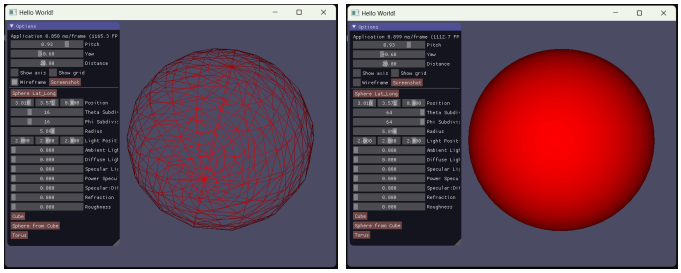
Ryan Holloway

CGRA 350: Assignment 1 - Report

To run the geometry generating methods, simply use the ImGui interface to click the geometry you would like to create. The geometry parameter controls change based on the geometry selected, however all the lighting parameter controls remain the same regardless of the geometry or current shader enacted. This means some of the lighting controls are not effective depending on which shader is enacted. The default shader is set to the original shader that came with the framework. To update the shaders requires manually entering the filename of the shader (see shader section for more details). Be sure to be selected on the ‘Sphere Lat_Long’ when testing the shaders.

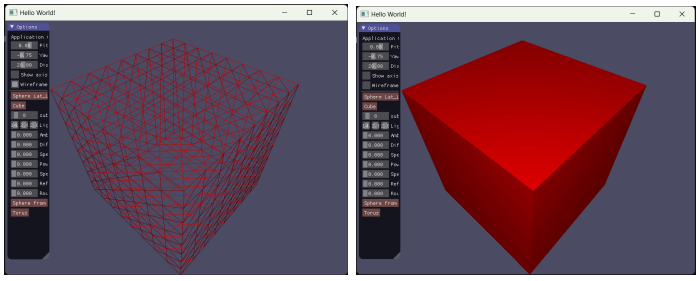
Sphere:

void sphere(int theta_subdiv, int phi_subdiv, float radius, float x_position, float y_position, float z_position);
Theta spans 0[, 2PI], while phi spans [0, PI]. All the vertices were generated around a centroid, so the sphere can change position. Scale the radius parameter to increase or decrease the size. Theta subdivisions and phi subdivisions can be altered independently.



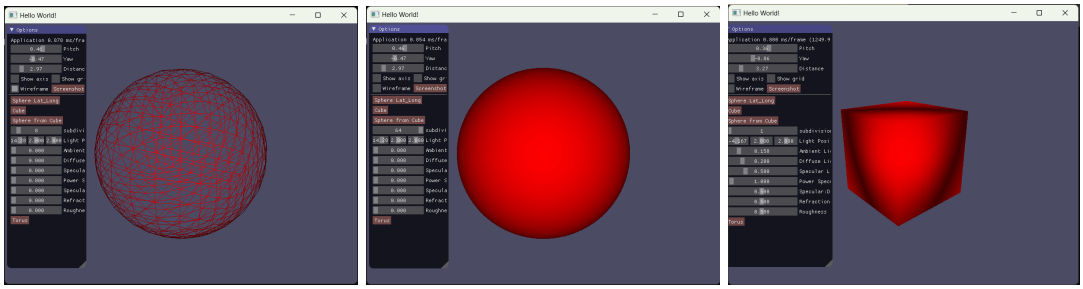
Cube:

void cube(int subdiv, float radius, float x_position, float y_position, float z_position);
Six individual faces are generated that align to create a cube.



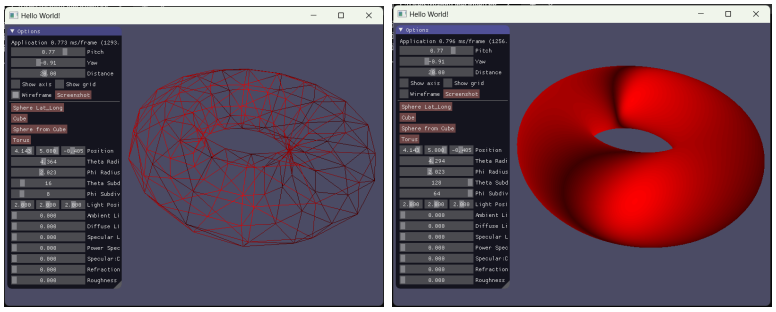
Sphere from Cube:

void cube_sphere(int subdiv, float radius, float x_position, float y_position, float z_position);
The position of the vertices from the cube are normalized to create the cube-sphere. Set the subdivisions to 1 to see it form into a cube.



