技术方法

基于 GPU 的全球地形数据加速绘制技术研究

李浩，相恒茂，孙久虎

（山东省国土测绘院，山东 济南　250102）

摘要：在对基于 GPU 大规模地形数据快速绘制方法的研究基础上，针对 GPU 的批量绘制的特点，提出了改进的几何体实例化方法，把它用在全球格网的快速绘制上。把地形数据通过重新编码存储在高度图中，再结合 Shader Model 3.0 引入的顶点纹理拾取技术，在 GPU 中进行解码，得到原始的 DEM 数据，避免了在 CPU 上执行顶点缓存的更新。实验证明该算法充分利用了新一代 GPU 的特性，降低了 CPU 的负担，能达到实时绘制的要求。

关键词：图形处理器；实时绘制；四叉树；瓦片块；高度图

中图分类号：TP391.41　　**文献标识码**：B

引文格式：李浩，相恒茂，孙久虎.基于 GPU 的全球地形数据加速绘制技术研究[J].山东国土资源，2015，31(7)：59－64.LI Hao，XIANG Hengmao，SUN Jiuhu.Research on Accelerated Rendering Technique of Global Terrain Based on GPU[J].Shandong Land and Resources，2015，31(7)：59－64.

0　引言

海量地形数据实时绘制技术在 GIS、虚拟现实、以及三维游戏中有非常重要的应用，一直是国内外研究的热点。在 GPU 加速绘制技术出现之前，大规模地形绘制算法的研究主要集中在地形数据的简化和裁剪技术。但是随着地形数据的增加和现实需求对清晰度和真实感的要求，基于 CPU 的数据简化和裁剪技术已不能满足大规模地形实时绘制的需求。人们对于绘制地形的范围、速度和效果的要求也随之不断提高。加上“数字地球”和“全球信息网格”[1]概念的相继提出，建立具有多分辨率、海量数据的大规模虚拟地形场景受到越来越多的关注，其中大规模地形、海量数据的实时绘制算法对整个系统有着至关重要的影响。

传统的地形绘制系统大都是基于 CPU 的地形绘制方法，地形数据也是通过 CPU 来采样，并没有充分发挥现代图形处理器 GPU 的性能。因此研究如何充分利用当前图形硬件的特点，使绘制算法更淋漓尽致地发挥 GPU 的性能，即研究面向 GPU 的绘制算法，对于提高大规模虚拟自然场景的实时绘制速率是非常有意义的。

2008 年的 SIGGRAPH 会议[2]上，著名的显卡生产商 NVIDIA 公司根据 GPU 的最新特性，实现了基于 GPU 的 Chunked LOD 的地形绘制算法，减少了 CPU 的负荷，通过图形处理器 GPU 使得地形绘制算法得到了很大的改进。

2010 年 Filip Strugar[3]提出了 CDLOD（Continuous Distance－Dependent Level of Detail for Rendering Heightmaps）绘制地形的方法，实现了一种基于 GPU 采样高度图来渲染地形的技术，这种方法改进了以往依据观察者到目标点距离为基础的 LOD 方法。

全球地形可视化框架的建立，除了海量数据的存储、调度、绘制以外，还涉及到许多其他方面的问题。该文的研究重点针对以下问题：

(1)由于地形可视化算法的要求，将原始地形数据处理到符合地形绘制要求的数据，往往要经过一个数据预处理过程，主要是把大规模的地形数据处理成高度图，利用 Shader Model 3.0[4]引入的顶点纹理拾取技术，对高度图进行采样，这种利用 GPU 采样的技术减轻了 CPU 的压力。

(2)在数据绘制中由于图形硬件的特殊性，“海量”数据渲染的数据级别比管理和调度中的数据级

收稿日期：2014－10－11；**修订日期**：2014－10－27；**编辑**：陶卫卫

作者简介：李浩(1984—)，男，工程师，主要从事地理信息系统应用和国土资源信息化建设工作；E－mail：919150382@qq.com

别要小得多，为了高效地管理、调度、绘制数据，必须设计一种科学合理的数据组织方案，能够快速地根据需要索引和调度数据，节约数据装配的时间，提高数据索引的速度，构建适合于 GPU 框架下的地形 LOD 数据模型。在此基础之上必须设计适应性强、效率高的数据引擎，该数据引擎能根据需要从文件系统或者数据库系统中快速索引和装配数据，满足实时的可视化需求[5-6]。

