

Informática Gráfica

Texturas

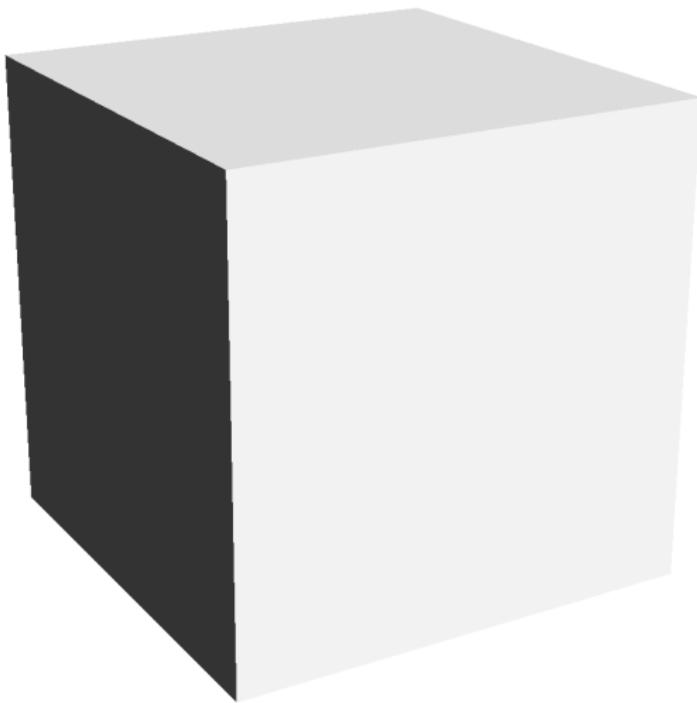
Juan Carlos Torres
Grupos C y D

Dpt. Lenguajes y Sistemas Informáticos
ETSI Informática y de Telecomunicación
Universidad de Granada

Curso 2024-25

Visualización sin textura

- Material uniforme asociado a la superficie.



Texturas

Visualización con textura

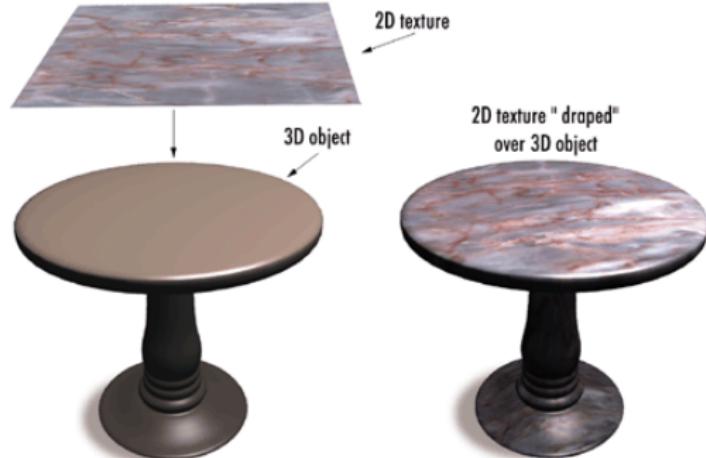
- Color no uniforme y afectado por iluminación.



Texturas

Representación de texturas

- La textura suele ser una imagen 2D.
- Se aplica a la superficie para modular el color de esta.

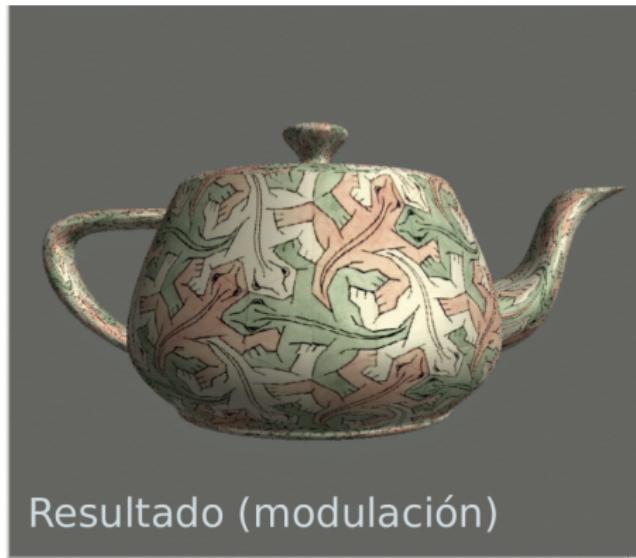


http://cf.ycdn.net/1.0.1.49/images/computer/_TEXMAP.GIF

Aplicación de texturas

Las texturas se pueden combinar con el cálculo de iluminación:

- Evaluación del MIL con reflectividades blancas (izq. abajo)
- Uso directo de colores de la textura (izq. arriba)
- Evaluación del MIL con reflectividades obtenidas de la textura (der.)

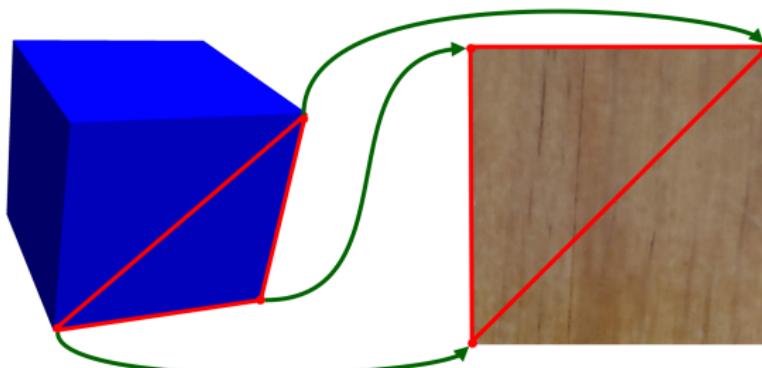


Resultado (modulación)

