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 06

NOMBRE COMPLETO: Sanchez Calvillo Saida Mayela

Nº de Cuenta: 318164481

GRUPO DE LABORATORIO: 02

GRUPO DE TEORÍA: 04

SEMESTRE 2025-2

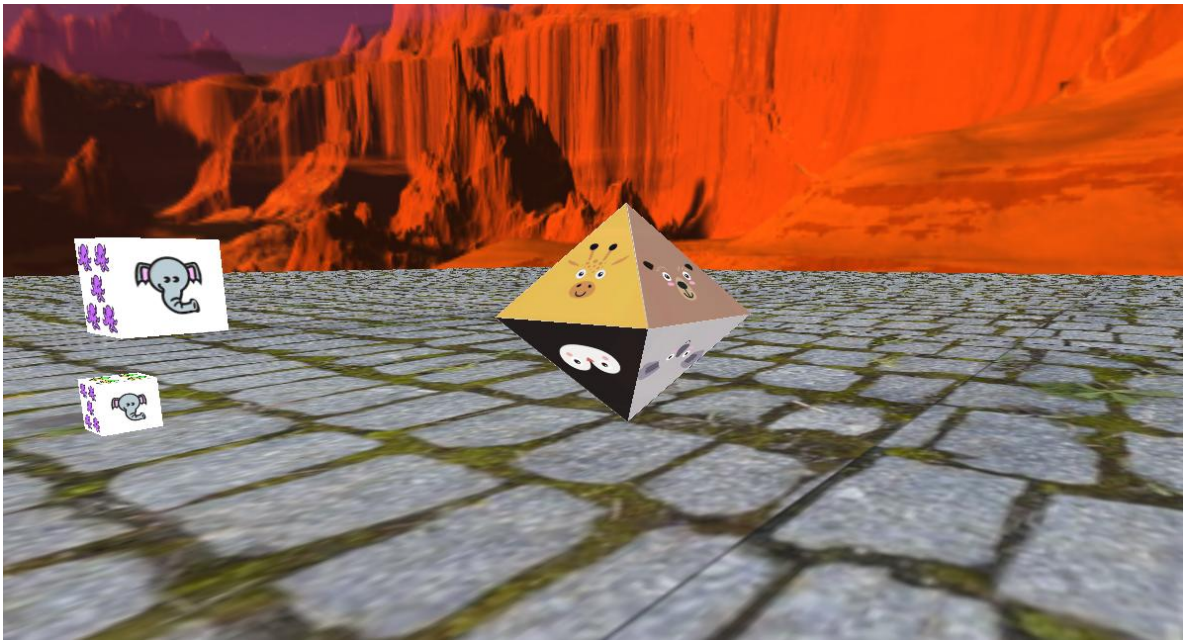
FECHA DE ENTREGA LÍMITE: 29 de marzo 2025

