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: Romero Dominguez Ricardo Damián

Nº de Cuenta: 319094493

GRUPO DE LABORATORIO: 02

GRUPO DE TEORÍA: 04

SEMESTRE 2025-2

FECHA DE ENTREGA LÍMITE: 03/09/2025

CALIFICACIÓN: _____

Índice

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.

```
// Crear Cabina
{
  model = glm::mat4(1.0);
  model = glm::translate(model, glm
   ::vec3(0.0f, 5.0f, -4.0f));
  modelaux = model;
  model = glm::scale(model, glm::
   vec3(5.0f, 5.0f, 5.0f));
  glUniformMatrix4fv(
   uniformProjection, 1, GL_FALSE,
   glm::value_ptr(projection));
  glUniformMatrix4fv(uniformView,
   1, GL_FALSE, glm::value_ptr(
   camera.calculateViewMatrix()));
  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
  color = glm::vec3(1.0f, 0.0f, 0.0
   f);
  glUniform3fv(uniformColor, 1, glm
   ::value_ptr(color)); //para
   cambiar el color del objetos
  meshList[0]->RenderMesh(); //
   dibuja cubo, pirámide triangular
   , pirámide base cuadrangular
}

```

1.1.2. Creación de la base

Posteriormente creamos la base con ayuda de los cambios realizados en el dibujo (escalamiento y rotación), 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 recalcular ya que se hicieron en el paso anterior

Código 2. Creación de base.

```
// Crear Base
{
  model = glm::scale(model, glm::
   vec3(1.0f, 0.5f, 1.0f));
  model = glm::translate(model, glm
   ::vec3(0.0f, -1.0f, 0.0f));
  glUniformMatrix4fv(uniformModel,
   1, GL_FALSE, glm::value_ptr(
   model));
  color = glm::vec3(0.2f, 0.0f, 0.7
   f);
  glUniform3fv(uniformColor, 1, glm
   ::value_ptr(color));
  meshList[4]->RenderMeshGeometry()
   ;
  model = modelaux;
}

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.

```
model = glm::translate(model, glm::
   vec3(-1.9f, -4.0f, 3.0f));
model = glm::rotate(model, glm::
   radians(90.0f), glm::vec3(1.0f,
   0.0f, 0.0f));
model = glm::scale(model, glm::vec3
   (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.

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

```

Para el dibujo de las demas ruedas solo se desplazaron y su rotaciónes 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.

```
  // Rueda izquierda trasera
  model = glm::translate(model, glm
   ::vec3(2.1f, 0.0f, 0.0f));

  modelaux = model;
  model = glm::rotate(model, glm::
   radians(mainWindow.getNeumatico2
   ()), glm::vec3(0.0f, 1.0f, 0.0f)
   );
  glUniformMatrix4fv(uniformModel,
   1, GL_FALSE, glm::value_ptr(
   model));
  meshList[2]->RenderMeshGeometry()
   ;
  model = modelaux;


  // Rueda derecha trasera
  model = glm::translate(model, glm
   ::vec3(0.0f, -4.5f, 0.0f));

  modelaux = model;
  model = glm::rotate(model, glm::
   radians(mainWindow.getNeumatico3
   ()), glm::vec3(0.0f, 1.0f, 0.0f)
   );
  glUniformMatrix4fv(uniformModel,
   1, GL_FALSE, glm::value_ptr(
   model));
  meshList[2]->RenderMeshGeometry()
   ;
  model = modelaux;

  // Rueda derecha frontal
  model = glm::translate(model, glm
   ::vec3(-2.1f, 0.0f, 0.0f));

  modelaux = model;
  model = glm::rotate(model, glm::
   radians(mainWindow.getNeumatico4
   ()), glm::vec3(0.0f, 1.0f, 0.0f)
   );
  glUniformMatrix4fv(uniformModel,
   1, GL_FALSE, glm::value_ptr(
   model));
  meshList[2]->RenderMeshGeometry()
   ;

}

model = modelauxRuedas;

```

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 articulaciónes

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

5. Volver al primer paso para la siguiente articulación

Código 6. Brazo y Jaula.

