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: García Soto Jean Carlo

Nº de Cuenta: 319226304

GRUPO DE LABORATORIO: 03

GRUPO DE TEORÍA: 04

SEMESTRE 2025-2

FECHA DE ENTREGA LÍMITE: 12/04/2025

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.

I. 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.

Primero se crea el spotlight de color verde en este caso, el cual le asignamos una atenuación y la posición relativa donde se encuentra el cofre y queremos que esta simule que sea una luz del cofre.

```
387    //luz verde para el cofre
388    spotLights[1] = SpotLight(0.0f, 1.0f, 0.0f,
389        15.0f, 1.0f,
390        -14.0f, 0.5f, -15.0f,        //Posición
391        0.0f, -1.0f, 0.0f,           //Vector de dirección
392        1.0f, 0.5f, 0.1f,            //Atenuación (No poner en ceros)
393        25.0f);
394    spotLightCount++;
```

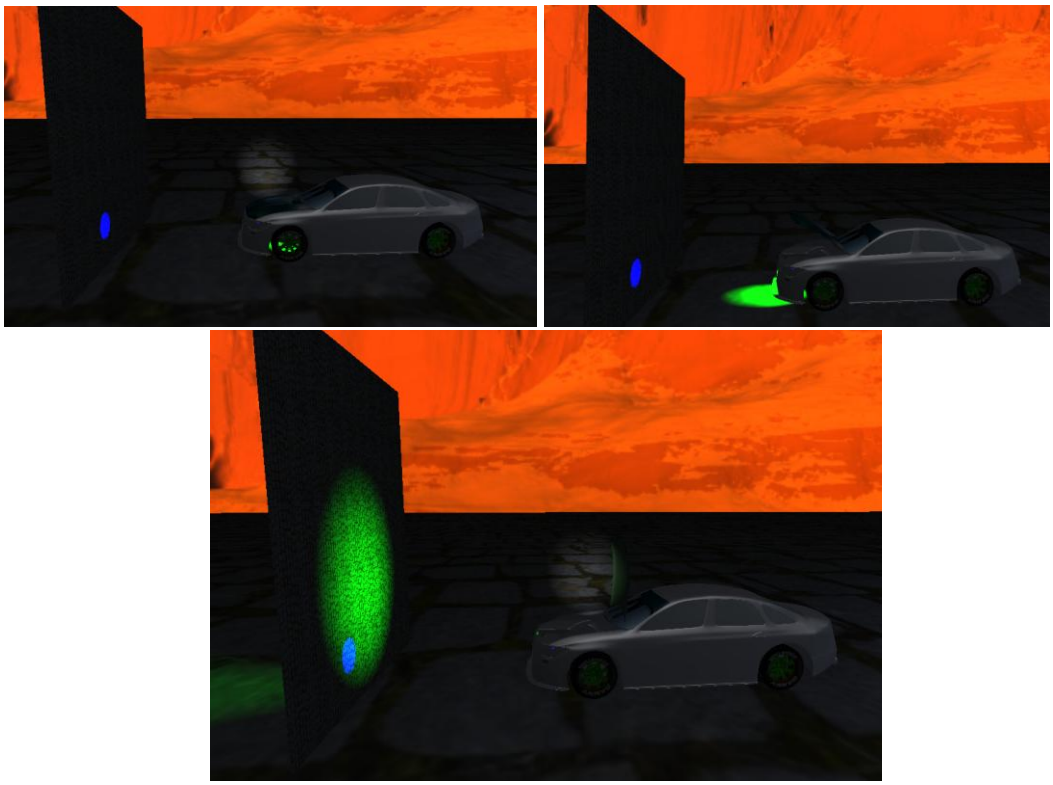
Después agregamos las siguientes instrucciones para que cambie su ángulo de acuerdo como rote el cofre, y dado que el carro puede avanzar y retroceder, le pasamos la translación por medio de la tecla de cuando se mueve el carro. Para las funciones de rotación y traslación se usaron las asignadas al carrito, es decir una que ya se había creado en anteriores prácticas.

```
494    //Mueve Luz verde del cofre de acuerdo a como rote el cofre     teclas V y B
495    glm::mat4 rotacionLuz = glm::rotate(glm::mat4(1.0f), glm::radians(mainWindow.getrotaCofre()), glm::vec3(0.0f, 0.0f, 1.0f));
496    glm::vec3 mueveLuz4 = glm::vec3(-13.0, 0.99, -10) + glm::vec3(mainWindow.getTransladaCarro(), 0.0f, 0.0f);      //Tecla 1 y 2
497    glm::vec3 direccion = glm::normalize(glm::vec3(rotacionLuz * glm::vec4(0.0f, -0.1f, 0.0f, 0.0f)));
498    spotLights[1].SetFlash(mueveLuz4, direccion);
```

