



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: Gabriel Patricio Balam Flores

Nº de Cuenta: 320280324

GRUPO DE LABORATORIO: 03

GRUPO DE TEORÍA: 04

SEMESTRE 2026-1

FECHA DE ENTREGA LÍMITE: 12/oct

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 #01

El primer ejercicio de la práctica consiste en crear un dado de ocho caras y aplicar su texturizado. La primera parte fue sencilla: construir el octaedro implica definir los triángulos y agregarlos a la mesh_list, como con cualquier otra figura. La parte más complicada fue el texturizado por código, ya que resulta difícil encontrar una imagen con las características adecuadas para facilitar el proceso. En mi caso, opté por crear mi propia textura utilizando GeoGebra y algunos íconos de números.

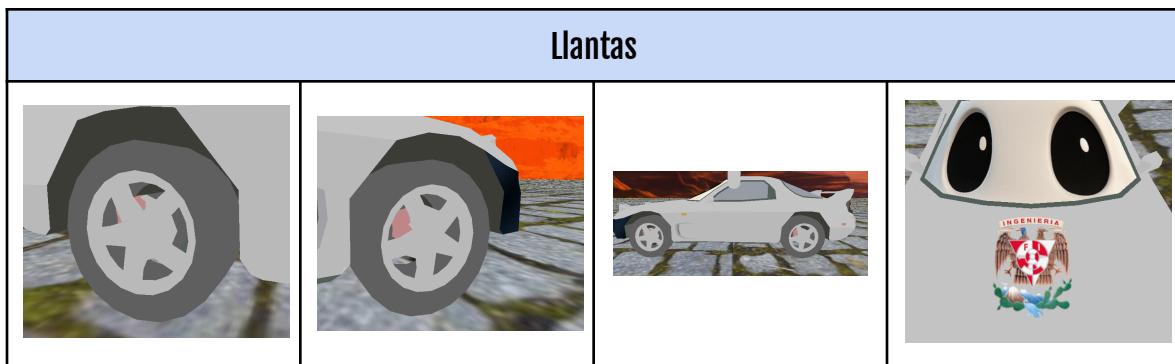
Textura	Dado	
Dado		

Crear forma	Crear forma / Textura
<pre>void CrearDado8() { unsigned int oct_indices[] = { // Up +x+z 0, 1, 2, // Up +x-z 3, 4, 5, // Up -x+z 6, 7, 8, // Up -x-z 9, 10, 11, // Bot +x+z 12, 13, 14, // Bot +x-z 15, 16, 17, // Bot -x+z 18, 19, 20, // Bot -x-z 21, 22, 23 }; GLfloat oct_vertices[] = { </pre>	<pre>GLfloat oct_vertices[] = { // Up +x+z 1 //x y z S T NX NY NZ 0.5f, 0.0f, 0.0f, 0.0f, 0.3f, 0.0f, 0.0f, -1.0f, //0 0.0f, 0.0f, 0.5f, 0.286f, 0.3f, 0.0f, 0.0f, -1.0f, //1 0.0f, 0.5f, 0.0f, 0.143f, 0.548f, 0.0f, 0.0f, -1.0f, //2 // Up +x-z 2 //x y z S T 0.5f, 0.0f, 0.0f, 0.143f, 0.548f, -1.0f, 0.0f, 0.0f, 0.0f, 0.0f, -0.5f, 0.428f, 0.548f, -1.0f, 0.0f, 0.0f, 0.0f, 0.5f, 0.0f, 0.286f, 0.3f, -1.0f, 0.0f, 0.0f, // Up -x+z 4 //x y z S T -0.5f, 0.0f, 0.0f, 0.572f, 0.3f, 0.0f, 0.0f, 1.0f, 0.0f, 0.0f, 0.5f, 0.286f, 0.3f, 0.0f, 0.0f, 1.0f, 0.0f, 0.5f, 0.0f, 0.428f, 0.548f, 0.0f, 0.0f, 1.0f, // Up -x-z 3 //x y z S T -0.5f, 0.0f, 0.0f, 0.428f, 0.548f, 1.0f, 0.0f, 0.0f, 0.0f, 0.0f, -0.5f, 0.143f, 0.548f, 1.0f, 0.0f, 0.0f, 0.0f, 0.5f, 0.0f, 0.286f, 0.796f, 1.0f, 0.0f, 0.0f, // Bot +x+z 5 //x y z S T NX NY NZ 0.5f, 0.0f, 0.0f, 0.428f, 0.548f, 0.0f, 0.0f, -1.0f, //0 0.0f, 0.0f, 0.5f, 0.714f, 0.548f, 0.0f, 0.0f, -1.0f, //1 0.0f, -0.5f, 0.0f, 0.572f, 0.3f, 0.0f, 0.0f, -1.0f, //2 // Bot +x-z 6 //x y z S T 0.5f, 0.0f, 0.0f, 0.858f, 0.3f, -1.0f, 0.0f, 0.0f, 0.0f, 0.0f, -0.5f, 0.572f, 0.3f, -1.0f, 0.0f, 0.0f, 0.0f, -0.5f, 0.0f, 0.714f, 0.548f, -1.0f, 0.0f, 0.0f, // Bot -x+z 7 //x y z S T -0.5f, 0.0f, 0.0f, 0.714f, 0.548f, 0.0f, 0.0f, 1.0f, 0.0f, 0.0f, 0.5f, 1.0f, 0.548f, 0.0f, 0.0f, 1.0f, 0.0f, -0.5f, 0.0f, 0.858f, 0.3f, 0.0f, 0.0f, 1.0f, // Bot -x-z 8 //x y z S T -0.5f, 0.0f, 0.0f, 1.0f, 0.548f, 1.0f, 0.0f, 0.0f, 0.0f, 0.0f, -0.5f, 0.714f, 0.548f, 1.0f, 0.0f, 0.0f, 0.0f, -0.5f, 0.0f, 0.858f, 0.796f, 1.0f, 0.0f, 0.0f, }; Mesh* dado = new Mesh(); dado->CreateMesh(oct_vertices, oct_indices, 192, 36); meshList.push_back(dado); </pre>
Importar textura	
<pre>dado8Texture = Texture("Textures/dado_8.png"); dato8Texture.LoadTextureA();</pre>	
Mostrar forma y textura	
<pre>model = glm::mat4(1.0); model = glm::translate(model, glm::vec3(-1.5f, 4.5f, -5.0f)); model = glm::scale(model, glm::vec3(1.5f, 1.5f, 1.5f)); glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); dato8Texture.UseTexture(); meshList[5]->RenderMesh();</pre>	

