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 04

NOMBRE COMPLETO: Arroyo Ramìrez Carlos Alberto

Nº de Cuenta: 320185865

GRUPO DE LABORATORIO: 03

GRUPO DE TEORÍA: 04

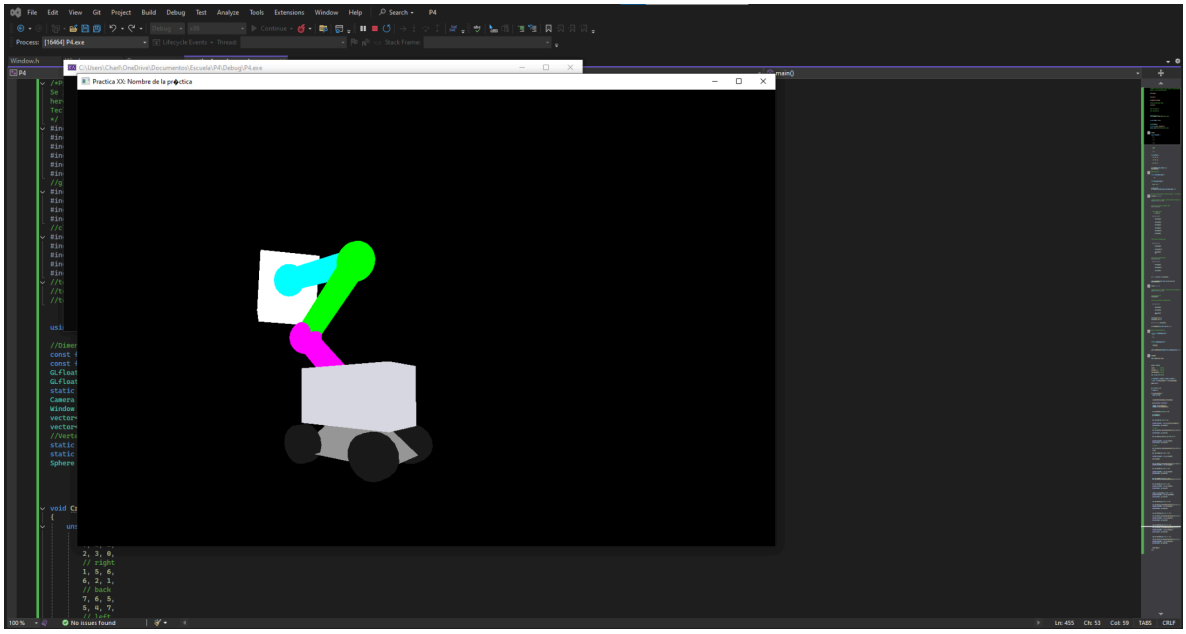
SEMESTRE 2026-1

FECHA DE ENTREGA LÍMITE: 20-09-2025

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.



Para esta práctica terminamos la grúa, para eso fue necesario crear una nueva matriz que la llame “root” con el objetivo de tener una matriz extra para empezar a hacer la parte de abajo de la grúa que consiste en una pirámide cuadrangular y las ruedas, la parte del codigo donde la agregue fue el siguiente:

```
glm::mat4 root(1.0f);
root = glm::translate(root, glm::vec3(0.0f, 5.5f, -4.0f));

glm::mat4 model(1.0f);
glm::mat4 modelaux(1.0f);
glm::mat4 otromodelo(1.0f);
```

Para cada elemento de la grúa, los códigos eran similares, aquellos que contenían movimiento se agregaron líneas adicionales, como el caso de las ruedas, cada una se controla de manera independiente con distintas teclas, el código donde se definen las ruedas es el siguiente:

```
// rueda izquierda trasera
glm::mat4 rueda = root;
rueda = glm::translate(rueda, glm::vec3(+2.0f, -2.5f, +2.0f));
rueda = glm::rotate(rueda, glm::radians(90.0f), glm::vec3(0, 0, 1));
rueda = glm::rotate(rueda, glm::radians(90.0f), glm::vec3(1, 0, 0));
rueda = glm::rotate(rueda, glm::radians(mainWindow.getarticulacion5()), glm::vec3(0, 1, 0)); // giro con K
rueda = glm::scale(rueda, glm::vec3(1.0f, 0.7f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(rueda));
color = glm::vec3(0.1f, 0.1f, 0.1f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[2]->RenderMeshGeometry();
```

