

Informática Gráfica

Juan Carlos Torres

Curso 2024/25

Dpto. Lenguajes y Sistemas Informáticos
Universidad de Granada

Disclaimer

You can edit this page to suit your needs. For instance, here we have a no copyright statement, a colophon and some other information. This page is based on the corresponding page of Ken Arroyo Ohori's thesis, with minimal changes.

CC BY-NC-SA

 This book is released into the public domain using the CC BY-NC-SA. This license enables reusers to distribute, remix, adapt, and build upon the material in any medium or format for noncommercial purposes only, and only so long as attribution is given to the creator. If you remix, adapt, or build upon the material, you must license the modified material under identical terms.

To view a copy of the CC BY-NC-SA code, visit:

<https://creativecommons.org/licenses/by-nc-sa/4.0/>

Colophon

This document was typeset with the help of KOMA-Script and L^AT_EX using the kaobook class.

6

Texturas

Hasta aquí hemos creado modelos que tienen colores uniformes, pero los objetos reales no suelen tener un color constante en toda su superficie, bien por cambios en la estructura del objeto, como las vetas de la madera, o bien por imperfecciones o acumulación de polvo.

La figura 6.1 muestra dos imágenes del mismo cubo, arriba dibujado sin aplicar textura y abajo aplicando una textura de madera.

Una textura es una imagen aplicada a la superficie del objeto para modificar su color.

Aplicar una textura a un polígono es como pegarle un vinilo (ver figura 6.2 <http://cf.ycdn.net/>). El nivel de detalle resultante dependerá de la resolución de la imagen usada como textura.

En la imagen de textura habrá normalmente zonas que no se usen. Por ejemplo en la textura de la imagen 6.2 solo se está usando una zona circular de la imagen.

Cuando se aplica a una malla completa podemos pensar en las texturas como recortables que montamos sobre la superficie del objeto. Por ejemplo la plantilla de la figura 6.3 se puede utilizar para aplicar color a las caras de un dado, como se muestra en la figura 6.4. Se puede conseguir el mismo resultado con diferentes imágenes, por ejemplo en la figura 6.3 se podría colocar el cuadrado del dos sobre el seis, o rotar la imagen.

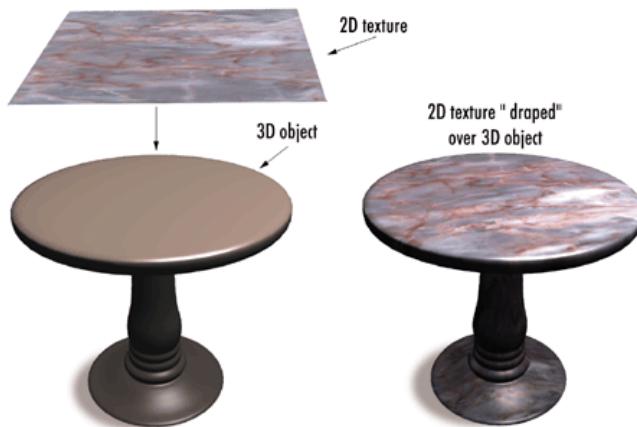


Figura 6.2: Aplicación de textura a una superficie.

6.1 Texturas	49
6.2 Parametrización	50
6.3 Texturas en OpenGL	52
6.4 Texturas en Unity	54
6.5 Bump Mapping	54
6.6 Ejercicios	55

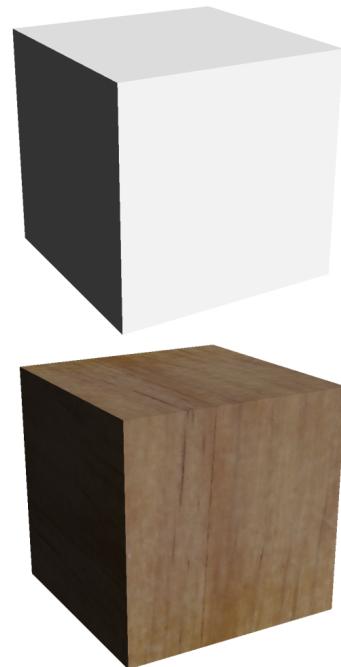


Figura 6.1: Dos imágenes del mismo cubo, sin textura arriba y con textura de madera abajo

