时空相关性在多物体碰撞检测中的应用

徐鸣凯 丁友东 王 肃
(上海大学计算机科学与工程学院, 上海 200072)

摘 要 尽管利用虚拟环境中物体运动的时空相关性来加速物体间的碰撞检测,可以取得不错的效果,但目前的时空相关性算法只应用在两个物体的碰撞检测中,而不能处理多个物体同时发生碰撞的情况;另外,利用时空相关性尚无法解决快速运动物体的碰撞检测问题。针对传统算法的这两个缺陷,在综合利用空间区域划分和时空相关性算法的基础上,通过修改虚拟对象的内部数据结构提出了一种改进算法,实验结果表明,该算法不仅可实现多个物体同时发生碰撞的检测,而且能保证算法在物体高速运动时的有效性。

关键词 时空相关性 碰撞检测 包围盒 链表 空间分割
中图法分类号: TP391.41 **文献标识码**: A **文章编号**: 1006-8961(2006)11-1704-05

The Application of Temporal-spatial Coherence in Multi-object Collision Detection

XU Ming-kai, DING You-dong, WANG Su
(*School of Computer Engineering and Science, Shanghai University, Shanghai* 200072)

Abstract It can make effect for utilizing temporal-spatial coherence to improve the speed of collision detection. But at present, temporal-spatial coherence algorithm can only fit for the situation of two objects. It can't solve the problem of multi-object collision detection. Further more, if there are some fast-move objects in virtual environment, this algorithm also can't obtain good effect. In order to solve the two problems, we bring forward an improved algorithm which can be used in multi-object collision detection and assure the validity when objects move fast in virtual environment.

Keywords temporal-spatial coherence, collision detection, bounding volume, list, space decomposition

1 引 言

碰撞问题作为计算机图形学、计算机动画、仿真机器人和虚拟现实等领域中的一个经典问题,多年来一直受到较多的关注。目前,大多数虚拟对象的几何模型都是由成千上万个基本几何元素(如四面体或三角形面片)组成。由于虚拟环境的几何复杂度使得碰撞检测的计算复杂度大大提高,并且消耗大量的计算机资源,而计算机的图形显示和虚拟环境的交互又要求实时完成,因此碰撞检测往往成为虚拟环境中的一个瓶颈。

一个碰撞检测系统要能尽可能实时准确地判断出任意两个物体是否发生了碰撞。若发生了碰撞,则还需要计算出精确的碰撞点。最简单的检测方法是对两个输入模型的基本几何元素进行两两相交的测试。这种穷举的方法虽然简单精确,但是其时间复杂度达到 $O(n^2)$[1],显然无法达到虚拟现实系统实时性的要求。于是,1996 年北卡罗莱纳州大学的 Gottschalk 等人开发研制了基于 OBB (oriented bounding box)层次包围盒树的碰撞检测算法,称为 RAPID 算法,在很大程度上提高了碰撞检测的速度。之后,国防科学技术大学的魏迎梅教授在层次包围盒算法的基础上,使用了时空相关性来加速碰撞检测。本文在此基础上通过修改虚拟对象的数据结构,并综合利用空间区域划分和时空相关性,提出了一种改进算法,该改进算法不仅可以应用于多个物体同时发生碰撞的情况,并能保证算法在物体高速运动情况下的

收稿日期:2006-08-23;**改回日期**:2006-09-10

第一作者简介:徐鸣凯(1983 ~),男,2004 年获南昌大学工学学士学位,现为上海大学硕士研究生。主要研究方向为图形图像、虚拟现实及多媒体等。E-mail: directcar@163.com

有效性,同时可尽可能地减少空间存储的花费。

2　算法描述

2.1　空间区域划分及预处理

假设整个虚拟空间中共有 n 个物体,分别为 O_1, O_2,…,O_n。在进行碰撞检测前,首先利用空间区域划分法[2,3],把整个虚拟空间均匀剖分成一个个小方块区域(如图 1 所示),接着在某一时刻检查有哪些物体占据着同一个小方块空间,然后只需要对占据了同一个小方块空间的这些物体进行碰撞检测即可。这样便可以迅速地剔除掉相距较远而根本不会发生碰撞的物体对。本文算法就是先在每一个物体的数据结构中添加一个 list 链表,然后在 list 中存放与该物体占据同一小方块空间的其他虚拟对象的标识。因为物体的相邻和碰撞具有自反性,所以为了防止重复,在设定的 list 中存放的物体标识,一定要比当前物体的标识序号大。例如对于物体$O_i(1<i<n)$,与它占据了同一个小方块空间的物体集合为$\{O_{j_1}, O_{j_2}, \cdots, O_{j_n}\}$,则这个物体 list 中存放的对象标识是$\{O_{j_i}\}(i<j_i<j_n)$。这就保证了占据同一块小方块空间中的物体两两都能进行碰撞检测,而不会遗漏任何物体对,也不会有任何的重复。现在,每一个物体的数据结构内都包含了当前这一时间采样点上可能与其发生碰撞的相邻物体的标识。下面就是对每一对相邻物体对,利用时空相关性进行碰撞检测。

