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: Uriarte Ortiz Enrique Yahir

Nº de Cuenta: 318234757

GRUPO DE LABORATORIO: 02

GRUPO DE TEORÍA: 04

SEMESTRE 2025-1

FECHA DE ENTREGA LÍMITE: Sábado 21 de Septiembre del 2024

CALIFICACIÓN: _____

REPORTE DE PRÁCTICA:

1. *Ejecución de los ejercicios.*
   1) Importar su modelo de coche propio dentro del escenario a una escala adecuada.
      Para esta actividad importé el modelo de coche que escogí en el programa 3ds Max, con las herramientas mostradas por el profesor en clase, y apresurándome a las demás actividades de la práctica, dejé únicamente el armazón o cuerpo del coche sin las llantas ni el cofre.
      Una vez con el cuerpo del coche separado y colocados los centros de eje correctamente en el modelo, exporte el cuerpo tipo ***.obj*** .

Fig. 1. *Modelo de carro en 3ds Max.*

Fig. 2. *Cuerpo/Base de carro en 3ds Max.*

Fig. 3. *Cuerpo/Base de carro en 3ds Max.*

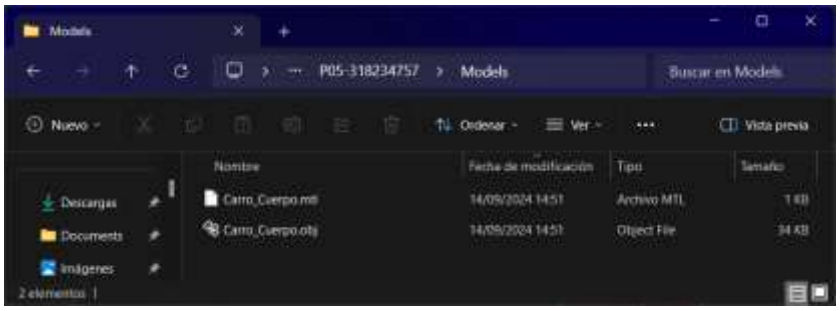

Fig. 4. *Archivos de Cuerpo/Base de carro tipo .obj .*

Ya en el código main del proyecto, agregué las partes correspondientes para emplear jerarquías y declarar los modelos tipo *.obj* que obtuve con 3ds Max.

```
Camera camera;
Model Carro_Cuerpo;
Skybox skybox;
```

Fig. 5. *Declaración de jerarquía para el cuerpo del carro.*

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

Fig. 6. *Declaración de archivo del cuerpo del carro.*

```
glm::mat4 model(1.0);        //Origen de las jererquias.
```

Fig. 7. *Declaración de jerarquía para el cuerpo del carro.*

Después declare el modelo con una escala correcta para poder visualizarse, rotado hacia en Y para tener una vista lateral al momento de ejecutar el código y ajustado en posición para dar la ilusión de que toca el piso. A si mismo decidí darle un color morado.

```
//ACTIVIDAD 1.- Importar su modelo de coche propio dentro del escenario a una escala adecuada.

model = glm::mat4(1.0);

//Carro Base.
color = glm::vec3(0.5f, 0.0f, 1.0f);
model = glm::translate(model, glm::vec3(0,-0.25f, 0.0f));
model = glm::rotate(model, glm::radians(90.0f), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.05f, 0.05f, 0.05f));
Cofre = model;
Llanta01 = model;
Llanta02 = model;
Llanta03 = model;
Llanta04 = model;
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Carro_Cuerpo.RenderModel();
```

Fig. 8. *Declaración del cuerpo del carro.*

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.

Para esta segunda actividad convino lo hecho en la actividad anterior de separar las 4 llantas, pero aun falta ajustarlas en posición al origen para que al momento de asignarles la rotación con las teclas estas giren adecuadamente.

