

Multiresolution Terrain Database Visualization

Kai Xu

School of Information Technology and Electrical Engineering
The University of Queensland, Brisbane, QLD 4072, Australia
kaixu@itee.uq.edu.au

Abstract. Digital terrain visualization has received increasing research interest in recent years. One main problem focus has been the large size of terrain data, which needs to be stored on secondary storage hence causing a performance slow-down in terrain visualization. This paper surveys secondary storage terrain visualization techniques with a particular focus on spatial database systems supporting terrain visualization using TIN. Problems that need further investigation in this area are discussed and the approach to improve secondary-storage terrain visualization performance is proposed.

1. Introduction

The visualization of large-scale terrain data has attracted growing interest in the last a few years. Because of the large size of most terrain data, computations and I/O operations involved in terrain visualization are so intensive that sometime even state-of-the-art workstations cannot handle it comfortably. To achieve acceptable running times, we need to make a tradeoff between accurate data representation and performance. Concerning the former, one can use a simpler approximation instead of the highly detailed terrain model. It may be desirable to use a model that captures very fine surface detail when creating the datasets. However not all applications will require such high details.

A *multiresolution terrain model* (MTM) is a model representation that captures a wide range of approximations of a terrain and which can be used to reconstruct any one of them on demand [1]. With MTM, we can reconstruct different terrain approximations for different view conditions or requirements.

The simplest MTM is a set of discrete terrain approximations (also called a *mesh*) with different *levels of detail* (LODs). The problem with this type of MTM is that it only supports uniform LOD, which means the LOD of different parts of a terrain surface must be the same. To illustrate consider a viewpoint that is positioned just above the terrain surface, looking out towards the horizon. The mesh used for this scene covers a quite large area. If a high LOD is used, detail would be unnecessarily complex on the distant part of terrain, because the screen size of those triangles would become smaller than one pixel. The total number of the triangles in such a mesh would be too many for the rendering system to handle. On the other hand, if a mesh with a low LOD is used the visual quality of terrain close to the viewpoint would be highly degraded. A number of MTMs that can change the LOD of the mesh according

to the viewing conditions, supporting *adaptive LODs*, are proposed (see [2] for a survey), i.e., mesh with higher LOD is used for the part close to the viewpoint, whereas mesh with lower LOD is used for the distant part of the terrain.

There exist many simplification algorithms to construct the MTMs that support adaptive LOD. Works surveyed in [2] support terrain data only, while others support more general surfaces (see [1, 3, 4] for a survey). Once these MTMs are constructed, how to extract a mesh for certain view conditions and how to update the mesh when the view conditions change are an important research problem of terrain visualization. Because of its large size, terrain data needs to be stored on secondary storage with access and retrieval speeds much slower than main memory. Secondary storage based terrain visualization brings many issues. We present a survey on secondary-storage terrain visualization 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 to the problem of terrain visualization. The problem of secondary-storage based terrain visualization and representative solutions proposed by previous research are discussed in Section 3. Applications of spatial database systems for terrain visualization are reviewed in Section 4. In Section 5, problems that need further attention are identified. New approach for the problem is proposed in Section 6.

2. Terrain visualization

There are two important operations in terrain visualization: one is to extract a mesh with adaptive LOD for certain view conditions, which is called *selective refinement*; the other is to examine the terrain from different angles or focus on different part of the terrain, which is known as *navigation*. In general, these visualization operations can be specified by two conditions [5]:

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 level of detail of the mesh to be extracted. Here is an example (Fig. 1) explains how LOD condition works.

