基于四叉树的缓存复用机制地形算法

宋省身[1],全吉成[1],赵秀影[1],王宏伟[1],王 宇[2]
(1. 空军航空大学 航空航天情报系,吉林 长春 130022;
2. 军事仿真技术研究所,吉林 长春 130022)

摘 要:为了减少地形数据的冗余,实现缓存复用,在研究了 Geo-CLipmap 算法的基础上,针对其采用嵌套网格一次只能渲染单张高程图的方案,提出了一种块状四叉树的数据结构,同时分开存储节点的位置信息和高程信息,有效地减少了数据大小和数据交换,实现了顶点和索引缓存的复用;利用掩膜填充不同分辨率网格之间的缝隙,避免了加入光照后的几何失真,并使用几何过渡消除几何体的跳变。经过实验测试,该算法能有效地压缩数据量,在稳定帧速的情况下,实现地形的无缝渲染。

关键词:大规模地形绘制;细节层次;缓存复用;几何过渡
中图分类号:TP391.9 **文献标识码**:A **文章编号**:1673-629X(2014)02-0046-04
doi:10.3969/j.issn.1673-629X.2014.02.011

A Terrain Algorithm of Cache Reusing Mechanism Based on Quadtree

SONG Xing-shen[1], QUAN Ji-cheng[1], ZHAO Xiu-ying[1], WANG Hong-wei[1], WANG Yu[2]
(1. Dept. of Aeronautics & Astronautics Intelligence, Aviation University of Air Force, Changchun 130022, China;
2. Dept. of Military Simulation Technology Institute, Changchun 130022, China)

Abstract: In order to decrease the redundancy in terrain data and reuse the cache, based on the research of Geo-Clipmap algorithm, which only renders single height map by using nested meshes, a data structure of chunk quadtree is proposed, which stores the position information and evaluation information of nodes separately, efficiently subtracting the size of datum and the transmittal between them by reusing vertex cache and index cache for rendering. By using masks filled in the seams between multiple resolution meshes, the algorithm successfully avoids geometry aliasing under illumination, meanwhile, it eliminates popping by using geometry morphs. As the experiment shows, this algorithm can effectively compress the size of data and render terrain seamlessly while maintaining the frame rate smoothly.

Key words: rendering of large terrain; LOD; cache reuse; geometry morph

0 引 言

地形的实时渲染是计算机图形学的重要分支,应用广泛。实现大规模的地形渲染面临的主要问题,是如何在保证帧速稳定的情况下,把高分辨率的地形数据完全地显示出来,同时避免多个像素重叠时造成的失真。为此提出了很多基于多分辨率 LOD 模型的地形简化算法,如实时优化自适应网格算法(ROAM)[1]、分块地形算法(Chunked LOD)[2]和 Geo-Clipmap 算法[3-6]等。

随着 GPU 的发展,过去的 LOD 算法不能为图形硬件提供足够的三角形数目或是占用太多 CPU 运算资源,成为了制约渲染效率的瓶颈。近年来的渲染技术多是利用着色器语言来处理地形数据并减少 CPU 负载,以充分利用 GPU 的批处理能力[7]。如 Geo-Clipmap 算法是在缓存中构建一张高度图的多分辨率金字塔,观察时以视点为中心生成嵌套网格,虽然利用图像压缩可以把整张纹理存在显存中,但对于大规模的地形调度这种方法就略显不足[8-9]。

文中算法是基于四叉树结构进行地形数据的组织,以块状模型为处理单元,这点类似于 Chunked

收稿日期:2013-04-18 **修回日期**:2013-07-25 **网络出版时间**:2013-11-29
基金项目:国家自然科学基金资助项目(61172508)
作者简介:宋省身(1990-),男,河南濮阳人,硕士研究生,研究方向为地形三维可视化;全吉成,教授,博士研究生导师,研究方向为数字地球军事应用;赵秀影,副教授,博士,研究方向为机械电子与信息处理。
网络出版地址:http://www.cnki.net/kcms/detail/61.1450.TP.20131129.1006.044.html

LOD,同时数据存取时采用双流缓存,将位置信息与高程信息分开存储,实现顶点和索引缓存的复用,便于对大规模数据进行存取。并利用此方法实现了对单幅高分辨率高程图分块渲染,验证了该算法的可行性。

