UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE INGENIERÍA ELÉCTRICA

INGENIERÍA EN COMPUTACIÓN

LABORATORIO DE COMPUTACIÓN GRÁFICA e INTERACCIÓN HUMANO COMPUTADORA

REPORTE DE PRÁCTICA Nº 06

NOMBRE COMPLETO: MORENO SANTOYO MARIANA

Nº de Cuenta: 319170252

GRUPO DE LABORATORIO: 11

GRUPO DE TEORÍA: 04

SEMESTRE 2025-2

FECHA DE ENTREGA LÍMITE: 27/03/2025

CALIFICACIÓN: _____

REPORTE DE PRÁCTICA:

1.- Ejecución de los ejercicios que se dejaron, comentar cada uno y capturas de pantalla de bloques de código generados y de ejecución del programa.

2.- Liste los problemas que tuvo a la hora de hacer estos ejercicios y si los resolvió explicar cómo fue, en caso de error adjuntar captura de pantalla

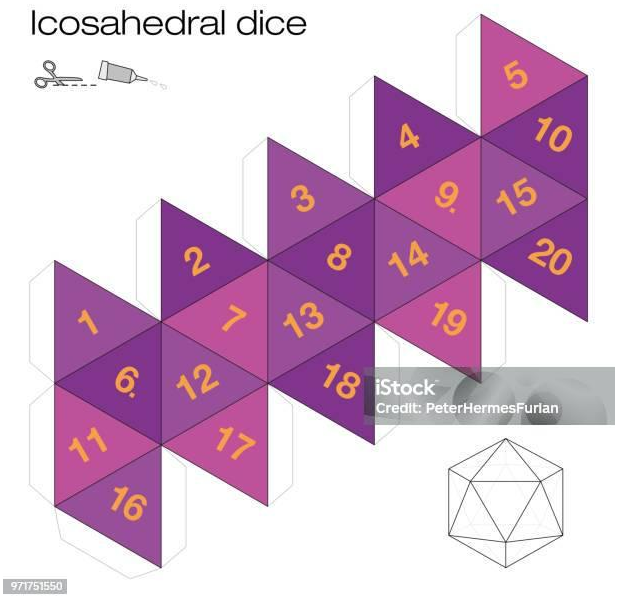
3.- Conclusión:

a. *Los ejercicios del reporte: Complejidad, Explicación.*
b. *Comentarios generales: Faltó explicar a detalle, ir más lento en alguna explicación, otros comentarios y sugerencias para mejorar desarrollo de la práctica*
c. *Conclusión*

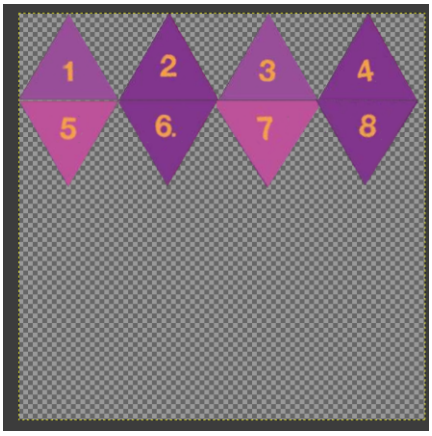
1. *Bibliografía en formato APA*

- Ejercicio 1: Crear un dado de 8 caras y texturizarlo por medio de código.

Para abordar correctamente el ejercicio del cubo de ocho caras, lo primero fue analizar su geometría. En este caso, se trataba de un octaedro, una figura compuesta por ocho triángulos equiláteros. Posteriormente, para la texturización del modelo, se buscó una imagen imprimible adecuada. Sin embargo, al no encontrar una imagen específica del octaedro, se optó por utilizar una imagen disponible en Internet correspondiente a un icosaedro de color morado.



Se decidió entonces trabajar directamente con los triángulos, y, de manera similar al ejercicio realizado en clase, se procedió a rotar los números en cada cara si era necesario, con el fin de que se vieran en la orientación correcta. Además, se hicieron ajustes como la eliminación de espacios en blanco y otros detalles que permitieran una presentación más limpia y funcional del modelo.



