文章编号:2095-476X(2014)03-0087-05

# 基于四叉树的大规模地形的生成算法

张娜, 殷知磊

(郑州轻工业学院 计算机与通信工程学院, 河南 郑州 450001)

**摘要**:提出了一种基于四叉树的多分辨率地形模型生成算法.该算法将视点相关、地势起伏等因素纳入细节层次的评价标准中,并在地形生成的过程中根据节点细分和渲染的规律,提出了合适的裂缝消除方法.地形绘制采用了视景裁剪、背面裁剪、三角扇形和静态数据预存等优化显示技术.实验结果表明,该算法能够快速实现大规模地形的实时可视化.

**关键词**:三维地形;四叉树;视点相关;多分辨率模型

**中图分类号**:TP391 **文献标志码**:A DOI:10.3969/j.issn.2095-476X.2014.03.020

## An algorithm for large-scale terrain generation based on quadtree

ZHANG Na, YIN Zhi-lei

(*College of Computer and Communication Engineering, Zhengzhou University of Light Industry, Zhengzhou* 450001, *China*)

**Abstract**: An algorithm for multi-resolution terrain model generation based on quadtree was put forward. This algorithm took elements as view-dependent and terrain relief into the evaluation standards for level of detail and raised appropriate methods to remove cracks in the course of terrain generation according to the law of node partition and rendering. In drawing terrain, optimized display technologies were used such as frustum culling, backface culling, triangle fan and static data storage. Test results showed that this algorithm could rapidly realize the real-time visualization of large-scale terrain.

**Key words**: three-dimensional terrain; quadtree; view-dependent; multi-resolution model

## 0 引言

三维地形场景实时绘制在虚拟现实、地理信息系统、仿真系统等领域应用广泛.对于地形可视化的研究,现已能够构建实时交互的三维场景.但是,客观存在的地理环境很复杂,使得数据信息具有无限性,且计算机软硬件技术的发展是有限的,因而要实现交互的地形绘制必须进行视点相关的模拟.目前,小规模的地形绘制已没有什么困难.对于大规模的、复杂的地形,渲染时需处理的数据量庞大,超出普通计算机CPU的计算能力,地形数据在地形绘制时也不能一次性载入内存,需依据可视范围(由视点的位置和视线方向确定)及一定的调度策略,动态地在计算机的内存、外存间调入调出数据.因而就必须对三维地形模型进行合理的设计,并优化地形数据的组织形式和调度策略.近年来,国内外学者对此进行了广泛研究,并提出了许多地形可视化算法[1-6].文献[1]基于外存有效组织大规模地形数据,并在运行时通过调度策略减少载入内存数据量以完成地形绘制,算法效率较低.文献[2]给出视相关连续细节层次(LOD)方法,解决了绘制同一区域块时在不同LOD间切换而产生的跳变问题,但不能很好地解决地形块间的裂缝问题.文献[3-5]通过使用高效的外存组织及预存取策略减少数

**收稿日期**:2013-12-16

**基金项目**:国家自然科学基金项目(61201447)

**作者简介**:张娜(1977—),女,河南省郑州市人,郑州轻工业学院讲师,硕士,主要研究方向为计算机图形图像处理.

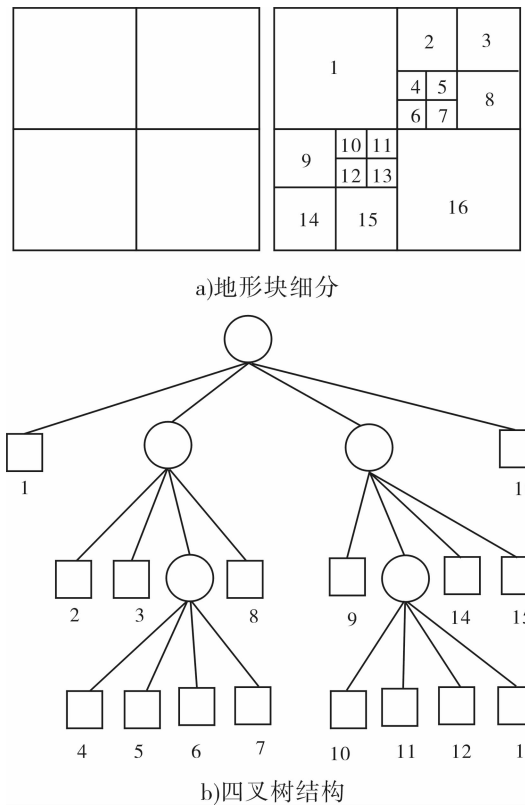
据的输入与输出,实现了实时绘制,但地形块的不同 LOD 间平滑性过渡不够好. 文献[6]采用 GC(Geometry Clipmap)方法对大规模地形进行高效绘制,该方法将高程纹理地形缓存在一组嵌套规则网格中(即压缩),运行时刻需将所需数据在内存中实时解压以完成地形实时绘制,但算法实现比较复杂. 在对前人算法研究和比较的基础上,本文拟提出一种基于四叉树结构的、视点相关的多分辨率大规模地形生成及绘制算法,并采用多项技术对地形绘制进行优化,以期实现大规模地形模型的快速实时生成.

