Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

GPU-based real-time terrain rendering: Design and implementation

Rui Zhai*, Ke Lu, Weiguo Pan, Shuangfeng Dai

University of Chinese Academy of Science, Chinese Academy of Science, No.19A Yuquan Road, Beijing 100049, China

A R T I C L E I N F O

Article history:
Received 11 April 2014
Received in revised form
14 July 2014
Accepted 2 August 2014
Communicated by M. Wang
Available online 24 August 2015

Keywords:
Terrain rendering
GPU-based
Restricted quadtree
Tessellation

A B S T R A C T

Terrain rendering has been a hot spot for many years. How to improve the rendering efficient and get more smoothing terrain with massive data has become the focus of the terrain rendering research recently. In this paper, we present a real time rendering algorithm based on GPU (Graphics Processing Unit) and tessellation technology. In the preprocessing stage, we build two separate restricted quadtree, logical tree and data-tree. The logical tree is used to reduce the memory usage and logical information is operated on logical tree. Then we compute the nodes (patch) that are satisfied with LOD (Level of Detail) and view frustum culling using logical quadtree. This stage is used to minimize nodes passed to GPU to save GPU memory. To fully use the GPU computational capabilities, we adopt the tessellation technique in the triangulation stage instead of the traditional triangulation methods. With the tessellation technique, all the triangulation work is done by the hardware which saves much more rendering time, and improves the rendering speed highly. In addition, by adjusting the tessellation factor, the terrain crack is avoided easily. In the experiment, it proves that the method can highly reduce the processing time and get a feasible result.

1. Introduction

Real-time terrain rendering has been a hot spot in computer graphics for many years, which is widely used in many applications, such as virtual reality, geo-information system (GIS), flight simulation and landscape editor.

In past decades, with the increasing size of the digital elevation models (DEMs), many real-time terrain rendering algorithms have been built to improve rendering effect [1–3]. Early methods drew the primitives directly, which were very simply to code but inefficient. Obviously, this method cannot solve increasing massive data. To improve computing and rendering ability the computing ability of GPU and hardware have made rapid progress.

With the development of hardware and GPU, the focus has shifted from traditional mesh simplification method to hardware-based programming period. CPU based algorithm has shifted to GPU or CPU-GPU based algorithm, and the main focus of these algorithms are different from traditional algorithms. There are new topics or more options need to be in consideration: Firstly, take consider of the rendering characteristic computer graphics, traditional method need to do triangulations manually on the CPU, but now triangulation is a more flexible option, it can be done manually or by the software, or it can operate on CPU or GPU. Same to triangulation, LOD selection is familiar situation, if the data is small, the data can be passed to GPU totally to select appropriate LOD layer or culling, if the data is huge, these work can be done on the CPU to reduce the information passed to GPU. Lastly, although the GPU calculation ability has made much progress in the past few years, the algorithms of triangulate mesh and merge crack between different meshes is still a challenge.

By analyzing the characteristic of GPU rendering pipeline and quadtree based data structure, we propose a real-time massive terrain data rendering method in this paper and the experiment shows that this method is efficient. There are mainly two stages in the framework, CPU stage and GPU stage. In the CPU stage, the LOD selection and culling is operated to reduce nodes that passed to CPU. In the GPU stage, we abandon triangulation algorithm which is a major work in most of the terrain algorithm. The triangulation and crack avoidance is done by setting the tessellation factor in the GPU pipeline. The organization of this paper is as follows. The related work will be described in Section 2. In Section 3, we will give the real time terrain rendering framework using tessellation technology. The computing result will be shown in Section 4. Finally, the conclusion and future work of this paper will be given in Section 5.

2. Related work

Terrain rendering has been studied for many years. In this section, we will introduce some related real time rendering methods. Early view-dependent algorithms were based on CPU,

* Corresponding author. Tel.: +86 18810400469
E-mail address: zhairui11b@mails.ucas.ac.cn (R. Zhai).

http://dx.doi.org/10.1016/j.neucom.2014.08.108
0925-2312/© 2015 Elsevier B.V. All rights reserved.

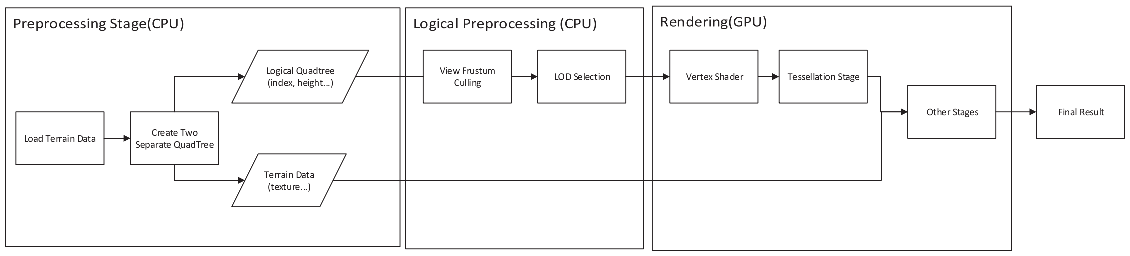

Fig. 1. Task scheduling and execution procedure.

according to the different data structures of these algorithms, these algorithms can be divided into two categories: Regular Grid based algorithm like Real-Time Optimally Adapting Mesh [1] (ROAM), and Triangulated Irregular Networks (TINs) based algorithm like Progressive Mesh [2].

