UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE INGENIERÍA ELÉCTRICA

INGENIERÍA EN COMPUTACIÓN

LABORATORIO DE COMPUTACIÓN GRÁFICA e INTERACCIÓN HUMANO COMPUTADORA

EJERCICIOS DE CLASE Nº 06

NOMBRE COMPLETO: MORENO SANTOYO MARIANA

Nº de Cuenta: 319170252

GRUPO DE LABORATORIO: 11

GRUPO DE TEORÍA: 04

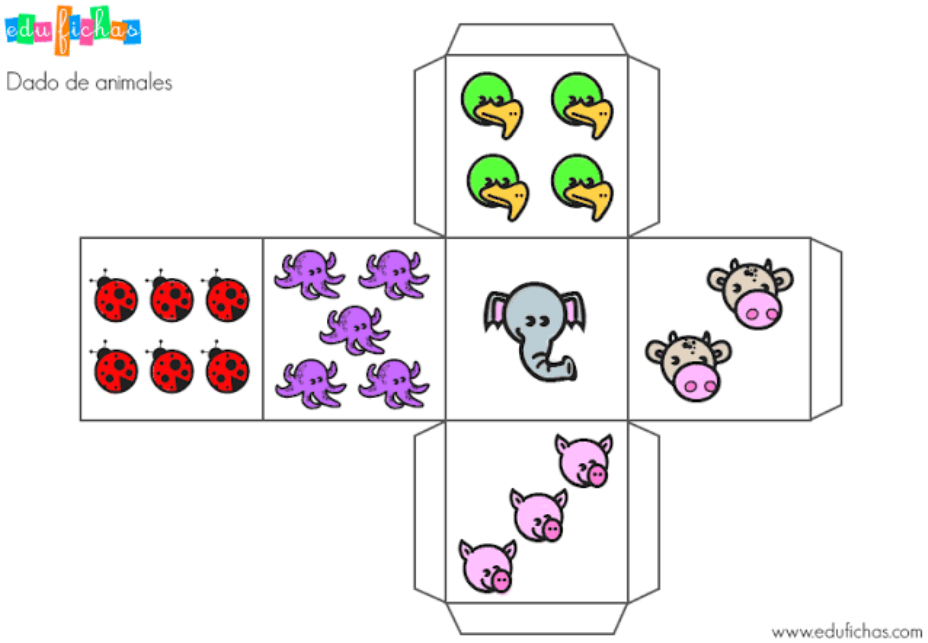
SEMESTRE 2025-2

FECHA DE ENTREGA LÍMITE: 24/04/2025

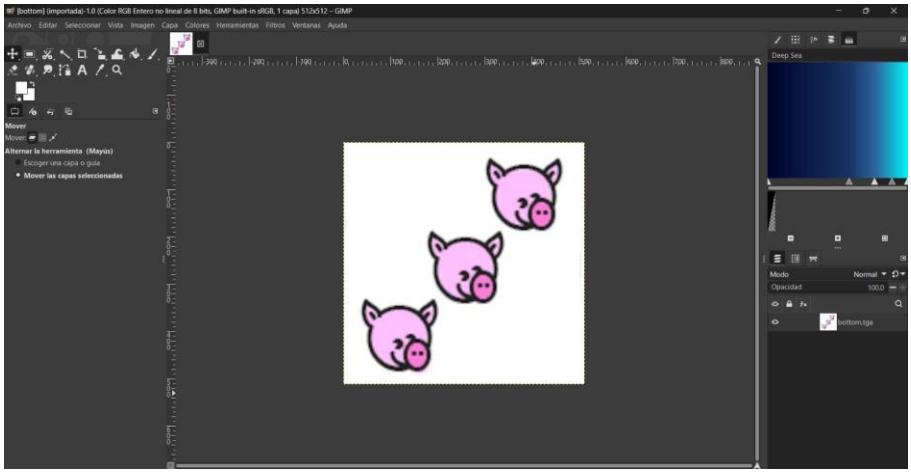
CALIFICACIÓN: _____

- **Ejercicio 1: Texturizar su cubo con la imagen dado_animales ya optimizada por ustedes**

