文章编号:1005－0523(2005)05－0088－04

# Visual C＋＋ 库函数 PtInRegion 存在的问题及修正

李海姣

(华东交通大学 土木建筑学院 江西 南昌 330013)

**摘要**:函数 PtInregion 是微软公司提供的 Visual C＋＋语言的一个库函数,用于判断一个给定点与一多边形的位置关系.当点在多边形的内部(含边界上时)返回一个大于 0 的值;但点在多边形的外部时返回 0.但函数 PtInregion 并不能总是可靠地返回正确结果,特别是当要判断的点位于另一多边形的边界时.而这一判断广泛地用于多边形的求交、求并以及求差等计算中.这一错误将导致最终致命性的错误.对此,本文通过调试过程中跟踪函数的返回值加以论证.并且通过改进计算几何的射线法,提供了一个新的判断函数,并将两个函数的返回值进行了比较.新函数不仅适用于判断由直线段围成的多边形,而且适用于由直线段和圆弧共同围成的多边形.同时提供了两个函数的原代码.

**关 键 词**:问题;成员函数 PtInRegion;新方法
**中图分类号**:TP391.4　　**文献标识码**:A

## 1 引言

首先,在三维可视化工程造价软件的开发过程中,使用频率最高的是计算各类多边形的交、并以及差,而这类运算中最为基础的操作是判断一个指定点是否在另一多边形的内部.Visual C＋＋中提供了一个库函数 PtInregion,但是当要判断的点位于另一多边形的边界上时,该函数有时不能返回正确结果.而这一判断是多边形求交、求并以及求差等计算的基础,这一错误将导致最终致命性的错误.对此,本文通过调试过程中跟踪函数的返回值给以证明.

本文通过改进计算几何的射线法,提供了一个新的判断函数,可准确地判断点与多边形的位置关系.文中在调试状态,计算了大量实例并跟踪函数的返回值得以证明.

## 2 库函数 BOOL PtInRegion(POINT point)简介

1) 函数原型:BOOL PtInRegion( POINT point );

2) 作用:判断点与另一多边形的位置关系,点在另一多边形内部(含边界)时返回一个大于零的数,在另一多边形外部时返回零.

3) 函数使用的参数:

库函数 PtInRegion(POINT point)是 CRgn 类的成员函数,而该类成员函数的参数均为 POINT 结构或 CPoint 对象的数组(即该类函数只能在设备坐标下才能正确工作),也就是说此时 point 的每个坐标分量均采用整型数据,而用户在图形操作中一般采用逻辑坐标,为提高计算精度,建议用户在调用该函数前先根据需要扩大逻辑坐标各分量的坐标值.本文在做对比试验前,将逻辑坐标的各分量均扩大了 10 000 倍.

4) 要使用库函数 PtInRegion(POINT point)必须先调用函数 CRgn::CreatePolygonRgn 生成需要判断位置关系的另一普通多边形;由于用户绘图时均采用逻辑坐标,数据一般为浮点型或双精度型,为了调用 CRgn 类的成员函数 BOOL PtInRegion( POINT point )判断点与另一多边形位置关系,需重新构造多边形的顶点数

**收稿日期**:2005－04－21
**作者简介**:李海姣(1977－),女,河南郑州人,硕士研究生,研究方向:三维 CAD 及图形算量软件算量软件开发.

中国知网 https://www.cnki.net

组.把用户的数据类型转化为整型数组.

5) 由于 CRgn 对象的析构函数不能删除初始化后的 CRgn 对象,所以在函数末尾需要调用其基类的成员函数 DeleteObject 来删除已定义的 CRgn 对象.

在对比试验时,本文把以上操作封装为一个函数 int PInsidePoly(MyPoint p, int tag),当点在另一普通多边形内部及边界时返回 1,否则返回 0.其伪代码如下:

```
//tag=1 多边形 1;tag=2 多边形 2
int PInsidePoly(MyPoint p, int tag)
  {//1
  ……
  POINT * ppoint,pi;  //定义 POINT 型数组及点
  ……
  ppoint=new POINT[Vcount];
  ……  //重新构造顶点数组
  CRgn rgn;
  rgn.CreatePolygonRgn(ppoint,Vcount,ALTERNATE); //生成多边形区域
  int k=rgn.PtInRegion(pi);  //调用库函数
  rgn.DeleteObject();  //对象析构
  delete ppoint;
  if(k>0) return 1;  //点在多边形内部或边界
  else return 0;  //点不在多边形内部
}
```

## 3　库函数存在问题的原因分析

从上述分析中可以看出,库函数之所以存在问题,原因大致有两方面:其一是因为库函数是 CRgn 类的成员函数,其参数均为整形数据,而用户一般采用浮点型甚至双精度型,这一转化过程中不可避免地要产生误差;当应用于带圆弧边的组合多边形时,需要将圆弧边离散化,离散本身是用较短的弦去拟合相应的圆弧.而且离散时所选择的角度的大小也会直接影响计算的精度,所选择的角度过小,算法的时间和空间复杂度都将增大;选择的角度偏大,在用较长的弦拟合圆弧时误差将显著增大.这一角度难以确定,通过对比试验知,在三维可视化工程造价软件的开发中,这一角度选择为 10 时,与本文提供的算法所求的结果较为接近.但这时库函数计算的时间和空间复杂度都已经很高.