6.1. Texturas

Para poder dibujar un objeto con una textura necesitamos, además de la imagen de la textura indicar como se aplica esta al objeto, es decir que punto de la imagen se dibuja en cada vértice de la malla. La figura 6.5 muestra este proceso. En el triángulo destacado en la imagen cada vértice tiene asociada una posición de la imagen, de forma que se le hace corresponder al triángulo del modelo un triángulo en la textura. Las posiciones de la imagen asociadas a los vértices son sus **coordenadas de textura** (u, v).

Las coordenadas de textura están normalizadas, tomando valores entre 0 y 1 en la textura. Es decir las dos esquinas de la diagonal principal tienen coordenadas (0,0) y (1,1). De esta forma las coordenadas son independientes del tamaño de la imagen.

Se pueden utilizar coordenadas de textura (t) mayores que 1 o menores que 0. Dependiendo de la configuración el efecto puede ser tomar el color del borde de la imagen ($\max(\min(t, 1), 0)$) o repetir la textura ($\text{frac}(t)$). Esta última configuración permite una superficie grande con una imagen de textura que se repite.

Normalmente no es posible asignar coordenadas de textura que permitan aprovechar toda la imagen, apareciendo zonas de la imagen no usadas. La figura 6.6 muestra las coordenadas de textura de una tetera, pueden observarse zonas en las que no hay ningún triángulo asociado y por tanto no se usarán. Lógicamente, interesa que el espacio perdido de la textura sea mínimo.

Se aprecia la existencia de agrupaciones de triángulos (se suelen llamar islas). Los vértices que están dentro de la isla tienen asociada su coordenada de textura en la imagen. Los que están en el borde de las islas pueden aparecer en mas posiciones de la imagen, y por tanto tendrán coordenadas de textura diferentes en sus triángulos. Por ejemplo, para el dado de la figura 6.3 el vértice de la esquina superior izquierda de la cara del 4 para la cara del 4, otra para la cara del 2 y otra para la del 1. En estos casos es necesario o bien subdividir la malla o bien almacenar mas de unas coordenadas de textura para cada vértice.

Al dibujar el modelo se consultan las coordenadas de textura de los vértices. Al rasterizar los triángulos se calculan las coordenadas de textura de cada fragmento interpolando las coordenadas de los vértices. Estas coordenadas se utilizan para obtener el color de textura que corresponde al fragmento, calculando el **texel**¹ que le corresponde multiplicando la coordenada (u o v) por la resolución horizontal o vertical de la imagen:

$$X_{tex} = u \cdot ResX$$

$$Y_{tex} = v \cdot ResY$$

Siendo $ResX \times ResY$ la resolución de la imagen. Los valores X_{tex} e Y_{tex} son reales que pueden coincidir con el centro de un texel. En caso contrario, que es lo habitual, podemos utilizar dos estrategias diferentes para obtener el color:

- Asignar el color del texel más cercano.

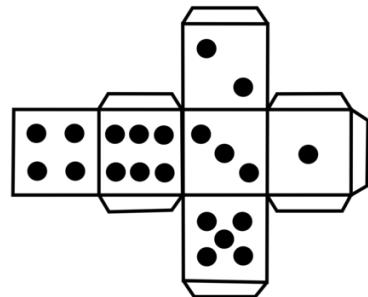


Figura 6.3: La textura para visualizar un cubo como un dado es semejante a un recortable del dado (<https://www.supercoloring.com/>)

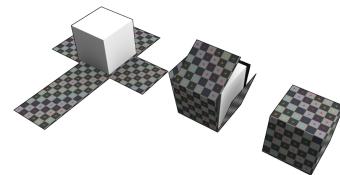


Figura 6.4: Aplicación de la textura de la figura 6.3 a un cubo.

