文章编号:1001 - 9081(2006)01 - 0138 - 02

# 骨骼蒙皮动画中网格优化的研究与实现

李东魁,陈雷霆,王洪福
(电子科技大学 计算机科学与工程学院,四川 成都 610054)
(ldk0517@sohu.com)

**摘 要**:在骨骼蒙皮动画中需要处理大量的网格三角形,而在很多骨骼蒙皮动画的应用中都需要很高的动画实时性,为了提高骨骼蒙皮动画的实时性,对网格的优化是其中一项重要的工作。该文提出了一种优化网格的方法,并予以实现。

**关键词**:骨骼蒙皮动画;网格优化;属性表;渐进式网格
**中图分类号**: TP317.4 **文献标识码**:A

# Research and implementation of mesh optimization in skinned mesh animation

LI Dong-kui, CHEN Lei-ting, WANG Hong-fu
(*College of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu Sichuan* 610054, *China*)

**Abstract:** In skinned mesh animation, large number of mesh triangles need to be dealt with, and high level of real-time is required in many skinned mesh animation applications. To improve the level of real-time of skinned mesh animation, optimizing the mesh is very important. In this paper a method to optimize the mesh was put forward and realized.

**Key words:** skinned mesh animation; mesh optimization; attribute table; progressive mesh

## 0 引言

三维角色动画在计算机动画中占有重要的地位,它广泛应用于游戏动画制作、电影动画制作和广告动画制作。三维角色动画的角色模型实际上是由造型工具构建的网格来表示的,这种网格通常由大量的三角形构成。

通常所讲的角色动画技术的一个重要特点是动画渲染需要耗费大量的时间,所以动画必须提前制作、渲染,之后作为视频文件播放这种动画是非实时的,固定的,且不易进行修改。然而随着虚拟现实、计算机游戏和传统动画制作软件的巨大发展,对实时角色动画的需求越来越大了,同时计算机技术的发展,尤其是消费级别的带有硬件加速功能的显卡技术的发展,为实时角色动画提供了硬件平台,也极大促进了实时角色动画的发展。

实时角色动画技术主要有三种类型:关节动画、单一网格模型动画、骨骼蒙皮动画。其中骨骼蒙皮动画可以看作是关节动画和单一网格模型动画的结合,它需要大量矩阵运算,处理大量的网格三角形。为了提高这种动画的实时性,就必须对网格进行优化。

## 1 网格优化概述

网格优化包括两个主要方面:

一是网格简化,很多三维网格模型由大量的三角形构成,如果不进行优化,在计算机处理中就会占用大量的存储空间,在装载、处理和网络传输时也占用大量的时间开销,解决办法是对网格进行简化,在保持原始三维模型造型特征,尽量不失真的情况下,去除一些冗余的、次要的三角形。网格简化有很多种的方法,如近平面合并算法、顶点删除算法、边折叠算法、重采样算法、优化法、点聚类法、多分辨率方法、基于小波分析的方法。这些算法国内外研究的较多,比较成熟。

第二种是对构成网格的三角形,顶点等数据做一些数据结构上的调整,通过数据结构的优化来获得处理效率的提高。本文提出的优化算法正是这方面的优化,它是根据网格属性对网格三角形做了顺序上的调整,从而获得处理性能上的提高。

## 2 基于属性表的网格优化算法

### 2.1 网格的数据表示

本文使用 DirectX 三维图形库制作骨骼蒙皮动画为例讲述这种三维模型网格的优化的方法。DirectX 自版本 DirectX 8.1 到现在版本 DirectX 9.0 中都用到了属性标识 3DXMESH 接口。这个接口通过封装以下成分而提供了基本的网格模型操作函数。

· 一个顶点缓冲(vertex buffer),包含了网格模型上所有的顶点。

· 一个索引缓冲(index buffer),包含了构成网格模型的所有三角形的三个顶点在顶点缓冲中的位置。

· 每个三角形的属性编号(attribute 属性标识)。是一个 32 位的整数,通过它,可以把网格模型的三角形分成若干组(每个组的成员具有相同的 Attribute 属性标识)。不同的组具有不同的纹理贴图或材质,可以分别绘制。本文的优化算法正是基于此。

### 2.2 属性表优化排序

一个 Mesh 由数个子集组成。子集是 Mesh 中的一组使

**收稿日期**:2005 - 07 - 22 **基金项目**:信息产业部电子信息产业发展基金资助项目

**作者简介**:李东魁(1981 - ),男,硕士研究生,主要研究方向:三维游戏引擎; 陈雷霆(1966 - ),男,教授,博士研究生,主要研究方向:数字图像处理、三维图形; 王洪福(1980 - ),男,硕士研究生,主要研究方向:三维游戏引擎.

