A Survey of Polygonal Simplification Algorithms
UNC Technical Report TR97-045

David Luebke
Department of Computer Science
University of North Carolina at Chapel Hill

1. ABSTRACT

Polygonal simplification is at once a very current and a very old topic in computer graphics. As early as 1976 James Clark described the benefits of representing objects within a scene at several resolutions, and flight simulators have long used hand-crafted multi-resolution models of airplanes to guarantee a constant frame rate [Clark 76, Cosman 81]. Recent years have seen a flurry of research into generating such multi-resolution representations of objects automatically by simplifying the polygonal geometry of the object. This paper surveys the field of polygonal simplification, describing the current state-of-the-art as well as attempting to identify the major issues and trends in the field to date.

2. INTRODUCTION

Polygonal models currently dominate the field of interactive three-dimensional computer graphics. This is largely because their mathematical simplicity allows rapid rendering of polygonal datasets, which in turn has led to widely available polygon rendering hardware. In addition, polygons serve as a sort of lowest common denominator for computer models, since almost any model representation (spline, implicit-surface, volumetric) may be converted with arbitrary accuracy to a polygonal mesh. For these and other reasons, polygonal models are the most common representation for visualization of medical, scientific, and CAD datasets.

In many cases the complexity of such models exceeds the ability of graphics hardware to render them interactively. Three approaches are used to alleviate this problem:

- Augmenting the raw polygonal data to convey more visual detail per polygon. Gouraud shading and texture mapping fall into this category.
- Using information about the model to cull away large portions of the model that are occluded from the current viewpoint. The visibility processing approach described by Seth Teller and Carlo Sequin is an excellent example [Teller 91].
- *Polygonal simplification* methods simplify the polygonal geometry of small or distant objects to reduce the rendering cost without a significant loss in the visual content of the scene. These methods are the subject of this paper.

Note that terrains, or tessellated height fields, are a special category of polygonal models. The regularity and two-dimensional nature of these models simplify some aspects of the simplification problem; most of the problems facing researchers in polygonal simplification have been solved a year or two earlier for the restricted domain of terrain datasets. At the risk of injustice to some elegant work on terrains, this survey focuses on solutions that apply to the more general realm of polygonal meshes.

A bewildering variety of simplification techniques have appeared in the recent literature; the next section attempts to classify the important similarities and differences among these techniques. A catalog of ten published algorithms follows, briefly describing each approach and placing it in this taxonomy. The next sections discuss some important issues and trends in the field of simplification, and speculate on possible avenues for future work. A few informal remarks close the paper.

3. TAXONOMY

The various simplification approaches described in the computer graphics literature of the last five years can be categorized along several axes. Some algorithms simplify the scene by iteratively removing polygons while others collapse multiple vertices together, some algorithms preserve topology while others ignore it, and so on. Polygonal simplification is by no means a solved problem; this section identifies some of the important areas in which existing solutions differ or resemble each other.

3.1 Mechanism:

A primary classification of any simplification algorithm is the underlying mechanism it uses to remove polygons from the scene. Nearly every simplification technique in the literature uses some variation or combination of four basic polygon elision mechanisms: sampling, adaptive subdivision, decimation, and vertex merging.

- *Sampling* schemes begin by sampling the geometry of the initial model. These samples can be points on the 2-D manifold surfaces in the model or voxels in a 3-D grid superimposed upon the model. The algorithm then tries to create a polygonal simplification that closely matches the sampled data. Varying the number of samples taken regulates the accuracy of the created simplification.
- *Adaptive subdivision* approaches create a very simple polygonal approximation of the scene, called the *base model*. The base model consists of triangles or squares, shapes that lend themselves to recursive subdivision. This process of subdivision is applied to the base model until the resulting surface lies within some user-specified threshold of the original surface. Conceptually simple, adaptive subdivision methods suffer two disadvantages. First, creating the base model involves the very problem of polygonal simplification that the algorithm is attempting to solve. For this reason adaptive subdivision approaches have been more popular with the specialized case of terrains, whose base model is simple to calculate. Second, a recursive subdivision of the base model may not be able to capture the exact geometry of the original model, especially around sharp corners and "creases" in the mesh [Hoppe 96].
- *Decimation* techniques iteratively remove vertices or faces from the mesh, retriangulating the resulting hole after each step. This process continues until it reaches a user-specified degree of simplification. Since most decimation algorithms do not permit a vertex or face removal that will change the local topology of the mesh, the decimation process may be unable to effect high degrees of simplification.
- *Vertex merging* schemes operate by merging two or more vertices of a triangulated model together into a single vertex, which can in turn be merged with other vertices. Triangles whose corners have been collapsed together become degenerate and can be eliminated, decreasing the total polygon count. Vertex merging approaches do not necessarily require manifold topology, though some algorithms have made that assumption implicitly. These algorithms use a limited vertex merge called an *edge collapse*, in

which only the two vertices sharing an edge are collapsed in each operation.

