

# 基于 Clipmap 的海量地形及纹理实时绘制方法

宫晓辉,温慧明,于 卓

(国网电力科学研究院 中电普华信息技术有限公司,北京 100191)

**摘 要:**随着数据采集手段的提高,所获得的地形几何数据与纹理数据已远超出内存容量,如何对海量地形与纹理数据进行实时绘制一直是研究的热点问题,文中致力于为该问题提供解决方案。clipmap 结构具有简单且利于硬件加速的特点,文中基于 clipmap 结构给出一种海量地形几何数据与纹理数据的绘制方法,并针对纹理数据量大的问题,给出一种基于缓冲区的纹理调度策略。实验表明,基于统一 clipmap 结构的地形绘制方法能较好地对海量地形数据及纹理数据进行实时绘制。

**关键词:**海量地形;clipmap;层次细节

中图分类号:TP391.9

文献标识码:A

文章编号:1673-629X(2012)10-0022-05

## Real-time Rendering Method of Massive Terrain and Texture Based on Clipmap

GONG Xiao-hui, WEN Hui-ming, YU Zhuo

(China Power, State Grid Electric Power Research Institute, Beijing 100191, China)

**Abstract:** Nowadays, with the rapid development of data acquiring skill, the data of massive terrain and its corresponding texture data is so big that can not fit in memory. How to render these huge data in real time is a hot topic in research area. It aims to provide a solution for the problem. Clipmap structure is simple and conducive to the hardware acceleration. Based on clipmap structure, give a drawing method of a massive terrain geometry data and texture data, and for the large amount of texture data give a kind of scheduling policy based on the texture of the buffer. The experiment shows that the terrain rendering method based on unified clipmap structure can render real-time massive terrain data and texture data better.

**Key words:** massive terrain; clipmap; level of detail

## 0 引 言

大规模自然环境的高效绘制一直是计算机图形学和虚拟现实中的重要研究内容,在军事仿真、计算机游戏、数字地球等领域中扮演着重要的角色。随着数据采集设备的快速发展,所获得的数据量也急剧增加,特别对于地形数据来说,地形的几何数据已经达到 TB 级,而对应的纹理数据更是几何数据的几十甚至上百倍。这些数据已远远超出当前硬件的内存容量,如何对海量地形的几何与纹理数据进行实时绘制,是当前研究的热点,也是文中需要解决的主要问题。

对海量地形进行高效的绘制,通常采用层次细节(Level of Detail, LOD)技术对地形数据进行预处理。在预处理过程中,首先对地形数据进行简化,形成分辨率较低,顶点数量较小的 LOD 模型。随后在运行过程

中,根据运行时刻绘制资源的不同情况调入不同分辨率的 LOD 模型,从而降低每个时刻绘制的面片数量。在简化过程中,根据方法的不同,又可分为全局简化方法与局部简化方法。运行过程中,根据视点位置的不同,又可分为基于视点的连续 LOD 显示方法与离散 LOD 显示方法。

2004 年 Losasso<sup>[1]</sup>给出了 Geometry ClipMap (GC)方法,这是一种基于视点的连续 LOD 方法,将地形看作是一张由高程数据组成的图像,采用 clipmap 结构以一组嵌套网格的方式对地形几何数据存储并绘制。在预处理阶段,将地形数据组织为以  $2n$  ( $n$  是大于等于 1 的整数)为步长进行采样的 mipmap (金字塔)结构<sup>[2]</sup>,同时以视点为中心,在 mipmap 结构的基础上,按精细层在内,粗糙层在外,层层嵌套的方式组织为 clipmap 结构。在 clipmap 结构中,每层数据所包含的顶点数量相同,用  $\text{clipsize} * \text{clipsize}$  表示,用  $\text{cliplevel}$  表示某层数据在 clipmap 结构中的层数。对于构建好的 clipmap 结构而言,以视点为所在层中心层(记为精

收稿日期:2012-02-03;修回日期:2012-05-10

基金项目:国家自然科学基金资助项目(60873159)