用相同属性渲染的三角形。这里的属性指的是材质、纹理、渲染状态。每一个子集用一个唯一的非负整数表示其属性标识,如 0,1,2,3 等。

Mesh 中的每一个三角形都与一个属性标识相关联,表示该三角形属于该子集。例如,在一个表示房子的 Mesh 中,组成地板的三角形具有属性标识 0,这就表示这些三角形属于子集 0;同样的,组成墙的三角形的属性标识为 1,他们属于子集 1。

三角形的属性标识存储在 Mesh 的属性缓冲中,这是一个 DWORD 数组。因为每个面对应属性缓冲中的一项,所以属性缓冲中的项目数等于 Mesh 中的面的个数。属性缓冲中的项目和索引缓冲定义的三角形一一对应;也就是说,属性缓冲的第 $I$ 项与索引缓冲中定义的第 $I$ 个三角形相对应。三角形 $I$ 有下面三个索引缓冲中的索引项定义:

$A = I*3$

$B = I*3 + 1$

$C = I*3 + 2$

我们可以从图 1 中看出网格顶点的索引表缓冲和属性缓冲的对应关系。

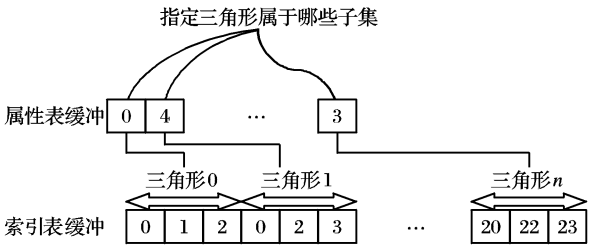



图 1　优化前的网格三角形排列

网格模型的渲染流程是:先设置某个三角形对应的材质和纹理,然后绘制这个三角形。如果我们直接就使用上述未优化的网格模型,我们就需要在 3D 程序中不断的切换当前的材质和纹理,这是非常耗时的,解决的办法就是我们需要按照相同的材质和纹理对索引表进行分组渲染,在图 1 中我们可以看出三角形 0 的属性标识是 0,也有其他若干三角形的属性标识为 0。

通常状况下,当从文件中读取一个网格模型后,所有的三角形在内存里都是属性杂乱放置的,就如图 1 所示。由此我们可以看出在绘制网格模型的各三角形组时,如果三角形是这样放置的,在绘制一组三角形时,就需要遍历整个网格模型的所有三角形,并判断该三角形的属性值是否和当前绘制组的属性值相同。如果相同,就绘制;否则跳过,判断下一三角形。例如,我们要画所有属性标识为 1 的三角形,就需要从头开始遍历:第一个三角形属性标识为 2,不画;第二个三角形属性标识为 4,不画;第三个三角形属性标识为 1,画;以此类推,在遍历到第 8 个三角形时,属性标识为 1,画。一直遍历到最后一个三角形(没有属性标识为 1 的,说明所有属性标识为 1 的三角形都已画完)。如果这时又要画属性标识为 2 的所有三角形,就得照前面方法再来一遍。这样由 $m$ 个三角形,分成 $n$ 组的网格模型,绘制一次就需要判断 $m*n$ 个三角形。在多边形数量很多且分组也很多时,就十分耗时。

优化的方法是把内存中所有的三角形按照图 2 的方式放置。

同时再定义一个结构体如下:

```
Attribute table(s, e):{
  {0,0},{1,2},{3,5},{6,7},{8,11},
                    //数字表示具有相同 ID 的一组三角形
  {12,13},{14,14},{15,15},{16,16}
                    //的起始位置和结束位置
}
```

s 代表开始(start),e 代表结束(end)。这样,把所有具有同样属性值的三角形放在一起,并在另一个缓冲中记录下每一组的起始位置(s)和结束位置(e)。这个缓冲称为属性表(Attribute table)。由于只需记录起始和结束两个值,所以属性表是非常小的。但它使程序的效率成倍增长。每次绘制一个组时,不必把整个网格模型上所有的三角形遍历一遍进行无谓的比较了,而是从要绘制的组的起始位置的三角形开始画,一直到结束位置的三角形为止。如果要画第一组,起始为 0,结束为 0,所以只画第 0 个三角形;画第二组,起始 1,结束 2,只需画第 1、2 个三角形;第三组,起始 3,结束 5,只需从第三个三角形开始画到第五个三角形。以此类推,当按照 Attribute table 绘制所有的组后,整个网个模型就被绘制了一遍。这样,在绘制整个网格模型的时候,每个三角形保证只被访问 1 次。