3.2 View-dependence

A few recent algorithms have introduced a methodology that blurs the once-clear distinction between simplification and rendering. The new approach performs simplification based on the viewer's position and provides another fundamental categorization:

- *View-independent simplification*, the traditional approach, computes several *levels of detail* (LODs) for every object in the scene. At run-time the rendering system selects an appropriate LOD for each object, usually based on distance. As the object recedes from the viewer, the original or highest level of detail is replaced with progressively coarser LODs.
- *View-dependent simplification* replaces multiple static levels of detail with a single dynamic representation of the model. This structure is continuously queried at run-time to produce a polygonal tessellation appropriate to the user's current viewpoint. In a view-dependent system a single object can span multiple levels of simplification, with nearby portions of the object shown at higher resolution than distant regions.

3.3 Error Metric:

Simplification methods can also be characterized by how they use an error metric to regulate the quality of the simplification. A surprising number of algorithms use no metric at all, but simply require the user to run the algorithm with different settings and explicitly select appropriate LOD switching distances. For large databases, however, this degree of user intervention is simply not practical. Those algorithms that utilize an error metric to guide simplification fall into two categories:

- *Fidelity-based simplification* techniques allow the user to specify the desired fidelity of the simplification in some form, then attempt to minimize the number of polygons without violating that fidelity constraint. Different approaches measure fidelity in different ways; a common definition is the maximum distance of the simplified surface from the original surface. Adaptive subdivision algorithms lend themselves nicely to fidelity-based simplification, simply subdividing the base model until the fidelity requirement is met.
- *Polygon-budget simplification* systems take as input a target number of polygons and attempt to maximize the fidelity of the simplified model without exceeding the specified polygon budget. This approach is a natural fit for decimation techniques, which are designed to remove vertices or faces one at a time and merely need to halt upon reaching the target number of polygons. As mentioned above, however, topology constraints often prevent decimation algorithms from reducing the polygon count below a certain level.

To be most useful, a simplification algorithm needs to support both fidelity-based and polygon-budget operation. Fidelity-based approaches are crucial for generating accurate images, whereas polygon-budget approaches are important for time-critical rendering. The user may well require the both of these tasks in the same system.

3.4 Topology:

In the context of polygonal simplification, topology usually refers to the structure of a connected 2-D manifold, or mesh. The *local topology* of a face, edge, or vertex refers to the geometric structure of that feature's immediate neighborhood. A simplification algorithm that preserves local topology will not alter this geometric structure, and will preserve the genus of the simplified surface. *Global topology* refers to the geometric structure of the entire surface. A simplification algorithm preserves global topology if it preserves local topology and does not create self-intersections within the simplified object. A self-intersection, as the name implies, occurs when two non-adjacent faces intersect each other.

Many real-world CAD models contain objects that violate local or global topology. Since interactive visualization of CAD databases is a primary application of polygonal simplification, the behavior of the various approaches when encountering such models is an important characteristic. Simplification algorithms can be separated into two camps:

- *Topology-preserving algorithms* tend to exhibit good fidelity. Since they preserve the genus of the simplified object, no holes will disappear during simplification. As a result the opacity of the object seen from a distance tends to remain roughly constant. This also limits the simplification possible with a topology-preserving algorithm, however, since objects of high genus cannot be simplified below a certain number of polygons without closing holes in the model. In addition, a topology-preserving approach requires a mesh with valid topology to begin with. Some algorithms, such as [Schroeder 92] ignore regions in the mesh with invalid local topology, leaving the regions unsimplified, while others simply crash.
- *Topology-modifying algorithms* do not necessarily preserve local or global topology. The algorithms can therefore close up holes in the model as simplification progresses, permitting drastic simplification beyond the scope of topology-preserving schemes. This drastic simplification often comes at the price of poor fidelity, however, and distracting artifacts as holes disappear from one LOD to the next. Some topology-modifying algorithms do not require valid topology in the initial mesh, which greatly increases their utility in real-world CAD applications. Some topology-modifying algorithms attempt to regulate the change in topology, but most are *topology-insensitive*, paying no heed to the initial mesh connectivity at all.

4. CATALOG

The intent of this section is not to provide an exhaustive list of work in the field of polygonal simplification, nor to select the "best" published papers, but rather to briefly describe a few important algorithms that span the taxonomy presented above. Most of the papers chosen represent influential advances in the field; a few provide more careful treatment of existing ideas.

4.1 Decimation of Triangle Meshes

One of the first published algorithms to simplify general polygonal models, this paper by William J. Schroeder, Jonathan A. Zarge, and William E. Lorenson [Schroeder 92] coined the term "decimation" for iterative removal of vertices. Schroeder's decimation scheme is designed to operate on the output of the Marching Cubes algorithm for extracting isosurfaces from volumetric data [Lorenson 87], and is still the most commonly used algorithm for this purpose. Marching Cubes output is often heavily overtesselated, with coplanar regions divided into many more polygons than necessary, and Schroeder's algorithm excels at removing this redundant geometry.

The algorithm operates by making multiple passes over all the vertices in the model. During a pass, each vertex is considered for deletion. If the vertex can be removed without violating the local topology of the neighborhood, and if the resulting surface would lie within a user-specified distance of the unsimplified geometry, the vertex and all its associated triangles are deleted. This leaves a hole in the mesh, which is retriangulated using a loop-splitting algorithm. The algorithm continues to iterate over the vertices in the model until no more vertices can be removed.

