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: Uriarte Ortiz Enrique Yahir

Nº de Cuenta: 318234757

GRUPO DE LABORATORIO: 02

GRUPO DE TEORÍA: 04

SEMESTRE 2025-1

FECHA DE ENTREGA LÍMITE: Sábado 07 de Septiembre del 2024

CALIFICACIÓN: _____

REPORTE DE PRÁCTICA:

1. *Ejecución de los ejercicios.*
   1) ***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 iniciar con esta actividad genere 4 jerarquías más en la función main del archivo main de la practica, cada una de acuerdo con el árbol jerárquico que se dio en el material para la práctica: la original, la jerarquía para la base y la primera llanta, las jerarquías para las otras tres llantas y la jerarquía para el brazo generado en el ejercicio de la práctica.
Lo primero en el código es definir la base, pues será el punto de inicio para las jerarquías, y ya que esta ya se había definido previamente en el código del ejercicio de la practica únicamente se copia, lo que si se modifica es guardar la información de translación de las jerarquías que van hacia la base y hacia los brazos.

```
glm::mat4 model(1.0);       //Origen de las jererquias.
glm::mat4 modelaux(1.0);    //Jerarquia con la base de la grua y primera llanta.
glm::mat4 modelaux2(1.0);   //Jerarquia con la segunda llanta.
glm::mat4 modelaux3(1.0);   //Jerarquia con la tercer llanta.
glm::mat4 modelaux4(1.0);   //Jerarquia con la cuarta llanta.
glm::mat4 modelaux5(1.0);   //Jerarquia con los brazos.

glm::vec3 color = glm::vec3(0.0f,0.0f,0.0f);
```

Fig. 1. *Definimos las jerarquías que usaremos.*

```
// ACTIVIDAD 1. Terminar la Grúa.   ---     ---     ---     ---     ---     ---     ---     ---

model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, 6.0f, -4.0f));
modelaux = model;  //Jerarquia con la base de la grua
modelaux5 = model; //Jerarquia con los brazos.

//Cuerpo (prisma rectangular).
model = glm::translate(model, glm::vec3(0.4f, -0.2f, 0.0f));
model = glm::scale(model, glm::vec3(1.3f, 0.8f, 1.0f));
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()));
color = glm::vec3(1.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh();
```

Fig. 2. *Definimos el cuerpo de la grúa.*

Después definí la primera articulación hacia la base, definí que su rotación fuera con respecto al eje de las Y, una vez ubicada más abajo que la base del cuerpo y definida de color azul, se guarda la información de translación para las jerarquías de las 4 llantas. Ya después definí la base, esta es una pirámide triangular, de base definí que fuera un poco más grande que el cuerpo, mientras que la altura intenté que fuera similar con la imagen de referencia, para su ubicación la coloque mas abajo en Y negativa y para ajustarla la moví un poquito hacia Z negativo.
Después llamo de nuevo a la jerarquía de la primera rama para ubicar la primera articulación de movimiento para la primera llanta, la ubico al extremo de la base y le asigno la rotación con respecto al eje de las Z, después guardo la

información de translación para la llanta, defino su escala y que sea de color azul. Por último, llamo de nuevo a la jerarquía y mando a imprimir un cilindro ajustado un poco más adelante en el eje Z, para que se pueda ver la articulación, pero además lo defino rotado 90° en el eje X para simular la llanta, y ya por último la defino de color negro

```
model = modelaux;

//Articulación 1
model = glm::translate(model, glm::vec3(0.4f, -0.47f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()), glm::vec3(0.0f, 1.0f, 0.0f));
modelaux = model;
model = glm::scale(model, glm::vec3(0.2f, 0.2f, 0.2f));
color = glm::vec3(0.0f, 0.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
sp.render();

model = glm::translate(model, glm::vec3(0.0f, -0.45f, 0.0f));
modelaux = model;   //Jerarquia con la base de la grua y primera llanta.
modelaux2 = model;  //Jerarquia con la segunda llanta.
modelaux3 = model;  //Jerarquia con la tercer llanta.
modelaux4 = model;  //Jerarquia con la cuarta llanta.

//Base (pirámide cuadrangular).
model = glm::translate(model, glm::vec3(0.0f, -1.0f, -0.1f));
model = glm::scale(model, glm::vec3(6.9f, 2.6f, 4.5f));
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()));
color = glm::vec3(1.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[1]->RenderMeshGeometry();
```

Fig. 3. *Definimos la base, donde contaremos las llantas.*

```
model = modelaux;

//Articulación Llanta 01.
model = glm::translate(model, glm::vec3(2.0f, -2.3f, 2.1f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()), glm::vec3(0.0f, 0.0f, 1.0f));
modelaux = model;
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
color = glm::vec3(0.0f, 0.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
sp.render();

modelaux = model;

//Llanta 01.
model = glm::translate(model, glm::vec3(0.0f, 0.0f, 1.2f));
model = glm::rotate(model, glm::radians(90.0f), glm::vec3(1.0f, 0.0f, 0.0f));
modelaux = model;
model = glm::scale(model, glm::vec3(2.5f, 1.5f, 2.5f));
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()));
color = glm::vec3(0.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[2]->RenderMeshGeometry();
```

Fig. 4. *Definimos la primera llanta con su articulación.*

Para las demás llantas use el mismo pedazo de código que el de la primera llanta, lo que cambia entre cada llanta son los valores de los ejes entre positivo y negativo, por ejemplo, para la segunda llanta está se encuentra en -X, -Y y +Z, para la tercera llanta está se encuentra en -X, -Y y -Z, y para la cuarta llanta está se encuentra en +X, -Y y -Z, en cuanto a los demás valores para la articulación y llanta correspondiente no hay ningún otro cambio.

