# An Exploration of Lighting Techniques in 3D Graphics Using OpenGl

Luca Napora

# Contents

**Abstract**

This paper explores various lighting techniques related to computer graphics by building several three-dimensional objects using C++ and OpenGL. The significant components of light in computer graphics, namely ambient, diffuse, and specular lighting, are implemented following the Phong Shading Model and the Gouraud Shading Model. These outputs are compared and analyzed to determine the best approach for simulating light in a three-dimensional environment.

# 1 Introduction

## 1.1 Project Goals

Computer graphics as a field is extensive. Many different and interconnecting components, like lighting, shadows, materials, and textures, need to be implemented in specific and complicated ways to create the desired scenes. This project is designed to take one of those components, lighting, and implement it in a three-dimensional environment to understand better how computer graphics work and lessen the daunting nature of the field. Furthermore, this project aims to analyze two commonly-used lighting models, the Phong Shading Model and the Gouraud Shading Model, to determine the strengths and weaknesses of both.

## 1.2 What are 3D Graphics?

In a broad sense, computer graphics refer to the representation of visual content on a computer. Computer graphics can be divided into two categories: two-dimensional and three-dimensional. 2D graphics center around the creation of images on a two-dimensional plane, like digital images, paintings, and drawings. 3D graphics involve the creation of images that have depth in addition to length and width and can be viewed from various angles. Both 2D and 3D graphics are commonplace in society. Special effects in movies and television, architectural models, and medical imaging are examples of computer graphics that appear daily. This paper will focus on 3D computer graphics and, more specifically, how to implement light effectively in three-dimensional environments.

## 1.3 The fundamental principles of 3D graphics

In computer graphics, light is a central component that allows one to express shape, form, and depth [6]. Several principles of computer graphics work in conjunction with each other to provide these effects, such as color, shadows, and reflection. Color is made up of three characteristics, hue, brightness, and saturation. Hue refers to the actual color of an object, such as red, blue, or yellow. Brightness is the amount of light an object reflects, while saturation refers to the intensity of the hue [5]. Shadows allow one to give depth and realism to a scene [9]. Shadows are created when an object blocks light. The shape and size of shadows can be used to convey information about the object's shape and position [9] Finally, reflection refers to the light that bounces off of an object and is reflected into a scene. It is used in graphics to give things the appearance of shiny or reflective surfaces [7].

# 2 Lighting Techniques

## 2.1 Light Sources

Lighting plays an essential part in computer graphics, especially when trying to create an accurate representation of the world. In real life, light consists of photons emitted from high energy sources that bounce around until they reach our eyes [3]. However, replicating this process on computers would require simulating an impossibly large amount of photons. Instead, lighting in computer graphics relies on using lighting models, which are mathematical algorithms used to simulate the behavior of light in a virtual 3D environment.

Most lighting models today are called "ADS" models, as they are based on three types of light: ambient light, diffuse light, and specular light [3].

### 2.1.1 Ambient Light Source

The types of light reflection mentioned earlier result from specific light sources. For example, ambient reflection results from an ambient light source, where light is constant without a direct source [10]. Instead of coming from a specific direction, an ambient light source creates light that fills an environment evenly. The lack of a direct light source makes ambient lighting

the easiest component of lighting to implement. However, objects can still block or occlude ambient lighting, casting shadows that may add complexity to the scene.

### 2.1.2 Point

Positional lighting, also known as point lighting, refers to light with a specific source. Think candles, lamps, light bulbs, etc [3]. Positional lighting is emitted uniformly in all directions from a particular point in space. The intensity of the light decreases with distance from the point source, according to the inverse square law, which states that the power of the light is inversely proportional to the square of the distance from the source [5] In addition, attenuation factors can be incorporated to illustrate the intensity of the light diminishing as the space grows [3] Positional lighting is perhaps best known for its use in video games, where different types of light sources like light bulbs and lamps are used to illuminate a scene.

### 2.1.3 Directional Lighting

Like ambient lighting, directional lighting doesn't have a light source or fixed position. However, directional lighting has a specific trajectory or direction. Directional lighting is often likened to the sun, a distant light source with a particular direction, but whose lighting is ambient once it reaches the environment. It is also computationally efficient, as the distance between the light source and the object being lit does not affect the light's intensity, simplifying the lighting calculations [5].

