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: Cadena Luna Iván Adrián

Nº de Cuenta: 318304188

GRUPO DE LABORATORIO: 01

GRUPO DE TEORÍA: 04

SEMESTRE 2025-1

FECHA DE ENTREGA LÍMITE: sábado 7 de septiembre de 2024

CALIFICACIÓN: _____

REPORTE DE PRÁCTICA 04 - Modelado Jerárquico.

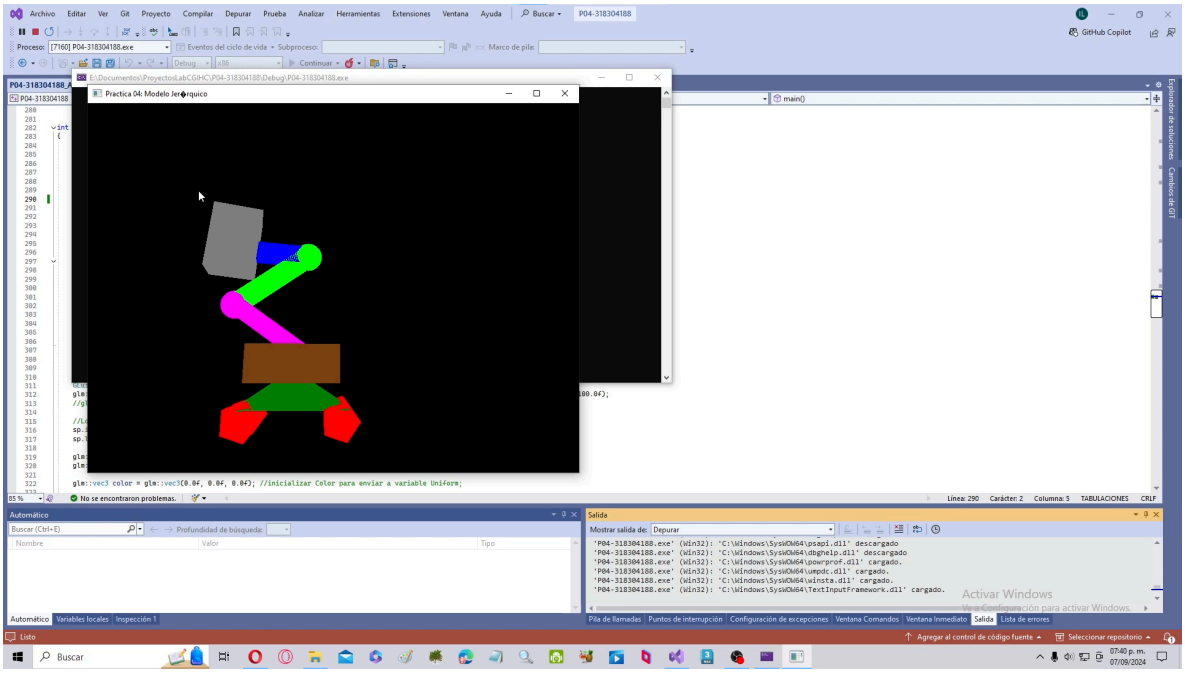
I. Ejecución de los ejercicios que se dejaron, comentar cada uno y capturas de pantalla de bloques de código generados y de ejecución del programa.

Los dos ejercicios se muestran de forma simultánea y están en el mismo main.

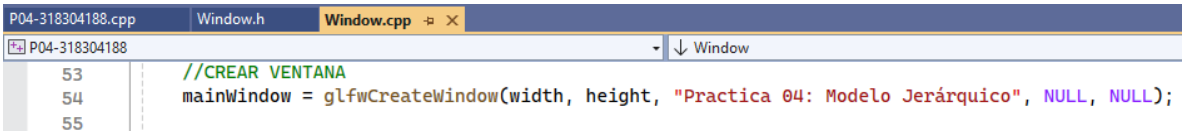
A. Terminar la Grúa con:

- **Cuerpo (prisma rectangular).**
- **Base (pirámide cuadrangular).**
- **4 llanta(4 cilindros) (con teclado se pueden girar las 4 llantas por separado).**

https://youtu.be/Q1vNhBSOUEo



Para comenzar este primer ejercicio de la práctica lo primero que hice fue modificar el título de despliegue de la ventana desde el archivo *Window.cpp*.



