UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE INGENIERÍA ELÉCTRICA

INGENIERÍA EN COMPUTACIÓN

LABORATORIO DE COMPUTACIÓN GRÁFICA e INTERACCIÓN HUMANO COMPUTADORA

Reporte de práctica Nº 07

NOMBRE COMPLETO: Tapia Ledesma Angel Hazel

Nº de Cuenta: 320070358

GRUPO DE LABORATORIO: 02

GRUPO DE TEORÍA: 02

SEMESTRE 2026-1

FECHA DE ENTREGA LÍMITE: 19-10-25

CALIFICACIÓN: _____

Actividades realizadas

1.-Agregar movimiento con teclado al helicóptero hacia adelante y atrás.
2.-crear luz spotlight de helicóptero de color amarilla que apunte hacia el piso y se mueva con el helicóptero
3.- Añadir en el escenario 1 modelo de lámpara texturizada (que usarán en su proyecto final) y crearle luz puntual blanca

Descripción de las actividades

Para la primer y segunda actividad, las cuales son enfocadas al helicóptero, lo que se hizo fue primero reutilizar la lógica del ejercicio de la práctica 7, ya que en el Window.cpp estaba el código para mover hacia adelante y atrás el carro, por lo que lo que se hizo fue reutilizar este código, solo cambiando la variable en la que se asignan los cambios de coordenada x, también las teclas con las que se avanza y retrocede, de igual forma, se reutilizó la línea del main que hacer que se traslade el objeto con base en el movimiento del teclado, el cual teníamos en el auto, solo cambiamos las variables correspondientes al helicóptero. En cuanto a la luz spotlight del helicóptero, lo que se hizo fue generar otra luz, lo importante aquí era poner la dirección para que la luz apuntara había el piso, la cual era –1 en y, una vez hecho esto, se reutilizó también el código correspondiente al faro del auto del ejercicio de la práctica 7, el cual es el que sirve para que la luz acompañe al helicóptero, cambiando lo necesario para referirnos ahora a la nueva luz creada y no a la del auto. Una vez referenciada esta luz, lo que se hizo fue cambiar los parámetros para el incremento con las teclas para el movimiento del helicóptero, también las coordenadas iniciales de la luz y finalmente el setFlash con dirección al piso, una vez hecho esto, el helicóptero ya se movía junto con la luz que ilumina el piso. Para el caso del avión, lo que hicimos fue usar el tipo de luz lineal, sin embargo, no se nota mucho, ya que helicóptero esta siempre a la misma altura, por lo que no cambia la distancia en la que se proyecta la luz.

Para el 3er punto de la práctica, primero se realizó el modelo en blender, posteriormente se le añadio el texturizado, usando el UV Map del modelo y después trabajando sobre el en blender. Una vez terminado el proceso, se exportó el modelo a en .obj a la carpeta de Models. Posteriormente se abrió el programa de Open GL para cargar el modelo dentro de nuestro programa, una vez cargado, en el main aplicamos las transformaciones para acomodarlo en el escenario, después de esto, se procedió a crear una nueva luz, de tipo Point Light, cómo esta iba a ser estática, solamente se trabajó en el momento en donde se crea, es decir fuera del while, para esta luz, solo se posiciono en el lugar correspondiente, que en este caso es el faro del poste, y se movieron los parámetros para las intensidades y el tipo de luz, en este

caso usamos la exponencial, para dar el efecto atenuación abrupta o más parecida a la realidad.
NOTA: Para que las luces se notaran más y se viera mejor el efecto de atenuación, se modificó la main light, en este caso se cambio a color negro y se puso en poca intensidad, para dar el efecto de noche o oscuridad.