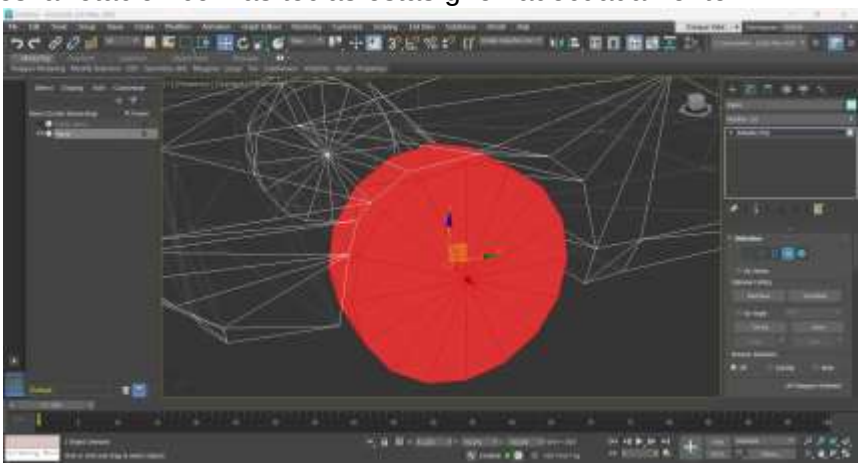

Fig. 9. *Selección de llanta 1 del carro en 3ds Max.*

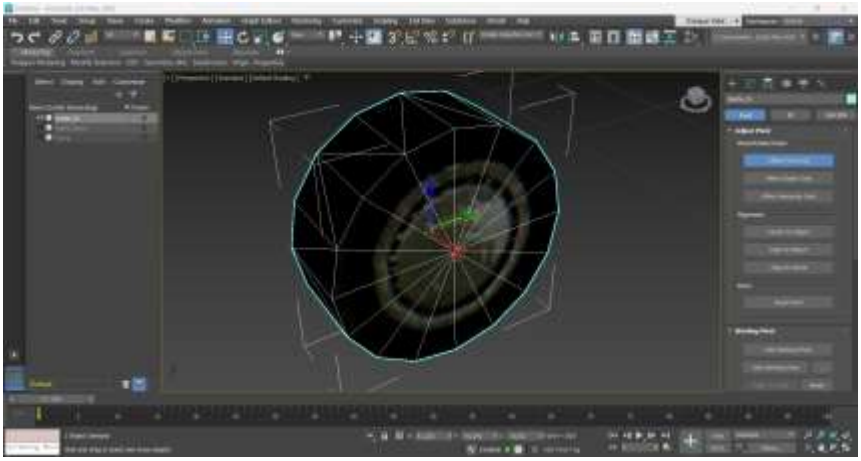

Fig. 10. *Llanta 1 separada del carro en 3ds Max.*

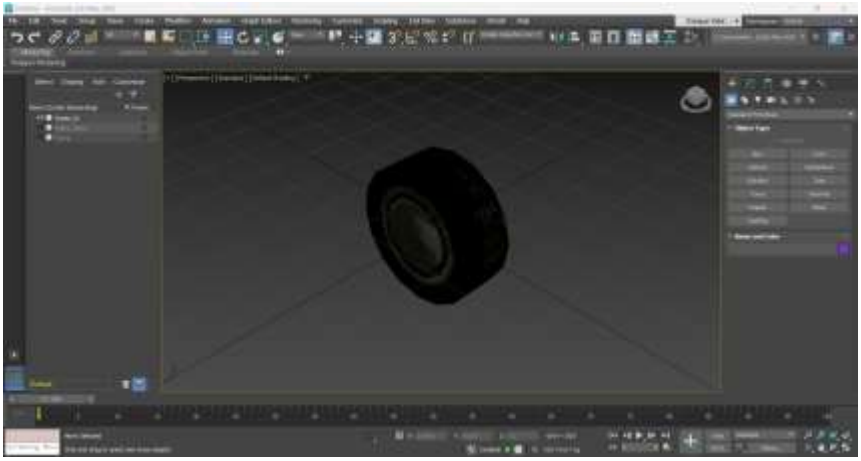

Fig. 11. *Llanta 1 ajustada al centro en 3ds Max.*

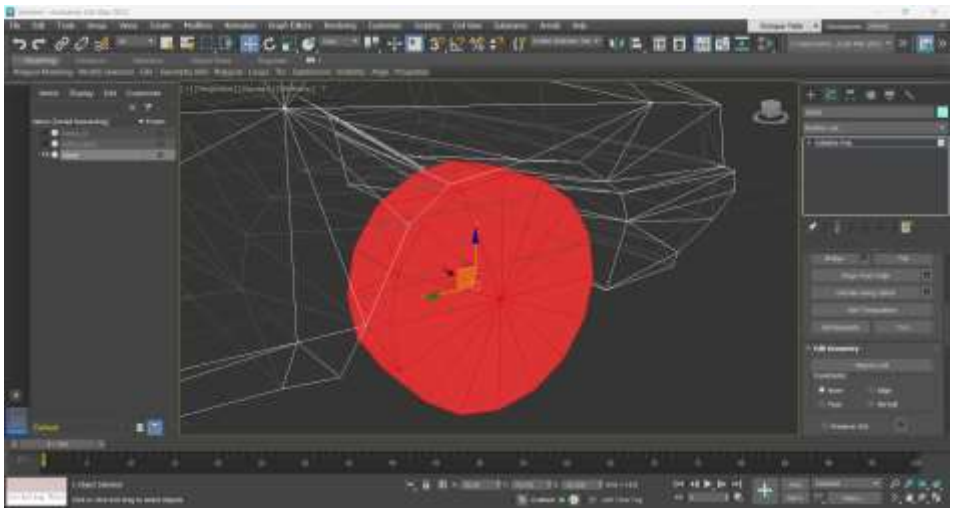

Fig. 12. *Selección de llanta 2 del carro en 3ds Max.*

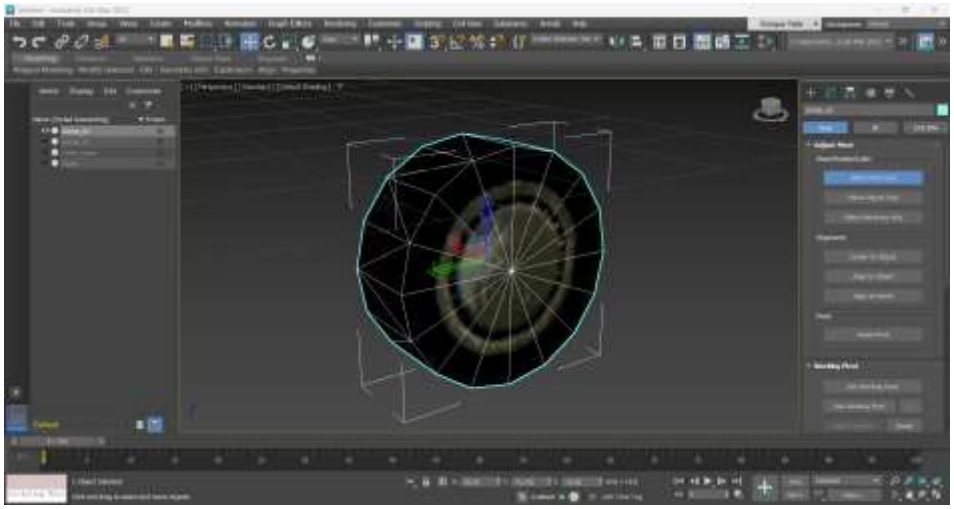

