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: Gabriel Patricio Balam Flores

Nº de Cuenta: 320280324

GRUPO DE LABORATORIO: 03

GRUPO DE TEORÍA: 04

SEMESTRE 2026-1

FECHA DE ENTREGA LÍMITE: 20/sept

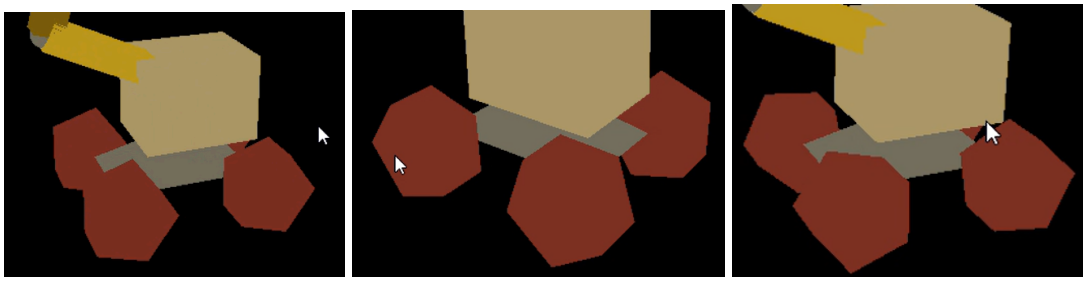
CALIFICACIÓN: _____

REPORTE DE PRÁCTICA

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

Ejercicio #01

El primer ejercicio consistía en terminar la grúa que se empezó en el ejercicio de clase, agregando el cuerpo, la base y las 4 llantas. Con el objetivo de que se vea la rotación de las llantas de una mejor manera, bajé la resolución de los cilindros como se muestra en las siguientes imágenes.

Ejecución del ejercicio


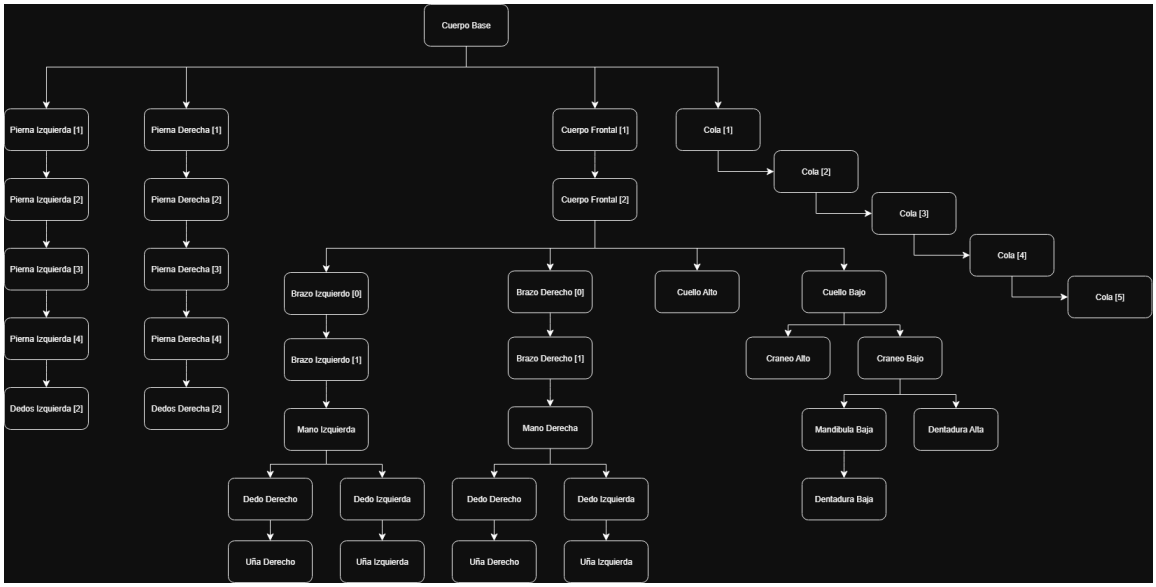
A continuación se muestran los controles para utilizar las articulaciones:

Ruedas	Brazo	Cabina
Q, E, R, T	Z, X, C	V

Ejercicio #02

El segundo ejercicio de la práctica consiste en la creación de un animal robot utilizando las formas básicas que hemos trabajado hasta el momento. En mi caso, decidí modelar un T-rex.

En la siguiente imagen se muestra el árbol de jerarquía del modelo.

Árbol de jerarquía


A continuación se muestran los controles para utilizar las articulaciones:

V: simular caminata

Cabeza	Brazo izquierdo	Pierna derecha
P: mandíbula inferior O: cuello y mandíbula inferior	L: brazo completo , : codo	Y: brazo completo H: codo B: tobillo
Cola	**Brazo derecho**	**Pierna izquierda**
F: horizontal G: vertical	K: brazo completo M: codo	U: brazo completo J: codo N: tobillo

Mientras realizaba la práctica, quise reducir el rango de rotación de las articulaciones para simular de manera más realista el movimiento de las extremidades del dinosaurio. Para lograrlo, utilicé un *if* que determina el signo del incremento o decremento del ángulo.

