

文章编号:1673-0291(2007)04-0124-04

# 视景仿真系统中三维地形的实时生成方法

邓 野,方卫宁,田生彩

(北京交通大学 机械与电子控制工程学院,北京 100044)

**摘 要:**三维地形是虚拟自然环境中不可缺少的因素,也是虚拟仿真领域中视景系统的重要组成部分.三维地形的巨大数据量一直是视景仿真研究中的难点与热点.本文提出用函数曲面来保存地形顶点数据,在渲染开始前预先载入和计算生成地形数据,而在渲染阶段则利用视点相关LOD(Level of Detail)模型来简化渲染数据,实现大规模地形网格的绘制.实验结果表明,该方法能有效减少实时绘制时的计算量,提高绘制速度,较好地保证了大规模三维地形的实时生成效果.

**关键词:**三维地形;函数曲面;LOD;三角网格

**中图分类号:**TP391.41

**文献标志码:**A

## A Real-Time Rendering Method of 3D Terrain in Visual Simulation System

DEN G Ye, FANG Wei-ning, TIAN Sheng-cai

(School of Mechanical and Electronic Control Engineering, Beijing Jiaotong University, Beijing 100044, China)

**Abstract:** 3D terrain is an indispensable factor in virtual natural environment and an important part of visual simulation system. The tremendous data of 3D terrain are always the difficulties and hotspots of visual simulation. In this paper, the vertex data of terrain were saved by using the function of curved surface. The data of terrain have been loaded and calculated before rendering. During the rendering, the data were reduced by view-dependent LOD (Level of Detail) model. This algorithm can carry out the rendering of large scale 3D terrain grid. The experimental results show that this algorithm can effectively reduce the computing amount of real time rendering, improve the speed and gain a satisfactory rendering result.

**Key words:** 3D terrain; function of curved surface; LOD; triangle grid

三维地形是虚拟自然环境中不可缺少的因素,也是虚拟仿真领域中视景系统的重要组成部分.由于三维地形通常与虚拟环境中物体的运动直接有关,更由于它的数据量庞大,因此在建模和实时显示两个方面都有较高的要求.人们在不断提升硬件处理能力的同时,也努力研究如何通过软件算法和技术来降低处理的数据量,以实现虚拟效果和硬件能力之间的平衡.

目前,大规模地形快速绘制技术的研究主要分

为三类:基于真实地形数据的地形生成及实时显示技术、基于分型技术的视景仿真技术、基于数据拟合的地形仿真技术<sup>[1]</sup>.近年来,基于真实地形数据的地形生成及实时显示技术的研究发展快速,其核心思想是通过减少实时绘制的顶点数据量来加快绘制速度.学者们提出了许多用于大规模地形快速绘制的技术,例如采用视截体(view-frustum)对不可见地形区块进行裁剪的简单四叉树、基于多细节层次(LOD)的模型等<sup>[2]</sup>.本文作者在结合现有技术的基

收稿日期:2006-03-15

作者简介:邓野(1979—),女,江西南昌人,博士生. email: dy945207@126.com

方卫宁(1968—),男,浙江杭州人,教授,博士,博士生导师.

基础上,提出了一种基于函数曲面和规则三角网格视点相关 LOD 技术的大规模地形实时生成绘制方法,提出用函数曲面来保存地形顶点数据,在渲染开始前预先载入和计算生成地形数据,而在渲染阶段则利用视点相关 LOD 模型来简化渲染数据,实现大规模三维地形的生成.它能够有效减少实时绘制时的计算量,提高绘制速度,较好地保证了大规模地形的实时生成效果.

1 基于函数曲面的地形数据描述

三维曲线和曲面可以由一组用户给定的数据点生成,也可以由一组给定的数学函数生成.当用函数描述曲面时,可以根据曲面上有限数量的点坐标来获得任意多个该表面上的三角面片以及顶点的法向量.不但可以大大减少模型的数据存储量,而且可以准确地定义模型和还原模型信息.表示曲面的函数方法主要有二次曲面、柔性物体、样条表示、三次样条插值、贝塞尔、B-样条、有理样条等等,它们被大量用于各种造型软件,但因其在线计算量较大,在需要实时渲染的环境中应用较少.如果把渲染模型以函数参数形式存储,在渲染之前用函数生成所需模型的初始渲染数据,这样就可以在不影响渲染系统性能的前提下发挥函数曲面的优势.

选用 Cardinal 样条描述地形曲面. Cardinal 样条是类似于 Hermite 样条的插值分段三次曲线,1 个 Cardinal 样条完全由 4 个连续控制点给出,中间 2 个控制点是曲线端点,其他 2 个点用于计算终点斜率.如图 1 所示.

