Rensselaer Polytechnic Institute

Advanced Computer Graphics, Spring 2014

Final Project

Evaluating the Quality of Triangle, Quadrilateral, and Hybrid Meshes Before and After Refinement

Author:
Rebecca Nordhauser

Professor and Advisor:
Dr. Barb Cutler

May 6, 2014

Contents

1 Motivation

When dealing with meshes it is important to have elements that not only properly represent the desired geometry, but also have properties that make them easy to work with. For example, elongated, abnormal elements would not work well in a radiosity simulation because areas that are far apart are being labeled with a single light value. If local changes were made to a mesh by moving a vertex that was in an irregular face, the change might affect areas of the mesh not near the original change. For this reason, these faces could cause a problem in animation. Because of situations like this, it is important to be alerted to these faces and be able to effectively improve them.

In this paper, I discuss methods of measuring the quality of an element. I also present visualizations of the mesh to show the quality of the faces before and after my refinement technique has been applied to the mesh.

2 Related Work

Knupp describes some desirable aspects in metrics used to evaluate mesh quality in Algebraic Mesh Quality Metrics. Some such properties are: not being dependent on size, not having units, having an 'ideal' to compare to, and being useful for many types of elements [Knupp, 2001]. In Spectral Surface Quadrangulation, Dong et al. describe how to use properties of a mesh's surface to create a high quality quadrangulation of the surface and calculate ideal locations for extraordinary vertices [Dong et al., 2006].

3 Mesh Quality

I have assigned each mesh a single overall quality value, ranked from 0 (low quality) to 1 (high quality). This number is an average of the quality values calculated for each face, which also range from 0 to 1.

3.1 Calculating Mesh Quality

One metric I choose to use is the ratio of the area of an element to the area of its inscribed circle. This metric meets many of the criteria discussed by [Knupp, 2001]. Any polygon, even ones with more sides than a quadrilateral, will have a bounding circle with a larger area, and the ratio is unitless.

Another technique I used to evaluate the quality of an element is the size of the smallest angle in an element compared to the angles in a regular polygon with that many sides. Without the use of this metric, quads that were intentionally made long and thin to match object geometry would get an extremely low quality score.

3.1.1 Calculating Area to Bounding Circle Area Ratio

To calculate the area of an arbitrary triangle, I simply used the formula $\frac{1}{2} * \|AB \times BC\|$, where A, B, and C refer to the vertices of the triangle. I calculated the ideal ratio of a triangle using the formula: Area of $\triangle AOB = \frac{1}{2} * r * r * \sin 120°$ and Area of $\triangle ABC = 3*$ Area of $\triangle AOB$. This ratio is $\frac{3\sqrt{3}}{4\pi}$

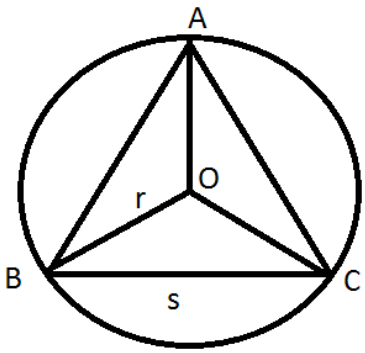

Figure 1: Above is an equilateral triangle in its bounding circle.

While calculating the area of a quad, I assumed that it was convex. To calculate the area of a concave quad would not be too difficult, but my algorithms later also assume convex quads. To calculate the area of the quad, I divided the quad up into two triangles, triangle ABC and triangle ACD, and then added their two areas together. This calculation assumes that the quad is planar, which is not necessarily true, but I use this approximate area value. For quads, I used the Figure 2 to determine the ideal area to inscribed circle area is $\frac{2}{\pi}$.

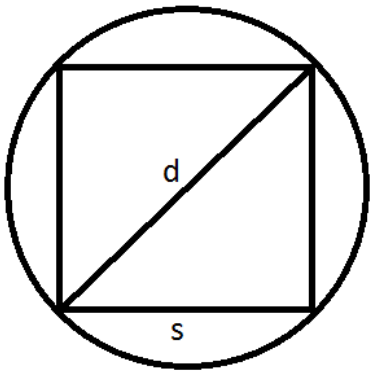

Figure 2: Above is a square in its bounding circle.

