Model Simplification Using Vertex-Clustering

Kok-Lim Low[†] and Tiow-Seng Tan[†]

National University of Singapore[‡]

Abstract

This paper presents a practical technique to automatically compute approximations of polygonal representations of 3D objects. It is based on a previously developed model simplification technique which applies vertex-clustering. Major advantages of the vertex-clustering technique are its low computational cost and high data reduction rate, and thus suitable for use in interactive applications. This paper advances the technique with careful consideration of approximation quality and smoothness in transitions between different levels of simplification, while maintaining its efficiency and effectiveness. Its major contributions include: accuracy in grading vertices for indication of their visual importance, robustness in clustering for better preservation of important features and consistencies between levels of simplification, thick-lines with dynamic normals to maximize visual fidelity, and exploitation of object and image space relationship for levels-of-simplification determination.

CR Categories and Subject Descriptors: I.3.5 [Computer Graphics]: Computer Geometry and Object Modeling - surface and object representations.

Additional Keywords: model simplification, levels of detail, vertex-clustering, thick-line.

[†] E-mail: {lowkokli | tants}@iscs.nus.sg

[‡] Department of Information Systems and Computer Science, National University of Singapore, Lower Kent Ridge Road, Singapore 119260.

This work is supported by the National University of Singapore under grant RP940641.

1997 Symposium on Interactive 3D Graphics, Providence RI USA

1 INTRODUCTION

Realistic-looking 3D models in many interactive 3D graphics applications may easily contain millions of polygons—far more than current workstations can render at interactive frame rates. Graphics acceleration techniques such as viewing frustum culling, visibility partitioning of scene [16] and hierarchical z-buffer [7] can, in general, improve graphics performance significantly, but become helpless in scenes where a large number of objects can be visible at the same time.

The key to an acceptable solution for real-time graphics of complex scenes is to have a series of geometric approximations that resemble the original models from all directions, but with increasingly lower rendering costs. Furthermore, the transitions between one approximation and the next must be barely noticeable in order to effectively reveal details as objects approach the viewpoint [13]. Recently, many model simplification techniques have been developed to tackle the problems. Some of these techniques are good for regular and dense triangular meshes approximating smooth surfaces [4, 8, 15, 17], while some are suitable for surface fitting of regularly spaced 3D digitized data points [3, 11], and others are able to deal with more general surface meshes [1, 2, 9, 10, 12].

We are interested in architectural, mechanical and hierarchical models for interactive CAD applications. Our models are usually complex, irregular (having very large as well as very small polygons), and have many sharp edges and corners which are visually very important. Also, they are often made up of a great number of simple polyhedra. As such, desirable techniques should be efficient, and effective in simplifying across polyhedra without first joining these polyhedra that intersect one another to form bigger polyhedra (which unnecessarily introduces many new polygons and computational cost). The vertex-clustering method introduced in [13, 14] satisfies the requirements, and can achieve high data reduction rate with very low computational cost by omitting the preservation of topology of the models. This paper is a thorough study of the vertex-clustering method with careful consideration of approximation quality and

smoothness in transitions between different levels of simplification during interactive viewing.

Section 2 provides the overview of the vertex-clustering method, and Section 3 discusses our enhancements towards quality approximations and smooth transitions between levels of simplification. Section 4 describes implementation issues to maintain the efficiency of the original technique, and Section 5 presents our experimental results. Lastly, Section 6 concludes the paper.

2 BASIC PRINCIPLE

In a synthetic scene, when an object is far away from the viewpoint, its image size is small. Due to the discreteness of the image space, many points on the object are mapped onto the same pixels, and this happens often when the object's model is complex and the image size is relatively small. For points mapped to the same pixel, only one point appears on the image at the pixel, and the others are eliminated by hidden-surface removal. This is a wastage in rendering as many of such points are processed but never make their way to the final image. A potential solution to cut down this wasteful processing is to find out which are the points that are going to fall onto the same pixel and use a new point to represent them. Only this new point is sent for rendering.

The vertex-clustering method applies the above principle. The clustering process determines the closeness of the vertices in the object space, and for those vertices found to be close to one another (which are likely to be mapped onto the same pixel), a new representative vertex is created to replace them. Indirectly, determining the closeness of the vertices also helps to determine the closeness of the polygons. For example, two rectangles is close together if their corresponding vertices are close to each other. When each pair of the corresponding vertices are represented by a new vertex, the two rectangles are indirectly fused to become one rectangle (after removal of the duplicate). By using different clustering-cell sizes, we will have different definition of "closeness", and this allows us to simplify the original model to models of different *levels of detail* (LODs).