作者简介:宫晓辉(1985-),女,硕士,研究方向为软件工程、图像处理、电力信息化。

细层,即  $\text{cliplevel}=0$ ),向外的分辨率依次递减,层次数增大,采样步长随之增加。

Asirvatham<sup>[3]</sup>将 clipmap 结构每层数据分解成多个规则块,并利用 GPU 对每个块数据的更新与绘制进行加速,可将除解压外的全部操作在 GPU 上进行,提高了绘制速度。但由于解压操作的特殊性,仍然将解压操作放在 CPU 上执行。文献[4]同样利用 clipmap 结构组织地形数据,在 GPU 中将 clipmap 结构各层数据做为单独纹理,分别进行存储与更新。Livny<sup>[5]</sup>将高程数据转换为纹理,利用 GPU 完成数据的预取与裂缝的混合,但对纹理的大小仍有限制。康宁<sup>[6]</sup>使用“裙”的方法,对 clipmap 结构每层数据都额外增加一个类似包围盒的结构,并使用包围盒结构中的数据与产生裂缝的区域进行融合,降低计算量的同时还较好地解决了裂缝,但数据精度仍有待提高。在 2009 年袁建锋<sup>[7]</sup>采取直接改变 clipmap 结构不同层边界对应顶点高程的方法避免裂缝,但在某些情况下仍会产生视觉不连续的情况。王春<sup>[8]</sup>通过降低 clipmap 结构两相邻层中较精细层对应顶点高度值的方法,避免了裂缝的产生。张浩<sup>[9]</sup>给出了一种不对称 clipmap 结构,在构造 clipmap 结构时,不再以视点为中心,而是以视点偏移后得到的中心点为新的中心,并通过使用不对称 clipmap 结构的更新方法,提高了原方法对网格的利用率。Clasen<sup>[10]</sup>还将 clipmap 结构扩展到球面,将球面地形数据通过映射关系建立以视点为中心的同心圆而不是 clipmap 结构中嵌套的正方形,同时给出了基于同心圆的更新与绘制方法,对地球表面的高程数据进行实时绘制。

目前在海量地形绘制中,对于地形几何数据来说,clipmap 具有结构简单且利于硬件加速的特点<sup>[11]</sup>,被广泛使用在上述所介绍方法中。对于地形纹理,常用四叉树结构进行组织<sup>[12]</sup>。但当地形几何数据和纹理数据都较大时,数据调度会成为绘制速度的瓶颈。因此需要一种更简单的方法对大规模地形几何和纹理数据进行高效绘制。

文中使用 clipmap 结构作为海量地形几何数据的 LOD 结构,特别对于地形纹理数据,同样使用 clipmap 结构进行组织。针对纹理数据量大,调度数据消耗大量时间的问题,给出一种基于缓冲区的纹理调度方法。

实验表明,文中给出的基于统一 clipmap 结构的地形绘制方法,可对海量地形几何数据与纹理数据进行实时绘制。

## 1 几何数据 clipmap 结构构建与绘制

### 1.1 clipmap 结构构建

地形数据 clipmap 结构的建立是通过将地形网格

数据进行多次采样得到,即根据地形大小,生成 clipmap 结构多层数据,每层由大小为  $\text{clipsize} * \text{clipsize}$  的“口”字形结构组成,这些结构彼此间嵌套。为保证两层网格之间的对齐,clipsize 最好为奇数,并可利用硬件对绘制进行加速。当地形数据量为  $M * N$  个顶点时,clipmap 大小 clipsize 及层数 cliplevel 有关系如公式(1):

$$(\text{clipsize}-1) * 2^{\text{cliplevel}} \geq \max(M, N) \quad (1)$$

### 1.2 clipmap 结构绘制

在绘制时,需要为 clipmap 结构每层都构建一个大小为  $\text{clipsize} * \text{clipsize}$  的顶点队列和一个大小为  $2 * \text{clipsize} * (\text{clipsize}-1)$  的索引队列。

