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: Aguilar Pérez José Ramón

Nº de Cuenta: 317515048

GRUPO DE LABORATORIO: 02

GRUPO DE TEORÍA: 04

SEMESTRE 2025-2

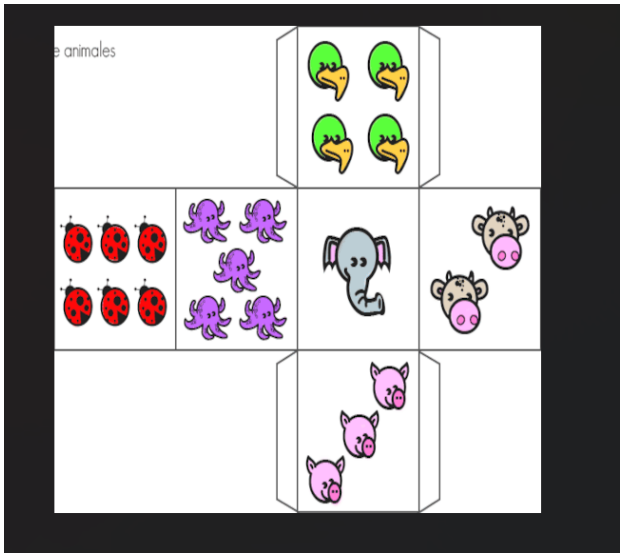
FECHA DE ENTREGA LÍMITE: 25/marzo/2025

CALIFICACIÓN: _____

EJERCICIOS DE SESIÓN:

1. **Actividades realizadas.**
   - **Ejercicio 1: Texturizar su cubo con la imagen dado_animales ya optimizada por ustedes**
   - **Ejercicio 2: Importar el cubo texturizado en el programa de modelado con la imagen dado_animales ya optimizada por ustedes.**

Para el primer ejercicio, se modificó la imagen para que tuviera las especificaciones necesarias. En primer lugar, se escaló a una proporción de $2^n x 2^n$ (512x512), se le dio un formato RGB y se importó en extensión .png para que así lo pueda trabajar el programa en OpenGL.



Una vez que la imagen ha sido optimizada, se calculan los vértices para cada una de las caras del cubo, el cual hace la función del dado. Es importante considerar la rotación de las caras para evitar resultados no deseados.

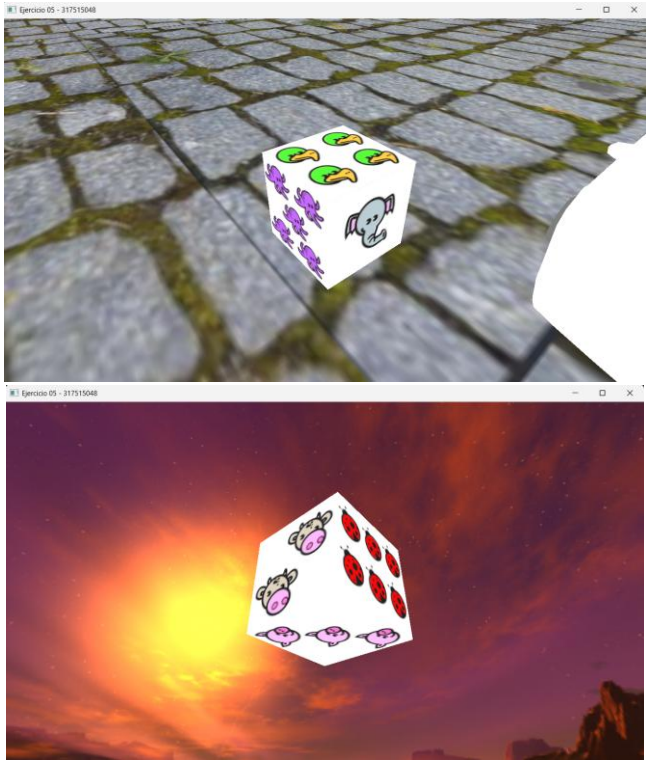
```
GLfloat cubo_vertices[] = {
    // front
    //x     y       z       S       T           NX      NY      NZ
    -0.5f, -0.5f,  0.5f,    0.51f,  0.34f,      0.0f,   0.0f,   -1.0f,  //0
    0.5f, -0.5f,  0.5f,     0.74f,  0.34f,      0.0f,   0.0f,   -1.0f,  //1
    0.5f,  0.5f,  0.5f,     0.74f,  0.64f,      0.0f,   0.0f,   -1.0f,  //2
    -0.5f,  0.5f,  0.5f,    0.51f,  0.64f,      0.0f,   0.0f,   -1.0f,  //3
    // right
    //x     y       z       S       T
    0.5f, -0.5f,  0.5f,     0.76f,  0.35f,      -1.0f,  0.0f,   0.0f,
    0.5f, -0.5f,  -0.5f,    0.99f,  0.35f,      -1.0f,  0.0f,   0.0f,
    0.5f,  0.5f,  -0.5f,    0.99f,  0.66f,      -1.0f,  0.0f,   0.0f,
    0.5f,  0.5f,  0.5f,     0.76f,  0.66f,      -1.0f,  0.0f,   0.0f,
    // back
    -0.5f, -0.5f, -0.5f,    0.24f,  0.34f,      0.0f,   0.0f,   1.0f,
    0.5f, -0.5f, -0.5f,     0.01f,  0.34f,      0.0f,   0.0f,   1.0f,
    0.5f,  0.5f, -0.5f,     0.01f,  0.64f,      0.0f,   0.0f,   1.0f,
    -0.5f,  0.5f, -0.5f,    0.24f,  0.64f,      0.0f,   0.0f,   1.0f,
```

```
    // left
    //x     y       z       S       T
    -0.5f, -0.5f,  -0.5f,   0.26f,  0.34f,      1.0f,   0.0f,   0.0f,
    -0.5f, -0.5f,  0.5f,    0.49f,  0.34f,      1.0f,   0.0f,   0.0f,
    -0.5f,  0.5f,  0.5f,    0.49f,  0.66f,      1.0f,   0.0f,   0.0f,
    -0.5f,  0.5f,  -0.5f,   0.26f,  0.66f,      1.0f,   0.0f,   0.0f,

    // bottom
    //x     y       z       S       T
    -0.5f, -0.5f,  0.5f,    0.51f,  0.32f,      0.0f,   1.0f,   0.0f,
    0.5f,  -0.5f,  0.5f,    0.74f,  0.32f,      0.0f,   1.0f,   0.0f,
     0.5f,  -0.5f,  -0.5f,  0.74f,  0.01f,      0.0f,   1.0f,   0.0f,
    -0.5f, -0.5f,  -0.5f,   0.51f,  0.01f,      0.0f,   1.0f,   0.0f,

    //UP
    //x        y       z       S       T
     -0.5f, 0.5f,  0.5f,    0.51f,  0.67f,      0.0f,   -1.0f,  0.0f,
     0.5f,  0.5f,  0.5f,    0.74f,  0.67f,      0.0f,   -1.0f,  0.0f,
      0.5f, 0.5f,  -0.5f,   0.74f,  0.99f,      0.0f,   -1.0f,  0.0f,
     -0.5f, 0.5f,  -0.5f,   0.51f,  0.99f,      0.0f,   -1.0f,  0.0f,

};
```