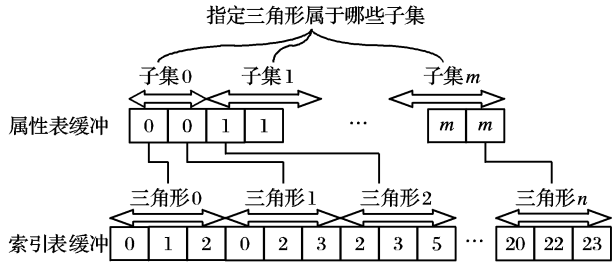



图 2　优化后的网格三角形排列

## 3　算法实现的结果

对上述算法,我们在通用的计算机(Pentium Ⅳ2.4GHz CPU、512M 内存和 64M 显存)上,利用VC++6.0 与 Direct3D 9.0SDK 实现了一个程序进行测试,在这个程序中绘制了一个场景网格模型和几个角色动画模型,总共绘制大约5000 个三角形,并大致上分成 6 组。如果不优化网格,每绘制一帧画面,就需要遍历约 5 000 * 6 = 35 000 个三角形。而其中有5/6的三角形是不需要绘制的。而做了优化之后,由于每个三角形只被访问一次,所以它效率起码提高了 6 倍(少判断了30000 个三角形)。再除去原来对所有冗余的三角形做的操作,效率提升了绝不仅仅是 6 倍。

从表 1 我们看出经过对属性表的优化,动画的帧速获得很大的提升,动画更加流畅。优化前,动画的帧速大约是30Fps,优化后,动画的帧速大约在 185Fps。可见该算法是有效的,它应用于动画中可显著提高动画的速度。

**表 1　优化前后的帧速比较**

| | 帧速/Fps | 三角形数 |
|---|---|---|
| 优化前 | 30.15 | 4968 |
| 优化后 | 185.76 | 4968 |

## 4　结语

本文提出了在骨骼蒙皮动画中对网格的属性表进行优化的算法,从理论和试验两个方面验证了这种算法的有效性,通过测试可知,该算法显著地提高了动画的速度。困扰骨骼蒙皮动画的一个问题就是动画速度问题,该算法从一个方面缓解了这种问题,所以该算法在骨骼蒙皮动画中具有实用意义,当然该算法同样可用于其他动画中。下一步的工作就是对网

(下转第 142 页)

算得到的脊线频率和脊线方向分别为 $f(x, y)$ 和 $\varphi(x, y)$，按下式对图像进行增强：

$$I_{EN}^{n,\Phi}(x,y) = w_1(f(x,y) - F_n) \cdot w_n(\varphi(x,y) - \Phi) I_c^{n,\Phi}(x,y)$$

其中 $w_1$ 和 $w_2$ 分别为窗口函数，其坐标零点位于窗口的中心。最后将所有尺度和所有方向增强后的结果合并起来，构成增强后的指纹图像。

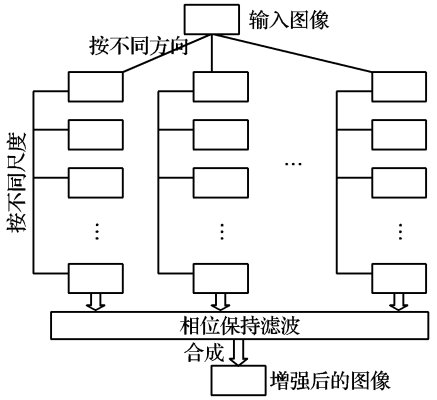
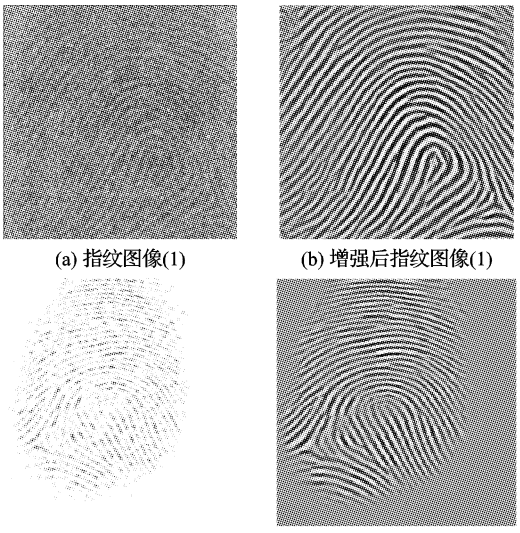



