Brett Stelly

Direct Lighting

**Diffuse Implementation**

To calculate the diffuse light, you will need to create a uniform variable in the fragment shader. So, in the fragment shader, declare a uniform variable of type vec3 and call it “lightDirection”. In order to send information to this uniform variable, we need to create a handle and send information to the handle from the rendering program.

In the rendering program, create a variable of type int. This will be the handle for the light’s direction. Initialize the handle using glGetUniformLocation and pass in the shader’s program and the name of the uniform variable that was created in the fragment shader.

Next, create a variable of type vec3. This variable will be the light’s direction. Initialize this variable with a vec3.

Once the handle and the variable for the light’s direction are created, we must now send the information to the fragment shader. To do this, use the function glUniform3fv and pass in the handle as the first argument and the reference to the first index of the variable.

Now that the information is sent to the fragment shader, we can do the calculations to get the diffuse lighting. In the shader file, create a vec3 variable and call it “vertexNormal”. Initialize this variable by calling normalize and pass in the vPosition.xyz. Next, create a variable of type float and call it “lambert”. This variable will be the Lambertian term. Assign this variable by calling the dot function and pass in -lightDirection and vertexNormal. Finally to calculate the diffuse term, create a new variable of type vec4. This variable will be assigned vColor multiplied by lambert multiplied by lightColor.

To add the diffuse to the FragColor multiply vColor by the sum of ambient plus diffuse.