Además, se optó por esta configuración para poder espejear las coordenadas del triángulo superior con el inferior y, al momento de aplicar la textura mediante código, evitar posibles errores, realizando únicamente algunos ajustes menores. También se eliminaron los espacios en blanco y se corrigieron otros detalles para lograr una presentación más limpia y funcional del modelo.

```
void CrearDado_OCHO()
{
    // Definición de los índices (8 caras * 3 vértices = 24 índices)
    unsigned int octahedron_indices[] = {
        0,  1,  2,   // Cara 1 (T1)
        3,  4,  5,   // Cara 2 (T2)
        6,  7,  8,   // Cara 3 (T3)
        9, 10, 11,   // Cara 4 (T4)
       12, 13, 14,   // Cara 5 (T5)
       15, 16, 17,   // Cara 6 (T6)
       18, 19, 20,   // Cara 7 (T7)
       21, 22, 23    // Cara 8 (T8)
    };


    GLfloat octahedron_vertices[] = {

    //1
     0.0f,   0.0f,  0.5f,    0.05f, 0.79f,    0.5774f,  0.5774f,  0.5774f,
     0.5f,   0.0f,  0.0f,    0.23f, 0.79f,    0.5774f,  0.5774f,  0.5774f,
     0.0f,   0.5f,  0.0f,    0.125f, 1.0f,    0.5774f,  0.5774f,  0.5774f,
    //4
     0.0f,  0.0f,  0.5f,    0.96f,  0.79f,   -0.5774f,  0.5774f,  0.5774f,
     0.0f,  0.5f,  0.0f,    0.86f,  1.0f,    -0.5774f,  0.5774f,  0.5774f,
    -0.5f,  0.0f,  0.0f,    0.75f,  0.79f,   -0.5774f,  0.5774f,  0.5774f,
    //8
     0.0f,  0.0f,  0.5f,    0.96f, 0.77f,    -0.5774f, -0.5774f,  0.5774f,
    -0.5f,  0.0f,  0.0f,    0.75f, 0.77f,    -0.5774f, -0.5774f,  0.5774f,
     0.0f, -0.5f,  0.0f,    0.85f, 0.60f,    -0.5774f, -0.5774f,  0.5774f,
    //5
     0.0f,  0.0f,  0.5f,    0.05f,  0.77f,     0.5774f, -0.5774f,  0.5774f,
     0.0f, -0.5f,  0.0f,    0.125f, 0.60f,     0.5774f, -0.5774f,  0.5774f,
     0.5f,  0.0f,  0.0f,    0.23f,  0.77f,     0.5774f, -0.5774f,  0.5774f,
    //2
     0.0f,  0.0f, -0.5f,    0.47f, 0.79f,      0.5774f,  0.5774f, -0.5774f,
     0.0f,  0.5f,  0.0f,    0.37f,  0.99f,     0.5774f,  0.5774f, -0.5774f,
     0.5f,  0.0f,  0.0f,    0.28f,  0.79f,     0.5774f,  0.5774f, -0.5774f,
    //3
     0.0f,  0.0f, -0.5f,    0.51f, 0.79f,     -0.5774f,  0.5774f, -0.5774f,
    -0.5f,  0.0f,  0.0f,    0.69f,  0.79f,    -0.5774f,  0.5774f, -0.5774f,
     0.0f,  0.5f,  0.0f,    0.615f,   1.0f,   -0.5774f,  0.5774f, -0.5774f,
    //7
     0.0f,  0.0f, -0.5f,   0.51f, 0.78f,      -0.5774f, -0.5774f, -0.5774f,
     0.0f, -0.5f,  0.0f,   0.615f,  0.60f,    -0.5774f, -0.5774f, -0.5774f,
    -0.5f,  0.0f,  0.0f,   0.69f,  0.78f,     -0.5774f, -0.5774f, -0.5774f,
    //6
     0.0f,  0.0f, -0.5f,   0.45f, 0.78f,      0.5774f, -0.5774f, -0.5774f,
     0.5f,  0.0f,  0.0f,   0.28f,   0.78f,      0.5774f, -0.5774f, -0.5774f,
     0.0f, -0.5f,  0.0f,   0.37f,  0.60f,      0.5774f, -0.5774f, -0.5774f
    };

    calcAverageNormals(octahedron_indices, 24, octahedron_vertices, 192, 8, 5);
    Mesh* dado_OCHO = new Mesh();
    dado_OCHO->CreateMesh(octahedron_vertices, octahedron_indices, 192, 24);
    meshList.push_back(dado_OCHO);
}
```

