Justin Dietrich, Ryan Scott
10/30/2022
CST - 310
Prof. Citro

Project 6

Specular Lighting, Objects, Illumination and Shaders

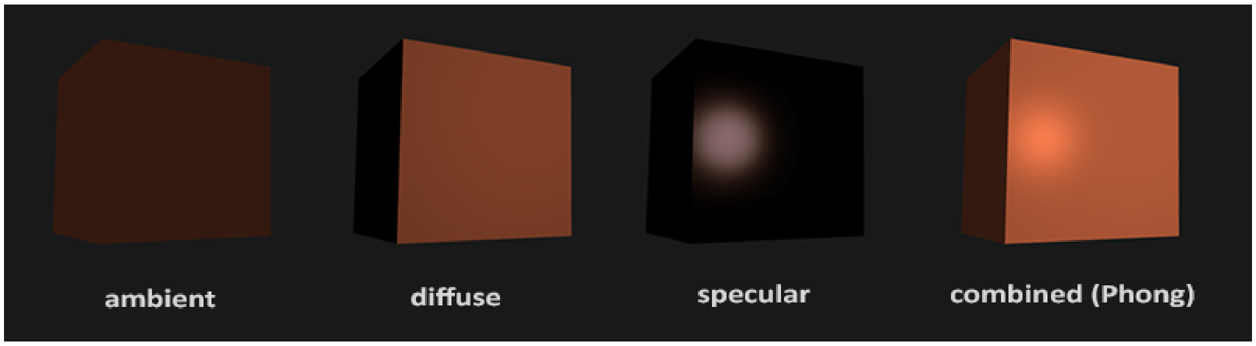
Documentation:

1. **Theoretical background, mathematical concepts, programming concepts, aesthetic decisions, etc.**

As lighting is extremely complex and depends on so many factors, we cannot calculate it in the real world due to our limited processing power. OpenGL lighting is based on simplified models that make it easier to process and are very similar to the reality of light, so it is an approximation of it. The Phong lighting model is one of many models based on our understanding of light's physics.

A **Phong** lighting model has 3 major components: *ambient*, *diffuse*, and *specular* lighting.

Below is how each component looks alone + combined:



2. **All the shininess values explained and added in the code.**

Each box has its own shininess value in its shader file. I have it so that there is a uniform integer that can be set in the main file. The shininess value is used in the calculation of the specular lighting color. It is seen in this section of the shader code:

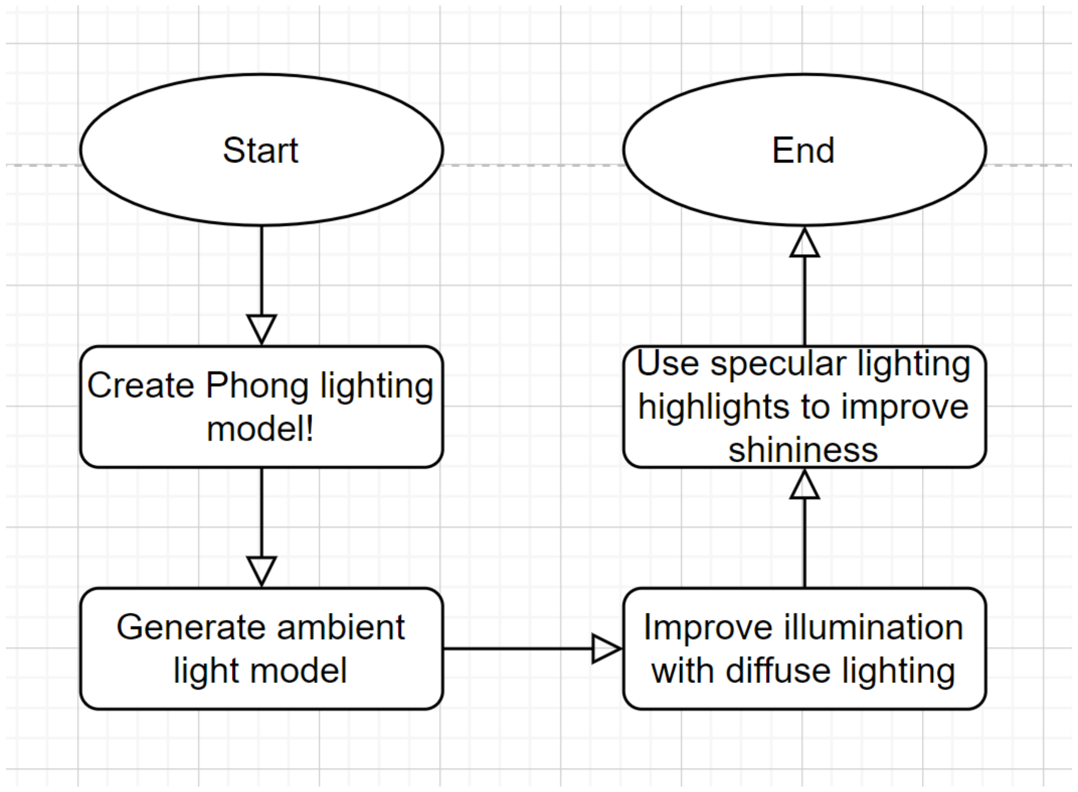
```
float spec = pow(max(dot(viewDir, reflectDir), 0.0), shininess);
    vec3 specular = specularStrength * spec * lightColor;

    vec3 result = (ambient + diffuse + specular) * objectColor;
```

