**计算机图形学**

**实验报告**

**姓名：徐宇顺**

**学号：031720106**

**一：程序的实现环境**

1. 计算机硬件配置：GTX1050 i5\_7300HQ
2. 操作系统：Windows10
3. 编程环境：visual studio 2017
4. 编程语言：c++

**二：算法思想说明**

1. **实验一：**

（以直线斜率k，k>=0，且k=<1）

端点为P0（X0，Y0）,（X1,Y1），过各行，各列像素中心构造一组虚拟网格线，按直线从起点到终点的顺序计算直线与各垂直网格线的交点，交点与网格线之间的误差为di，根据di确定该列网络中与此交点最近的像素点。当di>0.5时，直线更接近于像素点Pu（Xi+1，Yi+1）；当di<0.5时，更接近于像素点Pd（Xi+1，Yi）；当di=0.5时，与上述两像素一样接近，约定取Pd（Xi+1，Yi）。

误差项di，它的初始值为0，每走一步有di+1=di+k（k=dy/dx，为直线斜率）。一旦y方向上走了一步，就把它减去1,（此时可能出现负误差，这表明交点在所取网格点之下）。为计算方便，令ei=di-0.5，则当ei>0.5时，下一像素的y坐标增加1；否则，下一像素的y坐标不变，此时，ei的初值为-0.5，每走一步有ei+1=ei+k。当ei>0时（相当于y方向上走一步时）将ei减1。

由于算法中只用到误差项的符号，于是可以用2e△x来替代e。算法步骤如下：

1. 输入直线的两端点P0（X0,Y0）P1（X1,Y1）
2. 计算初始值△x，△y，e=-△x，x=X0，y=Y0；
3. 绘制点（x,y）
4. e 更新为e+2△y，判断e的符号，若e>0，则（x，y）更新为（x+1，y+1），同时将e更新为e-2△x，否则（x，y）更新为（x+1，y）。
5. **实验二：**

假设M是PU和Pd的中点，即有M（xi+1，yi-0.5）。则当F（xM，yM）<0时，M在园内，此时Pu离圆弧更近，应取Pu（xi+1，yi）为下一像素点。当F（xM，yM）>0，说明Pd离圆弧更近，应取Pd（xi+1，yi+1）。当F（xM，yM）=0时，在Pu和Pd之中随便取一个即可，约定取Pd。

构造判别式，

di=F（xM，yM）=F（xi+1，yi-0.5）=（xi+1）2+（yi-0.5）2-R2

当di<0时，下一点取Pu（xi+1，yi）；当di>=0，下一点取Pd（xi+1，yi+1）。

现在进行误差项的递推，在di<0的情况下，取正右方像素的Pu（xi+1，yi），欲判断再下一个像素应取哪一个，应计算

di+1=di+2xi+3

所以，沿正右方向，di的增量为2xi+3。di>=0时，应取右下方像素Pd（xi+1，yi-1），要判断再下一个像素，则要计算

di+1=di+2（xi-yi）+5

所以，沿右下方向，判别式di的增量为2（xi-yi）+5

di的初始值为 d0=1.25-R

由于使用的只是di的符号，因此可以用di-0.25代替di来摆脱小数。此时算法只涉及整数运算，这样，初始化运算d0=1.25-R对应于d0=1-R，判别式di<0对应于di<-0.25，此时的di始终是整数,di<-0.25等价于di<0.

画圆算法步骤如下：

1. 输入圆的半径R
2. 计算初始值d=1-R，x=0，y=R
3. 绘制点（x，y）及其在八分圆中的另外7个对称点
4. 判断d的符号。若d<0，则先将d更新为d+2x+3，再将（x，y）更新为（x+1，y）；否则先将d更新为d+2（x-y）+5，再将（x，y）更新为（x+1，y-1）
5. 当x<y时，重复步骤3和4，否则结束。
6. **实验三：**

