计算机学院实验报告

实验题目： 两种平面三角形点定位算法		学号：202300130183
日期：2025/3/3	班级：23 级智能班	姓名：宋浩宇

Email：2367651943@qq.com
202300130183@mail.sdu.edu.cn

实验目的：

1， 第一种算法为射线求交法（根据交点个数判定）；
2， 第二种算法任选；
3， 有余力的同学欢迎实现第三种算法；
4， 在实验 0 的平台上开发；
5， 要求写实验报告，具体提交方式助教发布；
6， 实验验收：原则上，本周实验，下周助教验收。本次实验，要求 3 月 9 号 24 点前完成实验报告提交，最晚在 3 月 10 号的实验课上完成验收，速度快的同学本周实验科可以找助教验收。
7， 另，请大家把 2 月 25 号课程上提到的，证明叉乘公式，融合在本实验一的报告中。

实验环境介绍：

软件环境：
主系统：Windows 11 家庭中文版 23H2 22631.4317
虚拟机软件：Oracle Virtual Box 7.1.6
虚拟机系统：Ubuntu 18.04.2 LTS
编辑器：Visual Studio Code
编译器：gcc 7.3.0
计算框架：Eigen 3.3.7

硬件环境：
CPU：13th Gen Intel(R) Core(TM) i9-13980HX 2.20 GHz
内存：32.0 GB (31.6 GB 可用)
磁盘驱动器：NVMe WD_BLACKSN850X2000GB
显示适配器：NVIDIA GeForce RTX 4080 Laptop GPU

解决问题的主要思路：

1. 关于实现射线求交法，对于非边界情况，我们首先要实现一个判断射线与线段是否有交点的算法，我们在此处使用叉积法来求解线段和射线的交点。

   对于射线 d 和线段的向量$\overrightarrow{AB}$，易得当二者的叉乘为 0 时，二者平行或共线，

   因此当二者叉乘为 0 时，我们判断一下 p 是否在该线段上，如果在则相交，如果不在则不相交。当二者的叉乘不为 0 时，我们可以计算线段和射线的交点位置。

   首先射线和线段可以用参数方程表示，分别为：

$$P(t) = P_0 + t \cdot \vec{d}, t \in [0, +\infty)$$

$$Q(s) = A + s \cdot (B - A), s \in [0,1]$$

   那么我们如果想要获取它们的交点，实际就是在求 P(t)＝Q(s)时 t 和 s 的取值，可以得到

$$t \cdot \vec{d} - s \cdot (B - A) = A - P_0$$

   然后我们就可以解出来 t 和 s 的值

$$t = \frac{(A - P_0) \times (B - A)}{\vec{d} \times (B - A)}$$

$$s = \frac{\vec{d} \times (A - P_0)}{\vec{d} \times (B - A)}$$

   接下来我们只需要判断 t 和 s 的取值是否在定义域里即可确定射线是否过线段。

   此外我们还需要处理额外的特殊情况。首先对于边界情况，即射线过三角形的顶点时和与三角形的一边共线时。对于我们的算法，我们判断点 P 发出的射线与三角形的交点时是轮流和三个线段判断是否相交，如果交点为 1 个，则 P 在三角形内部，而如果这个射线刚好经过三角形的顶点，则对于我们的算法来说，因为同时经过两个线段，所以也会被判定为有两个交点，因此我们可以在交点有两个时拓展我们的判定，我们用四个射线来进行判断，这四个射线相邻两个之间的夹角均为 90°，只要这四个射线有一个与三角形的交点个数是 1，就可以确定该点在三角形内，如果四个射线的交点都不是 1，则可以确定该点不在三角形内。另外还有一种处理方案，即设计射线方向时不是随意设计，而是取 PA 的方向向量，在加一个很小的偏转角度，当然这个很小的角度如果是固定值也有可能还是过这个交点，因此我们可以设置这个偏转角的角度为∠BPA 的一半，不过这个只是我的思路，我实际在实现算法的时候使用的是取四个射线的方法。而对于共线的情况，在上文已经提出过解决方法，此处不在赘述。