Ya por último ponemos una pared enfrente del carro para poder apreciar mejor la luz del cofre cuando no apunta al cofre sino adelante, se reutilizó el modelo del piso y solo se le modificó la orientación y la escala para que sea una pared pequeña y que no abarque una gran área.

```
677    //Pared
678    model = glm::mat4(1.0);
679    model = glm::translate(model, glm::vec3(-20.0f, -1.0f, -10.0f));
680    model = glm::scale(model, glm::vec3(0.5f, 1.0f, 0.5f));
681    model = glm::rotate(model, -90 * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
682    glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
683    glUniform3fv(uniformColor, 1, glm::value_ptr(color));
684    pisoTexture.UseTexture();
685    Material_opaco.UseMaterial(uniformSpecularIntensity, uniformShininess);
686    meshList[2]->RenderMesh();
```

Para la salida se modificó momentáneamente el ángulo de apertura de la luz azul del carro, para que se pudiera observar mejor la luz del cofre.



II. 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 con alguna bandera asignada por ustedes.

Se modificó primero la cantidad de spotlights de 4 a 5 en el archivo ***CommonValues.h*** y en ***shader_light.frag***, con el fin de no eliminar ninguna luz ya creada.

```
5   const int MAX_POINT_LIGHTS = 3;
6   const int MAX_SPOT_LIGHTS = 5;
7   #endif
```

```
const int MAX_POINT_LIGHTS = 3;
const int MAX_SPOT_LIGHTS = 5;
```

Se crea otro arreglo spotlight llamado ***spotLights2***, esto para poder implementar el prendido y apagado de las luces en cuestión.

```
89    SpotLight spotLights[MAX_SPOT_LIGHTS];          //5
90    SpotLight spotLights2[MAX_SPOT_LIGHTS];         //5
```

En el primer spotlight se agrega una luz roja que será para el faro trasero del carro, y el azul se deja como está. Y a la hora de crear el segundo arreglo, intercambiamos de lugar las luces roja y azul.

```
404
405     //se crean mas luces puntuales y spotlight
406     //Luz azul (Parte delantera del coche)
407     spotLights[3] = SpotLight(0.0f, 0.0f, 1.0f,
408         15.0f, 1.0f,                //Rango, color
409         -14.0f, 0.5f, -8.5f,        //Posición
410         -5.0f, 0.0f, 0.0f,          //Vector de dirección
411         1.0f, 0.0f, 0.01f,          //Atenuación (No poner en ceros)
412         25.0f);                     //angulo de apertura
413     spotLightCount++;
414
415     //Luz roja (Parte trasera del coche)
416     spotLights[4] = SpotLight(1.0f, 0.0f, 0.0f,
417         15.0f, 1.0f,                //Rango, color
418         -5.8f, 0.7f, -8.6f,         //Posición
419         1.0f, 0.0f, 0.0f,           //Vector de dirección
420         1.0f, 0.0f, 0.01f,          //Atenuación (No poner en ceros)
421         25.0f);                     //angulo de apertura
422     spotLightCount++;
423
424     unsigned int spotLightCount2 = 0;
425     spotLights2[0] = spotLights[0]; //Luz linterna
426     spotLightCount2++;
427     spotLights2[1] = spotLights[1]; //Luz Cofre
428     spotLightCount2++;
429     spotLights2[2] = spotLights[2]; //Luz helicoptero
430     spotLightCount2++;
431     spotLights2[3] = spotLights[4]; //Luz roja (parte trasera carro)
432     spotLightCount2++;
433     spotLights2[4] = spotLights[3]; //Luz azul (parte delantera carro)
434     spotLightCount2++;
```

