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 05

NOMBRE COMPLETO: Moreno Santoyo Mariana

Nº de Cuenta: 319170252

GRUPO DE LABORATORIO: 11

GRUPO DE TEORÍA: 04

SEMESTRE 2025-2

FECHA DE ENTREGA LÍMITE: 20 de marzo 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:

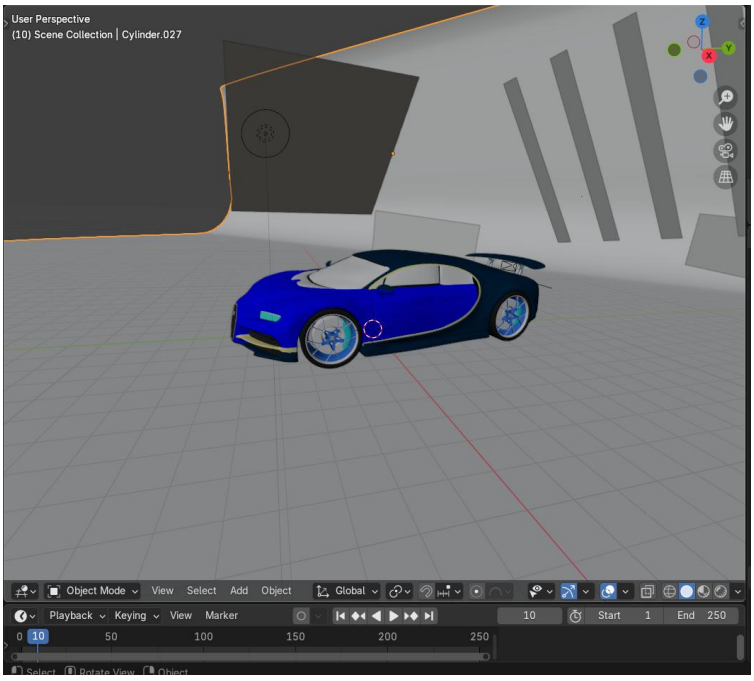
a. *Los ejercicios del reporte: Complejidad, Explicación.*
b. *Comentarios generales: Faltó explicar a detalle, ir más lento en alguna explicación, otros comentarios y sugerencias para mejorar desarrollo de la práctica*
c. *Conclusión*

1. *Bibliografía en formato APA*

1.- Importar su modelo de coche propio dentro del escenario a una escala adecuada.
2.- Importar sus 4 llantas y acomodarlas jerárquicamente, agregue el mismo valor de rotación a las llantas para que al presionar puedan rotar hacia adelante y hacia atrás.
3.- Importar el cofre del coche, acomodarlo jerárquicamente y agregar la rotación para poder abrir y cerrar.
4.- Agregar traslación con teclado para que pueda avanzar y retroceder de forma independiente

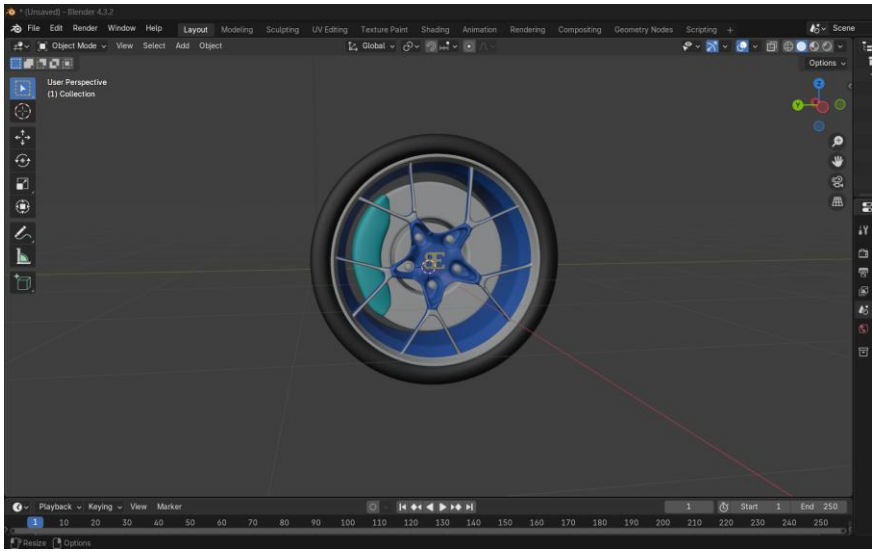
Para comenzar, se realizó la división del modelo en la plataforma Blender, tomando como referencia el video proporcionado por el profesor en Classroom. En mi caso, el modelo tenía una alta densidad de polígonos (high poly) en prácticamente todos sus elementos. Además, incluía algunos objetos de fondo, luces y otros componentes cuya función aún desconozco.

