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: Uriarte Ortiz Enrique Yahir

Nº de Cuenta: 318234757

GRUPO DE LABORATORIO: 02

GRUPO DE TEORÍA: 04

SEMESTRE 2025-1

FECHA DE ENTREGA LÍMITE: Martes 03 de Septiembre del 2024

CALIFICACIÓN: _____

EJERCICIOS DE SESIÓN:

1. *Actividades realizadas.*

1) Terminar de Construir la grúa con:

- ✓ Cuerpo de la grúa (prisma rectangular).
- ✓ Brazo de 3 partes, 4 articulaciones, 1 canasta.

Para comenzar con la actividad, primero agregué la base a la jerarquía, la cual la tuve que colocar antes que la primera articulación definida, para esto copié el código para definir el primer brazo, pero modificando la escala para ponerla acorde a la imagen de referencia, además heredé la translación de la figura a las demás para que se encontraran a la misma altura, luego de manera independiente moví el prisma rectangular un poco hacia X positivo y en Y un poco hacia el lado negativo, esto para acomodarla mejor con respecto a la primer articulación por que esta se encuentra dentro de la base, y por último para reconocer de mejor manera la base, la definí de color roja.

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

model = glm::translate(model, glm::vec3(0.0f, 6.0f, -4.0f));//Base
modelaux = model;
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();
model = modelaux;
```

Fig. 1. *Código para definir la base.*

Los dos primeros brazos son los mismos que se definieron durante la clase, pero las articulaciones tuve que cambiarles las escalas para que se pudieran identificar de mejor manera. Las articulaciones las definí de color azul, mientras que los brazos los definí de un tono rosa.

```
//Articulación 1
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()), glm::vec3(0.0f, 0.0f, 1.0f));
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(-2.0f, 2.0f, 0.0f));//Brazo N.1.
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));
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 = modelaux;
```

Fig. 2. *Código para definir la articulación 1 y el brazo 1.*

```
model = glm::translate(model, glm::vec3(2.5f, 0.0f, 0.0f));//Articulación 2
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()), glm::vec3(0.0f, 0.0f, 1.0f));
modelaux = model;
model = glm::scale(model, glm::vec3(0.8f, 0.8f, 0.8f));
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 = modelaux;

model = glm::translate(model, glm::vec3(0.0f, -2.5f, 0.0f));//Brazo N.2.
modelaux = model;
model = glm::scale(model, glm::vec3(1.0f, 5.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(1.0f, 0.0f, 0.5f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh();
model = modelaux ;
```

Fig. 3. *Código para definir la articulación 2 y el brazo 2.*

Para definir el tercer brazo copié el código del segundo brazo y lo modifiqué, lo primero fue definir la nueva posición con “translate”, de acuerdo con la posición de la herencia de la última figura, esta debía tener su largo hacia X, por lo que recorrí el centro de la figura hacia este eje en la mitad de largo que tienen los brazos, así mismo coloqué el código para heredar la coordenada a la siguiente articulación, en “scale” intercambié los valores entre X y Y, por lo mismo de que el largo de la figura ahora estaba en X y por último le asigne su color rosa.

```
model = glm::translate(model, glm::vec3(0.0f, -2.5f, 0.0f));//Articulación 3
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion3()), glm::vec3(0.0f, 0.0f, 1.0f));
modelaux = model;
model = glm::scale(model, glm::vec3(0.8f, 0.8f, 0.8f));
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 = modelaux;

model = glm::translate(model, glm::vec3(2.5f, 0.0f, 0.0f));//Brazo N.3
modelaux = model;
model = glm::scale(model, glm::vec3(5.0f, 1.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(1.0f, 0.0f, 0.5f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[0]->RenderMesh(); //dibuja cubo y pirámide triangular
model = modelaux;
```

Fig. 4. *Código para definir la articulación 3 y el brazo 3.*

Por último, definí la cabina y su articulación, primero la articulación, esta la recorrí con “translate” hacia la misma dirección que la articulación 2, ya que se encontraba en la misma dirección que esta, pero claro más arriba en Y. Donde más cambio, fue en la rotación, a diferencia de las demás articulaciones esta debía girar sobre el eje Y, desde una perspectiva de frente como en la imagen de referencia, pero ya observando la herencia de las figuras anteriores debía indicarle que girará con respecto a dos ejes, en mi caso la configuración fue que girara hacia menos X y hacia Y positivo al mismo tiempo, después declare la herencia para las coordenadas de la cabina.

Por último, para la cabina la recorrí en X para que se distinguiera de la articulación, copié la misma instrucción de rotación que el primer brazo para

acomodarla y la definí para que fuera un pequeño cubo, simulando la cabina, para que esta fuera también fácil de identificar la asigne de color verde.

