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: MORENO SANTOYO MARIANA

Nº de Cuenta: 319170252

GRUPO DE LABORATORIO: 11

GRUPO DE TEORÍA: 04

SEMESTRE 2025-2

FECHA DE ENTREGA LÍMITE: 09/04/2025

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.

2.- 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

3.- Conclusión:

a. *Los ejercicios del reporte: Complejidad, Explicación.*
b. *Comentarios generales: Faltó explicar a detalle, ir más lento en alguna explicación, otros comentarios y sugerencias para mejorar desarrollo de la práctica*
c. *Conclusión*

Bibliografía en formato APA

1. *Agregar un foco (que no sea luz de color blanco ni azul) que parte del cofre de su coche y al abrir y cerrar el cofre ilumina en esa dirección.*

Para evitar la necesidad de agregar una pared u otro elemento genérico, y con el objetivo de importar modelos que mantuvieran coherencia temática con mi personaje Pánico de Hércules, se optó por importar un modelo del propio Hércules. Este se utilizó para generar iluminación mediante el cofre del auto

```
Model herc;
```

```
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, -1.0f, 20.0f));
model = glm::scale(model, glm::vec3(2.5f, 2.5f, 2.5f));
model = glm::rotate(model, glm::radians(-180.0f), glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
herc.RenderModel();
```

Se generó un spotlight de color ámbar con el objetivo de no alterar en demasía el color original del modelo, preservando así su estética y coherencia visual.

```
spotLights[2] = SpotLight(1.0f, 0.85f, 0.65f, // color
    0.5f, 2.0f,
    0.0f, 1.5f, 1.0f,
    0.0f, 0.0f, 1.0f,
    1.0f, 0.0f, 0.0f,
    5.0f);
spotLightCount++;
```

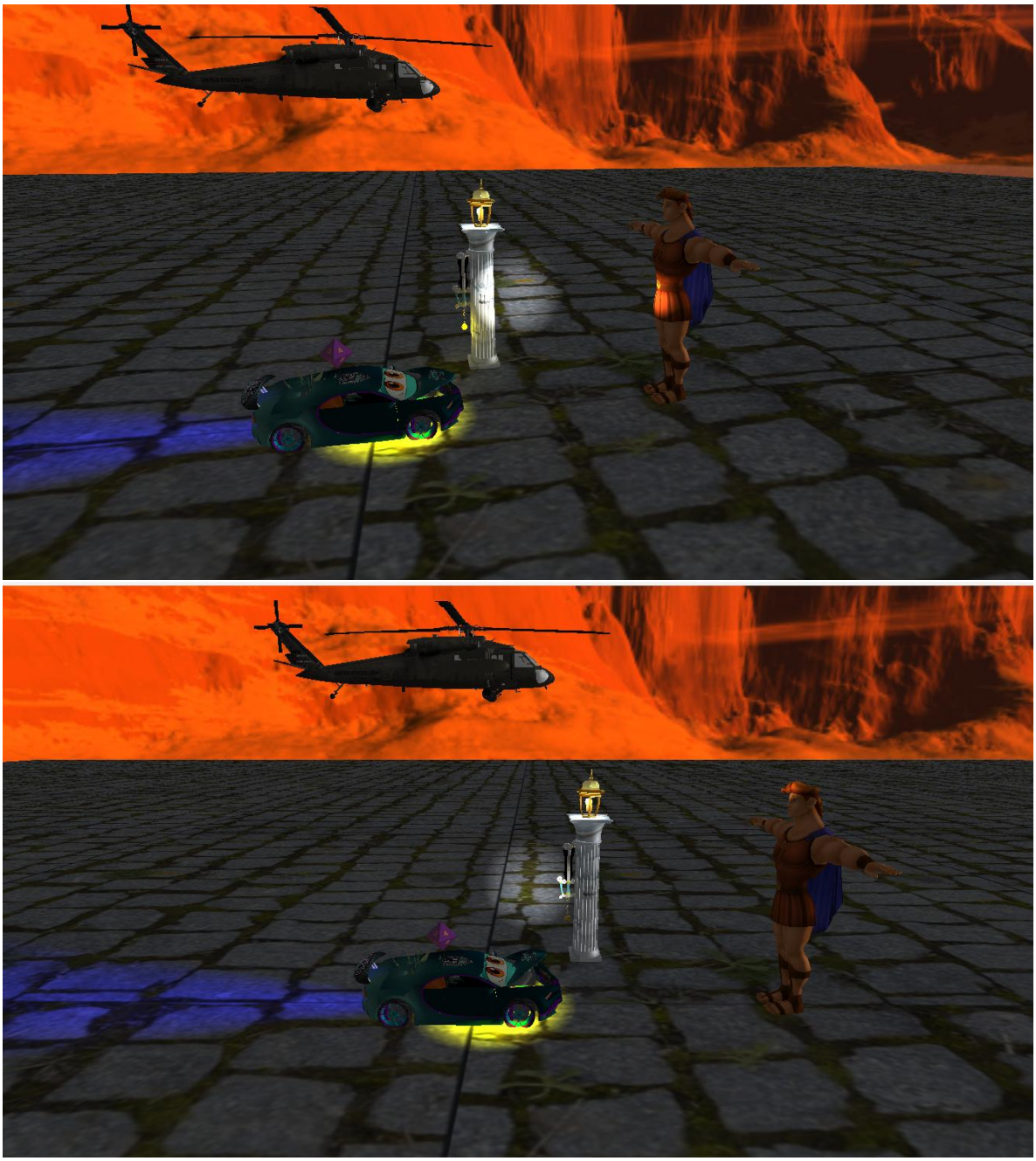
Basándose tanto en el movimiento de la luz del ejercicio de la práctica anterior como en la dirección del haz de la linterna, se llegó a la siguiente configuración para el presente ejercicio.

```
float a3 = mainWindow.getarticulacion3();
glm::mat4 rotation = glm::rotate(glm::mat4(1.0f), glm::radians(a3), glm::vec3(0.1f, 0.0f, 0.0f));
glm::vec3 lightPosition3 = glm::vec3(0.0f, 0.0f, 1.0f) + glm::vec3(0.0f, 0.0f, 0.1f) * (a1 + a2);
glm::vec3 lightDirection3 = glm::normalize(glm::vec3(rotation * glm::vec4(0.0f, 0.0f, 0.1f, 0.0f)));
spotLights[2].SetFlash(lightPosition3, lightDirection3);
```