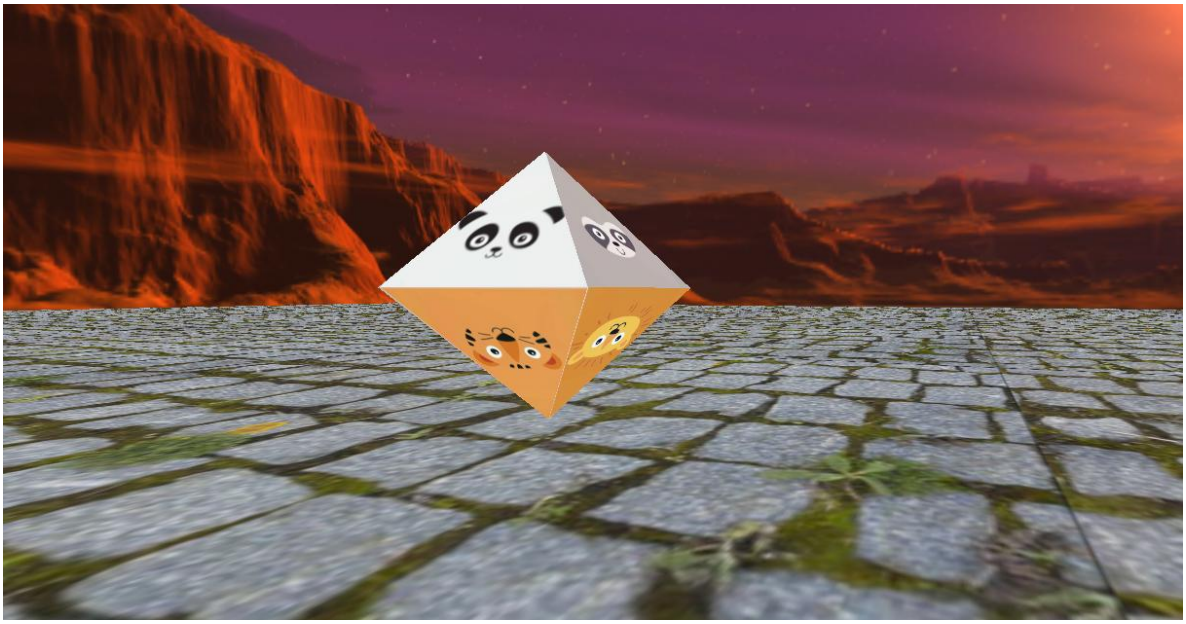
CALIFICACIÓN: _____

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.

Ejercicio 1: Crear un dado de 8 caras y texturizarlo por medio de código.





