UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE INGENIERÍA ELÉCTRICA

INGENIERÍA EN COMPUTACIÓN

LABORATORIO DE COMPUTACIÓN GRÁFICA e INTERACCIÓN HUMANO COMPUTADORA

REPORTE DE PRÁCTICA Nº 09

NOMBRE COMPLETO: Sanchez Calvillo Saida Mayela

Nº de Cuenta: 318164481

GRUPO DE LABORATORIO: 02

GRUPO DE TEORÍA: 04

SEMESTRE 2025-2

FECHA DE ENTREGA LÍMITE: 26 de abril 2025

CALIFICACIÓN: _____

REPORTE DE PRÁCTICA:

1.- Ejecución de los ejercicios

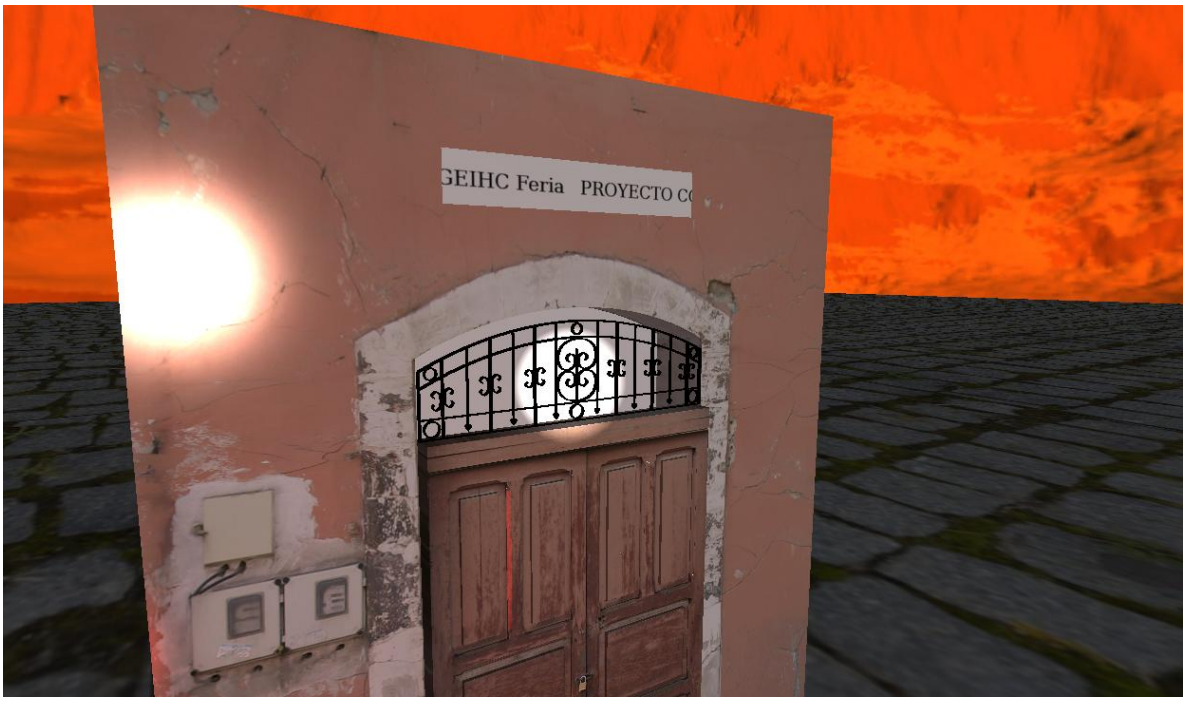
1.-Hacer que en el arco que crearon se muestre la palabra: “PROYECTO CGEIHC Feria” animado desplazándose las letras de izquierda a derecha como si fuera letrero LCD/LED de forma cíclica

