华东师范大学计算机科学与技术实验报告

实验课程：计算机图形学	年级：2018	实验成绩：
实验名称：画圆算法	姓名：李泽浩	实验日期：2021/03/30
实验编号：4	学号：10185102142	实验时间：13:00-14:40
指导教师：李洋	组号：	

一、实验目的

通过两种算法，进行圆的绘制，并尝试绘画椭圆。

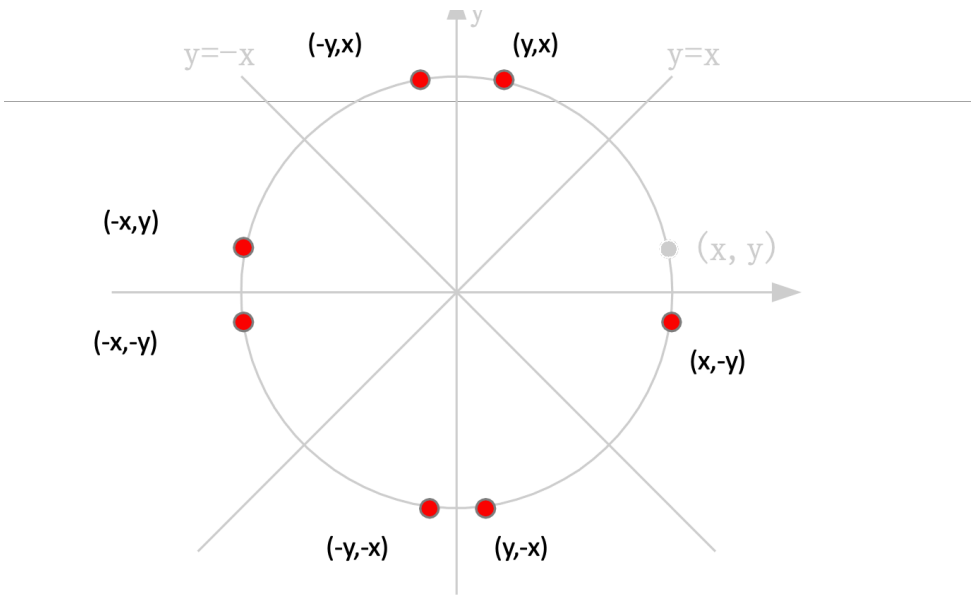
二、实验环境

windows 10 & C++ & visual stadio 2019

macos & python 3.7

三、实验内容

算法分析 8分法画圆：



【1】简单算法画圆

算法原理：利用其函数方程，直接离散计算

圆的函数方程为：$x^2 + y^2 = R^2$

$$x_{i+1} = x_i + 1 \quad x \in [0, R/\sqrt{2}]$$
$$y_{i+1} = round(\sqrt{R^2 - x_{i+1}^2})$$

【2】Bresenham算法

判别式：$d_i = F(x_M, y_M) = (x_i+1)^2 + (y_i - 0.5)^2 - R^2$

$$d_0 = 1.25 - R$$

当$d_i <= 0$：

$$\begin{cases} x_{i+1} = x_i + 1, \quad y_{i+1} = y_i \\ d_{i+1} = d_i + 2x_i + 3 \end{cases}$$

当$d_i > 0$：

$$\begin{cases} x_{i+1} = x_i + 1, \quad y_{i+1} = y_i - 1 \\ d_{i+1} = d_i + 2(x_i - y_i) + 5 \end{cases}$$

伪代码

- 0≤k≤1时Bresenham算法的算法步骤为：
- 1.输入直线的两端点P0(x0,y0)和P1(x1,y1)。
- 2.计算初始值△x、△y、d=0.5-k、x=x0、y=y0；
- 3.绘制点(x,y)。判断d的符号；
  - 若d<0，则(x,y)更新为(x+1,y+1)，d更新为d+1-k；
  - 否则(x,y)更新为(x+1,y)，d更新为d-k。
- 4.当直线没有画完时，重复步骤3。否则结束。

四、实验过程与分析

简单算法绘制圆

```
// implement draw circle with simple algorithm
void draw_circle_simple(int xc, int yc, int r,
```

```
                                    std::vector<std::pair<int, int>> &buffer) {
    int x=0,y=0;
    for(int i=0;i<r/sqrt(2);i++)
    {
        y = round(sqrt(r*r - x*x));
        x = i;
        buffer.emplace_back(y+xc, x+yc);
        buffer.emplace_back(x+xc, y+yc);
        buffer.emplace_back(x+xc, -y+yc);
        buffer.emplace_back(y+xc, -x+yc);
        buffer.emplace_back(-y+xc, -x+yc);
        buffer.emplace_back(-x+xc, -y+yc);
        buffer.emplace_back(-x+xc, y+yc);
        buffer.emplace_back(-x+xc, y+yc);
    }
}
```