## 4　函数 int PInsideRgn(MyPoint p, int tag) 简介

中国知网　https://www.cnki.net

1) 射线法原理[1]

给定简单多边形 P 及点 pi..设简单多边形 P 的顶点序列为 $p_1, p_2, \cdots p_n$,以点 pi 为端点,向左方作射线 L,如果 L 与 P 的边界不相交,则 pi 在 P 的外部.否则,L 和 P 的边界相交,记录交点的个数并依据交点数的奇偶性判定点 pi 是否在 P 的内部.具体地说,交点数为奇数时,点 pi 在多边形内,是偶数的时,pi 在多边形外.这一方法需要首先排除一些特殊情况,如下图 ABCD 所示.在图 A 中,L 和多边形的顶点相交,这时候交点只能计算一个;在图 B 中,L 和多边形顶点的交点不应被计算;在图 C 和 D 中,L 和多边形的一条边重合,这条边应该被忽略不计.因此,在采用射线法计算射线 L 和多边形的交点时,特例处理如下:

(1) 多边形的顶点和 L 相交的情况,如果该顶点是其所属的边上纵坐标较大的顶点,则计数,否则忽略;如图 A 和图 B 所示.

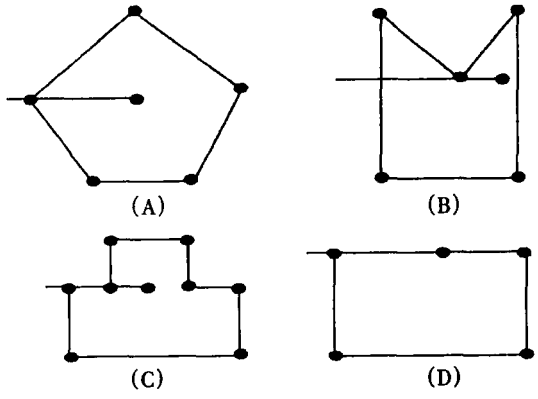


图 1

(2) 对于 pi 在多边形边上的情形,此时有多个交点,则引入两个函数分步判断,首先判断点与边的位置关系以确定点在边所在的直线上,接着,判断点是否在相应的边上.如图 C 和图 D.

在函数 PInsideRgn 中,为防止剔除图 D 所示情况,首先考虑②情况,即先判断点是否在多边形的边上,接着剔除多边形的所有水平边(即排除了图 C 所示情况),从而大大简化了计算.对组合多边形也做相同的处理.

2) 对函数的几点说明

(1) 为描述上述两类多边形,本文构造了多边形类,并采用了自定义的点结构和 Visual C++ 提供的 CArray 类数据类型来存储输入的各多边形的顶点数据,如下:

```
typedef struct
{
```

```
float x;
float y;
float z;
int r;  //标志点是否圆心,或是顺时针、逆时针圆弧
}MyPointr;
CArray <MyPointr,MyPointr>m_Poly V1;  //多边形 1 的顶点数组
CArray<MyPointr,MyPointr>m_Poly V2;  //多边形 2 的顶点数组
```

(2) 对上述特殊情况②,调用函数 PointToLRelation()判断点和线段的位置关系,该函数根据跨立法编写.在该函数的编写中,通过引入符号函数,将乘法改为减法,不仅降低了计算的时间复杂度,而且有效地截断了误差传递,从而显著地提高了计算精度.具体处理可参见本人撰写的《一个任意多边形求交的快速算法》,已被《计算机应用》录用,拟在年内正式刊出.

(3) 结果:如果点在多边形的外部,则返回 0;在边界上,返回 1;在另一多边形的内部时返回 2.

