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 4

NOMBRE COMPLETO: Araiza Valdés Diego Antonio

Nº de Cuenta: 423032833

GRUPO DE LABORATORIO: 02

GRUPO DE TEORÍA: 06

SEMESTRE: 2026-1

FECHA DE ENTREGA LÍMITE: 21/09/2025

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.
   1.1. Terminar la Grúa con:
   -cuerpo (prisma rectangular)
   -base (pirámide cuadrangular)
   -4 llantas (4 cilindros) con teclado se pueden girar cada una de las 4 llantas por separado

```
if (key == GLFW_KEY_G) { ... }
if (key == GLFW_KEY_H) { ... }
if (key == GLFW_KEY_J) { ... }
if (key == GLFW_KEY_C)
{
    theWindow->articulacion5 += 10.0;
}
if (key == GLFW_KEY_V)
{
    theWindow->articulacion6 += 10.0;
}
if (key == GLFW_KEY_B)
{
    theWindow->articulacion7 += 10.0;
}
if (key == GLFW_KEY_N)
{
    theWindow->articulacion8 += 10.0;
}
```

```
articulacion1 = 0.0f;
articulacion2 = 0.0f;
articulacion3 = 0.0f;
articulacion4 = 0.0f;
articulacion5 = 0.0f;
articulacion6 = 0.0f;
articulacion7 = 0.0f;
articulacion8 = 0.0f;
```

```
    GLfloat getarticulacion1() { return articulacion1; }
    GLfloat getarticulacion2() { return articulacion2; }
    GLfloat getarticulacion3() { return articulacion3; }
    GLfloat getarticulacion4() { return articulacion4; }
    GLfloat getarticulacion5() { return articulacion5; }
    GLfloat getarticulacion6() { return articulacion6; }
    GLfloat getarticulacion7() { return articulacion7; }
    GLfloat getarticulacion8() { return articulacion8; }

    ~Window();
private:
    GLFWwindow *mainWindow;
    GLint width, height;
    GLfloat rotax,rotay,rotaz, articulacion1, articulacion2, articulacion3,
    articulacion4, articulacion5, articulacion6, articulacion7, articulacion8;
```

Primero se agregaron las articulaciones faltantes (7 y 8) en los archivos Window.h y Window.cpp para poder controlar los 3 brazos, la canasta, y las 4 ruedas. Además, para las articulaciones 5 y 6 se cambio las letras a C y V

```
CrearCubo();//índice 0 en MeshList
CrearPiramideTriangular();//índice 1 en MeshList
CrearCilindro(10, 1.0f);//índice 2 en MeshList
CrearCono(25, 2.0f);//índice 3 en MeshList
CrearPiramideCuadrangular();//índice 4 en MeshList
CreateShaders();
```

Luego cambiamos el número de resolución del cilindro a uno más elevado (10), pero no tanto para que se observen las rotaciones.

```
//CREANDO LA CABINA
model = glm::mat4(1.0f);
model = glm::translate(model, glm::vec3(0.0f, 5.0f, -4.0f));
modelaux = model;
model = glm::scale(model, glm::vec3(5.0f, 3.0f, 2.0f));
model = glm::rotate(model, glm::radians(5.0f), glm::vec3(0.0f, 0.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.8f, 0.8f, 0.8f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
meshList[0]->RenderMesh();
```

Por consiguiente, se rotó la cabina 5 grados sobre el eje de las z para que se asemejara más al dibujo.

```
//CUERPO
model = glm::mat4(1.0f);
model = glm::translate(model, glm::vec3(1.5f, 4.5f, -4.0f));
modelaux = model;
model = glm::scale(model, glm::vec3(7.5f, 4.0f, 1.8f));
model = glm::rotate(model, glm::radians(5.0f), glm::vec3(0.0f, 0.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.7f, 0.7f, 0.7f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh();

//BASE
model = modelaux;
model = glm::translate(model, glm::vec3(0.0f, -2.5f, 0.0f));
modelaux = model;
model = glm::scale(model, glm::vec3(8.5f, 2.5f, 2.8f));
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();

//LLANTA 1
model = modelaux;
model = glm::translate(model, glm::vec3(-4.25f, -1.25f, 1.9f));
model = glm::scale(model, glm::vec3(2.0f, 2.0f, 1.0f));
model = glm::rotate(model, glm::radians(90.0f), glm::vec3(1.0f, 0.0f, 0.0f));
modelaux = model;
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion5()), glm::vec3(0.0f, 1.0f, 0.0f));
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();
```

Posteriormente, se creó el cuerpo siguiendo la estructura de la cabina (también con una rotación de 5 grados), pero con unas dimensiones más altas, largas y un poco más delgadas. Luego, siguiendo la misma estructura, esta vez sin rotar, creamos la base con la pirámide cuadrangular, con una longitud y anchura una unidad mayor al cuerpo, además guardamos la traslación en la matriz modelaux. Para la primera llanta reutilizamos la traslación y además la trasladamos de manera que quede como la llanta delantera derecha, la escalamos y la rotamos, luego guardamos esta configuración en modelaux, luego le asignamos la articulación, le dimos color, pasamos sus matrices al shader y después la dibujamos.