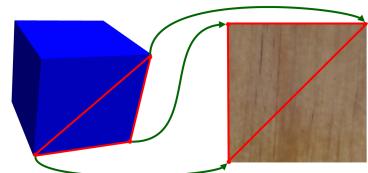


Figura 6.5: Las coordenadas de textura asocian una posición de la textura a cada vértice del polígono.

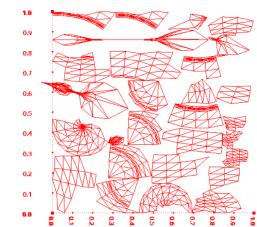


Figura 6.6: Coordenadas de textura creadas para la tetera de la parte inferior. Se muestra la zona de la imagen que se asocia a cada triángulo.

1: Se utiliza el término texel para referirse a los pixel de una textura.

- Hacer una interpolación bilineal de los cuatro texeles próximos.

La figura 6.7 muestra este proceso, en ella el punto rojo representa la posición (X_{tex}, Y_{tex}) obtenida y los signos + los centros de los texeles usados en el cálculo del color. En la parte superior se está usando solamente el texel mas cercano, en la inferior se interpola entre los cuatro texeles cercanos.

La figura 6.8 muestra el efecto de aplicar una textura de rayas a un cubo utilizando ambos métodos de filtrado.

En esta figura cada texel de la textura cubre varios pixeles de la superficie (estamos magnificado la textura). También puede ocurrir que varios texeles de la textura se encuentren en el mismo pixel de la superficie. En este último caso también podemos utilizar estas dos misma estrategias. Si varios texeles se proyectan en el mismo pixel podemos usar el color del más próximo al centro del pixel o hacer una interpolación bilineal entre todos los texeles que se encuentran el pixel.

Si el conjunto de texeles es grande el cálculo de la interpolación es costoso y puede comprometer el rendimiento de la aplicación. Esta situación ocurre cuando la textura tiene más resolución que el dibujo en pantalla de la superficie en la que se está aplicando. Para resolver este problema se pueden precalcular versiones simplificadas de la textura (mipmaps) y utilizar en cada momento la que tenga un tamaño adecuado (figura 6.9).

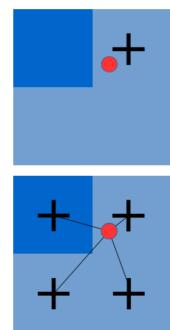


Figura 6.7: Estrategias de asignación del color, arriba usando solo el texel mas cercano, abajo interpolando entre los cuatro vecinos.

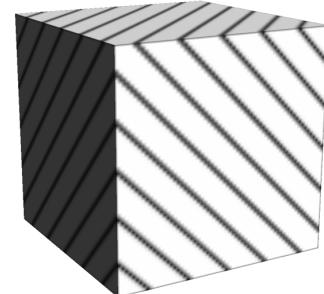
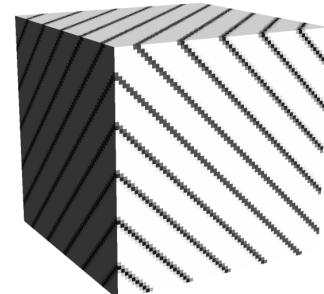


Figura 6.8: Aplicación de una textura a un cubo con ambos métodos de filtrado. Arriba el mas cercano, abajo interpolación.

6.2. Parametrización

La parametrización es el proceso de asignación de coordenadas de textura a un modelo. Cuando el modelo es pequeño y simple (p.e. un cubo) puede hacerse manualmente, pero no es posible en cualquier malla que no sea tan simple. No hay una única forma de asignar las coordenadas de textura a los vértices. la asignación depende de la textura, del modelo y del efecto que se quiera conseguir. En la figura 6.4 se esta aplicando una textura a un cubo de forma que cada cara utiliza una zona diferente de la textura, en este caso la textura sería como la mostrada en la figura 6.3. En la figura 6.5 se está aplicando una textura de madera, que se aplica completa a una cara del cubo, pudiendo aplicarse la misma textura a cada cara reutilizando las coordenadas de textura, generando un resultado como el mostrado en la figura 6.1.

