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: Aguilar Pérez José Ramón

Nº de Cuenta: 317515048

GRUPO DE LABORATORIO: 02

GRUPO DE TEORÍA: 04

SEMESTRE 2025-2

FECHA DE ENTREGA LÍMITE: 09/marzo/2025

CALIFICACIÓN: _____

REPORTE DE PRÁCTICA:

1.- Ejercicios realizados.

- **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 la realización de este ejercicio, primero se agregaron las instancias para las articulaciones extras, se tenían en un principio 6 articulaciones programadas, pero como ya se habían utilizada 4 de estas para los brazos y la canasta, fue necesario agregar 2 articulaciones extras, las cuales se les asignaron las teclas I y O.

```
Window::Window(GLint windowWidth, GLint windowHeight)
{
    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;
```

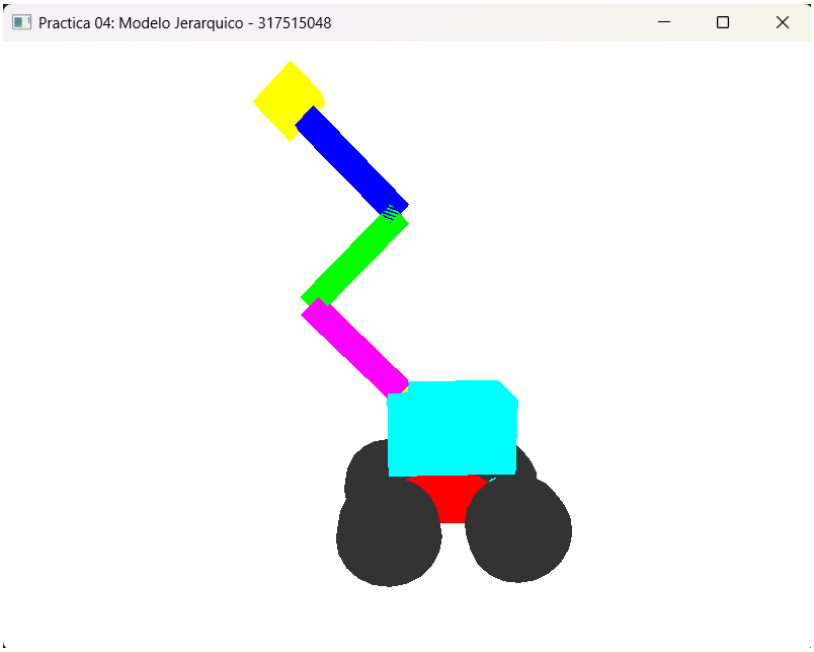
```
if (key == GLFW_KEY_I)
{
    theWindow->articulacion7 += 10.0;
}
if (key == GLFW_KEY_O)
{
    theWindow->articulacion8 += 10.0;
}
```

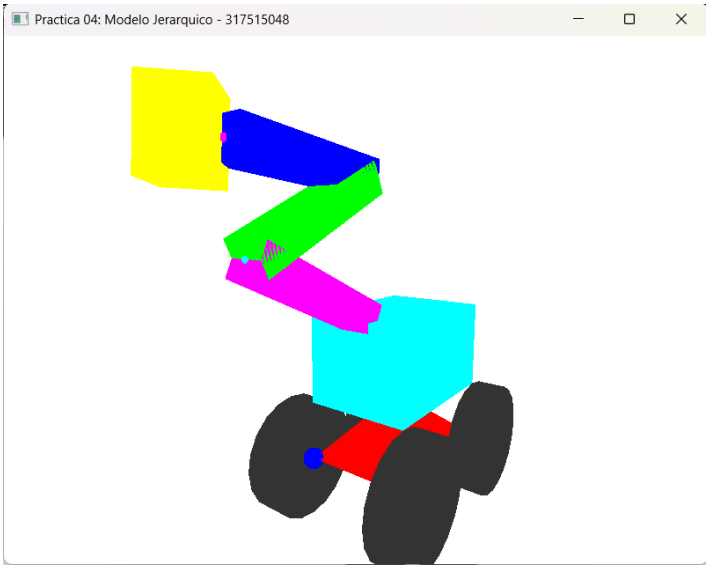
Después, se instanció la base con forma de pirámide cuadrada que servirá como punto de referencia para las llantas y sus articulaciones. En esta se hace uso de dos matrices auxiliares, con el fin de ir heredando las posiciones necesarias para las articulaciones y sus respectivas llantas.

```
//Base
model = glm::translate(model, glm::vec3(0.0f, -2.5f, 0.0f));
modelaux = model;
modelaux2 = model;
model = glm::scale(model, glm::vec3(5.0f, 2.0f, 3.5f)); //No se debe heredar
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));  //para cambiar el color
meshList[4]->RenderMesh(); //Pirámide cuadrada
model = modelaux;
```