Lo siguiente, teniendo bien presente que debería agregar más articulaciones para la parte de las llantas, tal y como se nos sugirió en la sesión, me moví al archivo *Window.h* y procedí a agregar las articulaciones necesarias para cada una de ellas. En este caso al ser 4 se agregaron en orden numérico siguiendo la secuencia previa.

```
P04-318304188.cpp | Window.h | Window.cpp
P04-318304188                                   Window
23   GLfloat getarticulacion1() { return articulacion1; }
24   GLfloat getarticulacion2() { return articulacion2; }
25   GLfloat getarticulacion3() { return articulacion3; }
26   GLfloat getarticulacion4() { return articulacion4; }
27   GLfloat getarticulacion5() { return articulacion5; }
28   GLfloat getarticulacion6() { return articulacion6; }
29   ////////adicionales para las llantas
30   GLfloat getarticulacion7() { return articulacion7; }
31   GLfloat getarticulacion8() { return articulacion8; }
32   GLfloat getarticulacion9() { return articulacion9; }
33   GLfloat getarticulacion10() { return articulacion10; }
```

En el mismo archivo también fue necesario establecer la rotación de dichas articulaciones, haciendo referencia a la función `GLfloat getrotay()`, `GLfloat getrotaz()` y `GLfloat getrotax()`.

```
P04-318304188.cpp | Window.h | Window.cpp
P04-318304188                                   Window                                   ManejaTeclad
39   GLfloat rotax,rotay,rotaz, articulacion1, articulacion2, articulacion3, articulacion4, articulacion5, articulacion6,
```

```
P04-318304188.cpp | Window.h | Window.cpp
P04-318304188                                   Window                                   ManejaTeclado(GLFWwindow * window, int key, int
39   , articulacion4, articulacion5, articulacion6, articulacion7, articulacion8, articulacion9, articulacion10;////////adicionales para las llantas
```

Completado ese punto, pasé nuevamente al archivo *Window.cpp* y era turno de implementar las articulaciones y sus teclas correspondientes para que fuera posible su rotación.

Elegí las teclas V, B, N y M para la rotación de cada llanta.

```
P04-318304188.cpp | Window.h | Window.cpp
P04-318304188                                   Win
13   Window::Window(GLint windowWidth, GLint windowHeight)
14   {
15       width = windowWidth;
16       height = windowHeight;
17       rotax = 0.0f;
18       rotay = 0.0f;
19       rotaz = 0.0f;
20       articulacion1 = 0.0f;
21       articulacion2 = 0.0f;
22       articulacion3 = 0.0f;
23       articulacion4 = 0.0f;
24       articulacion5 = 0.0f;
25       articulacion6 = 0.0f;
26       //adicionales para las llantas
27       articulacion7 = 0.0f;
28       articulacion8 = 0.0f;
29       articulacion9 = 0.0f;
30       articulacion10 = 0.0f;
```

P04-318304188.cpp | Window.h | Window.cpp

P04-318304188

```
159     //if adicionales para las llantas
160     if (key == GLFW_KEY_V)
161     {
162         theWindow->articulacion7 += 10.0;
163     }
164     if (key == GLFW_KEY_B)
165     {
166         theWindow->articulacion8 += 10.0;
167     }
168     if (key == GLFW_KEY_N)
169     {
170         theWindow->articulacion9 += 10.0;
171     }
172     if (key == GLFW_KEY_M)
173     {
174         theWindow->articulacion10 += 10.0;
175     }
176
```

Completado lo anterior, era momento de enfocarse en el archivo main.

- **Caso base**

```
355         //AQUÍ SE DIBUJA LA CABINA, LA BASE, LAS 4 LLANTAS
356
357         //'base'
358         model = glm::translate(model, glm::vec3(0.0f, 0.0f, -4.0f));
359         modelaux = model;
360         model = glm::scale(model, glm::vec3(6.0f, 2.0f, 2.5f));
361         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
362         color = glm::vec3(0.0f, 0.5f, 0.0f);
363         glUniform3fv(uniformColor, 1, glm::value_ptr(color));
364         meshList[4]->RenderMesh();
```

- **Caso primer llanta**

```
366         //'primer llanta'
367         model = modelaux;
368         model = glm::translate(model, glm::vec3(2.5f, -1.5f, -1.0f));
369         modelaux = model;
370         model = glm::rotate(model, glm::radians(mainWindow.getarticulacion7()), glm::vec3(0.0f, 0.0f, 1.0f));
371         model = glm::rotate(model, glm::radians(90.0f), glm::vec3(1.0f, 0.0f, 0.0f));
372         model = glm::scale(model, glm::vec3(1.0f, 0.94f, 1.0f));
373         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
374         color = glm::vec3(1.0f, 0.0f, 0.0f);
375         glUniform3fv(uniformColor, 1, glm::value_ptr(color));
376         meshList[2]->RenderMeshGeometry();
```

