User Guide

正点原子 littleVGL 开发指南

lv_mbox 消息对话框

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_mbox 消息对话框

1. 介绍

lv_mbox 控件是一个消息对话框,跟我们平常所见到的消息对话框有一点不同,那就是它没有专门的标题部分,不过我们可以通过往消息内容中加入\n 换行符来间接的实现标题部分,其实 lv_mbox 控件是由 lv_cont 容器,lv_label 标签,lv_btnm 矩阵按钮三个控件构成的,我们之前已经学习过了这三个控件的使用,所以再来学习 lv_mbox 就比较简单了,其中 lv_cont 容器是用来充当 lv_mbox 控件的背景,lv_label 标签是用来显示 lv_mbox 的消息内容,而 lv_btnm 矩阵按钮是用来充当 lv_mbox 的底部按钮栏,具体示意图如下所示:

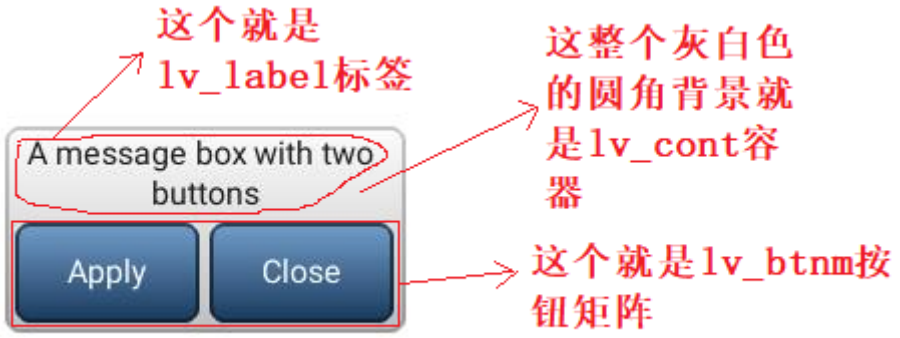

图 1.1 lv_mbox 控件的构成

因为我们的 lv_mbox 内部包含了 lv_cont 容器,而容器是具有大小自适应特性的,所以我们的 lv_mbox 控件在垂直方向上表现出了 LV_FIT_TIGHT 的自适应方式,即其宽度可以设置成固定值,而其高度会随着内部内容的大小而变化,而且其内部的 lv_label 标签固定为 LV_LABEL_LONG_MODE_BREAK 长文本模式,我们可以通过 lv_mbox_set_text(mbox, "Message")接口来设置 lv_mbox 的消息内容,通过 lv_mbox_add_btns(mbox, btn_str)接口来设置其底部按钮栏,当其中的按钮被点击时,它会跟 lv_btnm 矩阵按钮一样给它的事件回调函数发送 LV_EVENT_VALUE_CHANGED 事件,被点击按钮的 id 号会被当成事件自定义参数给传递过去,当 lv_mbox 控件被创建出来之后,默认就是处于打开状态的,我们可以通过 lv_mbox_start_auto_close(mbox, delay)接口在指定的 delay 时间之后自动关闭此消息对话框,而且在关闭时会有一种动画效果,至于动画时间的长短我们是可以通过 lv_mbox_set_anim_time(mbox, anim_time)接口来设置的.

2. lv_mbox 的 API 接口

2.1 主要数据类型

2.1.1 消息对话框样式数据类型

```
enum {
    LV_MBOX_STYLE_BG,
    LV_MBOX_STYLE_BTN_BG,
    LV_MBOX_STYLE_BTN_REL,
    LV_MBOX_STYLE_BTN_PR,
    LV_MBOX_STYLE_BTN_TGL_REL,
    LV_MBOX_STYLE_BTN_TGL_PR,
    LV_MBOX_STYLE_BTN_INA,
};
typedef uint8_t lv_mbox_style_t;
```

别看它具有 7 种样式,最后面的 6 种样式是用来修饰其内部的 lv_btnm 矩阵按钮的,使用方法和 lv_btnm 矩阵按钮章节中的内容是一样的,这里不过多介绍了.

