UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
FACULTAD DE INGENIERÍA
DIVISIÓN DE INGENIERÍA ELÉCTRICA
INGENIERÍA EN COMPUTACIÓN
COMPUTACIÓN GRÁFICA E INTERACCIÓN HUMANO COMPUTADORA

Práctica 8:Iluminación II

Nombre del alumno: Mendoza Flores Martin Eduardo
N° de cuenta: 318285687
Grupo de Laboratorio:7
Grupo de Teoría: 6
Semestre 2025-2

Fecha de entrega límite: 07/04/2025

Calificación:

Introducción:

En esta práctica se abordó el manejo avanzado de fuentes de luz en entornos 3D utilizando OpenGL. El enfoque principal fue implementar diferentes tipos de iluminación en una escena renderizada, lo cual es fundamental para lograr una visualización más realista y detallada en gráficos por computadora. La iluminación juega un papel crucial en la forma en que los objetos son percibidos, ya que define sombras, brillos y la profundidad de la escena. Por ello, comprender cómo se comportan y configuran distintos tipos de luz en OpenGL permite mejorar significativamente la calidad visual de los proyectos desarrollados.

Esta práctica incluye el uso de luces direccionales, luces puntuales y luces tipo spotlight, además de la implementación de transparencia en modelos 3D. Cada tipo de luz fue configurado cuidadosamente con sus respectivos parámetros como la dirección, color, atenuación y ángulos de corte, lo cual implicó modificar shaders y estructuras dentro del código fuente del proyecto. Adicionalmente, se integraron nuevos modelos 3D y texturas que permitieron poner en práctica los conceptos aprendidos de iluminación y blending.

Esta práctica permitió reforzar el conocimiento teórico sobre los tipos de fuentes de luz y su comportamiento en entornos tridimensionales, así como su implementación práctica utilizando funciones de la API de OpenGL.

Objetivo:

El objetivo principal de esta práctica fue comprender e implementar distintos tipos de fuentes de luz en una escena 3D utilizando OpenGL y shaders, así como aprender a controlar sus parámetros para lograr efectos visuales específicos.

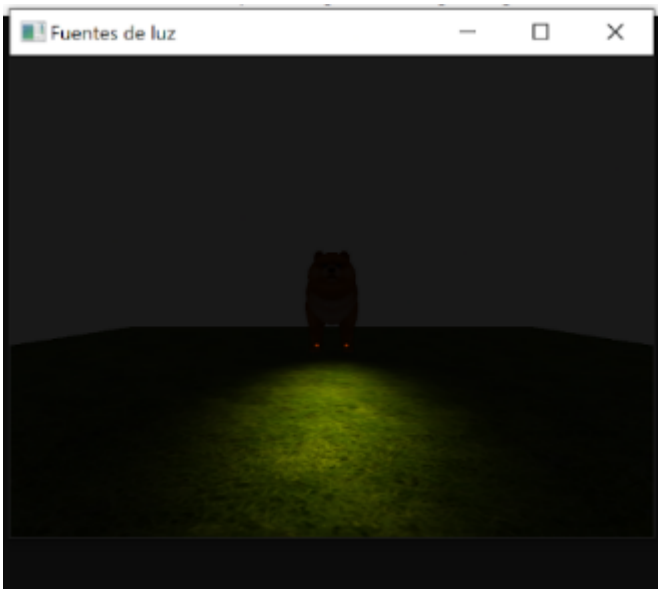
Se buscó responder qué efectos de iluminación pueden implementarse, por qué es importante entender su configuración y comportamiento, y para qué sirven en el diseño visual de entornos tridimensionales. Además, se trabajó en cómo aplicarlos adecuadamente dentro de una escena con modelos, texturas y movimiento de cámara.

También se exploró la aplicación del canal alfa para integrar a un objeto de transparencia, mostrando cómo integrar estas funcionalidades de manera correcta y eficiente dentro de un motor gráfico básico.

Desarrollo:

Primera Parte:

Primero es necesario agregar todos los archivos requeridos para que el programa `fuentesDeLuz.cpp`, el cual será el nuevo archivo *main* de la práctica para que funcione correctamente. También se deben incluir los *shaders* y demás recursos necesarios como los nuevos modelos que se usarán, asegurándose de colocarlos en las carpetas correspondientes dentro del proyecto. Si la configuración es correcta, al ejecutar el programa se visualizará una escena con el modelo redDog, un piso con textura de pasto y una luz puntual iluminando el entorno.