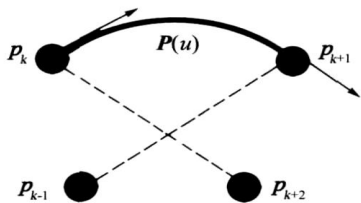


图 1 Cardinal 样条原理图

Fig. 1 Cardinal spline

设  $P(u)$  是 2 个控制点  $P_k$  和  $P_{k+1}$  间的参数三次函数式,则从  $P_{k-1}$  到  $P_{k+2}$  间的 4 个控制点用于建立 Cardinal 样条段的边界条件为

$$\begin{cases} P(0) = p_k \\ P(1) = p_{k+1} \\ P'(0) = \frac{1}{2}(1-t)(p_{k+1} - p_{k-1}) \\ P'(1) = \frac{1}{2}(1-t)(p_{k+2} - p_k) \end{cases} \quad (1)$$

式中,  $t$  是张力参数,用来控制 Cardinal 样条与输入控制点间的松紧程度,通过设定不同的  $t$  来产生不同弯曲程度的曲线段.函数的矩阵形式为

$$P(u) = [u^3 \quad u^2 \quad u \quad 1] \cdot M_c \cdot \begin{bmatrix} p_{k-1} \\ p_k \\ p_{k+1} \\ p_{k+2} \end{bmatrix} \quad (2)$$

$$M_c = \begin{bmatrix} -s & 2-s & s-2 & s \\ 2s & s-3 & s-2s & -s \\ -s & 0 & s & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad (3)$$

式中,  $s = (1 - t)/2$ .

如图 2 所示,已知曲面  $L$  上的 4 个控制点  $P_1, P_2, P_3, P_4$  及他们的切线.曲面上  $P(a, b)$  点的高度计算如下:设  $a, b$  分别是  $x$  和  $y$  方向上的坐标值,高度值是  $z$  坐标值,根据式 (2) 计算曲线  $P_1 P_2$  上  $x$  坐标为  $a$  的点坐标  $P_{u1}$  以及曲线  $P_3 P_4$  上  $x$  坐标为  $a$  的点坐标  $P_{u2}$ .按线性插值公式计算这两点处  $y$  方向上的切线  $P_{y-u1}$  和  $P_{y-u2}$ .最后由式 (2) 计算曲线  $P_{u1} P_{u2}$  上  $y$  坐标为  $b$  的点的高度值.

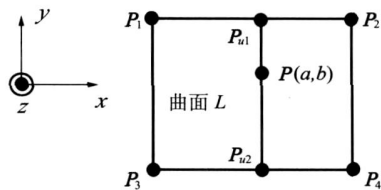


图 2 曲面表示

Fig. 2 Curved surface

2 地形的混合生成方法

一般来说,实时渲染应用中通常保存和载入足够精度的初始地形数据,渲染阶段运用视截体裁剪和各种 LOD 技术来简化渲染数据.对于大规模地形来说,这种方法要保存和载入大量地形顶点数据,由于硬盘存储器和内存之间传输速率的限制,以空间换时间的目标和方法往往得不到很好贯彻.本文作者提出用函数曲面来保存地形顶点数据,在渲染开始前预先载入和计算生成地形数据,而在渲染阶段则利用合适的方法来简化渲染数据.实际上地形数据的生成被分为了两个独立的阶段.

2.1 初始地形数据的创建

存储模型时,选定合适的分块大小作为最小可见精度时的网格密度,对每个网格内部采用 Cardinal 样条函数进行定义,并且以此网格精度定义纹理和纹理坐标.因为采用规则三角网数据,每个分块在

队列中的索引隐含了它们的位置,对单个分块而言只需保存顶点的高度信息、控制信息和纹理信息.一般来说最高精度与最小精度的级数相差越大,存储数据量可成倍减少的越多.生成地形数据时,根据最小可见精度分块内所保存的控制参数创建最高精度情况下的所有顶点数据.尽管此法存在定义纹理贴图的灵活性较差,只能针对选定的最小可见精度网格点进行纹理坐标定义的缺点,但对于大规模地形来说,贴图纹理坐标定义往往不需要很精确,而且在需要的情况下,也可以通过使用多层贴图多回渲染方法来解决.

在载入数据时,一次性计算网格块内的目标顶点的坐标和法线值,由于这种计算是在渲染开始之前完成的,所以实际上是离线进行的,并没有牺牲渲染时间.预处理阶段,可以根据误差限设置,计算每个网格实际所需的最高精度.建模时,构建一个最高精度的模型,然后构建一个索引表,记录一组连续 LOD 精度的模型顶点索引;渲染时,计算视点离模型的距离,然后选用一个合适分辨率的顶点索引,进行渲染.通过这种地形载入方法,可以大大减少地形初始数据载入和生成过程对系统性能的影响.另外,采用此种方法,也可以用高度场等原始数据直接作为地形数据,实时生成动态地形.

