UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE INGENIERÍA ELÉCTRICA

INGENIERÍA EN COMPUTACIÓN

LABORATORIO DE COMPUTACIÓN GRÁFICA e INTERACCIÓN HUMANO COMPUTADORA

Documentación técnica proyecto final : Campo de comida Nacional e Internacional

Nombre completo:

- Salazar Serrano Edgar

  N° de cuenta: 416101630

  Grupo de Laboratorio: 10

  Grupo de teoría: 6

- Olvera Martínez Rodrigo Iván

  N° de cuenta: 314036731

  Grupo de Laboratorio: 02

  Grupo de teoría: 6

Semestre 2023-1

Objetivo

Utilizar las herramientas y los conocimientos adquiridos en el laboratorio de computación Gráfica e interacción humano computadora para recrear un escenario virtual usando el lenguaje de programación C++ y Open GL, esto con la temática de puestos de comida nacionales, en donde el usuario pueda recorrerlo libremente e interactuar con un avatar y algunos objetos que tienen una función especifica

Alcance

Se pretende recrear un espacio 3d que simule una feria virtual que incluye puestos de comida, personajes, luces y animaciones, interacción con el usuario por medio de teclas y adicionalmente agregamos efectos de sonido.

Descripción técnica

1. Creación de objetos y texturizado

Para la creación de nuestros objetos en Open GL fue posible hacerlo mediante carga de modelos prediseñados en software de modelado 3D y diseñados por nosotros con primitivas geométricas de forma jerárquica en OpenGL .

Para texturizar los modelos se hace mediante imágenes que se cargan por separado a los modelos y se declaran dentro del código, en el caso de los modelos ya hechos en software 3d, se tienen sus respectivas texturas y solo hay que cargarlos.

Declaración de modelos y texturas.

```
Model jimmy_cabeza;
Model jimmy_tronco;
Model pierna_izq;
Model pierna_izq_abajo;
Model pierna_der;
Model pierna_der_abajo;
Model brazo_izq;
Model brazo_der;
Model mano_der;
Model mano_izq;
```

```
Texture mariscos;
Texture puesto;
Texture mariscos_comida;
Texture helados_caja;
Texture tortas_rotulo;
Texture tortas_front;
Texture rotulo_represion;
Texture tacos_rotulo;
Texture tacos_front;
Texture tacos_rotulo_mini;
Texture jugos_rotulo;
Texture jugos_front;
Texture jugos_mini;
Texture fuego;
```

Se declaran tambien la ubicación de los modelos y las texturas

ejemplo:

```
jugos_rotulo = Texture("Textures/jugos_rotulo.png");
jugos_rotulo.LoadTextureA();
jugos_front = Texture("Textures/jugos_front.png");
jugos_front.LoadTextureA();
jugos_mini = Texture("Textures/jugos_rotulo_mino.png");
jugos_mini.LoadTextureA();
```

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

aku = Model();
aku.LoadModel("Models/Aku_Aku_READY.fbx");

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

modelos a recrear:

- puestos callejeros de comida
- puesto de tortas
- puesto de tacos
- puesto de jugos
- puestos mariscos
- camión de helados y camion de hot dogs
- cohete espacial
- balón de futbol
- pavimento y lámparas de calle

imagenes de referencia para los puestos y el camión de helados:

1- personajes
    - shrek
    - Jimmy Neutron (Avatar)

Imágenes de Referencia

Para crear los puestos los diseñamos con figuras geométricas hechas en OpenGL principalmente cubos, en los cuales se les modificó la escala para obtener rectángulos de diversos tamaños, asi creamos los puestos y el camión de helados de manera jerárquica para poder mover, rotar o escalar los objetos libremente sin que se pierda la forma original.