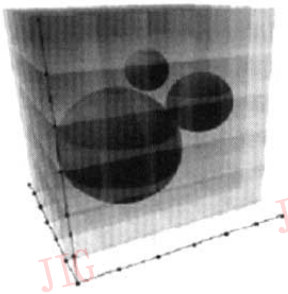

图 1　空间区域划分[2]

Fig. 1　Space decomposition

2.2　基于时空相关性的碰撞检测算法

在虚拟环境中,碰撞检测非常频繁,而且其在时间上的采样点也是十分密集的,因为只有这样才能保证流畅性,也正是由于这个原因,只要物体运动的相对速度不是很大,那么在相邻的碰撞检测采样点内,物体的位置和状态就不会发生很大变化。这就表明:如果两个物体在上一时间采样点上发生了碰撞,则在当前时间采样点上,它们也很可能仍发生碰撞,并且碰撞点与上一次碰撞点的位置非常接近。这就是物体运行的时空相关特性。因此,如果某一时间采样点上,物体 O_i 包围盒树的根节点 R 在遍历物体 O_j 的包围盒树中,且与其叶节点相交,则在下一时间采样点上,它很有可能仍然与该叶节点相交,或者与该叶节点位置相邻近的叶节点相交;同样,如果物体 O_i 包围盒树的根节点 R 与物体 O_j 包围盒树中的某一内部节点不相交,即可判定与该节点的所有后代节点都不相交,在下一时间点,它很有可能仍然与该内部节点不相交。如果物体对象越复杂,其包围盒树的深度也越大,则利用时空相关性加速碰撞检测的效果也越明显。对于简单物体,则相反。所以可以根据这一特点设定一个阈值 μ,当物体对象包围盒树的深度大于这一阈值 μ,则采用时空相关性来对碰撞检测进行加速。当然,如果在整个虚拟环境中存在高速运动的物体,即使利用时空相关性也达不到好的效果。但由于在先前的预处理过程中,已利用了空间区域划分的方法迅速去除了不相邻的物体,并在每一时间采样点上及时更新了物体数据结构内 list 中的值,因此便可以对上一时刻相邻,且在当前时刻仍旧相邻的物体对,使用时空相关性理论来加速碰撞检测。

由于现代硬件的发展,内存容量越来越大,而碰撞检测对实时性又要求较高,所以需采用文献[4]中的遍历跟踪策略,利用存储空间上的花费来提高碰撞检测的实时性。如果物体 O_i 数据结构中 list 链表的任一物体 O_j,其包围盒树的深度大于阈值 μ,就为 O_j 建立一个遍历跟踪表,那么在初次检测物体 O_i 与物体 O_j 的碰撞时,物体 O_i 包围盒树的根节点 R 需要遍历物体 O_j 的包围盒树,它有可能到达某些叶节点,也可能在中途的某些内部节点处就因判断出不相交,而中止递归过程返回。本文便把这些不相交的内部节点、不相交的叶子节点和相交的叶子节点存放在 O_j 的遍历跟踪表中。这些节点便标记了整个遍历过程。在下一时间采样点上,在进行空间区域划分后,再通过更新每一个物体的 list 链表,先把不再相邻的物体从链表中删除,再把新发现的相邻物体加入到 list 中。如果物体 O_j 仍然是物体 O_i 相邻链表 list 中的节点,且其遍历跟踪表中便存放着上一时刻的标记节点,则物体 O_i 就可以从这些

标记节点开始向下遍历物体 O_j 的包围盒树，而不必从根开始，这就等于遍历以跟踪表中每个标记节点为根的子树所构成的树林，由于省去了从根节点到标记节点之间的遍历过程，从而降低了遍历的深度。在这次遍历过程中，由于可能又会产生不同的标记节点，因此需要根据这些新的标记节点去更新这张遍历跟踪表。更新的原则[4]是：若跟踪表中在前一时刻与物体 O_i 根节点 R 不相交的某个内部节点，在当前时刻与根节点 R 相交，则从跟踪表中删除这个节点，并加入对这个节点的子树进行遍历时产生的新的标记节点。另外，如果表中有两个互为兄弟的节点在当前情况下都与根节点 R 不相交，便把这对兄弟节点从表中删除，而用它们的父节点取代。