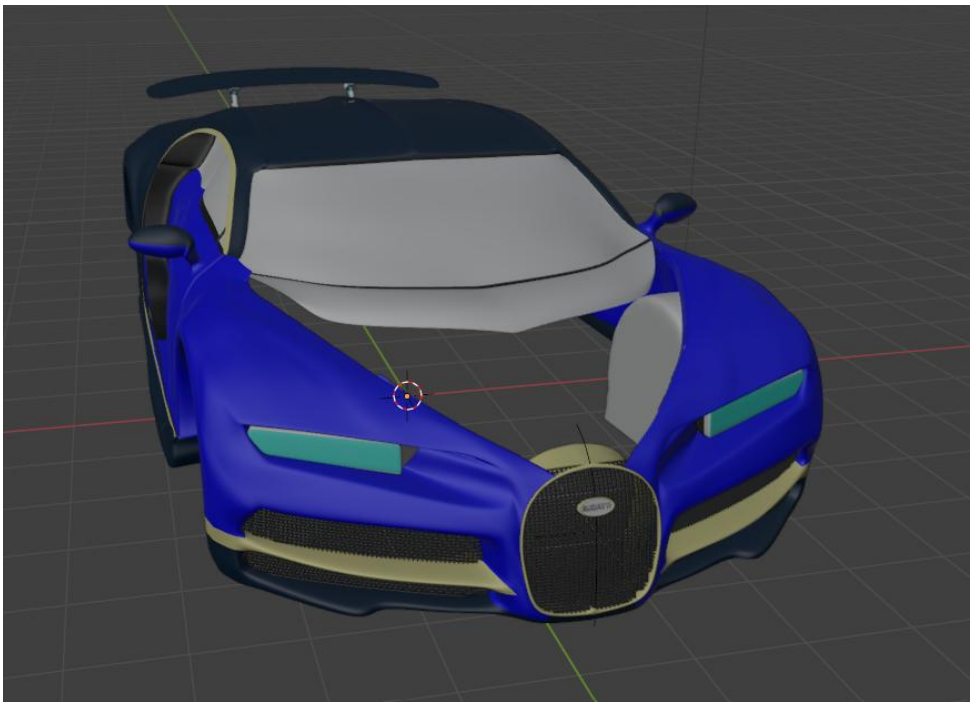
Lo que agregó dificultad a este proceso fue la eliminación de ciertos elementos que no sabía que podían afectar su traslado a OpenGL, como algunas líneas, polígonos de formas desconocidas y puntos de luz que aparecían marcados de esa manera en Blender.



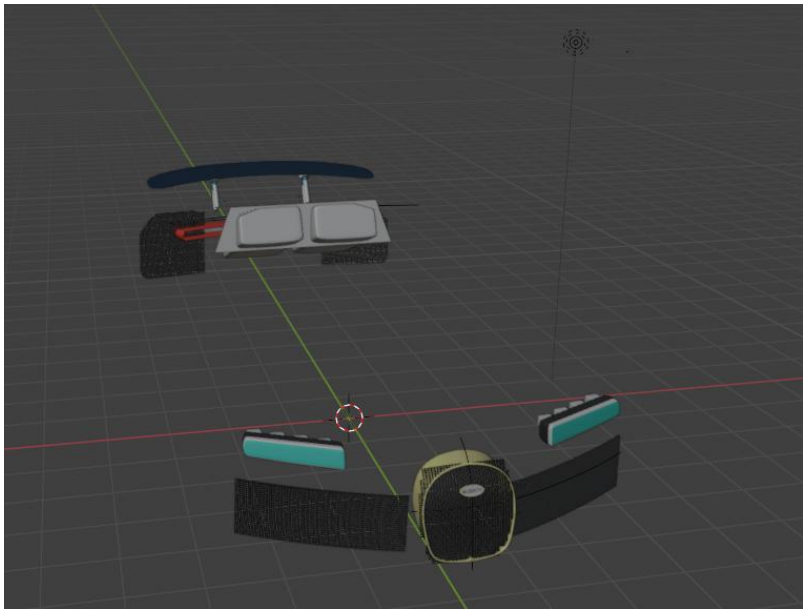
Se realizó la separación del modelo en tres partes: una para el cofre, otra para las llantas y otra para la carrocería. La separación de las llantas fue un proceso sencillo, pero el cofre representó un desafío considerable. Esto se debió a que Blender no lo detectaba como una geometría externa o una parte independiente de la carrocería.

