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 Nº8

NOMBRE COMPLETO: Araiza Valdés Diego Antonio

Nº de Cuenta: 423032833

GRUPO DE LABORATORIO: 02

GRUPO DE TEORÍA: 06

SEMESTRE: 2026-1

FECHA DE ENTREGA LÍMITE: 26/10/2025

CALIFICACIÓN: _____

REPORTE DE PRÁCTICA:

1. Ejecución de los ejercicios que se dejaron, comentar cada uno y capturas de pantalla de bloques de código generados y de ejecución del programa.
   1.1. Ejercicio 1: Agregar un spolight (que no sea luz de color blanco ni azul) que parta del cofre de su coche y al abrir y cerrar el cofre ilumine en esa dirección.
   1.2. Ejercicio 2: Agregar luz de tipo spotlight para el coche de tal forma que al avanzar (mover con teclado hacia dirección de X negativa) ilumine con un spotlight hacia adelante y al retroceder ((mover con teclado hacia dirección de X positiva) ilumine con un spotlight hacia atrás. Son dos spotlights diferentes que se prenderán y apagarán de acuerdo a alguna bandera asignada por ustedes.
   1.3. Ejercicio 3: Agregar otra luz de tipo puntual ligada a un modelo elegido por ustedes (no lámpara) y que puedan prender y apagar de forma independiente con teclado tanto la luz de la lámpara como la luz de este modelo (la luz de la lámpara debe de ser puntual, si la crearon spotlight en su reporte 7 tienen que cambiarla a luz puntual)

```
//LUZ ANILLO
pointLights[1] = PointLight(0.93725f, 0.72156f, 0.062745f,  //Color
    0.5f, 0.9f,                                             //Intensidad ambiental y difusa
    0.0f, 0.0f, 0.0f,                                       //Posición
    0.45f, 0.1f, 0.1f);                                     //Atenuación Constante, lineal y exponente
pointLightCount++;

//LUZ COFRE
spotLights[1] = SpotLight(1.0f, 0.96f, 0.176f, //Color
    10.0f, 10.0f,                              //Intensidad ambiental y difusa
    16.8f, 1.1f, 0.0f,                         //Posición
    1.0f, 0.0f, 0.0f,                          //Dirección
    0.1f, 0.0f, 0.05f,                         //Atenuación Constante, lineal y exponente
    25.0f);                                    //Edge
spotLightCount++;

//FARO DEL AUTO 1
spotLights[2] = SpotLight(0.0f, 0.0f, 1.0f,
    10.0f, 10.0f,
    18.8f, 1.0f, -5.28f,
    1.0f, 0.0f, 0.0f,
    0.0f, 0.0f, 0.05f,
    40.0f);
spotLightCount++;

//FARO DEL AUTO 2
spotLights[3] = SpotLight(0.0f, 0.0f, 1.0f, //Color
    10.0f, 10.0f,                           //Intensidad ambiental y difusa
    18.8f, 1.0f, 5.28f,                     //Posición
    1.0f, 0.0f, 0.0f,                       //Dirección
    0.0f, 0.0f, 0.05f,                      //Atenuación Constante, lineal y exponente
    40.0f);                                 //Edge
spotLightCount++;
```

Primero creamos la luz del anillo con un pointlight de un color “dorado”, con un spotlight creamos la luz del cofre, de un color amarillo, mirando hacia el eje X positivo y por último creamos las luces de los faros. Solo necesitamos 2 para representar las 4 luces del auto.

```
//SpotLight pointLightstemp;
PointLight pointLightstemp[MAX_POINT_LIGHTS];
GLint m = 0;
//SpotLight spotLightstemp;
SpotLight spotLightstemp[MAX_SPOT_LIGHTS];
GLint n = 0;
```

Luego creamos 2 arreglos temporales que nos permitirán mandar al shader solo las luces que deseamos.

```
    //LUZ LINTERNA
    spotLightstemp[n] = spotLights[0];
    n++;
    //LUZ COFRE
n = 2;  spotLightstemp[n] = spotLights[1];
m = 0;  n++;
```

