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: Sanchez Villalpando Johan

Nº de Cuenta: 422028657

GRUPO DE LABORATORIO: 04

GRUPO DE TEORÍA: 02

SEMESTRE 2025-2

FECHA DE ENTREGA LÍMITE: 09/03/25

CALIFICACIÓN: _____

REPORTE DE PRÁCTICA:

1.- Ejecución de los ejercicios que se dejaron, comentar cada uno y capturas de pantalla de bloques de código generados y de ejecución del programa.

El primer ejercicio nos solicita terminar con la grúa realizada en clase, añadiendo la base y las llantas de la grúa, siguiendo también con lo hecho en el ejercicio de clase, donde se terminó la parte de arriba, la cual incluye la tercer articulación, la canasta y la base rectangular.

El procedimiento es el mismo, sin embargo algo a tomar en cuenta es el árbol jerárquico de la grúa el cual nos dice que tanto la base como las llantas son “aparte” del brazo articulado, por ello para el caso de esta sección decidí utilizar una nueva variable auxiliar para el modelo, la cual será llamada modeloaux2.

Primero comenzamos generando la base para las llantas, la cual se trata de un cono más pequeño en Y, despues de este realizamos el guardado del modelo en la variable auxiliar, esto para guardar la traslación, pero no la escalación.

De este modo continúa con la articulación de una llanta, para mi caso utilicé 4 articulaciones nuevas, una para cada llanta y una tecla distinta.

```
//Para base cono

color = glm::vec3(1.0f, 0.0f, 1.0f);  //magenta
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
model = glm::translate(model, glm::vec3(0.0f, 3.5f, -4.0f));
modelaux2 = model;
model = glm::scale(model, glm::vec3(3.0f, 2.0f, 3.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));//FALSE ES PARA QUE NO SEA TRANS
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
meshList[4]->RenderMeshGeometry(); //cubo cabina
model = modelaux2;

//articulación llanta atras izq.
model = glm::translate(model, glm::vec3(1.5f, -1.5f, 1.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion5()), glm::vec3(0.0f, 0.0f, 1.0f));
//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();
```

```
//Llanta Atras izq.
model = glm::translate(model, glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 0.2f));
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
sp.render();

model = modelaux2;
```

Este proceso se repite con las otras 3 llantas, cada una con su articulación y variando en traslación para acomodarlas correctamente.

```
//inicia segunda llanta:
color = glm::vec3(1.0f, 0.0f, 1.0f);  //magenta
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
//articulación llantas frente
model = glm::translate(model, glm::vec3(-1.5f, -1.5f, 1.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion6()), glm::vec3(0.0f, 0.0f, 1.0f));
//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();

//Llanta Atras izq.
model = glm::translate(model, glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 0.2f));
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
sp.render();
model = modelaux2;
```

```
  //articulación llantas frente
  model = glm::translate(model, glm::vec3(-1.5f, -1.5f,-1.0f));
  model = glm::rotate(model, glm::radians(mainWindow.getarticulacion7()), glm::vec3(0.0f, 0.0f, 1.0f));
  //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();

  //Lanta frente der.
  model = glm::translate(model, glm::vec3(0.0f, 0.0f, -1.0f));
  model = glm::scale(model, glm::vec3(1.0f, 1.0f, 0.2f));
  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
  sp.render();

  //LLanta trasera derecha
  model = modelaux2;
  color = glm::vec3(1.0f, 0.0f, 1.0f);  //magenta
  glUniform3fv(uniformColor, 1, glm::value_ptr(color));
  //articulación llanta
  model = glm::translate(model, glm::vec3(1.5f, -1.5f, -1.0f));
  model = glm::rotate(model, glm::radians(mainWindow.getarticulacion8()), glm::vec3(0.0f, 0.0f, 1.0f));
  //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();
```

Al final de la ultima llanta, se vuelve a llamar a la variable auxiliar original, de este modo los brazos articulados van a conservar los cambios ya realizados durante la práctica y el ejercicio.

```
//Lanta trasera der.
model = glm::translate(model, glm::vec3(0.0f, 0.0f, -1.0f));
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 0.2f));
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
sp.render();
model = modelaux;
```

Adicionalmente, para añadir las nuevas articulaciones, fue necesario declararlas en el window.h y window.cpp

