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: Velazquez Caudillo Osbaldo

Nº de Cuenta: 320341704

GRUPO DE LABORATORIO: 02

GRUPO DE TEORÍA: 06

SEMESTRE 2026-1

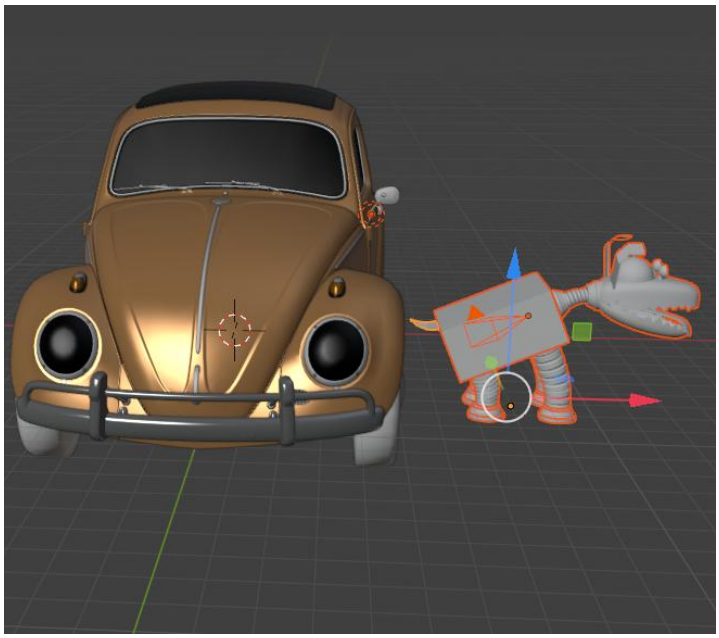
FECHA DE ENTREGA LÍMITE: 28/09/25

CALIFICACIÓN: _____

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.

Ejercicios

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



Aquí podemos la comparación de tamaño del auto con el Goddard que se realizó en el ejercicio, con base en esta escala del carro trabajamos. De esta manera nos aseguramos de tener una escala adecuada y que el modelo no quede muy pequeño.

```
Model Carro_C;
Model Carro_cofre;
Model Carro_llantaD;
Model Carro_llantaI;
```

Creamos instancias de Model para cada parte del carro, el cofre, las llantas derechas y las llantas izquierdas, además del carro sin todas esas partes.

```
Carro_C = Model();
Carro_C.LoadModel("Models/carro_modelo_cuerpo.fbx");

Carro_cofre = Model();
Carro_cofre.LoadModel("Models/carro_cofre.fbx");

Carro_llantaD= Model();
Carro_llantaD.LoadModel("Models/llanta_derecha.fbx");

Carro_llantaI = Model();
Carro_llantaI.LoadModel("Models/llanta_izquierda.fbx");
```

