基于 LOD 纹理映射的三维场景实时渲染技术

张 敏, 刘用姜

(福建信息职业技术学院, 福建 福州 350007)

摘要: 针对当前三维场景渲染中存在的始终采用高分辨率纹理造成纹理内存浪费,影响三维场景的实时操作,提出了基于 LOD 纹理映射的三维场景实时渲染技术,并阐述了该技术的实现手段,同时应用 OpenGL 语言进行算法的实验研究,实验表明应用 LOD 纹理映射技术能够优化纹理内存,提高三维场景实时渲染速度.

关键词: LOD 纹理; LOD 纹理映射; 实时渲染

中图分类号: TP391.9 **文献标志码**: A **文章编号**:1009-7821(2017)05-0056-06

Real-time rendering technique of 3d scene based on LOD texture mapping

ZHANG Min, LIU Yongjiang

(*Fujian Polytechnic of Information Technology*, *Fuzhou*, *Fujian* 350007, *China*)

Abstract: The current problem is that the 3D scene rendering always uses high resolution texture to cause the waste of texture memory, therefore affecting the real-time operation of 3D scene. This paper presents the real-time rendering technique of 3D scene based on LOD texture mapping, meanwhile illustrates the implementation of this technique and uses OpenGL to realize the algorithm.

Key words: LOD texture; LOD texture mapping; real-time rendering

0 引言

LOD 技术[1-7]和纹理映射技术都常用于加速绘制场景.LOD 技术在模型化简中应用得十分广泛,算法也非常成熟,目前已成功地应用于 3D 游戏、虚拟现实中,但在纹理方面的研究较少.研究表明,对于一个三维场景来说,纹理数据量往往比模型数据量大得多.当前大部分三维场景的制作,一般是一次性将所有的纹理放入内存中,为了追求场景的质量,往往采用较高分辨率且颜色丰富的纹理,像这种始终使用单一、最高分辨率的纹理,十分不利于场景的快速生成和实时显示.因此有必要研究 LOD 技术在纹理映射方面的应用,通过生成纹理的多分辨率细节层次模型[8],应用多分辨率的选择机制完成纹理映射过程,达到优化纹理内存的目的,从而提高三维场景的渲染速度.

收稿日期: 2017-02-13

通信作者: 张 敏(1987-),女,福建永泰人,讲师,硕士,主要从事三维 GLS 研究.E-mail:459032945@qq.com.

1　研究路线

图 1 为基于 LOD 纹理映射的三维场景实时渲染的技术路线，文章应用 OpenGL 语言进行实验验证.首先设置初始视点位置，根据初始的视点位置、观察方向、视角大小定义视锥体[9-10]，计算可见范围，确定可见的实体对象，从而加载模型和纹理数据.在可见的范围内根据场景与视点的距离，恰当地选择纹理细节层次即 LOD 纹理，能够在不损失图形细节的条件下加速场景的显示，提高系统的响应能力，实现实时操作.

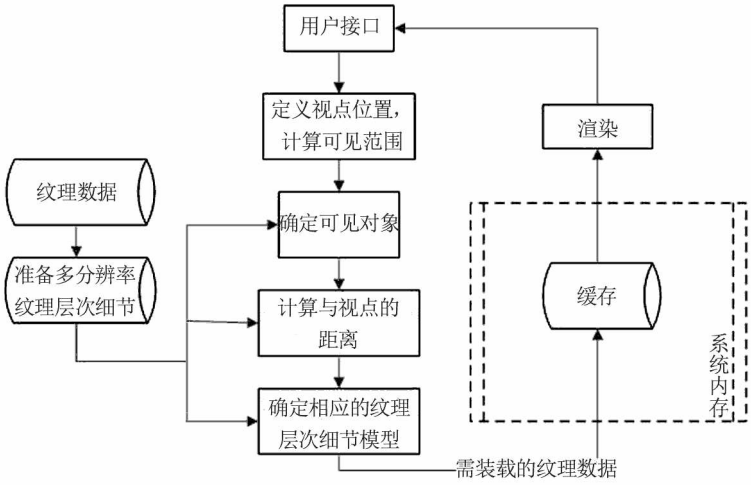

图 1　LOD 纹理映射的技术路线

Fig.1　The technical route of LOD texture mapping

2　LOD 纹理映射技术

LOD 纹理映射技术的基本思想[11]：预先生成一组不同分辨率的纹理集合（LOD 纹理），根据物体与观察者的距离远近，用不同分辨率的纹理来显示物体表面的细节，当物体距离观察者较近时，显示较高分辨率的纹理，当物体远离观察者时，显示较低分辨率的纹理.

