User Guide

正点原子 littleVGL 开发指南

lv_table 表格

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_table 表格

1. 介绍

lv_table 是 littleVGL 中的表格控件,谈到表格,我相信大家并不陌生,它可以非常友好而又直观的展示数据,它是由行(row),列(col)组成的,说的再直白一点,它就是由一个一个的单元格(cell)组成的,在 littleVGL 中的 lv_table 表格控件是非常轻量化的,经过了专门的优化,因为它的单元格(cell)只能存放文本形式的内容,不支持存放其他任何类型的对象或者控件,然后单元格中的文本内容是一个伪标签对象,即不是真正的 lv_label 标签对象,因为它是 lv_table 控件内部通过绘图函数纯绘制出来的效果,所以可以认为每个单元格基本没有内存消耗.

表格控件可以通过 lv_table_set_row_cnt(table, row_cnt)接口来设置其所占的行数,通过 lv_table_set_col_cnt(table, col_cnt)接口来设置其所占的列数,如果不进行单元格合并操作的话,那么此表格所具有的单元格数就等于 row_cnt*col_cnt,另外我们可以通过 lv_table_set_col_width(table, col_id, width)接口来给每一列设置单独的宽度,而表格控件的宽度就是所有列的宽度总和,然后这里有一点需要注意的是,每一行的高度我们是没办法来指定的,因为内部是根据字体大小,内间距等因素来自动计算出每一行的高度,即高度表现为自适应特性.

最后我们要来讲一下单元格,对于表格来说,它是最重要的,因为它是文本数据的载体,我们可以通过 lv_table_set_cell_value(table, row, col, "Content")接口来给每一个单元格设置文本内容,由 row 和 col 这俩个行列参数就可以确定一个单元格的具体位置了,此文本内容在内部会做拷贝操作的,所以不用担心在外部是否被释放了,然后接着可以通过 lv_table_set_cell_align(table, row, col, LV_LABEL_ALIGN_LEFT/CENTER/RIGHT)接口来设置文本内容在单元格中的水平对齐方式,在 lv_table 表格控件中,每一个单元格都具有四种类型,这是通过 lv_table_set_cell_type(table, row, col, type)接口来指定单元格具体为那一种类型的,那这四种类型之间有什么区别呢?其实没什么区别,之所以弄出四种类型,主要是为了让单元格在外观样式上能够得到区分,我们可以给每种不同类型的单元格单独指定一种样式,比如说表格的第一行一般都作为标题行,而标题行的外观样式一般都是比较突出醒目的,如果没有单元格类型这个功能,我们就没有办法实现标题行和数据行之间的样式区分了,当然了,作用远不止于此,比如还可以用来实现单元格的高亮显示等等,单元格还有一个重要的特性,那就是合并操作,不过只能实现水平方向上的合并,而不能实现垂直方向上的合并,即上下俩个单元格之间是没有办法合并为一个单元格的,对于水平方向上的合并操作是通过 lv_table_set_cell_merge_right(table, col, row, true)接口来完成的.

Name	Price
Apple	$7
Banana	$4
Citron	$6

图 1.1 lv_table 表格的外观

2. lv_table 的 API 接口

2.1 主要数据类型

2.1.1 表格样式数据类型

```
enum {
    LV_TABLE_STYLE_BG,    //表格的背景样式
    LV_TABLE_STYLE_CELL1,//表格的类型 1 单元格样式
    LV_TABLE_STYLE_CELL2,//表格的类型 2 单元格样式
    LV_TABLE_STYLE_CELL3,//表格的类型 3 单元格样式
    LV_TABLE_STYLE_CELL4,//表格的类型 4 单元格样式
};
typedef uint8_t lv_table_style_t;
```

我们说了表格中的单元格具有四种类型,而 LV_TABLE_STYLE_CELL1/2/3/4 就是用来分别修饰这四种类型的单元格.

