Chapter 3.
Triangulation Hierarchies

The trouble with most surface triangulation algorithms, as described in the previous chapter, is that they add triangle edges without considering the effect of these edges on the third dimension. Many surfaces — especially natural ones — form continuous linear patterns which aren't adequately represented by isolated points. This may be clearly seen in the hypsoshaded rendering[1] of digital elevation data shown in figure 3.1. These linear features form critical lines which are just as important as surface points in describing a surface.

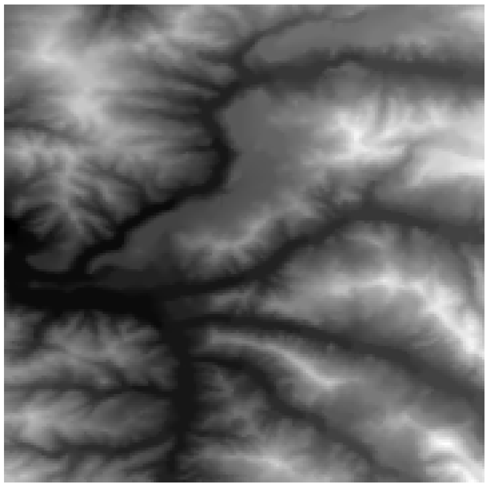
Figure 3.1. Hypsoshading reveals coherence of cartographic features

To retain integrity of the surface, a triangulation algorithm must consider the surface topology in determining placement of triangle edges. In this chapter I describe two hierarchical triangulations

1. Hypsoshading assigns a grey level to every elevation post, ranging from black at the lowest elevations to white at the highest elevations.

that I have developed which do consider surface topology. The first, a refined triangulation hierarchy, is based on Kirkpatrick’s point search strategy [Kir83]. It triangulates the surface in a bottom-up fashion to produce progressively coarser levels of detail. Critical points and edges in the model play a key role in deciding how points are eliminated and polygons are re-triangulated. Alternatively the second method, adaptive hierarchical triangulation, uses a top-down approach producing progressively finer levels of detail. Again, estimated locations of critical points and edges determine how these levels of detail are produced. This chapter describes the strategies and structures of both.

3.1. Refined triangulation hierarchy

The refined triangulation hierarchy builds triangulated levels of detail in a bottom-up manner, preserving the most critical points and approximating critical lines at each level. I began this work to meet the requirements described in [TSDB88] and outlined earlier in section 1.1.1.

3.1.1. Foundations

The refined triangulation hierarchy is based on a structure described by Kirkpatrick [Kir83] which was originally intended to facilitate optimal search in planar subdivisions. In his paper, Kirkpatrick demonstrated that his model facilitates O(n log n) search time on a structure built in O(n log n) time using O(n log n) storage space. I extended Kirkpatrick’s model to consider the special characteristics and requirements of terrain data bases. First, I enhanced his temporary search structure with additional data to facilitate fast level of detail retrieval and neighborhood search. These additional structures are described later on in this section. More importantly, I devised a means of measuring the importance of points to the terrain data base and a method for removing those points from the data base. The re-triangulation step respects critical line information, using triangle edges to generalize critical lines.

3.1.2. Input Data

This bottom-up algorithm requires an initial triangulation to start with. Because only coarser levels of detail are generated, this method can only preserve accuracy; it cannot improve on the accuracy of the initial triangulation. Yet any of the sources described in chapter 2.1 may be used to derive this initial surface.

If the data source is a gridded digital elevation model, the initial triangulation may be generated by creating two triangles for every neighborhood of four points. The orientation of these triangles (i.e. the direction of the diagonal) may be determined by considering both possibilities with respect to the bilinear surface patch defined by the four points.

This algorithm also expects critical line information, defining which edges in the initial triangulation are critical to the surface. These are the ridge lines, channel lines, and break lines discussed in chapter 2.2. Some data sources automatically provide critical edge information, such as ridge and drainage lines from digital line graph products. For an initial triangulation derived from contours, critical lines may be derived by first finding the slope lines, which intersect contour lines at right angles. A ridge is then a slope line from a peak descending to a pass, and then climbing up to another peak [D86]. Likewise, a channel is a slope line from a pit to a pale to another pit. For gridded data, numerous methods have been suggested for finding critical lines based on local neighborhoods [FL79, PD75, FB89, Sca90a].

3.1.3. Approach

A refined triangulation hierarchy is a rooted graph, where graph nodes represent triangular facets that approximate the terrain. Each depth level in the graph represents a single level of detail. Hence the collection of nodes at a given depth completely covers the area of interest with an irregular triangular mesh. Edges within the graph link triangles with the next higher and lower levels of detail. Because this structure is an acyclic graph, and not a tree, a triangle may have more than one parent in a coarser level of detail. The algorithm that generates this appears in figure 3.2.

3.1.3.1. Algorithm

The refined triangulation hierarchy is based on a number of key concepts which are defined here.

Point Significance is based on 1) the difference between the elevation of a point and the weighted average of adjacent vertices, and 2) the number of adjacencies for that point. The first measure assures that removal of that point will not significantly alter the geometry of the model, thereby retaining significant features. The second measure is used to choose between two points that have equal

```
procedure build_refined_triangulation_hierarchy (Point, Edge, $T_0$ )
Input :  A fine level-of-detail triangulation $T_0$ that references an Edge list —
            where critical edges are marked — and a Point set defining
            the surface.
Output :    $n$ triangulated levels of detail [$T_0$ .. $T_1$ ].
begin
    initialize adjacency list Adj[p] for each point $p$      Point ;
    level $n$ = 0;
    while $T_n$ can be reduced further do begin (*build next level of detail *)
        $n$ ++; (* starting next level of detail $n$ *)
        initialize $T_n$ with values from $T_{n-1}$ ;
        for each point $p$      Point that is not on the convex hull do begin
            calculate point significance value & store in Sig_List ;
            $p$ is OK to remove;
        end;
        sort Sig_List on point significance values in ascending order;
        for each point $p$ in Sig_List order do
            if it's OK to remove $p$ then begin (* remove $p$ from triangulation *)
                find polygon Poly which is the composite of all triangles at $p$ ;
                re-triangulate Poly , generalizing critical edges in Edge ;
            update triangulation $T_n$ and affected adjacency lists;
                for each point $q$      Adj[p] do begin
                    update Adj[q] ;
                    it's NOT OK to remove $q$ at this level of detail;
                end;
            end;
    end;
end;
```

Figure 3.2. Building the refined triangulation hierarchy

significance to the model. Removing one point from a triangulation reduces the number of triangles by no more than two, so more triangles may be removed if points with fewer adjacencies are selected. Maximizing the number of triangles removed causes refinement to occur more quickly, generates fewer hierarchical levels, and thereby decreases search time in the final structure.

Once significance factors have been determined for all of the points, the list of factors is sorted in ascending order. This sorted list acts as a queue of points to be removed from the current triangulation. However, not all points on the queue may be removed. This is because a point's significance is based on its relationship to its adjacencies. When a point is removed, its adjacencies must remain in the triangulation. Hence, points along the perimeter of a re-triangulated polygon are marked as "not OK" for that level, to ensure that they (and the new triangulation) remain in the current level of detail. Corner points on the convex hull of the triangulated area are never "OK" to remove.

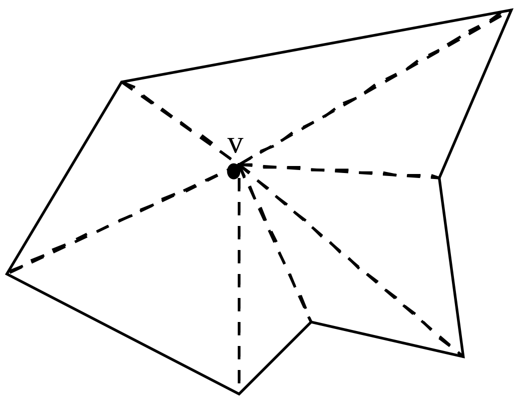

Figure 3.3. A star polygon is the union of all triangles that share vertex v

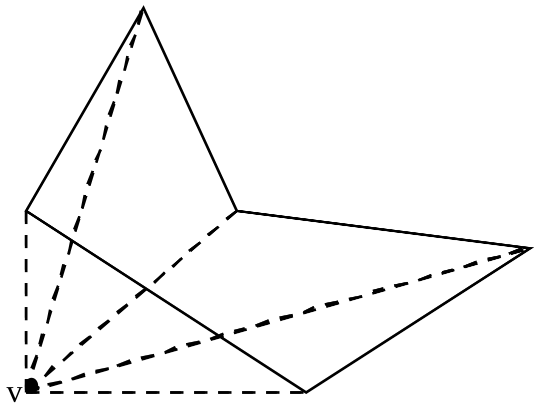

Figure 3.4. A corner point does not form the centroid of a star polygon

Star Polygons defined by the vertices are key to this algorithm. Each vertex v V can be thought of as defining a star polygon P_v. As shown in figure 3.3, v is the centroid of this polygon, and vertices adjacent to v form P_v's perimeter. Corner points which lie outside the perimeter of their neighbors, as shown in figure 3.4, form the exception and do not define polygons. Points that lie on the perimeter of their polygon form a special case that is easily handled.