Para que la luz roja (la de la parte trasera del carro) se mueva junto al carro cuando este se traslade, agregamos la configuración necesaria con el arreglo el cual andaremos mostrando dicha luz y se vea y mueva de acuerdo a lo planeado.

```
485     //Mueve Luz roja del carro de acuerdo si se mueve el carro o no
486     glm::vec3 mueveLuz2 = glm::vec3(-5.8f, 0.7f, -8.6f) + glm::vec3(mainWindow.getTransladaCarro(), 0.0f, 0.0f);
487     spotLights2[3].SetFlash(mueveLuz2, glm::vec3(1.0f, 0.0f, 0.0f));
```

Dado que vamos a estar intercambiando entre arreglos, y tenemos spotlights que se estarán moviendo y es importante sus traslaciones o no queramos que desaparezcan, los SetFlash() se aplicarán en esas luces.

```
476     glm::vec3 lowerLight = camera.getCameraPosition();
477     lowerLight.y -= 0.3f;
478     spotLights[0].SetFlash(lowerLight, camera.getCameraDirection());
479     spotLights2[0].SetFlash(lowerLight, camera.getCameraDirection());
480
481     //Mueve Luz azul del carro de acuerdo si se mueve el carro o no
482     glm::vec3 mueveLuz1 = glm::vec3(-14.0f, 0.5f, -8.5f) + glm::vec3(mainWindow.getTransladaCarro(), 0.0f, 0.0f);
483     spotLights[3].SetFlash(mueveLuz1, glm::vec3(-1.0f, 0.0f, 0.0f));
484
485     //Mueve Luz roja del carro de acuerdo si se mueve el carro o no
486     glm::vec3 mueveLuz2 = glm::vec3(-5.8f, 0.7f, -8.6f) + glm::vec3(mainWindow.getTransladaCarro(), 0.0f, 0.0f);
487     spotLights2[3].SetFlash(mueveLuz2, glm::vec3(1.0f, 0.0f, 0.0f));
488
489     //Mueve Luz amarilla del helicoptero de acuerdo si se mueve el helicoptero o no
490     glm::vec3 mueveLuz3 = glm::vec3(0.0f, 4.5f, 6.0) + glm::vec3(mainWindow.getmuevex(), 0.0f, 0.0f);      //Tecla Y y U
491     spotLights[2].SetFlash(mueveLuz3, glm::vec3(0.0f, -1.0f, 0.0f));
492     spotLights2[2].SetFlash(mueveLuz3, glm::vec3(0.0f, -1.0f, 0.0f));
493
494     //Mueve Luz verde del cofre de acuerdo a como rote el cofre     teclas V y B
495     glm::mat4 rotacionLuz = glm::rotate(glm::mat4(1.0f), glm::radians(mainWindow.getrotaCofre()), glm::vec3(0.0f, 0.0f, 1.0f));
496     glm::vec3 mueveLuz4 = glm::vec3(-13.0, 0.99, -10) + glm::vec3(mainWindow.getTransladaCarro(), 0.0f, 0.0f);      //Tecla 1 y 2
497     glm::vec3 direccion = glm::normalize(glm::vec3(rotacionLuz * glm::vec4(0.0f, -0.1f, 0.0f, 0.0f)));
498     spotLights[1].SetFlash(mueveLuz4, direccion);
499     spotLights2[1].SetFlash(mueveLuz4, direccion);
500
```

