任意多边形窗口的圆裁剪算法

杭后俊,孙丽萍
(安徽师范大学 数学计算机科学学院,安徽 芜湖 241000)

摘 要:圆的裁剪广泛应用于诸如计算机图形学、二维计算机动画以及机器人运动学等领域。讨论了圆关于任意多边形窗口的一个裁剪算法,按逆时针方向依次求出多边形裁剪窗口的每条边与圆的交点并且保证交点正确排序,对于交点序列中的任意两相邻的交点,采用"中点检测法"来判定以它们为端点的圆弧与裁剪窗口的位置关系,最后给出完整的裁剪算法。实现结果表明,不论从效率还是稳定性方面都取得了比较理想的效果。

关键词:裁剪窗口;裁剪算法;求交算法
中图分类号:TP391.41 **文献标识码**:A **文章编号**:1673-629X(2009)05-0235-03

An Algorithm for Circle Clipping Against Polygon Window

HANG Hou-jun, SUN Li-ping
(School of Mathematics and Computer Science, Anhui Normal University, Wuhu 241000, China)

Abstract: Circle clipping is applied widely computer graphics, such as two dimension computer animation and robot kinematics etc. Discusses a circle clipping algorithm against the polygon window in detail. The intersection points of the circle and each side of the clipping window are calculated and sorted correctly. Regarding two neighbor intersection points in sequence, "the middle point detecting method" is used to judge the position relations of circular arc and the clipping window. Finally, the whole clipping algorithm is gained. The result expresses that the algorithm is stable and efficient.

Key words: clipping window; clipping algorithm; intersection algorithm

0 引 言

确定图形中哪些部分落在显示区之内,哪些落在显示区之外,以便只显示落在显示区内的那部分图形,这个选择过程称为裁剪。图形的裁剪算法是光栅图形学的重要内容之一,在各种有关计算机图形学的专著中都对图形的裁剪进行专门的讨论[1,2]。对于广大用户来说,不论是使用图形软件库(如 OpenGL 等)开发绘图软件,还是直接使用各种图形软件进行图形对象的绘制与编辑,图形的裁剪是使用频率非常高的操作[3]。

多年来,许多学者专门撰写论文对有关基本图形的裁剪算法进行探讨[4~8]。对圆裁剪的讨论更多的侧重于矩形裁剪窗口[9,10],任意多边形窗口的圆裁剪算法讨论得很少。而作为图形裁剪的重要部分,圆的裁剪广泛应用于诸如计算机图形学、二维计算机动画以及机器人运动学等领域。例如,经常需要对两个或多个实体间进行碰撞、检测等。特别是在二维计算机动画中,用圆来表示二维实体的某些局部等等。人们往往希望先定义一个视窗(Viewport),然后在该视窗内显示图形对象。通过鼠标响应以弧擦除的方式来进行窗口的裁剪。因此,讨论圆关于一个多边形窗口的裁剪算法就显得非常有意义。

文中讨论了任意多边形窗口的圆裁剪算法问题,给出了一个快速裁剪算法,并对该算法进行了实现。

1 直线段与圆的求交算法

圆相对于多边形窗口裁剪的关键是要求出圆和窗口每条边的交点。因此,直线段与圆的求交算法是文中后续内容的基础。

如图 1,已知线段 P_1P_2,端点坐标为 $P(x_1,y_1)$,$P(x_2,y_2)$,则其方程为:

$$ax + by + c = 0 \tag{1}$$

其中 $a = y_0 - y_1, b = x_1 - x_0, c = x_0y_1 - x_1y_0$。

圆的方程为:

$$(x - x_0)^2 + (y - y_0)^2 = r^2 \tag{2}$$

收稿日期:2008-09-16
基金项目:安徽省自然科学基金(2006kj076B);安徽师范大学青年基金(2008xqn47)
作者简介:杭后俊(1965-),男,安徽无为人,副教授,研究方向为计算机图形学,CAGD等。

