UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE INGENIERÍA ELÉCTRICA

INGENIERÍA EN COMPUTACIÓN

COMPUTACIÓN GRÁFICA e INTERACCIÓN HUMANO COMPUTADORA

Manual Técnico

NOMBRE COMPLETO: Colón Palacios Emmanuel

Roldán Sánchez Alexis

Nº de Cuenta:

317254523

317193301

GRUPO DE TEORÍA: 04

SEMESTRE 2023-1

FECHA DE ENTREGA LÍMITE: 12/01/2023

CALIFICACIÓN: _____

Introducción

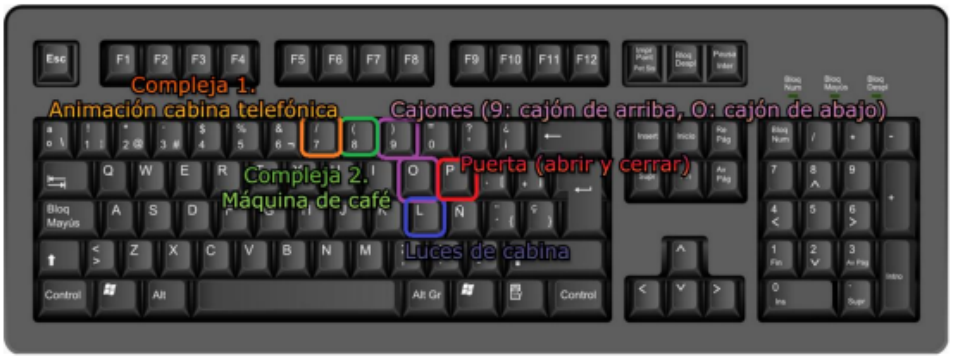
En el proyecto se trata de recrear la temática de Club Penguin y Danny Phantom, al realizarse el proyecto de manera individual hacemos una colaboración en la que parece un choque de multiversos. Es así que para la fachada hacemos uso de la estación telefónica de Club Penguin y la casa de Danny; y para el cuarto, nos ubicamos dentro del laboratorio de sus padres y de la estación telefónica, mismo que dentro del laboratorio y de la estación podremos encontrar distintos elementos para recrear.

Alcance

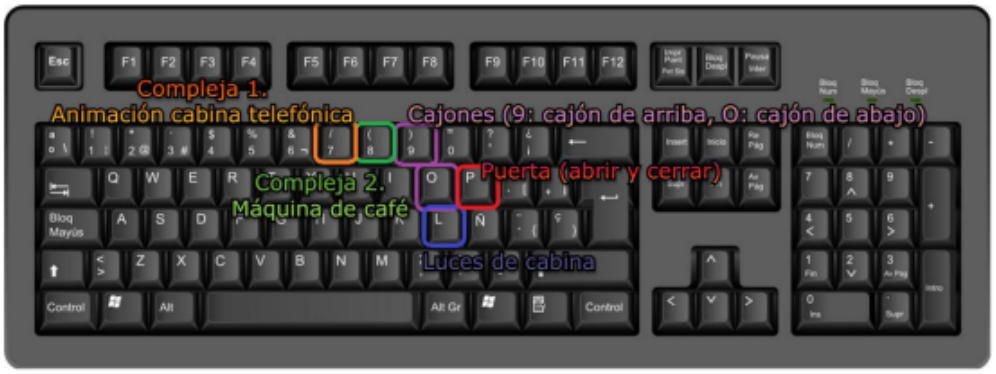
El ambiente para recrear será la fachada e interior de la Estación Pingüi-fónica del popular juego Club Penguin, cabe mencionar que tendrá la fachada del evento “Fiesta Medieval” y no la original, además de que el interior será el último recibido en actualizaciones, siendo recreado con 7 objetos seleccionados previamente y que serán mencionados una vez más en este documento.
Así también, se representará un espacio virtual basado en la casa de Danny Phantom y el laboratorio de sus padres, además de que habrá la representación de varios elementos para hacer una experiencia más inmersiva.
Se deberán añadir animaciones a los objetos, animaciones coherentes con el contexto del espacio representado, uso de iluminación y para poder ambientar más el espacio. Para obtener mayor grado de inmersión se colocó un SkyBox que simula un entorno nevado y un piso que simula estar en la nieve.
Será explorable gracias al manejo de la cámara introducida en el código.

