

引用格式:汤延辰,郭星,张功营,等. 一种多控制因子 LOD 大规模地形绘制算法 [J]. 微电子学与计算机,2019,36(4):99-104. [TANG Y C, GUO X, ZHANG G Y, et al. A LOD large-scale terrain rendering algorithm with multiple control factors [J]. Microelectronics & Computer, 2019, 36(4): 99-104.]

一种多控制因子 LOD 大规模地形绘制算法

汤延辰¹, 郭 星¹, 张功营², 丁 全³, 姚 振⁴

(1 安徽大学 计算机科学与技术学院, 安徽 合肥 230601; 2 国网安徽省电力有限公司
宿州供电公司, 安徽 宿州 234000; 3 国网安徽省电力有限公司 电力科学研究院,
安徽 合肥 230601; 4 国网安徽省电力有限公司 信息通信分公司, 安徽 合肥 230061)

摘 要: 提出一种基于动态 LOD(level-of-detail, 层次细节)的多控制因子三维地形渲染算法, 该算法改进传统 LOD 地形绘制算法, 提出一种多控制因子算法来构建 LOD 模型. 首先依据视距以及表面粗糙度, 以及视点速度构建节点评价系统, 同时利用四点视域剔除方法以及运用加边法来进行裂缝消除. 实验结果表明, 该算法相对传统四叉树算法, 有效减少无用顶点的绘制, 显著地提高渲染效率.

关键词: LOD; 多控制因子; 视域剔除; 裂缝消除

中图分类号: TP391.9

文献标识码: A

文章编号: 1000—7180(2019)04—0099—06

A LOD Large-Scale Terrain Rendering Algorithm with Multiple Control Factors

TANG Yan-chen¹, GUO Xing¹, ZHANG Gong-ying²,
DING Quan³, YAO Zheng⁴

(1 School of Computer Science and Technology, Anhui University, Hefei 230601, China;
2 Suzhou power supply company, State grid Anhui electric power co. LTD, Suzhou 234000, China;
3 Power science research institute of state grid Anhui electric power co. LTD, Hefei 230601, China;
4 Information communication branch, State grid Anhui electric power co. LTD, Hefei 230061, China)

Abstract: This paper proposes a kind of based on dynamic LOD (level-of-the detail) multi-control factors of the three-dimensional terrain rendering algorithm, this algorithm improved the traditional LOD terrain rendering algorithm. Firstly, the nodal evaluation system is constructed according to the horizon, surface roughness, and the point of view velocity, at the same time, the four-point view elimination method and the edge addition method are used to eliminate cracks. The experimental results show that compared with the traditional quadtree algorithm, this algorithm can effectively reduce the rendering of useless vertices and significantly improve the rendering efficiency.

Key words: LOD; multi-control factors; view elimination; eliminate cracks

1 引言

虚拟现实技术是一门在二十世纪末发展起来的多交互系统仿真技术^[1]. 随着近年来计算机图形

学技术的进步, 虚拟现实技术愈来愈应用于军事领域^[2], 工业领域^[3], 影视娱乐领域等多方面领域. 然而在面对数据量庞大的大规模地形实时绘制时, 当前计算机的计算能力并不能满足将高程图所对应的

每个点的数据全部绘制出来的要求,为此需要通过软件的方法来减少地形的图形复杂度(即减少所需绘制的三角形的面数)。

LOD 技术是指物体细节随着该物体与观察者的距离的增加而降低,该算法能通过减少构成物体的多边形面数从而显著提升渲染速度^[4]。在现实世界中人们观察物体时,遵循近大远小的规律,越远处的物体其细节精细程度应当越低,当物体距离人眼达到一定距离时,其在人眼中呈现出的印象可能仅仅是一个点,此时若是再用大量三角形去精确描绘该物体的纹理细节显然是不合理且没有必要的,对于远距离的物体仅仅使用一个点来进行描绘与使用大量三角形进行细节描绘对于视点处的观测来说没有任何区别。

LOD 算法分为静态 LOD^[5]和动态 LOD^[6],静态 LOD 为预先构造模型的多级细节层次表示,在绘制时根据所需的细节表示进行调用,但由于存放了多个细节层次表示,因此占据了大量的存储空间。动态 LOD 根据视点的变化实时动态生成细节层次模型,但生成模型带来的时间开销巨大,极大地影响渲染效率。