3.1.2 Calculating Smallest Angle

Sometimes the geometry of the mesh is not symmetric, it can be beneficial to have elements, especially quads, that are not a regular shape. Longer and thinner elements can sometimes better represent geometry with fewer edges, for example the lamp in Figure 3.

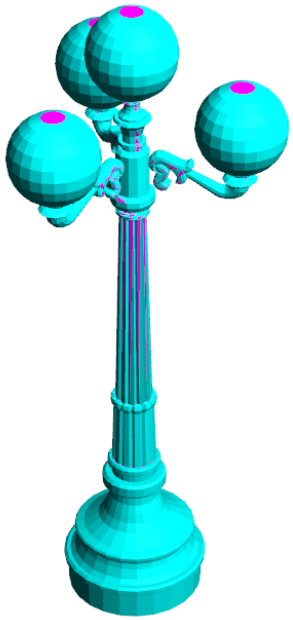

Figure 3: This mesh of a lamp has elements on the pole that are intensionally long and thin. There can be fewer elements because the vertical facets each only need to be represented with one face. In this visualization, quads are cyan and triangles are magenta. [Burkardt, 2012]

For this reason, another way I am measuring the quality of an element is the ratio of the smallest angle to the angles in a regular shape with that number of sides. The smallest angle ratio accounts for half of the element's quality value.

3.2 Quality Visualization

To show mesh quality, I used a color visualization. I chose to make faces that have quality values close to 1 a green color and faces that have low quality values have colors closer to red. An even fade from green to red, representing the quality is given by:

glm::vec3(1-quality, quality, 0.0);

Figure 4 is an example of the quality visualization.

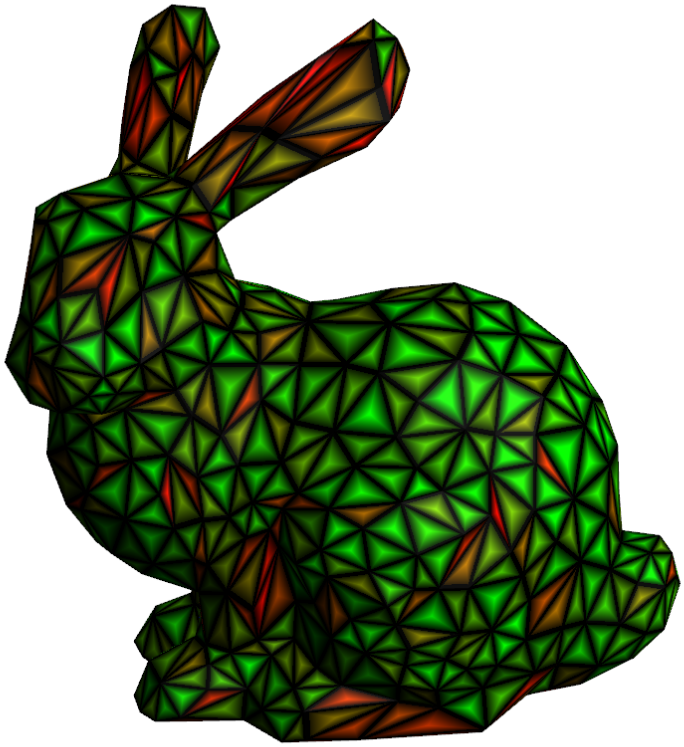

Figure 4: This bunny mesh is colored using the quality visualization. It can be seen that more irregular elements are more red and elements that are closer to equilateral are green.

