

大规模地形漫游中动态 LOD 算法研究

曹 敏, 杨长兴, 杨 炼

(中南大学 信息科学与工程学院, 湖南 长沙 410083)

摘 要:大规模的地形渲染技术一直是图形学里的热点问题之一。它在 GIS、飞行模拟器、视频游戏里有重要的作用。大规模地形渲染的两个主要问题是地形数据存储问题和三角形数目问题。针对 3D 视频游戏, 文中采用数据分块、局部数据显示以及视点相关的裁减策略来控制数据显示量, 使用了一种基于四叉树的 LOD 算法来解决大规模地形渲染中的三角形数目问题。实验结果表明综合使用上述方法, 有效地减少了显示数据计算量, 能满足 3D 游戏场景的交互式漫游的实时性要求。

关键词:LOD; 四叉树; 实时地形简化; 视点相关

中图分类号:TP391.41

文献标识码:A

文章编号:1673-629X(2008)10-0187-03

A Study of Dynamic LOD Arithmetic for Navigation of Large-scale Terrain

CAO Min, YANG Chang-xing, YANG Lian

(School of Information Science and Engineering, Central South University, Changsha 410083, China)

Abstract: The large scale terrain rendering is a hot issue in the computer graphics research field. It plays an important role in GIS (geographic information system), flight simulator and video game. The main two problem of the large scale terrain rendering are how to hold the terrain data and how to reduce the large amount triangles. For video game solution, to reduce computing, use view-dependent clipping policy, detach terrain data to some block and compute the data only for display, at the same time, use a quad-tree based LOD algorithm to reduce the triangles count. Experiments demonstrate that the foregoing method effectively reduces the computational time, sufficient for real-time interactive navigation over a scene of a 3D game.

Key words: LOD; quad-tree; real-time terrain simplification; view-dependent

0 引 言

大规模地形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。

在过去几十年里, 尽管图形硬件技术已经有了飞速发展, 但仍然不能满足大规模三维场景的可视化的需要。所以, 模型简化技术, 多分辨率表示和 LOD (Level Of Detail) 技术成为近年来研究的热点。现在的 3D 游戏场景越来越复杂, 由于受计算机内存大小或者操作系统管理能力的限制, 所有场景数据常驻内存是不现实的。

玩 3D 游戏实际上就是一个在游戏世界里实时漫游的过程, 介于场景的复杂性, 首先必须解决数据调度问题, 任意时刻只调度与视点相关的局部数据进行显示, 以此减少系统开销; 其次必须解决简化地形显示的问题, 在不影响视觉效果的情况下减少每帧绘制三角形数目, 提高显示速度。

数据调度算法常采用数据分块、局部数据显示以及视点相关的裁减策略来控制数据显示量。地形简化已有多种方法, 针对规则地形网格模型的简化, Lindstrom 等人引入受限四叉树 (restricted quad-tree triangulation, RQT) 概念, 解决了四叉树三角化的空洞问题, 简化效果较好^[1]。实时优化自适应地形格网 (ROAM) 由 Mark Duchaineau 等人提出^[2], 它通过建立了三角形二叉树结构, 并对三角形进行分裂与合并操作来生成地形连续 LOD 模型。文中在研究和总结现有动态 LOD 算法的基础上, 采用数据分块、数据动态调度和裁减技术来实现一个 3D 视频游戏中大规模地形的实时漫游。

收稿日期: 2008-01-04

基金项目: 湖南省自然科学基金资助项目 (06JJ5131); 湖南省教育科研资助项目 (07C388)

作者简介: 曹 敏 (1982-), 男, 硕士研究生, 研究方向为计算机图像处理, 3D 游引擎; 杨长兴, 教授, 研究方向为虚拟现实技术。

1 数据调度算法

1.1 数据分块处理

对数据进行分块处理,将原始地形数据分为 $L \times K$ 块数据,每块数据格网大小是 $(2^n + 1)$,相邻两块地形数据之间共用一条边,以保证数据绘制的连续性。如果分块地形大小不满足 $(2^n + 1)$,则可以通过将原始地形内插为 $(L \times 2^n + 1) \times (K \times 2^n + 1)$ 来保障。文中采用 33×33 的地形分块策略。纹理按照与地形相同坐标范围进行分块处理,为了确保显示速度,对纹理进行内插以保证纹理长宽是 $2^n \times 2^n$,同时对每个分块纹理数据建立纹理金字塔。

