文章编号:1001 - 9081(2009)08 - 2105 - 04

基于 GPGPU 的海量山地地形数据的实时绘制算法

王 春,马纯永,陈 戈
(中国海洋大学 信息科学与工程学院,山东 青岛 266100)
(chwvip@ hotmail. com)

摘 要:针对山地地形海量数据的特点,基于 GPU 的 Geometrical Clipmap 算法,应用简化的工作流程,结合 GPGPU 技术,采用了一种更为合理的高程数据组织交换模式,通过引入高程误差数据巧妙地解决不同分辨率之间的裂缝问题,并对高分辨率的遥感影像作为地形纹理的实现方法加以补充,进而实现可应用于虚拟现实系统的海量地形数据的实时可视化。

关键词:基于 GPU 的通用计算;几何体剪切图;山地地形;海量数据
中图分类号: TP391.9;TP391.41 **文献标志码**:A

Real-time rendering algorithm of massive mountainous terrain data based on GPGPU

WANG Chun, MA Chun-yong, CHEN Ge
(*College of Information Science and Engineering, Ocean University of China, Qingdao Shandong* 266100, *China*)

Abstract: Concerning the characteristics of the huge data in mountainous region, the General- Purpose Computation on Graphics Processors (GPGPU) technology based on Geometrical Clipmap algorithm was introduced. It adopted easier working process, compressed the elevation data and error data into the floating-point type veins, and drew accurate mountain body grid in Vertex Shader. By introducing elevation error data, the crack problem between different resolutions was resolved. The primitive drawn number was reduced by designing reasonable database management. Remote sensing image with high resolution was used as topography grain and the real-time visualization of enormous clipmap data in visual reality system could be realized.

Key words: General-Purpose Computation on Graphics Processor Unit (GPGPU); geometrical clipmap; mountain terrain; massive data

0 引言

海量地形数据实时绘制是计算机图形学、三维地理信息系统和虚拟现实与仿真领域的重要研究内容之一。由于地形数据的海量特征以及当前图形显示硬件条件的限制,给大地形的三维可视化带来了严峻的挑战。

针对大地形渲染,国内外涌现出了许多优秀的 LOD 算法:Lindstrom 等人提出一种基于规整网格的连续细节层次实时高度场绘制算法[1-3],以二叉树为基础的实时优化适应网格(Real-time Optimally Adapting Meshes,ROAM)[4],Hoppe 提出的视点相关递进网格(View Dependent Progressive Meshes,VDPM)[5-6]等。这些算法根据视点位置,每次渲染模型的一部分,并且距离视点远的地方使用比较粗糙的细节,距离视点近的地方使用比较精细的细节,这样可使每次送入渲染管线的三角形个数大大减少。

然而随着最新图形硬件的普及和绘制能力的提高,传统的 LOD 算法已经不能很好地适应当今的硬件架构,尤其针对山地地形起伏大、需要表现的特征细节多的特点,过多的 DP (Draw Primitive)调用让 GPU 大部分时间处于空闲状态,从而不能达到一个满意的渲染效果。针对山区地形的特征,本文基于 Geometry Clipmaps 算法,引入基于 GPU 的通用计算(General-Purpose Computation on Graphics Processor Unit, GPGPU)技术来进行地形绘制,着重利用显卡中图形处理单位(Graphic Processing Unit, GPU)的可编程性能来实现处理 3D 计算以外的物理计算应用。

1 GPGUP 特性及相关研究

最近几年,GPU 得到了迅速的发展,商用 GPU 已经从以往的固定流水线操作模式发展成可编程流水线模式,其编程指令经过了三个版本的更新,逐步发展成为可以直接运用的功能强大且成本较低的并行程序处理器。GPU 提供了可以完全支持向量的操作指令和符合 IEEE32 位浮点格式的顶点和像素处理功能,还将提供越来越大的存储带宽和计算能力。基于 GPU 的通用计算(GPGPU),着重于将 GPU 架构中的顶点处理器或像素/片元处理器替换成其他可执行的程序,把显卡的图形图像处理流程中的着色这一步替换为通用计算。随着第三代 GPU 的诞生,Shader 3.0 模型[7]带来了更大的指令长度、指令个数以及通用程序加子程序调用的程序形式,GPU 的可编程性有了很大程度的提升,同时也使得在像素处理器着色器中的流体数值模拟计算程序有了实现的可能。