Simplifications produced by the decimation algorithm possess an interesting feature: the vertices of the simplified model are a subset of the vertices of the original model. This is convenient for reusing normals and texture coordinates at the vertices, but can limit the fidelity of the simplifications, since the best approximation to the original surface can at times involve changing the positions of the vertices. The decimation algorithm accepts models with non-manifold vertices, but does not attempt to simplify those regions of the model.

4.2 Re-Tiling Polygonal Surfaces

Another of the first papers to address simplification of arbitrary polyhedral objects, this algorithm by Greg Turk [Turk 92] combines elements of the sampling and decimation mechanisms. The re-tiling algorithm works best on smoothly curved surfaces without sharp edges or discontinuities, preferring organic forms such as people or animals to mechanical shapes such as furniture or machine parts. Re-tiling provides a form of polygon-budget simplification by allowing the user to specify the number of vertices in the simplified model, but it is not obvious how to modify the algorithm to provide a fidelity metric.

The algorithm begins by randomly distributing the user-specified number of vertices over the surface of the model. The algorithm then simulates repulsion forces between the vertices, allowing nearby vertices to repel each other. Since the vertices are constrained to move within the surface, this repulsion tends to redistribute the randomly scattered vertices evenly across the surface. Next, the algorithm uses a method called *mutual tessellation* to construct an intermediate surface that contains both the new and original vertices. A local re-triangulation is applied to improve the aspect ratio of the resulting triangles. Finally, the original vertices are decimated from this surface, leaving the re-tiled model composed of the new vertices.

Among the contributions of this paper was the introduction of a method to smoothly interpolate between different levels of detail, a process for which Hughes Hoppe has since used the term "geomorph" [Hoppe 96].

4.3 Multi-Resolution 3D Approximations for Rendering Complex Scenes

This vertex-merging algorithm by Jarek Rossignac and Paul Borrel is one of the few schemes that neither requires nor preserves valid topology. The algorithm can therefore deal robustly with degenerate models with which other approaches have little or no success. This is a tremendous advantage for simplification of handcrafted CAD databases, a notoriously messy category of models.

The algorithm begins by assigning a perceptual importance to each vertex based upon two factors. Vertices associated with large faces are considered more important than vertices associated only with small faces, and vertices of high curvature (measured by the inverse of the maximum angle between any pair of edges incident to the vertex) are considered more important than vertices of low curvature. Next a three-dimensional grid is overlaid on the model and all vertices within each cell of the grid are collapsed to a single *representative vertex* for the cell, chosen according to the importance weighting calculated in the first step. The resolution of this grid determines the quality of the resulting simplification; a coarse grid will aggressively simplify the model while a fine grid will perform only minimal reduction. In the process of clustering, triangles whose corners are collapsed together become degenerate and disappear.

One unique feature of the Rossignac-Borrel algorithm is the fashion in which it treats these triangles. Reasoning that a triangle with two corners collapsed is simply a line and a triangle with three corners collapsed is simply a point, the authors choose to render such triangles using the line and point primitives of the graphics hardware, filtering out redundant lines and points. Thus a simplification of a polygonal object will generally be a collection of polygons, lines, and points. The resulting simplifications are therefore more accurate from a schematic than a strictly geometric standpoint. For the purposes of drastic simplification, however, the lines and points can contribute significantly to the recognizability of the object.

In addition to its inherent robustness, the Rossignac-Borrel algorithm can be implemented very efficiently and is one of the fastest algorithms known. However, the method suffers several disadvantages. Since topology is not preserved and no explicit error bounds with respect to the surface are guaranteed, the resulting simplifications can be less visually pleasing than those of slower algorithms. In addition, the simplification is sensitive to the orientation of the clustering grid, so two identical objects at different orientations can produce quite different simplifications. Finally, the algorithm does not lend itself to either fidelity-based or polygon-budget simplification, since the only way to predict how many triangles an LOD will have using a specified grid resolution is to perform the simplification.

4.4 Model Simplification Using Vertex Clustering

Kok-Lim Low and Tiow-Seng Tan have carefully examined the Rossignac-Borrel algorithm and invented a revised version that addresses some of these shortcomings [Low 97]. Observing that the spatial binning invoked by the 3-D grid is simply a form of vertex clustering, Low and Tan introduce a different clustering approach they call *floating-cell clustering*. In this approach the vertices are ranked by importance and a cell of user-specified size is centered on the most important vertex. All vertices falling within the cell are collapsed to the representative vertex and degenerate triangles are filtered out as in the Rossignac-Borrel scheme. The most important remaining vertex becomes the center of the next cell and the process is repeated. By eliminating the underlying grid, floating-cell clustering greatly reduces the sensitivity of the simplification to the position and orientation of the model. In addition, floating-cell simplification results vary less with cell size than the results of the uniform-subdivision approach.

Low and Tan also improve upon the criteria used for calculating vertex importance. Let θ be the maximum angle between all pairs of edges incident to a vertex. Though Rossignac and Borrel used $1/\theta$ to estimate the probability that the vertex lies on the silhouette, Low and Tan argue that $\cos(\theta/2)$ provides a better estimate.

