计算机图形学 系统技术报告

学号 161220175， 姓名 郑天烨
161220175@smail.nju.edu.cn

December 20, 2018

1 综述

针对《计算机图形学》课程开发的几何图形处理系统实现的功能包括：在UI界面中通过鼠标点击拖拽等方式可视化地输入二维图形的功能；编辑最近输入的二维图形的功能；裁剪直线的功能，支持的图形包括任意涂鸦、直线、矩形、圆、椭圆、多边形、填充多边形、B样条曲线、Bezier 曲线；变换二维图形的功能；用文件/路径选择器将绘制出来的图形保存为图像以及选取载入显示图片文件的存储和读取二维图形的功能；载入并显示一个 OFF 格式的三维模型的功能；

此外，系统中还支持选择画笔的形状大小，颜色，选择多边形填充颜色，清理画板，并提供了一个美观的用户使用界面。

在实现算法时要考虑许多实际的问题，在图形系统的画布上建立坐标系，由于在画布上是以像素的形式呈现、编辑以及输入输出图形的，所以图形绘制算法的一个主要的任务是将图形的数学表达式尽可能准确地绘制成画布上的像素点

2 算法介绍

2.1 二维图形的输入功能

2.1.1 直线

输入直线(或者线段)采用数值差分分析DDA算法，主要过程是，鼠标点击确定直线的起点，然后拖动鼠标实时改变直线的方向，对每一个时刻，鼠标起点与目前所在的点就确定了当前的直线。由两点坐标就可以求出直线方程。求出直线的斜率，对于具有不大于 1 的正斜率的直线，在 x 方向以单位间隔取样，以增量形式顺序计算每个 y 的值，依据该增量在 y 轴寻找最接近路径的整数值，这样选取坐标轴的目的是为了取较多的样本点，绘制出来的像素点不会过于稀疏.对于具有大于 1 的斜率的直线，在 y 方向以单位间隔取样算 x 方向的增量，在 x 方向取最接近直线的点；对于具有绝对值不大于 1 的负斜率的直线，在 x 方向取增量；对于具有绝对值大于 1 的负斜率的直线，在 y 方向取增量.

2.1.2 矩形

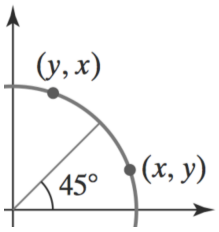
矩形的绘制只要确定对角的两个顶点就比较容易完成。在确定两顶点的坐标后，由于矩形的边都是水平或垂直的线，与像素点重合，故只需要单个坐标轴递增绘出即可.

2.1.3 圆/椭圆

圆和椭圆的绘制应用中点生成算法。中点生成算法可以避免平方根运算，直接采用像素与圆距离的平方作为判决依据，通过检验两个候选像素中点与圆周边界的相对位置关系，即圆周边界内或圆周边界外，来选择像素.

首先通过鼠标点击和拖动可以确定圆心和圆边界上的某个点，进而可以确定任意时刻的圆半径以及圆函数，以圆函数作为决策参数，将候选像素的中点位置的数据代入圆函数中，依据函数符号来判断其与圆的相对位置关系。若以圆心为原点建立坐标轴，考虑不同象限内圆弧的生成，结合候选像素中点在圆函数计算得的符号决定保留低像素还是高像素，椭圆也可以采用类似的算法。

由于圆和椭圆的对称性，椭圆可以只考虑一个象限内的生成，而圆可以只考虑 1/4 象限的生成，剩下的部分通过对称运算来节省计算量.



圆的对称性

由圆的方程

$$f_{circ}(x,y) = x^2 + y^2 - r^2$$

得到判定函数 $f_{circ}(x,y) < 0$ 时点 (x , y) 位于圆边界内，$f_{circ}(x,y) = 0$ 时点 (x , y) 位于圆边界上，$f_{circ}(x,y) > 0$ 时点 (x , y) 位于圆边界外。一般的决策过程是，假设在第一象限中画了点 (x_k, y_k) , 下一步需要确定点 $(x_k + 1, y_k)$ 和 $(x_k + 1, y_k - 1)$ 哪一个更接近圆。将这两个点的中点代入判定函数求决策参数

$$p_k = f_{circ}(x_k + 1, y_k - \frac{1}{2}) = (x_k + 1)^2 + (y_k - \frac{1}{2})^2 - r^2$$

