05-MS

The 2nd International Conference on Cybernetics and Information Technologies, Systems and Applications

11th International Conference on
Information Systems Analysis and Synthesis

July 14-17, 2005 Orlando, Florida ~ USA

PROCEEDINGS

Volume II

Edited by
Jose Aguilar
Hsing-Wei Chu
Elena D. Gugiu
Ilka Miloucheva
Naphtali Rishe

Organized by
International Institute of Informatics and Systemics
Member of the International Federation of Systems Research (IFSR)

Mesh Simplification Algorithm for Online 3D GIS

Naphtali D. RISHE, Ouri WOLFSON, Yanli SUN, Maxim CHEKMASOV,
Andriy SELIVONENKO, Scott GRAHAM
School of Computer Science, Florida International University
Miami, Florida 33199, USA

and

Ben WONGSAROJ, Keith MORREN, Royel HAYNES, Kiesha PIERRE, Asha BRITO
Division of Computer Sciences and Mathematics, Florida Memorial University
15800 NW 42nd Ave Miami Gardens, FL 33054, USA

ABSTRACT

3D GIS generally have very complex data models. In adapting these systems to the Internet, one has to take into account the limited computational power of the typical personal computer and the limited network bandwidth available to casual Internet users. For quality of service management of interactive 3D GIS presentations, feature preserving data reduction techniques are of critical importance. The technique discussed in this paper deals with terrain modeling. The terrain surface usually exhibits significant spatial coherence. Such data coherence can be found in grid meshes regardless of their resolutions. The mesh simplification algorithm proposed here reduces the geometric complexity in grid meshes by taking advantage of this coherence.

Keywords: Geographic information system, terrain modeling, three-dimensional visualization.

1. INTRODUCTION

3D (three-dimensional) GIS (geographic information systems) generally have very complex data models. While two-dimensional online GIS have been successfully implemented (we refer to [1] as an example), said complexity greatly limits wide adoption of 3D GIS applications for the Internet. We believe that employing feature preserving data reduction techniques allows the complexity of the 3D GIS data models to be reduced while presenting to the user a real-time realistic three-dimensional animation. For this reason we have developed the mesh simplification algorithm discussed in this paper.

For our purposes, mesh data is stored and retrieved in a grid format. Grid meshes are obtained by scanning the terrain surface at constant intervals. Some of the sample vertices may be unnecessary or redundant in representing the actual terrain geometry. For example, to represent the geometry of a large rectangular plane field, the four corner vertices of the plane field are sufficient. For a 400×200 meter^2 playground, a 20×20 meter grid mesh contains 200 facets and a 40×40 meter grid mesh contains only 50 facets. Eliminating these unnecessary sample vertices in a grid mesh is an effective approach to reducing the rendering complexity. Moreover, when viewing some near-flat areas at a distance, very small height variations in terrain may not be detectable by human perceptions. Ignoring those insignificant terrain height variances can also reduce the geometric complexity without impairing the rendering quality. Since mesh servers can provide mesh data of different resolutions in multiple LOD (level of detail) rendering, the goal for a mesh simplification algorithm is to eliminate those redundant or insignificant vertices in a grid mesh in order to reduce the facets of the terrain surface. The efficiency of the algorithm is very important, since it is used in real time for 3D animation. The mesh simplification algorithm is designed as a hybrid method of mesh decimation and compression. It can directly process rectangular meshes to avoid mesh triangulation overhead. The redundant or insignificant vertices are removed by edge straightening and adjacent rectangles are merged according to error metrics via quad-tree compression. The algorithm allows mesh data to be processed locally on the user's computer and the entire terrain mesh to be simplified in parallel. We refer to [2] for a detailed discussion of the data reduction techniques for online 3D GIS.

2. DESIGN CONSIDERATIONS AND REQUIREMENTS

Most feature-preserving mesh simplification algorithms that have been published are based on triangular meshes, [3, 4, 5, 6]. These algorithms are merited for topology preservation and controllability over a number of surfaces. They are widely used for obtaining simplified versions of various resolutions from excessively detailed meshes. On the other hand, their efficiency is not suitable for real-time use on the Internet. Moreover, to fit triangle-based algorithms, a grid mesh has to be transformed into a triangular mesh via triangulation algorithms.