由点到直线的距离公式可知，圆心(x_0, y_0)到P_1P_2的距离d为

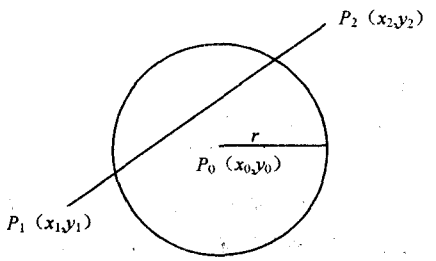

图 1 线段与圆的求交

$$d = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}}$$

若$d > r$，则表明线段与圆没有交点，否则，讨论交点求法。

容易求出，P_1P_2的参数方程为

$$\begin{cases} x = x_1 + (x_2 - x_1)t \\ y = y_1 + (y_2 - y_1)t \end{cases} \quad t \in [0,1] \qquad (3)$$

将(3)代入(2)中，化简整理得

$$At^2 + Bt + C = 0 \qquad (4)$$

其中

$$A = x_1^2 + x_2^2 + y_1^2 + y_2^2 - 2x_1x_2 - 2y_1y_2$$

$$B = 2(x_0x_1 + x_1x_2 + y_0y_1 + y_1y_2 - x_0x_2 - y_0y_2 - x_1^2 - y_1^2)$$

$$C = x_0^2 + x_1^2 + y_0^2 + y_1^2 - 2x_0x_1 - 2y_0y_1 - r^2$$

令$\triangle = B^2 - 4AC$

若$\triangle = 0$，则只有一个解：$t_1 = t_2 = \frac{-B}{2*A}$

若$\triangle > 0$，则有两个解：$t_1 = \frac{-B + \sqrt{\triangle}}{2A}$，$t_2 = \frac{-B - \sqrt{\triangle}}{2A}$

如果$0 \leqslant t_i \leqslant 1$，则将$t_i$代入式(3)所求出的点是线段与圆的交点，否则不是交点。

通过以上的讨论，给出直线段与圆的求交算法：

算法 1：直线段与圆的求交。

输入：线段P_1P_2和圆。

Step1：初始化：求a, b, c, d, A, B, C等。

Step2：求圆心到线段的距离d。若$d > r$，则无交点，转 Step6；否则转 Step3。

Step3：求解方程(4)。如果$\triangle = 0$，求出唯一实根t，转 Step4；否则，求出实根t_1, t_2，转 Step5。

Step4：若$t \in [0,1]$，则代入式(3)求出交点并保存到交点表中，转 Step6；否则，直接转 Step6。

Step5：如果$t_1 > t_2$，则交换它们的值，使$t_1 < t_2$，若$t_i \in [0,1]$，则代入式(3)求出交点并保存到交点表中。

Step6：结束。

完整的求交伪代码如下：

```
//将交点保存到交点表中
float intersec_point[8][2];
int number=0;
void point_list(t,x1,y1,x2,y2)
float t,x1,y1,x2,y2;
{float x,y;
x=x1+t*(x2-x1);
y=y1+t*(y2-y1);
intersec_point[number][0]=x;
intersec_point[number][1]=y;
number+=1;
}
//直线段与圆的求交算法
void LineCircle_intersection(x1,y1,x2,y2,x0,y0,r)
float x1,y1,x2,y2,x0,y0,r;
{
float a,b,c,a1,b1,c1,d,delta,t1,t2,t;
a=y0-y1;b=x1-x0;c=x0*y1-x1*y0;
d=abs(a*x0+b*y0+c)/sqrt(a*a+b*b);
a1=x1*x1+x2*x2+y1*y1+y2*y2-2x1*x2-2y1*
y2;
b1=2*(x0*x1+x1*x2+y0*y1+y1*y2-x0*x2-y0
*y2-x1*x1-y1*y1);
c1=x0*x0+x1*x1*+y0*y0+y1*y1-2*x0*x1-
2y0*y1-r*r;
delta=b1*b1-4*a1*c1;
if(d<=r)
{
if(delta<1e-6)
{
t=-b1/2*a1;
if(t>=0&&t<=1)
//将交点保存到交点表中
point_list(t,x1,y1,x2,y2);
}
else{
t1=(-b1+sqrt(delta))/2*a1;
t2=(-b1-sqrt(delta))/2*a1;
if(t1>t2){t=t1;t1=t2;t2=t;}
if(t1>=0&&t1<=1)
point_list(t1,x1,y1,x2,y2);
if(t2>=0&&t2<=1)
point_list(t2,x1,y1,x2,y2);
}
}
}
```

