A Fast Digital Terrain Simplification Algorithm with a Partitioning Method

Yoon-Sig Kang  Taixup Lee  Sung-Bong Yang  Woo-Chan Park

Department of Computer Science
Yonsei University
Seoul, 120-749, Korea
Tel: +82-2-361-2717 Fax: +82-2-365-2579
{noori, taixup, yang, chan}@mythos.yonsei.ac.kr

Abstract

In this paper we introduce a fast simplification algorithm for terrain height fields to produce a triangulated irregular network, based on the greedy insertion algorithm in [1,4,5]. Our algorithm partitions a terrain height data into rectangular blocks with the same size and simplifies blocks one by one with the greedy insertion algorithm. Our algorithm references only to the points and the triangles within each current block for adding a point into the triangulation. Therefore, our algorithm runs faster than the greedy insertion algorithm, which references all input points and triangles in the terrain. Our experiment shows that partitioning method runs from 4 to more than 20 times faster, and it approximates test height fields as accurately as the greedy insertion algorithms. Most greedy insertion algorithms suffer from elongated triangles that usually appear near the boundaries. However, we insert the four corner points into each block to produce the base triangulation of the block before the point addition step begins so that elongated triangles could not appear in the simplified terrain.

1. Introduction

Recently 3D graphics is becoming a more and more important subject in many applications to enhance the realism, for instance, airplane/vehicle simulations, medical/educational visualizations, 3D geographical information systems, and the entertainment. As a result, many 3D dedicated hardware devices and 3D visualization algorithms have been introduced [1],[2]. However, these 3D hardware devices still do not have enough performance to render full resolution 3D images in real time. For real time rendering, it is required to reduce the resolution of a 3D image to a lower resolution according to their importance of visibility. This reduction method is called simplification. Simplification is not only for 3D objects but also for terrain height fields. A terrain height field is a sample data for a planar terrain. Simplification for a terrain height field is also indispensable for many 3D graphics applications.

One of the most important aspects of terrain height field simplification is to reduce the number of primitive at a given resolution. Refinement greedy insertion algorithm is one of the methods to produce a triangulation with the highest quality. It builds the simplest triangulation for the first step and, on each pass, inserts a point with highest error as a vertex in the triangulation. On every pass, to find the inserted point, the algorithm recalculates errors of the altered points on the domain and compares the all remainder points. We introduce the partitioning algorithm which partitions the terrain into small rectangular blocks and simplifies each block with greedy insertion algorithm. The algorithm improves the execution time by reducing the searching area to find the highest error point and produces a triangulation as good quality as greedy insertion algorithm.

There have been many researches for the efficient representations for geometric terrains on computers [10],[11],[8],[9],[4]. One of the effective and convenient representations is the Triangulated Irregular Network (TIN) representation. In this representation, a terrain is represented by triangulated polygons each of which is made by joining three sampled points of the terrain as its three vertices.

In this paper, we introduce a fast simplification algorithm based on the refinement greedy insertion algorithm in [7]. Our algorithm produces a TIN as output. A terrain height field is obtained by determining the elevation value of each point in the rectangular mn grid. It can be represented as a set of \( (p, H(p)) \), where \( p \) is a point in the \( m \times n \) grid and \( H(p) \) denotes the elevation at the point \( p \) on the input sur-
face. A TIN of a height field can be viewed as a subset of the \( m \times n \) grid points.

The remainder of the paper is organized as follows. In Section 2, greedy insertion algorithm is explained. In Section 3, we introduce the partitioning algorithm. In Section 4 we show the result of our experimentation. And we present the conclusion in Section 5.

2. Related Work

In this part, we introduce the basic background knowledge of greedy insertion algorithms which may be helpful to understand our algorithm. The greedy insertion algorithm is to get the desired triangulation by recursively adding the most important points (max local/global error point) to the base triangulation as their name implies. The first step of greedy insertion algorithm is to make a base triangulation which has only two triangles by joining the four corner points of a terrain. After that, a point is selected by their local/global error and the point is added into the triangulation to increase the resolution of the triangulation. This step is repeated until the triangulation reaches to the desired resolution.

