



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 N° 06

NOMBRE COMPLETO: Vázquez Gómez Carlos Iván

N° de Cuenta: 4200551851

GRUPO DE LABORATORIO: 03

GRUPO DE TEORÍA: 04

SEMESTRE 2025-1

FECHA DE ENTREGA LÍMITE: 28/09/2024

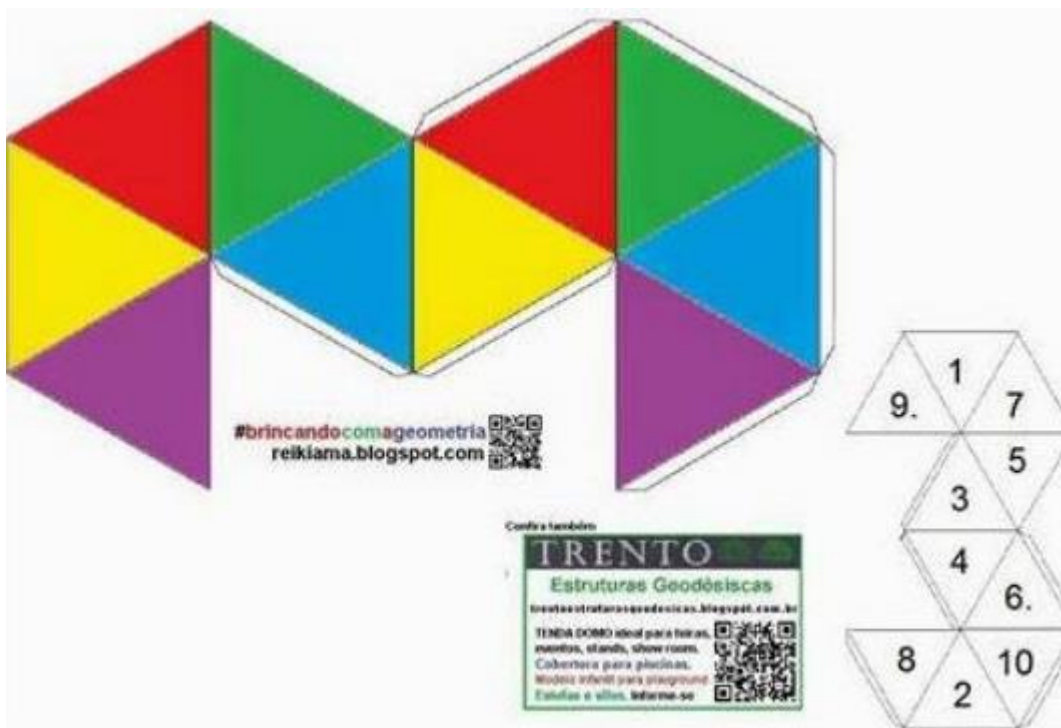
CALIFICACIÓN: _____

REPORTE DE PRÁCTICA:















1.- Actividades

Ejercicio 1: Crear un dado de 10 caras y texturizarlo por medio de código.

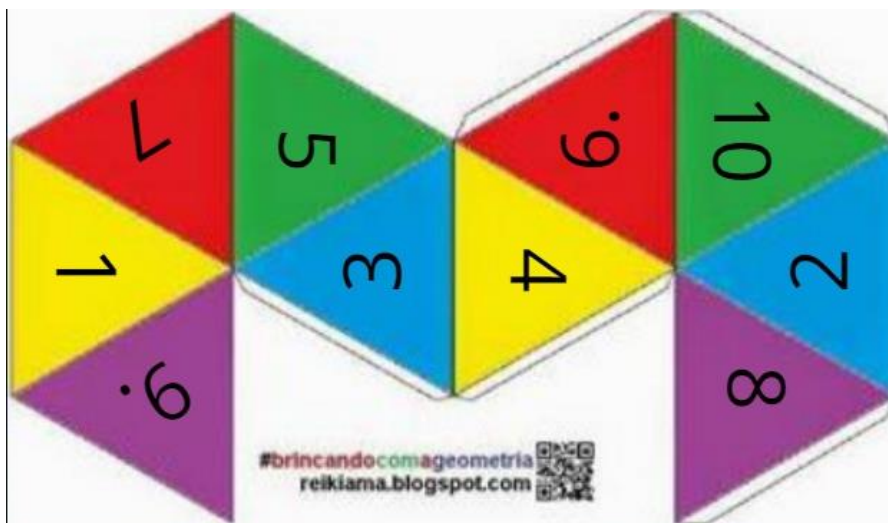
Para realizar este ejercicio tuve que utilizar la herramienta de GeoGebra para obtener los vértices para realizar la figura solicitada que era un dado de 10 caras, guiándonos con la siguiente imagen



