PYRAMID MANAGEMENT AND REAL-TIME VISUALIZATION OF MASSIVE 3-DIMENSIONAL TERRAIN DATA

KE Xi-lin[a, b], GUO Qing-sheng[a], LI He-yuan[b], CHEN Gang[b], GAO Ping[c], ZHOU Yan[b], HUANG Li-min[b]

[a]School of Resource and Environment Science, Wuhan University, Wuhan, China - chauchl@163.com,guoqingsheng@126.com
[b]Xi'an Research Institute of Surveying and Mapping, Xi'an, China
[c]Wuhan Ordnance n.c.o Academey, Wuhan, China - gaoping1220@163.com

KEY WORDS: Visualization, Level of Detail, Computer Vision, Simulation, Geography

ABSTRACT:

3-Dimensional terrain visualization has been widely used in the fields, such as games, geographic information system (GIS), remote sensing (RS) and virtual reality (VR). Rendering large scale 3D terrain interactively is now becoming a hot research topic for the fast growing application of 3D terrain visualization. There is a conflict between massive 3D terrain data and its fast rendering. On one hand the visualization of terrain scene is expected to be in the interactive frame rates without any delay, on the other hand the terrain scene is expected to be large enough to meet the need of understanding the environment without compromising visual quality. From the above mentioned, massive 3D terrain data and texture memory management become the performance bottleneck in the visualization of large scale terrain. The terrain rendering load must be controlled by reducing the number of rendered primitives using LoD(level-of-detail) techniques, adaptive extraction of LoD meshes, pyramid management of DEM(digital elevation model) and DOM(digital orthograph model) and real-time loading and rendering of massive 3-D terrain data. In this paper we will investigate and discuss the approaches, the pyramid management and rendering of massive 3-D terrain data in interactive frame rates.

1 INTRODUCTION

Interactive visualization of very large scale terrain data leads to several efficient problems. In the case of terrain visualization, the scene parts which can be seen are only small parts of the whole terrain. To improve the rendering performance, the terrain should be divided into several parts, according to their coordinate in the whole terrain. Through subdivision of the whole terrain, not the whole terrain but the visible terrain parts in the scene view are loaded into the view scene.

Apart from the subdivision of the whole terrain, the blocks which can be seen will not be in the same resolution for the different distance to the view point. So in the visualization of large scale terrain, scene clipping according to the field of view(FOV) and view-dependent simplification can improve the terrain rendering velocity and will not lose any rendering image quality.

The basic requirements are very important for massive 3-D terrain data management and visualization. They are as follows:
1) easy to manage the regular terrain with massive data
2) easy to search the appropriate terrain blocks in visualization
3) easy to load and unload the terrain data
4) easy to get the appropriate LoD terrain data of the same block
5) easy to render the massive terrain data in interactive frame rates with better precision

To meet the above-mentioned requirements in rendering the massive 3-D terrain data, pyramid management and dynamic loading of the terrain data is implemented. Using the pyramid management of the terrain data, multi-resolution terrain data, including DEM and DOM, are created based on the original data. On the foundation of the multi-resolution terrain data, continuous level-of-detail (CLoD) quadtree have been used to efficiently render the massive terrain data.

The paper is organized as follows. In section 2 we briefly review some related work on the large scale terrain rendering. Section 3 presents the detailed discussion of pyramid management techniques of terrain data and the dynamic loading and rendering methods of the terrain data, and section 4 discusses the programming test. The paper is concluded with a summary and outlook in section 5.

2 RELATED WORKS

In this section, a number of approaches developed over the past decades on real-time rendering of the massive terrain data are briefly introduced.

Lindstrom et al(1996) proposed the first real-time continuous level of detail terrain rendering algorithm. Lindstrom's algorithm is a block based on a bottom-up strategy. It first assesses the level of the block according to screen projection error metric, and it inserts those vertices which have a larger delta value than some threshold. To eliminate cracks between adjacent nodes with different levels, Lindstrom suggested adjacent blocks share vertices on their boundaries. The algorithm can exploit the coherence between frames to reduce the number of processed vertices per frame. It generates result meshes based on quadtree subdivision and the vertices chosen in the previous step. It proves to be an effective way to evaluate the importance of vertices in the original sampling grid.

In 1997, Duchaineauy et al can render his terrain with Real-time Optimally Adapting Meshes. Since ROAM has been widely used in games. ROAM learned much from Lindstrom's algorithm and it's much faster. ROAM performs triangle splitting and merging based on triangle tree structure, which is somehow similar to Lindstrom's binary vertex emphasized on bottom-up algorithm, ROAM uses a splitting diamond queue and a merging diamond queue to progressively updates in the well as other error metrics. The algorithm is error bound, it

guarantees the priority to be monotonous while doing a top-down subdivision. Besides its strict error control it can have an explicit control over triangle numbers and LOS(Line of Site). The algorithm exploits more of the coherence between frames and can maintain a rather stable frame rates.

