《计算机图形学》系统技术报告

郑来栋

(南京大学 计算机科学与技术系, 南京 210093)

摘 要: 摘要

关键词: 关键词

# 引言

引言

# 算法描述

## 图形绘制

### 直线绘制-Bresenham画线算法

### 圆绘制-中点圆算法

### 椭圆绘制-中点椭圆算法

### 矩形和多边形

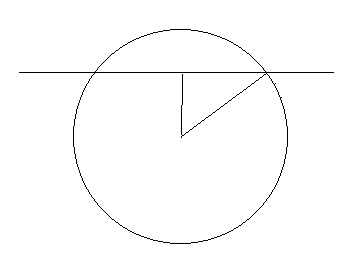
### Bezier曲线

## 图形填充

### 圆的填充

采用扫描填充，直接处理，对每条平行x轴的扫描线，计算其与圆两侧的交点，填充交点之间的线段即可。

扫描线y与圆两侧的交点可用勾股定理求出



center: P(cx, cy)

radius: r

for y from cy to cy + r – 1:

dy = y – cy

dx = sqrt()

left = cx – dx

right = cx + dx

for x from left + 1 to right – 1:

fillPoint(x, y)

fillPoint(x, 2 \* cy - y)

### 矩形填充

矩形填充比较简单，直接对四条边内部区域进行填充即可

for x from xmin to xmax:

for y from ymin to ymax:

fillPoint(x, y)

### 一般多边形的填充(扫描填充算法)

通过构建有序边表和活化边表，求出各条扫描线上的填充范围，实现了多边形的扫描填充算法。

边表的数据结构如下：

struct Edge

{

double x;

double dx;

int y\_upper;

}

## 几何变换

实现基本的几何变换，包括平移、旋转和缩放，只需实现对点的变换，其余几何图形可根据几何特征基于点进行变换。

设点P(x, y) ,

### 平移

设平移向量为t = (), 则P平移后的坐标为

基于点的平移，实现基本几何形状的平移

|  |  |
| --- | --- |
| 几何图形 | 平移方式 |
| 直线 | 分别平移线段端点 |
| 圆 | 平移圆心 |
| 椭圆 | 平移椭圆外接矩形左上角和右下角端点 |
| 矩形 | 平移左上角和右下角 |
| 多边形 | 分别平移每个顶点 |
| 曲线 | 分别平移每个控制点 |

### 旋转

设旋转基准点为C = (), 旋转角为α，则P旋转后的坐标可表示为

基于点的旋转，实现基本几何形状的旋转

|  |  |  |
| --- | --- | --- |
| 几何图形 | 旋转中心 | 旋转方式 |
| 直线 | 线段中点 | 分别旋转线段端点 |
| 圆 | -- | do nothing |
| 椭圆 | 椭圆中心 | 旋转椭圆外接矩形的顶点 |
| 矩形 | 矩形中心 | 旋转左上角和右下角的顶点 |
| 多边形 | 所有顶点的中心(取平均) | 分别旋转每个顶点 |
| 曲线 | 所有控制点的中心(取平均) | 分别旋转每个控制点 |

### 缩放

设缩放基准点为C = (), 缩放系数大小为s，则P旋转后的坐标可表示为

基于控制点的缩放也可以实现基本几何形状的缩放，其基准点的选取和旋转一样，都是选择“中心”

## 裁剪

设裁剪窗口是一个标准位置的矩形

W = {(x, y) | xmin ≤ x ≤ xmax, ymin ≤ y ≤ ymax}, 且舍弃裁剪窗口外的图形部分。

### 线段裁剪（梁友栋-Barsky算法）

设线段P = P1P2 ，端点分别是P1(x1, y1), P2(x2, y2)，其参数方程可以写成如下形式：

其中

如果点P(x, y)位于W内，则应满足

,

将该公式变形做统一之后记为

可得

我们可将这四种情况对应到裁剪窗口的四条边，对于pk和qk, k = 1, 2, 3, 4分别对应W的左、右、下、上边界, 从上式可以有如下观察

1. 线段平行于裁剪边界k(实际上是两条)

* 若, 线段在裁剪窗口外
* 若, 线段平行于裁剪边界且在窗口内
* 线段从外到内穿过裁剪边界第k侧延长线
* 线段从内到外穿过裁剪边界第k侧延长线

计算pk, qk

u0 = 0, u1 = 1

for i from 1 to 4:

if pi == 0:

if qi < 0:

return //在裁剪窗口外，直接舍弃

else

r = qi / pi //r表示直接与第i边的交点的参数值

if pi < 0

u1 = max(u1, r)

else

u2 = min(u2, r)

if u1 > u2:

return

update:

//保留参数在[u1, u2]内的部分

P1 = (x1 + u1\* Δx, y1 + u1\*Δy)

P2 = (x1 + u2\* Δx, y1 + u2\*Δy)

算法 梁友栋-Barsky

### 矩形裁剪

矩形裁剪较简单，我们只需要求出待裁剪矩形与裁剪窗口W的重叠部分即可

设待裁剪矩形左上角坐标为P1(x1, y1), 右下角坐标为P2(x2, y2)