1　基于高度图的地形组织方法

高度图是一个数组，数组中的每个成员指定地形顶点描述中的高度信息。可以把高度图想像成一个矩阵，因为每个元素都一一对应于每个地形网格中的顶点。

当保存高度图到磁盘上时，通常为高度图的每个元素分配 1 个 byte 的内存，所以高度的范围是 0～255，0～255 的范围对于地形的高度之间保持平滑过渡是足够用的。但为了在程序中匹配 3D 世界中的物体，可能需要的范围在 0～255 以外。例如，当研究整个地球时，最大的海拔高度 8 848.13 m，那么 0～255 的范围对于表示复杂的地形表面是完全不够用的。因此，当读取数据进应用程序时，给每个高度元素分配一个整型数（或浮点型），它允许缩放 0～255 范围之外的任何大小的物品。这样就可以有效地扩大高度图的应用范围。

高度图图形表示法之一是灰度图（grayscale map）。较黑的值表示地形中较低的地方，较白的值表现地形中较高的地方。

在 Shader Model 3.0 以前，虽然有很多学者把地形数据处理成高度图，但还都是利用 CPU 进行采样获取地形数据，效率并没有明显的改善。但是随着 GPU 的不断更新，Shader Model 3.0 的出现，完全改变了现状，Shader Model 3.0 引入的顶点纹理拾取技术，可以直接使用 GPU 的光栅化管道来进行处理高度图，将高程值存储在顶点纹理中，在顶点着色器里查找并获取顶点的高程值，避免了在 CPU 上执行顶点缓存的更新，这样可以节省大量的 CPU 资源。

其实高度图是一个很灵活的数据格式，为了应用的方便和需要，可以定义自己的高度图。论文中就是自定义的高度图，把高度图保存成 PNG。因为 PNG 有 4 个通道，而 GPU 中的 vector 也是四维的，刚好对应起来。

由于地形高度一般都是 float，所以假设地形高度为 h（单位为 m），把 h 分成两部分，整数部分为 m，小数点后的四位记为 n（其中 n 为四位数，）例如 $h=341.349\ 82$，则 $m=341$，$n=3498$。这样做是可行的，精度可以达到 0.1 mm，完全能够满足实际应用的需要。为了方便 GPU 进行采样对地形高程数据进行编码，计算 r，g，b，a（其中 r，g，b，a 为 PNG 的 4 个通道）：$r=\mathrm{m}/255$；$b=\mathrm{m}\%\ 255$；$g=n/255$；$a=n\%255$。

按照此编码，用 GPU 进行采样，通过逆运算，就可以得到原始的高程值。此过程是在 GPU 中并行计算的，可以充分发挥 GPU 的并行高效计算特性，可以加速采样的速率，最终会提高整个系统的效率。

基于上述高度图的特点和实际应用的需要，同时也为了适应全球框架下的系统结构，利用高度图把全球的 DEM 数据进行了重新的组织，高度图也采用了类似于影像数据的分层分块的思想，把全球的 DEM 数据划分成瓦片块，同时采用了四叉树的调度方式，能够快速的索引和调度到所需要的地形块数据，很好地适应当前大规模数据量的要求，能够达到一个很好的效果。

2　地形绘制算法和客户端绘制流程

该文采用 Geometry Instancing[7] 的思想，充分发挥 GPU 批量绘制的特性，来解决大规模地形实时高效绘制的问题，针对该文的基于 GPU 的地形绘制算法详细情况如下：①创建一个顶点缓存和一个索引缓存，用来存储实例模板；②根据视点的位置和视锥体的范围，确定视野范围内的所有地形块，并把它们起始点的经纬度、每个网格的跨度、起始点的纹理坐标的偏移以及层数等信息，作为实例的值传给 GPU；③在 GPU 中，通过每个实例的值，对原始的实例模板进行矩阵变换（平移、缩放、旋转等），得到每一个瓦片块的真实的值；④在 GPU 中，根据纹理坐标利用 HLSL 里面的采样器对高度图进行采样，并对采样的结果进行逆运算，就是地形的高度值；⑤上述的信息都准备好后，下一步就是渲染，把所有的实例送进渲染管道，进行批次渲染；⑥在漫游过程中，通过②的判断，传给 GPU 新的实例的值，