2.1　LOD 纹理生成

LOD 纹理生成方法是：每 4 个相邻的纹理像素的均值作为下一层相应的纹理像素，从最高分辨率的图像开始，通过滤波得到低一级分辨率的纹理图像，从而生成分辨率从高到低的一组图像.

实验应用 matlab 的 imresize（A，Scale，Method） 函数来生成 LOD 纹理，Scale 表示图像放大或者缩小的倍数，Method 表示图像放大或者缩小的方式.该函数生成从最高分辨率到最低分辨率的所有纹理，但在实际应用中，不需要生成这么多级别的 LOD 纹理，因此在函数应用过程中需要设置 LOD 级别，如图 2 所示为指定生成 4 个级别的 LOD 纹理.

2.2　LOD 纹理压缩

我们知道管理大数据量的纹理会对应用程序的效能产生影响，导致刷新速率慢、虚拟纹理内存不足等问题.应用 LOD 纹理，首先将大量的纹理数据从磁盘读入计算机中，然后传输到纹理内

存.如果显卡没有足够的内存空间来存储所有使用的纹理,那么就要不断付出从系统到显卡内存的巨大的传输代价.所以如果使用 LOD 纹理,同时又不希望给显卡内存带来过重的负担,就必须考虑使用纹理压缩技术[12],对纹理进行压缩,当使用纹理渲染时再进行解压缩.

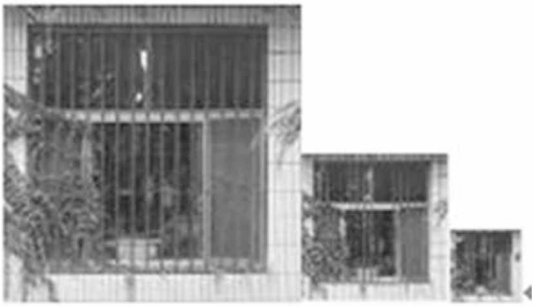

图 2 LOD 纹理

Fig.2 LOD Texture

由于在进行纹理渲染之前对纹理的解压缩操作是在 CPU 上进行的,然后将解压后的纹理数据从 CPU 传输到 GPU 上,这样 CPU 与 GPU 之间的传输带宽就成为绘制的瓶颈.因此实验采用 DXT 压缩方法,该方法是一种可以直接从压缩数据中绘制纹理图像的方法,它能够直接在图形硬件上实现.DXT 压缩纹理可以有效地节省内存空间,降低纹理的传输带宽,提高图形系统的整体性能.根据 alpha 通道,Direct3D 将 DXT 压缩算法分为 DXT1、DXT2、DXT3、DXT4、DXT5,分别对应 D3DFMT_DXT1 ~ D3DFMT_DXT5 共 5 种压缩纹理格式.其中,D3DFMT_DXT1 支持 15 位 RGB 和 1 位 alpha 图形格式,D3DFMT_DXT2、D3DFMT_DXT3 支持 12 位 RGB 和 4 位 alpha,D3DFMT_DXT4、D3DFMT_DXT5 则采取了线性插值方式生成 alpha.图 3 所示为不同 DXT 格式的压缩纹理.

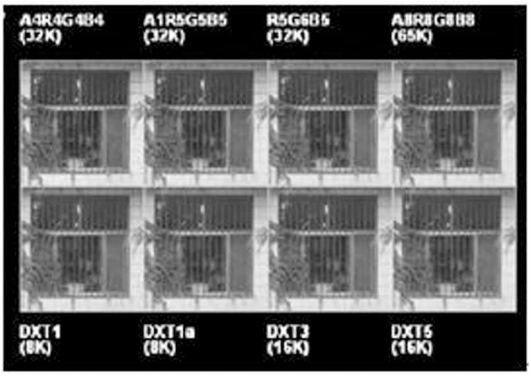

图 3 不同 DXT 格式的压缩纹理

Fig.3 The compressed textures of different DXT formats

2.3 LOD 纹理的调度

纹理的调度是指纹理的载入/载出过程.三维场景中的纹理数据和模型数据是分开存储的,纹理在映射到模型对象之前,必须首先进行纹理下载,即从硬盘加载到主存中,当需要显示纹理时,再将纹理从主存转移到纹理内存.那么生成的 LOD 纹理又是如何调度呢? 主要是通过实时计算模型表面与视点的距离来判断何时加载纹理,如表 1 所示,为不同距离对应的 LOD 级别.

表 1 距离与 LOD 级别的关系

Tab.1 The relationship between distance and LOD level

纹理级别(LOD n)	距离(Range)
LOD1	≤ 100
LOD2	[100,200]
LOD3	[200,300]
LOD4	[300,500]

如表 1 所示,假定生成 4 级分辨率的纹理,每一级分辨率对应一个切换距离(该距离为与模型表面视点的距离),切换距离的范围在 $[near, far]$ 之间,本次实验设置 $near = 1$, $far = 500$.运行结果如图 4 所示.

