UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE INGENIERÍA ELÉCTRICA

INGENIERÍA EN COMPUTACIÓN

LABORATORIO DE COMPUTACIÓN GRÁFICA e INTERACCIÓN HUMANO COMPUTADORA

REPORTE DE PRÁCTICA Nº 04

NOMBRE COMPLETO: Vázquez Reyes Sebastián

Nº de Cuenta: 318150923

GRUPO DE LABORATORIO: 11

GRUPO DE TEORÍA: 6

SEMESTRE 2024-2

FECHA DE ENTREGA LÍMITE: miércoles 6 de marzo de 2024

CALIFICACIÓN: _____

REPORTE DE PRÁCTICA:

1.1 Terminar la Grúa con: cuerpo(prisma rectangular), base (pirámide cuadrangular) y 4 llantas(4 cilindros) que con teclado se pueden girar las 4 llantas por separado

En el código del ejercicio ya había implementado las llantas, la base, y el cuerpo de la grúa. Solo corregí el Angulo de rotación de su brazo:

model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()), glm::vec3(0.0f, 0.0f, 1.0f));

Y agregue rotación a las llantas con la siguiente línea de código:

model = glm::rotate(model, glm::radians(mainWindow.getrotaz()), glm::vec3(0.0f, 1.0f, 0.0f));

Mediante modelado jerárquico, esta última instrucción se aplica a todas las llantas

1.2 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) y cola articulada o 2 orejas articuladas. (con teclado se puede mover la cola o cada oreja independiente.

Para el cuerpo de la arañita, utilicé el siguiente código:

```
////AQUI EMPIEZA EL DIBUJO DE LA ARAÑITA
//Cuerpo
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(30.0f, 5.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f));
modelaux2 = model;
modelaux = model;
color = glm::vec3(0.8f, 0.8f, 0.8f);
model = glm::scale(model, glm::vec3(5.0f, 5.0f, 4.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
sp.render();
```

Luego, es el código para la primera patita del animal

```
// PATA 1
//articulacion 1.1
model = modelaux;
model = glm::translate(model, glm::vec3(-3.0f, -3.0f, +2.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion5()), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::rotate(model, glm::radians(15.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::rotate(model, glm::radians(60.0f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(75.0f), glm::vec3(0.0f, 1.0f, 0.0f));
modelaux = model;
color = glm::vec3(0.3f, 0.3f, 0.3f);
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
sp.render();
//femur1
model = modelaux;
model = glm::translate(model, glm::vec3(-0.25f, 5.0f, 0.0f));
model = glm::scale(model, glm::vec3(1.0f, 10.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[0]->RenderMesh();

//articulacion1.2
model = modelaux;
model = glm::translate(model, glm::vec3(0.0f, 9.5f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion6()), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::rotate(model, glm::radians(100.0f), glm::vec3(0.0f, 0.0f, 1.0f));
modelaux = model;
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
sp.render();

//pierna
model = modelaux;
model = glm::translate(model, glm::vec3(0.0f, 4.5f, 0.0f));
model = glm::scale(model, glm::vec3(1.0f, 8.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
```

```
//pierna
model = modelaux;
model = glm::translate(model, glm::vec3(0.0f, 4.5f, 0.0f));
model = glm::scale(model, glm::vec3(1.0f, 8.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[0]->RenderMesh();

//Punta
model = modelaux;
model = glm::translate(model, glm::vec3(0.0f, 8.0f, 0.0f));
model = glm::rotate(model, glm::radians(200.0f), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::scale(model, glm::vec3(1.5f, 1.0f, 1.5f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
meshList[3]->RenderMeshGeometry();
```

El funcionamiento de este código es básicamente el mismo que en la grúa. La única diferencia son los ángulos en que se rotan las figuras, y sus posiciones para que se mantengan alejados de la figura de la grúa. Este código se repite para las 4 patitas de la araña, permitiendo rotar a todo el cuerpo de la araña al mismo tiempo en el eje x, o rotar las patas enteras o las últimas articulaciones de la araña al mismo tiempo

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

Para el ejercicio del animalito, se me sigue complicando aplicar los ángulos correctos para que las figuras roten a la posición en que deben estar para dar forma al animalito (más precisamente sus patitas).

3.- Conclusión:

a. Los ejercicios del reporte: Complejidad, Explicación.

En realidad los ejercicios no fueron un gran problema, el único detalle que me costo entender en los 2 fue el tema de los ángulos de rotación, básicamente calcular a que dirección deben apuntar para que se forme la figura correcta. Aunque por falta de tiempo no pude terminar la cabecita de la araña.

b. Comentarios generales: Faltó explicar a detalle, ir más lento en alguna explicación, otros comentarios y sugerencias para mejorar desarrollo de la práctica

El modelado jerárquico me ayudo a entender mejor como rotar diversos objetos sobre un mismo punto o eje sin necesidad de hacerlos rotar uno por uno sobre ese punto. Es un método muy útil para mover figuras completas. Aunque la explicación del código para esta practica estuvo algo pesada.

c. Conclusión

Al finalizar esta práctica aprendí como rotar y mover objetos completos sin necesidad de escribir líneas de código que los muevan a todos uno por uno, si no usando una sola instrucción para cada figura mediante el modelado jerárquico. También aprendí a como rotar cada pequeña parte de la figura entera para mostrar su movimiento independiente sobre el del resto de la figura. Por último, aprendí a calcular los ángulos de rotación necesarios para rotar una parte de la figura y darle la forma de una representación real.

Bibliografía

- Navarro, J., (2018). “SUPERFICIES CILÍNDRICAS Y CÓNICAS”. Recuperado el 21 de febrero de 2024 de https://proyectodescartes.org/uudd/materiales_didacticos/superficies_curiosas-1_JS/cono-cilindro.html