Chapter 7.
Conclusions

This dissertation has addressed three current problems with spatial data representations. First is the need for data representations that support multiple scales and precisions without losing critical information. Second is a requirement for spatial operations to exploit filtering techniques to improve performance. Third is a desire for merging techniques that will allow different data representations to exist separately yet work together so that different data representations may be used to their best advantage.

Three triangulation methods were presented. The adaptive hierarchical triangulation algorithm generates a structure with fixed levels of detail with a specified accuracy. The tree structure of this triangulation hierarchy supports pruning and filtering, and is therefore the basis of the manipulation algorithms described in the remaining chapters. Curvature equalization improves existing triangulations by ensuring that smooth areas are represented by relatively few triangles, and rough areas are represented by many more. This method is used to produce a good initial tessellation for the adaptive hierarchical triangulation. A distinguishing characteristic of all three methods is that they attempt to generalize critical lines on the surface with the triangle edges.

Algorithms for three spatial operations exploiting the adaptive hierarchical triangulation's tree structure were given.These operations — zoom, multi-resolution viewing, and line of sight calculation — represent typical time-critical visualization and analysis applications.

Techniques for merging the adaptive hierarchical triangulation with other data representations were described. These, too, exploit tree structures to improve performance of the merging algorithms. A significant contribution here is the polygonal line sweep, which can find all triangles inside an area without having to examine them all.

The realm of spatial data representations is rich with possibilities which I have merely touched upon. My work is far from over. I propose several research directions that I plan to pursue in the near future.

7.1. Curvature equalization for geometric modeling

Curvature equalization has already been shown to improve terrain models. Yet it merits further study to make it suitable for geometric modeling of true three-dimensional surfaces including geometric CAD models. There are several avenues I wish to pursue.

First, a better estimator of the second partial derivatives is necessary. True rotation invariance is required for the modeling of surface with sharp discontinuities such as boxes, buildings, and machine tools.

I would also like to explore additional criteria for determining what is a "good" solution for the placement of vertices. Although the current curvature estimator does detect critical lines such as ridges and channels, it loses its sense of convexity versus concavity. This could be used to ensure that triangle edges don't cut across parallel break lines that are close to one another.

Another question to pursue is whether the three steps of curvature equalization should be combined into one and, if so, how. In my experiments with the range data, I discovered that several iterations of the steps produced better results. A combination of these steps might begin to approach the split-and-merge analogy for finding the optimal shape.

7.2. Surface fitting with conics

Until now my focus has been on surface models for applications where performance is more important than accuracy. Although some surfaces would be more accurately represented by curved surface patches, using the techniques described in chapters 3 and 4 produces polygonal models that are “good enough” for their intended purpose. Yet for applications where accuracy is more important than performance — such as computer aided design and some scientific visualization — polygonal surface models are not always good enough. I would like to extend the work described in this dissertation to apply to those applications.

The trouble with having to choose between polygonal and curved surface models is that each one will be better than the other in particular cases. Frequently the object being modeled will have both flat and curved surfaces, suggesting a need for a hybrid approach. I propose to extend my work on polygonal surface models for such an approach.

Pavlidis has devised such a hybrid approach for approximating two-dimensional curves [Pav82]. As described in this paper, he finds a mixture of conic arcs and straight line segments to fit a given set of data points. Starting with a polygonal approximation of the curve, his method fits higher-order approximations to the data only where it is needed. This technique produces approximations that are both precise and efficient in their representation of curves such as the outlines of characters in a given font.

A key aspect of Pavlidis' algorithm is his characterization of vertices as hard (unlikely to be part of a conic), break (where two conic sections meet), and soft (not hard). His criteria for making these characterizations is motivated by Pascal's theorem, essentially basing the decisions on the angles at each of the data points.

I propose to extend Pavlidis' method to work with surfaces. Beginning with a triangulation that has undergone curvature equalization, edges in the triangulation may be characterized as hard, break, or soft.

7.3. Criteria for judging surface models

A qualitative measure is needed to more accurately compare surface models for a given object. With such a measure, surface models could be ranked in terms of how well they represent given surfaces. This may then be used to objectively characterize which surface models are best in which circumstances.

There are many possible ways of measuring model quality, some of which were discussed in chapter 2. In general, criteria for judging surface models fall into three categories:

- Accuracy of the model - mathematical, visual, and topological;
- Size of the model - number of elements and actual disk space; and

- Processing speed - the time it takes to visualize, analyze, or otherwise process the surface model on commercial hardware platforms.

Here I present an initial framework for a qualitative measure of surface models based on criteria in all three categories. Further analysis will be required to refine its definition.

7.3.1. Accuracy of the model

In general, accuracy may be defined as follows: given a surface model and the “truth” that it represents, how great is the deviation between the two? There are three different ways of looking at this. First there is the mathematical aspect, using distance measures to attribute the model with some numeric accuracy factor. The second aspect has more to do with the overall integrity of the model: how much of the model is determined by actual surface characteristics, and how much is determined by extrinsic factors such as point of view, sampling rate, or heuristics. Third is the visual side of the issue, the subjective observation of how “good it looks”. I discuss all three aspects below.