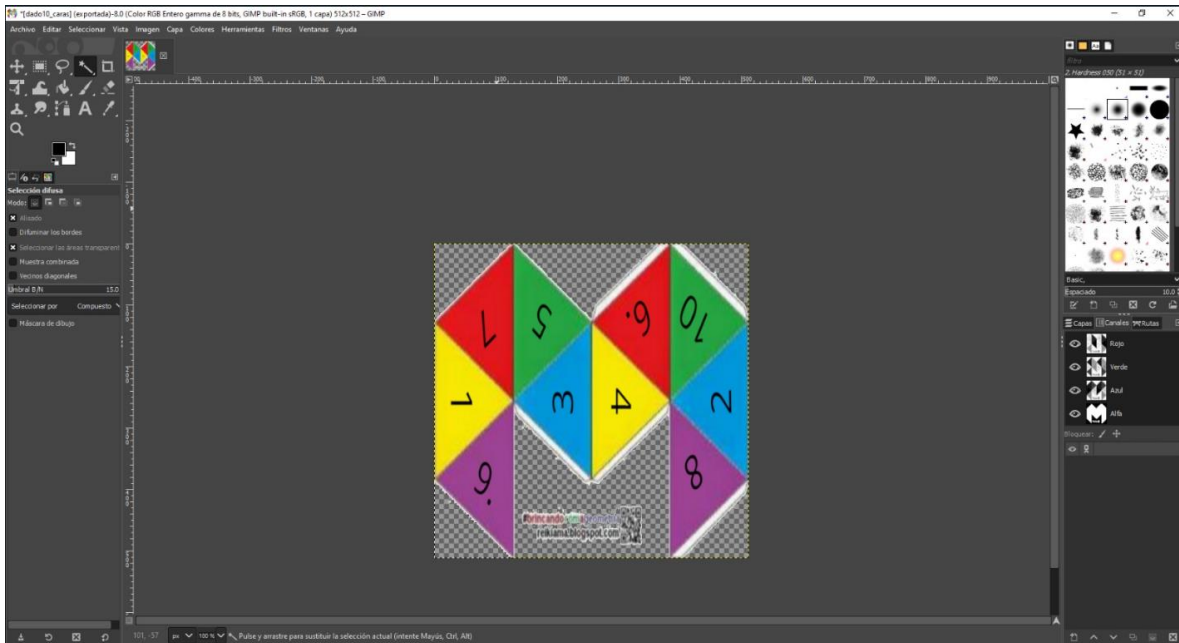
Puntos en GeoGebra

	A = (1, 0)	
	B = (0.31, 0.95)	...
	C = (-0.81, 0.59)	...
	D = (-0.81, -0.59)	...
	E = (0.31, -0.95)	...
	f : Recta(C, D) = $x = -0.81$...
	g : Recta(C, B) = $-0.36x + 1.12y = 0.95$...
	h : Recta(B, A) = $0.95x + 0.69y = 0.95$...
	i : Recta(A, E) = $0.95x - 0.69y = 0.95$...
	j : Recta(E, D) = $-0.36x - 1.12y = 0.95$...
	F = Interseca(EjeX, EjeY) = (0, 0)	...
	k = Segmento(F, B) = 1	...
	n = Segmento(F, C) = 1	...

La imagen siguiente se utilizará para texturizarlo mediante código (utilice Canva para colocarle los números manualmente)



Esta imagen se obtuvo mediante el programa de GIMP donde se puso de 512x512 con vector Alpha (la dirección de los números cambio, para poder tenerlos correctos en la hora de ejecución)



Se crea la textura en código

```
Texture Dado10caras;
```

Se carga la textura de la carpeta Textures

```
Dado10caras = Texture("Textures/dado10_caras.png");  
Dado10caras.LoadTextureA();
```