Asignación de teclas

- <u>Animaciones del ambiente/interior de Club Penguin</u>



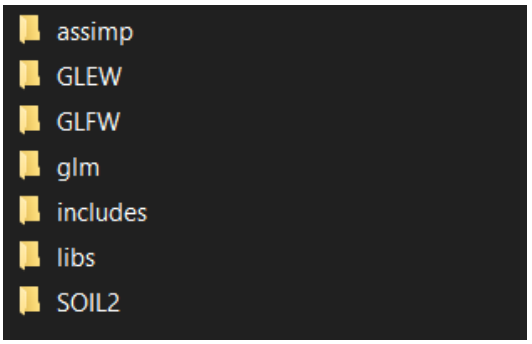
- Movimiento



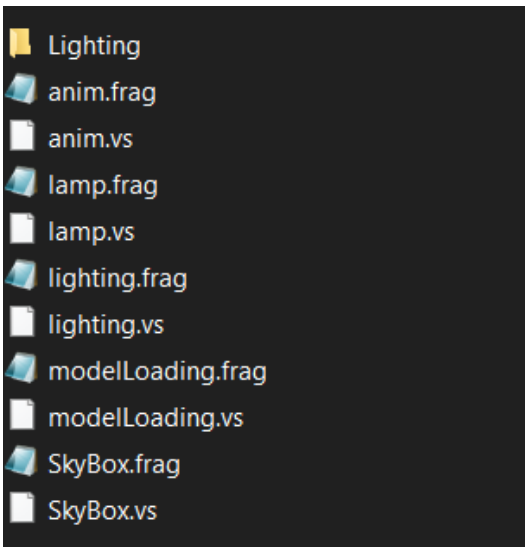
- Animaciones del ambiente/interior de Danny Phantom

Estructura del proyecto

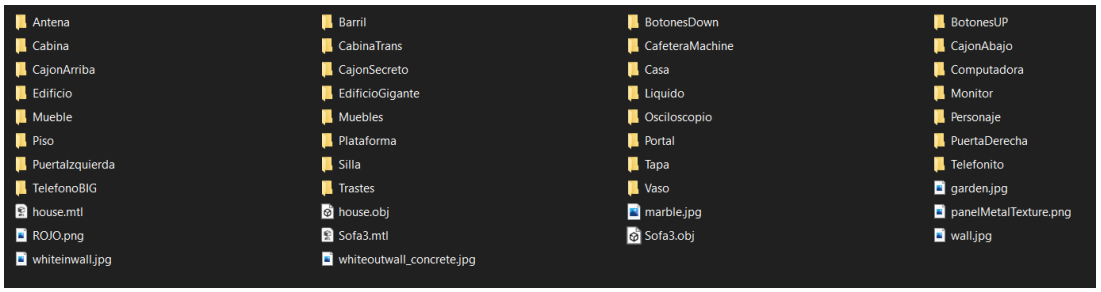
Para poder realizar el proyecto, se hizo uso de varias bibliotecas externas:



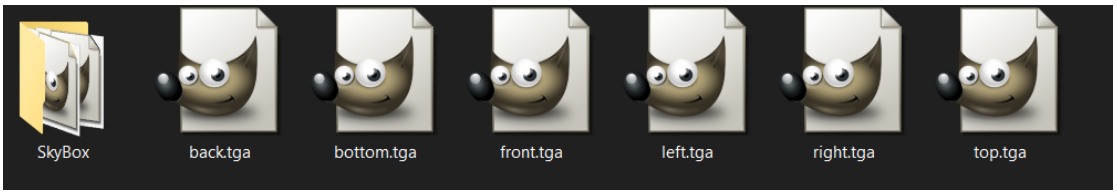
Así mismo, los Shaders usados son los siguientes:



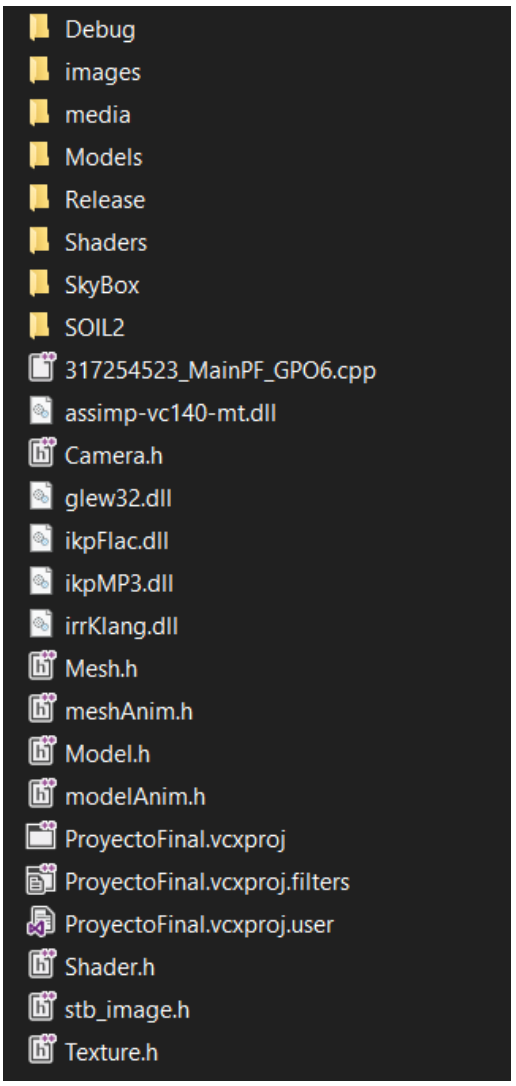
Los modelos usados en el proyecto se encuentran en la carpeta Models, en dónde organizados y separados se encuentran en formato .obj y con sus respectivas texturas:



El SkyBox se encuentra en .tga



Y en el directorio de proyecto final se encuentran los siguientes archivos:



Código

Geometría

Dentro de la función principal (el main) podemos encontrar la importación de modelos que se hace gracias a la biblioteca SOIL 2. Dentro de esta parte se importan también los avatares.

Importación de modelos de Club Penguin:

```
// Pinguino Jetpack modelo //
Model Pinwi((char*)"Models/Pinwin/Pinwin.obj");

// Fachada //
Model Edificio((char*)"Models/EdificioGigante/EdificioGigante.obj");
Model PuertaIzq((char*)"Models/PuertaIzquierda/PuertaIzquierda.obj");
Model PuertaDer((char*)"Models/PuertaDerecha/PuertaDerecha.obj");
Model CajonAbajo((char*)"Models/CajonAbajo/CajonAbajo.obj");
Model CajonArriba((char*)"Models/CajonArriba/CajonArriba.obj");

// 7 Objetos //
Model Cabina((char*)"Models/Cabina/Cabina.obj");
Model CabinaTrans((char*)"Models/CabinaTrans/CabinaTrans.obj"); //Transparente
Model CabinaBotonUP((char*)"Models/BotonesUP/BotonesUP.obj");
Model CabinaBotonDOWN((char*)"Models/BotonesDown/BotonesDown.obj");
Model CajonSecreto((char*)"Models/CajonSecreto/CajonSecreto.obj");
Model CafeteraMachine((char*)"Models/CafeteraMachine/CafeteraMachine.obj");
Model Monitor((char*)"Models/Monitor/Monitor.obj");
Model Mueble((char*)"Models/Mueble/Mueble.obj");
Model Silla((char*)"Models/Silla/Silla.obj");
Model TelefonoFijo((char*)"Models/Telefonito/Telefonito.obj");
Model TelefonoGigante((char*)"Models/TelefonoBIG/TelefonoBIG.obj");


// Modelos extra //
Model Piso((char*)"Models/Piso/Piso.obj");
Model Vaso((char*)"Models/Vaso/Vaso.obj");
Model Tapa((char*)"Models/Tapa/Tapa.obj");
Model Liquido((char*)"Models/Liquido/Liquido.obj");
```

Importación de modelos de Danny Phantom

```
//Importación de modelos OBJ Danny Phantom
//Fachada
Model Casa((char*)"Models/Casa/fachada.obj");
Model Calle((char*)"Models/Casa/calle.obj");
//Objetos
Model Puerta((char*)"Models/Casa/puerta.obj");
Model Barril((char*)"Models/Barril/barril.obj");
Model Computadora((char*)"Models/Computadora/computadora.obj");
Model Locker((char*)"Models/Muebles/locker.obj");
//Parte de las mesas
Model Osciloscopio((char*)"Models/Osciloscopio/osciloscopio.obj");
Model Trastes((char*)"Models/Trastes/trastes.obj");
Model Barra((char*)"Models/Muebles/barra.obj");

//Plattaforma
Model Plataforma((char*)"Models/Plataforma/plataforma.obj");
Model Disco((char*)"Models/Plataforma/discoAnim.obj");

//Modelo portal
Model Portal((char*)"Models/Portal/portalVerde.obj");
Model EstrucPortal((char*)"Models/Portal/portalEstructura.obj");

//Modelo de antena
Model Antena((char*)"Models/Antena/antena.obj");
Model Base((char*)"Models/Antena/base.obj");
Model Rotor((char*)"Models/Antena/rotor.obj");

//Personaje Danny Phantom
Model Danny((char*)"Models/Personaje/DannyPhantom.obj");
Model Fantasma((char*)"Models/DannyFantasma/DannyFantasma.obj");
```