构建 clipmap 结构每层顶点索引队列时,使用三角条带方式对 clipmap 结构每层数据进行组织,同时生成一个索引队列并同时计算三角条带的绘制序列。如图 1 所示,当  $n=9$  时的顶点队列,使用三角条带方式绘制第一行网格的索引顺序为(0,9,1,10,2,...,8,17)。在每层大小为  $\text{clipsize} * \text{clipsize}$  的情况下,clipmap 结构每层都有  $\text{clipsize}-1$  个条带需要绘制,每个条带的大小为  $2 * \text{clipsize}$ 。

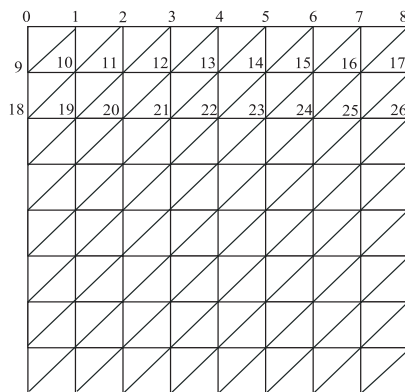


图 1  $n=9$  时的条带顶点关系示例

对非精细层 clipmap 数据进行绘制时,各层都分割为四个子条带,并对每个子条带都构建各自的顶点队列和索引队列。

## 2 clipmap 纹理构建

通常纹理数据远大于几何数据本身,且纹理的加速方法较少,同时还存在显卡对纹理数据大小的限制,因此需要对海量纹理数据给出一种高效的组织与调度方法。若采用纹理压缩的方法,运行阶段对纹理的解压操作也会消耗相当多计算时间,所以对纹理数据的压缩/解压方法在引擎中也并不适用。由于已经使用 clipmap 结构组织地形的几何数据,在绘制时刻将地形纹理数据与几何数据进行绑定,文中选择 clipmap 结构对大规模纹理数据进行组织。

在预处理阶段,对纹理进行空间划分(如图 2 所

示)后再按照四叉树的方式进行组织。但是分块后的纹理数据依然很大,因此必须保证分块后的纹理数据满足硬件的限制。所以文中方法首先对分块后的每块数据建立三级 mipmap 结构, mipmap 结构的纹理数据仍然不能直接使用,然后将多个纹理块的 mipmap 数据组织为 clipmap 纹理数据。

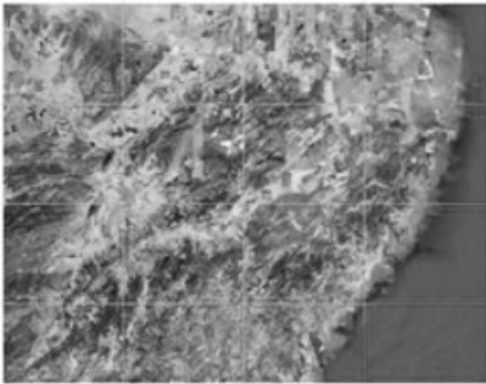


图 2 纹理分割

文中采用构建纹理缓冲区的方式,对纹理进行组织与调度。因此在运行时刻,构建一个 4 \* 4 块大小的纹理缓冲区,每块纹理数据的大小为 512 \* 512,在此缓冲区的基础上,实时生成对应的 clipmap 结构并绘制。

3 clipmap 结构更新方法

在运行时刻当视点移动时,需对地形的几何数据和纹理数据同时更新,文中采用了统一的 clipmap 结构,对于地形数据,采用“L”行更新。对于纹理数据,针对其数据量大的特点,首先更新其纹理缓冲区,再由缓冲区中数据生成对应的 clipmap 结构纹理,最后对 clipmap 纹理进行更新与绘制。其流程如图 3 所示:

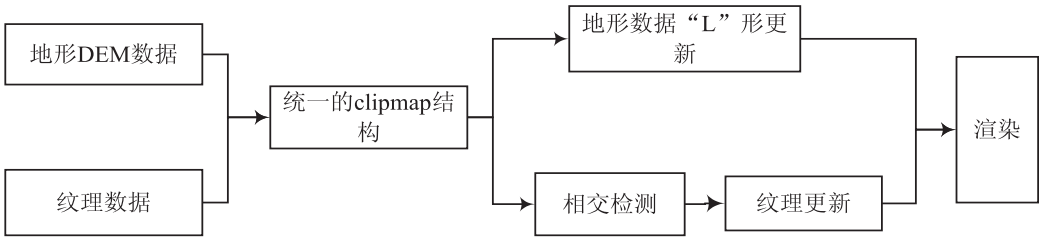


图 3 clipmap 结构更新流程

3.1 几何 clipmap 结构更新

对地形网格数据 clipmap 结构更新时,对 clipmap 结构各层次进行以下操作:

- 1) 计算所更新的顶点列(行)数。  
clipmap 结构每层维护自己中心点的坐标( center. x, center. y ),每当视点移动时,比较中心点坐标和当前视点(eyex, eyey),得到视点与上次绘制时该层中心点偏差的距离,进而通过下面公式(2)得到该层在

两个方向上需要更新的顶点的列数 move\_m 和行数 move\_n :

$$\begin{aligned} \text{move\_m} &= (\text{eyex} - \text{centerx}) / 2^{\text{cliplevel}} \\ \text{move\_n} &= (\text{eyey} - \text{centery}) / 2^{\text{cliplevel}} \end{aligned} \tag{2}$$

2) 更新顶点队列。  
得到所需更新的行(列)后,对顶点更新,顶点队列中有大量顶点是重复的,是可以不用再次更新的,具体更新步骤如下:

- (a) 根据 move\_m, move\_n 符号判断更新方向;
- (b) 计算需要绘制的新顶点,如 move\_m > 0 则新顶点为该层当前数据右侧 move\_m 列, n 行采样率为 1/2cliplevel 的所有顶点;
- (c) 计算新顶点在 clipmap 层次中的位置;
- (d) 按位置替换原有的顶点。

视点左移,上移和下移的情况类似,只是需要判断好所需加入新点的范围。再依次将各层完成以上的操作,直至最外层更新完毕。

3) 更新索引队列。

为避免更新索引队列时对显存的读操作,使用在绘制时将一行三角条带分解成两条分别绘制的方法来处理索引问题。

3.2 纹理 clipmap 结构更新

3.2.1 纹理缓冲区更新

绘制时,将纹理缓冲区中的数据导入显存中,并根据当前绘制顶点的位置,动态绑定不同层次不同块的纹理。当视点进行移动时,需要进行纹理缓冲区和显存中纹理的更新。由于纹理缓冲区的大小有限,因此在每次更新前都需对缓冲区中的数据进行判断。同时根据视点的位置信息,动态绑定该块的纹理。过程中需检测视点是否有变化,如果有,则将新的纹理块从外存调入内存。图 4 为纹理缓冲更新的示例图,从图中

可以看出,当视点往右移动时,所更新的浅灰色数据为从外存调入内存,此时视点所在纹理并没有更新。而当视点往下移动时,除将所更新的纹理块调入内存外,还需将视点所在纹理进行更新,如图中深灰色部分所示。

更新纹理 clipmap 结构每层数据时的步骤如下:

- (1) 计算当前视点的块号。

纹理 clipmap 结构各层都需保存当前视点所在的纹理分块块号( index\_x, index\_y )。当视点移动时,计算视点移动后所在的块号( index\_x1, index\_y1 )。



比较  $(index\_x, index\_y)$ ,  $(index\_x1, index\_x2)$ , 如果相同, 则不需要对纹理缓冲区进行更新; 如果不相同则按层次更新纹理缓冲区。视点所在块号计算如下公式(3):

