在GPU上基于物体空间的碰撞检测

喻家龙,姜太平,汪光阳
(安徽工业大学 计算机学院,安徽 马鞍山 243002)

摘 要:碰撞检测是模拟现实世界关键要素,也是计算机图形学研究的核心内容。提出了一种利用GPU的SIMD、遮挡查询和浮点纹理功能等特性进行碰撞检测的新方法。算法同时遍历一对基于传统的CPU方法的包围体层次树,在遍历的过程中进行所有必要的运算,包括最后的三角形相交测试都是在GPU的顶点着色器和像素着色器中进行的。测试结果显示达到预期的效果。不同于其它的基于GPU的方法,本算法的所有计算都在物体空间中进行,而且对多边形模型的形状和拓扑结构不作任何要求。

关键词:GPU;物体空间;碰撞检测
中图分类号:TP312 **文献标识码**:A **文章编号**:1673-629X(2009)09-0083-04

Object-Space Collision Detection on Programmable Graphics Hardware

YU Jia-long, JIANG Tai-ping, WANG Guang-yang
(School of Computer Science, Anhui University of Technology, Maanshan 243002, China)

Abstract: Collision detection is a key element of the simulation of the real world, but also is the core of the computer graphics research. Presents a novel method for checking the intersection on programmable graphics hardware utilizing its SIMD, occlusion query and floating-point texture capabilities. The algorithm simultaneously traverses a pair of bounding volume hierarchies, which is the traditional CPU-based approaches, performing all the necessary computations during this traversals, including the final triangle intersection tests, in vertex program and pixel program on the GPU. The test indicates that to achieve the expected result. Unlike other GPU-based methods, this method does all computations in object space and does not make any requirements on connectivity or topology.

Key words: GPU; object space; collision detection

0 引 言

利用计算机模拟真实的世界,不仅需要模拟虚拟的模型,同时还要模拟出物体的行为和具体的运动,使得虚拟的环境显得更加逼真。这关键就是要实现物体间的碰撞检测,即在不破坏物体的前提下,两个或多个物体不可能同时占有同一空间区域。目前,虚拟环境中三维模型越来越多,场景也越来越复杂,而实时性的要求却越来越高。因此,如何实现物体之间快速、精确的碰撞检测是计算机图形学中许多模拟算法的核心,也是文中研究的目的所在。

大体上,碰撞检测主要分为基于时间和空间两种方法,而基于空间又可分为基于物体空间和图像空间。基于物体空间的碰撞检测国内外已经有了大量的研究工作,采用如层次包围盒、几何推理、空间分割等技术[1,2]。一些经典的开源软件包,如RAPID、I-COLLIDE[3]、SWIFT++等,这些算法大都依赖于模型复杂度,都是在CPU中完成。

随着半导体技术的迅猛发展,当前GPU在性能上已经超越CPU,最主要的原因是采用流处理和SIMD(Single Instruction Multiple Data)相结合的体系架构。另一方面,GPU的体系结构也有了质的转变,固定渲染管线被可编程管线所取代。当前的GPU已经进化成一种通用的可编程处理器,而不仅只做与图形相关的事情。这也使得许多研究人员研究将GPU用于其他计算,如矩阵计算、光线跟踪及碰撞检测[4~8]等。

传统的基于CPU的碰撞检测,大多都是利用包围体层次结构来实现快速的碰撞检测。

文中研究把这项工作放到GPU上来,利用GPU来实现层次的碰撞检测。算法对包围体层次树进行遍

收稿日期:2008-12-03;**修回日期**:2009-03-05
基金项目:国家863项目(2007AA01Z338)
作者简介:喻家龙(1984-),男,硕士研究生,研究方向为三维图像处理;姜太平,博士,研究方向为三维图像处理;汪光阳,教授,研究方向为工业自动化。