本文采用在动态 LOD 的基础上进行改进,由于传统动态 LOD 算法生成地形模型效率低下的主要原因在于对视野之外的无需渲染地形节点依然进行渲染,本文通过模拟人眼的视角,在视觉显示效果不变的前提下,结合对传统 LOD 算法的视点距离以及地形粗糙程度因素的基础上,进一步考虑到传统 LOD 算法没有考虑到的移动速度因素不足因素,再使用视域剔除对多余节点进行消隐,实现对动态 LOD 算法的改进,提出一种基于多控制因子的 LOD 算法。通过实验表明,该算法可以有效的优化简化三维场景,通过不渲染大部分视野之外的地形,从而减少计算机对于三角形面数绘制的开销,在保证场景显示效果不变的前提下,显著提升运算的速度,同时该算法通过视野漫游速度的快慢来对场景进行模糊化,使场景渲染相对传统算法具有更好的真实性。

2 基于多控制因子的 LOD 算法介绍

2.1 算法描述

当加载数据量庞大的高程图数据时,由于构图点数过多,三角形数量庞大,地形结构复杂等问题导致当前计算机计算能力无法满足将高程图所有数据渲染的要求。为了降低计算开销,本文提出一种利用四叉树结构的多控制因子 LOD 算法,该算法可以

有效地减少图形复杂度,从而降低计算机运算负荷。本文算法中对于构建节点评价系统的影响因子如下:(1)距离视点越近的地形需要越多的三角形进行渲染,距离视点越远的地形需要越少的三角形进行渲染。(2)地形的起伏程度越大(即地形越复杂)则需要越多的三角形进行渲染。(3)视点的移动速度越快,则场景越模糊,意味着视点移动速度越快,则场景需要越少的三角形进行渲染。

本文使用的节点数据结构可用以下 C 语言结构来进行描述:

```
StructQuadNode //四叉树节点结构
{
    Vector3m_vCenter; //节点的中心点坐标
    Int m_nSize; //节点的宽度
    Intm_nLevel; //节点的细分程度
    Bool m_nHidden; //节点是否被消隐
    Bool m_nSegement; //节点是否需要细分
}
```

本文算法在建立节点评价系统的基础上,通过建立四点视域剔除方法来对视域之外的节点进行消隐,消隐之后的地形节点不再被渲染,随后进行裂缝检测并使用加点法对裂缝进行消除。

算法的流程图如图 1 所示,通过判定节点是否在视野范围以内来进行节点裁剪或根据多控制因子进行细分处理,最后进行裂缝消除来完善地形。

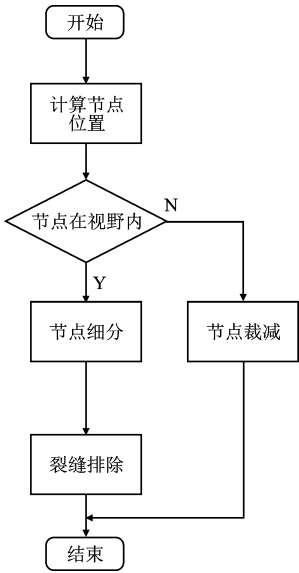


图 1 算法流程图

2.2 多控制因子描述

2.2.1 视点距离

物体离视点越远则物体越模糊,为渲染物体所需的三角形数目越少。但如果仅仅以物体距离视点

的距离作为划分标准的话,倘若在某个节点距离视点很远但该节点边长过大,对该节点不做细分的话同样会造成失真现象,或是某个节点距离视点距离很近且对细节描述已经足够,则无需对该节点再进行细分操作以节约资源.基于以上分析,因此,LOD 算法将视点距离作为一个重要影响因素.

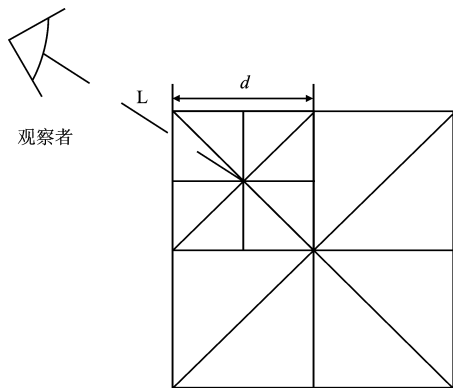


图2 视点距离计算原理图

如图2所示, L 为视点到某节点中心点的距离, d 为节点的边长,本文通过定义控制因子 C_1 来控制视点距离对于细节程度的影响关系.

$$f = \frac{L}{d \cdot C_1} \quad (1)$$

公式(1)表示根据视点距离的分割节点条件.从上式可以看出,当视点距离节点越近,即 L 值越小,节点的边长越大,即 d 值越大,则节点越可能被细分,控制因子 C_1 的值为常量,可根据需要修改,当 C_1 越大,则地形的细分程度越高.同时设置控制阈值 τ_1 ,当 $f < \tau_1$ 时,则对节点进行细分,即索引到该节点的子节点,直到 $f \geq \tau_1$.