4 Mesh Refinement

I implemented an algorithm to attempt to improve the quality of elements in a mesh, which creates more regular elements by making local changes to adjacent faces. The three methods I use for improvement are 'diagonal swapping', 'triangle merging', and 'triangle-quad recombination'. The algorithm is described below.

```
for each triangle (ABC) in the mesh
  for each adjacent triangle (BDC)
    if angle between normals < threshold
      if angle between normals of triangles ADC and ABD is less than threshold
        if the sum of the quality of new triangles is higher the old triangles
        and triangle merging
          remove triangles ABC and BDC
          insert triangles ADC and ABD
      if angle between quad ABCD and original triangles < threshold
        if new quad quality is better than old triangles and diagonal swapping
        triangles
        if new quad is convex
          remove triangles ABC and BDC
          insert quad ABCD
  for each adjacent quad
    test quality for splitting the quad into triangles both ways
    test combining the quad's triangle touching the original triangle with
    the original triangle
    if created quad is concave, disqualify this configuration
    make best change if any are better than the original quality
```

The implementation of this algorithm is in C++ using OpenGL. It is uses the framework built by Prof. Cutler for homework 1.[Cutler, 2014]

4.1 Diagonal Swapping

The algorithm using diagonal swapping improves the quality of pairs of triangles. Their normals are compared to make sure that they are within a threshold of each other as to not change the shape of the mesh significantly. Each pair of triangles that share an edge are compared to the triangles that would be formed if the edge they shared was changed to be the other diagonal of the quad they form together. The new triangles must also have normals below a threshold. This means that there is a maximum allowed dihedral angle for the newly created triangles. Figure 5 is an example of this change.

Figure 5: In this case, it improves the quality of the triangles to modify them, as shown

4.2 Triangle Merging

The triangle merging algorithm also works with pairs of triangles, so before selecting whether to do triangle merging or diagonal swapping, it must compare the resulting qualities of each on a per-triangle-pair basis. In the triangle merging algorithm, I compare the average quality of the triangle faces to the quality of the quad that would be formed by mering them into a single face. If the new quality value is higher and the resulting quad is convex, I remove the triangles and add the quad. To make sure the quad is convex, I look at each vertex that the two original triangles share. For both, I add together the angle that is centered at that vertex in each of the two triangles. If these angles are less than 180, the quad is convex.

4.3 Triangle-Quad Recombination

The Triangle-Quad Recombination refinement technique compares many options of change. First it divides the quad up into two triangles, and tests the quality of cutting the quad along each of the two diagonals. Next, in each of these cases the algorithm tries combining the triangle from the quad that is touching the original triangle with that triangle. Before a change is made, all five cases are considered. If any of these have a higher average element quality value than that of the original two faces, the best is put into the mesh. Figure 6 shows an example of triangle-quad recombination

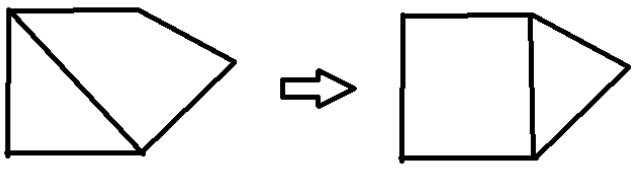

Figure 6: In this case of recombination, it was most beneficial to split the quad into two triangles and then combine one of those triangles with the original triangle to make a quad. The resulting mesh is on the right.

4.4 Visualizing the Change

To visualize the changes made, I turn the affected faces different colors immediately after the refinement is done. I color the 'diagonal swapped' faces magenta, the 'triangle merging' quads dark purple, and the 'triangle-quad recombination' quads and triangles yellow, as can be seen in Figure 7.

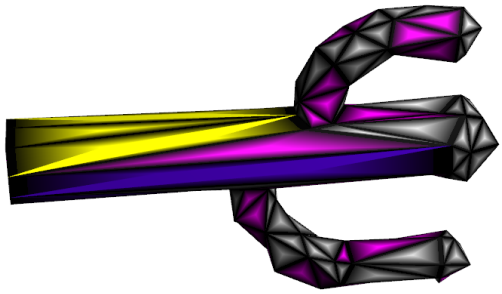

Figure 7: All three refinement operations have been preformed on this mesh, as can be seen by the magenta, purple, and yellow faces.

