Reporte de Práctica de Laboratorio de Computación Gráfica e Interacción Humano-Computadora

Información de la práctica

Nombre del alumno:	Martínez Pérez Brian Erik		
Número de Cuenta:	319049792	Práctica número:	7
Título de la práctica:	Materiales e Iluminación I en OpenGL		
Fecha de entrega:	20 de octubre de 2025		

Introducción: (descripción inicial sobre la práctica)

Esta práctica implementa la interacción física de la luz con las superficies de los objetos tridimensionales para aumentar el realismo de la superficie de modelos 3d, para ello se deberá crear una fuente de luz artificial tomando en cuenta Modelo de Iluminación de Phong, y las propiedades de los materiales. Además, se realizarán pruebas en distintos modelos, para observar cómo afecta visualmente la superficie de los cuerpos ante distintos ángulos de la fuente de iluminación.

Para ello se proporciona un código que ya implementar una fuente de iluminación, la cual se deberán de cambiar sus parámetros para verificar los efectos que se producen en distintos modelos y en sus texturas que estos contienen, para ello se debe comprender el concepto de Modelo de Iluminación de Phong y todas las componentes que la conforman.

Resumen Teórico: (introducción sobre los fundamentos teóricos en los que se basa la práctica y cuáles son los objetivos teóricos que persigue.)

Modelo de Iluminación de Phong es un modelo que descompone la luz en tres componentes: Luz Ambiental (luz de fondo), Luz Difusa (color principal) y Luz Especular (brillo/reflejo).

El componente *ambiental* representa la luz indirecta o de fondo que ha sido dispersada por el entorno y que, por lo tanto, ilumina todas las superficies de manera uniforme.

El componente *difuso* representa la luz que se dispersa por igual en todas las direcciones después de golpear una superficie mate. Esta es la parte que le da al objeto su color base.

El componente *especular* representa el brillo o reflejo concentrado que se ve en las superficies pulidas.

La luz "*direccional*" simula una fuente de luz que se encuentra infinitamente lejos de la escena, como el sol. La luz "*puntual*" simula una fuente de luz que irradia en todas las direcciones desde un único punto en el espacio, como una bombilla sin pantalla. La luz focal, o "*spotlight*", simula una fuente de luz que emite luz en una dirección específica dentro de un cono limitado, como una linterna o un faro de coche.

Objetivos:

- Entender el modelo de iluminación Phong y las componentes que la definen
- Conocer las diferencias entre las distintas fuentes de luz que existen (direccional, puntual y focal).
- Aplicar el modelo de iluminación Phong y las propiedades de los materiales en la creación de una fuente de iluminación.
- Visualizar los efectos de una fuente de iluminación en distintos ángulos para modelos con texturas aplicadas.

Reporte de Práctica de Laboratorio de Computación Gráfica e Interacción Humano-Computadora

Descripción de la práctica: (En esta sección se deben describir todos los pasos ejecutados durante la sesión práctica realizada en el laboratorio.)

Lo primero que realizamos fue el cargado de los archivos necesario para poder aplicar iluminación a los modelos que utilicemos, agregamos los archivos “lighting.frag” y “lighting.vs” en la sección de archivos de origen que se utilizaran como shaders. Además de agregar el código principal “Iluminacion.cpp” en la sección de Archivos de origen.

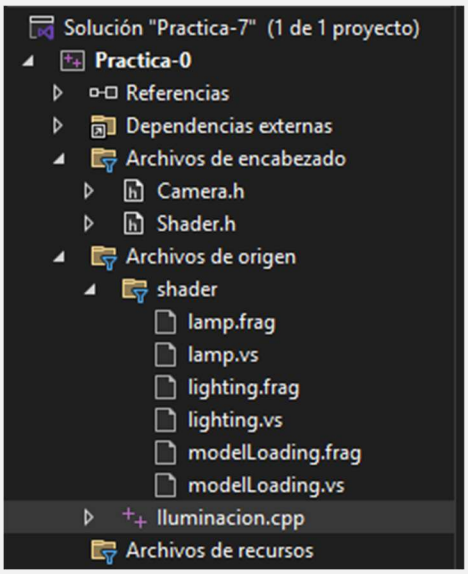


Imagen 1.1 – organización de los archivos utilizamos en la solución del proyecto.