Se empieza la creación del mesh “CrearDado10caras”

```
void CrearDado10caras()  
{  
    unsigned int dado10_indices[] = {  
        //1  
        0, 1, 2,  
        //2  
        3, 4, 5,  
        //3  
        6, 7, 8,  
        //4  
        9, 10, 11,  
        //5  
        12, 13, 14,  
        //6  
        15, 16, 17,  
        //7  
        18, 19, 20,  
        //8  
        21, 22, 23,  
        //9  
        24, 25, 26,  
        //10  
        27, 28, 29,  
    };  
}
```

```

GLfloat dado10_vertices[] = {

    // Numero 9
    //x   y       z       S       T       NX       NY       NZ
    0.0f, 1.0f, 0.0f, 0.22f, 0.46f, 0.0f, 0.0f, 0.0f, //0
    1.0f, 0.0f, 0.0f, 0.020f, 0.26f, 0.0f, 0.0f, 0.0f, //1
    0.31f, 0.0f, -0.95f, 0.22f, 0.04f, 0.0f, 0.0f, 0.0f, //2

    //Numero 1
    //x   y       z       S       T       NX       NY       NZ
    0.31f, 1.0f, 0.0f, 0.020f, 0.720f, 0.0f, 0.0f, 0.0f, //3
    -0.81f, 0.0f, -0.59f, 0.020f, 0.29f, 0.0f, 0.0f, 0.0f, //4
    0.0f, 1.0f, 0.0f, 0.22f, 0.492f, 0.0f, 0.0f, 0.0f, //5

    //Numero 7
    //x   y       z       S       T       NX       NY       NZ
    0.0f, 1.0f, 0.0f, 0.24f, 0.527f, 0.0f, 0.0f, 0.0f, //6
    -0.81f, 0.0f, -0.59f, 0.24f, 0.972f, 0.0f, 0.0f, 0.0f, //7
    -0.809f, 0.0f, 0.59f, 0.011f, 0.748f, 0.0f, 0.0f, 0.0f, //8

    //Numero 5
    //x   y       z       S       T       NX       NY       NZ
    -0.809f, 0.0f, 0.59f, 0.476f, 0.751f, 0.0f, 0.0f, 0.0f, //9
    0.309f, 0.0f, 0.95f, 0.255f, 0.97f, 0.0f, 0.0f, 0.0f, //10
    0.0f, 1.0f, 0.0f, 0.257f, 0.527f, 0.0f, 0.0f, 0.0f, //11

    //Numero 3
    //x   y       z       S       T       NX       NY       NZ
    0.0f, 1.0f, 0.0f, 0.257f, 0.496f, 0.0f, 0.0f, 0.0f, //12
    0.309f, 0.0f, 0.95f, 0.488f, 0.277f, 0.0f, 0.0f, 0.0f, //13
    1.0f, 0.0f, 0.0f, 0.488f, 0.722f, 0.0f, 0.0f, 0.0f, //14

    // Numero 4
    //x   y       z       S       T       NX       NY       NZ
    1.0f, 0.0f, 0.0f, 0.51f, 0.277f, 0.0f, 0.0f, 0.0f, //15
    0.0f, -1.0f, 0.0f, 0.738f, 0.503f, 0.0f, 0.0f, 0.0f, //16
    0.31f, 0.0f, -0.95f, 0.51f, 0.722f, 0.0f, 0.0f, 0.0f, //17

    //Numero 6
    //x   y       z       S       T       NX       NY       NZ
    0.31f, 0.0f, -0.95f, 0.517f, 0.746f, 0.0f, 0.0f, 0.0f, //18
    0.0f, -1.0f, 0.0f, 0.742f, 0.521f, 0.0f, 0.0f, 0.0f, //19
    -0.81f, 0.0f, -0.59f, 0.738f, 0.902f, 0.0f, 0.0f, 0.0f, //20

    //Numero 10
    //x   y       z       S       T       NX       NY       NZ
    -0.81f, 0.0f, -0.59f, 0.757f, 0.982f, 0.0f, 0.0f, 0.0f, //21
    0.0f, -1.0f, 0.0f, 0.757f, 0.517f, 0.0f, 0.0f, 0.0f, //22
    -0.809f, 0.0f, 0.59f, 0.976f, 0.748f, 0.0f, 0.0f, 0.0f, //23

    //Numero 2
    //x   y       z       S       T       NX       NY       NZ
    -0.809f, 0.0f, 0.59f, 0.984f, 0.722f, 0.0f, 0.0f, 0.0f, //24
    0.309f, 0.0f, 0.95f, 0.984f, 0.277f, 0.0f, 0.0f, 0.0f, //25
    0.0f, -1.0f, 0.0f, 0.767f, 0.496f, 0.0f, 0.0f, 0.0f, //25

    //Numero 8
    //x   y       z       S       T       NX       NY       NZ
    0.0f, -1.0f, 0.0f, 0.753f, 0.478f, 0.0f, 0.0f, 0.0f, //27
    0.309f, 0.0f, 0.95f, 0.976f, 0.248f, 0.0f, 0.0f, 0.0f, //28
    1.0f, 0.0f, 0.0f, 0.751f, 0.023f, 0.0f, 0.0f, 0.0f, //29

};

Mesh* dado10 = new Mesh();
dado10->CreateMesh(dado10_vertices, dado10_indices, 240, 30);

