A DYNAMIC MULTI-RESOLUTION MODEL AND IT'S APPLICATION TO TERRAIN RENDERING

Xu Qing Zhang Baoming Tan Bing Ma Dongyang

Institute of Surveying and Mapping, Information Engineering University
ZhengZhou 450052, P.R.China
xq64@371.net

Commission IV, WG IV/6

KEY WORDS: DEM, Level of Detail, Multi-resolution Model, View Dependent，Terrain Rendering, Texture Mapping

ABSTRACT:

In the field of terrain visualization, the most efficient tool for real time rendering of complex landscape is level of detail (LOD) technique. In this paper, an optimization quadtree-based algorithm for dynamic generation continuous levels of a given complex landscape in real time is developed. In this algorithm, the conception of view-dependent is introduced, the fast searching algorithm for quadtree nodes is realized and the algorithm for repairing the cracks intrinsically caused by quadtree data structure is improved. The experimental results shows that the algorithm proposed can dynamically generate multi-resolution terrain model and real time terrain rendering can be attained. Furthermore, an all-in-one terrain visualization system which integrates adaptive triangulation, dynamic scene paging and spatial data updating is realized, thus large-scale terrain visualization at interactive frame rate can be achieved.

1. INTRODUCTION

In recent years, with the rapid development of technology and the progress made in the fields of *Computer Vision, Science Visualization, Photogrammetry, Remote Sensing and Computer Graphics*, it's common to generate high detailed terrain model with the data gained by photogrammetry. These complex terrain models consist of millions of triangles or more satisfied people's demand for high sense of realism, however, the rendering performance of these data is seldom considered, which challenges the ability of rendering efficiency and computation capability of computers greatly. The most common approach is to exploit the traditional 3D graphics pipeline, which is optimized to transform and texture-map triangles and can provide high performance of geometry acceleration. But it's still far to meet the practical demand even with the most advanced graphics pipeline available. With Digital Earth project presented and the demand for large-scale terrain simulation, the visualization of large-scale Digital Terrain Model (DTM) proves to be the bottleneck.

General speaking, a complete system to display views of large datasets at high frame rates consists of components to manage disk paging of geometry and texture, LOD selection for texture blocks, LOD for triangle geometry, culling to the view frustum and triangle stripping. Thus, terrain simplification, multi-resolution representation and real-time dynamic rendering have become the focuses of large-scale terrain visualization. As to these topics, three kinds of effective methods were developed, i.e. the visibility preprocessing[1,2], the image-based rendering[3,4] and LOD(level of detail) algorithm[5-17]. Considering the efficiency and performance of terrain rendering, the most common used is LOD model, also called multi-resolution model, which is used to represent terrain models at multiple levels of detail. In this way, the complexity of the scene can be reduced and real-time smooth browse can be achieved.

2. RELATED WORK

Although the model representations achieved are not always the same, they can always fall into two types: the triangle-based multi-resolution terrain model[5-9] and tree-based multi-resolution terrain model [10-17].

Several methods use **Delaunay triangulation** to develop multi-resolution hierarchies [5, 6]. In particular, Cohen-Or and Lev-anoni [6] support on-line view-dependent LOD with temporal coherence, but must resort to "two-stage" geomorphs. Klein *et.al* [7] proposed one algorithm with precise error control mechanism, TINS are reconstructed by Delaunay triangulation when the viewpoint changed, so that any local simplification may leads to global triangulation and restrict the rendering efficiency. And now, the main idea is to save the information of *vertex split* and *vertex collapse* at the pre-triangulation stage, in the process of real-time rendering, only use the information saved to avoid the time-consuming work of triangulation. Such as the *Progressive Meshes* algorithm proposed by Hoppe[8]. This algorithm defines a continuous sequence of meshes for increasing accuracy and store an arbitrary mesh as a much coarser base mesh together with a sequence of n detail records, from which approximations of any desired complexity can be efficiently retrieved. Jie Li[9] introduced an algorithm called Sorted Decremental Triangular Mesh (SDTM) ,which is based on three transformations: Edge Collapsing, Surface Flattening and Line Straightening, through each transform can remove a vertex without holes and thus re-triangulation process can be avoid .

