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: Vázquez Gómez Carlos Iván

Nº de Cuenta: 4200551851

GRUPO DE LABORATORIO: 03

GRUPO DE TEORÍA: 04

SEMESTRE 2025-1

FECHA DE ENTREGA LÍMITE: 07 /09/2024

CALIFICACIÓN: _____

REPORTE DE PRÁCTICA:

1.- Ejercicios

Primera actividad

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 resolver esta actividad de la practica me guie en base al ejercicio planteado en hora de laboratorio, donde ahora nuestro origen seria la cabina y de ahí partiríamos su jerarquía con 2 matrices simulando los dos prismas.

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

//AQUÍ SE DIBUJA LA CABINA, LA BASE, LAS 4 LLANTAS
//Cabina
model = glm::translate(model, glm::vec3(0.0f, 10.0f, -4.0f));
model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f));
modelaux2 = model;
model = glm::scale(model, glm::vec3(6.5f, 4.0f, 3.5f));
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(0.478f, 0.255f, 0.067f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[0]->RenderMesh(); //dibuja cubo y pirámide triangular

model = modelaux2;
```

La implementación de las llantas no fue tan complicado como lo esperaba, ya que lo que nos enseño el profesor de ubicarnos en el origen de nuestro prisma fui entendiendo a detalle como poner las coordenadas correspondientes

```
//BASE LLANTAS

model = glm::translate(model, glm::vec3(0.0f, -3.0f, 0.0f));
modelaux2 = model;
model = glm::scale(model, glm::vec3(6.25f, 3.3f, 6.25f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(1.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[4]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro y cono


model = modelaux2;


//DIBUJO DE LA LLANTA 1
model = glm::translate(model, glm::vec3(2.0f, -2.85f, -2.0f));
model = glm::rotate(model, glm::radians(90.5f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion5()), glm::vec3(0.0f,1.0f, 0.0f));
model = glm::scale(model, glm::vec3(1.25f, 1.3f, 1.25f));

color = glm::vec3(0.0f, 0.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[2]->RenderMeshGeometry(); //dibuja cubo y pirámide triangular

model = modelaux2;


//DIBUJO DE LA LLANTA 2
model = glm::translate(model, glm::vec3(2.0f, -2.85f, 2.0f));
model = glm::rotate(model, glm::radians(90.5f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion6()), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::scale(model, glm::vec3(1.25f, 1.3f, 1.25f));

color = glm::vec3(0.0f, 0.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[2]->RenderMeshGeometry(); //dibuja cubo y pirámide triangular

model = modelaux2;
```

```
//DIBUJO DE LA LLANTA 3
model = glm::translate(model, glm::vec3(-2.0f, -2.85f, 2.0f));
model = glm::rotate(model, glm::radians(90.5f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion7()), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::scale(model, glm::vec3(1.25f, 1.3f, 1.25f));

color = glm::vec3(0.0f, 0.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[2]->RenderMeshGeometry(); //dibuja cubo y pirámide triangular

model = modelaux2;

//DIBUJO DE LA LLANTA 4
model = glm::translate(model, glm::vec3(-2.0f, -2.85f, -2.0f));
model = glm::rotate(model, glm::radians(90.5f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion8()), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::scale(model, glm::vec3(1.25f, 1.3f, 1.25f));

color = glm::vec3(0.0f, 0.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[2]->RenderMeshGeometry(); //dibuja cubo y pirámide triangular
```

```
model = modelaux2;
// Creando el brazo de una grúa
//articulacion1 hasta articulación5 sólo son puntos de rotación o articulación, en este caso no dibujaremos esferas que los representen

//para reiniciar la matriz de modelo con valor de la matriz identidad
model = glm::mat4(1.0);
//AQUÍ SE DIBUJA LA CABINA, LA BASE, LAS 4 LLANTAS

