



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 N° 04**

**NOMBRE COMPLETO:** Romero Dominguez Ricardo Damián

**N° de Cuenta:** 319094493

**GRUPO DE LABORATORIO:** 02

**GRUPO DE TEORÍA:** 04

**SEMESTRE 2025-2**

**FECHA DE ENTREGA LÍMITE:** 03/09/2025

**CALIFICACIÓN:** \_\_\_\_\_

## ■ Índice

<b>1</b>	<b>Ejecución de ejercicios</b>	<b>2</b>
1.1	Dibujo completo de grúa . . . . .	2
	Creación de cabina • Creación de la base • Creación de ruedas • Teclas para movimiento de ruedas • Creación de articulaciones • Teclas para movimiento de articulaciones	
1.2	Dibujo de mascota . . . . .	6
	Creación de cuerpo • Creación de cabeza • Creación de piernas	
<b>2</b>	<b>Problemas presentados</b>	<b>11</b>
<b>3</b>	<b>Conclusión</b>	<b>12</b>

## 1. EJECUCIÓN DE EJERCICIOS

### 1.1. Dibujo completo de grúa

#### 1.1.1. Creación de cabina

Para poder dibujar la grúa de manera correcta, primero se creó el primer objeto de nuestro árbol, el cuál sería la cabina en forma de cubo, donde su centro se posicionó en (0.0f, 5.0f, -4.0f), y después se escaló dicho cubo con ayuda de una variable auxiliar que nos ayudaría a no guardar los cambios del escalamiento sobre la matriz del modelo, pero si su traslación inicial.

**Código 1.** Creación de cabina.

```
1 // Crear Cabina
2 {
3     model = glm::mat4(1.0);
4     model = glm::translate(model, glm::
5         ::vec3(0.0f, 5.0f, -4.0f));
6     modelaux = model;
7     model = glm::scale(model, glm::
8         vec3(5.0f, 5.0f, 5.0f));
9     glUniformMatrix4fv(
10         uniformProjection, 1, GL_FALSE,
11         glm::value_ptr(projection));
12     glUniformMatrix4fv(uniformView,
13         1, GL_FALSE, glm::value_ptr(
14         camera.calculateViewMatrix()));
15     glUniformMatrix4fv(uniformModel,
16         1, GL_FALSE, glm::value_ptr(
17         model));
18     //la línea de proyección solo se
19     manda una vez a menos que en
20     tiempo de ejecución
21     //se programe cambio entre
22     proyección ortogonal y
23     perspectiva
24     color = glm::vec3(1.0f, 0.0f, 0.0
25         f);
26     glUniform3fv(uniformColor, 1, glm
27         ::value_ptr(color)); //para
28     cambiar el color del objetos
29     meshList[0]->RenderMesh(); //
30     dibuja cubo, pirámide triangular
31     , pirámide base cuadrangular
32 }
```

#### 1.1.2. Creación de la base

Posteriormente creamos la base con ayuda de los cambios realizados en el dibujo (escalamiento y rota-

ción), y al terminar se vuelven a los valores iniciales (traslación de cabina).

Cabe aclarar que en esta parte la matriz de vista y proyección se dejan de recalculer ya que se hicieron en el paso anterior

**Código 2.** Creación de base.

```
1 // Crear Base
2 {
3     model = glm::scale(model, glm::
4         vec3(1.0f, 0.5f, 1.0f));
5     model = glm::translate(model, glm
6         ::vec3(0.0f, -1.0f, 0.0f));
7     glUniformMatrix4fv(uniformModel,
8         1, GL_FALSE, glm::value_ptr(
9         model));
10    color = glm::vec3(0.2f, 0.0f, 0.7
11        f);
12    glUniform3fv(uniformColor, 1, glm
13        ::value_ptr(color));
14    meshList[4]->RenderMeshGeometry()
15        ;
16    model = modelaux;
17 }
18 modelauxRuedas = model;
```

#### 1.1.3. Creación de ruedas

Una vez acabada la base continuamos con el dibujo de las ruedas, donde se creó una variable auxiliar adicional para no guardar ninguno los cambios de los cambios que se realicen al modificar la matriz de modelo para dibujar las ruedas.

De esta manera, al haber creado un punto de restauración de la matriz de modelo, la acción consecuente fue trasladar un cilindro a su posición donde fungirá como la rueda izquierda frontal, rotarla para que la cara plana vea hacia el eje Z y escalar dicho cilindro para ajustar su forma a la de una llanta

**Código 3.** Preparar las ruedas.

```
1 model = glm::translate(model, glm::
2     vec3(-1.9f, -4.0f, 3.0f));
3 model = glm::rotate(model, glm::
4     radians(90.0f), glm::vec3(1.0f,
5     0.0f, 0.0f));
6 model = glm::scale(model, glm::vec3
7     (1.9f, 1.3f, 1.9f));
```

Al haber preparado el cilindro para el modelo, se prosiguió con usar la variable auxiliar para dibujar el cilindro y no guardar su rotación independiente (Letra Z).

