No. Cuenta: 316166496 Grupo: 8

Proyecto Final – Manual de usuario

LABORATORIO DE COMPUTACIÓN GRÁFICA

ING. CARLOS ALDAIR ROMAN BALBUENA

Fechas de entrega: 13 de mayo 2022

Introducción y objetivos:

En el siguiente documento se encontrará desarrollado el proyecto Final del laboratorio de Computación Gráfica e Interacción Humano Computadora. En él se utilizaron todas las herramientas aprendidas durante el curso desde modelación de objetos hasta animación sencilla y compleja para presentar un trabajo capaz de asemejarse al espacio de referencia basado en la serie de Netflix llamada Arcane. En este manual se explicará a detalle como se utilizan las funciones para activar las animaciones con las que se mueven los objetos establecidos, además de como en aspectos generales se cumple con cada uno de los puntos.

Alcance del proyecto:

El proyecto presentado tiene como objetivo crear un espacio virtual con modelado 3D que cumpla con 5 animaciones: 3 sencillas y 2 complejas. Donde cada pieza de modelado y la temática se asemeje a lo planteado en el primer archivo entregado de imagen de referencia. Se colocarán figuras geométricas con las que se podrá interactuar presionando alguna de las teclas.

Desarrollo:

- Imagen de referencia:

Como punto de partida es importante recordar la primer entrega de este proyecto en la cual se establecieron los puntos u objetos a crear, la arquitectura a alcanzar y la temática dentro o fuera de la fachada para que coincidiera con lo estipulado a desarrollar para respetar el aspecto a evaluar de realismo.

Se recreará una taberna llamada “la última gota” la cual está construida en la historia a base de chatarra debido a que no es una ciudad pionera en tecnología. Se podría categorizar como genero steampunk.

Fachada:

Cuarto:

Elementos dentro del cuarto:

1. Rocola
2. Mesas, Sillones de terciopelo y sillas
3. Botellas de vidrio, sifones y Barriles de madera
4. Dianas de prueba de tiro
5. Guantes de boxeador (Guantes Hextech de Vi)
6. Juguete de changuito con platillos
7. Hookah

Estos elementos son los que se plantearon en un inicio para recrear en OpenGl, pero se agregaron otros objetos como una esfera de disco y la barra del bartender.

- Desarrollo del proyecto:

El modelo desarrollado incluye los aspectos principales de la taberna.

- Fachada:

La fachada esta compuesta por ventanas y tuberías por las que pasa el vapor para proporcionar la energía. Además del cartel que anuncia el nombre de este en lo alto.

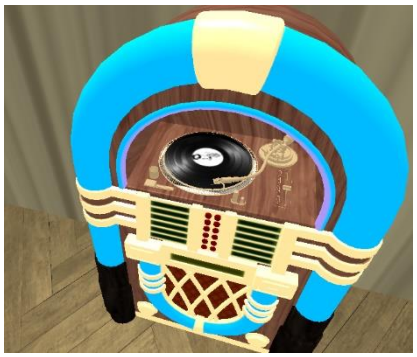
- Objetos inanimados:

Dentro del proyecto existen objetos que no tienen ningún tipo de movimiento por lo que se les considera como inanimados. Estos son los siguientes y componen parte de la decoración del interior.

- Barriles, mesas, sillones, bancos, guantes, Hookah, Barra de tragos.

- Animaciones:

Son aquellos que se activan luego de presionar una tecla. Se utilizaron 2 clasificaciones diferentes para ellas donde la primera son sencillas y las siguientes son complejas.



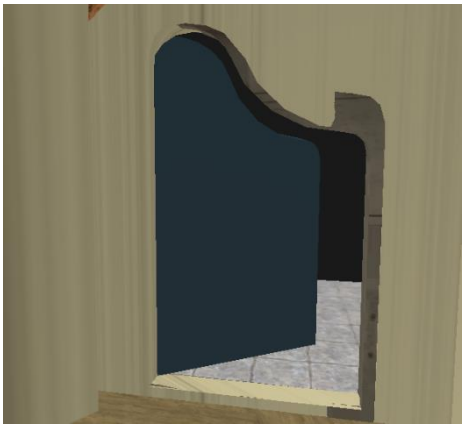
Animación sencilla:

1. El disco de la rocola:

Al presionar la letra U se activa la función que nos dará el proceso que aumentará de manera simple el valor de la rotación del disco para que simule como si se estuviera reproduciendo música.