5 Limitations

While doing this project, I had several limitations in order to complete the project within the time frame, many of these limitations included not handling corner cases in meshes. I require that all mesh elements are either quads or triangles (or will be converted to quads and triangles when the mesh is loaded into the system. All quads must also be convex. If they are not, their areas will not be calculated correctly, because I calculate the area of a quad by adding the areas of the two triangles that make up the quad. I also require that each pair of adjacent faces share at most two vertices. This means that each pair of faces can only share one edge, which is important for the diagonal swapping algorithm.

Another limitation of my algorithm is the way I handle 'quality improvement' of the mesh when deciding whether or not to change faces. Both Triangle Merging and Triangle-Quad Recombination can cause a change in the number of faces, but I still judge the quality of the faces I am looking at by averaging them. For example, if two triangles of high quality compared to the other faces were combined, the overall quality of the mesh could go down because these triangles now are only averaged into the mesh quality as one face.

6 Challenges

6.1 Finding Meshes

One challenge that I had was finding meshes to use to test my algorithm. I originally had downloaded some meshes, but later noticed that they had concave elements and elements that shared more than two vertices. Figure 8 (left) shows an example of a concave quad in a lamp model that I found. Another mesh I found, pictured in Figure 8 (right), is of a magnolia had most elements oriented in the same direction, but one element facing the wrong way. An element is facing the wrong way if their vertices are listed in the wrong order (clockwise versus

counterclockwise). It is important to know which side of the face is the 'front' side so the normal of the face can be determined.

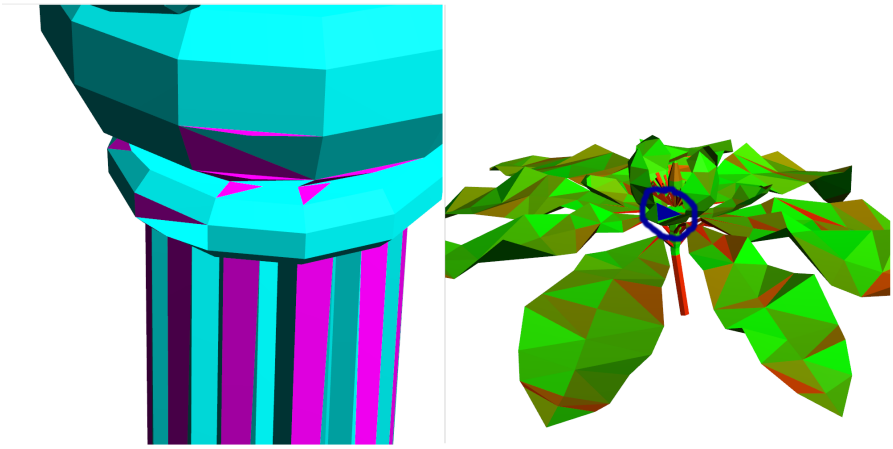

Figure 8: The closeup on the left shows some concave quad faces that do not work with my algorithm. The right image shows a mesh with one face in the opposite direction from the others.[Burkardt, 2012]

I also found some meshes that seemed to have faces either with the incorrect vertices or formatted differently. Because of my difficulties with finding meshes, I only tested on triangle meshes. Figure 9 shows a few of the meshes I decided not to use.

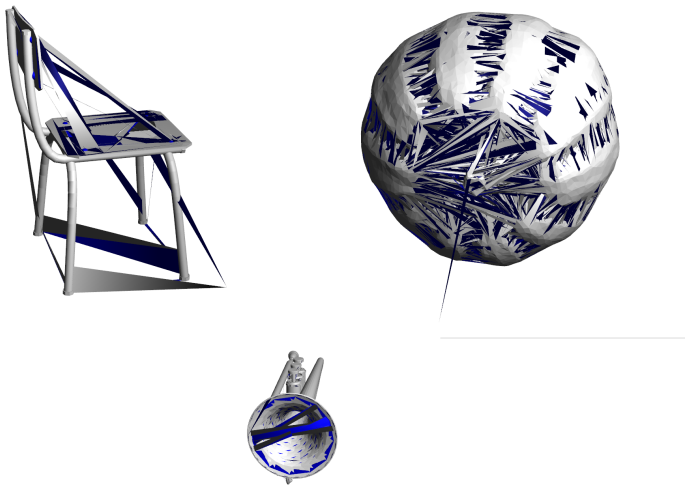

Figure 9: Some of these faces have vertices that were clearly not meant to be connected by a face or are in a format that I cannot parse.[Burkardt, 2012] [Durand, 2003]