下面利用一个简单的例子来说明这个过程。假设在 t_i 时刻，在虚拟空间中物体 O_i 的 list 链表里存放着相邻物体 $O_{j_1}, O_{j_2}, O_{j_3}, \cdots, O_{j_n}$，其中 O_{j_1} 的包围盒树如图 2 所示。碰撞检测时，首先物体 O_i 遍历 O_{j_1} 的包围盒树，由于到达内部节点 E_2, E_6 处就已经发现与它们的包围盒不相交，因此不需要再继续往下搜索。整个树中，只到达了叶子节点 E_{14}, E_{15}。则 E_2, E_6, E_{14}, E_{15} 就标记了整个遍历过程，这些节点就是标记节点，可把这些节点存放在对应的 list 节点 O_{j_1} 的遍历跟踪表中（如图 3 所示）。到了 t_{i+1} 的时刻，如果 O_{j_1} 仍然处

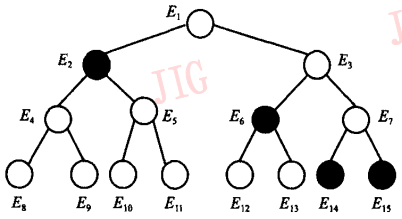

图 2 物体 O_{j_1} 的包围盒树

Fig. 2 Bounding volume of object O_{j_1}

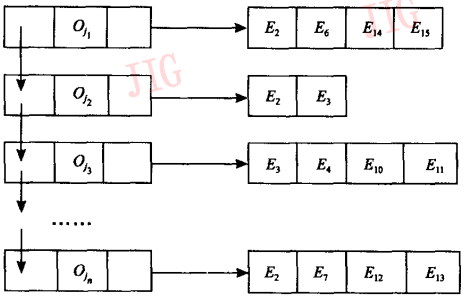

图 3 t_i 时刻物体 O_i 内部数据结构

Fig. 3 Data structure of object O_i on t_i

于 O_i 的 list 链表中，则在进行碰撞检测时，O_i 就只需要遍历节点 O_{j_1} 的跟踪表中标记节点的子树（如图 4 所示）。这样便降低了遍历的深度，也加快了进行包围盒检测的速度，从而加速了整个碰撞检测过程。对于物体 O_i 其他的相邻物体，具体做法是一致的。

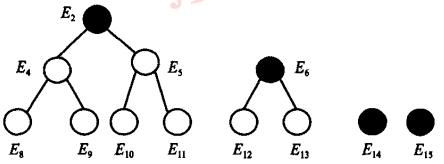

图 4 t_{i+1} 时刻需要遍历的子树

Fig. 4 The sub tree which need to trace on t_{i+1}

3 算法实现

本文的多体碰撞检测方法主要过程是：首先利用空间区域划分，迅速剔除不相邻的物体，这一点能有效地解决时空相关性对存在高速运动物体的情况不适用的问题，同时找出了可能发生碰撞的物体对；然后在物体的内部结构中增加一链表，用于存放与此物体相邻的需要与其进行碰撞检测的其他物体；在碰撞检测时候，设置一个阈值 μ，这个阈值可以使得在最有效的情况下利用时空相关性来检测碰撞。在利用时空相关性检测碰撞时，需要为每一对检测对象开辟一个遍历跟踪表，用来存放标记节点，这是一种通过花费存储空间来增强实时性的方法。但是如果检测的对象是简单物体，层次包围盒树的深度不大，那么利用时空相关性的意义就不大，这种方法虽然花费了存储空间，但对缩短检测时间并没有什么影响，有时甚至还会增加检测时间。因此，当物体包围盒树的深度大于阈值时，才利用时空相关性来加速碰撞检测，否则就使用传统的层次包围盒算法进行检测。这样整个算法就可取得利用尽可能少的存储空间来提高实时性的效果。