Miller[10] uses a quadtree to preprocess a height field defined on a uniform grid. For each frame at run time, a priority queue drives quadtree refinement top-down from the root, thus allowing specified triangle counts to be achieved directly. Taylor and Barrett [11] extract mesh approximations from rectangular **quadtree** hierarchies. Liu[12] presented the VDDMTM (view-dependent dynamic multi-resolution terrain model), the dynamic terrain simplification and the evaluation function are discussed and the bucket sorting method is adopted to repair cracks, but lacking of continuity constraints. Both Lindstrom et al. [13] and Duchaineau et al. [14] define **bintree** hierarchies, based on binary subdivision of right isosceles triangles, and demonstrate real-time view-dependent LOD. Because these representations are based on regular subdivision, they offer concise storage. Duchaineau et al. are able to create optimal approximating meshes through incremental changes at each frame. However, the meshes are only optimal within a restricted space of meshes, since the regular subdivision structure constrains both vertex locations and face connectivities. As a result, the approximations may be far from optimal when one considers the space of all possible triangulations of the domain. Chen Gang[15] concerns both quadtree-based and bintree-based algorithm .[17] depicted a visualization system integrated some key problems of terrain visulization.

Focusing on the LOD control and culling to the view frustum and dealing with in-memory geometry and texture management, Our algorithm, a quadtree-based view-dependent algorithm applying to continuous multi-resolution terrain rendering is proposed , which can maintain the original topological structure without adding additional vertexes and the properties of original vertexes can all be used directly.

3. MESH REPRESENTATION

3.1 Terrain Quadtree

When using quadtree to represent the terrain model, each node in the quadtree covers a rectangular region of the terrain. The root node which covers the whole terrain is the coarsest representation of the terrain, the lower nodes only cover a quarter of their father nodes. For two neighboring layers in the terrain quadtree, the top nodes have lower resolution than the bottom nods and hence the former induce more sampling errors than the later, thus less rendering quality, however, with fewer nodes to be rendered, the efficiency of the former is much higher than the later, which is key point in interactively browsing a terrain. Thus to gain a good balance between the efficiency and quality of terrain rendering, the key problem of quadtree-based terrain rendering is to decide a node set as a representation for the terrain quadtree considering both the static error (view-independent) and dynamic error (view-dependent).

In general, the original DEM data for terrain visualization can be expressed as a 2D matrix *fpDEM*[*nMaxRow*][*nmaxCol*].where *mMaxRow* and *nMaxCol* are the sampling points in the longitude and in latitude separately. When the DEM data are expressed as a terrain quadtree, each layer of the quadtree consists of the nodes gained by equidistantly resampling with a certain distance. For two neighboring layers, the top layer has half the resolution of the bottom layer. And any none-leaf node in DEM quadtree has four son nodes and each son node cover a quarter region of its father nodes.

3.2 Static Errors Evaluation

As the errors of terrain nodes can be defined accordingly to the terrain fluctuation, we give a static error evaluation function as following:

Suppose that $h(x, y)$ is the elevation function and $A(Star(P))$ the average error plane. Node static error can be expressed as formula (1):

$$\varepsilon = \int_{d(c_i)} \left| h(x,y) - A(Star(P)) \right| dxdy \quad (1)$$

Where average error plane is a plane with least-squares fit from all the sampling points .Node static errors are often be computed in the stage of preprocessing and be saved in the *fError* field.

3.3 Dynamic Error Evaluation

In order to establish the direct relation between the viewpoint and the chosen resolution level, thus the farer the distance between the model and the viewpoint, the coarser the representation is used, the nearer, the more complex, we try to represent the importance by distance between the model and the viewpoint. Just as shown in **Fig.1** , suppose the distance between the edge and the viewpoint is ***d***, the length of the project plane is ***L***, α is the field-of-view angle and is constant during the real time rendering, τ (pixels value) is the pixel number of the projected segment of the edge in the screen, and is the length of the line, which is parallel to the screen .

