User Guide

正点原子 littleVGL 开发指南

lv_chart 图表

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_chart 图表

1. 介绍

lv_chart 图表控件主要是由背景,水平垂直分割线,数据线(series)三部分构成的,其中数据线是 lv_chart 图表控件中最重要的部分,一条数据线就是一些数据点的集合,一条数据线可以被绘制成折线,柱状,散点,面积等几种主要的图表类型,或者是这几种图表类型共同组拼后的结果,即进行位或操作,而一个 lv_chart 图表控件中又可以同时绘制多条数据线,数量是不受限制的,每条数据线的颜色都可以单独指定,这是通过 lv_chart_add_series(chart, color)添加数据线接口来具体操作的,我们说过了数据线就是数据点的集合,说的再具体一点就是 y 坐标点的集合,至于数据点的个数我们是可以通过 lv_chart_set_point_count(chart, point_num)接口来设置的,对于数据值的修改或者更新,lv_chart 图表控件提供了如下四种方式:

1)直接手动修改数据点的值,如 ser1->points[2] = 45,这样修改完后,还得手动调用 lv_chart_refresh(chart)接口来刷新图表
2)使用 lv_chart_set_next(chart, ser, value)接口动态修改,这里又包含了 2 种更新方式,一种为左平移方式,另一种为环形覆盖方式
3)使用 lv_chart_init_points(chart, ser, value)接口把所有的数据点设置为同一个 value 值
4)使用 lv_chart_set_points(chart, ser, value_array)接口,传入数组的方式为数据线中的每一个数据点一一赋值

然后这里有一点需要注意的是,当数据点的值为 LV_CHART_POINT_DEF 时,那么此数据点将不会被绘制出来,而是被直接跳过了,会使整条数据线看起来有点断裂了的感觉.

前面说了数据点其实就是 y 坐标点,是不包含 x 坐标信息的,那么我们可以通过 lv_chart_set_range(chart, y_min, y_max)接口来设置 y 轴的一个数值范围,使每一个数据点在 y 轴上都有一个确切的位置,有的同学可能就会疑惑了,数据点中不包含 x 坐标信息,那么 x 轴上如何确定位置呢?在 lv_chart 图表控件中它是这样处理的,它会根据数据线中的数据点总个数(用 n 表示)在 x 轴上做 n 等分处理,也就是说每个数据点之间的水平间距是相等的,而且是等于图表控件的宽度除以 n,我想 littleVGL 这么做的目的是为了减少 lv_chart 图表控件的复杂性和减少内存开销.

为了使图表看起来更美观,更人性化,我们也可以通过 lv_chart_set_x/y_tick_text, lv_chart_set_x/y_tick_length, lv_chart_set_margin 等接口来给图表设置 x 轴或者 y 轴的刻度线和主刻度标题.

2. lv_chart 的 API 接口

2.1 主要数据类型

2.1.1 图表类型数据类型

```
enum {
    LV_CHART_TYPE_NONE          = 0x00,
    LV_CHART_TYPE_LINE          = 0x01,
    LV_CHART_TYPE_COLUMN        = 0x02,
    LV_CHART_TYPE_POINT         = 0x04,
    LV_CHART_TYPE_VERTICAL_LINE = 0x08,
    LV_CHART_TYPE_AREA          = 0x10,
};
typedef uint8_t lv_chart_type_t;
```

用来指定 lv_chart 图表控件中所有数据线的图表类型,一条数据线可以被绘制成折线图,柱状图,散点图,面积图,或者是这几种主要图表类型组拼之后的结果,因为这些值之间是可以进行位或操作的,比如为LV_CHART_TYPE_LINE|LV_CHART_TYPE_COLUMN,那么你会看到折线图和柱状图共存的形式

