第 35 卷 第 8 期 计 算 机 工 程 2009 年 4 月
Vol.35 No.8 Computer Engineering April 2009
·图形图像处理· 文章编号：1000—3428(2009)08—0230—03 文献标识码：A 中图分类号：TN919.84

基于投影网格的全球多分辨率地形绘制

康 来，吴玲达，宋汉辰，杨 冰
(国防科技大学信息系统与管理学院，长沙 410073)

摘 要：提出一种基于投影网格的全球多分辨率地形绘制算法。利用投影网格剖分球面，快速完成视域内球面的非均匀网格化。对地形数据分层组织、分块存储，通过可视区域计算，实现高效率的地形数据装载和更新。由于每帧网格数量保持基本不变，确保了稳定的帧率。实验结果表明该算法在全球三维地形实时绘制上能取得较好的视觉效果，计算量小、具有较高的效率。

关键词：投影网格；全球地形；实时绘制；多分辨率

Multi-resolution Global Terrain Rendering Based on Projected Grid

KANG Lai, WU Ling-da, SONG Han-chen, YANG Bing
(College of Information Systems & Management, National University of Defense Technology, Changsha 410073)

【Abstract】The paper proposes a multi-resolution global terrain rendering method based on projected grids. Using projected grids to subdivide the spherical surface enables fast asymmetric griding of the part which is visible. To improve the efficiency of data loading and updating, the terrain data is hierarchical organized and stored in tiles, and visible region is calculated beforehand. Rendering frame rate is steady because the amount of grids is settled. The experimental results show that the proposed method is ineffective on global 3D terrain real-time rendering, and it can generate visually convincing results with inexpensive computation.

【Key words】projected grid; global terrain; real-time rendering; multi-resolution

1 概述

全球三维地形绘制是指根据全球地形数据建立三维图形模型，并利用计算机图形技术完成实时渲染的过程。与传统的二维地图相比，计算机三维地形具有真实感强、交互灵活等特点。随着计算机图形硬件、计算机图形学和地形绘制算法不断发展，地形绘制技术已经广泛应用于虚拟现实、飞行模拟、计算机游戏、计算机动画、地理信息系统等领域。“数字地球”概念提出后，全球三维地形可视化逐渐成为研究热点。

近年来，学术界提出了许多有效的三维地形绘制算法。这些算法大致基于以下 3 种思想：

(1)分级算法(hierarchical algorithms)：在常见的数据结构的基础上采用递归的方法将目标区域逐渐细分。

(2)不规则三角网(triangular irregular networks)：充分考虑地形特点，用最少的三角形表现三维地形，这些三角形可能大小形状各异。

(3)基于 GPU 的算法(GPU-based algorithms)：利用图形卡强大的矩阵运算功能和图形卡缓存的快速访问特性，将网格数据保存在显存上，以加快场景绘制速度，文献[1-2]均采用了基于 GPU 的地形绘制方法。

目前，绝大多数三维地形算法都是将平面上的地形作为研究对象，已有的全球地形绘制算法屈指可数。文献[3]通过扩展 ROAM 算法(bin-tree hierarchy)将其运用在全球地形绘制上。文献[4]在 BDAM[5]的基础上提出了 P-Batched Dynamic Adaptive Meshes(P-BDAM)算法，并将其运用到星球地形绘制上。这些方法都是在立方体的基础上，将球体进行多层次剖分。文献[1]在几何裁减[2](geometry clipmaps)的基础上提出了利用球面几何裁减的方法绘制全球地形，这种算法对计算机硬件有很高的要求，难以在中低端个人 PC 上发挥作用。具有全球地形浏览功能的商业软件如 Google Earth, GeoFusion 等，在全球三维地形绘制上取得了很好的效果，但这些技术都是不公开的。

本文利用投影网格对球面上的可视区域进行动态非均匀划分，使显示在屏幕上的地形网格数量基本保持不变，以确保基本恒定的帧率。在将全球地形数据的分层组织的基础上，本文采用可视区域计算及数据调度算法，解决了数据量巨大和硬件实时处理能力有限之间的矛盾。

2 本文主要算法

文献[6]最早利用投影网格思想绘制动态水面，本文选择扩展他所提到的投影网格以绘制全球地形，主要基于以下原因：网格数量保持不变，每帧的计算量大致相同，能确保基本稳定的帧率；世界空间中离视点越远的物体采用越粗的网格表示，避免物体的走样；投影网格实现简单，网格之间不会出现裂缝。

