Normal-based simplification algorithm for meshes

Frutuoso G. M. Silva and Abel J. P. Gomes
IT - Networks and Multimedia Group
Department of Computer Science and Engineering
University of Beira Interior - Portugal
{fsilva, agomes}@di.ubi.pt

Abstract

This paper presents a new edge collapsing-based simplification algorithm for meshes. It is based on the variation of the vectors that are normal to faces around the collapsing edge. Its main novelty is that it uses the same criterion to choose and validate the collapsing edge. Besides, at the best knowledge of the authors, it is the fastest simplification algorithm found in the literature using the edge collapse operation. This simplification algorithm, in conjunction with its inverse algorithm (i.e. refinement algorithm), allows the automatic creation of a multiresolution schema, i.e. a sequence of meshes at different resolutions. Additionally, it makes a good trade-off between time performance and mesh quality.

1. Introduction

In computer graphics applications the complexity and detail of the geometric models increase everyday to cope with more and more sophisticated modeling systems and data acquisition technologies. Usually, these models are tessellated into polygonal approximations, called meshes. But their complexity makes it difficult their storage and visualization. The efficient manipulation of large meshes in interactive graphical applications began with the use of Level-Of-Detail (LOD) models, as it is the case of VRML [2] (Virtual Reality Modelling Language) file-format.

The concept of LOD allows us to have different representations of a geometric object, having different levels of accuracy and complexity. A LOD model consists of a fixed collection of independent meshes of different sizes, each representing the object at a given resolution. However, a LOD model needs much memory to store all representations for a single object because each subsidiary mesh is stored independently. Thus, the number of subsidiary meshes must be small in order to keep the memory requirements within reasonable bounds. Unfortunately, this causes abrupt changes between consecutive levels of detail.

Therefore, it seems inviable to think of storing LOD meshes for large objects because that requires huge amounts of memory for each model. This problem motivated the development of:

- **Simplification algorithms**. Large meshes can be replaced by approximations with far fewer cells because the result in visually undistinguishable for the human eye. This is particularly important because current modeling and scanning systems create huge meshes unnecessarily beyond the human vision accuracy. They were not designed for generating meshes with high visual quality and a minimal number of cells.
- **Multiresolution meshes**. A multiresolution mesh is a model that can provide a high number of subsidiary meshes representing a single object at different resolutions. It allows us to render a mesh at different levels of resolution, depending on its projective distance to a virtual viewer. However, unlike LOD meshes, only a single mesh is loaded in memory. Any new mesh is generated from the current one by applying simplification and refinement operations on it.

This paper is organized as follows. Related work appears in Section 2. Section 3 presents a new simplification algorithm and a multiresolution schema for meshes. Section 4 shows some results and comparisons. Lastly, Section 5 draws some conclusions and future work.

2. Related work

Much research has been done in the last few years on polygonal meshes, particularly in multiresolution and mesh simplification. Simplifying a polygonal mesh M_i consists of generating another mesh M_j with a less number of cells. The resulting mesh obeys a target criterion, which is normally a measure of a maximum admissible error. This er-

Proceedings of the Theory and Practice of Computer Graphics 2004 (TPCG'04)
0-7695-2137-1/04 $20.00 © 2004 IEEE

ror can be implicitly given by specifying a target number of cells.

2.1. Classes of simplification algorithms

There are several types of algorithms for simplifying a polygonal mesh. They can be broadly categorized into three classes, namely:

- **Cell decimation**. Basically, a cell decimation algorithm selects a cell (vertex, edge, or face) for removal, removes all star-adjacent cells, and re-meshes the resulting hole. These algorithms reduce the number of cells in a mesh, though preserving the original topology of the model. The main advantage is that the remaining vertices belong to the original mesh, i.e. it is a subset of the original data. This way, we can still reuse the vertex attributes. The algorithms due to [27, 16, 5, 17] are representatives of this class.
- **Vertex clustering**. The vertex clustering method simplifies a mesh by first enclosing it within a bounding box, which is then divided into a grid of boxed cells. The vertices of the mesh inside each boxed cell are clustered together into a single vertex, being the mesh faces updated accordingly. The clustering vertex may be the center of mass of the cluster or any other point in the boxed cell weighted by a specific criterion. Other algorithms use BSP (Binary Space Partitioning) tree. A BSP tree represents a recursive, hierarchical partitioning, or subdivision, of n-dimensional space into convex subspaces. Usually, vertex clustering algorithms produce rough approximations because they do not preserve the topology and geometry of the model, being the error bounded by the cell size. Some algorithms in this class are, for example, those due to [26, 1, 18].
- **Edge collapsing**. This method simplifies a mesh by iteratively collapsing edges into vertices. These algorithms use techniques of varying sophistication to determine which edges to collapse in which order. They tend to preserve the topology but they may change it if the collapsing pair of vertices are not connected by an edge. The edge collapse operation has the disadvantage that it may cause local surface inversion. The algorithms due to [12, 10, 7, 20] belong to this class.