- **Caso segunda llanta**

```
378         //'segunda llanta'
379         model = modelaux;
380         model = glm::translate(model, glm::vec3(-5.0f, 0.0f, 0.0f));
381         modelaux = model;
382         model = glm::rotate(model, glm::radians(mainWindow.getarticulacion8()), glm::vec3(0.0f, 0.0f, 1.0f));
383         model = glm::rotate(model, glm::radians(90.0f), glm::vec3(1.0f, 0.0f, 0.0f));
384         model = glm::scale(model, glm::vec3(1.0f, 0.94f, 1.0f));
385         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
386         color = glm::vec3(1.0f, 0.0f, 0.0f);
387         glUniform3fv(uniformColor, 1, glm::value_ptr(color));
388         meshList[2]->RenderMeshGeometry();
```

- **Caso tercera llanta**

```
390         //'tercer llanta'
391         model = modelaux;
392         model = glm::translate(model, glm::vec3(0.0f, 0.0f, 2.0f));
393         modelaux = model;
394         model = glm::rotate(model, glm::radians(mainWindow.getarticulacion9()), glm::vec3(0.0f, 0.0f, 1.0f));
395         model = glm::rotate(model, glm::radians(90.0f), glm::vec3(1.0f, 0.0f, 0.0f));
396         model = glm::scale(model, glm::vec3(1.0f, 0.94f, 1.0f));
397         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
398         color = glm::vec3(1.0f, 0.0f, 0.0f);
399         glUniform3fv(uniformColor, 1, glm::value_ptr(color));
400         meshList[2]->RenderMeshGeometry();
```

- **Caso cuarta llanta**

```
402         //'cuarta llanta'
403         model = modelaux;
404         model = glm::translate(model, glm::vec3(5.0f, 0.0f, 0.0f));
405         modelaux = model;
406         model = glm::rotate(model, glm::radians(mainWindow.getarticulacion10()), glm::vec3(0.0f, 0.0f, 1.0f));
407         model = glm::rotate(model, glm::radians(90.0f), glm::vec3(1.0f, 0.0f, 0.0f));
408         model = glm::scale(model, glm::vec3(1.0f, 0.94f, 1.0f));
409         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
410         color = glm::vec3(1.0f, 0.0f, 0.0f);
411         glUniform3fv(uniformColor, 1, glm::value_ptr(color));
412         meshList[2]->RenderMeshGeometry();
```

*Para este caso de las llantas es importante mencionar que conservé la declaración de sus “lados” en 5, de tal manera que en la ejecución se observan como pentágonos; decidí dejarlo así para que se pudiera apreciar mejor la rotación.

```
282 int main()
283 {
284     mainWindow = Window(800, 600);
285     mainWindow.Initialise();
286     //Cilindro y cono reciben resolución (slices, rebanadas) y Radio de circunferencia de la base y tapa
287 
288     CrearCubo();//índice 0 en MeshList
289     CrearPiramideTriangular();//índice 1 en MeshList
290     CrearCilindro(5, 1.0f);//índice 2 en MeshList  //SE VE COMO PENTÁGONO PORQUE ESTÁ EN 5 (ESQUINAS)
291     CrearCono(25, 2.0f);//índice 3 en MeshList
292     CrearPiramideCuadrangular();//índice 4 en MeshList
293     CreateShaders();
```

- **Caso cabina/cuerpo**

```
414         //'cabina/cuerpo'
415         model = modelaux;
416         model = glm::translate(model, glm::vec3(-2.5f, 2.8f, -1.0f));
417         modelaux = model;
418         model = glm::scale(model, glm::vec3(5.0f, 2.0f, 2.0f));
419         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
420         color = glm::vec3(0.478f, 0.255f, 0.067f);
421         glUniform3fv(uniformColor, 1, glm::value_ptr(color));
422         meshList[0]->RenderMesh();
```

- **Caso tercer brazo (recuperado del código *E04-318304188.cpp*)**

```
497             //para descartar la escala que no quiero heredar se carga la información de la matrix auxiliar
498             model = modelaux;
499 
500         //  -   //'tercer brazo' (azul)
501             model = glm::translate(model, glm::vec3(2.5f, 0.0f, 0.0f));
502             //para descartar la escala que no quiero heredar se carga la información de la matrix auxiliar
503             modelaux = model;
504             model = glm::scale(model, glm::vec3(5.0f, 1.0f, 1.0f));
505             glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
506             color = glm::vec3(0.0f, 0.0f, 1.0f);
507             glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
508             meshList[0]->RenderMesh(); //dibuja cubo y pirámide triangular
```