2.1. Voronoi Diagrams

Voronoi Diagram is a set of sites which divide the Euclidean plane. Each plan corresponds to one of the sites. Let \( p = \{p_1, p_2, \ldots, p_n\} \) be a set of points on the two dimensional Euclidean plane, these points are site, partition the Euclidean plane by assigning every point on the plane to its nearest site, and all the points on the plane are assigned to Voronoi region, \( V(p_i) \).

\[
V(p_i) = \{x : |p_i - x| \leq |p_j - x|, i \neq j\}
\]

If there is a point which has two nearest sites, it is on the edge of Voronoi diagram. And If a point has four nearest sites, the Voronoi diagram is degenerated to avoid the abnormality and the point does not have four nearest sites.

Delaunay Triangulation has two important properties which are as followed.

- There is a unique Voronoi diagram corresponding to a given point set.
- If \( v \) is a Voronoi vertex at the junction of regions \( V(p_1), V(p_2) \) and \( V(p_3) \) then \( v \) is the center of the circle \( C(v) \) determined by \( p_1, p_2, p_3 \). And the interior of a circumcircle \( C(v) \) contains no sites.

2.2. Delaunay Triangulation

In 1934 R. Delaunay proved that Voronoi diagram produce a planar triangulation by drawing the Voronoi sites \( P \) of incident Voronoi regions with straight lines. If there are not four sites which are cocircular [4].

2.3. Greedy Insertion Algorithm

Greedy insertion algorithm is one of the simplification algorithms. It is also called as refinement algorithm since the algorithm produce a triangulation from a minimal initial approximation to desired resolution by adding one or more points into the triangulation on each step. De Floriani’s greedy insertion algorithm starts from making initial triangulation with the four corner points of input height data. This initial triangulation has 2 triangles, which cover whole terrain with minimal approximation. The local error measurement is used to choose a point for adding into the triangulation. For a point \( p \), the measurement can be written as

\[
e(p) = |z - f(p)|,
\]

where \( p = (x, y, z) \) and \( f(p) \) is the height computed at \( p \) as a linear interpolation of the \( z \) values at the vertices of the triangle whose projection encloses the projection of \( p \) (Figure 2).
of P is interior to the circumcircle of any triangle of the triangulation. The three edges common to t are tested with the circle criterion. If any subtriangle is identified as not valid under Delaunay triangulation, the common edges are swapped with the opposite diagonal of its enclosing quadrilateral (Figure 3).

![Figure 3. The circle criterion (a) and the edge swapping (b)](image)

### 3. A fast terrain simplification with a partitioning method

Most greedy insertion algorithms take their execution times from several seconds to tens of seconds for generating a simplified triangulation for a given terrain height field of an ordinary size. In these algorithms all points and triangles have to be inspected to select a point, which is to be added into the triangulation. To reduce the number of points and triangles to be inspected, we employ a partitioning method. Our algorithm consists of three steps, partitioning the input terrain, base triangulation, and point addition. The code is given below.

```c
/* Partitioning the input terrain height field */
Partition a terrain into k x l blocks;
/* Initial Triangulation */
for each block b
  Make_InitialTriangles(b);
end for
/* Point Addition */
for each blocks b
  Init_Triangles(b);
  Init_Error(b);
  Find the local maximum error e and the point p
  by calling Get_Max_Error&Point(b);
  while e <= \tau do /* while the error is less than threshold */
    Insertion(p);
    Recalculation(b);
    Find the local maximum error e and the point p
    by calling Get_Max_Error&Point(b);
  end while
end for
```

#### 3.1. Partitioning a terrain height field

Partitioning method partitions a terrain into several smaller blocks and gives high-speed point addition. Since the input terrain is a set of sampled points, the common way to partition a terrain height field will be distributing the each sample point into exactly one block. In this way, each block does not share any information and has independent base triangles with adjacent blocks. Therefore, point addition for each block is done individually, and the merging process is needed for joining up the triangulations to one triangulation of the terrain. Also the point addition step for each block does not share any information with adjacent blocks and it may produce the unmatched triangles, as shown Figure 4 (a).

