

正点原子 littleVGL 开发指南

lv_img 图片

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_img 图片

1. 介绍

1.1 理论介绍

lv_img 就是一个图片控件,它就是根据你传入的图片源来显示你想要的图片,littleVGL 为了提供最大的灵活性,它支持如下三种图片源方式:

- 1) 内部 C 数组,用 lv_img_dsc_t 结构体来进行描述
- 2) 外部存储文件,比如 SD 卡或者 U 盘上的图片文件
- 3) LV_SYMBOL_XXX 形式的图标字体或者文本,此时 lv_img 图片就相当于一个 lv_label 标签控件

如果你确定好图片源之后,就可以通过 lv_img_set_src(img, src)接口来显示此图片,此接口内部会自动判断出 src 是属于哪一种图片源方式,然后选择相应的解析程序把图片给显示出来.

本章我们主要讲解的是上面的第 1 种和第 3 种图片源方式,对于第 2 种图片源方式我们这里稍微介绍一下,如果想详细研究,请参考 GUNlv_examples\lv_tutorial\6_images 目录下的官方例程,我们先来介绍一下第 1 种内部 C 数组图片源方式,为了产生一个图片的 C 数组,我们必须得借助官方提供的图片在线转换工具(<https://littlevgl.com/image-to-c-array>),你可以选择一张 png, jpg 或者 bmp 格式的图片文件,通过此在线图片转换工具的转换,你可以得到一个.c 文件,在这个.c 文件中,它存放的是各个颜色深度下的图片像素数据,它是以数组方式存放的,它内部是通过宏预处理指令来保证只有和当前颜色深度匹配的那个像素数据数组才会生效,这些 C 数组将会保存到微处理器的内部 flash 上,所以请注意微处理器的 flash 容量大小是否能够满足你们的项目需求,对于这样产生的 C 数组,我们是不能直接使用的,littleVGL 会用 lv_img_dsc_t 结构体对其进行一次封装,比如这里以一张红花图片为例,封装之后的格式如下所示:

```
lv_img_dsc_t red_flower = {
    .header.always_zero = 0,
    .header.w = 100, //图片的宽度
    .header.h = 75, //图片的高度
    .data_size = 7500 * LV_COLOR_SIZE / 8, //C 数组的大小,单位为字节
    .header.cf = LV_IMG_CF_TRUE_COLOR, //图片的转换格式
    .data = red_flower_map, //C 数组,也就是图片的核心像素数据
};
```

对于 red_flower 是不需要去修改的,它是根据图片在线转换工具中的相应配置来自动生成的,在有了这个 lv_img_dsc_t 结构体之后,我们就可以直接使用这个图片了,只不过你还需要在使用它的地方先用 LV_IMG_DECLARE(red_flower)宏来申明一下此图片,具体调用方式如下所示:

```
LV_IMG_DECLARE(red_flower);//先申明此图片
lv_img_set_src(img, &red_flower);//然后显示此图片
```

接着我们来稍微介绍一下第 2 种外部存储文件图片源方式,它是把图片数据文件放到了外部的储存介质上,比如 SD 卡或者 U 盘上,所以这里你必须得另外用到 littleVGL 的文件系统模块,然后对于这个外部存储文件的格式这里又可以分为俩大类,一类是图片的最原始格式,如.png 文件,它不需要经过任何转换,你只需要把这张.png 图片直接放到外部的存储介质上就可以了,但是对于这种方式,littleVGL 是不能直接支持的,你必须得外加实现 png 图片的解析库,可以参考官方的 https://blog.littlevgl.com/2018-10-05/png_converter 资料,对于另外一类是.bin 文件格式,它是需要经过在线图片转换工具的转换,拿到转换后的.bin 文件后,你就可以调用 lv_img_set_src(img, "S:folder1/my_img.bin")接口把图片给显示出来了,不过前提就是你得先实现 littleVGL 的文件系统驱动哦!

