init**(void)

Register the PNG decoder functions in LVGL

2.8.5 GIF decoder（GIF 解码器）

Allow using GIF images in LVGL. Based on https://github.com/lecram/gifdec

When enabled in `lv_conf.h` with `LV_USE_GIF` `lv_gif_create(parent)` can be used to create a gif widget.

`lv_gif_set_src(obj, src)` works very similarly to `lv_img_set_src`. As source, it also accepts images as variables (`lv_img_dsc_t`) or files.

GIF 解码器让我们可以在 LVGL 中使用 GIF 图像。基于 https://github.com/lecram/gifdec 实现。

如果在 `lv_conf.h` 中启用了 `LV_USE_GIF` 后，可以使用 `lv_gif_create(parent)` 创建 gif 组件，这与 `lv_img_create` 非常相似。

`lv_gif_set_src(obj, src)` 的工作方式与 `lv_img_set_src` 非常相似。作为源，它还接受图像作为变量 (`lv_img_dsc_t`) 或文件。

Convert GIF files to C array（将 GIF 文件转换为 C 数组）

To convert a GIF file to byte values array use LVGL's online converter. Select "Raw" color format and "C array" Output format.

要将 GIF 文件转换为字节值的数组，请使用 LVGL 的在线转换器。选择 "RAW" 颜色格式和 "C array" 输出格式。

Use GIF images from file（使用 GIF 文件）

For example:

```
lv_gif_set_src(obj, "S:path/to/example.gif");
```

Note that, a file system driver needs to be registered to open images from files. Read more about it here or just enable one in `lv_conf.h` with `LV_USE_FS_...`

示例：

```
lv_gif_create(lv_scr_act());
lv_gif_set_src(obj, "S:path/to/example.gif");
```

注意，需要注册文件系统驱动程序才能从文件中打开图像。点击这里阅读关于文件系统的更多信息，或直接在 `lv_conf.h` 中打开其中一个类似 `LV_USE_FS_...` 的宏。

Memory requirements（内存要求）

To decode and display a GIF animation the following amount of RAM is required:

- `LV_COLOR_DEPTH 8`: 3 x image width x image height
- `LV_COLOR_DEPTH 16`: 4 x image width x image height
- `LV_COLOR_DEPTH 32`: 5 x image width x image height

要解码和显示 GIF 文件，需要以下数量的 RAM (字节)：

- `LV_COLOR_DEPTH 8`：3 x 图像宽度 x 图像高度
- `LV_COLOR_DEPTH 16`：4 x 图像宽度 x 图像高度
- `LV_COLOR_DEPTH 32`：5 x 图像宽度 x 图像高度

拓展阅读

Example

Open a GIF image from file and variable

```
#include "../../lv_examples.h"
#if LV_USE_GIF && LV_BUILD_EXAMPLES

/**
 * Open a GIF image from a file and a variable
 */
void lv_example_gif_1(void)
{
    LV_IMG_DECLARE(img_bulb_gif);
    lv_obj_t * img;

    img = lv_gif_create(lv_scr_act());
    lv_gif_set_src(img, &img_bulb_gif);
    lv_obj_align(img, LV_ALIGN_LEFT_MID, 20, 0);

    img = lv_gif_create(lv_scr_act());
    /* Assuming a File system is attached to letter 'A'
     * E.g. set LV_USE_FS_STDIO 'A' in lv_conf.h */
    lv_gif_set_src(img, "A:lvgl/examples/libs/gif/bulb.gif");
    lv_obj_align(img, LV_ALIGN_RIGHT_MID, -20, 0);
}

#endif
```

```
#!/opt/bin/lv_micropython -i
import lvgl as lv
import display_driver
import fs_driver
from img_bulb_gif import img_bulb_gif_map

fs_drv = lv.fs_drv_t()
fs_driver.fs_register(fs_drv, 'S')
#
# Open a GIF image from a file and a variable
#
img_bulb_gif = lv.img_dsc_t(
    {
        "header": {"always_zero": 0, "w": 0, "h": 0,  "cf": lv.img.CF.RAW},
```

(下页继续)

(续上页)

```
        "data_size": 0,
        "data": img_bulb_gif_map,
    }
)
img1 = lv.gif(lv.scr_act())
img1.set_src(img_bulb_gif)
img1.align(lv.ALIGN.RIGHT_MID, -150, 0)

img2 = lv.gif(lv.scr_act())
# The File system is attached to letter 'S'

img2.set_src("S:bulb.gif")
img2.align(lv.ALIGN.RIGHT_MID, -250, 0)
```

API

Functions

lv_obj_t ***lv_gif_create**(*lv_obj_t* *parent)

void **lv_gif_set_src**(*lv_obj_t* *obj, const void *src)

void **lv_gif_restart**(*lv_obj_t* *gif)

Variables

const lv_obj_class_t **lv_gif_class**

struct **lv_gif_t**

Public Members

lv_img_t **img**

gd_GIF ***gif**

lv_timer_t ***timer**

lv_img_dsc_t **imgdsc**

uint32_t **last_call**

2.8.6 FreeType support

Interface to FreeType to generate font bitmaps run time.

Install FreeType

- Download Freetype from here
- `make`
- `sudo make install`

Add FreeType to your project

- Add include path: `/usr/include/freetype2` (for GCC: `-I/usr/include/freetype2 -L/usr/local/lib`)
- Add library: `freetype` (for GCC: `-L/usr/local/lib -lfreetype`)

Usage

Enable `LV_USE_FREETYPE` in `lv_conf.h`.

To cache the glyphs from the opened fonts, set `LV_FREETYPE_CACHE_SIZE >= 0` and then use the following macros for detailed configuration:

1. `LV_FREETYPE_CACHE_SIZE`:maximum memory(bytes) used to cache font bitmap, outline, character maps, etc. 0 means use the system default value, less than 0 means disable cache. Note: that this value does not account for managed FT_Face and FT_Size objects.
2. `LV_FREETYPE_CACHE_FT_FACES`:maximum number of opened FT_Face objects managed by this cache instance.0 means use the system default value. Only useful when LV_FREETYPE_CACHE_SIZE >= 0.
3. `LV_FREETYPE_CACHE_FT_SIZES`:maximum number of opened FT_Size objects managed by this cache instance. 0 means use the system default value. Only useful when LV_FREETYPE_CACHE_SIZE >= 0.

When you are sure that all the used font sizes will not be greater than 256, you can enable `LV_FREETYPE_SBIT_CACHE`, which is much more memory efficient for small bitmaps.

You can use `lv_ft_font_init()` to create FreeType fonts. It returns `true` to indicate success, at the same time, the `font` member of `lv_ft_info_t` will be filled with a pointer to an LVGL font, and you can use it like any LVGL font.

Font style supports bold and italic, you can use the following macros to set:

1. `FT_FONT_STYLE_NORMAL`:default style.
2. `FT_FONT_STYLE_ITALIC`:Italic style

3. `FT_FONT_STYLE_BOLD`:bold style

They can be combined.eg:`FT_FONT_STYLE_BOLD | FT_FONT_STYLE_ITALIC`.

Note that, the FreeType extension doesn't use LVGL's file system. You can simply pass the path to the font as usual on your operating system or platform.

Example

Open a front with FreeType

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES
#if LV_USE_FREETYPE

/**
 * Load a font with FreeType
 */
void lv_example_freetype_1(void)
{
    /*Create a font*/
    static lv_ft_info_t info;
    /*FreeType uses C standard file system, so no driver letter is required.*/
    info.name = "./lvgl/examples/libs/freetype/arial.ttf";
    info.weight = 24;
    info.style = FT_FONT_STYLE_NORMAL;
    info.mem = NULL;
    if(!lv_ft_font_init(&info)) {
        LV_LOG_ERROR("create failed.");
    }

    /*Create style with the new font*/
    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_text_font(&style, info.font);
    lv_style_set_text_align(&style, LV_TEXT_ALIGN_CENTER);

    /*Create a label with the new style*/
    lv_obj_t * label = lv_label_create(lv_scr_act());
    lv_obj_add_style(label, &style, 0);
    lv_label_set_text(label, "Hello world\nI'm a font created with FreeType");
    lv_obj_center(label);
}
#else
```

(下页继续)

(续上页)

```
void lv_example_freetype_1(void)
{
    /*TODO
     *fallback for online examples*/

    lv_obj_t * label = lv_label_create(lv_scr_act());
    lv_label_set_text(label, "FreeType is not installed");
    lv_obj_center(label);
}

#endif
#endif
```

```
#!/opt/bin/lv_micropython -i
import lvgl as lv
import display_driver
import fs_driver

info = lv.ft_info_t()
info.name ="./arial.ttf"
info.weight = 24
info.style = lv.FT_FONT_STYLE.NORMAL
info.font_init()

# Create style with the new font
style = lv.style_t()
style.init()
style.set_text_font(info.font)
style.set_text_align(lv.TEXT_ALIGN.CENTER)

# Create a label with the new style
label = lv.label(lv.scr_act())
label.add_style(style, 0)
label.set_text("Hello world\nI'm a font created with FreeType")
label.center()
```

Learn more

- FreeType tutorial
- LVGL's font interface

API

Enums

enum **LV_FT_FONT_STYLE**
Values:

enumerator **FT_FONT_STYLE_NORMAL**

enumerator **FT_FONT_STYLE_ITALIC**

enumerator **FT_FONT_STYLE_BOLD**

Functions

bool **lv_freetype_init**(uint16_t max_faces, uint16_t max_sizes, uint32_t max_bytes)
init freetype library

参数

- **max_faces** -- Maximum number of opened FT_Face objects managed by this cache instance. Use 0 for defaults.
- **max_sizes** -- Maximum number of opened FT_Size objects managed by this cache instance. Use 0 for defaults.
- **max_bytes** -- Maximum number of bytes to use for cached data nodes. Use 0 for defaults. Note that this value does not account for managed FT_Face and FT_Size objects.

返回 true on success, otherwise false.

void **lv_freetype_destroy**(void)
Destroy freetype library

bool **lv_ft_font_init**(*lv_ft_info_t* *info)
Creates a font with info parameter specified.

参数 **info** -- See *lv_ft_info_t* for details. when success, lv_ft_info_t->font point to the font you created.

返回 true on success, otherwise false.

void **lv_ft_font_destroy**(lv_font_t *font)

Destroy a font that has been created.

参数 **font** -- pointer to font.

struct **lv_ft_info_t**

Public Members

const char ***name**

const void ***mem**

size_t **mem_size**

lv_font_t ***font**

uint16_t **weight**

uint16_t **style**

2.8.7 QR code

QR code generation with LVGL. Uses QR-Code-generator by nayuki.

Get started

- Download or clone this repository
  - Download from GitHub
  - Clone: git clone https://github.com/lvgl/lv_lib_qrcode.git
- Include the library: `#include "lv_lib_qrcode/lv_qrcode.h"`
- Test with the following code:

```
const char * data = "Hello world";

/*Create a 100x100 QR code*/
lv_obj_t * qr = lv_qrcode_create(lv_scr_act(), 100, lv_color_hex3(0x33f), lv_color_
↪hex3(0xeef));

/*Set data*/
lv_qrcode_update(qr, data, strlen(data));
```

Notes

- QR codes with less data are smaller, but they scaled by an integer number to best fit to the given size.

Example

Create a QR Code

```c
#include "../../lv_examples.h"
#if LV_USE_QRCODE && LV_BUILD_EXAMPLES

/**
 * Create a QR Code
 */
void lv_example_qrcode_1(void)
{
    lv_color_t bg_color = lv_palette_lighten(LV_PALETTE_LIGHT_BLUE, 5);
    lv_color_t fg_color = lv_palette_darken(LV_PALETTE_BLUE, 4);

    lv_obj_t * qr = lv_qrcode_create(lv_scr_act(), 150, fg_color, bg_color);

    /*Set data*/
    const char * data = "https://lvgl.io";
    lv_qrcode_update(qr, data, strlen(data));
    lv_obj_center(qr);

    /*Add a border with bg_color*/
    lv_obj_set_style_border_color(qr, bg_color, 0);
    lv_obj_set_style_border_width(qr, 5, 0);
}

#endif
```

```c
#!/opt/bin/lv_micropython -i
import lvgl as lv
```

(下页继续)

(续上页)

```
import display_driver

bg_color = lv.palette_lighten(lv.PALETTE.LIGHT_BLUE, 5)
fg_color = lv.palette_darken(lv.PALETTE.BLUE, 4)

qr = lv.qrcode(lv.scr_act(), 150, fg_color, bg_color)
# Set data
data = "https://lvgl.io"
qr.update(data,len(data))
qr.center()
# Add a border with bg_color
qr.set_style_border_color(bg_color, 0)
qr.set_style_border_width(5, 0)
```

API

Functions

lv_obj_t ***lv_qrcode_create**(*lv_obj_t* *parent, lv_coord_t size, lv_color_t dark_color, lv_color_t light_color)

Create an empty QR code (an `lv_canvas`) object.

参数

- **parent** -- point to an object where to create the QR code
- **size** -- width and height of the QR code
- **dark_color** -- dark color of the QR code
- **light_color** -- light color of the QR code

返回 pointer to the created QR code object

lv_res_t **lv_qrcode_update**(*lv_obj_t* *qrcode, const void *data, uint32_t data_len)

Set the data of a QR code object

参数

- **qrcode** -- pointer to aQ code object
- **data** -- data to display
- **data_len** -- length of data in bytes

返回 LV_RES_OK: if no error; LV_RES_INV: on error

void **lv_qrcode_delete**(*lv_obj_t* *qrcode)

DEPRECATED: Use normal lv_obj_del instead Delete a QR code object

参数 **qrcode** -- pointer to a QR code object

Variables

const lv_obj_class_t **lv_qrcode_class**

2.8.8 Lottie player

Allows to use Lottie animations in LVGL. Taken from this base repository

LVGL provides the interface to Samsung/rlottie library's C API. That is the actual Lottie player is not part of LVGL, it needs to be built separately.

Build Rlottie

To build Samsung's Rlottie C++14-compatible compiler and optionally CMake 3.14 or higher is required.

To build on desktop you can follow the instructions from Rlottie's README. In the most basic case it looks like this:

```
mkdir rlottie_workdir
cd rlottie_workdir
git clone https://github.com/Samsung/rlottie.git
mkdir build
cd build
cmake ../rlottie
make -j
sudo make install
```

And finally add the `-lrlottie` flag to your linker.

On embedded systems you need to take care of integrating Rlottie to the given build system.

Usage

You can use animation from files or raw data (text). In either case first you need to enable `LV_USE_RLOTTIE` in `lv_conf.h`.

The `width` and `height` of the object be set in the *create* function and the animation will be scaled accordingly.

Use Rlottie from file

To create a Lottie animation from file use:

```
 lv_obj_t * lottie = lv_rlottie_create_from_file(parent, width, height, "path/to/
↪lottie.json");
```

Note that, Rlottie uses the standard STDIO C file API, so you can use the path "normally" and no LVGL specific driver letter is required.

Use Rlottie from raw string data

`lv_example_rlottie_approve.c` contains an example animation in raw format. Instead storing the JSON string a hex array is stored for the following reasons:

- avoid escaping " in the JSON file
- some compilers don't support very long strings

`lvgl/scripts/filetohex.py` can be used to convert a Lottie file a hex array. E.g.:

```
./filetohex.py path/to/lottie.json > out.txt
```

To create an animation from raw data:

```
extern const uint8_t lottie_data[];
lv_obj_t* lottie = lv_rlottie_create_from_raw(parent, width, height, (const char␣
↪*)lottie_data);
```

Getting animations

Lottie is standard and popular format so you can find many animation files on the web. For example: https://lottiefiles.com/

You can also create your own animations with Adobe After Effects or similar software.

Controlling animations

LVGL provides two functions to control the animation mode: `lv_rlottie_set_play_mode` and `lv_rlottie_set_current_frame`. You'll combine your intentions when calling the first method, like in these examples:

```
lv_obj_t * lottie = lv_rlottie_create_from_file(scr, 128, 128, "test.json");
lv_obj_center(lottie);
// Pause to a specific frame
```

(下页继续)

(续上页)

```
lv_rlottie_set_current_frame(lottie, 50);
lv_rlottie_set_play_mode(lottie, LV_RLOTTIE_CTRL_PAUSE); // The specified frame will␣
↪be displayed and then the animation will pause

// Play backward and loop
lv_rlottie_set_play_mode(lottie, LV_RLOTTIE_CTRL_PLAY | LV_RLOTTIE_CTRL_BACKWARD | LV_
↪RLOTTIE_CTRL_LOOP);

// Play forward once (no looping)
lv_rlottie_set_play_mode(lottie, LV_RLOTTIE_CTRL_PLAY | LV_RLOTTIE_CTRL_FORWARD);
```

The default animation mode is **play forward with loop**.

If you don't enable looping, a `LV_EVENT_READY` is sent when the animation can not make more progress without looping.

To get the number of frames in an animation or the current frame index, you can cast the `lv_obj_t` instance to a `lv_rlottie_t` instance and inspect the `current_frame` and `total_frames` members.

Example

Load a Lottie animation from raw data

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES
#if LV_USE_RLOTTIE

/**
 * Load an lottie animation from flash
 */
void lv_example_rlottie_1(void)
{
    extern const uint8_t lv_example_rlottie_approve[];
    lv_obj_t * lottie = lv_rlottie_create_from_raw(lv_scr_act(), 100, 100, (const␣
↪void *)lv_example_rlottie_approve);
    lv_obj_center(lottie);
}

#else
void lv_example_rlottie_1(void)
{
    /*TODO
     *fallback for online examples*/
```

(下页继续)

(续上页)

```
    lv_obj_t * label = lv_label_create(lv_scr_act());
    lv_label_set_text(label, "Rlottie is not installed");
    lv_obj_center(label);
}

#endif
#endif
```

```
#!/opt/bin/lv_micropython -i
import lvgl as lv
import display_driver
#
# Load a lottie animation from flash
#
from lv_example_rlottie_approve import lv_example_rlottie_approve

lottie = lv.rlottie_create_from_raw(lv.scr_act(), 100, 100, lv_example_rlottie_
↪approve)
lottie.center()
```

Load a Lottie animation from a file

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES
#if LV_USE_RLOTTIE

/**
 * Load an lottie animation from file
 */
void lv_example_rlottie_2(void)
{
    /*The rlottie library uses STDIO file API, so there is no driver letter for LVGL*/
    lv_obj_t * lottie = lv_rlottie_create_from_file(lv_scr_act(), 100, 100,
            "lvgl/examples/libs/rlottie/lv_example_rlottie_approve.json");
    lv_obj_center(lottie);
}

#else
void lv_example_rlottie_2(void)
{
```

(下页继续)

(续上页)

```
    /*TODO
     *fallback for online examples*/

    lv_obj_t * label = lv_label_create(lv_scr_act());
    lv_label_set_text(label, "Rlottie is not installed");
    lv_obj_center(label);
}

#endif
#endif
```

```
#!/opt/bin/lv_micropython -i
import lvgl as lv
import display_driver

lottie = lv.rlottie_create_from_file(lv.scr_act(), 100, 100,"lv_example_rlottie_
↪approve.json")
lottie.center()
```

API

Enums

enum **lv_rlottie_ctrl_t**

Values:

enumerator **LV_RLOTTIE_CTRL_FORWARD**

enumerator **LV_RLOTTIE_CTRL_BACKWARD**

enumerator **LV_RLOTTIE_CTRL_PAUSE**

enumerator **LV_RLOTTIE_CTRL_PLAY**

enumerator **LV_RLOTTIE_CTRL_LOOP**

Functions

lv_obj_t ***lv_rlottie_create_from_file**(*lv_obj_t* *parent, lv_coord_t width, lv_coord_t height, const char *path)

lv_obj_t ***lv_rlottie_create_from_raw**(*lv_obj_t* *parent, lv_coord_t width, lv_coord_t height, const char *rlottie_desc)

void **lv_rlottie_set_play_mode**(*lv_obj_t* *rlottie, const *lv_rlottie_ctrl_t* ctrl)

void **lv_rlottie_set_current_frame**(*lv_obj_t* *rlottie, const size_t goto_frame)

Variables

const lv_obj_class_t **lv_rlottie_class**

struct **lv_rlottie_t**

Public Members

lv_img_t **img_ext**

struct Lottie_Animation_S ***animation**

lv_timer_t ***task**

lv_img_dsc_t **imgdsc**

size_t **total_frames**

size_t **current_frame**

size_t **framerate**

uint32_t ***allocated_buf**

size_t **allocated_buffer_size**

size_t **scanline_width**

lv_rlottie_ctrl_t **play_ctrl**

size_t **dest_frame**

2.8.9 FFmpeg support

FFmpeg A complete, cross-platform solution to record, convert and stream audio and video.

Install FFmpeg

- Download FFmpeg from here
- `./configure --disable-all --disable-autodetect --disable-podpages --disable-asm --enable-avcodec --enable-avformat --enable-decoders --enable-encoders --enable-demuxers --enable-parsers --enable-protocol='file' --enable-swscale --enable-zlib`
- `make`
- `sudo make install`

Add FFmpeg to your project

- Add library: `FFmpeg` (for GCC: `-lavformat -lavcodec -lavutil -lswscale -lm -lz -lpthread`)

Usage

Enable `LV_USE_FFMPEG` in `lv_conf.h`.

See the examples below.

Note that, the FFmpeg extension doesn't use LVGL's file system. You can simply pass the path to the image or video as usual on your operating system or platform.

Example

Decode image

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES
#if LV_USE_FFMPEG


/**
 * Open an image from a file
 */
void lv_example_ffmpeg_1(void)
```

(下页继续)

(续上页)

```
{
    lv_obj_t * img = lv_img_create(lv_scr_act());
    lv_img_set_src(img, "./lvgl/examples/libs/ffmpeg/ffmpeg.png");
    lv_obj_center(img);
}

#else

void lv_example_ffmpeg_1(void)
{
    /*TODO
     *fallback for online examples*/

    lv_obj_t * label = lv_label_create(lv_scr_act());
    lv_label_set_text(label, "FFmpeg is not installed");
    lv_obj_center(label);
}

#endif
#endif
```

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs/100ask_lvgl_
↪docs/examples/libs/ffmpeg/lv_example_ffmpeg_1.py
```

Decode video

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES
#if LV_USE_FFMPEG

/**
 * Open a video from a file
 */
void lv_example_ffmpeg_2(void)
{
    /*birds.mp4 is downloaded from http://www.videezy.com (Free Stock Footage by␣
↪Videezy!)
     *https://www.videezy.com/abstract/44864-silhouettes-of-birds-over-the-sunset*/
    lv_obj_t * player = lv_ffmpeg_player_create(lv_scr_act());
    lv_ffmpeg_player_set_src(player, "./lvgl/examples/libs/ffmpeg/birds.mp4");
    lv_ffmpeg_player_set_auto_restart(player, true);
    lv_ffmpeg_player_set_cmd(player, LV_FFMPEG_PLAYER_CMD_START);
```

(下页继续)

(续上页)

```
    lv_obj_center(player);
}

#else

void lv_example_ffmpeg_2(void)
{
    /*TODO
     *fallback for online examples*/

    lv_obj_t * label = lv_label_create(lv_scr_act());
    lv_label_set_text(label, "FFmpeg is not installed");
    lv_obj_center(label);
}

#endif
#endif
```

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs/100ask_lvgl_
↪docs/examples/libs/ffmpeg/lv_example_ffmpeg_2.py
```

API

Enums

enum **lv_ffmpeg_player_cmd_t**

Values:

enumerator **LV_FFMPEG_PLAYER_CMD_START**

enumerator **LV_FFMPEG_PLAYER_CMD_STOP**

enumerator **LV_FFMPEG_PLAYER_CMD_PAUSE**

enumerator **LV_FFMPEG_PLAYER_CMD_RESUME**

enumerator **_LV_FFMPEG_PLAYER_CMD_LAST**

Functions

void **lv_ffmpeg_init**(void)

Register FFMPEG image decoder

int **lv_ffmpeg_get_frame_num**(const char *path)

Get the number of frames contained in the file

参数 **path** -- image or video file name

返回 Number of frames, less than 0 means failed

lv_obj_t ***lv_ffmpeg_player_create**(*lv_obj_t* *parent)

Create ffmpeg_player object

参数 **parent** -- pointer to an object, it will be the parent of the new player

返回 pointer to the created ffmpeg_player

lv_res_t **lv_ffmpeg_player_set_src**(*lv_obj_t* *obj, const char *path)

Set the path of the file to be played

参数

- **obj** -- pointer to a ffmpeg_player object
- **path** -- video file path

返回 LV_RES_OK: no error; LV_RES_INV: can't get the info.

void **lv_ffmpeg_player_set_cmd**(*lv_obj_t* *obj, *lv_ffmpeg_player_cmd_t* cmd)

Set command control video player

参数

- **obj** -- pointer to a ffmpeg_player object
- **cmd** -- control commands

void **lv_ffmpeg_player_set_auto_restart**(*lv_obj_t* *obj, bool en)

Set the video to automatically replay

参数

- **obj** -- pointer to a ffmpeg_player object
- **en** -- true: enable the auto restart

Variables

const lv_obj_class_t **lv_ffmpeg_player_class**

struct **lv_ffmpeg_player_t**

Public Members

lv_img_t **img**

lv_timer_t ***timer**

lv_img_dsc_t **imgdsc**

bool **auto_restart**

struct ffmpeg_context_s ***ffmpeg_ctx**

2.9 Others

2.9.1 Snapshot

Snapshot provides APIs to take snapshot image for LVGL object together with its children. The image will look exactly like the object.

Usage

Simply call API `lv_snapshot_take` to generate the image descriptor which can be set as image object src using `lv_img_set_src`.

Note, only below color formats are supported for now:

- LV_IMG_CF_TRUE_COLOR_ALPHA
- LV_IMG_CF_ALPHA_1BIT
- LV_IMG_CF_ALPHA_2BIT
- LV_IMG_CF_ALPHA_4BIT
- LV_IMG_CF_ALPHA_8BIT

Free the Image

The memory `lv_snapshot_take` uses are dynamically allocated using `lv_mem_alloc`. Use API `lv_snapshot_free` to free the memory it takes. This will firstly free memory the image data takes, then the image descriptor.

Take caution to free the snapshot but not delete the image object. Before free the memory, be sure to firstly unlink it from image object, using `lv_img_set_src(NULL)` and `lv_img_cache_invalidate_src(src)`.

Below code snippet explains usage of this API.