## 2.2 渲染数据的简化

在 Lindstrom 等学者提出的基于规则网格的连续 LOD 地形模型实时生成算法<sup>[3]</sup>和吴亚东等学者提出的基于连续细节层次的地形动态生成技术<sup>[2]</sup>的基础上提出两阶段地形动态生成技术,即:将地形以最小可见精度划分为网格阵列,根据屏幕误差确定网格阵列中每个网格的 LOD 级别,然后按照 LOD 级别对网格进行三角合并和拆分.它包括:

### (1) 最小可见精度的计算

最小可见精度时的网格密度由屏幕分辨率和远裁剪面距离共同决定,即

$$\frac{d_s}{w_s} = \frac{d_t}{2 \times D_t \times \tan(\theta/2)} \quad (4)$$

式中,  $d_s$  是网格的像素宽度,  $w_s$  是屏幕的像素宽度,  $d_t$  是网格的逻辑宽度,  $D_t$  是视点离网格的距离,  $\theta$  是视截体的横向视角.当  $d_s < 1$  时,表示该网格已经不能被分辨,而当  $D_t$  大于远裁剪面时,网格将被裁减掉,所以在这种条件下可以求得  $d_t$  的值,从而确定最小可见精度下的地形网格密度.

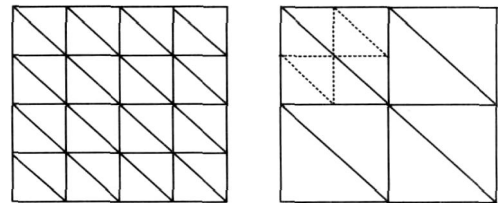
### (2) 计算网格的 LOD 级别

LOD 级别的确定采用基于屏幕误差 LOD 等级计算方法.该方法认为地形崎岖程度和地形与视点

的距离两个因素都会影响像素投影点的屏幕误差.换句话说,当地形过于崎岖,或者距离视点较近,都需要提高该地形的精度等级.这种方法并没有采用层次四叉树的方法来迭代生成地形三角形网,而是对所有可见范围内的最小可见精度网格进行相同的 LOD 计算.选取的最小可见精度和最高精度相差级别越大,效果就越好.基于网格进行 LOD 计算的实质就是大大减少实时计算投影点误差的计算量.

### (3) 网格内三角网的生成

网格内三角网仍然是规则网结构(如图 3 所示),当图 3(a)的精度降到图 3(b)精度时,图 3(a)中的每 4 个方格合并成了 1 个大方格,三角形数从 8 个减少到 2 个.



(a) 原始网格

(b) 简化网格

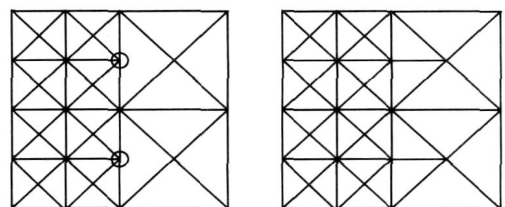
图 3 精度相差 1 的两个网格

Fig. 3 Two grids of precision 1

由于采用了规则的数据结构,当网格的最高精度确定后,网格内最高精度下的顶点数量和排列都是稳定的,网格内的各种 LOD 等级下的顶点排列也都是相同的.因此可以预先确定索引表,并使用三角带形式,类似于递进网格的方法.当使用显卡设备内存时,这种方法带来的性能优势是很明显的,同时很容易用着色器来实现.

## 2.3 几何裂缝的消除

当相邻网格的 LOD 精度不相同,两个子网格可能会出现裂缝图 T 接头(如图 4(a)所示),但是网格内的三角网不存在裂缝问题.因此需要在网格接缝处通过修改精度等级较高一端的网格来保持对低精度网格的无缝拼接.图 4(b)是采用添加一条边的方式来消除 T 接头,弥补裂缝.



(a) 存在 T 接头

(b) 加边消除裂缝

图 4 接头裂缝修改

Fig. 4 Improvement of cranny in connection

这种接缝修改会破坏三角网索引的规则性,使得必须在网格及其周围网格的 LOD 等级发生变化时更新一次索引队列,但是考虑到 LOD 等级的连续变化特性,只要保证网格与其周围网格的 LOD 等级之比相差为 1,那么仍然可以预先计算所有可能的索引排序.每个 LOD 等级下,所有可能的数为  $2^4$ ,即 16 种.对于这个数量级的内存量是不大的,因此可以在线用查表方式来更新三角网索引队列.