Aplicación de textura

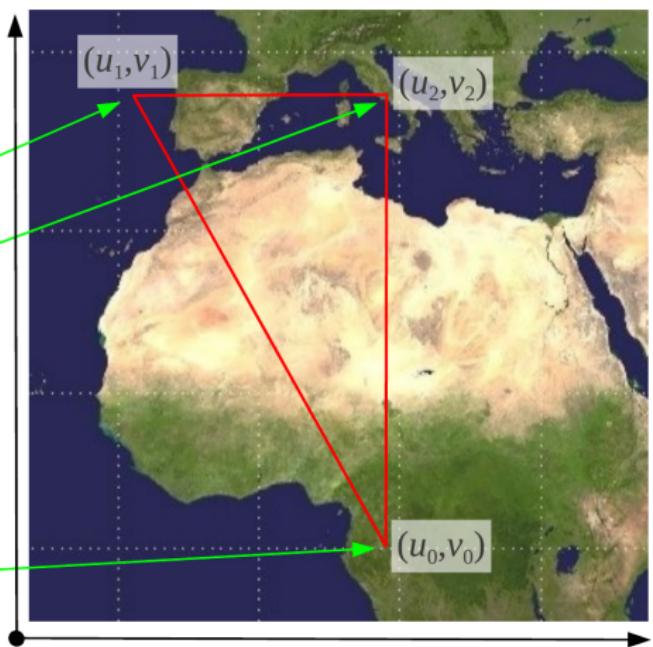
Coordenadas de textura

- Para aplicar la textura se hace corresponder cada vértice de la malla con un punto de la textura.
- La correspondencia se establece asociando al vértice las coordenadas del punto de la textura (coordenadas de textura).
- Las coordenadas de textura se dan normalizadas.



Coordenadas de textura

A cada vértice (i) de un triángulo del modelo se le debe asignar unas coordenadas de textura (u_i, v_i) , que le hacen corresponder un punto de la textura.

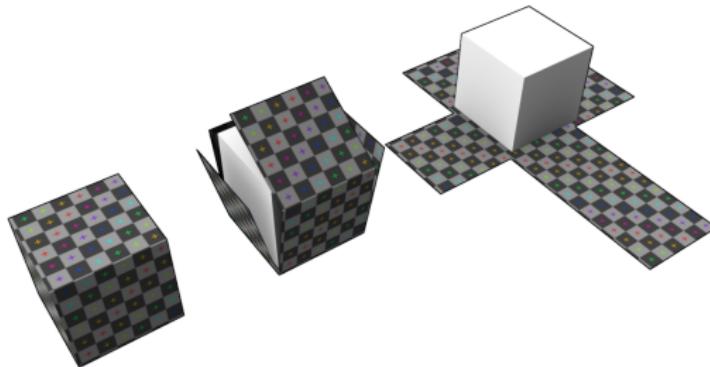


Aplicación de texturas

Las coordenadas de textura se puede obtener desplegando las caras del objeto

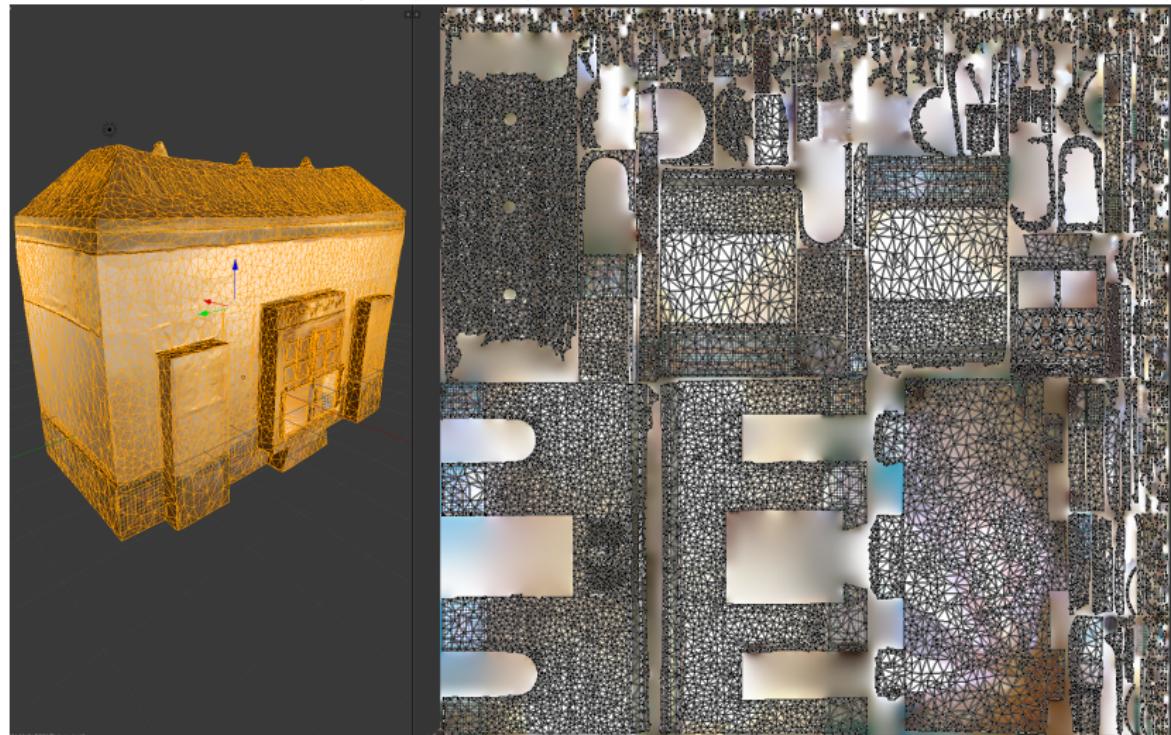
Al desplegar el modelo pueden aparecer:

- Islas
- Aristas abiertas formando costuras (seams) en el modelo
- Zonas no utilizadas en la textura



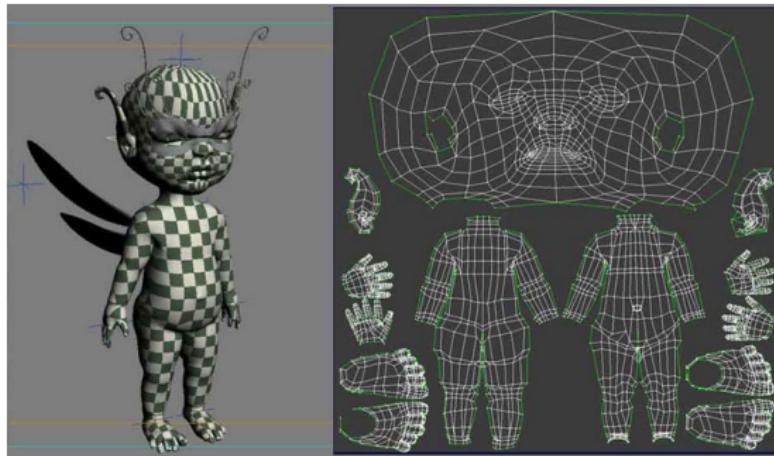
Ejemplo de textura

Las herramientas de modelado 3D contienen funciones para generar coordenadas de textura (ejemplo de parametrización automática con Blender).



