《计算机图形学》10月报告

171850505，宋昱豪，yhsong.nju@gmail.com

2019 年 10 月 31 日

1 综述

基于计算机图形学设计了一套画图系统。完成了画图软件。主要实现了基于各种算法的图元生成，图元编辑，文件的保存打开，基于文档文件的指令序列读取完成画图任务，基于鼠标交互的图元绘制。并在此基础上对于软件界面进行了美化，具有一定可用性。

关键词： openGL　　Qt gui设计　　计算机图形学

2 算法介绍

2.1 直线绘制算法

2.1.1 DDA算法

数字差分分析法（Digital Differential Analyzer, DDA）是利用计算两个坐标方向的差分来确定线段显示的屏幕像素位置的线段扫描转换算法。DDA算法的核心是增量式的思想。即利用光栅的特性消除了直线方程中的乘法，在x和y方向使用合适的增量来逐步沿直线的路径推导出各像素的位置。

现假设直线的方程为

$$y = m \times x + b \tag{1}$$

假设斜率$m \leq 1$，在x的维度上单位间隔采样并求每个采样点对应的y值，可以发现在对于$x_{k+1} = x_k + 1$的x采样时，$y_{k+1} = y_k + m$。因此在屏幕光栅的特点下，我们采用增量式的方法计算出直线路径上各像素点的位置。而当斜率m>1时，我们在y的维度上进行采样，对于$y_{k+1} = y_k + 1$时，$x_{k+1} = x_k + \frac{1}{m}$。在起始点和终点位置不同时增量的符号可能会改变。

```
1  if(aflag==DDA)
2     {
3         double delta_x, delta_y, x, y;
4         int dx, dy, dist;
5         dx = x2 - x1;
6         dy = y2 - y1;
```

```
        dist=(abs(dx)>abs(dy))?abs(dx):abs(dy);
        delta_x = (double)dx / (double)dist;
        delta_y = (double)dy / (double)dist;
        x = x1;
        y = y1;
        for (int i = 1; i <= dist; i++)
        {
            drawPoint(x,y);
            x += delta_x;
            y += delta_y;
         }
    }
```

2.1.2 Bresenham算法

在DDA算法中，由于斜率为浮点数，而浮点增量的连续迭加中取整误差的积累会使得计算得到的像素位置与实际上的像素位置偏离。同时浮点计算也十分耗时。而Bresenham算法通过引入整形参量定义来衡量两候选像素与直线路径上实际点在方向上的偏移。并通过对该参量的符号判定来决定像素点的选择。

假设依然以x的维度为基础，设目前已经绘制到第k步在点（x_k,y_k），下一个点的位置在（x_{k+1},y_k）或者（x_{k+1},y_{k+1}）。由于点x_{k+1}处y的值可以表示为

$$y = m \times x_{k+1} + b = m \times (x_k + 1) + b \tag{2}$$

此时两个候选的像素和线段的数学路径的垂直偏移为

$$\begin{cases} d_1 = y - y_k = m \times (x_k + 1) + b - y_k \\ d_2 = y_{k+1} - y = y_k + 1 - m \times (x_k + 1) + b \end{cases} \tag{3}$$