Se uso la articulación tres que ya se tenia de ejercicios anteriores:

```
if (key == GLFW_KEY_K)
{
    if (theWindow->articulacion3 > -10)
    {
    }
    else
    {
        theWindow->articulacion3 += 10.0;
    }
}
if (key == GLFW_KEY_L)
{
    if (theWindow->articulacion3 < -40)
    {
    }
    else
    {
        theWindow->articulacion3 -= 10.0;
    }
}
```

EJECUCIONES



2. *Agregar luz de tipo foco para el coche de tal forma que al avanzar (mover con teclado hacia dirección de X negativa) ilumine con un foco hacia adelante y al retroceder (mover con teclado hacia dirección de X positiva) ilumine con un foco hacia atrás. Son dos focos diferentes que se prenderán y apagarán de acuerdo con alguna bandera asignada por ustedes.*

Se creo un segundo arreglo de luces spotlight para ir alternando luz delantera y la luz trasera.

```
SpotLight spotLights[MAX_SPOT_LIGHTS];
SpotLight spotLights2[MAX_SPOT_LIGHTS];
```

```
unsigned int spotLightCount = 0;
unsigned int spotLightCount2 = 0;

//linterna
spotLights[0] = SpotLight(1.0f, 1.0f, 1.0f,
    0.0f, 2.0f,
    0.0f, 0.0f, 0.0f,
    0.0f, -1.0f, 0.0f,
    1.0f, 0.0f, 0.0f,
    5.0f);
spotLightCount++;
spotLights2[0] = spotLights[0]; // Copia la luz de la linterna al segundo arreglo
spotLightCount2++;

//helicoptero
spotLights[1] = SpotLight(0.8f, 0.8f, 0.0f, // color    // Color amarillo chillón (RGB)
    0.5f, 0.1f,           // aIntensity y dIntensity
    0.0f, 35.0f, 0.0f,     // Posición del helicóptero en el centro y a 5 unidades de altura
    0.0f, -10.0f, 0.0f,    // La luz apunta hacia abajo (eje Y negativo)
    1.0f, 0.0f, 0.00f,
    30.0f);
spotLightCount++;
spotLights2[1] = spotLights[1]; // Copia la luz del helicóptero al segundo arreglo
spotLightCount2++;
//luz de la estatua
spotLights[2] = SpotLight(1.0f, 0.85f, 0.65f, // color
    0.5f, 2.0f,
    0.0f, 1.5f, 1.0f,
    0.0f, 0.0f, 1.0f,
    1.0f, 0.0f, 0.0f,
    5.0f);
spotLightCount++;
spotLights2[2] = spotLights[2];
spotLightCount2++;

//auto
spotLights[3] = SpotLight(0.0f, 0.0f, 1.0f, // color
    0.5f, 2.0f,
    2.1f, 0.6f, 3.0f,
    0.0f, 0.0f, 1.0f,
    1.0f, 0.0f, 0.0f,
    15.0f);
spotLightCount++;

spotLights2[3] = SpotLight(0.0f, 0.0f, 1.0f, // color
    0.5f, 2.0f,
    2.1f, 0.6f, -3.0f,
    0.0f, 0.0f, -1.0f,
    1.0f, 0.0f, 0.0f,
    15.0f);
spotLightCount2++;

spotLights[4] = SpotLight(0.0f, 0.0f, 1.0f, // color
    0.5f, 2.0f,
    2.1f, 0.6f, -3.0f,
    0.0f, 0.0f, -1.0f,
    1.0f, 0.0f, 0.0f,
    15.0f);
spotLightCount++;

spotLights2[4] = SpotLight(0.0f, 0.0f, 1.0f, // color
    0.5f, 2.0f,
    2.1f, 0.6f, 3.0f,
    0.0f, 0.0f, 1.0f,
    1.0f, 0.0f, 0.0f,
    15.0f);
spotLightCount2++;
```

Se crea el siguiente condicional if para definir la posición tanto de la luz trasera como de la luz frontal basado en el movimiento que ya se tenia previamente implementado.

```
float a1 = mainWindow.getarticulacion1();
float a2 = mainWindow.getarticulacion2();
//glm::vec3 lightPosition = glm::vec3(0.0f, 0.0f, 0.0f) + glm::vec3(0.0f, 0.0f, 0.1f) * (a1 + a2);

if (mainWindow.getvalor3() == 1.0f)
{
    glm::vec3 lightPosition = glm::vec3(0.0f, 0.0f, 0.0f) + glm::vec3(0.0f, 0.0f, 0.1f) * (a1 + a2);
    spotLights[3].SetFlash(lightPosition, glm::vec3(0.0f, 0.0f, 1.0f));
}
else if (mainWindow.getvalor3() == 0.0f)
{
    glm::vec3 lightPosition = glm::vec3(0.0f, 0.0f, 0.0f) + glm::vec3(0.0f, 0.0f, 0.1f) * (a1 + a2);
    spotLights2[3].SetFlash(lightPosition, glm::vec3(0.0f, 0.0f, -1.0f));
}
```

Se manda llamar al shader con el siguiente condicional para alternar entre ambos arreglos de luces ya sea un caso default o luz trasera o delantera

```
if (mainWindow.getvalor3() == 1.0f)
{
    shaderList[0].SetSpotLights(spotLights, spotLightCount-1);
}
else if (mainWindow.getvalor3() == 0.0f)
{
    shaderList[0].SetSpotLights(spotLights2, spotLightCount2-1);
}
else
{
    shaderList[0].SetSpotLights(spotLights, spotLightCount);
}
```

Para realizar los dos condicionales anteriores fue necesario implementar la siguiente bandera de dentro de las teclas que mandan llamar a frontal o trasero, no se implementaron booleanos para evitar meterse en las cuestiones de conversión de tipos de datos.

