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 6

NOMBRE COMPLETO: Araiza Valdés Diego Antonio

Nº de Cuenta: 423032833

GRUPO DE LABORATORIO: 02

GRUPO DE TEORÍA: 06

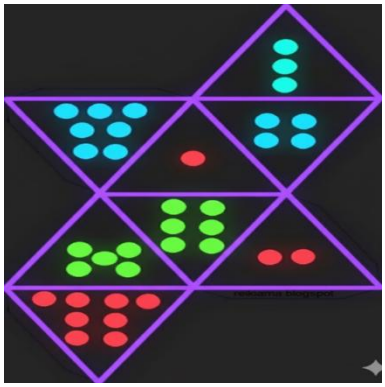
SEMESTRE: 2026-1

FECHA DE ENTREGA LÍMITE: 12/10/2025

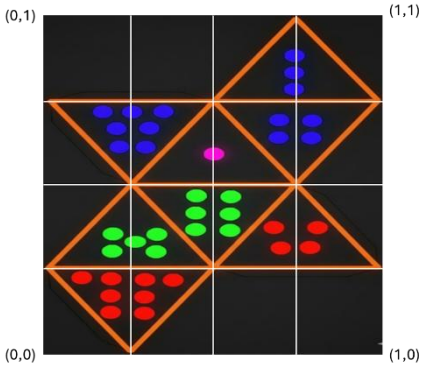
CALIFICACIÓN: _____

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.
   1.1. Crear un dado de 8 caras y texturizarlo por medio de código.



Primero le pedimos a Gemini que nos diera un cubo de 8 caras con colores neones y con ayuda de Gimp lo optimizamos de manera que midiera 512 x 512 px.




```
Texture dado8Texture;
dado8Texture = Texture("Textures/Dado80.png");
dado8Texture.LoadTextureA();
```

Luego, cuadriculamos la imagen, y con ayuda de eso sacamos las coordenadas de cada vértice para texturizar el dado. Y la importamos al código.

```
void CrearDado2()
{
    unsigned int cubo_indices2[] = {
        // cara 0
        0, 1, 2,
        // cara 1
        3, 4, 5,
        // cara 2
        6, 7, 8,
        // cara 3
        9, 10, 11,
        // cara 4
        12, 13, 14,
        // cara 5
        15, 16, 17,
        // cara 6
        18, 19, 20,
        // cara 7
        21, 22, 23

    };
// average normals
    GLfloat cubo_vertices2[] = {
    // face 0 (top, front, right)
    // x      y      z      S     T        NX        NY        NZ
     0.0f,  0.5f,  0.0f,   0.25f, 0.50f,   0.577350f,  0.577350f,  0.577350f, //0
     0.0f,  0.0f,  0.5f,   0.50f, 0.75f,   0.577350f,  0.577350f,  0.577350f, //1
     0.5f,  0.0f,  0.0f,   0.00f, 0.75f,   0.577350f,  0.577350f,  0.577350f, //2

    // face 1 (top, right, back)
     0.0f,  0.5f,  0.0f,   0.75f, 0.50f,   0.577350f,  0.577350f, -0.577350f, //3
     0.5f,  0.0f,  0.0f,   0.50f, 0.25f,   0.577350f,  0.577350f, -0.577350f, //4
     0.0f,  0.0f, -0.5f,   1.00f, 0.25f,   0.577350f,  0.577350f, -0.577350f, //5

    // face 2 (top, back, left)
     0.0f,  0.5f,  0.0f,   0.75f, 1.00f,  -0.577350f,  0.577350f, -0.577350f, //6
     0.0f,  0.0f, -0.5f,   0.50f, 0.75f,  -0.577350f,  0.577350f, -0.577350f, //7
    -0.5f,  0.0f,  0.0f,   1.00f, 0.75f,  -0.577350f,  0.577350f, -0.577350f, //8

    // face 3 (top, left, front)
     0.0f,  0.5f,  0.0f,   0.50f, 0.25f,  -0.577350f,  0.577350f,  0.577350f, //9
    -0.5f,  0.0f,  0.0f,   0.75f, 0.50f,  -0.577350f,  0.577350f,  0.577350f, //10
     0.0f,  0.0f,  0.5f,   0.25f, 0.50f,  -0.577350f,  0.577350f,  0.577350f, //11

    // face 4 (bottom, right, front)
     0.0f, -0.5f,  0.0f,   0.75f, 0.50f,   0.577350f, -0.577350f,  0.577350f, //12
     0.5f,  0.0f,  0.0f,   1.00f, 0.75f,   0.577350f, -0.577350f,  0.577350f, //13
     0.0f,  0.0f,  0.5f,   0.50f, 0.75f,   0.577350f, -0.577350f,  0.577350f, //14

    // face 5 (bottom, back, right)
     0.0f, -0.5f,  0.0f,   0.25f, 0.50f,   0.577350f, -0.577350f, -0.577350f, //15
     0.0f,  0.0f, -0.5f,   0.00f, 0.25f,   0.577350f, -0.577350f, -0.577350f, //16
     0.5f,  0.0f,  0.0f,   0.50f, 0.25f,   0.577350f, -0.577350f, -0.577350f, //17

    // face 6 (bottom, left, back)
     0.0f, -0.5f,  0.0f,   0.25f, 0.00f,  -0.577350f, -0.577350f, -0.577350f, //18
    -0.5f,  0.0f,  0.0f,   0.50f, 0.25f,  -0.577350f, -0.577350f, -0.577350f, //19
     0.0f,  0.0f, -0.5f,   0.00f, 0.25f,  -0.577350f, -0.577350f, -0.577350f, //20
```

```
    // face 7 (bottom, front, left)
    0.0f, -0.5f,  0.0f,   0.50f, 0.75f,  -0.577350f, -0.577350f,  0.577350f, //21
    0.0f,  0.0f,  0.5f,   0.25f, 0.50f,  -0.577350f, -0.577350f,  0.577350f, //22
    -0.5f,  0.0f,  0.0f,   0.75f, 0.50f,  -0.577350f, -0.577350f,  0.577350f, //23
    };

    Mesh* dado2 = new Mesh();
    dado2->CreateMesh(cubo_vertices2, cubo_indices2, 192, 24);
    meshList.push_back(dado2);

}
```

