

SLOPE法线生成算法的 LOD 技术 在地形渲染中的应用

宋双柱, 薛群, 孙立镛

(哈尔滨理工大学 计算机科学与技术学院, 黑龙江 哈尔滨 15008)

摘要: 为了研究 LOD 地形渲染技术在使用法向量进行地形光照等渲染的时候遇到的困难, 解决数据存储与动态计算法线向量的问题, 在分析当前使用的地形渲染技术的基础上, 针对三维地形渲染的常用方法, 提出了一种新的计算法线向量的方式算法, 简化了操作, 并提高了渲染效率。

关键词: 地形渲染; 斜面; LOD; 二叉树

中图分类号: TP 391 **文献标志码:** A **文章编号:** 1007-2683(2009)05-0063-05

Application of LOD-based SLOPE in Terrain Generation

SONG Shuang-zhu, XUE Qun, SUN Li-juan

(School of Computer Science and Technology, Harbin University of Science and Technology, Harbin 150080, China)

Abstract: To research the problem using lighting by normal direction in terrain generation, and save or change the data dynamically quickly, we provide a new idea for the calculation of the normal, using Multi-Resolution Terrain technology and Level of Detail technology and enhancing the generation efficiency and simplify the operation.

Key words: terrain generation; SLOPE; LOD; quad-tree

0 引言

室外场景的实时渲染技术是游戏编程世界中的热点技术。同时它在其它领域也有着同样重要的作用。如 GIS 系统, 飞行模拟系统, VR 系统以及数字地球技术等都离不开室外场景的实时渲染技术^[1]。一个优秀的室外场景实时渲染技术在保证实时性以外还能创造出非常逼真的、有说服力的虚拟自然环境。它除了需要能模拟出各种如雪地、草地、沙漠等地形以外, 还要能模拟出各种树木、杂草以及各种天气效果^[2]。

目前主要的地形渲染技术其中就层次细节

(LOD) 技术, 它是一种符合人的视觉特性的技术^[3-5]。当场景中的物体离观察者很远的时候, 它们经过观察、投影变换后在屏幕上往往只是几个像素甚至是一个象素。因此完全没有必要为这样的物体去绘制它的全部细节, 可以适当的合并一些三角形而不损失画面的视觉效果。对于一般的应用, 通常会为同一个物体建立几个不同细节层度的模型, 这样的技术应用在地形渲染中, 也称之为多分辨率地形 (Multi-Resolution Terrain)。

LOD 算法对场景的处理比较复杂, 但是它可以足够自由地去控制场景渲染, 更加方便的使用显卡的硬件加速功能。而且可以很容易的在场景中组合其他的物体。如树木, 太阳以及粒子系统等, 天空如

收稿日期: 2008-12-16

基金项目: 国家自然科学基金资助的项目 (60173055)

作者简介: 宋双柱 (1970—), 男, 硕士研究生, E-mail: SS2001@yahoo.cn;

孙立镛 (1944—), 男, 教授, 博士生导师。

它可以方便的让观察者以任意的角度去观察场景, 只要让摄影机旋转一定的角度就可以了。

而一个凹凸不平的表面是理所当然的需要更多的三角形去描绘的。

1 基于 LOD 算法的地形简化

LOD 技术是本文研究的基础和实现平台的地形渲染技术. 如图 1 所示。

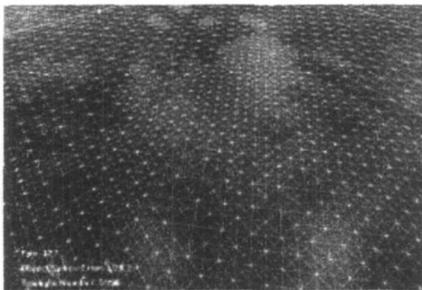


图 1 LOD 处理的地形网格

对于 LOD 中, 这些不同细节层度的网格可以在程序运行前建立的. 也可以是在运行时刻计算生成的. 可以从一个全细节的网格模型出发, 通过一系列简化操作生成低细节层度的模型. 简化操作可以分成 3 种: 顶点删除, 边压缩和面片收缩技术. 通过这样处理后, 可以在特定的场合下选择合适的模型, 而不必每次都选用全细节的模型, 这样可以大大的降低场景三角形数量。

