NATIONAL AUTONOMOUS UNIVERSITY OF MEXICO

FACULTY OF ENGINEERING

ELECTRICAL ENGINEERING DIVISION

COMPUTER ENGIENEERING

COMPUTER GRAPHICS and HUMAN COMPUTER INTERACTION

Final Project: Diorama of Everyday Life (Lucario y Frijolito)

DEVELOPMENT MANUAL

Students – Nº Count:

Arroyo Quiroz José Miguel – 317016136

Pérez Quintana Arturo – 317017164

GROUP: 04

SEMESTER 2023-2

DEADLINE DELIVERY: 14 june 2023

SCORE: ____________

TECHNICAL MANUAL

Elements included within the scenario:

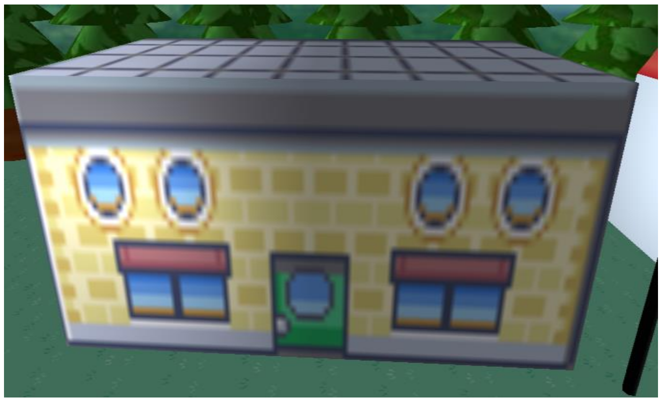
- Geometry

The main structure of the stage is made up of different buildings which are inspired by the models of the 3rd generation Pokémon classic games, these buildings are in such a way that they surround a central area in which the park belonging to the world is located. of one more show

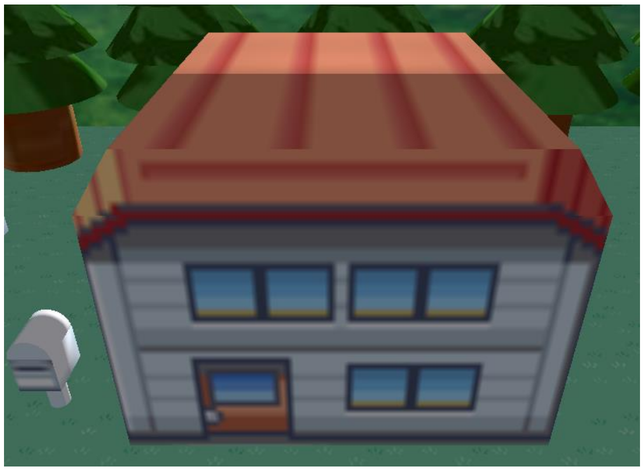
These models were created through the 3D Max 2023 modeling software, later they were exported in obj format, images from the game "Pokémon Leaf Green and Fire Red Edition" were used for their texture.

The different structures and the result already in the virtual environment are presented below:

Oak Laboratory



House

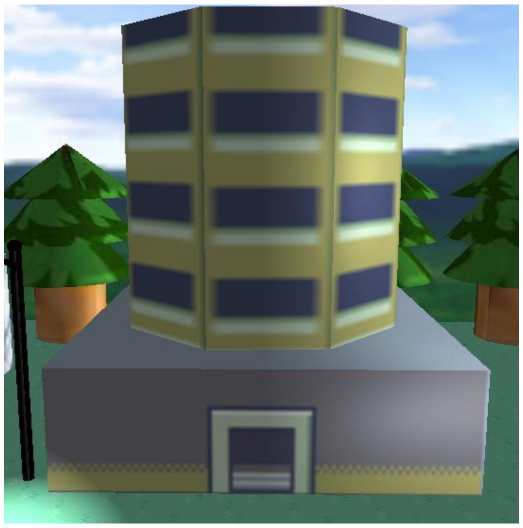


Pokémon Center

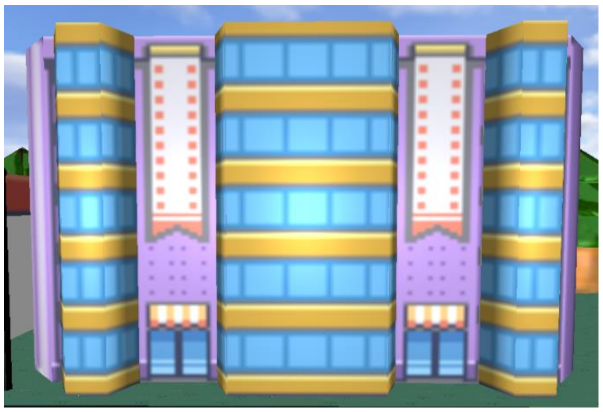
Pokémon Shop

Gym

Lavender Tower



Mall

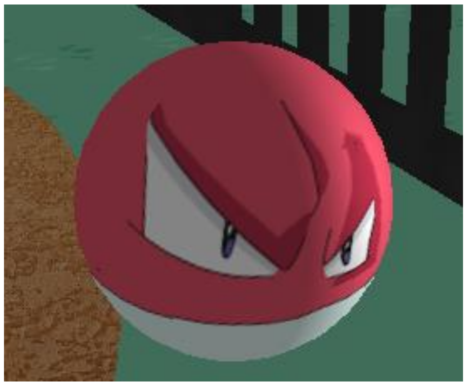


The previous models were exported together under the name "PuebloCentro.obj"

Lucario

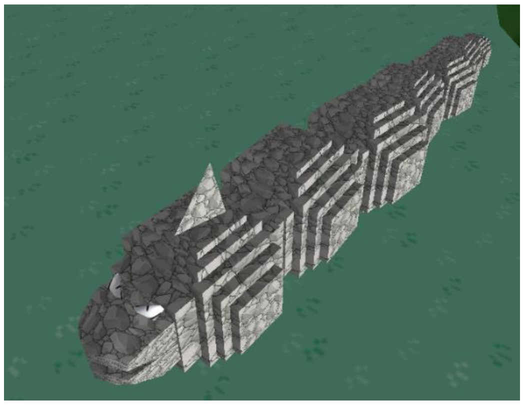
This model was downloaded from the internet, when downloading the model, it includes images to texturize the character, these images were edited and optimized to be used.

Voltorb



Electrode

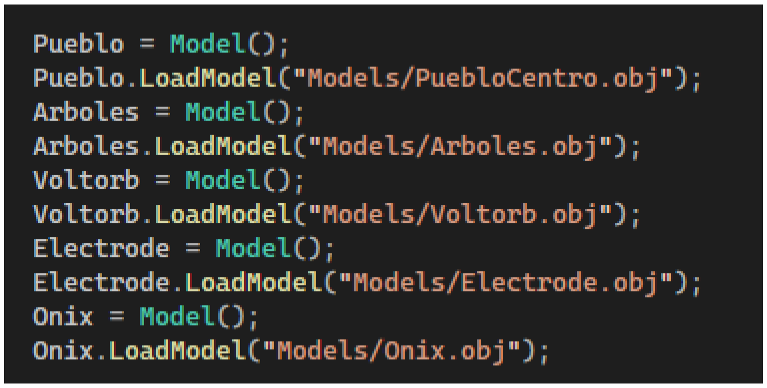
Onix



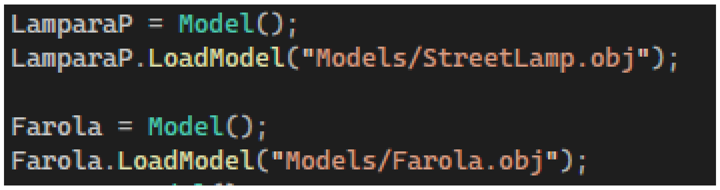
Others: Trees around, lampposts and poles.

Elements that were exported to the environment separately are those that have a special interaction with the environment like lighting or animation.

Loading of the Different models

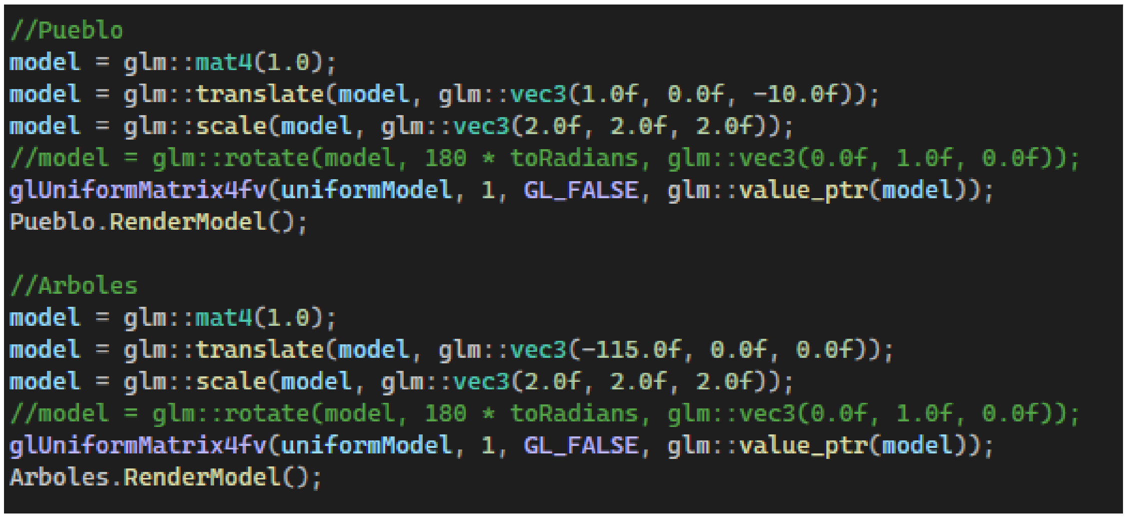


```
Pueblo = Model();
Pueblo.LoadModel("Models/PuebloCentro.obj");
Arboles = Model();
Arboles.LoadModel("Models/Arboles.obj");
Voltorb = Model();
Voltorb.LoadModel("Models/Voltorb.obj");
Electrode = Model();
Electrode.LoadModel("Models/Electrode.obj");
Onix = Model();
Onix.LoadModel("Models/Onix.obj");
```

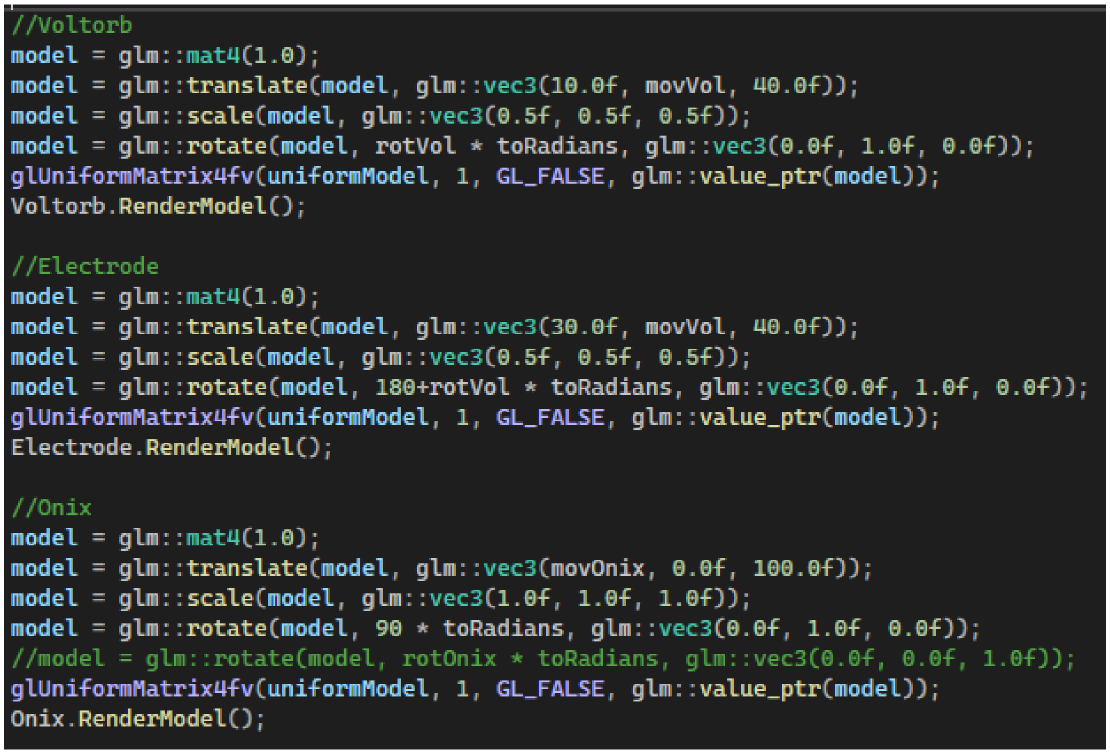


```
LamparaP = Model();
LamparaP.LoadModel("Models/StreetLamp.obj");

Farola = Model();
Farola.LoadModel("Models/Farola.obj");
```

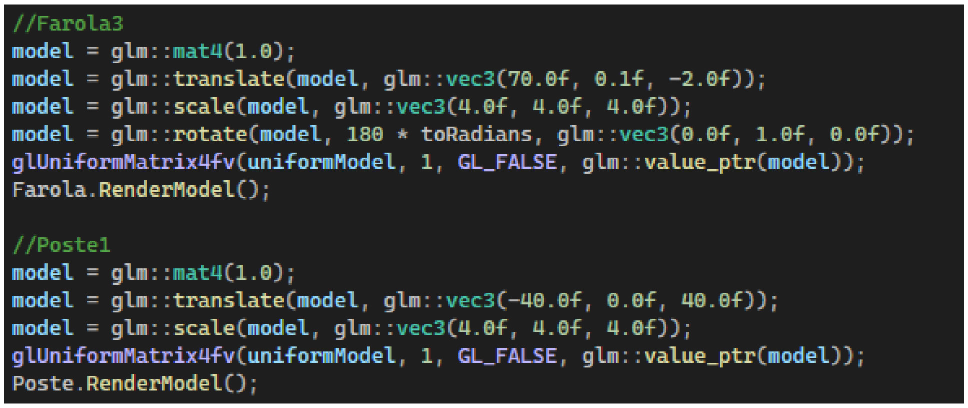
Stage Placement



```
//Pueblo
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(1.0f, 0.0f, -10.0f));
model = glm::scale(model, glm::vec3(2.0f, 2.0f, 2.0f));
//model = glm::rotate(model, 180 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Pueblo.RenderModel();

//Arboles
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-115.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(2.0f, 2.0f, 2.0f));
//model = glm::rotate(model, 180 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Arboles.RenderModel();
```



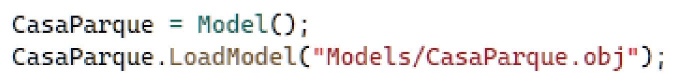
```
//Voltorb
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(10.0f, movVol, 40.0f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
model = glm::rotate(model, rotVol * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Voltorb.RenderModel();

//Electrode
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(30.0f, movVol, 40.0f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
model = glm::rotate(model, 180+rotVol * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Electrode.RenderModel();

//Onix
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(movOnix, 0.0f, 100.0f));
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
model = glm::rotate(model, 90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
//model = glm::rotate(model, rotOnix * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Onix.RenderModel();
```


```
//Farola3
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(70.0f, 0.1f, -2.0f));
model = glm::scale(model, glm::vec3(4.0f, 4.0f, 4.0f));
model = glm::rotate(model, 180 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Farola.RenderModel();

//Poste1
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-40.0f, 0.0f, 40.0f));
model = glm::scale(model, glm::vec3(4.0f, 4.0f, 4.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Poste.RenderModel();
```

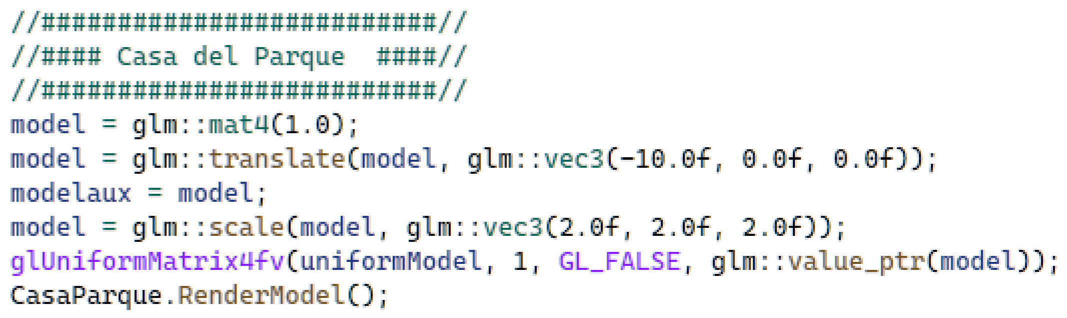
Park House

The main and most striking structure in the park is the house. This model was made with the Blender modeling program and images from the series were used to texturize it, in which the details of the house will be better observed. Subsequently, the corresponding files were created to be placed inside the model's folder, as well as the previously optimized texture.

Inside the code the model is loaded.


```
CasaParque = Model();
CasaParque.LoadModel("Models/CasaParque.obj");
```

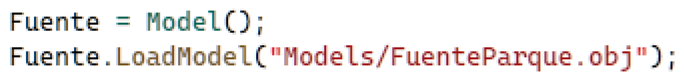
Later we moved and scaled the house to place it at the center of the stage.


```
//###########################//
//#### Casa del Parque  ####//
//###########################//
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-10.0f, 0.0f, 0.0f));
modelaux = model;
model = glm::scale(model, glm::vec3(2.0f, 2.0f, 2.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
CasaParque.RenderModel();
```

Fountain

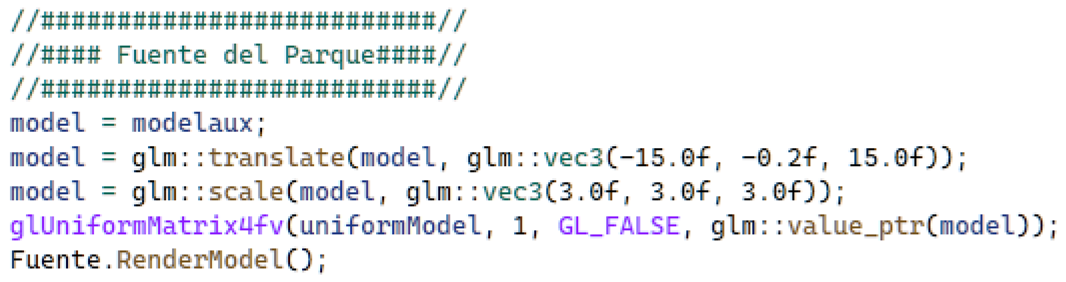
The fountain is a model obtained from the network, but since the texturing of the water did not have it, then it was decided to place one on it. In the same way, the model is loaded into the program and later the translation and scaling transformations are carried out to be more in line with the size of the other objects.

Load the program.



```
Fuente = Model();
Fuente.LoadModel("Models/FuenteParque.obj");
```

Put on stage.