Uso de modelos

Club Penguin

```
// Fachada Casa Pingui//
view = camera.GetViewMatrix();
model = glm::mat4(1);
model = glm::translate(model, glm::vec3(posX, posY, posZ));
model = glm::translate(model, glm::vec3(0.0f, 2.6f, 0));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
glUniform1f(glGetUniformLocation(lightingShader.Program, "activaTransparencia"), 0.0);
Edificio.Draw(lightingShader);

view = camera.GetViewMatrix();
model = glm::mat4(1);
model = glm::translate(model, glm::vec3(posX, posY, posZ));
model = glm::translate(model, glm::vec3(1.7f, 5.5f, 4.3f));
model = glm::rotate(model, glm::radians(rotPuertaD), glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
glUniform1f(glGetUniformLocation(lightingShader.Program, "activaTransparencia"), 0.0);
PuertaDer.Draw(lightingShader);
```

```
view = camera.GetViewMatrix();
model = glm::mat4(1);
model = glm::translate(model, glm::vec3(posX, posY, posZ));
model = glm::translate(model, glm::vec3(-1.7f, 5.5f, 4.3f));
model = glm::rotate(model, glm::radians(rotPuertaI), glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
glUniform1f(glGetUniformLocation(lightingShader.Program, "activaTransparencia"), 0.0);
PuertaIzq.Draw(lightingShader);

view = camera.GetViewMatrix();
model = glm::mat4(1);
model = glm::translate(model, glm::vec3(posX, posY, posZ));
model = glm::translate(model, glm::vec3(0.0f, 2.6f, movCajon));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
glUniform1f(glGetUniformLocation(lightingShader.Program, "activaTransparencia"), 0.0);
CajonArriba.Draw(lightingShader);

view = camera.GetViewMatrix();
model = glm::mat4(1);
model = glm::translate(model, glm::vec3(posX, posY, posZ));
model = glm::translate(model, glm::vec3(0.0f, 2.6f, movCajon2));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
glUniform1f(glGetUniformLocation(lightingShader.Program, "activaTransparencia"), 0.0);
CajonAbajo.Draw(lightingShader);

// Modelo pinwi jetpack //
view = camera.GetViewMatrix();
model = glm::mat4(1);
model = glm::translate(model, glm::vec3(posX, posY, posZ));
model = glm::translate(model, glm::vec3(-2, 2.8, 8.0));
model = glm::rotate(model, glm::radians(40.0f), glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
```

Danny Phantom

```
//Fachada casa de Danny Phantom

view = camera.GetViewMatrix();
model = glm::mat4(1);
model = glm::translate(model, glm::vec3(posX, posY, posZ));
model = glm::translate(model, glm::vec3(0, 3, 0));
model = glm::translate(model, glm::vec3(0.0f, 0.0F, -17.6f));
model = glm::rotate(model, glm::radians(90.0f), glm::vec3(0.0f, 1.0f, 0.0));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Casa.Draw(lightingShader);

//Calle
view = camera.GetViewMatrix();
model = glm::mat4(1);
model = glm::translate(model, glm::vec3(posX, posY, posZ));
model = glm::translate(model, glm::vec3(0, 3, 0));
model = glm::translate(model, glm::vec3(0.0f, 0.0F, -17.6f));
model = glm::rotate(model, glm::radians(90.0f), glm::vec3(0.0f, 1.0f, 0.0));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Calle.Draw(lightingShader);

//Puerta
view = camera.GetViewMatrix();
model = glm::mat4(1);
model = glm::translate(model, glm::vec3(posX, posY, posZ));
model = glm::translate(model, glm::vec3(0, 3, 0));
model = glm::translate(model, glm::vec3(0.0f, 0.0F, -17.6f));
model = glm::rotate(model, glm::radians(90.0f), glm::vec3(0.0f, 1.0f, 0.0));
model = glm::translate(model, glm::vec3(7.87f, 0.0f, 0.7f));
model = glm::rotate(model, glm::radians(rotPuerta), glm::vec3(0.0f, 1.0f, 0.0));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Puerta.Draw(lightingShader);


//Modelos DannyPhantom
//      //Antena
view = camera.GetViewMatrix();
model = glm::mat4(1);
model = glm::translate(model, glm::vec3(posX, posY, posZ));
model = glm::translate(model, glm::vec3(-3.9f, 11.75f, 0.0f));
model = glm::translate(model, glm::vec3(0, 1.9, 0));
model = glm::translate(model, glm::vec3(0.0f, 0.0F, -17.6f));
model = glm::rotate(model, glm::radians(90.0f), glm::vec3(0.0f, 1.0f, 0.0));
model = glm::rotate(model, glm::radians(rotRot), glm::vec3(0.0f, 1.0f, 0.0));
model = glm::rotate(model, glm::radians(antenaRot), glm::vec3(1.0f, 0.0f, 1.0));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Antena.Draw(lightingShader);

view = camera.GetViewMatrix();
model = glm::mat4(1);
model = glm::translate(model, glm::vec3(posX, posY, posZ));
model = glm::translate(model, glm::vec3(0, 1.9, 0));
model = glm::translate(model, glm::vec3(0.0f, 0.0F, -17.6f));
model = glm::rotate(model, glm::radians(90.0f), glm::vec3(0.0f, 1.0f, 0.0));
model = glm::translate(model, glm::vec3(0.0f, 1.5f, -3.0f));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
Base.Draw(lightingShader);

view = camera.GetViewMatrix();
model = glm::mat4(1);
model = glm::translate(model, glm::vec3(posX, posY, posZ));
model = glm::translate(model, glm::vec3(0, 1.9, 0));
model = glm::translate(model, glm::vec3(0.0f, 0.0F, -17.6f));
model = glm::rotate(model, glm::radians(90.0f), glm::vec3(0.0f, 1.0f, 0.0));
model = glm::translate(model, glm::vec3(0.0f, 1.5f, -3.0f));
```

