A RAPID UPDATING METHOD OF LOCAL HIGH-PRECISION TERRAIN DATA IN 3D TERRAIN SCENE

Hao LIU

Academy of Disaster Reduction and Emergency Management; Beijing Normal University, Beijing 100875, China
(lihaha1985@126.com)

ABSTRACT: In many applications, three dimensional terrain scene has to be updated in local area with high-precision data, in order to build a local high-precision scene. Aim at the dynamic updating of local terrain data, this paper puts forward a GPU tessellation-based "Mosaic" method of local high precision terrain. Through the interpolation and subdivision on the terrain grid within the updating area in GPU to increase the number of triangles, as a result, the geometrical accuracy of the local terrain model is improved without increasing the bandwidth of the memory and video memory. The results show that the precision of updating area meet the demand and this algorithm's execution efficiency is obviously better than the common method.
KEY WORDS: Three dimensional terrain scene; Local terrain updating; Real-time grid tessellation; GPU algorithms

1. INTRODUCTION

The 3D terrain scene has been widely applied in many fields, in many applications the user often pay more attention to the local area of the scene, take emergency rescue in the earthquake as an example, the user often focus on residential areas and indifferent to the mountains, so it is necessary to construct a more elaborate disaster scene within the important area. This paper aim at construct local fine scene with high precision data based on the large scale virtual scene, also is to realize the fast updating of local terrain scene.

At present, the 3D terrain application system usually update local terrain data by reprocessing the data, integrating the new data to the original data set, and then restarting the terrain visualization system, which is contrary to the original intention of the real-time simulation system. Dynamic updating takes grater advantages: (1) Considering the spatial distribution, the focused area may be any part of the scene, and may be chosen by user's impromptu decision. So it is difficult to construct the important virtual scenes once by data preprocessing. (2) Considering the continuous, local data are usually discontinuous, the simulation system can dynamically loading new data in real-time, without shutting down the system.

2. RELATED WORKS

The data organization and the rendering processes of the existing terrain rendering method[1-2] are interrelated and inseparable, is aimed at a single scale terrain rendering or similar scale, therefore can only satisfy the scene rendering of fixed size and form. In view of the present study, there is few aim at unified terrain rendering of multi scale, Li Wenqing[3] realized unified rendering of global terrain scene and local scene by GPU vertex and fragment program, at the rendering stage of global terrain and local with two separate class, which is able to render multi-scale terrain, but greatly increase the amount of calculation.

This paper takes advantage of the new features of modern GPU-Tessellation, which use the Tessellator (a internal hardware unit of GPU) to subdivide rough terrain mesh by adding an extra vertex, and greatly increase the amount of triangles.

In recent years, the GPU tessellation technology is gradually applied to terrain rendering. Story[4] and Ni[5] introduced DirectX11 GPU tessllation rendering pipeline, and the general principles and methods for real-time rendering of terrain with tessellation; Rollin[6] and San[7] discussed the properties of GPU tessellation of OpenGL and its application to terrain rendering; Cantlay[8] and Valdetaro[9] using GPU tessellation achieved multi-resolution terrain rendering; Yusov[10] realized terrain rendering by LOD selecting and Patch segments. Due to the patch is used as the processing unit, adjacent patches are relatively independent, terrain rendering algorithm based on GPU tessellation do not need hierarchy management, and the consistency problem between adjacent patches on different levels can be solved through shared edge subdivision. It has a natural advantage for local high precision terrain "mosaic"

3. OUR METHOD

Due to the terrain structure is in a fixed state when updating the local data, there is much difference in area and precision of local data and large scale terrain. Dynamic loading of new high precision data is seen as a "mosaic" operation in the existing low precision terrain. As the meshes of rendering terrain has been fixed, the new loaded data with higher precision are still sampled with old resolution, as a result, it is unable to reflect the accuracy of new loaded data. In this case, remote sensing images rendering as textures also appear fuzzy.

The problem that terrain meshes is oversized can be achieved through mesh refinement. However, there would be much redundancy if refinement is done to the whole scene, so refinement can only be done to the covered area of new data. The process is shown as Figure 1.

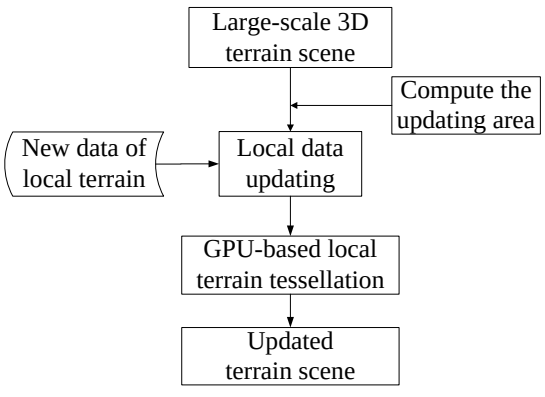

Figure 1. Flow chart of local terrain data updating

3.1 GPU tessellation

In order to increase the details of terrain meshes, extra terrain mesh points are needed, this could be completed by GPU. The development of GPU rendering pipeline might be divided into 3 stages: The first stage is only for simple geometric

transformations and drawing; The second stage is Geometry shader which is used to create new elements in GPU with low efficiency; The third stage is GPU tessellator, which is fixed in the DirectX11 graphics rendering pipeline as a device unit, can be used to increase details of meshes efficiently.
The GPU tessellation can be applied efficiently to terrain rendering by increasing details of rough terrain meshes, the accuracy of the terrain model is improved without increasing the transmission pressure between graphics memory and the main memory.

