



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 N° 06

NOMBRE COMPLETO: Reyes Romero Luis Fernando

N° de Cuenta: 318155320

GRUPO DE LABORATORIO: 11

GRUPO DE TEORÍA: 06

SEMESTRE 2024-2

FECHA DE ENTREGA LÍMITE: 19 de marzo de 2024

CALIFICACIÓN: _____

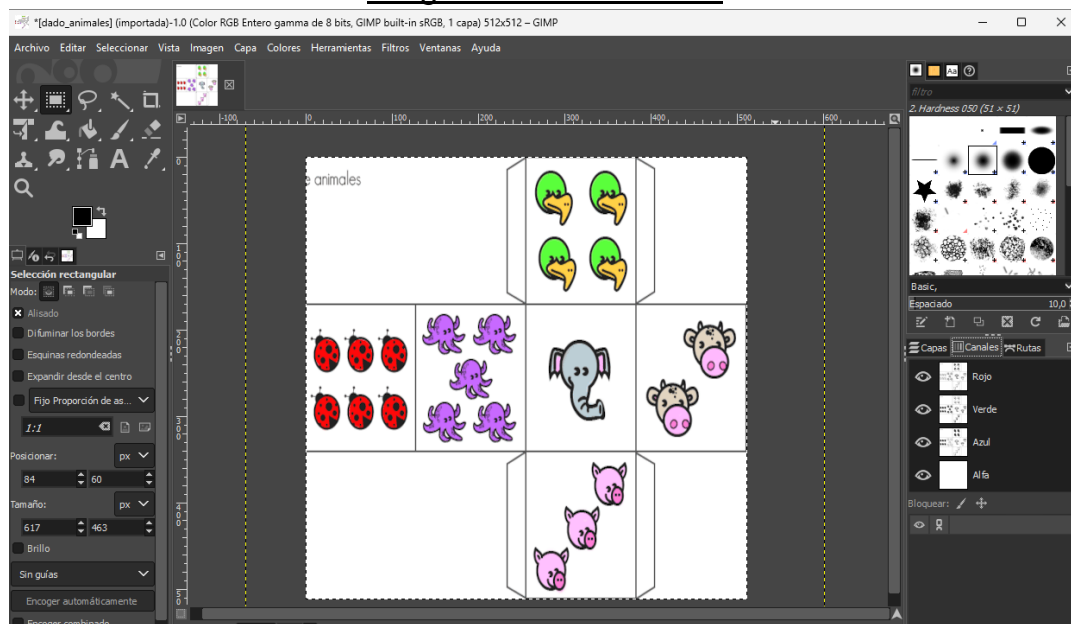
EJERCICIOS DE SESIÓN:

Actividades realizadas

- 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 la primera actividad de la práctica lo primero que tenemos que hacer es editar la imagen de los dados de los animales en nuestra herramienta GIMP, ya que con esta imagen vamos a realizar las dos actividades de la práctica lo que vamos a editar de la imagen será recortar los bordes que no necesitamos de la imagen. Además de cambiar el tamaño de la imagen para que se adapte a un tamaño de 2 a la n, después debemos de ver que la imagen este en formato RGB y después exportamos la imagen con la extensión .tga.

Imagen editada en GIMP



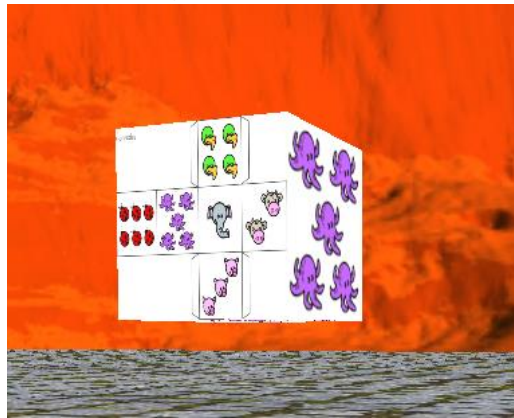
Después de tener la imagen nos vamos a nuestro código y lo primero que vamos a editar del código será la ruta de nuestra imagen recortada y para ello se modifica el código de la siguiente manera.