LV_CHART_TYPE_NONE: 不显示任何数据线(series)
LV_CHART_TYPE_LINE: 数据线将会被绘制成折线图
LV_CHART_TYPE_COLUMN: 数据线将会被绘制成柱状图
LV_CHART_TYPE_POINT: 数据线将会被绘制成散点图
LV_CHART_TYPE_AREA: 数据线将会被绘制成面积图
LV_CHART_TYPE_VERTICAL_LINE: 当数据点的个数和图表控件的宽度相等时才会起作用,主要是性能优化了,它会比 LV_CHART_TYPE_AREA 的渲染速度更快,我们了解即可,一般用不到

图 2.1.1.1 折线图

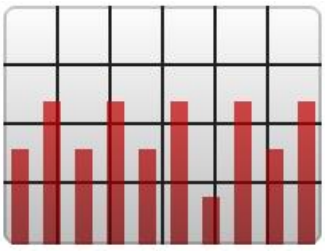

图 2.1.1.2 柱状图

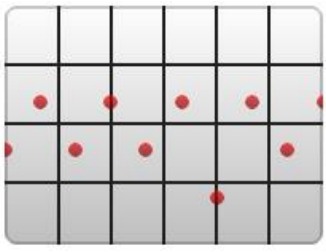

图 2.1.1.3 散点图

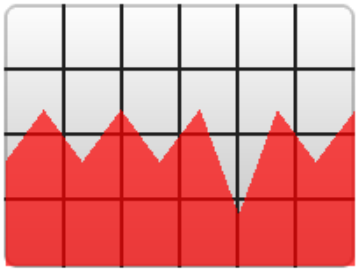

图 2.1.1.4 面积图

图 2.1.1.1 折线,柱状,散点共存图

2.1.2 数据点更新模式数据类型

```
enum {
    LV_CHART_UPDATE_MODE_SHIFT,
    LV_CHART_UPDATE_MODE_CIRCULAR,
};
typedef uint8_t lv_chart_update_mode_t;
```

此数据类型主要是用在 lv_chart_set_update_mode(chart,update_mode)接口中,用来确定 lv_chart_set_next 接口的数据点更新模式的

LV_CHART_UPDATE_MODE_SHIFT: 左平移方式,当用 lv_chart_set_next 接口来添加一个新的数据点时,它会先把最左边的数据点给删除掉,然后把新的数据点放到最右边,看起来有一种向左平移的效果

LV_CHART_UPDATE_MODE_CIRCULAR: 环形覆盖方式,用 lv_chart_set_next 接口产生出来的新数据点会从最左边到最右边的方式依次替换掉老数据点,当到了最右边时,又会重新跳到最左边来开始新的循环

2.1.3 轴刻度绘制方式数据类型

```
enum {
    LV_CHART_AXIS_SKIP_LAST_TICK     = 0x00,
    LV_CHART_AXIS_DRAW_LAST_TICK   = 0x01
};
typedef uint8_t lv_chart_axis_options_t;
```

这个数据类型主要是用在 lv_chart_set_x/y_tick_texts 接口中的,用来确定是否需要绘制最后面的一个主刻度.

LV_CHART_AXIS_SKIP_LAST_TICK：不绘制最后面的一个主刻度
LV_CHART_AXIS_DRAW_LAST_TICK：绘制最后面的一个主刻度

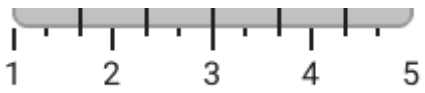

图 2.1.3.1 不绘制最后面的主刻度效果

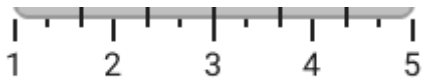

图 2.1.3.2 绘制最后面的主刻度效果

通过上面的效果对比,我们可以发现数字 5 那里有一个主刻度线的区别

2.1.4 数据线数据类型

```
typedef struct
{
    lv_coord_t * points;
    lv_color_t color;
    uint16_t start_point;
} lv_chart_series_t;
```

这是一个结构体,用来描述数据线的,其中 points 存放着所有的数据点,color 指定数据线的颜色,start_point 是归 littleVGL 库内部使用,我们用不到的

2.1.5 图表样式数据类型

```
enum {
    LV_CHART_STYLE_MAIN,
};
typedef uint8_t lv_chart_style_t;
```

图表的样式就一种,还算是简单的,我们主要是来介绍此样式中各字段的具体用途.