3.2 Local terrain "mosaic"

Local terrain "mosaic" algorithm is designed based on GPU tessellation, as shown as Figure 2. The initial terrain grid resolution is determined by the initial terrain data, when new data is loading, compute the coverage of new data first, and mark the mesh grids within the scope, then compute the tessellation coefficient through new data, the coefficient will determine the degree of subdivision. Finally, the mesh is tessellated and rendering.

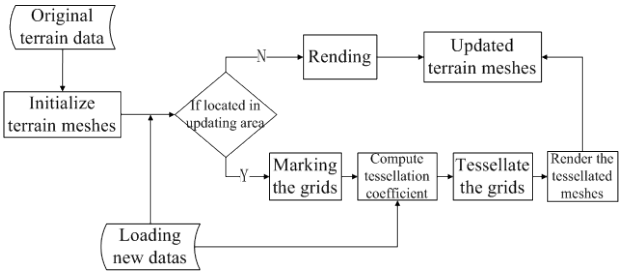

Figure 2. Flow chart of local terrain “Mosaic” algorithm

Figure 3 shows the processing flow of the local terrain “mosaic” in GPU. The processing of local terrain “mosaic” is embedded into GPU rendering pipeline, when updating, the area of new data is computed in real-time, then the area and spatial scale information are transferred to Hull shader, and then compute the tessellation coefficient which is then transferred to tessellator and tessellated the meshes.

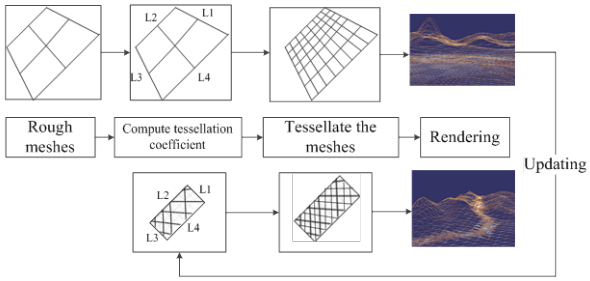

Figure 3. Rendering terrain with new graphics pipeline

3.3 Generate tessellation coverage map dynamically

The coverage of new data is random, and has irregular shape, so it is hard to accurately compute the coverage only by simply reading metadata. We rendered the coverage of new data into a gray map by Render to Texture in the pixel shader, the gray map records the area that would be tessellated. The coverage is rendered onto the gray map by orthogonal projection, as shown as Figure4. The gray map initialized in black, the initial value is 0, only the pixels that located within the coverage would be changed and located between [0-255], the image will be transfer into Hull shader.

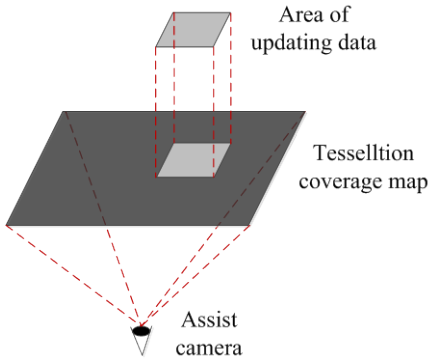

Fig.4 Get the coverage of new data with orthogonal projection

3.4 Compute the tessellation coefficient

The tessellation coverage map assigned the coverage of tessellation, but did not specify the degree of tessellation, which is decided by the coefficient, tessellation coefficient will be computed in the Hull shader. Usually, the following factors will be considered into the calculation of tessellation coefficient: side length of patch, elevation value, screen error and viewpoint. However, because the Hull shader has calculated tessellation coefficient in rendering phase for rough terrain mesh, the coefficient here(L) is a further tessellation parameters, which is added to the initial coefficient. The calculation process of L is as follows:
Suppose the minimum execution unit(Patch) is a rectangular, the sampling interval of initial terrain data is 30m, while the new data’s is 5m, tessellation is operated in diploid until the minimum interval is less than or equal to the target sampling interval(5m).

3.5 Tessellation control of tessellator

Tessellator is able to formulate tessellation coefficient for each edge of a patch, 4 kinds of tessellation mode are supported: Integer Power, Pow2, Fractional_even and Fractional_odd, due to the tessellation coefficient of each edge could be inconsistent, which is helpful to realize the smooth transition between different subdivision level, thereby eliminating the jump phenomenon between levels. We choose Fractional_even mode in order to realize the continuous terrain meshes, the segmentation results in Figure 5.