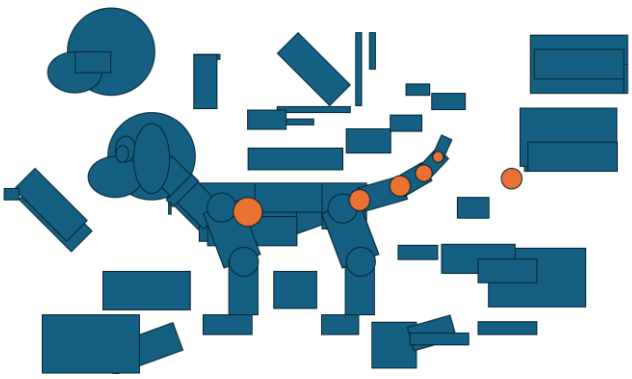
```
//LLANTA 2
model = modelaux;
model = glm::translate(model, glm::vec3(0.0f, -3.8, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion6()), glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.1f, 0.1f, 0.1f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[2]->RenderMeshGeometry();

//LLANTA 3
model = modelaux;
model = glm::translate(model, glm::vec3(4.25f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion7()), glm::vec3(0.0f, 1.0f, 0.0f));
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();
//LLANTA 4
model = modelaux;
model = glm::translate(model, glm::vec3(4.25f, -3.8f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion8()), glm::vec3(0.0f, 1.0f, 0.0f));
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();
```

Por consiguiente, dibujamos las siguientes llantas, reutilizando la configuración de modelaux, a cada una se le asigno su articulación y se trasladó tomando en cuenta la rotación y escalamiento previo, de modo que el eje X se mantiene, el eje Y se vuelve el Z y el Z se vuelve el -Y, así las traslaciones se hicieron sobre el eje X y Y, además, como se escaló sobre el eje X y Y (Ahora X y Z), los movimientos en X se hicieron la mitad de la distancia y los movimientos en Y (antes Z) se hicieron con la longitud normal.

1.2. Crear un animal robot 3d

-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



Primero se diseñó, un esquema del perro para facilitar la construcción del mismo, utilizando medidas aproximadas obtenidas gracias a la herramienta.

Window.h

```
    GLfloat getarticulacion1() { return articulacion1; }
    GLfloat getarticulacion2() { return articulacion2; }
    GLfloat getarticulacion3() { return articulacion3; }
    GLfloat getarticulacion4() { return articulacion4; }
    GLfloat getarticulacion5() { return articulacion5; }
    GLfloat getarticulacion6() { return articulacion6; }
    GLfloat getarticulacion7() { return articulacion7; }
    GLfloat getarticulacion8() { return articulacion8; }
    GLfloat getarticulacion9() { return articulacion9; }

    ~Window();
private:
    GLFWwindow *mainWindow;
    GLint width, height;
    GLfloat rotax,rotay,rotaz, articulacion1, articulacion2, articulacion3,
    articulacion4, articulacion5, articulacion6, articulacion7, articulacion8, articulacion9;
```

```
    // --- variables para el control (ping-pong) de articulacion1 ---
    GLfloat articulacion1Dir;   // 1.0f = incrementar, -1.0f = decrementar
    GLfloat articulacion1Min;   // límite mínimo
    GLfloat articulacion1Max;   // límite máximo
    GLfloat articulacion1Step;  // paso por pulsación (10)
```

Window.cpp

```
articulacion1 = 0.0f;
articulacion2 = 0.0f;
articulacion3 = 0.0f;
articulacion4 = 0.0f;
articulacion5 = 0.0f;
articulacion6 = 0.0f;
articulacion7 = 0.0f;
articulacion8 = 0.0f;
articulacion9 = 0.0f;

// Inicialización de control ping-pong para articulacion1
articulacion1Dir = 1.0f;    // empieza subiendo
articulacion1Min = -30.0f;
articulacion1Max = 30.0f;
articulacion1Step = 5.0f;  // +5° por pulsación
```

```
if (key == GLFW_KEY_F)

    //theWindow->articulacion1 += 10.0;
{
    float next = theWindow->articulacion1 + theWindow->articulacion1Dir * theWindow->articulacion1Step;

    if (next > theWindow->articulacion1Max) {
        theWindow->articulacion1 = theWindow->articulacion1Max;
        theWindow->articulacion1Dir = -1.0f; // invertir dirección para la siguiente pulsación
    }
    else if (next < theWindow->articulacion1Min) {
        theWindow->articulacion1 = theWindow->articulacion1Min;
        theWindow->articulacion1Dir = 1.0f;
    }
    else {
        theWindow->articulacion1 = next;
    }

}
if (key == GLFW_KEY_G) { ... }
if (key == GLFW_KEY_H) { ... }
if (key == GLFW_KEY_J) { ... }
if (key == GLFW_KEY_C) { ... }
if (key == GLFW_KEY_V) { ... }
if (key == GLFW_KEY_B) { ... }
if (key == GLFW_KEY_N) { ... }
if (key == GLFW_KEY_M)
{
    theWindow->articulacion9 += 10.0;
}
```