La realización del modelo del octaedro resultó sencilla una vez que se comprendió su estructura: básicamente, se compone de dos pirámides triangulares unidas por su base, una orientada hacia arriba y la otra hacia abajo. Con esta comprensión, se logró desarrollar el siguiente código final tanto para las texturas como para la construcción del modelo de la pirámide.

Se cargo la textura

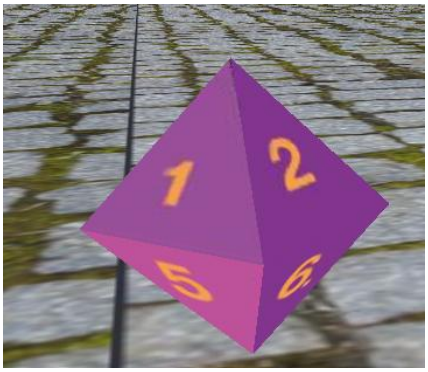
```
Texture logofiTexture;
Texture Dado_Ocho;
```

```
Dado_Ocho = Texture("Textures/cubocompleto.tga");
Dado_Ocho.LoadTextureA();
```

Se manda llamar al modelo con su textura

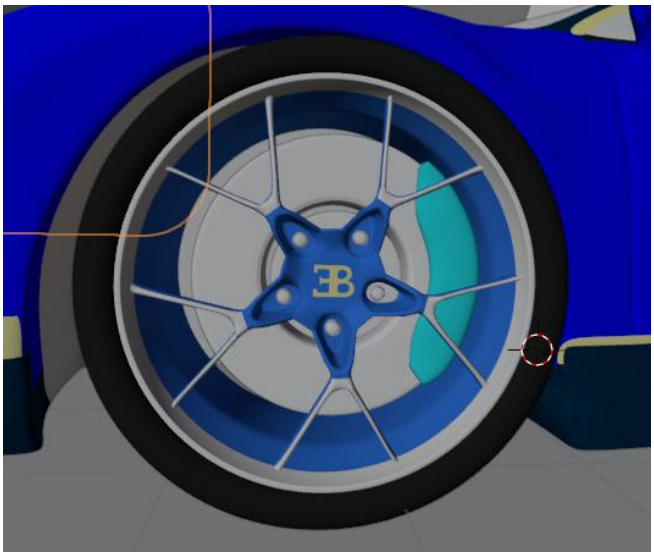
```
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-1.5f, 4.5f, -2.0f));
model = glm::scale(model, glm::vec3(2.0f, 2.0f, 2.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Dado_Ocho.UseTexture();
meshList[5]->RenderMesh();
```

Resultados



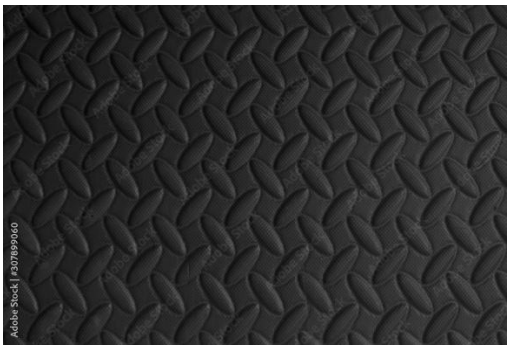
- **Ejercicio 2: Importar el modelo de su coche con sus 4 llantas acomodadas y tener texturizadas las 4 llantas (diferenciar caucho y rin)**

