SECONDARY STORAGE TERRAIN VISUALIZATION IN A CLIENT-SERVER ENVIRONMENT: A SURVEY

Kai Xu and Xiaofang Zhou
School of Information Technology and Electrical Engineering
The University of Queensland, Brisbane, QLD 4072
Australia

Abstract

With the constant increase in the size of terrain data, it is becoming less feasible to have all the data in main memory when performing visualization. The data exchange between main memory and secondary storage becomes a bottle-neck in terrain visualization. The slow-down of the visualization performance is even more obvious when visualizing terrain in a client-server environment because of the long response time caused by transferring large amount of data over the network. This paper surveys recent secondary storage terrain visualization techniques in a client-server environment in recent years, with a particular focus on reducing data exchange between main memory and secondary storage, and between the client and server using spatial indexes with Multiresolution Terrain Model (MTM). Previous works are compared according to their ability to reduce data retrieval, reduce network transmission, data caching and support for different data structures. Research problems that require further investigation are identified and discussed.

Key Words: Distributed, multiresolution, secondary storage, visualization

1. Introduction

When terrain data covering very large area becomes available, it raises many issues for visualization. As the size of terrain data increases, it is increasingly difficult to have all the data in main memory during visualization. The data exchange between main memory and secondary storage becomes a bottle-neck in terrain visualization. The increase of the data size also requires more processing power for visualization. Even state-of-the-art workstations cannot handle it comfortably if a large terrain is visualized at full resolution. Also this may not be necessary due to a limited resolution of the client-side display device. A Multiresolution Terrain Model (MTM) is proposed for this problem. The MTM captures a wide range of terrain approximations (meshes) from an original terrain data set. Each approximation represents the data at a different resolution and can be used to reconstruct the terrain for different viewing requirements. The mesh reconstructed from the MTM can be restricted within the area that is visible and the resolution of the mesh can vary according to the specification of the display device. Thus it can greatly reduce the amount of data retrieval from secondary storage and improve the performance of visualization.

Once the MTM is constructed, the method to extract an approximation mesh for user-specified view conditions and update the mesh when the view conditions change are important research problems of terrain visualization. Many visualization algorithms have been proposed for datasets small enough to fit into the main memory. However these methods do not work properly with secondary-storage data because of the much slower access speed of secondary storage compared to speed of the main memory.

With the wide application of the internet and client-server architecture, there is an increasing need to visualize the terrain data over network. Given the large size of terrain data, reducing the network transmission during terrain visualization raises an important research problem.

We present a survey on secondary-storage terrain visualization in a client-server environment in this paper. Some challenging problems that need further investigation in this area are also identified and discussed. The remainder of this paper is organized as follows. In section 2, we give a brief introduction of terrain visualization and main-memory visualization algorithms. In section 3, the research problems in secondary-storage terrain visualization in a client-server environment are discussed and representative solutions proposed by previous research are compared. Section 4 concludes the paper and outlines problems that need further attention.

2. Terrain visualization

As mentioned in the first section, there are two important operations in terrain visualization: one is to extract an approximation mesh according to user-specified view conditions, which is called selective refinement; the other is to update the mesh after a user changes the view conditions. These visualization operations can be specified by two conditions [1]:

1. Region of Interest (ROI) condition, which defines the part of a terrain relevant to the visualization, i.e., the part of a terrain that the user will see on the screen.
2. Level Of Detail (LOD) condition, which defines the required resolution of the mesh to be extracted. Fig.1

gives an example explaining how the LOD condition works.

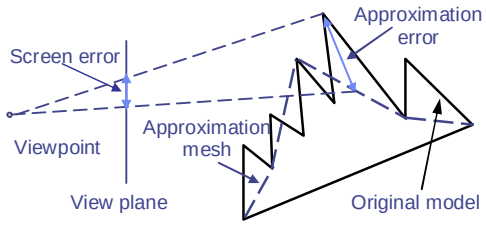

Fig.1 LOD condition

Suppose we have an original terrain model, its approximation mesh, and the approximation error of a vertex (as shown in Fig. 1). This error, however, is not the approximation error that will appear on the screen, which is called screen error. In the virtual space that contains the terrain model, there is a viewpoint and a view plane. The image that users see on the screen is the projection of the terrain model on the view plane according to the viewpoint. Subsequently the screen error that the users finally see is the projection of the approximation error on the view plane.