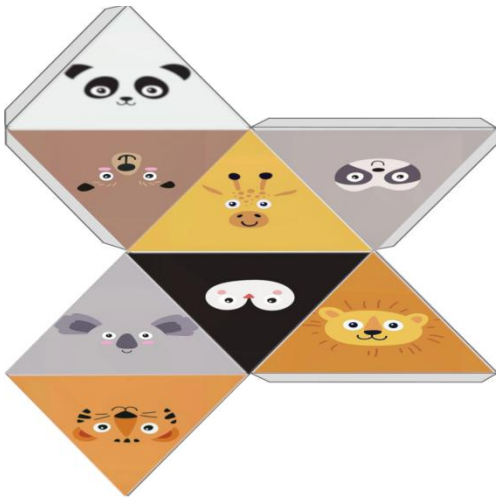
Primero creamos nuestro dado de 8 caras, esto lo hicimos guiándonos de la función para nuestro cubo, solo que ahora obtendríamos 24 vértices (3 por cada cara)

```
void CrearOctaedro()
{
    unsigned int octaedro_indices[] = {
        // Cara superior 1
        0, 1, 2,
        // Cara superior 2
        3, 4, 5,
        // Cara superior 3
        6, 7, 8,
        // Cara superior 4
        9, 10, 11,

        // Cara inferior 1
        12, 13, 14,
        // Cara inferior 2
        15, 16, 17,
        // Cara inferior 3
        18, 19, 20,
        // Cara inferior 4
        21, 22, 23,
    };

    GLfloat octaedro_vertices[] = {
        // Vértices          |   Coordenadas     |  Normales
        //x     y       z       S       T           NX      NY      NZ
```

```
    GLfloat octaedro_vertices[] = {
        // Vértices          |   Coordenadas     |  Normales
        //x     y       z       S       T           NX      NY      NZ
        // Mitad superior
        0.5f,  0.0f,  0.5f,    0.25f,   0.5f,      0.0f,   -1.0f,   -1.0f,// 0   enfrente jirafaa
        -0.5f, 0.0f,  0.5f,    0.75f,   0.5f,      0.0f,   -1.0f,   -1.0f,// 3
        0.0f,  0.6f,  0.0f,    0.5f,    0.75f,     0.0f,   -1.0f,   -1.0f,// 4

        -0.5f, 0.0f,  0.5f,    0.99f,   0.74f,      -1.0f,  -1.0f,    0.0f,// 3   derecha mapache
        -0.5f, 0.0f, -0.5f,    0.52f,    0.74f,     -1.0f,  -1.0f,    0.0f,// 2
        0.0f,  0.6f,  0.0f,    0.75f,   0.5f,      -1.0f,  -1.0f,    0.0f,// 4

        -0.5f, 0.0f, -0.5f,    0.01f,   0.75f,     0.0f,   -1.0f,    1.0f,// 2   atras  panda
        0.5f,  0.0f, -0.5f,    0.5f,    0.75f,     0.0f,   -1.0f,    1.0f,// 1
        0.0f,  0.6f,  0.0f,    0.25f,   0.99f,     0.0f,   -1.0f,    1.0f,// 4

        0.5f,  0.0f, -0.5f,    0.48f,    0.74f,     1.0f,   -1.0f,    0.0f,// 1   izquierda oso
        0.5f,  0.0f,  0.5f,    0.01f,   0.74f,      1.0f,   -1.0f,    0.0f,// 0
        0.0f,  0.6f,  0.0f,    0.25f,    0.5f,     1.0f,   -1.0f,    0.0f,// 4

        // Mitad inferior
        0.5f,  0.0f,  0.5f,    0.27f,   0.49f,     0.0f,   1.0f,    -1.0f,// 0   enfrente  pinguino
        -0.5f, 0.0f,  0.5f,    0.74f,   0.49f,     0.0f,   1.0f,    -1.0f,// 3
        0.0f, -0.6f,  0.0f,    0.5f,    0.25f,     0.0f,   1.0f,    -1.0f,// 5
```

```
        // Mitad inferior
        0.5f,  0.0f,  0.5f,    0.27f,   0.49f,     0.0f,   1.0f,    -1.0f,// 0   enfrente  pinguino
        -0.5f, 0.0f,  0.5f,    0.74f,   0.49f,     0.0f,   1.0f,    -1.0f,// 3
        0.0f, -0.6f,  0.0f,    0.5f,    0.25f,     0.0f,   1.0f,    -1.0f,// 5

        -0.5f, 0.0f,  0.5f,    0.98f,   0.26f,     -1.0f,  1.0f,    0.0f,// 3   derecha leon
        -0.5f, 0.0f, -0.5f,    0.51f,   0.26f,     -1.0f,  1.0f,    0.0f,// 2
        0.0f, -0.6f,  0.0f,    0.75f,   0.5f,      -1.0f,  1.0f,    0.0f,// 5

        -0.5f, 0.0f, -0.5f,    0.01f,   0.25f,     0.0f,   1.0f,    1.0f,// 2   atras tigre
        0.5f,  0.0f, -0.5f,    0.5f,    0.25f,     0.0f,   1.0f,    1.0f,// 1
        0.0f, -0.6f,  0.0f,    0.25f,   0.01f,     0.0f,   1.0f,    1.0f,// 5

        0.5f,  0.0f, -0.5f,    0.5f,    0.25f,     1.0f,   1.0f,    0.0f,// 1   izquierda Koala
        0.5f,  0.0f,  0.5f,    0.01f,   0.25f,     1.0f,   1.0f,    0.0f,// 0
        0.0f, -0.6f,  0.0f,    0.25f,   0.5f,      1.0f,   1.0f,    0.0f,// 5


    };

    Mesh* octaedro = new Mesh();
    octaedro->CreateMesh(octaedro_vertices, octaedro_indices, 192, 24);
    meshList.push_back(octaedro);
}
```

Para las coordenadas UV utilizamos la siguiente imagen



Las normales las dejamos iguales por el momento.

Después declaramos la textura

```
Texture brickTexture;
Texture dirtTexture;
Texture plainTexture;
Texture pisoTexture;
Texture dadoTexture;
Texture dadoTextureAnimal;
Texture logofiTexture;
Texture octaedroTexturaAnimal;
```

Y cargamos nuestra textura

```
octaedroTexturaAnimal = Texture("Textures/octaedro.png");
octaedroTexturaAnimal.LoadTextureA();
```

