四叉树地形金字塔模型面向对象的设计与实现①

Object-Oriented Design and Implementation of Quadtree Pyramid Terrain Model

王卫红 周 军 顾国民 (浙江工业大学 软件学院 浙江 杭州 310023)

摘 要: 针对海量真实地形数据的可视化与当前计算机硬件性能之间存在的矛盾，本文提出了一种分层分块金字塔模型和四叉树相结合的数据组织和管理方法。在.Net Framework 下采用 C#语言和 Managed DirectX 实现。实验表明该方法能有效管理地形数据，实现动态的实时装载和实时渲染地形数据。

关键词: 四叉树 金字塔模型 可视化

1 引言

地形数据实时可视化是计算机图形学、三维地理信息系统、虚拟现实与仿真等领域的重要研究内容之一[1]。由于地形数据通常都是以 GB 或者 TB 为单位，当前个人计算机无法将所有的数据一次性加载建立整个三维地形场景。近年来，国内外许多学者为解决这个矛盾进行了深入的研究。目前最受关注的是多分辨率层次细节模型，基本思想是：用不同的层次细节表示一个三维场景，层次细节的级别取决于区域与视点的关系，并随视点的移动实时更新各个区域的层次细节级别。

金字塔是一种多分辨率层次模型，在某种意思上可以说是一种连续多分辨率层次模型。它通常采用倍率的方法构建金字塔，形成多个分辨率层次。从金字塔的底层到顶层，分辨率越来越低，但是表示的范围不变[2]。分辨率的公式可以定义为：假设地形数据原始的分辨率为 R0 倍率为 m，则第 k 层地形数据的分辨率为 Rk=R0*M-k。因此，通过构建金字塔模型，可以为地形可视化系统提供不同分辨率的地形数据。将多分辨率地形数据与金字塔模型结合在一起，实时动态的加载数据，从而降低内存的消耗，使得当前个人机能实现大规模地形数据的可视化。

2 Managed DirectX和C#介绍

2.1 C#语言

C#是微软公司发布的一种面向对象的，运行于.NET Framework 之上的高级程序设计语言，它集合了 Java、C 和 C++语言的一些优点，但是 C#语法比 C++简单，在 C++中能完成的任务利用 C#也能完成。因此，对程序员来说 C#是一种高效的开发语言。

2.2 Managed DirectX

Managed DirectX(MDX)实质上是对 DirectX 进行了一个轻量级的封装，因此保存了 COM 接口 DirectX 的大部分原貌，可以用任何支持.NET 的语言进行开发。相对 COM 接口的 DirectX 具有以下优点：(1)对象模型更好，MDX 基于类库的组织结构，与用 COM 接口的处理方式相比更方便。(2)MDX 对大部分资源释放的操作都进行了封装，使用时不要考虑资源释放的问题。(3)应用更加广泛，可以让 DirectX 程序使用 XML、WebService 和智能客户端等技术。

3 分层分块方案与数据组织

3.1 分层分块方案

计算机硬件技术如 CPU 主频、内存容量、显卡性能等等在飞速发展的同时，计算机处理的数据量也在迅速增长，有的甚至超过硬件的发展速度，除此之外，处理难度和复杂度也在不断增加，地形数据就是一个例子。论文[3]针对传统分层分块方案索引速度慢、可