```
void update_snapshot(lv_obj_t * obj, lv_obj_t * img_snapshot)
{
    lv_img_dsc_t* snapshot = (void*)lv_img_get_src(img_snapshot);
    if(snapshot) {
        lv_snapshot_free(snapshot);
    }
    snapshot = lv_snapshot_take(obj, LV_IMG_CF_TRUE_COLOR_ALPHA);
    lv_img_set_src(img_snapshot, snapshot);
}
```

Use Existing Buffer

If the snapshot needs update now and then, or simply caller provides memory, use API `lv_res_t lv_snapshot_take_to_buf(lv_obj_t * obj, lv_img_cf_t cf, lv_img_dsc_t * dsc, void * buf, uint32_t buff_size);` for this case. It's caller's responsibility to alloc/free the memory.

If snapshot is generated successfully, the image descriptor is updated and image data will be stored to provided `buf`.

Note that snapshot may fail if provided buffer is not enough, which may happen when object size changes. It's recommended to use API `lv_snapshot_buf_size_needed` to check the needed buffer size in byte firstly and resize the buffer accordingly.

Example

Simple snapshot example

```
#include "../../lv_examples.h"
#if LV_USE_SNAPSHOT && LV_BUILD_EXAMPLES

static void event_cb(lv_event_t* e)
{
    lv_obj_t * snapshot_obj = lv_event_get_user_data(e);
    lv_obj_t * img = lv_event_get_target(e);
```

(下页继续)

(续上页)

```
    if(snapshot_obj) {
        lv_img_dsc_t* snapshot = (void*)lv_img_get_src(snapshot_obj);
        if(snapshot){
            lv_snapshot_free(snapshot);
        }

        /*Update the snapshot, we know parent of object is the container.*/
        snapshot = lv_snapshot_take(img->parent, LV_IMG_CF_TRUE_COLOR_ALPHA);
        if(snapshot == NULL)
            return;
        lv_img_set_src(snapshot_obj, snapshot);
    }
}

void lv_example_snapshot_1(void)
{
    LV_IMG_DECLARE(img_star);
    lv_obj_t * root = lv_scr_act();
    lv_obj_set_style_bg_color(root, lv_palette_main(LV_PALETTE_LIGHT_BLUE), 0);

    /*Create an image object to show snapshot*/
    lv_obj_t * snapshot_obj = lv_img_create(root);
    lv_obj_set_style_bg_color(snapshot_obj, lv_palette_main(LV_PALETTE_PURPLE), 0);
    lv_obj_set_style_bg_opa(snapshot_obj, LV_OPA_100, 0);
    lv_img_set_zoom(snapshot_obj, 128);
    lv_img_set_angle(snapshot_obj, 300);

    /*Create the container and its children*/
    lv_obj_t * container = lv_obj_create(root);

    lv_obj_center(container);
    lv_obj_set_size(container, 180, 180);
    lv_obj_set_flex_flow(container, LV_FLEX_FLOW_ROW_WRAP);
    lv_obj_set_flex_align(container, LV_FLEX_ALIGN_SPACE_EVENLY, LV_FLEX_ALIGN_CENTER,
↪ LV_FLEX_ALIGN_CENTER);
    lv_obj_set_style_radius(container, 50, 0);
    lv_obj_t * img;
    int i;
    for(i = 0; i < 4; i++) {
        img = lv_img_create(container);
        lv_img_set_src(img, &img_star);
        lv_obj_set_style_bg_color(img, lv_color_black(), 0);
```

(下页继续)

(续上页)

```
        lv_obj_set_style_bg_opa(img, LV_OPA_COVER, 0);
        lv_obj_set_style_transform_zoom(img, 400, LV_STATE_PRESSED);
        lv_obj_add_flag(img, LV_OBJ_FLAG_CLICKABLE);
        lv_obj_add_event_cb(img, event_cb, LV_EVENT_PRESSED, snapshot_obj);
        lv_obj_add_event_cb(img, event_cb, LV_EVENT_RELEASED, snapshot_obj);
    }
}

#endif
```

```
import gc
import lvgl as lv
from imagetools import get_png_info, open_png

# Register PNG image decoder
decoder = lv.img.decoder_create()
decoder.info_cb = get_png_info
decoder.open_cb = open_png

# Measure memory usage
gc.enable()
gc.collect()
mem_free = gc.mem_free()

label = lv.label(lv.scr_act())
label.align(lv.ALIGN.BOTTOM_MID, 0, -10)
label.set_text(" memory free:" + str(mem_free/1024) + " kB")

# Create an image from the png file
try:
    with open('../../assets/img_star.png','rb') as f:
        png_data = f.read()
except:
    print("Could not find star.png")
    sys.exit()

img_star = lv.img_dsc_t({
  'data_size': len(png_data),
  'data': png_data
})

def event_cb(e, snapshot_obj):
    img = e.get_target()
```

(下页继续)

(续上页)

```
    if snapshot_obj:
        # no need to free the old source for snapshot_obj, gc will free it for us.

        # take a new snapshot, overwrite the old one
        dsc = lv.snapshot_take(img.get_parent(), lv.img.CF.TRUE_COLOR_ALPHA)
        snapshot_obj.set_src(dsc)

    gc.collect()
    mem_used = mem_free - gc.mem_free()
    label.set_text("memory used:" + str(mem_used/1024) + " kB")

root = lv.scr_act()
root.set_style_bg_color(lv.palette_main(lv.PALETTE.LIGHT_BLUE), 0)

# Create an image object to show snapshot
snapshot_obj = lv.img(root)
snapshot_obj.set_style_bg_color(lv.palette_main(lv.PALETTE.PURPLE), 0)
snapshot_obj.set_style_bg_opa(lv.OPA.COVER, 0)
snapshot_obj.set_zoom(128)

# Create the container and its children
container = lv.obj(root)
container.align(lv.ALIGN.CENTER, 0, 0)
container.set_size(180, 180)
container.set_flex_flow(lv.FLEX_FLOW.ROW_WRAP)
container.set_flex_align(lv.FLEX_ALIGN.SPACE_EVENLY, lv.FLEX_ALIGN.CENTER, lv.FLEX_
↪ALIGN.CENTER)
container.set_style_radius(50, 0)

for i in range(4):
    img = lv.img(container)
    img.set_src(img_star)
    img.set_style_bg_color(lv.palette_main(lv.PALETTE.GREY), 0)
    img.set_style_bg_opa(lv.OPA.COVER, 0)
    img.set_style_transform_zoom(400, lv.STATE.PRESSED)
    img.add_flag(img.FLAG.CLICKABLE)
    img.add_event_cb(lambda e: event_cb(e, snapshot_obj), lv.EVENT.PRESSED, None)
    img.add_event_cb(lambda e: event_cb(e, snapshot_obj), lv.EVENT.RELEASED, None)
```

API

Functions

lv_img_dsc_t ***lv_snapshot_take**(*lv_obj_t* *obj, *lv_img_cf_t* cf)

Take snapshot for object with its children.

参数

- **obj** -- The object to generate snapshot.
- **cf** -- color format for generated image.

返回 a pointer to an image descriptor, or NULL if failed.

void **lv_snapshot_free**(*lv_img_dsc_t* *dsc)

Free the snapshot image returned by *lv_snapshot_take*

It will firstly free the data image takes, then the image descriptor.

参数 **dsc** -- The image descriptor generated by lv_snapshot_take.

uint32_t **lv_snapshot_buf_size_needed**(*lv_obj_t* *obj, *lv_img_cf_t* cf)

Get the buffer needed for object snapshot image.

参数

- **obj** -- The object to generate snapshot.
- **cf** -- color format for generated image.

返回 the buffer size needed in bytes

lv_res_t **lv_snapshot_take_to_buf**(*lv_obj_t* *obj, *lv_img_cf_t* cf, *lv_img_dsc_t* *dsc, void *buf, uint32_t buff_size)

Take snapshot for object with its children, save image info to provided buffer.

参数

- **obj** -- The object to generate snapshot.
- **cf** -- color format for generated image.
- **dsc** -- image descriptor to store the image result.
- **buff** -- the buffer to store image data.
- **buff_size** -- provided buffer size in bytes.

返回 LV_RES_OK on success, LV_RES_INV on error.

2.9.2 Monkey

A simple monkey test. Use random input to stress test the application.

Usage

Enable `LV_USE_MONKEY` in `lv_conf.h`.

First configure monkey, use `lv_monkey_config_t` to define the configuration structure, set the `type` (check *input devices* for the supported types), and then set the range of `period_range` and `input_range`, the monkey will output random operations at random times within this range. Call `lv_monkey_create` to create monkey. Finally call `lv_monkey_set_enable(monkey, true)` to enable monkey.

If you want to pause the monkey, call `lv_monkey_set_enable(monkey, false)`. To delete the monkey, call `lv_monkey_del(monkey)`.

Note that `input_range` has different meanings in different `type`:

- `LV_INDEV_TYPE_POINTER` No effect, click randomly within the pixels of the screen resolution.
- `LV_INDEV_TYPE_ENCODER` The minimum and maximum values of `enc_diff`.
- `LV_INDEV_TYPE_BUTTON` The minimum and maximum values of `btn_id`. Use `lv_monkey_get_indev()` to get the input device, and use `lv_indev_set_button_points()` to map the key ID to the coordinates.
- `LV_INDEV_TYPE_KEYPAD` No effect, Send random *Keys*.

Example

Touchpad monkey example

```
#include "../../lv_examples.h"
#if LV_USE_MONKEY && LV_BUILD_EXAMPLES

void lv_example_monkey_1(void)
{
    /*Create pointer monkey test*/
    lv_monkey_config_t config;
    lv_monkey_config_init(&config);
    config.type = LV_INDEV_TYPE_POINTER;
    config.period_range.min = 10;
    config.period_range.max = 100;
    lv_monkey_t * monkey = lv_monkey_create(&config);

    /*Start monkey test*/
```

(下页继续)

(续上页)

```
    lv_monkey_set_enable(monkey, true);
}

#endif
```

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs/100ask_lvgl_
↪docs/examples/others/monkey/lv_example_monkey_1.py
```

Encoder monkey example

```
#include "../../lv_examples.h"
#if LV_USE_MONKEY && LV_BUILD_EXAMPLES

void lv_example_monkey_2(void)
{
    /*Create encoder monkey test*/
    lv_monkey_config_t config;
    lv_monkey_config_init(&config);
    config.type = LV_INDEV_TYPE_ENCODER;
    config.period_range.min = 50;
    config.period_range.max = 500;
    config.input_range.min = -5;
    config.input_range.max = 5;
    lv_monkey_t * monkey = lv_monkey_create(&config);

    /*Set the default group*/
    lv_group_t * group = lv_group_create();
    lv_indev_set_group(lv_monkey_get_indev(monkey), group);
    lv_group_set_default(group);

    /*Start monkey test*/
    lv_monkey_set_enable(monkey, true);
}

#endif
```

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs/100ask_lvgl_
↪docs/examples/others/monkey/lv_example_monkey_2.py
```

Button monkey example

```
#include "../../lv_examples.h"
#if LV_USE_MONKEY && LV_BUILD_EXAMPLES

void lv_example_monkey_3(void)
{
    static lv_point_t btn_points[3];
    lv_coord_t hor_res = LV_HOR_RES;

    /*Create button monkey test*/
    lv_monkey_config_t config;
    lv_monkey_config_init(&config);
    config.type = LV_INDEV_TYPE_BUTTON;
    config.period_range.min = 50;
    config.period_range.max = 500;
    config.input_range.min = 0;
    config.input_range.max = sizeof(btn_points) / sizeof(lv_point_t) - 1;
    lv_monkey_t * monkey = lv_monkey_create(&config);

    /*Set the coordinates bound to the button*/
    btn_points[0].x = hor_res / 4;
    btn_points[0].y = 10;
    btn_points[1].x = hor_res / 2;
    btn_points[1].y = 10;
    btn_points[2].x = hor_res * 3 / 4;
    btn_points[2].y = 10;

    lv_indev_set_button_points(lv_monkey_get_indev(monkey), btn_points);

    /*Start monkey test*/
    lv_monkey_set_enable(monkey, true);
}

#endif
```

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs/100ask_lvgl_
↪docs/examples/others/monkey/lv_example_monkey_3.py
```

API

Typedefs

typedef struct _lv_monkey **lv_monkey_t**

Functions

void **lv_monkey_config_init**(*lv_monkey_config_t* *config)

Initialize a monkey config with default values

参数 **config** -- pointer to '*lv_monkey_config_t*' variable to initialize

lv_monkey_t ***lv_monkey_create**(const *lv_monkey_config_t* *config)

Create monkey for test

参数 **config** -- pointer to '*lv_monkey_config_t*' variable

返回 pointer to the created monkey

lv_indev_t ***lv_monkey_get_indev**(*lv_monkey_t* *monkey)

Get monkey input device

参数 **monkey** -- pointer to a monkey

返回 pointer to the input device

void **lv_monkey_set_enable**(*lv_monkey_t* *monkey, bool en)

Enable monkey

参数

- **monkey** -- pointer to a monkey
- **en** -- set to true to enable

bool **lv_monkey_get_enable**(*lv_monkey_t* *monkey)

Get whether monkey is enabled

参数 **monkey** -- pointer to a monkey

返回 return true if monkey enabled

void **lv_monkey_set_user_data**(*lv_monkey_t* *monkey, void *user_data)

Set the user_data field of the monkey

参数

- **monkey** -- pointer to a monkey
- **user_data** -- pointer to the new user_data.

void ***lv_monkey_get_user_data**(*lv_monkey_t* *monkey)*

Get the user_data field of the monkey

参数 **monkey** -- pointer to a monkey

返回 the pointer to the user_data of the monkey

void **lv_monkey_del**(*lv_monkey_t* *monkey)

Delete monkey

参数 **monkey** -- pointer to monkey

struct **lv_monkey_config_t**

Public Members

lv_indev_type_t **type**

< Input device type Monkey execution period

uint32_t **min**

uint32_t **max**

struct *lv_monkey_config_t*::[anonymous] **period_range**

The range of input value

int32_t **min**

int32_t **max**

struct *lv_monkey_config_t*::[anonymous] **input_range**

2.9.3 Grid navigation

Grid navigation (gridnav for short) is a feature that changes the currently focused child object as arrow keys are pressed.

If the children are arranged into a grid-like layout then the up, down, left and right arrows move focus to the nearest sibling in the respective direction.

It doesn't matter how the children are positioned, as only the current x and y coordinates are considered. This means that gridnav works with manually positioned children, as well as Flex and Grid layouts.

Gridnav also works if the children are arranged into a single row or column. That makes it useful, for example, to simplify navigation on a List widget.

Gridnav assumes that the object to which gridnav is added is part of a group. This way, if the object with gridnav is focused, the arrow key presses are automatically forwarded to the object so that gridnav can process the arrow keys.

To move the focus to the next widget of the group use `LV_KEY_NEXT/PREV` or `lv_group_focus_next/prev()` or the `TAB` key on keyboard as usual.

If the container is scrollable and the focused child is out of the view, gridnav will automatically scroll the child into view.

Usage

To add the gridnav feature to an object use `lv_gridnav_add(cont, flags)`.

`flags` control the behavior of gridnav:

- `LV_GRIDNAV_CTRL_NONE` Default settings
- `LV_GRIDNAV_CTRL_ROLLOVER` If there is no next/previous object in a direction, the focus goes to the object in the next/previous row (on left/right keys) or first/last row (on up/down keys
- `LV_GRIDNAV_CTRL_SCROLL_FIRST` If an arrow is pressed and the focused object can be scrolled in that direction then it will be scrolled instead of going to the next/previous object. If there is no more room for scrolling the next/previous object will be focused normally

`lv_gridnav_remove(cont)` Removes gridnav from an object.

Focusable objects

An object needs to be clickable or click focusable (`LV_OBJ_FLAG_CLICKABLE` or `LV_OBJ_FLAG_CLICK_FOCUSABLE`) and not hidden (`LV_OBJ_FLAG_HIDDEN`) to be focusable by gridnav.

Example

Basic grid navigation

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs/100ask_lvgl_
↪docs/examples/others/monkey/lv_example_gridnav_1.c
```

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs/100ask_lvgl_
↪docs/examples/others/monkey/lv_example_gridnav_1.py
```

Grid navigation on a list

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs/100ask_lvgl_
↪docs/examples/others/monkey/lv_example_gridnav_2.c
```

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs/100ask_lvgl_
↪docs/examples/others/monkey/lv_example_gridnav_2.py
```

Nested grid navigations

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs/100ask_lvgl_
↪docs/examples/others/monkey/lv_example_gridanav_3.c
```

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs/100ask_lvgl_
↪docs/examples/others/monkey/lv_example_gridanav_3.py
```

API

Typedefs

typedef int **_keep_pedantic_happy**

Enums

enum **lv_gridnav_ctrl_t**

Values:

enumerator **LV_GRIDNAV_CTRL_NONE**

enumerator **LV_GRIDNAV_CTRL_ROLLOVER**

If there is no next/previous object in a direction, the focus goes to the object in the next/previous row (on left/right keys) or first/last row (on up/down keys)

enumerator **LV_GRIDNAV_CTRL_SCROLL_FIRST**

If an arrow is pressed and the focused object can be scrolled in that direction then it will be scrolled instead of going to the next/previous object. If there is no more room for scrolling the next/previous object will be focused normally

Functions

void **lv_gridnav_add**(*lv_obj_t* *obj, *lv_gridnav_ctrl_t* ctrl)

Add grid navigation feature to an object. It expects the children to be arranged into a grid-like layout. Although it's not required to have pixel perfect alignment. This feature makes possible to use keys to navigate among the children and focus them. The keys other than arrows and press/release related events are forwarded to the focused child.

参数

- **obj** -- pointer to an object on which navigation should be applied.
- **ctrl** -- control flags from `lv_gridnav_ctrl_t`.

void **lv_gridnav_remove**(*lv_obj_t* *obj)

Remove the grid navigation support from an object

参数 **obj** -- pointer to an object

2.10 Contributing（贡献）

2.10.1 Introduction（介绍）

Join LVGL's community and leave your footprint in the library!

There are a lot of ways to contribute to LVGL even if you are are new to the library or even new to programming.

It might be scary to make the first step but you have nothing to be afraid of. A friendly and helpful community is waiting for you. Get to know like-minded people and make something great together.

So let's find which contribution option fits you the best and help you join the development of LVGL!

Before getting started here are some guidelines to make contribution smoother:

- Be kind and friendly.
- Be sure to read the relevant part of the documentation before posting a question.
- Ask questions in the Forum and use GitHub for development-related discussions.
- Always fill out the post or issue templates in the Forum or GitHub (or at least provide equivalent information). It makes understanding your contribution or issue easier and you will get a useful response faster.
- If possible send an absolute minimal but buildable code example in order to reproduce the issue. Be sure it contains all the required variable declarations, constants, and assets (images, fonts).
- Use Markdown to format your posts. You can learn it in 10 minutes.
- Speak about one thing in one issue or topic. It makes your post easier to find later for someone with the same question.

- Give feedback and close the issue or mark the topic as solved if your question is answered.
- For non-trivial fixes and features, it's better to open an issue first to discuss the details instead of sending a pull request directly.
- Please read and follow the Coding style guide.

加入 LVGL 的社区，在图书馆留下你的足迹！

有很多方法可以为 LVGL 做出贡献，即使您是库的新手，甚至是编程的新手。

迈出第一步可能会很可怕，但你没有什么可害怕的。一个友好而乐于助人的社区正等着您。结识志同道合的人，共同创造美好。

那么让我们来找出最适合您的贡献选项，并帮助您加入 LVGL 的开发！

在开始之前，这里有一些指导方针可以使贡献更顺畅：

- 善良和友好。
- 在发布问题之前，请务必阅读文档的相关部分。
- 在论坛 提出问题，并使用GitHub 进行与开发相关的讨论。
- 始终填写论坛或 GitHub 中的帖子或问题模板（或至少提供等效信息）。它使您更容易理解您的贡献或问题，并且您将更快地获得有用的响应。
- 如果可能，发送一个绝对最小但可构建的代码示例以重现问题。确保它包含所有必需的变量声明、常量和资产（图像、字体）。
- 使用 Markdown 来格式化您的帖子。你可以在 10 分钟内学会它。
- 在一个问题或主题中谈论一件事。以后有相同问题的人可以更轻松地找到您的帖子。
- 如果您的问题得到解答，请提供反馈并关闭问题或将主题标记为已解决。
- 对于重要的修复和功能，最好先打开一个问题来讨论细节，而不是直接发送拉取请求。
- 请阅读并遵循编码风格指南。

2.10.2 Pull request（拉取请求）

Merging new code into the lvgl, documentation, blog, examples, and other repositories happen via *Pull requests* (PR for short). A PR is a notification like "Hey, I made some updates to your project. Here are the changes, you can add them if you want." To do this you need a copy (called fork) of the original project under your account, make some changes there, and notify the original repository about your updates. You can see what it looks like on GitHub for LVGL here: https://github.com/lvgl/lvgl/pulls.

To add your changes you can edit files online on GitHub and send a new Pull request from there (recommended for small changes) or add the updates in your favorite editor/IDE and use git to publish the changes (recommended for more complex updates).

将新代码合并到 lvgl、文档、博客、示例和其他存储库中是通过 *Pull 请求 *（简称 PR）进行的。PR 是类似于“嘿，我对您的项目进行了一些更新。这是更改，如果需要，您可以添加它们”的通知。为此，您需要您帐户下的原始项目的副本（称为 fork），在那里进行一些更改，并将您的更新通知原始存储库。您可以在 GitHub 上查看 LVGL 的样子：https://github.com/lvgl/lvgl/pulls。

要添加您的更改，您可以在 GitHub 上在线编辑文件并从那里发送新的拉取请求（建议进行小改动）或在您喜欢的编辑器/IDE 中添加更新并使用 git 发布更改（推荐用于更复杂的更新）。

From GitHub（来自 GitHub）

1. Navigate to the file you want to edit.
2. Click the Edit button in the top right-hand corner.
3. Add your changes to the file.
4. Add a commit message on the bottom of the page.
5. Click the *Propose changes* button.

1. 导航到要编辑的文件。
2. 单击右上角的编辑按钮。
3. 将您的更改添加到文件中。
4. 在页面底部添加提交信息。
5. 单击提议更改按钮。

From command line（从命令行获取）

The instructions describe the main `lvgl` repository but it works the same way for the other repositories.

1. Fork the lvgl repository. To do this click the ”Fork” button in the top right corner. It will ”copy” the `lvgl` repository to your GitHub account (`https://github.com/<YOUR_NAME>?tab=repositories`)
2. Clone your forked repository.
3. Add your changes. You can create a *feature branch* from *master* for the updates: `git checkout -b the-new-feature`
4. Commit and push your changes to the forked `lvgl` repository.
5. Create a PR on GitHub from the page of your `lvgl` repository (`https://github.com/<YOUR_NAME>/lvgl`) by clicking the *”New pull request”* button. Don’t forget to select the branch where you added your changes.
6. Set the base branch. It means where you want to merge your update. In the `lvgl` repo fixes go to `master`, new features to `dev` branch.
7. Describe what is in the update. An example code is welcome if applicable.

