



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE INGENIERÍA ELÉCTRICA

INGENIERÍA EN COMPUTACIÓN

LABORATORIO DE COMPUTACIÓN GRÁFICA e
INTERACCIÓN HUMANO COMPUTADORA



MANUAL TÉCNICO

NOMBRES COMPLETOS - N° de Cuenta:

Arroyo Quiroz José Miguel - 317016136

Hernández Hernández Cristian - 317000234

GRUPOS DE LABORATORIO: 01

GRUPOS DE TEORÍA: 04

SEMESTRE 2023-2

FECHA DE ENTREGA LÍMITE: 28 de mayo, 2023

CALIFICACIÓN: _____

Manual Técnico

Elementos Incluidos dentro del escenario:

- Geometría

La estructura principal del escenario está formada por diferentes edificios los cuales están inspirados en los modelos de los juegos clásicos de Pokémon de 3ra generación, estos edificios están ubicados, de tal forma que rodean una zona central en la cual está ubicado el parque perteneciente al mundo de un show más.

Dichos modelos se crearon a través del software de modelado 3D Max 2023, posteriormente fueron exportados en formato obj, para su textura se usaron imágenes del juego “Pokémon Edición Verde Hoja y Rojo Fuego”.

A continuación, se presentan las diferentes estructuras y el resultado final ya en el ambiente virtual:

Laboratorio Oak



Casa



Centro Pokémon



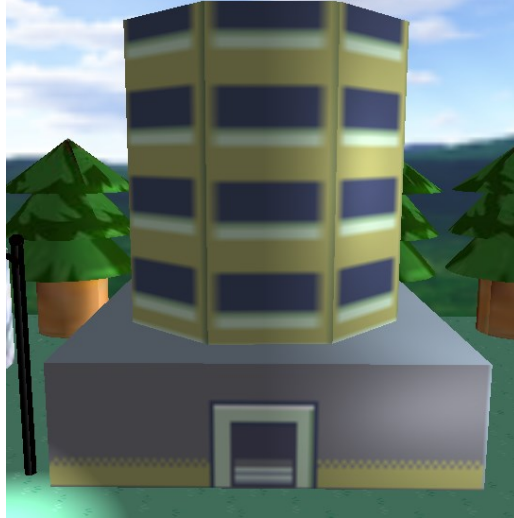
Tienda Pokémon



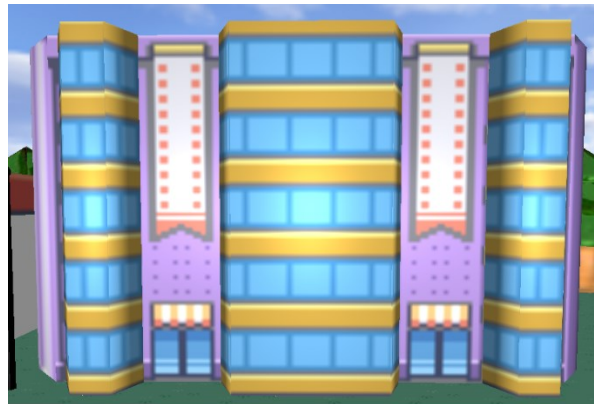
Gimnasio



Torre Lavanda



Centro Comercial



Los anteriores modelos fueron exportados en conjunto bajo el nombre de “PuebloCentro.obj”

Lucario

Este modelo fue descargado de internet, al descargar el modelo incluye imágenes para texturizar el personaje, dichas imágenes fueron editadas y optimizadas para poder utilizarse.