## 2.2 Components of Lighting

### 2.2.1 Ambient

There are many lighting components, but three major characteristics are used in most illumination models: ambient, diffuse, and specular lighting. As mentioned earlier, ambient light refers to light that has bounced off and reflected against so many surfaces that it no longer comes directly from a light source [5]. Ambient and diffuse lighting are often used together to provide a base illumination level. [5]

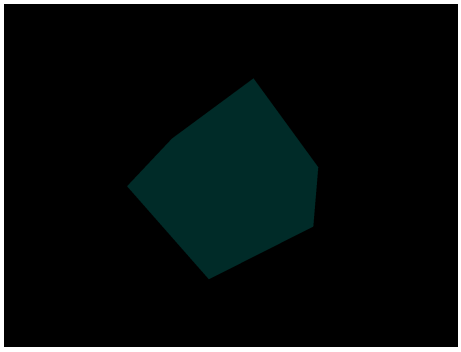

Figure 1: 3D Cube Illuminated with Ambient Light

Figure 1 illustrates a three-dimensional cube illuminated with only ambient light. As you can see, ambient light provides a constant light equal for each object fragment. With only ambient light, it is hard to distinguish the object. Figure 2 illustrates the same ambient light on a cube with a wooden crate texture. The texture and ambient light illuminate the cube to a greater degree.

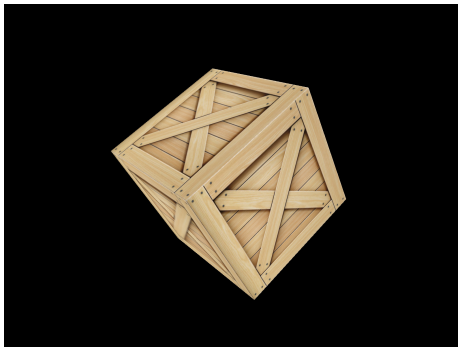

Figure 2: Textured 3D Cube Illuminated with Ambient Light

### 2.2.2 Diffuse

Diffuse lighting illustrates how light scatters and reflects off of surfaces. Diffuse lighting is often soft and even, allowing objects to have depth. Diffuse lighting is most obviously seen when the sides of an object that face the light source are brighter than those that don't. The intensity and color of diffuse lighting depend on several factors, including the light source's intensity and color, the surface's angle, and the surface material's properties. For example, a rough surface will scatter light more evenly and create a softer shadow than a smooth surface.

Figure 3: Diffuse Lighting

Figure 3 illustrates a cube with only diffuse lighting. As the light source revolves around the cube, the strength of the illumination is relative to each fragment's distance from the light source. The closer each section of the cube is, the brighter it will appear.

### 2.2.3 Specular

In contrast, specular lighting is the reflection of photons against an object in a particular direction. Specular lighting is often connected with shiny surfaces such as metal or glass, as it produces a bright reflection that creates the illusion of glossy and reflective materials. When a light source shines on a shiny surface, such as a polished metal ball, the light reflects off the surface in a very predictable way. The angle at which the light hits the surface is known as the angle of incidence, and the angle at which it bounces off is known as the angle of reflection. The intensity of the reflected light is determined by the angle of incidence, the angle of reflection, and the surface material. Incorporating specular lighting makes a scene appear more lifelike and realistic [7]

Figure 4: Specular Lighting

Specular lighting is hard to illustrate in isolation. Instead, figure 4 shows specular lighting and diffuse lighting in conjunction. Compared to just diffuse lighting in Figure 3, you can see the specular "shininess" created in the object's corner when the light source is directly above.

# 3 Illumination Models

## 3.1 Gouraud Shading

The information above has been primarily theoretical. In reality, implementing these lighting techniques relies on illumination models that calculate the values needed for each vector to create the desired lighting effect. One of the earliest techniques used in computer graphics was flat shading, which used only a single color for each side of an object, preventing the object from displaying reflected light [4]. Gouraud shading was introduced in the 1970s to allow for a more realistic appearance [4]. The general idea of Gouraud's model is to assign a color to each vertex of a polygon and then interpolate the colors across the polygon's surface. This is accomplished by calculating the color intensity at each point on the polygon using a linear interpolation of the colors at the vertices. The result is a smooth gradient of colors across the surface [4]. Because the color intensity calculations only occur at the vertices of the surface rather than every pixel on the surface, Gouraud's model is seen as computationally efficient. Also, Gouraud's model can deliver convincing shadow effects on curved surfaces, which earlier illumination models like flat shading fail to do [4]. However, the linear interpolation that this model relies on can cause bright or dark intensity streaks, called match bands, to appear on the surface of objects, especially curved surfaces [1].