8. If you need to make more changes, just update your forked `lvgl` repo with new commits. They will automatically appear in the PR.

这些说明描述了主要的 `lvgl` 存储库，但它的工作方式与其他存储库相同。

1. fork lvgl 仓库。为此，请单击右上角的“叉”按钮。它会将 `lvgl` 存储库“复制”到您的 GitHub 帐户（`https://github.com/<YOUR_NAME>?tab=repositories`）
2. 克隆您的分叉存储库。
3. 添加您的更改。您可以从 *master* 创建一个 *feature* 分支用于更新：`git checkout -b the-new-feature`
4. 提交并将您的更改推送到分叉的 `lvgl` 存储库。
5. 在你的 `lvgl` 存储库页面 (`https://github.com/<YOUR_NAME>/lvgl`) 上点击 *"New pull request"* 按钮，在 GitHub 上创建 PR。不要忘记选择添加更改的分支。
6. 设置基础分支。这意味着您要合并更新的位置。在 `lvgl` 存储库中，修复转到 `master`，`dev` 分支的新功能。
7. 描述更新内容。如果适用，欢迎提供示例代码。
8. 如果您需要进行更多更改，只需使用新提交更新您的分叉 `lvgl` 存储库。它们将自动出现在 PR 中。

Commit message format（commit 的格式）

In commit messages please follow the Angular Commit Format.

Some examples:

在提交消息中，请遵循 Angular Commit Format。

一些例子：

```
fix(img) update size if a new source is set
```

```
fix(bar) fix memory leak

The animations weren't deleted in the destructor.

Fixes: #1234
```

```
feat add span widget

The span widget allows mixing different font sizes, colors and styles.
It's similar to HTML <span>
```

```
docs(porting) fix typo
```

2.10.3 Developer Certification of Origin (DCO)（开发者原产地认证 (DCO)）

Overview（概述）

To ensure all licensing criteria are met for every repository of the LVGL project, we apply a process called DCO (Developer's Certificate of Origin).

The text of DCO can be read here: https://developercertificate.org/.

By contributing to any repositories of the LVGL project you agree that your contribution complies with the DCO.

If your contribution fulfills the requirements of the DCO no further action is needed. If you are unsure feel free to ask us in a comment.

为确保 LVGL 项目的每个存储库都满足所有许可标准，我们应用了一个称为 DCO（开发者原产地证书）的流程。

DCO 的文本可以在这里阅读：https://developercertificate.org/。

通过为 LVGL 项目的任何存储库做出贡献，您同意您的贡献符合 DCO。

如果您的捐款符合 DCO 的要求，则无需采取进一步行动。如果您不确定，请随时在评论中询问我们。

Accepted licenses and copyright notices（接受的许可和版权声明）

To make the DCO easier to digest, here are some practical guides about specific cases:

为了让 DCO 更容易消化，这里有一些关于特定案例的实用指南：

Your own work（你自己的作品）

The simplest case is when the contribution is solely your own work. In this case you can just send a Pull Request without worrying about any licensing issues.

最简单的情况是贡献完全是您自己的工作。在这种情况下，您可以只发送拉取请求而不必担心任何许可问题。

Use code from online source（使用来自网上的代码）

If the code you would like to add is based on an article, post or comment on a website (e.g. StackOverflow) the license and/or rules of that site should be followed.

For example in case of StackOwerflow a notice like this can be used:

如果您要添加的代码基于网站（例如 StackOverflow）上的文章、帖子或评论，则应遵循该网站的许可和/或规则。

例如，在 StackOwerflow 的情况下，可以使用这样的通知：

```
/* The original version of this code-snippet was published on StackOverflow.
 * Post: http://stackoverflow.com/questions/12345
 * Author: http://stackoverflow.com/users/12345/username
 * The following parts of the snippet were changed:
 * - Check this or that
 * - Optimize performance here and there
 */
 ... code snippet here ...
```

Use MIT licensed code（使用 MIT 许可代码）

As LVGL is MIT licensed, other MIT licensed code can be integrated without issues. The MIT license requires a copyright notice be added to the derived work. Any derivative work based on MIT licensed code must copy the original work's license file or text.

由于 LVGL 是 MIT 许可的，因此可以毫无问题地集成其他 MIT 许可代码。MIT 许可证要求在衍生作品中添加版权声明。任何基于 MIT 许可代码的衍生作品必须复制原始作品的许可文件或文本。

Use GPL licensed code（使用 GPL 许可代码）

The GPL license is not compatible with the MIT license. Therefore, LVGL can not accept GPL licensed code.

GPL 许可证与 MIT 许可证不兼容。因此，LVGL 不能接受 GPL 许可代码。

2.10.4 Ways to contribute（贡献方式）

Even if you're just getting started with LVGL there are plenty of ways to get your feet wet. Most of these options don't even require knowing a single line of LVGL code.

Below we have collected some opportunities about the ways you can contribute to LVGL.

即使您刚刚开始使用 LVGL，也有很多方法可以让您的脚变湿。大多数这些选项甚至不需要知道一行 LVGL 代码。

下面我们收集了一些关于您可以为 LVGL 做出贡献的方式的机会。

Give LVGL a Star（给 LVGL 一颗星）

Show that you like LVGL by giving it star on GitHub!

Star

This simple click makes LVGL more visible on GitHub and makes it more attractive to other people. So with this, you already helped a lot!

通过在 GitHub 上给它 Star 来表明你喜欢 LVGL！

<a class="github-button" href="https://github.com/lvgl/lvgl" data-icon="octicon-star" data-size="large" data-show-count="true" 咏叹调 -label="Star lvgl/lvgl on GitHub">Star

这个简单的点击使 LVGL 在 GitHub 上更显眼，并使其对其他人更具吸引力。所以有了这个，你已经帮了很多忙！

Tell what you have achieved（讲述你的成就）

Have you already started using LVGL in a *Simulator*, a development board, or on your custom hardware? Was it easy or were there some obstacles? Are you happy with the result? Showing your project to others is a win-win situation because it increases your and LVGL's reputation at the same time.

You can post about your project on Twitter, Facebook, LinkedIn, create a YouTube video, and so on. Only one thing: On social media don't forget to add a link to `https://lvgl.io` or `https://github.com/lvgl` and use the hashtag `#lvgl`. Thank you! :)

You can also open a new topic in the My projects category of the Forum.

The LVGL Blog welcomes posts from anyone. It's a good place to talk about a project you created with LVGL, write a tutorial, or share some nice tricks. The latest blog posts are shown on the homepage of LVGL to make your work more visible.

The blog is hosted on GitHub. If you add a post GitHub automatically turns it into a website. See the README of the blog repo to see how to add your post.

Any of these help to spread the word and familiarize new developers with LVGL.

If you don't want to speak about your project publicly, feel free to use Contact form on lvgl.io to private message to us.

您是否已经开始在模拟器、开发板或自定义硬件中使用 LVGL？是容易还是有什么障碍？你对结果满意吗？向他人展示您的项目是一个双赢的局面，因为它同时增加了您和 LVGL 的声誉。

您可以在 Twitter、Facebook、LinkedIn 上发布您的项目，创建 YouTube 视频等。只有一件事：在社交媒体上不要忘记添加一个链接到 `https://lvgl.io` 或 `https://github.com/lvgl` 并使用主题标签 `#lvgl`。谢谢！:)

您也可以在论坛的我的项目 类别中打开一个新主题。

LVGL 博客 欢迎任何人发帖。这是谈论您使用 LVGL 创建的项目、编写教程或分享一些不错的技巧的好地方。最新博文展示在【LVGL 首页】(https://lvgl.io)，让你的作品更显眼。

该博客托管在 GitHub 上。如果您添加帖子，GitHub 会自动将其转换为网站。请参阅博客存储库的 README 以了解如何添加您的帖子。

其中任何一项都有助于传播信息并使新开发人员熟悉 LVGL。

如果您不想公开谈论您的项目，请随时使用 lvgl.io 上的联系表格 私信给我们。

Write examples（撰写实例）

As you learn LVGL you will probably play with the features of widgets. Why not publish your experiments?

Each widgets' documentation contains examples. For instance, here are the examples of the Drop-down list widget. The examples are directly loaded from the lvgl/examples folder.

So all you need to do is send a *Pull request* to the lvgl repository and follow some conventions:

- Name the examples like `lv_example_<widget_name>_<index>`.
- Make the example as short and simple as possible.
- Add comments to explain what the example does.
- Use 320x240 resolution.
- Update `index.rst` in the example's folder with your new example. To see how other examples are added, look in the lvgl/examples/widgets folder.

当您学习 LVGL 时，您可能会使用小部件的功能。为什么不发表你的实验?

每个小部件的文档都包含示例。例如，这里是 下拉列表 小部件的示例。示例直接从 lvgl/examples 文件夹加载。

所以你需要做的就是向 lvgl 存储库发送一个*Pull request* 并遵循一些约定：

- 将示例命名为 `lv_example_<widget_name>_<index>`。
- 使示例尽可能简短。
- 添加注释以解释示例的作用。
- 使用 320x240 分辨率。
- 使用您的新示例更新示例文件夹中的 `index.rst`。要查看其他示例是如何添加的，请查看 lvgl/examples/widgets 文件夹。

Improve the docs（改进文档）

As you read the documentation you might see some typos or unclear sentences. All the documentation is located in the lvgl/docs folder. For typos and straightforward fixes, you can simply edit the file on GitHub.

Note that the documentation is also formatted in Markdown.

当您阅读文档时，您可能会看到一些拼写错误或不清楚的句子。所有文档都位于 lvgl/docs 文件夹中。对于拼写错误和直接修复，您只需在 GitHub 上编辑文件即可。

请注意，文档的格式也为 Markdown。

Report bugs（报告 bug）

As you use LVGL you might find bugs. Before reporting them be sure to check the relevant parts of the documentation.

If it really seems like a bug feel free to open an issue on GitHub.

When filing the issue be sure to fill out the template. It helps find the root of the problem while avoiding extensive questions and exchanges with other developers.

当您使用 LVGL 时，您可能会发现错误。在报告它们之前，请务必检查文档的相关部分。

如果它真的看起来像是一个错误，请随时打开 GitHub 上的问题。

提交问题时，请务必填写模板。它有助于找到问题的根源，同时避免广泛的问题和与其他开发人员的交流。

Send fixes（提交补丁）

The beauty of open-source software is you can easily dig in to it to understand how it works. You can also fix or adjust it as you wish.

If you found and fixed a bug don't hesitate to send a *Pull request* with the fix.

In your Pull request please also add a line to `CHANGELOG.md`.

开源软件的美妙之处在于您可以轻松深入了解它的工作原理。您也可以根据需要修复或调整它。

如果您发现并修复了错误，请毫不犹豫地发送带有修复程序的*Pull request*。

在您的拉取请求中，还请在 `CHANGELOG.md` 中添加一行。

Join the conversations in the Forum（参与论坛讨论）

It feels great to know you are not alone if something is not working. It's even better to help others when they struggle with something.

While you were learning LVGL you might have had questions and used the Forum to get answers. As a result, you probably have more knowledge about how LVGL works.

One of the best ways to give back is to use the Forum and answer the questions of newcomers - like you were once.

Just read the titles and if you are familiar with the topic don't hesitate to share your thoughts and suggestions.

Participating in the discussions is one of the best ways to become part of the project and get to know like-minded people!

如果某些事情不起作用，知道您并不孤单，这感觉很棒。在别人遇到困难时帮助他们甚至更好。

当您学习 LVGL 时，您可能会遇到问题并使用论坛来获得答案。因此，您可能对 LVGL 的工作原理有了更多的了解。

回馈的最佳方式之一是使用论坛并回答新人的问题 - 就像您曾经一样。

只需阅读标题，如果您熟悉该主题，请随时分享您的想法和建议。

参与讨论是成为项目的一部分并结识志同道合的人的最佳方式之一！

Add features（添加功能）

If you have created a cool widget, or added useful feature to LVGL feel free to open a new PR for it. We collect the optional features (a.k.a. plugins) in lvgl/src/extra folder so if you are interested in adding a new features please use this folder. The README file describes the basics rules of contribution and and also lists some ideas.

For further ideas take a look at the our *Roadmap* page. If you are interested in any of them feel free to share your opinion and/or participate in the the implementation.

Other features which are (still) not on the road map are listed in the Feature request category of the Forum.

When adding a new features the followings also needs to be updated:

- Update lv_conf_template.h
- Add description in the docs
- Add examples
- Update the changelog

如果您创建了一个很酷的小部件，或者为 LVGL 添加了有用的功能，请随时为它打开一个新的 PR。我们在 lvgl/src/extra 文件夹中收集可选功能（又名插件），因此如果您有兴趣添加新功能，请使用这个文件夹。README 文件描述了贡献的基本规则，并列出了一些想法。

有关更多想法，请查看我们的*Roadmap* 页面。如果您对其中任何一个感兴趣，请随时分享您的意见和/或参与实施。

其他（仍然）不在路线图中的功能列在论坛的 功能请求 类别中。

添加新功能时，还需要更新以下内容：

- 更新 lv_conf_template.h
- 在 docs 中添加说明
- 添加示例
- 更新 变更日志

Become a maintainer（成为维护者）

If you want to become part of the core development team, you can become a maintainer of a repository.

By becoming a maintainer:

- You get write access to that repo:
  - Add code directly without sending a pull request
  - Accept pull requests
  - Close/reopen/edit issues
- Your input has higher impact when we are making decisions

You can become a maintainer by invitation, however the following conditions need to met

1. Have > 50 replies in the Forum. You can look at your stats here
2. Send > 5 non-trivial pull requests to the repo where you would like to be a maintainer

If you are interested, just send a message (e.g. from the Forum) to the current maintainers of the repository. They will check if the prerequisites are met. Note that meeting the prerequisites is not a guarantee of acceptance, i.e. if the conditions are met you won't automatically become a maintainer. It's up to the current maintainers to make the decision.

如果您想成为核心开发团队的一员，您可以成为存储库的维护者。

通过成为维护者：

- 您可以对该存储库进行写访问：
  - 无需发送拉取请求，直接添加代码
  - 接受拉取请求
  - 关闭/重新打开/编辑问题
- 当我们做出决定时，您的意见会产生更大的影响

您可以通过邀请成为维护者，但需要满足以下条件

1. 论坛回复超过 50 条。你可以查看你的统计数据这里
2. 向您希望成为维护者的存储库发送 > 5 个重要的拉取请求

如果您有兴趣，只需向存储库的当前维护者发送一条消息（例如来自论坛）。他们将检查是否满足先决条件。请注意，满足先决条件并不能保证被接受，即如果满足条件，您将不会自动成为维护者。由当前的维护者做出决定。

Move your project repository under LVGL organization（将您的项目库移到 LVGL 组织下）

Besides the core `lvgl` repository there are other repos for ports to development boards, IDEs or other environment. If you ported LVGL to a new platform we can host it under the LVGL organization among the other repos.

This way your project will become part of the whole LVGL project and can get more visibility. If you are interested in this opportunity just open an issue in lvgl repo and tell what you have!

If we agree that your port fit well into the LVGL organization, we will open a repository for your project where you will have admin rights.

To make this concept sustainable there a few rules to follow:

- You need to add a README to your repo.
- We expect to maintain the repo to some extent:
  - Follow at least the major versions of LVGL
  - Respond to the issues (in a reasonable time)
- If there is no activity in a repo for 1 year it will be archived

除了核心 `lvgl` 存储库，还有其他存储库用于开发板、IDE 或其他环境的端口。如果您将 LVGL 移植到一个新平台，我们可以将它托管在 LVGL 组织下的其他存储库中。

这样您的项目将成为整个 LVGL 项目的一部分，并可以获得更多的可见性。如果您对这个机会感兴趣，只需打开一个 问题在 lvgl repo 并告诉您有什么！

如果我们同意您的移植非常适合 LVGL 组织，我们将为您的项目打开一个存储库，您将在其中拥有管理员权限。

为了使这个概念可持续，需要遵循一些规则：

- 你需要在你的 repo 中添加一个 README。
- 我们希望在一定程度上维持回购：
  - 至少遵循 LVGL 的主要版本
  - 回应问题（在合理的时间内）
- 如果 1 年内没有任何活动，它将被存档

2.11 Changelog

2.11.1 v8.2.0 31 January 2022

Overview

Among many fixes and minor updates these are the most important features in v8.2.0:

- Abstract render layer to make it easier to attack external draw engines
- Add `LV_FLAD_OVERFLOW_VISIBLE`. If enabled the children of an object won't be clipped to the boundary of the object
- Add ffmpeg decoder support to play videos and open a wide variety of image formats
- Add font fallback support
- Add gradient dithering support
- Add "monkey test"
- Add cmsis-pack support
- Add Grid navigation (`lv_gridnav`)

The GPU support for NXP microcontrollers is still not updated to the new draw architecture. See #3052

Breaking Changes

- :warning: feat(fs): add caching option for lv_fs-read `2979`
- :warning: feat(span): lv_spangroup_get_expand_width() adds a parameter `2968`
- :warning: arch(draw): allow replacing the draw engine `db53ea9`

Architectural

- arch(draw): separate SW renderer to allow replacing it `2803`
- arch: merge lv_demos `5414652`
- arch(sdl): migrated to use new backend architecture `2840`
- arch(env): move rt-thread into env_support folder `3025`
- arch(env): arch(env): move the cmake folder into the env_support folder `773d50f`
- arch(env): move the zephyr folder into the env_support folder `4bd1e7e`

New Features

- feat(cmsis-pack): prepare for lvgl v8.2.0 release 3062
- feat(gridnav): add lv_gridnav 2911
- feat: update the cmsis-pack to 0.8.3 3021
- feat(sdl): support rounded images 3012
- feat(cmsis-pack): add cmsis-pack support 2993
- feat(event): add preprocessing and stop bubbling features for events 3003
- feat(draw): add gradient dithering support 2872
- feat(symbols): add guards to LV_SYMBOL_* to allow redefining them 2973
- feat(obj): subdivide LV_OBJ_FLAG_SCROLL_CHAIN into ...CHAIN_HOR and ...CHAIN_VER 2961
- feat(draw): add draw_bg callback to draw_ctx #2934 2935
- feat(docs): add Chinese readme 2919
- feat(txt): add used_width parameter to _lv_txt_get_next_line() 2898
- feat(others) add monkey test 2885
- feat(rlottie): add animation control options 2857
- feat(lv_hal_indev): add missing lv_indev_delete() 2854
- feat(freetype): optimize memory allocation 2849
- feat(Kconfig): add FreeType config 2846
- feat(widgets): add menu widget 2603
- feat(refr): add reset function for FPS statistics 2832
- feat(Kconfig): add monitor position configuration 2834
- feat(examples) add micropython versions of the external library examples 2762
- feat(freetype): support bold and italic 2824
- feat(font) add fallback support and mem. font load option to FreeType 2796
- feat(lib) add ffmpeg video and image decoder 2805
- feat(obj): add LV_OBJ_FLAG_OVERFLOW_VISIBLE e7ac0e4
- feat(scrollbar): add more control over scrollbar paddings 4197b2f
- feat(dropdown): keep the list on open/close for simpler styling 9d3134b
- feat(qrcode) use destructor instead of lv_qrcode_delete() 318edd8
- feat(disp) allow decouplig the disp_refr timer 85cc84a

- feat(obj): add lv_obj_get_event_user_data() 53ececc
- feat(obj) add LV_OBJ_FLAG_SCROLL_WITH_ARROW 70327bd
- feat(slider): consider ext_click_area on the knob with LV_OBJ_FLAG_ADV_HITTEST 9d3fb41

Performance

- perf(sdl): optimize the use of SDL_RenderSetClipRect 2941
- perf(color): add faster lv_color_hex function 2864

Fixes

- fix(micropython) update examples for new API 3059
- fix: increase default value of LV_MEM_SIZE for lv_demo_widgets #3057 3058
- fix(cmsis-pack): fix issue #3032 3056
- fix(porting): add missing function prototypes 3054
- fix(kconfig): add missing default values 3050
- fix(canvas): force canvas to use sw draw 3045
- fix(rt-thread): use ARCH_CPU_BIG_ENDIAN to replace RT_USING_BIG_ENDIAN 3044
- fix(gradient): general cleanup and fix for alignment issues 3036
- fix(draw): rendering issues for vertical gradient with and without dithering 3034
- fix uninitialized variable 3033
- fix(lru): lower dependency for standard C functions 3024
- fix(env_support): move cmsis-pack to env_support folder 3026
- fix(doc): full covering opacity is 255, not 256 3022
- fix uninitialized variables 3023
- fix various issues for esp32 3007
- fix(sdl): fix clipped image drawing 2992
- fix(draw): missed bg_color renaming in the draw function 3002
- fix(porting): fix typo and an unmatched prototype 2998
- fix(conf) add missing LV_LOG_LEVEL default definition 2996
- fix(refr): crash if full_refresh = 1 2999
- fix(Kconfig): adapt to lvgl's built-in demos 2989

- fix(Makefile): compilation errors 2944
- fix(rlottie): fix variable name 2971
- fix(group): in lv_group_del() remove group from indev (lvgl#2963) 2964
- fix(obj): old parent's scroll is not updated in lv_obj_set_parent() 2965
- fix(fatfs) add missing cast 2969
- fix(snapshot) fix memory leak 2970
- fix(examples) move event callback registration outside loop in `lv_example_event_3` 2959
- fix(canvas): off by one error in size check in lv_canvas_copy_buf 2950
- fix(indev) add braces to avoid compiler warning 2947
- fix: fix parameter order in function prototypes 2929
- fix(style):add const qualifier for lv_style_get_prop() 2933
- fix(dropdown): in lv_dropdpwn_get_selected_str handle if there are no options 2925
- fix: lv_deinit/lv_init crash or hang 2910
- fix(rt-thread): improve the structure 2912
- fix: removed string format warnings for int32_t and uint32_t 2924
- fix(lv_fs_win32): add missing include of <stdio.h> 2918
- fix: use unsigned integer literal for bit shifing. 2888
- chore(lottie) move rlottie_capi.h to lv_rlottie.c 2902
- fix(qrcodegen) add brackets around assert calls 2897
- fix(list) guard image creation with LV_USE_IMG 2881
- fix(snapshot): make fake display size big enough to avoid align issue. 2883
- fix(sdl) correct makefile 2884
- fix(draw): fix set_px_cb memory write overflow crash. 2882
- fix(freetype): fix memset error 2877
- fix(span): fix align and break word 2861
- fix(refr): swap buffers only on the last area with direct mode 2867
- fix(arc) free memory when drawing full-circle arc 2869
- fix(indev): update lv_indev_drv_update to free the read_timer 2850
- fix(draw): fix memory access out of bounds when using blend subtract 2860
- fix(chart) add lv_chart_refresh() to the functions which modify the data 2841

- fix(conf) mismatched macro judgment 2843
- fix(ffmpeg): when disabled LV_FFMPEG_AV_DUMP_FORMAT makes av_log quiet 2838
- fix(rt-thread): fix a bug of log 2811
- fix(log): to allow printf and custom_print_cb to work at same time 2837
- fix(keyboard): add missing functions 2835
- fix(checkbox) remove unnecessary events 2829
- fix(qrcode): replace memcpy() with lv_memcpy() and delete useless macros 2827
- fix(font) improve builtin font source files generation process 2825
- fix(CMake) split CMakeLists.txt, add options, includes and dependencies 2753
- fix(obj): make lv_obj_fade_in/out use the current opa as start value 2819
- fix(qrcode):minimize margins as much as possible 2804
- fix(scripts): switch all scripts to python3 2820
- fix(event): event_send_core crash in special case. 2807
- fix(Kconfig) remove duplicate LV_BUILD_EXAMPLES configuration 2813
- fix(obj): in obj event use the current target instead of target 2785
- fix(draw_label): radius Mask doesn't work in Specific condition 2784
- fix(draw_mask): will crash if get_width/height < 0 2793
- fix(theme) make the basic theme really basic a369f18
- fix(arc): fix knob invalidation 345f688
- fix(theme): add arc, spinner and colorwheel to basic theme adc218a
- fix(conf) define LV_LOG_TRACE_... to 0 in lv_conf_internal.h to avoid warnings 305284c
- fix(draw): consider opa and clip corner on bg_img d51aea4
- fix(draw): add grad_cache_mem to GC_ROOTs 138db9c
- fix(bar, slider): fix shadow drawing on short indicators 364ca3c
- fix(theme): fix theme initialization issue introduced in 6e0072479 d231644
- fix(draw): add lv_draw_sw_bg 49642d3
- fix(draw) border_draw crash is special case 075831a
- fix(theme): fix crash in lv_theme_basic_init ca5f04c
- fix(draw): fix indexed image drawing 5a0dbcc
- fix(roller): clip overflowing text 5709528