Se agregó una articulación en los archivos Window.h y Window.cpp para así poder contar con 8 articulaciones para las patas (2 para cada una) y una para la cola. Además, se agregaron e inicializaron las variables 'articulacion1Dir', 'articulacionMin' 'articulacion1Max', 'articulacion1Step' y 'next', para poder controlar la rotación de la articulación 1, de modo que delimitamos la rotación de 30° a -30°, y cada vez que presionemos la letra F comprobará si ya excederíamos el límite de rotación, de no hacerlo rotará 5°, y de hacerlo cambiará el sentido de la rotación y además se asigna el valor de la rotación al valor máximo o mínimo (según sea el caso) de rotación, útil si la rotación máxima o mínima no es múltiplo de nuestro paso (en nuestro caso 30 si es múltiplo de 5).

```
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 modelauxC(1.0);//Inicializar matriz de Modelo 4x4 auxiliar para la jerarquia de la cola
glm::mat4 modelauxPiv(1.0);//Inicializar matriz de Modelo 4x4 auxiliar para la jerarquía del pivote para las articulaciones
glm::mat4 modelauxP(1.0);//Inicializar matriz de Modelo 4x4 auxiliar para la jerarquia de las patas
glm::mat4 modelauxPD1(1.0);//Inicializar matriz de Modelo 4x4 auxiliar para la jerarquía de la pata delantera izquierda
glm::mat4 modelauxPD2(1.0);//Inicializar matriz de Modelo 4x4 auxiliar para la jerarquía de la pata delantera derecha
glm::mat4 modelauxPT1(1.0);//Inicializar matriz de Modelo 4x4 auxiliar para la jerarquía de la pata trasera izquierda
glm::mat4 modelauxPT2(1.0);//Inicializar matriz de Modelo 4x4 auxiliar para la jerarquía de la pata trasera derecha
```

Por consiguiente, agregamos las siguientes matrices para facilitar la creación de la jerarquía y las articulaciones.

```
//CREANDO EL CUERPO SUPERIOR
model = glm::mat4(1.0f);
model = glm::translate(model, glm::vec3(0.0f, 4.5f, -4.0f));
modelaux = model;
modelauxP = model;
model = glm::scale(model, glm::vec3(5.0f, 3.0f, 3.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.2980f, 0.1490f, 0.1137f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
meshList[0]->RenderMesh();

//CREANDO PANZA SUPERIOR
model = modelaux;
model = glm::translate(model, glm::vec3(0.199f, -1.0f, 0.0f));
model = glm::scale(model, glm::vec3(4.6f, 1.5f, 2.8f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.9294f, 0.8078f, 0.6941f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh();

//CREANDO CUERPO MEDIO
model = modelaux;
model = glm::translate(model, glm::vec3(2.65f, 0.75f, 0.0f));
model = glm::scale(model, glm::vec3(4.6f, 1.5f, 2.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.2980f, 0.1490f, 0.1137f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh();
```

