User Guide

正点原子 littleVGL 开发指南

lv_canvas 画布

开发指南

正点原子

广州市星翼电子科技有限公司

修订历史

版本	日期	原因
V1.00	2020/05/01	第一次发布

lv_canvas 画布

1. 介绍

lv_canvas 是一个画布控件,你可以在它上面绘制任意图形,以及进行旋转等操作,但是此 lv_canvas 画布控件必须得依赖于一个 buffer 缓冲区,此 buffer 缓冲区必须得是全局的或者静态的,即只要保证在使用此画布控件期间,此 buffer 缓冲区不会被释放掉就可以了,可以通过 lv_canvas_set_buffer(canvas, buffer, width, height,cf)接口来给画布控件设置一个 buffer 缓冲区,此接口同时指定了画布绘图区域的宽和高,以及画布的颜色格式,注意了,这里传入的 width 和 height 参数是用来指定画布绘图区域的宽和高,并不等价于画布控件自身的宽和高,画布控件的宽和高是通过 lv_obj_set_size 接口来设置的,默认情况下,画布控件自身的大小是等于绘图区域的大小的,当设置画布控件的大小与绘图区域的大小不相等时,就会出现截取显示或重复填充显示等情况,为了方便大家理解,我们假定画布的绘图区域大小为 200*150,里面只绘制了一个”Hello world”文本和一个红色填充矩形,那么三种显示情况如下所示:

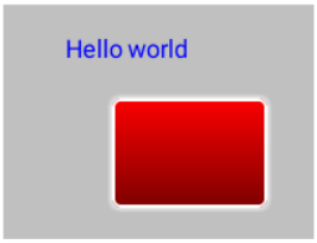

图 1.1 正常显示情况(画布控件的大小默认为 200*150)

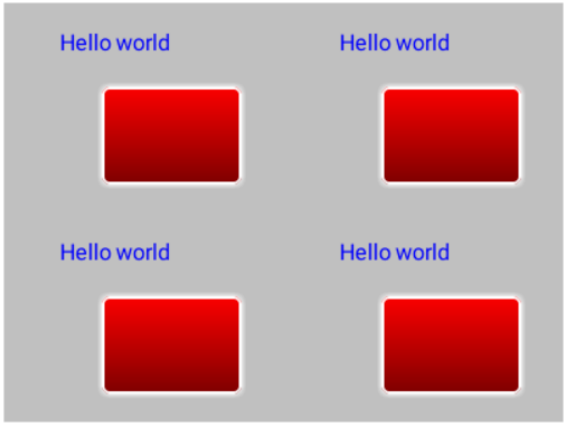

图 1.2 重复填充显示情况(画布控件的大小为 400*300)

图 1.3 截取显示情况(画布控件的大小为 100*75)

而最后的 cf 参数是用来指定画布颜色格式的,比如 LV_IMG_CF_TRUE_COLOR 等真彩色格式和 LV_IMG_CF_INDEXED_1BIT 等调色板格式,这里的颜色格式和图片在线转换工具中的颜色格式含义是一样的.当画布控件的颜色格式选择为调色板格式时,会比选择真彩色格

式多出一个步骤,因为调色板格式需要先构建出一个调色板,对于 LV_IMG_CF_INDEXED_1BIT 调色板格式,调色板中有 2 种颜色,对于 LV_IMG_CF_INDEXED_2BIT 调色板格式,调色板中有 4 种颜色,后面的调色板格式以此类推,所谓的构建调色板,就是对其里面的每一种颜色赋具体的颜色值,它是通过 lv_canvas_set_palette(canvas, id, color)接口来完成的,按理来说,调色板中有多少种颜色,那么此接口就需要被调用多少次,不论是真彩色格式还是调色板格式,最后都可以通过 lv_canvas_set_px(canvas, x, y,color)接口来设置任意像素点的颜色值,只不过当为调色板格式时,color 参数传入的是某颜色在调色板中的位置索引值,当为真彩色格式时,color 参数传入的才是真正的颜色值.

