

## 基于二叉树和 GPU 的无缝地形场景渲染方法

曹巍<sup>1\*</sup>, 段光耀<sup>2</sup>

(1. 中国科学院地理科学与资源研究所, 北京 100101; 2. 首都师范大学资源环境与旅游学院, 北京 100048)

(\* 通信作者电子邮箱 caowei@igsnrr.ac.cn)

**摘要:**设计了一种基于图形处理器(GPU)的无缝地形渲染方法。该方法基于二叉树构建多层次地形网格,该网格用基于行、列号的地形模板表示。在设计过程中,将高程数据转化为适于 GPU 读取的高程纹理图,再通过顶点纹理提取(VTF)技术从纹理图中采样出高程值用于渲染,整个过程在 GPU 端完成,提升了地形数据访问效率。同时,采用实时优化自适应网格(ROAM)算法的强制拆分法,通过控制相邻地形块的等级来消除裂缝。最后,采用 TriangleStrip 方式进行渲染,避免了相邻三角形中顶点坐标数据的重复传递,减少了传递到 GPU 的数据量。用两块地形数据对算法渲染效率进行了检验,并将算法与 Clipmap 算法进行了帧率对比。结果表明,该算法有效解决了分块数据的裂缝问题,达到了交互式地形渲染的要求。

**关键词:**地形渲染;二叉树;图形处理器;高程纹理;地形裂缝

**中图分类号:**TP391.411 **文献标志码:**A

### Seamless terrain rendering method based on binary tree and GPU

CAO Wei<sup>1\*</sup>, DUAN Guang-yao<sup>2</sup>

(1. Institute of Geographic Sciences and Natural Resources Research, Chinese Academy of Sciences, Beijing 100101, China;

2. College of Resource Environment and Tourism, Capital Normal University, Beijing 100048, China)

**Abstract:** In this paper a Graphic Processing Unit (GPU)-based and seamless method for terrain rendering was proposed. The method utilized binary tree to construct the terrain Level Of Detail (LOD) which was represented by row-column templates. In this method, the elevation data was transformed to elevation-texture and then the elevation-texture was used to extract elevation for rendering by Vertex Texture Fetch (VTF). And the whole process was handled by GPU so that the data accessing efficiency was greatly improved. The force-split method in Real-time Optimally Adapting Meshes (ROAM) was used in this method which solved terrain cracks by limiting the level of adjacent terrain blocks. The TriangleStrip primitive was used for terrain rendering which avoided transferring the same vertices' coordinates of adjacent triangles to GPU. Finally the paper tested the efficiency of the new method by two terrain datasets, and compared the frame rates between Clipmap and the new method. The results show that the new proposed method solves the terrain crack effectively and fulfills the demand of interactive terrain rendering.

**Key words:** terrain rendering; binary tree; Graphic Processing Unit (GPU); elevation-texture; terrain crack

## 0 引言

地形场景的实时渲染一直是计算机图形学的热点研究方向之一,它在虚拟现实、飞行模拟、交互式 3D 游戏以及地理信息系统(Geographic Information System, GIS)等多个领域有着广泛的应用。然而,随着地形场景规模的不断增大、分辨率的不断提高,用于表达真实感地形场景的三角形面片数量呈几何级数增长,渲染效率不断降低,这使得地形渲染技术面临严峻的挑战。为了解决该问题,不得不在渲染效率与渲染质量之间进行取舍,通过对复杂地形进行必要的简化来提高渲染帧率。然而随着图形处理器(Graphic Processing Unit GPU)性能的不提升,现代地形渲染算法不再只专注于地形简化,如何充分发挥 GPU 的特性成为关键<sup>[1]</sup>。

为了将地形简化与 GPU 特性更好的结合,本文重点探讨了以二分法构建多层次地形网格模板的方法,模板以顶点行列号替代地理坐标,通过对相邻模板等级的控制消除地形裂缝。同时算法利用 GPU 的可编程特性,利用顶点纹理提取

(Vertex Texture Fetch, VTF)技术从高程纹理图中提取出地形网格模板的真实三维信息,并按照三角形条带(TriangleStrip)方式进行渲染,实现大规模地形场景的三维可视化表达<sup>[2]</sup>。