2. 第二种方法，我们使用的是面积法。这种方法思路上比较简单，通过海伦公式，我们可以通过三角形的边长计算出三角形的面积。而对于点 P 来说，我们有以下两个关系：

$$S_{\triangle ABC} = S_{\triangle PAB} + S_{\triangle PBC} + S_{\triangle PAC}, P \in \{(x,y) | (x,y) \in \triangle ABC\}$$

$$S_{\triangle ABC} < S_{\triangle PAB} + S_{\triangle PBC} + S_{\triangle PAC}, P \notin \{(x,y)|(x,y) \in \triangle ABC\}$$

因此我们可以通过三角形的面积关系很容易地判断出 P 是否在三角形内部。

3. 第三种方法，这种方法主要是考虑到，可以用三角形边的向量作为基向量来表示 P 的坐标，即$\overrightarrow{AP} = u\overrightarrow{AB} + v\overrightarrow{AB}$

   我们可以分析得到，如果 P 想要在三角形的内部，我们必须要保证 u 和 v 都是大于 0 的，且两者的和应该是在[0,1]这个区间上的。因此我们只需要解出来 u 和 v 的大小即可判断 P 是否在三角形内部。
   想要解出来这两个未知数的值，我们需要以下这两个方程：

$$\overrightarrow{AP} \cdot \overrightarrow{AC} = (u\overrightarrow{AC} + v\overrightarrow{AB}) \cdot \overrightarrow{AC}$$

$$\overrightarrow{AP} \cdot \overrightarrow{AB} = (u\overrightarrow{AC} + v\overrightarrow{AB}) \cdot \overrightarrow{AB}$$

   而其中这些向量的值我们都是知道的，因此，我们可以解出 u 和 v 的值分别为：

$$u = \frac{(\overrightarrow{AB} \cdot \overrightarrow{AB}) \cdot (\overrightarrow{AP} \cdot \overrightarrow{AC}) - (\overrightarrow{AC} \cdot \overrightarrow{AB}) \cdot (\overrightarrow{AP} \cdot AB)}{(\overrightarrow{AC} \cdot \overrightarrow{AC}) \cdot (\overrightarrow{AB} \cdot \overrightarrow{AB}) - (\overrightarrow{AC} \cdot \overrightarrow{AB}) \cdot (\overrightarrow{AB} \cdot \overrightarrow{AC})}$$

$$v = \frac{(\overrightarrow{AC} \cdot \overrightarrow{AC}) \cdot (\overrightarrow{AP} \cdot \overrightarrow{AB}) - (\overrightarrow{AC} \cdot \overrightarrow{AB}) \cdot (\overrightarrow{AP} \cdot AC)}{(\overrightarrow{AC} \cdot \overrightarrow{AC}) \cdot (\overrightarrow{AB} \cdot \overrightarrow{AB}) - (\overrightarrow{AC} \cdot \overrightarrow{AB}) \cdot (\overrightarrow{AB} \cdot \overrightarrow{AC})}$$

   然后，我们只需要按上述方法判断即可。

4. 最后是关于叉乘计算公式的证明。

   在$\mathbb{R}^3$中我们有一组正交向量的单位向量$\vec{\imath}, \vec{j}, \vec{k}$,这三个向量可以展开这个空间。那我们假设有两个向量 a,b，我们就可以将 a 和 b 表示为$a = a_x i + a_y j + a_z k$和$b = b_x i + b_y j + b_z k$，我们根据叉乘的定义，向量对自己叉乘为 0，两个向量叉乘获得垂直于这两个向量张成的面的向量，长度是这两个向量的模长的积乘这两个向量夹角的正弦值，叉乘的两项交换后结果取相反数，且叉乘满足结合律，因此有以下推导：