Ejemplos de texturas

- Para que no se aprecien las costuras es necesario que la imagen de la textura sea continua a través de ellas.
- Si la parametrización no es uniforme distintas zonas del modelo tendrán distinto nivel de detalle



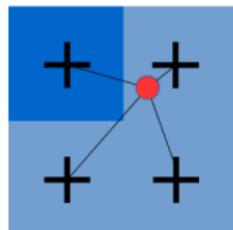
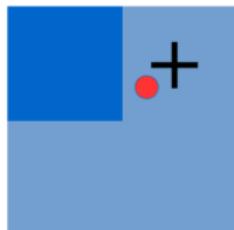
Consulta de texels en texturas de imagen

Cada texel tiene asociada un pequeño rectángulo contenido en $[0, 1]^2$. El texel en la columna i , fila j tendrá un área con centro en el punto (c_i, d_j)

La consulta del color de la textura en un punto (u, v) :

- **más cercano:** usar el color del texel cuyo centro sea más cercano a la posición (u, v) , es equivalente a seleccionar el texel cuya área contiene a (u, v) .
- **interpolación** realizar un interpolación (bilineal) entre los colores de los cuatro texels con centros más cercanos al punto (u, v) .

las diferencias entre ambos métodos son visibles cuando la proyección en la ventana de un texel ocupa varios pixels.

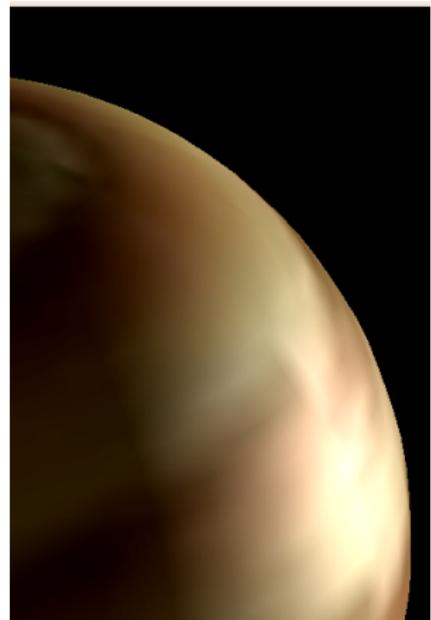


Consulta de texels en texturas de imagen

Textura de baja resolución, vista de cerca, que se visualiza usando los dos métodos:



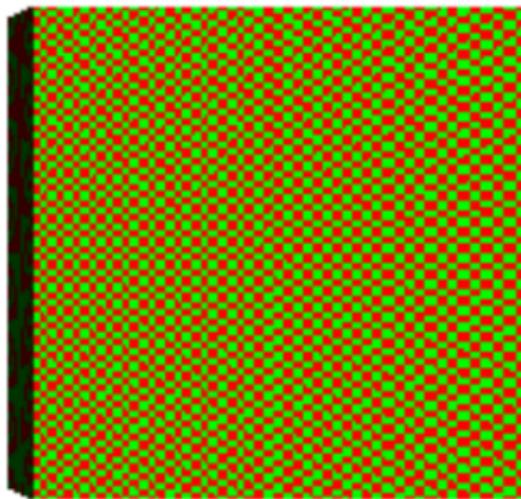
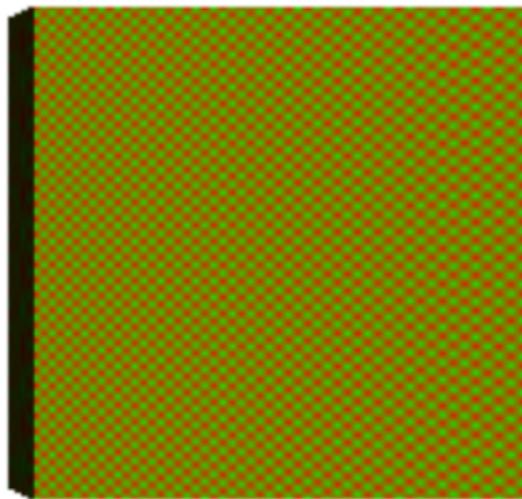
más cercano



interpolación bilineal

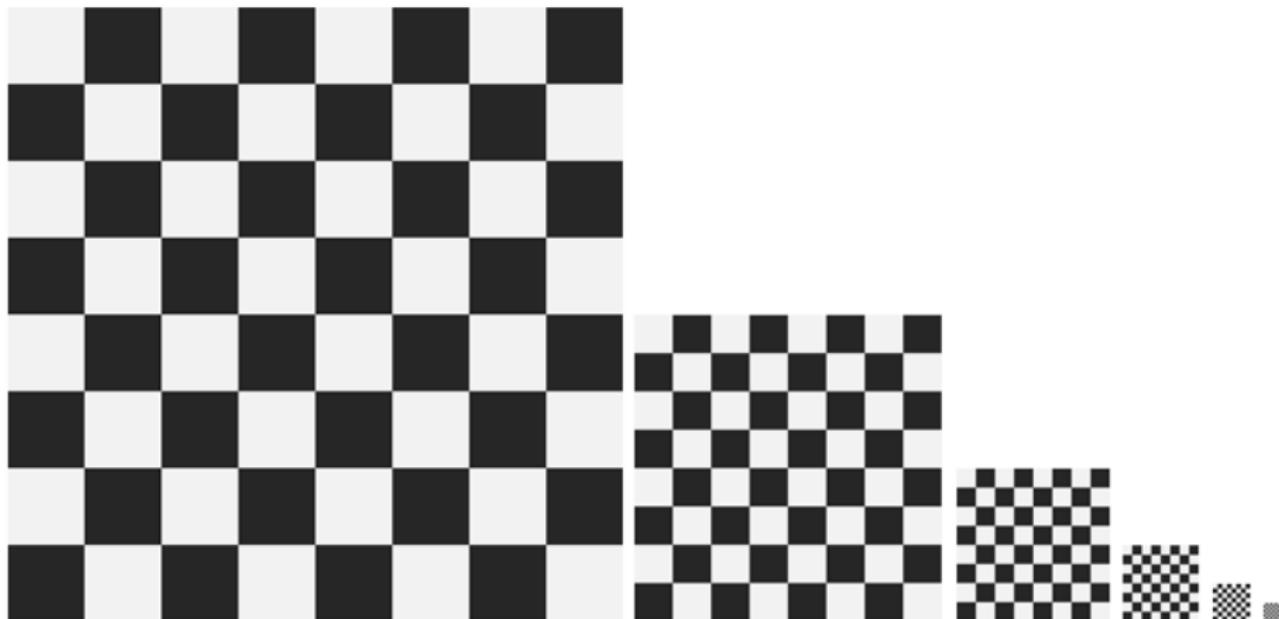
Consulta de texels en texturas de imagen

De forma análoga cuando en cada fragmento se proyectan varios texels se puede usar solo el mas cercano o hacer una interpolación bilineal :



MipMaps

Un mipmap es una serie de texturas preprocesadas en tamaños decrecientes. Se crean para optimizar la aplicación de la textura en objetos lejanos.