## 1 研究现状

在早期的地形渲染技术研究中,地形简化是提高地形渲染效率的最有效途径,大量研究集中于此<sup>[3]</sup>,其中 Hoppe<sup>[4]</sup>提出了针对不规则格网数据的渐进网格(Progressive Mesh)算法,该算法通过顶点的合并与拆分实现模型的简化与复原,它可以表现任何复杂地势,但会消耗大量的内存空间;Lindstrom等<sup>[5]</sup>针对规则格网数据提出了连续层次细节(Continuous Level Of Detail, CLOD)算法,它是一种自底向上的算法,采用屏幕误差来控制渲染地块的层次细节(Level Of Detail, LOD)等级,并采用二叉树来维持顶点之间的依赖性以消除裂缝;Duchaineau等<sup>[6]</sup>提出了实时优化自适应网格(Real-time Optimally Adapting Meshes, ROAM)算法,其同样采用二叉树方式对地形场景进行细节等级划分,其执行效率与各帧之间

收稿日期:2012-04-09;修回日期:2012-06-10。 基金项目:国家自然科学基金资助项目(41001300)。

作者简介:曹巍(1982-),男,湖北武汉人,助理研究员,博士,主要研究方向:地形渲染算法、三维地理信息系统; 段光耀(1986-),男,河南叶县人,硕士研究生,主要研究方向:地理信息系统。

变化的三角形数量相关,当相机移动较慢且幅度不大的情况下效果较好;Rottger 等<sup>[7]</sup>提出了基于四叉树方式的地形等级划分方法,它将完整的地形网格通过四叉树管理,按照自顶向下的顺序,通过判断视距和地形块的表面粗糙度来决定渲染地块的 LOD 等级。上述算法的执行对 CPU 依赖较大,因此受 CPU 三维图形绘制能力的限制,地形渲染效率会有所影响。随着 GPU 的发展,计算机硬件的图形处理能力得到不断提升,为了充分发挥硬件的优势,基于地形分块渲染的方法被普遍采用<sup>[8-9]</sup>,其中 Boer<sup>[10]</sup>对传统四叉树算法进行了改进,提出了 GeoMipMap 算法,该算法对地形区域按照固定大小进行分块,并对每个分块按照纹理 MipMap 方式进行等级划分和渲染;Losasso 等<sup>[11]</sup>提出了一种新的用于地形绘制的 LOD 数据结构 Geometry Clipmap,该方法把地形缓存在一组嵌套的规则网格中,当视点移动时,网格中的数据也相应地不断更新;Ulrich<sup>[12]</sup>提出了 Chunk LOD 算法,该算法的主要特征是通过垂直的“裙带”来消除地形块之间的裂缝;Livny 等<sup>[13]</sup>提出了一种面向 GPU 的无缝算法,它首先将地形划分成正方形块,然后沿对角线将正方形地形块四等分,形成可以具有不同分辨率的四块三角网,三角网之间通过特殊的条带进行缝合。上述算法均采取了地形分块的策略,并且改变了早期逐三角形渲染的方式,而采用 GPU 的批量渲染方式,渲染效率大大提升,但在解决分块形成的地形裂缝时存在算法实现较复杂、裂缝消除效果不佳等问题<sup>[14-15]</sup>。

## 2 多层次无缝地形网格构建方案

### 2.1 地形分块与分级

LOD 是构建大规模三维地形场景普遍采用的方法,其利用不同精细程度的地形场景替代原始的海量地形,通过减少三角形数量提高三维可视化系统的运行效率。对于规则网格数据,二分法和四分法是地形 LOD 构建时最常用的两种方法,分别采用二叉树和四叉树结构进行分级数据的存储,但是四分法对地形的简化程度相对较高,其简化后的场景会出现相对较大的几何变形,因此本文将选择二分法进行多层次地形网格的构建。