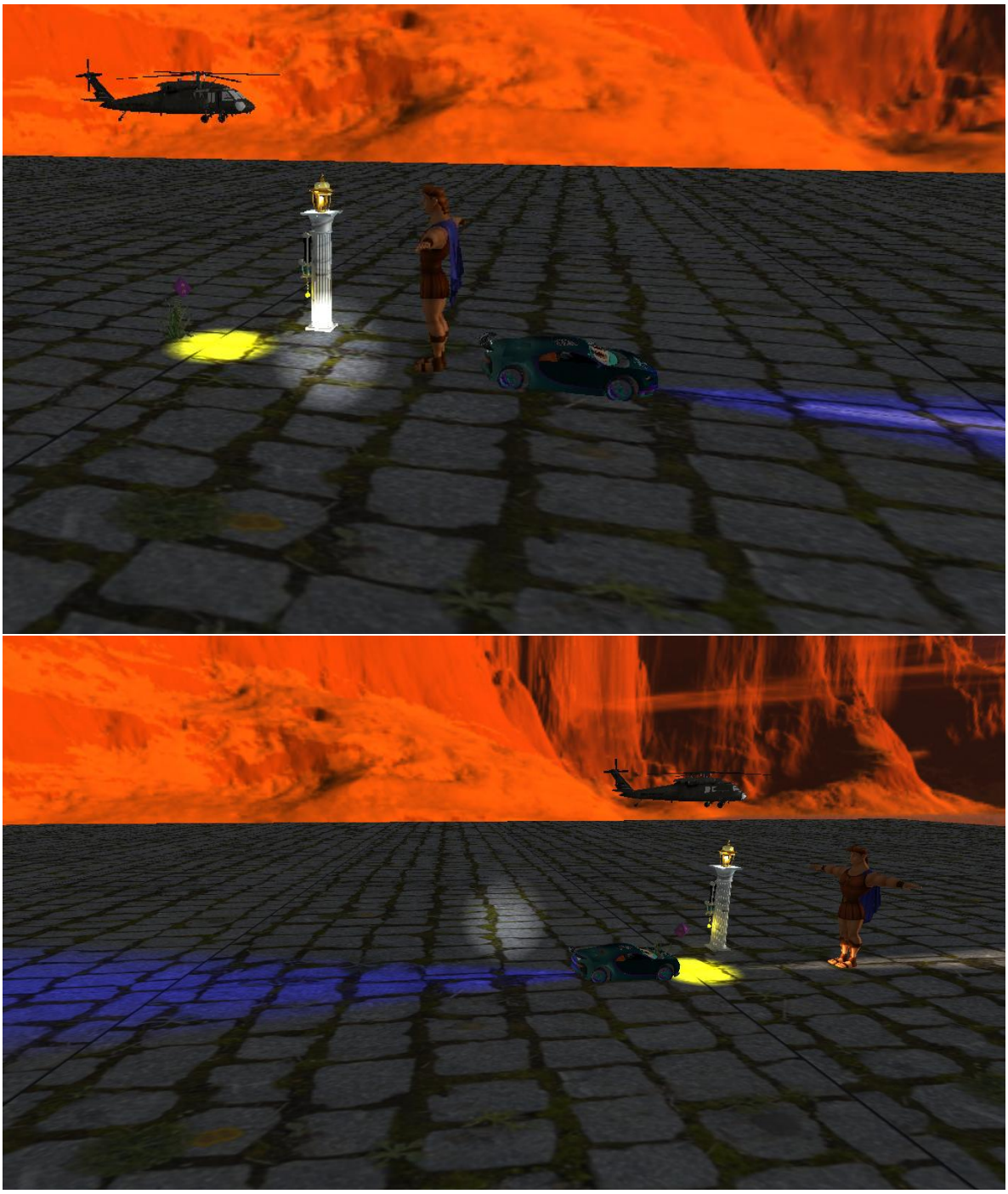
```
GLfloat getvalor3() { return valor3; }
```

```
GLfloat rotax, rotay, rotaz, articulacion1, articulacion2, articulacion3,valor,articulacion4, articulacion5,valor2 , valor3;
```

```
valor3 = 0.0f;
```