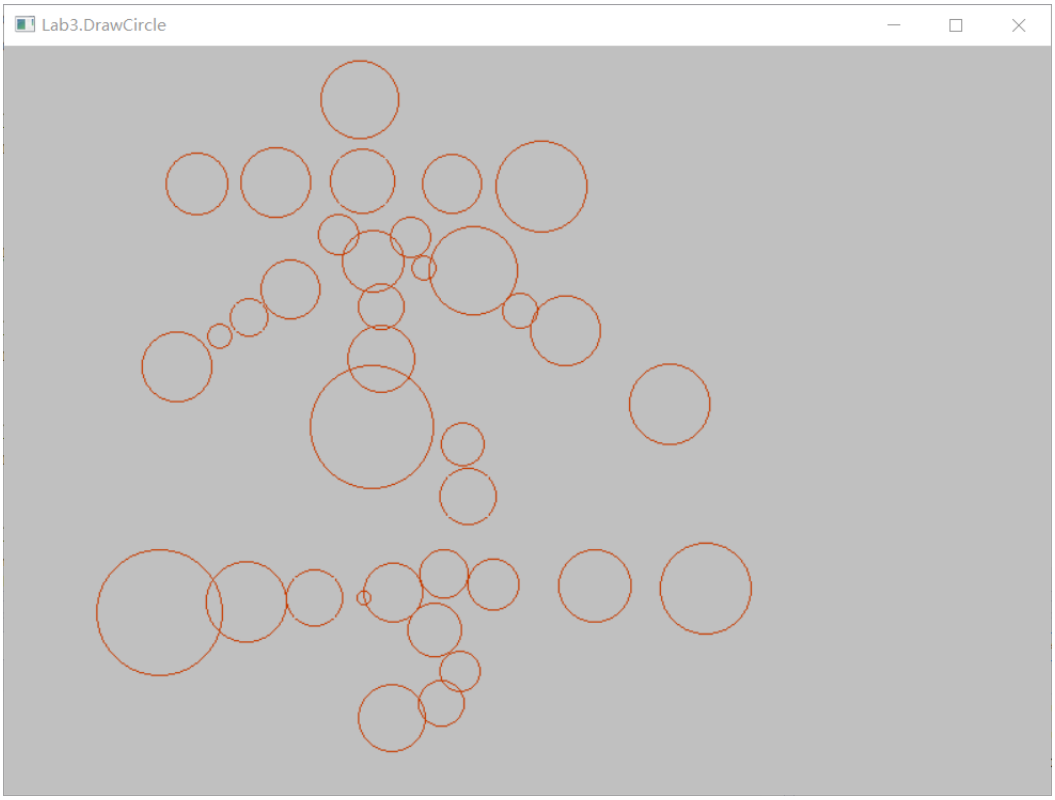
Bresenham算法绘制圆

```
// implement draw circle with midpoint algorithm
void draw_circle_midpoint_bresenham(int xc, int yc, int r,
                                    std::vector<std::pair<int, int>> &buffer) {
        int d = 1.25 - r;
        int x = 0, y = r;
        while (x <= y)
        {
            if(d < 0)
            {
                d = d + 2*x + 3;
                x++;
                buffer.emplace_back(y+xc, x+yc);
                buffer.emplace_back(x+xc, y+yc);
                buffer.emplace_back(x+xc, -y+yc);
                buffer.emplace_back(y+xc, -x+yc);
                buffer.emplace_back(-y+xc, -x+yc);
                buffer.emplace_back(-x+xc, -y+yc);
                buffer.emplace_back(-x+xc, y+yc);
                buffer.emplace_back(-x+xc, y+yc);
            }
            else
            {
                d = d + 2 * (x - y) + 5;
                x++;
                y--;
                buffer.emplace_back(y+xc, x+yc);
                buffer.emplace_back(x+xc, y+yc);
                buffer.emplace_back(x+xc, -y+yc);
```

```
                buffer.emplace_back(y+xc, -x+yc);
                buffer.emplace_back(-y+xc, -x+yc);
                buffer.emplace_back(-x+xc, -y+yc);
                buffer.emplace_back(-x+xc, y+yc);
                buffer.emplace_back(-x+xc, y+yc);
            }
        }
}
```

