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: Vargas López Miguel Adán

Nº de Cuenta: 421079522

GRUPO DE LABORATORIO: 3

GRUPO DE TEORÍA: 4

SEMESTRE 2025-1

FECHA DE ENTREGA LÍMITE: 07/08/2024

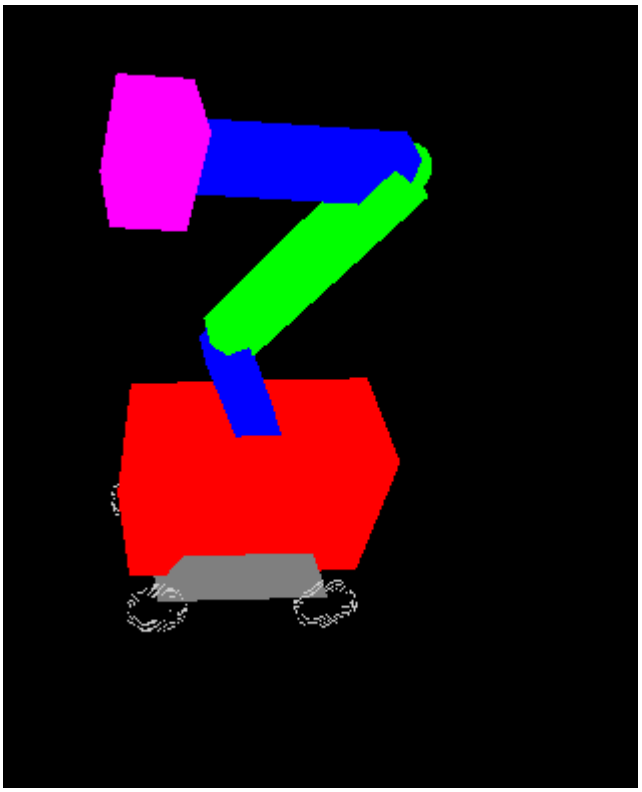
CALIFICACIÓN: _____

Ejercicio 1 Terminar la Grúa

Para terminar con la grúa agregue al centro una pirámide cuadrangular insertado en el prisma para darle forma al cuerpo de la base.

```
//AQUI SE DIBUJA LA CABINA, LA BASE, LAS 4 LLANTAS

//base piramide cuadrangular
model = glm::translate(model, glm::vec3(0.0f, -3.5f, 0.0f));
modelaux = model;
model = glm::scale(model, glm::vec3(4.5f, 5.0f, 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)); //para cambiar el color del objetos
meshList[4]->RenderMesh(); //dibuja cubo y pirámide triangular
model = modelaux;
```



Para las llantas utilice un arreglo de posiciones distintas para localizar las esquinas de la pirámide cuadrangular y así instanciar las llantas por medio de cilindros.