```

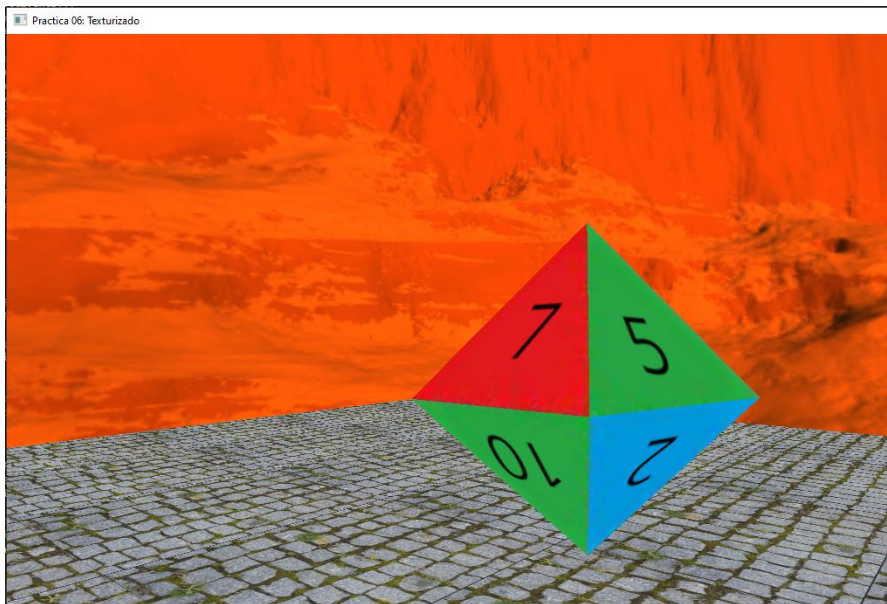
Se texturizo el dado mediante código

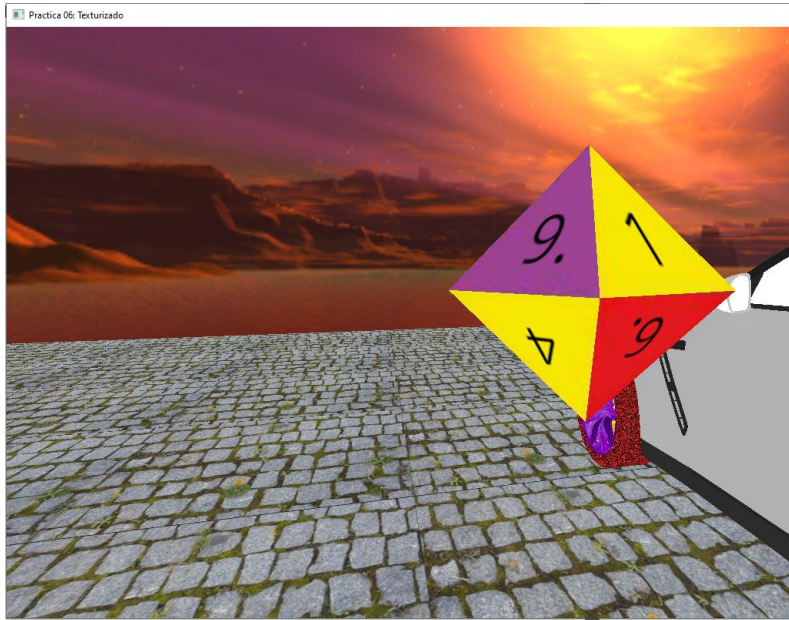
```

//Dado de Opengl
//Ejercicio 1: Texturizar su cubo con la imagen dado_animales ya optimizada por ustedes
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(50.5f, 40.5f, -2.0f));
model = glm::scale(model, glm::vec3(10.0f, 10.0f, 10.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Dado10caras.UseTexture();
meshList[5]->RenderMesh();

```

A continuación, se muestra la ejecución del primer ejercicio





Lo más complicado del ejercicio fue realizar la figura prismática, porque para texturizarlo fue muy sencillo, gracias a que entendí muy bien el ejercicio de clase donde nos guiamos en la imagen y lo dividimos por sectores, y así sabemos cuánto vale cada sección.

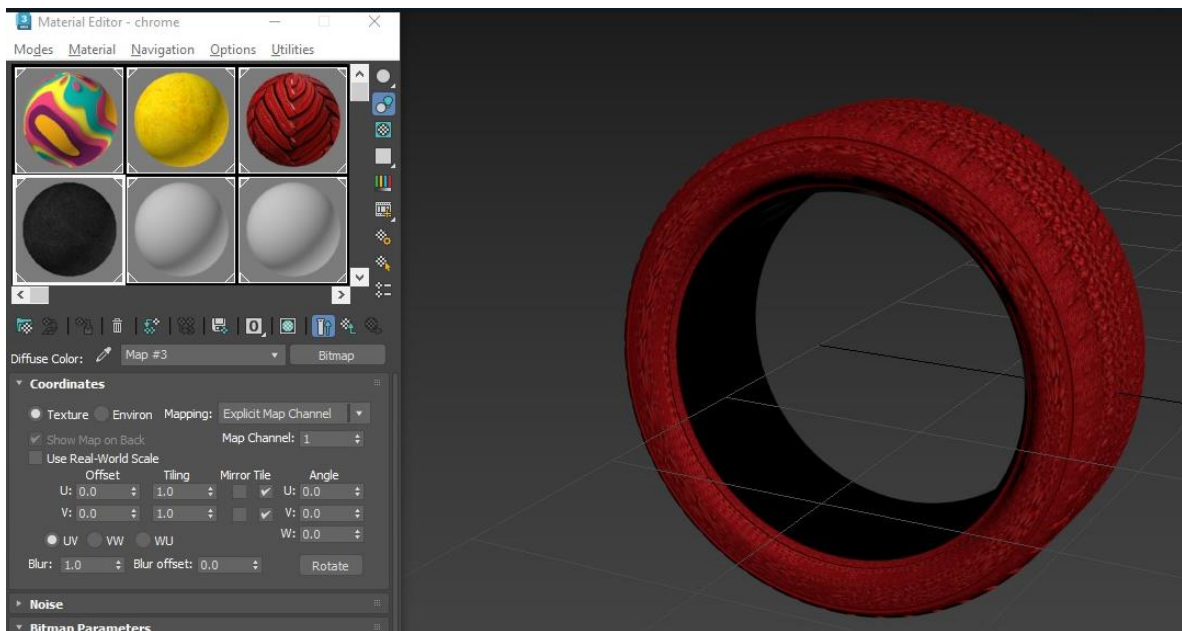
Ejercicio 2: Importar el modelo de su coche con sus 4 llantas acomodadas y tener texturizadas las 4 llantas (diferenciar caucho y rin)

Para realizar este ejercicio primero importe el modelo de mi coche, utilizando los mismos modelos de la practica 5