Primero se fueron instanciado una por una la articulación con su respectiva llanta, esto con el fin de evitar confusiones a la hora de utilizar las matrices auxiliares y heredar datos no deseados. Este proceso se hizo un total de 4 veces, las teclas que controlan las llantas son: K, L, I, O. K y L mueven las llantas del lado izquierdo de la grúa, mientras que I y O las del lado derecho.

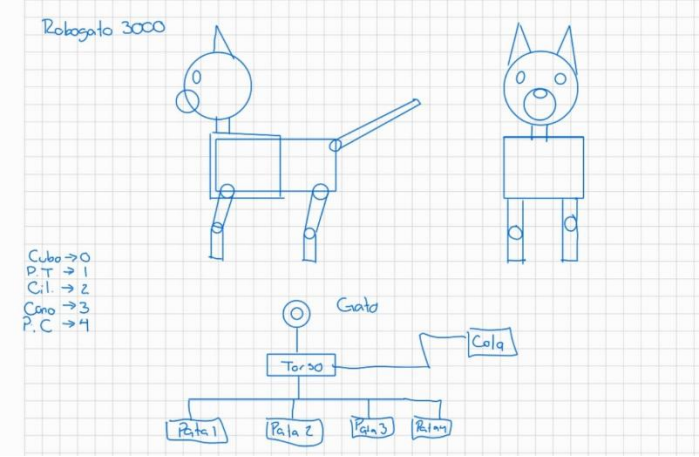
```
// Articulación llanta 1
model = glm::translate(model, glm::vec3(-2.5f, -1.0f, 1.75f)); // Traslación relativa a la base
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion5()), glm::vec3(0.0f, 0.0f, 1.0f));
modelaux2 = model; // Guardar el estado de la articulación 1 (usando modelaux2)
model = glm::scale(model, glm::vec3(3.5f, 3.5f, 3.5f)); // Escala
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(1.0f, 1.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); // Cambiar el color del objeto
sp.render(); // Dibujar la esfera

// Llanta 1 (cilindro)
model = modelaux2; // Heredar de la articulación 1
model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.5f)); // Traslación relativa a la articulación
model = glm::rotate(model, glm::radians(90.0f), glm::vec3(1.0f, 0.0f, 0.0f)); // Rotación adicional
model = glm::scale(model, glm::vec3(2.0f, 1.0f, 2.0f)); // Escala
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.2f, 0.2f, 0.2f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); // Cambiar el color del objeto
meshList[2]->RenderMeshGeometry(); // Dibujar el cilindro

// Reiniciar model a la matriz base antes de la articulación 2
model = modelaux;
```





- **Crear un animal robot 3d:**
  - **Instanciando cubos, pirámides, cilindros, conos, esferas, etc.**
  - **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 la creación del robot, se escogió como animal un gato y se hizo un boceto inicial de como estaría formado, así como la jerarquía que llevaría. Se decidió cambiar el cuerpo del gato de cubos a cilindros con esferas por motivos estéticos.