若点P1和P2完全在裁剪窗口内，则该直线段完全可见，“简取”之

若点P1和P2均在窗口外，且在窗口的同一侧，即满足4个条件之一：

x1<xw1且x2<xw1

x1>xwr且x2>xwr

y1<ywb且y2<ywb

y1>ywt且y2>ywt

则该直线段完全不可见，“简弃”之

若直线段两个条件都不满足，则该直线段可能与窗口相交，此时需要对直线段按交点进行分段，分段后重复上述处理

处理时，首先对直线段的断电进行编码，即对于直线段的任意端点（x,y），根据其坐标所在的区域，赋予它一个4位的二进制编码D3D2D1D0.编码规则如下：

若x<xwl，则D0=1，否则D0=0

若x>xwr,则D1=1，否则D1=0

若y<ywb，则D2=1，否则D2=0

若y>ywt,则D3=1，否则D3=0

根据编码规则，对直线的P1,P2编码得到code1，code2

若code1|code2=0，对直线段全取

若code1&code2!=0，就弃掉

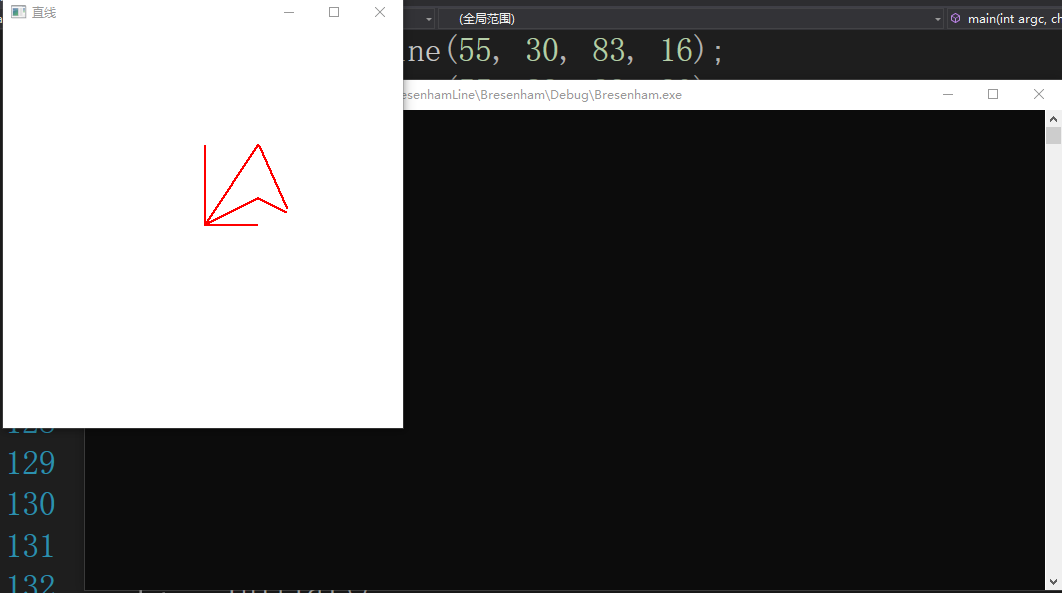
若均不成立，则求出直线段和窗口的交点，在交点处把直线一分为二，若另一端也在窗口外，则重复求出交点，舍弃掉直线外的，直到找到全部位于窗口内的直线。

**三：程序方法使用说明**

1. **实验一：**直接执行
2. **实验二：**在exe中输入半径的大小，按下回车，窗口内会出现绘制出的圆
3. **实验三：**初始有边界和一个直线；若需要裁剪，在窗口界面按下c会进行裁剪，裁剪后，按下x就会退出程序。若在边界内或在边界外则不需要裁剪。