- fix(align) fix LV_SIZE_CONTENT size calculation with not LEFT or TOP alignment 9c67642
- fix(draw): futher bg_img draw fixes 81bfb76
- fix(btnmatrix): keep the selected button even on release d47cd1d
- fix(sw): make knob size calculation more intuitive 5ec532d
- fix(switch): make knob height calculation similar to slider 0921dfc
- fix(span): explicitly set span->txt to the return value of lv_mem_realloc(#3005) a9a6cb8
- fix(example): update LVGL_Arduino.ino d79283c
- fix(draw) simplify how outline_pad is compnesated 81d8be1
- fix(obj) make LV_OBJ_FLAG_SCROLL_CHAIN part of the enum instead of define f8d8856
- fix(label): dot not add dots if the label height > 1 font line height 4d61f38
- fix(event): crash if an object was deleted in an event 9810920
- fix(build) fix sdl build with make 43729d1
- fix(config): fix anonymus choice 71c739c
- chore(docs): fix lv_list_add_text a5fbf22
- fix(png) check png magic number to be sure it's a png image 1092550
- fix(btnmatrix): fix crash if an empty btnmatrix is pressed 2392f58
- fix(mem/perf monitor): fix issue introduced in #2910 0788d91
- fix(layout) fix layout recalculation trigger in lv_obj_add/clear_fleg ee65410
- fix(obj) fix lv_obj_fade_in 4931384
- fix(draw): fix clipping children to parent 5c98ac8
- fix: remove symlinks to be accepted as an Ardunio library 6701d36
- chore: fix typos in FATFS config 74091c4
- fix(refr): fix missed buffer switch in double full-screen buffer + direct_mode 731ef5a
- chore(qrcode): fix warnings e9d7080
- docs(event): tell to not adjust widgets in draw events 933d67f
- fix(table, chart): fix memory leaks 8d52de1
- fix(event): handle object deletion in indev->fedback_cb bfc8edf
- fix(roller): snap on press lost fa9340c
- fix(dropdown) be sure the list is the top object on the screen cb7fc2b
- fix(img) fix invalidation issue on transformations d5ede0e

- fix(obj) fix comments of lv_obj_set_pos/x/y b9a5078

Examples

- example: add non-null judgment to lv_example_obj_2 2799
- example(table): fix text alignment b03dc9c

Docs

- docs(demos) update information to reflect new layout 3029
- docs(porting): remove duplicated content 2984
- docs(display) fix typo 2946
- docs(get-started) add introduction for Tasmota and Berry 2874
- docs fix spelling, parameter descriptions, comments, etc 2865
- docs: spelling fixes 2828
- docs(style) minor style fix 2818
- docs(porting/display) fix formatting 2812
- docs(roadmap) udpate 084439e
- docs(widgets) fix edit links 7ed1a56
- docs(contributing) update commit message format 1cd851f
- docs(porting): add more details about adding lvgl to your project 6ce7348
- docs(indev): add description about gestures 2719862
- docs(style): describe const styles 28ffae8
- docs(faq): add "LVGL doesn't start, nothing is drawn on the display" section 0388d92
- docs add demos 02a6614
- docs(fs): update fs interface description to the latest API 285e6b3
- docs(format) let wrap 4bf49a8
- docs(imgbtn) fix typo d792c5f
- docs(porting) clarify that displays must be registered before input devices 1c64b78
- docs(event) fix lv_event_get_original_target vs lv_event_get_current_target cdd5128
- docs(events) rename LV_EVENT_APPLY to LV_EVENT_READY (#2791) bf6837f
- docs(gpu): link style properties and boxing model 6266851

- docs(gesture): clarify gesture triggering with scrolling e3b43ee
- docs(contributing): remove the mentioning of the dev branch 00d4ef3
- docs(bar) fix default range eeee48b
- docs(event): tell to not adjust widgets in draw events 933d67f
- docs(switch) improve wording b4986ab
- docs(font) fix example to match v8 2f80896

CI and tests

- test(bar): add unit tests 2845
- test(switch): add initial unit test 2794
- test(demo) add tests for widget and stress demos 3bd6ad8
- test(dropdown) fix to pass again 918b3de
- test add support for using system heap 446b1eb
- ci remove formatting request workflow 6de89e4
- ci initial support for cross-architecture tests 7008770
- ci create handler for formatting requests 7af7849
- test(style) add test for gradient da8f345
- test(event) add test for #2886 51ef9c2
- ci add workflow to check code formatting a2b555e
- ci attempt to speed up cross tests 80408f7
- ci apply my updates to the verify-formatting action 02f02fa
- ci: add arduino linter action f79b00c
- ci update action be9722c
- ci more formatting action updates 1f6037c
- ci disable LeakSanitizer on dockerized tests c9e1927
- ci one last try at this for tonight dddafae
- ci try alternate checkout mechanism cb3de30
- test(style) fix compile error ba083df
- test(template) simplify _test_template.c b279f63
- ci force ccache to be saved every time a7c590f

- ci switch to codecov v2 `6b84155`
- ci more debugging for formatting action `2f8e4bc`
- ci inline apt-get commands `90e2b9f`
- ci(micropython) use ESP-IDF 4.4 `b34fe9e`
- ci add 5k stack limit `4122dda`
- ci force use of ccache in PATH `6de3fa8`
- ci add back stack usage check at 4 kilobytes `89135d6`
- ci temporarily disable stack usage check `1900c21`
- ci(cross) use python3 instead of python `df7eaa0`
- ci use specific version tag `59b4769`
- ci fix check style action `5bb3686`
- ci fix typo in formatting action `d1ccbf6`
- ci test formatting action `065d821`
- ci(micropython) switch to newer GCC action `1fa7257`
- ci(style) force color on diff to help highlight whitespace changes `04f47ea`
- ci(cross) install build-essential `772f219`
- ci force pushing to upstream branch `8277f78`
- ci ensure lvgl-bot is used to make commits `9fcf52a`

2.11.2 v8.1.0 10 November 2021

Overview

v8.1 is minor release so besides many fixes it contains a lot of new features too.

Some of the most important features are

- Built in support for SDL based GPU drawing
- Much faster circle drawing in the software renderer
- Several 3rd party libraries are merged directly into LVGL.
- Add LVGL as an RT-Thread and ESP32 component

Breaking Changes

- :warning: feat(calendar): add the header directly into the calendar widget 2e08f80

Architectural

- arch add small 3rd party libs to lvgl 2569

New Features

- feat(display) add direct_mode drawing mode 2460
- feat(conf): make LV_MEM_BUF_MAX_NUM configurable 2747
- feat(disp): add non-fullscreen display utilities 2724
- feat(rlottie) add LVGL-Rlottie interface as 3rd party lib 2700
- feat(rtthread): prepare for porting the device-driver of rt-thread 2719
- feat(fsdrv) add driver based on Win32 API 2701
- feat(span) indent supports percent for fix and break mode 2693
- feat(rt-thread): implement rt-thread sconscirpt 2674
- feat(lv_spinbox) support both right-to-left and left-to-right digit steps when clicking encoder button 2644
- feat add support for rt-thread RTOS 2660
- feat(disp): Enable rendering to display subsection 2583
- feat(keyboard): add user-defined modes 2651
- feat(event) add LV_EVENT_CHILD_CREATED/DELETED 2618
- feat(btnmatrix/keyboard): add option to show popovers on button press 2537
- feat(msgbox) add a content area for custom content 2561
- feat(tests): Include debug information to test builds 2568
- feat(drawing) hardware accelerated rendering by SDL2 2484
- feat(msgbox): omit title label unless needed 2539
- feat(msgbox): add function to get selected button index 2538
- feat(make) add lvgl interface target for micropython 2529
- feat(obj) add lv_obj_move_to_index(obj, index), renamed lv_obj_get_child_id(obj) to lv_obj_get_index(obj) 2514
- feat(obj) add lv_obj_swap() function 2461

- feat(mem) LV_MEM_POOL_ALLOC 2458
- feat(switch) add smooth animation when changing state 2442
- feat(anim) add interface for handling lv_anim user data. 2415
- feat(obj) add lv_is_initialized 2402
- feat(obj) Backport keypad and encoder scrolling from v7 `lv_page` to v8 `lv_obj` 2390
- feat(snapshot) add API to take snapshot for object 2353
- feat(anim) add anim timeline 2309
- feat(span) Add missing spangroup functions 2379
- feat(img) add img_size property 2284
- feat(calendar) improve MicroPython example 2366
- feat(spinbox) add function to set cursor to specific position 2314
- feat(timer) check if lv_tick_inc is called aa6641a
- feat(event, widgets) improve the paramter of LV_EVENT_DRAW_PART_BEGIN/END 88c4859
- feat(docs) improvements to examples 4b8c73a
- feat(obj) send LV_EVENT_DRAW_PART_BEGIN/END for MAIN and SCROLLBAR parts b203167
- feat(led) send LV_EVENT_DRAW_PART_BEGIN/END fcd4aa3
- feat(chart) send LV_EVENT_DRAW_PART_BEGIN/END before/after the division line drawing section. e0ae2aa
- feat(tests) upload coverage to codecov 4fff99d
- feat(conf) add better check for Kconfig default f8fe536
- feat(draw) add LV_BLEND_MODE_MULTIPLY cc78ef4
- feat(test) add assert for screenshot compare 2f7a005
- feat(event) pass the scroll aniamtion to LV_EVENT_SCROLL_BEGIN ca54ecf
- feat(obj) place the scrollbar to the left with RTL base dir. 906448e
- feat(log) allow overwriting LV_LOG_... macros 17b8a76
- feat(arc) add support to LV_OBJ_FLAG_ADV_HITTEST dfa4f5c
- feat(event) add LV_SCREEN_(UN)LOAD_START 7bae9e3
- feat(obj) add lv_obj_del_delayed() c6a2e15
- feat(docs) add view on GitHub link a716ac6
- feat(event) add LV_EVENT_SCREEN_LOADED/UNLOADED events ee5369e

- feat(textarea) remove the need of lv_textarea_set_align 56ebb1a
- feat(rt-thread): support LVGL projects with GCC/Keil(AC5)/Keil(AC6)/IAR 32d33fe
- feat(docs) lazy load individual examples as well 918d948
- feat: add LV_USE_MEM_PERF/MONITOR_POS acd0f4f
- feat(canvas) add lv_canvas_set_px_opa b3b3ffc
- feat(event) add lv_obj_remove_event_cb_with_user_data 4eddeb3
- feat(obj) add lv_obj_get_x/y_aligned 98bc1fe

Performance

- perf(draw) reimplement circle drawing algorithms 2374
- perf(anim_timeline) add lv_anim_timeline_stop() 2411
- perf(obj) remove lv_obj_get_child_cnt from cycle limit checks ebb9ce9
- perf(draw) reimplement rectangle drawing algorithms 5b3d3dc
- perf(draw) ignore masks if they don't affect the current draw area a842791
- perf(refresh) optimize where to wait for lv_disp_flush_ready with 2 buffers d0172f1
- perf(draw) speed up additive blending 3abe517

Fixes

- fix(bidi): add weak characters to the previous strong character's run 2777
- fix(draw_img): radius mask doesn't work in specific condition 2786
- fix(border_post): ignore bg_img_opa draw when draw border_post 2788
- fix(refresh) switch to portable format specifiers 2781
- fix(stm32) Mark unused variable in stm32 DMA2D driver 2782
- fix(conf): Make LV_COLOR_MIX_ROUND_OFS configurable 2766
- fix(misc): correct the comment and code style 2769
- fix(draw_map) use existing variables instead function calls 2776
- fix(draw_img): fix typos in API comments 2773
- fix(draw_img):radius Mask doesn't work in Specific condition 2775
- fix(proto) Remove redundant prototype declarations 2771
- fix(conf) better support bool option from Kconfign 2555

- fix(draw_border):draw error if radius == 0 and parent clip_corner == true 2764
- fix(msgbox) add declaration for lv_msgbox_content_class 2761
- fix(core) add L suffix to enums to ensure 16-bit compatibility 2760
- fix(anim): add lv_anim_get_playtime 2745
- fix(area) minor fixes 2749
- fix(mem): ALIGN_MASK should equal 0x3 on 32bit platform 2748
- fix(template) prototype error 2755
- fix(anim): remove time_orig from lv_anim_t 2744
- fix(draw_rect):bottom border lost if enable clip_corner 2742
- fix(anim) and improvement 2738
- fix(draw border):border draw error if border width > radius 2739
- fix(fsdrv): remove the seek call in fs_open 2736
- fix(fsdrv): skip the path format if LV_FS_xxx_PATH not defined 2726
- fix: mark unused variable with LV_UNUSED(xxx) instead of (void)xxx 2734
- fix(fsdrv): fix typo error in commit 752fba34f677ad73aee 2732
- fix(fsdrv): return error in case of the read/write failure 2729
- fix(refr) silence compiler warning due to integer type mismatch 2722
- fix(fs): fix the off-by-one error in the path function 2725
- fix(timer): remove the code duplication in lv_timer_exec 2708
- fix(async): remove the wrong comment from lv_async_call 2707
- fix(kconfig): change CONFIG_LV_THEME_DEFAULT_FONT to CONFIG_LV_FONT_DEFAULT 2703
- fix add MP support for LVGL 3rd party libraries 2666
- fix(png) memory leak for sjpg and use lv_mem_... in lv_png 2704
- fix(gif) unified whence and remove off_t 2690
- fix(rt-thread): include the rt-thread configuration header file 2692
- fix(rt-thread): fix the ci error 2691
- fix(fsdrv) minor fs issue 2682
- fix(hal) fix typos and wording in docs for lv_hal_indev.h 2685
- fix(hal tick): add precompile !LV_TICK_CUSTOM for global variables and lv_tick_inc() 2675
- fix(anim_timeline) avoid calling lv_anim_del(NULL, NULL) 2628

- fix(kconfig) sync Kconfig with the latest lv_conf_template.h 2662
- fix(log) reduce the stack usage in log function 2649
- fix(conf) make a better style alignment in lv_conf_internal.h 2652
- fix(span) eliminate warning in lv_get_snippet_cnt() 2659
- fix(config): remove the nonexistent Kconfig 2654
- fix(Kconfig): add LV_MEM_ADDR config 2653
- fix(log): replace printf with fwrite to save the stack size 2655
- fix typos 2634
- fix LV_FORMAT_ATTRIBUTE fix for gnu > 4.4 2631
- fix(meter) make lv_meter_indicator_type_t of type uint8_t 2632
- fix(span):crash if span->txt = "" 2616
- fix(disp) set default theme also for non-default displays 2596
- fix(label):LONG_DOT mode crash if text Utf-8 encode > 1 2591
- fix(example) in lv_example_scroll_3.py float_btn should only be created once 2602
- fix lv_deinit when LV_USE_GPU_SDL is enabled 2598
- fix add missing LV_ASSERT_OBJ checks 2575
- fix(lv_conf_internal_gen.py) formatting fixes on the generated file 2542
- fix(span) opa bug 2584
- fix(snapshot) snapshot is affected by parent's style because of wrong coords 2579
- fix(label):make draw area contain ext_draw_size 2587
- fix(btnmatrix): make ORed values work correctly with lv_btnmatrix_has_btn_ctrl 2571
- fix compiling of examples when cmake is used 2572
- fix(lv_textarea) fix crash while delete non-ascii character in pwd mode 2549
- fix(lv_log.h): remove the duplicated semicolon from LV_LOG_xxx 2544
- fix(zoom) multiplication overflow on 16-bit platforms 2536
- fix(printf) use __has_include for more accurate limits information 2532
- fix(font) add assert in lv_font.c if the font is NULL 2533
- fix(lv_types.h): remove c/c++ compiler version check 2525
- fix(lv_utils.c): remove the unneeded header inclusion 2526
- fix(Kconfig) fix the comment in LV_THEME_DEFAULT_DARK 2524

- fix(sprintf) add format string for rp2 port 2512
- fix(span) fix some bugs (overflow,decor,align) 2518
- fix(color) Bad cast in lv_color_mix() caused UB with 16bpp or less 2509
- fix(imgbtn) displayed incorrect when the coordinate is negative 2501
- fix(event) be sure to move all elements in copy "lv_obj_remove_event_cb" 2492
- fix(draw) use correct pointer in lv_draw_mask assertion 2483
- feat(mem) LV_MEM_POOL_ALLOC 2458
- fix(cmake) require 'main' for Micropython 2444
- fix(docs) add static keyword to driver declaration 2452
- fix(build) remove main component dependency 2420
- fix circle drawing algorithms 2413
- fix(docs) wrong spelling of words in pictures 2409
- fix(chart) fixed point-following cursor during vertical scroll in charts 2400
- fix(chart) fixed cursor positioning with large Y rescaling without LV_USE_LARGE_COORD 2399
- fix(grid.h) typos 2395
- fix(anim_timeline) heap use after free 2394
- fix(snapshot) add missing import on MicroPython example 2389
- fix(disp) Fix assert failure in lv_disp_remove 2382
- fix(span) modify the underline position 2376
- fix(color) remove extraneous _LV_COLOR_MAKE_TYPE_HELPER 2372
- fix(spinner) should not be clickable 2373
- fix(workflow) silence SDL warning for MicroPython 2367
- fix (span) fill LV_EVENT_GET_SELF_SIZE 2360
- fix(workflow) change MicroPython workflow to use master 2358
- fix(disp) fix memory leak in lv_disp_remove 2355
- fix(lv_obj.h)typos 2350
- fix(obj) delete useless type conversion 2343
- fix(lv_obj_scroll.h) typos 2345
- fix(txt) enhance the function of break_chars 2327
- fix(vglite): update for v8 e3e3eea

- fix(widgets) use lv_obj_class for all the widgets 3fb8baf
- fix(refr) reduce the nesting level in lv_refr_area 2df1282
- fix(pxp): update for v8 8a2a4a1
- fix(obj) move clean ups from lv_obj_del to lv_obj_destructor b063937
- fix (draw) fix arc bg image drawing with full arcs c3b6c6d
- fix(pxp): update RTOS macro for SDK 2.10 00c3eb1
- fix(textarea) style update in oneline mode + improve sroll to cursor 60d9a5e
- feat(led) send LV_EVENT_DRAW_PART_BEGIN/END fcd4aa3
- fix warnigs introduced by 3fb8baf5 e302403
- fix(roller) fix partial redraw of the selected area 6bc40f8
- fix(flex) fix layout update and invalidation issues 5bd82b0
- fix(indev) focus on objects on release instead of press 76a8293
- fix tests 449952e
- fix(dropdown) forget the selected option on encoder longpress e66b935
- fix(obj) improve how the focusing indev is determined a04f2de
- fix(workflow) speed up MicroPython workflow 38ad5d5
- fix(test) do not including anything in test files when not running tests 9043860
- fix tests 36b9db3
- fix(scroll) fire LV_EVENT_SCROLL_BEGIN in the same spot for both axes b158932
- fix(btnmatrix) fix button invalidation on focus change 77cedfa
- fix(tlsf) do not use <assert.h> c9745b9
- fix(template) include lvgl.h in lv_port_*_template.c files 0ae15bd
- fix(docs) add margin for example description b5f632e
- fix(imgbtn) use the correct src in LV_EVENT_GET_SELF_SIZE 04c515a
- fix(color) remove extraneous cast for 8-bit color 157534c
- fix(workflow) use same Unix port variant for MicroPython submodules ac68b10
- fix(README) improve grammar de81889
- fix(printf) skip defining attribute if pycparser is used ee9bbea
- fix(README) spelling correction 41869f2
- fix(color) overflow with 16 bit color depth fe6d8d7

- fix(docs) consider an example to be visible over a wider area `145a0fa`
- fix(codecov) disable uploading coverage for pull requests `27d88de`
- fix(arc) disable LV_OBJ_FLAG_SCROLL_CHAIN by default `f172eb3`
- fix(template) updpate lv_objx_template to v8 `38bb8af`
- fix(align) avoid circular references with LV_SIZE_CONTENT `038b781`
- fix(draw) with additive blending with 32 bit color depth `786db2a`
- fix(arc) fix arc invalidation again `5ced080`
- fix(align) fix lv_obj_align_to `93b38e9`
- fix(scroll) keep the scroll position on object deleted `52edbb4`
- fix(dropdown) handle LV_KEY_ENTER `8a50edd`
- fix various minor warnings `924bc75`
- fix(textarea) various cursor darwing fixes `273a0eb`
- fix(label) consider base dir lv_label_get_letter_pos in special cases `6df5122`
- fix(imgbtn) add lv_imgbtn_set_state `26e15fa`
- fix(printf) add (int) casts to log messages to avoid warnings on %d `d9d3f27`
- fix(test) silence make `7610d38`
- fix(test) silence make `37fd9d8`
- fix(calendar) update the MP example `0bab4a7`
- fix(scroll) fix scroll_area_into_view with objects larger than the parent `5240fdd`
- fix(msgbox) handle NULL btn map paramter `769c4a3`
- fix (scroll) do not send unnecessary scroll end events `3ce5226`
- fix(obj_pos) consider all alignements in contnt size calculation but only if x and y = 0 `5b27ebb`
- fix(img decoder) add error handling if the dsc->data = NULL `d0c1c67`
- fix(txt): skip basic arabic vowel characters when processing conjunction `5b54800`
- fix(typo) rename LV_OBJ_FLAG_SNAPABLE to LV_OBJ_FLAG_SNAPPABLE `e697807`
- fix(lv_printf.h): to eliminate the errors in Keil and IAR `f6d7dc7`
- fix(draw) fix horizontal gradient drawing `4c034e5`
- fix(dropdown) use LV_EVENT_READY/CANCEL on list open/close `4dd1d56`
- fix(table) clip overflowing content `8c15933`
- fix(test) add #if guard to exclude test related files from the build `c12a22e`

- fix(test) add #if guard to exclude test related files from the build `fc364a4`
- fix(freetype) fix underline calculation `76c8ee6`
- fix(style) refresh ext. draw pad for padding and bg img `37a5d0c`
- fix(draw) underflow in subpixel font drawing `6d5ac70`
- fix(scrollbar) hide the scrollbar if the scrollble flag is removed `188a946`
- fix(color): minor fixes(#2767) `a4978d0`
- fix(group) skip object if an of the parents is hidden `5799c10`
- fix(obj) fix size invalidation issue on padding change `33ba722`
- fix(label) do not bidi process text in lv_label_ins_text `e95efc1`
- fix(refr) set disp_drv->draw_buf->flushing_last correctly with sw rotation `c514bdd`
- fix(draw) fix drawing small arcs `8081599`
- fix(chart) invalidation with LV_CHART_UPDATE_MODE_SHIFT `d61617c`
- fix(build) fix micropython build error `54338f6`
- fix(draw) fix border width of simple (radius=0, no masking) borders `20f1867`
- fix(calendar) fix calculation today and highlighted day `8f0b5ab`
- fix(style) initialize colors to black instead of zero `524f8dd`
- fix(sjpg) remove unnecessary typedefs `c2d93f7`
- fix(label) fix clipped italic letters `2efa6dc`
- fix(draw) shadow drawing with large shadow width `f810265`
- fix(dropdown) add missing invalidations `33b5d4a`
- fix(dropdown) adjust the handling of keys sent to the dropdown `e41c507`
- fix(disp) be sure the pending scr load animation is finished in lv_scr_load_anim `eb6ae52`
- fix(color) fox color premult precision with 16 bit color depth `f334226`
- fix(obj_pos) save x,y even if the object is on a layout `a9b660c`
- fix(scrollbar) hide the scrollbar if the scrollable flag is removed `d9c6ad0`
- fix(dropdown) fix list position with RTL base direction `79edb37`
- fix(obj) fix lv_obj_align_to with RTL base direction `531afcc`
- fix(chart) fix sending LV_EVENT_DRAW_PART_BEGIN/END for the cursor `34b8cd9`
- fix(arduino) fix the prototype of my_touchpad_read in the LVGL_Arduino.ino `1a62f7a`
- fix(checkbox) consider the bg border when positioning the indicator `a39dac9`