地形作为一种特殊的几何物体, 在运用 LOD 法则的时候有一些特殊的技巧. 因为地形通常是一个规则的矩形网格. 其简化模式可以有两种: 规则的简化和非规则的简化, 规则的简化通常是对这个矩形网格采用自顶向下 (Up-to-Down)、分而治之的策略, 典型的有四叉树和二叉树, 它们从场景的最低细节层度开始, 按需要不断的提高细节. 非规则的简化通常是采用自底向上 (Down-to-Up) 的方法来处理的. 它的实现则通常比较少, 如图 2 所示^[6]。

实现 LOD 算法时, 除了如何对几何物体进行简化以外, 还有一个很重要的问题就是如何决定是否对一个物体进行简化, 或者说在某个时刻该如何决定使用哪个层次细节度的模型来表示物体. 需要建立一个评价系统, 由这个评价系统来决定要对物体简化到何种层度. 这种评价系统通常是视点相关的, 离视点远的物体通常只需要较少的细节, 反之则需要比较多的细节. 除此之外, 物体本身的特性也必须考虑在内. 比如说, 一个平坦的表面只需要很少的三角形就能较好表现出来。

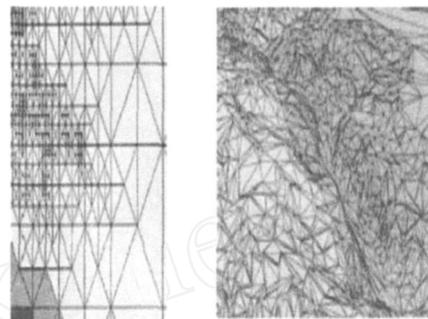


图 2 规则的简化 (左) 和非规则的简化方式 (右)

用 LOD 算法渲染地形的时候, 还有一个很重要的问题就是几何变形 (Geomorphing) 问题, 由于对一些细节的丢弃, 随着视点的移动, 远处原来没有的细节很可能会突然出现, 这种现象也称为“跳出” (“Pop”). 必须消除这种现象, 或者至少要把它控制在可以接受的范围以内^[7]。

由上可知, LOD 算法其实并不很复杂, 本文认为其关键处可概括如下:

1) 数据的存储布局. 数据在内存中的布局必须要方便算法的实现, 同时最好还要降低操作系统缺页中断的次数, 也就是降低内外存之间的数据交换的次数。

2) 如何在生成连续的 LOD 化的地形网格. 在地形 LOD 化过程中, 要让两个由不同层度的细节的区域之间能平滑的过度. 配合灯光技术是本文研究的关键。

3) 节点评价系统. 这个系统必须要使生成的网格能尽量的减少几何形变, 尽量的使画面质量能接近全分辨率时候的地形, 同时还要保证实时性。

2 SLOPE 法线计算方法的原理与实现

在提出基本的算法之前, 为了简单起见, 本文对要渲染的地形做如下的规定: 地形必须是一正方形区域, 同时采样间隔必须均匀。

如图 3 所示, 采用四叉树的概念来描述一个多分辨率地形. 图 3 中的每一个正方形为四叉树的一个节点, 每个节点保存了一定区域的信息, 包括: 中心点的高度, 从整个完整的地形出发, 递归地把地形不断的分割 (Sub-divide) 成相等的 4 个区域, 分割的

深度越大,则得到的分辨率越高.即分割深度每提高一层,采样密度提高一倍^[8].

