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: Gomez Enríquez Agustin

Nº de Cuenta: 317031405

GRUPO DE LABORATORIO: 3

GRUPO DE TEORÍA: 5

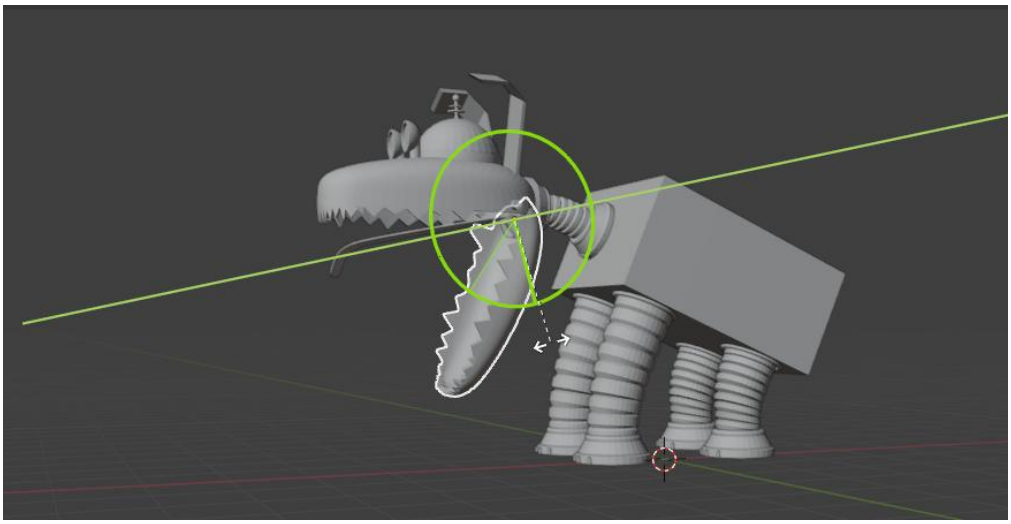
SEMESTRE 2026-1

FECHA DE ENTREGA LÍMITE: 24 – septiembre - 2025

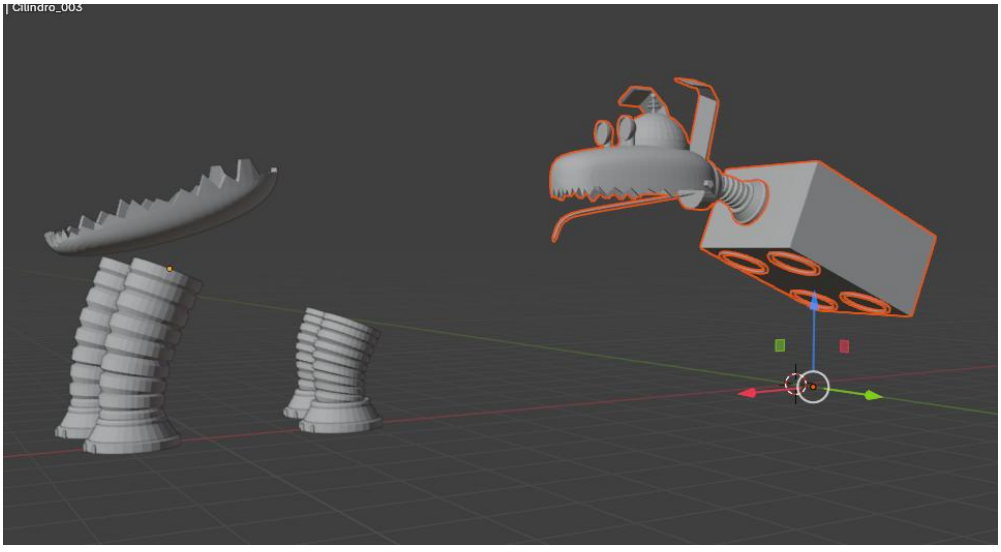
CALIFICACIÓN: _____

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**



Comenzamos por separar la mandíbula de Goddard desde blender y la exportamos como un modelo aparte



Hacemos lo mismo con cada extremidad y dejamos lo cabeza, cuerpo y cola como un solo modelo.

```
36      Camera camera;
37      Model Goddard_M; //Base
38      Model Goddard_pata1;
39      Model Goddard_pata2;
40      Model Goddard_pata3;
41      Model Goddard_pata4;
42
```

Creamos nuestras instancias para cada uno de los modelos

```
126     Goddard_M = Model();
127     Goddard_M.LoadModel("Models/goddard_base.obj");
128     Goddard_pata1 = Model();
129     Goddard_pata1.LoadModel("Models/goddard_pata1.obj");
130     Goddard_pata2 = Model();
131     Goddard_pata2.LoadModel("Models/goddard_pata2.obj");
132     Goddard_pata3 = Model();
133     Goddard_pata3.LoadModel("Models/goddard_pata3.obj");
134     Goddard_pata4 = Model();
135     Goddard_pata4.LoadModel("Models/goddard_pata4.obj");
136
```

Ahora, dentro del main vamos a cargar los modelos con la variable que ya creamos y los dejamos con el nombre que los guardamos (Yo escogí el tipo .obj)

```
159
160     // --- Pivotes de rotación en Y para las patas ---
161     float yOffPata1 = 2.6f;
162     float yOffPata2 = 2.6f;
163     float yOffPata3 = 1.5f;
164     float yOffPata4 = 1.5f;
165
```

Ya que tuve problemas con el pivote para las rotaciones a pesar de dejarlos bien en blender, decidí mover el pivote desde el loop.