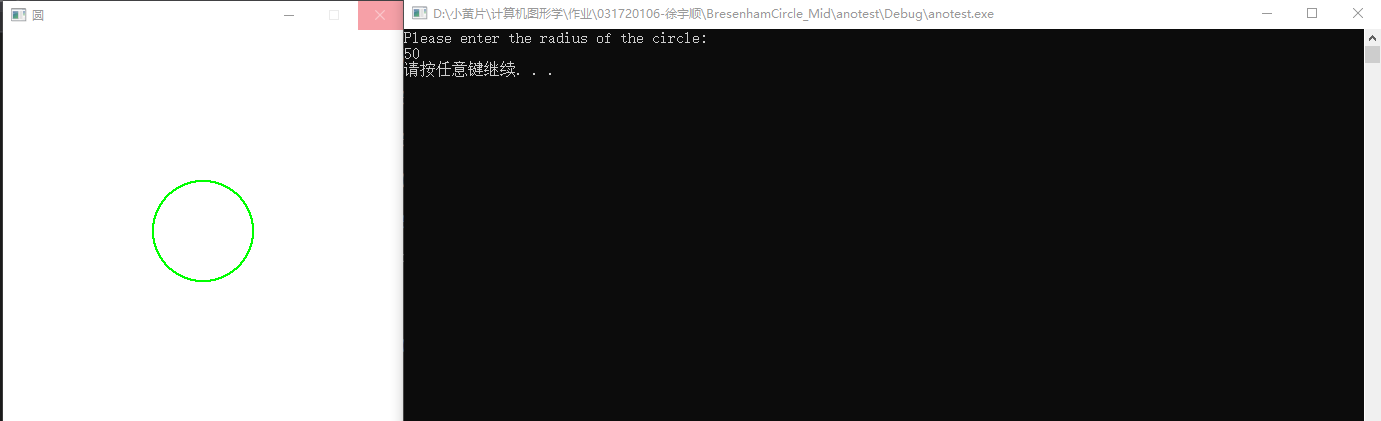
四：**实验结果：**

1. **实验一：**我选择的点为{（2,3），（55,83）}，{（2,3），（55,30）}，{（2,3），（2,83）}，{（2,3），（55,3）}，{（55,30），（83,16））}，{（55,83），（83，20）}六条直线，包括了斜率k>1,0<k<=1,-1=<k<0,k<-1.



运行结果如上图。

1. **实验二：**在exe中输入50，即可绘制一个半径为50的圆.