从图 1.1 中,我们可以看到绘图区域的背景颜色是银色的,其实它是通过 lv_canvas_fill_bg(canvas,color)接口来设置的,默认情况下,绘图区域背景是黑色的,除此之外,画布控件还给我们提供了一些其他的绘图 API 接口,如下所示:

1)在画布上绘制矩形

```
lv_canvas_draw_rect(canvas, x, y, width, heigth, &style);
```

2)在画布上绘制文本内容

```
lv_canvas_draw_text(canvas, x, y, max_width, &style, txt,
LV_LABEL_ALIGN_LEFT/CENTER/RIGTH);
```

3)在画布上绘制图片

```
lv_canvas_draw_img(canvas, x, y, &img_src, &style)
```

4)在画布上绘制线条

```
lv_canvas_draw_line(canvas, point_array, point_cnt, &style)
```

5)在画布上绘制多边形

```
lv_canvas_draw_polygon(canvas, points_array, point_cnt, &style)
```

6)在画布上绘制弧形

```
lv_canvas_draw_arc(canvas, x, y, radius, start_angle, end_angle, &style)
```

7)在画布上绘制一个经过旋转后的图片

```
lv_canvas_rotate(canvas, &img_dsc, angle, x, y, pivot_x, pivot_y)
```

2. lv_canvas 的 API 接口

2.1 主要数据类型

2.1.1 画布样式数据类型

```
enum {
    LV_CANVAS_STYLE_MAIN,
};
typedef uint8_t lv_canvas_style_t;
```

此样式中的 image.color 字段用来描述 LV_IMG_CF_ALPHA_...颜色格式下的基色,除此之外,基本用不到的,了解即可

2.2 API 接口

2.2.1 创建对象

```
lv_obj_t * lv_canvas_create(lv_obj_t * par, const lv_obj_t * copy);
```

参数:
par: 父对象
copy: 拷贝的对象,如果无拷贝的话,传 NULL 值

返回值:
返回创建出来的对象,如果返回 NULL 的话,说明堆空间不够了

2.2.2 设置缓冲区

```
void lv_canvas_set_buffer(lv_obj_t * canvas, void * buf, lv_coord_t w, lv_coord_t h, lv_img_cf_t cf);
```

参数:
canvas: 画布对象
buf: 缓冲区,必须得保证在画布使用期间,此缓冲区不能被释放了
w: 画布绘图区域的宽度
h: 画布绘图区域的高度
cf: 画布的颜色格式

请注意,上面的 w 和 h 是指绘图区域的宽和高,和画布控件自身的宽高并不是同一个概念,同时请保证 buf 缓冲区的大小和传入的 w, h 参数保持一致性,假定 buf 缓冲区的大小为 buf_size,那么它应该满足如下公式:

buf_size = LV_CANVAS_BUF_SIZE_...(w,h);

其中 LV_CANVAS_BUF_SIZE_...是一个和 cf 颜色格式相关的宏,此宏的所有可能值如下:

LV_CANVAS_BUF_SIZE_TRUE_COLOR
LV_CANVAS_BUF_SIZE_TRUE_COLOR_CHROMA_KEYED
LV_CANVAS_BUF_SIZE_TRUE_COLOR_ALPHA
LV_CANVAS_BUF_SIZE_INDEXED_1BIT
LV_CANVAS_BUF_SIZE_INDEXED_2BIT
LV_CANVAS_BUF_SIZE_INDEXED_4BIT
LV_CANVAS_BUF_SIZE_INDEXED_8BIT

假如 cf 参数为 LV_IMG_CF_TRUE_COLOR 时,那么 buf 缓冲区的定义应该如下所示:

```
static lv_color_t    buf[LV_CANVAS_BUF_SIZE_TRUE_COLOR(w, h)];
```

2.2.3 设置某像素的颜色值

```
void lv_canvas_set_px(lv_obj_t * canvas, lv_coord_t x, lv_coord_t y, lv_color_t c);
```

参数:

canvas: 画布对象
x: 像素的 x 坐标,是以画布控件的左上角为参考原点的
y: 像素的 y 坐标,是以画布控件的左上角为参考原点的
c: 像素的颜色值

请注意,当画布选择的 cf 颜色格式为 LV_IMG_CF_INDEXED_1BIT 等调色板格式时,c 参数传入的是某颜色在调色板中的位置索引值,当为 LV_IMG_CF_TRUE_COLOR 等真彩色格式时,c 参数传入的才是真正的颜色值.