Definición de un cubo en open GL:

```
unsigned int cubo_indices[] = {
    // front
    0, 1, 2,
    2, 3, 0,
    // right
    4, 5, 6,
    6, 7, 4,
    // back
    8, 9, 10,
    10, 11, 8,

    // left
    12, 13, 14,
    14, 15, 12,
    // bottom
    16, 17, 18,
    18, 19, 16,
    // top
    20, 21, 22,
    22, 23, 20,
};
```

```
GLfloat cubo_vertices[] = {
    //x     y       z       S       T           NX      NY      NZ
    -0.5f, -0.5f,  0.5f,    0.0f, 0.0f,       0.0f,   0.0f,   -1.0f, //0
    0.5f, -0.5f,  0.5f,     1.0f,   0.0f,       0.0f,   0.0f,   -1.0f, //1
    0.5f,  0.5f,  0.5f,     1.0f,   1.0f,       0.0f,   0.0f,   -1.0f, //2
    -0.5f,  0.5f,  0.5f,    0.0f,   1.0f,       0.0f,   0.0f,   -1.0f, //3
    // right
    //x     y       z       S       T
    0.5f, -0.5f,  0.5f,     0.0f, 0.0f,       -1.0f,  0.0f,   0.0f,
    0.5f, -0.5f,  -0.5f,    1.0f,   0.0f,       -1.0f,  0.0f,   0.0f,
    0.5f,  0.5f,  -0.5f,    1.0f,   1.0f,       -1.0f,  0.0f,   0.0f,
    0.5f,  0.5f,  0.5f,     0.0f,   1.0f,       -1.0f,  0.0f,   0.0f,
    // back
    -0.5f, -0.5f, -0.5f,    0.0f,   0.0f,       0.0f,   0.0f,   1.0f,
    0.5f, -0.5f, -0.5f,     1.0f,   0.0f,       0.0f,   0.0f,   1.0f,
    0.5f,  0.5f, -0.5f,     1.0f,   1.0f,       0.0f,   0.0f,   1.0f,
    -0.5f,  0.5f, -0.5f,    0.0f,   1.0f,       0.0f,   0.0f,   1.0f,

    // left
    //x     y       z       S       T
    -0.5f, -0.5f,  -0.5f,   0.0f, 0.0f,       1.0f,   0.0f,   0.0f,
    -0.5f, -0.5f,  0.5f,    1.0f,   0.0f,       1.0f,   0.0f,   0.0f,
    -0.5f,  0.5f,  0.5f,    1.0f,   1.0f,       1.0f,   0.0f,   0.0f,
    -0.5f,  0.5f,  -0.5f,   0.0f,   1.0f,       1.0f,   0.0f,   0.0f,

    // bottom
    //x     y       z       S       T
    -0.5f, -0.5f,  0.5f,    0.0f, 0.0f,       0.0f,   1.0f,   0.0f,
    0.5f,  -0.5f,  0.5f,    1.0f,   0.0f,       0.0f,   1.0f,   0.0f,
     0.5f,  -0.5f,  -0.5f,  1.0f,   1.0f,       0.0f,   1.0f,   0.0f,
    -0.5f, -0.5f,  -0.5f,   0.0f,   1.0f,       0.0f,   1.0f,   0.0f,

    //UP
    //x     y       z       S       T
     -0.5f, 0.5f,  0.5f,    0.0f, 0.0f,       0.0f,   -1.0f,  0.0f,
      0.5f,  0.5f,  0.5f,   1.0f,   0.0f,       0.0f,   -1.0f,  0.0f,
       0.5f, 0.5f,  -0.5f,  1.0f,   1.0f,       0.0f,   -1.0f,  0.0f,
      -0.5f, 0.5f,  -0.5f,  0.0f,   1.0f,       0.0f,   -1.0f,  0.0f,
};
```

```
Mesh* obj6 = new Mesh();
obj6->CreateMesh(cubo_vertices, cubo_indices, 256, 96);
meshList.push_back(obj6);
```

Despues para instanciar los objetos se declara una matriz para el modelo, la cual tendrá la información de las transformaciones geométricas que le hagamos, además de que se le asignan las texturas usadas en cada modelo:

Ejemplo para instanciar un objeto:

```
// puesto  tortas

    color = glm::vec3(1.0f, 1.0f, 1.0f);
    model = glm::mat4(1.0);
    model = glm::translate(model, glm::vec3(-60.0f, 1.0f, -10.0f));
    modelaux = model;
    model = glm::scale(model, glm::vec3(5.0f, 2.0f, 3.0f) * plus);
    glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
    glUniform3fv(uniformColor, 1, glm::value_ptr(color));
    puesto.UseTexture();
    Material_brillante.UseMaterial(uniformSpecularIntensity, uniformShininess);
    meshList[5]->RenderMesh();
    // parte superior
    color = glm::vec3(1.0f, 1.0f, 1.0f);
    model = modelaux;
    model = glm::translate(model, glm::vec3(0.0f, 2.0f, 0.0f) * plus);
    model = glm::scale(model, glm::vec3(5.0f, 2.0f, 3.0f) * plus);
    glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
    glUniform3fv(uniformColor, 1, glm::value_ptr(color));
    tortas_rotulo.UseTexture();
    Material_brillante.UseMaterial(uniformSpecularIntensity, uniformShininess);
    meshList[5]->RenderMesh();
```