El archivo “lighting.vs”, contiene el código que transformar las coordenadas de los vértices del modelo al espacio de recorte (Clip Space) y preparar los datos necesarios (posición, normal, coordenadas de textura) para que el Fragment Shader.

```
#version 330 core
layout (location = 0) in vec3 position;
layout (location = 1) in vec3 normal;
layout (location = 2) in vec2 texCoords;

out vec3 Normal;
out vec3 FragPos;
out vec2 TexCoords;

uniform mat4 model;
uniform mat4 view;
uniform mat4 projection;

void main()
{
    gl_Position = projection * view *  model * vec4(position, 1.0f);
    FragPos = vec3(model * vec4(position, 1.0f));
    Normal = mat3(transpose(inverse(model))) * normal;
    TexCoords = texCoords;
}
```

Imagen 1.2 – código del archivo lighting.vs

El archivo “lighting.frag”, contiene el código que calcula el color final de cada píxel en la pantalla mediante la implementación del Modelo de Iluminación de Phong para una fuente de luz.

```
#version 330 core
struct Material
{
    vec3 ambient;
    vec3 diffuse;
    vec3 specular;
    float shininess; #distancia de la fuente
};

struct Light #
{
    vec3 position;

    vec3 ambient;
    vec3 diffuse;
    vec3 specular;
};

in vec3 FragPos;
in vec3 Normal;
in vec2 TexCoords;

out vec4 color;
```

```
uniform vec3 viewPos;
uniform Material material;
uniform Light light;

void main()
{
    // Ambient
    vec3 ambient = light.ambient *material.diffuse;

    // Diffuse  angulo de la luz
    vec3 norm = normalize(Normal);
    vec3 lightDir = normalize(light.position - FragPos);
    float diff = max(dot(norm, lightDir), 0.0);
    vec3 diffuse = light.diffuse * diff * material.diffuse;

    // Specular brillo del objeto
    vec3 viewDir = normalize(viewPos - FragPos);
    vec3 reflectDir = reflect(-lightDir, norm);
    float spec = pow(max(dot(viewDir, reflectDir), 0.0), material.shininess);
    vec3 specular = light.specular * (spec * material.specular);

    vec3 result = ambient + diffuse + specular;
    color = vec4(result, 1.0f);
}
```

Imagen 1.3 – código del archivo lighting.frag

En el código principal tenemos una sección para las propiedades de la fuente de luz, se declaran variables para manipular la posición de la fuente, y también el movimiento de la propia, Vector de 3 posiciones con 3 ejes del plano cartesiano.

```
// Light attributes
glm::vec3 lightPos(0.5f, 0.5f, 2.5f);
float movelightPos = 0.0f;
GLfloat deltaTime = 0.0f;
GLfloat lastFrame = 0.0f;
float rot = 0.0f;
bool activanim = false;
```

Imagen 1.4 – código para las propiedades de la fuente de luz.

En otra sección del código mandamos a llamar y ejecutar los shaders utilizados.

```
// Setup and compile our shaders
Shader shader("Shader/modelLoading.vs", "Shader/modelLoading.frag");
Shader lampshader("Shader/lamp.vs", "Shader/lamp.frag");
Shader lightingShader("Shader/lighting.vs", "Shader/lighting.frag");
```

Imagen 1.5 – código para ejecutar los shaders.

También se utiliza “glUniform3f” para definir las propiedades de la luz, en este caso se definen las 3 componentes del Modelo de Iluminación de Phong.

```
// Set lights properties
glUniform3f(glGetUniformLocation(lightingShader.Program, "light.ambient"), 0.0f, 0.0f, 0.0f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "light.diffuse"), 0.0f, 0.0f, 0.0f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "light.specular"), 0.0f, 0.0f, 0.0f);
```

Imagen 1.6 – código para la evaluación de las propiedades de la luz.

Además, se definen las propiedades de los materiales que controlan cómo el objeto reacciona a los distintos componentes de la luz (ambiental, difusa y especular).

```
// Set material properties
glUniform3f(glGetUniformLocation(lightingShader.Program, "light.ambient"), 0.0f, 0.0f, 0.0f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "light.diffuse"), 0.0f, 0.0f, 0.0f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "light.specular"), 0.0f, 0.0f, 0.0f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "light.shininess"), 0.0f);
```