For selective refinement, a user specifies both the part of terrain they want to view and a maximum allowable screen error. These information together with the information from the internal rendering system (such as the position and orientation of viewpoint and view plane) formulates the ROI and LOD conditions for the selective refinement, which then extracts a mesh from MTM based on these parameters.

After a terrain model is rendered the user often navigates (pan, zoom or rotate) in the virtual space containing the model. These operations will change the ROI and/or LOD condition. For instance, the distance between the terrain model and the viewpoint decreases when user zooms in, the screen error (the projection of the actual approximation error) increases accordingly (i.e. the LOD condition changes), which could become larger than the maximum allowable screen error specified by the user. If this happens, current mesh should be replaced by a more detailed one. However, it may not be necessary to build an entirely new mesh from scratch if part(s) of the new mesh remained unchanged from current mesh. To this end an efficient visualization algorithm needs only to updates the necessary parts of current scene.

Many algorithms have been proposed to support selective refinement and navigation when terrain data can fit into main memory (see [2] for a survey). Most of these methods start with a coarse approximation of the terrain part that covers the ROI and then refine it gradually until the mesh satisfies the LOD condition (i.e., the screen error of every part in the mesh is no more than the maximum screen error specified by the user). The resulting mesh is then used for rendering.

3. Secondary storage terrain visualization in a client-server environment

In a client-server environment, the visualization of terrain data has three main steps:
1. Client sends a request to server and terrain data necessary for visualization is prepared at the server.
2. The data is transferred to the client through network.
3. The client receives the data and displays it.

Two different architectures can be used to implement this process. In the first architecture the server performs all the visualization computation according to the client request and sends the final results (approximation mesh) back to the client. The alternative has the server sending part of the MTM necessary for visualization to the client and all the computation is done at the client. The advantages of the second architecture are:
1. Server would be able to support more users. The computation involved in terrain visualization, such as selective refinement, is very time-consuming. The server would support a very limited number of users if all computation needs to be done at the server end. Distributed computing also complies with the principle of the client-server architecture.
2. The client doesn't have to wait for all the data before it can start displaying. As long as there is enough data for a rough approximation, it can be displayed at the client side. The mesh is then refined gradually as more data is received.

When comparing different methods, we assume they use the second method if not specified.

The key factor of improving the visualization performance is to reduce the amount of data exchange, both the data exchange between the main memory and secondary storage at the server, and the data exchange between the client and server over the network.

Another factor of performance improvement is to reduce the number of data exchange while transferring same amount of data. For instance, the main part of the selective refinement performed at the client is to refine the coarse mesh according to the parameters specified by the user. During this process, every time a part of the mesh needs to be refined, more detailed information about this part of terrain is needed. If not available at the client, the data has to be transferred from the server, which could require a large number of network transmissions and server I/O operations. It is desirable that all the data is transferred together.

When surveying the recent works, we compare them on four aspects:
1. ROI support. If a method can provide efficient ROI support, the data exchange can be restricted within the ROI, which means only data that is potentially visible is retrieved and transferred.
2. LOD support. A MTM has multiple descriptions (LODs) of the same terrain. With LOD support, only one LOD that is suitable for the current view conditions

support can greatly reduce the amount of data exchange.
3. Navigation support. User's navigation changes visualization parameters and the approximation mesh needs to change accordingly. With navigation support, data exchange only occurs when part of the new mesh is not available at the client. This is actually the data caching ability of the method.
4. MTM support. MTM support is the ability of the method to support different types of MTMs. It is desirable that a method can support different types of MTMs.

Generally, there are two types of MTMs according to their data structure: RSG (Regular Square Grid) and TIN (Triangulated Irregular Network). Because of the difference in the data structure, their visualization methods are quite different as well.