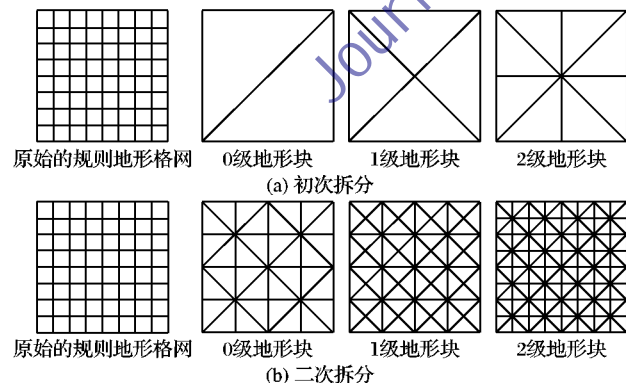


图1 地形场景 LOD 构建示意图

本文在构建地形场景 LOD 时,分两步进行:

- 1) 将原始地形进行二等分拆分,得到图 1 初次拆分的三角网,该操作主要用于确定地形块的等级。
- 2) 对各等级地形块进一步细分,形成与初次拆分对应的 0 级三角网、1 级三角网和 2 级三角网,细分同样采用二等分法,且需要满足两个条件:①同等级地形块细分后,内部三角形的大小必须相同;②各等级三角块细分后的三角形数量(以下用  $TriCount$  表示)必须一致,均满足  $TriCount = 2^n \times$

$2^n (n \geq 1)$ ,在图 1(b) 所示的二次拆分格网中,  $TriCount = 2^2 \times 2^2$ 。

依据上述方法,尽管各等级地形块内部三角形数量相同,但 LOD 等级高的地形块内部三角形更密集,其表达的地形越逼近于原始地形场景,等级低的则相反。 $TriCount$  可以根据需要进行调整,在地形起伏较小的情况下可以将其设置得更小,从而提高地形简化程度;反之可以设置更大,以便三角网中地形特征点被保留,从而降低可视化误差。地形场景的 LOD 等级数量由原始地形的网格大小以及  $TriCount$  决定,假设原始地形数据的格网大小为  $(2^M + 1) \times (2^M + 1)$ ,而  $TriCount = 2^n \times 2^n$ ,那么地形块最大划分等级为  $2 \times (M - n)$ 。

### 2.2 地形裂缝的消除

在实时渲染时,不同等级的地形块在它们的邻边处会产生裂缝,这是 LOD 方法存在的主要弊端之一,通常需要进行特殊的处理。本文针对 2.1 节提出的新的地形等级构建方案设计了相应的消除裂缝的方法。

新的地形等级构建方案最大的特点是各级地形块的直角边和斜边被等分,且不同的等级其等分的数量是相同的。针对这个特点,本文借鉴 ROAM 算法的强制拆分法来消除裂缝<sup>[3]</sup>。在具体实施时,强制拆分是针对地形块而并非最终的三角网,也即只针对地形等级构建时的第一步结果进行,其拆分原则具体如下:

- 1) 相邻地形块的等级之差不能超过 1,否则对低等级的进行强制拆分;
- 2) 当相邻地形块等级相同时,它们的连接关系必须满足直角边与直角边相邻,斜边与斜边相邻;
- 3) 当相邻地形块等级不同时(相差 1),它们的连接关系只能是高等级的斜边与低等级的直角边相邻。

图 2 为地形裂缝消除示意图。从图 2(a) 可看出地形块是无缝拼接的。图 2(b) 是以图(a)中右侧两相邻地形块为例载入内部三角网后的结果,灰边是相邻地形块的公共边,可以发现,该边处的三角网也是无缝拼接的。这主要是由于在构建内部三角网时,公共边被其两侧的三角网按照相同的数量进行了等分,两侧三角网在该边处没有出现点的错位,因此地形块在此处拼接时没有产生裂缝。当图 2(b) 中其他地形块载入内部三角网后会得到同样的结果,因此本文只用通过控制地形块的连接状态就能保证最终渲染的地形三角网是无缝的。

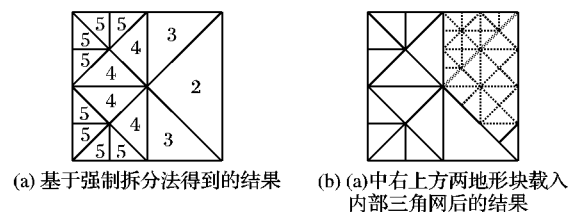


图2 地形裂缝消除示意图(数字为地形块的等级)

