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: García Soto Jean Carlo

Nº de Cuenta: 319226304

GRUPO DE LABORATORIO: 03

GRUPO DE TEORÍA: 04

SEMESTRE 2025-2

FECHA DE ENTREGA LÍMITE: 29/03/2025

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.

Para la creación del octaedro, se necesitó primero crear una función la cual nos creara esta figura, se usó como base el código del cubo, dándole 8 parámetros por cada vértice y 3 coordenadas por triangulo, modificamos de acuerdo con lo anterior la cantidad de parámetros a recibir y la cantidad de vértices. Y es importante importar la imagen para la textura, ya que esta se realizará a código.

```
38
39      Texture brickTexture;
40      Texture dirtTexture;
41      Texture plainTexture;
42      Texture pisoTexture;
43      Texture dadoTexture;
44      Texture logofiTexture;
45      Texture octaedroTexture;
```

```
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,    //Punta
        -1.0f, 0.0f,  0.0f,  0.02f, 0.34f,  -1.0f,  1.0f,  1.0f,    //Izquierda
        0.0f,  0.0f,  1.0f,  0.26f, 0.34f,  -1.0f,  1.0f,  1.0f,    //Derecha

        // Cara 2: top - front - right   normal: (1, 1, 1)
        //x     y       z       S   T       NX      NY      NZ
        0.0f,  1.0f,  0.0f,  0.285f,0.35f,   1.0f,  1.0f,  1.0f,
        0.0f,  0.0f,  1.0f,  0.4f,  0.65f,   1.0f,  1.0f,  1.0f,
        1.0f,  0.0f,  0.0f,  0.16f, 0.65f,   1.0f,  1.0f,  1.0f,
```

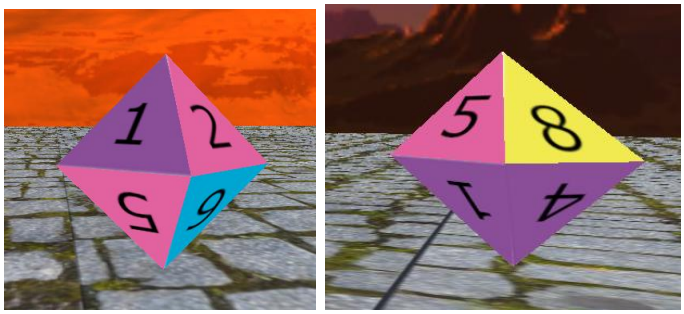
```
289
290     // Cara 3: top - right - back   normal: (1, 1, -1)
291     //x     y       z       S   T       NX      NY      NZ
292     0.0f,  1.0f,  0.0f,  0.43f, 0.65f,   1.0f,  1.0f, -1.0f,
293     1.0f,  0.0f,  0.0f,  0.3f,  0.34f,   1.0f,  1.0f, -1.0f,
294     0.0f,  0.0f, -1.0f,  0.55f, 0.34f,   1.0f,  1.0f, -1.0f,
295
296     // Cara 4: top - back - left   normal: (-1, 1, -1)
297     //x     y       z       S   T       NX      NY      NZ
298     0.0f,  1.0f,  0.0f,  0.573f,0.35f,  -1.0f,  1.0f, -1.0f,
299     0.0f,  0.0f, -1.0f,  0.68f, 0.65f,  -1.0f,  1.0f, -1.0f,
300     -1.0f, 0.0f,  0.0f,  0.45f, 0.65f,  -1.0f,  1.0f, -1.0f,
301
302     // Cara 5: bottom - front - left   normal: (-1, -1, 1)
303     //x     y       z       S   T       NX      NY      NZ
304     0.0f, -1.0f,  0.0f,  0.715f,0.64f,  -1.0f, -1.0f,  1.0f,
305     0.0f,  0.0f,  1.0f,  0.6f, 0.35f,  -1.0f, -1.0f,  1.0f,
306     -1.0f, 0.0f,  0.0f,  0.84f, 0.35f,  -1.0f, -1.0f,  1.0f,
307
308     // Cara 6: bottom - right - front   normal: (1, -1, 1)
309     //x     y       z       S   T       NX      NY      NZ
310     0.0f, -1.0f,  0.0f,  0.86f, 0.35f,   1.0f, -1.0f,  1.0f,
311     1.0f,  0.0f,  0.0f,  0.98f,0.64f,   1.0f, -1.0f,  1.0f,
312     0.0f,  0.0f,  1.0f,  0.75f,0.64f,   1.0f, -1.0f,  1.0f,
313
314     // Cara 7: bottom - back - right   normal: (1, -1, -1)
315     //x     y       z       S   T       NX      NY      NZ
316     0.0f, -1.0f,  0.0f,  0.57f, 0.99f,   1.0f, -1.0f, -1.0f,
317     0.0f,  0.0f, -1.0f,  0.45f, 0.68f,   1.0f, -1.0f, -1.0f,
318     1.0f,  0.0f,  0.0f,  0.68f, 0.68f,   1.0f, -1.0f, -1.0f,
```