// SE EMPIEZA EL DIBUJO DEL BRAZO
//articulación 1
model = glm::translate(model, glm::vec3(0.0f, 11.5f, -4.0f));
//rotación alrededor de la articulación que une con la cabina
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()), glm::vec3(0.0f, 0.0f, 1.0f));

//primer brazo que conecta con la cabina
// //Traslación inicial para posicionar en -Z a los objetos
//model = glm::translate(model, glm::vec3(0.0f, 0.0f, -4.0f));
//otras transformaciones para el objeto
model = glm::translate(model, glm::vec3(-1.0f, 2.0f, 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 linea 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
//meshList[2]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro y cono

//para descartar la escala que no quiero heredar se carga la información de la matrix auxiliar
model = modelaux;
//articulación 2
model = glm::translate(model, glm::vec3(2.5f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()), 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();
```

```
//segundo brazo
model = glm::translate(model, glm::vec3(0.0f, -2.5f, 0.0f));

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(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

model = modelaux;

//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();

// Crear instancias para completar el brazo y  la cabina. Imporante considerar que la cabina es el nodo padre.
//La cabina y el brazo deben de estar unidos a la cabina

model = modelaux;
//tercer brazo
model = glm::rotate(model, glm::radians(90.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::translate(model, glm::vec3(0.0f, -2.5f, 0.0f));
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(0.0f, 0.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[0]->RenderMesh(); //dibuja cubo y pirámide triangular
model = modelaux;

//articulacion 4 del tercer brazo
model = glm::translate(model, glm::vec3(0.0f, -2.5f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion4()), glm::vec3(1.0f, 1.0f, 0.0f));
modelaux = model;
```

```
//articulacion 4 del tercer brazo
model = glm::translate(model, glm::vec3(0.0f, -2.5f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion4()), glm::vec3(1.0f, 1.0f, 0.0f));
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();

