



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: CARBAJAL REYES IRVIN JAIR

Nº de Cuenta: 422042084

GRUPO DE LABORATORIO: 11

GRUPO DE TEORÍA: 04

SEMESTRE 2025-2

FECHA DE ENTREGA LÍMITE: 09 DE ABRIL DE 2025

CALIFICACIÓN: _____

Actividades

1. Agregar un spotlight (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, creamos el arreglo donde tendremos nuestra luz spotlight roja

```
86     SpotLight spotLights3[MAX_SPOT_LIGHTS];
```

Luego, agregamos su contador

```
299     unsigned int spotLightCount3 = 0;
```

Ahora, definimos el color, posición y dirección de nuestra luz.

```
310 //luz cofre
311 spotLights3[0] = SpotLight(1.0f, 0.0f, 0.0f,
312                             4.0f, 5.0f,
313                             5.0f, 10.0f, 0.0f,
314                             -5.0f, 0.0f, 0.0f,
315                             1.0f, 0.0f, 0.0f,
316                             30.0f);
317     spotLightCount3++;
```

En el archivo Window.cpp definimos las teclas con la que se moverá el cofre y los valores que tendrá para cuando este se cierre, y así apagar la luz del cofre.

```
133     if (key == GLFW_KEY_F)
134     {
135         theWindow->articulacion1 += 10.0;
136         if (theWindow->articulacion1 > 90.0f)
137         {
138             theWindow->articulacion1 = 90.0f;
139         }
140     }
141
142     if (key == GLFW_KEY_G)
143     {
144         theWindow->articulacion1 -= 10.0;
145         if (theWindow->articulacion1 < 0.0f)
146         {
147             theWindow->articulacion1 = 0.0f;
148         }
149     }
```

Con las teclas F y G se abrirá y cerrará el cofre respectivamente.

Al instanciar el coche colocamos la posición de las luces para que se muevan junto con él.

```
421     //Instancia del coche
422     model = glm::mat4(1.0);
423     model = glm::translate(model, glm::vec3(15.0f + mainWindow.getmuevex(), -12.0f, -1.5f));
424     spotLights[0].SetPos(glm::vec3(0.0f + mainWindow.getmuevex(), 10.0f, 0.0f));
425     spotLights3[0].SetPos(glm::vec3(5.0f + mainWindow.getmuevex(), 10.0f, 0.0f));
426     spotLights2[0].SetPos(glm::vec3(37.0f + mainWindow.getmuevex(), 5.0f, 0.0f));
```

Obtenemos el valor de la articulación 1, si este es mayor a cero, significa que el cofre está abierto y la luz se enciende, si es igual a cero, el cofre está cerrado y se apaga.

```
if (mainWindow.getarticulacion1() > 0.0) {
    shaderList[0].SetSpotLights(spotLights3, spotLightCount3);
}
else {
    shaderList[0].SetSpotLights(spotLights3, spotLightCount3 - 1);
}
```

Ejecución

F -> abre cofre

G -> cierra cofre





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.

Primero, creamos un arreglo para la luz delantera y otro para la luz trasera

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

Luego, los contadores de cada arreglo

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

Declaramos ambas luces

Luz frontal:

```
301 //luz frontal
302 spotLights[0] = SpotLight(1.0f, 1.0f, 0.0f,
303     4.0f, 5.0f,
304     -15.0f, 10.0f, 0.0f,
305     -5.0f, 0.0f, 0.0f,
306     1.0f, 0.0f, 0.0f,
307     30.0f);
308 spotLightCount++;
309
```

Luz trasera:

```
319 //luz trasera
320 spotLights2[0] = SpotLight(1.0f, 1.0f, 0.0f,
321     4.0f, 5.0f,
322     40.0f, 10.0f, 0.0f,
323     5.0f, 0.0f, 0.0f,
324     1.0f, 0.0f, 0.0f,
325     30.0f);
326 spotLightCount2++;
```

Colocamos dentro de Window.cpp lo que ocurre al presionar las teclas designadas

```
191 if (key == GLFW_KEY_X)
192 {
193     theWindow->muevex += 1.0;
194 }
195 if (key == GLFW_KEY_Z)
196 {
197     theWindow->muevex -= 1.0;
198 }
```

