COSC 4370 - Homework 3

Name: Juan Yanez PSID: 2029097

October 2022

1 Problem and Objective

The assignment allows practice for 3D viewing and implementing the Phong shading model. This is done with a red colored 3D cube. Camera.h, main.cpp, phong.vs, phon.frag need to be completed in order to get a proper cube that displays the Phong shading model.

2 Method

By adding ambient, diffuse, and specular lighting, the cube is shown with proper shading. This is applied on each pixel of the model to implement the shading model; each normal vector is interpolated across each triangle. The Phong shading model equation goes as follows.

$$[r_a,g_a,b_a] + [r_d,g_d,b_d]\max_0(\mathbf{n}\bullet\mathbf{L}) + [r_s,g_s,b_s]\max_0(\mathbf{R}\bullet\mathbf{L})^p$$

The ($[r_a,g_a,b_a]$) is the ambient lighting, which is a constant value across all pixels. The ($[r_d,g_d,b_d]\max_0(\mathbf{n}\bullet\mathbf{L})$) is the diffuse lighting, which is the dot product of n * L, L being the light source, which is a more matte reflection. The ($[r_s,g_s,b_s]\max_0(\mathbf{R}\bullet\mathbf{L})^p$) is for specular lighting, which is the highlights of the model. R is the mirror reflection vector and p is the specular power, which defines how much shine it gives.

3 Implementation

All the implementation is done in multiple files, including, main.cpp, Camera.h, phong.frag, phong.vs. Using the Phong shading model formula, the red cube is implemented with a light source which will apply ambient lighting, diffuse lighting, and specular lighting.

3.1 Main.cpp

In main.cpp, there is a projection matrix that needs to be set up. Projection is set to equal a function, projection(), which holds the field of view, aspect ratio, near clipping plane, and far clipping plane. The field of view is set to radians(45.0f) which creates a proper field of view. The aspect ratio is set to (GLfloat)800/600 which matches the screen size. The near clipping lane is 0.01f since it cannot be 0. The far clipping plane is set to 100.0f. This sets up the camera's properties.

3.2 Camera.h

Camera.h needs the view matrix function GetViewMatrix. This function can simply return a lookAt() call. The function lookAt() has an eye, center, and up vector. The eye vector is set to Position, center is set as Position + Front to define where the camera looks, and the up vector is set to Up which defines the upwards direction for the camera. This finished the camera's set up.

3.2 phong.vs

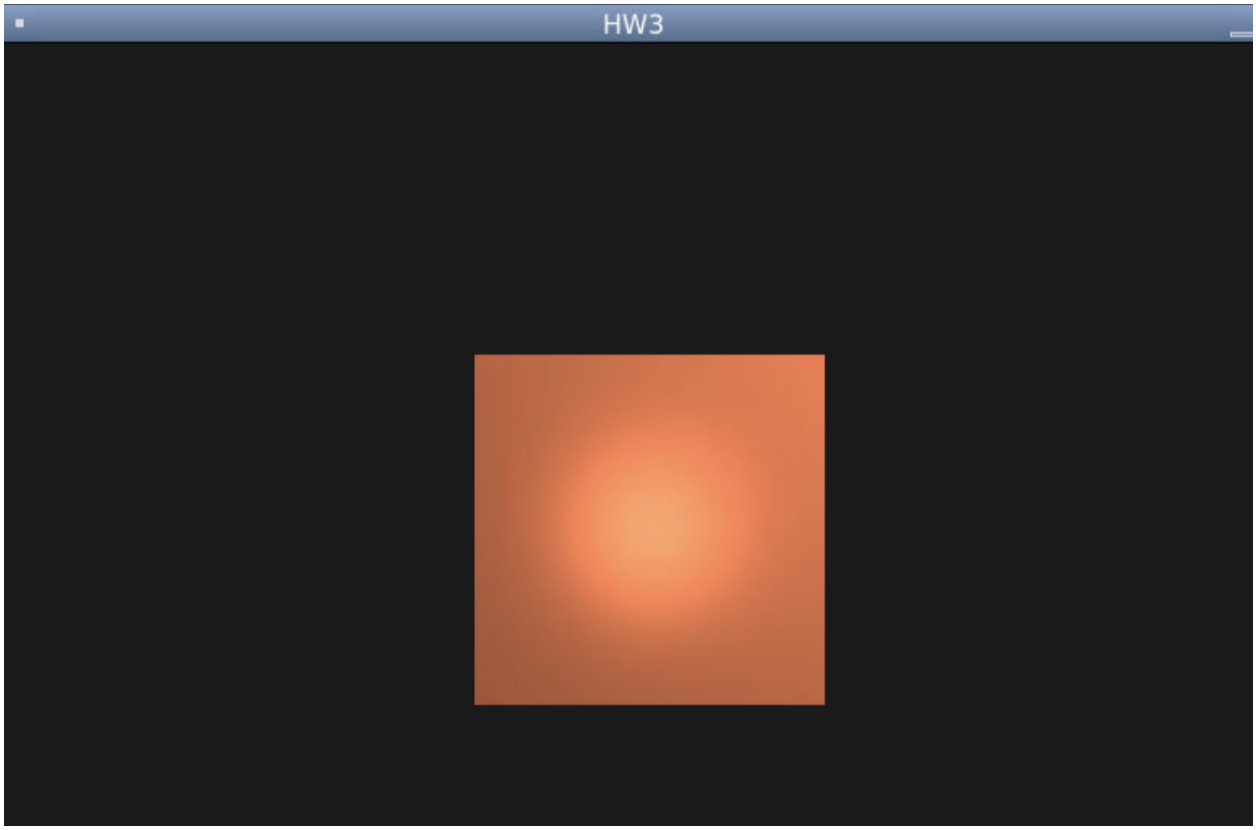
In phong.vs, the vertex shader is made which handles individual vertices. FragPos sets up the fragment's position, which is for pixels. FragPos is set to vec3(model*vec4(position,1.0)). gl_Position will hold the position of the vertex, this is set to projection*view*model*vec4(position,1.0). Normal is also set in this file, which sets up normal vectors for the diffuse lighting calculation, this is set to mat3(transpose(inverse(model)))*normal to have the light properly reflect from viewing angles.