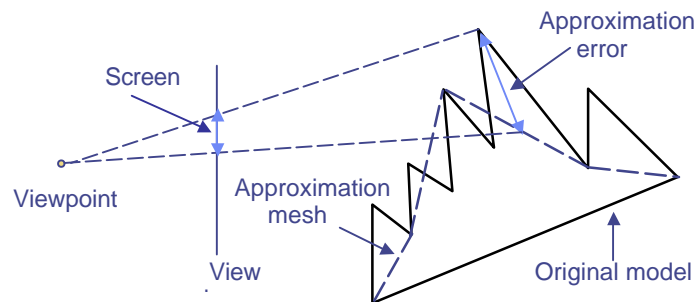


Fig. 1. LOD condition

Suppose we have an original terrain model, its approximation mesh, and the approximation error of a vertex (as shown in the 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, user specifies both part of the terrain they want to view and a maximum allowable screen error. This information together with the information from the internal rendering system (such as the position of viewpoint and view plane) formulates the ROI and LOD conditions for the selective refinement, which extracts a mesh from MTM based on these parameters.

After a terrain model is rendered the user often navigates (pan, zoom or rotate) through the virtual space containing the model. These operations will change the ROI or LOD condition, or both. However, it may not be necessary to build a new mesh entirely if part(s) of the new mesh remained unchanged from the previous mesh. Hence the mesh only needs to be updated for the difference between the old and new scene.

Many algorithms have been proposed to support selective refinement and navigation when terrain data can fit into main memory (see [6] for a survey). Most of these methods start with a coarse approximation of terrain parts that satisfy the ROI condition (i.e., within certain area) and refine it gradually until the mesh satisfies the LOD condition (i.e., the screen error of every triangle in the mesh is no more than the maximum screen error specified by the user). The resulting mesh is then used for rendering. However, relying solely on main memory for this is unrealistic.

3. Secondary storage visualization

Given its characteristically large size, it is difficult to have all the terrain data in main memory for visualization. Thus methods to support visualization of terrain data on secondary storage are needed.

The main part of most main memory visualization algorithms is to refine the mesh according to the LOD condition. During this process, every time part of the mesh needs to be refined additional information must be retrieved. While good in main memory, visualization performance cannot be duplicated efficiently with secondary storage, which requires too many I/O operations. Generally, there are two ways to store terrain data on secondary storage: file or spatial database. Disk paging techniques are proposed for terrain data stored in file, while spatial index is used to facilitate the visualization of terrain data in spatial database.

When surveyed the recent works regarding secondary-storage terrain visualization are compared on four aspects:

1. *Motivation*. Some of the works are devised specifically for terrain visualization, while others may be initially designed for MTM simplification, but are helpful to visualization as well.
2. *ROI support*. Whether a method can provide efficient terrain visualization or not is largely depend on whether it can support ROI (this one) and LOD (next one)

conditions efficiently. Some methods only support one efficiently; others may provide limited support of the two conditions.

3. LOD support.

4. *MTM support*. Some methods are designed for a specific type of MTM and can not be extended easily to support other types of MTM, which limits its application.

To improve the performance of secondary storage visualization some algorithms propose the use efficient disk paging techniques [7-10]. These algorithms try to “guess” what data may be needed for visualization and attempt to transfer them to main memory altogether in a block instead of bit by bit. The strategy that disk paging techniques use is trying to store terrain data that are close in space together in the file. One example is recently proposed work [10] that is based on the restricted quadtree. When storing data on secondary storage a technique similar to the Z-order space-filling curve is used allowing data close together in space are stored near each other on file, which improves the efficiency of reading data from secondary storage.

Disk paging techniques provide support for the LOD condition, but not for the ROI condition. The selective refinement algorithm must start with a coarse approximation and refine it gradually, which requires many I/O operations.

Disk paging is a general method and it doesn't have any specific requirement of the MTM allowing it to be extended to support all kinds of MTMs.

4. Spatial database support

Another way to manage the terrain data stored on secondary storage is to use a spatial database. To provide efficient support of ROI information locality information is needed. But when MTMs are constructed, most simplification algorithms focus on how to keep the approximation similar to the original model (i.e. introduce the minimum approximation error). So locality information in the original model is lost during the process of the simplification. A spatial index is needed to provide this information. The quadtree is a commonly used index that provides efficient access to locality-based information. Almost all the techniques reviewed in this section use a quadtree-type index to provide the ROI support.