//canasta
model = glm::rotate(model, glm::radians(-45.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::translate(model, glm::vec3(0.0f, -1.5f, 0.0f));
modelaux = model;
model = glm::scale(model, glm::vec3(2.5f, 3.0f, 2.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(1.0f, 1.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[0]->RenderMesh(); //dibuja cubo y pirámide triangular
```

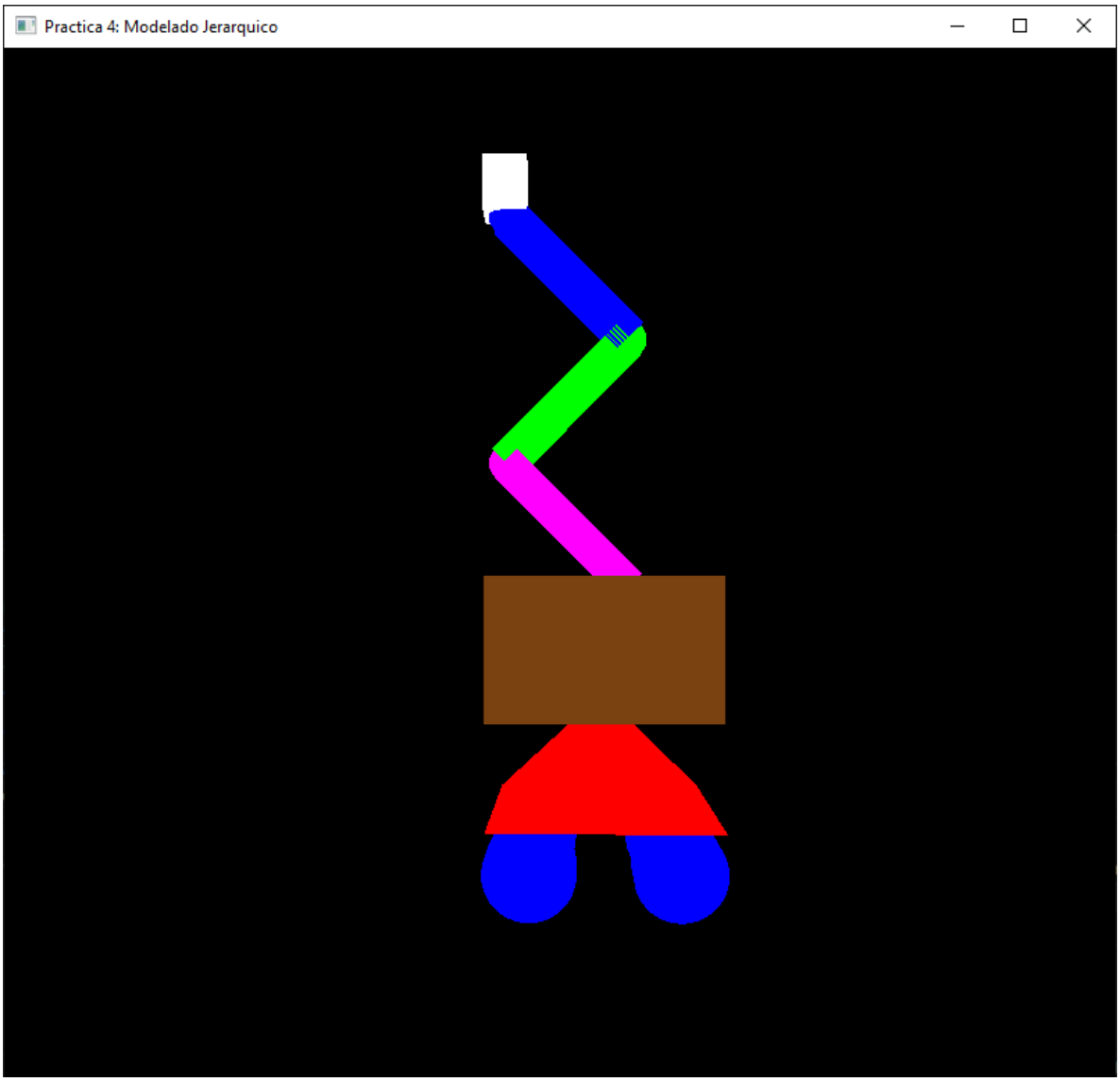
Se tuvieron que crear nuevas variables “articulaciones” para poder observar las rotaciones de la grúa simulada, porque el código que nos proporcionó el profesor solo venía hasta 6 articulaciones.

Por lo tanto, se modificaron los archivos Windows.cpp y Windows.h

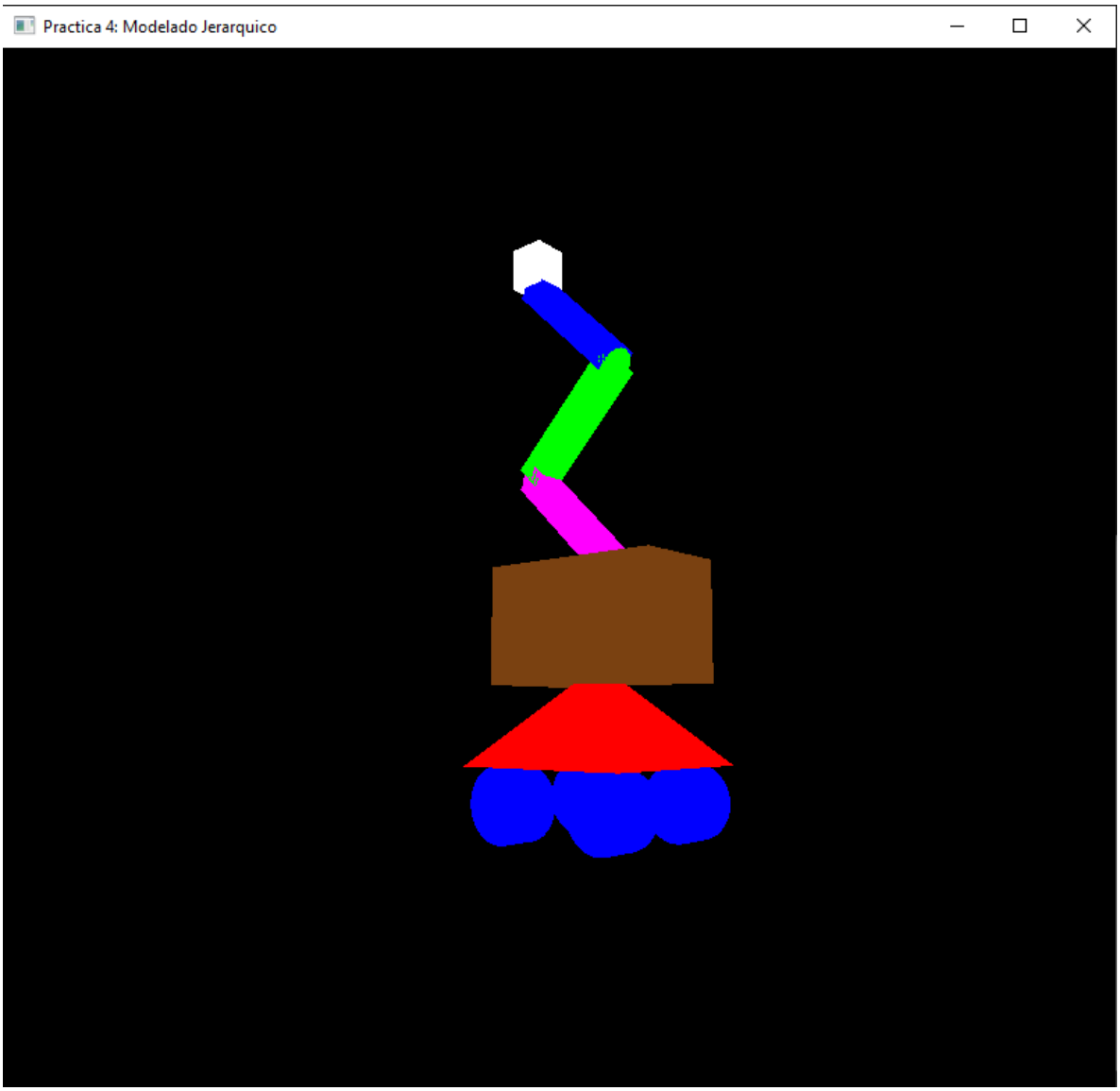
```
GLfloat getarticulacion1() { return articulacion1; }
GLfloat getarticulacion2() { return articulacion2; }
GLfloat getarticulacion3() { return articulacion3; }
GLfloat getarticulacion4() { return articulacion4; }
GLfloat getarticulacion5() { return articulacion5; }
GLfloat getarticulacion6() { return articulacion6; }
GLfloat getarticulacion7() { return articulacion7; }
GLfloat getarticulacion8() { return articulacion8; }
GLfloat getarticulacion9() { return articulacion9; }
GLfloat getarticulacion10() { return articulacion10; }
~Window();
ivate:
GLFWwindow *mainWindow;
GLint width, height;
GLfloat rotax,rotay,rotaz, articulacion1, articulacion2, articulacion3, articulacion4, articulacion5, articulacion6, articulacion7, articulacion8, articulacion9, articulacion10;
```