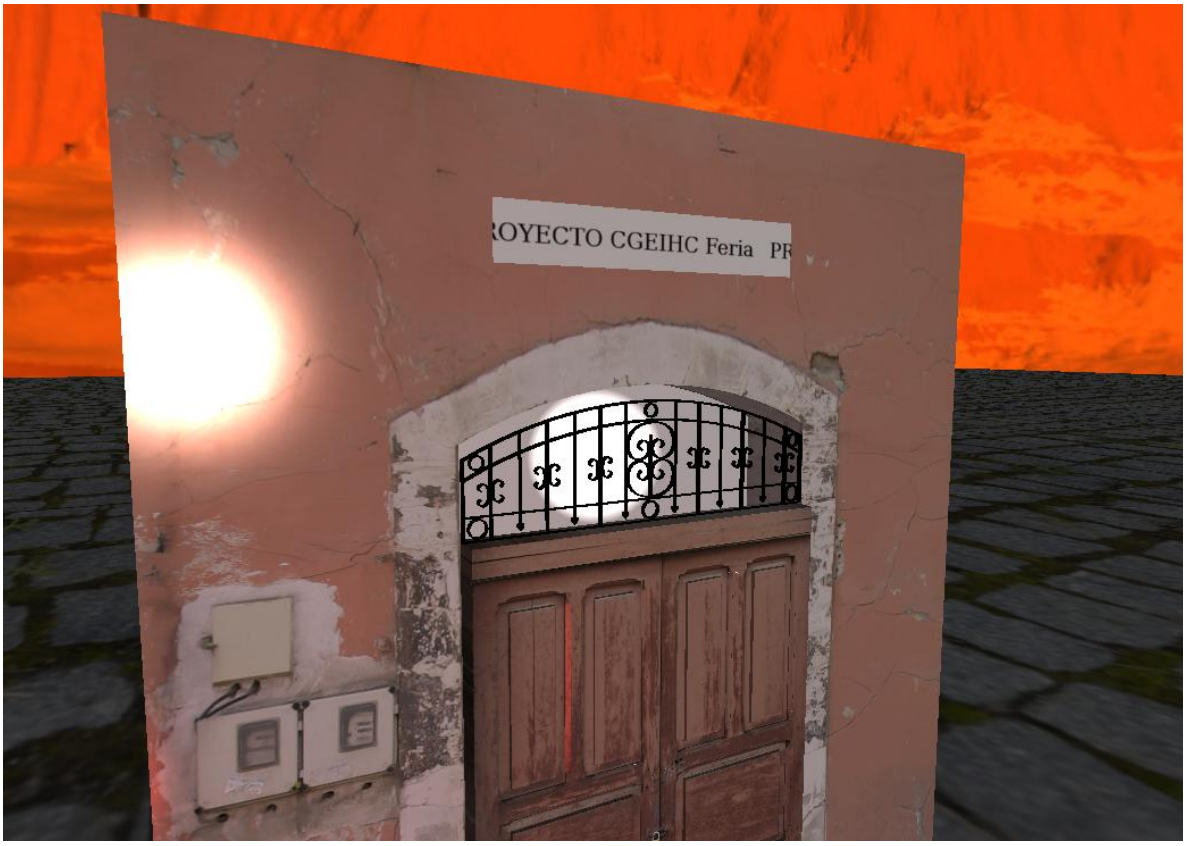
Primero se realizo nuestra textura para poder poner en nuestro letrero.

Renderizamos nuestra puerta y localizamos el lugar de nuestro letrero.

Después agregamos la condición de no desbordamiento de la variable y el movimiento en U para que simula desplazamiento y movimiento cíclico.

```
//Puerta
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(5.0f, 5.0f, -5.0));
//model = glm::rotate(model, -90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
puerta.RenderModel();

toffsetflechau += 0.001;
toffsetflechav = 0.000;
if (toffsetflechau > 1.0)
    toffsetflechau = 0.0;
toffset = glm::vec2(toffsetflechau, toffsetflechav);
//Letrero
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(4.9f, 9.7f, -4.85f));
model = glm::rotate(model, 90 * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(3.0f, 2.0f, 0.5f));
glUniform2fv(uniformTextureOffset, 1, glm::value_ptr(toffset));
color = glm::vec3(1.0f, 1.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Letrero.UseTexture();
Material_brillante.UseMaterial(uniformSpecularIntensity, uniformShininess);
meshList[4]->RenderMesh();
```


GEIHC Feria PROYECTO C


ROYECTO CGEIHC Feria PR

2.- Separar las cabezas (con todo y cuello) del Dragón y agregar las siguientes animaciones:

- **Movimiento del cuerpo ida y vuelta.**
- **Aleteo**
- **Cada cabeza se mueve de forma diferente de acuerdo a una función/algoritmo diferente (ejemplos: espiral de Arquímedes, movimiento senoidal, lemniscata, etc..)**
- **Cada cabeza debe de verse de un color diferente: roja, azul, verde, blanco, café.**

Primero agarramos nuestro proyecto del dragón y separamos las cabezas, después las renderizamos y las acomodamos en donde van y así formar a nuestro dragón.