Ruta para la imagen

```
datoTexture = Texture("Textures/dado_animales_recortado.tga");  
datoTexture.LoadTextureA();
```

Una vez hecho eso ejecutamos el código para ver que es lo que hace.

Ejecución del programa con la imagen recortada

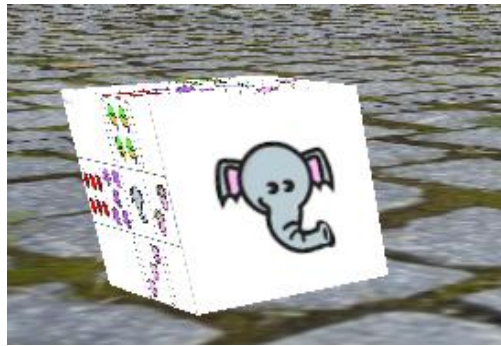


Podemos observar que las imágenes del dado están mal acomodadas y que en cada lado vemos la imagen que recortamos. Entonces, para poder ver un animal en cada cara del dado tenemos que modificar las coordenadas dentro de nuestro código. Como se menciona en la clase de laboratorio, el origen de la imagen se encuentra en la parte inferior izquierda, en ese lugar se encuentra nuestro punto (0,0) y la parte inferior derecha es el punto (0,1). Entonces si seguimos la lógica nuestros puntos superiores son el (0,1) y el (1,1) eso es muy importante saber, ya que al observar la imagen podemos observar que en el eje horizontal se parte en 4 vértices o 4 puntos entonces sí sabemos que nuestros puntos van de 0 a 1 cada vértice se encuentra 0.25 de distancia entre cada uno de ellos en el eje horizontal. En cambio, en el eje vertical solo hay 3 vértices o 3 puntos entonces la separación de los vértices son de 0.33. Cabe mencionar que como ya estamos hablando de texturizado va a ver nuevas coordenadas que serán S y T respectivamente donde S será nuestro eje horizontal y T será nuestro eje vertical de texturizado. Siguiendo esta lógica podemos ver que el elefante es nuestro animal que esta en la cara de frente y que el primer vértice se encuentra en (0.5, 0.33), el segundo (0.75, 0.33), el tercero (0.75, 0.66) y el cuarto (0.5, 0.33) entonces si ponemos estas coordenadas en nuestro código tenemos que ver en la cara frontal al elefante.

Modificación del código

```
GLfloat cubo_vertices[] = {  
    // front  
    //x      y      z      S      T      NX      NY      NZ      //  
    -0.5f, -0.5f, 0.5f, 0.50f, 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, 0.66f, 0.0f, 0.0f, -1.0f, //2  
    -0.5f, 0.5f, 0.5f, 0.50f, 0.66f, 0.0f, 0.0f, -1.0f, //3  
    // back  
    -0.5f, -0.5f, -0.5f, 0.50f, 0.34f, 0.0f, 0.0f, 1.0f, //4  
    0.5f, -0.5f, -0.5f, 0.74f, 0.34f, 0.0f, 0.0f, 1.0f, //5  
    0.5f, 0.5f, -0.5f, 0.74f, 0.66f, 0.0f, 0.0f, 1.0f, //6  
    -0.5f, 0.5f, -0.5f, 0.50f, 0.66f, 0.0f, 0.0f, 1.0f, //7  
    // left  
    -0.5f, -0.5f, 0.5f, 0.50f, 0.34f, 1.0f, 0.0f, 0.0f, //8  
    -0.5f, 0.5f, 0.5f, 0.50f, 0.66f, 1.0f, 0.0f, 0.0f, //9  
    -0.5f, -0.5f, -0.5f, 0.50f, 0.34f, 1.0f, 0.0f, 0.0f, //10  
    -0.5f, 0.5f, -0.5f, 0.50f, 0.66f, 1.0f, 0.0f, 0.0f, //11  
    // right  
    0.5f, -0.5f, 0.5f, 0.74f, 0.34f, 1.0f, 0.0f, 0.0f, //12  
    0.5f, 0.5f, 0.5f, 0.74f, 0.66f, 1.0f, 0.0f, 0.0f, //13  
    0.5f, -0.5f, -0.5f, 0.74f, 0.34f, 1.0f, 0.0f, 0.0f, //14  
    0.5f, 0.5f, -0.5f, 0.74f, 0.66f, 1.0f, 0.0f, 0.0f, //15  
};
```