① 基金项目:国家自然科学基金项目(60873033,60673063);浙江省自然科学基金重点项目(Z105391)
收稿时间:2008-09-27

扩展性差等缺点提出了新的分层分块方案。本文在其基础上，结合四叉树特性，提出了适合面向对象语言实现的金字塔模型分层分块规则，规则如下：

(1)规定全球地理坐标经度范围为[-180° ,+180°]，全球地理纬度范围为[-90° ,+90°]，此范围以外的坐标值均视为无效值；

(2)金字塔模型中的每一层对应一个层次细节级别，模型的分辨率越低，层次细节基本也越低；

(3)规定第 K+1 层的分辨率为第 K 层的 2 倍，这个 2 倍同时约束地形模型和纹理模型；

(4)规定第 0 层(最低分辨率层)以 36° 为单位，将整个地理坐标划分成 5＊10 正方形的小块。

(5)规定第 0 层之后的其他层，在第 0 层基础上采用图 1 的分块方式进行划分。

(6)规定每层块的编号从左到右，从上到下。

(7)规定被加载的数据的分辨率的级别由视点与区域的距离和最初设定的分块最小值(阈值)共同决定。当分块的值比阈值大时，加载数据的分辨率由视点与区域的距离决定。

前面 6 点是存储在外部设备上数据的划分，第 7 点是针对程序实现，确定动态加载数据的级别。

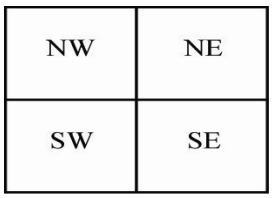

图 1　分块方式

3.2 地形数据组织和命名

一个好的地形数据在外部设备(如硬盘)上组织和命名方式，将大大提高系统动态获取数据的速度和效率。根据上面提出来的分层分块规则，我们采用的文件命名方法和地形数据组织结构(包括高程数据和纹理影像)规则如下：

(1)金字塔一层对应一个文件夹，文件夹名称以层所在的层次命名，如第 0 层，则文件夹命名为 0。

(2)在同一层内以行为单位，即一行对应一个文件夹。

(3)一个分块对应一个数据文件，文件的名称以他所对应的块在该层索引值决定，如一个文件对应某层第 13 行第 25 列的块，则该数据文件命名为“13_25.文件后缀”。

依照上述规则，假如一个在第 5 层第 8 行第 20 列的地形数据块对应文件的相对访问路径为“..\5\8\8_20.文件后缀”。系统在运行时，只要根据经纬度按照图 2 行列的计算公式，即可快速、准确定位所需的数据文件。公式中的 latitude 和 longitude 是某点的经纬度值，tilesize 则是在该分辨率级别下分块的大小。

行计算公式：

$$row = \left(abs\left(-90 - latitude\right)\%180\right) / tilesize$$

列计算公式：

$$col = \left(abs\left(-180 - longitude\right)\%360\right) / tilesize$$

图 2　行列计算公式

依照该方法组织和命名数据文件有以下几个优点：(1)符合分层分块的地形金字塔模型；(2)防止数据出现重命名的情况；(3)使得系统能通过简单的计算就能快速准确地得到所需数据的存放位置，提高系统的运行效率。

4　设计与实现

根据上述规则按照功能设计类结构，精简了代码，提高了程序的可读性和运行效率。类的成员(并未列出所有成员)和类之间的关系如图 3 所示。QuadeTileSet 是一个负责存放 QuadeTile 实例的容器类；QuadeTile 是这个结构的核心，它负责四叉树金字塔模型的生成和渲染；TextureLoad 和 TerrainLoad 分别负责从外部设备获取高程数据和纹理数据。QuadeTileSet.m_visibleTiles 中只保存第 0 层的数据 QuadeTile 实例，但是 QuadeTile 实例保存子、父节点之间的引用，使得四叉树中的节点能够方便相互访问。系统在判断该树是否应该继续留在内存，可以从顶层开始提高了判断的效率。假如果第 0 层跟节点都不再平截头体，那么对应金字塔也就不用再判断了，而且在移除时只要该 0 层数据移除就做到了将整棵树从系统移除。综上所述，该方法大大的提高了系统效率，特别是用户在三维场景漫游时，需要快速交互内存中的数据。但是第 0 层的分块不易过多，过多则计算量会增大；也不该太少，太少则不能体现该设计结构的作用，一个合理划分才能有效的提高效率，所以本文在第 0 层以 36° 为单位进行划分。

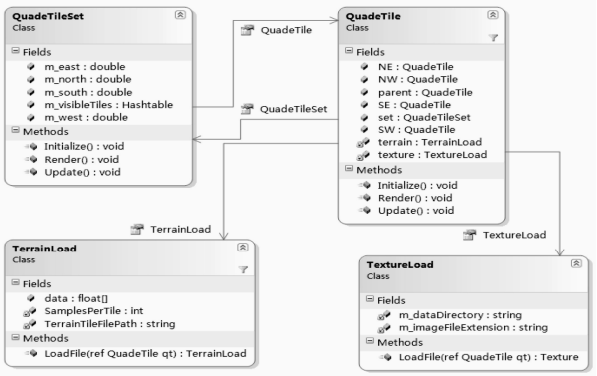

图 3　类关系图

QuadeTileSet 只生成和保存金字塔模型根据节点的实例，金字塔由 QuadeTile 来完成构建，对应于图 3 中 QuadeTile.Upadate 函数。步骤如下：