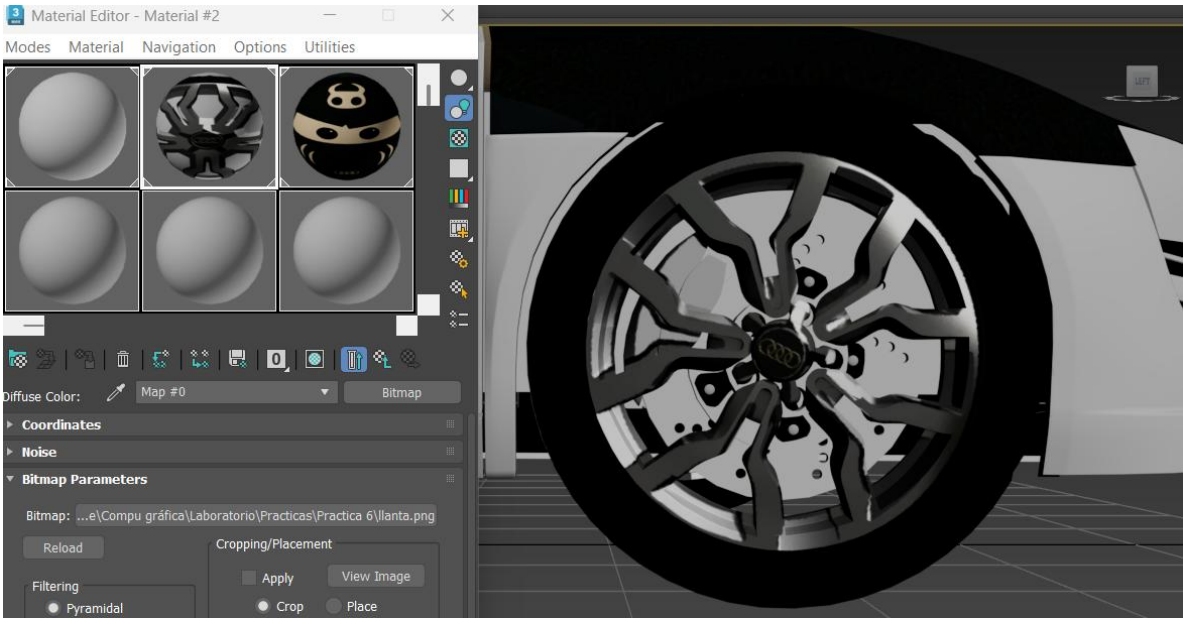
Como metimos nuestro octaedro dentro de nuestra Mesh List, recordamos que se encuentra en la posición 5 y la mandamos a llamar.

```
//EJERCICIO 1
//Octaedro
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(5.0f, 4.5f, -2.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
octaedroTexturaAnimal.UseTexture();
meshList[5]->RenderMesh();
```

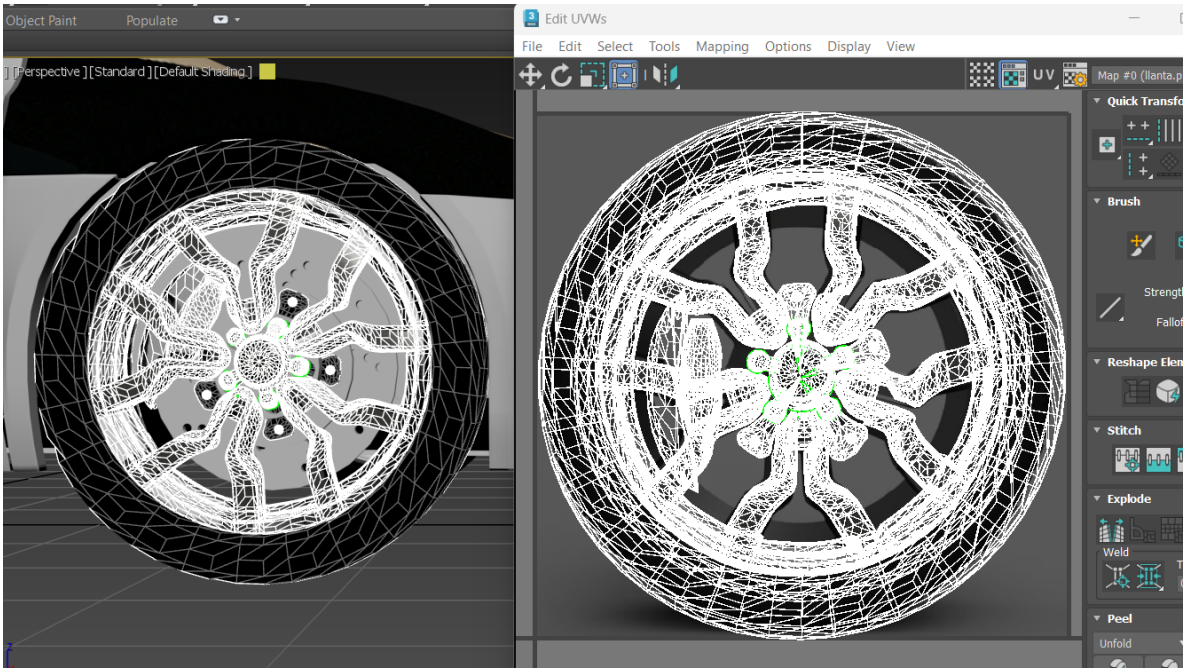
Ejercicio 2: Importar el modelo de su coche con sus 4 llantas acomodadas y tener texturizadas las 4 llantas (diferenciar caucho y rin)