1　算法介绍

针对大规模的地形数据,不同区域的分辨率需求也不尽相同,在渲染时通过实时地对原始数据进行插值与采样以实现多分辨率表达,一般的系统配置下几乎满足实时性需求。将插值采样的数据进行分层组织与存储,构建多分辨率金字塔可以弥补之前的不足,虽然这增加了数据量的大小,但减小了数据在传输与处理时网络带宽与图形硬件的压力。将世界分割为多个区域后,即可由着色器决定渲染哪些区域并判定渲染的细节程度。

1.1　数据结构与组织

假设将世界分割为许多的小区域,每个都以长方体表示,称之为区块(sector),将这些区块形成一个层次树,每个区块都由四个更小的子区块构成。进行地景渲染时,首先判断哪些区块是可见的,如果父区块不可见,那么孩子也不可见,反之则都可见。而当父区块与视椎体相交时,就选择它的子区块,将它继续当作父区块来测试其可见性。直到所有可见块被找到或完成了所有节点的遍历。

采用 Chunked LOD 算法构建了块状四叉树结构,即不再以单个顶点或三角形为处理单元,而是以包括纹理贴图在内的网格模型,将三角形打包成条带(strip)一次性完成渲染,充分利用 GPU 的批处理能力[10],如图 1 所示。

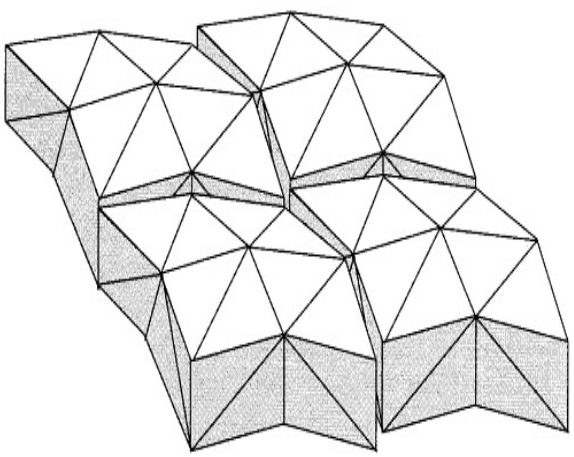


图 1　Chunked LOD 网格模型的结构图

把地形作为一张 2D 的高度图来处理,并预先对其使用 Geomipmap 进行滤波,构建一张包含了 L 层的 mipmap 金字塔。为这里,将每层的数据都以适当的尺寸保存为 2D 纹理,而不是认为的通过顶点数组将之线性化为 1D 的缓冲。金字塔的每一层各包含了 $n \times n$ 的采样点,为之存储了一张单通道的高程图,一张三通道的顶点法向图和三通道的纹理图。

这样,可以在着色器中定义如下的节点数据结构:

```
struct VSOUTPUT {
vector position:POSITION;
float height:TEXCOORD0;
float3 normal:TEXCOORD1;
float2 uv:TEXCOORD2;
}
```

让法线图与纹理的分辨率为高程的两倍,因为每个顶点只有一条法线的效果很模糊,同时在加入光照后也会很粗糙,而两倍分辨率的法线图可以通过插值创造更加柔和的模型。低分辨率的纹理限制视点的范围,使各个表面类型产生大面积的模糊,故纹理分辨率也不应小于高度图。

1.2　缓存的复用

对于一小部分区域单张高程图即可覆盖,一次性载入显存使用蛮力渲染并不存在太大的问题。然而面对大规模的地形这种做法并不现实,而进行分块载入后,每次都锁定一块顶点或索引缓存而修改其中部分的值在大多数显卡上都很低效。如果为每个地形块分配唯一的顶点数据,那么地形顶点的数据量将极其庞大,而在地形上出现两块相同地形的概率也很低,对这些数据的压缩也很难进行。但每个区块作为独立的模型,其顶点信息却可以被共享。

采用 Geo-Clipmap 所使用的方法,因为 2D 网格是规则的,同一层中的顶点坐标 (x,y) 均进行同样的变换,而不同层次之间也可以使用统一的放缩值。那么,使用一组单独的顶点缓冲和索引缓冲即可覆盖整个地形区域,对每个单元块进行放缩和变换。同时,每个网格保留其特有的高程信息以及面法线。

