基于层次块的大地形实时细节合成及渲染算法

张燕燕, 黄其涛, 韩俊伟

(哈尔滨工业大学 机电工程学院, 哈尔滨 150001)

摘 要:为解决渲染真实大地形场景面临的数据获取困难问题,减轻海量地形数据存储与调度的压力,提出了基于层次块实时生成可控的地形细节,完成地形渲染的新方法。采用了层次块结构组织采样数据。给出了基于层次块生成可控细节的方法,使用块网格细分增加采样点,提出了基于 Perlin 噪声的可控多重分形算法计算细节,保证了生成的细节受真实采样点属性控制。采用统一的细节选择标准完成了细节合成与 LOD 方法集成,其结果与原始地形块共同构成了动静结合的双层自适应层次结构,实现了视相关的大地形渲染。面向 GPU 实现了实时合成算法。实验分析表明了该方法能够利用有限的真实地形数据实时产生与实际地形特征相近的细节,完成大地形渲染。

关键词:计算机应用;大地形可视化算法;实时细节合成;层次细节模型;多重分形;GPU 算法
中图分类号:TP391 文献标识码:A 文章编号:1671-5497(2009)Sup. 2-0406-07

Real-time detail synthesis and rendering of large terrain based on data chunk

ZHANG Yan-yan, HUANG Qi-tao, HAN Jun-wei

(*School of Mechatronics Engineering, Harbin Institute of Technology, Harbin* 150001, *China*)

Abstract: A new method for real-time controllable detail synthesis and rendering of large terrain based on data chunk was proposed in order to overcome the difficulty in accessing the dataset for rendering the real large terrain, and reduce data storage and management pressure. Chunk hiberarchy is adopted to organize samples, and the controllable detail synthesis presented is chunk-based. Chunk subdivision is used to generate new samples and a Perlin-noise based controllable multifractal algorithm is given to compute details. The influencing factor for the fractal makes sure the details generated are controlled by the real terrain features. A uniform refine criterion is studied to integrate detail synthesis and the view-dependent LOD rendering mechanism. The new chunks synthesized together with original chunks form a two-level adaptive hierarchy, and large terrain rendering is completed. The method is implemented in a GPU-oriented way, and real-time synthesis algorithm is achieved on GPU. Experiments show that the method can real-time produce fine large terrain scene similar to the true terrain using limited terrain samples.

Key words: computer application; terrain rendering; real-time detail synthesis; level of details; multifractal; GPU algorithms

收稿日期:2008-10-28.
基金项目:教育部新世纪优秀人才支持计划项目(NCET_04_0325).
作者简介:张燕燕(1981 -),女,博士研究生.研究方向:视景仿真,三维可视化技术. E-mail:zhangyanyan963 @163.com
通信作者:韩俊伟(1964 -),男,教授,博士生导师.研究方向:电液伺服理论与应用,并联机器人和现代控制理论与应用,系统仿真与虚拟现实技术. E-mail:hjw @hit.edu.cn

实时渲染真实的大地形场景一直是近年来计算机图形学及相关领域的研究热点,在游戏、飞行训练、军事演习模拟等领域有着广泛的应用。为保证生成的地形能够反映真实的地表特征,目前常用的是基于真实地形采样数据的渲染方法[1-3]。而对于大规模地形的可视化,海量地形数据的存储及传输调度往往成为这类方法的一个主要问题,同时高精度真实采样数据获取的困难限制了其生成地表的精细程度。与此相对的是基于程序合成的渲染方法[4-7],其使用程序计算的方法生成地形渲染所需的地表数据,因而不存在数据获取问题,能够表现任意精细的层次细节,同时地形数据可根据渲染需要实时生成,数据量较小。但该方法生成的地形没有与真实的地形、地貌相联系,形状较难控制。因此将两种方法结合,即采用真实地形数据构造地形的大致形态,利用程序化的方法实时合成细节、增加地表细节特征是一种有效的方法。Boubekeur[8]给出了一种通用的自适应精化框架,能够使用粗糙的地形网格借助于位移图合成精细的地表。Dachsbacher[9]基于图像空间的自适应网格变形完成了细节合成。Losasso 等[10,11]使用最基本的分形技术将实时细节合成集成到其 Geometry Clipmaps 结构中,实现了大地形的实时渲染。这些研究大多以单个三角形作为细化的基础,而对于大规模地形,基本三角形的个数将受到系统资源的限制,阻碍生成精细的真实地表。同时其中的细节合成方法也很少考虑真实地形特征对细节生成的影响。另外,细节合成方法与LOD 技术集成完成大地形渲染方面也并未进行详细的研究。

