重庆交通大学
实验报告

班　　级：计科 专业 14 级 1 班

学　　号：631406010109

姓　　名：郭文浩

学　　院：信息科学与工程学院

实验项目名称：计算机图形学实验报告

实验所属课程：计算机图形学

实验室(中心)：软件实验室

指 导 教 师 ：何友全

实验完成时间：2017 年 5 月 4 日

直线 DDA 算法

一、实验目的

1、对画线函数有个基本的了解

2、实现网格的绘制和网格像素的填充

3、熟练掌握 DDA 算法的实现过程

二、实验内容

1、实现 DDA 直线生成算法

2、直线段起始点坐标和终点坐标可由自己输入正常的坐标

3、采用 GUI 界面实现该算法

三、实验结果

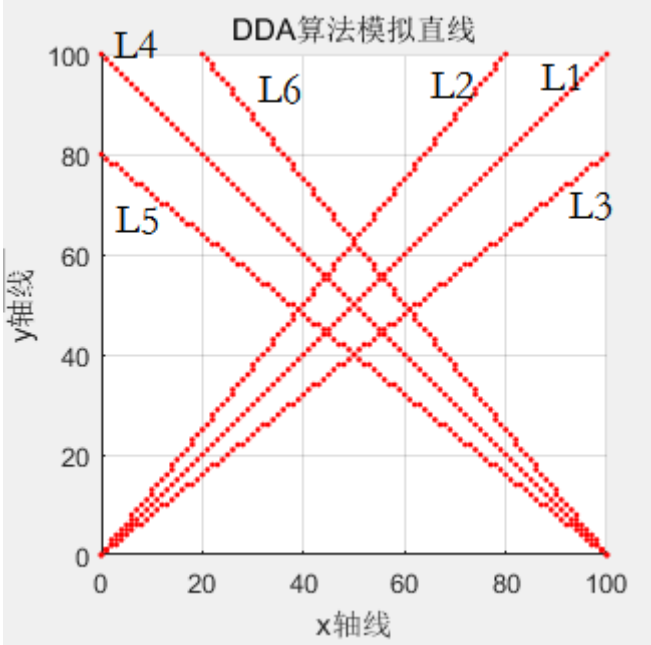
用 DDA 算法画了 6 条直线，基本覆盖了所有斜率，6 条直线坐标如下：

L1:(0,0),(100,100)　　L4:(0,100),(100,0)

L2:(0,0),(80,100)　　L5:(0,80),(100,0)

L3:(0,0),(100,80)　　L6:(20,100),(100,0)

程序运行如下：



四、实验分析和总结

1、实验原理

通过两点先求出斜率 k，然后可以通过斜率来判断是哪个方向增加的快。当|k|<=1 时，x 方向增加的快，x 每次+1，y 每次+k，由于显示器显示点时都是整数点，所以 y 要取整；当|k|>1 时，y 方向增加的快，y 每次+1，x 每次+k。循环执行，直到直线画完。

2、实验总结

这个算法经典的地方在于它增加的过程中，增加慢的那个方向的取整，正是因为这个取整的过程，导致了直线不直的现象。还有一点就是这个算法执行比较慢，若要拿到实际应用中去的话要做一定的优化。

五、源代码

```
function DDALine(x1,y1,x2,y2)
% DDALine(x1,y1,x2,y2)    DDA 算法绘直线
% x1                        第 1 个点的横坐标
% y1                        第 1 个点的纵坐标
% x2                        第 2 个点的横坐标
% y2                        第 2 个点的纵坐标
if(x1>x2)
    x=x1;   y=y1;
    x1=x2;  y1=y2;
    x2=x;   y2=y;
end
dx=x2-x1;
dy=y2-y1;
x=x1;
y=y1;
if(abs(dx)<abs(dy))
    max=dy;
else
```

```
    max=dx;
end
xIncrease=dx/abs(max);
yIncrease=dy/abs(max);
hold on;
for i=0:abs(max)
    plot(round(x),round(y),'.r');
    x=x+xIncrease;
    y=y+yIncrease;
end
hold off;
grid on;
xlabel('x 轴线');
ylabel('y 轴线');
title('DDA 算法模拟直线');
end
```