Some of these methods are based on MTMs using an RSG (Regular Square Grid) structure, which makes it much easier to work with a quadtrees. Others are based on MTMs that use a TIN (Triangulated Irregular Network).

4.1 RSG-based MTM

RSG-based MTMs have been used for terrain visualization for quite some time. The majority of these methods [8, 11-13] use restricted quadtrees [14], which provide efficient support for ROI conditions.

The contribution of [11] is an efficient screen-space error metric calculation, and scene culling and vertex selection according to this error metric. The method proposed in [12] uses a priority-queue driven mesh refinement and a combination of object- and screen-space as error metric. In [8], efficient rendering is achieved by organizing the terrain's constituent triangles into a triangle strip that follows the

Hamilton space-filling curve. In [13], 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 problem is that to solve the discontinuity problem [14], extra triangles are added in restricted quadtrees, which increases the number of triangles. While providing efficient ROI support, quadtree indexes only provide very limited support for the LOD condition. Furthermore these methods are based on the RSG, which means they can not support TIN based on MTM, which is widely used today.

4.2 TIN-based MTM

While there are currently many TIN-based MTMs available, few of them can provide efficient secondary-storage visualization.

Magillo [15] presented a method based on Multi-Triangulation (MT) [16], 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 triangulation refinement (Fig. 2).

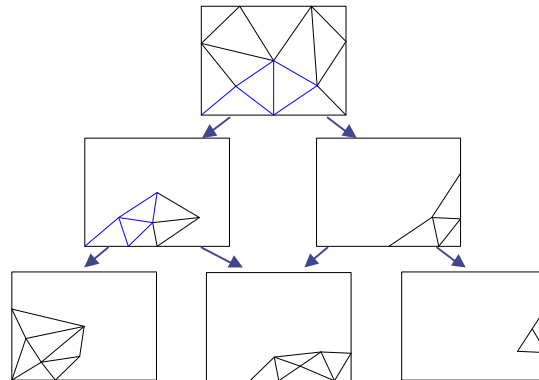


Fig. 2. Multi-Triangulation

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 keep 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 loaded into main memory and merged into a single MT (Fig. 3).



Fig. 3. Global MT and Local MT

This method is actually more focused on simplification rather than visualization. The terrain is partitioned so that each local MT can fit into the main memory and perform the simplification algorithm. It is easy to add a quadtree index to the local MTs, but the problem is there is no hierarchical relation between different local-MTs, limiting the method to storing all different LODs of same area collectively in one block. This means users have to retrieve a local-MT even when what they need is just one LOD.

One alternative, the Progressive Mesh (PM), is a multiresolution model built from a high-resolution triangulation through iterative edge-collapses. Hoppe [17] proposes a method to fit a progressive mesh [18] 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).

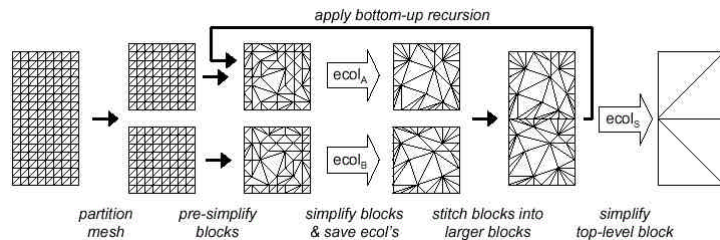


Fig. 4. Progressive with a quadtree

This method is similar to MTs in the sense it is more simplification-oriented: the terrain is partitioned so each grid can fit into the main memory and perform the simplification algorithm. As a by-product, the build-in quadtree index does help during visualization. But the segmenting and rejoining operations are expensive[19].