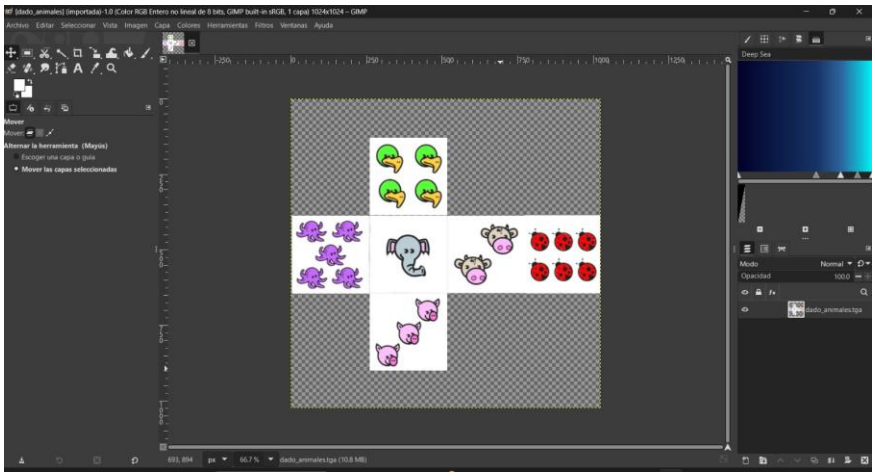
Para llevar a cabo la práctica, el primer paso fue tomar la imagen original proporcionada por el profesor y trabajarla utilizando la aplicación de edición de imágenes GIMP. Con el objetivo de abordar el proceso de texturización directamente desde OpenGL, se optó por reorganizar parcialmente la imagen —no de forma completa—, de modo que los cálculos relacionados con el posicionamiento de las texturas fueran lo más simples y eficientes posible.



Como parte del proceso de reorganización y optimización, se procedió a recortar cada cara de la imagen sin incluir pestañas, siguiendo un esquema específico. En este documento solo se adjunta una de dichas imágenes con el fin de mantener el reporte conciso; sin embargo, el repositorio de Git contiene todas las imágenes utilizadas.



Con ayuda del asistente ChatGPT, y con el objetivo de lograr una organización óptima que facilitara la localización de cada cara para el proceso de texturizado, se optó por utilizar la siguiente configuración para una imagen de dimensiones 1024x1024 píxeles. Esta resolución fue seleccionada por su característica de ser cuadrada y, además, por conservar la propiedad de ser una potencia de dos, lo cual es recomendable para un mejor rendimiento en OpenGL.



Finalmente, tras realizar algunos cálculos matemáticos, se determinaron los siguientes vértices. Estos fueron ajustados adicionalmente con el propósito de evitar imperfecciones en la visualización y asegurar una correcta alineación durante el proceso de texturizado.

```
// front (cara delantera: z = 0.5, normal = (0,0,1))
//x     y      z      S       T        NX     NY     NZ
- 0.5f, -0.5f,  0.5f,   0.260f,  0.375f,   0.0f,   0.0f,   1.0f,
 0.5f, -0.5f,  0.5f,   0.500f,  0.375f,   0.0f,   0.0f,   1.0f,
 0.5f,  0.5f,  0.5f,   0.500f,  0.620f,   0.0f,   0.0f,   1.0f,
-0.5f,  0.5f,  0.5f,   0.260f,  0.620f,   0.0f,   0.0f,   1.0f,

// right (cara derecha: x = 0.5, normal = (1,0,0))
//x     y      z      S       T        NX     NY     NZ
 0.5f, -0.5f,  0.5f,   0.520f,  0.375f,   1.0f,   0.0f,   0.0f,
 0.5f, -0.5f, -0.5f,   0.750f,  0.375f,   1.0f,   0.0f,   0.0f,
 0.5f,  0.5f, -0.5f,   0.750f,  0.620f,   1.0f,   0.0f,   0.0f,
 0.5f,  0.5f,  0.5f,   0.520f,  0.620f,   1.0f,   0.0f,   0.0f,

// back (cara trasera: z = -0.5, normal = (0,0,-1))
//x     y      z      S       T        NX     NY     NZ
-0.5f, -0.5f, -0.5f,   0.750f,  0.375f,   0.0f,   0.0f,  -1.0f,
 0.5f, -0.5f, -0.5f,   1.0f,    0.375f,   0.0f,   0.0f,  -1.0f,
 0.5f,  0.5f, -0.5f,   1.0f,    0.620f,   0.0f,   0.0f,  -1.0f,
-0.5f,  0.5f, -0.5f,   0.750f,  0.620f,   0.0f,   0.0f,  -1.0f,

// left (cara izquierda: x = -0.5, normal = (-1,0,0))
//x     y      z      S       T        NX     NY     NZ
-0.5f, -0.5f, -0.5f,   0.01f,   0.375f,  -1.0f,  0.0f,  0.0f,
-0.5f, -0.5f,  0.5f,   0.250f,  0.375f,  -1.0f,  0.0f,  0.0f,
-0.5f,  0.5f,  0.5f,   0.250f,  0.620f,  -1.0f,  0.0f,  0.0f,
-0.5f,  0.5f, -0.5f,   0.01f,   0.620f,  -1.0f,  0.0f,  0.0f,

// bottom (cara inferior: y = -0.5, normal = (0,-1,0))
// x     y      z      S       T        NX     NY     NZ
-0.5f, -0.5f,  0.5f,   0.260f,  0.370f,   0.0f,  -1.0f,  0.0f,
 0.5f, -0.5f,  0.5f,   0.490f,  0.370f,   0.0f,  -1.0f,  0.0f,
 0.5f, -0.5f, -0.5f,   0.490f,  0.125f,   0.0f,  -1.0f,  0.0f,
-0.5f, -0.5f, -0.5f,   0.260f,  0.125f,   0.0f,  -1.0f,  0.0f,


// UP (cara superior: y = 0.5, normal = (0,1,0))
//x     y      z      S       T        NX     NY     NZ
-0.5f,  0.5f,  0.5f,   0.260f,  0.625f,   0.0f,  1.0f,  0.0f,
 0.5f,  0.5f,  0.5f,   0.500f,  0.625f,   0.0f,  1.0f,  0.0f,
 0.5f,  0.5f, -0.5f,   0.500f,  0.870f,   0.0f,  1.0f,  0.0f,
-0.5f,  0.5f, -0.5f,   0.260f,  0.870f,   0.0f,  1.0f,  0.0f,
```

