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: Reyes Romero Luis Fernando

Nº de Cuenta: 318155320

GRUPO DE LABORATORIO: 11

GRUPO DE TEORÍA: 06

SEMESTRE 2024-2

FECHA DE ENTREGA LÍMITE: 10 de abril de 2024

CALIFICACIÓN: _____

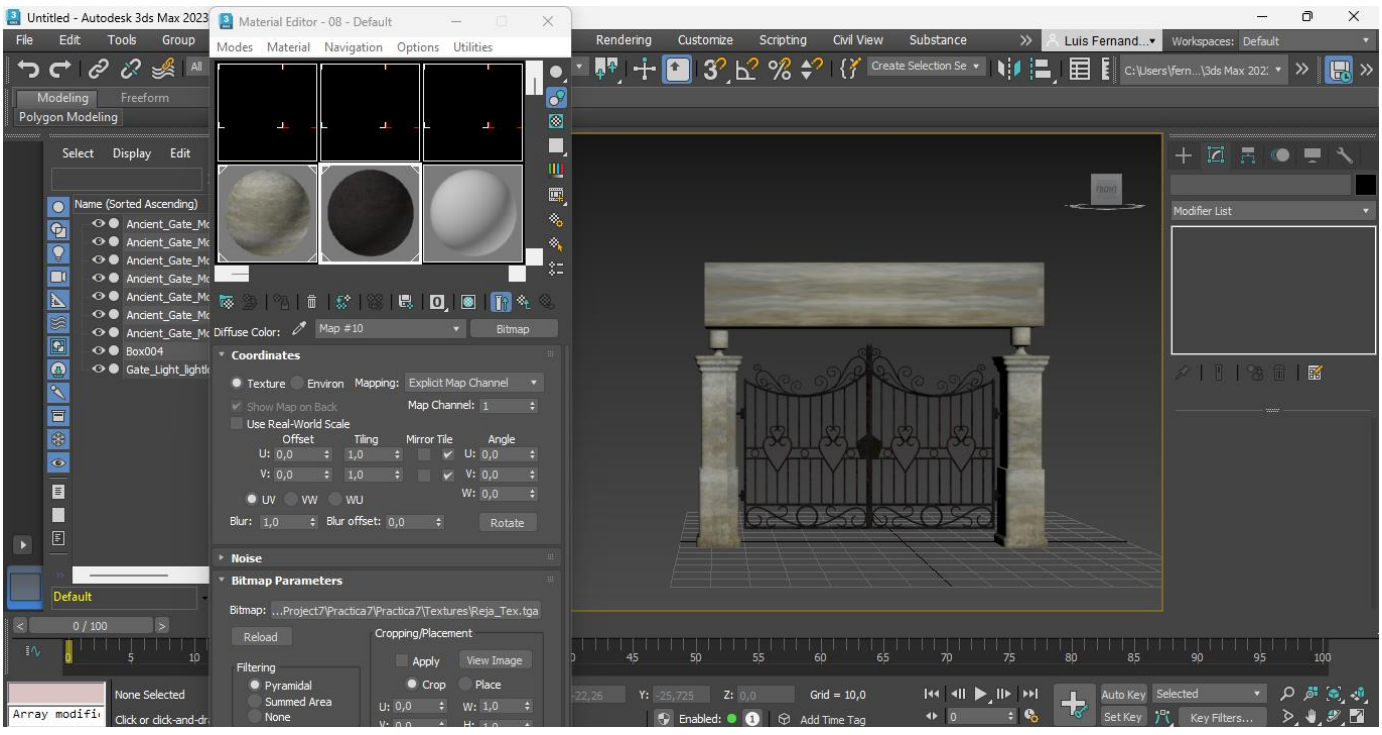
REPORTE DE PRÁCTICA:

Ejercicios

- Agregar un spolight (que no sea luz de color blanco) que ilumine a su puerta creada para el previo.
- 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.
- Agregar otra luz de tipo puntual ligada a un modelo elegido por ustedes y que puedan prender y apagar 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).

Para el primer ejercicio solo debemos de colocar una luz de tipo spotlight a la puerta que hicimos en el previo de esta práctica. Entonces lo que hacemos es crear la luz y el color de esta luz decidí que fuera de color dorado. Dicho esto, nos vamos al arreglo de luces SpotLight y creamos nuestra luz. Cabe mencionar, que nuestra puerta la debemos de texturizar.