Recently have appeared other type of simplification algorithms, designed by out-of-core simplification algorithms. As meshes have grown rapidly in recent years, out-of-core simplification has become an increasingly important tool for dealing with large meshes. These type of algorithms is based on spatial clustering [18, 19, 8] and surface segmentation [11, 24]. More information about the out-of-core algorithms can be found in [28, 14].

2.2. Simplification criteria for edge collapsing

The edge collapsing operation is standard. The main difference between the various edge collapsing-based simplification algorithms is the criterion they use to choose the next edge to collapse. A different criterion implies a different mesh quality, as well as a distinct processing time.

Hoppe [10] introduced an algorithm for constructing progressive meshes through a simplification algorithm based on the minimization of an energy function. A progressive mesh is basically a multiresolution mesh that can be simplified and refined by applying edge collapse and vertex split operations, respectively.

Garland and Heckbert [7] simplification algorithm follows a geometric criterion that is based on the minimization of the error associated with each new vertex. This error is defined as the sum of the squared distances to the set of planes surrounding the pair of the original collapsing vertices. This algorithm also allows collapsing pairs of vertices without an edge between them. This method has the advantage that it produces high quality approximations quickly.

Lindstrom and Turk [20] simplification algorithm selects the position of the new vertex so that the volume enclosed by the mesh is maintained and the volume of each tetrahedron swept out by a moving mesh triangle is minimized. Besides, the surface area near possible mesh boundaries is maintained and the per-triangle area changes are minimized.

Guskov et al. [9] introduced the normal meshes. A normal mesh is a multiresolution mesh, being each subsidiary mesh written as a normal offset from a coarser mesh. Normal meshes use the Garland and Heckbert simplification algorithm to produce a mesh hierarchy.

Silva and Gomes [29] presented a simplification algorithm based on the coplanarity of the faces around the collapsing edge. An edge is collapsed into a vertex if the angle between any two of those faces is less than a fixed value. It is fast but sometimes produces meshes with lengthy triangles, i.e. meshes with relatively low quality.

More recently, Kho and Garland [15] presented a user-guided approach for mesh simplification. The user can impose geometric constraints on specific surface regions in order to preserve various shape features. This means that the simplification algorithm is only applied to unconstrained regions. They also use the Garland and Heckbert simplification algorithm.

Generally, all simplification algorithms make a trade-off between speed and the quality of resulting mesh. For more details about mesh simplification algorithms see [3, 20, 25, 6, 13, 22, 21].

This paper presents a new simplification algorithm that is faster than those found in the literature, without losing too much mesh quality.

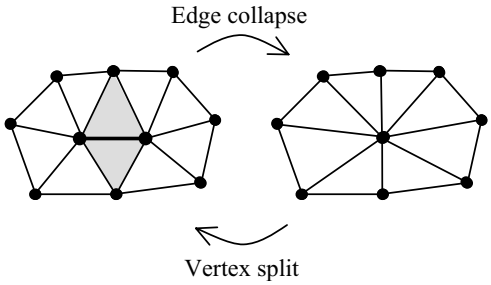

Figure 1. Edge collapse and vertex split operations.

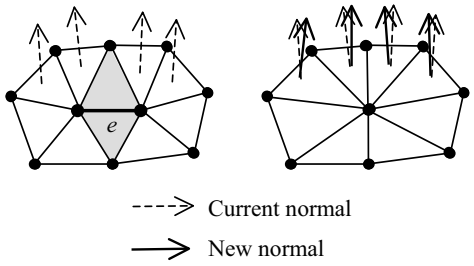

Figure 2. Variation of the face normal around the target edge.