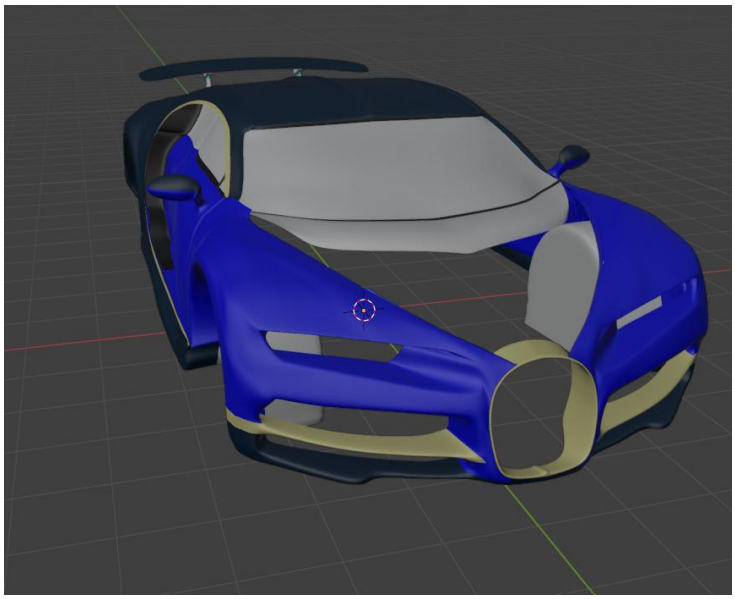
Dado mi desconocimiento sobre este tipo de automóviles, tuve que utilizar una herramienta similar a un lápiz de dibujo, pero aplicada a los vértices, seleccionando manualmente vértice por vértice para definir la parte que deseaba separar como el cofre. El resultado no fue perfecto, ya que no logré distinguir con total precisión las partes que debían separarse, pero el resultado obtenido fue razonable.

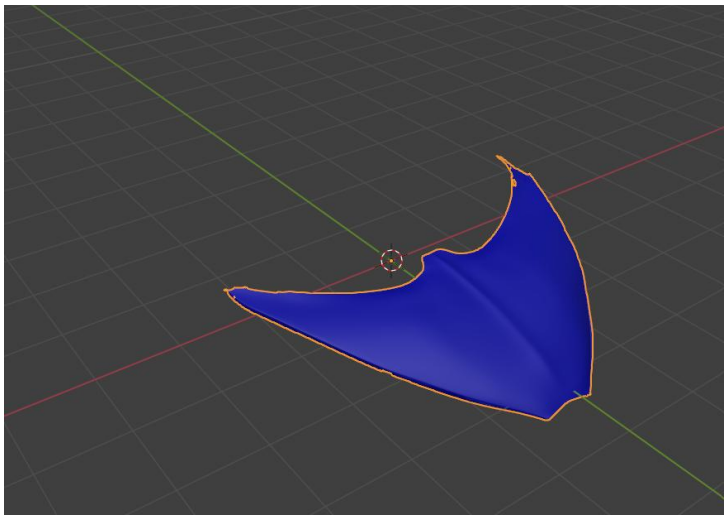




Al importar el modelo de la carrocería del automóvil a OpenGL, ciertos elementos estaban generando problemas. Por esta razón, se decidió, por comodidad, simplificar el modelo eliminándolos.







El código de Goddar para este modelo no resultó ser tan diferente. Se estableció la jerarquía adecuada para unir las cuatro llantas a la carrocería del automóvil y se utilizaron las funciones rotate y translate. Además, se asignaron las teclas F y G para mover el vehículo hacia adelante y hacia atrás, respectivamente, y k y l para mover el cofre del auto.

En este caso, fue necesario agregar las líneas de rotate a cada una de las llantas por separado para lograr el efecto de que todas giraran en la misma dirección al presionar la misma tecla. Esto se debe a que la rotación no es heredable; de lo contrario, las llantas traseras girarían sobre un eje que no les corresponde. Para la traslación, esta se aplicó directamente al nodo padre, el cual hereda el movimiento a todo el modelo, permitiendo que todas las partes se desplacen juntas de manera coherente.

