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: Tapia Ledesma Angel Hazel

Nº de Cuenta: 320070358

GRUPO DE LABORATORIO: 02

GRUPO DE TEORÍA: 02

SEMESTRE 2026-1

FECHA DE ENTREGA LÍMITE: 21-09-25

CALIFICACIÓN: _____

Actividades realizadas

1.- Terminar la Grúa con:
-cuerpo (prisma rectangular)
-base (pirámide cuadrangular)
-4 llantas (4 cilindros) con teclado se pueden girar cada una de las 4 llantas por separado
2.- 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

Descripción de las actividades

Para el desarrollo de la primera actividad, nos basamos en el ejercicio de la práctica, por lo que lo único restante para completar la grúa era el soporte de la base y las 4 llantas. Para el soporte de la base, utilizamos una pirámide cuadrangular, ocultando su punta dentro de la cabina de la grúa, en este caso, la jerarquía que se partió del modelo de la cabina, más que nada su posición, y se heredó a esta base, y lo único que se hizo fue "bajarla" un poco para que quedara un poco de la pirámide fuera, y diera este efecto de ocultar la punta en la cabina y que solo se viera parte de la base.

Para el caso de las llantas, se usó la jerarquía mediante el modelo del soporte, de igual modo más que nada su posición, la cual se heredó a las llantas, en el caso de las llantas, primero realizamos sus articulaciones, para ellas lo que se hizo fue rotarlas 90°, en principio, esta rotación no parecería tan lógica, porque para la articulación la vemos cómo una esfera, pero es muy útil a la hora de heredar esto a las llantas, ya que las llantas son cilindros rotados, por lo que esto nos ayuda mucho para ahorrar la rotación en cada llanta, con este cambio lo que se hizo fue actualizar el modelaux2, para que se heredara esta propiedad de la rotación, posteriormente, lo que se hizo fue adaptar la posición de la articulación al punto de la llanta, y asignarle una rotación mediante una tecla, en este caso fue al rededor del eje Y, ya que cómo está rotada, rotar sobre el eje Y la figura, da la impresión de que la llanta gira hacia adelante. Con esto acabamos la articulación en sí, ya que el resto es escalarlo, su color y mostrarlo mediante una esfera. A continuación, lo que se hizo para generar la llanta es aprovechar este modelo y simplemente dibujar el cilindro y escalarlo, ya que la posición y la rotación ya las tenía por el modelo de la articulación. Este proceso se realizó con las 4 llantas, lo único que cambiaba era su posición con respecto a la base de la grúa y la tecla con la que rotábamos la llanta.

Para la actividad 2 lo primero que se hizo fue seleccionar un animal, en este caso un puerquito porque en el proyecto iba a usar uno, lo que se realizó fue separar el cuerpo en geometrías más simples, en este caso, se realizó la cabeza, el cuerpo, las patas y

la cola. Se usó cómo punto “ancla” o principal el cuerpo y de ahí se partió la construcción del animal. Lo que se hizo después es ver dónde había articulaciones, en este caso se detectaron articulaciones en las patas, en 2 puntos, en donde se une la pata con el cuerpo y el de la “rodilla” del puerco, también se seleccionó la articulación de la cola y la de las orejas.

Cómo se mencionó, se partió del cuerpo, para este sencillamente escalamos la figura y la trasladamos para ajustarla a las dimensiones que queríamos, y de esta traslación, guardamos la matriz para después heredar esta posición a los demás elementos. Una vez acabado esto, seguimos con la cabeza, la cual se movió y se escaló, con respecto a la matriz que guardamos anteriormente, cuyo punto central era el punto del centro del cuerpo, posteriormente, guardamos ahora en otra matriz esta matriz de modelo, ya que tanto los ojos, cómo la nariz y las orejas, van a requerir la posición de la cabeza, por lo que por esto guardamos esta matriz, para esta jerarquía, y no perder la matriz del cuerpo, porque posteriormente será usada para las patas.

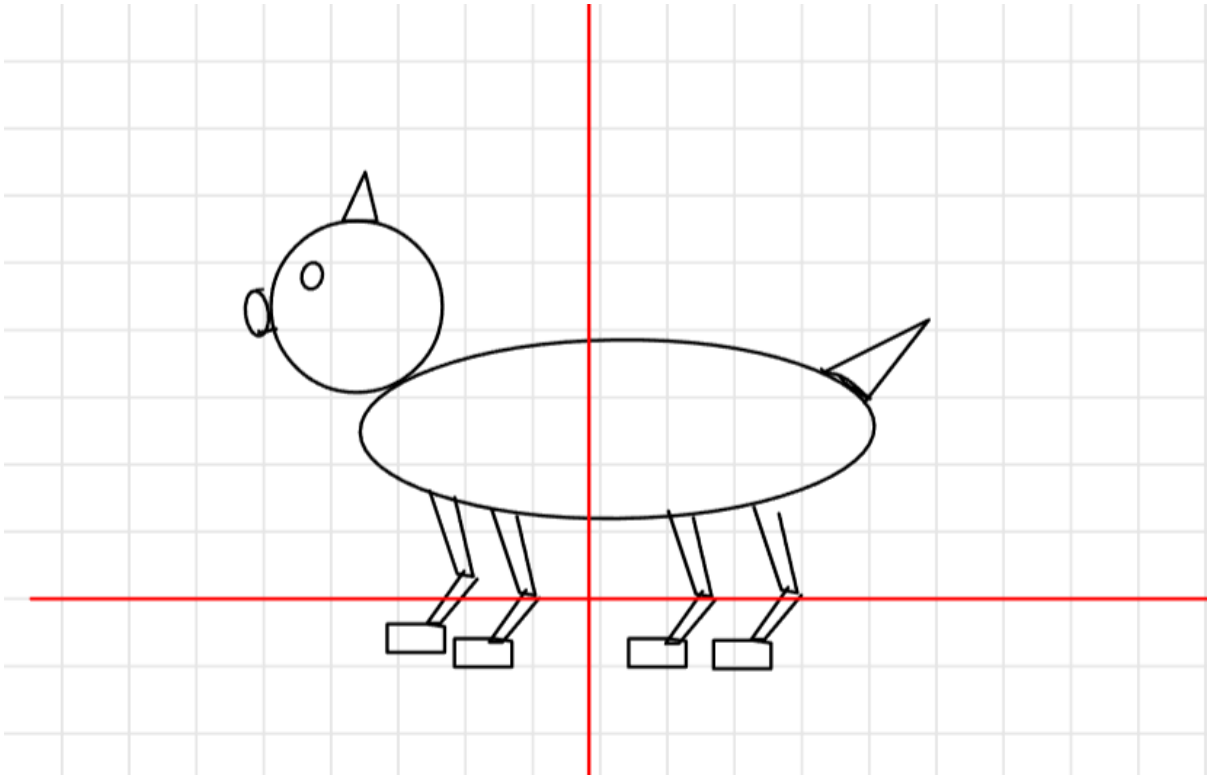
Una vez que tomamos la cabeza como “padre”, posicionamos los elementos, en este caso para los ojos solo hubo traslación, mientras que, para la nariz, si la rotamos, ya que es un cilindro rotado alrededor de z. Finalmente tenemos las orejas, las cuales se trasladaron y se rotaron, además de que para cada una se creó su articulación respectiva para dar el giro.