3. Normal-based simplification algorithm

3.1. Edge collapse operation

Collapsing an edge consists of homotopy-contracting it and its bounding vertices into a single vertex, as illustrated in Fig. 1. In this paper, this vertex is the midpoint of the collapsing edge, but it is not mandatory to do so. The collapsing edge can be contracted into another point of it. Overall, the edge-collapsing operation removes a vertex, three edges, and two faces. Remarkably, this operation is invertible (see Fig. 1), so enabling the construction of a multiresolution schema. The inverse operation is named vertex split and is used for refining meshes.

Splitting a vertex into two vertices requires we provide the coordinates of the new vertex, and which is the third vertex of each new face. Normally, this information is stored during the edge collapse operation, allowing us to roll back to the original mesh as in Hoppe's progressive meshes.

3.2. Simplification criterion

The simplification algorithm proposed in this paper, called Normal-based Simplification Algorithm (NSA),

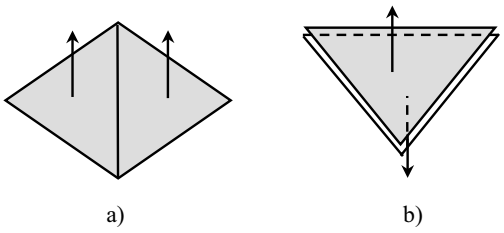

Figure 3. Coplanar faces.

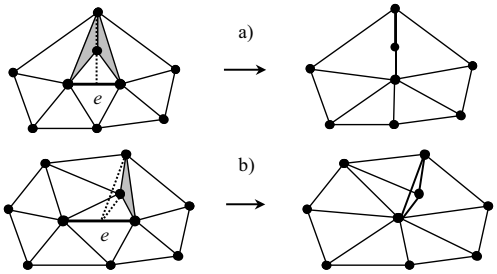

Figure 4. Invalid edge collapse operations.

is a kind of an edge collapsing-based simplification algorithm. It uses the edge collapsing operation to simplify a mesh.

An edge is only collapsed if the variation of the face normals around the target edge is within a given tolerance ε. The value of ε is the threshold for the angle between current normal and new normal after the edge collapse operation, as shown in Fig. 2. It is up to the user to choose the value of ε in a predefined interval. The bigger is ε, the more simplified is the mesh.

This geometric criterion implies that the region around the collapsing edge is nearly coplanar. In these circumstances, we say that a region is quasi-coplanar. But, the contrary is not true, i.e. the coplanarity does not ensure a minimal variation of the face normals. For example, Fig. 3 shows two examples of coplanar faces, the first having two faces with the same orientation, while the second has faces with opposite orientation.

The NSA algorithm guarantees the quasi-coplanarity of the region to simplify and, at the same time, prevents possible local folding (Fig. 3(b)) or partial face overlapping of the resulting mesh, because the edge collapse operation is only allowed if the variation of the normals is less than ε. For example, in Fig. 4 the collapsing of edge e produces a significant variation of the normals, i.e. the variation of the normals is greater than ε; consequently, the simplification is not allowed.

Besides, it produces very good approximations preserv-

ing the original visual shape and the boundaries of the original model (see Fig. 5). The mesh boundaries are preserved because the collapse operation is cancelled whenever at least one edge neighboring the collapsing edge has only a single face incident on it. This also means that the boundary edges are never collapsed.

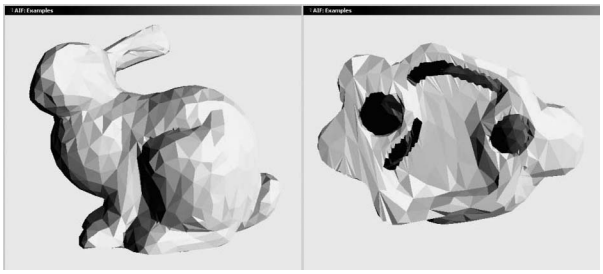

Figure 5. Left-side view and bottom-side view of a simplified bunny mesh with 2785 faces.

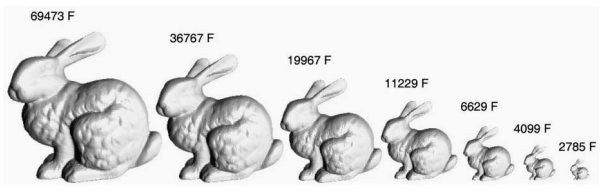

Figure 6. Sequence of simplified meshes for the bunny.

3.3. Edge collapsing validation

Most authors distinguish between the simplification criterion and the criterion that validates the edge collapse. On the contrary, we use the same criterion to simplify and validate edge collapse. This means that the normal-based criterion used to simplify a mesh is also used to validate the edge collapse operation.