收稿日期:2009 - 02 - 25;**修回日期**:2009 - 04 - 09。
基金项目:国家科技部中小企业创新基金项目(07C26223101272);上海市科委科技攻关项目(065115007)。
作者简介:王春(1982 -),男,山东菏泽人,博士研究生,主要研究方向:VRGIS、智能交通; 马纯永(1984 -),男,山东潍坊人,博士研究生,主要研究方向:VRGIS、城市规划; 陈戈(1965 -),男,山东青岛人,教授,博士生导师,主要研究方向:海洋遥感、VRGIS。

当前国内外基于GPU的大地形绘制已有不少研究,潘宏伟等人[8]将传统的LOD算法与VBO技术相结合引入了GPU算法。对于本文讨论的山地地形,传统的LOD算法本身存在的一些缺陷,导致绘制效率得不到最大限度的提高,康宁等人[9]讨论了Geometry Clipmaps算法,并提出了一些改进算法,但依赖于CPU的计算量过大,没有充分发挥GPU高速计算的优势。Asirvatham等人[10]利用GPU-Based Geometry Clipmaps算法讨论了地形的压缩与合成,提高了地形显示的效率和绘制效果,但同时也增加了算法实现的难度。

在此,我们采用了一种简单的实现方式,通过预处理各层高程纹理并结合有效的更新机制来实现地形网格的绘制与更新,将高程数据与误差数据压缩到一张浮点型纹理中,在GPU的顶点着色器中实现精确的山体网格绘制,并通过引入高程误差数据巧妙地解决不同分辨率之间的裂缝问题。最后详细探讨了高分辨率的遥感影像作为地形纹理的实现方法,进而实现海量数据的实时可视化。

2 地形数据的组织及误差的预处理

地形数据可分为两部分:水平网格坐标与高程值。本算法采用一系列以视点为中心的嵌套规则栅格来绘制地形,网格坐标通过VBO技术在显存中创建固定的顶点缓冲区与索引缓冲区,而高程值可转化为2D纹理存储。绘制时,经过适当的坐标变换,将缓冲区的数据动态地拼合成地形,并通过合理的纹理更新机制及精确的纹理采样得到对应的高程值,大大减少了Draw Primitive次数,从而提高地形绘制效率。绘制时栅格组织模式如图1所示。

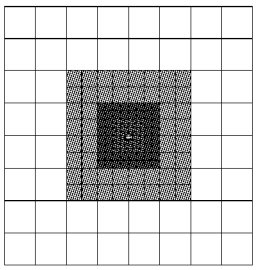

图1 地形绘制组织模式

2.1 坐标数据组织

采用多层次嵌套的规则栅格来绘制地形,使坐标数据具有相当的规律性与周期性,易于数据组织与管理。但不能简单地采用如图2所示的方式来组织坐标数据。因为在视点移动过程中,由于采样数据的离散性,需保证地形的更新步长为最外层单元栅格的边长,否则会导致渲染画面的晃动。但若地形数据层数过多,更新步长可能超过精细层的半径,造成视点附近为较粗糙层栅格,并导致视点附近的地形抖动过于明显,严重影响地形的绘制效果。

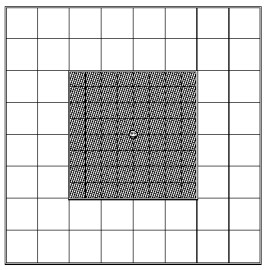

图2 简单坐标数据组织

为解决以上问题,本算法采用如图3所示的坐标组织方式,第L层地形跨越的区域大小为$2^{(L-1)} \times 2^n \times 2^n$,并在每层栅格的最外层保留一个与下一级分辨率一致的环行模块。视点更新过程中,各层首先进行自身内部调整,使得每层的更新步长为自身单元栅格的边长,从而解决之前提到的步长问题,保证栅格更新的实时性。同时,各层之间的更新相对独立,将不规则的环行区域绘制与变形控制在各层的内部,最大限度地保证坐标数据的规律性与周期性。