The above-mentioned work is an overview of real-time terrain rendering algorithms. There are also some other algorithms, for example Rottger's (1998) real-time generation of continuous levels of detail for height fields, Hoppe's(1998) smooth view-dependent level-of-detail control to terrain rendering and Jonathan's(2000) terrain rendering at high levels of detail. These methods differ more or less from each other, but they have something in common, such as view frustum culling and LoD method and crack eliminating, in similar way.

3 PYRAMID HIERARCHY MODEL

3.1 Multi-resolution hierarchy model

Pyramid data model is created based on the pre-process of the terrain data and the image data, the strategy is building a multi-resolution data model, including the DEM and DOM data. The highest level is given level 0, with the smallest resolution in the data; the lower the level is, the higher the data resolution is. The resolution of data in lowest level is the resolution of the original data.

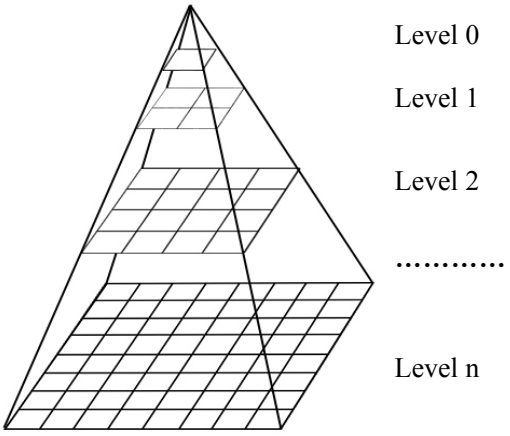

Figure 1. Pyramid Hierarchy Model

Provided that the horizontal range of the terrain is from dblMinCol to dblMaxCol and the vertical range of the terrain is from dblMinRow to dblMaxRow, the terrain root node number is M×N, the node size of the terrain is $(2^n+1)\times(2^n+1)$, generally 17×17 or 33×33. There is a common adjacent edge between the neighboring terrain nodes.

For texture DOM data, the size of the image data is compulsory set to be $2^k\times2^m$, generally 256×256 or 512×512.

If the current terrain level is l, then the terrain resolution at level l in horizontal and vertical can be computed by:

$$\begin{cases} RowSpace = \dfrac{(dblMaxRow - dblMinRow)}{M \times 2^l \times 2^n} \\ ColSpace = \dfrac{(dblMaxCol - dblMinCol)}{N \times 2^l \times 2^n} \end{cases}$$

Using the fixed root node, the resolution ratio of the DEM to DOM is 1 to 16 or 1 to 32. To affirm the rendering effect, the resolution of the horizontal and vertical direction had better have a consistent ratio.

The terrain block number is quadruple to that of the next higher LoD model, et al. The blocks are encoded with quadtree hierarchy code.

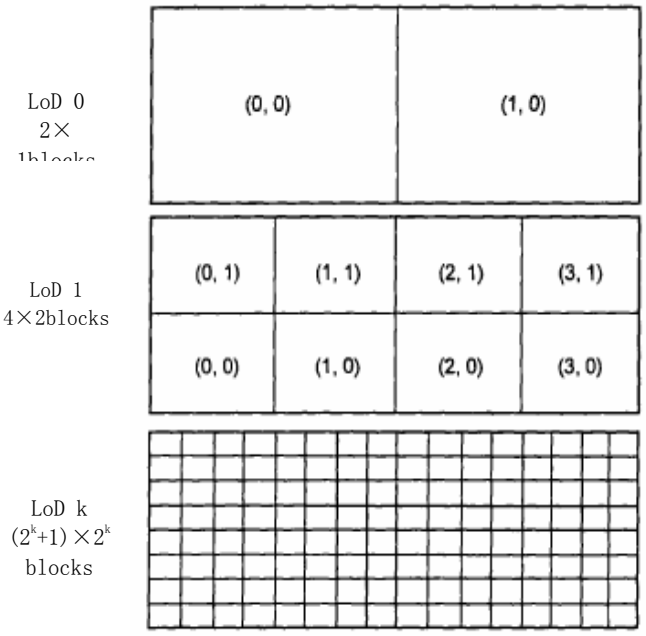

Figure 2. Leveling and blocking strategy on pyramid hierarchy

In data management, Quadtree structure is an effect method. In Quadtree structure, the node number in the lower is quadruple to the number of the higher level.

