一种基于 GPU 的碰撞检测算法①

GPU-Based Collision Detection Algorithm

苏 诺 季桂树 邓 拓 (中南大学 信息科学与工程学院 湖南 长沙 410083)

摘 要: 实时碰撞检测是计算机图形应用中不可缺少的组成部分。随着高性能可编程图形处理器（GPU）的发展，出现了许多利用 GPU 来解决复杂物体间的碰撞检测问题的方法。提出了一种基于 GPU 的对参数化表面的碰撞检测方法。通过使用几何图像表示的参数化表面，实时的生成 GPU 优化的包围体层次结构，然后在这个层次结构的基础上实现优化的基于 GPU 的层次碰撞检测算法。结果显示本方法可以有效的提高碰撞检测的速度，相对于在 CPU 上实现同样的层次结构遍历方法，基于 GPU 的方法可以将碰撞检测速度平均提高 13%左右。

关键字: 图形处理器 碰撞检测 几何图像

在很多计算机图形学和虚拟现实的应用中，碰撞检测(Collision Detection，CD)是一个最为基本而且非常重要的组成部分，主要应用领域包括虚拟制造、CAD/CAM、计算机动画、物理模拟、游戏、飞机和汽车模拟、机器人技术，以及几乎所有的 VR 模拟。目前大多数碰撞检测算法均在物体空间实现，而且采用了诸如层次包围盒、几何推理、空间分割等技术。一些经典的软件包有 RAPID, I-COLLIDE, QuickCD。传统的碰撞检测算法大都依赖于模型的表示和场景复杂度，计算均由 CPU 完成。这给系统，尤其是处理大规模场景的系统，带来了沉重的计算负担。碰撞检测是实时渲染的主要瓶颈之一。

近年来，随着图形处理器(GPU)性能的大幅度提高以及可编程特性的发展，人们开始将图形流水线的某些处理阶段以及某些图形算法(如光线跟踪、碰撞检测)从 CPU 向 GPU 转移。GPU 有着更高的浮点计算能力，Inter 公司 3.0GHz Core2 Quad 的 CPU 的浮点运算能力在 96Gflosps(每秒 10 亿次浮点运算)，而 NVIDIA GeForce 8800 GTX 的浮点运算能力能够达到 330Gflops，加上 GPU 内在的并行结构，进一步提高了 GPU 的性能优势。通过在 GPU 上实现图形算法，既可以使实时渲染的速度得到总体的提升，又可以节省 CPU 的资源[1]。

迄今为止，已经提出了一些 GPU 上的基于屏幕空间的碰撞检测方法，如文献[2,3]。可是许多屏幕空间的方法都要受到模型和拓扑性质的限制，如很多算法要求模型是闭合的，有些方法只可以处理凸体对象。屏幕空间的方法还有一个问题是没有考虑在渲染图元时引入的几何误差。即使保留了这些误差，屏幕的分辨率还是严重影响着算法的效果。还有些算法只报告对象的潜在碰撞集(PCS)，图元间的准确碰撞检测还是交由 CPU 完成。

本文中提出的基于 GPU 的碰撞检测算法是通过使用几何图像表示的参数化表面，实时的生成 GPU 优化的包围体四叉树层次结构，然后在这个层次结构的基础上实现优化的基于 GPU 的层次碰撞检测。

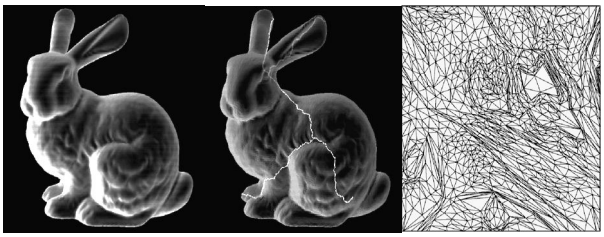
1 算法概述

本文算法使用几何图像(Geometry Image)来表示几何模型。几何图像的概念最早是由 Gu 等人[4]在 2002 年提出的。他们的目的是为了利用图像的方法来表示几何模型，从而达到对几何模型的压缩和快速绘制。曲面上的一些属性，如几何位置、法向量、颜色等信息转换为彩色图像中的像素信息。图形硬件可以对几何图像进行高效的存储和处理。为了生成几何图像，一个任意拓扑的网格首先被切割开，变为一个

