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 5

NOMBRE COMPLETO: Sanchez Villalpando Johan

Nº de Cuenta: 422028657

GRUPO DE LABORATORIO: 2

GRUPO DE TEORÍA: 4

SEMESTRE 2025-2

FECHA DE ENTREGA LÍMITE: 15/03/25

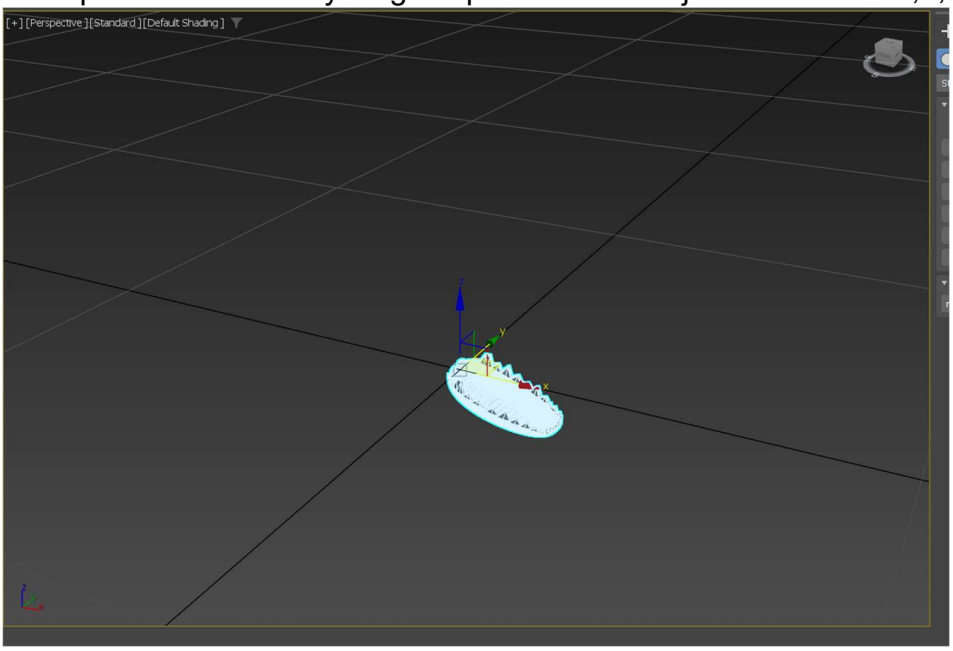
CALIFICACIÓN: _____

EJERCICIOS DE SESIÓN:

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

   Para esta práctica el ejercicio consiste en utilizar el modelo visto en la práctica, pero ahora separando mandíbula, cabeza, cuerpo y patas de Goddard.
   Para ello en

   Para ello fue necesario en 3ds max separar cada elemento, ajustar el pivote en la posición correcta y luego exportar con el objeto situado en 0,0,0.



   Luego se crean los modelos y se importan.

```
Camera camera;
Model Goddard_M;
Model Mandibula_M;
Model Cuerpo_M;
Model Cabeza_M;
Model patadelder_M;
Model patadelizq_M;
Model patatrasder_M;
Model patatrasizq_M;
```

```
Mandibula_M = Model(); //Creamos modelo
Mandibula_M.LoadModel("Models/mandibulag_obj.obj");

Cabeza_M = Model(); //Creamos modelo
Cabeza_M.LoadModel("Models/cabezag_obj.obj");

patadelder_M = Model(); //Creamos modelo
patadelder_M.LoadModel("Models/patag_delantera_der.obj");

patadelizq_M = Model(); //Creamos modelo
patadelizq_M.LoadModel("Models/patag_delantera_izq.obj");

patatrasder_M = Model(); //Creamos modelo
patatrasder_M.LoadModel("Models/patag_trasera_der.obj");

patatrasizq_M = Model(); //Creamos modelo
patatrasizq_M.LoadModel("Models/patag_trasera_izq.obj");
```

Luego, se crean los modelos siguiendo la jerarquía, donde el cuerpo es la parte principal, la cabeza está unida al cuerpo y a su vez la mandíbula está unida a la cabeza.
Las patas van unidas al cuerpo.

```
//Goddard
color = glm::vec3(0.0f, 0.0f, 0.0f); //modelo de goddard de color negro
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, 0.3f, -1.5f));
//se guarda el modelo del cuerpo con su traslación, para luego en la jerarquía seguir con cabeza y patas.
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));
Cuerpo_M.RenderModel();//modificar por el modelo sin las 4 patas y sin cola
//En sesión se separara una parte del modelo de Goddard y se unirá por jeraquía al cuerpo
model = modelaux;
//cabeza
color = glm::vec3(0.0f, 0.5f, 0.0f);
model = glm::translate(model, glm::vec3(1.35f, 0.8f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()), glm::vec3(0.0f, 0.0f, 1.0f));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Cabeza_M.RenderModel();//modificar por el modelo de mandibula
//mandibula unida a cabeza
color = glm::vec3(0.0f, 0.f, 0.5f);
model = glm::translate(model, glm::vec3(1.25f, 0.3f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()), glm::vec3(0.0f, 0.0f, 1.0f));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Mandibula_M.RenderModel();//modificar por el modelo de mandibula
model = modelaux; //se regresa al cuerpo para seguir con las patotas
```

