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: Arroyo Ramìrez Carlos Alberto

Nº de Cuenta: 320185865

GRUPO DE LABORATORIO: 03

GRUPO DE TEORÍA: 04

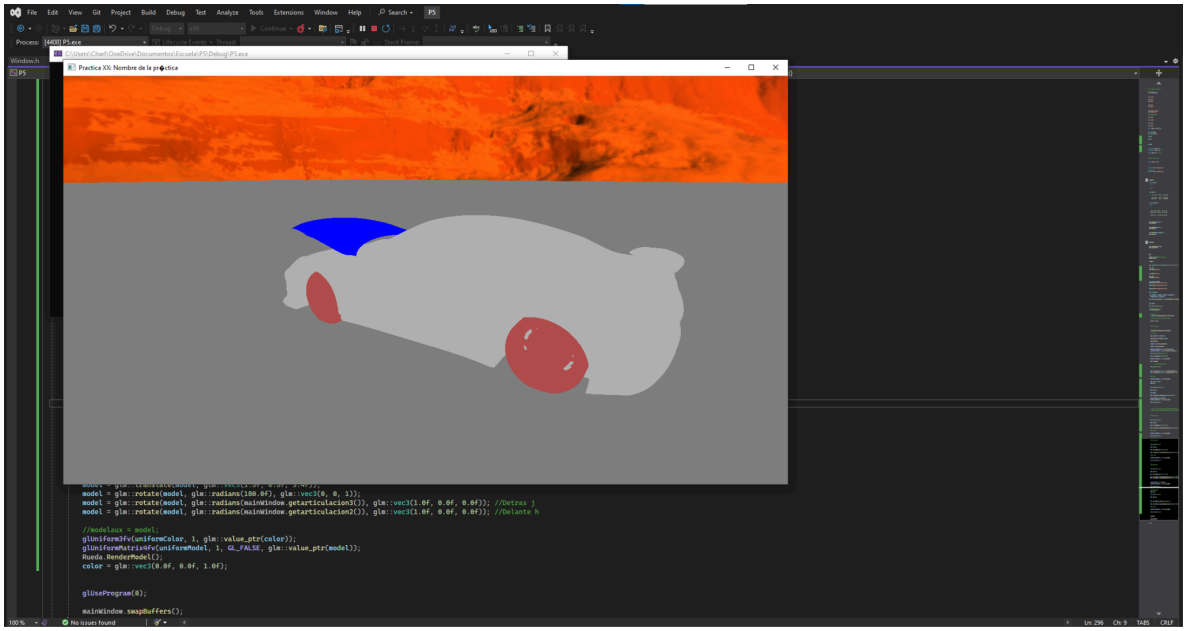
SEMESTRE 2026-1

FECHA DE ENTREGA LÍMITE: 27/09/2025

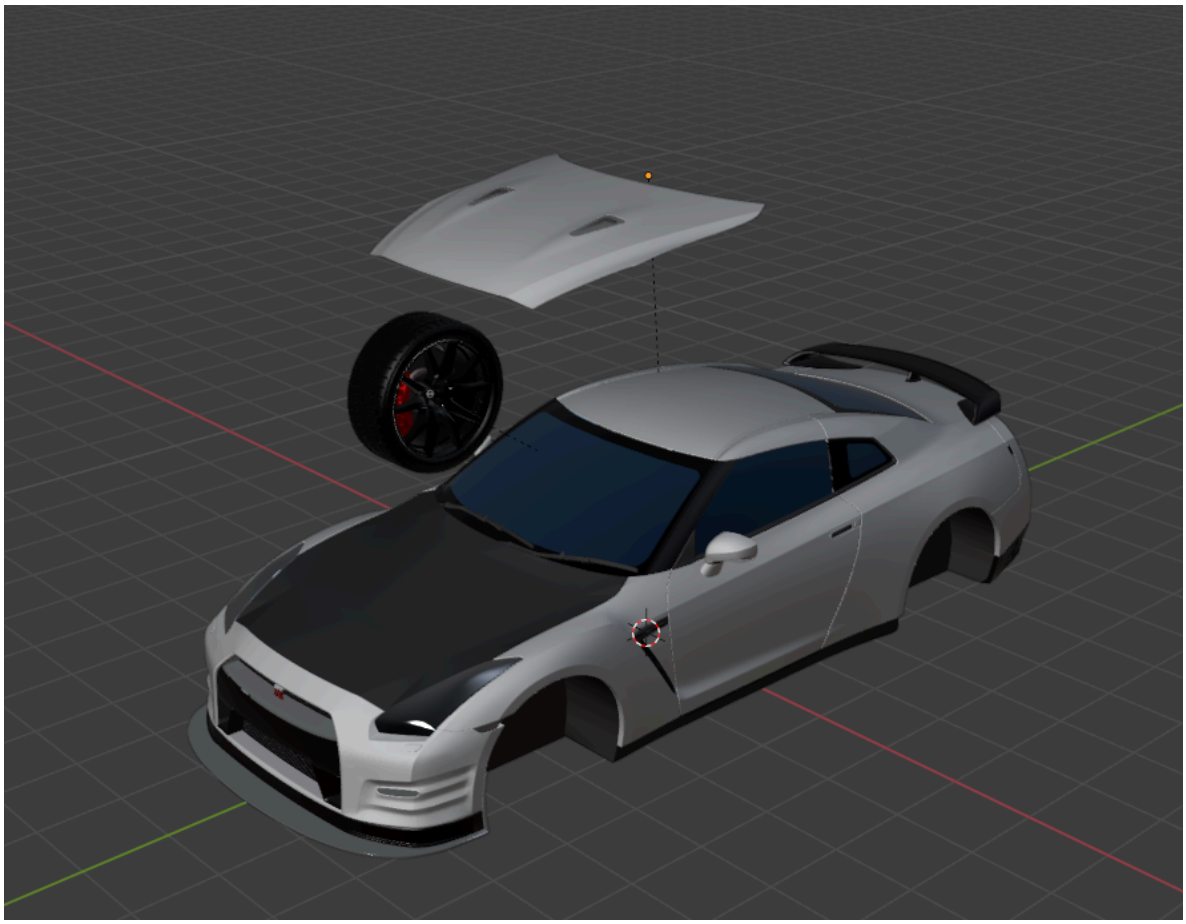
CALIFICACIÓN: _____

REPORTE DE PRÁCTICA:

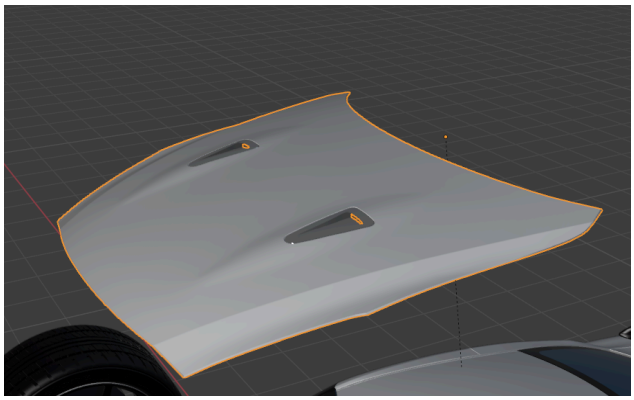
1.- Ejecución de los ejercicios que se dejaron, comentar cada uno y capturas de pantalla de bloques de código generados y de ejecución del programa.



En esta práctica realice un carro, todo inicio desde la exportación en Blender, tuve que unir todas las piezas de la rueda, todo el cuerpo del carro y el cofre, el resto de mi modelo 3d como los interiores no los tome en cuenta ya que no se ven, los exporte en formato obj:



Cabe aclarar que el centro del origen para cada pieza vario para mantener un buen ángulo de giro en el caso del cofre y la rueda:



Posteriormente guarde los modelos fbx a mi carpeta de Models que había utilizado en el ejercicio en clase 5 e hice algunas modificaciones pero cosas muy similares, para empezar declarar modelos hasta arriba de mi código:

```
Camera camera;
Model Cuerpo;
Model Rueda;
Model Cofre;
```

También declare ángulos de giro, para restringir el movimiento del cofre:

```
// Rangos (grados)
static constexpr float COFRE_MIN = -1.0f;  //  -1°
static constexpr float COFRE_MAX = 45.0f;  //  +45°
```

Después en mi main importe los modelos OBJ:

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

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

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

Dentro del while declare una variable extra para variar de una forma facil el ángulo de rotación del cofre:

```
//Limitando angulos:
float cofreDeg = glm::clamp(mainWindow.getarticulacion1(), COFRE_MIN, COFRE_MAX);
```

