# 《计算机图形学》系统技术报告

张帅

(南京大学 计算机科学与技术系, 南京 210093)

摘 要: 本绘图系统采用了面向对象设计，以C++11为基础，交互使用了Qt，基础图形的绘制使用了OpenGL，直线绘制使用了Bresenham算法，圆的绘制使用了中点圆生成算法，椭圆绘制用了中点椭圆生成算法，曲线绘制使用了3次Bézier曲线，多边形填充使用了多边形扫描填充算法，线段裁剪使用了梁友栋-Barsky算法，多边形裁剪使用了逐边裁剪算法。

## 1 算法概述

### 1.1 绘制算法

#### 1.1.1 直线绘制：Bresenham算法

直线绘制采用了Bresenham算法，此算法优点在于避免了浮点运算，运算量小，从而可以得到实时的屏幕响应。

设起始点到终点坐标为,,, …, ，第步的决策参数为，递推公式如下：

|  |  |
| --- | --- |
|  | (1.1.1-1) |
|  | (1.1.1-2) |

此处以斜率在0到1之间为例：

1. 计算，；

2. 绘制直线左侧的起始点，并计算初始决策变量；

3. 根据上述递推公式从开始，每次x递增1，并在此x位置处对应的y和决策变量p，直至。

对于斜率大于1的情况，只需将递增变量由x改为y，递推公式中与的关系改为与的关系即可。

对于斜率为负的情况，只需将递推公式中的改为即可。

#### 1.1.2 圆的绘制：中点圆生成算法

圆的绘制采用了中点圆算法，同样避免了浮点运算，使用递推的方式，节省了计算量。

根据圆的对称性，只需要绘制1/8圆便可得到完整的圆。从圆心正上方顺时针绘制1/8圆，然后通过对称绘制剩余7/8圆。如下为1/8圆绘制方法：

设圆心坐标为, 半径为r, 起始点到终点坐标为, , …, ，第步的决策参数为，递推公式如下：

|  |  |
| --- | --- |
|  | (1.1.2-1) |
|  | (1.1.2-2) |

圆绘制过程如下：

1. 绘制圆心正上方的点, 计算初始决策参数

；

2. 在处按照如上的递推公式(1.1.2-1)与(1.1.2-2)进行计算，得到与，并绘制点，其中；

3. 将2中所得的点对称到其他7/8圆得到相应的对称点并绘制；

4. 重复2~3，直至。

#### 1.1.3 椭圆绘制：中点椭圆生成算法

将椭圆按照象限分为4个部分，先绘制出第一象限的部分，然后根据对称性绘制出其余3个象限的部分。

第一象限的部分分为两个区域：区域1：切线斜率小于1；区域2：切线斜率大于1. 生成过程如下：

1. 输入椭圆中心和长短轴径;

2. 计算得到中心在原点的椭圆上的第一个点=;

3. 根据公式(1.1.3-2)计算区域1中决策参数的初值为;

4. 在区域1中每个位置处，从开始，反复按照公式(1.1.3-3)更新决策参数，若决策参数小于0则不变，反之加1，一直循环到为止，记第一个满足此条件的点为();

5. 在区域2的每个位置处，从开始，反复按照公式(1.1.3-6)更新决策参数，若决策参数小于0则不变，反之加1，一直循环到()为止;

6. 根据第一象限的点对称画出其他三个象限的点;

7. 将每个计算出的像素位置()平移到中心到()的椭圆轨迹上，并按平移坐标值画点()，其中.

区域1：

|  |  |  |
| --- | --- | --- |
| 决策参数： | . | (1.1.3-1) |
| 初始取值： | . | (1.1.3-2) |
| 更新公式： | . | (1.1.3-3) |

区域2：

|  |  |  |
| --- | --- | --- |
| 决策参数： |  | (1.1.3-4) |
| 初始取值： | . | (1.1.3-5) |
| 更新公式： | . | (1.1.3-6) |

#### 1.1.4 曲线绘制：三次Bézier曲线

在三阶Bézier曲线中有四个控制点，设第i个控制点为，则和分别为曲线的起点和终点。曲线上点的计算公式为：

|  |  |
| --- | --- |
|  | (1.1.4-1) |

将从0累计到1，从而可以计算得到曲线上从到的所有点。（其中每次增加0.001，以保证曲线的连续）

### 1.2 填充算法

#### 1.2.1 多边形填充：多边形扫描填充算法

先后使用了有序边表与活化边表，得到各扫描行的填充范围，从而进行填充。

边表使用了如下数据结构：

|  |
| --- |
| struct Edge  **{**  Edge**()** **{}**  Edge**(**double xi**,** double dx**,** int ymax**)**  **{**  **this->**xi **=** xi**;**  **this->**dx **=** dx**;**  **this->**ymax **=** ymax**;**  **}**  double xi**; //本扫描行该边的起始点横坐标**  double dx**; //该边y每增加1，x的增加量，即斜率的倒数**  int ymax**; //该边最高点的y值**  **};** |