为解决渲染真实大地形场景面临的数据获取困难问题,减轻海量地形数据存储与调度的压力。针对大地形特点,研究可控的实时细节合成方法,将真实地形数据与细节生成相结合,集成 LOD 技术,利用有限的地形采样数据,实时生成具有足够精细真实地表细节的地形场景是本文的重点。本文考虑大规模地形的特性,将原始地形数据组织成地形块层次,针对地形块结构设计网格细分和可控的多重分形算法,生成能够反映真实地表特征的动态地形块。通过统一的细节选择标准将细节合成与 LOD 技术集成,生成的块与真实地形块共同构成动静结合的双层自适应地形块层次结构,完成视相关的地形渲染。合成算法面向 GPU 实现以保证算法的实时性。力求使用有限的真实地形数据,实时生成与真实地形特征相近的地形细节,自适应地显示具有足够精度的真实地形场景。

1　算法概述

本算法是一种真实地形数据与程序化细节生成相结合的方法,采用真实地形数据块构造地形的大致形态,并据此控制细节合成,实时生成满足当前视点精度要求的地表特征,进而完成地形渲染,整个算法的框架如图 1 所示。算法将真实的地形数据组织成便于 GPU 处理的规则的分层数据块,实时细节合成以真实地形块为基础在 GPU 中实现。程序化的细节合成包括网格细分和细节高程生成两部分,生成的细节受视点位置以及地形高度,地表梯度,粗糙程度等地表特征控制。细节合成方法与基于块的视相关 LOD (Level of Details) 机制集成,即根据当前视点决定具有不同层次细节的地形块表示整个可见的地形区域,当现有的真实地形数据不能表现视点所需的地表细节时,使用实时细节合成方法形成动态的精细层次地形块。生成的动态地形块使用缓存机制管理,与原始地形块一起构成动静结合的双层自适应地形块层次结构。相应的地形场景采用分级渲染方法,对于能够使用静态地形块表示的地形区域直接渲染,否则渲染实时合成的动态地形块。

