一种改进 AABB 包围盒的碰撞检测算法

王立文,刘璧瑶,韩俊伟
WANG Li-wen,LIU Bi-yao,HAN Jun-wei

中国民航大学 中国民航机场地面特种设备研究基地,天津 300300
Civil Aviation University of China,Tianjin 300300,China

WANG Li-wen,LIU Bi-yao,HAN Jun-wei.Improvement AABB surrounds examination calculate way of collision box. Computer Engineering and Applications,2007,43(33):234-236.

Abstract: The text discussed and compared the principle and effectiveness of Axis-Aligned Bounding Boxes (AABB) method, Oriented Bounding Box (OBB) method and Discrete Orientation Polyhedral (K-DOP) method in detail,and improved AABB method.An idea of utilizing simplifying surrounding edge node bounding box to realize colliding was proposed for the first time. And the feasibility was demonstrated by some preliminary tests.Not only has improved the speed measured of collision detection, but also can get more detailed collision and measure information conveniently,met the need of colliding and responding dealing. This truly made visual system of flight simulator can measure collision of fictitious object in real-time.

Key words: collision detection;bounding box;flight simulator;algorithm

摘 要:详细分析比较基于包围盒的碰撞检测算法中的轴向包围盒法、方向包围盒法、离散方向多面体法的检测原理和检测效率,并改进了轴向包围盒碰撞检测算法,提出利用简化包围盒边缘节点实现碰撞检测的新设想,其可行性已被初步试验证实。不仅显著提高了碰撞检测的速度,并且可以便捷地得到更为详细的碰撞检测信息,满足了进一步进行碰撞响应处理的需要。使飞行模拟机的视景系统能够实时、准确地检测出虚拟物体间的碰撞。

关键词:碰撞检测;包围盒;飞行模拟机;算法
文章编号:1002-8331(2007)33-0234-03 **文献标识码:**A **中图分类号:**TP301.6;TP27

视景系统作为飞行模拟机的一个子系统,在模拟起飞和着陆过程中,可以提供机场、跑道及机场周围诸如候机楼、山川、田野等画面景象, 由于飞行员飞行时视觉信息占总信息的70%,因此视景系统内容的丰富度、逼真度和清晰度都会影响飞行模拟机的质量和飞行训练的效果。而碰撞检测是增强系统现实感的基础和关键之一,碰撞检测就是检测虚拟场景中不同对象之间是否发生了碰撞,如飞机与候机楼、山川、地面的碰撞,精确的碰撞检测对提高仿真的真实性、可信性,增强虚拟环境的沉浸感有着至关重要的作用。

目前,国内外有关包围盒碰撞检测算法,有基于包围盒的碰撞检测算法,基于距离计算的碰撞检测算法,基于维诺图的碰撞检测算法等,其中层次包围盒算法适用于复杂视景环境中的碰撞检测。在研究比较了 AABB 算法,OBB 算法和 K-DOP 算法优缺点的基础上改进了轴向算法,基本思想是舍弃 AABB 树的叶节点,即无需进行包围体间的相交测试,直接进行基本图元的测试,使总体相交测试的次数减少,提高算法效率;通过在飞行模拟机上进行实际运行,考察了视景碰撞检测的响应效果。

1 碰撞检测原理

碰撞检测系统的输入模型是构成几何对象的基本几何元素(通常是三角形)的集合,其任务是确定在某一时刻两个模型是否发生干涉,即它们的交集是否不为空,如发生碰撞,还需确定碰撞部位(参与碰撞的基本几何元素)。从几何上讲,碰撞检测表现为两个多面体的求交测试问题;按对象所处的空间可分为二维平面碰撞检测和三维空间碰撞检测。平面碰撞检测相对简单一些,已经有较为成熟的检测算法,而三维空间碰撞检测则要复杂的多[1,2]。按照是否考虑时间参数,碰撞检测又可分为连续碰撞检测和离散碰撞检测。

三维空间 R 用三维几何坐标系统 F_W 表示,其中有 N 个运动模型, 它们的空间位置和姿态随着时间而改变,F_i 表示第 I 个模型所占的空间。F_W 随着时间的变化形成四维坐标系统 C_W,模型 F_i 沿着一定的轨迹运动形成四维坐标系统 C_i 碰撞检测就是判断下式是否成立。

$$C_1 \cap C_2 \cap C_3 \cdots \cap C_n = \varnothing$$

碰撞检测算法要遍历所有的基本几何元素, 是最基本、也是速度最慢的碰撞检测算法。实际系统中为了提高检测速度,总体上分为空间分解法和层次包围盒法两大类。空间分解法是将虚拟空间分解为体积相等的小单元格,只对占据同一单元格或相邻单元格的几何对象进行相交测试。典型的空间分解法有八叉树法和二叉空间剖分法。空间分解法由于存储量大及灵活性不好,使用不如层次包围盒方法广泛。层次包围盒方法是利用体积略大而形状简单的包围盒把复杂的几何对象包裹起来,