Luces

Declaración de atributos iniciales de luces

```
// 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);
```

Definimos las posiciones de los pointlights y declaramos todos los que utilizaremos

```
// Positions of the point lights
glm::vec3 pointLightPositions[] = {
    glm::vec3(posX,posY + 11.4,posZ - 5.3),
    glm::vec3(posX + 2.0 ,posY + 6.1, posZ - 4.5),
    glm::vec3(posX,posY + 11.4,posZ + 2.0),
    glm::vec3(0,0,0),
    //Luces del portal :/
    glm::vec3(posX - 4.1,posY + 8.5, posZ - 9.35),
    glm::vec3(posX + 2.3, posY + 8.5, posZ - 9.35),

    //Luces Danny
    glm::vec3(posX,posY + 26.0,posZ - 15.3),
    glm::vec3(posX,posY + 12.0,posZ - 26.0)
};

//Declaramos luces
glm::vec3 LightP1;
glm::vec3 LightP2;
glm::vec3 LightP3;
//interior Danny
glm::vec3 LightP4;
glm::vec3 LightP7;

//Emergencia Danny
glm::vec3 LightP5;
glm::vec3 LightP6;
```

Parámetros de la luz direccional usada

```
// Directional light
glUniform3f(glGetUniformLocation(lightingShader.Program, "dirLight.direction"), -0.2f, -1.0f, -0.3f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "dirLight.ambient"), 0.3f, 0.3f, 0.3f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "dirLight.diffuse"), 0.3f, 0.3f, 0.3f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "dirLight.specular"), 0.5f, 0.5f, 0.5f);
```

Parámetros de las distintas pointLights casa Club Penguin

```
// 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"), 0.05f, 0.05f, 0.05f);
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);

// Point light 2
glm::vec3 lightColor2;
lightColor2.x = abs(0.5 * sin(glfwGetTime() * LightP2.x * 2));
lightColor2.y = abs(0.5 * sin(glfwGetTime() * LightP2.y * 2));
lightColor2.z = 0.5 * sin(glfwGetTime() * LightP2.z * 0.2);

glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[1].position"), pointLightPositions[1].x, pointLightPositions[1].y, pointLightPositions[1].z);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[1].ambient"), lightColor2.x, lightColor2.y, lightColor2.z);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[1].diffuse"), lightColor2.x, lightColor2.y, lightColor2.z);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[1].specular"), LightP2.x, LightP2.y, LightP2.z);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[1].constant"), 1.0f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[1].linear"), 0.045f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[1].quadratic"), 0.075f);

// Point light 3
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[2].position"), pointLightPositions[2].x, pointLightPositions[2].y, pointLightPositions[2].z);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[2].ambient"), 0.05f, 0.05f, 0.05f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[2].diffuse"), LightP3.x, LightP3.y, LightP3.z);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[2].specular"), LightP3.x, LightP3.y, LightP3.z);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[2].constant"), 1.0f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[2].linear"), 0.09f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[2].quadratic"), 0.032f);

// Point light 4
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[3].position"), pointLightPositions[6].x, pointLightPositions[6].y, pointLightPositions[6].z);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[3].ambient"), 0.05f, 0.05f, 0.05f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[3].diffuse"), LightP4.x, LightP4.y, LightP4.z);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[3].specular"), LightP4.x, LightP4.y, LightP4.z);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[3].constant"), 1.0f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[3].linear"), 0.09f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[3].quadratic"), 0.032f);
```