Antes de pasar a la construcción del RoboGato, es importante mencionar que hay que modificar el valor del radio de las esferas utilizado durante la construcción de la grúa (0.1). Para esta actividad se utilizó un valor de 0.5.

```
Sphere sp = Sphere(0.5, 20, 20);
```

Una vez hecho este cambio, se procede a realizar la construcción del robot. En primer lugar se instancia un cilindro que servirá como el torso del gato, además de que funge la función de nodo padre ya que de ahí se heredan las patas y la cola del gato.

```
//Torso del gato
model = glm::mat4(1.0f);
model = glm::translate(model, glm::vec3(0.0f, 4.0f, -4.0f));
modelaux = model;
modelaux2 = model;
model = glm::rotate(model, glm::radians(90.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::scale(model, glm::vec3(2.5f, 8.0f, 2.5f)); //No se debe heredar
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(0.15f, 0.15f, 0.15f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));  //para cambiar el color del objetos
meshList[2]->RenderMeshGeometry(); //Cilindro
```

Una vez que se tiene el torso, se proceden a instanciar las patas del gato, y siguiendo los pasos utilizados durante la construcción de las llantas de la grúa, se hace uso de matrices auxiliares para ir heredando valores significativos como lo son la posición de las patas. Es importante mencionar que cada pata tiene dos articulaciones: la que va pegada al torso del gato y la que sirve como el “codo”. Este proceso se repite 4 veces, y para la cola solo se cuenta con una articulación en la parte posterior del gato.

```
//Patas y cola
//Articulacion 1
model = modelaux;
model = glm::translate(model, glm::vec3(-3.0f, -1.5f, 1.5f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()), glm::vec3(0.0f, 0.0f, 1.0f));
modelaux2 = model; //Matrix auxiliar
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f)); //No se debe heredar
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.2f, 0.2f, 0.2f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
sp.render(); //Para dibujar la esfera
//Pata izquierda delantera
model = modelaux2;
model = glm::translate(model, glm::vec3(0.0f, -1.0f, 0.0f)); // Traslación relativa a la articulación
model = glm::scale(model, glm::vec3(0.8f, 2.5f, 0.8f)); // Escala
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.15f, 0.15f, 0.15f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); // Cambiar el color del objeto
meshList[2]->RenderMeshGeometry(); // Dibujar el cilindro
//Articulacion 2
model = modelaux2;
model = glm::translate(model, glm::vec3(0.0f, -2.5f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()), glm::vec3(0.0f, 0.0f, 1.0f));
modelaux2 = model; //Matrix auxiliar
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f)); //No se debe heredar
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.2f, 0.2f, 0.2f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
sp.render(); //Para dibujar la esfera
//Segunda parte de la Pata izquierda delantera
model = modelaux2;
model = glm::translate(model, glm::vec3(0.0f, -1.0f, 0.0f)); // Traslación relativa a la articulación
model = glm::scale(model, glm::vec3(0.8f, 2.5f, 0.8f)); // Escala
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.3f, 0.3f, 0.3f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); // Cambiar el color del objeto
meshList[2]->RenderMeshGeometry(); // Dibujar el cilindro
```

```
//Articulacion 9 -> Se mueve con la tecla N
model = modelaux;
model = glm::translate(model, glm::vec3(4.7f, 2.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion9()), glm::vec3(1.0f, 0.0f, 0.0f));
modelaux2 = model; //Matrix auxiliar
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f)); //No se debe heredar
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.2f, 0.2f, 0.2f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
sp.render(); //Para dibujar la esfera
//Cola
model = modelaux2;
model = glm::rotate(model, glm::radians(-135.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::translate(model, glm::vec3(0.0f, 2.5f, 0.0f)); // Traslación relativa a la articulación
modelaux2 = model;
model = glm::scale(model, glm::vec3(0.8f, 5.0f, 0.8f)); // Escala
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.15f, 0.15f, 0.15f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); // Cambiar el color del objeto
meshList[2]->RenderMeshGeometry(); // Dibujar el cilindro
model = modelaux2;
model = glm::translate(model, glm::vec3(0.0f, 4.0f, 0.0f)); // Traslación relativa a la articulación
model = glm::scale(model, glm::vec3(0.8f, 3.0f, 0.8f)); // Escala
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.3f, 0.3f, 0.3f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); // Cambiar el color del objeto
meshList[2]->RenderMeshGeometry(); // Dibujar el cilindro
```