循环利用上述步骤进行计算和渲染[8-10]。该系统是采用 C/S 架构的网络三维地理信息系统平台，因此要是把采集到的地形数据处理成高度图，存放在服务端，经过服务器，传输到客户端，并在客户端进行数据操作，最后实现在显卡上绘制的过程(图 1)。

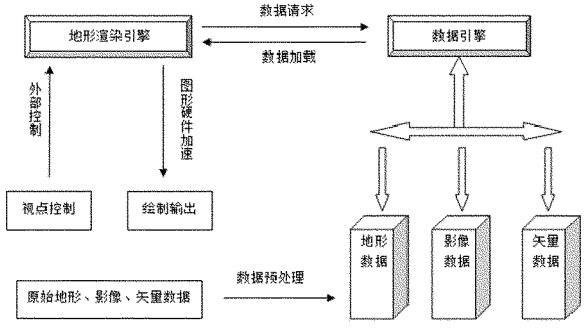


图 1　绘制流程

3　不同 LOD 之间的裂缝处理技术

大规模地形绘制算法中，相邻的不同 LOD 瓦片块之间会出现裂缝问题。对于不同的地形绘制算法，由于数据组织和调度的不同，裂缝的处理也不尽相同。根据实际需要利用 morphing 技术来消除相邻的不同 LOD 瓦片块之间产生的裂缝问题，得到了很好的效果。这样做改善了传统的绘制“裙边”的方法，因为“裙边”在某些情况下并不适用，比如要绘制地下管道，绘制“裙边”是一个致命的缺点。

Morphing 渐变[11]是 20 世纪 90 年代出现的一种革命性的计算机图形技术，该技术使得动画序列平滑且易于处理，即使在低档配置的计算机系统上也能正常运行。渐变是指随着时间或距离的变化把一个形状改变为另一个形状，可以称是 Mesh 网格模型。渐变网格模型的处理就是以时间轴或距离轴为基准，逐渐地改变网格模型顶点的坐标，从一个格网模型的形状渐变为另外一个(图 2)。

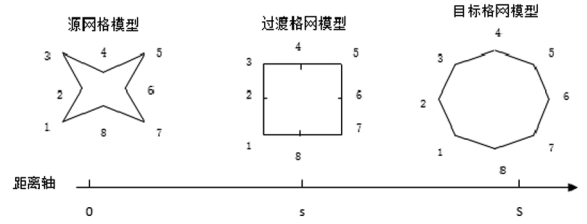


图 2　morphing 的过程

用这种方法，每一个顶点依据它自己的 LOD 的级别进行 morphing 运算，morphing 发生在每一块或每一个节点之间。每个节点都支持不同 LOD 层次(它自身和高细节层次或低细节层次)之间的转换。Morphing 操作是通过顶点着色器把高细节层次的每 8 个三角形平滑过渡到低细节层次的 2 个三角形，其余的 6 个三角形则转换为退化三角形而未被栅格化出来。这个过程可以在不同的 LOD 之间平滑的过渡，不会产生裂缝和 T 链接(图 3)。

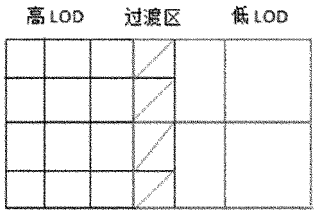


图 3　morphing 的实质

首先，通过计算观察者和顶点之间的距离来确定所需的 morphing 的渐变量。利用顶点位置来计算距离可以非常的接近，只要这个逼近函数在整个数据集域里是一致的。为了防止裂缝的产生，有必要使这个格网边界的节点和邻近格网的节点有一定的重复。在该系统中，x 代表纬度和 y 代表经度，通过它们可以变化到世界空间坐标系下，z 代表高程值，是通过采样高度图得到的，并用了统一的过滤器。