Parametrización procedural

Cuando la superficie del objeto se puede describir, aunque sea de forma aproximada, con una función, se pueden calcular las coordenadas de textura a partir de las coordenadas de los vértices. por ejemplo, para aplicar un imagen de la tierra con proyección mercator como textura a una esfera (figura 6.10) podemos calcular las coordenadas de textura en coordenadas esféricas, esto es, en función de los ángulos de longitud y latitud:

$$u = (\text{longitud} + 180)/360$$

$$v = (\text{latitud} + 90)/180$$

De esta forma estamos asignando coordenadas de textura como si estuviésemos proyectando los puntos en una esfera.

En general podemos calcular las coordenadas de textura usando una función que las calcule a partir de las coordenadas de los vértices:

$$u = f_u(x, y, z)$$

$$v = f_v(x, y, z)$$

Como casos particulares, podemos calcular las texturas proyectando las coordenadas del objeto en un cilindro o en un plano.

Desenrollado

La parametrización procedural es adecuada cuando el objeto tiene una estructura semejante a la de la superficie en la que se proyecta. En caso contrario es necesario utilizar métodos de cálculo de coordenadas de textura que tengan en cuenta la geometría del objeto. Para entender este proceso podemos pensar en un procedimiento para desplegar (*unwrap*) un poliedro realizando cortes en las aristas hasta poder aplanar su superficie. Para obtener una parametrización que se corresponda con la textura de la imagen 6.3 realizaríamos cortes en las aristas que separan la cara del 1 de las del 2, 5 y 4, y en las aristas que separan las caras del 4 y 6 de las del 2 y el 5.

Este proceso se puede realizar de forma totalmente automática o asistida. En los métodos asistidos el usuario puede indicar que aristas se van a cortar (va a ser costuras en la textura). Una forma trivial de desenvolver un poliedro es generar costuras en todas las aristas, de este modo que cada triángulo se proyecta en la textura de forma independiente. Esta parametrización no suele ser aceptable.

Parametrización con MeshLab

Blender permite realizar la parametrización tanto de forma asistida como automática.

MeshLab por su parte permite realizar parametrizaciones triviales, planas, procedurales (indicando la función de proyección) y automáticas (usando un algoritmo de crecimiento de regiones de Voronoi). La parametrización de la figura 6.6 se ha generado con este método.

Los métodos de parametrización se encuentran en el grupo de filtros *Textura*.

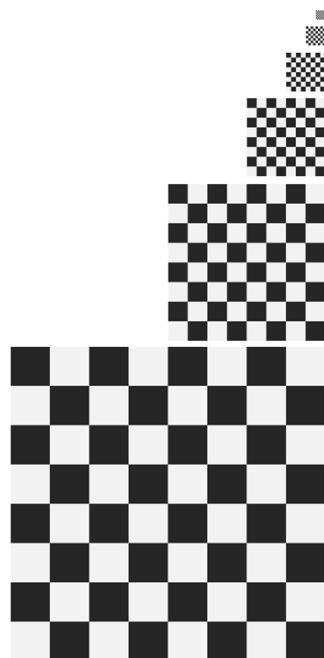


Figura 6.9: Niveles de mipmaps generados a partir de la textura inferior. El tamaño de cada imagen es la mitad de la anterior.

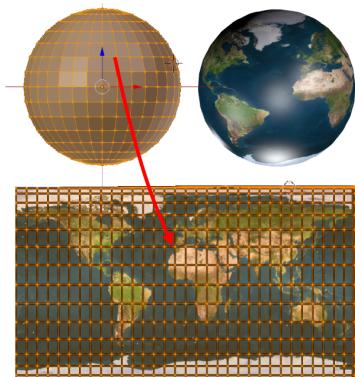


Figura 6.10: Para aplicar la imagen de abajo a una esfera se pueden calcular las coordenadas de textura en función de los ángulos de longitud y latitud.

