一种基于 OpenGL 的三维人体运动模型实现

陈忠[1]，赵学辉[2]，孙秋瑞[1]
1.北京师范大学信息科学与技术学院，北京（100875）
2.北京师范大学教育技术学院，北京（100875）
E-mail: Coocoochen@163.com

摘　要：随着计算机图形学的发展，虚拟现实技术已成为当前最为流行的研究领域之一，而虚拟人物造型及人体骨骼动画则是虚拟现实技术中一大难点。OpenGL 是目前广泛流行的一种三维图形编程接口，在传统方法中，利用 OpenGL 对人体动画进行模拟时，由于 OpenGL 是基于顶点坐标和面片的操作，需要保存动画中每一个关键帧的顶点及面片信息，所以通常要进行大量的计算。而 BVH 格式的文件提供了关于人体关节的运动信息，这将大大减少动画的计算量。本文提出了一种基于 BVH 文件格式的模型实现人体动画的技术，并给出实验结果。

关键词：骨骼动画；OpenGL；关键帧；BVH
中图分类号： TP31

1. 引言

OpenGL 是一个强大的三维图形开发接口，它可以利用几种基本的图形元素构建任何三维模型[1]，但在进行复杂的三维建模时，直接使用 OpenGL 中的图元绘制函数或曲面绘制命令是不太现实的，尤其涉及到复杂的动画模型时，而人体运动模型则是一种复杂的三维动画模型。因此，一般情况下可以利用目前较为流行的三维建模工具，如 3DMAX，MAYA 等软件进行建模，再在 OpenGL 中读取模型，这样就可以减少编程的工作量，为日后修改模型提供了方便。在传统的三维动画构建中，通常采用基于关键帧动画的技术来实现，而关键帧动画涉及大量的顶点及面片信息，存储和计算量庞大。而骨骼动画不需要像关键帧动画那样存储每一个顶点数据，它只需存储每一帧的骨骼信息，因此大大减少了动画模拟的计算量。因此，针对传统方法的各种缺陷，本文提出一种基于骨骼模型的人物运动模型的算法，通过读取 BVH 格式文件的动画模型，实现对人体动画的模拟。

2. BVH 文件介绍

BVH 是 Biovision Hierarchy 的缩写，它是由 Biovision 公司开发的一种描述动作捕获的数据文件格式[2]。这种文件描述的人体动画十分逼真，因为它可以通过真实的人体模特穿上带有传感器的特殊衣服捕获动画[3]，这就比用软件制作出来的动画更为形象逼真；BVH 文件来源也相当广泛，且易于制作，它可以利用 3DMAX、POSER 等软件制作；此外，这种文件是以文本形式存储的，因此操作简单，容易开发。

2.1 BVH 文件的结构

BVH 文件由两个部分组成：骨架信息和关键帧数据块。

骨架信息按树型结构组织，定义了从根“ROOT”节点到终端节点“END SITE”的数据描述，包括各节点的偏移信息和旋转度。对于根节点，还包含 X,Y,Z 坐标，即模型在三维场景中的坐标值；对于非根节点，则只含有偏移量和旋转度，偏移量是指该节点针对它的父节点的偏移位置。骨架信息以关键字“HIERARCHY”开头，然后定义根节点，再定义根节点下的每一个子节点。根节点无父节点，每一个非根节点只有一个父节点，可以有 0 个或多个子节点。“OFFSET”关键字用来定义本节点针对父节点的偏移量，根节点的 x、y、z 三个坐标的

偏移量均为 0。接下来是关键字“CHANNELS”，它给出了关于 channel 的个数和名称，对于根节点，它由 6 个 channels 组成，分别为：Xposition，Yposition，Zposition 以及 Zrotation，Xrotation 和 Yrotation。它们分别表示根节点在三维场景中的 XYZ 坐标位置以及在 Z 轴、X 轴和 Y 轴上的旋转分量。对于非根节点，只有 3 个 channel 值，非根节点只需要记录旋转分量而无需记录节点在三维场景中的坐标位置，因为非根节点的坐标位置可以通过计算父节点的位置以及本节点针对于父节点的偏移位置和旋转分量来获得。当定义完一个节点后，可以用关键字“JOINT”定义子节点。而对于没有子节点的终节点，其标识关键字为“END SITE”。理论上，一个节点可以拥有无穷个子节点[2]。