Cuando terminamos con la cabeza, volvimos a la matriz de modelo que almacenaba el cuerpo, y de aquí partimos para la construcción de las patas, en este caso, hubo tanta traslación cómo rotación, ya que las patas el puerquito no son totalmente rectas, así que se hicieron 2 rotaciones y 2 articulaciones, la del “muslo” con el cuerpo y la del “muslo” con el resto de la pata, poniendo la articulación en la unión de estas zonas.

Finalmente, realizamos con base en la matriz que almacenaba la posición del cuerpo, la cola, para esto, lo que hicimos fue trasladar y generar un punto de articulación, y sobre ese punto, poner la cola.

Esto para la parte del modelado, ya que, para ambos casos, tanto la actividad 1 como 2, tuvimos que agregar las teclas para las rotaciones adicionales, en este caso se ocuparon las teclas

F G H J K L ; ’ U I O P



Código generado:

Actividad 1:

```
// Creando la base
model = modelaux;
model = glm::translate(model, glm::vec3(0.0f, -1.5f, 0.0f));
modelaux2 = model;
model = glm::scale(model, glm::vec3(5.0f, 3.0f, 2.0f));
color = glm::vec3(0.7f, 0.7f, 0.7f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[4]->RenderMesh();
```

```
// Creando las llantas
// Articulación delantera izquierda
model = modelaux2;
model = glm::rotate(model, glm::radians(90.0f), glm::vec3(1.0f, 0.0f, 0.0f));
modelaux2 = model;
model = glm::translate(model, glm::vec3(-2.5f, 1.5f, 1.6f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion5()), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::scale(model, glm::vec3(1.0, 1.0f, 1.0f));
color = glm::vec3(0.4f, 0.4f, 0.4f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
sp.render();

// Llanta Delantera izquierda
model = glm::scale(model, glm::vec3(1.5f, 1.0f, 1.5f));
color = glm::vec3(0.3f, 0.3f, 0.3f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[2]->RenderMeshGeometry();

// Articulacion delantera derecha
model = modelaux2;
modelaux2 = model;
model = glm::translate(model, glm::vec3(-2.5f, -1.5f, 1.6f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion6()), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::scale(model, glm::vec3(1.0, 1.0f, 1.0f));
color = glm::vec3(0.4f, 0.4f, 0.4f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
sp.render();

// Llanta Delantera derecha
model = glm::scale(model, glm::vec3(1.5f, 1.0f, 1.5f));
color = glm::vec3(0.3f, 0.3f, 0.3f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[2]->RenderMeshGeometry();

// Articulacion trasera izquierda
model = modelaux2;
modelaux2 = model;
model = glm::translate(model, glm::vec3(2.5f, 1.5f, 1.6f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion7()), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::scale(model, glm::vec3(1.0, 1.0f, 1.0f));
color = glm::vec3(0.4f, 0.4f, 0.4f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
sp.render();
```

```
// Llanta trasera izquierda
model = glm::scale(model, glm::vec3(1.5f, 1.0f, 1.5f));
color = glm::vec3(0.3f, 0.3f, 0.3f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[2]->RenderMeshGeometry();

// Articulacion trasera izquierda
model = modelaux2;
modelaux2 = model;
model = glm::translate(model, glm::vec3(2.5f, -1.5f, 1.6f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion8()), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::scale(model, glm::vec3(1.0, 1.0f, 1.0f));
color = glm::vec3(0.4f, 0.4f, 0.4f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
sp.render();

// Llanta trasera izquierda
model = glm::scale(model, glm::vec3(1.5f, 1.0f, 1.5f));
color = glm::vec3(0.3f, 0.3f, 0.3f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[2]->RenderMeshGeometry();
```