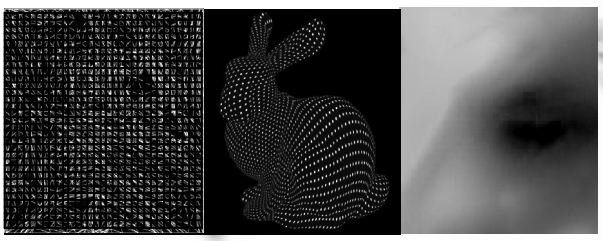
① 收稿时间:2008-12-24

© 中国科学院软件研究所 http://www.c-s-a.org.cn

与圆盘(disk)同胚的开网格，然后在一个矩形域内对这个开网格进行参数化，最后以完全正则的模式在矩形区域上进行采样，用像素的 RGB 分量分别存储顶点的 x，y，z 坐标值。如图 1 所示。



(a) 原网格对象　(b) 切割网格　(c) 参数化



(d) 采样　(e) 采样　(f) 生成的几何图像

图 1 几何图像生成过程

本文的算法由生成 AABB 树、重叠测试与层次遍历、精确相交测试三个阶段组成。

(1) 生成 AABB 树：碰撞检测的第一步是对每个几何图像创建 AABB 树。对于静态几何图像，这步只需要进行一次，而对于动态几何图像，我们需要对每一帧重复进行此步操作。

(2) 重叠测试与层次遍历：若当前分属于两个 AABB 树的节点对不重叠，则不需要对这两个节点的子节点组成的节点对再进行重叠测试；若当前节点对重叠，则需要对它们的子节点组成的节点对进行重叠测试。这需要对两个 AABB 树同时进行遍历。在基于 CPU 的方法中常用的深度优先遍历并不能有效的利用 GPU 的并行性。所以我们采用广度优先遍历，这样可以对在同一层次的节点对上的 AABB 同时进行重叠测试。

(3) 精确相交测试：若 AABB 树的倒数第二层检测到相交，则需要对相应的最后一层(叶子层)的节点对进行精确的三角形/三角形相交测试。若检测到相交，则说明两个 AABB 树所包围的两个对象碰撞。

2 生成AABB树

包围盒层次法是碰撞检测算法中广泛使用的一种方法，它曾经在计算机图形学的许多领域(如光线跟踪)中得到很深入的研究。其基本思想是用体积略大而几何特性简单的包围盒来近似地描述复杂几何对象，进而通过构造树状层次结构可以越来越逼近对象的几何模型，直到几乎完全获得对象的几何特性。在对两物体进行碰撞检测时，首先采用适当的层次遍历方法对位于各层次的包围盒进行重叠测试，由于求包围盒的交比求物体的交简单，因此可以很快速排除许多不相交的物体，若相交，则只需对重叠包围盒所包含的网格片段进行进一步的相交测试。轴向包围盒 AABB (axis-aligned bounding boxe)在碰撞检测的研究历史中使用的最久最广，一个物体的 AABB 被定义为包含该物体，且边平行于坐标轴的最小六面体。AABB 最大的特点是计算两个 AABB 重叠非常简单。

用坐标值最小和最大的两个点来表示一个 AABB。对于一个给定的大小为 ru×rv 几何图像，我们用分别存储了最小点和最大点的几何图像来表示 AABB 树的叶子层。也就是纹理元(i，j)存储的 AABB 包围了给定的几何图像中纹理元(i，j)，(i+1，j)，(i，j+1)，(i+1，j+1)所定义的四个顶点形成的四边形，其中 $i \in \{0,\cdots, ru-2\}$，$j \in \{0,\cdots, rv-2\}$，如图 2 所示。因此生成层次结构的叶子层对应了对求四个纹理元中存储的向量的最小和最大分量的计算。这可以通过单道简单的片段着色器程序完成。