TINs based algorithms use irregular triangulated meshes to represent terrain [4,5]. TINs based data structures store the mesh relationship, node connection, and information of each point. In the simplification stage, most traditional methods need adjacent triangles information, and the triangulation is pre-computed in a preprocessing stage. For example, De Berg et al. [6] use Delaunay triangulations to constraint the meshes. Wahl et al. [3] precompute TINs with object-space error bound, then use compression techniques to reduce memory and bandwidth requirements of the triangles. Recently, Hu et al. [7] proposed a new parallel view-dependent Level of Detail algorithm. This method assigns a value to each node to guide the node split and collapse algorithm, which make it possible to execute on GPU.

Another data structure to represent terrain is regular grids like restricted quadtree [8], bintrees [1,9,10]. Comparing to TINs, the structure of regular grids is simpler. In most case, the position or coordinate can be calculated by its index. It is easier to store and manipulate, and more GPU-friendly. Comparing to TINs, regular grid based methods are less efficient for general mesh simplification, but more suitable for terrain based rendering [11]. Although regular data structure is easy to handle, most traditional regular grids based methods, like continuous LOD quadtree [12], restricted quadtree triangulation [13], 4–8 meshes [14,15], and right-triangulated irregular networks [9] need complex preprocessing step to compute triangulation of the regular grid. Later, with the development of hardware technology and GPU programmability, more and more GPU-based algorithms have proposed, like BDAM [16], Geometry clipmaps [17]. Lossaso et al. [18] use the fragment processor to perform mesh subdivision. Comparing to traditional CPU based algorithms, rendering speed has made much more progress. Some researchers combine TINs and regular grid data structure to make a compromise between flexibility and management, in [19], the author adopt popular TINs based mesh simplification method [20] with regular raw data to find the sensitive node or patch to display feature sensitive terrain, this work can be done on GPU. In total, the main improvement of these algorithms are to design new data structure that more adapt to parallel management and new framework to improve the use of GPU memory and buffer. However, most of these algorithms need the preprocessing stage to compute triangulations and loading during rendering [21,22].

Traditional GPU pipeline contains vertex shader stage and fragment shader stage. Later, two most commonly graphics APIs: DirectX and OpenGL introduce geometry shader stage, which makes it possible to add and destroy points on the GPU. That makes tessellation can be used in rendering framework [23]. Before the tessellation is done by the rendering pipeline, many researchers have studied this problem [24–26]. Most of these works is done manually, these algorithms are complex and hard to implement with limited time. Tessellation is so widely used in 3D graphics rendering, DirectX and OpenGL both add tessellation stage in rendering pipeline to improve rendering speed and reduce coding complexity. Comparing to traditional methods, tessellation has many advantages. Firstly, tessellation supports scalable-rendering techniques, such as continuous LOD or view-dependent LOD which can be calculated on the fly. Secondly, tessellation technique in DirectX and OpenGL pipeline supports displacement mapping technology, which can save lots of memory and bandwidth, makes it possible to rendering more detailed information. Lastly, current tessellation processing primitive is patch, same as popular GPU-based LOD rendering method. This makes it easier to combining current LOD rendering method with tessellation technology.

From the above introduction, a newly terrain rendering method based on the tessellation technology is proposed, and better visual effects can be get with low time consumption. Details of the algorithm will be shown in the following sections.

3. Algorithm

In this paper, combined with the related newly tessellation technology, a no programmable triangulation method is proposed which will highly reduce the time cost in triangulation stage. The main structure of our algorithms contains the following parts.

3.1. *Preprocessing stage*

In the preprocessing stage, raw terrain data is converted to destination format. The initial terrain data is heightmap, which is a widely used format to store digital terrain information and used in terrain rendering, bump mapping and displace mapping. But the raw data is not suitable for current GPU based rendering method. In order to make full use of the GPU ability and parallel method, the basic data structure of our algorithms is based on restricted quadtree, which is optimized to avoid crack and T-junction in terrain rendering structure.

In our structure, a top-down mesh simplification method is adopted as shown in Fig. 2. The LOD level begins from $level_0$ to level_{n-1}, $level_0$ is the coarsest level, and level_{n-1} represents the most detailed resolution terrain. In a quadtree structure, it is easy to calculate the node index, if the index of node x is k, then the index of its four children nodes are $4(k+1)$, $4(k+1)+1$, $4(k+1)+2$, $4(k+1)+3$.

In the memory, two separate but related datasets, the logical data and non-logical data, are built to minimize the need of the GPU cache. These two structures share the same framework, but have different function.

Logical tree is the main tree in processing stage. Logical tree stores geometry information and logical information, it is used to guide selecting LOD level and nodes at run time. For each logical quadtree node, it contains pointer to sons, parent, height information, split status, pointer to its corresponding non-logical tree, and status information. Every node has three statuses: Split, Rendering