Siguiendo esta técnica tenemos objetos 3d creados por modelado geométrico y jerárquico

Instanciando el modelo de nuestro avatar

```
// ######################3      Jimmy Articulado ##############################3

//Personaje

//tonco
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(posX + 1.8f, -0.5f + posY, posZ + 1.9f));
//model = glm::translate(model, glm::vec3(1.8f, 0.0f,  1.9f));
model = glm::rotate(model, rotJimmy * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
//Timmy.UseMaterial(uniformSpecularIntensity, uniformShininess);
jimmy_tronco.RenderModel();
modelauxJimmy = model;
model = glm::translate(model, glm::vec3(0.052f, 0.45f, 2.6f));
model = glm::rotate(model, patear * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::rotate(model, rotRodDerS * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
pierna_der.RenderModel();
model = glm::translate(model, glm::vec3(-0.08f, -0.7f, 0.07f));
model = glm::rotate(model, rotRodDer * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
pierna_der_abajo.RenderModel();
model = modelauxJimmy;
model = glm::translate(model, glm::vec3(0.1, 0.537f, 1.731));
model = glm::rotate(model, rotRodIzqS * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
pierna_izq.RenderModel();
model = glm::translate(model, glm::vec3(-0.07f, -0.82f, -0.08));
model = glm::rotate(model, rotRodIzq * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
pierna_izq_abajo.RenderModel();
model = modelauxJimmy;
model = glm::translate(model, glm::vec3(0.0f, 2.65f, 3.0f));
model = glm::rotate(model, rotBraDerS * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::rotate(model, giroDer * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
brazo_der.RenderModel();
model = glm::translate(model, glm::vec3(-0.054f, -0.7f, 0.32f));
model = glm::rotate(model, rotBraDer * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::rotate(model, giroIzq * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
mano_der.RenderModel();
model = modelauxJimmy;
model = glm::translate(model, glm::vec3(0.05f, 2.65f, 1.4f));
model = glm::rotate(model, rotBraIzqS * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
brazo_izq.RenderModel();
model = glm::translate(model, glm::vec3(0.04f, -0.6f, -0.47f));
model = glm::rotate(model, rotBraIzq * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
mano_izq.RenderModel();
glDisable(GL_BLEND);

```

Para este modelo utilizamos Blender, que es un software de modelado 3d, para recortar las partes móviles del avatar que son brazos y piernas para que se puedan mover en la animación.

2. Iluminación

Para lograr que las formas creadas puedan verse y tengan efecto de profundidad se hace uso de la iluminación. Se tienen 3 tipos de iluminación: Point Light, Spot Light y Dirección light.

En este escenarios usamos una luz direccional (Direction Ligth) que funciona como luz ambiental por ejemplo, la luz del sol. Tambien usamos luces tipo SpotLight que funcionan como las luces de un faro.

- Declaración de Direction light:

Como parámetro se definió una luz blanca (codigo RGB = (1,1,1)) , la luz tiene una dirección negativa en el eje Y y negativa en el eje Z.

```
mainLight = DirectionalLight(1.0f, 1.0f, 1.0f,
    0.3f, 0.3f,
    0.0f, -1.0f, -1.0f);
```

- Declaración de Spotlight:

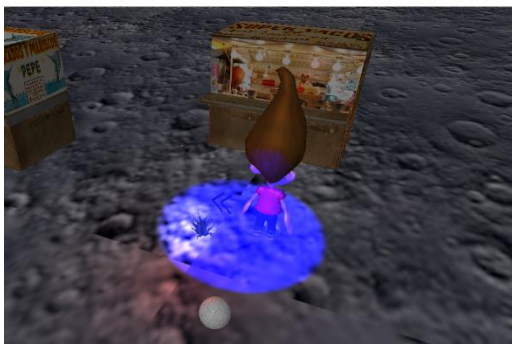
Se modifican los parámetros de color, dirección posición para la iluminación

```
spotLights[2] = SpotLight(1.0f, 1.0f, 1.0f,
    0.0f, 2.0f,
    0.0f, 0.0f, 0.0f,
    0.0f, -5.0f, 0.0f,
    1.0f, 0.0f, 0.0f,
    50.0f);
spotLightCount++;
```

Resultados de iluminación:

- Luz que acompaña el movimiento del coche de helados y show de luces.

```
//luz del carro de helado
    glm::vec3 helados(movCoche, 16.0f , 15.0f);
    spotLights[2].SetFlash(helados, glm::vec3(0.0f, -1.0f, 0.0f));
    spotLights[2].SetPos(helados);
```



3. Skybox

Se implementa un Skybox para poder representar el cielo en el escenario, en nuestro caso usamos un Skybox que representa al espacio.

