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: Casillo Martinez Diego Leonardo

N.º de Cuenta: 319041538

GRUPO DE LABORATORIO: 11

GRUPO DE TEORÍA: 06

SEMESTRE 2024-2

FECHA DE ENTREGA LÍMITE: 04/09/2024

CALIFICACIÓN: _____

REPORTE DE PRÁCTICA:

1.- Desarrollo

Ejercicio 1: este ejercicio plantea **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 se hico una matriz de apoyo 2 con la cual generamos un prisma triangular en donde salvamos el modelo antes de que se declaren las articulaciones, para así no generar problemas, esto se hizo de la siguiente manera:

```
//------- Base de llantas
model = glm::translate(model, glm::vec3(0.00f, 3.0f, -4.0f));
modelaux = model; // solo   quiero que se herede traslacion la escala no
model = glm::scale(model, glm::vec3(3.5f, 2.5f, 3.5f));  // Ajusta el tamaño del cubo
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));

color = glm::vec3(1.0f, 0.0f, 1.0);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
// Renderizar el cubo que rota con la articulación
meshList[4]->RenderMeshGeometry();  // Renderiza el cubo
```

Una vez completado este paso, generamos una a una las llantas de la grúa. Para ello, consideramos que se ha creado un cilindro, el cual se rota 90 grados alrededor del eje para que adquiera la forma de una llanta. Luego, se realizó una traslación para posicionar las llantas correctamente. Este proceso se replicó para cada llanta, con la ventaja de que solo tuvimos que trasladar un único eje. Esta operación se llevó a cabo de la siguiente manera:

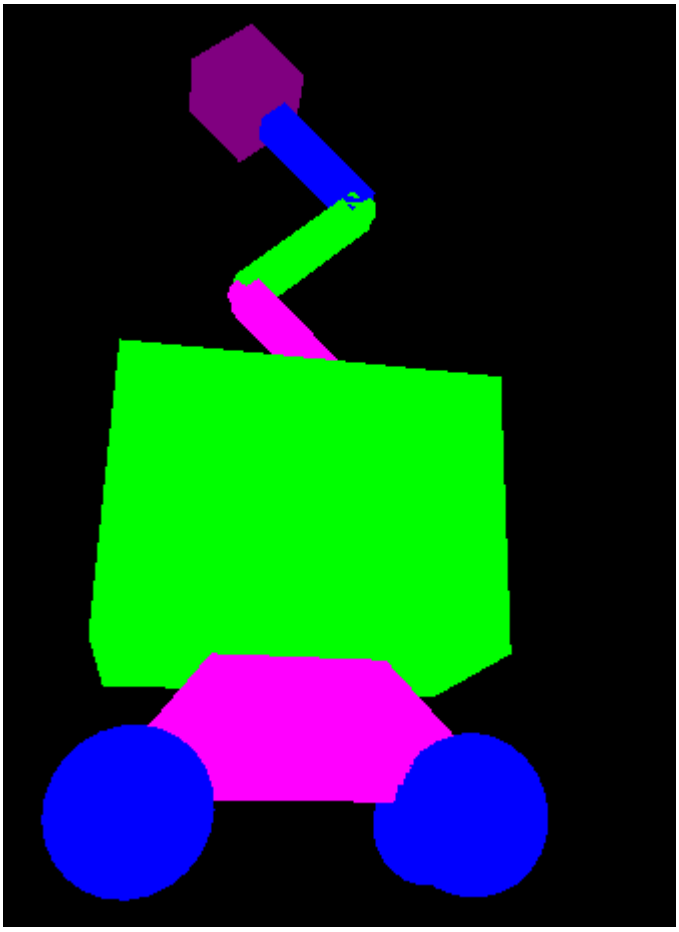
```
model = modelaux;
model = glm::translate(model, glm::vec3(-1.5f, -1.5f, 1.7f));
model = glm::rotate(model, glm::radians(-90.0f), glm::vec3(1.0f, 0.0f, 0.0f));
modelaux = model;
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion5()), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::scale(model, glm::vec3(.7f, .2f, .7f));

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

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

// Renderizar el cubo que rota con la articulación
meshList[2]->RenderMeshGeometry();
```