Fig. 13. *Llanta 2 separada del carro en 3ds Max.*

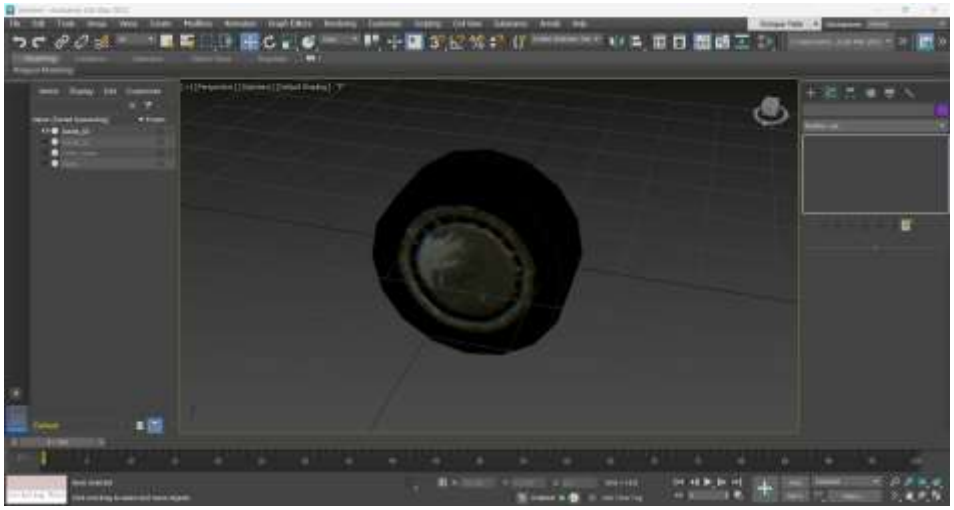

Fig. 14. *Llanta 1 ajustada al centro en 3ds Max.*

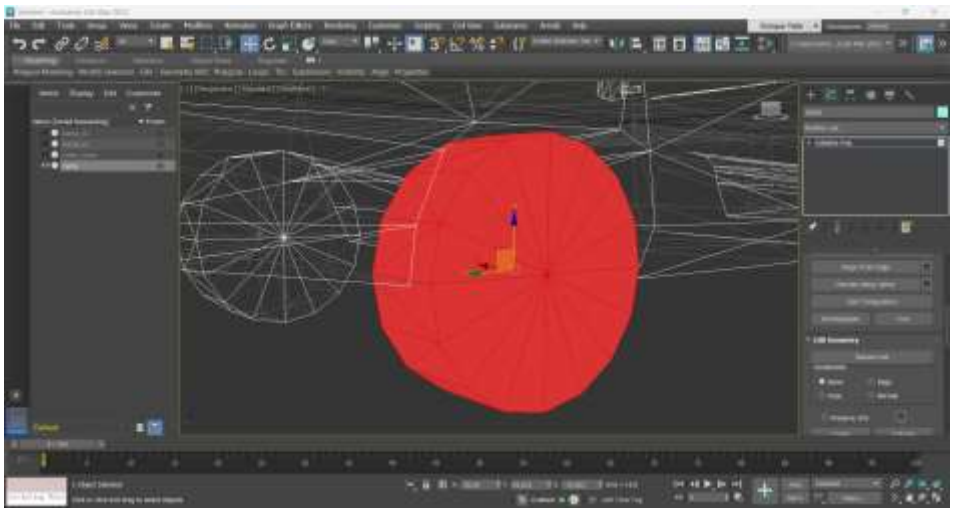

Fig. 15. *Selección de llanta 3 del carro en 3ds Max.*

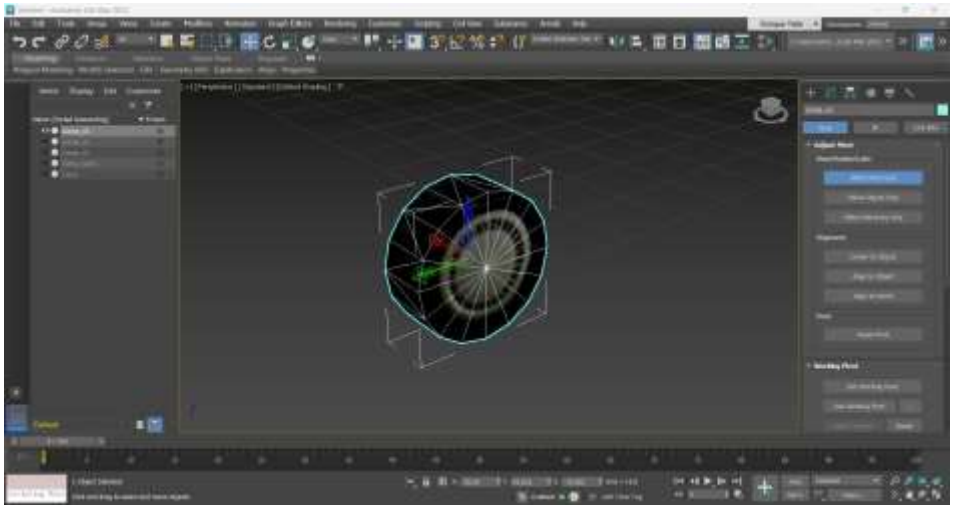

Fig. 16. *Llanta 3 separada del carro en 3ds Max.*

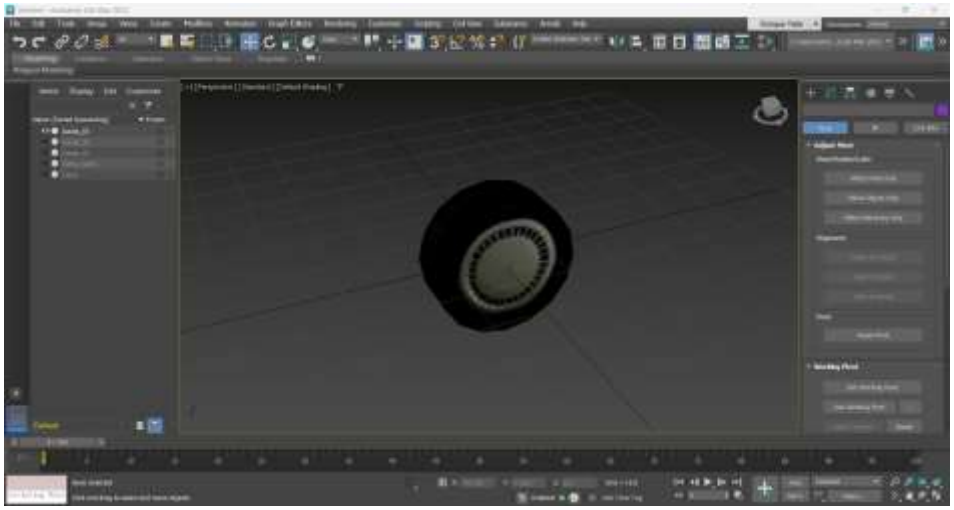

Fig. 17. *Llanta 3 ajustada al centro en 3ds Max.*