```
if (key == GLFW_KEY_Z)
{
    theWindow->articulacion7 += 10.0;
}
if (key == GLFW_KEY_I)
{
    theWindow->articulacion8 += 10.0;
}
if (key == GLFW_KEY_O)
{
    theWindow->articulacion9 += 10.0;
}
if (key == GLFW_KEY_P)
{
    theWindow->articulacion10 += 10.0;
}
```

Vista de frente de la grúa



Vista de lado de la grúa.



Se agregará un video para observar su funcionamiento

Segunda actividad

Crear un animal robot 3d

-Intanciando 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

Este ejercicio no se logro completar, ya que me llevo bastante tiempo entender el funcionamiento de las jerarquías y como funcionaban entre sí, y no logre a completarlo en su totalidad, ya que falto la cabeza y las orejas del animal.

```
//CREACION DE ARAÑA ROBOT, PATA ARTICULADA
//Cuerpo
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, 0.0f, -4.0f));
model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f));
modelaux2 = model;
modelaux = model;
model = glm::scale(model, glm::vec3(3.0f, 6.0f, 3.5f));
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, 1.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[2]->RenderMeshGeometry(); //dibuja cubo, pirámide triangular, pirámide base cuadrangular
modelaux2 = model;
modelaux = model;

//Articulacion 1 Torso-brazo1
model = glm::translate(model, glm::vec3(1.0f, -0.5f, 0.5f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()), glm::vec3(0.0f, 0.0f, 1.0f));
modelaux = model;
//dibujar una pequeña esfera
model = glm::scale(model, glm::vec3(0.15f, 0.15f, 0.15f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(1.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
sp.render();

//Brazo1
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));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(1.0f, 1.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[0]->RenderMesh();
model = modelaux;
```

```
//Articulacion 2 Brazo1
model = glm::translate(model, glm::vec3(-5.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()), glm::vec3(0.0f, 0.0f, 1.0f));
modelaux = model;
//dibujar una pequeña esfera
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(1.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
sp.render();

//Brazo1 Parte 2
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));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(1.0f, 1.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[0]->RenderMesh();

model = modelaux2;

//Articulacion 3 Torso-brazo2
model = glm::translate(model, glm::vec3(-1.0f, -0.5f, 0.5f));
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(0.15f, 0.15f, 0.15f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(1.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
sp.render();

//Brazo2
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));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(1.0f, 1.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[0]->RenderMesh();
model = modelaux;
```

```
//Articulacion 4 Brazo1
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
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(1.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
sp.render();

//Brazo2 Parte 2
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));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(1.0f, 1.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[0]->RenderMesh();

model = modelaux2;

//Articulacion 5 Torso-brazo3
model = glm::translate(model, glm::vec3(1.0f, -0.5f, -0.5f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion5()), glm::vec3(0.0f, 0.0f, 1.0f));
modelaux = model;
//dibujar una pequeña esfera
model = glm::scale(model, glm::vec3(0.15f, 0.15f, 0.15f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(1.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
sp.render();

//Brazo3
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));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(1.0f, 1.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[0]->RenderMesh();
model = modelaux;
```

```
//Articulacion 6 Brazo1
model = glm::translate(model, glm::vec3(-5.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion6()), glm::vec3(0.0f, 0.0f, 1.0f));
modelaux = model;
//dibujar una pequeña esfera
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(1.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
sp.render();

//Brazo3 Parte 2
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));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(1.0f, 1.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[0]->RenderMesh();

model = modelaux2;

//Articulacion 7 Torso-brazo4
model = glm::translate(model, glm::vec3(-1.0f, -0.5f, -0.5f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion7()), glm::vec3(0.0f, 0.0f, 1.0f));
modelaux = model;
//dibujar una pequeña esfera
model = glm::scale(model, glm::vec3(0.15f, 0.15f, 0.15f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(1.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
sp.render();