1.2 基于视点的选择性绘制

为了控制显示数据块的数据量,文中采用与视点相关的数据调度策略:设视点位置为 P ,视点到远截面的距离是 R 。首先确定以视点为中心,半径为 R 的圆相交的数据块,如图 1 所示。对于在圆内的数据块则利用视景体进行裁减,对于在视景体的数据块,则调入内存进行绘制。游戏中,一般情况下视点的变化是连续的,因此当移动视点时,需要切换的数据量很小,对速度影响不大由于数据作了分块处理,只需对分块数据作一次裁减判断,减少了裁减判断时间。裁减采用包围盒的方式进行,包围盒一般有矩形、正方形和球体三种。文中采用正方形包围盒在整个节点的中心位置(长、宽和高各一半的位置)建立包围盒,包围盒的半径是数据块长、宽、高最大值的一半,裁减采用上、下、左、右、远五个裁减面,如图 1 所示。

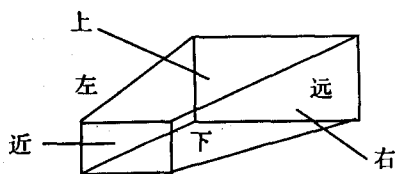


图 1 裁减策略

通过将数据显示范围控制在与视点距离为 R 的圆内以及在圆内进行数据裁减策略,有效降低了每次显示的数据量,提高了显示速度。

2 动态 LOD 算法研究

2.1 相关研究

在过去的几年中,已经有相当的数量的实用的算法被开发出来。Bryan Turner 在他的论文中提到,LOD 地形法则可以由三篇优秀的论文来概括^[1~3],Hoppe 描述了一个 Progressive Mesh 的模型,它是使用自底向上的模式^[3]。Lindstrom 则使用了一种基于四叉树的数据结构,他用四叉树递归地把一个地形分割

成一个个小块(tessellates)并建立一个近似的高度图,Duchaineau 描述了一个基于二元三角树结构的法则 ROAM(实时优化自适应网格)。

2.2 基于四叉树的 LOD 模型

在每个分块地形内部,采用自顶向下的四叉树方法进行地形分割。如图 2 所示,图 2 中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。根据地形条件,将地形不断分割成为四个相等区域,分割深度越大则得到的地形分辨率高。

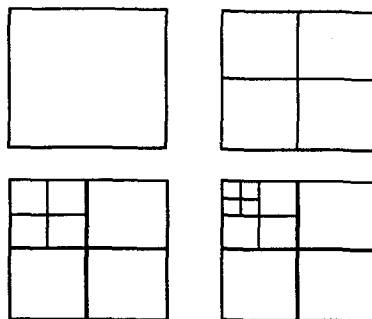


图 2 节点分割示意图

采用自顶向下的绘制方式,必须给出一个评价准则,判断何时停止对节点进行分割关于评价准则,目前常用的是屏幕投影像素误差方式^[4],针对 3D 视频游戏地形,文中采用如下的评价方式^[5]来设置评价准则:

$$f = \frac{l}{d \times r \times C \times C_2} < 1$$

l 是视点到节点的距离; d 是节点大小; r 是节点粗糙程度; C 是可调节系数; C_2 为粗糙度调节因子。当 $f < 1$ 时,需要继续分割节点。这个评价准则含义是:节点离视点越近,节点粗糙程度越高,需要用高分辨率地形来显示。

2.3 裂缝消除

当绘制节点时,相邻不同分辨率节点之间会产生 T 型裂缝,必须消除这种裂缝常用的办法有两种:一种是在两个不同分辨率节点之间加上一条边,如图 3(b)所示;另一种方法是在不同分辨率格网之间减去一条边,如图 3(c)所示。从理论上来说,第一种方法更加全面,因为相邻两个节点层次可以是任意级别,但是这样实现起来比较复杂,绘制效率较低。第二种方法比较简单,只要满足相邻节点层次差别不超过 1 就可以,文中采用第二种方法来进行裂缝消除。

采用了地形分块技术,除了考虑同一块地形内部的接边问题外,还必须考虑相邻分块地形之间的接边问题。为此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,标着问号的表示没有被访问到。如