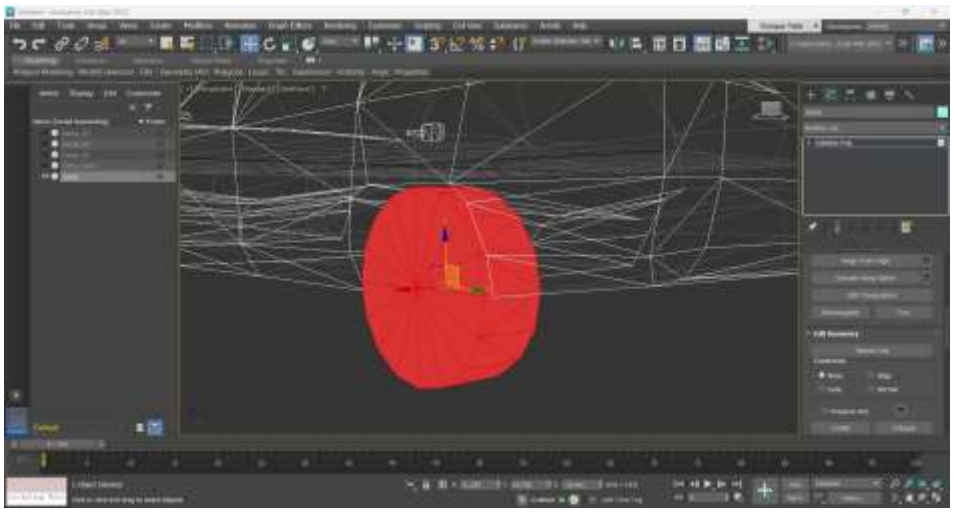

Fig. 18. *Selección de llanta 4 del carro en 3ds Max.*

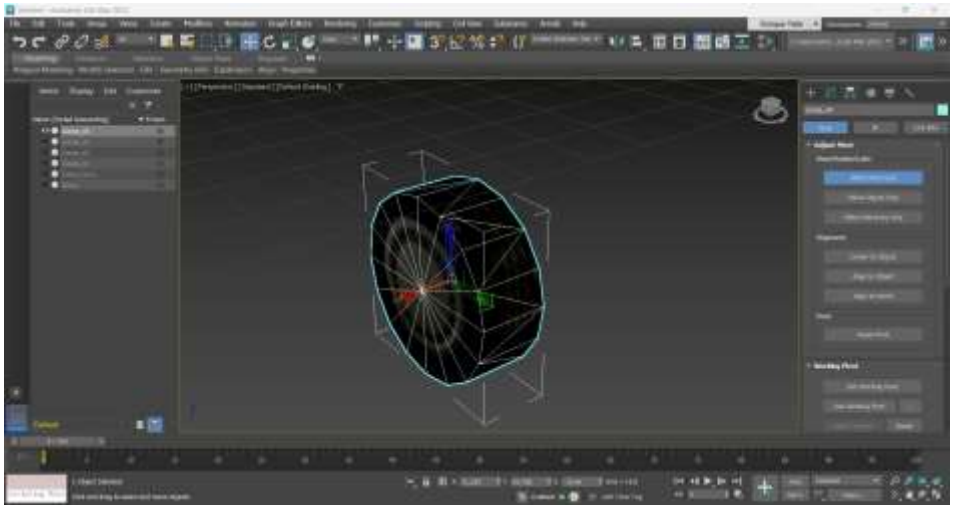

Fig. 19. *Llanta 4 separada del carro en 3ds Max.*

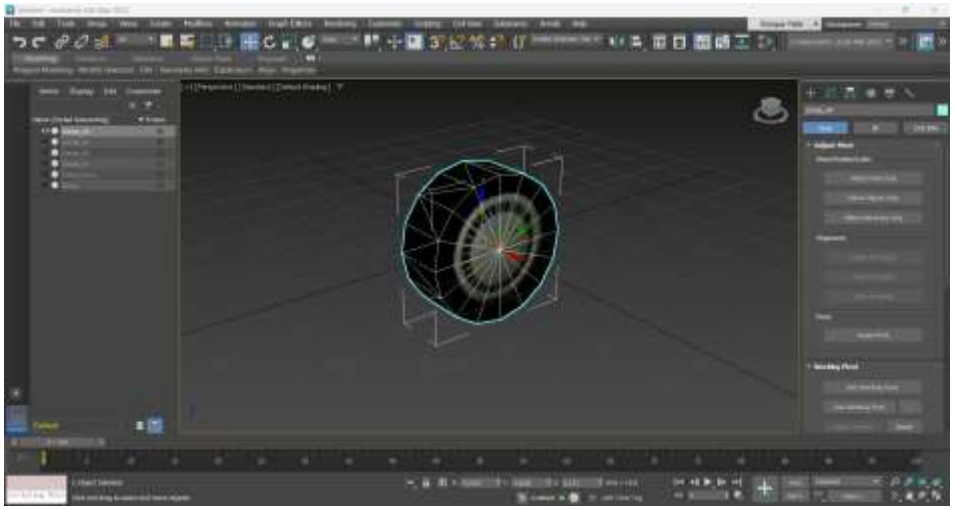

Fig. 20. *Llanta 4 ajustada al centro en 3ds Max.*

Con cada llanta separada, ajustada y colocando sus centros de eje correctamente en el modelo, exporte cada como tipo ***.obj*** .