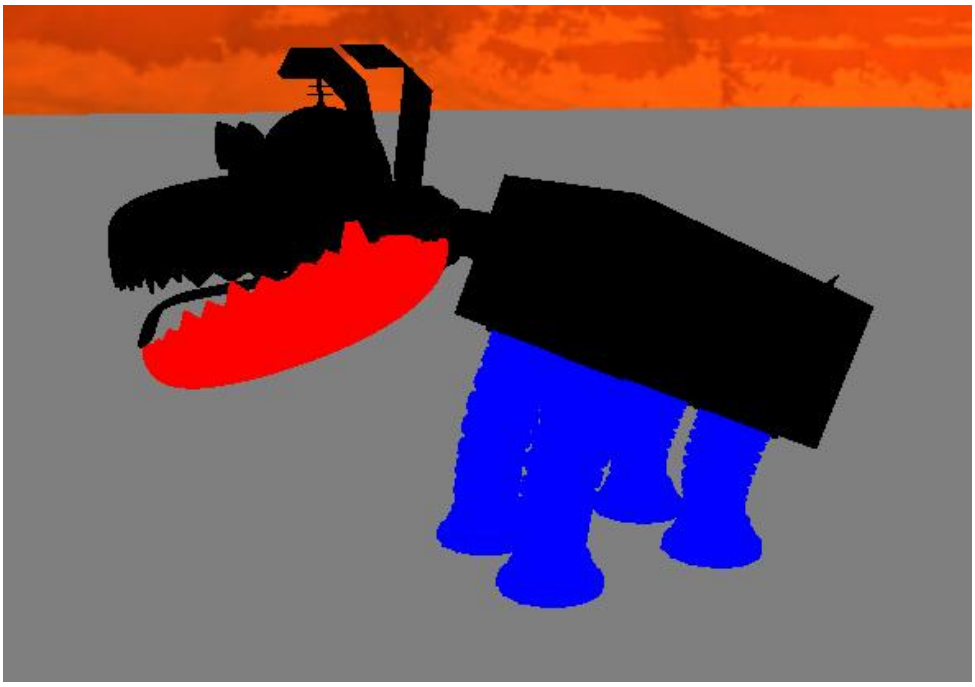
```
221         // ------------ GODDARD BASE ------------
222         color = glm::vec3(0.0f, 0.0f, 0.0f);
223         model = glm::mat4(1.0f);
224         model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
225         glUniform3fv(uniformColor, 1, glm::value_ptr(color));
226         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
227         Goddard_M.RenderModel();
228
229         // ------------ GODDARD MANDIBULA ------------
230         color = glm::vec3(1.0f, 0.0f, 0.0f);
231         model = glm::mat4(1.0f);
232         model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.0f));
233         glUniform3fv(uniformColor, 1, glm::value_ptr(color));
234         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
235         Goddard_mandibula.RenderModel();
236
237
238         // Color para las patas (azul)
239         color = glm::vec3(0.0f, 0.0f, 1.0f);
240         glUniform3fv(uniformColor, 1, glm::value_ptr(color));
241
242         // ------------ PATA 1 ------------
243         modelaux = model;                                       // parte desde la base
244         modelaux = glm::translate(modelaux, glm::vec3(0.0f, 0.0f, 0.0f)); // anclaje (ajústalo a tu modelo)
245         modelaux = glm::translate(modelaux, glm::vec3(0.0f, yOffPata1, 0.0f)); // SUBIR pivote
246         modelaux = glm::rotate(modelaux, glm::radians(mainWindow.getarticulacion1()), glm::vec3(0.0f, 0.0f, 1.0f)); // ROTAR
247         modelaux = glm::translate(modelaux, glm::vec3(0.0f, -yOffPata1, 0.0f)); // REGRESAR desde pivote
248         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(modelaux));
249         Goddard_pata1.RenderModel();
```

```
251         // ------------ PATA 2 ------------
252         modelaux = model;
253         modelaux = glm::translate(modelaux, glm::vec3(0.0f, 0.0f, 0.0f));
254         modelaux = glm::translate(modelaux, glm::vec3(0.0f, yOffPata2, 0.0f));
255         modelaux = glm::rotate(modelaux, glm::radians(mainWindow.getarticulacion2()), glm::vec3(0.0f, 0.0f, 1.0f));
256         modelaux = glm::translate(modelaux, glm::vec3(0.0f, -yOffPata2, 0.0f));
257         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(modelaux));
258         Goddard_pata2.RenderModel();
259
260         // ------------ PATA 3 ------------
261         modelaux = model;
262         modelaux = glm::translate(modelaux, glm::vec3(0.0f, 0.0f, 0.0f));
263         modelaux = glm::translate(modelaux, glm::vec3(0.0f, yOffPata3, 0.0f));
264         modelaux = glm::rotate(modelaux, glm::radians(mainWindow.getarticulacion3()), glm::vec3(0.0f, 0.0f, 1.0f));
265         modelaux = glm::translate(modelaux, glm::vec3(0.0f, -yOffPata3, 0.0f));
266         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(modelaux));
267         Goddard_pata3.RenderModel();
268
269         // ------------ PATA 4 ------------
270         modelaux = model;
271         modelaux = glm::translate(modelaux, glm::vec3(0.0f, 0.0f, 0.0f));
272         modelaux = glm::translate(modelaux, glm::vec3(0.0f, yOffPata4, 0.0f));
273         modelaux = glm::rotate(modelaux, glm::radians(mainWindow.getarticulacion4()), glm::vec3(0.0f, 0.0f, 1.0f));
274         modelaux = glm::translate(modelaux, glm::vec3(0.0f, -yOffPata4, 0.0f));
275         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(modelaux));
276         Goddard_pata4.RenderModel();
```