We use the dot product of the viewing angle and the face angle and raise it to the power of our shininess value. With a higher shininess value, you get a more accurate representation of the light that is being reflected.

3. **A detailed algorithm (flowchart) depicting the logic of your approach to generating the cubes with different shininess.**

Here is some pseudocode to give an idea of how the cubes are generated.

```
Generate cube vertex data;
For (each cube to be placed)
{
        Generate shader with shininess value; (shininess = 2^(i+1))
        Create cube model with shader and vertex data;
        Translate and rotate cube appropriately with glm;
}
Generate light;
```


Start
End
Create Phong lighting model!
Use specular lighting highlights to improve shininess
Generate ambient light model
Improve illumination with diffuse lighting

4. **A detailed algorithm for meshes, illumination, and shininess that implements each cube after a query is entered.**

Meshes

- The ambient lighting model is a simplified approach to global illumination with a less expensive algorithm.

- The addition of ambient lighting to a scene is very simple. To do this, we take the color of the light, multiply it by a constant ambient value, multiply that with the object's color, and use the result as the color of the fragment in the cube's shader.
- Cube vertex data is entered at the beginning of the code. This data is used in the creation of each cube model. Each model is translated and rotated to the appropriate spot using glm.

Illumination

- While ambient lighting alone doesn't produce the most interesting results, diffuse light will start giving the object a significant visual impact.

- When a light source is aligned close to an object's fragments, diffuse lighting gives it more brightness.

- To accomplish this, the vector that is perpendicular to the vertex' surface must be used in conjunction with a directed light ray.

