Mesh Simplification

Keerti Sekhar Sahoo and Xiao Mao
University of San Francisco

Abstract

There are various method to simplify the triangular geometry of small, distant and unimportant portions of the model and reduce the rendering process at the same time maintaing the quality of the scenes outcome. In this paper we present a simple and effective technique for simplifying triangular meshes. It is based on decimation of triangles and vertex clustering techniques and produces a final output that resembles the original object from all the view points, but contains an considerably fewer number of faces and vertices or primitives.

1 Introduction

Our goal is to approximate a complex, high-resolution triangular mesh with simpler geometry that resembles the original mesh but with fewer primitives. Simplifying a mesh requires us to take into account the visual importance of primitives involved in the process. The effective way to reduce the level of complexity of the mesh without introducing any geometric error calls for considering the primitives those are not visible like the inner side of the object. This paper simplifies the selection process by classifying each vertex into an appropriate category. By implementing this classification, we decide the complexity level of the primitive and based on the defined criteria the primitive becomes the candidate for the simplification process or excluded from the process. In case of vertex clustering methodology, each vertex is assigned a weight value based on the visual importance of the vertex. The rest of the steps for the approaches revolves around our known premise, which is classify the primitives based on importance and merge vertices based on proximity. The algorithms are elaborated in the paper.

2 First Approach: Decimation of Triangles

This approach is simple and the goal is to reduce the total number of triangles in a triangle mesh. And preserve the original topology and a good approximation of the original geometry

2.1 Overview

In this approach first we classify the vertices according to their visual importance and their local topology. After the classification is done, we go through all the vertices, where each vertex is considered as a candidate for deletion and it is decided based on our decimation criteria. In this process, after deletion of a vertex, the surrounding triangles disappear from the mesh and create a loop or hole. The resulting hole is then considered for local triangulation. The decimation process is carried on till the user specified iterations and the triangulation is carried on as well. For this we have an efficient data structure, which represent the relations between different primitives of the mesh. E.g. relationship between vertices and their surrounding triangles or relationship between each vertex and its surrounding vertices.

2.2 Data Structure

Since we have a large amount of data, the data structure needs to be efficient, while still providing the functionality we need. Most importantly it must allow the efficient retrieval of all the primitives of a vertex, i.e. finding all the neighboring vertices or edges of each vertex. This data structure is similar to hierarchical ring structure. This consists of structures for storing the triangles neighboring each vertex, neighboring vertices of each vertex, connected edges for each vertex and finally the vertices for each triangle.

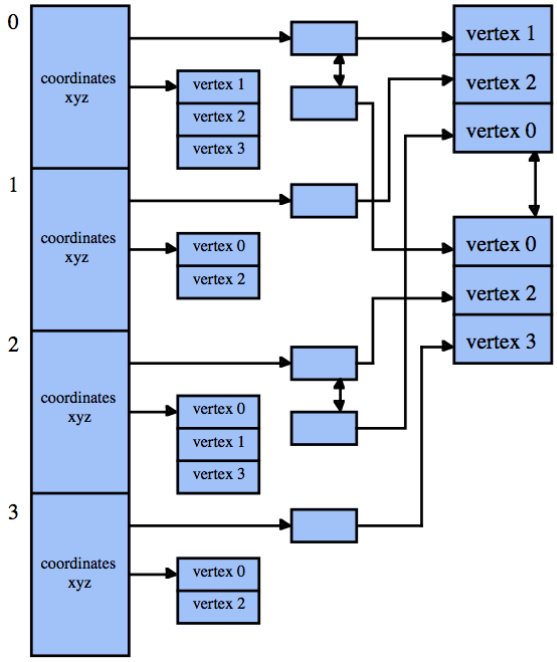

Figure 1

2.3 Methodology

Basically for each vertex, three stpes are required:

2.3.1 Classifying Local Geometry and Topology

In this approach we classify each vertex based on three criteria i.e. by topology, by its geometry and by its neighboring normals. There are five types of classification for all the vertices of a mesh.1) Simple: A vertex is considered as simple vertex if it is surrounded with complete cycle of triangles and each edge that uses the vertex is exactly used by two triangles. 2) Complex: If the edge is not used by two triangles, the corresponding vertex is a complex vertex. 3) Boundary: If a vertex is surrounded by semi cycle of triangles then its a boundary vertex. If an edge is shared by two triangles, the angle between their face normals is calculated and if its greater than

a specified angle, the edge is called a feature edge. 4) Interior: If a vertex is used by two feature edges its called interior vertex. 5) If the vertex is used by one or three or more feature edges, its called a corner vertex.

