UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Facultad de ingeniería

Laboratorio de Computación Gráfica e Interacción Humano-Computadora

MANUAL TÉCNICO

Alumnos:

Diana Paola Sanjuan Aldape

Profesor:

Ing. Jose Roque Román Guadarrama

Grupo 11

Semestre 2022-1

ÍNDICE

I. INTRODUCCIÓN

C++ es un lenguaje de programación diseñado en 1979 por Bjarne Stroustrup. La intención de su creación fue extender al lenguaje de programación C mecanismos que permiten la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, C++ es un lenguaje híbrido.

Posteriormente se añadieron facilidades de programación genérica, que se sumaron a los paradigmas de programación estructurada y programación orientada a objetos. Por esto se suele decir que el C++ es un lenguaje de programación multiparadigma.

Actualmente existe un estándar, denominado ISO C++, al que se han adherido la mayoría de los fabricantes de compiladores más modernos. Existen también algunos intérpretes, tales como ROOT.

El nombre "C++" fue propuesto por Rick Mascitti en el año 1983, cuando el lenguaje fue utilizado por primera vez fuera de un laboratorio científico. Antes se había usado el nombre "C con clases". En C++, la expresión "C++" significa "incremento de C" y se refiere a que C++ es una extensión de C.

II. PROPÓSITO

Realizar un ambiente virtual, en este caso un recorrido por el mundo de Los padrinos mágicos, cumpliendo los requisitos planteados.

III. ALCANCE

Este proyecto abarca una parte de la casa que se ve en la serie “Los padrinos mágicos” así como el cuarto del protagonista Timmy Turner y el patio que se presenta en la serie. Además de poder interactuar con algunos de los objetos que se encuentran dentro o fuera de la casa.

IV. DESCRIPCIÓN TÉCNICA

Para la implementación de este proyecto fue necesario considerar los requerimientos del mismo, es decir, el cómo debía estar implementado el escenario.

A. OBJETOS

Para la creación de objetos en OpengL se puede hacer de dos maneras:

Contar con una herramienta externa de Modelado 3D, este tipo de modelado reduce considerablemente el tiempo de realización, no ahondaré en el manejo de Blender en el presente manual ya que no es el objetivo de este.

La segunda forma es por modelado geométrico y jerárquico dentro de Visual Studio, el proceso es más tardado y a veces más complicado.