2. Esfera de disco en el techo:

Al presionar la letra P se mantendrá girando la bola reflectante en el techo, si se vuelve a presionar esta se parará de inmediato.



3. Puerta:

Si se presiona la letra K se moverá la puerta de entrada, abriéndose y cerrándose.

Animación compleja:

4. Diana:

La animación compleja de la Diana consiste en activar y desactivar estados al modo del coche de la práctica 11 para que al momento de llegar a un punto cambie de dirección.

La Diana se moverá como si recorriera un rectángulo para que se le pueda atinar en movimiento. Para activar el animación se debe oprimir la i y si se presiona la o se detendrá en cualquier punto.

5. Changuito con platillos:

La animación compleja del mono tocando los platillos se hace utilizando una técnica conocida como Keyframes. Para que este funcione se debe presionar la L cada vez que se desee iniciar el movimiento de aplaudir.

• Diagrama de Gantt:

	Marzo			Abril					Mayo	
Actividades	11	18	25	1	8	15	22	29	6	12
Imagen de referencia										
Eliminación de funciones del profesor innecesarias para el proyecto.										
Creación de la fachada										
Carga y diseño de modelados										
Animaciones										
Corrección de errores										
Documentación										

Documentación del código:

Podemos dividir el código en tres partes fácilmente reconocibles y podemos poner el código que explica perfectamente que hace cierta parte del código para que quede claro como funciona. Así mismo, se colocan las partes más relevantes del código, de las cuales se modificaron:

1) La primera parte es donde se hacen las declaraciones de variables para su uso, la importación de librerías, modelos o creación de figuras geométricas a través de vértices para su construcción en el interior. Esa sección abarcará desde el comienzo del código hasta las primeras funciones.

```
/*Desarrolló 316166496 - Díaz Alan */
//Proyecto Final de Lab. Computación Gráfica
//Proyecto Final: Modelado en 3D del bar "The Last Drop" de la serie animada Arcane


//Library import - Importación de las librerias de opengl
#include <iostream>
#include <cmath>
#include <GL/glew.h> // GLEW
#include <GLFW/glfw3.h>  // GLFW
#include "stb_image.h"// Other Libs
#include <glm/glm.hpp>
#include <glm/gtc/matrix_transform.hpp>
#include <glm/gtc/type_ptr.hpp> // GLM Mathematics
#include "SOIL2/SOIL2.h"//Load Models

// Libraries for the control of camera, shaders, and correct texturization
// Librerias para el control de camara, texturizado y añadido de shaders
#include "Shader.h"
#include "Camera.h"
#include "Model.h"
#include "Texture.h"
#include "modelAnim.h"

// Function declarations - Declaracion de funciones
void KeyCallback(GLFWwindow *window, int key, int scancode, int action, int mode); //Función
que detecta el uso de una tecla - Key using function
void MouseCallback(GLFWwindow *window, double xPos, double yPos); //Detección del mouse -
Mouse Detector
void DoMovement(); //Función activa algún proceso mientras se presione la tecla - While
pressing a key a process is done
void animacion(); //Contiene los procesos para animación de objetos - Holds the instructions
for the animation process.

// Establece las dimensiones de la ventana - Window dimensions
const GLuint WIDTH = 800, HEIGHT = 600; //Ancho y alto
int SCREEN_WIDTH, SCREEN_HEIGHT;

// Valores predeterminados para la camara - Camera settings at the beginning
Camera  camera(glm::vec3(-100.0f, 2.0f, -45.0f));
GLfloat lastX = WIDTH / 2.0;
GLfloat lastY = HEIGHT / 2.0;
bool keys[1024];
bool firstMouse = true;
float range = 0.0f;
float rot = 90.0f;

//Variables para aumentar o disminuir la rotacion de algún objeto animado
// Values which save the rotation degrees for some objects in the animation process
```

```
float rotpuerta = 0; //rotar la puerta - Rotate door
float rotdisco = 0; //Rotar esfera reflectiva - Rotate disco Ball
float rotJuke = 0; //Rotar disco en rocola - Rotate Vinil disq
float movCamera = 0.0f;

//Variables para designar el movimiento de la Diana en cada eje
// Bull'seye parameters to move it on real time
float movKitY = 0.0;
float movKitZ = 0.0;
float rotKit = 0.0;

//Boleanos para activar o desactivar un estado en el recorrido de la diana.
//Booleans to switch between states for the Bull'seye movement.
bool circuito = false;
bool recorrido1 = true;
bool recorrido2 = false;
bool recorrido3 = false;
bool recorrido4 = false;

// Atributos de la luz en el medio - Light attributes
glm::vec3 lightPos(0.0f, 0.0f, 0.0f);
glm::vec3 PosIni(-95.0f, 1.0f, -45.0f);
glm::vec3 lightDirection(0.0f, -1.0f, -1.0f);

bool active;


// Deltatime
GLfloat deltaTime = 100.0f;     // Time between current frame and last frame
GLfloat lastFrame = 0.0f;       // Time of last frame

// Variables de posición inicial para la implementacion de animacion Keyframes
// KeyFrames animation process has initial axis values
float posX =PosIni.x, posY = PosIni.y, posZ = PosIni.z, rotRodIzq = 0, rotManIzq;
float rotManDer;


//Definición de frames máximos posibles a guardar
// Max frames for the technique
#define MAX_FRAMES 9
int i_max_steps = 190;
int i_curr_steps = 0;
typedef struct _frame
{
      //Variables para GUARDAR Key Frames
      float posX;        //Variable para PosicionX
      float posY;        //Variable para PosicionY
      float posZ;        //Variable para PosicionZ
      float incX;        //Variable para IncrementoX
      float incY;        //Variable para IncrementoY
      float incZ;        //Variable para IncrementoZ
      float rotInc;
      float rotInc3;
      float rotInc4;
      float rotManIzq; //Variable para Incremento en la rotación Mano Izq -- Increments
rotation value for the left hand of the monkey
      float rotManDer;//Variable para Incremento en la rotación Mano Der -- Increments
rotation value for the right hand of the monkey

}FRAME;

FRAME KeyFrame[MAX_FRAMES];
int FrameIndex = 0;            //Indice de la tupla para los frames - frameindex holder.
bool play = false;
```

```
bool aux = false;
bool puerta = false;
bool disco = false;
bool Juke = false;
int playIndex = 0;

// Posiciones de las luces de punto - Positions of the point lights
glm::vec3 pointLightPositions[] = {
	glm::vec3(posX,posY,posZ),
	glm::vec3(0,0,0),
	glm::vec3(0,0,0),
	glm::vec3(0,0,0)
};

glm::vec3 LightP1;
```