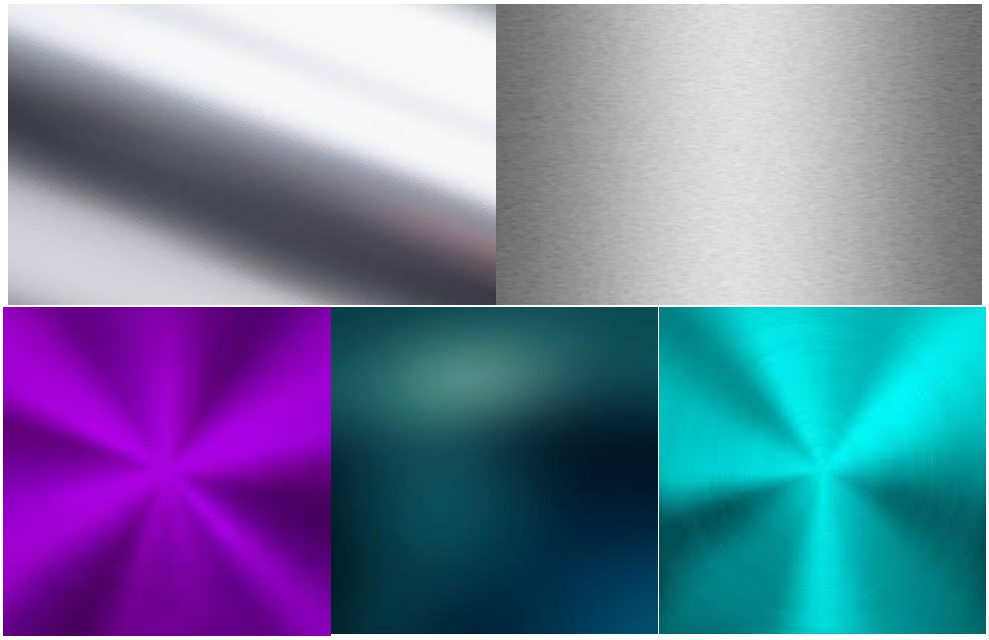
Para esta parte del ejercicio, la texturización se realizó utilizando Blender. Partiendo del modelo original, lo único que se modificó fue la aplicación de nuevas texturas con el objetivo de asignar distintos colores a las partes del automóvil, en especial a las llantas. Se optó por un esquema de colores compuesto por naranja, morado, verde aqua, verde aqua en un tono más oscuro y gris metálico para las partes metálicas del vehículo. Esta elección permitió darle una apariencia más atractiva y variada al modelo sin alterar su estructura base.