In our model, the mesh data does not contain excessive geometry details because it is retrieved with proper resolutions from the mesh servers. The simplification process aims to smooth the terrain surface and to eliminate insignificant and redundant vertices contained in grid meshes. Apart from the effectiveness of mesh complexity reduction, the run-time efficiency of the simplification process is also a major concern. It is desirable that the algorithm be able to directly work on rectangular mesh to avoid triangulation overhead. A local mesh processing strategy is also preferred to allow the entire terrain mesh to be efficiently processed progressively in parallel.

3. MESH SIMPLIFICATION ALGORITHM

The mesh simplification algorithm is a hybrid algorithm that combines quad-tree compression and edge straightening. It works directly on the grid mesh tiles and can be performed in parallel. Mesh data is smoothed by error metrics and a controlled edge straightening method. The geometry elements are compressed by merging rectangles to reduce the number of facets in mesh data.

3.1. Per-Tile Processing

When a given mesh tile is retrieved from the mesh server in grid format, the algorithm recursively divides the mesh tile into 2×2 sub-regions until the bottom level grid cells are reached (in which case a sub-region contains a single rectangular facet). Then the algorithm recursively merges those 2×2 sub-regions into larger regions in a bottom-up fashion. The simplification occurs during the sub-region combination processes. When combining the 2×2 sub-regions, the algorithm examines all adjacent rectangular facets in different sub-regions along the sub-region borders. For two rectangular facets sharing a common edge, if their connecting vertices can be removed by the edge straightening process, then they are merged into a larger rectangle. The pseudo code for the simplification process is shown on Figure 1.

The edge straightening is controlled by edge linearity error metrics. Given two edges AB and BC in same orientation (either x(A)=x(B)=x(C) or y(A)=y(B)=y(C), where x(V), y(V) and h(V) denote the x, y, and z values of vertex V), the edge linearity error metric of vertex B (denoted as e(B)) is calculated as follows:

- When y(A) = y(C):

$$e(B) = \left| \frac{h(C)-h(A)}{x(C)-x(A)} \times (x(B)-x(A)) + h(A) - h(B) \right|$$

- When x(A) = x(C):

$$e(B) = \left| \frac{h(C)-h(A)}{y(C)-y(A)} \times (y(C)-y(A)) + h(A) - h(B) \right|$$

A small e(B) value indicates that AB and BC are close to a straight line and a zero e(B) means they are exactly a straight line. If e(B) is less than the error metric threshold at vertex B (denoted as T(B)), then AB and BC can be straightened into a straight line AC and vertex B can be removed.

```
/* Pseudo code for mesh simplification procedure */

simplifyMesh( Region*pRegion, double[][] mesh, double res,
                        int X1, int Y1, int X2, int Y2)

{
   int MidX = (X1+X2) / 2;
   int MidY = (Y1+Y2) / 2;

   if (X1==X2 && Y1==Y2){
            Rectangle * ret = new Rectangle;
            ret->x = X1;                    ret->y = Y1;
            ret->width = ret->width = 1;;
            pRegion->add(ret);
   }
   else if (X1==X2){
            Region top, bottom;
            //splitting along y
            simplifyMesh(&top, mesh, res, X1, Y1, X1, MidY);
            simplifyMesh(&bottom, mesh, res, X1, MidY+1, X1, Y2);

            //merging sub-regions
            mergeRegion(pRegion, &top, &bottom, mesh, res, TOP_BOTTOM);
   }
   else     if (Y1==Y2)  {
            Region left, right;

            //splitting along x
            simplifyMesh(&left, mesh, res, X1, Y1, MidX, Y1);
            simplifyMesh(&right, mesh, res, MidX+1, Y1, X2, Y1);

            //merging sub-regions
            mergeRegion(pRegion, &left, &right, mesh, res, LEFT_RIGHT);
   }
   else     {
            Region top_left, top_right, bottom_left, bottom_right;
            Region top, bottom;

                     //2x2 splitting
            simplifyMesh(&top_left, mesh, res, X1, Y1, MidX, MidY);
            simplifyMesh(&top_right, mesh, res, MidX+1, Y1, X2, MidY);
            simplifyMesh(&bottom_left, mesh, res, X1, MidY+1, MidX, Y2);
            simplifyMesh(&bottom_right, mesh, res, MidX+1, MidY+1, X2, Y2);

            //merging sub-regions
            mergeRegion(&top, &top_left, &top_right, mesh, res,
                     LEFT_RIGHT);
            mergeRegion(&bottom, &bottom_left, &bottom_right, mesh, res,
                     LEFT_RIGHT);
            mergeRegion(pRegion, &top, &bottom, mesh, res, TOP_BOTTOM);
   }
}
```

Figure 1. Pseudo code for mesh simplification procedure.