Se utilizó un único modelo de llanta, ya que las cuatro llantas en el diseño son exactamente iguales. Como resultado, al final solo se cargaron tres modelos: uno para la carrocería, otro para el cofre y uno más para las llantas. Además, se asignaron colores diferentes a la carrocería y al cofre para facilitar su distinción visual.

```
Buggatti_M = Model();
Buggatti_M.LoadModel("BUGATTI/Carroceria.obj");
BuggattilLanta_M = Model();
BuggattilLanta_M.LoadModel("BUGATTI/llantaOrigen.obj");
Buggatticapo_M = Model();
Buggatticapo_M.LoadModel("BUGATTI/CAPOORIGEN.obj");
```

CARGA DEL AUTO + TRSLACIONES DEL MODELO COMPLETO CON EL USO DE LAS TECLAS DE LAS ROTACIONES

```
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());
model = glm::translate(model, glm::vec3(0.0f, 0.0f, 0.1f) * mainWindow.getarticulacion2());



modelaux = model;

color = glm::vec3(1.0f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));

Buggatti_M.RenderModel();
```

CARGA DE LAS LLANTAS

```
model = modelaux;

model = glm::translate(model, glm::vec3(2.5f, 0.0f, 2.8f));
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));
color = glm::vec3(0.0f, 1.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
BuggattilLanta_M.RenderModel();

model = modelaux;

model = glm::translate(model, glm::vec3(-2.5f, 0.0f, 2.8f));
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, 0.0f, -5.0f));
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, 0.0f, -5.0f));
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();
```

WINDOW.CPP DE LAS LLANTAS

```
if (key == GLFW_KEY_F)
{
    theWindow->articulacion1 += 10.0;
}

if (key == GLFW_KEY_G)
{
    theWindow->articulacion2 -= 10.0;
}
```