在该系统中渐变量用 morphK 表示，范围为 0～1，其中 0 表示没有用 morphing，具有很高的细节层次格网，1 表示完全的 morphing，具有只有原来 1/4 的三角形个数，morphK 被用来逐渐地从一个渐变格网顶点过渡到对应的没有渐变的格网。假设格网索引为(i,j)，如果 i 和 j 中至少有一个是奇数，则这个点就被标示为过渡顶点，同时也可以得到它们附近的坐标为$(i-i/2, j-j/2)$的点属于非过渡顶点。

下面是用 HLSL 语言写的过渡顶点的代码：

```
const float2 g_gridDim = float2(32,32);
float2 morphVertex(float2 gridPos, float2 vertex, float morphK){
    float2 fracPart = frac( gridPos.xy * g_gridDim.xy * 0.5 ) * 2.0 / g_gridDim.xy;
    return vertex.xy - fracPart * g_quadScale.xy * morphK;
}
```

最后，z 是根据纹理坐标通过 GPU 采样高度图获得的，纹理坐标是利用二次线性插值计算出来的，

当一个节点的所有顶点都过渡到低细节层次的顶点时，网格则有效地包含原来 1/4 的和低细节层次相匹配的三角形，因此可以无缝的取代它。

4　基于 GPU 的硬件裁剪和背面剔除技术

早期的游戏直接用包含三角形列表表示场景，尽管大部分三角形对当前画面没有贡献，但每帧绘制时必须全部处理它们。考虑到游戏角色只能位于某个房间中，如果房门是关闭的，游戏角色可见的三角形全部位于当前的房间中，只占到全部三角形数目的 1%。一个自然的想法是剔除那些明显不可见的 99%的三角形。因此，除了场景组织外，还需要考虑裁剪的问题，主要包括视域裁剪(View - Frustum Culling)技术、背面裁剪(Back - Face Culling)技术和遮挡裁剪(Occlusion Culling)技术[12](图 4)。

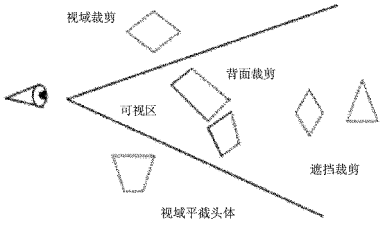


图 4　裁剪技术

5　系统性能分析

该系统是在局域网中实现的 C/S 模式，实验是基于全球的地形数据进行测试。

服务器的配置：戴尔 Precision T1500；RAM 6.00GB；CPU Intel(R) Core(TM) i3 540 3.07GHz 四核处理器；Apache HTTP Server 2.2.17。

客户端的配置：CPU Intel(R) Core(TM) i5 2.67GHz；RAM 2.98G；NVIDIA NVS 3100M 显卡。

开发环境：VS2005，DirectX 9.0，HLSL。

实验数据：全球 DEM 数据采用 90 m 分辨率的 SRTM 数据和全球影像数据。

以下是该系统的一些效果图对比情况。

该系统整合了全球的影像数据，采用了分层分块的金字塔的组织方式进行存储和管理，无论漫游到任何一个地方，该系统都会快速的索引和调度到所需要的数据。同时利用高度图把全球的 DEM 数据进行了重新的组织，高度图也采用了类似于影像数据的分层分块的思想。不论是影像还是地形，采用了四叉树的调度方式，能够很好的适应当前大规模数据量的要求，能够达到一个很好的效果。

图 5 所示，地形格网以线框模式进行绘制，可以很清晰地看到地形的格网构成，这就是利用 GPU 采样高度图还原原始地形后的结果，其中颜色较亮的部分为地形高度较高的地方，颜色较暗的部分为地形高度较低的地方。这也证明了利用高度图来组织地形数据、通过 GPU 进行采样的可行性和有效性。

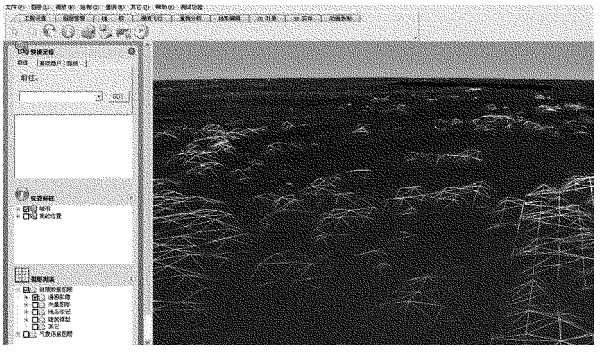


图 5　地形格网模式

图 6 所示，地形格网以填充模式进行绘制，贴上影像纹理，可以显示地表外貌。当在系统的漫游的过程中，你会感受到地形的高低起伏的变化和清晰可见的影像数据，同时还增加了光照计算，让周围的环境更像现实世界，增强了系统的可视化的效果。

