软件设计文档

课程：图形学

任课老师：孙正兴

姓名：金鑫

学号：121220307

CONTENT

1.1 编写目的

对软件进行模块级别的详细设计说明，以便编写代码人员参考，也方便维护人员在软件维护过程中的维护工作。

本文档的阅读对象为软件的开发和使用人员。

1.2 背景

项目名称：Drawing Board

项目的提出：孙正兴老师

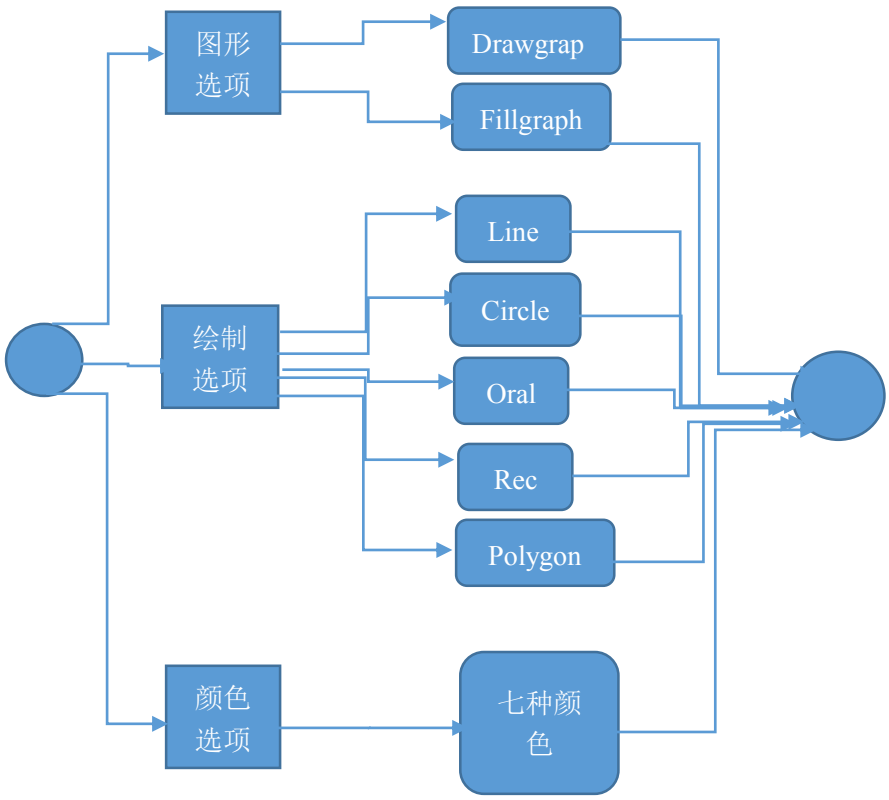
项目的开发者：金鑫

1.3 定义

A. 系统：Drawing Board

B. 用户：使用该系统的用户

2. 程序系统的结构



3. 算法描述

3.1Bresenham 算法

|m|＜1的Bresenham画线算法

(1). 输入线的两个端点，并将左端点存贮在(x_0,y_0)中；

(2). 将(x_0,y_0)装入帧缓冲器，画第一个点；

(3). 计算常量：Δx、Δy、$2\Delta y$和$2\Delta y-2\Delta x$，

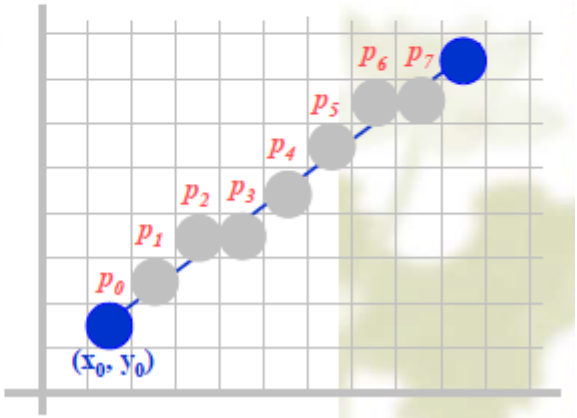
起始位置(x_0,y_0)的决策参数p_0计算为：$p_0=2\Delta y-\Delta x$