```
model = glm::translate(model, glm::vec3(2.5f, 0.0f, 0.0f));//Articulación 4
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion4()), glm::vec3(-1.0f, 1.0f, 0.0f));
modelaux = model;
model = glm::scale(model, glm::vec3(0.8f, 0.8f, 0.8f));
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 = modelaux;

model = glm::translate(model, glm::vec3(2.0f, 0.0f, 0.0f));//Cabina
model = glm::rotate(model, glm::radians(135.0f), glm::vec3(0.0f, 0.0f, 1.0f));
modelaux = model;
model = glm::scale(model, glm::vec3(2.5f, 2.5f, 2.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.0f, 1.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[0]->RenderMesh(); //dibuja cubo y pirámide triangular

glUseProgram(0);
mainWindow.swapBuffers();
```

Fig. 5. *Código para definir la articulación 4 y la Cabina.*

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

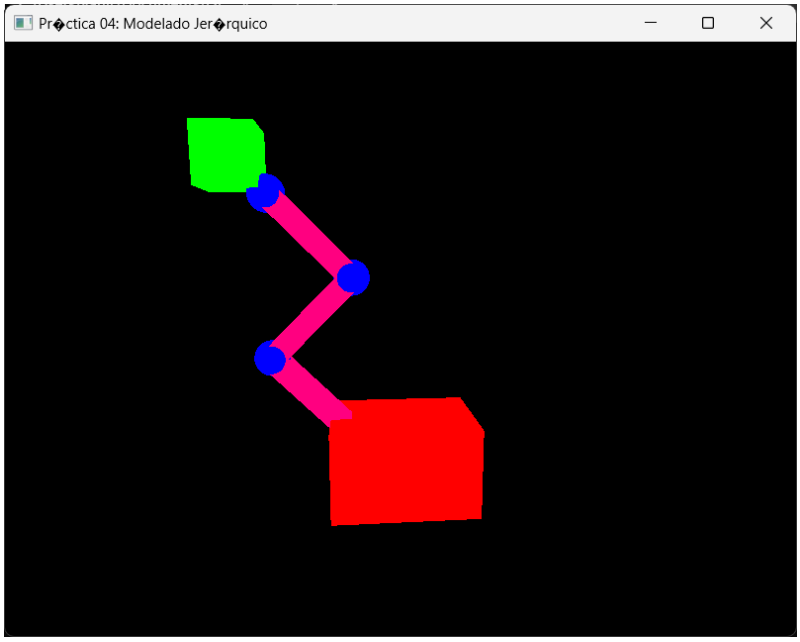

Fig. 6. *Ejecución del programa.*

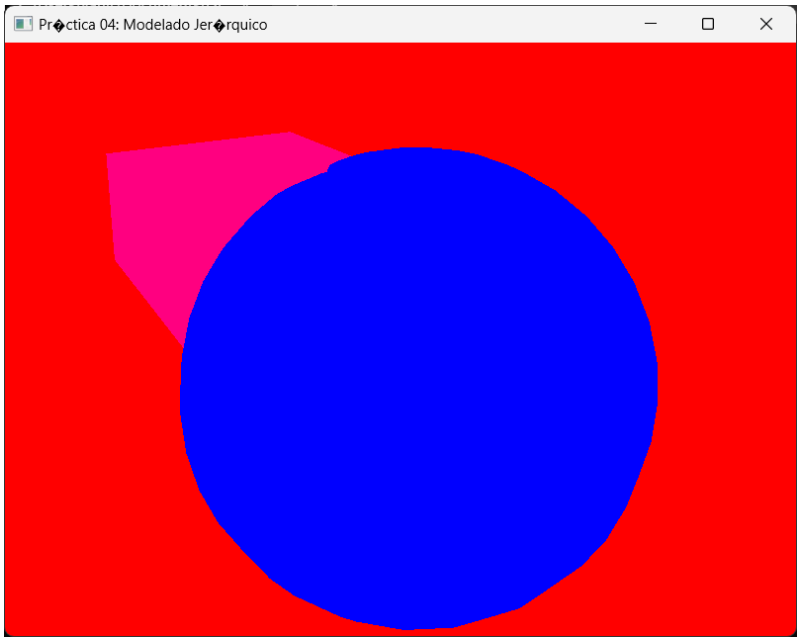

Fig. 7. *Ejecución del programa, articulación dentro de la base.*