LV_TABLE_STYLE_BG: 用来修饰表格的背景,只用了样式中的 body 字段,默认值为 lv_style_plain_color
LV_TABLE_STYLE_CELL1/2/3/4: 用来分别修饰四种类型的单元格,只用了样式中的 body 字段,默认值为 lv_style_plain

2.2 API 接口

2.2.1 创建对象

```
lv_obj_t * lv_table_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 设置表格所具有的行数

```
void lv_table_set_row_cnt(lv_obj_t * table, uint16_t row_cnt);
```

参数:
table: 表格对象
row_cnt: 表格所具有的总行数

2.2.3 设置表格所具有的列数

```
void lv_table_set_col_cnt(lv_obj_t * table, uint16_t col_cnt);
```

参数:

table: 表格对象

col_cnt: 表格所具有的总列数,最大值不能超过 LV_TABLE_COL_MAX 的值,而 LV_TABLE_COL_MAX 宏的值是在 lv_table.h 头文件中定义的,默认值为 12,如果你觉得最大 12 列满足不了你的需求,那么你可以手动修改 LV_TABLE_COL_MAX 宏的值

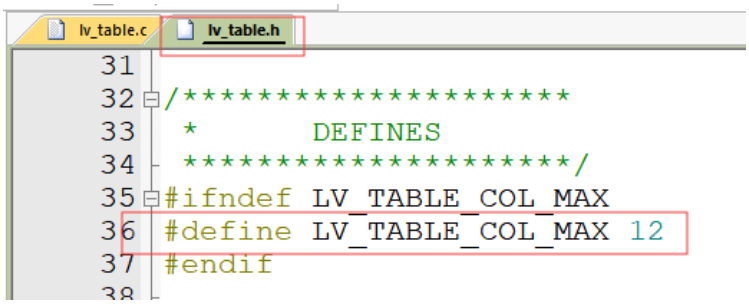

图 2.2.3.1 LV_TABLE_COL_MAX 宏的定义

2.2.4 设置单元格的文本内容

```
void lv_table_set_cell_value(lv_obj_t * table, uint16_t row, uint16_t col, const char * txt);
```

参数:

table: 表格对象

row: 单元格所在的行,从 0 开始

col: 单元格所在的列,从 0 开始

txt: 单元格的文本内容,此文本内容在内部会进行拷贝操作,所以不用担心其在外部是否被释放了

由传入的 row 和 col 俩个参数,就可以定位到一个具体的单元格了,当文本内容的宽度大于所在列的宽度时,文本内容会自动进行换行的,从而也会导致此单元格的高度增加,即整行的高度增加,如果你想手动让文本内容换行的话,那么你可以使用\n 转义字符,但这种手动换行的效果有一点特殊,它会在换行的俩段文本之间加一根和此单元格等宽的水平横线,请看如下例子:

1) 不换行时的效果

```
lv_table_set_cell_value(table1, 3, 0, “Xiong jia yu”);
```

Name	Price
Apple	$7
Banana	$4
Xiong jia yu	$999999

图 2.2.4.1 不换行时的效果

2) 使用\n 手动换行的效果

```
lv_table_set_cell_value(table1, 3, 0, “Xiong \njia yu”);
```

Name	Price
Apple	$7
Banana	$4
Xiong jia yu	$999999

图 2.2.4.2 手动换行时的效果

3) 自动换行时的效果

```
lv_table_set_cell_value(table1, 3, 0, “Xiong jia yu, Are you ok?”);
```

Name	Price
Apple	$7
Banana	$4
Xiong jia yu, Are you ok?	$999999

图 2.2.4.3 自动换行时的效果

2.2.5 设置某一列的宽度

```
void lv_table_set_col_width(lv_obj_t * table, uint16_t col_id, lv_coord_t w);
```

参数:
table: 表格对象
col_id: 列的位置,从 0 开始
w: 列的宽度

如果不设置列的宽度,那么列宽度的默认值为 LV_DPI,表格控件的宽度就是所有列的宽度总和,你用 lv_obj_set_size 接口来给表格控件设置宽度是无效的,然后这里有一点需要注意的是,每一行的高度我们是没办法来指定的,因为内部是根据字体大小,内间距等因素来自动计算出每一行的高度,即高度表现为自适应特性

2.2.6 设置单元格中文本的水平对齐方式

```
void lv_table_set_cell_align(lv_obj_t * table, uint16_t row, uint16_t col, lv_label_align_t align);
```

参数:
table: 表格对象
row: 文本内容所在的行,从 0 开始
col: 文本内容所在的列,从 0 开始

align: 水平对齐方式,有如下三个可选值
LV_LABEL_ALIGN_LEFT:居左对齐
LV_LABEL_ALIGN_CENTER:居中对齐
LV_LABEL_ALIGN_RIGHT:居右对齐

2.2.7 设置单元格的类型