直线 Bresenham 算法

一、实验目的

1、对画线函数有个基本的了解

2、实现网格的绘制和网格像素的填充

3、熟练掌握直线 Bresenham 算法的实现过程

二、实验内容

1、实现 Bresenham 直线生成算法

2、直线段起始点坐标和终点坐标可由自己输入正常的坐标

3、采用 GUI 界面实现该算法

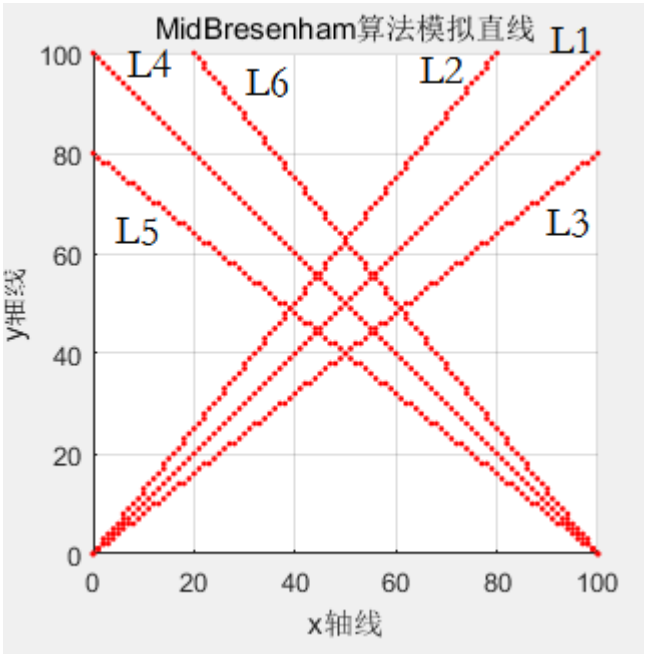
三、实验结果

用 MidBresenham 算法画了 6 条直线，基本覆盖了所有斜率，6 条直线坐标如下：L1:(0,0),(100,100)　　L4:(0,100),(100,0)

L2:(0,0),(80,100)　　L5:(0,80),(100,0)

L3:(0,0),(100,80)　　L6:(20,100),(100,0)

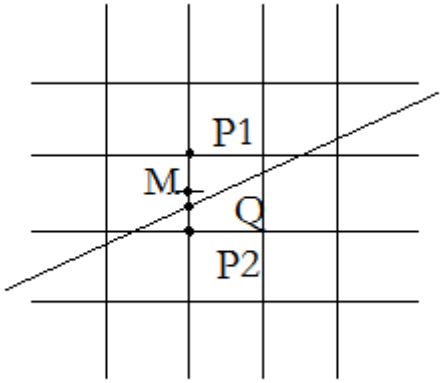
程序运行如下：



四、实验分析和总结

1、实验原理

如下图，引入点 P1、点 P2 的中点 M，利用中点 M 和交点 Q 判断，当 M 在 Q 上方时，说明交点靠近下半部分，直线的点应该描在 P2 的位置；当 M 在 Q 下方时，说明交点靠近上半部分，直线的点应该描在 P1 的位置。



引入判定式：$d_i = \mathrm{F}(x_M, y_M) = \mathrm{F}(x_i + 1, y_i + 0.5) = y_i + 0.5 - \mathrm{k}(x_i + 1 - \mathrm{b})$ 当$d_i < 0$时，M 在 Q 的下方，取 P1，$d_{i+1} = d_i + 1 - k$；当$d_i \geq 0$时，M 在 Q 的上方，取 P2，$d_{i+1} = d_i - k$。循环判断，直到直线画完。

2、实验总结

这个算法经典的地方也在于它增加的过程，但是这个算法比 DDA 算法稍微有优势一些，它每次循环只用判断 d，再生成新的 d，只做一次计算，而 DDA 算法每次都要对 x、y 增加计算，每次循环要计算两次，所以 MidBresenham 要稍微优化些，但仍有很大的改进空间。