LLANTA ORIGINAL

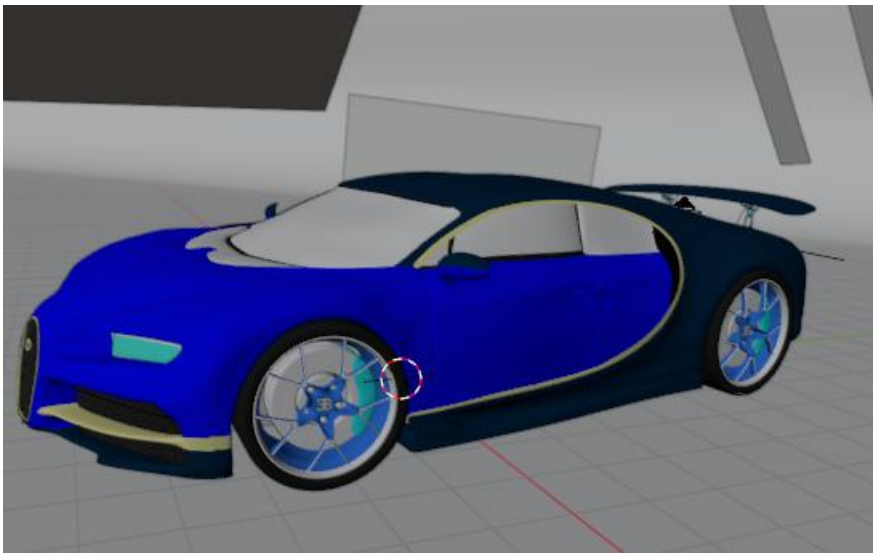
RESULTADO



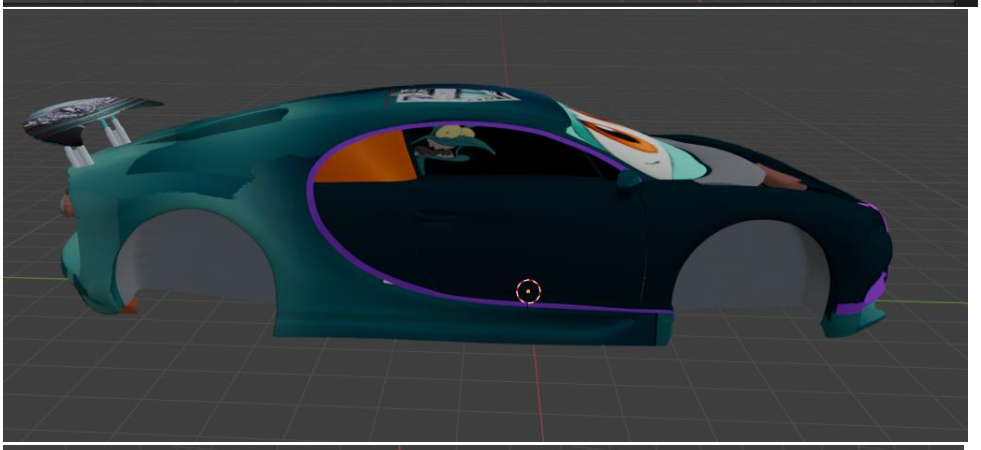


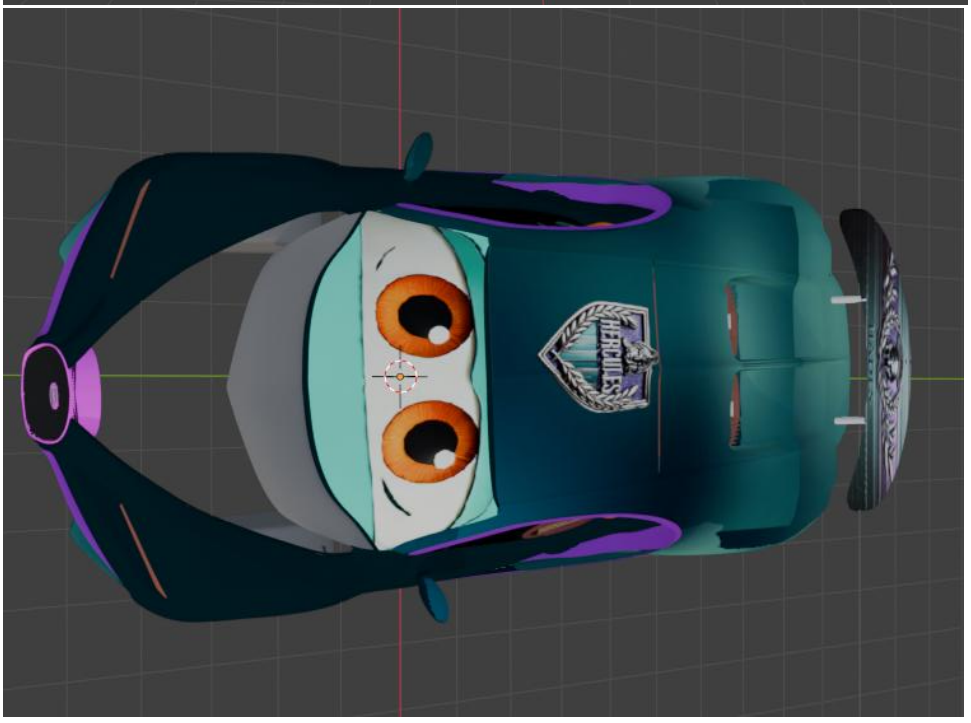
TEXTURAS

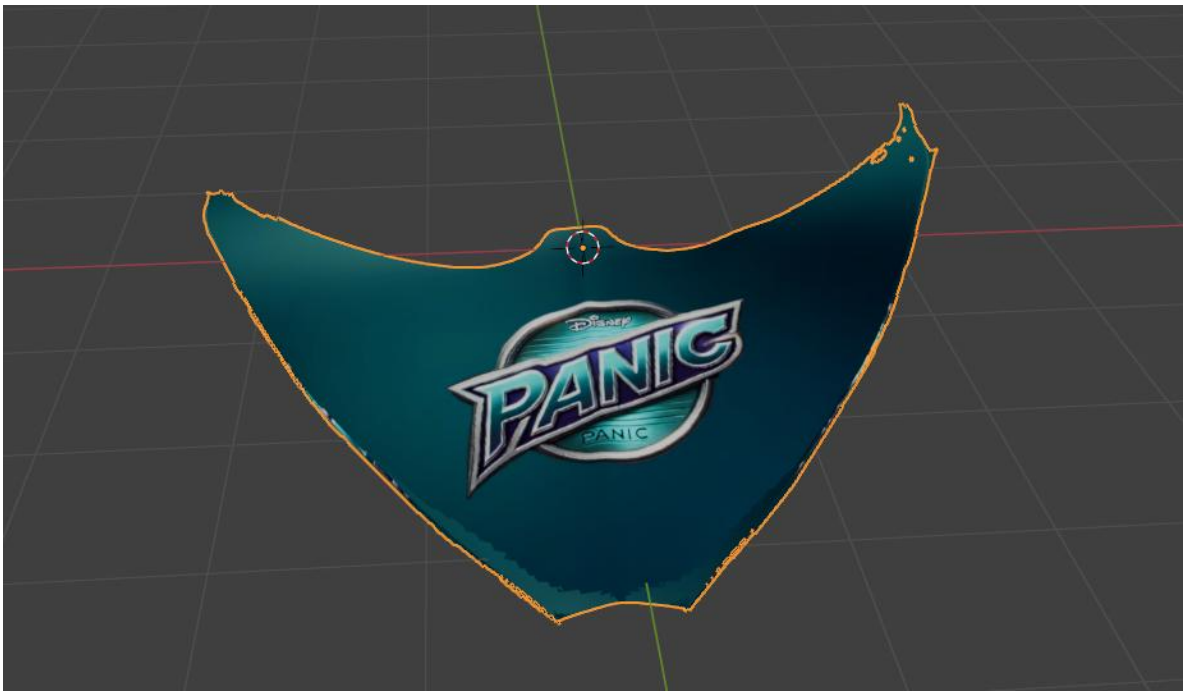
- Ejercicio 3: Texturizar la cara del personaje de la imagen tipo cars en el espejo (ojos) y detalles en cofre de su propio modelo de coche




Object Mode
View
Select
Add
Object
Global
Options
User Perspective
(1) Collection | Text.002_Text.002




HERCULES



MODELO JERARQUICO

```
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, -1.0f, 0.0f));
model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.1f) * mainWindow.getarticulacion1()); //traslacion del cuerpo adelante
model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.1f) * mainWindow.getarticulacion2()); //traslacion del cuerpo hacia atras
modelaux = model;

glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Buggatti_M.RenderModel();

model = modelaux;

model = glm::translate(model, glm::vec3(2.5f, 0.7f, 2.825f));
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()), glm::vec3(1.0f, 0.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
BuggattilLanta_M.RenderModel();

model = modelaux;

model = glm::translate(model, glm::vec3(-2.5f, 0.7f, 2.825f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()), glm::vec3(1.0f, 0.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
BuggattilLanta_M.RenderModel();

model = modelaux;

model = glm::translate(model, glm::vec3(-2.5f, 0.7f, -4.975f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()), glm::vec3(1.0f, 0.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
BuggattilLanta_M.RenderModel();

model = modelaux;

model = glm::translate(model, glm::vec3(2.5f, 0.7f, -4.975f));
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()), glm::vec3(1.0f, 0.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
BuggattilLanta_M.RenderModel();

model = modelaux;

model = glm::translate(model, glm::vec3(0.3f, 1.6f, 2.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion3()), glm::vec3(1.0f, 0.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Buggatticapo_M.RenderModel();
```