HomeWork 4
Loop Subdivision

Fall 2012

Submitted By -

Nitin Agrahara Ravikumar

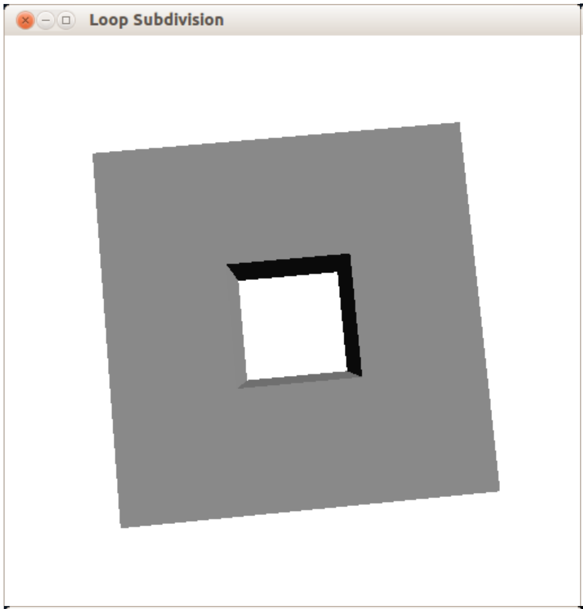
73714398

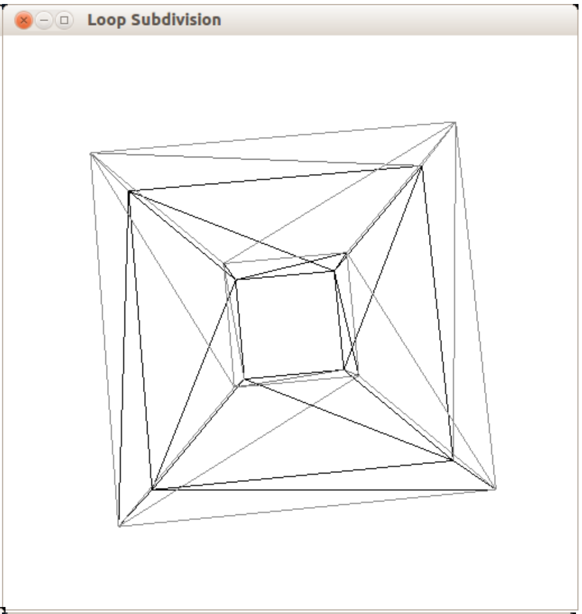
Structure of the Program

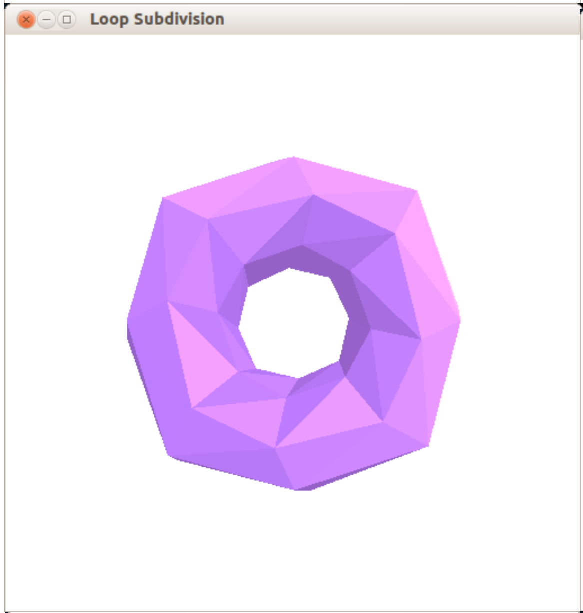
The loop subdivision assignment has been done using the Nate Robbins program employed for the second homework. The working of the program is in most ways similar to the method employed earlier. Once the program starts, the user has to select an object to load. The Torus object is preloaded. To move through the subdivision process the user has to press the **'y'** key. To view the triangles the user may use the **'w'** key to switch between wire frame and shaded object.

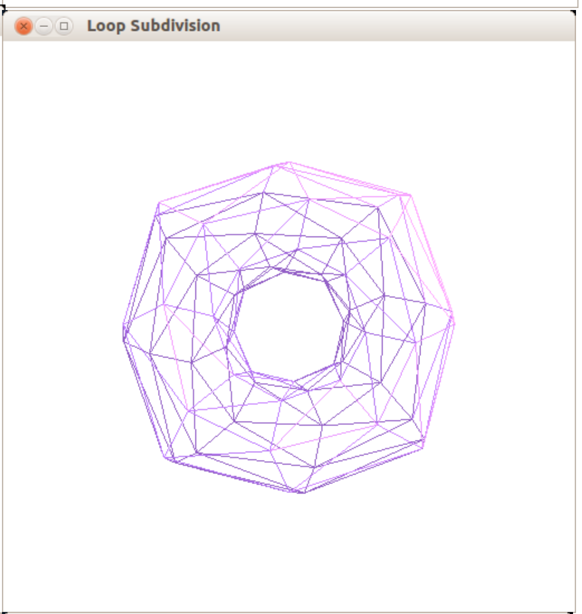
The data structure used for the implementation is an extension of the existing GLMTriangle data structure used by Nate Robbins. Instead of indexing and storing vertices in a efficient way so as to reduce space costs, this implementation uses a repeated vertices method in which each triangle points to its own list of vertices. This is computationally far more expensive than implementing an optimal data structure such as a winged edge of half edge data structure.

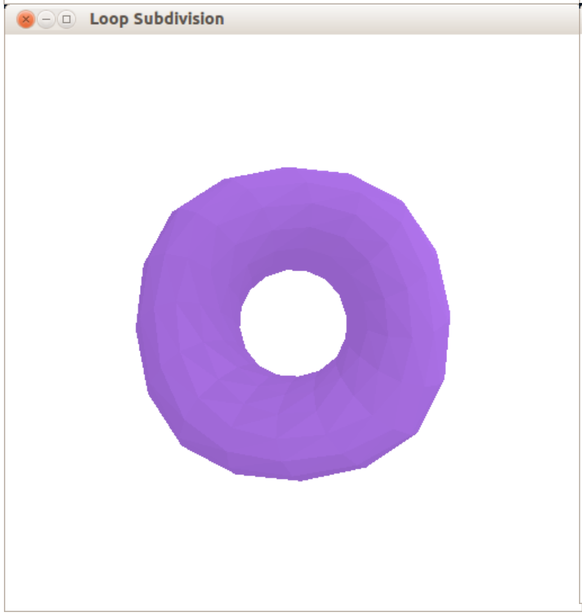
The implementation is shown below using snapshots of several different objects. The colors used for shading are all same. The program has to be re-run for each mesh loaded. Runtime changing of the mesh shows incorrect results.