Algo a considerar es que al rotar 90 grados el eje de profundidad paso a ser el Y, y esto se puede ver al momento de establecer las traslaciones de las llantas traseras:

```
model = modelaux;
model = glm::translate(model, glm::vec3(3.5f, 3.5f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion7()), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::scale(model, glm::vec3(.7f, .2f, .7f));

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

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

// Renderizar el cubo que rota con la articulación
meshList[2]->RenderMeshGeometry();
```

Nota: algo a considerar es que se declararon 2 articulaciones nuevas para que se pueda rotar cada llanta

Con esto el resultado que obtenemos es el siguiente:



En donde para rotar las llantas hacemos uso de las teclas: K,L,Z,X, algo a considerar es que no se ve tan apreciable la rotación y se tiene que acercar a la base de las llantas.

Ejercicio 2:

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.

Esta práctica es similar a la anterior. En ella, estructuramos todo desde cero partiendo de un modelo jerárquico de un gato, donde:

- **La base principal** es el torso, compuesto por un cubo y un cilindro.
- **Sus hijos** son las patas, que a su vez tienen dos segmentos:
    - Articulación
    - Parte superior de la pata
    - Articulación
    - Parte inferior de la pata

Además, se añadió una articulación al cuello del gato, simulando un collar y permitiendo que la cabeza pueda rotar. También se incluyeron las siguientes partes de la cola:

- Articulación
- Parte 1 de la cola
- Articulación
- Parte 2 de la cola

Para las patas podemos ver la siguiente estructura de código:

Esta práctica es similar a la anterior. En ella, estructuramos todo desde cero partiendo de un modelo jerárquico de un gato, donde:

- **La base principal** es el torso, compuesto por un cubo y un cilindro.
- **Sus hijos** son las patas, que a su vez tienen dos segmentos:
    - Articulación
    - Parte superior de la pata
    - Articulación
    - Parte inferior de la pata

Además, se añadió una articulación al cuello del gato, simulando un collar y permitiendo que la cabeza pueda rotar. También se incluyeron las siguientes partes de la cola:

- Articulación
- Parte 1 de la cola
- Articulación

- Parte 2 de la cola

Para las patas podemos ver una estructura de código muy similar a la cola que se plasma de la siguiente forma:

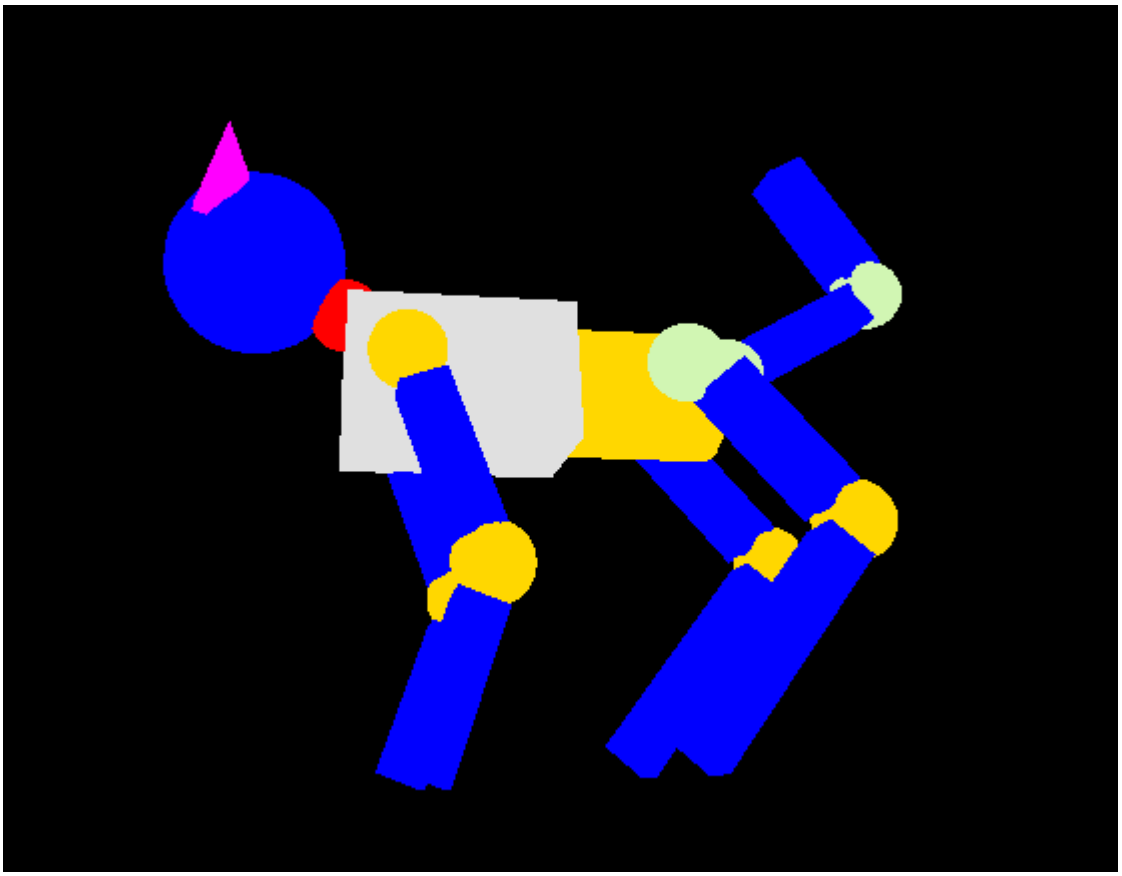
```
//------------------  (Articulación 1) ------------------

model = glm::translate(model, glm::vec3(-0.6f, 5.0f, -3.0f));
modelaux2 = model;// guardamos el punto inciial de la articulacion ya qure es lo unico que deseo se guarde

// Aplicar la rotación de la articulación 1
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()), glm::vec3(0.0f, 0.0f, 1.0f));
modelaux = model;
//todo lo de abajo no se hereda
// Escalar la esfera para que sea pequeña
model = glm::scale(model, glm::vec3(0.4f, 0.4f, 0.4f));
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 para la esfera (articulación)
color = glm::vec3(1.0f, 0.843f, 0.0f);  // Color oro para la esfera
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
// Dibujar la esfera (articulación 1)
sp.render();  // Renderiza la esfera

// ---------------- - Cubo que rota con la Articulación 1 ---------------- -
model = modelaux;  // heredamos el modelo en donde se encuentra ubicada nuestra articulacion 1
// Trasladar el cubo a la posición deseada, en este caso debajo de la esfera (articulación 1)
model = glm::translate(model, glm::vec3(0.4f, -1.1f, 0.05f));  // Ajusta la traslación del cubo
model = glm::rotate(model, glm::radians(20.0f), glm::vec3(0.0f, 0.0f, 1.0f));
modelaux = model;
model = glm::scale(model, glm::vec3(.5f, 1.6f, .5f));  // Ajusta el tamaño del cubo
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
// Color para el cubo
color = glm::vec3(0.0f, 0.0f, 1.0);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
// Renderizar el cubo que rota con la articulación
meshList[0]->RenderMesh();  // Renderiza el cubo
```

```
//------------------  (Articulación 2) ------------------
model = modelaux;
model = glm::rotate(model, glm::radians(-20.0f), glm::vec3(0.0f, 0.0f, 1.0f));
// para evitar problemaas futuros al momento de realizar las otras patas
// anulamos pde mientras el eje en z rotado anteriormente
model = glm::translate(model, glm::vec3(0.4f, -1.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()), glm::vec3(0.0f, 0.0f, 1.0f));
modelaux = model;
model = glm::scale(model, glm::vec3(0.4f, 0.4f, 0.4f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
// Color para la esfera (articulación)
color = glm::vec3(1.0f, 0.843f, 0.0f);  // Color oro para la esfera
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
// Dibujar la esfera (articulación 1)
sp.render();  // Renderiza la esfera
// ---------------- - Cubo que rota con la Articulación 2 ---------------- -
model = modelaux;
model = glm::translate(model, glm::vec3(-0.4f, -1.1f, 0.05f));  // Ajusta la traslación del cubo
model = glm::rotate(model, glm::radians(-20.0f), glm::vec3(0.0f, 0.0f, 1.0f));
// como esta hasta el ultimo en la gerarquia ni siquiera se va a heredar algo de la escala
model = glm::scale(model, glm::vec3(.5f, 1.6f, .5f));  // Ajusta el tamaño del cubo
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
// Color para el cubo
color = glm::vec3(0.0f, 0.0f, 1.0);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
// Renderizar el cubo que rota con la articulación
meshList[0]->RenderMesh();  // Renderiza el cubo
model = modelaux2; // regresamos a la parte baser de nuestro modelo
```

