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: MORENO SANTOYO MARIANA

Nº de Cuenta: 319170252

GRUPO DE LABORATORIO: 11

GRUPO DE TEORÍA: 04

SEMESTRE 2025-2

FECHA DE ENTREGA LÍMITE: 13/03/2024

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.

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

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*

1. *Bibliografía en formato APA*

- *Terminar la Grúa con:*
  - *cuerpo(prisma rectangular)*
  - *base (pirámide cuadrangular)*
  - *4 llantas(4 cilindros) con teclado se pueden girar las 4 llantas por separado]*

Para realizar este ejercicio de la práctica, primero fue necesario corregir el error cometido en el ejercicio previo, ya que se estaba reiniciando la matriz y esto rompía la jerarquía del procedimiento. Aunque no se muestra una captura, en el resto del código no hay ningún reinicio de la matriz.

Posteriormente, para llevar a cabo el ejercicio, se optó por utilizar únicamente dos matrices auxiliares: una para la parte superior y otra para la parte inferior. Por esta razón, en esta parte de la práctica solo se solicitaron las ruedas y la base. El reporte se enfoca en este aspecto.

```
glm::mat4 model(1.0);//Inicializar matriz de Modelo 4x4
glm::mat4 modelaux(1.0);//Inicializar matriz de Modelo 4x4 auxiliar para la jerarquía
glm::mat4 modelaux2(1.0);//Inicializar matriz de Modelo 4x4 auxiliar para la jerarquía
```

CODIGO PARA REALIZAR LA BASE (PIRAMIDE TRIANGULAR)

```
//*********************************************************************************************************************
//                                        CREACION DE LA BASE DONDE VAN LAS RUEDAS DE LA GRUA
//*********************************************************************************************************************

model = glm::translate(model, glm::vec3(0.0f, -2.0f, 0.0f));

modelaux2 = model;

model = glm::scale(model, glm::vec3(4.0f, 1.0f, 4.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));

color = glm::vec3(0.5f, 0.5f, 0.5f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));

meshList[4]->RenderMesh();
model = modelaux2;
```