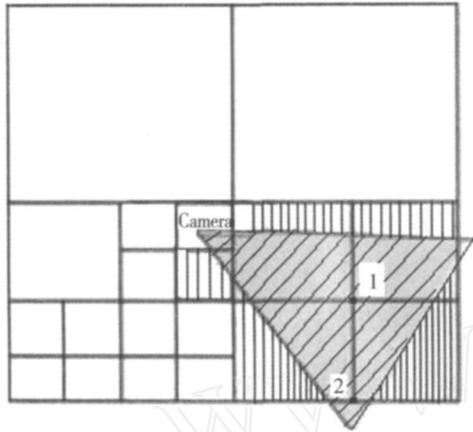


图 3 一个地形的四叉树表示

采用四叉树的概念来表示多分辨率地形有许多优点:一个最直接有效的受益就是裁剪,如图 3 所示,其中用斜线填充的阴影区域为观察者能看到的部分.很容易知道观察者能看到的只是竖线填充阴影区域中的节点 1 和 2,其它白色的节点则根本不需要考虑.因此,可以在节点递归分割的初期只花很少的代价就可直接把这些看不到的区域简单的丢弃掉.

有了地形的逻辑表示后,还要建立一个节点评价系统来判定一个节点何时需要被继续分割,何时被直接丢弃(当这个节点不能被观察者看到的时候,节点将被直接丢弃).如果一个节点没有被丢弃,也不需要继续分割,那么这个节点将被送入渲染 AP 进行图元渲染.

2.1 数据存储

地形数据通常存储在高度图里,在内存的结构即为一个二维数组.二叉树可以有顺序结构和链式结构,同理四叉树也可以采用类似的顺序结构.不同的是这里采用二维数组而不是一维数组.把全分辨率的地形数据存储在一个二维数组中,四叉树节点的信息(9个顶点的信息)可以直接通过索引在数组中读取.同时还要建立一个和这个地形数据数组大小相同的标志数组,这个标志数组指示四叉树节点的状态.如果一个节点需要被继续分割,则把相应的位置标记为 1,否则标记为 0,如图 4 所示,标着问号的表示没有被访问到,(注意没有被访问到的地方的数值是不确定的).

由图 4 的数组易知地形大小要满足要求(对于不满足大小要求的地形,必须把把它分割成满足要

求的大小,然后进行拼接.

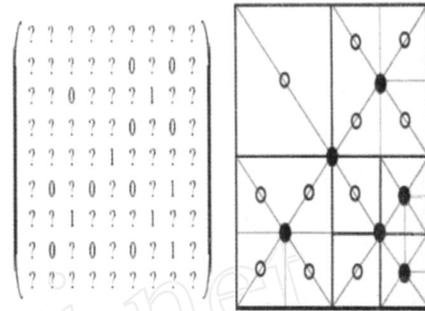


图 4 一个地形记数意图与对应地形

二维数组的存取是按行或者按列的.考虑到观察者在场景中移动时候的区域性,本文尝试使用了一种按区域存储的二维数组,即在物理上把地形数据分成等大小的块,块的大小不能太大,也不能太小.

现在的地形渲染一般都使用高度图来作为数据源,高度图中每一个像素的颜色都代表着一个高度值,就为其 y 值,其 x, z 值为程序生成的规格数据.所以将所有这些顶点依次用三角形带绘制出来,即形成一块地形.

由于高度图中没有包含法线信息,无法进行光照,可以用张图的 R, G, B 来储存法线信息,用 A 来储存高度值.但是这需要为它写一个完善的地形编辑器了,而且这个法线是不可更改的.

如果要做顶点波动的水面,这时又需要用当前顶点的法线去查询 Cubemap 来产生反射效果.这时就必须动态更新法线信息了,如何快速的实时计算,就是我们需要面对的问题, Slope 方法就可以解决此问题^[9].

图 5 为一个网格图,假设有顶点 p00, p10, p20 ..., 首先见图中深色部分,如果要计算 p31 此点的法线.

p00	p10	p20	p30	p40	p50
p01	p11	p21	p31	p41	p51
p02	p12	p22	p32	p42	p52

图 5 SLOPE 计算网格

Slope法线的算法伪代码如下:

```
p[3][1]. normal x = p[4][1]. position y - p[2][1]. position y;
p[3][1]. normal y = - k;
p[3][1]. normal z = p[3][2]. position y - p[3][0]. position y;
p[3][1]. normal normalize(); //归一化
[/code]
```

所以推广到任意点为

```
[code]
p[a][b]. normal x = p[a+1][b]. position y - p[a-1][b]. position y;
p[a][b]. normal y = - k;
p[a][b]. normal z = p[a][b+1]. position y - p[a][b-1]. position y;
p[a][b]. normal normalize(); //归一化
```

程序中 k 值是个调整值,反复调整才能达到最好的效果,如果两个顶点的水平距离较大, k 值应该大点,反之小点,边缘特殊处理:当顶点为 p_{00} , p_{10} , p_{20} , p_{01} 这些边缘顶点的时候,用以上方法计算,必然导致数组越界,所以这里强制定义边缘处顶点法线为 $(0, 1, 0)$.