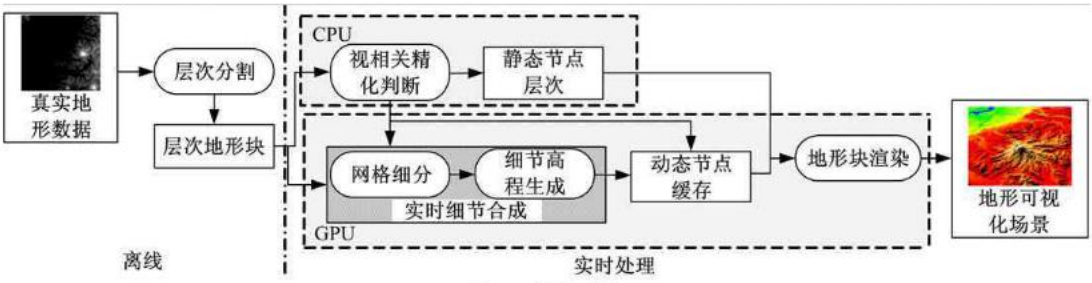


图 1　算法框架

Fig. 1　Algorithm architecture

2 层次地形块

为适应地形大规模的特性,同时发挥 GPU 并行处理的能力(见第 4 节),算法将原始地形采样数据组织成由固定大小的规则地形数据块组成的金字塔层次结构。图 2(a)为该结构的俯视图(以三层数据为例),不同颜色的线框表明该层地形块数据覆盖的区域,框内与层次对应的顶点表明该地形块的采样数据集。层次结构采用自底向上的方式构造,其中顶层的数据采用 2 倍率隔行采样的方式由相邻底层获得。每层被分隔为大小相同的地形块,单个地形块如图 2(b)所示(以 5 × 5 采样点为例),采样点均匀分布,具有相同的顶点数和拓扑结构。数据块的数据集为 P_l(见式(1)),$H(x,z)$ 为高程采样函数,i,j 为块内的网格坐标,x, z 为物体空间坐标,l 为层数)。根据算法需要,将每个地形块具体的地形数据扩充为多个属性通道存储,分别保存顶点的位置坐标,法向量和粗糙度值。

$$P_l = \begin{cases} V_{i,j} \mid V_{i,j} = H(x,z), 0 \leqslant i,j \leqslant p_{\text{size}}, \\ x = p_{\text{xoff}} + 2^l \quad i, z = p_{\text{zoff}} + 2^l \quad j \end{cases} \tag{1}$$

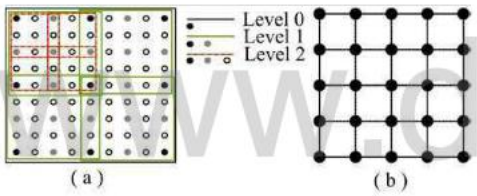


图 2 地形块及其层次结构示意图

Fig. 2 Terrain chunk and its hierarchy

3 实时细节合成

3.1 网格细分

为增加地形块的细节,需要首先添加额外的地形数据点,增加地形块的采样率。新数据点的初始属性由原网格点采用相应的细分函数 S 获得,如式(2)。其中 Λ 是与当前计算顶点 $V_{i,j}$ 相关的顶点集。细分函数依据具体的细分机制而异,最简单的方法是对原始地形块的数据点进行线形插值生成新的数据点。但是由于线形插值是 C^0 连续的,这必然会使地形表面产生比较尖锐的锯齿状特征,为此需要使用一个较为平滑的细分函数计算新增的数据点。

$$V_{i,j} = S(\Lambda, i, j) \tag{2}$$

Catmull-Clark 细分可保证 C^2 连续,本文以 Catmull-Clark 细分理论作为细分的基础[12-13,17]。细分采用二倍率的扩充方式进行,每个地形块可细分为由四个等大小的地形块构成的精细网格。用于生成新增顶点的细分函数可表示为式 3,其中 Λ 为以 $V_{i,j}$ 为中心的相邻粗糙地形块的顶点集(顶点集的构造依据待求顶点的属性而不同),$M^{i,j}$ 为顶点集对应的权重矩阵。细分的平面网格结构如图 3 所示(以 5x5 采样点为例)。这里为保证整个地形网格细分的一致性,在细分某个地形块时先将其边界向外拓展一个网格单位的采样点(如图 3 所示的蓝色区域),扩展区的采样点从与该块相邻的地形块中获得用于块边界的细分。

$$\Lambda = \begin{bmatrix} V^0_{i-1,j+1} & V^0_{i,j+1} & V^0_{i+1,j+1} \\ V^0_{i-1,j} & V^0_{i,j} & V^0_{i+1,j} \\ V^0_{i-1,j-1} & V^0_{i,j-1} & V^0_{i+1,j-} \end{bmatrix}, M^{i,j} = M(i,j)$$

$$V_{i,j} = \sum_{l=1}^{3}\sum_{k=1}^{3}\Lambda_{l,k} \quad M^{i,j}_{l,k} \tag{3}$$