Después, iniciamos con la creación del perro, creamos el cuerpo superior, lo trasladamos y a partir de este actualizamos el valor de modelaux y modeluxP para posteriormente crear las patas. Además, se creo la panza y el cuerpo medio.

```
//CREANDO PANZA MEDIO
model = modelaux;
model = glm::translate(model, glm::vec3(2.65f, -0.2f, 0.0f));
model = glm::rotate(model, glm::radians(30.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::scale(model, glm::vec3(3.8f, 1.5f, 2.3f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.9294f, 0.8078f, 0.6941f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh();

//CREANDO CUERPO INFERIOR
model = modelaux;
model = glm::translate(model, glm::vec3(4.95f, 0.3f, 0.0f));
model = glm::scale(model, glm::vec3(2.3f, 2.37f, 2.8f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.2980f, 0.1490f, 0.1137f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh();

//ARTICULACION COLA1
model = modelaux;
model = glm::translate(model, glm::vec3(5.75f, 0.6f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()), glm::vec3(0.0f, 1.0f, 0.0f));
modelauxPiv= 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(0.5f, 0.5f, 0.5f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
sp.render();
model = modelauxPiv;
//CREANDO COLA1
model = glm::translate(model, glm::vec3(1.15f, 0.3f, 0.0f));
modelauxC = model;
model = glm::rotate(model, glm::radians(16.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::scale(model, glm::vec3(2.9f, 1.25f, 1.2f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.2980f, 0.1490f, 0.1137f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh();

//ARTICULACION COLA2
model = modelauxC;
model = glm::translate(model, glm::vec3(0.7f, 0.2f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()), glm::vec3(0.0f, 1.0f, 0.0f));
modelauxPiv = 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(0.5f, 0.5f, 0.5f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
sp.render();
model = modelauxPiv;

//CREANDO COLA2
model = glm::translate(model, glm::vec3(1.20f, 0.6f, 0.0f));
modelauxC = model;
model = glm::rotate(model, glm::radians(30.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::scale(model, glm::vec3(2.0f, 1.06f, 0.9f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.2980f, 0.1490f, 0.1137f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh();
```

```
//ARTICULACION COLA3
model = modelauxC;
model = glm::translate(model, glm::vec3(0.4f, 0.3f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()), glm::vec3(0.0f, 1.0f, 0.0f));
modelauxPiv = model;
model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.3f));
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)); //para cambiar el color del objetos
sp.render();
model = modelauxPiv;

//CREANDO COLA3
model = glm::translate(model, glm::vec3(0.7f, 0.6f, 0.0f));
modelauxC = model;
model = glm::rotate(model, glm::radians(47.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::scale(model, glm::vec3(1.3f, 0.83f, 0.65f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.2980f, 0.1490f, 0.1137f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh();

//ARTICULACION COLA4
model = modelauxC;
model = glm::translate(model, glm::vec3(0.2f, 0.3f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()), glm::vec3(0.0f, 1.0f, 0.0f));
modelauxPiv = model;
model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.3f));
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)); //para cambiar el color del objetos
sp.render();
model = modelauxPiv;
//CREANDO COLA4
model = glm::translate(model, glm::vec3(0.3f, 0.7f, 0.0f));
model = glm::rotate(model, glm::radians(75.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::scale(model, glm::vec3(1.4f, 0.6f, 0.4f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.2980f, 0.1490f, 0.1137f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh();
```