Inicializamos el conteo de luces spotlight en 2 (ya que la luz del cofre y la linterna nunca se apagan) y el de las luces pointlight en 0 dentro del while, además, pasamos al arreglo que mandaremos al shader las 2 luces.

```
    GLfloat getmuevex() { return muevex; }
    GLint getDirection() { return direction; }
    GLfloat getmuevex2() { return muevex2; }
    bool getShouldClose() {
        return  glfwWindowShouldClose(mainWindow);}
    bool* getsKeys() { return keys; }
    void swapBuffers() { return glfwSwapBuffers(mainWindow); }
    GLfloat getsetPattern() { return setpattern; }
    GLfloat getarticulacion6() { return articulacion6; }
    GLboolean getOnLigth() { return onlight;  }
    GLboolean getOnLigth2() { return onlight2; }

    ~Window();
private:
    GLFWwindow *mainWindow;
    GLint width, height;
    GLfloat setpattern, articulacion6;
    GLboolean onlight, onlight2;
// --- variables para el control (ping-pong) de articulacion6 ---
GLfloat articulacion6Dir;   // 1.0f = incrementar, -1.0f = decrementar
GLfloat articulacion6Min;   // límite mínimo
GLfloat articulacion6Max;   // límite máximo
GLfloat articulacion6Step;  // paso por pulsación

//Bandera muevex
GLint direction;
```

Luego, dentro de Window.h declaramos los métodos y variables que usaremos para mover el auto, el cofre, y prender y apagar las luces.

```
articulacion6 = 0.0f;
onlight = true;
onlight2 = true;
direction = 0;

// Inicialización de control ping-pong para articulacion6
articulacion6Dir = 1.0f;    // empieza bajando
articulacion6Min = 0.0f;
articulacion6Max = 45.0f;
articulacion6Step = 5.0f;  // +5° por pulsación
```

```
if (key == GLFW_KEY_C)
{
    float next = theWindow->articulacion6 + theWindow->articulacion6Dir * theWindow->articulacion6Step;

    if (next > theWindow->articulacion6Max) {
        theWindow->articulacion6 = theWindow->articulacion6Max;
        theWindow->articulacion6Dir = -1.0f; // invertir dirección para la siguiente pulsación
    }
    else if (next < theWindow->articulacion6Min) {
        theWindow->articulacion6 = theWindow->articulacion6Min;
        theWindow->articulacion6Dir = 1.0f;
    }
    else {
        theWindow->articulacion6 = next;
    }
}
```

Después, dentro de Window.cpp, inicializamos las variables y reutilizamos la función para mover el cofre de las prácticas pasadas.

```
glm::vec3 lightWorldDir;
glm::vec3 lightWorldPos;
```

```
//COFRE
model = modelaux;
model = glm::translate(model, glm::vec3(5.5846875f, 1.8315f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion6()), glm::vec3(0.0f, 0.0f, 1.0f));
lightWorldPos = glm::vec3(model * glm::vec4(10.0f, -1.9f, 0.0f, 1.0f));
lightWorldDir = glm::normalize(glm::mat3(model) * glm::vec3(1.0f, 0.0f, 0.0f));
spotLights[1].SetFlash(lightWorldPos, lightWorldDir);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glEnable(GL_BLEND);
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
F1Cofre_M.RenderModel();
```

Para terminar el primer ejercicio, declaramos el coche, y a 2 vectores declarados previamente fuera del while, les asignamos la posición (multiplicando la matriz model por el vector de traslación al punto deseado), y la dirección (multiplicando la matriz sin la traslación por el vector de dirección), le damos estas propiedades a la luz, activamos el canal Alpha y le creamos el modelo.

```
if (key == GLFW_KEY_Y)
{
    theWindow-> muevex += 1.0;
    theWindow->direction = 1;
}
else if (key == GLFW_KEY_U)
{
    theWindow-> muevex -= 1.0;
    theWindow->direction = -1;
}
if ((key == GLFW_KEY_Y && action == GLFW_RELEASE) || (key == GLFW_KEY_U && action == GLFW_RELEASE)){
    theWindow->direction = 0;
}
```