公式(1)中的视距 L 在传统算法中可以直接计算视点 (x_0, y_0, z_0) 与目标点 (x_1, y_1, z_1) 的距离,即:

$$L = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2 + (z_1 - z_0)^2} \quad (2)$$

因为上述计算过程中涉及开方运算,而在大规模地形计算中节点数量庞大,因此大量的开方运算会严重导致算法效率降低.本文将采用如下方法来计算视距:

$$L = |x_1 - x_0| + |y_1 - y_0| + |z_1 - z_0| \quad (3)$$

上述计算过程不涉及开方运算,可以快速计算视距,增加运算效率.

2.2.2 表面粗糙度

在2.2.1节中介绍了视点距离对LOD算法的影响,但仅仅考虑视点距离这一因素显然不够的.当

面对平坦地形时,仅仅需要一个三角形面即可完整表示,但当面对崎岖地形,在小范围内地形的变化幅度较大时,用较少的三角形面便不能完整的表现出地形的全貌.因此,本文算法在考虑视点距离的基础上,综合考虑地形表面粗糙度对LOD算法的影响,当一个节点的崎岖程度较大时,即使该节点距离视点很远,依然需要将其进行细分.

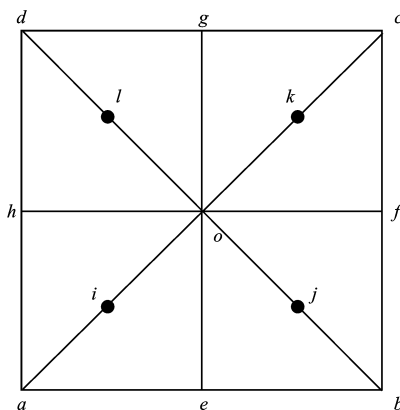


图3 地形复杂度计算原理图

图3表示一个节点示意图,对节点所表示地形建立包围盒,点 a, b, c, d 为地形节点的四个顶点, o 点为节点的中心点, e 点为与 ab 中点的地形坐标点,同理 f, g, h, i, j, k, l 分别为与 $bc, cd, ad, ao, bo, co, do$ 中点的地形坐标点.

因为在平坦地形上无需细分,而仅仅在崎岖地形上才需对节点进行细分,因此本文通过上述点来进行地形复杂度的评判.设 H_1, H_2, H_3, H_4 为各节点边的中点的高度值与节点对应边的端点的高度值的平均值之差,如 H_1 为 e 点高度值与 a 点高度值和 b 点高度值的平均值之差; H_5, H_6, H_7, H_8 为各节点端点到中心点的中点的高度值与对应节点端点与中心点高度值的平均值之差,如 H_5 为 i 点高度值与 a 点高度值和 o 点高度值平均值之差.利用公式(4)可以度量地形的表面粗糙度.

$$f = C_2 \sum_{i=1}^8 H_i \quad (4)$$

公式(4)表示根据地形粗糙程度的节点分割条件.设置控制阈值 τ_2 ,当 $f > \tau_2$ 时,则对节点进行细分,即索引到该节点的子节点,直到 $f \leq \tau_2$. C_2 为常量,表示地形粗糙程度的控制因子.本文算法通过截取8个中心点(即 e, f, g, h, i, j, k, l 点),通过中心点 Y 坐标值减去两端 Y 坐标值来判断地形的粗糙程度,当结果越大时则地形越复杂.可以看出,当地形越复杂时,则细分程度将会越高,控制因子 C_2 的值可根据需

要修改,当 C_2 越大,则地形的细分程度越高.

2.2.3 视点移动速度

一般的 LOD 算法仅仅考虑视点距离以及地形粗糙程度这两个因素,且认为通过这两个因素调节 C_1 和 C_2 的值便能够使渲染效果和渲染速度达到相对稳定状态. 本文在此基础上增加了的控制因子,得到了出色的效果,在研究过程中发现,当人体进行快速运动时,周围物体的轮廓也会变得模糊,且速度越快时,物体变得越模糊,因此将视点移动速度加入控制因子可以很明显的提高渲染的真实效果.