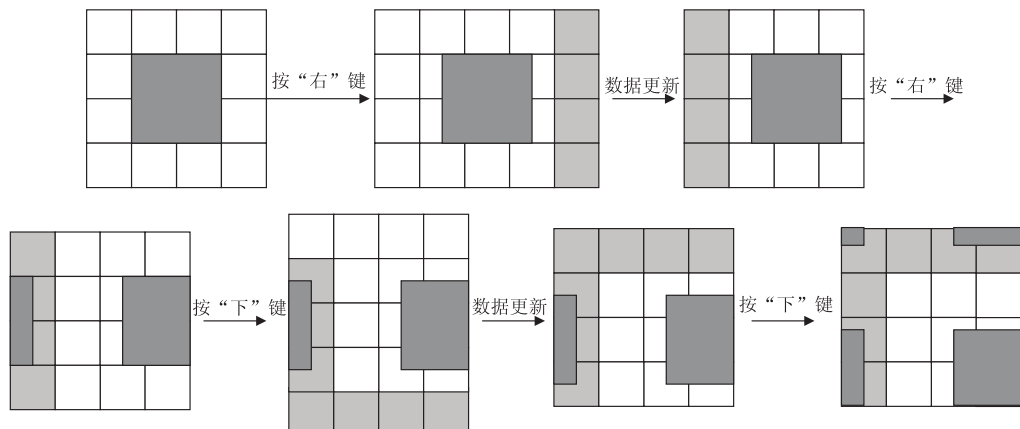


图 4 clipmap 纹理缓冲更新

$$index\_x = (eye\_x + size\_x / 2) / tex\_x \quad (3)$$

$$index\_y = (eye\_y + size\_y / 2) / tex\_y$$

(2) 更新纹理缓冲区。

由上步计算后得知需要加载新的纹理块后, 需要更新纹理缓冲区。此时比较  $(index\_x, index\_y)$  和  $(index\_x1, index\_y1)$ , 可判断纹理缓冲区的移动方向。更新的顺序为先沿  $x$  方向更新, 再沿  $y$  方向更新。考虑到绘制时纹理的绑定与纹理缓冲区中的纹理块的存储位置有关, 因此需对仍在缓冲区但位置不同的纹理数据进行复制后, 移动到相应的位置。并将不在纹理缓冲区中的数据从外存调入。更新后, 新的纹理数据被导入显存, 其纹理 ID 被记录在纹理缓冲区中, 当绘制时, 根据相应的纹理 ID 进行绑定。

(3) 更新纹理坐标。

因为纹理和网格数据的对应关系不因视点移动改变, 故纹理坐标的计算方法与建立 clipmap 相同。但网格数据的更新产生了新的坐标点, 因此需对这些坐标点重新计算纹理坐标, 并存入相关的纹理坐标数组中。

(4) 纹理绘制数据的更新。

当视点移动后, 绘制所需的各层 clipmap 数据也需要进行更新。考虑到回字形区域纹理缓冲区的相对位置情况较为复杂, 决定采用根据视点位置, 对各层各个 clipmap 数据的参数值进行重新计算, 来完成纹理绘制数据的更新。

### 3.2.2 纹理坐标更新方法

在更新纹理坐标过程中, 如图 2 所示纹理被分割成更小的部分, 对每一部分都应该进行各自纹理坐标的更新, 此外还需要将顶点的坐标和顺序统一并与几何数据相对应, 计算十分复杂。文中对 clipmap 结构各层与纹理数据的相交情况进行总结, 根据文中给出的

相交情况及相交顶点, 可以自动计算出纹理顶点, 从而加快计算速度。

(1) 纹理数据与 clipmap 结构相交顶点数为 2 的情况: 在相交顶点为 2 个的情况下, 根据纹理数据与

clipmap 结构相交顶点位置关系的不同, 分为四个方向。此时 clipmap 结构与纹理数据部分重叠, 因此在绘制纹理数据时仅对非重叠的一个子纹理条带进行绘制。

(2) 纹理数

据与 clipmap 结构相交顶点数为 3 的情况: 在相交顶点数量为 3 的情况下, clipmap 结构将纹理数据分割为两个条带, 根据 clipmap 结构与纹理数据相交位置的不同, 分为四个方向, 对分割后形成的两个纹理子条带进行绘制。