Para el ejercicio 2, dentro de Window.cpp se agregó al funcionamiento de muevex un cambio de valor en dirección para saber el valor del auto y se agrego un método que nos permita saber en que momento se suelta una u otra tecla.

```
//FAROS
if (mainWindow.getDirection() > 0) {
    spotLights[2].SetFlash(glm::vec3(16.8 + modelaux[3].x, -1.08f + modelaux[3].y, -5.28 + modelaux[3].z), glm::vec3(1.0f, 0.0f, 0.0f));
    spotLights[3].SetFlash(glm::vec3(16.8 + modelaux[3].x, -1.08f + modelaux[3].y, 5.28 + modelaux[3].z), glm::vec3(1.0f, 0.0f, 0.0f));
    spotLightstemp[n] = spotLights[2];
    n++;
    spotLightstemp[n] = spotLights[3];
    n++;
}
else if (mainWindow.getDirection() < 0) {
    spotLights[2].SetFlash(glm::vec3(-16.8 + modelaux[3].x, -1.08f + modelaux[3].y, -3.0 + modelaux[3].z), glm::vec3(-1.0f, 0.0f, 0.0f));
    spotLights[3].SetFlash(glm::vec3(-16.8 + modelaux[3].x, -1.08f + modelaux[3].y, 3.0 + modelaux[3].z), glm::vec3(-1.0f, 0.0f, 0.0f));
    spotLightstemp[n] = spotLights[2];
    n++;
    spotLightstemp[n] = spotLights[3];
    n++;
}
```

Así, dentro del while se comprueba el valor de la dirección y dependiendo del mismo cambiamos la dirección y posición de los faros, relativos al cuerpo del vehículo y se los mandamos al shader temporal aumentando el contador de luces.

```
if (key == GLFW_KEY_O && action == GLFW_PRESS)
{
    theWindow->onlight = !theWindow->onlight;
}

if (key == GLFW_KEY_P && action == GLFW_RELEASE)
{
    theWindow->onlight2 = !theWindow->onlight2;
}
```

Para el tercer ejercicio, clonamos la función utilizada en el ejercicio 8, lo que nos permite tener banderas independientes.

```
//ANTORCHA
if (mainWindow.getOnLigth())
{
    pointLightstemp[m] = pointLights[0];
    m++;
}
//ANILLO
if (mainWindow.getOnLigth2())
{
    pointLightstemp[m] = pointLights[1];
    m++;
}
```

Posteriormente, dentro del while, comprobamos el valor de estas y si este es verdadero las mandamos al shader y aumentamos el contador de luces.

```
//TORCH
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, -1.0f, -10.0f));
modelaux = model;
pointLights[0].SetPos(glm::vec3(modelaux[3].x, 4.1f + modelaux[3].y, modelaux[3].z));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Torch_M.RenderModel();
glDisable(GL_BLEND);

//ANILLO
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(40.0f, 1.65f, 0.0f));
modelaux = model;
pointLights[1].SetPos(glm::vec3(modelaux[3].x, modelaux[3].y, modelaux[3].z));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Anillo_M.RenderModel();
```

Finalmente, arreglamos declaramos el anillo, y le pasamos jerárquicamente la posición a la luz correspondiente al mismo.

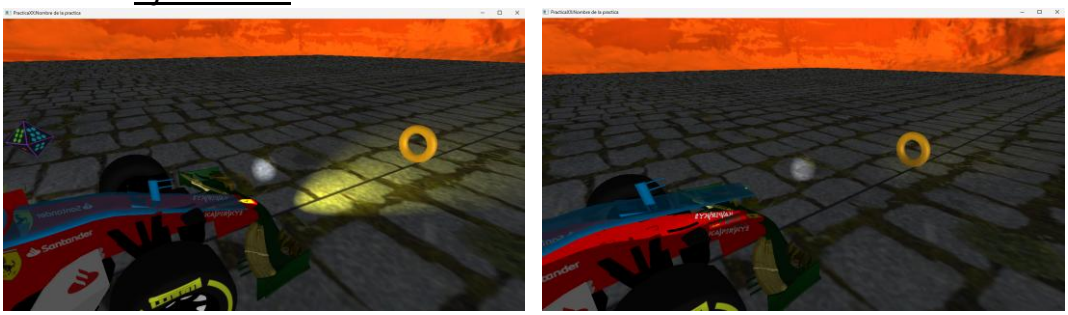
```
void SetPos(glm::vec3 pos);
```

```
void PointLight::SetPos(glm::vec3 pos)
{
    position = pos;
}
```