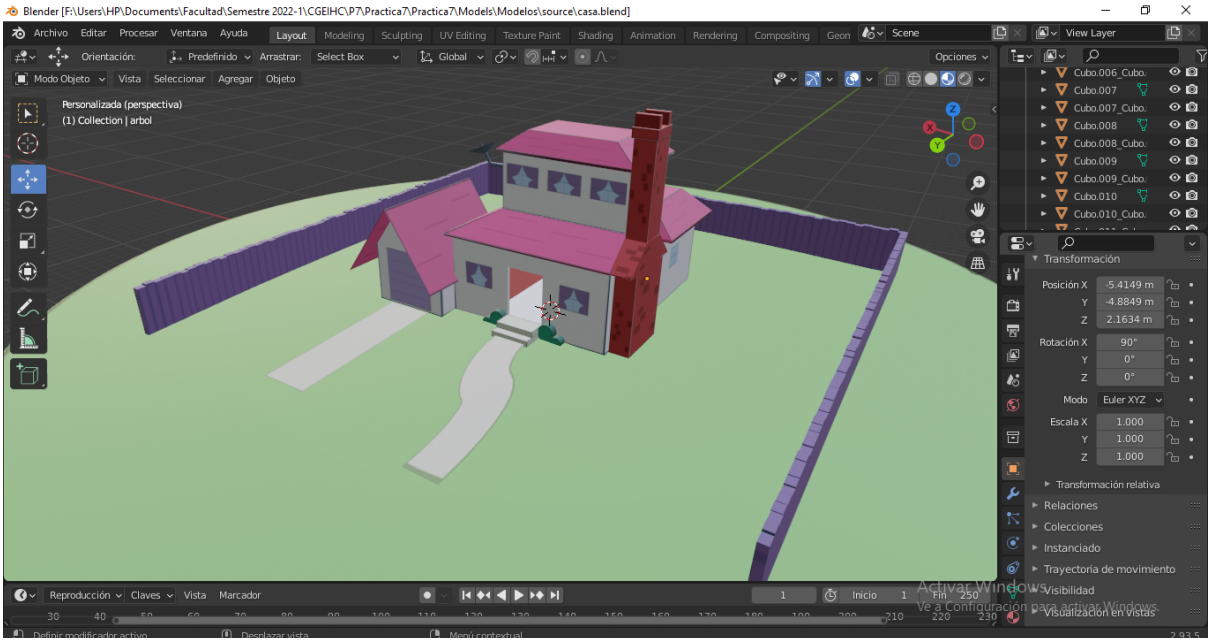

Fig 1. Interfaz de Blender

Para ambos tipos de modelado se requiere texturizar los objetos, dichas texturas deberán ser nxn y pueden ser de formatos distintos, se usó tga en esta aplicación.

Las texturas se declaran:

```
Texture brickTexture;
Texture dirtTexture;
Texture pisoTexture;
Texture Tagave;
Texture MunecoNieve;
Texture zanahoria;
Texture boton;
Texture sombrero;
Texture verde;
Texture brazoMuneco;
```

Fig 2 . Declaración de Texturas

A continuación si los modelos son externos igual deben de declararse:

```
Model Camino_M;
Model casa;
Model sillon;
Model cama;
Model buro;
Model tv;
Model escritorio;
Model compu;
Model silla;
Model puerta;
Model marcoPuerta;
Model arbol;
Model regalo;
Model regaloTapa;
Model regaloCaja;
Model arbolRegalo;
Model pelota;
//Robot
Model Cuerpo;
Model PiernaDer;
Model PiernaIzq;
Model BrazoDer;
Model BrazoIzq;
//Dino
Model Dino;
Model bocaDino;
//Tanque
Model cuerpoTank;
Model torreta;
Model ruedas;
Model llantaGrande;
Model llantaChica;
//Personaje
Model timmy_cabeza;
Model timmy_tronco;
Model pierna_izq;
Model pierna_izq_abajo;
Model pierna_der;
Model pierna_der_abajo;
Model brazo_izq;
Model brazo_der;
Model mano_der;
Model mano_izq;
```

Fig 3. Declaración de modelos

Se establece la ubicación de los mismos, modelos y texturas respectivamente.

```
brickTexture = Texture("Textures/brick.png");
brickTexture.LoadTextureA();
dirtTexture = Texture("Textures/dirt.png");
dirtTexture.LoadTextureA();
pisoTexture = Texture("Textures/piso2.tga");
pisoTexture.LoadTextureA();
Tagave = Texture("Textures/Agave.tga");
Tagave.LoadTextureA();
MunecoNieve = Texture("Textures/munecoNieve.png");
MunecoNieve.LoadTexture();
zanahoria = Texture("Textures/zanahoria.png");
zanahoria.LoadTexture();
boton = Texture("Textures/boton.png");
boton.LoadTexture();
sombrero = Texture("Textures/sombrero.png");
sombrero.LoadTexture();
verde = Texture("Textures/verde.png");
verde.LoadTexture();
brazoMuneco = Texture("Textures/arbolAnimCentro.png");
brazoMuneco.LoadTexture();
Material_brillante = Material(4.0f, 256);
Material_opaco = Material(0.3f, 4);
Timmy = Material(0.2, 1);
```

Fig 4. Ubicación de Texturas

```
        //Casa
        casa = Model();
        casa.LoadModel("Models/casaCompleta.obj");
        //Sala
        sillon = Model();
        sillon.LoadModel("Models/sillon.obj");
        tv = Model();
        tv.LoadModel("Models/tv.obj");
        marcoPuerta = Model();
        marcoPuerta.LoadModel("Models/marco_puerta.obj");
        puerta = Model();
        puerta.LoadModel("Models/puerta.obj");
        arbol = Model();
        arbol.LoadModel("Models/arbol.obj");
        regalo = Model();
        regalo.LoadModel("Models/regalo.obj");
        pelota = Model();
        pelota.LoadModel("Models/pelota.obj");
        regaloTapa = Model();
        regaloTapa.LoadModel("Models/regaloTapa.obj");
        regaloCaja = Model();
        regaloCaja.LoadModel("Models/regaloCaja.obj");
        arbolRegalo = Model();
        arbolRegalo.LoadModel("Models/arbolAnim.obj");
pierna_izq.LoadModel("Models/Timmy/pierna_izq_arriba.obj");
pierna_izq_abajo = Model();
pierna_izq_abajo.LoadModel("Models/Timmy/pierna_izq_abajo.obj");
brazo_der = Model();
brazo_der.LoadModel("Models/Timmy/brazo_sup_der.obj");
brazo_izq = Model();
brazo_izq.LoadModel("Models/Timmy/brazo_sup_izq.obj");
mano_der = Model();
mano_der.LoadModel("Models/Timmy/muneca_der.obj");
mano_izq = Model();
mano_izq.LoadModel("Models/Timmy/muneca_izq.obj");
//Dino
Dino = Model();
Dino.LoadModel("Models/Dinosaurio/dinosaurio.obj");
bocaDino = Model();
bocaDino.LoadModel("Models/Dinosaurio/bocadinosaurio.obj");
//Tanque
cuerpoTank = Model();
cuerpoTank.LoadModel("Models/Tank/estructura2.obj");
ruedas = Model();
ruedas.LoadModel("Models/Tank/ruedas.obj");
llantaGrande = Model();
llantaGrande.LoadModel("Models/Tank/llantaGrande.obj");
llantaChica = Model();
llantaChica.LoadModel("Models/Tank/llantaChica.obj");
torreta = Model();
torreta.LoadModel("Models/Tank/torreta.obj");
```