- **Caso articulación4 (recuperado del código *E04-318304188.cpp*)**

```
513 //  -   //"articulación4" extremo izquierdo del tercer brazo (tercer brazo a canasta)
514             model = glm::translate(model, glm::vec3(2.5f, 0.0f, 0.0f));
515             model = glm::rotate(model, glm::radians(mainWindow.getarticulacion4()), glm::vec3(0.0f, 1.0f, 0.0f)); //LÍNEA - PRUEBA PARA QUE LA CANASTA GIRE BIEN
516             //para descartar la escala que no quiero heredar se carga la información de la matrix auxiliar
517             modelaux = model;
518 
519             //dibujar una pequeña esfera -> conexión entre 'tercer brazo' y 'canasta trabajador'
520             model = glm::scale(model, glm::vec3(0.8f, 0.8f, 0.8f));
521             glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
522             sp.render();
```

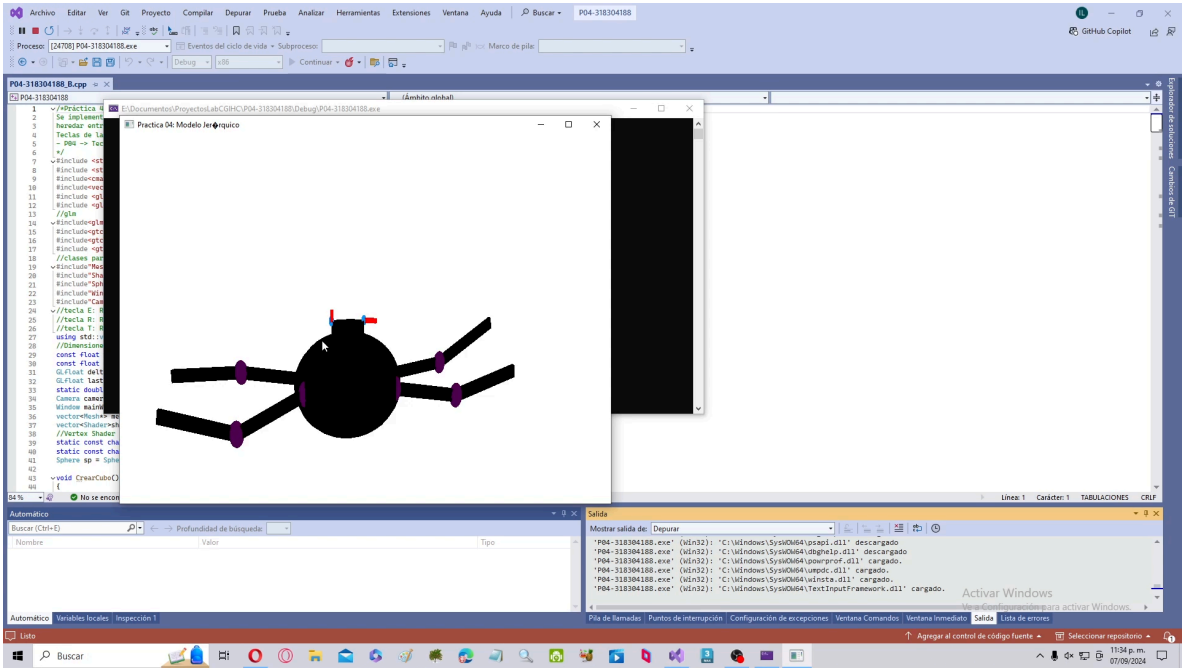
- **Caso canasta (recuperado del código *E04-318304188.cpp*)**

```
524             //para descartar la escala que no quiero heredar se carga la información de la matrix auxiliar
525             model = modelaux;
526 
527         //  -   //'canasta'
528             model = glm::translate(model, glm::vec3(0.575f, -0.5f, 0.0f));
529             //para descartar la escala que no quiero heredar se carga la información de la matrix auxiliar
530             modelaux = model;
531             model = glm::scale(model, glm::vec3(2.75f, 3.75f, 2.25f));
532             glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
533             color = glm::vec3(0.5f, 0.5f, 0.5f);
534             glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
535             meshList[0]->RenderMesh(); //dibuja cubo y pirámide triangular
```

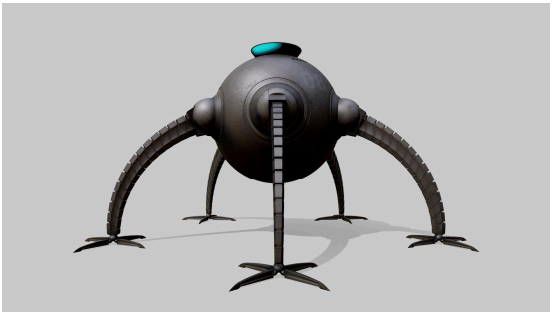
B. Crear un animal robot 3d:

- **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 (con teclado se puede mover la cola o cada oreja independiente).**

https://youtu.be/HhNxXuMpbXU



Para este ejercicio decidí basarme en las siguientes imágenes de referencia, siendo la segunda una versión de los Omnidroid de la película de *Los Increíbles*.



