Decimation - Lab 2 TNM079

Jonathan Bosson
jonbo665@student.liu.se

Friday 29th July, 2016

Abstract

The paper discusses how decimation of a mesh works and the improvements as well as difficulties that comes with removing data from a mesh. It takes up an evaluation technique proposed by Garland and Heckbert which uses quadric matrices to determine which edges in the mesh that can be collapsed and removing minimal detail from the model.

1 Introduction

Decimation is about collapsing edges from your mesh to reduce its resolution. It is done by removing the edges connecting two vertices and replacing them with a vertex point. This will let the mesh data structure take less storage and allow for faster load times, but it can lose a lot of the models detail. To know which edges are good to remove, and where to put the new vertex point, without altering the overall shape of the mesh there is a cost evauluation introduced by Garland and Heckbert which is based on quadric error matrices.

2 Background

Garland and Heckbert introduces a general algorithm to decimate edge and non-edge contractions in [2] through quadric error matrics. However, this paper will only describe the edge-cases in order to keep implementation simple. Using this algorithm can all edges in the mesh be evaluated in which magnitude each destroy the topology of the model and thus can the best edges be collapsed in a decimation.

The first step for this evaluation is to find the quadrics of each vertex point in the mesh. The vertex quadric is a sum of all the faces' quadric in its 1-ring. The face quadric is the distance of a vertex from a plane, and thus calculated by

$$K = \begin{pmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{pmatrix} \tag{1}$$

where (a, b, c) is the normal vector, $-d = ax + by + cz$ and (x, y, z) is the position of the face. Since an edge is attached to two vertices are both of them summed up to an edge quadric Q.

The error estimate of collapsing an edge can then be calculated by

$$\Delta(\vec{v}) = \vec{v}^t Q \vec{v} \tag{2}$$

where $\vec{v}$ is the position of the new vertex that will replace the edge. Three equations are given by deriving $\Delta(\vec{v})$ by x, y and z. After further deduction we can see that the result is the same as the original Q but its fourth row replaced by the zero vector $\vec{0} = (0, 0, 0, 1)$.

Now $Q'\vec{v}$ is a quantification of the error in collapsing current edge and replacing it with vertex $\vec{v}$. Equation 4 is used in order to find the best possible position to place the new vertex point $\vec{v}$.

$$Q'\vec{v} = \vec{0} \tag{3}$$

Since Q' is a squarematrix will it be invertible if its determinant is nonzero. This is helpful since if inverting Q' is possible we can find the best possition of $\vec{v}$ by

$$\vec{v} = Q'^{-1}Q'\vec{v} = Q'^{-1}\vec{0} \tag{4}$$

If the matrix is singular and an inverted Q' cannot be found can a binary search for the best position be done. In the implementation of this lab however was only three positions considered. Each of the initial vertex position, v_1 or v_2, of the vertices attached to the edge at hand as well as the position in between them, $(v_1 + v_2)/2$.

Note that Q' is only used to calculate the position for the lowest cost whereas Q is used for the cost itself. Once the best position for the new vertex has been found can equation 5 be used to calculate the cost.

$$\vec{v}^t Q \vec{v} \tag{5}$$

Once the calculations are done for all edges in the mesh will they be sorted through a heap data structure and the decimation function can then efficiently collapse all lowest cost edges until the new resolution has been achieved.

3 Results

In this lab were three functions implemented; *createQuadricForVert*, *createQuadricForFace* as well as *computeCollapse*.

createQuadricForVert will, as its name suggest, create the quadric of each vertex that is needed for the cost calculation. This function will step through the 1-ring and call on *createQuadricForFace* for each face. The face-function will return a quadric like 1 which is then summed up with all other face quadrics in the 1-ring for the vertex quadric Q.

computeCollapse recieves an edge and collects the two vertex quadrics it is connected to. Each edge also has two variables called *cost* and *position*. The quadrics are summed up to create the quadric Q for the calculations. A copy of Q is created where the last row is replaced to the zero vector $\vec{0} = (0, 0, 0, 1)$.

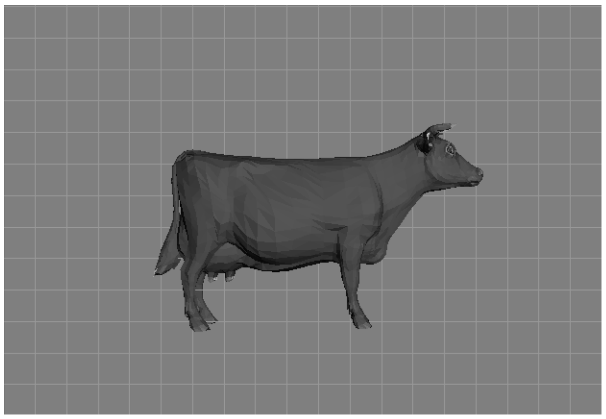

Figure 1: A cow model in its original resolution

If this new quadric Q' is not singular can the matrix be inversed, and thus the *position* can be calculated through 4. If the quadric is singular will instead an evaluation of which position is the best between the position of each vertex as well as the mid point in-between them. The *cost* is lastly calculated through 5.