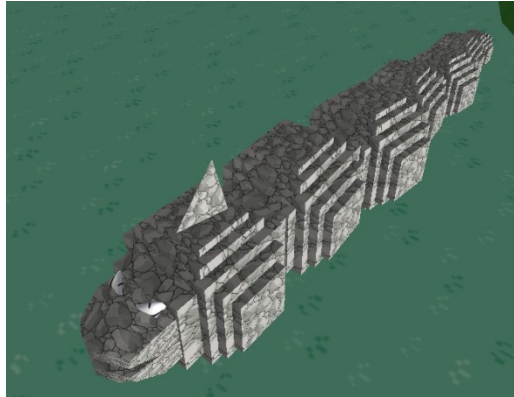
Voltorb



Electrode



Onix



Otros: Arboles a los alrededores, farolas y postes.



Los elementos que fueron exportados al entorno por separado son aquellos que tienen una interacción especial con el entorno como iluminación o animación.

Carga de los Diferentes modelos

```
Pueblo = Model();
Pueblo.LoadModel("Models/PuebloCentro.obj");
Arboles = Model();
Arboles.LoadModel("Models/Arboles.obj");
Vultorb = Model();
Vultorb.LoadModel("Models/Vultorb.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");
```

Colocación en el escenario

```
//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();
```

```
//Vultorb
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));
Vultorb.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();

```

Casa del Parque

La estructura principal y más llamativa del parque es la casa. Este modelo se realizó con el programa de modelado Blender y para texturizarlo se utilizaron imágenes de la serie en la cual se observará de mejor manera los detalles de la casa. Posteriormente se crearon los archivos correspondientes para ser colocados dentro de la carpeta de modelos, así como la textura previamente optimizada.

Dentro del código se carga el modelo.

```

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

```

Posteriormente trasladamos y escalamos la casa para colocarlo al centro del escenario

```

//#####
//#### 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();

```

Fuente

La fuente es un modelo obtenido de la red, pero como el texturizado del agua no la poseía entonces se decidió colocarle una. De igual forma, el modelo se carga al programa y posteriormente se realizan las transformaciones de traslación y escalado para quedar más acorde al tamaño de los demás objetos.

Carga al programa.

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

Puesta en escenario.

```
//#####  
//#### 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();
```

Área de la dulcería: Sillas, Mesas y Botes de basura

Para esta área se decidió obtener los modelos de la silla y mesa de internet. Se crearon manualmente los botes de basura y la estructura de la tienda. Con estos modelos se creó un mini escenario con ayuda de los programas de modelado para poder cargar el conjunto de objetos al programa. Todos los modelos fueron texturizados manualmente ya que tanto la silla como la mesa, no contaban con las imágenes.

Carga de modelo del área.

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

Vista en escenario.

```
//#####  
//#### 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();
```

Área infantil: Tobogán, columpios, rampa sube y baja.

Al igual que el área de la dulcería también para el área infantil también se generó un mini escenario con los modelos de los juegos. Los columpios y las rampas fueron separados del modelo base para poder ser utilizados como objeto de animación y así moverse independientemente.

Carga de modelos.

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

Puesta en escena.

```
//#####
//#### 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();

```

Otros: Reja, entrada, árboles y arbustos del parque

En el caso de la reja, su objetivo es cubrir el perímetro del parque y se colocaron de forma jerárquica para saber la posición más fácilmente de la última reja también se utilizó para colocar la entrada. Los árboles y arbustos sólo fueron colocados dentro del parque.

```

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");

```

Puesta en escena de los modelos.

```

//#####
//#### 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();

```

Para el caso de los árboles y arbustos, sólo se colocará algunos ejemplos.

```
//#####//
//#### 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();
```

Arbustos.

```
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();

```

Personaje Secundario: Mordecai

El personaje secundario es Mordecai. Se utilizó un modelo de internet para representar el personaje. Se encuentra situado frente a la casa del parque.

```

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

```

En escena.

```

//#####//
//#### 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();

```

Vista de los objetos en la ejecución del programa:





- Avatar

Como ya se mencionó anteriormente, este modelo fue descargado de internet, el modelo contenía una textura por defecto, pero dicha textura no podía ser exportada a OpenGL, así que se volvió a texturizar usando las texturas que nos proporcionó el autor del modelo.

