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: Barragán Pilar Diana

Nº de Cuenta: 318147981

GRUPO DE LABORATORIO: 03

GRUPO DE TEORÍA: 04

SEMESTRE 2025-1

FECHA DE ENTREGA LÍMITE: 29 de septiembre de 2024

CALIFICACIÓN: _____

Práctica 6: Texturizado

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

Para crear el dado de diez caras que se pide utilice el código de la pirámide cuadrangular adaptándolo para que fuera una pirámide pentagonal, cambiando los valores de los vértices para que se dibujará cada uno de los triángulos y la punta de la pirámide para poner las coordenadas de la respectiva textura con la cara correspondiente, también cambie los índices para dibujar cada cara por separado.

```
void CrearPiramidePentagonal()
{

    unsigned int piramidepentagonal_indices[] = {
        // Caras triangulares que conectan la punta con cada lado del pentágono
        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

        // Base pentagonal (hacemos 3 triángulos para formar la base)
        0, 1, 4,  // Primer triángulo
        4, 7, 10,  // Segundo triángulo
        10, 13, 0   // Tercer triángulo
    };



    GLfloat piramidepentagonal_vertices[] = {
        //X     Y      Z            S    T           NX     NY     NZ
        0.5f, -0.5f,  0.0f,       0.29f, 0.57f,    0.0,  0.0f,   -1.0f,  // Vértice 0  //0
        0.15f, -0.5f,  0.47f,     0.04f, 0.57f,    0.0f,  0.0f,   -1.0f,  // Vértice 1  //1
        0.0f,  0.5f,  0.0f,       0.17f, 0.98f,    0.0f,  0.0f,   -1.0f,   // Vértice 5 (punta de la pirámide) //2

        0.15f, -0.5f,  0.47f,     0.63f, 0.57f,    -1.0f, 0.0f, 0.0f,  // Vértice 1  //3
        -0.4f, -0.5f,  0.29f,     0.37f, 0.57f,    -1.0f, 0.0f, 0.0f,  // Vértice 2  //4
        0.0f,  0.5f,  0.0f,       0.5f, 0.98f,     -1.0f, 0.0f, 0.0f,   // Vértice 5 (punta de la pirámide) //5

        -0.4f, -0.5f,  0.29f,     0.95f, 0.57f,    0.0f,  0.0f,  1.0f,  // Vértice 2  //6
        -0.4f, -0.5f, -0.29f,     0.72f, 0.57f,    0.0f,  0.0f,  1.0f,  // Vértice 3   //7
       0.0f,  0.5f,  0.0f,        0.83f, 0.98f,    0.0f,  0.0f,  1.0f,   // Vértice 5 (punta de la pirámide) //8

        -0.4f, -0.5f, -0.29f,     0.29f, 0.07f,     1.0f,  0.0f,  0.0f,  // Vértice 3   //9
         0.15f, -0.5f, -0.47f,    0.05f, 0.07f,     1.0f,  0.0f,  0.0f,  // Vértice 4   //10
         0.0f,  0.5f,  0.0f,      0.165f, 0.43f,    1.0f,  0.0f,  0.0f,   // Vértice 5 (punta de la pirámide) //11

         0.15f, -0.5f, -0.47f,    0.62f, 0.07f,     0.0f,   1.0f,   0.0f,  // Vértice 4   //12
         0.5f, -0.5f,  0.0f,      0.38f, 0.07f,     0.0f,   1.0f,   0.0f,  // Vértice 0  //13
         0.0f,  0.5f,  0.0f,      0.5f, 0.43f,      0.0f,   1.0f,   0.0f   // Vértice 5 (punta de la pirámide) //14
    };

    Mesh* piramide = new Mesh();
    piramide->CreateMesh(piramidepentagonal_vertices, piramidepentagonal_indices, 120, 24);
    meshList.push_back(piramide);
}
```

Figura 1. Función de la pirámide Pentagonal.

En el programa de GIMP modifique la imagen con la textura de las caras que le iba a asignar, cambiando la escala y ordenando los triángulos, exporte el archivo y declare en el código dos texturas la numero uno y la número dos, luego cargue los modelos especificando la ruta.

```
Texture brickTexture;
Texture dirtTexture;
Texture plainTexture;
Texture pisoTexture;
Texture dadoTexture;
Texture dadodiez1Texture;
Texture dadodiez2Texture;
Texture logofiTexture;
```

```
//dadoTexture = Texture("Textures/dado-de-numeros.png");
dadoTexture = Texture("Textures/dado_animales.tga");
dadoTexture.LoadTextureA();
dadodiez1Texture = Texture("Textures/dadodiez_1.tga");
dadodiez1Texture.LoadTextureA();
dadodiez2Texture = Texture("Textures/dadodiez_2.tga");
dadodiez2Texture.LoadTextureA();
logofiTexture = Texture("Textures/escudo_fi_color.tga");
logofiTexture.LoadTextureA();
```

Figura 2. Declaración y carga de las texturas.