In fact, the edge collapse may cause local surface inversion (is not always a valid operation), unless we have some way to control overlapping faces (Fig. 4(a)) or quasi-overlapping faces (Fig. 4(b)), leaving the mesh with infinitesimal cavities. For example, the dark faces in Fig. 4(a) overlap after collapsing the edge e. Similarly, the dark face in Fig. 4(b) flips after collapsing the edge e, leaving there an infinitesimal cavity.

To solve these problems many authors use some sort of heuristics (see, for example, [12]). Basically, they use a threshold value for the maximum dihedral angle of edges in the star of each vertex, instead of checking the mesh for self-intersections after collapsing an edge because this operation is time-consuming. However this solution prevents not only the invalid cases (Fig. 4) but also the appearance of lengthy triangles.

Other authors also use face normal-based criterion before and after the collapse operation [7], but they use a different simplification criterion, as refereed in Section 2.2. If the face normal flips the collapse operation is disallowed (see Fig. 4). Note that our simplification algorithm simplifies mesh only in regions approximately coplanar, after which it produces a little variation on the faces normal.

3.4. Multiresolution schema

The NSA simplification algorithm is ”memoryless” in the sense that it only uses the current approximation to compute the next one. No information about the original shape or previous approximations is retained.

For a given error ε, the NSA algorithm produces a finite number of approximations for a model. Fig. 6 shows a sequence of bunny meshes produced by NSA algorithm with a pre-defined error $\varepsilon = 0.025$. We have applied a scale factor to this bunny mesh that reduces the size of it from an approximation to another, simulating the effect of increasing the distance to a virtual viewer. Fig. 5 shows the last simplification of bunny mesh with 2785 faces pictured in Fig. 6.

Thus, each iteration of the NSA algorithm produces a more simplified mesh. To get a more uniform simplification over the mesh and to produce a sequence of simplified meshes, we have imposed the following restriction: for each collapsing edge, all incident edges at the resulting vertex are prevented to be collapsed in the current approximation. This disallows simplification process in regions that have been already simplified and speeds up the simplification algorithm for each approximation. Another consequence is that the area of each mesh triangle does not change significantly, increasing the mesh quality (see, for example, Fig. 5).

The refinement of a mesh is carried out by applying the vertex-split operation (i.e. the inverse operation of edge collapse shown in Fig. 1) to the vertices created during the last mesh simplification. This requires that these vertices store at least the position of one of the original collapsing vertices. To distinguish vertices created at different mesh simplifying approximations, we use a vector of dynamic vectors, each one storing the vertices created at each simplify-

ing approximation. Thus, it is always possible to roll back to the original mesh by using the refinement algorithm.

3.5. The data structure

Data structures for simplification algorithms are normally compact and therefore more suitable for geometric data storage and transmission than the traditional b-reps (boundary representations) data structures. This is in part due to the fact that they store and process cells, i.e. manifolds without holes.

The data structure supporting the normal-based simplification algorithm is the AIF data structure [30]. It can accommodate simplicial complexes, and more generally cell complexes but, unlike other b-rep data structures, no oriented cells (say, half-edges, co-edges, directed edges or so) are associated to the cells themselves (vertices, edges, and faces). Note that AIF data structure is not topologically-oriented provided that it does not include any oriented cells. AIF is an orientable, but not an oriented, data structure. This means that a geometric orientation can be topologically induced on the mesh. Thus, we have an orientation mechanism to induce a geometric orientation on the mesh. Inducing an orientation on a mesh means that the same or consistent orientation is induced on all faces, being then calculated a normal for each of them.

Besides, the AIF data structure uses a single adjacency and incidence operator, called mask operator, to retrieve all adjacency and incidence information. According to Ni and Bloor [23], the AIF data structure accommodates an optimal representation in the class C_4^9 for 2-dimensional meshes. Basically, we say that vertices are adjacent to edges, and edges are adjacent to faces; equivalently, faces are incident on edges, and edges are incident at vertices. In fact, the AIF data structure implements an incidence scheme that directly represents four out of nine adjacency and incidence relations between the cells. The remaining five topological relations are easily determined by querying the data structure through the mask operator. Note that most of the data structures belong to the class C_2^9, i.e. they have only explicitly represented two relations in data structure, being the remaining seven implicitly represented. Consequently, they are less responsive than AIF data structure. Besides, the query operator (mask operator) handles a mesh locally by using the adjacency and incidence relations stored in the data structure. Thus, its time performance holds independently of the mesh size. This is very important for handling large meshes, in particular in real-time operations. More information about mask operator and the comparisons between AIF and other data structures can be found in [30].