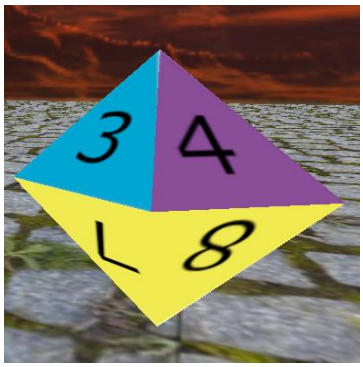
```
319
320         // Cara 8: bottom - left - back   normal: (-1, -1, -1)
321         //x     y       z       S   T       NX      NY      NZ
322         0.0f, -1.0f,  0.0f,  0.44f, 0.02f,  -1.0f, -1.0f, -1.0f,
323         -1.0f, 0.0f,  0.0f,  0.55f, 0.3f,  -1.0f, -1.0f, -1.0f,
324         0.0f,  0.0f, -1.0f,  0.31f, 0.3f,  -1.0f, -1.0f, -1.0f
325
326     };
327     Mesh* octaedro = new Mesh();
328     octaedro->CreateMesh(octaedro_vertices, octaedro_indices, 192, 24);
329     meshList.push_back(octaedro);
330 };
```

```
470     //Dado de Opengl
471     //Ejercicio 1: Crear un dado de 8 caras y texturizarlo por medio de código
472     model = glm::mat4(1.0);
473     model = glm::translate(model, glm::vec3(0.0f, 5.0f, -5.0f));
474     model = glm::rotate(model, glm::radians(mainWindow.getrotaz()), glm::vec3(0.0f, 0.0f, 1.0f));   //Rotar con T
475     model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f));   //Rotar con R
476     model = glm::rotate(model, glm::radians(mainWindow.getrotax()), glm::vec3(1.0f, 0.0f, 0.0f));   //Rotar con E
477     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
478     octaedroTexture.UseTexture();
479     meshList[5]->RenderMesh();
```

Después con el software de GIMP se modificó la imagen que va a ser nuestra textura, se orientó a una forma en la que fuera fácil de manipular las coordenadas de los pixeles, y también pusimos los números en la figura, orientándolos de acuerdo en donde está la base de cada triangulo, para que a la hora de texturizarlo en OpenGL fuera más fácil manipularlo.

Se hizo que los números de la parte inferior de octaedro estuvieran de cabeza, cuidando que a la hora de mirarlos en la orientación correcta los números no estuvieran espejeados.





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

En estos dos ejercicios se hizo el texturizado por medio de blender, dado que se reutilizó el modelo del carro de la practica pasada, abrimos los modelos .obj y texturizamos cada cosa que se nos pedía.