2) La segunda parte es aquella en la que se cargan los modelos, las luces y shaders dentro de la función main, así mismo, se transforman con funciones básicas ahí mismo para que tomen su lugar correspondiente.

```
// Cargamos los modelos y los nombramos - We load each 3d model and assign it to a model name.
	Model Mesa_circular((char*)"Models/Mesa_circular/Mesa_circular.obj");
	Model Hookah((char*)"Models/Hookah/Hookah.obj");
	Model Sofa((char*)"Models/Sofa/Sofa.obj");
	Model Guante((char*)"Models/Guante/Guante.obj");
	Model Luces((char*)"Models/House/Luces_esfericas.obj");
	Model palco((char*)"Models/House2/palco.obj");
	Model tuberias((char*)"Models/House2/tuberias.obj");
	Model wall((char*)"Models/House3/wall.obj");
	Model wall1((char*)"Models/House3/wall1.obj");
	Model wall2((char*)"Models/House3/wall2.obj");
	Model wall3((char*)"Models/Houseall/puerta_cortada.obj");
	Model wallcita((char*)"Models/Houseall/paredcita.obj");
	Model vapor((char*)"Models/House3/vapor.obj");
	Model altar((char*)"Models/House4/altar.obj");
	Model beer((char*)"Models/HouseAll/beer.obj");
	Model ventana((char*)"Models/HouseAll/ventana.obj");
	Model piso((char*)"Models/HouseAll/piso.obj");
	Model Puerta2((char*)"Models/HouseAll/puerta_azul.obj");
	Model techo((char*)"Models/Techo/techo.obj");
	Model Jukebox((char*)"Models/Jukebox/Jukebox.obj");
	Model Disque((char*)"Models/Jukebox/Jukebox_disque.obj");
	Model Luz_techo((char*)"Models/Luz_techo/Luz_techo.obj");
	Model Barra((char*)"Models/Barra/Barra.obj");
	Model Stool((char*)"Models/Stool/Stool.obj");
	Model Monkey((char*)"Models/Changuito/Monkey.obj");
	Model Monkey_izq((char*)"Models/Changuito/Monkey_Izq.obj");
	Model Monkey_der((char*)"Models/Changuito/Monkey_Der.obj");
	Model Diana((char*)"Models/Diana/Diana.obj");
	Model ball((char*)"Models/Disco/ball.obj");
	Model Chain((char*)"Models/Disco/Chain.obj");
	Model Barril((char*)"Models/Barriles/Barril.obj");


	//Inicialización de KeyFrames - KeyFrame Initialize
	// Cada Keyframe podra guardar la posicion o rotacion de la figura por cada indice
	// Saves each frame the times the key has been used
	for(int i=0; i<MAX_FRAMES; i++)
	{
		KeyFrame[i].posX = 0;
		KeyFrame[i].incX = 0;
		KeyFrame[i].incY = 0;
		KeyFrame[i].incZ = 0;
```

```
            KeyFrame[i].rotManIzq = 0;
            KeyFrame[i].rotManDer = 0;


//Establecemos los frames en cada indice de KeyFrames - Set Monkey movement for every index in
KeyFrames function
      rotManDer = -10;
      rotManIzq = 10;
      saveFrame();
      rotManDer = 18;
      rotManIzq = -18;
      saveFrame();
      rotManDer = -10;
      rotManIzq = 10;
      saveFrame();
      rotManDer = 18;
      rotManIzq = -18;
      saveFrame();
      rotManDer = -10;
      rotManIzq = 10;
      saveFrame();



// Here we set all the uniforms for the 5/6 types of lights we have. We have to set them
manually and index
            // the proper PointLight struct in the array to set each uniform variable.
            // // == ==========================
            // Aqui debemos de establecer los 5 de 6 tipos de luces que tenemos. Lo hacemos
manualmente para establecer cada una de las variables.
            // == ==========================
            // 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"),
0.2f, 0.2f, 0.2f);
            glUniform3f(glGetUniformLocation(lightingShader.Program, "dirLight.specular"),
0.5f, 0.5f, 0.5f);


            // Point light 1
            glUniform3f(glGetUniformLocation(lightingShader.Program,
"pointLights[0].position"), pointLightPositions[0].x, pointLightPositions[0].y,
