Surface Representation and Geometric Modeling

Exercise 3 - Mesh Decimation

Handout date: 10/25/2010

Submission deadline: 11/8/2010, Midnight

Note

Copying of code (either from other students or from external sources) is strictly prohibited! Any violation of this rule will lead to expulsion from the course.

What to submit

A .zip compressed file renamed to "Exercise*n*-YourName.zip" where *n* is the number of the current exercise sheet. It should contain:

- A Microsoft Visual Studio 2008 file solution with all source files and project files. In order to avoid sending build files, close your Visual Studio solution and run the file "cleanSolution.bat" located in the top directory of the framework before you submit.
- A "readme.txt" file containing a description on how you solved each exercise (use the same numbers and titles) and any problems encountered.
- Send your solutions to mirela@stanford.edu before the submission deadline.
  No late submissions will be accepted.

Reference Software

See 03-Decimation.exe for an example of how your program should run.

Mesh Decimation

In this exercise you will implement surface simplification using quadric error metrics. The tasks you will implement to complete the framework to a functional mesh decimation algorithm are as follows:

- Computing initial quadrics for vertices

- Testing edge collapse for triangle flips
- Computing the collapse priority as the sum of two quadrics
- Implementing the global decimation loop: get best halfedge, collapse, update neighborhood

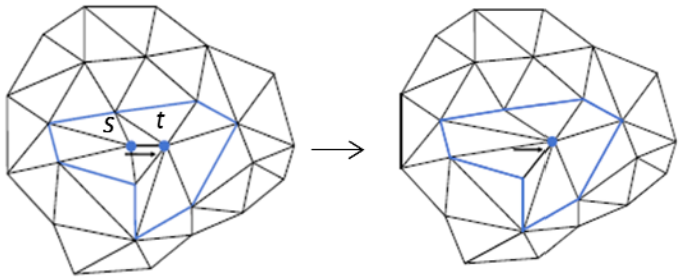
Framework

A new project, Decimation has been added to the framework from the previous exercise. It contains only two files and it compiles to a command line tool that reads a mesh and outputs a decimated version of it. The program takes three parameters: the ratio between the initial vertex count and the final vertex count, the input mesh file (.off) and the output mesh file (.off).

4.0 Preparation

The decimation algorithm uses the quadric error measure to collapse halfedges. It operates along the following principles:

- A pair of vertices may be collapsed to a single vertex *only* if they are connected by an edge and do not result in a triangle flip.
- A halfedge $h = s \rightarrow t$ is collapsed to t:



- A halfedge $h = s \rightarrow t$ may be assigned a priority $pr(h)$ based on the quadric error. Thus we may assign to every vertex s a priority equal to that of the halfedge emanating from s having minimum priority:

$$pr(s) = \min_{h=s\rightarrow t} pr(h)$$

- The collapse *target* of s is $targ(s)$ = the other endpoint of this halfedge h . Hence the priority queue may store vertices instead of halfedges. Each vertex in the queue has the properties: priority $pr(s)$ and target $targ(s)$.
- In the main loop of the algorithm we pop the vertex s having minimum priority from the queue. Then the halfedge $s \rightarrow targ(s)$ is collapsed such that s coincides with $t = targ(s)$. The properties of t and other affected vertices are updated on the queue.

4.1 Quadric from triangle

Complete the init() function in deci.cc so that it calculates vertex quadrics from the quadrics of the incident triangles. Remember, the quadric associated with a vertex p is the sum of the quadrics of the triangles, i.e. the sum of the squared distances to planes q_i of the triangles incident on p:

$$\sum_i dist(q_i,p)^2 = \sum_i p^T Q_i p = p^T \left(\sum_i Q_i \right) p \equiv p^T Q p$$

where $p = (x, y, z, 1)^T$ are the homogeneous coordinates of the vertex, $q_i = (a_i, b_i, c_i, d_i)^T$ the unit length vector containing the coefficients of q_i's plane equation: $a_i x + b_i y + c_i z + d_i = 0$, and the matrix Q_i is:

$$Q_i = q_i \, q_i^T$$

Use the Quadricd class.

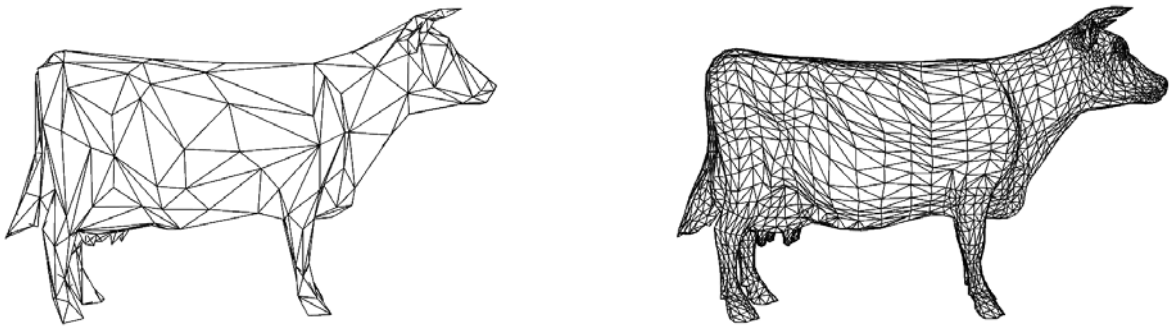


Figure 1: Simplification of the cow mesh to 10% of the vertex count

4.2 Collapse testing with normals

Implement the is_collapse_legal() function to test if the collapse would lead to triangle flipping. To do this, examine every triangle adjacent to the vertex which changes during the collapse. For every triangle compare the original normal with the normal which would result after the collapse. If the angle between these normals is larger than π/4, the collapse is illegal.

Hint: Make sure the mesh is unchanged after you return from this function.

4.3 Priority of a halfedge

Compute the priority of a halfedge using the sum of its two end-vertex quadrics and return it in the function priority().

4.4 Simplification main loop

Implement the main loop of the decimate() function. In every iteration there are three steps:

- Remove the vertex with the lowest priority from the queue by calling the begin() function.
- In case the corresponding collapse is legal - collapse the halfedge corresponding to this vertex.
- Update the properties of the vertices on the queue whose adjacent triangles have been changed using the enqueue_vertex() function.

Your final result should look like Figure 1 for the simplification to 10% of the vertex count.

Hint: Note that the enqueue_vertex() functions works for both vertices which are already in the queue and completely new ones.)