6.3. Texturas en OpenGL

El procesamiento de texturas en OpenGL se encuentra inicialmente desactivado. Para activarlos se debe usar la orden glEnable. La orden glDisable se pueden usar para desactivarlo: toda l:

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

Cuando están habilitadas las texturas, se consulta la textura activa cada vez que un polígono se proyecta en un pixel, antes de calcular el color de dicho pixel, el color de la textura sustituye a las reflectividades del material (usualmente a la difusa y la ambiental)².

OpenGL puede gestionar más de una textura a la vez, aunque solo una está activa en cada momento. Para cargar una nueva textura hay que pedirle a OpenGL un identificador de textura llamando a la función glGenTextures:

```
GLuint idTex ;
glGenTextures( 1, &idTex ) ; // Crea un identificador de textura y
    lo devuelve en idTex
```

En el estado interno de OpenGL se almacena el identificador de la textura activa, que es la que se usará en cualquier operación de visualización de primitivas y en cualquier operación de configuración de la funcionalidad de texturas. Para cambiar la textura activa se usa la función glBindTexture:

```
glBindTexture( GL_TEXTURE_2D, idTex ) ; // Activa la textura 2D
    idTex
```

El primer parámetro, GL_TEXTURE_2D, indica que la textura es bidimensional (en esta lección solamente abordaremos texturas 2D pero también existen otros tipos de textura, como las 1D y las 3D).

La imagen de la textura se le debe pasar a OpenGL, podemos usar para ello la función glTexImage2D, que envía la imagen de la textura como un puntero a la cadena de bytes que contiene la imagen, organizada como una secuencia de texels. Los parámetros de la función indican como está empaquetada la imagen. Por ejemplo, la siguiente llamada

```
glTexImage2D( GL_TEXTURE_2D,
    0,           // nivel de mipmap (para imagenes multiresolucion)
    GL_RGB,     // formato interno
    ancho,      // num. de columnas (GLsizei)
    alto,       // num de filas (GLsizei)
    0,           // tamano 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 *)
);
```

indica que se va a transferir (a la textura activa) la imagen almacenada en texels, como una textura 2D; para el nivel mipmap 0; que los texels se deben guardar en RGB; que la imagen tiene ancho x alto texels, sin borde; que la imagen se pasa en RGB con cada componente almacenada como unsigned byte, es decir como intensidades entre 0 y 255. El tamaño de la imagen (tanto en ancho como en alto) debe ser potencia de 2.

2: Si la iluminación está desactivada, el color de la textura sustituye al especificado con glColor.

En el buffer de texels, los tres bytes de cada texel se almacenan contiguos. Los $3n_x$ bytes de cada fila de texels se almacenan contiguos, de izquierda a derecha, y las n_y filas se almacenan contiguas, desde abajo hacia arriba. Con este esquema la imagen ocupará, lógicamente, $3n_x n_y$ bytes consecutivos en memoria.

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).

En GLU hay una alternativa más simple a `glTexImage2D` 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,       // num. de columnas (arbitrario) (GLsizei)
    alto,        // num 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 *)
);
```

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 usando la función `glTexParameter`. Para seleccionar el texel con centro más cercano al centro del pixel se usa:

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

Para 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);
```

De la misma forma se puede controlar el comportamiento en cuando los texels son más pequeños que los pixeles usando como segundo parámetro `GL_TEXTURE_MIN_FILTER`.

También se usa la función `glTexParameter` para fijar el comportamiento cuando las coordenadas de textura se salen del intervalo (0, 1), utilizando como segundo parámetro `GL_TEXTURE_WRAP_S` para la coordenada de textura u y `GL_TEXTURE_WRAP_T` para la v:

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
```

Las coordenadas de textura se asignan a los vértices con la función `glTexCoord2f`. El código del listado 6.1 dibuja un triángulo dando las coordenadas de textura de sus vértices.

El siguiente fragmento de código muestra la inicialización y uso de la textura:

```
GLuint *texName;
unsigned char* image;

// Lectura de imagen
```

Listing 6.1: Creación un triángulo asignando coordenadas de textura a los vértices.

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

```

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

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);