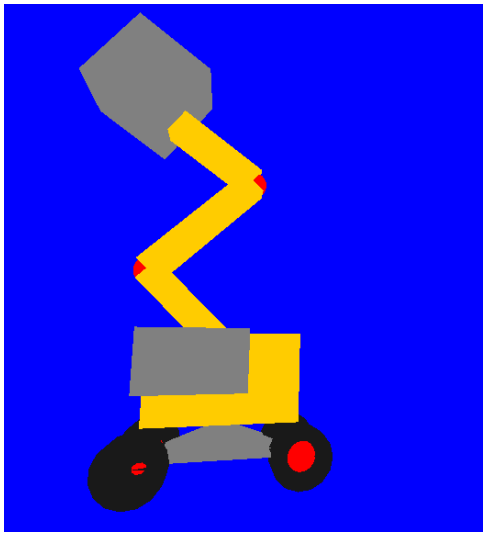
CODIGO PARA LA CREACION DE LAS RUEDAS

Para realizar esta parte, se utilizó la jerarquía para modelar las ruedas. Estas se crearon con una esfera que simula el eje, aunque en este caso el cilindro no fue alterado significativamente para mantener su forma perfecta. Esto se hizo porque, de haberlo modificado, la rotación de las ruedas no sería perceptible a simple vista. La jerarquía se empleó exclusivamente para gestionar las rotaciones de cada rueda, permitiendo que cada una gire de manera independiente con telas.

```
//**************************************Creacion de la rueeda lateral izquierda trasera de la grua******

model = glm::translate(model, glm::vec3(2.5f, -0.5f, 2.0f));

modelaux2 = model;


model = glm::rotate(model, glm::radians(mainWindow.getarticulacion5()), glm::vec3(0.0f, 0.0f, 1.0f));

model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));

color = glm::vec3(1.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));

sp.render();

model = glm::rotate(model, glm::radians(90.0f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(2.0f, 2.0f, 2.0f));

glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));

color = glm::vec3(0.1f, 0.1f, 0.1f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));

meshList[2]->RenderMeshGeometry();
model = modelaux2;

//**************************************Creacion de la rueeda lateral derecha trasera de la grua********

model = glm::translate(model, glm::vec3(0.0f, 0.0f, -4.0f));

modelaux2 = model;

model = glm::rotate(model, glm::radians(mainWindow.getarticulacion6()), glm::vec3(0.0f, 0.0f, 1.0f));

model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));

color = glm::vec3(1.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));

sp.render();

model = glm::translate(model, glm::vec3(0.0f, 0.0f, 1.0f));

model = glm::rotate(model, glm::radians(90.0f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(2.0f, 2.0f, 2.0f));

glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));

color = glm::vec3(0.1f, 0.1f, 0.1f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));

meshList[2]->RenderMeshGeometry();
model = modelaux2;
```

```
//*************************************Creacion de la rueeda lateral derecha frontal de la grua**********************************

model = glm::translate(model, glm::vec3(-5.0f, 0.0f, 0.0f));


modelaux2 = model;

model = glm::rotate(model, glm::radians(mainWindow.getarticulacion7()), glm::vec3(0.0f, 0.0f, 1.0f));

model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));

color = glm::vec3(1.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));

sp.render();
model = glm::translate(model, glm::vec3(0.0f, 0.0f, 1.0f));

model = glm::rotate(model, glm::radians(90.0f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(2.0f, 2.0f, 2.0f));

glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));

color = glm::vec3(0.1f, 0.1f, 0.1f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));

meshList[2]->RenderMeshGeometry();
model = modelaux2;
//*************************************Creacion de la rueeda lateral izquierda frontal de la grua**********************************

model = glm::translate(model, glm::vec3(0.0f, 0.0f, 4.0f));

modelaux2 = model;

model = glm::rotate(model, glm::radians(mainWindow.getarticulacion8()), glm::vec3(0.0f, 0.0f, 1.0f));

model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));

color = glm::vec3(1.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));

sp.render();

model = glm::translate(model, glm::vec3(0.0f, 0.0f, 1.0f));

model = glm::rotate(model, glm::radians(90.0f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(2.0f, 2.0f, 2.0f));

glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));

color = glm::vec3(0.1f, 0.1f, 0.1f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));

meshList[2]->RenderMeshGeometry();

model = modelaux;
```

RESULTADO:



- ***Crear un animal robot 3d***
    - ***Intanciando cubos, pirámides, cilindros, conos, esferas:***
        - ***4 patas articuladas en 2 partes (con teclado se puede mover las dos articulaciones de cada pata)***
        - ***cola articulada o 2 orejas articuladas. (con teclado se puede mover la cola o cada oreja independiente***

IMÁGENES DE REFERENCIA:

Para la realización del perro robot, se tomaron como referencia dos imágenes. Como se observa en ellas, los perros tienen las patas articuladas en tres secciones. En este caso, se optó por un diseño similar, con tres articulaciones en cada pata: una que simula la articulación de la cadera, otra en el tobillo con movimiento en sentido inverso, y una tercera que permite un balanceo lateral, imitando el movimiento de un perro feliz. Además, se agregó una articulación en el cuello.

Basándose en el color del perro de la segunda imagen, se decidió modelar el cuerpo y sus articulaciones con esferas, mientras que las extremidades y la cola se representaron con cilindros deformados. Como detalle adicional, se incluyó la nariz.

Para la implementación, se utilizaron cuatro matrices auxiliares: una para cada pata y una específicamente para la cola. Por conveniencia, se agregó una matriz extra para la cabeza, permitiendo heredar detalles como las orejas y la nariz.