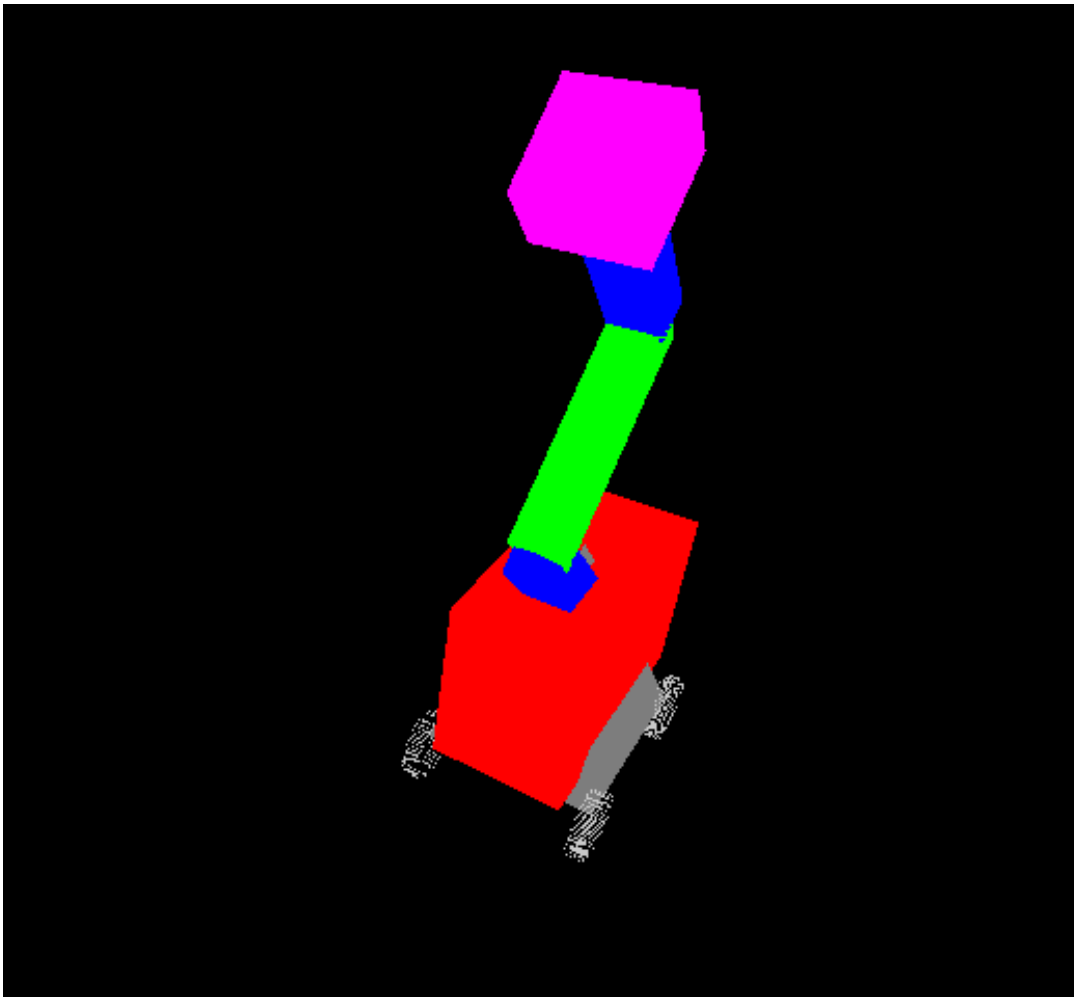
```
glm::vec3 cornerPositions[] = {
    glm::vec3(-2.25f, -2.8f, -2.0f),
    glm::vec3(2.25f, -2.8f, -2.0f),
    glm::vec3(-2.25f, -2.8f,  2.0f),
    glm::vec3(2.25f, -2.8f,  2.0f)
};

for (int i = 0; i < 4; i++) {
    model = modelaux;
    // Posicionar el cilindro en una de las esquinas
    model = glm::translate(model, cornerPositions[i]);
    // Escalar el cilindro (ajusta el valor según el tamaño deseado)
    model = glm::scale(model, glm::vec3(0.8f, 0.8f, 0.4f));
    model = glm::rotate(model, glm::radians(90.0f), glm::vec3(1.0f, 0
    glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(mode
    color = glm::vec3(0.8f, 0.8f, 0.8f);
    glUniform3fv(uniformColor, 1, glm::value_ptr(color));
    meshList[2]->RenderMesh();

}
```

```
glm::vec3 co
    glm::vec
    glm::vec
    glm::vec
    glm::vec
};
```

Teniendo un resultado final como el siguiente:



Los problemas en primera instancia para este problema fueron manejar las llantas ya que no daba con las traslaciones ni rotaciones necesarias para asemejar las llantas con los cilindros pero con la ayuda del for pude instanciarse, sin embargo, al hacer esto no pude establecer las articulaciones para cada llanta y hacerlas girar.