LV_CHART_STYLE_MAIN: 样式中的 body 字段用来修饰图表的背景,样式中的 line 字段用来修饰水平垂直分割线和 x/y 轴上的刻度线,样式中的 text 字段用来修饰 x/y 轴上的文本标题

2.2 API 接口

2.2.1 创建对象

```
lv_obj_t * lv_chart_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 添加数据线

```
lv_chart_series_t * lv_chart_add_series(lv_obj_t * chart, lv_color_t color);
```

参数:
chart: 图表对象
color: 指定被添加数据线的颜色

返回值:
返回被添加的数据线对象

拿到数据线对象之后,是有作用的,因为后面的某些接口会用到此对象

2.2.3 清除某数据线中的所有数据点

```
void lv_chart_clear_serie(lv_obj_t * chart, lv_chart_series_t * serie);
```

参数:
chart: 图表对象
serie: 数据线对象

这个实现原理就是用 LV_CHART_POINT_DEF 给所有的数据点进行赋值,前面说过值为 LV_CHART_POINT_DEF 的数据点是不会被绘制出来的,注意,清除之后界面是不会自动刷新的,此时你可以调用 lv_chart_refresh(chart)来手动刷新图表

2.2.4 设置水平和垂直分割线的条数

```
void lv_chart_set_div_line_count(lv_obj_t * chart, uint8_t hdiv, uint8_t vdiv);
```

参数:
chart: 图表对象
hdiv: 水平分割线的条数,默认为 3 条
vdiv: 垂直分割线的条数,默认为 5 条

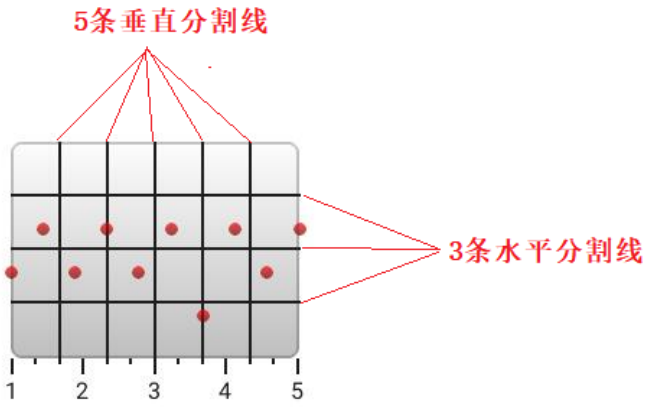

图 2.2.4.1 水平垂直分割线

2.2.5 设置 y 轴的数值范围

```
void lv_chart_set_range(lv_obj_t * chart, lv_coord_t ymin, lv_coord_t ymax);
```

参数:
chart: 图表对象
ymin: 最小值,默认值为 0,对应 y 轴的最底部点
ymax: 最大值,默认值为 100,对应 y 轴的最顶部点

2.2.6 设置图表的类型

```
void lv_chart_set_type(lv_obj_t * chart, lv_chart_type_t type);
```

参数:
chart: 图表对象
type: 图表类型,有如下 6 个可选值:
LV_CHART_TYPE_NONE: 不显示
LV_CHART_TYPE_LINE: 折线图
LV_CHART_TYPE_COLUMN: 柱状图
LV_CHART_TYPE_POINT: 散点图
LV_CHART_TYPE_VERTICAL_LINE: 基本用不到,了解即可
LV_CHART_TYPE_AREA: 面积图

这个接口设置了所有数据线的统一图表类型,我们是不能为某条数据线单独指定图表类型的

2.2.7 设置数据点的个数

```
void lv_chart_set_point_count(lv_obj_t * chart, uint16_t point_cnt);
```

参数:
chart: 图表对象
point_cnt: 数据点的个数

这个接口设置了每条数据线所具有的数据点个数,我们是不能为某条数据线单独指定数据点个数的

2.2.8 设置数据线的透明度