作者简介:王立文(1962-),男,满族,教授,博导,民航特聘专家,研究方向为航空地面特种设备、六自由度并联机器人、智能控制系统与过程控制系统;刘璧瑶(1981-),女,汉族,硕士,研究方向为飞行模拟机视景系统开发研究。

在进行碰撞检测时首先进行包围盒之间的相交测试;如果包围盒相交,再进行几何对象之间精确的碰撞检测。

2　碰撞检测算法

2.1　轴向包围盒(AABB)检测算法

AABB 在碰撞检测的研究历史中使用得最久最广,一个给定对象的 AABB 被定义为包含该对象且边平行于坐标轴的最小的正六面体。因此,描述一个 AABB,仅需六个标量。在构造 AABB 时,需沿着物体局部坐标系的轴向(X,Y,Z)来构造,所以所有的 AABB 具有一致的方向。基于 AABB 的碰撞检测系统有 I-COLLIDE,SOLID 等。

给定对象,飞机的 AABB 的碰撞检测计算十分简单,只需分别计算飞机简化模型中各个元素的顶点的 x 坐标、y 坐标和 z 坐标的最大值和最小值即可,因此计算飞机模型的 AABB 只需 6 m 次比较运算,存储 AABB 只需 6 个浮点数。但是,AABB 的紧密性相对较差,尤其是对于沿斜对角方向放置的瘦长形对象,用 AABB 将留下很大的边角空隙,从而导致大量冗余的包围盒相交测试。

两个 AABB 间的相交测试,可根据两个 AABB 相交当且仅当它们在三个坐标轴上的投影区间均相交。通过投影,可以将三维求交问题转化为一维求交问题。考察两个包围盒分别向三个坐标轴的投影的重叠情况,即可得出测试结果,当对象发生旋转后,需要对 AABB 进行更新。根据定义 AABB 的 6 个最大最小值的组合,可以得到 AABB 的 8 个顶点,对这 8 个顶点进行相应的旋转,并根据旋转后的顶点计算新的 AABB。当对象发生变形后,可以重新计算 AABB 树中发生变形了的叶结点的 AABB,然后自下向上由子结点的 AABB 合成父结点的 AABB,利用这种方法,只需要 6 次比较运算即完成一个结点的更新,其效率远远高于重新构造包围盒树。

2.2　方向包围盒(OBB)检测算法

一个物体的 OBB 被定义为包含该对象且相对于坐标轴方向任意的正六面体。在三维空间中 *OBB* 可用中点 C、半边长 r_1、r_2、r_3 和相互垂直的单位向量 $\boldsymbol{v}^1$、$\boldsymbol{v}^2$、$\boldsymbol{v}^3$ 共 14 个参数表示:

$$R=\{C+ar_1v^1+br_2v^2+cr_3v^3 | a,b,c\in[-1,1]\}$$

OBB 间的相交测试基于分离轴理论。若两个 OBB 在一条轴线上(不一定是坐标轴上)的投影不重叠,则这条轴称为分离轴。若一对 *OBB* 间存在一条分离轴,则可以判定这两个 *OBB* 不相交。对任何两个不相交的凸三维多面体,其分离轴要么与任一多面体的某一个面,要么同时垂直于每个多面体的某一条边。因此,对一对 *OBB*,只需测试 15 条可能是分离轴的轴(每个 *OBB* 的 3 个面方向再加上每个 *OBB* 的 3 个边方面的两两组合),只要找到一条这样的分离轴,就可以判定这两个 OBB 是不相交的,如果这 15 条轴都不能将这两个 *OBB* 分离,则它们是相交的。两个 *OBB* 的相交测试最多需要 15 次比较运算、60 次加减运算、81 次乘法运算和 24 次绝对值运算。

2.3　离散多面体(k-DOP)检测算法

k-DOP 的概念最早由 Kay 和 Kajiya[9]提出,他们在分析了以往采用的层次包围盒进行光线跟踪计算的缺点后,提出了一个高效的场景层次结构应满足的条件。综合起来就是各层次包围盒都应尽可能紧密地包裹其中所含有的景物。作为叶节点,景物自身即是最紧的包围盒,但由于包围盒的选取还要求光线与包围盒的求交测试尽可能简单,因此应选取形状比较简单的球体、圆柱体、长方体等作为包围盒。但这些形状简单的包围盒具有包裹景物不紧的缺点,Kay 和 Kajiya 提出了根据景物的实际形状选取若干组不同方向的平行平面对包裹一个景物或一组景物的层次包围盒技术。