In addition, Low and Tan extend the concept of drawing degenerate triangles as lines, calculating an approximate width for those lines based on the vertices being clustered and drawing the line using the thick-line primitive present in most graphics systems. The appearance of these lines is further improved by giving the line a normal to be shaded by the standard graphics lighting computations. This normal is dynamically assigned at run-time to give the line a cylinder-like appearance.

Finally, Low and Tan address the lack of a fidelity metric in the original algorithm by noting that the clustering size used to create an LOD can be related to the maximum number of pixels each cluster can cover. This provides a rough fidelity metric, allowing the user to specify that no LOD will be used unless it clusters only vertices within *n* pixels of each other. This observation has been made elsewhere and forms the basis for the Luebke-Erikson algorithm, described next.

4.5 View-Dependent Simplification of Arbitrary Polygonal Environments

This vertex-merging algorithm by David Luebke and Carl Erikson is one of the first to provide interactive view-dependent simplification of arbitrary polygonal scenes [Luebke 97]. The algorithm, referred to as *Hierarchical Dynamic Simplification* or HDS, was designed for visualization of very complex CAD models and, like the Rossignac-Borrel approach, neither requires nor preserves manifold topology. Rather than representing the scene as a collection of objects, each at several levels of detail, in the HDS algorithm the entire model comprises a single large data structure. This structure is the *vertex tree*, a hierarchy of vertex clusters which is queried to generate a simplified scene.

The entire system is dynamic; nodes to be collapsed or expanded are continuously chosen based on their projected size. The screenspace extent of each node is monitored: as the viewpoint shifts, certain nodes in the vertex tree will fall below the size threshold. These nodes will be *folded* into their parent nodes and the now-redundant triangles removed. Other nodes will increase in apparent size to the user and will be *unfolded* into their constituent child nodes, introducing new vertices and new triangles. The user selects the screenspace size threshold and may adjust it during the course of a viewing session for interactive control over the degree of simplification. HDS maintains an *active list* of visible polygons for rendering. Since frame-to-frame movements typically involve small changes in viewpoint, and therefore modify the active list by only a few polygons, the method takes advantage of temporal coherence for greater speed.

Any vertex merging simplification technique can be used to construct the vertex tree; the paper describes how two such techniques were implemented. The first is the *tight octree*, an adaptive hierarchical variation of spatial binning. The second is a hybrid approach based on edge collapses, in which adjacent vertices are merged until local curvature criteria would be violated by further edge collapses. The remaining vertices are then clustered together using a tight octree.

In addition, any set of run-time criteria can be plugged into the HDS framework in the form of a function that folds and unfolds the appropriate nodes. The paper describes the implementation of three such criteria: a screenspace error threshold, a silhouette test, and a triangle budget. The screenspace error threshold, described above, provides a form of fidelity-based simplification. The silhouette test uses a pre-calculated "cone of normals" to determine whether a vertex cluster is currently on the silhouette. Clusters on the silhouette are tested against a tighter screenspace threshold than clusters in the interior. Finally, HDS implements triangle-budget simplification by maintaining a priority queue of vertex clusters. The cluster with the largest screenspace error is unfolded and its children placed in the queue. This process is repeated until unfolding a cluster would violate the triangle budget.

4.6 Voxel-Based Object Simplification

Taosong He, Lechan Hong, Arie Kaufman, Amitabh Varshney, and Sidney Wang introduced this sampling algorithm to address the problem of topology. As mentioned above, the constraints of topology-preserving algorithms often limit their ability to perform drastic simplification. Topology-insensitive approaches such as the Rossignac-Borrel algorithm do not suffer these constraints, but reduce the topology of their models in a haphazard and unpredictable fashion. Voxel-based simplification is an attempt to simplify topology in a gradual and controlled manner using the robust and well-understood theory of signal processing.

The algorithm begins by creating a volumetric representation of the model, superimposing a three-dimensional grid of voxels over the polygonal geometry. The value of each voxel is determined from the density of polygons within that voxel. Next the algorithm applies a low-pass filter by resampling and convolving the volume. The result is another volumetric representation of the object with lower resolution. Sampling theory guarantees that small, high-frequency features will be eliminated in the low-pass filtered volume. The Marching Cubes algorithm is applied to this volume to generate a simplified polygonal model. Since Marching Cubes can create redundant geometry, a standard topology-preserving algorithm is required as a post-process.

Unfortunately, high-frequency details such as sharp edges and squared-off corners seem to contribute greatly to the perception of shape. As a result, the voxel-based simplification algorithm performs poorly on models with such features. This greatly restricts its usefulness on mechanical CAD models, which are perhaps the most likely models to have complex topologies.

4.7 Simplification Envelopes

Simplification envelopes, presented by Jonathan Cohen, Amitabh Varshney, Dinesh Manocha, Greg Turk, Hans Weber, Pankaj Agrawal, Frederick Brooks, and William Wright, provide a method of guaranteeing fidelity bounds while enforcing global as well as local topology [Cohen 96]. Simplification envelopes *per se* are more of a framework than an individual algorithm, and the authors of this paper present two examples of algorithms within this framework.