```
void lv_table_set_cell_type(lv_obj_t * table, uint16_t row, uint16_t col, uint8_t type);
```

参数:
table: 表格对象
row: 单元格所在的行,从 0 开始
col: 单元格所在的列,从 0 开始
type: 单元格的类型,可选值为 1, 2, 3, 4 ,分别对应四种类型

如果不设置的话,那么单元格的默认类型为 1,即第一种类型,我们一般把表格的第一行作为标题行,在外观样式上是比较突出醒目的,所以我们可以把第一行中的所有单元格的类型设置为 2 或者其他非 1 值,这样就可以和数据行在外观上有区分了

2.2.8 是否禁止某单元格中文本的自动换行功能

```
void lv_table_set_cell_crop(lv_obj_t * table, uint16_t row, uint16_t col, bool crop);
```

参数:
table: 表格对象
row: 单元格所在的行,从 0 开始
col: 单元格所在的列,从 0 开始
crop: 如果为 true 的话,那么代表禁止自动换行功能,当文本内容超过所在列的宽度时,超出内容将会被截断,如果为 false 的话,那么代表使能自动换行功能,默认值就是为 false 的

2.2.9 合并右边的单元格

```
void lv_table_set_cell_merge_right(lv_obj_t * table, uint16_t row, uint16_t col, bool en);
```

参数:
table: 表格对象
row: 单元格所在的行,从 0 开始
col: 单元格所在的列,从 0 开始
en: 是否把(row,col)单元格和(row,col+1)单元格合并为一个单元格

2.2.10 设置样式