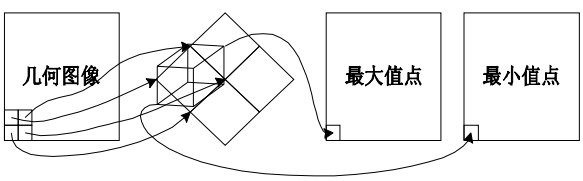


图 2 生成最大/最小点纹理

在确定了叶子层后，可以采用自底向上的方式来完成剩余层次的建立。这样，就可以存储整个层次结构于两个包含了叶子层的几何图像的 mipmap 层次

© 中国科学院软件研究所 http://www.c-s-a.org.cn

中。从叶子层开始，分别用最小和最大滤波器分别建立两个 mipmap。这两个滤波器可以通过单道简单片段着色器程序来实现，每个着色器程序需要对个 mipmap 层次每层用一次。在本文的后面，我们称 mipmap 化了的几何图像为层次图像。

3 重叠测试和层次遍历

3.1 重叠测试

对于那些每一帧产生的动态几何图像，可以假设几何图像的位置和层次图像中的 AABB 是在世界空间中给出。设包围盒 A 和 B 的最大点最小点分别是(boxMinA，boxMaxA)和(boxMinB，boxMaxB)，其中最大点和最小点为含有三个分量的向量。那么两个 AABB 重叠，当且仅当向量

min(boxMax A, boxMax B) − max(boxMin A, boxMin B)没有负的分量。

3.2 层次遍历

基本思想是用步广度优先遍历处理 AABB 树，对每层做一步遍历。若用完全两两测试的方法，需要对来自一个对象的每个包围盒和另一个对象的每个包围盒都进行一次重叠测试。而现在所采用的层次遍历的方法只需要对那些上一层的遍历中其父 AABB 被检测到重叠的 AABB 对进行测试。因此，这就需要一些额外的引用作为输入，这些引用指向上一层次被检测到重叠的 AABB 对，也就是标出了当前层次那些 AABB 对需要被检测。假设，上一步遍历得到一个这种重叠 AABB 对引用的表，那么这个表可以作为当前步遍历的输入，对于表中的每个输入，当前步遍历有 16 个重叠测试需要执行。

遍历的每一步中，考虑着色器对每个检查的 AABB 对的输出，如果存在一个重叠，则引用指向这个 AABB 对；否则，输出一个空引用(即一个无效的纹理坐标)，用来标记相应的与后续遍历无关的片段。显然我们应该在用输出流作为后续的输入之前，从中去掉那些空引用。否则每步遍历后，这个引用表的大小将变为原来的 16 倍，到最后一步遍历时，就需要对所有叶子 AABB 对进行处理，这将变为和非层次化结构的两两检测的方法一样。遍历将在没有重叠对剩余时停止，这可以用遮挡查询(occlusion query)很容易的检测到。

3.3 非均匀流压缩

流压缩(stream reduction)是去除数据流中不需要的元素的过程[5]。在多道(multi-pass)GPU 算法中，一个通道的输出流将作为另一通道的输入流使用。在第一个通道过后，有些数据元素将不再被需要(在本算法中即那些空引用)，所以应该对这些元素标记后删除。因为片段着色器不具备在内存的特定位置写入数据的能力，这些不需要的元素仍将存在于输出流中，所以在数据流进入下一个通道前，应通过流压缩删除无用的元素。

要把非压缩流(在本文中即含有空引用的引用表)转化为相应的压缩流，第一步要确定非压缩流中的非空输入在压缩流中的位置，将其作为索引值，存储于一个与非压缩流同样大小的索引表中。第二步，根据这个索引表把非压缩流转化成等价的压缩流。