Código generado:

Actividad 1 y 2

Window.cpp

```cpp
Window::Window(GLint windowWidth, GLint windowHeight)
{
    width = windowWidth;
    height = windowHeight;
    muevex = 2.0f;
    muevexhelicoptero = 2.0f;
    for (size_t i = 0; i < 1024; i++)
    {
        keys[i] = 0;
    }
}
```

```cpp
if (key == GLFW_KEY_I)
{
    theWindow->muevexhelicoptero += 1.0;
}
if (key == GLFW_KEY_O)
{
    theWindow->muevexhelicoptero -= 1.0;
}
```

Window.H

```cpp
class Window
{
public:
    Window();
    Window(GLint windowWidth, GLint windowHeight);
    int Initialise();
    GLfloat getBufferWidth() { return bufferWidth; }
    GLfloat getBufferHeight() { return bufferHeight; }
    GLfloat getXChange();
    GLfloat getYChange();
    GLfloat getmuevex() { return muevex; }
    GLfloat getmuevexhelicoptero() { return muevexhelicoptero; }
    bool getShouldClose() {
        return  glfwWindowShouldClose(mainWindow);}
    bool* getsKeys() { return keys; }
    void swapBuffers() { return glfwSwapBuffers(mainWindow); }

    ~Window();
private:
    GLFWwindow *mainWindow;
    GLint width, height;
    bool keys[1024];
    GLint bufferWidth, bufferHeight;
    void createCallbacks();
    GLfloat lastX;
    GLfloat lastY;
    GLfloat xChange;
    GLfloat yChange;
    GLfloat muevex;
    GLfloat muevexhelicoptero;
    bool mouseFirstMoved;
    static void ManejaTeclado(GLFWwindow* window, int key, int code, int action, int mode);
    static void ManejaMouse(GLFWwindow* window, double xPos, double yPos);

};
```

P07 – 320070358.cpp

```
// Luz para el helicoptero
spotLights[3] = SpotLight(1.0f, 1.0f, 0.0f,
    0.7f, 0.5f,
    3.0f, 2.0f, 5.0f,
    -1.0f, 0.0f, 0.0f,
    0.0f, 0.3f, 0.0f,
    15.0f);
spotLightCount++;
```

```
float posxhelicoptero = 0;
```

```
// Helicoptero
posxhelicoptero = mainWindow.getmuevexhelicoptero();
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f + posxhelicoptero, 5.0f, 12.0));
model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.3f));
model = glm::rotate(model, -90 * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::rotate(model, 90 * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Blackhawk_M.RenderModel();

// Luz en el helicoptero
glm::vec3 helicopteroLight = glm::vec3(0.0f, 5.0f, 12.0);
helicopteroLight.x += posxhelicoptero;
spotLights[3].SetFlash(helicopteroLight, glm::vec3(0.0f, -1.0f, 0.0f));
```

Actividad 3

P07-320070358.cpp

```
Model Poste_M;
```

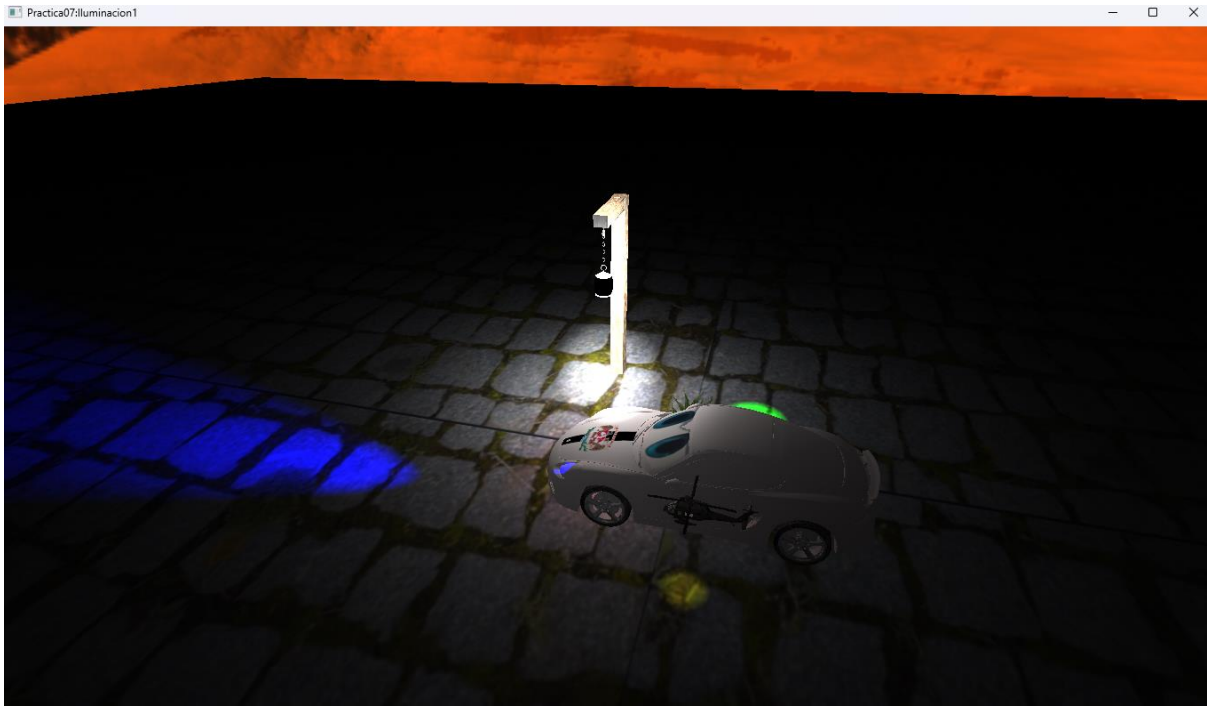
```
Poste_M = Model();
Poste_M.LoadModel("Models/poste.obj");
```