Triangulation Criteria determine how to re-connect the points along the perimeter of a star polygon after its centroid is removed. It is desirable to retain the general shapes of critical lines, as they characterize terrain topology. However, if the initial triangulation represents the terrain economically and accurately, then nearly every vertex will lie on a critical line. The problem is to retain the shapes of these lines while removing points from them. I solve this problem, relying on critical line information in edge list *E*, as follows.

First, find critical edges for which *v* is an endpoint, and try to connect the other endpoints of these critical edges to one another. If several possible combinations exist, new connections that approximate the old critical lines are given preference. Figure 3.5 shows how this would be done in one case. Successful connections of critical edge endpoints are marked as new critical edges in the tri-

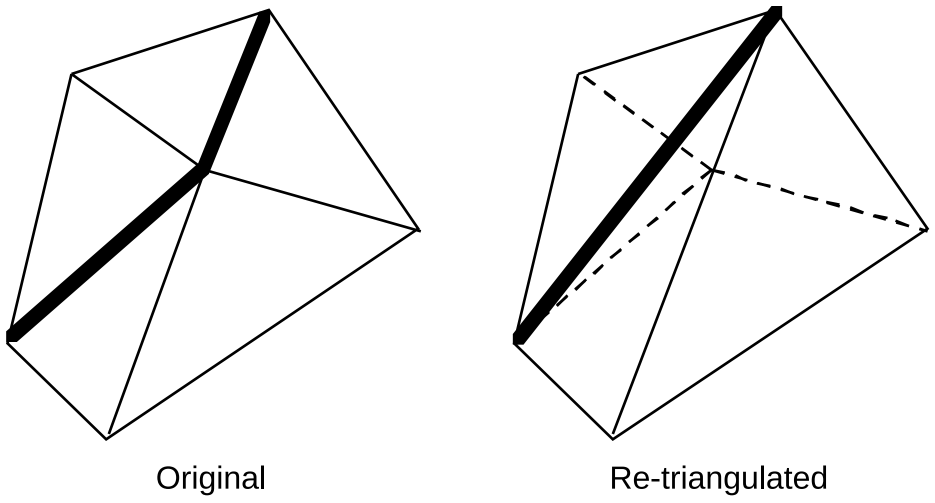

Figure 3.5. Re-triangulation approximates critical lines

angulation. These new connections slice the polygon into smaller polygons. After all critical edges have been considered, the smaller star polygons are simply triangulated with an O(n log n) algorithm [FM84].

3.1.3.2. Data Structures

The final triangulation hierarchy contains O(log n) levels of detail. This hierarchy is supplemented with additional information to facilitate visualization and analysis of the resulting data base. A header record indicates the area of coverage and contains pointers to the various resolution levels, and the starting records for the point, triangle, and polygon data. A resolution level is characterized by the number of points at that level (N) and a list of indices representing the triangles and their neighbors at that resolution. Points contain 3D coordinate values, and are ordered in the file so that the first point is used by all resolution levels, and the last point is used only in the highest resolution. Thus, only points 1..N need to be retrieved for a given resolution level. Triangles, which are defined by 3 point indices each, are also arranged to reflect the order in which they were created. In addition, the triangle points to a polygon for tree traversal. The polygon is a collection of adjacent triangles, used to find the children of those triangles. This is necessary because a node may have more than one parent.

3.1.4. Results

I implemented this algorithm on a VAX 8530 and tested it on data sets taken from the Defense Mapping Agency’s Digital Terrain Elevation Data

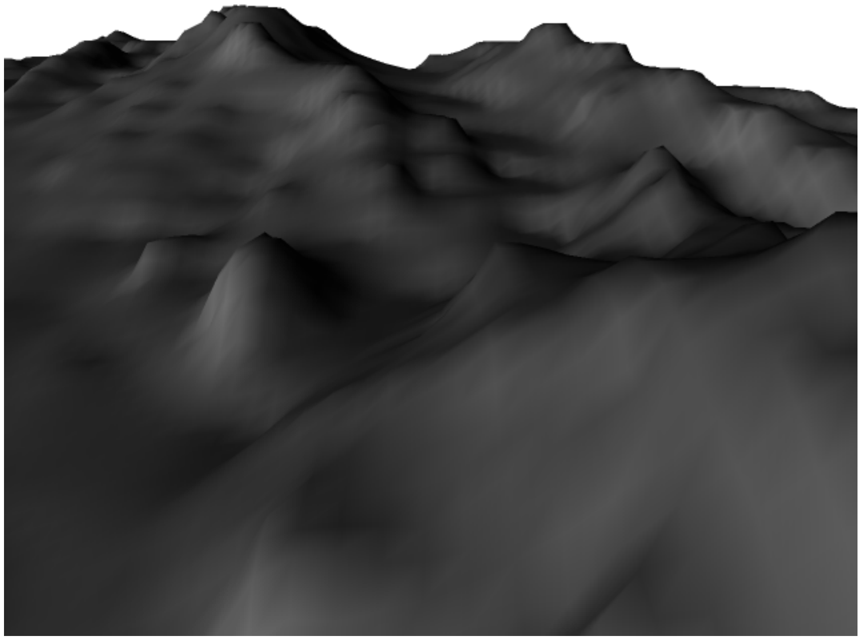

Figure 3.6. Original digital elevation model with 130,050 triangles

(DTED Level 1)[1]. I selected an area of interest in southern Nevada, replete with both large flat areas and mountainous regions, covered by 256x256 elevation posts. Figure 3.6 shows a view of the original data, represented by 130,050 triangles. A refined triangulation hierarchy was generated for this region in 7.75 CPU minutes containing 27 levels of detail, starting with an initial triangulation of critical lines containing 11,774 triangles.

To test my results, I selected three levels of detail from my refined triangulation hierarchy containing approximately 10,000, 5000, and 2500 triangles. These represent compression ratios (in terms of number of triangles) over the

1. DTED Level 1 elevation points are three seconds of arc apart, approximately 100 meters near the equator. Data accuracy is generally 30 meters vertically and 180 meters horizontally.

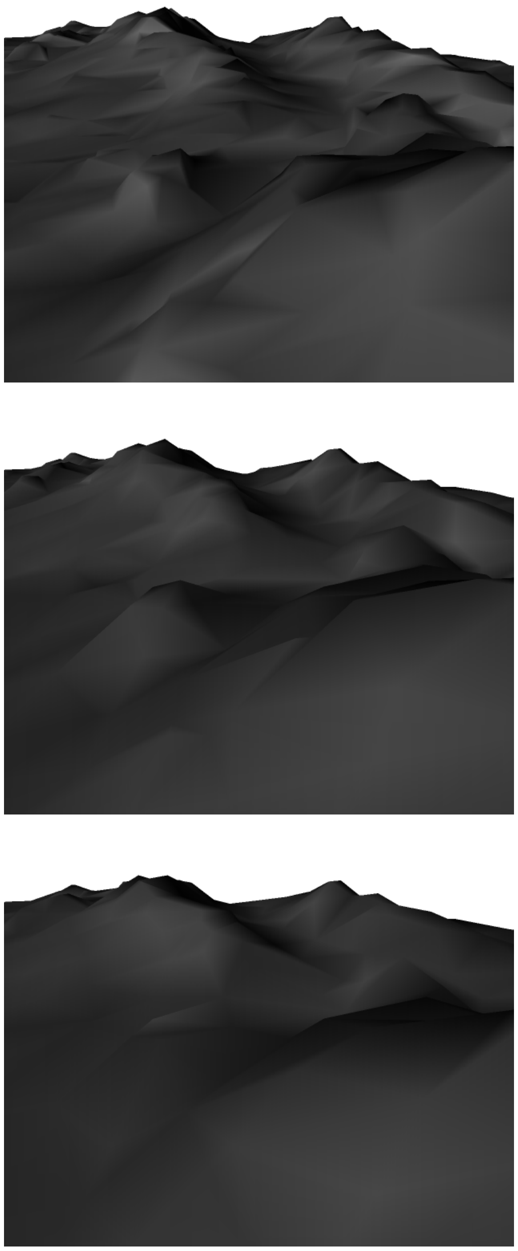

Figure 3.8. Three levels of detail from a refined triangulation hierarchy with approximately 10000, 5000, and 2500 triangles (top to bottom)