图 4 所示。图 4 中,黑点表示需要继续分割节点,空心点表示不需要分割节点。

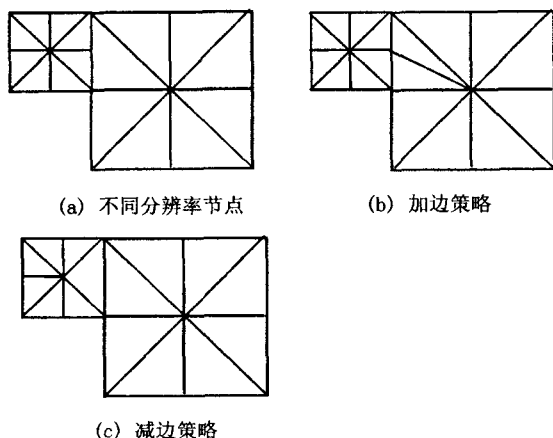


图 3 裂缝消除策略

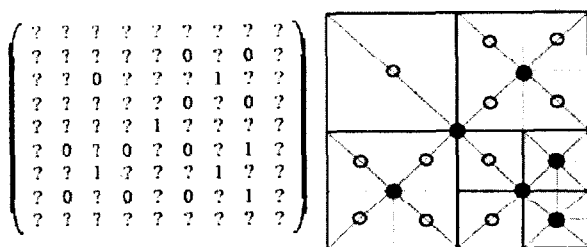


图 4 地形标记数组示意图

在渲染网格之前须更新四叉树,生成符合规范的四叉树,前面曾提到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 5(a)给出了一个符合规范的四叉树例子,图 5(b)给出了一个非法的四叉树例子。为了保证相邻节点分辨率不超过 1,制定分割规则为:如果当前节点的上下左右 4 个节点都已经参与分割,并且当前节点满足分割公式要求,则本节点才可参与分割;否则,直接绘制本节点。

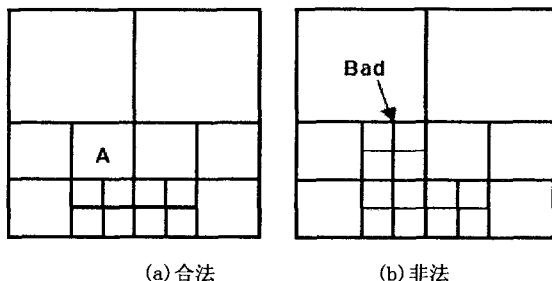


图 5 符合规范的和不符合规范的四叉树例子

对于每个分块地形,初始节点都认为是可分割节点。对分割过程中任意节点,其上下左右节点坐标:

$$\text{Node } L = (n\text{Row}, n\text{Col} - d/2)$$

$$\text{Node } R = (n\text{Row}, n\text{Col} + d/2)$$

$$\text{Node } T = (n\text{Row} - d/2, n\text{Col})$$

$$\text{Node } B = (n\text{Row} + d/2, n\text{Col})$$

其中, $n\text{Row}$, $n\text{Col}$ 分别是当前节点在分块地形中

的行、列坐标; d 是当前网格大小。对于初始节点, $d = n\text{Row} = n\text{Col} = 16$, 以后每分割一级节点, d 减半。

上述公式中上下左右节点中行、列坐标有可能行列值小于零或者大于地形分块大小。为了保障分块地形之间数据接边的完整性,此时需要判断该分块地形的上下左右分块的地形对应节点是否已经分割。

2.4 广度优先算法

通常地形的渲染,采用的是两次遍历四叉树的方法,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。程序使用两个队列,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(简单交换两个队列就可以了)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,就把它们送入渲染 API 进行渲染。不可见的节点则直接丢弃。

3 纹理 LOD 技术

当地形叠加纹理时,由于每块纹理数据量比地形大,因此如果全部载入最精细纹理,则绘制速度明显降低;同时,全部采用最细节纹理显示所有地形也不符合人的视觉习惯。为此文中采用纹理与视点相关策略,根据视点与节点距离确定纹理级别。视点离节点越近,则纹理级别越低,细节越丰富;视点离节点越远,纹理级别越高,细节越少。纹理级别采用如下公式:

$$l/2d < n\text{Level}$$

其中, l 是视点到数据块中心点的距离; d 是当前网格的大小; $n\text{Level}$ 是当前纹理的级别。

当视点移动时,为了避免频繁调度纹理,采用对上述公式所采用的近似的原则——如果上一次绘制时的纹理级别与当前绘制时的纹理级别相差为 1,且上一次绘制的纹理级别低于本次需要的纹理级别,继续使用当前纹理进行绘制;否则采用调度新级别的纹理数据进行绘制。通过这样一种策略,可以进一步减少每次绘制时调度纹理的数据量,提高显示速度。

4 实验和结论

采用上述方法,使用 Visual C++ 6.0 和 direct3D

(下转第 193 页)