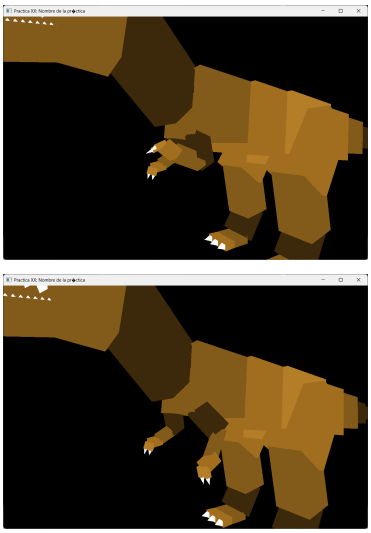
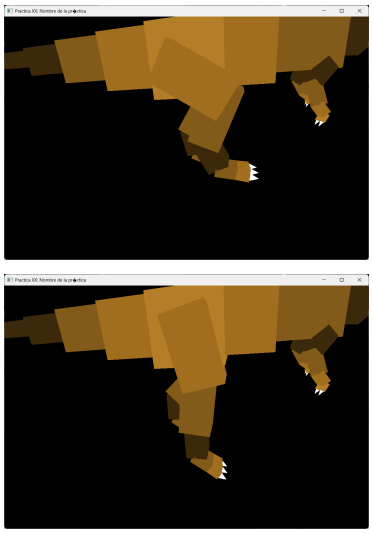
Bloques de código para restringir rango de la articulación

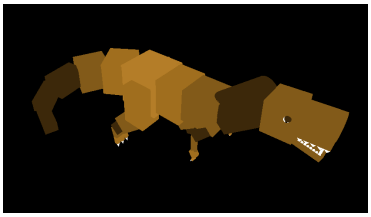
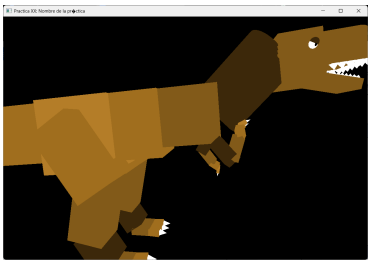
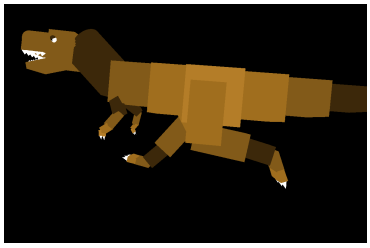
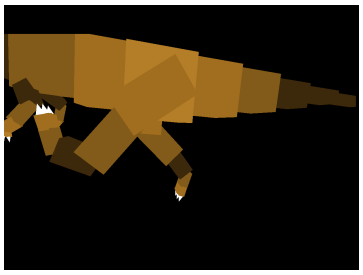
```
if (key == GLFW_KEY_F or key == GLFW_KEY_V)
{
    theWindow->articulacion1 += 2.5 * theWindow->dir1;

    if (theWindow->articulacion1 == 45)
        theWindow->dir1 *= -1.0f;
    else if (theWindow->articulacion1 == -45)
        theWindow->dir1 *= -1.0f;
}
```

También quise crear botones que controlaran dos o más articulaciones al mismo tiempo, para simular movimientos más interesantes. Con la tecla V se representa una especie de caminata, en la que se mueven simultáneamente las piernas, los brazos, la cola y la mandíbula.

A continuación, se muestran algunas imágenes que ilustran el movimiento de las articulaciones.

