一种双层次细节地表模型的实时绘制算法

管 莉,张胜超,朱 斌,郝重阳
GUANG Li,ZHANG Sheng-chao,ZHU Bin,HAO Chong-yang

西北工业大学 电子与信息工程研究所,西安 710072
Institute of Electronic and Information Engineering,Northwestern Polytechnical University,Xi'an 710072,China

GUANG Li,ZHANG Sheng-chao,ZHU Bin,et al.Real-time rendering arithmetic of two levels of detailed terrain.Computer Engineering and Applications,2008,44(25):173-175.

Abstract: A two levels of detail terrain is presented in this paper.During the lower Level Of Detailed (LOD) terrain,we simply render the whole terrain using ROAM mesh independent of view.During the high LOD terrain,we render the fluctuant detail of terrain's height using symmetrical mesh along view center.We control upper LOD terrain's range by adjusting projective area proportion between upper LOD terrain and the whole terrain.Transitional area of the two level terrain is pushed away from the view.Based on the resolution difference between lower level and upper level,we calculate the transitional area's width,and ensure the two LOD terrain's smooth transition by rendering the areal color based on image special Per-Pixel displacement texture method.

Key words: two levels of detail;transitional area;terrain mesh

摘 要:提出了一种双层次细节地表模型的快速绘制算法。建立了双层次细节地表模型,在模型的低层次细节中,采用了视无关的ROAM网格粗略地表示整个地形地貌;在高层次细节中,借助以视点为中心的均匀网格描述地形高度起伏的细节。通过调节高层细节与整个地形投影面积的比例,控制高层次细节覆盖范围,把两层模型的过渡区域推向离视点较远的位置。过渡区域采用了基于图像空间的逐像素纹理混合与偏移方法生成该区域的颜色,保证双层次细节的平滑过渡。实验证明,该方法在保持逼真视觉效果的同时,能够较大规模地提高三维地形绘制速度。

关键词:双层次细节;过渡带;地形网格
DOI:10.3778/j.issn.1002-8331.2008.25.052 **文章编号:**1002-8331(2008)25-0173-03 **文献标识码:**A **中图分类号:**TP391

1 引言

提出了一种三维地表模形的实时绘制算法,首先建立了双层次细节地表模型,在模型的低层次细节中,采用了视无关的ROAM[1]网格粗略地表示整个地形地貌。在高层次细节中,借助以视点为中心的均匀网格描述地形高度起伏的细节。整个算法在GPU进行了设计与实现,使用了Vertex Texture Fetch[2]和基于硬件加速的纹理偏移[3](GPU Based Displacement Texture)计算的技术,在保证逼真绘制效果的同时,进一步加快地表模型的绘制速度。

2 算法设计

2.1 建立双层次细节地表模型

设地表模型G由低细节层Level0与高细节层Level1组成,即G=<Level0,Level1>。Level0使用视无关的,仅以地形区域高度变化程度所引起的累积(屏幕)误差为剖分依据的ROAM网格进行整个地形的粗略表示。Level1则采用以视点为中心的均匀网格进行表示,其覆盖范围随视线方向的变化而变化。

设整个地形地平面区域长为l,宽为w,视点为V,E是中心视线与地平面的交点,那么在整个地形区域内,G所表示的地形高度Z为:

$$Z=\begin{cases} funZ(level1,E_x+x,E_y+y) & -l/2\leqslant x\leqslant l/2,-w/2\leqslant y\leqslant w/2 \\ funZ(level0,E_x+x,E_y+y) & \text{otherwise} \end{cases}$$

其中,$funZ$为根据平面位置高度获取函数,l、w为变量,用来控制Level1覆盖范围,它们随视线方向的变化而变化。双层次细节模型示意图如图1所示。

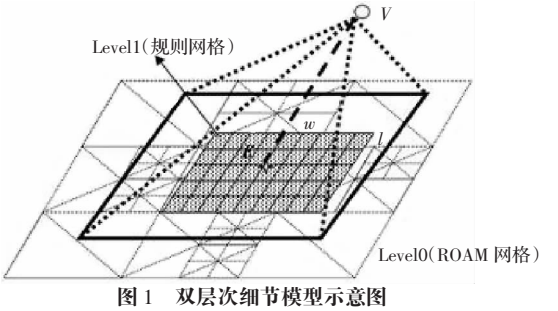


图1 双层次细节模型示意图

基金项目:国家高等学校博士学科点专项科研基金项目(No.20040699015)。
作者简介:管莉(1967-),女,博士研究生,主要研究方向为虚拟现实、算机仿真、人工智能等。
收稿日期:2007-10-17 **修回日期:**2008-03-24