pointLightPositions[0].z);
            glUniform3f(glGetUniformLocation(lightingShader.Program,
"pointLights[0].ambient"), 0.5f, 0.5f, 0.5f);
            glUniform3f(glGetUniformLocation(lightingShader.Program,
"pointLights[0].diffuse"), LightP1.x, LightP1.y, LightP1.z);
            glUniform3f(glGetUniformLocation(lightingShader.Program,
"pointLights[0].specular"), LightP1.x, LightP1.y, LightP1.z);
            glUniform1f(glGetUniformLocation(lightingShader.Program,
"pointLights[0].constant"), 1.0f);
            glUniform1f(glGetUniformLocation(lightingShader.Program,
"pointLights[0].linear"), 0.09f);
            glUniform1f(glGetUniformLocation(lightingShader.Program,
"pointLights[0].quadratic"), 0.032f);
```

```
//Carga de modelo
            // == ==========================
            // En esta parte se hace la carga de todos los modelos de nuestro proyecto. Les
aplicamos transformaciones basicas como el escalado por 2 para que sean el doble de su tamaño
y se aprecien bien.
            // Las traslaciones para que aparezcan en el punto exacto dentro o fuera de la
fachada y por último las rotaciones para que estén bien orientados.
            // == ==========================
            // In this section we load all the  models for our project. In here I had the
duty to transform this figures with the functions of scale, rotate or translate to get them
            // in the right position inside or outside the building.
            // == ==========================

            //Guante
            view = camera.GetViewMatrix();
            glm::mat4 model(1);
            tmp = model = glm::translate(model, glm::vec3(52, 10.3, -53));
            model = glm::translate(model, glm::vec3(posX, posY, posZ));
            model = glm::scale(model, glm::vec3(2));
            model = glm::rotate(model, glm::radians(2 * rot), glm::vec3(1.0f, 0.0f, 0.0));
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
            Guante.Draw(lightingShader);


            //Mesa
            view = camera.GetViewMatrix();
            model = glm::mat4(1);
            tmp = model = glm::translate(model, glm::vec3(85, 0.2, -80));
            model = glm::translate(model,glm::vec3(posX,posY,posZ));
            model = glm::scale(model, glm::vec3(5));
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
            Mesa_circular.Draw(lightingShader);

            //Mesa 2
            view = camera.GetViewMatrix();
            model = glm::mat4(1);
            tmp = model = glm::translate(model, glm::vec3(52, 0.2, -53));
            model = glm::translate(model, glm::vec3(posX, posY, posZ));
            model = glm::scale(model, glm::vec3(5));
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
            Mesa_circular.Draw(lightingShader);

            //Mesa 3
            view = camera.GetViewMatrix();
            model = glm::mat4(1);
            tmp = model = glm::translate(model, glm::vec3(110, 0.4, -45));
            model = glm::translate(model, glm::vec3(posX, posY, posZ));
            model = glm::scale(model, glm::vec3(5));
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
            Mesa_circular.Draw(lightingShader);

            //Monkey
            view = camera.GetViewMatrix();
            model = glm::mat4(1);
            tmp = model = glm::translate(model, glm::vec3(110, 13.2, -45));
            model = glm::translate(model, glm::vec3(posX, posY, posZ));
            model = glm::scale(model, glm::vec3(.2));
            glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
            Monkey.Draw(lightingShader);
```