- fix(dropdown) send LV_EVENT_VALUE_CHANGED to allow styling of the list dae7039
- fix(group) fix infinite loop bdce0bc
- fix(keyboard) use LVGL heap functions instead of POSIX b20a706
- fix(blend) fix green channel with additive blending 78158f0
- fix(btnmatrix) do not show pressed, focused or focus key states on disabled buttons 3df2a74
- fix(font) handle the last pixel of the glyphs in font loader correctly fa98989
- fix(table) fix an off-by-one issue in self size calculation ea2545a
- fix shadowed variable e209260
- fix shadowed variable df60018
- fix(chart) be sure the chart doesn't remain scrolled out on zoom out ad5b1bd
- fix(docs) commit to meta repo as lvgl-bot instead of actual commit author f0e8549
- fix(table) invalidate the table on cell value change cb3692e
- fix(group) allow refocusing objects 1520208
- fix(tabview) fix with left and right tabs 17c5744
- fix(msgbox) create modals on top layer instead of act screen 5cf6303
- fix(theme) show disabled state on buttons of btnmatrix, msgbox and kayboard 0be582b
- fix(label) update lv_label_get_letter_pos to work with LV_BASE_DIR_AUTO too 580e05a
- fix(label) fix in lv_label_get_letter_pos with when pos==line_start 58f3f56
- fix(gif) replace printf statement with LVGL logging 56f62b8
- fix(docs) add fsdrv back 64527a5
- fix(table) remove unnecessary invalidation on pressing 6f90f9c
- fix(chart) draw line chart indicator (bullet) fba37a3
- fix(anim) return the first anim if exec_cb is NULL in lv_anim_get() fb7ea10
- fix(label) fix lv_label_get_letter_on with BIDI enabled 192419e
- fix(checkbox) add missing invalidations bb39e9d
- fix(draw) fix gradient calculation of the rectangle is clipped 13e3470
- fix(chart) fix typo in 655f42b8 6118d63
- fix(example) fix lv_example_chart_2 89081c2
- fix(calendar) fix the position calculation today ad05e19
- fix(tick) minor optimization on lv_tick_inc call test b4305df

- fix(docs) use let instead of const for variable which gets changed 3cf5751
- fix(theme) fix the switch style in the default theme 0c0dc8e
- fix(tlsf) undef printf before define-ing it cc935b8
- fix(msgbox) prevent the buttons being wider than the msbgox 73e036b
- fix(chart) don't draw series lines with < 1 points 655f42b
- fix(tests) remove src/test_runners when cleaning 6726b0f
- fix(label) remove duplicated lv_obj_refresh_self_size a070ecf
- fix(colorwheel) disable LV_OBJ_FLAG_SCROLL_CHAIN by default 48d1c29
- fix(obj) do not set the child's position in lv_obj_set_parent d89a5fb
- feat: add LV_USE_MEM_PERF/MONITOR_POS acd0f4f
- fix(scroll) in scroll to view functions respect disabled LV_OBJ_FLAG_SCROLLABLE 9318e02
- fix(flex) remove unused variable 747b6a2
- feat(canvas) add lv_canvas_set_px_opa b3b3ffc
- fix(textarea) allow using cursor with not full bg_opa c9d3965
- fix(txt) _lv_txt_get_next_line return 0 on empty texts 82f3fbc
- fix(btnmatrix) always update row_cnt 86012ae
- fix(scroll) minor fixes on obj scroll handling a4128a8
- fix(table) consider border width for cell positions f2987b6
- fix(log) be sure LV_LOG_... is not empty if logs are disabled 47734c4
- fix(arc) fix LV_ARC_MODE_REVERSE df3b969
- fix(obj) in lv_obj_move_to_index() do not send LV_EVENT_CHILD_CHANGED on all changed child 32e8276
- feat(event) add lv_obj_remove_event_cb_with_user_data 4eddeb3
- fix(draw) fix shadow drawing with radius=0 4250e3c
- fix(msgbox) directly store the pointer of all children eb5eaa3
- fix(draw) use the filtered colors in lv_obj_init_draw_xxx_dsc() functions 78725f2
- fix(arc) fix full arc invalidation 98b9ce5
- chore(led) expose LV_LED_BRIGHT_MIN/MAX in led.h 3f18b23
- fix(group) keep the focused object in lv_group_swap_obj a997147
- fix(obj) swap objects in the group too in lv_obj_swap() 52c7558
- fix(theme) use opacity on button's shadow in the default theme c5342e9

- fix(win) enable clip_corner and border_post by default 493ace3
- fix(draw) fix rectangle drawing with clip_corner enabled 01237da
- fix(arc) fix other invalidation issues b0a7337
- feat(obj) add lv_obj_get_x/y_aligned 98bc1fe
- fix(calendar) fix incorrect highlight of today adbac52
- fix(arc, meter) fix invalidation in special cases 0f14f49
- fix(canvas) invalidate the image on delete a1b362c
- fix(msgbox) return the correct pointer from lv_msgbox_get_text 50ea6fb
- fix(bidi) fix the handling of LV_BASE_DIR_AUTO in several widgets 7672847
- fix(build) remove main component dependency (#2420) f2c2393
- fix(meter) fix inner mask usage c28c146
- fix(log) fix warning for empty log macros 4dba8df
- fix(theme) improve button focus of keyboard 2504b7e
- fix(tabview) send LV_EVENT_VALUE_CHANGED only once 933d282
- fix(obj style) fix children reposition if the parent's padding changes. 57cf661
- fix(template) update indev template for v8 d8a3d3d
- fix(obj) detecting which indev sent LV_EVENT_FOCUS f03d4b8
- fix(roller) adjust the size of the selected area correctly 01d1c87
- fix(imgbtn) consider width==LV_SIZE_CONTENT if only mid. img is set 7e49f48
- fix(flex) fix NULL pointer dereference 97ba12f
- fix(obj, switch) do not send LV_EVENT_VALUE_CHANGED twice 713b39e
- fix(coords) fix using large coordinates 428db94
- fix(chart) fix crash if no series are added c728b5c
- fix(meter) fix needle image invalidation 54d8e81
- fix(mem) add lv_ prefix to tlsf functions and types 0d52b59
- fix(pxp) change LV_COLOR_TRANSP to LV_COLOR_CHROMA_KEY to v8 compatibility 81f3068

Examples

- example(chart) add area chart example 2507
- example(anim) add demo to use cubic-bezier 2393
- feat(example) add lv_example_chart_9.py 2604
- feat(example) add lv_example_chart_8.py 2611
- feat(example) chart example to add gap between the old and new data 2565
- feat(example) add lv example list 2 2545
- feat(examples) add MicroPython version of lv_example_anim_3 and allow loading roller font dynamically 2412
- feat(examples) added MP version of second tabview example 2347
- fix(example):format codes 2731
- fix(example) minor fixes in lv_example_chart_2.py 2601
- feat(example) add text with gradient example 462fbcb
- fix(example_roller_3) mask free param bug 2553
- fix(examples) don't compile assets unless needed 2523
- fix(example) scroll example sqort types 2498
- fix(examples) join usage 2425
- fix(examples) add missing lv.PART.INDICATOR 2423
- fix(examples) use lv.grid_fr for MicroPython 2419
- fix(examples) remove symlinks 2406
- fix(examples) import 'u'-prefixed versions of modules 2365
- fix(examples) remove cast in MP scripts 2354
- fix(examples) fix MicroPython examples and run the examples with CI 2339
- fix(examples) align with renamed Micropython APIs 2338
- fix(examples) adjust canvas example for MicroPython API change 52d1c2e
- fix(example) revert test code 77e2c1f
- feat(example) add checkbox example for radio buttons d089b36
- feat(example) add text with gradient example 462fbcb
- fix(examples) exclude example animimg images if animimg is disabled 4d7d306
- fix(example) adjust the object sizes in lv_example_anim_timeline_1() 71a10e4
- fix(example) revert text code from lv_example_checkbox_2 28e9593

Docs

- docs: fix typo 2765
- docs(colorwheel) fix old API names 2643
- docs(display) fix typo 2624
- docs add static for lv_indev_drv_t 2605
- docs(animimg) add to extra widgets index and fix example 2610
- docs(animimg) Add missing animation image page 2609
- docs(group) remove reference to lv_cont which is gone in v8 2580
- docs(style) use correct API name for local styles 2550
- docs(all) Proofread, fix typos and add clarifications in confusing areas 2528
- docs(flex) update flex.md 2517
- docs more spelling fixes 2499
- docs fix typo: arae -> area 2488
- docs(readme) fix typo: hosing → hosting. 2477
- docs update company name and year 2476
- docs fix typos 2472
- docs(overview) fix typo 2465
- docs(bar) fix typos in widget examples 2463
- docs(overview) fix typo 2454
- docs(chart) typos 2427
- docs(layout) add internal padding paragraph to grid and flex layout p⋯2392
- docs(porting) fix indev example to remove v7 bool return 2381
- docs(README) fix broken references 2329
- docs(grid) typo fix 2310
- docs(color) language fixes 2302
- docs(lv_obj_style) update add_style and remove_style function headers 2287
- docs(contributing) add commit message format section 3668e54
- docs minor typo fixes 84c0086
- docs(arduino) update some outdated information 9a77102
- docs(keyboard) add note regarding event handler 255f729

- docs minor CSS fix acbb680
- docs minor CSS improvements 7f367d6
- docs(keyboard) change LV_KEYBOARD_MODE_NUM to LV_KEYBOARD_MODE_NUMBER 6e83d37
- docs(textarea) clarify the use of text selection bg_color 65673c0
- docs list all examples on one page 25acaf4
- docs(examples) add MicroPython examples 6f37c4f
- docs(filesystem) update to v8 7971ade
- docs(style) complete the description of style the properties 55e8846
- docs example list fixes cd600d1
- docs(style) complete the description of style the properties ff087da
- docs(README) update links, examples, and add services menu 3471bd1
- docs(color) update colors' docs 9056b5e
- docs update lv_fs.h, layer and align.png to v8 31ab062
- docs(color) minor fix ac8f453
- docs update changelog c386110
- docs(extra) add extra/README.md 8cd504d
- docs add lazy load to the iframes of the examples c49e830
- docs(os) add example and clarify some poinits d996453
- docs(rlottie) fix build error ce0b564
- docs include paths in libs f5f9562
- docs libs fixes 8e7bba6
- docs(obj) add comment lv_obj_get_x/y/width/height about postponed layout recalculation 533066e
- docs fix example list ed77ed1
- docs describe the options to include or skip lv_conf.h 174ef66
- docs(overview) spelling fixes d2efb8c
- docs(table) describe keypad/encoder navigation 749d1b3
- docs update CHANGELOG 0f8bc18
- docs(image) mention the frame_id parameter of lv_img_decoder_open 2433732
- docs(arduino) update how to use the examples 06962a5
- docs(rlottie): fix typo in commands ed9169c

- docs(indev, layer) update lv_obj_set_click() to lv_obj_add_flag() bcd99e8
- docs update version support table e6e98ab
- docs fix example list c6f99ad
- docs(examples) add <hr/> to better separate examples a1b59e3
- docs(checkbox) update the comment lv_checkbox_set_text_static 3e0ddd0
- docs(grid) fix missing article da0c97a
- docs(display) fix grammar in one spot 5dbea7d
- docs(style) fix typo in style property descriptions 4e3b860
- docs(flex) fix typo in flex grow section e5fafc4
- docs(indev) clarify purpose of `continue_reading` flag 706f81e
- docs(license) update company name and year 7c1eb00
- docs fix typo 8ab8064
- docs add libs to the main index 1a8fed5
- docs add btn_example.png 8731ef1
- docs(btnmatrix) fix typo with set_all/clear_all parameters 51a82a1

CI and tests

- ci(micropython) fix git fetch 2757
- test(txt) initial unit tests and general code cleanup/fixes 2623
- test add setUp and tearDown to test template 2648
- test(arc) add initial unit tests 2617
- ci(micropython) add ESP32 and STM32 tests 2629
- test(checkbox) add initial tests 2551
- test(ci) build and run tests in parallel. 2515
- ci(tests) run tests using ctest 2503
- ci(tests) add dependency on GNU parallel 2510
- ci(tests) use common script to install development prereqs 2504
- test convert Makefile to CMake 2495
- test Refactor unit test scripts. 2473
- test(font_loader) migrate the exisiting font loader test bc5b3be

- test add build test again, add dropdown test, integrate gcov and gvocr e35b1d0
- test(dropdown) add tess for keypad and encoder 4143b80
- test add keypad and encoder emulators e536bb6
- tests add mouse emulator 2ba810b
- tests add README b765643
- test add move tests to test_cases and test_runners directories e9e010a
- test fix CI build error c38cae2
- ci add config for 8bpp 3eacc59
- test move more source files to src folder 3672f87
- test update CI for the new tests a3898b9
- test cleant up report folder b9b4ba5
- test fix build error 61cda59
- test(font_loader) migrate the existing font loader test d6dbbaa
- test add move tests to test_cases and test_runners directories d2e735e
- test add 3rd party libs to all tests and also fix them 7a95fa9
- test(arc): add test case for adv_hittest e83df6f
- ci create check for lv_conf_internal.h 5d8285e
- test fix warning and docs build error d908f31
- ci(micropython) add rp2 port 1ab5c96
- test(dropdown) remove dummy test case 9fb98da
- ci(codecov) hide statuses on commits for now 0b7be77
- ci(docs) run apt-get update before installation f215174
- test fix LV_USE_LOG_LEVEL -> LV_LOG_LEVEL typo 80f0b09
- ci(micropython) add GCC problem matcher ab316a0
- test convert Makefile to CMake (#2495) 9c846ee

Others

- chore: replace (void)xxx with LV_UNUSED(xxx) 2779
- animation improvement 2743
- Improve LV_FORMAT_ATTRIBUTE usage 2673
- Fix typo in commands to build rlottie 2723
- del(.gitmodules): delete .gitmodules 2718
- lv_obj_draw_part_dsc_t.text_length added 2694
- expose LV_COLOR_DEPTH and LV_COLOR_16_SWAP in micropython 2679
- sync lvgl/lv_fs_if 2676
- build: always enable CMake install rule in default configuration 2636
- build: fix lib name in CMakeLists 2641
- build: remove use of 'project' keyword in CMakeLists 2640
- build add install rule to CMakeList.txt 2621
- Fixed row size calculation 2633
- arch add small 3rd party libs to lvgl 2569
- Kconfig: Add missing options 2597
- Espressif IDF component manager 2521
- chore(btnmatrix) removed unnecessary semicolon 2520
- Update README.md 2516
- Corrected a function name in obj.md 2511
- Simple spelling fixes 2496
- added lv_obj_move_up() and lv_obj_move_down() 2467
- Fix buf name error for "lv_port_disp_template.c" and optimize the arduino example 2475
- Fix two examples in the docs with new v8 api 2486
- kconfig: minor fix for default dark theme option 2426
- doc(table) update doc on cell merging 2397
- added example lv_example_anim_timeline_1.py 2387
- refactor(printf) add printf-like function attribute to _lv_txt_set_text_vfmt and lv_label_set_text_fmt 2332
- Update win.md 2352
- Nxp pxp vglite v8 dev 2313

- More Snapable --> Snappable replacements 2304
- Spelling and other language fixes to documentation 2293
- Update quick-overview.md 2295
- adding micropython examples 2286
- format run code-formtter.sh d67dd94
- Update ROADMAP.md 2b1ae3c
- Create .codecov.yml e53aa82
- refactor(examples) drop JS-specific code from header.py ef41450
- make test run on mseter and release/v8.* 227402a
- Update release.yml 0838f12
- refactor(examples) drop usys import from header.py ad1f91a
- Update ROADMAP.md a38fcf2
- Revert "feat(conf) add better check for Kconfig default" a5793c7
- remove temporary test file a958c29
- start to implement release/patch 1626a0c
- chore(indev) minor formatting 79ab3d2
- add basic patch release script 1c3ecf1
- chore(example) minor improvements on lv_example_list_2 bb6d6b7
- tool: add changelog_gen.sh to automatically generate changelog 6d95521
- update version numbers to v8.1.0-dev 8691611
- chore(test) improve prints ea8bed3
- chore(test) improve prints 0c4bca0
- chore: update lv_conf_internal.h 41c2dd1
- chore(format) lv_conf_template.h minor formatting 3c86d77
- chore(docs) always deploy master to docs/master as well 6d05692
- Update CHANGELOG.md 48fd73d
- Fix compile errors 6c956cc
- Update textarea.md 6d8799f
- chore(assert) add warning about higher memory usage if LV_USE_ASSERT_STYLE is enabled 33e4330
- Update page.html 9573bab

- chore(docs) force docs rebuild 4a0f413
- Fix typo error in color.md 572880c
- Update arc.md 2a9b9e6
- Update index.rst 9ce2c77
- chore(docs) minor formatting on example's GitHub link 75209e8
- chore(lv_conf_template) fix spelling mistake 9d134a9
- Update CHANGELOG.md 8472360
- chore(stale) disable on forks 93c1303
- Revert "fix(tests) remove src/test_runners when cleaning" ae15a1b
- style fix usage of clang-format directives 2122583
- Revert "fix(indev) focus on objects on release instead of press" f61b2ca

2.11.3 v8.0.2 (16.07.2021)

- fix(theme) improve button focus of keyboard
- fix(tabview) send LV_EVENT_VALUE_CHANGED only once
- fix(imgbtn) use the correct src in LV_EVENT_GET_SELF_SIZE
- fix(color) remove extraneous cast for 8-bit color
- fix(obj style) fix children reposition if the parent's padding changes.
- fix(color) remove extraneous _LV_COLOR_MAKE_TYPE_HELPER (#2372)
- fix(spinner) should not be clickable (#2373)
- fix(obj) improve how the focusing indev is determined
- fix(template) update indev template for v8
- fix(printf) skip defining attribute if pycparser is used
- refactor(printf) add printf-like function attribute to _lv_txt_set_text_vfmt and lv_label_set_text_fmt (#2332)
- fix(template) include lvgl.h in lv_port_*_template.c files
- fix(obj) detecting which indev sent LV_EVENT_FOCUS
- fix (span) fill LV_EVENT_GET_SELF_SIZE (#2360)
- fix(arc) disable LV_OBJ_FLAG_SCROLL_CHAIN by default
- fix (draw) fix arc bg image drawing with full arcs
- fix(disp) fix memory leak in lv_disp_remove (#2355)

- fix warnings introduced by 3fb8baf5
- fix(widgets) use lv_obj_class for all the widgets
- fix(obj) move clean ups from lv_obj_del to lv_obj_destructor
- fix(roller) fix partial redraw of the selected area
- fix(roller) adjust the size of the selected area correctly
- fix(obj) delete useless type conversion (#2343)
- fix(lv_obj_scroll.h) typos (#2345)
- fix(scroll) fire LV_EVENT_SCROLL_BEGIN in the same spot for both axes
- fix(btnmatrix) fix button invalidation on focus change
- fix(textarea) style update in oneline mode + improve scroll to cursor
- fix(tlsf) do not use <assert.h>
- fix(imgbtn) consider width==LV_SIZE_CONTENT if only mid. img is set
- fix(refr) reduce the nesting level in lv_refr_area
- fix(txt) enhance the function of break_chars (#2327)
- fix(pxp): update RTOS macro for SDK 2.10
- fix(vglite): update for v8
- fix(pxp): update for v8
- fix(flex) fix layout update and invalidation issues
- fix(flex) fix NULL pointer dereference
- fix(obj, switch) do not send LV_EVENT_VALUE_CHANGED twice
- fix(color) overflow with 16-bit color depth
- fix(coords) fix using large coordinates
- fix(chart) fix crash if no series are added
- fix(chart) invalidation with LV_CHART_UPDATE_MODE_SHIFT
- fix(align) fix lv_obj_align_to G
- fix(table) invalidate the table on cell value change
- fix(label) remove duplicated lv_obj_refresh_self_size
- fix(draw) underflow in subpixel font drawing
- fix (scroll) do not send unnecessary scroll end events

2.11.4 v8.0.1 (14.06.2021)

- docs(filesystem) update to v8 7971ade4
- fix(msgbox) create modals on top layer instead of act screen 5cf6303e
- fix(colorwheel) disable LV_OBJ_FLAG_SCROLL_CHAIN by default 48d1c292
- docs(grid) typo fix (#2310) 69d109d2
- fix(arduino) fix the prototype of my_touchpad_read in the LVGL_Arduino.ino 1a62f7a6
- fix(meter) fix needle image invalidation 54d8e817
- fix(mem) add lv_ prefix to tlsf functions and types 0d52b59c
- fix(calendar) fix the position calculation today ad05e196
- fix(typo) rename LV_OBJ_FLAG_SNAPABLE to LV_OBJ_FLAG_SNAPPABLE e697807c
- docs(color) language fixes (#2302) 07ecc9f1
- fix(tick) minor optimization on lv_tick_inc call test b4305df5
- Spelling and other language fixes to documentation (#2293) d0aaacaf
- fix(theme) show disabled state on buttons of btnmatrix, msgbox and keyboard 0be582b3
- fix(scroll) keep the scroll position on object deleted 52edbb46
- fix(msgbox) handle NULL btn map parameter 769c4a30
- fix(group) allow refocusing objects 1520208b
- docs(overview) spelling fixes d2efb8c6
- Merge branch 'master' of https://github.com/lvgl/lvgl 45960838
- feat(timer) check if lv_tick_inc is called aa6641a6
- feat(docs) add view on GitHub link a716ac6e
- fix(theme) fix the switch style in the default theme 0c0dc8ea
- docs fix typo 8ab80645
- Merge branch 'master' of https://github.com/lvgl/lvgl e796448f
- feat(event) pass the scroll animation to LV_EVENT_SCROLL_BEGIN ca54ecfe
- fix(tabview) fix with left and right tabs 17c57449
- chore(docs) force docs rebuild 4a0f4139
- chore(docs) always deploy master to docs/master as well 6d05692d
- fix(template) update lv_objx_template to v8 38bb8afc
- docs(extra) add extra/README.md 8cd504d5

- Update CHANGELOG.md 48fd73d2
- Update quick-overview.md (#2295) 5616471c
- fix(pxp) change LV_COLOR_TRANSP to LV_COLOR_CHROMA_KEY to v8 compatibility 81f3068d
- adding micropython examples (#2286) c60ed68e
- docs(color) minor fix ac8f4534
- fix(example) revert test code 77e2c1ff
- fix(draw) with additive blending with 32-bit color depth 786db2af
- docs(color) update colors' docs 9056b5ee
- Merge branch 'master' of https://github.com/lvgl/lvgl a711a1dd
- perf(refresh) optimize where to wait for lv_disp_flush_ready with 2 buffers d0172f14
- docs(lv_obj_style) update add_style and remove_style function headers (#2287) 60f7bcbf
- fix memory leak of spangroup (#2285) 33e0926a
- fix make lv_img_cache.h public because cache invalidation is public 38ebcd81
- Merge branch 'master' of https://github.com/lvgl/lvgl 2b292495
- fix(btnmatrix) fix focus event handling 3b58ef14
- Merge pull request #2280 from lvgl/dependabot/pip/docs/urllib3-1.26.5 a2f45b26
- fix(label) calculating the clip area 57e211cc
- chore(deps): bump urllib3 from 1.26.4 to 1.26.5 in /docs b2f77dfc
- fix(docs) add docs about the default group 29bfe604

2.11.5 v8.0.0 (01.06.2021)

v8.0 brings many new features like simplified and more powerful scrolling, new layouts inspired by CSS Flexbox and Grid, simplified and improved widgets, more powerful events, hookable drawing, and more.

v8 is a major change and therefore it's not backward compatible with v7.

Directory structure

- The `lv_` prefix is removed from the folder names
- The `docs` is moved to the `lvgl` repository
- The `examples` are moved to the `lvgl` repository
- Create an `src/extra` folder for complex widgets:

  - It makes the core LVGL leaner
  - In `extra` we can have a lot and specific widgets
  - Good place for contributions

Widget changes

- `lv_cont` removed, layout features are moved to `lv_obj`
- `lv_page` removed, scroll features are moved to `lv_obj`
- `lv_objmask` the same can be achieved by events
- `lv_meter` added as the union of `lv_linemeter` and `lv_gauge`
- `lv_span` new widget mimicking HTML `<span>`
- `lv_animing` new widget for simple slideshow animations
- + many minor changes and improvements

New scrolling

- Support ”elastic” scrolling when scrolled in
- Support scroll chaining among any objects types (not only `lv_pages`s)
- Remove `lv_drag`. Similar effect can be achieved by setting the position in `LV_EVENT_PRESSING`
- Add snapping
- Add snap stop to scroll max 1 snap point

New layouts

- CSS Grid-like layout support
- CSS Flexbox-like layout support

Styles

- Optimize and simplify styles
- State is saved in the object instead of the style property
- Object size and position can be set in styles too

Events

- Allow adding multiple events to an object
- A `user_data` can be attached to the added events

Driver changes

- `lv_disp_drv_t`, `lv_indev_drv_t`, `lv_fs_drv_t` needs to be `static`
- `...disp_buf...` is renamed to `draw_buf`. See an initialization example here.
- No partial update if two screen sized buffers are set
- `disp_drv->full_refresh = 1` makes always the whole display redraw.
- `hor_res` and `ver_res` need to be set in `disp_drv`
- `indev_read_cb` returns `void`. To indicate that there is more that to read set `data->continue_reading = 1` in the `read_cb`

Other changes

- Remove the copy parameter from create functions
- Simplified File system interface API
- Use a more generic inheritance
- The built-in themes are reworked
- `lv_obj_align` now saved the alignment and realigns the object automatically but can't be used to align to other than the parent
- `lv_obj_align_to` can align to an object but doesn't save the alignment
- `lv_pct(x)` can be used to set the size and position in percentage
- There are many other changes in widgets that are not detailed here. Please refer to the documentation of the widgets.