五、实验过程总结

生成的圆和椭圆图片如下：



六、附录

利用Python实现中点法绘制圆和椭圆

圆(circle.py)：

```
# -*- coding: utf-8 -*-
import wx
import time

class CIRCLE(wx.Frame):
    def __init__(self):
```

```
        #定义窗口的大小和位置
        super().__init__(None, title='中点画圆法', size=(1000, 1000), pos=(30,
30))

        #输入参数
        self.x, self.y, self.r = map(int, input("请依次输入圆心坐标和半径:
\n").split())
        #将事件与函数进行绑定
        self.Bind(wx.EVT_PAINT, self.On_Paint)

        #颜色表
        self.color = '#0000FF'

    #绘图函数
    def DrawCircle(self, a, b, R, dc):
        L4=[] #存储第四象限的数组(对于窗口的坐标系而言是第一象限)
        x, y, p = 0, R, 1 - R
        #将p设置为1-R能够避免浮点运算，而且其绘图的拟合程度较好
        while (x < y):
            if p < 0:
                p += (2 * x + 3)
            else:
                p += (2 * x - 2 * y + 5)
                y -= 1
            x+=1
            L4.append((x,y)) #八分之一点的集合
            L4.append((y,x)) #将点做关于y=-x(对于窗口坐标系而言是y=x)的对称
            #L4所得点的集合是四分之一圆弧的点

        L3 = [(-1*e[0],e[1])for e in L4] #第三象限
        L2 = [(-1 * e[0], -1*e[1]) for e in L4] #第二象限
        L1 = [(e[0], -1*e[1]) for e in L4] #第一象限

        #为绘图时按照顺时针绘画，将各个象限进行排序
        L1.sort()
        L2.sort()
        L3.sort(reverse=True)
        L4.sort(reverse=True)

        #按照一四三二的象限顺序生成总的点集
        p_points = [(e[0]+a,e[1]+b)for e in L1+L4+L3+L2]
        points = list(set(p_points))
        points.sort(key=p_points.index)

        #这里利用time.sleep显示出动画效果
        #为了颜值设置了每隔STEP个点换一支画笔颜色
        STEP = 400
        for i,point in enumerate(points):
            if i % STEP == 0:
```

```
                dc.SetPen(wx.Pen(self.color))
            dc.DrawPoint(point)
            time.sleep(0.003)

    #定义绘图消息的接收
    def On_Paint(self, event):
        dc = wx.PaintDC(self)
        a, b, R = self.x, self.y, self.r
        self.DrawCircle(a, b, R, dc)

app = wx.App() #app实例化

circle = CIRCLE() #窗口类实例化

circle.Show() #进行绘图

app.MainLoop() #主循环
```

椭圆(oval.py):

```
import wx
import time


class OVAL(wx.Frame):
    def __init__(self):
        #定义窗口的大小和位置
        super().__init__(None, title='中点画椭圆', size=(1000, 1000), pos=(30,
30))

        #输入参数
        self.Ox, self.Oy, self.a, self.b = map(
            int,
            input("请依次输入椭圆中心坐标和长短半轴：\n").split())
        #将事件与函数进行绑定
        self.Bind(wx.EVT_PAINT, self.On_Paint)

        #颜色
        self.color = '#0000FF'

    #绘图函数
    def DrawOval(self, Ox, Oy, a, b, dc):
        changed = 0 if a > b else 1  #用来判断取x或y为主方向
        a, b = max(a, b), min(a, b)
        point_list4 = []  #存储第四象限的点集
        A = a**2
        B = b**2
```

```
        x0 = int((A**2 / (A + B))**0.5)  #记录切线斜率为1的点横轴坐标
        x = 0
        y = b

        p = B + A * (0.25 - b)

        while x <= x0:
            if changed:
                point_list4.append((y, x))
            else:
                point_list4.append((x, y))
            if p < 0:
                p += B * (3 + 2 * x)
            else:
                p += B * (3 + 2 * x) + A * (2 - 2 * y)
                y -= 1
            x += 1

        p = B * (x + 0.5)**2 + A * (y - 1)**2 - A * B

        while y > 0:
            if changed:
                point_list4.append((y, x))
            else:
                point_list4.append((x, y))
            if p > 0:
                p += A * (3 - 2 * y)
            else:
                p += A * (3 - 2 * y) + B * (2 * x + 2)
                x += 1
            y -= 1
        point_list1 = [(x, -1 * y) for x, y in point_list4]  #第一象限的点
        point_list2 = [(-1 * x, -1 * y) for x, y in point_list4]  #第二象限的点
        point_list3 = [(-1 * x, y) for x, y in point_list4]  #第三象限的点
        #设置顺时针顺序进行绘画
        point_list1.sort()
        point_list2.sort()
        point_list3.sort(reverse=True)
        point_list4.sort(reverse=True)
        points_O = point_list1 + point_list4 + point_list3 + point_list2

        #平移坐标系
        points = [(x + Ox, y + Oy) for x, y in points_O]

        #这里利用time.sleep显示出动画效果
        #为了颜值设置了每隔STEP个点换一支画笔颜色

        STEP = 400
        for i, point in enumerate(points):
```

```
            if i % STEP == 0:
                dc.SetPen(wx.Pen(self.color))
            dc.DrawPoint(point)
            time.sleep(0.002)

    #定义绘图消息的接收
    def On_Paint(self, event):
        dc = wx.PaintDC(self)
        self.DrawOval(self.Ox, self.Oy, self.a, self.b, dc)


app = wx.App()  #app实例化

oval = OVAL()  #窗口类实例化

oval.Show()  #进行绘图

app.MainLoop()  #主循环
```