华东师范大学计算机科学与技术学院实验报告

实验课程：计算机图形学	**年级：**2018 级	**实验成绩：**
实验名称：二维几何算法	**姓名：**董辰尧	
实验编号：5	**学号：**10185102144	**实验日期：**2021-4-6
指导教师：王长波、李洋	**组号：**	**实验时间：**13:00-14:30

一、实验目的

利用操作系统 API 实现基本二维几何操作。

二、实验内容与实验步骤

实现基本 2 维向量操作
计算两个线段的相交点的坐标
判断一个点是否在多边形内部

三、实验环境

Visual studio 2017

四、实验过程与分析

这次试验助教已经非常好的写出了框架，只需要我们补充代码就好了。
和原来一样，创建一个空项目，先导入代码

接下来的实验可以分为 3 步
1.在 geometry.hpp 中完成向量类 struct Vector 的定义。

```cpp
struct Vector {
    float x, y;
    Vector(float x, float y) : x(x), y(y) {}

    Vector operator+(const Vector &rhs) const { return  Vector(x + rhs.x, y + rhs.y); }
    Vector operator-(const Vector &rhs) const { return  Vector(x - rhs.x, y - rhs.y); }
    Vector operator*(float t) const { return   Vector(x * t, y * t); }
    // dot product
    float dot_prod(const Vector &rhs) const { return x * rhs.x + y * rhs.y; }

    // cross product in 2D return the directed area of triangle between u and v
    float cross_prod(const Vector &rhs) const { return x * rhs.y - y * rhs.x; }
};
```

向量的加减点乘叉乘等等已经在上节课讲过了

2.实现 geometry.cpp 中判断直线交点函数。

```cpp
bool is_intersection(const Segment &A, const Point &p) {
    // @TODO: implement here
      Vector a(A.e.x - A.s.x, A.e.y - A.s.y);
      Vector b(p.x - A.s.x, p.y - A.s.y);
      float r;
      r = a.cross_prod(b);
      r = int(r + 0.5);
      if (r == 0 and p.x <= max(A.e.x, A.s.x) and p.y <= max(A.e.y, A.s.y) and p.x >= min(A.e.x, A.s.x) and p.y >=min(A.e.y, A.s.y))
      {
          return true;
      }
    return false;

}

// return false means no intersection
// the intersection point should store in 'inter_point'
bool calc_intersection(const Segment &A, const Segment &B, Point &inter_point) {
    // @TODO: implement here
    // ax + by + c = 0
      float a1, b1, c1, a2, b2, c2, x, y;
      a1 = A.s.y - A.e.y;
      b1 = A.e.x - A.s.x;
      c1 = A.s.x * A.e.y - A.e.x * A.s.y;
      a2 = B.s.y - B.e.y;
      b2 = B.e.x - B.s.x;
      c2 = B.s.x * B.e.y - B.e.x * B.s.y;
      x = (b1 * c2 - c1 * b2) / (b2 * a1 - a2 * b1);
      y = (a1 * c2 - a2 * c1) / (a2 * b1 - a1 * b2);
      inter_point.x = x;
      inter_point.y = y;
      if (is_intersection(A, inter_point) and is_intersection(B, inter_point))
      {
          return true;
      }
    return false;
```

这个函数分为两部分：

一个是判断点是否在线段上的函数，我选择让线段上的两个点和另一个点组成了两个向量，然后让这两个向量进行叉乘，如果结果为 0 的话，证明三点一线，然后再判断点的横纵坐标是否在线段端点横纵坐标以内。