New release policy

- We will follow Release branches with GitLab flow
- Minor releases are expected in every 3-4 month
- `master` will always contain the latest changes

Migrating from v7 to v8

- First and foremost, create a new `lv_conf.h` based on `lv_conf_template.h`.
- To try the new version it's recommended to use a simulator project and see the examples.
- When migrating your project to v8
  - Update the drivers are described above
  - Update the styles
  - Update the events
  - Use the new layouts instead of `lv_cont` features
  - Use `lv_obj` instead of `lv_page`
  - See the changes in Colors
  - The other parts are mainly minor renames and refactoring. See the functions' documentation for descriptions.

2.11.6 v7.11.0 (16.03.2021)

New features

- Add better screen orientation management with software rotation support
- Decide text animation's direction based on base_dir (when using LV_USE_BIDI)

Bugfixes

- fix(gauge) fix needle invalidation
- fix(bar) correct symmetric handling for vertical sliders

2.11.7 v7.10.1 (16.02.2021)

Bugfixes

- fix(draw) overlap outline with background to prevent aliasing artifacts
- fix(indev) clear the indev's `act_obj` in `lv_indev_reset`
- fix(text) fix out of bounds read in `_lv_txt_get_width`
- fix(list) scroll list when button is focused using LV_KEY_NEXT/PREV
- fix(text) improve Arabic contextual analysis by adding hyphen processing and proper handling of lam-alef sequence
- fix(delete) delete animation after the children are deleted

- fix(gauge) consider paddings for needle images

2.11.8 v7.10.0 (02.02.2021)

New features

- feat(indev) allow input events to be passed to disabled objects
- feat(spinbox) add inline get_step function for MicroPython support

Bugfixes

- fix(btnmatrix) fix lv_btnmatrix_get_active_btn_text() when used in a group

2.11.9 v7.9.1 (19.01.2021)

Bugfixes

- fix(cpicker) fix division by zero
- fix(dropdown) fix selecting options after the last one
- fix(msgbox) use the animation time provided
- fix(gpu_nxp_pxp) fix incorrect define name
- fix(indev) don't leave edit mode if there is only one object in the group
- fix(draw_rect) fix draw pattern stack-use-after-scope error

2.11.10 v7.9.0 (05.01.2021)

New features

- feat(chart) add lv_chart_remove_series and lv_chart_hide_series
- feat(img_cache) allow disabling image caching
- calendar: make get_day_of_week() public
- Added support for Zephyr integration

Bugfixes

- fix(draw_rect) free buffer used for arabic processing
- fix(win) arabic process the title of the window
- fix(dropdown) arabic process the option in lv_dropdown_add_option
- fix(textarea) buffer overflow in password mode with UTF-8 characters
- fix(textarea) cursor position after hiding character in password mode
- fix(linemeter) draw critical lines with correct color
- fix(lv_conf_internal) be sure Kconfig defines are always uppercase
- fix(kconfig) handle disable sprintf float correctly.
- fix(layout) stop layout after recursion threshold is reached
- fix(gauge) fix redraw with image needle

2.11.11 v7.8.1 (15.12.2020)

Bugfixes

- fix(lv_scr_load_anim) fix when multiple screens are loaded at the same time with delay
- fix(page) fix LV_SCROLLBAR_MODE_DRAG

2.11.12 v7.8.0 (01.12.2020)

New features

- make DMA2D non blocking
- add unscii-16 built-in font
- add KConfig
- add lv_refr_get_fps_avg()

Bugfixes

- fix(btnmatrix) handle arabic texts in button matrices
- fix(indev) disabled object shouldn't absorb clicks but let the parent to be clicked
- fix(arabic) support processing again already processed texts with _lv_txt_ap_proc
- fix(textarea) support Arabic letter connections
- fix(dropdown) support Arabic letter connections
- fix(value_str) support Arabic letter connections in value string property
- fix(indev) in LV_INDEV_TYPE_BUTTON recognize 1 cycle long presses too
- fix(arc) make arc work with encoder
- fix(slider) adjusting the left knob too with encoder
- fix reference to LV_DRAW_BUF_MAX_NUM in lv_mem.c
- fix(polygon draw) join adjacent points if they are on the same coordinate
- fix(linemeter) fix invalidation when setting new value
- fix(table) add missing invalidation when changing cell type
- refactor(roller) rename LV_ROLLER_MODE_INIFINITE -> LV_ROLLER_MODE_INFINITE

2.11.13 v7.7.2 (17.11.2020)

Bugfixes

- fix(draw_triangle): fix polygon/triangle drawing when the order of points is counter-clockwise
- fix(btnmatrix): fix setting the same map with modified pointers
- fix(arc) fix and improve arc dragging
- label: Repair calculate back `dot` character logical error which cause infinite loop.
- fix(theme_material): remove the bottom border from tabview header
- fix(imgbtn) guess the closest available state with valid src
- fix(spinbox) update cursor position in lv_spinbox_set_step

2.11.14 v7.7.1 (03.11.2020)

Bugfixes

- Respect btnmatrix's `one_check` in `lv_btnmatrix_set_btn_ctrl`
- Gauge: make the needle images to use the styles from `LV_GAUGE_PART_PART`
- Group: fix in `lv_group_remove_obj` to handle deleting hidden objects correctly

2.11.15 v7.7.0 (20.10.2020)

New features

- Add PXP GPU support (for NXP MCUs)
- Add VG-Lite GPU support (for NXP MCUs)
- Allow max. 16 cell types for table
- Add `lv_table_set_text_fmt()`
- Use margin on calendar header to set distances and padding to the size of the header
- Add `text_sel_bg` style property

Bugfixes

- Theme update to support text selection background
- Fix imgbtn state change
- Support RTL in table (draw columns right to left)
- Support RTL in pretty layout (draw columns right to left)
- Skip objects in groups if they are in disabled state
- Fix dropdown selection with RTL basedirection
- Fix rectangle border drawing with large width
- Fix `lv_win_clean()`

2.11.16 v7.6.1 (06.10.2020)

Bugfixes

- Fix BIDI support in dropdown list
- Fix copying base dir in `lv_obj_create`
- Handle sub pixel rendering in font loader
- Fix transitions with style caching
- Fix click focus
- Fix imgbtn image switching with empty style
- Material theme: do not set the text font to allow easy global font change

2.11.17 v7.6.0 (22.09.2020)

New features

- Check whether any style property has changed on a state change to decide if any redraw is required

Bugfixes

- Fix selection of options with non-ASCII letters in dropdown list
- Fix font loader to support LV_FONT_FMT_TXT_LARGE

2.11.18 v7.5.0 (15.09.2020)

New features

- Add `clean_dcache_cb` and `lv_disp_clean_dcache` to enable users to use their own cache management function
- Add `gpu_wait_cb` to wait until the GPU is working. It allows to run CPU a wait only when the rendered data is needed.
- Add 10px and 8ox built in fonts

Bugfixes

- Fix unexpected DEFOCUS on lv_page when clicking to bg after the scrollable
- Fix `lv_obj_del` and `lv_obj_clean` if the children list changed during deletion.
- Adjust button matrix button width to include padding when spanning multiple units.
- Add rounding to btnmatrix line height calculation
- Add `decmopr_buf` to GC roots
- Fix division by zero in draw_pattern (lv_draw_rect.c) if the image or letter is not found
- Fix drawing images with 1 px height or width

2.11.19 v7.4.0 (01.09.2020)

The main new features of v7.4 are run-time font loading, style caching and arc knob with value setting by click.

New features

- Add `lv_font_load()` function - Loads a `lv_font_t` object from a binary font file
- Add `lv_font_free()` function - Frees the memory allocated by the `lv_font_load()` function
- Add style caching to reduce access time of properties with default value
- arc: add set value by click feature
- arc: add `LV_ARC_PART_KNOB` similarly to slider
- send gestures event if the object was dragged. User can check dragging with `lv_indev_is_dragging(lv_indev_act())` in the event function.

Bugfixes

- Fix color bleeding on border drawing
- Fix using 'LV_SCROLLBAR_UNHIDE' after 'LV_SCROLLBAR_ON'
- Fix cropping of last column/row if an image is zoomed
- Fix zooming and rotating mosaic images
- Fix deleting tabview with LEFT/RIGHT tab position
- Fix btnmatrix to not send event when CLICK_TRIG = true and the cursor slid from a pressed button
- Fix roller width if selected text is larger than the normal

2.11.20 v7.3.1 (18.08.2020)

Bugfixes

- Fix drawing value string twice
- Rename `lv_chart_clear_serie` to `lv_chart_clear_series` and `lv_obj_align_origo` to `lv_obj_align_mid`
- Add linemeter's mirror feature again
- Fix text decor (underline strikethrough) with older versions of font converter
- Fix setting local style property multiple times
- Add missing background drawing and radius handling to image button
- Allow adding extra label to list buttons
- Fix crash if `lv_table_set_col_cnt` is called before `lv_table_set_row_cnt` for the first time
- Fix overflow in large image transformations
- Limit extra button click area of button matrix's buttons. With large paddings it was counter-intuitive. (Gaps are mapped to button when clicked).
- Fix `lv_btnmatrix_set_one_check` not forcing exactly one button to be checked
- Fix color picker invalidation in rectangle mode
- Init disabled days to gray color in calendar

2.11.21 v7.3.0 (04.08.2020)

New features

- Add `lv_task_get_next`
- Add `lv_event_send_refresh`, `lv_event_send_refresh_recursive` to easily send `LV_EVENT_REFRESH` to object
- Add `lv_tabview_set_tab_name()` function - used to change a tab's name
- Add `LV_THEME_MATERIAL_FLAG_NO_TRANSITION` and `LV_THEME_MATERIAL_FLAG_NO_FOCUS` flags
- Reduce code size by adding: `LV_USE_FONT_COMPRESSED` and `LV_FONT_USE_SUBPX` and applying some optimization
- Add `LV_MEMCPY_MEMSET_STD` to use standard `memcpy` and `memset`

Bugfixes

- Do not print warning for missing glyph if its height OR width is zero.
- Prevent duplicated sending of `LV_EVENT_INSERT` from text area
- Tidy outer edges of cpicker widget.
- Remove duplicated lines from `lv_tabview_add_tab`
- btnmatrix: handle combined states of buttons (e.g. checked + disabled)
- textarea: fix typo in lv_textarea_set_scrollbar_mode
- gauge: fix image needle drawing
- fix using freed memory in _lv_style_list_remove_style

2.11.22 v7.2.0 (21.07.2020)

New features

- Add screen transitions with `lv_scr_load_anim()`
- Add display background color, wallpaper and opacity. Shown when the screen is transparent. Can be used with `lv_disp_set_bg_opa/color/image()`.
- Add `LV_CALENDAR_WEEK_STARTS_MONDAY`
- Add `lv_chart_set_x_start_point()` function - Set the index of the x-axis start point in the data array
- Add `lv_chart_set_ext_array()` function - Set an external array of data points to use for the chart
- Add `lv_chart_set_point_id()` function - Set an individual point value in the chart series directly based on index
- Add `lv_chart_get_x_start_point()` function - Get the current index of the x-axis start point in the data array
- Add `lv_chart_get_point_id()` function - Get an individual point value in the chart series directly based on index
- Add `ext_buf_assigned` bit field to `lv_chart_series_t` structure - it's true if external buffer is assigned to series
- Add `lv_chart_set_series_axis()` to assign series to primary or secondary axis
- Add `lv_chart_set_y_range()` to allow setting range of secondary y-axis (based on `lv_chart_set_range` but extended with an axis parameter)
- Allow setting different font for the selected text in `lv_roller`

- Add `theme->apply_cb` to replace `theme->apply_xcb` to make it compatible with the MicroPython binding
- Add `lv_theme_set_base()` to allow easy extension of built-in (or any) themes
- Add `lv_obj_align_x()` and `lv_obj_align_y()` functions
- Add `lv_obj_align_origo_x()` and `lv_obj_align_origo_y()` functions

Bugfixes

- `tileview` fix navigation when not screen sized
- Use 14px font by default to for better compatibility with smaller displays
- `linemeter` fix conversation of current value to "level"
- Fix drawing on right border
- Set the cursor image non-clickable by default
- Improve mono theme when used with keyboard or encoder

2.11.23 v7.1.0 (07.07.2020)

New features

- Add `focus_parent` attribute to `lv_obj`
- Allow using buttons in encoder input device
- Add lv_btnmatrix_set/get_align capability
- DMA2D: Remove dependency on ST CubeMX HAL
- Added `max_used` propriety to `lv_mem_monitor_t` struct
- In `lv_init` test if the strings are UTF-8 encoded.
- Add `user_data` to themes
- Add LV_BIG_ENDIAN_SYSTEM flag to lv_conf.h in order to fix displaying images on big endian systems.
- Add inline function lv_checkbox_get_state(const lv_obj_t * cb) to extend the checkbox functionality.
- Add inline function lv_checkbox_set_state(const lv_obj_t * cb, lv_btn_state_t state) to extend the checkbox functionality.

Bugfixes

- `lv_img` fix invalidation area when angle or zoom changes
- Update the style handling to support Big endian MCUs
- Change some methods to support big endian hardware.
- remove use of c++ keyword 'new' in parameter of function lv_theme_set_base().
- Add LV_BIG_ENDIAN_SYSTEM flag to lv_conf.h in order to fix displaying images on big endian systems.
- Fix inserting chars in text area in big endian hardware.

2.11.24 v7.0.2 (16.06.2020)

Bugfixes

- `lv_textarea` fix wrong cursor position when clicked after the last character
- Change all text related indices from 16-bit to 32-bit integers throughout whole library. #1545
- Fix gestures
- Do not call `set_px_cb` for transparent pixel
- Fix list button focus in material theme
- Fix crash when a text area is cleared with the backspace of a keyboard
- Add version number to `lv_conf_template.h`
- Add log in true double buffering mode with `set_px_cb`
- `lv_dropdown`: fix missing `LV_EVENT_VALUE_CHANGED` event when used with encoder
- `lv_tileview`: fix if not the {0;0} tile is created first
- `lv_debug`: restructure to allow asserting in from `lv_misc` too
- add assert if `_lv_mem_buf_get()` fails
- `lv_textarea`: fix character delete in password mode
- Update `LV_OPA_MIN` and `LV_OPA_MAX` to widen the opacity processed range
- `lv_btnm` fix sending events for hidden buttons
- `lv_gaguge` make `lv_gauge_set_angle_offset` offset the labels and needles too
- Fix typo in the API `scrllable` -> `scrollable`
- `tabview` by default allow auto expanding the page only to right and bottom (#1573)
- fix crash when drawing gradient to the same color

- chart: fix memory leak
- `img`: improve hit test for transformed images

2.11.25 v7.0.1 (01.06.2020)

Bugfixes

- Make Micropython working by adding the required variables as GC_ROOT
- Prefix some internal API functions with _ to reduce the API of LVGL
- Fix built-in SimSun CJK font
- Fix UTF-8 encoding when `LV_USE_ARABIC_PERSIAN_CHARS` is enabled
- Fix DMA2D usage when 32 bit images directly blended
- Fix lv_roller in infinite mode when used with encoder
- Add `lv_theme_get_color_secondary()`
- Add `LV_COLOR_MIX_ROUND_OFS` to adjust color mixing to make it compatible with the GPU
- Improve DMA2D blending
- Remove memcpy from `lv_ll` (caused issues with some optimization settings)
- `lv_chart` fix X tick drawing
- Fix vertical dashed line drawing
- Some additional minor fixes and formattings

2.11.26 v7.0.0 (18.05.2020)

Documentation

The docs for v7 is available at https://docs.littlevgl.com/v7/en/html/index.html

Legal changes

The name of the project is changed to LVGL and the new website is on https://lvgl.io

LVGL remains free under the same conditions (MIT license) and a company is created to manage LVGL and offer services.

New drawing system

Complete rework of LVGL's draw engine to use "masks" for more advanced and higher quality graphical effects. A possible use-case of this system is to remove the overflowing content from the rounded edges. It also allows drawing perfectly anti-aliased circles, lines, and arcs. Internally, the drawings happen by defining masks (such as rounded rectangle, line, angle). When something is drawn the currently active masks can make some pixels transparent. For example, rectangle borders are drawn by using 2 rectangle masks: one mask removes the inner part and another the outer part.

The API in this regard remained the same but some new functions were added:

- `lv_img_set_zoom`: set image object's zoom factor
- `lv_img_set_angle`: set image object's angle without using canvas
- `lv_img_set_pivot`: set the pivot point of rotation

The new drawing engine brought new drawing features too. They are highlighted in the "style" section.

New style system

The old style system is replaced with a new more flexible and lightweighted one. It uses an approach similar to CSS: support cascading styles, inheriting properties and local style properties per object. As part of these updates, a lot of objects were reworked and the APIs have been changed.

- more shadows options: *offset* and *spread*
- gradient stop position to shift the gradient area and horizontal gradient
- `LV_BLEND_MODE_NORMAL/ADDITIVE/SUBTRACTIVE` blending modes
- *clip corner*: crop the content on the rounded corners
- *text underline* and *strikethrough*
- dashed vertical and horizontal lines (*dash gap*, *dash_width*)
- *outline*: a border-like part drawn out of the background. Can have spacing to the background.
- *pattern*: display and image in the middle of the background or repeat it
- *value* display a text which is stored in the style. It can be used e.g. as a light-weighted text on buttons too.
- *margin*: similar to *padding* but used to keep space outside the object

Read the Style section of the documentation to learn how the new styles system works.

GPU integration

To better utilize GPUs, from this version GPU usage can be integrated into LVGL. In `lv_conf.h` any supported GPUs can be enabled with a single configuration option.

Right now, only ST's DMA2D (Chrom-ART) is integrated. More will in the upcoming releases.

Renames

The following object types are renamed:

- sw -> switch
- ta -> textarea
- cb -> checkbox
- lmeter -> linemeter
- mbox -> msgbox
- ddlist -> dropdown
- btnm -> btnmatrix
- kb -> keyboard
- preload -> spinner
- lv_objx folder -> lv_widgets
- LV_FIT_FILL -> LV_FIT_PARENT
- LV_FIT_FLOOD -> LV_FLOOD_MAX
- LV_LAYOUT_COL_L/M/R -> LV_LAYOUT_COLUMN_LEFT/MID/RIGHT
- LV_LAYOUT_ROW_T/M/B -> LV_LAYOUT_ROW_TOP/MID/BOTTOM

Reworked and improved object

- `dropdown`: Completely reworked. Now creates a separate list when opened and can be dropped to down/up/left/right.
- `label`: `body_draw` is removed, instead, if its style has a visible background/border/shadow etc it will be drawn. Padding really makes the object larger (not just virtually as before)
- `arc`: can draw background too.
- `btn`: doesn't store styles for each state because it's done naturally in the new style system.

- `calendar`: highlight the pressed datum. The used styles are changed: use `LV_CALENDAR_PART_DATE` normal for normal dates, checked for highlighted, focused for today, pressed for the being pressed. (checked+pressed, focused+pressed also work)
- `chart`: only has `LINE` and `COLUMN` types because with new styles all the others can be described. LV_CHART_PART_SERIES sets the style of the series. bg_opa > 0 draws an area in LINE mode. `LV_CHART_PART_SERIES_BG` also added to set a different style for the series area. Padding in `LV_CHART_PART_BG` makes the series area smaller, and it ensures space for axis labels/numbers.
- `linemeter`, `gauge`: can have background if the related style properties are set. Padding makes the scale/lines smaller. scale_border_width and scale_end_border_width allow to draw an arc on the outer part of the scale lines.
- `gauge`: `lv_gauge_set_needle_img` allows use image as needle
- `canvas`: allow drawing to true color alpha and alpha only canvas, add `lv_canvas_blur_hor/ver` and rename `lv_canvas_rotate` to `lv_canvas_transform`
- `textarea`: If available in the font use bullet (`U+2022`) character in text area password

New object types

- `lv_objmask`: masks can be added to it. The children will be masked accordingly.

Others

- Change the built-in fonts to Montserrat and add built-in fonts from 12 px to 48 px for every 2nd size.
- Add example CJK and Arabic/Persian/Hebrew built-in font
- Add ° and ”bullet” to the built-in fonts
- Add Arabic/Persian script support: change the character according to its position in the text.
- Add `playback_time` to animations.
- Add `repeat_count` to animations instead of the current ”repeat forever”.
- Replace `LV_LAYOUT_PRETTY` with `LV_LAYOUT_PRETTY_TOP/MID/BOTTOM`

Demos

- lv_examples was reworked and new examples and demos were added

New release policy

- Maintain this Changelog for every release
- Save old major version in new branches. E.g. `release/v6`
- Merge new features and fixes directly into `master` and release a patch or minor releases every 2 weeks.

Migrating from v6 to v7

- First and foremost, create a new `lv_conf.h` based on `lv_conf_template.h`.
- To try the new version it suggested using a simulator project and see the examples.
- If you have a running project, the most difficult part of the migration is updating to the new style system. Unfortunately, there is no better way than manually updating to the new format.
- The other parts are mainly minor renames and refactoring as described above.

2.12 Roadmap

This is a summary for planned new features and a collection of ideas. This list indicates only the current intention and it can be changed.

