IMAT2908 CW: Lighting

|  |  |
| --- | --- |
| **Full Name: Akash Rai** | **P-Number: P2529627** |
| **GitHub Username: Jetsam1** | **Word Count:1300** |

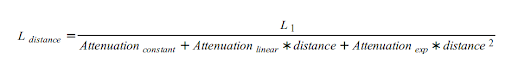
## Planning

For this assignment, we had to use phong shading to add lighting to a scene of a teapot on a green plane in a grey background. My plan for this coursework was to introduce ambient lighting, specular lighting, and diffuse lighting into the scene through the phong.frag file included in the GitHub repository. I wanted to shade the teapot in a way that lit the nozzle end but would slowly fade out as it got to the handle. This is because it would show a level of intricacy to the lighting techniques used to achieve such an effect.

## Implementation

The first thing I did to implement the shading was add a struct to the .frag file called reflectiveness. This is because the variables in this struct will show the reflective properties of any material included, for this reason the uniform for reflectiveness was called material. Within this struct were 3 vec3’s called Kd, Ks and Ka as well as a float called shine. The 3 K’s were used to alter the reflective properties of each type of light denoted by the second letter meaning that Kd was the reflectiveness of the diffuse light, Ks was the reflectiveness of the specular light and that Ka was the reflectiveness of the ambient light. The shine value would then provide a value for the shininess of the scene when used in the scenediffuse.cpp.

The next implemented struct was that of LightIntensity which contains 3 vec3s named Ld, Ls and La standing for diffuse light, specular light, and ambient light respectively. These are values that will be between 0 and 1 and will contain the strength of each type of lighting used on different surfaces. Due to its use of denoting the intensity of the different lighting methods, the function call for it was called light.

Following this was the calculation of attenuation using the formula[1]

The formula was implemented by introducing a float function called Attenuation which takes a vec3 as an argument which would be the light ray its calculating. The function itself contains 4 values called aC, aL, aQ and dist standing for constant attenuation, linear attenuation, quadratic attenuation, and distance respectively. These values would then be returned because of the formula shown in [1], through returning (1/aC+(aL\*dist) +(aQ\*dist\*dist)) where the distance is calculated using the function length, taking the light as an argument. The effect that this would have on the final teapot would be that it would allow the teapot to brighten and dim according to distance is changed.

Following on from the attenuation function are the lighting functions which are ambient() specularLight(camPos, reflectedVector) and diffuseLight(L). these 3 functions are the lighting functions that carry out the lighting commands. The ambient() function returns the material ambient reflection multiplied by the ambient light intensity. This is so that the ambient light emitted off objects is tuned according to the object emitting the light. The specularLight function which takes in the camera position and the reflected vector of the light. Within the function itself, there is a vec3 calculation for the intensity of the light and follows the same calculations as the ambient light but using the specular variables rather than the ambient variables. The next calculation within the function is a calculation finding the dot product of the reflected vector of the light and the camera position to work out the angle at which the light has to project. The function then returns the intensity to the power of the strength of the light multiplied by the angle to the power of the shininess of the object. The diffuse lighting function takes a vec3 as an argument, this vec3 is the light ray. The function itself is a vec4, this is to homogenize the vectors due to the relative position of the teapot constantly changing throughout the scene, the other lighting values remain at vec3 due to the calculations not changing as such when the camera is moved. The function includes a vec4 variable called Id which is a homogenized vector of the light intensity of the diffuse light multiplied by the dot product of the normal vector and the light vector. We then clamp this variable to cast the lighting rays from the object itself. The function returns the material diffuse reflectiveness (as a homogenized vector) multiplied by the clamped vector.

The main function within the frag file was the next thing I tackled, and it starts by normalizing the light vector by going from the vertical position to the light position (b-a vector). Following on from this, I declared a float called attenuation and made it equal to the Attenuation function whilst passing in the normalized light vector as the argument. Following on from this, the ambient light was set using a new vec3 equal to the ambient function. New variables to introduce the diffuse lighting and the specular lighting. The diffuse light function takes the normalized light vector as an argument whereas the specular light function takes 2 new variables as arguments, one of which is the camera position and the other being the reflected ray. The camera position is calculated in a similar way to which the light ray is calculated but this time using the eye position rather than the light position. As for the reflected ray, its calculated using the reflect function within openGL and passing in the negative normalized light ray and the normal vector. The final line of the main sets the frag colour which is the vec4 that is the phong shader used withing the scenediffuse.cpp to implement the lighting. This value was set to be the sum of the diffuse lighting plus homogenized vectors of both ambient light and specular light which are both multiplied by the attenuation values.

After completing the main within the .frag file, I moved onto the scenediffuse.cpp to implement the lighting changes. In the function setLightParams, the lighting settings were added using prog.setUniform with the light type under the uniform set in the frag file and the appropriate sub lighting to what the light was, alongside the vec3 values of the light, an example of this would be prog.setUniform("light.Ld", 0.9f, 0.9f, 0.9f) which is the setting for diffuse lighting. The light was then set into a predefined vec3. Following this was the changes to the render function which consists of setting the material reflectiveness in a similar way to how light intensity was done in the setLightParams function and then render that object; this is done through setting the light reflectiveness using the material uniform set in the .frag file and then calling for the object to render then setting values for the teapot and then calling for the teapot to render. The all values apart from the shininess were pre-defined.

I then decided to implement toon shading using its own function that will take in the vec3 of the light ray as an argument. The first thing done creating a new vec4 variable and then creating a float calculated by doing the dot product of the normal and the light ray. This then corresponds to a value range that will be met by one of 5 if statements that will change the strength of the light accordingly by using the appropriate variable value in the return calculation. In the main, the function uses the same conditions as diffuse light, but would not work alongside the diffuse light function so it was commented out.

## Conclusion

The product of this was a teapot with toon shading using phong lighting techniques which achieved the initial goal of having the nozzle end of the teapot lit whilst leaving the handle end less lit. I found that the technique that advanced this the most was the toon shading as the colour gradient gradually decreases both on the plane and the teapot.

## References

Use IEEE style (does not count towards 2000 words limit)

[1] <http://ogldev.atspace.co.uk/www/tutorial20/tutorial20.html> openGL dev tutorials attenuation

Accessed 03/05/21.