![Figure 4. (a) unmatched (b) The boundary points (black) of blocks shared by adjacent blocks](image)

In order to avoid such problems, we make the blocks share boundary sample points with adjacent blocks as shown in Figure 4 (b). In the figure, the black square points are shared by at least 2 blocks and used to make initial triangles. And the black dots place on the shared edge with 2 adjacent blocks. After making initial triangulation, each base triangle in a block shares their edges with a triangle in its adjacent block. And each initial triangle is connected by shared edge to one triangulation of the terrain. After that, adding a point into a block, it influences the change to the adjacent blocks.

One of the important aspects of partitioning is the number of blocks. If we increase the number of blocks for partitioning a terrain, the simplification time will become faster since the number of points considered for searching the maximum local error point is reduced. But in the other hand, increasing the block number may causes for generating a lower quality of triangulation. Our algorithm inserts the corner points (we call it initial point) of the blocks to generate base triangulation. The number of initial point is in direct proportion to the number of blocks. If we partitioning a terrain into too many blocks the blocks could have
lower local error than the threshold $\tau$. In these blocks, no more points are inserted and the feature point could not inserted.

Our algorithm could reduce the simplification time by increasing the block number as long as the triangulation includes the feature points. Considering the threshold $\tau$, size of a height field, and complexity of the height field can choose the most propriety block number.

3.2. Initial triangulation

Partitioning algorithm builds two triangles with the four corner points of each block. These two initial triangles of each block cover the block as shown in Figure 5. When the number of blocks is $k \times l$, partitioning algorithm inserts $(k + 1) \times (l + 1)$ initial points to make initial triangulation. The initial points may increase the error of approximated triangulation, since the points are inserted into the triangulation without guaranteeing whether they are feature points. Therefore, it is needed to reduce the initial points for reduce the error of approximated triangulation. The other hand, reducing the number of block requires more execution time. Therefore, the decision is needed for number of blocks, which is sufficient to both quality and speed. We tested the errors of several approximated triangulations with different numbers of blocks.

![Figure 5. The initial triangulation](image)

3.3. Point addition

There are two cases to reference and change the triangles that are not in the current block when a point is added into the current block. The first is very clear, if a point is added on the boundary of two blocks, then the triangles which have the edge will be split into two smaller triangles not only for the triangle which is in the current block but also for the triangle which is not in the current block. The second is when a point is added into a boundary triangle that has the boundary edge of a block. The triangle is split into 3 smaller triangles. If one of the three split triangles which has the boundary edge has to switch the common edge for Delaunay Triangulation condition with adjacent triangle which is not in the current block. After common edge switching the triangles occupying through more than two adjacent blocks will be generated. With these two cases, the changing of a current triangulation propagates to adjacent blocks.

![Figure 6. A point added on the boundary shared by 2 blocks](image)

![Figure 7. A point added on a boundary triangle](image)

4. Experimentation

We tested five different terrain digital elevation models (DEM) on a PC with an Intel Pentium II 400Mhz CPU and 128Mbyte RAM. The same DEMs were used in testing the simplification algorithm in [1] and are shown in Table 1. Since the five DEMs show very similar results, we project Crater for the others.