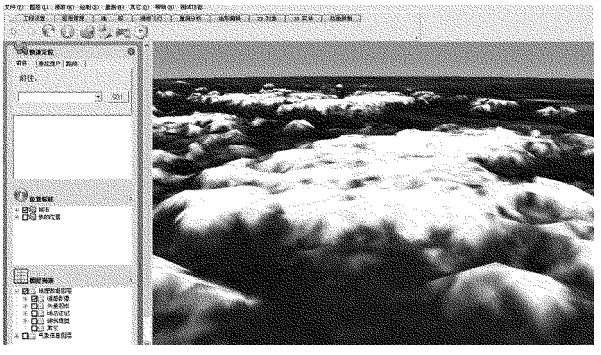


图 6　地形填充模式

图 7 所示，为裂缝处理前的效果，箭头所指的地方可以清晰的看到不同 LOD 之间产生的裂缝。由于采用的是四叉树和分层分块的组织方法，不可避免地出现这种裂缝的情况，因为不可能全球只创建一个巨大网格，把所有的数据都存放在这个网格上。因此产生了如图 8 所示的效果。

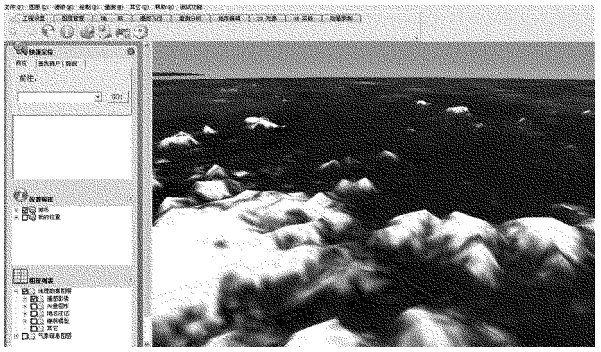


图 7　裂缝消除前

对于裂缝的处理，很多学者都有自己的方法，其中最常用的就是通过在不同的 LOD 之间绘制“裙边”来消除裂缝，但是绘制“裙边”并不是一个万能的方案，当需要绘制地形的物体时，比如目前研究很多的地下管道的绘制问题，绘制“裙边”是一个致命的缺点。该文把 morphing 技术的思想用来解决裂缝问题，虽然稍微增加了一点计算量，但是达到了很好的效果，如图 8 所示，裂缝明显的消除了，而且如果要绘制地下的物体，这样做更有必要了。因此 morphing 技术来解决地形的裂缝是一个很好的选择。

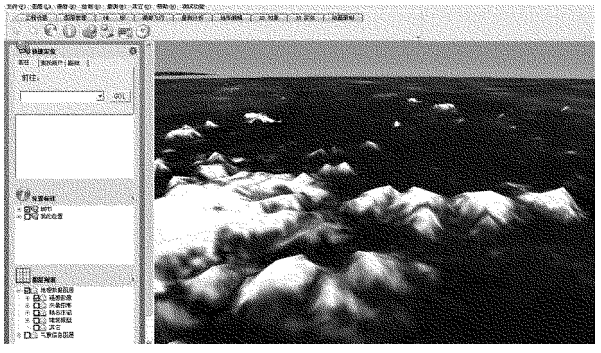


图 8　裂缝消除后

图 9 显示的是随着每一帧绘制的三角形面片的增多，帧率的变化情况。随着每帧绘制的三角形数量的增加，无论是原来基于 CPU 的算法还是基于 GPU 改进的算法帧率都会下降。如图 10 所示，在 3 万个面片之内，帧率可以达到 40 帧以上，完全可以满足漫游过程中实时显示的需求，不会有任何的延迟。结合 Geometry Instancing 的思想，把创建原始的一个顶点缓存和一个索引缓存，通过矩阵变换（平移、缩放、旋转等），便可以得到视野范围内的所有的地形块，在传输带宽不会造成系统瓶颈，充分利用现代显卡高速并行计算的特点，大大地减少了顶点缓存和索引缓存的创建，降低了 CPU 的负担。所以要权衡利弊，实现 GPU 和 CPU 之间的负载均衡，这样不仅可以利用 GPU 的特性，更能把 CPU 从复杂的浮点计算中解放出来，集中精力处理其他的一些问题，这样会得到更好的结果。

