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 07

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: 03 de marzo de 2024

CALIFICACIÓN: _____

REPORTE DE PRÁCTICA:

Ejercicios

- Agregar movimiento con teclado al helicóptero hacia adelante y atrás.
- Crear luz spotlight de helicóptero de color amarilla que apunte hacia el piso y se mueva con el helicóptero
- Importar 1 modelo de lámpara y crearle luz puntual blanca.

Para poder darle movimiento al helicóptero tenemos que crear una nueva variable “muevex” en el archivo Window.h y en el archivo Window.cpp vamos a escribir lo que debe de hacer esta variable. Dicho esto, lo primero que vamos a hacer es crear nuestra nueva variable que tendrá el nombre de “muevex2” y la declaramos en el archivo Window.h.

Modificaciones en Window.h

```
GLfloat getmuevex() { return muevex; }
GLfloat getmuevex2() { return muevex2; }
```

```
GLfloat muevex;
GLfloat muevex2;
```

Después, en el archivo Window.cpp vamos a declarar que letras del teclado estará ligada esta variable.

Modificaciones en Window.cpp

```
Window::Window(GLint windowWidth, GLint windowHeight)
{
    width = windowWidth;
    height = windowHeight;
    muevex = 2.0f;
    muevex2 = 2.0f;
    for (size_t i = 0; i < 1024; i++)
    {
        keys[i] = 0;
    }
```

```
if (key == GLFW_KEY_H)
{
    theWindow->muevex2 += 1.0;
}
if (key == GLFW_KEY_J)
{
    theWindow->muevex2 -= 1.0;
}
```

Esto quiere decir que con las letras H y J se mueve el helicóptero adelante y hacia atrás. Finalmente, nos vamos en nuestro código main y vamos donde estamos renderizando el helicóptero y colocamos el movimiento en la función traslate del helicóptero precisamente en el eje x, ya que, en ese eje es donde queremos que se mueva el helicóptero.

```
//Helicoptero
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f + mainWindow.getmuevex2(), 5.0f, 6.0));
model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.3f));
model = glm::rotate(model, -90 * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::rotate(model, 90 * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Blackhawk_M.RenderModel();
```

De esta forma se cumple la primera actividad y pasamos a la siguiente. La siguiente actividad es crear una luz de tipo Spotlight de color amarillo, que apunte hacía abajo y que se mueva con el helicóptero. Para hacer esta actividad lo primero que debemos de hacer es crear nuestra luz Spotlight de color amarillo y para crear este color tenemos que combinar el rojo y verde. Dicho esto, vamos a nuestro código main y creamos nuestra luz amarilla.

Luz amarilla ligada al helicóptero

```
//Luz del helicoptero
spotLights[3] = SpotLight(1.0f, 1.0f, 0.0f,
    1.0f, 2.0f,
    0.0f, 0.0f, 0.0f,
    0.0f, 0.0f, 0.0f,
    1.0f, 0.0f, 0.0f,
    15.0f);
spotLightCount++;
```

Cabe mencionar, que tanto la posición y la dirección no es necesario colocarla en este bloque, ya que después vamos a definir su dirección y su posición. Después nos vamos a donde se renderiza el helicóptero para colocar la función necesaria para ligar la luz al helicóptero. Aquí es donde definimos la posición y la dirección de la luz.