Modelo



Creación Jerárquica

Del modelo original descargado, se tuvo que editar, con el fin de separar cada una de sus extremidades para ser importada por separado y así poder controlar cada uno de estos elementos por separado.

```
//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, 0.0f, 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, 45 * 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, 45 * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
LucBraIzq.RenderModel();

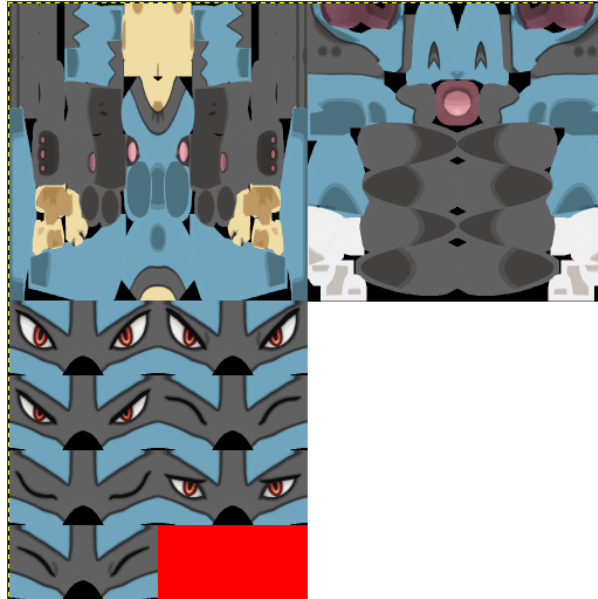
//Pierna Derecha
model = glm::mat4(1.0);
model = modelLuc;
//model = glm::rotate(model, 45 * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
LucPierDer.RenderModel();

//Pierna Izquierda
model = glm::mat4(1.0);
model = modelLuc;
//model = glm::rotate(model, 45 * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
LucPierIzq.RenderModel();

```

Textura

La imagen creada y optimizada para el correcto texturizado del personaje se muestra a continuación:



Animación

La animación del avatar consiste en recorrer el escenario por afuera del parque en un ciclo consecutivo.

```
//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;
    }
}
```

La primera parte del código permite al modelo ir moviendo alguna extremidad de un lado a otro, independientemente de hacia dónde este caminando el avatar.

```

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;
    }
}
}

```

La segunda parte de la animación le permite al avatar recorrer el escenario en bucle, todas estas animaciones se activan al momento de presionar la tecla O y se desactivan al presionar la tecla P.

- **Recorrido**

Para esta parte de los requerimientos, únicamente fue posible implementar la cámara en tercera persona, la cual se encuentra ligada al plano XZ, para lograr esto, fue necesario cambiar la declaración original de la cámara y posteriormente al momento de pasar los parámetros del control del mouse, solo permitir el giro hacia los lados.



- **Iluminación**

Iluminación de tipo puntual:

Para este punto se crearon tres lámparas de tipo poste, las cuales fueron ubicadas y repartidas alrededor del parque, estas luces prenden y apagan automáticamente al cambiar a la noche.

```
//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;
}
```

Para esta implementación, se cambia el contador de luces de tipo Puntual, cuando es de día se coloca en 0 para que todas estén apagadas y cuando se hace de noche el contador se coloca en 3 para mostrar todas las luces.



Iluminación de tipo reflector:

Estas luces se encuentran situadas en las 3 farolas que se encuentran fuera del parque, activándose cuando se presiona la tecla Z y desactivándose con la tecla X. Todas estas fuentes de luz alumbran hacia el piso.