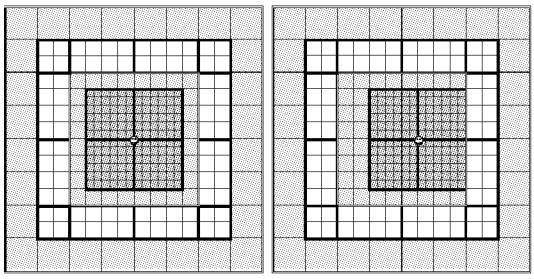

图3 改良坐标数据组织

2.2 误差数据的预处理

由于分辨率不同而造成的裂缝问题是地形绘制的难点之一。在本算法中,采用了相对简单的裂隙消除算法。为消除裂缝,在数据组织过程中,需对各采样点进行误差的预处理,其中误差数据是指裂缝处精细层数据高程值与粗糙层数据均值的差。由于各层的更新步长为每层单元栅格边长,使每个坐标点都有唯一的误差,对于任一坐标点(col,row)的误差δ计算算法如下(其中记H(col,row)为(col,row)点的高程值):

```
if((col为奇 && row为奇) || (col==0 && row== 0))
  δ=0;
else if(col==0 && row!=0)
{int k=row的2n的最大约数;
δ=(H(0,row+k)+H(0,row-k))/2-H(0,row));}
else if(col!=0 && row==0)
{int k=col的2n的最大约数;
δ=(H(col+k,0)+H(col-k,0))/2-H(row,0));}
else {int k =col与row的最大公约数)
{if(k为2n的整数倍(n>0))
 {col=col/k; row=row/k;
  if(col为偶)
    δ=(H(col*k,row*k+k)+H(col*k,row*k-k))/2-
       H(col*k,row*k));
  else if(row为偶)
    δ=(H(col*k+k,row*k)+H(col*k-k,row*k))/2-
       H(col*k,row*k));
  else δ=0;}
 else δ=0;}
```

3 基于GPGPU的地形绘制

基于GPGPU的海量山体地形实时绘制流程如图4所示。

3.1 栅格绘制

在网格的组织过程中,采用VBO技术为每层网格创建了对应的Vertex Buffer。如图3所示,假设单层分辨率为$(2n+1)*(2n+1)$,第一层数据的顶点缓冲区可分为三部分:大小为$(2n-1-1)*(2n-1-1)$的复用缓冲区,外层的环形网格及调整后的U型数据。其他层则需另外创建两个可复用的顶点缓冲区,大小分别为$(2n-2-1)*(2n-2-1)$和$(2n-2+1)*(2n-2+1)$。具体绘制网格的过程中,需要根据视点

位置确定各层栅格中心点的坐标及当前绘制的状态以选择合适的 Vertex Buffer,通过合适的坐标变换来拼合地形。

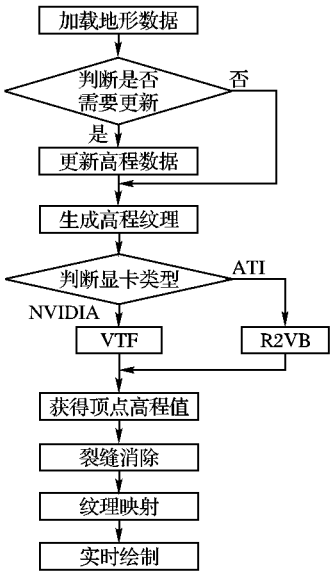

图 4　基于 GPGPU 的山体地形实时绘制流程

假设精细层的栅格边长为 CellSize,并枚举层内的绘制状态为: ENUM STATE { LEFTDOWN = 0, LEFT, LEFTUP, DOWN, CENTER, UP, RIGHTDOWN, RIGHT, RIGHTUP}。则第 L 层的绘制中心及绘制状态对应的关系为:

$$X = (\text{int})floor[Camera.x/(2L * CellSize)] * (2L * CellSize)$$

$$Z = (\text{int})floor[Camera.z/(2L * CellSize)] * (2L * CellSize)$$

$$STATE = (X\%2 + 1) + (Y\%2 + 2)$$

3.2　高程采样

根据各层中心点的坐标(X,Z) 可计算当前层对应包含高程信息的纹理坐标, 假设地形总的分辨率为$(2m + 1) * (2m + 1)$,精细层的分辨率为$(2n + 1) * (2n + 1)$,则第L层中心点的 U,V 计算公式为:$U = (X/2m) * (2(m - n - 1 - L))$,$V = (Y/2m) * (2(m - n - 1 - L))$。