3　改进算法的检测结果及分析

(1)AABB 树由 $2\times N-1$ 个节点组成,其中 N 是几何体中基本图元(通常是三角形)的数目。完全 AABB 树有 N 个叶节点和 $N-1$ 个内部节点,每个叶节点包含一个指向基本图元的指针和包围基本图元的包围体。在进行碰撞检测时,遇到测试两个叶节点的情况,需要首先进行包围体间(BV/BV)的相交测试。如果包围体相交测试成立,则可进行基本图元的相交测试,确定精确位置。

改进算法的基本思想:是舍弃 AABB 树的叶节点,即无需要进行包围体间的相交测试,直接进行基本图元间的测试。因为如果基本图元相交,则包围体间的相交测试就可以省略;如果基本图元不相交,则改进所付出的代价就是测试包围体比测试基本图元节省时间。例如,包围体间的测试结果是相交,但由于精确的基本图元与包围体间的测试结果表明没有相交,这就意味着相交测试会更早提前结束,从而提高算法的效率。一般情况下,正常显示所需的刷新率在 25 帧/s 以上,以一个由 7 万多个三角形、132 个物体组成的场景为例,稳定的帧频在 19.0 ms 至 20.3 ms 之间。经过多组试验对比,这种在程序中动态读取场景模型信息的方式,通常不会超过 2 ms,如曲线图 1 所示。对三种算法进行测试,为了屏蔽地形及场景等其他因素对试验结果的影响,场景中在三处不同位置放置运动体(飞机:三角形数,3072),结果如图 2 曲线所示。由曲线数据分析可知:改进的算法进行优化处理,所以无论是否发生碰撞,所需的检测时间几乎相同,因而非常耗费系统资源。

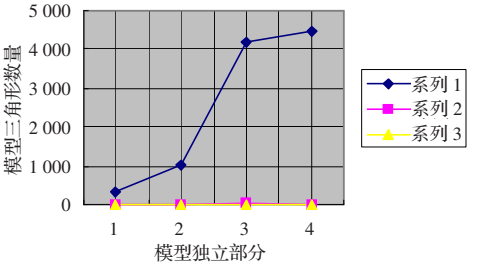

图 1　动态读取模型试验结果曲线图

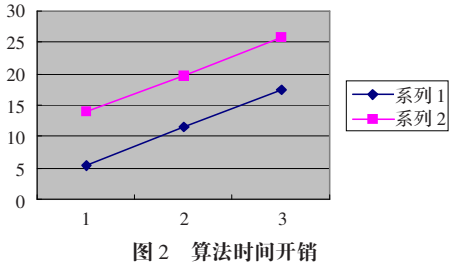

图 2　算法时间开销

(2)VEGA 开发环境下是以 OpenFlight 文件加载模型的。因此可以根据已经公开的 Flt 文件格式,通过直接读取 Flt 文件获得几何模型信息,进而构建包围树。

①通过调用 vgGetObjpfNode(VgObj*obj)获得物体的 Pfnode 句柄。

②遍历 Pfnode 子树,利用 pflsOfType(pfnode,pfGetGeodeClassType())得到 pfGeode 节点。

③在 Performer 中,模型的点、面存储于 pfGeoSet 节点。调用 pfGeoSet()函数可以得到 pfGeoSet 句柄。

④调用函数 pfGetSetAttrLists()获得模型的点信息列表。通过 pfGetSetPrimType()确定列表信息的存储顺序。可以使用 pfGSetAttr()修改几何模型。

基本代码如下:

```
Void numGeo(pfNode * node)
{
pfNode * m_node;
pfGeoSet * GeoSetnode;
pfVec3 * cords;
ushort * cords;
childnum=pfGetNumChildren(node);
for(i=0;i<childnum;i++)
{
 m_node=pfGetChild(node,i);
 if(pfIsOType(m_node,pfGetGeodeClassType()))
 {
 Gchildnum=pfGetNumGSets(m_node);
   //////////提取 Pflist 信息//////////
 For(j=0;j<gchildnum;j++)
 {
 Geosetnode=pfGetSet(m_node,j);
 pfQueryGSet(Geosetnode,PFQGSET_NUM_VERTS,&mcount);
 pfGetSetAttrLists(tmpGnode,PFGS_COORD3,(VOID **)
                   &coords,&coordsi);
 VerBuffer.Add(cords,pfGetSetPrimType(Geosetnode));
 }
 }
 else
          numGeo(m_node);
     }
}
```