五、源代码

```
function MidBresenhamLine(x1,y1,x2,y2)
% MidBresenhamLine(x1,y1,x2,y2)    Bresenham 中点算法绘直线
% x1                                第 1 个点的横坐标
% y1                                第 1 个点的纵坐标
% x2                                第 2 个点的横坐标
% y2                                第 2 个点的纵坐标
```

```
if(x1>x2)
    x=x1;   y=y1;
    x1=x2;  y1=y2;
    x2=x;   y2=y;
end
dx=x2-x1;
dy=y2-y1;
k=dy/dx;
if(k>0)
    if(k>1)
        d=dy-2*dx;
        upIncrease=2*dy-2*dx;
        downIncrease=-2*dx;
        x=x1;y=y1;
        hold on;
        while(y<=y2)
            plot(round(x),round(y),'.r');
            y=y+1;
            if(d<0)
                x=x+1;
                d=d+upIncrease;
            else
                d=d+downIncrease;
            end
        end
    else
        d=dx-2*dy;
        upIncrease=2*dx-2*dy;
        downIncrease=-2*dy;
```

```
        x=x1;y=y1;
        hold on;
        while(x<=x2)
            plot(round(x),round(y),'.r');
            x=x+1;
            if(d<0)
                y=y+1;
                d=d+upIncrease;
            else
                d=d+downIncrease;
            end
        end
    end
else
    if(k>-1)
        dy=-dy;
        d=dx-2*dy;
        upIncrease=2*dx-2*dy;
        downIncrease=-2*dy;
        x=x1;y=y1;
        hold on;
        while(x<=x2)
            plot(round(x),round(y),'.r');
            x=x+1;
            if(d<0)
                y=y-1;
                d=d+upIncrease;
            else
                d=d+downIncrease;
```

```
            end
        end
    else
        dy=-dy;
        d=dy-2*dx;
        upIncrease=2*dy-2*dx;
        downIncrease=-2*dx;
        x=x1;y=y1;
        hold on;
        while(y>=y2)
            plot(round(x),round(y),'.r');
            y=y-1;
            if(d<0)
                x=x+1;
                d=d+upIncrease;
            else
                d=d+downIncrease;
            end
        end
    end
end
grid on;
xlabel('x 轴线');
ylabel('y 轴线');
title('MidBresenham 算法模拟直线');
end
```