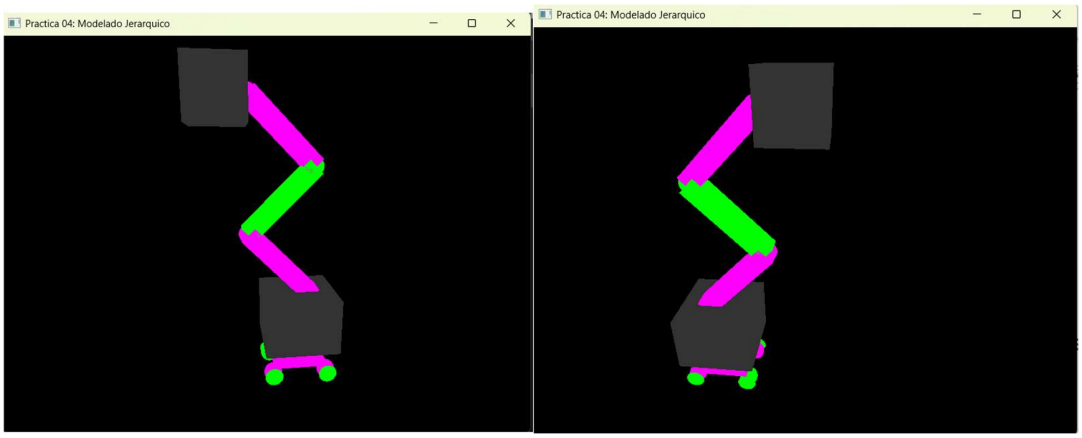
```
GLfloat getarticulacion7() { return articulacion7; }
GLfloat getarticulacion8() { return articulacion8; }
```

```
    articulacion7 = 0.0f;
    articulacion8 = 0.0f;
```

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

```
GLfloat rotax,rotay,rotaz, articulacion1, articulacion2, articulacion3, articulacion4, articulacion5, articulacion6, articulacion7, articulacion8;
```

Finalmente obtenemos el resultado deseado:



Para la rotación de las llantas es necesario acercarse bastante para apreciar el giro, sin embargo, se puede solucionar con llantas texturizadas o usando llantas más irregulares y no tan circulares (bajando la resolución de la esfera por ejemplo).

Para el **siguiente ejercicio** lo que se solicita es realizar una “mascota” con articulaciones y empleando las figuras vistas en clase, aplicando también las jerarquías.

En mi caso decidí crear un perro, compuesto de 4 patas, una cola, hocico, naríz, 2 orejas, y una cabeza.

Para la jerarquía, la parte principal es el cuerpo, compuesto por un cuadrado, del cual se desprenden 2 bloques, el bloque de las patas/cola y el bloque de la cabeza que incluye también hocico, nariz y orejas.

Para este ejercicio se repitió el uso del modelaux2, para facilitar el acomodo, empezando por el cuerpo, el cual es un cubo escalado y trasladado, luego esta se guarda en el modelaux, porque se usará como partida para la cabeza.

```
//Aquí inicia ejercicio 02 de la práctica

//Para el cuerpo
color = glm::vec3(0.66f, 0.55f, 0.42f);  //café claro
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
model = glm::translate(model, glm::vec3(0.0f, 0.0f, -4.0f));
modelaux = model;
model = glm::scale(model, glm::vec3(5.0f, 3.0f, 3.0f));
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 = modelaux;
```

Para las piernas ya se utiliza el modelaux2, en este caso reciclé lo usado para los brazos de la grúa, con unos cambios en escala.

```
//articulación pata delantera der
color = glm::vec3(0.3f, 0.17f, 0.007f);  //café oscuroooooooOO
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
model = glm::translate(model, glm::vec3(-1.5f, -2.0f, -1.0f));
//se mueve con f
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()), glm::vec3(0.0f, 0.0f, 1.0f));
//dibujar una pequeña esfera
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
modelaux2 = model; //guarda esfera
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
sp.render();
```

```
//Pata delantera derecha parte 1
model = glm::translate(model, glm::vec3(0.0f, -1.8f, 0.0f));
model = glm::scale(model, glm::vec3(1.0f, 2.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.66f, 0.55f, 0.42f);  //café claro
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[0]->RenderMesh(); //dibuja cubo y pirámide triangular

model = modelaux2;

//articulación pata delantera der parte 2
color = glm::vec3(0.3f, 0.17f, 0.007f);  //café oscuroooooooOO
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
model = glm::translate(model, glm::vec3(0.0f, -3.5f, 0.0f));
//se mueve con g
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()), glm::vec3(0.0f, 0.0f, 1.0f));
//dibujar una pequeña esfera
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
sp.render();
```

