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: Medrano Miranda Daniel Ulises

Nº de Cuenta: 318045351

GRUPO DE LABORATORIO: 03

GRUPO DE TEORÍA: 04

SEMESTRE 2025-1

FECHA DE ENTREGA LÍMITE: 26/Octubre/2024

CALIFICACIÓN: _____

REPORTE DE PRÁCTICA:

1.- Ejecución de los ejercicios que se dejaron, comentar cada uno y capturas de pantalla de bloques de código generados y de ejecución del programa.

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

```
61    float toffsetanunciou = 0.0f;
62    float toffsetanunciov = 0.0f;
63    float toffsetanunciocambiav = 0.0f;


93    Texture AnuncioTexture;


390       AnuncioTexture = Texture("Textures/LetreroMonopoly2.tga");
391       AnuncioTexture.LoadTextureA();


974           toffsetanunciou += 0.0005;
975           toffsetanunciov = 0.0;
976
977           //para que no se desborde la variable
978           if (toffsetanunciou > 0.9) {
979               toffsetanunciou = 0.0;
980               toffsetanunciocambiav -= 0.33333;
981           }
982           /*if (toffsetanunciov > - 0.98)
983               toffsetanunciov = 0;*/
984           toffset = glm::vec2(toffsetanunciou, toffsetanunciocambiav);
985           model = modelauxLetrero;
986           model = glm::rotate(model, 90 * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
987           model = glm::translate(model, glm::vec3(0.0f, 0.05f, 0.0));
988           model = glm::scale(model, glm::vec3(5.8f, 1.4f, 1.4f));
989           glUniform2fv(uniformTextureOffset, 1, glm::value_ptr(toffset));
990           glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
991           color = glm::vec3(1.0f, 1.0f, 1.0f);
992           glUniform3fv(uniformColor, 1, glm::value_ptr(color));
993           AnuncioTexture.UseTexture();
994           Material_brillante.UseMaterial(uniformSpecularIntensity, uniformShininess);
995           meshList[6]->RenderMesh();
```