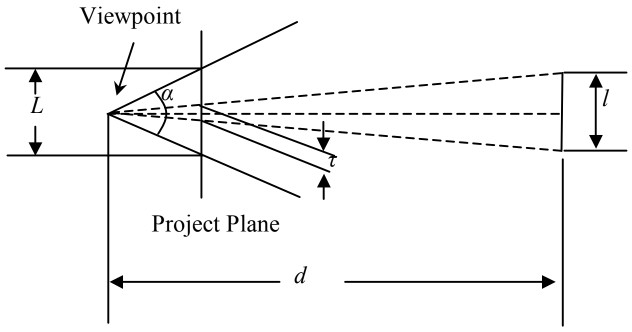


Fig.1 The principle of projective projection

We can easily get the following formula:

$$\tau = \frac{l \times L \times \lambda}{2 \times tg\frac{\alpha}{2} \times d} \quad (2)$$

where λ is the number of pixels per world coordinate unit in the screen coordinate system.

According to formula (2) , τ computed is the maximal length of projected segment of this line for a given ***d*** .To a certain edge, the larger the distance ***d*** is the, the less the τ is. At a given error tolerance, when the distance ***d*** is far enough, the length of its projected segment can be less than τ so that the line can be ignored.

General speaking, there are two key factors that closely related to the dynamic error: the static error and the distance between the node and the viewpoint. Considering all these factors together, the following dynamic error evaluation function is given:

$$e(Ci) \approx \frac{k' \bullet L}{\alpha \bullet D} \quad (3)$$

where *L* is length of the terrain that the node covered, *D* is the distance between the center of the node and the viewpoint, α is field-of-view， and k' can be used as a parameter to dynamically adjust system performance so that a comparatively steady rate is sustained .

4. REPARING CRACKS

4.1 The Cause of Cracks

Using terrain quadtree to serve the purpose of multi-resolution representation, there may exist cracks intrinsically. The cause of cracks can be showed in **Fig.2**.Nodes C_1 and C_3 has lower resolution than its neighboring node C_2, which cause the cracks (terrain region uncovered) exsist. In **Fig.2**，There is crack lies between nodes C_1 and C_2, so to C_1 and C_3.

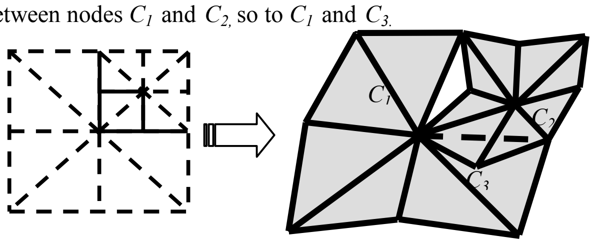


Fig.2 the cause of cracks

4.2 Repairing Cracks

The most common used method to repair cracks is restricted quadtree method **[10]**, which demands that the resolution between one node and its neighboring nodes be restricted to at most one layer. Zhanping Liu[3] proposed one bucket sorting method to repair cracks, but fails to keep the topological structure of cracks regions. Cheng Gang[4] adopt one method that keeps the right topological structure with the dependency of vertexes. There are some still other methods that adding additional triangles to cover the cracks, which may cause discontinuity. We proposed a new method to repair cracks in this section which can be expressed as following:

For the sake of convenience, some basic concepts are presented here. And any node mentioned here belongs to the set of the sampling vertexes in the original terrain model.

Definition 1: the son nodes of each none-leaf node in DEM quadtree are named as **Fig.6**，and the upper, the lower, the left, the right are defined as the regions covered by nodes [3，2]、[0，1]、[3，0]、[1，2] separately.