Actividad 2

```
// Animal puerquito
// Cuerpo
model = glm::mat4(1.0f);
model = glm::translate(model, glm::vec3(0.0f, 3.5f, -4.0f));
modelaux = model;
modelaux2 = model;
model = glm::scale(model, glm::vec3(6.0f, 3.0f, 4.0f));
color = glm::vec3(1.0f, 0.71f, 0.75f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
sp.render();

// Articulacion cabeza
model = modelaux2;
model = glm::translate(model, glm::vec3(-5.0f, 2.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
color = glm::vec3(0.6f, 0.6f, 0.6f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
sp.render();

// Cabeza
model = glm::translate(model, glm::vec3(0.0f, 2.0f, 0.0f));
modelaux2 = model;
model = glm::scale(model, glm::vec3(3.0f, 3.0f, 3.0f));
color = glm::vec3(1.0f, 0.71f, 0.75f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
sp.render();

// Nariz
model = modelaux2;
model = glm::translate(model, glm::vec3(-3.0f, 0.8f, 0.0f));
model = glm::rotate(model, glm::radians(90.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::scale(model, glm::vec3(0.5f, 1.0f, 1.0f));
color = glm::vec3(1.0f, 0.41f, 0.70f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[2]->RenderMeshGeometry();

// Ojo derecho
model = modelaux2;
model = glm::translate(model, glm::vec3(-2.0f, 1.2f, -1.0f));
model = glm::rotate(model, glm::radians(90.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::scale(model, glm::vec3(0.8f, 0.8f, 0.8f));
color = glm::vec3(0.0f, 0.0f, 0.0f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
sp.render();
```

```
// Ojo izquierdo
model = modelaux2;
model = glm::translate(model, glm::vec3(-2.0f, 1.2f, 1.0f));
model = glm::rotate(model, glm::radians(90.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::scale(model, glm::vec3(0.8f, 0.8f, 0.8f));
color = glm::vec3(0.0f, 0.0f, 0.0f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
sp.render();

// Articulacion oreja derecha
model = modelaux2;
model = glm::translate(model, glm::vec3(0.0f, 2.5f, -1.5f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.8f, 0.8f, 0.8f));
color = glm::vec3(0.0f, 0.0f, 1.0f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
//sp.render();

//Oreja derecha
model = glm::translate(model, glm::vec3(0.0f, 0.3f, 0.0f));
model = glm::rotate(model, glm::radians(-100.0f), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::scale(model, glm::vec3(2.5f, 3.0f, 2.0f));
color = glm::vec3(1.0f, 0.41f, 0.70f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[1]->RenderMesh();

// Articulacion oreja izquierda
model = modelaux2;
model = glm::translate(model, glm::vec3(0.0f, 2.5f, 1.5f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion3()), glm::vec3(0.0f, -1.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.8f, 0.8f, 0.8f));
color = glm::vec3(0.0f, 0.0f, 1.0f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));

// Oreja izquierda
model = glm::translate(model, glm::vec3(0.0f, 0.3f, 0.0f));
model = glm::rotate(model, glm::radians(-100.0f), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::scale(model, glm::vec3(2.5f, 3.0f, 2.0f));
color = glm::vec3(1.0f, 0.41f, 0.70f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[1]->RenderMesh();
```

```
// Patas
// Articulación pata - cuerpo frontal derecha
model = modelaux;
model = glm::translate(model, glm::vec3(-4.0f, -1.2f, -1.5f));
model = glm::rotate(model, glm::radians(30.0f), glm::vec3(0.0f, 0.0f, 0.75f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion4()), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
color = glm::vec3(1.0f, 0.71f, 0.75f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
//sp.render();

// Muslo frontal derecho
model = glm::translate(model, glm::vec3(0.0f, -1.5f, 0.0f));
modelaux2 = model;
model = glm::scale(model, glm::vec3(0.5f, 3.0f, 1.0f));
color = glm::vec3(1.0f, 0.71f, 0.75f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[2]->RenderMeshGeometry();

// Articulacion rodilla derecha
model = modelaux2;
model = glm::translate(model, glm::vec3(0.0f, -1.5f, 0.0f));
model = glm::rotate(model, glm::radians(-60.0f), glm::vec3(0.0f, 0.0f, 0.75f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion5()), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
color = glm::vec3(1.0f, 0.71f, 0.75f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
//sp.render();

// Pata frontal derecha
model = glm::translate(model, glm::vec3(0.0f, -1.5f, 0.0f));
modelaux2 = model;
model = glm::scale(model, glm::vec3(0.5f, 3.0f, 1.0f));
color = glm::vec3(1.0f, 0.71f, 0.75f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[2]->RenderMeshGeometry();

// Pie frontal derecho
model = modelaux2;
model = glm::translate(model, glm::vec3(-0.75f, -1.5f, 0.0f));
model = glm::rotate(model, glm::radians(30.0f), glm::vec3(0.0f, 0.0f, 0.75f));
model = glm::scale(model, glm::vec3(1.5f, 0.5f, 1.0f));
color = glm::vec3(0.2f, 0.2f, 0.2f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[2]->RenderMeshGeometry();
```

```
// Articulación pata - cuerpo frontal izquierda
model = modelaux;
model = glm::translate(model, glm::vec3(-4.0f, -1.2f, 1.5f));
model = glm::rotate(model, glm::radians(30.0f), glm::vec3(0.0f, 0.0f, 0.75f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion6()), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
color = glm::vec3(1.0f, 0.71f, 0.75f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
//sp.render();

// Muslo frontal izquierdo
model = glm::translate(model, glm::vec3(0.0f, -1.5f, 0.0f));
modelaux2 = model;
model = glm::scale(model, glm::vec3(0.5f, 3.0f, 1.0f));
color = glm::vec3(1.0f, 0.71f, 0.75f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[2]->RenderMeshGeometry();

// Articulacion rodilla frontal izquierda
model = modelaux2;
model = glm::translate(model, glm::vec3(0.0f, -1.5f, 0.0f));
model = glm::rotate(model, glm::radians(-60.0f), glm::vec3(0.0f, 0.0f, 0.75f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion7()), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
color = glm::vec3(1.0f, 0.71f, 0.75f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
//sp.render();

// Pata frontal izquierda
model = glm::translate(model, glm::vec3(0.0f, -1.5f, 0.0f));
modelaux2 = model;
model = glm::scale(model, glm::vec3(0.5f, 3.0f, 1.0f));
color = glm::vec3(1.0f, 0.71f, 0.75f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[2]->RenderMeshGeometry();

// Pie frontal izquierdo
model = modelaux2;
model = glm::translate(model, glm::vec3(-0.75f, -1.5f, 0.0f));
model = glm::rotate(model, glm::radians(30.0f), glm::vec3(0.0f, 0.0f, 0.75f));
model = glm::scale(model, glm::vec3(1.5f, 0.5f, 1.0f));
color = glm::vec3(0.2f, 0.2f, 0.2f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[2]->RenderMeshGeometry();
```

```
// Articulación pata - cuerpo trasera derecha
model = modelaux;
model = glm::translate(model, glm::vec3(4.0f, -1.2f, -1.5f));
model = glm::rotate(model, glm::radians(30.0f), glm::vec3(0.0f, 0.0f, 0.75f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion8()), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
color = glm::vec3(1.0f, 0.71f, 0.75f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
//sp.render();

// Muslo trasero derecho
model = glm::translate(model, glm::vec3(0.0f, -1.5f, 0.0f));
modelaux2 = model;
model = glm::scale(model, glm::vec3(0.5f, 3.0f, 1.0f));
color = glm::vec3(1.0f, 0.71f, 0.75f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[2]->RenderMeshGeometry();

// Articulacion rodilla derecha trasera
model = modelaux2;
model = glm::translate(model, glm::vec3(0.0f, -1.5f, 0.0f));
model = glm::rotate(model, glm::radians(-60.0f), glm::vec3(0.0f, 0.0f, 0.75f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion9()), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
color = glm::vec3(1.0f, 0.71f, 0.75f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
//sp.render();

// Pata trasera derecha
model = glm::translate(model, glm::vec3(0.0f, -1.5f, 0.0f));
modelaux2 = model;
model = glm::scale(model, glm::vec3(0.5f, 3.0f, 1.0f));
color = glm::vec3(1.0f, 0.71f, 0.75f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[2]->RenderMeshGeometry();

// Pie trasero derecho
model = modelaux2;
model = glm::translate(model, glm::vec3(-0.75f, -1.5f, 0.0f));
model = glm::rotate(model, glm::radians(30.0f), glm::vec3(0.0f, 0.0f, 0.75f));
model = glm::scale(model, glm::vec3(1.5f, 0.5f, 1.0f));
color = glm::vec3(0.2f, 0.2f, 0.2f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[2]->RenderMeshGeometry();
```

```
// Articulación pata - cuerpo trasera izquierda
model = modelaux;
model = glm::translate(model, glm::vec3(4.0f, -1.2f, 1.5f));
model = glm::rotate(model, glm::radians(30.0f), glm::vec3(0.0f, 0.0f, 0.75f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion10()), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
color = glm::vec3(1.0f, 0.71f, 0.75f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
//sp.render();

// Muslo trasero izquierdo
model = glm::translate(model, glm::vec3(0.0f, -1.5f, 0.0f));
modelaux2 = model;
model = glm::scale(model, glm::vec3(0.5f, 3.0f, 1.0f));
color = glm::vec3(1.0f, 0.71f, 0.75f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[2]->RenderMeshGeometry();

// Articulacion rodilla izquierda trasera
model = modelaux2;
model = glm::translate(model, glm::vec3(0.0f, -1.5f, 0.0f));
model = glm::rotate(model, glm::radians(-60.0f), glm::vec3(0.0f, 0.0f, 0.75f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion11()), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
color = glm::vec3(1.0f, 0.71f, 0.75f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
//sp.render();

// Pata trasera izquierda
model = glm::translate(model, glm::vec3(0.0f, -1.5f, 0.0f));
modelaux2 = model;
model = glm::scale(model, glm::vec3(0.5f, 3.0f, 1.0f));
color = glm::vec3(1.0f, 0.71f, 0.75f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[2]->RenderMeshGeometry();

// Pie trasero izquierdo
model = modelaux2;
model = glm::translate(model, glm::vec3(-0.75f, -1.5f, 0.0f));
model = glm::rotate(model, glm::radians(30.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::scale(model, glm::vec3(1.5f, 0.5f, 1.0f));
color = glm::vec3(0.1f, 0.1f, 0.1f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[2]->RenderMeshGeometry();
```

```
// Articulación Cola
model = modelaux;
model = glm::translate(model, glm::vec3(5.0f, 1.0f, 0.0f));
model = glm::rotate(model, glm::radians(-30.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion12()), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
color = glm::vec3(0.6f, 0.6f, 0.6f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
//sp.render();

// Cola
model = glm::translate(model, glm::vec3(0.0f, 2.0f, 0.0f));
modelaux2 = model;
model = glm::scale(model, glm::vec3(0.3f, 4.0f, 0.2f));
color = glm::vec3(1.0f, 0.71f, 0.75f);
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[3]->RenderMeshGeometry();
```