Se aplica el siguiente algoritmo para saber que luz debe de verse, el cual estas funciones se encuentran en los archivos ***Window.h*** y ***Window.cpp.*** Por ultimo la ***BanderaLuzFaro*** va a ser nuestra bandera que nos ayudará para saber que luz mostrar, por lo que solo se tienen dos casos solo es necesario una bandera, y como esto solo sucede cuando avanza o retrocede el carro se agrega en la parte donde se presiona la tecla en cuestión.

```
507     if (mainWindow.getBanderaLuzFaro() == 1.0f) {
508         shaderList[0].SetSpotLights(spotLights, spotLightCount-1);
509     }
510     else{
511         shaderList[0].SetSpotLights(spotLights2, spotLightCount2 - 1);
512     }
```

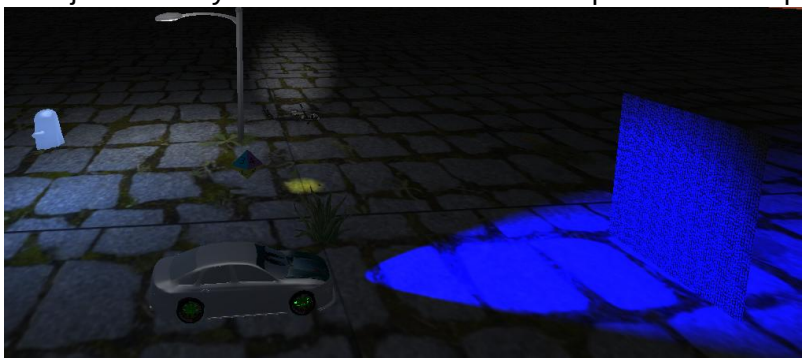
Window.h

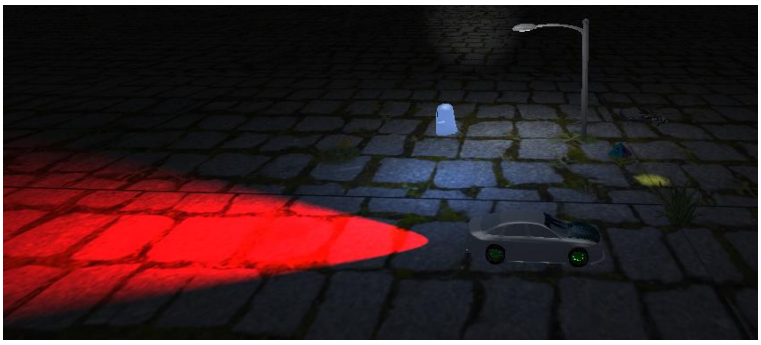
```
25      GLfloat getBanderaLuzFaro() { return BanderaLuzFaro; }

47      GLfloat BanderaLuzFaro;
```

Window.cpp

```
25      BanderaLuzFaro = 1.0f;      //La luz delantera estará prendida por defecto y la trasera apagada
174
175     //Transladar el carro
176     if (key == GLFW_KEY_1)
177     {
178         theWindow->TransladaCarro -= 0.5; //traslada sobre el eje x
179         theWindow->rotaLlantas += 10.0; //Rota las llantas de acuerdo al sentido al que avanza el carro
180
181         theWindow->BanderaLuzFaro = 1.0f;       //Si avanza se prende la bandera del frente
182
183     }
184     if (key == GLFW_KEY_2)
185     {
186         theWindow->TransladaCarro += 0.5; //traslada sobre el eje x
187         theWindow->rotaLlantas -= 10.0; //Rota las llantas de acuerdo al sentido al que avanza el carro
188
189         theWindow->BanderaLuzFaro = 0.0f;       //Si retrocede apaga la bandera del frente
190
191     }
```

Ya por ultimo ejecutamos y vemos el resultado de la implementación que se hizo.





III. 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)

Mediante una IA se creó el modelo de un fantasma ya texturizado, para así ya poder importarlo a OpenGL, sin problema, a este fantasma se le puso una luz puntual de color azul clarito, el cual se alcanza a diferenciar el tono del color entre este y la lámpara a comprar. Y mandamos a renderizar para poder hacer los ajustes que necesite, el cual fue en la escala y se alejó un poco de la lámpara para que se alcancen a distinguir sus luces.

```
68
69      //Modelo del fantasma
70      Model Fantasma_M;
71
```

```
333     Fantasma_M = Model();
334     Fantasma_M.LoadModel("Models/Fantasma.obj");
335
```

```
700
701     //Ejercicio 3
702     //Modelo del fantasma
703     model = glm::mat4(1.0);
704     model = glm::translate(model, glm::vec3(20.0f, 5.0f, 5.0f));
705     model = glm::scale(model, glm::vec3(3.0f, 3.0f, 3.0f));
706     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
707     Fantasma_M.RenderModel();
```