Declaración de un skybox

```
std::vector<std::string> skyboxFaces;
skyboxFaces.push_back("Textures/Skybox/skyRt.tga");
skyboxFaces.push_back("Textures/Skybox/skyLf.tga");
skyboxFaces.push_back("Textures/Skybox/skyDwn.tga");
skyboxFaces.push_back("Textures/Skybox/skyUp.tga");
skyboxFaces.push_back("Textures/Skybox/skyBk.tga");
skyboxFaces.push_back("Textures/Skybox/skyFront.tga");

skybox = Skybox(skyboxFaces);
```

Resultado:

4. Animaciones

Usamos dos tipos de animación, una animación automática y animaciones que dependen de la entrada por teclado.

Estas son :

1. Jimmy (Avatar)
   Declaración de funciones en Window.h

```
GLfloat getrecorrido() { return recorrido; }
GLfloat getrotJimmy() { return rotJimmy; }
GLfloat getrotpatear() { return patear; }
```

Definición del comportamiento del avatar en Window.cpp

```
if (key == GLFW_KEY_UP)
{
    theWindow->rotJimmy = -90.0f;
}
if (key == GLFW_KEY_DOWN)
{
    theWindow->rotJimmy = 90.0f;
}
if (key == GLFW_KEY_RIGHT)
{
    theWindow->rotJimmy = 0.0f;
}
if (key == GLFW_KEY_LEFT)
{
    theWindow->rotJimmy = 180.0f;
}
```

Con las teclas de flechas el avatar comenzara a caminar en la dirección indicada.

```
}
if (key == GLFW_KEY_X)
{
    theWindow->recorrido = true;
}

if (key == GLFW_KEY_SPACE)
{
    theWindow->patear = true;
}
```

- Secuencia de Animación del Avatar:

```
//Recorrido Jimmy

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

            }
        }
        if (!auxMovJimmy)
        {
            rotRodDerS -= 0.4f;
            rotRodIzqS += 0.4f;
            rotRodIzq = 0.0f;
            rotRodDer -= 0.5f;
            rotBraDerS += 0.4f;
            rotBraIzqS -= 0.4f;
            if (rotRodDerS < -30)
            {
                auxMovJimmy = true;
            }
        }
        if (posZ < -7.7f)
        {
            rotRodDerS = 0.0f;
            rotRodIzqS = 0.0f;
            rotRodIzq = 0.0f;
            rotRodDer = 0.0f;
            rotBraDerS = 0.0f;
            rotBraIzqS = 0.0f;
            recorrido1 = false;
            SoundEngine->play2D("media/luces.mp3", false);

        }
    }
/*  if (movPuerta)
    {

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

            }
        }
        if (!auxMovJimmy)
        {

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

            }
        }
        if (!auxMovJimmy)
        {
            rotRodDerS -= 0.4f;
            rotRodIzqS += 0.4f;
            rotBraDerS += 0.4f;
            rotBraIzqS -= 0.4f;
            if (rotRodDerS < -20)
            {
                auxMovJimmy = true;
            }
        }
        if (posZ < -17.7f)
        {
            rotRodDerS = 0.0f;
            rotRodIzqS = 0.0f;
            rotBraDerS = 0.0f;
            rotBraIzqS = 0.0f;
            recorrido4 = false;
            /*engine->play2D("media/puertaCierra.wav",
                false, false, true);
            cerrarPuerta = true;*/
        }
    }
```

- Animación automática del modelo jerárquico. Este recorrerá el escenario hacia el frente y de reversa indefinidamente

```
// animacion coche
if (avanza) {
    if (!(movCoche > -300.0f))
        avanza = false;

    movCoche -= movOffset * deltaTime;
    //printf("avanza%f \n ",movCoche);
    rotllanta += rotllantaOffset * deltaTime;



}
else {
    if (!(movCoche < 300.0f))
        avanza = true;

    movCoche += movOffset * deltaTime;
    rotllanta -= rotllantaOffset * deltaTime;


}
```