Archivos modificados para las teclas

Window.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; }
GLfloat getarticulacion11() { return articulacion11; }
GLfloat getarticulacion12() { return articulacion12; }
```

Window.cpp

```
Window::Window(GLint windowWidth, GLint windowHeight)
{
    width = windowWidth;
    height = windowHeight;
    rotax = 0.0f;
    rotay = 0.0f;
    rotaz = 0.0f;
    articulacion1 = 0.0f;
    articulacion2 = 0.0f;
    articulacion3 = 0.0f;
    articulacion4 = 0.0f;
    articulacion5 = 0.0f;
    articulacion6 = 0.0f;
    articulacion7 = 0.0f;
    articulacion8 = 0.0f;
    articulacion9 = 0.0f;
    articulacion10 = 0.0f;
    articulacion11 = 0.0f;
    articulacion12 = 0.0f;

    for (size_t i = 0; i < 1024; i++)
    {
        keys[i] = 0;
    }
}
```

```
if (key == GLFW_KEY_E)
{
    theWindow->rotax += 10.0;
}
if (key == GLFW_KEY_R)
{
    theWindow->rotay += 10.0; //rotar s
}
if (key == GLFW_KEY_T)
{
    theWindow->rotaz += 10.0;
}
if (key == GLFW_KEY_F)
{
    theWindow->articulacion1 += 10.0;
}

if (key == GLFW_KEY_G)
{
    theWindow->articulacion2 += 10.0;
}
if (key == GLFW_KEY_H)
{
    theWindow->articulacion3 += 10.0;
}
if (key == GLFW_KEY_J)
{
    theWindow->articulacion4 += 10.0;
}
if (key == GLFW_KEY_K)
{
    theWindow->articulacion5 += 10.0;
}
if (key == GLFW_KEY_L)
{
    theWindow->articulacion6 += 10.0;
}
if (key == GLFW_KEY_SEMICOLON)
{
    theWindow->articulacion7 += 10.0;
}
if (key == GLFW_KEY_APOSTROPHE)
{
    theWindow->articulacion8 += 10.0;
}
if (key == GLFW_KEY_U)
{
    theWindow->articulacion9 += 10.0;
}
if (key == GLFW_KEY_I)
{
    theWindow->articulacion10 += 10.0;
}
```

```
if (key == GLFW_KEY_O)
{
    theWindow->articulacion11 += 10.0;
}
if (key == GLFW_KEY_P)
{
    theWindow->articulacion12 += 10.0;
}
if (key == GLFW_KEY_D && action == GLFW_PRESS)
{
    const char* key_name = glfwGetKeyName(GLFW_KEY_D, 0);
    //printf("se presiono la tecla: %s\n",key_name);
}