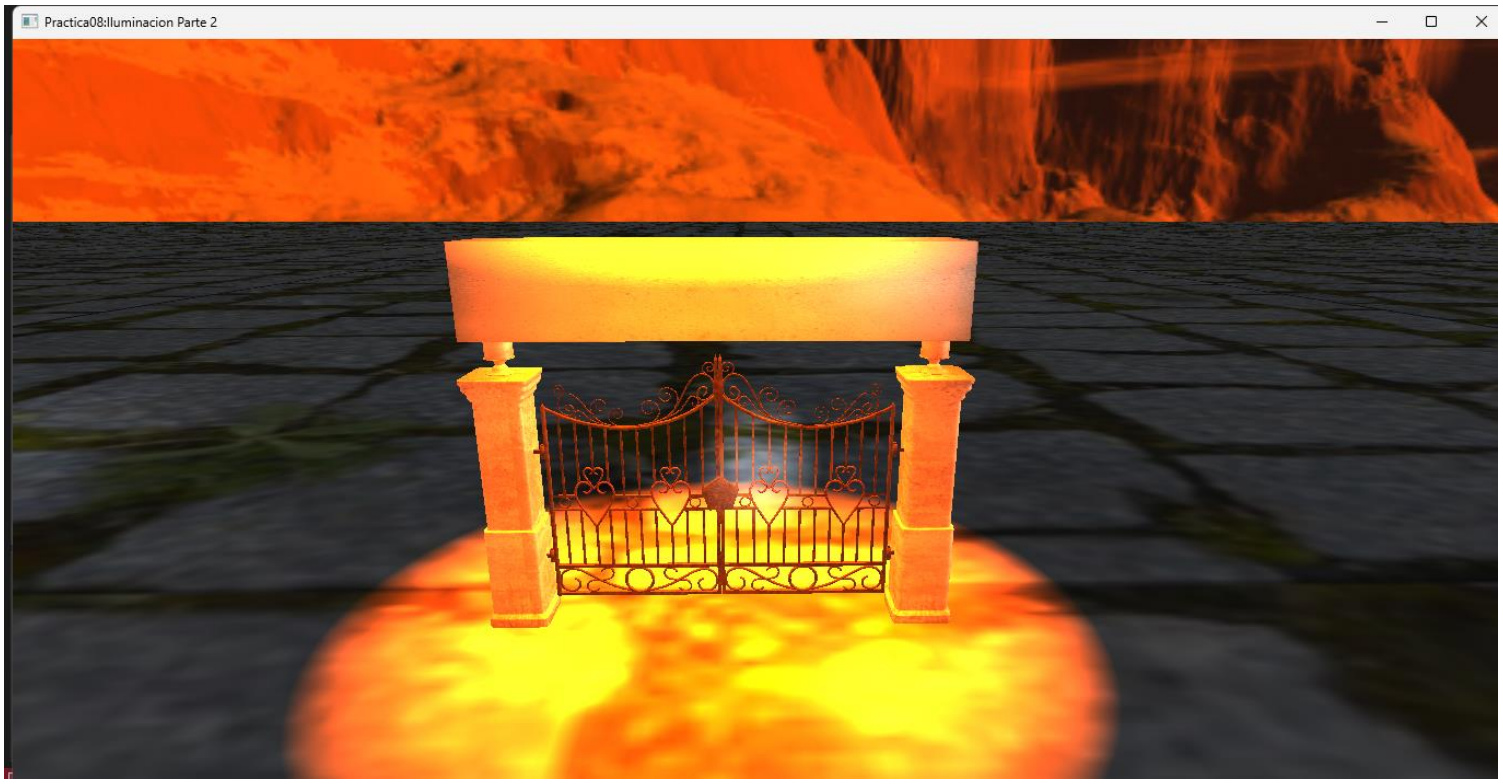
Textura de la puerta



Luz en el código

```
//Luz de la puerta
spotLights[2] = SpotLight(1.0f, 0.2706f, 0.0f,
    1.0f,  2.0f,
   -0.5f,  11.0f, 29.3f,
    0.0f, -11.0f, 0.0f,
    0.3f,  0.0f,  0.0f,
    20.0f);
spotLightCount++;
```