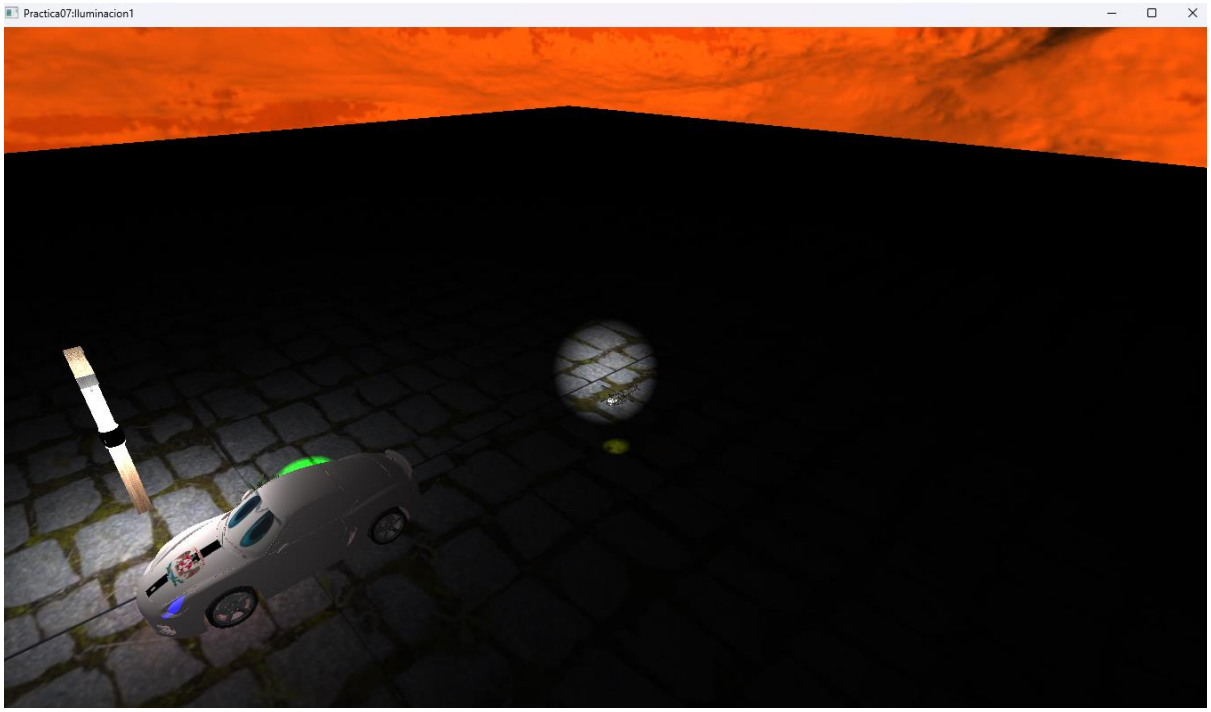
```
// Luz para el poste
pointLights[1] = PointLight(1.0f, 1.0f, 1.0f,
    0.6f, 0.8f,
    -8.0f, 10.0f, -7.0f,
    0.0f, 0.0f, 0.01f);
pointLightCount++;
```

```
//Poste
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-8.0f, 7.0f, -12.0f));
model = glm::scale(model, glm::vec3(3.0f, 3.0f, 3.0f));
modelaux = model;
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Poste_M.RenderModel();
```