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 */

    glTexCoord2f( 1, 0.0 );
    glVertex3f( x, 0, 0 );

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

```

6.4. Texturas en Unity

Unity puede aplicar texturas a los objetos de forma fácil. En primer lugar debemos cargar la imagen de la textura como un assets. En el panel inspector de la textura se puede ajustar los parámetros de la textura (figura 6.11).

Aplicamos la textura desplazando el asset sobre la malla del objeto. En el inspector del objeto veremos que aparece un material con la textura en el campo *Albedo*. Aparecerá un nuevo assets de material creado a partir del material que tenía el objeto y la textura.

6.5. Bump Mapping

Hemos visto como utilizar texturas para modificar el color de las superficies, pero este no es el único uso de las texturas. También se pueden usar texturas para modificar las normales generando sensación de relieve, modificar la superficie, almacenar información de oclusión.

Bump mapping es el uso de texturas para alterar localmente las normales, generando sensación de relieve en una superficie plana. La textura usada puede tener un solo canal que almacena el desplazamiento de la superficie que se quiere representar respecto a la geometría (height map) o el vector de desplazamiento de la normal (normal map). Un normal map es una imagen en color en la que cada texel representa el vector que se debe sumar a la normal almacenada en el modelo (figura 6.12).

Unity permite hacer bump mapping usando mapas de normales. Al cargar la imagen de la textura se debe configurar el tipo de textura como "Normal Map". La figura 6.13 muestra el resultado de la aplicación de la textura de la figura 6.12 como mapa de normales a un cubo.

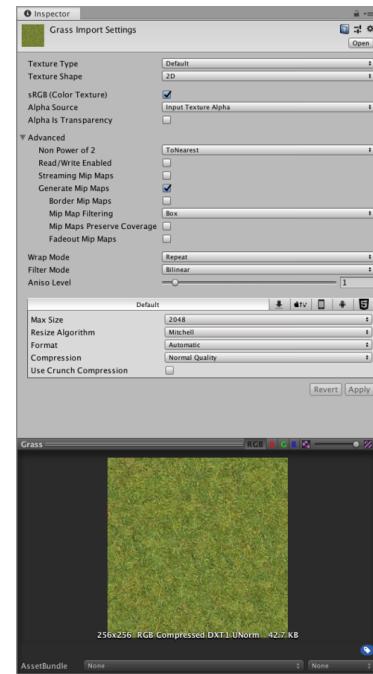


Figura 6.11: Panel de parámetros de una textura en Unity 3D.

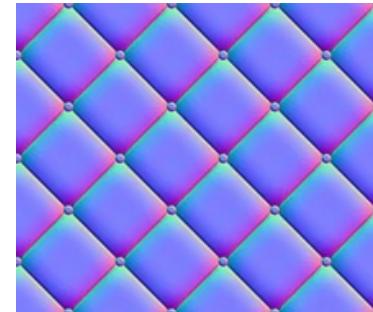


Figura 6.12: Mapa de normales.

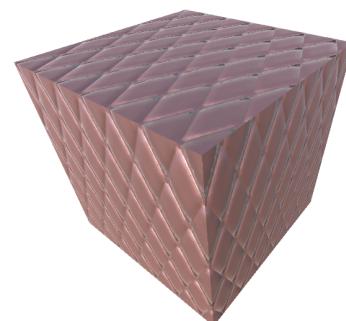


Figura 6.13: Visualización de un cubo con mapa de normales.

6.6. Ejercicios

1. Explica como diseñarías el modelo de la pirámide de Keops para aplicar como textura una fotografía aérea como la mostrada en la figura 6.14.
2. Como alternativa a la representación realizada en el ejercicio 1 plantea la posibilidad de crear la textura usando fotografías de cada una de las caras como la mostrada en la figura 6.15.
3. ¿Como se debe preparar la textura y que parámetros se debe utilizar para aplicarla al suelo de un escenario usando la imagen de la figura 6.16.
