CGRA 350 A1

Shikhar Mishra

August 2024

1 Introduction

The assignment requires to implicitly create geometrical shapes and apply advanced shading models that are more physically based than the phong shading model. The report explains the functions used in the assignment and the results for the different parts.

2 Functions

2.1 Geometry

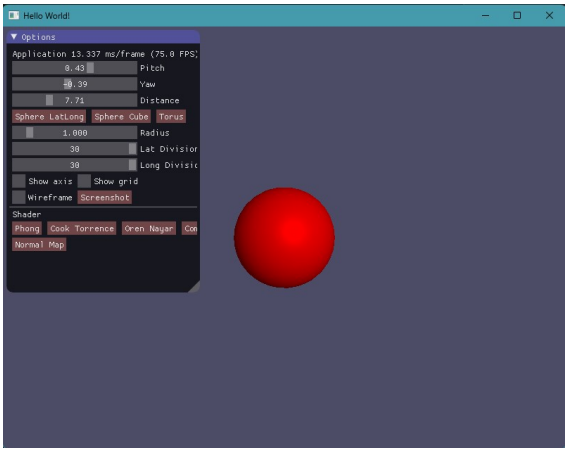
For the three different geometrical shapes, I have created functions, load_sphere_lat_long(), load_sphere_cube() and load_tor_lat_long(). These functions create the three geometries of the core, completion and challenge part of the geometry section. They are hooked on to buttons in ImGUI menu and clicking on the buttons loads the respective shapes. Depending on the shape and parameters, ImGUI sliders show up to control the parameters.

The load_sphere_lat_long() function takes 3 inputs radius, number of latitude divisions and number of longitudinal divisions. This is the only function that passes tangent, bi-tangent and UV coordinates for its mesh. (That is useful for later parts of the assignment).

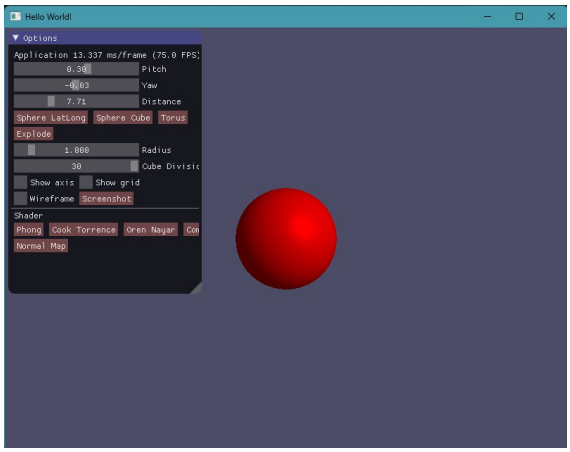
The load_sphere_cube() function takes 3 arguments, the radius, explode distance and number of divisions. The radius defines the size and the number of divisions on the cube face to generate vertices that get mapped on to the sphere. The explode distance is a fun parameter that I implemented to split the faces of the cube(although they retain the curvature).

The load_tor_lat_long() creates a torus with 4 parameters, the radius, thickness, number of latitude divisions and the number of longitudinal divisions.

The below figures show the images of these shapes generated. The default shader used for this implements the Phong lighting model with fixed parameters.

