CGRA 350 Assignment 1 – Jackson Tume

300446258

Part 1: Geometry

Core:

Brief introduction:

The function for the core part of my assignment is quite simple, it loops through the number of Lat and Long divisions needed and gets 4 points every time through the loop. The Top Left, Top Right, Bottom Left and Bottom Right. It uses these 4 points to create 2 triangles making up a square, adds the 2 faces to a vector. Repeats until complete.

How to run:

Have the imGui checkbox checked for Sphere Lat Long, play around with the sliders to get different results.

Completion:

Brief introduction:

The function for the completion part of my assignment is similar in theory to how I did the core. With grabbing 4 points each time through the loop, this time only looping through the number of needed subdivisions. The difference this time, it needs to run 6 times, one time for each face of the square(Top,Bottom,Left,Right,Front,Back).

How to run:

Have the imGui checkbox checked for Sphere Cube to Sphere, play around with the slider to get different results. ***WARNING*** the program works fine if clicking the slider, but because of the amount of data being computed, it *can* crash if the slider is moved around quickly.

*Challenge: *Not Attempted**

Part 2: Shaders

Core:

Brief introduction:

Only the Cook-Torrance shader was implemented for the core. It works by first storing some calculations that make it handy after, then it can move on to the actual Cook-Torrance steps. The first step being calculating the reflectance, then using Beckmann we can calculate the distribution, and the final step being calculating the shadows. Adding them all up with the Cook-Torrance formula gives us the desired results. **Note There is a bit of weird stuff that happens with my implementation, I couldn't figure it out, but it was quite odd.

How to run:

Have the imGui checkbox checked for Cook-Torrance.

Completion:

Brief introduction:

I used the stb_image header file to easily load in my image data. After setting up some default parameters for OpenGL textures, the images are bind and sent to the fragment shader. Depending on the version running (There are two final colours for the texture, the default one is the normal map version. If you comment it out and uncomment the other one it will show the version without normal maps and just the texture. You can find this in the file “texture_frag.glsl”), the normal map version will do a couple small quick calculations to figure out the colour values and specular values and such. The version with just the Texture and no normal maps will simply just use the texture with no other fancy calculations.

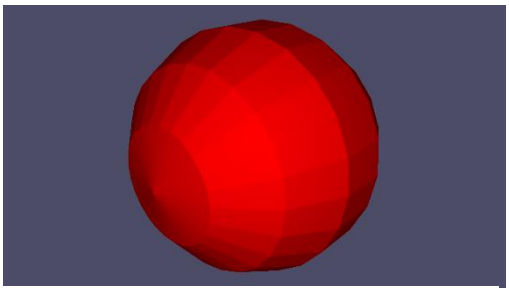
How to run:

Have the imGui checkbox checked for Texture.

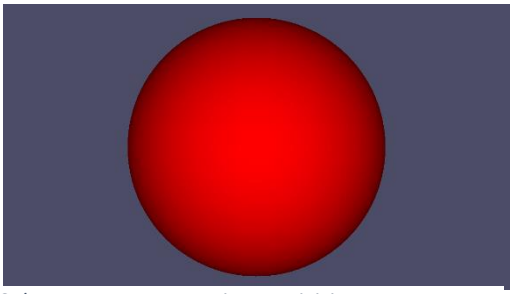
*Challenge: *Not Attempted**

Part 3: Questions *Not Attempted*

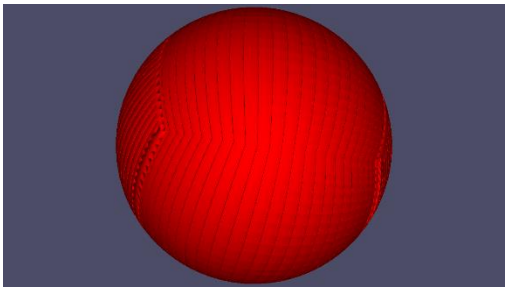
Results:



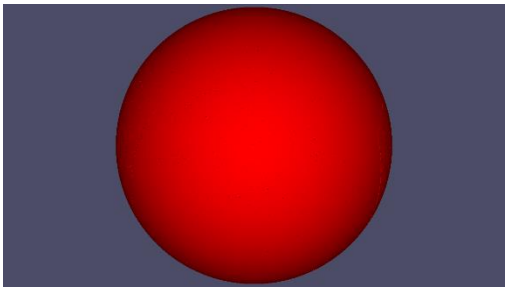
Sphere Lat-Long – Minimum Divisions



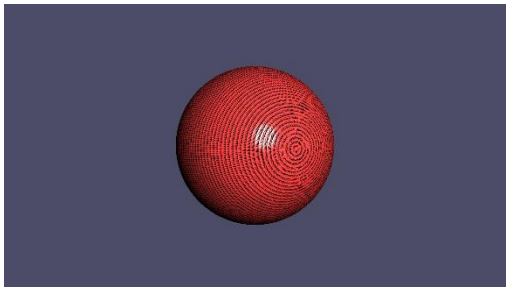
Sphere Lat-Long – Maximum Divisions



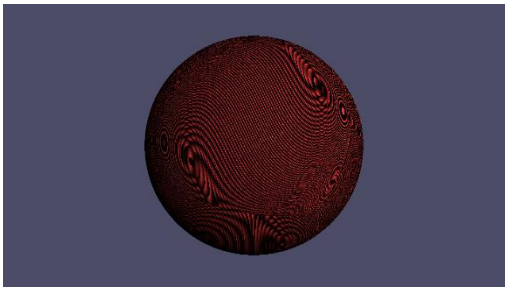
Sphere From Cube– Minimum Divisions



Sphere From Cube– Minimum Divisions



Sphere Lat-Long– Cook-Torrance



Sphere From Cube– Cook-Torrance

Sphere Lat-Long– Texture (With Normals)

Sphere From Cube– Texture (With Normals)