Segunda Parte:
Posteriormente se nos pidió modificar los parámetros de iluminación para que toda la escena pudiera visualizarse correctamente. Para lograrlo se ajustó la luz direccional (*Directional light*) aumentando los valores de la componente ambiental. Específicamente, se estableció la luz ambiental en valores máximos (`1.0f, 1.0f, 1.0f`), mientras que las componentes difusa y especular se desactivaron (`0.0f`). El código utilizado fue el siguiente:

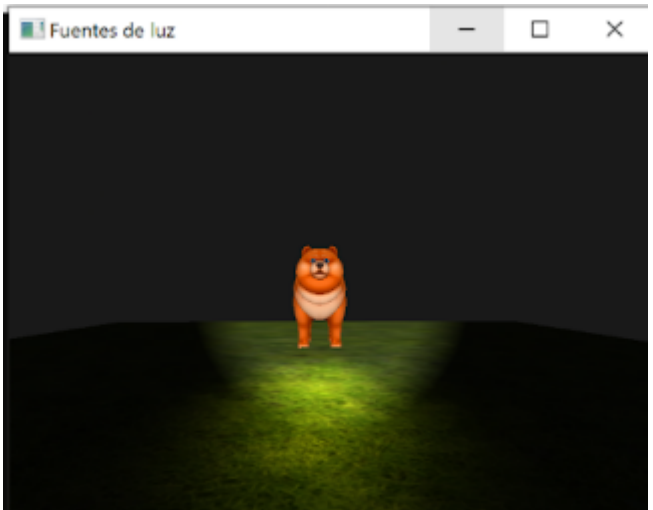
```
// Directional light
glUniform3f(glGetUniformLocation(lightingShader.Program, "dirLight.direction"), -0.2f, -1.0f, -0.3f);
glUniform3f(glGetUniformLocation(lightingShader.Program,"dirLight.ambient"),1.0f,1.0f,1.0f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "dirLight.diffuse"), 0.0f, 0.0f, 0.0f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "dirLight.specular"),0.0f, 0.0f, 0.0f);
```

Tercera Parte:

En esta parte se nos pidió implementar una luz tipo *spotlight* (luz focal) que siguiera la posición y dirección de la cámara, simulando el efecto de una linterna que ilumina hacia donde el usuario observa. Para lograr esto se configuraron los parámetros de la luz puntual con base en la posición (`camera.GetPosition()`) y dirección frontal (`camera.GetFront()`) de la cámara. Esto permite que la fuente de luz se mueve dinámicamente con la vista del usuario.

Además se ajustaron las propiedades de iluminación ambiental y difusa con valores moderados (`0.5f`), mientras que la componente especular fue desactivada (`0.0f`). También se definieron los coeficientes de atenuación (constant, linear y quadratic), estableciendo sólo el valor constante en `1.0f`, lo que significa que la luz no disminuye con la distancia. Finalmente se definieron los ángulos de corte del cono de luz (`cutOff` y `outerCutOff`) controlando qué tan enfocado está el haz.

```
glUniform3f(glGetUniformLocation(lightingShader.Program, "spotLight.position"), camera.GetPosition().x, camera.GetPosition().y, camera.GetPosition().z);

glUniform3f(glGetUniformLocation(lightingShader.Program, "spotLight.direction"), camera.GetFront().x, camera.GetFront().y, camera.GetFront().z);

glUniform3f(glGetUniformLocation(lightingShader.Program, "spotLight.ambient"), 0.5f, 0.5f, 0.5f);

glUniform3f(glGetUniformLocation(lightingShader.Program, "spotLight.diffuse"), 0.5f, 0.5f, 0.5f);

glUniform3f(glGetUniformLocation(lightingShader.Program, "spotLight.specular"), 0.0f, 0.0f, 0.0f);

glUniform1f(glGetUniformLocation(lightingShader.Program, "spotLight.constant"), 1.0f);

glUniform1f(glGetUniformLocation(lightingShader.Program, "spotLight.linear"), 0.0f);

glUniform1f(glGetUniformLocation(lightingShader.Program, "spotLight.quadratic"), 0.0f);

glUniform1f(glGetUniformLocation(lightingShader.Program, "spotLight.cutOff"), glm::cos(glm::radians(12.0f)));

glUniform1f(glGetUniformLocation(lightingShader.Program, "spotLight.outerCutOff"), glm::cos(glm::radians(18.0f)));
```



Cuarta Parte:
En esta parte se nos solicitó mover las luces puntuales (*point lights*) a la escena para mejorar la iluminación local y dar mayor realismo al entorno. En este caso, se movieron varias posiciones para estas luces utilizando un arreglo de vectores `glm::vec3`, con coordenadas distribuidas en las esquinas del escenario:

```
glm::vec3 pointLightPositions[] = {
        glm::vec3(2.0f, 0.2f, 2.0f),
        glm::vec3(-2.0f, 0.2f, 2.0f),
        glm::vec3(-2.0f, 0.2f, -2.0f),
        glm::vec3(2.0f, 0.2f, -2.0f)
};
```