(4). 从$k=0$开始，在每个离散取样点x_k处，进行下列检测：

- 若$p_k<0$，画点(x_{k+1},y_k)，且：$p_{k+1}=p_k+2\Delta y$；
- 若$p_k>0$，画点(x_{k+1},y_{k+1})，且：$p_{k+1}=p_k+2\Delta y-2\Delta x$。

(5). $k=k+1$；

(6). 重复步骤4，共Δx次。



3.2 中点圆算法

1. 输入圆半径r和圆心(x_c,y_c)。圆心在原点的圆周上的第一点：

   $(x_0,y_0)=(0,r)$

2. 计算圆周点$(0,r)$的初始决策参数值为：$p_0=5/4-r$；

3. 从$k=0$开始每个取样位置x_k处完成下列检测：

   - 若$p_k<0$，选择像素位置：(x_{k+1},y_k)；

     且：$p_{k+1}=p_k+2x_{k+1}+1$；(决策参数增量计算)

   - 若$p_k<0$，选择像素位置：(x_{k+1},y_k-1)；

     且：$p_{k+1}=p_k+2x_{k+1}+1-2y_{k+1}$。 (决策参数增量计算)

   其中：$2x_{k+1}=2x_k+2$，且 $2y_{k+1}=2y_k-2$。

4. 确定其它七个八分圆中的对称点。

5. 计算出的像素位置(x,y)移动到中心在(x_c,y_c)的圆路径上，

   - 即：对像素位置进行平移：$x=x+x_c$，$y=y+y_c$；

6. 重复步骤3到5，直止$x\geqslant y$。

3.3 中点椭圆算法

1. 输入r_x、r_y和(x_c,y_c)，得到中心在原点的椭圆的第一个点：$(x_0,y_0)=(0,r_y)$；

2. 区域1决策参数初值：$p1_0=r_y^2-r_x^2r_y+r_x^2/4$

3. 区域1每个x_k位置处，$k=0$开始循环测试：

   - $p1_k<0$，选择像素：(x_{k+1},y_k)，

   且：$p1_{k+1}=p1_k+2r_y^2x_{k+1}+r_y^2$。

   - $p1_k>0$，选择像素：(x_{k+1},y_k-1)，

   且：$p1_{k+1}=p1_k+2r_y^2x_{k+1}-2r_x^2y_{k+1}+r_y^2$。

   其中：$2r_y^2x_{k+1}=2r_y^2x_k+2r_y^2$；

   $2r_x^2y_{k+1}=2r_x^2y_k-2r_x^2$，

   - 循环至：$2r_y^2x\geqslant 2r_x^2y$

4. 区域1最后点(x_1,y_1)计算区域2参数初值：

   $p2_0=r_y^2(x_1+1/2)^2+r_x^2(y_1-1)-r_x^2r_y^2$

5. 区域2每个y_k位置处，$k=0$开始循环检测：

   - $p2_k>0$，选择像素：(x_k,y_k-1)，

   且：$p2_{k+1}=p2_k-2r_x^2y_{k+1}+r_x^2$，

   - 否则，选择像素：(x_{k+1},y_k-1)，

   且：$p2_{k+1}=p2_k+2r_y^2x_{k+1}-2r_x^2y_{k+1}+r_x^2$，

   - 与区域1相同的x和y增量计算。

   - 循环至$(r_x,0)$

6. 对称：确定其它三个像限对称点。

7. 平移：将每个像素位置(x,y)平移到中心在(x_c,y_c)的椭圆轨迹上，并按坐标值画点：$x=x+x_c$，$y=y+y_c$

4. 程序实现

4.1Bresenham 算法程序实现

```
public void drawline(Graphics g, int x1, int y1, int x2, int y2){
                    int dx = x2 - x1;
                    int dy = y2 - y1;
                    int ux;
                    if (dx > 0) ux = 1; else ux = -1;

                    int uy;
                    if (dy > 0) uy = 1; else uy = -1;
                    int x = x1, y = y1, eps;//eps为累加误差

                    eps = 0;
                    if (dx < 0) dx = -dx;
                    if (dy < 0) dy = -dy;
                    if (dx > dy)
                    {
                        for (x = x1; x != x2+ux; x += ux)
                        {
                             //SetPixel(img, x, y);
                             g.drawLine(x, y, x, y);
                             eps += dy;
                             if ((eps << 1) >= dx)
                             {
                                  y += uy; eps -= dx;
                             }
                        }
                    }
                    else
                    {
                        for (y = y1; y != y2+uy; y += uy)
                        {
                             g.drawLine(x, y, x, y);
                             eps += dx;
                             if ((eps << 1) >= dy)
                             {
                                  x += ux; eps -= dy;
                             }
                        }
                    }
```

```
            }
```

