REAL-TIME RENDERING OF LARGE TERRAIN ON MOBILE DEVICE

JiangWen [a], BaoshanZhu[a], FanWang[a]

[a] Zhengzhou Institute of Surveying and Mapping, No.66, Longhai Road, Zhengzhou, China - kissfro9642@sina.com

Commission VI, WG VI/4

KEY WORDS: Terrain rendering, Multi-resolution, Level of detail, Quad-tree, Adaptive rendering, Mobile device

ABSTRACT:

Terrain rendering is an important factor in representation of virtual scene. If terrains are large and detailed, there will be huge amount of data, so it is necessary to reduce the complexity of the rendered scene in real-time on mobile device. This paper proposes a multi-resolution technique to simplify the scenes and improve the speed of terrains rendering. Firstly, the full terrain height-field is divided into regular tiles, and then the appropriate level of detail is computed and generated dynamically, allowing for smooth changes of resolution across area of the surface. Each visible tile is then rendered using a computed triangle strip in an adaptive way according to viewpoint. The method is different from the triangle-based LOD algorithms and is optimized for modern to minimize CPU usage during rendering. The key of the technique is to develop an adaptive LOD framework that can optimally feed the graphic pipeline. At last, this paper also proposes a method of removing cracks on the meshes boundary.

1. INTRODUCTION

Terrain rendering on mobile devices plays an essential role in wide range of applications such as video games, virtual reality, 3D environmental analysis, personal navigation and many geographic information system (GIS) applications. Despite mobile devices have seen dramatic improvements in last few years, the mobile devices are still clearly less capable than desktop computers in many ways. They run at a lower speed, display in smaller size and have lower resolution, there is less memory for running the programs and for storing them, and the battery of device can not last for long.

On the other hand, terrain data obtained from the natural environment is usually very huge, and rendering accurate terrain implies the manipulation of very massive data sets which may contain billions of samples (e.g. triangles, points .etc.) and all those samples must be computed one by one instantly. In general, real-time rendering of three dimensional computer graphics requires faster than 15 frames pre second (FPS). Such a complexity introduces two main limitations: it might not be possible to store the entire data sets in random-access memory (RAM) and/or to perform its rendering in real-time on mobile device.

In fact, rendering 3D terrains on mobile devices is still a very complex task because of the vast computational power required to achieve a usable performance. There are many visualization techniques that have been developed for PCs and workstations. However, using these same approaches for mobile devices introduces some unresolved problems. Most of the existing approaches are simplification of a triangulated model that represents terrain surface. Some solutions entirely rely on CPU whereas others use both CPU and GPU (sometimes using programs). GPU are able to render millions of triangles per second and even more when using triangles strips. However the GPU accelerated method still requires the full CPU power to compute vertex indices at every frame, even if the method is very efficient, it is not practicable when targeting mobile devices. Indeed, although some recent mobile devices dispose of GPU, they are not yet programmable.

Aim at the target of real-time large terrain rendering on mobile platform; this paper propose a dynamic adaptive multi-resolution modelling to represent terrain based on quad-tree. The solution can be decomposed in two main parts. The first one is pre-processing. The main purpose is to construct multi-resolution digital elevation models (DEM) to represent terrains. The quad-tree structure is the key point in this part. The second one is dedicate to render a maximum number of triangles in view region. The purpose of this part is reducing the triangles to be drawn and maintaining the largest polygon area around the viewpoint. By loading tiles which are in view-field into client memory dynamically, it can free some memory and reduce the burden of CPU.

2. PREVIOUS WORKS

There are two familiar methods in the terrain rendering domain. The first one brings together methods that have been designed for terrain models which fit in memory (level of details technique). The second one gathers the algorithms designed for the rendering of large terrain data which can not be loaded into memory completely (out-of-core techniques).