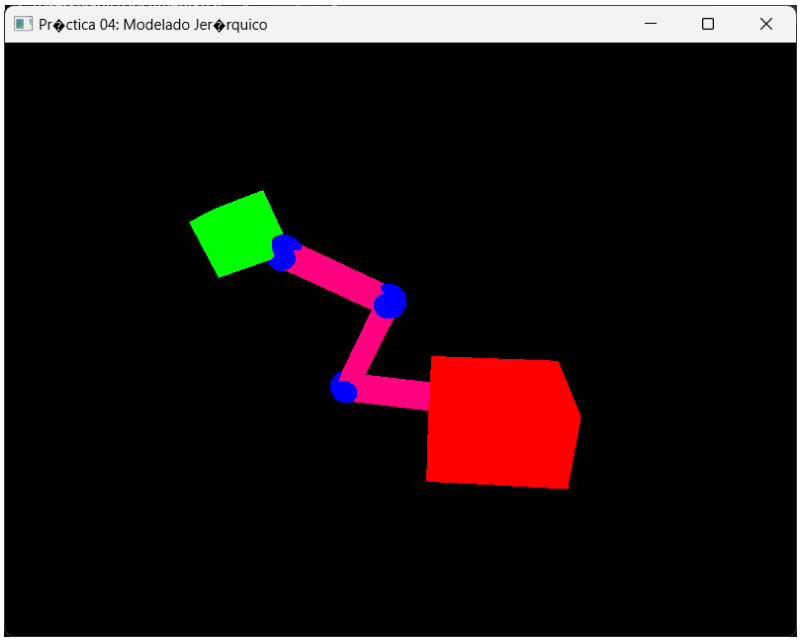

Fig. 8. *Ejecución del programa, movimiento de primera articulación.*

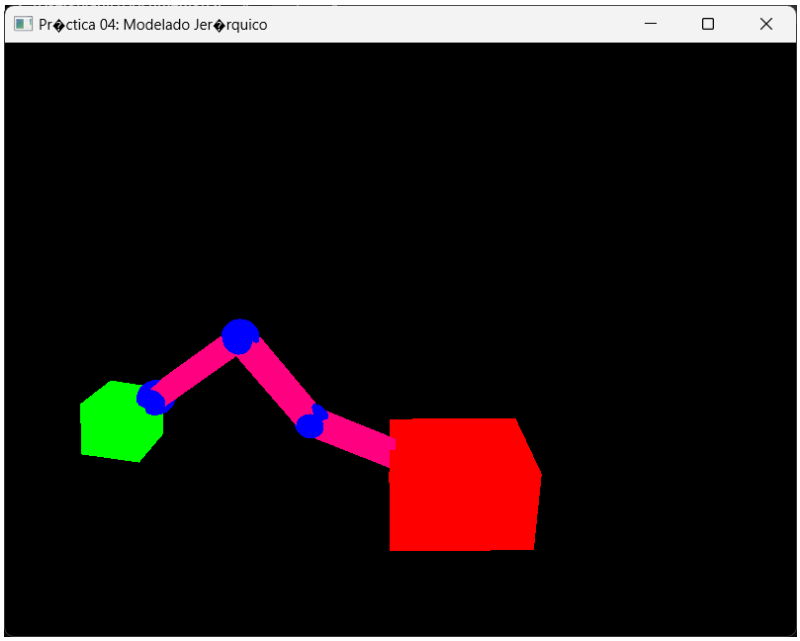

Fig. 9. *Ejecución del programa, movimiento de segunda articulación.*

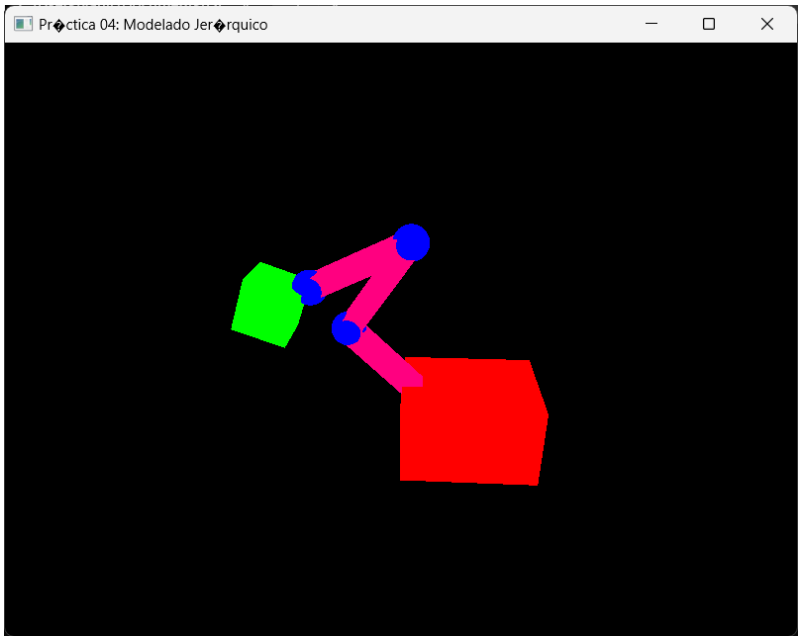

Fig. 10. *Ejecución del programa, movimiento de tercera articulación.*

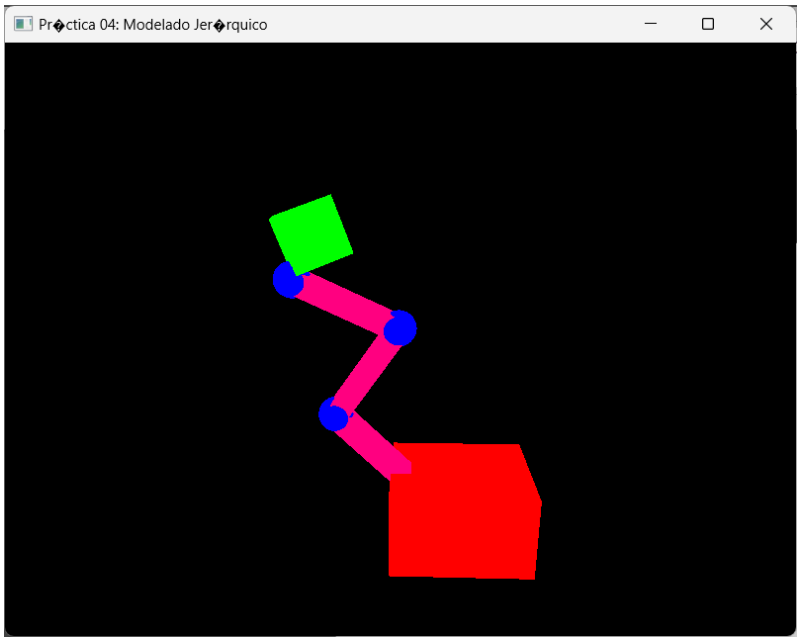

Fig. 11. *Ejecución del programa, movimiento de cuarta articulación.*

Si se requiere ver mejor la ejecución del ejercicio, agrego liga de un video como evidencia de la ejecución:
https://drive.google.com/drive/folders/1qJew2lPRRn5WXpT_ORjwkx7ZqHL11gLU?usp=sharing

2. *Problemas presentados.*

En general en esta actividad no presente problemas, lo complicado fue identificar la posición de la base y la cabina, la forma en que solucione estas complicaciones fue realizando pequeñas modificaciones en los ejes y las escalas de acuerdo con cómo los observaba al momento de ejecutarlo. También otra cosa complicada fue el sentido de giro para la cabina, pero una vez que analice la posición de los ejes que tenía la figura tras la herencia de las rotaciones de ángulo de las demás figuras, fue fácil definir los ejes sobre los que giraría, por lo demás no tuve otros problemas.

3. *Conclusión:*

Después de realizar el ejercicio propuesto, considero que la complejidad que tenía era normal, entendiendo las direcciones y las formas en que se definían las jerarquías de las figuras para su rotación y ángulos de inclinación, realizar los elementos faltantes del brazo era sencillo, considero que no falto explicar algún elemento de los ejercicios, todo se entendió, y fue un gran apoyo en casa los videos de explicación ya que en lo personal si suelo olvidar algunos pequeños detalles de las explicaciones de clase. Esta práctica a comparación de la anterior no fue tan complicada, los conceptos de jerarquía vistos en esta actividad me parecieron interesantes, más que nada porque lo veo como aplicación futura en el proyecto final para definir los movimientos a personajes, objetos o edificios. Al final considero que la practica tuvo un buen nivel.