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: Mino Guzmán Yara Amairani

Nº de Cuenta: 422017028

GRUPO DE LABORATORIO: 03

GRUPO DE TEORÍA: 04

SEMESTRE 2025-2

FECHA DE ENTREGA LÍMITE: 29 marzo 2025

CALIFICACIÓN: _____

REPORTE DE PRÁCTICA:

1.- Actividades:

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

```
Texture brickTexture;
Texture dirtTexture;
Texture plainTexture;
Texture pisoTexture;
Texture dadoTexture;
Texture logofiTexture;
Texture octaedroTexture;
```

Definí la textura para el octaedro.

```
void CrearOctaedro()
{
    unsigned int octaedro_indices[] = {
        // Parte superior
        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      // Cara 8

    };
```

```
GLfloat octaedro_vertices[] = {
    // Cara 1: top - left - front   normal: (-1, 1, 1)
    //x     y       z       S    T      NX      NY      NZ
    0.0f,  1.0f,  0.0f,  0.14f, 0.65f,  -1.0f,  1.0f,  1.0f,
    -1.0f, 0.0f,  0.0f,  0.02f, 0.35f,  -1.0f,  1.0f,  1.0f,
    0.0f,  0.0f,  1.0f,  0.26f, 0.35f,  -1.0f,  1.0f,  1.0f,

    // Cara 2: top - front - right   normal: (1, 1, 1)
    //x     y       z       S   T       NX      NY      NZ
    0.0f,  1.0f,  0.0f,  0.28f, 0.35f,   1.0f,  1.0f,  1.0f,
    0.0f,  0.0f,  1.0f,  0.40f, 0.65f,   1.0f,  1.0f,  1.0f,
    1.0f,  0.0f,  0.0f,  0.16f, 0.65f,   1.0f,  1.0f,  1.0f,

    // Cara 3: top - right - back   normal: (1, 1, -1)
    //x     y       z       S   T       NX      NY      NZ
    0.0f,  1.0f,  0.0f,  0.43f, 0.65f,   1.0f,  1.0f, -1.0f,
    1.0f,  0.0f,  0.0f,  0.3f, 0.35f,   1.0f,  1.0f, -1.0f,
    0.0f,  0.0f, -1.0f,  0.55f, 0.35f,   1.0f,  1.0f, -1.0f,
```

```
// Cara 4: top - back - left   normal: (-1, 1, -1)
//x     y       z       S    T       NX      NY      NZ
0.0f,  1.0f,  0.0f,  0.57f, 0.35f,  -1.0f,  1.0f, -1.0f,
0.0f,  0.0f, -1.0f,  0.65f, 0.65f,  -1.0f,  1.0f, -1.0f,
-1.0f, 0.0f,  0.0f,  0.45f, 0.65f,  -1.0f,  1.0f, -1.0f,

// Cara 5: bottom - front - left   normal: (-1, -1, 1)
//x     y       z       S    T       NX      NY      NZ
0.0f, -1.0f,  0.0f,  0.71f, 0.65f,  -1.0f, -1.0f,  1.0f,
0.0f,  0.0f,  1.0f,  0.61f, 0.35f,  -1.0f, -1.0f,  1.0f,
-1.0f, 0.0f,  0.0f,  0.81f, 0.35f,  -1.0f, -1.0f,  1.0f,

// Cara 6: bottom - right - front   normal: (1, -1, 1)
//x     y       z       S    T       NX      NY      NZ
0.0f, -1.0f,  0.0f,  0.86f, 0.35f,   1.0f, -1.0f,  1.0f,
1.0f,  0.0f,  0.0f,  0.95f, 0.65f,   1.0f, -1.0f,  1.0f,
0.0f,  0.0f,  1.0f,  0.79f, 0.65f,   1.0f, -1.0f,  1.0f,
```

```
// Cara 7: bottom - back - right   normal: (1, -1, -1)
//x     y       z       S   T       NX      NY      NZ
0.0f, -1.0f,  0.0f,  0.57f, 0.95f,   1.0f, -1.0f, -1.0f,
0.0f,  0.0f, -1.0f,  0.67f, 0.69f,   1.0f, -1.0f, -1.0f,
1.0f,  0.0f,  0.0f,  0.45f, 0.69f,   1.0f, -1.0f, -1.0f,

// Cara 8: bottom - left - back   normal: (-1, -1, -1)
//x     y       z       S   T       NX      NY      NZ
0.0f, -1.0f,  0.0f,  0.43f, 0.01f,  -1.0f, -1.0f, -1.0f,
-1.0f, 0.0f,  0.0f,  0.3f, 0.30f,  -1.0f, -1.0f, -1.0f,
0.0f,  0.0f, -1.0f,  0.55f, 0.30f,  -1.0f, -1.0f, -1.0f
```