## 3 地形网格模板的构建

对于规则格网形式的地形数据,顶点的平面位置通常可以通过顶点行列号、格网左上角或左下角顶点坐标、顶点采样间隔计算得到,而顶点的高程值同样基于行列号可以从原始数据中方便的检索出来。针对该特性,本文以行列号代替顶点地理坐标进行分级操作,构建出不同等级的地形三角网,该方法得到的多级三角网只与原始地形网格模板的大小有关,用于多层次地形场景的渲染。具体方法如下:

1) 构建一个初始三角网模板,该模板记录 0 级三角网中所有顶点的行列号,由于同时存在两个 0 级三角网,本文选择其中之一的左上方 0 级三角网,如图 1 所示。

2) 依据 2.1 节的地形分级策略,各级三角网中顶点数量相同,排列位置和几何形状相似,因此对初始三角网模板(0 级三角网)中的顶点行列号进行旋转、缩放、平移等几何变换

操作后,可以分别形成 1 级、2 级、3 级、...、 $n$  级三角网的顶点行列号。

在进行几何变换时,需要考虑两种情况:同级三角网之间的转换和不同等级三角网之间的转换。同级三角网之间的转换只涉及旋转和平移操作,而不同等级三角网之间的转换还需要进行缩放操作。

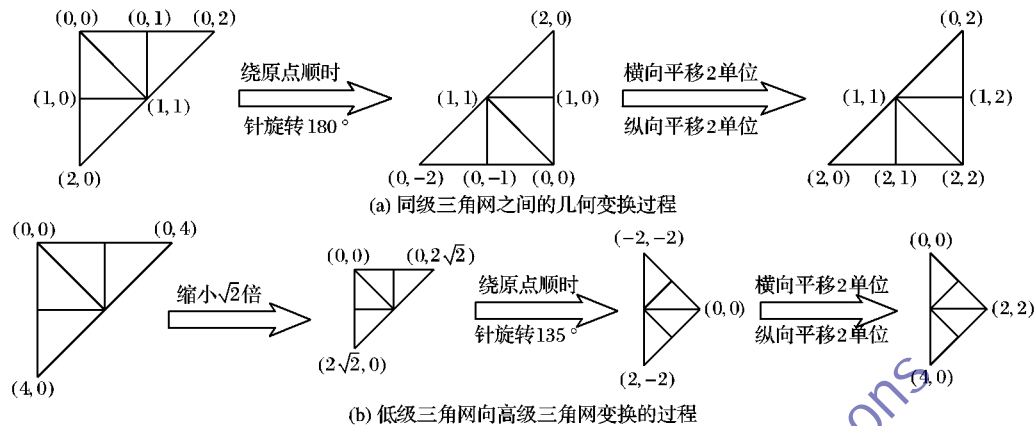


图 3 三角网模板之间的几何变换

基于地形网格模板的解决方案具有 3 方面的优点:

1) 该方法将地形网格顶点的平面位置属性与高程属性进行了分离,使得多层次地形网格分级结果能被具有相同顶点数量(顶点行、列数)的不同地形块共用,实现一次分级多次使用;

2) 该方法将复杂的 LOD 分级过程转化为简单的几何变换操作,使算法的实现简单化;

3) 该方法具有较好的扩展性,在不同的需求下,当需要改变地形网格中三角形数量或者顶点间距时,只需对原始模板进行调整,其他所有网格也将自动调整。

#### 4 渲染条带的构建

目前三维渲染的基本图元主要包含 6 种(如图 4 所示)。

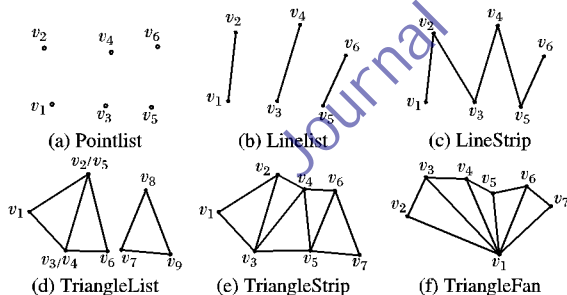


图 4 三维渲染 6 种常用基本图元

本文将选择 TriangleStrip 方式,主要有两方面的原因:

1) 地形格网的形态与 TriangleStrip 较为接近; 2) 采用 TriangleStrip 方式渲染,可以减少向 GPU 传递的数据量,从而大幅提高渲染的效率。因为按照传统的渲染方式,渲染一个三角形必须向 GPU 提交 3 个顶点,而在地形三角网中,顶点常被多个三角形共用,因此渲染一个完整的三角网会向 GPU 重复提交多个坐标相同的顶点,而每个顶点坐标至少由 3 个浮点数据组成,因此重复提交的数据量非常大。而 TriangleStrip 渲染方式对于相邻三角形的公共顶点不需要重复提交,以图 4(e) 中 7 个顶点组成的三角形条带为例,要绘制该三角形条带,只用向 GPU 提交 7 个顶点坐标,但是坐标的提交顺序必须遵循一定的规律,因为 GPU 在渲染时总是从提交的顶点中选取连续的三个顶点构建三角形,也即前一个

三角形的后两个点将参与下一个三角形构建,其坐标不需要重复提交,因此图 4(e) 中顶点的排列顺序为:  $v_1, v_2, v_3, v_4, v_5, v_6, v_7$ 。

本文三角网中顶点的连接顺序与图 4(e) 的条带存在一些差异,将按照图 5 中带箭头路径确定条带中三角形的连接顺序,确保三角网中所有三角形在条带中只出现一次。

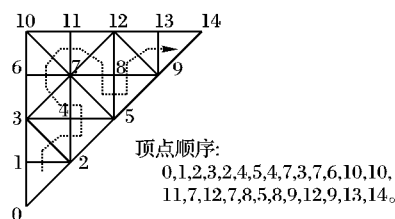


图 5 三角形渲染顺序示意图

在确定顶点的连接顺序时分两步进行:首先确定条带中三角形的连接顺序;然后确定各顶点的排列顺序。具体的操作流程如图 6~7 所示。图 6 中 *FirstTri* 表示条带中的第一个三角形, *ListTri* 为存储条带三角形的数组,存储顺序为条带中三角形的排列顺序。图 7 中 *CurrentTri* 表示当前进行判断的三角形, *NeighbourTri* 表示 *CurrentTri* 的相邻三角形, *ListTri* 同图 6 中对应变量, *Verts* 为最终存储的顶点数组,  $v_1, v_2, v_3$  表示 *Verts* 数组中的最后 3 个顶点。图 5 中的顶点顺序是针对图 5 中的三角网,按照图 6~7 的流程得到的结果。

#### 5 地形网格的渲染

本文方法中多层次地形网格的构建在数据预处理阶段完成,数据渲染阶段只需提取合适等级的三角网渲染即可,具体流程如图 8 所示,其中虚线部分完全在 GPU 端实现。

在地形可视性裁剪阶段,本文方法为地形块构造三棱体包围盒与视景体作相交检测,只有与视景体相交或包含其中的地形块才是有效地形块,并进入后续的操作过程。相交检测依照自底向上的方式进行,先检测父节点然后检测子节点,当父节点包含于视景体内时,其所有子节点也包含于视景体内,子节点不需重复检测;当父节点在视景体外部时,所有子节点也在视景体外,子节点也不需重复检测;当父节点与视景体相交时,所有子节点需逐一检测。



在地形 LOD 筛选阶段,本文方法采用了一个简化的判定准则,通过判断视点距地形块中心点的距离与地形块面积的比值确定地形等级,如式(1)所示。该方法避免了复杂的地形等级判定过程,使算法容易实现,同时缩短了运算时长。

$$\frac{L}{Area} < ErrorMetric \quad (1)$$

其中: $L$ 表示视点离地形块中心的距离(视距), $Area$ 表示地形块的水平投影面积, $ErrorMetric$ 为预设的 LOD 拆分阈值。拆分阈值增大时,在同样的视距条件下,只有 $Area$ 变小不等式才能成立, $Area$ 变小意味着地形块被拆分,地形网格变精细;反之,拆分阈值减小时,在同样的视距条件下, $Area$ 变大不等式才能成立,表明需要降低地形网格分辨率。因此通过调整该阈值可以控制地形网格的 LOD 等级。