Definition 2: As to each node select into the node set, its diagonal triangulation refers to line from one corner to the center of its parent node, as shown in **Fig.3.**

Definition 3: The triangulation of quadtree consists of two patterns: directly diagonal triangulation (in **Fig.4 (a)**) and center-holding triangulation (in **Fig.4 (b)**).

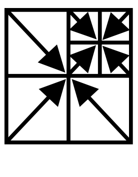


Fig.3 diagonal triangulation

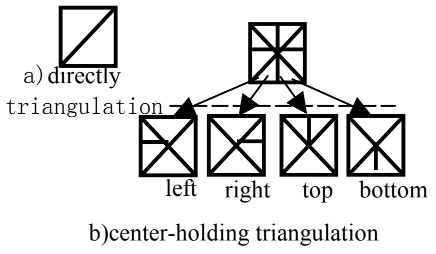


Fig.4 quadtree triangulation

The following is the method to repair cracks :

1) If the nodes selected have higher resolution than its neighboring nodes or with the same , the directly diagonal triangulation is executed.

2)If the nodes selected have lower resolution than its neighboring nodes ,at first, we executes the diagonal triangulation, and then judge whether the son nodes satisfy 1)，this process only involves those son nodes near their higher resolution neighboring nodes. If not, the iterate process will continue until 1) is satisfied.

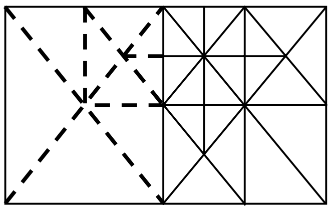


Fig.5 repairing cracks between the left and the right

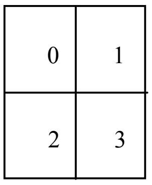


Fig.6 the index of son nodes

For example, as **Fig.5**，left node（pTreeNode1）has lower resolution than the right neighboring node（pTreeNode2）, the dashed is the lines should be added.

The follwing is the iterative process:

```
BEGIN:
SeamErase（pTreeNode1，pTreeNode2） {
pTreeNode1—>bRight EQ TRUE
IF  pTreeNode1 lies in the 1 region of parent node
        pTreeNode1—>pParent->bTop=TRUE;
ELSE  IF  pTreeNode1lies in the 2 region of parent node
        pTreeNode1—>pParent->bBottom=TRUE;
SeamErase （ pTreeNode1->pSon[1], pTreeNode2->pSon[0]）
SeamErase （ pTreeNode1->pSon[3], pTreeNode2->pSon[2]）
}
END
```

The other relationship between one node and his neighborings can be delt with just as the same.

5. REAL-TIME TERRAIN RENDERING

Dynamic nodes selection is the process to choose an appropriate node set *S* as a representation from the terrain quadtree so that all the nodes in set S should not only cover the whole terrain but satisfy the given error threshold, furthermore, any two nodes in set *S* should not intersect.

In real-time rendering, the key problem is to maintain a relatively steady browsing rate. Thus it's necessary to use the culling to view frustum, which is a dynamic process of choosing only those tree nodes visible to be rendered with the changes of viewpoint and moving directions. This may reduce the triangles to be rendered further.

Fig.7 illustrates the changing regions applied to the active mesh as a user moves forward and to the left through a model. Regions of the model entering the view frustum (on the left an upper) are instantaneously choosed to be rendered, while regions leaving the view frustum (on the right and near the viewer) are instantaneously out of rendering. **Fig.8** shows the culling to view frustum, the regions involved in the two blue lines are visible in the frustum, and the details of regions far from the viewpoint are instantaneous coarsening.