Para la cabeza se plasma de la siguiente forma:

```
//-----   PARA EL ROSTRO -----------------
///----  ARTICULACION DEL CUELLO QUE SIMULA COLLAR ROJO
model = glm::translate(model, glm::vec3(-.6f, .8f, -1.0f));
model = glm::rotate(model, glm::radians(40.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion9()), glm::vec3(0.0f, 0.0f,
modelaux = model;
//todo lo de abajo no se hereda
// Escalar la esfera para que sea pequeña
model = glm::scale(model, glm::vec3(0.4f, .4f, 0.4f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(1.0f, 0.0f, 0.0f);  // Color oro para la esfera
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
// Dibujar la esfera (articulación 1)
sp.render();  // Renderiza la esfera
//-------- PARA EL ROSTRO ---------------
model = modelaux;
model = glm::translate(model, glm::vec3(-.4f, 1.1f, 0.0f));
model = glm::rotate(model, glm::radians(-40.0f), glm::vec3(0.0f, 0.0f, 1.0f));
modelaux = model;
//todo lo de abajo no se hereda
// Escalar la esfera para que sea pequeña
//model = glm::scale(model, glm::vec3(0.4f, .4f, 0.4f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(.0f, 0.0f, 1.0f);  // Color oro para la esfera
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
// Dibujar la esfera (articulación 1)
sp.render();  // Renderiza la esfera
//--- --------------PARA LAS OREJAS (IZQUIERDA )-----------------
model = modelaux;
model = glm::translate(model, glm::vec3(-.30f, .7f, -.8f));
modelaux = model;
model = glm::rotate(model, glm::radians(-30.0f), glm::vec3(1.0f, 0.0f, 0.0f));

model = glm::scale(model, glm::vec3(.2f, 1.0f, .2f));  // Ajusta el tamaño del cubo
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
// Color para el cilindro
```

Algo a considerar es que aquí se hizo uso de 11 articulaciones con teclas F, G,H,J,K,L,Z,X,C,V,B, las cuales van en orden para las patas de la 1 a la 4 la cabeza y los dos segmentos de cola. Gracias a ello tenemos el siguiente resultado:



2.- problemas que surgieron:

Durante esta práctica, inicialmente surgieron problemas de rotación debido a una falta de comprensión completa sobre el uso de matrices auxiliares. Sin embargo, una vez que se dominó este concepto, la práctica se volvió mucho más clara y fluida. Este entendimiento facilitó la implementación de las rotaciones correctamente, haciendo que la actividad fuera tanto sencilla como entretenida.

3.- Conclusión:

Este ejercicio no presento una gran dificultad, sin embargo, me ayudaron a mejorar mi percepción espacial y mi comprensión de las rotaciones. Además, este ejercicio me permitió entender mejor cómo funciona el modelo jerárquico y las matrices auxiliares.

En general, el ejercicio fue entretenido. Aunque al principio no entendí completamente el video que se subió, comprendí bien la explicación del profesor. Después de ver el video, me quedó completamente claro cómo funciona el modelo jerárquico.

4.- Bibliografía:

Tutorial 3 : Matrices. (s. f.). https://www.opengl-tutorial.org/es/beginners-tutorials/tutorial-3-matrices/