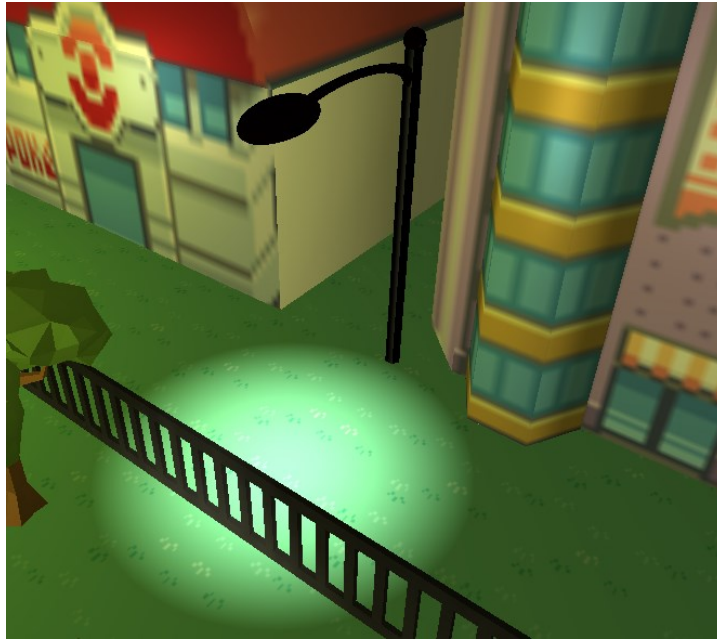
```
unsigned int spotLightCount = 0;
spotLights[0] = SpotLight(1.0f, 1.0f, 1.0f,
    1.0f, 0.1f,
    0.0f, 20.0f, -68.0f,
    0.0f, -1.0f, 0.0f,
    1.0f, 0.0f, 0.0f,
    30.0f);
spotLightCount++;

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++;
```


Se utiliza una bandera para saber que tecla se está oprimiendo.

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



Iluminación del SkyBox:

El transcurso del día y noche tiene periodo de un minuto. Cada que pasa el minuto la bandera llamada 'dia' se alterna entre verdadero y falso. Cuando esto sucede, el skybox cambia su textura de pasar a un cielo iluminado a uno oscuro

```
//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;  
}
```

Además de cambiar el fondo, cambiará la intensidad de la luz direccional y también las luces de tipo pointlight no se renderizarán para permanecer apagados durante el día y encendidos durante la noche.

Código de cambio.

```
//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;
}
```

Cambio de las texturas en el skybox:

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

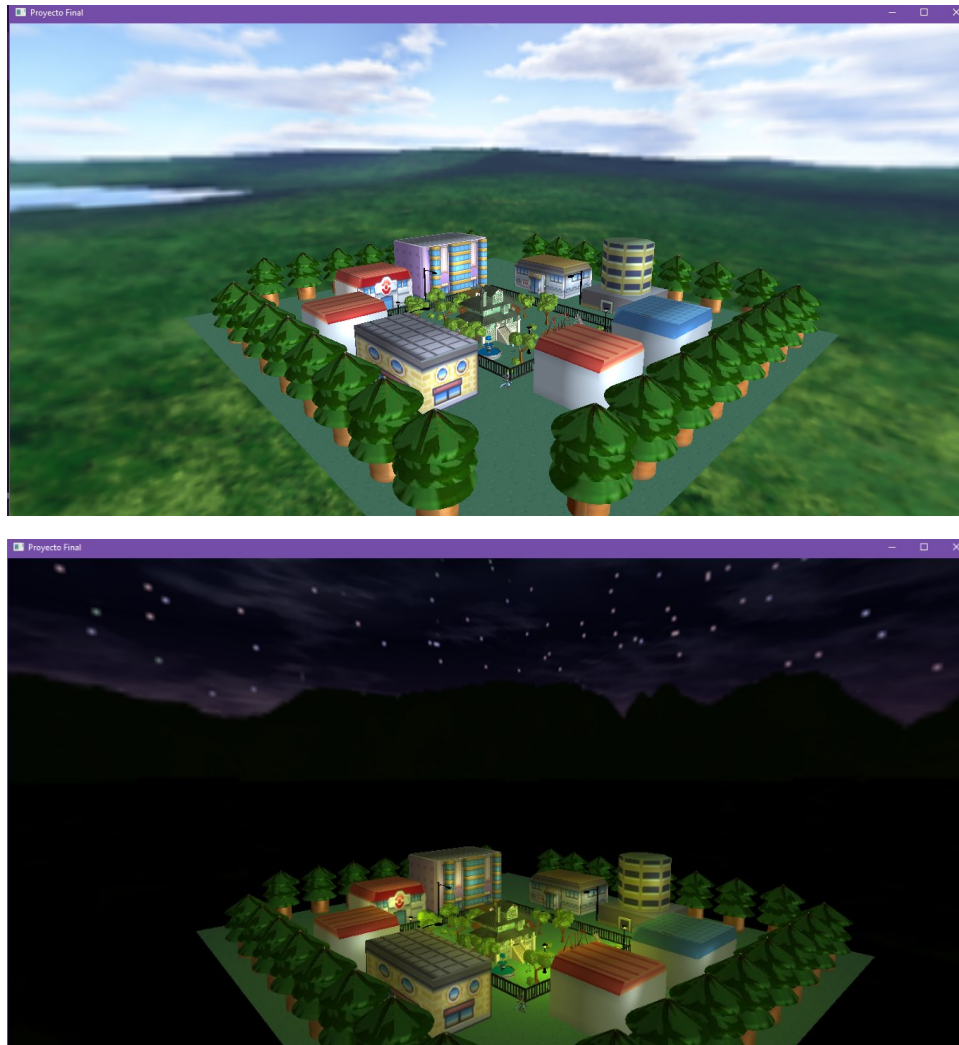
skyboxFacesDia.push_back("Textures/Skybox/Day-Skybox_rt.tga");
skyboxFacesDia.push_back("Textures/Skybox/Day-Skybox_lf.tga");
skyboxFacesDia.push_back("Textures/Skybox/Day-Skybox_dn.tga");
skyboxFacesDia.push_back("Textures/Skybox/Day-Skybox_up.tga");
skyboxFacesDia.push_back("Textures/Skybox/Day-Skybox_bk.tga");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

Texturas del SkyBox mostradas en ejecución



- **Animación**

Simple

Para las animaciones simples, tenemos el movimiento de los columpios y el movimiento de las rampas de sube y baja, así como el movimiento de un Pokémon. Todas estas animaciones se activan utilizando la tecla 'O' y se detienen con la tecla 'P'.

Movimiento de los columpios:

Los columpios rotan dando un efecto de balanceo y además es incremental a medida que pasa el tiempo, es decir, aumenta su altura con la que se rota. Se repetirá el movimiento hasta que se detenga la animación.

```
// 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 inicial
    if (rotColumpio < -0.1f) {
        rotColumpio += rotColumpioOffset * deltaTime;
    }
    else if (rotColumpio > 0.1f) {
        rotColumpio -= rotColumpioOffset * deltaTime;
    }
    incRot = 0.0f;
}
}
```

Movimiento de la rampa de sube y baja:

La rampa rota en un eje central y cada que llega al suelo se vuelve a regresar hasta que se detenga la animación.

```
// 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 inicial
    if (rotSyB < -0.1f) {
        rotSyB += rotColumpioOffset * deltaTime;
    }
    else if (rotColumpio > 0.1f) {
        rotSyB -= rotColumpioOffset * deltaTime;
    }
}
}
```

Movimiento de un Pokémon:

Los personajes de Voltorb y Electrode, hacen una animación en la cual saltan y giran a la vez en bucle.

```
//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;
}
```

Compleja

Textura con movimiento:

Se utilizó una textura en forma de nube de humo que saldrá por la chimenea de la casa del parque. Dicha textura se trasladará y rotará para dar un efecto de que sale humo de la chimenea.