```
// Articulaciónes
{
  // Articulación 1
  model = glm::translate(model, glm::vec3(0.0f, 1.0f, 0.0f));
  //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 línea de proyección solo se manda una vez a menos que en tiempo de ejecución
  //se programe cambio entre proyección ortogonal y perspectiva
  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 pequena esfera
  model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
  glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
  sp.render();

  model = modelaux;
  //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));
52  modelaux = model;
53
54
55  //dibujar una pequena esfera
56  model = glm::scale(model, glm::
     vec3(0.5f, 0.5f, 0.5f));
57  glUniformMatrix4fv(uniformModel,
     1, GL_FALSE, glm::value_ptr(
     model));
58  sp.render();
59
60  // Crear instancias para
     completar el brazo y  la cabina.
      Imporante considerar que la
     cabina es el nodo padre.
61  //La cabina y el brazo deben de
     estar unidos a la cabina
62
63  model = modelaux;
64  //tercer brazo
65  model = glm::translate(model, glm
     ::vec3(2.5f, 0.0f, 0.0f));
66  model = glm::rotate(model, glm::
     radians(90.0f), glm::vec3(0.0f,
     0.0f, 1.0f));
67  modelaux = model;
68  model = glm::scale(model, glm::
     vec3(1.0f, 5.0f, 1.0f));
69  glUniformMatrix4fv(uniformModel,
     1, GL_FALSE, glm::value_ptr(
     model));
70  color = glm::vec3(0.0f, 1.0f, 5.0
     f);
71  glUniform3fv(uniformColor, 1, glm
     ::value_ptr(color)); //para
     cambiar el color del objetos
72  meshList[0]->RenderMesh(); //
     dibuja cubo y pirámide
     triangular
73
74  model = modelaux;
75
76  //articulación 4 extremo derecho
     del segundo brazo
77  model = glm::translate(model, glm
     ::vec3(0.0f, -2.5f, 0.0f));
78  model = glm::rotate(model, glm::
     radians(-135.0f), glm::vec3(0.0f
     , 0.0f, 1.0f));
79  model = glm::rotate(model, glm::
     radians(mainWindow.
     getarticulacion4()), glm::vec3
     (1.0f, 0.0f, 0.0f));
80  modelaux = model;
81
82  //dibujar una pequena esfera
83  model = glm::scale(model, glm::
     vec3(0.5f, 0.5f, 0.5f));
84  glUniformMatrix4fv(uniformModel,
     1, GL_FALSE, glm::value_ptr(
     model));
85  sp.render();
86
87  model = modelaux;
88  //Jaula
89  //model = glm::translate(model,
     glm::vec3(2.5f, 0.0f, 0.0f));
90  modelaux = model;
91  model = glm::scale(model, glm::
     vec3(5.0f, 3.0f, 3.0f));
92  glUniformMatrix4fv(uniformModel,
     1, GL_FALSE, glm::value_ptr(
     model));
93  color = glm::vec3(0.5f, 0.5f, 0.0
     f);
94  glUniform3fv(uniformColor, 1, glm
     ::value_ptr(color)); //para
     cambiar el color del objetos
95  meshList[0]->RenderMesh(); //
     dibuja cubo y pirámide
     triangular
96
97  model = modelaux;
98
99  //articulación 4 extremo derecho
     del segundo brazo
100 model = glm::translate(model, glm
     ::vec3(0.0f, -2.5f, 0.0f));
101 model = glm::rotate(model, glm::
     radians(mainWindow.
     getarticulacion5()), glm::vec3
     (0.0f, 0.0f, 1.0f));
102 modelaux = model;
103 }
104
105
```

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.

```
// Cuerpo
{
  model = glm::mat4(1.0);
  model = glm::translate(model, glm
   ::vec3(0.0f, 5.0f, -4.0f));
  modelaux = model;
  model = glm::scale(model, glm::
   vec3(2.0f, 1.5f, 1.5f));
  glUniformMatrix4fv(
   uniformProjection, 1, GL_FALSE,
   glm::value_ptr(projection));
  glUniformMatrix4fv(uniformView,
   1, GL_FALSE, glm::value_ptr(
   camera.calculateViewMatrix()));
  glUniformMatrix4fv(uniformModel,
   1, GL_FALSE, glm::value_ptr(
   model));
  color = glm::vec3(0.0f, 0.1f, 0.1
   f);
  glUniform3fv(uniformColor, 1, glm
   ::value_ptr(color));
  sp.render();
  model = modelaux;
}

```

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.