## 1 多分辨率地形模型

多分辨率几何模型是指不同区域(部分)具有不同细节层次的模型[7],其基本思想是用不同 LOD 来构造或近似模拟场景. 当与视点的距离不同时,场景模型(整体或局部分辨率依赖视点变化)的 LOD 可以不同,并随视点的移动做相应的变化,同时根据地表特征的变化,场景模型在突变处和缓变处也可采用不同 LOD. 在不影响场景视觉效果的前提下,采用 LOD 技术可以提高大规模地形的绘制速度. 当前生成地形的算法多数建立在数字高程模型(DEM)数据的基础上,且因规则网格表示的 DEM 与树结构具有天然的一致性,故可用四叉树这种数据结构来表示地形模型.

### 1.1 四叉树地形模型

四叉树结构是按照空间区域将 4 个象限递归分割的一种数据结构,常被用来进行空间数据的组织. 利用四叉树数据结构建立多分辨率地形模型的主要思想为:对 DEM 地形数据行列进行不同精度的等间距网格采样(采样数目为$(2^n+1)\times(2^n+1)$),并由地形块的大小和所在 LOD 层次数来决定采样数据间距. 采样的结果形成了四叉树中一层节点,且每个节点对应整个地形的一块区域. 本文按照自上而下的方式构造四叉树地形模型,即对整个地形模型进行层次划分,并在划分过程中计算节点的误差. 具体构造过程:整个地形作为四叉树的根节点,先把四叉树根节点 4 等分划分为子块,之后检查划分得到的各个子块分辨率(即计算对应四叉树上节点的误差值)是否达到了要求,满足要求的子块不再向下划分,否则把此子块再 4 等分为更小的子块,一直如此递归下去,直到所有划分得到的子块都满足分辨率要求为止. 多分辨率四叉树数据结构如图 1 所示,图 1a)中每个地形块对应四叉树上的 1 个节点,四叉树的每个非叶子节点有 4 个等分父节点对应区域的子节点. 若得到的子节点误差大于限定的阈值,依次对 4 个子节点所在区域继续进行划分,直到每个子节点的误差小于给定的阈值. 在这里,相邻数据块(地形块)间共享边界行和列,且通过对高分辨率的数据块隔行隔列采样获得低分辨率的数据块.




a)地形块细分

b)四叉树结构

图 1 多分辨率四叉树数据结构

### 1.2 地形模型误差计算

基于四叉树数据结构进行多分辨率地形绘制时,为提高绘制效率,应尽可能地避免不必要的节点细分,以减少数据冗余. 对于是否细分节点,通常可通过 2 个因素来确定:节点距视点的距离;地形自身的粗糙程度.

一般情况下,越靠近人的视点,地形的细节程度越高;反之则越低. 距视点相同距离时,对应四叉树节点的地形块越大,该地形块具有的细节程度越高;反之则越低. 从中可得到如下公式:

$$\frac{l}{d} < C_1 \qquad ①$$

其中,$l$ 为节点的中心点到视点的距离;$d$ 为节点的大小(即节点所对应的地形块的宽度);$C_1$ 是细分程

度调整因子. $C_1$ 值减小,表示要描述的地形细节减少;反之,则细节增多. 当需处理的节点的数据信息符合 ① 式时,则该节点进一步细分.