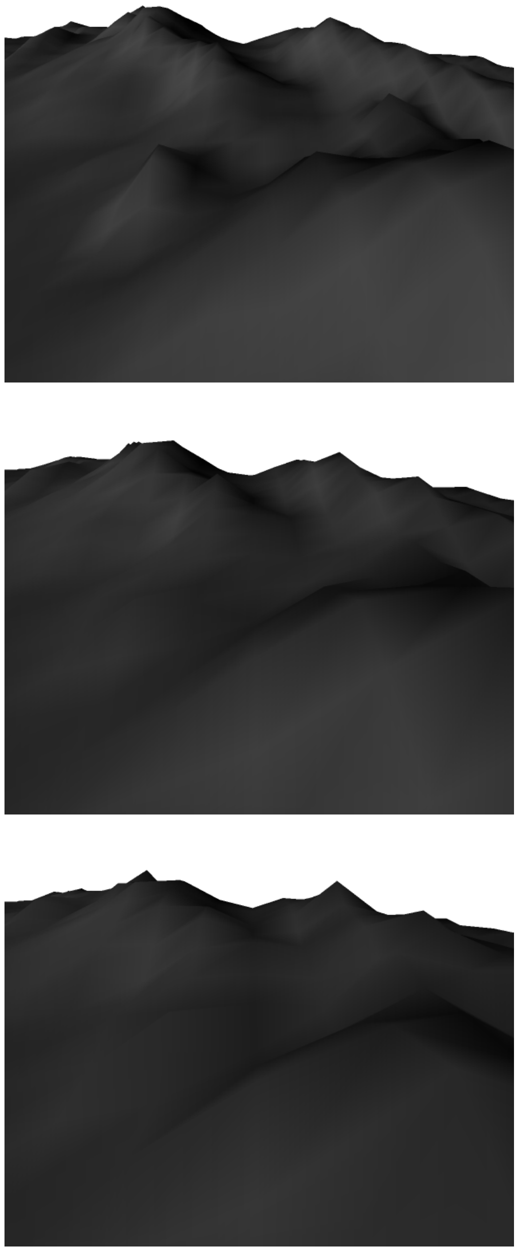

Figure 3.8. Three levels of detail from a subsampled grid with approximately 10000, 5000, and 2500 triangles (top to bottom)

original terrain model of 13:1, 26:1, and 52:1 respectively. Figure 3.7 shows these levels of detail with the coarsest representation in the bottom frame. As expected, the major features retain their basic positions and characteristics (compare to figure 3.6).

For comparison, I subsampled the elevation matrix to produce terrain models with approximately 10,000, 5000, and 2500 triangles. Figure 3.8 shows these models rendered from the same view seen in figures 3.6 and 3.7. Notice how the subsampled terrain features radically change shape and even shift across the landscape.

For a more analytic comparison of the models, I projected the original elevation data onto each of the six surface models and measured the distance (error). As shown in tables 3.1 and 3.2, levels of detail in the refined triangulation hierarchy had relatively small average and maximum errors. In contrast, errors in the subsampled grids were much larger, with errors generally ten times greater than those in the same-size TIN. This demonstrates that the levels of detail in the refined triangulation hierarchy retain critical information, producing a more accurate terrain representation with fewer triangles.

Number of Polygons	Terrain Models	
	TIN	Grid
10,000	8	75
5,000	9	94
2,500	11	130

Table 3.1. Average Error: Refined Triangulation (TIN) vs. Subsampled Grid at different levels of detail (error in meters)

Number of Polygons	Terrain Models	
	TIN	Grid
10,000	119	1260
5,000	125	1701
2,500	143	1774

Table 3.2. Maximum Error: Refined Triangulation (TIN) vs. Subsampled Grid at different levels of detail (error in meters)

3.1.5. Reflections

I am not the first — or last — to suggest building a triangulation hierarchy from the bottom-up. Several others have defined similar approaches since this work was first published. DeFloriani [DeF89], Lee [Lee91], and Schroeder et al. [SZL92] are examples, all having developed similar bottom-up heuristics for producing the TIN, which essentially discards the “least important" point at each stage.

What makes this work unique is that it also attempts to generalize critical lines in the refinement process. This approach minimizes the degradation of the surface model at coarser levels of detail. Yet there are drawbacks to this and the other bottom-up approaches. First, they do not allow for refinement down to a specified level of accuracy corresponding to each level of detail. Second, the algorithm produces more levels of detail than are generally needed. Finally, the resulting hierarchical structure is an acyclic graph, not a tree, which complicates the transition from one level of detail to another. It was these concerns that drove continuing research leading to the second solution.

3.2. Adaptive hierarchical triangulation

Adaptive hierarchical triangulation achieves the same surface integrity as the refined triangulation hierarchy, constructing levels of detail in a top-down fashion. Yet adaptive hierarchical triangulation has three advantages over the earlier approach. First, it requires no prior knowledge of which triangle edges and vertices are critical to the surface. Second, the resulting hierarchy is a tree which lends itself well to manipulations such as multi-resolution views. Third, the levels in the hierarchy comply with specified error tolerances.

In this structure, every hierarchy level corresponds to a different level of detail, approximating the surface within a given tolerance or maximum error. The top level is the coarsest, containing the smaller number of triangles which approximate the surface within some tolerance t_0. The i+1th level is related to the ith level as follows. Tolerance t_{i+1} is smaller than t_i. For each triangle T_j^i of the ith level there exists a set of triangles [$T_{j_1}^{i+1}..T_{j_n}^{i+1}$] at the i+1th level, such that

$$\bigcup_{k=1}^{n} T_{j_n}^{i+1} = T_j^i$$

The number of descendant triangles (n) can be any integer greater than or equal to one. The resulting tree structure provides the many benefits of a hierarchical organization while eliminating unnecessary intermediate triangulation levels. Each level of the hierarchy corresponds to a guaranteed level of accuracy. Because this algorithm focuses on the topology of a surface, it reduces the number of long and slivery triangles within each level of detail. Only triangles from the required levels of detail are saved, reducing the number of triangles that must be stored and searched. The tree structure of the adaptive hierarchical triangulation facilitates adaptive pruning and simplifies tasks such as rendering multiple-resolution views (see chapter 5). And aside from a few initial specifications, this algorithm produces the data base automatically. These features add up to a triangulation that provides good accuracy in a model that can be rapidly produced, searched, rendered, and otherwise manipulated.

3.2.1. Foundations

Adaptive hierarchical triangulation's top-down method was inspired by another hierarchical triangulation strategy described by DeFloriani et al. [DFNP84]. One advantage of DeFloriani's method is that it produces a tree structure that facilitates easier pruning and filtering via levels of detail, an important characteristic for time-critical applications [DM93]. Yet as illustrated in figure 2.3, simply splitting a triangle at the point of greatest error can generate a mass of unnecessarily slivery triangles that poorly approximate the surface.

My extension of this idea is based on two premises. First, that many of the critical points on a surface — especially terrain — are merely part of some larger critical line feature. Second, that for triangles that are sufficiently small, critical line features generally extend beyond the bounds of any one triangle. When a critical edge travels through a triangle, points significantly distant from the triangle surface should be found at the triangle edges as well as within the interior.