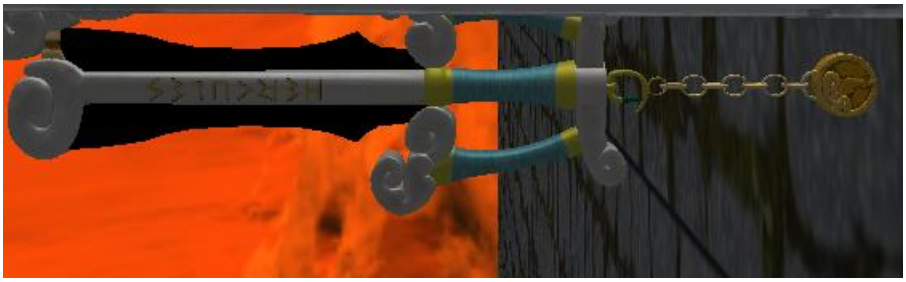
```
if (key == GLFW_KEY_F)
{
    theWindow->articulacion1 += 10.0;
    theWindow->valor3 = 1.0;

}
if (key == GLFW_KEY_G)
{
    theWindow->articulacion2 -= 10.0;
    theWindow->valor3 = 0.0;

}
```

EJECUCIONES



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 foco en su reporte 7 tienen que cambiarla a luz puntual) atrás.*

Para evitar la necesidad de crear un modelo abstracto o uno que entrara en conflicto con la temática actual, se eligió utilizar la medalla de Hércules, acompañada de una luz dorada, con el fin de mantener la coherencia estética del proyecto.



```
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(8.0f, 5.0f,6.5f));
model = glm::scale(model, glm::vec3(0.7f, 0.7f, 0.7f));
model = glm::rotate(model, glm::radians(-90.0f), glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
medal.RenderModel();
```

Se creó un segundo arreglo de point lights para contemplar todos los casos posibles y facilitar la realización de combinaciones de manera sencilla.

```
PointLight pointLights1[MAX_POINT_LIGHTS];
PointLight pointLights2[MAX_POINT_LIGHTS];
```

```
unsigned int pointLightCount_ARRAY1 = 0;

pointLights1[0] = PointLight(1.0f, 0.843f, 0.0f,
    0.3f, 0.5f,
    7.70f, 2.3f, 6.0f,
    0.0f, 0.1f, 0.5f);
pointLightCount_ARRAY1++;

pointLights1[1] = PointLight(1.0f, 1.0f, 1.0f,
    0.25f, 2.0f,
    7.70f, 11.70f, 7.800f,
    0.0f, 0.1f, 0.5f);
pointLightCount_ARRAY1++;

unsigned int pointLightCount_ARRAY2 = 0;

pointLights2[0] = PointLight(1.0f, 1.0f, 1.0f,
    0.25f, 2.0f,
    7.70f, 11.70f, 7.800f,
    0.0f, 0.1f, 0.5f);
pointLightCount_ARRAY2++;

pointLights2[1] = PointLight(1.0f, 0.843f, 0.0f,
    0.3f, 0.5f,
    7.70f, 2.3f, 6.0f,
    0.0f, 0.1f, 0.5f);
pointLightCount_ARRAY2++;
```

Se implementaron los diferentes casos con un switch case para evitar anidar if tanto:

```
int state = 0;

if (mainWindow.getvalor() == 0.0f && mainWindow.getvalor2() == 0.0f)
    state = 0;
else if (mainWindow.getvalor() == 1.0f && mainWindow.getvalor2() == 0.0f)
    state = 1;
else if (mainWindow.getvalor() == 0.0f && mainWindow.getvalor2() == 1.0f)
    state = 2;
else if (mainWindow.getvalor() == 1.0f && mainWindow.getvalor2() == 1.0f)
    state = 3;

switch (state)
{
case 0:
    // Situación inicial: ambas luces apagadas (o en estado base)
    shaderList[0].SetPointLights(pointLights1, pointLightCount_ARRAY1);
    break;
case 1:
    // Sólo la medalla encendida
    shaderList[0].SetPointLights(pointLights1, pointLightCount_ARRAY1 - 1);
    break;
case 2:
    // Sólo la luz del faro encendida
    shaderList[0].SetPointLights(pointLights2, pointLightCount_ARRAY2 - 1);
    break;
case 3:
    // Ambas luces encendidas / o apagadas según la lógica que desees
        shaderList[0].SetPointLights(pointLights1, pointLightCount_ARRAY1 - 2);
    break;
default:
    break;
}
```