Ya en código solo era cosa de importar y ubicar nuestros modelos ya texturizados. Se le agregaron de nuevo las rotaciones que ya se tenían de la práctica pasada y al mismo tiempo se respetó la jerarquización.

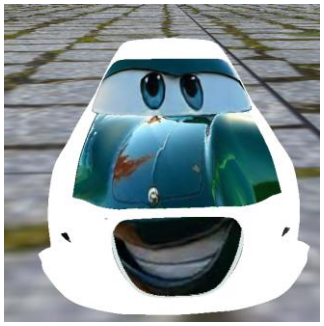
```
51      //Modelos para ejercicio de la practica
52      Model KitCarro_M;   //ojos y carro
53      Model LlantaIzqDelantera_M;
54      Model LlantaDerDelantera_M;
55      Model LlantaIzqTrasera_M;
56      Model LlantaDerTrasera_M;
57      Model Rin_M;
58      Model Cofre_M;
59      Model Parrilla_M;
```

```
368
369     KitCarro_M = Model();
370     KitCarro_M.LoadModel("Models/Carro_Ojos.obj");
371     LlantaIzqDelantera_M = Model();
372     LlantaIzqDelantera_M.LoadModel("Models/LlantaIzqDelantera.obj");
373     LlantaIzqTrasera_M = Model();
374     LlantaIzqTrasera_M.LoadModel("Models/LlantaIzqTrasera.obj");
375     LlantaDerDelantera_M = Model();
376     LlantaDerDelantera_M.LoadModel("Models/LlantaDerDelantera.obj");
377     LlantaDerTrasera_M = Model();
378     LlantaDerTrasera_M.LoadModel("Models/LlantaDerTrasera.obj");
379     Rin_M = Model();
380     Rin_M.LoadModel("Models/Rin.obj");
381     Cofre_M = Model();
382     Cofre_M.LoadModel("Models/Cofre.obj");
383     Parrilla_M = Model();
384     Parrilla_M.LoadModel("Models/Parrilla.obj");
```

```
//Dado de Opengl
//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, 5.0f, -5.0f));
model = glm::rotate(model, glm::radians(mainWindow.getrotaz()), glm::vec3(0.0f, 0.0f, 1.0f));   //Rotar con T
model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f));   //Rotar con R
model = glm::rotate(model, glm::radians(mainWindow.getrotax()), glm::vec3(1.0f, 0.0f, 0.0f));   //Rotar con E
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
octaedroTexture.UseTexture();
meshList[5]->RenderMesh();

//Ejercicio 2
//Carro sin llantas y cofre
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, -1.7f, -5.0f));
model = glm::translate(model, glm::vec3(mainWindow.getTransladaCarro(), 0.0f, 0.0f));   // Mover(transladar) con 1 y 2
modelaux = model;
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
KitCarro_M.RenderModel();       //Muestra carro y ojos

//Llanta derecha delantera
model = modelaux;
model = glm::translate(model, glm::vec3(-3.1f, 0.4f, -1.65f));
model = glm::rotate(model, glm::radians(mainWindow.getrotaLlantas()), glm::vec3(0.0f, 0.0f, 1.0f)); //Rotar con Z y X
modelaux2 = model;
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
LlantaDerDelantera_M.RenderModel();     //Muestra Llanta

//Rin derecha delantera
model = modelaux2;
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Rin_M.RenderModel();        //Muestra rin

//Llanta derecha trasera
model = modelaux;
model = glm::translate(model, glm::vec3(2.7f, 0.4f, -1.65f));
model = glm::rotate(model, glm::radians(mainWindow.getrotaLlantas()), glm::vec3(0.0f, 0.0f, 1.0f));
modelaux2 = model;
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
LlantaDerTrasera_M.RenderModel();       //Muestra Llanta

//Rin derecha delantera
model = modelaux2;
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Rin_M.RenderModel();        //Muestra rin

//Llanta Izquierda trasera
model = modelaux;
model = glm::translate(model, glm::vec3(2.7f, 0.4f, 1.65f));
model = glm::rotate(model, glm::radians(mainWindow.getrotaLlantas()), glm::vec3(0.0f, 0.0f, 1.0f));
modelaux2 = model;
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
LlantaIzqTrasera_M.RenderModel();       //Muestra Llanta

//Rin Izquierda trasera
model = modelaux2;
model = glm::rotate(model, 180 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Rin_M.RenderModel();        //Muestra rin

//Llanta Izquierda Delantera
model = modelaux;
model = glm::translate(model, glm::vec3(-3.1f, 0.4f, 1.65f));
model = glm::rotate(model, glm::radians(mainWindow.getrotaLlantas()), glm::vec3(0.0f, 0.0f, 1.0f));
modelaux2 = model;
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
LlantaIzqDelantera_M.RenderModel();     //Muestra Llanta

//Rin Izquierda Delantera
model = modelaux2;
model = glm::rotate(model, 180 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Rin_M.RenderModel();        //Muestra rin

//Cofre
model = modelaux;
model = glm::translate(model, glm::vec3(-2.15f, 1.99f, 0.0f));
model = glm::rotate(model, glm::radians(mainWindow.getrotaCofre()), glm::vec3(0.0f, 0.0f, 1.0f));   //Rotar con V y B
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Cofre_M.RenderModel();      //Muestra cofre
```

```
550
551         //Parrilla
552         model = modelaux;
553         model = glm::translate(model, glm::vec3(-4.82f, 0.95f, 0.0f));
554         glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
555         Parrilla_M.RenderModel();       //Muestra parrilla
556
557
```