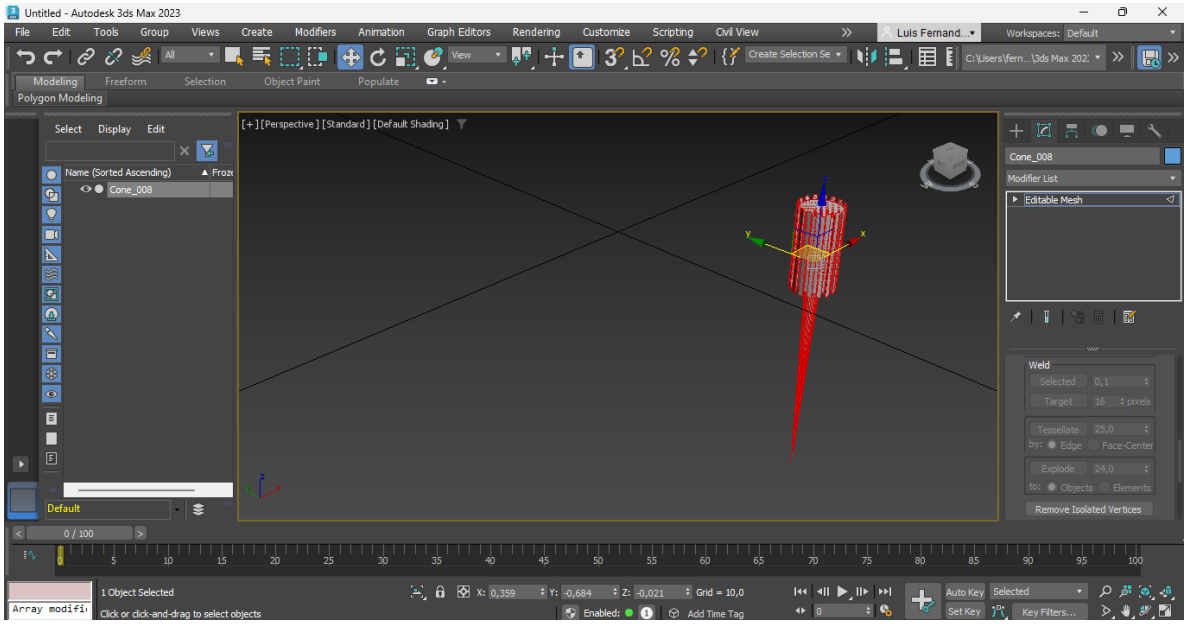
```
//Luz ligada al helicoptero
glm::vec3 helPosition = glm::vec3(0.0f + mainWindow.getmuevex2(), 1.2f, 6.0f);
glm::vec3 helDirection = glm::vec3(0.0f, -1.2f, 0.0f);

glm::vec3 spotLightPosition2 = helPosition;
glm::vec3 spotLightDirection2 = helDirection;
spotLights[3].SetFlash(spotLightPosition2, spotLightDirection2);
```

Podemos observar que la dirección de la luz esta en -y debido a que se requiere que la luz apunte hacia abajo. Algo que me pareció interesante es que tanto en la posición y la dirección en el eje y se debe de poner el mismo valor para poder ver la luz apuntando hacia abajo como pide la actividad, ya que si en la dirección cambiaba este valor en el eje y no se puede ver la luz y si ponía el valor positivo la luz se colocaba arriba del helicóptero. Finalmente, para que la luz se mueva con el helicóptero vamos a colocar la función de “getmuevex2” en la posición del eje x, ya que esta función es la misma que tiene el helicóptero para que tenga movimiento, de esta forma la luz se mueve cada ves que el helicóptero se mueva.

Finalmente, para la última actividad lo que vamos a hacer es cargar un modelo de una lampara y crearle una luz blanca de tipo puntual. Para este ejercicio nos debemos de poner de acuerdo con nuestro equipo para no repetir el tipo de lampara, ya que después podemos usar estos modelos para el proyecto. Como mencionamos en nuestro proyecto de propuesta vamos a usar postes de luz, focos de luz, senderos iluminados. Entonces, la lampara que yo escogí es una lampara para exteriores. Dicho esto, primero lo que hacemos es buscar nuestro modelo de la lampara y cargarlo en el software de 3ds Max para poder ver que el modelo este creado a partir de triángulos para que se pueda dibujar sin problema alguno.

Lampara en 3ds Max



Después de exportar el modelo lo vamos a renderizar y observamos donde está colocada la lampara para que después coloquemos de forma correcta

nuestra luz puntual. Una vez visto donde se encuentra nuestra lampara, vamos a crear nuestra luz puntual. Para esto nos vamos donde se crean las luces de tipo puntual y creamos nuestra luz blanca, y vamos modificando la posición de la luz para que quede en el centro de nuestra lampara debido a que nuestra lampara es de exteriores.

Se crea un nuevo modelo para la lampara

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

Ruta del modelo de la lampara

```
Lampara = Model();
Lampara.LoadModel("Models/lampara1.obj");
```

Renderizado de la lampara

```
//Lampara
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, 0.0f, 5.0f));
model == glm::scale(model, glm::vec3(0.1f, 0.1f, 0.1f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Lampara.RenderModel();
```

Luz blanca de la lampara

```
//luz de la lampara
pointLights[1] = PointLight(1.0f, 1.0f, 1.0f,
    0.0f, 1.0f,
   -1.5f, 0.0f, 12.5f,
    0.3f, 0.2f, 0.1f);
pointLightCount++;
```

Ejecución del programa

Una vez hecho todos los cambios mencionados anteriormente, compilamos en código y observamos nuestras salidas.