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

Se creó la función void CrearOctaedro() para poder crear la figura; posteriormente, se definieron medidas en S y T de acuerdo con el número que quería que estuviera en cada cara. La imagen utilizada fue la siguiente:

```
brickTexture = Texture("Textures/brick.png");
brickTexture.LoadTextureA();
dirtTexture = Texture("Textures/dirt.png");
dirtTexture.LoadTextureA();
plainTexture = Texture("Textures/plain.png");
plainTexture.LoadTextureA();
pisoTexture = Texture("Textures/piso.tga");
pisoTexture.LoadTextureA();
dadoTexture = Texture("Textures/dado_animales_.png");
dadoTexture.LoadTextureA();
logofiTexture = Texture("Textures/escudo_fi_color.tga");
logofiTexture.LoadTextureA();
octaedroTexture = ("Textures/octaedro_numeros.png");
octaedroTexture.LoadTextureA();
```

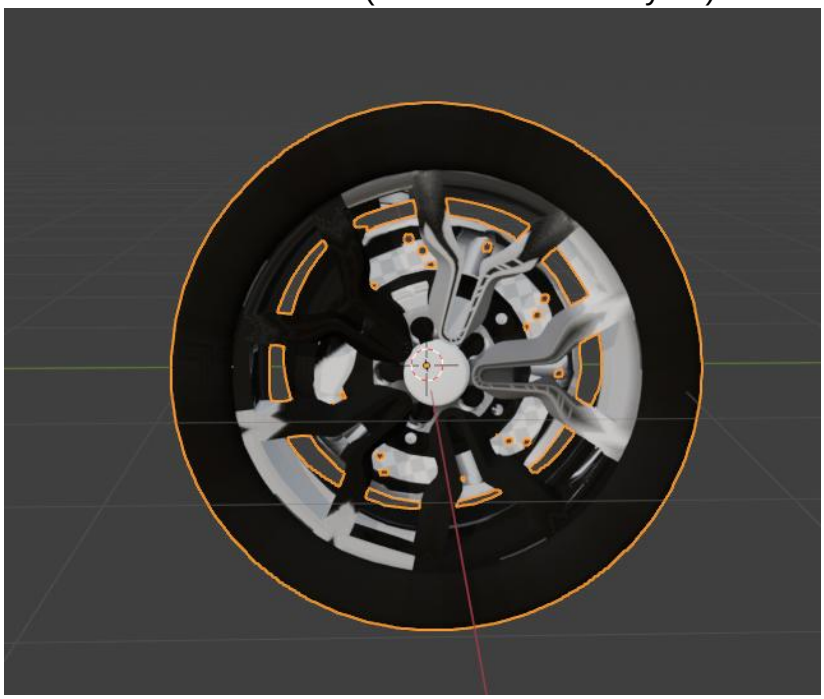
Cargué la imagen que se utilizó para texturizar el octaedro.

```
//Reporte de practica
//Ejercicio 1: Crear un dado de 8 caras y texturizarlo por medio de código
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, 4.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
octaedroTexture.UseTexture();
meshList[5]->RenderMesh();
```

Esta parte del código muestra el octaedro en pantalla, con esta pequeña parte yo fue guiándome para que cada número se viera en cada cara de mi octaedro.

Ejecución:

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



Utilizando blender, se texturizaron las llantas, debido a que no me permitía separar el rin ya que el modelo de auto que yo escogí lo tenía pegado, tuve que texturizarlo todo junto. La imagen utilizada para realizar la texturización fue la siguiente:

```
Model LlantaDD_M;
Model LlantaTD_M;
Model LlantaDI_M;
Model LlantaTI_M;
```

Definí el modelo para cada llanta.

```
LlantaDD_M = Model();
LlantaDD_M.LoadModel("Models/LlantaDDtext.obj");
LlantaTD_M = Model();
LlantaTD_M.LoadModel("Models/LlantaTDtext.obj");
LlantaDI_M = Model();
LlantaDI_M.LoadModel("Models/LlantaDItext.ob");
LlantaTI_M = Model();
LlantaTI_M.LoadModel("Models/LlantaTItext.obj");
```

Cargué los archivos .obj utilizados para cada llanta.

```
//Ejercicio 2 y 3:  Importar el modelo de su coche con sus 4 llantas acomodadas y tener texturizadas l
// Texturizar la cara del personaje de la imagen tipo cars en el espejo (ojos) y detalles en cofre y p
//Instancia del coche
//Auto
//color = glm::vec3(0.0f, 0.0f, 0.5f);
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, -1.7f, 0.0f));
model = glm::translate(model, glm::vec3(0.0f, 0.0f, mainWindow.getmueveCarro()));
modelaux = model; //Se guarda la transformación del cuerpo
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Auto_M.RenderModel();

//LLANTA DELANTERA DERECHA
model = modelaux;
color = glm::vec3(0.0f, 0.0f, 0.0f); //Llanta de color negro
model = glm::translate(model, glm::vec3(-2.3f, 1.0f, 3.6f));
model = glm::rotate(model, glm::radians(mainWindow.getrotaLlantas()), glm::vec3(1.0f, 0.0f, 0.0f));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
LlantaDD_M.RenderModel();       //Muestra Llanta
```

Lo primero que se muestra es el auto, ya que es de donde se partió para realizar la jerarquización; posteriormente, se están mostrando las llantas.