Level1 覆盖范围是可变的,由它经过投影后在屏幕上的面积与 Level0 层在屏幕上的面积的比例(或像素数之比)决定的。这样可以按照实际需要选择 Level1 网格的大小,尽可能地减少不必要的计算开销,另一方面可以保证两层次细节之间的过渡区域远离视点,减少视觉上的跳变。下面给出 l、w 的确定方法:

首先将 Level1 网格分为多个离散的层次,分别用 $M0$、$M1$、$M2\cdots MI\cdots M\max$ 表示,MI 的大小为 2^i*2^i,MAX 为常数,根据机器可实时处理的最大网格数量来设定的。在每一帧中具体选用哪一层网格进行绘制是根据上一帧投影面积的计算进行的。用 funProjective( )表示投影面积计算函数,R 表示面积百分比,具体算法如下:

第一步:设定投影面积百分比控制范围[$a1\%$,$a2\%$],在本文实验中使用了[50%,80%],保证了近地视点应用的需求;

第二步:初始化,在绘制的第一帧(t=1)时,选择 $M_{\max}$ 进行绘制,即 $l=w=2^{\max}$;

第三步:计算第 t 帧两层次细节投影面积百分比 R_t:

$$R_t=fun\ \mathrm{Pr}\ ojective(V_t,M_i^t)/fun\ \mathrm{Pr}\ ojective(V_t,Level0)$$

第四步:将 R_t 与阈值[$a1\%$,$a2\%$]进行比较,选择第 t+1 帧网格 $M_j^{t=1}$:

$R_t<a_1\%$时,$j=\min(i+1,MAX)$;

$a_1\%\leqslant R_t\leqslant a_2\%$时,$j=i$;

$R_t>a_2\%$时,$j=\max(i-1,0)$。

2.2　双层模型的绘制

对于双层次细节模型,由于 Level0 层是视无关的网格,而 Level1 层随视点的变化在 Level0 层上浮动。那么,在绘制过程中,如果不做任何处理,直接绘制第 Level0 层、Level1 层网格将出现如图 2 的双层模型相互遮盖现象,导致 Level1 层网格所表现的地形起伏细节丢失。

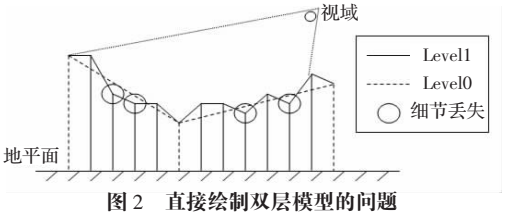


图 2　直接绘制双层模型的问题

为了解决这个问题,最直观的方法是进行几何空间的镶嵌。但由于两个层次网格分辨率、拓扑结构均不相同,直接进行几何空间的镶嵌算法较为复杂,使得每一帧的计算开销增加过多。所以采用了基于图像掩码的方式进行双层次网格图像空间的融合,即先绘制 Level1 层网格,并记录它在帧缓存中所占有的位置,形成一张二值掩码图像,然后绘制 Level0 层网格,根据掩码图像的指导,在 Level1 层占位的区域不着色,具体流程如图 3 所示。

2.3　过渡带的计算

当地形具有一定规模时,Level0 层表示地形全貌,而 Level1 层根据视点位置表示近视点的局部地形,两个层次之间的分辨率相差较远,因此需要在 Level1 与 Level0 之间建立过渡带。由于在 Level0 层使用了视无关的 ROAM 网格,网格的局部分辨率与局部地形起伏程度紧密相关,因此,Level0 层的分

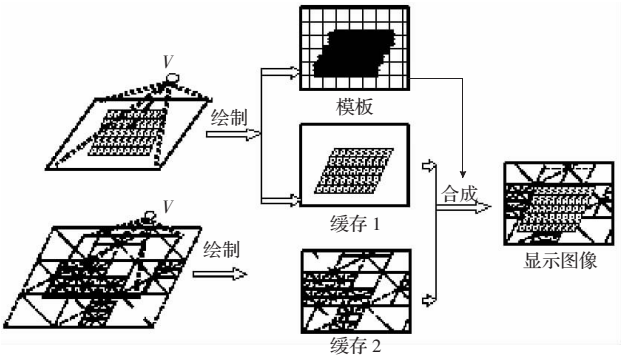


图 3　双层模型绘制流程

辨率是非均匀的,通过定义规则地形数据的网格的平均分辨来计算过渡带宽度。

设网格的平均分辨率 r 为网格上的顶点数目 $numVertex$ 与网格所覆盖的地形高程采样点数目 $numSample$ 之比,即 $r=numVertex/numSample$。那么,过渡带的宽度 w_{trans} 计算公式为:

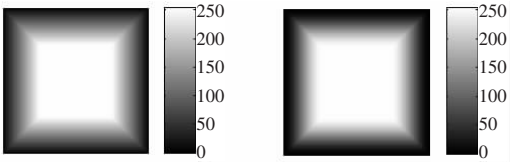
$$w_{trans}=\frac{r_{level0}}{r_{level1}}\times\alpha\times w_{level1}$$

其中 α 为分辨率比例调节因子。

确定过渡区域的宽度 w_{trans} 后,过渡区域着色过程分为以下几个主要步骤:

第一步:建立过渡区域。将 Level1 网格向四周扩大 w_{trans} 宽度,此时 Level1 与 Level0 重叠覆盖的区域即为过渡区域,记录该区域在当前视点下投影到帧缓存中的位置,进行标记。

第二步:生成过渡区域插值权重模版。为了在过渡区域产生平滑过渡的视觉效果,根据 Level1 的尺寸及过渡区域宽度 w_{trans},生成渐变的插值权重模版,以纹理贴图的方式映射到 Level1 及由扩大 Level1 网格形成的过渡区域网格上。模版尺寸可采用固定大小,例如 256×256。渐变函数可选取线性函数或 GAUSSIAN 函数,渐变结果范围标准到[0,1]。图 4 为本文采用的过渡区域插值权重模版示意图(图示范围为[0,255],方便显示)。



(a)线性渐变模版　(b)GAUSSIAN 渐变模版

图 4　过渡区域插值权重模版示意图

第三步:逐像素计算纹理偏移与着色。根据第一步过渡区域的生成方式,可知在 Level0,Level1 粗细两层网格中均对该区域进行了绘制,因此,对于该区域内每个标记像素在 Level0,Level1 层上均有对应点,可以获取相应的纹理坐标,以进行纹理坐标的插值计算,产生纹理偏移。过渡区域在帧缓存中 i,j 坐标像素的新纹理坐标 $texcoorduv_{trans}^{ij}$ 的具体计算公式为:

$$texcoorduv_{trans}^{ij}=texcoorduv_{level1}^{i,j}\times\beta_{i,j}+texcoorduv_{level0}^{i,j}\times(1-\beta_{i,j})$$

其中 $texcoorduv_{level0}^{i,j}$、$texcoorduv_{level1}^{i,j}$ 分别为对应像素位置在 Level0,Level1,粗细两层网格中的纹理坐标值,$\beta_{i,j}$ 则为取自过渡区域插值权重模版的对应位置的权值。最后,按照新纹理坐标对过渡区域进行着色。

2.4　算法实现流程

2.4.1　初始化过程

(1)初始化一系列不同分辨率的规整网格,尺寸分别为32×32,64×64,128×128,256×256,512×512,创建一个顶点缓冲区数组以及相应的索引缓冲区数组,用来存放这些规整网格,并且为每一个分辨率的网格,创建一个过渡带。

(2)对地形高程数据进行平滑滤波处理,本文采用的地形高程数据为 4 096×4 096 点,并且把该高程数据存入到一张 4 096×4 096 的顶点纹理(VertexTex)当中,另外还将一张影像创建为 4 096×4 096 的影像纹理(GeoTex)。

2.4.2　Frame 循环过程

CPU 阶段:

(1)根据地形高程数据、视点位置 *vEye* 以及观察方向 *vView*,求出视线与地面的交点,把这个交点作为视相关网格的中心位置 *Pos*,为了防止产生抖动,把交点位置取整后得到的中心位置 *CenterPos* 作为视相关网格的中心位置。将中心位置 *CenterPos* 作为常量传入到顶点着色器[4]当中。

(2)设置顶点纹理(VertexTex)和影像纹理(GeoTex),观察矩阵 ***mView*** 以及投影矩阵 ***mProj***。

GPU 阶段:

(1)第一个渲染 pass:渲染第二层的视相关网格,渲染结果输出到纹理 Tex1(纹理格式为 A8R8G8B8)中(渲染到纹理[5] render to texture)。

①根据上一 frame 第二层网格与第一层网格投影面积之比选定一个分辨率的网格。

②把纹理的 RGBA 值清空为 0。

③在顶点着色器当中进行偏移,把规整网格偏移到视点附近,然后根据每个顶点的水平坐标位置,计算出顶点的纹理坐标,然后使用这个纹理坐标对存放着地形高程数据的顶点纹理进行采样,采得的纹理值也就是这个顶点的地形高程值,用这个值去偏移顶点的 y 值。

④像素处理阶段,根据纹理坐标对影像纹理进行采样,采得 RGB 颜色值作为着色值,并且 *alpha* 值设为 1.0。这样得到了一张 RGBA 格式的纹理,RGB 保存的是渲染的颜色值,*alpha* 作为一个混合的 MASK。