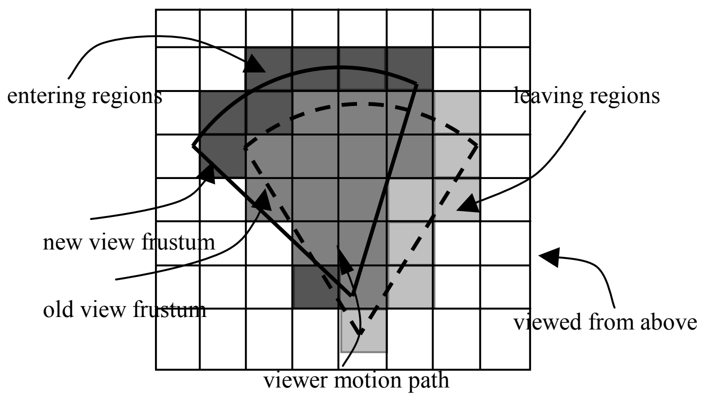


Fig.7 The changes of viewpoint

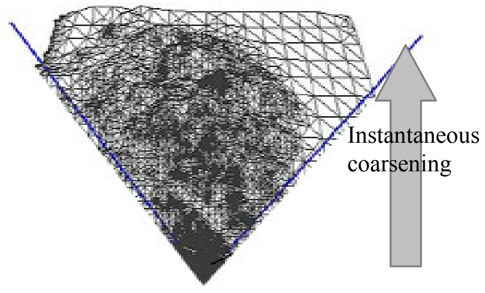


Fig.8 the example of culling to view frustum

In real-time browsing, this is an iterative process and the nodes are adapted dynamically with the changes of error threshold, viewpoint and view direction.

6. SCENE MANAGEMENT AND UPDATING

6.1 Storage and Retrieval

However, the triangulation model described above can't satisfy the demand for large-scale terrain visualization. As to very large terrain database, The input terrain data and texture data are partitioned into blocks of $(2^k+1) \times (2^k+1)$ grid points respectively. This partitioning provides efficient spatial selectivity and physical clustering on external storage. To accelerate the process of data initialization, the errors of each terrain nodes and the normals of each grid points are pre-processed. When stored, all blocks are stored in its corresponding files and data in each file are stored as a quadtree node, furthermore, its child nodes may be set to null for some less fluctuant area. In this way, the terrain data in node can be easily retrieved and the coordinates, normal vector and texture coordinates of grid points can all be used directly, which deserve the demand for some more space for storage.

6.2 Dynamic Scene Updating

To solve the problem of visualizing very large terrain database, a window paging concept as shown in Figure 11 is introduced because we can't load the whole terrain data in main memory and only some blocks of the whole terrain data is necessary to a certain frame. As shown is Figure 9, our visible scenes always represents a window onto the real world in accordance with the current viewpoint and view parameters. Assume that the whole blocks we considered is $n \times n$ and the page of each frame $m \times m(m<<n)$, and the viewpoint is always at the center of the window page. As depicted in Figure 11, with the viewpoint moving, only a few patches need to be updated. Furthermore, the window page is not updated for every small variation of view position and view frustum parameters, an update only occurs when the variation exceeds a specific threshold. This help to reduce the data management costs significantly severe loss of display quality. The process of updating can be achieved with two work threads, one thread for scene updating and the other for displaying, especially for multi-processor computer.

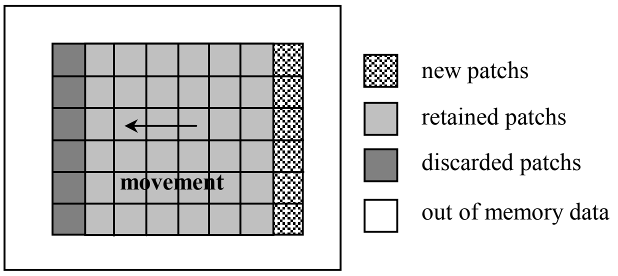


Fig.9 dynamic scene update

7. EXPERIMENTS

With the above algorithm proposed, the browsing of large-scale terrain with high rate is achieved. **Fig 10** illustrates the effect of texture terrain visualization, (a) is the textured full-resolution terrain model, (b) is the simplified multi-resolution model and (c) the simplified model with texture mapping. Compared with (a), even with error threshold of 5 meters, the sense of realism and significant visual details are retained. Thus texturing may improve the visual impression of the terrain and allow higher error tolerance. **Fig.11** illustrates the multi-resolution representations with different error threshold (10,5,2 and 1 meter), with the error threshold decreasing, the terrain are rendered with more and more details. **Fig.12** illustrates the effects of view-dependent LOD control, where the center of the picture is the current view position, the view direction is directly to the north and the viewer motion path is from left to right, the rendering regions with the highest resolution are always at the view position.