绘制地形时,地形的粗糙度也对最终渲染效果有一定影响. 越复杂的地形,需要的描述细节越多;反之,越简单的地形,需要的描述细节越少. 图 2 表示一个四叉树节点及其子节点的粗糙程度信息. $\Delta H_1$—$\Delta H_6$ 分别表示节点中心点(节点中心点与2个对角线中点插值产生 2 个误差) 和 4 个边中点是否被剖分时产生的6个高度误差值,$\Delta H_7$—$\Delta H_{10}$ 表示 4 个子节点的粗糙度. 粗糙程度的评价公式为

$$\frac{1}{r} < C_2 \qquad ②$$

其中,$r$ 为 10 个误差值($\Delta H_1$—$\Delta H_{10}$) 中最大的 1 个;$C_2$ 为一个可以调节的参数,$C_2$ 值越大,地形细节越多.

依据公式 ① 和 ②,对距视点距离和地形粗糙程度进行综合考虑,得到评价公式

$$f = \frac{l}{d \times r \times C_1 \times C_2}$$

上式表示的节点细分误差评价标准充分考虑了视点相关性和地形粗糙度 2 个因素. 这一评价标准计算简单、形式直观,且系统的性能可通过改变参数来进行灵活的调整.

### 1.3　地形模型裂缝消除

在四叉树地形算法中当相邻地形块间具有不同 LOD 时,LOD 高的地形块比 LOD 低的地形块具有更多的高程信息,所以当两者共享同一条边时常会产生裂缝问题. 关于消除地形裂缝问题的方法,常用的有减边法和加边法,与加边法相比,减边法能更快捷、高效地消除裂缝,且容易实现. 为了提高绘制效率,本文采用减边法消除相邻地形节点间的裂缝(见图3). 图3a)中,节点 Q 和节点 M 的交接点 C 和 B 处会产生裂缝. 通过将节点 Q 的细分标志改为 FALSE(即减少细分的边),可消除节点相邻边上交接点 B 处的裂缝. 因一个节点的细分标志为不分割时,其子节点的细分标志也应为不分割,所以要进一步消除交接点 C 处的裂缝,需把节点 Q 的子节点 E 和 F 的细分标志也改为 FALSE. 同样地,子节点 H 的细分标志也需要改为 FALSE. 这样,如图3b)所示,相邻节点间的裂缝就完全消除了.

为了减少四叉树节点的计算量,本文算法采用地形分块技术. 若仅根据上述节点评价标准对节点进行强制细分,对相邻节点的状态不去考虑,相邻

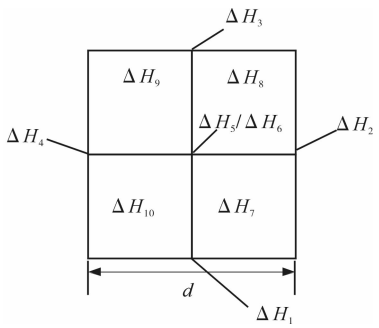



图 2　四叉树节点的粗糙程度信息

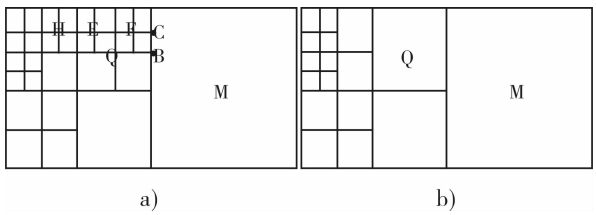



图 3　节点减边消除裂缝示意图

地形节点间仍会产生裂缝现象. 因而,除了符合节点评价标准外,细分节点时还需确保邻接节点间的分辨率之差小于等于 1,这需要另外遵循条件:若满足节点误差评价标准要求,且相邻的 4 个节点(即上下左右节点)都已参与细分;则当前节点可参与细分;否则直接绘制当前节点.

## 2　算法流程及优化

### 2.1　算法基本流程

多分辨率四叉树网格地形模型生成及地形绘制算法流程如图 4 所示. 首先对分块地形依据视景体裁剪确定视域内的地形块,之后根据视点和限定的误差阈值对四叉树节点进行误差判断来细分节点,再采用相应方法消除地形块间的裂缝,最后通过遍历节点生成误差范围内的四叉树结构地形模型. 地形绘制采用背面裁剪和三角扇形技术,并选择与四叉树叶节点相应的纹理图像进行纹理映射.