$$\begin{aligned} a \times b &= (a_x i + a_y j + a_z k) \times (b_x i + b_y j + b_z k) \\ &= a_x i \times (b_x i + b_y j + b_z k) + a_y j \times (b_x i + b_y j + b_z k) \\ &\quad a_z k \times (b_x i + b_y j + b_z k) \\ &= a_x b_x (i \times i) + a_x b_y (i \times j) + a_x b_z (i \times k) + \\ &\quad a_y b_x (j \times i) + a_y b_y (j \times j) + a_y b_z (j \times k) + \\ &\quad a_z b_x (k \times i) + a_z b_y (k \times j) + a_z b_z (k \times k) \\ &= (a_y b_z - a_z b_y) i + (a_z b_x - a_x b_z) j + (a_x b_y - a_y b_x) k \end{aligned}$$

至此，我们本次实验的所有推导和问题解决都完成了。

实验步骤与实验结果：

一、射线交点个数判别法

首先我们先完成射线交点法判断点是否在三角形的内部。

根据上一部分进行的推导我们可以将过程拆分成几个函数。

Vector2f 类型的叉乘的定义，有以下代码：

```
double vec2_cross(Vector2f a, Vector2f b)
{
    return a.x() * b.y() - a.y() * b.x();
}
```

判定射线是否过线段，有以下代码：

```
bool is_crossed(Vector2f A, Vector2f B, Vector2f P, Vector2f p0)
{
    if (vec2_cross(p0, B - A) == 0.0)
    {
        double k = (B.y() - A.y()) / (B.x() - A.x());
        double b = A.y() - k * A.x();
        double x = (P.y() - b) / k;
        double y = k * x + b;
        if (x >= 0 && x <= 1 && y >= 0 && y <= 1)
        {
            return true;
        }
        return false;
    }
    double tem1 = vec2_cross(A - P, B - A);
    double tem2 = vec2_cross(p0, B - A);
    double tem3 = vec2_cross(A - P, p0);
    double t = tem1 / tem2;
    double s = tem3 / tem2;
    if (t >= 0 && s >= 0)
    {
        return true;
    }
    return false;
}
```

在这个函数中，我们内置了判定在共线的情况下射线的起点是否在线段上的判断。

计算射线与三角形交点个数（顶点处记为两个）

```
int rectangle_cross(Vector2f A, Vector2f B, Vector2f C, Vector2f P,
Vector2f p0)
{
    int cnt = 0;
    if (is_crossed(A, B, P, p0))
    {
```

```
        cnt++;
    }
    if (is_crossed(B, C, P, p0))
    {
        cnt++;
    }
    if (is_crossed(C, A, P, p0))
    {
        cnt++;
    }
    return cnt;
}
```

在此基础上，我们就可以写出判定点是否在三角形内部的函数了。

```
bool is_inside_rectangle(Vector2f A, Vector2f B, Vector2f C, Vector2f
P)
{
    Vector2f p0(1.0f, 0.0f);
    Vector2f p1(-1.0f, 0.0f);
    Vector2f p2(0.0f, 1.0f);
    Vector2f p3(0.0f, -1.0f);
    if (rectangle_cross(A, B, C, P, p0) == 1 || rectangle_cross(A, B,
C, P, p1) == 1 || rectangle_cross(A, B, C, P, p2) == 1 ||
rectangle_cross(A, B, C, P, p3) == 1)
    {
        return true;
    }
    return false;
}
```

我们对于过顶点的边界情况的处理方案使用的是我们在上一部分分析时提到的第一个方案。

二、面积判别法

根据上一部分的分析，我们只需要实现三角形面积求取的海伦公式即可：

```
double Helen_formula(Vector2f A, Vector2f B, Vector2f C)
{
    double a = (A - B).norm();
    double b = (B - C).norm();
    double c = (C - A).norm();
    double s = (a + b + c) / 2.0;
    double area = sqrt(s * (s - a) * (s - b) * (s - c));
    return area;
}
```

