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: Aguilar Pérez José Ramón

Nº de Cuenta: 317515048

GRUPO DE LABORATORIO: 02

GRUPO DE TEORÍA: 04

SEMESTRE 2025-2

FECHA DE ENTREGA LÍMITE: 29/marzo/2025

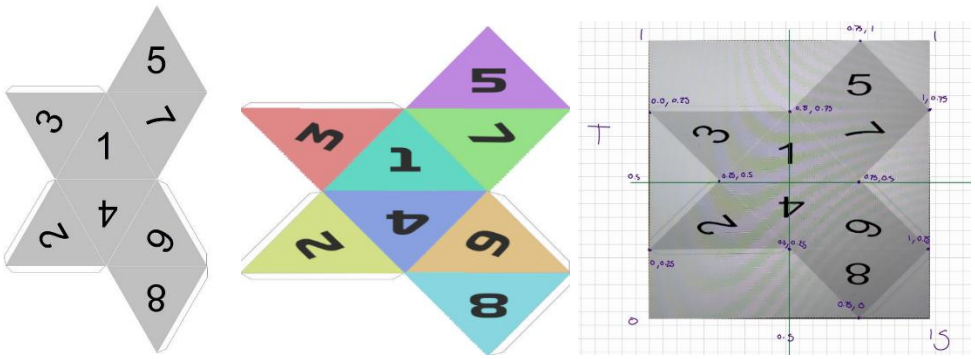
CALIFICACIÓN: _____

REPORTE DE PRÁCTICA:

1.- Ejercicios realizados.

- **Ejercicio 1: Crear un dado de 8 caras y texturizarlo por medio de código.**
- **Ejercicio 2: Importar el modelo de su coche con sus 4 llantas acomodadas y tener texturizadas las 4 llantas (diferenciar caucho y rin)**
- **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 el primer ejercicio, se descargó una plantilla como guía para el dibujo del dado. En GIMP, se agregaron colores y se hicieron los cambios necesarios para que OpenGL pudiera trabajar la imagen (se escaló a 512x512, modo RGB y formato PNG). Ya con la imagen generada, se calcularon los vértices ST necesarios.



Ya que se tenía la imagen del dado, se procedió a establecer una función para generar el octaedro. Partiendo como guía la función *CrearDado(),* se instanció la función *CrearDado8(),* la cuál recibe los 24 índices necesarios para generar el dado. Es importante declarar correctamente los vértices ST para evitar rotaciones no deseadas

```
void CrearDado8()
{
    unsigned int octa_indices[] = {
        // Caras superiores
        //Superior 1
        0, 1, 2,
        //Superior 2
        3, 4, 5,
        //Superior 3
        6, 7, 8,
        //Superior 4
        9, 10, 11,

        // Caras inferiores
        //Inferior 1
        12, 13, 14,
        //Inferior 2
        15, 16, 17,
        //Inferior 3
        18, 19, 20,
        //Inferior 4
        21, 22, 23,
    };
```

```
GLfloat octa_vertices[] = {
    //Parte superior
    //x     y      z      S       T       NX      NY      NZ
    0.5f,  0.0f,  0.5f,    0.73f,  0.51f,    0.0f,   0.0f,   -1.0f,
    -0.5f, 0.0f,  0.5f,    0.27f,  0.51f,    0.0f,   0.0f,   -1.0f, //Frontal
    0.0f,  0.7f,  0.0f,    0.5f,   0.74f,    0.0f,   0.0f,   -1.0f,

    0.5f,  0.0f,  -0.5f,   0.99f,  0.74f,    -1.0f,  0.0f,   0.0f,
    0.5f,  0.0f,  0.5f,    0.75f,  0.51f,    -1.0f,  0.0f,   0.0f, //Derecha
    0.0f,  0.7f,  0.0f,    0.52f,  0.74f,    -1.0f,  0.0f,   0.0f,

    -0.5f, 0.0f,  0.5f,    0.25f,  0.52f,    1.0f,   0.0f,   0.0f,
    -0.5f, 0.0f,  -0.5f,   0.01f,  0.74f,    1.0f,   0.0f,   0.0f, //Izquierda
    0.0f,  0.7f,  0.0f,    0.48f,  0.74f,    1.0f,   0.0f,   0.0f,

    -0.5f, 0.0f,  -0.5f,   0.75f,  0.99f,    0.0f,   0.0f,   1.0f,
    0.5f,  0.0f,  -0.5f,   0.98f,  0.76f,    0.0f,   0.0f,   1.0f, //Trasera
    0.0f,  0.7f,  0.0f,    0.52f,  0.76f,    0.0f,   0.0f,   1.0f,


    // Parte inferior
    //x     y      z      S       T
    0.5f,  0.0f,  0.5f,    0.73f,  0.49f,    0.0f,   0.0f,   -1.0f,
    -0.5f, 0.0f,  0.5f,    0.27f,  0.49f,    0.0f,   0.0f,   -1.0f, //Frontal
    0.0f, - 0.7f,  0.0f,   0.5f,   0.26f,    0.0f,   0.0f,   -1.0f,

    0.5f,  0.0f,  -0.5f,   0.98f,  0.26f,    -1.0f,  0.0f,   0.0f,
    0.5f,  0.0f,  0.5f,    0.75f,  0.49f,    -1.0f,  0.0f,   0.0f, //Derecha
    0.0f,  -0.7f,  0.0f,   0.52f,  0.26f,    -1.0f,  0.0f,   0.0f,

    -0.5f, 0.0f,  0.5f,    0.25f,  0.49f,    1.0f,   0.0f,   0.0f,
    -0.5f, 0.0f,  -0.5f,   0.02f,  0.26f,    1.0f,   0.0f,   0.0f, //Izquierda
    0.0f,  -0.7f,  0.0f,   0.48f,  0.26f,    1.0f,   0.0f,   0.0f,

    -0.5f, 0.0f,  -0.5f,   0.75f,  0.02f,    0.0f,   0.0f,   1.0f,
    0.5f,  0.0f,  -0.5f,   0.98f,  0.23f,    0.0f,   0.0f,   1.0f, //Trasera
    0.0f,  -0.7f,  0.0f,   0.52f,  0.23f,    0.0f,   0.0f,   1.0f,
};

Mesh* dado8 = new Mesh();
dado8->CreateMesh(octa_vertices, octa_indices, 192, 24);
meshList.push_back(dado8);
```