为此建立两个类,一个是 cTerrain,它代表整个地形,来帮助裁剪出实际需要的部分地形,而这个地形被划入网格对应的区块。这些区块使用第二个类表示,cTerrainSection,此类中存储了每个区块的高程及法线。例如,一个 256×256 个顶点的地形,将创建 cTerrain 来存储整个数据集。cTerrain 对象进一步将地形细分为 32×32 的顶点区块,创建总共 64 个 cTerrainSection 对象来表示每个区块。既然所有的 cTerrainSection 对象都是等尺寸、等数量顶点的,就可建立一个单独的索引缓存对象并使用它来渲染地形的各个区块,针对不同层级,只需将相应的步长值向右移一位,如图 2 所示。

在构建索引缓存时,为了使用一组缓存连接所有顶点,还需要生成许多 0 面积的三角形,否则会导致 T 型连接。

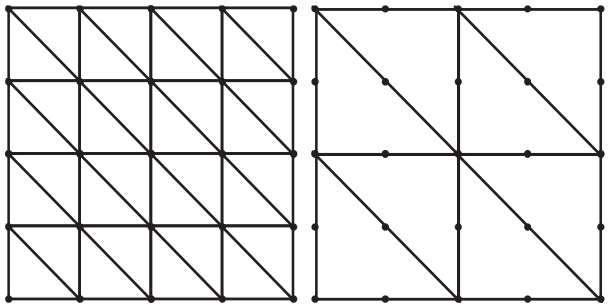


图 2 使用不同的步长值从相同的顶点缓冲获取的不同细节层次

构建索引缓冲的代码如下所示：

```
bool cTerrain::buildIndexBuffer(
int gridVerts//网格边上的顶点个数
int step) //索引缓冲使用的步长值
{int totalStrips=gridVerts-1;//条带的个数
int indice_tripwise=gridVerts<<1;//每个条带占两行顶点
//由于0 三角的存在,顶点个数增加
int total_indice = (indice_stripwise * totalStirps)+(totalStrips<
<1)-2;
int * index = new int[total_indice];
int vert,start_vert=0;
for (int j=0;j<total_strips;++j)
{//生成一行的索引缓冲
vert=start_vert;
for (int k=0;k<xVerts;++k)
{ *(index++) = vert;
 *(index++) = vert + gridVerts * step;
vert+=xStep;
}
start_vert+=lineStep;
if (j+1<total_strips)
{//在每一行的末尾增加0 三角
 *(index++)=(vert-xStep)+lineStep;
 *(index++)=start_vert;
}}}
```

构建顶点缓冲的具体做法，即是将每个块的位置和纹理坐标(x,y)收缩在(0,1)的范围内。在渲染时，将坐标放大至需要的比例，加以合适的偏移值即可移动到对应的世界位置。而各个地形块使用一系列相似的xy位置和uv纹理数据，将它们一次性存储于 cTerrain 对象内渲染每个 cTerrainSection 对象时再引用。这样构建了两组顶点数据流：存储于 cTerrain 的共享顶点数据和各个 cTerrainSection 的独立顶点数据，大大减少了其占用的内存空间。

构建顶点缓冲代码如下：

```
bool cTerrain::create(
cTexture * heightMap,
const cRect3d& worldExtents,
int shift)
{
//设置放缩比例
m_tableWidth=(uint16)heightMap->width();
m_tableHeight=(uint16)heightMap->height();
m_mapScale.x=m_worldSize.x/m_tableWidth;
m_mapScale.y=m_worldSize.y/m_tableHeight;
m_mapScale.z=m_worldSize.z/255.0f;
//设置初始偏移
m_sectorCountX=m_tableWidth>>m_sectorShift;
m_sectorCountY=m_tableHeight>>m_sectorShift;
m_sectorSize.set( m_worldSize.x/m_sectorCountX,
m_worldSize.y/m_sectorCountY);
cVector2 vert(0.0f,0.0f);
sLocalVertex * pVerts=
new sLocalVertex[m_sectorVerts * m_sectorVerts];
//将共享的顶点与纹理坐标写入流中
for (int y=0;y<m_sectorVerts;++y)
{vert.set(0.0f,y * cellSize.y);
for (int x=0;x<m_sectorVerts;++x)
{
pVerts[(y * m_sectorVerts)+x].xyPosition=vert;
pVerts[(y * m_sectorVerts)+x].localUV.set(
(float)x/(float)(m_sectorVerts-1),
(float)y/(float)(m_sectorVerts-1));
vert.x+=cellSize.x;
}}}
```

