UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE INGENIERÍA ELÉCTRICA

INGENIERÍA EN COMPUTACIÓN

LABORATORIO DE COMPUTACIÓN GRÁFICA e INTERACCIÓN HUMANO COMPUTADORA

EJERCICIOS DE CLASE Nº 05

NOMBRE COMPLETO: Vázquez Reyes Sebastián

Nº de Cuenta: 318150923

GRUPO DE LABORATORIO: 11

GRUPO DE TEORÍA: 6

SEMESTRE 2024-2

FECHA DE ENTREGA LÍMITE: Lunes 11 de marzo de 2024

CALIFICACIÓN: _____

EJERCICIOS DE SESIÓN:

- **Actividades realizadas. Una descripción de los ejercicios y capturas de pantalla de bloques de código generados y de ejecución del programa**

1. **Importar por separado y agregar jerarquía a: Cuerpo, cabeza, mandíbula inferior y cada una de las 4 patas como un solo modelo**

Para esta parte, termine de crear los modelos de las patas, que vienen anexados en el .zip, y los importe de la siguiente manera:

```
//------------*INICIA DIBUJO DE GODDARD-------------------*
//cuerpo y cola
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, 2.0f, -1.5f));
model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f));
modelaux = model;
color = glm::vec3(0.0f, 0.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
goddard_cuerpo.RenderModel();
//En sesión se separara una parte del modelo de Goddard y se unirá por jeraquía al cuerpo
//Cabeza
model = modelaux;
model = glm::translate(model, glm::vec3(-1.2f, 0.5f, 0.0f));
color = glm::vec3(0.0f, 1.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
goddard_craneo.RenderModel();

//mandibula inferior
model = modelaux;
model = glm::translate(model, glm::vec3(-2.25f, 0.8f, -0.2f));
color = glm::vec3(1.0f, 0.0f, 1.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
goddard_mandibula.RenderModel();

//pata delantera derecha
```

La variable modelaux almacena la traslación y rotación con la que inicia el cuerpo de Goddard. La cabeza y mandíbula comparten estas operaciones, y a partir de ahí se trasladan a su posición.

Para las patas, realice el siguiente código:

```
//pata delantera derecha
model = modelaux;
model = glm::translate(model, glm::vec3(-1.25f, -0.4f, 0.8f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()), glm::vec3(0.0f, 0.0f, 1.0f));
color = glm::vec3(1.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
pata1.RenderModel();

// pata delantera izquierda
model = modelaux;
model = glm::translate(model, glm::vec3(-1.25f, -0.4f, -1.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()), glm::vec3(0.0f, 0.0f, -1.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()), glm::vec3(0.0f, 0.0f, -1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
pata2.RenderModel();
//pata trasera derecha
model = modelaux;
model = glm::translate(model, glm::vec3(0.8f, -1.3f, 0.8f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()), glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()), glm::vec3(0.0f, 0.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
pata3.RenderModel();

//pata trasera izquierda
model = modelaux;
model = glm::translate(model, glm::vec3(0.8f, -1.3f, -1.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()), glm::vec3(0.0f, 0.0f, -1.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()), glm::vec3(0.0f, 0.0f, -1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
pata4.RenderModel();
```

Las patas comparten las mismas operaciones que el cuerpo y la cabeza. A cada una de las patas agregue las rotaciones necesarias para que se muevan cada vez que se presione la tecla F y G. El perrito mueve las patas derechas hacia adelante al presionar la tecla F, y las patas izquierdas hacia adelante al presionar la tecla G. Las patas izquierdas tienen un signo negativo en el eje sobre el que rotan para que su rotación sea inversa al de las patas derechas y parezca que el perrito camina.

2. **Agregar rotaciones a las patas de forma independiente para que se muevan como si fuera a avanzar Goddard, limitar la rotación a 45° en Cada sentido**

Para la rotación cree el siguiente código en el archivo window.cpp:

```
if (key == GLFW_KEY_F)
{
    if (angulos > -45.0) {
        theWindow->articulacion1 -= 5.0;
        angulos -= 5.0;
    }
}

if (key == GLFW_KEY_G)
{
    if (angulos < 45.0) {
        theWindow->articulacion2 += 5.0;
        angulos += 5.0;
    }
}
```

Se utiliza una variable alterna de tipo float llamada “angulos” para controlar el ángulo máximo al que pueden llegar en cada extremo las patitas. Cuando se presiona la tecla G, las patas derechas se mueven en sentido horario, y cuando se presiona la tecla F, se

mueven en sentido antihorario. Cada vez que se mueven las patitas 5 grados, se suma 5 a esta variable

- **Problemas Presentados**

  Cuando intentaba hacer rotar a las patitas, en vez de utilizar una variable alterna, dentro del if colocaba la propia expresión: “`theWindow->articulacionX <= 45°`”. Termine descubriendo que cuando hacia esto, la articulación correspondiente no se limitaba a alcanzar 45 grados en algún sentido, si no que más bien obligaba al programa a solo utilizar 45 grados en la dirección que le colocaba. Cuando se acababan estos 45 grados, la patita dejaba de rotar. Por eso termine optando por usar una variable que se actualizaba con el valor en que avanzaban o retrocedían las patitas. Fuera de esto, no presente ningún problema

- **Conclusión:**

  Durante este ejercicio aprendí a definir límites de rotación para una figura, además, repasé el modelado jerárquico para compartir coordenadas de traslación y operaciones de rotación.