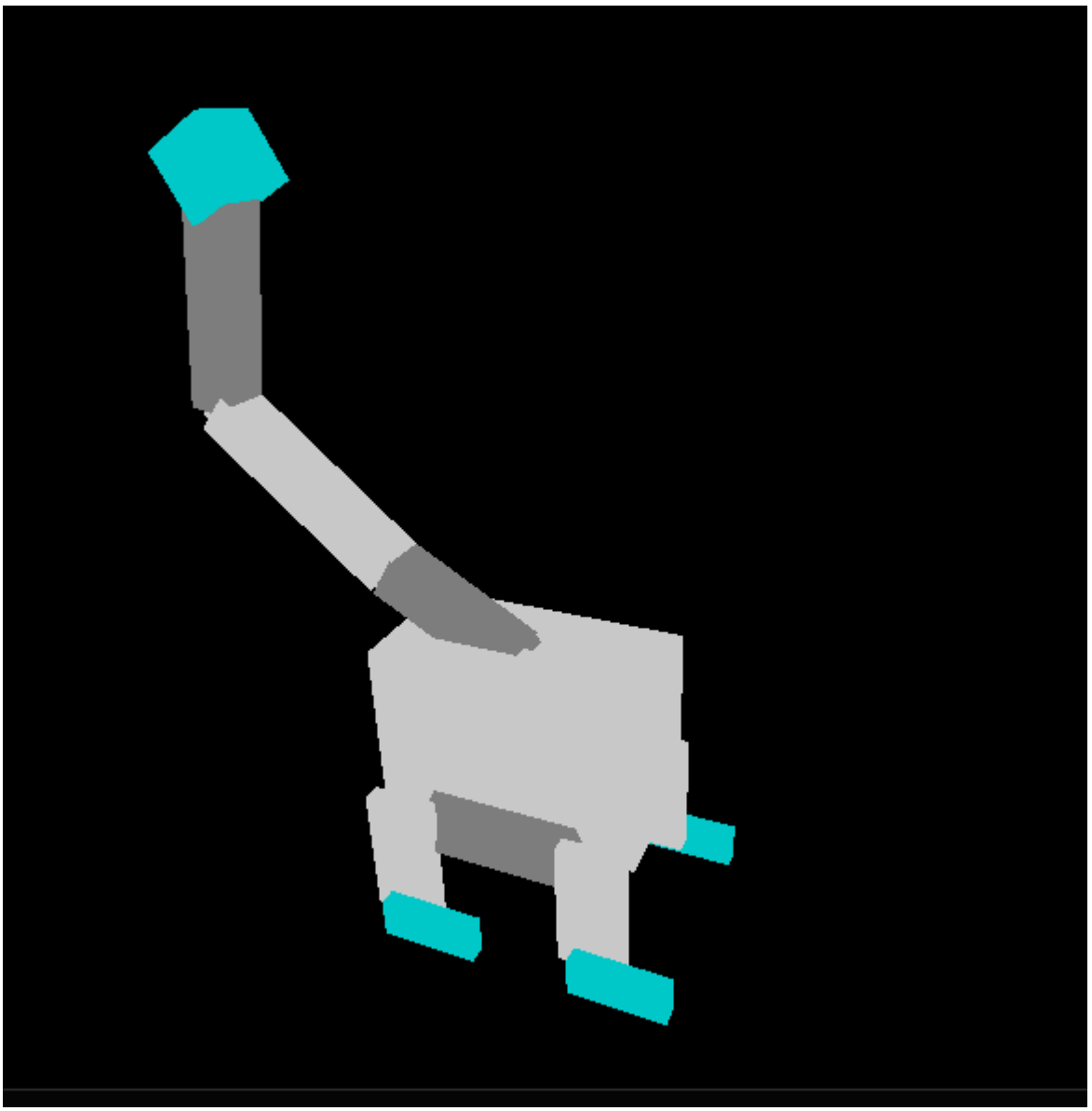
Ejercicio 2 Crear un animal robot 3d

En este programa me base en lo hecho en el ejercicio pasado y si bien, no era la misma figura a realizar, me di cuenta que realmente una buena parte de la estructura ya estaba hecha, tenía un cuerpo, una base, las llantas podría cambiarlas por patas, la grúa podrá modificarla un poco para asemejarse a una cola, etc.