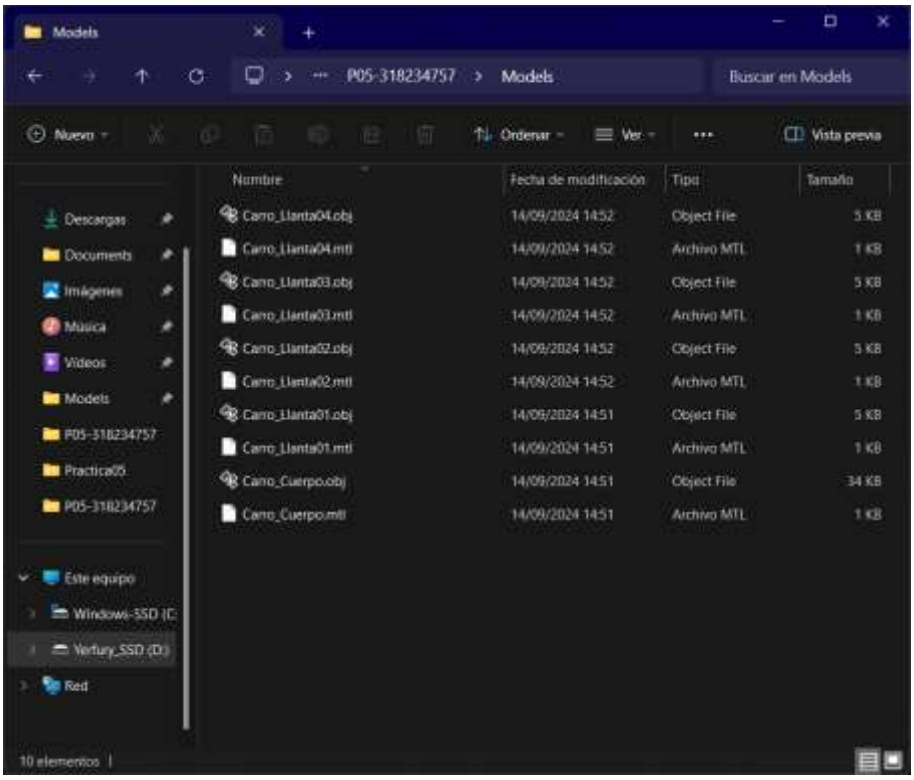

Fig. 21. *Archivos de cada llanta del carro tipo .obj .*

En el código main del proyecto, nuevamente agregué las partes correspondientes para emplear jerarquías y declarar los modelos tipo *.obj* que obtuve con 3ds Max.

```
Model Carro_Llanta01;
Model Carro_Llanta02;
Model Carro_Llanta03;
Model Carro_Llanta04;
```

Fig. 22. *Declaración de jerarquía para cada llanta del carro.*

```
Carro_Llanta01 = Model();
Carro_Llanta01.LoadModel("Models/Carro_Llanta01.obj");
Carro_Llanta02 = Model();
Carro_Llanta02.LoadModel("Models/Carro_Llanta02.obj");
Carro_Llanta03 = Model();
Carro_Llanta03.LoadModel("Models/Carro_Llanta03.obj");
Carro_Llanta04 = Model();
Carro_Llanta04.LoadModel("Models/Carro_Llanta04.obj");
```

Fig. 23. *Declaración de archivos de cada llanta del carro.*

```
glm::mat4 Llanta01(1.0);    //Jerarquia con llanta 1.
glm::mat4 Llanta02(1.0);    //Jerarquia con llanta 2.
glm::mat4 Llanta03(1.0);    //Jerarquia con llanta 3.
glm::mat4 Llanta04(1.0);    //Jerarquia con llanta 4.
```

Fig. 24. *Declaración de jerarquía para el cuerpo del carro.*

Dentro del código las 4 llantas tienen la misma estructura, primero manda a llamar a la jerarquía correspondiente de acuerdo a la posición original del cuerpo del carro,

después se asigna la translación de posición y la rotación, que en este caso como un carro las 4 llantas rotan al mismo tiempo sobre X con la misma tecla, después se guardan las translaciones y posiciones en la jerarquía de cada llanta. Para la posición de las llantas, la llanta 1 y 2 están en la misma posición de Y y Z pero cambiando la posición con signo en X, lo mismo ocurre para las llantas 3 y 4. Además de esto, todas las llantas son del mismo color gris.

```
//ACTIVIDAD 2.- Importar sus 4 llantas.

//Articulación - Llanta01. Derecha.
model = Llanta01;
color = glm::vec3(0.5f, 0.5f, 0.5f);
model = glm::translate(model, glm::vec3(30.0f, -20.0f, 70.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()), glm::vec3(1.0f, 0.0f, 0.0f));
Llanta01 = model;
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Carro_Llanta01.RenderModel();

//Articulación - Llanta02. Izquierda.
model = Llanta02;
color = glm::vec3(0.5f, 0.5f, 0.5f);
model = glm::translate(model, glm::vec3(-30.0f, -20.0f, 70.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()), glm::vec3(1.0f, 0.0f, 0.0f));
Llanta02 = model;
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Carro_Llanta02.RenderModel();

//Articulación - Llanta03. Izquierda.
model = Llanta03;
color = glm::vec3(0.5f, 0.5f, 0.5f);
model = glm::translate(model, glm::vec3(-30.0f, -21.0f, -55.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()), glm::vec3(1.0f, 0.0f, 0.0f));
Llanta03 = model;
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Carro_Llanta03.RenderModel();

//Articulación - Llanta04. Derecha.
model = Llanta04;
color = glm::vec3(0.5f, 0.5f, 0.5f);
model = glm::translate(model, glm::vec3(30.0f, -21.0f, -55.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()), glm::vec3(1.0f, 0.0f, 0.0f));
Llanta04 = model;
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Carro_Llanta04.RenderModel();
```

Fig. 25. *Declaración de cada llanta del carro.*

Por último, en los archivos Window declaro la articulación para la rotación de las llantas, con K avanzaran hacia adelante y con L hacia atrás.

```
GLfloat getarticulacion2() { return articulacion2; }
```

Fig. 26. *Declaración de articulación para las llantas del carro.*

```
articulacion2 = 0.0f;
```

Fig. 27. *Declaración de articulación para las llantas del carro.*

```
//Rotar la llanta 01, 02, 03 y 04.
if (key == GLFW_KEY_K) {theWindow->articulacion2 += 10.0;}
if (key == GLFW_KEY_L) {theWindow->articulacion2 -= 10.0;}
```

Fig. 28. *Declaración de articulación para las llantas del carro.*

3) Importar el cofre del coche, acomodarlo jerárquicamente y agregar la rotación para poder abrir y cerrar.

Para esta tercera actividad nuevamente convino la separación realizada en la primer actividad ya que el cofre también se encuentra separado, pero aún falta ajustarlas en posición al origen para que al momento de asignarle la rotación esta gire adecuadamente.

