
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:** Uriarte Ortiz Enrique Yahir

**Nº de Cuenta:** 318234757

**GRUPO DE LABORATORIO:** 02

**GRUPO DE TEORÍA:** 04

**SEMESTRE 2025-1**

**FECHA DE ENTREGA LÍMITE:** Martes 17 de Septiembre del 2024

**CALIFICACIÓN: _____**

## *EJERCICIOS DE SESIÓN:*

***1.*** *Actividades realizadas.*

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

Para esta actividad lo primero fue separar el modelo de Goddard proporcionado en las partes que se solicitaron, utilizando las herramientas mostradas por el profesor en la sección, primero seleccionando el área del modelo a separar, después realizando un ***"detach"*** para guardarlas por separado, luego ajustando el centro de cada parte de acuerdo con cómo debía verse su rotación en la segunda actividad.

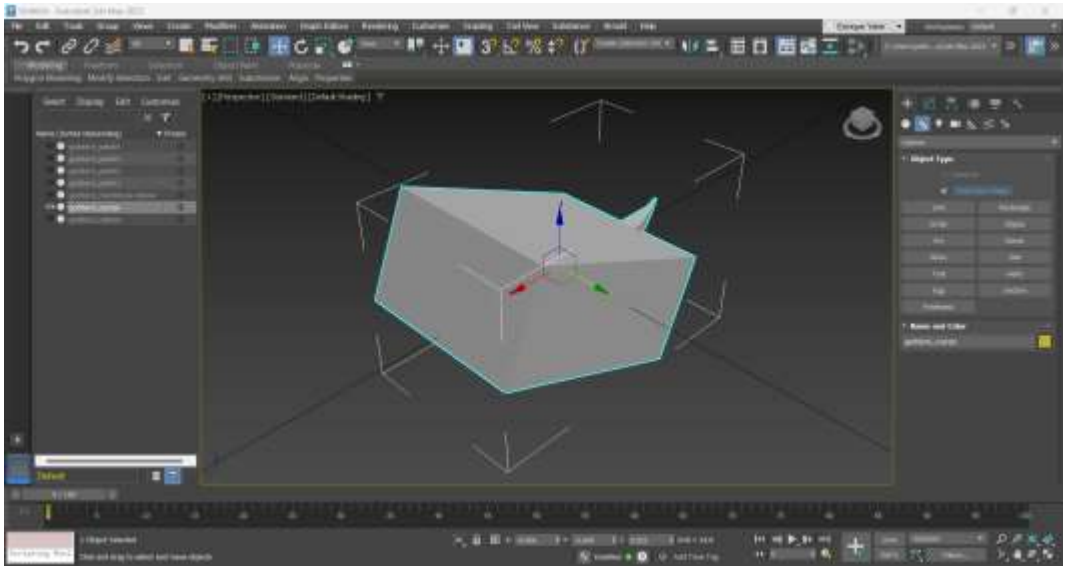

***Fig. 1.*** *Separación de modelo con 3ds Max 2023, cuerpo de goddard.*

***Fig. 2.*** *Separación de modelo con 3ds Max 2023, cabeza de goddard.*

Principalmente en la mandíbula y las 4 patas era necesario ajustar correctamente la posición de los centros, como lo visto en clase con la mandíbula.