3) 射线法函数源代码

```
//
int PInsideRgn(MyPoint p, int tag)
{//1
  int JDNum=0; //记录交点的个数
  MyPoint pL; //构造以点 P 为右端点的射线
  pL.x=-(1E35); //代表负无穷大
  pL.y=p.y;
  pL.z=p.z;
  if(tag==1)
  {//2-1
    for(int i=1; i<Vcount1; i++)  {//3
  if(PointToLRelation(m_Poly V1[i-1],m_Poly V1[i],p)==0) //点在多边形的边上
  {
    float minX=(float)min(m_Poly V1[i-1].x,m_Poly V1[i].x);
    float maxX=(float)max(m_Poly V1[i-1].x,m_Poly V1[i].x);
    float minY=(float)min(m_Poly V1[i-1].y,m_Poly V1[i].y);
    float maxY=(float)max(m_Poly V1[i-1].y,m_Poly V1[i].y);
    if (p.x>=minX && p.x<=maxX && p.y>=minY &&p.y<=maxY)
    return 1;
  }
  else {//3-2
    if(fabs(m_Poly V1[i-1].y-m_Poly V1[i].y)>0.0000001) //不是水平边
    {//4
      if((PointToLRelation(p,pL,m_Poly V1[i-1])==0 &&
        m_Poly V1[i-1].y>m_Poly V1[i].y)||
        (PointToLRelation(p,pL,m_Poly V1[i])==0 &&
        m_PolygonV1[i].y>m_Poly V1[i-1].y))
          JDNum++;
      else {//5
        int n1=SameSide2(p,pL,m_Poly V1[i-1],m_Poly V1[i]);
        if(n1==41 || n1==42) {
          int n2=SameSide2(m_Poly V1[i-1],m_Poly V1[i],p,pL);
          if(n1==41 && n2==41||(n1==41 && n2==42)||(n1==42 && n2==41) ||
            (n1==42 && n2==42))
            JDNum++; }}//5
    }//4  }//3-2
  }//3
  if(JDNum%2==1) return 2; //点在多边形内部
  else return 0; //点在多边形外部 (JDNum%2==0)
}//2-1
else //tag==2
{//2-2
  for(int i=1; i<Vcount2; i++)  {//3
    if (PointToLRelation(m_PolygonV2[i-1],m_PolygonV2[i],p)==0)} //点在多边形的边上
    float minX, minY, maxX, maxY;
    minX=(float)min(m_Poly V2[i-1].x,m_Poly V2[i].x);
    maxX=(float)max(m_Poly V2[i-1].x,m_Poly V2[i].x);
    minY=(float)min(m_Poly V2[i-1].y,m_Poly V2[i].y);
    maxY=(float)max(m_Poly V2[i-1].y,m_Poly V2[i].y);
    if (p.x>=minX && p.x<=maxX && p.y>=minY &&p.y<=maxY)
    return 1;  }
  else  {//3-2
    if(fabs(m-Poly V2[i-1].y-m_PolygonV2[i].y)>0.0000001) //不是水平边
    {//4
```

中国知网 https://www.cnki.net

```
    if((PointToLRelation(p,pL,m_Poly V2[i-1])==0) &&
m_Poly V2[i-1].x<p.x)  {
      if(m_Poly V2[i-1].y>m_Poly V2[i].y)
        JDNum++;  }
      else if(PointToLRelation(p,pL,m_Poly V2[i])==0 &&
m_Poly V2[i].x<p.x )  {
        if(m_Poly V2[i].y>m_Poly V2[i-1].y)
          JDNum++;  }
      else
        int n1=SameSide2(p,pL,m_Poly V2[i-1],m_Poly
V2[i]);
        int n2=SameSide2(m_Poly V2[i-1],m_Poly V2
[i],p,pL);
        if((n1==41 && n2==41)||(n1==41 && n2==
=42)||(n1==42 && n2==41) ||
          (n1==42 && n2==42))
          JDNum++;  }
    }//4  }//3-2
  }//3
  if(JDNum%2==1) return 2; //点在多边形内部
  else return 0; //点在多边形外部
  }//2-2
}
```

## 5　函数复杂度分析

从大量算例得知,库函数不能直接应用于组合多边形的几何运算,要应用库函数并且达到与本文提供的算法相同的精度,必须将圆弧边离散化,且需要离散为圆心角为 10 的弦(代替相应的弧).试想如果组合多边形中带一圆心角为 450 的弧,就要将一段弧转化为 45 条直线段边来处理,不仅大大增加了运算量,而且也提高了对计算过程中存储空间的要求,即提高了算法的时间和空间复杂度.而新算法仍将圆弧作为一个对象处理,不必离散,并且由于跨立法、误差截断技术以及包围盒排除法的采用,大大降低了算法的时间以及空间复杂度.

## 6　结　论

在三维可视化工程造价软件的开发过程中,使用频率最高的是计算各类多边形的交、并以及差,而这类运算中最为基础的操作是判断一点是否在另一多边形的内部.通过本文提供的函数可以大大降低多边形求交、并以及差的运算的时间和空间复杂度.同时赋予了开发者更大的灵活度,如可以根据实际需要构造由其它形式的曲线所围成的多边形,只要依据本文算法将函数加以扩充,就可以处理这类图形对象的判断.

**参考文献:**

[1] 周培德. 计算几何[M]. 北京:清华大学出版社,2000.

# The Problem of Memberfunction PtInRegion in Visual C++ Language and Amending

**LI Hai-jiao**

(School of Civil Engineering and Architecture, East China Jiaotong University, Nanchang 330013, China)

**Abstract**: The problem in PtInRegion, which is one of the memberfunctions in Visual C++ Language provided by Microsoft Company, is presented in this paper. This function is used to caculate the relation between a point and a polygon. The problem is that PtInRegion function mentioned above can not always compute the correct result, especially when the point which will be judged is exactly on the edge of the other polygon. And this kind of judging is widely used in computing two polygons intersection, union and difference et. So it has great effect on the finally result. In this paper, the problem of function mentioned above is distinctly proved through tracing the whole debuging process and the function values. At the same time a new method and the compare between two functions are provided. The new method is based on the computational geometry. And it can be used not only in the orginal polygon, but also in the polygon including arc edge. The codes of two functions are offered in the paper.

**Key words**: problem; memberfunction PtInRegion, new method

中国知网 https://www.cnki.net