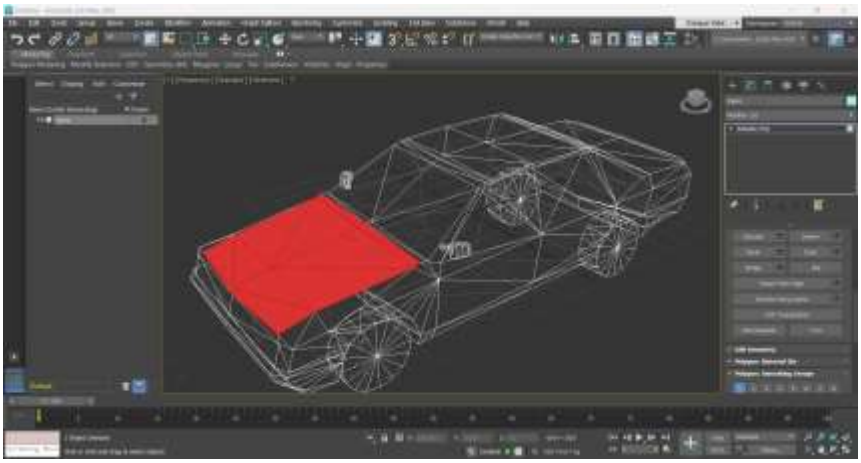

Fig. 29. *Selección de cofre del carro en 3ds Max.*

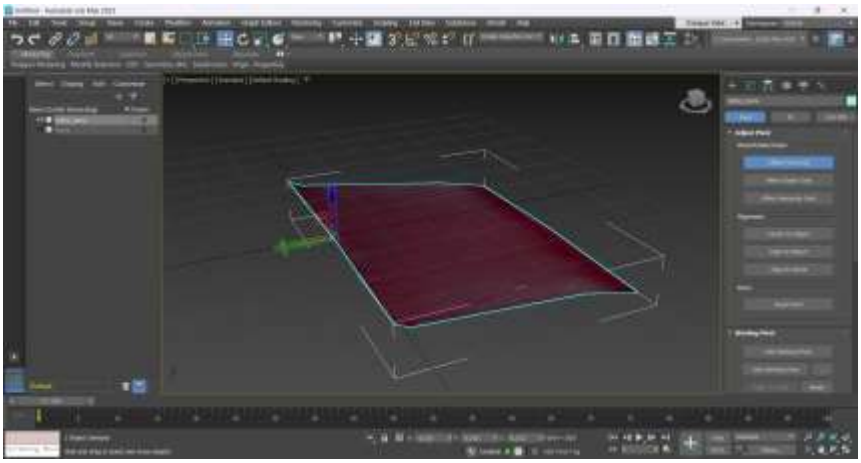

Fig. 30. *Cofre del carro en 3ds Max.*

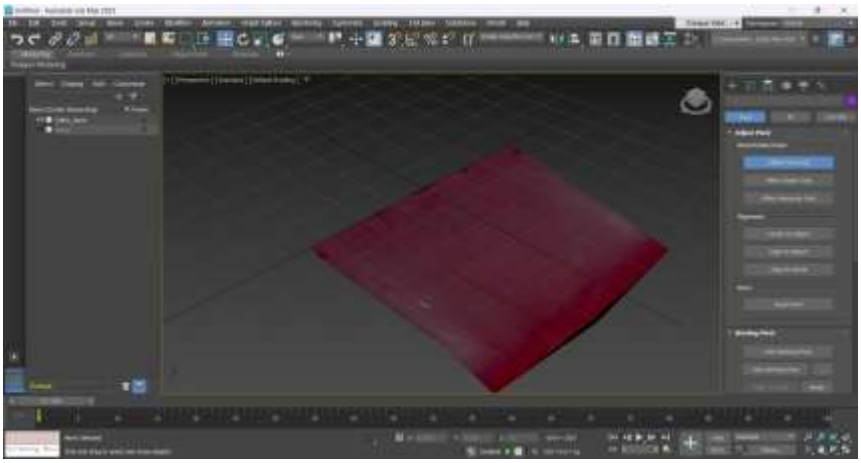

Fig. 31. *Cofre del carro ajustada al centro en 3ds Max.*

Con el cofre colocando correctamente para una rotación adecuada, exporte cada como tipo ***.obj*** .

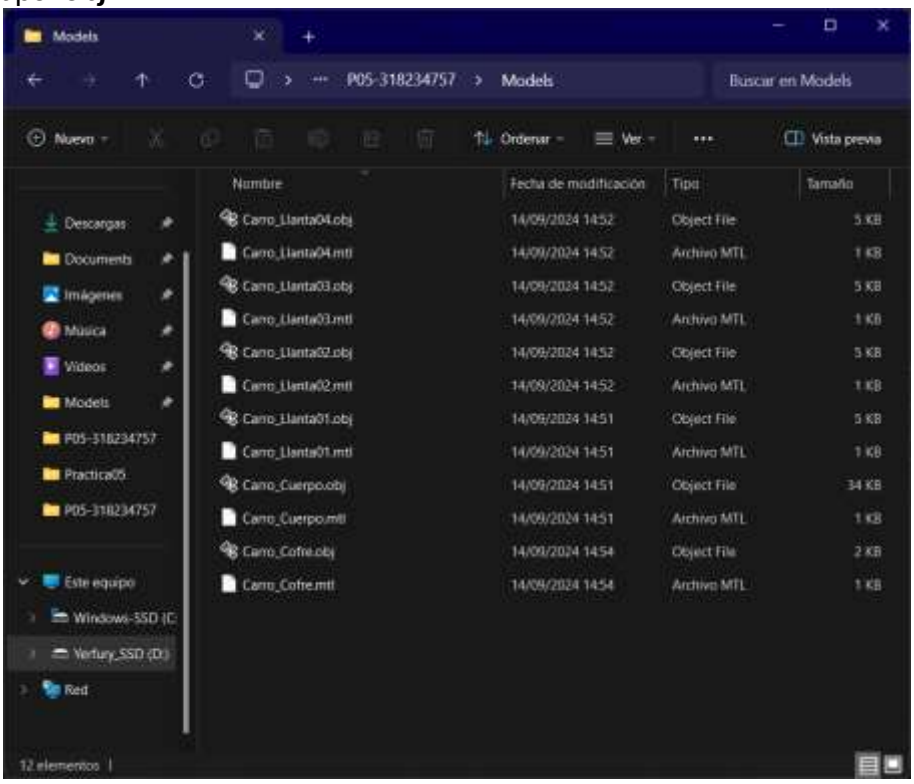

Fig. 32. *Archivo del cofre del carro tipo .obj .*

En el código main del proyecto, nuevamente agregué las partes correspondientes para emplear jerarquía y declarar el modelo tipo *.obj* que obtuve con 3ds Max.

```
Model Carro_Cofre;
```

Fig. 33. *Declaración de jerarquía para cada llanta del carro.*

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

Fig. 34. *Declaración de archivos de cada llanta del carro.*

```
glm::mat4 Cofre(1.0);        //Jerarquia con Cofre.
```