Specifically, the process has the following steps: (1) *grading*—a weight is computed for each vertex according to its visual importance, (2) *triangulation*—polygons are divided into triangles, (3) *clustering*—vertices are grouped into clusters based on geometric proximity, (4) *synthesis*—a vertex representative is computed to replace the vertices in each cluster and thus simplified some triangles into edges and points, (5) *elimination*—duplicated triangles, edges and points are removed, and (6) *adjustment of normals*—normals of resulting edges and triangles are reconstructed. Figure 1 depicts these steps.

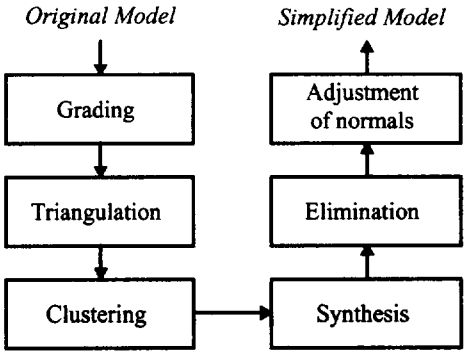

Figure 1: Steps in the vertex-clustering method.

3 ENHANCEMENTS

Though the original vertex-clustering method is able to produce high data reduction rate with very low computational cost, the approximation quality and smoothness in transitions between different levels of detail remain a concern in achieving visual fidelity and good animation. The followings are our contributions in these aspects. Interestingly, all enhancements maintain the efficiency of the method for 3D interactive applications.

3.1 Accuracy in Grading

Grading of vertices, being the first step in the simplification process, is very crucial to the success of the method. Two factors were proposed in [13] to determine the visual importance of a vertex. They are (1) the vertex's probability of lying on the object's silhouettes from an arbitrary viewing direction, and (2) size of the faces bounded by the vertex. With this, Factor 1 may be estimated using the inverse of the maximum angle between all pairs of incident edges on the vertex, and Factor 2 using the length of the longest among all of the edges incident upon the vertex.

The estimation for Factor 2 is logical, but that of Factor 1 is not clear. Let θ be the maximum angle between all pairs of incident edges on the candidate vertex. There is no clear indication as to how $\frac{1}{\theta}$ reflects the candidate vertex's probability of lying on the object's silhouettes from an arbitrary viewing direction, and moreover, because $\frac{1}{\theta}$ tends to infinity when θ approaches zero, its value will

overshadow that of Factor 2 when θ is small. In fact, $\cos\frac{\theta}{2}$ will be a more appropriate estimation for Factor 1. Here is the argument (see Figure 2):

Let V be the candidate vertex to be assigned weight. Suppose θ is the maximum angle between all pairs of incident edges on V. Now, we center a sphere of radius r around V, and also let V be the apex of a cone with tip angle θ and the base of the cone touches the sphere. A point on the sphere to V will represent a viewing direction from the point to V.

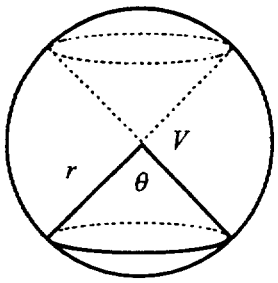

Figure 2: Grading estimation for Factor 1.

The apex of the cone represents a corner of the object, and the inside of the lower cone is the interior of the object. The only directions in which V cannot be seen are directions from the cap below the base of the cone. So if we choose any arbitrary viewing direction, the probability of seeing V on the silhouettes is

$$\frac{\text{area of sphere - area of bottom cap - area of top cap}}{\text{area of sphere}}$$

$$= \frac{4\pi r^2 - 2\pi r^2(1-\cos\frac{\theta}{2}) - 2\pi r^2(1-\cos\frac{\theta}{2})}{4\pi r^2}$$

$$= \cos\frac{\theta}{2}$$

3.2 Robust Floating-Cell Clustering

In [13], a simple clustering process is used, in which the object's bounding box is uniformly subdivided into cells, and vertices falling within one cell form a cluster. Let us call this the *uniform-subdivision clustering*. Here, we introduce an equally simple clustering method, called *floating-cell clustering*, which can significantly improve the quality of approximations and smoothness in LOD transitions. The following is an outline of the method:

1) sort all vertices in the vertex-list in non-increasing order on their weights,

2) the vertex with the highest weight will be the center of a new clustering-cell, and all vertices fall within the cell are removed from the vertex-list and replaced by a unique representative vertex (the representative is not added to the vertex-list).