Para este ejercicio se buscó una imagen que asemejara a la llanta de nuestro carro, y como es un Audio R8 se buscó una llanta de dicho modelo de auto, hasta que se encontró una parecida

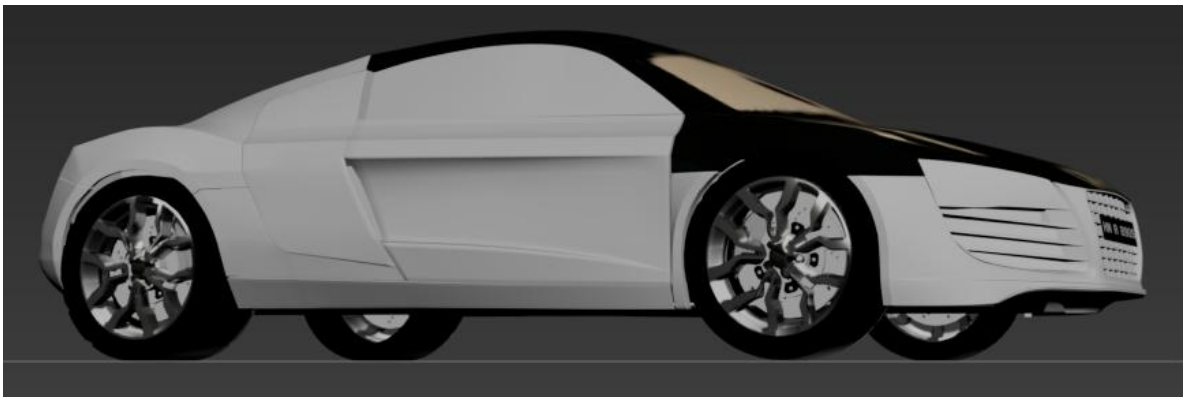
Luego incluimos esta textura en nuestro “Material editor” con las configuraciones correspondientes y la pasamos a nuestra llanta.



Una vez pasada la textura a nuestro objeto, abrimos el Unwrap UV para poder acomodar nuestra llanta a dicha textura, moviendo vértice por vértice, línea por línea o figura por figura.



Este proceso se repitió 4 veces, una por cada llanta que tenemos. Lo que nos da un resultado así.