The simplification envelopes of a surface consist of two *offset surfaces*, or copies of the surface offset no more than some distance ε from the original surface. The *outer envelope* is created by displacing each vertex of the original mesh along its normal by ε. Similarly, the *inner envelope* is created by displacing each vertex by $-\varepsilon$. The envelopes are not allowed to self-intersect; where curvature of the original surface would create such self-intersection, ε is locally decreased.

Once created, these envelopes can guide the simplification process. The algorithms described in the paper are both decimation approaches that iteratively remove triangles or vertices and re-triangulate the resulting holes. By keeping the simplified surface within the envelopes, these algorithms can guarantee, first, that global topology is respected, and second, that the simplified surfaces deviates in no place by more than ε from the original surface. The resulting simplifications tend to have very good fidelity.

Where fidelity and topology preservation are crucial, simplification envelopes are an excellent choice. The ε error bound is also an attractive feature of this approach, providing a natural means for calculating LOD switching distances. Though the algorithms presented in the paper are based on a decimation approach, a vertex-merging algorithm based on simplification envelopes is easy to imagine. However, the very strengths of the simplification envelopes technique are also its weaknesses. The strict preservation of topology and the careful avoidance of self-intersections greatly curtail the approach's capability for drastic simplification. The construction of offset surfaces also demands an orientable manifold; topological imperfections in the initial mesh can hamper or prevent simplification. Finally, the algorithms for simplification envelopes are intricate and difficult to program. Writing a robust system based on simplification envelopes is a substantial undertaking.

4.8 Mesh Optimization

This paper by Hughes Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle, describes a complex sampling approach, which evolved out of the authors' work on surface reconstruction of laser-scanned datasets [Hoppe 93]. Surface reconstruction is the problem of creating a three-dimensional mesh from a collection of sample points. Mesh optimization, as the name suggests, treats simplification as an optimization problem. The number of vertices in the simplification and its deviation from the original are explicitly modeled as an energy function to be minimized.

The algorithm begins by sampling the mesh, taking a number of randomly placed samples in addition to the vertices of the original mesh. These sample points are used to measure deviation from the original. Next a random edge of the mesh is picked and one of three operations attempted at random: edge collapse, edge split, or edge swap. An inner loop then adjusts the positions of vertices to minimize the energy function for the next configuration. If the overall energy is not reduced or the topology is violated, the randomly selected edge operation is undone. Another random edge is picked and the process repeats, iterating until repeated attempts suggest that the energy function has reached a local minimum.

The careful simplification performed by the mesh optimization algorithm produces models of very high fidelity. The algorithm seems to be especially well suited for mechanical CAD models, capturing sharp features very nicely. Though topology is preserved, with the consequent limits on simplification, mesh optimization appears excellent at simplifying right up to those limits. Unfortunately, the algorithm is extremely slow, requiring hours to simplify a single machine part. In addition, mesh optimization probably qualifies as the most complex simplification scheme published, and a robust implementation seems quite difficult.

4.9 Progressive Meshes

This vertex-merging algorithm by Hughes Hoppe follows up on the mesh optimization approach. As described above, mesh optimization used the three techniques of edge collapse, edge split, and edge swap in random order to reduce an explicitly modeled energy function. The progressive meshes paper builds on the discovery that the edge collapse operation alone suffices to achieve high-quality simplification. The main contributions of the paper are the *progressive mesh*, a new representation for polygonal models based on edge collapses, and a topology-preserving simplification algorithm for generating progressive meshes.

A progressive mesh consists of a simple *base mesh*, created by a sequence of edge collapse operations, followed by a stream of *vertex split* records. A vertex split (or *vsplit*) is the dual of an edge collapse (or *ecol*). Each vsplit replaces a vertex by two edge-connected vertices, creating one additional vertex and two additional triangles. The vsplit records in a progressive mesh correspond to the edge collapse operations used to create the base mesh. Applying all of the vsplit records to the associated base mesh will recapture the original model exactly; applying a subset of the vsplit records will create an intermediate simplification. Since each vertex split creates two triangles (one for boundary edges), triangle-budget simplification is easily implemented by applying the vsplit records in order until the specified triangle budget is reached. In fact, the stream of vsplit records encodes a continuum of simplifications from the base mesh up to the original model. The vertex split and edge collapse operations are quite fast and may be applied at run-time to smoothly transition between levels of detail.

The quality of the intermediate simplifications depends entirely on the order of ecol operations used to create the base mesh. Hoppe describes a careful simplification algorithm to generate these edge collapses. The algorithm, like the mesh optimization algorithm, models fidelity explicitly as an energy function to be minimized. All edges that can be collapsed are evaluated according to their effect on this energy function and sorted into a priority queue. The ecol operation which most decreases the energy function is taken from the head of the queue and performed. Since this may change how collapsing nearby edges will affect the energy function, those edges are re-evaluated and resorted into the queue. This process repeats until topological constraints prevent further simplification. The remaining edges and triangles comprise the base mesh, and the sequence of ecol operations performed becomes (in reverse order) the stream of vsplit operations.