Para la implementación se creo la siguiente bandera para seleccionar los diferentes casos

```
GLfloat getvalor() { return valor; }
GLfloat getvalor2() { return valor2; }
```

```
GLfloat rotax, rotay, rotaz, articulacion1, articulacion2, articulacion3,valor,articulacion4, articulacion5,valor2 , valor3;
```

```
valor = 0.0f;
articulacion4 = 0.0f;
articulacion5 = 0.0f;
valor2 = 0.0f;
valor3 = 0.0f;
```

Para implementar una funcionalidad en la que las lámparas se enciendan y se apaguen de forma inmediata con la misma tecla, se desarrolló la siguiente lógica. Esto se hizo con el objetivo de evitar que las teclas se reiniciaran y entraran en un bucle no deseado. (Para usar menos teclas).

```
static bool hPressed = false;
if (key == GLFW_KEY_H && action == GLFW_PRESS) {
    if (!hPressed) {
        if (theWindow->valor == 0.0f)
            theWindow->valor = 1.0f;
        else if (theWindow->valor == 1.0f)
            theWindow->valor = 0.0f;
        hPressed = true;
    }
}
if (key == GLFW_KEY_H && action == GLFW_RELEASE) {
    hPressed = false;
}

static bool hPressed2 = false;
if (key == GLFW_KEY_J && action == GLFW_PRESS) {
    if (!hPressed2) {
        if (theWindow->valor2 == 0.0f)
            theWindow->valor2 = 1.0f;
        else if (theWindow->valor2 == 1.0f)
            theWindow->valor2 = 0.0f;
        hPressed2 = true;
    }
}
if (key == GLFW_KEY_J && action == GLFW_RELEASE) {
    hPressed2 = false;
}
```

EJECUCIONES

Conclusiones y Comentarios

A lo largo del desarrollo del ejercicio, se buscó mantener una coherencia temática con el personaje de Pánico de la película Hércules, evitando el uso de modelos genéricos o abstractos que pudieran romper la estética del entorno. Para ello, se importaron modelos directamente relacionados, como el de Hércules, y se utilizaron elementos como la medalla del personaje, la cual fue iluminada con una luz dorada para reforzar la identidad visual del proyecto.

Se implementó un spotlight color ámbar que parte del cofre del coche del personaje, diseñado para iluminar únicamente cuando el cofre está abierto. Esta decisión se tomó con la intención de no alterar significativamente el color original del modelo, conservando su integridad visual. Además, inspirados en los ejercicios previos de la práctica, se aplicaron conceptos de movimiento de luces y dirección para configurar adecuadamente la interacción de las luces con los elementos del entorno.

En cumplimiento de los requerimientos, se añadieron dos luces tipo spotlight que responden al movimiento del vehículo: una que se activa al avanzar (movimiento en dirección X negativa) y otra al retroceder (dirección X positiva). Ambas luces fueron programadas con banderas que controlan su encendido y apagado de forma lógica, asegurando un comportamiento adecuado y realista en cada dirección.

Adicionalmente, se creó un segundo arreglo de luces puntuales (point lights) para abarcar múltiples casos posibles y permitir combinaciones diversas de manera sencilla. Una de estas luces fue asignada a un modelo distinto a una lámpara —cumpliendo así con los requerimientos del reporte— y puede ser controlada de manera independiente mediante teclado, al igual que la luz puntual de la lámpara previamente modificada desde su configuración de foco.

Finalmente, para evitar bucles no deseados en la activación y desactivación de las luces mediante teclado, se diseñó una lógica que permite alternar el estado de las luces con una sola tecla, asegurando así una experiencia de usuario fluida y sin errores de reinicio

BIBLIOGRAFIA

Sketchfab. (n.d.). Hercules beetle. https://sketchfab.com/3d-models/hercules-e228580a4a2242d2a24b25106dee1b9d

Sketchfab. (n.d.). Hercules. https://sketchfab.com/3d-models/hercules-medallion-da31bc8502a94b988f79b0f2fead2b31