A General Algorithm for Triangular Meshes Simplification

BOŠTJAN PIVEC
University of Maribor
Faculty of EE and CS
Smetanova 17, 2000 Maribor
SLOVENIA

VID DOMITER
University of Maribor
Faculty of EE and CS
Smetanova 17, 2000 Maribor
SLOVENIA

Abstract: This article introduces the algorithm for simplification of 2D and 3D triangular meshes. The algorithm evaluates and removes vertices. A hash table is used to speed up the process of selecting the appropriate vertex for its removal. Therefore, this part of the algorithm is executed in constant time $O(1)$. In spite of simplification, triangular mesh models still keep majority of their own essential characteristics. With simplification we also ease transfer of such models through the web.

Key–Words: Simplification, Triangular Mesh, Average Plane, Hash Table, 3D Model

1 Introduction

Nowadays, the majority of free-form geometric objects are represented with triangular meshes. Due to high performance of today's computers, meshes can be described with more than 1.000.000 triangles. Such meshes can be rendered in real-time even on low-cost personal computers. However, the problem occurs when transferring such large meshes through the web. Namely, beside geometric data (coordinates of vertices), triangular meshes are also described by topological information, which defines how triangles fit together. Hence, it follows that special-purpose methods for topology compression have been proposed [1, 2]. Fortunately, high precision of transferred triangular meshes is not always required. Sometimes we are satisfied only with good visual representation of the geometric model. So we can afford to lose some data with simplification. There are three main approaches to choose from:

- The most frequently used methods are based on Schroeder's et. al. simplification algorithm [3, 4]. First, vertices are evaluated and then they are incrementally removed from the mesh according to their importance.
- Edge simplification methods evaluate and remove edges. Removed edge is replaced with a vertex [5].
- Simplification based on triangles is possible in theory yet practical solutions have not been reported.

The algorithm presented in this article is based upon Schroeder's vertex simplification method [3]. To speed-up the search for the most suitable vertex to be removed, a hash table has been used [6]. Krivograd et.al. suggested this approach for 2.5D triangular meshes [7]. The presented algorithm expands Krivograd's work in 3D.

2 Algorithm

The algorithm supports 2D, 2.5D and 3D triangular meshes. For the method, on which it is based upon, it is characteristic that we have to evaluate all vertices correctly, before we can remove them. Input in algorithm is a list of vertices and a list of triangles. The algorithm works in the following steps:

1. Evaluation of all vertices of the triangular mesh.
2. Arrangement of evaluated vertices into the hash table.
3. Selection of the most proper vertex for removal.
4. Removal of the selected vertex from the hash table and triangular mesh.
5. Removal of the surrounding triangles of the deleted vertex.
6. Triangulation of the empty space which occurs as a consequence of removed triangles.
7. Reevaluation of neihghbouring vertices of deleted vertex.
8. Rearrangement of reevaluated vertices in the hash table.
9. Returning to step 3 until final condition is reached.

2.1 Evaluation of vertices

The first step of our algorithm is the evaluation of all vertices in a triangular mesh. In this way, each vertex is evaluated according to its importance. Lower is the evaluation factor, the less important is the vertex. There are different possibilities for the evaluation criteria. For example, in 2.5D, two approaches are proposed [7]:

- The vectors connecting the chosen vertex and its neighbouring vertices are constructed. After that, angles between these vectors and the appropriate plane is calculated. Clearly, for 2.5D triangular meshes this plane is plane XY. The average value of all angles between this plane and vectors, which are defined by chosen vertex and its neighboring vertices, is considered as an evaluation factor.
- All distances from the evaluated vertex and its neighbouring vertices to XY plane are computed. The evaluation factor is an average difference of distances between chosen vertex and its neighbouring vertices regarding the XY plane.

In our approach, the second possibility is generalized to 3D triangular meshes. Instead of using an XY plane, we use an average plane on which the evaluation factor is calculated. The evaluation factor represents the distance between the evaluated vertex and the line defined by its neighbouring vertices, if the evaluated vertex has only two neighbours. This case can occur only in the corners of non-closed triangular meshes. The most frequent is the situation where the evaluated vertex has three or more neighbouring vertices. In this case, the average plane has to be calculated.

The first step in determination of the average plane is calculation of all possible combinations of three arbitrary neighbouring vertices. Each combination defines a plane in the space described by a normal vector. The average normal vector, which determines the average plane, is calculated as the avarage value of all normal vectors. Thus the evaluation factor represents the average distances between the selected vertex (v_1 in figure 1) and its neighbouring vertices regarding the average plane.