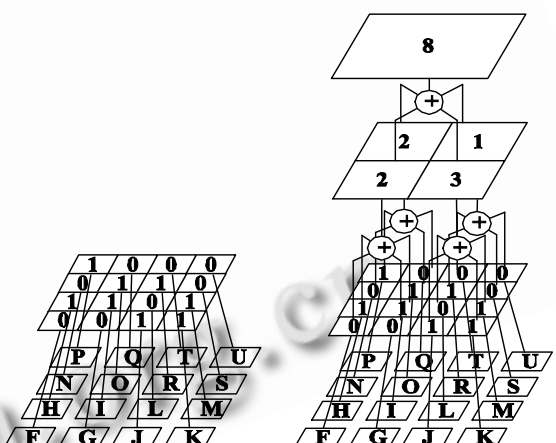


图 3　非空元素个数计算

本文算法第一步的操作同文献[6]类似，但因为这里处理的是2D流，所以需要构造一个四叉的层次结构，将其存储在一个 mipmap 2D 纹理里。为了简便起见，假设最底层(base level)的纹理和非压缩流的大小完全一样。通过一单道着色器，把底层纹理中与非压缩流中空引用对应的纹理元设为 0，与非空引用对应的设为 1。其它的层次通过标准的 mipmap 生成功能来构造。这样，最终的四叉树中的节点保存了在非压缩 2D 流里特定矩形区中的非空引用的个数。如图 3 所示。

第二步，基于这个层次结构，用一个额外的 mip-

© 中国科学院软件研究所 http://www.c-s-a.org.cn

map2D 纹理构造一个索引表，它的底层纹理的大小与非压缩流一致，用于存储非压缩流的项在压缩流中的位置。定义索引值为在各项深度优先遍历 mipmap 层次之前，被访问的非空元素的个数。根据这个定义，计算出 mipmap 层次结构中所有层次的索引，尽管确定各项在压缩流中的位置只需要底层中的索引。用广度优先遍历，自顶向下的确定各 mipmap 层次中的索引值，即通过把父节点的索引值加上从父节点到当前节点之间需要访问的非空元素的个数。如图 4 所示。

最终，把非压缩流中的元素写入相应的索引值给出的位置，从而构造出等价的压缩流。

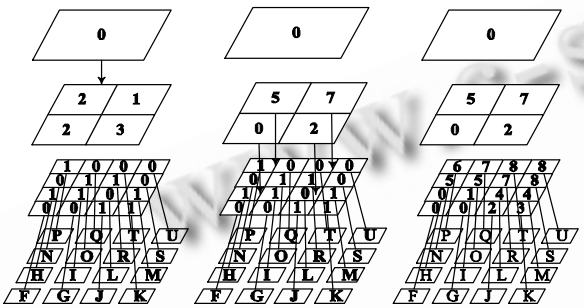


图 4　逐层计算各节点之前访问的非空叶节点个数

4　精确相交测试

AABB 树的倒数第二层检测到相交，则需要对相应的最后一层(叶子层)的节点对进行精确的三角形/三角形相交测试。本文用片段着色程序实现了 SAT 方法来判断两个三角形是否相交[7]。共有 11 个轴需要检查：分别与每个三角形表面法线平行的轴，和分别取自两三角形的边的两两组合(叉乘)形成的轴。将三角形投影到每个轴上，对投影区间进行重叠测试。如果在 11 个轴上的投影区间都重叠，则两个三角形一定相交。如果检查到某个轴上的投影区间不相交，则两个三角形不相交，立即停止检查。不相交三角形对对应的片段将被片段程序丢弃。

使用现代 GPU 的遮挡查询特性，来确定相交三角形对的数量。前述的丢弃所有对应不相交三角形对的片段程序执行过后，实际写入输出缓冲区的像素数即相交三角形对的个数。遮挡查询可以快速返回实际写入输出缓冲区的像素数，从而也就得到了相交三角形对的数量。

5　结果与分析

采用VC++、OpenGL、GLSL在PC机(Mobile Inter Celeron M380　1.6GHz，1GB 内存，ATI Mobility Radeon X600 256M 显存)作为测试环境，用下面的测试方案来对算法的表现进行测试：两个相同的球(三角形面片数量为 1024)相隔一定距离，让其中一个球以固定步长绕一个固定轴旋转。在每一步中，对两个对象检测碰撞，并计算出整个旋转一周的平均碰撞检测时间。随后，每次稍微缩小两个对象的距离，再重复进行整个过程。为了比较基于 GPU 和基于 CPU 的碰撞检测方法的表现，使用同样的遍历策略在 CPU 上实现了 AABB 树的遍历方法，并进行了同样的测试。测试结果如图 5。