Along with progressive meshes, Hoppe introduces a very nice framework for handling surface attributes of a mesh during simplification. Such attributes are categorized as *discrete* attributes, associated with faces in the mesh, and *scalar* attributes, associated with corners of the faces in the mesh. Common discrete attributes include material and texture identifiers; common scalar attributes include color, normal, and texture coordinates. Hoppe also describes how to model some of these attributes in the energy function, allowing normals, color, and material identifiers to guide the simplification process.

Finally, Hoppe has recently extended progressive meshes to perform view-dependent simplification at run time [Hoppe 97], independently developing a system similar in many ways to the Luebke-Erikson algorithm described above.

4.10 Multiresolution Analysis of Arbitrary Meshes

This adaptive subdivision algorithm by Matthias Eck, Tony DeRose, Tom Duchamp, Hughes Hoppe, Michael Lounsbery, and Werner Stuetzle uses a compact wavelet representation to guide the recursive subdivision process [Eck 95]. By adding or subtracting wavelet coefficients the algorithm can smoothly interpolate between levels of detail. The algorithm provides fidelity-based simplification by using enough wavelet coefficients to guarantee that the simplified surface lies within a user-specified distance of the original model.

A chief contribution of this paper is a method for finding a simple base mesh that exhibits *subdivision connectivity*, which means that the original mesh may be recovered by recursive subdivision. As mentioned above, finding a base mesh is simple for terrain datasets but difficult for general polygonal models of arbitrary topology. Eck's algorithm creates the base mesh by growing voronoi-like regions across the triangles of the original surface. When these regions stop growing, a Delauney-like triangulation is formed from the voronoi sites and the base mesh formed in turn from the triangulation.

This algorithm possesses the usual disadvantages of strict topology-preserving approaches: manifold topology is absolutely required in the input model, and the shape and genus of the original object limit the potential for drastic simplification. The fidelity of the resulting simplifications is quite high for smooth organic forms, but the algorithm has difficulty capturing sharp features in the original model unless the features happen to fall along a division in the base mesh [Hoppe 96].

4.11 Surface Simplification Using Quadric Error Metrics

This recent view-independent vertex-merging algorithm by Michael Garland and Paul Heckbert strikes perhaps the best balance yet between speed, fidelity, and robustness. The algorithm proceeds by iteratively merging pairs of vertices, which may or may not be connected by an edge. Candidate vertex pairs are selected at the beginning of the algorithm according to a user-specified distance threshold *t*. Candidate pairs include all vertices which are connected by an edge, plus all vertex pairs separated by less than *t*. The major contribution of the algorithm is a new way to represent the error introduced by a sequence of vertex merge operations, called the *quadric error metric*. The quadric error metric of a vertex is a matrix that represents the sum of the squared distances from the vertex to the planes of neighboring triangles.

The error introduced by a vertex merge operation can be quickly derived from the sum of the quadric error metrics of the vertices being merged, and that sum will become the quadric error metric of the merged vertex. At the beginning of the algorithm, all candidate vertex pairs are sorted into a priority queue according to the error calculated for merging them. The vertex pair with the lowest merge error is removed from the top of the queue and merged. The errors of all vertex pairs involving the merged vertices are then updated and the algorithm repeats.

Quadric error metrics provide a fast, simple way to guide the simplification process with relatively minor storage costs. The result algorithm is extraordinarily fast; the authors report simplifying a 70,000 triangle model to 100 triangles in 15 seconds. The visual fidelity of the resulting simplifications is quite high, especially at high levels of simplification. Since disconnected vertices closer than *t* are allowed to merge, the algorithm does not require manifold topology, though it can be made to preserve topology by setting *t* to zero. One disadvantage of the algorithm is that the number of vertex pairs, and hence the running time of the algorithm, approaches $O(n^2)$ as *t* approaches the size of the model. In addition, the choice of a good value for *t* is very model-specific and can be difficult to automate. Finally, the algorithm as presented does not take vertex attributes such as color, normal, and texture information into account. Within these limitations, however, this simple-to-implement algorithm appears to be the best combination of efficiency, fidelity, and generality currently available. Many future algorithms will no doubt build on concepts introduced by Garland and Heckbert.

5. ISSUES AND TRENDS

5.1 Mechanism

The field of polygonal simplification appears to be converging on vertex merging as the underlying mechanism for polygon reduction. All four surface simplification papers in the SIGGRAPH '97 conference, for example, present algorithms based on merging vertices [Hoppe 97, Luebke 97, Garland 97, Popovic 97]. The simplicity and robust nature of vertex merging no doubt play a large part in this trend. Earlier work by Hoppe and Ronfard has probably played a part as well by demonstrating that high-quality simplification is possible with an algorithm based entirely on edge collapses [Hoppe 96, Ronfard 96]. Representations such as progressive meshes and the HDS vertex tree provide a very general framework for experimenting with different simplification strategies, including the relatively new view-dependent criteria. Settling on this emerging standard will hopefully allow the field of polygonal simplification to make faster strides in other important issues.