3.1 RSG-based MTM visualization

RSG-based MTMs [3-6] have been used for terrain visualization for quite some time. The majority of these methods are based on restricted quadtrees [7], which is a variation of the quadtree. Instead of grid, restricted quadtree uses right triangle to describe the terrain surface. The close relation to the quadtree gives them a build-in quadtree-like index and provides them with good ROI support. The ROI can be represented using the quadtree and data retrieval can be restricted within these triangular grids. These methods also provide good navigation support: the data needed for navigation update can be easily located with quadtree index and only the triangular grids that change during the navigation need to be updated.

The contribution of [3] is an efficient screen-space error metric calculation, and scene culling and vertex selection according to this error metric. The method proposed in [4] uses a priority-queue driven mesh refinement and a combination of object-space and screen-space as error metric. In [5], efficient rendering is achieved by organizing the terrain's constituent triangles into a triangle strip that follows the Hamilton space-filling curve. In [6], a hybrid data structure is proposed that can incorporate a TIN as part of the RSG data structure thus allowing some irregular or important parts of the terrain to be described properly with the triangles. The method proposed in [8] uses a technique similar to the Z-order space-filling curve. This allows data close together in space to be stored near each other, which improves the efficiency of reading data when performing the top-down mesh refinement. The work proposed in [9] focuses on the terrain visualization in a client-server environment and the data structure used for the server is slightly different from the data structure for the client: a quadtree is used for the server for storage and retrieval while a binary triangle tree is used for the client for efficient rendering of the terrain data. Another work [10] based on the binary tree uses a more compact data representation to reduce data retrieval and transmission.

Because the quadtree index these methods use is a good way to represent locality information but not LOD information, a common problem of these methods is that they provide very limited support for the LOD condition. Furthermore these methods are based on the RSG, which means they cannot support TIN-based MTMs.

3.2 TIN-based MTM visualization

There are mainly three types of methods for using TIN-based MTMs, which are quite different from the methods for RSG-based MTMs.

The first type of methods [11] [12] use the hierarchy of the MTM itself as a spatial index. In these methods the selective refinement is performed at the server and then the extracted mesh is sent to the client for visualization. In [11] a compact data structure is proposed to reduce data transmission when sending the approximation mesh to the client. This work is continued in [12] and the compact data structure is further improved.

The second type of methods uses spatial index that is a by-product during the process of MTM construction [13] [14]: because of the large size, the whole dataset cannot be loaded into main memory altogether to perform the simplification. The simplification algorithm then divides the whole dataset into smaller parts so that each part can fit into main memory and be simplified separately. As a result, the MTM is made up of a number of smaller MTMs that can be displayed independently or together with each other. The visualization algorithm then retrieves only the parts that intersect with the ROI from secondary storage.

Magillo [13] presented a method based on Multi-Triangulation (MT) [15], a general framework of MTMs. An MT is a triangulation hierarchy. Each node in the MT is a triangulation where the root is an initial coarse approximation, and each child node refines part of triangulation of its parent node. These nodes are connected to each other according to the triangulation refinement (Fig. 2).

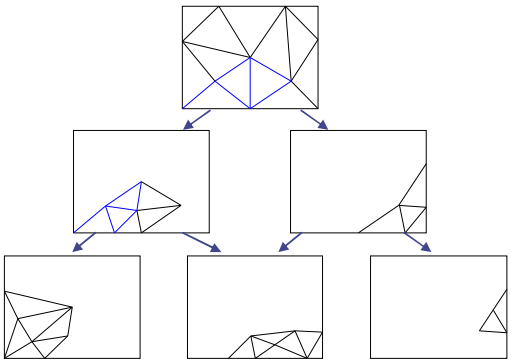

Fig.2 Multi-Triangulation

To manage large terrain data, this method partitions a large MT (called global MT) into smaller ones (called local MT). Every local MT is built separately but still keeps their boundary compatible to each other at different LODs. The resulting local MTs can have boundary shapes other than a rectangle. To visualize the terrain, several local MTs are retrieved and merged into a single MT (Fig. 3).