6.2 Wireframe

I found adding the option to have a wireframe and colored elements to be a challenge. In the homework 1 code base, once a variable face color was introduced, the same method for calculating where the wireframe edge should be could not be used.

I attempted to pass more information into the vertex shader, but when shading, I always found the 'center color' variable to contain the color black. Currently the wireframe is a thick black line that then fades into the face color. It is functional enough to be useful, but should look nicer. This is an example of what the wireframe looked like before.

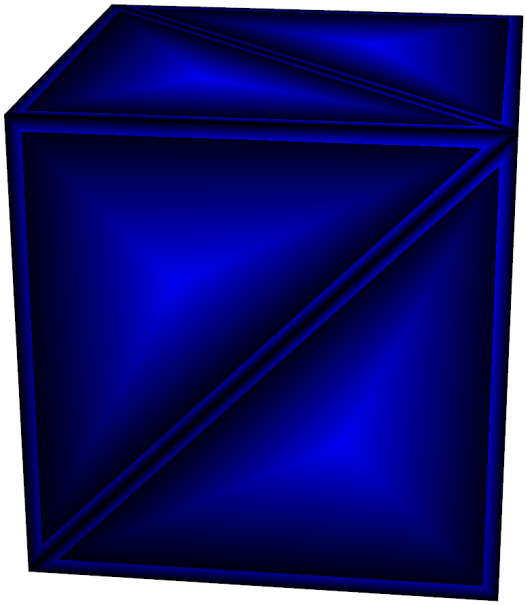

Figure 10: Wireframe appearance at early stage of development.

6.3 Vertex Ordering

When putting new faces into the mesh in the diagonal swapping algorithm, I had a hard time getting the vertices to consistently be ordered correctly. Sometimes having them in reverse order caused some of the new triangles to be oriented in the wrong direction (facing in). Figure 11 is an example of this bug. Having these triangles turn blue highlights to the user some of the faces that were changed and helped me make sure that my algorithm was correct. This inspired me to change the color of refined faces, as described in section 4.4.

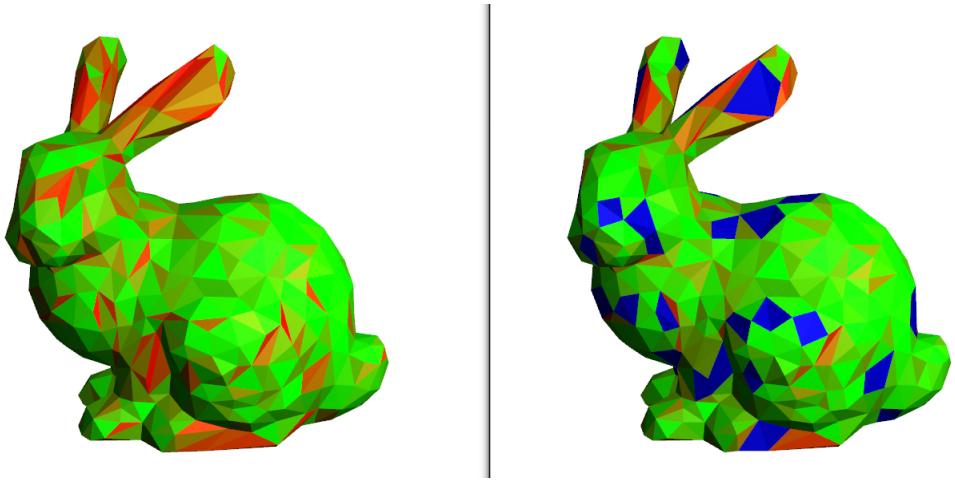

Figure 11: On the left is a bunny with the quality visualization, and on the right is an example of the vertex ordering bug. With these two meshes side by side, some of the triangles that were modified with diagonal swapping can easily be located.