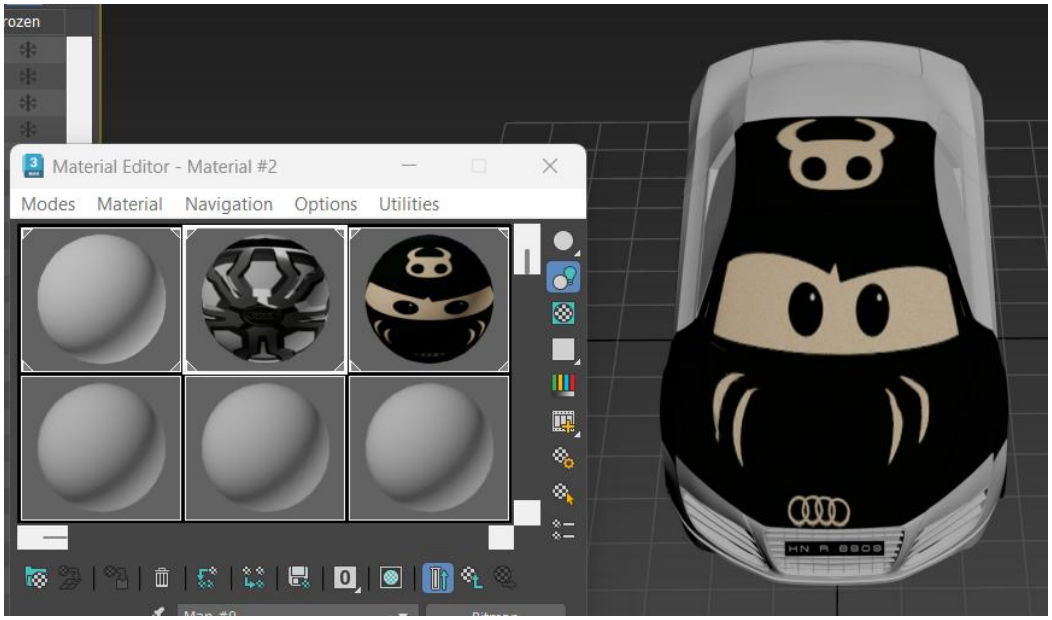
Para exportarlo, recordamos lo que vimos la practica pasada. Cada objeto lo ponemos en el centro de nuestro plano, lo exportamos en formato .obj y nos preparamos para empezar a modificar nuestro código.

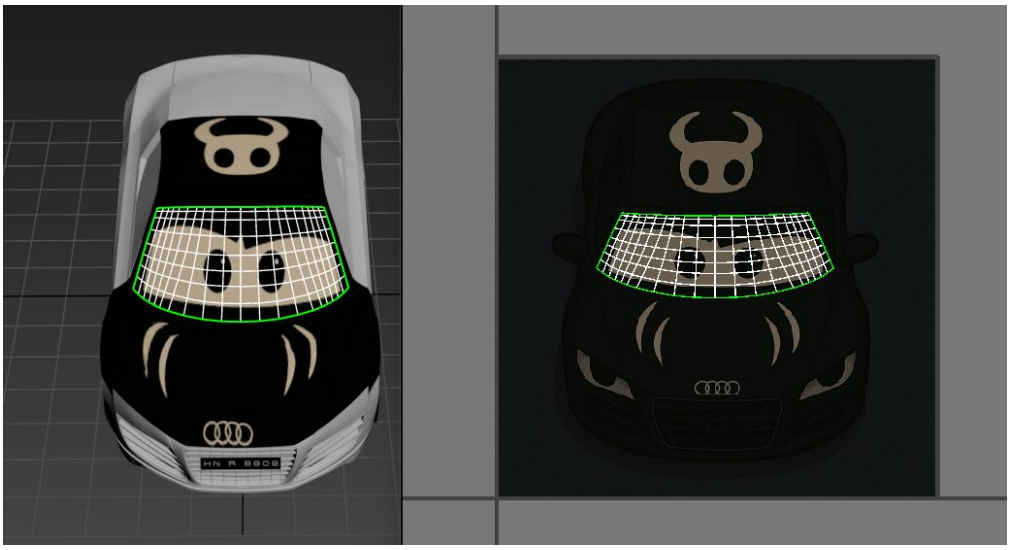
-La parte de código será explicado en el ejercicio 3.

Ejercicio 3: Texturizar la cara del personaje de la imagen tipo cars en el espejo (ojos) y detalles en cofre y parrilla de su propio modelo de coche

La imagen de nuestro coche estilo animación cars se cambió, ahora usamos esta para este ejercicio

Como lo hicimos con las llantas, primero se exporto la textura y se puso sobre el objeto que queríamos texturizar





Y así con cada parte que queremos texturizar.