4. Results and comparisons

4.1. NSA results

As shown in Fig. 7, the performance of the NSA algorithm depends not only on the model size, but also the number of subsidiary meshes (or, equivalently, approximations steps) created by the algorithm. As expected, bigger meshes have longer simplification times (Fig. 7 -Simplification times). In practice, we noted that there is a finite number of simplification steps for each model for a given error ε. That is, for an specific error ε, every model tends to a minimum number of faces, after which no more simplifications can be done. This subsidiary mesh with a minimum number of cells is called minimum mesh. Looking at Fig. 7 -Number of approximations, and assuming the same error ε for all the models, we see that the NSA produces five subsidiary meshes for the Dinosaur model, seven for the Rabbit model and six for the Bunny, Horse, and Venus models. Thus, the NSA algorithm potentially produces more subsidiary meshes for large meshes than for small meshes.

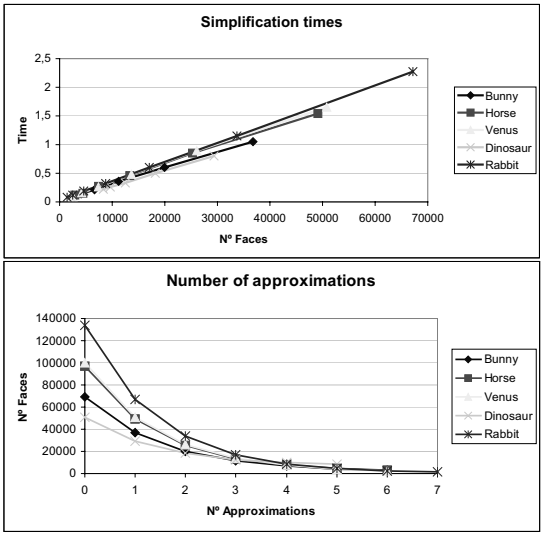

Figure 7. Performance of the NSA algorithm.

Fig. 8 shows the simplification and refinement times concerning the Bunny mesh for two different error values, $\varepsilon = 0.1$ and $\varepsilon = 0.025$. Looking at Fig. 8, we note that different error values originate subsidiary meshes with different number of faces. However, the simplification and refinement times do not change that much even when the error changes significantly. In last page, Fig. 12, 13, 14 and 15 show more results for the models mentioned in this paper,

specifically the original mesh and the corresponding minimum mesh for each model, with $\varepsilon = 0.025$.

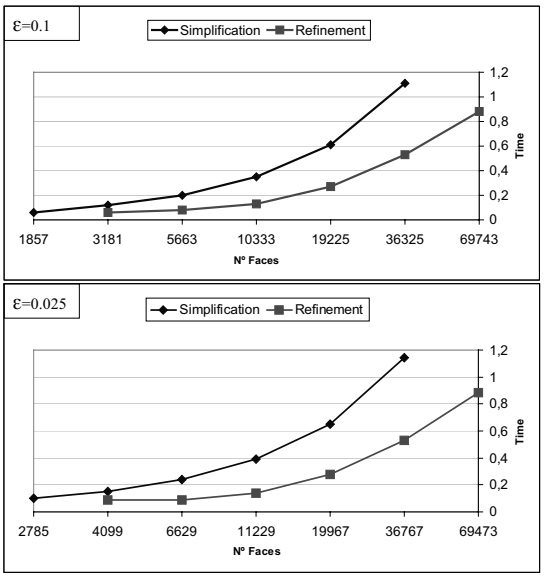

Figure 8. Simplification and refinement times for the bunny mesh with $\varepsilon = 0.1$ and $\varepsilon = 0.025$.

4.2. Time performance comparison

The time performance comparison tests were done on a PC equipped with 1.6GHz Intel Pentium 4, 768MB of memory, running the Windows 2000 OS. Note that all times come in seconds.