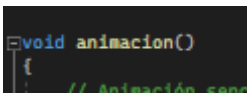
Parámetros de las distintas pointLights casa Danny Phantom

```
//Point light 5
glm::vec3 lightColor5;
lightColor5.x = abs(sin(glfwGetTime() * LightP5.x));
lightColor5.y = abs(sin(glfwGetTime() * LightP5.y));
lightColor5.z = sin(glfwGetTime() * LightP5.z);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[4].position"), pointLightPositions[4].x, pointLightPositions[4].y, pointLightPositions[4].z);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[4].ambient"), lightColor5.x, lightColor5.y, lightColor5.z);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[4].diffuse"), lightColor5.x, lightColor5.y, lightColor5.z);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[4].specular"), LightP5.x, LightP5.y, LightP5.z);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[4].constant"), 1.0f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[4].linear"), 0.009f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[4].quadratic"), 0.032f);

//Luces portal
//Point light 6
glm::vec3 lightColor6;
lightColor6.x = abs(sin(glfwGetTime() * LightP6.x));
lightColor6.y = abs(sin(glfwGetTime() * LightP6.y));
lightColor6.z = sin(glfwGetTime() * LightP6.z);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[5].position"), pointLightPositions[5].x, pointLightPositions[5].y, pointLightPositions[5].z);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[5].ambient"), lightColor6.x, lightColor6.y, lightColor6.z);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[5].diffuse"), lightColor6.x, lightColor6.y, lightColor6.z);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[5].specular"), LightP6.x, LightP6.y, LightP6.z);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[5].constant"), 1.0f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[5].linear"), 0.009f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[5].quadratic"), 0.032f);


// Point light 7
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[6].position"), pointLightPositions[7].x, pointLightPositions[7].y, pointLightPositions[7].z);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[6].ambient"), 0.05f, 0.05f, 0.05f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[6].diffuse"), LightP7.x, LightP7.y, LightP7.z);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[6].specular"), LightP7.x, LightP7.y, LightP7.z);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[6].constant"), 1.0f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[6].linear"), 0.09f);
glUniform1f(glGetUniformLocation(lightingShader.Program, "pointLights[6].quadratic"), 0.032f);
```

En la función animación vienen diferentes animaciones



Por ejemplo, la animación de la puerta de la estación

```
// Animación sencilla 1. Puerta Abrir/Cerrar, activado con tecla P //
if (abierto) {

    if (abrir)
    {
        printf("\nQue se abran las puertas!");
        rotPuertaD -= 0.2;
        rotPuertaI += 0.2;
        if (rotPuertaD < -66.0f)
        {
            abrir = false;
        }

    }

    if (cerrar)
    {
        printf("\nQue se cierren las puertas!");
        rotPuertaD += 0.2;
        rotPuertaI -= 0.2;
        if (rotPuertaD >= 0.0f)
        {
            cerrar = false;

        }
    }

}
```

Animación de cajones

```
if (abiertoCajon) {

    if (abrirCajon)
    {
        movCajon += 0.01;
        if (movCajon >= 0.5f)
        {
            abrirCajon = false;
        }

    }

    if (cerrarCajon)
    {
        movCajon -= 0.01;
        if (movCajon <= 0.02f)
        {
            cerrarCajon = false;

        }
    }

}

if (abiertoCajon2) {

    if (abrirCajon2)
    {
        movCajon2 += 0.01;
        if (movCajon2 >= 0.5f)
        {
            abrirCajon2 = false;
        }

    }

    if (cerrarCajon2)
    {
        movCajon2 -= 0.01;
        if (movCajon2 <= 0.02f)
        {
            cerrarCajon2 = false;

        }
```

Animación del café (Por Keyframes)

```
//Animación compleja 1. Café
if (play)
{
    if (i_curr_steps >= i_max_steps) //end of animation between frames?
    {
        playIndex++;
        if (playIndex > FrameIndex - 2) //end of total animation?
        {
            printf("\nCafe servido!");
            playIndex = 0;
            play = false;
        }
        else //Next frame interpolations
        {
            i_curr_steps = 0; //Reset counter
            //Interpolation
            interpolation();
        }
    }
    else
    {
        //Draw animation

        cafeKF += KeyFrame[playIndex].cafeInc;
        tapaKF += KeyFrame[playIndex].tapaInc;
        tapaVKF += KeyFrame[playIndex].tapaVInc;
        vasoVKF += KeyFrame[playIndex].vasoVInc;
        vasoKF += KeyFrame[playIndex].vasoInc;

        i_curr_steps++;
    }

}
```