在 Vertex Shader 中通过中心点 U,V 坐标,计算每个坐标点的 U,V, 然后利用函数 tex2Dbias( )对该层对应的纹理数据进行采样,取得纹理数据 T,通过公式 $H = floor(T)/10n$ 解压计算该坐标点的高程值。

3.3　GPGPU 中的高程纹理生成

经过以上误差的预处理过程,每个坐标点对应唯一的高程与误差。GPGPU 技术中二维数组对应一张浮点型的纹理,坐标值相当于纹理中的 U,V。为了减少 GPU 计算中的纹理采样次数,将各点的高程与误差数据压缩到一个浮点型数据中。如果高程 H 的精度为 10^{-n},差值 δ 的最大值不超过 $10m$,则保存到纹理中的数据为 $T = H * 10n + \delta/10m$。

当前显卡支持纹理的最大尺度一般为 4 096 × 4 096。若地形的尺寸在 4 096 × 4 096 以内,整个地形的绘制可以维持一张纹理,但是如果地形的网格数大于该尺寸,则需要为每层创建一个 2D 浮点型纹理,并且各层纹理随视点的移动而更新。为保证数据更新不影响网格的绘制,对各层 $(2n + 1) * (2n + 1)$ 的网格,可创建$(2n + 1) * (2n + 1)$ 的2D 纹理。

3.4　高程图的更新

如果地形的尺寸超过了 4 096 × 4 096,在数据组织过程中,对应各层 $(2n + 1) * (2n + 1)$ 的栅格创建了一张$(2n + 1) * (2n + 1)$ 的 2D 纹理,显然该纹理不能覆盖整个地形区域,因此视点的移动过程中,需要合理的纹理的更新机制才能保证网格数据采样值的有效性与合理性。

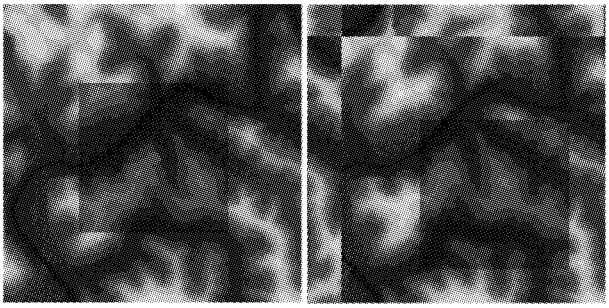

图 5　高程纹理更新示意图

如图 5 所示:纹理采用以块为单位的环行更新机制,将整个地形高程纹理进行分层、分块、编码管理。并为每块纹理数据设置一个标记位,标记当前模块是否在显存中。实现时,可将 $(2n + 1) * (2n + 1)$ 纹理切割为 8 × 8 的纹理块,在视点移动时,判断其周围的 64 块纹理是否在显存中,如果不在则触发局部纹理更新。其中对于编号为(M,N) 的纹理模块,在显存对应的更新编号为$(M\%8,N\%8)$。

3.5　裂缝消除

本算法中消除裂缝的基本思想是:通过引入误差数据,调整精细层对应点的高程值,保证在裂缝处精细层的高程值等于粗糙层的均值,从而快速地消除裂缝。为此需要通过以上取得的纹理数据解压获取误差数据,其计算公式为: $\delta = frac(T) * 10m$。得到误差数据后在 Vertex Shader 中对每个点需要进行以下融合处理(假设点相对中心的偏移分别为 λx, λy):

$$Alpha = \max(STEP(1,(\text{abs}(\lambda x)/(2n - 1 - 2)), STEP(1,(\text{abs}(\lambda y)/(2n - 1 - 2))))$$

$$H = H + Alpha * \delta$$

4　纹理映射

在大地形仿真中,地形纹理数据数据量庞大,通常超过系统内存容量。通过大规模纹理的实时装载[11]可以有效解决从硬盘到系统内存及从系统内存到纹理内存之间的调度问题,实现大规模高分辨率影像纹理的实时映射。