Coordenadas de textura

Las coordenadas de textura se definen en el intervalo $(0, 1)$.

Cuando se salen de este intervalo se pueden:

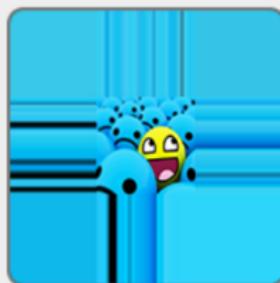
- Truncar.
- Repetir.



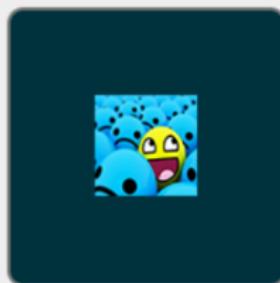
GL_REPEAT



GL_MIRRORED_REPEAT



GL_CLAMP_TO_EDGE



GL_CLAMP_TO_BORDER

<https://learnopengl.com/Getting-started/Textures>

Mallas con textura

Para asociar texturas a las mallas de polígonos es necesario guardar las coordenadas de textura de los vértices.

Cada vértice puede tener asociadas coordenadas de textura diferentes en cada polígono.

La estructura de datos debe permitir asociar mas de una coordenada de textura por vértice.



Parametrización

Asignación explícita o procedural

La asignación de coordenadas de textura se puede hacer de dos formas:

- **Asignación explícita:** La coordenadas de textura forman parte de la definición del modelo de escena $(v_0, u_0), (v_1, u_1), \dots, (u_{n-1}, v_{n-1})$ Que se habrán calculado/asignado al crear el objeto:
 - se puede hacer manualmente en objetos sencillos, o bien
 - de forma asistida usando algoritmos de parametrización (p.ej. MeshLab o Blender).
- **Asignación procedural:** Se define una función f que se implementa como un subprograma CoordText (\mathbf{p}) que se invoca para calcular las coordenadas de textura (acepta un punto \mathbf{p} como parámetro y devuelve el par (u, v) con las coords. de textura de \mathbf{p}).

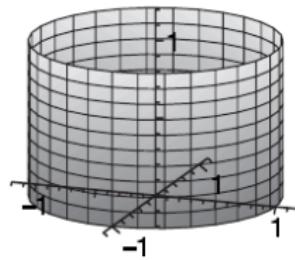
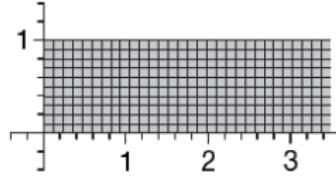
Parametrización procedural

Procedimiento

- Envolver el objeto con una superficie paramétrica.
- Asignar como coordenadas textura la del punto correspondiente en la superficie.
- La parametrización es la inversa de la función paramétrica que define la superficie.

$$\text{parameterization: } f(u, v) = (\cos u, \sin u, v)$$

$$\text{inverse: } f^{-1}(x, y, z) = (\arccos x, z)$$



Funciones para asignación procedural

Los tipos de funciones f más frecuentes son:

- **Funciones lineales** (proyección en un plano): el punto $\mathbf{p} = (x, y, z)$ se proyecta sobre un plano y se expresa como un par (x', y') de coordenadas en dicho plano, que se interpretan como coordenadas de textura.
- **Coordenadas polares** (proyección en una esfera): el punto \mathbf{p} se expresa en coordenadas polares como una terna (α, β, r) , los valores u y v se obtienen de α y β .
- **Coordenadas cilíndricas** (proyección en un cilindro): el punto \mathbf{p} se expresa en coordenadas cilíndricas como una terna (α, y, r) , los valores u y v se obtienen de α e y .
- **Coordenadas paramétricas**: se pueden usar si la malla aproxima una superficie paramétrica (p.ej. la tetera, hecha de superficies paramétricas tipo B-spline)

Ejemplo de coordenadas esféricas

Vemos un esquema de la proyección (izq.) y el resultado en varios objetos (der.)