Fig 5. Ubicación de modelos

Finalmente, se procede a instanciar los objetos, para ello a veces es necesario declarar variables auxiliares:

```
glm::mat4 model(1.0);
glm::mat4 modelaux(1.0);
glm::mat4 modelauxTimmy(1.0);
glm::mat4 modelauxTank(1.0);
glm::mat4 modelauxRegalo(1.0);
glm::mat4 modelauxBrazo(1.0);
glm::mat4 modelauxMuneco(1.0);
glm::mat4 modelauxMunecoBotones(1.0);
glm::mat4 modelauxMunecoBrazos(1.0);
```

Fig 6. Matrices auxiliares

Ahora si, se resetea la matriz y se instancian los objetos en la posición requerida.

```
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, -6.0f, 0.0f));
model = glm::scale(model, glm::vec3(30.0f, 1.0f, 30.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
pisoTexture.UseTexture();
meshList[2]->RenderMesh();
//Casa
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, -3.0f, -15.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
casa.RenderModel();
modelaux = model;
model = glm::translate(model, glm::vec3(5.2f, 0.0f, -1.0f));
model = glm::scale(model, glm::vec3(4.0f, 4.0f, 3.5f));
model = glm::rotate(model, 180 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
sillon.RenderModel();
model = modelaux;
model = glm::translate(model, glm::vec3(-0.8f, 0.0f, -1.0f));
model = glm::rotate(model, 180 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
tv.RenderModel();
```

Fig 7. Objetos instanciados

Para fines prácticos, solo mostraré algunos objetos.

```
//Muñeco de nieve
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(10.0f, -3.0f, 3.8f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//F
MunecoNieve.UseTexture();
sp.render(); //Renderiza esfera
model = glm::translate(model, glm::vec3(0.0f, 2.5f, 0.0f));
model = glm::scale(model, glm::vec3(0.7f, 0.7f, 0.7f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
sp.render();
model = glm::translate(model, glm::vec3(0.0f, 2.5f, 0.0f));
model = glm::scale(model, glm::vec3(0.7f, 0.7f, 0.7f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
sp.render();
modelauxMuneco = model;
model = glm::translate(model, glm::vec3(0.0f, 0.3f, 2.5f));
model = glm::rotate(model, 90 * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.3f, 1.2f, 0.3f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
zanahoria.UseTexture();
meshList[1]->RenderMesh();
model = modelauxMuneco;
model = glm::translate(model, glm::vec3(0.7f, 0.8f, 1.7f));
model = glm::scale(model, glm::vec3(0.13f, 0.13f, 0.07f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
boton.UseTexture();
```

Fig 8. Objetos creados por modelado geométrico y jerárquico

B. ILUMINACIÓN

Para lograr luz en el escenario se consideran 3 tipos de iluminación, luz de punto (Point Light), luz de faro (Spotlight) y luz direccional (Directional Light), cada una de estas luces cumplieron un papel fundamental en el escenario y su implementación fue de la siguiente forma.