最后我们来说一下第 3 种图片源方式,当你使用此方式时,lv_img 控件其实就是一个 lv_label 标签控件,它就是用来显示图标字体或文本的,比如当你想要显示一个 OK 图标字体时,你可以直接调用 lv_img_set_src(img, LV_SYMBOL_OK)接口进行显示,或者调用 lv_img_set_src(img, LV_SYMBOL_OK "OK")接口来和文本进行混合显示,当你如果只想显示文本或者显示以文本开头的内容时,内容最前面你必须得加上 LV_SYMBOL_DUMMY 标记符,这个 LV_SYMBOL_DUMMY 宏的作用就是用来让系统能够正确的识别这是第 3 种图片源方式的,无其他任何作用,正确的调用应如下:

```
lv_img_set_src(img, LV_SYMBOL_DUMMY "Hello"); //只显示 Hello 文本
lv_img_set_src(img, LV_SYMBOL_DUMMY "OK" LV_SYMBOL_OK); //显示 OK 文本和 OK 图标
```

用一句话来高度总结就是:当你显示的内容是以文本开头时,你就必须得在最前面加上 LV_SYMBOL_DUMMY 宏来作为标记.

接着我们来说一下图片如何实现带有透明度显示,比如实现背景透明显示等,除了采用第 2 种图片源方式中的 png 解析库外,lv_img 中还支持如下俩种透明度方式:

Chrome keying 方式: 当图片的背景色与 lv_conf.h 中的 LV_COLOR_TRANSP 宏所定义的颜色相同时,那么图片中的背景色将不会被绘制出来,相当于实现了背景透明显示,比如把 LV_COLOR_TRANSP 宏的值给配置为纯绿色,然后原始图片如图 1.1 所示,最终的显示效果如图 1.2 所示

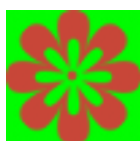


图 1.1.1 纯绿色背景的原始图片



图 1.1.2 最终的背景透明显示效果

Alpha byte 方式: 每一个像素增加一个额外的 alpha 透明度字节,此种方式可以实现任意地方透明显示,但是缺点就是存储空间会增加,而且渲染速度也会变慢

lv_img 图片控件支持在运行时进行图片重绘色,或者说是颜色混合,它是将图片中的每一个像素点与指定的颜色按照指定的强度进行混合操作,这里的颜色是通过样式中的 image.color 字段来指定,强度是通过样式中的 image.intense 字段来指定的,当 image.intense 字段的值为 0 时,代表不使能图片重绘色功能,否则就是使能了重绘色功能.

lv_img 控件还有一个大小自动适配的功能,这是通过 lv_img_set_auto_size(img, true)接口来使能的,不过默认情况下就是使能的,所谓的大小自动适配就是指图片为多大,那么 lv_img 控件就为多大,当不使能大小自动适配,而且 lv_img 控件的大小比图片的大小还要大时,那么

在剩下的空间处,图片将会被重复的绘制,就跟贴瓦片一样,利用这个特性,我们可以用一张很窄的图片来实现一个全屏的背景壁纸,就如下面的效果所示:




图 1.1.3 图片重复绘制效果

其中上面的一张红花图片的大小为 100x75,而 lv_img 控件的大小为 200*150,正好可以重复绘制 4 张图片,有些时候,我们可能想调整图片在 lv_img 控件中的显示位置,它可以通过 lv_img_set_offset_x(img, x_ofs)接口和 lv_img_set_offset_y(img, y_ofs)接口来完成.

1.2 图片在线转换工具的使用

在本章中,学会图片在线转换工具的使用也是非常重要的,我们打开浏览器,输入 <https://littlevgl.com/image-to-c-array> 网址,对于国内用户来说,速度可能有点感人,没办法,大家忍忍吧,打开之后,可以看到如下所示的界面:

 LittlevGL
 GET STARTED
 LIVE DEMOS
 DOCUMENTATION
 TOOLS
 DOWNLOAD
 FORUM
 B

How to use the generated file in LittlevGL?

- For C arrays
 - Copy the result C file into your LittlevGL project
 - In a C file of your application declare the image as: `LV_IMG_DECLARE(my_image_name);`
 - Set the image for an `lv_img` object: `lv_img_set_src(img1, &my_image_name);`
- For external binary files (e.g. SD card)
 - Set up a new driver. To learn more read the [Tutorial](#).
 - Set the image for an `lv_img` object: `lv_img_set_src(img1, "S:/path/to/image");`

Image file

浏览...