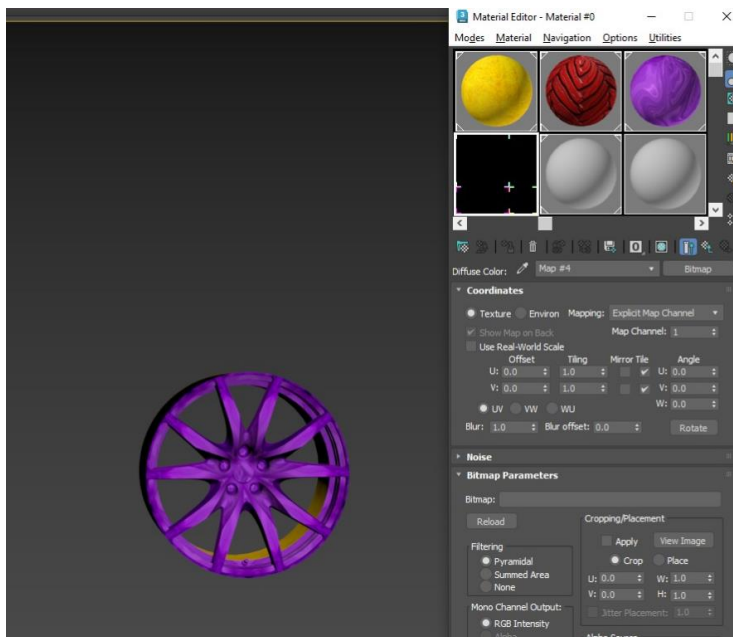
```
Model cuerpo_carro, cofre, llanta_der, llanta_izq; //modelos de carros
cuerpo_carro = Model();
cuerpo_carro.LoadModel("Models/basecarpop2.obj");
cofre = Model();
cofre.LoadModel("Models/cofre2.obj");
llanta_der = Model();
llanta_der.LoadModel("Models/llantaop2.obj");
llanta_izq = Model();
llanta_izq.LoadModel("Models/llantaizq.obj");
```

Se uso 3Ds MAX para realizar esta actividad.