2.2 Hash table

The smallest visual change in a triangular mesh is caused by deleting a vertex with the smallest evaluation factor. After deleting such vertex, the evaluation factor of its neighbouring vertices has to be recomputed. After that, the vertex with the smallest evaluation factor has to be found again. In this way, $O(n^2)$ time complexity is reached. To speed up the selection of the vertex for its removal, a hash table is applied as

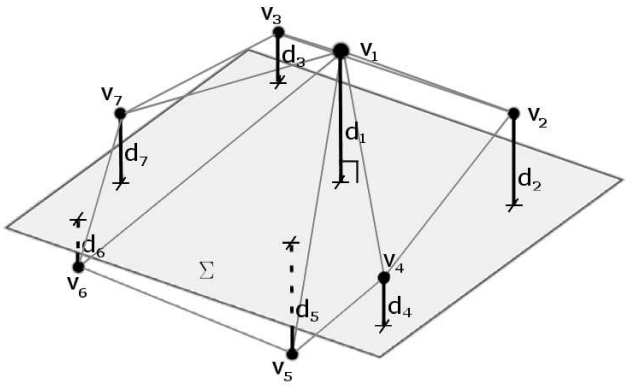

Figure 1: Evaluation of the vertex v_1 according to average plane Σ

suggested by Franc and Skala [6] (figure 2). Vertices are inserted into hash table according to the evaluation factor. The same heuristic for constructing the hash table as proposed in [7], has been used.

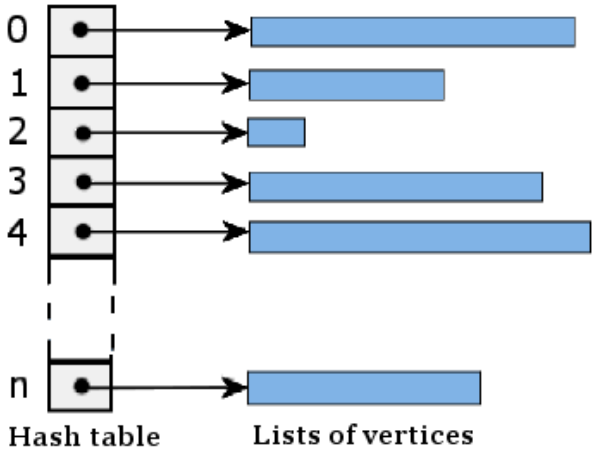

Figure 2: Structure of our hash table where **n** is number of intervals

The algorithm always selects the first vertex from the lowest non empty hash table entry. The surrounding neighbours of removed vertex have to be evaluated again. Normally, reevaluation change their evaluation factor and sometimes also the interval in the hash table. The reevaluated vertices are always placed at the end of the list of vertices in the corresponding interval of the hash table. In this way, the local simplification of the triangular mesh is prevented.

The hash table is also used to preserve basic shape of the geometric objects. This can be accomplished by preserving the last list of vertices in the hash table,

where the edge vertices and vertices with the highest evaluation factor are stored.

2.3 Triangulation

Together with the deleted vertex, all triangles that are defined with this vertex are removed, too. The result is a hole in the triangular mesh which should be filled with new triangles. The outer edges of removed triangles form a polygon which has to be triangulated. This polygon is not necessarily planar. Therefore, all vertices of this polygon are mapped on the average plane (see section 2.1). In figure 3 vertices v_2 to v_6 are projected onto vertices v_2' to v_6' of the planar polygon.

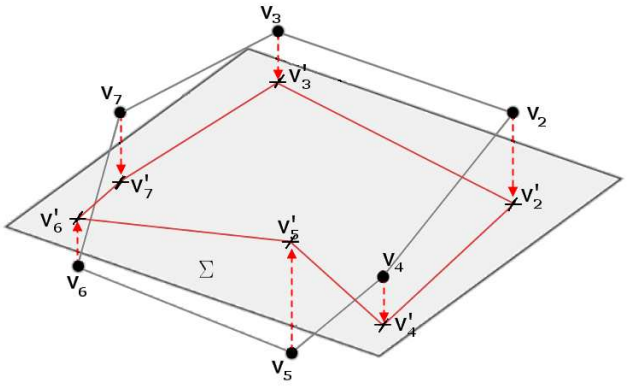

Figure 3: Projection of vertices on average plane Σ

If the polygon is convex, the triangulation is trivial, but if it is concave, well known ear cutting [8] method is applied. In figure 4 we can see a triangulation of the concave polygon.