(2)第二个渲染 pass:渲染过渡带,渲染结果输出到纹理 Tex2(纹理格式为 A16fR16fG16fB16f)

①在顶点着色器当中进行偏移,整网格偏移到视点附近,然后根据每个顶点的水平坐标位置,计算出顶点的纹理坐标,然后使用这个纹理坐标对存放着地形高程数据的顶点纹理进行采样,采得的纹理值也就是这个顶点的地形高程值,用这个值去位移顶点的 y 值。

②把纹理坐标值(UV)作为 RG 值输出,B 值存储为一个插值系数,*alpha* 作为混合的 mask。

(3)第三个渲染 Pass:渲染第一层网格,并且同时混合第二层网格的结果,最终的渲染结果输出到纹理 tex3(A8R8G8B8)。

①在顶点处理阶段,对 ROAM 网格的顶点进行观察和投影变换。

②在像素处理阶段进行混合(利用 Shader model3.0 的动态流控特性)。以屏幕坐标作为纹理坐标,取纹理 Tex1,如果 alpha Mask 的值为 1.0,说明这个像素是第二层网格上的,应该用 Tex1.rgb 着色。如果 alpha 为 0,则对 Tex2 进行采样,判断 Tex2 的 alpha Mask,如果 mask 为 1 则说明是像素在过渡带上,应使用偏移后的纹理坐标进行着色。如果为 0,则说明像素在第一层网格上,应该用原有的纹理坐标进行着色。

3　实验结果分析

实验使用的地形高程数据尺寸为 4 096×4 096,数据点采样间隔为 32 m,高程值采用的 16 Byte 的浮点数存放浮点纹理当中,与传统的高程数据相比,这种 16 Byte 的浮点纹理高程数据存放形式既保证了精度,又节约了存储空间。

实验的硬件环境采用了 3.0 GHz Pentium4 CPU 的 PC 机,1 GB 内存,NV 8800GTS 显卡,256 兆显存。实验程序在 Visual Studio 2005 环境下使用 C++和 DirectX 9 完成,使用了 Shader model 3.0[7]。图 5 显示了使用本文方法对该数据进行三维绘制的效果,窗口大小为 1 280 像素×1 024 像素,视域夹角为 15°,近地面低空视点,低细节层 Level0 的 ROAM 静态网格三角形数量控制在 2 048 个。图 5 则给出了对应的地形场景图与线框图,可以明显看出本文采用的地表模型的双层次细节网格及过渡区域(用红色绘制出)。

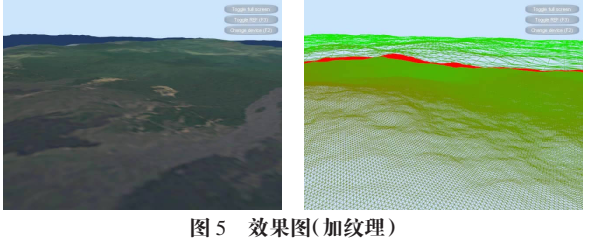


图 5　效果图(加纹理)

在实验中,对细层次网格 Level1 层覆盖范围进行控制的阈值[$a1\%$,$a2\%$]设置为[50%,80%],具体实验数据表 1 给出。从实验数据可以看出,本文方法可以根据视角的变化,自适应地调节双层次细节之间的比例关系,使其达到一个较为理想的稳定范围,在保证良好视觉效果的同时,尽可能减少不必要的计算开销,提高绘制效率。实验证明,本文方法可以满足近地低空视点多钟视角三维实时应用的需求。

本文使用 4 096×4 096 地形数据,开启视域裁减,实验环境如上,测得本文算法的平均绘制速度为 190 帧/s。在同一环境和同等规模的测试数据的条件下,将本文方法的绘制速度和其他方法的速度进行了对比实验,表 2 给出具体实验数据。本文方法建立两层细节的地表模型,并对它们进行图像空间的无缝融合,大大降低了实时处理的计算量,因此,在绘制速度方面较其他方法具有一定的优势。

表 1　Level1 层网格覆盖范围切换实验数据表

视点高度/m	视角/°	Level1 当前尺寸(点×点)	两层次投影面积百分比/%	绘制三角形数量/个	帧率/(帧/s)	Level1 预测尺寸(点×点)
100	15	1025×1025	73	2 099 220	64	1024×1024
100	30	513×513	68	526 336	82	513×513
100	45	513×513	87	526 336	130	257×257
100	60	257×257	62	133 120	181	257×257
100	75	257×257	77	133 120	193	257×257
100	90	257×257	82	133 120	185	129×129

表 2　地形数据绘制速度实验数据对比表

方法名称	ROAM 法	GeoMipMap 法	Chuncked LOD 法	本文算法
速度/(帧/s)	120	131	147	190

(下转 198 页)