一次碰撞检测的计算过程如下：

（1）构造一个数组 Vector，并把当前虚拟空间中的所有虚拟物体对象存放在 Vector 中。Vector 的大小为 Vector. size( )。

（2）遍历数组 Vector，并对其中每一个物体构造 list 链表。链表中存放当前时刻与该物体相邻的其他物体对象的标识。其伪代码如下：

```
for( int i = 0; i < Vector. size( ); i + + )
{
```

```
for(int j = i + 1, j < Vector.size();j + +)
{
 if(Vector[i]与 Vector[j]相邻且 j 不在 Vector[i].list 中)
 // 把物体 O_j 的标示插入物体 O_i 的相邻链表中
    Vector[i].list.insert(j);
 else if(Vector[i]与 Vector[j]不相邻且 j 在 Vector[i].list
        中)
    //把物体 O_j 的标识从链表中删除
    Vector[i].list.delete(j);
}
}
```

(3) 利用此时已经构造好的数据结构,对每一个虚拟物体 Vector[i] 的 list 链表的节点进行遍历。链表的节点数为 Vector[i].list.size()。对当前物体 Vector[i] 和其 list 中每个节点存放的物体进行碰撞检测。其伪代码如下:

```
for(int k = 0;k < Vector[i].list.size();k + +)
{
 if(如果 Vector[i].list[k]的包围盒树深度大于阈值 μ)
 {
  利用时空相关性进行碰撞检测;
  更新这一节点的遍历跟踪表;
 }
 else
  利用传统包围盒方法进行检测;
}
```

为了实现以上算法,在 CPU 主频为 1.5GHz, 内存为 768M 的计算机上,使用 6 个复杂的自由运动的汽车模型进行了试验。整个实验用到的模型共有 21 957 个面片数,分别利用传统 RAPID 包围盒方法和本文中提到的方法进行碰撞检测。在运行过程中记录每一步碰撞检测所需要的时间,和运行了 1000 步的平均碰撞检测时间。效果如图 5、图 6 所示:图

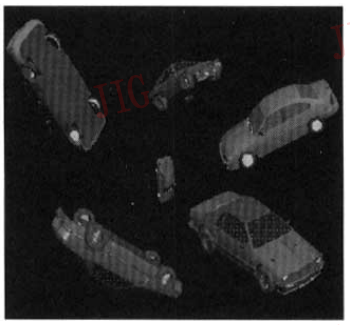

图 5　初始时刻

Fig. 5　Initial state

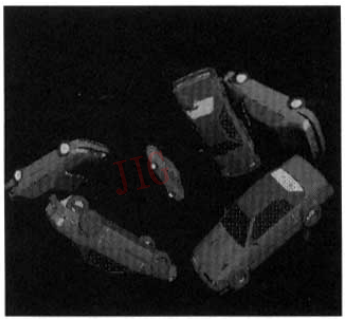

图 6　检测到碰撞发生的时刻

Fig. 6　The time collision happened

5 为初始情况,图 6 为某一时刻发生碰撞时的截图。

实验得出的传统 RAPID 包围盒碰撞检测方法的平均碰撞检测时间为 53.697ms,本文方法的平均碰撞检测时间为 46.328ms。可见,利用本文的方法进行碰撞检测,其平均时间小于传统包围盒方法所需要的时间,虽然多耗费了一小部分内存空间,但在实时性上却取得了较好的效果。

4　结　论

本文针对传统时空相关性算法的缺陷,提出了一种改进算法,该算法不仅能够解决多物体的碰撞检测,同时能保证在存在高速运动物体情况下的有效性。其中阈值的设定不但可以优化整个算法的空间花费,而且可以作为碰撞检测系统的输入值,由用户根据情况自行设定。

该算法还存在着一些不足,有待进一步完善。在并行计算快速发展的今天,用并行算法来改进现有的碰撞检测算法,把高性能计算与碰撞检测相结合将会进一步提高碰撞检测的实时性。

参考文献(References)

1　Zhang Qin, Huang Kun, Li Guang-ming. Improvement of collision detection algorithm based on OBB[J]. Huazhong University of Science & Technology, 2003, **1**:46 ~ 48. [章勤,黄琨,李光明. 一种基于 OBB 的碰撞检测算法的改进[J]. 华中科技大学学报, 2003, **31**(1):46 ~ 48.]

2　Fan Zhao-wei. Research on real time collision detection [D]. Zhejiang University, 2003,10. [范昭炜. 实时碰撞检测技术研究[D]. 浙江大学, 2003,10.]

3　Jimenez P, Thomas F, Torras. C: 3D collision detection: a survey [J]. Computers&Graphics, 2001, **25**(2):269 ~ 285.

4　Wei Ying-mei. Research on collision detection in virtual environment [D]. National University of Defense Technology, 2000. [魏迎梅. 虚拟环境中碰撞检测问题的研究[D]. 国防科学技术大学, 2000.]