Assignment 1

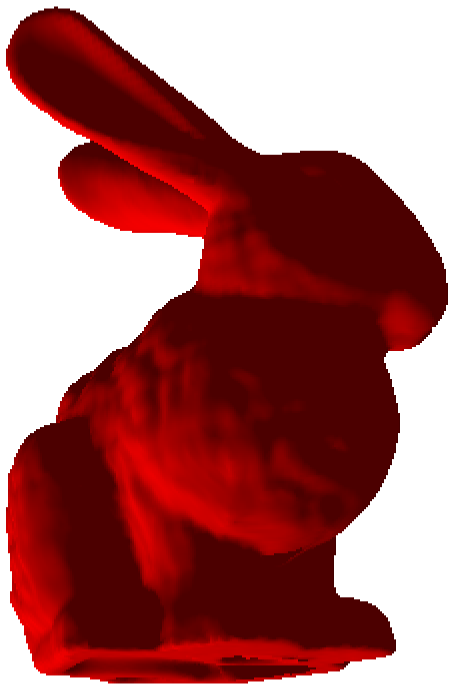
CSCI 3090

Anthony Di Donato

100517433

Part 1.

I had quite a few problems with this part. One major one that I figured out near the end was the problems I was having with the view. It seemed like no matter what I did the bunny would just not show up when I displayed using modelViewPerspective instead of just model. The problem I was having was actually due to a few things. The first problem was that I had my eyez variable set to a positive value, and for some reason the model was actually being drawn at the origin or maybe slightly behind the origin. To actually find the model I changed the wasd controls to move on the x and z axis and after moving slightly backwards I finally saw the bunny (sideways, but I fixed that as well). The other problem was just that due to the actual size of the bunny it would be near impossible to find without having an idea of where it was. Another problem I had was the normals, it seemed like no matter what I did the normal just wouldn't show up correctly. I triple checked all the calculations to confirm that I did everything right but still it looked like a mess. Once again there was a few problems but the main issue besides calculating normal wrong was that I needed to do a depth test, which didn't work initially because I was drawing the model the wrong way, then when I drew it based on the view it still didn't work because to find the model I set the near field to 0 on the view, and the depth test just doesn't work with that. After changing it to 0.1 the model now looks good.



Part 2.

Realistically I just ran out of time for this one. (Spent too much time trying to fix the problems I had with the bunny)

What I did get done for this part is getting the points along the line that should be drawn, I put a comment tag that looks like this // ********************important*********** to show where that all is. I also read in the points and put them in an array, basically most of the stuff besides actually drawing it.

As for how I calculated the points, I did a loop that runs 2 less than the amount of points on the spline times, that takes the values at the initial point, and the 4 following points, and multiplies them by the B array (once for each of x y and z). Then I looped through u 10 times, for values between 0 and 1 and multiplied the Bx/By/Bz arrays by that at each point to get a single value.