Finalmente, para formar el dado coloque una pirámide pentagonal debajo de otra pirámide pentagonal las traslade y escale para que se posicionarán correctamente.

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

Para importar las llantas y texturizarlas en 3ds Max utilice el modelo de auto de la practica numero cinco, importando las llantas para que utilizando el Slade Material Editor me fuera posible poner dos texturas distintas en un mismo objeto como lo es el caucho de la llanta y el rin. Después de tener texturizadas las llantas simplemente las exporte.

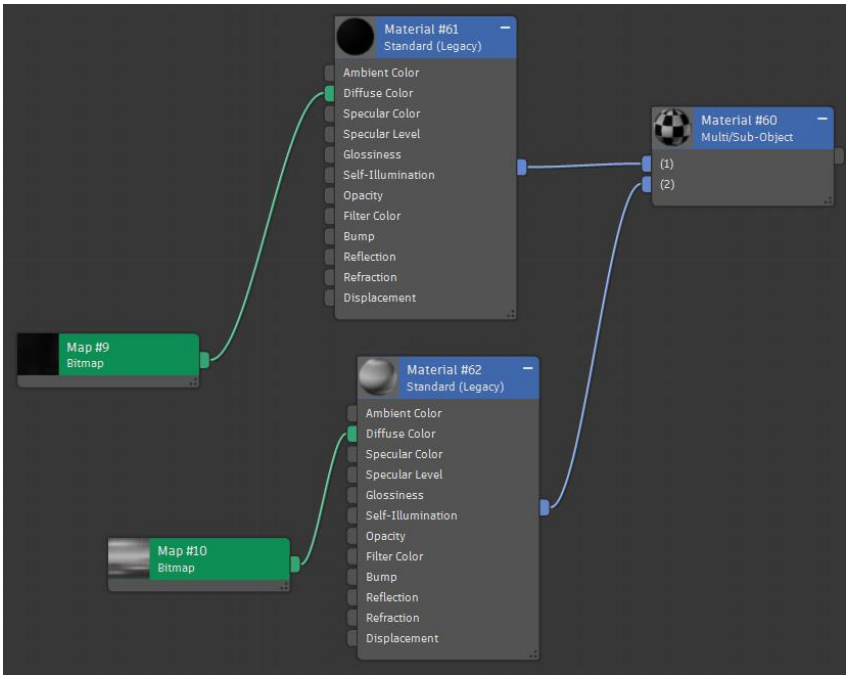

Figura 3. Texturizado de llantas.

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 este ultimo ejercicio para texturizar el carro realice el mismo procedimiento que con las llantas, importe cada textura a la que le había realizado modificaciones anteriormente en GIMP y con Slade Material Editor seleccionando el standard (legacy)->Diffuse->bit map fui aplicando cada textura a cada parte del carro.

En el código al igual que en la practica 5 declare los modelos con las partes del carro que son el cofre, el cuerpo del carro y las llantas, después cargue los modelos especificando su ruta

```
Model Dado_A;
Model Carro;
Model Llanta1;
Model Llanta2;
Model Llanta3;
Model Llanta4;
Model Capo;
```

```
Carro = Model();
Carro.LoadModel("Models/carro.obj");
Llanta1 = Model();
Llanta1.LoadModel("Models/llanta1.obj");
Llanta2 = Model();
Llanta2.LoadModel("Models/llanta2.obj");
Llanta3 = Model();
Llanta3.LoadModel("Models/llanta3.obj");
Llanta4 = Model();
Llanta4.LoadModel("Models/llanta4.obj");
Capo = Model();
Capo.LoadModel("Models/capo.obj");
```

Figura 4. Declaración y carga de modelos.