Adaptive hierarchical triangulation incorporates this idea as follows. It starts by calculating an error value for each point in the original elevation matrix. This error value represents the degree to which the triangulation deviates from the actual elevation model at that particular point. It then finds for each triangle four points with maximum error values: one inside the triangle, and one on each of the three edges. These points are then used to split the triangles that need refinement.

Figure 3.9 shows the five ways that a triangle may be refined. In all cases adaptive hierarchical triangulation approximates the surface form by splitting edges if necessary, thereby reducing the number of splits or refinements required

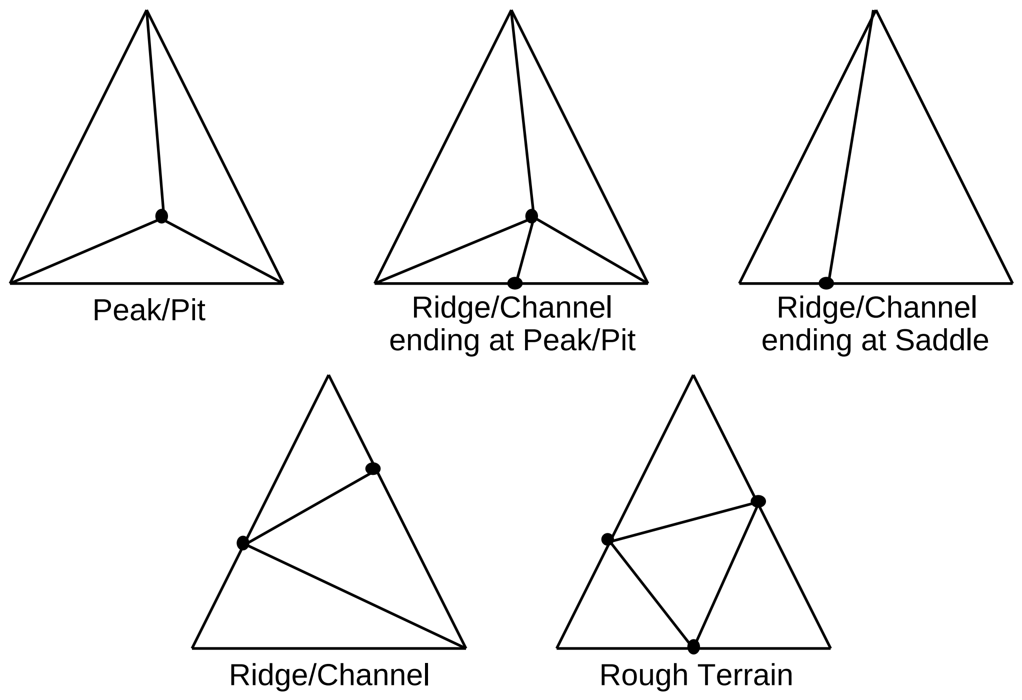
Peak/Pit
Ridge/Channel ending at Peak/Pit
Ridge/Channel ending at Saddle
Ridge/Channel
Rough Terrain

Figure 3.9. Triangle splitting strategies

to achieve a desired level of detail. If an isolated peak or pit resides within the triangle, it is split at that central peak or pit point as shown. If a single ridge or channel line travels up to that peak or pit, the triangle is split where that line crosses the edge of the triangle and at the central peak or pit point. If, however, a single ridge or channel line enters the triangle and ends at a saddle point or flat, then the center point is insignificant and the triangle is split by one edge as shown. If a ridge or channel line passes through the triangle, significant errors will be found on two edges of the triangle. A line connecting these points approximates the topographic line, and an additional edge splits the remaining quadrilateral into the least slivery pair of triangles possible. Finally, if a triangular patch corresponds to a rapidly fluctuating surface, it is likely that many of the points in that triangle have significant errors. Splitting this type of triangle on all edges partitions high-frequency regions which may then be further refined.

This refinement technique strives primarily to add edges that approximate critical lines on the surface, and secondarily to minimize the number of slivery triangles in the model. Of course, triangles with very sharp angles may be inevitable for some types of data. For example, the model of a steep cliff will have large differences in elevation values between adjacent points as illustrated in figure 3.10. Then triangles with very sharp angles are inevitable. When this occurs, the only way to reduce sliveriness and retain surface integrity is to split the slivers into smaller tringles with less acute angles.

3.2.2. Data

Adaptive hierarchical triangulation depends on surface data in a grid format to provide data for successively finer levels of detail. The grid format provides 2 advantages. First, a position on the surface identifies a single grid cell,

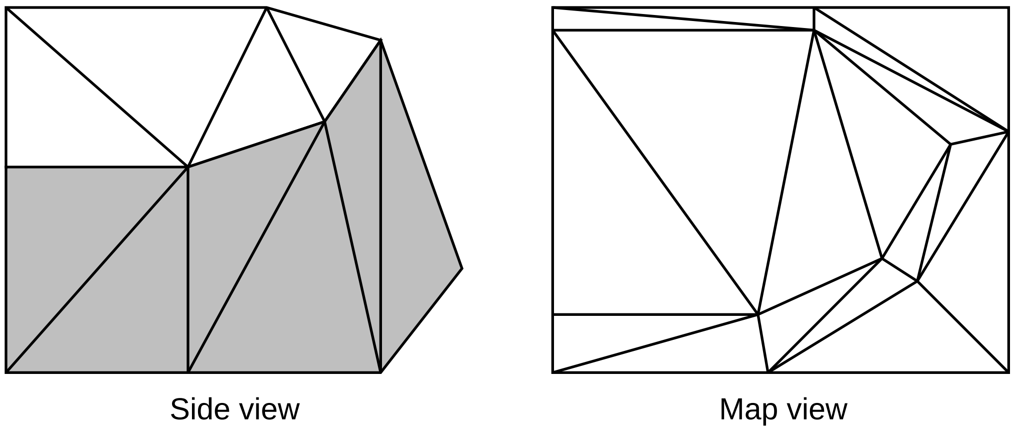

Figure 3.10. Some slivers in the model are inevitable, as in this triangulation (right) of a steep cliff (left)

which is known to correspond to exactly one point. Hence points may be indexed in constant time. Second, the nearest neighbors of a point are those points residing in neighboring grid squares. Furthermore, the neighborhood has a sense of direction (i.e. north, east, south or west of the point). Hence locality information may also be found in constant time.

Yet as described in chapter 2, not all data sources are in a grid format. Therefore it is useful to handle irregular data as well. I have overcome this problem by generating an adaptive grid structure for irregular data. There are two key issues to this problem. First, each grid square may contain no more than one point (although for irregular data, some grid cells will inevitably contain no points). Second, the grid structure works best if it is nearly square (i.e. it has the same number of rows and columns). The method works as follows.

First, determine the desired number of rows and columns, which is $\sqrt{n}$ for a surface defined by n points. Sort the points on their x coordinate, then begin collecting data in columns. A column may not contain two points with the same y coordinate, and ideally should consist of $\sqrt{n}$ points. Once the columns have been created, sort the points on their y coordinates and collect rows of points similarly. A point-to-grid and grid-to-point mapping is maintained for rapid searching while the triangulation is being constructed. Figure 3.11 shows an example of an irregular grid constructed in this manner.

In addition to the underlying discrete elevation data, adaptive hierarchical triangulation requires some initial triangulation. This may be as simple as splitting the area of interest in half. However, the initial triangulation is important because all edges introduced at this stage persist throughout all levels of detail. Initial triangulations are discussed at greater length in the next chapter.