因此,本文将视点移动速度作为本文算法的一个重要控制因子加入,由于视点移动速度相对前个因素重要程度较低,因此本文将其放在最低的优先级,且仅当视点处于漫游状态时,该因素才发挥作用,当视点处于静止状态时,该因素不起作用. 在加入视点移动速度作为控制因子后,当视点移动速度越快,则场景越模糊(即场景细分程度越低),视点移动速度越慢,则场景越清晰(即场景细分程度越高).

由于本文算法需要根据物体的实时速度来调节地形的细分程度,因此在算法中必须计算出物体的瞬时速度. 由于计算速度需要涉及位移因素,在传统计算过程中需要涉及开方处理,影响运算速度,因此,本文采用一种新的计算方法来计算出速度的同比值,计算公式为:

$$v = \max((|x_1 - x_2| + |y_1 - y_2| + |z_1 - z_2|)/t, 1) \tag{5}$$

式中, v 代表视点的瞬时速度; x_1, y_1, z_1 分别代表在当前时刻视点在三维世界坐标系下 x, y, z 坐标; x_2, y_2, z_2 分别代表上一帧时刻视点在三维世界坐标系下的 x, y, z 坐标; t 代表当前时刻与上一帧时刻的时间间隔. 若速度过慢,则直接将速度设为 1,使速度因素不影响节点的细分.

2.3 综合节点评价系统

本文算法在传统 LOD 技术考虑视点距离与地形粗糙程度的基础上进行改良,并增加了视点移动速度因素,让本文的渲染算法更加符合真实人眼视野. 在综合考虑视点距离,地形粗糙程度和视点移动速度三个因素之后,本文提出如下 LOD 评价函数 f :

$$f = \frac{L \cdot \tau}{d \cdot v \cdot C_1 \cdot C_2 \cdot \sum_{i=1}^8 H_i} \tag{6}$$

式中, $\tau = \tau_2 / \tau_1$,若评价函数 $f < 1$,则对该节点进行

细分操作,将该节点等分为四个子节点,否则该节点不再细分,直接对该节点进行渲染. 从公式(6) 中可以看出,评价函数 f 与视点距离节点距离 L 呈正比关系,与节点边长 d ,地形表面粗糙度,以及视点移动速度 v 呈反比关系,即当节点距离视点越近,移动速度越慢,节点边长越大,地形粗糙程度越大时,该节点越被细分,反之亦然.

上述结论与本文在先前分析相同,在实际应用中可通过调节控制因子 C_1 与 C_2 来调节物体的细分程度,使其达到完美的效果. 本算法充分考虑各个因素对地形细分的影响,极大程度上还原人眼真实的渲染效果.

2.4 视域剔除

在面向大规模地形渲染的过程中,绝大部分的节点都是在视锥体^[7] 之外,将无效节点进行剔除消隐对于大规模地形的漫游十分重要,可以极大地减少无用节点的绘制,进而减少显卡的负担.

本文算法采用四点视域剔除法进行对无用节点的消隐,仅判断节点的四个端点是否在视锥体内,四个端点只要发现有一个端点在视锥体以内,便对节点进行保留,只有当所有端点都不在视锥体以内,才对该地形节点进行消隐处理.

算法产生的效果如图 4 所示. 图 4 中 V 为当前视点,箭头方向为视野朝向,深色节点为算法所需渲染节点,虚线内部为视野大小,虚线以外的节点在当前视点不可见. 图 5(a)为未经过视域剔除所需渲染的节点,图 5(b)为本文算法所需渲染节点,本文算法通过对用户不关心的不可见节点进行裁剪,从而大幅度提高渲染效率.

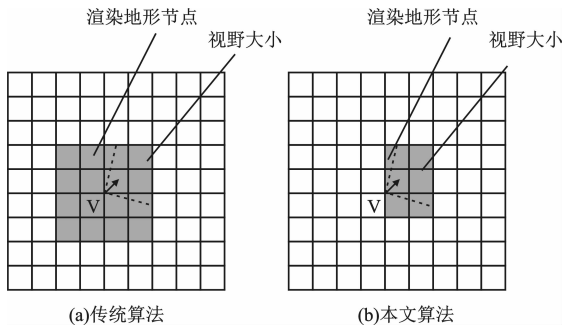


图 4 渲染效果

本文算法通过视域剔除可以将大量无用节点不进行细分而直接剔除,从而极大地减少计算开销,大幅度增加画面流畅度,且对于用户来说并没有感到画面产生任何变化.