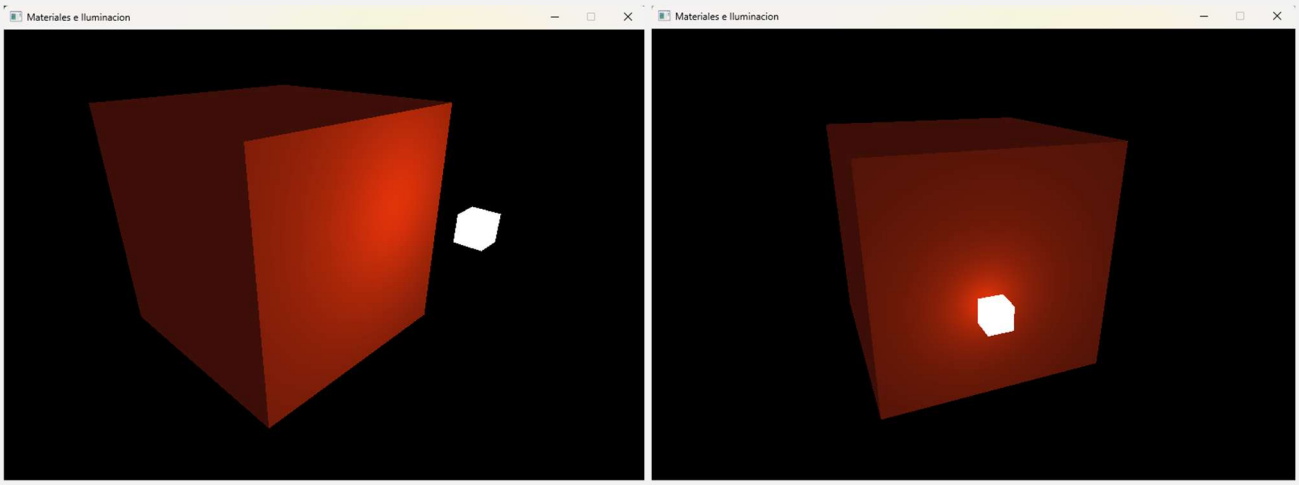
Imagen 1.7 – código para la evaluación de las propiedades de los materiales.

Probando los siguientes valores para la iluminación y el material.

```
glUniform3f(glGetUniformLocation(lightingShader.Program, "light.ambient"), 0.3f, 0.3f, 0.3f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "light.diffuse"), 0.8f, 0.7f, 0.1f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "light.specular"), 0.0f, 0.0f, 0.0f);

glUniform3f(glGetUniformLocation(lightingShader.Program, "material.ambient"), 0.5f, 0.5f, 0.5f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "material.diffuse"), 0.8f, 0.2f, 0.1f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "material.specular"), 0.5f, 0.2f,0.1f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "material.shininess"), 0.7f);
```

Obtenemos los siguientes resultados visuales de iluminación y de material sobre un cubo de color rojizo.



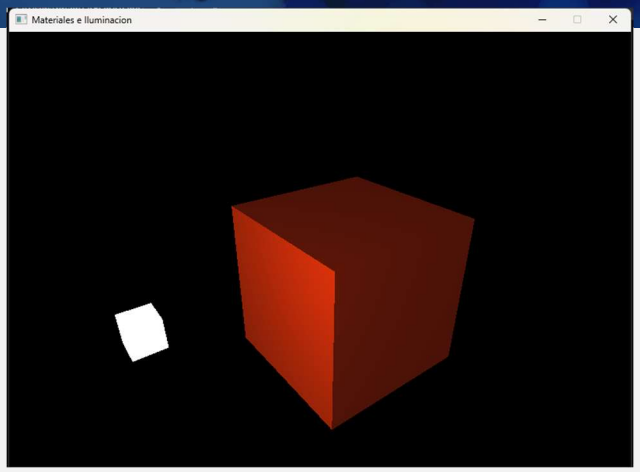
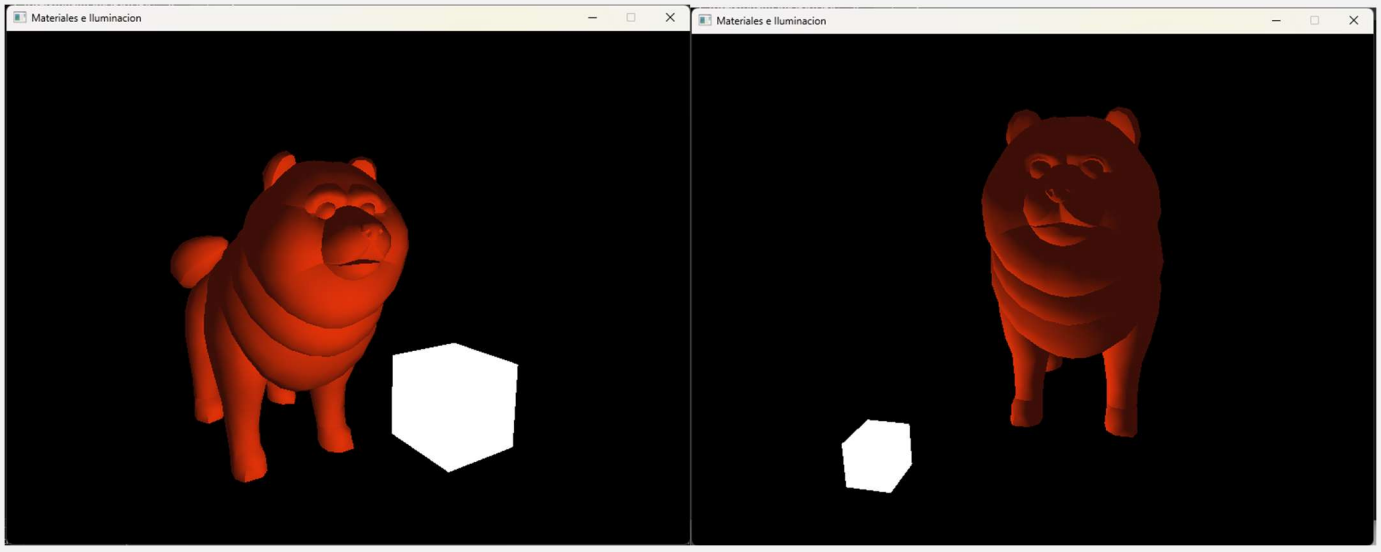