#### Código 4. Primer rueda.

```

1  modelaux = model;
2  model = glm::rotate(model, glm::
    radians(mainWindow.getNeumatico1
    ()), glm::vec3(0.0f, 1.0f, 0.0f)
    );
3  glUniformMatrix4fv(uniformModel, 1,
    GL_FALSE, glm::value_ptr(model)
    );
4  color = glm::vec3(0.2f, 0.5f, 0.7f)
    ;
5  glUniform3fv(uniformColor, 1, glm::
    value_ptr(color)); //para
    cambiar el color del objetos
6  meshList[2]->RenderMeshGeometry();
7  model = modelaux;
8

```

Para el dibujo de las demas ruedas solo se desplazaron y su rotaciones de guardaron con ayuda de la matrix auxiliar para no afectar la rotación de las demás.

De igual manera, al término del dibujo de todas las ruedas, se regresó al estado inicial con ayuda de la matrix auxiliar creada **modelauxRuedas**

#### Código 5. Primer rueda.

```

13  ::vec3(0.0f, -4.5f, 0.0f));
14
15  modelaux = model;
16  model = glm::rotate(model, glm::
    radians(mainWindow.getNeumatico3
    ()), glm::vec3(0.0f, 1.0f, 0.0f)
    );
17  glUniformMatrix4fv(uniformModel,
    1, GL_FALSE, glm::value_ptr(
    model));
18  meshList[2]->RenderMeshGeometry()
    ;
19  model = modelaux;
20
21  // Rueda derecha frontal
22  model = glm::translate(model, glm
    ::vec3(-2.1f, 0.0f, 0.0f));
23
24  modelaux = model;
25  model = glm::rotate(model, glm::
    radians(mainWindow.getNeumatico4
    ()), glm::vec3(0.0f, 1.0f, 0.0f)
    );
26  glUniformMatrix4fv(uniformModel,
    1, GL_FALSE, glm::value_ptr(
    model));
27  meshList[2]->RenderMeshGeometry()
    ;
28
29  }
30  model = modelauxRuedas;
31

```

#### 1.1.4. Teclas para movimiento de ruedas

- Rueda izquierda frontal - Z
- Rueda izquierda trasera - X
- Rueda derecha trasera - C
- Rueda derecha frontal - V

Una vez que se terminaron de dibujar las ruedas, se prosiguió con el brazo de la grúa, de modo que de manera general, el proceso para cada articulación fue el siguiente

#### 1.1.5. Creación de articulaciones

1. Trasladar y rotar (con teclado) en el punto donde se va a mover la articulación
2. Trasladar y rotar para acomodar la articulación en su posición inicial
3. Guardar un punto de recuperación de la matriz
4. Escalar y dibujar la articulación

```

1  // Rueda izquierda trasera
2  model = glm::translate(model, glm
    ::vec3(2.1f, 0.0f, 0.0f));
3
4  modelaux = model;
5  model = glm::rotate(model, glm::
    radians(mainWindow.getNeumatico2
    ()), glm::vec3(0.0f, 1.0f, 0.0f)
    );
6  glUniformMatrix4fv(uniformModel,
    1, GL_FALSE, glm::value_ptr(
    model));
7  meshList[2]->RenderMeshGeometry()
    ;
8  model = modelaux;
9
10
11  // Rueda derecha trasera
12  model = glm::translate(model, glm