2.2 网格的渲染

地形网格的渲染最终也是通过一个递归的过程来实现的.首先遍历整个四叉树,当到达四叉树的叶子的时候,即一个节点不再被分割的时候,就可以把这个节点给绘制出来.

本文采用三角形扇 (Triangle Fan) 的方式来绘制节点,这是一种很自然的方式,因为一个节点包含了一个中心点和若干个围绕着中心点的点,这样的排列刚好形成一个三角形扇^[10],如图 6 所示.

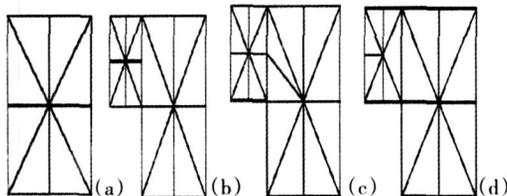


图 6 网格渲染

生成网格的时候,还有一个注意的地方就是两个不同分辨率的节点拼接处会产生 T 形裂缝,如图 6(b) 所示.必须消除这种裂缝,图 6(c) 演示了在拼接地方增加一条边的方法来消除裂缝,图 6(d) 则采用了去掉一条边的方法.相对说,第一种方法更加的

复杂,但是也更加全面,因为拼接处的两个节点的分辨率可以相差任意大.第二种方法则更加简单,它要求拼接处的两个节点的层次差距最多不超过 1.

2.3 网格的生成

在渲染网格之前,必须更新四叉树,生成符合规范的四叉树,在前一小节中曾给出相邻的两个节点的层次最大不能相差 1,否则在拼接的地方会出现裂缝,图 7 给出了一个符合与不符合规范的四叉树例子.

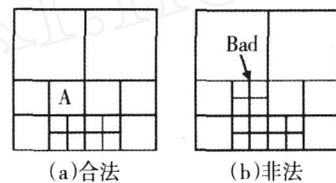


图 7 四叉树例子

通常采用的是两次遍历四叉树的方法,在第一次遍历的时候,生成地形网格,第二次渲染网格,同时消除节点之间的层次差异.

这里采用一种更加有效的方法来生成四叉树——按广度优先的原则遍历四叉树.即一次生成所有属于同一个层次的节点,这样就只需要遍历四叉树一次.可以使用两个队列,一个队列保存着当前正在处理的层次的所有节点,另外一个队列则保存着处理当前层次节点后生成的所有的下一个层次的节点.当处理当前层次节点的时候,把分割生成的下一个层次节点都送入另一个队列种,当处理完所有当前层次队列中的节点以后,就可以进入下一个层次的节点处理.对那些不要继续分割的节点和已经到达最大分辨率的节点,就把它送入渲染 API 进行渲染,不可见的节点则直接丢弃.

上述做法具有许多优点:首先因为每检查一个节点的时候,和该节点层次相同的节点都已经生成.可以通过检查所有和这个节点相邻的节点,看它们是不是存在,如果它们都存在,则可以对这个节点进行继续分割,反之则不能对它进行分割.同时,还可以在第一次遍历四叉树的时候有足够的信息绘制三角型扇,在渲染一个节点的时候,只需要检查分辨率比该节点小的节点.而这些节点在此之前已经全部生成.其次,还无需每次都复位四叉树的状态.

能够生成的使用 SLOPE 法线的效果图,如图 8 所示.



图 8 应用 SLOPE法线的 LOD 地形渲染

3 结 语

本文提出了使用 SLOPE 法线计算方法作为 LOD 的地形渲染技术的支持,使用一种简单有效的计算方法获取灯光渲染等需要的地表法向量,在不需要额外存储空间并且可以动态计算的情况下较好的实现了地面灯光渲染的法向量计算.并详细分析了 LOD 技术的优点,充分的结合了成熟的 LOD 技术,较好的实现了预想效果.

