Report for PyPainter

魏楚
5130309032
致远学院计算机方向

2015 年 6 月 9 日

1 简介

PyPainter主要实现了画图的一部分功能，以及提供了打开/保存为bmp格式文件的功能。本程序利用Windows7下的python2.7.8开发完成，仅使用了python2.7.8 的自带标准库Tkinter, os 和math。

2 使用

运行程序后会出现一个简洁的界面，分为菜单栏，工具栏和画布三部分。

2.1 基础功能

界面左边为工具栏，第一块用于设置线条的颜色，第二块用于设置填充颜色，第三块用于设置画笔，最后是清屏与退出按钮。颜色的配置采用RGB，每个值的范围为0-255，初始时线条颜色为黑色（0,0,0），填充颜色为白色（255,255,255）。

画笔依次为铅笔，线段，多边形，矩形，椭圆和填充，使用方法与Windows自带的画图基本相同。

2.1.1 铅笔

铅笔的颜色为设定的线条颜色，在画布上按下鼠标并移动，松开即停止。

2.1.2 线段

线段的颜色为设定的线条颜色，在画布上按下鼠标并移动调整，松开即结束，两个端点分别为按下时所在点与松开时所在点。

2.1.3 多边形

多边形的线条颜色为设定的线条颜色，没有填充颜色。第一条边的画法与画线段相同，之后各点可以直接单击决定，也可以按下鼠标后拖动调整，最后一个点处双击结束。

2.1.4 矩形

矩形的线条颜色为设定的线条颜色，填充颜色为设定的填充颜色。在画布上按下鼠标并拖动调整，松开即停止，矩形的两个端点分别为按下时所在点与松开时所在点。

2.1.5 椭圆

同矩形

2.1.6 填充

填充的颜色为设定的填充颜色，点击画布上某点即可像Windows自带的画图一样填充。由于填充过程实际上是用floodfill画了许多像素点，当填充区域过大时速度会很慢，而且画布上元素多了以后画其他东西也会很卡，建议尽量少用或者只填充小区域，如填充100x100的矩形可以很快完成。

2.2 扩展功能

2.2.1 清屏

直接点击左下角的Clear按钮即可。

2.2.2 退出

点击左下角Quit按钮，或者点击菜单栏的File–Exit，或者直接点关闭按钮。

2.2.3 帮助

点击菜单栏Help里面的Help和About可以查看帮助和关于信息。

2.2.4 打开bmp文件

点击菜单栏File–Open，输入文件名即可打开，默认值为1.bmp。

注意：读入时是把每个像素点以create_line函数画在画布上的，画布上元素特别多，因此进一步绘图时会比较卡。

2.2.5 保存bmp文件

点击菜单栏File–Save，输入文件名即可保存，默认值为2.bmp。

3 实现

3.1 bmp文件分析

本程序的一个重点是bmp文件读取/保存。为了实现这一功能，我查阅了一些资料（http://www.cnblogs.com/kingmoon/archive/2011/04/18/2020097.html），并利用16进制编辑器打开了bmp文件进行分析（期间发现上述网页中有一点小错误），最终学会了（无压缩的）bmp文件的编码方式，具体实现见open_image和save_image函数。

粗略地说，bmp文件是二进制文件，前54个字节保存了一些诸如图像大小、编码方式等信息，后面依次保存了所有像素点的RGB信息。

开始时我是逐字节读入/保存的，速度特别慢。后来我利用一个字符串作为缓冲，如读入时把文件头外的所有信息全部读入字符串buf中，这样速度提升了好几倍。

3.2 画图

由于并未找到获取画布上某点RGB值的接口，我用了一个二维数组table记录所有点的RGB信息。画图时界面上用了Tkinter自带的几个函数，拖动时利用tag删除上一次画的东西并重画一个，松开后再画一个无tag的，并根据所画图形通过计算更改数组里面对应点的RGB值。

填充工具采用了floodfill算法，由于是以像素为单位填充的，所以特别慢。因此，我每填充1024个像素就刷新一下画布，防止用户误解为未进行操作。