Por último, para el brazo, mando a llamar a la jerarquía “modelaux5” donde guarde en un inicio las translaciones del cuerpo y coloco el código generado durante el ejercicio de la práctica 4, ya que este contiene esa parte.

```
model = modelaux2;

//Articulación Llanta 02.
model = glm::translate(model, glm::vec3(-2.0f, -2.3f, 2.1f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion3()), glm::vec3(0.0f, 0.0f, 1.0f));
modelaux2 = model;
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
color = glm::vec3(0.0f, 0.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
sp.render();

modelaux2 = model;

//Llanta 02.
model = glm::translate(model, glm::vec3(0.0f, 0.0f, 1.2f));
model = glm::rotate(model, glm::radians(90.0f), glm::vec3(1.0f, 0.0f, 0.0f));
modelaux2 = model;
model = glm::scale(model, glm::vec3(2.5f, 1.5f, 2.5f));
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()));
color = glm::vec3(0.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[2]->RenderMeshGeometry();
```

Fig. 5. *Definimos la segunda llanta con su articulación.*

```
model = modelaux3;

//Articulación Llanta 03.
model = glm::translate(model, glm::vec3(-2.0f, -2.3f, -2.1f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion4()), glm::vec3(0.0f, 0.0f, 1.0f));
modelaux3 = model;
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
color = glm::vec3(0.0f, 0.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
sp.render();

modelaux3 = model;

//Llanta 03.
model = glm::translate(model, glm::vec3(0.0f, 0.0f, -1.2f));
model = glm::rotate(model, glm::radians(90.0f), glm::vec3(1.0f, 0.0f, 0.0f));
modelaux3 = model;
model = glm::scale(model, glm::vec3(2.5f, 1.5f, 2.5f));
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()));
color = glm::vec3(0.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[2]->RenderMeshGeometry();
```

Fig. 6. *Definimos la tercera llanta con su articulación.*

```
model = modelaux4;

//Articulación Llanta 04
model = glm::translate(model, glm::vec3(2.0f, -2.3f, -2.1f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion5()), glm::vec3(0.0f, 0.0f, 1.0f));
modelaux4 = model;
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
color = glm::vec3(0.0f, 0.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
sp.render();

modelaux4 = model;

//Llanta 04.
model = glm::translate(model, glm::vec3(0.0f, 0.0f, -1.2f));
model = glm::rotate(model, glm::radians(90.0f), glm::vec3(1.0f, 0.0f, 0.0f));
modelaux4 = model;
model = glm::scale(model, glm::vec3(2.5f, 1.5f, 2.5f));
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()));
color = glm::vec3(0.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[2]->RenderMeshGeometry();

//Instrucciones de articulaciones y brazos desarrollado en el ejercicio.
```

Fig. 7. *Definimos la cuarta llanta con su articulación.*

```
model = modelaux5;

//Articulación 1.
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion6()), glm::vec3(0.0f, 0.0f, 1.0f));
modelaux5 = model;
model = glm::scale(model, glm::vec3(0.2f, 0.2f, 0.2f));
color = glm::vec3(0.0f, 0.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
sp.render();

//Brazo N.1.
model = glm::translate(model, glm::vec3(-2.0f, 2.0f, 0.0f));
model = glm::rotate(model, glm::radians(135.0f), glm::vec3(0.0f, 0.0f, 1.0f));
modelaux5 = model;
model = glm::scale(model, glm::vec3(5.0f, 1.0f, 1.0f));
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()));
color = glm::vec3(1.0f, 0.0f, 0.5f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh();
model = modelaux5;
```

Fig. 8. *Mismo código el ejercicio de practica para el brazo.*

Como el número de articulaciones en esta ocasión fueron 9 en total, entre los que ya tenia generados en el ejercicio de la práctica y los nuevos para la parte de la base y las llantas, en el archivo “Window.h” defino las otras tres que me faltan ya que el código ya tenía definidas las primeras 9

```
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; }
```

Fig. 9. *Definimos las articulaciones en el archivo Window.h*

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

Fig. 10. *Definimos las articulaciones en el archivo Window.h*

Y luego las defino en el archivo “Window.cpp”

```
width = windowWidth;
height = windowHeight;
rotax = 0.0f;
rotay = 0.0f;
rotaz = 0.0f;
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;
```