2　主要内容及算法

经过以上的讨论,下面来研究圆的裁剪问题。

给定任意多边形裁剪窗口,圆的半径为 r,圆心在 (x_0,y_0),如图 2。处理思想是,按逆时针方向依次求裁剪窗口的每条边与被裁剪圆的交点。求 $P_iP_{i+1}(i=1,2,\cdots,m$,令 $P_{m+1}\equiv P_1)$ 与圆的交点可用算法 1 求出。

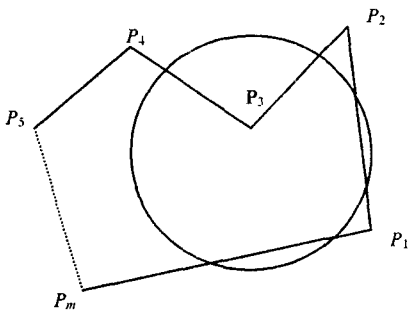

图 2　圆与裁剪窗口示

首先,如果实际交点数 number = 0,则说明被裁剪圆要么落在多边形窗口内,要么落在多边形窗口外。

1) 若被裁剪圆圆心在多边形窗口内(可用交点计数检验法),则绘制整个圆。

2) 若被裁剪圆圆心在多边形窗口外,整个圆被裁剪。

以下的讨论均假设 number > 0。

假设窗口的所有边与圆的交点参数序列为 $t_1,t_2,\cdots,t_k$(k 最多取到 2m,实际值可在编程时得到),其中 $t_1,t_2,\cdots,t_k$ 的顺序可由下面的方法确定:

① 若 t_i,t_j 是属于同一条边 P_iP_{i+1} 与圆的交点参数,则较小的一个在较大的前面(在直线段与圆的求交算法中已经这样做了),即 $t_i<t_j$。

② 若 t_i,t_j 是两条不同边与圆的交点,则按逆时针方向进行排序,若边在前面的,那么 t 值就在前面。

设参数 t_i 对应的交点为 Q_i,如果参数序列已经按上述方法正确排序,那么交点 $Q_1,Q_2,\cdots,Q_k$ 就是正确的顺序。

下面讨论判断两相邻交点之间的圆弧是否被裁剪的方法。

设 $Q_1,Q_2,\cdots,Q_k$ 是多边形窗口与被裁剪圆的所有交点,并已按上述方法正确排序。对两相邻的交点 $Q_i,Q_{i+1},(0\leqslant i\leqslant k$,令 $Q_{k+1}\equiv Q_1)$ 之间的圆弧是否画出,可采用中点检测法。请注意,圆弧是有方向的,圆弧 Q_iQ_{i+1} 表示以 Q_i 为起点,以 Q_{i+1} 为终点按逆时针方向的圆弧;圆弧 $Q_{i+1}Q_i$ 表示以 Q_{i+1} 为起点,以 Q_i 为终点按逆时针方向的圆弧。

如图 3,将圆的方程写成参数形式:

$$\begin{cases}x=x_0+r\cos\alpha\\ y=y_0+r\sin\alpha\end{cases}\quad\alpha\in(0,2\pi)\qquad(5)$$

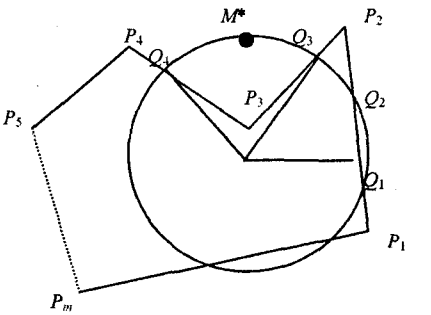

图 3　圆弧的中点检测法

将 Q_i 点坐标代入上述参数方程可求出 α_i,将 Q_{i+1} 点坐标代入,求出 α_{i+1}。为了正确求出圆弧 Q_iQ_{i+1} 的中点,分以下三种情况进行讨论。