2.4　LOD 纹理的管理

LOD 纹理的管理是通过定义两个列表来实现的，一个是纹理列表 texList，一个是图像列表 imageList. 首先为当前场景中所有正在使用的纹理创建一个纹理列表，然后在绘制阶段将这个列表加载进纹理内存中. 当场景发生变化，纹理需要更新时，首先将不需要的纹理和图像从列表中删除，应用 pfIdleTex( ) 将纹理从纹理内存中卸载，然后从磁盘读入将要显示的纹理图像，并添加到图像列表和纹理列表进行绑定. 或者直接通过 SubloadTex 进行纹理更新.

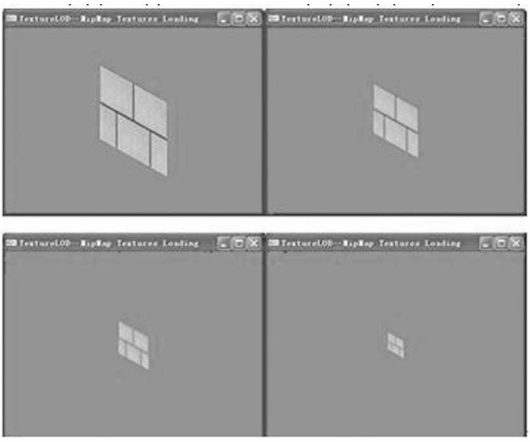

图 4　不同 LOD 纹理的加载

Fig.4　The loading of different LOD textures

3　LOD 纹理映射技术实现

图 5 所示为 LOD 纹理映射的实现过程. 运用 OpenGL Perfermor 实现 LOD 纹理映射时，遵守以下原则：pfPushState( ) 保存当前状态；pfPushIndentMatrix( ) 保存当前矩阵堆栈；执行 OpenGL 命令，进行特效渲染；pfPopMatrix( ) 恢复原始矩阵状态；pfPopState( ) 恢复原始绘制状态.

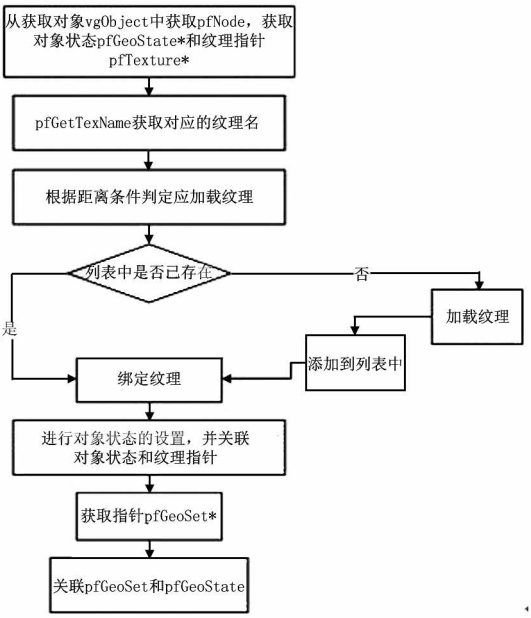

图 5　LOD 纹理映射的实现过程

Fig.5　The implementation process of LOD texture mapping

以下是纹理映射实现的代码：

```
for( int j=0;j<pfGetNum(list2);j++)
{
①获取对象状态以及纹理指针、对应纹理路径名
gstate=(pfGeoState*)pfGet(list2,j);
pfTexture*tex=(pfTexture*) pfGetGStateAttr(gstate,PFSTATE_TEXTURE);
char *path=pfGetTexName(tex);
②从纹理路径名中读取纹理名，根据距离条件加载对应纹理
……
if(距离条件)
  OpenTextures(TexName[i]);
③判断纹理是否在列表中，即判断是否已经加载了对应的纹理
若未加载，则加载纹理 LoadTextures( )，并进行纹理参数设置，绑定纹理；
```

若已加载,则直接返回,使用已加载的纹理.

④对象状态设置,绑定该对象状态

```
pfGStateMode(gstate,PFSTATE_ENTEXTURE,PF_ON);
pfGStateMode(gstate,PFSTATE_ENTEXGEN,PF_ON);
……
pfApplyGState(gstate);
```

⑤关联对象状态和新纹理指针

```
pfGStateAttr (gstate, PFSTATE_TEXTURE, pftex);
……
```

⑥关联 pfGeoSet 和 pfGeoState,以及将 pfGeoSet 加入 pfGeode 中

```
pfGSetGState (gset, gstate);
pfAddGSet(geode, gset);
}
```

图 6 所示为应用上述算法实验的结果.