Terminamos de crear el cuerpo y luego seguimos con la cola, para la cual seguimos la estructura de los brazos de la grúa e hicimos uso de modelauxPiv y modelauxC. Es importante mencionar que en cada articulación se hizo uso de la misma función de rotación (articulación 1), así cada parte de la cola girara 5°.

```
//CREANDO CUELLO
model = modelaux;
model = glm::translate(model, glm::vec3(-3.0f, 1.2f, 0.0f));
modelaux = model;
model = glm::rotate(model, glm::radians(-45.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::scale(model, glm::vec3(3.8f, 1.5f, 1.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.2980f, 0.1490f, 0.1137f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh();
```

```
//CREANDO CUELLO2
model = modelaux;
model = glm::translate(model, glm::vec3(0.0f, -0.6f, 0.0f));
model = glm::rotate(model, glm::radians(-45.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::scale(model, glm::vec3(3.8f, 1.5f, 1.4f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.9294f, 0.8078f, 0.6941f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh();

//CREANDO COLLAR
model = modelaux;
model = glm::translate(model, glm::vec3(-0.3f, 0.0f, 0.0f));;
model = glm::rotate(model, glm::radians(-45.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::scale(model, glm::vec3(0.5f, 2.1f, 1.6f));
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));
meshList[0]->RenderMesh();
//CREANDO EBILLA
model = modelaux;
model = glm::translate(model, glm::vec3(-1.0f, -1.1f, 0.0f));;
model = glm::rotate(model, glm::radians(-90.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::scale(model, glm::vec3(0.4f, 0.15f, 0.4f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(1.0f, 0.5960f, 0.1607f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[2]->RenderMeshGeometry();

//CREANDO CRANEO
model = modelaux;
model = glm::translate(model, glm::vec3(-1.875f, 1.875f, 0.0f));
modelaux = model;
model = glm::scale(model, glm::vec3(2.25f, 2.25f, 2.25f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.2980f, 0.1490f, 0.1137f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
sp.render(); //dibuja esfera

//CREANDO HOCICO
model = modelaux;
model = glm::translate(model, glm::vec3(-1.8f, -0.52f, 0.0f));
model = glm::scale(model, glm::vec3(1.4f, 0.7f, 1.12f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.2980f, 0.1490f, 0.1137f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
sp.render(); //dibuja esfera
//CREANDO HOCICO2
model = modelaux;
model = glm::translate(model, glm::vec3(-1.3f, -1.05f, 0.0f));
model = glm::scale(model, glm::vec3(1.4f, 0.7f, 0.9f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.8039f, 0.5607f, 0.3764f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
sp.render(); //dibuja esfera
```

```
//CREANDO NARIZ
model = modelaux;
model = glm::translate(model, glm::vec3(-2.9f, -0.2f, 0.0f));
model = glm::scale(model, glm::vec3(0.35f, 0.2f, 0.366f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.14f, 0.03f, 0.011f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
sp.render(); //dibuja esfera

//CREANDO OREJA1
model = modelaux;
model = glm::translate(model, glm::vec3(0.0f, -0.1f, 2.15f));
model = glm::rotate(model, glm::radians(-15.0f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.9f, 1.8f, 0.3f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.2580f, 0.1240f, 0.0837f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
sp.render(); //dibuja esfera
//CREANDO OREJA2
model = modelaux;
model = glm::translate(model, glm::vec3(0.0f, -0.1f, -2.15f));
model = glm::rotate(model, glm::radians(15.0f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.9f, 1.8f, 0.3f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.2580f, 0.1240f, 0.0837f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
sp.render(); //dibuja esfera
//CREANDO OJO1_1
model = modelaux;
model = glm::translate(model, glm::vec3(-1.85f, 0.37f, 0.75f));
model = glm::rotate(model, glm::radians(-6.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::scale(model, glm::vec3(0.3f, 0.665f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(1.0f, 1.00f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
sp.render(); //dibuja esfera

//CREANDO OJO1_2
model = modelaux;
model = glm::translate(model, glm::vec3(-1.87f, 0.37f, 0.79f));
model = glm::rotate(model, glm::radians(-6.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::rotate(model, glm::radians(-2.0f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.3f, 0.62f, 0.45f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.36f, 0.1490f, 0.1137f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
sp.render(); //dibuja esfera

//CREANDO OJO1_3
model = modelaux;
model = glm::translate(model, glm::vec3(-1.88f, 0.36f, 0.81f));
model = glm::rotate(model, glm::radians(-6.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::rotate(model, glm::radians(-2.0f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.3f, 0.58f, 0.42f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
sp.render(); //dibuja esfera
```

```
//CREANDO OJO2_1
model = modelaux;
model = glm::translate(model, glm::vec3(-1.85f, 0.37f, -0.75f));
model = glm::rotate(model, glm::radians(-6.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::scale(model, glm::vec3(0.3f, 0.665f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(1.0f, 1.00f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
sp.render(); //dibuja esfera

//CREANDO OJO2_2
model = modelaux;
model = glm::translate(model, glm::vec3(-1.87f, 0.37f, -0.79f));
model = glm::rotate(model, glm::radians(-6.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::rotate(model, glm::radians(2.0f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.3f, 0.62f, 0.45f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.36f, 0.1490f, 0.1137f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
sp.render(); //dibuja esfera

//CREANDO OJO2_3
model = modelaux;
model = glm::translate(model, glm::vec3(-1.88f, 0.36f, -0.81f));
model = glm::rotate(model, glm::radians(-6.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::rotate(model, glm::radians(2.0f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.3f, 0.58f, 0.42f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
sp.render(); //dibuja esfera
```

Después de terminar la cola, nos seguimos con el cuello y la cabeza, para la cual solo hicimos uso de modelaux, el cual movimos al centro del cráneo para facilitar la creación de los ojos, orejas y hocico.

```
//ARTICULACION PATA DELANTERA IZQUIERDA1
model = modelauxP;
model = glm::translate(model, glm::vec3(-1.35f, 0.25f, 1.5f));
modelauxPD1 = model;
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()), glm::vec3(0.0f, 0.0f, -1.0f));
modelauxPiv = model;
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.2880f, 0.1390f, 0.1037f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
sp.render();
model = modelauxPiv;

//CREANDO PATA DELANTERA IZQUIERDA1
model = glm::translate(model, glm::vec3(0.53f, -1.38f, 0.25f));
modelauxPD1 = model;
model = glm::rotate(model, glm::radians(21.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::scale(model, glm::vec3(2.0f, 3.0f, 1.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.2980f, 0.1490f, 0.1137f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh();

//ARTICULACION PATA DELANTERA IZQUIERDA2
model = modelauxPD1;
model = glm::translate(model, glm::vec3(0.53f, -1.4f, 0.0f));
modelauxPD1 = model;
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion3()), glm::vec3(0.0f, 0.0f, -1.0f));
modelauxPiv = model;
model = glm::scale(model, glm::vec3(0.75f, 0.75f, 0.66f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.2880f, 0.1390f, 0.1037f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
sp.render();
model = modelauxPiv;
//CREANDO PATA DELANTERA IZQUIERDA2
model = glm::translate(model, glm::vec3(0.0f, -1.38f, 0.0f));
modelauxPD1 = model;
model = glm::scale(model, glm::vec3(1.5f, 2.8f, 1.3f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.2980f, 0.1490f, 0.1137f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh();

//CREANDO PIE PATA DELANTERA IZUQUIERDA
model = modelauxPD1;
model = glm::translate(model, glm::vec3(-0.8f, -1.9f, 0.0f));
model = glm::scale(model, glm::vec3(2.5f, 1.0f, 1.2f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.2280f, 0.0790f, 0.0437f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh();
```