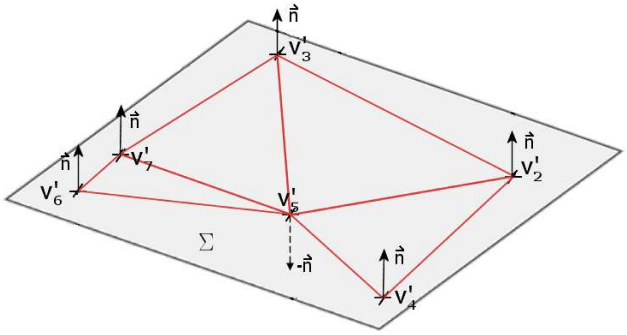

Figure 4: Triangulation of concave polygon

When the triangulation of the planar polygon is finished the newly constructed triangles are mapped in the triangular mesh. Figure 5 shows the innitial polygon filled with new triangles.

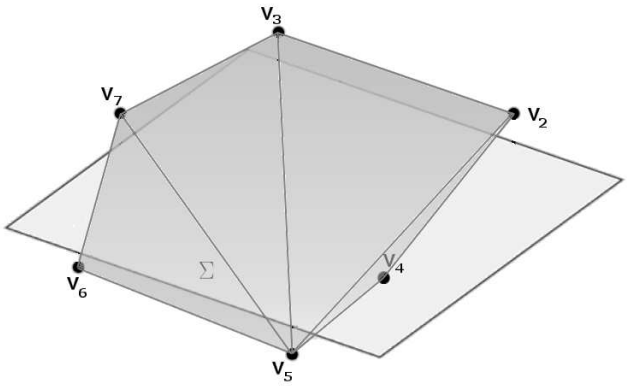

Figure 5: Result of triangulation

3 Results

In this chapter we estimate time complexity of the algorithm as follows:

- Evaluation of vertices. l neighbouring vertices of the evaluated vertex are needed for the evaluation factor calculation. As $l << n$ we can suppose that constant time $O(1)$ is needed. Therefore, all vertices are evaluated in time $O(n)$.
- Hash table. Constructing the hash table and filling it with vertices is done in linear time $O(n)$. In each step, the selected vertex is deleted from the hash table in constant time $O(1)$. There are $l << n$ reevaluated vertices. Each is inserted into the hash table in constant time as well.
- The triangulation using the ear cutting method of a polygon having l vertices is done in time $O(l^2)$. But, as $l << n$ the task can be considered as finished in constant time $O(1)$ per removed vertex.

Thus, the common time complexity of the algorithm is $O(n)$.

The algorithm was tested on various triangular meshes found on the web. The simplification of the pumpkin model is presented in figures 6, 7 and 8.

We simplified the mesh to a level that still assures good visual appearance. Results of the simplification are shown in table 1. The graph in figure 9 confirms good behaviour of the algorithm. All measurements were performed on a computer with AMD Athlon XP 2800+ processor and 1GB DDR 333Mhz RAM.