***Fig. 3.*** *Separación de modelo con 3ds Max 2023, mandíbula de goddard.*

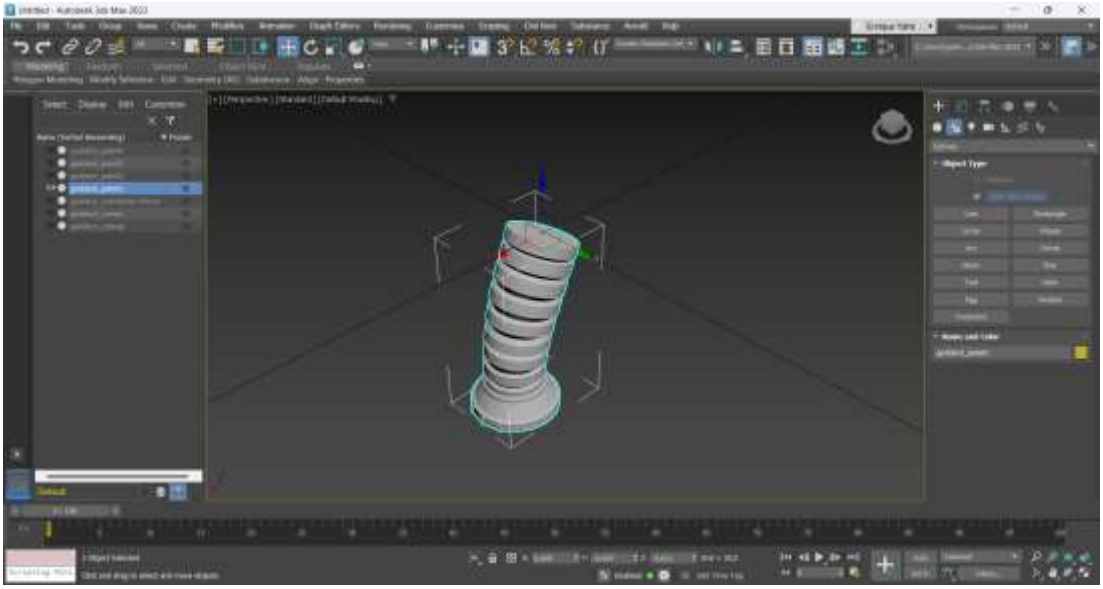

***Fig. 4.*** *Separación de modelo con 3ds Max 2023, pierna 1 de goddard.*

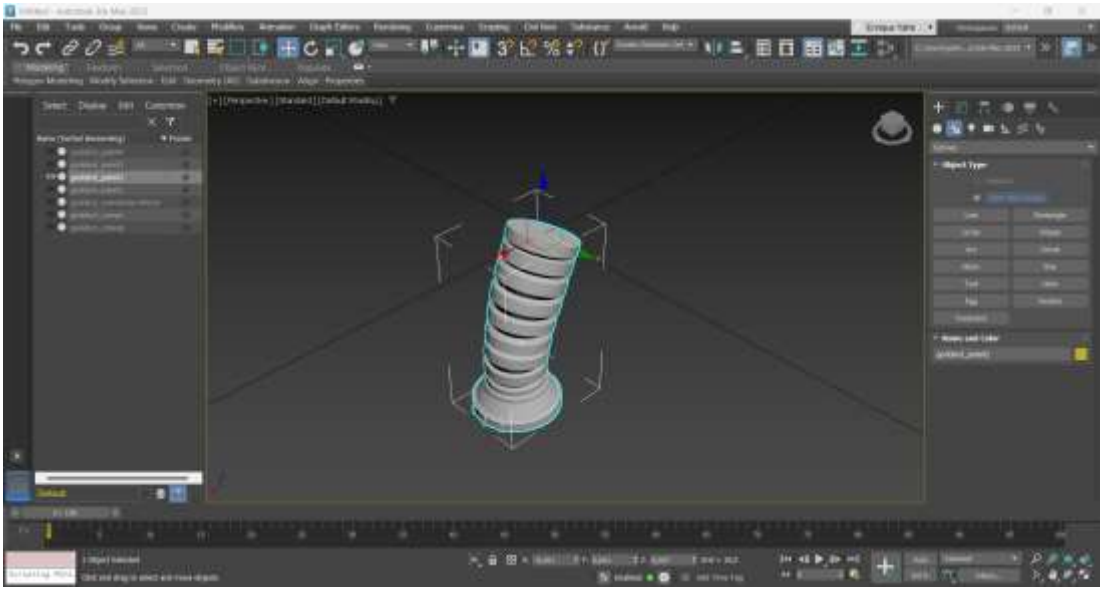

***Fig. 5.*** *Separación de modelo con 3ds Max 2023, pierna 2 de goddard.*

***Fig. 6.*** *Separación de modelo con 3ds Max 2023, pierna 3 de goddard.*

***Fig. 7.*** *Separación de modelo con 3ds Max 2023, pierna 4 de goddard.*

Decidí que al momento de exportarlos estos fueran de tipo .obj.