Por consiguiente, empezamos a crear las patas, empezamos con la pata delantera izquierda, para la cual hicimos uso de 2 articulaciones, con sus respectivas rotaciones individuales y de modelauxP y modelauxPD1.

```
//ARTICULACION PATA DELANTERA DERECHA1
model = modelauxP;
model = glm::translate(model, glm::vec3(-1.35f, 0.25f, -1.5f));
modelauxPD2= model;
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion4()), glm::vec3(0.0f, 0.0f, -1.0f));
modelauxPiv = model;
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.2880f, 0.1390f, 0.1037f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
sp.render();
model = modelauxPiv;

//CREANDO PATA DELANTERA DERECHA1
model = glm::translate(model, glm::vec3(0.53f, -1.38f, -0.25f));
modelauxPD2 = model;
model = glm::rotate(model, glm::radians(21.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::scale(model, glm::vec3(2.0f, 3.0f, 1.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.2980f, 0.1490f, 0.1137f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh();

//ARTICULACION PATA DELANTERA DERECHA2
model = modelauxPD2;
model = glm::translate(model, glm::vec3(0.53f, -1.4f, 0.0f));
modelauxPD2 = model;
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion5()), glm::vec3(0.0f, 0.0f, -1.0f));
modelauxPiv = model;
model = glm::scale(model, glm::vec3(0.75f, 0.75f, 0.66f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.2880f, 0.1390f, 0.1037f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
sp.render();
model = modelauxPiv;
//CREANDO PATA DELANTERA DERECHA2
model = glm::translate(model, glm::vec3(0.0f, -1.38f, 0.0f));
modelauxPD2 = model;
model = glm::scale(model, glm::vec3(1.5f, 2.8f, 1.3f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.2980f, 0.1490f, 0.1137f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh();

//CREANDO PIE PATA DELANTERA DERECHA
model = modelauxPD2;
model = glm::translate(model, glm::vec3(-0.8f, -1.9f, 0.0f));
model = glm::scale(model, glm::vec3(2.5f, 1.0f, 1.2f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.2280f, 0.0790f, 0.0437f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh();
//ARTICULACION PATA TRASERA IZQUIERDA1
model = modelauxP;
model = glm::translate(model, glm::vec3(4.9f, 0.25f, 1.5f));
modelauxPD1 = model;
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion6()), glm::vec3(0.0f, 0.0f, -1.0f));
modelauxPiv = model;
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.2880f, 0.1390f, 0.1037f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
sp.render();
model = modelauxPiv;
```