图 4 指纹增强算法结构

## 4 实验结果



(a) 指纹图像(1) (b) 增强后指纹图像(1)

(c) 指纹图像(2) (d) 增强后指纹图像(2)

图 5 实验结果

为了验证本文提出算法的有效性，对两种低质量指纹图像进行增强。第一种是由于采集设备等原因使得图像中含有大量的随机背景噪声(如图 5(a))。第二种是由于皮肤老化，使得指纹图像中脊线呈断续状态(如图 5(c))。

首先利用傅利叶变换的方法来提取指纹图像的脊线走向和频率信息。把图像分为大小为 16×16 的图像块，图像块相互之间重叠 4 个像素。然后利用插值的方法求得图像每一个点的脊线走向和频率信息。设定 Gabor 小波的尺度数和方向数分别设为 6 和 8，取 $w_1$ 和 $w_2$ 为 Bartlett 窗口函数和 Hamming 窗口函数。

增强后的图像如图 5(b)、(d)所示。对于第一种指纹图像，在增强后的图像中已经完全消除了背景噪声，同时也增强了脊线和谷线之间的对比。对于第二种情况，增强后的指纹图像中断续的脊线已经变为连续状态。总之，对于这两种低质量的指纹图像来说，在实施增强算法后，指纹图像的质量都有了显著的提高。

## 5 结语

图像的相位信息对于保持图像中的特征具有重要的作用，本文通过实验说明了相位信息对指纹图像中脊线形状和结构的表达具有十分重要的作用。在指纹增强算法中如果保持图像的相位信息不变，则有助于指纹图像中脊线结构的保持。

基于上述思想，本文提出了一种基于相位保持的指纹图像增强算法，首先利用傅里叶变换的方法获取指纹图像的局部脊线走向和频率信息，然后在相位保持的原则下对指纹图像进行增强。最后给出了针对含有随机噪声干扰以及老化皮肤的指纹图像的增强结果。通过增强后的图像可以知道，该算法能够显著增强这两种图像的质量，说明该算法是有效的。

**参考文献：**

[1] JAIN AK, ROSS A, PRABHAKAR S. An Introduction to Biometric Recognition[J]. IEEE Transactions on circuits and system for video technology, 2004, 14(1):4 - 20.

[2] KOVESI P. Phase Preserving Denoising of Images[A]. Proceeding of DICTA1999 Conference[C]. Perth, Australia, 1999.

[3] OPPENHEIM AV, LIM JS. The importance of phase in signals[J]. Proc. IEEE, 1981, 69(5):529 - 541.

[4] MORRONE C, Owens RA. Feature detection from local energy[J]. Pattern Recognition Letters, 1987, 6(5):303 - 313.

[5] FLEET DJ, JEPSON AD. Computation of component image velocity from local phase information[J]. International Journal of Computer Vision, 1990, 5(1):77 - 104.

[6] CHIKKERUR S. Fingerprint Image Enhancement Using Contextual Filters in the Fourier Domain[A]. Proceeding of CSEGSA2004 Conference[C]. Buffalo, USA. 2004.

(上接第 139 页)

格的简化。保持原始模型造型特征，在尽量不失真的情况下，减少原模型的三角形数目。这样做的好处是可以提高绘制的速度，减少网格的存储空间，同时提高网络传送速度。另外一个问题是，较小的、距离较远的 Mesh 不需要太多的三角形，多了纯粹是浪费，所以，在渲染时，没有必要在这些根本表现不出来的地方浪费时间。解决的办法是使用渐进式网格(progressive mesh)。绘制较小或者距离较远的对象，使用精细水准低的网格，而在绘制较大或者距离较近的对象时使用精细水准较高的网格。

**参考文献：**

[1] JOSEPH M. Shape optimization via mesh elements[J]. Collection of Technical Papers, 2004, 45(7): 224 - 232.

[2] ALLA S. Skinning 3 D meshes [J]. International Conference of Shape Modeling, 2003, 65(5): 274 - 285.

[3] GUINGAND M. Analysis and optimization of the loaded meshing of face gears[J]. Journal of Mechanical Designe, 2005, 127(1):135 - 143.

[4] 成迟薏, 潘志庚, 石教英. 递进网格的一种快速生成算法[J]. 中国图象图形学报, 1999, 3(11):946 - 950.

[5] 陈洪亮, 谭建荣. 一种基于渐变物理属性的三角网格简化方法[J]. 计算机学报, 2000, 23(3):285 - 292.

[6] 张丽艳, 周儒荣, 唐杰, 等. 带属性的三角网格模型简化算法研究[J]. 计算机辅助设计与图形学学报, 2002, 14(3):199 - 203.