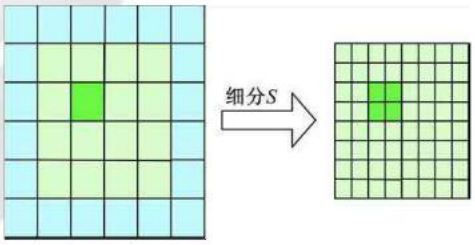


图 3 基于块的网格细分

Fig. 3 Chunk subdivision diagram

3.2 程序化细节生成

由上述网格细分生成的地表过于平滑,且并没有为表面增加更多的细节。因此要生成更加真实的地表外观,需要使用受真实地形数据控制的程序化细节生成方法增加表面的细节丰富程度,即在上述细分顶点位置的基础上移动指定的位移 $V_{i,j}{}^d$。这里为达到减少数据存储和传输量的目的,算法使用细节位移函数实时计算该位移偏移量并与细分的顶点属性合成最终的数据点。

分形布朗运动是描述真实地形最好的随机过程[14],考虑到上述基于块的细分过程的特点,本文使用由具有不同比例自相似样本迭代叠加的分形过程模拟真实的细节偏移量。另一方面传统的分形布朗算法本身生成的细节具有各向同性的特点,这对于以地形块为细化单位的大地形来说必然会带来生成的细节过于单调的现象。因此多重分形的思想被用于改进基本的分形函数,保证分形计算的结果依赖于地形顶点处具体的地形特征。多重分形的位移函数可表示为式(4),这里选

取梯度 Perlin 噪声函数[15]作为分形函数的基函数。该噪声具有较好的频谱特性,且当噪声函数控制网格的分辨率与地形块一致时,能保证在地形块细分前的顶点处随机值为 0,这对于以递归细分的地形块为基础的并行合成算法来说(见第 4 节),以地形块网格为处理对象不改变前细分层已经形成的属性特征,保证整个分形表面的一致性与稳定性。

$$V_{i,j}^{d} = \sum_{l=k_0}^{octaves+k_0-1} Noise\left(lacunarity^{l} \quad V_{i,j}\right) \quad lacunarity^{-Hl} \quad Mul \tag{4}$$

式(4)中 $lacunarity$ 可视为相邻两个频段的频率比率,对于以二倍率细分为特点的细分算法,$lacunarity$ 可固定为 2。分形迭代开始于最精细的原始地形块,起始的噪声函数的频率由 k_0 决定,对于本文的细分算法(噪声函数控制网格的分辨率与地形块一致) k_0 为 0 即可。$octaves$ 代表分形迭代的次数,反映细节的精细程度。本文将该系数同细分的次数关联,每细分一次所用位移函数的 $octaves$ 加 1,这样便将视点对细节程度的影响引入位移函数中。H 与分形维数相关,反映合成地形的粗糙程度,需依据具体的地形特征确定。

此外,Mul 体现多重分形的思想,负责依据真实的地表特征改变分形迭代的结果,其值与单个顶点处的属性值相关,不存在边界不连续问题,如式 5 所示。

$$Mul = M\left(h_{i,j}, s_{i,j}, e_{i,j}\right) \tag{5}$$

顶点属性根据自然界中地形特征的影响因素选取,其中 $h_{i,j}$ 代表顶点的基本高程值,一般较低的地形(如山谷)具有较为平缓的地形特征;$s_{i,j}$ 代表顶点的高程梯度,梯度越大的地形往往具有比较尖锐的特征;$e_{i,j}$ 代表顶点的粗糙程度,可反映地表的材质属性(草地和岩石的地形表面具有不同的粗糙度),为方便计算 M 可选取为基本的线性函数。按照上述的分析,最终用于细节合成的位移函数可表示为式(6)。

$$V_{i,j}^{d} = \sum_{l=0}^{octaves-1} Noise\left(2^{l} \quad V_{i,j}\right) \quad 2^{-Hl} \quad M\left(h_{i,j}, s_{i,j}, e_{i,j}\right) \tag{6}$$