For two rectangles ABCD and CDEF shown on Figure 2, the rectangles can be merged into a larger rectangle ABFE if and only if both vertices C and D can be removed. For a rectangular mesh, since the four corner vertices of a rectangle may not be coplanar, removing a vertex of a rectangle may cause a topological inconsistency (crack) in the mesh without cascading adjustment of other neighboring vertices. Except for the linearity error metrics, the topology of neighboring rectangles must also be considered at the same time when straightening edges and merging rectangles.

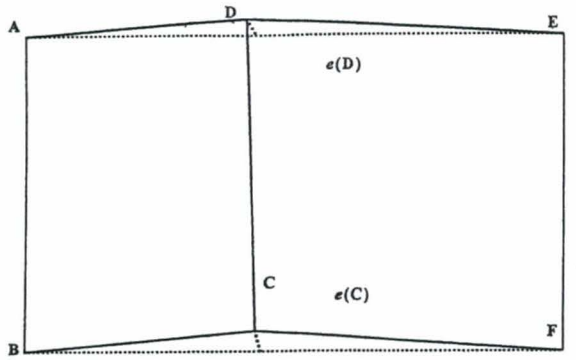

Figure 2. Illustration of Merging Two Rectangles.

For two rectangles ABCD and CDEF, based on the edge linearity error metrics values e(C) and e(D) of vertices C and D, the simplification algorithm works as follows:

1. If e(C) and e(D) are all zeros then merge the two rectangles into
   ABFE
2. Else in case of e(C) ≤ T(C) and e(D) ≤ T(D)
   1. If ABCD and CDEF are all grid cells
      (at the lowest level), then merge them.
   2. Otherwise check the neighboring rectangles and merge them only when merging edges ADE and BCF is safe.

In case 2.2 above, if AD and DE are not edges of grid cells, they can only be safely straightened into AE if and only if there exist two other rectangles ADVU and DEWV, see Figure 3; otherwise straightening ADE without adjusting h(V) accordingly may result in a triangular hole near vertex V, see Figure 4.

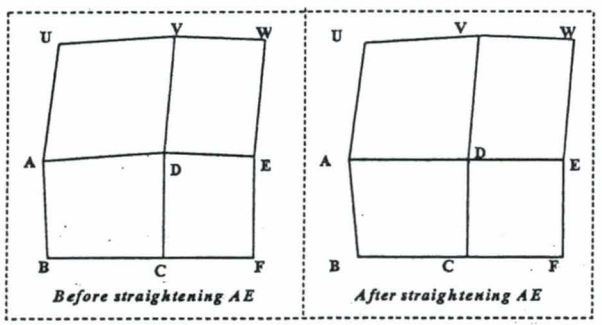

Figure 3. Example of Safe Edge Straightening.

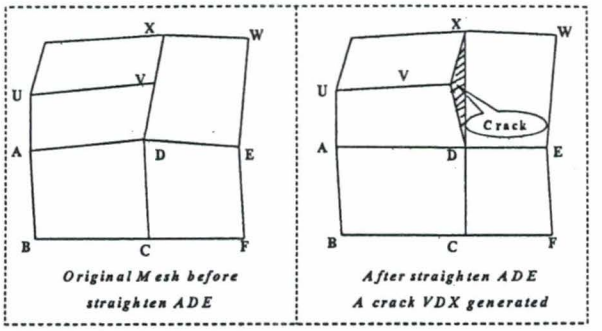

Figure 4. Mesh Crack Resulting from Improper Edge Straightening.

Since merging of rectangles occurs only along the borders of adjacent sub-regions, in each sub-region, the rectangles along the borders are separately indexed with respect to the four borders (top, left, right, and bottom) to accelerate the rectangle search and comparison.