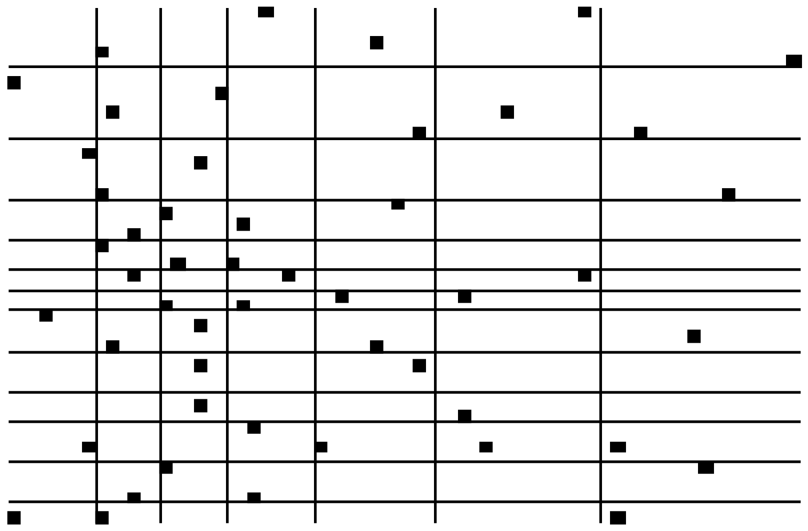

Figure 3.11. Adaptive grid structure for irregular data

3.2.3. Approach

Adaptive hierarchical triangulation produces a specified number of levels of detail, each with a specified accuracy requirement specified as an error tolerance given in meters. For each level of detail, the above steps are repeated until all triangles meet the given accuracy requirements for that level. This accuracy is checked by projecting all original grid points to the surface of the triangulated model and measuring the difference. Intermediate triangles, used to produce but not included in the final triangulation for the current level of detail, are discarded. This reduces the number of levels in the hierarchy and the number of triangles within each level, making faster search, display, and processing possible. For some applications such as Project 2851 [Luf89] there is a limit on the number of triangles allowed in each level of detail. In general these limits are imposed by the simulators that must render these levels of detail in real time. If such polygon constraints are more important than the level of error, adaptive hierarchical triangulation is easily modified to check the number of triangles and terminate a level when the limit is approached.

In addition to the elevation data in some grid and the level of detail information, this algorithm maintains several other data structures supporting development of the adaptive hierarchical triangulation. Each data point has an associated ground position and elevation, and is also associated with zero, one or two triangles in a membership list. These are the triangles that that point can split into smaller triangles. Triangle vertices are already part of the triangulation and hence cannot be added as splitting points. Therefore triangle vertices are associated with no triangles. Of the remaining points in the elevation matrix, a point that does not lie near any triangle edges is associated with the single triangle that the point's ground position lies within. A point that lies near an edge is associated with the two triangles that share that edge; a point is considered to be near the edge if the edge intersects its grid cell. The distance to that edge is stored along with the triangle associations. Because a point may be close to more than one edge, each membership entry is actually a linked list of triangle associations, sorted in order of increasing distances to a line.

Multiple triangle membership is important because lines in a triangulation frequently fall between grid points, thereby reducing the number of points on an edge. I compensate for this by considering points close to a line to be on that line. If several lines are close to a point, that point only belongs to the line closest to it. As shown in figure 3.12, splitting at these points will introduce minor bends along the edges. To retain a strict hierarchy, bends are restricted to ensure that they are less than a grid cell's width from the edge that was divided. By allowing

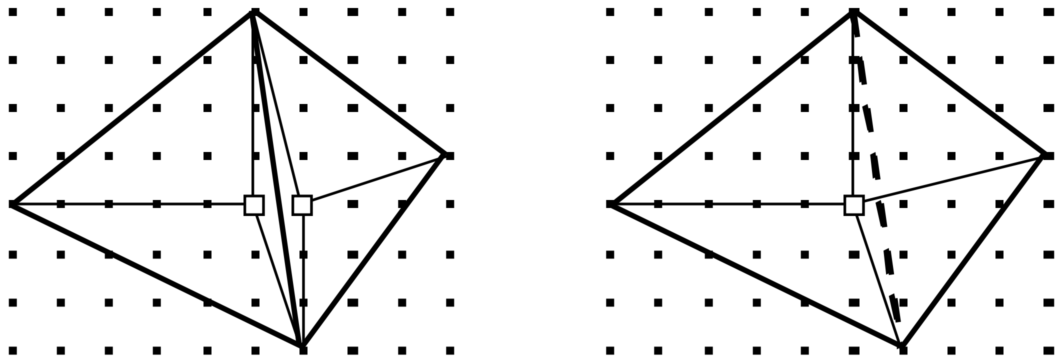

Figure 3.12. Allowing edges to bend, to avoid unnecessary slivery triangles

lines to bend I am able to introduce fewer points to the model than I would if those points were considered to be inside the triangles. This produces fewer and less slivery triangles.

Three point indices define a triangle. Each triangle is associated with a level of detail and contains pointers to its parent, its children, and its three neighbors (ie the triangles that share its edges). In addition, triangles have temporary structures keeping track of their splitting points, the maximum error found within them, and the number of edges to be split. A flag indicates whether the triangle meets the accuracy standards of the current level.

3.2.3.1. Main Program

Besides retrieving the input data and writing the results to a data base, the main program executes an outer loop once for each specified level of detail. At the start of iteration *i*, the current triangulation level *i* is a copy of level *i-1*, the finest level of detail generated thus far. When *i* = 1, this is the initial triangulation. At the conclusion of iteration *i*, triangulation level *i* is complete. All triangles at

```
procedure triangulate (Triangle, Point , L, E )
Input :  An initial triangulation Triangle that references a Point  set defining
         the surface, the desired number of levels of detail L  for the model,
         and a set of error tolerances E , one for each level.
Output :  L triangulations of the surface meeting the given error criteria.
begin
    initialize lists of 3 neighbors  for each triangle and 0..2 triangle memberships
        for each Point ;
    for  each level of detail i   do
        repeat
            for each triangle t     Triangle  do
                error[t] = measure_error (t, Point, membership );
        for each triangle t     Triangle  do
                for  each edge j  := 1 to 3  do
                    if error[t][j].distance  is significant then
                        ensure neighbor[t][j]  has significant error on that edge;
        for each triangle t     Triangle  with some significant error do
                case of significant error on
                1 edge : if error[t][4].distance  is significant then
                                split_1_edge_plus_center  (T, SP1, SP2 );
                            else
                                split_1_edge (T, SP1 );
                2 edges :   split_2_edges  (T, SP1, SP2 );
                3 edges :   split_3_edges  (T, SP1, SP2, SP3 );
                interior : split_center (T, SP1 );
                end;
            for each triangle t     Triangle  with no significant error do
                update neighbor  and membership  relations;
        until all errors are within E[i];
    write resulting hierarchy to data base;
end;
```

Figure 3.13. Adaptive hierarchical triangulation

this new level of detail are linked to parent triangles in level *i-1*. An inner loop iterates until all triangles in the current triangulation meet the specified accuracy requirements (error tolerance) of the current level of detail. Pseudocode for the main program appears in figure 3.13.

3.2.3.2. Measuring Errors

The level of error within a triangle is found by taking all grid points within the boundaries of that triangle and projecting them to the surface of the triangle. The aforementioned point membership list determines which triangle or edge to project each point to. An error value for a point is the difference between this projected elevation and the elevation given in the original matrix. Once the errors have been found, maximum error in each category is found in linear time. The algorithm shown in figure 3.14 outlines the method for finding these error values. Input to this algorithm is a triangle *T* for which errors are found. Output of this algorithm is an *Error* list of four points furthest from the plane of triangle *T*, and their respective distances to the triangle. These points determine how the triangle will be split (if at all) later on.

3.2.3.3. Significance of Error

Every point that yields the maximum error in some category for a triangle is a candidate splitting point. Whether or not that point is used depends on its significance. I employ two alternate strategies to determine significance of a point. My first strategy is to consider a point significant if its error is greater than the

```
function measure_error (T , Point, membership );
Input :     A triangle t,  the Point  set it's meant to approximate, and the
            membership  of each point.
Output :    Returns Error  for 4 regions of triangle T   (on each edge and within
            triangle interior), each represented by a point index  and measured
            error distance  for the point farthest from T  ‘s surface in that region.
begin
    for  i  := 1 to 4 do Error[i].distance := 0;
    calculate planar equation of triangle T ;
    for each p     Point   such that T     membership[p]  do begin
        calculate distance  from p  to its projection onto the plane of T ;
        case of  membership[p]  on
        1st edge :  if distance  > Error[1].distance  then begin
                        Error[1].index  := p ;
                        Error[1].distance  := distance ;
                    end;
        2nd edge :  if distance  > Error[2].distance  then begin
                        Error[2].index  := p ;
                        Error[2].distance  := distance ;
                    end;
        3rd edge :  if distance  > Error[3].distance  then begin
                        Error[3].index  := p ;
                        Error[3].distance  := distance ;
                    end;
        interior :  if distance  > Error[4].distance  then begin
                        Error[4].index  := p ;
                        Error[4].distance  := distance ;
                    end;
        end;
    end;
    return Error ;
end;
```

Figure 3.14. Measuring errors at the points

given error tolerance for the current level of detail. My alternate strategy is to consider a point significant if the given value is more than some percentage of the maximum error found within a triangle. In either case, a point is insignificant if its error falls at or below the threshold for the current level of detail. If a triangle has no significant error points in any of its four categories, then it is not split at that iteration. The main program ensures that if a point splits one triangle on its edge, then the neighboring triangle is also split on that edge.