图 6 实验结果

Fig.6 The experimental results

4 实验结果分析

实验环境:Microsoft Windows XP 操作系统,Intel(R) Core(TM)2 Duo CPU E7300 @2.66GHz 处理器,NVIDIA GeForce 9600 GT 显卡,2.00GB 内存, 1440 × 900 屏幕分辨率.

1)内存.

内存的使用是跟随视点而改变的,纹理根据模型与视点的距离而动态进行加载卸载.例如某一时刻内存使用 27 406 350 Bytes,随着视点推进,某一时刻内存为 28 204 518 Bytes.

2)纹理下载速度.

由于纹理数据基本不大,因此本次实验中纹理下载速度基本保持在 13~18 ms 之间,如图 7 所示.

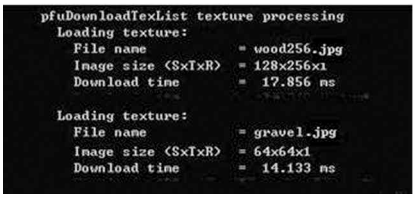

图 7 纹理下载速度

Fig.7 The texture download speed

3)帧率与帧时.

帧率即帧频率,指在单位时间内系统所能渲染出的帧数,单位为帧/秒(fps).帧率与图形系统所需填充的屏幕像素的数量、场景中节点状态的变化次数、渲染场景的复杂程度等有关,一般来说要达到实时渲染的效果,帧率应该保持在 20~30 fps.从图 8 可以看出,本次实验基本保持在这个帧率范围内,图 8 中出现的“尖峰”是漫游时由于视点切换需要重新加载纹理造成的.纹理加载从外存到系统内存,再到纹理内存,需要耗费一定的时间.

帧时(帧时间)表示系统渲染一帧的时间.一般来说,帧率与帧时成反比关系,渲染一帧的时间越短,那么每秒钟系统渲染的帧数就越多.本次实验帧时基本处于 22 msec 左右.

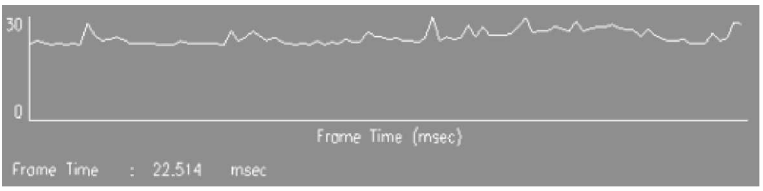

图 8　帧时

Fig.8　The frame time

5　结论

实验表明，基于 LOD 纹理映射技术能够明显提高三维场景的实时渲染速度，在今后三维城市建设、系统开发中可以引用该技术，但本次实验采用的是单线程处理模式，为了加快绘制速率，可以采用多线程方式，充分利用多 CPU 核处理器的并行能力.

参考文献：

[1] PAUL S H, MICHAEL G. Survey of polygonal surface simplification algorithms[R]. School of Computer Science. Carnegie Mellon University,1997.

[2] MÖLLER T, HAINES E, HOFFMAN N. Real-time rendering[J].DBLP,1999,6(10):582-585.

[3] HOFMANN J S. A survey of real-time rendering algorithms[J].CSci,2000:7-14.

[4] 刘学慧，吴恩华.虚拟现实的图形生成技术[J].中国图像图形学报,1997,2(4):205-212.

[5] 齐敏，郝重阳，佟明安.三维地形生成及实时显示技术研究进展[J].中国图像图形学报,2000:269-275.

[6] 淮永建.虚拟场景中实时图形绘制关键技术研究[D].西安：西北工业大学,2002.

[7] 许妙忠.虚拟现实中三维地形建模和可视化技术及其算法研究[D].武汉：武汉大学,2003.

[8] 张雯，王守尊，万强.基于视点的多分辨率纹理选择方法[J].计算机应用,2006,26(11):2 719-2 720.

[9] COHEN-OR D, CHRYSANTHOU Y L, SILVA C T. et al. A survey of visibility for walkthrough applications[J]. Visualization & Computer Graphics IEEE Transactions on,2003,9(3):412-431.

[10] 高丽娜.虚拟现实引擎中可见性裁减问题的研究[D].南京：南京工业大学,2006.

[11] 杜莹，武玉国，游雄.全球虚拟地形环境中 Mipmap 纹理技术研究[J].测绘科学技术学报,2006,23(5):355-358.

[12] 于美娇，姚润泽. 虚拟地形场景中的纹理压缩技术[C]//.第六届全国地图学与 GIS 学术会议,2006:1-5.

[责任编辑：陈小诗　　英文审校：许艺萍]