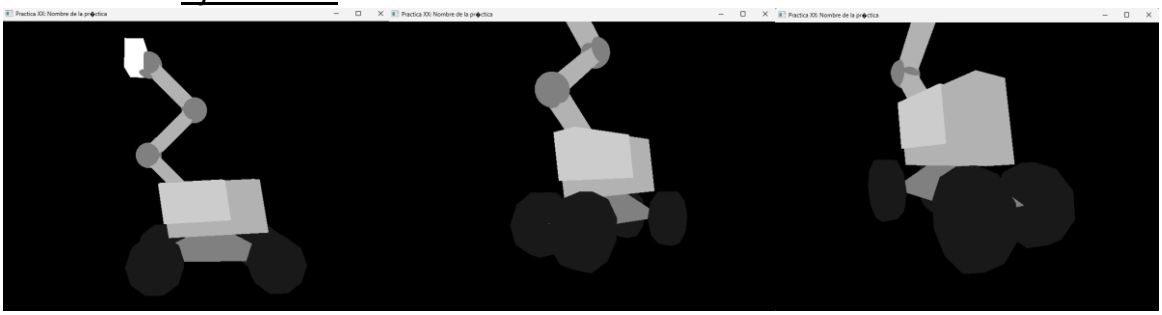
```
//CREANDO PATA TRASERA IZQUIERDA1
model = glm::translate(model, glm::vec3(0.53f, -1.38f, 0.25f));
modelauxPD1 = model;
model = glm::rotate(model, glm::radians(21.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::scale(model, glm::vec3(2.0f, 3.0f, 1.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.2980f, 0.1490f, 0.1137f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh();

//ARTICULACION PATA TRASERA IZQUIERDA2
model = modelauxPD1;
model = glm::translate(model, glm::vec3(0.53f, -1.4f, 0.0f));
modelauxPD1 = model;
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion7()), glm::vec3(0.0f, 0.0f, -1.0f));
modelauxPiv = model;
model = glm::scale(model, glm::vec3(0.75f, 0.75f, 0.66f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.2880f, 0.1390f, 0.1037f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
sp.render();
model = modelauxPiv;

//CREANDO PATA TRASERA IZQUIERDA2
model = glm::translate(model, glm::vec3(0.0f, -1.38f, 0.0f));
modelauxPD1 = model;
model = glm::scale(model, glm::vec3(1.5f, 2.8f, 1.3f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.2980f, 0.1490f, 0.1137f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh();

//CREANDO PIE PATA TRASERA IZQUIERDA
model = modelauxPD1;
model = glm::translate(model, glm::vec3(-0.8f, -1.9f, 0.0f));
model = glm::scale(model, glm::vec3(2.5f, 1.0f, 1.2f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.2280f, 0.0790f, 0.0437f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh();

//ARTICULACION PATA TRASERA DERECHA1
model = modelauxP;
model = glm::translate(model, glm::vec3(4.9f, 0.25f, -1.5f));
modelauxPD1 = model;
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion8()), glm::vec3(0.0f, 0.0f, -1.0f));
modelauxPiv = model;
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.2880f, 0.1390f, 0.1037f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
sp.render();
model = modelauxPiv;

//CREANDO PATA TRASERA DERECHA1
model = glm::translate(model, glm::vec3(0.53f, -1.38f, -0.25f));
modelauxPD1 = model;
model = glm::rotate(model, glm::radians(21.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::scale(model, glm::vec3(2.0f, 3.0f, 1.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.2980f, 0.1490f, 0.1137f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh();
```

```
//ARTICULACION PATA TRASERA DERECHA2
model = modelauxPD1;
model = glm::translate(model, glm::vec3(0.53f, -1.4f, 0.0f));
modelauxPD1 = model;
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion9()), glm::vec3(0.0f, 0.0f, -1.0f));
modelauxPiv = model;
model = glm::scale(model, glm::vec3(0.75f, 0.75f, 0.66f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.2880f, 0.1390f, 0.1037f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
sp.render();
model = modelauxPiv;
//CREANDO PATA TRASERA DERECHA2
model = glm::translate(model, glm::vec3(0.0f, -1.38f, 0.0f));
modelauxPD1 = model;
model = glm::scale(model, glm::vec3(1.5f, 2.8f, 1.3f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.2980f, 0.1490f, 0.1137f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh();

//CREANDO PIE PATA TRASERA DERECHA
model = modelauxPD1;
model = glm::translate(model, glm::vec3(-0.8f, -1.9f, 0.0f));
model = glm::scale(model, glm::vec3(2.5f, 1.0f, 1.2f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.2280f, 0.0790f, 0.0437f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh();

glUseProgram(0);
mainWindow.swapBuffers();
```

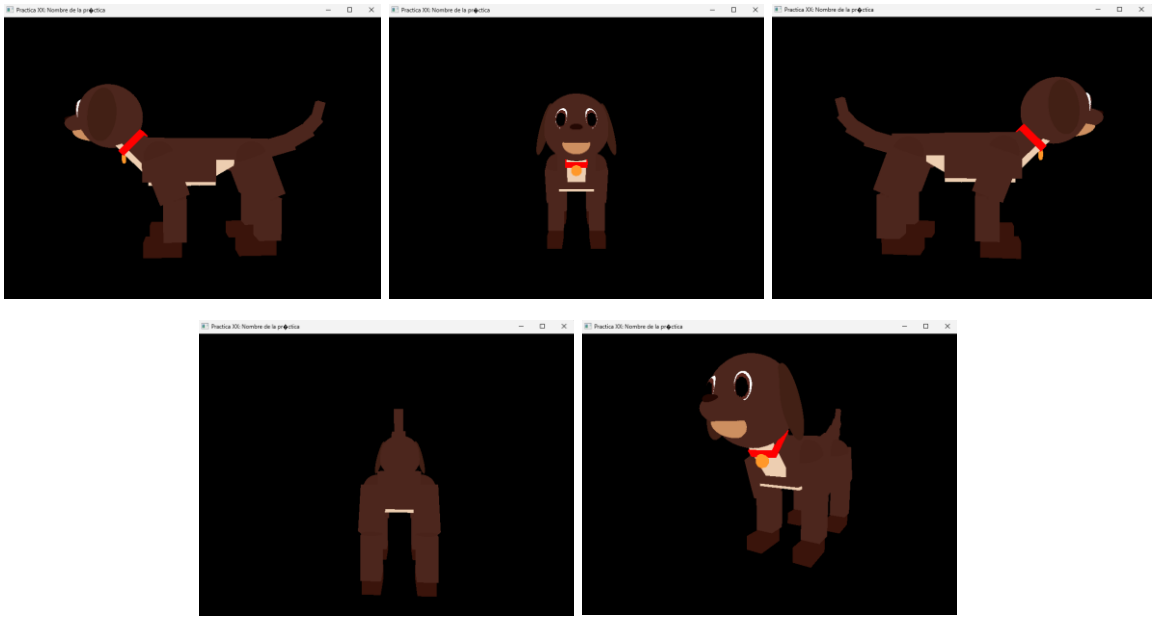
Finalmente, solo copiamos la estructura de la primera pata para hacer las otras 3,desplazando la primer pata y articulación en sentido contrario en Z y en sentido positivo en X. Para cada una de las patas hicimos uso de 2 articulaciones, con sus respectivas rotaciones individuales, de modelauxP y de su respectiva matriz de apoyo (modelauxPD2, modelauxPT1, modelauxPT2).

