NORMAL MAPPING

Partimos del ejemplo **fixed_camera_nmapping.**
El objetivo es crear un material básico de normal mapping con iluminación difusa.

- Convertimos el mapa de normales data/wavy.jpg en formato dds->dxt1 usando GIMP

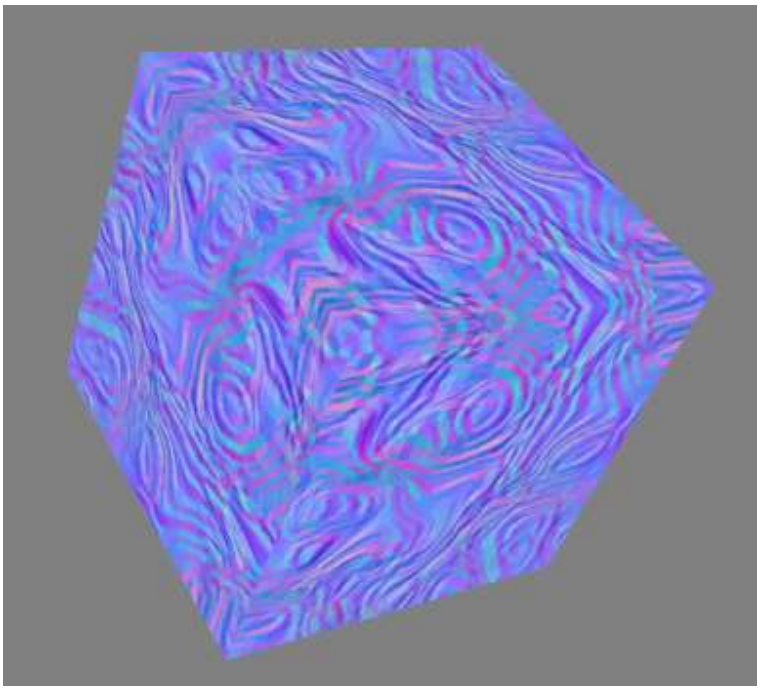
-Cargamos el dds en una nueva textura; nos aseguramos de que antes de renderizar activamos las dos unidades de sampling y de que hacemos el binding de las dos texturas

-Para poder usar los samplers tenemos que configurarlos (en inicializacion)
En el fragment shader declaramos dos shaders:

```
uniform sampler2D tex_image;
uniform sampler2D tex_normals;
```

Y desde el programa principal:

```
glUseProgram(gShaderProgram);
unsigned int sampler_diffuse = glGetUniformLocation(gShaderProgram, "tex_diffuse");
unsigned int sampler_normals = glGetUniformLocation(gShaderProgram, "tex_normals");
glUniform1i(sampler_diffuse, 0); // Texture unit 0: textura color para luz difusa
glUniform1i(sampler_normals, 1); // Texture unit 1: normal map
```



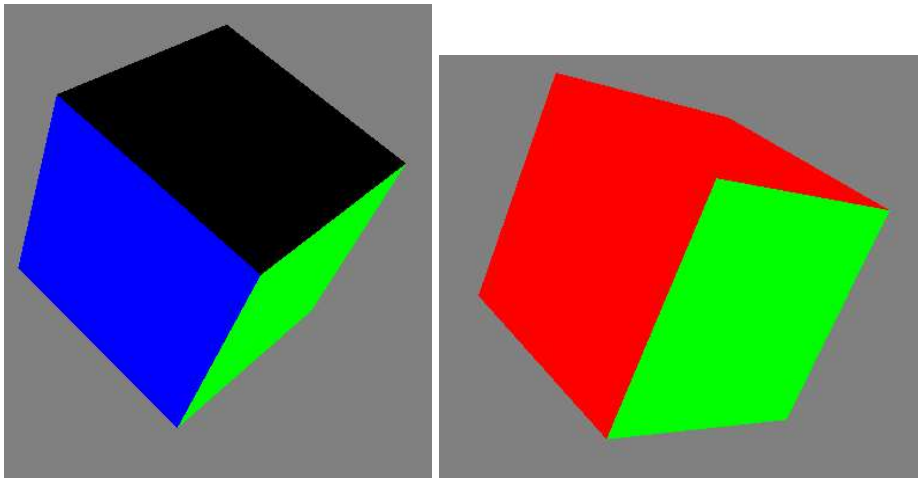
-Creamos un nuevo tipo de vértice que tendrá dos componentes nuevos (ambos tipo **vec3**): Normal y Tangent. Obviaremos Bitangent ya que se puede deducir desde el fragment shader.

-En tiempo de inicialización, con un algoritmo propio configuraremos los nuevos vértices a partir de los antiguos, por grupos de cuatro.
Esto es una simplificación del algoritmo completo de maya, que tiene vértices compartidos. Nuestro cubo no los tiene así que podemos resolverlo a nivel de cara.
Para cada cara (grupo de 4 vertices):
1.- Obtenemos la referencia a 3 puntos cualesquiera de la cara, respetando el sentido de giro (usando los índices)
2.- Calculamos la normal y la tangente (**T,N**) para esa cara:

```
// Sacamos vectores de un punto del quad respecto a 2 vecinos (cualesquiera)
glm::vec3 edge1 = gMesh[i1].pos - gMesh[i0].pos;
glm::vec3 edge2 = gMesh[i2].pos - gMesh[i0].pos;
glm::vec2 deltaUV1 = gMesh[i1].uv - gMesh[i0].uv;
glm::vec2 deltaUV2 = gMesh[i2].uv - gMesh[i0].uv;
// Obtenemos la normal
glm::vec3 N = glm::normalize(glm::cross(edge1, edge2));
// Calculo de la tangente y bitangente
float f = 1.0f / (deltaUV1.x * deltaUV2.y - deltaUV2.x * deltaUV1.y);
glm::vec3 T(0.0f,0.0f,0.0f);
T.x = f * (deltaUV2.y * edge1.x - deltaUV1.y * edge2.x);
T.y = f * (deltaUV2.y * edge1.y - deltaUV1.y * edge2.y);
T.z = f * (deltaUV2.y * edge1.z - deltaUV1.y * edge2.z);
T = glm::normalize(T);
```

3.- Copiamos posición y vu de los 4 vértices originales, y añadimos los vectores **T, y N** que son los mismo para los 4 vértices de la cara.