We also performed some rendering and scene updating tests based the platform of Windows 2000, the hardware is PII400 with Oxygen GVX1 card. Our whole test scene was a mountainous terrain consising of 48 × 87 patchs, each 6.5km×6.5km large and the always in-memory blocks is 9*9. **Fig 13** shows the scene map. **Table 1** shows the performance of flying through the terrain. **Fig 14** is one frame at interactive frame rate.

TABLE 1 Frame rates comparision

nuber of triangles	5274	10275	15783	20487	43784
frames per second	86	45.1	35.7	22.1	14.1

8. CONCLUSION

The real-time rendering of large-scale surface is one of the most challenging problem in terrain visualization. We have proposed an optimization quadtree-based algorithm for dynamic generation continuous levels of a given complex landscape in real time. The experimental results shows that if the rendered triangulation can be constricted to 1000~2000 and the using of texture mapping, the algorithm can provided a high frame rates maintaining high image quality. Even under the ordinary circumstance, real-time dynamic browsing can be attained.

Our future work will focus on the management of geometry and texture data and the dynamic data-loading from disk interactively. Adapting our algorithm to very large terrain simulation, such as global terrain data is still a challenging problem.

REFERENCES

[1]Teller,S.,Manocha,D.,Hudson.T. et al., Visibility culling using hierarchical occlusion maps *Computer Graphics* 1997,31(3):77

[2]Wang,Y.G.,Bao,H.J.,Peng,Q.S., Accelerated walkthroughs of virtual environments based on visibility preprocessing and simplification. *Computer Graphics Forum*,1998,17(3):187

[3] Sun Hongmei, Ren Jicheng. et.al. Implementing interaction in image-based rendering. *The sixth International Conference on Computer Aided Design and Computer Graphics*, December 1-3,1999,Shanghai, China.

[4]Chen,S..E,QuickTime VR: an imaged-based approach to virtual environment navigation.*Computer Graphics*,1995,29(4):29

[5]Cignoni, P..Puppo,E.. et.al. Representation and visualization of terrain surfaces at variable resolution. *The Visual Computer* 13 (1997),pp.199-217.

[6]Cohen-Or,D.. and Levanoni,Y. Temporal continuity of levels of detail in Delaunay triangulated terrain. In *Visualization'96 Proceedings*(1996),IEEE,pp.37-42.

[7] Reinhard Klein, Gunther Liebich, and W.Straser. Mesh reduction with error control. In *Proceedings of Visualization'96*(1996),pp.311-318.

[8] Hoppe H. Progressive mesh. Computer Graphics *.SIGGRAPH'96 Processing.*

[9] Li Jie, Tang Zesheng. Real-time,continuous level of detail rendering for 3D comples models. In *Proceedings of CAD & Graphics'97*. Fifth international conference on CAD &CG December 2-5,1997,Shenzhen,China.

[10]Mark C. Miller. Multiscale compression of digital terrain data to meet real time rendering rate constraints. PhD thesis, University of California, Davis,1995.

[11] Taylor,D.C and Barrett,W.A. An algorithm for continuous resolution polygonalizations of a discrete surface. In *Proceedings of Graphics Interface'94*(1994),pp.33-42.

[12] Liu Zhanping, Wang Hongwu, et.al. VDDMTM: a view-dependent dynamic multi-resolution terrain model. *The sixth international conference on Computer Aided Design and Computer Graphics*, December 1-3,1999,Shanghai,China.

