NATIONAL AUTONOMOUS UNIVERSITY OF MEXICO

FACULTY OF ENGINEERING

ELECTRICAL ENGINEERING DIVISION

COMPUTER ENGINEERING

COMPUTER GRAPHICS and HUMAN-COMPUTER INTERACTION

Technical Manual

FULL NAME: Colón Palacios Emmanuel

Roldán Sánchez Alexis

Account Number:

317254523

317193301

THEORY GROUP: 04

SEMESTER 2023-1

DEADLINE DATE: 12/01/2023

QUALIFICATION:________

Introduction

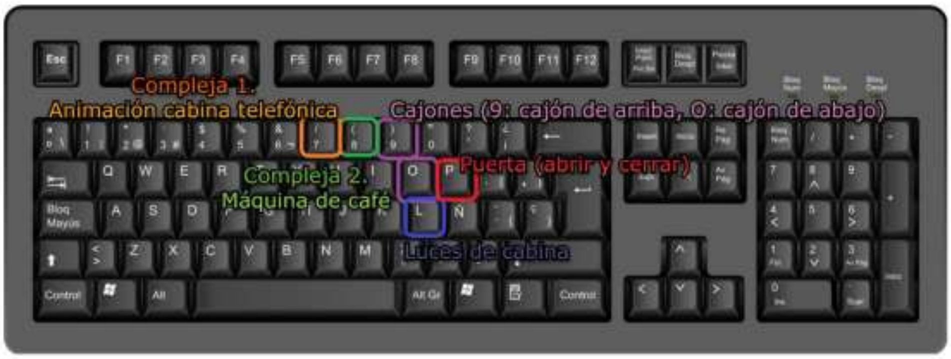
In theproject it is about recreating the theme of Club Penguin and Danny Phantom, when the project is carried out individually we make a collaboration in which it seems a clash of multiverses. Thus, for the façade we make use of the Club Penguin telephone station and Danny's house; and for the fourth, we are located inside the laboratory of his parents and the telephone station, same as inside the laboratory and the station we can find different elements to recreate.

Scope

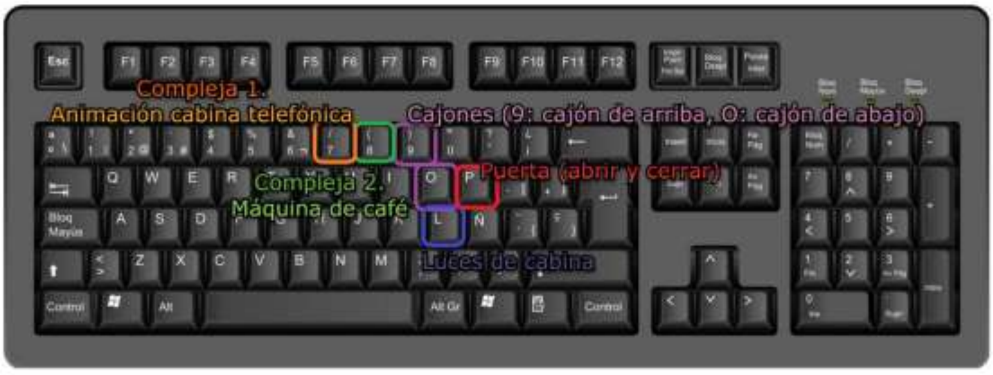
The environment to recreate will be the façade and interior of the Penguin Penguin Station of the popular game Club Penguin, it is worth mentioning that it will have the façade of the event "Medieval Festival" and not the original, in addition to the interior will be the last received in actualizaciones, being recreated with 7 previously selected objects and that will be mentioned once again in this document.
Also, a virtual space based on Danny Phantom's house and his parents' laboratory will be represented, in addition to the representation of several elements to make a more immersive experience.
Animations should be added to the objectives,animations consistent with the context of the space represented, use of lighting and to be able to set the space more. To obtain a greater degree of immersion, a SkyBox was placed that simulates a snowy environment and a floor that simulates being in the snow.
Itwill be explorable thanks to the operation of the camera introduced in the code.

Key assignment

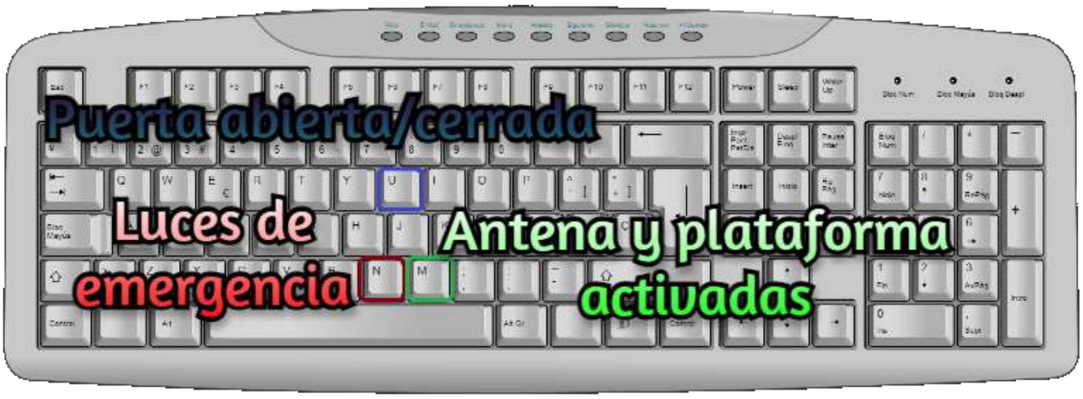
- <u>Club Penguin</u> Ambient/Indoor Animations