在纹理的绑定过程中,由于纹理数据所对应的地理位置固定,嵌套网格随视点的移动而动态变化,所以在栅格的更新过程中会造成单个顶点缓冲区跨越不同纹理区域的情况。为解决以上跨越情况的纹理映射,数据组织时需在显存中创建两种类型的索引缓冲区,即沿 X 轴方向创建的 *IndexBufferx* 与沿 Z 轴方向创建的 IndexBufferz,针对不同的情况我们可以方便地调用 IndexBufferx(z) 来进行绘制。

5　实验结果

我们利用上述算法,对 4 097 × 4 097 的地形数据和 32 768 × 32 768 的纹理数据进行分块分层,其中地形数据块的大小为 257 × 257,纹理数据块的大小为 512 × 512。

计算机的配置为 P4 双核 1.86 GHz,内存为 2 GB,显卡为 NVIDIA Quadro FX4500,平均速度能达到:128 fps,CPU 的平

均消耗为14%。对该地形数据采取传统的金字塔绘制方式与该算法进行了绘制性能及效果的比较,试验结果与地形、视点移动速度等多方面因素相关,以下结果仅供参考。

表1 金字塔算法与本算法性能对比

算法名称	平均帧速/fps	CPU资源平均消耗/%
金字塔算法	34	42
本文算法	128	14

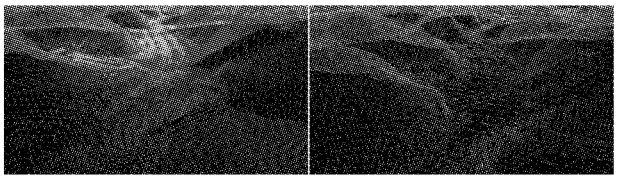

图6 本文算法与金字塔算法网格显示对比

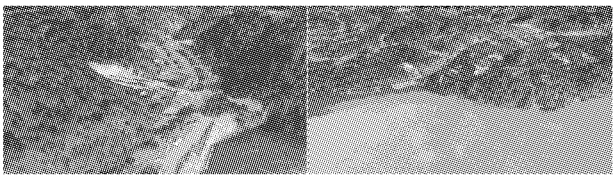

图7 乐山市地形运行效果

以上实验表明,基于Geometry Clipmaps算法,并采用GPGPU技术对山体地形进行绘制,通过引入误差数据,调整精细层对应点的高程值,快速消除裂缝;创建索引缓冲区,实现大规模高分辨率影像纹理的实时映射,可以优化绘制性能,减少CPU资源消耗,提高系统效率。

本文对海量山体地形数据可视化的相关理论、方法及应用技术进行了分析和归纳,随着图形硬件技术的发展,大规模地形可视化算法逐渐利用硬件可编程语言将部分算法移植到GPU上执行。本文还对模型简化引起的裂缝等空间连续性问题进行了阐述,并且提出了相关算法。随着数字摄影测量技术的发展,我们可以很方便地获得高分辨率的高程和纹理数据,同时硬件厂商不断提升图形硬件的性能,相信这将为山体地形可视化技术的研究提供更大的发展空间。

参考文献:

[1] LINDSTROM P, KOLLER D, RIBARSK W, *et al*. Real-time continuous level of detail rendering of height fields [C]// SIGGRAPH'96 Proceedings. New York: ACM, 1996: 109-118.

[2] LINDSTROM P, PASCUCCI V. Terrain simplification simplified: A general framework for view-dependent out-of-core visualization [J]. IEEE Transaction on Visualization and Computer Graphics, 2002, 8(3):239-254.

[3] LINDSTROM P, KOLLER D, RIBARSKY W, *et al*. Real-time, continuous level of detail rendering of height fields [C]// Proceedings of the ACM SIGGRAPH Conference on Computer Graphics. New York: ACM, 1996: 109-117.