3) repeat (2) for the next highest weighted vertex in the vertex-list.

Using this method of clustering, besides cubes, the clustering-cells can be other simple shapes, such as spheres. The following 3 subsections discuss the issues addressed by the floating-cell clustering.

3.2.1 Preservation of Important Features

Here is an interesting observation on uniform-subdivision clustering. When the heaviest vertex in the clustering-cell is not at the cell's center, the probability of including yet another heavily weighted vertex in the same cell can be high due to the large coverage of distance away from the heaviest vertex. It is notably undesirable to include many heavily weighted vertices in the same cell as their merging into one can greatly destroy important features of the models. On the other hand, the heaviest vertex is at the cell's center for floating-cell clustering, and thus reduce the coverage of distance away from the vertex. Therefore, it can greatly improves the preservation of important visual information. Furthermore, by having the most visually important vertices at the centers of the cells, their maximum errors are kept within half the cell's diagonal when vertices are replaced by their representative vertices.

3.2.2 Consistencies of Simplification

It is not favorable to derive a simplified model directly from yet another simplified model, as errors tend to accumulate in the lower LOD model. However, if all levels of simplification are derived directly from the original model, they may have significant differences which cause undesired impact on the smooth transitions between LODs. This deficiency is prominent in the uniform-subdivision clustering. In particular, uniform-subdivision is sensitive to the positioning of cells, caused by the shift in the cells or change in clustering-cell size for different levels of simplification. Figure 3 shows how a triangle can be simplified inconsistently due to differences in cell positions.

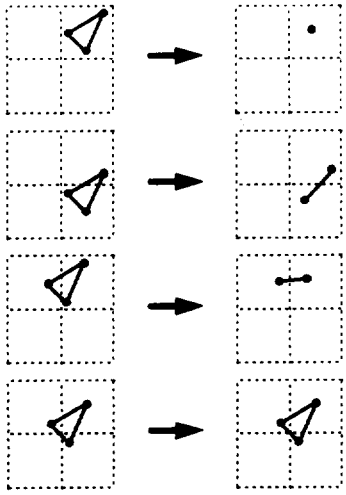

Figure 3: Inconsistent simplification.

On the other hand, the floating-cell clustering greatly minimizes this problem by using the weights of the vertices to control the positioning of the cells. This allows consistent cell positions across different levels of simplification.

3.2.3 Quality of Approximation

Here is yet another interesting problem of vertex-clustering where thin triangles are not collapsed to edges, instead, become much wider triangles. Figure 4 shows an example of such cases. In the diagram, vertex V_1 is heavier than V_2. Assuming we are using the floating-cell clustering, then V_3 belongs to cell C_1 instead of C_2. If weighted-average is used to synthesize the representative vertices R_1 and R_2, the triangle becomes much wider than before. This effect is very noticeable and can degrade the approximation quality significantly.

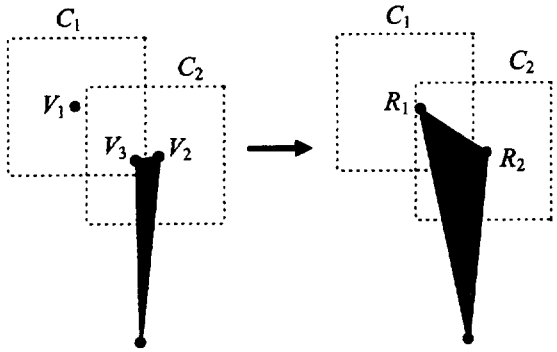

Figure 4: Thin triangle becomes much wider.

A simple variation to the floating-cell clustering greatly reduces such cases. The only change needed is on which cell a vertex belongs to when it falls within the intersection of two or more clustering-cells. For the original floating-cell clustering, this vertex is assigned to the cell which has the heaviest center vertex by the order of visit to vertices, whereas, for the variant, this vertex is assigned to the cell with the nearest center. Interestingly, such variation can be integrated easily, as discussed in Section 4, into the floating-cell scheme without additional computational cost.