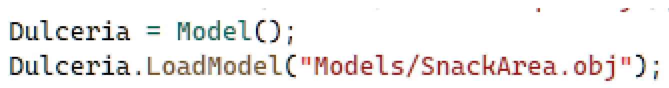


```
//############################//
//#### Fuente del Parque####//
//############################//
model = modelaux;
model = glm::translate(model, glm::vec3(-15.0f, -0.2f, 15.0f));
model = glm::scale(model, glm::vec3(3.0f, 3.0f, 3.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Fuente.RenderModel();
```

Candy store area: Chairs, Tables and Trash Cans

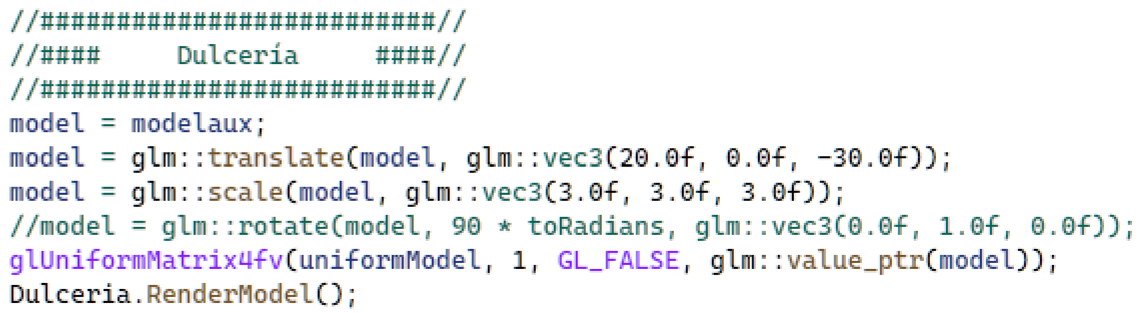
For this area it was decided to obtain models of chairs and tables from the internet. The trash cans and the structure of the store were created manually. With these models, a mini scenario was created with the help of the modeling programs to be able to load the set of objects into the program. All the models were textured manually since both the chair and the table did not have the images.

Area model loading.



```
Dulceria = Model();
Dulceria.LoadModel("Models/SnackArea.obj");
```

View on stage.

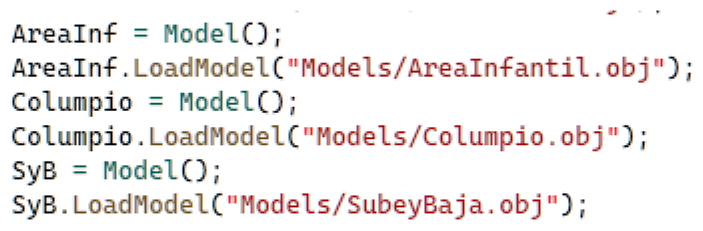


```
//############################//
//####     Dulcería     ####//
//############################//
model = modelaux;
model = glm::translate(model, glm::vec3(20.0f, 0.0f, -30.0f));
model = glm::scale(model, glm::vec3(3.0f, 3.0f, 3.0f));
//model = glm::rotate(model, 90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Dulceria.RenderModel();
```

Children's area: Slide, swing, ramp up and down.

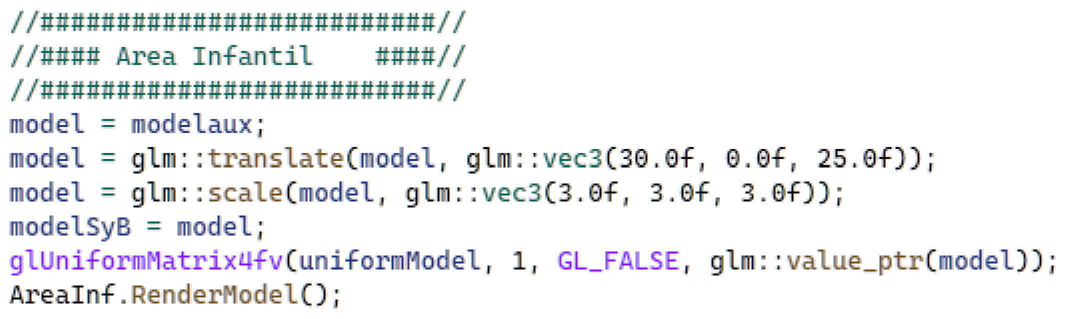
Like the candy store area, a mini stage was also created for the children's area with the game models. The swings and ramps were separated from the base model to be used as an animation object to move independently.

Model loading.

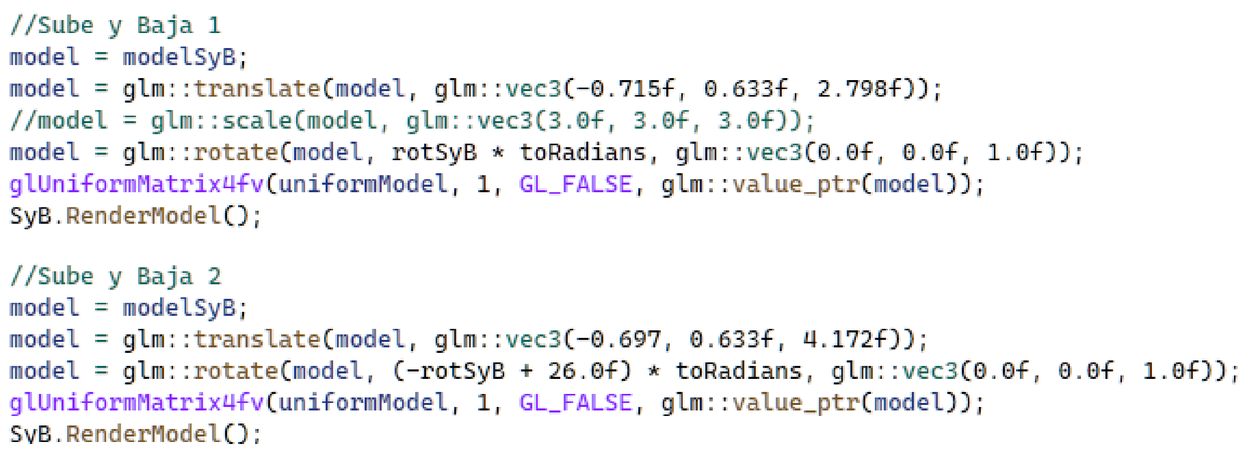


```
AreaInf = Model();
AreaInf.LoadModel("Models/AreaInfantil.obj");
Columpio = Model();
Columpio.LoadModel("Models/Columpio.obj");
SyB = Model();
SyB.LoadModel("Models/SubeyBaja.obj");
```

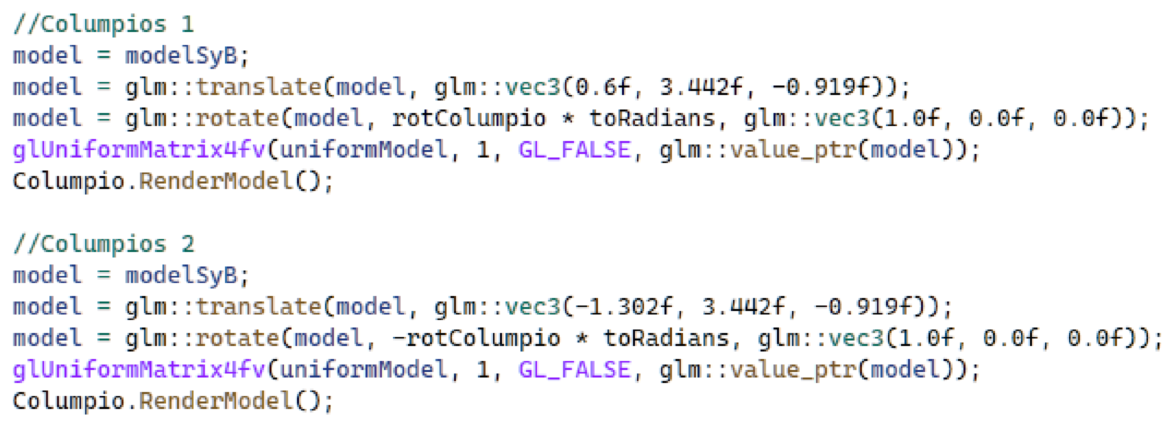
Staging.



```
//############################//
//#### Area Infantil    ####//
//############################//
model = modelaux;
model = glm::translate(model, glm::vec3(30.0f, 0.0f, 25.0f));
model = glm::scale(model, glm::vec3(3.0f, 3.0f, 3.0f));
modelSyB = model;
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
AreaInf.RenderModel();
```



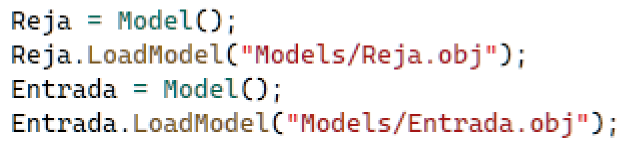
```
//Sube y Baja 1
model = modelSyB;
model = glm::translate(model, glm::vec3(-0.715f, 0.633f, 2.798f));
//model = glm::scale(model, glm::vec3(3.0f, 3.0f, 3.0f));
model = glm::rotate(model, rotSyB * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
SyB.RenderModel();

//Sube y Baja 2
model = modelSyB;
model = glm::translate(model, glm::vec3(-0.697, 0.633f, 4.172f));
model = glm::rotate(model, (-rotSyB + 26.0f) * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
SyB.RenderModel();
```



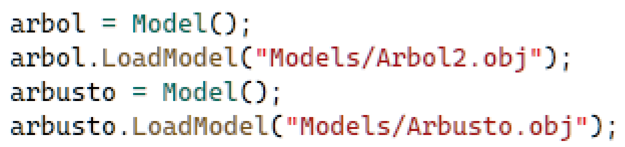
```
//Columpios 1
model = modelSyB;
model = glm::translate(model, glm::vec3(0.6f, 3.442f, -0.919f));
model = glm::rotate(model, rotColumpio * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Columpio.RenderModel();

//Columpios 2
model = modelSyB;
model = glm::translate(model, glm::vec3(-1.302f, 3.442f, -0.919f));
model = glm::rotate(model, -rotColumpio * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Columpio.RenderModel();
```

Others: Fence, entrance, trees and shrubs of the park

In the case of the fence, its objective is to cover the perimeter of the park and they were placed in a hierarchical way to more easily know the position of the last fence, it was also used to place the entrance. The trees and shrubs were only placed inside the park.

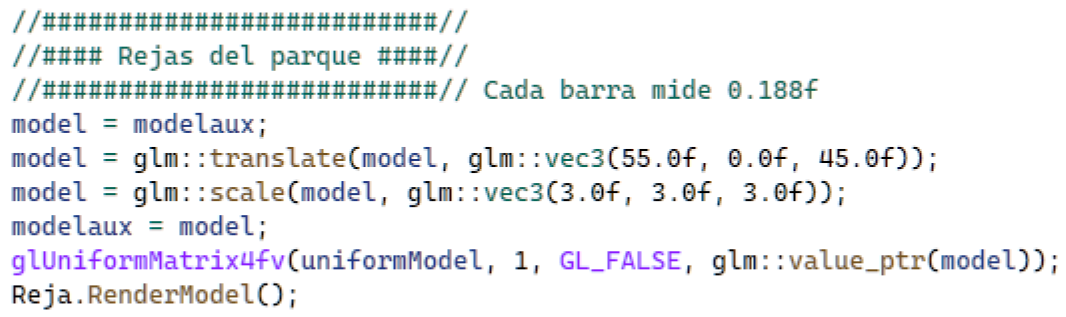


```
Reja = Model();
Reja.LoadModel("Models/Reja.obj");
Entrada = Model();
Entrada.LoadModel("Models/Entrada.obj");
```



```
arbol = Model();
arbol.LoadModel("Models/Arbol2.obj");
arbusto = Model();
arbusto.LoadModel("Models/Arbusto.obj");
```

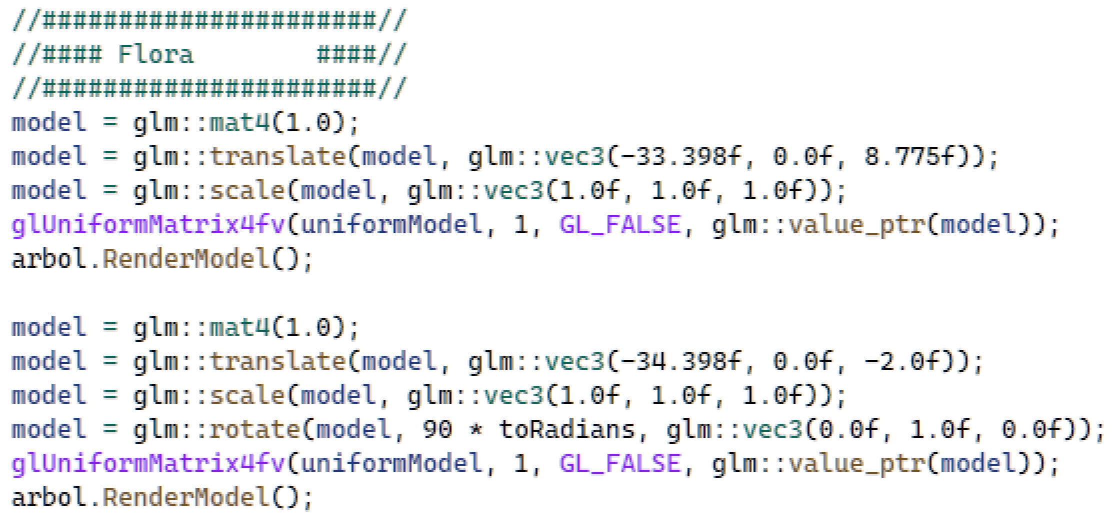
Staging of the models.



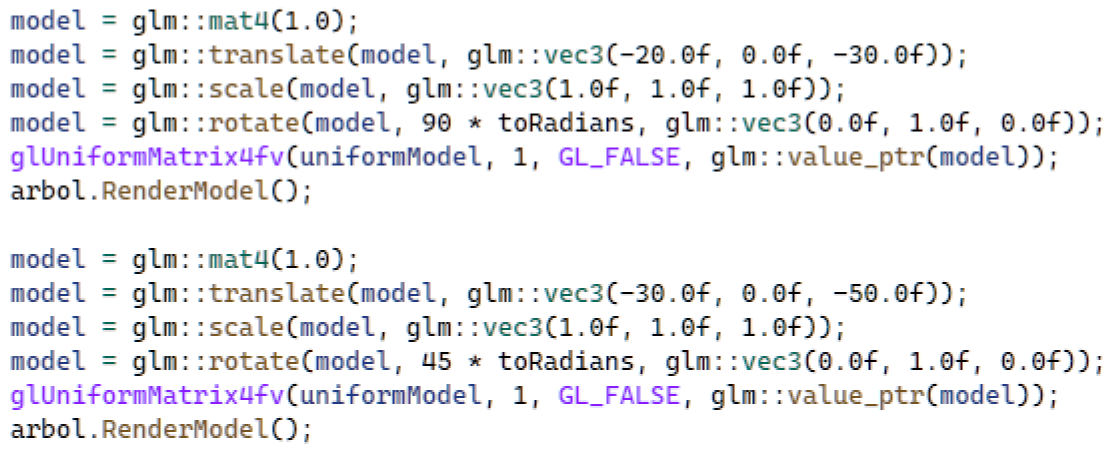
```
//###########################//
//#### Rejas del parque ####//
//###########################// Cada barra mide 0.188f
model = modelaux;
model = glm::translate(model, glm::vec3(55.0f, 0.0f, 45.0f));
model = glm::scale(model, glm::vec3(3.0f, 3.0f, 3.0f));
modelaux = model;
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Reja.RenderModel();
```

```
model = glm::translate(model, glm::vec3(0.0f, 0.0f, -12.0f + 0.188f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Reja.RenderModel();

model = glm::translate(model, glm::vec3(0.0f, 0.0f, -12.0f + 0.188f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Reja.RenderModel();

model = glm::translate(model, glm::vec3(0.0f, 0.0f, -12.0f));
model = glm::rotate(model, 90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Reja.RenderModel();

model = glm::translate(model, glm::vec3(0.0f, 0.0f, -12.0f + 0.188f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Reja.RenderModel();

model = glm::translate(model, glm::vec3(0.0f, 0.0f, -6.0f + 0.376f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Reja.RenderModel();


model = glm::translate(model, glm::vec3(0.0f, 0.0f, -12.0f));
model = glm::rotate(model, 90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Reja.RenderModel();

model = glm::translate(model, glm::vec3(0.0f, 0.0f, -12.0f + 0.188f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Reja.RenderModel();

model = glm::translate(model, glm::vec3(0.0f, 0.0f, -12.0f + 0.188f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Reja.RenderModel();

model = glm::translate(model, glm::vec3(0.0f, 0.0f, -12.0f));
model = glm::rotate(model, 90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Reja.RenderModel();

model = glm::translate(model, glm::vec3(0.0f, 0.0f, -12.0f + 0.188f));
//model = glm::rotate(model, 90* toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Entrada.RenderModel();

//model = glm::rotate(model, -90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::translate(model, glm::vec3(0.0f, 0.0f, -6.0f + 0.376f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Reja.RenderModel();
```

In the case of trees and shrubs, only a few examples will be given.

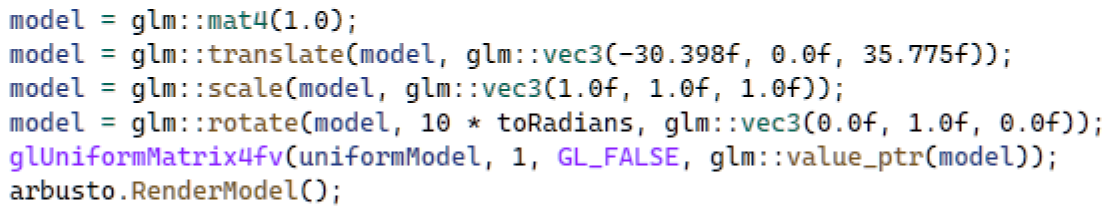


```
//########################//
//#### Flora        ####//
//########################//
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-33.398f, 0.0f, 8.775f));
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
arbol.RenderModel();

model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-34.398f, 0.0f, -2.0f));
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
model = glm::rotate(model, 90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
arbol.RenderModel();
```

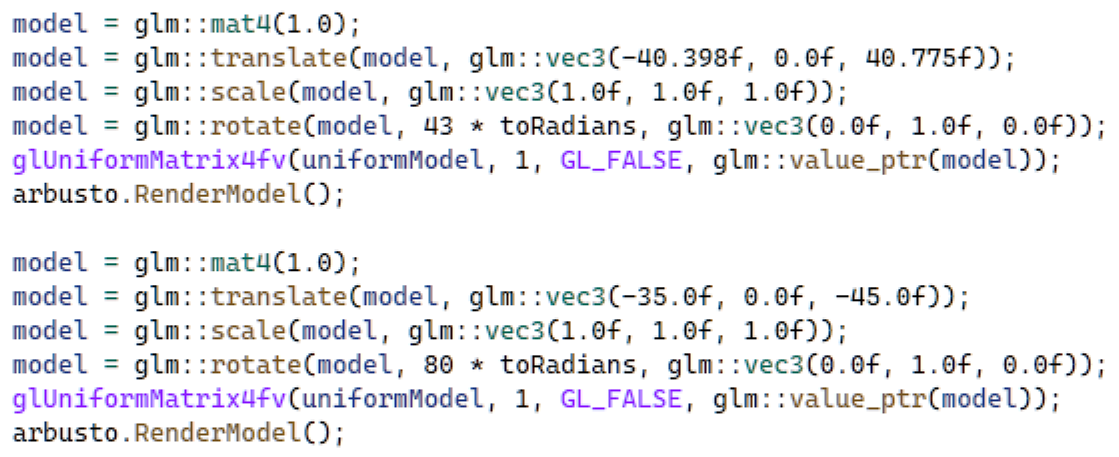


```
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-20.0f, 0.0f, -30.0f));
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
model = glm::rotate(model, 90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
arbol.RenderModel();

model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-30.0f, 0.0f, -50.0f));
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
model = glm::rotate(model, 45 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
arbol.RenderModel();
```

Shrubbery.