### 2.2　视景体裁剪

基于四叉树算法绘制地形时,为提高绘制速度,只绘制位于可视范围内的地形节点,即裁剪屏幕显示区域外的节点. 裁剪掉不在视线范围内的节点就是对参与绘制的地形块节点的可见性进行选择. 剔除节点时,首先将场景中的不可见部分利用裁减算法去除,再利用四叉树算法对剩余可见部分动态建模. 这样,视景体内部的地形节点在场景显示时能够被看到,不在视景体内的节点会被裁减掉.

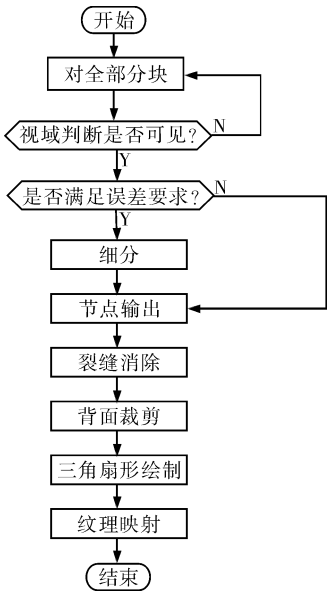



图 4 算法流程图

四叉树上每个节点对应一块具体的有尺寸地形区域,因而需要根据地形节点和视景体的空间位置关系情况进行节点的可见性选择. 地形节点与视景体的空间位置关系有包含、相交、分离 3 种. 执行四叉树算法时,首先根据根节点是否在可视范围中(即完全包含于或部分包含于视景体中),判断是否对其细分. 如果根节点位于可见范围中,就将根节点划分成 4 个子节点,之后,重复这个过程,迭代地向下细分节点,直到根据细分标准不能继续细分时,这一递归过程终止. 对于那些不在可视区域的节点,算法将其直接忽略掉,并按三角扇形方式输出那些判断为可见但不能再被细分的节点.

### 2.3 背面裁剪

进行三维地形漫游时,通常视点方向几乎平行于地面,显示场景中近一半的多边形因背向视点方向而不可见,勿需绘制,这就需要采用背面裁剪技术. 背面裁剪是在绘制地形前,将与视点方向背离的多边形剔除掉,只绘制面向视点的多边形,以此来减少绘制的多边形数目. 删减多边形时,计算视点方向与多边形表面法向量的夹角,当夹角大于 90°时,剔除该多边形;否则绘制该多边形.

### 2.4 静态信息的预先计算

为了进一步提高地形绘制的速度,在算法实施前,预先计算出一些能够确定的静态数据并保存起来. 本文依据由视相关参数(节点距视点距离及节点误差调节因子等)预先计算出的各地形节点是否需再细分的状况,对四叉树上的数据节点进行预处理. 若某个节点需继续细分,则四叉树上该节点标识成可用状态;若不需进一步细分,则标识成不可用状态;节点若没有访问到,则标识为不确定状态. 此外,可以预先计算好场景中光照的法向量.

### 2.5 三角扇形绘制

三角扇形是指多个三角形互相连接,共享同一个节点. 绘制地形时,因四叉树所具有的一些固有特性使得在算法中非常适合使用三角扇形优化技术.

OPENGL 在绘制具有三角扇形特征的三角形时,可采用快速的优化方法进行绘制,而勿需将各个三角形都绘制一遍. 在采用四叉树结构算法对地形细分后,四叉树上的每个节点都具有三角扇形特征. 一个节点绘制时,因与其邻接之节点的状态都是已确定的,故可加快算法的绘制速度. 绘制时的具体操作是:将节点的中心点作为中点,以节点的左上角点作为起始点,沿顺时针方向依次游历每个角点及边点. 游历到的每个角点都需要绘制,而对于边点是否要绘制则要根据邻接节点状态来判断:如果邻接节点为可用状态,则该边点应绘制;否则就不绘制.

## 3 实验结果

本文在 Windows XP 下使用 VC + + 6.0 和 OPENGL 进行仿真实验. 实验所使用的硬件设备的主要配置为:P4 2.4G CPU,2G 内存,GetForce4 显卡. 文中选择了多个 DEM 数据对不同算法进行了测试对比. 表 1 给出了当地形数据未采用 LOD 技术、采用 LOD 技术和 LOD 技术 + 优化技术(背面裁剪) 3 种不同算法时的网格简化率和相应的绘制速度.