- <u>Movement</u>

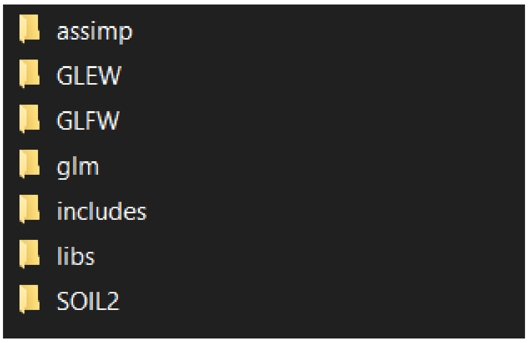


- Ambient/interior animations of Danny Phantom



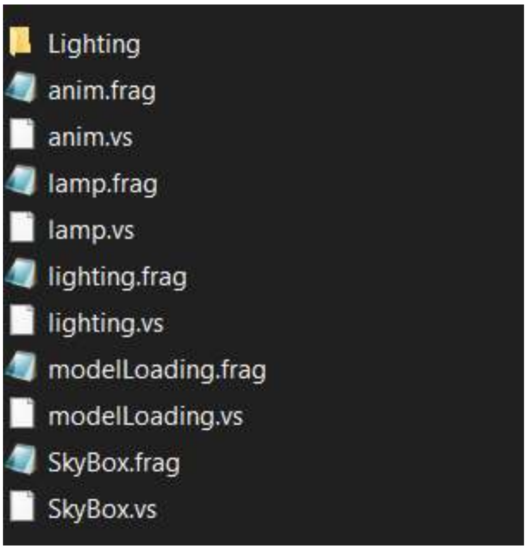
Project structure

In order to carry out the project, several external libraries were used:



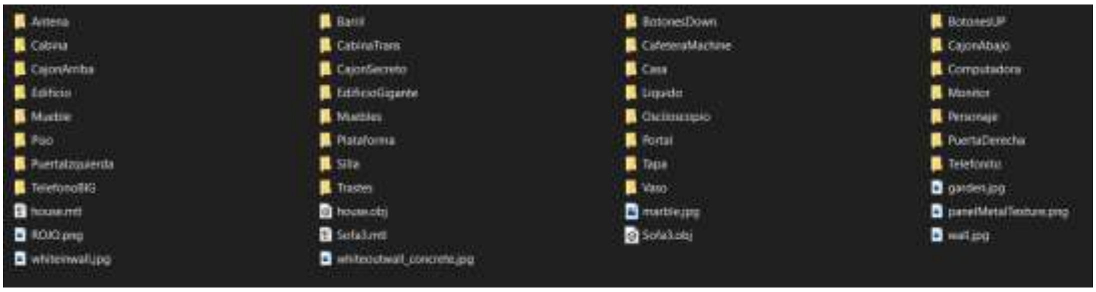
assimp
GLEW
GLFW
glm
includes
libs
SOIL2

Likewise, the Shaders used are the following:



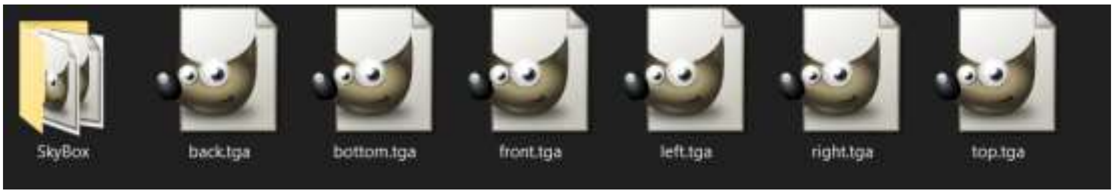
Lighting
anim.frag
anim.vs
lamp.frag
lamp.vs
lighting.frag
lighting.vs
modelLoading.frag
modelLoading.vs
SkyBox.frag
SkyBox.vs

The models used in the project are located in the Models folder, where organized and separated are in . OBJ and with their respective textures:



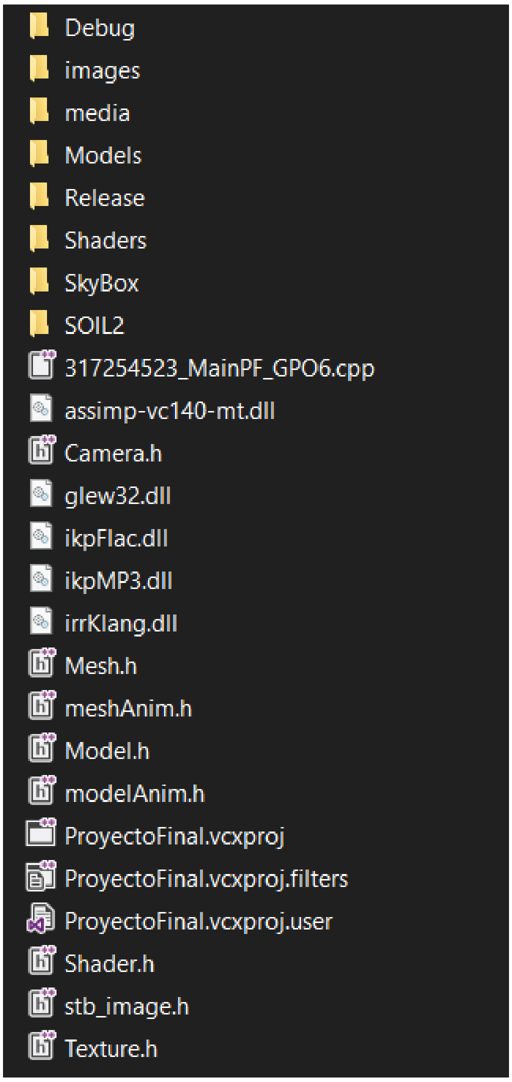
Antena
Barril
BotonesDown
BotonesUP
Cabina
CabinaTrans
CafeteraMachine
CajonAbajo
CajonArriba
CajonSecreto
Casa
Computadora
Edificio
EdificioGigante
Liquido
Monitor
Mueble
Muebles
Osciloscopio
Personaje
Piso
Plataforma
Portal
PuertaDerecha
PuertaIzquierda
Silla
Tapa
Telefonito
TelefonoBIG
Trastes
Vaso
garden.jpg
house.mtl
house.obj
marble.jpg
panelMetalTexture.png
ROJO.png
Sofa3.mtl
Sofa3.obj
wall.jpg
whitemwall.jpg
whiteoutwall_concrete.jpg