这里用 drawLine(x, y, x, y)来代替填充一个点。

4.2 中点圆算法程序实现

```
public void drawcircle(Graphics g){
                    int xc = point[0][0], yc = point[0][1];
                    int r = (point[0][0] - point[1][0]) * (point[0][0] -
point[1][0]) + (point[0][1] - point[1][1]) * (point[0][1] - point[1][1]);
                    r = (int) Math.sqrt(r);
                    int xk = 0, yk = r;
                    double pk = 5 / 4 - r;
                    for (;xk <= yk; xk ++)
                    {
                          g.drawLine(xk + xc, yk + yc, xk + xc, yk + yc);
                          g.drawLine(yk + xc, xk + yc, yk + xc, xk + yc);
                          g.drawLine(yk + xc, -xk + yc, yk + xc, -xk + yc);
                          g.drawLine(xk + xc, -yk + yc, xk + xc, -yk + yc);
                          g.drawLine(-xk + xc, -yk + yc, -xk + xc, -yk +
yc);
                          g.drawLine(-yk + xc, -xk + yc, -yk + xc, -xk +
yc);
                          g.drawLine(-yk + xc,  xk + yc, -yk + xc, xk +
yc);
                          g.drawLine(-xk + xc, yk + yc, -xk + xc, yk + yc);
                          if (comboBox.getSelectedIndex() == 1)
                          {
                                g.drawLine(xk + xc, yk + yc, xk + xc, -yk
+ yc);
                                g.drawLine(yk + xc, xk + yc, yk + xc, -xk
+ yc);
                                g.drawLine(-yk + xc, xk + yc, -yk + xc, -
xk + yc);
                                g.drawLine(-xk + xc, yk + yc, -xk + xc, -
yk + yc);
                          }
                          if (pk < 0){
                                pk = pk + 2 * (xk + 1) + 1;
                          }
                          else{
                                pk = pk + 2 * (xk + 1) + 1 - 2 * (yk - 1);
                                yk --;
                          }
                    }
```

```
            }
```