```

## 5. Volver al primer paso para la siguiente articulación

### Código 6. Brazo y Jaula.

```
1 // Articulaciones
2 {
3     // Articulación 1
4     model = glm::translate(model, glm
5         ::vec3(0.0f, 1.0f, 0.0f));
6     //rotación alrededor de la
7     articulación que une con la
8     cabina
9     model = glm::rotate(model, glm::
10         radians(mainWindow.
11         getarticulacion1()), glm::vec3
12         (0.0f, 0.0f, 1.0f));
13
14     //primer brazo que conecta con la
15     cabina
16     // //Traslación inicial para
17     posicionar en -Z a los objetos
18     //model = glm::translate(model,
19     glm::vec3(0.0f, 0.0f, -4.0f));
20     //otras transformaciones para el
21     objeto
22     model = glm::translate(model, glm
23         ::vec3(-1.0f, 2.0f, 0.0f));
24     model = glm::rotate(model, glm::
25         radians(135.0f), glm::vec3(0.0f,
26         0.0f, 1.0f));
27     modelaux = model;
28     model = glm::scale(model, glm::
29         vec3(5.0f, 1.0f, 1.0f));
30     glUniformMatrix4fv(uniformModel,
31         1, GL_FALSE, glm::value_ptr(
32         model));
33     //la línea de proyección solo se
34     manda una vez a menos que en
35     tiempo de ejecución
36     //se programe cambio entre
37     proyección ortogonal y
38     perspectiva
39     color = glm::vec3(1.0f, 0.0f, 1.0
40         f);
41     glUniform3fv(uniformColor, 1, glm
42         ::value_ptr(color)); //para
43     cambiar el color del objetos
44     meshList[0]->RenderMesh(); //
45     dibuja cubo, pirámide triangular
46     , pirámide base cuadrangular
47     //meshList[2]->RenderMeshGeometry
```

```
24    ()); //dibuja las figuras geomé
25     tricas cilindro y cono
26
27     //para descartar la escala que no
28     quiero heredar se carga la
29     información de la matrix
30     auxiliar
31     model = modelaux;
32     //articulación 2
33     model = glm::translate(model, glm
34         ::vec3(2.5f, 0.0f, 0.0f));
35     model = glm::rotate(model, glm::
36         radians(mainWindow.
37         getarticulacion2()), glm::vec3
38         (0.0f, 0.0f, 1.0f));
39     modelaux = model;
40     //dibujar una pequena esfera
41     model = glm::scale(model, glm::
42         vec3(0.5f, 0.5f, 0.5f));
43     glUniformMatrix4fv(uniformModel,
44         1, GL_FALSE, glm::value_ptr(
45         model));
46     sp.render();
47
48     model = modelaux;
49     //segundo brazo
50     model = glm::translate(model, glm
51         ::vec3(0.0f, -2.5f, 0.0f));
52
53     modelaux = model;
54     model = glm::scale(model, glm::
55         vec3(1.0f, 5.0f, 1.0f));
56     glUniformMatrix4fv(uniformModel,
57         1, GL_FALSE, glm::value_ptr(
58         model));
59     color = glm::vec3(0.0f, 1.0f, 0.0
60         f);
61     glUniform3fv(uniformColor, 1, glm
62         ::value_ptr(color)); //para
63     cambiar el color del objetos
64     meshList[0]->RenderMesh(); //
65     dibuja cubo y pirámide
66     triangular
67
68     model = modelaux;
69
70     //articulación 3 extremo derecho
71     del segundo brazo
72     model = glm::translate(model, glm
73         ::vec3(0.0f, -2.5f, 0.0f));
74     model = glm::rotate(model, glm::
75         radians(mainWindow.
76         getarticulacion3()), glm::vec3
```

```

52     (0.0f, 0.0f, 1.0f));
53     modelaux = model;
54
55     //dibujar una pequena esfera
56     model = glm::scale(model, glm::
57         vec3(0.5f, 0.5f, 0.5f));
58     glUniformMatrix4fv(uniformModel,
59         1, GL_FALSE, glm::value_ptr(
60         model));
61     sp.render();
62
63     // Crear instancias para
64     //completar el brazo y la cabina.
65     //Importante considerar que la
66     //cabina es el nodo padre.
67     //La cabina y el brazo deben de
68     //estar unidos a la cabina
69
70     model = modelaux;
71     //tercer brazo
72     model = glm::translate(model, glm
73         ::vec3(2.5f, 0.0f, 0.0f));
74     model = glm::rotate(model, glm::
75         radians(90.0f), glm::vec3(0.0f,
76         0.0f, 1.0f));
77     modelaux = model;
78     model = glm::scale(model, glm::
79         vec3(1.0f, 5.0f, 1.0f));
80     glUniformMatrix4fv(uniformModel,
81         1, GL_FALSE, glm::value_ptr(
82         model));
83     color = glm::vec3(0.0f, 1.0f, 5.0
84         f);
85     glUniform3fv(uniformColor, 1, glm
86         ::value_ptr(color)); //para
87     //cambiar el color del objetos
88     meshList[0]->RenderMesh(); //
89     //dibuja cubo y pirámide
90     //triangular
91
92     model = modelaux;
93
94     //articulación 4 extremo derecho
95     //del segundo brazo
96     model = glm::translate(model, glm
97         ::vec3(0.0f, -2.5f, 0.0f));
98     model = glm::rotate(model, glm::
99         radians(-135.0f), glm::vec3(0.0f
100         , 0.0f, 1.0f));
101     model = glm::rotate(model, glm::
102         radians(mainWindow.
103         getarticulacion4()), glm::vec3
104

```

```

80     (1.0f, 0.0f, 0.0f));
81     modelaux = model;
82
83     //dibujar una pequena esfera
84     model = glm::scale(model, glm::
85         vec3(0.5f, 0.5f, 0.5f));
86     glUniformMatrix4fv(uniformModel,
87         1, GL_FALSE, glm::value_ptr(
88         model));
89     sp.render();
90
91     model = modelaux;
92     //Jaula
93     //model = glm::translate(model,
94         glm::vec3(2.5f, 0.0f, 0.0f));
95     modelaux = model;
96     model = glm::scale(model, glm::
97         vec3(5.0f, 3.0f, 3.0f));
98     glUniformMatrix4fv(uniformModel,
99         1, GL_FALSE, glm::value_ptr(
100         model));
101     color = glm::vec3(0.5f, 0.5f, 0.0
102         f);
103     glUniform3fv(uniformColor, 1, glm
104         ::value_ptr(color)); //para
105     //cambiar el color del objetos
106     meshList[0]->RenderMesh(); //
107     //dibuja cubo y pirámide
108     //triangular
109
110     model = modelaux;
111
112     //articulación 4 extremo derecho
113     //del segundo brazo
114     model = glm::translate(model, glm
115         ::vec3(0.0f, -2.5f, 0.0f));
116     model = glm::rotate(model, glm::
117         radians(mainWindow.
118         getarticulacion5()), glm::vec3
119         (0.0f, 0.0f, 1.0f));
120     modelaux = model;
121 }
122