NOTA:cabe resaltar se agregaron las normales correctas y la llamada a la función average normals que no viene en classroom, pero sí en las instrucciones justo arriba del código modificado

Se cargó la textura

```
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-de-numeros.png");
dadoTexture.LoadTextureA();
dadoTexture_Animales = Texture("Textures/dado_animales.tga");
dadoTexture_Animales.LoadTextureA();
logofiTexture = Texture("Textures/escudo_fi_color.tga");
logofiTexture.LoadTextureA();
```

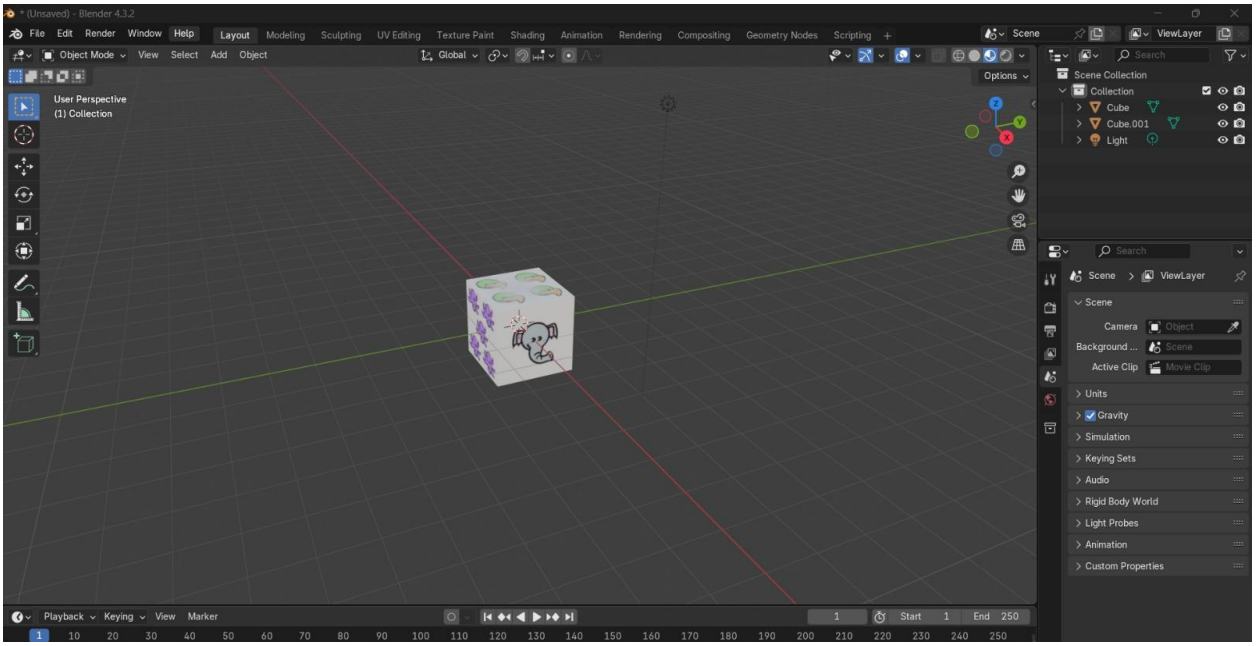
Se inicializo el modelo con su textura

```
//Dado de Opengl
//Ejercicio 1: Texturizar su cubo con la imagen dado_animales ya optimizada por ustedes
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-1.5f, 4.5f, -2.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
dadoTexture_Animales.UseTexture();
meshList[4]->RenderMesh();

//Ejercicio 2:Importar el cubo texturizado en el programa de modelado con
//la imagen dado_animales ya optimizada por ustedes

//Dado importado
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-3.0f, 3.0f, -2.0f));
model = glm::scale(model, glm::vec3(0.05f, 0.05f, 0.05f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Dado_M.RenderModel();
```