INICIALIZACION

```
//*********************************************************REALIZACION DEL PERRO ROBOT **************

model = glm::mat4(1.0);

model = glm::translate(model, glm::vec3(0.0f, 0.0f, -4.0f));

model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));

glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
```

TORSO

```
//************************************** TORSO ***************************************************************************
color = glm::vec3(0.91f, 0.08f, 0.48f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
sp.render();

modelaux = model;

model = glm::translate(model, glm::vec3(2.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(1.5f, 1.5f, 1.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(1.0f, 0.75f, 0.8f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));

sp.render();

model = modelaux;


modelaux = model;
model = glm::translate(model, glm::vec3(-2.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(1.5f, 1.5f, 1.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(1.0f, 0.75f, 0.8f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));

sp.render();
model = modelaux;
```

PATA1

La realización de las patas fue prácticamente la misma en todas, con la única diferencia de algunas traslaciones. Por esta razón, en el reporte solo se anexa el código correspondiente a las tres articulaciones de las patas, omitiendo el movimiento de traslación de la pata inicial. En el código, la estructura de la pata se repite.

```
//*********************************************ARTICULACION DE LA PATITA 01 ***************************

model = glm::translate(model, glm::vec3(2.0f, 0.0f, 1.5f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()), glm::vec3(0.0f, 0.0f, 1.0f));

modelaux2 = model;

model = glm::scale(model, glm::vec3(0.7f, 0.7f, 0.7f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));

color = glm::vec3(1.0f, 0.0f, 0.5f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));

sp.render();

model = glm::translate(model, glm::vec3(0.0f, -1.5f, -0.1f));

model = glm::rotate(model, glm::radians(90.0f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(1.2f, 0.6, 2.0f));

glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));

color = glm::vec3(0.96f, 0.45f, 0.63f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));

meshList[2]->RenderMeshGeometry();

model = modelaux2;
```

```
//*********************************************ARTICULACION DE LA PATITA 02 ***************************

model = glm::translate(model, glm::vec3(0.0f, -2.3f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()), glm::vec3(0.0f, 0.0f, -1.0f));

modelaux2 = model;

model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));

color = glm::vec3(1.0f, 0.0f, 0.5f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));

sp.render();

model = glm::translate(model, glm::vec3(0.0f, -1.5f, -0.1f));

model = glm::rotate(model, glm::radians(90.0f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(1.2f, 1.0f, 1.5f));

glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));

color = glm::vec3(0.96f, 0.45f, 0.63f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));

meshList[2]->RenderMeshGeometry();


model = modelaux2;

//*********************************************ARTICULACION DE LA PATITA 03 ***************************

model = glm::translate(model, glm::vec3(0.0f, -1.5f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion3()), glm::vec3(0.0f, 0.0f, 1.0f));

modelaux2 = model;

model = glm::scale(model, glm::vec3(0.4f, 0.4f, 0.45f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));

color = glm::vec3(1.0f, 0.0f, 0.5f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));

sp.render();

model = glm::translate(model, glm::vec3(1.3f, -1.0f, 0.3f));

//model = glm::rotate(model, glm::radians(90.0f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(2.0f, 0.5f, 1.5f));

glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));

color = glm::vec3(0.96f, 0.45f, 0.63f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));

meshList[2]->RenderMeshGeometry();

model = modelaux;
```

COLA

```
//*********************************************ARTICULACION DE LA colita ***********************************

model = glm::translate(model, glm::vec3(-3.0f, 1.0f, 0.0));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion13()), glm::vec3(1.0f, 0.0f, 0.0f));

modelaux6 = model;

model = glm::scale(model, glm::vec3(0.7f, 0.7f, 0.7f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));

color = glm::vec3(1.0f, 0.0f, 0.5f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));

sp.render();

model = glm::translate(model, glm::vec3(-0.70f, 1.0f, 0.0));
model = glm::rotate(model, glm::radians(90.0f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(60.0f), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.5f, 0.25f, 1.0f));

glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));

color = glm::vec3(0.96f, 0.45f, 0.63f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));

meshList[2]->RenderMeshGeometry();

model = modelaux;
```