3 裂缝消除

在地形渲染时,如果相邻两个节点的细分程度不同,则可能会在两个节点的连接处产生三角形裂缝,非常影响渲染质量.常用的裂缝消除方法有加边法和减边法,其中减边法的实现策略较简单,容易实现,但会降低地形精度,且不能实现相邻节点细分程度大于1的情况的裂缝消除.因此,本文选择加边法进行裂缝消除.

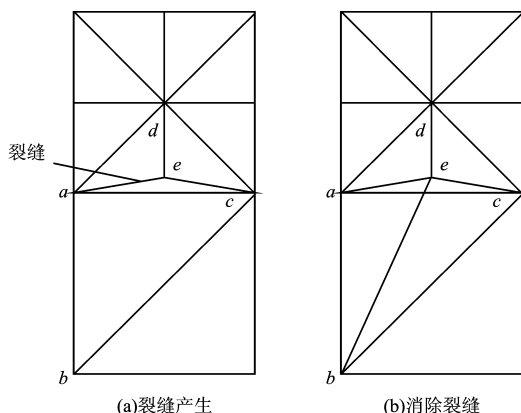


图5 裂缝示意图

图5(a)中为一个地形节点的俯视图,三角形面 abc 属于细分程度为1的节点,三角形面 ade 与 cde 属于细分程度为2的节点,且三角形面 ade 与 cde 分属两个不同的子节点.图5(a)中的阴影部分就是裂缝.

裂缝产生的原因可由公式(7)表述:

$$H(e) \neq H(a) + H(c)/2 \quad (7)$$

裂缝处理方法如图5(b)所示,通过将细分程度较低的节点增加边 be 来平衡高度差,使块间裂缝消除.

对每一个地形节点,可能产生上、下、左、右四个方向的地形裂缝,在对进行渲染前,通过当前节点的每个方向进行节点细分程度判断,如果发现两个节点的细分程度不同,则采用加边法来消除裂缝.

4 实验结果与分析

为验证本文算法的有效性和优越性,本文选取某地区高程图7000个采样点,最小划分为 16×16 网格,实验采用C++语言和OpenGL API实现算法.

本文通过将本文算法与传统限制四叉树LOD算法进行对同一地形进行渲染,计算不同状态下地

形块的渲染数.实验的对比结果如图6和图7所示.

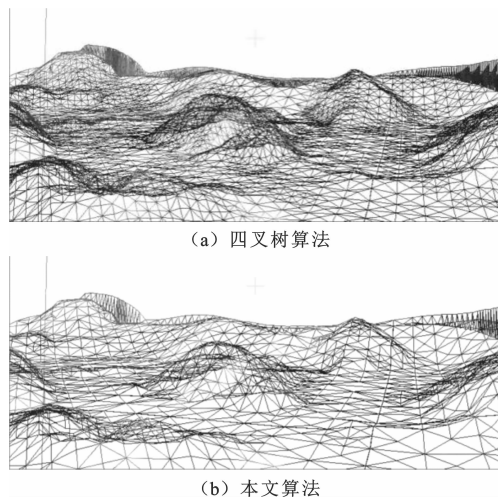


图6 网格生成结果

图6(a)为四叉树算法所生成的网格图,图6(b)为漫游状态下本文算法所生成的网格图,四叉树算法采用文献[8]提供的算法.从图中可知,距离视点越近,且地形越复杂的节点细分程度越高,而距离视点越远,地形越简单的节点细分程度越低,实验对四叉树算法和本文算法采用相同的 C_1 和 C_2 值,可以明显看出本文算法所生成地形三角形面数比四叉树算法生成的地形三角形面数数量要大幅度减少,有效减少计算机的运算压力.

图7为基于图6形成的网格图上贴上纹理图的渲染结果,图7(a)为四叉树算法的渲染图,图7(b)为本文算法的渲染图.

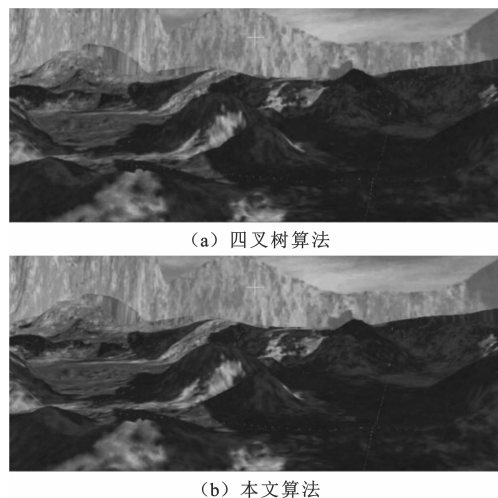


图7 最终渲染结果

从图7中可以明显看出本文算法所渲染出的地形与四叉树方法从用户视角来说区别不大,而三角形面数大量减少,渲染速度更快.