Primero al presionar X o Y se desplazará en el eje X hacia atrás o hacia adelante respectivamente, sumando o restando de uno en uno la translación en dicho eje.

```

175     if (key == GLFW_KEY_X && action == GLFW_PRESS)
176     {
177         theWindow->articulacion9 = 2.0;
178     }
179     else if (key == GLFW_KEY_X && action == GLFW_RELEASE)
180     {
181         theWindow->articulacion9 = 0.0;
182     }
183     if (key == GLFW_KEY_Z && action == GLFW_PRESS)
184     {
185         theWindow->articulacion9 = 1.0;
186     }
187     else if (key == GLFW_KEY_Z && action == GLFW_RELEASE)
188     {
189         theWindow->articulacion9 = 0.0;
190     }

```

Luego, al presionar X se le asignará el valor de 2.0 a articulacion9, y al soltarla el valor de 0.0, de manera similar, al presionar Z se le asignará el valor de 1.0 a la misma variable y 0.0 al soltarla. Esto nos permite obtener valores determinados al mover el carro hacia adelante y hacia atrás.

```

387     if (mainWindow.getarticulacion9() == 0.0f) {
388         if (mainWindow.getarticulacion1() > 0.0) {
389             shaderList[0].SetSpotLights(spotLights3, spotLightCount3);
390         }
391         else {
392             shaderList[0].SetSpotLights(spotLights3, spotLightCount3 - 1);
393         }
394     }
395     else {
396         if(mainWindow.getarticulacion9() == 1.0)
397             shaderList[0].SetSpotLights(spotLights, spotLightCount);
398         else {
399             if (mainWindow.getarticulacion9() == 2.0f) {
400                 shaderList[0].SetSpotLights(spotLights2, spotLightCount2);
401             }
402         }
403     }

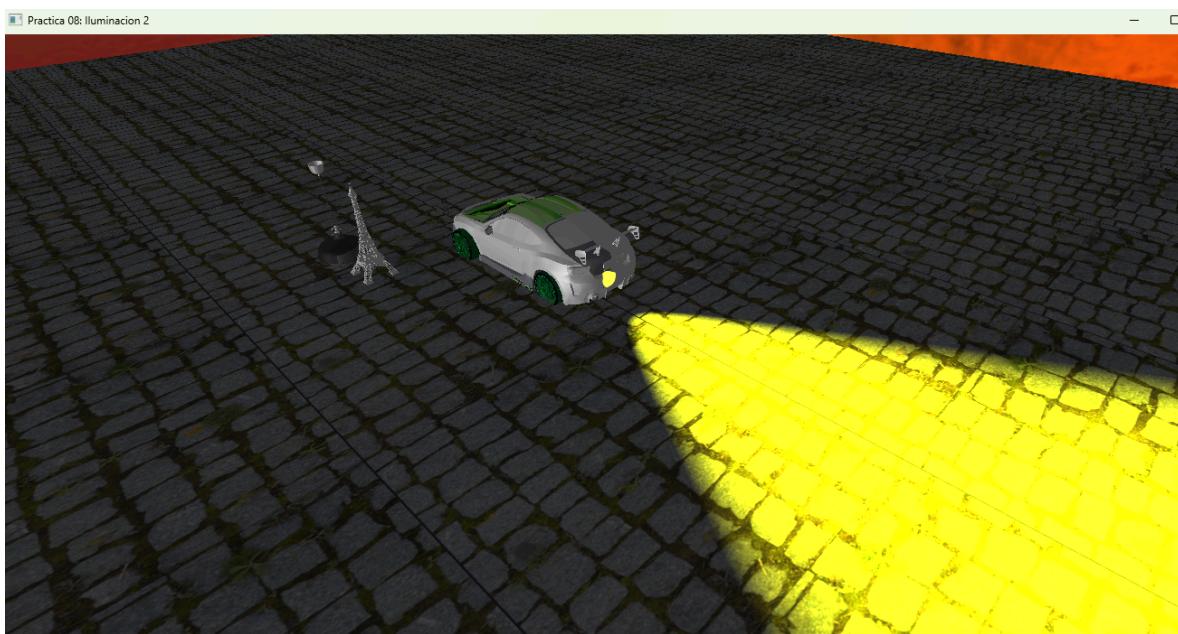
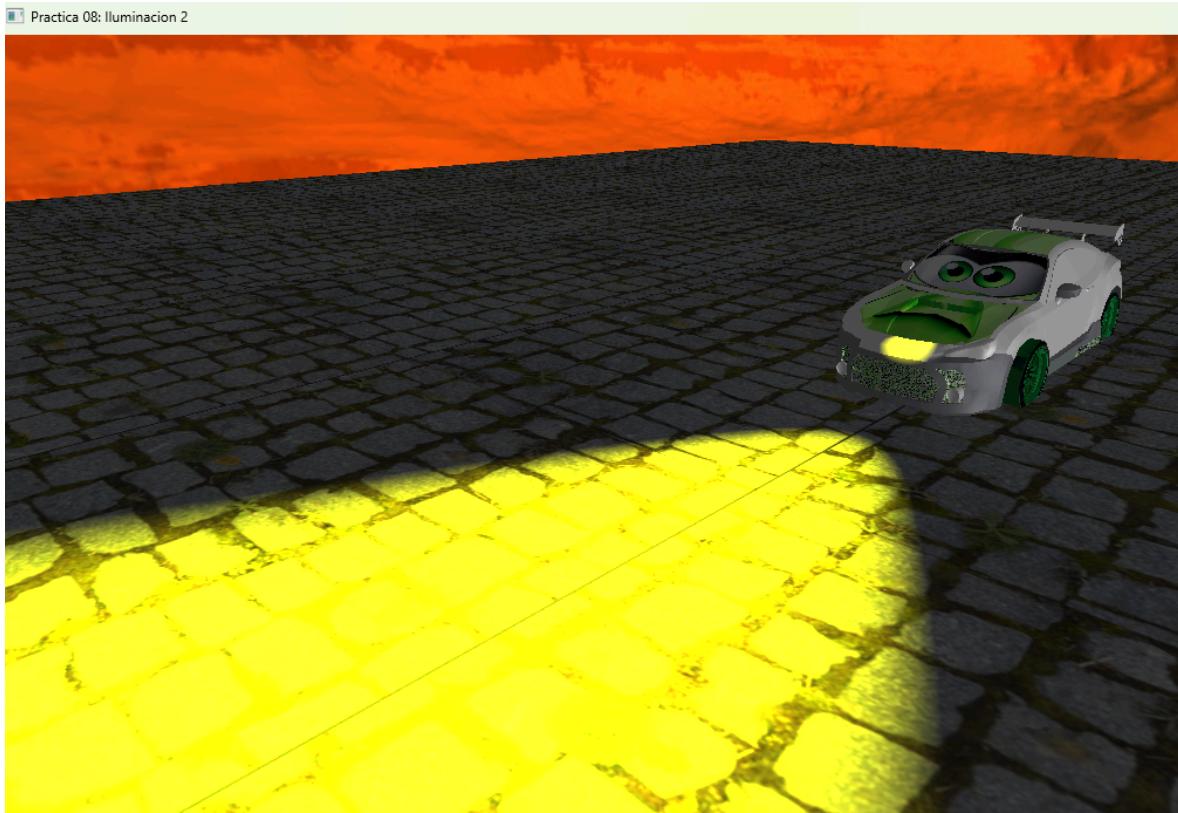
```

De este modo, si el valor de articulacion9 es cero, se podrá abrir el cofre y mostrar la luz roja. Si el valor es 1.0 mostrará la luz delantera y si el valor es 2.0 se mostrará la luz trasera.