5.2 Error metrics

The lack of an agreed-upon definition of fidelity seriously hampers comparison of results among algorithms. Most simplification schemes use some sort of distance-based metric in which fidelity of the simplified surface is assumed to vary with the distance of that surface from the original mesh. The edge-collapsing approach of Guèziec preserves the enclosed volume of the simplified surface to within a user-specified tolerance [Guèziec 97]. Such metrics are useful for certain CAD applications, such as finite element analysis, and for certain medical and scientific applications, such as co-registering surfaces or measuring volumes. Probably the most common use of polygonal simplification, however, is to speed up rendering for visualization of complex databases. For this purpose, the most important measure of fidelity is not geometric but perceptual: does the simplification *look* like the original?

Unfortunately, the human visual system remains imperfectly understood, and no well-defined perceptual metric exists to guide simplification. Existing distance- and volume-based metrics, while certainly a useful approximation, suffer one definite deficiency by not taking the surface normal into account. Since lighting calculations are usually interpolated across polygons, for example, deviation in a vertex's normal can be even more visually distracting than deviation in its position. As another example, consider a pleated polygonal sheet. A single polygon spanning the width of the sheet may have minimal distance error but can have very different reflectance properties. In their survey of multiresolution methods for fast rendering, Garland and Heckbert propose that fidelity of simplification methods should be measured with perceptual tests using human viewers, or with a good *image-based* error metric. As a starting point for such a metric, they suggest the sum of the squared distances in RGB color space between corresponding pixels. [Garland 94].

5.3 View-dependence

View-dependent algorithms are quite new to the field of general polygonal simplification, and possess some definite advantages over view-independent approaches. View-independent methods are less general, making some implicit assumptions regarding object size. To begin with, physically large objects must be subdivided. Consider a model of a ship, for example: the hull of the ship should be divided into several sections, or the end furthest from the user will be tessellated as finely as the nearby hull. View-dependent techniques do not have this problem, since a single object can be rendered at multiple levels of detail. In addition, physically small objects may need to be combined, especially for drastic simplification. The diesel engine of that ship might consist of ten thousand small parts; from far away a roughly engine-shaped block makes a better approximation than ten thousand tetrahedra. Again, view-dependent techniques can be designed to automatically merge objects without requiring the user to explicitly establish a hierarchy of objects to be merged [Luebke 97]. Finally, view-dependent methods offer the possibility of more sophisticated simplification criteria. Some examples include preservation of silhouettes [Hoppe 97, Luebke 97, Xia 96], preservation of specular highlights [Xia 96], and aggressive simplification of backfacing regions [Hoppe 97]. View-independent algorithms can address none of these criteria.

However, view-dependence also suffers some significant drawbacks. View-dependent methods inherently involve more run-time computation than view-independent approaches. When the CPU rather than the graphics subsystem is the limiting factor in rendering performance, view-dependent approaches become less attractive. Also, view-dependent simplification is by nature an immediate-mode technique, a disadvantage since most current rendering hardware favors retained-mode display lists. Experiments on an SGI Onyx with InfiniteReality graphics, for example, indicate that Gouraud-shaded depth-buffered unlit triangles render two to three times faster in a display list than in a tightly optimized immediate mode display loop [Aliaga 97]. For these reasons view-dependent techniques seem unlikely to completely supplant view-independent techniques in the near future.

5.4 Geometry Compression

A field closely related to polygonal simplification is *geometry compression*. Rather than attempting to produce simpler representations of a polygonal model, geometry compression focuses on minimizing the storage requirements of a given mesh. Deering introduced the first geometry compression algorithm for general 3-D polygonal models. His approach applies quantization and lossy compression to attributes such as the position, normal, and color of vertices, achieving compression rates of 6-10:1 [Deering 95]. Taubin and Rossignac extend this idea by compressing the topological connectivity of polygons in the mesh. Decomposing the triangulated model into a tree of linear triangle strips allows significant compression of connectivity information, averaging roughly two bits per triangle [Taubin 96]. The Taubin-Rossignac algorithm has since been incorporated into a proposal for the next-generation binary format of VRML, the Virtual Reality Modeling Language, and the authors of the proposal report compression ratios of 50:1 or more for large VRML models [Taubin 97].

5.5 Progressive Transmission

As the bandwidth and processing power available to home users increase, 3-D graphics seem likely to undergo a mass-market debut similar to that which has recently shaken the hypertext-oriented World Wide Web. Indeed, VRML and browser plug-ins may well bring such a revolution about via the Web. The evolution of the WWW has underscored the importance of *progressive transmission* algorithms. These algorithms transmit a coarse version of the data first, followed by a stream of refinements, which the receiving process uses to reconstruct the original. The progressive mesh representation is by design well suited for progressive transmission of polygonal models [Hoppe 96]. If the mainstream debut of 3-D graphics occurs on the scale of the WWW, polygonal simplification algorithms may well be measured by their ability to support compression and progressive transmission.