7.3.1.1. Mathematical measures

Chapter 2 discusses several mathematical measures of accuracy that are typically used. Another common measure is root mean square error [KG91] which may be expressed as

$$\mu = \sqrt{\frac{\sum (z_i - z'_i)^2}{\sum z_i^2}}$$

The Defense Mapping Agency uses two methods to define accuracy [DMA90]: absolute accuracy and relative accuracy. Absolute accuracy is the statistical evaluation of both random and systematic errors yielding the uncertainty of a point with respect to some datum such as WGS84 (World Geodetic System 1984) or MSL (Mean Sea Level). It is expressed as an error measurement with some probability. Relative or point-to-point accuracy is the statistical evaluation of all random errors encountered in determining the position of one point with respect to another. This relative accuracy is expressed as an error measured over a specified distance with some probability

Each of these accuracy measures, absolute and relative, has two components: horizontal and vertical accuracy. Both are calculated assuming the normal distribution function. Vertical accuracy measures errors in elevation values, expressed as a linear error with 90% probability. With σ_x expressing sample standard deviations of elevation expressed in meters, vertical accuracy is calculated as

$$\mu = \pm 1.646\, \sigma_x.$$

Horizontal accuracy measures ground position errors, expressed as a circular error with 90% probability. With $\sigma_{x,y}$ expressing sample standard deviations measured in meters for latitude and longitude, horizontal accuracy is calculated as

$$\mu = \pm 1.073\, (\sigma_x + \sigma_y).$$

7.3.1.2. Integrity measures

Accuracy may also be measured by how well the model conforms to actual surface characteristics, i.e. whether vertices conform to critical points (pits, peaks, passes) and edges conform to critical lines (ridges, channels, break lines) on the surface. Another way to look at this is to consider how much of the resulting model is dictated by a particular orientation of view or other arbitrary factor that is independent of the object being modeled. For example, grid models contain points measured at regular intervals determined by some standard sampling rate, starting at a convenient geographic origin. If the samples are too far apart, this may easily miss important features on the surface. On the other hand, a finer sampling rate may produce far more points than are necessary to the model. Other modeling algorithms that add points and edges without regard to the surface topology are quadtree subdivisions — always splitting a region into four equal quadrants — and Delaunay-based triangulation methods — connecting nearby points on the projected plane to define planar patches that don't necessarily conform to the true surface.

Poor integrity in a model may be measured by considering the importance of each point in the mathematical accuracy measures described above. For example, a weight ω_i could be assigned to each point $p_i = (x_i, y_i, z_i)$ and then incorporated into the root mean square error estimation:

$$\mu = \sqrt{\frac{\sum \omega_i (z_i - z'_i)^2}{\sum \omega_i z_i^2}}$$

Weight ω_i may be a continuous value based on some combination of the Gaussian and mean curvature at each point, such as the combinations described in [MS88] or [GHL89]. Alternatively, ω_i may be a discrete value based on whether a point is a peak, pit, pass, ridge, ravine, slope, break, or flat point [PD75].

7.3.1.3. Visual measures

Visual accuracy measures how closely the model resembles reality. Although the simplest way to evaluate visual accuracy is by inspection, the results of inspection are difficult to quantify. More objective criteria are needed. Chapter 2 describes one measurement — sliveriness of the polygonal patches — that reflects visual quality of a model.

Shading also plays an important role in our perception of shapes. Continuous gradations of tone suggest smooth surfaces, while discontinuities in the color suggest discontinuities on the surface. This is the principle behind the smooth shading techniques (e.g. Gouraud, Phong) that let us get away with modeling smooth surfaces with polygons. This is also the reason why models with slivery polygons “look wrong”.

In computer graphics, shading of the model is calculated using surface normals. Therefore if the surface normals are wrong, the shading will look wrong. We suggest measuring the error of these normals as another means of qualifying visual accuracy.

The accuracy of the surface normal at some point may be measured in terms of the cosine of the angle between the true surface normal and the model’s surface normal. This will yield a value between 1 (the normals coincide) and -1 (the model normal points in the opposite direction). Accuracy of the normals over the entire model may then be taken as a maximum, average, or root mean square:

$$\mu = \sqrt{\frac{\sum (\cos\phi_i + 1)^2}{\sum 4}}$$

7.3.2. Size of the model

Chapter 2 describes the impact that size has on the performance of applications. Two aspects of model size are discussed: number of surface patches and storage space. Because both sizes are easily measured — and have been discussed at length earlier — no further discussion is warranted here.

7.3.3. Processing speed

Model representation and organization also impact an application’s performance. Chapter 2 describes why these are important. Here, I present some ideas on how to quantify these factors.