Ejecución

Actividad 1 y 2 (Movimiento del helicóptero con “I”, “O”)

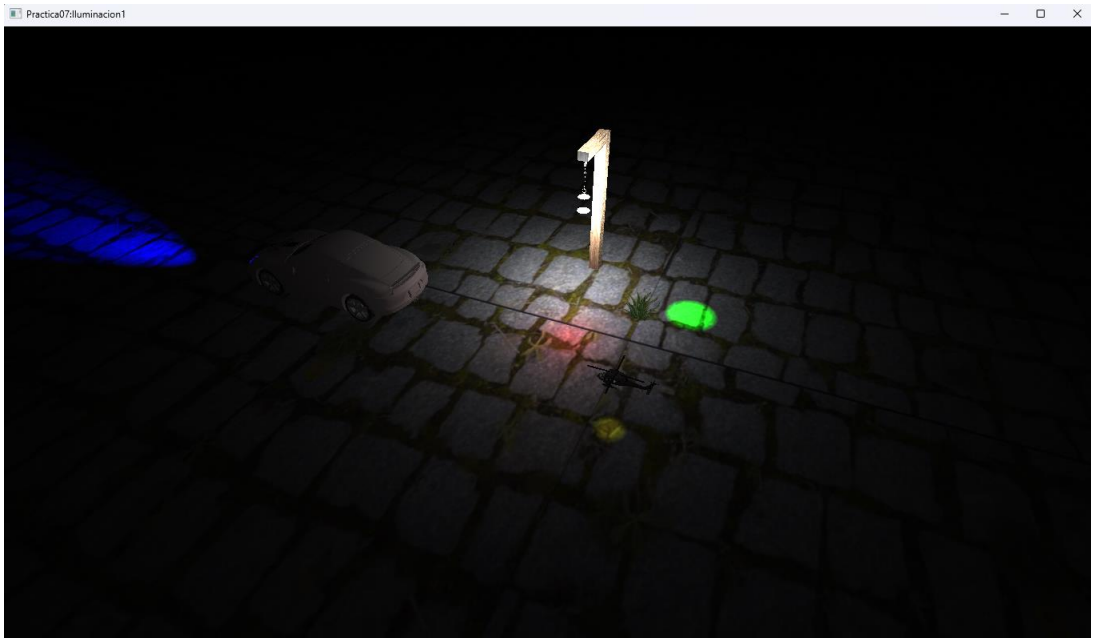




Video:

https://youtu.be/kdSFqjFzBxY

Actividad 2 y 3



Problemas presentados

Para esta práctica no se encontró ningún inconveniente, ya que realmente con haber hecho el ejercicio en clase, esta práctica era similar, ya que se usaba la spotlight, solo había que cambiar dirección, coordenadas y adaptar la lógica para que siguiera al helicóptero, en el caso de la Point light, tampoco hubo complicaciones, ya que solo había que posicionarla y adaptar el tipo de iluminación.

Conclusión:

Esta práctica fue muy buena e interesante, ya que nos permitió conocer y aplicar el nuevo tema, el cual es sobre la iluminación, fue bastante completo, ya que pudimos ver los tipos de luces, como la directional, point light y spotlight, así como observar sus diferencias y cómo actúan cuando sus parámetros son modificados, para una mejor compresión al poder visualizarlas.

La complejidad de esta práctica fue sencilla, considero que realmente no era muy tardado ni difícil adaptar el código para el movimiento del helicóptero junto con su luz, ni tampoco la carga del modelo texturizado a open GL, y en el caso de poner la point light fue similar, algo bastante sencillo, lo más complejo quizá es la lógica para mover la luz junto con el helicóptero, pero si teníamos el ejercicio de clase hecho, pues esa parte ya estaba casi que hecha.

No cuento con ningún comentario para esta práctica, todo lo aprendido en laboratorio fue suficiente para entender cómo funcionan las luces y sus parámetros, a excepción de cons, lin y exp, pero estos quedaron claros con el previo, el resto era aplicar cosas que ya sabíamos o eran más lógica de programación.