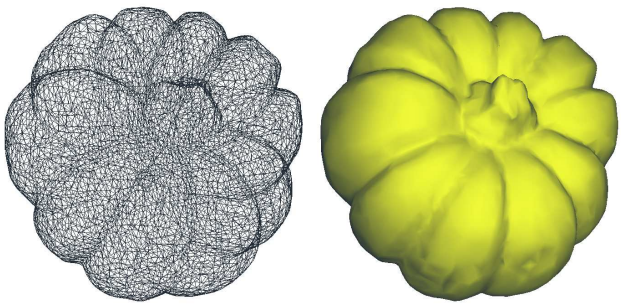

Figure 6: Pumpkin model with 5002 vertices and 10000 triangles

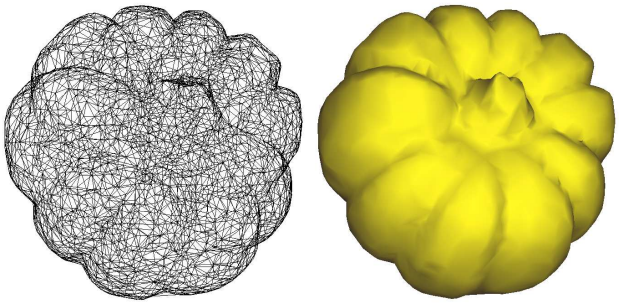

Figure 7: Simplified model with 2501 vertices and 4998 triangles

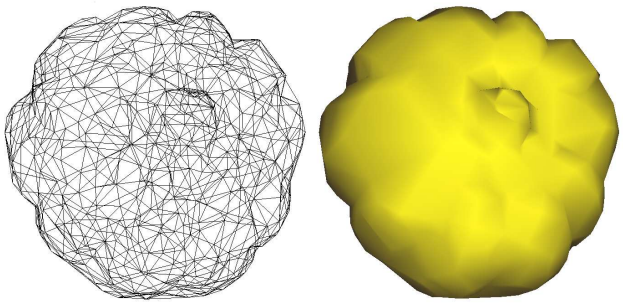

Figure 8: Simplified model with 501 vertices and 998 triangles

Table 1: Simplification results

Mesh	%	Vertices	Triangles	Time (s)
Pumpkin	90	4502	9000	0.062
	80	4002	8000	0.125
	70	3502	7000	0.188
Vertices: 5002	60	3002	6000	0.25
Triangles: 10000	50	2501	4998	0.297
	40	2001	3998	0.359
	30	1501	2998	0.406
	20	1001	1998	0.438
	10	501	998	0.484

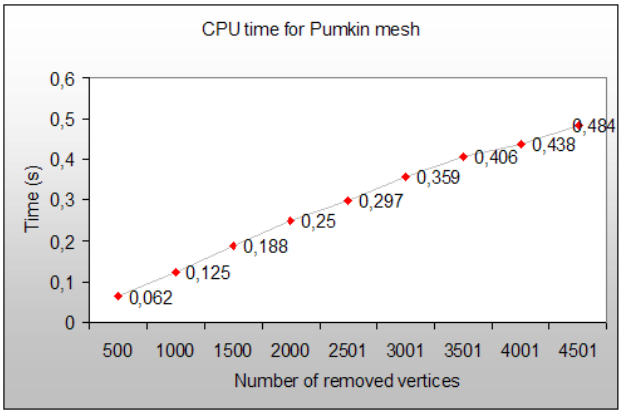

Figure 9: Time graph for simplification of Pumpkin model

4 Conclusion

This paper presents an efficient and general algorithm for simplification of triangular meshes. It is based on Schroeder's et. al. idea of vertex removal. In general, the selection of vertex for its removal requires $O(n^2)$ time. Krivograd's et. al. applied a hash table for accelerating this selection. However, their method is limited only to 2.5D triangular meshes. In this paper, we have developed an algorithm that operates in 3D. For this, the new evaluation criteria is proposed and implemented.

Acknowledgements: The research project was supported by the University of Maribor, Faculty of Electrical Engineering and Computer Science. The first author received research grant for one year.

References:

[1] C. Touma, and C. Gotsman, *Triangle Mesh Compression*, In: Graphics Interface, 1998, pp 26-34.

[2] P. Alliez, and M. Desbrun, *Valence-Driven Connectivity Encoding for 3D Meshes*, Computer Graphics Forum, 2001, vol. 20, no. 3, pp. 480-489.

[3] William J. Schroeder, Jonathan A. Zarge, William E. Lorensen, *Decimation of Triangle Meshes*, Computer Graphics, 1992, vol. 26, no. 2, pp 65-70

[4] Michael Garland, Paul S. Heckbert, *Fast Polygonal Approximation of Terrains and Height Fields*, technical report, CMU-CS-95-181, 19. September 1995

[5] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, Werner Stuetzle, *Mesh Optimization*, ACM Computer Graphics - SIGGRAPH'93, Anaheim, California, Unated States, 1993, pp. 19-26

[6] M. Franc, V. Skala, *Parallel Triangular Mesh Decimation Without Sorting*, SCCG Proceedings, Budmerice, 2001, pp. 69-75

[7] Sebastian Krivograd, Borut Žalik, Franc Novak, *Triangular mesh decimation and undecimation for engineering data modelling*, Inf. MIDEM, September 2002, vol. 32, no. 3

[8] Marko Lamot, Borut Žalik, *A fast polygon triangulation algorithm based on uniform plane subdivision*, Computer & Graphics, 2003, vol. 23, no. 2, pp. 239-253