1) 若 $\alpha_i<\alpha_{i+1}$,则 $\alpha=\frac{1}{2}(\alpha_i+\alpha_{i+1})$。

2) 若 $\alpha_i\geqslant\alpha_{i+1}$,且 $\alpha_i+\alpha_{i+1}<\pi$,则 $\alpha=\frac{1}{2}(\alpha_i+\alpha_{i+1})+\pi$。

3) 若 $\alpha_i\geqslant\alpha_{i+1}$,且 $\alpha_i+\alpha_{i+1}>\pi$,则 $\alpha=\frac{1}{2}(\alpha_i+\alpha_{i+1})-\pi$。

将 α 代入圆的参数方程(4)求得点 $M^*(x^*,y^*)$,点 $M^*(x^*,y^*)$ 与裁剪窗口的位置关系判定可采用"交点计数检验法"。

(1) 若 M^* 在窗口外,则圆弧 Q_iQ_{i+1} 被裁剪。为了保证裁剪后的图形仍然封闭,应绘制出以 Q_i 为起点,以 Q_{i+1} 为终点的线段。

(2) 若 M^* 在窗口内,圆弧 Q_iQ_{i+1} 被保留。

通过以上的讨论,给出圆关于一个多边形窗口的裁剪算法。

算法 2:圆关于多边形窗口的裁剪。

输入:多边形窗口 $P_1P_2\cdots P_m$,被裁剪圆 $O(x_0,y_0,r)$。

Step1:按照逆时针方向对多边形窗口的每条边 $P_iP_{i+1}(i=1,2,\cdots,m$,令 $P_{m+1}\equiv P_1)$ 依次调用直线段与圆的求交算法求出交点(如果有的话)。并将交点存入数组 intersec_point。如果实际交点数 number=0,转 Step2;否则,转 Step3。

Step2:若被裁剪圆圆心在多边形窗口内,则绘制整个圆;否则,整个圆被裁剪。转 Step5。

Step3:依次考察两相邻的交点 $Q_i\equiv$ intersec_point[i] 和 $Q_{i+1}\equiv$ intersec_point [$i+1$] ($0\leqslant i<$ number, intersec_point[number] = intersec_point[0]) 所组成有向圆弧 Q_iQ_{i+1},求出该段圆弧的中点 M^*。

Step4:采用交点计数检验法,若 M^* 在窗口外,则

(下转第 241 页)

容"按钮,就会通过 P2P 网络下载新的更新文件。

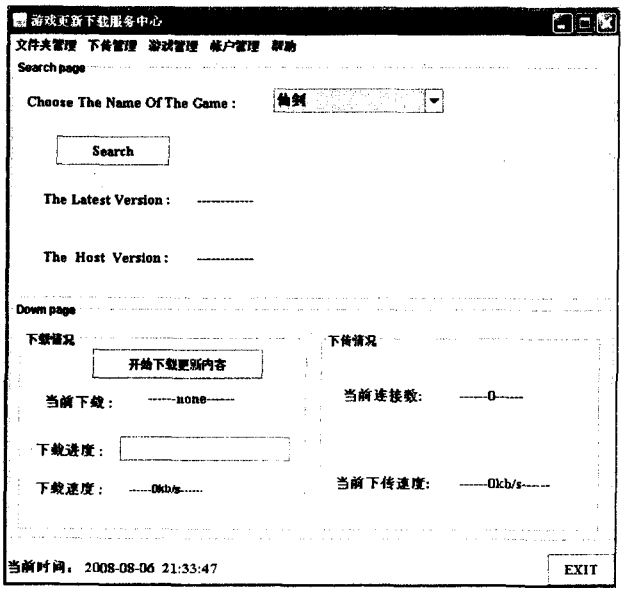

图 2　游戏更新下载服务中心主界面

4　结束语

JXTA 作为专为 P2P 网络构建的平台和协议，为开发 P2P 上层应用程序提供了统一、便利、通用的底层平台，具有很强的独立性和可扩展性。JXTA 模块作为 JXTA 网络服务的核心组件，为创建 P2P 应用提供了可伸缩性、健壮性和可扩展性。通过这种机制开发者可以创建可协同操作的、可用性高且健壮的服务。

参考文献:

[1] Oaks S, Rraversat B, Gong L. JXTA 技术手册[M]. 技　桥译. 北京:清华大学出版社,2004:9－15.