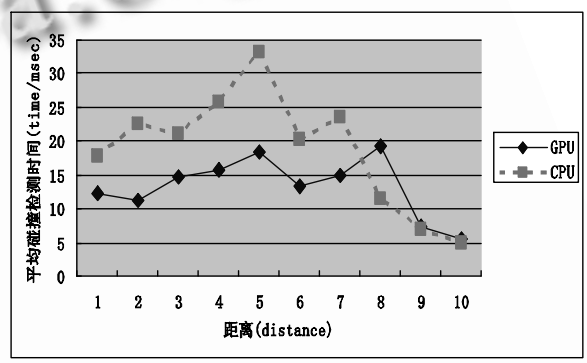


图 5　平均碰撞检测时间随距离的变化曲线

通过比较 GPU 实现的和 CPU 实现的算法的各步的表现，结果显示 GPU 上实现的包围盒重叠测试和三角形相交测试的表现最多可以快两倍，特别是在层次结构的较低层次上时。实验结果表明，算法有效地利用了可编程 GPU 高性能的浮点计算能力，相对于在 CPU 上实现的同算法，基于 GPU 的碰撞检测算法充分利用了图形硬件的并行特点，具有更高的效率。然而，GPU 上的实现需要生成大量的节点索引，这在一定程度上影响了加速效果。

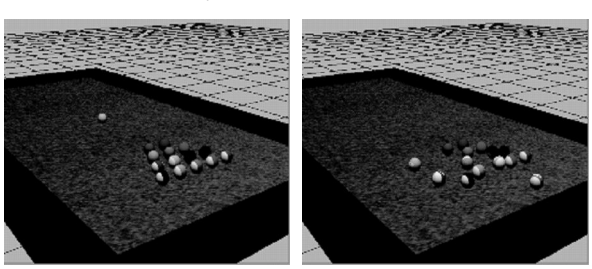


图 6　算法在模拟台球中的应用

(下转第 33 页)

© 中国科学院软件研究所 http://www.c-s-a.org.cn

(上接第 68 页)

总的来说，算法在 GPU 上的实现要比在 CPU 上的实现快。图 6 为算法在模拟台球中的应用效果。

6 结束语

通过利用几何图像在 GPU 上实时的生成包围体层次结构，并对对其进行高效的遍历的基础上，实现了一种基于 GPU 的层次碰撞检测方法。与基于 CPU 实现的类似过程比较，本方法可以获得更快的碰撞检测时间。此外，优于几何图像的优点，可以较容易的对本方法进行扩展，使其可以应用于样条表面或可变形表面的碰撞检测和自相交检测。

参考文献

1 范昭炜,万华根,高曙明.基于流的快速碰撞检测算法.软件学报, 2004,15(10):1505－1514.

2 Govindaraju NK, Lin MC, Manocha D. Quick-cullide: Fast inter- and intra-object collision culling using graphics hardware. The IEEE Conference on Virtual Reality, 2005:59－66.

3 Knott D, Pai DK. CInDeR: Collision and interference detection in real-time using graphics hardware. Graphics Interface, 2003:73－80.

4 Gu X, Gortler Sj, Hoppe H. Geometry images. ACM Transactions on Graphics, 2002:355－361.

5 Horn D. Stream Reduction operations on GPGPU Applications. GPU Gems2, M. Pharr, Ed. Addison Wesley, 2005:573－589.

6 Gress A, Zachmann G. Object-space interference detection on programmable graphics hardware. Geometric Modeling and Computing, 2004:311－328.

7 Christer Ericson. Real-Time Collision Detection. 2005.

8 Benes B, Villanueva NG. GI-COLLIDE: collision detection with geometry images, Spring Conference on Computer Graphics, 2005:95－102.

9 Floater M. Parametrization and smooth approximation of surface triangulations, CAGD, 1997:231-250.

10 Kai Hormann, Bruno Lévy, Alla sheffer. Mesh Parameterization. Siggraph Course Notes 2007.

© 中国科学院软件研究所 http://www.c-s-a.org.cn