```
// Cabeza
modelaux = model;
model = glm::translate(model, glm::
   vec3(-2.2f, 0.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1,
    GL_FALSE, glm::value_ptr(model)
   );
sp.render();


// Ojo derecho inferior
model = glm::translate(model, glm::
   vec3(-0.85f, 0.2f, -0.25f));
model = glm::scale(model, glm::vec3
   (0.2f, 0.2f, 0.2f));
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
sp.render();

// Ojo derecho medio
model = glm::translate(model, glm::
   vec3(0.5f, 1.0f, -0.8f));
glUniformMatrix4fv(uniformModel, 1,
    GL_FALSE, glm::value_ptr(model)
   );
sp.render();

// Ojo derecho superior
model = glm::translate(model, glm::
   vec3(0.5f, 1.0f, -0.8f));
glUniformMatrix4fv(uniformModel, 1,
    GL_FALSE, glm::value_ptr(model)
   );
sp.render();

model = modelaux;


modelaux = model;
model = glm::translate(model, glm::
```

```
    vec3(-2.2f, 0.0f, 0.0f));
model = glm::rotate(model, glm::
    radians(15.0f), glm::vec3(0.0f,
    1.0f, 0.0f));
model = glm::rotate(model, glm::
    radians(80.0f), glm::vec3(1.0f,
    0.0f, 0.0f));
// Ojo derecho inferior
model = glm::translate(model, glm::
    vec3(-0.85f, 0.2f, -0.25f));
model = glm::scale(model, glm::vec3
    (0.2f, 0.2f, 0.2f));
glUniformMatrix4fv(uniformModel, 1,
     GL_FALSE, glm::value_ptr(model)
    );
sp.render();

// Ojo derecho medio
model = glm::translate(model, glm::
    vec3(0.5f, 1.0f, -0.8f));
glUniformMatrix4fv(uniformModel, 1,
     GL_FALSE, glm::value_ptr(model)
    );
sp.render();

// Ojo derecho superior
model = glm::translate(model, glm::
    vec3(0.5f, 1.0f, -0.8f));
glUniformMatrix4fv(uniformModel, 1,
     GL_FALSE, glm::value_ptr(model)
    );
sp.render();

model = modelaux;

```

1.2.3. Creaciónde 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 circulo 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.

```
// Pierna izquierda frontal
{
        modelaux1 = model;
        model = glm::rotate(model,
    glm::radians(mainWindow.
    getarticulacion1()), glm::vec3
    (0.0f, 1.0f, 0.0f));
        model = glm::translate(
    model, glm::vec3(-2.0f, 1.3f,
    1.5f));
        model = glm::rotate(model,
    glm::radians(-60.0f), glm::vec3
    (1.0f, 0.0f, 0.0f));
        model = glm::rotate(model,
    glm::radians(-40.0f), glm::vec3
    (0.0f, 1.0f, 0.0f));
        modelaux = model;
        model = glm::scale(model,
    glm::vec3(0.1f, 0.1f, 3.0f));
        glUniformMatrix4fv(
    uniformModel, 1, GL_FALSE, glm::
    value_ptr(model));
        color = glm::vec3(0.0f, 0.1
    f, 0.1f);
        glUniform3fv(uniformColor,
    1, glm::value_ptr(color));
        meshList[0]->RenderMesh();
    //dibuja cubo, pirámide
    triangular, pirámide base
    cuadrangular

        model = modelaux;


        model = glm::translate(
    model, glm::vec3(0.0f, 0.0f, 1.5
    f));
        modelaux = model;
        model = glm::scale(model,
    glm::vec3(0.2f, 0.2f, 0.2f));
        glUniformMatrix4fv(
    uniformModel, 1, GL_FALSE, glm::
    value_ptr(model));
        sp.render();
        model = modelaux;

        modelaux = model;
        model = glm::rotate(model,
    glm::radians(mainWindow.
    getarticulacion2()), glm::vec3
    (0.0f, 1.0f, 0.0f));
        model = glm::rotate(model,
    glm::radians(40.0f), glm::vec3
    (0.0f, 1.0f, 0.0f));
        model = glm::rotate(model,
    glm::radians(60.0f), glm::vec3
```