RESULTADOS

- **Ejercicio 2: Importar el cubo texturizado en el programa de modelado con la imagen dado_animales ya optimizada por ustedes**

Posteriormente, para este ejercicio se utilizó la imagen modificada en el apartado anterior. Siguiendo los pasos indicados en el video de referencia, se procedió a realizar la exportación del modelo ya texturizado en formato OBJ. Este proceso resultó ser más sencillo, preciso y rápido en comparación con métodos anteriores, lo que facilitó considerablemente el flujo de trabajo.

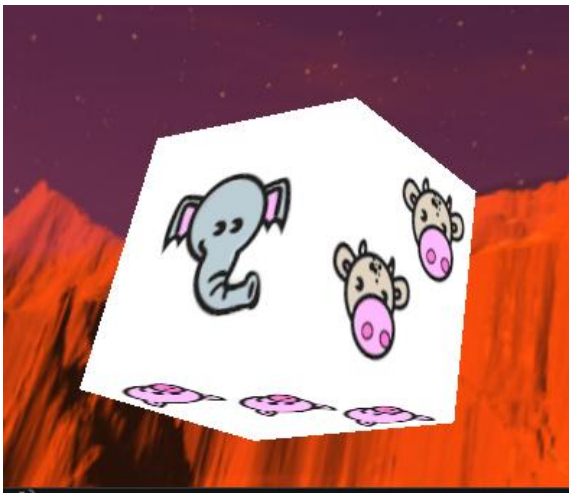


Se realizó la carga del modelo de manera normal y se ejecutó de manera normal

```
Kitt_M = Model();
Kitt_M.LoadModel("Models/kitt_optimizado.obj");
Llanta_M = Model();
Llanta_M.LoadModel("Models/llanta_optimizada.obj");
Dado_M = Model();
Dado_M.LoadModel("Models/cubo_texture.obj");
```

```
//Dado de Opengl
//Ejercicio 1: Texturizar su cubo con la imagen dado_animales ya optimizada por ustedes
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-1.5f, 4.5f, -2.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
dadoTexture_Animales.UseTexture();
meshList[4]->RenderMesh();

//Ejercicio 2:Importar el cubo texturizado en el programa de modelado con
//la imagen dado_animales ya optimizada por ustedes

//Dado importado
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(-3.0f, 3.0f, -2.0f));
model = glm::scale(model, glm::vec3(0.05f, 0.05f, 0.05f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Dado_M.RenderModel();
```

RESULTADOS



Conclusiones / Comentario Final

En general, la práctica permitió comprender de manera más clara el proceso de texturización y organización de imágenes para su uso en OpenGL. Uno de los aspectos que más trabajo me costó fue familiarizarme con el programa de edición de imágenes, ya que al principio me resultó complicado recortar y organizar las texturas correctamente. Sin embargo, al avanzar con la práctica, especialmente en la parte donde se exporta el modelo ya texturizado desde Blender en formato OBJ, el proceso se volvió mucho más sencillo, rápido y preciso. Considero que los videos de apoyo fueron útiles para entender los pasos a seguir, aunque pienso que el video relacionado con la edición de imágenes podría mejorar incluyendo un poco más de detalle o explicaciones adicionales para facilitar su comprensión.