3.2.3.4. Splitting Triangles

Although a triangle may be split in five different ways as shown in figure 3.9, the five routines that do this splitting are actually very similar. The algorithm below (figure 3.15) demonstrates how a triangle is split on three edges as an example. Input to the routine is a triangle *T* and indices of the three points — *SP1, SP2, SP3* — with maximum error on triangle *T*'s three edges.

All five triangle-splitting routines rely on two important local procedures for adding edges to the triangulation. One routine adds a new edge connecting two splitting points. Endpoints of the new edge are given null membership values, so those points no longer belong to any triangle and therefore are not candidates for splitting a triangle again. The second routine splits an existing edge into two new edges, possibly adding a bend in the middle. If the edge has not already been split (because the triangle sharing that edge was processed first), this routine adds two new edges by calling the other routine twice. Otherwise this routine finds the new neighbors added by the split and returns those. The primary task of both routines is updating point membership values alongside the new line.

```
procedure split_3_edges (T, SP1, SP2, SP3 );
Input :     A triangle t and 3 splitting points SP1, SP2, SP3 .
Output :    Adds elements to Triangle, updates neighbor and membership
            relations.
begin
    split 1st edge of T at point SP1;
    split 2nd edge of T at point SP2;
    split 3rd edge of T at point SP3;
    if any splitting point is co-linear with an edge it does not split then begin
    generate a polygon from the 3 vertices of T and SP1, SP2, SP3 ;
        split_anomalies (polygon );
    end;
    else begin (* typical triangle split *)
        add 3 new edges connecting splitting points SP1, SP2, and SP3 ;
        create 4 new Triangle entries;
        update neighbor and membership relations;
    end;
end;
```

Figure 3.15. Split triangle at all 3 edges

Each new triangle is assigned a distinct triangle index. I do not need to keep intermediate triangles, so to save space I recycle the index of the triangle that was split. After the new triangles have been generated, I update point memberships and triangle neighbors to facilitate splitting in the next iteration.

3.2.3.5. Special Triangulation

Associating points with nearby lines can introduce a rare anomaly which must be treated as a special case. Consider figure 3.16. When two edges of a triangle form a very obtuse angle, the points defining those edges are nearly co-linear. Because the algorithm may select a splitting point that is not quite on the

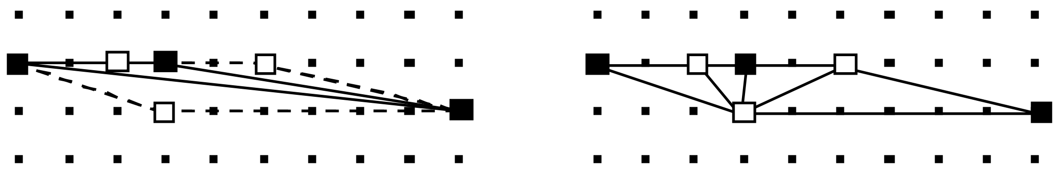

Figure 3.16. Very long and thin triangles can cause anomalies that must be handled specially by the triangulation algorithm

edge, the selected point may actually be co-linear with the adjacent edge as shown. When this happens, the aforementioned triangulation schemes will produce at least one “triangle” that is actually a line. Therefore a separate triangulation algorithm must be called.

The algorithm that handles this special triangulation appears in figure 3.17. A triangle’s vertices and splitting points are passed to the procedure as the vertices of a polygon to be triangulated. The procedure considers each group of

```
procedure split_anomalies (Polygon.);
Input :     A list of points defining a Polygon..
Output :    Adds elements to Triangle that triangulate the given Polygon.,
            updates neighbor and membership relations.
begin
    define previous and successor relationships for all points in Polygon.;
    while Polygon. is not a triangle or line do begin
        find sequence of 3 points on Polygon. forming the least slivery triangle;
    generate new Triangle with an edge connecting 1st & 3rd points;
        update previous and successor relationships;
    end;
    if remaining Polygon. is a triangle then
        generate new Triangle ;
    update neighbor and membership relations;
end;
```

Figure 3.17. Split long, thin triangles

three consecutive points on the polygon, selecting the triple forming the least slivery triangle. That triangle is then removed, leaving a polygon with one less vertex on its perimeter. This is repeated until the remaining polygon is a triangle. In my tests this special triangulation was required less than 0.1% of the time.

3.2.3.6. Asymptotic Analysis

The run time of this algorithm can be evaluated in terms of V, the number of vertices (points) in the original elevation matrix. Given that E is the number of edges and R is the number of regions in the graph, Euler's formula for a planar graph shows that

$$V - E + R = 2$$

In a triangulation, V may be partitioned into two categories, V_{in} and V_{out}, such that for all v V_{out}, v is a vertex on the outer boundary of the triangulation. For a rectangular area of interest where $V = m \times n$,

$$V_{out} = 2(m-1 + n-1) = 2m + 2n - 4.$$

A triangulation of all V vertices will produce a planar graph where R-1 regions are bounded by 3 edges, and the remaining outer region is bounded by V_{out} edges along the perimeter of the area of interest. Each edge bounds two regions, so

$$2E = 3(R-1) + V_{out}.$$

Therefore, the number of triangles T in a triangulation of all V points in the elevation matrix is

$$T = 2V - V_{out} - 2.$$

Because no triangulation may ever have a number of triangles T which is greater than 2V, T = O(V).

In the main program, initialization takes O(V) time to initialize point memberships and neighbor values. The outer loop executes a constant number of times (specified by a user). Within the next loop, *measure_error* is called once for each triangle in the current triangulation. Yet the end result of all these procedure calls is that each point in the original elevation matrix is projected to the plane of a triangle — once, twice, or not at all, depending on the number of triangles in the point's membership — and that result is compared to the original elevation value. Finding maximum error is done simultaneously, so the overall cost of all calls to *measure_error* is O(V). The next step compares the splitting points of each triangle to all three of its neighbors, which also takes O(V) time.

The innermost loop of the main program splits each triangle that does not meet the accuracy requirements for the current level of detail. For each triangle that is split — and there are O(V) of them — a constant number of new triangles is generated (between two and four) and a constant number of neighbor pointers is updated (three per new triangle). Adding edges simply updates the membership values of points in the original elevation matrix. Once again, over all iterations of the loop this is done once, twice, or not at all for each point in the matrix. Therefore, the time to run all iterations of this inner loop is also O(V).

Therefore, the overall run time of this algorithm is N x O(V) where N is the number of iterations of the repeat loop in the main program. In the worst case, this will start with an initial triangulation of two triangles and generate a single level of detail which contains all V vertices. Then, in each iteration of the repeat loop, every triangle will be split into two to four new triangles, until all points are included. This indicates that there may be up to O(log V) iterations of the repeat loop. Hence the worst case running time of the algorithm is O (V log V).

3.2.4. Data Structure

The adaptive hierarchical triangulation structure is designed to facilitate fast spatial search and data retrieval. It is composed of three parts: a header, a vertex list, and a triangle list.

The header indicates what levels of detail are available and how to retrieve them. Each level of detail is represented by an error tolerance, i.e. the maximum error allowed at that level of detail. It is accompanied by the number of vertices and triangles used in that level of detail, and a pointer to the sequence of triangles. A grid width/length is also stored for fast point location.

Each vertex is represented by three-dimensional coordinates which may be encoded to reduce storage space. All vertices used in level of detail I are also vertices of finer level of detail I+1. Vertices are listed in the order that they appear in the model, so that all N vertices used at a particular level of detail are stored contiguously (numbered 1..N). This improves data retrieval times for a single level of detail. Any vertex introduced as a splitting point on the edge of a coarser-level triangle also has a reference to the edge that it split. This is useful for zooming, as described in chapter 5.