参 考 文 献:

- [1] MAGDA S, KRIEGMAN D, Fast Texture Synthesis on Arbitrary meshes [C]//Proceedings of Eurographics Workshop on Rendering, Leuven, 2003: 82 - 89.

- [2] GUO Yinzhong, WANG Yan. An A synchronous Messaging Model with Time Independent Invocation Based on Distributed Objects [J]. Journal of Communication and Computer, 2006, 3 (5): 33 - 39.
- [3] 潘季亮. 基于 LOD 的大规模真实感室外场景实时渲染技术的初步研究 [J]. 自然科学进展, 2002, 12 (6): 665 - 668.
- [4] 邵艳红. LOD 技术及其算法 [J]. 软件导刊, 2008, 7 (10): 48 - 50.
- [5] 夏旭, 周长胜, 郑直. 多分辨率下的 LOD 地形简化技术研究 [J]. 2008, 23 (4): 38 - 42.
- [6] DAI Hum in LI Zan. Non-synchronous Deformation Effect of Particle in Sheet Metal Forming Process [J]. Computer Aided Drafting, Design and Manufacture, 2005, 12 (5): 14 - 18.
- [7] YANG Jingzhou, ABDELM K. Approximate Swept Volumes of NURBS Surfaces or Solids [J]. Computer Aided Geometric Design, 2005, 22 (1): 1 - 26.
- [8] WANG Bin, WANG Wenping, YONG Junhai, et al. Synthesizing 2D directional Moving Texture [C]//Proceedings of the 21st Spring Conference on Computer Graphics, Budmerice, 2005: 169 - 175.
- [9] CHEN Zhuoning, ZHANG Fen. Collaborative Design in PDM/3D CAD Integrated Environment [J]. Wuhan University Journal of Natural Sciences, 2006, 11 (3): 642 - 648.
- [10] 彭群生, 胡国飞. 三角网格的参数化 [J]. 计算机辅助设计与图形学学报, 2004, 16 (6): 25 - 30.

(编辑:温泽宇)

(上接第 62 页)

了可执行文件,制作了帮助文件且打包成安装文件.该程序可以通过快捷键来快捷方便地截取整个屏幕、应用程序窗口和选定矩形区域的图像,在截取图像后可根据先前用户的设置将截取的图像保存至文件、发送到剪贴板或保存到指定文件夹并自动连续命名.其实,通过异型窗体不仅可以确定矩形、圆形、椭圆和三角形等截取区域,稍加改变就可以用来确定任意多边形区域或任意闭合曲线区域.其中的图像截取和光标截取技术也可应用在中多媒体网络教室和网络监控等软件开发中.

参 考 文 献:

- [1] 肖宁. 利用 VB 6.0 和 Windows API 函数制作屏幕抓取程序 [J]. 辽宁师专学报, 2006 (2): 45 - 87.

- [2] 高西宽, 刘泊, 马熙源. 基于 VB 与 MATLAB 混合编程的数字水印软件设计 [J]. 哈尔滨理工大学学报, 2008 (4): 18 - 20.
- [3] 赵仕元. Visual Basic 6.0 对 API 函数引用的方法 [J]. 机械工程与自动化, 2008 (3): 173 - 174.
- [4] 马龙, 王慕坤. 蓝牙技术链路级安全机制的研究 [J]. 哈尔滨理工大学学报, 2003 (4): 15 - 17.
- [5] 袁亚丽. 浅谈 API 函数在 VB 开发中的应用 [J]. 河北北方学院学报, 2007 (5): 47 - 49.
- [6] 常晓波, 刘颖. Visual Basic 6.0 高级编程 [M]. 北京: 清华大学出版社, 2003.
- [7] 郭勇. Windows API 接口大全 [M] 北京: 电子工业出版社, 1995.
- [8] 郝智泉, 吕汉兴, 程臻. 基于局域网的视频图像传输与监视系统 [J]. 计算机应用研究, 2003 (6): 62 - 63.

(编辑:付长纓)