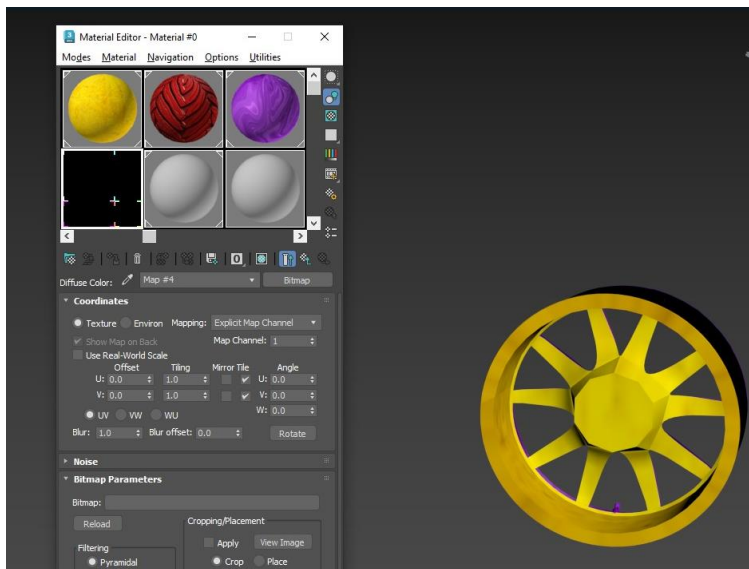
Texturizando las llantas



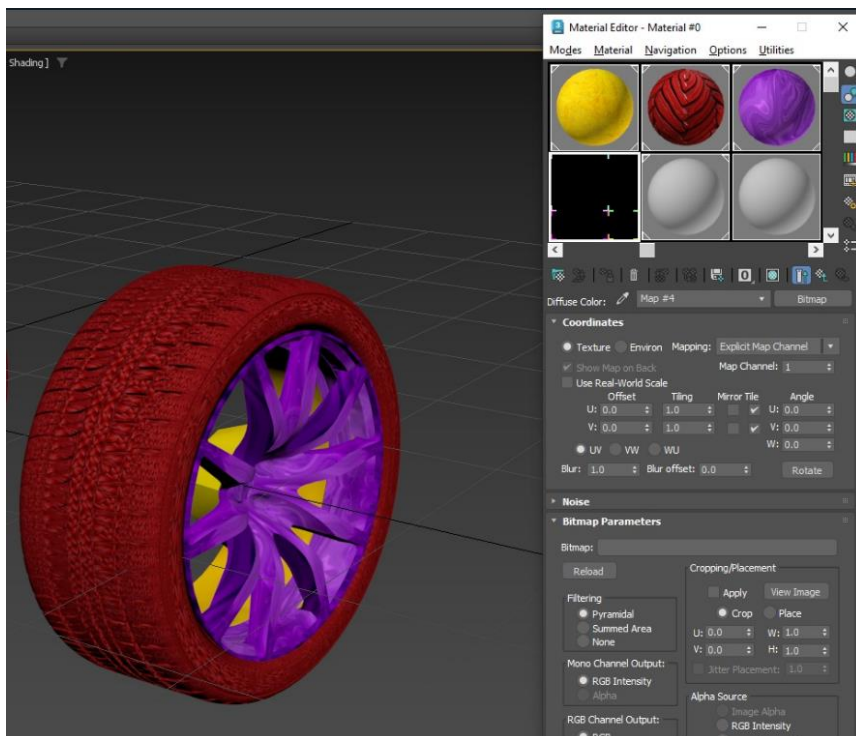
Texturizando los rines



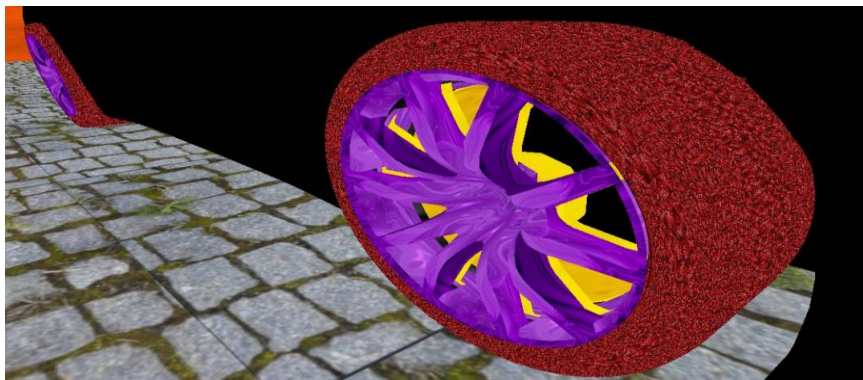
Parte trasera del rin



Llanta y Rin juntos listos para exportar



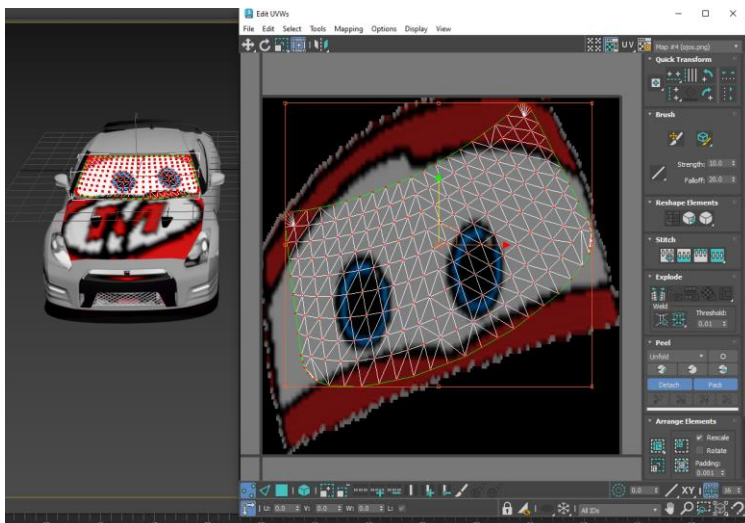
Las siguientes imágenes se muestran las llantas exportadas



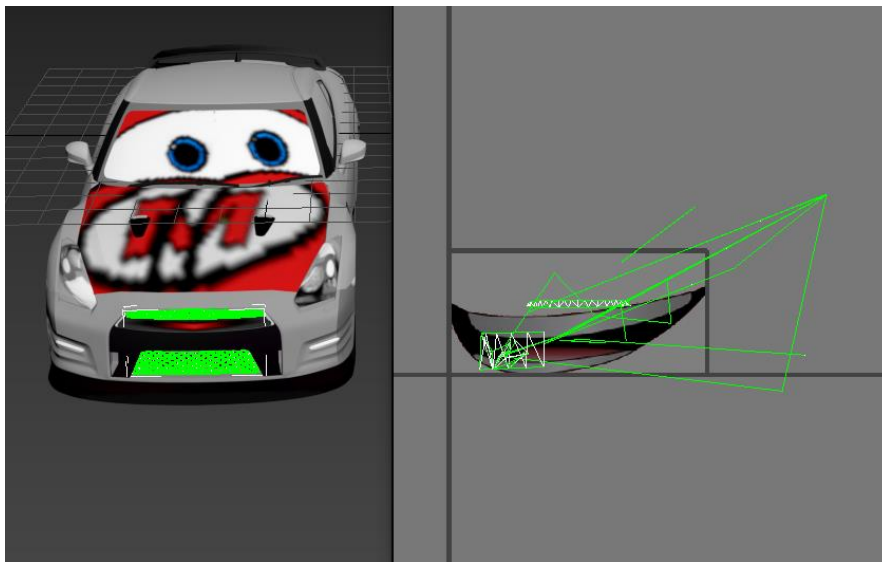
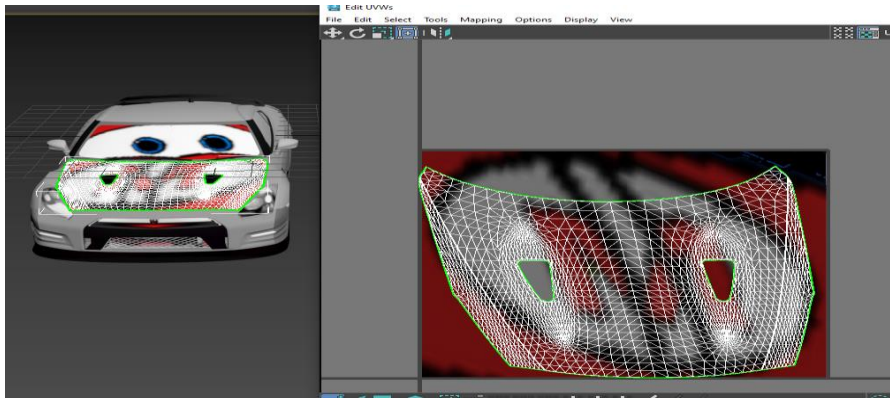
Ejercicio 3: Texturizar la cara del personaje de la imagen tipo cars en el espejo (ojos) y detalles en cofre y parrilla de su propio modelo de coche

Para texturizar como si fuera un personaje de cars no fue tan complicado, excepto por la boca y el bigote de la imagen que proporcione, porque por mas que acomodaba se veía muy raro.