1.3 裂缝的消除

Chunked LOD 采用了加入外围的技术来消除缝隙，比起强迫相邻区块继续细分或粗化，该方法在需要渲染的网格边界上使用一圈多边形将之包围起来，这些多边形即垂直向下的三角形条带，用来遮挡网格间的裂缝[11]。而这些多出来的外围却会造成网格的不连续性，在平坦的网格中产生垂直的突起，当视点距离较远时，这些突起难以被发觉，但视点靠近或者细节层次的分辨率很低时，裙边就变得明显起来，而加入面法线进行光照混合和纹理贴图时，就会产生明显的畸变，使得本身并不精确的地形网格更加清晰可见[12]（见图 3）。

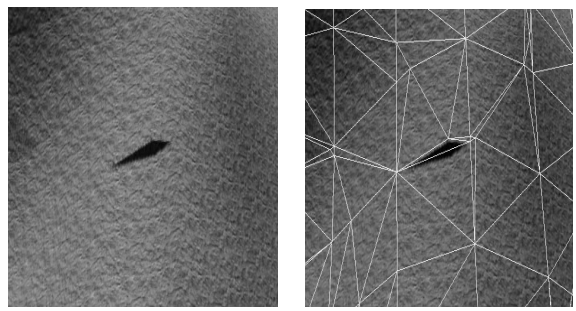
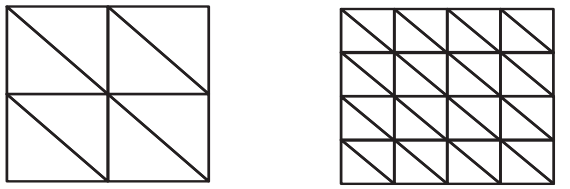


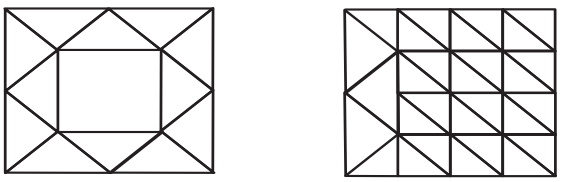
图 3 裙边技术在加入光照后的失真

这里重新定义了索引缓存来构建一个连接缝掩膜，通过与原有的地形网格做与运算来消除缝隙。

如图 4 所示,当两个不同分辨率的地形块进行拼接时,索引缓冲的形状如(c)所示,强制要求高分辨率的网格与之进行与运算后,得到(d)所示的索引缓冲结构,这样即可进行无缝拼接。



(a) 第 l 层顶点索引　　(b) 第 $l+1$ 层顶点索引



(c)拼接缝顶点索引　　(d)交运算后的顶点索引

图 4　不同细节层次拼接缝的消除

1.4　几何过渡

当视点在地形上移动时,不同 LOD 层级之间的交替会引起视觉上的突变,为了防止这种现象的发生,使用几何变形在相邻的 LOD 层级之间进行平滑的过渡。

对于给定的索引和顶点缓存,通过缩放和偏移来获取(x,y) 所对应的世界坐标,并通过顶点纹理拾取为每个单独的地形块读取相应的高度值,接下来,对同一位置(x,y) 在不同层次的高度值进行线性插值即可。

$$H=(1-\alpha)h_l+\alpha\times h_{l-1} \tag{1}$$

这里 h_l 对应较高层次的高度值,h_{l-1} 对应粗糙层次的高度值,由于四叉树的结构特性,相隔两层 LOD 之间的顶点数是四倍关系,对 h_{l-1} 的获取还需对粗糙网格中的相邻顶点进行一次中值运算,$h_{l-1}=(h_1+h_2)/2$。α 是过渡系数,计算公式如下:

$$\alpha=\max(\alpha_x,\alpha_y) \tag{2}$$

$$\alpha_x=\min\left(\max\left(\frac{|x-x_v|-((n-1)/2-w-1)}{w},0\right),1\right) \tag{3}$$

α_y 的计算同理。其中,(x_v,y_v) 是视点所在位置;n 为分块大小;w 表示过渡带的宽度。这里 $n=256$,$w=n/10$。

然而,这种算法的一种不足之处在于四叉树中的一个节点仅支持相邻两层 LOD 间的过渡,这样就限制了视野的收容范围或者是四叉树的深度。因为固定的索引缓存只能够在两种细节层次间进行拼接,增加视野的收容范围可以使拼接区域相互远离而不会出现交叉,但也增加了渲染的负担;增加四叉树的深度以增加数据的颗粒度,四叉树的数据量也会变大;减小过渡带的宽度也是一种解决办法,但在不规则的地形区块上会使过渡变得显而易见。为此,在不同的区域,需要设置不同的过渡带宽度以解决可能出现的缝隙问题。

