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 08

NOMBRE COMPLETO: Roberto Aburto López

Nº de Cuenta: 319131996

GRUPO DE LABORATORIO: 03

GRUPO DE TEORÍA: 06

SEMESTRE 2026-1

FECHA DE ENTREGA LÍMITE: 25 de octubre de 2025

CALIFICACIÓN: _____

REPORTE DE PRÁCTICA:

1. Agrega un **spotlight** (que no sea blanco ni azul) que salga del cofre de tu coche y que, al abrir o cerrar el cofre, ilumine en esa dirección.

Para este ejercicio me basé en lo realizado en prácticas anteriores, donde separé el cofre del coche y le di un movimiento limitado de 40°. Después, añadí la luz en la posición del cofre para que se moviera junto con él.

Window.ccp

```
// Cofre
    if (key == GLFW_KEY_K)
    {
        if (theWindow->articulacion1 > 40)
        {
        }
        else
        {
            theWindow->articulacion1 += 10.0;
        }
    }
    if (key == GLFW_KEY_J)
    {
        if (theWindow->articulacion1 < 10)
        {
        }
        else
        {
            theWindow->articulacion1 -= 10.0;
        }
    }
```

En esta parte del código se configura la luz del cofre (verde). Primero se define un desplazamiento (cofreOffset) para ajustar la posición exacta donde estará la luz respecto al coche. Luego, se calcula la posición del cofre sumando ese desplazamiento a la posición general del vehículo.

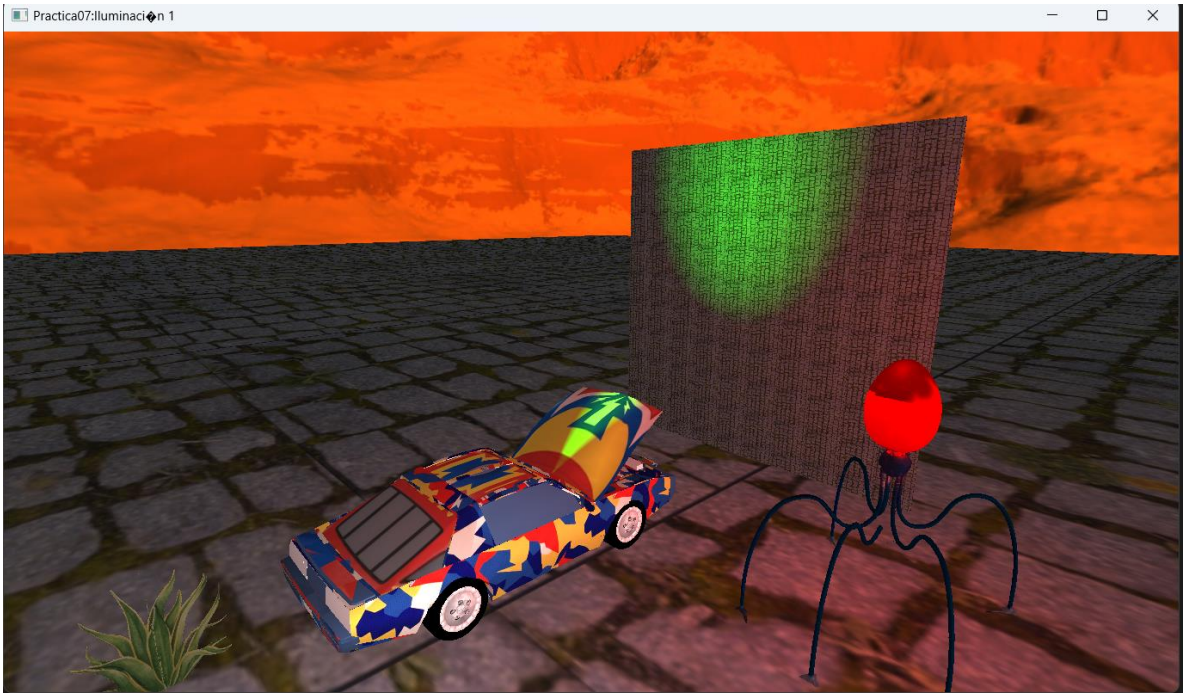
Después, se obtiene el ángulo de apertura del cofre (cofreAngle) y se usa para calcular la dirección del haz de luz. Con las funciones cos() y sin() se determina hacia dónde apunta la luz según el movimiento del cofre, y finalmente se normaliza ese vector para mantener la dirección correcta.