//Brazo4
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));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(1.0f, 1.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[0]->RenderMesh();
model = modelaux;
```

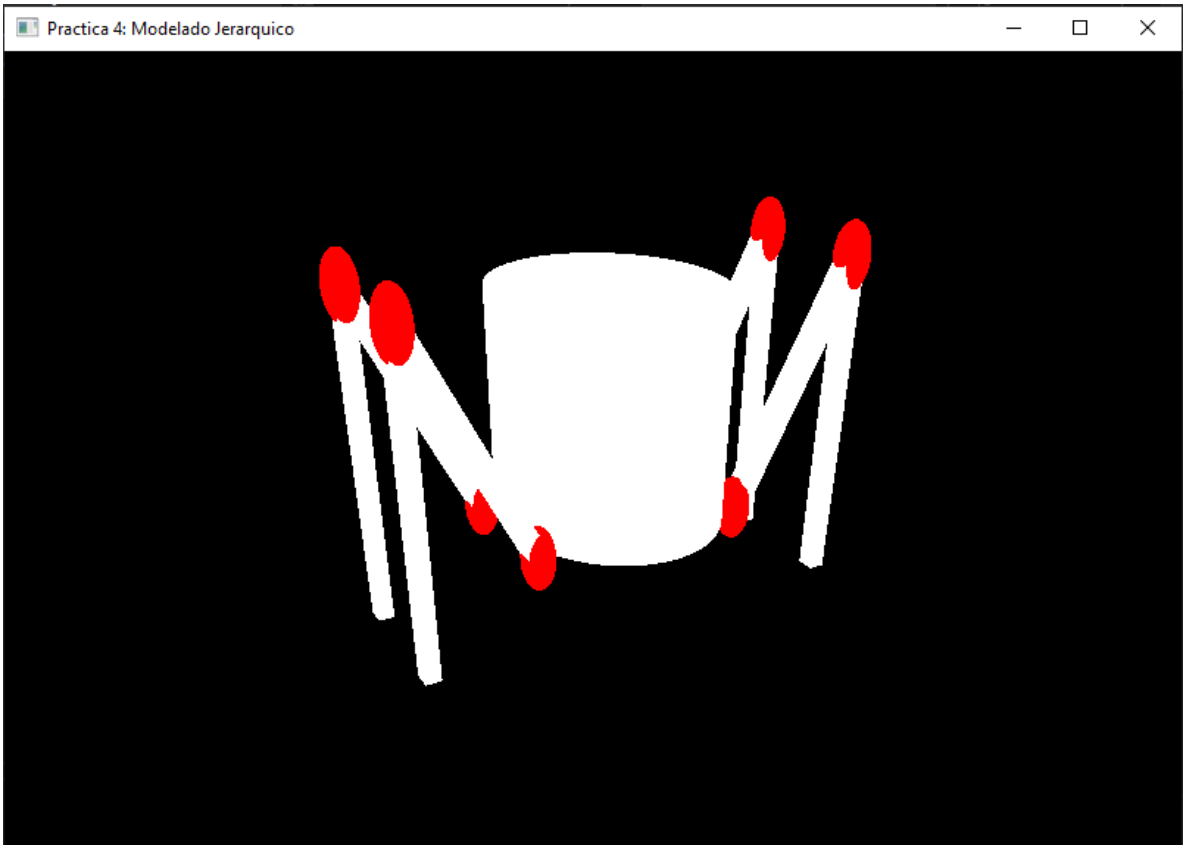
```
//Articulacion 8
model = glm::translate(model, glm::vec3(5.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion8()), glm::vec3(0.0f, 0.0f, 1.0f));
modelaux = model;
//dibujar una pequeña esfera
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(1.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
sp.render();

//Brazo4 Parte 2
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));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(1.0f, 1.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
meshList[0]->RenderMesh();