Ejecución del programa



Efectivamente podemos ver el animal que nosotros escogimos, siguiendo la lógica que se explico anteriormente se modifica el código para cada una de las coordenadas del dibujo del animal. Cabe mencionar que es muy importante tener en cuenta en que coordenadas se esta dibujando nuestro animal porque podemos poner mal las coordenadas provocando un acomodo mal de la figura.

Modificación del programa

```
// right
//x      y      z      S      T
0.5f, -0.5f, 0.5f, 0.75f, 0.34f, -1.0f, 0.0f, 0.0f,
0.5f, -0.5f, -0.5f, 1.00f, 0.34f, -1.0f, 0.0f, 0.0f,
0.5f, 0.5f, -0.5f, 1.00f, 0.66f, -1.0f, 0.0f, 0.0f,
0.5f, 0.5f, 0.5f, 0.75f, 0.66f, -1.0f, 0.0f, 0.0f,
// back
-0.5f, -0.5f, -0.5f, 0.25f, 0.34f, 0.0f, 0.0f, 1.0f,
0.5f, -0.5f, -0.5f, 0.00f, 0.34f, 0.0f, 0.0f, 1.0f,
0.5f, 0.5f, -0.5f, 0.00f, 0.66f, 0.0f, 0.0f, 1.0f,
-0.5f, 0.5f, -0.5f, 0.25f, 0.66f, 0.0f, 0.0f, 1.0f,
// left
//x      y      z      S      T
-0.5f, -0.5f, -0.5f, 0.25f, 0.34f, 1.0f, 0.0f, 0.0f,
-0.5f, -0.5f, 0.5f, 0.50f, 0.34f, 1.0f, 0.0f, 0.0f,
-0.5f, 0.5f, 0.5f, 0.50f, 0.66f, 1.0f, 0.0f, 0.0f,
-0.5f, 0.5f, -0.5f, 0.25f, 0.66f, 1.0f, 0.0f, 0.0f,
// bottom
//x      y      z      S      T
-0.5f, -0.5f, 0.5f, 0.50f, 0.33f, 0.0f, 1.0f, 0.0f,
0.5f, -0.5f, 0.5f, 0.74f, 0.33f, 0.0f, 1.0f, 0.0f,
0.5f, -0.5f, -0.5f, 0.74f, 0.00f, 0.0f, 1.0f, 0.0f,
-0.5f, -0.5f, -0.5f, 0.50f, 0.00f, 0.0f, 1.0f, 0.0f,
```

```

// bottom
//x      y      z      S      T