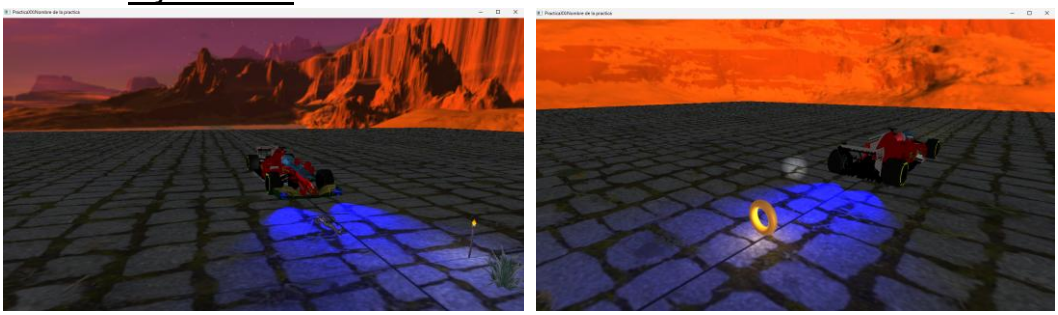
Para esto, declaramos el método SetPos a la luz pointlight

Evidencias:

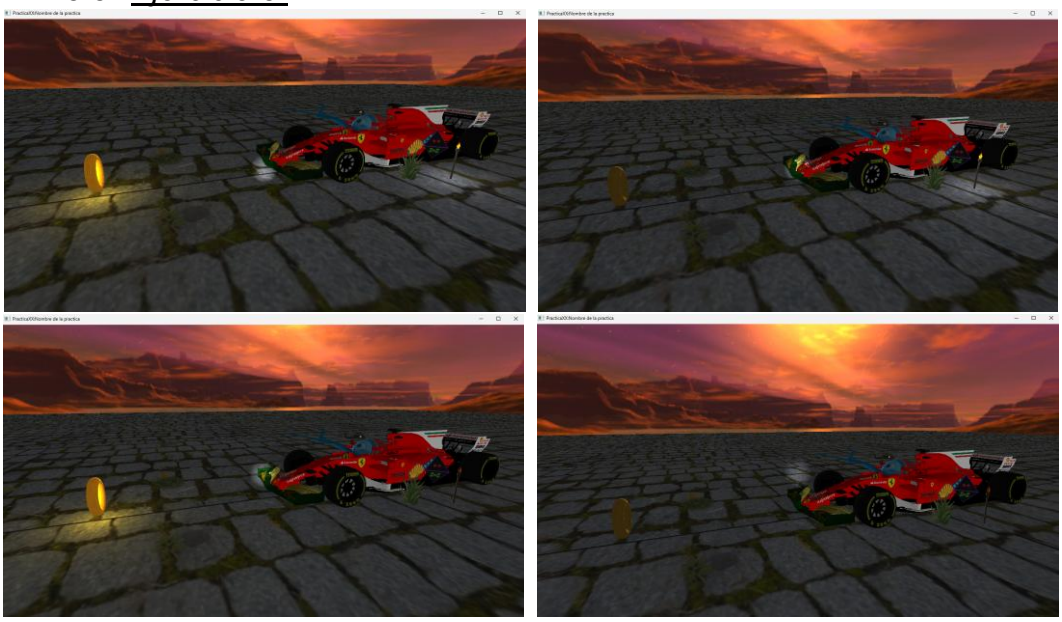
1.3.1. *Ejercicio 1:*



1.3.2. *Ejercicio 2:*



1.3.3. *Ejercicio 3:*



1. Liste los problemas que tuvo a la hora de hacer estos ejercicios y si los resolvió explicar cómo fue, en caso de error adjuntar captura de pantalla
   R: No hubo problemas.

2. Conclusión:
   1. Los ejercicios del reporte: Complejidad, Explicación.
      Respecto a los ejercicios realizados, fueron bastante adecuados a lo visto en clase sin embargo si fue algo complicado entender como hacer que la luz del cofre rotara y se moviera junto con él.

   2. Comentarios generales: Faltó explicar a detalle, ir más lento en alguna explicación, otros comentarios y sugerencias para mejorar desarrollo de la práctica.
      La explicación fue muy buena, por lo que el prender y apagar una luz, así como su jerarquía no hubo problema, sin embargo, me hubiera gustado ver un ejemplo más sencillo ya que el que vimos y la manera en que realicé el ejercicio solo funciona para cuando las luces que prenderemos están al final por lo que tuve que buscar otra forma de hacerlo.

   3. Conclusión
      Personalmente creo que se cumplieron los objetivos de la práctica. Con el ejercicio y la práctica aprendí la importancia de establecer correctamente las normales en los objetos y mejoré mi conocimiento en el manejo de modelos e iluminación. Además, aprendí a reutilizar luces (cambiando su dirección, y posición dinámicamente) y a cambiar la posición y dirección de otra cuando un objeto es rotado.