Para **el movimiento de ida y vuelta**, primero agregamos una variable independiente para la posición del dragón: *movDragon*.

```
bool avanza;
float movDragon = 0.0f;
```

Para limitar el área se utilizó una condición *if-else* basada en una bandera booleana llamada *avanza*, tal cual lo hicimos con el coche:

La lógica es:

- Cuando avanza es verdadero, tanto el coche como el dragón se mueven hacia la izquierda (restando movOffset * deltaTime).
- Cuando alcanzan el límite de **-**250, *avanza* cambia a falso, y ambos comienzan a moverse hacia la derecha o “de regreso” (sumando movOffset * deltaTime).
- Al llegar a 350, avanza se reinicia a verdadero, cerrando el ciclo de ida y vuelta.

```
if (avanza) {
    if (movDragon > -250.0f)
    {
        movDragon -= movOffset * deltaTime;
    }
    else
    {
        avanza = false;
    }
}
else {
    if (movDragon < 350.0f)
    {
        movDragon += movOffset * deltaTime;
    }
    else
    {
        avanza = true;
    }
}
```

```
model = glm::mat4(1.0);
//model = glm::translate(model, glm::vec3(0.0f, 5.0f+sin(glm::radians(angulovaria)), 6.0));
//model = glm::translate(model, glm::vec3(0.0f - angulovaria * 0.1, 5.0f + 7.0 * sin(glm::radians(angulovaria * 2.0)), 6.0));
model = glm::translate(model, glm::vec3(movDragon - 50.0f, 5.0f + 7.0 * sin(glm::radians(angulovaria * 2.0)), 6.0));
model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.3f));
```

Se quito *angulovaria* ya que este controlaba el movimiento en x e indicaba que solo se movía de acuerdo con el reloj interno y se iba a mover muy lejos sin limite

El **aleteo** ya se implementó en el ejercicio.

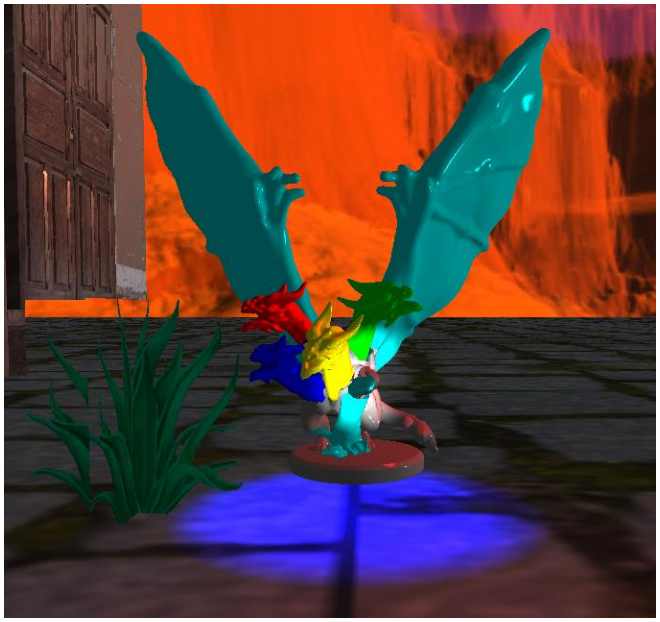
Para la **animación de las cabezas**, cada cabeza tiene una función matemática diferente que controla su rotación para simular un comportamiento distinto y realista:

Cabeza	***Movimiento***	***Función usada***	***Descripción Visual***
1	Movimiento senoidal	sin(angulovaria * 0.10f)	Se mueve arriba y abajo muy lentamente. Oscilación tipo "asentir".
2	Espiral de Arquímedes	angulovaria * 20.0f	Gira constantemente en su base sobre el eje Y. Rotación continua como radar.
3	Lemniscata lenta (∞)	sin(2.0f * angulovaria * 0.5f)	Oscila pero inclinada, con un patrón más rápido. Como un balanceo medio extraño.
4	Movimiento circular	cos(angulovaria * 0.5f)	Se mueve en un pequeño giro lateral (muy discreto) a ritmo lento.
5	Movimiento parabólico	0.05f * angulovaria²	Gira cada vez más rápido (porque es parabólico), como si se estuviera "torciendo" de más.