LV_MBOX_STYLE_BG: 修饰其背景和消息内容的,即修饰其内部的 lv_cont 容器和 lv_label 标签的,使用样式中的 body 字段来修饰背景,使用样式中的 text 字段来修饰消息内容,默认值为 lv_style_pretty
LV_MBOX_STYLE_BTN_BG: 修饰矩阵按钮的背景,默认值为 lv_style_trans

2.2 API 接口

2.2.1 创建对象

```
lv_obj_t * lv_mbox_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 设置底部按钮栏

```
void lv_mbox_add_btns(lv_obj_t * mbox, const char ** btn_mapaction);
```

参数:
mbox: 消息框对象
btn_mapaction: 按钮映射表,必须以””空字符串结尾,如下所示:

```
const char * const btn_mapaction [] = {"ok", “\n” ,"close", ""};
```

另外还得保证此 btn_mapaction 是静态的或者全局的

此接口的使用方法和矩阵按钮中的 lv_btnm_set_map(const lv_obj_t * btnm, const char * map[])接口的使用方法是一样的

2.2.3 设置消息内容

```
void lv_mbox_set_text(lv_obj_t * mbox, const char * txt);
```

参数:
mbox: 消息框对象
txt: 消息内容

我们前面说过此消息对话框是没有专门的标题部分,但是我们可以通过此接口和\n 换行符来间接的实现标题部分,示意代码如下:

```
//我们自己定义一个接口,来设置其内部的消息内容标签是否使能文本重绘色功能
//因为 littleVGL 没有提供这样的接口,所以我们得自己来实现
void mbox_set_msg_recolor(lv_obj_t * mbox,bool en)
{
    lv_mbox_ext_t * ext = lv_obj_get_ext_attr(mbox);//获取控件的扩展字段
    lv_label_set_recolor(ext->text,en);//ext->text 就是消息对话框内部的标签对象
}

mbox_set_msg_recolor(mbox1,true);//使能消息内容的重绘色功能,这样标题就可以加色
//设置消息内容,附带设置标题
lv_mbox_set_text(mbox1,”#FF0000 Title#\nThis is a message”);
```

图 2.2.3.1 标题效果

2.2.4 设置动画时长

```
void lv_mbox_set_anim_time(lv_obj_t * mbox, uint16_t anim_time);
```

参数:
mbox: 消息框对象
anim_time: 动画时长,单位为 ms,当设置为 0 时代表无动画效果

这个是设置消息对话框被关闭时所做动画的时长,经笔者测试,发现此 API 接口好像不起作用

2.2.5 设置消息对话框自动关闭

```
void lv_mbox_start_auto_close(lv_obj_t * mbox, uint16_t delay);
```

参数:
mbox: 消息框对象
delay: 定时多少时间之后自动关闭此消息对话框,单位为 ms,当设置为 0 时代表立即关闭

当此消息对话框被关闭时,其所占的资源也会被全部删除,相当于执行了 lv_obj_del(mbox)操作

2.2.6 取消消息对话框的自动关闭

```
void lv_mbox_stop_auto_close(lv_obj_t * mbox);
```

参数:
mbox: 消息框对象

在消息对话框还未被自动关闭之前,你可以通过此接口来取消自动关闭功能

2.2.7 设置样式

```
void lv_mbox_set_style(lv_obj_t * mbox, lv_mbox_style_t type, const lv_style_t * style);
```

参数:
mbox: 消息框对象
type: 设置哪一部分的样式,有如下 7 个可选值:
LV_MBOX_STYLE_BG: 用来修饰背景和消息内容的
LV_MBOX_STYLE_BTN_BG //下面这 6 个都是用来修饰其内部矩阵按钮的
LV_MBOX_STYLE_BTN_REL,
LV_MBOX_STYLE_BTN_PR,
LV_MBOX_STYLE_BTN_TGL_REL,
LV_MBOX_STYLE_BTN_TGL_PR,
LV_MBOX_STYLE_BTN_INA,
style: 样式

2.2.8 是否使能底部按钮栏的文本重绘色

```
void lv_mbox_set_recolor(lv_obj_t * mbox, bool en);
```

参数:
mbox: 消息框对象
en: 是否使能

这其实就是在设置其内部的矩阵按钮是否使能文本重绘色功能,说的再直白点,就是底部按钮上的文本内容是否支持文本重绘色,注意此 API 接口必须得放在 lv_mbox_add_btns 接口的后面进行调用

2.2.9 获取当前被点击的按钮 id

```
uint16_t lv_mbox_get_active_btn(lv_obj_t * mbox);
```

参数:
mbox: 消息框对象

返回值:
返回当前被点击的按钮 id,如果获取失败,则返回 LV_BTNM_BTN_NONE

此接口的用法和矩阵按钮中的 lv_btnm_get_active_btn 接口的用法是一样,一般放在事件回调函数中进行调用

2.2.10 获取内部的矩阵按钮对象