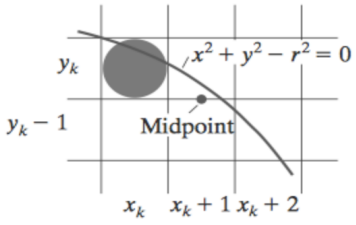
类似地，下一步的决策参数为

$$p_{k+1} = f_{circ}(x_{k+1} + 1, y_{k+1} - \frac{1}{2}) = (x_{k+1} + 1)^2 + (y_{k+1} - \frac{1}{2})^2 - r^2$$

由以上两式得

$$p_{k+1} = p_k + 2(x_k + 1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1$$

其中 y_{k+1} 是 y_k 还是 y_{k-1} , 取决于 p_k 的符号，故可以由递推运算避免复杂的乘方运算，由决策参数决定下一个点的位置，画椭圆的方法也是类似的



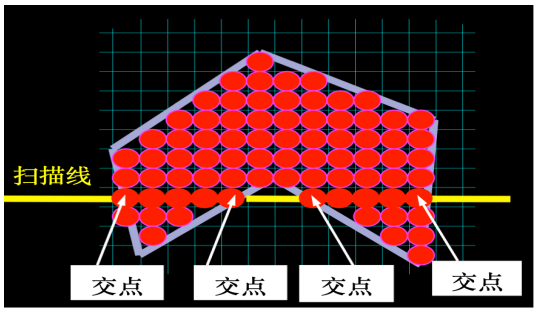
2.1.4 多边形

对于多边形，可以采取直线（线段）拼接的方法来生成，关键在于记录绘制过程中鼠标点击的点作为下一个线段的起始点；多边形中每两个顶点确定后就用绘制直线的算法来绘制

2.1.5 填充区域

区域填充在实现上的难点在于需要识别出哪些是多边形或者闭合曲线内部的点，具体地，在判断扫描线与多边形边的交点时，由于多边形的边不是纯粹的数学上的直线，而是看起来像一小段一小段拼起来的，所以一条扫描线与多边形的交点可能有连续的点。在代码中采用的策略是，区别存储边的点和顶点，若在扫描中有连续的边的点，则连续的点视为一个点，若遇到顶点，判断顶点是否为“尖端”的点，是则作为两个位置相同的点储存，否则作为一个点储存。

区域填充采用扫描填充图元生成的方法。对每条横越多边形的扫描线，先确定其与多边形的交点的位置（一般有偶数个交点，若只有 1 个交点，则说明该行有一个尖角，该行就不填充）并把交点按横坐标排序，自左向右分对配对存储。最后对交点坐标取整，并将每对交点之间的一行的区域填充颜色即可



需要考虑扫描线与顶点相交的处理。需要区别共享顶点的两条表位于扫描线的同侧还是异侧

2.1.6 贝塞尔曲线

当只有两个控制点 P_0 和 P_1 时，曲线点的生成公式是

$$P_0^1 = t * P_0 + (1-t) * P_1, (0 < t < 1)$$

表现为经过P_0 和 P_1的线段，当贝塞尔曲线有三个控制点P_0，P_1 和P_2时，除了上式，还有

$$P_1^1 = t * P_1 + (1-t) * P_2, (0 < t < 1)$$

$$P_0^2 = t * P_0^1 + (1-t) * P_0^2$$

依次类推，贝塞尔曲线的递推计算公式是

$$P_i^k = \begin{cases} P_i & k = 0 \\ (1-t)P_i^{k-1} + tP_{i+1}^{k-1} & k = 1,2,\cdots,n,\ i = 0,1,\cdots,n-k \end{cases}$$

