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: Roberto Aburto López

Nº de Cuenta: 319131996

GRUPO DE LABORATORIO: 03

GRUPO DE TEORÍA: 06

SEMESTRE 2026-1

FECHA DE ENTREGA LÍMITE: 28 de octubre de 2025

CALIFICACIÓN: _____

EJERCICIOS DE SESIÓN:

1. Agregar que el número cambiante sea a una velocidad visible.

Para este ejercicio tuve que crear unas variables que me permitieran incrementar poco a poco un valor flotante hasta que se cumpliera la condición del if. De esta manera, cada segundo (tiempoNumero >= 1.0f), el valor de toffsetnumerocambiau aumenta ligeramente, generando un desplazamiento en la textura. Cuando este valor supera 1.0, se reinicia a 0.0 para que el ciclo se repita de forma continua.

```
//------------ número cambiante con velocidad visible -----------------
tiempoNumero += deltaTime;
if (tiempoNumero >= 1.0f) {
    toffsetnumerocambiau += 0.005;
    if (toffsetnumerocambiau > 1.0)
        toffsetnumerocambiau = 0.0;
    tiempoNumero = 0.0f;
}
toffsetnumerov = 0.0;
toffset = glm::vec2(toffsetnumerocambiau, toffsetnumerov);
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-10.0f, 10.0f, -6.0f));
model = glm::rotate(model, 90 * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(3.0f, 3.0f, 3.0f));
glUniform2fv(uniformTextureOffset, 1, glm::value_ptr(toffset));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(1.0f, 1.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
NumerosTexture.UseTexture();
Material_brillante.UseMaterial(uniformSpecularIntensity, uniformShininess);
meshList[6]->RenderMesh();
```

2. Alternar entre la textura del número 1 y 2 de forma automática.

Implementé un sistema para cambiar automáticamente entre dos texturas (número 1 y número 2) después de cierto tiempo.

Primero, se usa la variable tiempoTextura, que se incrementa constantemente con deltaTime. Cuando su valor supera 1.0 segundo, se aumenta ligeramente la variable toffsetnumerocambiau2, que sirve como un contador auxiliar para controlar el cambio. Una vez que este contador supera el valor de 1.0, se reinicia a 0.0 y se invierte el valor de la variable booleana usarNumero1 con usarNumero1 = !usarNumero1;. Esto provoca que cada cierto intervalo el programa alterne entre las dos texturas de forma automática.

```
// ------- cambiar automáticamente entre textura número 1 y número 2 ------
tiempoTextura += deltaTime;
if (tiempoTextura >= 1.0f) {
    toffsetnumerocambiau2 += 0.02;
    if (toffsetnumerocambiau2 > 1.0)
    {
        toffsetnumerocambiau2 = 0.0f;
        usarNumero1 = !usarNumero1;
    }
    tiempoTextura = 0.0f;
}


toffsetnumerou = 0.0;
toffsetnumerov = 0.0;
toffset = glm::vec2(toffsetnumerou, toffsetnumerov);
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-13.0f, 10.0f, -6.0f));
model = glm::rotate(model, 90 * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(3.0f, 3.0f, 3.0f));
glUniform2fv(uniformTextureOffset, 1, glm::value_ptr(toffset));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(1.0f, 1.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));

if (usarNumero1) {
    Numero1Texture.UseTexture();
}
else {
    Numero2Texture.UseTexture();
}

Material_brillante.UseMaterial(uniformSpecularIntensity, uniformShininess);
meshList[5]->RenderMesh();
```

3. Separarle las alas al dragón, hacer que el dragón avance 20 unidades de forma senoidal aleteando y regrese (loop)

Para este ejercicio se programó la animación del dragón para que se desplazara de un lado a otro y mantuviera un movimiento constante de aleteo.

Se utilizó la variable avanzaDragon para controlar la dirección de su avance, mientras es verdadera, el dragón se mueve hacia adelante incrementando tanto su posición como su ángulo para simular el vuelo del dragón.

Cuando alcanza un límite de 10.0f, cambia de dirección girando 180 grados y comienza a desplazarse en sentido contrario. Al llegar al otro extremo, con un valor de -10.0f, vuelve a invertir su dirección repitiendo el ciclo.

```
//Dragon

// ---------------------- ANIMACION DEL DRAGON -------------------------
if (avanzaDragon) {
    if (dragonavance < 10.0f)
    {
        angulovaria += 7.0f * deltaTime;
        dragonavance += 0.1f * deltaTime;
    }
    else
    {
        avanzaDragon = false;
        giroDragon += 180;
    }
}
else
{
    if (dragonavance > -10.0f)
    {
        angulovaria += 7.0f * deltaTime;
        dragonavance -= 0.1f * deltaTime;
    }
    else
    {
        avanzaDragon = true;
        giroDragon += 180;
    }
}
```

Por otro lado, la animación de las alas se controló mediante la variable aleteo, que aumenta o disminuye su valor para crear un movimiento de subida y bajada. Cuando las alas alcanzan un ángulo máximo de 40.0f, el movimiento se invierte hasta llegar a -40.0f, logrando así un aleteo continuo y natural. En conjunto, ambas animaciones hacen que el dragón parezca volar de forma realista, avanzando y batiendo sus alas de manera sincronizada.

```
// ------------- ANIMACION DE ALAS (ALETEO) ---------------------------
if (subiendo) {
    if (aleteo < 40.0f)
    {
        aleteo += 3.0f * deltaTime; // Velocidad de aleteo (ajusta según
prefieras)
    }
    else
    {
        subiendo = false;
    }
}
else
{
    if (aleteo > -40.0f)
    {
        aleteo -= 3.0f * deltaTime;
    }
    else
    {
        subiendo = true;
    }
}
```

Modelo jerárquico

```
    // ---------------- DRAGON ----------------------
//dragonavance para moverlo hacia adelante y atrás en X
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(dragonavance -5.0f, 5.0f + sin(glm::radians(angulovaria)), 6.0));
modelaux2 = model;
model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.3f));
model = glm::rotate(model, 180 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::rotate(model, giroDragon * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
Material_brillante.UseMaterial(uniformSpecularIntensity, uniformShininess);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Dragon_M.RenderModel();

//Ala derecha
model = modelaux2;
model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
model = glm::rotate(model, 180 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::rotate(model, giroDragon * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::rotate(model, -aleteo * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.3f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
AlaDerecha_M.RenderModel();

//Ala Izquierda
model = modelaux2;
model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
model = glm::rotate(model, 180 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::rotate(model, giroDragon * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::rotate(model, aleteo * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.3f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
AlaIzquierda_M.RenderModel();
```

Resultados

Video:

https://drive.google.com/file/d/1gw9GgrLX4DYzVHUZNmTVcnjQraSdSJ6h/view?usp=sharing

Conclusión:

No se presentaron problemas durante la práctica. Me gustó la explicación y también que se realizaran ejercicios como el del auto en clase, ya que ayudaron a resolver dudas y entender mejor la lógica de la animación. No se me complicó porque, sin querer, ya había hecho animaciones en prácticas anteriores, como la del helicóptero, lo que me facilitó comprender el funcionamiento.