```
void lv_table_set_style(lv_obj_t * table, lv_table_style_t type, const lv_style_t * style);
```

参数:
table: 表格对象

type: 设置哪一部分的样式,有如下 5 个可选值:

LV_TABLE_STYLE_BG: 表格的背景样式

LV_TABLE_STYLE_CELL1: 表格的类型 1 单元格样式

LV_TABLE_STYLE_CELL2: 表格的类型 2 单元格样式

LV_TABLE_STYLE_CELL3: 表格的类型 3 单元格样式

LV_TABLE_STYLE_CELL4: 表格的类型 4 单元格样式

style: 样式

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 3 个自定义样式,分别用来修饰三种类型的单元格,第一个样式用于修饰普通的数据行,第二个样式用于修饰标题行,即第一行,第三个样式用于对某个单元格进行红色高亮显示,然后接着创建一个表格控件,设置其 4 行 3 列,以及每列的宽度,再接着初始化所有单元格的文本内容,以及文本内容的水平对齐方式,当按下 KEY0 按键时,来设置(1,0)单元格是否禁止自动换行功能,当按下 KEY1 按键时,来设置是否将(3,0)单元格和(3,1)单元格合并为一个单元格,当按下 WKUP 按键时,设置(2,0)单元格进行手动换行操作.

3.2 硬件设计

本例程所用到的硬件有:

1) 液晶屏
2) KEY0,KEY1,WKUP 按键

3.3 软件设计

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

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

lv_style_t cell1_style;
lv_style_t cell2_style;
lv_style_t cell3_style;
lv_obj_t * table1;
bool is_cell_crop = false;//默认是使能文本自动换行功能的
bool is_cell_merge = false;//是否合并单元格,默认不合并

#define TABLE_COL_CNT     3     //表格的列数
#define TABLE_ROW_CNT     4     //表格的行数

//定义每一行单元格的文本内容
const char * const TABLE_CELL_VALUE[TABLE_ROW_CNT][TABLE_COL_CNT] = {
    {"Name","Work","Math"},//第一行,作为标题行
```

```
    {"Zhend dian yuan zi","95","90"},//数据行
    {"Xiong jia yu","56","99"},//数据行
    {"Fish","88","93"}//数据行
};
//每一列的宽度
const uint16_t TABLE_COL_WIDTH[TABLE_COL_CNT] = {100,60,60};

//例程入口
void lv_table_test_start()
{
    uint16_t row,col;

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

    //1.创建样式
    //1.1 创建第一种单元格样式,用来修饰普通数据单元格
    lv_style_copy(&cell1_style,&lv_style_plain);
    cell1_style.body.border.width = 1;
    cell1_style.body.border.color = LV_COLOR_BLACK;

    //1.2 创建第二种单元格样式,用来修饰标题行中的单元格
    lv_style_copy(&cell2_style,&lv_style_plain_color);
    cell2_style.body.border.width = 1;
    cell2_style.body.border.color = LV_COLOR_BLACK;
    cell2_style.body.main_color = LV_COLOR_SILVER;//银色的背景
    cell2_style.body.grad_color = LV_COLOR_SILVER;
    cell2_style.text.color = LV_COLOR_BLACK;

    //1.3 创建第三种单元格样式,用来修饰特殊的单元格,比如进行数据高亮显示等等
    lv_style_copy(&cell3_style,&lv_style_plain);
    cell3_style.body.border.width = 1;
    cell3_style.body.border.color = LV_COLOR_BLACK;
    cell3_style.text.color = LV_COLOR_RED;//文本颜色为红色,高亮显示

    //2.创建表格
    table1 = lv_table_create(scr,NULL);
    lv_table_set_col_cnt(table1,TABLE_COL_CNT);//设置表格的总列数
    lv_table_set_row_cnt(table1,TABLE_ROW_CNT);//设置表格的总行数
    //设置每一列的宽度以及标题行的文本对齐方式和单元格类型
    for(col=0;col<TABLE_COL_CNT;col++)
    {
        //设置每一列的宽度
        lv_table_set_col_width(table1,col,TABLE_COL_WIDTH[col]);
```

```
            //标题行的文本内容居中对齐
            lv_table_set_cell_align(table1,0,col,LV_LABEL_ALIGN_CENTER);
            lv_table_set_cell_type(table1,0,col,2);//设置为第二种单元格类型
        }

        //给(2,1)单元格设置为第三种类型,具有红色高亮显示的外观
        lv_table_set_cell_type(table1,2,1,3);

        //设置所有单元格的文本内容
        for(row=0;row<TABLE_ROW_CNT;row++)
        {
            for(col=0;col<TABLE_COL_CNT;col++)
            {
                lv_table_set_cell_value(table1,row,col,(const  char*)TABLE_CELL_VALUE
[row][col]);//设置文本内容
            }
        }

        //设置表格的背景样式,为透明
        lv_table_set_style(table1,LV_TABLE_STYLE_BG,&lv_style_transp_tight);

        //设置第一种单元格的样式
        lv_table_set_style(table1,LV_TABLE_STYLE_CELL1,&cell1_style);

        //设置第二种单元格的样式
        lv_table_set_style(table1,LV_TABLE_STYLE_CELL2,&cell2_style);

        //设置第二种单元格的样式
        lv_table_set_style(table1,LV_TABLE_STYLE_CELL3,&cell3_style);
        lv_obj_align(table1,NULL,LV_ALIGN_CENTER,0,0);//设置表格与屏幕居中对齐
    }

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

        if(key==KEY0_PRES)
        {
            is_cell_crop = !is_cell_crop;
            //设置(1,0)单元格是否禁止自动换行功能
            lv_table_set_cell_crop(table1,1,0,is_cell_crop);
            lv_obj_invalidate(table1);//得手动刷新 table1 表格
        }else if(key==KEY1_PRES)
```

```
        {
            is_cell_merge = !is_cell_merge;

            //是否将(3,0)单元格和(3,1)单元格合并为一个单元格
            lv_table_set_cell_merge_right(table1,3,0,is_cell_merge);
        }else if(key==WKUP_PRES)
        {
            lv_table_set_cell_value(table1,2,0,"Xiong\njia yu");//用\n 转义字符进行手动换行
        }
    }
```

3.4 下载验证

把代码下载进去之后,我们可以看到如下所示的界面效果:

Name	Work	Math
Zhend dian yuan zi	95	90
Xiong jia yu	56	99
Fish	88	93

图 3.4.1 初始界面效果

然后我们按一下 KEY0 按键,禁止掉(1,0)单元格的自动换行功能,即表现为截断效果,如下图所示:

Name	Work	Math
Zhend dian yu	95	90
Xiong jia yu	56	99
Fish	88	93

图 3.4.2 禁止自动换行效果

再接着我们按一下 KEY1 按键,让(3,0)单元格和(3,1)单元格合并为一个单元格,如下图所示:

Name	Work	Math
Zhend dian yuan zi	95	90
Xiong jia yu	56	99
Fish		93

图 3.4.3 单元格合并效果

最后我们按一下 WKUP 按键,让(2,0)单元格进行手动换行操作,如下图所示:

Name	Work	Math
Zhend dian yuan zi	95	90
Xiong jia yu	56	99
Fish	88	93

图 3.4.4 手动换行效果

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