2.1 Previous terrain LOD techniques

Terrain LOD algorithms use a hierarchy of mesh refinement operations to adapt the surface, and the methods are widely used in large terrain rendering now. LOD can decimate polygons thus reducing complexity of computation without affecting the quality of scenes. There are two schemes to choose proper LOD. One is based on the flatness of terrain surface. Large and coarse meshes are used at even regions, while tiny and refined meshes are represented at fluctuant regions. The other is based on the distance away from the eyes' position. Regions which are nearer to the eyes' position are rendered in refined meshes and the further regions are rendered in coarse meshes.

During pre-processing, terrains are represented as multi-resolution meshes, which can be generated from bottom to top (or refined-to-coarse, in which a full resolution model is created at first. Then triangles are merged recursively until a screen space error tolerance is exceeded), or from top to bottom, (or coarse-to-refined, which generates a coarsest-grained model at first, then refines it). The computation complexity depends on the vertex number in the original mesh model. So the latter is much simpler.

The multi-resolution representation are arranged in one or more quad-trees (or its equivalent, triangle bin-trees), or represented as wavelets. At run time, proper levels are selected.

Most of the following approaches are based on the management of triangulated irregular networks (TINs) which provide the best approximation for a given number of faces, but require the tracking of mesh adjacencies and refinement dependencies. The mesh is refined in real-time according different strategies. [Lindstrom et al. 1996] introduce a real-time smooth and continuous LOD reduction using a mesh defined by right triangles recursively subdivided according a user-specified image quality metric. Some hierarchies use Delaunay triangulations [e.g. Cohen-Or and Levanoni 1996; Cignoni et al 1997; Rabinovich and Gotsman 1997] while others allow arbitrary connectivities [e.g. De Floriani et al 1997; Hugues Hoppe 1998; El-Sana and Varshney 1999]. In [Duchaineau et al. 1997], the authors introduced ROMAing method as a very efficient algorithm based on triangle diamonds managed with split and merge operations performed using priority queues. The algorithm now is widely used in games industry, but its implementation is tedious according to [Blow 2000]. In 2002, [Levenberg] propose to reduce the CPU overhead of the previous binary-triangle-tree-based LOD algorithms by manipulating aggregate triangles instead of simple triangles.

In a recent paper, Losasso and Hoppe [2004] apply the clipmap [Tanner et al. 1998] concept to geometry for large terrains rendering. Their GPU accelerated method is based on a set of nested regular grids centered about the viewer. Geometry continuity is guaranteed by using transition regions between two grid levels using the GPU vertex shader. They use a compression algorithm to load the full terrain model in memory. However, this still requires the full CPU power to compute vertex indices at every frame. In a more recent paper, Asirvatham and Hoppe [2005] enhanced the approach by performing nearly all computations on the GPU. Furthermore, even if the method is very efficient, it relies on shaders, which is not practicable to handheld devices and/or mobile devices.

Ideally, view-dependent LOD algorithms adaptively refine and coarsen the mesh based on screen-space geometric error, the deviation in pixels between the mesh and the original terrain. Screen-space error combines the effects of (1) viewer distance, (2) surface orientation, and (3) surface geometry. Since surface orientation seldom provides significant LOD gain, many schemes choose to ignore it. One common refinement criterion [Blow 2000] stores at each vertex a radius defining an enclosing sphere. The pre-computed radius encodes the local surface approximation error, such that the neighborhood of the vertex is refined if and only if the viewpoint enters the sphere. In view-dependent algorithms, a terrain can be thought of as a displacement map over trivial planner geometry. Some recent papers have proposed hardware schemes for adaptive tessellation of displacement maps [Gumhold and Hüttner 1999; Doggett and Hirche 2000; Moule and McCool 2002]. So far these schemes have only been simulated on relatively simple grids, and they assume that the entire grid is memory-resident.

2.2 Out-of-core technique

With this aim in view, some other approaches propose to perform either out-of-core rendering (local solution) or streaming (networked solution) of the models.