Another method [20] proposed to use “constrained Implicit TIN” to incorporate terrain features (such as road, river) with multiresolution terrain data. Terrain data and Feature data are stored separately. Terrain data is stored using a constrained Delaunay triangulation [21], and the feature data is stored using a line generalization tree [22]. Both data are indexed with a quadtree similar to PMR-quadtree [23]. During the reconstruction of the approximation mesh, the terrain surface is extracted first. Then the feature data is inserted into the terrain (Fig. 5).

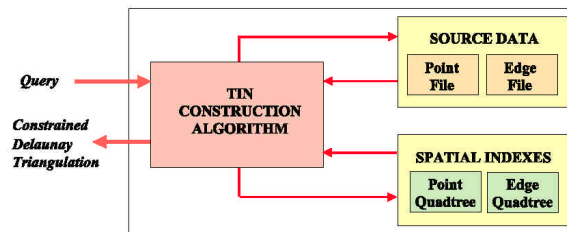


Fig. 5. Constrained implicit TIN

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

5. Research problem

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

Table 1. Comparison of secondary-storage terrain visualization methods

	Motivation	ROI support	LOD support	MTM support
Disk paging technique	Visualization	Good	No	All MTMs
Restricted quadtree	Visualization	Good	Very limited	RSG-based
Global and Local MT	Simplification	Limited	No	TIN-based
Progressive mesh with a quadtree	Simplification	Good	Limited	TIN-based
Constrained implicit TIN	Visualization	Good	Good	Constrained Delaunay triangulations

Methods that use disk paging techniques attempt to store together on disk terrain data that may be needed together for visualization. However, these methods cannot provide support for the LOD conditions.

Spatial database can provide a new approach to the problem. With the help of spatial indexes, efficient support of both ROI and LOD conditions could be provided. Quadtrees are currently a popular choice for ROI support, but further research is needed before such indexes can reliably provide efficient support for LOD conditions.

Quite some research has been done for RSG-based MTM, but again work is needed to improve the support for the LOD condition. It would be ideal if these methods can be extended to support TIN-based MTMs as well.

There is relative less work for TIN-based MTMs, and none of them can provide efficient support for both ROI and LOD condition, and support different types of TIN-based MTM at the same time. Research in this area is still in its early stage and there is much space for improvement.

6. Research proposal

One way to provide LOD support is to use LOD information as a new dimension in the spatial index. The concept of adding a scale dimension in spatial index is first

mentioned in the paper [24]. In [25] this work is continued and shows that using a 3D R-tree on both locality and scale dimensions gives better performance than having individual R-tree and B-tree indexes on the two dimensions.

To provide efficient support of LOD conditions, LOD information needs to be incorporated into the spatial index. Future work in this direction could utilize a 3D-quadtree index with the approximation error as a new dimension.

In the example shown in the Fig.1, after the user specifies the ROI and maximum screen error parameters, different part of the terrain will be assigned an approximation error value according to its distance to the viewpoint and other factors such as the position of the triangle. With the locality information stored in Quadtree index, the data retrieved will be constrained within ROI, but all LODs of that area needs to be retrieved because there is no information in the index to help choose among LODs. If the approximation error is added into the index, only the one LOD that suits for the specified LOD conditions needs to be retrieved. Fig.6 illustrates this (shaded area indicates the data needs to be retrieved for visualization).

