QuecPython

提升 LCD 局部刷屏效率的优化建议

版本： 1.0

日期： 2022-12-10

状态： 临时文件

上海移远通信技术股份有限公司始终以为客户提供最及时、最全面的服务为宗旨。如需任何帮助，请随时联系我司上海总部，联系方式如下：

上海移远通信技术股份有限公司
上海市闵行区田林路 1016 号科技绿洲 3 期（B 区）5 号楼　　邮编：200233
电话：+86 21 51086236　　邮箱：info@quectel.com

或联系我司当地办事处，详情请登录：
http://quectel.com/cn/support/sales.htm

如需技术支持或反馈我司技术文档中的问题，可随时登陆如下网址：
http://quectel.com/cn/support/technical.htm
或发送邮件至：support@quectel.com

前言

上海移远通信技术股份有限公司提供该文档内容用以支持其客户的产品设计。客户须按照文档中提供的规范、参数来设计其产品。由于客户操作不当而造成的人身伤害或财产损失，本公司不承担任何责任。在未声明前，上海移远通信技术股份有限公司有权对该文档进行更新。

版权申明

文档历史

修订记录

版本	日期	作者	变更表述
1.0	2022-12-10	高建	创建文件

目录

文档历史 3
目录 4
案例问题： 5
问题分析： 5
优化建议： 6

案例问题：

这几天有个客户反映这样一个问题，他们的 LCD 设备是一块 240*320 的 ST7789 显示屏，需求包括局部颜色刷屏和中间区域的文字显示，且因为中文字库庞大，他们没用使用 LVGL。在实际运行中发现，LCD 局部刷屏效率低下，显示文字的效率也相当低下，故向我们求助。

问题分析：

在分析客户脚本的时候，发现客户都是调用的 LCD 的 lcd_write 接口进行局部刷屏的，其中局部颜色刷屏的代码如下：

```
# 区域填充
    def lcd_Fill(self, x, y, xe, ye, color):
        for i in range(y, ye + 1):
            buf = []
            for z in range(x, xe + 1):
                buf.append(color & 0xff)
                buf.append(color >> 8)
            self.lcd.lcd_write(bytearray(buf), x, i, xe, i)
```

看到使用了一个两层嵌套的 for 循环去让一个列表里 append 颜色 buf，然后一行行地去往设备中写入 buf。这就有两个影响性能的问题：1. 数据准备时间太长；2. 一行行执行 lcd_write 效率低。

然后再看客户的显示文字的接口：

```
#显示单个ascll码
    def __display_ascll(self,str,x,y,fc,fb):
        rgb_buf = []
        data = self.__ascll_zk_get(str)
        for h in range(0, 2):  # 行数
            for w in range(0, 8):  # 数据宽度
                for l in range(0, 8):  # 列数
                    if (data[(h * 8) + l] >> w) & 0x01 == 0x00:
                        rgb_buf.append(fb & 0xff)
                        rgb_buf.append(fb >> 8)
                    else:
                        rgb_buf.append(fc & 0xff)
                        rgb_buf.append(fc >> 8)
        self.lcd.lcd_write(bytearray(rgb_buf), x, y, x + 7, y + 15)
```

可以看到用了一个三层的嵌套 for 循环，这些执行完之后还是一个字符，一段文字每个字符都得重复这样的流程，效率显然不行。

优化建议：

针对上面的性能问题，我们做出如下建议：

1. 局部颜色刷屏的时候，不要采用 for 循环生成 buf 的形式，换采取以下方法：

```
def fill(lcd, x_s, y_s, x_e, y_e, color):
    tmp = color.to_bytes(2, 'little')
    count = (x_e - x_s + 1) * (y_e - y_s + 1)

    color_buf = tmp * count

    lcd.lcd_write(color_buf, x_s, y_s, x_e, y_e)
```

这里直接用 bytes 相乘的方法生成 buf，最后一次写入区域 buf，效率能提高不少。

2. 文字显示的问题，我们建议客户将文字 buf 生成之后，并入要写的区域 buf 中，再一次性写入刷屏。经用户测试，显示效率也得到了提高。