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

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

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

```

Comentarios

En la creación de este proyecto tuvimos varios inconvenientes en diversas actividades, como lo fue en la animación al momento de crear la lógica de programación, en el modelado puesto que algunos objetos no contaban con texturizado, así como su creación, también está la cuestión de la iluminación direccional y su relación con el cambio del skybox. También hubo algunos problemas al colocar elementos en escena. Finalmente se puede mencionar que se aplicó cada uno de los elementos vistos en el laboratorio que abarcan desde la creación de primitivas hasta las animaciones de los objetos.

Miguel: Dentro del desarrollo y creación de este proyecto se tuvo en cuenta las ideas del boceto original y de la propuesta, sin embargo, durante el desarrollo del proyecto surgieron algunas complicaciones tales como el funcionamiento de las extremidades del avatar, las cuales no tenía un correcto pivote y, por lo tanto, no giraban en el sentido que debería razón por la cual no se pudo implementar una animación más realista.

Otro de los inconvenientes fue el manejo de la cámara, al ser uno de los elementos menos empleados a lo largo del proyecto no se pudo lograr ese cambio entre cámaras a pesar de tener las características referentes a cada una de ellas, por ello se optó por dejar solo la cámara ligada al plano.

Aun con todas estas dificultades, consideró que se mostró los conocimientos adquiridos a lo largo del curso, implementando la mayoría de los temas observados en clase.

Bibliografía

Modelos Descargados

Lucario

Autor: poke master

Licencia: Uso personal

Procedencia: Free 3D

Link de Descarga: <https://free3d.com/es/modelo-3d/lucario-pokemon-64994.html>

Farola

Autor: tyrosmith

Licencia: Uso personal

Procedencia: Free 3D

Link de Descarga: <https://free3d.com/es/modelo-3d/street-light-lamp-61903.html>

Poste

Autor: koraybeybozkurt

Licencia: Uso personal

Procedencia: Free 3D

Link de Descarga: <https://free3d.com/es/modelo-3d/street-lamb-317863.html>

Fuente de agua

Autor: alexalphagame1

Licencia: Uso personal

Procedencia: cgtrader

Link de Descarga: <https://www.cgtrader.com/free-3d-models/architectural/architectural-street/fountain-b091febb-45d6-4f80-b53a-0200fae8a558>

Tobogan

Autor: Krammer Peter

Licencia: Uso personal

Procedencia: Free 3d Models

Link de Descarga: <https://archive3d.net/?a=download&id=73e19444>

Árboles y arbustos

Autor: Flamazilla

Licencia: Uso personal

Procedencia: cgtrader

Link de Descarga: <https://www.cgtrader.com/free-3d-models/exterior/landscape/low-poly-forest-nature-set-free-trial>

Juegos de parque

Autor: Deshan

Licencia: Uso personal

Procedencia: sketchfab

Link de Descarga: <https://skfb.ly/6RpYK>

Texturas

Pared

Autor: ArthurHidden

Licencia: Uso personal

Procedencia: freepik

Link de Descarga: https://www.freepik.es/foto-gratis/fondo-textura-pared_11176149.htm

Metal

Autor: freepik

Licencia: Uso personal

Procedencia: freepik

Link de Descarga: https://www.freepik.es/foto-gratis/primer-plano-fondo-abstracto-metal_12558749.htm

Skybox Noche

Autor: majorhood410

Licencia: Uso personal

Procedencia: freepng.es

Link de Descarga: <https://www.freepng.es/png-baq9l8/download.html>

Skybox Día

Autor: desconocido

Licencia: Uso personal

Procedencia: freepng.es

Link de Descarga: <https://www.freepng.es/png-o8w4e2/>

Mordecai

Autor: senjen

Licencia: Uso personal

Procedencia: models-resource

Link de Descarga: https://www.models-resource.com/browser_games/fusionfallheroes/model/6725/