if (key >= 0 && key < 1024)
{
    if (action == GLFW_PRESS)
    {
        theWindow->keys[key] = true;
        //printf("se presiono la tecla %d'\n", key);
    }
    else if (action == GLFW_RELEASE)
    {
        theWindow->keys[key] = false;
        //printf("se solto la tecla %d'\n", key);
    }
}
```

Ejecución

Actividad1

TECLAS (American Keyboard):

Articulaciones del brazo (F, G, H, J)

Estas articulaciones se encuentran realizadas en el ejercicio de clase, por lo que no se incluyen en este documento para que no se haga tan largo, pero pueden consultarse ahí

Articulaciones llantas (K L ; ‘)

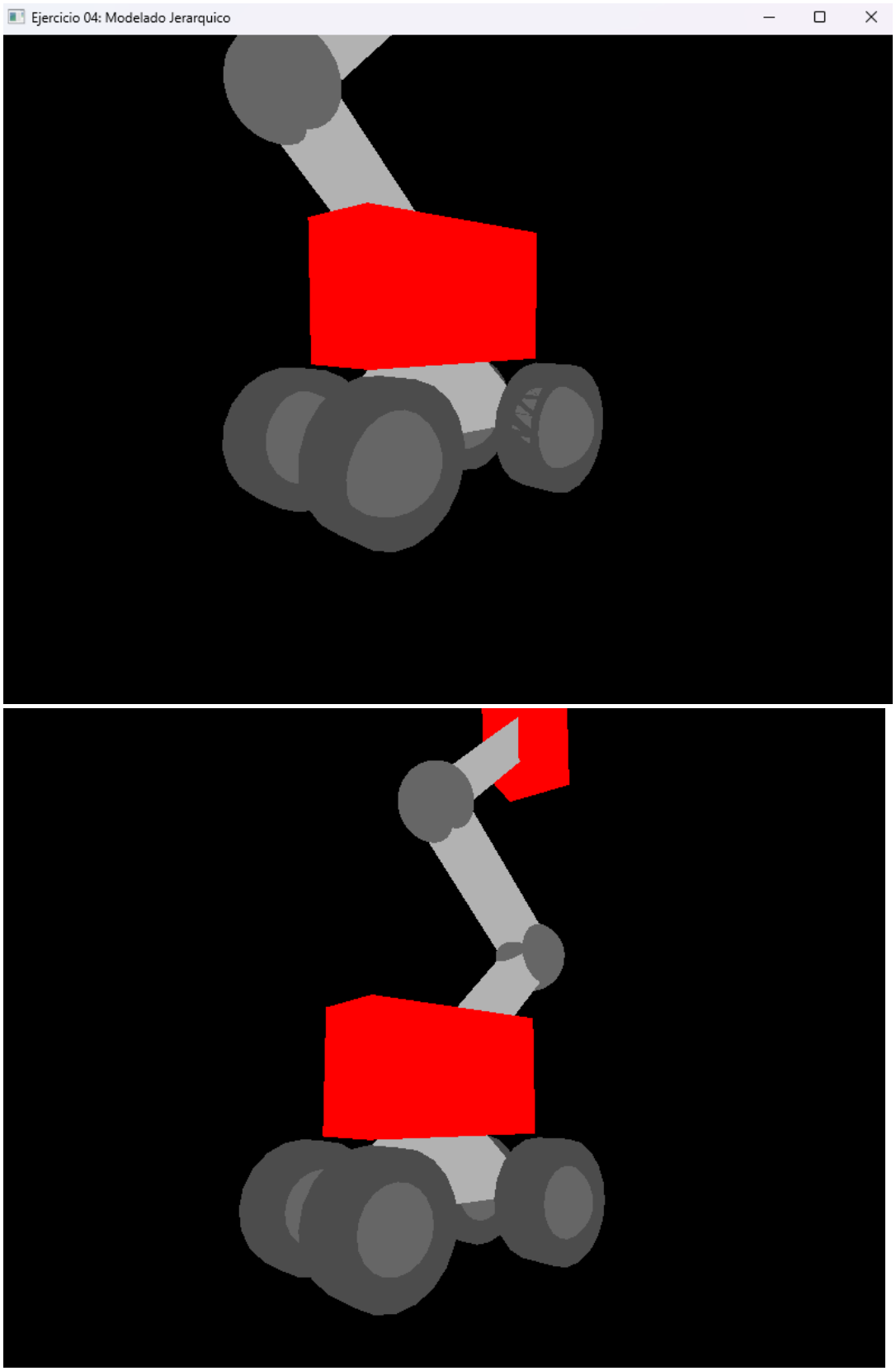
En imagen no se nota tanto el giro, pero para eso está el video, ahí se logra apreciar mejor, se realiza con las letras

K – llanta delantera izquierda

L – llanta delantera derecha

; - llanta trasera izquierda

‘ - llanta trasera derecha

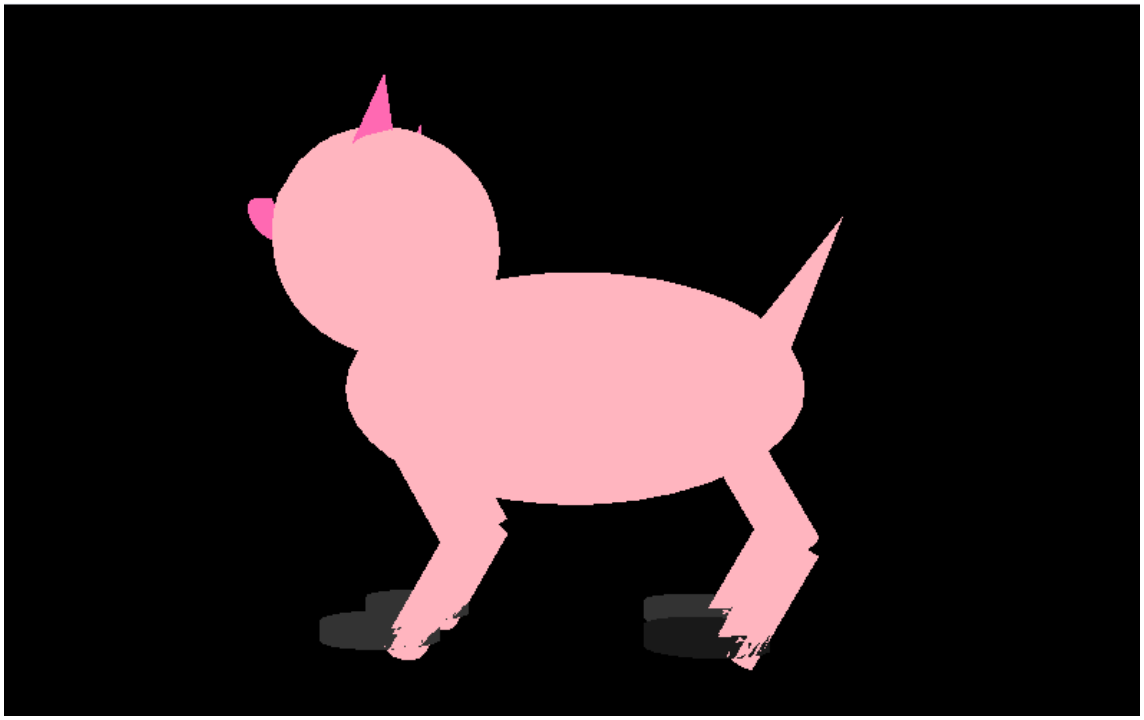


Video:

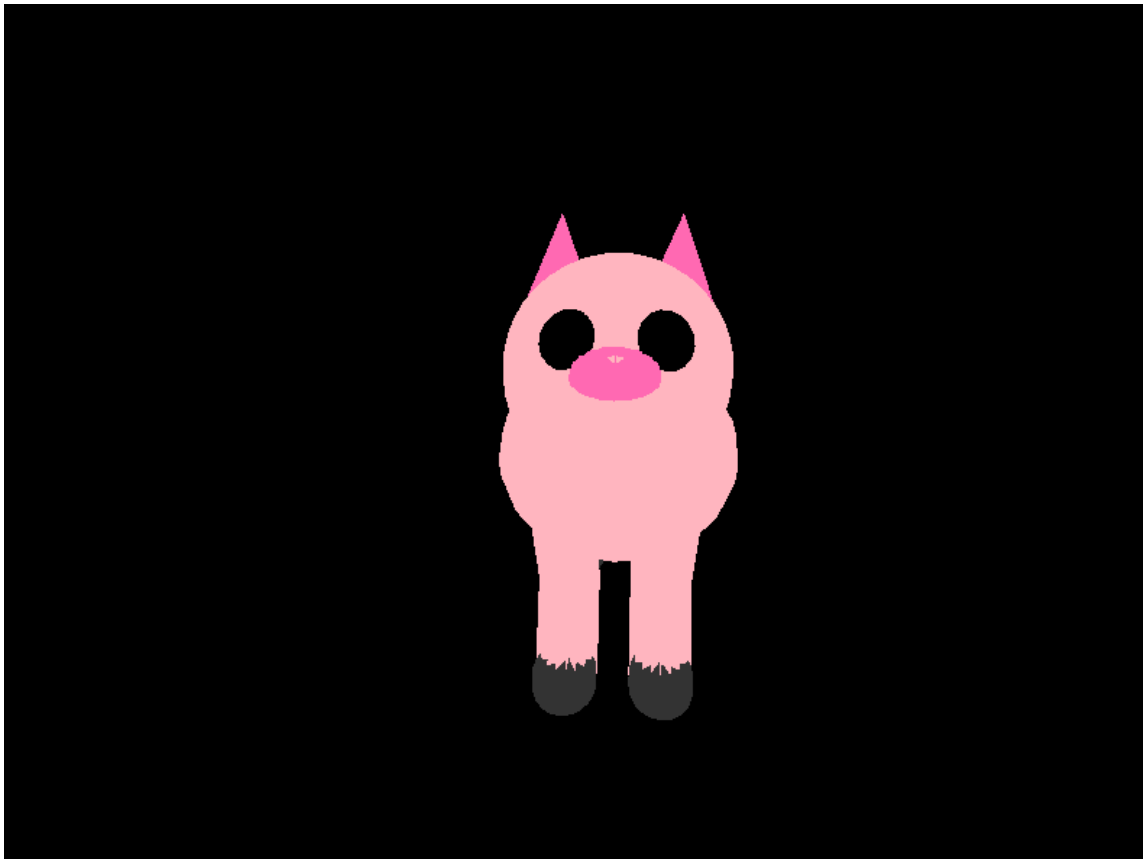