Para que cada cabeza se viera de color distinto, solo se uso un modelo de color para poder cambiar de color nuestras cabezas por individual.

```
//EL angulo varia afecta el angulo sinuidal
model = glm::mat4(1.0);
//model = glm::translate(model, glm::vec3(0.0f, 5.0f+sin(glm::radians(angulovaria)), 6.0));
//model = glm::translate(model, glm::vec3(0.0f - angulovaria * 0.1, 5.0f + 7.0 * sin(glm::radians(angulovaria * 2.0)), 6.0));
model = glm::translate(model, glm::vec3(movDragon - 50.0f, 5.0f + 7.0 * sin(glm::radians(angulovaria * 2.0)), 6.0));
model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.3f));
model = glm::rotate(model, -90 * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
modelaux = model;
modelaux2 = model;
Material_brillante.UseMaterial(uniformSpecularIntensity, uniformShininess);
/*color = glm::vec3(0.0f, 1.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));*/
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
dragonCuerpo.RenderModel();

//Cabeza 1 del dragon  -  Movimiento circular
model = modelaux;
model = glm::translate(model, glm::vec3(-2.1f, 0.7f, 3.4f));   //+x, -z, +y
float radio = 0.5f;
model = glm::rotate(model, glm::radians(10.0f * sin(angulovaria * 0.10f)), glm::vec3(1.0f, 0.0f, 0.0f)); // Oscila en X
color = glm::vec3(1.0f, 0.0f, 0.0f); // Rojo
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Cabeza1.RenderModel();

//Cabeza 2 del dragon -  Movimiento espiral de Arquímedes
model = modelaux;
model = glm::translate(model, glm::vec3(-1.8f, -0.3f, 3.5f));
model = glm::rotate(model, glm::radians(angulovaria * 20.0f), glm::vec3(0.0f, 1.0f, 0.0f));
color = glm::vec3(0.0f, 1.0f, 0.0f); // verde
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Cabeza2.RenderModel();
```

```
//Cabeza 3 del dragon  -  Movmiento tipo lemniscata
model = modelaux;
model = glm::translate(model, glm::vec3(-2.1f, 0.7f, 2.6f));
model = glm::rotate(model, glm::radians(20.0f * sin(2.0f * angulovaria * 0.5f)), glm::vec3(1.0f, 1.0f, 0.0f));
color = glm::vec3(0.0f, 0.0f, 1.0f); // azul
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Cabeza3.RenderModel();

//Cabeza 4 del dragon - Movimiento circular
model = modelaux;
model = glm::translate(model, glm::vec3(-2.4f, 0.3f, 2.8f));
model = glm::rotate(model, glm::radians(15.0f * cos(angulovaria * 0.5f)), glm::vec3(1.0f, 0.0f, 1.0f));
color = glm::vec3(1.0f, 1.0f, 0.0f); // ?
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Cabeza4.RenderModel();

//Cabeza 5 del dragon  -  Movmiento parabolico
model = modelaux;
model = glm::translate(model, glm::vec3(-2.2f, -0.4f, 2.7f));
model = glm::rotate(model, glm::radians(0.05f * angulovaria * angulovaria), glm::vec3(0.0f, 1.0f, 1.0f));
color = glm::vec3(0.0f, 1.0f, 1.0f); // ?
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Cabeza5.RenderModel();
```



2.- Liste los problemas que tuvo a la hora de hacer estos ejercicios y si los resolvió explicar cómo fue, en caso de error adjuntar captura de pantalla

El único problema que hubo fue decidir si las cabezas del dragón hacían movimientos rotacionales o traslacionales. Se llego a la conclusión de que eran rotacionales ya que se mueven en su eje, pero no cambian de lugar mientras lo hace, significa que tiene una base estable y de ahí hacia movimientos. Esto hizo que nuestras líneas de código fueran más complicadas porque no se pudo agregar mucho código y solo se usaron funciones ya conocidas para simular el movimiento.

3.- Conclusión:

a. Los ejercicios del reporte: Complejidad, Explicación.

Los ejercicios de esta práctica presentaron un nivel de complejidad intermedio, ya que implicaron no solo la transformación geométrica básica de objetos (translate, rotate, scale), sino también la animación dependiente del tiempo utilizando deltaTime y variables auxiliares como angulovaria.

Se requería comprender cómo separar movimientos independientes para diferentes partes de un mismo modelo (en este caso, las cabezas y alas del dragón) y cómo implementar movimientos personalizados.

b. Comentarios generales: Faltó explicar a detalle, ir más lento en alguna explicación, otros comentarios y sugerencias para mejorar desarrollo de la práctica

Durante el desarrollo de la práctica, algunos conceptos como la diferencia entre usar angulovaria directamente o controlar movimientos con variables auxiliares (movCoche) pudieron explicarse con mayor detalle, para entender mejor cuándo es necesario un control manual y cuándo el tiempo puede gobernar un movimiento.

Asimismo, se podría mejorar el desarrollo explicando más despacio el orden correcto de transformaciones (translate antes que rotate, por ejemplo) y su impacto visual en los objetos.

c. Conclusión

La práctica permitió implementar exitosamente animaciones dinámicas y movimientos independientes en modelos 3D complejos.

El trabajo desarrollado demuestra la importancia de estructurar correctamente tanto la jerarquía como las transformaciones geométricas y de tener control detallado sobre la posición, rotación y animación de cada componente en una escena tridimensional.

4.- Bibliografía en formato APA

- OpenAI. (2025). ChatGPT [Large language model]. https://chatgpt.com