3.3 Thick-Edges

Very often, when elongated parts of the object are reduced to just edges, they become less visible when rendered as one-pixel-wide lines. If the actual thickness (and the distance from the camera) of a elongated part in the object space is known, we can deduce its pixel-width on the screen. Then, by rendering the edge using thick-line primitive with varying pixel-width, we can approximately reproduce the actual visual effects of the elongated part.

3.3.1 Thickness

Thickness of an edge can be estimated during the clustering process. Let T be a set of triangles where each triangle has its vertices falling in clusters A and B only. Let V_A be the set of vertices that belong to cluster A and each is used by at least a triangle in T. Let P_A be the average position of all the vertices in V_A. And, V_B and P_B are similarly defined. Let L be the straight line passing through P_A and P_B. Let D_A be the average perpendicular distance of all points in V_A to the line L, and likewise for D_B. Then, the thickness of the edge, formed by collapsing triangles in T, is estimated by $\frac{2D_A + 2D_B}{2} = D_A + D_B$.

3.3.2 Dynamic Normals

To improve the shading or intensity of a thick-line which renders a thick-edge, we can give it a dynamic normal instead of a fixed one. This dynamic normal is only computed during rendering. It is acceptable to assume that the elongated part approximated by the edge is cylinder-like as models are usually some closed surface. With this assumption, the dynamic normal can be computed as the vector which is perpendicular to the edge and is also in the plane containing the edge and the camera. The angle between this dynamic normal and the vector from any point on the edge to the camera is acute. The dynamic normal is calculated using the formula $(V_1 - C) \times (V_2 - C) \times (V_2 - V_1)$ where V_1, V_2 are the edge's endpoints and C is the camera's position.

4 IMPLEMENTATION

As speed of simplification is a major concern in the vertex-clustering method, this section focuses on the change in computational cost resulted by the enhancements.

Grading. The change in this step is merely the use of $\cos\frac{\theta}{2}$ instead of $\frac{1}{\theta}$, where θ is the maximum angle between all pairs of incident edges on the vertex. So the time complexity has not changed. However, since $\cos\frac{\theta}{2}$ is slower to compute than $\frac{1}{\theta}$, we can use a small look-up table for $\cos\frac{\theta}{2}$ to achieve the necessary speed and accuracy.

Floating-cell clustering. The floating-cell clustering method requires the vertices to be sorted in non-increasing order on their weights. This step can be accomplished in $O(N \log N)$ time where N is the number of vertices in the model. For each model, this sorting step needs to be done only once, regardless of how many levels of simplified models are to be generated.

After the sorting step, the actual vertex-clustering can be done in $O(N)$ time for generating each simplified model. We first uniformly subdivide the object space into cubical grids, each with size equal to the clustering-cell, assuming we are using cubical cells. The main idea here is, whenever a new clustering-cell is created, we associate it to the grid that contains the cell's center vertex. Though clustering-cells can intersect one another, a cell's center vertex can never be inside another cell. This leads to a maximum of 8 clustering-cells that can be associated to each grid.

To start the actual clustering, we first look at a vertex, starting from the head of the vertex-list. Immediately, we can know which grid it belongs to (by truncation of vertex's coordinates). Now, we need to find out whether this vertex belongs to any existing clustering-cells. This is done by checking the cells associated to the grid which contains the vertex, and also the other 7 adjacent grids which are nearest to the vertex. Only the cells associated to these 8 grids can contain the candidate vertex.

We then check the distances from the candidate vertex to the center vertices of all the cells associated to these 8 grids. For the original floating-cell clustering, we pick the cell with the heaviest center vertex among the cells that contain the candidate vertex, whereas for the improved variant described in Section 3.2.3, we pick the cell with the nearest center from the candidate vertex. The candidate vertex is then assigned to the selected cell. If the candidate vertex does not belong to any of these cells, we create a new cell with the candidate vertex as the center, and associate the cell to the grid which contains it.

Estimating thickness of edges. After the clustering step, for each triangle which has exactly two vertices belonging to the same cluster, we associate it to the two clusters which contain its vertices. Each cluster has a binary search tree to store these associations, where each node contains an index to the associated triangle and the cluster number of the other cluster which also contains one or two of the triangle vertices. Then for each cluster A, we search the binary tree for those triangles which have their other vertices in another cluster B. Let T be this set of triangles. Let V_A be the set of T's vertices in A, and similarly for V_B. With these, $(D_A + D_B)$ can be computed as described in Section 3.3.1.

All the triangles in T, except one, can now be deleted from the model. The thickness is assigned to the only triangle left. All the associations of these triangles in T can be removed from the binary trees of clusters A and B.