https://youtu.be/9KFDrk6yCTw

Actividad 2

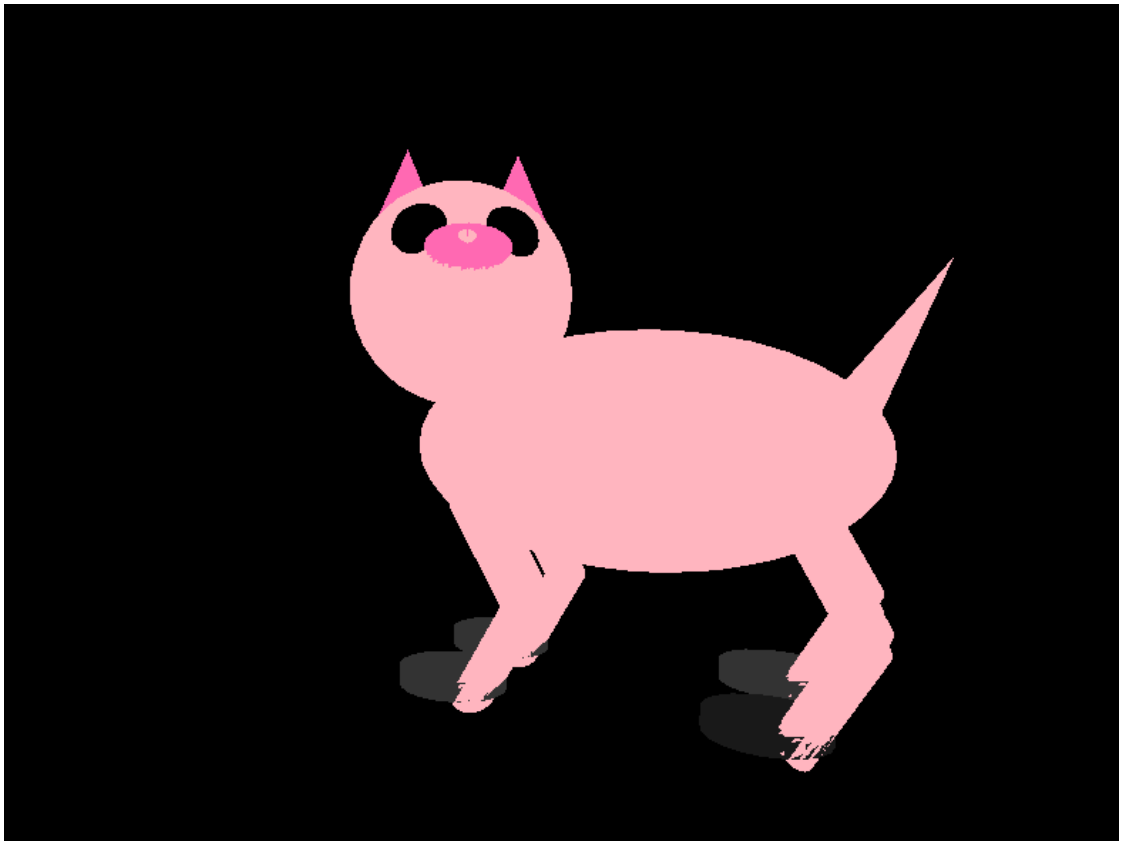
Vista lateral



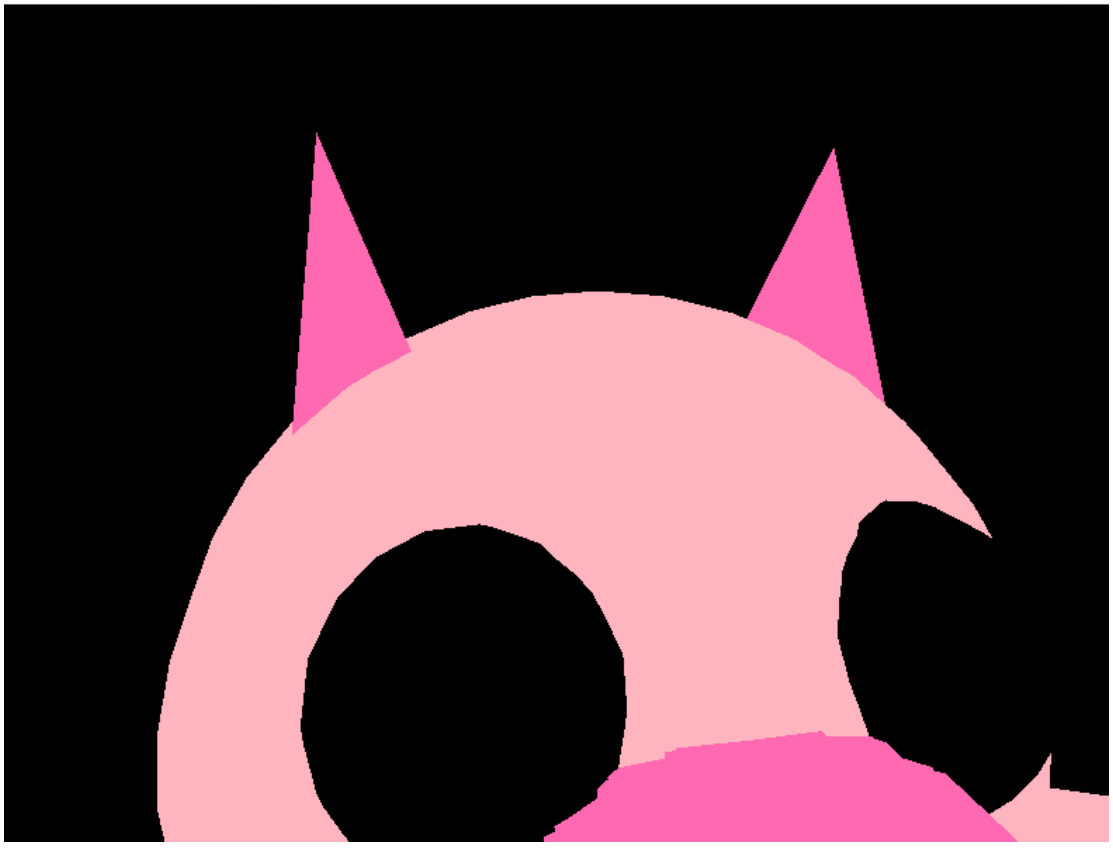
Vista frontal



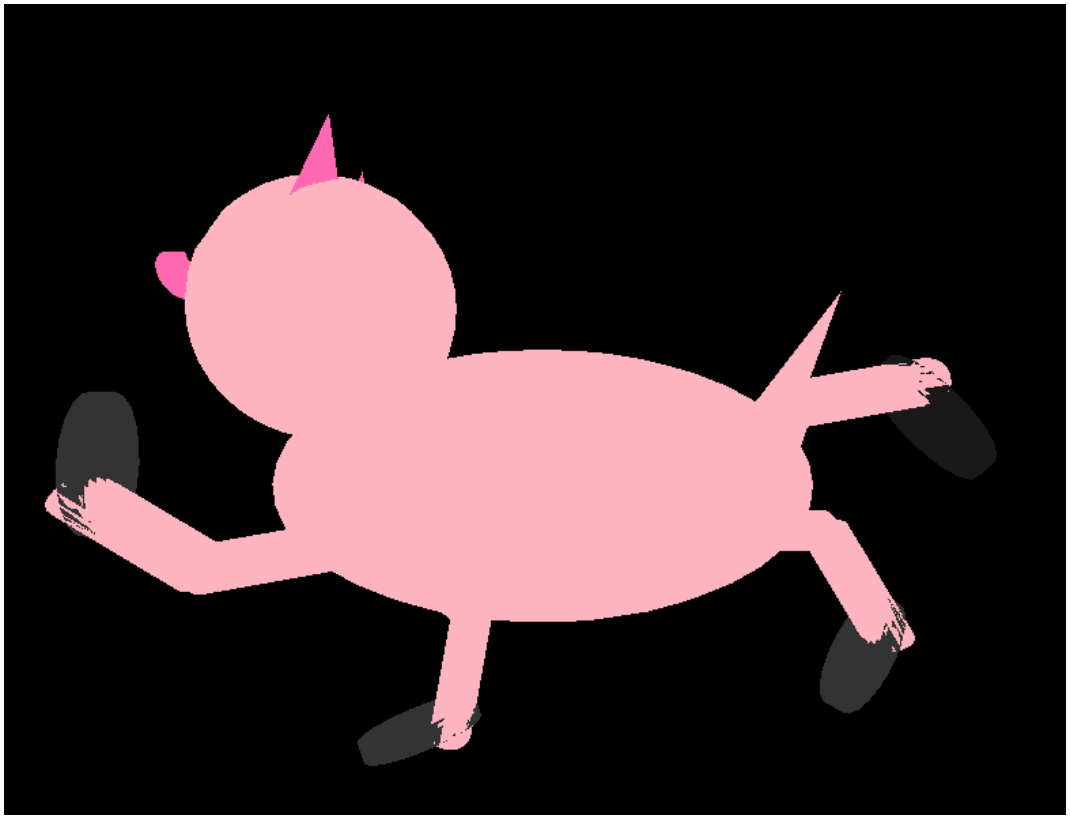
Articulación de la cabeza (F)



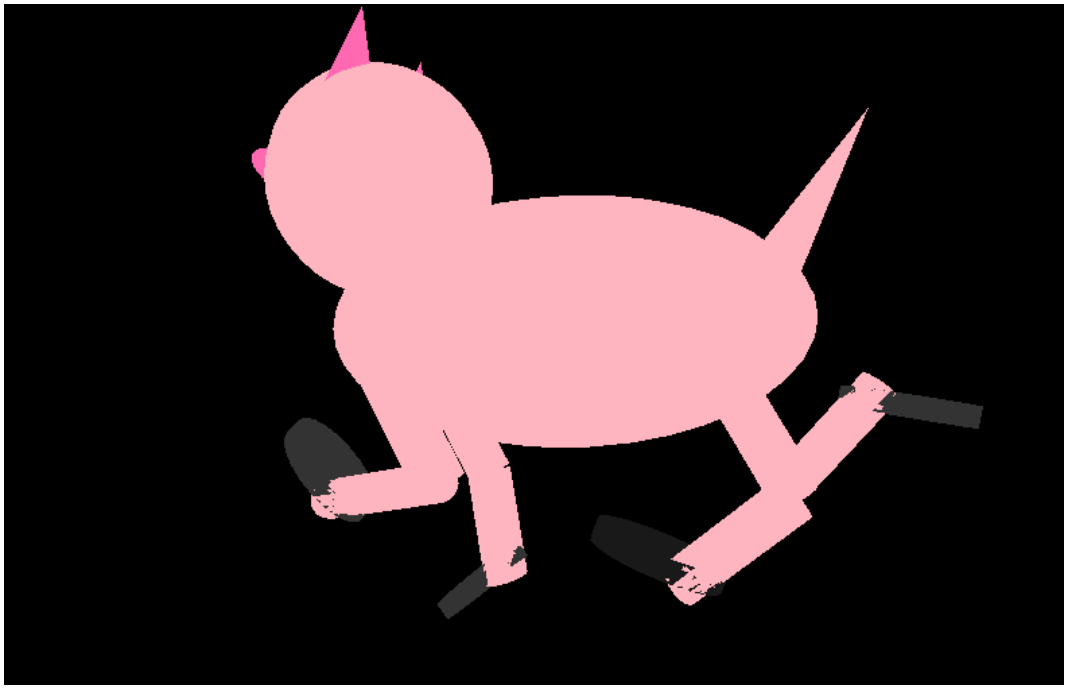
Articulación de las orejas (G, H)



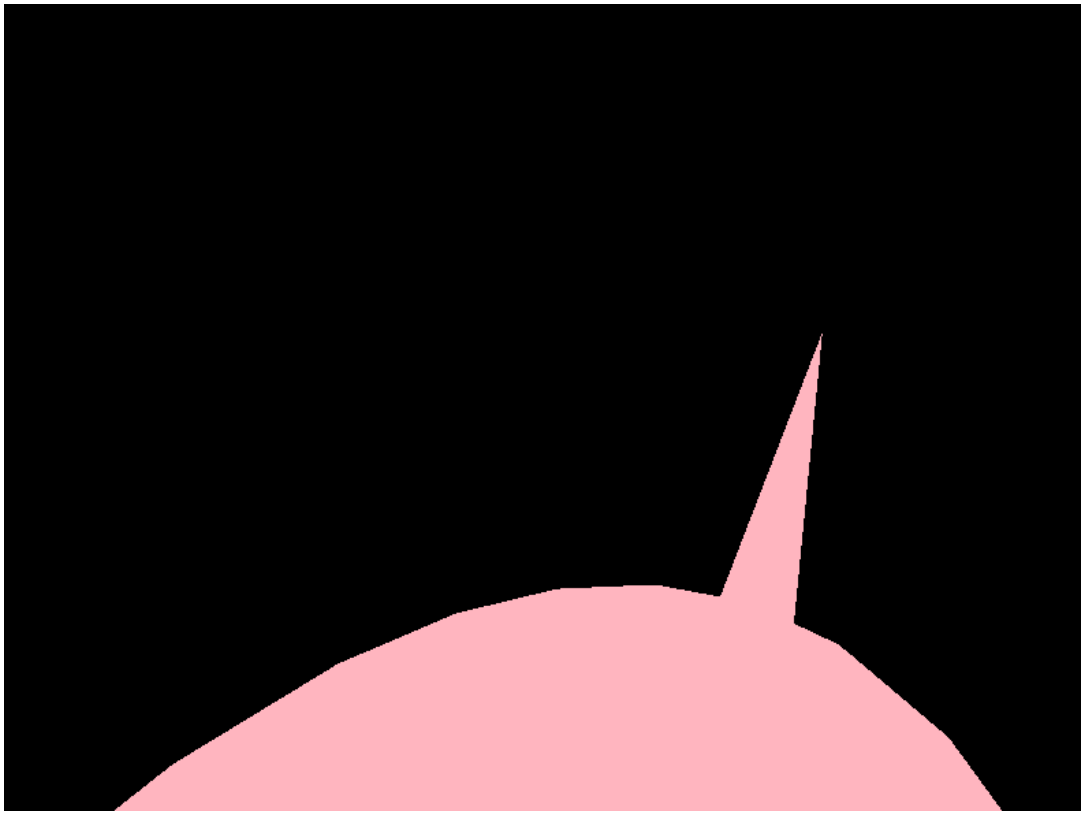
Articulación de muslos (J, L, ‘, I)



Articulación de patas o rodillas (K, ;, U, O)



Articulación de la cola (P)



Video:
https://youtu.be/l4dHj_nRIUM

Problemas presentados

En esta práctica solo se presentaron 2 problemas, el primero fue a cerca de las matrices de modelo, ya que al querer guardar el "estado" de un punto del modelo en concreto para ser reutilizado en la jerarquía, asigné mal la variable y se escaló todo aun más de lo que se debía, ya que guardé el estado después del escalamiento, no antes, por lo que se aplicó un doble escalamiento, esto se resolvio simplemente guardando el estado de la matriz auxiliar antes del escalamiento.