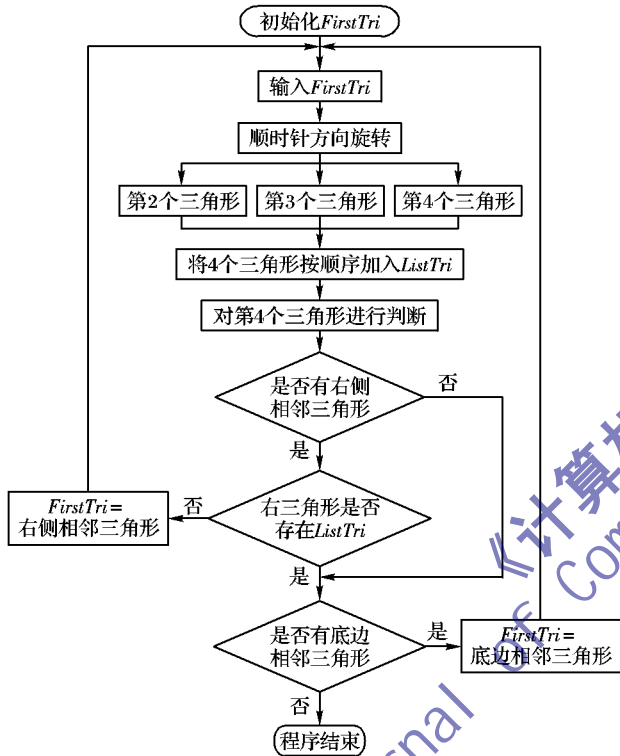


图6 确定三角形条带中三角形排列顺序的流程

在 GPU 纹理采样阶段,为了发挥现代图形显卡的优势,本文方法对传统的 DEM 数据进行了格式转换,将其转换为高程纹理图(该纹理并非用于地形纹理映射,纹理映射所需纹理来源于遥感影像或航片),以便于 GPU 快速访问。然后在顶点着色器中(Vertex Shader),采用 VTF 技术进行 GPU 编程,实现从高程纹理图中提取真实高程值。在提取高程值时,第3章地形网格模板中的行列号将用于推算高程纹理图中的纹理坐标,以便从正确的位置采样出高程值。以下为运行于 GPU 端的程序代码,用于根据顶点行列号提取真实三维坐标以及顶点对应的纹理坐标,代码中 Pos. x 和 Pos. z 即为地形网格模板中的顶点行列号。与传统的方法相比,基于 GPU 编程的方式实现更容易,并且数据存储与计算主要集中在 GPU 端,能够大大节省 CPU 的资源消耗。

```

VS_OUTPUT Transform ( float4 Pos; POSITION, float2 TexCoord;
TEXCOORD0)
{
    VS_OUTPUT Out = (VS_OUTPUT)0;
    float u = Pos. x/DataSize;
    float v = Pos. z/DataSize;
    Out. Pos. x = TopLeftX + Pos. x * deltaX;

```

```

    Out. Pos. y = tex2Dlod( sampler, float4(u,v,0,0)).
    r * 255 * ScaleZ;
    Out. Pos. z = TopLeftY + Pos. z * deltaY;
    Out. Pos. w = 1;
    Out. Pos = mul( Out. Pos, WorldViewLProj);
    u = Pos. x/DataSize;
    v = Pos. z/DataSize;
    Out. TexCoord = float2(u,v);
    return Out;
}