3.3　细节合成方法与视相关 LOD 集成

通过为不同的地形区域指定不同的细节等级,视相关 LOD 方法能够有效地减少算法的运算负载。在本文的动态自适应地形块层次结构中,细节合成方法通过为层次构造动态节点的方式与 LOD 算法集成,动态节点依据 LOD 算法的需求实时生成。其中集成的一个关键问题是需要一个视相关的细节选择标准来决定是否需要使用细节合成方法生成精细的地形块以及控制合成的精细程度。

考虑到算法中动静结合的双层自适应地形块层次结构,这个细节选择标准应当保证合成数据块和真实数据块的选择标准一致。传统的地形算法使用地形块的屏幕空间误差作为细分标准。而在增加了细节合成的算法中地形块的误差计算必然依赖于细分和细节合成,且理论上此过程可无限进行,这就为地形块的误差计算带来了一定的困难。为此,本文选用一种简单的地形块三角形密度作为精化的标准[16],即定义地形精细度为视点观察地形块时视锥范围内平均每弧度所观察到的三角形个数。如图 4 所示,这里假设视锥裁剪面的轴线与地形块的轴线方向一致,则在 $L1$ 和 $L2$ 方向上地形精细度分别为 $\rho_1 = x^2/Dh$ 和 $\rho_2 = x/D$,其中 D 为地形块的大小,x 为视点到地形块的距离,h 为视点到地面的高度。设 d 为地形块的采样间隔,n 为规则地形块的单边采样点数,则 LOD 的精化标准可由式(7)计算。比较地形块精度 f 与指定的三角形密度阈值 τ,当 $f \leqslant \tau$ 时地形块需要精化。另外当上述二者轴线不一致时计算的 f 值较大,则保证地形块具有足够的细节。

$$f = \max\left(\rho_1, \rho_2\right) \quad 2\left(n-1\right)^2 = \frac{x^2}{d\ h} \quad 2(n-1) \tag{7}$$

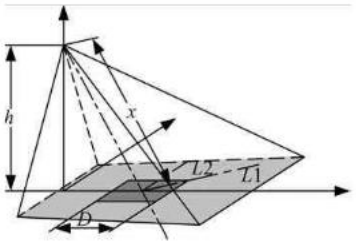


图 4　视锥与地形块的关系

Fig. 4　Relationship between view frustum and terrain chunk

4　面向 GPU 的算法实现

基于上述理论,算法采用规则的地形块结构且在细分和细节生成的过程中每个采样点数据的生成不依赖于同层次的其他数据,这些奠定了算

法面向 GPU 实现的基础。因此本算法将运算量较大的网格细分和细节生成部分放到 GPU 中完成,这样可以利用 GPU 像素着色器并行运算的功能加速运算,同时减轻 CPU 同 GPU 间的带宽压力。具体地,将待处理的地形块数据(多个属性通道)存储为纹理形式,将细分和细节合成算法映射为像素着色单元,通过渲染和要计算的纹理区域相应的矩形激活 GPU 程序,完成对应区域的相关计算。计算过程的结果仍然以纹理的形式存在,最终的动态地形块节点数据被输出到 VBO (Vertex Buffer Object)中供场景渲染使用,减少同 CPU 交换的数据量。渲染程序直接根据节点块相应的 VBO 和索引数组完成渲染。