```
    (1.0f, 0.0f, 0.0f));
        model = glm::translate(model, glm::vec3(0.05f, -2.3f, 0.8f));
        model = glm::rotate(model, glm::radians(-75.0f), glm::vec3(1.0f, 0.0f, 0.0f));
        model = glm::rotate(model, glm::radians(145.0f), glm::vec3(1.0f, 0.0f, 0.0f));
        model = glm::scale(model, glm::vec3(0.1f, 0.1f, 5.0f));
        glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
        meshList[0]->RenderMesh(); //dibuja cubo, pirámide triangular, pirámide base cuadrangular

        model = modelaux1;
}


// Pierna izquierda intermedia
{
        modelaux2 = model;
        model = glm::rotate(model, glm::radians(mainWindow.getarticulacion3()), glm::vec3(0.0f, 1.0f, 0.0f));
        model = glm::translate(model, glm::vec3(-1.0f, 1.3f, 1.8f));
        model = glm::rotate(model, glm::radians(-60.0f), glm::vec3(1.0f, 0.0f, 0.0f));
        model = glm::rotate(model, glm::radians(-30.0f), glm::vec3(0.0f, 1.0f, 0.0f));
        modelaux = model;
        model = glm::scale(model, glm::vec3(0.1f, 0.1f, 3.0f));
        glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
        meshList[0]->RenderMesh(); //dibuja cubo, pirámide triangular, pirámide base cuadrangular

        model = modelaux;


        model = glm::translate(model, glm::vec3(0.0f, 0.0f, 1.5f));
        modelaux = model;
        model = glm::scale(model, glm::vec3(0.2f, 0.2f, 0.2f));
        glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
        sp.render();
        model = modelaux;

        modelaux = model;
        model = glm::rotate(model, glm::radians(mainWindow.getarticulacion4()), glm::vec3(0.0f, 1.0f, 0.0f));
        model = glm::rotate(model, glm::radians(30.0f), glm::vec3(0.0f, 1.0f, 0.0f));
        model = glm::rotate(model, glm::radians(60.0f), glm::vec3(1.0f, 0.0f, 0.0f));
        model = glm::translate(model, glm::vec3(0.05f, -2.3f, 0.8f));
        model = glm::rotate(model, glm::radians(-75.0f), glm::vec3(1.0f, 0.0f, 0.0f));
        model = glm::rotate(model, glm::radians(145.0f), glm::vec3(1.0f, 0.0f, 0.0f));
        model = glm::scale(model, glm::vec3(0.1f, 0.1f, 5.0f));
        glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
        meshList[0]->RenderMesh(); //dibuja cubo, pirámide triangular, pirámide base cuadrangular

        model = modelaux2;
}


// Pierna izquierda trasera
{
```

```
        modelaux3 = model;
        model = glm::rotate(model, glm::radians(mainWindow.getarticulacion5()), glm::vec3(0.0f, 1.0f, 0.0f));
        model = glm::translate(model, glm::vec3(0.0f, 1.3f, 1.8f));
        model = glm::rotate(model, glm::radians(-60.0f), glm::vec3(1.0f, 0.0f, 0.0f));
        model = glm::rotate(model, glm::radians(-20.0f), glm::vec3(0.0f, 1.0f, 0.0f));
        modelaux = model;
        model = glm::scale(model, glm::vec3(0.1f, 0.1f, 3.0f));
        glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
        meshList[0]->RenderMesh(); //dibuja cubo, pirámide triangular, pirámide base cuadrangular

        model = modelaux;


        model = glm::translate(model, glm::vec3(0.0f, 0.0f, 1.5f));
        modelaux = model;
        model = glm::scale(model, glm::vec3(0.2f, 0.2f, 0.2f));
        glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
        sp.render();
        model = modelaux;