八分法画圆

一、实验目的

1、对画圆函数有个基本的了解

2、实现网格的绘制和网格像素的填充

3、熟练掌握圆的中点八分法画圆的实现过程

二、实验内容

1、实现八分法画圆生成算法

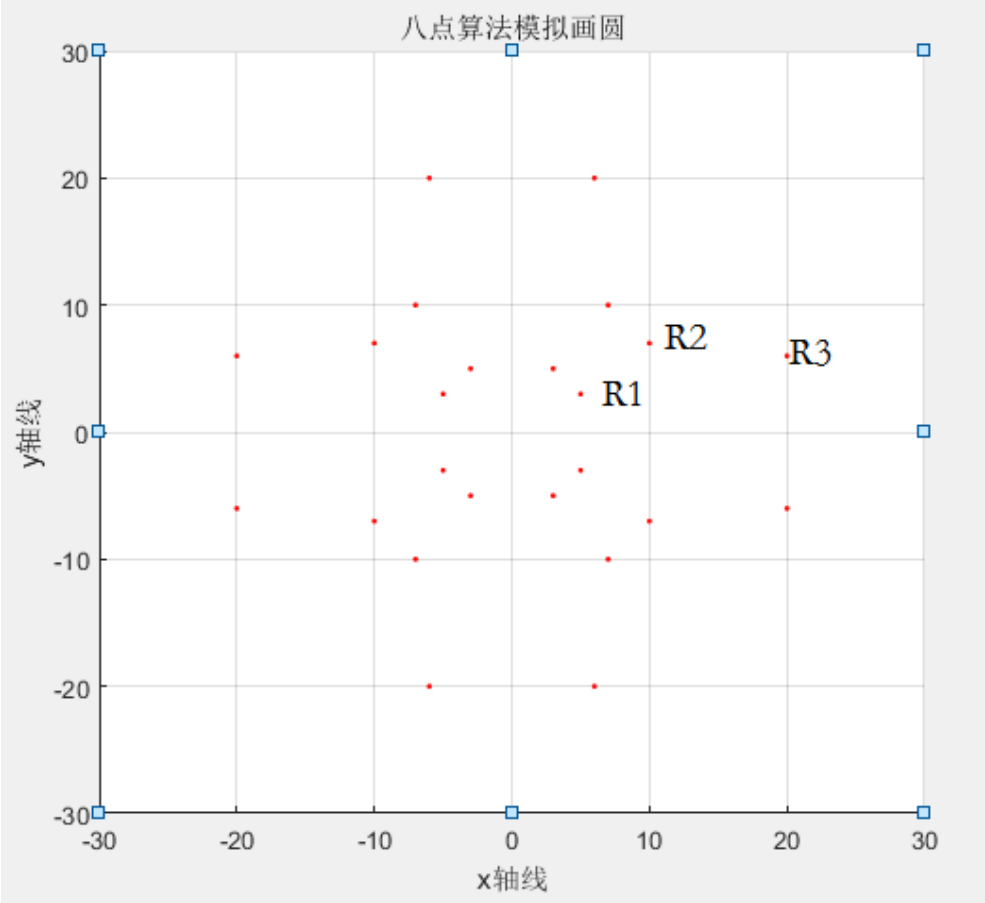
2、输入圆上已知一点得到其他在圆周上关于 4 条对称轴的 7 个点

3、采用 GUI 界面实现该算法

三、实验结果

此次实验我利用八分法画圆画了 3 个圆，每个圆其实是由 8 个点组成的，3 个圆的基点为:R1:(5,3)　R2:(10,7)　R3:(20,6)。

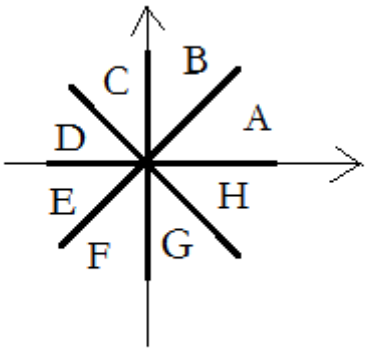
程序运行如下：



四、实验分析

1、实验原理

由于对称性原理，二维坐标轴其实可以根据 y=0,x=0,y=x,y=-x 这 4 条直线划分为如下 8 个区域，我们在画圆时其实只用画出任意一个区域的一段圆弧，然后根据对称原理就可以得到完整的圆。所以在这个实验中我们先验证对称原理，通过 1 个点对称出 8 个点来模拟圆。



2、实验总结

对于任意一点 P(x,y)，对称后的结果为：

(x,y)　　(x,-y)　　(-x,y)　　(-x,-y)

(y,x)　　(y,-x)　　(-y,x)　　(-y,-x)

五、源代码

```
function CirclePoint(x,y)
% CirclePoint(x,y)    八点法算法绘圆
% x                          横坐标
% y                          纵坐标
hold on
plot(x,y,'.r');
plot(y,x,'.r');
plot(-x,y,'.r');
```

```
plot(y,-x,'.r');
plot(x,-y,'.r');
plot(-y,x,'.r');
plot(-x,-y,'.r');
plot(-y,-x,'.r');
hold off
grid on;
xlabel('x 轴线');
ylabel('y 轴线');
title('八点算法模拟画圆');
end
```