Pero se muestra a continuación la texturización del espejo (ojos)

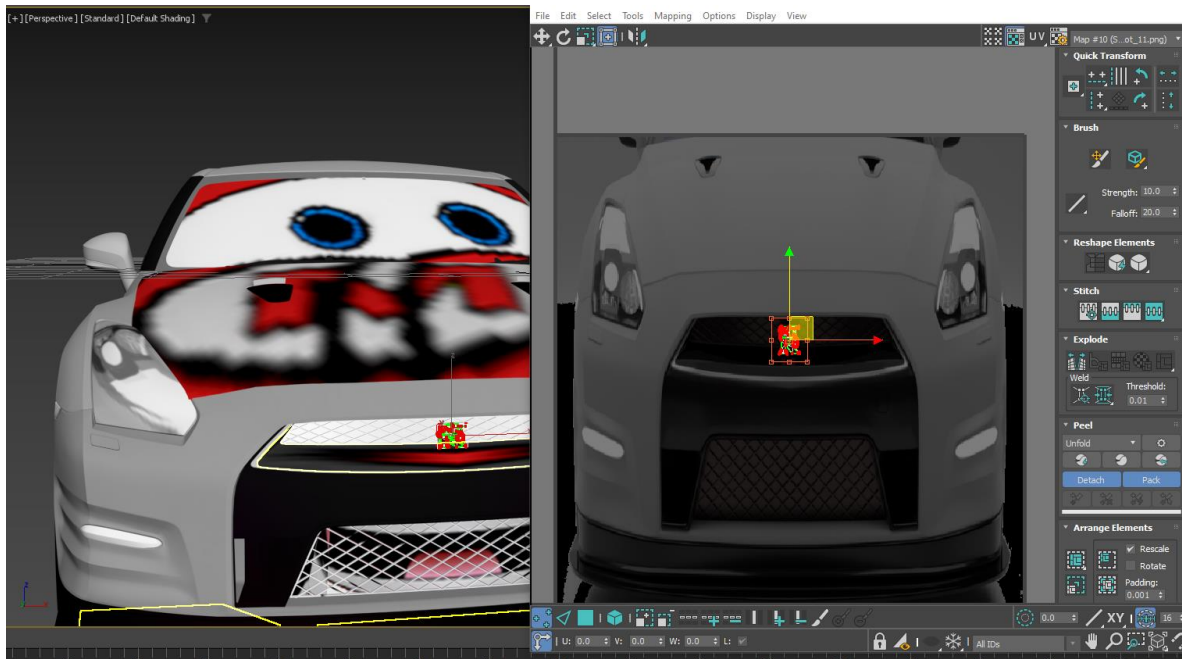


Ahora, se texturiza el cofre



Aquí aprendí, que puedes escoger que parte de un modelo unido puedes texturizar y el otro no se selecciona.

En cuanto a los detalles del modelo como tal, no había tantos, pero coloque los mas relevantes como fue el GT R



Los demás detalles solo era cuestión de color que se fue agregando con la misma textura en todo el carro.