Cabeza	Brazo izquierdo	Pierna derecha
		

Cola	Brazo derecho	Pierna izquierda
 	 	 

Fragmento de código

```
//para reiniciar la matriz de modelo con valor de la matriz identidad
model = glm::mat4(1.0);
modelaux = model;
modelbase = model;
modelaux2 = model;

// Posicionamiento inicial ---------------------------------------------------------------------------
model = glm::translate(model, glm::vec3(0.0f, 6.0f, -4.0f));
model = glm::rotate(model, glm::radians(5.0f), glm::vec3(0.0f, 0.0f, 1.0f));

modelaux2 = model;

// Cuerpo Base ---------------------------------------------------------------------------
modelaux = model;
model = glm::scale(model, glm::vec3(4.0f, 4.0f, 4.0f));

glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
color = glm::vec3(0.7098f, 0.502f, 0.1647f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh(); //dibuja cubo, pirámide triangular, pirámide base cuadrangular


// Cuerpo Enfrente [1] ---------------------------------------------------------------------------
model = modelaux;
model = glm::translate(model, glm::vec3(3.0f, 0.25f, 0.0f));

modelaux = model;
model = glm::scale(model, glm::vec3(3.0f, 3.5f, 3.5f));

glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.6392f, 0.4471f, 0.1216f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh(); //dibuja cubo, pirámide triangular, pirámide base cuadrangular


// Cuerpo Enfrente [2] ---------------------------------------------------------------------------
model = modelaux;
model = glm::translate(model, glm::vec3(3.0f, 0.25f, 0.0f));
modelarm = model;

modelaux = model;
model = glm::scale(model, glm::vec3(3.0f, 3.0f, 3.0f));

glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.5098f, 0.3569f, 0.1137f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh(); //dibuja cubo, pirámide triangular, pirámide base cuadrangular

// Articulación del cuello bajo ---------------------------------------------------------------------------
model = modelaux;
model = glm::translate(model, glm::vec3(1.75f, 0.5f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion7()), glm::vec3(0.0f, 0.0f, 1.0f));

// Cuello bajo [1] ---------------------------------------------------------------------------
model = glm::translate(model, glm::vec3(1.25f, 0.5f, 0.0f));


modelaux = model;
model = glm::rotate(model, glm::radians(35.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::scale(model, glm::vec3(4.0f, 2.25f, 2.25f));


glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.251f, 0.1647f, 0.0471f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh(); //dibuja cubo, pirámide triangular, pirámide base cuadrangular
```

```
// Cuello alto [2] ---------------------------------------------------------------------------
model = modelaux;
model = glm::translate(model, glm::vec3(0.0f, 0.75f, 0.0f));

modelaux = model;
model = glm::rotate(model, glm::radians(90.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::rotate(model, glm::radians(35.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::scale(model, glm::vec3(1.2f, 4.0f, 1.2f));


glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.251f, 0.1647f, 0.0471f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[2]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro y cono

// Craneo ---------------------------------------------------------------------------
model = modelaux;
model = glm::translate(model, glm::vec3(2.75f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(-5.0f), glm::vec3(0.0f, 0.0f, 1.0f));

modelbase = model;
modelaux = model;
model = glm::scale(model, glm::vec3(2.0f, 2.5f, 2.5f));


glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.5098f, 0.3569f, 0.1137f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[0]->RenderMesh(); //dibuja cubo, pirámide triangular, pirámide base cuadrangu

// Ojo izquierdo ---------------------------------------------------------------------------
model = modelbase;
model = glm::translate(model, glm::vec3(0.5f, 0.5f, 1.25f));
model = glm::scale(model, glm::vec3(0.2f, 0.2f, 0.2f));
modelaux = model;

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));
sp.render(); //dibuja las figuras geométricas cilindro y cono

// Iris
model = modelaux;
model = glm::translate(model, glm::vec3(0.2f, -0.2f, 0.75f));
model = glm::scale(model, glm::vec3(0.3f, 0.5f, 0.3f));

glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
sp.render(); //dibuja las figuras geométricas cilindro y cono

// Parpado
model = modelaux;
model = glm::translate(model, glm::vec3(0.3f, 0.45f, 0.2f));
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));

glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(0.251f, 0.1647f, 0.0471f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
sp.render(); //dibuja las figuras geométricas cilindro y cono


// Ojo Derecho ---------------------------------------------------------------------------
model = modelbase;
model = glm::translate(model, glm::vec3(0.5f, 0.5f, -1.25f));
model = glm::scale(model, glm::vec3(0.2f, 0.2f, 0.2f));
modelaux = model;

glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
color = glm::vec3(1.0f, 1.0f, 1.0f);
```

2.- Liste los problemas que tuvo a la hora de hacer estos ejercicios y si los resolvió explicar cómo fue, en caso de error adjuntar captura de pantalla.

Contratiempos

Esta práctica puede parecer complicada a primera vista. Sin embargo, una vez que uno se adapta a los movimientos de las formas mediante las rotaciones, se convierte en un ejercicio de acomodar las figuras hasta posicionarlas correctamente. Por lo tanto, no encontré mayores contratiempos ni problemas relacionados con el tema de la práctica.

3.- Conclusión:

a. Los ejercicios del reporte: Complejidad, Explicación.
b. Comentarios generales.
c. Conclusión

1. Bibliografía en formato APA

Conclusión

El modelado jerárquico permite manipular las partes de un modelo sin tener que preocuparse por los cálculos o movimientos de las piezas dependientes, lo que facilita lograr un desplazamiento o una rotación adecuados. Sin este tipo de modelado, animar cada pieza de manera independiente sería un proceso complejo y tedioso.

La construcción del árbol de jerarquía proporciona una base sólida para iniciar el modelado del dinosaurio. Uno de los aspectos más relevantes de este esquema es que ofrece una idea clara del número de variables auxiliares necesarias para la creación del modelo y, en consecuencia, simplifica la planificación del trabajo.

Bibliografía

- Página principal. (26 de septiembre de 2018). *Wiki de OpenGL* . Recuperado el 29 de agosto de 2025 de http://www.khronos.org/opengl/wiki_opengl/index.php?title=Main_Page&oldid=14430 .