-0.5f, -0.5f,  0.5f,  0.50f,  0.33f,  0.0f,  1.0f,  0.0f,
 0.5f, -0.5f,  0.5f,  0.74f,  0.33f,  0.0f,  1.0f,  0.0f,
 0.5f, -0.5f, -0.5f,  0.74f,  0.00f,  0.0f,  1.0f,  0.0f,
-0.5f, -0.5f, -0.5f,  0.50f,  0.00f,  0.0f,  1.0f,  0.0f,

//UP
//x      y      z      S      T
-0.5f,  0.5f,  0.5f,  0.50f,  0.66f,  0.0f, -1.0f,  0.0f,
 0.5f,  0.5f,  0.5f,  0.75f,  0.66f,  0.0f, -1.0f,  0.0f,
 0.5f,  0.5f, -0.5f,  0.75f,  1.00f,  0.0f, -1.0f,  0.0f,
-0.5f,  0.5f, -0.5f,  0.50f,  1.00f,  0.0f, -1.0f,  0.0f,

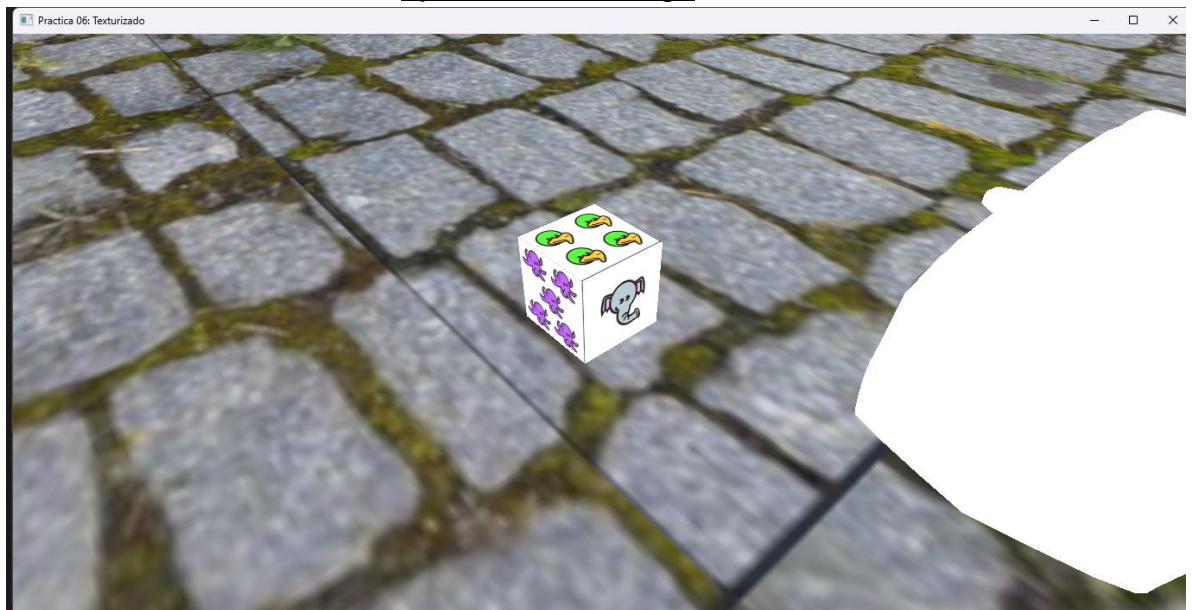
};

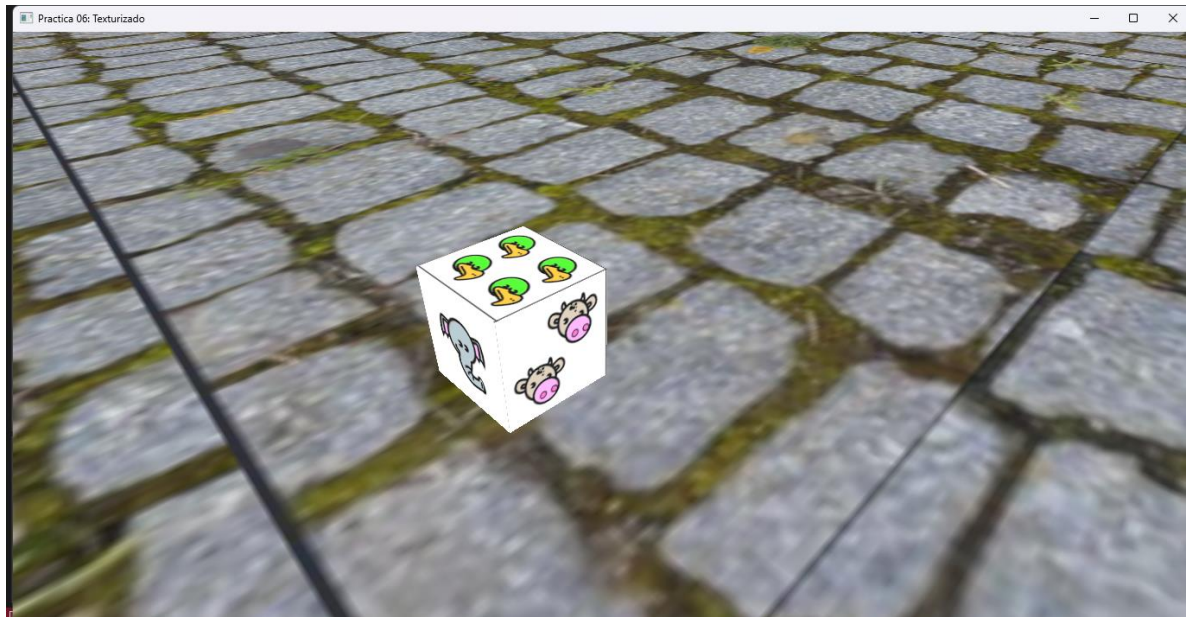
Mesh* dado = new Mesh();
dado->CreateMesh(cubo_vertices, cubo_indices, 192, 36);
meshList.push_back(dado);

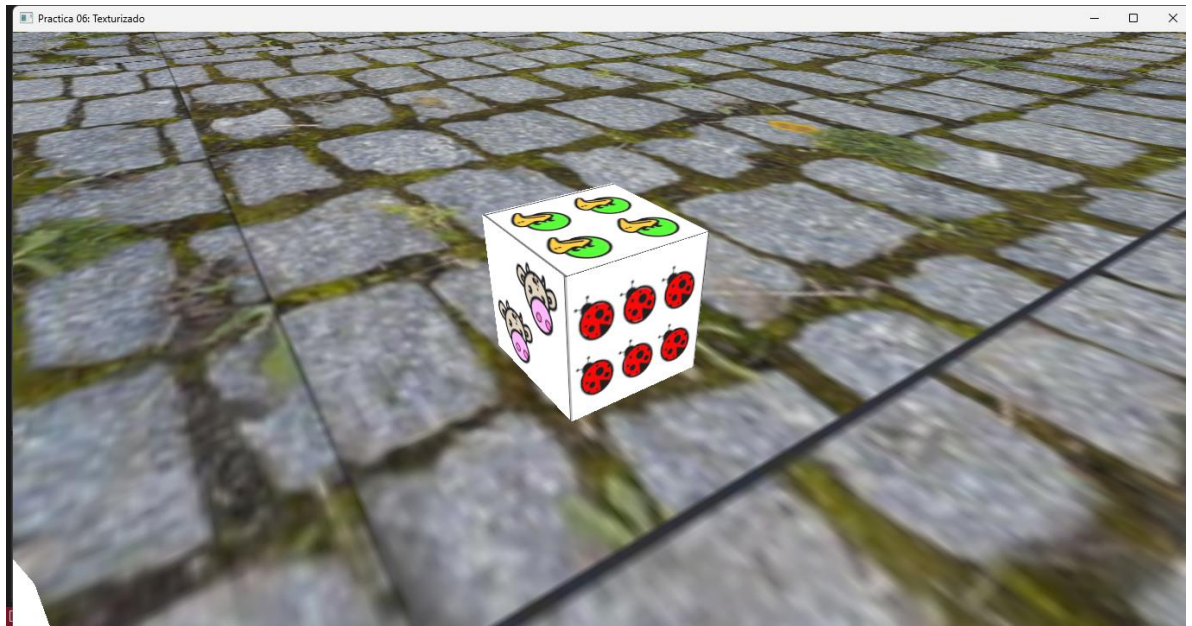
```