Name

Name of output .C files (E.g: red_flower)

Color format

True color

Alpha byte Add a 8 bit Alpha value to every pixel

Chroma keyed Make LV_COLOR_TRANSP (lv_conf.h) pixels to transparent

Output format

C array

Dithering

☐ Dithering of True color images

Convert

Old version for v5.1.1

图 1.2.1 图片在线转换工具的界面

整体上来说,此工具还是比较简单的,我们现在来介绍一下每一个配置项的含义.

Image file: 从你的 PC 电脑上选择你想要转换的图片

Name: 生成的.c 文件名称

Color format: 颜色格式,有如下 14 个选项值,可以分为 4 类

[第 1 类,真彩色格式]

这类格式的好处就是显示出来的图片不失真,但是占用存储空间大

True color: 真彩色格式

True color with alpha: 带有 alpha 透明度的真彩色格式

True color chroma keyed: 带有 chroma keyed 透明度的真彩色格式

[第 2 类,调色板格式]

这类格式的好处就是占用存储空间小,缺点就是不能显示太鲜艳的图片

Indexed 2 colors: 当图片中的颜色种类不超过 2 种时,可以采用此格式

Indexed 4 colors: 当图片中的颜色种类不超过 4 种时,可以采用此格式

Indexed 16 colors: 当图片中的颜色种类不超过 16 种时,可以采用此格式

Indexed 256 colors: 当图片中的颜色种类不超过 256 种时,可以采用此格式

[第 3 类,纯阴影格式]

这类格式中的像素点只有 alpha 通道的数据,没有 R,G,B 等三个颜色通道的数据

Alpha only 2 shades: alpha 通道占 1 位,每一个像素有 2 种阴影等级

Alpha only 4 shades: alpha 通道占 2 位,每一个像素有 4 种阴影等级

Alpha only 16 shades: alpha 通道占 4 位,每一个像素有 16 种阴影等级

Alpha only 256 shades: alpha 通道占 8 位,每一个像素有 256 种阴影等级

[第 4 类,原始数据格式]

当你用到外部 png 解析库时,才会用到此类格式,它可以转换得到 png 图片的原始数据,而且是以 C 数组的方式存储在微处理器的内部 flash 上,如果你是把 png 图片直接放到外部存储介质上,如 SD 卡,U 盘等,那么你就用不到此类格式了

Raw: 原始数据格式

Raw with alpha: 带有 alpha 透明度的原始数据格式

Raw as chroma keyed: 带有 chroma keyed 透明度的原始数据格式

Output format: 输出格式,会根据选择的颜色格式不同出现不同的选项值,但整体上来说,它具有如下 5 个选项值

C array: 输出.c 文件,即 C 数组格式

Binary RGB332: 输出.bin 文件,即二进制格式,颜色为 RGB332 格式

Binary RGB565: 输出.bin 文件,即二进制格式,颜色为 RGB565 格式

Binary RGB565 Swap: 输出.bin 文件,即二进制格式,颜色为 RGB565 Swap 格式

Binary RGB888: 输出.bin 文件,即二进制格式,颜色为 RGB888 格式

Binary: 输出.bin 文件,即二进制格式,颜色格式自动确定

Dithering: 是否使能真彩色图片的颜色抖动显示,使能之后,转换得到的数据是已经经过抖动处理了的,所以它是不会增加运行时开销的,而 Dithering 抖动显示技术的好处就是它欺骗你的眼睛,使用有限的色彩让你看到比实际图象更多色彩的显示方式,通过在相邻像素间随机的加入不同的颜色来修饰图象,通常这种方式被用于颜色较少的情况下

Convert: 当你的所有配置项都设置好后,你可以点击此按钮来进行转换

2. lv_img 的 API 接口

2.1 主要数据类型

2.1.1 图片样式数据类型

```
enum {  
    LV_IMG_STYLE_MAIN,  
};  
typedef uint8_t lv_img_style_t;
```

这里面就一种样式,比较简单,我们主要是来介绍一下此样式里某些字段的含义
image.color 字段: 重绘色时的混合颜色或者图标字体和文本的颜色
image.intense 字段: 重绘色时的混合强度,当此值为 0 时,代表不使能图片重绘色功能
image.opa 字段: 图片的整体透明度