        modelaux = model;
        model = glm::rotate(model, glm::radians(mainWindow.getarticulacion6()), glm::vec3(0.0f, 1.0f, 0.0f));
        model = glm::rotate(model, glm::radians(20.0f), glm::vec3(0.0f, 1.0f, 0.0f));
        model = glm::rotate(model, glm::radians(60.0f), glm::vec3(1.0f, 0.0f, 0.0f));
        model = glm::translate(model, glm::vec3(0.05f, -2.3f, 0.8f));
        model = glm::rotate(model, glm::radians(-75.0f), glm::vec3(1.0f, 0.0f, 0.0f));
        model = glm::rotate(model, glm::radians(145.0f), glm::vec3(1.0f, 0.0f, 0.0f));
        model = glm::scale(model, glm::vec3(0.1f, 0.1f, 5.0f));
        glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
        meshList[0]->RenderMesh(); //dibuja cubo, pirámide triangular, pirámide base cuadrangular


        model = modelaux1;
}

model = glm::rotate(model, glm::radians(-180.0f), glm::vec3(0.0f, 1.0f, 0.0f));

// Pierna derecha frontal
{
        modelaux1 = model;
        model = glm::rotate(model, glm::radians(mainWindow.getarticulacion7()), glm::vec3(0.0f, 1.0f, 0.0f));
        model = glm::translate(model, glm::vec3(2.0f, 1.3f, 1.5f));

        model = glm::rotate(model, glm::radians(-60.0f), glm::vec3(1.0f, 0.0f, 0.0f));
        model = glm::rotate(model, glm::radians(40.0f), glm::vec3(0.0f, 1.0f, 0.0f));
        modelaux = model;
        model = glm::scale(model, glm::vec3(0.1f, 0.1f, 3.0f));
        glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
        meshList[0]->RenderMesh(); //dibuja cubo, pirámide triangular, pirámide base cuadrangular

```

```
133         model = modelaux;
134
135
136         model = glm::translate(
    model, glm::vec3(0.0f, 0.0f, 1.5
    f));
137         modelaux = model;
138         model = glm::scale(model,
    glm::vec3(0.2f, 0.2f, 0.2f));
139         glUniformMatrix4fv(
    uniformModel, 1, GL_FALSE, glm::
    value_ptr(model));
140         sp.render();
141         model = modelaux;
142
143         modelaux = model;
144         model = glm::rotate(model,
    glm::radians(mainWindow.
    getarticulacion8()), glm::vec3
    (0.0f, 1.0f, 0.0f));
145         model = glm::rotate(model,
    glm::radians(-40.0f), glm::vec3
    (0.0f, 1.0f, 0.0f));
146         model = glm::rotate(model,
    glm::radians(60.0f), glm::vec3
    (1.0f, 0.0f, 0.0f));
147         model = glm::translate(
    model, glm::vec3(0.05f, -2.3f,
    0.8f));
148         model = glm::rotate(model,
    glm::radians(-75.0f), glm::vec3
    (1.0f, 0.0f, 0.0f));
149         model = glm::rotate(model,
    glm::radians(145.0f), glm::vec3
    (1.0f, 0.0f, 0.0f));
150         model = glm::scale(model,
    glm::vec3(0.1f, 0.1f, 5.0f));
151         glUniformMatrix4fv(
    uniformModel, 1, GL_FALSE, glm::
    value_ptr(model));
152         meshList[0]->RenderMesh();
    //dibuja cubo, pirámide
    triangular, pirámide base
    cuadrangular
153
154
155         model = modelaux1;
156 }
157
158 // Pierna derecha intermedia
159 {
160         modelaux2 = model;
161         model = glm::rotate(model,
    glm::radians(mainWindow.
    getarticulacion9()), glm::vec3
    (0.0f, 1.0f, 0.0f));
162         model = glm::translate(
    model, glm::vec3(1.0f, 1.3f, 1.8
    f));
163         model = glm::rotate(model,
    glm::radians(-60.0f), glm::vec3
    (1.0f, 0.0f, 0.0f));
164         model = glm::rotate(model,
    glm::radians(30.0f), glm::vec3
    (0.0f, 1.0f, 0.0f));
165         modelaux = model;
166         model = glm::scale(model,
    glm::vec3(0.1f, 0.1f, 3.0f));
167         glUniformMatrix4fv(
    uniformModel, 1, GL_FALSE, glm::
    value_ptr(model));
168         meshList[0]->RenderMesh();
    //dibuja cubo, pirámide
    triangular, pirámide base
    cuadrangular
169
170         model = modelaux;
171
172
173         model = glm::translate(
    model, glm::vec3(0.0f, 0.0f, 1.5
    f));
174         modelaux = model;
175         model = glm::scale(model,
    glm::vec3(0.2f, 0.2f, 0.2f));
176         glUniformMatrix4fv(
    uniformModel, 1, GL_FALSE, glm::
    value_ptr(model));
177         sp.render();
178         model = modelaux;
179
180         modelaux = model;
181         model = glm::rotate(model,
    glm::radians(mainWindow.
    getarticulacion10()), glm::vec3
    (0.0f, 1.0f, 0.0f));
182         model = glm::rotate(model,
    glm::radians(-30.0f), glm::vec3
    (0.0f, 1.0f, 0.0f));
183         model = glm::rotate(model,
    glm::radians(60.0f), glm::vec3
    (1.0f, 0.0f, 0.0f));
184         model = glm::translate(
    model, glm::vec3(0.05f, -2.3f,
    0.8f));
185         model = glm::rotate(model,
```