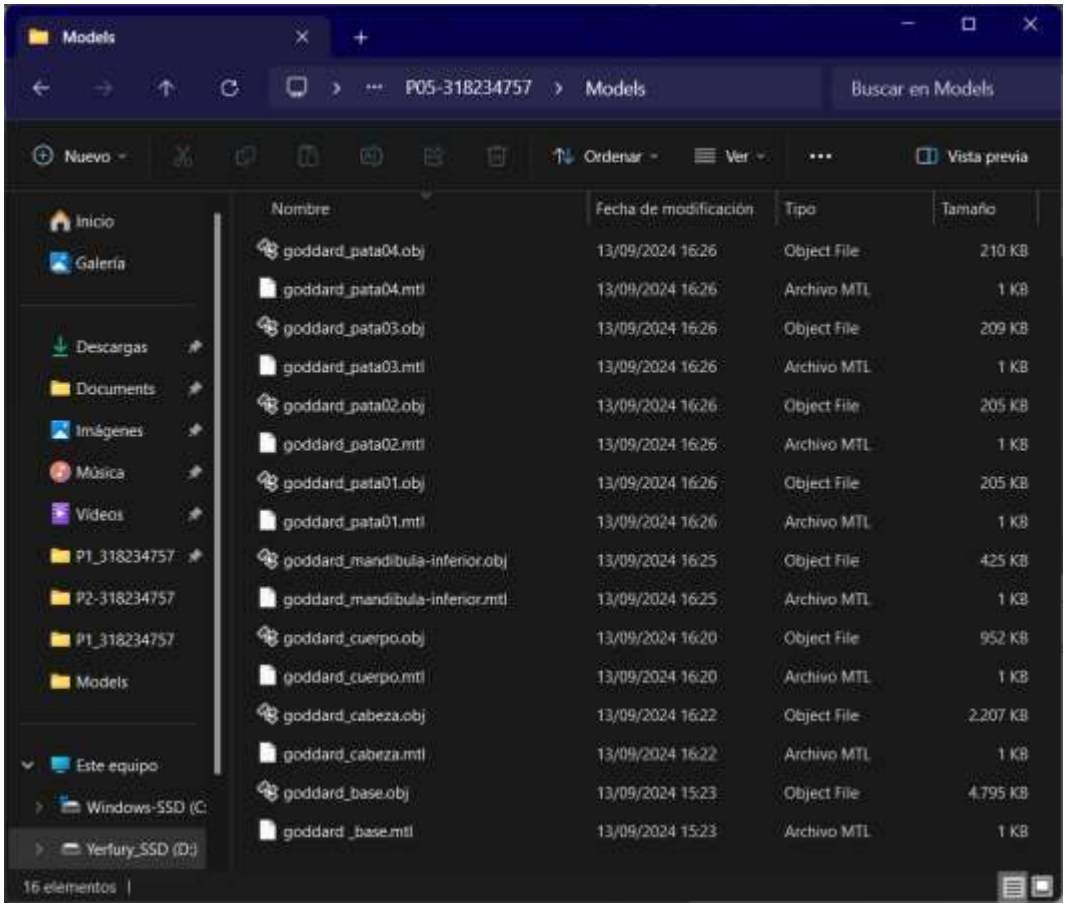


***Fig. 8.*** *Modelos de las partes de Goddard como archivos .obj.*

En esta actividad necesite 5 articulaciones, una para la mandíbula y una para cada pata, siendo estas las que se definen en los archivos ***"window.h"*** y ***"window.cpp"***.

```
GLfloat getarticulacion1() { return articulacion1; }
GLfloat getarticulacion2() { return articulacion2; }
GLfloat getarticulacion3() { return articulacion3; }
GLfloat getarticulacion4() { return articulacion4; }
GLfloat getarticulacion5() { return articulacion5; }
```

***Fig. 9.*** *Declaración de Articulaciones en archivo Window.h.*

```
private:
    GLFWwindow *mainWindow;
    GLint width, height;
    GLfloat rotax,rotay,rotaz, articulacion1, articulacion2, articulacion3, articulacion4, articulacion5;
```