(3)试验结果及分析

试验运行环境:CPU 主频 AMD2400+,内存 1 GB DDR,显卡 128 M,VEGA 版本为 3.7, 三台计算机和边缘融合计算机,采用三通道视景模式。

在开发平台 VEGA 上,实验碰撞检测及响应,观察视角选定在通道外侧,图 3 所示,飞机坠入地面的测试效果,飞机的基本图元与跑道平面发生碰撞延迟,机头已坠入地面,还没有来得及发生响应;而图 4,飞机没有坠入草地平面,及时产生碰撞响应的效应。若将观察视角选定在通道内部,同样可以感受到飞机的碰撞延迟和及时碰撞响应效果。

图 3 飞机延迟撞地面效果图

图 4 飞机撞地面及时响应效果图

4 结束语

碰撞检测是视景仿真重要的组成部分, 由于实时性的要求,在虚拟场景中,一般是用相对简单的包围盒包裹虚拟对象,并用包围盒代替虚拟对象进行碰撞检测。

本文分析比较了 AABB,OBB 和 K-DOPs 三种包围盒碰撞检测算法的优缺点,并在此基础上,根据实际天津机场视景的情况,提出了一种改进的轴向包围盒算法。该算法简化了轴向包围盒中的边缘节点,遍历节点量减小,使得 VEGA 基本代码程序的循环次数减少,这样提高了视景屏幕的刷新率,实现了碰撞检测时间的缩短,加快了视景碰撞的响应速度,提供了优秀的视觉仿真效果。(收稿日期:2007 年 6 月)

参考文献:

[1] 王志强.碰撞检测问题研究综述[J].软件学报,1999,10(5):545-551.
[2] 陈学文,丑武胜,刘静华,等.基于包围盒的碰撞检测算法研究[J].计算机工程与应用,2005,41(5):46-50.
[3] 周之平,张飒兵,吴介一,等.基于矩形包围盒的多边形碰撞检测算法[J].中国图像图形学报,2004(11).
[4] 潘振宽,李建波.基于压缩的 AABB 树的碰撞检测算法[J].计算机科学,2005(2).
[5] 马登武,叶文,李瑛.基于包围盒的碰撞检测算法综述[J].系统仿真学报,2006(4).

(上接 189 页)

INSERT INTO $L\_I\_V_i$
(SELECT $Level\_Id$,$newID$ FROM $Level\_V_i$ WHERE $Level\_Name$='R');

(2)找出新插入的维实例标识,放入临时集合变量 temp 中

SELECT $Level\_Id$,$Instance\_Id$ INTO $temp$
FROM $L\_I\_V_i$
WHERE $Instance\_Id$ NOT IN (SELECT $Instance\_Id$ FROM $I\_A\_value\_V_i$);

(3)向"维实例-属性值映射表"中插入新实例第一个属性的值

INSERT INTO $I\_A\_value\_V_i$
SELECT Instance_Id,$L\_A\_V_i$.$Attribute\_Id$,'a'
FROM temp,$Attribute\_V_i$,$L\_A\_V_i$
WHERE $L\_A\_V_i$.$Level\_Id$ =temp.$Level\_Id$
AND $L\_A\_V_i$.$Attribute\_Id$=$Attribute\_V_i$.$Attribute\_Id$
AND $Attribute\_V_i$.$Attribute\_Name$='A';

插入新实例其它属性值的命令类似。

4 结束语

设计数据仓库的版本结构,存储数据仓库在不同时期的多维模式状态,对 OLAP 查询分析有着重要的意义,可以帮助决策者正确理解不同时期数据的意义,从而做出正确的分析与决策。本文基于关系模型设计了数据仓库模式版本的对象结构,给出了版本维护的基本方法。进一步的研究是如何借助模式版本库实现对事实数据的跨版本查询。(收稿日期:2007 年 4 月)

参考文献:

[1] Inmon W H.Building the Data Warehouse[M].3rd ed.王志海,译.北京:机械工业出版社,2003.
[2] Bebel B,Eder J,Koncilia C,et al.Creation and management of versions in multiversion data warehouses [C]//ACM SAC 2004,March 2004.
[3] Eder J,Koncilia C,Morzy T.The COMET metamodel for temporal data warehouses"[C]//Proc 14th CAISE02 Conference,2002:83-99.
[4] Body M,Miquel M,Bédard Y,et al.A multidimensional and multiversion structure for OLAP applications [C]//Proc DOLAP'2002 Conf,2002:1-6.
[5] Grandi F.A relational multi-schema data model and query language for full support of schema versioning [C]//Proc 10th SEBD, 2002.
[6] Golfarelli M,Lechtenborger J.Schema versioning in data warehouses [C]//LNCS 3289:ER Workshops,2004:415-428.