```

### 1.1.6. Teclas para movimiento de articulaciones

- Primer Brazo - F
- Segundo Brazo - G
- Tercer Brazo - H
- Jaula - J

## 1.2. Dibujo de mascota

Para el dibujo de la mascota, se eligió a una araña pensando en que dado que va a ser la mascota que nos acompañará en el proyecto final y tomando en cuenta el hecho de que la temática de mi elección fue billy y mandy, mi mascota sería una araña pensando en el hijo de Billy.

Es así como tanto para el cuerpo como para la cabeza, se dibujan dos esferas, donde el cuerpo es más largo en su eje Z.

### 1.2.1. Creación de cuerpo

Para la creación del cuerpo, se define la matriz de modelo, se traslada a la posición inicial (0.0f, 5.0f, -4.0f) y posteriormente se usa una matriz auxiliar para escalar la esfera y no guardar los cambios, y dado que es el primer objeto que creamos (al menos para este ejercicio), se definen las matrices de proyección y de vista.

**Código 7.** Creación de cuerpo.

```
1 // Cuerpo
2 {
3     model = glm::mat4(1.0);
4     model = glm::translate(model, glm::
5         ::vec3(0.0f, 5.0f, -4.0f));
6     modelaux = model;
7     model = glm::scale(model, glm::
8         vec3(2.0f, 1.5f, 1.5f));
9     glUniformMatrix4fv(
10         uniformProjection, 1, GL_FALSE,
11         glm::value_ptr(projection));
12     glUniformMatrix4fv(uniformView,
13         1, GL_FALSE, glm::value_ptr(
14         camera.calculateViewMatrix()));
15     glUniformMatrix4fv(uniformModel,
16         1, GL_FALSE, glm::value_ptr(
17         model));
18     color = glm::vec3(0.0f, 0.1f, 0.1
19         f);
20     glUniform3fv(uniformColor, 1, glm
21         ::value_ptr(color));
22     sp.render();
23     model = modelaux;
24 }
```

### 1.2.2. Creación de cabeza

Para la creación de la cabeza se traslado una esfera sin rotar, pero de igual manera se empleó la

jerarquía heredada de su posición para la creación de los ojos, donde simplemente se escaló un ojo y de igual manera se empleó la jerarquía heredada de su escalamiento para posicionar los demás ojos.

**Código 8.** Creación de cuerpo.

```
1 // Cabeza
2 modelaux = model;
3 model = glm::translate(model, glm::
4     vec3(-2.2f, 0.0f, 0.0f));
5 glUniformMatrix4fv(uniformModel, 1,
6     GL_FALSE, glm::value_ptr(model)
7 );
8 sp.render();
9
10 // Ojo derecho inferior
11 model = glm::translate(model, glm::
12     vec3(-0.85f, 0.2f, -0.25f));
13 model = glm::scale(model, glm::vec3
14     (0.2f, 0.2f, 0.2f));
15 glUniformMatrix4fv(uniformModel, 1,
16     GL_FALSE, glm::value_ptr(model)
17 );
18 color = glm::vec3(1.0f, 1.0f, 1.0f)
19 ;
20 glUniform3fv(uniformColor, 1, glm::
21     value_ptr(color)); //para
22     cambiar el color del objetos
23 sp.render();
24
25 // Ojo derecho medio
26 model = glm::translate(model, glm::
27     vec3(0.5f, 1.0f, -0.8f));
28 glUniformMatrix4fv(uniformModel, 1,
29     GL_FALSE, glm::value_ptr(model)
30 );
31 sp.render();
32
33 // Ojo derecho superior
34 model = glm::translate(model, glm::
35     vec3(0.5f, 1.0f, -0.8f));
36 glUniformMatrix4fv(uniformModel, 1,
37     GL_FALSE, glm::value_ptr(model)
38 );
39 sp.render();
40
41 model = modelaux;
42
43 modelaux = model;
44 model = glm::translate(model, glm::
```



```

    vec3(-2.2f, 0.0f, 0.0f));
31 model = glm::rotate(model, glm::
    radians(15.0f), glm::vec3(0.0f,
    1.0f, 0.0f));
32 model = glm::rotate(model, glm::
    radians(80.0f), glm::vec3(1.0f,
    0.0f, 0.0f));
33 // Ojo derecho inferior
34 model = glm::translate(model, glm::
    vec3(-0.85f, 0.2f, -0.25f));
35 model = glm::scale(model, glm::vec3
    (0.2f, 0.2f, 0.2f));
36 glUniformMatrix4fv(uniformModel, 1,
    GL_FALSE, glm::value_ptr(model)
    );
37 sp.render();
38
39 // Ojo derecho medio
40 model = glm::translate(model, glm::
    vec3(0.5f, 1.0f, -0.8f));
41 glUniformMatrix4fv(uniformModel, 1,
    GL_FALSE, glm::value_ptr(model)
    );
42 sp.render();
43
44 // Ojo derecho superior
45 model = glm::translate(model, glm::
    vec3(0.5f, 1.0f, -0.8f));
46 glUniformMatrix4fv(uniformModel, 1,
    GL_FALSE, glm::value_ptr(model)
    );
47 sp.render();
48
49 model = modelaux;
50