Finalmente se retoma el modelaux para iniciar con la siguiente pata, anclada al cuerpo.

```
//Pata delantera derecha parte 2
model = glm::translate(model, glm::vec3(0.0f, -1.8f, 0.0f));
model = glm::scale(model, glm::vec3(1.0f, 2.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.66f, 0.55f, 0.42f);  //café claro
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[0]->RenderMesh(); //dibuja cubo y pirámide triangular

model = modelaux;
```

Después de terminar las 4 patas, se vuelve al modelaux, para iniciar la cabeza, con un cuello hecho con un cilindro, luego la cabeza anclado sobre el cuello, con un cubo.

```
model = modelaux; //volvemos a nodo padre

//cilindro para  cuello
color = glm::vec3(0.3f, 0.17f, 0.007f);  //café oscuroooooooOO
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
model = glm::translate(model, glm::vec3(-2.0f, 1.712f, 0.0f));
modelaux2 = model; //no heredar esacla
model = glm::scale(model, glm::vec3(0.6f, 0.4f, 0.6f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[2]->RenderMeshGeometry(); //dibuja cubo y pirámide triangular
model = modelaux2;

//Cabeza5
color = glm::vec3(0.66f, 0.55f, 0.42f);  //café claro
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
model = glm::translate(model, glm::vec3(0.0f, 1.1f, 0.0f));
model = glm::scale(model, glm::vec3(2.0f, 2.0f, 2.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.66f, 0.55f, 0.42f);  //café claro
meshList[0]->RenderMesh(); //dibuja cubo y pirámide triangular
```

Para la trompa y la naríz se realiza lo mismo, modificando la escala y el traslado del cubo origina de la cabeza, para la trompa/hocico y la nariz.

```
//Trompa
color = glm::vec3(0.3f, 0.17f, 0.007f);  //café oscuroooooooOO
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
model = glm::translate(model, glm::vec3(-0.7f, -0.15f, 0.0f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[0]->RenderMesh(); //dibuja cubo y pirámide triangular

//Nariz
color = glm::vec3(0.0f, 0.0f, 0.0f);  //NEGRO
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
model = glm::translate(model, glm::vec3(-0.6f, 0.15f, 0.0f));
model = glm::scale(model, glm::vec3(0.15f, 0.3f, 0.3f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[0]->RenderMesh(); //dibuja cubo y pirámide triangular
model = modelaux2;
```

Se termina con las 2 orejas usando pirámides cuadrangulares y luego se regresa al cuerpo para usarlo como ancla de la cola.

```
//orejaizq
color = glm::vec3(0.3f, 0.17f, 0.007f);  //café oscuroooooooOO
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
model = glm::translate(model, glm::vec3(0.0f, 1.0f, 1.2f));
model = glm::scale(model, glm::vec3(0.8f, -1.5f, 0.4f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[4]->RenderMesh(); //dibuja cubo y pirámide triangular

//orejader
model = glm::translate(model, glm::vec3(0.0f, 0.0f, -6.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[4]->RenderMesh(); //dibuja cubo y pirámide triangular

model = modelaux;
```

Inicia la cola, la cual se crea igual que una pata, con la diferencia de que la segunda parte de la cola, es un cono, para dar el efecto puntiagudo.

```
//COLA
//articulación cola
color = glm::vec3(0.3f, 0.17f, 0.007f);  //café oscuroooooooOO
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
model = glm::translate(model, glm::vec3(2.8f, 0.0f, 0.0f));
//se mueve con c
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion9()), glm::vec3(0.0f, 0.0f, 1.0f));
//dibujar una pequeña esfera
model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.3f));
modelaux2 = model; //guarda esfera
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
sp.render();

//cola parte 1
model = glm::translate(model, glm::vec3(0.0f, -1.8f, 0.0f));
model = glm::scale(model, glm::vec3(1.0f, 2.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.66f, 0.55f, 0.42f);  //café claro
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[0]->RenderMesh(); //dibuja cubo y pirámide triangular
model = modelaux2;

//cola articulación 2
color = glm::vec3(0.3f, 0.17f, 0.007f);  //café oscuroooooooOO
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
model = glm::translate(model, glm::vec3(0.0f, -3.5f, 0.0f));
//se mueve con v
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion10()), glm::vec3(0.0f, 0.0f, 1.0f));
//dibujar una pequeña esfera
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
sp.render();

//Cola parte final
model = glm::translate(model, glm::vec3(0.0f, -1.8f, 0.0f));
model = glm::scale(model, glm::vec3(0.3f, -4.0f, 0.3f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.66f, 0.55f, 0.42f);  //café claro
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[3]->RenderMeshGeometry(); //dibuja cubo y pirámide triangular
```