Figure 2 wears a strong resemblance with the original cow model 1 even though the resolution is massively reduced. Figure 5 has 25% of the triangles in 2 but it can be seen that further reduction destroys the topology due to the complexity of the model.

3.1 Reference list

References

[1] M. E. Dieckmann, *Lecture Slides for TNM079*, **Lecture 3**, 2016.

[2] Michael Garland and Paul S. Heckbert, *Surface Simplification Using Quadric Error Metrics*

This report aims for grade 3.

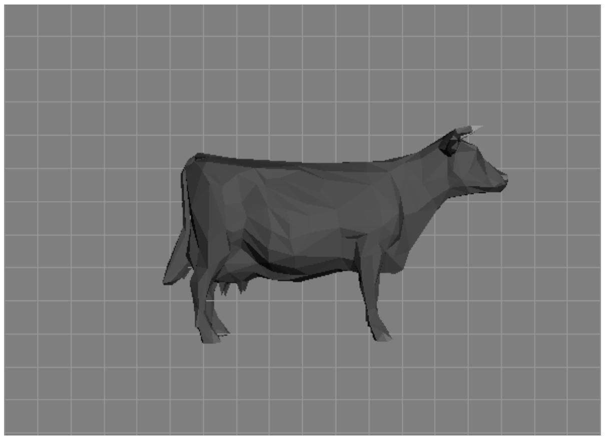

Figure 2: A cow model decimated to 994 triangles

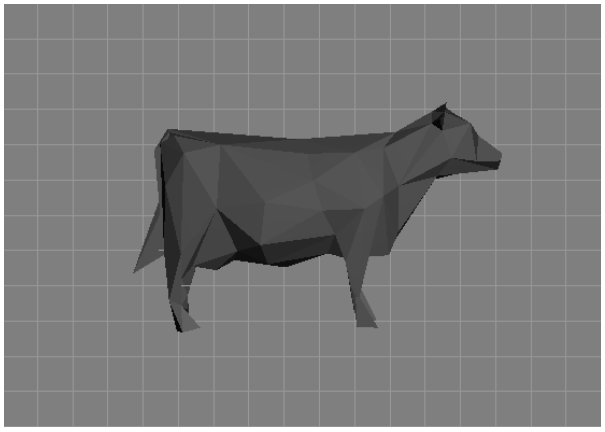

Figure 4: A cow model decimated to 248 triangles

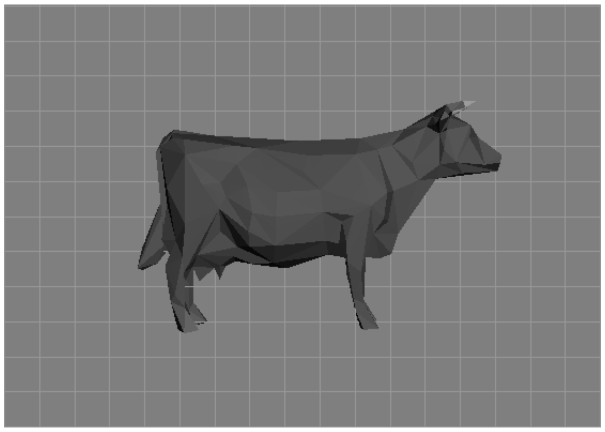

Figure 3: A cow model decimated to 532 triangles

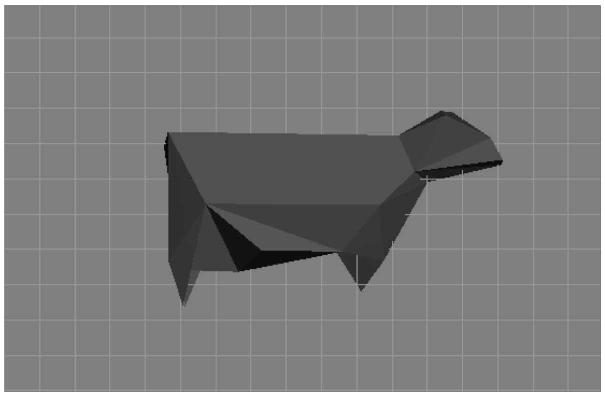

Figure 5: A cow model decimated to 64 triangles