Fig. 11. *Definimos las articulaciones en el archivo Window.cpp*

Por último, en el mismo archivo asigno las teclas, la distribución de estas fue así para tener un poco mas de orden desde mi perspectiva para poder mover las articulaciones de cada parte de grúa de acuerdo si están en la parte de arriba o debajo del cuerpo.

```
//Teclas rotar Jerarquia 1
if (key == GLFW_KEY_H){theWindow->articulacion1 += 10.0;}
if (key == GLFW_KEY_V){theWindow->articulacion2 += 10.0;}
if (key == GLFW_KEY_B){theWindow->articulacion3 += 10.0;}
if (key == GLFW_KEY_N){theWindow->articulacion4 += 10.0;}
if (key == GLFW_KEY_M){theWindow->articulacion5 += 10.0;}

//Teclas rotar Jerarquia 2
if (key == GLFW_KEY_Y){theWindow->articulacion6 += 10.0;}
if (key == GLFW_KEY_U){theWindow->articulacion7 += 10.0;}
if (key == GLFW_KEY_I){theWindow->articulacion8 += 10.0;}
if (key == GLFW_KEY_O){theWindow->articulacion9 += 10.0;}
```

Fig. 12. *Definimos las articulaciones en el archivo Window.cpp*

Ya al momento de ejecutar al código la impresión de las figuras fue la siguiente:

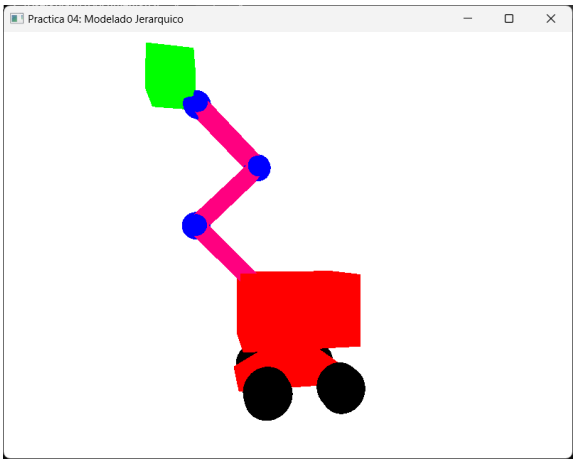

Fig. 13. *Ejecución de la actividad.*

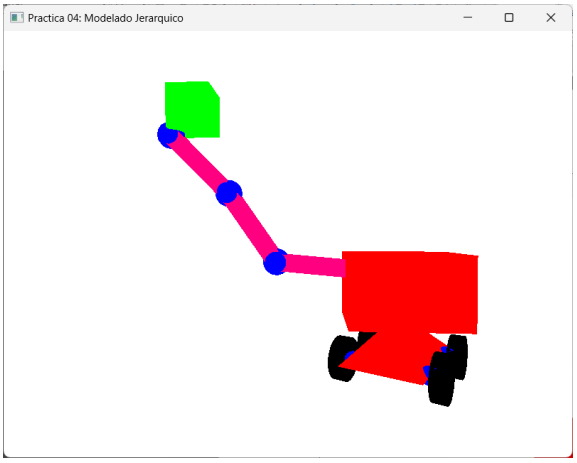

Fig. 14. *Ejecución de la actividad.*

Como evidencia para la práctica, realice un video donde se muestra la ejecución, el archivo se llama ***P04-318234757_Ejercicio1.mp4***, adjunto el enlace de la carpeta compartida:
https://drive.google.com/drive/folders/1cDzVU-oxT-JNwWP7DCOPhtpTUhCzl6-H?usp=sharing

2) ***Crear un animal robot 3d***

- *Instanciando 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.*

Para esta actividad el animal que escogí fue la araña robot que se proponía en el material de la práctica.

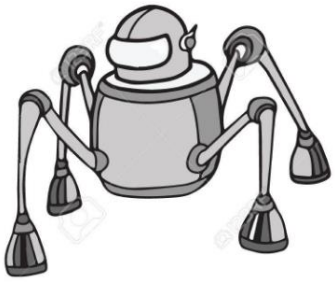

Fig. 15. *Animal robot escogido, araña robot.*

Tomé en cuenta en esta ocasión desde el principio el último paso que hice de la actividad anterior, definir el número de articulaciones. Analizando la imagen de prueba determine que para una pata de la araña necesitaba 3 articulaciones (las 2 partes de la pata y el pie), mas las 2 articulaciones de las orejas, dando un total de 14 puntos de articulación.
En el archivo “Window.h” defino las nuevas articulaciones.

```
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; }
GLfloat getarticulacion10() { return articulacion10; }
GLfloat getarticulacion11() { return articulacion11; }
GLfloat getarticulacion12() { return articulacion12; }
GLfloat getarticulacion13() { return articulacion13; }
GLfloat getarticulacion14() { return articulacion14; }
```

Fig. 16. *Definimos las articulaciones en el archivo Window.h*

```
private:
    GLFWwindow *mainWindow;
    GLint width, height;
    GLfloat rotax,rotay,rotaz, articulacion1, articulacion2, articulacion3, articulacion4,
            articulacion5, articulacion6, articulacion7, articulacion8, articulacion9,
            articulacion10, articulacion11, articulacion12, articulacion13, articulacion14;
```

Fig. 17. *Definimos las articulaciones en el archivo Window.h*

Luego las defino en el archivo “Window.cpp”

```
width = windowWidth;
height = windowHeight;
rotax = 0.0f;
rotay = 0.0f;
rotaz = 0.0f;
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;
articulacion10 = 0.0f;
articulacion11 = 0.0f;
articulacion12 = 0.0f;
articulacion13 = 0.0f;
articulacion14 = 0.0f;
```

