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: Sanchez Calvillo Saida Mayela

Nº de Cuenta: 318164481

GRUPO DE LABORATORIO: 02

GRUPO DE TEORÍA: 04

SEMESTRE 2025-2

FECHA DE ENTREGA LÍMITE: 22 de marzo 2025

CALIFICACIÓN: _____

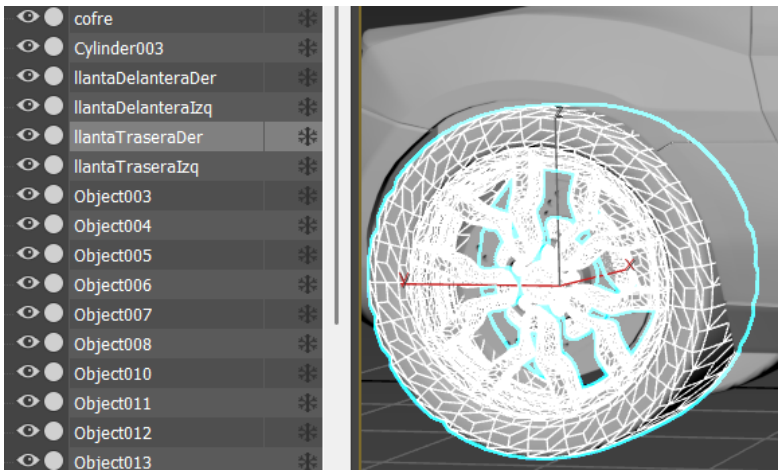
REPORTE DE PRÁCTICA:

1.- Importar su modelo de coche propio dentro del escenario a una escala adecuada.
2.- Importar sus 4 llantas y acomodarlas jerárquicamente, agregar 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

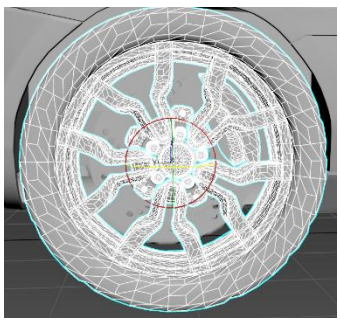
Primero identificamos que partes del carro tenemos que crear como objetos para poder exportar una por una:

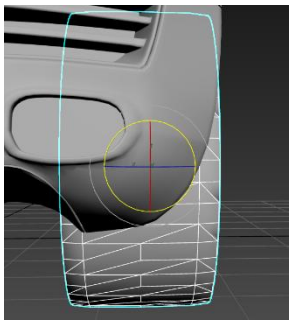
- Cuerpo del carro
- Cofre
- Llanta delantera derecha
- Llanta delantera izquierda
- Llanta trasera derecha
- Llanta trasera izquierda

Después seleccionamos las partes de la llanta que queremos que sean parte de nuestro objeto y las hacemos objetos independientes

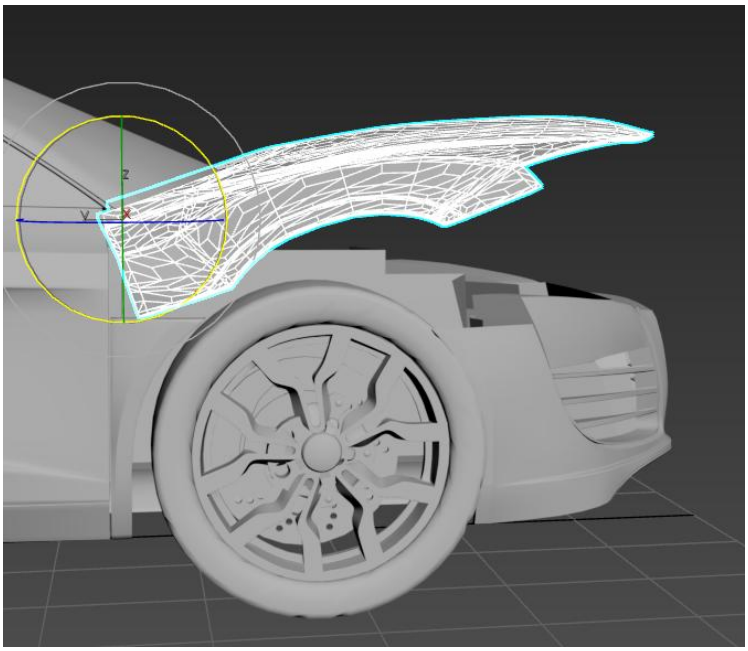


Acomodamos el pivote donde debería girar físicamente nuestra llanta





Así con cada llanta, con el cofre y con el cuerpo completo del auto



Ya que tenemos cada objeto creado, vamos a exportar cada uno de manera independiente a un formato *.obj* y lo colocamos en la carpeta *models* de nuestro proyecto.