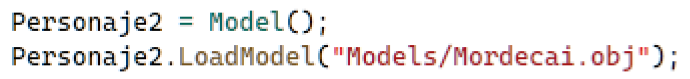


```
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-30.398f, 0.0f, 35.775f));
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
model = glm::rotate(model, 10 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
arbusto.RenderModel();
```

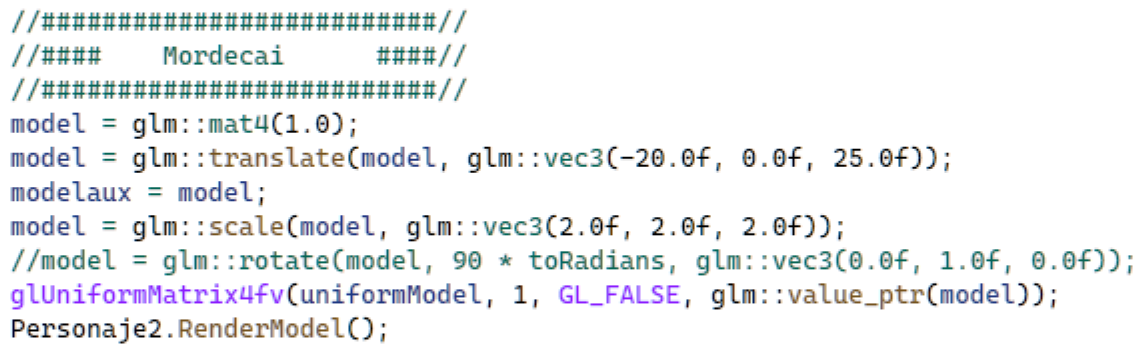

```
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-40.398f, 0.0f, 40.775f));
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
model = glm::rotate(model, 43 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
arbusto.RenderModel();

model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-35.0f, 0.0f, -45.0f));
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
model = glm::rotate(model, 80 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
arbusto.RenderModel();
```

Supporting Character: Mordecai

The secondary character is Mordecai. An internet model was used to represent the character. It is located opposite the park house.

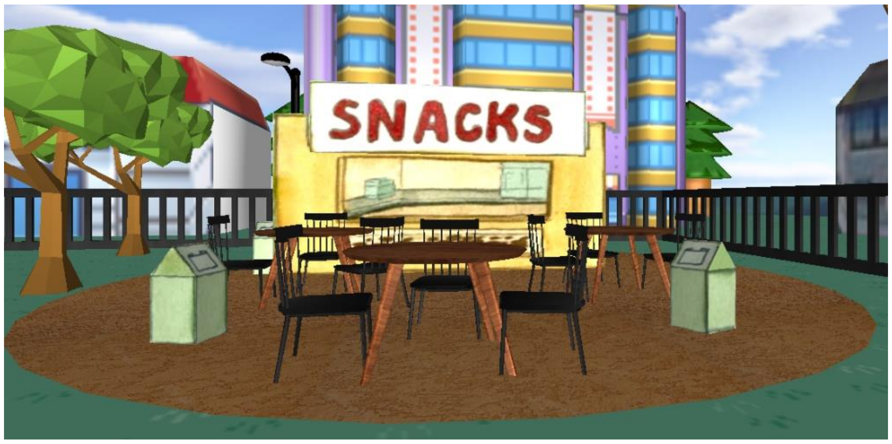

```
Personaje2 = Model();
Personaje2.LoadModel("Models/Mordecai.obj");
```

On stage.


```
//############################//
//####    Mordecai      ####//
//############################//
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-20.0f, 0.0f, 25.0f));
modelaux = model;
model = glm::scale(model, glm::vec3(2.0f, 2.0f, 2.0f));
//model = glm::rotate(model, 90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Personaje2.RenderModel();
```

View of the objects in the execution of the program:

ACKS

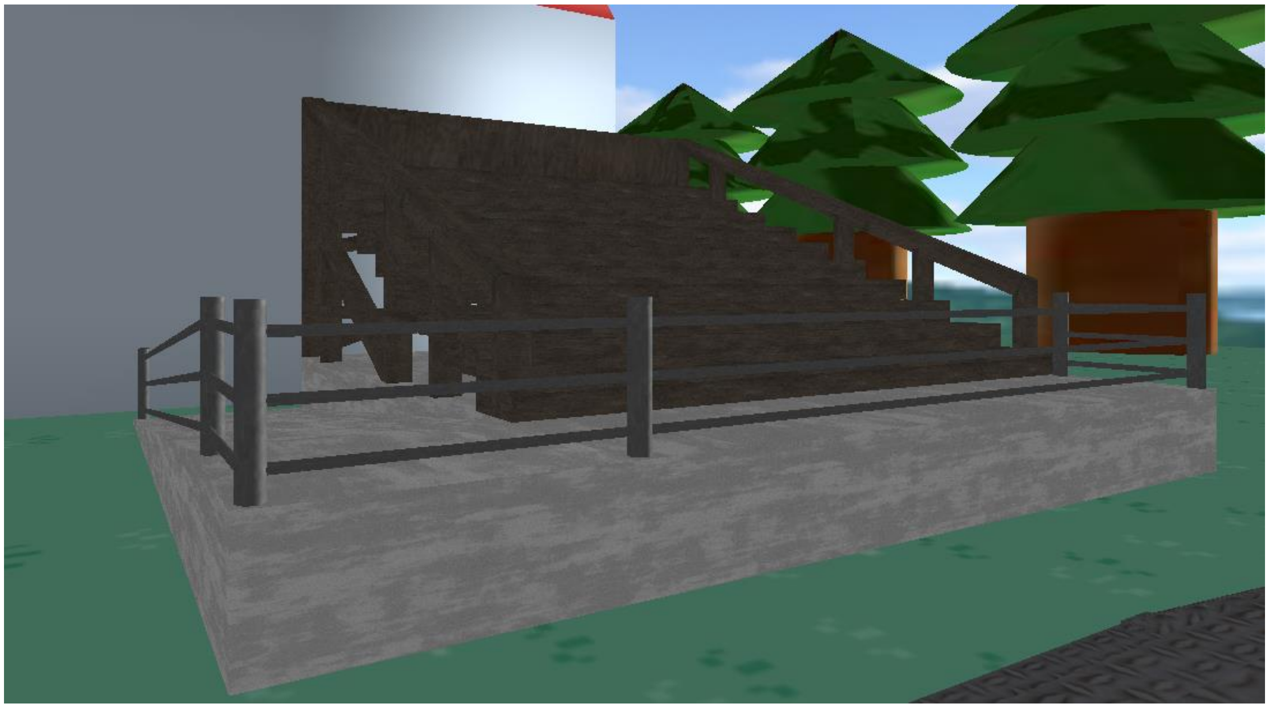

SNACKS

SNA

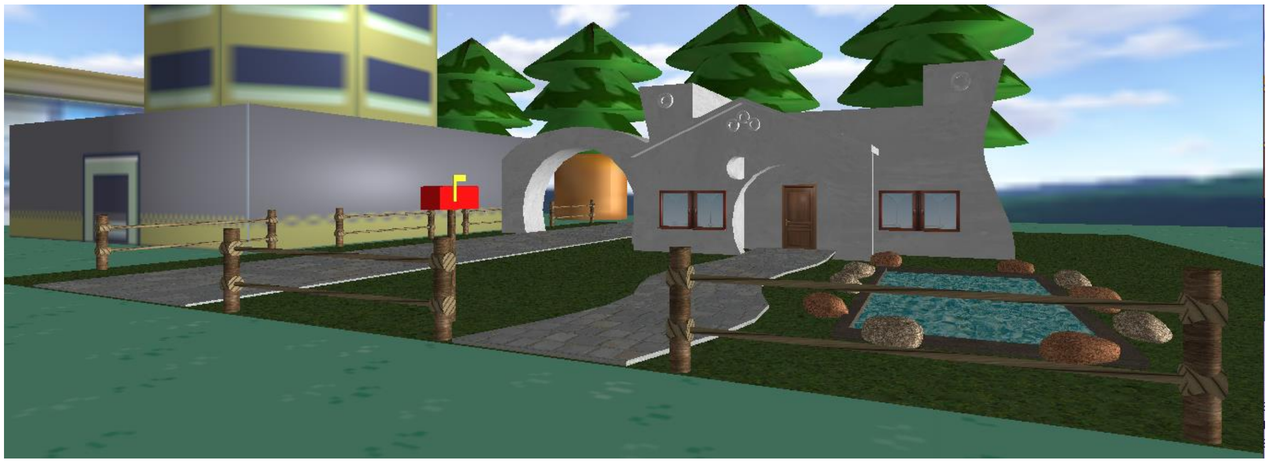
Frijolito (added)

School bus:

Steps:

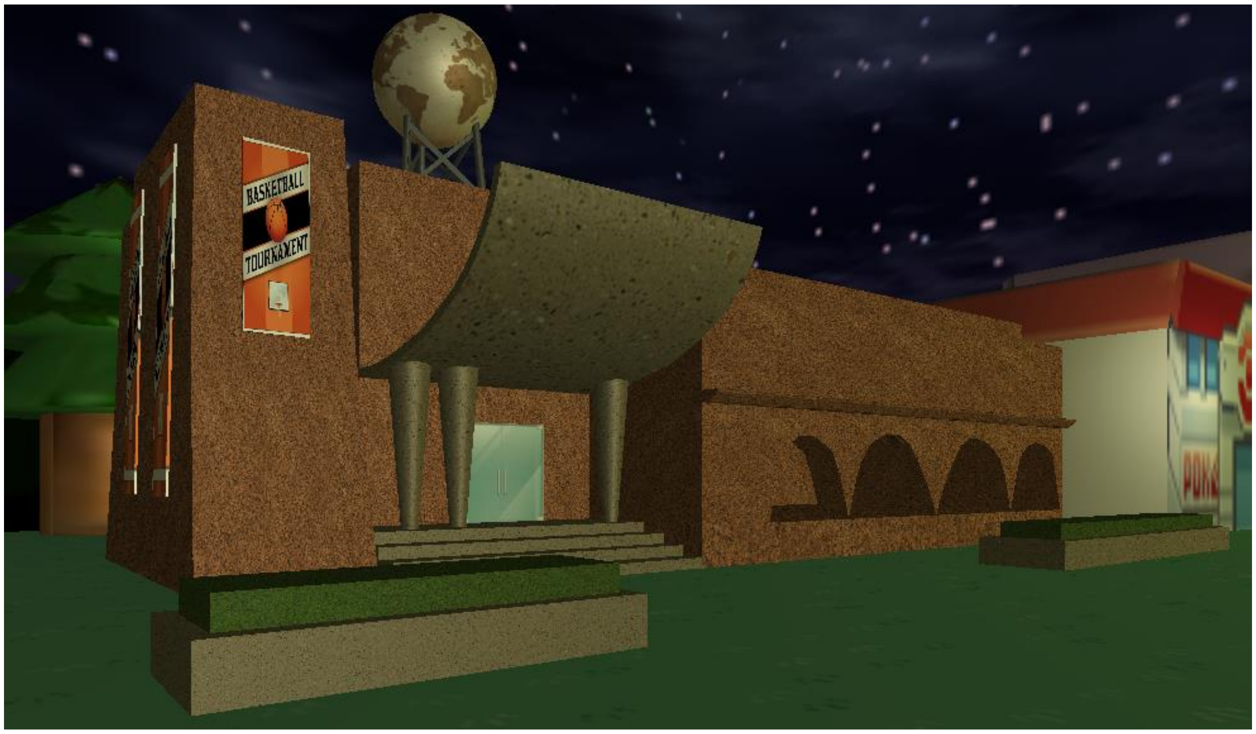


Frijolito house:

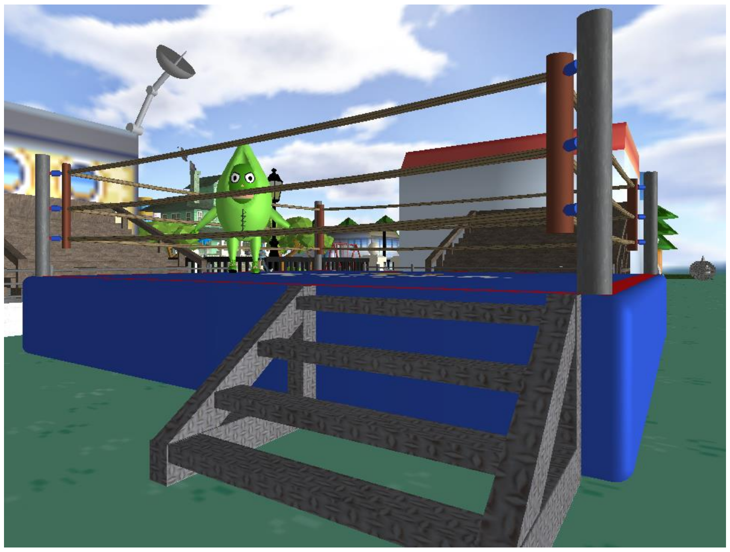


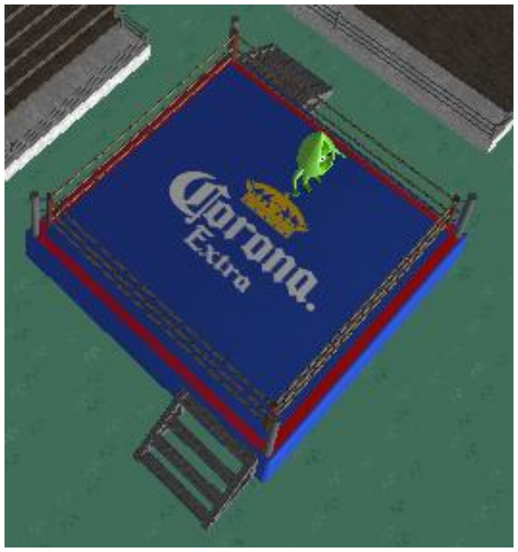
In this case, there are two models, one is the house with the garden and the other is the pond in which the texture animation was implemented to simulate the animation of the smoke in the central house.

School:



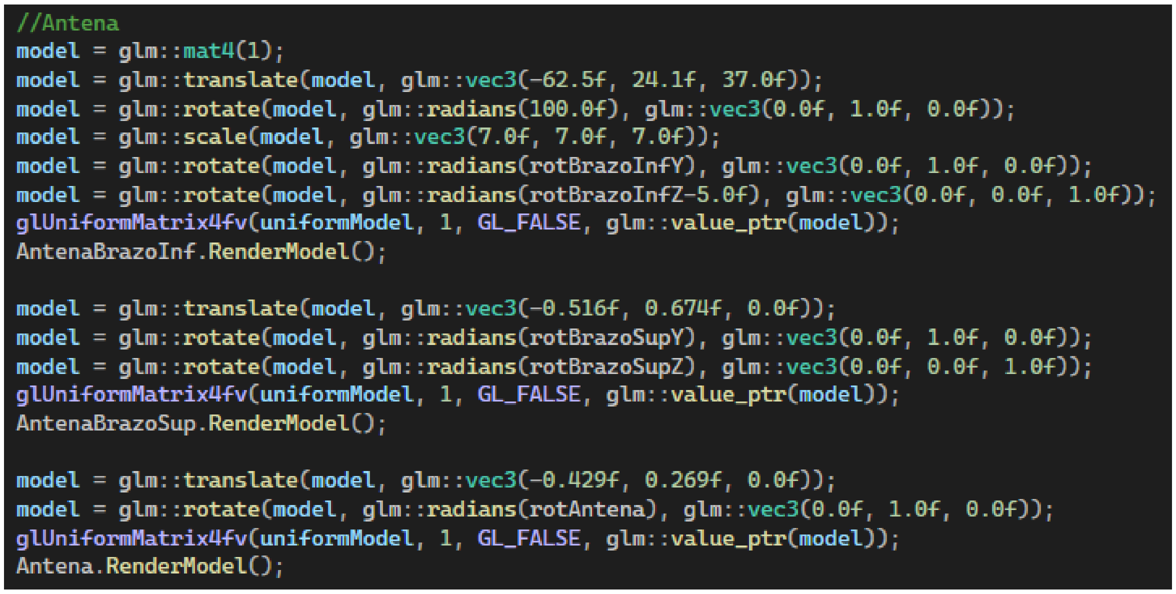
Ring:





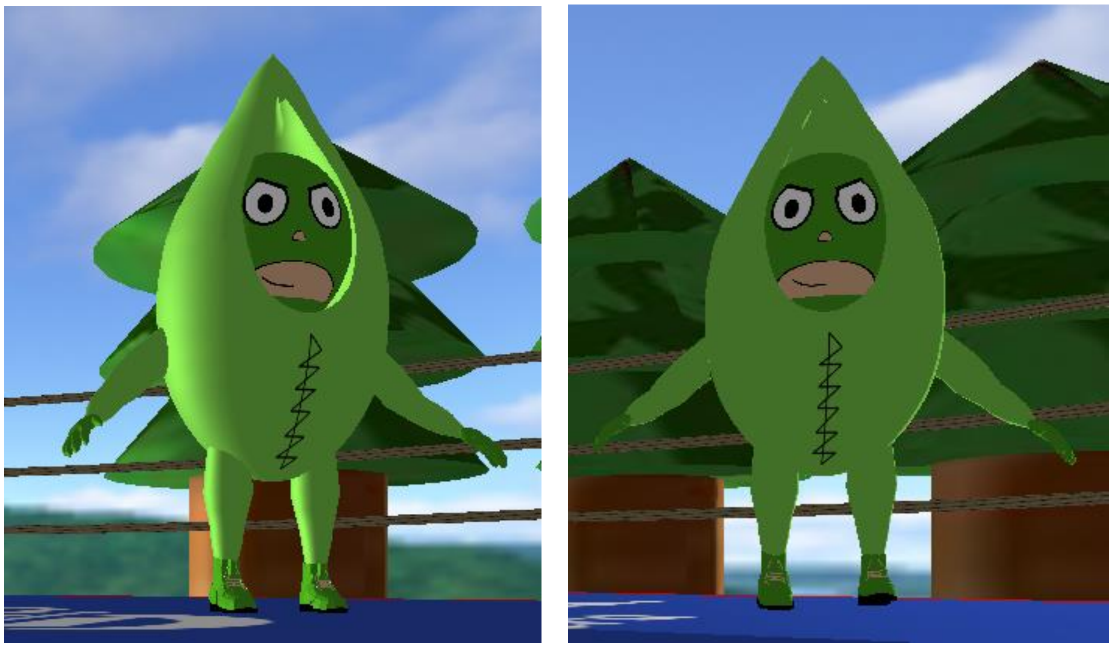
Antenna:

This object was handled with hierarchical modeling, separated into two arms plus the antenna itself, in order to implement keyframe animation to carry out a sequence.