***Fig. 10.*** *Declaración de Articulaciones en archivo Window.h.*

```
articulacion1 = 0.0f;
articulacion2 = 0.0f;
articulacion3 = 0.0f;
articulacion4 = 0.0f;
articulacion5 = 0.0f;
```

***Fig. 11.*** *Declaración de Articulaciones en archivo Window.cpp.*

Ya en el archivo principal declaro los modelos los model a los que les asignare cada uno de las partes de goodard, y despues mas abajo le asigno la ubicación dentro de la carpeta del proyecto donde se encuentran los modelos a emplear.

```
Camera camera;
Model Goddard_Cuerpo;
Model Goddard_Cabeza;
Model Goddard_MandInf;
Model Goddard_Pata01;
Model Goddard_Pata02;
Model Goddard_Pata03;
Model Goddard_Pata04;
Skybox skybox;
```

***Fig. 12.*** *Declaración de partes de goddard.*

```
Goddard_Cuerpo = Model();
Goddard_Cuerpo.LoadModel("Models/goddard_cuerpo.obj");
Goddard_Cabeza = Model();
Goddard_Cabeza.LoadModel("Models/goddard_cabeza.obj");
Goddard_MandInf = Model();
Goddard_MandInf.LoadModel("Models/goddard_mandibula-inferior.obj");
Goddard_Pata01 = Model();
Goddard_Pata01.LoadModel("Models/goddard_pata01.obj");
Goddard_Pata02 = Model();
Goddard_Pata02.LoadModel("Models/goddard_pata02.obj");
Goddard_Pata03 = Model();
Goddard_Pata03.LoadModel("Models/goddard_pata03.obj");
Goddard_Pata04 = Model();
Goddard_Pata04.LoadModel("Models/goddard_pata04.obj");
```

***Fig. 13.*** *Declaración de archivos .obj de las partes de goddard.*

Las jerarquias que utilizare seran una para cada elemento que tendra movimiento, ademas claro de la jerarquia original para el cuerpo y la cabez que no tendran moviento.

```
glm::mat4 model(1.0);       //Origen de las jererquias.
glm::mat4 mandibula(1.0);   //Jerarquia con mandibula.
glm::mat4 pata_01(1.0);     //Jerarquia con pata 1.
glm::mat4 pata_02(1.0);     //Jerarquia con pata 2.
glm::mat4 pata_03(1.0);     //Jerarquia con pata 3.
glm::mat4 pata_04(1.0);     //Jerarquia con pata 4.
```

***Fig. 14.*** *Jerarquías para cada parte de goddard.*

Modifique el color del piso a un tono mas anaranjado, esto para que se viera mas acorde con el escenario colocado desde mi punto de vista, ademas de que el tono gris lo iba a escoger mejor para todo el cuerpo de goddard.

```
color = glm::vec3(1.0f, 0.4f, 0.1f); //Dibujo de Piso.
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, -2.0f, 0.0f));
model = glm::scale(model, glm::vec3(30.0f, 1.0f, 30.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
meshList[2]->RenderMesh();
```

***Fig. 15.*** *Asignación de color de piso.*

Para empezar a declarar las partes de goddard, comenze con el cuerpo, le asigne un tono de gris neitro y lo ubique un poco mas arriba, asi mismo guarde la ubicación de translacion para las otras 5 jerarquias. Despues la siguiente parte del cuerpo de goddard en su babeza, esta sigue la jerarquia de mandibula, con “translate” la ajuste en posicion un poco mas adelante en X y Y, ademas de asignarle un tono de gris un poco mas claro para poder identificarlo.

```
//Goddard Base.
color = glm::vec3(0.5f, 0.5f, 0.5f);
model = glm::translate(model, glm::vec3(0.0f, 1.0f, 0.0f));
mandibula = model;
pata_01 = model;
pata_02 = model;
pata_03 = model;
pata_04 = model;
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Goddard_Cuerpo.RenderModel();

//Goddard Cabeza.
model = mandibula;
color = glm::vec3(0.6f, 0.6f, 0.6f);
model = glm::translate(model, glm::vec3(1.6f, 0.7f, 0.0f));
mandibula = model;
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Goddard_Cabeza.RenderModel();
```