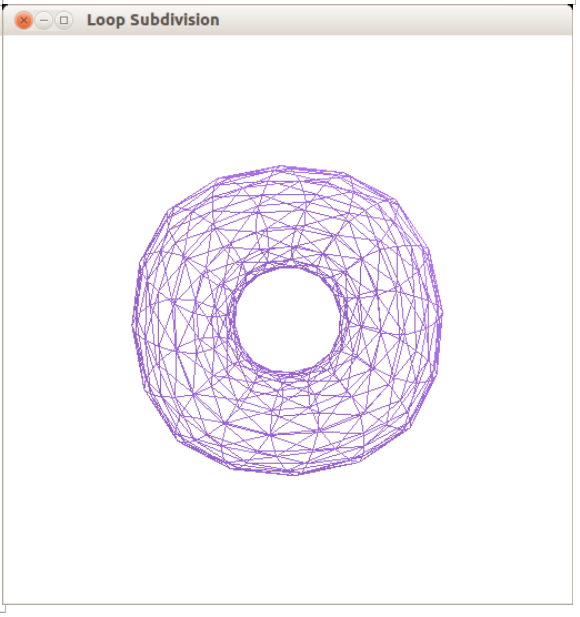

Loop Subdivision

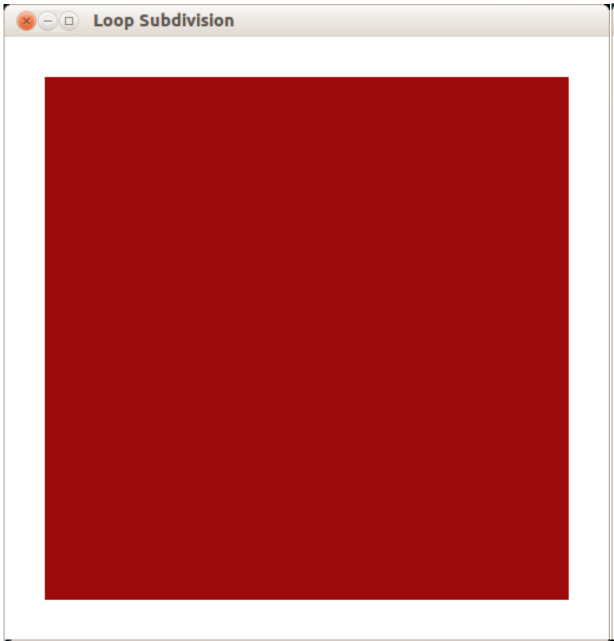

Loop Subdivision

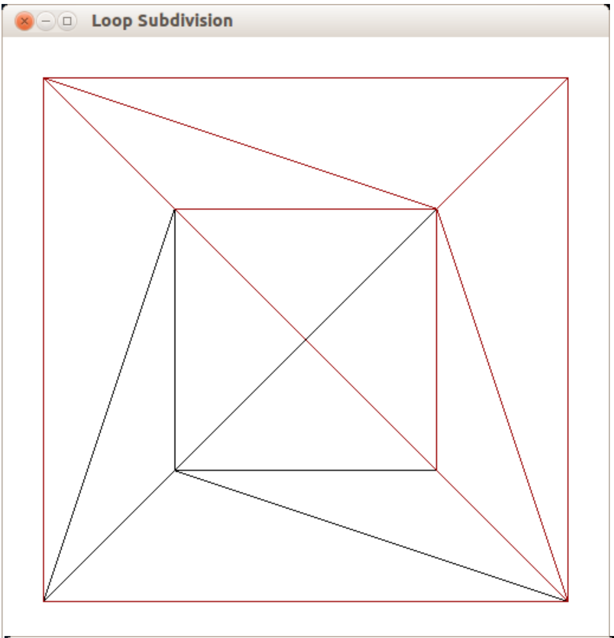

Loop Subdivision

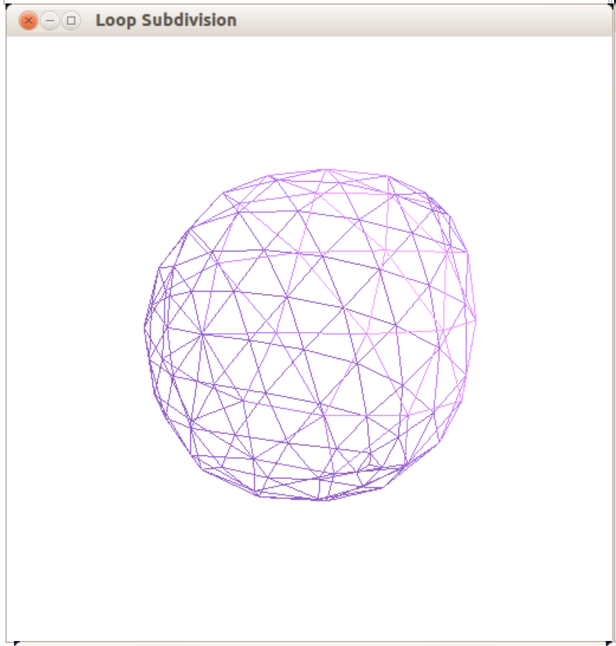

Loop Subdivision

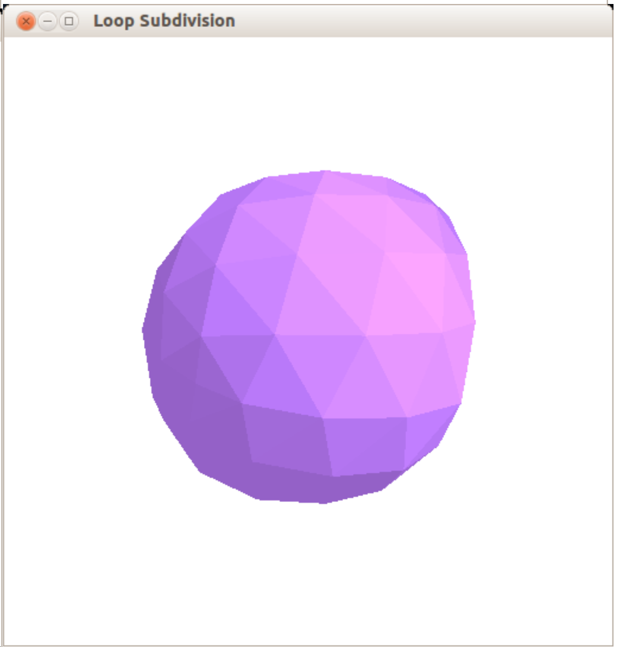

Loop Subdivision

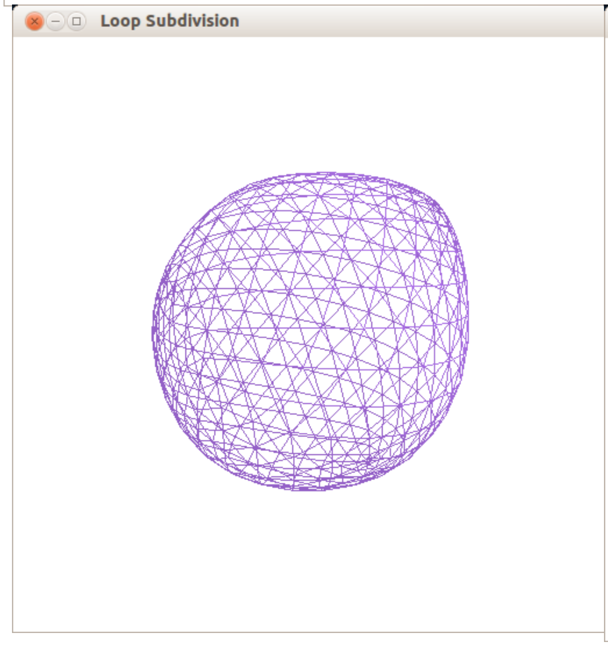

Loop Subdivision

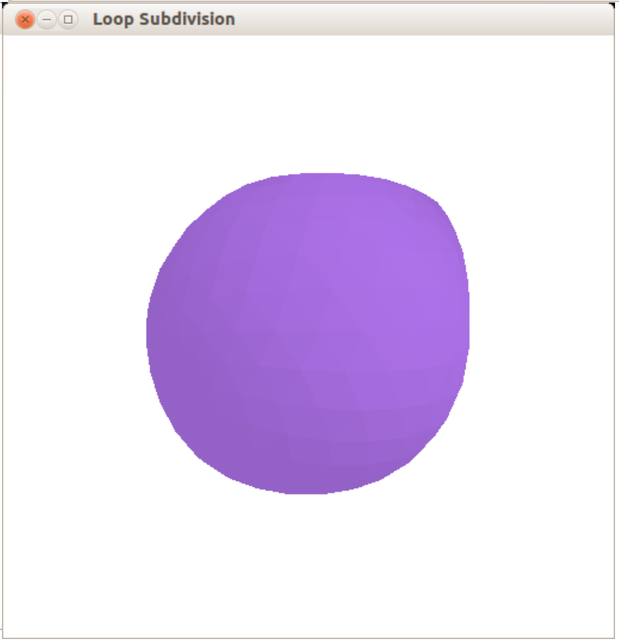

Loop Subdivision

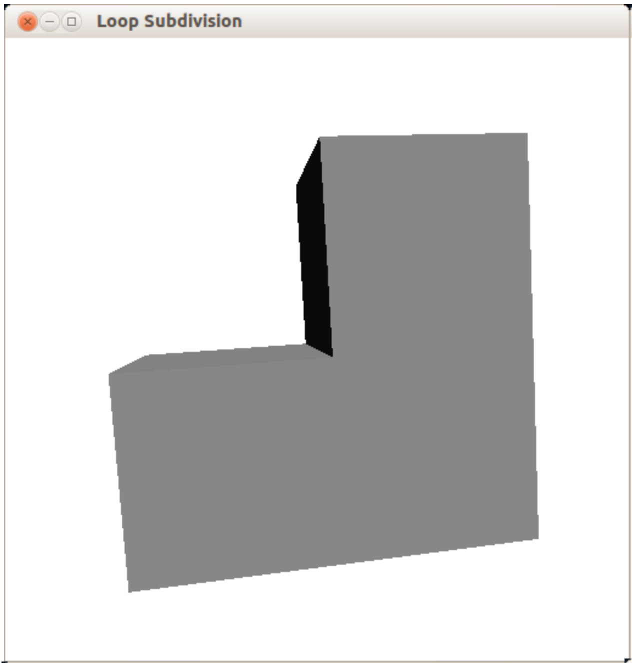

Loop Subdivision

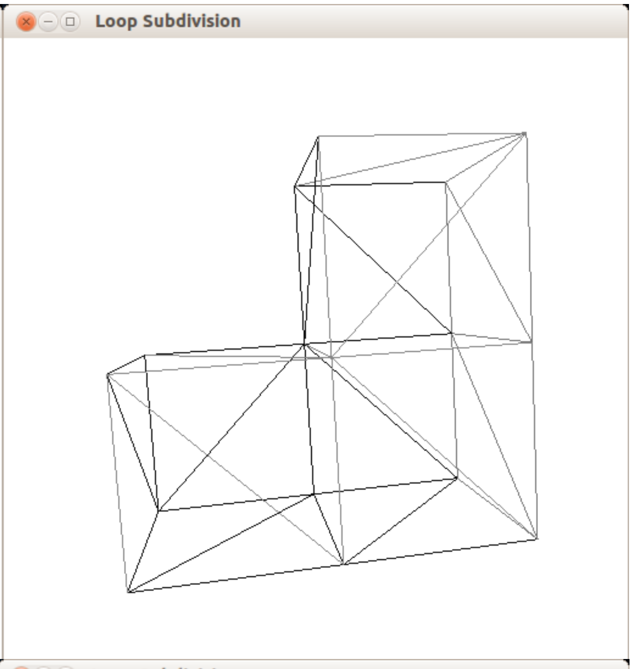

Loop Subdivision

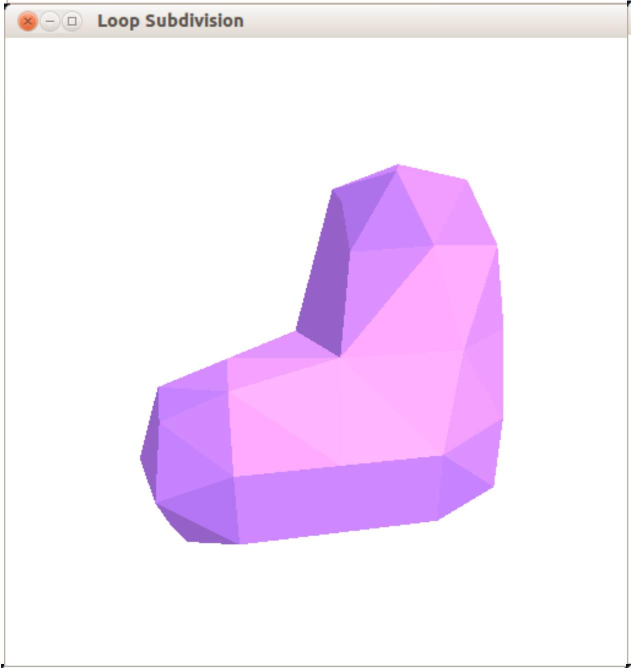

Loop Subdivision

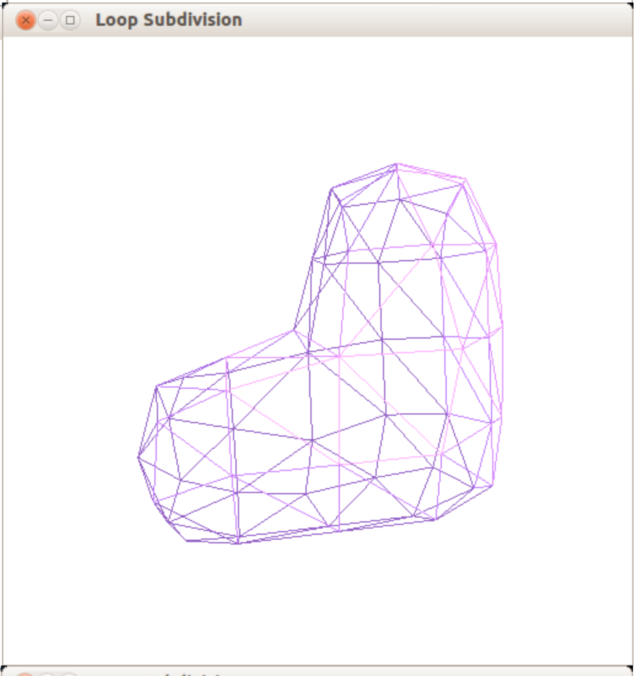

Loop Subdivision