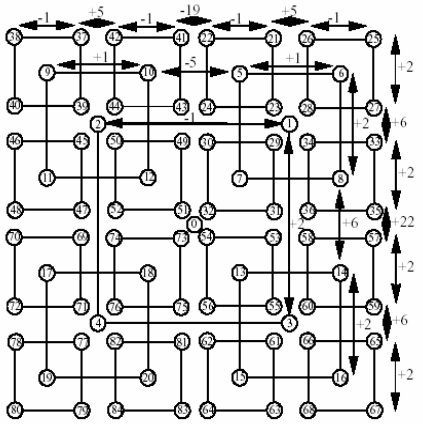

Figure 3 Quadtree structure in the management of The terrain data

In figure 4, the quadtree hierarchy levels are demonstrated. With the root being level 0, given the grid size dimension $n = 2^k+1$ and points $P_{i,j}; i,j = 0,1,\cdots n-1$, the points on lower levels $l \geq 0$ are given by $L_l = \{ P_{i,j} | h_l = 0,1,\cdots,2^l; i,j \in \{ 0, d_l, \cdots, h_l \bullet \cdot d_l \} \wedge P_{i,j} \notin L_{l-1} \}$. Note that the resolution $d_l = (n-1) / 2^l$ is level dependent. The center-vertices L_l center denote points of L_l which are

located in the center of a quadtree block on level ι - 1(white point in Figure 3).

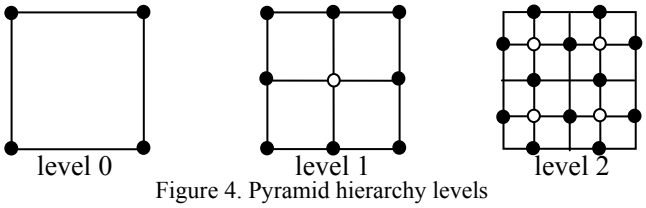

Figure 4. Pyramid hierarchy levels

3.2 Multi-level texture compressing techniques

In rendering the massive terrain data, the amount of DEM data is very large, the amount of DOM data is also very large. If the original resolution DOM data is used in visualization of the massive terrain data, the data would be excessive to the computer memory and lead to render the data in slow frame rates. And in rendering the terrain with high resolution DOM data, it is not suitable for view custom, for the scene with different view distance has different view effect. In effect, the scene near to view will be seen in detail and that far distant to the view is rough. So for texture image data, LoD technique is also applicable and useful.

In this paper, the texture data is processed with multi-level blocking. First the texture is blocked on the terrain scope and its location. Then each block of the texture data is processed with multi-level, using the resolution as 256 × 256,128 × 128,64 × 64,32 × 32,16 × 16. The resolution of the texture block is depended on its distance to the view, far with lower resolution and near with higher resolution.

Even with multi-level texture, the texture data capacity of a terrain block is about 256KB. If a scene covers almost 60 terrain blocks, the texture needing to be dealt with would be exceed 15MB. The reality is that the terrain block number will be much more than 60, so the demand to store, load and visualize the texture data is very ineffective and consume the computer capability and memory. Texture data compression is a good way to solve the problem. The author adopts the DXT1 format to compress the texture data.

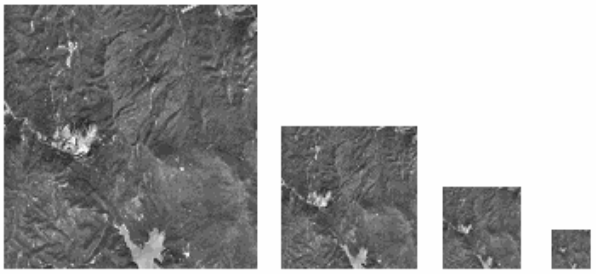

Figure 5. Multi-resolution texture data in DXT1 format

Compressed with DXT1 format, the texture data is loaded in high speed and consuming little capability and memory, which can improve the efficiency of the rendering of massive terrain.

3.3 Dynamic loading and rendering of the massive terrain data

In the visualization of the massive terrain data, firstly the terrain nodes which are in the view are determined. Based on the distance to the view, the terrain nodes are rendered in appropriate different level and the texture resolution of the terrain nodes is ascertained. The terrain is rendered based on the terrain node information.

In determining which nodes are in the view scene, the view matrix is computed. From the view matrix, each terrain node can be estimated by its four corner points. If anyone of its four points is in the view matrix, the terrain node can be seen in the view scene.