如图 5 所示,这里为减少不必要的运算量,同时保证生成细节的可控性,需将平滑细分和位移合成分开进行,二者映射为不同的像素着色单元。对于细分,待处理的地形块的属性数据保存在地形纹理 1 中,细分结果指定为纹理 2,其中的每个像素对应细分网格的一个采样点,两纹理中地形块数据分布如图 2 所示。程序激活后着色单元采样纹理 1 中的属性信息,像素点并行执行式(3)所示的细分运算。一次细分完成后纹理 1 与纹理 2 交换角色准备下一次细分。细分结束的地形块数据执行细节合成运算。其中梯度 Perlin 噪声函数以向量纹理和索引置换纹理的形式给出,变化纹理采样坐标的比例可完成不同频率的噪声值计算,细分深度等常数值以常量的形式传入着色器,细节位移值按照式(6)得出并叠加到细分结果上,最终的节点数据输出到 VBO 中。

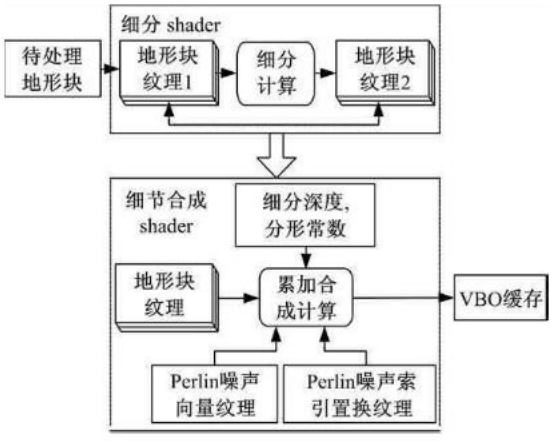


图 5 基于 GPU 的细节合成算法实现

Fig. 5 GPU-oriented implementation for detail synthesis

5 实验结果及分析

本算法实验的硬件平台为 P4 CPU 3.06 GHz,内存 512 MB,显卡 NVIDIA GeForce 7300GT。原始数据选取 Puget Sound 区域的低分辨率数据集 Puget1K (http:// www.cc.gatech.edu / projects / large_models/),数据分辨率为 1025 ×1025,水平采样间隔为 160 m,垂直分辨率为 0.1 m。原始数据覆盖较大的地形规模,数据经预处理生成地形块的层次结构,定义每个地形块含有 65 ×65 个采样点。

算法采用 C++ 实现,像素着色器部分使用 CG 语言完成。程序在 Windows 平台下运行,其水平视场角定义为 90°,窗口分辨率为 1024 × 768,规定视点沿预定义的路径按照固定的方式漫游。程序运行的渲染效果和帧率曲线如图 6(b)和图 7 所示。结果表明本算法能够利用有限的地形数据,以较高的帧率完成具有较丰富细节的真实大地形场景的实时渲染。

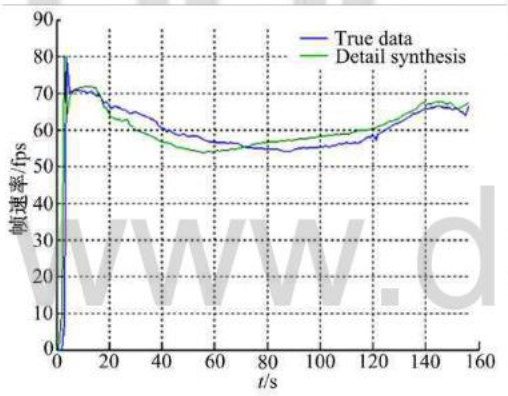
为说明其中可控的细节合成方法的作用和有效性,将本文算法和直接渲染原始数据方法的运行结果进行比较。二者渲染效果见图 6,图 6(a)为直接使用原始采样数据渲染的效果,图 6(b)为启用实时细节合成的效果。对比可看出集成可控细节合成的渲染方法能够在没有足够真实数据源的情况下生成较为真实的地形场景,且生成的精细地形具有和原始地形相似的地形轮廓和符合地形特征的细节。比较文献[8]中的图 11、文献[9]中的图 7 和文献[10]中的图 7 中细节合成渲染方法的渲染效果图,在使用相似规模的低分辨率数据集的情况下,本文的渲染算法获得的地形(图 6(b))具有更加精细和真实的细节特征,避免了细节过于单调的现象。即使当视点贴近地面时仍能够提供足够的符合地形特征的细节信息。