Ejecución del código



Ejercicio de las luces del coche

Para este ejercicio lo primero que vamos a hacer es crear dos arreglos de luces de tipo spotlight donde uno de ellos será para las luces delanteras del coche y el otro arreglo son para las luces traseras.

```
//Arreglo de luces delanteras del coche
SpotLight spotLights2[MAX_SPOT_LIGHTS];

//Arreglo de luces traseras del coche
SpotLight spotLights3[MAX_SPOT_LIGHTS];
```

Después vamos a definir los parámetros de las luces. Cabe mencionar, que la posición y la dirección no la colocamos porque necesitamos que las luces estén ligadas al movimiento que tiene el coche. Entonces, la posición y la dirección de las luces se van a definir después.

Luces delanteras

```
unsigned int spotLightCount2 = 0;
//Arreglo de luces para la parte delantera del coche
spotLights2[0] = SpotLight(0.0f, 0.0f, 1.0f,
    7.0f, 2.0f,
    0.0f, 1.0f, 0.0f,
    0.0f, 0.0f, 0.0f,
    1.0f, 0.09f, 0.09f,
    10.0f);
spotLightCount2++;

spotLights2[1] = SpotLight(0.0f, 0.0f, 1.0f,
    7.0f, 2.0f,
    0.0f, 1.0f, 0.0f,
    0.0f, 0.0f, 0.0f,
    1.0f, 0.09f, 0.09f,
    10.0f);
spotLightCount2++;
```

Luces traseras

```
//Arreglo de luces para la parte trasera del coche
unsigned int spotLightCount3 = 0;
spotLights3[0] = SpotLight(1.0f, 0.0f, 0.0f,
    20.0f, 2.0f,
    0.0f, 0.0f, 0.0f,
    0.0f, 0.0f, 0.0f,
    1.0f, 0.09f, 0.09f,
    10.0f);
spotLightCount3++;

spotLights3[1] = SpotLight(1.0f, 0.0f, 0.0f,
    20.0f, 2.0f,
    0.0f, 0.0f, 0.0f,
    0.0f, 0.0f, 0.0f,
    1.0f, 0.09f, 0.09f,
    10.0f);
spotLightCount3++;
```

Después de hacer las modificaciones anteriores ahora lo que vamos a hacer es ligar cada una de estas luces con el movimiento del coche. A continuación,

se muestran las modificaciones que se hicieron para ligar cada una de estas luces.

Luces delanteras del coche

```
float muevex = mainWindow.getmuevex();

//Luces delanteras de color azul del coche
glm::vec3 carPosition = glm::vec3(-3.2f + muevex, -1.08f, 2.7f);
glm::vec3 carDirection = glm::vec3(-1.0f, 0.0f, 0.0f);

glm::vec3 spotLightPosition= carPosition;
glm::vec3 spotLightDirection = carDirection;
spotLights2[0].SetFlash(spotLightPosition, spotLightDirection);

glm::vec3 carPosition3 = glm::vec3(-3.2f + muevex, -1.08f, -0.0f);
glm::vec3 carDirection3 = glm::vec3(-1.0f, 0.0f, 0.0f);

glm::vec3 spotLightPosition4 = carPosition3;
glm::vec3 spotLightDirection4 = carDirection3;
spotLights2[1].SetFlash(spotLightPosition4, spotLightDirection4);
```

Como mencionamos anteriormente aquí es donde se coloca la posición de la luz y la dirección de esta. Además, para no colocar toda la función “mainWindow.getmuevex()” lo declaramos como “muevex” solamente.

Luces traseras del coche

```
//Luces traseras de color rojo del coche
glm::vec3 carPosition2 = glm::vec3(4.0f + muevex, -1.08f, 0.0f);
glm::vec3 carDirection2 = glm::vec3(1.0f, 0.0f, 0.0f);

glm::vec3 spotLightPosition3 = carPosition2;
glm::vec3 spotLightDirection3 = carDirection2;
spotLights3[0].SetFlash(spotLightPosition3, spotLightDirection3);

glm::vec3 carPosition4 = glm::vec3(4.0f + muevex, -1.08f, 2.7f);
glm::vec3 carDirection4 = glm::vec3(1.0f, 0.0f, 0.0f);

glm::vec3 spotLightPosition5 = carPosition4;
glm::vec3 spotLightDirection5 = carDirection4;
spotLights3[1].SetFlash(spotLightPosition5, spotLightDirection5);
```

