UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE INGENIERÍA ELÉCTRICA

INGENIERÍA EN COMPUTACIÓN

LABORATORIO DE COMPUTACIÓN GRÁFICA e INTERACCIÓN HUMANO COMPUTADORA

EJERCICIOS DE CLASE Nº 09

NOMBRE COMPLETO: Vázquez Reyes Sebastián

Nº de Cuenta: 318150923

GRUPO DE LABORATORIO: 11

GRUPO DE TEORÍA: 6

SEMESTRE 2024-2

FECHA DE ENTREGA LÍMITE: sábado 13 de abril de 2024

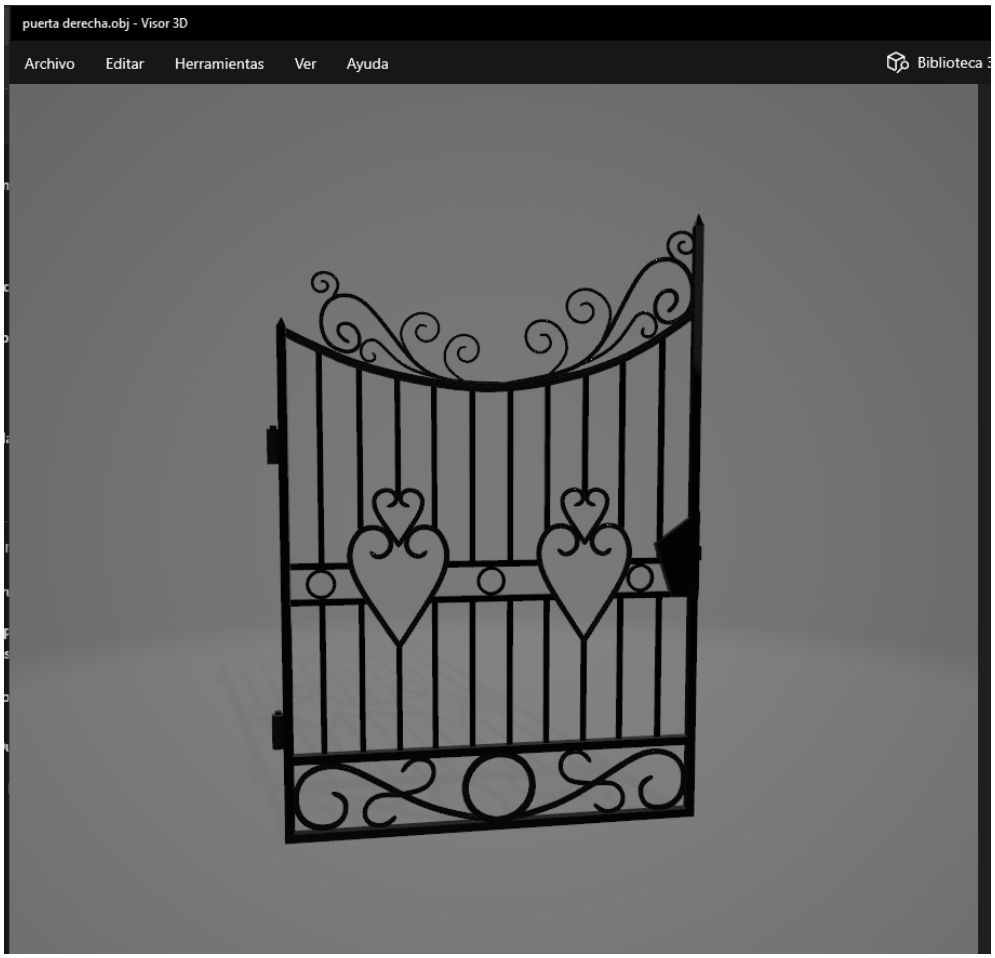
