《计算机图形学》11月实验报告

刘扬 171850524

一、综述

实验完成进度：

1、十月完成内容

(1)鼠标绘制直线、圆、椭圆、多边形

(2)鼠标打开、保存、重置画布

(3)鼠标点击选择颜色、线条粗细

2、十一月完成内容

(1)鼠标拖拽图元进行平移，点击图元进行旋转与缩放

(2)鼠标拖拽产生矩形作为裁剪窗口对图元进行裁剪

(3)鼠标点击图元内部进行染色

二、算法介绍

1、线生成的DDA算法

(1)设线段起点和终点分别为(x0,y0),(x1,y1)，根据这两点求出直线方程ax + by +c = 0。

(2)建立函数F(x,y) = ax + by + c；根据F的正负，可以得知点(x,y)与上述直线的关系：

F(x,y) > 0，点(x,y)在直线上方

F(x,y) = 0，点(x,y)在直线上

F(x,y) < 0，点(x,y)在直线下方

(3)不妨假设直线斜率介于0到1之间：

画线时，若当前点为(xi ,yi )，则下一个点在P1(xi+1,yi)和P2(xi+1,yi+1)之间选 择。令M为这两个点的中点(xi+1,yi+0.5)，将M带入函数F，求得函数值为d。

根据d取值得正负从待选择点中选择点：

d < 0，M在直线下方，直线离P2更近，选择点P2；

d = 0，M在直线上，直线距离P1，P2一样近，选择P1；

d > 0，M在直线上方。直线距离P1更近，选择P1。

(4)分析d的递推表达式：

若d >= 0，则选择P1，之后选择哪个点时，d1 = F(xi+2,yi+0.5) = d + a;

若d < 0，则选择P2，之后选择哪个点时，d1 = F(xi+2,yi+1.5) = d + a + b；

初始情况为d0 = a + 0.5b;

(5)之后通过迭代的方式更新d的值，同时选择在线上的点，便可以画出线段。为了避免 浮点数的运算，使用2d来代替d，提高运算速度。

优点：DDA算法利用光栅特性消除了直线方程中的乘法。

缺点：浮点数在迭代中的误差会逐渐积累，使像素偏离实际的直线。

浮点数的运算和取整操作比较耗时。

2、线段生成的Bersenham算法

(1)假设线段斜率在0到1之间，线段左端点是(x0,y0)。沿x方向单位间距取样。

(2)在第k步选择点(xk,yk)，则第k+1步的点将在(xk+1,yk+1)和(xk+1,yk)中进 行选择。

在x = xk+1时，在线段上有y = mxk+1 + b = m(xk+1)+b;

计算两个候选点与线段上点的垂直偏移：

d1 = y-yk = m(xk+1)+b-yk

d2 = yk+1-y = yk+1-m(xk+1)-b

距离差分d1-d2 = 2m(xk+1)-2yk+2b-1；将m = Δy/Δx带入，

得：Δx(d1-d2) = 2Δyxk-2Δxyk+c，此方程仅包含整数运算。

(3)令pk = Δx(d1-d2) = 2Δyxk-2Δxyk+c，这是第k步得决策参数。

pk > 0，选择较高像素

pk < 0，选择较低像素

决策参数可以通过递推式计算：

若pk > 0，pk+1 = pk + 2Δy - 2Δx

若pk < 0，pk+1 = pk + 2Δy

(4)沿x方向单位间距取样，重复上述过程Δx次便可以画出线段。

3、中点圆生成算法

使用点与圆心距离得平方作为判定依据，避免了平方根运算。适用于所有的圆锥曲 线，且误差在半个像素内。

考虑圆的对称性，仅需要计算八分之一圆弧得像素位置即可，其他部分通过对称计 算得到。以Π/4到Π/2的八分之一圆弧为例：

(1)定义圆函数：fcircle(x,y) = x2+y2-r2

fcircle(x,y) < 0，(x,y)位于圆周内

fcircle(x,y) = 0，(x,y)位于圆周上

fcircle(x,y) > 0，(x,y)位于圆周外

(2)决策参数pk = fcircle(xk+1,yk-0.5);代表候选两点的中点距离圆心的距离

pk > 0，中点在圆外，选择点(xk+1,yk-1)

pk = 0，中点在圆上，选择点(xk+1,yk-1)

pk < 0，中点在圆内，选择点(xk+1,yk)