Como ahora son 9 articulaciones, se tuvo que agregar esta última en *Window.h* y *Window.cpp,* donde se le asignó la letra N. Quedando así:

- Pata delantera izquierda: F, G
- Pata trasera izquierda: H, J
- Pata delantera derecha: K, L
- Pata trasera derecha: I, O
- Cola: N

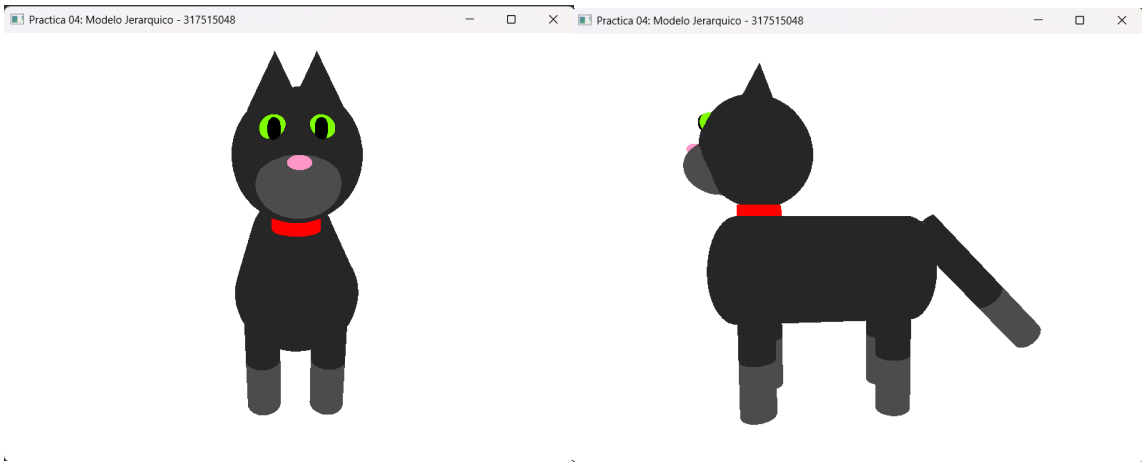
```
if (key == GLFW_KEY_N)
{
    theWindow->articulacion9 += 10.0;
}
```