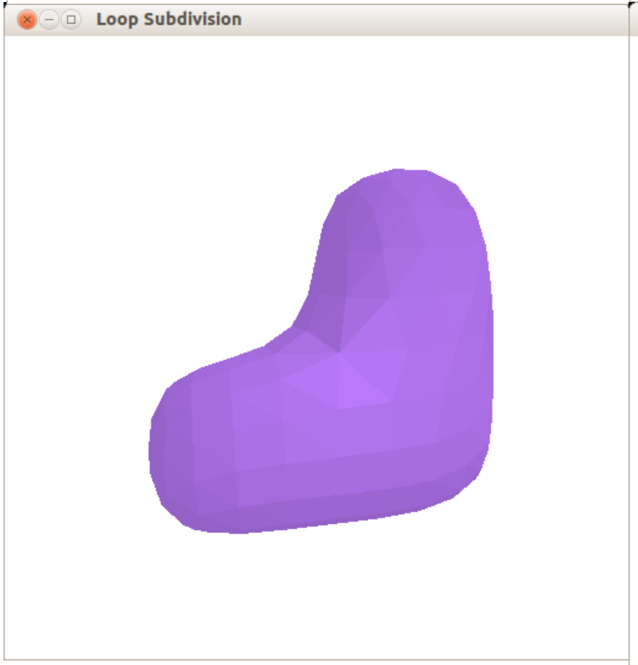

Loop Subdivision

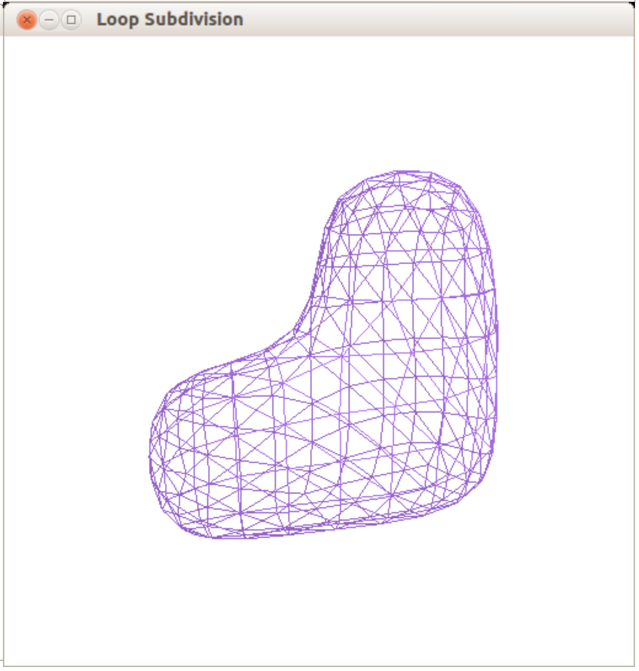

Loop Subdivision

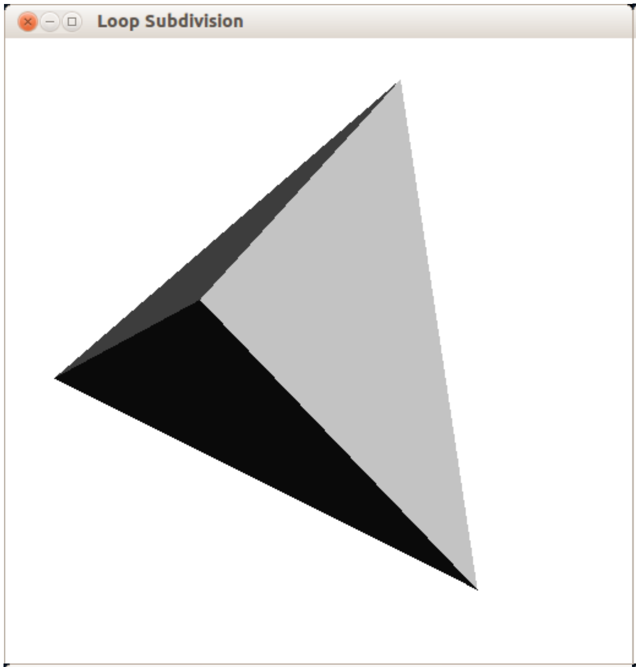

Loop Subdivision

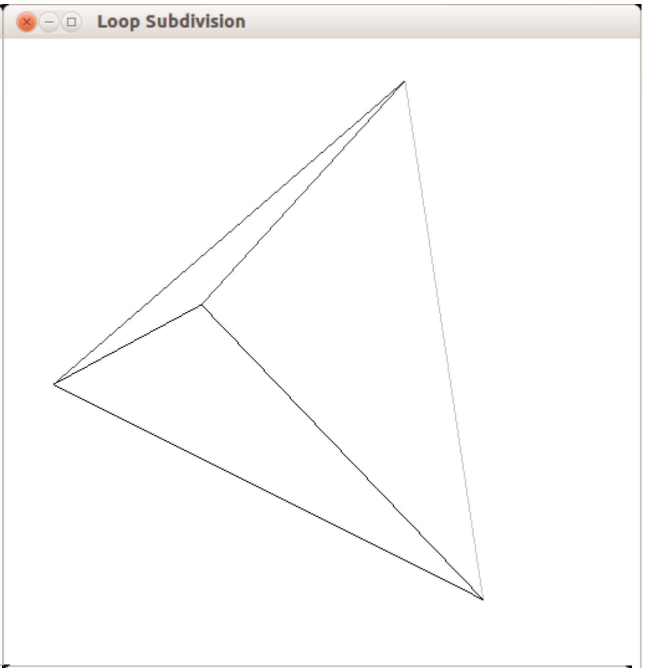

Loop Subdivision

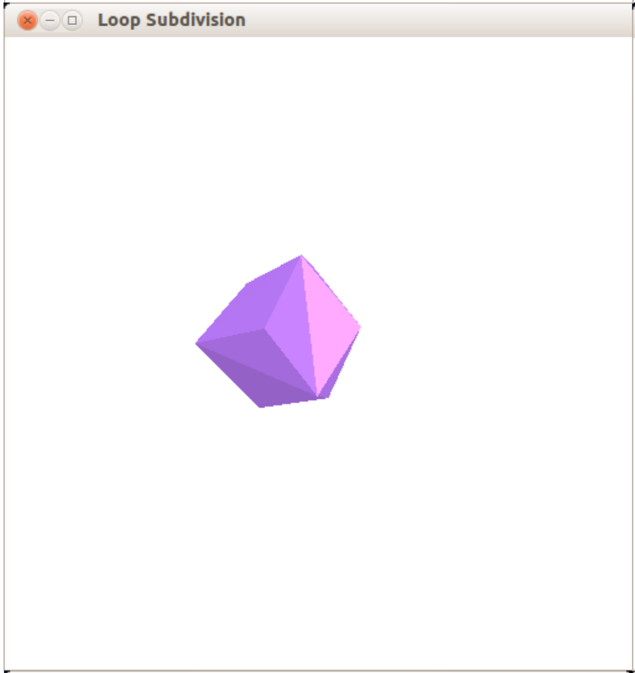

Loop Subdivision

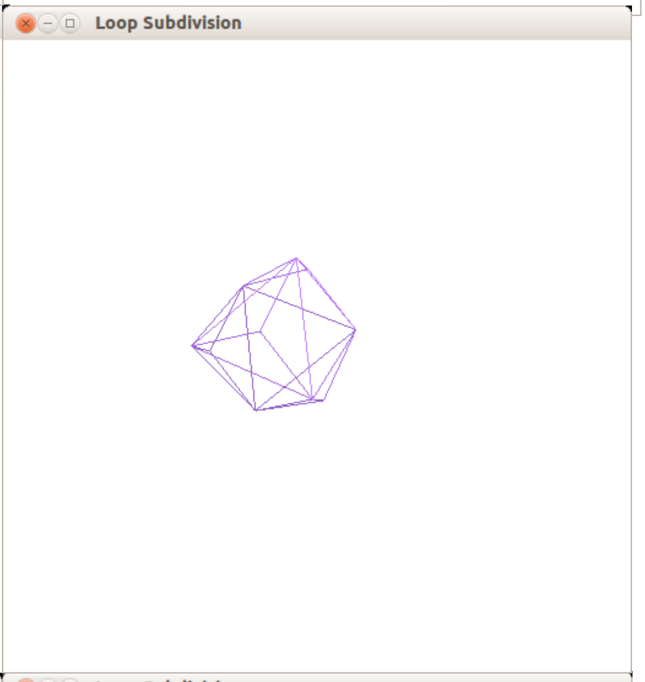

Loop Subdivision

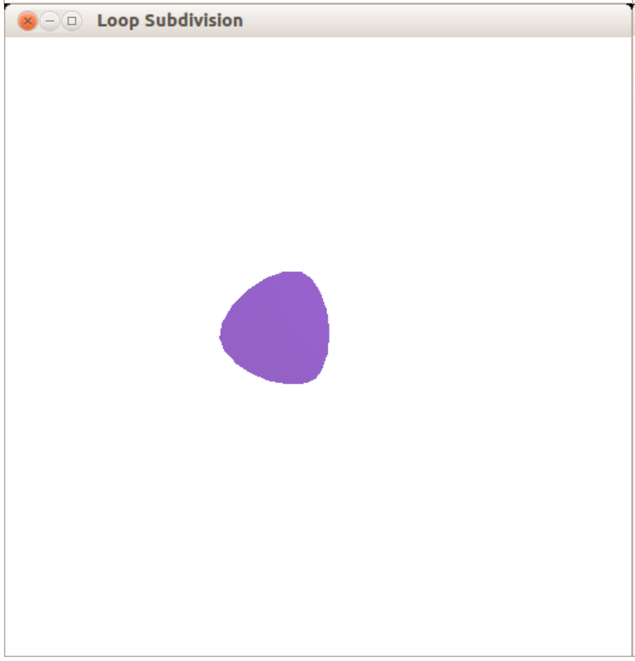

Loop Subdivision

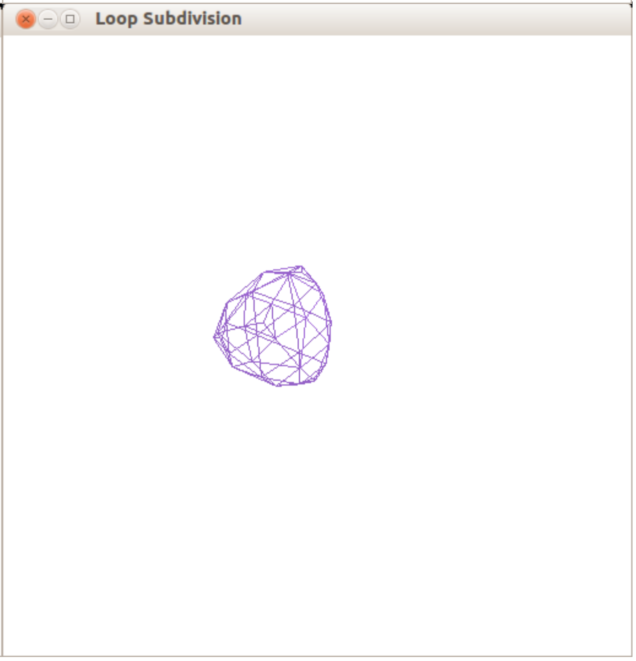

Loop Subdivision

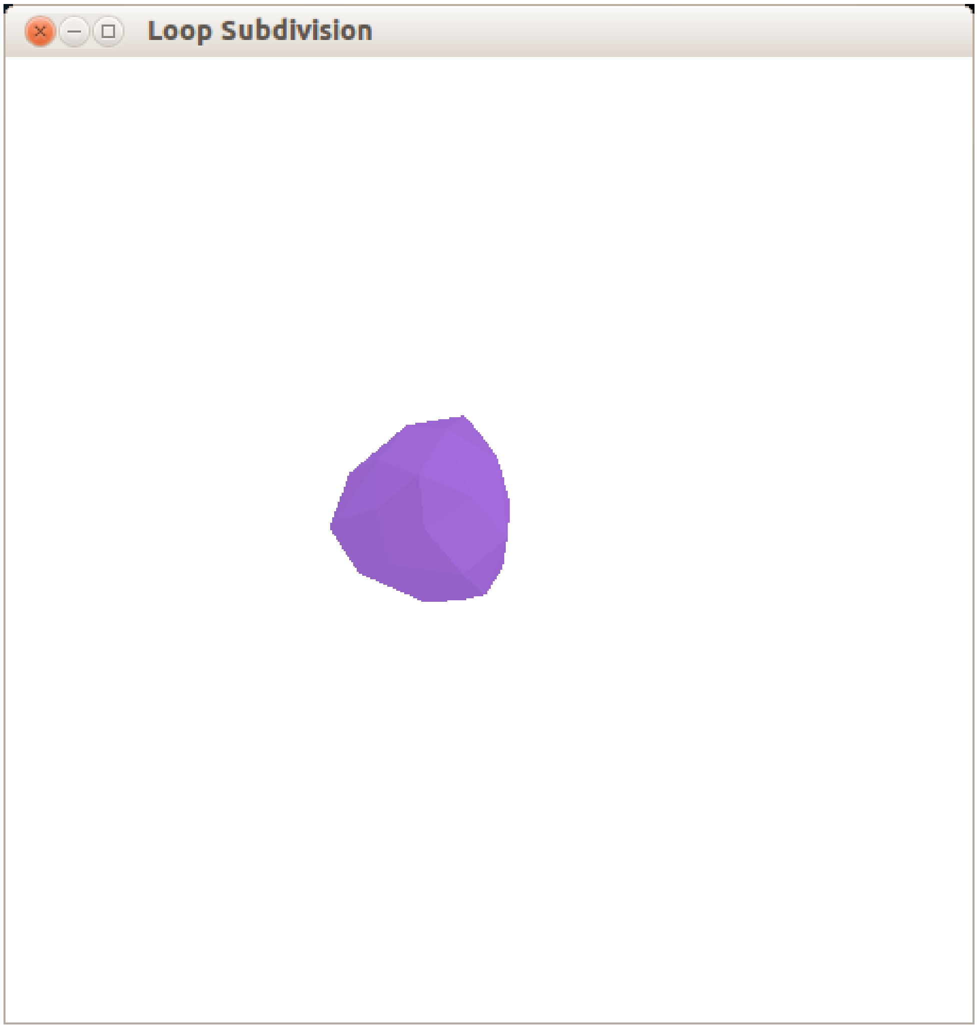

Loop Subdivision

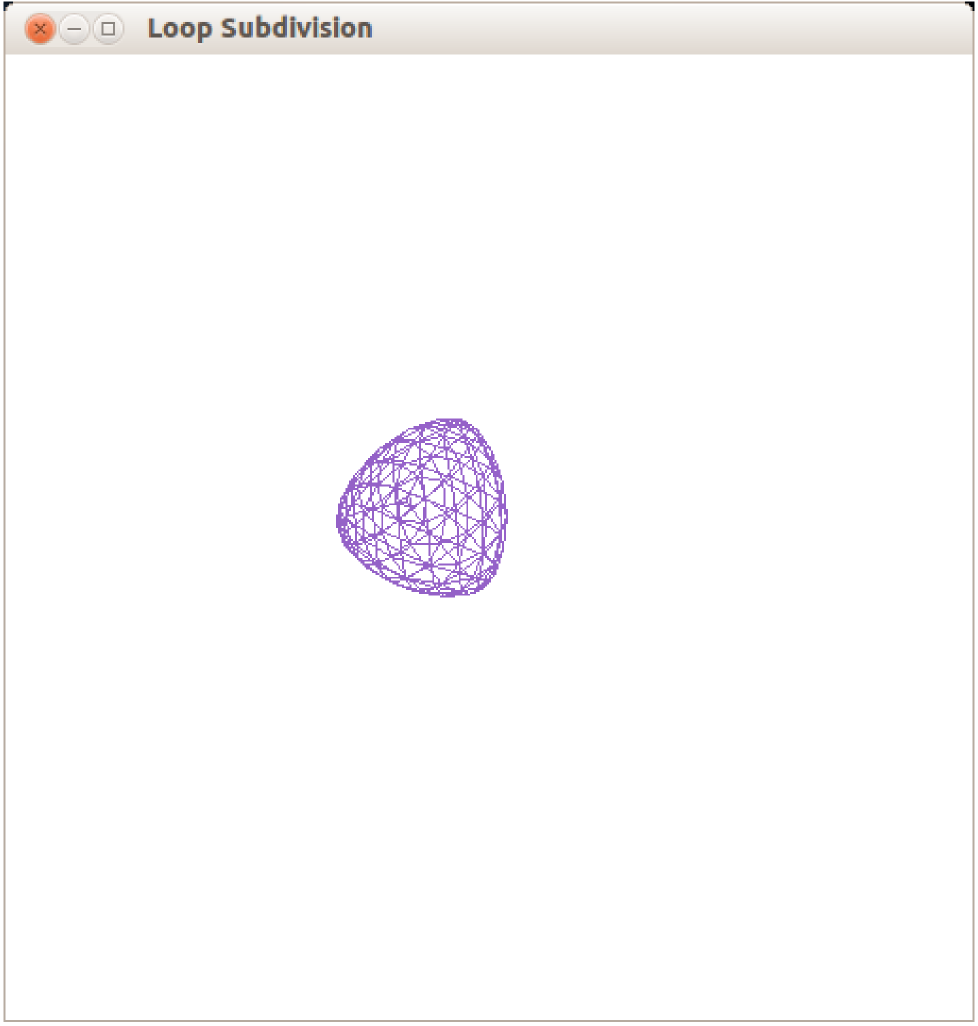

Loop Subdivision

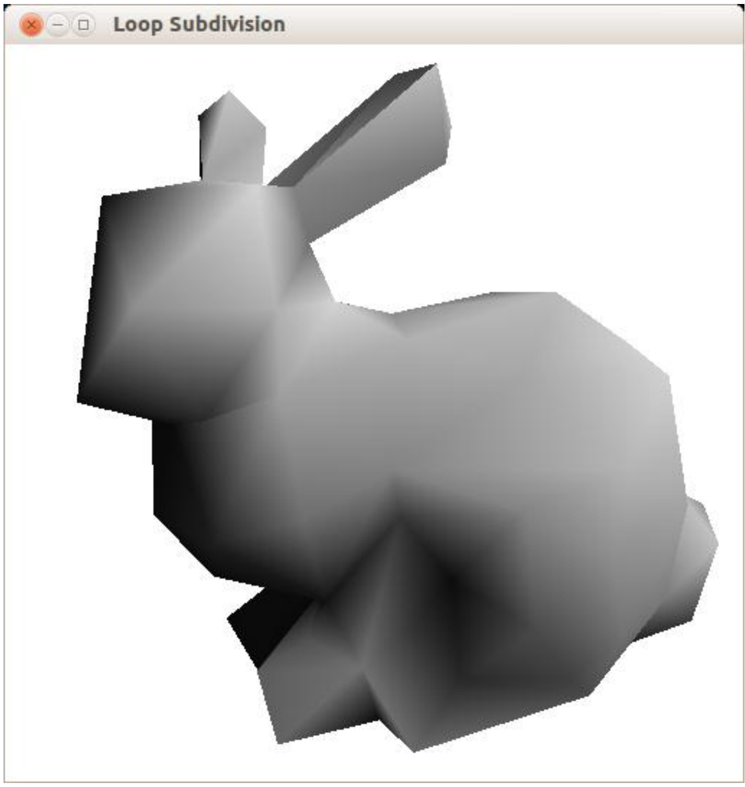

Loop Subdivision

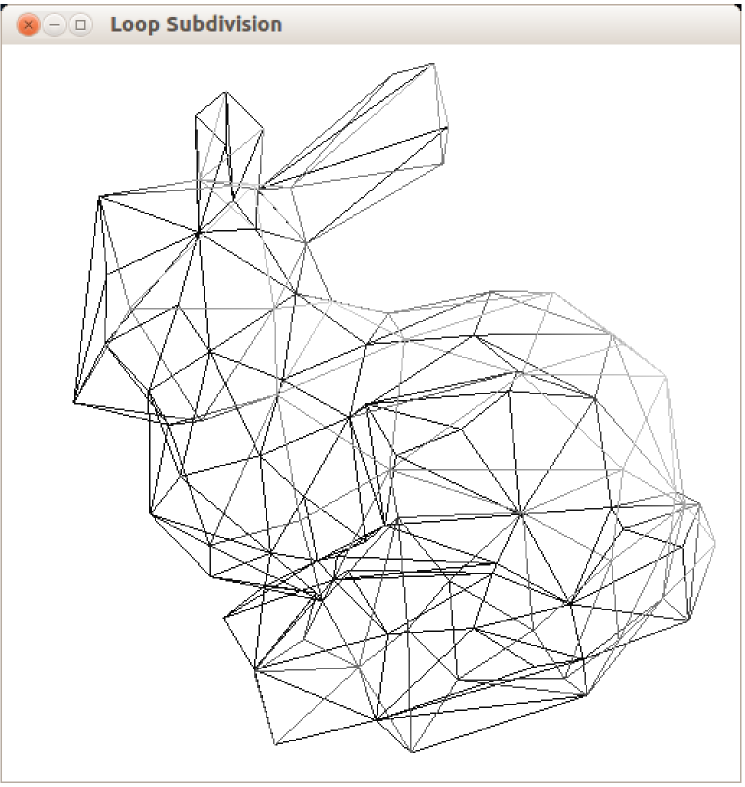