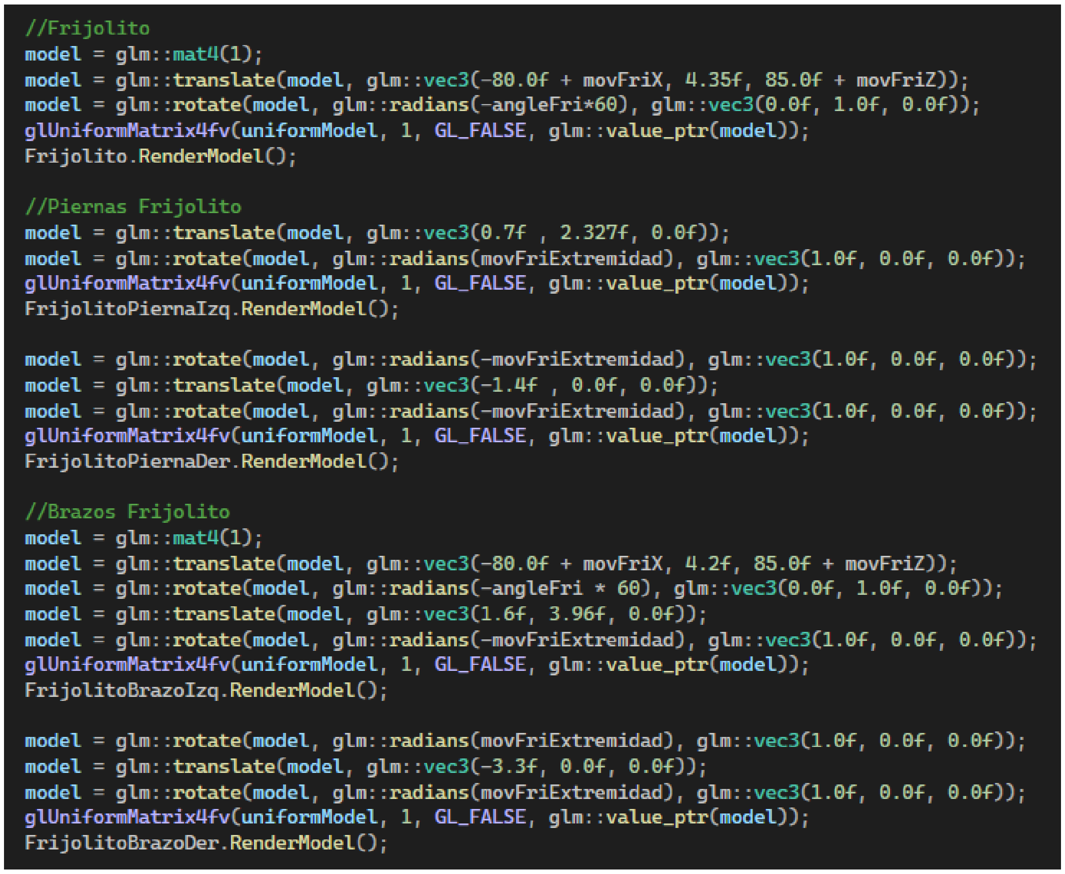


```
//Antena
model = glm::mat4(1);
model = glm::translate(model, glm::vec3(-62.5f, 24.1f, 37.0f));
model = glm::rotate(model, glm::radians(100.0f), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::scale(model, glm::vec3(7.0f, 7.0f, 7.0f));
model = glm::rotate(model, glm::radians(rotBrazoInfY), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::rotate(model, glm::radians(rotBrazoInfZ-5.0f), glm::vec3(0.0f, 0.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
AntenaBrazoInf.RenderModel();

model = glm::translate(model, glm::vec3(-0.516f, 0.674f, 0.0f));
model = glm::rotate(model, glm::radians(rotBrazoSupY), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::rotate(model, glm::radians(rotBrazoSupZ), glm::vec3(0.0f, 0.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
AntenaBrazoSup.RenderModel();

model = glm::translate(model, glm::vec3(-0.429f, 0.269f, 0.0f));
model = glm::rotate(model, glm::radians(rotAntena), glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Antena.RenderModel();
```

Frijolito Character:



This last model used hierarchical modeling to obtain a more natural animation when walking, as its arms and legs move.

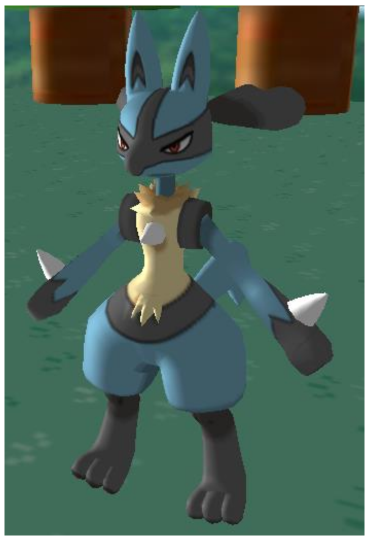


```
//Frijolito
model = glm::mat4(1);
model = glm::translate(model, glm::vec3(-80.0f + movFriX, 4.35f, 85.0f + movFriZ));
model = glm::rotate(model, glm::radians(-angleFri*60), glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Frijolito.RenderModel();

//Piernas Frijolito
model = glm::translate(model, glm::vec3(0.7f , 2.327f, 0.0f));
model = glm::rotate(model, glm::radians(movFriExtremidad), glm::vec3(1.0f, 0.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
FrijolitoPiernaIzq.RenderModel();

model = glm::rotate(model, glm::radians(-movFriExtremidad), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::translate(model, glm::vec3(-1.4f , 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(-movFriExtremidad), glm::vec3(1.0f, 0.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
FrijolitoPiernaDer.RenderModel();

//Brazos Frijolito
model = glm::mat4(1);
model = glm::translate(model, glm::vec3(-80.0f + movFriX, 4.2f, 85.0f + movFriZ));
model = glm::rotate(model, glm::radians(-angleFri * 60), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::translate(model, glm::vec3(1.6f, 3.96f, 0.0f));
model = glm::rotate(model, glm::radians(-movFriExtremidad), glm::vec3(1.0f, 0.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
FrijolitoBrazoIzq.RenderModel();

model = glm::rotate(model, glm::radians(movFriExtremidad), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::translate(model, glm::vec3(-3.3f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(movFriExtremidad), glm::vec3(1.0f, 0.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
FrijolitoBrazoDer.RenderModel();
```

- Avatar

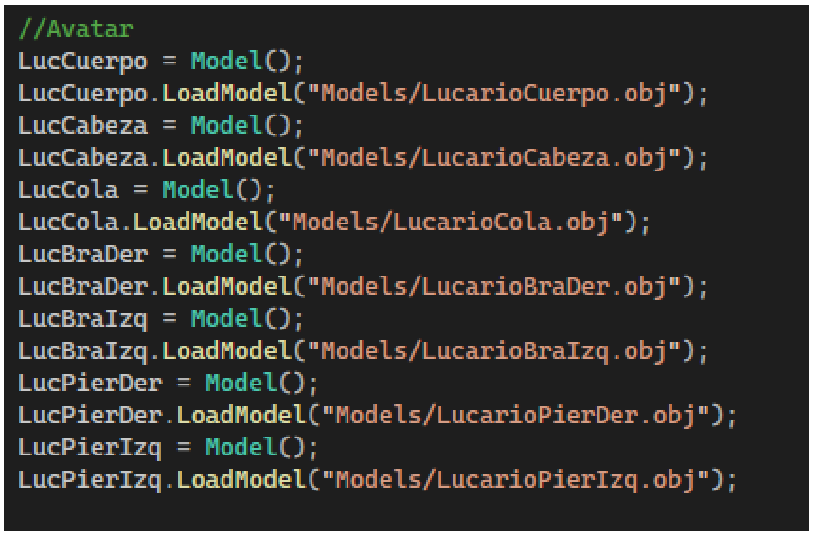
As mentioned above, this model was downloaded from the internet, the model contained a default texture, but this texture could not be exported to OpenGL, so it was re-textured using the textures provided by the model author.

Model

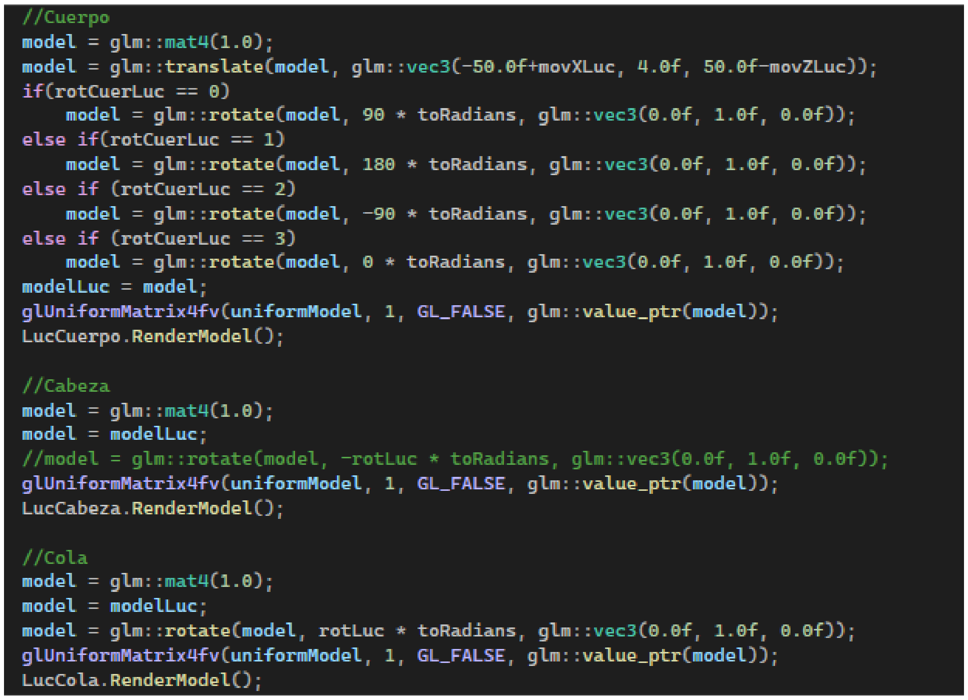


Hierarchical Creation

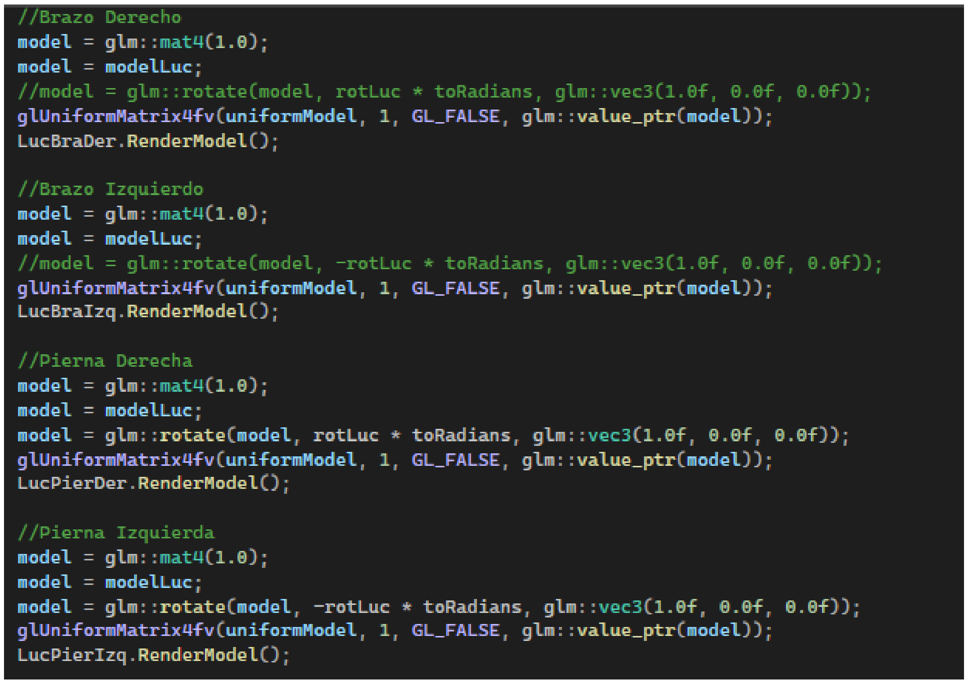
From the original downloaded model, it had to be edited, to separate each of its extremities to be imported separately and thus be able to control each of these elements separately.



```
//Avatar
LucCuerpo = Model();
LucCuerpo.LoadModel("Models/LucarioCuerpo.obj");
LucCabeza = Model();
LucCabeza.LoadModel("Models/LucarioCabeza.obj");
LucCola = Model();
LucCola.LoadModel("Models/LucarioCola.obj");
LucBraDer = Model();
LucBraDer.LoadModel("Models/LucarioBraDer.obj");
LucBraIzq = Model();
LucBraIzq.LoadModel("Models/LucarioBraIzq.obj");
LucPierDer = Model();
LucPierDer.LoadModel("Models/LucarioPierDer.obj");
LucPierIzq = Model();
LucPierIzq.LoadModel("Models/LucarioPierIzq.obj");
```



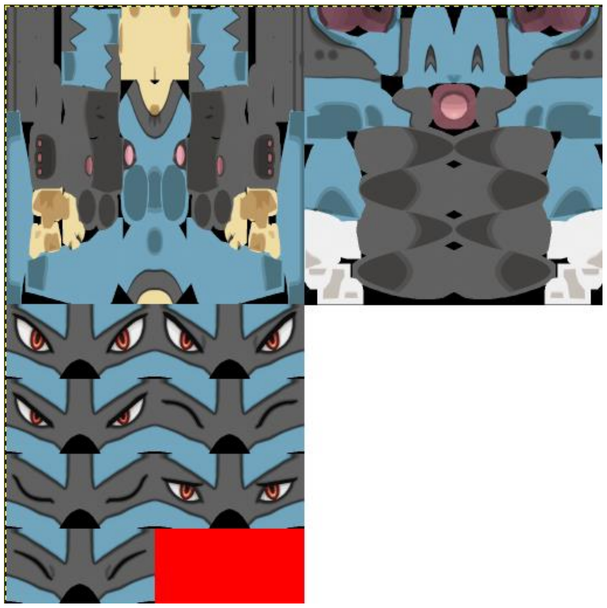
```
//Cuerpo
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-50.0f+movXLuc, 4.0f, 50.0f-movZLuc));
if(rotCuerLuc == 0)
    model = glm::rotate(model, 90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
else if(rotCuerLuc == 1)
    model = glm::rotate(model, 180 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
else if (rotCuerLuc == 2)
    model = glm::rotate(model, -90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
else if (rotCuerLuc == 3)
    model = glm::rotate(model, 0 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
modelLuc = model;
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
LucCuerpo.RenderModel();

//Cabeza
model = glm::mat4(1.0);
model = modelLuc;
//model = glm::rotate(model, -rotLuc * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
LucCabeza.RenderModel();

//Cola
model = glm::mat4(1.0);
model = modelLuc;
model = glm::rotate(model, rotLuc * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
LucCola.RenderModel();
```



```
//Brazo Derecho
model = glm::mat4(1.0);
model = modelLuc;
//model = glm::rotate(model, rotLuc * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
LucBraDer.RenderModel();

//Brazo Izquierdo
model = glm::mat4(1.0);
model = modelLuc;
//model = glm::rotate(model, -rotLuc * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
LucBraIzq.RenderModel();

//Pierna Derecha
model = glm::mat4(1.0);
model = modelLuc;
model = glm::rotate(model, rotLuc * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
LucPierDer.RenderModel();

//Pierna Izquierda
model = glm::mat4(1.0);
model = modelLuc;
model = glm::rotate(model, -rotLuc * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
LucPierIzq.RenderModel();
```

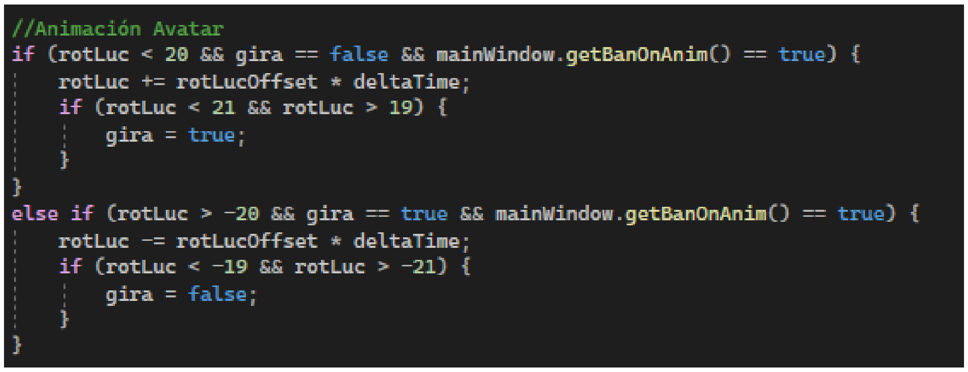
Texture

The image created and optimized for the correct texturing of the character is shown below:

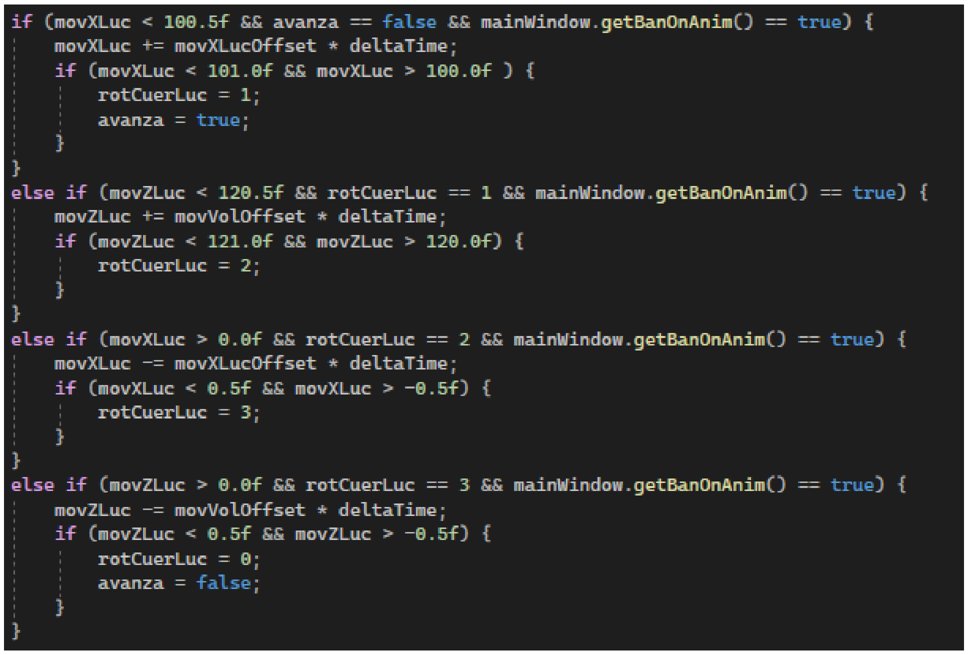


Animation

The animation of the avatar consists of going around the stage outside the park in a consecutive cycle.


```
//Animación Avatar
if (rotLuc < 20 && gira == false && mainWindow.getBanOnAnim() == true) {
    rotLuc += rotLucOffset * deltaTime;
    if (rotLuc < 21 && rotLuc > 19) {
        gira = true;
    }
}
else if (rotLuc > -20 && gira == true && mainWindow.getBanOnAnim() == true) {
    rotLuc -= rotLucOffset * deltaTime;
    if (rotLuc < -19 && rotLuc > -21) {
        gira = false;
    }
}
```

The first part of the code allows the model to move a limb from one side to another, regardless of where the avatar is walking.


```
if (movXLuc < 100.5f && avanza == false && mainWindow.getBanOnAnim() == true) {
    movXLuc += movXLucOffset * deltaTime;
    if (movXLuc < 101.0f && movXLuc > 100.0f ) {
        rotCuerLuc = 1;
        avanza = true;
    }
}
else if (movZLuc < 120.5f && rotCuerLuc == 1 && mainWindow.getBanOnAnim() == true) {
    movZLuc += movVolOffset * deltaTime;
    if (movZLuc < 121.0f && movZLuc > 120.0f) {
        rotCuerLuc = 2;
    }
}
else if (movXLuc > 0.0f && rotCuerLuc == 2 && mainWindow.getBanOnAnim() == true) {
    movXLuc -= movXLucOffset * deltaTime;
    if (movXLuc < 0.5f && movXLuc > -0.5f) {
        rotCuerLuc = 3;
    }
}
else if (movZLuc > 0.0f && rotCuerLuc == 3 && mainWindow.getBanOnAnim() == true) {
    movZLuc -= movVolOffset * deltaTime;
    if (movZLuc < 0.5f && movZLuc > -0.5f) {
        rotCuerLuc = 0;
        avanza = false;
    }
}
```

The second part of the animation allows the avatar to loop around the stage, all of these animations are activated by pressing the O key and deactivated by pressing the P key.