Ya en la parte del código, primero vamos a declarar las instancias que necesitamos para cada parte de nuestro carro.

```
Camera camera;
Model Cuerpo_Carro;
Model cofre;
Model llantaDelanteraDer;
Model llantaDelanteraIzq;
Model llantaTraseraDer;
Model llantaTraseraIzq;
```

Después se inicializan los objetos *Model* donde cada uno representa un objeto 3D diferente y cargamos los archivos que acabamos de crear en formato *.obj*

```
Cuerpo_Carro = Model();
Cuerpo_Carro.LoadModel("Models/cuerpo_carro.obj");

cofre = Model();
cofre.LoadModel("Models/cofre.obj");

llantaDelanteraDer = Model();
llantaDelanteraDer.LoadModel("Models/llantaDelanteraDer.obj");

llantaDelanteraIzq = Model();
llantaDelanteraIzq.LoadModel("Models/llantaDelanteraIzq.obj");

llantaTraseraDer = Model();
llantaTraseraDer.LoadModel("Models/llantaTraseraDer.obj");

llantaTraseraIzq = Model();
llantaTraseraIzq.LoadModel("Models/llantaTraseraIzq.obj");
```

Para la función main, y como en el ejercicio de Goddart, primero vamos a llamar al cuerpo para poder acomodar las piezas posteriormente.

Inicializamos la matriz y escalamos nuestro carro ya que es demasiado grande. Esta escala la vamos a heredar para que los objetos también se escalen y embonen correctamente.

Como necesitamos que el carro se mueva (traslade) para adelante y para atrás, vamos a agregar aquí la función de translación y le asignamos dos teclas

- F para mover hacia adelante
- G para mover hacia atrás

```
//Carro
//Translacion con teclas F y G
color = glm::vec3(0.0f, 0.0f, 0.0f); //modelo de carro color negro
model = glm::mat4(1.0);
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
model = glm::translate(model, glm::vec3(0.0f, 0.0f, mainWindow.getarticulacion1()));
model = glm::translate(model, glm::vec3(0.0f, 0.0f, -mainWindow.getarticulacion2()));
//se guarda el modelo del cuerpo con su traslación, para luego en la jerarquía seguir con cofre y llantas
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));
Cuerpo_Carro.RenderModel();//modificar por el modelo sin las 4 llantas y sin cofre
model = modelaux;
```

Seguimos con las demás piezas y teniendo en cuenta que antes de exportarlas las habíamos colocado en el centro del plano, vamos a tener que trasladar cada una de ellas y hacer que embone correctamente.

De igual forma, a cada pieza le asignamos una tecla para mover sobre el mismo eje, pero en sentido opuesto y dar el efecto visual de que el cofre abre o cierra o que las llantas giran para un lado y para el otro.

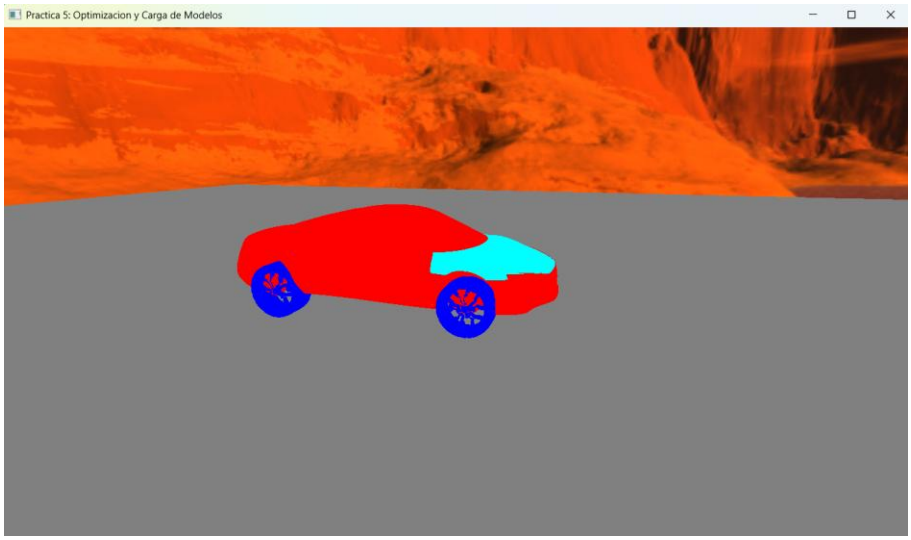
```
//Cofre
//Rotacion con teclas H y J
color = glm::vec3(0.0f, 3.0f, 3.0f);
model = glm::translate(model, glm::vec3(0.0f, 24.0f, 25.0f));
//model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion3()), glm::vec3(-1.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion4()), glm::vec3(1.0f, 0.0f, 0.0f));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
cofre.RenderModel();//modificar por el modelo de mandibula
model = modelaux;
```