Para la luz del ambiente, es decir, la luz del sol y de la noche se utilizó una luz con un código RGB de 0.9, 0.95, 1.0, y la dirección se ubicó hacia el eje y positivo.

```
mainLight = DirectionalLight(0.9f, 0.95f, 1.0f,
    1.0f, 1.0f,
    0.0f, 1.0f, 0.0f);
```

Fig 9. Parámetros Directional Light

Las Point Lights se utilizan para iluminar el interior de la casa en la noche, las posiciones, direcciones, color y valores de la ecuación cuadrática se observan a continuación.

```
pointLights[0] = PointLight(1.0f, 1.0f, 1.0f,
    0.0f, 1.0f,
    5.0f, 2.0f, -15.0f,
    1.0f, 1.0f, 1.0f);
pointLightCount++;
pointLights[1] = PointLight(1.0f, 0.0f, 1.0f,
    0.0f, 1.0f,
    2.0f, 7.8f, -13.0f,
    1.0f, 1.0f, 0.1f);
pointLightCount++;
```

Fig 10. Parámetros Point Lights

Finalmente, la Spotlight se utiliza para iluminar la sala de la casa en modo fiesta.

```
spotLights[0] = SpotLight(0.0f, 1.0f, 0.0f,
    0.5f, 2.0f,
    2.09f, 1.51f, -16.0f,
    -0.09f, -0.6f, -0.76f,
    1.0f, 0.0f, 0.0f,
    5.0f);
spotLightCount++;

spotLights[1] = SpotLight(1.0f, 0.0f, 0.0f,
    1.0f, 2.0f,
    //1.2f, 0.9f, -20.0f,
    4.09f, 1.51f, -15.0f,
    0.45f, -0.7f, -56.0f,
    1.0f, 0.0f, 0.0f,
    5.0f);
spotLightCount++;

spotLights[2] = SpotLight(1.0f, 0.0f, 1.0f,
    1.0f, 2.0f,
    3.09f, 1.51f, -18.0f,
    -0.54f, -0.36f, -0.75f,
    1.0f, 0.0f, 0.0f,
    5.0f);
spotLightCount++;
```

Fig 11. Parámetros Spotlights

Para ello fueron necesarias 3 funciones, que permiten alternar la dirección de las luces

```
glm::vec3 dirLight(dirLightx, -1.0f, -0.4);
spotLights[0].SetDir(dirLight);
glm::vec3 dirLight1(dirLightFx, -1.0f, -0.4f);
spotLights[1].SetDir(dirLight1);
glm::vec3 dirLight2(-0.0, -1.0f, dirLightFz);
spotLights[2].SetDir(dirLight2);
```

Fig 12. Funciones Movimiento Spotlights

También se implementó un SkyBo, el cual se explicará más adelante, este se configuró junto con un cambio de luz en el escenario, cuando se hace de noche la luz se modifica a los valores: (0.4f, 0.4f, 0.8f)

C. SKYBOX

Se implementaron 2 skybox, uno para la noche y otro para el día, esto se logra de la siguiente manera:

Primero se declaran los Skybox:

```
Skybox skyboxDia;
Skybox skyboxNoche;

glm::vec3 luzNoche(0.4f, 0.4f, 0.8f);
glm::vec3 luzDia(0.9f, 0.9f, 1.0f);
```

Fig 13. Declaración de Skybox

Para que cambiara automáticamente se agregó un ciclo.

```
if (skyDia)
{
    skyboxDia.DrawSkybox(camera.calculateViewMatrix(), projection);
    mainLight.SetDiaNoche(luzDia);
    ciclo += 1.0f;
    if (ciclo > 1000)
    {
        skyDia = false;
        skynoche = true;
    }
    spotLightCount = -1;

}
if (skynoche)
{
    skyboxNoche.DrawSkybox(camera.calculateViewMatrix(), projection);
    ciclo -= 1.0f;
    mainLight.SetDiaNoche(luzNoche);
    if (ciclo < 0)
    {
        skynoche = false;
        skyDia = true;
    }
}
```