Finalmente ejecutamos el programa y observamos que nuestro cubo esta de forma correcta.

Ejecución del código

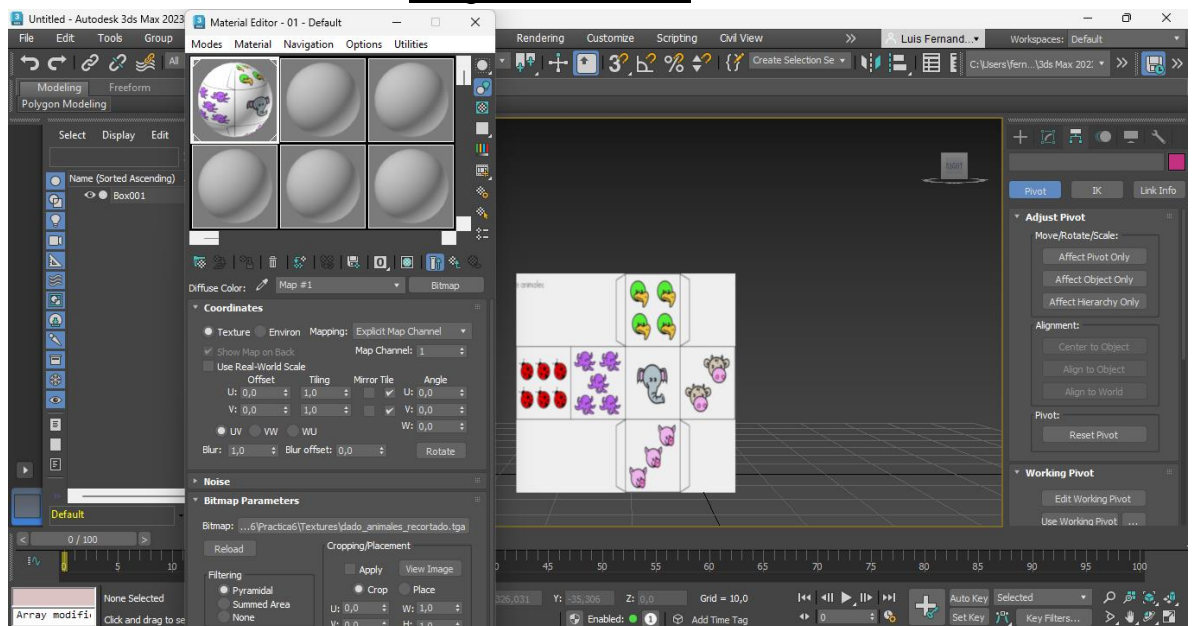






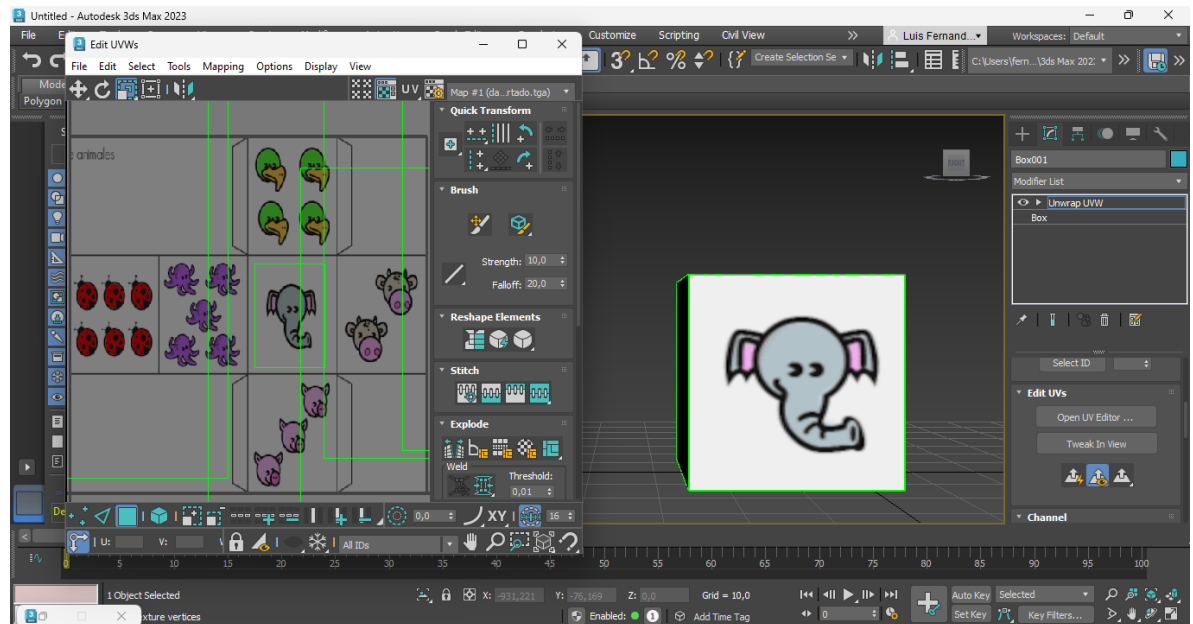
Para la segunda actividad hacemos uso del programa de 3ds Max, para este ejercicio no hay mucha explicación, ya que para poder acomodar las imágenes de nuestro cubo se siguen los pasos del manual que el profesor nos dio como material a consultar. Una vez que se siguieran los pasos del manual vamos a observar que cada cara del cubo se muestra nuestra imagen recortada como en la actividad anterior.