然后我们就可以得到一个判断的函数:

```
bool is_inside_rectangle_with_Helen_formula(Vector2f A, Vector2f B,
Vector2f C, Vector2f P)
```

```
{
    double area = Helen_formula(A, B, C);
    double area1 = Helen_formula(A, B, P);
    double area2 = Helen_formula(A, P, C);
    double area3 = Helen_formula(B, P, C);
    if (area1 + area2 + area3 - area >= -0.00001 && area1 + area2 +
area3 - area <= 0.00001)
    {
        return true;
    }
    return false;
}
```

这里存在一个问题，就是因为我们的计算机计算精度的问题，关于面积的大小比较我们无法直接用判等的方法，而必须使用作差后判断差的大小来近似认为是否相等。

三、拆解基向量判别法

这个方法具体的名字无从查证，只能暂时以这种方法的思路命名。根据我们在上一部分中的分析，我们可以得到以下代码：

```
bool is_inside_triangle_with_noname_method(Vector2f A, Vector2f B,
Vector2f C, Vector2f P)
{
    Vector2f AB = B - A;
    Vector2f AC = C - A;
    Vector2f AP = P - A;
    double u = (AB.dot(AB) * AP.dot(AC) - (AC.dot(AB) * AP.dot(AB))) /
(AC.dot(AC) * AB.dot(AB) - AB.dot(AC) * AC.dot(AB));
    double v = (AC.dot(AC) * AP.dot(AB) - (AB.dot(AC) * AP.dot(AC))) /
(AB.dot(AB) * AC.dot(AC) - AC.dot(AB) * AB.dot(AC));
    if (u >= 0 && v >= 0 && u + v <= 1)
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

与另外几种方法相比，这种方法效率最高。

实际上还有一个效率更高的方法，但是我不会证明，所以没有纳入实验。

我们可以获得三个向量 PA,PB,PC，我们将这三个向量两两组合求叉乘，即求 PA×PB、PB×PC、PC×PA 这三个叉乘的值，判断这三者是否同号即可，如果同号，则 P 在三角形内部，如果异号则在三角形外部。这种方法应该是最普遍的方法，因为只需要计算三次叉乘，且这个判别可以使用位运算来完成，效率极高。

实验中存在的问题及解决：

问题 1：射线交点判别法中，如何处理边界情况？

答 1：对于共线来说，我们只需要构造线段方程，将 P 点代入即可判断。而对于射线过顶点的情况，我们给出了两种解决方案。我们用四个射线来进行判断，这四个射线相邻两个之间的夹角均为 90°，只要这四个射线有一个与三角形的交点个数是 1，就可以确定该点在三角形内，如果四个射线的交点都不是 1，则可以确定该点不在三角形内。这种方法思路最简单，并且也容易实现，缺点是可拓展性较弱，并且计算量会随着图形的顶点个数增加而以 O(n)的趋势增长。另外还有一种处理方案，即设计射线方向时不是随意设计，而是取 PA 的方向向量，在加一个很小的偏转角度，当然这个很小的角度如果是固定值也有可能还是过这个交点，因此我们可以设置这个偏转角的角度为∠BPA 的一半。不过这个只是我的思路，我实际在实现算法的时候使用的是取四个射线的方法。

问题 2：在面积判别法中，如何处理浮点数精度问题？

答 2：因为众所周知的原因，我们在数学推导上非常理想的面积的相等关系会因为计算机内部 sqrt 函数的实现方式的原因以及无限小数的原因而导致实际的数值计算并不是完全准确的，这也使得在数学推导上等价的两个表达式的计算结果可能是不相等的。因此我们需要在浮点数的判定中加入误差量来判定两个浮点数的相等关系，在代码上体现出来的就是做差看绝对值是否在[0, η]（η是个很小的数，可能是 0.000…01，只要在 double 或者 float 的精度范围内即可）