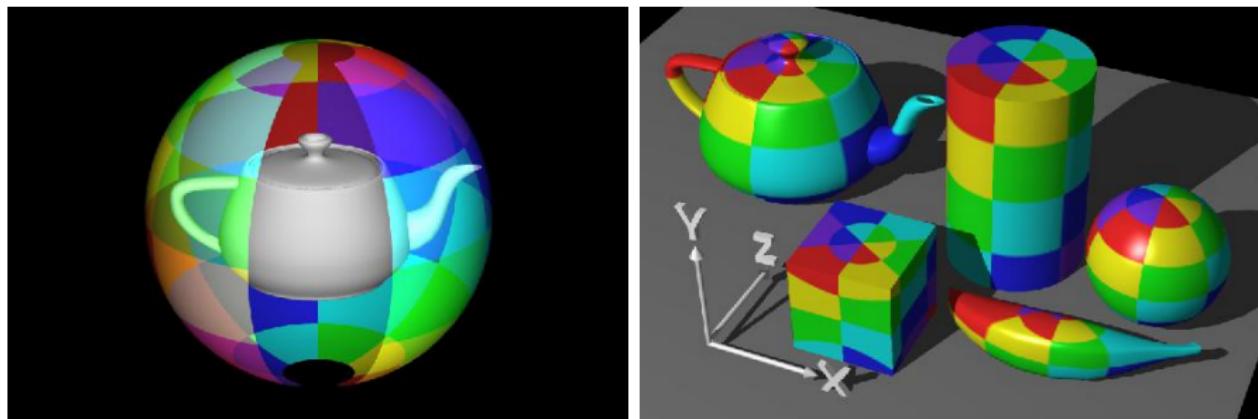
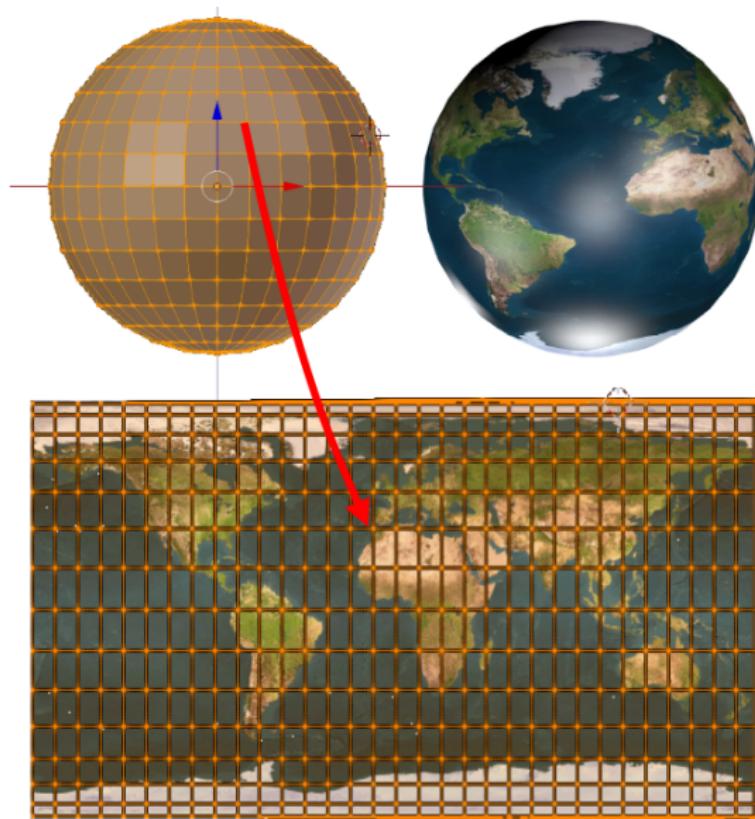


Imagen de Rosalee Wolfe, en [Teaching Texture Mapping Visually](#)
de SIGGRAPH 97 Education Slide Set.

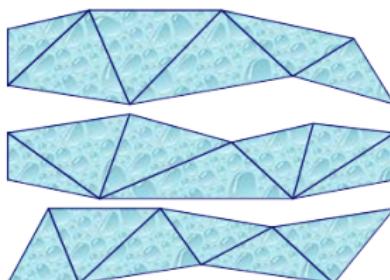
Modelo de la tierra



Ejemplo de método de parametrización trivial

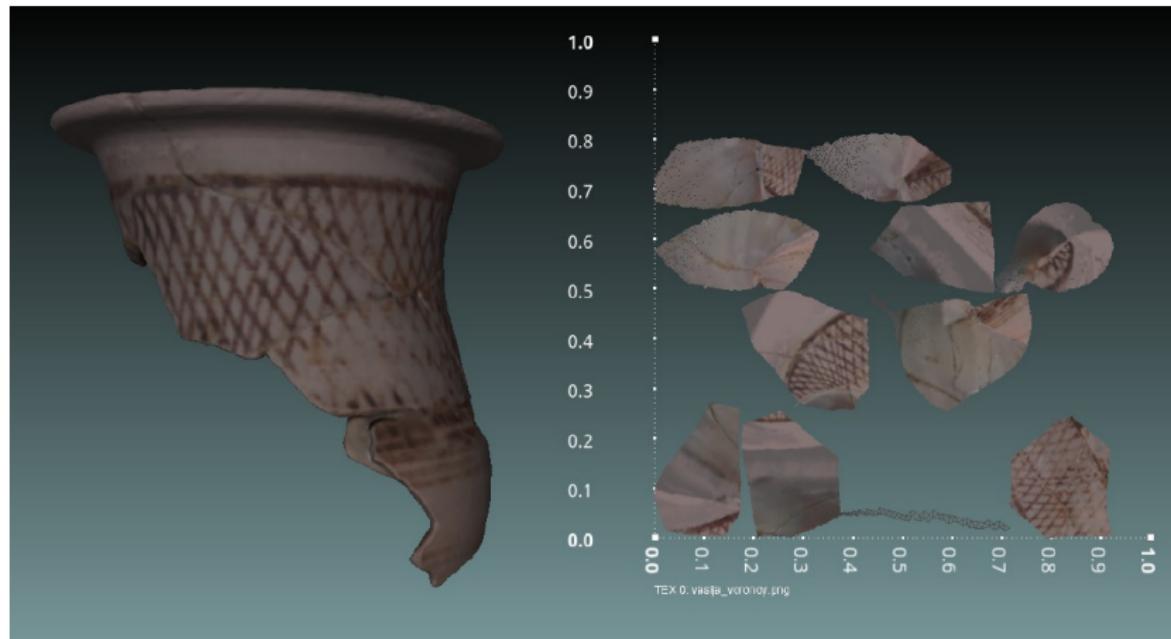
Se puede parametrizar representando cada triángulo de forma independiente o desplegando cintas de triángulos.

- Genera muchas costuras (todas las aristas).
- Genera muchos espacios perdidos (aprox. 50 %).
- Genera muchas islas (una por triángulo).