Por último, con spotLights[0].SetFlash(cofrePosition, direccionLuz) se actualiza la posición y dirección del spotlight, haciendo que la luz se mueva y apunte de acuerdo al ángulo de apertura del cofre.

```
// --------------------- LUZ DEL COFRE (VERDE) ----------------------

        glm::vec3 cofreOffset = glm::vec3(3.3f, 1.4f, 0.0f);
        glm::vec3 cofrePosition = glm::vec3(mainWindow.getmuevex(), 0.6f, -3.0f) + cofreOffset;
        float cofreAngle = glm::radians(mainWindow.getarticulacion1());
        glm::vec3 direccionBase = glm::vec3(1.0f, 0.0f, 0.0f);
        float cosAngle = cos(cofreAngle);
        float sinAngle = sin(cofreAngle);
        glm::vec3 direccionLuz = glm::vec3(cosAngle, sinAngle,0.0f);
        direccionLuz = glm::normalize(direccionLuz);
        spotLights[0].SetFlash(cofrePosition, direccionLuz);
```

```
// --------------------------------  Cofre --------------------------------
        model = modelaux;
        model = glm::translate(model, glm::vec3(3.3f, 1.4f, 0.0f));
        model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
        model = glm::rotate(model, -90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
        model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()), glm::vec3(0.1f,
0.0f, 0.0f));
        glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
        Cofre_M.RenderModel();
```

Resultados


Practica07:Iluminaci&#65533;n 1

2. Añade dos luces tipo spotlight para el coche:
   - Una que se encienda al avanzar (cuando se mueve en dirección X negativa) e ilumine hacia adelante.
   - Otra que se encienda al retroceder (cuando se mueve en dirección X positiva) e ilumine hacia atrás.
     Ambas deben activarse o desactivarse según una bandera que controles en tu código.

En este ejercicio se implementó un sistema de spotlights que se activan y desactivan automáticamente según la dirección de movimiento del coche. En el archivo *Window.h* se añadieron dos banderas booleanas, spotlightAvanzar y spotlightRetroceder, junto con sus métodos getSpotlightAvanzar() y getSpotlightRetroceder(), los cuales indican si las luces deben encenderse. En *Window.cpp*, ambas banderas se inicializan en *false* dentro del constructor para que las luces comiencen apagadas. Luego, en la función ManejaTeclado(), se configuraron las teclas Y y U: al presionar Y, se activa la luz trasera (azul) y se apaga la delantera; al presionar U, se enciende la luz delantera (roja) y se apaga la trasera; y al soltar cualquiera de las dos, ambas se apagan.

Window.h

```
// Spotlights del coche
    bool getSpotlightAvanzar() { return spotlightAvanzar; }
    bool getSpotlightRetroceder() { return spotlightRetroceder; }
```

```
// Banderas para spotlights del coche
    bool spotlightAvanzar;
    bool spotlightRetroceder;
```

Window.cpp

```
if (key == GLFW_KEY_Y && action == GLFW_PRESS)
    {
        theWindow->muevex += 1.0;
        // Activar spotlight de retroceder (AZUL - direcci□n +X)
        theWindow->spotlightRetroceder = true;
        theWindow->spotlightAvanzar = false;
    }
    if (key == GLFW_KEY_U && action == GLFW_PRESS)
    {
        theWindow->muevex -= 1.0;
        // Activar spotlight de avanzar (ROJO - direcci□n -X)
        theWindow->spotlightAvanzar = true;
```

```
        theWindow->spotlightRetroceder = false;
    }

    // Apagar spotlights cuando se sueltan las teclas
    if ((key == GLFW_KEY_Y || key == GLFW_KEY_U) && action == GLFW_RELEASE)
    {
        theWindow->spotlightAvanzar = false;
        theWindow->spotlightRetroceder = false;
    }
```