[2] Osais Y, Abdala S. A Multilayer Peer－to－Peer Framework for Distributed Synchronous Collaboration[J]. Internet Computing. IEEE,2006,10(6):33－41.

[3] JXSE 2.5 Programmers Guide[M]. [s. l.]:2002－2007 Sun Microsystems, Inc,2007.

[4] 陈　锋,罗逢吉,俊　浩. 基于 JXTA 的 P2P 文件共享系统的实现研究[J]. 计算机科学,2007,34(12):126－150.

[5] 姜兆华,杨　斌. 基于 JXTA 和 P2P 的资源发布系统研究[J]. 计算机与信息技术,2008(z1):60－62.

[6] 杨文俊. P2P 网络系统中节点自组织管理机制[J]. 计算机技术与发展,2006,16(7):57－60.

[7] 邢长明,刘方爱. 基于 P2P 的网格资源发现机制研究[J]. 计算机技术与发展,2006,16(8):21－23.

(上接第 237 页)

圆弧被裁剪,仅绘制出以 intersec_ point[i] 为起点,以 intersec_ point[$i+1$] 为终点的线段;若 M^* 在窗口内,则画出该段圆弧。

Step5:结束。

对本算法进行了实现,结果表明,不论从效率还是稳定性方面都取得了比较理想的效果。

3　结束语

讨论了圆关于任意多边形窗口的一个裁剪算法。处理思想是,按逆时针方向依次求裁剪窗口的每条边与被裁剪圆的交点。如果实际交点数 number = 0,若被裁剪圆圆心在多边形窗口内,则绘制整个圆;否则,整个圆被裁剪。在其他情况下,保证交点表中的交点正确排序,从而组成了交点序列,依次考察两相邻的交点所组成有向圆弧,求出该段圆弧的中点。采用交点计数检验法,若中点在窗口外,则圆弧被裁剪;若中点在窗口内,则画出该段圆弧。最后给出了一个完整的裁剪算法。从实现结果来看,用该算法能够取得较好的效果。

另外,在算法中,可以先分别求出多边形窗口的矩形包围盒和被裁剪圆的外切正方形包围盒,快速排除整个圆在窗口外的情形,提高裁剪效率。

参考文献:

[1] 孙家广. 计算机图形学[M]. 北京:清华大学出版社,1998.

[2] 孙家广,胡事民. 计算机图形学基础教程[M]. 北京:清华大学出版社,2005.

[3] Hearn D. Computer Graphics with OpenGL[M]. 3rd Edition. 北京:电子工业出版社,2004.

[4] WU X, Rokne J. Double－Step incremental generation of lines and circles[J]. Computer Vision, Graphics, and Image Processing,1987,37(3):331－334.

[5] Liang Y D, Brasky B A. A New Concept and Method for Line Clipping[J]. ACM Transactions on Graphics,1984,3(1):1－22.

[6] Nicholl T M, Lee D T, Nicholl R A. An Efficient New Algorithm for 2D Line Clipping: Its development and analysis[J]. Computer Graphics,1987,21(4):253－262.

[7] 韩明峰. 基于一般多边形窗口的线裁剪[J]. 微机发展(现更名:计算机技术与发展),1999,9(2):49－50.

[8] 范延军,孙燮华. 一种基于几何关系编码的高效凸多边形线裁剪算法[J]. 计算机应用与软件,2007,24(2):148－150.

[9] Sun Yan, Tang Di. An Algorithm for Curve Clipping Against the Rectangular Window[J]. Computer Applications and Software,2003,20(5):35－36.

[10] 刘勇奎. 计算机图形学的基础算法[M]. 北京:科学出版社,2001:62－70.