Animación para la plataforma y de la rotación de la antena.

```
//Animación plataforma por Keyframes
if (play3)
{
    if (i_curr_steps3 >= i_max_steps3) //end of animation between frames?
    {
        playIndex3++;
        if (playIndex3 > FrameIndex3 - 2)   //end of total animation?
        {
            playIndex3 = 0;
            play3 = false;
        }
        else //Next frame interpolations
        {
            i_curr_steps3 = 0; //Reset counter
            //Interpolation
            interpolation3();
        }
    }
    else
    {
        //Draw animation
        plataRota += KeyFrame3[playIndex3].plataRotaInc;
        traslaRota += KeyFrame3[playIndex3].traslaRota;
        plataScaleX += KeyFrame3[playIndex3].plataScaleX * 0.01;
        plataScaleZ += KeyFrame3[playIndex3].plataScaleZ * 0.01;
        //Antena
        rotRot += KeyFrame3[playIndex3].rotRotInc;
        antenaRot += KeyFrame3[playIndex3].antenaRotInc;

        i_curr_steps3++;
    }
```

Las animaciones de los cajones y las puertas se hacen en la función DoMovement

```
if (keys[GLFW_KEY_U])
{
    activar = true;
}


if (keys[GLFW_KEY_P])
{
    abierto = true;
    if (rotPuertaD < -65.0f) {

        cerrar = true;
        abrir = false;
    }
    else if (rotPuertaD > 0.0f) {
        printf("\nQue se abran las puertas!");
        cerrar = false;
        abrir = true;

    }

}

if (keys[GLFW_KEY_9])
{
    abiertoCajon = true;
    if (movCajon >= 0.5f) {
        printf("\nCajon Cerrado!");
        cerrarCajon = true;
        abrirCajon = false;
    }
    else if (movCajon <= 0.02f) {
        printf("\nCajon Abierto!");
        cerrarCajon = false;
        abrirCajon = true;

    }

}

if (keys[GLFW_KEY_O])
{
    abiertoCajon2 = true;
    if (movCajon2 >= 0.5f) {
        printf("\nCajon Cerrado!");
        cerrarCajon2 = true;
        abrirCajon2 = false;
    }
    else if (movCajon2 <= 0.02f) {
        printf("Cajon Abierto!");
        cerrarCajon2 = false;
        abrirCajon2 = true;

    }
```

Las animaciones de las luces y las que son por KeyFrames se hacen en KeyCallback

```
if (keys[GLFW_KEY_8])
{
    if (play == false && (FrameIndex > 1))
    {
        printf("\nSirviendo café!");
        resetElements();
        //First Interpolation
        interpolation();

        play = true;
        playIndex = 0;
        i_curr_steps = 0;
    }
    else
    {
        printf("\n Cafe servido!");
        play = false;
    }

}

if (keys[GLFW_KEY_7])
{
    if (play2 == false && (FrameIndex2 > 1))
    {
        printf("\nSecuencia secreta activada, desbloqueando cajón secreto!");
        resetElements2();
        //First Interpolation
        interpolation2();

        play2 = true;
        playIndex2 = 0;
        i_curr_steps2 = 0;
    }
    else
    {
        printf("\nCajón mostrado!");
        play2 = false;
    }

}

if (keys[GLFW_KEY_M])
{
    if (play == false && (FrameIndex3 > 1))
    {

        resetElements3();
        //First Interpolation
        interpolation3();
```

Para el uso de la animación por KeyFrames se hace uso de la interpolación, de diferentes funciones y elementos, necesarios para poder cubrir las necesidades de este tipo de animación.

Primero definimos e inicializamos los elementos de los Keyframes, como una estructura y todo lo necesario.

```
#define MAX_FRAMES 9
int i_max_steps3 = 190;
int i_curr_steps3 = 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 rotRodIzq;
    float rotInc;

    // Primera animación //
    float cafeKF = 0.0f, cafeInc = 0.0f;
    float tapaKF = 0.0f, tapaInc = 0.0f;
    float tapaVKF = 0.0f, tapaVInc = 0.0f;
    float vasoVKF, vasoVInc;
    float vasoKF = 0.0f, vasoInc = 0.0f;

    // Segunda animación //
    float botonUKF = 0.0f, botonDKF = 0.0f, cajonSecretoKF = 0.0f;
    float botonUInc = 0.0f, botonDInc = 0.0f, cajonSecretoInc = 0.0f;


    //Animacion plataforma
    float plataRota;
    float plataRotaInc;

    float traslaRota;
    float traslaRotaInc;

    float plataScaleX;
    float plataScaleXInc;

    float plataScaleZ;
    float plataScaleZInc;

    //Animacion antena
    float rotRot;
    float rotRotInc;

    float antenaRot;
    float antenaRotInc;

}FRAME;
```

```
for (int w = 0; w < MAX_FRAMES; w++)
{
    KeyFrame2[w].botonUKF = 0;
    KeyFrame2[w].botonDKF = 0;
    KeyFrame2[w].cajonSecretoKF = 0;
}

KeyFrame2[0].botonUKF = botonUKF;
KeyFrame2[0].botonDKF = botonDKF;
KeyFrame2[0].cajonSecretoKF = cajonSecretoKF;


KeyFrame2[1].botonUKF = 5.185;
KeyFrame2[1].botonDKF = botonDKF;
KeyFrame2[1].cajonSecretoKF = cajonSecretoKF;


KeyFrame2[2].botonUKF = 5.185;
KeyFrame2[2].botonDKF = 5.185;
KeyFrame2[2].cajonSecretoKF = cajonSecretoKF;

KeyFrame2[3].botonUKF = 5.185;
KeyFrame2[3].botonDKF = 5.185;
KeyFrame2[3].cajonSecretoKF = 4.88;


KeyFrame2[4].botonUKF = 5.15;
KeyFrame2[4].botonDKF = 5.15;
KeyFrame2[4].cajonSecretoKF = 4.88;
```