数据块则用来存放运动帧信息。数据块以关键字“MOTION”开始，关键字“FRAMES”定义了帧数，“Frame Time”定义每一帧的播放时间，如 0.03333 则表示采用每帧播放 0.03333 秒，即每秒播放 30 帧。接下来就是每帧的实际数据，它对应了骨架信息提供的每一个节点。对于根节点来说，平移量为 OFFSET 与 MOTION 定义的平移量之和，对于子节点，平移信息来自骨架信息中的 OFFSET，而旋转信息则来自 MOTION 定义的数据部分，叶子节点没有旋转分量。下图给出了一个 BVH 文件例子。

```
HIERARCHY
ROOT Hips
{
	OFFSET  0.00    0.00    0.00
	CHANNELS 6 Xposition Yposition Zposition Zrotation Xrotation Yrotation
	JOINT LeftHip
	{
		OFFSET   3.0     0.000000        0.000000
		CHANNELS 3 Zrotation Xrotation Yrotation
		JOINT LeftKnee
		{
			OFFSET   0.000000       -18.0    0.000000
			CHANNELS 3 Zrotation Xrotation Yrotation
			JOINT LeftAnkle
			{
				OFFSET   0.000000       -17.0    0.000000
				CHANNELS 3 Zrotation Xrotation Yrotation
				End Site
				{
					OFFSET   0.000000       -3.0     0.000000
				}
			}
		}
	}
	JOINT RightHip
	{
		OFFSET  -3.0     0.000000        0.000000
		CHANNELS 3 Zrotation Xrotation Yrotation
		......
	}
	......
}
MOTION
Frames:  1256
Frame Time: 0.033333
0.00    39.88  -35.01  -1.79   -18.43  -1.74    5.02   -0.34    0.03    6.61
42.19   7.67   -3.87   -7.61    1.40    3.65    15.11  -0.95    2.33    11.06
15.20  -7.25   -10.08   1.61    4.89    18.33   11.12  -17.68   0.00    0.60
40.98   9.99    22.36   0.00   -30.82   11.92   0.00    0.00    0.00    10.97
0.00    12.96  -37.45  -2.92    9.36   -180.00 -80.59   157.44  0.00    0.00
0.00   -35.59   ......
```