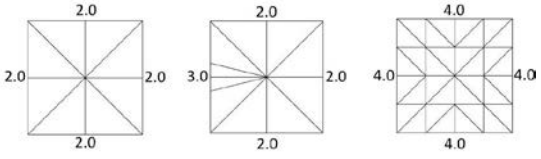

Fig.5 Effect of Fractional_even mode with different coefficient

4. EXPERMENT RESULTS

To verify the correctness and effectiveness of the algorithm, we completed the algorithm in the windows environment based on Direct3D and C++. When dynamic loading new terrain data, the system first generate a coverage map according to the geographical scope, then calculate the tessellation coefficient based on the precision of new data and original data, the sampling interval of the original data is 90m, while the new data's is 5m.

Figure 6 shows the effect of local high precision terrain mosaic, the mesh density of updated area is significantly higher than the surrounding area, and there is not cracks between different levels while the mesh density has a great leap, which is the advantage of GPU tessellation based method. Effect of drawing, Figure 7 shows the effect of remote sensing image rendering within the updated area.

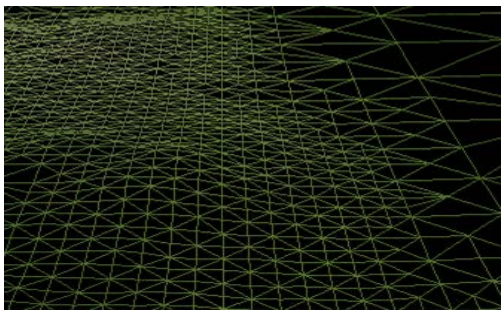

Figure 6. Effect of local high-precision terrain "Mosaic"

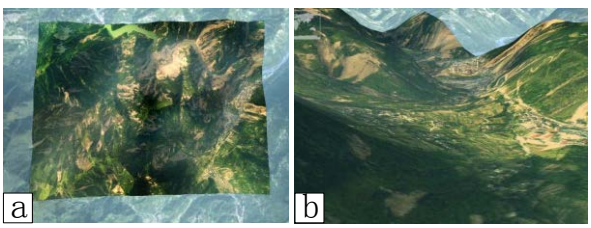

Figure 7. Effect of RS image overlaying on the updated terrain

In addition, we tested the execution efficiency of our method, as is shown in Table 1, the sampling precision of original terrain mesh is 90m, the local elevation data's is 30m and 5m, we compared our method with exiting GPU algorithm[3] and CPU algorithm, the results show that efficiency of our algorithm is better than that of the exiting methods.

Precision	Quantity	CPU	GPU	Our
30m	500MB	82.23s	45.23s	28.15s
30m	800MB	130.57s	75.42s	33.27s
5m	600MB	160.23s	87.15s	36.35s
5m	1000MB	262.11s	110.35s	44.84s

Table 1. Time consuming of dynamic data loading

5. CONCLUSION

Aiming at the dynamic updating of local terrain data, from the accuracy and execution efficiency, this paper proposed a rapid updating method for local terrain "mosaic" based on GPU real-time tessellation. Access to the coverage of updated data, tessellation coefficients is calculated and mesh tessellation is completed in GPU. Experiment results show that, the accuracy and efficiency of our algorithm are both superior to the existing algorithms.

6. REFERENCES

[1] Du Ying. A research on key Technologies of Global Multi-resolution virtual terrain environment. The PLA Information Engineering University, 2005.

[2] Yu Zhuo, Liang Xiaohui, Ma Shang, et al. A real-time rendering method for large terrain including details in different resolutions. Journal of Computer Research and Development, 2010, 47:988-995.

[3] Li Wenqing. Research and implementation of VV-Ocean System on marine environment simulation and data visualization. Ocean University of China, 2011.

[4] Story J, Cebenoyan C. Tessellation performance. (2010-03) [2011-10-11]. http://developer.download.nvidia.com/presentations/2010/gdc/Tessellation_Performance.pdf

[5] Ni T Y. DX11 tessellation. (2010-08) [2011-10-11]. http://www.nvidia.asia/content/asia/event/siggraphasia2010/presos/Ni_Tessellation.pdf.

[6] Rollin P,Oster B.OpenGL and CUDA-based teselltion. (2011-08)[2011-10-11]. http://www.nvidia.com/content/siggraph/Rollin_Oster_OpenGL_CUDA.pdf.

[7] San Jose C. OpenGL 4. 0 Tessellation for professional applications.(2010-09)[2011-10-11]. http://www.nvidia.com/content/GTC-2010/pdfs/2227_G-TC2010.pdf

[8] Cantlay I. DirectX 11 terrain tessellation.(2011-01)[2011-08-16]. http://developer.nvidia.com/sites/default/files/akamai/gamedev/files/sdk/11/TerrainTessellation_White-Paper.pdf

[9] Valdetaro A, Nunes G, Raposo A, et al. LOD terrain rendering by local parallel processing on GPU, Games and Digital Entertainment (SBGAMES), 2010 Brazilian Symposium on,vol.,no.,pp.182,188,8-10Nov.2010.

[10] Yusov E,Shevtsov M. High-performance terrain rendering using hardware tessellation. Journal of WSCG, 2011,19(3):85-92.