Uno de los elementos importantes para la animación es el reseteo de los frames.

```
void resetElements(void)
{
    cafeKF = KeyFrame[0].cafeKF;
    tapaKF = KeyFrame[0].tapaKF;
    tapaVKF = KeyFrame[0].tapaVKF;
    vasoVKF = KeyFrame[0].vasoVKF;
    vasoKF = KeyFrame[0].vasoKF;

}
```

Además, para el cálculo de los frames intermedios, hacemos uso de la interpolación, la cuál se calcula con las funciones de interpolación.

```
void interpolation(void)
{

    KeyFrame[playIndex].cafeInc = (KeyFrame[playIndex + 1].cafeKF - KeyFrame[playIndex].cafeKF) / i_max_steps;
    KeyFrame[playIndex].tapaInc = (KeyFrame[playIndex + 1].tapaKF - KeyFrame[playIndex].tapaKF) / i_max_steps;
    KeyFrame[playIndex].tapaVInc = (KeyFrame[playIndex + 1].tapaVKF - KeyFrame[playIndex].tapaVKF) / i_max_steps;
    KeyFrame[playIndex].vasoVInc = (KeyFrame[playIndex + 1].vasoVKF - KeyFrame[playIndex].vasoVKF) / i_max_steps;
    KeyFrame[playIndex].vasoInc = (KeyFrame[playIndex + 1].vasoKF - KeyFrame[playIndex].vasoKF) / i_max_steps;
}
```

Conclusiones

Roldán Sánchez Alexis

Considero que el objetivo de este proyecto, el cual era el de aplicar todas los temas y conocimientos adquiridos a lo largo del curso, considero que fue cumplido. Por ejemplo, desde el modelado, la iluminación y animaciones fueron apartados que están dentro del código, aplicados de buena manera y correctamente.

Fue un poco turbulento el desarrollo de este proyecto, debido a que, principalmente, se unieron dos proyectos que fueron trabajados de manera individual por requerimientos de laboratorio. Es así, que incluir y de alguna manera “acoplar” un código a otro trajo consigo muchas fallas y problemas en las que fue necesario la intervención de mucho tiempo. De ahí en fuera, el desarrollo de los diferentes requerimientos se realizó con tiempo y algunas trabas. El manejo del código y funcionamiento de la cámara no fue visto en laboratorio, por lo que trabajar con él fue muy difícil.

Puedo terminar diciendo que fue un proyecto del que fue requerido mucho tiempo y esfuerzo, pero los conocimientos y temas vistos fueron aplicados de buena manera y, con ello, estudiados y arraigados correctamente.

Colón Palacios Emmanuel

El juntar los dos proyectos representó un nuevo desafío a nivel código, incluso ayudó a la resolución de bugs que existían anteriormente gracias a que nos exigió repensar el funcionamiento de librerías y algoritmos dentro del código.
A nivel de precio computacionalmente hablando, noté que nos exigía un poco más de recursos, incluso tuvimos problemas con un modelo que hacía que el código se rompiera desde la ejecución, así que tuvimos algunas limitantes, sin embargo, fueron mínimas a comparación de todo lo agregado, implementado y acondicionado para el ambiente virtual de los dos proyectos.

Conjuntas

Fuimos capaces de aplicar los conocimientos de computación gráfica a nivel práctico y teórico en nuestros respectivos proyectos individuales en el laboratorio de computación gráfica, para posteriormente juntarlos en un mismo ambiente que tuviera una dinámica con características específicas solicitadas por el profesor de teoría. Consideramos que en el curso faltaron ver algunos detalles más específicos, sin embargo y considerando las adversidades que transcurrieron en el semestre, obtuvimos un conocimiento más que aceptable en la asignatura, tanto así que logramos realizar todas las especificaciones que fueron solicitadas al momento del planteamiento del proyecto para la parte de teoría de la materia.