A major advantage of the simplification algorithm presented in Section 3 is that its processing time is shorter than others found in the literature [20, 3]. This is mainly due to the normal-based geometric criterion itself, as well as its supporting AIF data structure. Fig. 9 compares the total processing time spent by the fastest simplification known so far (according the words of Lindstrom and Turk [20]), called Quadric Metrics (Qslim) algorithm [7], and the NSA algorithm in the creation of a multiresolution mesh for the Rabbit and Bunny meshes. The Quadric Metrics algorithm is freeware and available from http://www.cs.cmu.edu/~ garland, which has permitted to run this and our algorithm on the same machine for a real comparison.

In general, as shown in Fig. 10, the NSA is faster than the Qslim algorithm for different meshes. Fig. 10 shows the simplification times for the models depicted in Fig. 12, 13, 14 and 15.

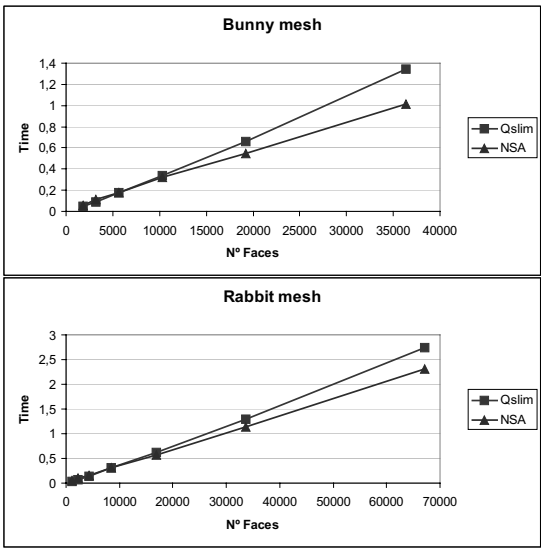

Figure 9. Simplification times comparison for Qslim and NSA algorithms.

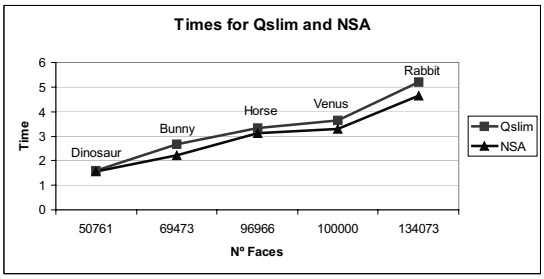

Figure 10. Simplification times comparison between NSA and Qslim algorithms.

4.3. Mesh quality comparison

The mesh quality can be evaluated by different methods. We have chosen the Metro geometric comparison tool [4]. This tool reads two meshes, the original mesh and a simplified mesh, and then computes the maximum geometric error, the mean geometric error, and the Hausdorff distance of the simplified mesh with respect to the original one.

Fig. 11 show the quality of two simplified meshes, the Bunny and Venus meshes, generated by the NSA and Qslim algorithms, against the Hausdorff distance. The Hausdorff distance, and consequently the mesh quality, is similar for meshes generated through both algorithms, with the exception of those near the minimum meshes. This is so

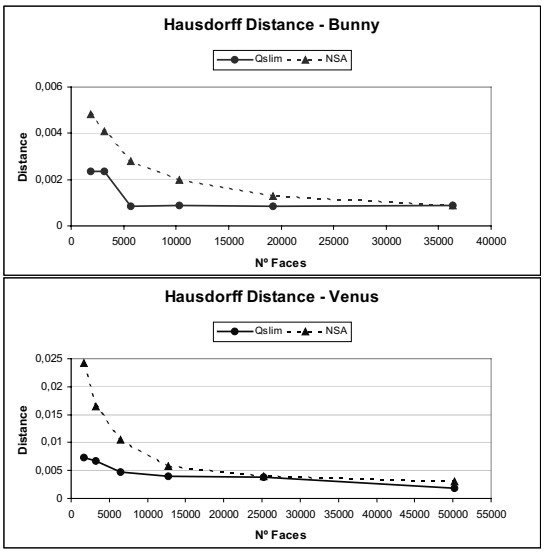

Figure 11. Mesh quality evaluation for Qslim and NSA.

because the NSA algorithm always selects the midpoint of the collapsing edge to create a new vertex, while the Qslim algorithm chooses the point of the collapsing edge that minimizes the error and still the fact that NSA algorithm is "memoryless". However, this is not actually a big problem because minimum meshes are supposed to be far away from the virtual observer.

5. Conclusions and future work

A new simplification algorithm for meshes has been presented in this paper. Its main contribution is the same criterion used to choose and validate the collapsing edge. Besides, at the best knowledge of the authors, it is the fastest simplification algorithm found in the literature in its class.