先假设两个矩形相交，相交后的矩形为C, 设C的左上角坐标为(xc1, yc1), 右下角坐标为(xc2, yc2), 则其顶点坐标可通过如下式子计算出来

最后判断不等式 是否满足即可判断出是否有交集，如果相交，则重设带裁剪矩形左上角和右下角的坐标。

### 一般多边形裁剪(Sutherland-Hodgeman算法)

依次用裁剪窗口的四条边进行裁剪，每一次裁剪过程中，依次检查多边形的每条边，根据边与裁剪窗口对应当前用于裁剪的边界的位置关系，决定将哪些点加入到新顶点集中。

设当前用裁剪窗口的边Ei (0≤i≤3分别对应上、右、下和左边界), 当前检查边E = P1P2, P1为起点，P2为终点(因为检查过程是有序的，所以边必须有方向性)。设Ei和E的交点为V(若有)

1. P1在Ei外侧空间，P2在Ei内侧空间

依次添加V和P2

1. P1和P2都在内侧空间

添加P2

1. P1在E内侧空间，P2在Ei外侧空间

添加V

1. P1和P2都在外侧空间

不添加

输入: vertex //多边形顶点

W(xmin, ymin, xmax, ymax) // 裁剪窗口

initialise cut\_edges // cut\_edges是裁剪窗口的上，右，下，左四条边

for i in 0 to 3 do:

vertex\_temp = {}

Ei = cut\_edges[]

for j in 0 to vertex.size – 2 do:

start = vertex[j], end = vertex[j+1]

E = Line(start, end)

if outside(start, Ei)

if !outside(end, Ei)

V = intersect(E, Ei) //求交点坐标

vertex\_temp.add(V)

vertex\_temp.add(end)

else

if outside(end, Ei)

V = intersect(E, Ei) //求交点坐标

vertex\_temp.add(V)

else

vertex\_temp.add(end)

vertex = vertex\_temp //作为下一轮的输入

vertex\_temp = {}

# 系统框架

## 用户界面

用户界面主要包括两大部分，一是顶部的TooBar，用于实现图形输入，编辑等菜单选项，改变系统状态。二是中间的绘图区，继承自QOpenGLWidget，用于处理绘制图形，处理鼠标事件等。

## 模块划分及类层次结构

### 图形类

IShape

Poly

Curve

Oval

Rect

Circle

Line

CutBox

1. 所有的图形类都继承自IShape，IShape实现了一些基本方法，且定义了一套接口，如draw()，translate()，rotate()等。
2. 利用C++的多态性，绘制系统不需要知道图形的种类然后分情况绘制。
3. CutBox类继承自Rect类，用于裁剪窗口的绘制
4. Curve类实现的是Brazier曲线

### 绘制类

**FigureGenerator**

**PolygonGenerator**

**LineGenerator**

**CurveGenerator**

**OvalGenerator**

**RectangleGenerator**

**CircleGenerator**

1. 绘制类均继承自FigureGenerator，用于实现图形的绘制功能
2. 图形输入功能来自于鼠标事件，依据当前状态和鼠标事件进行不同的动作, 下表是对不同图形，不同的鼠标事件应当采取的动作

|  |  |  |  |
| --- | --- | --- | --- |
|  | MousePress | MouseMove | MouseRealse |
| 直线 | 以点击点为起点和终点创建一条直线 | 更新终点 |  |
| 圆 | 以点击点为圆心，半径为1创建一个圆 | 重新计算并设置半径 |  |
| 椭圆 | 创建一个椭圆，以点击点的坐标初始化其外接矩形 | 重新设置外接矩形的一个顶点 | 绘制完成 |
| 矩形 | 以点击点为起点和终点创建一个矩形 | 重新设置一个顶点 |  |
| 多边形 | 以点击坐标添加一个顶点，如果多边形已封闭则结束绘制 | 更新最后一个顶点坐标 | 更新最后一个顶点坐标(跟踪鼠标) |
| 曲线 | 添加一个控制点 | 不做操作 | 不做操作 |

其中，

1. 对于多边形的绘制，由于不确定用户要输入多少顶点，因此当用户点击时就需要添加顶点，当多边形封闭时标志绘制结束(最后一个点和第一个点重合)
2. 对于曲线的绘制，同样需要点击时添加控制点，当用户再次点击ToolBar的曲线绘制按钮时，一条曲线绘制结束。
3. 如果这些功能在一个类里实现，将有很大的复杂度，难以控制，所以将每个图形的绘制抽象出一个类来实现，它们都能处理鼠标事件，这样设计每一个类结构简单容易控制。

### 图形编辑类

编辑类主要用于处理图形选中情况下的图形编辑，如修改顶点位置，平移，拖动矩形边长等等。

**Editor**

**PolyEditor**

**LineEditor**

**CurveEditor**

**OvalEditor**

**RectEditor**

**CircleEditor**

References:

1. …

附中文参考文献:

[9] 郭亮,唐稚松.三机冗余容错系统的描述和验证.软件学报,2003,14(1):28-35. http://www.jos.org.cn/1000-9825/14/28.htm

[11] 王兴伟.分布式多媒体系统服务质量管理与组通信机制的研究[博士学位论文].沈阳:东北大学,1998.