即如上图。

1. **实验三：**

输入直线的起止点（250,50），（50,250），在窗口上回绘制出矩形边界，和直线。



在窗口范围内，按下c,即可裁剪直线：



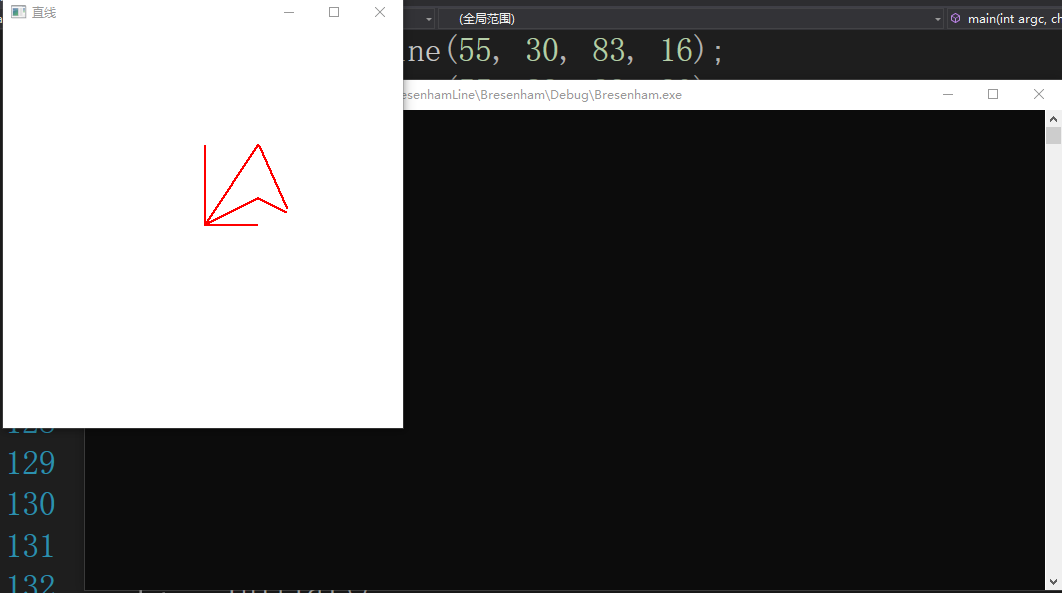
按下x即可退出。

**五：程序使用范围：**

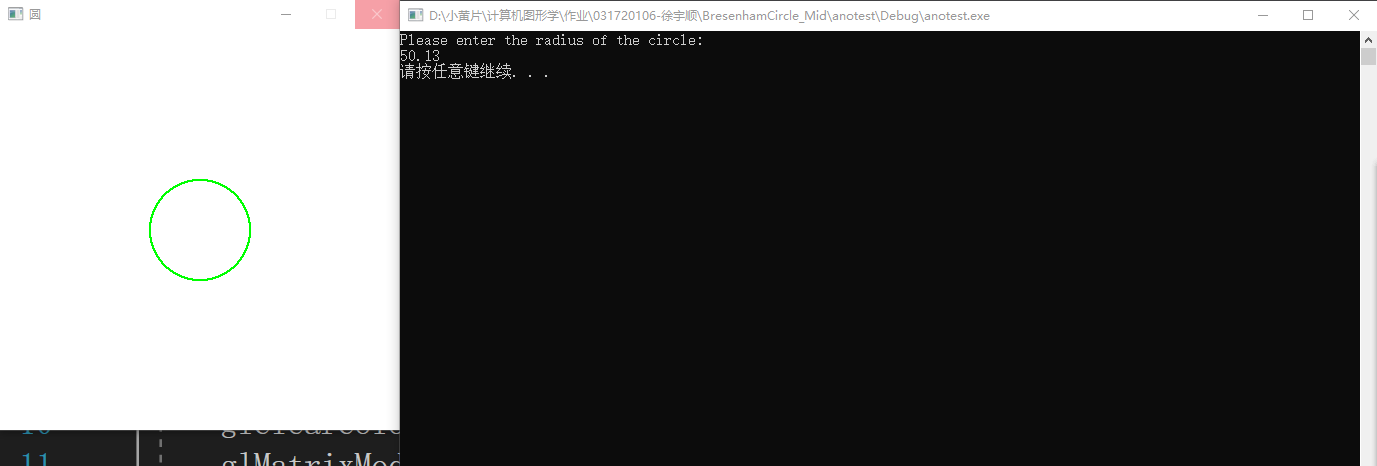
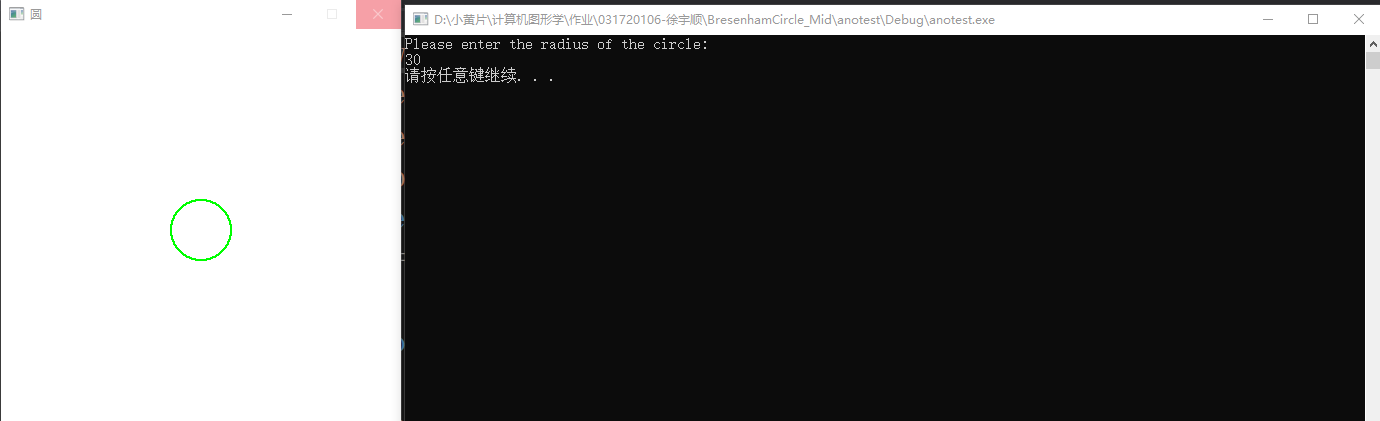
1. **实验一：**适用于任意斜率
2. **实验二：**使用于任意半径的花园
3. **实验三：**适用于任何情况的直线裁剪

