基于 GPU 的真实感地形绘制

Realistic Terrain Rendering based on GPU

史胜伟 姜昱明 朱新蕾 （西安电子科技大学计算机学院 陕西警官职业学院公安信息技术系 陕西 710071）

摘　要： 如何高效地处理与显示大规模地形数据目前仍是一个挑战，在分析了大规模地形建模与可视化中存在的主要问题后，充分利用 GPU 的强大功能，在 GPU 上进行可见性剔除，实现了 CPU 与 GPU 的并行处理。同时充分利用 GPU 的渲染功能，提高可视化的真实感和实时漫游的速率，实现了大规模三维地形的实时漫游。

关键词： 三维地形可视化 ROAM 真实感 GPU 并行处理

大规模三维地形的实时漫游面临两大困难问题[1]：原始地形的数据量远远超过了系统的实时处理能力；与地表模型相对应的高精度纹理更加重了交互漫游的难度。针对这两个问题最常用的解决方法有：一种方法是采用视点相关自适应层次细节技术（LOD）简化整个场景的复杂度。另一种方法是采用可见性剔除技术，包括视域剔除、背向面剔除和遮挡剔除。

近年来，随着图形处理器（GPU）性能的大幅度提高以及可编程特性的发展，人们开始将图形流水线的某些处理阶段以及某些图形算法从 CPU 向 GPU 转移。除了计算机图形学本身的应用，还涉及到其他领域的计算，以至于近年来通用计算成为 GPU 的应用之一，并成为研究热点[2]。本文主要研究了三维地形可视化和 GPU 批处理方法，一方面对大规模地形模型进行简化以减少数据量；另一方面充分利用 GPU 的强大功能解决 CPU 的瓶颈问题，提高三维地形可视化的真实感，实现对大规模三维地形的实时漫游。

1　基于 ROAM 算法的建模

Duchaineauy[3] 描述了一个基于三角形二叉树结构的 ROAM（实时优化自适应网格）算法。其基本思想是：对地形进行三维显示时，根据视点的位置和视线的方向计算视点距离地形表面的三角区块的距离，再根据目标格网的空间粗糙程度来判断是否对地形表面的三角区块进行一系列基于三角形二叉分割的分解和合并，最终生成逼近真实地形的简化连续三角化地形表面。

假设初始 DEM 数据是正方形网格，首先沿一条对角线将其分割为两个等腰直角三角形，这样就得到了第一层地形块，并把它们定义为顶层（L = 0）。每一个地形块，其都是一个简单的等腰直角三角形，从它的顶点到对面斜边的中点对其进行分割，生成两个新的等腰直角三角形。如此方法，递归地重复对其子三角形进行分割，直到达到希望的细节层次。如图 1 所示：

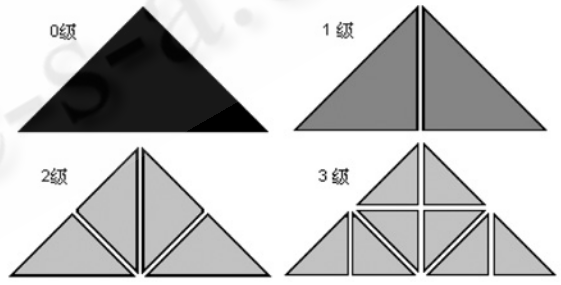


图 1　三角形二叉树分割

该方法构造的三角形二叉树，可以容易地实现地表网格模型的细化和粗化。当用一对子三角形来表示其父三角形区域时，就完成了细化，称之为父三角形的分裂；反之，用父三角形替代其子三角形对时即完成了粗化，称之为子三角形的合并（如图 2）。由于这里的细化和粗化都是在局部完成的，可以连续的进行，分割合并速度非常快，可以实时动态更新地形网格模型，满

足实时漫游的需要。

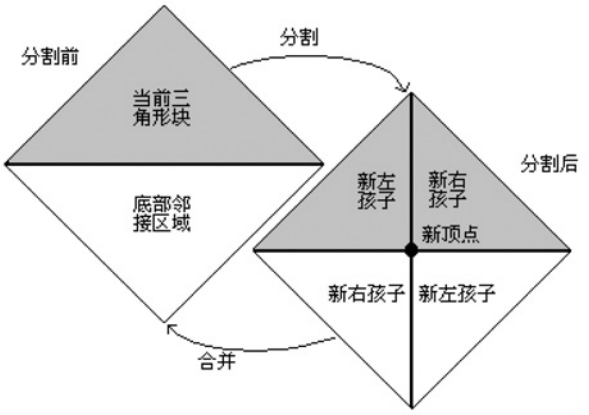


图 2 地形块的分割与合并

1.1 数据结构定义

ROAM 算法产生的地形块由左右两个等腰直角三角形构成,也就是从块对角线进行分割时产生的两个三角形,每个三角形形成一个独立的三角形二叉树,每个小的地形块由两个二叉树组成。若需要更细的细节时,可用同样的方法再进一步分割。由 ROAM 算法可得到每一地形块的邻接关系图,如图 3 所示:

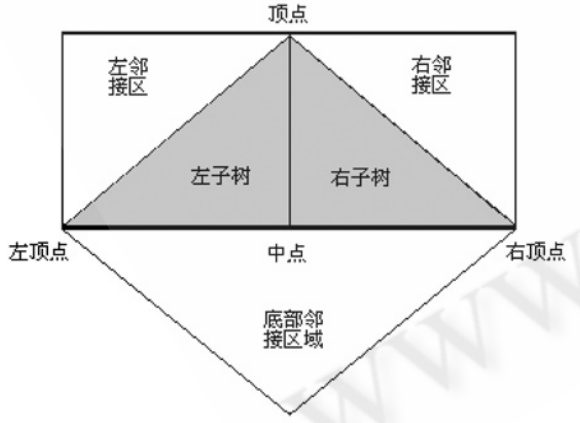


图 3 二叉树邻接关系

由上面的邻接关系图定义每一个节点的数据结构如下:

```
struct TriTreeNode
 {
TriTreeNode * left_child ;//左孩子
TriTreeNode * right_child ;//右孩子
TriTreeNode * base_neighbor ;//底部邻接区块
TriTreeNode * left_neighbor ;//左邻区块
TriTreeNode * right_neighbor ;//右邻区块
} ;
```

1.2 裂缝处理

对地形块上相邻的三角形二叉树分割时,分割后可能会使格网产生裂缝。这个裂缝是由于对相邻的地形块分割过程中,边界上形成了一边是更细的层次,而另一边是它的上一级层次。由于他们的高程值是近似的计算,这样便导致分割后得到的新点的高程值与原来边界上的高程值的不一致,形成了裂缝。其根本原因是由于相邻地形块的细节层次不同造成的。

为了解决这个问题,考察 ROAM 格网的邻节点,它有这样一个有趣规律:一个节点和它的邻节点只存在两种关系:共直角边关系(如左右邻接区)和共斜边关系(如底部邻接区)。进行格网分割时,可以利用这个规律以使相邻的节点在分割时同时进行。具体做法:对于一个节点,只有当它与它的底部邻接节点呈相互下邻关系时才进行分割(参看图 3),这样可以保证在分割时与它相邻结点也同时进行分割,从而得到一致的细节层次,因此在网格上不会出现裂缝。

分割一个网格时存在三种可能:①节点是正方形块的一部分,直接分割它和它的底部邻接区域(参看图 2);②节点是网格的最外的一条边,只分割这个节点;③节点不是正方形的一部分,强制分割。强制分割指的是递归的遍历整个网格直到发现正方形样的节点或网格边。分割流程为:当分割一个节点时,首先看是不是正方形的一部分,如果不是,然后在它的底部邻接节点上调用分割操作,第二个分割操作仍做同样的工作,只要发现节点可以递归的分割,就一直分割下去,直到有满足前两种情况的网格出现并执行分割,再按原路径回朔分割,最终完成分割操作。如图 4 所示,要分割标号为 1 的节点,其斜边不是网格最外的边也不能与底部邻接节点 2 号节点组成正方形,将 1 号节点入栈,同时其底部邻接节点 2 号节点作为当前要分割的节点。如此一直遍历到 4 号节点,成功分割后返回,分割 3 号节点,最后分割 1 号节点。

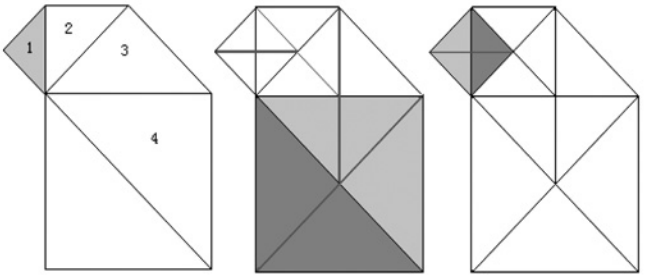


图 4　强制分割操作

分割地形的同时递归地构建三角形二叉树,其中树的叶节点中保存着需要进入图形渲染流水线的三角形序列;每两个三角形二叉树组成一个正方形网格;无数个这样的网格共同构成地形模型。在主程序中保存着所有已调入内存的三角形块的索引以便于实时渲染,同时保存它们之间的层次关系,如图 5 所示。