由表 1 可知,若 DEM 网格数据量大于1 025 × 1 025后,在普通 PC 机器上绘制大规模地形时,地形画面的绘制速度明显下降. 对比 LOD 技术和 LOD +

表 1 不同网格点 DEM 数据采用不同算法的简化率及绘制速度比较

| DEM 数据 | 原始地形绘制速度/(f·s$^{-1}$) | LOD 技术 | | LOD + 优化技术(背面裁剪) | |
|---|---|---|---|---|---|
| | | 网格简化率/% | 绘制速度/(f·s$^{-1}$) | 网格简化率/% | 绘制速度/(f·s$^{-1}$) |
| 257 × 257 | 90 | 63.1 | 105 | 69.7 | 99 |
| 513 × 513 | 60 | 66.2 | 85 | 73.3 | 80 |
| 1 025 × 1 025 | 34 | 70.4 | 57 | 76.4 | 54 |
| 2 049 × 2 049 | 25 | 79.7 | 45 | 84.6 | 42 |
| 4 097 × 4 097 | 18 | 84.2 | 37 | 87.7 | 35 |

优化技术(背面裁剪)2 种方法可以看出,使用背面剔除算法后增加了地形网格的简化率,但因地形网格绘制前需判断视线与网格法向量间的关系,故影响了地形绘制的效率,但就综合情况来看,它的效率与传统的四叉树 LOD 技术相差不多.

本文使用 LOD + 优化技术(背面裁剪)算法不仅使得三维地形网格简化很多,并且能够得到画面流畅的绘制地形. 图 5 为本文算法基于 4 097 × 4 097的 DEM 数据所生成的多分辨率三维地形网格模型. 在上述软硬件条件下,依据所得到的三维地形模型能够实现较大范围地形的实时动态显示,平均绘制帧速率在 35 f/s,满足了系统的实时性要求.

## 4　结论

本文在已有的地形模型生成算法基础上,提出了一种基于四叉树结构的多分辨率大规模地形生成算法. 该算法采用高效的地形生成方法,并通过使用一些系统关键优化技术,提高了地形的实时绘制速度. 实验结果表明,该算法实现简单、速度快,能够满足用户对大规模地形的实时可视化的要求.

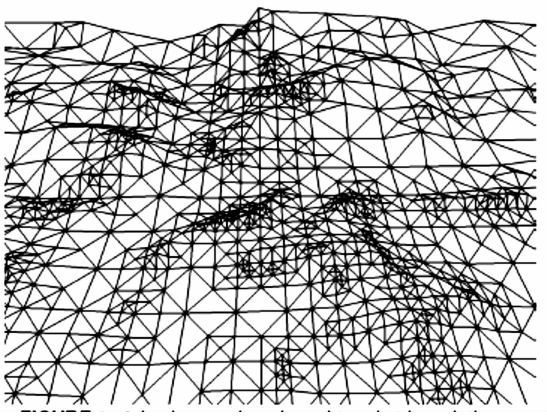


图 5　多分辨率地形网格模型

**参考文献:**


[1] Lindstrom P, Pascucci V. Terrain simplification simplified: a general framework for view-dependent out-of-core visualization[J]. IEEE Transaction on Visualization and Computer Graphics, 2002, 8(3): 239.

[2] Hoppe H. Smooth view-dependent level-of-detail control and its application to terrain rendering[C] // Proceedings of Visualization'98, NC: IEEE, 1998: 35.

[3] 李胜,冀俊峰,刘学慧,等. 超大规模地形场景的高性能漫游[J]. 软件学报, 2006, 17(3): 535.

[4] Cignoni P, Ganovelli F, Gobbetti E, et al. Planet-sized batched dynamic adaptive meshes (P-BDAM)[C] //Proceedings of the 14th IEEE Visualization, Seattle: IEEE, 2003: 147.

[5] 谭力恒,魏迎梅. 基于区块自适应网格的全球地形快速绘制算法[J]. 系统仿真学报, 2012, 24(1): 146.

[6] Losasso F, Hoppe H. Geometry clipmaps: terrain rendering using nested regular grids[J]. ACM Transaction on Graphics, 2004, 23(3): 769.

[7] 李晨辉,王长波. 基于层次划分的大规模 LOD 地形绘制[J]. 东华大学学报:自然科学版, 2010, 36(4): 351.