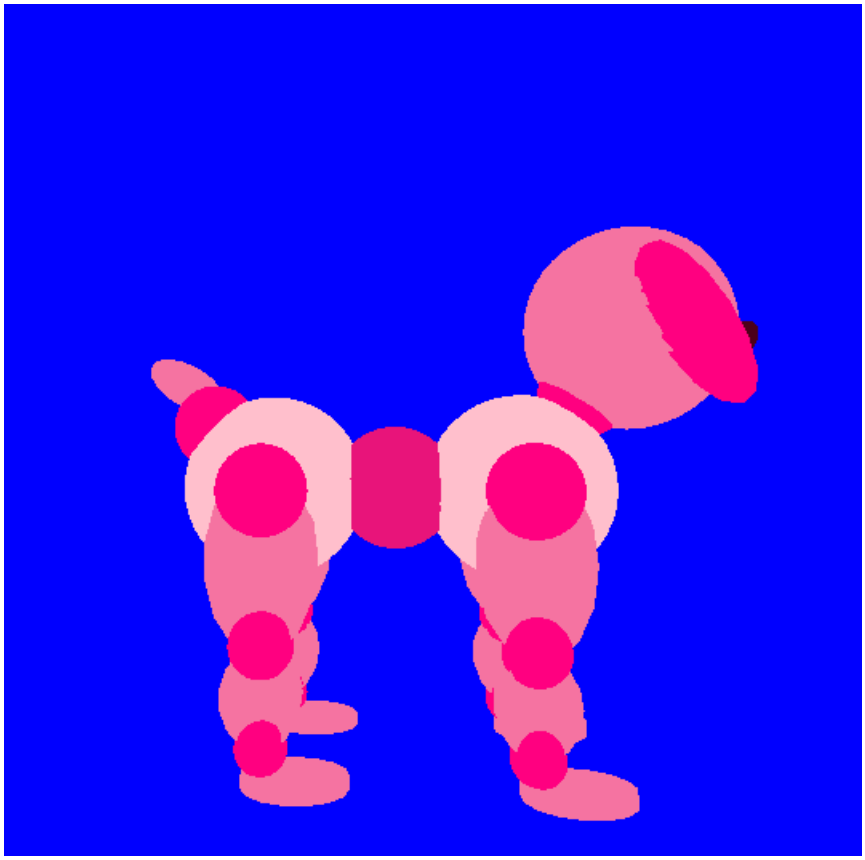
CABEZA

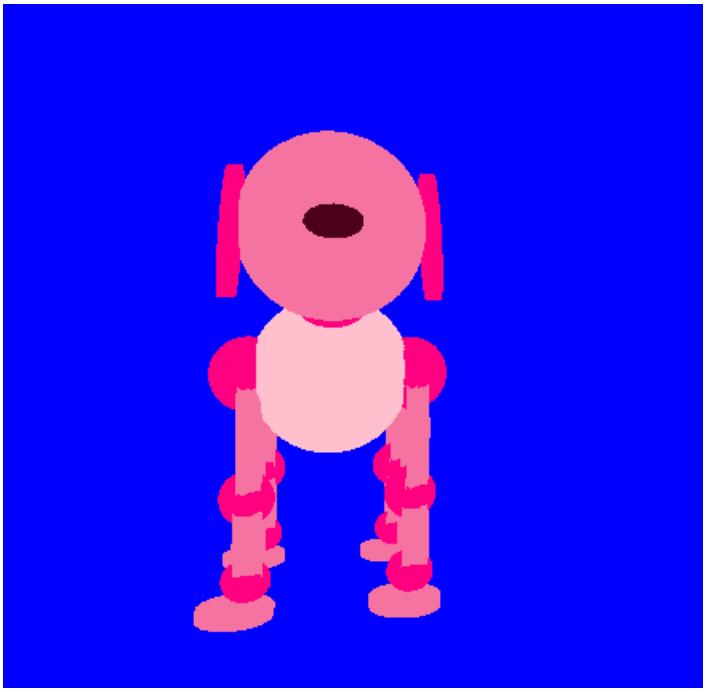
```
//*********************************************************************articulacion del cuello y cabeza

model = glm::translate(model, glm::vec3(3.0f, 1.5f, 0.0));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion14()), glm::vec3(0.0f, 1.0f, 0.0f));

modelaux7 = model;

model = glm::scale(model, glm::vec3(0.7f, 0.7f, 0.7f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));

color = glm::vec3(1.0f, 0.0f, 0.5f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));

sp.render();

model = glm::translate(model, glm::vec3(1.0f, 1.5f, 0.0));
model = glm::scale(model, glm::vec3(2.3f, 2.3f, 2.3f));

glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));

color = glm::vec3(0.96f, 0.45f, 0.63f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));

sp.render();
```

