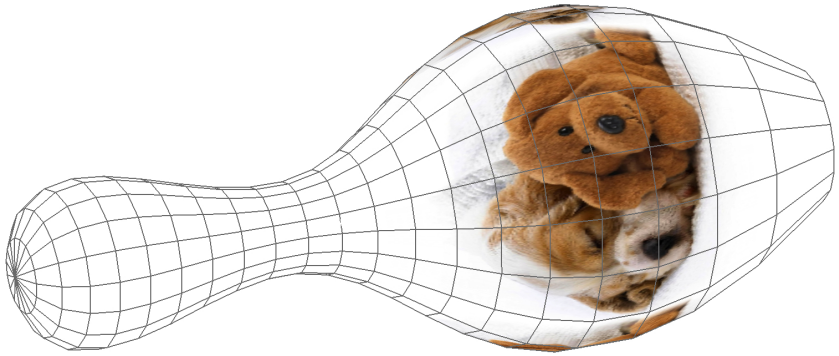


Tema 5: Texturas

José Ribelles

Departamento de Lenguajes y Sistemas Informáticos, Universitat Jaume I

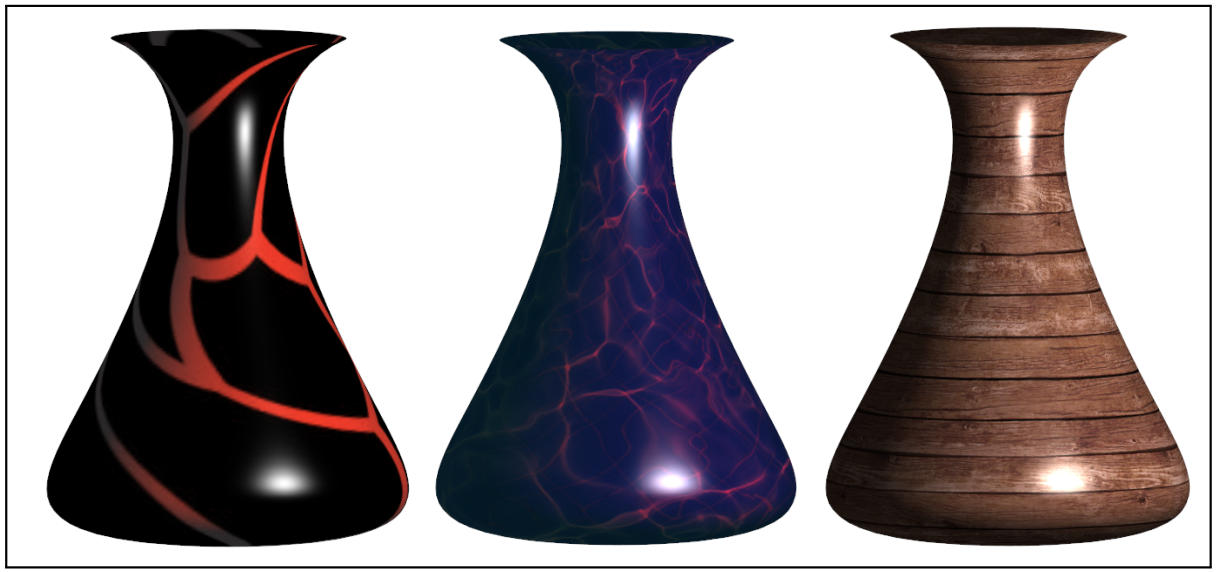
VJ1221 - Informática Gráfica

Contenido

Hoy veremos...