4 特点及意义

基于 VML 和脚本技术的 WebGIS 具有一次开发永久受用的特点,只需将需要发布的数据转换为 VML 格式并嵌入 html 文件,然后引入脚本库,就可以实现特定地图的 WebGIS 应用。同时这种 WebGIS 无需在客户端安装插件,简便可靠。此外由于数据只需传输一次,大大提高了网络的运行效率和减轻了服务器的负担。当然,由于目前只有 IE 浏览器支持 VML 的显示,因此面向的客户群体有一定的局限。

5 结束语

文中研究的 WebGIS 开发方法,旨在利用一种标记语言来替代传统的 WebGIS 开发过程。目前基于 XML 的可用于描述矢量图形的标记语言有多种,但利用这些标记语言进行 WebGIS 开发还很不成熟。利用 XML 可以对复杂的数据结构进行统一描述,并且利用现有的网络协议就能进行传输^[7],同时对数据的处理

也有标准化的方法可循,这样大大提高了开发效率,也使开发成本得以降低。

参考文献:

- [1] 卓 泳. WebGIS 技术剖析[EB/OL]. 1998. <http://www.gisforum.net/show.aspx?cid=27&id=801>.
- [2] 夏立民,王 华. 基于 VML 的矢量图形动态生成过程的研究[J]. 计算机技术与发展,2006,16(11):218-221.
- [3] 美洲豹. Thinking in VML[EB/OL]. 2004. <http://www.itlearner.com/code/vml/index.html>.
- [4] W3C. Vector Markup Language (VML)[EB/OL]. 1998. <http://www.w3.org/TR/NOTE-VML>.
- [5] 王 沛,冯曼菲. 征服 Ajax Web 2.0 开发技术详解[M]. 北京:人民邮电出版社,2006.
- [6] ESRI. ESRI Shapefile Technical Description[R]. An ESRI White Paper. USA:ESRI,1998.
- [7] 朱渭宁,黄杏元,马劲松. XML—WebGIS 发展的解决之道[EB/OL]. 2003. http://www.gisky.com/Article_Show.asp?ArticleID=371.

(上接第 186 页)

- [5] IETF RFC2866. RADIUS Accounting[S]. [s.l.]:[s.n.], 2000.
- [6] IETF RFC3748. Extensible Authentication Protocol (EAP)[S]. [s.l.]:[s.n.],2004.
- [7] IETF RFC4016. Protocol for Carrying Authentication and Network Access (PANA)[S]. [s.l.]:[s.n.],2005.

(上接第 189 页)

做了一个 Demo。测试 Demo 数据大小为 $8k \times 8k$ 格网,每个格网 2 个字节。纹理大小 $20\,000 \times 20\,000$ 格网,每个像素 3 字节。Demo 数据采用 33×33 格网大小进行数据分块处理。预处理时,纹理数据建立了 5 级金字塔结构。测试环境为:

硬件环境:NVIDIA GeForce4/MX440 64M DDR VRAM 显卡、512M 内存、PIV 1.7G CPU

软件环境:Windows 2000 sp4 操作系统、DirectX9.0c

测试结果显示数据块之间接边良好,画面流畅,平均绘制速度为 40 帧/s(3D 视频游戏一般来说 30 帧/s 以上就能较好满足交互式需求),每帧绘制三角形个数为 12 000,demo 运行时系统消耗内存的数据为 70M。

通过试验证明,算法对地形有明显简化效果。通过与视点相关方式调度数据,根据视点位置动态生成地形模型,在不影响视觉效果的同时大大加快了绘制速度,算法基本上能够做到绘制速度与数据量无关。

参考文献:

- [1] Lindstrom P, Koller D, Ribarsky W, et al. Real-time, continuous level of detail rendering of height fields [C]//SIG GRAPH'96 Proc Computer Graphics Proceedings, Ann Conf Series. New Orleans, LA: ACM SIG GRAPH, 1996: 109-118.
- [2] Duchaineau M, Wolinsky M. ROAMing Terrain: Real-time Optimally Adapting Meshes [C]//IEEE Visualization '97 Proceedings. [s.l.]:[s.n.]. 1997: 81-88.
- [3] Hoppe H. Smooth view-dependent level-of-detail control and its application to terrain rendering[C]//Proc of Visualization'98. Los Alamitos, Calif: IEEE Computer Society Press, 1998: 35-42.
- [4] 赵友兵,潘志庚,石教英. 视点相关的地形 LOD 模型的动态生成算法[J]. 软件学报,1999,10(增刊):251-254.
- [5] Ögren A. Continuous Level of Detail In Real-Time Terrain Rendering [D/OL]. 2000. <http://www.cs.umu.se/~tdv94aog/ROAM.pdf>.