图 1　BVH 文件样例

2.2 BVH 文件定义的骨架结构

一个完整的 BVH 文件所描述的人体骨架结构如图 2 所示：

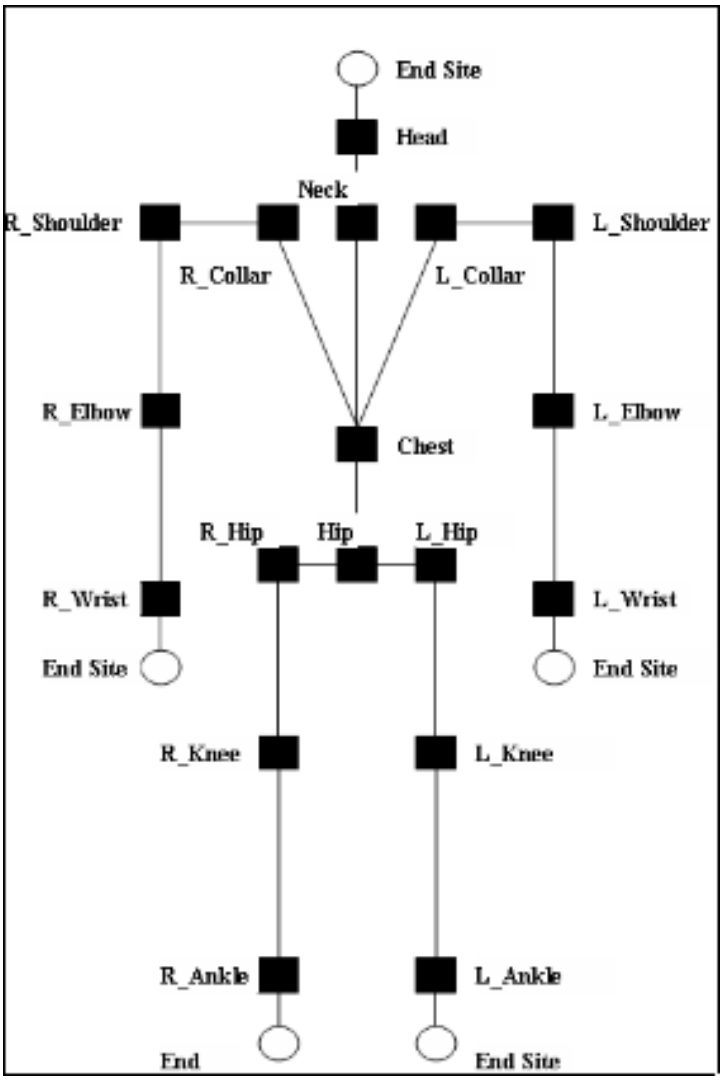

图 2　人体骨骼模型

其中，根节点为 Hip，它控制整个模型在三维场景中的坐标位置，其它节点的位置可以通过计算与该节点的偏移量和旋转度计算获得。

3. 骨骼模型的计算

要正确计算当前节点在场景中的位置，需要使用变换矩阵模型[4]。一个顶点的当前位置 v 通过与变换矩阵 M 相乘后得到目标位置 v’，可以用公式（3.1）来表示：

$$v'=M\times v \qquad (3.1)$$

模型中的每一个骨骼都包含平移以及旋转信息（BVH 文件不包含缩放信息，因此在这里不讨论缩放的处理），将它们组合就可以形成最终的变换矩阵 M，变换矩阵 M 的计算公式如下：

$$M=T\times R \qquad (3.2)$$

公式（3.2）中，T 表示平移矩阵，R 为旋转矩阵。

因此，节点 v 的目标位置计算可以通过公式（3.3）表示：

$$v'=T\times R\times v \qquad (3.3)$$

由 BVH 文件所描述的骨骼模型是一种树型结构，除根节点外，每一个关键帧数据中它并没有给出节点在三维场景中的绝对位置，所有的非根节点在三维场景中的位置都是通过计算其相对于父节点的旋转度及偏移量，再乘以父节点当前的模型矩阵获得的，而父节点的位置则又是通过父节点的 OFFSET 和 ROTATION 及父节点的父节点模型矩阵计算出来的……如此回溯，直至找到根节点。那么，模型中所有节点在场景中的目标位置 v'计算如下：

$$v'=M\times M_{parent}\times M_{grandparent}\ldots\times M_{root}\times v \qquad (3.4)$$

4. 骨骼模型的算法实现

针对 BVH 文件给定的骨骼模型，可以建立如下数据结构进行描述：

```
struct Node     //定义节点信息
{
    float offX,offY,offZ;     //存储三个偏移分量
    float xRot,yRot,zRot;    //存储三个旋转分量
    Node *Parent;                  //存储所在节点的父节点
    Vector<Node*> Children;     //存储当前节点的所有子节点
    String NodeName;             //节点名称
};

class Bvh          //定义 BVH 骨架模型类
{
public:
    Node *Root;                          //存储根节点
    int FrameLocation, curFrame;       //存储运动帧信息（帧信息所在位置、当前所在帧）
    int FrameNumber, FrameLength;    //帧数、每帧长度
    float FrameTime;                     //每帧显示的时间
    bool LoadFile(String &);             //导入 BVH 文件

    Bvh();                               //构造函数
    ~Bvh();                              //析构函数
    void ReadSkeleton();                 //读取骨骼信息
    void ReadFrame();                    //读取运动帧信息
    void Draw();                         //绘制骨骼模型
};
```

定义好骨架模型的数据结构后，便可以通过调用 ReadSkeleton()函数读取 BVH 文件，生成模型的静态结构，最后调用 Draw()函数绘制骨骼模型，在 Draw()函数中调用递归函数 DrawRecursive()完成对整个骨架树模型的绘制。在绘制骨骼模型时，可以通过调用 OpenGL 的 glTranslatef()函数以及 glRotatef()函数来完成模型矩阵的变换运算。另外，在绘制每个骨架节点时需要注意将当前的模型视图矩阵进行压栈操作，以免影响后续不受当前节点影响的节点在场景中的位置，完成所有子节点绘制后，再将视图矩阵还原。OpenGL 中对当前模型