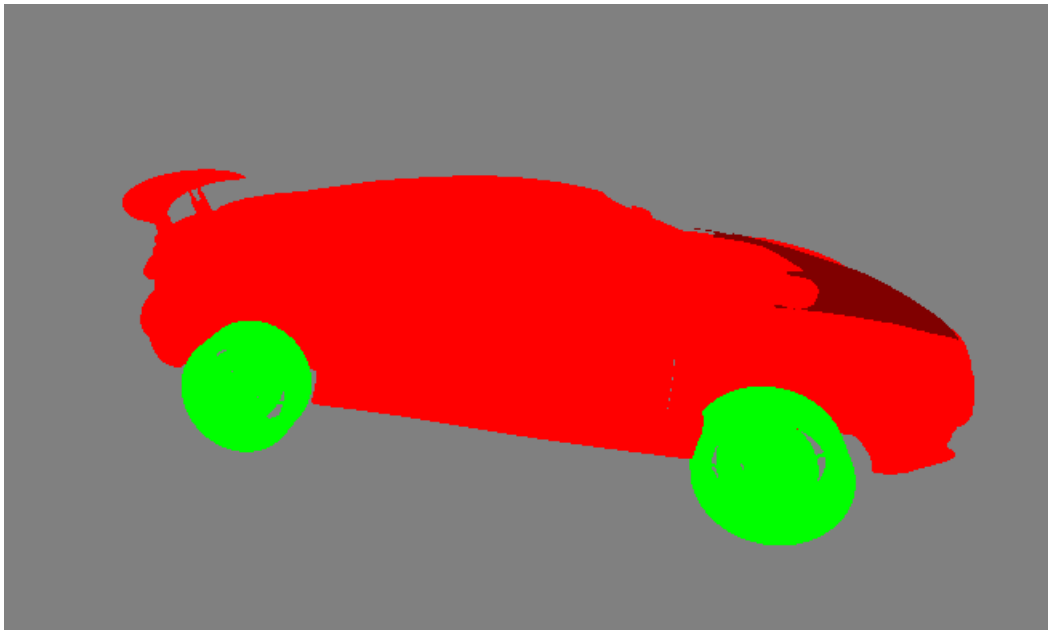
CARGA DEL COFRE

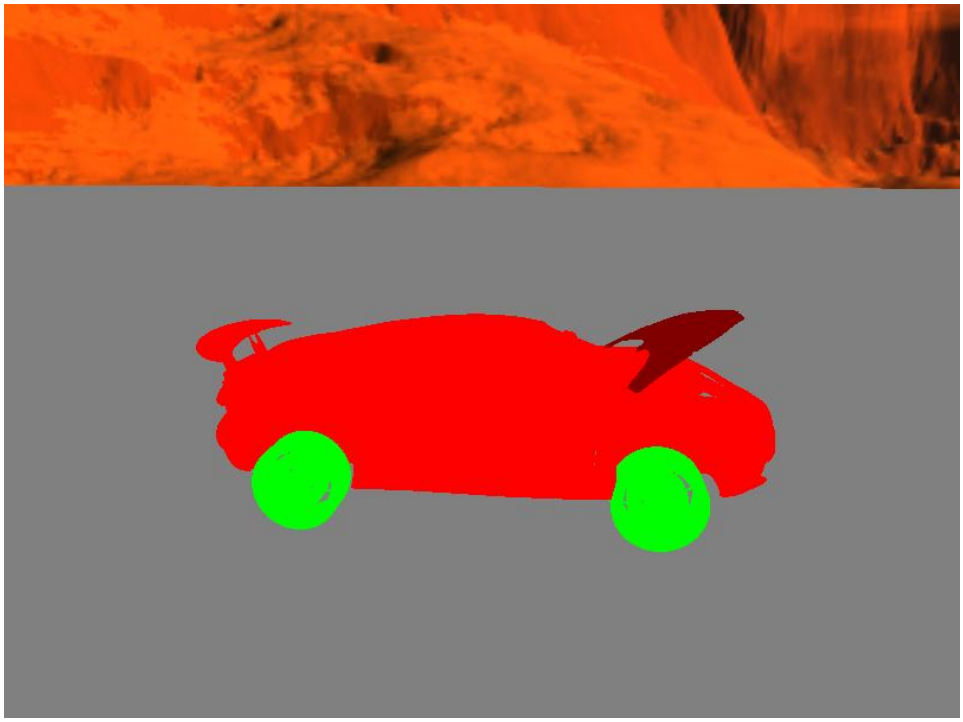
```
model = modelaux;

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));
color = glm::vec3(0.5f, 0.0f, 0.0f);
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Buggatticapo_M.RenderModel();
```

WINDOW.CPP DEL COFRE

```
if (key == GLFW_KEY_K)
{
    if (theWindow->articulacion3 > -10)
    {
    }
    else
    {
        theWindow->articulacion3 += 10.0;
    }
}


if (key == GLFW_KEY_L)
{
    if (theWindow->articulacion3 < -40)
    {
    }
    else
    {
        theWindow->articulacion3 -= 10.0;
    }
}
```

RESULTADO FINAL





CONCLUSIONES

Durante esta práctica, pude comprender en mayor profundidad los desafíos que implica la manipulación y optimización de modelos 3D para su integración en OpenGL . Desde el inicio, el proceso presentó dificultades debido a la complejidad del modelo de automóvil que seleccioné. Si bien la separación de elementos como las llantas fue sencilla, la división del cofre resultó ser un reto considerable, ya que Blender no lo detectaba como una geometría independiente. Esto me obligó a utilizar herramientas de selección de vértices manualmente, lo que requirió tiempo y precisión, aunque el resultado final, aunque no perfecto, fue aceptable.

Otro problema significativo fue la necesidad de eliminar ciertos elementos del modelo original para evitar errores al importarlo a OpenGL. Algunos polígonos de formas desconocidas, líneas y puntos de luz generaban conflictos, lo que me llevó a simplificar la carrocería eliminando estos elementos problemáticos. Este proceso fue clave para garantizar una correcta visualización y funcionamiento del modelo en el entorno de desarrollo.

En cuanto a la programación en visual, la jerarquización de los nodos fue esencial para lograr un movimiento coherente del automóvil. Fue necesario aplicar las transformaciones de rotate a cada llanta por separado, ya que la rotación no es heredable; de lo contrario, las llantas traseras habrían girado sobre un eje incorrecto. En cambio, la traslación sí pudo aplicarse al nodo padre, asegurando que todo el modelo se moviera de manera conjunta.

Para optimizar el uso de recursos, solo se utilizó un único modelo de llanta, ya que las cuatro eran idénticas, lo que permitió reducir la cantidad de archivos cargados. Además, se seleccionaron colores distintos para la carrocería y el cofre, lo que facilitó su identificación dentro del entorno de trabajo.

Si bien logré obtener un resultado funcional, debo admitir que me arrepiento un poco de haber elegido un modelo de automóvil tan complicado. La cantidad de problemas que enfrenté debido a su alta densidad de polígonos y la necesidad de ajustes manuales hicieron que la práctica fuera más difícil de lo esperado. Sin embargo, esta experiencia me permitió aprender valiosas lecciones sobre la manipulación de modelos 3D, la optimización para motores gráficos y la importancia de seleccionar adecuadamente los recursos con los que se trabajará en un proyecto.