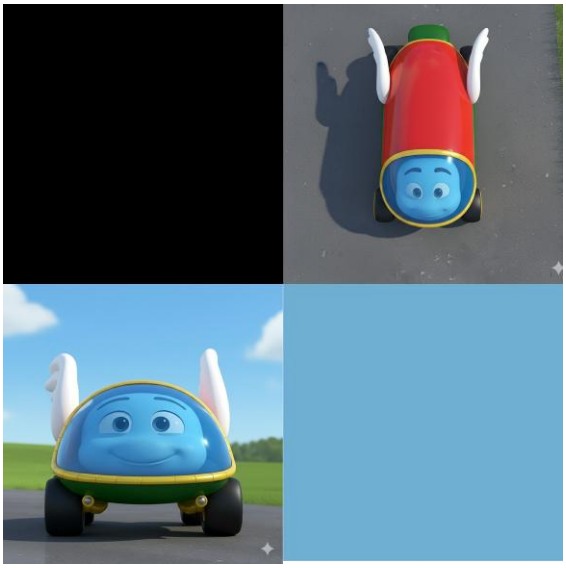
```
CrearDado2();
```

Posteriormente, creamos la función CrearDado2, para declarar el dado de 8 caras y ahí sustituimos los valores de S y T que calculamos anteriormente. Posteriormente la declaramos dentro del main.

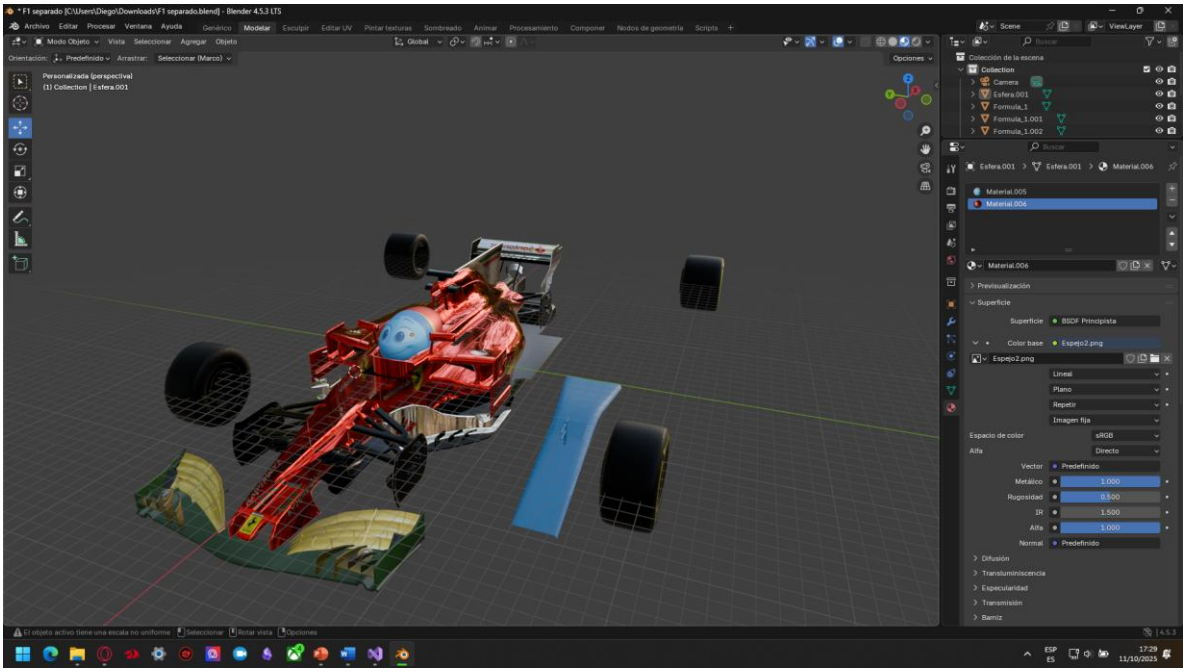
```
//DADO
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-3.0f, 4.5f, -8.0f));
model = glm::scale(model, glm::vec3(3.0f, 3.0f, 3.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
dado8Texture.UseTexture();
meshList[5]->RenderMesh();
```

Luego lo declaramos y dibujamos dentro de OpenGL.