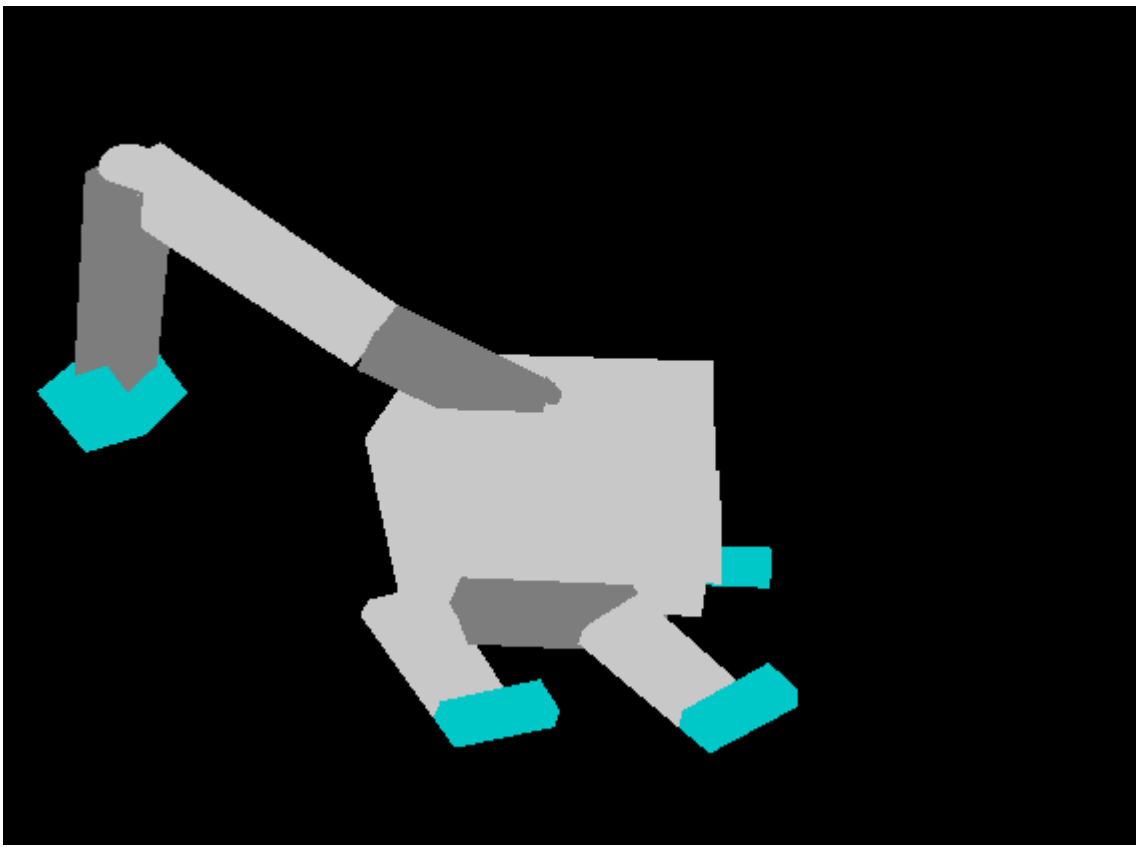
Las patas las tenía definidas en ciertas posiciones que ajuste con base a lo hecho en las llantas anteriores, sin embargo, para evitar el hecho de que no podía configurarles articulaciones, ahora instancie cada pata en individual, junto a su pie, además de asignarle una articulación definida en Window.cpp y Window.h para que al mover la pata, se mueva el pie pero igualmente, el pie se mueva solo.

```
// Pata 1
model = modelaux;
model = glm::translate(model, glm::vec3(-2.25f, -2.8f, -2.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()), glm::vec3(0.0f, 0.0f, 1.0f)); // R
model = glm::scale(model, glm::vec3(1.5f, 3.0f, 0.4f)); // Escalar la pata
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));
meshList[0]->RenderMesh();
// Pie 1
//model = modelaux;
model = glm::translate(model, glm::vec3(0.5f, -0.5f, -1.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion8()), 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(1.5f, 1.0f, 0.3f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.0f, 0.8f, 0.8f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh();
```