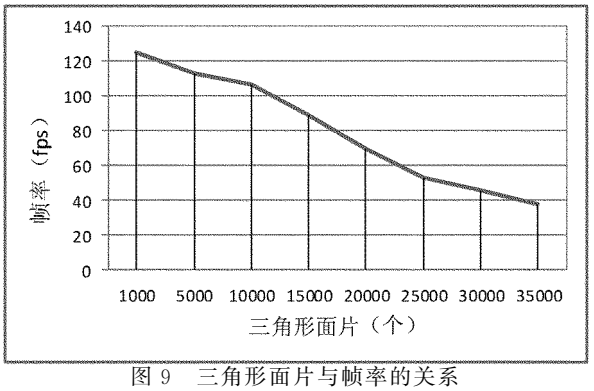
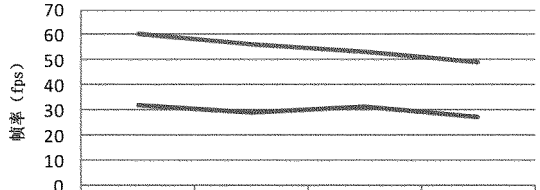


图 9　三角形面片与帧率的关系



	level = 0	level = 4	level = 8	level = 12
本算法	60	56	53	49
原算法	32	29	31	27

图 10　渲染帧率对比

图 9 给出了在不同分辨率大小的情况下，基于 GPU 的 Geometry Instancing 算法和原始的基于 CPU 的算法的帧率对比。从实验数据可以清晰的分析出，GPU 的计算时间要远小于 CPU 的计算时间。从图 10 中曲线的分布可以清晰的看出，比原始的基于 CPU 的地形绘制算法快一倍，效果也比较好，动态漫游能够达到 50 帧每秒左右，而且节省了很多的 CPU 资源，随着计算量的增大，GPU 表现出更加优越的性能。

6　结论

（1）在大规模地形场景下为了满足实时绘制的要求，该系统采用了金字塔（先分层后分块）的方式来组织地形数据，把地形数据存储为高度图，并且该文采用了灵活的方式对高度图进行编码，可以在 GPU 中快速解码还原得到原始的地形数据。

（2）为了充分利用现代 GPU 图形硬件的高速计算能力，减轻 CPU 的负载，该文在绘制方法上进行了改进，原来需要创建可视区内的所有格网，采用了 Geometry Instancing 地形绘制方法，只需要创建

一个模板格网，然后通过偏移和缩放就能够得到整个可视区内的所有其他格网信息，大大减少了构建格网所花费的时间，提高了绘制速度。

(3)随着图形硬件的发展，使得 GPU 资源相比于 CPU 非常的丰富，所以尽量把计算消耗比较大的处理操作交给 GPU 来处理：结合 Shader Model 3.0 引入的顶点纹理拾取技术，将高程值存储在顶点纹理中，在顶点着色器里查找并获取顶点的高程值，避免了在 CPU 上执行顶点缓存的更新；在构建模板格网时就预留了裂缝处理信息，所以可以在 GPU 中直接缝合裂缝，无需其他的操作；利用 GPU 硬件实现裁剪和背面剔除，减少不必要的数据绘制，加速可视化；为了得到更加逼真的效果，最后给出了光照和法线的动态计算，以往有很多地形绘制算法中都是通过法线图来获得顶点的法线信息，而在网络的环境下，通过传输法线图来获得数据，一定会增加网络传输的压力，最终将影响绘制的效率，由于高速并行图形处理器 GPU 的出现，完全可以把法线在 GPU 上动态生成，然后把法线映射到对应的点上，而且法线动态计算对 GPU 负载的增加很小，不会影响绘制性能。