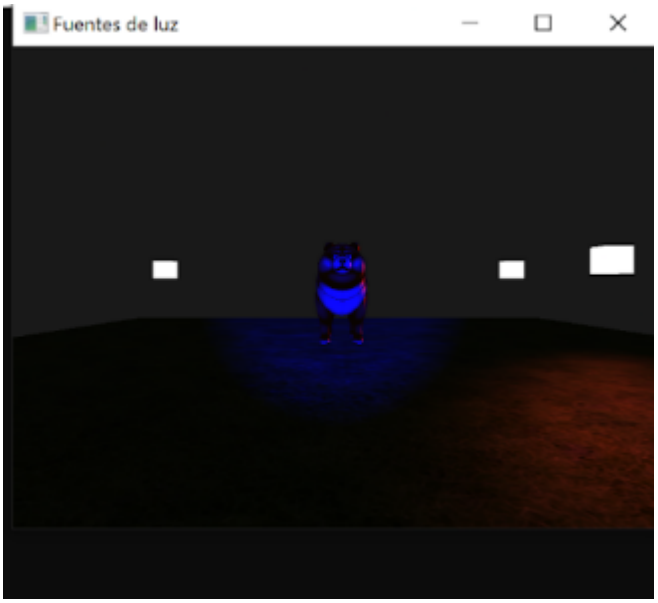
Para cada luz puntual, se establecieron sus propiedades utilizando las funciones `glUniform3f` y `glUniform1f`, enviando al *shader* los valores correspondientes. En el siguiente fragmento se muestra la configuración para la primera luz:

```
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[0].position"), pointLightPositions[0].x, pointLightPositions[0].y, pointLightPositions[0].z);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[0].ambient"), lightColor.x, lightColor.y, lightColor.z);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[0].diffuse"), lightColor.x, lightColor.y, lightColor.z);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[0].specular"), 1.0f, 0.2f, 0.2f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[0].constant"), 1.0f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[0].linear"), 0.045f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[0].quadratic"), 0.075f);
```

La posición define el punto exacto donde se encuentra la fuente de luz.

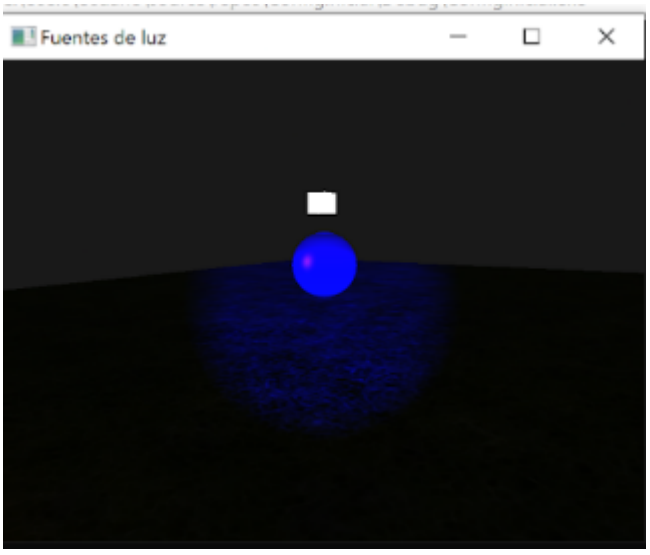
La luz ambiental y difusa se definen con el color base `lightColor`, mientras que la especular se configuró con un tono rojizo para destacar los reflejos.

Los factores constant, linear y quadratic determinan la atenuación de la luz con la distancia, logrando un efecto más natural en cómo se propaga la luz.



Quinta Parte:

En esta parte de la práctica se nos pidió sustituir el modelo original del perro por el modelo llamado ball. Inicialmente el modelo se renderizaba de forma normal, sin ningún tipo de efecto especial. Posteriormente se implementó el uso de transparencia para modificar la forma en que se visualiza el modelo en la escena.