表 1 两种算法测试对比

实验参数	四叉树算法	本文算法
采样点数	7 000	7 000
原始三角形面数	15 284	14 796
静止状态三角形面数	5 552	3 264
漫游状态三角形面数	5 552	2 840
漫游状态帧数	79	220

从表 1 可以看出,本文相比四叉树传统地形渲染算法具有更好的渲染速度,尤其在漫游状态下,帧数比其它算法产生了极大突破. 本文由于考虑在传统四叉树算法的基础上对节点进行了视距裁剪,裁去了大量节点,送去 GPU 处理的三角形面数量极大的减少,减少了 GPU 的负担. 同时,本算法在漫游过程中加入了视点移动速度控制因子,使得视点漫游时的三角形面数量得到减少,增加漫游状态下的单帧绘制速度,极大地提高了用户体验的真实感. 本文算法在裂缝可预见性的基础上使用加边法对裂缝进行消除,使地形可以更良好地呈现给用户. 此外,本算法虽然减少了大量的三角形面,最终渲染效果与传统限制四叉树算法相比从用户角度来说并没有多少区别.

5 结束语

针对传统 LOD 算法在面对超大规模地形数据处理速度不够,在应用到地理研究,影视娱乐,网络游戏等领域时不能满足用户对于流畅性的需求,本文基于传统 LOD 算法的基础上,在保证用户对地形的视觉效果不变的基础下,增加了移动速度这个传统 LOD 没有考虑到的控制因子,同时对视点距离和地形粗糙程度两个控制因子进行优化,使用视域剔除消隐无用节点并使用加点法来消除地形裂缝,使构成地形的三角形面在其传送给显卡之前便大大减少其数量,极大的增加视点在静止状态和漫游状态下的渲染效率. 本文通过实验对比传统四叉树算法,证明本文算法能够显著减少地形三角形面数,尤其在漫游状态下,其单帧绘制时间更短,呈现

给用户的渲染效果也更加真实.

在未来的工作中,可以在以下两点展开研究:

- (1)现阶段使用加点法删除裂缝,然而加点法需要对整个地形进行遍历并重新进行三角形细分,增加了问题的复杂性,影响效率. 未来可研究无需遍历节点也能进行的裂缝消除法,进一步提高算法效率.
- (2)进一步研究其它视相关因素,通过更多的因素来更强地增加算法的效率和场景真实性.

参考文献:

[1] 王磊,毛利民,李骞. 基于 LOD 的三维地形可视化[J]. 计算机与信息技术,2007(7):38-39.

[2] 原庆红,韩燮. 基于视点的分块 LOD 大规模地形生成算法研究[J]. 微电子学与计算机, 2011, 28(6): 41-45.

[3] 朱丽洁,慕晓冬,刘鑫昌. 某导弹阵地的三维地形生成与实时显示[J]. 微电子学与计算机, 2006, 23(9): 73-75.

[4] 邓益民,汤智谦,李红兵,等. 基于 LOD 的大规模输电线路场景实时渲染算法优化[J]. 计算机与现代化, 2017(1):115-118.

[5] 夏杰,陈奇. 基于静态 LOD 的海量地形绘制中的接缝算法[J]. 计算机工程与设计, 2010, 31(22): 4847-4850.

[6] 魏子衿,肖丽. 基于区域分解的并行动态 LOD 构建算法[J]. 计算机工程与应用,2018(6):168-177.

[7] 于海燕,张帅,余沛文,等. 视锥体裁剪几何算法研究[J]. 图学学报,2017,38(1):1-4.

[8] 王倩,高保禄,高锐军,等. 基于四叉树包围球和屏幕误差的 LOD 算法[J]. 微电子学与计算机, 2016, 33(5):127-132.

作者简介:

汤延辰 男,(1994-),硕士研究生. 研究方向为三维可视化、虚拟现实技术. E-mail:tycl23cc@qq.com.

郭星 男,(1983-),博士,讲师. 研究方向为三维可视化、模式识别与信号处理.