```

### 1.2.3. Creación de piernas

Finalmente, para la creación de las piernas, estas son cubos escalados y rotados que giran al rededor del centro del cuerpo (primer parte de la pierna), y que giran al rededor de un círculo que une las articulaciones (segunda parte de la pierna).

De igual manera se emplearon más matrices auxiliares para evitar guardar en la matriz de modelo la rotación que genera cada pierna ya que están declaradas de manera jerárquica.

#### Código 9. Creación de cuerpo.

```

1 // Pierna izquierda frontal
2 {
3     modelaux1 = model;

```

```

4     model = glm::rotate(model,
    glm::radians(mainWindow.
    getarticulacion1()), glm::vec3
    (0.0f, 1.0f, 0.0f));
5     model = glm::translate(
    model, glm::vec3(-2.0f, 1.3f,
    1.5f));
6     model = glm::rotate(model,
    glm::radians(-60.0f), glm::vec3
    (1.0f, 0.0f, 0.0f));
7     model = glm::rotate(model,
    glm::radians(-40.0f), glm::vec3
    (0.0f, 1.0f, 0.0f));
8     modelaux = model;
9     model = glm::scale(model,
    glm::vec3(0.1f, 0.1f, 3.0f));
10    glUniformMatrix4fv(
    uniformModel, 1, GL_FALSE, glm::
    value_ptr(model));
11    color = glm::vec3(0.0f, 0.1
    f, 0.1f);
12    glUniform3fv(uniformColor,
    1, glm::value_ptr(color));
13    meshList[0]->RenderMesh();
    //dibuja cubo, pirámide
    triangular, pirámide base
    cuadrangular
14
15    model = modelaux;
16
17
18    model = glm::translate(
    model, glm::vec3(0.0f, 0.0f, 1.5
    f));
19    modelaux = model;
20    model = glm::scale(model,
    glm::vec3(0.2f, 0.2f, 0.2f));
21    glUniformMatrix4fv(
    uniformModel, 1, GL_FALSE, glm::
    value_ptr(model));
22    sp.render();
23    model = modelaux;
24
25    modelaux = model;
26    model = glm::rotate(model,
    glm::radians(mainWindow.
    getarticulacion2()), glm::vec3
    (0.0f, 1.0f, 0.0f));
27    model = glm::rotate(model,
    glm::radians(40.0f), glm::vec3
    (0.0f, 1.0f, 0.0f));
28    model = glm::rotate(model,
    glm::radians(60.0f), glm::vec3

```



```

29     (1.0f, 0.0f, 0.0f));
30     model = glm::translate(
31     model, glm::vec3(0.05f, -2.3f,
32     0.8f));
33     model = glm::rotate(model,
34     glm::radians(-75.0f), glm::vec3
35     (1.0f, 0.0f, 0.0f));
36     model = glm::rotate(model,
37     glm::radians(145.0f), glm::vec3
38     (1.0f, 0.0f, 0.0f));
39     model = glm::scale(model,
40     glm::vec3(0.1f, 0.1f, 5.0f));
41     glUniformMatrix4fv(
42     uniformModel, 1, GL_FALSE, glm::
43     value_ptr(model));
44     meshList[0]->RenderMesh();
45     //dibuja cubo, pirámide
46     triangular, pirámide base
47     cuadrangular
48
49     model = modelaux1;
50 }
51
52 // Pierna izquierda intermedia
53 {
54     modelaux2 = model;
55     model = glm::rotate(model,
56     glm::radians(mainWindow.
57     getarticulacion3()), glm::vec3
58     (0.0f, 1.0f, 0.0f));
59     model = glm::translate(
60     model, glm::vec3(-1.0f, 1.3f,
61     1.8f));
62     model = glm::rotate(model,
63     glm::radians(-60.0f), glm::vec3
64     (1.0f, 0.0f, 0.0f));
65     model = glm::rotate(model,
66     glm::radians(-30.0f), glm::vec3
67     (0.0f, 1.0f, 0.0f));
68     modelaux = model;
69     model = glm::scale(model,
70     glm::vec3(0.1f, 0.1f, 3.0f));
71     glUniformMatrix4fv(
72     uniformModel, 1, GL_FALSE, glm::
73     value_ptr(model));
74     meshList[0]->RenderMesh();
75     //dibuja cubo, pirámide
76     triangular, pirámide base
77     cuadrangular
78
79     model = modelaux2;
80 }
81
82 // Pierna izquierda trasera
83 {