7.3.3.1. Model representation

Some model representations carry an extra overhead because pre-processing steps are required before they are usable by an application. Therefore model representation may be ranked in terms of the complexity of the algorithms that operate on the model. Complexity is generally expressed as O(f(n)), where n is the size of the input. A sequence of complexity expressions may be assigned discrete ranks where constant time algorithms score best, and exponential algorithms score worst. For example, the expressions ranked from best to worst could be

$$O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(n^n).$$

Complexity may also be measured as a value from 0 (worst) to 1 (best) using a continuous function based on the complexity. For example, an algorithm with O(f(n)) complexity could be given a rank of

$$^1/_{f(10)}.$$

7.3.3.2. Model organization

Model organization — which affects performance of spatial operations on that model —may also be measured in terms of algorithmic complexity. For example, one function common to many spatial operations is spatial search (such as “what triangle is this point in?”). therefore the complexity of spatial search operations will play a role in the quality measure of the surface model.

Data retrieval times — from disk to main memory — may also be measured in terms of the number of average number of seeks required, and the average number of blocks retrieved.

7.3.4. A qualitative measure of surface models

In order to effectively qualify surface models for comparison, the measures described above must be weighted and combined in some fashion. The quality measure — or figure of merit — must reflect tradeoffs between accuracy and performance. Because different applications have different requirements, this figure of merit must also be tunable.

We propose representing this figure of merit with a weighted sum, similar to the type used to grade exams and evaluate students in the school systems. First, the measures are normalized to a value from 0 (worst) to 1 (best). Second, these normalized accuracy measures are multiplied by the appropriate weights, and summed together. The appropriate weights depend on the application that the model is intended to support. For a more general measure, accuracy, size, processing speed should have equal weight. Finally, to get a recognizable score between 0 and 100, the result is divided by the sum of the weights, then multiplied by 100.

We proceed with an example. First, the measures are defined:

a_m =	mathematical accuracy based on RMS error with the error of each point weighted by its importance: $^1/_{(\mathrm{RMSE} + 1)}$,
a_s =	visual accuracy based on sliveriness S of the model: $^{4\pi}/_S$,
a_n =	accuracy of the normals based on RMS error of the cosines,
s_p =	size measured as number of polygons P in the model compared to number of polygons Q in the original data (assuming the entire original dataset were polygonized): $1 - ^{(P-1)}/_Q$,
s_d =	size measured as number of bytes B stored on the disk: $^1/_B$,
p_r =	processing speed as supported by the model representation, expressed in terms of the complexity of the k algorithms required where $O(f_i(n))$ is the complexity of the i^{th} algorithm and $g(n) = \max(f_1(n),.., f_k(n))$: $^1/_{g(10)}$,

p_{om} = processing speed as supported by the model organization in memory, expressed in terms of the complexity of the k algorithms required where $O(f_i(n))$ is the complexity of the i^{th} algorithm and $g(n) = \max(f_1(n),.., f_k(n))$: $^1/_{g(10)}$, and

p_{od} = processing speed as supported by the model organization on disk, expressed as the average number of seeks S required to retrieve a block of data: $^1/_S$.

Second, the weights for the measures are defined:

A_m = weighting factor for mathematical accuracy,

A_s = weighting factor for visual accuracy,

A_n = weighting factor for accuracy of the normals,

S_p = weighting factor for size in terms of number of polygons in the model,

S_d = weighting factor for size in terms of number of bytes stored on the disk,

P_r = weighting factor for processing speed as supported by the model representation,

P_{om} = weighting factor for processing speed as supported by the model organization in memory, and

P_{od} = weighting factor for processing speed as supported by the model organization on disk.

The final figure of merit M is calculated by

$$\frac{A_m a_m + A_s a_s + A_n a_n + S_p s_p + S_d s_d + P_r p_r + P_{om} p_{om} + P_{od} p_{od}}{A_m + A_s + A_n + S_p + S_d + P_r + P_{om} + P_{od}}$$

7.4. Dynamic terrain

Early on I stated that the data being modeled by these algorithms is relatively static. Yet this does not mean that they will never change. On the contrary, it is highly desirable to be able to adaptively update portions of the model to represent changes. Some of the applications that will require this are civil engineering packages (where, for example, roads are carved out of the landscape), battle simulations (where explosions and such modify local portions of the terrain), and scientific visualization (where the data is frequently changing over time).

Local changes such as these will generally be described with an irregular set of data points which may or may not have explicit interconnections (representing critical lines on the surface) specified. I believe that the adaptive hierarchical triangulation will readily accept such changes. Finding the area that needs to be changed is analogous to merging an area with the triangulation. The new points and lines need only be considered at the levels of detail where they cause the model to violate the error conditions. A new strategy needs to be devised for considering imposed critical lines in the triangulation, although I do not believe this will be terribly difficult. The consideration of edges is, after all, a key philosophy of this method.

7.5. Database integration

To date, the algorithms described in this dissertation have only been implemented on a small scale and tested on small datasets. To be truly useful to large spatial database applications such as simulations, civil engineering, and virtual reality, a more rigorous storage and retrieval scheme needs to be devised. This will include a paging strategy for retrieving pieces of the model as needed, and communications links with heterogeneous databases to make real-time merging of surface, vector, and raster data a reality.