文章编号:1001 - 9081(2008)03 - 0683 - 03

蒙皮骨骼动画的碰撞检测研究

熊 涛,付鹤岗
(重庆大学 计算机学院,重庆 400030)
(xiongtao4359@163.com)

摘 要:提出了一种利用蒙皮骨骼的特点改进的碰撞检测算法,该算法使用层次包围体为基础,改进了 OBB 树的生成和更新。实验证明,该算法提高了碰撞检测的效率,能满足复杂 3D 人物的碰撞检测需求。

关键词:蒙皮骨骼;碰撞检测; OBB 树
中图分类号: TP391.41 **文献标志码**:A

Research of collision detection in skinned mesh

XIONG Tao, FU He-gang
(*School of Computer, Chongqing University, Chongqing* 400044, *China*)

Abstract: A collision detection using the trait of skinned mesh was proposed, which was based on the hierarchical bounding volume and ameliorated the building and refitting of OBB tree. The experiment demonstrates that the algorithm enhances the efficiency and could fit the demand for the collision detection of complex 3D characters.

Key words: skinned mesh; collision detection; OBB tree

在很多计算机图形学和虚拟现实应用中,碰撞检测(Collision Detection)是一个最为基本而且重要的组成部分,主要应用领域包括虚拟制造、计算机动画、游戏、机器人技术、CAD/CAM 等。随着计算机硬件的发展,虚拟人物越来越逼真,动作也越来越细腻,这就要求虚拟人物需要用更多的三角形来绘制,但这样就使得虚拟人物的碰撞检测需要消耗更多的时间,因而虚拟人物的碰撞检测对算法的效率要求很高。目前,蒙皮骨骼成为了渲染 3D 虚拟人物的主要主要技术,它克服了关节动画中的接缝问题,而且比单一的网格模型动画更加灵活,因此在虚拟现实、电子游戏、动画制作等领域得到了广泛应用。本文将介绍一种利用蒙皮骨骼的特征改进的碰撞检测算法,以提高碰撞检测的效率和精确性。

1 蒙皮骨骼动画

在骨骼蒙皮动画中,角色由按照一定层次组织起来的骨骼和蒙在其上作为皮肤的网格模型组成[1]。角色要作出各种动作,需要依靠骨骼的各种旋转、移动变化来驱动网格顶点的坐标变化。但与关节动画不同的是,某一骨骼上的顶点并不会只受该骨骼的影响,它往往会受到几个骨骼的影响。为了表示骨骼对顶点的影响,蒙皮骨骼引入了权值。使用它可以解决关节动画中出现的关节与关节之间连接不自然的问题。在 DirectX 的.X 模型文件中,SkinWeights 结构存放了骨骼对顶点的影响权值。图 1 表现了蒙皮骨骼中权值的使用效果。

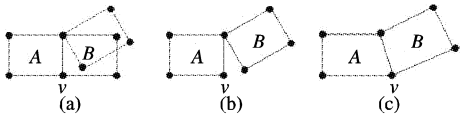

图 1 权值的使用

图 1(a)中,A、B 矩形代表两个骨骼,点 V 是即与 A 相关,也与 B 相关的顶点,虚线矩形代表骨骼 B 旋转后的位置。如果不使用权值,骨骼 B 旋转后,在关节处会发生断裂现象,如图 1(b)。使用权值计算后,关节处将结合得更自然,如图 1(c)所示。令骨骼 A 对 V 的影响权值为 W_A,骨骼 B 对 V 的权值为 W_B,A 的变化矩阵为 $\boldsymbol{M}_A$,B 的变化矩阵为 $\boldsymbol{M}_B$,则 V 的新的坐标 V_1 的计算公式为:

$$V_1 = V \times \boldsymbol{M}_A \times W_A + V \times \boldsymbol{M}_B \times W_B$$

2 算法的设计与实现

在传统的层次包围体法中,包围体的创建和更新时并没有充分利用模型的结构特点,只是对模型进行平均划分。本文利用模型具有骨骼结构的特点,改进了包围体的创建和更新算法,并对包围体的更新提出了两种算法,以针对不同系统的需求。