Ejercicio #02

El segundo ejercicio de la práctica tomó más tiempo, aunque no resultó tan complicado. El modelo que elegí estaba bien diseñado, lo que me permitió dividir fácilmente el rin y el caucho. Una vez hecha esta separación, solo fue necesario aplicar la textura a cada uno de los elementos para finalizar el ejercicio.

También creé una imagen que contenía el escudo de la facultad, ya que si solo añadía el logo directamente, se producía un efecto de repetición en toda la superficie del coche, algo que no quería que sucediera.



Ejercicio #03

En este ejercicio tuve varias dudas de diseño. La imagen que elegí en el ejercicio previo tiene un diseño “sencillo”, lo que dificulta identificar qué partes del modelo deben modificarse. El primer inciso consistía en agregar los ojos al parabrisas; este proceso fue sencillo: primero edité la imagen en GIMP para obtener únicamente los ojos y luego utilicé el editor UV para ajustarlos de la mejor manera posible.

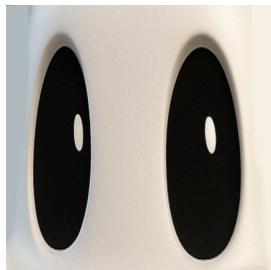
Otro elemento importante era la ropa. En este caso, no supe exactamente qué tanto agregar, por lo que opté por colocar únicamente en la parrilla. El posicionamiento de la textura no permite apreciar los pliegues, aunque sí es posible distinguir que se trata de ropa.

Por último, en mi imagen original el cofre no tenía ningún elemento distintivo, así que decidí texturizarlo completamente de blanco y añadir algunos detalles, como las luces traseras y los cuernos característicos de mi personaje.

Coche



Textura



Importar mi carro

```
// Mi carro =====
model = glm::mat4(1.0);
modelaux = model;

model = glm::translate(model, glm::vec3(0.0f, 2.3f, -28.0f));
model = glm::scale(model, glm::vec3(5.0f, 5.0f, 5.0f));

auxmodel = model;

// Para el cuerpo del carro
model = glm::translate(model, glm::vec3(0.44f, 0.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Carro_cuerpo.RenderModel();

// Para el cofre
model = auxmodel;
model = glm::translate(model, glm::vec3(0.777f, 0.3625f, -0.00f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Carro_cofre.RenderModel();

// Para la llanta izquierda delantera
model = auxmodel;
model = glm::translate(model, glm::vec3(1.8f, -0.4f, 0.9f));
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Carro_llanta_1.RenderModel();

// Para la llanta izquierda trasera
model = auxmodel;
model = glm::translate(model, glm::vec3(-1.5f, -0.4f, 0.9f));
model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Carro_llanta_1.RenderModel();

// Para la llanta derecha delantera
model = auxmodel;
model = glm::translate(model, glm::vec3(1.8f, -0.4f, -0.9f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Carro_llanta_1.RenderModel();

// Para la llanta derecha trasera
model = auxmodel;
model = glm::translate(model, glm::vec3(-1.5f, -0.4f, -0.9f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Carro_llanta_1.RenderModel();
```

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.

Contratiempos

El principal problema de esta práctica fue determinar si realmente estaba cumpliendo con los requisitos del ejercicio. Debido a la sencillez de mi modelo e imagen, no supe qué otros detalles podría agregar.

3.- Conclusión:

- a. Los ejercicios del reporte: Complejidad, Explicación.
 - b. Comentarios generales.
 - c. Conclusión
1. Bibliografía en formato APA

Conclusión

Esta práctica fue muy enriquecedora debido a la variedad de herramientas que se necesitaron para completar todas las actividades. En todos los ejercicios utilicé GIMP para la edición de imágenes de texturas, lo que me permitió familiarizarme con su interfaz y sus funciones.

En OpenGL apliqué texturas por código en el dado de ocho caras. Los ángulos característicos de esta figura impidieron “automatizar” el proceso de asignación de coordenadas UV, lo que me ayudó a reflexionar más sobre cada paso.

Por último, en los dos ejercicios finales, Blender fue una herramienta clave para la texturización mediante imágenes. Pude reforzar los conocimientos adquiridos en prácticas anteriores y aprender a utilizar de manera más efectiva la sección de UV.

Bibliografía

- Página principal. (26 de septiembre de 2018). *Wiki de OpenGL* . Recuperado el 29 de agosto de 2025 de http://www.khronos.org/opengl/wiki_OpenGL/index.php?title=Main_Page&oldid=14430 .