```
void lv_chart_set_series_opa(lv_obj_t * chart, lv_opa_t opa);
```

参数:
chart: 图表对象
opa: 透明度,默认值为 LV_OPA_COVER

2.2.9 设置数据线的宽度

```
void lv_chart_set_series_width(lv_obj_t * chart, lv_coord_t width);
```

参数:
chart: 图表对象
width: 数据线的宽度

当数据线是折线图时,它就是设置线的宽度,当数据线是散点图时,它就是设置点的半径,此 width 宽度值对柱状图无意义,因为柱状图是内部自动计算宽度的

2.2.10 设置数据线的黑阴影效果

```
void lv_chart_set_series_darking(lv_obj_t * chart, lv_opa_t dark_eff);
```

参数:
chart: 图表对象
dark_eff: 黑阴影的透明度,当设为 0 时,相当于没有黑阴影效果

此黑阴影效果只对散点图或者柱状图有效果,设置之后图表更美观一点

2.2.11 给所有数据点设置统一的值

```
void lv_chart_init_points(lv_obj_t * chart, lv_chart_series_t * ser, lv_coord_t y);
```

参数:
chart: 图表对象
ser: 数据线对象
y: 数据点的值

2.2.12 给所有数据点一一赋值

```
void lv_chart_set_points(lv_obj_t * chart, lv_chart_series_t * ser, lv_coord_t y_array[]);
```

参数:
chart: 图表对象
ser: 数据线对象
y_array: 数组形式,存放着每个数据点的值

2.2.13 添加新的数据点

```
void lv_chart_set_next(lv_obj_t * chart, lv_chart_series_t * ser, lv_coord_t y);
```

参数:
chart: 图表对象
ser: 数据线对象

y: 数据点的值

往指定的 ser 数据线中添加值为 y 的新数据点,在添加新数据点时,这里有 2 种数据更新模式,请看 lv_chart_set_update_mode 接口

2.2.14 设置数据更新模式

```
void lv_chart_set_update_mode(lv_obj_t * chart, lv_chart_update_mode_t update_mode);
```

参数:
chart: 图表对象
update_mode: 数据更新模式,有如下 2 个可选值:
LV_CHART_UPDATE_MODE_SHIFT: 左平移方式,当用 lv_chart_set_next 接口来添加一个新的数据点时,它会先把最左边的数据点给删除掉,然后把新的数据点放到最右边,看起来有一种向左平移的效果
LV_CHART_UPDATE_MODE_CIRCULAR: 环形覆盖方式,用 lv_chart_set_next 接口产生出来的新数据点会从最左边到最右边的方式依次替换掉老数据点,当到了最右边时,又会重新跳到最左边来开始新的循环

2.2.15 设置样式

```
static inline void lv_chart_set_style(lv_obj_t * chart, lv_chart_style_t type, const lv_style_t * style);
```

参数:
chart: 图表对象
type: 设置哪部分的样式,目前就 LV_CHART_STYLE_MAIN 一个可选值
style: 样式

2.2.16 设置 X 或 Y 轴上的刻度线长度