Instancia del modelo (se envían los datos de movimiento como parámetro de la translación)

```
// Carro de helados
        //rectangulo 1 (caja)
        model = glm::mat4(1.0);
        color = glm::vec3(1.0f, 1.0f, 1.0f);
        model = glm::translate(model, glm::vec3(movCoche, 0.0f, 30.0f) ); //posicion
        modelaux = model;
        model = glm::translate(model, glm::vec3(0.5f, 0.25f, -5.0f) * plus);
        model = glm::scale(model, glm::vec3(2.5f, 1.25f, 1.0f) * plus); // escalando
        glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE
        glUniform3fv(uniformColor, 1, glm::value_ptr(color));
        Material_brillante.UseMaterial(uniformSpecularIntensity, uniformShininess);
        helados_caja.UseTexture();
        meshList[5]->RenderMesh(); // dibujando cubo
```

- Animación de Shrek

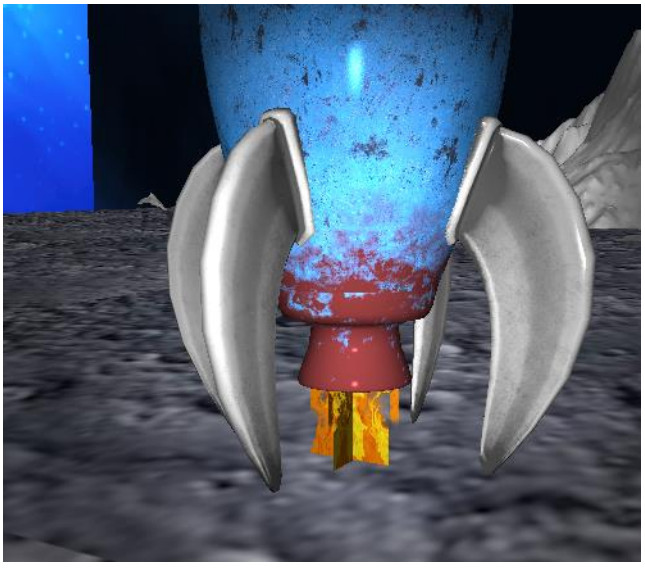
Se recortaron las partes del codo y hombro del modelo de shrek para simular el movimiento del brazo

```
//animación shrek

if (avanzaS) {
    if (!(movShrek > -70.0f))
        avanzaS = false;

    movShrek -= movOffset * deltaTime;
}
else {
    if (!(movShrek < 0.0f))
        avanzaS = true;

    movShrek += movOffset * deltaTime;
}
```

```
//##################  SHREK

model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-60, -2.0f, 3.5f));
modelaux=model;
model = glm::rotate(model, -90 * toRadians, glm::vec3(0.0f, 90.0f, 0.0f));
model = glm::scale(model, glm::vec3(12.0f, 12.0f, 12.0f));
model = glm::rotate(model, -90 * toRadians, glm::vec3(0.0f, 90.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Material_opaco.UseMaterial(uniformSpecularIntensity, uniformShininess);
shrek.RenderModel();

model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-61.2f, 8.75f, 3.8f));
model = glm::scale(model, glm::vec3(12.0f, 12.0f, 12.0f));
model = glm::rotate(model, 180 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::rotate(model, movShrek* toRadians, glm::vec3(1.0f, 0.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Material_opaco.UseMaterial(uniformSpecularIntensity, uniformShininess);
shrek_brazo.RenderModel();

model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-62.8f, 7.65f, 2.8f));
model = glm::scale(model, glm::vec3(12.0f, 12.0f, 12.0f));
model = glm::rotate(model, 180 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::rotate(model, movShrek*2* toRadians, glm::vec3(1.0f, 1.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Material_opaco.UseMaterial(uniformSpecularIntensity, uniformShininess);
shrek_antebrazo.RenderModel();
```

