《计算机图形学》 5月报告

学号，姓名，171240511@smail.nju.edu.cn

2020 年 5 月 31 日

1 综述

在老师所给框架代码上进行修改完善,完成了所有要求的指令输入输出,GUI界面实现了所CLI部分的所有功能,包括设置画笔颜色、重置画布、保存画布、两种算法绘制线段、两种算法绘制多边形、绘制椭圆、两种算法绘制曲线、平移、旋转、缩放和两种算法裁剪线段.

额外实现了点击画布右、下、右下角边框调整画布大小.

2 算法介绍

2.1 绘制线段

2.1.1 DDA算法

基本思想是记下起点,然后让长的一边变量不断加一,短的一边则不断加斜率乘1后近似为int值.代码参照[6].

2.1.2 Bresenham算法

基本思想其实和DDA是一样的.只是Bresenham算法通过变形避免了浮点运算.这样一来,既提高了运算的效率,又避免了浮点数不断累加造成的长线段误差.所以本质上,Bresenham是DDA算法的优化形式.代码参照[6].

2.1.3 代码处理

- 特判使得能够处理两端点相等的情况.
- 因为讲义中说不要求像素级一致,所以和伪代码一样只亮一边端点,即线段点的范围$[P_0, P_1)$.

2.2 绘制多边形

默认指令中的点是排好序的, 直接调用线段绘制算法按顺序连点.

2.3 绘制椭圆

使用ppt上的中点椭圆算法, 先画出1/4椭圆, 然后对称. 画1/4椭圆时,根据是否到达切线斜率为-1的位置, 来判断选点的方向是基于x还是基于y, 再根据实际椭圆上的点离哪个点更近取近似.

实际代码要先求出中心和长短轴, 然后以中心为原点求点, 再映射回原来的坐标系.

对称的时候考虑了x轴和y轴上的特例, 不然x=0和y=0的点对称后相当于1个点出现了两次.

代码流程参考书上ppt相关部分.

2.4 绘制曲线

2.4.1 Bezier曲线

书上给出了Bezier曲线关于控制点的参数方程和矩阵形式, 但直接这样计算参数u对应的点效率太低, 于是又给出了de Casteljau递推算法来快速计算一个u值对应的曲线上点的坐标.

有了这个递推算法后, 由于参数u位于[0,1]之间,理论上我们只要将[0,1]分割的足够小,就可以得到精度足够高的一条Bezier曲线.

但这样分割, 为了确保精度, 我们往往要将步长设置的非常小, 从而导致计算次数过多. 于是书上给出了一种二分逼近的办法.也就是每次先计算u=1/2处的点, 然后用这个点将曲线分为两段, 再对这两段曲线进行u值的二分, 重复这个二分的操作, 最后每段曲线的控制多边形会很接近理论曲线, 就可以直接拿控制多边形当作实际曲线.

实现参考[8]中6.3.3和6.3.4.

2.4.2 B-spline曲线

B样条曲线的绘制本质上和Bezier没什么不同, 都是要根据递推式求参数u对应的离散点. 但具体实现起来还是有所不同的.

B样条曲线实际上是按照次数进行分段绘制的, k次B样条就是每k+1个控制点一段, 相邻两段有k个公共控制点, 这也解释了为什么它具有局部性.

不过, 由于它的控制多边形不像Bezier曲线那样可以随着二分不断逼近, 在具体绘制一段B样条曲线时, 只能设置步长密集取点, 这既导致了步长取密时曲线会变粗(重叠部分), 也导致了效率的降低.

实现参考[7].

2.4.3 代码处理

B-spline曲线绘制其中一段时, 目前选择取100个离散点, 再往大取GUI绘制时就会很卡.

2.5 平移

平移部分参考ppt, 直接令各图形的控制点偏移[dx,dy]即可.

2.6 旋转

旋转部分参考ppt, 直接套用公式, 各图形的控制点相对于旋转中心[x,y]顺时针旋转r度即可.

2.7 缩放

缩放部分参考ppt, 直接套用公式, 控制点相对于缩放中心[x,y]缩放s倍即可.

2.8 裁剪

2.8.1 Cohen-Sutherland算

这个算法的特别之处在于位运算的思想,用4位二进制数表示左右上下四个位置的是否. 接着再分类讨论不同情况下线段裁剪时, 就可以用位运算来判断怎么裁剪线段. 它会重复裁剪直到线段完全在窗口内或者完全在窗口外. 代码参考[5].

2.8.2 Liang-Barskey算法法

这个算法不同于Cohen-Sutherland算法, 它希望一步直接将线段裁剪出来, 而不是重复裁剪. 它用$P = P_1 + u * (P2 - P1)$来表示直线P, 并分情况判断和裁剪窗口相交点处的u_1, u_2值, 根据u_1, u_2的情况来判断结果. 代码参考[2]和ppt.

3 系统介绍

3.1 CLI框架

3.2 GUI框架

gui部分中目前仍按原始代码用QListWidget记录图元, 能鼠标控制两种算法绘制线段, 多边形的鼠标绘制尚未完成.
画笔颜色选择参考了[3], 画笔颜色选择应在绘制图元之前
重置画布
原来的代码由于点击绘制线段和在画布上press mouse都会使id加1,因此图元编号并不连续,对此进行了修改.
保存画布的代码参考了[1]
gui绘制多边形也实现了,最后一条边离起始点5×5范围内闭合.
图形没完成的会在点击任意按钮时完成
平移时按下的点为[x0,y0], 移动时为[x1,y1], 计算出dx,dy并paint.目前边界得松手才会更新
编辑时有些操作参数较多,需要点击两次,比如旋转,这时如果点击一次就切走做别的,第一次点击选择的点仍然会保留在对应item中.但是每次进行编辑操作前都会调用selectedTransClear()函数,确保选择的item的相应参数恢复到初始状态.
缩放基本的实现思路和旋转一致.

裁剪后,如果线段什么点也没剩下,仍然保留QListWiget中该图元,但是无法对其进行任何操作

4 总结

4.1 遇到的一些问题

- gui中sys.exit(app.exec_())改为app.exec_(),自己手动退出,不然Spyder运行手动退出时会报错. 此外,还得加上del app,不然重新运行的时候也会有点问题.参考[4]
- 框架代码是鼠标一动就将MyItem中的图元绘制, 这导致绘制直线的时候直线随着拖动不断绘制,这包括了最开始端点重合的情况, 对于DDA会出现除0错误,因此要特判.
- 画笔颜色选择分两步,一个是会用QColorDialog调出调色版,另一个是要将画笔颜色传给MyItem,我用全局变量g_penColor传递.

参考文献

[1] How to save images from qgraphicsview? https://stackoverflow.com/questions/35191327/how-to-save-images-from-qgraphicsview?r=SearchResults.

[2] liang-barskey算法. https://blog.csdn.net/sinat_34686158/article/details/78745492.

[3] Pyqt5中文教程. http://code.py40.com/pyqt5/24.html.

[4] simple ipython example raises exception on sys.exit(). https://stackoverflow.com/questions/10888045/simple-ipython-example-raises-exception-on-sys-exit.

[5] 运用python实现cohen-sutherland直线段裁剪算法. https://www.cnblogs.com/aliali/p/12623642.html.

[6] David F.Rogers, 石教英, and 彭群生. 计算机图形学的算法基础. 机械工业出版社, 2002.

[7] HachiLin. B样条曲线——de boor递推算法实现. https://blog.csdn.net/Hachi_Lin/article/details/89812126.

[8] 孙正兴. 计算机图形学教程. 机械工业出版社, 2006.