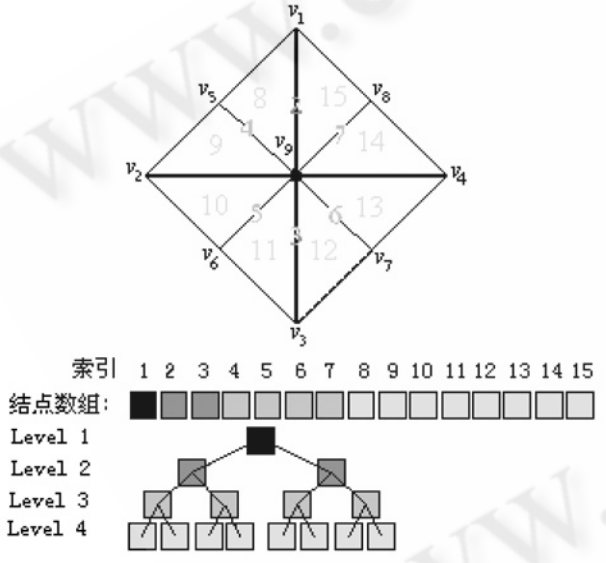


图 5　地形块组织图

2　基于 GPU 的实时绘制

对于大规模三维地形实时漫游,CPU 的计算量相当大,因此一直是实时漫游的一个瓶颈。GPU 首先由 nVidia 公司于 1999 年提出,GPU 的特征是:SIMD 并行处理,多管道流处理器;高密集的运算;不需要在 CPU 与 GPU 之间进行数据交换;在顶点级和像素级提供了灵活的可编程特性[4]。

2.1 基于 GPU 的可见性剔除

可见性剔除技术,即从场景中剔除对于当前视点不可见的区域,包括视锥体外、背面和被遮挡的区域。在视点不断变化过程中,有时会有大部分区域位于视锥体外,用视域剔除方法可以裁剪掉大量地形网格。另外,地形的起伏特征也会导致不同的地形区域的相互遮挡,这样我们可以通过遮挡剔除进一步裁剪掉视锥体内被遮挡的区域。目前,针对地形场景存在很多种遮挡剔除算法,但基本上都是以 CPU 为核心的,通过牺牲额外的 CPU 时间进行遮挡剔除,从而减少送入图形流水线的多边形数目[5]。

目前的图形加速卡已能提供遮挡查询功能,其具体过程是:首先启用遮挡查询,然后绘制物体或其包围区域,但并不立即写入缓冲区,再通过对深度测试和模板测试的像素数量进行计数并返回一个值,如果返回值为零,则物体被遮挡,可以剔除,否则将物体送入图形流水线进行绘制,同时更新深度缓冲区和模板缓冲区。

深度测试的通用方法是 Z 缓存(Z – buffer)技术,在 Z – buffer 中记录着每个距离观察点最近的像素点,由此可决定表面的可见性。首先使用 glDepthFunc() 确定测试的条件,再用 glEnable(GL_DEPTH_TEST) 启动深度测试。如果即将画的像素比已经存在的像素离观察点近,则将其颜色分别存于颜色缓存和 Z – buffer,替代原来的值。

2.2 基于 GPU 的实时渲染

三维真实感图形的常用手段是光照和纹理映射等。

OpenGL 定义了几种简单光照模型:辐射光、环境光、漫射光、镜面光。可以通过函数 void glLight {if} {v} (GLenum light, GLenum pname, TYPE param) 来创建具有某种特性的光源。OpenGL 最多支持 8 个不同的光源,可以组合出较为复杂的光照效果。

OpenGL 纹理映射是一个相当复杂的过程,基本步骤如下:定义纹理、控制滤波、说明映射方式、绘制场景,给出顶点的纹理坐标和几何坐标。二维纹理定义的函数:void glTexImage2D(GLenum target,Glint level, GLint components, GLsizei width, glsizei height, GLint border, GLenum format,GLenum type, const GLvoid * pixels);

此外，OpenGL 显示列表的设计能优化程序运行性能，提高三维地形的生成与渲染，实现 GPU 与 CPU 的并行处理。它被设计成命令高速缓存，而不是动态数据库缓存。显示列表适用于以下场合 矩阵操作，光栅位图和图像，光、材质和光照模型，纹理，多边形的图案填充模式等。