Fig 14. Condicionales para Skybox

D. ANIMACIONES

Para realizar ciertas tareas tanto de animaciones como de Vistas, Audio e Iluminación se dependía de ciertas entradas ingresadas mediante el teclado, para esto se usa el archivo Windows.cpp que utiliza un encabezado Windows.h, donde se declaran las variables y se generan funciones que serán llamadas en el código utilizado para realizar las tareas indicadas.
Las declaraciones de todas las funciones ingresadas por teclado es el siguiente:

```
GLfloat getBufferWidth() { return bufferWidth; }
GLfloat getBufferHeight() { return bufferHeight; }
GLfloat getXChange();
GLfloat getYChange();
GLfloat getmuevex() { return muevex; }
GLfloat getrota() { return rota; }
GLfloat getmov_helices() { return mov_helices; }
GLfloat getmovy() { return movy; }
GLfloat getmovx() { return movx; }
GLfloat getmovz() { return movz; }
GLfloat getrot() { return rot; }
GLfloat getposLightx() { return posLightx; }
GLfloat getposLighty() { return posLighty; }
GLfloat getdirLightx() { return dirLightx; }
GLfloat getdirLighty() { return dirLighty; }
GLfloat getposLightFx() { return posLightFx; }
GLfloat getposLightFz() { return posLightFz; }
GLfloat getdirLightFx() { return dirLightFx; }
GLfloat getdirLightFz() { return dirLightFz; }
GLfloat getcircuitoTanq() { return circuitoTanq; }
GLfloat getRegaloAnim() { return RegaloAnim; }
GLfloat getmagia() { return magia; }
GLfloat getProyectil() { return Proyectil; }
GLfloat getvistaCuarto() { return vistaCuarto; }
GLfloat getvistaAerea() { return vistaAerea; }
GLfloat getvistaSala() { return vistaSala; }
```

```
GLFWwindow* mainWindow;
GLint width, height;
bool keys[1024];
GLint bufferWidth, bufferH
void createCallbacks();
GLfloat lastX;
GLfloat lastY;
GLfloat xChange;
GLfloat yChange;
GLfloat muevex;
GLfloat rota;
GLfloat mov_helices;
GLfloat movy;
GLfloat movx;
GLfloat movz;
GLfloat rot;
GLfloat posLightx;
GLfloat posLighty;
GLfloat dirLightx;
GLfloat dirLighty;
GLfloat posLightFx;
GLfloat posLightFz;
GLfloat dirLightFx;
GLfloat dirLightFz;
GLfloat circuitoTanq;
GLfloat RegaloAnim;
GLfloat magia;
```

Fig 15. Archivo Windows.h, declaración de funciones y de variables

```
if (key == GLFW_KEY_ESCAPE && action == GLFW_PF
{
    glfwSetWindowShouldClose(window, GL_TRUE);
}
if (key == GLFW_KEY_H)
{
    theWindow-> circuitoTanq = true;
    theWindow->RegaloAnim = false;
    theWindow->magia = false;
}
if (key == GLFW_KEY_J)
{
    theWindow->RegaloAnim = true;
    theWindow->circuitoTanq = false;
    theWindow->magia = false;
}
if (key == GLFW_KEY_K)
{
    theWindow->magia = true;
    theWindow->circuitoTanq = false;
    theWindow->RegaloAnim = false;
}

if (key == GLFW_KEY_L)
```

```
if (key == GLFW_KEY_O)
{
    theWindow->vistaSala = true;
    theWindow->vistaCuarto = false;
    theWindow->vistaAerea = false;
    theWindow->Vista = false;
}
if (key == GLFW_KEY_N)
{
    theWindow->espectaculoLuces = true;
}
if (key == GLFW_KEY_M)
{
    theWindow->espectaculoLuces = false;
}
if (key == GLFW_KEY_Z)
{
    theWindow->movPuertaF = true;
}
if (key == GLFW_KEY_X)
{
    theWindow->recorrido = true;
}
```

Fig 16. Archivo Windows.cpp

- Tanque

Un tanque rueda y apunta hacia la casa de Timmy Turner y hace mediante una instrucción por teclado, ubicada en el archivo previamente mencionado.

```
if (mainWindow.getcircuitoTanq())
{
    if (movTanqX < 20.0f)
    {
        //printf("Movimiento x = %f", movT
        movTanqX += 3.0 * deltaTime;
        rotLlantas += 10.0f * deltaTime;
    }
    if (rotTorretaIzq)
    {
        rotTorreta += 5.0f * deltaTime;
        if (rotTorreta >= 90)
        {
            rotTorretaIzq = false;
            rotTorretaRegreso = true;
        }
    }
    if (rotTorretaRegreso)
    {
        rotTorreta -= 5.0f * deltaTime;
        if (rotTorreta < 0)
        {
            rotTorretaRegreso = false;
            rotTorretaDer = true;

        }
    }
```

```
    if (rotTorretaRegreso)
    {
        rotTorreta -= 5.0f * deltaTime;
        if (rotTorreta < 0)
        {
            rotTorretaRegreso = false;
            rotTorretaDer = true;

        }
    }
    if (rotTorretaDer)
    {
        rotTorreta -= 5.0f * deltaTime;
        if (rotTorreta < -90)
        {
            rotTorretaDer = false;
            rotTorretaRegreso2 = true;
        }
    }
    if (rotTorretaRegreso2)
    {
        rotTorreta += 5.0f * deltaTime;
        if (rotTorreta > 0)
        {
            rotTorretaRegreso2 = false;
            rotTorretaIzq = true;
        }
    }
```

