Adaptive Subdivision of Triangular Meshes

CS 6491 Project 2: Phase 2

By: Anil Rohatgi
Gtg618b
Georgia Institute of Technology

Introduction:

The overall purpose of this project is to create a triangular mesh generator that can be used to create arbitrary geometry in a virtual environment. For phase two of this project, we were expected to implement regional subdivision of a triangular mesh. This will allow for more control over desired regions of the mesh where vertices are scarce, but avoid the expense of subdividing the entire mesh. Figure 1 below shows an example of an original mesh, and a subdivided version of the same mesh.

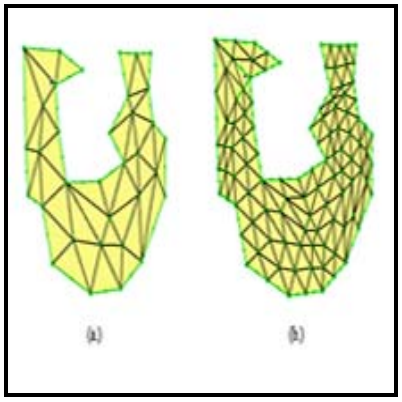

Figure 1. Subdivision example. (a.) is the original mesh. (b.) subdivided mesh

Subdivision Theory:

Each mesh is composed of numerous triangles and vertices. The control over the surface and shape of the mesh is based upon these control structures. In order to allow for more control over a mesh, more vertices and triangles must be introduced. This is the basic theory behind subdivision.

To do so, each triangle in the mesh must be divided into more triangles, adding more control points to the overall shape. However, this process can be computationally expensive, as the data structures used to describe a mesh become increasingly large and complex. It is for this reason that an adaptive method of subdivision needed to be generated in order to only subdivide regions where extra control is needed. An example of a locally subdivided mesh is shown as figure 2.

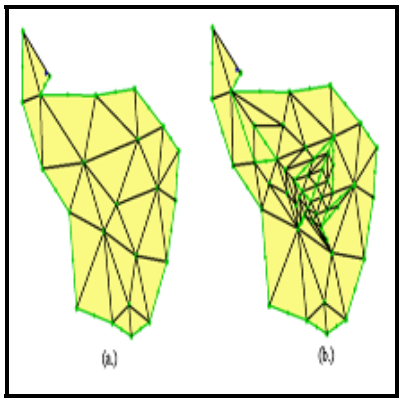

Figure 2. Locally subdivided mesh. (a.) is the original mesh, and (b.) is the locally subdivided mesh. Note only triangles in the center of the mesh have been split.

Algorithm:

The issue behind locally splitting triangles is, by simply inscribing a new triangle within the old ones creates

vertices that do not connect over the surface of the mesh between triangles that were split and those that were not. This leads to discontinuities and holes in the mesh that will eventually make it difficult to control. An example of this error is shown in figure 3.

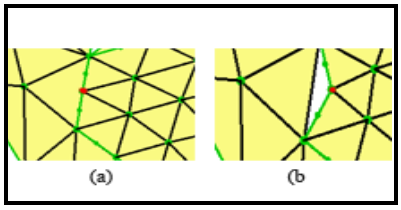

Figure 3. Vertex connection errors. Note the red vertex exists in one triangle and not the other. This leads to discontinuities as shown in (b)

To avoid this problem, two different schemes for the subdivision of triangles had to be implemented.

The first method is used when all three edges of a triangle are within the subdivision region. In this case, splitting the triangle into four new triangles by inscribing a new triangle within the original one is sufficient. Figure 4 graphically shows this procedure.

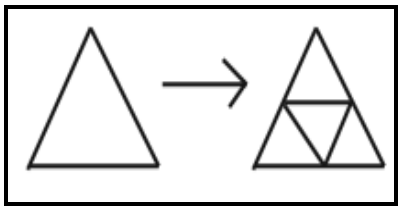

Figure 4. Splitting a triangle with two or more vertices within the subdivision region

However, when only one vertex lies within the subdivision region is where discontinuities occur. When this situation arises, the subdivision scheme used is shown in figure 5.

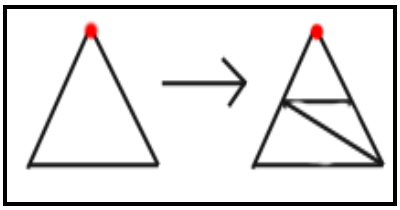

Figure 5. Subdivision with only one vertex within the subdivision region. The red vertex is the vertex within the region.

Note with this scheme, there is no vertex created along the edge that does not lie within the subdivision region. This avoids discontinuities in the mesh surface. When these techniques are implemented together, a mesh can be effectively subdivided in pieces.

The Program:

The subdivision program that was written works as follows. First, using the editor designed in phase one, a user can generate any arbitrary mesh. Next, to initiate subdivision mode, the “c” key is pressed and a red circle outlining the subdivision region appears over the cursor as shown in figure 6.