- Textura animada de fuego
  Se multiplica un valor por una variable Delta time para enviar los datos a una variable uniform y así modificar la posición de la textura conforme al tiempo y se obtiene la animación del fuego moviéndose.

```
1
2           //textura con movimiento
3           toffsetu += 0.0 * deltaTime;
4           toffsetv -= 0.01 * deltaTime;
5           //para que no se desborde la variable
6           if (toffsetu > 1.0)
7               toffsetu = 0.0;
8           if (toffsetv > 1.0)
9               toffsetv = 0;
10          toffset = glm::vec2(toffsetu, toffsetv);
11  //fuego
12          color = glm::vec3(1.0f, 1.0f, 1.0f);
13          model = modeltacos;
14          model = glm::translate(model, glm::vec3(-2.0f, 2.0f, 1.7f) * plus);
15          model = glm::scale(model, glm::vec3(0.5f, 1.0f, 0.5f) * plus);
16          glUniform2fv(uniformTextureOffset, 1, glm::value_ptr(toffset));
17          glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
18          glUniform3fv(uniformColor, 1, glm::value_ptr(color));
19          fuego.UseTexture();
20          Material_brillante.UseMaterial(uniformSpecularIntensity, uniformShininess);
21          meshList[3]->RenderMesh();
```



Keyframes

```
//KEYFRAMES DECLARADOS INICIALES

KeyFrame[0].patear = 90.0f;
KeyFrame[0].movAvion_x = 0.0f;
KeyFrame[0].movAvion_y = 0.0f;
KeyFrame[0].giroAvion = 0;
KeyFrame[0].movCoh_x = 0.0f;
KeyFrame[0].movCoh_y = 0.0f;
KeyFrame[0].giroCoh = 0.0f;

KeyFrame[1].movAvion_x = 2.0f;
```

- Animación de patear pelota

Jimmy gira y patea un balón de futbol que hace un recorrido parabólico al presionar la tecla de espacio

```
if (key == GLFW_KEY_X)
{
    theWindow->recorrido = true;
}

if (key == GLFW_KEY_SPACE)
{
    theWindow->patear = true;
```

```
KeyFrame[FrameIndex].patear = patear;
```

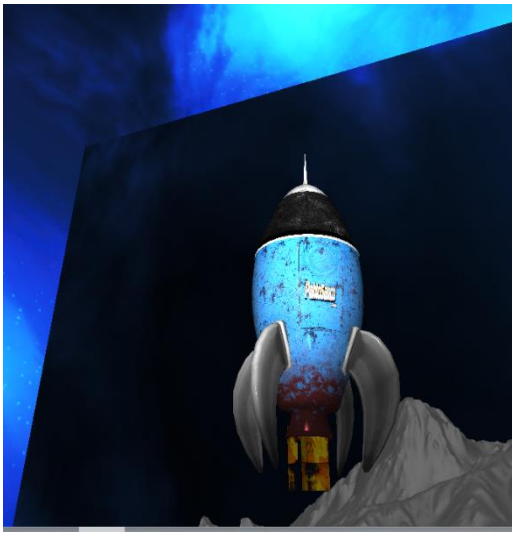
```
//#####################
KeyFrame[playIndex].patearInc = (KeyFrame[playIndex + 1].patear - KeyFrame[playIndex].patear) / i_max_steps;
```

```
model = glm::mat4(1.0);
posblackhawk = glm::vec3(posXavion + movAvion_x, posYavion + movAvion_y, posZavion);
model = glm::translate(model, posblackhawk);
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
model = glm::rotate(model, giroAvion * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::rotate(model, -90 * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::rotate(model, -90 * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Material_brillante.UseMaterial(uniformSpecularIntensity, uniformShininess);
balon.RenderModel();
```

- Cohete espacial

```
glm::mat4 modelcohete(1.0);
model = glm::mat4(1.0);
modelcohete = model;
poscohete = glm::vec3(posXcoh + movCoh_x, posYcoh + movCoh_y, posZcoh);
model = glm::translate(model, poscohete);
model = glm::scale(model, glm::vec3(10.0f, 10.0f, 10.0f));
model = glm::rotate(model, giroCoh * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Material_brillante.UseMaterial(uniformSpecularIntensity, uniformShininess);
cohete.RenderModel();
```

```
//NEW// Keyframes
float posXavion = 4.0, posYavion = -0.2, posZavion = 5.0;
float   movAvion_x = 0.0f, movAvion_y = 0.0f, patear=0.0;

float posXcoh = -80.5, posYcoh = 14.0f, posZcoh = -10.0f;
float   movCoh_x = 0.0f, movCoh_y = 0.0f;
```