第 1 步：判断根节点是否已经初始化，否则进行初始化话(负责加载对应的高程数据和纹理数据，转换成系统能够渲染的数据结构，按照图 1 的方式进行划分保存)，是则进入第 2 步；

第 2 步：根据当前视点与区域的位置关系判断是否需要加载更高分辨率的数据；不需要如果当前是第 0 层节点则跳出程序，终止循环，否则返回上一层；需要则进入第 3 步；

第 3 步：判断当前节点是否在平截头体内，不是进入第 6 步，是则进入第 4 步；

第 4 步：判断当前节点的分块是否已经超过最小分块值，是返回上一层，否进入第 5 步；

第 5 步：按照图 1 对当前节点进行分块，生成四个新的实例分别给 NE，NW，SE，SW；新生成每个实例被当作根节点又从第 1 步开始循环；

第 6 步：释放当前节点，程序回到上一层。当第 0 层四个子节点都被构造过之后则该函数运行结束，四叉树金字塔构建完成。

渲染同样也由 QuadeTile 来完成，对应于图 3 中 QuadeTile.Render 函数，相对四叉树金字塔的构建要简单得多，采用深度优先的方法，步骤如下：

第 1 步：从第 0 层根节点开始，判断是否初始化，没有如果是第 0 层跟节点，则跳出程序，否则返回上一层；有则进入第 2 步；

第 2 步：分别判断 NE，NW，SE，SW 四个子节点，若不为 null，则将该节点作为根节点，进入第 1 步，若为 null 进入第 3 步；

第 3 步：渲染对应区域的当前节点的数据，例如 NW 为 null，则渲染当前节点 NW 区域的数据。

第 4 步：当第 0 层的四个子节点都完成遍历则渲染结束，退出该函数。

如果构建的四叉树金字塔模型如图 4 所示，则渲染虚线框中的节点。

如果构建的四叉树金字塔模型如图 4 所示，则渲染虚线框中的节点。

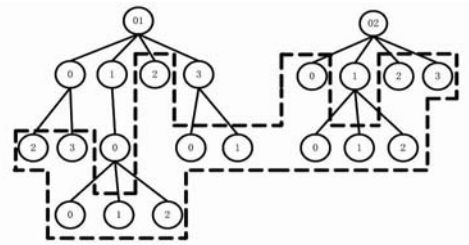

图 4　四叉树结构

高程数据和纹理数据的组织方法，我们在 2.2 节已经详细说明。TextureLoad 采用下面的方法确定需定位的数据：Path.Combine (m_dataDirectory, String.Format(@"{0}\{1}\{1} _{2}.{3}", qt.Level, qt.Row, qt.Col, m_imageFileExtensio n))，其中 qt.Row 和 qt.Col 的值是按照图 2 的公式计算出来，公式中使用的经纬度为对应该块中心点的经纬度值，TerrainLoad 也是该方法准确的定位加载的文件。

5　总结

实验表明分层分块金字塔模型和四叉树相结合组织和管理大规模地形数据通过面向对象方法实现是可行的，满足大规模地形数据在可视化时对实时性、有效加载数据和快速渲染的要求。父节点和子节点通过引用来访问，省去了索引值的计算提高了大大提高程序的运行效率。

参考文献

1 戴晨光,邓雪清,张永生.海量地形数据实时可视化算法.计算机辅助设计与图形学学报, 2004,16(11).

2 Laguna P, et al. Low-pass differentiators for biological signals with known spectra: Application to ECG signal processing. IEEE Trans. on BME (S0018-9294). 2000, 37(4).

3 杜莹,武玉国,王晓明,等.全球多分辨率虚拟地形环境的金字塔模型研究.系统仿真学报, 2006,18(4).

4 Robinson S， Nagel C. 李敏波译.C#高级编程,北京: 清华大学出版社, 2005.