El resto de ruedas sigue el mismo código pero cambiando la traslación, tambien modifique el número de caras para el cilindro para que se vean mas redondas:

```
int main() {
    mainWindow = Window(800, 600);
    mainWindow.Initialise();

    CrearCubo();                    // meshList[0]
    CrearPiramideTriangular();      // meshList[1]
    CrearCilindro(16, 1.0f);         // meshList[2]
    CrearCono(25, 2.0f);            // meshList[3]
    CrearPiramideCuadrangular(); // meshList[4]
    CreateShaders();
```

Finalmente el código adicional al ejercicio en clase fue la creación de la base con una pirámide cuadrangular, la defini de la siguiente forma:

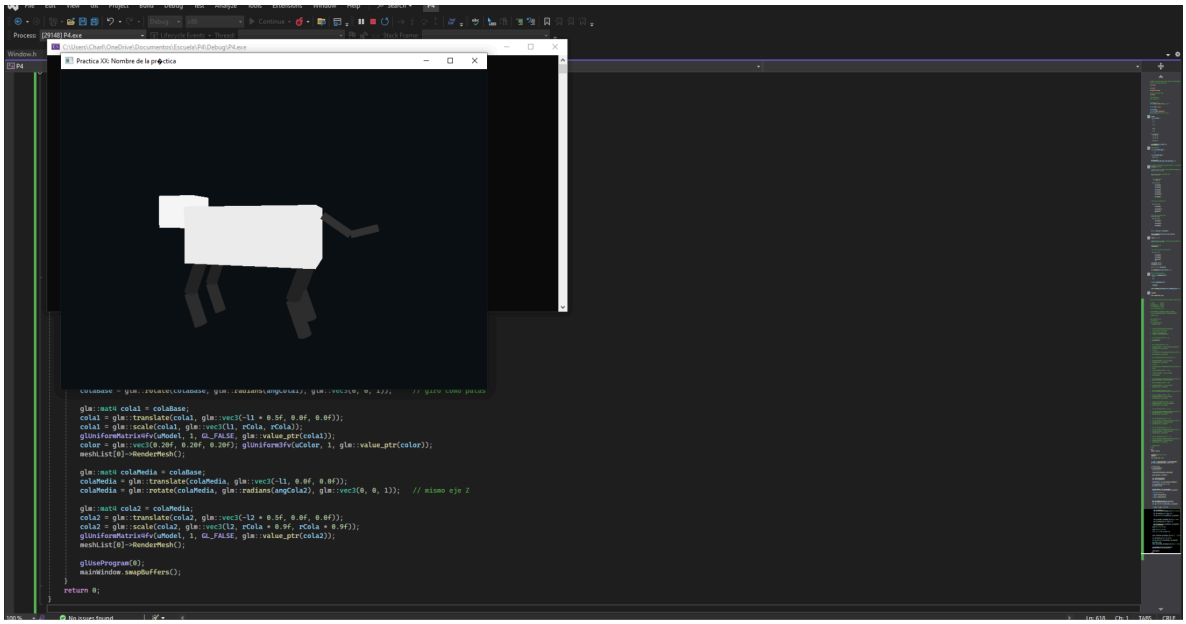
```
// base
otromodelo = root;
otromodelo = glm::translate(otromodelo, glm::vec3(0.0f, -1.5f, 0.0f));
otromodelo = glm::scale(otromodelo, glm::vec3(5.0f, 3.0f, 3.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(otromodelo));
color = glm::vec3(0.6f, 0.6f, 0.6f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[4]->RenderMesh();
```

Para el segundo ejercicio decidí hacer una vaca, la cual como centro decidí poner el torso.

```
glm::mat4 root(1.0f);
root = glm::translate(root, glm::vec3(0.0f, 1.8f, -6.0f));

glm::mat4 model(1.0f), base(1.0f);

// torso
model = root; base = model;
model = glm::scale(model, glm::vec3(6.0f, 2.6f, 3.2f));
glUniformMatrix4fv(uModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.92f, 0.92f, 0.92f); glUniform3fv(uColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh();
```



