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: Araiza Valdés Diego Antonio

Nº de Cuenta: 423032833

GRUPO DE LABORATORIO: 02

GRUPO DE TEORÍA: 06

SEMESTRE: 2026-1

FECHA DE ENTREGA LÍMITE: 24/09/2026

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

   1.1. Actividad 1: Importar por separado y agregar jerarquía:
   - Cuerpo con cabeza y cola
   - mandíbula inferior
   - Cada una de las 4 patas independiente

```
Camera camera;
Model Cuerpo_M;
Model Mandibula_M;
Model PDI_M;
Model PDD_M;
Model PTI_M;
Model PTD_M;
```

```
Cuerpo_M = Model();
Cuerpo_M.LoadModel("Models/Cuerpo.obj");
Mandibula_M = Model();
Mandibula_M.LoadModel("Models/mandibula.obj");
PDI_M = Model();
PDI_M.LoadModel("Models/PataDelanteraIzquierda.obj");
PDD_M = Model();
PDD_M.LoadModel("Models/PataDelanteraDerecha.obj");
PTI_M = Model();
PTI_M.LoadModel("Models/PataTraseraIzquierda.obj");
PTD_M = Model();
PTD_M.LoadModel("Models/PataTraseraDerecha.obj");
```

Declaramos los modelos y los instanciamos con su archivo .obj correspondiente.

```
//------------*INICIA DIBUJO DE NUESTROS DEMÁS OBJETOS------------------*
//CUERPO
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, 0.435f, 0.0f));
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 1.0f, 0.0f));
color = glm::vec3(0.4f, 0.4f, 0.4f);
modelaux = model;
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 con cola

//MANDIBULA
model = modelaux;
model = glm::translate(model, glm::vec3(-2.95f, 0.9f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()), glm::vec3(0.0f, 0.0f, 1.0f));
color = glm::vec3(0.36f, 0.36f, 0.36f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Mandibula_M.RenderModel();
```

```
//pata delantera derecha
model = modelaux;
model = glm::translate(model, glm::vec3(-1.2f,-0.34f, -0.7f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()), glm::vec3(0.0f, 0.0f, 1.0f));
color = glm::vec3(0.30f, 0.30f, 0.30f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
PDD_M.RenderModel();

// pata delantera izquierda
model = modelaux;
model = glm::translate(model, glm::vec3(-1.2f, -0.34f, 0.7f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion3()), glm::vec3(0.0f, 0.0f, 1.0f));
color = glm::vec3(0.30f, 0.30f, 0.30f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
PDI_M.RenderModel();
//pata trasera derecha
model = modelaux;
model = glm::translate(model, glm::vec3(0.45f, -1.02f, -0.7f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion4()), glm::vec3(0.0f, 0.0f, 1.0f));
color = glm::vec3(0.30f, 0.30f, 0.30f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
PTD_M.RenderModel();

//pata trasera izquierda
model = modelaux;
model = glm::translate(model, glm::vec3(0.45f, -1.02f, 0.7f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion5()), glm::vec3(0.0f, 0.0f, 1.0f));
color = glm::vec3(0.30f, 0.30f, 0.30f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
PTI_M.RenderModel();

glUseProgram(0);

mainWindow.swapBuffers();
```

Posteriormente creamos una matriz nueva para crear el cuerpo, lo trasladamos hacia arriba y lo rotamos 180°, guardamos su posición en modelaux y a patir de ahí creamos las otras partes del cuerpo (la mandíbula y las 4 patas). Para cada una de estas partes, se les asignó su rotación correspondiente.