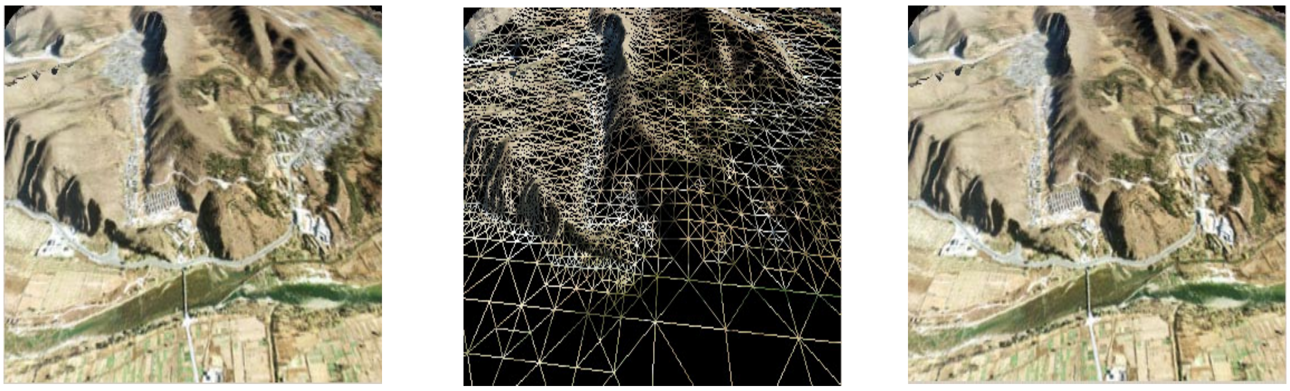
[13] P.Lindstrom, D.Koller, W.Ribarsky,L.F.Hodges,N.Faust,and G.A.Turner. Real-time,continuous level of detail rendering of height fields.In *Proceeding SIGGRAPH 96*,P:108-118.ACM SIGGRAPH.

[14] Mark Duchaineau, Murray Wolinsky *et al*, ROAM Terrain:Real-Time Optimally Adapting Meshes. In Visualization'97 Proceedings(1997),IEEE,pp.81-88.

[15] Chen Gang. A Research on Multi-resolution Surface Description and Real-time Rendering for Virtual Terrain Environment. PhD thesis, Institute of Surveying and Mapping, 2000, Zhengzhou,China.

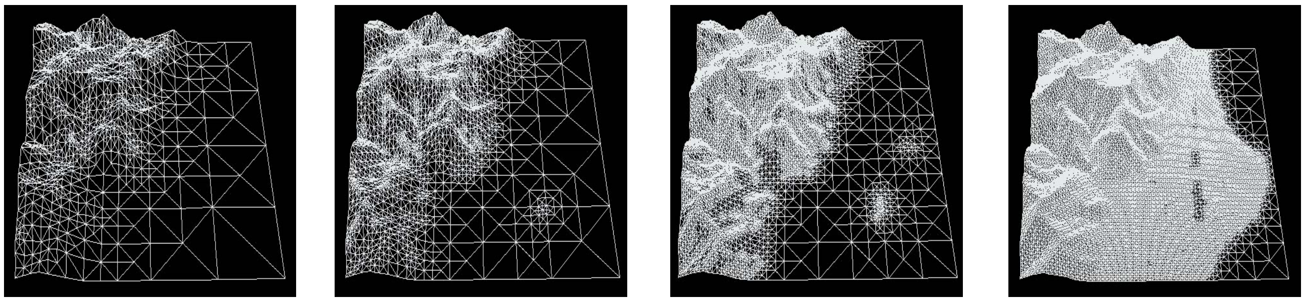
[16]Hoppe H. Smooth view_dependent level_of_detail control and its application to terrain rendering. http://research.microsoft.com/～hoppe.