历,并且在遍历过程中的所有计算,包括最后的三角形相交测试,都在GPU中完成。算法对多边形模型的任何形状及拓扑结构都是适用的。

1 三角形在GPU上的相交测试

假设两个三角形构成的网格模型(A和B)进行相交检测,最原始的办法就是把A中的所有三角形与B中的所有三角形进行相交检测。在GPU中,$m * n$个三角形相交测试需要调用$m * n$次像素着色器。但是在GPU中,这些像素着色器是可以并行执行的,它们之间没有依赖关系,所以能很好地提高效率。

对于三角形之间的相交测试,可以利用已有的很多较为高效的算法,当然不同的算法对于程序的执行效率肯定是不同的。文中采用分离轴判定方法来判断两个三角形的相交,判断是否存在一个轴使得两个三角形在其上面的投影是分离的。若像素着色器中执行的三角形不相交,则将其丢弃(在Cg和HLSL中使用Clip指令,在OpenGL的ARB片段程序中使用KIL指令)。在像素着色器执行过后,通过GPU的遮挡查询功能可以得出相交的三角形的数量。根据GPU遮挡查询特性可以返回写入到缓存中的像素点数,而丢弃的像素着色器并没有写入缓存,所以写入到缓存中的就是相交的三角形对的数量。

用于相交检测的三角形顶点信息都保存在显存中。在GPU中,使用三个浮点纹理坐标来保存三角形的3个顶点信息。当然为了检测两个不同物体空间的三角形模型的碰撞检测,还需要保存用于空间转换的转换矩阵。在顶点着色器中进行顶点的空间转换,再将转换的三角形顶点作为像素着色器的输入信息,执行三角形的相交检测。

2 层次碰撞检测

上述利用原始的方法实现两物体之间的碰撞检测。虽然能够利用GPU的并行性,但在数量庞大的三角形对面前,它同样是非常低效的。在本部分中,将详细介绍文中所提出的算法思想及实现过程,通过在GPU中利用层次包围体来提高相交检测的效率。

2.1 层次包围球的创建

传统的基于CPU的方法,通过同时遍历两个物体的包围体来检测其碰撞。文中算法使用球形来作为包围体,是基于效率性、实用性和精确性之间平衡点的考虑[9],是因为:

1)创建简单,碰撞的精度好。因为层次树到叶子节点才为真正的三角形,相对整个场景来说,这些三角形的数量很少,因此碰撞检测进行到此层时,误差将很小。

2)进行重叠测试的计算量很小,因此只需计算两球心之间的距离与球半径和的大小,就可完成重叠测试,这便于减少运算实践,提高检测效率。

物体包围球层次树的创建是在CPU中进行的。树的每个内部和叶子节点都为一个包围球。创建的层次树必须为一棵平衡树,是为了提高树遍历的效率。每一层的数量为:$n(L) = \lceil \frac{n(L+1)}{2} \rceil$。在遍历两个包围球树$S$和$T$的过程中,当检测到父节点为重叠时,则访问其子节点$(S_i, T_j)$,否则包围球之间不相交终止遍历。文中算法采用广度优先遍历策略,因为深度优先的策略并不适用于GPU并行执行的特点。另外,如果两棵树层次不同,则需要在一棵树的顶部增加节点以使其相同。

2.2 算法的过程

物体包围球层次树创建好之后,接下来的数据处理则由GPU来处理。在树遍历过程中,进行重叠测试,需要使用一个缓冲区(Index Buffer)来暂存这些节点对的索引,这个缓冲区用来索引到层次树包围球节点的索引以获取所需信息(如球的中心点和半径)。

另外,还需分配一个2D缓冲区(Count Buffer),用于保存相交节点的重叠数。它的行数与层次树的层数相同,在遍历树的过程中创建。首先,假设对L层各节点进行节点对包围球的重叠测试,如果测试通过,则将子节点的数量写入到重叠数缓冲区的L行中,否则写入0。如果重叠数缓冲区内包含的都为0(通过硬件的遮挡查询来判断),表明所有的节点对包围球测试都没有通过,则可判断此两个物体肯定不相交。2.3中介绍了层次树中包围球的重叠测试。否则,树的遍历将迭代进行下去,直到叶子节点。在遍历层次树到下一层之前,还需做两步工作。首先要实时更新重叠数缓冲区,它是实时动态变化的,然后根据重叠数缓冲区来创建索引缓冲区。这两个步骤将在2.4详细介绍。如果遍历到达了叶子节点,则执行三角形对的相交检测。这个算法流程图如图1所示。