Después solo dibuje y ajuste algunas transformaciones como la traslación y rotación en el cuerpo, también guarde la matriz para poder usarla y parentar las ruedas y el cofre para que se muevan junto con el cuerpo:

```
//------------*INICIA DIBUJO DE NUESTROS DEMÁS OBJETOS------------------*
//Cuerpo
color = glm::vec3(0.7f, 0.7f, 0.7f);

model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, -1.7f, -(mainWindow.getarticulacion2())/100));
model = glm::translate(model, glm::vec3(0.0f, 0.0f, -(mainWindow.getarticulacion3()) / 100));
glm::mat4 modelCoche = model;

//modelaux = model;
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Cuerpo.RenderModel();
color = glm::vec3(0.1f, 0.0f, 0.0f);
modelaux = model;

//Cofre
```

```
//Cofre
color = glm::vec3(0.0f, 0.0f, 1.0f); //Cofre azul

model = glm::mat4(1.0);

model = modelCoche;
model = glm::translate(model, glm::vec3(0.0f, 2.0f, -2.0f));
model = glm::rotate(model, glm::radians(cofreDeg), glm::vec3(1.0f, 0.0f, 0.0f)); //F y G
//modelaux = model;
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Cofre.RenderModel();
color = glm::vec3(0.0f, 0.0f, 1.0f);
```

```
//rueda delantera izquierda
modelaux = model;

color = glm::vec3(0.7f, 0.3f, 0.3f);

model = glm::mat4(1.0);
model = modelCoche;
model = glm::translate(model, glm::vec3(-1.5f, 0.5f, -3.4f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion3()), glm::vec3(-1.0f, 0.0f, 0.0f)); //Detras j
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()), glm::vec3(-1.0f, 0.0f, 0.0f)); //Delante h

//modelaux = model;
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Rueda.RenderModel();
color = glm::vec3(0.0f, 0.0f, 1.0f);
```

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.

Me tomo un poco de tiempo ajustar la rotación de las ruedas, ya que era aplicar un giro de 180 grados pero eso significa que para que las 4 ruedas giraran parejo al avanzar o retroceder, los controles deben ser lo contrario:

```
//rueda delantera izquierda
modelaux = model;

color = glm::vec3(0.7f, 0.3f, 0.3f);

model = glm::mat4(1.0);
model = modelCoche;
model = glm::translate(model, glm::vec3(-1.5f, 0.5f, -3.4f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion3()), glm::vec3(-1.0f, 0.0f, 0.0f)); //Detras j
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()), glm::vec3(-1.0f, 0.0f, 0.0f)); //Delante h

//modelaux = model;
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Rueda.RenderModel();
color = glm::vec3(0.0f, 0.0f, 1.0f);
```

```
//rueda delantera derecha
modelaux = model;

color = glm::vec3(0.7f, 0.3f, 0.3f);

model = glm::mat4(1.0);
model = modelCoche;
model = glm::translate(model, glm::vec3(1.5f, 0.5f, -3.4f));
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0, 0, 1));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion3()), glm::vec3(1.0f, 0.0f, 0.0f)); //Detras j
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()), glm::vec3(1.0f, 0.0f, 0.0f)); //Delante h

//modelaux = model;
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Rueda.RenderModel();
color = glm::vec3(0.0f, 0.0f, 1.0f);
```

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

En general, este ejercicio me resultó de una complejidad media, ya que fue muy parecido al que hicimos en clase con el perro robot. Gracias a eso, no tuve tantas dificultades para completarlo. La principal diferencia fue que ahora pude controlar la traslación del coche, lo que me permitió mover todo el modelo de manera independiente.

Otro reto fue el manejo de las ruedas: al reutilizar un solo modelo y querer que las partes externas quedarán expuestas, tuve que aplicar un giro de 180 grados y, además, invertir los controles simplemente agregando un -1 en el vector. Esto me ayudó a que todas girarán de forma correcta cuando el coche avanzaba o retrocedía.

Algo importante de esta práctica fue reforzar el tema de las jerarquías, ya que el cuerpo se convirtió en el padre y tanto el cofre como las ruedas quedaron como

hijos. Con esto repasé la idea de cómo al mover el cuerpo, automáticamente se trasladan también las demás piezas.

No tengo algún comentario adicional o sugerencia para la práctica, con el ejercicio en clase fue suficiente para entender los conceptos.

En conclusión aprendí que este ejercicio fue muy similar al del perro robot y por eso no se me complicó mucho, además entendí cómo aplicar la traslación del coche y el ajuste de las ruedas con un giro de 180° e invertir controles. También repasé el uso de jerarquías, lo que me ayudó a reforzar lo visto en prácticas anteriores.

1. Bibliografía en formato APA

LearnOpenGL. (s. f.). Model Loading / Assimp. Recuperado de https://learnopengl.com/Model-Loading/Assimp

GLFW. (s. f.). Input guide. Recuperado de https://www.glfw.org/docs/3.3/input_guide.html

Bock, D. (s. f.). Hierarchical Transformations in OpenGL. Recuperado de https://remy.parkland.edu/~dbock/Class/csc231/Lecture/HierarchicalTransformations.html

Lighthouse3d.com. (s. f.). Importing 3D Models with Assimp. Recuperado de https://www.lighthouse3d.com/cg-topics/code-samples/importing-3d-models-with-assimp/