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.

```
65    float giraAlas = 0.0f;
66    float giraAlasOffset = 0.8f;
67    bool gira = true;
68    float mueveDragon = 0.0f;
69    float mueveDragonOffset = 0.15f;
70    bool vuela = true;
71    float giroDragon = 0.0f;

104   Model CuerpoDragon_M;
105   Model AlaDerecha_M;
106   Model AlaIzquierda_M;
107   Model CabezaEnMedio_M;
108   Model CabezaIzquierdaArriba_M;
109   Model CabezaIzquierdaAbajo_M;
110   Model CabezaDerechaArriba_M;
111   Model CabezaDerechaAbajo_M;
```

```
CuerpoDragon_M = Model();
CuerpoDragon_M.LoadModel("Models/CuerpoDragon.obj");
AlaDerecha_M = Model();
AlaDerecha_M.LoadModel("Models/AlaDerecha.obj");
AlaIzquierda_M = Model();
AlaIzquierda_M.LoadModel("Models/Alaizquierda.obj");
CabezaEnMedio_M = Model();
CabezaEnMedio_M.LoadModel("Models/CabezaEnMedio.obj");
CabezaIzquierdaArriba_M = Model();
CabezaIzquierdaArriba_M.LoadModel("Models/CabezaArribaIzquierda.obj");
CabezaDerechaArriba_M = Model();
CabezaDerechaArriba_M.LoadModel("Models/CabezaArribaDerecha.obj");
CabezaIzquierdaAbajo_M = Model();
CabezaIzquierdaAbajo_M.LoadModel("Models/CabezaAbajoIzquierda.obj");
CabezaDerechaAbajo_M = Model();
CabezaDerechaAbajo_M.LoadModel("Models/CabezaAbajoDerecha.obj");
```

```
        // - - - - - - DRAGÓN - - - - - -
        // Cuerpo Dragon
        printf("%f\n", mueveDragon);
        if (vuela) {
            if (mueveDragon < 50.0f) {
                mueveDragon += mueveDragonOffset * deltaTime;
            }
            else {
                vuela = !vuela;
                giroDragon = 180.0f;
            }
        }
        else {
            if (mueveDragon > -50.0f) {
                mueveDragon -= mueveDragonOffset * deltaTime;
            }
            else {
                vuela = !vuela;
                giroDragon = 0.0f;
            }
        }
        model = glm::mat4(1.0);
        model = glm::translate(model, glm::vec3(0.0f - mueveDragon/2, 5.0f - (3*sin(glm::radians(angulovaria*5))), 6.0f));
        model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.3f));
        model = glm::rotate(model, giroDragon * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
        modelauxDragon = 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));
        CuerpoDragon_M.RenderModel();

        if (gira) {
            if (giraAlas < 70.0f){
                giraAlas += giraAlasOffset * deltaTime;
            }
            else {
                gira = !gira;
            }
        }
        else {
            if (giraAlas > 0.0f){
                giraAlas -= giraAlasOffset * deltaTime;
            }
            else{
                gira = !gira;
            }
        }
        model = modelauxDragon;
        model = glm::translate(model, glm::vec3(0.5f, 1.0, -0.5));
        model = glm::rotate(model, -giraAlas * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
        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));
        AlaDerecha_M.RenderModel();

        model = modelauxDragon;
        model = glm::translate(model, glm::vec3(0.5f, 1.0, 0.5));
        model = glm::rotate(model, giraAlas * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
        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));
        AlaIzquierda_M.RenderModel();
```

```
1064        model = modelauxDragon;
1065        model = glm::translate(model, glm::vec3(-2.0f, 0.65, 0.0));
1066        model = glm::rotate(model, sin(glm::radians(angulovaria)), glm::vec3(1.0f, 0.0f, 0.0f));
1067        Material_brillante.UseMaterial(uniformSpecularIntensity, uniformShininess);
1068        /*color = glm::vec3(0.0f, 1.0f, 0.0f);
1069        glUniform3fv(uniformColor, 1, glm::value_ptr(color));*/
1070        glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
1071        CabezaEnMedio_M.RenderModel();
1072
1073        model = modelauxDragon;
1074        model = glm::translate(model, glm::vec3(-1.7f, 0.45, 0.75));
1075        model = glm::rotate(model, sin(glm::radians(angulovaria)) , glm::vec3(0.0f, 0.0f, 1.0f));
1076        Material_brillante.UseMaterial(uniformSpecularIntensity, uniformShininess);
1077        /*color = glm::vec3(0.0f, 1.0f, 0.0f);
1078        glUniform3fv(uniformColor, 1, glm::value_ptr(color));*/
1079        glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
1080        CabezaIzquierdaAbajo_M.RenderModel();
1081
1082        model = modelauxDragon;
1083        model = glm::translate(model, glm::vec3(-1.7f, 0.38, -0.82));
1084        model = glm::rotate(model, sin(glm::radians(angulovaria)), glm::vec3(0.0f, 1.0f, 1.0f));
1085        Material_brillante.UseMaterial(uniformSpecularIntensity, uniformShininess);
1086        /*color = glm::vec3(0.0f, 1.0f, 0.0f);
1087        glUniform3fv(uniformColor, 1, glm::value_ptr(color));*/
1088        glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
1089        CabezaDerechaAbajo_M.RenderModel();
1090
1091        model = modelauxDragon;
1092        model = glm::translate(model, glm::vec3(-1.3f, 1.5, 0.55));
1093        model = glm::rotate(model, cos(glm::radians(angulovaria)), glm::vec3(1.0f, 0.0f, 1.0f));
1094        Material_brillante.UseMaterial(uniformSpecularIntensity, uniformShininess);
1095        /*color = glm::vec3(0.0f, 1.0f, 0.0f);
1096        glUniform3fv(uniformColor, 1, glm::value_ptr(color));*/
1097        glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
1098        CabezaIzquierdaArriba_M.RenderModel();
1099
1100        model = modelauxDragon;
1101        model = glm::translate(model, glm::vec3(-1.3f, 1.5, -0.60));
1102        model = glm::rotate(model, cos(glm::radians(angulovaria*3)), glm::vec3(0.0f, 1.0f, 0.0f));
1103        Material_brillante.UseMaterial(uniformSpecularIntensity, uniformShininess);
1104        /*color = glm::vec3(0.0f, 1.0f, 0.0f);
1105        glUniform3fv(uniformColor, 1, glm::value_ptr(color));*/
1106        glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
1107        CabezaDerechaArriba_M.RenderModel();
```

Link al video demostrativo:

https://mega.nz/file/wt0x3BaC#9IYw9MpJHvTiORgfJz9vS49WVmJzP3MD6I242JMH0J0

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

En esta ocasión no hubo problemas como tal, sino más bien saber qué funciones introducir para que las cabezas del dragón se movieran.

3.- Conclusión:

a. Los ejercicios del reporte: Complejidad, Explicación.

La complejidad fue relativamente media, lo más difícil fue separar el modelo en muchas partes, texturizarlo y arreglar los huecos. Fue más tardado que complicado.

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

Me gustó esta práctica y creo que entendí cosas que antes no había entendido.

c. Conclusión

En esta ocasión los objetos de la práctica, a mi parecer, se cumplieron en su totalidad, el dragón vuela en dos direcciones y gira al cambiar la misma y también presenta movimientos independientes en cada una de las cabezas. Por otro lado, el letrero muestra el mensaje solicitado.