决策参数可以通过递推式计算：

若pk < 0，pk+1 = fcircle(xk+2,yk-0.5)

若pk > 0，pk+1 = fcircle(xk+2,yk-1.5)

4、中点椭圆生成算法

椭圆生成算法与圆类似，标准位置的椭圆四分象限对称，只需要计算第一象限的椭 圆像素位置，其他三个象限通过对称计算得到。

(1)定义椭圆函数fellipse(x,y) = ry2x2+rx2y2-rx2ry2

fellipse(x,y) > 0，(x,y)位于椭圆外

fellipse(x,y) = 0，(x,y)位于椭圆上

fellipse(x,y) < 0，(x,y)位于椭圆内

(2)第一象限区域分割条件是曲线斜率为-1处，每次迭代都需要检测当前位置的斜率， 区域交替条件2ry2x >= 2rx2y

决策参数根据pk = fellipse(xk+1,yk-0.5)定义;

在区域1(斜率介于0到-1之间)：

p10 = ry2-rx2ry+rx2/4

若p1k < 0，选择点(xk+1,yk)，plk+1 = p1k+2ry2xk+1+ry2

若p1k > 0，选择点(xk+1,yk-1)，p1k+1 = p1k+2ry2xk+1-2rx2yk+1+ry2

循环至2ry2x >= 2rx2y

在区域2：

p20 = ry2-rx2ry+rx2/4

若p2k < 0，选择点(xk,yk-1)，p2k+1 = p2k-2rx2yk+1+rx2

若p2k > 0，选择点(xk+1,yk-1)，p2k+1 = p2k+2ry2xk+1-2rx2yk+1+rx2

循环至点(rx,0)

(3)

5、基本几何变换

(1)平移

设平移向量是(tx,ty)，点的原位置是(x,y)。

则平移后点的位置是(x+tx, y+ty)。

(2)旋转

点(x,y)以(xr,yr)作为旋转中心，顺时针旋转角度为θ后

x1 = xr + (x – xr)cosθ - (y – yr)sinθ

y1 = yr + (x – xr)sinθ + (y – yr)cosθ

(3)缩放

点(x,y)相对点(xf,yf)缩放，缩放系数时s时：

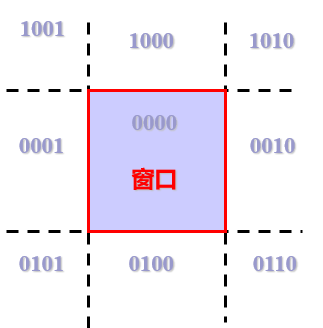
x1 = x\*s + xf(1-s)

y1 = y\*s + yf(1-s)

6、Cohen-Sutherland裁剪算法

最早、最流行的线段裁剪算法。核心思想是通过编码测试来减少计算交点的次数。

(1)将区域分为九个部分，4位二进制码代表了该部分与裁剪窗口的位置关系。



点在窗口上方的区域，编码第1位置为1，否则置为0；

点在窗口下方的区域，编码第2位置为1，否则置为0；

点在窗口右边的区域，编码第3位置为1，否则置为0；

点在窗口左边的区域，编码第4位置为1，否则置为0。

(2)通过区域码可以快速判断两种情况的线段

完全在窗口内部的线段，其两个端点的编码都是0000；

完全在窗口外部的线段，其两个端点至少在某一位上都是1。

(3)对于其他不能判断的线段，需要进行求交运算

按照左-右-上-下的顺序用裁剪窗口边界检查线段端点，确定需要裁剪掉的部 分，剩余部分继续与其他裁剪边界进行比较和求交，直至线段被完全遗弃或者找 到位于窗口内的一段线段为止。