2.12.1 v8.2

See #2790

2.12.2 Ideas

- Reconsider color format management for run time color format setting, and custom color format usage. (Also RGB888)
- Make gradients more versatile
- Image transformations matrix
- Switch to RGBA colors in styles
- Consider direct binary font format support
- Simplify `group`s. Discussion is here.
- lv_mem_alloc_aligned(size, align)
- Text node. See #1701
- CPP binding. See Forum

- Optimize font decompression
- Need static analyze (via coverity.io or something else)
- Support dot_begin and dot_middle long modes for labels
- Add new label alignment modes. #1656
- Support larger images: #1892
- Curved text on path
- Variable binding improvements like Redux?
- Functional programming support, pure view? See here
- Circle layout. See #2871

2.13 项目实战

2.13.1 在 windwos 模拟器运行 lvgl(v8.0)

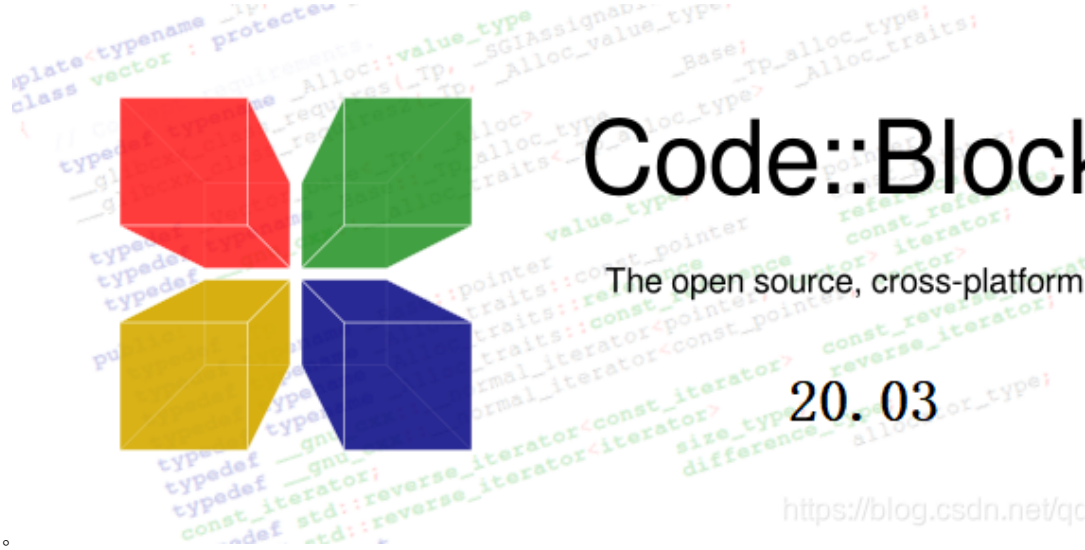
Code::Blocks 上运行

Code::Blocks 是一个免费开放源码的全功能的跨平台 C/C++ 集成开发环境。使用 Code::Blocks 模拟器体验或开发 lvlg，开箱即用很方便，相比 VS 更加轻量级。

获取资料

- 百度云下载：链接：https://pan.baidu.com/s/14renDMjP6xBVh0bGy_yAWA 提取码：root

获取 Code::Blocks 并安装



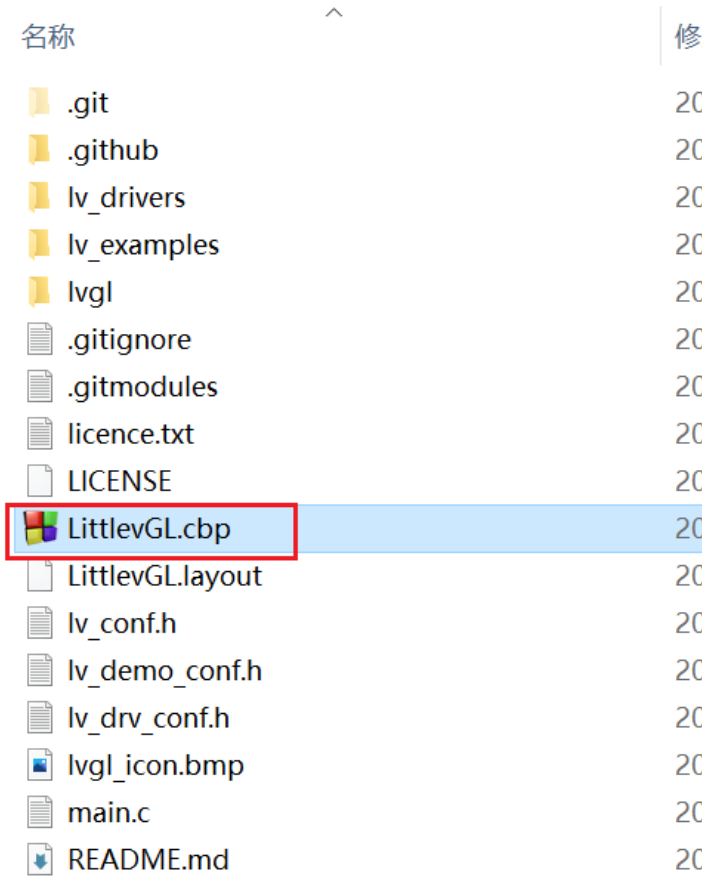
使用了当前的最新版本 `20.03` 。

软件安装包在资料中的这个位置：

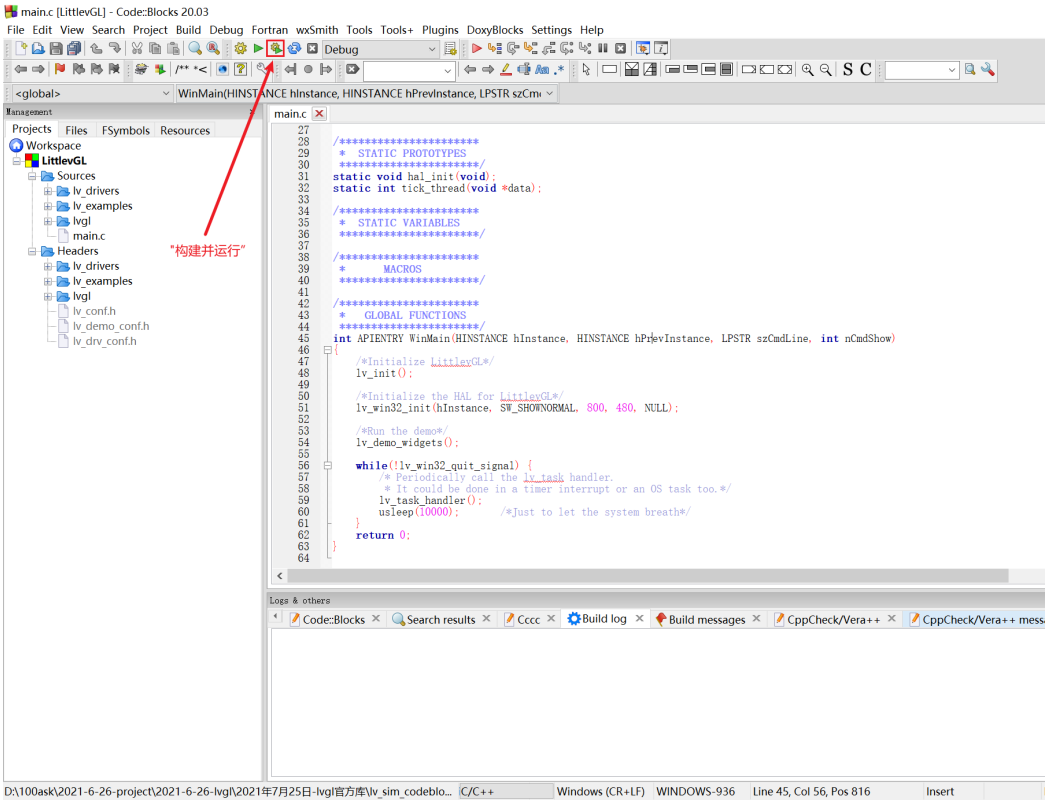
下载之后直接打开即可安装，安装过程按照软件提示进行安装即可，最后启动并打开 Code::Blocks，下一步准备通过 Code::Blocks 打开一个 lvgl 示例工程。

获取示例源码并运行

下载资料，从这里获取 lvgl 示例工程源码：

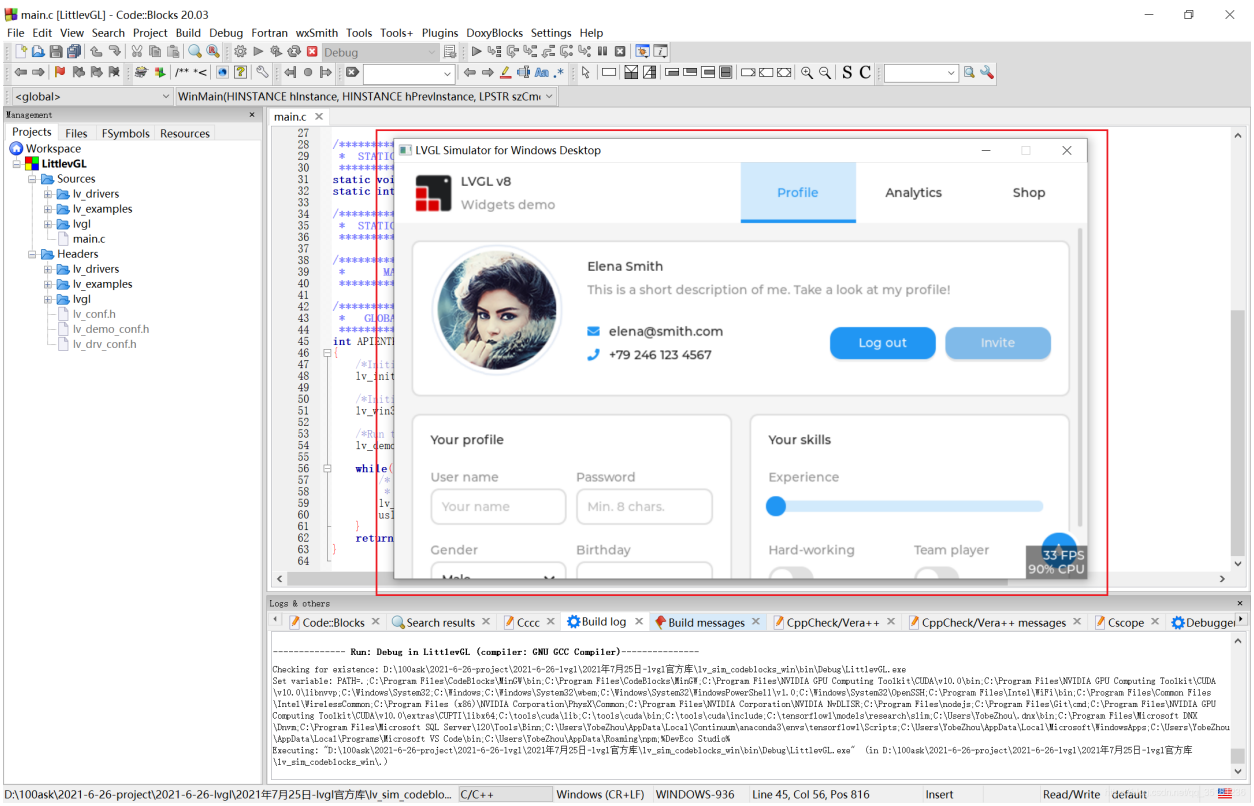


解压后,进入目录双击`LittlevGL.cbp`可直接打开项目工程:



单击图示的按钮，构建并运行项目：

运行结果：



尽情享受 LVGL 带来的惊喜吧！

在 Eclipse 上运行

TODO

在 vscode 上运行

TODO

2.13.2 STM32F103 LVGL GUI DEMO 效果

已实现了以下功能：

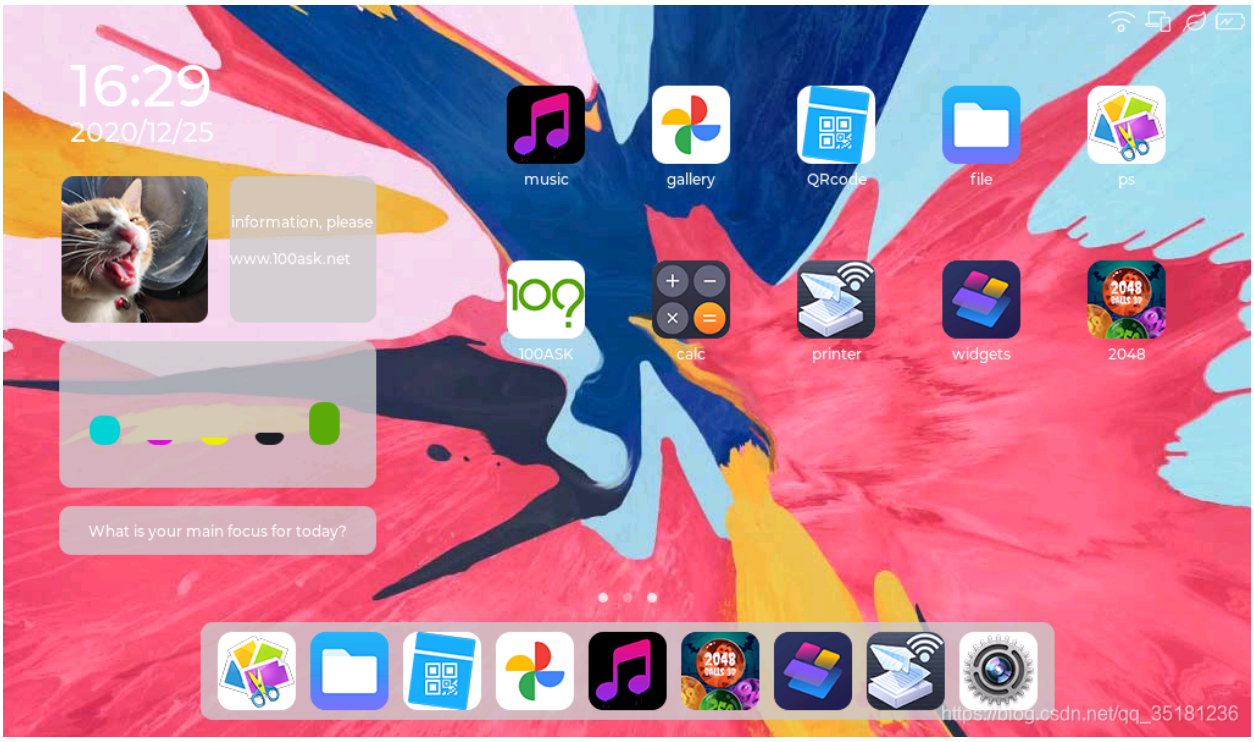
- 模仿 windows10 风格的文件浏览器
- 时钟
- 二维码生成器
- 系统说明
- 温湿度采集
- 2048 小游戏
- 音乐播放器
- 贪吃蛇小游戏

- 计算器
- 寄存器位查看工具
- 系统主题切换
- 板载硬件测试

STM32F103 LVGL GUI DEMO 源码

stm32f103 源码 git 仓库 (点击跳转)

2.13.3 STM32MP157 LVGL GUI DEMO 效果



- 图库
- 二维码生成器
- 文件浏览器
- 集成 lvgl 官方 demo：图片编辑器
- 集成 lvgl 官方 demo：音乐播放器
- 集成 lvgl 官方 demo：打印机
- 集成 lvgl 官方 demo：压力测试
- 2048 小游戏

- TODO...

STM32MP157 LVGL GUI DEMO 源码

stm32mp157-lvgl 源码 git 仓库 (点击跳转)

stm32mp157-buildroot 源码 git 仓库 (点击跳转)

2.13.4 IMX6ULL LVGL GUI DEMO 效果

IMX6ULL Linux LVGL GUI V2.0

Linux lvgl gui 2.0 和大家见面啦!

- 全新的架构，功能更强大
- 二次开发非常方便
- 独立的应用之间使用 dbus 通信
- GUI 基于 lvgl 8.1 开发，长期更新支持 lvgl 8.x
- 还有更多细节等你来探索!
- ...

  演示视频：https://www.bilibili.com/video/BV1nT4y1R7rz

源码

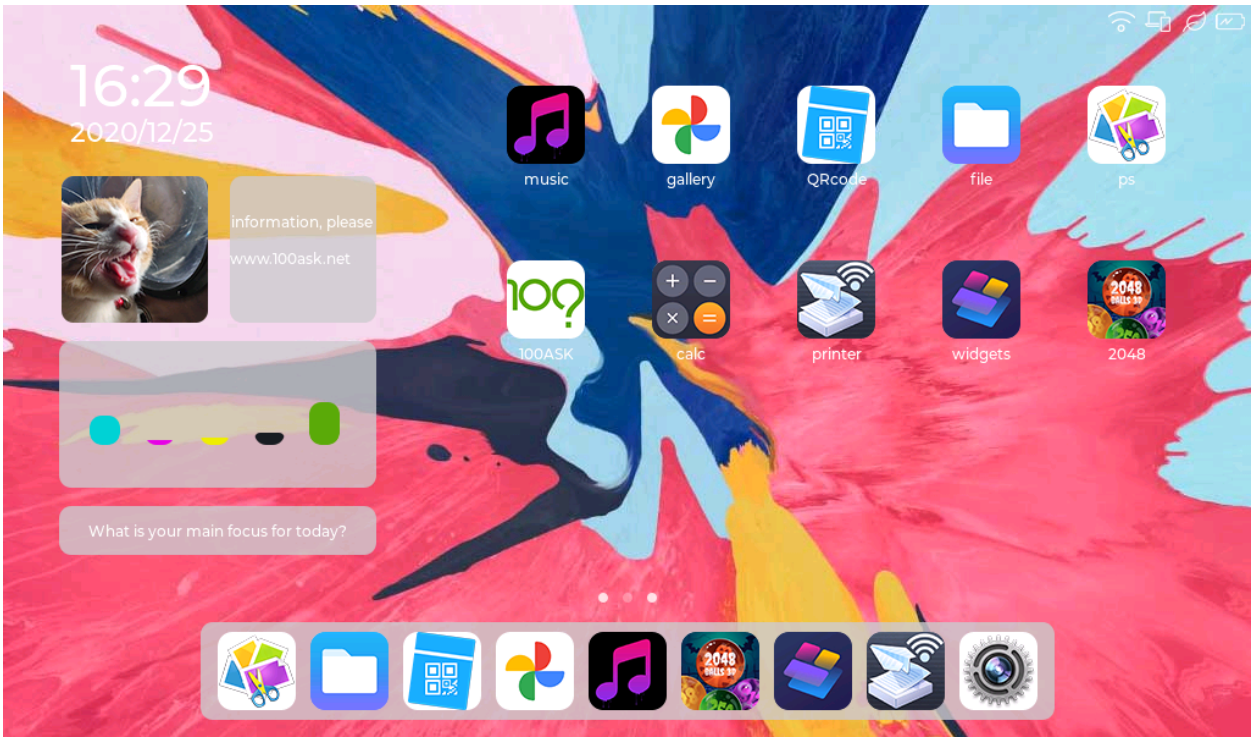
IMX6ULL Linux LVGL GUI V2.0 源码 git 仓库 (点击跳转)

IMX6ULL Linux LVGL GUI 1.0

- 图库
- 二维码生成器
- 文件浏览器
- 集成 lvgl 官方 demo：图片编辑器
- 集成 lvgl 官方 demo：音乐播放器
- 集成 lvgl 官方 demo：打印机
- 集成 lvgl 官方 demo：压力测试
- 2048 小游戏

- TODO...

  演示视频：https://www.bilibili.com/video/BV1cU4y1b7dY



源码

IMX6ULL Linux LVGL GUI 1.0 源码 git 仓库 (点击跳转)

2.13.5 ESP32 LVGL GUI DEMO 效果

TODO

2.14 开发实用工具

2.14.1 显示中文

要在 lvgl 中使用中文显示，我们需要用到两个东西：字体文件和字体转换器。

字体文件我们可以使用开源的字体或者自己制作出来，准备好了字体文件之后使用字体转换器即可转换成可以在 lvgl 上使用的字体格式。

下面给提供了开源可免费商用的字体文件共大家使用，下载好字体文件之后，再使用字体转换器转换字体即可。

中文字体文件

中文使用 unicode 编码^1。

思源字体

更多字体

等宽字体

等宽字体（英语：Monospaced Font）是指字符宽度相同的电脑字体。与此相对，字符宽度不尽相同的电脑字体称为比例字体。^1

在传统西文印刷中，比例字体可以提高单词的可读性。但因打字机及早期的电脑画面显示等由于技术的局限，无法进行字母宽度的比例调整，因此将每个字符都制作成一样的宽度，从而形成了等宽字体。在等宽字体中，字母“i”、“j”显得两侧余白较多，而字母“w”、“m”等的笔画显得相当拥挤。

但是随着图形用户界面主流的更新和电脑技术的提高，处理比例字体的局限性得到了突破，因此现在排版上显得比较自然的比例字体的使用已经相当普及。

东亚语言中，方块字基本上都作为等宽字体处理，如各个地区的汉字、日语假名的全角字符、韩语谚文音节等字符都是等宽的。但是一些中文、日文字体中，由于同时涵盖西文字符，因此也含有比例字体，造成一个字体中两种类型混合的局面。

东亚文字的标点符号有时会随标准规定、挤压处理而存在宽度可变的情况；CSS 中有一个 font-variant-east-asian 设定可以选择宽度策略。

Windows 简体中文操作系统中，旧版本的默认字体中易宋体全部是等宽字体，而 Windows Vista 的默认字体微软雅黑中，等宽中应属半角的西文部分是比例字体，全角字符是等宽字体。

等宽字体文件下载

更多字体

常用汉字

常用 495 个汉字

雕虎的一了是我不在人们有来他这上着个地到大里说就去子得也和那要下看天时过出小么起你都把好还多没为又可家学只以主会样年想生同老中十从自面前头道它后然走很像见两用她国动进成回什边作对开而己些现山民候经发工向事命给长水几义三声于高手知理眼志点心战二问但身方实吃做叫当住听革打呢真全才四已所敌之最光产情路分总条白话东席次亲如被花口放儿常气黄五第使写军木珍吧文运再果怎定许快明行因别飞外树物活部门无往船望新带队先力完却站代员机更九您每风级跟笑啊孩万少直意夜比阶连车重便斗马哪化太指变社似士者干石满梅日决百原拿群究各六本思解立河村八难早论吗根共让相研今其书坐接应关信觉步反处记将千找争领或师结块跑谁草越字加脚紧爱等习阵怕月青半火法题建赶位唱海七女任件感准张团屋离色脸片科倒睛利世刚且由送切星导晚表够整认响雪流未场该并底深刻平伟忙提确近亮轻讲农古黑告界拉名呀土清阳照办史改历转画造嘴此治北必服雨穿内识验传业菜爬睡兴形量咱观苦体众通冲合破友度术饭公旁房极南枪读沙岁线野坚空收算至政城劳落钱特围弟胜教热展包歌类渐强数乡呼音答哥际旧神座章帮啦受系令跳非何牛取入岸敢掉忽种装顶急戴林停息句区衣般报叶压慢叔背细艳佐

（下页继续）

(续上页)