矩阵压栈和出栈的函数分别是 glPushMatrix()和 glPopMatrix()。完成模型绘制的算法采用递归的方式实现：

```
void DrawRecursive(Node *n)
{
    glPushMatrix();                          //将当前模型视图矩阵压栈
    glTranslatef( n->offX，n->offY，n->offZ); //对视图矩阵进行平移变换
    glRotatef(n->xRot, 1.0f,0.0f,0.0f);      // x 轴上的旋转变换
    glRotatef(n->yRot, 0.0f,1.0f,0.0f);      // y 轴上的旋转变换
    glRotatef(n->zRot, 0.0f,0.0f,1.0f);      // z 轴上的旋转变换
    glutSolidSphere(1.0,20,16);              //绘制当前节点（这里用点阵模型）
    for(int i=0;i<n->Children.Len; i++)      //遍历当前节点的所有子节点
    {
        DrawRecursive(n->Children[i]);       //对所有子节点递归调用绘制函数
    }
    glPopMatrix();                           //还原视图矩阵
}
```

以上是实现该骨骼动画模型的关键数据结构和算法，此外，还需要对 OpenGL 进行相应的设置，如视口、光照、材质等，通过调用相关的 OpenGL 函数即可实现。

5. 程序实验结果

运行采用本文算法编写的程序，其结果如图 3 和图 4 所示，两个图分别给出了该骨骼动画的的点阵模型及火柴棒模型表示。

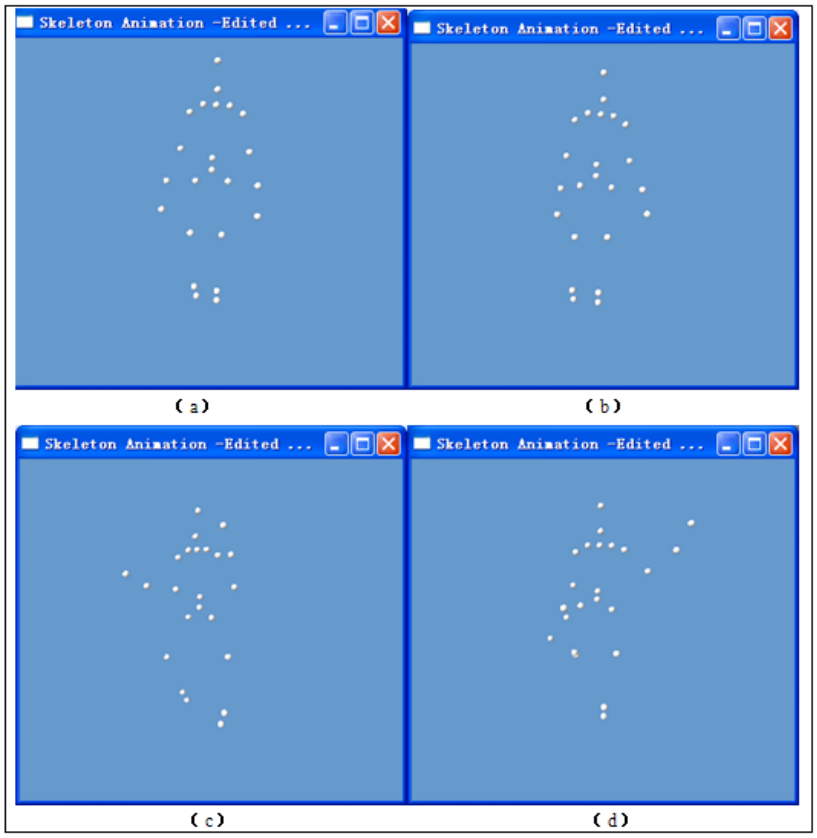

图 3 点阵模型中截取的四个关键帧

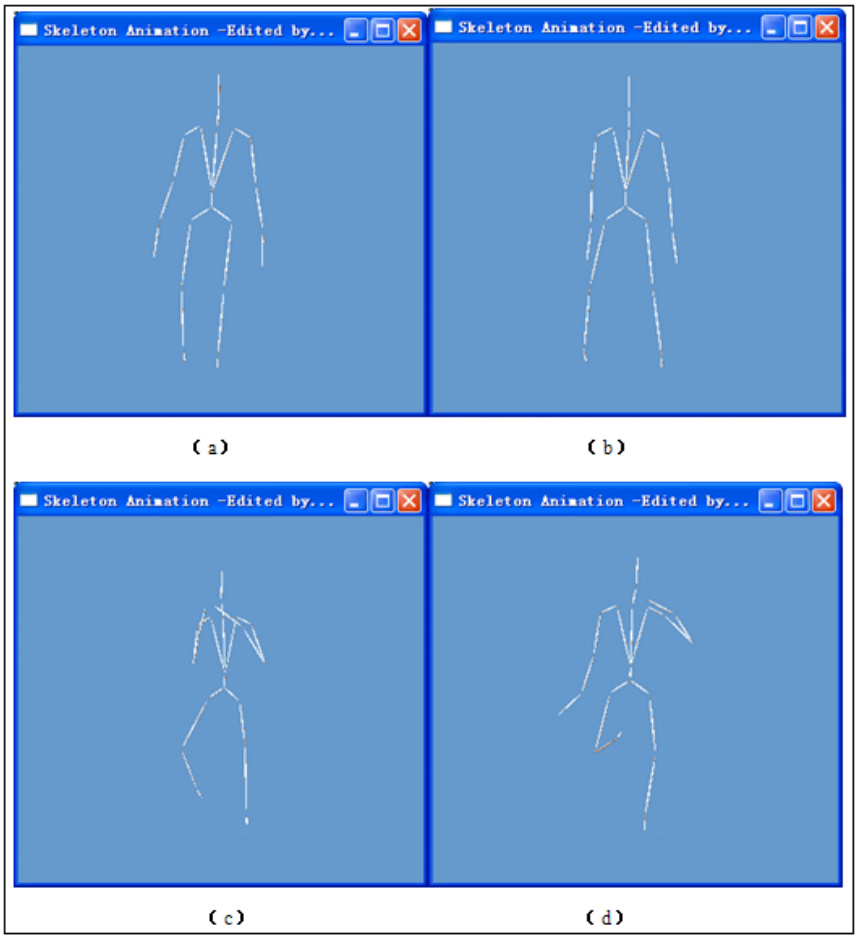

图 4 火柴棒模型中截取的四个关键帧

6. 总结与展望

由于三维骨骼动画的一些优势，如存储和计算量小、易于制作、生成动画逼真等，骨骼动画的应用相当广泛，在 3D 游戏、影视动画、虚拟现实等方面的应用十分普遍，这也是目前动画技术的一个重要分支。

本文对骨骼动画以及基于 BVH 文件格式的骨架模型进行了深入探讨和研究，提出了一种基于该模型格式的算法，并给出了实验结果。在此基础上，可以对该模型加以改进，对骨架模型进行蒙皮，并作纹理贴图处理，就可以形成真实感较强的人物模型。这也将是作者今后进一步研究的内容。

参考文献

[1] Dave Shreiner. Mason Woo. Jackie Neider .Tom Davis 著，徐波 等译. OpenGL 编程指南[M]. 北京:机制工业出版社，2006.
[2] M.Meredith, S.Maddock. Motion Capture File Formats Explained [Z] . Department of Computer Science, University of Sheffield, 2001.