Para lograr este efecto fue necesario activar el canal alfa utilizando las funciones de mezcla de OpenGL. Primero se inicializa la matriz del modelo con `glm::mat4(1)` para evitar transformaciones previas. Luego, se activa el *blending* con:

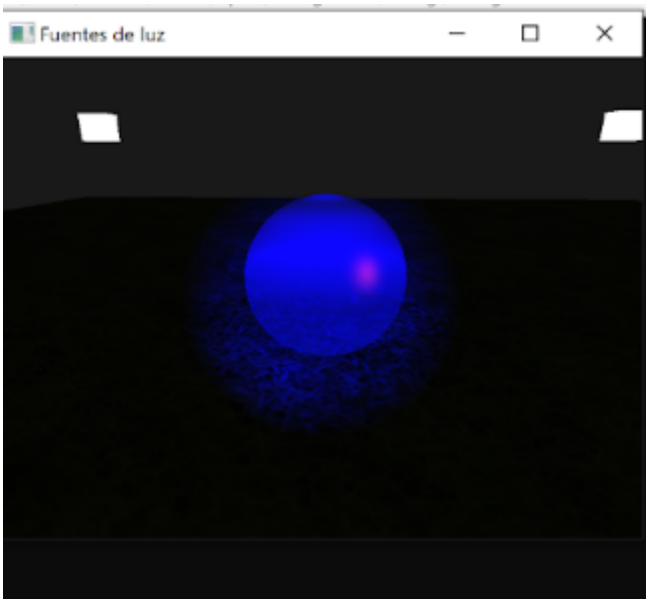
```
glEnable(GL_BLEND);
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
```

Esto permite combinar los valores de color del modelo con el fondo según el valor del canal alfa (transparencia). Posteriormente, se pasa la matriz de transformación del modelo y se activa una variable booleana en el shader para indicar que se debe aplicar transparencia:
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
glUniform1i(glGetUniformLocation(lightingShader.Program, "transparency"), 1);

Finalmente se dibuja el modelo con `Dog.Draw(lightingShader);` (en este caso, haciendo referencia al modelo ball) y se desactiva la opción de mezcla para evitar que afecte otros objetos:

glDisable(GL_BLEND);
glBindVertexArray(0);

Con esta implementación el modelo ball adquiere un efecto translúcido lo que mejora la apariencia visual de la escena y demuestra el uso correcto de la transparencia en objetos 3D.



Conclusiones:

Durante esta práctica aprendí a implementar y controlar distintas fuentes de luz dentro de una escena en OpenGL, lo cual me permitió entender mejor el impacto que cada tipo de iluminación tiene sobre el entorno y los modelos 3D. También aprendí cómo ajustar propiedades como la dirección, intensidad y color de la luz, así como a simular efectos como el enfoque de una linterna o la propagación natural de la luz con distancia.

Además entendí cómo se puede aplicar transparencia a un modelo utilizando blending, una técnica muy útil en el desarrollo de videojuegos o visualizaciones avanzadas. Me di cuenta de que muchos de los efectos visuales que vemos en aplicaciones gráficas se logran con configuraciones bastante precisas de shaders y parámetros de iluminación.

En general la práctica logró unir teoría y práctica, permitiendo aplicar conocimientos previos sobre vectores, materiales y shaders, y ver resultados visibles inmediatos en una escena tridimensional. La experiencia fue muy útil para consolidar mis habilidades en programación gráfica y entender mejor cómo se construyen entornos visuales más inmersivos.

Referencias:

Kubisch, C. (n.d.). *Transparency and Blending*. NVIDIA Developer. Retrieved from https://developer.nvidia.com/content/transparency-and-blending

Shreiner, D., Sellers, G., Kessenich, J., & Licea-Kane, B. (2013). *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 4.3* (8.ª ed.). Addison-Wesley.

Rost, R. J., Licea-Kane, B., Ginsburg, D., Kessenich, J., Lichtenbelt, B., Malan, R., & Weiblen, M. (2009). *OpenGL Shading Language* (3.ª ed.). Addison-Wesley.

Angel, E., & Shreiner, D. (2012). *Interactive Computer Graphics: A Top-Down Approach with Shader-Based OpenGL* (6.ª ed.). Pearson.

de Vries, J. (n.d.). *Blending*. LearnOpenGL. Recuperado de https://learnopengl.com/Advanced-OpenGL/Blending

Stanis, T. (n.d.). *OpenGL Blending Tutorial*. Recuperado de https://users.polytech.unice.fr/~buffa/cours/synthese_image/DOCS/Tutoriaux/Nehe/lesson8.htm

Lighthouse3D. (n.d.). *GLSL Tutorial – Point Lights*. Recuperado de https://www.lighthouse3d.com/tutorials/glsl-tutorial/point-lights/

Lighthouse3D. (n.d.). *GLSL Tutorial – Spotlights*. Recuperado de https://www.lighthouse3d.com/tutorials/glsl-tutorial/spotlights/

OGLdev. (n.d.). *Tutorial 20 - Point Light*. Recuperado de https://www.ogldev.org/www/tutorial20/tutorial20.html

OGLdev. (n.d.). *Tutorial 21 - Spot Light*. Recuperado de https://www.ogldev.org/www/tutorial21/tutorial21.html