Fig. 35. *Declaración de jerarquía para el cuerpo del carro.*

Para diferenciar el cofre del coche lo asigne de un tono de azul fuerte, su estructura de jerarquía es la misma que la de cada llanta, claro que ajustando la translación y asignándole la rotación con otra tecla sobre el eje X en dirección negativa.

```
//ACTIVIDAD 3.- Importar el cofre del coche.

//Articulación - Cofre de carro.
model = Cofre;
color = glm::vec3(0.3f, 0.0f, 1.0f);
model = glm::translate(model, glm::vec3(0.0f, 11.0f, 50.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()), glm::vec3(-1.0f, 0.0f, 0.0f));
Cofre = model;
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Carro_Cofre.RenderModel();
```

Fig. 36. *Declaración de cada llanta del carro.*

Por último, en los archivos Window declaro la articulación para la rotación del cofre con un límite de 45 grados, con la letra Z para levantarla y X para devolverla a la posición de cerrado del coche.

```
GLfloat getarticulacion1() { return articulacion1; }
```

Fig. 37. *Declaración de articulación para el cofre del carro.*

```
articulacion1 = 0.0f;
```

Fig. 38. *Declaración de articulación para el cofre del carro.*

```
//Abrir el cofre.
if (key == GLFW_KEY_Z) {
    if (theWindow->articulacion1 > 46.0) {}
    else {theWindow->articulacion1 += 2.0;}}
if (key == GLFW_KEY_X){
    if (theWindow->articulacion1 < 1.0){}
    else {theWindow->articulacion1 -= 2.0;}}
```

Fig. 39. *Declaración de articulación para el cofre del carro.*

4) Agregar traslación con teclado para que pueda avanzar y retroceder de forma independiente.

Para esta ultima actividad tuve que modificar el *translate* del código para que de acuerdo con la tecla se moviera el carro sobre eje X, para esto le asigne en el valor de la posición de X una articulación.

```
//Carro Base.
color = glm::vec3(0.5f, 0.0f, 1.0f);
model = glm::translate(model, glm::vec3(mainWindow.getarticulacion3(),-0.25f, 0.0f));
model = glm::rotate(model, glm::radians(90.0f), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.05f, 0.05f, 0.05f));
Cofre = model;
```

Fig. 40. *Modificaciones en Translate para mover el coche.*

Por último, en los archivos Window declaro la articulación para el movimiento del coche, siendo con la letra G que se mueva hacia adelante y con la letra H se mueva en reversa con respecto al eje X.

```
GLfloat getarticulacion3() { return articulacion3; }
```

Fig. 41. *Declaración de articulación movimiento del carro.*

```
GLint width, height;
GLfloat rotax,rotay,rotaz, articulacion1, articulacion2, articulacion3;
```

Fig. 42. *Declaración de articulación movimiento del carro.*

```
articulacion3 = 0.0f;
```

Fig. 43. *Declaración de articulación movimiento del carro*.

```
//Moviento del coche
if (key == GLFW_KEY_G) {theWindow->articulacion3 += 0.3;}
if (key == GLFW_KEY_H) {theWindow->articulacion3 -= 0.3;}
```

Fig. 44. *Declaración de articulación movimiento del carro.*

Al momento de ejecutar el código con todas las actividades, lo mostrado en pantalla es lo siguiente:

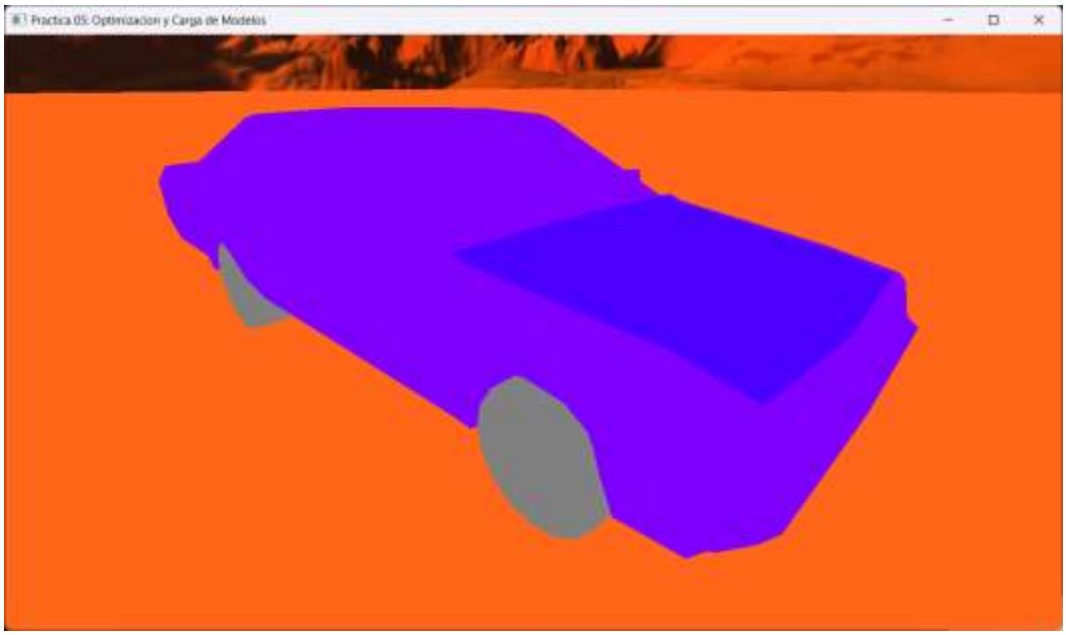

Fig. 45. *Ejecución del código.*

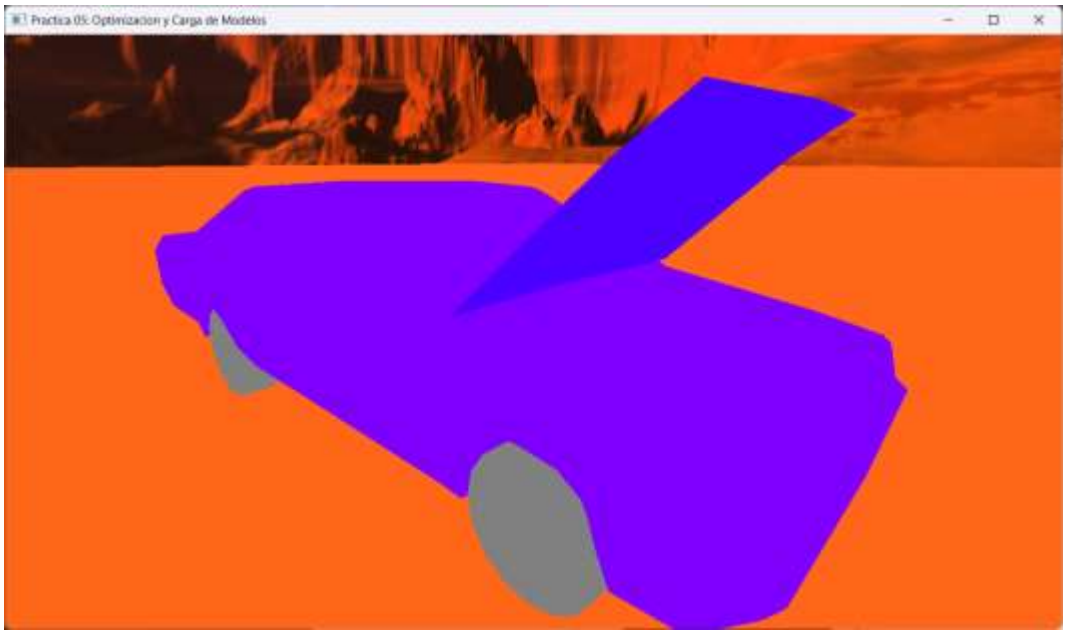

Fig. 46. *Ejecución del código.*

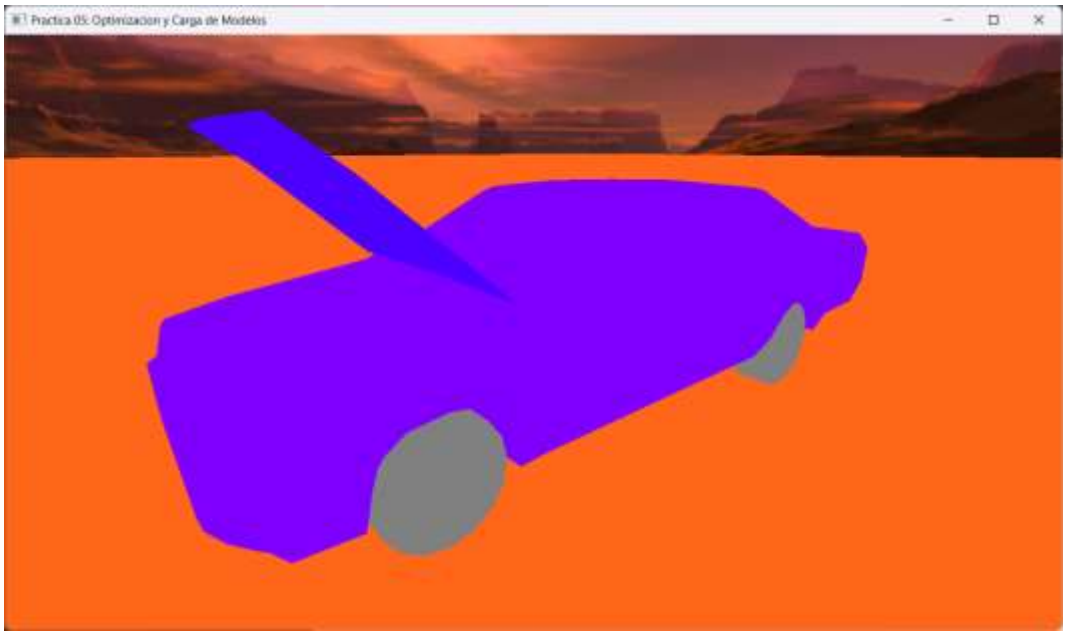

Fig. 47. *Ejecución del código.*

Como evidencia para la práctica, realice un video donde se muestra la ejecución, el archivo se llama ***P05-318234757_Ejercicios.mp4***, adjunto el enlace de la carpeta compartida:
https://drive.google.com/file/d/1T7nPGk67-J3z1JOq3GM-_nomS_LCMKDh/view?usp=sharing

2. *Problemas presentados.*
   En las primeras 3 actividades no presente problemas, la separación de las partes del coche en 3ds Max, así como la asignación de jerarquías, escalas, translaciones y movimientos con una tecla o más, y la asignación de limitar el ángulo de rotación de estas es algo que ya he realizado en prácticas anteriores, por lo que realizarlas en esta no tuvo complicación.
   Para la 4 actividad fue donde tuve mas complicaciones, ya que no entendía como debía definir la estructura del translate para el movimiento del coche, en un inicio había considerado solo cambiar la palabra rotate por translate en la línea de código de rotate para la articulación, pero me mandaba error. Intente buscar una solución a mi error en internet, pero no encontré nada relacionado, por lo que al final opte mejor por preguntarle al profesor en clase de teoría, dando me la solución donde en realidad la estructura de translate para asignar el movimiento era solo colocar la parte de articulación en el eje de translate hacia el cual quería que se moviera, con esto al final la ejecución de la actividad fue correcta de acuerdo con lo solicitado. Al final todos los ejercicios se pudieron realizar exitosamente.

3. *Conclusión:*
   Tras realizar los ejercicios de la practica me parece que la complejidad era normal, las actividades de separar por partes un modelo en 3ds Max, asignar jerarquías, escalas, translaciones, movimientos con una tecla o más y la limitar ángulos de rotación es algo que ya había realizado en ejercicios y practicas anteriores, pero estas actividades sirvieron como practica para no olvidar como se implementan. Donde presente mas problemas fue en la actividad numero 4, ya que anteriormente solo habíamos asignado

teclas para movimientos de rotación y de translación en los ejes, pero con la asistencia del profesor la última actividad creo fue la más fácil de todas. Aun así, me pareció interesante la implementación en esta práctica para emplear modelos descargados de internet y usar movimientos con teclas no solo para rotarlos. Al final consideró que esta práctica cumplió su objetivo de introducirnos al uso de modelos en el código, siendo esto a mi parecer una buena práctica para el desarrollo del proyecto final.

4. *Bibliografía.*

    - *Audi Car. The Models Resource. https://www.models-resource.com/pc_computer/cryoffear/model/57122/. Accedido septiembre 21, 2024.*