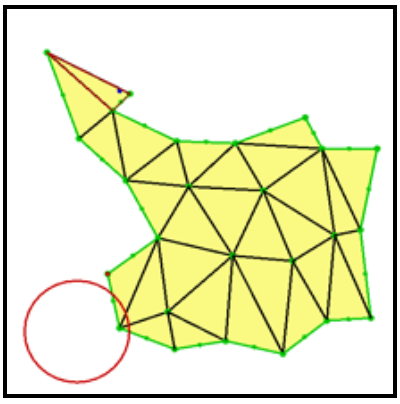

Figure 6. Subdivision mode with control circle

The circle can be maneuvered over the mesh to any location. Once the user has

decided upon where to subdivide the mesh, he left clicks the mouse. Once this occurs, the vertices and edges within the influence of the circle turn red, and the mesh subdivides the appropriate triangles. This is shown in figure 7.

The radius of influence can be altered to be any size depending on the application of the user. Using this program, the user can create detailed meshes, without excessively triangulating unnecessary regions of the mesh.

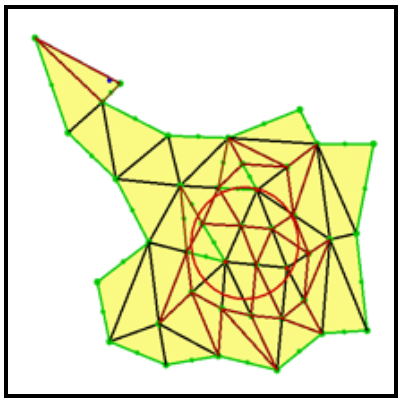

Figure 7. Post subdivision

Figures:

Included below are some figures demonstrating the capabilities of this program.

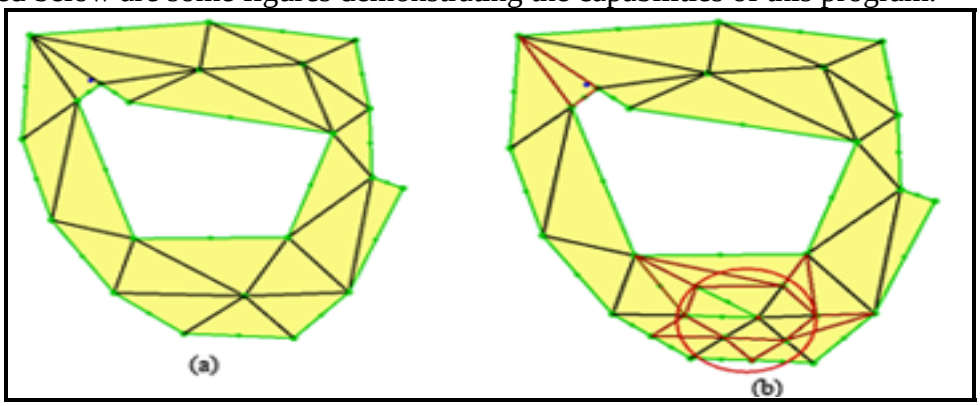

Figure 8. Ring object subdivided once in lower region

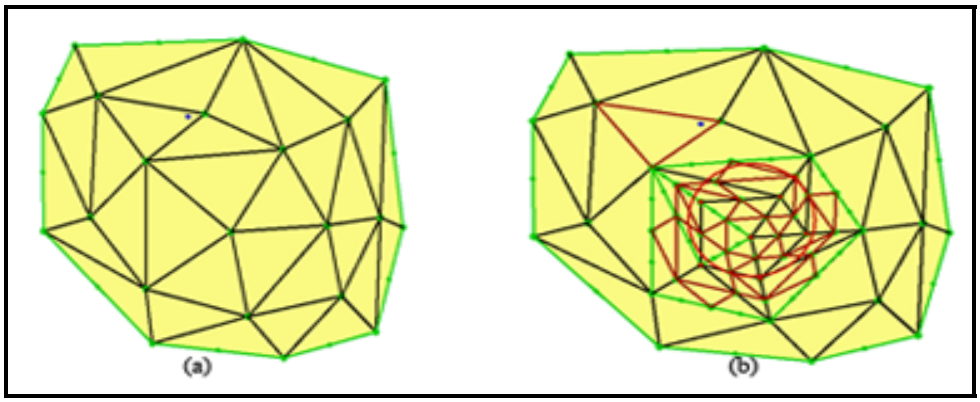

Figure 9. Closed object subdivided twice in center

Problems Encountered:

Numerous problems occurred while trying to get this local subdivision scheme to work.

The first difficult problem was coding the two different subdivision schemes for the triangles depending on how they were oriented in respect to the subdivision region. Although it seems simple in concept, maintaining the counter clockwise convention when adding new triangles to the mesh became a huge source for careless errors and painful debugging.

The second large problem became welding together new vertices created by the subdivision that were shared by multiple triangles. This was accomplished by writing a completely new function to search for vertices that had the same value, and referencing them to one another so they essentially behave as one unit.

The third large problem was in calculating the Otable for the subdivided mesh. In the previous program, this was done intelligently on a case by case basis, but once the subdivision occurred, all the previous information was invalid. Keeping track of individual corners on a case by case basis became overwhelmingly complicated, so a separate function was written to recalculate the entire Otable when a subdivision occurred.