### 3 实验分析

图 5 是应用本文作者提出的三维地形实时生成方法得到的 2 张渲染结果图.主机配置为 AMD 公司 Athlon 2500 + CPU,512M 的 DDR333 内存以及 Nvidia 公司的 GeForce4 MX440 (64M 显存)图形加速卡.该图像中共渲染了 156 278 个三角形.采用分页地形管理以及基于贴图进行网格分组的多回渲染技术的情况下,平均帧速约为 55 帧/s (顶点、颜色及纹理缓冲均位于显存),较大地提高了绘制速度,满足了实时交互的要求.图 5 中不同灰度的部分代表不同精度的网格,可以看出,不同精度的网格边缘过渡平滑,较好的消除了几何裂缝,真实地再现实际

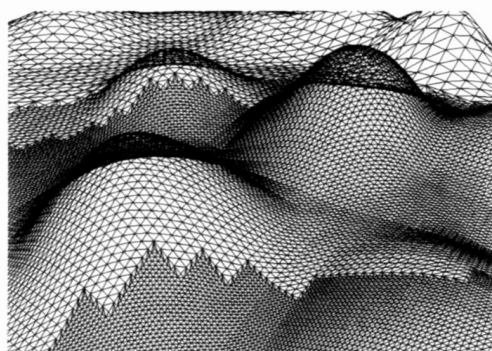
地形的变化.

### 4 结论

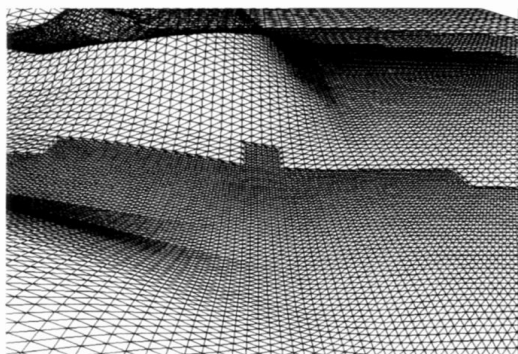
用函数曲面来保存地形顶点数据,在渲染开始前预先载入和计算生成地形数据,而在渲染阶段利用视点相关 LOD 模型来简化渲染数据的方法,可实现大规模地形网格的绘制.实验结果表明,此方法在保持地形场景逼真的情况下,较大地提高了绘制速度,满足了实时交互的要求,可在普通微机上实现列车视景仿真系统的大面积三维地形的快速生成.

### 参考文献:

- [1] 王源,刘建永,江南,等.视点相关实时 LoD 地形模型动态构网算法[J].测绘学报,2003,32(1):47-52.  
Wang Yuan, Liu Jian-yong, Jiang Nan, et al. A Dynamic Triangulation Algorithm for the View-Dependent and Real-Time LoD Model of Terrain[J]. Acta Geodaetica Et Cartographica Sinica, 2003, 32(1):47-52. (in Chinese)
- [2] 吴亚东,刘玉树.基于连续细节层次的地形动态生成技术[J].北京理工大学学报,2000,20(5):15-22.  
Wu Ya-dong, Liu Yu-shu. Dynamic Generation of Continuous Level of Detail for Terrain[J]. Journal of Beijing Institute of Technology, 2000, 20(5):15-22. (in Chinese)
- [3] Peter Lindstrom, David Koller, William Ribarsky, et al. Real Time, Continuous Level of Detail Rendering of Height Fields[C]. Proceedings of SIGGRAPH '96, Los Angeles: Siggraph, 1996:109-118.
- [4] Hoppe H. Smooth View2 Dependent Level-of-Detail Control and Its Application to Terrain Rendering[C]. Proceedings of SIGGRAPH '97, Los Angeles:Siggraph, 1997:189-198.
- [5] Willem H. Fast Terrain Rendering Using Geometrical MipMapping [EB/OL]. [2004-05-01]. <http://www.flipcode.com/tutorials/geomipmaps.pdf>.
- [6] 李之棠,郑晓颖,马昊.不规则三角网数据结构的研究[J].微型电脑应用,2000,16(11):20-22.  
Li Zhi-tang, Zheng Xiao-ying, Ma Hao. A Study on the Data Structure of Triangulated Irregular Network[J]. Microcomputer Applications, 2000, 16(11):20-22. (in Chinese)
- [7] Donald Hearn, Pauline Baker M. 计算机图形学[M].第3版.蔡士杰译.北京:电子工业出版社,2005.  
Donald Hearn, Pauline Baker M. Computer Graphics with OpenGL[M]. 3rd ed. Cai Shijie transl. Beijing: Publishing House of Electronics Industry, 2005. (in Chinese)



(a) 地形起伏



(b) 地形平缓

图5 混合方法渲染地形网格

Fig.5 Rendering the terrain grid