圆的 Bresenham 算法

一、实验目的

1、对画圆函数有个基本的了解

2、实现网格的绘制和网格像素的填充

3、熟练掌握圆的中点 Bresenham 算法的实现过程

二、实验内容

1、实现 Bresenham 圆生成算法

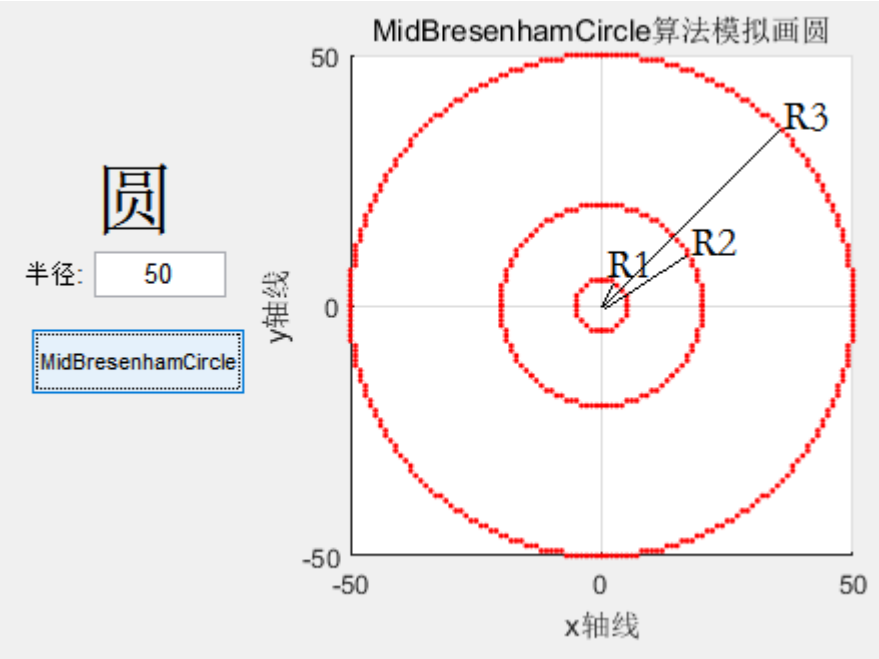
2、输入圆的半径通过原点生成相应的圆

3、采用 GUI 界面实现该算法

三、实验结果

这个实验中利用 Bresenham 算法画出八分之一圆弧，然后每次画点时都调用上个实验封装的八点画圆函数，这样就得到了完整的圆，此次实验共画了 3 个圆，分别是：R1=5，R2=20，R3=50。

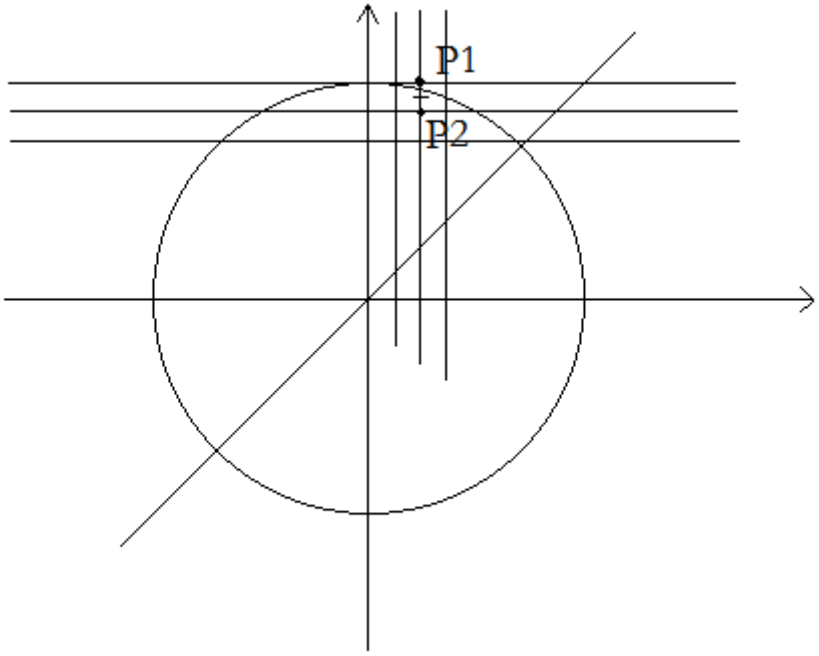
程序运行结果如下：



四、实验分析

1、实验原理

在如下图区域中,取 P1、P2 的中点 M，P1P2 与圆的交点 Q，M、Q 图中未标出，若 M 在 Q 的下方，则说明圆离上面那个点近，画点 P1；若 M 在 Q 的上方，则说明圆离下面那个点近，画点 P2。



判定式：$d_i = \mathrm{F}(x_M, y_M) = \mathrm{F}(x_i + 1, y_i - 0.5) = (x_i + 1)^2 + (y_i - 0.5)^2 - \mathrm{R}^2$

当$d_i < 0$时，M 在 Q 的下方，取 P1，$d_{i+1} = d_i + 2x_i + 3$；当$d_i \geq 0$时，M 在 Q 的上方，取 P2，$d_{i+1} = d_i + 2(x_i - y_i) + 5$。循环判断，直到 y>x 时结束。