Fig 17. Animación tanque

- Cajita de Música

Una caja de música, con un árbol de navidad en su interior, gira mientras se escucha una tonada navideña.

```
if (mainWindow.getRegaloAnim() )
{
    if (Regalo)
    {
        rotTapa += 6.5f * deltaTime;
        movArbol += 0.1f * deltaTime;
        while (i < 4) {
            engine->play2D("media/caja_musica.wav", false, false, true);
            printf("i = %i", i);
            i++;
            break;
        }
        if (movArbol > 1.15)
        {
            Regalo = false;
        }
    }
    rotArbol += 4.0f * deltaTime;
}
```

Fig 18. Animación Cajita Musical

- Magia

Un dinosaurio pequeño, crece del tamaño del cuarto de Timmy.

```
if (mainWindow.getmagia())
{
    if (crecer < 0.09)
    {
        crecer += 0.02f * deltaTime;
        movy  += 0.15f * deltaTime;
        movx -= 0.5f * deltaTime;
        while (i < 2)  {
            i ++;
            engine->play2D("media/bell.wav", false, false, true);
            printf("i = %i", i);
            break;
        }
    }
    else
    {
        magia = false;
    }

}
```

Fig 19. Animación Magia

- Timmy

Timmy se dirige a su casa, para llegar y prender la música y las luces para la fiesta de navidad.

```
if (mainWindow.getrecorrido())
{
    if (recorrido1)
    {
        //posX += 0.5f;
        rotTimmy = 90.0f;
        posZ -= 0.009;
        if (auxMovTimmy)
        {
            rotRodDer = 0.0f;
            rotRodDerS += 0.4f;
            rotRodIzqS -= 0.4f;
            rotRodIzq -= 0.5f;
            rotBraDerS -= 0.4f;
            rotBraIzqS += 0.4f;
            if (rotRodDerS > 30)
            {
                auxMovTimmy = false;

            }
        }
        if (!auxMovTimmy)
        {
            rotRodDerS -= 0.4f;
            rotRodIzqS += 0.4f;
            rotRodIzq = 0.0f;
            rotRodDer -= 0.5f;
            rotBraDerS += 0.4f;
            rotBraIzqS -= 0.4f;
            if (rotRodDerS < -30)
            {
                auxMovTimmy = true;
            }
        }
        if (posZ < -7.7f)
        {
            rotRodDerS = 0.0f;
            rotRodIzqS = 0.0f;
            rotRodIzq = 0.0f;
            rotRodDer = 0.0f;
            rotBraDerS = 0.0f;
            rotBraIzqS = 0.0f;
            recorrido1 = false;
            engine->play2D("media/puertaAbre.wav",
                false, false, true);
            movPuerta = true;
```