```

最后,根据数据预处理阶段生成的三角网顶点索引顺序、VTF 提取到的三维真实坐标, GPU 按照 TriangleStrip 方式构建出三角网并完成渲染。

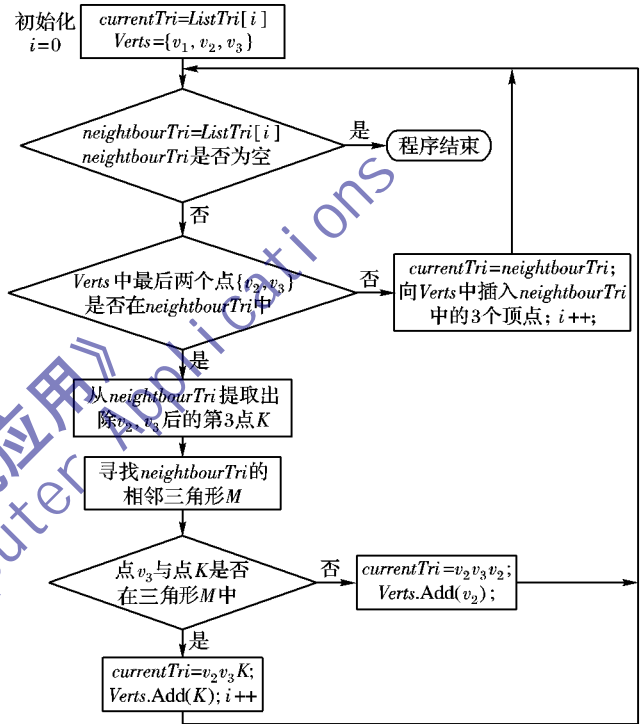


图7 确定三角形条带中顶点排列顺序的流程

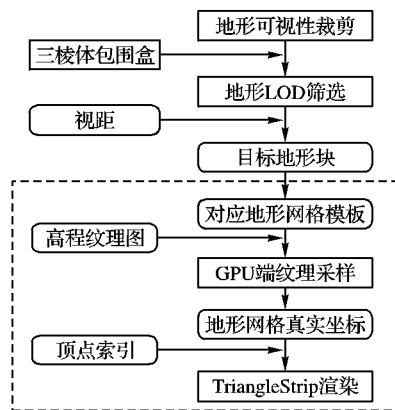


图8 地形网格渲染流程

## 6 实验与分析

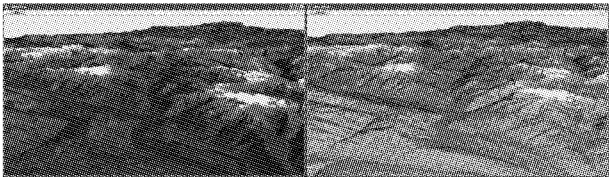
本文选择加拿大巴格布省立公园地区和四川省北川地区两块区域进行算法实验,硬件环境为 Intel Core i5 3.2 GHz CPU, 1 GB 内存, NVIDIA GeForce 310 显卡, 512 MB 显存。算法采用 DirectX 9.0c 三维图形接口实现,着色器语言为 HLSL (High Level Shader Language)。实验过程中对 2 块地形的 LOD 阈值进行了适当调整,以此控制三角网的 LOD 等级,具体数据如表 1 所示。可以发现,随着阈值的增大,LOD 精细度

提高,渲染三角形的数量增加,帧速降低,但最低帧速足以满足交互式操作的需要。在 LOD 阈值较低时,场景中地形三角

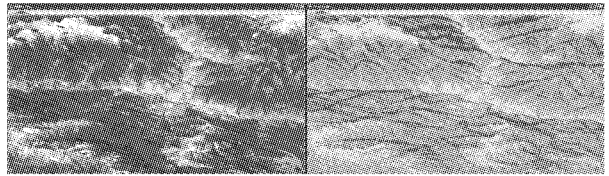
网的等级也较低,但渲染结果的几何变形较小(如图 9 所示,从图中看出场景中没有地形裂缝产生)。

表 1 不同地形数据的渲染结果表

区域名称	原始格网顶点数量	TriCount	LOD 阈值	渲染地块数量	帧速/fps
巴格布省立公园	2 048 × 2 048	128 × 128	5	20	59.98
			6	27	46.14
			12	38	35.24
四川省北川地区	4 096 × 4 096	128 × 128	6	15	59.98
			15	19	44.03
			30	28	35.48



(a) 巴格布省立公园地区渲染结果



(b) 四川省北川地区渲染结果

图 9 算法按照最小 LOD 阈值渲染的结果

最后,本文将同样基于 GPU 的 Clipmap 算法与本文算法进行了渲染效率的对比,将两种算法在同一地形数据(巴格布省立公园地区)同一漫游路径不同时刻的渲染帧率进行了比较,结果见表 2。可以发现,本文算法渲染帧率高于 Clipmap 算法,其主要原因在于本文对不同 LOD 地形场景的筛选、裂缝的处理等过程进行了简化,同时将地形分级模板、网格条带的构建安排在数据预处理阶段完成,有效减轻了实时渲染的工作量。