- Show de luces con el uso de Spotlights, se activará con la tecla X

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

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

Sonido

Se tienen dos sonidos, uno para el cohete al momento de despegar y otro para el show de luces

cohete

```
bool c = false;
void inputKeyframesAudio(bool* keys){
    if (keys[GLFW_KEY_C] && c == false ){
        SoundEngine->play2D("media/conteo.mp3",c);
        SoundEngine->play2D("media/conteo.mp3", c);
    }

}
```

Luces

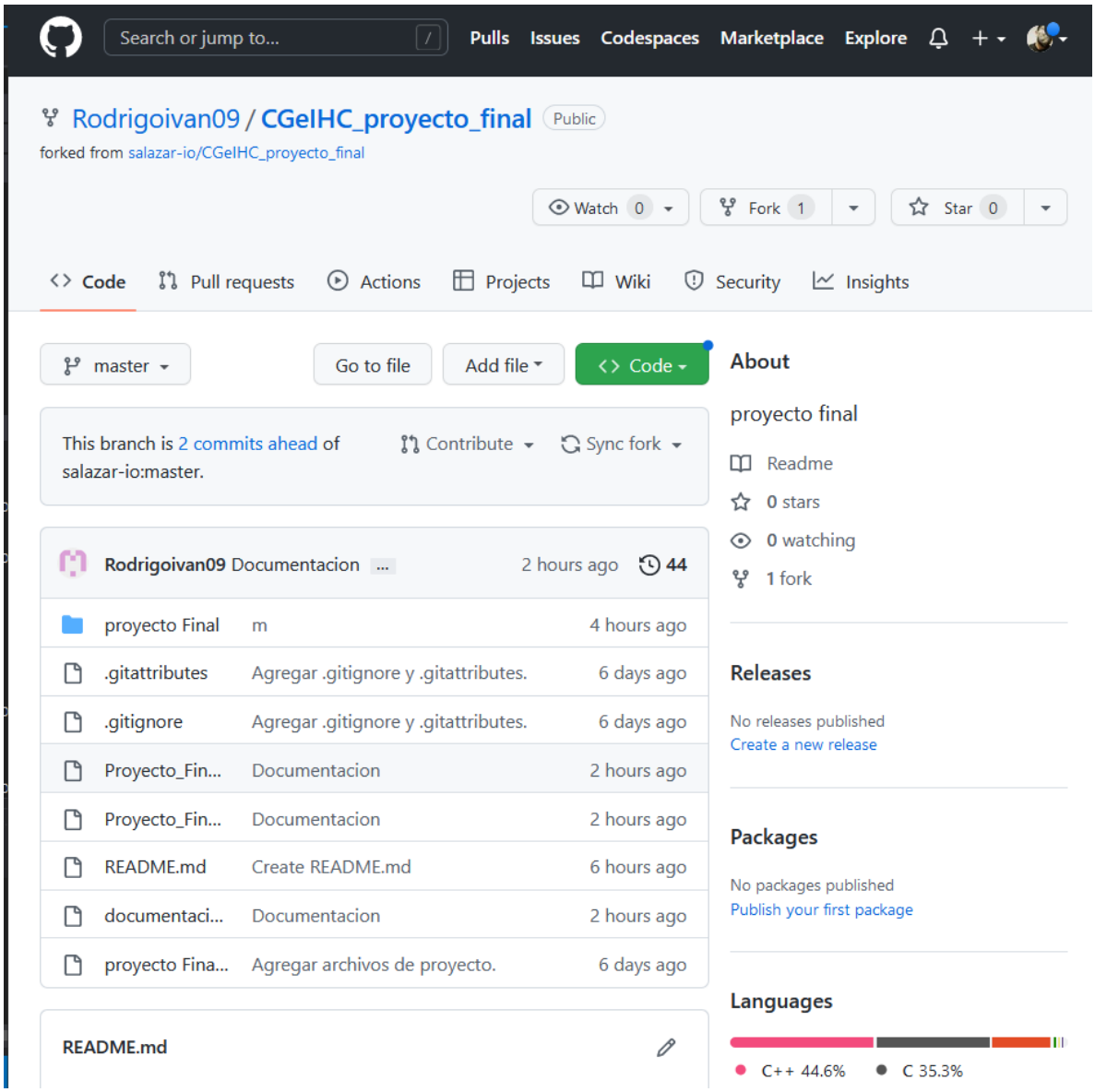
```
    }
    if (posZ < -7.7f)
    {
        rotRodDerS = 0.0f;
        rotRodIzqS = 0.0f;
        rotRodIzq = 0.0f;
        rotRodDer = 0.0f;
        rotBraDerS = 0.0f;
        rotBraIzqS = 0.0f;
        recorrido1 = false;
        SoundEngine->play2D("media/luces.mp3", false);

    }
}
```