[Pajarola 1998] extends the restricted quad-tree triangulation of Lindstrom [1996] with another vertex selection algorithm and amore intuitive triangle strip construction method. This is combined with dynamic scene management and progressive meshing to perform out-of-core rendering. More recently [Cignoni et al. 2003b; Cignoni et al. 2003a] described a technique for out-of-core management and rendering of large textured terrains named batched dynamic adaptive meshes (BDAM). BDAM is based on a pair of bin-trees of small TINs that are computed and optimized off-line. The batched host-to-graphics communication model guarantees overall geometric continuity, exploits programmable GPU's, a compressed out of core representation and a speculative pre-fetching for hiding disk latency. These solutions are still impracticable for our objectives since they rely on low latencies between mass storage and main memory. Furthermore, these solutions also present high CPU costs.

Other methods rely on the web. [Reddy et al. 1999] described TerraVision Ⅱ that is a geo-referenced VRML97 terrains viewer. A quad-tree hierarchy of the VRML97 LOD mode which induces a lot of data redundancy and no care is taken to ensure continuity between different grid levels. A more advanced solution proposed by [Aubault 2003] relies on a wavelet encoding to perform terrain streaming and multi-resolution rendering. Still, this very efficient solution requires to fetch the entire model into server's memory and to perform costly computations on it.

3. TERRAIN REPRESENTATION

A terrain (elevations) can be defined in several ways. First of all it can be defined as an arbitrary mesh also known as Triangulated Irregular Networks (TINs). This method does not put any restriction on the terrain, and has been used in terrain rendering. TINs provide the best approximation for a given number of terrain faces. However the algorithms are very complex, consume more memory, and are not very efficient for view-dependent simplification. So another method is proposed to define the terrain as a height map, which is a grid structure that is equally spaced in the x and y directions. The z value is used as the height information. The grids data are simple and disciplinary, and consume less memory. But grids DEM are not very flexible to describe terrains with uniform criterion. If the grid space of height field is too wide, it tends to lost detail of terrains especially at fluctuant region. If the grid space is too narrow, there will be a lot of redundancy. In order to solve the problem, this paper proposes a dynamic adaptive multi-resolution modelling to represent terrain based on quad-tree. It has chosen terrain representation as height map as it allows fast collision detection between moving objects (including camera) and the terrain. It also supports use of hierarchical data structures for fast and easy view frustum culling.

Considering visualization of very large real world digital terrain

model data, which will not fit into main memory entirely, especially for terrain walking through application, it must to organize the digital height map data in tiles. The tiles management algorithm aims at maintaining the largest square area around the view point; those squares of tiles assure the user can always look around at any point of view. The size of the tile is set according to the available main memory which makes it adaptive to the mobile device that is used for visualization. On the other hand, in order to subdivide those tiles based on quad-tree properly, this paper uses a simple strategy. For each level *l*, with grid spacing $S_l = 2^{-l}$ in world space, it is let the desired active region be the square of size $nS_l \times nS_l$. Each tile is represented by hierarchical quad-tree data structure. The top-level node in this tree structure represents the area of the entire tile, its children each represent one fourth of the terrain area, their children in turn each cover one sixteenth of the area, etc, and then each tile will be encode and store into a array (see figure 1).

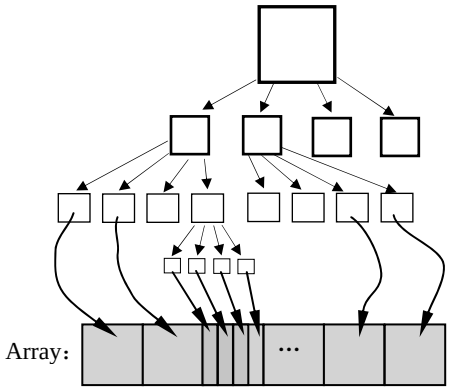


Figure1. Hierarchy of tile based on quad-tree

One advantage of having this particular terrain block layout is that one such block can be optimized for rendering, using one draw-primitive call for the entire block and, even better, using grid indexing to get rid of multiple transformations of vertices. Pyramid representation is used to define each block of size. A pyramid is a multi-resolution hierarchy for a data set. Same approach when dealing with image textures is known as mipmapping. It helps in dynamic multi-resolution level of detail mesh simplification of height map data.