CALIFICACIÓN: _____

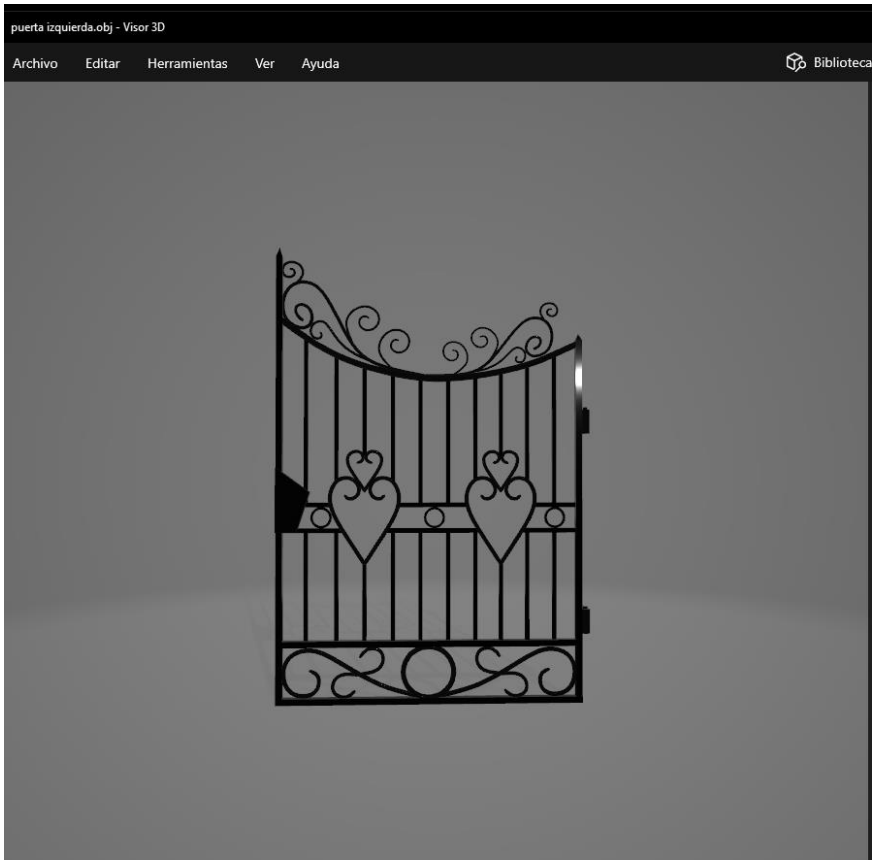
EJERCICIOS DE SESIÓN:

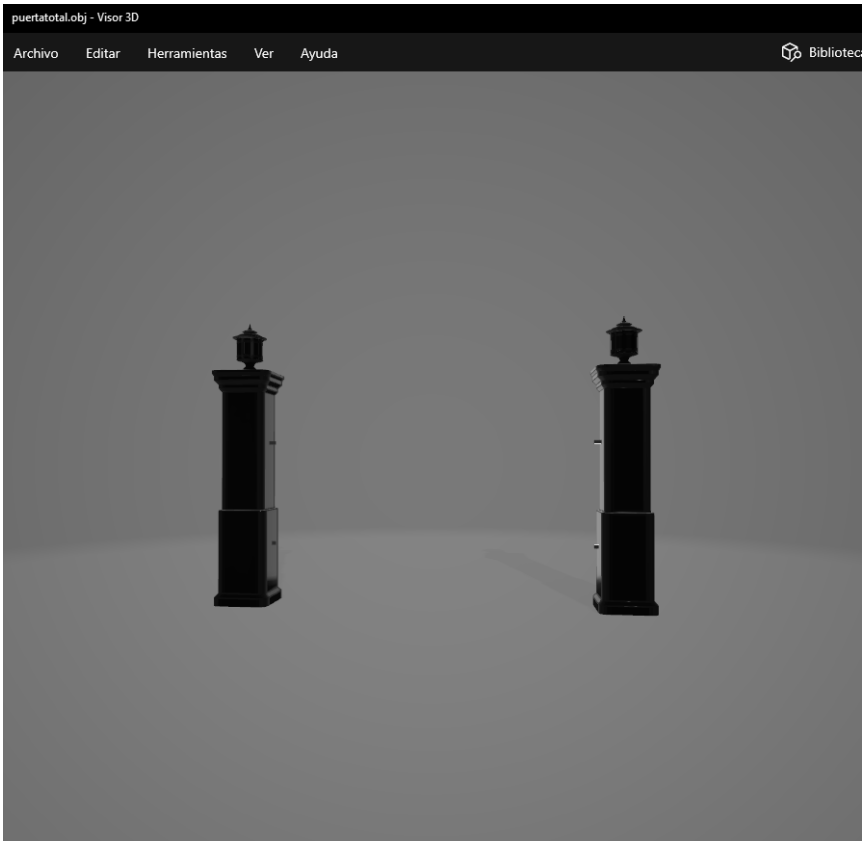
- **Actividades realizadas. Una descripción de los ejercicios y capturas de pantalla de bloques de código generados y de ejecución del programa**

1. **La parte derecha (viendo de frente a la puerta) se abrirá por medio de una rotación hacia afuera y volverá a cerrar, esto de forma cíclica.**