六：测试

1. **实验一：**{（2,3），（55,83）}，{（2,3），（55,30）}，{（2,3），（2,83）}，{（2,3），（55,3）}，{（55,30），（83,16））}，{（55,83），（83，20）}运用了以上这6条直线作为每个不同斜率范围内的例子。其中包括，k>1，k=<1&&k>0，k<0&&k>=-1，k<-1，每个都已经遍历。



1. **实验二：**



适用于整数，也适用于浮点数。

1. **实验三：**

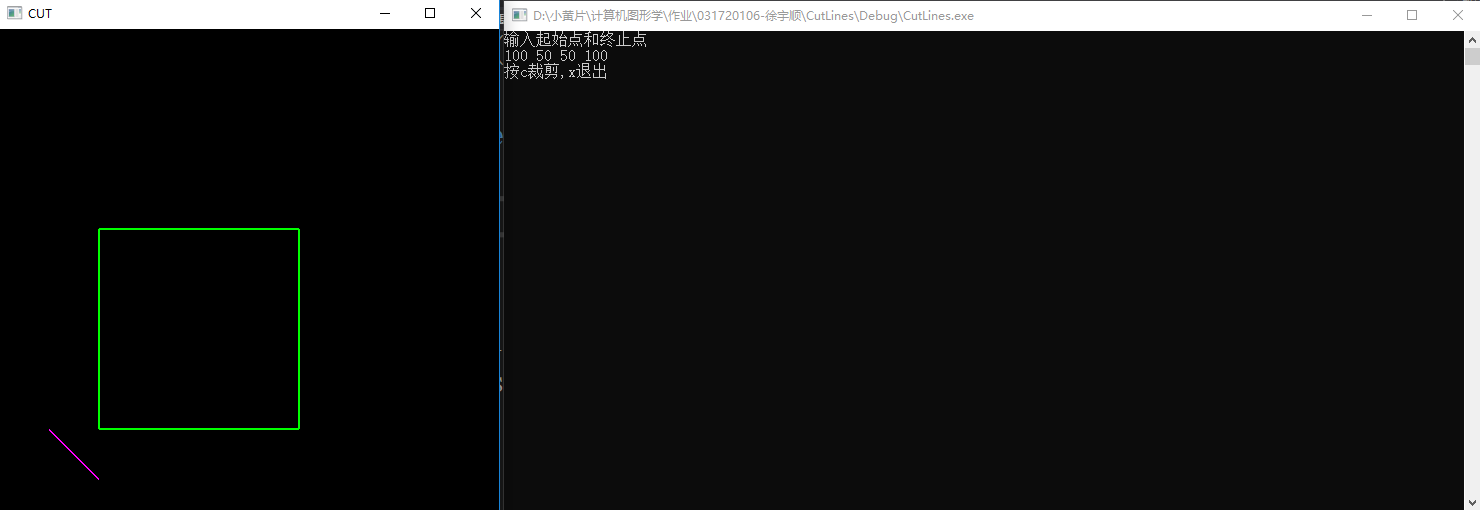
输入直线的起止点（250,50），（50,250），在窗口上回绘制出矩形边界，和直线。



在窗口范围内，按下c,即可裁剪直线：



按下x即可退出。



输入（100,50），（50,100）按下c则不会进行裁剪，但按下x依然退出。