Each triangle within the level of detail is defined by three references to a point list. Three neighbor references and a value representing twice the area of the triangle aid point location and line following within that level of detail . Child references and one parent reference aid traversal between levels of detail.

Triangles are ordered such that their indices indicate their relative positions on the surface. As with the vertices, all triangles defining a level of detail are stored contiguously for fast retrieval of a single level of detail.

Data retrieval is greatly simplified by this arrangement of the data. If only one level of detail is required, the following steps are taken:

- Read the header record to identify the desired level of detail, based either on error tolerance or number or triangles,
- Seek the start of the vertex list, and read in the appropriate number of vertices,
- Seek the first triangle in this level of detail, then read in the appropriate number of triangles.

The entire hierarchy may also be retrieved at once. To reduce the amount of data that this encompasses, a regular tiling scheme such as those proposed by Goodchild or Fekete [Goo89, Fek90] may be used to subdivide the globe.

3.2.5. Results

I selected eight areas of interest (AOI) as test data, each one covering 75x75 elevation points from the Defense Mapping Agency's DTED Level 1. I selected these areas to represent a variety of terrain types. Areas 1 and 2 cover an area of Alabama where plateaus rise steeply out of an otherwise flat region. Areas 3 and 4 represent parts of Montana with low rolling hills. Areas 5 and 6 depict a region in Nevada where the Rocky Mountains begin to ascend from the plains. Areas 7 and 8 feature Oregon's Cascade Mountain range. In each AOI a triangulation employing all 5625 contains 10,952 triangles.

I implemented the adaptive hierarchical triangulation program with varying parameters to see which behaved best. The first parameter determines how the significance of a point p's error e_p is evaluated. Error e_p may be considered significant compared to

- tolerance value t_i for level i, so that $e_p \geq t_i$, or
- a percentage N of the maximum error $e_{t_{max}}$ found for current triangle t, so that $e_p \geq e_{t_{max}}$.

The second parameter determines when a triangle should be split at both a significant center point and one edge (as shown in figure 3.9). Center point c may be considered significant compared to

- error e_v of splitting point v on the edge of the triangle, so that $e_c \geq e_v$, or
- the same value used to determine the significance of all other points, as determined by the first parameter.

These two parameters gave me four test options. Option 1 uses tolerance to determine significance, and requires a center point to be at least as significant as an edge point in order to be used. Option 2 uses 75% of the maximum error within a triangle to determine significance, and also requires a center point to be at least as significant as an edge point. Options 3 and 4 are like options 1 and 2 respectively, except that a center point's significance is determined by the usual measures. As a basis of comparison, I implemented DeFloriani's first algorithm [DFNP84] and ran it with the same test data.

All four options and DeFloriani et al.'s hierarchical triangulation algorithm [DFNP84] were executed on the eight AOIs, producing triangulations with a maximum error of 10 meters at the highest level of detail. For my algorithm, I chose to generate five levels of detail with error tolerances of 160, 80, 40, 20, and

10 meters. I measured the average sliveriness of the triangles, the average number of children for interior nodes, the number of triangles at the highest level of detail, and the total number of triangles in the hierarchy. I also generated several perspective views to visually verify the results.

Table 3.3 shows how slivery the resulting triangles were. I measured sliveriness of each triangle using the formula described in chapter 2.3, then calculated the average sliveriness for the entire structure. I divided the result by the sliveriness ratio for an equilateral triangle to show how close to equilateral the triangles were. The best results for each AOI — i.e. the least slivery triangulation, represented by the lowest sliveriness value — are shown in bold text in the table. On the average, most triangles have angles much sharper than sixty degrees. With the DeFloriani algorithm (DFNP), some angles are as small as 0.25 degrees. For all AOI's, adaptive hierarchical triangulation produced less slivery triangulations,

AOI	DFNP	Option 1	Option 2	Option 3	Option 4
1	32.294	**5.037**	5.071	6.301	6.578
2	52.487	**9.864**	12.107	11.074	11.107
3	35.889	6.047	**5.739**	6.398	5.998
4	50.682	11.619	**11.164**	12.581	12.854
5	56.835	15.054	**8.521**	14.329	8.676
6	40.932	**5.461**	5.578	7.376	7.437
7	51.261	**4.336**	5.029	6.089	7.367
8	39.925	**5.873**	6.112	6.805	7.153

Table 3.3. Measures of sliveriness, values normalized to 1 for an equilateral triangle

using all four options. Options 1 and 2 seem to work about equally well, indicating that the best measure of point significance is determined by data characteristics. Options 3 and 4 consistently performed a little worse. This led me to conclude that a center point should only be included in the division of a triangle if it is more significant than the edge point.

Search time in a hierarchical triangulation is determined by the number of levels that must be searched, and the number of child nodes that must be examined at each level. The number of levels in DeFloriani's structure depends on the number of iteration levels required to build the triangulation. Non-leaf nodes in this hierarchy have either two or three children. Adaptive hierarchical triangulation, on the other hand, guarantees a fixed number of levels in the hierarchy, but can split a parent triangle into any number of children. Table 3.4 shows that the

	Number of Levels		**Average Number of Children**	
AOI	**DFNP**	**Option 1**	**DFNP**	**Option 1**
1	15	5	2.5	2.8
2	17	5	2.4	2.4
3	17	5	2.5	3.6
4	17	5	2.5	3.6
5	19	5	2.4	2.4
6	18	5	2.4	2.5
7	17	5	2.3	2.4
8	18	5	2.5	2.3

Table 3.4. Comparison of hierarchies

average number of children in my hierarchy is not significantly larger than that in DeFloriani's structure. I expect search times to be comparable.

Table 3.5 shows the total number of triangles in the hierarchy. The best results for each test area — i.e those with the fewest total triangles — are shown in bold text. Notice that the options that produced the fewest total triangles also produced the least slivery triangles. Table 3.6 shows the number of triangles at the highest level of detail, with a maximum error of 10 meters at each point. This should be compared to 10,952 triangles for the original grid. As shown in the tables, adaptive hierarchical triangulation did not produce significantly fewer triangles than DeFloriani et al.'s algorithm.

Figures 3.18 and 3.19 illustrate the merits of the adaptive hierarchical triangulation. These figures show two different views of an area of interest in Oregon (AOI 7) modeled with (a) the original grid data, (b) DeFloriani et al.'s

AOI	**DFNP**	**Option 1**	**Option 2**	**Option 3**	**Option 4**
1	2918	**2848**	**2848**	3208	3195
2	4198	**3914**	4048	4344	4364
3	2576	1834	**1786**	2022	2055
4	2007	1563	**1539**	1737	1757
5	5433	5169	**5040**	5508	5372
6	3935	**3254**	3263	3624	3604
7	4908	**4520**	4611	4995	5109
8	7962	**7795**	7972	8312	8516

Table 3.5. Total number of triangles in the hierarchy.

AOI	DFNP	Option 1	Option 2	Option 3	Option 4
1	**1741**	1836	1832	1942	1935
2	2474	**2306**	2354	2442	2452
3	1547	1327	**1291**	1439	1470
4	1211	1129	**1125**	1196	1237
5	3185	**2986**	2992	3127	3135
6	2318	1979	**1974**	2167	2137
7	2883	**2644**	2673	2899	2901
8	4568	4334	**4323**	4436	4586

Table 3.6. Number of triangles in the finest level of detail (error tolerance = 10m)

hierarchical structure, and (c) adaptive hierarchical triangulation using option 1. The original grid data is represented by 10,952 triangles. The model generated with DeFloriani et al.'s hierarchical structure contains 2883 triangles for a maximum error of 10 meters. The adaptive hierarchical triangulation model contains 2644 triangles, also for a maximum error of 10 meters.