Imagen 2.1 – efectos de la iluminación sobre la superficie del cubo.

Despues reemplazamos el cubo por un modelo precargado, en este caso usamos el modelo del perro, para observar como afecta la ilumincacion a un objeto tridimensional que no tiene forma geometrica regular. Para cargar el modelo del perro se utilizo el siguiente codigo.

```
Model red_dog((char*)"Models/RedDog.obj");
glm::mat4 model(1);
model = glm::scale(model, glm::vec3(3.0f, 3.0f, 3.0f));
glUniformMatrix4fv(glGetUniformLocation(lightingShader.Program, "model"), 1, GL_FALSE,
glm::value_ptr(model));
glBindVertexArray(VAO);
red_dog.Draw(lightingShader);
```



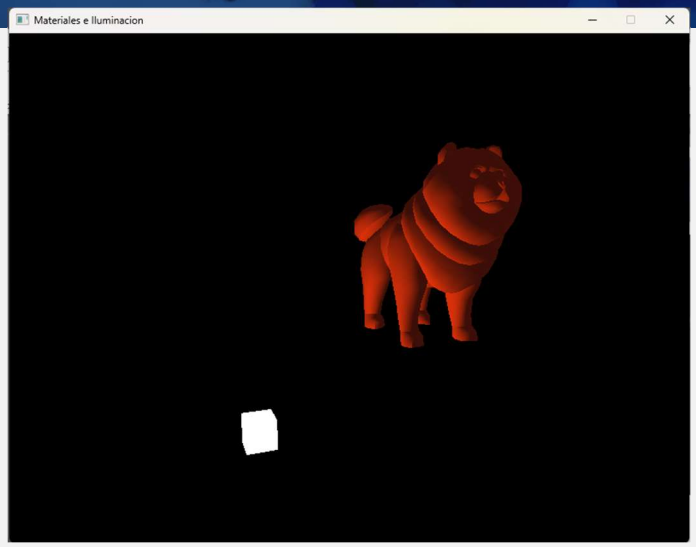
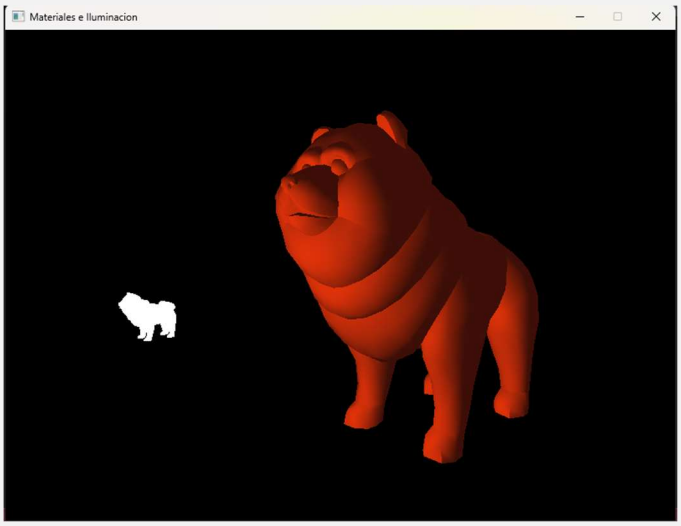
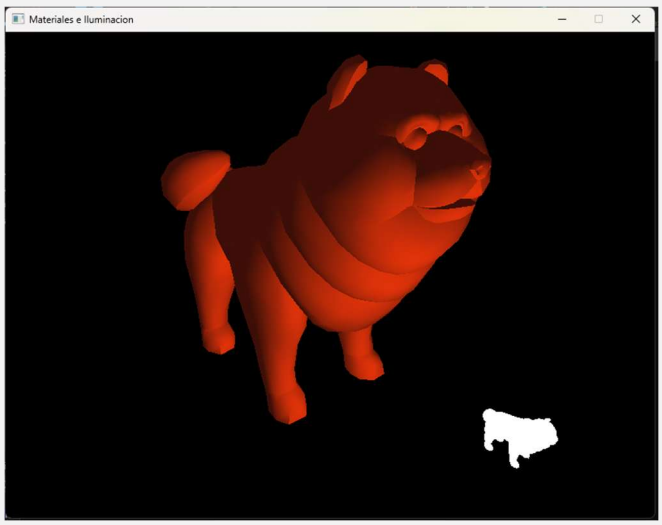