[4] DUCHAINEAU M, WOLINSKY M, SIGETI D E. Roaming terrain: real-time optimally adapting meshes [C]// Proceedings of SIGGRAPH'97. New York: ACM, 1996: 81-88.

[5] HOPPE H. View-dependent refinement of progressive meshes[C]// Proceedings of SIGGRAPH'97. New York: ACM, 1997: 189-198.

[6] LOSASSO F, HOPPE H. Geometry clipmaps: terrain rendering using nested regular grids [J]. ACM Transactions on Graphics (TOG), 2004, 23(3): 769-776.

[7] BUCK I, HANRAHAN P. Data parallel computation on graphics hardware [EB/OL]. [2009-02-01]. http://hci.stanford.edu/cstr/reports/2003-03.pdf.

[8] 潘宏伟, 李辉, 廖昌阊, 等. 一种基于现代GPU的大地形可视化算法[J]. 系统仿真学报, 2007,19(14):3241-3244,3275.

[9] 康宁, 徐青, 周杨, 等. 一种基于图形硬件的海量地形实时可视化算法[J]. 系统仿真报, 2007,19(17):3988-3992.

[10] ASIRVATHAM A, HOPPE H. Terrain rendering using GPU-based geometry clipmaps[EB/OL]. [2008-12-31]. http://research.microsoft.com/~hoppe/gpugcm.pdf.

[11] 陆艳青. 海量地形数据实时绘制的技术研究[D]. 杭州: 浙江大学, 2003.

(上接第2094页)

信息的摄像头改为红外传感器,将需识别的图像数据改为离散数据,降低了需处理的数据的复杂度;同时通过定义一些符合人体步态特征的特殊的复合步态模式,实现对通道中的物体和行人进行识别。实验表明算法是有效的。

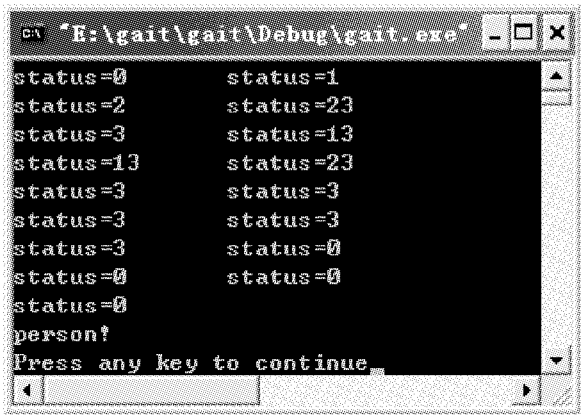

图7 算法程序的运行结果

参考文献:

[1] 郑晓雯, 王仑杰, 李彩琴, 等. 基于步态的人体身份检测与识别[J]. 计算机工程与应用, 2004,40(5):82-83,130.

[2] HAYFRON-ACQUAH J, NIXON M, CARTER J. Automatic gait recognition by symmetry analysis[J]. Pattern Recognition Letters, 2003,24(3): 2175-2183.

[3] LITTLE J, BOYD J. Recognizing people by their gait: The shape of motion[J]. Journal of Computer Vision Research, 1998, 1(2): 2-32.

[4] YAM C, NIXON M, CARTER J. Gait recognition by walking and running: A model-based approach [C]// Proceedings of Asia Conference on Computer Vision. Melbourne, Australia: [s.n.], 2002, I: 1-6.

[5] 王亮,胡卫明,谭铁牛. 基于步态的身份识别[J]. 计算机学报, 2003,26(3):1-8.

[6] NIYOGI S, ADELSON E. Analyzing and recognizing walking figures in XYT[C]// IEEE Computer Society Conference on Computer Vision and Pattern Recognition. [S.l.]: IEEE, 1994: 469-474.

[7] GREEN R D, GUAN L, BURNE J A. Video analysis of gait for diagnosing movement disorders [J]. Journal of Electronic Imaging, 2000, 4(1): 16-21.

[8] CHENG J C, MOURA J M F. Capture and representation of human walking in live video sequence [J]. IEEE Transactions on Multimedia, 1999,1(2):144-156.

[9] 曲日. 城市轨道交通中闸机智能识别系统及其识别技术的研究[D]. 天津: 天津大学,2005.