3.2.6. Observations

Adaptive hierarchical triangulation apparently has great potential for approximating surfaces with the desired accuracy — both mathematical and visual — using a reduced set of polygonal patches. Yet the outcome of this triangulation depends on the initial triangulation, which guides the development of successive levels of detail. This is because adaptive hierarchical triangulation only

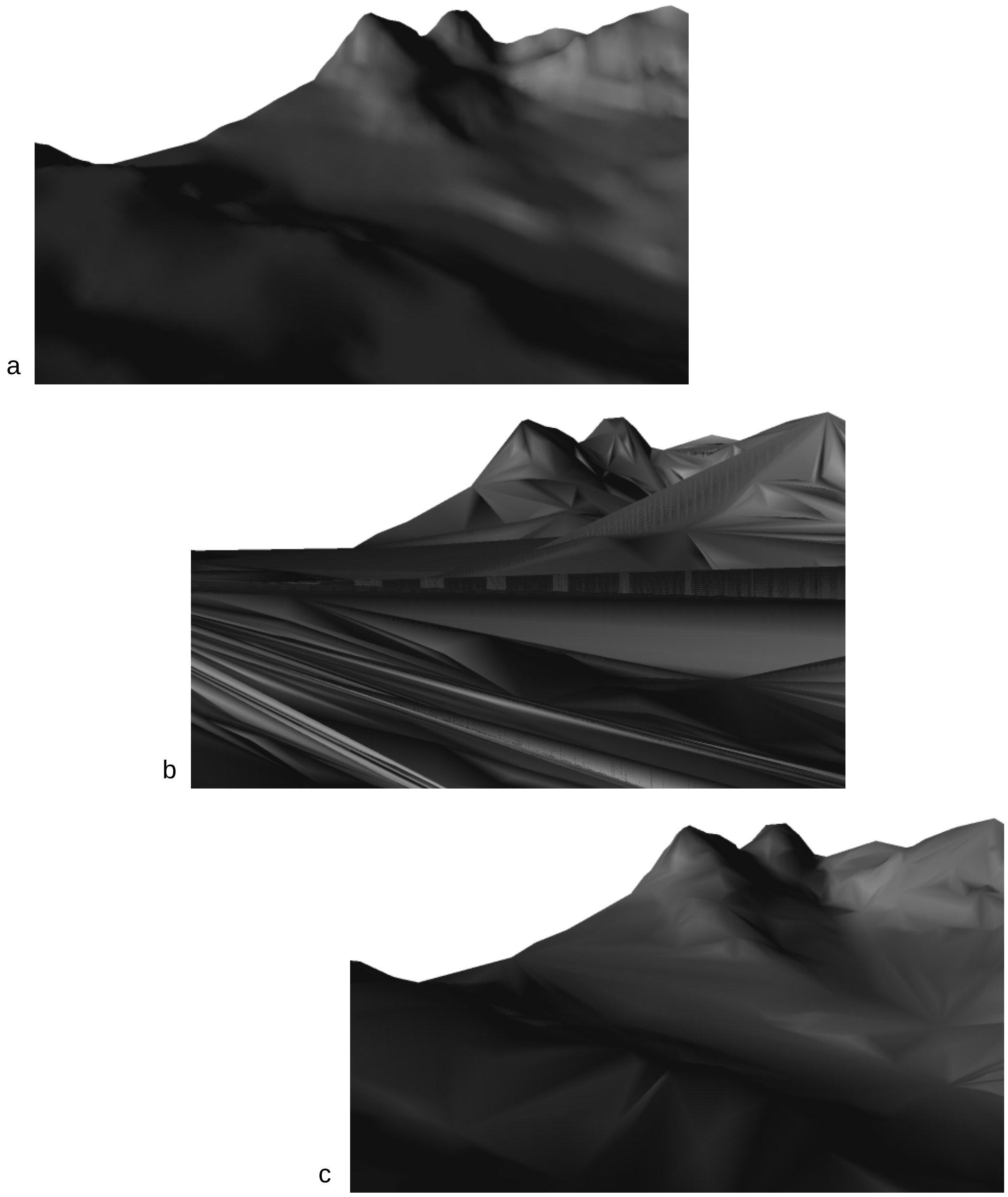

Figure 3.18. Perspective view of AOI 7 represented by
(a) original digital elevation model, (b) DeFloriani et al.'s hierarchical structure, and
(c) adaptive hierarchical triangulation

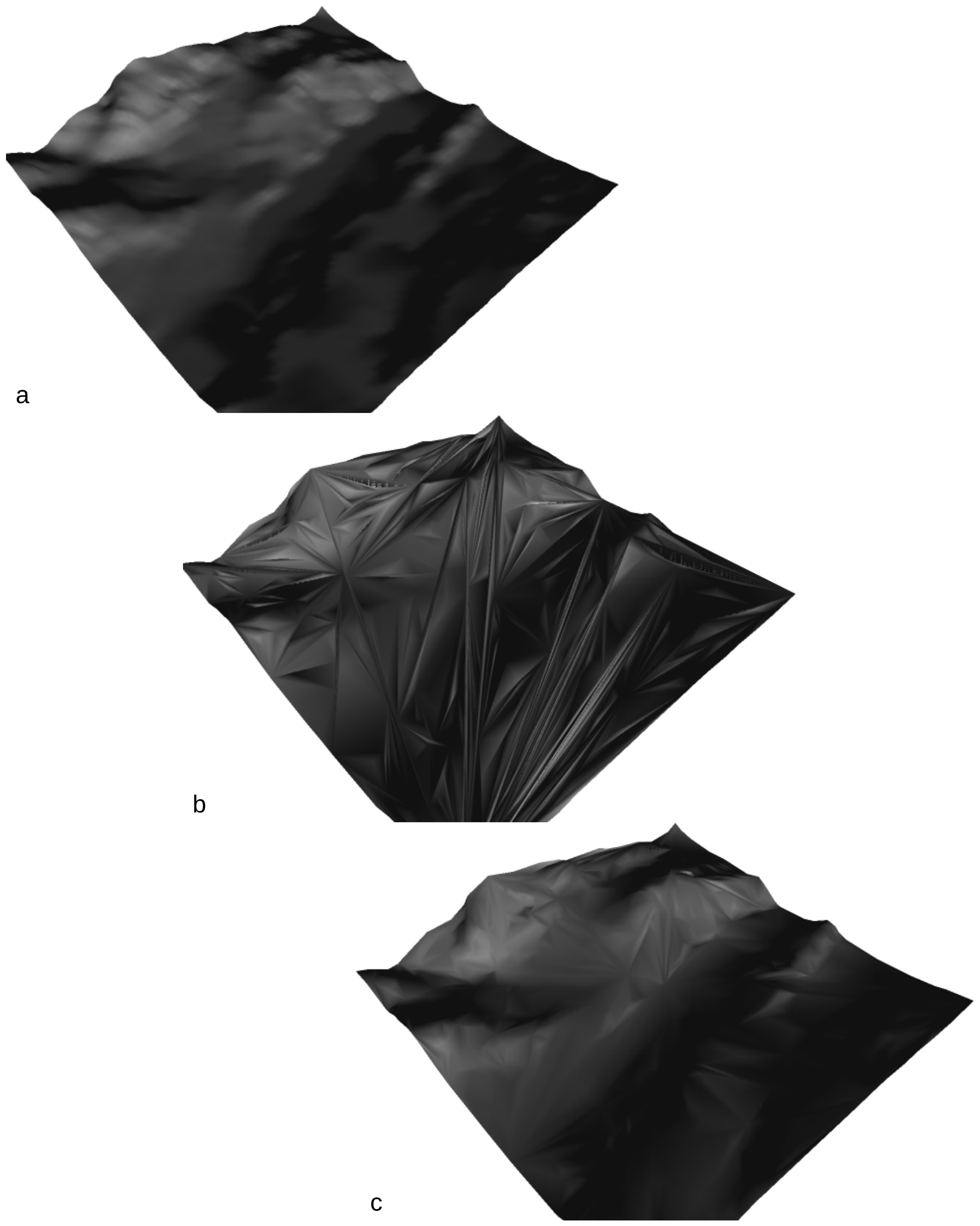

Figure 3.19. Long shot of AOI 7 represented by
(a) original digital elevation model, (b) DeFloriani et al.'s hierarchical structure, and
(c) adaptive hierarchical triangulation

approximates edges across existing triangles. Artifacts, such as edges cutting in the wrong direction, persist through all levels of detail. Furthermore, if the initial triangles are too large, covering an area that many critical lines travel through, this triangulation method can generate false critical lines across those triangles. Therefore the model would be greatly improved if a good initial triangulation could reliably be found. Ideally, the initial triangulation should not contain false edges or cover large areas of detail with disproportionately large triangles. This was the motivating factor for the research described in the next chapter.