model = modelaux2;
```

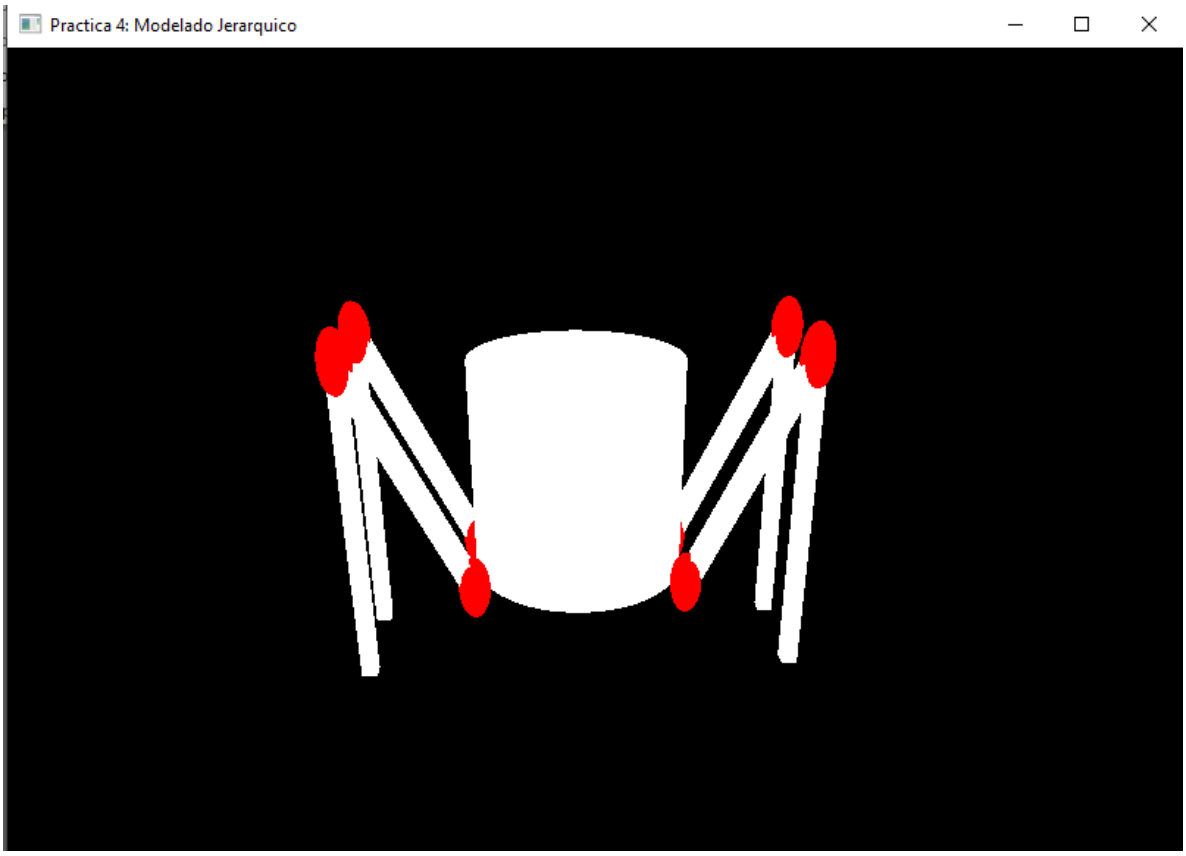
Este animal escogido tuvo mucha similitud a las manos de la grúa, porque era la misma manera de resolverlo, lo único diferente fue crear la cabeza que fue lo que me falto para terminar el ejercicio.

Vista de lado



A demás, para poder mover todas las articulaciones del animal se tuvieron que crear 2 nuevas “articulaciones” en los archivos Windows.cpp y Windows.h

Vista de frente



2.- Problemas

Uno de los mayores problemas que presente fue el entendimiento de las jerarquías, ya que cuando resolví el ejercicio de clases pensé que lo había hecho de una manera satisfactoria, pero al momento de intentar colocar el mismo código a mi actividad 1 de la practica me percate que estaba mal jerarquizado, lo que hizo que empezara de 0 lo que ya había realizado en el ejercicio de clase.

3.- Conclusión

Cada practica que realizamos se va notando una dificultad mayor a la anterior, esta es la primera vez que no logro acabar las actividades solicitadas, y no por falta de tiempo, sino porque no entendía como hacerlo correcto, y aun así sigo teniendo muchas dudas sobre la jerarquización, para mi es un tema un difícil de entender a la primera, esto me da a entender que necesito poner mas de mi parte y estudiar porque esto es algo “básico” para lo que sigue.