The SkyBox is located in . .tga



SkyBox
back.tga
bottom.tga
front.tga
left.tga
right.tga
top.tga

And in the final project directory are the following files:



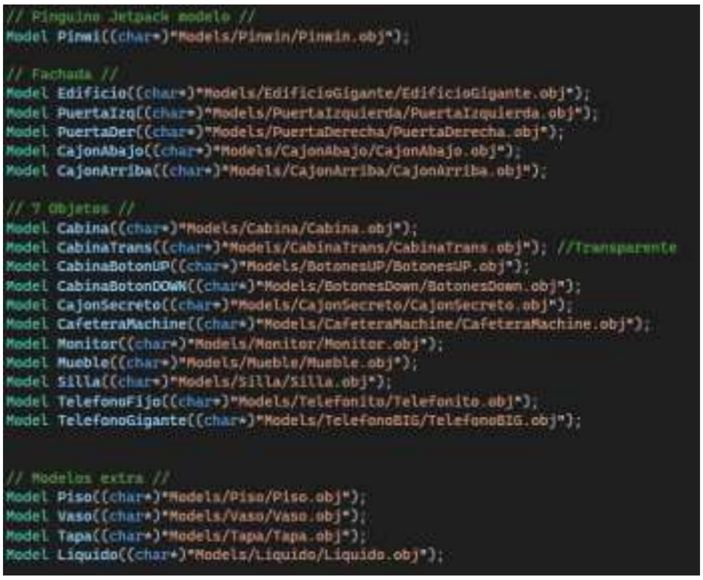
Debug
images
media
Models
Release
Shaders
SkyBox
SOIL2
317254523_MainPF_GPO6.cpp
assimp-vc140-mt.dll
Camera.h
glew32.dll
ikpFlac.dll
ikpMP3.dll
irrKlang.dll
Mesh.h
meshAnim.h
Model.h
modelAnim.h
ProyectoFinal.vcxproj
ProyectoFinal.vcxproj.filters
ProyectoFinal.vcxproj.user
Shader.h
stb_image.h
Texture.h

Code

Geometry

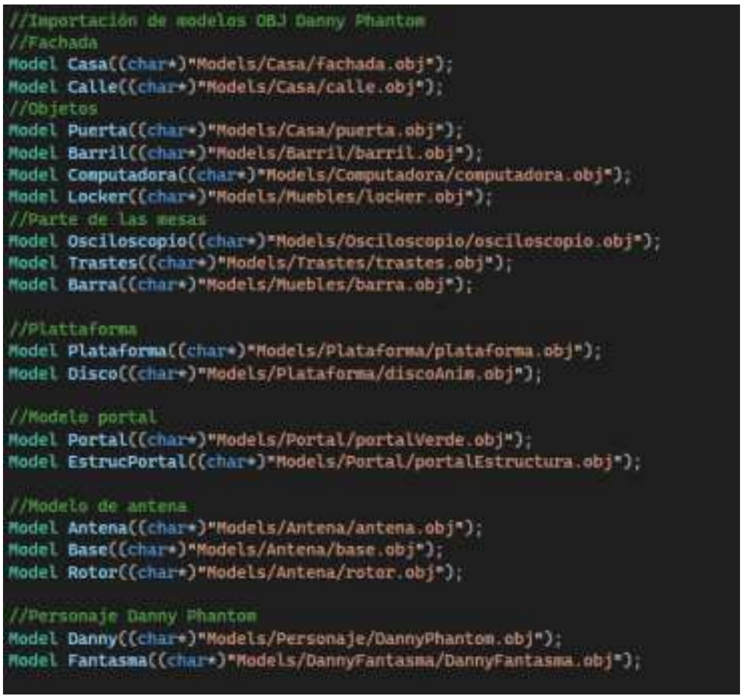
Within the main function (the main) we can find the import of models that is done thanks to the SOIL 2 library. Within this part the avatars are also imported.

Importing Club Penguin models:



```
// 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");
```