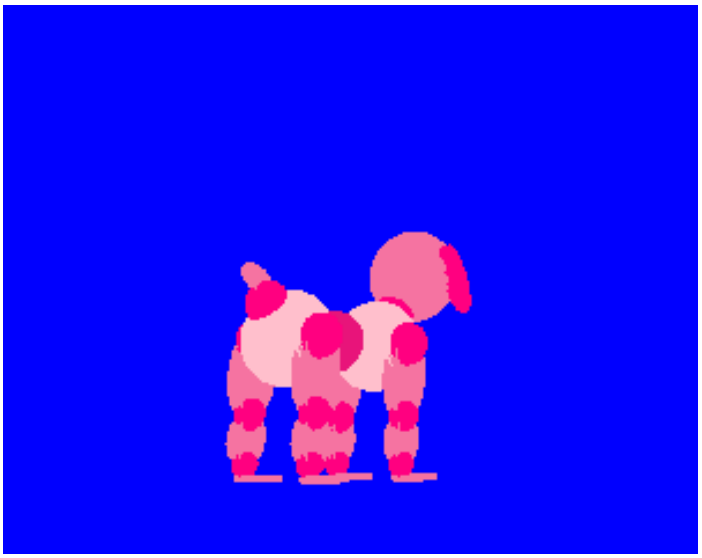
OREJAS

```
model = modelaux7;

model = glm::translate(model, glm::vec3(1.2f, 0.9f, 1.6f));

modelaux7 = model;

model = glm::rotate(model, glm::radians(90.0f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(30.0f), glm::vec3(0.0f, 1.0f, 0.0f));

model = glm::scale(model, glm::vec3(0.5f, 0.25f, 1.3f));

glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));

color = glm::vec3(1.0f, 0.0f, 0.5f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));

meshList[2]->RenderMeshGeometry();

model = modelaux7;

modelaux7 = model;

model = glm::rotate(model, glm::radians(90.0f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(30.0f), glm::vec3(0.0f, 1.0f, 0.0f));

model = glm::translate(model, glm::vec3(0.0f, -3.2f, 0.0f));
model = glm::scale(model, glm::vec3(0.5f, 0.25f, 1.3f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(1.0f, 0.0f, 0.5f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));

meshList[2]->RenderMeshGeometry();

model = modelaux7;

modelaux7 = modelaux;

modelaux7 = model;

model = glm::rotate(model, glm::radians(90.0f), glm::vec3(0.0f, 0.0f, 1.0f));

        model = glm::translate(model, glm::vec3(0.0f, -1.2f, -1.6f));
model = glm::scale(model, glm::vec3(0.25f, 0.25f,0.45f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));


color = glm::vec3(0.3f, 0.0f, 0.1f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));

meshList[2]->RenderMeshGeometry();
```

RESULTADO:







Conclusiones

En esta práctica, se logró implementar una estructura jerárquica para la construcción y animación de un perro robot, asegurando que sus movimientos fueran realistas y articulados de manera adecuada. Se corrigieron errores previos, como la reinicialización de la matriz, lo que permitió mantener la jerarquía funcional del modelo.

Para la animación, se optó por utilizar matrices auxiliares que facilitaron la manipulación de las distintas partes del modelo. Se implementaron tres articulaciones en cada pata, respetando la referencia visual de los perros reales, y se añadieron movimientos adicionales en el cuello y la cola para mejorar el dinamismo del modelo.

En cuanto al modelado, se utilizó una combinación de esferas para el cuerpo y articulaciones, junto con cilindros deformados para las extremidades y la cola, logrando así una apariencia más orgánica. Además, para optimizar el código, se reutilizó la misma estructura en todas las patas, variando únicamente las traslaciones necesarias.

En general, la práctica permitió comprender la importancia de la jerarquía en la animación de estructuras complejas y demostró cómo la organización del código influye en la eficiencia y claridad del proyecto.