Fig. 18. *Definimos las articulaciones en el archivo Window.cpp*

Y en el mismo archivo asigno las teclas, la distribución de estas fue así para tener un poco más de orden desde mi perspectiva para poder mover las articulaciones de cada parte de pata de la araña.

```
//Teclas para rotar pierna 1 de la araña.
if (key == GLFW_KEY_Z) { theWindow->articulacion1 += 10.0; }
if (key == GLFW_KEY_X) { theWindow->articulacion2 += 10.0; }
if (key == GLFW_KEY_C) { theWindow->articulacion3 += 10.0; }

//Teclas para rotar pierna 2 de la araña.
if (key == GLFW_KEY_V) { theWindow->articulacion4 += 10.0; }
if (key == GLFW_KEY_B) { theWindow->articulacion5 += 10.0; }
if (key == GLFW_KEY_N) { theWindow->articulacion6 += 10.0; }

//Teclas para rotar pierna 3 de la araña.
if (key == GLFW_KEY_F) { theWindow->articulacion7 += 10.0; }
if (key == GLFW_KEY_G) { theWindow->articulacion8 += 10.0; }
if (key == GLFW_KEY_H) { theWindow->articulacion9 += 10.0; }

//Teclas para rotar pierna 2 de la araña.
if (key == GLFW_KEY_J) { theWindow->articulacion10 += 10.0; }
if (key == GLFW_KEY_K) { theWindow->articulacion11 += 10.0; }
if (key == GLFW_KEY_L) { theWindow->articulacion12 += 10.0; }

//Teclas para rotar oreja 1 de la araña
if (key == GLFW_KEY_M) { theWindow->articulacion13 += 10.0; }
//Teclas para rotar oreja 2 de la araña
if (key == GLFW_KEY_P) { theWindow->articulacion14 += 10.0; }
```

Fig. 19. *Definimos las articulaciones en el archivo Window.cpp*

Para tener mayor control de las dimensiones y el número de jerarquías para cada pata, realice un árbol jerárquico sencillo de este caso como guía para poder realizar la actividad, y también realice sobre el dibujo de araña robot trazos para definir las dimensiones aproximadas con las que definiría los elementos del modelo en 3D.