Imagen 2.2 – efectos de la iluminación sobre el modelo cargado.

Después cambiamos la forma de la lampara, en este caso sustituimos la forma cubica, por el modelo del perro. Para ello modificamos las líneas de código donde dibujamos a la fuente de luz cubica.

```
lampshader.Use();
glUniformMatrix4fv(glGetUniformLocation(lampshader.Program, "projection"), 1, GL_FALSE,
glm::value_ptr(projection));
glUniformMatrix4fv(glGetUniformLocation(lampshader.Program, "view"), 1, GL_FALSE,
glm::value_ptr(view));
model = glm::mat4(1.0f);
model = glm::translate(model, lightPos + movelightPos);
model = glm::scale(model, glm::vec3(0.3f));
glUniformMatrix4fv(glGetUniformLocation(lampshader.Program, "model"), 1, GL_FALSE,
glm::value_ptr(model));
red_dog.Draw(lampshader);
```





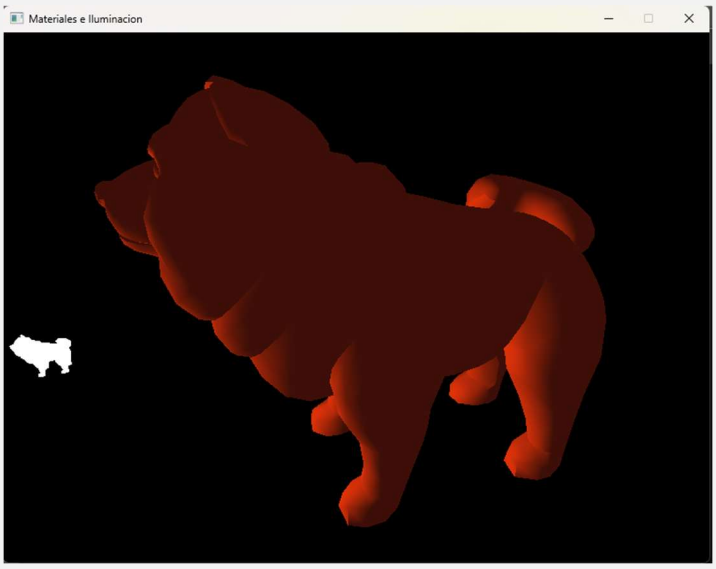


Imagen 2.3 – cambio del modelo de la fuente de luz.

En esta práctica no solo queríamos ver como se le afectaba la luz a un objeto solido de un solo color, también se modificó el código para poder observar como se visualizaba el modelo con texturas aplicadas al exponerlo a una fuente de luz. Para ello se utilizaron las siguientes líneas de código.

```
image = stbi_load("Models/Texture_albedo.jpg", &textureWidth, &textureHeight, &nrChannels, 0);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, textureWidth, textureHeight, 0, GL_RGB, GL_UNSIGNED_BYTE,
image);
glGenerateMipmap(GL_TEXTURE_2D);
```

También se modificó el código del archivo “lighting.frag”, agregando las líneas de Código:

```
uniform sampler2D texture_diffuse;
color = vec4(result, 1.0f)*texture(texture_diffuse,TexCoords);
```