Loop Subdivision

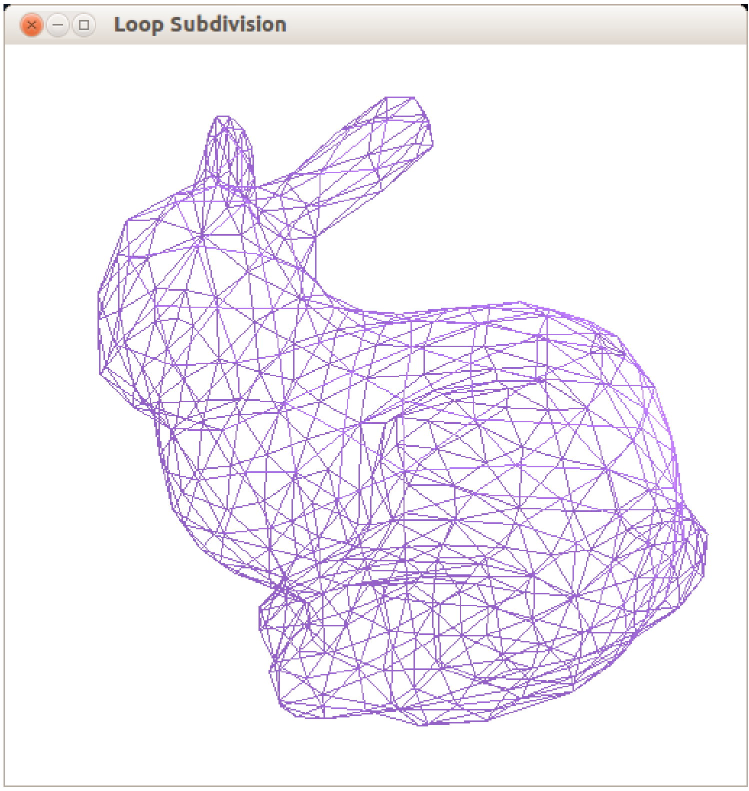

Loop Subdivision

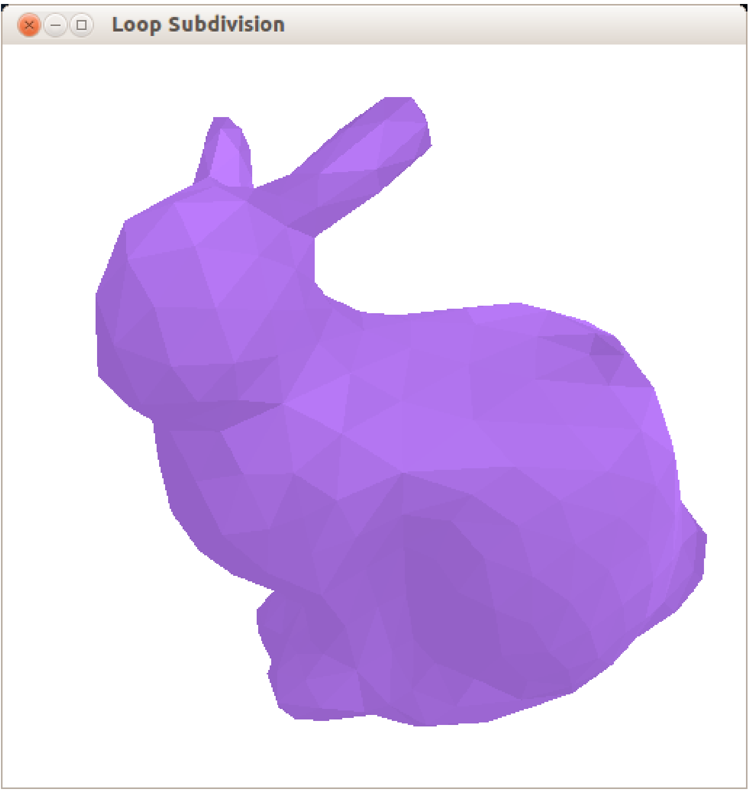

Loop Subdivision

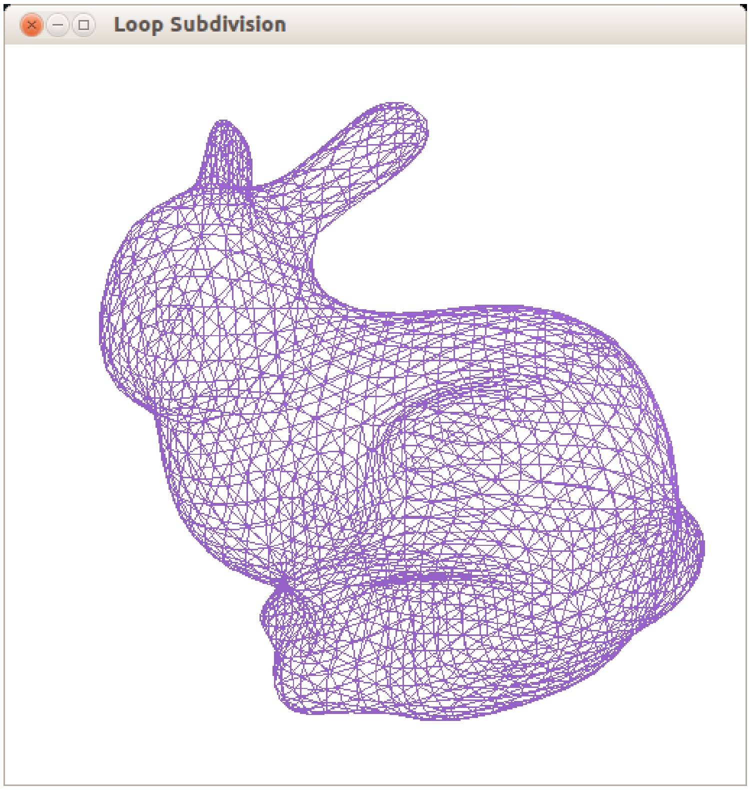

Loop Subdivision

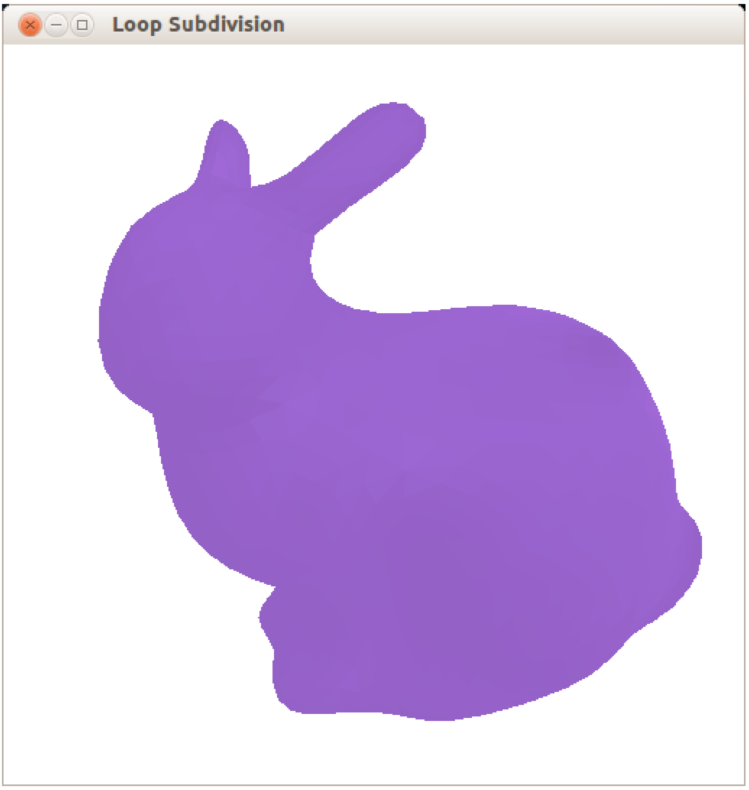

Loop Subdivision

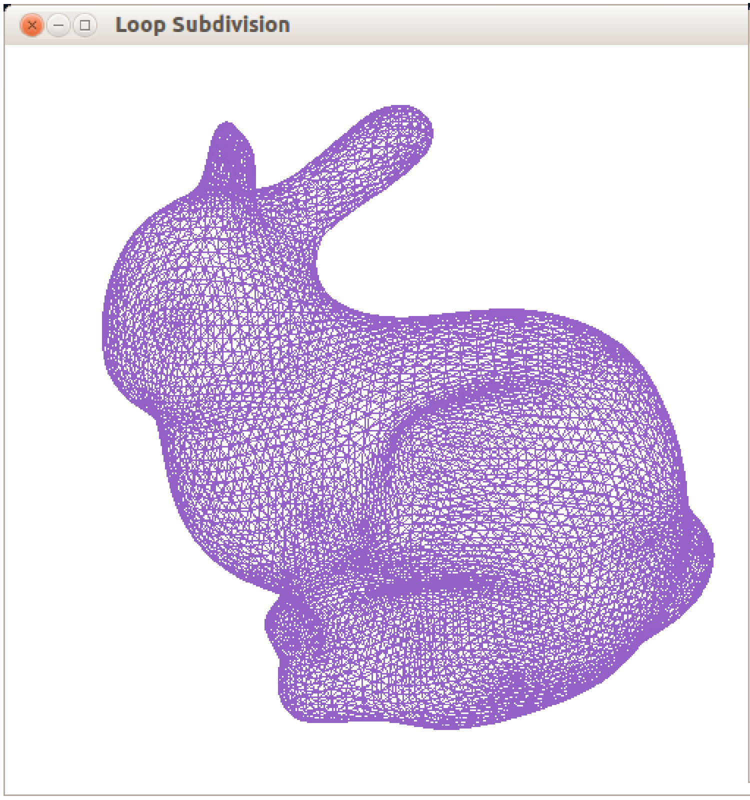

Loop Subdivision

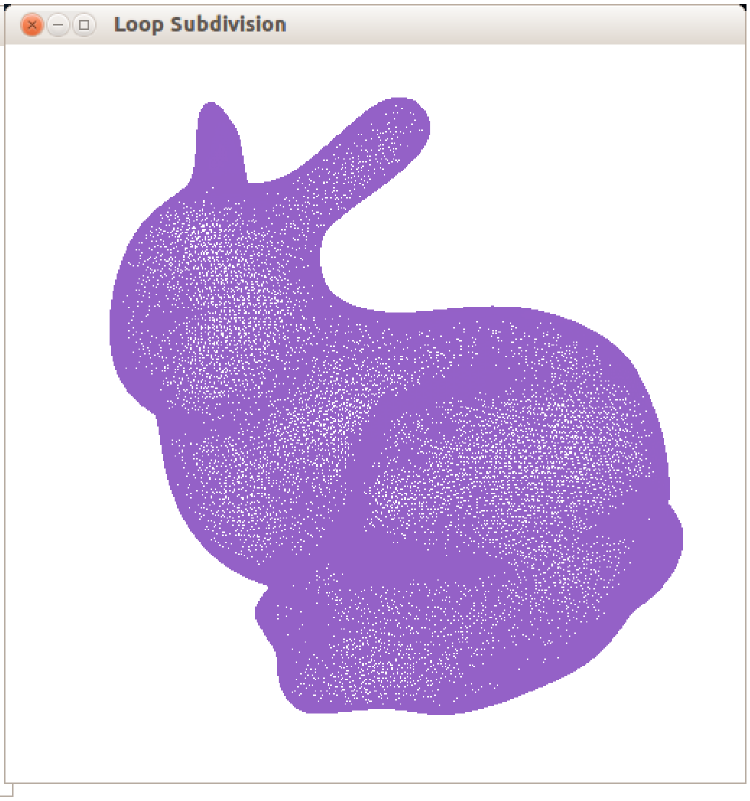

Loop Subdivision


Loop Subdivision


Loop Subdivision


Loop Subdivision


Loop Subdivision


Loop Subdivision

Shading vs. Geometry

Instead of using Phong shading model (where normals are interpolated for every pixel inside a triangle), would it make sense to subdivide the triangulation and use flat shading?