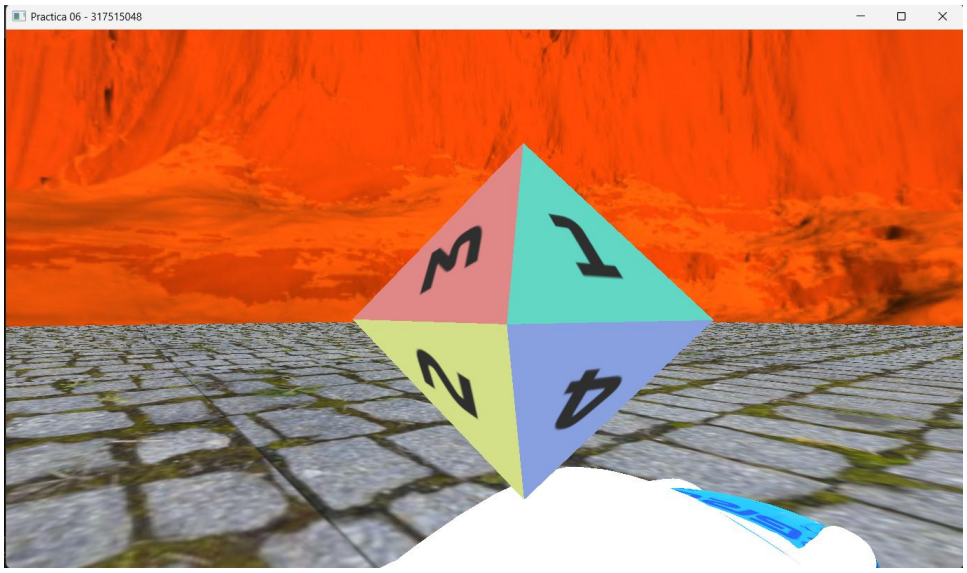
Después, se declara la textura del dado y se instancia dentro del main del programa.