[17] Renato Pajarola, ETH Zurich. Large scale terrain visualization using the restricted quadtree triangulation. http://www.vterrain.org (accessed 18 Mar. 2001)



a) full-resolution model (textured) b) multi -resolution model (grid) c) multi-resolution model (textured)

Fig 10 textured terrain visualization



a) 10 meter max. error 13% b) 5 meter max. error 27% c) 2 meter max. error 47% d) 1 meter max. error 86%

Fig 11 triangulated surface with different error threshold

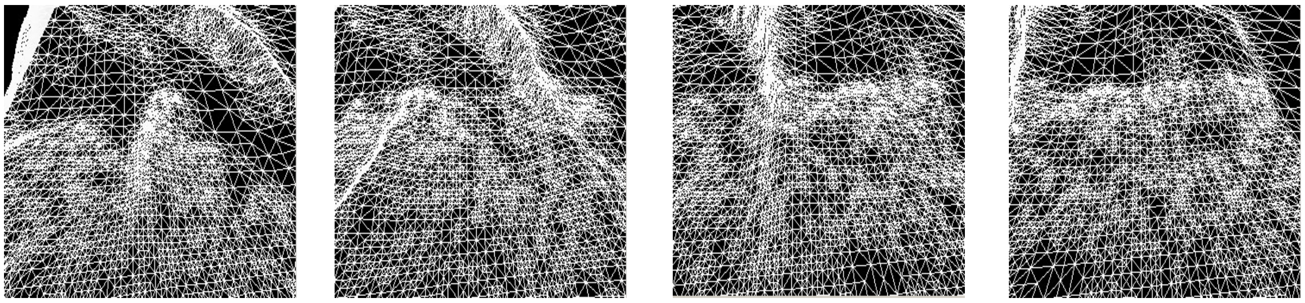


Fig 12 moving of view position (from left to right)

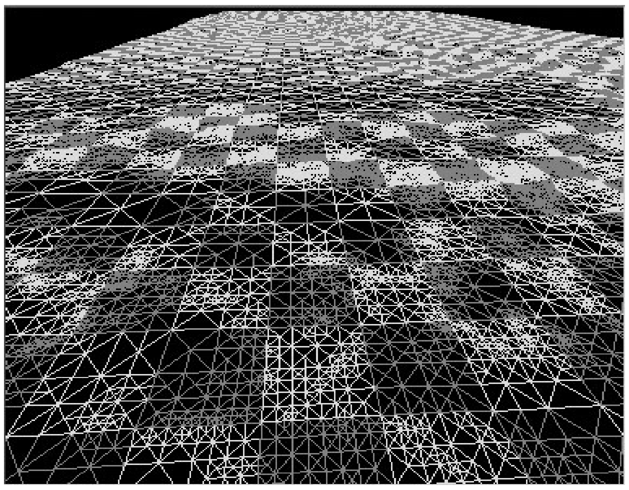


Fig 13 scene map

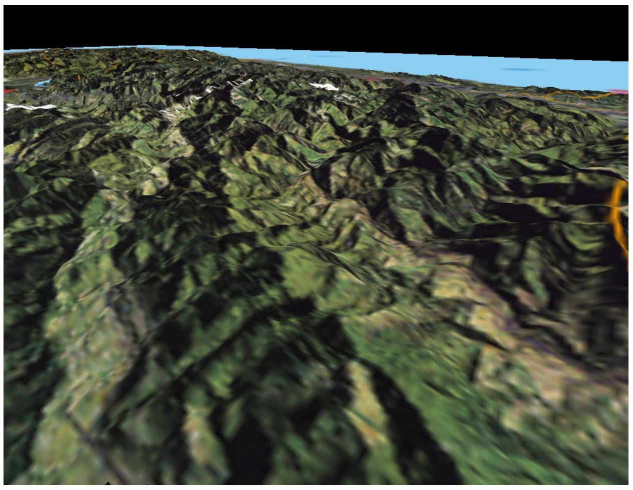


Fig 14 one frame of flying through (num:15783 FPS:35.7)