表 2 Clipmap 算法与本文算法渲染帧率对比

检测序号	X/m	Y/m	Z/m	帧率/fps	
				本文算法	Clipmap 算法
1	570 696	5 568 045	1 386	60.04	43.60
2	568 915	5 569 755	1 386	56.16	36.86
3	567 290	5 571 315	1 386	51.18	40.35
4	565 759	5 572 785	1 386	47.53	34.48
5	564 227	5 574 255	1 386	47.53	35.89
6	562 790	5 575 635	1 386	45.28	34.45
7	561 321	5 577 045	1 386	46.26	36.42
8	559 821	5 578 485	1 386	47.29	26.60
9	558 290	5 579 955	1 386	48.23	31.98
10	556 821	5 581 365	1 386	46.31	34.92

上述实验结果表明,本文算法在渲染效率方面达到了交互式地形渲染的需求,且场景中无地形裂缝产生,适于地形场景的实时渲染。

7 结语

本文设计了一种基于 GPU 编程的无缝地形渲染算法,通过该算法对两块不同大小的地形场景进行了渲染,并将渲染结果与 Clipmap 算法进行了对比,结果表明本文算法在保证较高渲染帧率的同时解决了地形裂缝问题,取得了较好的渲染效果,因此适于大规模地形场景的实时渲染。随着 GPU 性

能的进一步提升,尤其是 CPU 与 GPU 之间带宽的提高,以及 GPU 可编程能力的逐渐强大,发展基于 GPU 的地形渲染算法将是提高地形渲染效率的最佳选择。

参考文献:

[1] DICK C, KRUGER J, WESTERMANN R. GPU ray - casting for scalable terrain rendering[C]// Proceedings of the 30th Annual Conference of the European Association for Computer Graphics. Munich, Germany: [ s. n. ], 2009: 43 - 50.

[2] PAJAROLA R. Large scale terrain visualization using the restricted quadtree triangulation[C]// VIS '98: Proceedings of the Conference on Visualization. Washington, DC: IEEE Computer Society, 1998: 19 - 26.

[3] STRUGAR F. Continuous distance-dependent level of detail for rendering heightmaps[J]. Journal of Graphics, GPU, and Game Tools, 2009, 14(4): 57 - 74.

[4] HOPPE H. Progressive meshes[C]// Proceedings of ACM SIGGRAPH 1996. New York: ACM Press, 1996: 99 - 108.

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

[6] DUCHAINEAU M, WOLINSKY M, SIGETI D E, et al. ROAMing terrain: Real-time optimally adapting meshes[C]// Proceedings of Visualization 1997. Piscataway, NJ: IEEE Press, 1997: 81 - 88.

[7] ROTTGER S, HEIDRICH W, SLUSALLEK P, et al. Real - time generation of continuous levels of detail for height fields[EB/OL]. [2010-10-10]. <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.116.8235>.

[8] 王春,马纯永,陈戈.基于 GPGPU 的海量山地地形数据的实时绘制算法[J].计算机应用,2009,29(8):2105 - 2108.

[9] 王冬,张豫南,林成地,等.基于纹理数组的大规模地形绘制算法[J].计算机应用,2010,30(7):1832 - 1834.

[10] de BOER W H. Fast terrain rendering using geometrical MipMapping [EB/OL]. [2009-06-08]. [http://www.flipcode.com/archives/article\\_geomipmaps.pdf](http://www.flipcode.com/archives/article_geomipmaps.pdf).

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

[12] ULRICH T. Rendering massive terrains using chunked level of detail control[J]. SIGGRAPH Course Notes, 2002, 3(5).

[13] LIVNY Y, KOGAN Z, EL-SANA J. Seamless patches for GPU-based terrain rendering[J]. The Visual Computer: International Journal of Computer Graphics, 2009, 25(3): 197 - 208.

[14] PAJAROLA R, GOBBETTI E. Survey on semi-regular multiresolution models for interactive terrain rendering[J]. Visual Computer, 2007, 23(8): 583 - 605.

[15] 张小虎,邵永社,叶勤.基于自适应四叉树的地形 LOD 算法[J].计算机应用,2009,29(8):2596 - 2598.