International Journal of Computational Geometry & Applications

COMPUTATIONAL GEOMETRY COLUMN 33

JOSEPH O'ROURKE
Department of Computer Science
Smith College, Northampton, MA 01063, USA
`orourke@cs.smith.edu`

Received 6 March 1998
Communicated by D. T. Lee

ABSTRACT
Several recent SIGGRAPH papers on surface simplification are described.

Keywords: Mesh simplification, decimation.

The stringent demands of real-time graphics have engendered a need for simplification of object models. Here several papers on aspects of the problem for 3D polygonal models are described at a high level.

1. Levels of Detail

A consensus may be emerging in favor of the representation of complex models as a single, hierarchical data structure that represents many levels of detail simultaneously, from the simplified root to the fully detailed leaves. The hierarchy is known under various names: vertex tree,[7] merge tree,[9] vertex hierarchy,[6] progressive mesh,[5] progressive simplicial complex.[8] We will use *vertex tree* here to refer to the generic concept. Typically the tree is binary, with each node represting a vertex of some simplification of the original model. The two children vertices v_1 and v_2 of their parent v are merged (or identified, or unified, or contracted), moving up the tree to produce v, which inherits all the triangles incident to its children. Viewing the same process in reverse, the parent v splits to generate its children at an increased level of detail. How the coordinates of v relate to those of its children depends on the particular hierarchy implementation: e.g., whichever of $\{v_1, v_2, \frac{1}{2}(v_1 + v_2)\}$ is "best,"[8] or a position that minimizes some geometric error.[4] Some schemes[5,9,6] restrict v_1 and v_2 to be connected by an edge at their model level, in which case the upward tree movement is an *edge contraction*. This has the advantage of preserving the abstract topology[a] of the model, an advantage that becomes an impediment to massive simplification of complex models. Thus much

[a] The reason for the qualification is that guaranteeing preservation of geometric simplicity requires careful placement of v, care not usually exercised for pragmatic reasons. An exception is Ref. [1].

recent work[4,7,8] countenances arbitrary vertex pair identification, which may, for example, merge separate topological components. Before addressing which pairs of vertices should be unified, we turn to how a model vertex tree can be utilized by a graphics system.

2. View-dependent Simplification

The vertex tree is constructed from the fully detailed original model in a preprocessing phase that can take anywhere from a few seconds to (in one cited instance[8]) over 22 hours. In any one frame, the model is rendered from a list of *active vertices*[6] which represent a variable-detail frontier within the vertex tree.[7] Each vertex node points to relevant incident faces (triangles) with enough information for rendering. Before a new frame is rendered, the list of active vertices is traversed, and a decision made whether to split a node to increase detail, merge two nodes to simplify, or leave as is. This decision is based on a *screen-space* error criterion. The idea is that what matters is what the user sees—*object-space* geometric errors are less relevant. Projected surface deviation[6] and silhouette preservation[7] among other heuristics have been used. It is of course crucial that these computations be fast, and indeed impressive real-time behavior on graphics workstations has been achieved.

3. Construction of Hierarchy

In contrast to reliance on screen-space error to decide which nodes to display, geometric and topological considerations dominate the initial tree construction. One method[3] computes inner and outer *simplification envelopes* for an object, both within ϵ of the original surface, and threads a simplification between. An algorithm for this difficult subproblem approximates a global optimum by greedily accumulating threading triangles that "cover" (in projection) many vertices. Successively larger values of ϵ lead to vertex clustering that could be represented in a vertex tree.[b]

Other methods more directly follow the vertex tree structure, choosing to unify the pair of vertices that minimize error.[4,8] Because it is prohibitive to consider all pairs of vertices, a crude winnowing is performed, based on topology (vertices connected by an edge) and geometric proximity (Euclidean distance between vertices[4]; "tight" octree clustering[7]; Delaunay edges between components[8]). Some schemes assume manifold objects[3,5,9]; others permit arbitrary simplicial complexes.[4,7,8] In the surviving list of candidate pairs, a merging "cost" is computed for each pair, based on various geometric-based heuristics: estimation of the deviation of v from neighboring face planes by an "error quadric" in Ref. [4] and a mixture of distance, area-stretching, and folding penalties in Ref. [8]. For each merge, between-level dependencies must be carefully arranged to permit subsequent swift navigation through the hierarchy.