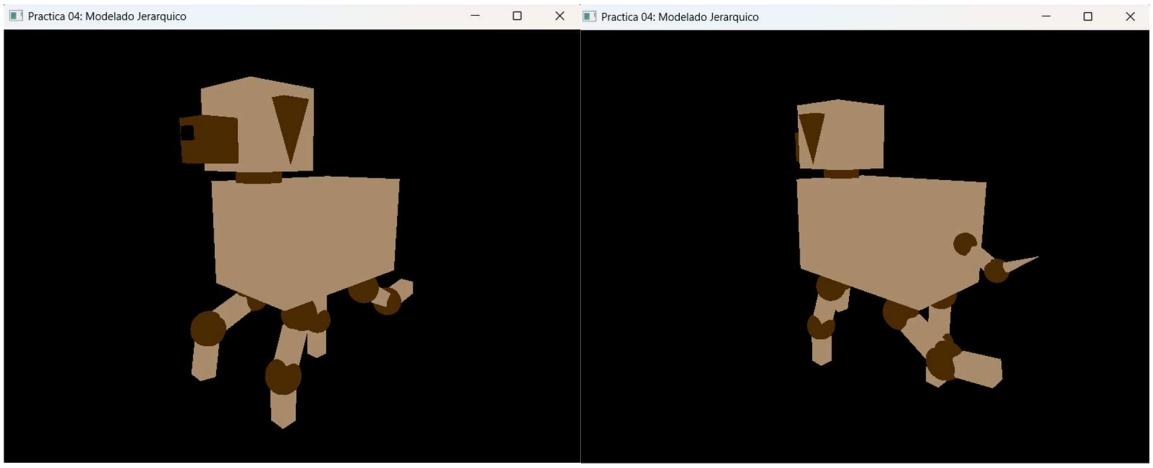
Adicionalmente se añaden las opciones para las teclas extra, ya que no alcanza con las que estaban.

```
if (key == GLFW_KEY_Z)
{
    theWindow->articulacion7 += 10.0;
}
if (key == GLFW_KEY_X)
{
    theWindow->articulacion8 += 10.0;
}
if (key == GLFW_KEY_C)
{
    theWindow->articulacion9 += 10.0;
}
if (key == GLFW_KEY_V)
{
    theWindow->articulacion10 += 10.0;
}
```

```
articulacion7 = 0.0f;
articulacion8 = 0.0f;
articulacion9 = 0.0f;
articulacion10 = 0.0f;
```

```
GLfloat getarticulacion7() { return articulacion7; }
GLfloat getarticulacion8() { return articulacion8; }
GLfloat getarticulacion9() { return articulacion9; }
GLfloat getarticulacion10() { return articulacion10; }
```

Finalmente tenemos el resultado esperado.



2.- Liste los problemas que tuvo a la hora de hacer estos ejercicios y si los resolvió explicar cómo fue, en caso de error adjuntar captura de pantalla.

Para el ejercicio 1, el principal problema fue en comprender las jerarquías y el uso de variables auxiliares para heredar transformaciones, una vez entendido esto, fue más sencillo ir generando los elementos.

Para el caso del ejercicio 2 fue similar, una vez entendida la jerarquía fue más trabajo de creatividad para utilizar las figuras más variadas.

3.- Conclusión:

a. Los ejercicios del reporte: Complejidad, Explicación.

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

Como conclusión esta práctica me pareció entretenida, al principio me perdí en el laboratorio y la sentí algo compleja, pero una vez entendí las jerarquías y las herencias, fue todo más sencillo, me pareció interesante el concepto de mover las partes y utilizar anclas o articulaciones, me hace entender mejor también el funcionamiento de los personajes en los videojuegos y lo que vamos a trabajar en el proyecto.

La dificultad me pareció buena, aunque un poco confuso al principio el uso de los auxiliares y las jerarquías.

Bibliografía en formato APA

- *LOsorno, C. (2016, abril 18). JERARQUÍA DE OBJETOS EN OPENGL. Blogspot.com. Recuperado el 08 de Marzo, de https://openglgraficacion.blogspot.com/2016/04/jerarquia-de-objetos-en-opengl.htm*
- *Tutorial 3 : Matrices*. (s/f). Opengl-tutorial.org. Recuperado el 26 de febrero de 2025, de https://www.opengl-tutorial.org/es/beginners-tutorials/tutorial-3-matrices/