2.1 投影网格球面剖分

将屏幕分割为 $m\times n$ 个大小相同的网格，在屏幕上得到 $(m+1)\times(n+1)$ 个间隔均匀的采样点。对于屏幕上的每个采样点 $\boldsymbol{C}_s$，都能在视锥体的近裁减面上找到与之对应的采样点 $\boldsymbol{N}_s$。过视点 $\boldsymbol{V}$ 与 $\boldsymbol{N}_s$ 的射线与球面的交点便是本文要求的模型空间中的顶点。通过这种“反投影”，球面上 2 个采样点之

基金项目：国家“863”计划基金资助项目(2006AA01Z319)
作者简介：康 来(1983－)，男，博士研究生，主研方向：虚拟现实技术；吴玲达，教授、博士生导师；宋汉辰，副教授；杨 冰，副教授
收稿日期：2008-11-27 **E-mail：**lkang.vr@gmail.com

间的距离与他们跟视点的距离成反比，这样便达到了将视锥体内的球面非均匀分割的目的。投影网格变换的步骤如图 1 所示。将球面上所有采样点连接起来就可以将视域内的球面网格化，视域外的球面将不会被绘制，因此同时达到了视域裁减的目的。

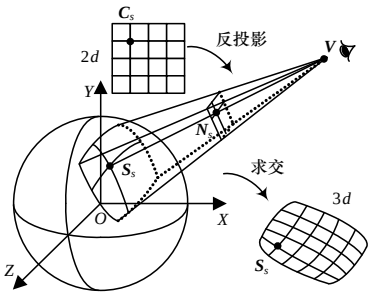


图 1　基于投影网格的球面分割

N_s 与 C_s 之间的关系由如下公式确定:

$$N_s = M_{vp}^{-1} M_{pj}^{-1} M_{mv}^{-1} C_s$$

其中，M_{mv}, M_{pj}, M_{vp} 分别是模型视点矩阵、投影矩阵、视口变换矩阵。

求得 N_s 后便能根据射线 VN_s (其中 V 为视点位置)与球面方程计算出屏幕各个采样点在球面上的对应点，如图 1 所示。首先由 V 和 N_s 可以得到射线 VN_s 的方程:

$$P(t) = V + Dt (t \geqslant 0) \tag{1}$$

其中，$D(d_1, d_2, d_3)$ 为射线方向，且为单位向量。

球面方程为:

$$(P - O) \cdot (P - O) = R^2 \tag{2}$$

将式(1)代入式(2)，得:

$$(V + Dt - O) \cdot (V + Dt - O) = R^2$$

整理得:

$$at^2 + bt + c = 0$$

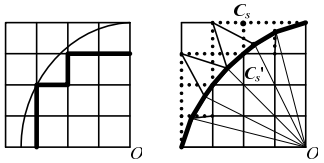
其中，$a = D \cdot D = 1$; $b = 2D \cdot (V - O)$; $c = (V - O) \cdot (V - O) - R^2$。

当 $b^2 - 4ac < 0$ 时，射线与球面不相交;

当 $b^2 - 4ac \geqslant 0$ 时，射线与球面的交点为

$$t_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2}$$

由于屏幕上的采样点间隔均匀，当视点离球面较远时，这将导致球面边缘处出现“锯齿”(见图 2(a))。为了解决这一问题，可以增加屏幕上采样点个数，以更好地拟合球面边缘。但这种方法并不能达到很好的视觉效果，而且会大大增加计算量。因此边缘采样点需要特殊处理。对于球面边缘采样点 C_s，将其位置变换至其与球面最近的点 C_s'。将所有的球面边缘采样点经过这种处理后的结果如图 2(b)所示。



(a)原始边缘采样　　(b)边缘拟合

图 2　边缘采样

在投影网格球面剖分过程中，由于高程的影响，还需要修改视锥体上下截面参数。如图 3 所示，假设球面半径为 R，高程最大为 $R_{\max}$，最小为 $R_{\min}$。为了简化问题，以二维情况为例，视锥体上下截面与球面分别交于 S_1 和 S_2，但这并不意味着只需要绘制 S_1 与 S_2 之间的部分。对于球面顶点 S_1'，当其高程为 $R_{\min}$ 时，该点落入视域；对于 S_2'，当其高程为 $R_{\max}$ 时，该点落入视域。实际上，位于 S_1' 与 S_2' 之间的球面区域都可能位于视锥体内，所以需要延伸视锥体的范围，对 S_1' 与 S_2' 之间的部分进行采样。这样，不管视点位置和视线方向如何变化，视锥体内都不会由于高程起伏而出现“黑边”。