2.3 包围球的重叠测试

节点对索引缓冲对应所有的包围球对。像素着色器对包围球进行重叠测试,首先根据节点对索引缓冲区中的索引值i获得节点的包围球的属性值,也就是两个纹理坐标值:中心点和半径。在像素着色器程序中使用这两个纹理坐标值进行包围球的重叠测试。抛弃所有不发生重叠的包围球对,否则将通过重叠测试,将节点的子节点的数量写入到重叠数缓冲所对应的行中。

2.4 节点对索引缓冲的建立

下面的方法来更新重叠数缓冲区、节点对索引缓冲区和对应的顶点数组。

重叠数缓冲区的创建如图 2 所示。每一行相应的值等于下一行相应两个值的和。

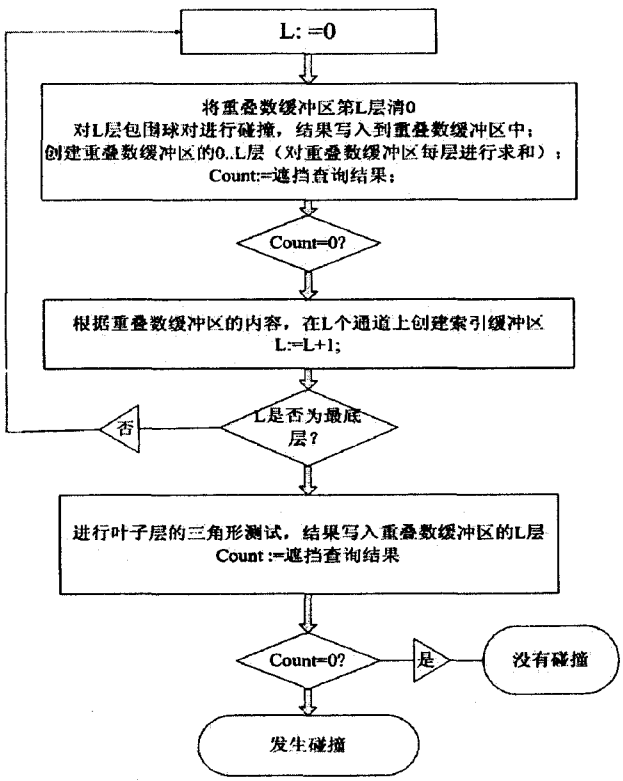

图 1　算法流程图

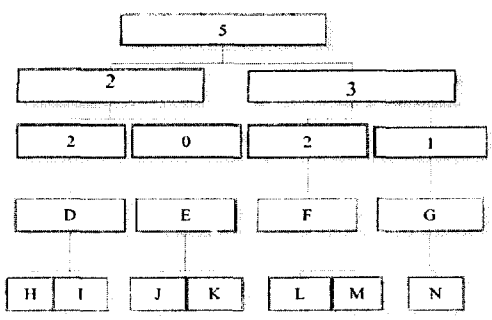

图 2　重叠数求和

接下来创建节点对索引缓冲。节点对索引缓冲区中每行的长度等于层次树对应层其父节点通过重叠测试的节点数,这也等于重叠数缓冲区中对应层所有数值的求和。节点对索引缓冲中每行的第 n 个成员对应于层次树对应层的第 n 个节点,这个节点可以通过根节点遍历得到。根据重叠数缓冲区中的值,可以准确推断出搜索到此节点的遍历路线,从而创建出节点对索引缓冲区。