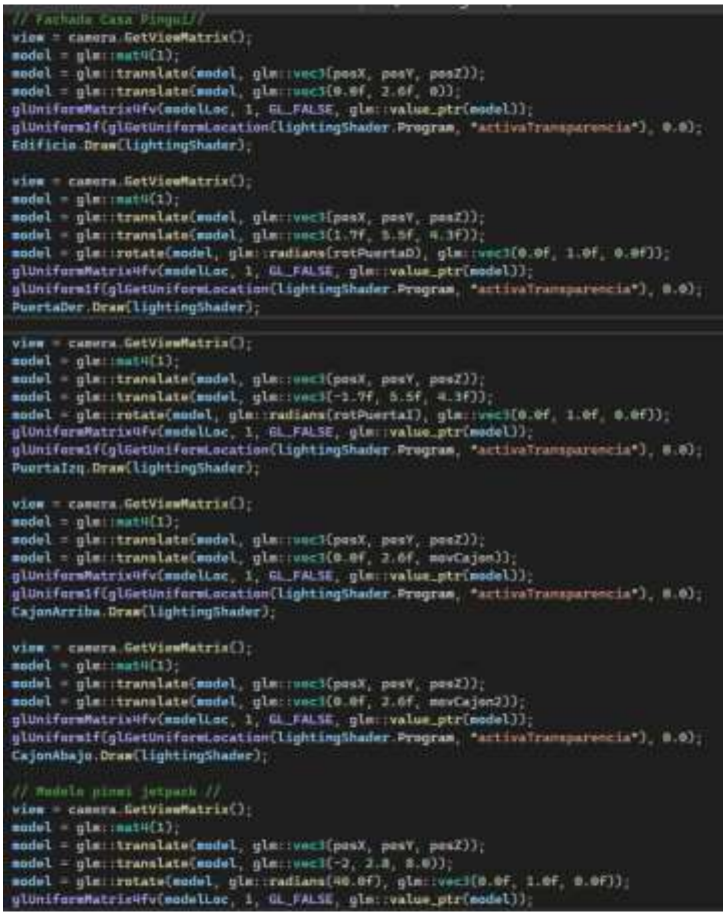
Importing Danny Phantom models



```
//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");
```

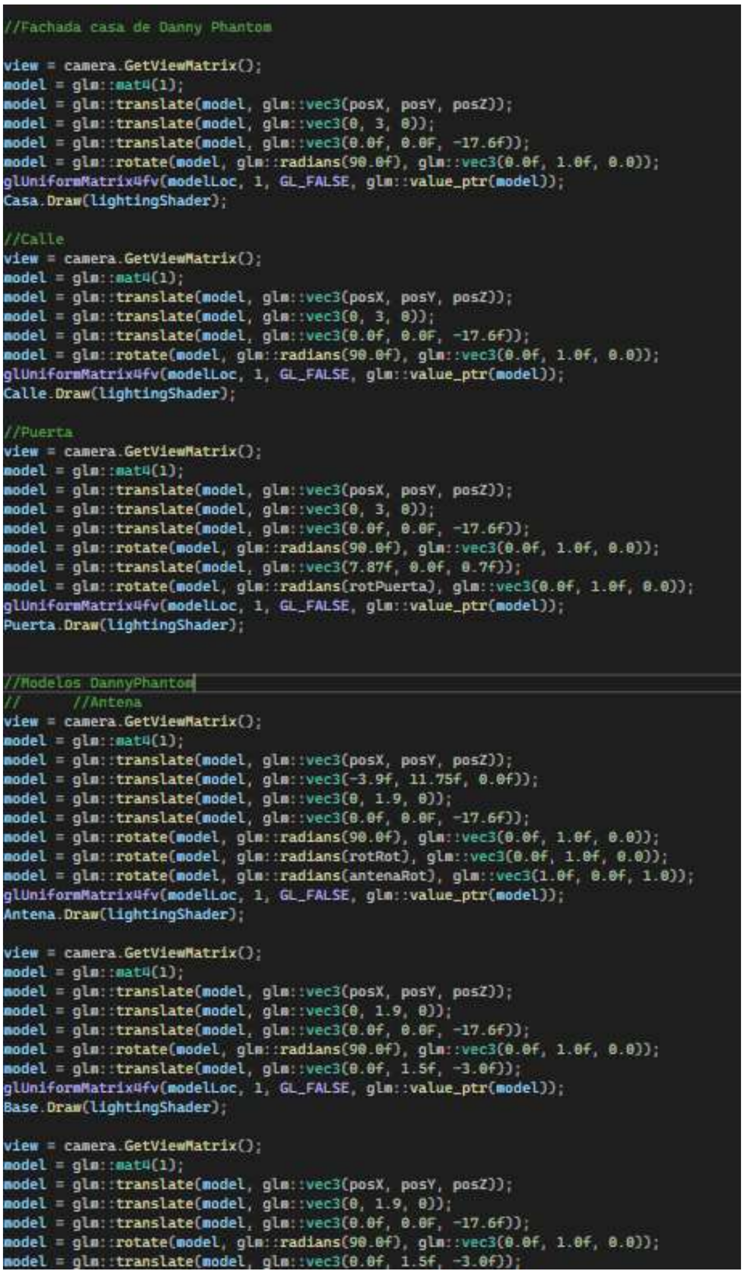
Using models

Penguin Club



```
// 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, 2.0f, 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, 2.0f, 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, 2.0f, movCajon2));
glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
glUniform1f(glGetUniformLocation(lightingShader.Program, "activaTransparencia"), 0.0);
CajonAbajo.Draw(lightingShader);

// Modelo pinai 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));
```

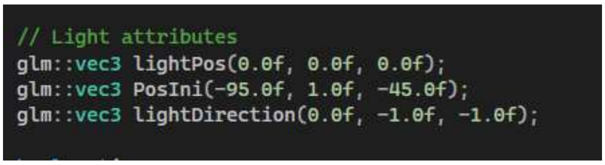
Danny Phantom



```
//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));
```

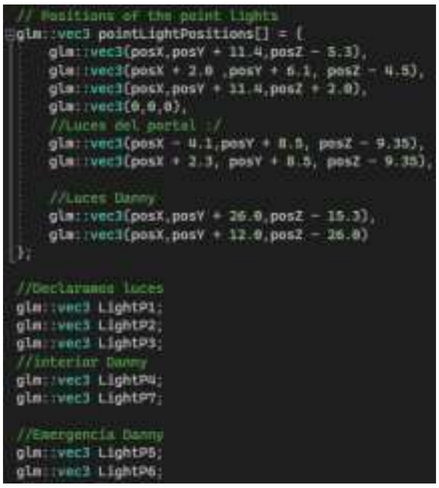
Lights

Declaration of initial attributes of lights



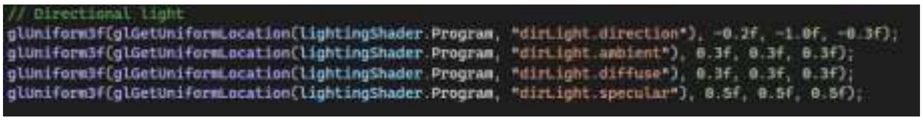
```
// Light attributes
glm::vec3 lightPos(0.0f, 0.0f, 0.0f);
glm::vec3 PosIni(-95.0f, 1.0f, -45.0f);
glm::vec3 lightDirection(0.0f, -1.0f, -1.0f);
```