En este caso, utilicé la formación de las armas del robot como si fueran sus orejas. Aproveché para darle a cada una una rotación diferente, de tal manera que una puede moverse de atrás para adelante para atacar enfrente y atrás y a la otra puede moverse a los lados para abarcar el panorama planar.

Declaré el cuerpo como una esfera y utilicé una referencia dentro del mismo cuerpo del robot (a modo de orientación y para hacer también una relación con donde Mr. Increíble se escondía).

Siguiendo el modelo del árbol jerárquico fue relativamente sencilla la parte de estructurar cómo ir modelando cada parte del robot.

- **Caso cuerpo esférico**

```
541         /*******************************************************************************/
542 //  EJERCIO B - Crear un animal robot 3D
543 
544         //'cuerpo esférico'
545         model = glm::mat4(1.0);
546         model = glm::translate(model, glm::vec3(0.0f, 0.0f, -4.0f));
547         model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f));
548         modelaux2 = model;
549         modelaux = model;
550         color = glm::vec3(0.0f, 0.0f, 0.0f);
551         model = glm::scale(model, glm::vec3(5.0f, 5.0f, 4.0f));
552         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
553         glUniform3fv(uniformColor, 1, glm::value_ptr(color));
554         sp.render();
```

- **Caso cuerpo referencia**

```
556         //'cuerpo referencia'
557         model = glm::mat4(1.0);
558         model = glm::translate(model, glm::vec3(0.0f, 0.0f, -4.0f));
559         model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f));
560         modelaux2 = modelaux = model;
561         model = glm::scale(model, glm::vec3(3.0f, 6.0f, 2.0f));
562         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
563         glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
564         glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
565         color = glm::vec3(0.0f, 0.0f, 0.0f);
566         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
567         meshList[2]->RenderMeshGeometry(); //dibuja cubo, pirámide triangular, pirámide base cuadrangular
568         modelaux2 = modelaux = model;
```

- **Caso articulación1**

```
570         //"articulacion1" del cuerpo al primer brazo
571         model = glm::translate(model, glm::vec3(1.32f, 0.0f, 1.0f));
572         model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()), glm::vec3(0.0f, 0.0f, 1.0f));
573         modelaux = model;
574         //dibujar una pequeña esfera -> efecto conexión entre cuerpo y primer brazo
575         model = glm::scale(model, glm::vec3(0.2f, 0.2f, 0.2f));
576         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
577         color = glm::vec3(0.31f, 0.0f, 0.31f);
578         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
579         sp.render();
```

- **Caso primer brazo**

```
581         //'primer brazo'
582         model = glm::translate(model, glm::vec3(3.5f, 3.5f, 0.0f));
583         model = glm::rotate(model, glm::radians(-135.0f), glm::vec3(0.0f, 0.0f, 1.0f));
584         modelaux = model;
585         model = glm::scale(model, glm::vec3(10.0f, 1.0f, 1.0f));
586         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
587         //la línea de proyección solo se manda una vez a menos que en tiempo de ejecución
588         //se programe cambio entre proyección ortogonal y perspectiva
589         //glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
590         glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
591         color = glm::vec3(0.0f, 0.0f, 0.0f);
592         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
593         meshList[0]->RenderMesh();
```

- **Caso articulación2**

```
595         //para descartar la escala que no quiero heredar se carga la información de la matrix auxiliar
596         model = modelaux;
597
598         //"articulación2" del primer brazo al primer antebrazo
599         model = glm::translate(model, glm::vec3(-5.0f, 0.0f, 0.0f));
600         model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()), glm::vec3(0.0f, 0.0f, 1.0f));
601         modelaux = model;
602         //dibujar una pequeña esfera -> efecto conexión entre primer brazo y primer antebrazo
603         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
604         color = glm::vec3(0.31f, 0.0f, 0.31f);
605         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
606         sp.render();
```

- **Caso primer antebrazo**

```
608          //'primer antebrazo'
609          model = glm::translate(model, glm::vec3(3.5f, 3.5f, 0.0f));
610          model = glm::rotate(model, glm::radians(-135.0f), glm::vec3(0.0f, 0.0f, 1.0f));
611          modelaux = model;
612          model = glm::scale(model, glm::vec3(10.0f, 1.0f, 1.0f));
613          glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
614          //la línea de proyección solo se manda una vez a menos que en tiempo de ejecución
615          //se programe cambio entre proyección ortogonal y perspectiva
616          glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
617          color = glm::vec3(0.0f, 0.0f, 0.0f);
618          glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
619          meshList[0]->RenderMesh();
```