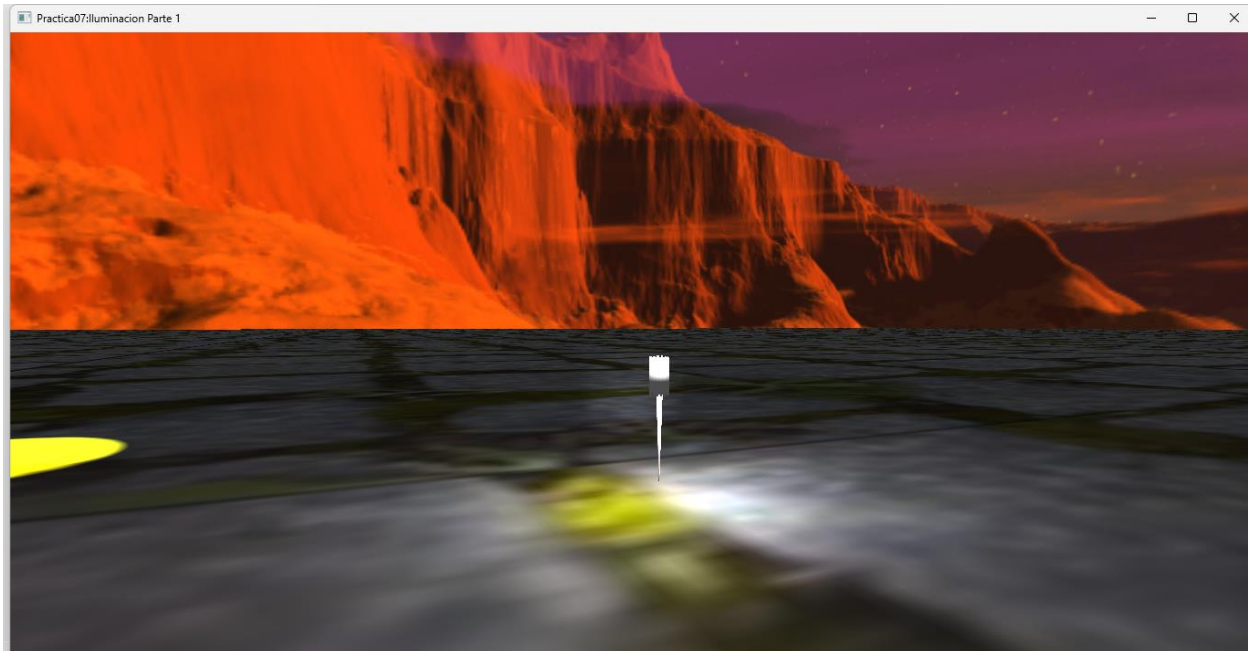
Helicóptero con luz amarilla

Helicóptero con movimiento hacia adelante

Podemos ver que la luz avanza cada vez que el helicóptero avanza.

Helicóptero con movimiento hacia atrás

Modelo de la lampara con la luz blanca de tipo puntual



Lista de problemas

- Para esta práctica no hubo ningún problema

Conclusión

Esta práctica me agrado bastante, porque fue muy sencilla de hacer gracias a la explicación que se dio en el laboratorio. Además, logramos observar los tipos de luz que existen y que parámetros tiene cada una de ellas, ya que si observamos bien nuestra luz puntual no nos pide la dirección de la luz como las luces de tipo Spotlight. Siento que la explicación de todos los nuevos archivos (códigos) que se usaron para esta práctica estuvo bien, ya que se explicó cada uno de estos y que es lo que hacia por esa razón a la hora de manipular las luces se me facilito hacerlo. Gracias a todo esto logramos aprender y comprender la implementación de fuentes de iluminación dentro de OpenGL y GLSL.

Bibliografía en formato APA

No se usó ninguna fuente externa para está práctica. Con lo visto y explicado en la clase de laboratorio se logró hacer está práctica.