Por lo cual con algunas modificaciones de colores y de tamaños, instanciando las patas y los pies tenía un resultado como el siguiente:



Con el cual podía mover tanto la cola como sus patas gracias a las articulaciones.



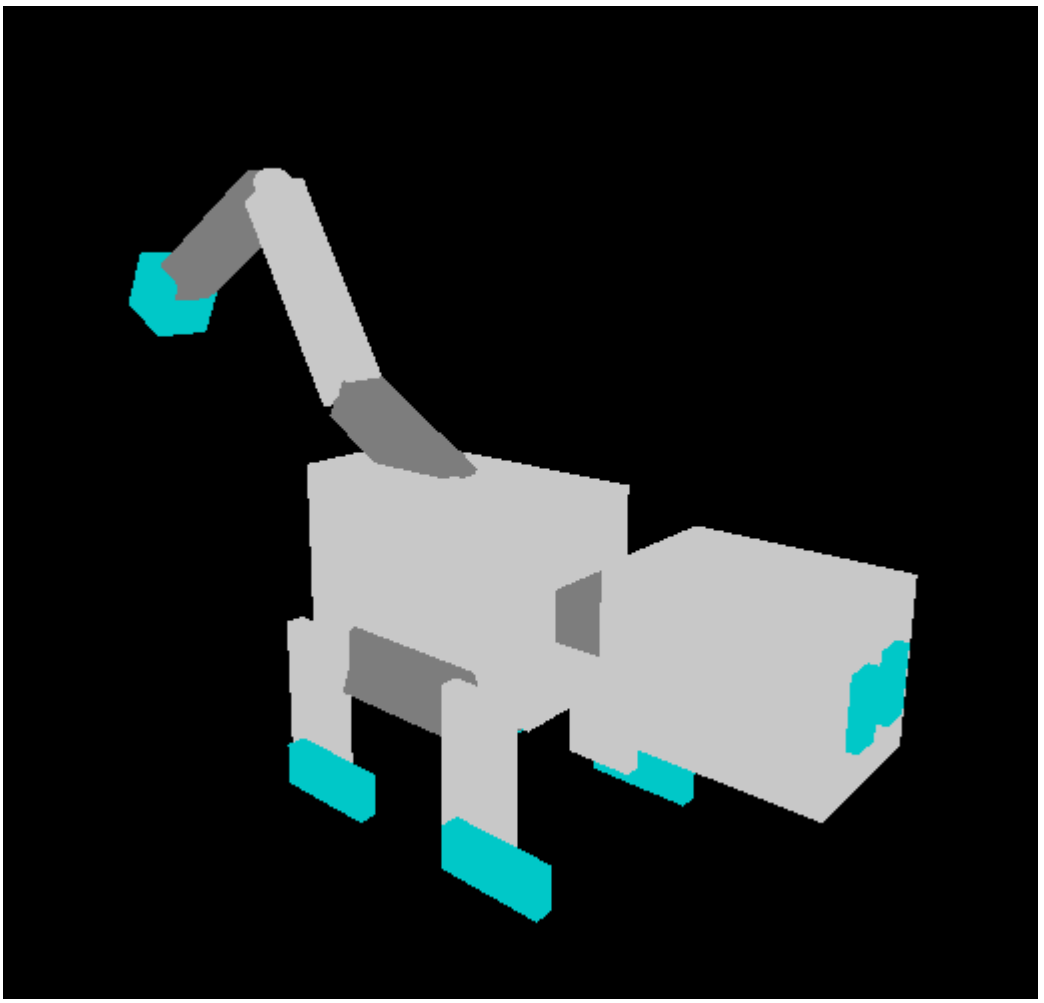
Otro aspecto a desarrollar era la cabeza, la cual tenia una jerarquía que partía del cuello, para seguir con la cabeza y después con los ojos y sus orejas por lo cual los instancie de la siguiente manera:

```
// cuello
model = modelaux;
model = glm::translate(model, glm::vec3(3.5f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::scale(model, glm::vec3(2.0f, 1.0f, 1.5f)); // Escalar la pata
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[0]->RenderMesh();

// cabeza
model = glm::translate(model, glm::vec3(-1.5f, 0.0f, 0.0f));
//model = glm::rotate(model, glm::radians(90.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::scale(model, glm::vec3(2.0f, 3.0f, 2.0f)); // Escalar la pata
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));
meshList[0]->RenderMesh();


// ojo1
model = glm::translate(model, glm::vec3(-0.5f, 0.0f, 0.2f));
//model = glm::rotate(model, glm::radians(90.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::scale(model, glm::vec3(0.1f, 0.4f, 0.2f)); // Escalar la pata
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.0f, 0.8f, 0.8f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh();
```