Finalmente, en *P08-319131996.cpp* se define el comportamiento de los spotlights: el azul se posiciona en el lado positivo del eje X (+8.0f) e ilumina hacia adelante cuando el coche retrocede, mientras que el rojo se coloca en el lado negativo (-6.0f) e ilumina hacia atrás al avanzar. Si están apagados, ambas luces se mueven fuera del rango visible (-1000 en Y). De esta forma, las luces responden dinámicamente al movimiento del coche mediante las teclas asignadas.

```
// ------------------------ LUZ DEL COCHE -------------------------------
        glm::vec3 carPosition = glm::vec3(mainWindow.getmuevex(), 0.6f, -3.0f);

        // Faro Trasero AZUL (retroceder - direcci□n +X) - spotLights[1]
        if (mainWindow.getSpotlightRetroceder()) {
            glm::vec3 carLightOffsetRight = glm::vec3(8.0f, 0.0f, 0.0f);
            spotLights[1].SetFlash(carPosition + carLightOffsetRight, glm::vec3(1.0f,
0.0f, 0.0f));
        }
        else {
            // Apagar el spotlight movi□ndolo muy lejos
            spotLights[1].SetFlash(glm::vec3(0.0f, -1000.0f, 0.0f), glm::vec3(0.0f, -
1.0f, 0.0f));
        }

        // Faro Frontal ROJO (avanzar - direcci□n -X) - spotLights[2]
        if (mainWindow.getSpotlightAvanzar()) {
            glm::vec3 carLightOffsetLeft = glm::vec3(-6.0f, 0.0f, 0.0f);
            spotLights[2].SetFlash(carPosition + carLightOffsetLeft, glm::vec3(-1.0f,
0.0f, 0.0f));
        }
        else {
            // Apagar el spotlight movi□ndolo muy lejos
            spotLights[2].SetFlash(glm::vec3(0.0f, -1000.0f, 0.0f), glm::vec3(0.0f, -
1.0f, 0.0f));
        }
```

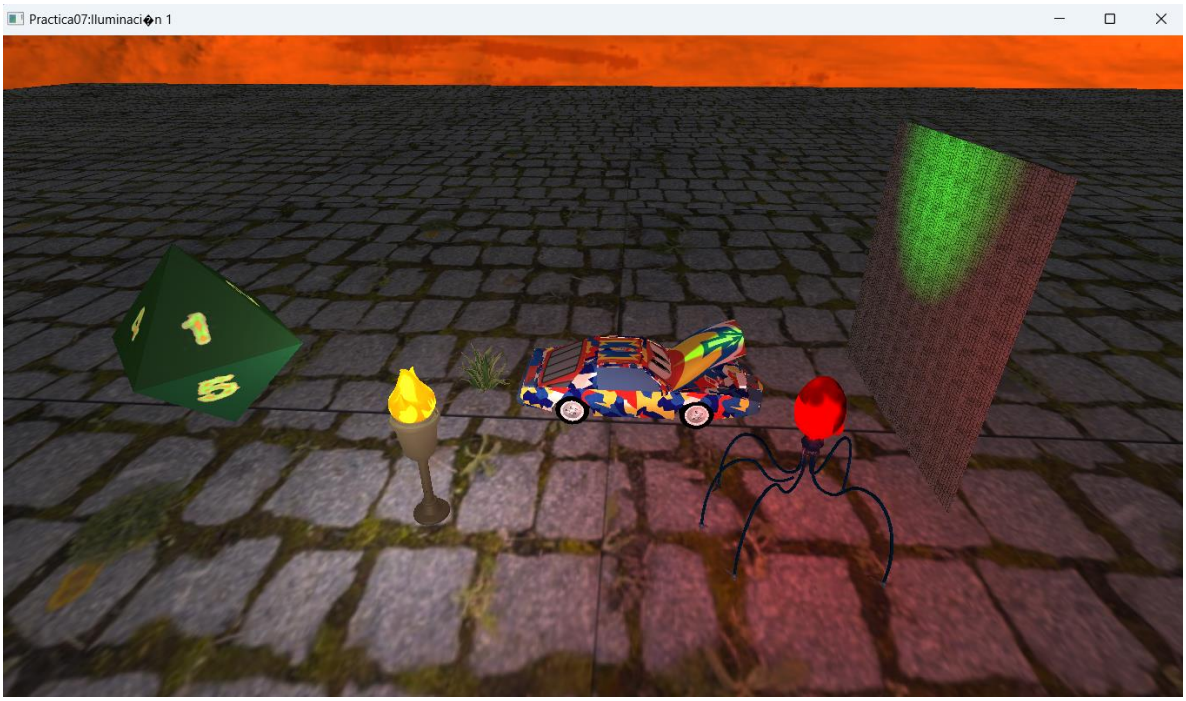
3. Agrega una luz puntual vinculada a algún modelo de tu elección (que no sea una lámpara). Esta luz y la luz de la lámpara deben poder encenderse y apagarse de forma independiente con el teclado. Si en el reporte 7 la lámpara fue creada como spotlight, cámbiala ahora a luz puntual.