```
lv_obj_t * lv_mbox_get_btnm(lv_obj_t * mbox);
```

参数:
mbox: 消息框对象

返回值:
返回其内部的矩阵按钮对象

拿到此矩阵按钮对象之后,我们就可以通过矩阵按钮对象自己专有的 API 接口来设置其特性了,注意此 API 接口必须得放在 lv_mbox_add_btns 接口的后面进行调用,否则会返回 NULL

2.2.11 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 4 种样式来修饰消息对话框,然后接着创建一个按钮,此按钮的作用就是用来再次打开消息对话框的,最后接着创建一个消息对话框,为其设置了消息内容,在消息内容中附带实现了标题的功能,然后为其设置了 Apply 和 Close 俩个按钮,当点击 Apply 按钮时,会马上把此消息对话框给关闭掉,当点击 Close 按钮时,会定时 2 秒后把此消息对话框给关闭掉.

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_mbox_test.c 和 lv_mbox_test.h 俩个文件,其中 lv_mbox_test.c 文件的内容如下:

```
#include "lv_mbox_test.h"
#include "lvgl.h"


lv_style_t bg_style;
lv_style_t btnm_bg_style;
lv_style_t btn_rel_style;
lv_style_t btn_pr_style;
lv_obj_t * open_btn;
lv_obj_t * mbox1;
const char * const btns_map[] ={"#5FB878 Apply#","\n","#ff0000 Close#",""};

lv_obj_t * mbox_create(lv_obj_t * parent);//函数申明



//事件回调函数
void event_handler(lv_obj_t * obj,lv_event_t event)
{
```

```
    uint16_t btn_id;
    if(obj==open_btn)//是打开按钮
    {
        if(event==LV_EVENT_RELEASED)
        {
            //重新在创建一个消息对话框
            mbox1 = mbox_create(lv_scr_act());
        }
    }else if(obj==mbox1)//是消息对话框
    {
            if(event==LV_EVENT_VALUE_CHANGED)
            {
                //获取按钮 id
                //也可以使用 btn_id = *((uint16_t*)lv_event_get_data())的方式
                btn_id = lv_mbox_get_active_btn(obj);

                if(btn_id==0)//Apply 按钮
                {
                    //马上关闭消息对话框
                    //也可以使用 lv_obj_del(obj);只不过 lv_obj_del 是立即关闭,而且无关
                    //闭动画效果
                    lv_mbox_start_auto_close(obj,0);
                }else if(btn_id==1)//Close 按钮
                {
                    //定时关闭消息对话框
                    lv_mbox_start_auto_close(obj,2000);//定时 2 秒
                }
            }
    }
}

//我们自己定义一个接口,来设置其内部的消息内容标签是否使能文本重绘色功能
//因为 littleVGL 没有提供这样的接口,所以我们得自己来实现
void mbox_set_msg_recolor(lv_obj_t * mbox,bool en)
{
    lv_mbox_ext_t * ext = lv_obj_get_ext_attr(mbox);//获取控件的扩展字段
    lv_label_set_recolor(ext->text,en);//ext->text 就是消息对话框内部的标签对象
}

//创建自己封装之后的消息对话框
//parent:父对象
//返回值: 返回创建出来的消息对话框对象
lv_obj_t * mbox_create(lv_obj_t * parent)
{
```

```
        #define MBOX_WIDTH              220   //消息对话框的宽度
        #define MBOX_BTN_HEIGHT     30     //其内部每个按钮的高度

        //其内部含有多少个按钮,我们这里只有 Apply 和 Close 俩个按钮
        #define MBOX_BTN_NUM          2

        lv_obj_t * mbox = lv_mbox_create(parent,NULL);//创建消息对话框

        //使能消息内容的文本重绘色,这样就可以对标题进行加色区分了
        mbox_set_msg_recolor(mbox,true);

        //设置消息内容,附带设置标题,同时对标题重绘色
        lv_mbox_set_text(mbox,"#007AFF Title#\nThis is a message");
        lv_mbox_add_btns(mbox,(const char**)btns_map);//设置按钮映射表
        lv_mbox_set_recolor(mbox,true);//使能其内部按钮的文本重绘色功能
        lv_obj_set_width(mbox,MBOX_WIDTH);//设置固定的宽度,高度会自适应的
        lv_obj_align(mbox,NULL,LV_ALIGN_CENTER,0,0);//设置与父对象居中对齐
        lv_obj_set_event_cb(mbox,event_handler);//设置事件回调函数

        //设置消息对话框的背景样式
        lv_mbox_set_style(mbox,LV_MBOX_STYLE_BG,&bg_style);

        //设置其内部矩阵按钮的背景样式
        lv_mbox_set_style(mbox,LV_MBOX_STYLE_BTN_BG,&btnm_bg_style);

        //设置按钮释放状态的样式
        lv_mbox_set_style(mbox,LV_MBOX_STYLE_BTN_REL,&btn_rel_style);

        //设置按钮按下状态的样式
        lv_mbox_set_style(mbox,LV_MBOX_STYLE_BTN_PR,&btn_pr_style);