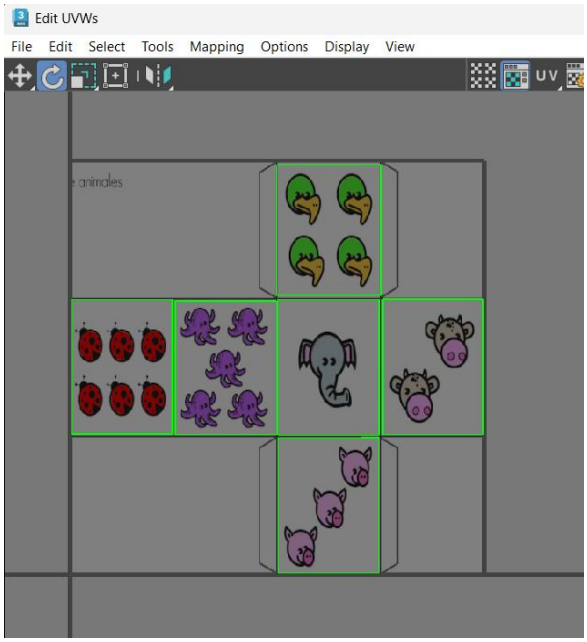
Finalmente, se instancia el cubo dentro del main del programa, así como la textura del dado con la imagen optimizada.

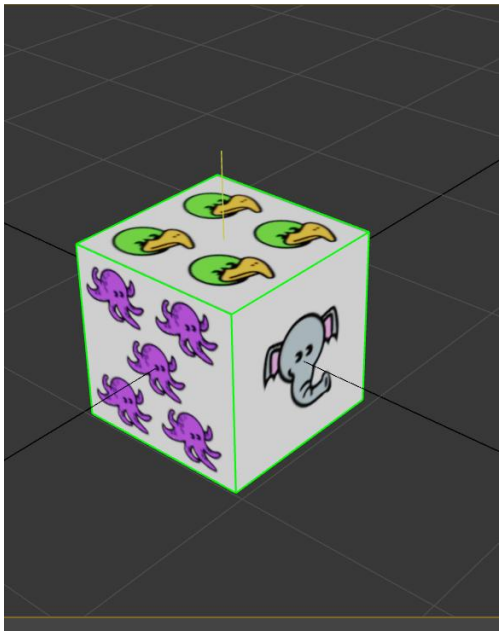
```
dadoTexture = Texture("Textures/dado_animales.png");
dadoTexture.LoadTextureA();
```

```
//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.UseTexture();
meshList[4]->RenderMesh();
```



Para el segundo ejercicio, se creó un cubo dentro de 3dsMax, donde la textura de este será la imagen optimizada del dado de animales. Dentro del menú de edición de UVWs, se ajustaron las caras del cubo para que coincidieran con las imágenes del dado, además de que estas deben concordar con lo obtenido con código anteriormente.