Este ejercicio fue muy sencillo, ya que básicamente consistió en repetir lo que había hecho con la antorcha: cargar el modelo y agregarle la luz de tipo point light.

```
// ----------- Control de la luz de la antorcha con la tecla L ----------
        static bool luzAntorchaKeyPressed = false;
        if (mainWindow.getsKeys()[GLFW_KEY_L]) {
            if (!luzAntorchaKeyPressed) {
                luzAntorchaEncendida = !luzAntorchaEncendida;
                luzAntorchaKeyPressed = true;
            }
        }
        else {
            luzAntorchaKeyPressed = false;
        }

        // -------- Control de la luz del Alien con la tecla M ------------
        static bool luzAlienKeyPressed = false;
        if (mainWindow.getsKeys()[GLFW_KEY_M]) {
            if (!luzAlienKeyPressed) {
                luzAlienEncendida = !luzAlienEncendida;
                luzAlienKeyPressed = true;
            }
        }
        else {
            luzAlienKeyPressed = false;
        }

        // ------------------- CONFIGURACI□N DE LUCES --------------------
        unsigned int pointLightCount = 0;

        // ANTORCHA - siempre en posici□n 0
        if (luzAntorchaEncendida) {
            pointLights[pointLightCount] = PointLight(1.0f, 1.0f, 1.0f, // Color blanco
                2.0f, 1.0f, // Intensidad ambiental, intensidad difusa
                -10.0f, 9.0f, 10.0, // Vector de posicion
                1.0f, 0.09f, 0.032f); // ecuacion de segundo grado
            pointLightCount++;
        }

        // ALIEN - en la siguiente posici□n disponible
```

```
        if (luzAlienEncendida) {
            pointLights[pointLightCount] = PointLight(1.0f, 0.0f, 0.0f, // Color rojo
                2.0f, 1.0f, // Intensidad ambiental, intensidad difusa
                10.0f, 9.0f, 10.0, // Vector de posicion
                1.0f, 0.09f, 0.032f); // ecuacion de segundo grado
            pointLightCount++;
        }
```

Resultado Final:



Video:

https://drive.google.com/file/d/1qQmQThYJF4ADfYc7HOkWs_3Tpd_N_-1_/view?usp=sharing

Conclusión

Esta práctica me pareció algo complicada, sobre todo al momento de enlazar la luz al cofre para que se moviera con el ángulo correcto. Después de varios intentos, logré ajustar su posición y dirección usando funciones trigonométricas como cos y sin, lo que permitió que el spotlight siguiera correctamente el movimiento del cofre al abrirse y cerrarse. También implementé el sistema de luces delanteras y traseras que se activan automáticamente según la dirección del coche, utilizando banderas y controles con el teclado. Al final, el resultado fue satisfactorio, ya que pude integrar varios tipos de luces (spotlight y puntual) de forma dinámica y controlada, comprendiendo mejor cómo manipular su posición, dirección y comportamiento en función del movimiento de los objetos.