7 Results and Conclusion

I tested my algorithms on 17 different meshes [Burkardt, 2012] [Durand, 2003] [Cutler, 2014] with initial quality values ranging from 0.397 to 0.721. Since I was only able to find triangle meshes that complied with my requirements, I compared only my diagonal swapping method and my triangle merging refinements. When testing on different angle thresholds I used all three methods, but triangle-quad recombination only happened once quads were created with triangle merging. Because of this restriction, the triangle-quad recombination on its own would never modify any of the meshes.

7.1 Angle Threshold

As expected, as the threshold for the allowed angle between shapes increased, so did the number of refinements and the quality improvement. As can be seen in Figure 12 the amount of refinement increased until about 60°, where the threshold angle was no longer a limitation which operations could be preformed. Even though there was more element quality improvement with higher thresholds, I started to notice small changes in mesh shape at about 30° with larger shape changes at higher threshold angles. Figure 13 shows an example of a mesh that was refined with too high of a threshold angle.

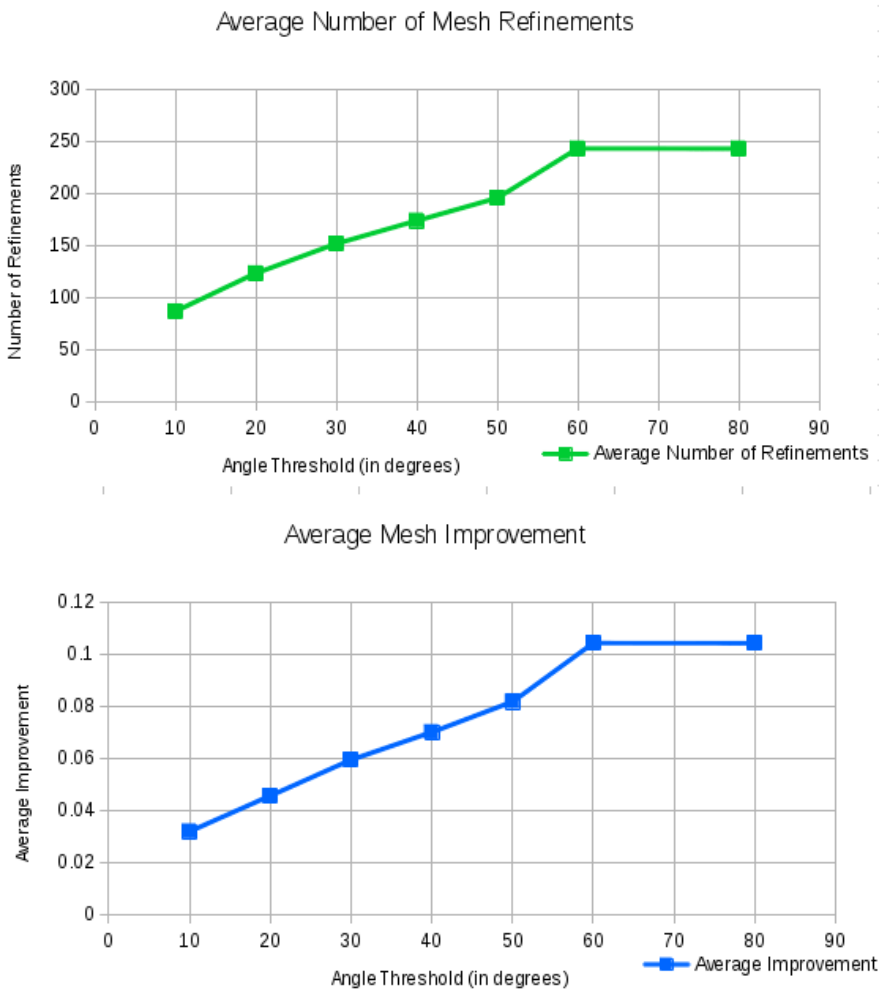

Figure 12: Plotted points are the average number of refinements done and average improvement when run on the 17 meshes I used with the angle threshold given.