Fig 20. Recorrido de Timmy (primera parte)

```
if (movPuerta)
{

    if (abrirPuerta)
    {
        rotPuerta -= 20.0f * deltaTime;
        if (rotPuerta < -90)
        {
            abrirPuerta = false;
            recorrido2 = true;
        }
    }
}
if (recorrido2)
{
    rotRodDerS = 60.0f;
    rotRodDer = -30.0f;
    posY += 0.008f;
    posZ -= 0.008f;
    printf("Pos y = %f", posY);
    if (posY > 0.3f)
    {
        rotRodDer = 0.0f;
        rotRodDerS = 0.0f;
        rotRodIzqS = 60.0f;
        rotRodIzq = -30.0f;
        posZ -= 0.008f;
        posY += 0.008f;
        printf("pos y = %f", posY);
        if (posY > 0.75)
        {
            rotRodIzqS = 0.0f;
            rotRodIzq = 0.0f;
            rotRodIzqS = 0.0f;
            rotRodIzq = 0.0f;
            posZ -= 0.008f;
            printf("pos z = %f", posZ);
            if (posZ < -0.5f)
            {
                recorrido2 = false;
                recorrido3 = true;
            }
        }
    }
}
if (recorrido3)
{
    posZ -= 0.009;
    if (auxMovTimmy)
    {
        rotRodDerS += 0.4f;
        rotRodIzqS -= 0.4f;
        rotBraDerS -= 0.4f;
        rotBraIzqS += 0.4f;
        if (rotRodDerS > 20)
        {
            auxMovTimmy = false;

        }
    }
    if (!auxMovTimmy)
    {
        //movTimmyZ -+ 0.006f;
        rotRodDerS -= 0.4f;
        rotRodIzqS += 0.4f;
        rotBraDerS += 0.4f;
        rotBraIzqS -= 0.4f;
        if (rotRodDerS < -20)
        {
            auxMovTimmy = true;
        }
    }
    if (posZ < -11.7f)
    {
        rotRodDerS = 0.0f;
        rotRodIzqS = 0.0f;
        rotBraDerS = 0.0f;
        rotBraIzqS = 0.0f;
        posY = 0.3f;
        recorrido3 = false;
        EspectaculoLuces = true;
        //cerrarPuerta = true;
        recorrido4 = true;
    }
}
if(recorrido4)
{
    posZ -= 0.009;
    if (auxMovTimmy)
    {
        rotRodDerS += 0.4f;
        rotRodIzqS -= 0.4f;
        rotBraDerS -= 0.4f;
        rotRodIzqS -= 0.4f;
        rotBraDerS -= 0.4f;
        rotBraIzqS += 0.4f;
        if (rotRodDerS > 20)
        {
            auxMovTimmy = false;

        }
    }
    if (!auxMovTimmy)
    {
        rotRodDerS -= 0.4f;
        rotRodIzqS += 0.4f;
        rotBraDerS += 0.4f;
        rotBraIzqS -= 0.4f;
        if (rotRodDerS < -20)
        {
            auxMovTimmy = true;
        }
    }
    if (posZ < -17.7f)
    {
        rotRodDerS = 0.0f;
        rotRodIzqS = 0.0f;
        rotBraDerS = 0.0f;
        rotBraIzqS = 0.0f;
```

Fig 21. Animación Timmy (segunda parte)

```
                rotBrazq3 = 0.0f;
                recorrido4 = false;
                engine->play2D("media/puertaCierra.wav",
                    false, false, true);
                cerrarPuerta = true;
            }
        }
        if (cerrarPuerta)
        {
            rotPuerta += 20.0f * deltaTime;
            if (rotPuerta > 0)
            {
                cerrarPuerta = false;
                recorrido = false;
            }
        }
    }
```

Fig 22. Animación Timmy (tercera parte)

- Espectáculo de luces

Ya mencionado anteriormente, con ayuda de las SpotLights, se observa un espectáculo de luces al interior de la casa de los Turner.

```
if ((EspectaculoLuces) || (mainWindow.getespectaculoLuces()))
{
    if (luces)
    {
        if (dirLightx < 0.5)
        {
            dirLightx += 0.01f;
            dirLightFx -= 0.01f;
            dirLightFz -= 0.01f;
        }
        else
        {
            luces = false;
            luces2 = true;
        }
    }
    if (luces2)
    {
        if (dirLightx > -0.5)
        {
            dirLightx -= 0.01f;
            dirLightFx += 0.01f;
            dirLightFz += 0.01f;
        }
        else
        {
            luces2 = false;
            luces = true;
        }
    }
}
```

Fig 23. Espectáculo de luces

F. VISTAS

Se consideran 4 vistas: la aérea, la frontal, la del cuarto de Timmy Turner y finalmente la sala de la casa.

Primero se inicializan las vistas:

```
camera = Camera(glm::vec3(-2.0f, 1.0f, 20.0f), glm::vec3(0.0f, 1.0f, 0.0f), -90.0f, 0.0f, 10.0f, 0.5f);
cameraVistaCuarto = Camera(glm::vec3(7.0f, 5.5f, -13.5f), glm::vec3(0.0f, 1.0f, 0.0f), 180.0f, 0.0f, 5.0f, 0.5f);
cameraVistaAerea = Camera(glm::vec3(0.0f, 55.0f, -15.0f), glm::vec3(0.0f, 1.0f, 0.0f), -90.0f, -90.0f, 5.0f, 0.5f);
cameraVistaSala = Camera(glm::vec3(4.0f, -0.6f, -9.0f), glm::vec3(0.0f, 1.0f, 0.0f), -90.0f, 0.0f, 10.0f, 0.5f);
```