Una vez teniendo nuestros modelos volví a importar los modelos, pero ahora texturizados

```

/*CUERPO CARRO*/
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, 36.0f, 0.0f));
////////model = glm::translate(model, glm::vec3(0.0f, 18.0f, mainWindow.getavanzar())); //se mueve el carro
modelaux = model;
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
cuerpo_carro.RenderModel(); // se muestra el cuerpo del carro

////Cofre
//model = modelaux;
//model = glm::translate(model, glm::vec3(-15.0f, -1.5f, 50.0f));
//model = glm::rotate(model, glm::radians(mainWindow.getarticulacion2()), glm::vec3(-1.0f, 0.0f, 0.0f));*/
//glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
//cofre.RenderModel(); //Se muestra el cofre

//llanta trasera lado derecho
model = modelaux;
model = glm::translate(model, glm::vec3(-46.5f, -24.5f, -40.0f));
model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()), glm::vec3(-1.0f, 0.0f, 0.0f));*/
color = glm::vec3(1.0f, 1.0f, 1.0f); //llanta color blanco
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
llanta_der.RenderModel(); // se muestra llanta vista derecha

//llanta trasera lado izquierdo
model = modelaux;
model = glm::translate(model, glm::vec3(18.0f, -24.5f, -40.0f));
//model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()), glm::vec3(-1.0f, 0.0f, 0.0f));*/
color = glm::vec3(1.0f, 1.0f, 1.0f); //llanta color blanco
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
llanta_izq.RenderModel(); // se muestra llanta vista izquierda

//llanta delantera lado derecho
model = modelaux;
model = glm::translate(model, glm::vec3(-46.5f, -24.5f, 63.0f));
//model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()), glm::vec3(-1.0f, 0.0f, 0.0f));*/
color = glm::vec3(1.0f, 1.0f, 1.0f); //llanta color blanco
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
llanta_der.RenderModel(); // se muestra llanta vista derecha

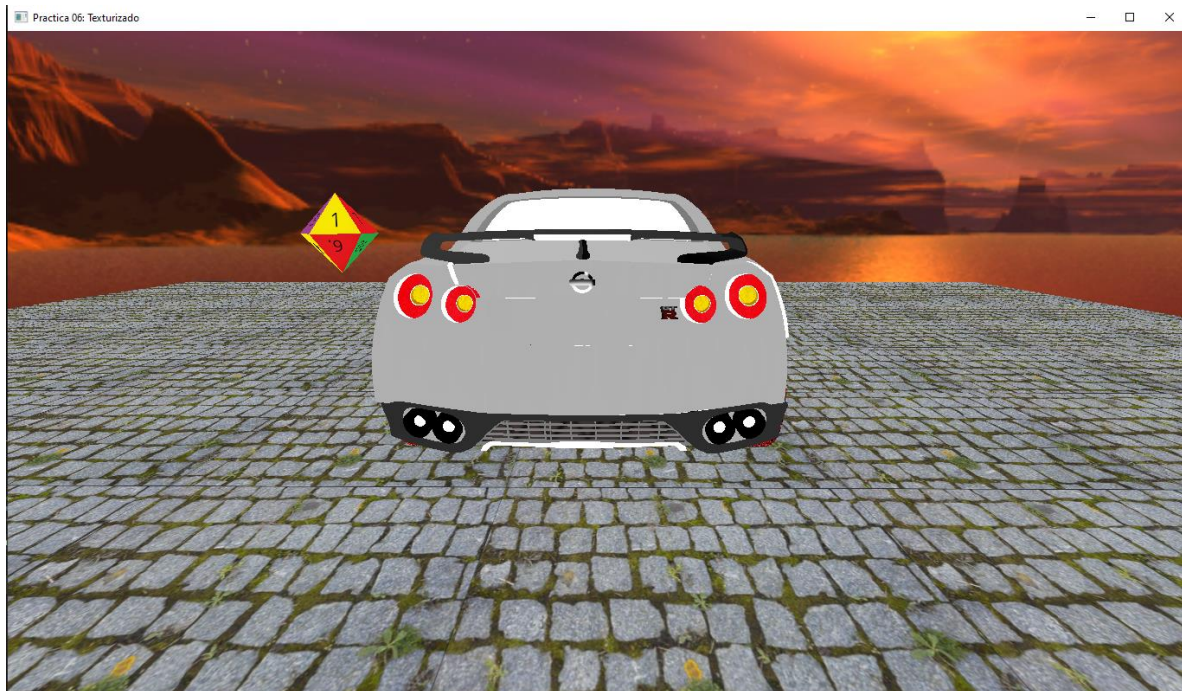
//llanta delantera lado izquierdo
model = modelaux;
model = glm::translate(model, glm::vec3(18.0f, -24.5f, 63.0f));
//model = glm::rotate(model, glm::radians(mainWindow.getarticulacion1()), glm::vec3(-1.0f, 0.0f, 0.0f));*/
color = glm::vec3(1.0f, 1.0f, 1.0f); //llanta color blanco
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
llanta_izq.RenderModel(); // se muestra llanta vista izquierda

```


En este caso quite las articulaciones, porque no eran necesarias para realizar en el ejercicio. Solo puse la jerarquía y no se agrega el cofre, porque exporté todo el modelo junto al cofre.

A continuación, se mostrará ejecuciones del código

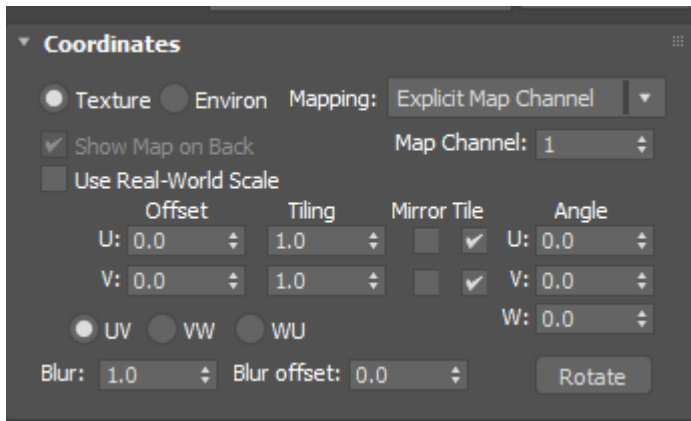




En este ejercicio, me di cuenta que la imagen que proporcione al profesor no fue la mejor, porque los elementos que solicito no se podían seleccionar de una manera correcta, porque salían con elementos que no eran necesarios, por ejemplo, en el cofre no se puede tener el cofre completo (todo el circulo), porque empieza a desfigurarse.

2.- Problemas

Para el ejercicio 3 se me complico bastante, porque se me olvido en un inicio como podía modificar las texturas para que se acomodaran lo más posible a la realidad, y lo que estaba haciendo era modificarlas desde los parámetros que nos dan, se muestra imagen a continuación



Pero estas coordenadas al momento de exportar no salían nada, y me tuve que ver los documentos proporcionados por el profesor para recordar como tenía que modificarlos correctamente. Otro problema que presente fueron las escalas de la figura que proporcione, porque no me salían como yo esperaba al momento de texturizarlas.

3.- Conclusión

Lo que fue el ejercicio de clase y la práctica se me hicieron interesantes de como se texturiza nuestros modelos, obviamente la forma más rápida de texturizar sería mediante 3D Max, porque cuando se texturiza por código debemos de hacer uso de las primitivas, y aun así siento que es mucho trabajo a diferencia de solo usar la herramienta y escalarlo de una manera correcta.