At the first level subdivision the use of flat division would not be a good choice because the number of triangles in the mesh is still comparatively less. As we progress with the subdivision and the triangles become more in count and the surface becomes smoother, the flat shading would be the better choice. Since the triangles become quite small and the transition from one triangle to the other becomes more and more smooth. But when we move to deeper and deeper levels of subdivision, the triangles become very small in size and the at one point might not very bigger than a pixel At this point, both flat and Phong model will be doing the same thing. Hence there is a threshold till where flat is better than Phong.

As mentioned above, at lower levels of subdivision the computational cost of flat shading would be far lesser than that of the Phong model. But as we progress and the triangles becomes smaller and smaller the cost of the both would turn out to be equivalent.

Subdivision Shading discussed in http://dx.doi.org/10.1145/1409060.1409095

The author in the paper stresses on the point that as we use subdivision for the mesh, in the same manner we can use the subdivision on the normals of the vertices to achieve a better shading effect.

The subdivision for the normals can be calculated by an algorithm describe in the paper which involves the normalization of the linear combination of all the involved vectors, mapping the input normals orthogonally this computed value and scale them to have their length from the origin with respect to the angle to the value computed. Then perform linear combination on this map and map it back onto the sphere by a rotational value. And repeat the above steps. This way the subdivision along with the mesh will lead to a faster and better way of shading. The end result is better than that of calculated using flat shading using just surface normals and computationally far less expensive than Phong shading. The author states that several tweaks can be done so as so make it more widely usable with different kind of meshes and that the best results are available if the normal interpolation is done after one level of subdivision of the surface.

Acknowledgements

As mentioned earlier, the assignment is done using the original program written by Nate Robbins found at:

http://user.xmission.com/~nate/smooth.html

Apart from the obj files provided in the above package, several other meshes such as the bunny, torus and knot were downloaded from various sites such as:

http://www.cs.rpi.edu/~cutler/classes/advancedgraphics/S09/hw1_meshes/meshes.zip

http://groups.csail.mit.edu/graphics/classes/6.838/S98/demos/Adjacency/AllDemos/torus.obj