目前主流的碰撞检测算法主要分为两类:一类是空间分解法和层次包围体法。空间分解法通常适用于稀疏环境中分布比较均匀的几何对象间的碰撞检测,主要的例子有 BSP 树、k-d 树和八叉树等;层次包围体通常适用于由大量的顶点构成的可运动模型间的碰撞检测,如 3D 人物模型、3D 工具模型等,主要的例子有 AABB 层次树[2]、OBB 层次树[3]、k-Dop[4]层次树等。对于蒙皮骨骼角色,本文选择层次包围体法并选择 OBB 层次树作为碰撞检测的方法。由于蒙皮骨骼角色的运动是依靠其骨骼的转动变化或平移变化来带动顶点的变化,而 OBB 盒的方向可以是任意方向的。用 OBB 能包围得更紧密,而且对骨骼方向的变化有更好的效果[5]。

2.1 包围盒的创建

在建模时,通常把人体的几十块最重要的骨骼作为模型的骨骼,对于每一块骨骼的顶点,本文对其创建一个 OBB 包围盒,作为 OBB 树的叶子节点。在模型文件的 SkinWeights 结构包含了一个骨骼对应的顶点。一个给定物体的方向包围盒(OBB)定义为包含该对象且相对于坐标轴方向任意的最小的正六面体,本文利用文献[6]的方法计算 OBB 包围盒。在

收稿日期:2007 - 09 - 26;**修回日期**:2007 - 12 - 07。

作者简介:熊涛(1982 -),男,四川乐山人,硕士研究生,主要研究方向:图形图像处理; 付鹤岗(1950 -),男,重庆人,副教授,主要研究方向:软件工程、图形图像处理、电子商务。

X 文件的 SkinWeights 结构中,数组 vertexIndices 为与该骨骼相关联的顶点的索引,利用该索引值,便可在 Mesh 结构中找到顶点的具体值,并计算出该骨骼的 OBB 包围盒。模型骨骼对应的顶点坐标是相对于模型的局部坐标,计算出的 OBB 坐标也是相对于模型的局部坐标。因而要从局部坐标转化为世界坐标。设一顶点坐标中为 (v_x,v_y,v_z),把其转换为 $P=(p_x,p_y,p_z)$,$Q=(q_x,q_y,q_z)$,$R=(r_x,r_y,r_z)$ 的坐标系,需要进行的运算如式(1):

$$\begin{aligned} v_p' &= P\cdot V = p_xv_x + p_yv_y + p_zv_z \\ v_Q' &= Q\cdot V = q_xv_x + q_yv_y + q_zv_z \\ v_R' &= R\cdot V = r_xv_x + r_yv_y + r_zv_z \end{aligned} \tag{1}$$

该算法记为矩阵,如式(2)所示:

$$\boldsymbol{V}' = (v_x \quad v_y \quad v_z \quad 1)\times\begin{bmatrix} p_x & q_x & r_x & 0 \\ p_y & q_y & r_y & 0 \\ p_z & q_z & r_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

2.2 层次树的生成

由于先确定了树的子节点,本文将采用自下而上而非传统的自上而下的方式来生成 OBB 树。采用这种方式,是因为先生成的是 OBB 树的叶子节点,这是由蒙皮骨骼模型的特点决定的,即:模型已经被划分成了多个骨骼,我们是在骨骼的基础上创建 OBB 的叶子节点的。

2.2.1 两节点的合并策略

当两个节点合并为一个父节点,计算父节点的 OBB 包围盒时,如果采用文献[6]的算法,重新计算新节点的凸包,通过凸包来计算协方差矩阵并计算特征向量来计算 OBB 包围盒的方向和大小,无疑会使得算法的时间复杂度大大增加。令几何元素集合 S_L 和 S_R 为两个节点,$S=S_L\cup S_R$,$B(S_L)$ 和 $B(S_R)$ 分别为 S_L 和 S_R 的 OBB 包围盒,则 $B(B(S_L)\cup B(S_R))$ 的轴方向与 $B(S)$ 轴方向相同,但紧密性略差于 $B(S)$,因而,通过计算 $B(B(S_L)\cup B(S_R))$ 来代替计算父 OBB 包围盒 $B(S)$,虽然效果比计算 $B(B(S_L)\cup B(S_R))$ 略差,但由于计算 $B(B(S_L)\cup B(S_R))$ 只涉及到两个包围盒的各 8 个顶点,因此效率得到大大的提升。