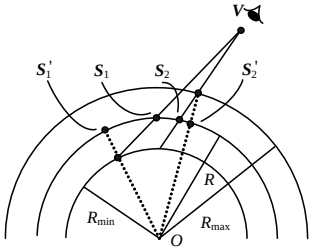


图 3　视域延伸

通过上述反投影、求交、边缘采样及视域延伸等步骤，便能跟据视点的当前位置快速完成视域内球面的网格剖分。

2.2　可视区域计算

可视区域计算是指通过计算直接定位到可视区域，为后续数据调度服务的过程。假设视锥体为圆锥体，可视区域的最大范围为视锥体与球面相切与所包含的曲面，V 为视点位置，Y 轴指向地球北极，地球半径为 1。先考虑一种最简单的情况，视点 V 位于 Y 轴，与球心的距离为 d。对于球面上任意一点 $P_1(P_{1x}, P_{1y}, P_{1z})$，若在世界坐标系下满足:

$$P_{1y} \geqslant \frac{1}{d} \tag{3}$$

则 P_1 可见。在视点与球心距离保持不变的情况下，视点位置的变化可以用参数 (θ, φ) 来表示，同样可视区域的变化也可以用这 2 个参数来表示。下面将推导他们之间的关系。

假定视点 V_2 与球心的距离为 d，方位角为 (θ_v, φ_v)，V_2 可由 V_1 先绕 Y 轴旋转 φ_v，再绕 X 轴旋转 θ_v 得到。同样，V_2 下的可视区域也可由 V_1 下的可视区域经过上述旋转得到。由于 V_1 下球面上点的可视性可由式(3)快速判断，因此对于 V_2 下球面上的点，可以将其变换到 V_1 下进行判断。V_2 下给定球面上任一点 $P_2(\theta, \varphi)$，有:

$$P_2 = \begin{bmatrix} \sin(\theta)\sin(\varphi) \\ \cos(\theta) \\ \sin(\theta)\cos(\varphi) \end{bmatrix} \tag{4}$$

V_1 下与之对应的点 P_1 为:

$$P_1 = R_x(-\theta_v) R_y(-\varphi_v) P_2 \tag{5}$$

其中，R_x 为绕 Y 轴的旋转矩阵；R_y 为绕 X 轴的旋转矩阵。将式(4)和旋转矩阵代入式(5)，有:

$$P_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(-\theta_v) & -\sin(-\theta_v) \\ 0 & \sin(-\theta_v) & \cos(-\theta_v) \end{bmatrix} \begin{bmatrix} \cos(-\varphi_v) & 0 & \sin(-\varphi_v) \\ 0 & 1 & 0 \\ -\sin(-\varphi_v) & 0 & \cos(-\varphi_v) \end{bmatrix} \begin{bmatrix} \sin(\theta)\sin(\varphi) \\ \cos(\theta) \\ \sin(\theta)\cos(\varphi) \end{bmatrix} =$$

$$\begin{bmatrix} \cos(\varphi_v)\sin(\theta)\sin(\varphi) - \sin(\varphi_v)\sin(\theta)\cos(\varphi) \\ \sin(\theta_v)\sin(\varphi_v)\sin(\theta)\sin(\varphi) + \cos(\theta_v)\cos(\theta) + \sin(\theta_v)\cos(\varphi_v)\sin(\theta)\cos(\varphi) \\ \cos(\theta_v)\sin(\varphi_v)\sin(\theta)\sin(\varphi) - \sin(\theta_v)\cos(\theta) + \cos(\theta_v)\cos(-\varphi_v)\sin(\theta)\cos(\varphi) \end{bmatrix}$$

令:

$$f(\theta, \varphi) = \sin(\theta_v)\sin(\varphi_v)\sin(\theta)\sin(\varphi) + \cos(\theta_v)\cos(\theta) + \sin(\theta_v)\cos(\varphi_v)\sin(\theta)\cos(\varphi) - 1/d \tag{6}$$

由式(3)，当 $f(\theta,\varphi)\geqslant 0$ 时，$\boldsymbol{P}_2$ 可见。据此可以计算出视点在不同位置下的可视区域。装载地形数据时，只需要装载可视区域内的地形数据，以提高绘制效率。

2.3 地形数据组织与调度

全球地形数据按照金字塔结构组织，并采用四叉树结构模型来存储。根据视点距离选择不同分辨率的数据，以达到既不产生视觉损失又能加快场景绘制速度的目的。假设原始数据的边长为 (2^n+1)，地形数据的金字塔结构可以按如下方法建立：对原始地形数据进行双线性插值，使数据边长变为原始数据的一半，得到下一层次的地形数据，用同样的方法对插值所得的地形数据进行递归处理，直到数据分辨率达到最低要求。对于金字塔结构中每个层次，按照四叉树结构存储。四叉树结构建立如下：将原始数据分割为 4 个大小相同的子块，则 4 个子块的行列数均变为原来的 1/2，然后再对 4 个子块进行递归分割，直到每个子块的大小达到最小的分割要求。