For the test, we measured the generation time and the triangulation quality. To show the goodness of the algorithm, we compounded and compared our algorithm with the Michael Garland and Paul Heckbert’s algorithm III (FPA) in [4] FPA has the expected time of $O((m+n)\log m)$, where $n$ is the number of input points and $m$ is the number of vertices in the reconstructed triangulation. When we partition a terrain into $k \times l$ grid blocks, only the points in the current block are considered in choosing the point which has the maximum local error. For the sake of convenience, we assume that the same number of points is added to each block. Then the expected time of our algorithm for each block is $O((\frac{m+n}{kl})\log \frac{m}{kl})$. And the expected time for the
whole terrain simplification is \( O((m + n) \log \frac{m}{M}) \), since there are \( k \times l \) blocks.

### Table 1. The five digital elevation models

<table>
<thead>
<tr>
<th>Name</th>
<th>Dimensions</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ashby</td>
<td>346 x 452 (156392 pts)</td>
<td>Ashby Gap, Virginia</td>
</tr>
<tr>
<td>Crater</td>
<td>336 x 459 (154224 pts)</td>
<td>West half of Crater Lake, Oregon</td>
</tr>
<tr>
<td>Ozark</td>
<td>369 x 457 (154378 pts)</td>
<td>Desert around Mt. Tefert, California</td>
</tr>
<tr>
<td>NTC</td>
<td>1024 x 1024 (1048576 pts)</td>
<td>Ozark, Missouri</td>
</tr>
<tr>
<td>West US</td>
<td>1024 x 1024 (1048576 pts)</td>
<td>Section of Idaho/Wyoming border</td>
</tr>
</tbody>
</table>

Figure 8 shows the running times of our algorithm with 12 × 12 blocks on the DEMs. Figure 9 shows the running times of our algorithm with several block numbers on Crater DEM. The figure shows that the running time depends on the number of blocks and even the partitioned terrain with 6 × 6 blocks is produced four times faster than FPA. It is clear that partitioning the terrain into smaller blocks is the way to improve the cache hit rate. Partitioning algorithm reduces the input points that are referenced for each path to add a point into triangulation in simplification sequence.

Figure 10 shows that partitioning a terrain with 6 × 6 blocks has the quality as good as greedy insertion algorithm. In Figure 11 shows that partitioning a terrain is not always increase the mean error. The mean error is increased when the number of block is increased. But after a certain resolution, the block number does not increase the mean error.

To test the quality of partitioning triangulations, we compared the mean errors of the various number of blocks on Crater. The error is the mean interpolation error between the height of input point \( P \) and height value of projection \( P \) on generated triangulation, which is described in Eq. (1), where \( PN \) is the number of input points, \( h_{\text{max}} \) and \( h_{\text{min}} \) are the maximum/minimum height value of input point and \( RP_i \)'s are the height values of input point and \( SP_i \)'s are the height value of \( RP_i \)'s projection points on approximated surface.

\[
\frac{1}{PN} \sum_{i=0}^{PN} \left( \frac{|RP_i - SP_i|}{h_{\text{max}} - h_{\text{min}}} \right)
\]

Figures 10 (a) shows the real image on Crater DEM, and Figure 12 (b), (c), and (d) are the produced triangulation by partitioning algorithm from it. Figure 12 (d) is a 6 × 6 blocked triangulation image with only 471 points. However
the image has the most important feature of the original image such as peaks, pits, ridges, and valleys.

Figure 12. (a) The original image of the Crater DEM (154,224pts) (b) A simplified triangulation (52,800pts, 48 x 48) (c) A simplified triangulation (4,711pts, 24 x 24) (d) A simplified triangulation (471pts, 6 x 6)

5. Conclusion

In this paper we presented the new partitioning approach to simplify a grid height data into a reduced point set of TIN. By the experiments, the performance is improved greatly and the mean error of a reconstructed triangulation is as low as the greedy insertion algorithms by partitioning method. And it is clear that to partition the terrain into smaller blocks is the way to improve the cache hit rate. Partitioning algorithm reduces the input points that are referenced for each path to add a point into triangulation in simplification sequence. In the future, we will develop a parallel simplification algorithm using the potential parallelism in our partitioning method. Since each block can be simplified in parallel, if we avoid the conflict of triangulation between the blocks.

References