- ## Route

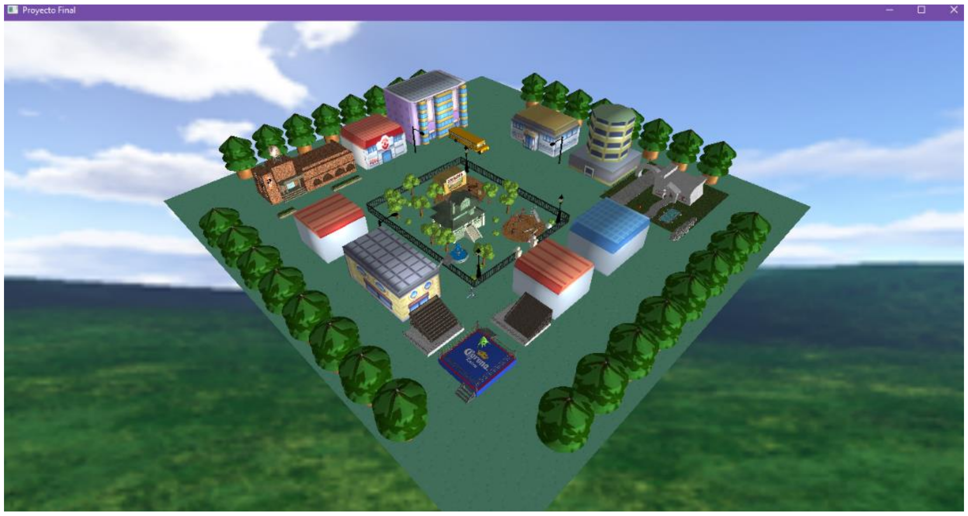
Third Person Camera

For the tour, there is a camera linked to the XZ plane, which allows us to freely roam the stage, without the possibility of moving the camera up or down, it is only possible to move in different directions by turning left and right.

Proyecto Final

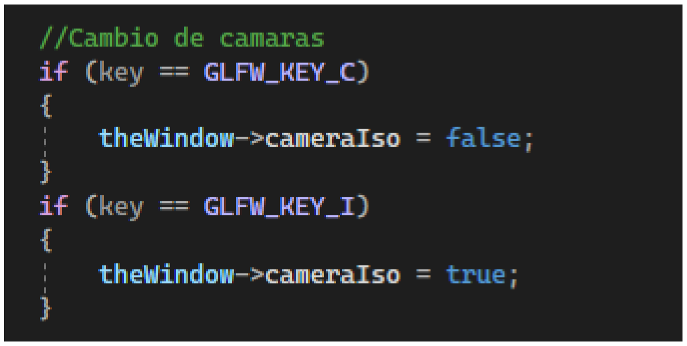
Isometric Camera

In addition to the third-person camera, to go around the scene there is a second Isometric type camera, which allows us to observe the whole scene, allowing us to zoom in or out of the camera.

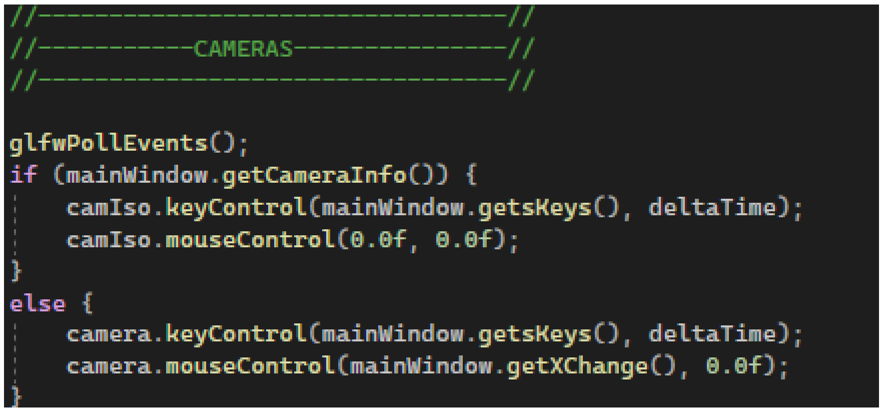


Camera Position Saving

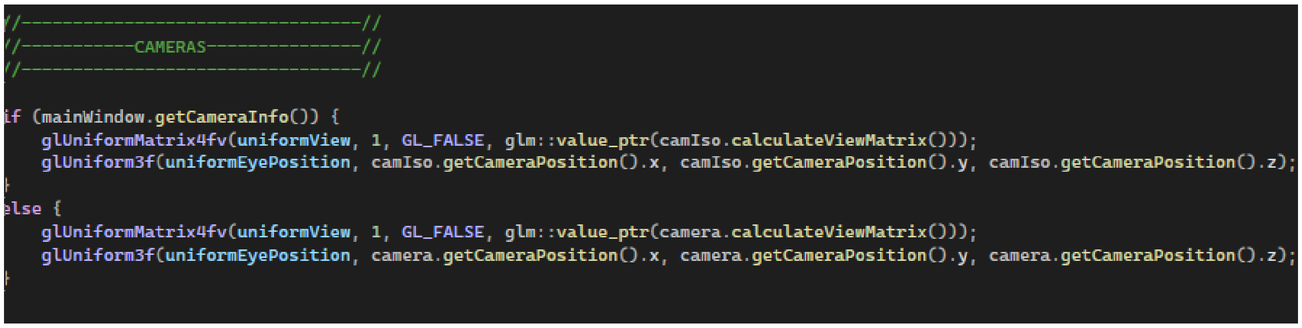
It starts with the Isometric camera, pressing the C key changes to the third-person camera, at any time you can press the I key to return to the Isometric camera, in both cases the camera will save the position where you had stayed before.



```
//Cambio de camaras
if (key == GLFW_KEY_C)
{
    theWindow->cameraIso = false;
}
if (key == GLFW_KEY_I)
{
    theWindow->cameraIso = true;
}
```



```
//-----------------------------------//
//------------CAMERAS----------------//
//-----------------------------------//

glfwPollEvents();
if (mainWindow.getCameraInfo()) {
    camIso.keyControl(mainWindow.getsKeys(), deltaTime);
    camIso.mouseControl(0.0f, 0.0f);
}
else {
    camera.keyControl(mainWindow.getsKeys(), deltaTime);
    camera.mouseControl(mainWindow.getXChange(), 0.0f);
}
```

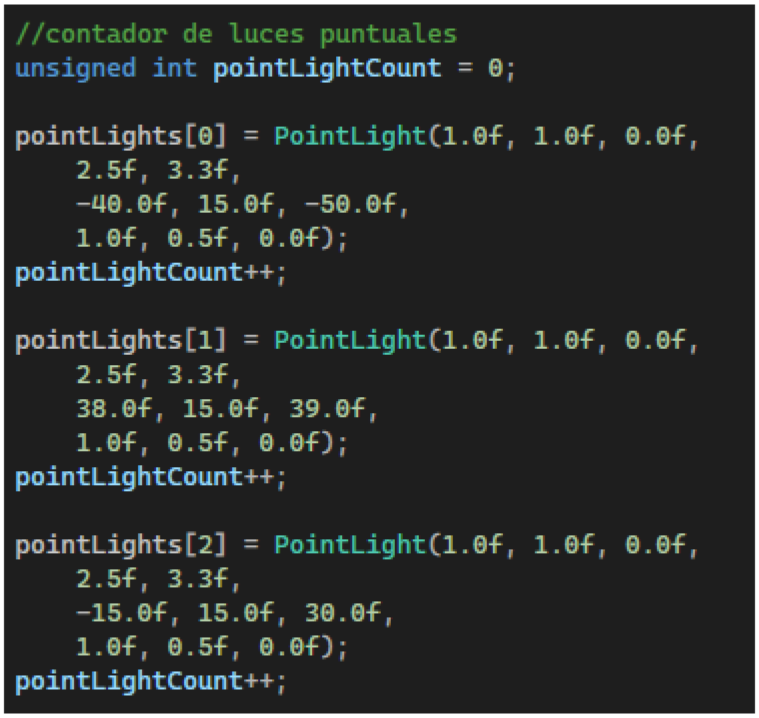


```
/---------------------------------//
/-----------CAMERAS---------------//
/---------------------------------//

if (mainWindow.getCameraInfo()) {
    glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camIso.calculateViewMatrix()));
    glUniform3f(uniformEyePosition, camIso.getCameraPosition().x, camIso.getCameraPosition().y, camIso.getCameraPosition().z);
}
else {
    glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
    glUniform3f(uniformEyePosition, camera.getCameraPosition().x, camera.getCameraPosition().y, camera.getCameraPosition().z);
}
```

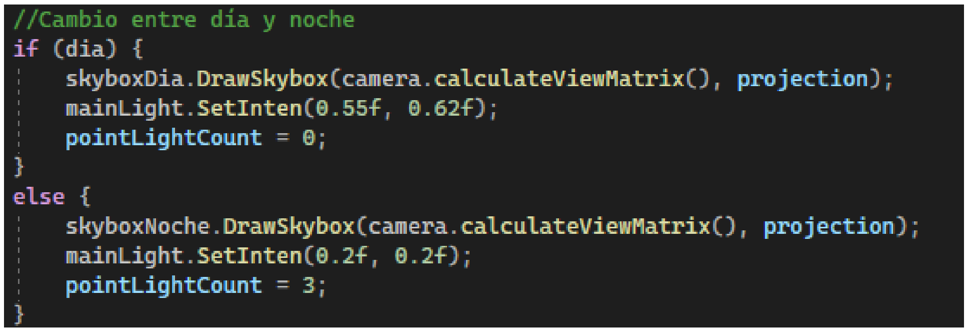
- ## Illumination

Pointlight:

For this point, three post-type lamps were created, which were located and distributed around the park, these lights turn on and off automatically when changing to night.



```
//contador de luces puntuales
unsigned int pointLightCount = 0;

pointLights[0] = PointLight(1.0f, 1.0f, 0.0f,
    2.5f, 3.3f,
    -40.0f, 15.0f, -50.0f,
    1.0f, 0.5f, 0.0f);
pointLightCount++;

pointLights[1] = PointLight(1.0f, 1.0f, 0.0f,
    2.5f, 3.3f,
    38.0f, 15.0f, 39.0f,
    1.0f, 0.5f, 0.0f);
pointLightCount++;

pointLights[2] = PointLight(1.0f, 1.0f, 0.0f,
    2.5f, 3.3f,
    -15.0f, 15.0f, 30.0f,
    1.0f, 0.5f, 0.0f);
pointLightCount++;
```

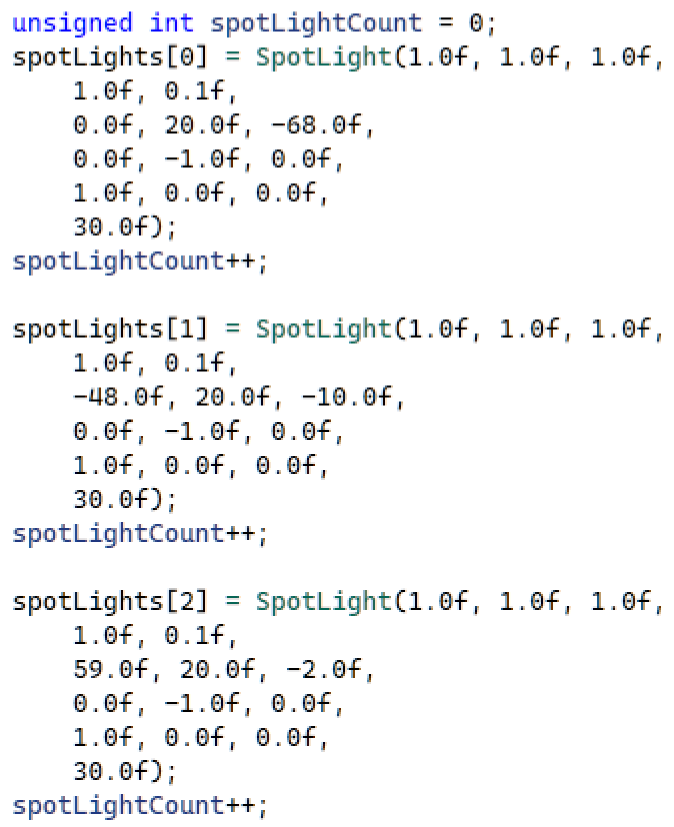


```
//Cambio entre día y noche
if (dia) {
    skyboxDia.DrawSkybox(camera.calculateViewMatrix(), projection);
    mainLight.SetInten(0.55f, 0.62f);
    pointLightCount = 0;
}
else {
    skyboxNoche.DrawSkybox(camera.calculateViewMatrix(), projection);
    mainLight.SetInten(0.2f, 0.2f);
    pointLightCount = 3;
}
```

For this implementation, the Punctual type lights counter is changed, when it is day it is set to 0 so that all are off and when it gets dark the counter is set to 3 to show all the lights.

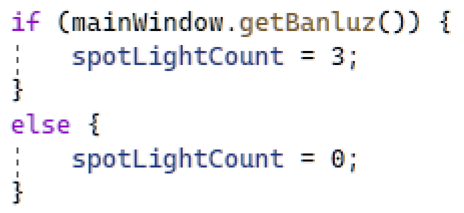
Spotlight:

These lights are in the 3 lampposts that are outside the park, activating when the Z key is pressed and deactivating with the X key. All these light sources shine towards the ground.



```
unsigned int spotLightCount = 0;
spotLights[0] = SpotLight(1.0f, 1.0f, 1.0f,
    1.0f, 0.1f,
    0.0f, 20.0f, -68.0f,
    0.0f, -1.0f, 0.0f,
    1.0f, 0.0f, 0.0f,
    30.0f);
spotLightCount++;

spotLights[1] = SpotLight(1.0f, 1.0f, 1.0f,
    1.0f, 0.1f,
    -48.0f, 20.0f, -10.0f,
    0.0f, -1.0f, 0.0f,
    1.0f, 0.0f, 0.0f,
    30.0f);
spotLightCount++;

spotLights[2] = SpotLight(1.0f, 1.0f, 1.0f,
    1.0f, 0.1f,
    59.0f, 20.0f, -2.0f,
    0.0f, -1.0f, 0.0f,
    1.0f, 0.0f, 0.0f,
    30.0f);
spotLightCount++;
```

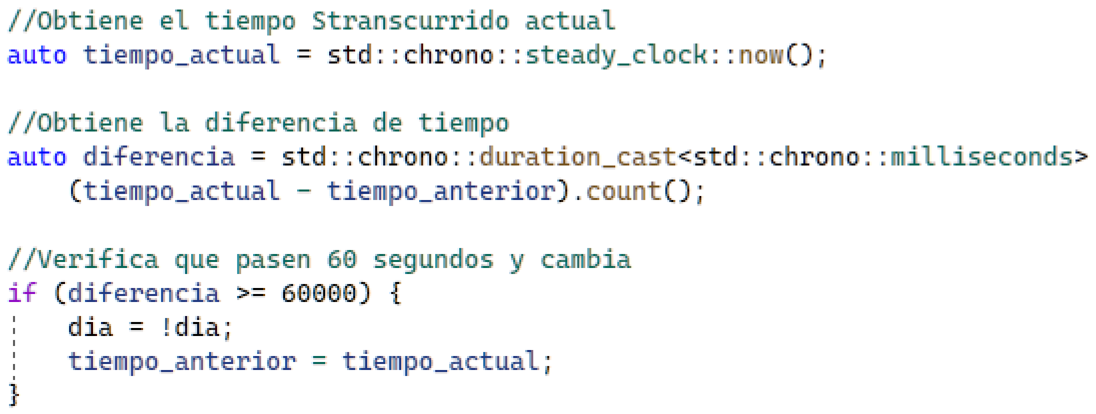
A flag is used to know which key is being pressed.



```
if (mainWindow.getBanluz()) {
    spotLightCount = 3;
}
else {
    spotLightCount = 0;
}
```

SkyBox lighting:

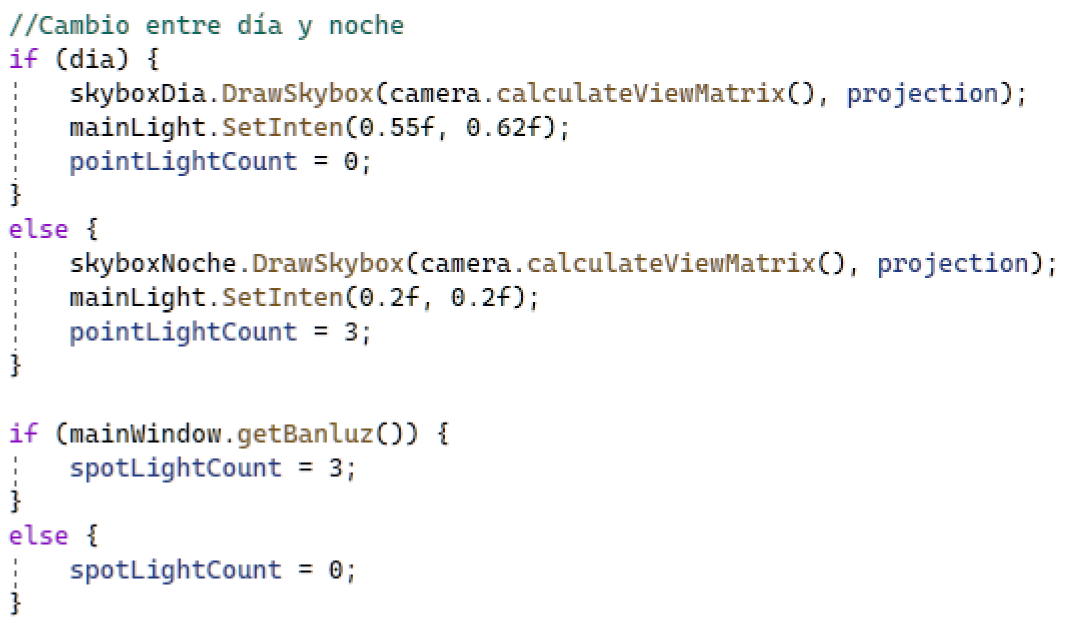
The course of day and night has a period of one minute. Every minute the flag called 'day' alternates between true and false. When this happens, the skybox changes its texture from a light sky to a dark one.



```
//Obtiene el tiempo Stranscurrido actual
auto tiempo_actual = std::chrono::steady_clock::now();

//Obtiene la diferencia de tiempo
auto diferencia = std::chrono::duration_cast<std::chrono::milliseconds>
    (tiempo_actual - tiempo_anterior).count();

//Verifica que pasen 60 segundos y cambia
if (diferencia >= 60000) {
    dia = !dia;
    tiempo_anterior = tiempo_actual;
}
```

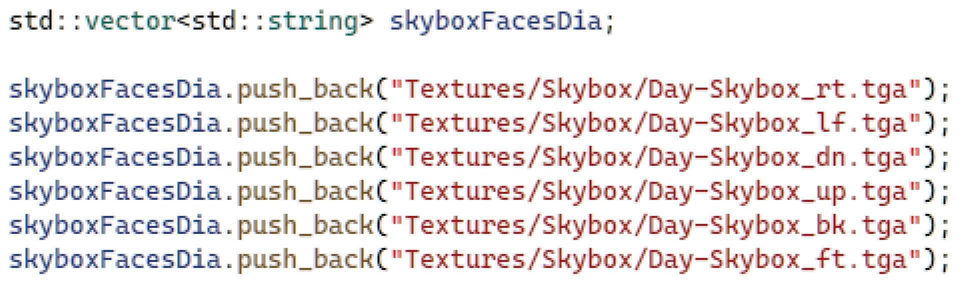
In addition to changing the background, it will change the intensity of the directional light and the pointlights will not render to stay off during the day and on at night.