1.2. Actividad 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

```
// --- variables para el control (ping-pong) de articulacion1 ---
GLfloat articulacion1Dir;   // 1.0f = incrementar, -1.0f = decrementar
GLfloat articulacion1Min;   // límite mínimo
GLfloat articulacion1Max;   // límite máximo
GLfloat articulacion1Step;  // paso por pulsación

// --- variables para el control (ping-pong) de articulacion2 ---
GLfloat articulacion2Dir;   // 1.0f = incrementar, -1.0f = decrementar
GLfloat articulacion2Min;   // límite mínimo
GLfloat articulacion2Max;   // límite máximo
GLfloat articulacion2Step;  // paso por pulsación

// --- variables para el control (ping-pong) de articulacion3 ---
GLfloat articulacion3Dir;   // 1.0f = incrementar, -1.0f = decrementar
GLfloat articulacion3Min;   // límite mínimo
GLfloat articulacion3Max;   // límite máximo
GLfloat articulacion3Step;  // paso por pulsación

// --- variables para el control (ping-pong) de articulacion4 ---
GLfloat articulacion4Dir;   // 1.0f = incrementar, -1.0f = decrementar
GLfloat articulacion4Min;   // límite mínimo
GLfloat articulacion4Max;   // límite máximo
GLfloat articulacion4Step;  // paso por pulsación

// --- variables para el control (ping-pong) de articulacion5 ---
GLfloat articulacion5Dir;   // 1.0f = incrementar, -1.0f = decrementar
GLfloat articulacion5Min;   // límite mínimo
GLfloat articulacion5Max;   // límite máximo
GLfloat articulacion5Step;  // paso por pulsación
```

Para este ejercicio, copiamos y pegamos la estructura de la práctica anterior para cada una de las articulaciones, ya que en esta ya habíamos implementado una articulación delimitada por ángulos. En el archivo window.h se declararon las variables anteriores.

```
// Inicialización de control ping-pong para articulacion1
articulacion1Dir = 1.0f;    // empieza bajando
articulacion1Min = 0.0f;
articulacion1Max = 30.0f;
articulacion1Step = 5.0f;  // +5° por pulsación

// Inicialización de control ping-pong para articulacion2
articulacion2Dir = -1.0f;    // empieza subiendo
articulacion2Min = -45.0f;
articulacion2Max = 45.0f;
articulacion2Step = 5.0f;  // +5° por pulsación

// Inicialización de control ping-pong para articulacion3
articulacion3Dir = -1.0f;    // empieza subiendo
articulacion3Min = -45.0f;
articulacion3Max = 45.0f;
articulacion3Step = 5.0f;  // +5° por pulsación

// Inicialización de control ping-pong para articulacion4
articulacion4Dir = -1.0f;    // empieza subiendo
articulacion4Min = -45.0f;
articulacion4Max = 45.0f;
articulacion4Step = 5.0f;  // +5° por pulsación

// Inicialización de control ping-pong para articulacion5
articulacion5Dir = -1.0f;    // empieza subiendo
articulacion5Min = -45.0f;
articulacion5Max = 45.0f;
articulacion5Step = 5.0f;  // +5° por pulsación
```

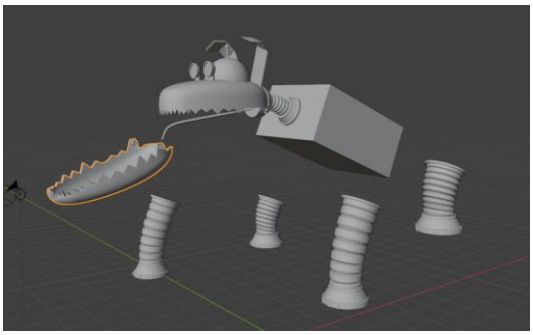
En el window.cpp inicializamos las variables para indicar el sentido del primero movimiento de la articulación y delimitarlo, de modo que la mandíbula se mueve de 0 a 30°, empezando a moverse hacia abajo y las patas van de -45° a 45°, empezando a moverse hacia arriba.