-Adaptar la carga de la malla al nuevo tipo configurando y añadiendo las funciones **glVertexAttribPointer.** Pasamos los vectores **N y T** por el vertex shader y nos aseguramos por medio de debugger y/o visualización de que las normales y tangentes tienen buen aspecto



- Transformar los vectores **T,N** en el vertex shader llevándolos a orientación de mundo

```
mat3 model3x3 = mat3(model);
mT = model3x3 * T;
mN = model3x3 * N;
```

-Ahora podemos operar el normal map en el fragment shader. Para ello necesitamos la matriz de transformación del espacio tangencial **TBN.** Puesto que nos falta **B** lo deducimos a partir de **T,N** y construimos la matriz.

```
vec3 _mB = cross(mN, mT);
mat3 TBN = mat3(mT,_mB,mN);
```

-Antes de operar la normal, necesitamos re-escalarla a vectores unitarios con signo

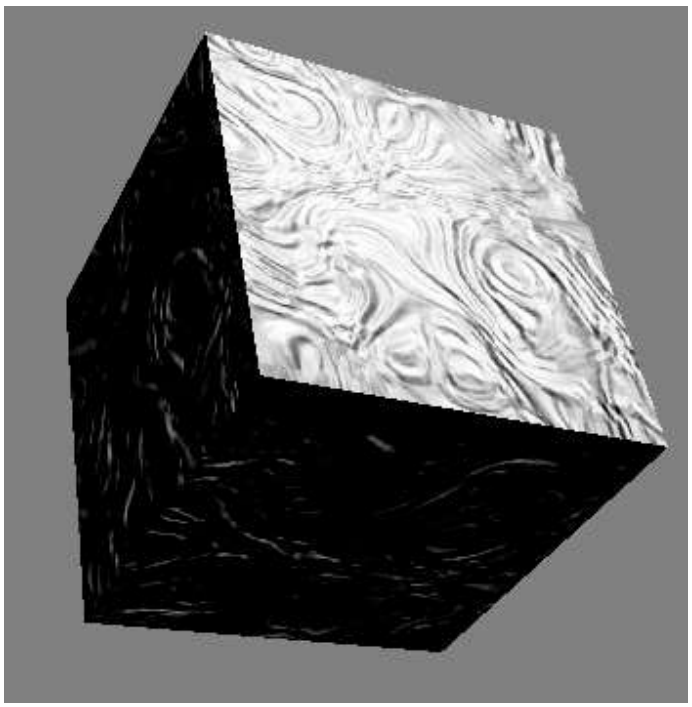
```
N = N * 2.0 - 1.0;
N = normalize(N);
```

La normalización es necesaria puesto que los formatos de textura carecen de precisión

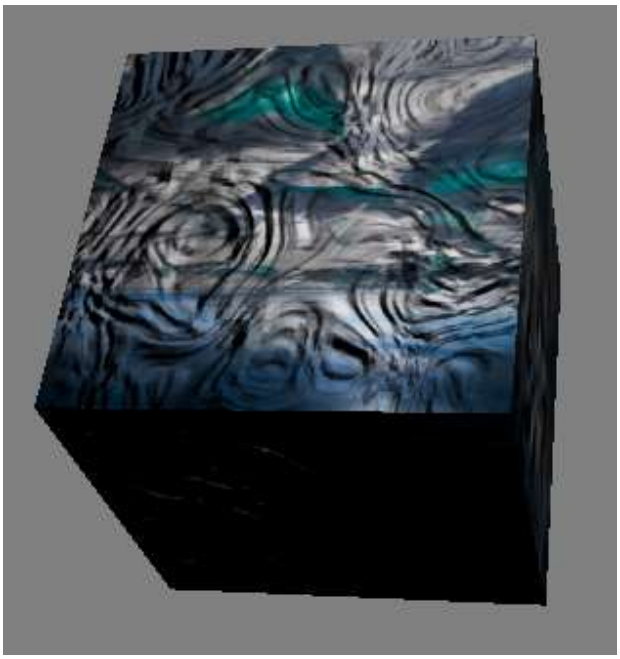
-Llevamos la normal a espacio de mundo

```
N = TBN * N
```

-Obtenemos la luz difusa usando esta nueva normal



- Por último, multiplicamos la iluminación por el color del texel para obtener:



OBJETIVO EXTRA

La compresión DXT1/BC1 es demasiado débil para normal maps. Podéis ver en la textura convertida que el resultado es excesivamente “blocky”.

La forma más común de usar la compresión BC consiste en eliminar el canal Z de la textura (el azul). Esto mejora la calidad de la compresión notablemente.

Una vez cargada la textura, el fragment shader puede deducir la Z ya que

$$x^2+y^2+z^2 = 1$$

Para borrar el canal Z, podeis usar en GIMP la funcion **Color/Components/Decompose**, borrar el canal azul, y luego usar **Color/Componenet/Recompose**