2.2 API 接口

2.2.1 创建对象

```
lv_obj_t * lv_img_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 设置图片源

```
void lv_img_set_src(lv_obj_t * img, const void * src_img);
```

参数:

img: 图片对象

src_img: 图片源,有 3 种方式,详情请看 1.1 理论介绍

2.2.3 设置大小自动适配

```
void lv_img_set_auto_size(lv_obj_t * img, bool autosize_en);
```

参数:

img: 图片对象

autosize_en: 是否使能大小自适应

当使能之后,lv_img 控件的大小将会根据所显示图片的大小来自动确定

2.2.4 设置 x 轴上的偏移量

```
void lv_img_set_offset_x(lv_obj_t * img, lv_coord_t x);
```

参数:

img: 图片对象

x: x 轴上的偏移量

这是设置图片在 lv_img 控件中的 x 轴偏移量

2.2.5 设置 y 轴上的偏移量

```
void lv_img_set_offset_y(lv_obj_t * img, lv_coord_t y);
```

参数:

img: 图片对象

y: y 轴上的偏移量

这是设置图片在 lv_img 控件中的 y 轴偏移量

2.2.6 设置样式

```
static inline void lv_img_set_style(lv_obj_t * img, lv_img_style_t type, const lv_style_t * style);
```

参数:

img: 图片对象

type: 设置哪一部分的样式,目前就只有 LV_IMG_STYLE_MAIN 这一个可选值

style: 样式

默认情况下,lv_img 控件的样式是为 NULL 的,即从父对象上继承样式

2.2.7 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

创建一个自定义样式来修饰图片对象,然后接着创建 4 个 lv_img 图片对象,第 1 个图片对象用来显示图标字体和文本,第 2 个图片对象用来显示 True color 颜色格式的图片,第 3 个图片对象用来显示 True color with alpha 颜色格式的图片,第 4 个图片对象用来显示 True color chroma keyed 颜色格式的图片,并设置第 4 个图片对象不使能大小自动适配功能,设置其固定大小为 150*50,它的宽度是当前显示图片的 3 倍,高度相等,目的是为了给大家演示图片重复绘制现象,当按下 KEY0 按键时,会将样式中的 image. Intense 字段值加 10,当按下 KEY1 按键时,会将样式中的 image.opa 字段值减 10,这样大家就可以清楚地看到图片重绘色现象和透明度变化现象.然后本例程需要三张图片素材,我已经把它们放到了资料目录下,如下图所示:

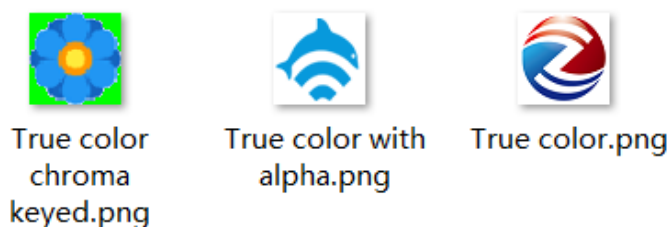


图 3.1.1 图片素材

这三张图片的大小都是 50*50 的,在背景上各有要求,分来用来演示不同的颜色格式效果, True color chroma keyed.png 图片是用来演示 True color chroma keyed 颜色格式的,所以他的背景色为纯绿色,而 True color with alpha.png 图片是用来演示 True color with alpha 颜色格式的,所以它的背景为透明,而 True color.png 图片是用来演示 True color 颜色格式的,所以它的背景不透明,为纯白色,而这三张图片在使用图片在线转换工具时,对于 Color format 配置项,分别选择它们对应的值,对于 Output format 配置项,它们都固定为”C array”选项值,对于 Dithering 配置项,它们都不使能.

3.2 硬件设计

本例程所用到的硬件有:

- 1) 液晶屏
- 2) KEY0,KEY1 按键

3.3 软件设计

在 GUI_APP 目录下创建 lv_img_test.c 和 lv_img_test.h 两个文件,其中 lv_img_test.c 文件的内容如下:


```
#include "lv_img_test.h"
#include "lvgl.h"
#include "key.h"

lv_style_t img_style;