***Fig. 16.*** *Declaración de cuerpo y cabeza de goddard.*

Despues sigue la mandibula, esta continua en la mis ma jerarquia que la cabeza, asigne un tono aun mas claro para idnetificarla mejor y despues de unos tanteos de medidas la acomode lo mejor posible con respecto a la cabeza, asi mismo le asigne la rotacion en -Z con una articulacion, para mas adelante con una tecla asignar el movimiento.

```
//Articulación - Goddard Mandibula.
model = mandibula;
color = glm::vec3(0.7f, 0.7f, 0.7f);
model = glm::translate(model, glm::vec3(1.52f, 0.25f, 0.852f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()), glm::vec3(0.0f, 0.0f, -1.0f));
mandibula = model;
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Goddard_MandInf.RenderModel();
```

***Fig. 17.*** *Declaración de mandíbula y su articulación para goddard.*

Para las patas delanteras, siendo estas la pata 1 y 2, utilize en cada una la jerarquia correspondiente, para el ajuste de translate la distancia es la misma en los ejes X y Y, pero en el eje Z es donde cambia el signo unicamente, ademas cada pata tiene el mismo tono que la mandibula. Ademas tambien le asigne a cada una la rotacion correspondiente para el eje Z

```
//Articulación - Pata01. Izquierda.
model = pata_01;
color = glm::vec3(0.7f, 0.7f, 0.7f);
model = glm::translate(model, glm::vec3(1.5f, -0.2f, 0.7f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()), glm::vec3(0.0f, 0.0f, -1.0f));
pata_01 = model;
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Goddard_Pata01.RenderModel();

//Articulación - Pata02. Derecha.
model = pata_02;
color = glm::vec3(0.7f, 0.7f, 0.7f);
model = glm::translate(model, glm::vec3(1.5f, -0.2f, -0.7f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion3()), glm::vec3(0.0f, 0.0f, -1.0f));
pata_02 = model;
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Goddard_Pata02.RenderModel();
```

***Fig. 18.*** *Declaración de pata 1 y 2 de goddard con sus articulaciones.*

Para las patas traseras, la pata 3 y 4, es la misa estrcutura, solo ajustando las posiciones, las jererquias y las rotaciones.

```
//Articulación - Pata03. Izquierda.
model = pata_03;
color = glm::vec3(0.7f, 0.7f, 0.7f);
model = glm::translate(model, glm::vec3(-0.1f, -0.9f, 0.7f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion4()), glm::vec3(0.0f, 0.0f, -1.0f));
pata_03 = model;
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Goddard_Pata03.RenderModel();

//Articulación - Pata04. Derecha.
model = pata_04;
color = glm::vec3(0.7f, 0.7f, 0.7f);
model = glm::translate(model, glm::vec3(-0.1f, -0.9f, -0.7f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion5()), glm::vec3(0.0f, 0.0f, -1.0f));
pata_04 = model;
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Goddard_Pata04.RenderModel();
```

***Fig. 18.*** *Declaración de pata 3 y 4 de goddard con sus articulaciones.*

***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 limitar la rotacion de cada articulacion modifique las intrucciones para asignar el movimiento a cada tecla en el archivo “Window.cpp”, dentro de este solo le asigne un if, donde si el angulo de rotacion es mayor a 45° no haga nada, pero si no que ejecute el movimeinto con la tecla, para asignar el movimiento de direccion contraria, le asigno las mismas indicaciones pero cambiandole el angulo a que sea 0° a otra tecla. De este modo limito la rotacion de la mandibula, con Z para moverla hacia abajo y con X apara moverla a su poscion original.

```
//Rotar la mandibula.
if (key == GLFW_KEY_Z) {
    if (theWindow->articulacion1 > 45.0) {}
    else {theWindow->articulacion1 += 2.0;}}
if (key == GLFW_KEY_X){
    if (theWindow->articulacion1 < 0.0){}
    else {theWindow->articulacion1 -= 2.0;}}
```

***Fig. 19.*** *Declaración de teclas para la rotación limitada de la mandíbula.*

Para las patas es la misma estructura, estas pueden rotar 45° tanto hacia adelante como hacia atrás.