第二个是求交点，由于两条线段已知，就可以确定两条直线方程，解方程即可得到交点。接着用上面判断点是否在线上的函数检验交点是在线段上还是在线段延长线上。

3.判断点是否在多边形内的函数。

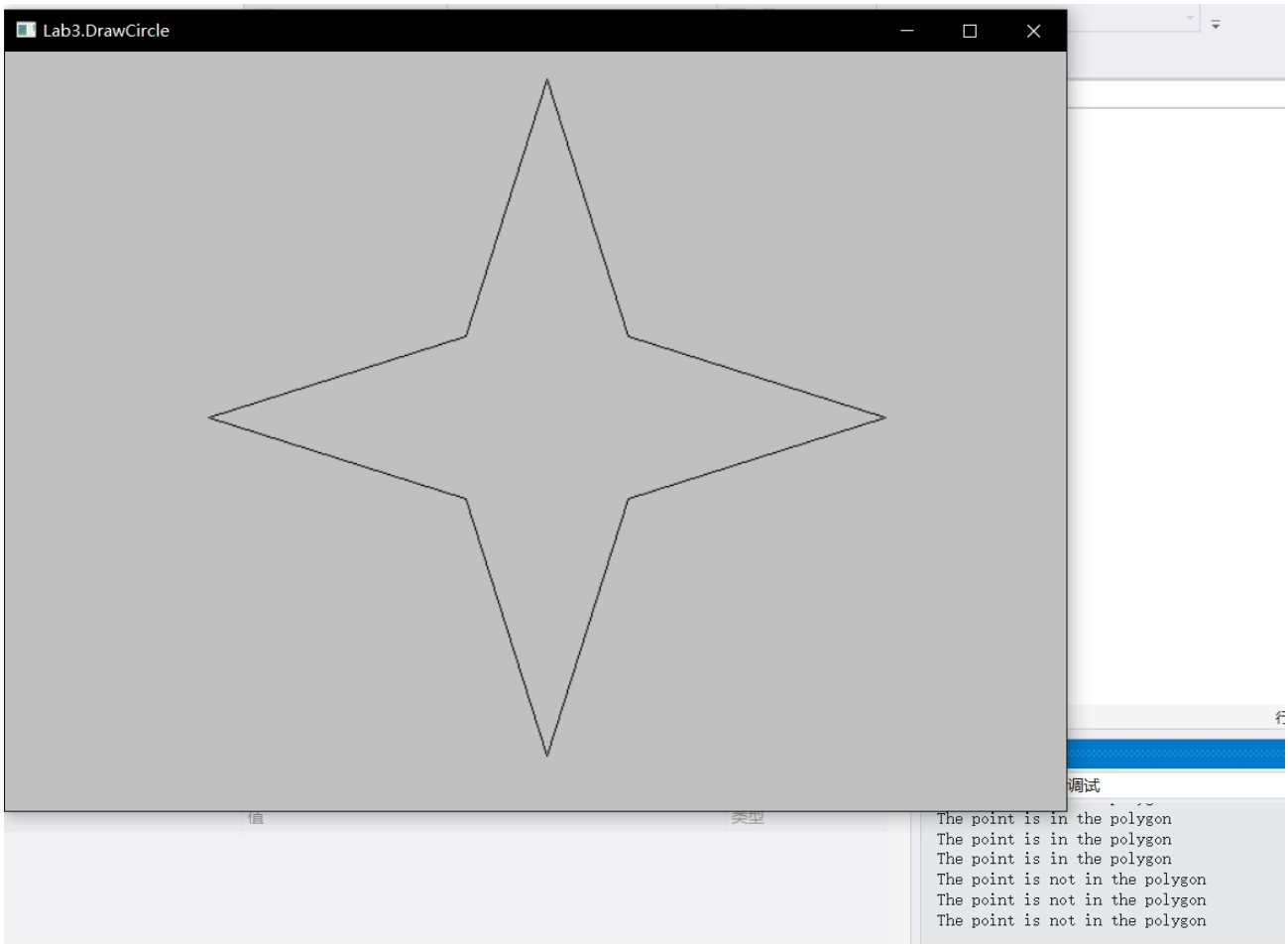
```
130  bool is_point_inside_polygon(const Point &p, const Polygon &poly) {
131    // @TODO: implement here
132      int nCount = poly.size();
133      int nCross = 0;
134      for (int i = 0; i < nCount; i++)
135      {
136          Point p1 = poly[i];
137          Point p2 = poly[(i + 1) % nCount];// 点P1与P2形成连线
138
139          if (p1.y == p2.y)
140              continue;
141          if (p.y < min(p1.y, p2.y))
142              continue;
143          if (p.y >= max(p1.y, p2.y))
144              continue;
145          // 求交点的x坐标（由直线两点式方程转化而来）
146
147          double x = (double)(p.y - p1.y) * (double)(p2.x - p1.x) / (double)(p2.y - p1.y) + p1.x;
148
149          // 只统计p1p2与p向右射线的交点
150          if (x > p.x)
151          {
152              nCross++;
153          }
154
155      }
156      if ((nCross % 2) == 1) return true;
157
158
159    return false;
160
161  }
```

我用的画射线法，从点做任意一条射线吗，数一下交点的数量，奇数就在内部，否则在外部。由于可以画任意射线，我选择了和 y 轴平行的射线，计算比较方便，然后计算交点即可。

五、实验结果总结

1. 求交点：有交点则显示无交点则不显示

2. 判断点是否在封闭图形内部。


Lab3.DrawCircle
调试
The point is in the polygon
The point is in the polygon
The point is in the polygon
The point is not in the polygon
The point is not in the polygon
The point is not in the polygon