Then the tiles will be subdivided according to terrains. Intuitively, those fluctuant areas hold more details and need refine mesh to represent and even areas only need coarse mesh to represent (see Figure 2).

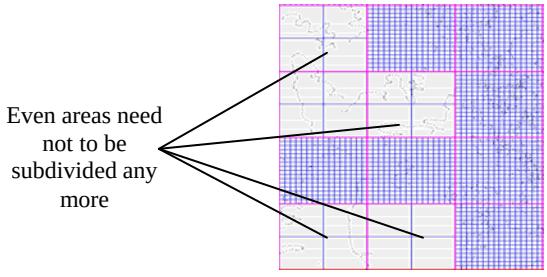


Figure 2. Adaptive subdivision of tiles based on quad-tree

To subdivide the tiles, a criterion or a set of criteria is needed. According to the criterion, terrains are represented in various resolution meshes. In this paper, the criterion is difference height difference of terrains, and the algorithm sets a threshold of height to compare with the maximum height difference of subdividing grids.

Algorithm 1 Adaptive tiling based on quad-tree

```
if not subdividing tile then
   MAX = maximum height difference of tile
   if MAX > THRESHOLD then
    subdivide the tile into square grids
   max = maximum height difference of grid
   if max > THRESHOLD then
     continue to subdivide the grid
     else
          stop subdividing, encode the grids which been
       subdivided and record them into database
     end if
```

4. ADAPTIVE LOADING TILES AND RENDERING

For mobile devices have less memory to run programs, it is impossible to load triangles entirely of terrains into memory (the client's memory). With our solution the database is made of a set of tiles, each one containing the regular terrain elevations of a tile. A metadata file which records a description of the tiles grid (tiles size, number of tiles, positioning etc.) is first fetched (or loaded) and its information are then used to manage the adaptive tiling.

4.1 Tile data structure

In theory, tiles can be divided in any size and any forms, but for the sake of simplicity this paper only consider the specific case of tile of square form, and the specific case of tile of size $w = h = \left(2^n + 1\right)$ and its data store in an array whose resolution ($w \times h$) stores elevation value of the sampled terrain area. The memory representation of a tile is a vertex buffer that embeds the 3D coordinates together with their properties just as texture coordinates normal.

Structure of triangle

```
typedef struct
{
    m_TriangleVex[3]      ;  //  vertex 3D coordinates
    m_TriTextCoord[3]     ;  //  texture coordinates
    m_TriangleNormal[3]   ;  //  vertex normals
} STRUCT_TRIANGLE
```

Tile data structure

```
typedef vector<STRUCT_TRIANGLE> VEC_TRIANGLE ;
typedef map<GRID_CODE, VEC_TRIANGLE ,lpGridCode>
MAP_DEMTRIS_MESH ;
```

The multi-resolution is generated by creating a set of coarse to fine tiles. A tile is defined by connecting only the vertices with ($i \times j$) coordinates (within the $w \times h$ array). By the grid index, we can find the tiles which want to be load in client memory soon.

4.2 Adaptive rendering tiles

View-frustum culling techniques are used to control substantial amount of polygons in the rendering pipeline. Only those tiles

which lie in frustum view region will be loaded, and search those tiles is by grid index (see figure 3a). If viewpoint moves over an adjacent tile the algorithm will tend to maintain a square of tiles centered on this new tile (which will be load in client memory new and becomes the current tiles). At the same time, the algorithm will remove some far tiles which are not within the field-of-view in order to free memory for the fetching of new tiles (see figure 3b). As figure 3a indicated, most of the memory of mobile device was consumed at the step. Note that the algorithm implicitly handles the case where viewpoint jumps to a new tile that is not adjacent to the current one. The quad-tree representation of tile data enables very fast view frustum culling.