(3) clipmap 结构与纹理数据相交顶点数为 4 有七种情况, 对七个相交方向分割后生成的三个纹理子条带进行绘制。

最后给出 clipmap 结构各层与纹理数据相交判断的算法流程:

步骤 1: 判断当前 clipmap 纹理数据的层次, 如果是精细层, 转到步骤 2, 如果不是精细层, 转到步骤 5;

步骤 2: 求得精细层数据的空间坐标, 同时求得 clipmap 几何数据的包围盒;

步骤 3: 对两包围盒进行相交判断, 求得相交顶点的数量, 相交方向与相交顶点坐标的数组;

步骤 4: 根据顶点数量、相交方向及顶点坐标数组自动生成纹理条带并绘制;

步骤 5: 得到当前 clipmap 纹理数据四个子块的空间坐标, 同时求得 clipmap 几何数据的包围盒;

步骤 6: 对于每个纹理子块, 分别与 clipmap 几何数据的包围盒进行求交, 得到各自的相交顶点的数量、相交方向与相交顶点坐标的数组;

步骤 7: 根据顶点数量、相交方向及顶点坐标数组, 对四个纹理子块自动生成各自的纹理条带并绘制。

## 4 实验结果与分析

本实验对引擎中大规模地形与纹理进行了测试。对文中方法进行测试, 使用的实验平台 CPU 为 3.0GHZ, 内存为 2GB, 显卡为 NVIDIA Geforce 8800GTS, 显存 356MB, 所有实验均在 Windows XP 下基于 Open-

GL 实现。

大规模地形数据顶点数量为  $16384 * 16384$ , 纹理数据的分辨率为  $16384 * 16384$ , 将其中的纹理数据组织为  $32 * 32$  共 1024 块  $512 * 512$  像素的 24 位 BMP 图像。绘制速度在 30 帧/秒。图 5 给出 clipmap 网格效果图, 图 6 给出总体效果图。

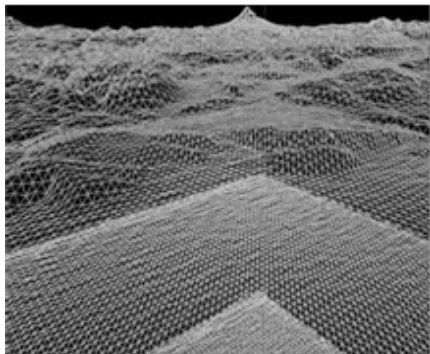


图 5 地形数据 clipmap 结构网格绘制效果

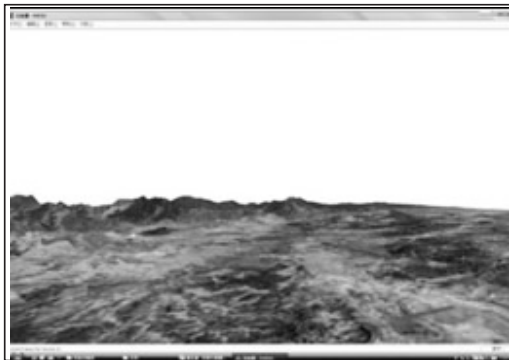


图 6 网格与纹理综合效果

## 5 结束语

文中给出一种基于 clipmap 结构的海量地形几何及纹理实时绘制方法, 通过统一 clipmap 结构, 加快了几何数据的更新效率, 提高绘制速度。同时针对纹理