Finalmente, para el cuello del gato se utilizó un cilindro, mientras que la cabeza está conformada por varias esferas que simulan su hocico, nariz y ojos, mientras que las orejas se hicieron a partir de pirámides triangulares.

```
//Collar
model = glm::mat4(1.0f);
model = glm::translate(model, glm::vec3(-3.0f, 6.75f, -4.0f));
modelaux = model;
modelaux2 = model;
model = glm::scale(model, glm::vec3(1.0f, 0.5f, 1.0f)); //No se debe heredar
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));  //para cambiar el color del objetos
meshList[2]->RenderMeshGeometry(); //Pirámide cuadrada

//Cabeza
model = modelaux;
model = glm::translate(model, glm::vec3(0.0f, 2.6f, 0.0f));
modelaux = model;
model = glm::scale(model, glm::vec3(5.0f, 5.0f, 5.0f)); //No se debe heredar
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.15f, 0.15f, 0.15f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));  //para cambiar el color del objetos
sp.render();
//Hocico
model = modelaux;
model = glm::translate(model, glm::vec3(-1.7f, -0.7f, 0.0f));
modelaux = model;
model = glm::scale(model, glm::vec3(3.0f, 2.4f, 3.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.3f, 0.3f, 0.3f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));  //para cambiar el color del objetos
sp.render();
//Nariz
model = modelaux;
model = glm::translate(model, glm::vec3(-1.0f, 0.8f, 0.0f));
modelaux = model;
model = glm::scale(model, glm::vec3(0.8f, 0.5f, 0.8f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(1.0f, 0.588f, 0.784f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
sp.render();
```

```
//Ojo derecho
model = modelaux;
model = glm::translate(model, glm::vec3(0.0f, 0.0f, -1.5f));
modelaux = model;
model = glm::scale(model, glm::vec3(1.0f, 0.9f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.5f, 1.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color de
sp.render();
model = glm::translate(model, glm::vec3(-0.32f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.5f, 0.8f, 0.5f));
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)); //para cambiar el color de
sp.render();
//Oreja Izquierda
model = modelaux;
model = glm::translate(model, glm::vec3(1.3f, 1.6f, 0.0f));
modelaux = model;
model = glm::rotate(model, glm::radians(-90.0f), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::scale(model, glm::vec3(2.0f, 2.0f, 3.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.15f, 0.15f, 0.15f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color de
meshList[1]->RenderMesh();
//Oreja Derecha
model = modelaux;
model = glm::translate(model, glm::vec3(0.0f, 0.0f, 1.5f));
modelaux = model;
model = glm::rotate(model, glm::radians(-90.0f), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::scale(model, glm::vec3(2.0f, 2.0f, 3.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.15f, 0.15f, 0.15f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));  //para cambiar el color d
meshList[1]->RenderMesh();
```

```
//Pecho
model = modelaux2;
model = glm::translate(model, glm::vec3(-1.0f, -2.75f, 0.0f));
modelaux2 = model;
model = glm::scale(model, glm::vec3(2.5f, 5.0f, 5.0f)); //No se debe heredar
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.15f, 0.15f, 0.15f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));  //para cambiar el color del objetos
sp.render();
//Parte trasera
model = modelaux2;
model = glm::translate(model, glm::vec3(8.0f, 0.0f, 0.0f));
modelaux2 = model;
model = glm::scale(model, glm::vec3(2.5f, 5.0f, 5.0f)); //No se debe heredar
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.15f, 0.15f, 0.15f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));  //para cambiar el color del objetos
sp.render();
```


Practica 04: Modelo Jerarquico - 317515048
Practica 04: Modelo Jerarquico - 317515048


Practica 04: Modelo Jerarquico - 317515048

2.- Problemas presentados

No se presentaron problemas significativos a la hora de realizar los ejercicios.

3.- Conclusión:

a. Los ejercicios del reporte: Complejidad, Explicación.

El ejercicio de esta práctica fue muy interesante de realizar, ya que la complejidad de este era más elevada de lo que estaba acostumbrado. Gracias a la explicación del funcionamiento de las matrices auxiliares, así como la creación de articulaciones en el laboratorio, la práctica se llevó a cabo de manera exitosa.

b. Comentarios generales: Faltó explicar a detalle, ir más lento en alguna explicación, otros comentarios y sugerencias para mejorar desarrollo de la práctica

La explicación de la implementación de una matriz auxiliar para la generación de figuras fue clara y precisa. Además, también fue buena la explicación de como se heredan los datos para así generar las articulaciones y objetos pegados a estas.

c. Conclusión

Esta práctica me permitió comprender mejor cómo funcionan las matrices que generan las figuras en el plano, además de descubrir que se pueden heredar ciertos valores que permiten generar nuevas figuras con los valores heredados. Esto último fue de gran valor, ya que ahora es más fácil generar figuras compuestas de varios elementos, ya que antes calculaba a mano alzada la distancia necesaria o la rotación para formar un dibujo compuesto, y gracias al empleo de matrices auxiliares, ya se hereda l aposición deseada para la composición de dibujos en 3D.

1. Bibliografía en formato APA

- OpenGL Programming/Basics/Color. (s/f). Wikibooks.org. Recuperado el 14 de febrero de 2025, de https://en.wikibooks.org/wiki/OpenGL_Programming/Basics/Color
- Tutorial 3: Matrices. (s/f). Opengl-tutorial.org. Recuperado el 1 de marzo de 2025, de http://www.opengl-tutorial.org/es/beginners-tutorials/tutorial-3-matrices/