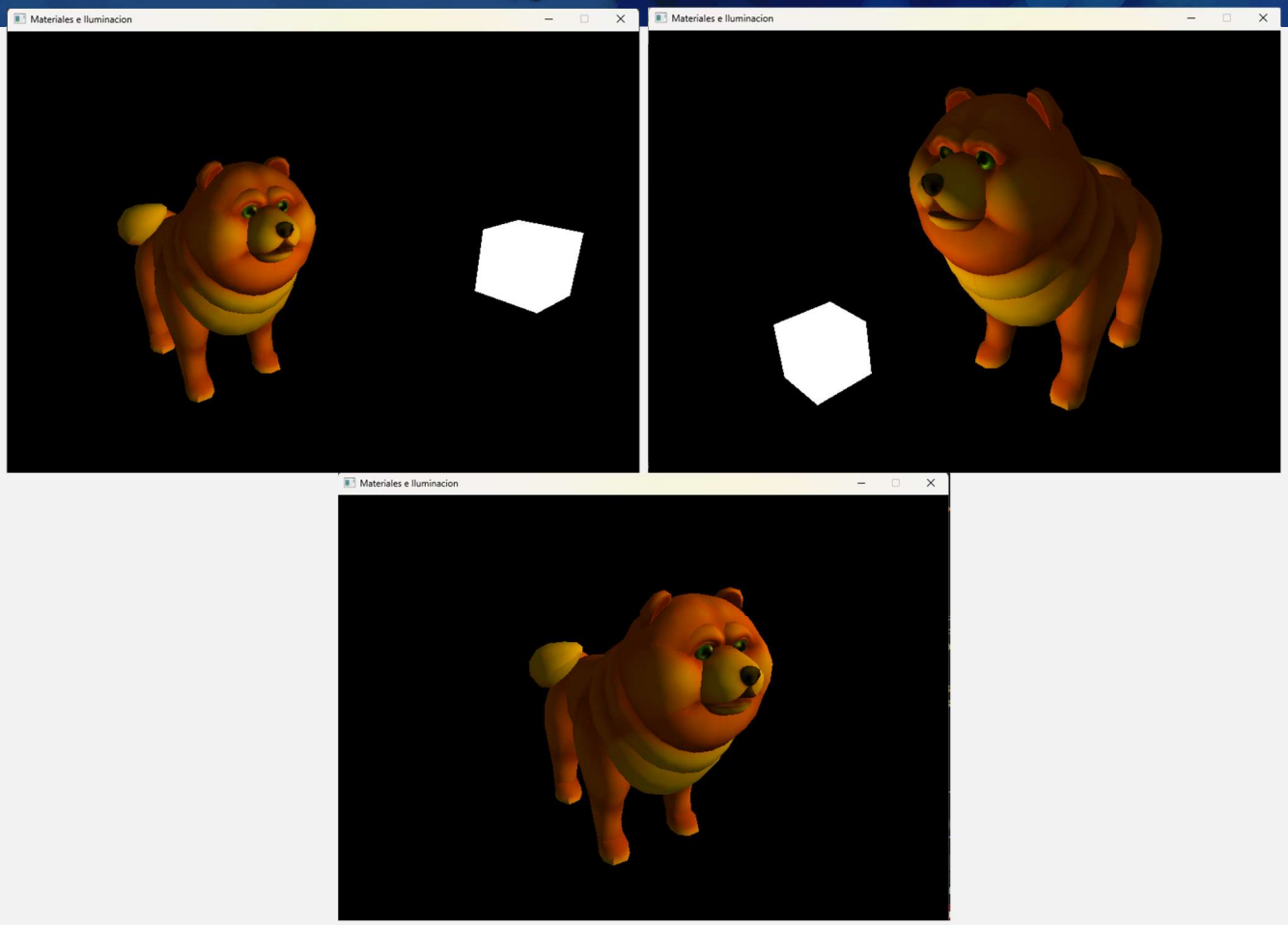


Imagen 2.4 – agregando texturas al modelo y visualizando su efecto ante la luz.

Ahora que ya podemos observar como son afectados los modelos contexturas al exponerse a una fuente de luz, procedemos a cambiar el modelo cargado, en este caso cargamos el modelo de una casa de perro hecha con madera, para ver como se cambia su visualización ante una fuente de luz, para ello se modificaron las siguientes lienas de código.

```
float  rotacion = 100.0f;

Model house((char*)"Models/house.fbx");

image = stbi_load("Models/doghouse0908_PBR_BaseColor.png", &textureWidth, &textureHeight,
&nrChannels, 0);

glm::mat4 model(1);
model = glm::rotate(model, glm::radians(rotacion), glm::vec3(-100.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(3.0f, 3.0f, 3.0f));
glUniformMatrix4fv(glGetUniformLocation(lightingShader.Program, "model"), 1, GL_FALSE,
glm::value_ptr(model));
glBindVertexArray(VAO);
house.Draw(lightingShader);
```