7　展望

通过对大规模地形实时绘制算法的讨论，提出了自己的算法，在一定程度上提高了实时绘制效率，增强了绘制效果，但仍然存在许多问题，有待进一步深入研究。另外随着图形硬件的飞速发展，虚拟现实技术的不断成熟，新的算法、理论和应用层出不穷，面对日新月异的技术更新，更需要不断探索实践。在网络大规模地形实时绘制过程中，要保证实时性和绘制效果的真实性，不仅要考虑客户端绘制速度，还要考虑网络带宽、数据安全性和稳定性，服务器负载等方面才能达到真正实时绘制的要求，该文将在以下几个方面进一步深入的研究：

(1)依据网络带宽，要采用合理的地形数据存储方式和压缩方法，保证网络传输的及时性，同时采用合理的缓存机制和预取策略，对数据进行预取和缓存，保证绘制的实时性，满足漫游者的需求。

(2)该文的剖分算法是基于球体的，但是球体剖分有一个很大的缺点就是两极的误差会很大，如果有需要两极方面的精确应用时，就不能很好的满足实际需要。因此需要进一步研究新型的地形剖分算法，比如椭球体剖分等，使更加符合真实的地球形状，满足两极应用的需求。

(3)充分利用新一代的图形硬件 GPU 的特性，通过使用 DirectX 10 和 DirectX 11 的新特性，如 Geometry Shader(几何体着色器)、Compute Shader(提供了多线程处理能力)和 Tessellation(镶嵌化细分技术)等，可以快速生成大规模的复杂的地形，具有更多的层次细节，提高了可视化的效果，增强了沉浸感，是人们更加相信他所漫游的环境。

参考文献：

[1] 李德仁，肖志峰，朱欣焰，等.空间信息多级格网的划分方法及编码研究[J].测绘学报，2006，(1)：17－19.

[2] R.Pajarola，E. Gobbetti. Survey on Semi－Regular Multiresolution Models for Interactive Terrain Rendering. The Visual Computer. 2007，23(8)：583－605.

[3] Filip Strugar. Journal of graphics，GPU and game tools. 2010，7(1).

[4] Gerasimov，Philip，Randima Fernando，and Simon Green. Shader Model 3.0：Using Vertex Textures. NVIDIA white paper DA－01373－001_v00，June 2004.

[5] Asirvatham A，Hoppe H. Terrain Rendering Using GPU－Based Geometry Clipmaps. 2005：27－45.

[6] J. Schneider，R. Westermann. GPU Friendly High－quality Terrain Rendering. Journal of WSCG，2006，14(1)：49－56.[7] Cebenoyan，Cem. Graphics Pipeline Performance. 2004.

[8] developer.nvidia.com. NVIDIA 官方网站.

[9] O'Rorke，John. Managing Visibility for Per－Pixel Lighting. 2004.

[10] Dudash，Bryan. Technical Report：Instancing. In NVIDIA SDK 9.5. Available online at http://developer.nvidia.com. 2005.

[11] Bent D Larsen，Niels J. Christensen. Real－time terrain rendering using smooth hardware optimized level of detail[J]. Journal of WSCG，(S1213－6972)，2003，11(1).

[12] Daniel Cohen－Or，Yiorgos Chrysanthou，Claudio Silva and Fredo Durand. A survey of visibility for walkthrough applications. IEEE Transaction on Visualization and Computer Graphics，2003，9(3)：412－431.

Research on Accelerated Rendering Technique

# of Global Terrain Based on GPU

LI Hao, XIANG Hengmao, SUN Jiuhu

(Shandong Surveying and Mapping Institute of Land and Resources, Shandong Jinan 250102, China)

Abstract: Through studying the method for making large scale terrain datas in real-time based on graphic processing units, pointing to the characteristics of the batch of rendering of GPU, an improvement method of geometry instancing has been put forward. It can be used to render global grid quickly. The terrain datas are recoded and stored in height maps. Combing with the vertex texture fetch capability in Shader Model 3.0, and decoded in GPU, the primitive DEM datas can be gained. It will avoid carrying out vertex buffer on CPU. This algorithm can not only reduce the CPU load, but also reach a higher frame rate.

Key words: Graphic processing unit; real-time mapping; quad tree; tiled block; height maps