Imagen de 3ds Max



Entonces una vez que tengamos nuestro cubo de esta forma lo que vamos a hacer es escalar cada una de las caras para que muestre los animales en orden. A continuación, se muestra la imagen dentro del programa con las caras ya escaladas.

Caras escaladas



Ya que tengamos acomodado nuestro cubo con las caras de animales correctamente vamos a exportar nuestro cubo con la extensión .obj. Cabe mencionar que debemos de verificar que nuestro cubo este hecho a partir de triángulos para que en OpenGL se pueda ver el cubo. Una vez creado el objeto y exportarlo a la carpeta de modelos dentro de nuestro proyecto creamos una variable de tipo modelo dentro de nuestro código para que se pueda abrir e imprimir el objeto con la textura deseada.

Modificaciones de código

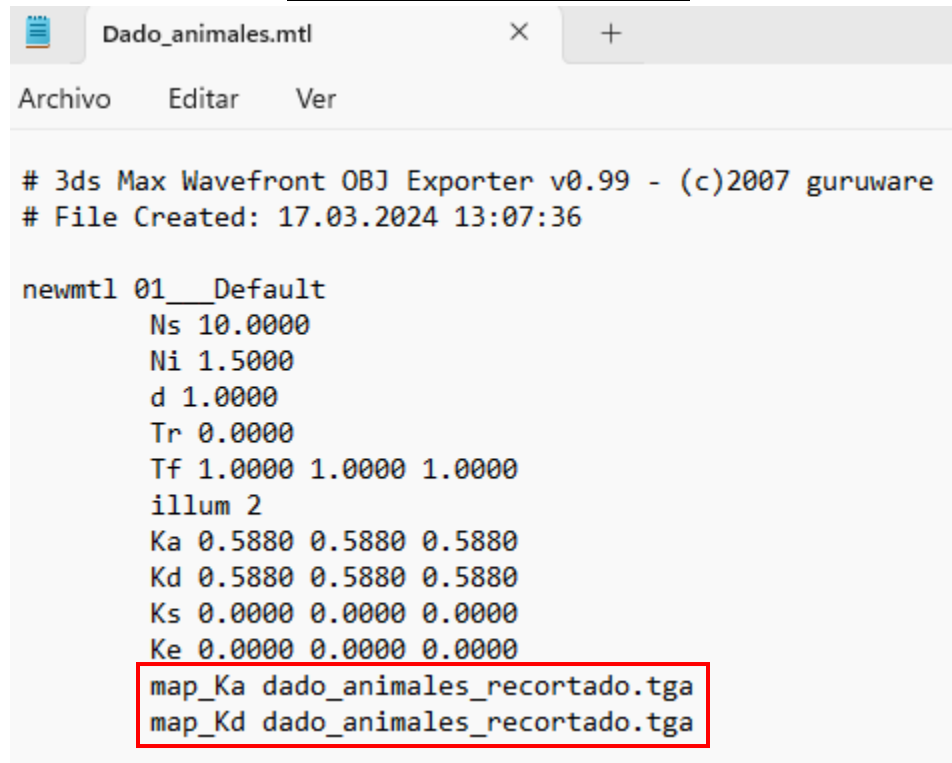
```
Dado_M = Model();
Dado_M.LoadModel("Models/Dado_animales.obj");
```

```
//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(0.03f, 0.03f, 0.03f));
//model = glm::rotate(model, -90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Dado_M.RenderModel();
```