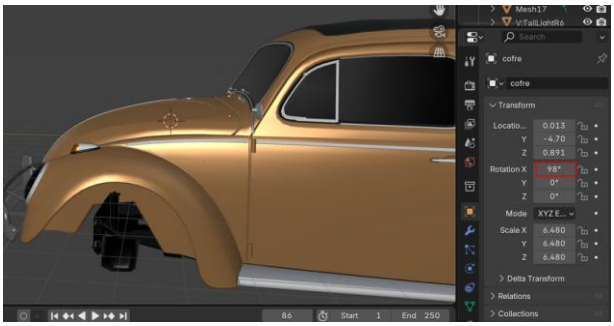
Dentro del main lo que hacemos es importar nuestros modelos .fbx de cada parte para poder hacer uso de ellas.

```
// INICIA DIBUJO DEL PISO
color = glm::vec3(0.5f, 0.5f, 0.5f); //piso de color gris
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();

//CARRO
//Cuerpo sin llantas ni cofre
color = glm::vec3(0.278, 0.631, 0.702);
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, -1.2f, -1.5f));
model = glm::translate(model, glm::vec3(0.0f, 0.0f, mainWindow.getarticulacion2()*0.1));
model = glm::translate(model, glm::vec3(0.0f, 0.0f, mainWindow.getarticulacion3()*0.1));
modelaux = model;
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Carro_C.RenderModel();

//cofre
model = modelaux;
color = glm::vec3(0.78f, 0.80f, 0.85f);
model = glm::translate(model, glm::vec3(0.0f, 0.1f, 0.76f));
model = glm::rotate(model, glm::radians(8.0f), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()), glm::vec3(1.0f, 0.0f, 0.0f));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Carro_cofre.RenderModel();
```

```
//llantas delanteras
//llanta derecha
model = modelaux;
color = glm::vec3(0.0f, 0.0f, 0.0f);
model = glm::translate(model, glm::vec3(0.65f, -0.47f, 1.3f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion3()), glm::vec3(1.0f, 0.0f, 0.0f));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Carro_llantaD.RenderModel();
//llanta izquierda
model = modelaux;
color = glm::vec3(0.0f, 0.0f, 0.0f);
model = glm::translate(model, glm::vec3(-0.7f, -0.47f, 1.3f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion3()), glm::vec3(1.0f, 0.0f, 0.0f));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Carro_llantaI.RenderModel();

//Llantas traseras
//llanta derecha
model = modelaux;
color = glm::vec3(0.0f, 0.0f, 0.0f);
model = glm::translate(model, glm::vec3(0.65f, -0.47f, -1.1f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion3()), glm::vec3(1.0f, 0.0f, 0.0f));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Carro_llantaD.RenderModel();

//llanta izquierda

model = modelaux;
color = glm::vec3(0.0f, 0.0f, 0.0f);
model = glm::translate(model, glm::vec3(-0.7f, -0.47f, -1.1f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()), glm::vec3(1.0f, 0.0f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion3()), glm::vec3(1.0f, 0.0f, 0.0f));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Carro_llantaI.RenderModel();
```

Se agregan las traslaciones necesarias para que nuestro modelo quede completo y las partes estén en su lugar, para el caso del cofre se tuvo que agregar una pequeña rotación, y esto por que al cambiar de lugar el pivote hizo que ya no quedara en su lugar como debería, en este caso Blender fue de mucha ayuda, ya que nos dice el ángulo que tiene la parte respecto al pivote, entonces en este lo único que se hizo fue agregarle una rotación sobre el eje x para que bajar un poco y quedara en la posición correcta.

Así es como se ve el cofre con el ángulo por defecto y el pivote en el centro para que tuviera la rotación correcta, como se puede ver pareciera que estuviera abierto, agregando 8 grados mas podemos tener el cofre cerrado por completo con el pivote tal cual como lo tenemos.



Para las llantas lo que se hizo únicamente fue utilizar dos modelos, el derecho e izquierdo y reutilizarlos, aquí únicamente hicimos uso de traslaciones para que se acomoda correctamente al modelo del auto, esto fue de mucha ayuda porque las coordenadas solamente varían en el x o y dependiendo de la posición.

Ahora para agregarle rotación a las llantas hacemos uso de las articulaciones en este caso se puede ver que solamente usamos articulación 2 y 3 para cada una de las llantas, la articulación 2 hace que las llantas roten hacia adelante, mientras que la 3 rota hacia atrás.

Para la parte de la traslación, como se hace uso de jerarquía pues quiere decir que tanto el cofre como las llantas se van a trasladar si el cuerpo del carro se traslada porque es el nodo padre.

```
//CARRO
//Cuerpo sin llantas ni cofre
color = glm::vec3(0.278, 0.631, 0.702);
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, -1.2f, -1.5f));
model = glm::translate(model, glm::vec3(0.0f, 0.0f, mainWindow.getarticulacion2()*0.1));
model = glm::translate(model, glm::vec3(0.0f, 0.0f, mainWindow.getarticulacion3()*01));
modelaux = model;
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Carro_C.RenderModel();
```

Aquí es en donde agregamos las traslaciones para que el auto se mueva adelante y hacia atrás, hacemos uso de las articulaciones para que, al presionar las teclas, entonces tome ese valor el eje z.