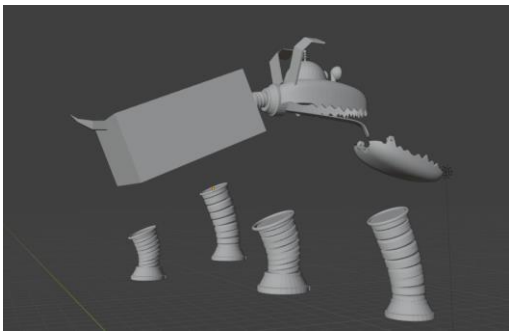
```
if (key == GLFW_KEY_F)
{
    float next = theWindow->articulacion1 + theWindow->articulacion1Dir * theWindow->articulacion1Step;

    if (next > theWindow->articulacion1Max) {
        theWindow->articulacion1 = theWindow->articulacion1Max;
        theWindow->articulacion1Dir = -1.0f; // invertir dirección para la siguiente pulsación
    }
    else if (next < theWindow->articulacion1Min) {
        theWindow->articulacion1 = theWindow->articulacion1Min;
        theWindow->articulacion1Dir = 1.0f;
    }
    else {
        theWindow->articulacion1 = next;
    }
}
if (key == GLFW_KEY_G)
{
    float next = theWindow->articulacion2 + theWindow->articulacion2Dir * theWindow->articulacion2Step;

    if (next > theWindow->articulacion2Max) {
        theWindow->articulacion2 = theWindow->articulacion2Max;
        theWindow->articulacion2Dir = -1.0f; // invertir dirección para la siguiente pulsación
    }
    else if (next < theWindow->articulacion2Min) {
        theWindow->articulacion2 = theWindow->articulacion2Min;
        theWindow->articulacion2Dir = 1.0f;
    }
    else {
        theWindow->articulacion2 = next;
    }
}
if (key == GLFW_KEY_H)
{
    float next = theWindow->articulacion3 + theWindow->articulacion3Dir * theWindow->articulacion3Step;

    if (next > theWindow->articulacion3Max) {
        theWindow->articulacion3 = theWindow->articulacion3Max;
        theWindow->articulacion3Dir = -1.0f; // invertir dirección para la siguiente pulsación
    }
    else if (next < theWindow->articulacion3Min) {
        theWindow->articulacion3 = theWindow->articulacion3Min;
        theWindow->articulacion3Dir = 1.0f;
    }
    else {
        theWindow->articulacion3 = next;
    }
}
if (key == GLFW_KEY_J)
{
    float next = theWindow->articulacion4 + theWindow->articulacion4Dir * theWindow->articulacion4Step;

    if (next > theWindow->articulacion4Max) {
        theWindow->articulacion4 = theWindow->articulacion4Max;
        theWindow->articulacion4Dir = -1.0f; // invertir dirección para la siguiente pulsación
    }
    else if (next < theWindow->articulacion4Min) {
        theWindow->articulacion4 = theWindow->articulacion4Min;
        theWindow->articulacion4Dir = 1.0f;
    }
    else {
        theWindow->articulacion4 = next;
    }
}
```

```
if (key == GLFW_KEY_K)
{
    float next = theWindow->articulacion5 + theWindow->articulacion5Dir * theWindow->articulacion5Step;

    if (next > theWindow->articulacion5Max) {
        theWindow->articulacion5 = theWindow->articulacion5Max;
        theWindow->articulacion5Dir = -1.0f; // invertir dirección para la siguiente pulsación
    }
    else if (next < theWindow->articulacion5Min) {
        theWindow->articulacion5 = theWindow->articulacion5Min;
        theWindow->articulacion5Dir = 1.0f;
    }
    else {
        theWindow->articulacion5 = next;
    }
}
```

Finalmente, dentro el mismo archivo, copiamos y pegamos la estructura que utilizamos en la práctica anterior para que cada vez que se rote una articulación comprobar su ángulo y de sobrepasarlo cambiar la dirección de rotación.

1.3. Evidencias:





2. Problemas presentados. Listar si surgieron problemas a la hora de ejecutar el código
   R: No hubo ningún problema.

3. Conclusión:
   3.1. Los ejercicios de la clase: Complejidad, explicación
   El ejercicio fue bastante adecuado, no hubo mayor complicación para su realización. Si bien, no entendí la jerarquía porque en el código venia repetido 4 veces modelaux = model, por cada pata, y ya no se hacía uso de model, mejor decidí hacer caso omiso.

   3.2. Comentarios generales: Faltó explicar a detalle, ir más lento en alguna explicación, otros comentarios y sugerencias.
   Respecto a la explicación, quedo algo inconforme, ya que esta vez fue la vez que más me perdí durante la explicación. Comentó que iba a explicar mientras configuramos pero empezó a explicar y se siguió de lleno. Además, igual que en el ejercicio anterior, con el ritmo que lleva durante la explicación, no es posible estar siguiendo todos los pasos y poniendo atención a la explicación al mismo tiempo, considero que lo ideal sería que realizara la explicación y nos pasara el código o modelo que realizó durante la explicación, así entenderíamos como trabajar y no nos atrasaríamos.