2、实验总结

这个画圆算法极大的减小了画圆的计算量，本来要画的完整的圆，在这里只计算了八分之一，然后利用对称原理画出了其余部分，单从执行效率方面还是一个不错的算法。

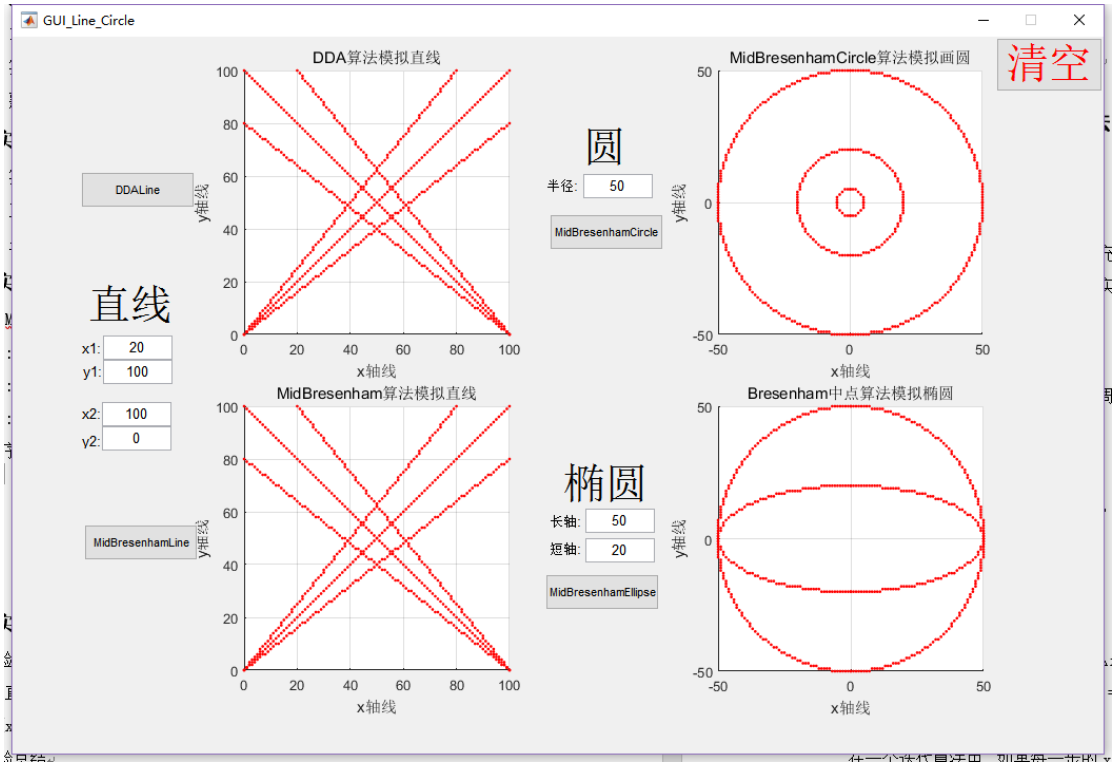
五、源代码

```
function MidBresenhamCircle(r)
% MidBresenhamCircle(r)    Bresenham 中点算法绘圆
% r                                半径
x=0;y=r;d=1-r;
```

```
hold on;
while(x<=y)
    CirclePoint(x,y);
    if(d<0)
        d=d+2*x+3;
    else
        d=d+2*(x-y)+5;
        y=y-1;
    end
    x=x+1;
end
grid on;
xlabel('x 轴线');
ylabel('y 轴线');
title('MidBresenhamCircle 算法模拟画圆');
hold off;
end
```

实验体会

综合这四次实验，个人感觉还是很不错的，掌握了计算机在显图形时后台的算法，自己写了这些算法后，才明白了为什直线不直、圆不圆的这些现象。写这些算法时，感觉还是思想很重要，一个问题想明白了，很快就能写的出来。产生了兴趣之后，又完成了椭圆的算法，并把它们都集成在了 GUI 中，附程序运行图：



之前从来没用过 MATLAB，这次用了以后感觉 MATLAB 使用起来的确十分方便，在学习验证方面很强，上手很容易，但是个人感觉这款软件写的代码很不严谨，可能是自己用其他语言用习惯了吧，对于没有数据类型这一点实在很难接受。总之，这次试验还是很成功的。

附（实验源码以及相关资料已上传至我的 github）：

https://github.com/GuoWenhao1996/ComputerGraphicsProject