2.2.4 构建调色板

```
void lv_canvas_set_palette(lv_obj_t * canvas, uint8_t id, lv_color_t c);
```

参数:

canvas: 画布对象
id: 给调色板中的哪一个位置赋颜色值,从 0 开始,id 最大值等于此调色板的颜色种类数减 1
c: 颜色值

只有当画布选择的 cf 颜色格式为 LV_IMG_CF_INDEXED_1BIT(有 2 种颜色), LV_IMG_CF_INDEXED_2BIT(有 4 种颜色), LV_IMG_CF_INDEXED_4BIT(有 16 种颜色), LV_IMG_CF_INDEXED_8BIT(有 256 种颜色)中的一个时,此 API 接口才有效,选择不同的调色板格式,其具有的颜色种类数也是不同的,如果想要实现绘制透明的效果,只需要给调色板中的某个位置赋一个 LV_COLOR_TRANS 透明色,然后在想要透明的地方,调用这种颜色就可以了

2.2.5 设置样式

```
void lv_canvas_set_style(lv_obj_t * canvas, lv_canvas_style_t type, const lv_style_t * style);
```

参数:
canvas: 画布对象
type: 设置哪一部分的样式,目前只有 LV_CANVAS_STYLE_MAIN 这一个可选值
style: 样式

2.2.6 获取画布所对应的图片源

```
lv_img_dsc_t * lv_canvas_get_img(lv_obj_t * canvas);
```

参数:
canvas: 画布对象

返回值:
返回画布所对应的图片源

拿到此图片源后,我们可以利用 lv_img 图片章节中的 lv_img_set_src 接口把此图片给显示出来,显示出来的效果和画布中的效果是一模一样的,例子如下所示:

```
lv_obj_t * img = lv_img_create(lv_scr_act(),NULL);//创建图片对象
lv_img_dsc_t * img_src = lv_canvas_get_img(canvas);//获取画布所对应的图片源
lv_img_set_src(img, img_src);//设置图片源
```

2.2.7 在画布中绘制矩形

```
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_style_t * style);
```

参数:
canvas: 画布对象
x: 矩形的 x 坐标,是以画布控件的左上角为参考原点的
y: 矩形的 y 坐标,是以画布控件的左上角为参考原点的
w: 矩形的宽度
h: 矩形的高度
style: 用于修饰矩形的样式

是否为填充的矩形,这取决于传入的 style 样式

2.2.8 在画布中绘制文本内容

```
void lv_canvas_draw_text(lv_obj_t * canvas, lv_coord_t x, lv_coord_t y, lv_coord_t max_w,
const lv_style_t * style, const char * txt, lv_label_align_t align);
```

参数:
canvas: 画布对象
x: 文本内容的 x 坐标,是以画布控件的左上角为参考原点的
y: 文本内容的 y 坐标,是以画布控件的左上角为参考原点的
max_w: 最大宽度,当文本内容的长度超过此大小时,会自动换行的
style: 用于修饰文本内容的样式
txt: 文本内容
align: 文本内容在指定的 max_w 宽度区域内的水平对齐方式

2.2.9 在画布中绘制图片

```
void lv_canvas_draw_img(lv_obj_t * canvas, lv_coord_t x, lv_coord_t y, const void * src,
const lv_style_t * style);
```

参数:
canvas: 画布对象
x: 图片的 x 坐标,是以画布控件的左上角为参考原点的
y: 图片的 y 坐标,是以画布控件的左上角为参考原点的
src: 图片源
style: 用于修饰图片的样式

2.2.10 在画布中绘制线条

```
void lv_canvas_draw_line(lv_obj_t * canvas, const lv_point_t * points, uint32_t point_cnt,
const lv_style_t * style);
```