2.2.2 层次树生成流程

在 X 文件中,模型的骨骼构成一棵骨架树,本文要以此生成一棵 OBB 层次树。设 TreeBone( ) 为模型的骨架树,TreeOBB((TreeBone( ))为对一颗骨架树生成一棵 OBB 二叉树,OBB 树的生成算法描述如下:TreeOBB(TreeBone(盆骨))

1)找到 TreeBone( )的左叶子骨骼;

2)找到该骨骼的父骨骼,利用合并策略将该骨骼的 OBB 包围盒与父骨骼的 OBB 包围盒合并为一个新的节点 V;

3)父骨骼是否有兄弟骨骼;

4)有则计算 TreeOBB(TreeBone(兄弟骨骼)),合并 V 与 TreeOBB(TreeBone(兄弟骨骼)),无则跳到2);

5)结束。

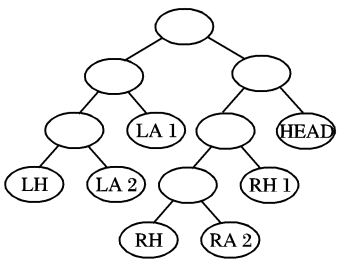

图 2 OBB 生成树

以 TreeBone(左上臂)为例,设左上臂为 LA1,左前臂为 LA2,左手为 LH,右上臂为 RA1,右前臂为 RA2,右手为 RH,头为 HEAD,则生成的 OBB 二叉树如图 2。

2.3 层次树的更新

随着蒙皮骨骼模型的运动,就需要实时地更新 OBB 树,由于模型运动是靠骨骼的旋转或移动来实现的,因而我们更新 OBB 树时,就需要根据骨骼的变化而相应地变化。在 X 文件中,animationset 结构保存若干关键帧,每一关键帧指出了骨骼在该时刻相对于父骨骼坐标系的变换以及位置、朝向等变动[7]。我们需要利用这些关键帧计算出在某一时刻骨骼的插值矩阵 $\boldsymbol{M}_{\text{animation}}$,即在这一时候该骨骼相对于父骨骼的位置矩阵。通过插值矩阵,可计算出该骨骼相对于模型中心的位置矩阵:$\boldsymbol{M}_{\text{combined}}=\boldsymbol{M}_{\text{animation}}\times\boldsymbol{M}_{\text{parent}}$。而骨骼相对于原来骨骼的变化矩阵 $\boldsymbol{M}_{\text{transformed}}=\boldsymbol{M}_{\text{combined}}\times\boldsymbol{M}_{\text{boneoffset}}$。

在关节动画中,与一个骨骼相关联的顶点只受本骨骼的影响,因而其顶点位置相对于本骨骼是没有变化的,其 OBB 包围盒的更新时也就只需更新包围盒的方向和中心坐标,而无需更新包围盒的大小。而在蒙皮骨骼中,由于与一个骨骼相关联的顶点还可能受其他骨骼的影响,其顶点位置相对于本骨骼可能是会发生变化的,所以其 OBB 包围盒的更新还要考虑到包围盒大小的更新。传统算法中,OBB 包围盒的更新需要重新利用 Gottschalk 的方法计算出新骨骼的凸包,并利用凸包计算出协方差矩阵来计算新包围盒的方向、大小和中心坐标。但这样会消耗大量的时间,实时系统中,这是不能接受的。本文提出一种新的更新 OBB 包围盒的方法,可以大大提高效率。设某骨骼 OBB 包围盒的方向为 du、dv、dw,大小为 lu,lv,lw,则骨骼的变化矩阵为 $\boldsymbol{M}_{\text{transformed}}$,骨骼变化后 OBB 的新方向为 du'、dv'、dw',大小为 lu',lv',lw'。

2.3.1 包围盒方向的更新

由于在一根骨骼上的顶点大部分都只受该骨骼的影响,即大部分顶点的坐标相对于骨骼是不会变化的,因而可以设定其包围盒的方向相对于骨骼不发生变化,方向相对于世界坐标的变化就可以用骨骼的变化矩阵来表示,则:

$$\begin{aligned} du' &= du\times\boldsymbol{M}_{\text{transformed}} \\ dv' &= dv\times\boldsymbol{M}_{\text{transformed}} \\ dw' &= dw\times\boldsymbol{M}_{\text{transformed}} \end{aligned} \tag{3}$$