El 2do error fue sobre la asignación de teclas, ya que no consideré que las teclas se guardan con respecto al teclado americano, por lo que a la variable que en mi teclado en español era la "ñ" y "{" en realidad eran ";" y " ' ". Esto lo resolví viendo con que tecla funcionaba realmente cuando las asigné y ahí me di cuenta de este error, además de que revisé la documentación de GLFW Keys, así que solo reasigné las teclas con respecto al teclado americano.

Conclusión:

Esta práctica me pareció muy buena, en realidad creo que fue bastante divertida, ya que el hecho de hacer que se "movieran" cosas a través de las articulaciones y el

teclado fue algo bastante interesante, además que el hecho de usar más matrices auxiliares para la jerarquía, fue algo que ayudó mucho a que este tema se comprendiera y darnos cuenta de su importancia, en especial la flexibilidad que nos da para trasladar o rotar objetos con respecto a otros.

La complejidad de esta práctica me pareció regular, quizá un poco fácil ya entendiendo bien la jerarquía, ya que al inició me costó un poco entenderla, tuve que revisar el código de la grúa un par de veces, pero con el trabajo hecho en el ejercicio 4, quedó bastante más claro, y para la actividad 1 y 2 de esta práctica, ya todo fue más sencillo, prácticamente sin casi errores.

Solo tengo una “recomendación” la cuál es más que nada referida a las teclas para mover los modelos, en el salón usamos F G H J, pero en la práctica se planteaba usar más objetos que pudieran moverse, implicando más teclas, entonces algo que propondría es que se diera como un “estándar” de teclas para asignar, ya que creo que para modelos que ocupen muchas articulaciones, cómo el animal que tuvimos que crear, quizá pueda resultar algo confuso para alguien que no diseño el programa, el cómo mover los objetos.

Bibliografía

- GLFW. (Febrero, 2024). “GLFW: Keyboard key tokens”. Recuperado el 20 de septiembre de 2025 de: https://www.glfw.org/docs/3.3/group__keys.html