Dado que para poder visualizar un poco mejor la parrilla se imprimió o mostró desde otro modelo al igual que los rines, por lo que se tuvieron que hacer 2 modelos uno para el rin, y otro para la parrilla.



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

En general no hubo problemas, solo que fue algo complicado a la hora de ubicar bien las normales de cada cara o triangulo del octaedro, ya que si se ponía en la posición equivocada la textura no se veía.

Y en el caso del carro, se me complicó a la hora de texturizar en blender, dado que hay muchos vértices y es complicado a la hora de ajustar los vértices en una estructura o en este caso la imagen donde las dimensiones no son las mismas dado que se enciman demasiado los vértices y se hay que tener que ubicar lo mejor posible para que el texturizado se lo más legible o limpio posible. Creo yo que fue el cofre y las llantas lo que más se me complicó.

3.- Conclusión:

a. Los ejercicios del reporte: Complejidad, Explicación.

Considero que los ejercicios fueron sencillos pero muy elaborados, dado que no teníamos previo manejo para texturizar, fue demasiado tardado, puede que haya mejores formas para poder realizarlo, por lo que siento que fue muy tardado a pesar de que el ejercicio era sencillo,

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

Creo que estuvo bien explicado la práctica, lo único que creo que estaría bien es mencionar sobre las consideraciones que hay que tomar a la hora de escoger la imagen del carro para texturizarlo, ya que la imagen puede que no esté en el mejor ángulo para poder maniobrar o trabar para el texturizado, por lo que se llega a complicar bastante, y por ello pienso que estaría bien que esas consideraciones se puedan decir para escoger la imagen.

c. Conclusión

Con esta práctica pude aprender el como texturizar por medio de código y en un software de modelado 3D por lo que ahora puedo entender el cómo se realiza este proceso, al igual de que consideraciones se puede hacer para facilitar texturizado.

1. Bibliografía en formato APA

- G-Truc Creation. (s.f.). GLM 0.9.9 Manual. Recuperado de https://chromium.googlesource.com/external/github.com/g-truc/glm/%2B/0.9.9-a2/manual.md
- Blender Foundation. (s.f.). *Selecting — Blender Manual*. Recuperado de https://docs.blender.org/manual/en/2.80/modeling/meshes/selecting.html