## 3.2 Phong illumination model

The Phong illumination model was created in 1975 by Bui Tuong Phong to meet some of the limitations of the Gouraud model. The Phong model is based on the three components of light reflection from a surface discussed earlier: ambient, diffuse, and specular. These components are combined to calculate the color of each pixel on the object's surface [8].

Phong's model calculates elements per pixel. It is less efficient than Gouraud's model, which makes calculations per vector. However, it allows for more realistic lighting effects, as tools such as ambient, diffuse, and specular lighting can be implemented to greater effect. [8]

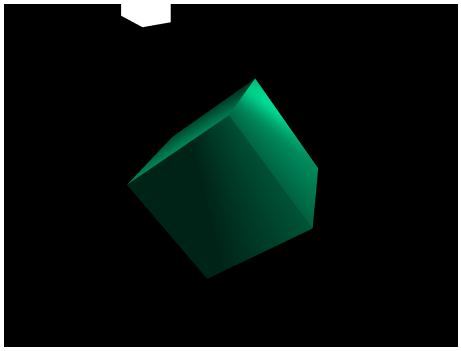

Figure 5: The Phong Shading Model

## 3.3 Blinn-Phong illumination model

The Phong model was extended in 1977 by James Blinn. Referred to as the Blinn-Phong illumination model, this extension provides a more efficient and computationally less expensive way to calculate specular highlights. The Blinn-Phong model uses an approximation to the half-angle vector instead of the reflection vector used in the Phong model to estimate the specular reflection component. This approximation reduces the number of trigonometric functions that need to be computed, making the model faster and more efficient.

Compared to the Phong model, the Blinn-Phong model produces more realistic and visually appealing results with less computational overhead. It also works well with various materials, including rough and non-metallic surfaces. However, it still has some limitations, such as not accounting for subsurface scattering or other complex light interactions [2].

In the 1990s, ray tracing was introduced, which traces the path of each ray of light as it interacts with objects in the environment. Since then, global illumination, real-time rendering, and physically-based rendering have also expanded the possibilities of conveying accurate lighting in 3D computer environments.

## 3.4 Tools

This project is centered on using OpenGl, an application programming interface (API) designed to help create two-dimensional and three-dimensional computer graphics. OpenGL was implemented using C++ and several libraries, including GLFW and GLAD. GLFW (Graphics Library Framework) is an open-source library used most often for creating OpenGl windows and handling user input. GLAD, or GL ADapter, is a library that provides an efficient way to load OpenGL functions. An open-source image-loading library called stbimage.h was also used to integrate images of any type into the project. This was used when making the textured box shown in Figure 2.

## 3.5 Procedure

### 3.5.1 Object Creation

The first step of the software project was to create an OpenGL window and make it the current context. This required the utilization of GLFW. The result is shown in Figure 6.

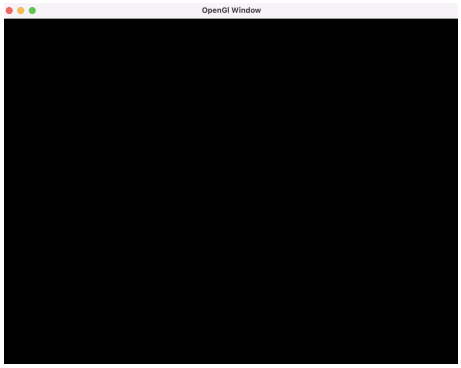


Figure 6: The OpenGL Window

As this project aimed to use a three-dimensional object to illustrate the different lighting techniques available in computer graphics, two objects were needed: the cube in the middle of the scene and an object that casts light while revolving around the cube. Creating objects with OpenGl can be a tedious process and requires the introduction of shaders, which specify how the object will be rendered. Here are the steps necessary to create these objects:

1. Define the object's vertices, indices, and other attributes. To simplify the process, I

used a cube shape for both the central object and the light source.