```
void lv_chart_set_x_tick_length(lv_obj_t * chart, uint8_t major_tick_len, uint8_t minor_tick_len);
void lv_chart_set_y_tick_length(lv_obj_t * chart, uint8_t major_tick_len, uint8_t minor_tick_len);
```

参数:
chart: 图表对象
major_tick_len: 主刻度线的长度
minor_tick_len: 次刻度线的长度

为了让大家更容易理解什么是主刻度线和次刻度线,请看下图

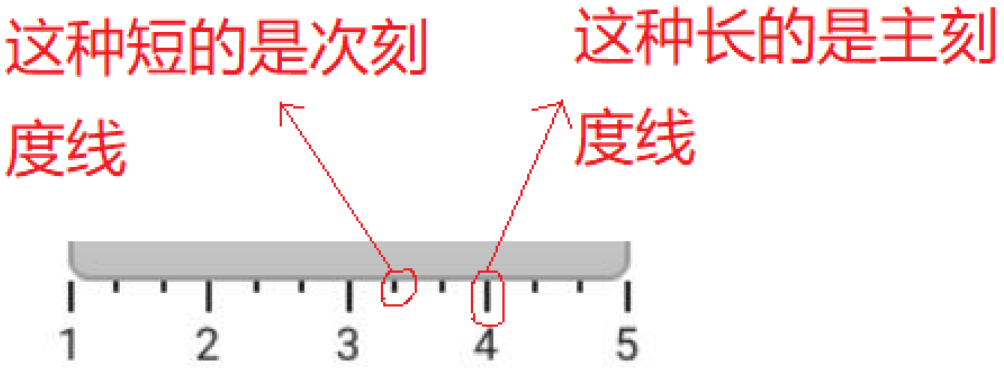

图 2.2.16.1 主次刻度线图解

2.2.17 设置 X 或 Y 轴上的刻度线数量和主刻度线标题

```
        void lv_chart_set_x_tick_texts(lv_obj_t * chart, const char * list_of_values, uint8_t
num_tick_marks, lv_chart_axis_options_t options);
        void lv_chart_set_y_tick_texts(lv_obj_t * chart, const char * list_of_values, uint8_t
num_tick_marks, lv_chart_axis_options_t options);
```

参数:

chart: 图表对象

list_of_values: 每个主刻度线对应的文本标题,标题之间需要用\n 换行符来作为间隔标志,传入了多少个标题就意味着有多少个主刻度线,如下面例子所示:

const char * list_of_values = "1\n2\n3\n4\n5";

这是传入了 1, 2, 3, 4, 5 总共五个标题,也意味着有五个主刻度线

num_tick_marks: 相邻的俩个主刻度线之间又有 num_tick_marks-1 个次刻度线

options: 决定是否绘制最末尾的一个主刻度线,有如下 2 个可选值:

LV_CHART_AXIS_SKIP_LAST_TICK:不绘制最后面的一个主刻度

LV_CHART_AXIS_DRAW_LAST_TICK:绘制最后面的一个主刻度

最后给大家举一个小例子如下:

```
        lv_chart_set_x_tick_texts(chart, "1\n2\n3\n4\n5", 3, LV_CHART_AXIS_DRAW_LAS
T_TICK);
        lv_chart_set_x_tick_length(chart,10, 3);
        lv_chart_set_margin(chart,30);//这个接口后面讲解,必须得设置,否则看不到效果的
```

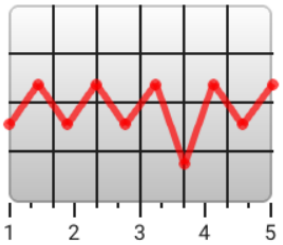

图 2.2.17.1 刻度数量和主刻度标题演示效果

2.2.18 设置刻度区域的高度

```
void lv_chart_set_margin(lv_obj_t * chart, uint16_t margin);
```

参数:
chart: 图表对象
margin: 刻度区域的高度,单位像素

如果我们的图表在 x 或者 y 轴上设置了刻度线和主刻度标题的话,那么我们必须得调用 lv_chart_set_margin 接口来设置刻度区域的高度,否则可能会看不到完整的效果,我们设置的刻度区域高度必须得满足以下要求:

刻度区域高度 margin > (主刻度线长度 major_tick_len + 主刻度标题的文本高度)

其中主刻度标题的文本高度取决于样式中所用字体的大小,最后为了方便大家理解,给出一张图解.

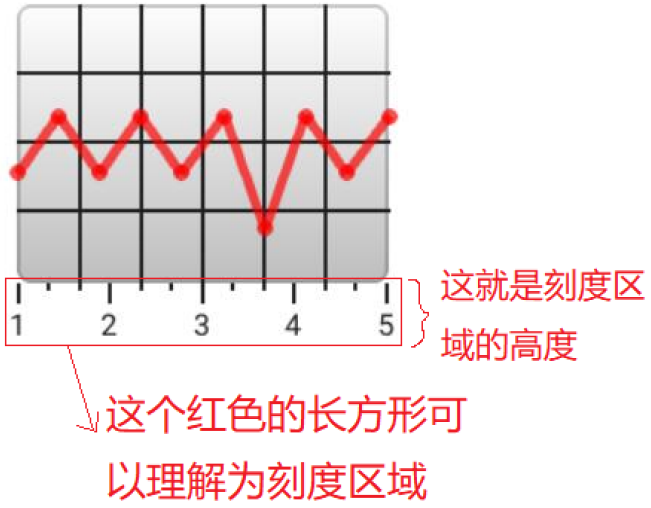

图 2.2.18.1 刻度区域高度的图解

2.2.19 刷新图表

