发信人: RovingCloud (寻找当年的OI感觉), 信区: ACMICPC
标　题: 【原创】惊喜发现判断点在多边形内外的超简单算法
发信站: 逸仙时空 Yat-sen Channel (Wed Mar 28 01:27:19 2007)

今天学图形学的时候发现了一个求多边形内外的超简单算法，这个算法是源自《计算机图形学基础教程》（孙家广，清华大学出版社），在该书的48-49页，名字可称为“改进的弧长法”。该算法只需O(1)的附加空间，时间复杂度为O(n)，但系数很小；最大的优点是具有很高的精度，只需做乘法和减法，若针对整数坐标则完全没有精度问题。而且实现起来也非常简单，比转角法和射线法都要好写且不易出错。

首先从该收中摘抄一段弧长法的介绍：“弧长法要求多边形是有向多边形，一般规定沿多边形的正向，边的左侧为多边形的内侧域。以被测点为圆心作单位圆，将全部有向边向单位圆作径向投影，并计算其中单位圆上弧长的代数和。若代数和为0，则点在多边形外部；若代数和为2π则点在多边形内部；若代数和为π，则点在多边形上。”

按书上的这个介绍，其实弧长法就是转角法。但它的改进方法比较厉害：将坐标原点平移到被测点P，这个新坐标系将平面划分为4个象限，对每个多边形顶点P[i]，只考虑其所在的象限，然后按邻接顺序访问多边形的各个顶点P[i]，分析P[i]和P[i+1]，有下列三种情况：
(1) P[i+1]在P[i]的下一象限。此时弧长和加π/2；
(2) P[i+1]在P[i]的上一象限。此时弧长和减π/2；
(3) P[i+1]在Pi的相对象限。首先计算f=y[i+1]*x[i]-x[i+1]*y[i]（叉积），若f=0，则点在多边形上；若f<0，弧长和减π；若f>0，弧长和加π。

最后对算出的代数和和上述的情况一样判断即可。

实现的时候还有两点要注意，第一个是若P[i]的某个坐标为0时，一律当正号处理；第二点是若被测点和多边形的顶点重合时要特殊处理。

以上就是书上讲解的内容，其实还存在一个问题。那就是当多边形的某条边在坐标轴上而且两个顶点分别在原点的两侧时会出错。如边(3,0)-(-3,0)，按以上的处理，象限分别是第一和第二，这样会使代数和加π/2，有可能导致最后结果是被测点在多边形外。而实际上被测点是在多边形上（该边穿过该点）。

对于这点，我的处理办法是：每次算P[i]和P[i+1]时，就计算叉积和点积，判断该点是否在该边上，是则判断结束，否则继续上述过程。这样牺牲了时间，但保证了正确性。

具体实现的时候，由于只需知道当前点和上一点的象限位置，所以附加空间只需O(1)。实现的时候可以把上述的“π/2”改成1，“π”改成2，这样便可以完全使用整数进行计算。不必考虑顶点的顺序，逆时针和顺时针都可以处理，只是最后的代数和符号不同而已。整个算法编写起来非常容易。

针对以上算法，我写了一个代码，拿ZOJ 1081 Points Within进行测试，顺利Accepted。这证明该算法的正确性还是可以保障的。

以下附上我的代码：

```
// ZOJ 1081 , 改进弧长法判点在形内形外
#include <stdio.h>
#include <math.h>
const int MAX = 101;
struct point {
    int x, y;
} p[MAX];

int main()
{
    int n , m , i , sum , t1 , t2 , f , prob = 0;
    point t;
    while ( scanf( "%d" , &n ) , n )
   {
        if( prob ++ ) printf ( "\n" );
        printf ( "Problem %d:\n" , prob );
        scanf ( "%d" , &m );
        for ( i = 0; i < n; i ++ )
            scanf ( "%d%d" , &p[i].x , &p[i].y );
        p[n] = p[0];
        while ( m -- )
       {
            scanf ( "%d%d" , &t.x , &t.y );
```

```
            for ( i = 0; i <= n; i ++ ) p[i].x -= t.x , p[i].y -= t.y;    // 坐标平移
            t1 = p[0].x>=0 ? ( p[0].y>=0?0:3 ) : ( p[0].y>=0?1:2 );        // 计算象限
            for ( sum = 0 , i = 1; i <= n; i ++ ) {
                if ( !p[i].x && !p[i].y )
                    break;

             // 被测点为多边形顶点
                f = p[i].y * p[i-1].x - p[i].x * p[i-1].y;

             // 计算叉积
                if ( !f && p[i-1].x*p[i].x <= 0 && p[i-1].y*p[i].y <= 0 )
                    break;  // 点在边上

                t2 = p[i].x>=0 ? ( p[i].y>=0?0:3 ) : ( p[i].y>=0?1:2 );   // 计算象限

                if ( t2 == ( t1 + 1 ) % 4 )
                    sum += 1;
             // 情况1
                else if ( t2 == ( t1 + 3 ) % 4 )
                    sum -= 1;
             // 情况2
                else if ( t2 == ( t1 + 2 ) % 4 ) // 情况3
             {
                    if ( f > 0 ) sum += 2;
                    else sum -= 2;
             }
                t1 = t2;
          }
            if ( i<=n || sum ) printf( "Within\n" );
            else printf("Outside\n" );
            for ( i = 0; i <= n; i ++ ) {
                p[i].x += t.x , p[i].y += t.y;       // 恢复坐标
          }
       }
    }
     return 0;
}
```

--
※ 来源:. 逸仙时空 Yat-sen Channel http://bbs.zsu.edu.cn [FROM: 172.18.32.219]