Solo necesite cubos y cilindros, así que en este main no agregue el resto de figuras geométricas en el main:

```
int main() {
    // ventana
    mainWindow = Window(800, 600);
    mainWindow.Initialise();

    // geometrías
    CrearCubo();                  // meshList[0] = cubo
    CrearCilindro(48, 0.55f); // meshList[1] = cilindro
    CreateShaders();
```

Las patas y cola siguieron una estructura similar, las matrices las nombre como partes del cuerpo para mayor entendimiento:

```
// patas (dos segmentos)
float largoSup = 1.6f, largoInf = 1.6f, radio = 0.55f;

auto pata = [&](glm::vec3 p) {
    glm::mat4 cadera = base;
    cadera = glm::translate(cadera, p);
    cadera = glm::rotate(cadera, glm::radians(angSup), glm::vec3(0, 0, 1)); // igual que patas

    glm::mat4 muslo = cadera;
    muslo = glm::translate(muslo, glm::vec3(0, -largoSup * 0.5f, 0));
    muslo = glm::scale(muslo, glm::vec3(radio, largoSup, radio));
    glUniformMatrix4fv(uModel, 1, GL_FALSE, glm::value_ptr(muslo));
    color = glm::vec3(0.15f, 0.15f, 0.15f); glUniform3fv(uColor, 1, glm::value_ptr(color));
    meshList[1]->RenderMeshGeometry();

    glm::mat4 rodilla = cadera;
    rodilla = glm::translate(rodilla, glm::vec3(0, -largoSup, 0));
    rodilla = glm::rotate(rodilla, glm::radians(angInf), glm::vec3(0, 0, 1)); // mismo eje

    glm::mat4 pierna = rodilla;
    pierna = glm::translate(pierna, glm::vec3(0, -largoInf * 0.5f, 0));
    pierna = glm::scale(pierna, glm::vec3(radio, largoInf, radio));
    glUniformMatrix4fv(uModel, 1, GL_FALSE, glm::value_ptr(pierna));
    color = glm::vec3(0.18f, 0.18f, 0.18f); glUniform3fv(uColor, 1, glm::value_ptr(color));
    meshList[1]->RenderMeshGeometry();
    };
```

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

Lo más complicado eran acomodar y ajustar todas las figuras, también los ángulos de rotación se me complicaron, las extremidades giraban, mal, así que tuve que cambiarlo.

3.- Conclusión:

a. Los ejercicios del reporte: Complejidad, Explicación.
b. Comentarios generales: Faltó explicar a detalle, ir más lento en alguna explicación, otros comentarios y sugerencias para mejorar desarrollo de la práctica
c. Conclusión

En esta práctica armé dos modelos jerárquicos (la grúa y la “vaca”) y entendí que el truco no es dibujar formas sino colgar todo de una raíz y respetar el orden de transformaciones. Cuando coloqué bien los pivotes (por ejemplo, en la base de la cola o en las caderas) y unifiqué el eje de rotación, las animaciones empezaron a verse naturales. También me sirvió mucho separar cada parte en pasos cortos: trasladar al pivote, rotar, luego escalar, y sólo entonces mandar la matriz modelo al shader junto con la vista y la proyección.

Otro aprendizaje fue la coherencia de controles: con G muevo todas las caderas, con H todas las rodillas, y con otras teclas controlo cabeza y cola. Eso me simplificó probar poses sin enredarme. En el ejercicio en clase me perdí un poco de la explicación por tratar de hacer mi código, pero ya después nos lo compartió, me hubiera gustado ya tener el código y solo ir viendo el desarrollo en la clase.

La clave fue armar todo con jerarquías claras, pivotes bien puestos y el orden correcto de transformaciones.

1. Bibliografía en formato APA

GLFW. (s. f.). Input guide. https://www.glfw.org/docs/3.3/input_guide.html

Khronos Group. (2021, 1 de febrero). Viewing and Transformations. OpenGL Wiki. https://www.khronos.org/opengl/wiki/Viewing_and_Transformations.

Khronos Group. (2012, 3 de agosto). Coordinate Transformations. OpenGL Wiki. https://www.khronos.org/opengl/wiki/Coordinate_Transformations