Torus:

void torus(int theta_subdiv, int phi_subdiv, float theta_radius, float phi_radius,
float x_position, float y_position, float z_position);
I used trigonometry and projection to generate the positions of the vertices. This geometry can scale the theta radius and phi radius, as well as change position. Theta subdivisions and phi subdivisions can be altered independently.



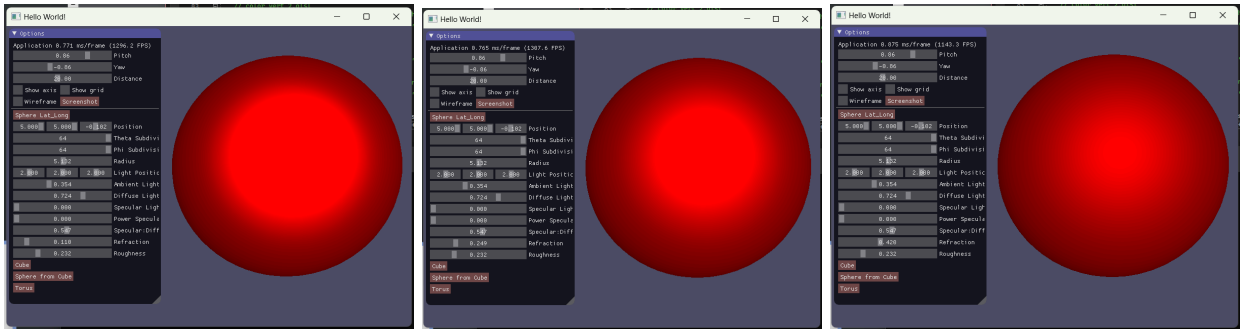
Cook-Torrance:

Hard-code these shaders into line 96 and 97 of Application.cpp.
color_vert_3.glsl
color_frag_3.glsl

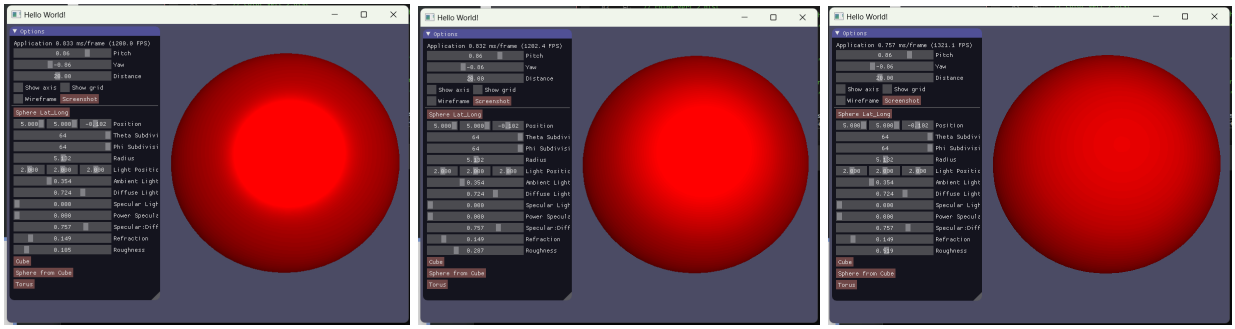
Active ImGui controls; Ambient Light, Diffuse Light, Specular:Diffuse Ratio, Refraction, Roughness

The lighting parameter controls have various combinations that can generate different looks. Please experiment with altering the values to get a better idea of how they interact with each other.

Refraction;
For the sake of creating these screenshots, I left other parameters the same while altering the refraction values. Refraction values, 0.110, 0.249, 0.420, respectively.



Roughness;
For the sake of creating these screenshots, I left other parameters the same while altering the roughness values. Roughness values, 0.105, 0.287, 0.519, respectively.



Oren-Nayar (implemented into Blinn-Phong as the diffuse component):
Hard-code these shaders into line 96 and 97 of Application.cpp.
color_vert_4.glsl
Color_frag_4.glsl

Active ImGui Controls; Ambient Light, Specular Light, Roughness

For the sake of creating these screenshots, I left other parameters the same while altering the roughness values. Roughness values, 0.011, 0.343, 0.641, respectively.