Figure 2

2.4 Evaluating Decimation Criteria

In this approach the decimation criteria is evaluated based on the distance of the vertex from an average plane. Let X be the candidate vertex. So all the triangles surrounding X are identified and the base point on the average plane is calculated by the formulae:

$$B = \sum \frac{(A_i * C_i)}{(A_i)} \tag{1}$$

Where B is base point on the average plane. Ci is the center and Ai is the area of triangle.
Then the distance from the vertex to the plane is calculated by the following formulae: (X - B).n as shown in the figure below:

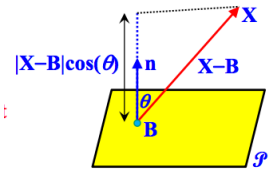

Figure 3

Where n is the face normal for the average plane.

2.5 Discussion

In this approach, we have implemented the functionalities till the second step of the algortihm. Which is classifying the vertices of a mesh into five categories based on their visual importance and the next step is to calculate the distance of the vertices to the average plane. According to the distance, the decimation criteria is decided for a candidate vertex. If the vertex is within specified distance, the vertex is deleted otherwise it is retained.

3 Second Approach: Vertex Clustering

The goal of this approach is to simplify a mesh by reducing number of triangles and at the same time maintaining the topology of the original geometry. This methodology revolves around the premise of selecting a representative vertex for a cluster of vertices in a mesh and the clustering process is incorporated with the new set of primitives.

3.1 Overview

In this approach first we calculate a weight value for each vertex based on the perceptual importance of the primitive in the mesh. The next step is to decide a clustering cell size and based on the size of the cell. The most important factor influencing the LOD in this approach is the cell size, which is specified by the user and the mesh is simplified according to the selected value for the cell size. In the following step a representative vertex is selected for individual cluster followed by the triangulation and finally normals are adjusted according to the simplified vertex coordinates.

3.2 Methodology

3.2.1 Grading

A weight is computed for each vertex of the mesh and stored in a data structure. The weight defines the relative perceptual importance of the vertex. In this approach we take into consideration two factors to determine the visual importance of a vertex. Based on 1) Number of faces bounded by the vertex. 2) The inverse of the maximum angle between all pairs of incident edges on the vertex.

3.2.2 Clustering

In this method, based on the proximity, the vertices of the mesh are grouped into clusters. Clusters are numbered in the order they are created. Our approach is that a bounding box containing the object is uniformly subdivided into cells. Then the representative vertex is selected among the cluster vertices and the unique vertex replaces the vertices falling within one cell. The clustering procedure takes as parameters the bounding box. In this case we have used solids bounding box for the entire model.

3.2.3 Synthesis

In this approach we have calculated a representative vertex for each cluster using the associated weight values for the vertices present in a cell and the representative vertex is the weighted average of the values within a cell. E.g. The representative vertex of P1, P2.....Pk in the same cell is their weighted average, which is calculated as:

$$P = \frac{w1P1 + w2P2 + ...wkPk}{w1 + w2 + ...wk} \tag{2}$$

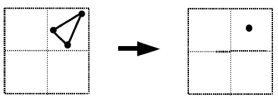

Figure 4: *Representative vertex selection.*

Since the mapping here is many to one, this method reduces the total number of vertices from the mesh.

3.2.4 Retriangulation

This approach is based on the premise that if 3 new generated edges can form a loop, then we create a triangle using these 3 edges.

3.2.5 Recalculation of Normals

Final step is to compute new normals for all the triangles using the entire simplified vertex coordinates. The normals are only used for rendering.