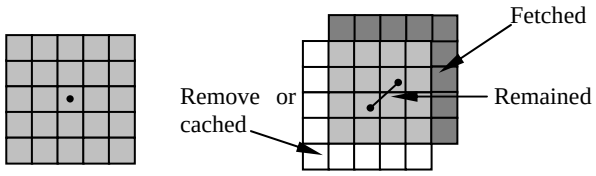


Figure3. Tiles management and adaptive loading
a). A square area centered on the viewpoint.
b). Square area preservation on viewpoint move

Farther in more, considering the tiles are made of a set of triangles, the triangles of the tiles will be judged one by one, and those trigngles which lie in the field-of-view are fetched. At last those triangles that lie out of view frustum will be removed.

4.3 Tile rendering and cracks eliminated

When the resolution levels are different between two adjacent tiles, there will be gaps on tiles boundaries which create a very unpleasant visual effect (showing in figure 4a) Classical approaches [Larsen and Christensen 2003; Lossa and hoppe 2004] consist in modifying the geometry of the tile's border by introducing new vertices and edges. Another classical method called filleting, introduced by Sun and also implemented in the NASA's World Wind remarkable earth viewer is to add a band of vertical triangles around the edges of each tile. This band is stretched down to the lowest terrain elevation. Each side of the band is textured by stretching the corresponding line/column of texels. This scheme is fast but quite disgraceful to see for the user, especially if a neighbour tile has not been load yet.

In this paper, another method which divides the triangles which are in coarse tile compulsively (see figure 4b) is proposed. If there appear cracks for the different resolution levels between adjacent tiles, the triangles which lie on the boundary in coarse resolution tiles will be divided compulsively to suit the fine resolution tiles. Even if this solution is not perfect, it is fast, simple to implement and gives satisfying results most of time.

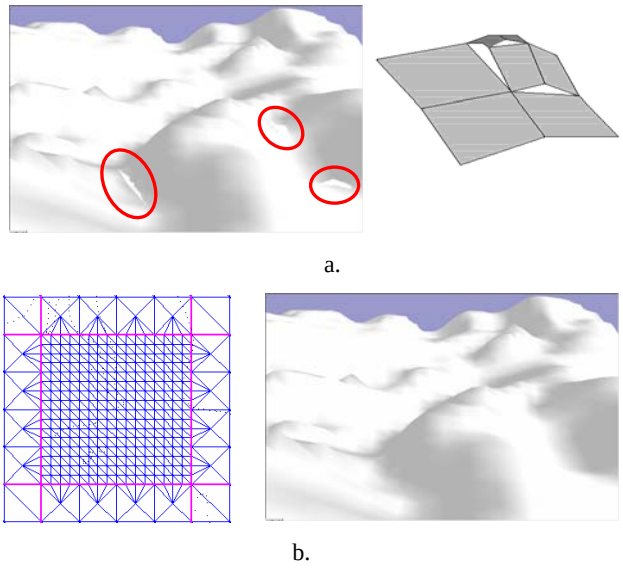


Figure 4: Eliminating cracks. a) Cracks appear on tiles borders when adjacent tiles have different levels. b) Using dividing compulsively technique, crack effects are eliminated or attenuated.

5. RESULTS AND CONCLUSION

This paper proposes an adaptive approach to render large terrains. By this means, it simplify the scenes' complexity efficiently and reduce the amount of data and therefore of graphical primitives to render in real-time.

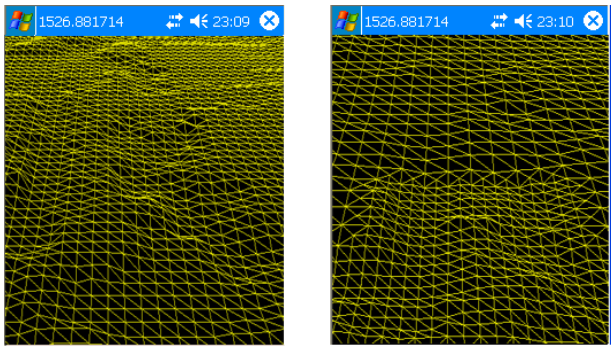
5.1 Experiment and results

Our experiment selected Hp2110 as the experiment platform, and its primary configurations are shown in table 1.