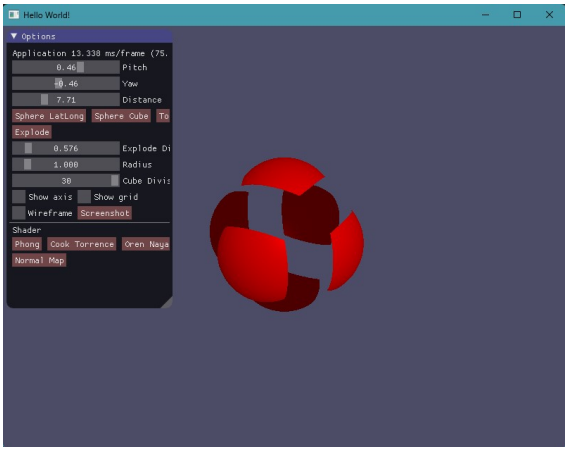


(a) Sphere Lat Long

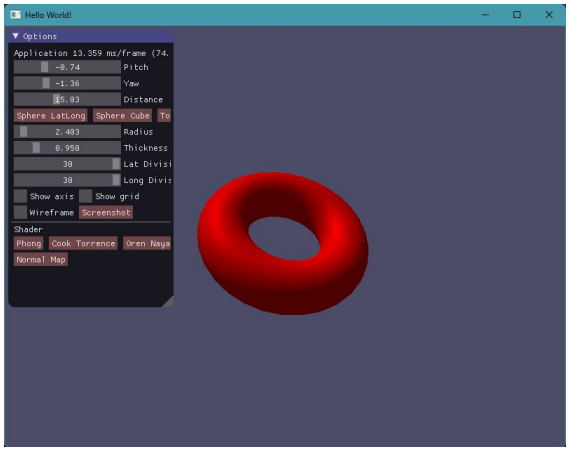


(b) Sphere from Cube

Figure 1



(a) Sphere from cube but exploded



(b) Torus

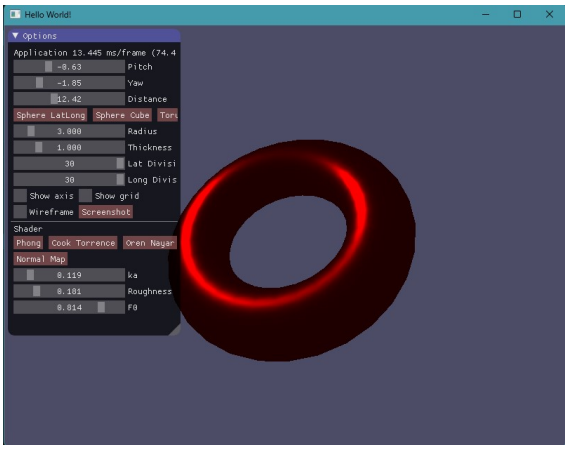
Figure 2

2.2 Shader programming

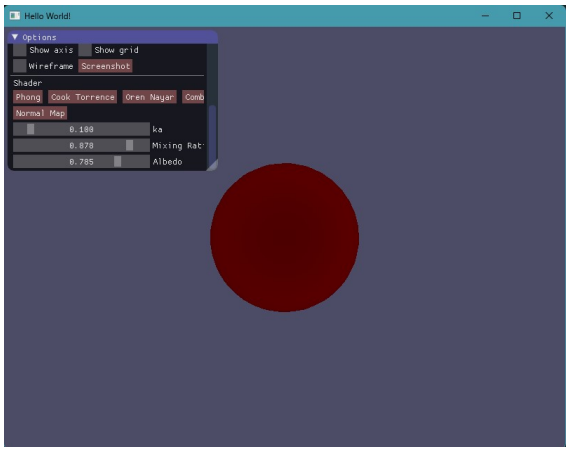
Similar to the geometry features, different shaders have been written in different files and there are different loader functions to attach them to the pipeline.The different shader functions implement the following shading techniques.

Cook Torrence: The vertex shader used is the color_vert.glsl(common used for some other shader programs) and the fragment shader is named cook_torrence.glsl. This shader program only implements the cook torrence specular component and has an ambient lighting control. The parameters used as uniforms are F0 for shlick approximation of Fresnel Reflectance, and roughness for normal distribution. Ka parameter controls the ambient lighting. For the Fresnel reflectence, I use the Shlick approximation. For the normal distribution function, I used the GGX approximation and for the geometry function I used kelemen approximation that combines the geometry function and the denominator cosine terms. All the information for implementation of these functions has been referred from the the website 'https://blog.selfshadow.com/publications/s2013-shading-course/#course_content' in the pdf labelled course notes.

For Oren Nayar shading model: I use the methods mentioned in the blog 'https://mimosa-pudica.net/improved-oren-nayar.html'. Again this only implements the Oren Nayar diffuse component and some ambient lighting. The uniform parameters used here are Rho and albedo. I may have not used the best naming convention while implementing this, but the term Rho is the mixing ration of the diffuse and non diffuse term as suggested by the approximation in the blog. The term Albedo has been used to represent reflectivity term that I do not fully understand.



(a) Cook Torrence specular shading

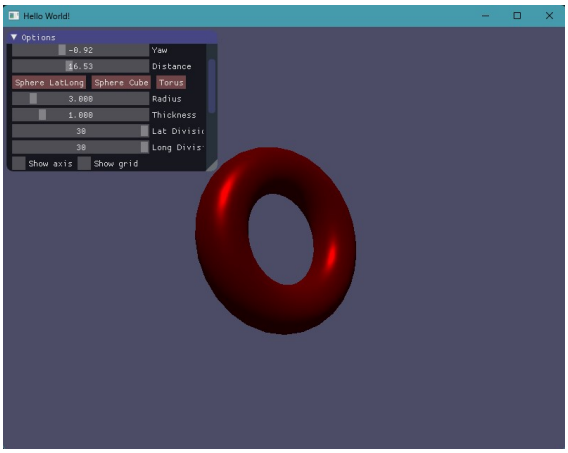


(b) Oren Nayar Diffuse shading

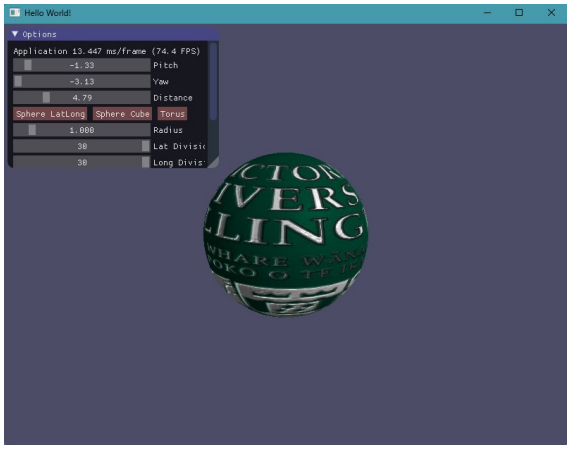
Figure 3

Finally, the button Combo, selects a shader that combines the Cook Torrence specular component and the Oren Nayar diffuse component into a single shader program with all control parameters.

The normal mapping shader implements the completion part of the shader programming section, which uses bump_vert.glsl and bump_frag.glsl shaders. This only works with the sphere lat long shape, as that is the only shape which has tangents, bi tangents and UV texture coordinates defined in the mesh. It takes in 2 textures, uses one for normal mapping and other for color from the texture to generate final lighting and color shading. This runs on top of the simple phong light shading model.



(a) Cook Torrence and Oren Nayar combo shader



(b) Normal Mapping shader

Figure 4