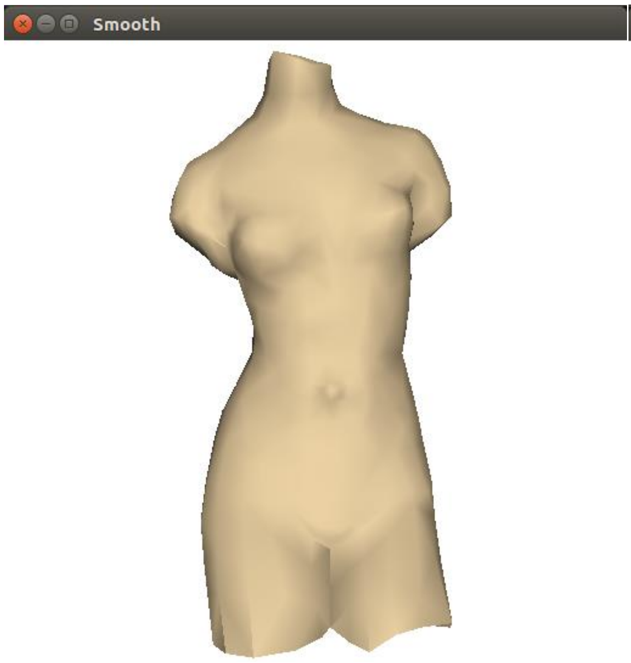
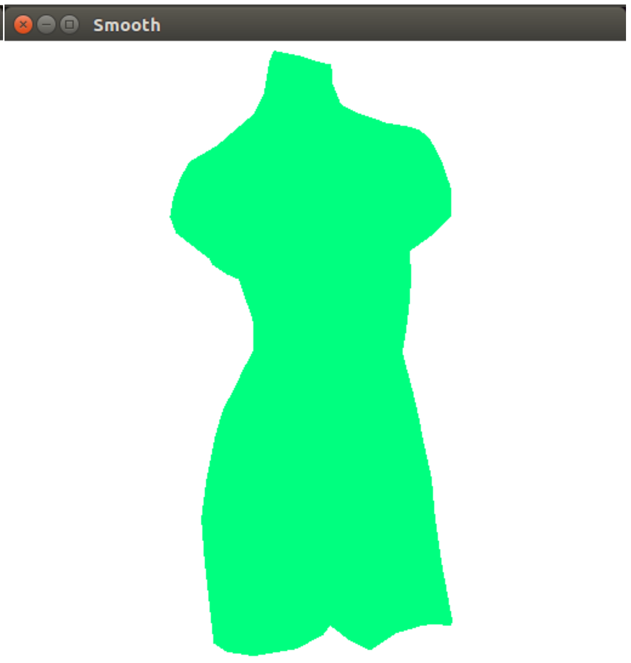
Kyler Ferrell-Clegg
CAP4730

Homework 2 Report

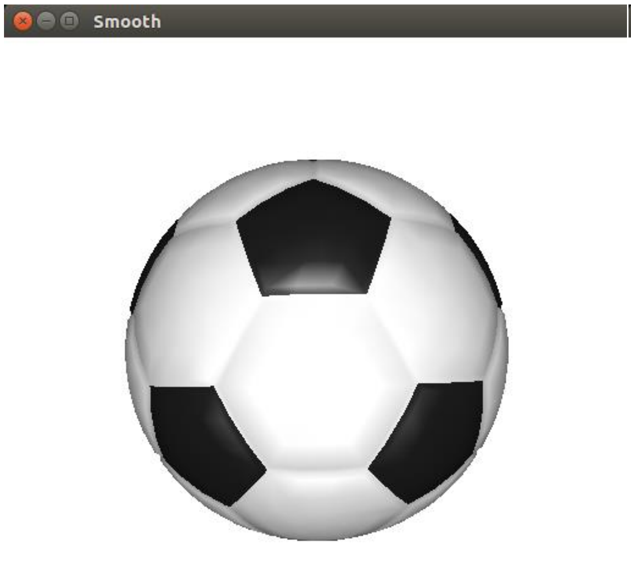
In this assignment we were tasked with rasterizing a model in FreeGLUT using an OpenGL program called Smooth. The first part of the assignment was to modify the smooth program in order to add a rasterization pipeline. The pipeline goes through all of the triangles in the model and rasterizes each triangle separately. This was one of the more difficult parts of the homework however I eventually got it to loop through all of the triangles correctly. Basically, I go through all of the groups in the model and loop through all of the triangles in that group using the correct indices. Next, I found the 2-D coordinates (x,y) of each of the 3 vertices of the current triangle I was working with. This required us to use gluProject() which returned (x,y,z) coordinates. Then, I simply looped through each of the pixels on the 512x512 screen and filled in with a color other than white (green). Finally to display I use one call of glDrawPixels() with our RGB pixel matrix values. The result is displayed below.