Dado que vamos a apagar y prender los pointLights, tenemos que crear otro arreglo de este mismo, invirtiendo las posiciones de las luces puntuales.

```
87      PointLight pointLights[MAX_POINT_LIGHTS];          //3
88      PointLight pointLights2[MAX_POINT_LIGHTS];         //3
```

```
unsigned int pointLightCount = 0;
//Declaración de primer luz puntual
pointLights[0] = PointLight(0.4f, 0.6f, 1.0f,   //azul claro para el fantasma
    1.0f, 1.0f,
    20.0f, 5.0f, 5.0f,
    1.0f, 0.09f, 0.03f);    //con, lin, exp
pointLightCount++;
//Lampara
pointLights[1] = PointLight(1.0f, 1.0f, 1.0f,   //Blanca
    2.0f, 1.0f,
    12.0f, 10.0f, 20.0,
    1.0f, 0.09f, 0.03f);    //con, lin, exp
pointLightCount++;

//Segundo arreglo para apagar la luz de la lámpara
unsigned int pointLightCount2 = 0;
pointLights2[0] = pointLights[1];       //Luz puntuntual Lámpara
pointLightCount2++;
pointLights2[1] = pointLights[0];       //Luz puntual azul claro
pointLightCount2++;
```

Ahora, como cada luz se debe apagar independientemente del estado en la que se encuentre la otra luz, vamos a tener 4 posibles casos: Lámpara y fantasma prendidos, lámpara prendida y fantasma apagado, lámpara apagada y fantasma prendido, y lámpara y fantasma apagados. Por lo que vamos a crear una variable la cual nos diga en que estado nos encontramos de acuerdo con dos banderas que nos darán el estado de cada modelo, el cuales son ***EstadoLampara*** y ***EstadoFantasma***.

```
//Entra al estado 0 por defecto, es decir que está prendido la luz del fantasma
//Y también la luz de la lámpara
//Tecla L --> Luz de la lámpara
//Tecla K --> Luz del fantasma
int estado = 0;
if (mainWindow.getEstadoLampara() == 1.0f && mainWindow.getEstadoFantasma() == 1.0f)
    estado = 0;
else if (mainWindow.getEstadoLampara() == 1.0f && mainWindow.getEstadoFantasma() == 0.0f)
    estado = 1;
else if (mainWindow.getEstadoLampara() == 0.0f && mainWindow.getEstadoFantasma() == 1.0f)
    estado = 2;
else if (mainWindow.getEstadoLampara() == 0.0f && mainWindow.getEstadoFantasma() == 0.0f)
    estado = 3;
```

Una vez identificando en que estado nos encontramos, mandamos al shader que arreglo muestre y las luces que queremos de ese arreglo. Y dado que es guardamos con un número entero el estado en el que nos encontramos es más fácil de implementar con un Switch-Case y no por medio de if's concatenados

```
switch (estado) {
    case 0: //Ambos PointLight prendidos
        shaderList[0].SetPointLights(pointLights, pointLightCount);
        break;
    case 1: //Solo la lámpara está prendida
        shaderList[0].SetPointLights(pointLights2, pointLightCount - 1);
        break;
    case 2: //Solo el fantasma está prendido
        shaderList[0].SetPointLights(pointLights, pointLightCount2 - 1);
        break;
    case 3: //Todos los PointLight están apagados
        shaderList[0].SetPointLights(pointLights2, pointLightCount2 - 2);
        break;
    default:
        break;
}
```

Para obtener los valores de los estados del modelo del fantasma y de la lámpara se implementó eso en el ***Window***.h y ***Window.cpp***, y parte de esto ya se había hecho al agregar el botón de prendido y apagado de la lámpara, así que solo faltaría el del fantasma.