4.3 中点椭圆算法程序实现

```
public void draworal(Graphics g){
                    int xc = point[0][0], yc = point[0][1];
                    int rx = (point[0][0] - point[1][0]) * (point[0][0] -
point[1][0]) + (point[0][1] - point[1][1]) * (point[0][1] - point[1][1]);
                    rx = (int) Math.sqrt(rx);
                    int ry = (point[0][0] - point[2][0]) * (point[0][0] -
point[2][0]) + (point[0][1] - point[2][1]) * (point[0][1] - point[2][1]);
                    ry = (int) Math.sqrt(ry);
                    int xk = 0, yk = ry;
                    System.out.println("Hello World!");
                    System.out.println(count);
                    double p1k = ry * ry - rx * rx * ry + rx * rx / 4;
                    for (;ry * ry * xk <= rx * rx * yk; xk ++){
                          //System.out.println(xk);
                          g.drawLine(xk + xc, yk + yc, xk + xc, yk + yc);
                          g.drawLine(xk + xc, -yk + yc, xk + xc, -yk + yc);
                          g.drawLine(-xk + xc, -yk + yc, -xk + xc, -yk +
yc);
                          g.drawLine(-xk + xc, yk + yc, -xk + xc, yk + yc);

                          if (comboBox.getSelectedIndex() == 1)
                          {
                                g.drawLine(xk + xc, yk + yc, xk + xc, -yk
+ yc);
                                g.drawLine(-xk + xc, yk + yc, -xk + xc, -
yk + yc);
                          }
                          if (p1k < 0){
                                p1k = p1k + 2 * ry * ry * (xk + 1) + ry *
ry;
                          }
                          else{
                                p1k = p1k + 2 * ry * ry * (xk + 1) - 2 *
rx * rx * (yk - 1)+ ry * ry;
                                yk --;
                          }
                    }
                    double p2k = ry * ry * (xk + 1 / 2) * (xk + 1 / 2) + rx
* rx * (yk - 1) * (yk - 1) - rx * rx * ry * ry;
```

```
                    //double p2k = ry * (xk + 1 / 2) * 2 + rx * (yk - 1) *
2 - rx * ry * 2 + 1 / 2;
                    for (;xk < rx || yk >= 0; ) {
                          g.drawLine(xk + xc, yk + yc, xk + xc, yk + yc);
                          g.drawLine(xk + xc, -yk + yc, xk + xc, -yk + yc);
                          g.drawLine(-xk + xc, -yk + yc, -xk + xc, -yk +
yc);
                          g.drawLine(-xk + xc, yk + yc, -xk + xc, yk + yc);

                          if (comboBox.getSelectedIndex() == 1)
                          {
                                g.drawLine(xk + xc, yk + yc, xk + xc, -yk
+ yc);
                                g.drawLine(-xk + xc, yk + yc, -xk + xc, -
yk + yc);
                          }
                          if (p2k > 0) {
                                p2k = p2k - 2 * rx * rx * (yk - 1) + rx *
rx;
                                yk --;
                          }
                          else {
                                p2k = p2k + 2 * ry * ry * (xk + 1) - 2 *
rx * rx * (yk - 1) + rx * rx;
                                xk ++; yk --;
                          }
                    }
              }
```

5. 操作介绍

5.1 线画图元：线

Step1:选择 Drawgraph
Step2：选择 Line
Step3：选择颜色（8 种）
Step4：在屏幕上用鼠标点两个点，一个作为线段左端点，一个作为线段右端点
Step5： 点击 Finish 按钮，显示图形

5.2 线画图元：圆

Step1:选择 Drawgraph
Step2：选择 Circle
Step3：选择颜色（8 种）
Step4：在屏幕上用鼠标点两个点，一个作为圆心，另一个作为圆上任意一点
Step5： 点击 Finish 按钮，显示图形

5.3 线画图元：椭圆

Step1:选择 Drawgraph
Step2：选择 Oral
Step3：选择颜色（8 种）
Step4：在屏幕上用鼠标点三个点，一个作为中心，另外两个点到中心的距离作为长短力矩
Step5： 点击 Finish 按钮，显示图形

5.4 线画图元：矩形

Step1:选择 Drawgraph
Step2：选择 Rec
Step3：选择颜色（8 种）
Step4：在屏幕用鼠标上两个点，作为矩形对角两个顶点
Step5： 点击 Finish 按钮，显示图形

5.5 线画图元：多边形

Step1:选择 Drawgraph

Step2：选择 Polygon
Step3：选择颜色（8 种）
Step4：在屏幕上用鼠标点若干点，作为多边形顶点
Step5： 点击 Finish 按钮，显示图形

5.6 填充图元：圆

Step1:选择 Fillgraph
Step2：选择 Circle
Step3：选择颜色（8 种）
Step4：在屏幕上用鼠标点两个点，一个作为圆心，另一个作为圆上任意一点
Step5： 点击 Finish 按钮，显示图形

5.7 填充图元：椭圆

Step1:选择 Fillgraph
Step2：选择 Oral
Step3：选择颜色（8 种）
Step4：在屏幕上用鼠标点三个点，一个作为中心，另外两个点到中心的距离作为长短力矩
Step5： 点击 Finish 按钮，显示图形

5.8 填充图元：矩形

Step1:选择 Fillgraph
Step2：选择 Rec
Step3：选择颜色（8 种）
Step4：在屏幕用鼠标上两个点，作为矩形对角两个顶点
Step5： 点击 Finish 按钮，显示图形

6. 类介绍

- 类表

所属包	名称	标识符	数据项	操作	层次关系
Graphics	主界面	Home	ContentPane Panel comboBox comboBox_1 comboBox_2	drawimage	继承