比较本算法和经典的基于真实地形采样数据的渲染方法 Ulrich[2]。本算法的原始数据集为 Puget1K,包含的数据采样点为 1025 ×1025。使用 Ulrich 方法渲染真实高分辨率数据集 Puget16K,包含的数据采样点为 16385 ×16385。二者运行性能的对比曲线见图 7(a)。可看出细节合成方法在初始阶段和场景复杂度突然增加时帧率较低,但随后帧率逐渐提高,达到较高的帧率水平。因而集成细节合成的渲染算法使用实时合成的方法生成地形细节,数据量较小,能够极大地

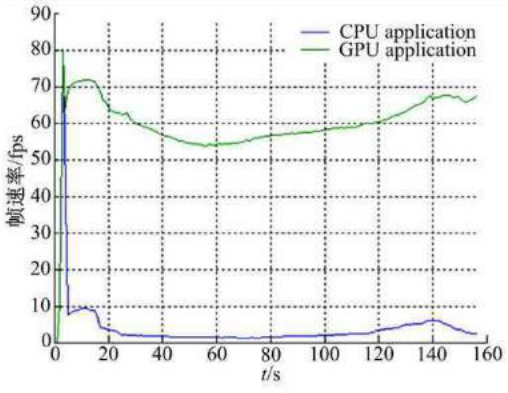
图 6　基于原始采样数据渲染与基于细节合成渲染的效果对比图

Fig. 6　Comparison of effect snapshots between with and without detail synthesis

减少整体数据的存储和传输调度负载。同时算法的整体性能并未有明显的损失。算法在普通的硬件平台下运行的最低帧率不低于 50 fps,平均帧率达到 60 fps 以上,能够保证算法运行的实时性。因而本文的算法是有效的、可行的。这里使用自适应 LOD 精化方法能够减轻算法运行的负载,另外面向 GPU 的算法设计在维持算法的效率方面发挥了重要的作用。



(a)



(b)

图 7　帧速率对比曲线

Fig. 7　Comparison graph of frame rate

为进一步说明本文面向 GPU 算法设计的有效性,将细节合成的主要工作交给 CPU 完成,比较面向 CPU 算法实现和面向 GPU 算法实现的运行效率,其性能曲线如图 7(b)所示。比较表明面向 GPU 的算法平均帧率超过面向 CPU 的实现方法十倍以上。因而本文面向 GPU 的算法设计是有效的。细节合成在 GPU 中完成,充分发挥了像素着色器的并行运算能力,使其承担主要的实时合成运算量,从而降低了对 CPU 资源的依赖,减少了 CPU 和 GPU 间的数据传输量,有利于算法整体性能的提升。

6　结束语

提出了一种基于层次块的大地形实时细节合成及渲染方法,能够利用有限的真实地形数据,采用可控的程序化方法实时生成地形细节,自适应地显示具有足够精度、符合真实地表特征的地形场景。算法减弱了渲染大规模真实地形场景时对获取高精度数据源的要求,在保持渲染质量的前提下,减轻了可视化过程中海量数据存储与传输的压力。

针对大规模地形的特性,区别于基于单个三角形的细分方法,本文将原始数据采样点组织成地形块层次,以地形块作为细节合成的基础,细节合成受视相关 LOD 机制控制,合成的动态地形块与原始地形块层次集成,共同构成了动静结合的双层自适应地形块层次结构,实现了大地形的视相关渲染。细节合成算法针对地形块设计,在平滑的网格细分基础上,使用一种基于 Perlin 噪声的可控多重分形算法计算地形细节,生成的细节能够受真实采样点控制,产生了与真实地表特点相近的细节,避免了细节的重复现象。此外,算法采用面向 GPU 实现的设计,将网格细分和细节生成放到 GPU 上进行,保证了整个算法的实时执行效率。

参考文献:

[1] Lindstrom P, Koller D, Ribarsky W, et al. Real-time, continuous level of detail rendering of height fields[C] //Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques. USA, ACM Press, 1996:109-118.