```
void lv_chart_refresh(lv_obj_t * chart);
```

参数:
chart: 图表对象

有些操作之后,图表是不会自动刷新界面的,此时我们可以通过此 API 接口来实现手动刷新,其实此 API 接口内部就是在调用 lv_obj_invalidate(chart);

2.2.20 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表控件,然后我们接着创建一个 chart1 图表,设置好它的各种属性,比如透明度,y 轴数值范围,刻度线,数据点个数等等,然后我们往 chart1 图表中添加 series1 和 series2 俩条数据线,series1 是采用数组一一赋值和手动直接修改的方式,而 series2 是采用统一赋值的方式,当按下 KEY0 键时,会来回切换 chart1 的图表类型,当按下 KEY1 键时,会往 series1 数据线上添加新的数据点,当按下 KEY2 键时,会来回切换数据点的更新模式

3.2 硬件设计

本例程所用到的硬件有:

1) 液晶屏
2) KEY0,KEY1,KEY2 按键

3.3 软件设计

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

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

lv_style_t scr_style;

#define POINT_COUNT      10    //每条数据线所具有的数据点个数
lv_style_t main_style;
const lv_coord_t series1_y[POINT_COUNT] = {30,60,40,60,20,60,50,80,60,80};
lv_obj_t * chart1;
lv_chart_series_t * series1;
lv_chart_series_t * series2;


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

```
    lv_style_copy(&scr_style,&lv_style_plain_color);
    scr_style.body.main_color = LV_COLOR_CYAN;
    scr_style.body.grad_color = scr_style.body.main_color;

//给屏幕设置一个纯色的背景,让我们的图表显示更明显,同时可以解决色点现象
    lv_obj_set_style(scr,&scr_style);

    //1.创建样式
    lv_style_copy(&main_style,&lv_style_pretty);
    main_style.body.main_color = LV_COLOR_WHITE;//主背景为纯白色
    main_style.body.grad_color = main_style.body.main_color;
    main_style.body.border.color = LV_COLOR_BLACK;//边框的颜色
    main_style.body.border.width = 3;//边框的宽度
    main_style.body.border.opa = LV_OPA_COVER;
    main_style.body.radius = 0;
    main_style.line.color = LV_COLOR_GRAY;//分割线和刻度线的颜色
    main_style.text.color = LV_COLOR_BLUE;//主刻度标题的颜色

    //2.创建图表对象
    //2.1  创建图表对象
    chart1 = lv_chart_create(scr,NULL);
    lv_obj_set_size(chart1,180,200);//设置图表的大小
    lv_obj_align(chart1,NULL,LV_ALIGN_CENTER,20,0);//设置对齐方式

    //设置为散点和折线的组合
    lv_chart_set_type(chart1,LV_CHART_TYPE_POINT|LV_CHART_TYPE_LINE);

    //设置数据线的透明度,不设置的话,则 LV_OPA_COVER 是默认值
    lv_chart_set_series_opa(chart1,LV_OPA_80);
    lv_chart_set_series_width(chart1,4);//设置数据线的宽度
    lv_chart_set_series_darking(chart1,LV_OPA_80);//设置数据线的黑阴影效果
    lv_chart_set_style(chart1,LV_CHART_STYLE_MAIN,&main_style);//设置样式