7、Liang-Barsky参数裁剪算法

(1)设线段端点为P1(x1,y1)，P2(x2,y2)；线段的参数方程为：

x = x1 + u(x2 – x1) = x1 + uΔx

y = y1 + u(y2 – y1) = y1 + uΔy

0 <= u <= 1

(2)设裁剪窗口的左下角和右上角的坐标分别是(xmin,ymin)和(xmax,ymax)。

则位于裁剪窗口内的点的坐标满足：

xmin <= x1 + uΔx <= xmax

ymin <= y1 + uΔy <= ymax

上述四个不等式可以表示为：

u\*pk <= qk, k = 1, 2, 3, 4

p1 = -Δx, q1 = x1 - xmin

p2 = Δx, q2 = xmax – x1

p3 = -Δy, q3 = y1 - ymin

p4 = Δx, q4 = ymax – y1

(3)若pk = 0，则线段平行于窗口某边界

qk < 0，则线段在窗口边界外，应被舍弃；

qk >= 0，则线段在窗口内

若pk < 0，线段从裁剪边界延长线的外部延伸到内部；

若pk > 0，线段从裁剪边界延长线的内部延伸到外部。

(4)在pk ≠ 0时，可以计算出直线与窗口边界直线的交点为u = qk / pk；

对于pk < 0，计算rk = qk/pk，u1取0和rk中的最大值；

对于pk > 0，计算rk = qk/pk，u2取1和rk中的最小值；

若u1 > u2，则线段在裁剪窗口外，应当舍弃；否则，将u1和u2带入参数方 程，可以计算出裁剪后的线段的端点。

8、贝塞尔曲线生成算法

(1)n次Bernstein基函数: BEZi,n(u) = C(n,i)\*ui\*(1-u)n-i

(2)Berstein多项式作为混合函数描述贝塞尔曲线:

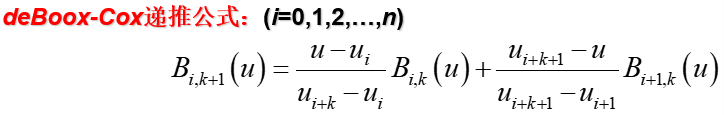


根据贝塞尔函数的混合函数形式的描述,可以知道贝塞尔曲线上的每一点都是 所有控制点的一个加权平均.

(3)算法实现时,需要让u从0逐渐变化到1,可以选择让u每次递增一个较小的值 如0.0001.针对每一个u,让k从0到n,计算每一个BEZk,n(u),这便是u取当前 值时,第k个控制点的权重,满足所有权重和是1.用所有控制点的加权平均作为u 值对应的贝塞尔曲线上的一点.这样就可以画出一整条贝塞尔曲线.

9、B样条生成算法

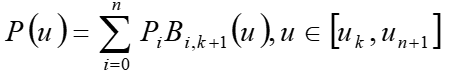
(1)B样条基函数:



u∈[ui,ui+1)时, Bi,1(u)=1,否则Bi,1(u)=0;

(u-ui)/(ui+k-ui)为0/0时,取值为0;

(2)对于n+1个控制点的k次B样条,需要在其参数轴上选择n+k+2个点进行分 割,一般情况下,u0=0,un+k+1=1,所有点之间等距.B样条的定义域是[uk,un+1],其 参数曲线是:



在u的定义域上,让u从uk逐渐变化到un+1,每次都计算对应的点,最终将得到B 样条曲线.