In general, it provides good results in terms of shape preservation, time performance, and mesh quality. The NSA algorithm was primarily developed for time efficiency purposes. Mesh quality was not a priority goal, but we ended up by getting it. To preserve the model shape, we imposed some restrictions on the collapsing edges, as explained before in the paper.

In the near future, we expect to develop a prototype of an interactive animation system that integrates our multiresolution schema in order to represent model actors at different levels-of-detail, depending on their projective distance to a virtual viewer.

Acknowledgments

The models are courtesy of: Cyberware, Stanford University and 3D Cafe.

References

[1] D. Brodsky and B. Watson. Model simplification through refinement. In *Proceedings of Graphics Interface 2000*, pages 221–228, 2000.

[2] R. Carey, G. Bell, and C. Marrin. The reality modeling language iso/iec 14772-1. Technical report, The VRML Consortium Incorporated, 1997.

[3] P. Cignoni, C. Montani, and R. Scopigno. A comparison of mesh simplification algorithms. *Computers and Graphics*, 22(1):37–54, 1997.

[4] P. Cignoni, C. Rocchini, and R. Scopigno. Metro: measuring error on simplified surfaces. Technical report, B4-01-01-96, I.E.I-C.N.R., Pisa, Italy, 1996.

[5] J. Cohen, A. Varshney, D. Manocha, G. Turk, H. Weber, P. Agarwal, F. Brooks, and W. Wright. Simplification envelopes. In *Proceedings of Siggraph 96*, pages 119–128. ACM, 1996.

[6] M. Garland. Multiresolution modeling: Survey & future opportunities. In *Eurographics '99 - State of the Art Reports*, pages 111–131. ACM, 1999.

[7] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *Proceeding of Siggraph 97*, pages 209–216, 1997.

[8] M. Garland and E. Shaffer. A multiphase approach to efficient surface simplification. In *Proceedings of IEEE Visualization*, 2002.

[9] I. Guskov, K. Vidimce, W. Sweldens, and P. Schroder. Normal meshes. In K. Akeley, editor, *Proceedings of Siggraph 2000*, pages 95–102. ACM, 2000.

[10] H. Hoppe. Progressive meshes. In *Proceeding of Siggraph 96*, pages 99–108, 1996.

[11] H. Hoppe. Smooth view-dependent level-of-detail control and its application to terrain rendering. In *Proceeding of IEEE Visualization*, pages 35–42, 1998.

[12] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. In *Proceedings of Siggraph 93*, pages 19–26, 1993.

[13] A. Hubeli and M. Gross. A survey of surface representations for geometric modeling. Technical report, #335, Swiss Federal Institute of Technology Zurich, 2000.

[14] M. Isenburg, P. Lindstrom, S. Gumhold, and J. Snoeyink. Large mesh simplification using processing sequences. In *IEEE Visualization*, pages 465–472, 2003.

[15] Y. Kho and M. Garland. User-guided simplification. In *Proceedings of ACM Symposium on Interactive 3D Graphics*. ACM Press, 2003.

[16] R. Klein, G. Liebich, and W. Straßer. Mesh reduction with error control. In R. Yagel and G. M. Nielson., editors, *Proccedings of IEEE Visualization 96*, pages 311–318. IEEE Computer Society, 1996.

[17] L. Kobbelt, S. Campagna, and H.-P. Seidel. A general framework for mesh decimation. In *Graphics Interface*, pages 43–50, 1998.

[18] P. Lindstrom. Out-of-core simplification of large polygonal models. In *Proceedings of Siggraph 2000*, pages 259–262. ACM, 2000.