2　实　验

实验设计使用了一块地形数据大小为 16 385×8 193的高度图,原始网格中共包含 268,435,456 个三角形,共构建了 8 层 LOD。窗口大小的分辨率为 1 280×1 024,使用的语言为 C++与 OpenGL。表 1 比较了不同分块大小对于帧速和内存占用的影响,图 5 为当前视点下的地形框线和模拟光照效果。

表 1　不同分块的性能比较

分块大小	单个节点缓存占用内存空间/kB	视景内的块数	漫游帧速/s
32×32	737	573	85
128×128	4 735	295	75
256×256	8 532	85	56

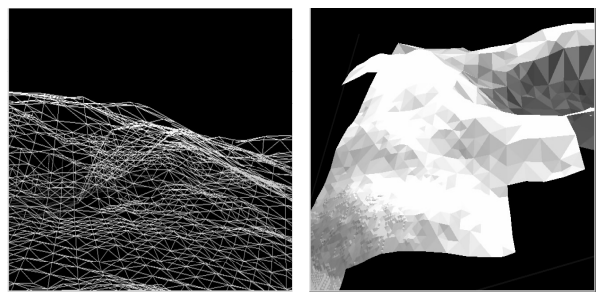


图 5　仿真效果图

3　结束语

由上述实验可得:该算法采用的缓存复用可以较好地节省内存空间,数据结构清晰,支持顶点纹理拾取和大规模数据的分块处理,同时利用掩膜对不同分辨率层级的拼接可以实现缝隙的消除,漫游过程中的几何过渡能减缓视觉上的突跳感,平滑地转变各层间的高程差异。但算法中仍存在一些不足,如视点距离的计算仍是一种二维距离,并未考虑视点高度的差异,因为视点高程的计算会增加顶点搜索的次数而限制实时性;几何过渡的单一性也限制了视点观察的最小距离,否则会使拼接位置显露出来。今后将会扩展改进现有缝隙连接方法,提高该技术的应用能力。

参考文献:

[1] Duchaineau M, Wolinsky M, Sigeti D E. Roaming terrain: Real time optimally adapting meshes[C]//Proceedings of the 8th conference on visualization. Los Alamitos: IEEE Computer Society Press, 1997: 81-88.

[2] Ulrich T. Rendering massive terrains using chunked level of

(下转第 54 页)

4 实验结果分析

用“特征选择方法一”表示改进前的贪婪搜索算法特征选择方法,用“特征选择方法二”表示文中新提出的基于锦标赛排序的特征选择方法。在这一节中将两种不同的特征选择方法作用于原始数据集来生成新的数据集。经过特征选择方法后,把用特征选择方法一生成的新的数据集称为新数据集一,把用特征选择方法二生成的新的数据集称为新数据集二。其中新数据集一包含特征 28 个,新数据集二包含特征 25 个。

图 4 为两种特征选择方法在特征选择过程中 DNCG@1 的变化过程。

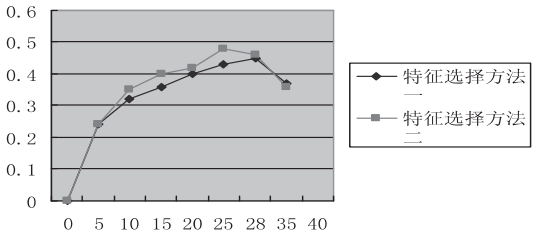


图 4 DNCG@1 结果图

从图中可以明显看出:

(1)无论是特征选择方法一还是特征选择方法二,都在挑选到 25 个左右的时候 DNCG 值达到最高,然后开始逐渐地下降。这充分说明用所有特征参与排序训练得到的模型并不一定是最好的模型。

(2)这两种不同的特征选择方法,选择出来的特征并不完全相同,但是却几乎都可以提升检索性能。所以面向排序学习方法的特征选择过程并不是对单独的特征的选择,而是要选择出一个特征子集,子集中的每个特征对最终的排序函数都有一定的贡献,特征的组合方式也对最终的排序函数有一定的贡献。

(3)相比于特征选择方法一,特征方法二能更高效地从所有特征组成的特征集合中挑选出适合的特征子集。

5 结束语

文中基于针对 QA 问题的排序学习算法中特征选择方法的现状,提出了其在实际应用中的一些弊端,并由此进行了改进。针对具体的 QA 问题,把锦标赛排序的思想应用于排序学习的特征选择中,提出了锦标赛排序的特征选择算法,并从理论和实际两个不同的方面展开研究,取得了一定的成果。

参考文献:

[1] Tetsuya S, Daisuke I, Noriko K. Using graded-relevance metrics for evaluating community QA answer selection[C]//Proc of WSDM'11. [s. l.]:[s. n.],2011.