装载地形数据时只需要装载当前视点下可见区域的数据。为了使不同视点下数据装载开销基本不变，需要根据可视区域的大小选择相应的地形数据层次。当可视区域面积较大时，采用分辨率较低的数据；当可视区域面积较小时，采用分辨率较高的数据。定义采样密度为单位面积上(经纬度乘积)采样点的个数，$d_l(l\geqslant 1)$ 为第 l 层地形数据的采样密度，显然有：

$$d_l = 4^{l-1}d_1(l\geqslant 1)$$

定义每次数据装载量不超过 m（m 的最小取值为地形数据中第一层的数据量），可视区域的面积为 s，根据下式确定数据层次：

$$L=\left\lfloor \log_4\frac{m}{d_1 s}+1\right\rfloor$$

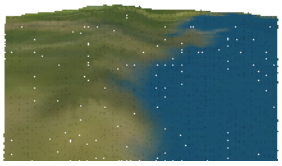
m 的取值要根据计算机性能在帧率和视觉效果之间折中。当视点保持不动时，地形数据只需要装载一次，不需要更新。视点移动时，由于视点位置往往是连续变化的，因此场景前后两帧的地形数据大部分都是相同的，这部分数据不需要更新。这种数据更新方法在文献[7]中有详细介绍。

3 实验结果

笔者采用 OpenGL 图形库，在 VC++6.0 环境下编写了测试程序。实验环境是：Windows XP 操作系统，Pentium 4 主频 2 GHz 处理器，256 MB 内存，NVIDIA GeForce3 Ti 500 显卡。实验程序实现了全球地形网格的动态创建、地形数据的动态调度，另外还加入了灵活的摄像机控制，从而实现了视点相关的全球三维地形多分辨实时绘制。视线俯仰角保持不变时，绘制的网格数量不会因为视点高度变化而变化；视点高度保持不变时，绘制的网格数量随视线俯仰角变化而变化，但不会超过预先设定的采样网格数量。所以本文的方法可以保证基本的绘制帧率，并能根据 PC 机的性能和用户要求，调节采样网格数量以达到最好的视觉效果。图 4 是采样点数量为 16 641(129×129)时用户从不同角度观察地球的实验程序截图。

(a)视点离地面很远

(b)视点离地面较远



(c)视点离地面较近

图 4 视点在不同位置的视图

4 结束语

本文详细介绍了基于投影网格的全球地形绘制方法，该方法实现简单，视觉效果较好。同时，本文的方法也适用于数字地球上洋面的绘制。随着计算机硬件的不断发展，GPU 强大的计算处理能力在计算机图形处理上具有越来越明显的优势，如何将本文的方法移植到 GPU 上，以进一步提高绘制效率，是今后的工作的重点。另外，本文未考虑地形遮挡因素，这将在今后的工作中进一步完善。

参考文献

[1] Clasen M, Hege H C. Terrain Rendering Using Spherical Clipmaps[C]//Proc. of the Conference of the European Association for Computer Graphics. Vienna, Austria: [s. n.], 2006: 91-98.

[2] Asirvatham A, Hoppe H. Terrain Rendering Using GPU-based Geometry Clipmaps[M]. [S. l.]: Addison-Wesley, 2005.

[3] Hill D. An Efficient, Hardware-accelerated, Level-of-detail Rendering Technique for Large Terrains[D]. Toronto, Canada: University of Toronto, 2002.

[4] Cignoni P, Ganovelli F, Gobbetti E, et al. Planet-sized Batched Dynamic Adaptive Meshes(P-BDAM)[C]//Proc. of the IEEE Conference on Visualization. Seattle, USA: [s. n.], 2003: 147-155.

[5] Cignoni P, Ganovelli F, Gobbetti E, et al. BDAM-batched Dynamic Adaptive Meshes for High Performance Terrain Visualization[C]// Proc. of the Conference of the European Association for Computer Graphics. Granada, Spain: [s. n.], 2003: 505-514.

[6] Johanson C. Real-time Water Rendering[D]. Lund, Sweden: Lund University, 2004.

[7] Tanner C C, Migdal C J, Jones M T. The Clipmap: A Virtual Mipmap[C]//Proc. of the Annual Conference on Computer Graphics and Interactive Techniques. Orlando, FL, USA: [s. n.], 1998: 151-158.

编辑　任吉慧