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

Lo único que se texturizó fue el parabrisas y la parrilla.

```
Model Auto_M;
```

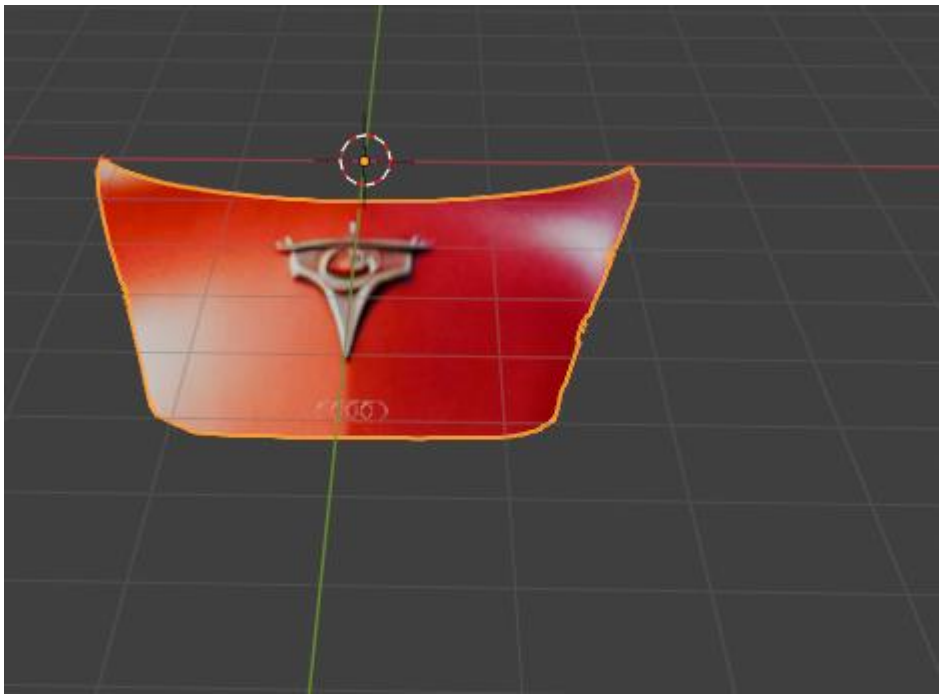
Definí el modelo del auto.

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

Cargué el archivo.obj que se utilizó.

```
//Auto
//color = glm::vec3(0.0f, 0.0f, 0.5f);
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, -1.7f, 0.0f));
model = glm::translate(model, glm::vec3(0.0f, 0.0f, mainWindow.getmueveCarro()));
modelaux = model; //Se guarda la transformación del cuerpo
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Auto_M.RenderModel();
```

El auto fue lo primero que cargué, ya que de aquí inicié mi jerarquización.



Para el cofre, también seleccioné una parte del auto de cars que se utilizó y lo texturicé.

```
Model Cofre_M;
```

Definí el modelo del cofre.

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

Cargué el archivo .obj que se utilizó.

```
//COFRE
model = modelaux;
color = glm::vec3(1.0f, 0.0f, 0.0f); //Cofre de color verde oscuro
model = glm::translate(model, glm::vec3(0.01f, 2.5f, 2.62f));
model = glm::rotate(model, glm::radians(mainWindow.getrotaCofre()), glm::vec3(1.0f, 0.0f, 0.0f));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Cofre_M.RenderModel();      //Muestra cofre
glUseProgram(0);
```

Lo último que jerarquicé fue el cofre, el resultado final se mostrará a continuación.

Ejecución:

3.- Conclusión:

a. Los ejercicios del reporte fueron los adecuados para poder comprender el uso de gimp para el caso del octaedro y seguir mejorando en blender para la texturización.
b. Las explicaciones fueron claras y, gracias a eso, se realizó con éxito la práctica; aunque se debería de explicar también el uso de blender en clase.
c. En conclusión, la práctica me ayudó a explorar nuevas cosas de la computación gráfica, también me ayudó a entender mejor el uso de blender y gimp y a comprender mejor la jerarquización.

Bibliografía

- learnopengl.com. (s.f.). Transformations. LearnOpenGL. https://learnopengl.com/Getting-started/Transformations
- OpenGL. (s.f.). OpenGL Programming Guide (The Red Book). Khronos Group. https://www.khronos.org/registry/OpenGL-Refpages/gl4/
- TutorialsPoint. (s.f.). OpenGL - Transformations. https://www.tutorialspoint.com/opengl/opengl_transformations.htm
- Scratchapixel. (s.f.). Understanding 3D Transformations. https://www.scratchapixel.com/lessons/mathematics-physics-for-computer-graphics/geometry
- GeeksforGeeks. (2021). Translation in Computer Graphics. https://www.geeksforgeeks.org/translation-in-computer-graphics/