3.3 Previous Results

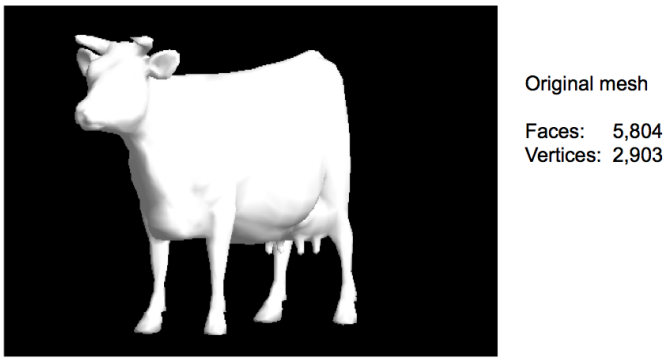

Figure 5: *Original mesh before simplification.*

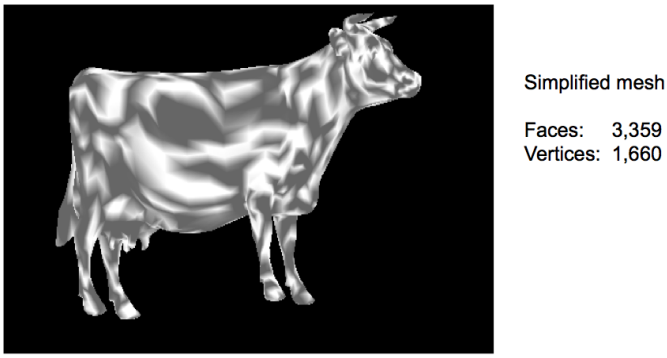

Figure 6: *Final mesh after simplification.*

This was the output of our approach during Beta release. As evident from the simplified version that the normals are not proper, which is beacuse after triangulation the order of the vertices in each face was not consistent and some face normals were not correct. Still we were able to get a simplified version of the orginal mesh. Which is in this case, the simplified version contains approximately 53 percent less number of primitives than the original mesh. This worked fine with a original mesh having less no of primitives. But for a mesh with hundreds of thousands of primitives the results were different:

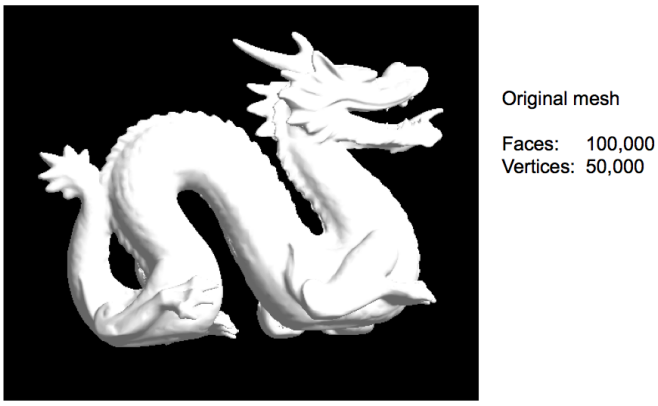

Figure 7: *Original mesh with large no of primitives.*

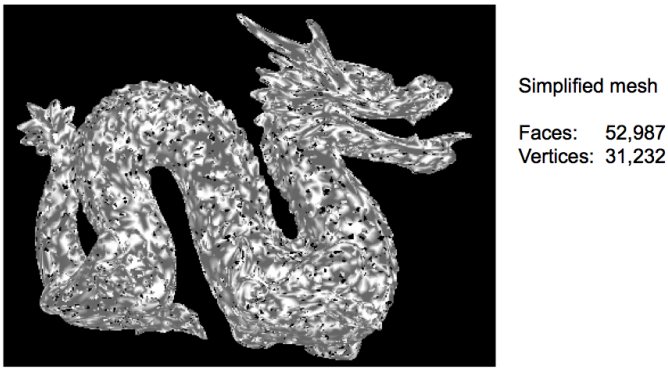

Figure 8: *Simplified mesh with holes.*

In the above simplification, the original mesh has large number of primitives, which is in hundreds of thousands in number. The outcome was simplified version of the orginal one but it had holes in it. So the main challenge was to fill these holes or consider a better calculation for the triangulation process to eliminate the generation of holes at the first place.

3.4 Different hole filling technique

To overcome the above mentioned problem, we used a third party library called MeshFix. The library was supposed to fill the holes and also take care of the order of the vertices in each face, when the mesh size is huge. And the outcome was not as expected and which is as follows:

Figure 9: *Mesh simplified with MeshFix.*

As we can see, the outcome was actually a simplified version of our original mesh and importantly the normals were proper. But as evident, the simplification process was decimating large portion of the mesh and which was clearly not going to help us for getting our desired result. Then, we found a different approach from another paper to fill the holes ourselves and to analyze the outcome.

3.5 New approach for filling holes

3.5.1 Hole Identification

The first step of hole-filling is to detect all the holes in the given triangular mesh model. Since a vertex-based topological structure is used in this project, all the boundary vertices can be easily identified by checking the numbers of their 1-ring trianges and 1-ring edges. i.e if the two numbers of a vertex are not equal, the vertex is a boundary vertex. Once a boundary vertex is identified, the set of boundary edges are traced from it. If all the boundary edges form a closed loop, they make up a hole. This is the way to identify holes.