[2] Ulrich T. Rendering massive terrains using chunked level of detail control[C] //ACM SIGGRAPH 2002 Course Notes. USA, ACM Press, 2002.

[3] Larsen B D, Christensen N J. Real-time terrain ren-

dering using smooth hardware optimized level of detail[J]. Journal of WSCG, 2003, 11(2):282-289.

[4] Zhou H, Sun J, Turk G. Terrain synthesis from digital elevation models[J]. IEEE Transactions on Visualization and Computer Graphics, 2007, 13(4): 834-848.

[5] Schneider J, Boldte T, Westermann R. Real-time editing, synthesis, and rendering of infinite landscapes on GPUs[C] //Vision Modeling and Visualization 2006, Holland: IOS Press, 2006: 145-153.

[6] 李广鑫,吴自力,丁振国,等. 一种面向虚拟环境的真实感地形生成算法[J]. 西安电子科技大学学报:自然科学版, 2004, 31(5):728-731.
Li Guang-xin, Wu Zi-li, Ding Zhen-guo, et al. A modeling algorithm for generating terrain in virtual environment[J]. Journal of Xidian University (Nature Science Edition), 2004, 31(5):728-731.

[7] 孙轶红,焦永和. 基于视点和分形理论的真实感地形实时生成方法[J]. 系统仿真学报, 2004, 16(12):2729-2731.
Sun Yi-hong, Jiao Yong-he. Real-time generating of realistic terrain based on view and fractal[J]. Journal of System Simulation, 2004, 16(12): 2729-2731.

[8] Boubekeur T, Schlick C. A flexible kernel for adaptive mesh refinement on GPU[J]. Computer Graphics Forum, 2008, 27(1):102-114.

[9] Dachsbacher C, Stamminger M. Rendering procedural terrain by geometry image warping[C] //Proceedings of Eurographics Symposium on Rendering, 2004:103-110.

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

[11] Losasso F, Hoppe H. Terrain Rendering Using GPU-Based Geometry Clipmaps. GPU Gems2[M]. USA: Addison-Wesley, 2005:27-45.

[12] Catmull E, Clark J. Recursively generated B-spline surfaces on arbitrary topological surfaces[J]. Computer-Aided Design, 1978, 10(6): 350-355.

[13] 冯月萍,钟慧湘,张利永,等. 用C_C细分模式实现特殊效果模拟[J]. 吉林大学学报:工学版, 2005, 35(4):398-402.
Feng Yue-ping, Zhong Hui-xiang, Zhang Li-yong, et al. Generating special effect with C-C subdivision [J]. Journal of Jilin University (Engineering and Technology Edition), 2005, 35(4):398-402.

[14] Ebert D S, Musgrave F K, Peachey D, et al. Texturing & Modeling-A Procedural Approach[M]. USA: Morgan Kaufmann Publishers, 2003: 489-506.

[15] Perlin K. An image synthesizer[J]. ACM SIGGRAPH Computer Graphics, 1985, 19(3): 287-296.

[16] 廖昌阊,李辉,潘宏伟,等. 宏三角形的大规模地形漫游算法[J]. 电子科技大学学报, 2008, 37(1): 120-123.
Liao Chang-chang, Li Hui, Pan Hong-wei, et al. A marco-triangle-based algorithm for large terrain rendering[J]. Journal of University of Electronic Science and Technology of China, 2008, 37(1): 120-123.

[17] Bunnell M. Adaptive Tessellation of Subdivision Surfaces with Displacement Mapping. GPU Gems2 [M]. USA: Addison-Wesley, 2005:109-122.