- **Caso articulación3**

```
621         //para descartar la escala que no quiero heredar se carga la información de la matrix auxiliar
622         model = modelaux2;
623
624         //"articulación3" del cuerpo al segundo brazo
625         model = glm::translate(model, glm::vec3(-1.32f, 0.0f, 1.0f));
626         model = glm::rotate(model, glm::radians(mainWindow.getarticulacion3()), glm::vec3(0.0f, 0.0f, 1.0f));
627         modelaux = model;
628         //dibujar una pequeña esfera -> efecto conexión entre cuerpo y segundo brazo
629         model = glm::scale(model, glm::vec3(0.2f, 0.2f, 0.2f));
630         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
631         color = glm::vec3(0.31f, 0.0f, 0.31f);
632         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
633         sp.render();
```

- **Caso segundo brazo**

```
635          //'segundo brazo'
636          model = glm::translate(model, glm::vec3(-3.5f, 3.5f, 0.0f));
637          model = glm::rotate(model, glm::radians(135.0f), glm::vec3(0.0f, 0.0f, 1.0f));
638          modelaux = model;
639          model = glm::scale(model, glm::vec3(10.0f, 1.0f, 1.0f));
640          glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
641          //la línea de proyección solo se manda una vez a menos que en tiempo de ejecución
642          //se programe cambio entre proyección ortogonal y perspectiva
643          glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
644          color = glm::vec3(0.0f, 0.0f, 0.0f);
645          glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
646          meshList[0]->RenderMesh();
```

- **Caso articulación4**

```
//para descartar la escala que no quiero heredar se carga la información de la matrix auxiliar
model = modelaux;

//"articulación4" del segundo brazo al segundo antebrazo
model = glm::translate(model, glm::vec3(5.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion4()), glm::vec3(0.0f, 0.0f, 1.0f));
modelaux = model;
//dibujar una pequeña esfera -> efecto conexión entre segundo brazo y segundo antebrazo
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.31f, 0.0f, 0.31f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
sp.render();
```

- **Caso segundo antebrazo**

```
//'segundo antebrazo'
model = glm::translate(model, glm::vec3(-3.5f, 3.5f, 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(10.0f, 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(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(0.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[0]->RenderMesh();
```

- **Caso articulación5**

```
//para descartar la escala que no quiero heredar se carga la información de la matrix auxiliar
model = modelaux2;

//"articulación5" del cuerpo al tercer brazo
model = glm::translate(model, glm::vec3(1.32f, 0.0f, -1.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion5()), glm::vec3(0.0f, 0.0f, 1.0f));
modelaux = model;
//dibujar una pequeña esfera -> efecto conexión entre cuerpo y tercer brazo
model = glm::scale(model, glm::vec3(0.2f, 0.2f, 0.2f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.31f, 0.0f, 0.31f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
sp.render();
```

- **Caso tercer brazo**

```
//'tercer brazo'
model = glm::translate(model, glm::vec3(3.5f, 3.5f, 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(10.0f, 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(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(0.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[0]->RenderMesh();
```

- **Caso articulación6**

```
701         //para descartar la escala que no quiero heredar se carga la información de la matrix auxiliar
702         model = modelaux;
703
704         //"articulación6" del tercer brazo al tercer antebrazo
705         model = glm::translate(model, glm::vec3(-5.0f, 0.0f, 0.0f));
706         model = glm::rotate(model, glm::radians(mainWindow.getarticulacion6()), glm::vec3(0.0f, 0.0f, 1.0f));
707         modelaux = model;
708         //dibujar una pequeña esfera -> efecto conexión entre tercer brazo y tercer antebrazo
709         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
710         color = glm::vec3(0.31f, 0.0f, 0.31f);
711         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
712         sp.render();
```

- **Caso tercer antebrazo**

```
714         //'tercer antebrazo'
715         model = glm::translate(model, glm::vec3(3.5f, 3.5f, 0.0f));
716         model = glm::rotate(model, glm::radians(-135.0f), glm::vec3(0.0f, 0.0f, 1.0f));
717         modelaux = model;
718         model = glm::scale(model, glm::vec3(10.0f, 1.0f, 1.0f));
719         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
720         //la línea de proyección solo se manda una vez a menos que en tiempo de ejecución
721         //se programe cambio entre proyección ortogonal y perspectiva
722         glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
723         color = glm::vec3(0.0f, 0.0f, 0.0f);
724         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
725         meshList[0]->RenderMesh();
```