3.5.2 Incorporating MeshFix library

Once holes were identified, the hole filling process was carried out. And subsequently the MeshFix library was used to analyze the outcome. The result gave us a simplified version of the original mesh but still the challenge was with the computation of normals.

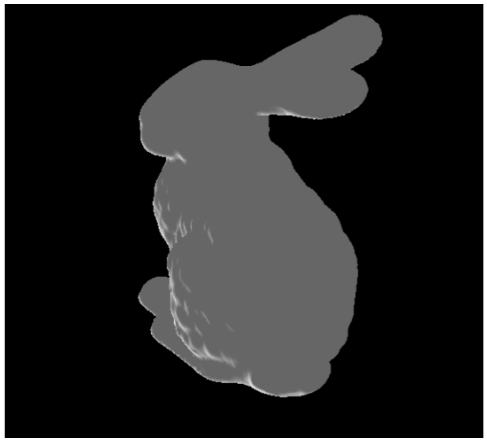

Figure 10: *Simplified mesh.*

3.6 Retriangulation

In this approach, we revamped the triangulation process by making some changes to it. And the approach is to replace old vertex indices in old faces using the new generated vertex (i.e. the representative vertex). The implementation is as follows:

Old Faces: 1,3,4 and 2,5,6

Cell: Vertices: 1,2,3 Index of representative vertex: 10

New Faces: 10,10,4 and 10,5,6

As per the approach, before clustering we had faces with vertices 1,3,4 and 2,5,6 and after creating a cluster based on a user specified cell size, we get the index of representative vertex as 10. Hence all the vertices falling within that cell are replaced with the representative index i.e. 10. So, after this implementation the new faces for the old faces (1,3,4 and 2,5,6) become 10,10,4(which is in this case is an edge and it needs to be deleted from the simplified model)and 10,5,6. And to our surprise the implementation worked as expected.