Window.h

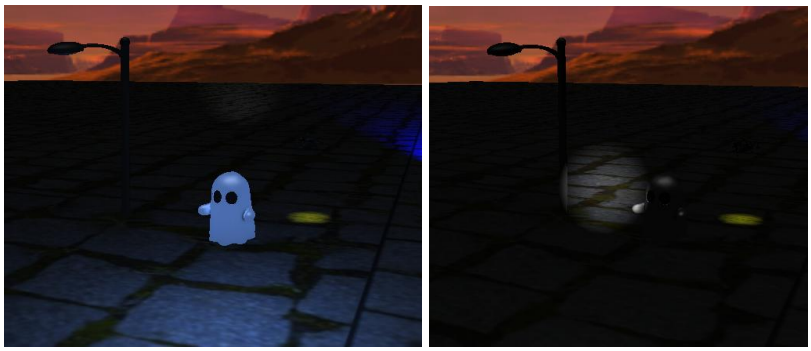
```
23      GLfloat getEstadoLampara() { return EstadoLampara;  }
24      GLfloat getEstadoFantasma() { return EstadoFantasma; }
```

```
46      GLfloat EstadoLampara, EstadoFantasma;
```

Window.cpp

```
23      EstadoLampara = 1.0f;       //Variable para prender y apagar la luz (Estado inicial prendido)
24      EstadoFantasma = 1.0f;      //Variable para prender y apagar la luz (Estado inicial prendido)
193     //Apagar y prender la luz de la lampara
194     if (key == GLFW_KEY_L && action == GLFW_PRESS)
195     {
196         if (theWindow->EstadoLampara == 0.0)    //Prende si el estado anterior estaba apagado
197         {
198             theWindow->EstadoLampara = 1.0;
199         }
200         else if (theWindow->EstadoLampara == 1.0)//Apaga si el estado anterior estaba prendido
201         {
202             theWindow->EstadoLampara = 0.0;
203         }
204
205     }
206     //Apagar y prender la luz del fantasma
207     if (key == GLFW_KEY_K && action == GLFW_PRESS)
208     {
209         if (theWindow->EstadoFantasma == 0.0)       //Prende si el estado anterior estaba apagado
210         {
211             theWindow->EstadoFantasma = 1.0;
212         }
213         else if (theWindow->EstadoFantasma == 1.0)  //Apaga si el estado anterior estaba prendido
214         {
215             theWindow->EstadoFantasma = 0.0;
216         }
217
218     }
```

Una vez implementado esto, ya solo falta ejecutar y probar el 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

Considero yo que no tuve un problema al punto de considerar que sabia nada de como hacerlo, solo por decir algo, me tardé en pensar en cómo implementar el tercer ejercicio ya que manejábamos 4 estados a diferencia de 2, y dado que no se del todo que cosas puedo usar de programación en C# o no, me tardé un poco más de los esperado.

3.- Conclusión:

a. **Los ejercicios del reporte: Complejidad, Explicación.**
   A mi parecer los ejercicios estuvieron a nivel de lo esperado, ya que los primero dos ejercicios fue implementar lo ya visto y el tercero también solo que un poco más complejo por así decirlo.

b. **Comentarios generales: Faltó explicar a detalle, ir más lento en alguna explicación, otros comentarios y sugerencias para mejorar desarrollo de la práctica**
   A mi parecer todo fue explicado bien y como ya he mencionado anteriormente, fue implementar lo ya visto con anteriores prácticas, por lo que puedo decir que la explicación dada fue la necesaria.

c. **Conclusión**
   Con esta práctica considero que he podido mejorar mi forma de manejar las luces, de como prender y apagar luces, hacer intercambios de arreglos, y como manejar cada luz de forma independiente y aplicarle las transformaciones de rotación y translación dado alguna entrada por teclado.

1. **Bibliografía en formato APA**

- Tripo3D. (s.f.). *Modelo 3D*. https://www.tripo3d.ai/app/model/4eb5fbee-dae9-49d6-8bd6-a26a74765f77