CPU	Intel PXA270
Memory	64 Mb (32Mb RAM + 32Mb ROM)
Screen resolution	320×240
Storage	1GB SD card
Operating System	Pocket PC 2003

Table 1. Configurations of Hp2110

The experiment first measured the difference between the simplify scene and primal scene. The amount of primary triangles in client memory is about 70,000 (see figure 5a), and after being simplified, the amount of triangles reduced to 16064, almost 77.1 percent triangles are removed or cached (seeing figure 5b).



a b

Figue5. Comparison between primary scenes and simplified scenes. a) Primary terrain scenes. b) Simplified terrain scenes

The second experiment is performed on a data set that model an area sampled ZhengZhou region in HeNan province, China. The whole experiment area is about 400Km^2. Each frame contains about 2,000～3,000 triangles. Before simplified, the frame rate can not beyond 4 Fps. After simplified the scenes, the target frame rate equal to 8 per second (see figure 6).

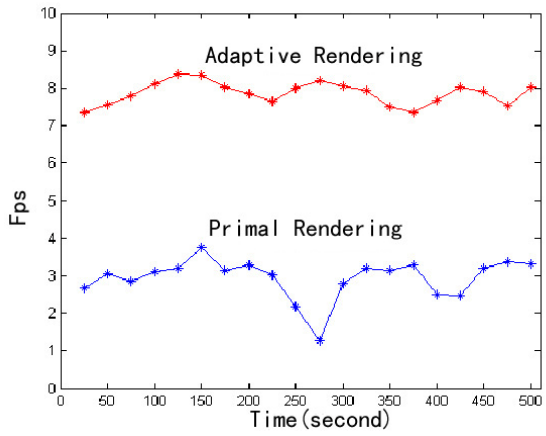


Figure 6: Comparison between adaptive rendering and primary rendering

5.2 Conclusion and future work

Rendering 3D large terrains on mobile devices is still considered a formidable task. This paper proposed a solution for 3D visualization of data on mobile devices and we showed that interactive frame rates could be achieved with relatively large amount of displayed data. Around a tiling algorithm and a per tile multi-resolution data structure, this paper designed an adaptive technique to improve rendering efficiency. On the other hand, the approach of simplify rendering scene rely on the surface of terrains, so it won't be very effective in mountain area.

In the future, we will focus on the extension of our scheme and use a multi-resolution data structure for the progressive and adaptive transmission of each tile as well. In this way, the tiles will be loaded only when needed. Moreover, this will allow a faster loading of visible tiles and offer the possibility to load farther tiles at lowest resolution. We can see how 3D graphics as well as 3D visualization of data is becoming more and more as common on mobile devices as on desktop computers.

REFERENCE

Arul Asirvatbam, Hugues Hoppe, Terrain Rendering Using GPU-Based Geometry Clipmaps. PP. 27-45

Frank Losasso, Hugues Hoppe, Geometry Climaps: Terrain Rendering Using Nested Regular Grids.

Joachim Pouderoux, Jean-Eudes Marvie, Adaptive Streaming and Rendering of Large Terrains using Strip Masks

Jungwon Yoon, Jeha Ryu, A Novel Navigation Algorthm for Locomotion Interfaces with Programmable platforms

Ming Fan, Min Tang, 2003. A Review of Real-time Terrain Rendering Techniques, The 8th International Conference on Computer Supported Cooperative Work in Design Proceedings, pp. 685-691

Miran Mosmondor, Hrvoje Komericki, Igor S. Pandzic., 2006. 3D Visualization on mobile devices. Telecommun Syst,32, pp. 181-191.