3.7 Current Results

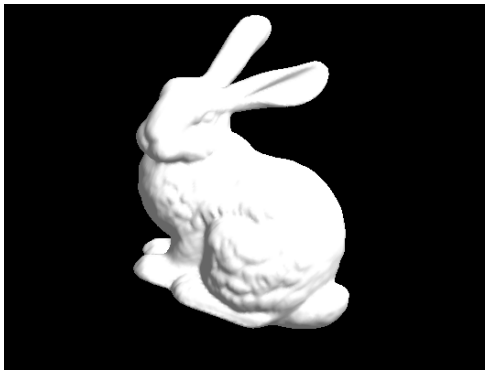

Figure 11: *Original Bunny with number of Faces : 69666 and Vertices :34835.*

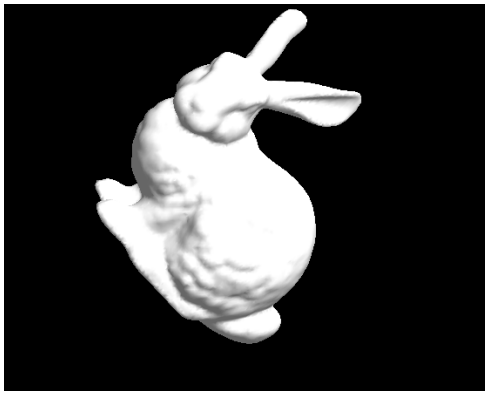

Figure 12: *Simplified Bunny with number of Faces : 37125 and Vertices :18542.*

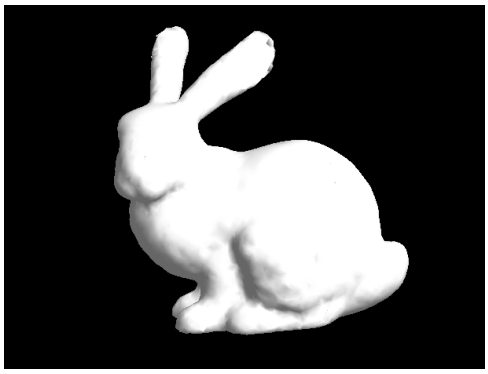

Figure 13: *Simplified Bunny with number of Faces : 11167 and Vertices :5548.*

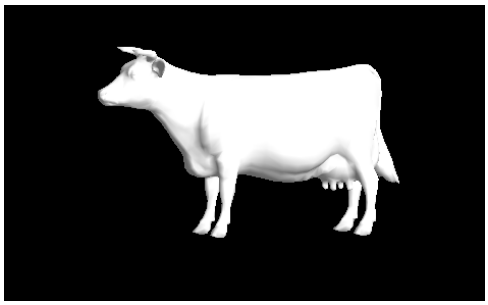

Figure 14: *Original Cow with number of Faces : 5804 and Vertices :2903.*

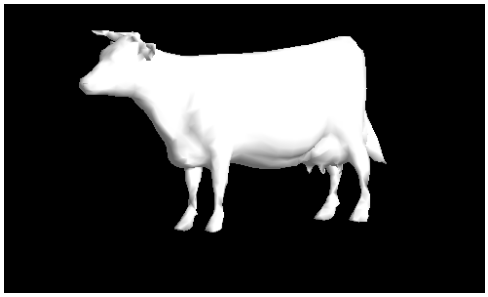

Figure 15: *Simplified Cow with number of Faces : 3359 and Vertices :1660.*

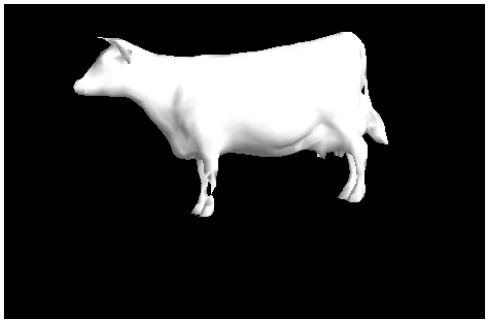

Figure 16: *Simplified Cow with number of Faces : 2144 and Vertices :1058.*

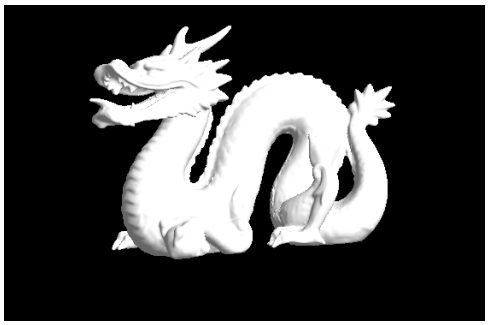

Figure 17: *Original Dragon with number of Faces : 100000 and Vertices :50000.*

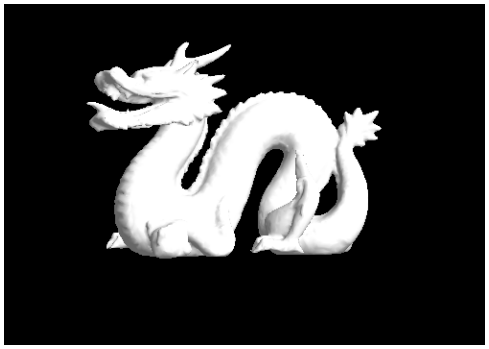

Figure 18: *Simplified Dragon with number of Faces : 33806 and Vertices :16600.*

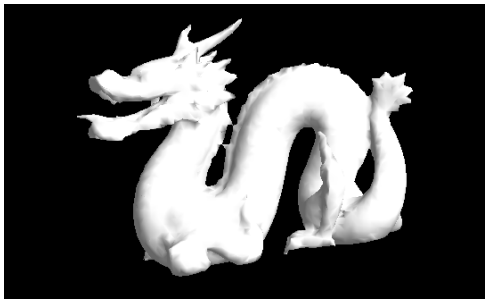

Figure 19: *Simplified Dragon with number of Faces : 10957 and Vertices :5304.*

4 Discussion

The approaches in this project are promising in the sense that we got our expected output though they do not include any significantly complex calculations.

The overall process was challenging in terms of testing the intermediary steps to check the success and filling the holes.

We learnt some good approaches/algorithms and data structure implementations. We came across many methodologies for reducing a mesh size and now we know the back end processes that are carried out in order to compress a mesh.

If we are given a chance to start the project again, we will probably use a more efficient data structure. Other than this, we have the desired result and the data structure is still efficient.

5 Conclusion

Normally triangular meshes are rendered using sophisticated hardware and with the help of accelerator. We have presented a technique, which computes one or several simplified meshes from an original mesh to expedite the process of loading a mesh online with better optimization of bandwidth. These methods produce approximate version of the original mesh with fewer primitives and at the same time preserving the overall visual information of the model. Using these methodologies we could have several simplified versions of the original mesh, which can be stored in addition to the original model.

6 References

- Model Simplification Using Vertex-Clustering by Kok-Lim Low and Tiow-Seng Tan
- Multi-resolution 3D approximations for rendering complex scenes by J Rossignac, P Borrel
- Decimation of triangle meshes by WJ Schroeder, JA Zarge, WE Lorensen
- Mesh Fix :https://code.google.com/p/meshfix/