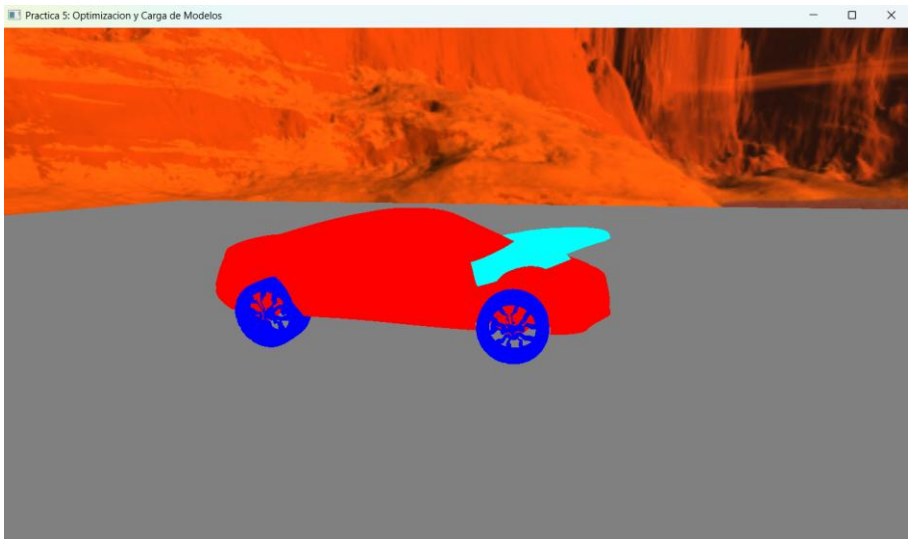
A todas las llantas se le van a agregas las mismas teclas para que puedan girar al mismo tiempo

```
//llanta Delantera Derecha
//Rotacion con teclas K y L
color = glm::vec3(0.0f, 0.0f, 1.0f);
model = glm::translate(model, glm::vec3(-23.0f, 6.2f, 35.8f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion5()), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion6()), glm::vec3(-1.0f, 0.0f, 0.0f));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
llantaDelanteraDer.RenderModel();
model = modelaux;

//llanta Delantera Izquierda
model = glm::translate(model, glm::vec3(23.0f, 6.2f, 35.8f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion5()), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion6()), glm::vec3(-1.0f, 0.0f, 0.0f));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
llantaDelanteraIzq.RenderModel();
model = modelaux;
```

Le asignamos un color distinto a cada pieza para poder observarlas mejor





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

El único problema fue que no había colocado la configuración de propiedades de *assimp-vc140-mt.lib;*

Esto se arregló y todo comenzó a funcionar bien.

3.- Conclusión:

a. Los ejercicios del reporte: Complejidad, Explicación.

La complejidad de tu práctica radica en la manipulación de modelos 3D en OpenGL, la aplicación de transformaciones (traslación, rotación, escala) y la implementación de una jerarquía de modelos.

b. Conclusión

En esta práctica se ha logrado implementar una aplicación gráfica en OpenGL que carga y renderiza modelos 3D, en particular de un coche, aplicando transformaciones como escalado y traslación, y preparando el escenario para una jerarquía de modelos donde el cuerpo del coche es la parte principal y otras partes (como el cofre y las llantas) se conectan a él. Además, se ha incorporado interacción con el usuario para mover el coche en el eje Z utilizando teclas específicas.

Se ha aprendido a cargar modelos, aplicar transformaciones y preparar una jerarquía de modelos, lo cual es fundamental en el desarrollo de aplicaciones gráficas más complejas.