Para la parte de código, vamos a declarar nuestros modelos y los llenamos con nuestro formato correspondiente.

```
Model Kitt_M;
Model Llanta_M;
Model Carro;
Model LlantaDelanteraIzq;
Model LlantaDelanteraDer;
Model LlantaTraseraIzq;
Model LlantaTraseraDer;
Model Cofre;
Model Parabrisas;
Model Techo;
//Model Dado_M;
Model Cubo_Animales;
```

```
Kitt_M.LoadModel("Models/kitt_optimizado.obj");
Carro = Model();
Carro.LoadModel("Models/CarroTexturizado.obj");
LlantaDelanteraIzq = Model();
LlantaDelanteraIzq.LoadModel("Models/LlantaDelanteraIzq.obj");
LlantaDelanteraDer = Model();
LlantaDelanteraDer.LoadModel("Models/LlantaDelanteraDer.obj");
LlantaTraseraIzq = Model();
LlantaTraseraIzq.LoadModel("Models/LlantaTraseraIzq.obj");
LlantaTraseraDer = Model();
LlantaTraseraDer.LoadModel("Models/LlantaTraseraDer.obj");
Llanta_M = Model();
Llanta_M.LoadModel("Models/llanta_optimizada.obj");
Cofre = Model();
Cofre.LoadModel("Models/Cofre.obj");
Parabrisas = Model();
Parabrisas.LoadModel("Models/Parabrisas.obj");
Techo = Model();
Techo.LoadModel("Models/Techo.obj");
Cubo_Animales = Model();
Cubo_Animales.LoadModel("Models/dado_animales.obj");
```

Como cada objeto antes de exportarlo lo colocamos y el centro de nuestro plano, cada uno lo vamos a trasladar y acomodar donde va y así darle la forma a nuestro modelo exportado

```
//Instancia del coche
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(20.0f, -1.92f, -5.0f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
model = glm::scale(model, glm::vec3(0.5f, 0.5f, 0.5f));
model = glm::translate(model, glm::vec3(0.0f + mainWindow.getmuevex() , -0.5f, -3.0f));
modelaux = model;
model = glm::rotate(model, -90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Carro.RenderModel();

//Cofre
model = glm::translate(model, glm::vec3(0.0f + mainWindow.getmuevex(), -0.5f, -3.0f));
model = modelaux;
model = glm::rotate(model, -90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Cofre.RenderModel();

//Parabrisas
model = glm::translate(model, glm::vec3(0.0f + mainWindow.getmuevex(), -0.5f, -3.0f));
model = modelaux;
model = glm::rotate(model, -90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Parabrisas.RenderModel();
```

```
//Techo
model = glm::translate(model, glm::vec3(0.0f + mainWindow.getmuevex(), -0.5f, -3.0f));
model = modelaux;
model = glm::rotate(model, -90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Techo.RenderModel();

//Llanta delantera Derecha
model = modelaux;
model = glm::translate(model, glm::vec3(-35.8f, 10.0f, 24.0f));
model = glm::rotate(model, -90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
LlantaDelanteraIzq.RenderModel();

//Llanta Delantera Izquierda
model = modelaux;
model = glm::translate(model, glm::vec3(-35.8f, 10.0f, -24.0f));
model = glm::rotate(model, -90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
//model = glm::scale(model, glm::vec3(0.4f, 0.4f, 0.4f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
LlantaDelanteraDer.RenderModel();

//Llanta Trasera Izquierda
model = modelaux;
model = glm::translate(model, glm::vec3(41.0f, 10.0f, -24.0f));
model = glm::rotate(model, 90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
//model = glm::scale(model, glm::vec3(0.4f, 0.4f, 0.4f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
LlantaTraseraIzq.RenderModel();
```

```
//Llanta trasera derecha
model = modelaux;
model = glm::translate(model, glm::vec3(41.0f, 10.0f, 24.0f));
model = glm::rotate(model, 90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
//model = glm::scale(model, glm::vec3(0.4f, 0.4f, 0.4f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
LlantaTraseraDer.RenderModel();
```

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 primer problema que se tuvo fue entender como se modificaban las texturas dentro de nuestros objetos con Unwrap, pero se resolvió viendo el video que el profesor nos compartió en el classroom.

3.- Conclusión:

La explicación es buena, solo que sigue siendo rápida y aunque se vea el video del classroom, aun me gustaría que el profesor nos diera más técnicas para resolver las cosas después de que entregamos nuestros trabajos.

a. Conclusión

El texturizado es un poco impráctico ya que si queremos que se vea bien, tenemos que mover vértice por vértice y puede llegar a ser un problema, también porque se ve afectada la luz o los reflejos si movemos mal los vértices.