UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE INGENIERÍA ELÉCTRICA

INGENIERÍA EN COMPUTACIÓN

LABORATORIO DE COMPUTACIÓN GRÁFICA e INTERACCIÓN HUMANO COMPUTADORA

EJERCICIOS DE CLASE Nº 4

NOMBRE COMPLETO: Sanchez Villalpando Johan

Nº de Cuenta: 422028657

GRUPO DE LABORATORIO: 2

GRUPO DE TEORÍA: 4

SEMESTRE 2025-2

FECHA DE ENTREGA LÍMITE: 04/03/25

CALIFICACIÓN: _____

EJERCICIOS DE SESIÓN:

1. **Actividades realizadas. Una descripción de los ejercicios y capturas de pantalla de bloques de código generados y de ejecución del programa**

   Para este ejercicio se requiere avanzar con la grúa vista en clase, añadiendo la base, el ultimo brazo y la cabina, con sus respectivas rotaciones y articulaciones.

   Primero se agrega la base, para la cual se usa un cubo normal, en mi caso elegí un color gris, usando el cubo que ya tenemos en la meshlist.

```
//Para la base

color = glm::vec3(0.2f, 0.2f, 0.2f);  //gris
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
model = glm::translate(model, glm::vec3(0.0f,5.0f, -4.0f));
model = glm::scale(model, glm::vec3(4.0f, 3.0f, 4.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
meshList[0]->RenderMesh(); //cubo cabina

model = glm::mat4(1.0);
```

   Para la siguiente articulación y brazo bastó con copiar el código de un brazo y articulación anterior (1 y 2) y rotar 135° para tener el ángulo deseado, sin modificar el escalado.

```
//articulación 3 extremo derecho del segundo brazo
model = glm::translate(model, glm::vec3(0.0f, -2.5f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion3()), glm::vec3(0.0f, 0.0f, 1.0f));
modelaux = model;
//dibujar una pequeña esfera
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
sp.render();
model = modelaux;

//Brazo 3
model = glm::translate(model, glm::vec3(2.5f, 0.0f, 0.0f));
//model = glm::rotate(model, glm::radians(135.0f), glm::vec3(0.0f, 0.0f, 1.0f));
modelaux = model;
model = glm::scale(model, glm::vec3(5.0f, 1.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
//la línea de proyección solo se manda una vez a menos que en tiempo de ejecución
//se programe cambio entre proyección ortogonal y perspectiva
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, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[0]->RenderMesh(); //dibuja cubo, pirámide triangular, pirámide base cuadrangular
```

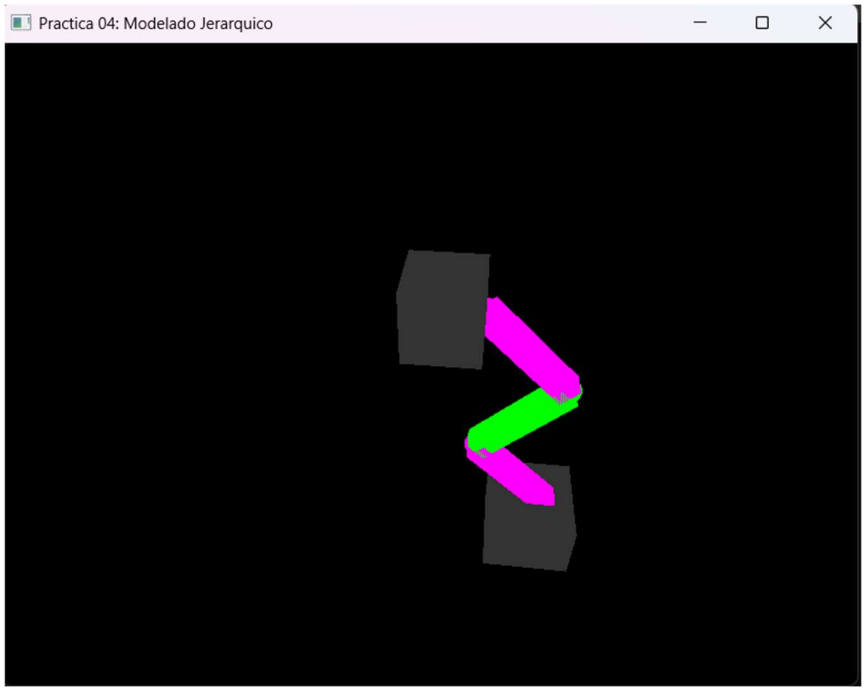
Finalmente, el detalle más “complejo” fue acomodar el ángulo para que la canasta se vea bien, para ello la esfera de la articulación se tuvo que rotar -135°, a modo que cuando se aplique una rotación en la canasta, esta gire correctamente en el eje Y, y no de la apariencia de girar sobre otro eje debido al ángulo de rotación.

```
//Para articulación antes de canasta
//articulación 5
model = glm::translate(model, glm::vec3(2.5f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(-135.0f), glm::vec3(0.0f, 0.0f, 1.0f));
modelaux = model;
//dibujar una pequeña esfera
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
sp.render();
model = modelaux;

//Para canasta
color = glm::vec3(0.2f, 0.2f, 0.2f);  //gris
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
model = glm::translate(model, glm::vec3(-1.5f, 0.0f, 0.0f));
modelaux = model;
model = glm::scale(model, glm::vec3(3.0f, 3.0f, 3.0f));
//model = glm::rotate(model, glm::radians(135.0f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion4()), glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANSPUESTA
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
meshList[0]->RenderMesh(); //cubo cabina
model = modelaux;
```

El cambio en el ángulo de la esfera fue heredado a la canasta.
En la canasta se aplicó una rotación de 135° para logar el efecto similar a una grua normal, además debido a la rotación en -135° de la articulación, la traslación tuvo que realizarse en -1.5f, en lugar de 2.5f.

Finalmente se obtiene el resultado deseado.



2. **Problemas presentados. Listar si surgieron problemas a la hora de ejecutar el código**

   El principal problema fue al rotar la canasta, ya que sin la rotación de -135° en la articulación, al querer rotar la canasta sobre el Eje Y, esta aparentaba rotar sobre otro eje, sin importar el eje sobre el cual se elija la rotación, esta se mantenía sin ser la correcta.

3. **Conclusión:**
   a. **Los ejercicios de la clase: Complejidad, explicación**
   b. **Comentarios generales: Faltó explicar a detalle, ir más lento en alguna explicación, otros comentarios y sugerencias.**

   El ejercicio fue interesante, ya que empezamos a trabajar con teclas y articulaciones, lo que da movimiento a las figuras, esto próximamente se verá más aplicado en el proyecto final.

   El hecho de encadenar figuras y anclarlas a un eje me parece un concepto interesante y que da pie a muchas funciones, como los videojuegos, también plantea la duda de cómo es un modelo articulado de algún videojuego AAA.

   Sobre la explicación, la sentí algo acelerada, por lo que me perdí en varias partes, lo que dificultó un poco entender bien todo lo que se había realizado.