    //设置每条数据线所具有的数据点个数,如果不设置的话,则默认值是 10
    lv_chart_set_point_count(chart1,POINT_COUNT);
    lv_chart_set_div_line_count(chart1,4,4);//设置水平和垂直分割线
    lv_chart_set_range(chart1,0,100);//设置 y 轴的数值范围,[0,100]也是默认值

    //设置 y 轴的主刻度线长度和次刻度线长度
    lv_chart_set_y_tick_length(chart1,10,3);

    lv_chart_set_y_tick_texts(chart1,"100\n80\n60\n40\n20\n0",2,LV_CHART_AXIS_D
RAW_LAST_TICK);
```

```
        //设置 x 轴的主刻度线长度和次刻度线长度
        lv_chart_set_x_tick_length(chart1,10,3);

        //设置 x 轴的刻度数和主刻度标题
        lv_chart_set_x_tick_texts(chart1,"0\n2\n4\n6\n8\n10",2,LV_CHART_AXIS_DRAW_
LAST_TICK);
        lv_chart_set_margin(chart1,40);//设置刻度区域的高度

        //2.2 往图表中添加第 1 条数据线
        series1 = lv_chart_add_series(chart1,LV_COLOR_RED);//指定为红色
        lv_chart_set_points(chart1,series1,(lv_coord_t*)series1_y);//初始化数据点的值
        series1->points[1] = 70;//也可以采用直接修改的方式
        lv_chart_refresh(chart1);//如果是采用直接修改的方式,请最好调用一下刷新操作

        //2.3 往图表中添加第 2 条数据线
        series2 = lv_chart_add_series(chart1,LV_COLOR_BLUE);//指定为蓝色
        lv_chart_init_points(chart1,series2,90);//给所有数据点设置统一的值

    }

    //按键处理
    void key_handler()
    {
        static u8 type_index = 0;//图表的模式
        static lv_chart_update_mode_t update_mode = LV_CHART_UPDATE_MODE_
SHIFT; //数据点更新模式

        u8 key = KEY_Scan(0);

        if(key==KEY0_PRES)
        {
            //为了使数据线的图表类型变化看得更明显,我们把 series2 数据线给隐藏
            //了,防止干扰视线
            lv_chart_clear_serie(chart1,series2);

            //来回切换图表类型
            lv_chart_set_type(chart1,(lv_chart_type_t)(0x01<<type_index));
            type_index++;
            if(type_index==6)
                type_index = 0;
        }else if(key==KEY1_PRES)
        {
```

```
            //往 series1 数据线上添加新的数据点
            //可以切换到不同的数据点更新模式来观看不同的效果
            lv_chart_set_next(chart1,series1,40);
        }else if(key==KEY2_PRES)
        {
            //来回切换数据点的更新模式
        //如果不设置的话,则默认是 LV_CHART_UPDATE_MODE_SHIFT 左平移模式
            update_mode = !update_mode;
            lv_chart_set_update_mode(chart1,update_mode);