//图片申明
LV_IMG_DECLARE(true_color);
LV_IMG_DECLARE(true_color_with_alpha);
LV_IMG_DECLARE(true_color_chroma_keyed);

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

    //1.创建图片的样式
    lv_style_copy(&img_style,&lv_style_transp);
    img_style.image.color = LV_COLOR_RED;//图片重绘色时的混合颜色或者文本的颜色
    img_style.image.intense = 0;//暂时不使能重绘色功能
    img_style.image.opa = LV_OPA_COVER;//透明度

    //2.创建显示图标字体和文本的图片对象
    lv_obj_t * img1 = lv_img_create(scr,NULL);
    //以文本开头,前面必须得加 LV_SYMBOL_DUMMY
    lv_img_set_src(img1,LV_SYMBOL_DUMMY"Icon font: \"LV_SYMBOL_AUDIO\" audio");
    lv_img_set_style(img1,LV_IMG_STYLE_MAIN,&img_style);//设置样式
    lv_obj_align(img1,NULL,LV_ALIGN_IN_TOP_MID,0,10);

    //3.创建显示 True color 格式的图片对象
    lv_obj_t * img2 = lv_img_create(scr,NULL);
    lv_img_set_src(img2,&true_color);
    lv_img_set_style(img2,LV_IMG_STYLE_MAIN,&img_style);//设置样式
    lv_obj_align(img2,img1,LV_ALIGN_OUT_BOTTOM_MID,0,10);

    //4.创建显示 True color with alpha 格式的图片对象
    lv_obj_t * img3 = lv_img_create(scr,NULL);
    lv_img_set_src(img3,&true_color_with_alpha);
    lv_img_set_style(img3,LV_IMG_STYLE_MAIN,&img_style);//设置样式
    lv_obj_align(img3,img2,LV_ALIGN_OUT_BOTTOM_MID,0,10);

    //5.创建显示 True color chroma keyed 格式的图片对象
```

```
//同时需要把 lv_conf.h 中 LV_COLOR_TRANSP 宏的值改为 LV_COLOR_GREEN 纯
//绿色,目的就是保持了和此图片的背景色一致,
//不过 LV_COLOR_TRANSP 的默认值就是纯绿色的
lv_obj_t * img4 = lv_img_create(scr,NULL);
lv_img_set_src(img4,&true_color_chroma_keyed);
lv_img_set_style(img4,LV_IMG_STYLE_MAIN,&img_style);//设置样式
lv_img_set_auto_size(img4,false);//不使能大小自动适配

//设置 lv_img 控件的宽度是 true_color_chroma_keyed 图片宽度的 3 倍,高度相等,可以
//看到在水平方向上有图片重复绘制现象
lv_obj_set_size(img4,150,50);
lv_obj_align(img4,img3,LV_ALIGN_OUT_BOTTOM_MID,0,10);

}

//按键处理
void key_handler()
{
    u8 key = KEY_Scan(0);

    if(key==KEY0_PRES)
    {
        //修改样式中的重绘色强度
        img_style.image.intense += 10;
        if(img_style.image.intense>=250)
            img_style.image.intense = 0;
        lv_obj_report_style_mod(&img_style);//刷新使用了此样式的对象
    }else if(key==KEY1_PRES)
    {
        //修改样式中的透明度
        img_style.image.opa -= 10;
        if(img_style.image.opa<=10)
            img_style.image.opa = LV_OPA_COVER;
        lv_obj_report_style_mod(&img_style);//刷新使用了此样式的对象
    }
}
```

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

然后我们可以多按几下 KEY0 键,来观察图片重绘色效果,如下图所示:

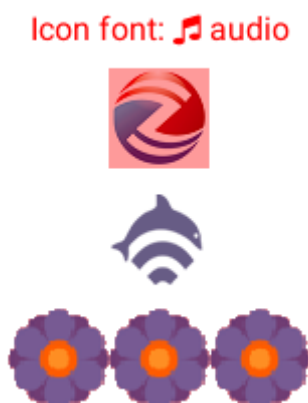


图 3.4.2 图片重绘色效果

当然了,你也可以多按几下 KEY1 键,来观察图片透明度变化的效果

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