3.2. Edge Straightening Threshold

The straightening error metrics threshold T controls the simplification result. Generally, a large threshold results in a coarse mesh and a small threshold results in a fine mesh. Therefore, a large threshold should be used for simplifying low-resolution meshes and a small threshold should be used for simplifying high-resolution meshes. The threshold T for a mesh tile is determined by view height and the mesh resolution. Experimentally, the value of the threshold was calculated to be:

$$T(V) = \min\left(\frac{H}{800}, \frac{Res}{25}\right)$$

Where Res is the coarse mesh resolution and H is the viewing height.

3.3. Parallel Processing

Since the terrain mesh is simplified tile by tile, the entire simplified terrain surface composed of independently simplified mesh tiles should remain spatially continuous. In the multi-resolution terrain model, meshes in different LOD sub-regions have different resolutions and their simplification processes are controlled using different error metrics thresholds. When two adjacent mesh tiles of different resolutions are processed separately with different error metrics thresholds, the simplified tiles may not match each other and cracks may appear at the connecting borders.

In the multi-LOD rendering model construction process, mesh tiles across sub-region borders are treated specially. For two adjacent mesh tiles A and B, where A has higher resolution than B, the border of A that connects with B has to be adjusted with B's resolution to prevent cracks. In the simplification process, the adjusted border of A is simplified using an error metrics threshold derived from B's resolution instead of A's own resolution. The

threshold for edge straightening for a vertex V in mesh tile A is defined as:

$$T(V)=\begin{cases}\min\left(\frac{H}{800},\frac{ResB}{25}\right) & \text{For vertex V in the border area with adjusted resolution}\\ \min\left(\frac{H}{800},\frac{ResA}{25}\right) & \text{For other vertices in mesh tile A}\end{cases}$$

Where: ResA is the coarse resolution of the mesh tile A.

ResB is the coarse resolution of the mesh tile B.

Therefore in the rendering model, a mesh tile always seamlessly connects with its neighboring mesh tiles and their border edges are always simplified in the same manner even if they are of different resolutions. This scheme ensures that the entire simplified terrain surface is spatially continuous and crack-free.

4. ANALYSIS AND EXPERIMENT RESULT OF MESH SIMPLIFICATION ALGORITHM

The result of using the mesh simplification algorithm is a bounded error while preserving the mesh geometric topology. Figure 5 shows a comparison of original and simplified surfaces of the same area of terrain.

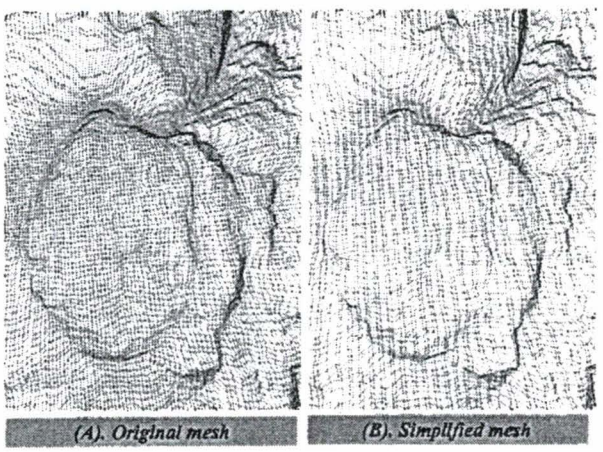

Figure 5. Comparison of the Original and Simplified Surfaces.

The goal of the mesh simplification algorithm is to efficiently reduce the mesh complexity and thus to speed up the terrain rendering process. According to experimental results, the algorithm can reduce the number of facets by 40 to 85 percent compared to the original mesh. An example of mesh tile simplification comparison is shown in Figure 6. The 32-meter resolution mesh tile is taken from the South Florida area and the simplification result is obtained using a 700-meter viewing height. The 64×64 original mesh is shown in part (A) and simplified mesh of 1137 facets is shown in part (B). The facet reduction rate in this example is about 72 percent.