[2] Suzan V, Hans H, Daphne T. Learning to rank QA data[C]//Proc of SIGIR. [s. l.]:[s. n.],2009.

[3] Tao Qin, Liu Tieyan, Zhang Xudong, et al. Learning to rank relational objects and its application to Web search[C]//Proc of WWW 2008. [s. l.]:[s. n.],2008.

[4] 郑实福,刘 挺,秦 兵,等. 自动问答综述[J]. 中文信息学报,2002,16(6):46-52.

[5] 陈 彬,洪家荣,王亚东. 最优特征子集选择问题[J]. 计算机学报,1997,20(2):17-22.

[6] Feng P, David A, Franco S. Feature selection for ranking using boosted trees[C]//Proc of CIKM'09. [s. l.]:[s. n.],2009.

[7] 代六玲,黄河燕,陈肇雄. 中文文本分类中特征抽取方法的比较研究[J]. 中文信息学报,2004,18(1):26-32.

[8] 刘丽珍,宋瀚涛. 文本分类中的特征选取[J]. 计算机工程,2004,30(4):14-15.

[9] 章 兰. 一种基于 VSM 模型的动态文本分类器的设计[D]. 苏州:苏州大学,2004.

[10] 万 忠,张燕平,张 铃,等. 基于覆盖算法决策界的特征选择算法[J]. 计算机技术与发展,2006,16(4):84-87.

[11] Hua Guichun, Zhang Min, Liu Yigun, et al. Hierarchical feature selection for ranking[C]//Proc of WWW 2010. [s. l.]:[s. n.],2010.

(上接第 49 页)

detail control[C]//Proc of annual conference series on computer graphics. San Antonio, Texas: ACM Press, 2002.

[3] Losasso F, Hoppe H. Geometry clipmaps: Terrain rendering using nested regular grids[C]//Proc of annual conference series on computer graphics. New York: ACM Press, 2004: 769-776.

[4] 宫晓辉,温慧明,于 卓. 基于 CLipmap 的海量地形及纹理实时绘制方法[J]. 计算机技术与发展,2012,22(10):22-26.

[5] 邹 海,徐 军,褚维翠. 基于 OpenGL 的三维地形的模拟[J]. 计算机技术与发展,2011,21(6):239-241.

[6] 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):5-8.

[7] 申闫春,朱幼虹,温转萍,等. Chunklod 地形的过程化细节度增强算法[J]. 系统仿真学报,2008,20(21):5763-5766.

[8] 张立民,闫文君. 基于 GPU 的大规模地形数据绘制算法[J]. 计算机与现代化,2012(1):145-150.

[9] 刘慧媛. 三维海底地形绘制方法研究与实现[D]. 哈尔滨:哈尔滨工程大学,2009.

[10] 马淑芳. 基于 GPU 加速的分形地形生成方法[D]. 大连:大连理工大学,2008.

[11] 廖学军,王荣峰. 数字战场可视化与应用[M]. 北京:国防工业出版社,2010:39-42.

[12] Strugar F. Continuous distance-dependent level of detail for rendering heightmaps[J]. Journal of graphics GPU and game tools, 2009, 14(4):57-74.

基于四叉树的缓存复用机制地形算法

作者： 宋省身， 全吉成， 赵秀影， 王宏伟， 王宇， SONG Xing-shen， QUAN Ji-cheng， ZHAO Xiu-ying， WANG Hong-wei， WANG Yu

作者单位： 宋省身,全吉成,赵秀影,王宏伟,SONG Xing-shen,QUAN Ji-cheng,ZHAO Xiu-ying,WANG Hong-wei(空军航空大学 航空航天情报系,吉林 长春,130022)， 王宇,WANG Yu(军事仿真技术研究所,吉林 长春,130022)

刊名： 计算机技术与发展

ISTIC

英文刊名： Computer Technology and Development

年，卷(期)： 2014(2)