下面将介绍如何使用此方法在 GPU 中实现节点对索引缓冲的创建。

重新分配一个与节点对索引缓冲大小相同的临时缓冲区。它的每个成员包含两个分量,当前节点索引(记为 CurrentNode Index)用于保存在遍历过程中当前被访问的节点,当前子索引节点(记为 CurrentChild Index)用于保存被搜索的那个节点。

一开始,需要对这个临时缓冲区进行初始化,将每一行的当前节点索引都设为 0,表明从根节点开始搜索,而子索引节点赋予以 0 开始的依次递增的枚举值。在像素着色器程序中进行如下过程:

```
For each pass i=1,…,L
{
    读取 CountBuffer 第 i 行第 2 * currentnodeindex 个值;
    If(此值大于 currentchildindex)
      Currentnodeindex=2 * currentnodeindex;
      Currentchildindex=currentchildindex;
    Else
      Currentnodeindex=2 * currentnodeindex+1;
      将重叠数值减去 currentchildindex;
}
```

这样 L 个通道完成之后,可以求得这个临时缓冲区。再根据这个临时缓冲区和当前节点对索引缓冲的内容,就可获得新的节点对索引缓冲。这个过程如图 3 所示。

3 实验与结果分析

包围球树是用纹理矩形来存储的,包含的每个纹素都为 32 位的纹理坐标。在算法中描述的重叠数缓冲区和节点对索引缓冲区作为临时的缓冲区,是用作渲染操作的目标纹理。

文中算法是在 VS. Net2005,使用 Direct3D HLSL 进行实现的,并在 CPU P4 双核 2.80GH、显卡为 NVIDIA Geforce7600 上进行了实验测试。测试的场景是距离指定的两个相同物体模型之间的碰撞,一个物体绕着一个固定的轴进行旋转,实时对两个物体进行碰撞检测,最后计算平均的碰撞时间。减小两物体的距离,再重复进行上述的操作步骤。

文中的算法与基于 CPU 的碰撞检测方法进行了性能上的对比,在 CPU 上也是使用相同的遍历策略。测试的模型和结果如图 4 和表 1 所示。

表 1　利用 CPU 和 GPU 进行的对比

距离(cm)	平均碰撞时间(ms)	
	利用 GPU 进行层次碰撞检测	利用 CPU 进行层次碰撞检测
3.0	56	62
2.5	51	55
2.0	40	46
1.0	26	30
0.5	19	18