Finalmente ejecutamos el programa y vemos que efectivamente se observan ambos cubos. Aquí hay que mencionar que en nuestro archivo .mtl se tiene

que borrar una parte del archivo que corresponde a las dos últimas líneas del archivo se tiene que eliminar la ruta que viene hasta el nombre de nuestro archivo de origen esto se hace para que se pueda dibujar el objeto de forma correcta. A continuación, se muestra la modificación.

Modificación del archivo .mtl



```
# 3ds Max Wavefront OBJ Exporter v0.99 - (c)2007 guruware
# File Created: 17.03.2024 13:07:36

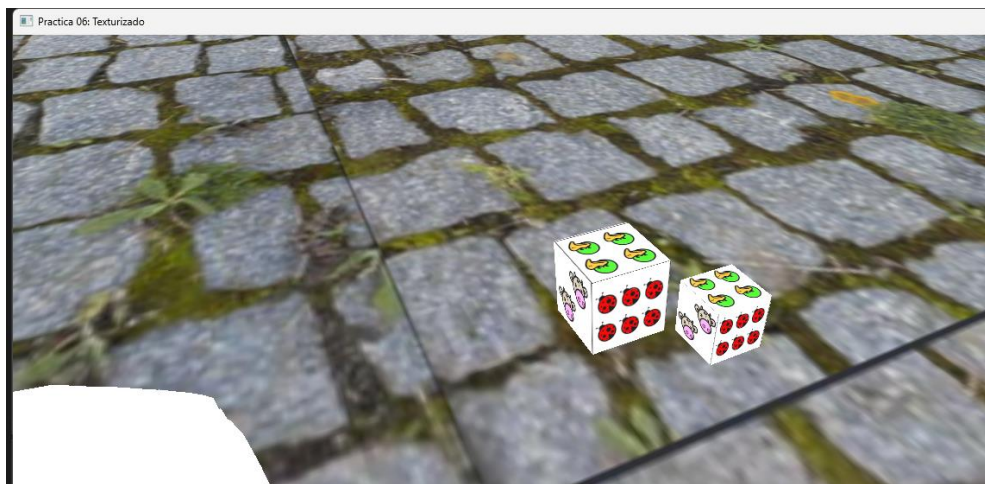
newmtl 01__Default
  Ns 10.0000
  Ni 1.5000
  d 1.0000
  Tr 0.0000
  Tf 1.0000 1.0000 1.0000
  illum 2
  Ka 0.5880 0.5880 0.5880
  Kd 0.5880 0.5880 0.5880
  Ks 0.0000 0.0000 0.0000
  Ke 0.0000 0.0000 0.0000
  map_Ka dado_animales_recortado.tga
  map_Kd dado_animales_recortado.tga
```

Ejecución del programa



Ya solo para finalizar si queremos ver los cubos de la misma forma, ya que en el programa de 3ds Max se creo el cubo con las caras acomodadas de forma diferente se descomenta la rotación y ejecutamos otra vez el programa.

Ejecución del programa con la rotación del cubo





De esta forma ya podemos ver ambos cubos en la misma dirección y podemos que cada cara de los cubos está acomodada de la misma forma.

Problemas presentados.

No se presentó ningún problema, ya que gracias a la explicación en la clase de laboratorio se explicó muy bien la manera de hacer las actividades. Además de que los videos que nos proporcionó el profesor son de gran ayuda.

Conclusión

Gracias a este ejercicio se comprendió de mejor manera como darle textura a un objeto. De igual forma, creo que este ejercicio está muy bien para ser nuestro primer ejercicio de texturizado. Además de que en este ejercicio se explican las dos formas de texturizar un objeto, la primera es texturizar el objeto mediante puro código tomando como referencia las coordenadas de una imagen y la otra que es por medio de 3ds Max que a mi parecer se me hace más sencillo y fácil de hacer.