Con K para mover hacia atrás y con L para mover hacia adelante la pata 1.

Con G para mover hacia atrás y con H para mover hacia adelante la pata 2.

Con N para mover hacia atrás y con M para mover hacia adelante la pata 3.

Con C para mover hacia atrás y con V para mover hacia adelante la pata 4.

```
//Rotar la pata 01.
if (key == GLFW_KEY_K) {
    if (theWindow->articulacion2 > 45.0) {}
    else { theWindow->articulacion2 += 2.0;}}
if (key == GLFW_KEY_L) {
    if (theWindow->articulacion2 < -45.0) {}
    else { theWindow->articulacion2 -= 2.0;}}

//Rotar la pata 02.
if (key == GLFW_KEY_G) {
    if (theWindow->articulacion3 > 45.0) {}
    else { theWindow->articulacion3 += 2.0;}}
if (key == GLFW_KEY_H) {
    if (theWindow->articulacion3 < -45.0) {}
    else { theWindow->articulacion3 -= 2.0;}}
```

```
//Rotar la pata 03.
if (key == GLFW_KEY_N) {
    if (theWindow->articulacion4 > 45.0) {}
    else { theWindow->articulacion4 += 2.0;}}
if (key == GLFW_KEY_M) {
    if (theWindow->articulacion4 < -45.0) {}
    else { theWindow->articulacion4 -= 2.0;}}

//Rotar la pata 04.
if (key == GLFW_KEY_C) {
    if (theWindow->articulacion5 > 45.0) {}
    else { theWindow->articulacion5 += 2.0;}}
if (key == GLFW_KEY_V) {
    if (theWindow->articulacion5 < -45.0) {}
    else { theWindow->articulacion5 -= 2.0;}}
```

***Fig. 20 y Fig. 21.*** *Declaración de teclas para la rotación limitada de las patas.*

Ejecucion del programa:

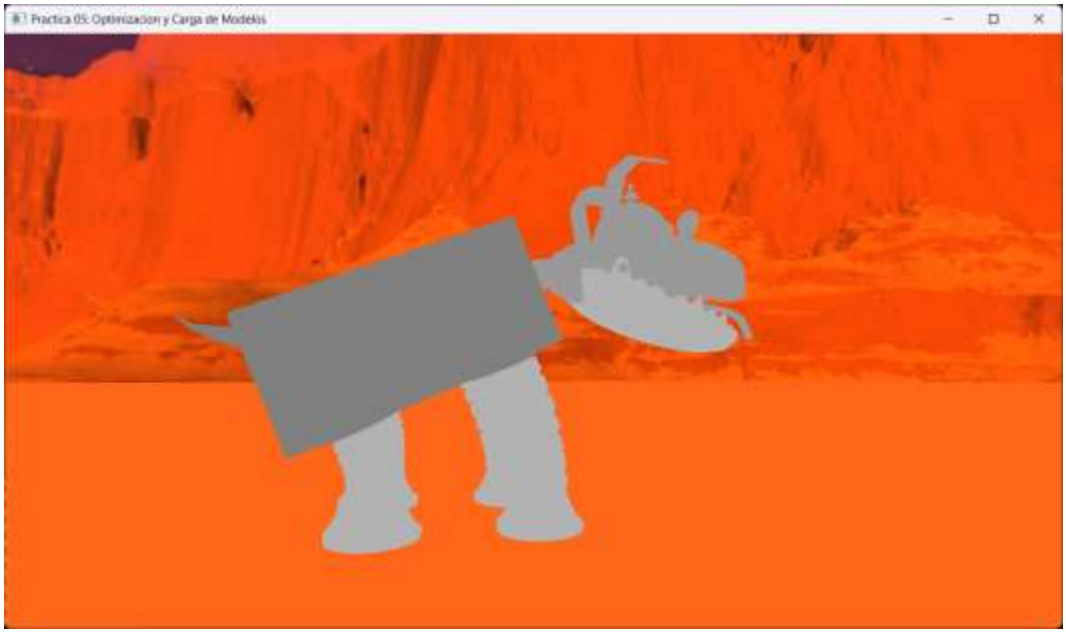


***Fig. 22.*** *Ejecución de Programa.*

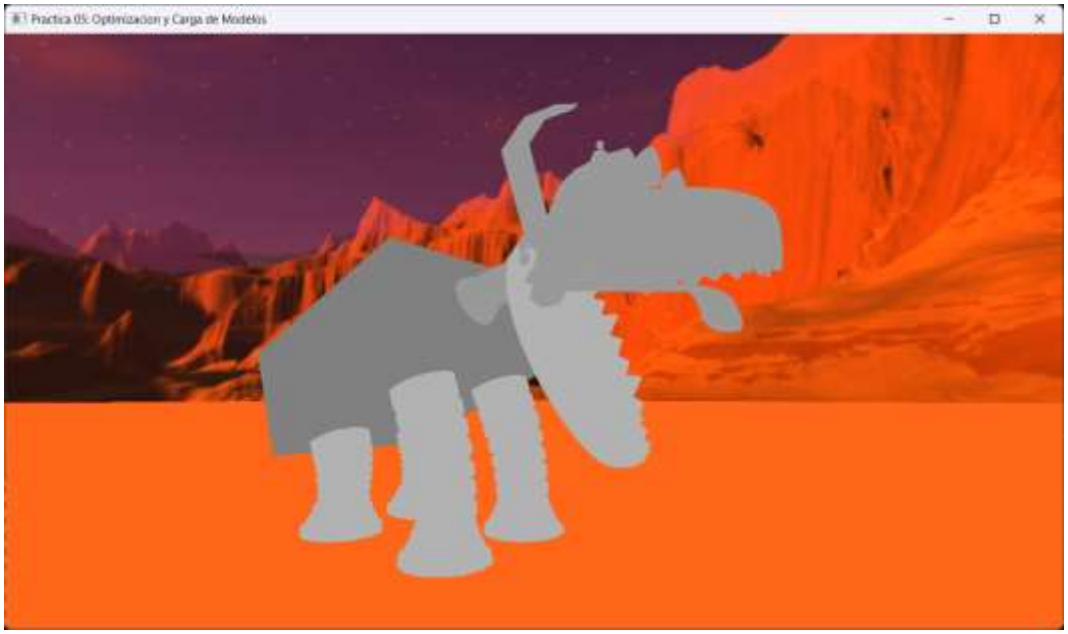


***Fig. 23.*** *Ejecución de Programa, mandíbula a 45°.*


***Fig. 24.*** *Ejecución de Programa, pierna 1, 2, 3 y 4.*


***Fig. 25.*** *Ejecución de Programa, pierna 1, 2, 3 y 4.*

***2.*** *Problemas presentados.*

Para estos ejercicios no presente problemas, lo más complicado fue ajustar la posición de cada elemento del cuerpo ya que lo demás de asignar movimientos y jerarquías me quedo bastante claro con la practica anterior, por lo demás no tuve otros problemas.

***3.*** *Conclusión:*

Después de realizar el ejercicio propuesto, considero que la complejidad que tenían los ejercicios era normal, realmente se pareció mucho a mi parecer a las actividades que realizamos en la práctica anterior, solo agregándole en esta ocasión la implementación de modelos separados en 3ds Max, en lugar de figuras generadas con el mismo código, las 2 actividades se entendieron muy bien, y nuevamente fue un gran apoyo en casa los videos de explicación ya que en lo personal si suelo olvidar algunos pequeños detalles de las explicaciones de clase. Al final considero que la practica tuvo un buen nivel, al integrar elementos extra para poder agregar modelos en 3d en lugar de las figuras que solíamos realizar, estos conceptos agregados en los ejercicios me parecen una aplicación futura en el proyecto final para limitar los movimientos a personajes u objetos.