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 09-2

NOMBRE COMPLETO: Barragán Pilar Diana

Nº de Cuenta: 318147981

GRUPO DE LABORATORIO: 03

GRUPO DE TEORÍA: 04

SEMESTRE 2025-1

FECHA DE ENTREGA LÍMITE: 28 de octubre de 2024

CALIFICACIÓN: _____

Práctica 9-2: Animación Avanzada

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

Para lograr lo que se me pedía en este primer inciso solamente utilice lo aprendido en clase respecto a la animación de texturas, puesto que a partir de la textura con la línea escrita que se nos pidió simplemente en el código la importe y con ayuda de los parámetros `toffsetletrasu` y `toffsetletrasv` ya que no se puede afectar directamente a la variable uniform le di el movimiento como de un letrero led de forma cíclica, para finalmente solo acomodarlas en el modelo de arco y letrero que se tenía con anterioridad.

```
///Letras
toffsetletrasu += 0.001;
toffsetletrasv = 0.000;
//para que no se desborde la variable
if (toffsetletrasu > 1.0)
    toffsetletrasu = 0.0;
//if (toffsetv > 1.0)
//  toffsetv = 0;
//printf("\ntfosset %f \n", toffsetu);
//pasar a la variable uniform el valor actualizado
toffset = glm::vec2(toffsetletrasu, toffsetletrasv);

model = glm::mat4(1.0);
//model = glm::translate(model, glm::vec3(-2.0f, 3.0f, -6.0f));
model = glm::translate(model, glm::vec3(-0.1f, 2.5f, 0.5f));
model = glm::rotate(model, 90 * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(2.0f, 3.0f, 3.0f));
glUniform2fv(uniformTextureOffset, 1, glm::value_ptr(toffset));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Letras.UseTexture();
Material_brillante.UseMaterial(uniformSpecularIntensity, uniformShininess);
meshList[7]->RenderMesh();
```

Figura 1. Código para que se muestre frase como letrero LCD/LED.

2. Separar las cabezas 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.

Para mover el cuerpo del dragón realice lo mismo que en el ejercicio sobre el recorrido de practicas anteriores de un vehículo, utilice la una variable de

movimiento que en el primer if delimitaba hasta donde terminaba el recorrido y se realizaba un giro de 180 grados hacía el sentido contrario para regresar y así sucesivamente de forma cíclica.

```
//ida y vuelta
if (adelante) {
    if (movHidra > -38.0f) {
        movHidra -= movOffsetH * deltaTime;
    }
    else {
        printf("rotHidra %f \n ", rotHidra);
        if (rotHidra < 180.0f)
        {
            rotHidra += rotHidraOffset * deltaTime;
        }
        else {
            adelante = false;
            //rotHidra = 0.0f;
        }

    }
}
else {
    if (movHidra < 0.0f) {
        movHidra += movOffsetH * deltaTime;
    }
    else {
        //printf("rotHidra %f \n ", rotHidra);
        if (rotHidra < 180.0f)
        {
            rotHidra += rotHidraOffset * deltaTime;
        }
        else {
            adelante = true;
            rotHidra = 0.0f;
        }
    }
}
```

Figura 2. Código para el movimiento de ida y vuelta

Para el aleteo primero separe las alas en 3ds Max para posteriormente importar el modelo y posicionar todo de forma correcta junto con las alas, después con una variable de rotación y con ayuda de las sentencias if, delimite la rotación hacia arriba y debajo del ala derecha para después hacer lo mismo con el ala izquierda.

```
//Movimiento para el ala derecha
if (avanza) {
    //printf("rotAla %f \n ", rotAla);
    if (rotAla1 < 30.0f)
    {
        rotAla1 += rotalaOffset1 * deltaTime;
    }
    else {
        //rotAla = -20.0f;
        avanza = !avanza;
    }
}
else {

    if (rotAla1 > -20.0f)
    {
        rotAla1 -= rotalaOffset1 * deltaTime;
    }
    else {
        rotAla1 = -20.0f;
        avanza = true;
    }

}
```

```
//Movimiento para el ala izquierda
if (avanza2) {
    //printf("rotAla %f \n ", rotAla);
    if (rotAla2 > -30.0f)
    {
        rotAla2 -= rotalaOffset2 * deltaTime;
    }
    else {
        //rotAla2 = 20.0f;
        avanza2 = !avanza2;
    }
}
else {

    if (rotAla2 < 20.0f)
    {
        rotAla2 += rotalaOffset2 * deltaTime;
    }
    else {
        rotAla2 = 20.0f;
        avanza2 = true;
    }

}
```

Figura 3. Código del movimiento para el ala derecha e izquierda.