```
    glm::radians(-75.0f), glm::vec3
    (1.0f, 0.0f, 0.0f));
186         model = glm::rotate(model,
    glm::radians(145.0f), glm::vec3
    (1.0f, 0.0f, 0.0f));
187         model = glm::scale(model,
    glm::vec3(0.1f, 0.1f, 5.0f));
188         glUniformMatrix4fv(
    uniformModel, 1, GL_FALSE, glm::
    value_ptr(model));
189         meshList[0]->RenderMesh();
    //dibuja cubo, pirámide
    triangular, pirámide base
    cuadrangular
190
191
192         model = modelaux2;
193 }
194
195
196 // Pierna derecha trasera
197 {
198         modelaux3 = model;
199         model = glm::rotate(model,
    glm::radians(mainWindow.
    getarticulacion11()), glm::vec3
    (0.0f, 1.0f, 0.0f));
200         model = glm::translate(
    model, glm::vec3(0.0f, 1.3f, 1.8
    f));
201         model = glm::rotate(model,
    glm::radians(-60.0f), glm::vec3
    (1.0f, 0.0f, 0.0f));
202         model = glm::rotate(model,
    glm::radians(20.0f), glm::vec3
    (0.0f, 1.0f, 0.0f));
203         modelaux = model;
204         model = glm::scale(model,
    glm::vec3(0.1f, 0.1f, 3.0f));
205         glUniformMatrix4fv(
    uniformModel, 1, GL_FALSE, glm::
    value_ptr(model));
206         meshList[0]->RenderMesh();
    //dibuja cubo, pirámide
    triangular, pirámide base
    cuadrangular
207
208         model = modelaux;
209
210
211         model = glm::translate(
    model, glm::vec3(0.0f, 0.0f, 1.5
    f));
```

```
212         modelaux = model;
213         model = glm::scale(model,
    glm::vec3(0.2f, 0.2f, 0.2f));
214         glUniformMatrix4fv(
    uniformModel, 1, GL_FALSE, glm::
    value_ptr(model));
215         sp.render();
216         model = modelaux;
217
218         modelaux = model;
219         model = glm::rotate(model,
    glm::radians(mainWindow.
    getarticulacion12()), glm::vec3
    (0.0f, 1.0f, 0.0f));
220         model = glm::rotate(model,
    glm::radians(-20.0f), glm::vec3
    (0.0f, 1.0f, 0.0f));
221         model = glm::rotate(model,
    glm::radians(60.0f), glm::vec3
    (1.0f, 0.0f, 0.0f));
222         model = glm::translate(
    model, glm::vec3(0.05f, -2.3f,
    0.8f));
223         model = glm::rotate(model,
    glm::radians(-75.0f), glm::vec3
    (1.0f, 0.0f, 0.0f));
224         model = glm::rotate(model,
    glm::radians(145.0f), glm::vec3
    (1.0f, 0.0f, 0.0f));
225         model = glm::scale(model,
    glm::vec3(0.1f, 0.1f, 5.0f));
226         glUniformMatrix4fv(
    uniformModel, 1, GL_FALSE, glm::
    value_ptr(model));
227         meshList[0]->RenderMesh();
    //dibuja cubo, pirámide
    triangular, pirámide base
    cuadrangular
228
229
230         model = modelaux1;
231 }
232
```

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