3) La tercera sección corresponde a las funciones extras que permiten la animación de ciertas figuras apretando alguna tecla una sola vez.

```
void animacion()
{

		//Movimiento del changuito - Monkey's movement
		// Cuando play sea verdadero nos dejará entrar a su proceso donde a través de la interpolacion se puede animar el movimiento del objeto.
		//Play will be true when we press L in our keyboard, and it'll use interpolation to make the movement from one site to another.
		if (play)
		{
			if (i_curr_steps >= i_max_steps) //end of animation between frames?
			{
				playIndex++;
				if (playIndex>FrameIndex - 2)   //end of total animation?
				{
					printf("termina anim\n");
					playIndex = 0;
					play = false;
				}
				else //Next frame interpolations
				{
					i_curr_steps = 0; //Reset counter
											//Interpolation
					interpolation();
				}
			}
			else
			{
				//Dibujamos la animación - Draw animation
				posX += KeyFrame[playIndex].incX;
				posY += KeyFrame[playIndex].incY;
				posZ += KeyFrame[playIndex].incZ;

				rotRodIzq += KeyFrame[playIndex].rotInc;
				rotManIzq += KeyFrame[playIndex].rotInc3;
				rotManDer += KeyFrame[playIndex].rotInc4;


				i_curr_steps++;
			}

		}

		/* Si circuito está activo entonces se producirá el movimiento de la Diana que se actualiza cuando rebase cierto rango para avanzar en otro eje.
		   If it's active then our Bullseye will move in a square shape*/
		if (circuito)
		{
			//Ejercicio 1 Movimiento sobre el rectangulo
			if (recorrido1)
			{
				movKitY += 0.25f; //Mueve la diana en el eje Y  -  Moves the bullseye in Y axis
				if (movKitY > 11.5)
				{
					recorrido1 = false; //Desactivamos el estado para poder entrar al siguiente de inmediato
					recorrido2 = true; // We deactivate the states to change the direction in which it continues moving.
```

```
                        }
                  }
                  if (recorrido2)
                  {
                        movKitZ -= 0.25f; //Mueve la diana en el eje Z  -  Moves the
bullseye in Z axis
                        if (movKitZ < -11)
                        {
                              recorrido2 = false;
                              recorrido3 = true;

                        }
                  }

                  if (recorrido3)
                  {
                        movKitY -= 0.25f; //Mueve la diana en el eje Y  -  Moves the
bullseye in Y axis
                        if (movKitY < 3.2)
                        {
                              recorrido3 = false;
                              recorrido4 = true;
                        }
                  }

                  if (recorrido4)
                  {
                        movKitZ += 0.25f; //Mueve la diana en el eje Z  -  Moves the
bullseye in Z axis
                        if (movKitZ > 11)
                        {
                              recorrido4 = false;
                              recorrido1 = true;
                        }
                  }
            }

            //Si la puerta está activada entonces oscilará su valor de rotación para que
simule que está abriendose
            // If the door is activated the movement between one and othe state is declared
            if (puerta)
            {
                  if (55 > rotpuerta) {
                        rotpuerta += 0.25f;
                  }
                  if (55 <= rotpuerta) {
                        aux = true;
                        puerta = false;
                  }
            }
            if (aux) {
                        rotpuerta -= 0.25f;
                        if (0 > rotpuerta) {
                              puerta = true;
                              aux = false;
                        }
                  }


            // Para la esfera de disco se aumentará en razon de .3 para que gira lentamente.
            // If activated it while spin by itself
            if (disco) {
                  rotdisco += 0.3f;
            }
```

```
		// Para el disco dentro de la rocola se aumentará en razon de .3 para que gira lentamente,´nuevamente.
		// If activated it while spin by itself

		if (Juke) {
			rotJuke += 0.3f;
		}
	}



void KeyCallback(GLFWwindow *window, int key, int scancode, int action, int mode)
{
	//Luego de presionar la L se hace la interpolación para mover la figura en un eje.
	//After activating L the interpolation is done to make it rotate in the same axis.

	if (keys[GLFW_KEY_L])
	{
		if (play == false && (FrameIndex > 1))
		{

			resetElements();
			//First Interpolation
			interpolation();

			play = true;
			playIndex = 0;
			i_curr_steps = 0;
		}
		else
		{
			play = false;
		}

	}


	//Si presionamos la tecla cambiará el valor de su boleano por defecto. En este caso pasará de false a true
	//When pushing the Key it will activate the movement detailed on the animation function
	if (keys[GLFW_KEY_K])
	{
		puerta = not puerta;

	}

	if (keys[GLFW_KEY_P])
	{
		disco = not disco;

	}

	if (keys[GLFW_KEY_U])
	{
		Juke = not Juke;