三、系统介绍

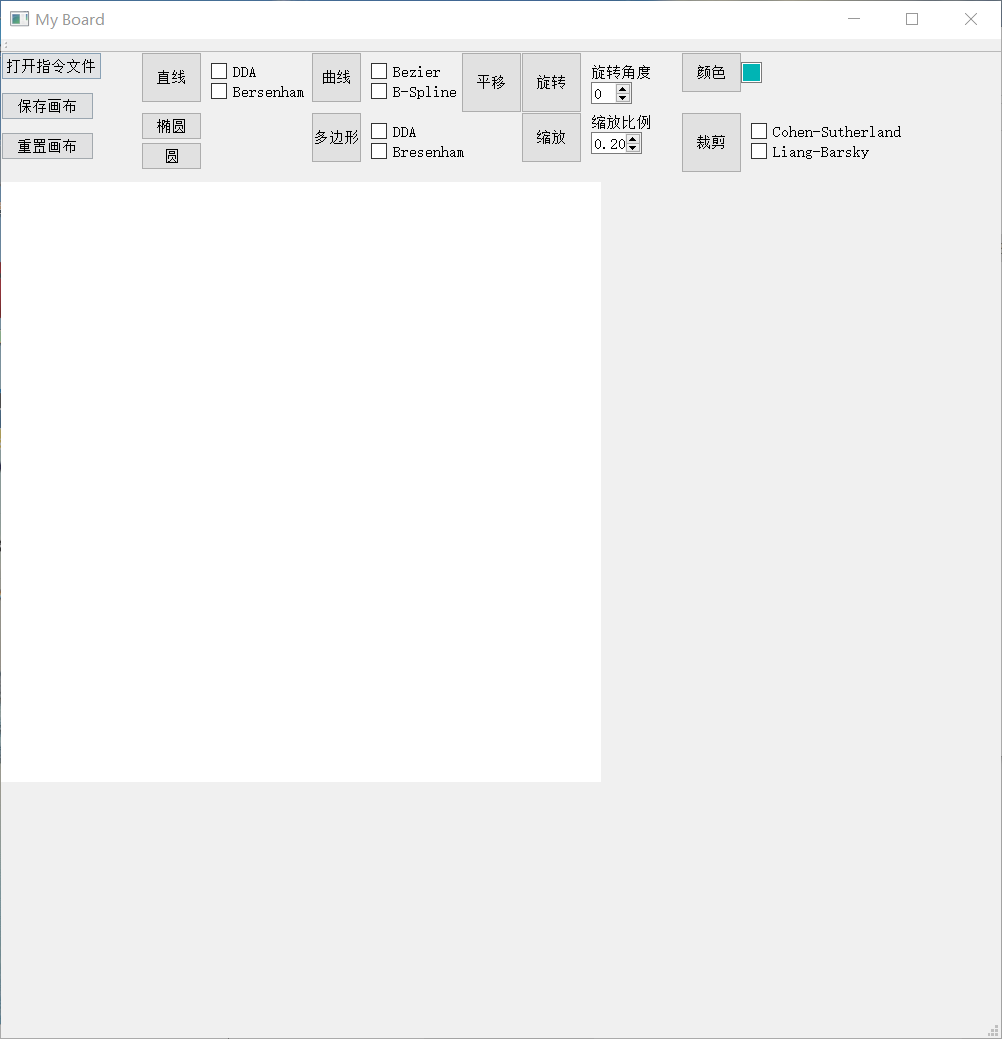
系统主要由以下三部分构成：

1、主窗口部分，对应类mainwindow

将主窗口视为画板，所有鼠标操作都将通过主窗口进行。

主窗口主要包含画布和许多按钮，点击按钮再使用鼠标在画布上进行操作。

运行时的主窗口如下：



mainwindow类中主要函数包括四个，分别是对paintEvent、mousePressEvent、mouseMoveEvent和mouseReleaseEvent的重写。paintEvent重写画图事件，画图操作必须在此函数中进行调用。剩余三个函数是对鼠标事件的重写，实现鼠标操作。

除此以外，窗口的每一个按钮都对应一个槽函数，槽函数实现都比较简单，将处理工作主要集中到三个处理鼠标事件的函数中。

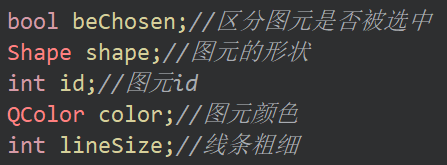
2、图元描述部分，对应graphUnit文件夹中所有的类

使用类graphUnit描述所有图元共有的属性，每个特定的图元都是graphUnit类的 派生类。

这一部分的框架如下：

|  |  |
| --- | --- |
| 文件 | 功能 |
| graphUnit(.h .cpp) | 图元类，描述所有类型的图元共有的属性 |
| line(.h .cpp) | 直线类 |
| circle(.h .cpp) | 圆类 |
| ellipse.h(.h .cpp) | 椭圆类 |
| polygon(.h .cpp) | 多边形类 |
| rectangle(.h .cpp) | 矩形类 |