```

```

54     model = modelaux;
55
56     model = glm::translate(
57     model, glm::vec3(0.0f, 0.0f, 1.5
58     f));
59     modelaux = model;
60     model = glm::scale(model,
61     glm::vec3(0.2f, 0.2f, 0.2f));
62     glUniformMatrix4fv(
63     uniformModel, 1, GL_FALSE, glm::
64     value_ptr(model));
65     sp.render();
66     model = modelaux;
67
68     modelaux = model;
69     model = glm::rotate(model,
70     glm::radians(mainWindow.
71     getarticulacion4()), glm::vec3
72     (0.0f, 1.0f, 0.0f));
73     model = glm::rotate(model,
74     glm::radians(30.0f), glm::vec3
75     (0.0f, 1.0f, 0.0f));
76     model = glm::rotate(model,
77     glm::radians(60.0f), glm::vec3
78     (1.0f, 0.0f, 0.0f));
79     model = glm::translate(
80     model, glm::vec3(0.05f, -2.3f,
81     0.8f));
82     model = glm::rotate(model,
83     glm::radians(-75.0f), glm::vec3
84     (1.0f, 0.0f, 0.0f));
85     model = glm::rotate(model,
86     glm::radians(145.0f), glm::vec3
87     (1.0f, 0.0f, 0.0f));
88     model = glm::scale(model,
89     glm::vec3(0.1f, 0.1f, 5.0f));
90     glUniformMatrix4fv(
91     uniformModel, 1, GL_FALSE, glm::
92     value_ptr(model));
93     meshList[0]->RenderMesh();
94     //dibuja cubo, pirámide
95     triangular, pirámide base
96     cuadrangular
97
98     model = modelaux2;
99 }
100
101 // Pierna izquierda trasera
102 {

