Núm. Cuenta: 314017347 Grupo: 12

DOCUMENTACIÓN Y MANUAL DE USUARIO

Laboratorio de Computación gráfica e interacción humano computadora.

Profesor: Carlos Aldair Román Balbuena

Alumno: Hernández Jiménez Diana Lisset

Manual de usuario

MANEJO DE CÁMARA

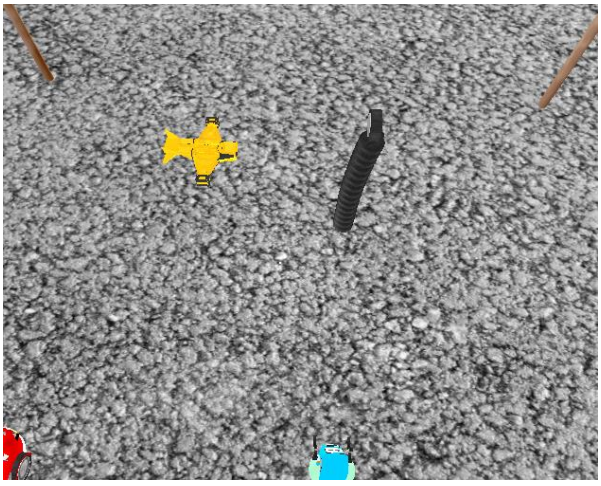
En este proyecto se utilizó la misma cámara que manejamos en las prácticas. Las teclas que se usan para poder desplazarse en el ambiente gráfico son:

- **W para ir hacia enfrente**
- **S para ir hacia atrás**
- **A para ir hacia la izquierda**
- **D para ir hacia la derecha.**

También usamos el mouse para poder mover la cámara.

Animaciones

Animación del dron con la tecla V: Con este se usaron funciones seno y coseno para poder simular un movimiento más curvo. Se ubica posicionado encima de la torre de llantas.



Núm. Cuenta: 314017347 Grupo: 12

Objetivo

- El alumno deberá aplicar y demostrar los conocimientos adquiridos durante todo el curso.
- El alumno creara 5 animaciones de los objetos presentados en las imágenes de referencia o implementados en el entorno, teniendo en cuenta el contexto en el que se encuentran.

Diagrama de Gantt

Alcances del proyecto y limitantes

A lo largo del proyecto se pudo comprender de mejor manera los conceptos vistos en las clases por lo tanto se pudo cumplir con los objetivos del proyecto y con los tiempos del mismo.

Se tuvieron ciertos problemas con los modelos debido a que mi software no me permitía exportar correctamente los modelos en tipo obj, debido a este problema se tuvo que suplir el modelo de un coche por un drone.

Documentacion de Codigo

Microsoft Visual Studio es un entorno de desarrollo integrado para Windows y macOS. Es compatible con múltiples lenguajes de programación, tales como C++, C#, Visual Basic .NET, F#, Java, Python, Ruby y PHP, al igual que entornos de desarrollo web, como ASP.NET MVC, Django, etc., a lo cual hay que sumarle las nuevas capacidades en línea bajo Windows Azure en forma del editor Mónaco.

Para este proyecto utilizamos el lenguaje de programación C++.

Para la creación de nuestro código comenzamos con la instancia de ciertas librerias que han sido mandadas a llamar en el encabezado del programa, como por ejemplo:

```
#include <iostream>
#include <cmath>

// GLEW
#include <GL/glew.h>

// GLFW
#include <GLFW/glfw3.h>

// Other Libs
#include "stb_image.h"
```

Creamos algunas funciones para implementarlas al final de nuestro programa, además de que creamos variables que utilizaremos ya sea en nuestras animaciones o en nuestro main.

```
// Function prototypes
void KeyCallback(GLFWwindow* window, int key, int scancode, int action, int mode);
void MouseCallback(GLFWwindow* window, double xPos, double yPos);
void DoMovement();
void animacion();

//Drone
bool animacionD = false;
float posDronX = -8.0f, posDronZ = -8.0f, angDron = 0.0f, rotDron = 1.570596f;
bool dirDron = false;
```

En nuestro main haremos las llamadas a nuestros modelos, es decir declaramos las rutas de nuestros modelos para poder implementarlo y situarlos en el mapa.

```
Model Llanta((char*)"Models/Llanta/llanta.obj");
Model Piso((char*)"Models/Esfera/piso.obj");
Model Luiguis((char*)"Models/Casa/Casa.obj");
```

```
// Casa
model = glm::mat4(1);
model = glm::translate(model, glm::vec3(35.0f, 0.0f, 20.0f));
model = glm::scale(model, glm::vec3(5.0f, 5.0f, 5.0f));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Luiguis.Draw(lightingShader);
```

En el main agregaremos los vértices e índices que utilizaremos como es el caso de los vértices del Skybox. A demás de crear nuestras matrices para formar nuestras figuras o posicionarlas.

```
GLuint VBO, VAO, EBO;
glGenVertexArrays(1, &VAO);
glGenBuffers(1, &VBO);
glGenBuffers(1, &EBO);
```

Las siguientes instrucciones nos muestran las matrices que ocuparemos para nuestro shader.

```
glUniformMatrix4fv(viewLoc, 1, GL_FALSE, glm::value_ptr(view));
glUniformMatrix4fv(projLoc, 1, GL_FALSE, glm::value_ptr(projection));
```

En nuestro main también podemos agregar instrucciones para jugar con las luces de nuestro escenario como es el caso de:

```
// Directional light
glUniform3f(glGetUniformLocation(lightingShader.Program, "dirLight.direction"), -0.2f, -1.0f, -0.3f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "dirLight.ambient"), 1.0f, 1.0f, 1.0f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "dirLight.diffuse"), 0.0f, 0.0f, 0.0f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "dirLight.specular"), 0.0f, 0.0f, 0.0f);
```

Terminando el main ocupamos nuestras funciones creadas al principio del programa

```
void animacion()
{

    //Animacion dron

    if (animacionD) {
        angDron += 0.01f;
        posDronX = posDronX + (0.2f * sin(angDron));
        posDronZ = posDronZ + (0.2f * cos(angDron));
        if (angDron > glm::radians(360.0f)) {
            angDron = 0.0f;
```

En la función KeyCallback es donde se define que realizara el programa al pulsar una tecla, como en el caso del dron que usamos la tecla V

```
if (glfwGetKey(window, GLFW_KEY_V) == GLFW_PRESS)
    animacionD ^= true;
```

Finalmente tenemos el control de cámara al utilizar nuestro mouse

```
void MouseCallback(GLFWwindow* window, double xPos, double yPos)
{
```