We define the positions of the pointlights and declare all the ones we will use



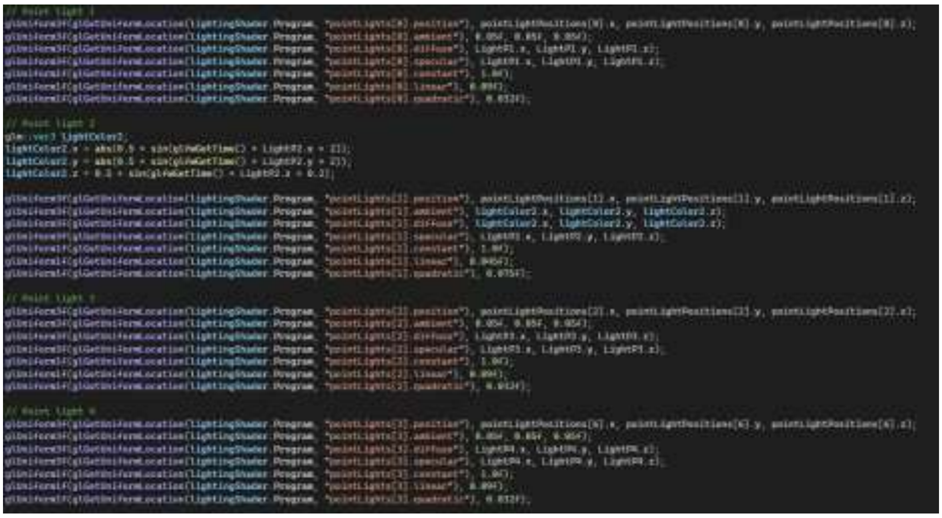
```
// Positions of the point lights
glm::vec3 pointLightPositions[] = {
    glm::vec3(posX,posY + 11.4,posZ - 5.3),
    glm::vec3(posX + 2.0 ,posY + 6.1, posZ - 4.5),
    glm::vec3(posX,posY + 11.4,posZ + 2.0),
    glm::vec3(0,0,0),
    //Luces del portal :/
    glm::vec3(posX - 4.1,posY + 8.5, posZ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

Parameters of the directional light used



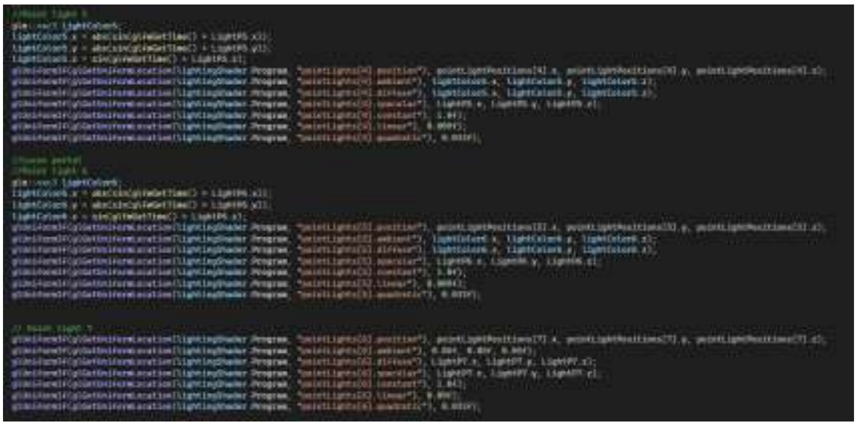
```
// Directional light
glUniform3f(glGetUniformLocation(lightingShader.Program, "dirLight.direction"), -0.2f, -1.0f, -0.3f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "dirLight.ambient"), 0.3f, 0.3f, 0.3f);
glUniform3f(glGetUniformLocation(lightingShader.Program, "dirLight.diffuse"),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

Parameters of the different 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 = 0.3 + 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);
```

Parameters of the different pointLights house Danny Phantom

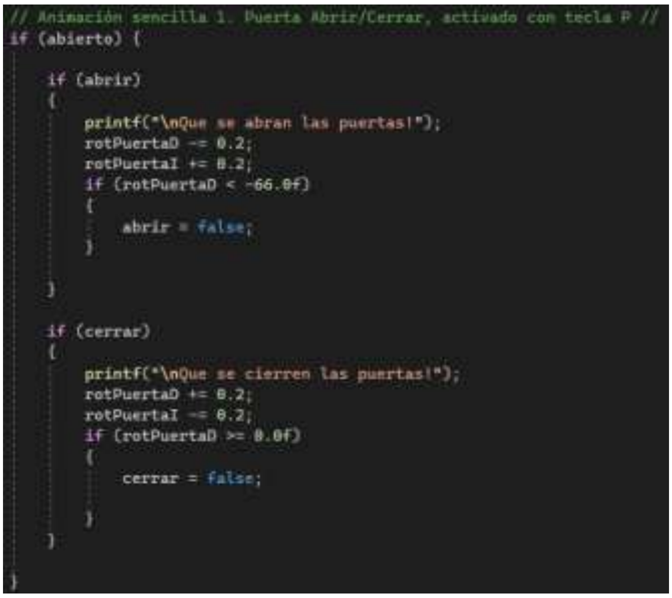