```

83        modelaux3 = model; 84        model = glm::rotate(model, 85        glm::radians(mainWindow. 86        getarticulacion5()), glm::vec3 87        (0.0f, 1.0f, 0.0f)); 88        model = glm::translate( 89        model, glm::vec3(0.0f, 1.3f, 1.8 90        f)); 91        model = glm::rotate(model, 92        glm::radians(-60.0f), glm::vec3 93        (1.0f, 0.0f, 0.0f)); 94        model = glm::rotate(model, 95        glm::radians(-20.0f), glm::vec3 96        (0.0f, 1.0f, 0.0f)); 97        modelaux = model; 98        model = glm::scale(model, 99        glm::vec3(0.1f, 0.1f, 3.0f)); 100        glUniformMatrix4fv( 101        uniformModel, 1, GL_FALSE, glm:: 102        value_ptr(model)); 103        meshList[0]->RenderMesh(); 104        //dibuja cubo, pirámide 105        triangular, pirámide base 106        cuadrangular 107 108        model = modelaux; 109 110        model = glm::translate( 111        model, glm::vec3(0.0f, 0.0f, 1.5 112        f)); 113        modelaux = model; 114        model = glm::scale(model, 115        glm::vec3(0.2f, 0.2f, 0.2f)); 116        glUniformMatrix4fv( 117        uniformModel, 1, GL_FALSE, glm:: 118        value_ptr(model)); 119        sp.render(); 120        model = modelaux; 121 122        modelaux = model; 123        model = glm::rotate(model, 124        glm::radians(mainWindow. 125        getarticulacion6()), glm::vec3 126        (0.0f, 1.0f, 0.0f)); 127        model = glm::rotate(model, 128        glm::radians(20.0f), glm::vec3 129        (0.0f, 1.0f, 0.0f)); 130        model = glm::rotate(model, 131        glm::radians(60.0f), glm::vec3 132        (1.0f, 0.0f, 0.0f)); 133        model = glm::translate( 134        model, glm::vec3(0.05f, -2.3f,	0.8f)); 108        model = glm::rotate(model, 109        glm::radians(-75.0f), glm::vec3 110        (1.0f, 0.0f, 0.0f)); 111        model = glm::rotate(model, 112        glm::radians(145.0f), glm::vec3 113        (1.0f, 0.0f, 0.0f)); 114        model = glm::scale(model, 115        glm::vec3(0.1f, 0.1f, 5.0f)); 116        glUniformMatrix4fv( 117        uniformModel, 1, GL_FALSE, glm:: 118        value_ptr(model)); 119        meshList[0]->RenderMesh(); 120        //dibuja cubo, pirámide 121        triangular, pirámide base 122        cuadrangular 123 124        model = modelaux1; 125        } 126        model = glm::rotate(model, glm:: 127        radians(-180.0f), glm::vec3(0.0f 128        , 1.0f, 0.0f)); 129 130        // Pierna derecha frontal 131        { 132            modelaux1 = model; 133            model = glm::rotate(model, 134            glm::radians(mainWindow. 135            getarticulacion7()), glm::vec3 136            (0.0f, 1.0f, 0.0f)); 137            model = glm::translate( 138            model, glm::vec3(2.0f, 1.3f, 1.5 139            f)); 140 141            model = glm::rotate(model, 142            glm::radians(-60.0f), glm::vec3 143            (1.0f, 0.0f, 0.0f)); 144            model = glm::rotate(model, 145            glm::radians(40.0f), glm::vec3 146            (0.0f, 1.0f, 0.0f)); 147            modelaux = model; 148            model = glm::scale(model, 149            glm::vec3(0.1f, 0.1f, 3.0f)); 150            glUniformMatrix4fv( 151            uniformModel, 1, GL_FALSE, glm:: 152            value_ptr(model)); 153            meshList[0]->RenderMesh(); 154            //dibuja cubo, pirámide 155            triangular, pirámide base 156            cuadrangular
---	---

```

133     model = modelaux;
134
135
136     model = glm::translate(
137 model, glm::vec3(0.0f, 0.0f, 1.5
138 f));
139
140     modelaux = model;
141     model = glm::scale(model,
142 glm::vec3(0.2f, 0.2f, 0.2f));
143     glUniformMatrix4fv(
144 uniformModel, 1, GL_FALSE, glm::
145 value_ptr(model));
146     sp.render();
147     model = modelaux;
148
149     modelaux = model;
150     model = glm::rotate(model,
151 glm::radians(mainWindow.
152 getarticulacion8()), glm::vec3
153 (0.0f, 1.0f, 0.0f));
154     model = glm::rotate(model,
155 glm::radians(-40.0f), glm::vec3
156 (0.0f, 1.0f, 0.0f));
157     model = glm::rotate(model,
158 glm::radians(60.0f), glm::vec3
159 (1.0f, 0.0f, 0.0f));
160     model = glm::translate(
161 model, glm::vec3(0.05f, -2.3f,
162 0.8f));
163     model = glm::rotate(model,
164 glm::radians(-75.0f), glm::vec3
165 (1.0f, 0.0f, 0.0f));
166     model = glm::rotate(model,
167 glm::radians(145.0f), glm::vec3
168 (1.0f, 0.0f, 0.0f));
169     model = glm::scale(model,
170 glm::vec3(0.1f, 0.1f, 5.0f));
171     glUniformMatrix4fv(
172 uniformModel, 1, GL_FALSE, glm::
173 value_ptr(model));
174     meshList[0]->RenderMesh();
175 //dibuja cubo, pirámide
176 triangular, pirámide base
177 cuadrangular
178
179     model = modelaux1;
180 }
181
182 // Pierna derecha intermedia
183 {
184     modelaux2 = model;
185     model = glm::rotate(model,

```

```

162 glm::radians(mainWindow.
163 getarticulacion9()), glm::vec3
164 (0.0f, 1.0f, 0.0f));
165     model = glm::translate(
166 model, glm::vec3(1.0f, 1.3f, 1.8
167 f));
168     model = glm::rotate(model,
169 glm::radians(-60.0f), glm::vec3
170 (1.0f, 0.0f, 0.0f));
171     model = glm::rotate(model,
172 glm::radians(30.0f), glm::vec3
173 (0.0f, 1.0f, 0.0f));
174     modelaux = model;
175     model = glm::scale(model,
176 glm::vec3(0.1f, 0.1f, 3.0f));
177     glUniformMatrix4fv(
178 uniformModel, 1, GL_FALSE, glm::
179 value_ptr(model));
180     meshList[0]->RenderMesh();
181 //dibuja cubo, pirámide
182 triangular, pirámide base
183 cuadrangular
184
185     model = modelaux;
186
187     model = glm::translate(
188 model, glm::vec3(0.0f, 0.0f, 1.5
189 f));
190     modelaux = model;
191     model = glm::scale(model,
192 glm::vec3(0.2f, 0.2f, 0.2f));
193     glUniformMatrix4fv(
194 uniformModel, 1, GL_FALSE, glm::
195 value_ptr(model));
196     sp.render();
197     model = modelaux;
198
199     modelaux = model;
200     model = glm::rotate(model,
201 glm::radians(mainWindow.
202 getarticulacion10()), glm::vec3
203 (0.0f, 1.0f, 0.0f));
204     model = glm::rotate(model,
205 glm::radians(-30.0f), glm::vec3
206 (0.0f, 1.0f, 0.0f));
207     model = glm::rotate(model,
208 glm::radians(60.0f), glm::vec3
209 (1.0f, 0.0f, 0.0f));
210     model = glm::translate(
211 model, glm::vec3(0.05f, -2.3f,
212 0.8f));
213     model = glm::rotate(model,