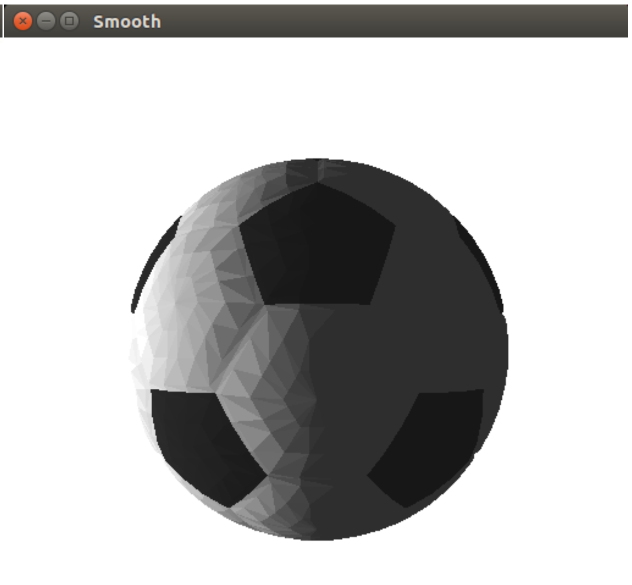


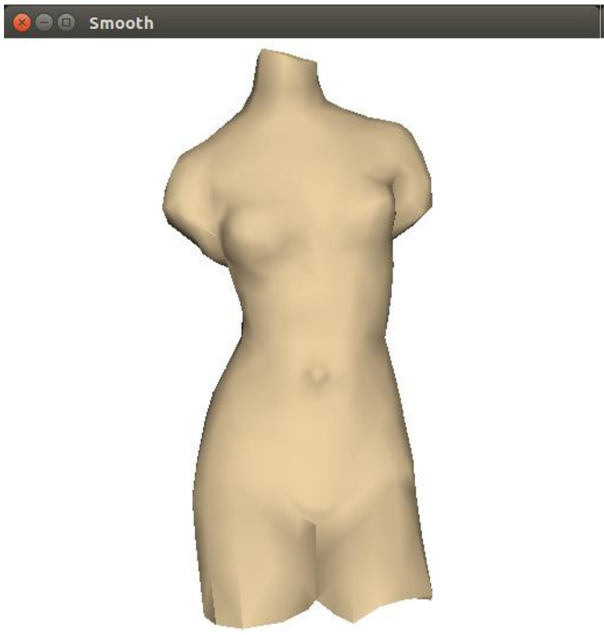


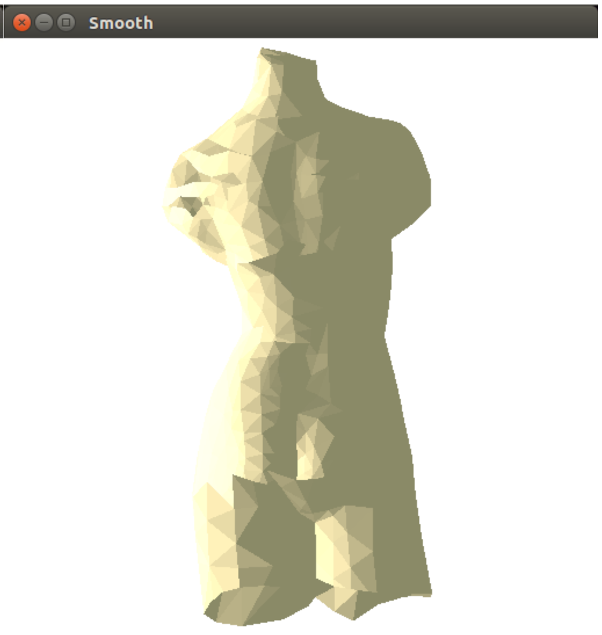
The next stage dealt with Z-Buffering and Flat shading. Z-Buffering was super simple to implement. The approach I took was simply have a matrix of dimensions [512][512][1] so we have a single z value associated to each pixel. Then simply check to see whether the current z value we are working with is less than or greater than the one that is currently stored for that pixel. For flat shading I simply took the normal, ambient, diffuse, specular and shininess values for each of the vertices/triangles. The results of this process are displayed below. I choose to scale the values by 0.9 for most of the lighting (ambient, diffuse, specular) and had a unique coefficient for each. For ambient I simply used the normalx value with some value I applied. For diffuse I again used the normalx value but adjusted it slightly different. For specular there was unique component called shininess that I utilized along with the coefficient using in diffuse shading.

My implementation is slower than the original implementation of smooth because I use a very inefficient approach to rendering the object. I have multiple inner loops that go through all of the data and calculations which results in having the program sometimes take multiple seconds to compile and execute.









Kyler Ferrell-Clegg
CAP4730