```
//Point light 5
glm::vec3 LightColor5;
LightColor5.x = abs(sin(glfwGetTime() + LightP5.x));
LightColor5.y = abs(sin(glfwGetTime() + LightP5.y));
LightColor5.z = sin(glfwGetTime() + LightP5.z);
glUniform3f(glGetUniformLocation(lightingShader.Program, "pointLights[4].position"), pointLightPositions[4].x, 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"), 0.09f);
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"), 0.09f);
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);
```

In the animation function come different animations

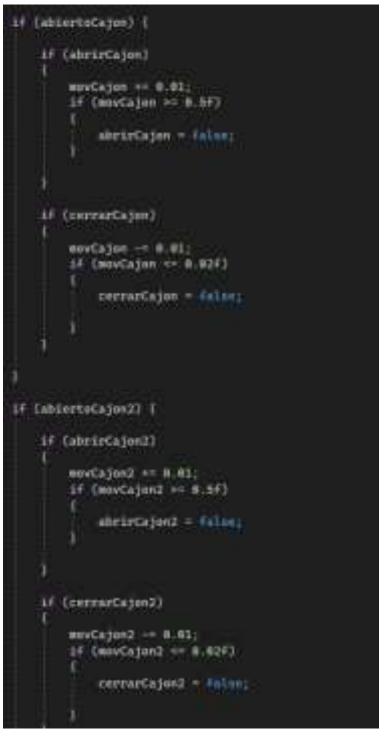
```
void animacion()
{
    // Animación senc
```

For example, the animation of the station door



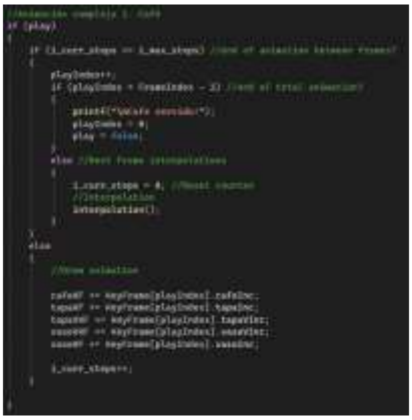
```
// Animación sencilla 1. Puerta Abrir/Cerrar, activado con tecla P //
if (abierto) {

    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;

        }
    }

}
```

Drawer animation



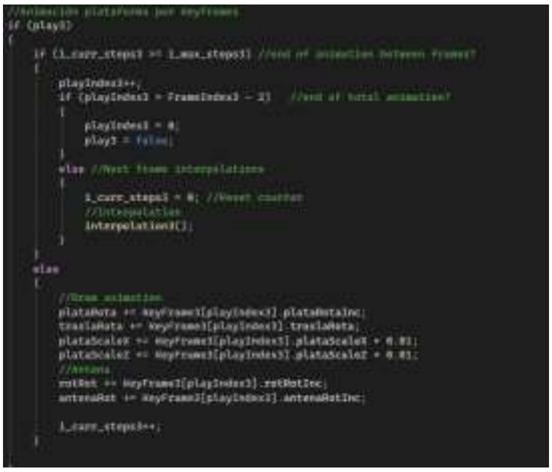
```
if (abiertoCajon) {

    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;

        }
```

Coffee animation (By Keyframes)



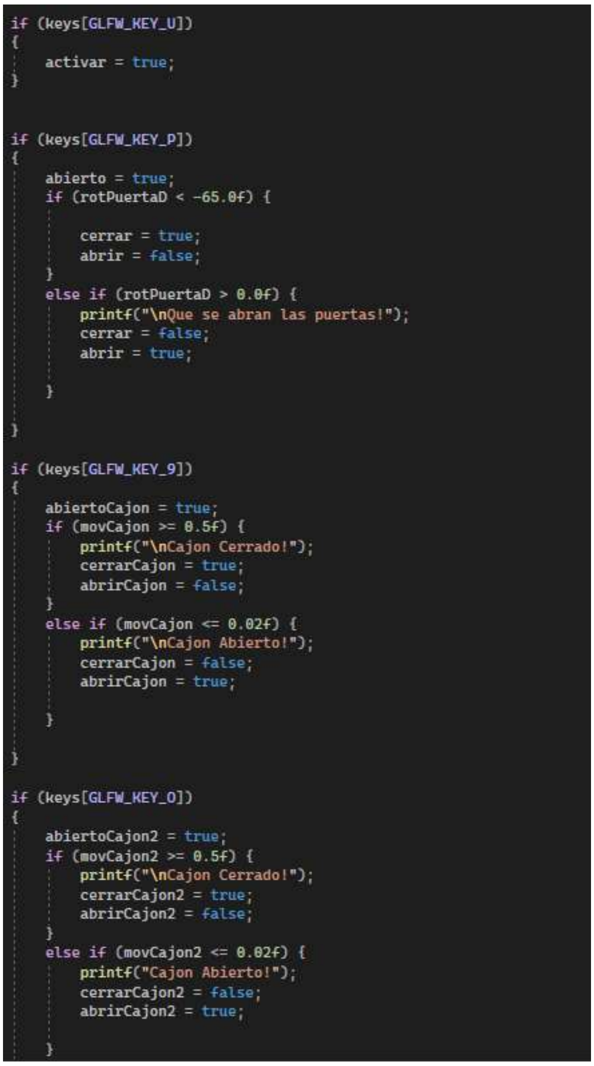
```
//Animación compleja 1: Café
if (play)
{
    if (i_curr_steps >= i_max_steps) //end of animation between frames?
    {
        playIndex++;
        if (playIndex > FrameIndex - 2) //end of total animation?
        {
            printf("\nCafe servido");
            playIndex = 0;
            play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