Fig 24. Inicialización de vistas

```
camera.keyControl(mainWindow.getsKeys(), deltaTime);
camera.mouseControl(mainWindow.getXChange(), mainWindow.getYChange());
cameraVistaCuarto.keyControl(mainWindow.getsKeys(), deltaTime);
cameraVistaAerea.keyControl(mainWindow.getsKeys(), deltaTime);
cameraVistaSala.keyControl(mainWindow.getsKeys(), deltaTime);
```

Fig 25. Movimiento de cámara por teclado

```
if (mainWindow.getVista() || (vista3persona))
{
    glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
    glUniform3f(uniformEyePosition, camera.getCameraPosition().x, camera.getCameraPosition().y,
        camera.getCameraPosition().z);
}
if (mainWindow.getvistaAerea())
{
    glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(cameraVistaAerea.calculateViewMatrix()));
    glUniform3f(uniformEyePosition, cameraVistaAerea.getCameraPosition().x, cameraVistaAerea.getCameraPosition().y,
        cameraVistaAerea.getCameraPosition().z);
}
if (mainWindow.getvistaCuarto())
{
    glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(cameraVistaCuarto.calculateViewMatrix()));
    glUniform3f(uniformEyePosition, cameraVistaCuarto.getCameraPosition().x, cameraVistaCuarto.getCameraPosition().y,
        cameraVistaCuarto.getCameraPosition().z);
    glm::vec3 posTimmy(7.5f, 5.5f, -13.5f);
}
if (mainWindow.getvistaSala())
{
    glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(cameraVistaSala.calculateViewMatrix()));
    glUniform3f(uniformEyePosition, cameraVistaSala.getCameraPosition().x, cameraVistaSala.getCameraPosition().y,
        cameraVistaSala.getCameraPosition().z);
}
```

Fig 26. Asignación de letras de teclado para cambio de vista

G. AUDIO

Se consideraron distintos sonidos de audio, uno que funciona como música de fondo y sonidos para complementar algunas animaciones y sonidos de ambiente, el abrir y cerrar de una puerta.

- Música

```
engine->play2D("media/Cancion.wav", true, false, true);
```

Fig 27.Línea de código para música de fondo

- Sonidos complementarios

Para la animación de magia

```
if (mainWindow.getmagia())
{
    if (crecer < 0.09)
    {
        crecer += 0.02f * deltaTime;
        movy  += 0.15f * deltaTime;
        movx -= 0.5f * deltaTime;
        while (i < 2)  {
            i ++;
            engine->play2D("media/bell.wav", false, false, true);
            printf("i = %i", i);
            break;
        }
    }
    else
    {
        magia = false;
    }
```

Fig 28. Sonido animación magia

Para la animación de la caja de música

```
if (mainWindow.getRegaloAnim() )
{
    if (Regalo)
    {
        rotTapa += 6.5f * deltaTime;
        movArbol += 0.1f * deltaTime;
        while (i < 4) {
            engine->play2D("media/caja_musica.wav", false, false, true);
            printf("i = %i", i);
            i++;
            break;
        }
        if (movArbol > 1.15)
        {
            Regalo = false;
        }
    }
    rotArbol += 4.0f * deltaTime;
}
```

Fig 29. Sonido animación Caja musical

- Sonidos de ambiente

```
if (mainWindow.getmovPuertaF())
{
    if (abrirPuerta)
    {
        while (i < 2) {
            i++;
            engine->play2D("media/puertaAbre.wav", false, false, true);
            printf("i = %i", i);
            break;
        }

        rotPuerta -= 20.0f * deltaTime;
        if (rotPuerta < -90)
        {
            abrirPuerta = false;
            cerrarPuerta = true;
            engine->play2D("media/puertaCierra.wav", false, false, true);
        }
    }
    if (cerrarPuerta)
    {

        rotPuerta += 20.0f * deltaTime;
        if (rotPuerta > 0)
        {
            cerrarPuerta = false;
            recorrido = false;
        }
    }
```

Fig 30. Sonido al abrir y cerrar la puerta

V. REFERENCIAS

Página utilizada para descargar algunos objetos : https://www.tinkercad.com/

Repositorio GitHub: https://github.com/DianaAldape/CGEIHC.git