LOD determination. There is a correspondence between each pixel space in the image space and each clustering-cell in the object space. Let D be the distance of an object from the viewpoint, which can be estimated from a bounding sphere of the object. Assuming the viewing window is a square with aspect ratio 1 and the field of view of the camera is FOV. Then the *object-space-to-pixel-space-ratio* R is just

$$R = \frac{2D\tan\frac{FOV}{2}}{\text{window's height}}.$$

For rendering, we would like to choose a model with the largest clustering-cell width which is less than or equal to R. The value of R is also used in the computation of the pixel-widths of thick-lines. For a thick-edge with estimated thickness T, the pixel-width of the thick-line is $\frac{T}{R}$.

The above LOD determination criterion is based on a *one-cell-width-to-one-pixel-width* basis. This criterion can generally be relaxed to allow a one-cell-width-to-X-pixel-width basis where $X > 1$. Experiments have shown that good values for X range from 5 to 10, depending on how well the perceptual information are preserved. The higher the value of X, the lower the LOD will be selected for an object with a distance D from the camera. So, by increasing the value of X, we trade visual accuracy for faster rendering. As a result of our enhancements to the vertex-clustering method, which generally produces good

simplified models, an even higher X value can be used to further speed up the rendering process with hardly noticeable sacrifice in accuracy. This also gives more leeway for adaptive LOD determination algorithms [6] to maintain high constant frame rates.

5 EXPERIMENTAL RESULTS

Experiments have been carried out to compare the results of the original vertex-clustering method [13] to that of our improved version. The comparisons were made on the approximation quality of the simplified models, their rendering costs (in terms of number of triangles, edges and points) and the smoothness in transitions between LODs.

Some of the experimental results are shown in the color plates. The following table shows the number of vertices, triangles, edges and points each model has:

plate	vertices	triangle	edges	points
original models				
1(a)	782	1528	0	0
1(b)	1254	2512	0	0
1(c)	406	808	0	0
1(d)	1947	3840	0	0
1(e)	2720	5100	0	0
1(f)	1742	3480	0	0
simplified using uniform-subdivision clustering				
2(a)	28	39	6	0
2(b)	101	193	2	0
2(c)	28	52	2	1
2(d)	16	36	0	0
2(e)	92	94	34	0
2(f)	32	60	0	0
simplified using floating-cell clustering				
3(a)	20	10	9	0
3(b)	60	109	10	0
3(c)	23	40	6	1
3(d)	16	20	3	0
3(e)	95	18	78	0
3(f)	32	60	0	0
edges rendered using thick-lines				
4(a)	20	10	9	0
4(b)	60	109	10	0
4(c)	23	40	6	1
4(d)	16	20	3	0
4(e)	95	18	78	0
consistencies in simplification				
5(a)(i)	92	94	34	0
5(a)(ii)	109	137	25	0
5(b)(i)	95	18	78	0
5(b)(ii)	102	36	78	0

Plate 1(a) to 1(f) shows the original models. Plate 2(a) to 2(f) are the models simplified using uniform-subdivision clustering, and Plate 3(a) to 3(f) are corresponding ones from the floating-cell clustering, using the same clustering-cell size and shape (cube). It is not difficult to see that the models in Plate 3 are better approximations compared to those in Plate 2.

Plate 4(a) to 4(e) show the effect of rendering thick-edges using thick-line primitives. The thick-lines, together with its dynamic normals, are quite successful in reproducing the actual visual effects of the original elongated parts.

Plate 5 shows the simplified models of our leafless tree and the same tree with a green pot. The green pot has forced a change in the bounding box of the model, and thus causes differences in cell positioning for the uniform-subdivision clustering. As a result, as shown in Plate 5(a)(i) and 5(a)(ii), the two simplified models produced by the uniform-subdivision clustering are difference in regions far from the bottom of the tree. On the other hand, the two models simplified by the floating-cell clustering are the same, except for the pot. This demonstrates the simplification consistencies of the floating-cell clustering.

Additionally, our experiment shows that the use of different cell sizes, when producing models of different levels of simplification, often causes change in cell positioning in the uniform-subdivision clustering. This results in undesirable differences in consecutive levels of simplification that has adverse effect on the smoothness in transitions between LODs during interactive viewing. This, however, is not an issue for floating-cell clustering.