```

```

186     glm::radians(-75.0f), glm::vec3
187     (1.0f, 0.0f, 0.0f));
188     model = glm::rotate(model,
189     glm::radians(145.0f), glm::vec3
190     (1.0f, 0.0f, 0.0f));
191     model = glm::scale(model,
192     glm::vec3(0.1f, 0.1f, 5.0f));
193     glUniformMatrix4fv(
194     uniformModel, 1, GL_FALSE, glm::
195     value_ptr(model));
196     meshList[0]->RenderMesh();
197     //dibuja cubo, pirámide
198     triangular, pirámide base
199     cuadrangular
200
201     model = modelaux2;
202 }
203
204 // Pierna derecha trasera
205 {
206     modelaux3 = model;
207     model = glm::rotate(model,
208     glm::radians(mainWindow.
209     getarticulacion11()), glm::vec3
210     (0.0f, 1.0f, 0.0f));
211     model = glm::translate(
212     model, glm::vec3(0.0f, 1.3f, 1.8
213     f));
214     model = glm::rotate(model,
215     glm::radians(-60.0f), glm::vec3
216     (1.0f, 0.0f, 0.0f));
217     model = glm::rotate(model,
218     glm::radians(20.0f), glm::vec3
219     (0.0f, 1.0f, 0.0f));
220     modelaux = model;
221     model = glm::scale(model,
222     glm::vec3(0.1f, 0.1f, 3.0f));
223     glUniformMatrix4fv(
224     uniformModel, 1, GL_FALSE, glm::
225     value_ptr(model));
226     meshList[0]->RenderMesh();
227     //dibuja cubo, pirámide
228     triangular, pirámide base
229     cuadrangular
230
231     model = modelaux;
232
233     model = glm::translate(
234     model, glm::vec3(0.0f, 0.0f, 1.5
235     f));

```

```

212     modelaux = model;
213     model = glm::scale(model,
214     glm::vec3(0.2f, 0.2f, 0.2f));
215     glUniformMatrix4fv(
216     uniformModel, 1, GL_FALSE, glm::
217     value_ptr(model));
218     sp.render();
219     model = modelaux;
220
221     modelaux = model;
222     model = glm::rotate(model,
223     glm::radians(mainWindow.
224     getarticulacion12()), glm::vec3
225     (0.0f, 1.0f, 0.0f));
226     model = glm::rotate(model,
227     glm::radians(-20.0f), glm::vec3
228     (0.0f, 1.0f, 0.0f));
229     model = glm::rotate(model,
230     glm::radians(60.0f), glm::vec3
231     (1.0f, 0.0f, 0.0f));
232     model = glm::translate(
233     model, glm::vec3(0.05f, -2.3f,
234     0.8f));
235     model = glm::rotate(model,
236     glm::radians(-75.0f), glm::vec3
237     (1.0f, 0.0f, 0.0f));
238     model = glm::rotate(model,
239     glm::radians(145.0f), glm::vec3
240     (1.0f, 0.0f, 0.0f));
241     model = glm::scale(model,
242     glm::vec3(0.1f, 0.1f, 5.0f));
243     glUniformMatrix4fv(
244     uniformModel, 1, GL_FALSE, glm::
245     value_ptr(model));
246     meshList[0]->RenderMesh();
247     //dibuja cubo, pirámide
248     triangular, pirámide base
249     cuadrangular
250
251     model = modelaux1;
252 }

```

## 2. PROBLEMAS PRESENTADOS

Uno de los problemas presentados fue la rotación de las articulaciones en cada uno de los ejercicios, ya que si bien se pudo comprender cómo es que funcionan las rotaciones al modificar la matriz del modelo, aún es complicado orientarse cuando el eje cambia según las traslaciones, escalamientos y rotaciones de cada parte del modelo.

### 3. CONCLUSIÓN

Dada a consumación de las actividades que integran esta práctica fue posible comprender la importancia del uso del modelado jerárquico, ya que este auxilio con el posicionamiento de varios objetos del modelo, haciendo que en vez de calcular una nueva posición, solo se tuviera que mover en un solo eje a partir de la ultima posición del último objeto dibujado.

De igual manera, el tener en mente un diagrama jerárquico ayudó a comprender con que objeto se empezaría a dibujar el modelo y de esta manera tomar en cuenta la jerarquía de cada uno de los objetos para poder completar un modelo que rote y se mueva de manera correcta.

Finalmente, en cuánto a comentarios se refiere, para esta práctica quedó bastante más claro cómo es que funciona el código, tanto en cuestión de como se mueven los objetos en el espacio como de manera técnica, de modo que se evitaron líneas de código que solo generaban más carga de la necesaria

#### ■ Referencias