Para realizar las actividades primero dividí la puerta en 3 secciones: la puerta derecha, la izquierda, y el resto del modelo de la puerta (en mi caso, los pilares y lámparas que se colocan en estos pilares). Durante el proceso, me confundí y terminé guardando a las puertas al revés, es decir, el archivo con nombre “puerta izquierda.obj” es en realidad la puerta derecha, y viceversa. No los cambie de nombre porque me di cuenta de esto cuando ya casi tenía terminado mi código y cambiar los nombres me iba a hacer cambiar varias cosas. A continuación se muestran los 3 modelos en los que dividí mi puerta:




puerta izquierda.obj - Visor 3D
Archivo
Editar
Herramientas
Ver
Ayuda
Biblioteca


puertatotal.obj - Visor 3D
Archivo
Editar
Herramientas
Ver
Ayuda
Biblioteca

Y esta es la instancia de mi puerta:

```
//-------------------------DIBUJO DE LA PUERTA-----------------------------

model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-15.0f, 2.0f, -20.0));
model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.3f));
modelaux = model;
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
puertatotal.RenderModel();

//Esta es la puerta izquierda
model = modelaux;
model = glm::translate(model, glm::vec3(-18.0f, 12.75f, +3.5));
model = glm::translate(model, glm::vec3(0.0f, 0.0f, traspt2));
model = glm::translate(model, glm::vec3(movpt2, 0.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
puertaderecha.RenderModel();

//Esta es la puerta derecha
model = modelaux;
model = glm::translate(model, glm::vec3(21.0f, 13.5f, +3.5));
model = glm::rotate(model, rotpt1 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
puertaizquierda.RenderModel();
```

Aunque se renderiza primero el modelo “puertaderecha”, esta es en realidad la puerta izquierda, el comentario intenta avisar de esta pequeña confusión.

La sección del código que se encarga de rotar a la puerta derecha es la siguiente:

```
//condiciones para la animacion de la puerta derecha
if (rota) {
    if (rotpt1 < 90) {
        rotpt1 += rotpt1Offset * deltaTime;
    }
    else {
        rota = false;
    }
} else {
    if (rotpt1 > 0) {
        rotpt1 -= rotpt1Offset * deltaTime;
    }
    else {
        rota = true;
    }
}
```

Es similar a la sección del carro, se encarga de verificar que la variable que guarda los ángulos de rotación sea menor a 90, cuando llega a 90, se cambia el valor de la variable booleana rota, y ahora pasamos a rotar inversamente, disminuyendo el ángulo de rotación hasta cero. En la captura de la instancia de la puerta podemos ver que la puerta derecha ya tiene una rotación asociada a esta variable en el eje de las Y.

2. **La parte izquierda (viendo de frente a la puerta se abrirá deslizándose hacia la izquierda y volverá a cerrar, esto de acuerdo a un contador de tiempo que durará 2 segundos (2 segundos totalmente abierta, 2 segundos totalmente cerrada...)**

Para la puerta izquierda, construí el siguiente código:

```
//condiciones para la animacion de la puerta izquierda
if (mover && tiempo) {
    if (traspt2 < 35) {
        traspt2 += traspt2Offset * deltaTime;
    }
    else if (movpt2 >-15) {
        movpt2 -= traspt2Offset * deltaTime;
    }
    else {
        mover = false;
        tiempo = false;
        abrir = true;
    }
}
else if (abrir && tiempo) {
    if (movpt2 < 0) {
        movpt2 += traspt2Offset * deltaTime;
    }
    else if (traspt2 > 0) {
        traspt2 -= traspt2Offset * deltaTime;
    }
    else {
        mover = true;
        tiempo = false;
        abrir = false;
    }
}
else {
    if (cuenta == 120) {
        tiempo = true;
        cuenta = 0;
    }
    else {
        cuenta += 1;
    }
}
```

Como el programa esta limitado a mostrar 60 frames por segundo (FPS), solo se necesitó crear una variable que contara hasta 120 (2 segundos en frames) dentro del while. Este código primero se encarga de verificar que las variables booleanas mover (que permite el movimiento hacia adelante) y tiempo (que se activa al pasar los 2 segundos) sean true. Cuando lo son, entonces se mueve la puerta hacia adelante, y luego se abre la puerta hacia la izquierda, como la puerta de un centro comercial. Cuando se termina de abrir, la variable tiempo toma el valor de false y debemos esperar otros 2 segundos a que se active en true, para poder realizar el proceso contrario. Al abrirse completamente, se activa la variable abrir y se desactiva la variable mover, cuando abrir y tiempo son true, entonces se cierra la puerta (irónicamente), primero volviendo a su posición original en el eje de las X, y luego regresando a su posición original en el eje de las Z. Luego todo el proceso se repite. En la instancia de la puerta se puede observar que se manejan 2 traslaciones diferentes para la puerta izquierda, una para moverla hacia adelante y otra para abrirla.

- **Problemas Presentados**

  Al inicio, la puerta y el carro no retrocedían en su rotación y traslación, respectivamente, si no que se quedaban quietos al llegar al límite. Resulta que esto se debía a que las variables booleanas envueltas en ambos procesos se declaraban como true dentro del while, provocando que con cada ejecución se quedaran en while y no cambiaran realmente de estado. Sacándolas del while, conseguí que ambos objetos se movieran correctamente.

- **Conclusión:**

  En este ejercicio, aprendí como animar por medio de banderas y condicionales, además, también aprendí a dividir en partes un modelo y texturizarlo por separado utilizando como base el archivo mtl del modelo completo original.