2. Create a Vertex Buffer Object (VBO) to store the vertex data and additional attributes.

3. Create a Vertex Array Object (VAO) to store the layout of the vertex data.

4. Define the vertex shader: The vertex shader is a program that runs on the GPU for each vertex of the object. It takes the vertex data as input and performs transformations such as scaling, rotation, or translation. The vertex shader was written in GLSL and passed to the program as a string. GLSL is similar to C-language but tailored for use with computer graphics.

5. Define the fragment shader: The fragment shader is a program that runs on the GPU for each pixel of the object. It determines the color and other properties of each pixel. The fragment shader was also written in GLSL.

6. Compile the shaders: The vertex and fragment shaders are compiled into binary code that can be executed on the GPU.

7. Link the shaders: The vertex and fragment shaders are linked to form a shader program. For this project, the central cube and the light cube had different shader programs.

8. Bind the shader program: Bind the shader program to the current OpenGL context.

9. Set up the shader attributes: Specify how the vertex data is laid out in the VAO and how it is used in the shader program by calling the glVertexAttribPointer and glEnableVertexAttribArray functions.

10. Draw the object: Draw the object by calling the appropriate OpenGL drawing function, such as glDrawElements or glDrawArrays.

### 3.5.2 Implementation of the Phong Shading Model

Ambient lighting was the first and most straightforward type of light to implement. Because it is a constant for all parts of an object, it can be done in a relatively short number of calculations. First, you choose a color for the light. Then you multiply it by a small constant ambient factor, multiply that with the object's color, and pass it to the fragment shader, where it is added to the final color of the object.

Diffuse lighting requires more calculations. Because the brightness of diffuse lighting

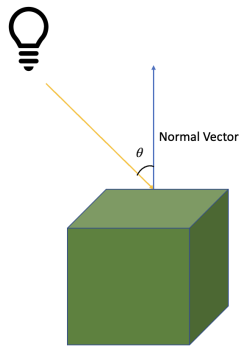


Figure 7: The Diffuse Lighting Calculations

depends on the light source's proximity to the fragment, it is necessary to measure the angle between the light ray and the fragment. The most popular way to do this is what is known as the Lambertian model, which says that the amount of light reflected from a surface is proportional to the cosine of the angle between the surface normal and the direction of the incident light. A normal vector, or a vector perpendicular to the fragment's surface, is used to do this. The normal vector is the blue arrow in Figure 7. The light ray that is depicted as a yellow line in Figure 7 is found by calculating the difference between the light source's position and the fragment's position. The strength of the diffuse lighting is then found by taking the dot product of the normal vector and the light ray [5].

With ambient and diffuse lighting implemented, the final component of the Phong Shading Model to be incorporated is specular lighting. Specular lighting is calculated much like diffuse lighting, but it considers where the viewer is. Think, for example, of a car on a sunny day. The shiny reflection that is formed on the car's surface changes in size based on where you stand. In much the same way, specular lighting is strongest wherever the viewer sees the light reflected on the surface. Figure 8 illustrates how this is done. A reflection vector is formed by reflecting the light direction around the normal vector, creating the orange line you see. Then the angle between the reflection vector and the viewer is found. The smaller the angle, the more intense the strength of the specular lighting. Again a dot product is used. This time it is between the viewer's direction and the reflection vector. The specular calculation is then joined with the ambient and diffuse components to complete the Phong Shading Model.

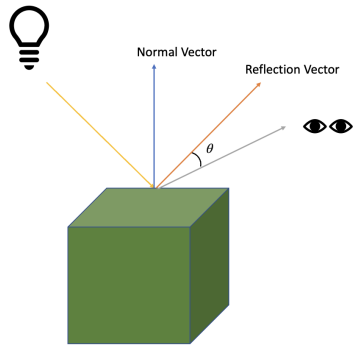


Figure 8: The Specular Lighting Calculations

### 3.5.3 Implementation of the Gouraud Shading Model

The Gouraud Shading Model works much like the Phong Shading model, except instead of using the surface normals when performing calculations, it uses vertex normals. This means most of the work is done in the vertex shader instead of the fragment shader. An example of the Gouraud Shading Model is shown in Figure 9.