参数:
canvas: 画布对象
points: 用于构成线条的点集合
point_cnt: 点的个数
style: 用于修饰线条的样式

2.2.11 在画布中绘制多边形

```
void lv_canvas_draw_polygon(lv_obj_t * canvas, const lv_point_t * points, uint32_t
point_cnt, const lv_style_t * style);
```

参数:
canvas: 画布对象
points: 用于构成多边形的点集合
point_cnt: 点的个数

style: 用于修饰多边形的样式

是否为填充的多边形,这取决于传入的 style 样式

2.2.12 在画布中绘制弧形

```
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_style_t * style);
```

参数:
canvas: 画布对象
x: 弧形的圆心 x 坐标,是以画布控件的左上角为参考原点的
y: 弧形的圆心 y 坐标,是以画布控件的左上角为参考原点的
r: 弧形的半径
start_angle: 弧形的起始角度,范围为[0,360]
end_angle: 弧形的终止角度,范围为[0,360]
style: 用于修饰弧形的样式

是以底边正中间点为 0 度,然后逆时针方向是增加度数,绘制弧形时,也是按逆时针方向从起点绘制到终点的

2.2.13 在画布中绘制旋转后的图片

```
void lv_canvas_rotate(lv_obj_t * canvas, lv_img_dsc_t * img, int16_t angle, lv_coord_t offset_x, lv_coord_t offset_y, int32_t pivot_x, int32_t pivot_y);
```

参数:
canvas: 画布对象
img: C 数组方式的图片源,对于图标字体,外部文件等方式的图片源是不支持的
angle: 顺时针旋转的角度
offset_x: 图片的 x 坐标,是以画布控件的左上角为参考原点的
offset_y: 图片的 y 坐标,是以画布控件的左上角为参考原点的
pivot_x: 旋转点的 x 坐标,是以画布控件的左上角为参考原点的
pivot_y: 旋转点的 y 坐标,是以画布控件的左上角为参考原点的

为了大家能更直观的理解,举一个例子,我们这里就直接把画布的绘图区域当成图片,旋转 60 度后,再绘制到画布中去,当然了,你也可以用图片在线转换工具来生成一张图片进行实验,示意代码如下:

```
#define CANVAS_WIDTH  200
#define CANVAS_HEIGHT  150
lv_color_t buf[LV_CANVAS_BUF_SIZE_TRUE_COLOR(CANVAS_WIDTH, CANVAS_HEIGHT)];
lv_obj_t * canvas = lv_canvas_create(lv_scr_act(), NULL);
lv_canvas_set_buffer(canvas, buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_IMG_CF_TRUE_COLOR);
```

```
lv_canvas_fill_bg(canvas,  LV_COLOR_GRAY);//将画布的背景清成灰色
lv_canvas_draw_rect(canvas, 50, 40, 100, 70, &style);//在画布中绘制一个红色的填充矩形
```

此时得到的效果如下图所示:

图 2.2.13.1 绘图区域未旋转之前的效果

```
//重点,接着把画布的绘图区域生成一个图片源
lv_color_t buf_tmp[CANVAS_WIDTH * CANVAS_HEIGHT];
memcpy(buf_tmp, buf, sizeof(buf_tmp));
lv_img_dsc_t img_src;
img_src.data = (void *)buf_tmp;
img_src.header.cf = LV_IMG_CF_TRUE_COLOR;
img_src.header.w = CANVAS_WIDTH;
img_src.header.h = CANVAS_HEIGHT;

lv_canvas_fill_bg(canvas, LV_COLOR_GRAY);//将画布的背景再次清成灰色
//选择 60 度后,再绘制到画布中去
lv_canvas_rotate(canvas, & img_src, 60, 0, 0, CANVAS_WIDTH / 2, CANVAS_HEIGHT /
2);
```

旋转后得到的效果如下图所示:

图 2.2.13.2 绘图区域旋转后的效果

2.2.14 将某个绘图缓冲区拷贝到画布中

```
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);
```

参数:

canvas: 画布对象
to_copy: 绘图缓冲区
x: 画布中的 x 坐标
y: 画布中的 y 坐标
w: to_copy 绘图缓冲区对应的宽度
h: to_copy 绘图缓冲区对应的高度