Change code.

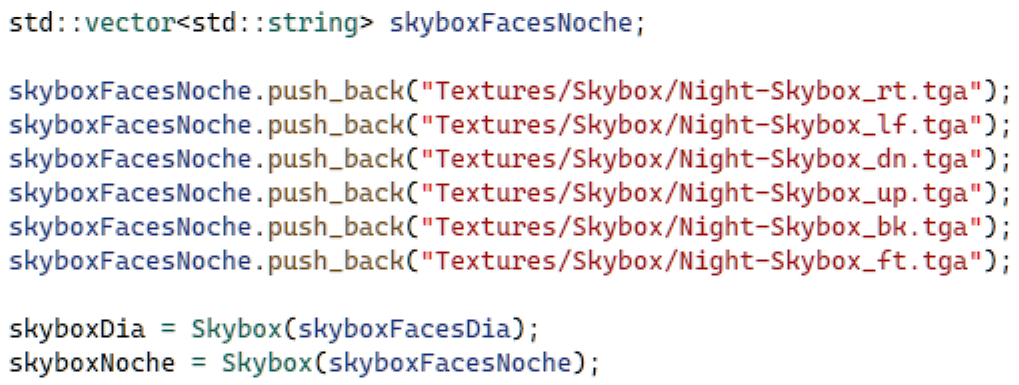


```
//Cambio entre día y noche
if (dia) {
    skyboxDia.DrawSkybox(camera.calculateViewMatrix(), projection);
    mainLight.SetInten(0.55f, 0.62f);
    pointLightCount = 0;
}
else {
    skyboxNoche.DrawSkybox(camera.calculateViewMatrix(), projection);
    mainLight.SetInten(0.2f, 0.2f);
    pointLightCount = 3;
}

if (mainWindow.getBanluz()) {
    spotLightCount = 3;
}
else {
    spotLightCount = 0;
}
```

Changing the textures in the skybox:



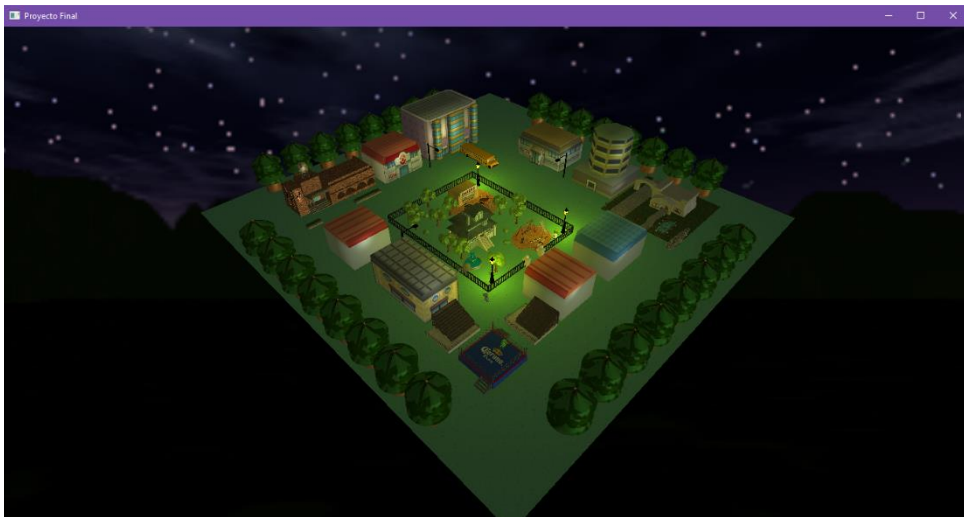
```
std::vector<std::string> skyboxFacesDia;

skyboxFacesDia.push_back("Textures/Skybox/Day-Skybox_rt.tga");
skyboxFacesDia.push_back("Textures/Skybox/Day-Skybox_lf.tga");
skyboxFacesDia.push_back("Textures/Skybox/Day-Skybox_dn.tga");
skyboxFacesDia.push_back("Textures/Skybox/Day-Skybox_up.tga");
skyboxFacesDia.push_back("Textures/Skybox/Day-Skybox_bk.tga");
skyboxFacesDia.push_back("Textures/Skybox/Day-Skybox_ft.tga");
```



```
std::vector<std::string> skyboxFacesNoche;

skyboxFacesNoche.push_back("Textures/Skybox/Night-Skybox_rt.tga");
skyboxFacesNoche.push_back("Textures/Skybox/Night-Skybox_lf.tga");
skyboxFacesNoche.push_back("Textures/Skybox/Night-Skybox_dn.tga");
skyboxFacesNoche.push_back("Textures/Skybox/Night-Skybox_up.tga");
skyboxFacesNoche.push_back("Textures/Skybox/Night-Skybox_bk.tga");
skyboxFacesNoche.push_back("Textures/Skybox/Night-Skybox_ft.tga");

skyboxDia = Skybox(skyboxFacesDia);
skyboxNoche = Skybox(skyboxFacesNoche);
```

SkyBox textures displayed while running:

Proyecto Final



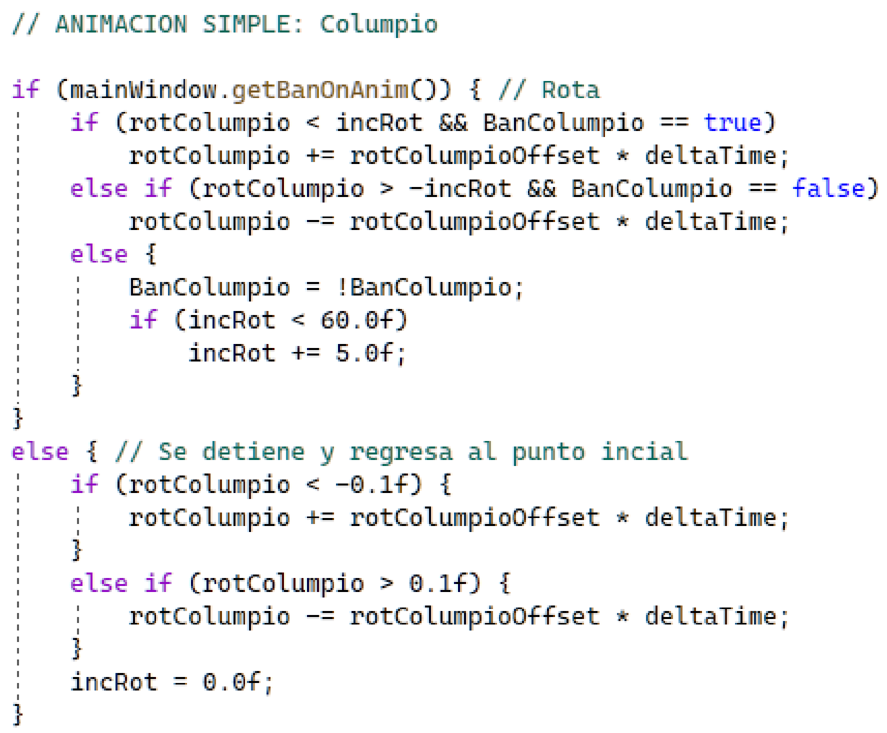
- ## Animation

Simple

For simple animations, we have the movement of the swings and the movement of the ramps of the up and down, as well as the movement of a Pokémon. All these animations are activated using the 'O' key and stopped with the 'P' key.

Swing movement:

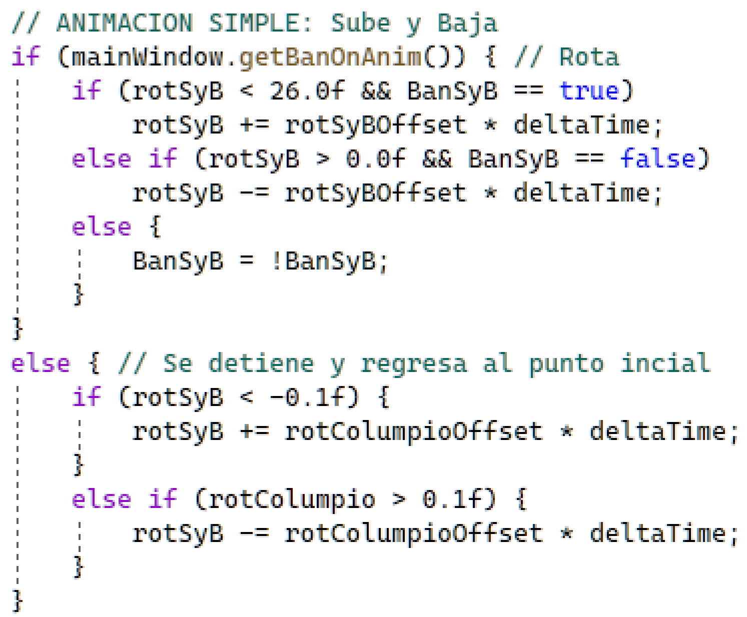
The swings rotate giving a rocking effect and it is also incremental as time goes by, that is, the height with which it rotates increases. The movement will repeat until the animation stops.



```
// ANIMACION SIMPLE: Columpio

if (mainWindow.getBanOnAnim()) { // Rota
    if (rotColumpio < incRot && BanColumpio == true)
        rotColumpio += rotColumpioOffset * deltaTime;
    else if (rotColumpio > -incRot && BanColumpio == false)
        rotColumpio -= rotColumpioOffset * deltaTime;
    else {
        BanColumpio = !BanColumpio;
        if (incRot < 60.0f)
            incRot += 5.0f;
    }
}
else { // Se detiene y regresa al punto incial
    if (rotColumpio < -0.1f) {
        rotColumpio += rotColumpioOffset * deltaTime;
    }
    else if (rotColumpio > 0.1f) {
        rotColumpio -= rotColumpioOffset * deltaTime;
    }
    incRot = 0.0f;
}
```

Movement of the ramp up and down:

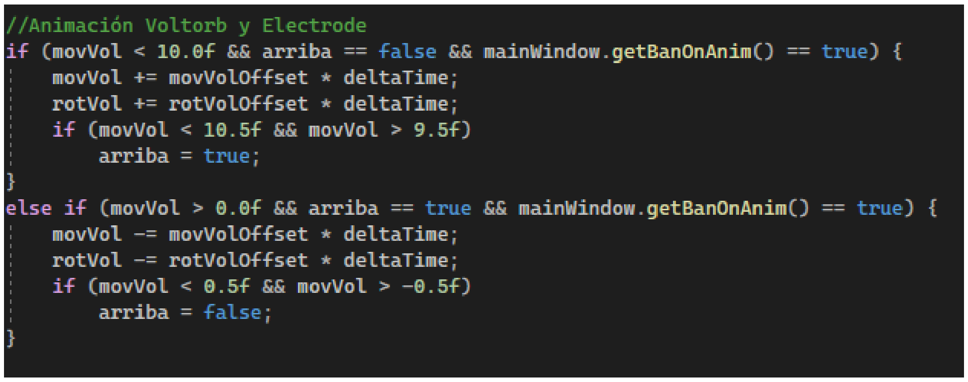
The ramp rotates in a central axis and each time it reaches the ground it turns back until the animation stops.



```
// ANIMACION SIMPLE: Sube y Baja
if (mainWindow.getBanOnAnim()) { // Rota
    if (rotSyB < 26.0f && BanSyB == true)
        rotSyB += rotSyBOffset * deltaTime;
    else if (rotSyB > 0.0f && BanSyB == false)
        rotSyB -= rotSyBOffset * deltaTime;
    else {
        BanSyB = !BanSyB;
    }
}
else { // Se detiene y regresa al punto incial
    if (rotSyB < -0.1f) {
        rotSyB += rotColumpioOffset * deltaTime;
    }
    else if (rotColumpio > 0.1f) {
        rotSyB -= rotColumpioOffset * deltaTime;
    }
}
```

Voltorb and Electrode movement:

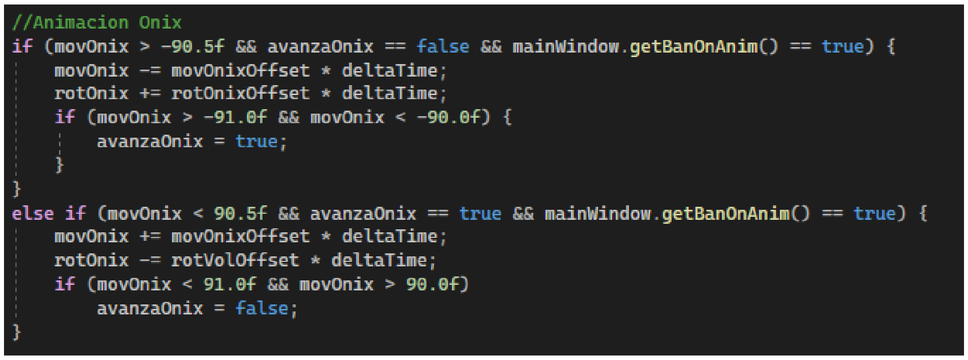
The characters of Voltorb and Electrode, make an animation in which they jump and spin at the same time in a loop.



```
//Animación Voltorb y Electrode
if (movVol < 10.0f && arriba == false && mainWindow.getBanOnAnim() == true) {
    movVol += movVolOffset * deltaTime;
    rotVol += rotVolOffset * deltaTime;
    if (movVol < 10.5f && movVol > 9.5f)
        arriba = true;
}
else if (movVol > 0.0f && arriba == true && mainWindow.getBanOnAnim() == true) {
    movVol -= movVolOffset * deltaTime;
    rotVol -= rotVolOffset * deltaTime;
    if (movVol < 0.5f && movVol > -0.5f)
        arriba = false;
}
```

Onyx movement:

Onix's character performs an animation in which it moves forward rotating on its own axis, upon reaching the end of its journey, it turns around and spins back.

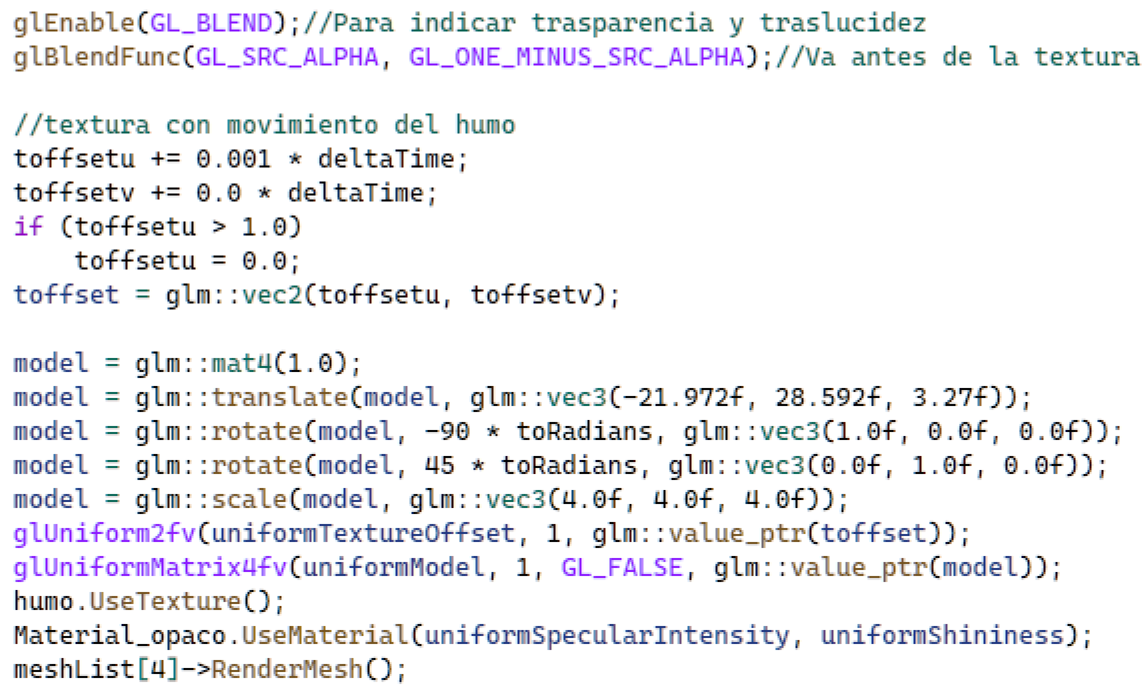


```
//Animacion Onix
if (movOnix > -90.5f && avanzaOnix == false && mainWindow.getBanOnAnim() == true) {
    movOnix -= movOnixOffset * deltaTime;
    rotOnix += rotOnixOffset * deltaTime;
    if (movOnix > -91.0f && movOnix < -90.0f) {
        avanzaOnix = true;
    }
}
else if (movOnix < 90.5f && avanzaOnix == true && mainWindow.getBanOnAnim() == true) {
    movOnix += movOnixOffset * deltaTime;
    rotOnix -= rotVolOffset * deltaTime;
    if (movOnix < 91.0f && movOnix > 90.0f)
        avanzaOnix = false;
}
```

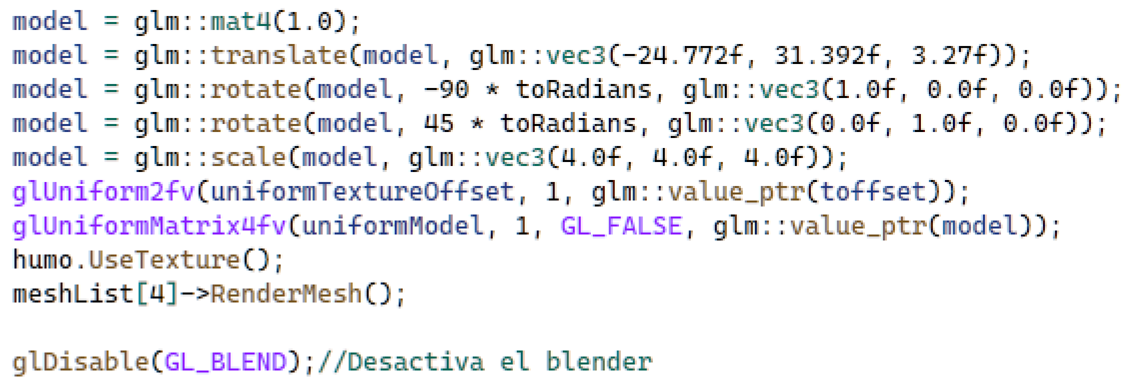
Complex

Moving texture:

A texture in the form of a cloud of smoke was used that will come out of the chimney of the house in the park. This texture will translate and rotate to give an effect of smoke coming out of the chimney.