6 CONCLUSION

Our studies and implementation of the robust floating-cell clustering coupled with the proposed grading scheme have resulted in quality approximations and smooth transitions between different levels of simplification, while maintaining the efficiency and effectiveness of the original vertex-clustering method. Besides, the simple idea of rendering edges (resulted from simplification) using thick-lines of varying widths and dynamic normals can improve visual fidelity of the displayed image. Additionally, as a result of the good approximations, on-line level-of-detail determination with relaxed criterion is applicable to further speed up the rendering process with hardly noticeable sacrifice in visual accuracy.

Acknowledgments

We would like to thank Kelvin Sung and Teh Hung Chuan for their helpful discussions.

References

[1] Algorri, Maria-Elena and Schmitt, Francis. *"Mesh Simplification"*. Eurographics '96, pp.C77-C86, 1996.

[2] Cohen, J., Varshney, A., Manocha, D., Turk, G., Weber, H., Agarwal, P., Brooks, F. and Wright, W. *"Simplification Envelopes"*. Computer Graphics (SIGGRAPH '96 Proceedings), 1996.

[3] DeHaemer, Michael J., Jr. and Zyda, Michael J. *"Simplification of Objects Rendered by Polygonal Approximations"*. Computers and Graphics, Vol. 15, No. 2, pp. 175-184, 1991.

[4] Eck, M., DeRose, T., Duchamp, T., Hoppe, H., Lounsbery, M., Stuetzle, W. *"Multiresolution Analysis of Arbitrary Meshes"*. Computer Graphics (SIGGRAPH '95 Proceedings), pp. 173-182, 1995.

[5] Erikson, Carl. *"Polygonal Simplification: An Overview"*. Technical Report TR96-016, Department of Computer Science, CB #3175, Sitterson Hall, UNC-Chapel Hill, Chapel Hill, NC 27599-3175, 1996.

[6] Funkhouser, Thomas A. and Séquin, Carlo H. *"Adaptive Display Algorithm for Interactive Frame Rates During Visualization of Complex Virtual Environments"*. Computer Graphics (SIGGRAPH '93 Proceedings), pp. 247-254, 1993.

[7] Green, N., Kass, M., and Miller, G. *"Hierarchical z-buffer Visibility"*. Computer Graphics (SIGGRAPH '93 Proceedings), pp. 231-238, 1993.

[8] Hamann, Bernd. *"A Data Reduction Scheme for Triangulated Surfaces"*. Computer Aided Geometric Design, Vol. 11, pp. 197-214, 1994.

[9] Hoppe, H., DeRose, T., Duchamp, T., McDonald, J. and Stuetzle, W. *"Mesh Optimization"*. Computer Graphics (SIGGRAPH '93 Proceedings), pp. 19-26, 1993.

[10] Hoppe, Hugues. *"Progressive Meshes"*. Computer Graphics (SIGGRAPH '96 Proceedings), 1996.

[11] Lindstrom, P., Koller, D., Ribarsky, W., Hodges, L. F., Faust, N. and Tuner, G. *"Real-Time, Continuous Level of Detail Rendering of Height Fields"*. Computer Graphics (SIGGRAPH '96 Proceedings), 1996.

[12] Ronfard, Rémi, and Rossignac, Jarek. *"Full-Range Approximation of Triangulated Polyhedra"*. Eurographics '96, pp.C67-C76, 1996.

[13] Rossignac, Jarek and Borrel, Paul. *"Multi-Resolution 3D Approximations for Rendering Complex Scenes"*. Modeling in Computer Graphics, B. Falcidieno and T. L. Kunii, Eds. Springer-Verlag, pp. 455-465, 1993.

[14] Rossignac, Jarek. *"Geometric Simplification"*. Course Notes 32 on *"Interactive Walkthrough of Large Geometric Databases"*, SIGGRAPH '95, pp. D1-D14, 1995.

[15] Schroeder, William J., Zarge, Jonathan A. and Lorensen, William E. *"Decimation of Triangle Meshes"*. Computer Graphics (SIGGRAPH '92 Proceedings), pp. 65-70, 1992.

[16] Teller, Seth J. *"Visibility Computations in Densely Occluded Polyhedral Environments"*. Ph.D. thesis, Computer Science Division (EECS), University of California, Berkeley, 1992.

[17] Turk, Greg. *"Re-Tiling Polygonal Surfaces"*. Computer Graphics (SIGGRAPH '92 Proceedings), pp. 55-64, 1992.