1.2. *Importar el modelo de su coche con sus 4 llantas acomodadas y tener texturizadas las 4 llantas (diferenciar caucho y rin)*
1.3. *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*



Primero obtuvimos y optimizamos las imágenes que utilizaríamos en Blender para texturizar el modelo y las colocamos en la carpeta texture.



Luego, dentro del modelo que teníamos separado, acomodamos cada una de las texturas obtenidas previamente, además hicimos uso de las texturas que contenía el auto. Para esto se separó el ala delantera y se texturizó individualmente, además se creo una esfera la cual transformamos y texturizamos de igual forma aparte. Por último, exportamos los objetos individualmente.

```
Model F1Cuerpo_M;
Model F1Cofre_M;
Model LlantaDI_M;
Model LlantaDD_M;
Model LlantaTI_M;
Model LlantaTD_M;
F1Cuerpo_M = Model();
F1Cuerpo_M.LoadModel("Models/F1Cuerpo.obj");
F1Cofre_M = Model();
F1Cofre_M.LoadModel("Models/F1Cofre.obj");
LlantaDI_M = Model();
LlantaDI_M.LoadModel("Models/LlantaDI.obj");
LlantaDD_M = Model();
LlantaDD_M.LoadModel("Models/LlantaDD.obj");
LlantaTI_M = Model();
LlantaTI_M.LoadModel("Models/LlantaTI.obj");
LlantaTD_M = Model();
LlantaTD_M.LoadModel("Models/LlantaTD.obj");
```

Posteriormente, declaramos los modelos y los cargamos en el código

```
//COFRE
glEnable(GL_BLEND);
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
model = modelaux;
model = glm::translate(model, glm::vec3(5.5846875f, 1.8315f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
F1Cofre_M.RenderModel();

//LLANTA DELANTERA IZQUIERDA
model = modelaux;
model = glm::translate(model, glm::vec3(11.175f, -0.6375f, -5.7375f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
LlantaDI_M.RenderModel();

//LLANTA DELANTERA DERECHA
model = modelaux;
model = glm::translate(model, glm::vec3(11.175f, -0.7175f, 5.7375f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
LlantaDD_M.RenderModel();

//LLANTA DELANTERA IZQUIERDA
model = modelaux;
model = glm::translate(model, glm::vec3(-13.65, -0.6795f, -5.325f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
LlantaTI_M.RenderModel();

//LLANTA DELANTERA DERECHA
model = modelaux;
model = glm::translate(model, glm::vec3(-13.65f, -0.7005f, 5.4375f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
LlantaTD_M.RenderModel();
```

Finalmente, reutilizamos el código de la práctica anterior, declaramos y dibujamos el auto, eliminando las partes de las articulaciones y activamos el canal Alpha para la transparencia del cofre.

Evidencias:

1.3.1. Ejercicio 1:

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
   R: Si, fue bastante complicado texturizar la esfera, y no fue posible exportar la textura metalizada.

3. Conclusión:
   1. Los ejercicios del reporte: Complejidad, Explicación.
      Respecto a los ejercicios realizados, creo que fueron bastante difíciles con respecto al ejercicio en clase, si bien no fueron tan complicados, el salto de nivel fue bastante.

   2. Comentarios generales: Faltó explicar a detalle, ir más lento en alguna explicación, otros comentarios y sugerencias para mejorar desarrollo de la práctica.
      Respecto a la explicación no tengo mucho que decir, respecto al ejercicio fue bastante adecuada, pero para la práctica se quedó algo corta, me hubiera gustado saber otras cosas como el cómo cambiar la textura de una parte sin que se modifique la de las otras.

3. Conclusión

   Personalmente creo que se cumplieron los objetivos de la práctica. Con el ejercicio y la práctica aprendí como texturizar objetos desde OpenGL, y desde Blender. Con esto aprendí a hacer uso de herramientas como Gimp para optimizar imágenes, aprendí a obtener las coordenadas de S y T para acomodar las texturas desde OpenGL y aprendí a colocar/cambiar/eliminar/separar/acomodar las texturas desde Blender, permitiéndome exportar los objetos ya texturizados. Además, aprendí a importar una esfera, suavizarla y me di una idea de lo complicado que es texturizarla.