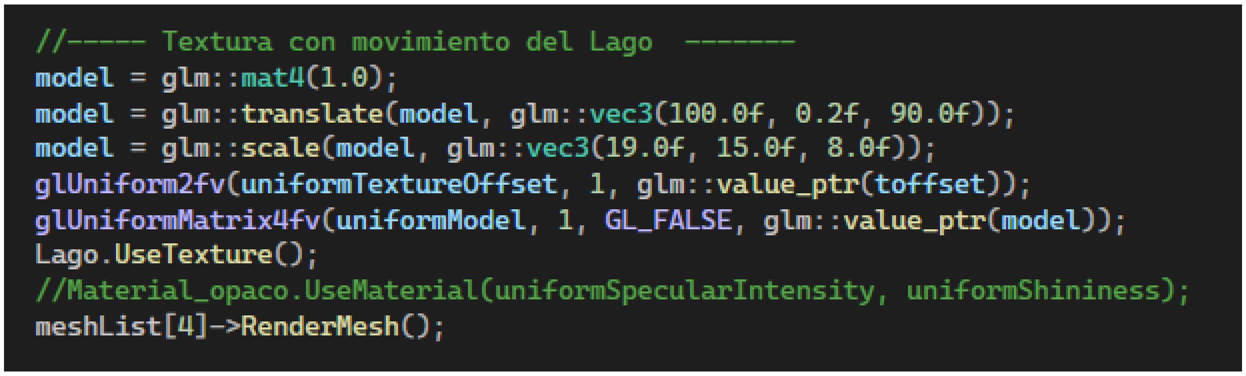
```
glEnable(GL_BLEND);//Para indicar trasparencia y traslucidez
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);//Va antes de la textura

//textura con movimiento del humo
toffsetu += 0.001 * deltaTime;
toffsetv += 0.0 * deltaTime;
if (toffsetu > 1.0)
    toffsetu = 0.0;
toffset = glm::vec2(toffsetu, toffsetv);

model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-21.972f, 28.592f, 3.27f));
model = glm::rotate(model, -90 * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::rotate(model, 45 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::scale(model, glm::vec3(4.0f, 4.0f, 4.0f));
glUniform2fv(uniformTextureOffset, 1, glm::value_ptr(toffset));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
humo.UseTexture();
Material_opaco.UseMaterial(uniformSpecularIntensity, uniformShininess);
meshList[4]->RenderMesh();
```



```
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-24.772f, 31.392f, 3.27f));
model = glm::rotate(model, -90 * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::rotate(model, 45 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::scale(model, glm::vec3(4.0f, 4.0f, 4.0f));
glUniform2fv(uniformTextureOffset, 1, glm::value_ptr(toffset));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
humo.UseTexture();
meshList[4]->RenderMesh();

glDisable(GL_BLEND);//Desactiva el blender
```

Pond

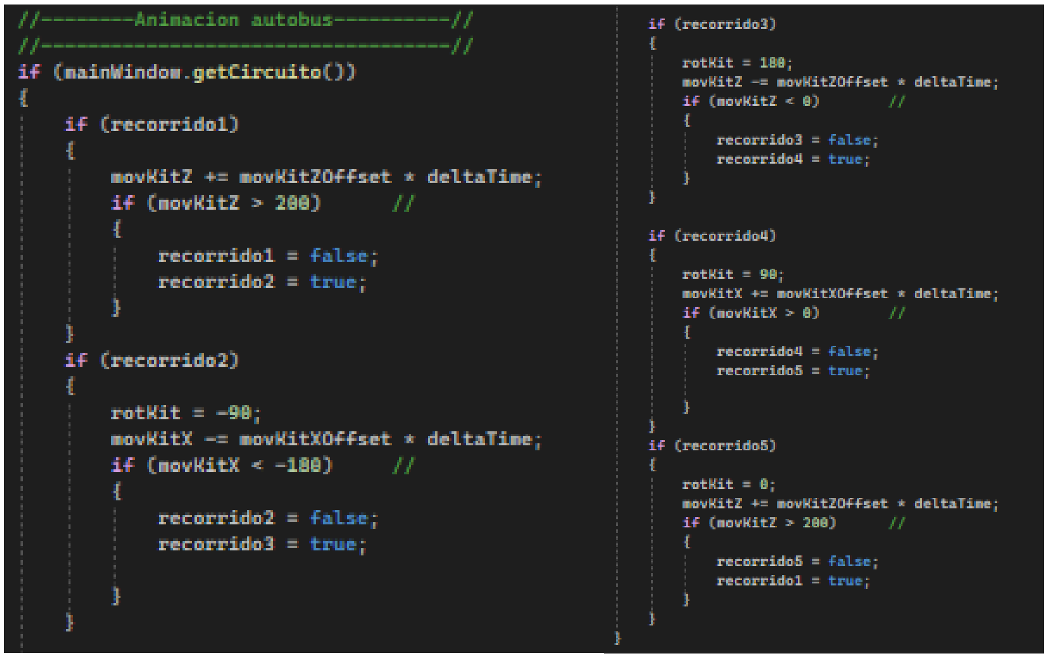
Technique like that used in smoke



```
//----- Textura con movimiento del Lago  -------
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(100.0f, 0.2f, 90.0f));
model = glm::scale(model, glm::vec3(19.0f, 15.0f, 8.0f));
glUniform2fv(uniformTextureOffset, 1, glm::value_ptr(toffset));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Lago.UseTexture();
//Material_opaco.UseMaterial(uniformSpecularIntensity, uniformShininess);
meshList[4]->RenderMesh();
```

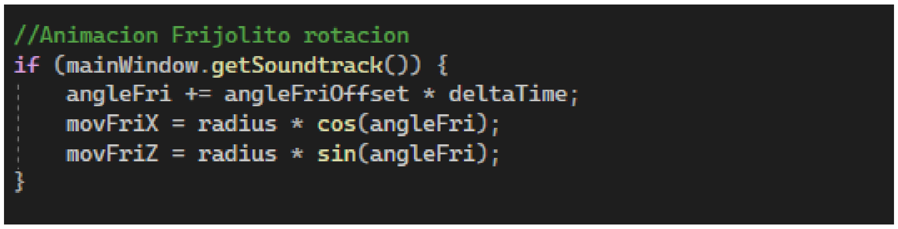
Bus

A square-shaped path is drawn where movements are combined on a single axis at a time, then it is rotated and translated on another axis:

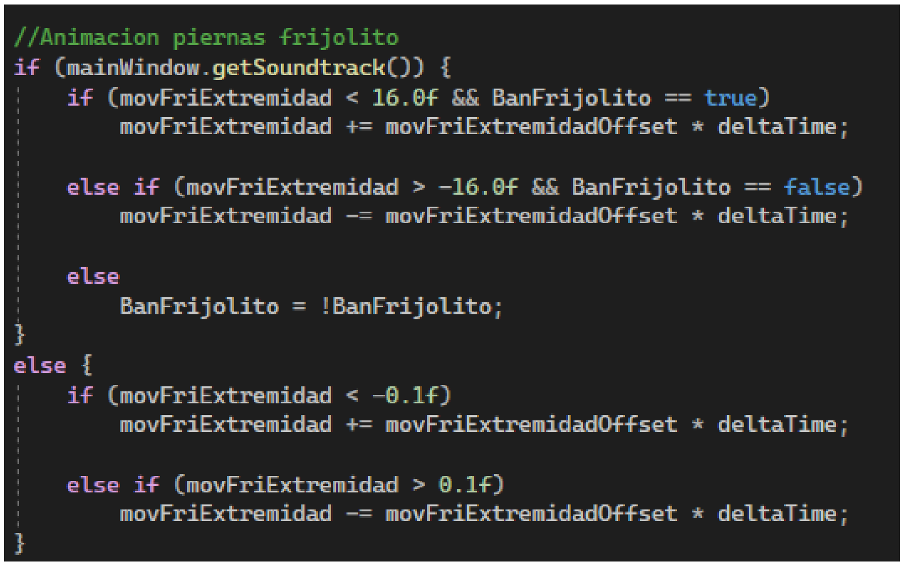


```
//--------Animacion autobus----------//
//----------------------------------//
if (mainWindow.getCircuito())
{
    if (recorrido1)
    {
        movKitZ += movKitZOffset * deltaTime;
        if (movKitZ > 200)      //
        {
            recorrido1 = false;
            recorrido2 = true;
        }
    }
    if (recorrido2)
    {
        rotKit = -90;
        movKitX -= movKitXOffset * deltaTime;
        if (movKitX < -100)     //
        {
            recorrido2 = false;
            recorrido3 = true;

        }
    }
    if (recorrido3)
    {
        rotKit = 180;
        movKitZ -= movKitZOffset * deltaTime;
        if (movKitZ < 0)        //
        {
            recorrido3 = false;
            recorrido4 = true;
        }
    }

    if (recorrido4)
    {
        rotKit = 90;
        movKitX += movKitXOffset * deltaTime;
        if (movKitX > 0)        //
        {
            recorrido4 = false;
            recorrido5 = true;

        }
    }
    if (recorrido5)
    {
        rotKit = 0;
        movKitZ += movKitZOffset * deltaTime;
        if (movKitZ > 200)      //
        {
            recorrido5 = false;
            recorrido1 = true;
        }
    }
}
```

Frijolito



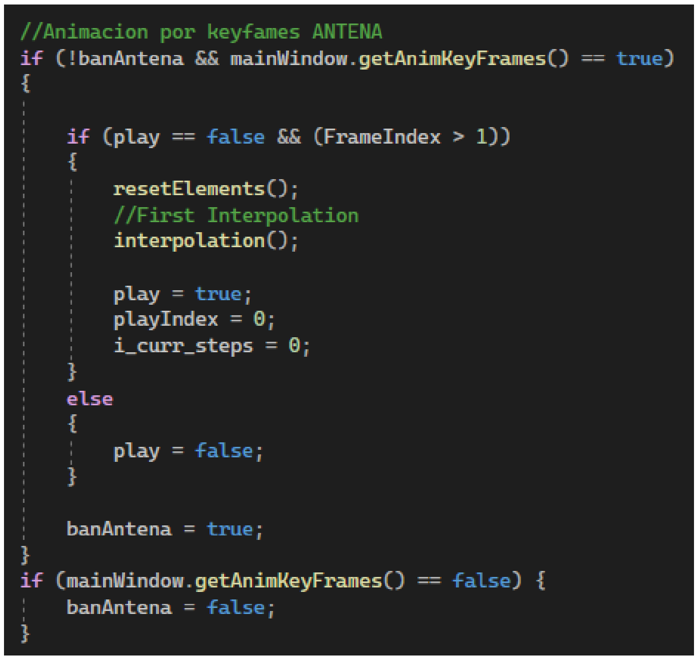
```
//Animacion Frijolito rotacion
if (mainWindow.getSoundtrack()) {
    angleFri += angleFriOffset * deltaTime;
    movFriX = radius * cos(angleFri);
    movFriZ = radius * sin(angleFri);
}
```



```
//Animacion piernas frijolito
if (mainWindow.getSoundtrack()) {
    if (movFriExtremidad < 16.0f && BanFrijolito == true)
        movFriExtremidad += movFriExtremidadOffset * deltaTime;

    else if (movFriExtremidad > -16.0f && BanFrijolito == false)
        movFriExtremidad -= movFriExtremidadOffset * deltaTime;

    else
        BanFrijolito = !BanFrijolito;
}
else {
    if (movFriExtremidad < -0.1f)
        movFriExtremidad += movFriExtremidadOffset * deltaTime;

    else if (movFriExtremidad > 0.1f)
        movFriExtremidad -= movFriExtremidadOffset * deltaTime;
}
```

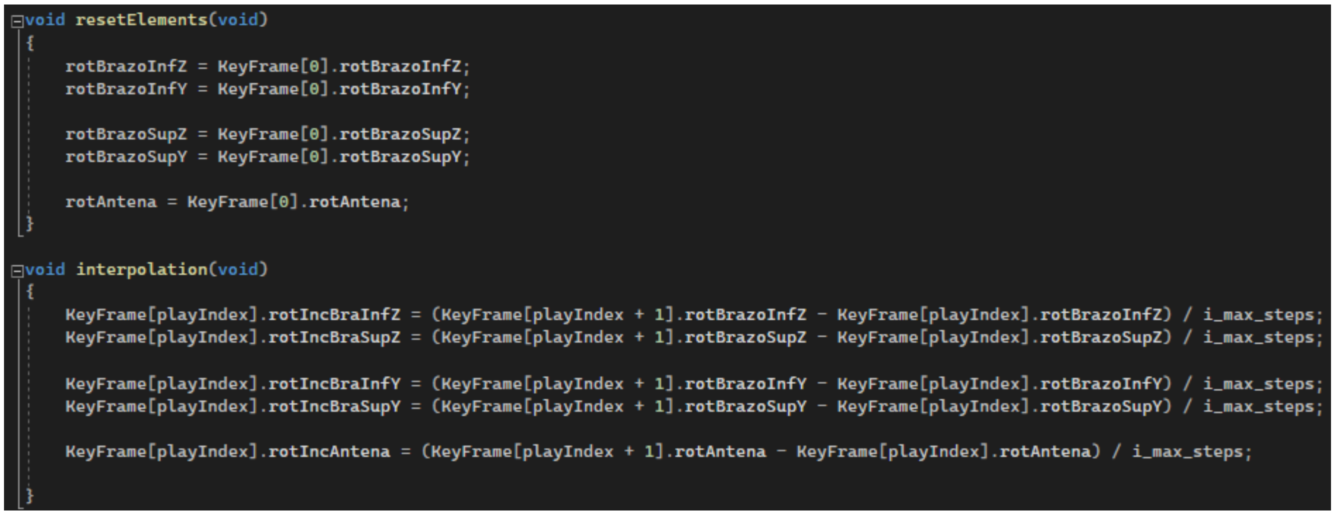
Keyframes

It was implemented in the antenna where the sequence is activated by keyboard:

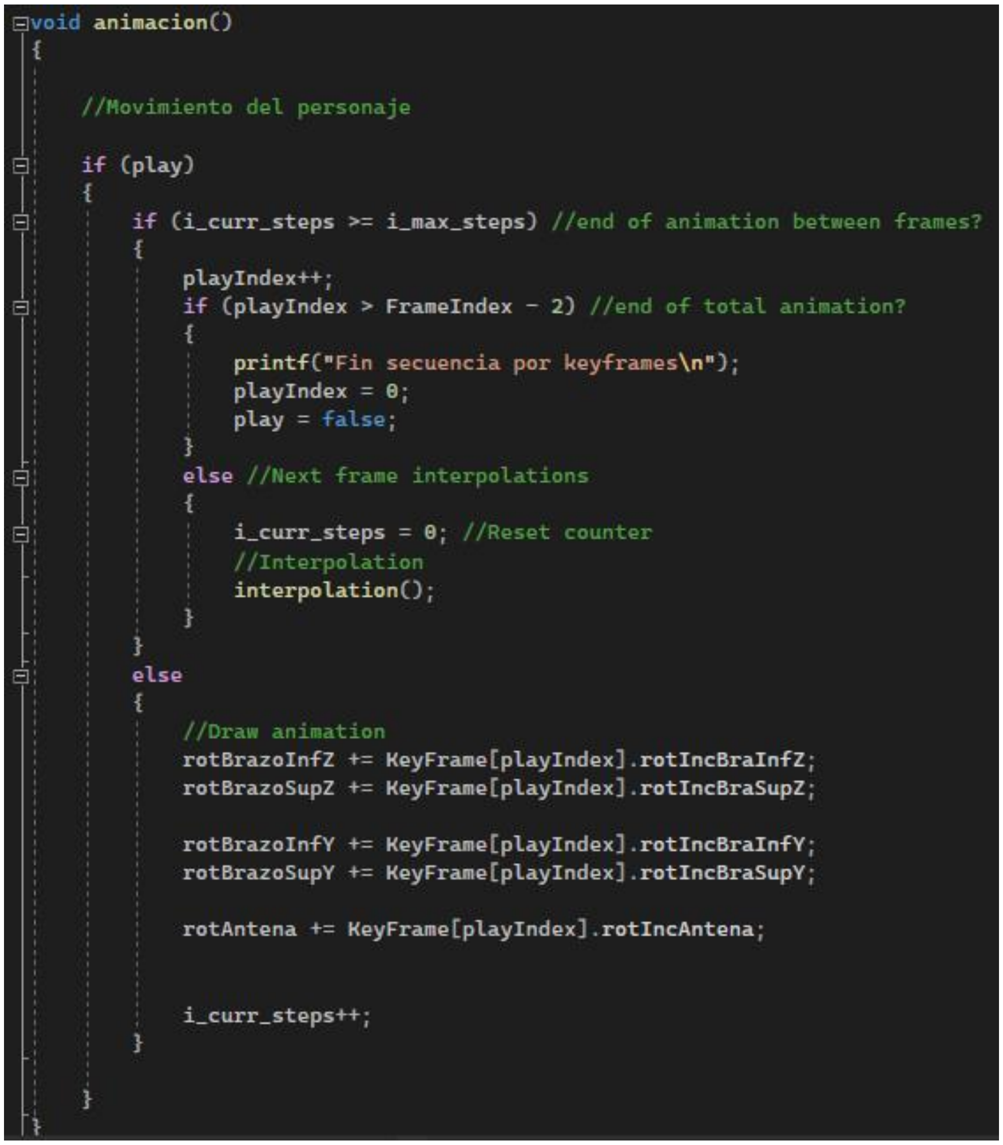


```
//Animacion por keyfames ANTENA
if (!banAntena && mainWindow.getAnimKeyFrames() == true)
{

    if (play == false && (FrameIndex > 1))
    {
        resetElements();
        //First Interpolation
        interpolation();

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

    banAntena = true;
}
if (mainWindow.getAnimKeyFrames() == false) {
    banAntena = false;
}
```