a. 生成有序边表：扫描多边形中所有边，跳过所有点y值相同的水平边，将所有边最下方的点加入到有序边表中，填入对应的xi，根据该边起始点计算出dx。若该边最高点在上下侧，则ymax为该边的最高点y值；若该边最高点在左右两侧，则ymax为该边最高点y值-1（为了防止重复绘制顶点带来的内外侧判断错误）。

b. 根据有序边表生成活化边表：扫描当前扫描线及以下的所有有序边表，将ymax大于等于当前y值的Edge项加入到本扫描线对应的活化边表中，加入时xi需要根据当前y值和有序边表中的xi，dx计算，其余项保持不变。然后根据xi对当前扫描线的Edge项从小到大排序。

c. 根据活化边表生成各扫描行填充范围：遍历当前扫描行所有Edge项，两两扫描，从索引为偶数的项开始，填充到下一个项的xi为止。

#### 1.2.2 圆的填充算法

圆的填充实现较为简单，通过圆的方程计算出圆轮廓上纵坐标为对应左右侧点的横坐标值和，填充和中间的所有点即可。设圆心为，半径为，和计算公式如下

|  |  |
| --- | --- |
|  | (1.2.2-1) |

### 1.3 变换算法

图形变换包括平移、旋转和缩放。设原先点为，变换后点为。

#### 1.3.1 平移

设平移向量为，有

|  |  |
| --- | --- |
|  | (1.3.1-1) |

#### 1.3.2 旋转

设旋转基准点为，顺时针旋转角为，有

|  |  |
| --- | --- |
|  | (1.3.2-1) |

#### 1.3.3 缩放

设缩放基准点为，在方向和方向的缩放系数分别为和，有

|  |  |
| --- | --- |
|  | (1.3.3-1) |

### 1.4 裁剪算法

设裁剪窗口矩形横坐标范围为, 纵坐标范围为.

#### 1.4.1 线段裁剪：梁友栋-Barsky算法

设线段为, , ,

设变量表示第侧为从内到外/从外到内，表示外->内；表示线段平行于该边界；表示内->外。

|  |  |
| --- | --- |
|  | (1.4.1-1) |

定义变量如下

|  |  |
| --- | --- |
|  | (1.4.2-2) |

设直线的参数，时在点，时在点.

设，则为直线与第i边界交点对应的直线参数u的值。

设线段裁剪之后两端点对应值分别为. 不妨设.

对四个方向的和进行遍历，

若，说明线段平行于该侧边界。若此时，说明线段在外部，则整个线段均被裁掉，直接返回；否则线段在内部，本边界与线段没有交点，不做处理；

若，说明从外到内穿过该边界，令;

若，说明从内到外穿过该边界，令

遍历完成之后，若则说明整个线段在裁剪窗口之外，直接舍弃；反之保留区间内的线段。

#### 1.4.2 多边形裁剪：逐边裁剪算法

设多边形原先顶点表为vertices.

用4条边界线依次对多边形进行裁剪，

设本次裁剪得到的新顶点表为vtxs，

对vertices表中各顶点i进行遍历，

若点vertices[i]在边界线外侧，

若点vertices [(i+1)%vertexes.size()]在边界线外侧，vtxs不加入任何点；

若点vertices [(i+1)%vertexes.size()]在边界线内侧，vtxs加入i与i+1形成的边与边界线的交点，和i+1号顶点；

若点vertices[i]在边界线内侧，

若点vertices [(i+1)%vertexes.size()]在边界线外侧，vtxs只加入i与i+1形成的边与边界线的交点；

若点vertices [(i+1)%vertexes.size()]在边界线内侧，vtxs只加入i+1号顶点

顶点遍历完成之后将新顶点表vtxs赋给vertices.

所有边界裁剪完成之后的顶点表即为新多边形的顶点表，根据此顶点表生成新多边形即可。

## 2 系统框架设计

### 2.1 开发环境

|  |  |
| --- | --- |
| 操作系统 | Windows 10家庭中文版 |
| 编程语言 | C++ 11 |
| 外部库 | Qt 5.4，OpenGL 2.7 |
| 编译器 | MinGW 4.9.1 32bit |
| IDE | Qt Creator 3.3.0 |

### 2.2 系统架构概览

用Qt向用户提供交互，用OpenGL提供底层绘制，考虑到Qt与OpenGL的兼容性，使用了继承自QWidget的子类QGLWidget实现Qt环境下的OpenGL绘制。