数据量大的问题, 给出一种基于缓冲区的纹理调度方法。实验表明, 文中方法较好地海量地形几何数据与纹理数据进行实时绘制。

## 参考文献:

- [1] Losasso F, Hoppe H. Geometry Clipmaps: Terrain Rendering Using Nested Regular Grids[J]. ACM Transactions on Graphics, 2004, 23(3): 769-776.
- [2] Williams L. Pyramidal Parametrics[J]. Computer Graphics, 1983, 17(3): 1-11.
- [3] Asirvatham A, Hoppe H. GPU GEM 2[M]. [s. l.]: Addison-Wesley, 2005.
- [4] Seoane A, Taibo J, Hernández L. Hardware-independent Clipmapping[J]. Journal of WSCG, 2007, 15(1-3): 177-185.
- [5] Livny Y, Sokolovsky N, Grinshpoun T. Persistent Grid Mapping: A GPU-based Framework for Interactive Terrain Rendering[J]. The Visual Computer, 2008, 24(2): 139-153.
- [6] 康宁, 徐青, 周杨. 一种基于图形硬件的海量地形实时可视化算法[J]. 系统仿真学报, 2007, 19(17): 3988-3992.
- [7] 袁建锋, 崔铁军, 姚慧敏. 一种基于 GPU 的大规模地形实时生成算法[J]. 海洋测绘, 2009, 29(1): 35-38.
- [8] 王春, 马纯永, 陈戈. 基于 GPGPU 的海量山地地形数据的实时绘制算法[J]. 计算机应用, 2009, 29(8): 2105-2108.
- [9] 张浩. 不对称的 GeometryClipmap 算法[D]. 武汉: 华中科技大学, 2005.
- [10] 张莉, 唐立文. 基于四叉树的海量空间数据无缝组织研究[J]. 计算机技术与发展, 2011, 21(1): 77-81.
- [11] 白皓, 龚光红, 丁莹. 基于 Clipmap 的大规模地形可视化技术研究[J]. 中国体视学与图像分析, 2009, 14(2): 202-209.
- [12] 吴颖, 张新家, 茹芬. 基于四叉树分割的连续 LOD 漫游地形绘制[J]. 计算机技术与发展, 2011, 21(4): 5-9.

(上接第 21 页)

- [3] 欧阳彝华, 黄芳, 周敏. 基于灰度直方图的肝脏图像检索[J]. 计算机技术与发展, 2009, 19(9): 125-127.
- [4] Du Yajuan, Pan Quan, Zhang Hongcai. Application of a new moment invariant features on image recognition[J]. Journal of Systems Engineering and Electronics, 1999, 31(10): 71-85.
- [5] 李鑫环, 陈立潮. 基于多小波分析与 SOFM 的 MR 图像分割算法研究[J]. 计算机技术与发展, 2009, 19(9): 104-107.
- [6] Yang Yubin, Lin Hui, Zhu Qing. Content-based 3D model retrieval[J]. Chinese Journal of Computer, 2004, 27(10): 1298-1310.
- [7] Ankerst M. 3D Shape Histograms for Similarity Search[C]//SSD'99. Hong Kong: [s. n.], 1999: 207-226.

- [8] 张鑫, 陈梅, 王翰虎, 等. 基于视觉特征和领域本体的 Web 信息抽取[J]. 计算机技术与发展, 2011, 21(2): 58-61.
- [9] 于二丽, 周宁宁. 基于 Hausdorff 距离的图像匹配并行算法设计与实现[J]. 计算机技术与发展, 2011, 21(9): 28-31.
- [10] 王冠, 丁友东, 魏小成. 基于改进 Sobel 算子的文物图像检索[J]. 计算机技术与发展, 2011, 21(10): 51-54.
- [11] Ankerst M, Kastenmuller G. 3D Shape Histograms for Similarity Search and Classification in Spatial Databases[C]//SSD'99. London: Springer-Verlag, 1999: 207-226.
- [12] 崔晨阳. 三维模型检索中关键技术的研究[D]. 杭州: 浙江大学, 2005.

# 基于Clipmap的海量地形及纹理实时绘制方法

作者: [宫晓辉](#), [温慧明](#), [于卓](#)  
作者单位: [国网电力科学研究院 中电普华信息技术有限公司, 北京 100191](#)  
刊名: [计算机技术与发展](#)  
英文刊名: [Computer Technology and Development](#)  
年, 卷(期): 2012(10)

本文链接: [http://d.g.wanfangdata.com.cn/Periodical\\_wjtz201210008.aspx](http://d.g.wanfangdata.com.cn/Periodical_wjtz201210008.aspx)