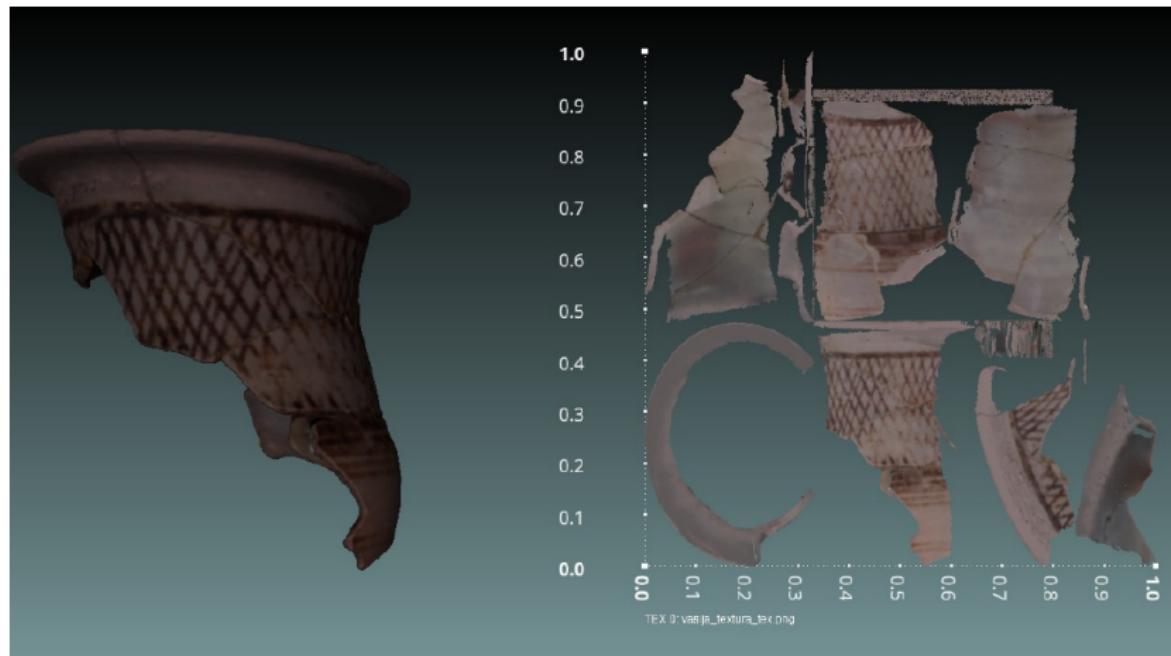
Parametrización con MeshLab

MeshLab dispone de un filtro para generar coordenadas de textura de mallas de polígonos: “Parametrization; Voronoi Atlas” que se encuentra en el grupo de filtros “Texture”.



Parametrización con Blender

Blender puede parametrizar usando la función “UV Unwrap” (mas información en Manual de Blender y en Video explicativo).



Texturas 2D en OpenGL

Activación y desactivación

Las ordenes **glEnable** y **glDisable** se pueden usar para activar o desactivar toda la funcionalidad de OpenGL relacionada con las texturas, que se encuentra inicialmente desactivada:

```
glEnable( GL_TEXTURE_2D ) ; // habilita texturas  
glDisable( GL_TEXTURE_2D ) ; // deshabilita texturas
```

Cuando se habilitan las texturas hay una textura activa en cada momento, que se consulta cada vez que un polígono se proyecta en un pixel, antes de calcular el color de dicho pixel:

- con iluminación desactivada, el color de la textura sustituye al especificado con **glColor**
- con iluminación activada, el color de la textura sustituye a las reflectividades del material (usualmente a la difusa y la ambiental).

todas las operaciones de texturas requieren que esta funcionalidad esté activada.

Identificadores de textura. Creación

OpenGL puede gestionar más de una textura a la vez. Para diferenciarlas usa un valor entero único para cada una de ellas, que se denomina **identificador de textura** (*texture name*) (de tipo **GLuint**).

- Para crear o generar un nuevo identificador de textura único (distinto de cualquiera ya existente) usamos:

```
GLuint idTex ;  
// hace idTex igual a un nuevo identificador  
glGenTextures( 1, &idTex );
```

Textura activa

En el estado interno de OpenGL hay en cada momento un identificador de textura activa:

- Cualquier operación de visualización de primitivas usará la textura asociada a dicho identificador.
- Cualquier operación de configuración de la funcionalidad de texturas se referirá a dicha textura activa.

Para cambiar el identificador de textura activa podemos hacer:

```
glBindTexture( GL_TEXTURE_2D, idTex );  
// activa textura con identificador 'idTex'
```

Alojamiento en RAM de imágenes de textura

Antes de usar una textura en OpenGL (de tamaño $n_x \times n_y$), es necesario alojar en la memoria RAM una matriz con los colores de sus texels:

- Cada texel se representa (usualmente) con tres bytes (enteros sin signo entre 0 y 255), que codifican la proporción de rojo, verde y azul, respecto al valor máximo (255).
- Los tres bytes de cada texel se almacenan contiguos, usualmente en orden RGB.
- Los $3n_x$ bytes de cada fila de texels se almacenan contiguos, de izquierda a derecha.
- Las n_y filas se almacenan contiguas, desde abajo hacia arriba.
- Se conoce la dirección de memoria del primer byte, que llamamos **texels** (es un puntero de tipo **void ***)

con este esquema la imagen ocupará, lógicamente, $3n_xn_y$ bytes consecutivos en memoria.

Especificación de los texels de la imagen de textura

En cualquier momento podemos especificar cual será la imagen de textura asociada al identificador de textura activa, con **glTexImage2D**:

```
glTexImage2D( GL_TEXTURE_2D,  
0,           // nivel de mipmap (para imágenes multiresolución)  
GL_RGB,    // formato interno  
ancho,      // núm. de columnas (potencia de dos:  $2^n$ ) (GLsizei)  
alto,       // núm de filas (potencia de dos:  $2^m$ ) (GLsizei)  
0,           // tamaño del borde, usualmente es 0  
GL_RGB,    // formato y orden de los texels en RAM  
GL_UNSIGNED_BYTE, // tipo de cada componente de cada texel  
texels     // puntero a los bytes con texels (void *)  
);
```

Al llamar a esta función, OpenGL leerá los bytes de la RAM y los copiará en otra memoria (típicamente la memoria de vídeo o de la GPU, en un formato interno).

Especificación de la imagen con GLU

Hay una alternativa preferible a `glTexImage2D` en GLU que no requiere imágenes de tamaño potencia de dos, y que además genera automáticamente versiones a múltiples resoluciones (*mip-maps*)

```
gluBuild2DMipmaps( GL_TEXTURE_2D,  
GL_RGB,      // formato interno  
ancho,        // núm. de columnas (arbitrario) (GLsizei)  
alto,         // núm de filas (arbitrario) (GLsizei)  
GL_RGB,      // formato y orden de los texels en RAM  
GL_UNSIGNED_BYTE,  
// tipo de cada componente de cada texel  
texels       // puntero a los bytes con texels (void *)  
);
```

(esta función hace copias escaladas de la imagen para adaptarla a tamaños potencias de dos a distintas resoluciones).

Configuración de texturas

En el estado de OpenGL, hay un conjunto de atributos o parámetros que determinan la apariencia de las texturas. Estos parámetros determinan, entre otros aspectos:

- Como se usa el color de los texels en el MIL con ilum. activada (que reflectividades del material son obtenidas de la textura activa).
- Como se selecciona el texel o texels a partir de una coords. de textura (más cercano o interpolación).
- Como se selecciona el texel cuando las coords. de textura no están en el rango [0, 1] (replicado o truncamiento).
- Si se asignan explícitamente coordenadas o bien OpenGL las genera proceduralmente, y en este caso como se hace.