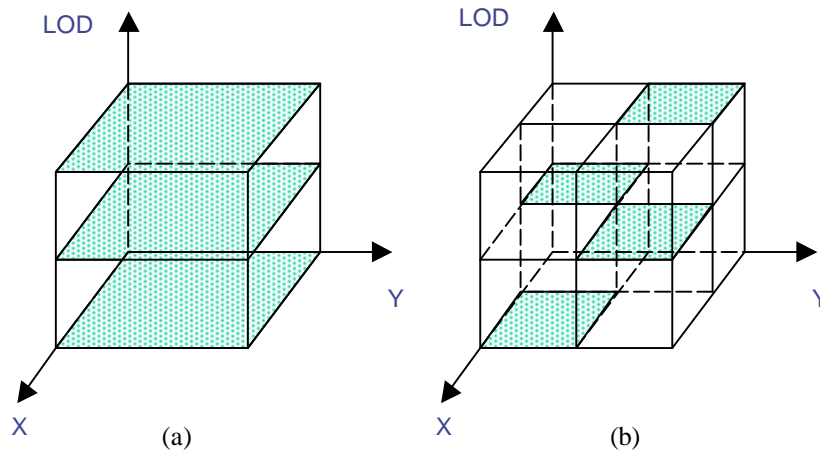


Fig. 6 Data retrieval with (a) Quadtree (b) 3D-Quadtree

In Fig.6(a), all the LODs within the ROI need to be retrieved because there is no information in the index to help choose among the LODs. In Fig.6(b), only one LOD is selected for each subset of the ROI with the help of the approximation error stored in the 3D quadtree index, which can largely reduce the amount of data retrieval.

This raises a new problem: if the terrain is made up of 10,000 vertices, the data retrieval will cause 10,000 disk I/O operations: one LOD is selected for each vertex and then retrieved from disk because even two adjacent vertices could have different LODs. So many disk I/O will largely slow down the visualization. To solve it, a filter-refinement strategy can be employed instead of performing selective refinement directly on the disk. Fig. 7 shows the framework of it.

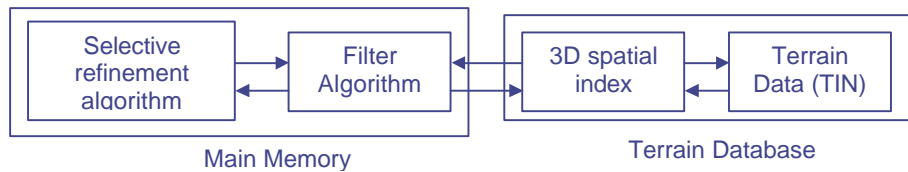


Fig. 7 Filter-refinement strategy

To perform a selective refinement, a filter algorithm will retrieve terrain data with reasonable looser constrains from the database (secondary storage); then in the main memory, a selective refinement algorithm is applied to finish the extraction. With the help of the 3D quadtree index, the terrain data can be first constrained within certain area, and then further reduced by the constrains in the approximation error dimension. The data retrieved from database will be something like in Fig.8.

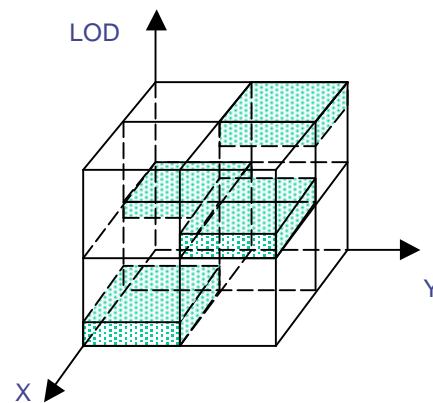


Fig. 8 Data retrieved by the filter algorithm

The filter-refinement algorithm should be able to:

1. Guarantee all the data necessary are retrieved
2. Data retrieved can fit into the main memory
3. Keep the balance between the number of disk I/O operations and the amount of data need to be retrieved
4. Extra data have a high chance to be used in later navigation.

Finally, the new approach can support different TIN-based MTMs because it does not have any specific requirement for the MTM itself.

Acknowledgement: Mincom Pty Ltd has provided us access to their terrain visualization and analysis products and data. Dan Sketcher has been worked on part of this as his Honors project.

References

- [1] M. Garland, "Multiresolution Modeling: Survey & Future Opportunities," presented at EUROGRAPHICS '99, 1999.
- [2] P. S. Heckbert and M. Garland, "Survey of Polygonal Surface Simplification Algorithms," presented at SIGGRAPH'97, 1997.
- [3] E. Puppo and R. Scopigno, "Simplification, LOD and Multiresolution Principles and Applications," *Eurographics*, vol. 16, 1997.