Sin las orejas tenía un resultado como el siguiente:



Las orejas serían instanciadas despues de color gris, siguiendo la jerarquia y aprovechando las transformaciones anteriores.

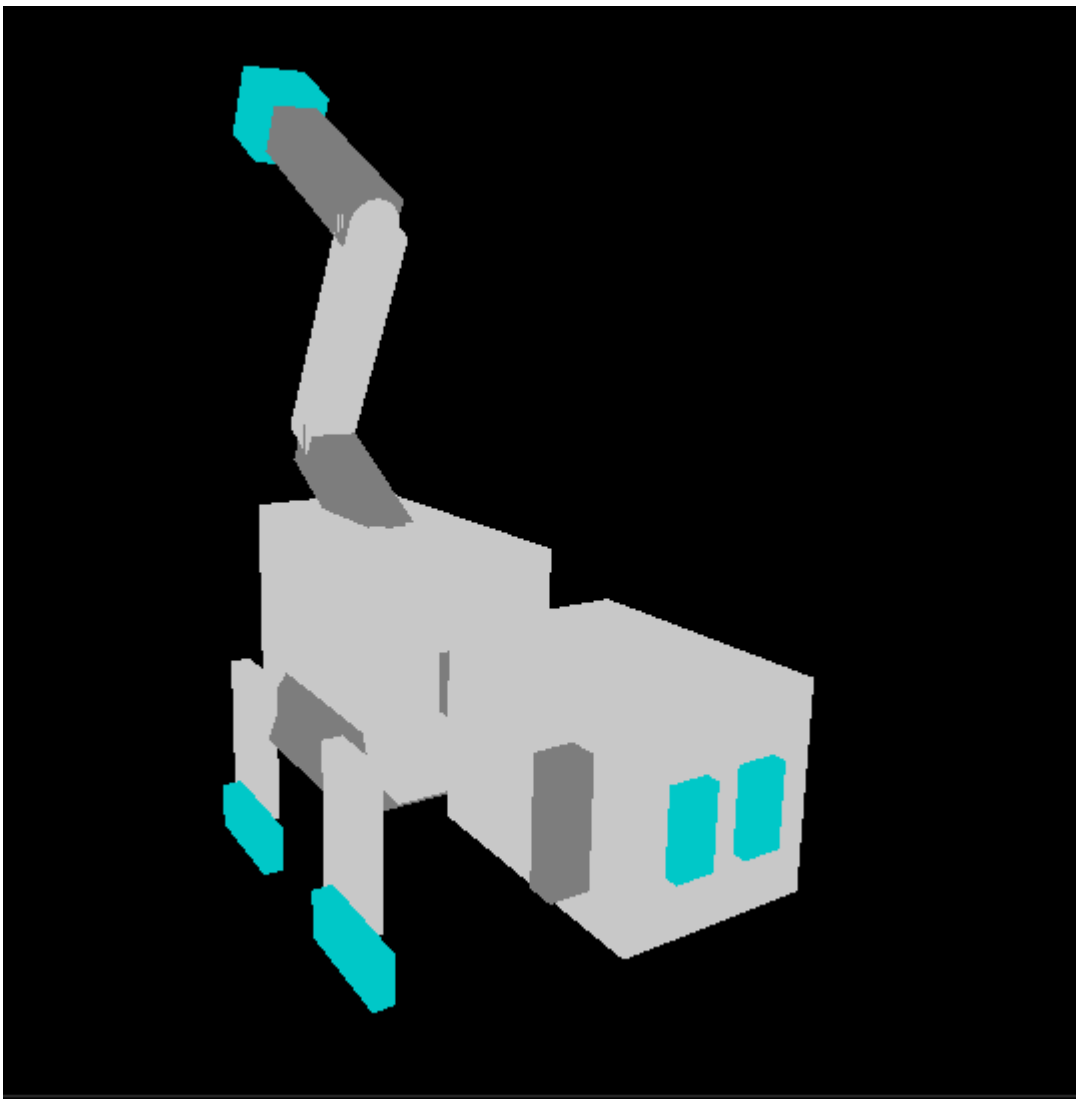
```
// ojo2
model = glm::translate(model, glm::vec3(0.0f, 0.0f, -1.9f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, gl                    odel));
color = glm::vec3(0.0f, 0.8f, 0.8f);
glUniform3fv(uniformColor, 1, glm::value_ptr(col
meshList[0]->RenderMesh();

// oreja1
model = glm::translate(model, glm::vec3(2.0f, 0.0f, -2.0f));
model = glm::scale(model, glm::vec3(1.0f, 1.5f, 1.0f)); // Escalar laoreja
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[0]->RenderMesh();

// oreja2
model = glm::translate(model, glm::vec3(0.0f, 0.0f, 6.0f));
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[0]->RenderMesh();
```

(float)(0.0F)
Buscar en línea

Para tener un resultado como el siguiente:



Articulaciones Patas:

Pata 1 se mueve con F

Pie 1 se mueve con

Pata 2 se mueve con K

Pie 2 se mueve con

Pata 3 se mueve con L

Pie 3 se mueve con

Pata 4 se mueve con M

Pie 4 se mueve con

Articulaciones Cola:

La cola se mueve en 3 partes

Cola totalmente se mueve con G,

A la mitad se mueve con H
Solo la parte final se mueve con J

Las complicaciones en este ejercicio fueron mayormente las articulaciones ya que debía considerar un orden y la jerarquía, después de entender esto me fue más fácil tanto instanciar las primitivas como asignar las articulaciones en el caso de las patas y de la cola.

Conclusiones

Con esta práctica tuve dificultades al principio ya que en la teoría me parecía muy claro la razón de la jerarquía y su funcionamiento, pero en la práctica en los primeros ejercicios me costó entenderlo y ver su importancia dentro, pero a su vez entendí que lejos de complicar la forma de instanciar varios objetos y relacionarlos, da más facilidad para ello y a su vez, exige tener rigor y orden a la hora de hacer las instancias de cada objeto, ya que sin ello, estaremos muy lejos de tener los resultados que buscamos sin seguir correctamente la jerarquía. No solo facilita y guía, proporciona una idea interesante como pensar en puntos compartidos por dos objetos, con los cuales podemos implementar lo que en este caso hemos llamado como articulaciones y darle algo de vida a nuestro objeto mayor.