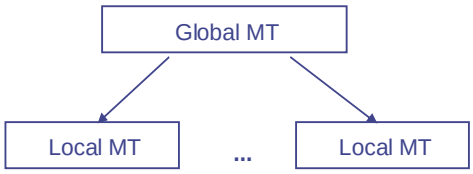

Fig.3 Global MT and Local MT

One alternative, the Progressive Mesh (PM) [16], is a multiresolution model built from a high-resolution triangulation through iterative edge-collapses. Hoppe [14] proposes a method to fit a progressive mesh in a quadtree. The model is built starting from the leaf level by applying edge-collapses in each quadtree node independently. Next, every four adjacent nodes are merged and the process is iterated. In order to avoid collapses occurring across different quadtree nodes, the simplification process inside each node is forced to preserve the boundary vertices (Fig.4).

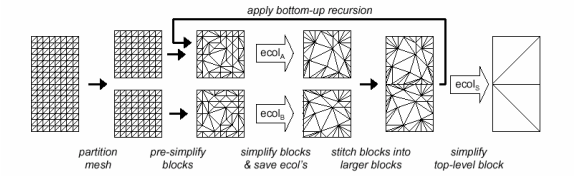

Fig.4 Progressive mesh with a quadtree

Because these methods are simplification oriented, the partition is not accurate enough for visualization purposes. So the support for ROI is not very good. These methods do not provide good support of LOD condition either. In [13] all different LODs of the same area are stored collectively in one local-MT, which means users have to retrieve entire local-MT even when they just need one LOD. This also raises the problem for navigation support: if the data needed for update is stored in another “local MT”, that “local MT” has to be transferred to the client, which could cause substantial network traffic. In [14] different LODs can be accessed individually, but the structure is fixed once the simplification is done. If any change is needed, the MTM needs to be reconstructed, which is very time consuming.

The third type of method uses a separate spatial index designed specifically for visualization purposes. “Constrained Implicit TIN” [17] is proposed to incorporate terrain features (such as road, river) with multiresolution terrain data. Terrain data is stored using a constrained Delaunay triangulation [18], and the feature data is stored using a line generalization tree [19]. Both data are indexed with a quadtree similar to PMR-quadtree [20]. During the reconstruction of the approximation mesh, the terrain surface is extracted first. Then the feature data is inserted into the terrain (Fig. 5).

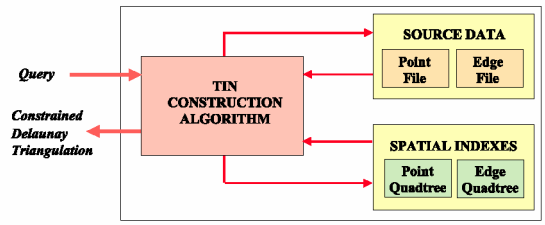

Fig.5 Constrained implicit TIN

This method is designed specifically for constrained Delaunay triangulations, which has a clear level structure. The problem arises because there is no such level structure in most MTMs and it cannot be extended easily to work with other MTMs.

4. Conclusion

A summary of all the methods reviewed is listed in Table 1.

Spatial index	ROI support	LOD support	Navigation support	MTM support
Restricted quadtree	Good	Limited	Good	RSG-based
MTM hierarchy	No	No	No	TIN-based
Simplification index	Limited	Limited	Limited	TIN-based
Constrained implicit TIN	Good	Good	Good	Constrained Delaunay triangulation

Table 1 Comparison of visualization methods

Methods that are based on restricted quadtrees can provide good support for ROI and navigation because of the quadtree index. However, these methods do not support LOD very well and can only support RSG-based MTMs. Methods which use the MTM hierarchy as a spatial index performs all visualization computation at server side. And they do not provide support for ROI, LOD or navigation function for client-server environment. Methods that use a spatial index created during the simplification process do not work very well either. Because these methods are simplification oriented, hence limits their ability for visualization. Constrained implicit TIN uses an extra spatial index designed specifically for visualization, which gives good support for ROI, LOD and navigation. The only problem is that it can only work with layer MTMs.