2.3.2 包围盒大小的更新

求出新包围盒的方向后,便可把相对于骨骼发生变化的顶点坐标在 3 个方向投影,将其最大最小投影距离差与包围盒的原大小比较,大的就为新 OBB 包围盒的大小。对于相对骨骼没有发生变化的顶点,便可省略这部分计算。在 X 文件的 SkinWeights 结构中,存在一个骨骼对该骨骼上的顶点的影响权值,权值不为 1 的顶点便是相当于骨骼发生变化的顶点。由于相对于骨骼发生变化的顶点往往是在骨骼与骨骼间的关节处,其数量较少,因而需要的计算也较少,可大大提高效率。设发生变化的顶点为 v^i。

$$\begin{aligned} lu' &= \max(\max(Project(du,v^i)) - \\ &\quad \min(Project(du,v^i))),lu) \\ lv' &= \max(\max(Project(dv,v^i)) - \\ &\quad \min(Project(dv,v^i))),lv) \\ lw' &= \max(\max(Project(dw,v^i)) - \\ &\quad \min(Project(dw,v^i))),lw) \end{aligned} \tag{4}$$

由于在一根骨骼上,大部分的顶点只受到该骨骼的影响,受别的骨骼影响的顶点在加权计算后与原来的变化也不是特别大,因而在许多碰撞检测并不要求特别精确的系统中,往往

只需更新包围盒的方向,这样将有效地减少检测的时间。这种不需更新包围盒大小的算法本文称为算法一,而对于精确碰撞检测系统,算法需要更新包围盒的大小,本文称为算法二。

3　实验

实验在 CPU 主频为 2.66 GHz,内存为 512 MB,显存为 64 MB 的 PC 上进行。编程工具采用 VC ++6.0 和 DirextX9.0。用 cFrame、cMesh、cAnimation 类来分别存储模型的骨骼、顶点和动作信息,并用 cOBBTree 类存储 OBB 层次树。对于层次树的节点信息,创建了 sNode 结构来存储,结构包括了 OBB 包围盒的大小、方向、中心位置和与之对应的骨骼。实验中的模型为 DirectX SDK 中的 tiny 模型,实验的效果如图 3 所示。

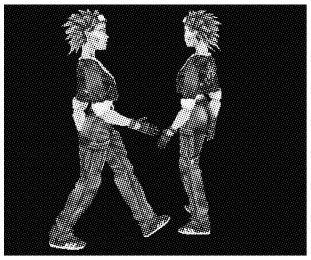

图 3　实验效果

表 1　算法性能比较

三角形个数 6 840	算法运行时间/ms		
	RAPID	算法一	算法二
构建 OBB 树	435	310	310
层次树更新	435	30	105

表 1 列出了算法一、算法二和经典碰撞检测算法 RAPID 算法的性能比较。

算法一和算法二在初始构建 OBB 层次树方法都是相同的,因而消耗的时间是一样的。由于算法一只需要更新包围盒的方向,不需要更新大小,因而比算法二的更新时间缩短了不少。同时两算法的层次树更新算法比较 RAPID 算法明显具有优势。

参考文献:

[1] 金容俊. 3D 游戏编程[M]. 马晓阳, 刘娟. 北京: 电子工业出版社, 2006: 208 - 212.

[2] COHEN J D, LIN M C, MANOCHA D, *et al.* I-COLLIDE an interactive and exact collision Detection system for large-scale environments[C]// Proceedings of ACM Interactive 3D Graphics Conference. New York: ACM Press, 1995: 189 - 196.

[3] AKENINE-MOLLER T, HAINES E. 实时计算机图形学[M]. 普建涛, 译. 北京: 北京大学出版社, 2004: 362 - 365.

[4] KLOSOWSKI J, HELD H, MITCHELL J S B, *et al.* Efficient collision detection using bounding volumes hierarchical of k-DOPs[J]. IEEE Transactions on Visualization and Computer Graphics, 1998, 4(1): 21 - 36.