图元共有的属性目前包括以下属性：



区分图元是否被选中用于对图元的平移、旋转和缩放操作，作用与使用命令行交互 时的id相同。这些操作都是针对某个特定的图元进行的，所以有必要记录鼠标当前针 对哪个图元进行操作。

shape指示该图元的类型。当需要针对某类图元特殊处理时，依赖此信息强制类型 转换graphUnit类的指针。

color和lineSize是描述绘制图元的线条的属性的，线条属性包括颜色和粗细。

3、图元控制部分

类Controller主要负责维护存储图元的数据结构和调用图元绘制算法。

包含的函数有：

|  |  |
| --- | --- |
| addGraph | 向图元向量中添加新图元 |
| removeGraph | 从图元向量中移除图元 |
| reset | 重置画布 |
| paint | 绘制图元 |
| choseGraph | 选中图元 |
| getAngle | 计算向量与x轴正方向夹角 |

类Transmation用于图元的二维变换，包括平移、旋转、缩放与裁剪。

包含的函数有：

|  |  |
| --- | --- |
| translate | 图元平移 |
| rotate | 图元旋转 |
| scale | 图元缩放 |
| clipCohenSutherland | Cohen-Sutherland线段裁剪算法 |
| clipLiangBarsky | 梁友栋-Barsky参数裁剪算法 |

类Dyer用于给图元染色。

包含的函数有：

|  |  |
| --- | --- |
| inOutTest | 检测鼠标点击的点是否在图元内部 |
| dyeGraph | 对图元进行染色 |

四、总结

实验过程中遇到的问题：

1、使用鼠标拖动进行旋转和缩放时，效果不佳。

鼠标操作如何得到旋转和缩放的参数由个人定义，且无法得到精确的参数。鼠标移 动较快时，效果与理想情况不符合。故后来改为通过键盘输入参数，鼠标点击图元进行 对应的操作。

2、编码裁剪算法，当无法快速确定线段与裁剪窗口之间的关系时，需要用裁剪边界检 查线段，这部分需要考虑的情况较多，实现中出现很多次bug。

3、鼠标画椭圆圆出现卡死的情况

对于区分两个区域的边界条件，我未严格区分>与≥，所以当椭圆长半轴和短半轴 都是0时，会产生死循环。

4、后期实验过程中，将图元类的成员全部修改为了public，这导致部分函数不是必需 的。我准备在之后实验中删除冗余的函数。

参考文献

调色板的调用 <https://blog.csdn.net/achenwenping/article/details/34105961>

重写鼠标事件<https://blog.csdn.net/naibozhuan3744/article/details/79144667>

鼠标绘图时的实时显示<https://blog.csdn.net/marlene0312/article/details/5221261>

保存图片<https://blog.csdn.net/weixin_39485901/article/details/88056369>

主窗口坐标转换为画布坐标 <https://blog.csdn.net/owen7500/article/details/51035078/>

CheckBox用法 <https://blog.csdn.net/liang19890820/article/details/50976944>

计算两个向量顺时针的夹角 ：

<https://www.ilovematlab.cn/forum.php?mod=viewthread&s_tid=NumBestAnswers&tid=477717&page=1>

绘图算法来源于课本和课程课件：

DDA算法 课本page59

Bersenham画线算法 课本page60

中点圆生成算法 课本page64

中点椭圆生成算法课本page66

二维平移变换 课本page105

二维旋转变换 课本page106

二维缩放变换 课本page106

编码裁剪算法 课本page93

梁友栋-Barsky线段裁剪算法 课本page95