Ejecución

X -> carro hacia atrás

Z -> carro hacia adelante



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)

Primero, se crearon dos arreglos para luces puntuales

```
82     PointLight pointLights[MAX_POINT_LIGHTS];  
83     PointLight pointLights2[MAX_POINT_LIGHTS];
```

Agregamos sus contadores

```
276     //contador de luces puntuales  
277     unsigned int pointLightCount = 0;  
278     unsigned int pointLightCount2 = 0;
```

Creamos ambas luces en el primer arreglo

```
280     //Luz lámpara  
281     pointLights[0] = PointLight(1.0f, 1.0f, 1.0f,  
282                             3.0f, 5.0f,  
283                             -8.0f, 19.0f, 30.0f,  
284                             0.7f, 0.2f, 0.1f);  
285     pointLightCount++;  
286  
287     //Luz torre  
288     pointLights[1] = PointLight(1.0f, 1.0f, 0.0f,  
289                             3.0f, 5.0f,  
290                             8.0f, 9.0f, 30.0f,  
291                             0.7f, 0.2f, 0.1f);  
292     pointLightCount++;
```

En el segundo arreglo solo colocamos la luz de la torre

```
294     pointLights2[0] = pointLights[1];  
295     pointLightCount2++;
```

De este modo, le asignamos teclas para los valores de encendido y apagado dentro de Window.cpp

Lámpara:

```

151   if (key == GLFW_KEY_O && action == GLFW_PRESS)
152   {
153       // Cambiar el estado de la lámpara
154       if (theWindow->articulacion3 == 0.0)
155       {
156           theWindow->articulacion3 = 1.0; // Encender
157       }
158       else
159       {
160           theWindow->articulacion3 = 0.0; // Apagar
161       }
162   }

```

Torre:

```

163   if (key == GLFW_KEY_P && action == GLFW_PRESS)
164   {
165       // Cambiar el estado de la lámpara
166       if (theWindow->articulacion4 == 0.0)
167       {
168           theWindow->articulacion4 = 1.0; // Encender
169       }
170       else
171       {
172           theWindow->articulacion4 = 0.0; // Apagar
173       }
174   }

```

De este modo, podemos plantear el siguiente condicional

```

370   if (mainWindow.getarticulacion3() == 1.0 && mainWindow.getarticulacion4() == 1.0) {
371       shaderList[0].SetPointLights(pointLights, pointLightCount);
372   }
373   else {
374       if (mainWindow.getarticulacion3() == 1.0 && mainWindow.getarticulacion4() == 0.0) {
375           shaderList[0].SetPointLights(pointLights, pointLightCount-1);
376       }
377       else {
378           if (mainWindow.getarticulacion3() == 0.0 && mainWindow.getarticulacion4() == 1.0) {
379               shaderList[0].SetPointLights(pointLights2, pointLightCount2);
380           }
381           else {
382               shaderList[0].SetPointLights(pointLights, pointLightCount-2);
383           }
384       }
385   }