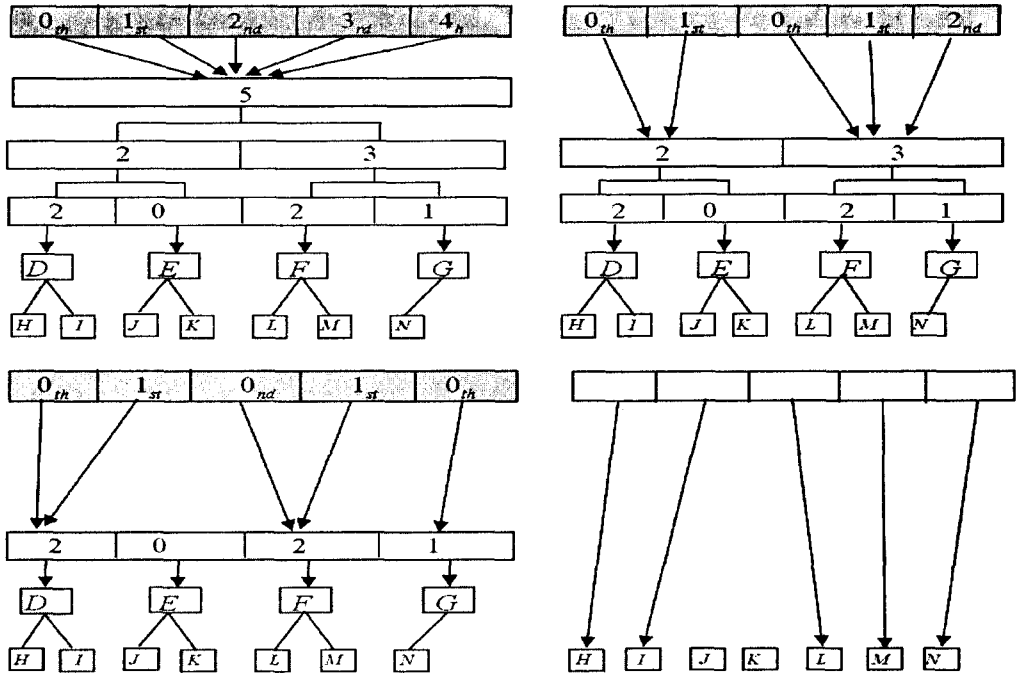

图 3 创建节点对索引(灰色的为临时缓冲区的子索引节点)

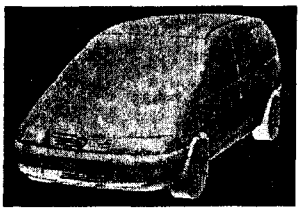

图 4 测试模型

4 算法的优化

上述算法对两个物体层次包围树的遍历,都是从树的最高层开始的。如果一个层至少有一半包围球对相交,即使比例更小,那么遍历都可以从此层开始。这无疑对效率将是一个不小的提升,因为层次树的遍历计算量也很大。然而从哪一层开始遍历从而达到更大的效率,必须提前知道包围球对的重叠数量,显然这是不可能的。

在实时的应用程序中,物体都是平滑移动的。因此可以利用基于时间连贯性的启发式的推理:每一个物体对,在对其层次树遍历的过程中,如果在某一层包围球对的重叠率大于所指定的重叠率(重叠率可通过硬件的遮挡查询获得),标记此遍历层。那么在下一帧,对层次树的遍历就可以从此层开始。根据此层的重叠数,又可以调整下一次碰撞检测开始遍历的层。

5 结束语

提出了在 GPU 中进行碰撞检测的新方法。利用 GPU 的一些特性实现场景中三维物体的实时碰撞检测,有效减轻了 CPU 的工作负荷。后续的工作可以在大场景中多物体之间的碰撞检测中展开,提高算法的性能,以及使用不同的包围体对其算法性能的影响,当然上段所提及的优化算法也可作为后续工作。

参考文献:

[1] 裘 初,费广正,石民勇.可编程图形硬件综述[J].北京广播学院学报,2004,11(3):13-19.

[2] 周之平,吴介一,白伟冬,等.基于矩形包围盒的多边形碰撞检测算法[J].中国图象图形学报,2004,9(11):1294-1303.

[3] Govindaraju N K,Redon S,Lin M,et al. CULLIDE: Interactive Collision Detection Between Complex Models in Large Environments using Graphics Hardware[C]//in Proc of Graphics Hardware. San Diego,California:[s. n.],2003: 25-32.

[4] 朱连章,庄 华.基于图像空间的复杂模型碰撞检测算法[J].计算机工程与设计,2007,28(15):3675-3681.

[5] 范昭炜,万华根,高曙明.基于流的实时碰撞检测算法[J].软件学报,2004,15(10):1505-1514.

[6] Baciu G,Wong S,Sun H. Recode: An Image-Based Collision Detection Algorithm[J]. Journal of Visualization and Computer Animation,1999,10(4):181-192.

[7] Baciu G,Wong S K. Image-based techniques in a hybrid collision detector[J]. IEEE Transactions on Visualization and Computer Graphics,2003,15(6):254-271.

[8] Knott D,Pai D K. CInDeR: Collision and interference detection in real-time using graphics hardware[C]//in Proc. of Graphics Interface. Canada:[s. n.],2003:73-80.

[9] 王 季,翟正军,蔡小斌,等.基于球深度纹理的实时碰撞检测算法[J].系统仿真学报 2007,19(11):2503-2506.