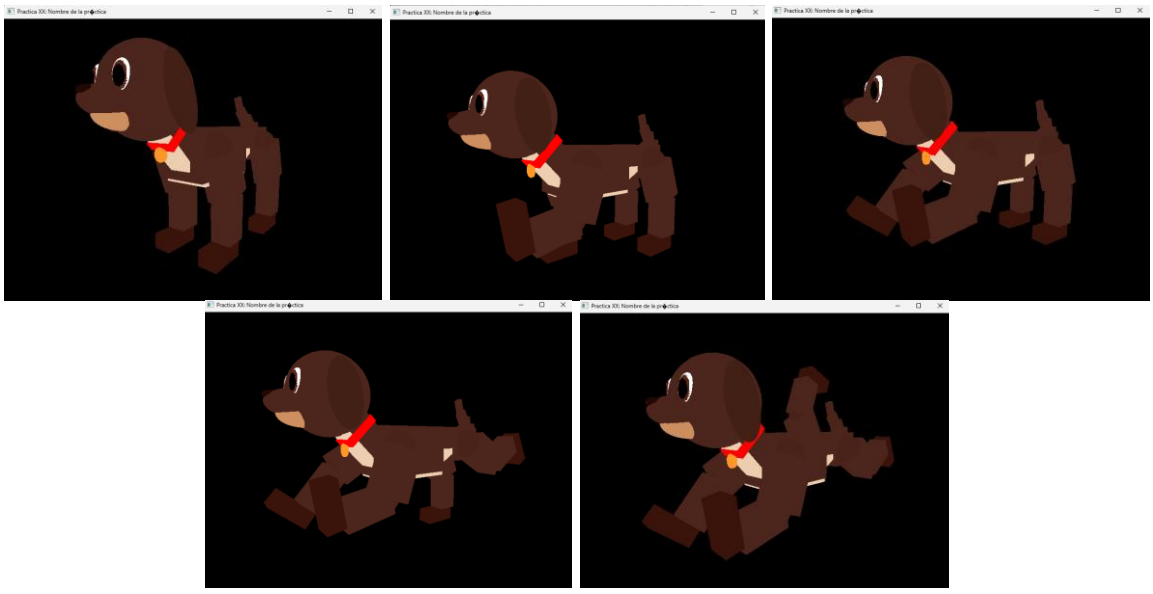
Evidencias:

1.2.1. *Ejercicio 1:*



1.2.2. *Ejercicio 2:*





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
   R: Surgió un error de no poder escribir el archivo en un momento dado de estar realizando el armado, como cerrar el archivo no lo resolvió reinicie la computadora y quedó resuelto.

3. Conclusión:
   1. Los ejercicios del reporte: Complejidad, Explicación.
      Respecto a los ejercicios realizados, creo que fueron bastante adecuados al nivel de lo visto en clase, solo aprendí como limitar la rotación, sin embargo, el segundo si fue bastante largo.

   2. Comentarios generales: Faltó explicar a detalle, ir más lento en alguna explicación, otros comentarios y sugerencias para mejorar desarrollo de la práctica.
      Respecto a la explicación no tengo mucho que decir, fue bastante buena y reutilizamos gran parte del código, sin embargo, me costó algo de trabajo entender la matriz modelaux, ahora me doy cuenta de que es algo relativamente sencillo.

   3. Conclusión
      Personalmente creo que se cumplieron los objetivos de la práctica. Con el ejercicio aprendí la necesidad de tener un diagrama jerárquico de los modelos, así como a implementar uno para el desarrollo de una estructura compleja. Con este ejercicio aprendí a trabajar con jerarquías en la creación de modelos, haciendo uso de matrices ‘pivote’ que guarden las transformaciones realizadas y con ello entendí cómo funciona la cadena de transformaciones. Además, aprendí a delimitar la rotación, hacer que estas cambien de sentido y hacer animaciones más fluidas como la de la cola, utilizando la misma rotación.