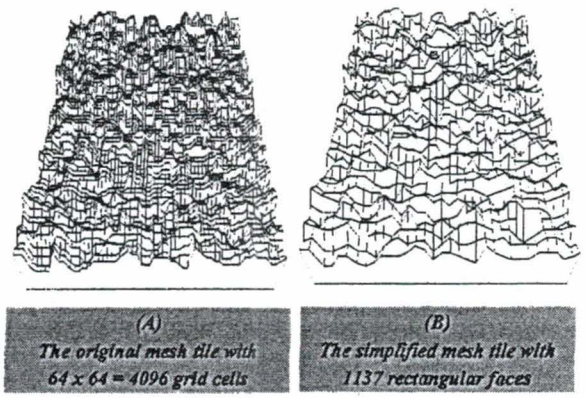

Figure 6. Example of Mesh Tile Complexity Reduction.

Although the present algorithm is far from optimal in terms of geometric element reduction rate when compared to well-known triangle-based mesh simplification algorithms (e.g. incremental edge collapsing), it is suitable for real-time use in the Internet environment due to its simplicity and efficiency. There is no need for mesh triangulation in this algorithm. It does not need to maintain extra data structures (such as a priority queue) and no iterative error metrics computation or priority re-calculation is required.

Since the simplification procedure is designed on a per-tile processing basis, the local simplification strategy brings about a significant performance enhancement:

- the entire terrain surface can be simplified in parallel.
- the result of simplification of an individual tile can be cached as long as its error threshold remains the same. This means that only a small portion of mesh data in the rendering model has to be processed for each frame.

To evaluate the performance improvement resulting from the mesh simplification algorithm, we experimentally compared the execution times of the rendering process directly over the original grid mesh (scheme A) with the execution times of simplifying the mesh plus rendering over the simplified mesh (scheme B). In the experiment, the two rendering schemes were implemented with two programs A and B that are almost identical apart from the mesh processing strategies. The two programs were implemented in a single-threaded manner to prevent the results from being compromised by multithreading initiation and synchronization overhead. All texture and mesh data was pre-downloaded to the local drive of the

computer and the time for data reading and decoding is excluded from execution time measuring. Since the mesh simplification results differ with the roughness of the underlying terrain, two datasets were used in the experiment:

- Dataset#1: 40 texture/mesh tiles over the South Florida area.
- Dataset#2: 40 texture/mesh tiles over the Northwest California area.

For the two datasets, all mesh tiles are 64×64 grids at 32-meter resolution and all texture images are 512×512 pixels in size with a resolution of 4 meters/pixel. The two testing programs simulate viewing at an altitude of 1,000 meters with a 40 degree fovy angle and zero viewing rotation angles (the field of view (fovy) specifies the angle of the view volume).

The test was performed on a Microsoft Windows 2000 workstation with a single Intel Pentium 4 Processor (2.8GHz) and 512 Megabytes of memory without an advanced graphics card. The experimental result is shown in Table 1.

Dataset	Rendering time (ms)				Performance Improvement(%)
	Scheme A	Scheme B			
		Simplification	Rendering	Total	
1	166	27	112	139	16.3
2	178	30	121	151	15.2

Table 1. Comparison of Rendering Performance Experiment.

The results of the experiment show that simplification can effectively reduce the execution time of the overall rendering process by about 15 percent in these cases.

6. ACKNOWLEDGEMENTS

This material is based on work supported by the National Science Foundation under Grants No HRD-0317692, EIA-0320956, and EIA-0220562.

5. REFERENCES

[1] TerraFly: A Web-Enabled Application for Visualization and Manipulation of Remotely Sensed Data, available at http://terrafly.fiu.edu/tf-whitepaper.pdf

[2] Yanli Sun, 3D TerraFly – Quality of Service Management of Online Interactive 3D GIS Presentation, PhD dissertation, Florida International University, 74 p., 2004.

[3] Rossignac J. Borrel P., Multi-resolution 3D approximations for Rendering Complex scenes, In Falcidieno, B. and Kunii T.L. (Eds), Modeling in Computer Graphics, Springer-Verlag, pp. 455-465, 1993.

[4] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle, Mesh Optimization. In SIGGRAPH '93 Conference Proceedings, pp. 19-26, 1993.

[5] Ronfard R. and Rossignac, J., Full-range approximation of triangulated polyhedra, Computer Graphics Forum, Proceedings of Eurographics, 1996.

[6] P. Lindstrom, D. Koller, W. Ribarsky, L. Hodges, N. Faust, and G. Turner, "Real-time Continuous Level of Detail Rendering of Height Fields," Proc. ACM SIGGRAPH'96, pp. 109-118, 1996.