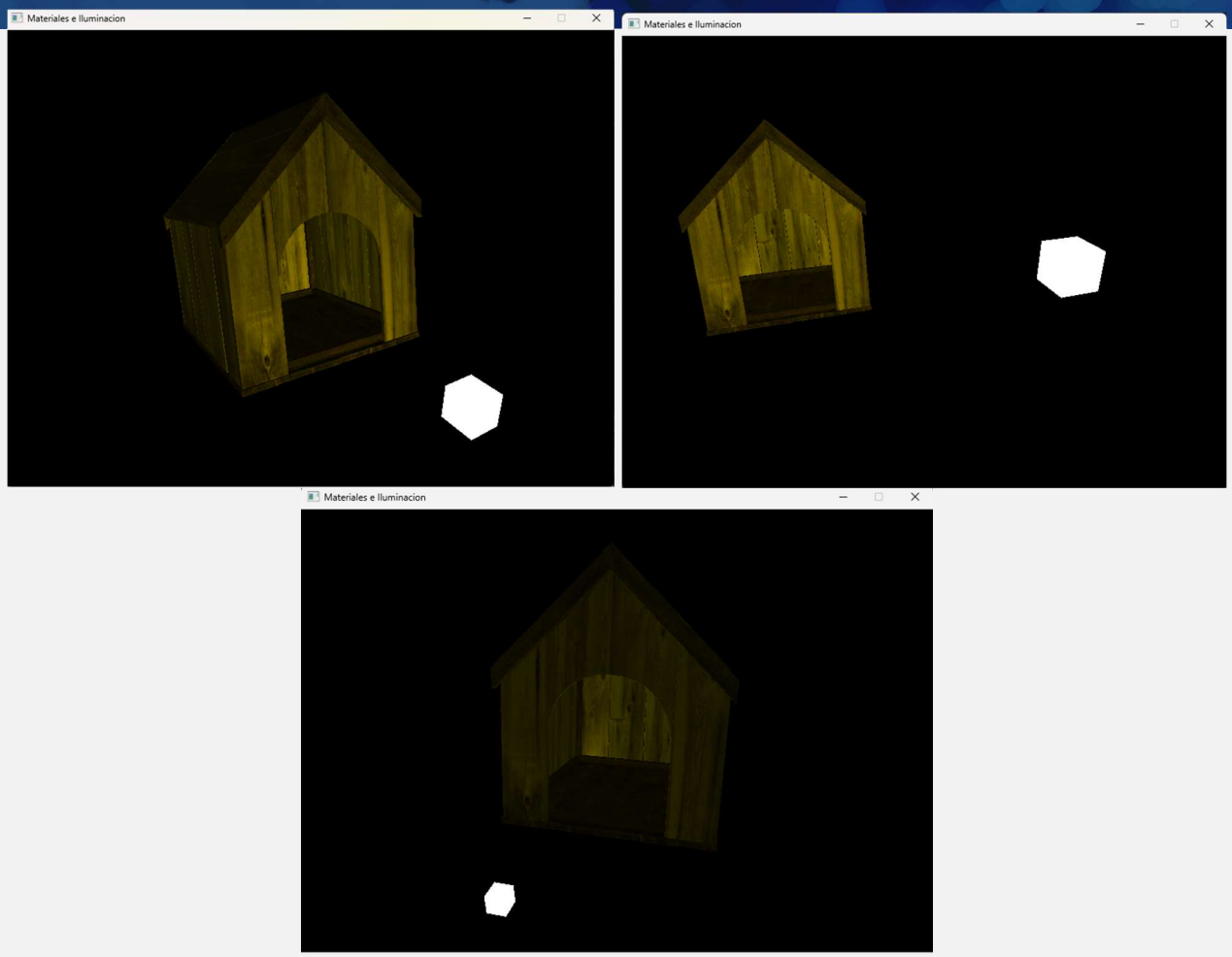


Imagen 2.5 – cambio del modelo, para observar otro efecto de la luz.

Ya por último se volvió a cambiar la forma de la fuente de luz, por otro modelo cargado, en este caso se escogió un trofeo, el cual se descargo de internet y se ocupo como lampara. Se agregaron las siguientes líneas de código para poder utilizarlo.

```
Model cup((char*)"Models/wctrophy_lowpoly.fbx");

lampshader.Use();
glUniformMatrix4fv(glGetUniformLocation(lampshader.Program, "projection"), 1, GL_FALSE,
glm::value_ptr(projection));
glUniformMatrix4fv(glGetUniformLocation(lampshader.Program, "view"), 1, GL_FALSE,
glm::value_ptr(view));
model = glm::mat4(1.0f);
model = glm::translate(model, lightPos + movelightPos);
model = glm::scale(model, glm::vec3(0.3f));
glUniformMatrix4fv(glGetUniformLocation(lampshader.Program, "model"), 1, GL_FALSE,
glm::value_ptr(model));
cup.Draw(lampshader);
```

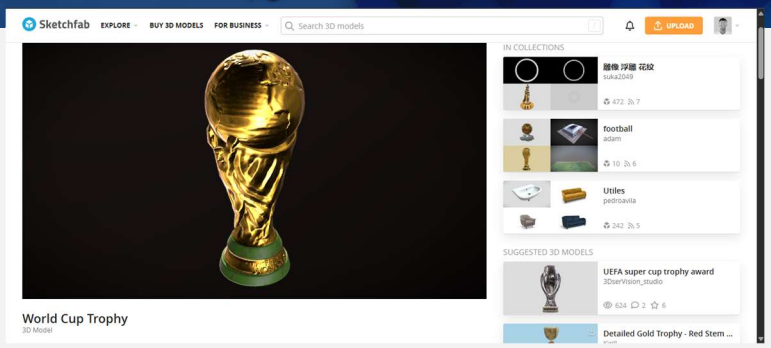


Imagen 2.6 – modelo utilizado como fuente de iluminación.