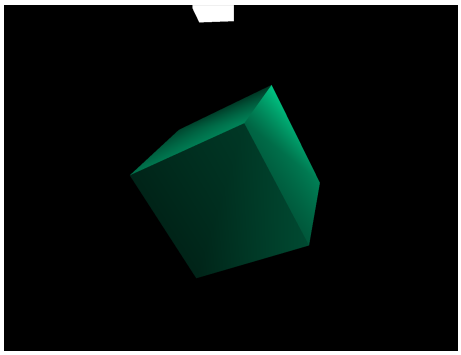

Figure 9: The Gouraud Shading Model

# 4 Analysis

The Gouraud Shading Model and the Phong Shading Model produce very similar outputs. Both models provide a simplified way to represent the millions of photons that interact with each other in real life to produce light. Furthermore, both models are formed from a base of three components: ambient, diffuse, and specular light. However, there is a small difference

in the specular lighting between each model. Figure 10 shows an example of the Gouraud Shading Model with the specular lighting strength set to 5.0 instead of the 1.0 strength that is shown in Figure 9.

As you can see, a visible "stripe" runs through a side of the central cube. This is a result of linear fragmentation, which occurs because the interpolation of colors between the vertices of the triangles that form each side of the cube is done in a linear manner [9]. In other words, color values are calculated by taking the weighted average of the colors at the vertices, and this can lead to the visible color banding that you see in Figure 10. Linear fragmentation is a downside of the Gouraud approach and one of the improvements that the Phong Model was created to address. However, this comes at the expense of efficiency. The Phong model requires additional calculations for each pixel to calculate the lighting and shading values based on the angle of incidence of the light source. As a result, it is generally more computationally expensive than the Gouraud model.

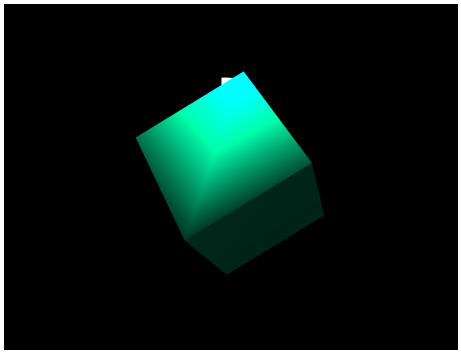

Figure 10: Gouraud Model with Intense Specular Light

# 5 Conclusion

The Phong Shading Model and the Gouraud Shading Model are both realistic and accurate approaches for implementing light in a three-dimensional environment. While the Phong Model is less expensive, it reduces the risk of linear fragmentation compared to the Gouraud model. Future work could include adding lighting maps so that the light behaves differently depending on the fragment's texture and adding light casters.

# References


[1] Gouraud Shading in Computer Graphics, Oct. 2022. Section: Computer Graphics.

[2] J. F. Blinn. Models of light reflection for computer synthesized pictures. *ACM SIGGRAPH Computer Graphics*, 11(2):192–198, July 1977.

[3] V. S. Gordon and J. L. Clevenger. *Computer Graphics Programming in OpenGL with C++*. Mercury Learning & Information, Bloomfield, UNITED STATES, 2021.

[4] H. Gouraud. Continuous Shading of Curved Surfaces. *IEEE Transactions on Computers*, C-20(6):623–629, June 1971. Conference Name: IEEE Transactions on Computers.

[5] E. Haines, N. Hoffman, and T. Akenine-Mo¨ller. *Real-Time Rendering, Fourth Edition*. A K Peters/CRC Press, Boca Raton, 4th edition edition, Aug. 2018.

[6] J. Hughes, A. v. Dam, M. McGuire, D. Sklar, J. Foley, S. Feiner, and K. Akeley. *Computer Graphics: Principles and Practice*. Addison-Wesley Professional, Upper Saddle River, New Jersey, 3rd edition edition, July 2013.

[7] S. Marschner and P. Shirley. *Fundamentals of Computer Graphics*. A K Peters/CRC Press, Boca Raton, 4th edition edition, Dec. 2015.

[8] B. T. Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317, June 1975.

[9] A. Watt. *3D Computer Graphics*. Addison-Wesley, Harlow, England ; Reading, Mass, 3rd edition edition, Dec. 1999.

[10] S. Zhukov, A. Iones, and G. Kronin. An ambient light illumination model. In G. Drettakis and N. Max, editors, *Rendering Techniques '98*, Eurographics, pages 45–55, Vienna, 1998. Springer.