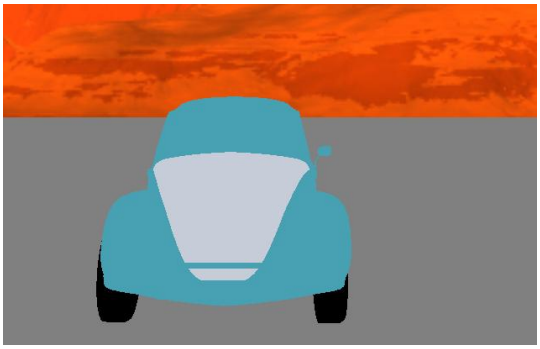
```
if (key == GLFW_KEY_F)
{
    static int dir = -1;
    theWindow->articulacion1 += 2.0f * dir;
    if (dir < 0 && theWindow->articulacion1 <= -55.0f) {
        theWindow->articulacion1 = -55.0f;
        dir = +1;
    }
    else if (dir > 0 && theWindow->articulacion1 >= 0.0f) {
        theWindow->articulacion1 = 0.0f;
        dir = -1; // al llegar, invierte: ahora a abrir
    }
}
//adelante
if (key == GLFW_KEY_E)
{
    theWindow->articulacion2 += 5.0; //rotar sobre el eje y 10 grados
}
//atras
if (key == GLFW_KEY_Q)
{
    theWindow->articulacion3 += -5.0;
}
```

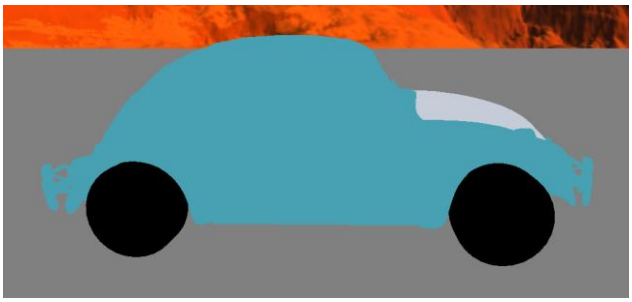
En el Window.cpp tenemos las teclas que nos ayudan en los movimientos

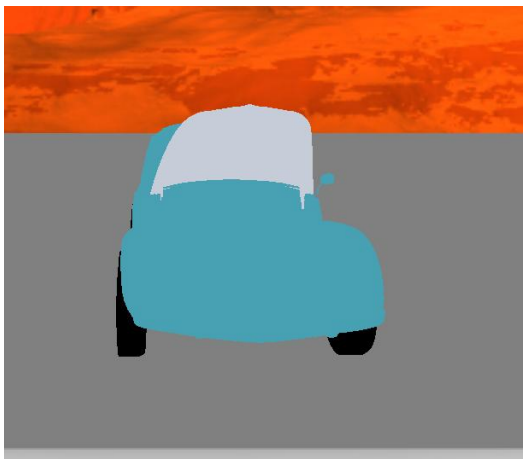
La tecla F corresponde a la articulación 1 que se encarga del movimiento del cofre entonces lo que hacemos es que cuando abra tope hasta -55° grados una vez que llega ahí va a comenzar a bajar hasta llegar a 0° grados que hace referencia al inicio del objeto como lo tenemos definido (98° grados).

Las teclas E y Q se encarga del movimiento de las llantas (rotación) así como de la traslación del objeto, la tecla E rota 5 grados, mientras que la tecla Q lo hace -5 grados esto nos da la rotación de las llantas al frente y atrás.

Ejecución







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

Uno de los problemas principales fue que los modelos no se veían todos los detalles, entonces eso provocaba que para acomodar las piezas fuera mas complicado, sobre todo en el cofre ese fue el más complicado porque no encontraba la forma de acomodarlo correctamente, me fue muy difícil hasta que me di cuenta que había que aplicarle una pequeña rotación por que se modifica el pivote del objeto tal y como lo explico en el desarrollo Blender fue de mucha ayuda para el ángulo.

Conclusión:

Esta práctica no fue muy complicada, prácticamente todo fue trasladar los objetos a su posición correcta, y agregarle movimiento por medio de las articulaciones, fue tardada por tener que estar seleccionando ciertas partes del carro y a veces no se seleccionaban todos los triángulos o se seleccionaban otros que no eran, esta me agrado un poco más, ya que de igual forma solo tomamos el cuerpo del carro como nodo padre y los demás como nodos hijos, esto me ayudo a comprender un poco más cómo funciona la jerarquía y de igual forma me ayudo para saber y entender mas el funcionamiento de Blender.