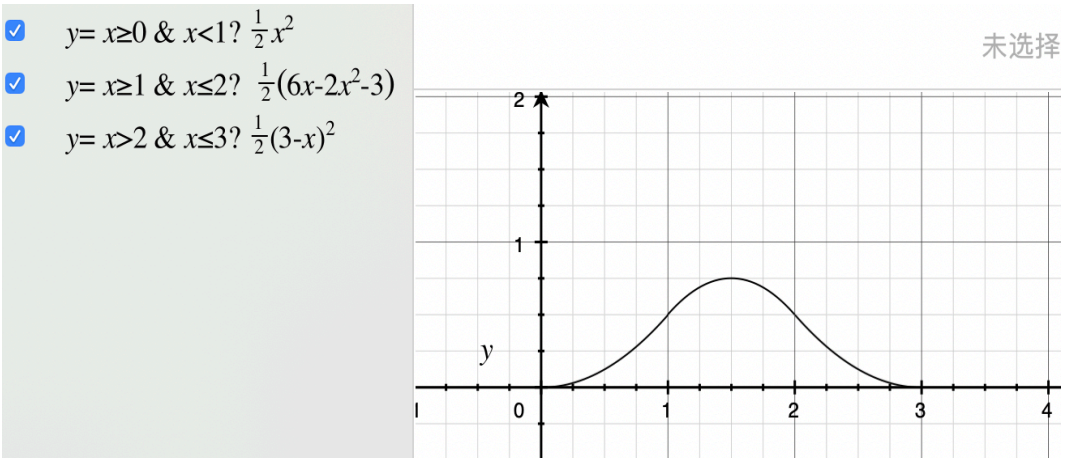
由于有递推公式，具体实现时可以用递归函数来实现

2.1.7 B样条曲线

B样条曲线的定义是，给定 n+1 个控制点 $\{P_0, P_1, ..., P_n\}$，每个控制点都对应着一个基函数 $N_{i,p}(u)$，其中 u 为自变量，i 为第 i 个节点， p 为曲线的次数（次数 = 阶数 - 1），则B样条曲线为

$$C(u) = \sum_{i=0}^{n} N_{i,p}(u) P_i$$

代码中实现的是二次 B样条曲线，所以选用 u 的定义域的下限为 2，不规定上限，每次增加一个固定的量 0.1，也就是说 B样条曲线上的点由所有的控制点来加权决定，对于基函数 $N_{i,p}(u)$ 可以用 Cox-de Boor 公式得出，代码中实现的二次曲线的第 1 个基函数如下图



第 i 个基函数的图将上图向 x 轴正方向平移 i-1 个单位得到

2.2 二维图形的编辑/裁剪功能

直线裁剪采用Cohen-Sutherland裁剪算法。该算法的核心是通过编码测试来减少需要计算交点的次数。在确定裁剪窗口后，依据窗口将画板分成若干个区域

1001	1000	1010
0001	0000	0010
0101	0100	0110

其中中间的矩形就是裁剪框所在的区域。对于每一条需要处理的直线，首先将其线段端点按照区域赋予四位二进制码，即区域码 区域码各位从右到左编号：位1：上边界；位2：下边界；位3：右边界；位4：左边界；区域码为 1 表示端点落在相应的位置上，为 0 表示端点不在相应位置上。算法接下来就能根据线段端点的区域码快速进行判断。

- 完全在裁剪窗口内的线段：两端点的区域码均为 0000
- 完全在裁剪窗口外的线段：两端点区域码逻辑与结果不为0000（当中存在某一位均为 1）
- 不能完全确定的情况：将线段的端点与裁剪边界进行比较和求交，确定应该裁剪的部分，按照左-右-上-下的顺序不断进行，直到线段被完全舍弃或者剩下的线段完全在窗口中

2.3 二维图形的变换功能

平移 **图**形的二维变换通过对图形上的每一个点进行变换操作，把原先点擦除并在新点上绘制来实现，对于特定的像素点，平移通过坐标值的加减即可完成，沿 x 轴正向移动则将所有点的坐标的 x 分量加上位移量，沿 x 轴负向移动则 x 坐标减去位移量，y 轴类似，设 x 轴位移量为 x0 ， y 轴位移量为 y0 (可正可负)，则新坐标的计算公式很简单

$$x^{'} = x + x0$$

$$y^{'} = y + y0$$

旋转 **旋**转由图形上的所有点绕某个旋转轴（基准点）旋转一个旋转角度完成。设基准点为 $(x0, y0)$ ，旋转角为 θ ，旋转前的坐标为 $(x1, y1)$ ，旋转后的坐标为 $(x2, y2)$ ，则有

$$x2 = x0 + (x1 - x0)cos\theta - (y1 - y0)sin\theta$$

$$y2 = y0 + (x1 - x0)sin\theta + (y1 - y0)con\theta$$

这个变换也可以由矩阵来完成，旋转矩阵为

$$R = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & con\theta \end{bmatrix}$$

缩放 **缩**放由图形上的所有点的横坐标值和纵坐标值各乘以一个缩放系数 Sx 和 Sy 来完成。缩放系数小于 1 时将缩小对象，大于 1 时将扩大对象

$$x^{'} = x \times Sx$$

$$y^{'} = y \times Sy$$

写成矩阵形式

$$\begin{bmatrix} x^{'} \\ y^{'} \end{bmatrix} = \begin{bmatrix} Sx & 0 \\ 0 & Sy \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix}$$

