Project 6: Specular Lighting

Jonathon Moore

CST-310

College of Engineering: Software Engineering

Professor Citro

GitHub Link: <https://github.com/hydrenoid/CST-310>

Project *6*: Specular Lighting

1. Project Explanation

This OpenGL program aims to illustrate fundamental lighting effects using shaders. By employing OpenGL techniques, we render 3D cubes with varying shininess and light interaction. We use transformation matrices such as translation and rotation to position eight 3D cubes, showcasing different lighting effects. Matrix operations facilitate transformations and data passage to shader programs, which are compiled and linked using vertex and fragment shaders tailored for distinct lighting effects and rendering tasks. To display each cube's shininess value, we utilize FreeType for text rendering, a library designed for font handling and text display. The main rendering loop encompasses all rendering processes, with each cube featuring unique lighting conditions and transformations. This project comprehensively explores OpenGL facets, including shader programming, vertex buffer objects, and text rendering, to demonstrate 3D lighting effects. On top of this usability has been added by key input to adjust the shininess value of one of the cubes to allow for real time feedback on how the object is being affected by different shininess values.

1. Specular Lighting

Some of the main changes that I made to the shader was adding a uniform variable that is set in the shader that will act as the shininess value. This allows the programmer to only use one shader for all of the cubes rather than having to have a different one simply for each shader value, while also allowing for the manipulation of an object’s shininess value within your code as the program is running. Specifically focusing on the specular reflection, it represents the bright spots on a surface where the light is directly reflected towards the viewer. These bright spots, also known as specular highlights, are what give objects a shiny or glossy appearance. The shininess value controls the size and intensity of these highlights.

1. Mathematical Explanation

In OpenGL shaders, the calculation of shininess in the Phong reflection model involves several geometric and linear algebraic concepts. The specular reflection component is calculated based on the angle between the reflected light direction and the view direction. This angle is determined using the law of reflection, which states that the angle of incidence is equal to the angle of reflection. To compute this angle, we first need to calculate the reflection vector, which represents the direction in which light is reflected off the surface. The reflection vector R can be calculated using the incident light direction L and the surface normal N as follows:

Next, we calculate the view vector V, which points from the surface point towards the viewer. This vector is calculated as the negative of the vector from the surface point to the camera position. The specular reflection intensity is then calculated using the reflection vector RR, the view vector V, and the shininess factor shininess as:

In this equation, the dot product (R⋅V) measures the alignment between the reflection and view vectors. Higher values indicate that the light is more aligned with the view direction, leading to a stronger specular highlight. The shininess factor shininess controls the size and sharpness of the specular highlight, with higher values producing smaller and more focused highlights.

Resources

*Welcome to OpenGL*. Learn OpenGL, extensive tutorial resource for learning Modern OpenGL. (n.d.-a). https://learnopengl.com/

Wikimedia Foundation. (2024, April 4). *Phong reflection model*. Wikipedia. https://en.wikipedia.org/wiki/Phong\_reflection\_model