这两个点距离理想点的距离的差分为

$$d_1 - d_2 = 2 \times (x_k + 1) - 2 \times y_k + 2 \times b - 1 \tag{4}$$

为了达到消除浮点数运算的目的，令$m = frac\Delta y\Delta x$，设变量

$$p_k = \Delta x(d_1 - d_2) = 2\Delta y x_k - 2\Delta x y_k + c \tag{5}$$

同时由于c为常数，所以在循环计算中被消除，因为Δx = 1>0，所以p_k的符号与$d_1 - d_2$的符号相同。当y_k处的像素点比y_{k+1}处的像素点更加接近理想线段时，$p_k < 0$，反之则大于零。所以我们可以根据p_k的符号来决定k+1步走到的位置。

```
else if(aflag==Bresenham)
    {
        int x, y, dx, dy, s1, s2, p;
```

```
4           x=x1; y=y1;
5           dx=abs(x2-x1);
6           dy=abs(y2-y1);
7           s1=(x2>x1)?1:-1;
8           s2=(y2>y1)?1:-1;
9           if(dy<dx)
10          {
11              p=2*dy-dx;
12              for(int i=1;i<=dx;i++)
13              {
14                  drawPoint(x,y);
15                  if(p>=0)
16                  {
17                      y=y+s2;
18                      p=p-2*dx;
19                  }
20                  x=x+s1;
21                  p=p+2*dy;
22              }
23          }
24          else
25          {
26              p=2*dx-dy;
27              for(int i=1;i<=dy;i++)
28              {
29                  drawPoint(x,y);
30                  if(p>=0)
31                  {
32                      x=x+s1;
33                      p=p-2*dy;
34                  }
35                  y=y+s2;
36                  p=p+2*dx;
37              }
38          }
39      }
```

2.2 圆形绘制算法

采用中点圆生成算法，为了简化运算，只计算出八分之一的圆上的点，其他的点由计

算单得到的八分之一圆旋转对称得到。在中点圆算法中，和画直线的Bresenham算法类似，判断的参数是由点到中心的距离计算得到的。计算的是两个候选点的中点到圆心的距离。

$$\begin{cases} p_{k+1} = f_{circle}(x_{k+2}, y_k - frac12) & p_k < 0 \\ p_{k+1} = f_{circle}(x_{k+2}, y_k - frac32) & p_k \geq 0 \end{cases} \tag{6}$$

计算推导可得

$$\begin{cases} p_{k+1} = p_k + 2x_k + 3 & p_k < 0 \\ p_{k+1} = p_k + 2x_{k+1} - 2y_k + 5 & p_k \geq 0 \end{cases} \tag{7}$$

在计算中设置p_0为$5 - 4 \times r$ 初始化第一个点为(0,r)开始计算

2.3 中点椭圆绘制算法

椭圆绘制算法类似于中点圆生成算法，但是中点椭圆绘制算法中只能计算出四分之一的图形，剩下的用对称可以得到。在这四分之一图形中我们又要分成两个部分来计算点的位置。我们以一个以原点为中心的椭圆为例，我们只需要计算其第一象限内的点的位置。区域一即为其切线斜率小于1的部分，区域二为大于1的部分。在区域一中，我们有参数

$$\begin{cases} p_{k+1} = p_k + 2r_y^2 x_k + 3r_y^2 & p_k < 0 \\ p_{k+1} = p_k + 2r_y^2 x_{k+1} - 2r_x^2 y_k + 3r_y^2 & p_k \geq 0 \end{cases} \tag{8}$$

在区域二中参数变化为

$$\begin{cases} p_{k+1} = p_k - 2r_x^2 y_k + 3r_x^2 & p_k < 0 \\ p_{k+1} = p_k + 2r_y^2 x_{k+1} - 2r_x^2 y_k + 3r_x^2 & p_k \geq 0 \end{cases} \tag{9}$$

2.4 矩形绘制算法

矩形绘制算法即根据获得到的两个点的坐标，分别作为矩形对角线的两个端点，生成其他端点并调用画线算法即可完成

3 系统介绍

3.1 系统架构

系统基于Qt gui设计完成了软件。主要分为两个模块，一个是主窗口模块，一个是图形库模块。

主窗口模块主要完成了gui的交互功能，图形库则主要完成了图元的绘制，包含了软件的各种操作。

3.2 实现功能

3.2.1 图元绘制

目前完成了直线DDA算法和Bresenham算法的绘制，圆形基于中点圆算法的绘制，椭圆基于中点椭圆算法的绘制，另外添加了矩形的基于直线绘制的绘制。

3.2.2 图元操作

目前完成了选择图元，颜色改变，像素点粗细的改变，保存画布，调整画布大小，读取text文本文件并实现其中已实现操作的指令读取绘制的功能。支持鼠标操作。

4 总结