        //最好在消息对话框全部初始化完成之后,再来设置其内部矩阵按钮的各种特性,否
        //则有可能得不到预期的效果
        lv_obj_t * btnm_of_mbox = lv_mbox_get_btnm(mbox);//获取其内部的矩阵按钮对象

        //设置矩阵按钮的大小
lv_obj_set_size(btnm_of_mbox,MBOX_WIDTH,MBOX_BTN_HEIGHT*MBOX_BTN_NUM);

        return mbox;
    }


    //例程入口
    void lv_mbox_test_start()
```

```
{
    lv_obj_t * scr = lv_scr_act();//获取当前活跃的屏幕对象

  //1.创建 4 种样式
  //1.1  创建消息对话框的背景样式
  lv_style_copy(&bg_style,&lv_style_plain_color);
  bg_style.body.main_color = LV_COLOR_MAKE(250,250,250);//背景颜色
  bg_style.body.grad_color = bg_style.body.main_color;
  bg_style.body.radius = 10;//圆角半径
  bg_style.body.border.width = 1;//边框宽度
  bg_style.body.border.color = LV_COLOR_MAKE(150,150,150);//边框颜色
  bg_style.body.shadow.color = bg_style.body.border.color;//阴影颜色
  bg_style.body.shadow.width = 6;//阴影的宽度

  //设置其内部的消息内容与消息对话框上边框之间的距离
  bg_style.body.padding.top = 10;

  //设置其内部的矩阵按钮与消息对话框底边框之间的距离
  bg_style.body.padding.bottom = 0;
  bg_style.body.padding.inner = 10;//设置消息内容与矩阵按钮之间的距离
  bg_style.text.color = LV_COLOR_BLACK;//消息内容的文本颜色

  //1.2  创建矩阵按钮的背景样式
  lv_style_copy(&btnm_bg_style,&lv_style_transp_tight);

//设置各种内边距全部为 0,其实 lv_style_transp_tight 样式默认就是各种内边距全为 0 的
  btnm_bg_style.body.padding.top = 0;
  btnm_bg_style.body.padding.left = 0;
  btnm_bg_style.body.padding.right = 0;
  btnm_bg_style.body.padding.bottom = 0;
  btnm_bg_style.body.padding.inner = 0;

  //1.3  创建按钮释放状态的样式
  lv_style_copy(&btn_rel_style,&lv_style_transp);
  btn_rel_style.body.border.part = LV_BORDER_TOP;//只绘制上边框
  btn_rel_style.body.border.width = 1;//边框的宽度
  btn_rel_style.body.border.color = bg_style.body.border.color;//边框的颜色

  //1.4  创建按钮按下状态的样式
  lv_style_copy(&btn_pr_style,&btn_rel_style);
  btn_pr_style.body.opa = LV_OPA_COVER;//完全不透明
  btn_pr_style.body.border.part = LV_BORDER_FULL;//绘制四边
  btn_pr_style.body.main_color = LV_COLOR_MAKE(200,200,200);//背景颜色
  btn_pr_style.body.grad_color = btn_pr_style.body.main_color;
```

```
    //2.创建一个按钮来打开消息对话框
    open_btn = lv_btn_create(scr,NULL);
    lv_obj_set_size(open_btn,150,60);
    lv_obj_set_event_cb(open_btn,event_handler);//设置按钮的事件回调函数
    lv_obj_t * label1 = lv_label_create(open_btn,NULL);
    lv_label_set_text(label1,"Open mbox");
    lv_obj_align(open_btn,NULL,LV_ALIGN_CENTER,0,0);//与屏幕居中对齐

    //3.创建封装之后的消息对话框
    mbox1 = mbox_create(scr);
}
```

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:

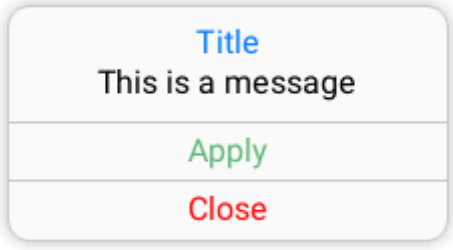

图 3.4.1 初始界面效果

然后我们可以通过点击 Apply 或者 Close 按钮来把此消息对话框给关闭掉,其中 Apply 是马上关闭,而 Close 是定时 2 秒后关闭,消息对话框被关闭时会有动画效果.

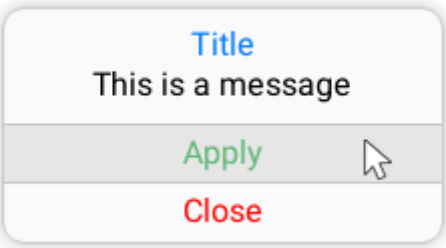

图 3.4.2 按钮点击效果

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