- **Caso articulación7**

```
727         //para descartar la escala que no quiero heredar se carga la información de la matrix auxiliar
728         model = modelaux2;
729
730         //"articulación7" del cuerpo al cuarto brazo
731         model = glm::translate(model, glm::vec3(-1.32f, 0.0f, -1.0f));
732         model = glm::rotate(model, glm::radians(mainWindow.getarticulacion7()), glm::vec3(0.0f, 0.0f, 1.0f));
733         modelaux = model;
734         //dibujar una pequeña esfera -> efecto conexión entre cuerpo y segundo brazo
735         model = glm::scale(model, glm::vec3(0.2f, 0.2f, 0.2f));
736         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
737         color = glm::vec3(0.31f, 0.0f, 0.31f);
738         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
739         sp.render();
```

- **Caso cuarto brazo**

```
741         //'cuarto brazo'
742         model = glm::translate(model, glm::vec3(-3.5f, 3.5f, 0.0f));
743         model = glm::rotate(model, glm::radians(135.0f), glm::vec3(0.0f, 0.0f, 1.0f));
744         modelaux = model;
745         model = glm::scale(model, glm::vec3(10.0f, 1.0f, 1.0f));
746         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
747         //la línea de proyección solo se manda una vez a menos que en tiempo de ejecución
748         //se programe cambio entre proyección ortogonal y perspectiva
749         glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
750         color = glm::vec3(0.0f, 0.0f, 0.0f);
751         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
752         meshList[0]->RenderMesh();
```

- **Caso articulación8**

```
754         //para descartar la escala que no quiero heredar se carga la información de la matrix auxiliar
755         model = modelaux;
756
757         //"articulación8" del cuarto brazo al cuarto antebrazo
758         model = glm::translate(model, glm::vec3(5.0f, 0.0f, 0.0f));
759         model = glm::rotate(model, glm::radians(mainWindow.getarticulacion8()), glm::vec3(0.0f, 0.0f, 1.0f));
760         modelaux = model;
761         //dibujar una pequeña esfera -> efecto conexión entre cuarto brazo y cuarto antebrazo
762         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
763         color = glm::vec3(0.31f, 0.0f, 0.31f);
764         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
765         sp.render();
```

- **Caso cuarto antebrazo**

```
767         //'cuarto antebrazo'
768         model = glm::translate(model, glm::vec3(-3.5f, 3.5f, 0.0f));
769         model = glm::rotate(model, glm::radians(135.0f), glm::vec3(0.0f, 0.0f, 1.0f));
770         modelaux = model;
771         model = glm::scale(model, glm::vec3(10.0f, 1.0f, 1.0f));
772         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
773         //la línea de proyección solo se manda una vez a menos que en tiempo de ejecución
774         //se programe cambio entre proyección ortogonal y perspectiva
775         glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
776         color = glm::vec3(0.0f, 0.0f, 0.0f);
777         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
778         meshList[0]->RenderMesh();
```

- **Caso cabeza**

```
780         //para descartar la escala que no quiero heredar se carga la información de la matrix auxiliar
781         model = modelaux2;
782
783         //'cabeza'
784         model = glm::translate(model, glm::vec3(0.0f, 0.5f, 0.0f));
785         modelaux3 = model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f));
786         model = glm::scale(model, glm::vec3(0.5f, 1.0f, 0.5f));
787         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
788         glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
789         color = glm::vec3(0.0f, 0.0f, 0.0f);
790         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
791         meshList[2]->RenderMeshGeometry(); //dibuja cubo, pirámide triangular, pirámide base cuadrangular
```

- **Caso articulación9**

```
793         //para descartar la escala que no quiero heredar se carga la información de la matrix auxiliar
794         model = modelaux3;
795
796         //"articulación9" de la cabeza a la primer arma
797         model = glm::translate(model, glm::vec3(0.5f, 0.5f, 0.0f));
798         model = glm::rotate(model, glm::radians(mainWindow.getarticulacion9() - 180), glm::vec3(0.0f, 1.0f, 0.0f));
799         modelaux = model;
800         //dibujar una pequeña esfera -> efecto conexión entre la cabeza y la primer arma
801         model = glm::scale(model, glm::vec3(0.08f, 0.08f, 0.08f));
802         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
803         color = glm::vec3(0.0f, 0.588f, 1.0f);
804         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
805         sp.render();
```

- **Caso primer arma**

```
807         //para descartar la escala que no quiero heredar se carga la información de la matrix auxiliar
808         model = modelaux;
809
810         //'primer arma'
811         model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.2f));
812         model = glm::scale(model, glm::vec3(0.08f, 0.08f, 0.5f));
813         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
814         glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
815         color = glm::vec3(1.0f, 0.0f, 0.0f);
816         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
817         meshList[0]->RenderMesh(); //dibuja cubo, pirámide triangular, pirámide base cuadrangular
```