	}

	//Debido a esta función cada vez que presionamos la tecla solo cuenta como si se apretara una vez, gracias a esto si mantenemos
	//la tecla oprimida aún así solo se activará una vez el comando
```

```
	//This instructions let us push a key without updating more than once the value of the variable while still pressing the key.
	if (GLFW_KEY_ESCAPE == key && GLFW_PRESS == action)
	{
		glfwSetWindowShouldClose(window, GL_TRUE);
	}

	if (key >= 0 && key < 1024)
	{
		if (action == GLFW_PRESS)
		{
			keys[key] = true;
		}
		else if (action == GLFW_RELEASE)
		{
			keys[key] = false;
		}
	}
	// Prende la luz 1
	//Turns on the light 1
	if (keys[GLFW_KEY_SPACE])
	{
		active = !active;
		if (active)
			LightP1 = glm::vec3(1.0f, 0.0f, 0.0f);
		else
			LightP1 = glm::vec3(0.0f, 0.0f, 0.0f);
	}
}

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

	if (firstMouse)
	{
		lastX = xPos;
		lastY = yPos;
		firstMouse = false;
	}

	GLfloat xOffset = xPos - lastX;
	GLfloat yOffset = lastY - yPos;  // Reversed since y-coordinates go from bottom to left

	lastX = xPos;
	lastY = yPos;

	camera.ProcessMouseMovement(xOffset, yOffset);
}



// Moves/changes the camera positions based on user input
void DoMovement()
{

	if (keys[GLFW_KEY_1])
	{

		movCamera = 2.5f;//Manda una velocidad de 0.01 a la camara automatica

	}

	// Inicia el movimiento para la diana
	// Activates the animation for the bullseye
	if (keys[GLFW_KEY_I])
```

```
	{
		circuito = true;
	}

	//Detiene la diana en cualquier punto
	// Deactivates the animation for the bullseye
	if (keys[GLFW_KEY_O])
	{
		circuito = false;
	}

	// Controles de la camara
	// Camera controls
	if (keys[GLFW_KEY_W] || keys[GLFW_KEY_UP]) //Permite que la camara se mueva en vertical
- Let's the camera move up
	{
		camera.ProcessKeyboard(FORWARD, deltaTime);

	}

	if (keys[GLFW_KEY_S] || keys[GLFW_KEY_DOWN]) //Permite que la camara se mueva en
vertical, hacia abajo - Let's the camera move down
	{
		camera.ProcessKeyboard(BACKWARD, deltaTime);


	}

	if (keys[GLFW_KEY_A] || keys[GLFW_KEY_LEFT]) //Permite que la camara se mueva hacia la
izquierda - Let's the camera move left
	{
		camera.ProcessKeyboard(LEFT, deltaTime);


	}

	if (keys[GLFW_KEY_D] || keys[GLFW_KEY_RIGHT]) //Permite que la camara se mueva a la
derecha - Let's the camera move to the right
	{
		camera.ProcessKeyboard(RIGHT, deltaTime);
	}


}




//FUnción para guardar el estado de la figura en el preciso momento en el que se mande a
llamar.
// SaveFrame function as its name says, saves the KeyFrame in that moment.
void saveFrame(void)
{

	printf("posx %f\n", posX);

	KeyFrame[FrameIndex].posX = posX; //Guardamos la posición con respecto al eje - Saves
the axis position
	KeyFrame[FrameIndex].posY = posY;
	KeyFrame[FrameIndex].posZ = posZ;

	KeyFrame[FrameIndex].rotManIzq = rotManIzq; //GUardamos la rotación de la figura - We
save the new rotation of the hand
	KeyFrame[FrameIndex].rotManDer = rotManDer;
```

```
	FrameIndex++;
}


// Regresa la posición actual de la pieza a la original- returns every movement in the figure
to the beginning preset.
void resetElements(void)
{
	posX = KeyFrame[0].posX;
	posY = KeyFrame[0].posY;
	posZ = KeyFrame[0].posZ;

	rotManIzq = KeyFrame[0].rotManIzq;
	rotManDer = KeyFrame[0].rotManDer;

}

// FUncion de interpolación para calcular las posiciones intermedias entre la primera y
segunda ubicación - Interpolation of first and last position
void interpolation(void)
{

	KeyFrame[playIndex].incX = (KeyFrame[playIndex + 1].posX - KeyFrame[playIndex].posX) /
i_max_steps;
	KeyFrame[playIndex].incY = (KeyFrame[playIndex + 1].posY - KeyFrame[playIndex].posY) /
i_max_steps;
	KeyFrame[playIndex].incZ = (KeyFrame[playIndex + 1].posZ - KeyFrame[playIndex].posZ) /
i_max_steps;

	KeyFrame[playIndex].rotInc3 = (KeyFrame[playIndex + 1].rotManIzq -
KeyFrame[playIndex].rotManIzq) / i_max_steps;
	KeyFrame[playIndex].rotInc4 = (KeyFrame[playIndex + 1].rotManDer -
KeyFrame[playIndex].rotManDer) / i_max_steps;
}
```

Conclusiones y comentarios finales:

Finalmente, en este proyecto se aplicaron todas las técnicas de modelado y animación vistas en el curso con el fin de demostrar el conocimiento aprendido o reforzarlo en todo caso. Me quedo satisfecho con el esfuerzo que le metí al proyecto porque fueron demasiadas horas y pese a que me encontré bastantes adversidades a la hora de desarrollarlo se puede decir que llegue a la meta, cumplí con la entrega, además de que pese a que no es exacto a las imágenes de referencia me aproximé a ello. La habilidades que he adquirido en Maya, GIMP, Github y conocimientos para programar en Visual me asombran porque logré dominarlo en un solo semestre.

Me habría gustado conocer desde el principio sobre animación compleja para poder definir desde ese momento que objetos animar porque si se reducen las posibilidades una vez que ya fijaste una temática y en las imágenes de referencia no existen más objetos con movilidad.

En conclusión, logré animar los objetos en cuestión y se modeló el espacio de forma exitosa.