Shininess

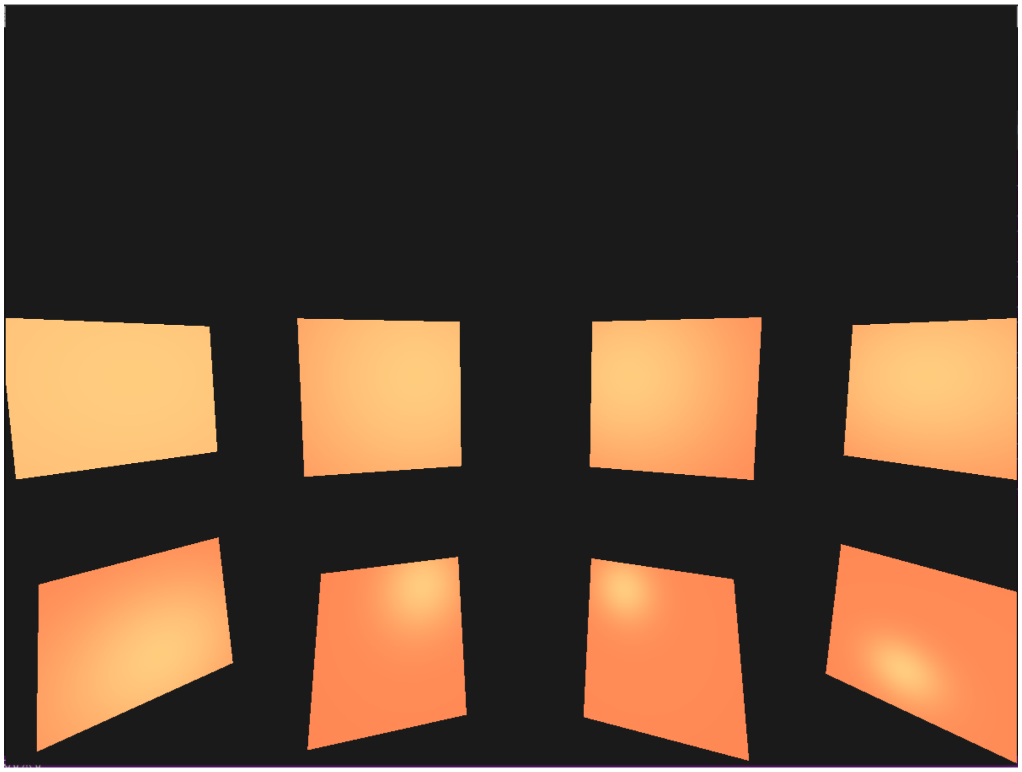
- The shininess of a cube comes from its specular lighting highlights.

- Using the normal vector, we calculate the reflection vector. We then calculate the angle between the reflection vector and the view direction. When we look at the light reflected from the surface, we see a bit of a highlight due to the close angle between them.

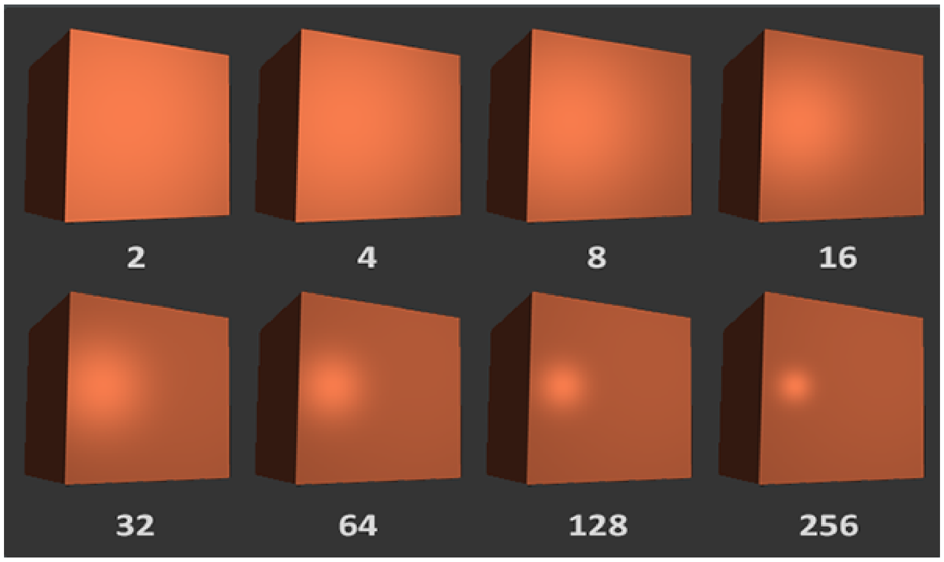
- We first calculate the dot product between the view direction and the reflect direction (ensuring it is not negative) and then raise it to powers of 2. An object with a higher shininess value will reflect light more realistically instead of scattering it around, thus resulting in a smaller highlight.

Starting the code will bring you to this screen:

Moving the camera with WASD and arrows, I was able to get this image.



The cubes get more shiny from top left to bottom right. In order to get a reflection from a single light source on all cubes, I had to rotate and move them around from their positions in the target image. The target image looks like to result of multiple different programs put together, and would not be possible to remake in a single run. Our cubes are arranged in a quartercircle around the light source so you can see a reflection on all of them.. You can see that they get shinier, but it’s not as defined since the light source is probably farther away than the target image.



Sources:

https://learnopengl.com/Lighting/Basic-Lighting