[5] 魏迎梅. 虚拟环境中的碰撞检测的问题研究[D]. 长沙: 国防科学技术大学, 2000: 22 - 25.

[6] STEFAN C. OBB Tree, a hierarchical structure for rapid interference detection[C]// Computer Graphics Proceeding. New York: ACM Press, 1996: 171 - 180.

[7] ADAMS J. Advanced animation with DirectX[M]. Premier Press: 2003: 108 - 123.

(上接第 679 页)

和排列后的序列都转化为 256 × 256 大小的二维矩阵,这样就产生了一个置换模板。用这个模板去置乱 Lena 图像。再将产生的混沌伪随机二值序列,依次取 8 bit 长度子序列转化为十进制无符号整数,堆积为 256 × 256 的二维矩阵,用此矩阵和置乱后的 Lena 图像做异或运算,结果见图 3(b),解密过程为加密过程的逆过程。图 3(c ~ d)为原图和密图的灰度直方图。图 3(e)为用正确密钥 $x_1^0 = 0.826\,547$, $x_2^0 = 1.234\,655$, $x_3^0 = 0.653\,673$, $x_4^0 = 1.443\,638$ 解密后的图像。图 3(f)是用 $\overline{x}_1^0 = 0.826\,547\,000\,000\,01$, $\overline{x}_2^0 = 1.234\,655$, $x_3^0 = 0.653\,673$, $\overline{x}_4^0 = 1.443\,638$ 的错误密钥进行解密后得到的图像,其中 $\overline{x}_1^0$ 与 x_1^0 相差 10^{-14},其余三个初值不变。可见密钥敏感性很高,密钥相差即使仅有 10^{-14}都无法正确解密。图 3(g ~ h)为对密图分别添加 5% 的椒盐噪声和随机噪声后解密的图像。图 3(i)为对密图的中心进行大小面积为 100 × 100 的剪切。图 3(j)为对密图进行中心 100 × 100 剪切后的解密图像。图 3(k)为对密图进行 JPEG 压缩后的解密的图像,JPEG 压缩的品质因子为 50。可见,密图遭受污染和损失后仍可以正确解密,且解密效果不错。

4　结语

本文提出一种新的二值混沌序列的生成方法,对生成的混沌二值序列的性能进行了分析,并和文献[5,6]中的量化方法进行了比较。以 Matlab7.0 为平台,反复进行了大量的实验,结果表明这种方法生成的二值序列具有很好的伪随机性能,而且这种方法适合任何混沌系统产生的混沌序列,对任何混沌系统和系统的任何参数状态产生的混沌序列进行量化时,都能得到性能很好的伪二值序列。以四维混沌系统来生成混沌序列,目的是为了提高系统复杂性,提高伪混沌二值序列的安全性。把这种方法产生的二值序列用在图像加密,得到了较好的结果。

参考文献:

[1] MATHEWS R A J. On the derivation of a chaos encryption algorithm [J]. Cryptologia, 1989, 4: 29 - 42.

[2] ZHANG HAN, NISHIO Y, SASASE I, *et al.* A new image encryption algorithm based on chaos system[C]// International Conference on Robotics, Intelligent and Signal Processing. Changsha, China: IEEE, 2003: 778 - 782.

[3] PARKER A T, SHORT K M. Reconstructing the key stream from a chaotic encryption scheme [J]. IEEE Transactions on Circuits and System - I: Fundamental Theory and Application, 2001, 48(5): 624 - 630.

[4] 王相生, 王小港, 甘骏人. 基于可变参数混沌的序列密码的设计[J]. 计算机工程, 2001, 27(9): 103 - 165.

[5] 孙秀花, 戴跃伟, 王执铨. 混沌序列产生方法及其在图像加密中的应用[J]. 南京师范大学学报, 2004, 4(1): 56 - 78.

[6] 赵莉, 张雪锋, 范九伦. 一种改进的混沌序列产生方法[J]. 计算机工程与应用, 2006, 23: 31 - 33.

[7] QI GUO-YUAN, CHEN GUAN-RONG. Analysis and circuit implementation of a new 4D chaotic system[J]. Physica A, 2006, 352: 386 - 397.

[8] 吴伟陵. 信息处理与编码[M]. 北京: 人民邮电出版社, 1999.