```

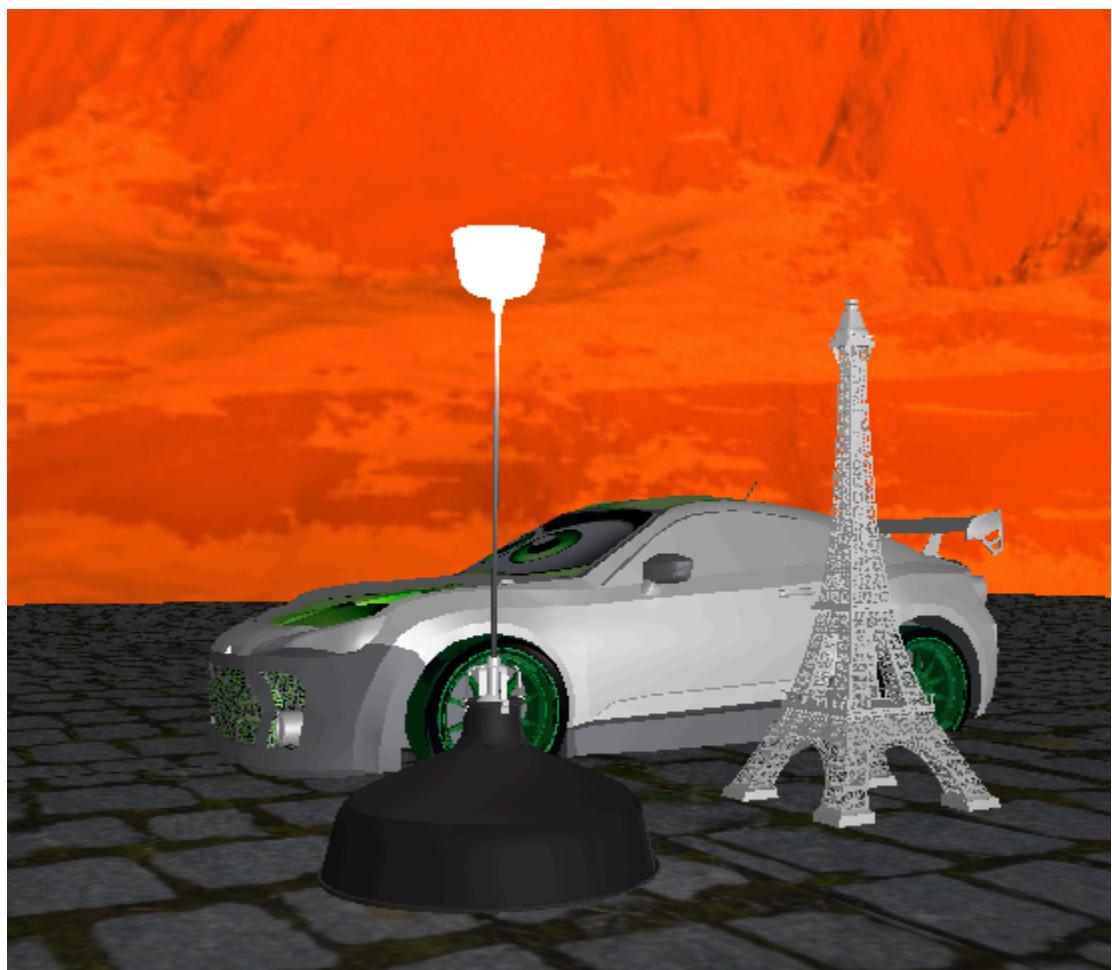
Si ambos están prendidos, colocamos solo el arreglo 1. Si solo la lámpara está encendida, colocamos el arreglo 1 sin la última luz (torre). Si solo la torre está encendida, colocamos el arreglo 2. Y si ambos están apagados, colocamos el arreglo 1 sin ambas luces.

Ejecución

O -> enciende/apaga lámpara

P -> enciende/apaga torre









Video de Ejecución

<https://drive.google.com/file/d/1mb7G4J707OmHoQfKRfDddLe-rlwLqHwh/view?usp=sharing>

Problemáticas

Durante el desarrollo de esta práctica se presentó la problemática en el acomodo de las luces en los arreglos, ya que para el primer ejercicio no hubo problemas, sino hasta colocar las luces delanteras y traseras y agregar las banderas para que se dejaran de mostrar, ya que como algunas luces se encontraban en el mismo arreglo, se apagaban todas o en algunas ocasiones ya no aparecían en los momentos en que se debían mostrar. La solución fue designar arreglos independientes, de este modo, no se mostraban al mismo tiempo y solo fue necesario definir correctamente las condicionales para que aparezcan en el momento deseado.

Otra problemática presentada fue el designar una bandera para que las luces frontales y traseras de la actividad dos se apagaran, ya que al principio se apagaban al recorrer 30.0f tanto en una dirección como en la contraria. Sin embargo, se encontró la manera de cambiar el valor de nuestra bandera al dejar de presionar la tecla que designe la dirección del carro, colocando un cero cuando el carro dejara de moverse, por lo que se usó ese parámetro para determinar el momento en el que nuestras luces se deben apagar.

Conclusiones

La dificultad de esta práctica es un poco más alta que las anteriores, ya que se necesita tener un entendimiento completo de cómo es que las luces se almacenan en un arreglo y tener presente detalles específicos como el saber que no se pueden mostrar dos arreglos de luces al mismo tiempo pero si se puede cambiar entre arreglo de luces en tiempo de ejecución, lo que nos permitió realizar los ejercicios propuestos de forma satisfactoria, y resolver las problemáticas presentadas. Se entendió como iluminar superficies, objetos, y sobre todo cómo interactuar con las luces por medio de teclado.

Bibliografía

- Khronos Group. (s.f.). How lighting works.
https://www.khronos.org/opengl/wiki/How_Lighting_Works