        cafeAF += KeyFrame[playIndex].cafeInc;
        tapaAF += KeyFrame[playIndex].tapaInc;
        tapaRAF += KeyFrame[playIndex].tapaRInc;
        vasoRAF += KeyFrame[playIndex].vasoRInc;
        vasoAF += KeyFrame[playIndex].vasoInc;

        i_curr_steps++;
    }

}
```

Animation for the platform and antenna rotation.



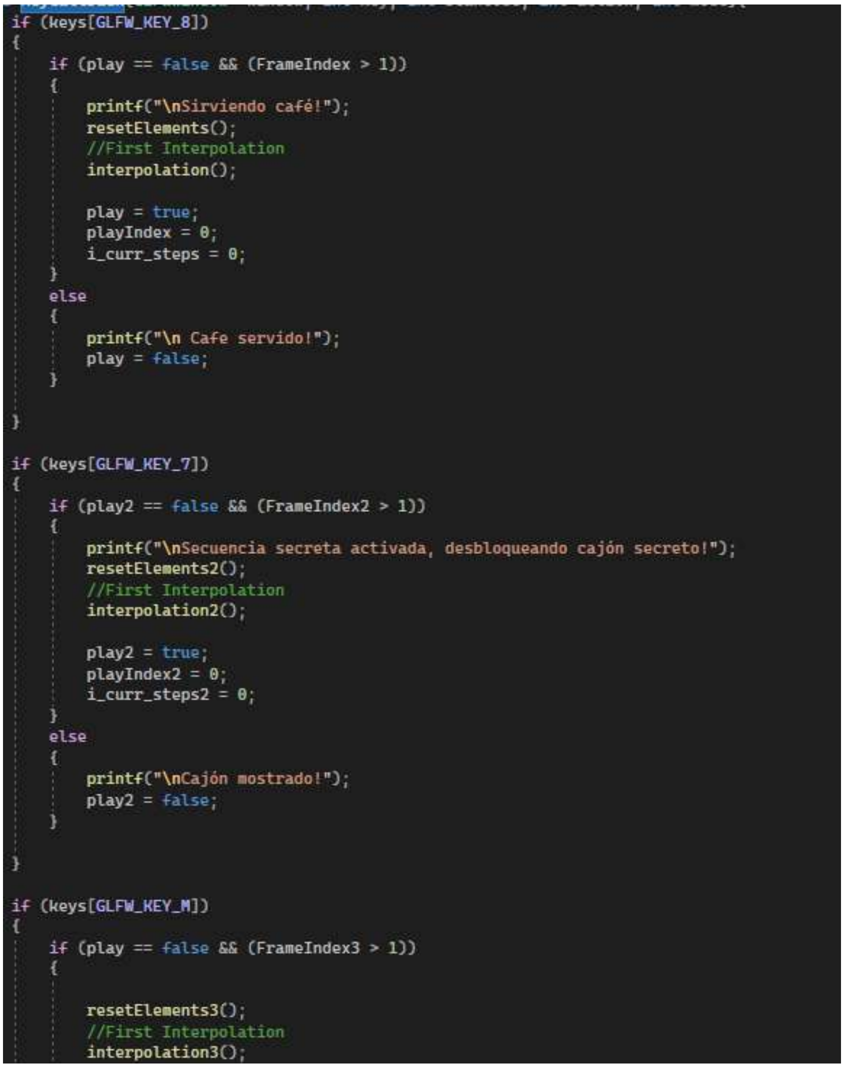
```
//Animación plataforma por KeyFrames
if (play3)
{
    if (i_curr_steps3 >= i_max_steps3) //end of animation between frames?
    {
        playIndex3++;
        if (playIndex3 > FrameIndex3 - 2)   //end of total animation?
        {
            playIndex3 = 0;
            play3 = false;
        }
        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++;
    }
}
```

Drawer and door animations are made in the DoMovement function



```
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;

    }
```

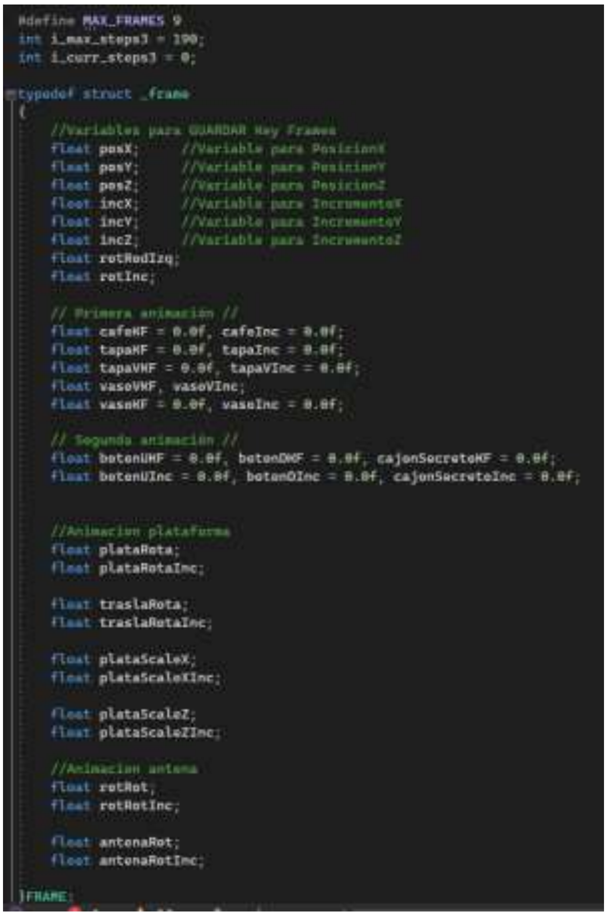
The animations of the lights and those that are by KeyFrames are made in KeyCallback



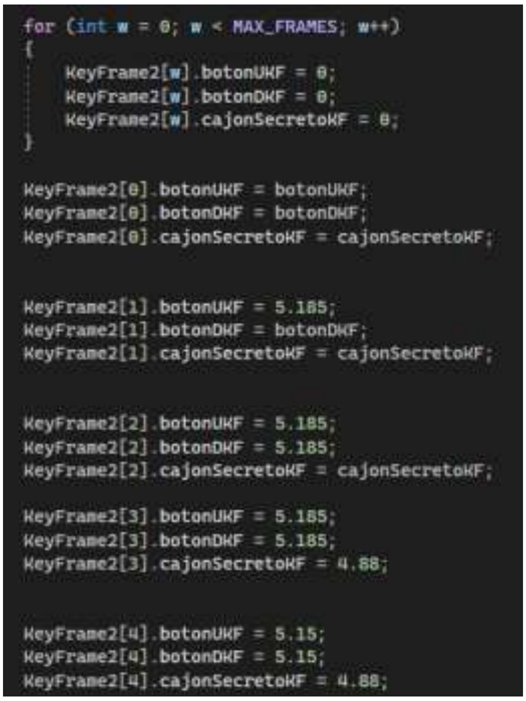
```
if (keys[GLFW_KEY_8])
{
    if (play == false && (FrameIndex > 1))
    {
        printf("\nSirviendo café!");
        resetElements();
        //First Interpolation
        interpolation();