常用 3500 个汉字

一丁七万丈三三上下不与丐丑专且世丘丙业丛东丝丢两严丧个中丰串临丸丹为主丽举乃久么义之乌乍乎乏乐乒乓乔乖乘乙九乞也习乡书买乱乳了予争事二于亏云互五井亚些亡交亥亦产亩享京亭亮亲人亿什仁仅仆仇今介仍从仑仓仔他仗付仙代令以仪们仰仲件价任份仿企伊伍伏伐休众优伙会伞伟传伤伦伪伯估伴伶伸伺似伽但位低住佑体何余佛作你佣佩佳使侄侈例侍供依侠侣侥侦侧侨侮侯侵便促俄俊俏俐俗俘保信俩俭修俯俱俺倍倒倔倘候倚借倡倦债值倾假偎偏做停健偶偷偿傀傅傍储催傲傻像僚僧僵僻儒儡儿允元兄充兆先光克免兑兔党兜兢入全八公六兰共关兴兵其具典养兼兽冀内冈册再冒冕冗写军农冠冤冬冯冰冲决况冶冷冻净凄准凉凌减凑凛凝几凡凤凫凭凯凰凳凶凸凹出击函凿刀刁刃分切刊刑划列刘则刚创初删判刨利别刮到制刷券刹刺刻刽剂剃削前剑剔剖剥剧剩剪副割剿劈力劝办功加务劣动助努劫励劲劳势勃勇勉勋勒勘募勤勺勾勿匀包匆匈匕化北匙匠匣匪匹区医匾匿十千升午半华协卑卒卓单卖南博卜占卡卢卤卦卧卫印危即却卵卷卸卿厂厅历厉压厌厕厘厚原厢厦厨去县参又叉及友双反发叔取受变叙叛叠口古句另叨只叫召叭叮可台史右叶号司叹叼叽吁吃各吆合吉吊同名后吏吐向吓吕吗君吝吞吟吠否吧吨吩含听吭吮启吱吴吵吸吹吻吼呀呆呈告呐呕员呛呜呢周味呵呻呼命咆和咏咐咒咕咖咙咧咨咪咬咱咳咸咽哀品哄哆哈响哎哑哗哟哥哨哩哪哭哮哲哺哼唁唆唇唉唐唠唤唧唬售唯唱唾啃啄商啊啡啤啥啦啸啼喂善喇喉喊喘喜喝喧喳喷喻喽嗓嗜嗡嗤嗦嗽嘀嘛嘉嘱嘲嘴嘶嘻嘿器噩噪噱噶噻嚣囊囚四回因团囤园困囱围固国图圃圆圈土圣在地场圾址均坊坎坏坐坑块坚坛坝坞坟坠坡坤坦坪坯坷垂垃垄型垒垦垢垫垮埂埋城域埠培基堂堆堕堡堤堪堰堵塌塑塔塘塞填境墅墓墙增墨墩壁壕壤士壮声壳壶壹处备复夏夕外多夜够大天太夫夭央夯失头夷夸夹夺奄奇奈奉奋奏契奔奕奖套奠奢奥女奴奶奸她好如妄妆妇妈妒妓妖妙妥妨妹妻姆姊始姐姑姓委姚姜姥姨姻姿威娃娄娇娘娜娩娱娶婆婉婚婴婶婿媒媚媳嫁嫂嫉嫌嫡嫩嬉子孔孕字存孙孝孟季孤学孩孵孽宁它宅宇守安宋完宏宗官宙定宛宜宝实宠审客宣室宦宪宫宰害宴宵家容宽宾宿寂寄密寇富寒寓寝寞察寡寥寨寸对寺寻导寿封射将尉尊小少尔尖尘尚尝尤就尸尺尼尽尾尿局屁层居屈屉届屋屎屏屑展属屠屡履屯山屹屿岁岂岔岖岗岛岩岭岳岸峡峦峭峰峻崇崎崔崖崩崭嵌巍川州巡巢工左巧巨巩巫差己已巴巷巾币市布帅帆师希帐帕帖帘帚帜帝带席帮常帽幅幌幔幕幢干干平年并幸幻幼幽广庄庆庇床序庐库应底店庙府庞废度座庭庵庶康庸廉廊廓延廷建开异弃弄弊式弓引弛弟张弥弦弧弯弱弹强归当录形彤彩彪彬彭彰影役彻彼往征径待很徊律徐徒徙得徘御循微德徽心必忆忌忍志忘忙忠忧快忱念忽忿怀态怎怒怔怕怖怜思怠急性怨怪怯总恃恋恍恐恒恕恢恤恨恩恬恭息恰恳恶恼悄悉悍悔悟悠患悦您悬悯悲悴悼情惊惋惑惕惜惠惦惧惨惩惫惭惯惰想惶惹愁愈愉意愕愚感愤愧愿慈慌慎慕慢慧慨慰慷憋憎憔憨憾懂懈懊懒懦戈戏成我戒或战戚截戳戴户房所扁扇手才扎扑扒打扔托扛扣执扩扫扬扭扮扯扰扳扶批扼找承技抄把抑抒抓投抖抗折抚抛抠抡抢护报披抬抱抵抹押抽拂拄担拆拇拉拌拍拐拒拓拔拖拗拘拙招拜拟拢拣拥拦拧拨择括拭拯拱拴拷拼拾拿持挂指按挎挑挖挚挟挠挡挣挤挥挨挪挫振挺挽捂捅捆捉捌捍捎捏捐捕捞损捡换捣捧据捶捷捺捻掀掂授掉掌掏掐排掖掘掠探接控推掩措掰掷掸掺揉揍提插揖握揣揩揪揭援揽揿搁搂搅搏搓搔搜搞搪搬搭携摄摆摇摊摔摘摧摩摸摹撇撑撒撕撞撤撩撬播撮撰撵撼擂擅操擎擒擦攀攒攘支收改攻放政故效敌敏救教敛敞敢散敦敬数敲整敷文斋斑斗料斜斟斤斥斧斩断斯新方施旁旅旋族旗无既日旦旧旨早旬旭旱时旷旺昂昆昌明昏易昔昙星映春昧昨昭是昵昼显晃晋晌晒晓晕晚晤晦晨普景晰晴晶智晾暂暇暑暖暗暮暴曙曲更曹曼曾替最月有朋服朗望朝期朦木未末本术朱朴朵机朽杀杂权杆杈杉李杏材村杖杜束杠条来杨杭杯杰松板极构枉析枕林枚果枝枢枣枪枫枯架枷柄柏某柑柒染柔柜柠查柬柱柳柴柿栅标栈栋栏树栓栖栗校株样核根格栽桂桃桅框案桌桐桑档桥桦桨桩桶梁梅梆梗梢梦梧梨梭梯械梳检棉棋棍棒棕棘棚棠森棱棵棺椅植椎椒椭椰椿椟楚楞楣楷楼概榄榆榔榕榛榜榨榴槐槽樊樟模横樱橄橘橙橡橱檀檐檩檬欠次欢欣欧欲欺款歇歉歌止正此步武歧歪歹死歼殁殉殊残殖殴段殷殿毁毅母每毒比毕毙毛毡毫毯氏民氓气氛氢氧氨氮氯水永汁求汇汉汗汛汞江池污汤汪汰汹汽汾沈沉沐沙沛沟没沥沦沧沪沫沮河沸油治沼沽沾沿泄泉泊泌法泛泞泡波泣泥注泪泰泳泵泻泼泽泾洁洋洒洗洛洞津洪洲活洼洽派流浅浆浇浊测济浑浓浙浦浩浪浮浴海浸涂消涉涌涎涕涛涝涡涣涤润涧涨涩涮涯液涵淀淆淋淌淑淘淡淤淫淮深淳混淹添清渊渐渔渗渠渡渣渤温港渴游渺湃湖湘湾湿溃溅溉源溜溢溪溯溶溺滋滑滞滔滚带满滤滥滨滩滴漂漆漏漓演漠漩漫漱漾潘潜潦潭潮澄澈澎澜澡澳激濒瀑灌火灭灯灰灵灶灸灼灾灿炉炊炎炒炕炫炬炭炮炸点炼炼烂烈烘烙烛烟烤烦烧烫热烹焊焕焙焚焦焰然煌煎煞煤照煮熄熊熏熔熙熟熬燃燎燕燥爆爪爬爱爵父爷爸爹爽片版牌牍牙牛牡牢牧物牲牵特牺犀犁犬犯状犹狂狈狐狗狞狠狡独狭狮狰狱狸狼猎猖猛猜猩猪猫猬献猴猾猿玄率玉王玖玛玩玫环现玲玷玻珊珍珠班球琅理琼琐琢琳琴琼瑞瑟瑰璃璧瓜瓢瓣瓤瓦瓮瓶瓷甘甚甜生甥用甩甫甬田由甲申电男甸画畅界畏畔留畜略畦番畴畸疆疏疑疗疙疚疟疤疫疮疯疲疹疼疾病症痉痒痕痘痛痢痪痰痴痹瘟瘤瘦瘩瘪瘫瘾瘸癌癣癫登白百皂的皆皇皮皱皿盅盆盈益盎盐监盒盖盗盘盛盟目盯盲直相盹盼盾省眉看真眠眨眯眶眷眼着睁睛睡督睦睬睹睾瞄瞎瞒瞧瞪瞬瞳瞻蠢矛矢知矩矫短矮石矶矿码砂砌砍研砖砚砰破砸砾础硅硕硝硫硬确硼硼碌碍碎碑碗碘碟碧碰碱碳碴碾磁磅磕磨磷磺礁示礼社祈祖祝神祟祠祥票祭祷祸禀禁福离禽禾秀私秃秆秉秋种科秒秕秘租秤秦秧秩秫积称秸移秽稀程稍税稚稠稳稻稼稽稿穆穗穴究穷空穿突窃窄窍窑窒窗窘

(下页继续)

(续上页)

字符集编码范围

汉字 Unicode 编码范围

Unicode 是全球文字统一编码。它把世界上的各种文字的每一个字符指定唯一编码，实现跨语种、跨平台的应用。

中文用户最常接触的是汉字 Unicode 编码。中文字符数量巨大，日常使用的汉字数量有数千个，再加上生僻字，数量达到数万个。下面这个表格将中文字符集的 Unicode 编码范围列出 [^2]：

更详细的内容请参考：中日韓統一表意文字

拉丁字母 Unicode 编码范围

基本拉丁字母共有 95 个字符，其中 52 个属于拉丁字母，剩下的 43 个属于基本字符。

有 33 个字符被定义为“ASCII 标点及符号”，有时也被称为“ASCII 特殊字符”。

下面这个表格将基本拉丁字母字符集的 Unicode 编码范围列出 [^3]：

更详细的内容请参考：Unicode 字符列表

综上所述，我们可以这样简单总结下来：

- 数字 0-9：0123456789
- 小写英文字母：abcdefghijklmnopqrstuvwxyz
- 大写英文字母：ABCDEFGHIJKLMNOPQRSTUVWXYZ
- 标点及符号：`!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~`

```
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
↪`abcdefghijklmnopqrstuvwxyz{|}~
```

ASCII 码表

ASCII 码使用指定的 7 位或 8 位二进制数组合来表示 128 或 256 种可能的字符。标准 ASCII 码也叫基础 ASCII 码，使用 7 位二进制数（剩下的 1 位二进制为 0）来表示所有的大写和小写字母，数字 0 到 9、标点符号，以及在美式英语中使用的特殊控制字符 [^4]。

更详细的内容请参考：ASCII

```
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
↪`abcdefghijklmnopqrstuvwxyz{|}~
```

字体转换工具

lvgl 官方在线转换工具

lvgl 官方提供的字体转换器已经非常好了，只要手里有字体文件之后就可以提取转换想要的字体。

lvgl 官方字体转换器地址：https://lvgl.io/tools/fontconverter

使用教程

https://www.bilibili.com/video/BV1Ya411r7K2?p=15

本站提供的内容仅用于个人学习、研究或欣赏。我们不保证内容的正确性。通过使用本站内容随之而来的风险与本站无关!

访问者可将本网站提供的内容或服务用于个人学习、研究或欣赏，以及其他非商业性或非盈利性用途，但同时应遵守著作权法及其他相关法律的规定，不得侵犯本网站及相关权利人的合法权利。

本网站内容原作者如不愿意在本网站刊登内容，请及时通知本站，予以删除。

常用汉字

常用 495 个汉字

雕虎的一了是我不在人们有来他这上着个地到大里说就去子得也和那要下看天时过出小么起你都把好还多没为又可家学只以主会样年想生同老中十从自面前头道它后然走很像见两用她国动进成回什边作对开而己些现山民候经发工向事命给长水几义三声于高手知理眼志点心战二问但身方实吃做叫当住听革打呢真全才四已所敌之最光产情路分总条白话东席次亲如被花口放儿常气黄五第使写军木珍吧文运再果怎定许快明行因别飞外树物活部门无往船望新带队先力完却站代员机更九您每风级跟笑啊孩万少直意夜比阶连车重便斗马哪化太指变社似士者干石满梅日决百原拿群究各六本思解立河村八难早论吗根共让相研今其书坐接应关信觉步反处记将千找争领或师结块跑谁草越字加脚紧爱等习阵怕月青半火法题建赶位唱海七女任件感准张团屋离色脸片科倒睛利世刚且由送切星导晚表够整认响雪流未场该并底深刻平伟忙提确近亮轻讲农古黑告界拉名呀土清阳照办史改历转画造嘴此治北必服雨穿内识验传业菜爬睡兴形量咱观苦体众通冲合破友度术饭公旁房极南枪读沙岁线野坚空收算至政城劳落钱特围弟胜教热展包歌类渐强数乡呼音答哥际旧神座章帮啦受系令跳非何牛取入岸敢掉忽种装顶急戴林停息句区衣般报叶压慢叔背细艳佐

常用 3500 个汉字

一丁七万丈三三上下不与丐丑专且世丘丙业丛东丝丢两严丧个中丰串临丸丹为主丽举乃久么义之乌乍乎乏乐乒乓乔乖乘
乙九乞也习乡书买乱乳了予争事二于亏云互五井亚些亡交亥亦产亩享京亭亮亲人亿什仁仅仆仇今介仍从仑仓仔他仗付仙
代令以仪们仰仲件价任份仿企伊伍伏伐休众优伙会伞伟传伤伦伪伯估伴伶伸伺似佃但位低住佑体何余佛作你佣佩佳使侄
侈例侍供依侠侣侥侦侧侨侮侯侵便促俄俊俏俐俗俘保信俩俭修俯俱俺倍倒倔倘候倚借倡倦债值倾假偎偏做停健偶偷偿傀
傅傍储催傲傻像僚僧僵僻儒儡儿允元兄充兆先光克免兑兔党兜兢入全八公六兰共关兴兵其具典养兼兽冀内冈册再冒冕冗
写军农冠冤冬冯冰冲决况冶冷冻净凄准凉凌减凑凛凝几凡凤凫凭凯凰凳凶凸凹出击函凿刀刁刃分切刊刑划列刘则刚创初
删判刨利别刮到制刷券刹刺刻刽剂剃削前剑剔剖剥剧剩剪副割劃劈力劝办功加务劣动助努劫励劲劳势勃勇勉勋勒勘募勤
勺勾勿匀包匆匈匕化北匙匠匣匪匹区医匾匿十千升午半华协卑卒卓单卖南博卜占卡卢卤卦卧卫印危即却卵卷卸卿厂厅历
厉压厌厕厘厚原厢厦厨去县参又叉及友双反发叔取受变叙叛叠口古句另叨只叫召叭叮可台史右叶号司叹叼叽吁吃各吆合
吉吊同名后吏吐向吓吕吗君吝吞吟吠否吧吨吩含听吭吮启吱吴吵吸吹吻吼呀呆呈告呐呕员呛呜呢周味呵呻呼命咆和咏咐
咒咕咖咙咧咨咪咬咱咳咸咽哀品哄哆哈响哎哑哗哟哥哨哩哪哭哮哲哺哼唁唆唇唉唐唠唤唧唬售唯唱唾啃啄商啊啡啤啥啦
F啸啼喂善喇喉喊喘喜喝喧喳喷喻喽嗓嗜嗡嗤嗦嗽嘀嘱嘲嘴嘶嘹嘿器噩噪噱噶噻嚼囊囚四回因团囤园困囱围固国图
圃圆圈土圣在地场圾址均坊坎坏坐坑块坚坛坝坞坟坠坡坤坦坪坯垂垃垄型垒垦垫垮垛垢垦垫垮埂埋城域埠培基堂堆堕堡堤
堪堰堵塌塑塔塘塞填境墅墓墙增墨墩壁壕壤士壮声壳壶壹处备复夏夕外多夜够大天太夫夭央夯失头夷夸夹夺奄奇奈奉奋
奏契奔奕奖套奠奢奥女奴奶奸她好如妄妆妇妈妒妓妖妙妥妨妹妻姆姊始姐姑姓委姚姜姥姨姻姿威娃娄娇娘娜娩娱娶婆婉
婚婴婶婿媒媚媳嫁嫂嫉嫌嫡嫩嬉子孔孕字存孙孝孟季孤学孩孵孽宁它宅宇守安宋完宏宗官宙定宛宜宝实宠审客宣室宦宪
宫宰害宴宵家容宽宾宿寂寄密寇富寒寓寝寞察寡寥寨寸对寺寻导寿封射将尉尊小少尔尖尘尚尝尤就尸尺尼尽尾尿局屁层
居屈屉届屋屎屏屑展属屠屡履屯山屹屿岁岂岔岖岗岛岩岭岳岸峡峦峙峭峰峻崇崎崔崖崩崭嵌巍川州巡巢工左巧巨巩巫差己
已巴巷巾币市布帅帆师希帐帕帖帘帚帜帝带席帮常帽幅幌幔幕幢干干平年并幸幻幼幽广庄庆庇床序庐库应底店庙府庞废
度座庭庵庶康庸廉廊廓延廷建开异弃弄弊式弓引弛弟张弥弦弧弯弱弹强归当录形彤彩彪彬彭彰影役彻彼往征径待很徊律
徐徒徙得徘御循微德徽心必忆忌忍志忘忙忠忧快忱念忽忿怀态怎怒怔怕怖怜思怠急性怨怪怯总恃恋恍恐恒恕恢恤恨恩恬
恭息恰恳恶恼悄悉悍悔悟悠患悦您悬悯悲悴悼情惊惋惑惕惜惠惦惧惨惩惫惭惯惰想惶惹愁愈愉意愕愚感愤愧愿慈慌慎慕
慢慧慨慰慷憋憎憔憨憾懂懈懊懒懦戈戏成我戒或战戚截戳戴户房所扁扇手才扎扑打扔托扛扣执扩扫扬扭扮扯扰扳扶批
扼找承技抄把抑抒抓投抖抗折抚抛抠抡抢护报披抬抱抵抹押抽拂拄担拆拇拉拌拍拐拒拓拔拖拗拘拙招拜拟拢拣拥拦拧拨
择括拭拯拱拳拴拷拼拾拿持挂指按挎挑挖挚挟挠挡挣挤挥挨挪挫振挺挽捂捅捆捉捌捍捐捕捞损捡换捣捧据捶捷捺捻
掀掂授掉掌掏掐排掖掘掠探接控推掩措掰掷掸掺揉揍描提插揖握揣揩揪揭援揽搀搁搂搅搏搓搔搜搞搪搬搭携摄摆摇摊摔
摘摧摩摸摹撇撑撒撕撞撤撩撬播撮撰撵撼擂擅操擎擒擦攀攒攘支收改攻放政故效敌敏救教敛敞敢散敦敬数敲整敷文斋斑
斗料斜斟斤斥斧斩断斯新方施旁旅旋族旗无既日旦旧旨早旬旭旱时旷旺昂昆昌明昏易昔昙星映春昧昨昭是昵昼显晃晋晌
晒晓晕晚晤晦晨普景晰晴晶智晾暂暇暑暖暗暮暴曙曲更曹曼曾替最月有朋服朗望朝期朦木未末本术朱朴朵机朽杀杂权杆
杈杉李杏材村杖杜束杠条来杨杭杯杰松板极构枉析枕林枚果枝枢枣枪枫枯架枷柄柏某柑柒染柔柜柠查柬柱柳柴柿栅标栈
栋栏树栓栖栗校株样核根格栽桂桃桅框案桌桐桑档桥桦桨桩桶梁梅梆梗梢梦梧梨梭梯械梳检棉棋棍棒棕棘棚棠森棱棵棺
椅植椎椒椭椰椿楔楚楞楣楷楼概榄榆榔榕榛榜榨榴槐槽樊樟模横樱橄橘橙橡橱檀檐檩檬欠次欢欣欧欲欺款歇歉歌止正此
步武歧歪歹死歼殃殊残殖殴段殷殿毁毅母每毒比毕毙毛毡毫毯氏民氓气氛氢氧氨氮氯水永汁求汇汉汗汛汞江池污汤汪
汰汹汽沃沈沉沐沙沛沟没沥沦沧沪沫沮河沸油治沼沽沾沿泄泉泊泌法泛泞泡波泣泥注泪泰泳泵泻泼泽洁洋洒洗洛洞津洪
洲活洼洽派流浅浆浇浊测济浑浓浙浦浩浪浮浴海浸涂消涉涌涎涕涛涝涡涣涤润涧涨涩涮涯液涵淀淆淋淌淑淘淡淤淫淮深
淳混淹添清渊渐渔渗渠渡渣渤温港渴游渺湃湖湘湾湿溃溅溉源溜溢溪溯溶溺滋滑滓滔滚滞满滤滥滨滩滴漂漆漏漓演漠漩
漫漱漾潘潜潦潭潮澄澈澎澜澡澳激濒瀑灌火灭灯灰灵灶灸灼灾灿炉炊炎炒炕炫炬炭炮炸点炼炼烂烈烘烙烛烟烤烦烧烫热
烹焊焕焙焚焦焰然煌煎煞煤照煮熄熊熏熔熙熟熬燃燎燕燥爆爪爬爱爵父爷爸爹爽片版牌牍牙牛牡牢牧物牲牵特牺犀犁犬
犯状犹狂狈狐狗狞狠狡独狭狮狰狱狸狼猎猖猛猜猩猪猫猬献猴猾猿玄率玉王玖玛玩玫环现玲玷玻珊珍珠班球琅理琉琐琢
琳琴琼瑞瑟瑰璃璧瓜瓢瓣瓤瓦瓮瓶瓷甘甚甜生甥用甩甫甬田由甲申电男甸画畅界畏畔留畜略畦番畴畸疆疏疑疗疙疚疟疤疫
疮疯疲疹疼疾病症痊痒痕痘痛痢痪痰痴痹瘟瘤瘦瘩瘪瘫瘾瘸癌癣癫登白百皂的皆皇皮皱皿盅盆盈益盎盐监盒盔盖盗盘盛
盟目盯盲直相盹盼盾省眉看真眠眨眯眶眷眼着睁睛睡督睦睬睹睫瞄瞎瞒瞧瞪瞬F瞳瞻矗矛矢知矩矫短矮石矶矿码砂砌砍研
砖砚砰破砸砾础砷硅硕硝硫硬确硼碉碌碍碎碑碗碘碟碧碰碱碳碴碾磁磅磕磨磷磺礁示礼社祈祖祝神祟祠祥票祭祷祸禀禁禄
离禽禾秀私秃秆秉秋种科秒秕秘租秤秦秧秩秫积称秸移秽稀程稍税稚稠稳稻稼稽稿穆穗穴究穷空穿突窃窄窍窑窒窗窘
窜窝窟窥窿立竖站竞竟章竣童竭端竹竿笆笋笑笔笛笞符笨第笼等筋筏筐筑筒答策筛筝筷筹签简箍箕算管箩箫箭箱箴箸
篙篡篮篱篷簇簸簿籍米类籽粉粒粗粘粟粤粥粪粮粱粹精糊糕糖糙糟糠糯糊系紊素索紧紫累絮繁纠红纤约级纪纫纬纯纱纲
纳纵纷纸纹纺纽线练组绅细织终绊绍绎经绑绒结绕绘给络绝绞统绢绣继绩绪续绰绳维绵绷绸综绽绿缀缅缆缎缓缔缕编缘
缚缝缠缤缨缩缭缰缴缸缺罐网罕罗罚罢罩罪置署羊美羔差羡群羹羽翁翅翎翔翘翠翩翰翻翼耀老考者而要耐耕耗耙耳耸

(续上页)

字符集编码范围

汉字 Unicode 编码范围

Unicode 是全球文字统一编码。它把世界上的各种文字的每一个字符指定唯一编码，实现跨语种、跨平台的应用。

中文用户最常接触的是汉字 Unicode 编码。中文字符数量巨大，日常使用的汉字数量有数千个，再加上生僻字，数量达到数万个。下面这个表格将中文字符集的 Unicode 编码范围列出[^2]：

更详细的内容请参考：中日韓統一表意文字

拉丁字母 Unicode 编码范围

基本拉丁字母共有 95 个字符，其中 52 个属于拉丁字母，剩下的 43 个属于基本字符。

有 33 个字符被定义为“ASCII 标点及符号”，有时也被称为“ASCII 特殊字符”。

下面这个表格将基本拉丁字母字符集的 Unicode 编码范围列出[^3]：

更详细的内容请参考：Unicode 字符列表

综上所述，我们可以这样简单总结下来：

- 数字 0-9：0123456789
- 小写英文字母：abcdefghijklmnopqrstuvwxyz
- 大写英文字母：ABCDEFGHIJKLMNOPQRSTUVWXYZ
- 标点及符号：`!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~`

```
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
↪`abcdefghijklmnopqrstuvwxyz{|}~
```

ASCII 码表

ASCII 码使用指定的 7 位或 8 位二进制数组合来表示 128 或 256 种可能的字符。标准 ASCII 码也叫基础 ASCII 码，使用 7 位二进制数（剩下的 1 位二进制为 0）来表示所有的大写和小写字母，数字 0 到 9、标点符号，以及在美式英语中使用的特殊控制字符[^4]。

更详细的内容请参考：ASCII

```
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
↪`abcdefghijklmnopqrstuvwxyz{|}~
```

字体转换工具

lvgl 官方在线转换工具

lvgl 官方提供的字体转换器已经非常好了，只要手里有字体文件之后就可以提取转换想要的字体。

lvgl 官方字体转换器地址：https://lvgl.io/tools/fontconverter

使用教程

https://www.bilibili.com/video/BV1Ya411r7K2?p=15

本站提供的内容仅用于个人学习、研究或欣赏。我们不保证内容的正确性。通过使用本站内容随之而来的风险与本站无关!

访问者可将本网站提供的内容或服务用于个人学习、研究或欣赏，以及其他非商业性或非盈利性用途，但同时应遵守著作权法及其他相关法律的规定，不得侵犯本网站及相关权利人的合法权利。

本网站内容原作者如不愿意在本网站刊登内容，请及时通知本站，予以删除。

2.14.2 自定义 SYMBOL(图标) 字体

- 文档教程：https://forums.100ask.net/t/topic/1013
- 视频教程：https://www.bilibili.com/video/BV1Ya411r7K2?p=55

Unicode 转换为 utf-8 的工具

- 工具 1：https://www.cogsci.ed.ac.uk/~richard/utf-8.cgi
- 工具 2：https://www.qqxiuzi.cn/bianma/Unicode-UTF.php

2.14.3 让 lvgl 支持中文输入

`lv_chinese_ime` 是在 lv_keyboard 的基础上编写的一个自定义部件，它和*lv_keyboard* 没有什么区别，只是增加了支持中文输入法 (拼音) 的功能。

所以我们将其称为：**支持中文输入法的 LVGL 键盘 (lv_keyboard) 部件增强插件**。

正常来说，只要是 lvgl 能运行的环境 `lv_chinese_ime` 也能够运行！影响因素主要有两点：使用的字库文件大小和使用的词库大小。

`lv_chinese_ime` 使用起来非常简单，后续自定义拓展功能也很方便，更多功能敬请期待。

演示视频：https://www.bilibili.com/video/BV1DY41147xX

源码仓库

- github: https://github.com/100askTeam/lv_chinese_ime
- gitee: https://gitee.com/weidongshan/lv_chinese_ime

2.14.4 lv_lib_100ask

`lv_lib_100ask` 是基于 lvgl 库的各种开箱即用的方案参考或对 lvgl 库各种组件的增强接口。

源码仓库

- github: https://github.com/100askTeam/lv_lib_100ask
- gitee: https://gitee.com/weidongshan/lv_lib_100ask

2.15 联系我们

【1】韦东山老师官方在线学习平台

【2】单片机机工程师如何提升自己进阶嵌入式 Linux?

【3】访问 B 站主页查看更多精彩内容

【4】韦东山老师 CSDN 博客

【5】打开手机微信扫一扫，关注公众号【百问科技】，获取更多嵌入式系统干货文章

【6】打开手机微信扫一扫，微信学习小程序学习更多课程！

2.16 ? 加入技术交流群聊一起学习!

如果微信群过期或无法加入，请添加下面的微信备注：进 LVGL 微信群

索引

符号

L