Selección de texels

OpenGL permite especificar como se seleccionarán los texels en cada consulta posterior de la textura activa, cuando el pixel actual es igual o más pequeño que el texel que se proyecta en él:

- seleccionar el texel con centro más cercano al centro del pixel:

```
glTexParameterI( GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST );
```

- hacer interpolación bilineal entre los cuatro texels con centros más cercanos al centro del pixel:

```
glTexParameterI( GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR );
```

y cuando el pixel es mas grande usando el parámetro
GL_TEXTURE_MIN_FILTER

Asignación explícita

Cuando la generación automática de coords. de textura está desactivada, es necesario especificar a OpenGL las coords. de textura de cada vértice. En el estado interno hay un par (s, t) que se asignan a cada vértice que se envía con **glVertex**.

Para cambiar el par (s, t) actual podemos usar **glTexCoord2f**:

```
glBegin(GL_TRIANGLES);
glTexCoord2f( s0, t0 ); glVertex3f( x0, y0, z0 );
glTexCoord2f( s1, t1 ); glVertex3f( x1, y1, z1 );
glTexCoord2f( s2, t2 ); glVertex3f( x2, y2, z2 );

// ..... otros triángulos ....
glEnd();
```

Ejemplo

```
glGenTextures(1, texName);
 glBindTexture(GL_TEXTURE_2D, *texName);

/* Set parameters */
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);

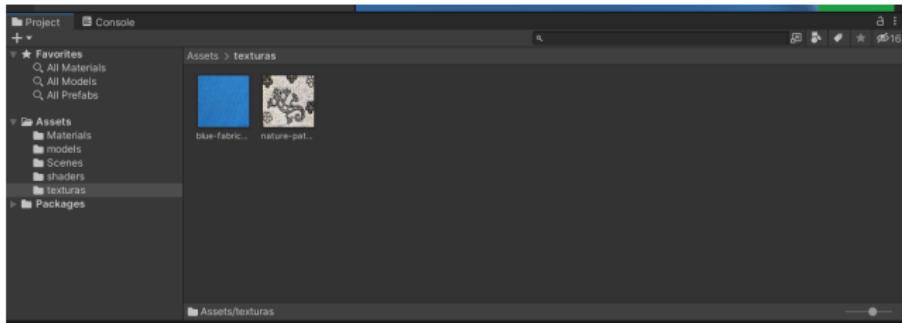
/* Build mipmaps */
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, w, h, 0, GL_RGB, GL_UNSIGNED_BYTE, image);
...
glBegin(GL_QUAD_STRIP){
    glNormal3f( 0.0, 0.0, -1.0 ); /* Vertical hacia atras */

        // Se asignan coordenadas en el espacio de textura (u, v) a aplicar
    glTexCoord2f( 1, 0.0 );
    glVertex3f( x, 0, 0 );

    glTexCoord2f( 0.0, 0.0 );
    glVertex3f( 0, 0, 0 );
    ...
}
glEnd();
```

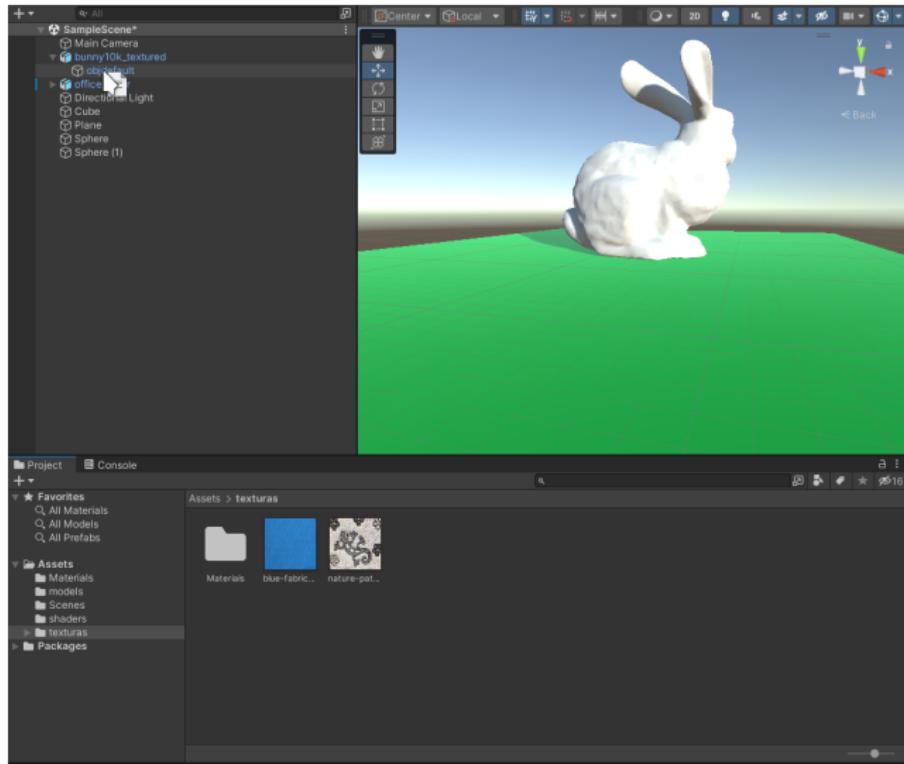
Texturas en Unity

Unity puede aplicar texturas a los objetos de forma fácil (siempre que tengan coordenadas de textura). Debemos cargar la imagen de la textura como un assets.



