



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 N° 07

NOMBRE COMPLETO: Casillo Martinez Diego Leonardo

N.º de Cuenta: 319041538

GRUPO DE LABORATORIO: 11

GRUPO DE TEORÍA: 06

SEMESTRE 2024-2

FECHA DE ENTREGA LÍMITE: 02/10/2024

CALIFICACIÓN: _____







REPORTE DE PRÁCTICA:

1.- Desarrollo

Ejercicio 1: este ejercicio Plantea el siguiente problema:

Agregar movimiento con teclado al helicóptero hacia adelante y atrás.

Para este ejercicio se utilizó el modelo.obj del helicóptero y a este se le separaron las hélices superior y lateral para poder importarlas en la carpeta model de la siguiente forma:

	Helice2	02/10/2024 06:37 p. m.	Archivo MTL	1 KB
<input type="checkbox"/> 	Helice2	02/10/2024 06:37 p. m.	3D Object	21 KB
	Helicoptero	02/10/2024 05:03 p. m.	Archivo MTL	3 KB
	Helicoptero	02/10/2024 05:03 p. m.	3D Object	3,998 KB
	Helice1	02/10/2024 05:01 p. m.	Archivo MTL	1 KB
	Helice1	02/10/2024 05:01 p. m.	3D Object	133 KB

Una vez hecho esto se agrego un pequeño modelo jerárquico en donde lo que se heredaba era la traslación del cuerpo principal de la siguiente forma:

```
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, 5.0f, 6+mainWindow.getDesplazamiento()));
modelaux = model;
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
/*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));
Helicoptero.RenderModel();
//Helice1
model = modelaux;
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Helice1.RenderModel();
//Helice2
model = modelaux;
model = glm::translate(model, glm::vec3(0.15, 0.65f, 4.9f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Helice2.RenderModel();
```

A su vez se le agregaron rotaciones y traslación variables asignando la tecla H y J, esto se logro agregando el control con Windows.cpp de la siguiente forma:

```

if (key == GLFW_KEY_H)
{
    if (theWindow->Desplazamiento < -100) {
    }
    else {
        theWindow->articulacion1 -= 20.0;
        theWindow->articulacion2 += 20.0;
        theWindow->Desplazamiento -= 1;
    }
}

if (key == GLFW_KEY_J)
{
    if (theWindow->Desplazamiento > 110) {
    }
    else {
        theWindow->articulacion1 += 20.0;
        theWindow->articulacion2 += 20.0;
        theWindow->Desplazamiento += 1;
    }
}

```

Gracias a ello se puede visualizar como cuando se tocan las teclas H y J se desplazan para atrás o para delante de la siguiente forma:



Ejercicio 2: este ejercicio Plantea el siguiente problema:

crear luz spotlight de helicóptero de color amarilla que apunte hacia el piso y se mueva con el helicóptero

Para este ejercicio se definió una spotlight de la siguiente forma:

```

//luz fija
spotLights[0] = SpotLight(1.0f, 1.0f, 0.0f, // Color
    2.5f, 5.0f, // componentes ambiental y difu
    0.0f, 4.2f, 3.5f, // Posición inicial
    0.0f, -1.0f, 0.0f, // Dirección hacia donde apunt
    1.0f, 0.03f, 0.05f, // atenuaciones
    20.0f); // Ángulo de apertura
spotLightCount++;

```

A este solamente se le puso la dirección a donde apunta la luz en el eje Y negativo y se posiciono en la punta del helicóptero para simular una luz que proviene del helicóptero

Y para lograr hacer que se pueda mover esta luz con el helicóptero se hizo lo siguiente:

```
spotLights[0].SetFlash(glm::vec3(0.0f, 4.2f, 3.5f+ mainWindow.getDesplazamiento()), glm::vec3(0.0f, -1.0f, 0.0f));
```




Y gracias a eso se puede hacer el efecto de que la luz se mueve con el helicóptero y se logra ver el siguiente resultado.



Ejercicio 3: este ejercicio Plantea el siguiente problema:

Añadir en el escenario 1 modelo de lámpara texturizada (diferente a los que usarán en su proyecto final) y crearle luz puntual blanca

Para este ejercicio primero se busco e importo una lampara a la cual se le agrego una textura y se agrego en las carpetas models y texture de la siguiente forma:

 Lampara	02/10/2024 08:04 p. m.	Archivo MTL	1 KB
 Lampara	02/10/2024 07:47 p. m.	3D Object	58 KB
 lamp	02/10/2024 08:00 p. m.	GIMP 2.10.38 TGA	1,025 KB

Una vez realizado esto se importo en nuestro proyecto de la siguiente forma:

```
RenderModel();  
//Lampara  
model = glm::mat4(1.0);  
model = glm::translate(model, glm::vec3(6.0f, -1.0f, 0.0f));  
model = glm::scale(model, glm::vec3(1.5f, 1.5f, 1.5f));  
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));  
Lampara.RenderModel();
```

Y se le agrego una luz puntual de la siguiente forma:

```
unsigned int pointLightCount = 0;  
//Declaración de primer luz puntual  
pointLights[0] = PointLight(1.0f, 1.0f, 1.0f,  
    0.0f, 1.0f,  
    6.0f, 1.5f, 0.0f,  
    0.1f, 0.07f, 0.05f);  
pointLightCount++;
```

En donde se trasladó en la posición de la lampara logrando el siguiente resultado:



2. Problemas que surgieron:

Durante esta práctica, el único problema que surgió es que inicialmente al importar la lampara no funcionaban las texturas, pero esto se debía a que la textura estaba en formato .png al ponerlo en .tga y cambiar el .mlt del objeto se resolvió de manera rápida.

3. Conclusión:

Este ejercicio, aunque al principio tuve problemas con la importación de la lámpara, se resolvió de manera sencilla. En general, la práctica fue entretenida y, gracias a esta y a la clase teórica, me quedó mucho más claro cómo funcionan las luces. Es cuestión de práctica para entender qué representa cada posición en el arreglo de cada luz; sin embargo, con una pequeña guía, este ejercicio fue entretenido y desafiante.

4.- Bibliografía:

LearnOpenGL - Basic Lighting. (s. f.). <https://learnopengl.com/Lighting/Basic-Lighting>