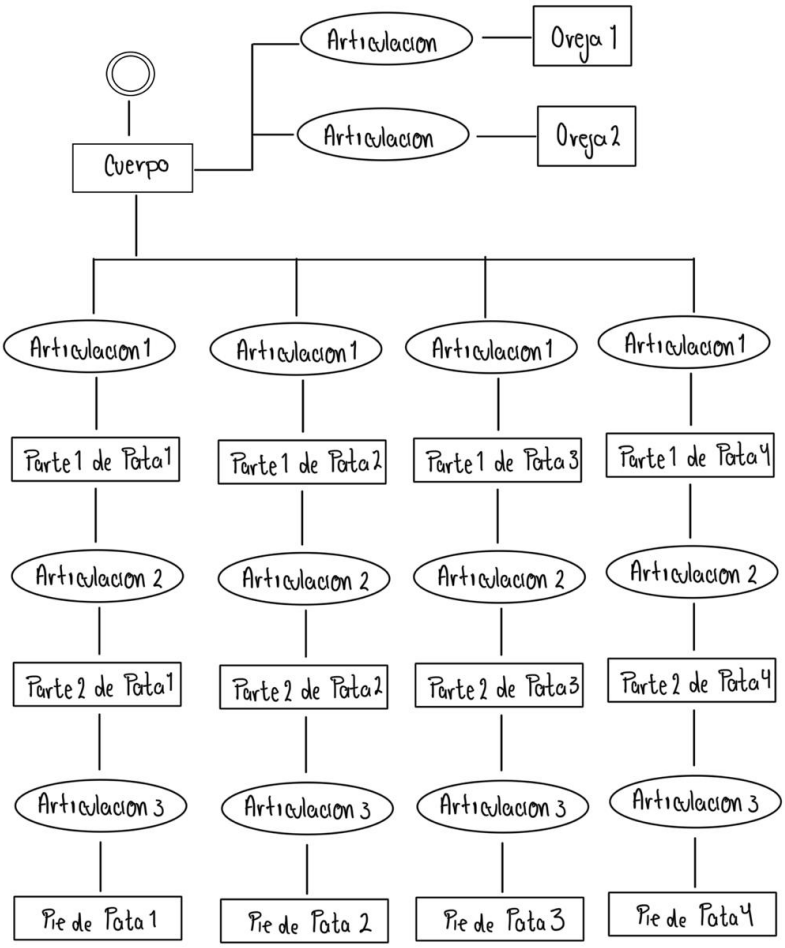

Fig. 21. *Árbol jerárquico sencillo de la Araña robot.*

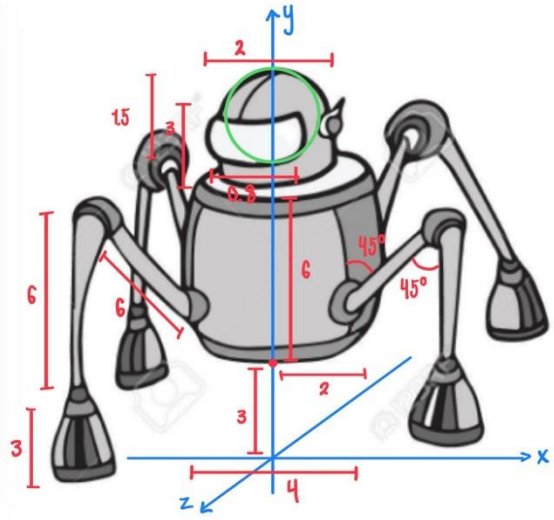

Fig. 21. *Araña robot con trazos.*

Ya en el archivo main, en la función main de este, con ayuda del árbol jerárquico identifique 6 jerarquías distintas, me posicione desde un inicio en la altura que definí en el trazo sobre el robot y guardo la información de translación para las jerarquías.
Luego defino el cuerpo de la araña como un cilindro, como sobre esta base voy a definir las 4 patas, mando a guardar la información de translación para las jerarquías de cada una, defino su escala de acuerdo con los trazos y el color, un tono gris que se asemeja a la imagen.

```
// ACTIVIDAD 2. Crear un animal robot 3d    ---     ---     ---     ---     ---     ---

model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, 3.0f, -20.0f));
modelaux = model;   //Jerarquia con la primera pierna.
modelaux2 = model;  //Jerarquia con la segunda pierna.
modelaux3 = model;  //Jerarquia con la tercera pierna.
modelaux4 = model;  //Jerarquia con la cuarta pierna.
modelaux5 = model;  //Jerarquia con la primer oreja.
modelaux6 = model;  //Jerarquia con la segunda oreja.

//Cuerpo.
model = glm::translate(model, glm::vec3(0.0f, 3.0f, 0.0f));
modelaux = model;
modelaux2 = model;  //Jerarquia con la segunda pierna.
modelaux3 = model;  //Jerarquia con la tercera pierna.
modelaux4 = model;  //Jerarquia con la cuarta pierna.
model = glm::scale(model, glm::vec3(4.0f, 6.0f, 4.0f));
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()));
color = glm::vec3(0.55f, 0.55f, 0.55f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[2]->RenderMeshGeometry();
```

Fig. 22. *Definimos el cuerpo de la araña.*

```
model = modelaux;

//Cabeza.
model = glm::translate(model, glm::vec3(0.0f, 4.5f, 0.0f));
modelaux5 = model;  //Jerarquia con la primer oreja.
modelaux6 = model;  //Jerarquia con la segunda oreja.
model = glm::scale(model, glm::vec3(2.0f, 3.0f, 2.0f));
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()));
color = glm::vec3(0.6f, 0.6f, 0.6f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[2]->RenderMeshGeometry();

//Visor.
model = glm::translate(model, glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::scale(model, glm::vec3(0.8f, 0.3f, 0.3f));
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()));
color = glm::vec3(0.9f, 0.9f, 0.9f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
sp.render();

//Tapa superior.
model = glm::translate(model, glm::vec3(0.0f, 1.6f, -3.3f));
model = glm::scale(model, glm::vec3(1.25f, 1.5f, 3.4f));
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()));
color = glm::vec3(0.6f, 0.6f, 0.6f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
sp.render();
```

Fig. 23. *Definimos la parte de la cabeza de la araña.*

Después definí un poco más los elementos de la araña, como son la cabeza, el visor que tiene, y para darle ese toque de cabeza esférica, definí una esfera ajustada al cilindro que utilizamos como cabeza, como ninguno de estos elementos tiene articulación no es necesario guardar las coordenadas de translación en jerarquías.
Una vez definidas toda la estructura principal del animal, su tronco, definí la articulación de la primera pata, en este caso la ubique en una esquina del cilindro y le asigne la rotación en los ejes X y Y para que tuviera una rotación mas acorde con el área de movimiento, mando a guardar la información de translación, defino la escala y el color de un gris más oscuro. Para el primer parte de la pata, la roto -45° en -X y Z y la acomodo en la distancia adecuada que toque la articulación, la escalo para que tenga el largo que había definido y su grosor, así como el color que tendrá, el mismo que el de la cabeza.

```
model = modelaux;

//Articulación 1 pierna 1.
model = glm::translate(model, glm::vec3(2.8f, -1.0f, 2.8f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()), glm::vec3(1.0f, 0.0f, 1.0f));
modelaux = model;
model = glm::scale(model, glm::vec3(0.7f, 0.7f, 0.7f));
color = glm::vec3(0.3f, 0.3f, 0.3f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
sp.render();

model = modelaux;

//Pierna 1, primer parte.
model = glm::translate(model, glm::vec3(1.8f,2.5f,1.8f));
model = glm::rotate(model, glm::radians(-45.0f), glm::vec3(-1.0f, 0.0f, 1.0f));
modelaux = model;
model = glm::scale(model, glm::vec3(0.5f, 6.0f, 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()));
color = glm::vec3(0.6f, 0.6f, 0.6f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[2]->RenderMeshGeometry();
```