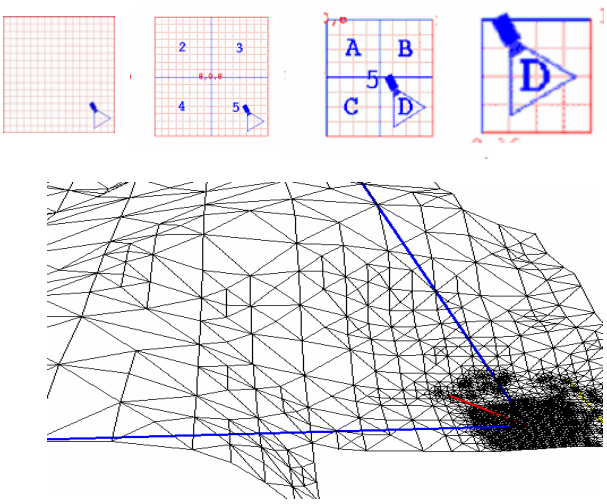

Figure 6. Determining the terrain nodes in the view scene

After the terrain nodes in the view scene are computed, they are rendered in different LoD level according to the distance to the view and their rough error metrics. By optimizing the computer capability and memory, the terrain nodes are loaded real-time. When they are leaving the view scene, they are unloaded from memory, but if they are going into the view scene, they are loaded dynamically. In this way, the memory to allocate the terrain nodes are fixed and limited to some proportion so as that the massive terrain data can be rendered in low performance capability.

3.4 Eliminating the terrain cracks in the adjacent terrain nodes

Under the pyramid hierarchy model, the terrain nodes are rendered using different LoD level for the different distance from the node to the view. That is to say the terrain nodes near to the view are rendered by higher resolution but the terrain nodes far from the view are rendered by lower resolution. For the terrain are rendered by different resolution, terrain cracks between the adjacent nodes would appear.

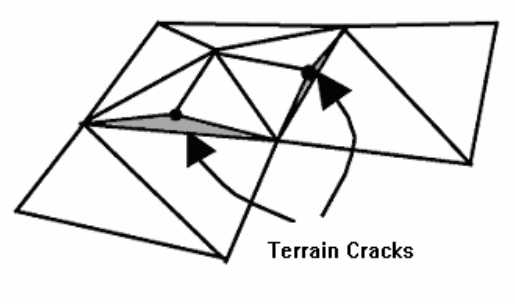

Figure 7. Cracks between the adjacent nodes in the terrain

In eliminating the terrain cracks, there are two methods, adding triangular or subtracting triangular generally. In this paper, subtracting triangular is implemented. In the adjacent lines between nodes of different resolution, the triangular in the higher resolution nodes will not be rendered. The dashed lines in figure 8 are the lines that will not be rendered in eliminating the terrain cracks in this paper.

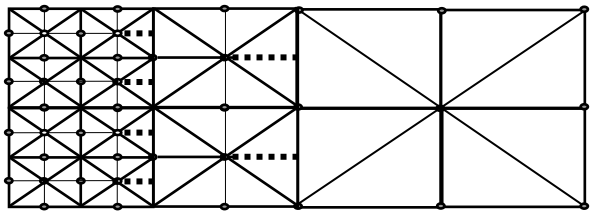

Figure 8. Strategy to eliminate the terrain cracks

4 EXPERIMENT RESULTS OF THE ALGORITHM

The authors have implemented our algorithm, achieving frame rates of 15fps on PⅣ1.60GB computer with 512M RAM and NVIDIA GeForce2 MX100/200 acceleration under Windows Visual C++ OpenGL Environment.

In experimental test, the terrain data are over 4GB, including DEM data and DOM data. Seven-level pyramid hierarchy model is created. After DXT1 format compression, the DOM data becomes less than 700MB, with the preprocess time 30 minutes.

In rendering of the terrain data, the average speed is 60 fps, with the triangular number about 20000, in interactive frame rates.

5 CONCLUSION AND FUTURE WORKS

The authors present a dynamic rendering algorithm for massive terrain data. From the experimental test, the algorithm, creating pyramid hierarchy data model and dynamic rendering of terrain, can support the massive terrain data.

But now the terrain model is managed in file format on the basis of single-computer mode, no-supporting the multi-user to access the data synchronously. So the research on the basis of database is to be carried out and taken steps for further research.

REFERENCES

[1] P.Lindstrom,D.Koller,et al. Real-time continuous level of detail rendering of height field.Computer Graphics(SIGGRAPH'96 Proceedings) ,P109-118,1996

[2] Duchaineau,M.et al. ROAMing Terrain: Real-Time Optimally Adapting Meshes. IEEE Visualization 1997,P81-88.Nov.1997

[3] Roettger, Stefan, Wolfgang Heidrich, Philipp Slusallek, and Hans-Peter Seidel. Real-Time Generation of Continuous Levels of Detail for Height Fields. In V. Skala, editor, Proc. WSCG '98, pages 315–322, 1998

[4] DuYing, A Research on Key Technologies for Global Multi-resolution Virtual terrain Environment, dissertation for doctoral degree, Information Engineering University of PLA, 2005