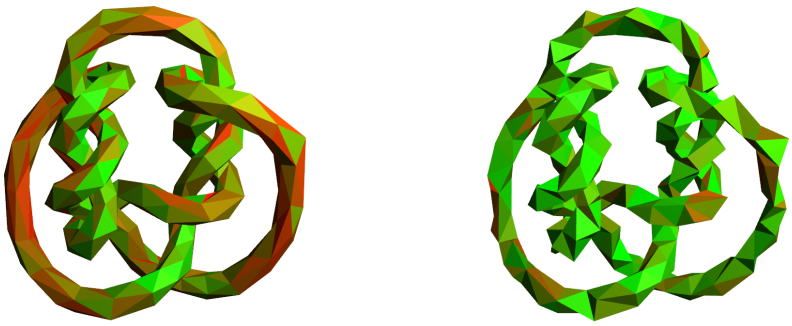

Figure 13: On the left is a mesh before refinement and on the right is the same mesh after it has been refined with a threshold angle of 60°.

7.2 Comparison of Diagonal Swapping and Triangle Merging

To compare these techniques, I limited my algorithm to only allow one type of refinement when running on each mesh. I found the diagonal swapping improved the mesh by almost three times as much with an average of 0.03611 compared to the 0.01209 improvement that triangle merging did. There was even more of a difference in the number of operations performed: an average of 106.5 for diagonal swapping and 15.53 for triangle merging. This means that each individual triangle merging operation was over twice as effective in improving quality than diagonal swapping.

8 Future Work

8.1 Wireframe

For this project, the implementation of the wireframe I used served its purpose. For the future, I would like to make it so that the wireframe is black and the inner portion of the face is a solid color, as opposed to a fade from black to the color the face is supposed to be. This would also make the wireframe look better for quads.

8.2 Use Dependent Calculation of Element Quality

Because different aspects of an element are useful for different applications, it could be advantageous to choose which metrics used to evaluate a mesh based on what the mesh will be used for. For example, if a mesh will be used for an expensive task, it may be important to represent the geometry with as few faces as possible while retaining geometry. If it is going to be used for a task like radiosity, it may be more important to have no long, thin elements.

8.3 Selection of Angle Threshold

In this paper I chose what angle threshold to used based off of running my algorithm on many meshes at various thresholds. I noticed very little mesh deformation with a 20 degree angle as the threshold, but with higher thresholds, there were obvious shape changes in the mesh. In the future, I would like to come up with a more rigorous method of choosing what threshold to use. This could be done by doing a user study and asking users when they start to notice change in the meshes or by making sure that after the change the mesh does not exceed some value that can be calculated.

References

[Burkardt, 2012] Burkardt, John(2012). OBJ Files. *people.sc.fsu.edu/ jburkhardt/data/obj/obj.html*.

[Cutler, 2014] Cutler, Barb. (2014). Homework 1: Simplification and Subdivision Surfaces. *www.cs.rpi.edu/ cutler/classes/advancedgraphics/S14/hw1_meshes.php* .

[Dong et al., 2006] Dong, Shen; Bremer, Peer-Timo; Garland, Michael; Pascucci, Valerio; and Hart, John C. (2006). Spectral Surface Quadrangulation. *SIGGRAPH*

[Durand, 2003] Durand, Fredo and Cutler, Barb (2003). Sample .obj Files *groups.csail.mit.edu/graphics/classes/6.837/F03/models*.

[Knupp, 2001] Knupp, M. Patrick. (2001). Algebraic Mesh Quality Metrics *SIAM. J Sci. Comput.*, 23:193–218.