QMainWindow负责与用户交互，并以标签页的形式集成了QGLWidget；

QGLWidget负责绘制本标签页的内容，并集成了各类FigureControl，用于将本标签页的交互信号送给不同的图形控制类；

FigureControl负责该类型图形的交互，有LineControl、CircleControl、EllipseControl、PolygonControl等，用于分别提供不同图形的不同交互方式；

Figure负责保存该图形的数据信息，包括顶点、圆心、半径以及轮廓点、填充点等，集成了Point类，图形绘制通过调用Point的draw函数完成；

Point负责实现绘制单个普通点或标记点，并向上提供其他常用功能函数支持。

系统核心架构如下图

QMainWindow

GLWidget

FigureControl

Figure

Point

界面交互：管理多个QGLWidget的标签页，接收键盘鼠标信息

标签页绘制：管理FigureControl的子类，实现本标签页的绘制

单图形管理：管理该图形的Figure类，向上提供该种图形的绘制接口

图形数据：管理本图形数据，提供图形绘制、修改、获取数据的接口

点信息：作为Figure的基础，提供绘制、比较、获取信息等接口

OpenGL

提供轮廓点、标记点、填充点绘制功能

其中GLWidget派生出一个GL3DWidget，作为3D标签页，用于3D图形的绘制；在FigureControl层有一个CutWindow类，用于绘制裁剪窗口。系统架构中的聚集关系如下图

QMainWindow

GLWidget

GL3DWidget

FigureControl

CutWindow

Figure

Point

Point类作为轮廓和填充的构成部分聚集在Figure类中；Figure类的数组放在FigureControl中，由FigureControl处理本Figure的相关交互；CutWindow用于管理裁剪窗口；GLWidget为2D画布的标签页，GL3DWidget继承自GLWidget，作为3D画布标签页；所有标签页聚集在QMainWindow中，由QMainWindow与用户交互并向下发送消息。

### 2.3 图形管理类：FigureControl

图形管理类之间的关系较为简单，主要用于存储本图形的数据，并进行与本图形相关的交互，如鼠标点击、键盘按下、图形绘制、设置聚焦、裁剪当前图形等，类继承关系如下图

FigureControl

LineControl

CircleControl

EllipseControl

CurveControl

PolygonControl

### 2.4 图形类：Figure

Figure为所有图形的虚基类，继承关系如下图

Figure

SimpleFigure

Polygon

Line

Circle

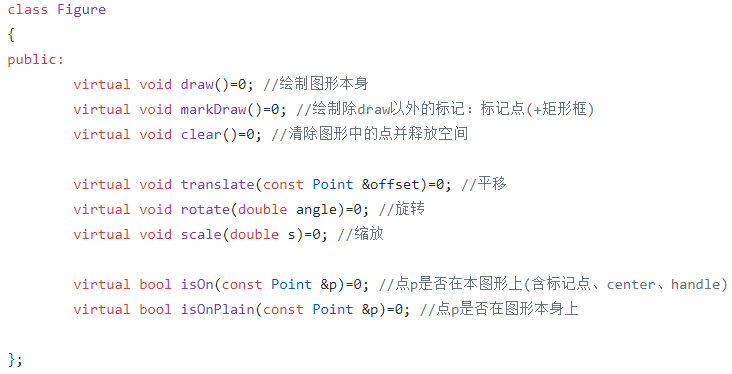
Ellipse

Area

Curve

Figure类提供所有图形都应有的共同接口：

* 基础绘制：绘制图形、绘制图形标记点、清空图形
* 基础变换：平移、旋转、缩放
* 判断某点是否在该图形上



SimpleFigure是对所有简单图形的抽象，SimpleFigure的轮廓均直接由点构成。而Polygon的边为Line，所以不属于SimpleFigure。SimpleFigure中提供了通用的draw()、clear()、isOn()。

Area是对所有可填充图形的抽象，用于填充点管理，提供了与填充相关的函数。

Line、Circle、Ellipse、Curve、Polygon分别为直线、圆、椭圆、曲线、多边形。其中由于Ellipse和Polygon与OpenGL库中命名冲突，所以在实现中命名为MyEllipse和MyPolygon，但相应代码文件名仍为Ellipse和Polygon.

## 3 软件测试

### 3.1 直线测试

#### 3.1.1 绘制

点击绘制不同斜率直线以验证算法正确性，在各种斜率情况下，直线绘制都有很好的效果，如下图

|  |  |  |  |
| --- | --- | --- | --- |
| k=0 | 0<k<1 | k=1 | k>1 |
| k不存在 | -1<k<0 | k=-1 | k<-1 |

#### 3.1.2 平移

拖动中心标记点改变直线位置，直线形状完全保持并跟随箭头自由移动，如下图

|  |  |  |
| --- | --- | --- |
| 初始状态  C:\Users\帅\AppData\Local\Temp\ScreenToGif\Recording\2017-12-26 19-49-01\Clipboard\3\22.png | 右移一段距离  C:\Users\帅\AppData\Local\Temp\ScreenToGif\Recording\2017-12-26 19-49-01\Clipboard\4\58.png | 下移一段距离  C:\Users\帅\AppData\Local\Temp\ScreenToGif\Recording\2017-12-26 19-49-01\Clipboard\6\84.png |