6. CONCLUSION

The field of polygonal simplification, first heralded by Clark's seminal 1976 paper, virtually exploded out of nowhere in 1992. A flurry of papers have presented algorithms of every description, varying widely in mechanism, metric, and general approach. Now the field appears to be coming of age. Algorithms are converging on a common underlying mechanism, method-independent image-based fidelity metrics are being proposed, and techniques such as view-dependence are increasing the scope and generality of simplification methods. Recent algorithms possess excellent fidelity, robustness, and speed; the Garland-Heckbert algorithm, though not without limitations, shows promise of all three. As the field of polygonal simplification moves forward, researchers need to address the issues of a unified framework for simplification, better perceptual fidelity metrics, the role of view-dependence, and the relationship of polygonal simplification to compression and progressive transmission.

7. REFERENCES

[Aliaga 97] Aliaga, Daniel. "SGI Performance Tips" (Talk). For more information see: *http://www.cs.unc.edu/~aliaga/IR-perf.html*.

[Clark 76] Clark, James. "Hierarchical Geometric Models for Visible Surface Algorithms," *Communications of the ACM*, Vol. 19, No 10, pp 547-554.

[Cohen 96] Cohen, Jon, A. Varshney, D. Manocha, G. Turk, H. Weber, P. Agarwal, F. Brooks, W. Wright. "Simplification Envelopes", *Computer Graphics*, Vol. 30 (SIGGRAPH 96).

[Cosman 81] Cosman, M., and R. Schumacker. "System Strategies to Optimize CIG Image Content". *Proceedings Image II Conference* (Scotsdale, Arizona), 1981.

[Deering 95] Deering, Michael. "Geometry Compression", *Computer Graphics,* Vol. 29 (SIGGRAPH 95).

[Eck 95] Eck, Matthias, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, W. Stuetzle. "Multiresolution Analysis of Arbitrary Meshes", *Computer Graphics*, Vol. 29 (SIGGRAPH 95).

[Garland 94] Garland, Michael, and P. Heckbert. "Multiresolution Modeling for Fast Rendering". *Proceedings of Graphics Interface '94* (1994).

[He 95] Taosong He, L. Hong, A. Kaufman, A. Varshney, and S. Wang. "Voxel-Based Object Simplification". *Proceedings Visualization 95*, IEEE Computer Society Press (Atlanta, GA), 1995, pp. 296-303.

[Hoppe 93] Hoppe, Hughes. "Mesh Optimization", *Computer Graphics*, Vol. 27 (SIGGRAPH 93).

[Hoppe 96] Hoppe, Hughes. "Progressive Meshes", *Computer Graphics*, Vol. 30 (SIGGRAPH 96).

[Hoppe 97] Hoppe, Hughes. "View-Dependent Refinement of Progressive Meshes", *Computer Graphics*, Vol. 31 (SIGGRAPH 97).

[Lorenson 87] Lorenson, William, and H. Cline. "Marching Cubes: A High Resolution 3D Surface Construction Algorithm", *Computer Graphics*, Vol. 21 (SIGGRAPH 87).

[Low 97] Low, Kok-Lim, and T.S. Tan. "Model Simplification Using Vertex Clustering". In *1997 Symposium on Interactive 3D Graphics* (1995), ACM SIGGRAPH, pp. 75-82.

[Luebke 97] Luebke, David, and C. Erikson. "View-Dependent Simplification of Arbitrary Polygonal Environments", *Computer Graphics*, Vol. 31 (SIGGRAPH 97).

[Popovic 97] Popovic, Jovan, and H. Hoppe. "Progressive Simplicial Complexes", *Computer Graphics*, Vol. 31 (SIGGRAPH 97).

[Ronfard 96] Ronfard, Rèmi, and J. Rossignac. "Full-range Approximation of Triangulated Polyhedra", *Computer Graphics Forum,* Vol. 15 (Eurographics 96).

[Rossignac 92] Rossignac, Jarek, and P. Borrel. "Multi-Resolution 3D Approximations for Rendering Complex Scenes", pp. 455-465 in *Geometric Modeling in Computer* Graphics, Springer-Verlag, Eds. B. Falcidieno and T.L. Kunii, Genova, Italy, 6/28/93-7/2/93. Also published as IBM Research Report RC17697 (77951) 2/19/92.

[Schroeder 92] Schroeder, William, J. Zarge and W. Lorenson, "Decimation of Triangle Meshes", *Computer Graphics*, Vol. 26 (SIGGRAPH 92)

[Taubin 96] Taubin, Gabriel, and J. Rossignac. "Geometric Compression through Topological Surgery", IBM Research Technical Report RC-20340 (1996).

[Taubin 97] Taubin, Gabriel (Chair). "VRML Compressed Binary Format Working Group Home Page". For more information see: *http://www.vrml.org/WorkingGroups/vrml-cbf/cbfwg.html*.

[Teller 91] Teller, Seth, and C. Sequin. "Visibility Preprocessing for Interactive Walkthroughs", *Computer Graphics*, Vol. 25 (SIGGRAPH 91).

[Turk 92] Turk, Greg. "Re-tiling Polygonal Surfaces", *Computer Graphics,* Vol. 26 (SIGGRAPH 92).

[Xia 96] Xia, Julie, and A. Varshney. "Dynamic View-Dependent Simplification for Polygonal Models", *Visualization 96*.