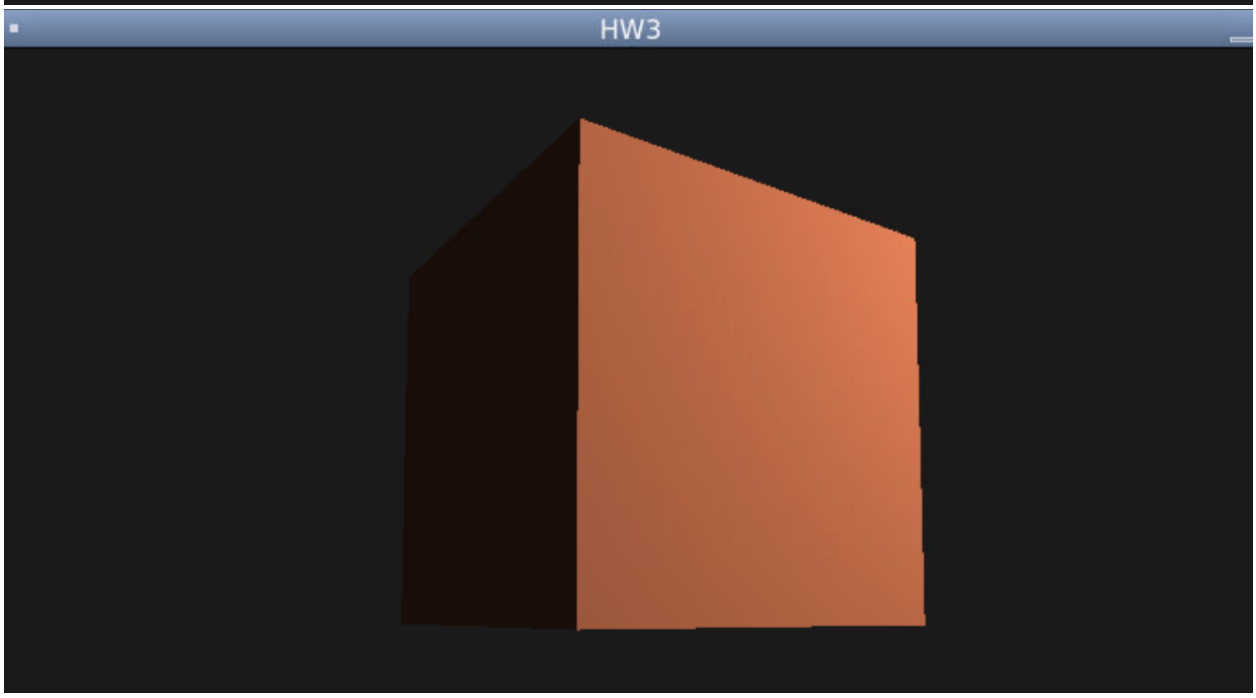
3.2 phong.frag

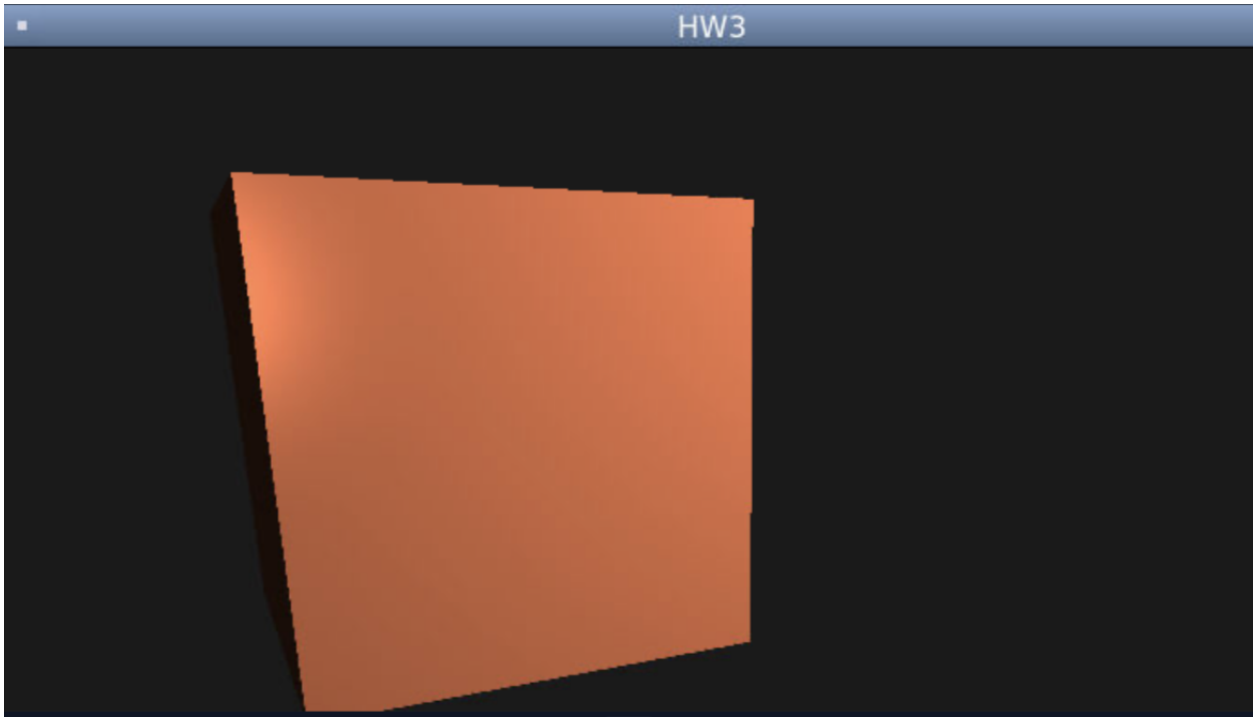
Phong.frag is where everything comes together for the Phong shading model. Ambient light is calculated first, which is just a constant value 0.1*lightColor to create an ambient lighting effect. Diffuse lighting has a few more parts, which is diffuse * lightColor. Diffuse is equal to max(dot(normal, directionLight, 0.0) which will simulate the matte reflection from the model. DirectionLight is a variable that is set to normalize(lightPos - FragPos) to calculate the light direction. Lastly is the specularLight which is set to 0.5 * specular * lightColor, this simulates the sharp reflective highlights on the model. Specular is a variable which calculates how the lighting will be simulated, this is equal to pow(max(dot(fromViewDirection, reflectionDirectionObj), 0.0), shine). FromViewDirection is normalize(viewPos - FragPos) and reflectionDirectionObs is reflect(-fromViewDirection, normal). Shine is how reflective or bright the specular lighting will look. These three variables are derived from the Phong shading model formula. The last step to finalize the lighting is by setting color to vec4((ambientLight + diffuseLight + specularLight) * objectColor, 1.0f) which sets the shading onto the model.

4 Results

The results require a script to be ran in order to create a 3d environment where the model can be viewed with its applied Phong shading model.


HW3


HW3


HW3