这个 API 接口的作用就是将 to_copy 绘图缓冲区拷贝到 canvas 画布中的(x,y)位置上去.但是有一个前提就是必须得保证 to_copy 绘图缓冲区的颜色格式和 canvas 画布的颜色格式保持一致,下面举一个简单的例子,示意代码如下所示:

```
//画布 1 绘图区域的大小
#define CANVAS1_WIDTH  100
#define CANVAS1_HEIGHT  100
//画布 2 绘图区域的大小
#define CANVAS2_WIDTH  150
#define CANVAS2_HEIGHT  150


//创建画布 1
lv_color_t buf1[LV_CANVAS_BUF_SIZE_TRUE_COLOR(CANVAS1_WIDTH, CANVAS1_HEIGHT)];
lv_obj_t * canvas1 = lv_canvas_create(lv_scr_act(), NULL);
lv_canvas_set_buffer(canvas1,buf1,CANVAS1_WIDTH,CANVAS1_HEIGHT, LV_IMG_CF_TRUE_COLOR);
//在画布 1 上绘制一个 Hello 文本内容
lv_canvas_draw_text(canvas1, 10, 10, CANVAS1_WIDTH, &style, "Hello", LV_LABEL_ALIGN_LEFT);

//创建画布 2
lv_color_t buf2[LV_CANVAS_BUF_SIZE_TRUE_COLOR(CANVAS2_WIDTH, CANVAS2_HEIGHT)];
lv_obj_t * canvas2 = lv_canvas_create(lv_scr_act(), NULL);
lv_canvas_set_buffer(canvas2,buf2,CANVAS2_WIDTH,CANVAS2_HEIGHT, LV_IMG_CF_TRUE_COLOR);
//把画布 1 的绘图缓冲区拷贝到画布 2 中的(20,20)坐标上
lv_canvas_copy_buf(canvas2, buf1,20,20, CANVAS1_WIDTH, CANVAS1_HEIGHT);
```

2.2.15 备注

还有几个 get 获取类型的 API 接口我这里就不列举出来了,比较简单的

3.例程设计

3.1 功能简介

先自定义一个样式,用来修饰画布中的各种绘图形状,然后接着创建 2 个画布控件,第一个画布控件主要是用来演示各种绘图接口的,比如画矩形,画文本内容,画弧形,画线条等,而第二个画布控件主要是用来演示调色板格式的.

3.2 硬件设计

本例程所用到的硬件有:

1) 液晶屏

3.3 软件设计

在 GUI_APP 目录下创建 lv_canvas_test.c 和 lv_ canvas _test.h 俩个文件,其中 lv_ canvas _test.c 文件的内容如下:

```
#include "lv_canvas_test.h"
#include "lvgl.h"

lv_style_t style;
const lv_point_t points[] = {{50,90},{70,50},{90,90}};//构成线条的点集合

//画布 1,用来演示一些绘图 API 接口
#define CANVAS1_WIDTH     100 //画布 1 的宽度
#define CANVAS1_HEIGHT    100 //画布 1 的高度
//因为内部 sram 不够用,所以我们把画布的缓存区定义在外部 ram 上,注意需要跳过 lvgl
//的帧缓存区
lv_color_t canvas1_buf[LV_CANVAS_BUF_SIZE_TRUE_COLOR(CANVAS1_WIDTH,CANVAS1_HEIGHT)]  __attribute__((at(0X68000000+LV_HOR_RES_MAX*LV_VER_RES_MAX*2)));

//画布 2,用来演示调色板格式
#define CANVAS2_WIDTH     40 //画布 2 的宽度
#define CANVAS2_HEIGHT    40 //画布 2 的高度
```

```
    //画布 2 比较小,所以它的缓冲区可以直接定义在内部 sram 上
    lv_color_t canvas2_buf[LV_CANVAS_BUF_SIZE_INDEXED_1BIT(CANVAS2_WIDTH,
CANVAS2_HEIGHT)];

    //例程入口
    void lv_canvas_test_start()
    {
        lv_obj_t * scr = lv_scr_act();//获取当前活跃的屏幕对象

        //1.创建样式
        lv_style_copy(&style,&lv_style_plain);
        style.body.main_color = LV_COLOR_RED;
        style.body.grad_color = LV_COLOR_RED;
        style.line.width = 2;
        style.line.color = LV_COLOR_GREEN;
        style.text.color = LV_COLOR_BLUE;

        //2.创建画布 1,用来演示一些绘图 API 接口
        lv_obj_t * canvas1 = lv_canvas_create(scr,NULL);
        lv_canvas_set_buffer(canvas1,canvas1_buf,CANVAS1_WIDTH,CANVAS1_HEIGHT,
LV_IMG_CF_TRUE_COLOR);//设置缓冲区,真彩色格式
        //设置与屏幕的对齐方式
        lv_obj_align(canvas1,NULL,LV_ALIGN_IN_TOP_MID,0,20);
        lv_canvas_fill_bg(canvas1,LV_COLOR_GRAY);//将背景清成灰色
        lv_canvas_draw_rect(canvas1,10,10,60,20,&style);//绘制一个红色的填充矩形
        lv_canvas_draw_text(canvas1,0,35,CANVAS1_WIDTH,&style,"Hello",LV_LABEL_A
LIGN_CENTER);//绘制文本内容
        lv_canvas_draw_arc(canvas1,30,60,20,270,90,&style);//绘制弧形
        //绘制线条
        lv_canvas_draw_line(canvas1,points,sizeof(points)/sizeof(lv_point_t),&style);

        //3.创建画布 2,用来演示调色板格式
        lv_obj_t * canvas2 = lv_canvas_create(scr,NULL);
        lv_canvas_set_buffer(canvas2,canvas2_buf,CANVAS2_WIDTH,CANVAS2_HEIGHT,
LV_IMG_CF_INDEXED_1BIT);//设置缓冲区,真彩色格式
        //设置与画布 1 的对齐方式
        lv_obj_align(canvas2,canvas1,LV_ALIGN_OUT_BOTTOM_MID,0,20);
        //构建调色板,对于 LV_IMG_CF_INDEXED_1BIT 颜色格式而言,总共有 2 种颜色
        lv_canvas_set_palette(canvas2,0,LV_COLOR_GREEN);//第一种为绿色
        //第二种为红色,也可以换成 LV_COLOR_TRANSP 透明色看看效果哦
        lv_canvas_set_palette(canvas2,1,LV_COLOR_RED);
        //定义 2 个颜色
        lv_color_t color0;
        lv_color_t color1;
```

```
    color0.full = 0;//指向调色板中的第一种颜色
    color1.full = 1;//指向调色板中的第二种颜色

      lv_canvas_fill_bg(canvas2,color0);//把背景填充成绿色
      //在画布 2 的正中间绘制一个 20*20 大小的矩形
      uint16_t x,y;
      for(y=10;y<30;y++)
      {
          for(x=10;x<30;x++)
          {
              lv_canvas_set_px(canvas2,x,y,color1);
          }
    }
}
```

3.4 下载验证

把代码下载进去之后,可以看到如下所示的初始界面效果:

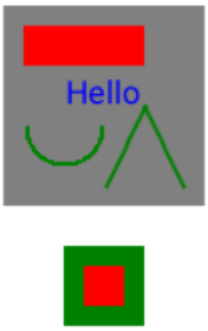

图 3.4.1 初始界面效果

4. 资料下载

正点原子公司名称 ：广州市星翼电子科技有限公司

LittleVGL 资料连接 ：www.openedv.com/thread-309664-1-1.html

原子哥在线教学平台：www.yuanzige.com

正点原子淘宝店铺 ：https://openedv.taobao.com

正点原子官方网站 ：www.alientek.com

正点原子 B 站视频 ：https://space.bilibili.com/394620890

电话：020-38271790 传真：020-36773971

请下载原子哥 APP，数千讲视频免费学习，更快更流畅。

请关注正点原子公众号，资料发布更新我们会通知。

扫码下载“原子哥”APP

扫码关注正点原子公众号