3 实验结果

运用本文上面提到的方法，首先建立多分辨率模型并动态地装入地形模型，随后在程序运行过程中交互地改变视点并根据视点参数实时地细化网格，最后充分利用 GPU 的强大功能绘制网格、纹理贴图、实时渲染，很好地实现了真实感三维地形的实时交互性漫游。

实验测试系统使用 nVidia Geforce FX 5200 GPU 芯片，内存为 521M，CPU 为 Intel Celeron 2. 0GHz，使用 Microsoft Visual Studio 2005 在 WinXP 下开发完成。

实验程序运行结果实现了流畅地漫游三维地形，纹理贴图后的真实感三维地形漫游帧速率在 16 – 39 帧/秒之间，网格地形的漫游帧速率在 60 帧/秒以上。图 6 是网格简化的对比图，图 7 是真实感三维地形漫游时的部分截图。

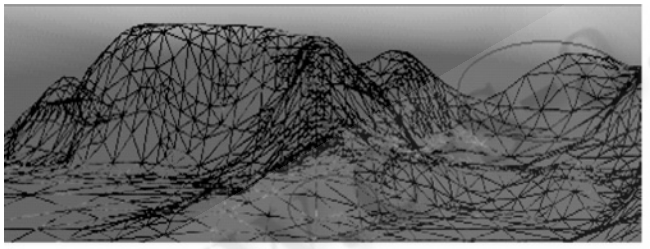
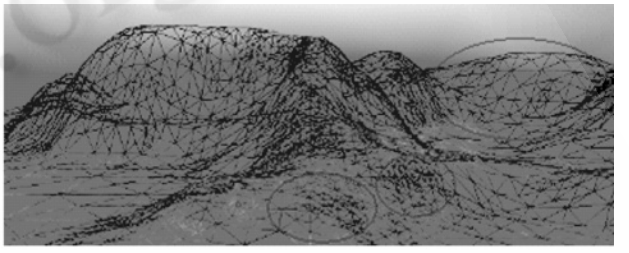


图 6a 远视点的地形网格模型



b 近视点的地形网格模型

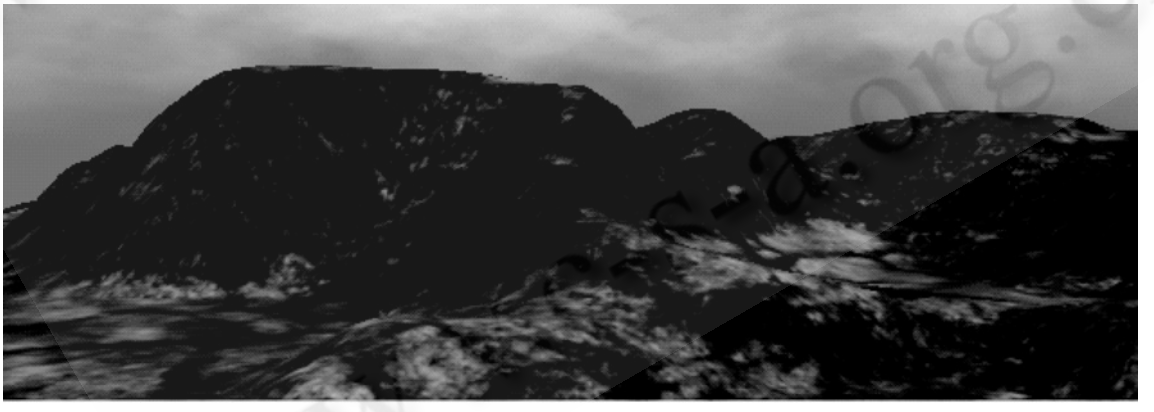


图 7 真实感三维地形模拟

参考文献

1 李胜，冀俊峰，刘学慧，等. 超大规模地形场景的高性能漫游. 软件学报，2006，17(3): 535 – 545.

2 吴恩华，柳有权. 基于图形处理器(GPU)的通用计算. 计算机辅助设计与图形学学报，2004，(5): 601 – 612.

3 Mark Duchaineauy. Roaming terrain Real – time optimally adapting meshes. Proceedings of IEEE Visualization´97. Phoenix. 1997. P81 – 88.

4 张建勋，刘全利，陈 庄. 基于可编程的快速体绘制技术. 重庆大学学报(自然科学版). 2005，(7): 67 – 70.

5 达来，曾亮，李思昆. 基于 GPU 的地形遮挡剔除算法. 系统仿真学报. 2006，(11): 3165 – 3171.