Estas modificaciones nos aseguran que cada vez que se mueva el coche las luces se muevan con este mismo, pero lo que nos pide el ejercicio es que cuando el coche avanza hacía adelante las luces delanteras se prendan, pero las luces traseras se apaguen y viceversa cuando avance hacia atrás el

coche las luces traseras se prendan y las delanteras se apaguen. Para lograr esto vamos a modificar los archivos Window.h y Window.cpp respectivamente. Lo primero que vamos a hacer es ir al archivo Window.h para declarar las funciones necesarias para lograr lo que se nos pide en este ejercicio. Vamos a escribir funciones parecidas a las funciones que se hicieron para el ejercicio de está práctica el cual consistía en prender y apagar la luz de nuestra lámpara. Entonces, vamos a definir 4 funciones donde dos de ellas van a ser de tipo booleano y van a regresar la variable "prender" y "prender2" respectivamente. Las otras dos funciones serán las encargadas de tener el estado de estas variables.

Funciones para prender las luces

```
//Funciones para que prendan las luces delanteras del coche
bool GetPrenderLuz() { return prender; }
void PrenderLuz(bool state) { prender = state; }

//Funciones para que prendan las luces traseras del coche
bool GetPrenderLuz2() { return prender2; }
void PrenderLuz2(bool state) { prender2 = state; }
```

```
bool apagar = false;
bool prender = false;
bool prender2 = false;
```

Después de tener estas funciones, vamos al archivo Window.cpp y tenemos que letra del teclado hace que el coche se mueva hacia adelante para colocar las condiciones necesarias. Para este caso la letra U es la encargada de hacer que el coche se mueva hacia adelante.

Condiciones para la letra U

```
if (key == GLFW_KEY_U)
{
    theWindow-> muevex -= 1.0;
    theWindow->PrenderLuz(true);
    theWindow->PrenderLuz2(false);
}
```

Las condiciones que le pasamos a la letra U es que al ser presionada el coche debe de avanzar hacia adelante y que "PrenderLuz()" sea verdadera y si nos vamos a nuestro archivo Window.h observamos que PrenderLuz()

es la función para que se prenda las luces delanteras. De igual forma, pasamos como parámetro “false” a la función PrenderLuz2() que esta función corresponde a las luces traseras del coche. De esta forma aseguramos que al presionar la tecla U las luces delanteras del coche se van a prender y las traseras se van a apagar. Hacemos lo mismo para letra Y solo que cambiamos los parámetros para que las luces traseras se prendan y las delanteras se apaguen.

Condiciones para la letra Y

```
if (key == GLFW_KEY_Y)
{
    theWindow-> muevex += 1.0;
    theWindow->PrenderLuz2(true);
    theWindow->PrenderLuz(false);
}
```

Después de hacer estas modificaciones vamos nuevamente a nuestro archivo main y nos vamos a la parte donde se le pasa la información de las luces al shader. Una vez estando aquí, nuestra primera condición será que si PrenderLuz() es verdadera al shader se le van a pasar la información de las luces del arreglo que contiene las luces de tipo spotlight que se refieren a las luces delanteras del coche. En caso contrario si PrenderLuz2() es verdadero se le van a pasar al shader la información de las luces del arreglo que contiene las luces traseras. Por esta razón elegimos hacer un arreglo para cada tipo de luces para facilitar el manejo de datos. Finalmente ejecutamos y observamos la salida.

Ejecución del código

Aquí hay algo que tenemos que resaltar y es que si presionamos la tecla U o Y para que el coche avance o retroceda las luces de tipo spotlight principal desaparecen, ya que no se le esta pasando la información al shader y algo que también me di cuenta es que al shader no se le puede pasar información de dos arreglos del mismo tipo de luces. Entonces, para resolver este problema es que vamos a crear una nueva función en Window.h y en Window.cpp para que al presionar la tecla I las luces de tipo spotlight del primer arreglo vuelvan a aparecer.