#### 3.1.3 旋转

拖动旋转标记点让直线旋转，如下图

|  |  |  |
| --- | --- | --- |
| 初始状态 | 顺时针旋转一个锐角 | 顺时针旋转一个钝角 |

#### 3.1.4 缩放

点击放大/缩小改变直线大小，如下图

|  |  |  |
| --- | --- | --- |
| 初始状态 | 放大两次 | 缩小两次 |

#### 3.1.5 裁剪

绘制不同的裁剪窗口对线段进行裁剪

|  |  |
| --- | --- |
| 初始状态 | 全部在裁剪窗口内，直线没有任何变化 |
| 初始状态 | 部分在裁剪窗口内，在窗口外的直线部分被裁掉 |
| 初始状态 | 全部在裁剪窗口外，直线被删掉 |

### 3.2 圆测试

#### 3.2.1 绘制

点击后拖动绘出一个圆

|  |  |
| --- | --- |
| 半径较小 | 半径较大 |

#### 3.2.2 填充

点击填充图标进行填充

|  |  |
| --- | --- |
| 填充前 | 填充后 |

#### 3.2.3 平移

拖动中心标记点移动圆

|  |  |
| --- | --- |
| 初始位置 | 移动后 |

#### 3.2.4 旋转

拖动旋转标记点旋转圆

|  |  |
| --- | --- |
| 初始状态 | 旋转一定角度后 |

#### 3.2.5 缩放

点击放大/缩小改变圆的大小

|  |  |  |
| --- | --- | --- |
| 初始状态 | 放大 | 缩小 |

### 3.3 椭圆测试

#### 3.3.1 绘制

点击并拖动绘制出一个椭圆

|  |  |
| --- | --- |
| 横轴>纵轴 | 横轴<纵轴 |

#### 3.3.2 平移

|  |  |
| --- | --- |
| 初始状态 | 平移 |

#### 3.3.3 旋转

拖动旋转标记点至水平或垂直位置

|  |  |
| --- | --- |
| 初始状态 | 旋转90度 |

#### 3.3.4 缩放

|  |  |
| --- | --- |
| 初始状态 | 旋转90度 |

### 3.4 多边形测试

#### 3.4.1 绘制

点击绘制多个点，以第一个点为终点得到一个多边形

|  |  |  |
| --- | --- | --- |
| 绘制折线 | 单击确定顶点 | 单击起始点完成绘制 |

#### 3.4.2 填充

|  |  |
| --- | --- |
| 初始状态 | 填充后 |

#### 3.4.3 平移

|  |  |
| --- | --- |
| 初始状态 | 平移后 |

#### 3.4.4 旋转

|  |  |
| --- | --- |
| 初始状态 | 旋转后 |

#### 3.4.5 缩放

|  |  |  |
| --- | --- | --- |
| 初始状态 | 放大 | 缩小 |

#### 3.4.6 裁剪

|  |  |
| --- | --- |
| 初始状态 | 全部在裁剪窗口内，没有任何变化 |
| 初始状态 | 部分在裁剪窗口内，在窗口外的部分被裁掉 |
| 初始状态 | 全部在裁剪窗口外，多边形被删掉 |

### 3.5 曲线测试

#### 3.5.1 绘制

点击4个控制点自动生成贝塞尔曲线

|  |  |  |
| --- | --- | --- |
| 绘制折线 | 单击确定顶点 | 单击起始点完成绘制 |

#### 3.5.2 平移

|  |  |
| --- | --- |
| 初始状态 | 平移后 |

#### 3.5.3 旋转

|  |  |
| --- | --- |
| 初始状态 | 旋转后 |

#### 3.5.4 缩放

|  |  |  |
| --- | --- | --- |
| 初始状态 | 放大 | 缩小 |

## 4 结束语

本技术报告详尽地介绍了绘图系统所使用的所有算法，以及系统整体架构。报告中通过图表解释了消息如何从用户的鼠标键盘操作一层层传递到底层OpenGL，然后绘制出丰富多彩的图形。另外，报告中也对图形绘制算法进行了充分的测试，以确保图形绘制能够方便快捷地完成。

**参考文献**

[1] 孙正兴,周良,郑洪源,谢强.计算机图形学教程.北京:机械工业出版社,2006.

[2] Edward Angel,段菲.OpenGL编程基础(第3版).北京:清华大学出版社,2008.

[3] 陈家骏,郑滔.程序设计教程用C++语言编程(第3版).北京:机械工业出版社,2015.