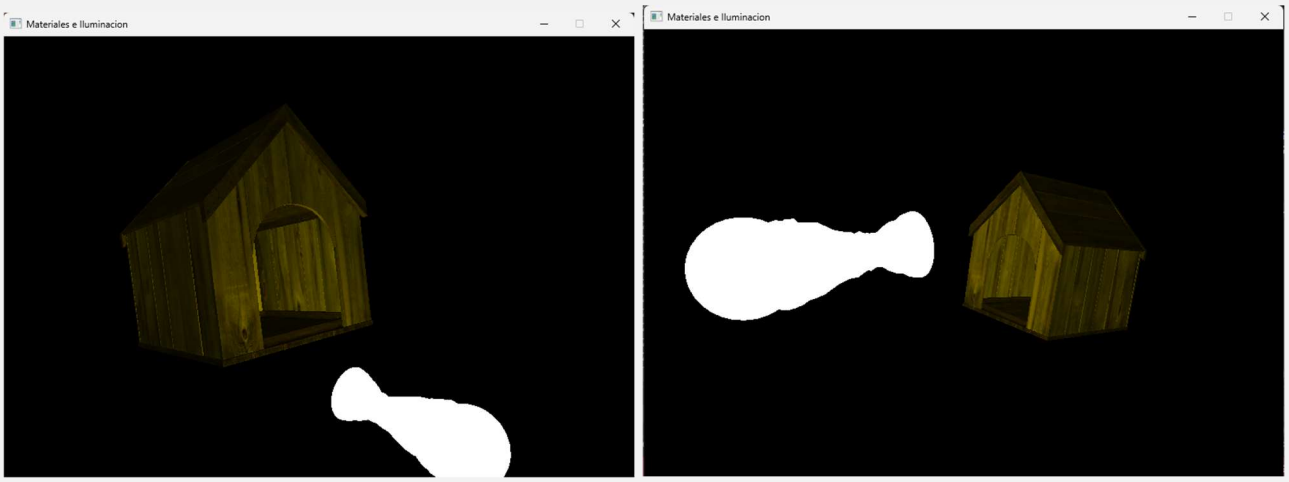


Imagen 2.7 – prueba de carga de modelo como fuente de iluminación.

Resultados: (explicar lo que se logró o aquello que no lograron realizar o comprender)

- Entendí el modelo de iluminación Phong y las capas que lo componen.
- Logre aplicar el concepto de fuente de iluminación.
- Visualice como afecta tener una fuente de iluminación en modelos 3d que tienen texturas incluidas, además de que movíamos la posición de la fuente de luz, para observar el efecto en los modelos utilizados.
- Logramos cambiar la forma de la fuente de iluminación, sustituyendo el cubo de fuente de iluminación por modelos descargados de internet.

Conclusiones:

En esta practica logre entender el modelo de iluminación Phong, cada una de las capas logra sumar un efecto visual de realismo de iluminación, dentro de los entornos físicos que creamos, en este caso aplicamos el modelo de iluminación a una fuente de luz, la cual para probar sus efectos se agregaron 2 modelos, en ellos se vieron los cambias que sufrían en su superficie al acercar y alejar la fuente que producía la luz. También se logro ver como se ve un modelo con texturas al exponerlo a una fuente de iluminación, a lo que mi parecer seria como modelar un cuarto oscuro, donde solo existe un foco, y dependiendo la posición de la fuente de luz, es como podremos visualizar a los modelos que tengamos en la escena.

Bibliografía consultada:

Interactivas y Computación Gráfica, T. [@ArturoVMS]. (s/f). *Materiales e Iluminación en OpenGL Conceptos Básicos y Ejemplo Práctico [Tutorial]* [[Object Object]]. Youtube. Recuperado el 19 de octubre de 2025, de https://www.youtube.com/watch?v=iI8muNf7-8g&

World Cup Trophy - Download Free 3D model by waimus. (2018, julio 5). https://sketchfab.com/3d-models/world-cup-trophy-c4ae2dd470194a81b856ad84620d8beb