Funciones para que aparezcan las primeras luces

```
//Funciones para que vuelvan las spotLights normales
bool GetLucesSpothLights() { return luces; }
void LucesSpothLights(bool state) { luces = state; }
```

```
bool apagar = false;
bool prender = false;
bool prender2 = false;
bool luces = false;
```

Condiciones para la letra I

```
if (key == GLFW_KEY_I)
{
    theWindow->LucesSpothLights(true);
    theWindow->PrenderLuz(false);
    theWindow->PrenderLuz2(false);
}
```

Nos vamos al archivo main y vamos a colocar una nueva condición para que al presionar la letra I, las luces del coche se apaguen y vuelvan las luces que teníamos en un inicio.

```
//Condicion para que vuelvan las primeras luces SpotLight
else if (mainWindow.GetLucesSpothLights()) {
    shaderList[0].SetSpotLights(spotLights, spotLightCount);
}
```

Finalmente, ejecutamos el código y observamos que efectivamente al presionar la tecla I las luces que estaban en un principio vuelven a aparecer.

Ejecución del código

Al presionar Y

Al presionar I

Ejercicio tres

Para el último ejercicio de esta práctica vamos a cargar un nuevo modelo colocarle una luz de tipo puntual y que dicha luz se pague y prenda al mismo tiempo que la luz de nuestra lampara. El modelo que yo escogí fue una banca, entonces lo primero que vamos a hacer para este ejercicio es colocarle una la luz puntual a la banca.

Luz de la banca

```
//luz de la banca
pointLights[2] = PointLight(1.0f, 1.0f, 1.0f,
    0.3f, 0.3f,
    5.5f, 0.0f, 11.5f,
    0.3f, 0.2f, 0.1f);
pointLightCount++;
```

De igual forma, vamos a crear un nuevo modelo para la banca, una nueva dirección y renderizar el modelo.

```
Model Kitt_M;
Model Llanta_M;
Model Blackhawk_M;
Model Coche;
Model Cofre;
Model Llanta1;
Model Llanta2;
Model Llanta3;
Model Llanta4;
Model Lampara;
Model Puerta;
Model Banca;
```

```
Banca = Model();
Banca.LoadModel("Models/Banca.obj");
```

```
//Banca
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(5.0f, -1.0f, 11.5f));
model = glm::scale(model, glm::vec3(0.02f,0.02f,0.02f));
model = glm::rotate(model, 90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Banca.RenderModel();
```

Para que la luz que tiene la banca se apague y se prenda al mismo tiempo que la lámpara, no es necesario crear una nueva función, ya que ambas son luces de tipo puntual y están en el mismo arreglo de luces solo basta con

decrementar el contador en nuestra condición para que se prendan y apaguen las luces que se hizo para el ejercicio de clase para esta práctica.

Condición para que no se tomen en cuenta ambas luces

```
//Condición para apagar la luz de nuestra lámpara y de la banca
if (mainWindow.GetApagarLuz()) {
    shaderList[0].SetPointLights(pointLights, pointLightCount-2);
}
else {
    shaderList[0].SetPointLights(pointLights, pointLightCount);
}
```

Esto nos quiere decir que si “GetApagarLuz()” es verdadera no se va a tomar en cuenta las últimas dos luces del arreglo de luces puntuales que son la luz de la lámpara y la luz de la banca. En caso contrario si “GetApagarLuz()” es falsa ambas luces se van a prender. Una vez hecho esto vamos a compilar nuestro código y observar nuestra salida.

Ejecución del código

Al presionar Z

Al presionar X

De igual manera, en los entregables se va a adjuntar un vídeo para que se visualice de mejor manera cada uno de estos ejercicios.

Lista de problemas

- No hubo ningún problema para esta práctica.

Conclusión

Estás prácticas de iluminación se me han hecho muy divertidas más que nada por la interacción de las luces como, por ejemplo, la luz de la lámpara que se apaga y se prende hace una ilusión de una verdadera lámpara, de igual forma, las luces del coche. Además, los ejercicios que se han dejado en estas prácticas han sido al nivel de clase, ya que el profesor nos explica el código de manera concreta y la solución de los ejercicios nos da una idea de cómo resolverlos. Para está práctica lo único que me confundió es la redacción de los ejercicios porque no supe si era necesario hacer dos luces delanteras y dos traseras para el coche o solo bastaba con una luz delantera y una trasera, por esa razón decidí colocar dos y dos para simular las luces de un coche.

Bibliografía en formato APA

Para esta práctica no se hizo uso de ninguna fuente externa debido a que la información proporcionada por el profesor en el laboratorio fue suficiente para hacer esta práctica.