[19] P. Lindstrom and C. Silva. A memory insensitive technique for large model simplification. In *Proceedings of IEEE Visualization 2001*, pages 121–126. IEEE Computer Society, 2001.
[20] P. Lindstrom and G. Turk. Fast and memory efficient polygonal simplification. In *Proceedings of IEEE Visualization 98*, pages 297–286. IEEE Computer Society, 1998.
[21] D. Luebke, M. Reddy, J. D. Cohen, A. Varshney, B. Watson, and R. Huebner. *Level Of Detail For 3D Graphics*. Morgan Kaufmann, 2002.
[22] D. P. Luebke. A developer's survey of polygonal simplification algorithms. *IEEE Computer Graphics and Applications*, 21(3):24–35, 2001.
[23] X. Ni and M. S. Bloor. Performance evaluation of boundary data structures. *IEEE Computer Graphics and Applications*, 14(6):66–77, 1994.
[24] C. Prince. Progressive meshes for large models of arbitrary topology. Technical report, Master's thesis, University of Washington, 2000.
[25] E. Puppo and R. Scopigno. Simplification, lod and multiresolution - principles and applications. In *Eurographics '97 Tutorial Notes*. ACM, 1997.
[26] J. Rossignac and P. Borrel. Multi-resolution 3d approximations for rendering complex scenes. In *Modeling in Computer Graphics: Methods and Applications*, pages 455–465, 1993.
[27] W. J. Schroeder, J. A. Zarge, and W. E. Lorensen. Decimation of triangle meshes. In *Proceedings of Siggraph 92*, pages 65–70. ACM, 1992.
[28] C. T. Silva, Y.-J. Chiang, J. El-Sana, and P. Lindstrom. Out-of-core algorithms for scientific visualization and computer graphics. In *Proceedings of IEEE Visualization 2002, Tutorial #4*. IEEE Society Press, 2002.
[29] F. Silva and A. Gomes. Adjacency and incidence framework - a data structure for efficient and fast management of multiresolution meshes. In *Proceedings of GRAPHITE 2003*, pages 159–166. ACM Press, 2003a.
[30] F. Silva and A. Gomes. AIF - a data structure for polygonal meshes. In *Proceedings of Computational Science and Its Applications (ICCSA 2003) - Workshop on Computer Graphics and Geometric Modeling*, pages 478–487. LNCS Vol. 2669, part III, V. Kumar, M. Gravilova, C. Tan and P. L'Ecuyer (eds.), Springer-Verlag, 2003b.

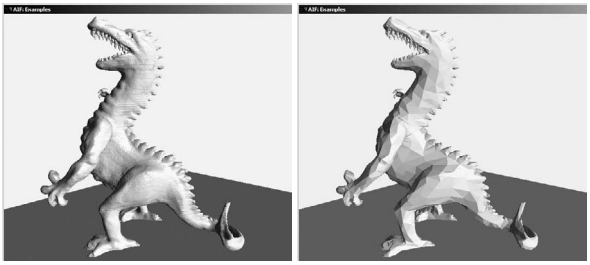

Figure 12. Dinosaur mesh, the original 50761 faces and simplified 8309 faces after 5 steps.

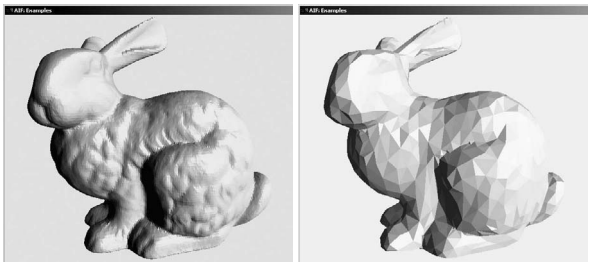

Figure 13. Bunny mesh, the original 69473 faces and simplified 2785 faces after 6 steps.

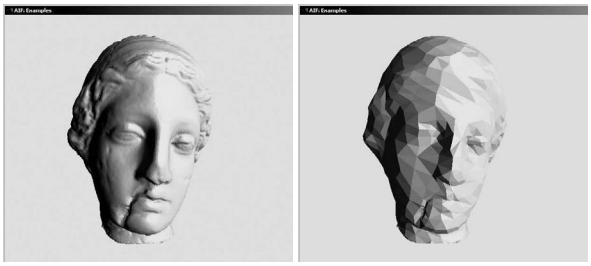

Figure 14. Venus mesh, the original 100000 faces and simplified 2546 faces after 6 steps.

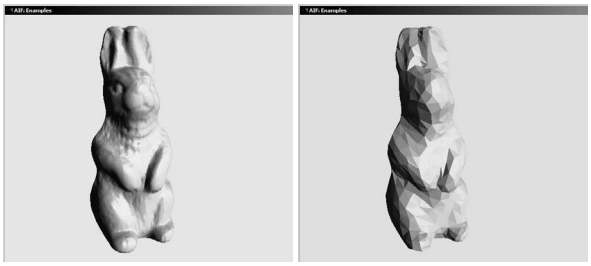

Figure 15. Rabbit mesh, the original 134073 faces and simplified 1421 after 7 steps.