Por último, simplemente utilizando el código de la practica cinco con sus respectivos traslate, acomode el carro con sus cuatro llantas y el cofre para que sea correcto su diseño.

```
//CARRO
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f, -2.0f, -1.5f));
model = glm::translate(model, glm::vec3(0.0f, 6.5f, -1.5f));
modelaux = model;
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Carro.RenderModel();

//Capo
model = modelaux;
model = glm::translate(model, glm::vec3(0.4f, 2.2f, 9.5f));
//model = glm::rotate(model, glm::radians(mainWindow.getangulomandibula()), glm::v
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Capo.RenderModel();

//Llanta 1 delantera derecha
model = modelaux;
model = glm::translate(model, glm::vec3(7.0f, -3.4f, 13.4f));
//model = glm::rotate(model, glm::radians(mainWindow.getangulollanta()), glm::vec3
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Llanta1.RenderModel();
```

```
//Llanta2 delantera izquierda
model = modelaux;
model = glm::translate(model, glm::vec3(-6.0f, -3.4f, 12.4f));
//model = glm::rotate(model, glm::radians(mainWindow.getangulopata2()), glm::ve
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Llanta2.RenderModel();

//Llanta 3 trasera derecha
model = modelaux;
model = glm::translate(model, glm::vec3(7.0f, -3.4f, -9.5f));
//model = glm::rotate(model, glm::radians(mainWindow.getangulollanta()), glm::v
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Llanta3.RenderModel();

//Llanta 4 trasera derecha
model = modelaux;
model = glm::translate(model, glm::vec3(-6.0f, -3.4f, -9.5f));
//model = glm::rotate(model, glm::radians(mainWindow.getangulollanta()), glm::v
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Llanta4.RenderModel();
```

Figura 6. Jerarquía y acomodo del carro.

EJECUCIÓN DEL PROGRAMA:

Figura 7. Ejecución del dado de diez caras y el carro.

Figura 8. Ejecución del dado de diez caras y el carro.

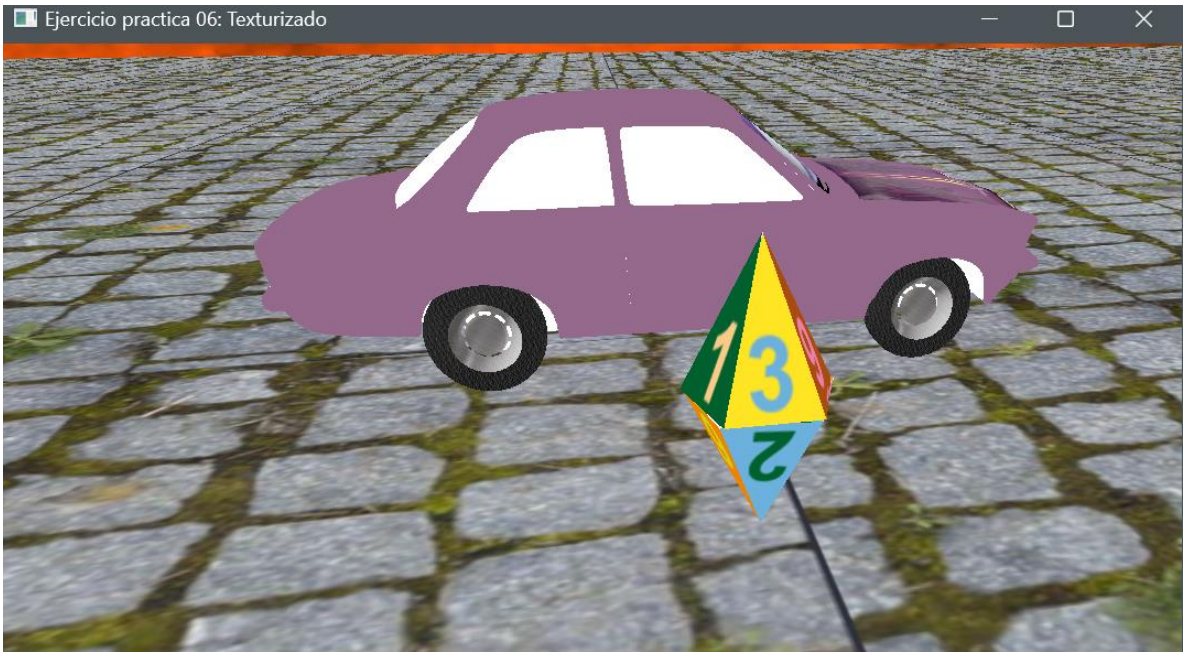

Figura 9. Ejecución del dado de diez caras y el carro.

Figura 10. Ejecución del dado de diez caras y el carro.

Al compilar el código de esta práctica no se me presento ningún inconveniente respecto a la programación de este, puesto que todo funciono con normalidad.

CONCLUSIÓN:

Con la realización de esta práctica pude relacionarme y aprender más del programa 3ds Max puesto que el aplicar las texturas fue algo difícil debido a la poca practica que tengo implementándolo, sin embargo, al realizar el carro me quedo más claro como se utiliza correctamente el programa. De igual modo el texturizar desde el código en opengl, también tuvo sus complicaciones pues debí ser más precisa en las coordenadas de los vértices respecto a la imagen.