- [4] P. Cignoni, C. Montani, and R. Scopigno, "A comparison of mesh simplification algorithms," *Computers & Graphics*, vol. 22, pp. 37-54, 1998.
- [5] L. De Floriani, P. Magillo, and E. Puppo, "VARIANT: a system for terrain modeling at variable resolution," *GeoInformatica*, vol. 4, pp. 287-315, 2000.
- [6] L. De Floriani, P. Magillo, and E. Puppo, "Efficient implementation of multi-triangulations Proceedings of Visualization '98," Research Triangle Park, NC, USA, 1998.
- [7] D. Davis, T.-y. Jiang, W. Ribarsky, and N. Faust, "Intent, Perception, and Out-of-Core Visualization Applied to Terrain," Graphics, Visualization & Usability (GVU) Center GIT-GVU-98-12, 1998.
- [8] R. Pajarola, "Large scale terrain visualization using the restricted quadtree triangulation Proceedings of Visualization '98," Research Triangle Park, NC, USA, 1998.
- [9] J. El Sana and C. Yi Jen, "External memory view-dependent simplification," *Computer Graphics Forum*, vol. 19, pp. C139-50, 528, 2000.
- [10] P. Lindstrom and V. Pascucci, "Visualization of Large Terrains Made Easy," presented at IEEE Visualization, San Diego, California, 2001.
- [11] 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 Proceedings of 23rd International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'96)," New Orleans, LA, USA, 1996.
- [12] M. Duchaineau, M. Wolinsky, D. E. Sigiety, M. C. Miller, C. Aldrich, and M. B. Mineev Weinstein, "ROAMing terrain: Real-time Optimally Adapting Meshes Proceedings," Phoenix, AZ, USA, 1997.
- [13] K. Baumann, J. Dollner, K. Hinrichs, and O. Kersting, "A hybrid, hierarchical data structure for real-time terrain visualization Proceedings of Computer Graphics International 1999," Silicon Graphics Int, 1999.
- [14] B. Von Herzen and A. Barr, "Accurate Triangulations of deformed, intersecting surfaces," *Computer Graphics*, vol. 21, pp. 103-110, 1987.
- [15] P. Magillo and V. Bertocci, "Managing large terrain data sets with a multiresolution structure," 2000.
- [16] E. Puppo, "Variable resolution terrain surfaces," 1996.
- [17] H. Hoppe, "Smooth view-dependent level-of-detail control and its application to terrain rendering Proceedings of Visualization '98," Research Triangle Park, NC, USA, 1998.
- [18] H. Hoppe, "Progressive meshes Proceedings of 23rd International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'96)," New Orleans, LA, USA, 1996.
- [19] E. Shaffer and M. Garland, "Efficient Adaptive Simplification of Massive Meshes," presented at IEEE Visualization 2001, 2001.
- [20] D. B. Kidner, J. M. Ware, A. J. Sparkes, and C. B. Jones, "Multiscale terrain and Topographic Modelling with the Implicit TIN," *Transactions in GIS*, vol. 4, pp. 379 - 408, 2000.
- [21] L. De Floriani, "A pyramidal data structure for triangle-based surface description," *IEEE Computer Graphics and Applications*, vol. 9, pp. 67-78, 1989.
- [22] C. B. Jones and I. M. Abraham, "Line Generalisation in a global cartographic database," *Cartographica*, vol. 24, pp. 32 - 45, 1987.
- [23] R. C. Nelson and H. Samet, "A consistent hierarchical representation for vector data," *Computer Graphics*, vol. 20, pp. 197-206, 1986.
- [24] M. Horhammer and M. Freeston, "Spatial indexing with a scale dimension," 1999.
- [25] S. Zhou and C. B. Jones, "Design and Implementation of Multi-scale Databases," presented at Advances in Spatial and Temporal Databases, 7th International Symposium, SSTD 2001, Redondo Beach, CA, USA, 2001.