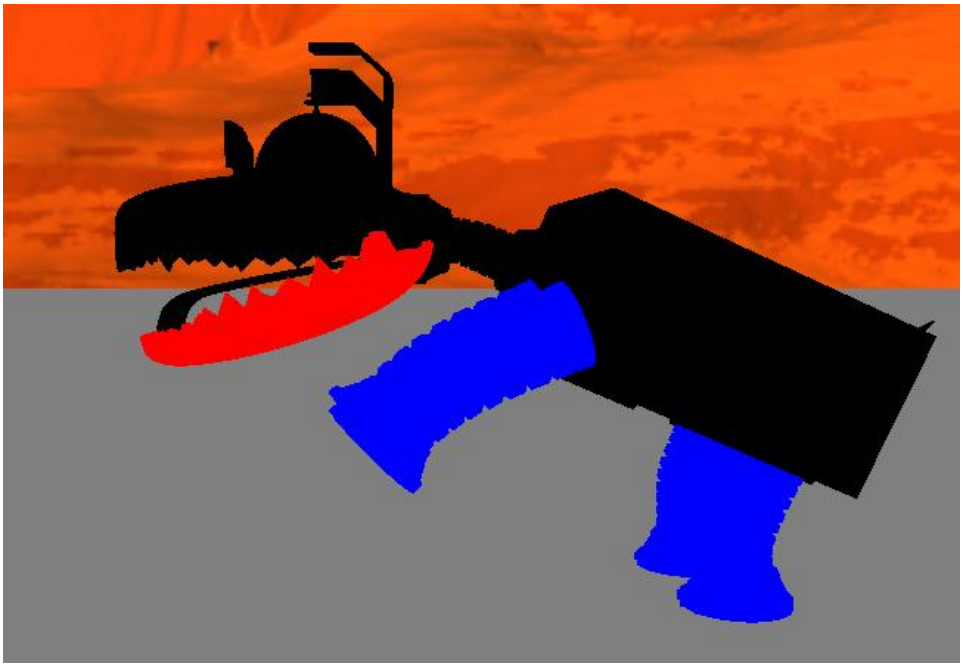
Definimos las propiedades de cada una de las patas con posiciones colores y agregamos la función de movimiento para cada una de las patas.

```
173 void Window::updateArticulaciones(float dt)
174 {
175     // SHIFT invierte sentido
176     int dir = (keys[GLFW_KEY_LEFT_SHIFT] || keys[GLFW_KEY_RIGHT_SHIFT]) ? -1 : 1;
177
178     auto avanzar = [&](float& ang, bool held) {
179         if (!held) return;
180         ang += dir * artSpeed * dt;
181         if (ang > artMax) ang = artMax;
182         if (ang < artMin) ang = artMin;
183         };
184
185     // Teclas 1..4 controlan cada pata
186     avanzar(articulacion1, keys[GLFW_KEY_1]);
187     avanzar(articulacion2, keys[GLFW_KEY_2]);
188     avanzar(articulacion3, keys[GLFW_KEY_3]);
189     avanzar(articulacion4, keys[GLFW_KEY_4]);
190
191     // 0 resetea
192     if (keys[GLFW_KEY_0]) {
193         articulacion1 = articulacion2 = articulacion3 = articulacion4 = 0.0f;
194     }
195 }
```

Dentro del window.cpp creamos el movimiento para las patas.



Asi la ejecución de cada modelo todo junto



Y así la animación para el movimiento de patas.

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

Tuve dificultades para posicionar el pivote de las patas, porque lo trabajé así en blender pero a la hora de animarlo en opengl se movía al centro del modelo y hacia una rotación que movía la pata fuera del modelo base.

Conclusión

Esta practica fue mi acercamiento al trabajar con modelos y pequeñas animaciones por teclado, sin embargo, aun tengo problemas para crear algunos efectos o transiciones. Ma allá de eso, logre aprender a trabajar modelos con blender ya sea en .obj o en. fbx. Personalmente sentí mas fácil trabajar con los .obj.