There are considerable tradeoffs between speed of construction and the "quality" of the resulting hierarchy, and much of the research focus has been on the details

[b] The work in Ref. [3] is not, however, geared toward constructing a hierarchy.

glossed here, which determine these tradeoffs. We now sketch the error quadrics of Ref. [4] to give a sense of some of these details.

4. Error Quadrics

Define the distance $d(v, P)$ of a point $v = [x\, y\, z\, 1]^T$ to a set P of planes to be the sum of the squares of the distances from v to the planes in P. For a single plane $p = [a\, b\, c\, d]^T \in P$, the square distance is $(v^T p)(p^T v) = v^T(pp^T)v$. Summing this expression over all $p \in P$ yields $d(v, P) = v^T Q v$, where Q is a 4×4 matrix.

Note that the distance between v and all the planes containing faces incident to v is zero, because v lies on each of these planes. Now consider v to be the parent of two vertices v_1 and v_2, whose unification somehow yielded v. We define the cost of the merge, or the error associated with v, to be $d(v, P_1) + d(v, P_2) = v^T(Q_1 + Q_2)v$, where P_i are the planes containing faces incident to v_i, and Q_i the corresponding matrices, $i = 1, 2$. Note that if a face is incident to both v_1 and v_2, its plane will be in both P_1 and P_2, and will be doubly weighted by the matrix sum $Q_1 + Q_2$.

This leaves how to chose v. For Q fixed, $v^T Q v = v^T(Q_1 + Q_2)v$ is a quadric surface. Its level surface $v^T Q v = \epsilon$ is a (potentially degenerate) ellipsoid specifying a region of space within which any v has an error of at most ϵ. The v used in Ref. [4] is the center of this ellipsoid.

5. Evaluation

Currently the efficacy of a proposed simplification algorithm is evaluated by a mixture of run-time data, intuition, and visual appeal. In some sense the ultimate arbiter is the (inevitable) SIGGRAPH video. It would seem useful to place evaluation on a more firm theoretical footing. With much of the code in the public domain, experimental comparisons are now appearing.[2]

Acknowledgements

I thank Michael Garland, Paul Heckbert, Hugues Hoppe, David Luebke, and Dinesh Manocha for comments.

(Written 6 March 1998)

References

1. J. Cohen, D. Manocha, and M. Olano. Simplifying polygonal models using successive mappings. In *Proc. IEEE Visualization '97*, pages 395–402, 1997.
2. P. Cignoni, C. Montani, and R. Scopigno. A comparison of mesh simplification algorithms. In *Computers & Graphics*, volume 22. Pergamon Press. To appear; `http://miles.cnuce.cnr.it/cg/bibliography.html`.
3. J. Cohen, A. Varshney, D. Manocha, G. Turk, H. Weber, P. Agarwal, F. Brooks, and W. Wright. Simplification envelopes. In *Proc. SIGGRAPH '96*, pages 119–128, 1996.
4. M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *Proc. SIGGRAPH '97*, pages 209–216, 1997.

5. H. Hoppe. Progressive meshes. In *Proc. SIGGRAPH '96*, pages 99–108, 1996.
6. H. Hoppe. View-dependent refinement of progressive meshes. In *Proc. SIGGRAPH '97*, pages 189–198, 1997.
7. D. Luebke and C. Erikson. View-dependent simplification of arbitrary polygonal environments. In *Proc. SIGGRAPH '97*, pages 199–208, 1997.
8. J. Popović and H. Hoppe. Progressive simplicial complexes. In *Proc. SIGGRAPH '97*, pages 217–224, 1997.
9. J. Xia and A. Varshney. Dynamic view-dependent simplification for polygonal models. In *Proc. IEEE Visualization '96*, pages 327–334, 1996.