5. Control de versiones.

Se utilizó Git y GitHub para trabajar en el proyecto. Con GitHub se creó un repositorio en line que contiene el proyecto y usando Git en el entorno de desarrollo Visual Studio se controlaron los cambios para que tengamos el mismo codigo.

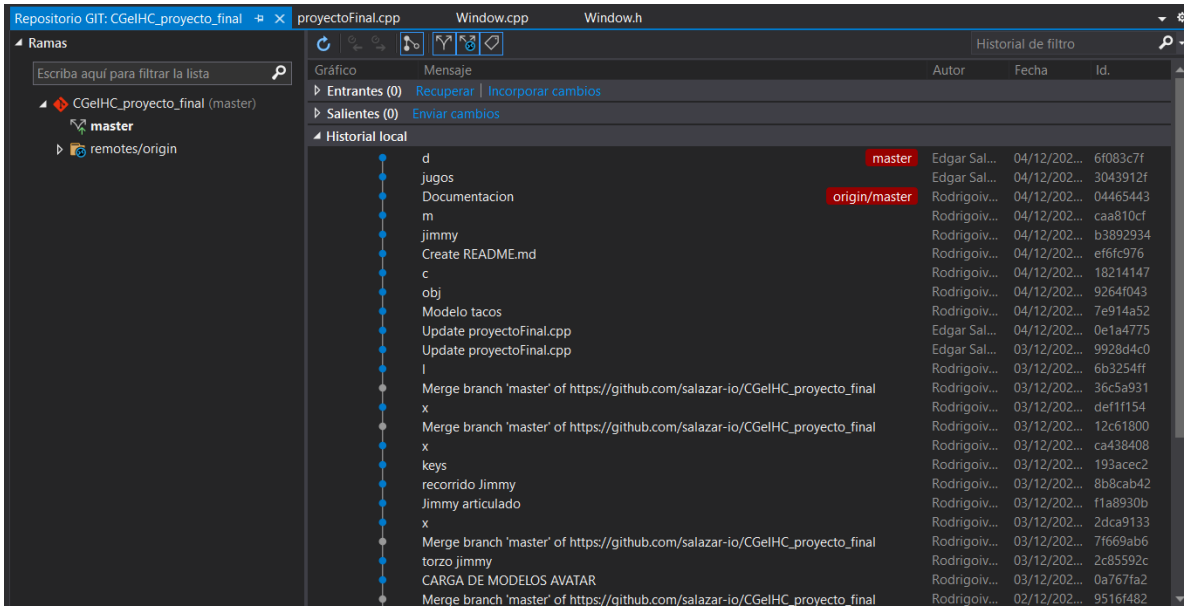
Link del repositorio de este proyecto:

https://github.com/Rodrigoivan09/CGeIHC_proyecto_final



Uso de git en Visual Studio: Desde Visual Studio se clona el repositorio, con lo cual automáticamente se descarga el proyecto en una carpeta local, adicional a esto es importante

hacer la configuración inicial para trabajar con GLFW y Assimp. De esta manera cada colaborador puede actualizar el proyecto y agregar archivos de manera conjunta en un mismo repositorio.



Bibliografía

Algunas imágenes fueron tomadas de los siguientes articulos para ser usadas como texturas

- Times, T. N. Y. (2022, 30 agosto). *El efímero arte del rótulo en Ciudad de México*. The New York Times. Recuperado 5 de octubre de 2022, de https://www.nytimes.com/es/interactive/2022/07/21/espanol/ciudad-de-mexico-rotulos.html
- Mayo, C. R., & Rejón, K. (2022, mayo 26). *La villana de la gráfica popular de CDMX*. Volcánicas; Revista Volcánicas. https://volcanicas.com/la-villana-de-la-grafica-popular-de-cdmx/

El puesto de tortas se obtuvo del siguiente video

- https://www.youtube.com/watch?v=ZGh1RFKt3bk

Modelos 3D descargados

- https://sketchfab.com/features/free-3d-models
- https://free3d.com/