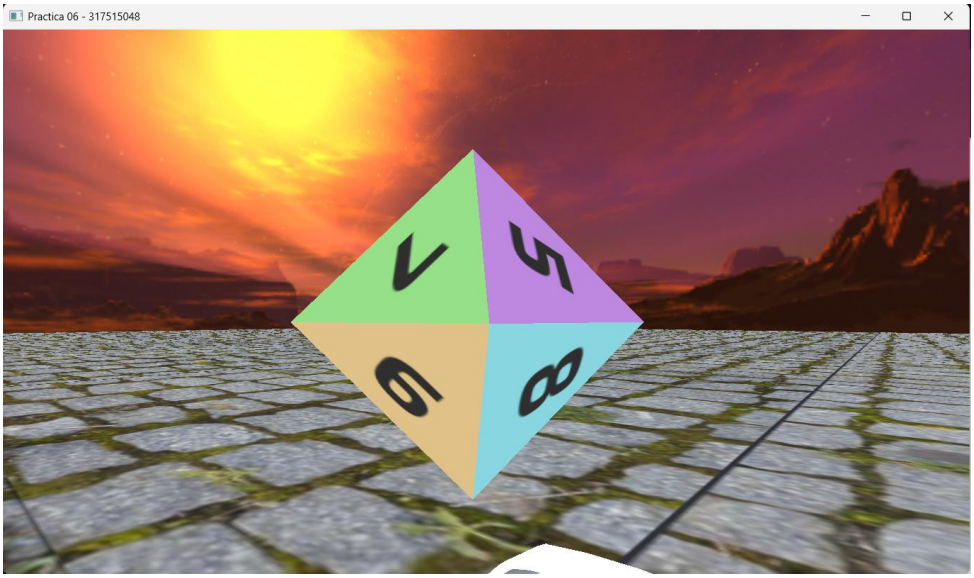
```
Texture dadoTexture;
Texture dado8Texture;
```

```
dado8Texture = Texture("Textures/dado_8_c.png");
dado8Texture.LoadTextureA();
```

Finalmente, se genera el dado con ayuda de *RenderMesh(),* cabe resaltar que el índice de este nuevo dado dentro de la lista *meshList* es el 5.

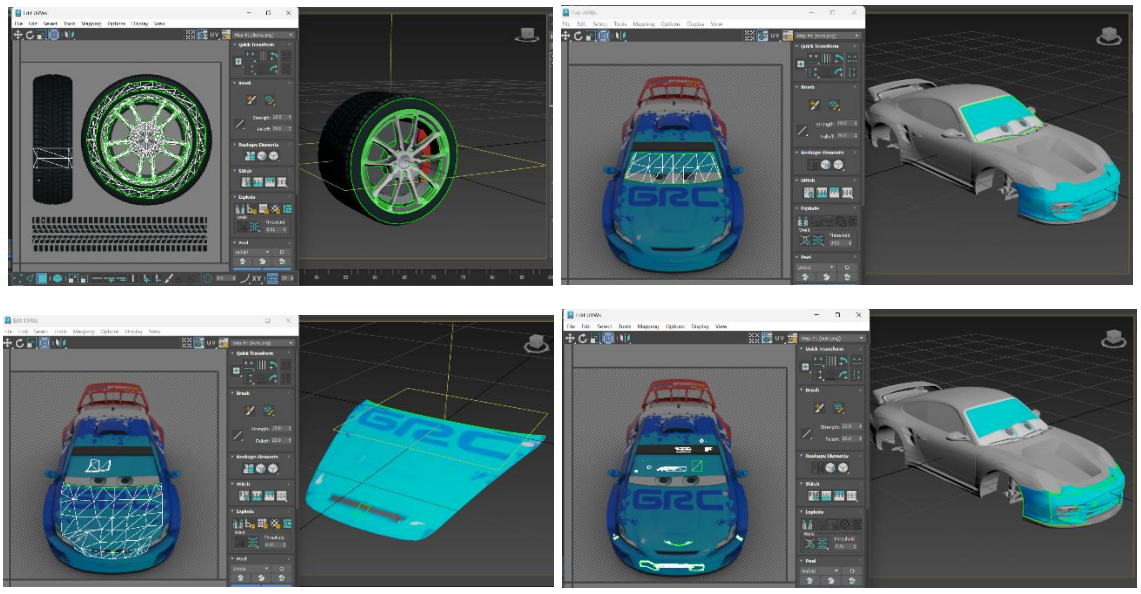
```
/********************* Generando Dado de 8 caras ************************/
//Dado de 8 caras
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-1.5f, 4.5f, -2.0f));
model = glm::scale(model, glm::vec3(3.0f, 3.0f, 3.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
dado8Texture.UseTexture();
meshList[5]->RenderMesh();
```





Para el coche, se trabajó con el modelo del Porsche 911 de la práctica pasada, y el personaje utilizado fue Raoul ÇaRoule de la película Cars 2. Se trabajaron 3 partes distintas del coche: cofre, llantas por separado y el coche en sí. Para cada uno de estos modelos, se texturizó a partir de una captura del personaje.