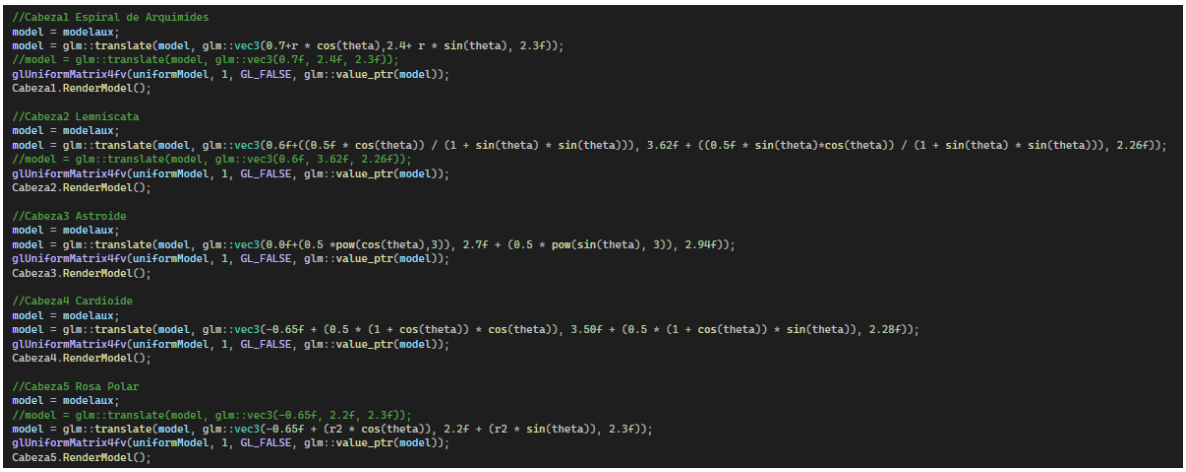
Finalmente para hacer que cada cabeza trazará un movimiento distinto dependiendo de una función matemática lo primero que realice fue separa cada una de las cabezas del modelo en 3ds Max y para que cada una tuviera un color distinto las texturice de esta forma, luego simplemente importe los modelos y los acomode con ayuda del translate. Para las funciones primero busque cuales podría realizar, sus funciones que estaban en coordenadas polares y sus transformaciones a cartesianas para asignarle en x o en y los valores de dichas ecuaciones, las que elegí fueron; espiral de Arquímedes, Lemniscata, Astroide, Cardioide y la Rosa Polar, declare sus respectivas variables necesarias para el calculo de x y de y, como el angulo theta, a y b, k que es el número de pétalos de la rosa y los cálculos de los radios, por ultimo solo coloque las ecuaciones en el translate de x y en y para cada una de las cabezas.

```
//HIDRA
model = glm::mat4(1.0);
//printf("\nmovHidra %f \n", movHidra);
model = glm::translate(model, glm::vec3(movHidra +0.0f, 5.0f+sin(glm::radians(angulovaria*2)), 6.0));
//model = glm::translate(model, glm::vec3(0.0f, 5.0f, 6.0));
model = glm::scale(model, glm::vec3(0.6f, 0.6f, 0.6f));
model = glm::rotate(model, -90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::rotate(model, rotHidra * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
modelaux = 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));
Cuerpo.RenderModel();

//Ala Derecha
model = modelaux;
model = glm::translate(model, glm::vec3(0.3f, 2.7f, 0.48f));
//model = glm::rotate(model, -90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::rotate(model, rotAla1 * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
//model = glm::rotate(model, 30.0f * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
AlaD.RenderModel();

//Ala Izquierda
model = modelaux;
model = glm::translate(model, glm::vec3(-0.6f, 2.9f, 0.6f));
//model = glm::rotate(model, -90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::rotate(model, rotAla2 * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
//model = glm::rotate(model, -30.0f * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
AlaI.RenderModel();
```

Figura 4. Código del dragón y sus respectivas transformaciones.



```
//Cabeza1 Espiral de Arquimides
model = modelaux;
model = glm::translate(model, glm::vec3(0.7+r * cos(theta),2.4+ r * sin(theta), 2.3f));
//model = glm::translate(model, glm::vec3(0.7f, 2.4f, 2.3f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Cabeza1.RenderModel();

//Cabeza2 Lemniscata
model = modelaux;
model = glm::translate(model, glm::vec3(0.6f+((0.5f * cos(theta)) / (1 + sin(theta) * sin(theta))), 3.62f + ((0.5f * sin(theta)*cos(theta)) / (1 + sin(theta) * sin(theta))), 2.26f));
//model = glm::translate(model, glm::vec3(0.6f, 3.62f, 2.26f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Cabeza2.RenderModel();

//Cabeza3 Astroide
model = modelaux;
model = glm::translate(model, glm::vec3(0.0f+(0.5 *pow(cos(theta),3)), 2.7f + (0.5 * pow(sin(theta), 3)), 2.94f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Cabeza3.RenderModel();

//Cabeza4 Cardioide
model = modelaux;
model = glm::translate(model, glm::vec3(-0.65f + (0.5 * (1 + cos(theta)) * cos(theta)), 3.50f + (0.5 * (1 + cos(theta)) * sin(theta)), 2.28f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Cabeza4.RenderModel();

//Cabeza5 Rosa Polar
model = modelaux;
//model = glm::translate(model, glm::vec3(-0.65f, 2.2f, 2.3f));
model = glm::translate(model, glm::vec3(-0.65f + (r2 * cos(theta)), 2.2f + (r2 * sin(theta)), 2.3f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Cabeza5.RenderModel();
```

Figura 5. Cabezas del dragón son sus respectivas ecuaciones.

Al compilar el código de esta práctica no se me presento ningún inconveniente respecto a la programación de este, puesto que todo funciono con normalidad.

CONCLUSIÓN:

El realizar esta práctica me permitió conocer más a fondo el tema de la animación de cada vez más elementos como lo son en este caso las texturas igual que todos los tipos de movimiento que se pueden generar modificando el código para conocerlo de mejor manera y poder relacionarnos mejor con distintas implementaciones de estos ejercicios como el que un modelo se mueva o una textura puesto que existen infinitas manera de realizar movimiento.