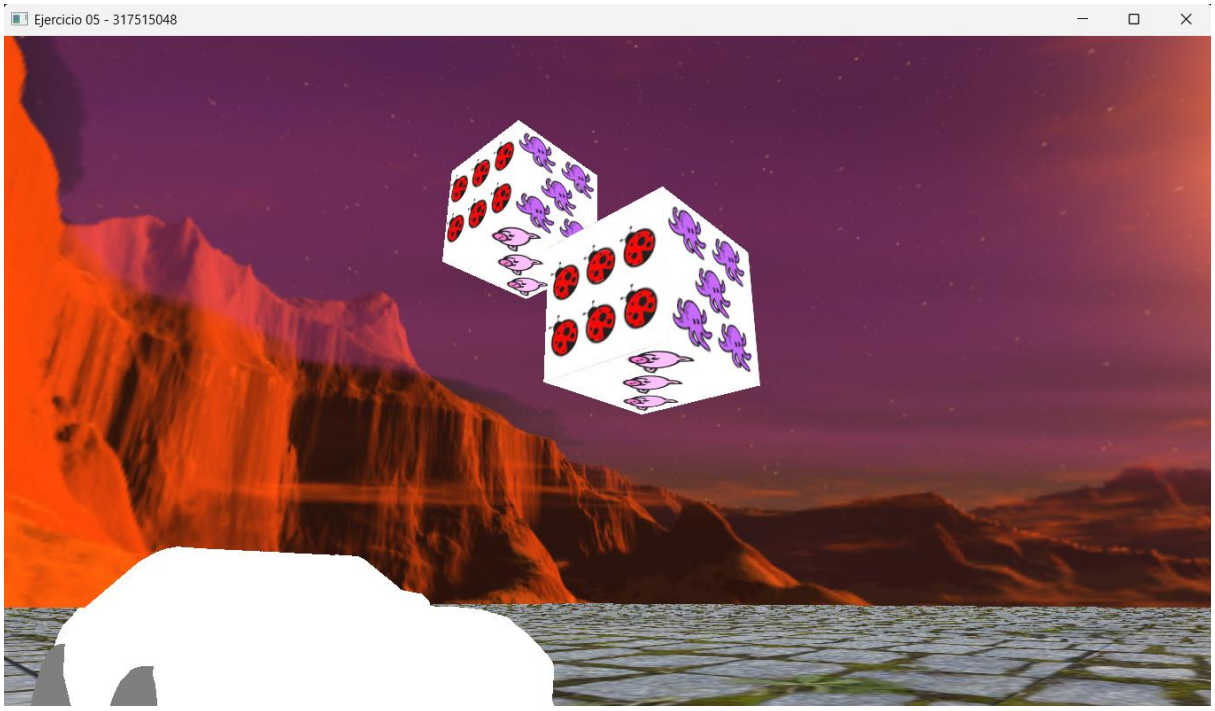
Una vez que se tenga el modelo listo, se exporta al programa principal, es importante añadir bien su mapa a la hora de exportar, de lo contrario no se verá correctamente las texturas añadidas.

```
Dado_M = Model();
Dado_M.LoadModel("Models/dado_animales.obj");
```

```
//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(0.1f, 0.1f, 0.1f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
glUniform3fv(uniformColor, 1, glm::value_ptr(color));
Dado_M.RenderModel();
```

Al compararlo con el dado generado con código, se puede apreciar que son casi iguales, respetando principalmente la orientación de las caras de los animales.

Dado izquierdo -> 3dsMax. Dado derecho -> OpenGL.



Dado izquierdo -> OpenGL. Dado derecho -> 3dsMax.

2. Problemas presentados.

No se presentaron problemas a la hora de realizar el ejercicio.

3. Conclusión

a. Los ejercicios de la clase: Complejidad, explicación

Los ejercicios solicitados tuvieron una complejidad no tan elevada, ya que gracias a la buena explicación de como optimizar la imagen y obtener los vértices de esta, así como el proceso de texturizado en 3dsMax, se logró realizar los ejercicios sin contratiempos.

b. Comentarios generales: Faltó explicar a detalle, ir más lento en alguna explicación, otros comentarios y sugerencias.

La explicación general del ambiente de trabajo de GLIMP, la optimización de imágenes y el proceso de texturizado para modelos fue muy claro y preciso. Fue de gran ayuda el manual proporcionado sobre el texturizado en 3dsMax, ya que si te llegabas a perder en un paso, puedes ayudarte con el manual.