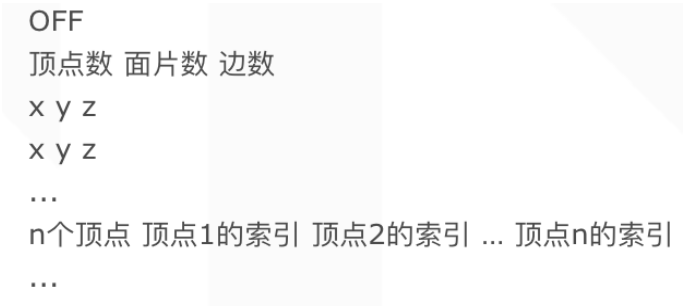
2.4 二维图形的存储和读取功能

直线的存储可以存储直线上的两个点，对于线段来说要存储开始坐标和结束坐标；矩形的存储可以存储两个对角点的坐标；圆的存储可以存储圆心和圆周上的一点或者圆心与半径；椭圆的存储需要存储椭圆的中心以及半长轴长半短轴长或者椭圆中心以及最右边和最高点的坐标；多边形的存储存储分别存储构成多边形的每一条线段；任意形状就存储构成形状的每一个点的坐标；填充区域在存储时只标识出需要填充，具体绘制时才进行填充。

若要将画板上的所有图形存储为图片，则将画板上的内容作为图片储存即可

2.5 三维模型的显示功能

需要载入并显示一个 OFF 格式的三维模型。物体文件格式 (.off) 文件用与表示给定了表面多边形的模型的几何体。OFF 文件是以 OFF 关键字开始的 ASCII 文件，接着的一行说明顶点的数量、面片的数量、边的数量，顶点按每行列出 x, y, z 的坐标。在顶点列表之后，面片按照每行一个列表。对于每个面片，顶点的数量是指定的，接下来是顶点的索引列表。



OFF格式文件示意

读取 OFF 文件时，按照格式解析并在空间直角坐标系中显示，具体是根据每个面的顶点索引列表由空间中的若干顶点确定一个面，在三维图形库建立的空间直角坐标系中绘制一个个的小面片，最后三维模型就由若干这种小面片构成。此外，由于面片比较细，为了让模型看起来更清楚，给相邻面片赋上不同而相近的颜色值。

3 系统介绍

3.1 平台

- **编程语言： Java**
- **JDK版本： jdk 10.0.2**
- **图形界面库： swing/awt**
- **三维图形库： OpenGL**
- **操作系统： MacOS Mojave**
- **IDE： Intellij IDEA 2018.2**

程序包含一个绘图面板以及旁边的控制面板，鼠标在绘图面板上操作以绘制出图形，控制面版包含一些按钮等控制组件，可以控制选择什么图形的绘制功能，还可以选择存储当前的面

板以供下次绘制，还可以打开新窗口显示三维图形；在绘制上，还有可以选择消除锯齿，控制画笔粗细和颜色等功能

4 总结

软件系统的描述如上，在实现完善过程中要一边上课学习课件中的算法知识和思想，这是实现的核心，同时要学习 java 及其图形库，这是实现系统功能的技术路线。整个项目写下来洋洋洒洒几千行代码，在学习图形界面的开发的过程中，我采用的方法是先根据教程写一个简单的 demo ，然后上手开始开发，边编码边查资料边学习，在尝试实现图形学领域的各个算法的过程中也熟悉了图形界面程序开发的流程步骤，在解决问题的过程中学习新知识。

参考文献

[1] 图标包. *http://ico.58pic.com/pack/378.html*.

[2] 音效网. *http://www.yinxiao.net/*.

[3] clairvoyant. 中点画圆算法. *http://www.cnblogs.com/clairvoyant/p/5528067.html*.

[4] deeebug. [转]off格式文件解析. *https://blog.csdn.net/hjq376247328/article/details/47039935*.

[5] heyuchang666. 计算机图形学(四)-几何变换-1-基本的二维几何变换(三)-缩放. *https://blog.csdn.net/heyuchang666/article/details/60964831*.

[6] hnfxs. 贝塞尔曲线原理(简单阐述). *https://www.cnblogs.com/hnfxs/p/3148483.html*.

[7] shenlan282. [转][图形学] b样条曲线 - 原理和c++实现的演示程序（附源码）. *https://blog.csdn.net/cgcoder/article/details/77884022*.

[8] Vicent-Chen. Bresenham算法理解. *https://blog.csdn.net/cjw_soledad/article/details/78886117*.

[9] ZYcat. [裁剪]线段的裁剪——cohen-sutherland算法及代码实现. *https://blog.csdn.net/ZY_cat/article/details/78290047*.

[4] [1] [2] [6] [7] [9] [5] [3] [8]