For this, the following functions were used:

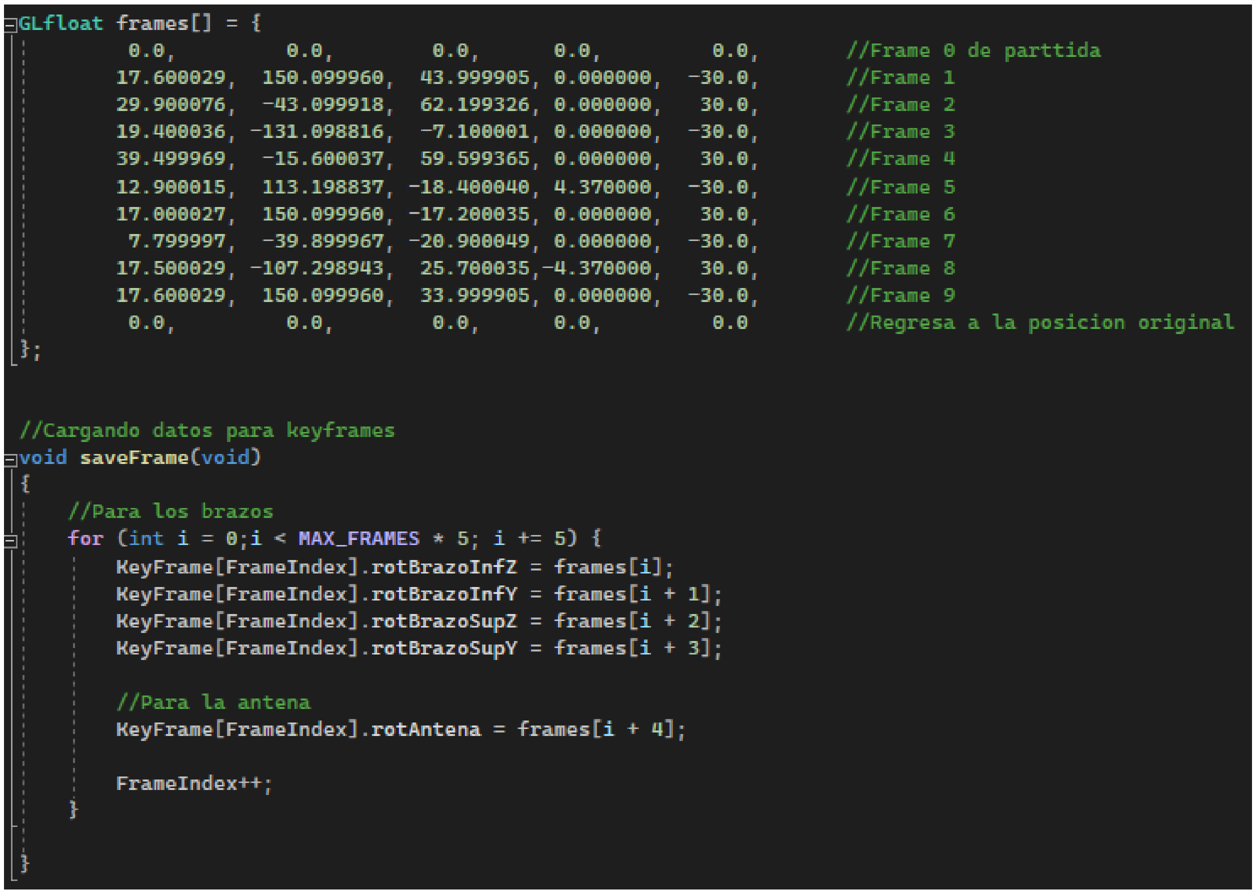


```
void resetElements(void)
{
    rotBrazoInfZ = KeyFrame[0].rotBrazoInfZ;
    rotBrazoInfY = KeyFrame[0].rotBrazoInfY;

    rotBrazoSupZ = KeyFrame[0].rotBrazoSupZ;
    rotBrazoSupY = KeyFrame[0].rotBrazoSupY;

    rotAntena = KeyFrame[0].rotAntena;
}

void interpolation(void)
{
    KeyFrame[playIndex].rotIncBraInfZ = (KeyFrame[playIndex + 1].rotBrazoInfZ - KeyFrame[playIndex].rotBrazoInfZ) / i_max_steps;
    KeyFrame[playIndex].rotIncBraSupZ = (KeyFrame[playIndex + 1].rotBrazoSupZ - KeyFrame[playIndex].rotBrazoSupZ) / i_max_steps;

    KeyFrame[playIndex].rotIncBraInfY = (KeyFrame[playIndex + 1].rotBrazoInfY - KeyFrame[playIndex].rotBrazoInfY) / i_max_steps;
    KeyFrame[playIndex].rotIncBraSupY = (KeyFrame[playIndex + 1].rotBrazoSupY - KeyFrame[playIndex].rotBrazoSupY) / i_max_steps;

    KeyFrame[playIndex].rotIncAntena = (KeyFrame[playIndex + 1].rotAntena - KeyFrame[playIndex].rotAntena) / i_max_steps;

}
```



```
void animacion()
{


    //Movimiento del personaje

    if (play)
    {
        if (i_curr_steps >= i_max_steps) //end of animation between frames?
        {
            playIndex++;
            if (playIndex > FrameIndex - 2) //end of total animation?
            {
                printf("Fin secuencia por keyframes\n");
                playIndex = 0;
                play = false;
            }
            else //Next frame interpolations
            {
                i_curr_steps = 0; //Reset counter
                //Interpolation
                interpolation();
            }
        }
        else
        {
            //Draw animation
            rotBrazoInfZ += KeyFrame[playIndex].rotIncBraInfZ;
            rotBrazoSupZ += KeyFrame[playIndex].rotIncBraSupZ;

            rotBrazoInfY += KeyFrame[playIndex].rotIncBraInfY;
            rotBrazoSupY += KeyFrame[playIndex].rotIncBraSupY;

            rotAntena += KeyFrame[playIndex].rotIncAntena;


            i_curr_steps++;
        }

    }
}
```

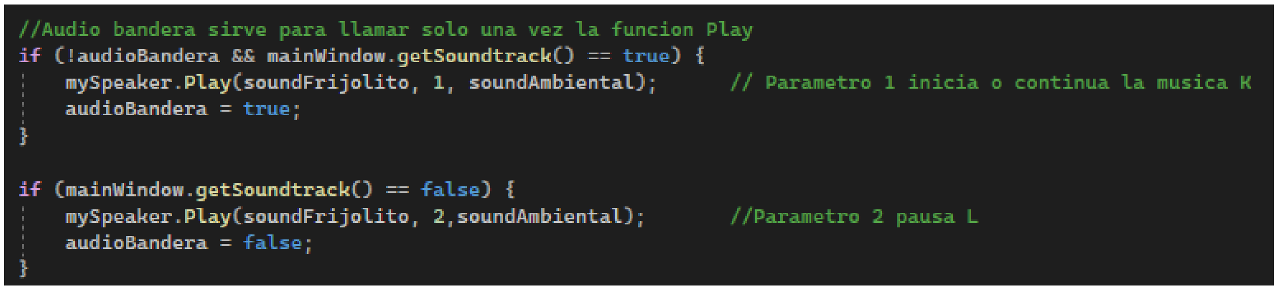
To save the keyframes, the function was modified to load the data directly from an array:



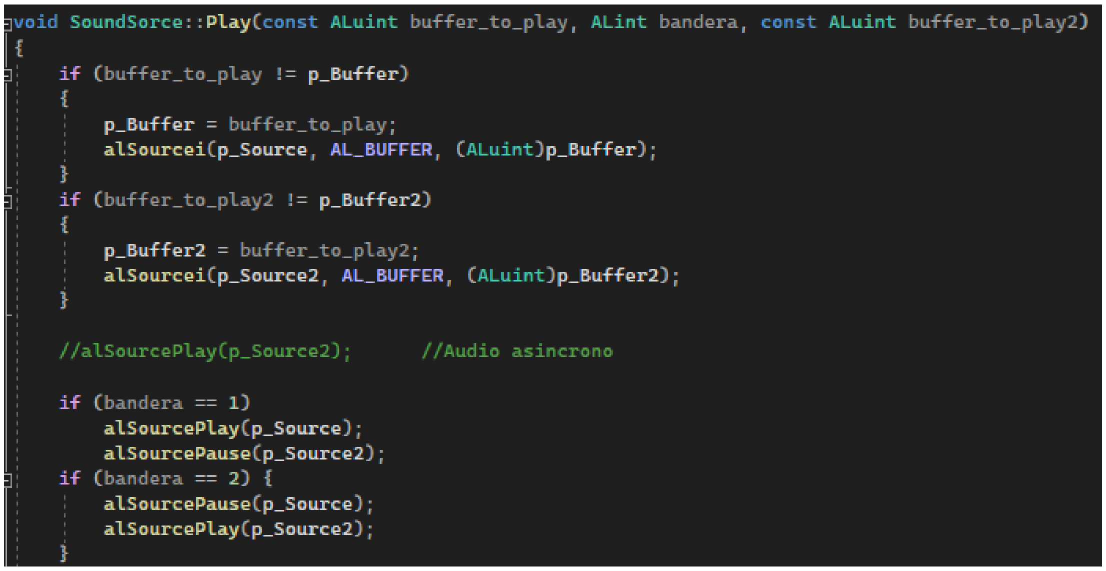
```
GLfloat frames[] = {
         0.0,         0.0,        0.0,      0.0,         0.0,       //Frame 0 de parttida
        17.600029,  150.099960,  43.999905, 0.000000,  -30.0,       //Frame 1
        29.900076,  -43.099918,  62.199326, 0.000000,   30.0,       //Frame 2
        19.400036, -131.098816,  -7.100001, 0.000000,  -30.0,       //Frame 3
        39.499969,  -15.600037,  59.599365, 0.000000,   30.0,       //Frame 4
        12.900015,  113.198837, -18.400040, 4.370000,  -30.0,       //Frame 5
        17.000027,  150.099960, -17.200035, 0.000000,   30.0,       //Frame 6
         7.799997,  -39.899967, -20.900049, 0.000000,  -30.0,       //Frame 7
        17.500029, -107.298943,  25.700035,-4.370000,   30.0,       //Frame 8
        17.600029,  150.099960,  33.999905, 0.000000,  -30.0,       //Frame 9
         0.0,         0.0,        0.0,      0.0,         0.0        //Regresa a la posicion original
};


//Cargando datos para keyframes
void saveFrame(void)
{
    //Para los brazos
    for (int i = 0;i < MAX_FRAMES * 5; i += 5) {
        KeyFrame[FrameIndex].rotBrazoInfZ = frames[i];
        KeyFrame[FrameIndex].rotBrazoInfY = frames[i + 1];
        KeyFrame[FrameIndex].rotBrazoSupZ = frames[i + 2];
        KeyFrame[FrameIndex].rotBrazoSupY = frames[i + 3];

        //Para la antena
        KeyFrame[FrameIndex].rotAntena = frames[i + 4];

        FrameIndex++;
    }

}
```

Audio

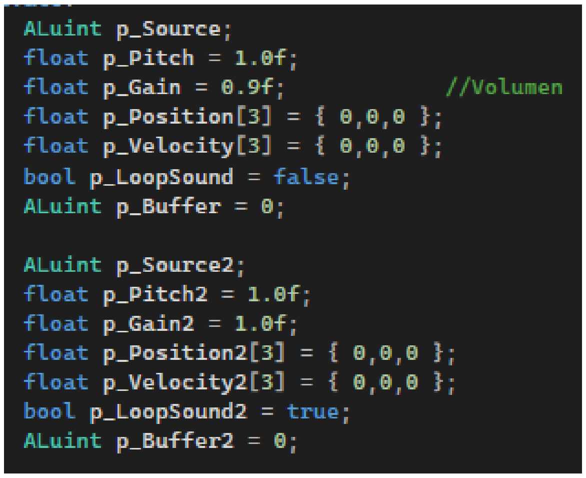
The OpenAL library was used for two implemented audios, environmental sound is the one that is played from the beginning and corresponds to the melody of Pokémon, this, the second is a spatial type audio for which the K key was adapted to activate it and the letter L to stop it that corresponds to the theme of a lot of struggle, each audio has its own buffer and has different characteristics configured such as loop playback and volume.



```
//Audio bandera sirve para llamar solo una vez la funcion Play
if (!audioBandera && mainWindow.getSoundtrack() == true) {
    mySpeaker.Play(soundFrijolito, 1, soundAmbiental);      // Parametro 1 inicia o continua la musica K
    audioBandera = true;
}

if (mainWindow.getSoundtrack() == false) {
    mySpeaker.Play(soundFrijolito, 2,soundAmbiental);       //Parametro 2 pausa L
    audioBandera = false;
}
```



```
void SoundSorce::Play(const ALuint buffer_to_play, ALint bandera, const ALuint buffer_to_play2)
{
    if (buffer_to_play != p_Buffer)
    {
        p_Buffer = buffer_to_play;
        alSourcei(p_Source, AL_BUFFER, (ALuint)p_Buffer);
    }
    if (buffer_to_play2 != p_Buffer2)
    {
        p_Buffer2 = buffer_to_play2;
        alSourcei(p_Source2, AL_BUFFER, (ALuint)p_Buffer2);
    }

    //alSourcePlay(p_Source2);      //Audio asincrono

    if (bandera == 1)
        alSourcePlay(p_Source);
        alSourcePause(p_Source2);
    if (bandera == 2) {
        alSourcePause(p_Source);
        alSourcePlay(p_Source2);
    }
```



```
ALuint p_Source;
float p_Pitch = 1.0f;
float p_Gain = 0.9f;            //Volumen
float p_Position[3] = { 0,0,0 };
float p_Velocity[3] = { 0,0,0 };
bool p_LoopSound = false;
ALuint p_Buffer = 0;

ALuint p_Source2;
float p_Pitch2 = 1.0f;
float p_Gain2 = 1.0f;
float p_Position2[3] = { 0,0,0 };
float p_Velocity2[3] = { 0,0,0 };
bool p_LoopSound2 = true;
ALuint p_Buffer2 = 0;
```

Both sounds are played from the same output device.

Comments:

Miguel: Within the development of this project, the knowledge acquired throughout the course was applied, both from the practical and the theoretical part, the project created for the laboratory was taken as a base, implementing some changes and improvements, among which the implementation of the changing cameras, correcting animations, adding new models and animations, as well as implementing both environmental and special effect audio.

Among the different software and tools that were used, the use of 3D Max for the creation of the models and the application of the texture stands out, as well as GIMP for editing images to be used as textures for the models, as well as the implementation of the OpenAL library for handling audio within the virtual environment.

Regarding the result, I feel very happy and satisfied with the result we achieved, in addition to covering the different requirements requested, I loved the result, although the initial proposal was not this way, the way in which they were combined the different universes seemed correct to me. I really liked the way we worked as a team, having different laboratory professors we were able to complement our knowledge and provide solutions to each of the difficulties that arose along the way.

Arturo: During the preparation of this project, theoretical and practical knowledge acquired both in theory class and in the laboratory was put into practice. In my case, it was difficult to adapt to the code structure that my partner had, since I was in a different laboratory group (with Professor Aldair), seeing various .h and .cpp files puzzled me at first, but as I analyzed and progressed in inserting my models, animations, making changes such as the implementation of cameras and audio, I became quite familiar. It also helps to see the changes or animations that my partner already had implemented to better understand the structure of the code, in this case the change from day to night of the skybox and the implementation of lighting do not bring any change other than moving place some lamps to test and I just limited myself to analyze how they had been implemented.

For the models I used the Maya software where little by little it was easier for me to create objects as I learned new tools and when texturing I put into practice several tips learned in the laboratory to achieve better results, the use of GIMP to create textures was also very useful, but complex to learn, especially for elements that did not have a uniform shape, such as frijolito.

Regarding the result of the project, I feel very satisfied with what we did and how each one of us contributed changes, improvements and solutions to the problems that arose, although at the beginning of the project when we planned what it would be like, I did not imagine it this way. if it met my expectations in terms of combining universes.

Bibliography

Downloaded Models

Lucario
Author: poke master
License: Personal use
Origin: Free 3D
Download link: https://free3d.com/es/modelo-3d/lucario-pokemon-64994.html

Streetlight
Author: tyrosmith
License: Personal use

Origin: Free 3D
Download link: https://free3d.com/es/modelo-3d/street-light-lamp-61903.html

Post
Author: koraybeybozkurt
License: Personal use
Origin: Free 3D
Download link: https://free3d.com/es/modelo-3d/street-lamb-317863.html

Water source
Author: alexalphagame1
License: Personal use
Origin: cgtrader
Download link: https://www.cgtrader.com/free-3d-models/architectural/architectural-street/fountain-b091febb-45d6-4f80-b53a-0200fae8a558

Slide
Author: Krammer Peter
License: Personal use
Origin: Free 3d Models
Download link: https://archive3d.net/?a=download&id=73e19444

Trees and shrubs
Author: Flamazilla
License: Personal use
Origin: cgtrader
Download link: https://www.cgtrader.com/free-3d-models/exterior/landscape/low-poly-forest-nature-set-free-trial

Park games
Author: Deshan
License: Personal use
Origin: sketchfab
Download link: https://skfb.ly/6RpYK

Textures

Wall
Author: ArthurHidden
License: Personal use
Origin: freepik
Download link: https://www.freepik.es/foto-gratis/fondo-textura-pared_11176149.htm

Metal
Author: freepik
License: Personal use
Origin: freepik
Download link: https://www.freepik.es/foto-gratis/primer-plano-fondo-abstracto-metal_12558749.htm

Skybox Night
Author: majorhood410
License: Personal use
Origin: freepng.es
Download link: https://www.freepng.es/png-baq9l8/download.html

Skybox Day
Author: Unknown
License: Personal use
Origin: freepng.es
Download link: https://www.freepng.es/png-o8w4e2/

Mordecai
Author: senjen
License: Personal use
Origin: models-resource
Download link: https://www.models-resource.com/browser_games/fusionfallheroes/model/6725/

Variety of textures for frijolito models:
Author: Unknown
License: Educative use
Origin: Ambientcg
Download link: https://ambientcg.com/lis

Music:
Much Fight | *Neighborhood boys*
WMG (on behalf of MCM Mexico); BMI - Broadcast Music Inc., LatinAutorPerf, UMPI, UMPG Publishing.
https://www.youtube.com/watch?v=BZAkzCw17GM&ab_channel=ChicosdeBarrio-Topic

Pokémon Intro
Author: Unknown
License: Free
Origin: Free MP3 Sounds

Download link:
http://sonidosmp3gratis.com/download.php?id=17146&sonido=pokemon%204