- **Caso articulación10**

```
819         //para descartar la escala que no quiero heredar se carga la información de la matrix auxiliar
820         model = modelaux3;
821
822         //"articulación10" de la cabeza a la segunda arma
823         model = glm::translate(model, glm::vec3(-0.5f, 0.5f, 0.0f));
824         model = glm::rotate(model, glm::radians(mainWindow.getarticulacion10() - 180), glm::vec3(1.0f, 0.0f, 0.0f));
825         modelaux = model;
826         //dibujar una pequeña esfera -> efecto conexión entre la cabeza y la segunda arma
827         model = glm::scale(model, glm::vec3(0.08f, 0.08f, 0.08f));
828         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
829         color = glm::vec3(0.0f, 0.588f, 1.0f);
830         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
831         sp.render();
```

- **Caso segunda arma**

```
833         //para descartar la escala que no quiero heredar se carga la información de la matrix auxiliar
834         model = modelaux;
835
836         //'segunda arma'
837         model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.2f));
838         model = glm::scale(model, glm::vec3(0.08f, 0.08f, 0.5f));
839         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
840         glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
841         color = glm::vec3(1.0f, 0.0f, 0.0f);
842         glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
843         meshList[0]->RenderMesh(); //dibuja cubo, pirámide triangular, pirámide base cuadrangular
844
845 //FIN DEL EJERCICIO B - Crear un animal robot 3D*/
846         /*******************************************************************************/
```

Cabe mencionar también la creación de modelaux2 y modelaux3 para almacenar las transformaciones jerárquicas, las cuales intervinieron para transformar una parte de los objetos y otras para otras partes en los diferentes casos del robot.

```
320     glm::mat4 model(1.0);//Inicializar matriz de Modelo 4x4
321     glm::mat4 modelaux(1.0);//Inicializar matriz de Modelo 4x4 auxiliar para la jerarquía
322     glm::mat4 modelaux2(1.0);//Inicializar matriz de Modelo 4x4 auxiliar para la jerarquía (Caso Ejercicio 2)
323     glm::mat4 modelaux3(1.0);//Inicializar matriz de Modelo 4x4 auxiliar para la jerarquía (Caso Ejercicio 2)
```

También cabe mencionar que en el archivo P04-318304188_Completo se encuentran ambos programas en un sólo main, solo hace falta comentar y descomentar entre sí.

II. 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.

Olvidé un buen rato revisar la función `CrearCilindro`, por lo que tardé en darme cuenta de la razón por la que mis llantas parecían pentágonos. Pensé que era un problema dentro del segmento de código de las figuras o que algo en la sintaxis del código estaba impidiendo la correcta visualización de estas. Revisé varias veces que el índice en la *MeshList* fuera el correcto para los cilindros, hasta que en una de esas revisiones me dí cuenta de que los “lados”/triángulos que conformaban los bordes eran 5, como lo mencioné en la parte de muestra del código.

```
282  int main()
283  {
284      mainWindow = Window(800, 600);
285      mainWindow.Initialise();
286      //Cilindro y cono reciben resolución (slices, rebanadas) y Radio de circunferencia de la base y tapa
287
288      CrearCubo();//índice 0 en MeshList
289      CrearPiramideTriangular();//índice 1 en MeshList
290      CrearCilindro(5, 1.0f);//índice 2 en MeshList  //SE VE COMO PENTÁGONO PORQUE ESTÁ EN 5 (ESQUINAS)
291      CrearCono(25, 2.0f);//índice 3 en MeshList
292      CrearPiramideCuadrangular();//índice 4 en MeshList
293      CreateShaders();
```

III. Conclusiones:

A. Los ejercicios del reporte: Complejidad, Explicación.

Considero a la práctica en un nivel intermedio debido a lo tardado que requirió establecer cada uno de los objetos y figuras y la jerarquía entre ellas. La explicación fue suficiente y nuevamente con ayuda del video facilitó su comprensión.

B. Comentarios generales:

Es un poco abrumador construir tanto pero a la vez resulta bastante útil saber cómo "unir" los elementos y darles detalles, aunque después de esta experiencia pienso que deberemos establecer los resultados esperados desde antes.

C. Conclusión.

El uso del modelo auxiliar simplificó notablemente la realización de esta práctica, ya que ayuda a reducir varias líneas de código. Creo que este enfoque lo hace más eficiente, aunque puede resultar algo incómodo de manejar, ya que es necesario calcular la traslación a partir de las coordenadas principales. Además, lo más complicado de esta práctica fue lograr la rotación de las figuras y las articulaciones, ya que a veces uno de estos giros afectaba por completo al otro.

IV. BIBLIOGRAFÍA CONSULTADA

[1] —