Dentro de 3dsMax, se trabajaron los modelos por separado y se texturizaron a partir de la imagen anterior. En la parte de la parrilla, debido al ángulo de la foto de referencia se tuvo que improvisar un poco a la hora de texturizar las partes no visibles.



Una vez que se tuvieron los modelos texturizados, se importaron al proyecto y al programa principal.

```
Model Porsche_M;
Model Cofre_M;
Model LlantaTrasDer_M;
Model LlantaTrasIzq_M;
Model LlantaDelDer_M;
Model LlantaDelIzq_M;
```

```
Porsche_M = Model();
Porsche_M.LoadModel("Models/porsche.obj");
Cofre_M = Model();
Cofre_M.LoadModel("Models/cofre.obj");
LlantaTrasDer_M = Model();
LlantaTrasDer_M.LoadModel("Models/llantatrasder.obj");
LlantaTrasIzq_M = Model();
LlantaTrasIzq_M.LoadModel("Models/llantatrasizq.obj");
LlantaDelDer_M = Model();
LlantaDelDer_M.LoadModel("Models/llantadelder.obj");
LlantaDelIzq_M = Model();
LlantaDelIzq_M.LoadModel("Models/llantadelizq.obj");
```

Finalmente, una vez que se importaron los modelos al programa, se instanciaron cada uno de estos por medio de jerarquía.

```
/******************** Generando Porsche con texturas **********************/
//Instancia del coche
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f + mainWindow.getmuevex(), -0.5f, -3.0f));
model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.3f));
model = glm::rotate(model, -90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
modelaux = model;
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Porsche_M.RenderModel();

//Cofre
model = modelaux;
model = translate(model, glm::vec3(-0.28f, 3.22f, -8.8f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Cofre_M.RenderModel(); //Cofre del porsche

//Llanta delantera derecha
model = modelaux;
model = translate(model, glm::vec3(6.8f, -1.4f, -11.0f));
modelaux = model;
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
LlantaDelDer_M.RenderModel();


//Llanta delantera izquierda
model = modelaux;
model = translate(model, glm::vec3(-13.6f, 0.0f, 0.0f));
modelaux = model;
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
LlantaDelIzq_M.RenderModel();

//Llanta trasera izquierda
model = modelaux;
model = translate(model, glm::vec3(0.3f, 0.0f, 22.0f));
modelaux = model;
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
LlantaTrasIzq_M.RenderModel();

//LLanta trasera derecha
model = modelaux;
model = translate(model, glm::vec3(13.3f, 0.0f, 0.0f));
modelaux = model;
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
LlantaTrasDer_M.RenderModel();
```


Practica 06 - 317515048
Practica 06 - 317515048
Practica 06 - 317515048
Practica 06 - 317515048

2.- Problemas presentados

A la hora de asignar el Bitmap, se texturizaba todo el modelo del carro, aunque se tuviera como objetos separados la parrilla y el parabrisas. Una vez texturizadas las partes deseadas, se texturizaron con color básico las partes no deseadas del coche.

3.- Conclusión:

a. Los ejercicios del reporte: Complejidad, Explicación.

La elaboración de esta práctica fue muy interesante de realizar, ya que permitió comprender de mejor manera las diferencias y retos de texturizar por medio de código y texturizar “manualmente” por medio de 3dsMax. La complejidad dependía bastante del modelo de auto elegido, por ejemplo, en el Porsche las ventanas tenían doble capa por alguna razón y había partes empotradas de la carrocería dentro de la parrilla.

b. Comentarios generales: Faltó explicar a detalle, ir más lento en alguna explicación, otros comentarios y sugerencias para mejorar desarrollo de la práctica

La explicación de la asignación de vértices ST, así como la texturización dentro de 3dsMax fue clara y precisa. Además, fue de gran ayuda el manual proporcionado para el proceso de texturizado dentro de 3dsMax.

c. Conclusión

Esta práctica me permitió comprender mejor cómo texturizar objetos tanto en OpenGL como en 3dsMax, mientras que por código se necesitan los vértices ST de la imagen, en 3dsMax es más intuitivo, pero hay que tener cuidado con los triángulos del modelo, ya que, si se manipulan mucho, la imagen podría quedar distorsionada/estirada. La realización de esta practica fue muy interesante ya que me permitió entender como poner en práctica los conocimientos de texturizado para el proyecto final.

1. Bibliografía en formato APA

- 3ds Max Quick Start Guide. (s/f). Autodesk.com. Recuperado el 20 de marzo de 2025, de https://www.autodesk.com/learn/ondemand/curated/3ds-max-quick-start-guide