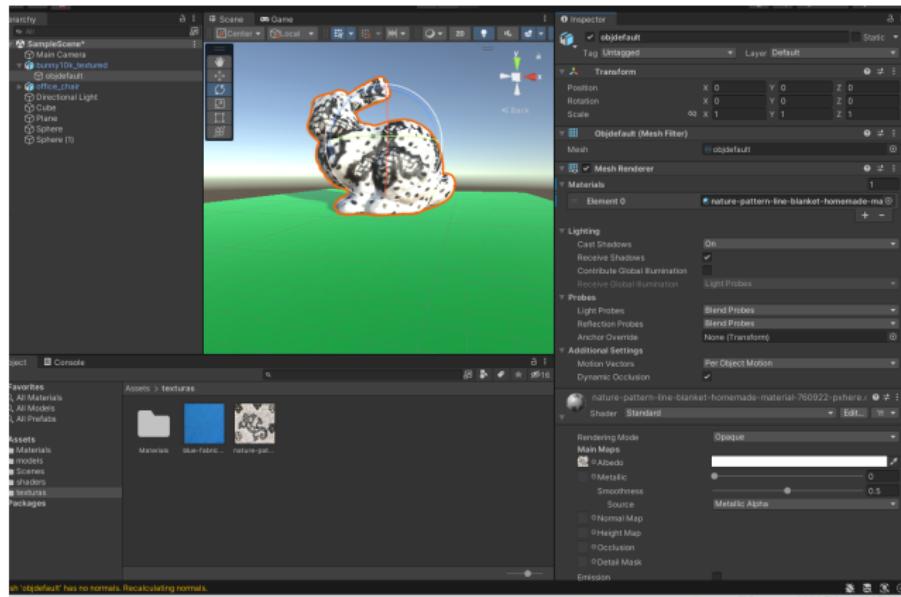
Texturas en Unity

Aplicamos la textura desplazando el asset sobre la malla del objeto.



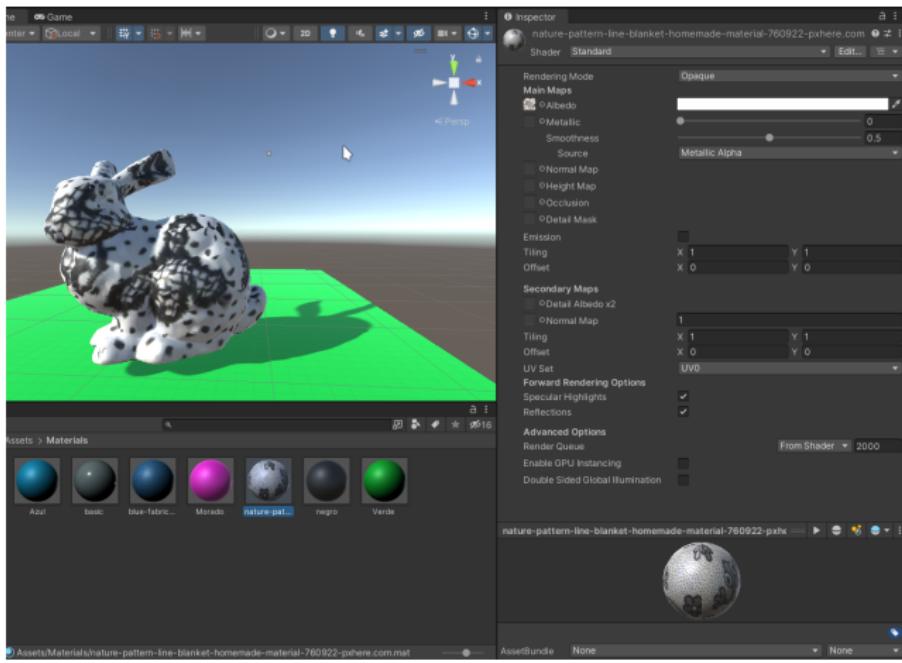
Texturas en Unity

En el inspector del objeto veremos que aparece un material con la textura en el campo *Albedo*.

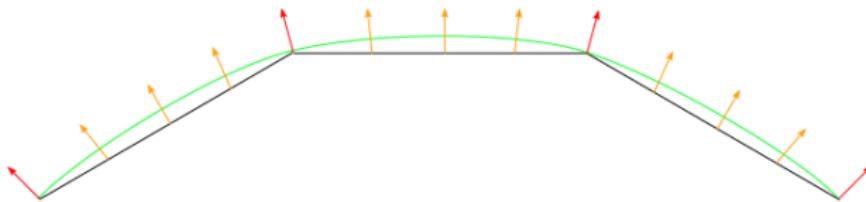


Texturas en Unity

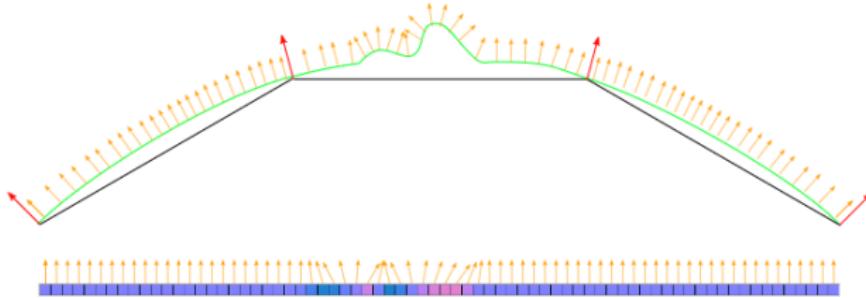
Y aparecerá un nuevo assets de material creado a partir del material que tenía el objeto y la textura.



Relieve (Bump mapping)



Se utiliza una textura para almacenar modificaciones locales de las normales.

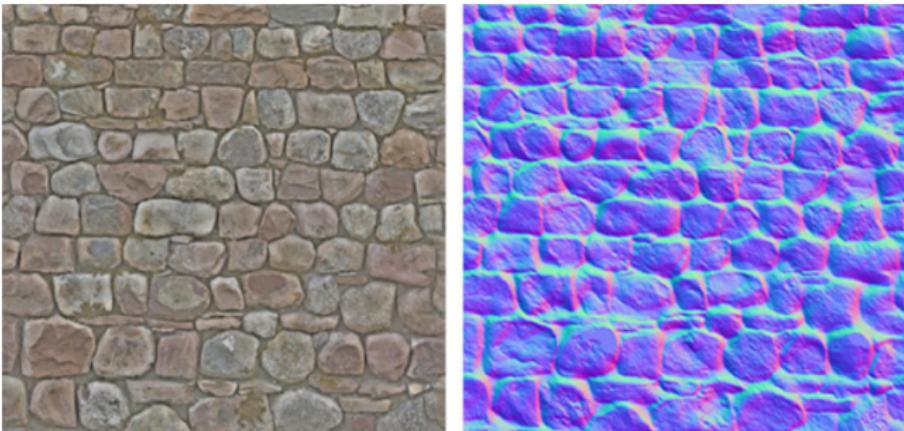


<https://docs.unity3d.com/Manual/StandardShaderMaterialParameterNormalMap.html>

Unity: Normal map

Unity permite hacer bump mapping usando mapas de normales.

La textura se debe configurar como "Normal Map"



Unity: Normal map