Fig. 24. *Definimos la primer parte de la pata 1 de la araña.*

```
model = modelaux;

//Articulación 2 pierna 1.
model = glm::translate(model, glm::vec3(0.0f, 2.5f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()), glm::vec3(1.0f, 1.0f, 1.0f));
modelaux = model;
model = glm::scale(model, glm::vec3(0.7f, 0.7f, 0.7f));
color = glm::vec3(0.3f, 0.3f, 0.3f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
sp.render();

model = modelaux;

//Pierna 1, segunda parte.
model = glm::translate(model, glm::vec3(2.1f, -1.8f, 2.1f));
model = glm::rotate(model, glm::radians(60.0f), glm::vec3(-1.0f, 0.0f, 1.0f));
modelaux = model;
model = glm::scale(model, glm::vec3(0.5f, 6.0f, 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()));
color = glm::vec3(0.6f, 0.6f, 0.6f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[2]->RenderMeshGeometry();
```

Fig. 25. *Definimos la segunda parte de la pata 1 de la araña.*

Para la segunda parte de la primera pata ubico la articulación en el extremo contrario del último cilindro creado, pero antes ubico mando a llamar a la jerarquía para que ubique esta nueva parte de la pata con respecto a la

translación guardada de la parte anterior de la articulación y rotación, para su rotación la defino en los 3 ejes, esto le da un poco mas de realismo en el movimiento a mi parecer, ajusto la escala y el color como el de la articulación anterior. Después defino la segunda parte de la pata, esta con un ángulo de inclinación de 60° hacia -X y Z, guardo la ubicación en la jerarquía, y ajusto el tamaño y color.
Para la tercera parte, que es ya la que va con el pie, defino de nueva la articulación ajustada con el ultimo cilindro que genere, asigno la rotación hacia Y y -Z, guardo la translación en la jerarquía, ajusto escala de la articulación y color. Para la parte del pie decidí utilizar un cono metido un poco hacia la articulación, la roto -15° para que se vea mejor el pie, simulando que esta tocando el piso o una superficie, ajusto la escala y el color.

```
model = modelaux;

//Articulación 3 pierna 1.
model = glm::translate(model, glm::vec3(0.0f, -3.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion3()), glm::vec3(0.0f, 1.0f, -1.0f));
modelaux = model;
model = glm::scale(model, glm::vec3(0.7f, 0.7f, 0.7f));
color = glm::vec3(0.3f, 0.3f, 0.3f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
sp.render();

model = modelaux;

//Pie 1.
model = glm::translate(model, glm::vec3(0.0f, -1.0f, 0.0f));
model = glm::rotate(model, glm::radians(-15.0f), glm::vec3(-1.0f, 0.0f, 1.0f));
modelaux = model;
model = glm::scale(model, glm::vec3(0.8f, 3.0f, 0.8f));
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()));
color = glm::vec3(0.6f, 0.6f, 0.6f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[3]->RenderMeshGeometry();
```

Fig. 26. *Definimos la parte del pie de la pata 1 de la araña.*

```
model = modelaux2;

//Articulación 1 pierna 2.
model = glm::translate(model, glm::vec3(-2.8f, -1.0f, 2.8f));
model = glm::rotate(model, glm::radians(-90.0f), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion4()), glm::vec3(1.0f, 0.0f, 1.0f));
modelaux2 = model;
model = glm::scale(model, glm::vec3(0.7f, 0.7f, 0.7f));
color = glm::vec3(0.3f, 0.3f, 0.3f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
sp.render();

model = modelaux2;

//Pierna 2, primer parte.
model = glm::translate(model, glm::vec3(1.8f, 2.5f, 1.8f));
model = glm::rotate(model, glm::radians(-45.0f), glm::vec3(-1.0f, 0.0f, 1.0f));
modelaux2 = model;
model = glm::scale(model, glm::vec3(0.5f, 6.0f, 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()));
color = glm::vec3(0.6f, 0.6f, 0.6f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[2]->RenderMeshGeometry();
```

Fig. 27. *Definimos la primer parte de la pata 1 de la araña.*

Con la primera pata terminada, para generar la segunda decidí girar en -90° la articulación sobre el eje Y, ubicándola en el extremo contrario de lo que sería el frente de la araña, además cambie los signos en las coordenadas del extremo, esto para ajustar mejor la posición y pareciera espejo de la primera pata, ahora

estando en -X, -Y y Z, todo lo demás del código es igual a lo de la primera pata, salvo por las articulaciones definidas por cada tecla para que roten de manera independiente y los guardados de traslación en su respectiva jerarquía. Esto mismo lo realice con las otras 2 patas, para el caso de la tercera pata ahora estando en X, -Y y -Z, con una rotación en el eje Y de 90°, y para el caso de la cuarta pata ahora estando en -X, -Y y -Z , con una rotación en el eje Y de 180°.

```
model = modelaux3;

//Articulación 1 pierna 3.
model = glm::translate(model, glm::vec3(2.8f, -1.0f, -2.8f));
model = glm::rotate(model, glm::radians(90.0f), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion7()), glm::vec3(1.0f, 0.0f, 1.0f));
modelaux3 = model;
model = glm::scale(model, glm::vec3(0.7f, 0.7f, 0.7f));
color = glm::vec3(0.3f, 0.3f, 0.3f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
sp.render();

model = modelaux3;

//Pierna 3, primer parte.
model = glm::translate(model, glm::vec3(1.8f, 2.5f, 1.8f));
model = glm::rotate(model, glm::radians(-45.0f), glm::vec3(-1.0f, 0.0f, 1.0f));
modelaux3 = model;
model = glm::scale(model, glm::vec3(0.5f, 6.0f, 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()));
color = glm::vec3(0.6f, 0.6f, 0.6f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[2]->RenderMeshGeometry();
```