        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();
```

For the use of animation by KeyFrames, interpolation is used, of different functions and elements, necessary to meet the needs of this type of animation.

First we define and initialize the elements of the Keyframes, such as a structure and everything necessary.

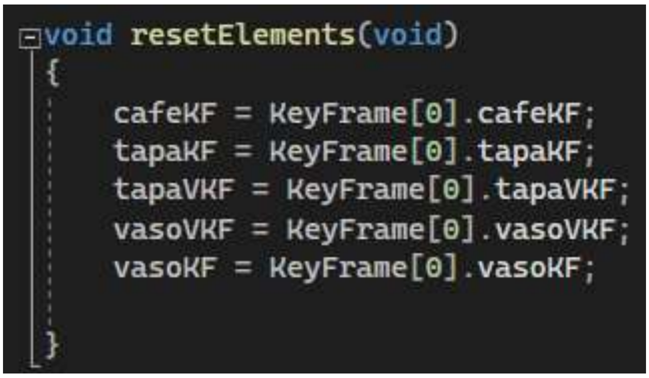


```
#define MAX_FRAMES 9
int i_max_steps3 = 190;
int i_curr_steps3 = 0;

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;
```



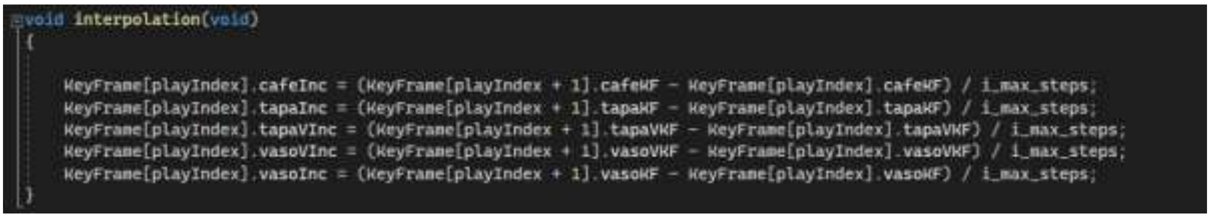
```
for (int w = 0; w < MAX_FRAMES; w++)
{
    KeyFrame2[w].botonUKF = 0;
    KeyFrame2[w].botonDKF = 0;
    KeyFrame2[w].cajonSecretoKF = 0;
}

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;
```

One of the important elements for animation is the reset of theframes.



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

}
```

In addition, for the calculation of intermediate frames, we make use of interpolation, which is calculated with interpolation functions.



```
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;
}
```

Conclusions

Roldán Sánchez Alexis

I consider that the objective of this project, which was to apply all the topics and knowledge acquired throughout the course, I consider that it was fulfilled. For example, from modeling, lighting and animations were removed fromthe code, applied in a good way and correctly.

The development of this project was a bit turbulent, because, mainly, two projects were joined that were worked individually by laboratory requirements. Thus, including and somehow "coupling" one code to another brought with it many failures and problems in which it was necessary to intervene for a long time. From then on, the development of the different requirements was done with time and some obstacles. The management of the code and operation of the camera was not seen in the laboratory, so working with it was very difficult.

I can conclude by saying that it was a project that required a lot of time and effort, but the knowledge and topics seen were applied in a good way and, with it, studied and rooted correctly.

Colon Palacios Emmanuel

Putting the two projects together represented a new challenge at the code level, even helping to solve bugs that previously existed thanks to the fact that it required us to rethink the operation of libraries and algorithms within the code.
At the price level computationally speaking, I noticed that it demanded a little more resources, we even had problems with a model that made the code breakfrom execution, so we had some limitations, however, they were minimal compared to everything added, implemented and conditioned for the virtual environment of the two projects.

Joint

We were able to apply the knowledge ofgraphics at a practical and theoretical level in our respective individual projects in the computer graphics laboratory, to later bring them together in the same environment that had a dynamic with specific characteristics requested by thetheory teacher. We consider that in the course they lacked to see some more specific details, however and considering the adversities that elapsed in the semester, we obtained a more than acceptable knowledge in the subject, so much so that we managed to make all the specifications that were requested at the time of the approach of the project for the part of theory of the matter.