En el código se crea el modelo del cuerpo, luego se guarda con la variable auxiliar y se crea cabeza y mandíbula, luego se regresa al cuerpo para poner los pies.

```
//pata delantera derecha
//h adelante j atras
color = glm::vec3(0.0f, 0.0f, 0.5f);
model = glm::translate(model, glm::vec3(1.1f, -0.15f, 0.6f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion3()), glm::vec3(0.0f, 0.0f, 1.0f));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
patadelder_M.RenderModel();//modificar por el modelo de pata
model = modelaux;

// pata delantera izquierda
//k adelante l atras
color = glm::vec3(0.0f, 0.0f, 0.5f);
model = glm::translate(model, glm::vec3(1.15f, -0.32f, -0.6f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion4()), glm::vec3(0.0f, 0.0f, 1.0f));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
patadelizq_M.RenderModel();//modificar por el modelo de pata
model = modelaux;
```

```
//pata trasera derecha
//z adelante x atras
color = glm::vec3(0.0f, 0.0f, 0.5f);
model = glm::translate(model, glm::vec3(-0.6f, -0.85f, 0.6f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion5()), glm::vec3(0.0f, 0.0f, 1.0f));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
patatrasder_M.RenderModel();//modificar por el modelo de pata
model = modelaux;

//pata trasera izquierda
//c adelante v atras
color = glm::vec3(0.0f, 0.0f, 0.5f);
model = glm::translate(model, glm::vec3(-0.6f, -0.925f, -0.6f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion6()), glm::vec3(0.0f, 0.0f, 1.0f));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
patatrasizq_M.RenderModel();//modificar por el modelo de pata
model = modelaux;
```

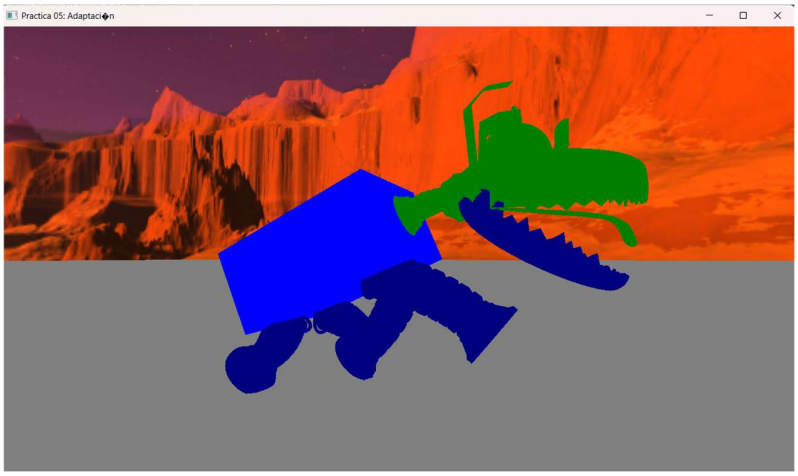
Para las teclas y el límite a 45° se realizaron modificaciones en el windows.cpp.
(La modificación para añadir tecla C y V, se tomó de la realizada en la práctica 4)

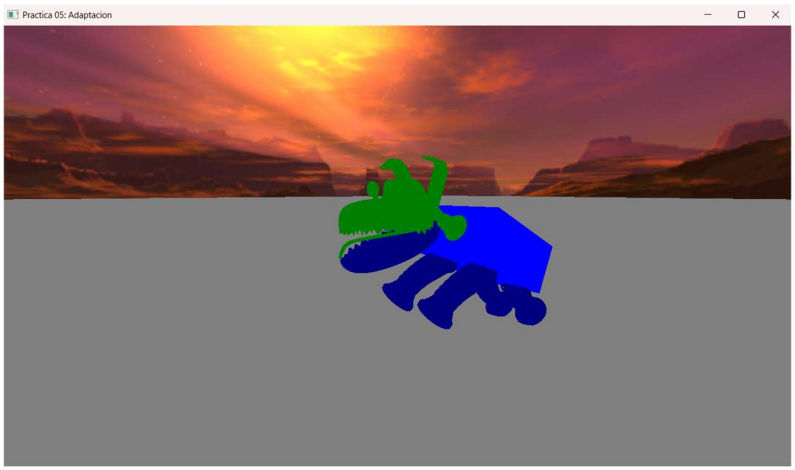
```
if (key == GLFW_KEY_H)
{
    if (theWindow->articulacion3 <= 45) {
        theWindow->articulacion3 += 10.0;
    }

}
if (key == GLFW_KEY_J)
{
    if (theWindow->articulacion3 >= -45) {
        theWindow->articulacion3 -= 10.0;
    }
}
```

La modificación consiste en agregar un condicional en las articulaciones, para poder realizar aumentos o disminuciones de valor requiere que no pase de 45° en la tecla que va hacia adelante y -45° en la tecla que va hacia atrás, de este modo cada articulación ahora tiene 2 modos, hacia delante y hacia atrás.
También se agregó articulación a la cabeza y mandíbula, pero sin límite.

Tenemos el resultado final con movimiento máximo a +-45°





2. **Problemas presentados. Listar si surgieron problemas a la hora de ejecutar el código**

   Este ejercicio no tuvo complicaciones, solamente poner el límite a 45° ya que estaba intentando hacerlo desde el main y me encontraba con muchos problemas, una vez haciéndolo en window.cpp funcionó bien.

3. **Conclusión:**
   a. **Los ejercicios de la clase: Complejidad, explicación**

b. **Comentarios generales: Faltó explicar a detalle, ir más lento en alguna explicación, otros comentarios y sugerencias.**

.

El ejercicio fue entretenido ya que no se tuvo que modelar por código triángulo por triángulo y nos centramos más en darle funciones, lo único complicado fue ponerle límite a la rotación, sin embargo, analizando el código se llegó una solución.

Con respecto a la parte de 3ds max, fue interesante trabajar con modelos 3D y la explicación fue muy buena, además la guía es muy útil si te pierdes en alguna parte o si lo olvidas a la hora de hacer el ejercicio o la práctica.