4. Se quiere crear un escenario urbano en el que se representarán edificios como paralelepípedos de diferente tamaño, cuya altura, en metros, es siempre múltiplo de 3 y menor o igual que 15. La anchura de cada pared del paralelepípedo, en metros, es múltiplo de 2, hasta un máximo de 16. Para darrealismo a la escena, se aplicara la textura de la imagen 6.17. El espacio de cada ventana (delimitado con un recuadro punteado en la imagen) debe ocupar una zona del edificio de 3m de altura por 2m de anchura. La anchura de cada pared del paralelepípedo, en metros, es múltiplo de 2, hasta un máximo de 16. Explique cómo crearía la geometría, la topología y asignaría las coordenadas de textura en el constructor de un paralelepípedo definido como:

`Edificio::Edificio(int ancho, int alto, int profundo);`

5. ¿Como se puede adaptar el ejercicio anterior para que el sistema funcione cuando las paredes tengan mas de 16 m?
6. Explica como diseñarías el modelo de un tetraedro para aplicarle como textura la imagen mostrada en la figura 6.18.
7. Genera un modelo de tierra usando un mapa mundi como textura como el mostrado en la figura 6.10. Prográmalos en OpenGL.
8. ¿Es posible generar el modelo de la tierra usando como textura dos vistas desde posiciones opuestas como la mostrada en la figura 6.19.
9. Explica como generarias un modelo 3D de un edificio usando textura la imagen de un recortable de cartulina como el de la figura 6.20. Indica como representarías el modelo y la textura.



Figura 6.14: Vista aérea de la pirámide de Keops.



Figura 6.15: Vistalateral de la pirámide de Keops.



Figura 6.16: Imagen para textura de suelo.

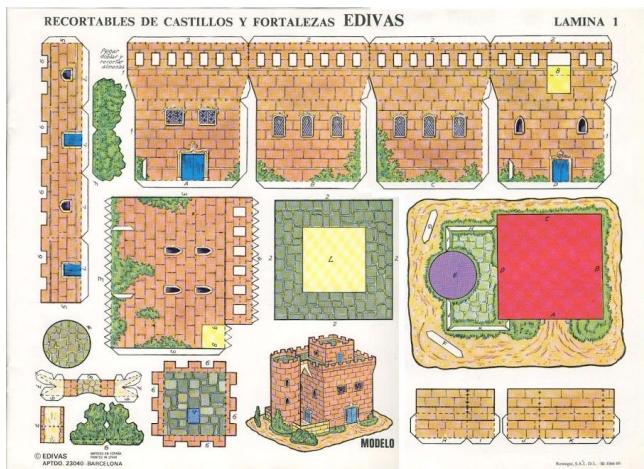


Figura 6.20: Recortable de un castillo.

10. Explica como se puede utilizar texturas para generar un efecto de animación de una superficie, como el movimiento de la hierba con el viento.
11. Plantea como se puede modificar el ejercicio 7 para que se muestre la mitad de la tierra de noche.
12. Se desea incluir una valla publicitaria digital en una escena. La valla ha de mostrar dos imágenes alternadas en el tiempo, intercambiándolas cada 10 segundos. El cambio de una imagen a otra se realiza mediante un desplazamiento vertical que dura un segundo. El objeto que representa la valla es un rectángulo. Diseña la textura y el modelo.



Figura 6.17: Imagen para textura de pared de edificio.

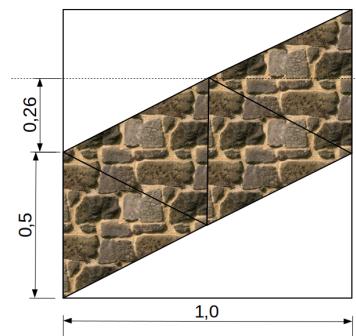


Figura 6.18: Imagen para aplicar como textura a un tetraedro.

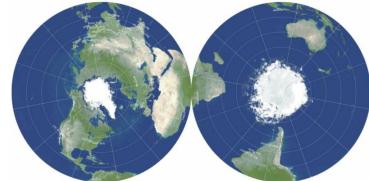


Figura 6.19: Dos imágenes de la tierra.