A lot of research has been done for RSG-based MTM visualization, but more work is needed to improve the support for the LOD condition. It would be ideal if these methods can also be extended to support TIN-based MTMs. There is relatively less work for TIN-based MTMs. So far none of them can provide support for all the functions needed in terrain visualization (ROI, LOD and navigation) and support different types of TIN-based MTMs at the same time. Research in this area is still in its early stage and there is much room for improvement.

5. Acknowledgement

The authors would like to thank Mincom Pty Ltd for providing access to their terrain visualization and analysis products and data. We also acknowledge Dan Sketcher who has worked on part of this as his Honors project.

References

[1] De Floriani, L., P. Magillo, and E. Puppo, VARIANT: a system for terrain modeling at variable resolution, *GeoInformatica*, *4*(3), 2000, 287-315.
[2] De Floriani, L., P. Magillo, and E. Puppo, Efficient implementation of multi-triangulations, *Proc. IEEE Visualization '98*, Research Triangle Park, NC, USA, 1998, 576
[3] Lindstrom, P., et al., Real-time, continuous level of detail rendering of height fields, *Proc. 23rd International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'96)*, New Orleans, LA, USA, 1996, 528
[4] Duchaineau, M., et al., ROAMing terrain: Real-time Optimally Adapting Meshes, *Proc. IEEE Visualization '97*, Phoenix, AZ, USA, 1997, 585
[5] Pajarola, R., Large scale terrain visualization using the restricted quadtree triangulation, *Proc. IEEE Visualization '98*, Research Triangle Park, NC, USA, 1998, 576
[6] Baumann, K., et al., A hybrid, hierarchical data structure for real-time terrain visualization, *Proc. Computer Graphics International 1999*, Silicon Graphics Int, 1999, xi+249
[7] Von Herzen, B. and A. Barr, Accurate Triangulations of deformed, intersecting surfaces, *Computer Graphics*, *21*(4), 1987, 103-110.
[8] Lindstrom, P. and V. Pascucci, Visualization of Large Terrains Made Easy, *Proc. IEEE Visualization*, San Diego, California, 2001, To appear
[9] Aasgaard, R. and T. Sevaldrud, Distributed Handling of Level of Detail Surfaces with Binary Triangle Trees, *Proc. ScanGIS'2001*, 2001, 45-58
[10] Gerstner, T., Multiresolution Visualization and Compression of Global Topographic Data, *GeoInformatica, to appear*, 2001.
[11] De Floriani, L., et al., Dynamic view-dependent multiresolution on a client-server architecture, *Computer Aided Design*, *32*(13), 2000, 805-823.
[12] Danovaro, E., et al., Compressing multiresolution triangle meshes, *Proc. Advances in Spatial and Temporal Databases. 7th International Symposium SSTD 2001*, 2001, xi+542
[13] Magillo, P. and V. Bertocci, Managing large terrain data sets with a multiresolution structure, *Proc. 11th International Workshop on Database and Expert Systems Applications.*, 2000, xxvii+1164
[14] Hoppe, H., Smooth view-dependent level-of-detail control and its application to terrain rendering, *Proc. IEEE Visualization '98*, Research Triangle Park, NC, USA, 1998, 576
[15] Puppo, E., Variable resolution terrain surfaces, *Proc. 8th Canadian Conference on Computational Geometry. CCCG '96.*, 1996, 344
[16] Hoppe, H., Progressive meshes, *Proc. 23rd International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'96)*, New Orleans, LA, USA, 1996, 528
[17] Kidner, D.B., et al., Multiscale terrain and Topographic Modelling with the Implicit TIN, *Transactions in GIS*, *4*(4), 2000, 379 - 408.
[18] De Floriani, L., A pyramidal data structure for triangle-based surface description, *IEEE Computer Graphics and Applications*, *9*(2), 1989, 67-78.
[19] Jones, C.B. and I.M. Abraham, Line Generalisation in a global cartographic database, *Cartographica*, *24*1987, 32 - 45.
[20] Nelson, R.C. and H. Samet, A consistent hierarchical representation for vector data, *Computer Graphics*, *20*(4), 1986, 197-206.