Fig. 28. *Definimos la primer parte de la pata 3 de la araña.*

```
model = modelaux4;

//Articulación 1 pierna 4.
model = glm::translate(model, glm::vec3(-2.8f, -1.0f, -2.8f));
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion10()), glm::vec3(1.0f, 0.0f, 1.0f));
modelaux4 = model;
model = glm::scale(model, glm::vec3(0.7f, 0.7f, 0.7f));
color = glm::vec3(0.3f, 0.3f, 0.3f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
sp.render();

model = modelaux4;

//Pierna 4, primer parte.
model = glm::translate(model, glm::vec3(1.8f, 2.5f, 1.8f));
model = glm::rotate(model, glm::radians(-45.0f), glm::vec3(-1.0f, 0.0f, 1.0f));
modelaux4 = model;
model = glm::scale(model, glm::vec3(0.5f, 6.0f, 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()));
color = glm::vec3(0.6f, 0.6f, 0.6f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[2]->RenderMeshGeometry();
```

Fig. 29. *Definimos la primer parte de la pata 4 de la araña.*

Para las orejas realizo la misma parte de código que las patas para la parte del pie, ya que utilizara en estos conos también, primero ajusto la translación de las articulaciones para que se encuentren en los extremos de la cabeza, para estos casos se utiliza las jerarquías de “modelaux5” y “modelaux6” respectivamente con la oreja 1 y la oreja 2; asigno la tecla de rotación, ajusto la escala y el color. Para la parte de la oreja la ajusto en ubicación para que se encuentre sobre la articulación, de modo que parezca un helado de cabeza, por lo que indico una rotación del cono de -45°, dando ese aspecto de inclinación.

Para la oreja contraria únicamente modifico la ubicación, basta con cambiar el signo del eje de las X de positivo a negativo y dejo las mismas instrucciones de rotación, claro asignado su respectiva tecla para la rotación y su jerarquía.

```
model = modelaux5;

//Articulacion Oreja 1
model = glm::translate(model, glm::vec3(2.2f, 0.5f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion13()), glm::vec3(1.0f, 0.0f, 0.0f));
modelaux5 = model;
model = glm::scale(model, glm::vec3(0.4f, 0.4f, 0.4f));
color = glm::vec3(0.3f, 0.3f, 0.3f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
sp.render();

model = modelaux5;

//Oreja 1
model = glm::translate(model, glm::vec3(0.0f, 0.8f, -0.8f));
model = glm::rotate(model, glm::radians(-45.0f), glm::vec3(1.0f, 0.0f, 0.0f));
modelaux5 = model;
model = glm::scale(model, glm::vec3(0.2f,2.0f, 0.2f));
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()));
color = glm::vec3(0.8f, 0.8f, 0.8f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[3]->RenderMeshGeometry();
```

Fig. 30. *Definimos la primer oreja de la araña.*

```
model = modelaux6;

//Articulacion Oreja 2
model = glm::translate(model, glm::vec3(-2.2f, 0.5f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion14()), glm::vec3(1.0f, 0.0f, 0.0f));
modelaux5 = model;
model = glm::scale(model, glm::vec3(0.4f, 0.4f, 0.4f));
color = glm::vec3(0.3f, 0.3f, 0.3f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
sp.render();

model = modelaux5;

//Oreja 2
model = glm::translate(model, glm::vec3(0.0f, 0.8f, -0.8f));
model = glm::rotate(model, glm::radians(-45.0f), glm::vec3(1.0f, 0.0f, 0.0f));
modelaux5 = model;
model = glm::scale(model, glm::vec3(0.2f, 2.0f, 0.2f));
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()));
color = glm::vec3(0.8f, 0.8f, 0.8f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[3]->RenderMeshGeometry();
```

Fig. 31. *Definimos la segunda oreja de la araña.*

Ya al momento de ejecutar al código la impresión de las figuras fue la siguiente:

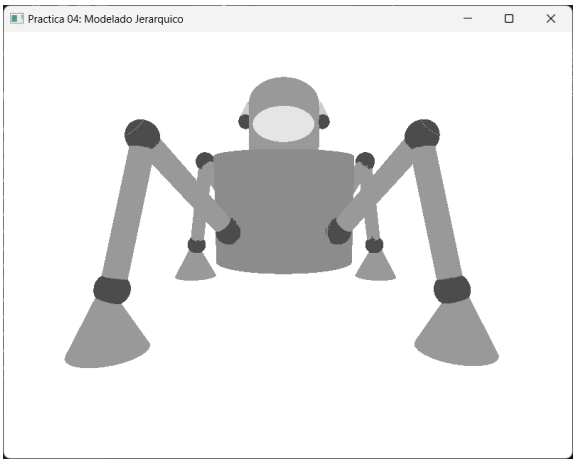

Fig. 32. *Ejecución de la actividad.*

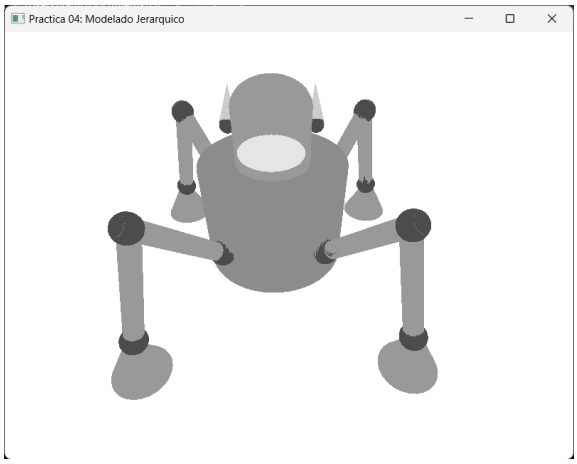

Fig. 33. *Ejecución de la actividad.*

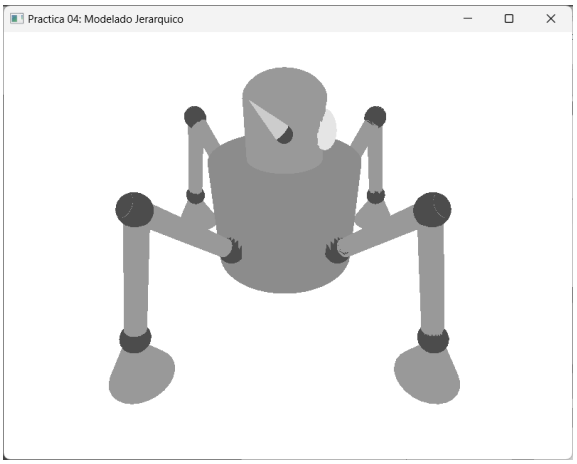

Fig. 34. *Ejecución de la actividad.*

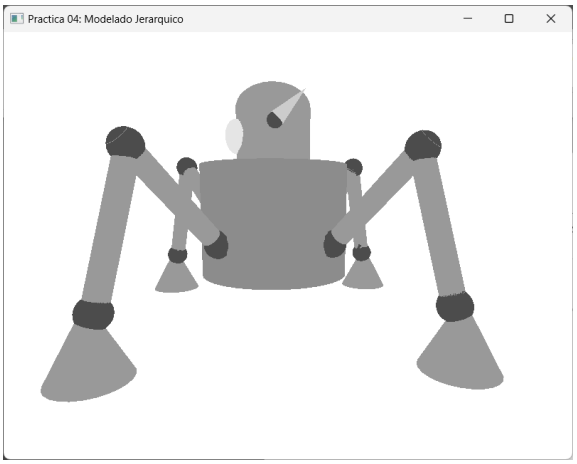

Fig. 35. *Ejecución de la actividad.*

Como evidencia para la práctica, realice un video donde se muestra la ejecución, el archivo se llama ***P04-318234757_Ejercicio2.mp4***, adjunto el enlace de la carpeta compartida:
https://drive.google.com/drive/folders/1cDzVU-oxT-JNwWP7DCOPhtpTUhCzl6-H?usp=sharing

2. *Problemas presentados.*
   En la primera actividad no presente problemas graves, la principal complicación que tuve fue recordar como definir las jerarquías ya que a veces cuando quería definir la primera articulación de la base está o no apareció o el que no aparecía era la misma base, pero después de analizar el código con jerarquía proporcionado logue entender dónde colocar las indicaciones de las jerarquías, con lo que el problema se resolvió.
   Para la segunda actividad tuve problemas después de crear la primera pata por completo, para saber cómo realizar las otras 3, incluso con la solución que planté de rotar las articulaciones, las patas aun estaban en un ángulo incorrecto, pero después de analizar la ubicación del punto donde coloque la articulación principal de la primera pata, entendí que la solución era solo cambiar algunos signos de la ubicación de acuerdo con donde quisiera ponerla. Por lo que al final ambos ejercicios se pudieron realizar exitosamente.

3. *Conclusión:*
   Tras realizar los ejercicios de la practica me parece que la complejidad era normal en ambos, pero el segundo era más laborioso de realizar, ya que entendiendo el funcionamiento de las jerarquías y como se debían definir, estos ejercicios se hacían sencillos de entender, quizás tenga algo que ver el animal que elegí para el ejercicio 2, creo que fue fácil. Aun así, me pareció interesante la implementación en esta práctica para poder definir articulaciones y sus sentidos de rotación, considero que la actividad fue buena para poner en práctica las funciones de rotar y desplazar que se aprendieron en prácticas anteriores, además de la nueva definición de rotar con teclas físicas. Al final consideró que esta práctica cumplió su objetivo de introducirnos a la rotación de articulaciones a través de las teclas, siendo esto a mi parecer una buena práctica para el desarrollo del proyecto final.

4. *Bibliografía.*
   - *No empleada*