[3] 朱强，张越挺，潘云鹤. 基于紧身衣的人体动画研究[BD/CD] . 软件学报，2002.
[4] 徐明亮 卢红星 王琬 著，OpenGL 游戏编程[M]. 北京:机械工业出版社，2008.
[5] James D. Foley, Andries van Dam, Steven K. Feiner & John F. Hughes. Computer Graphics: Principles and Practice[M] . Addison-Wesley, 1990

An Implementation of Human Animation Based on OpenGL

Chen Zhong[1], Zhao XueHui[2], Sun QiuRui[1]
1.College of Computer Science and Technology, Beijing Normal University, Beijing (100875)
2.College of Educational Technology, Beijing Normal University, Beijing (100875)

Abstract

As the development of computer graphics, virtual reality has become one of the most popular research fields, to create a human animation model is a difficult task. OpenGL is widely used 3D graphical interface for programming. Traditionally, people use OpenGL to simulate human animation. But OpenGL is based on vertices and faces, so every key frame needs to save the information of all these vertices and faces. Thus, large amounts of calculations are necessary. The BVH file supports a simple skeletal model to complete such work, and it needs much less information to store and calculate. In this article, an algorithm to implement human animation by BVH model is proposed, and the result of the application is given too.

Keywords: Skeletal animation; OpenGL; key frame; BVH