            //为了使效果看起来更明显,我们将 series1 数据线的值初始化到原始状态
            lv_chart_set_points(chart1,series1,(lv_coord_t*)series1_y);
        }
    }
```

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:

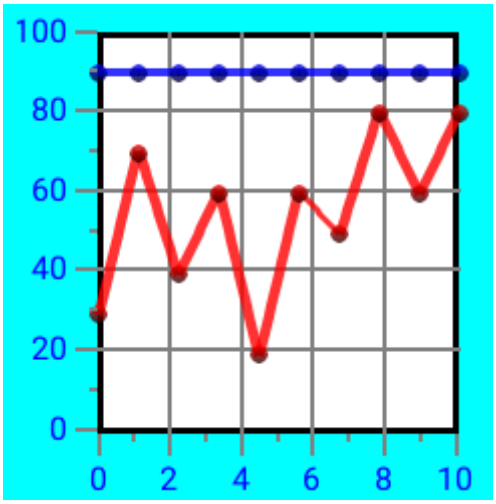

图 3.4.1 初始界面效果

然后我们可以按下 KEY0 键来切换图表类型,可以依次看到如下所示界面效果:

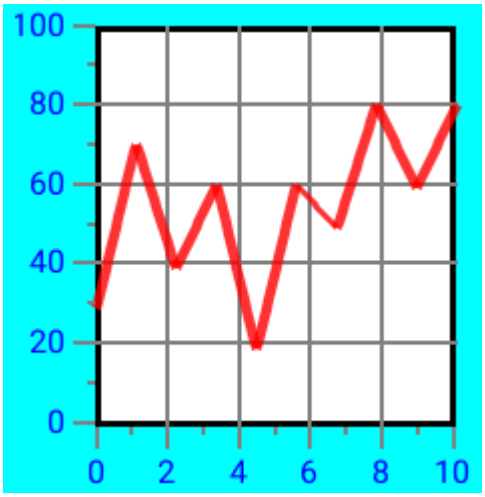

图 3.4.2 折线图

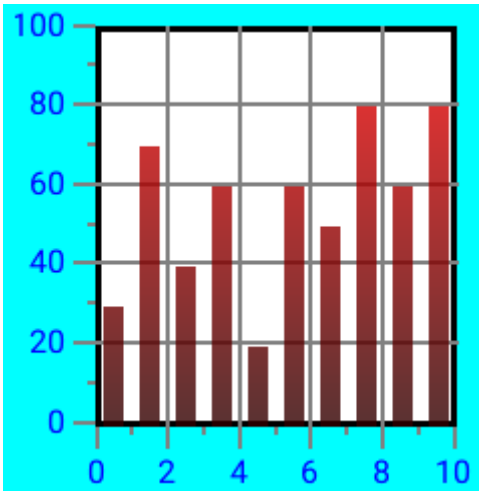

图 3.4.3 柱状图

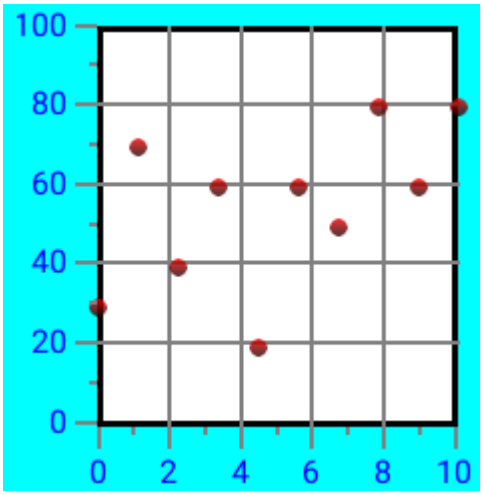

图 3.4.4 散点图

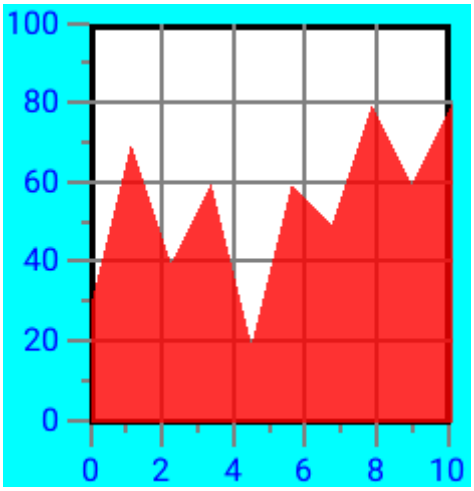

图 3.4.5 面积图

注意上面是为了使图表类型效果观看的更明显,所以在按下 KEY0 键时,我们用代码把 series2 数据线给隐藏了,最后我们还可以按下 KEY1 键来往 series1 数据线中添加新的数据点,以及配合 KEY2 键来改变数据点的更新模式

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