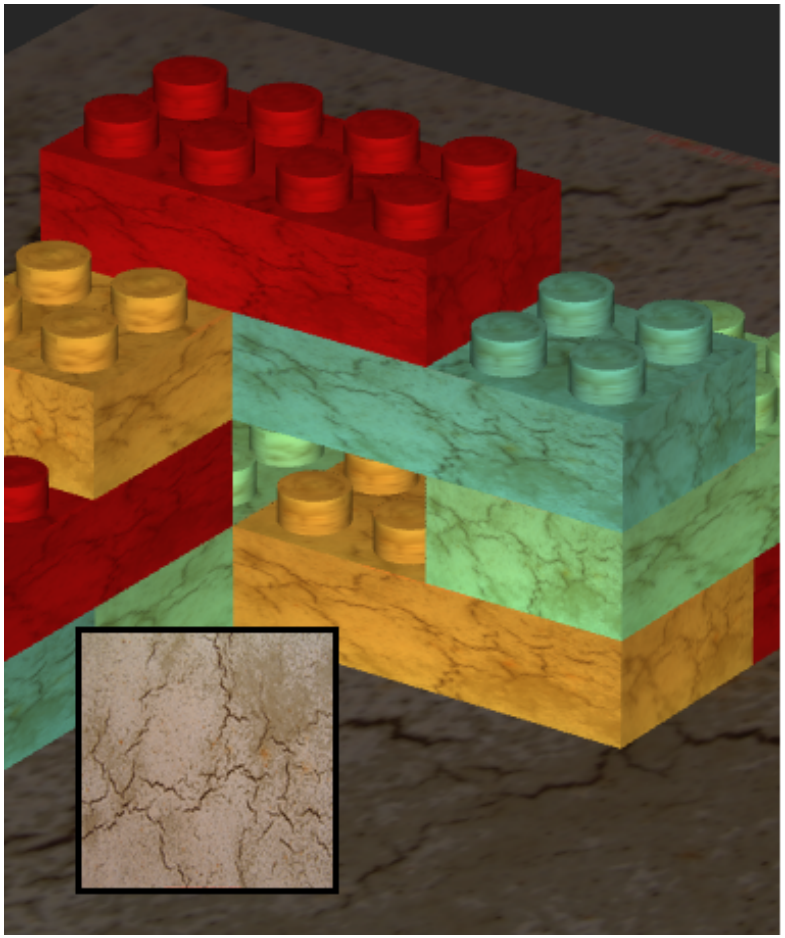
Introducción

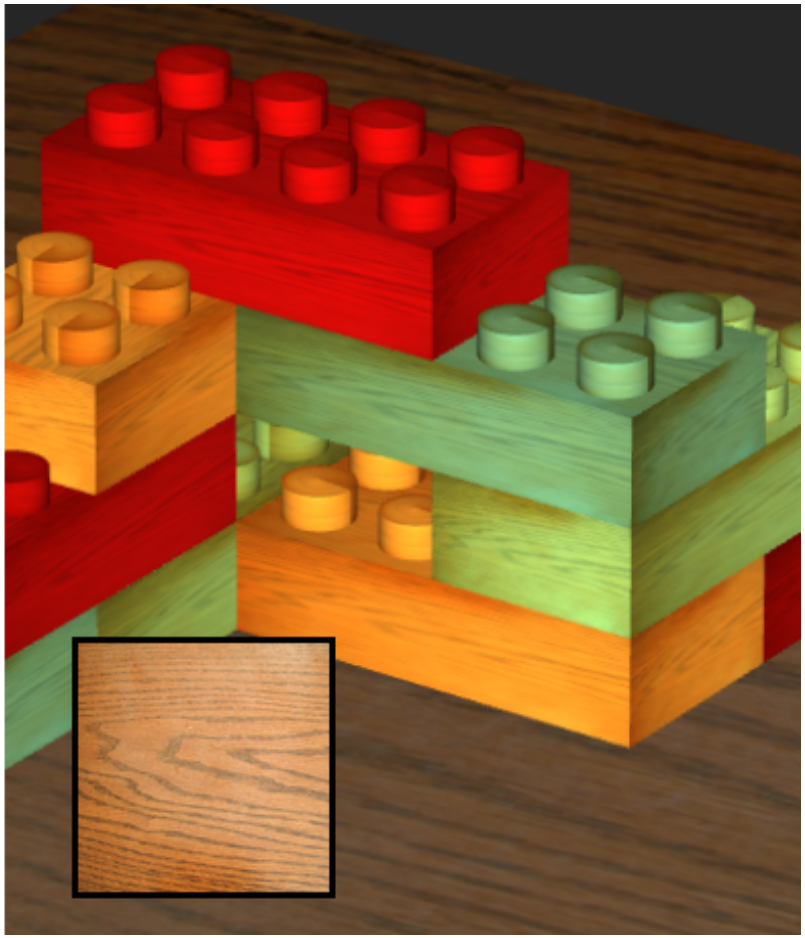
- El uso de texturas para aumentar el realismo visual de las aplicaciones es muy frecuente.
- Quizá el caso más habitual es utilizar imágenes 2D a modo de mapa de color de manera que el valor definitivo de un determinado píxel dependa de la iluminación de la escena y de la textura.



Introducción

A menudo la textura se combina con las propiedades del material.





Introducción

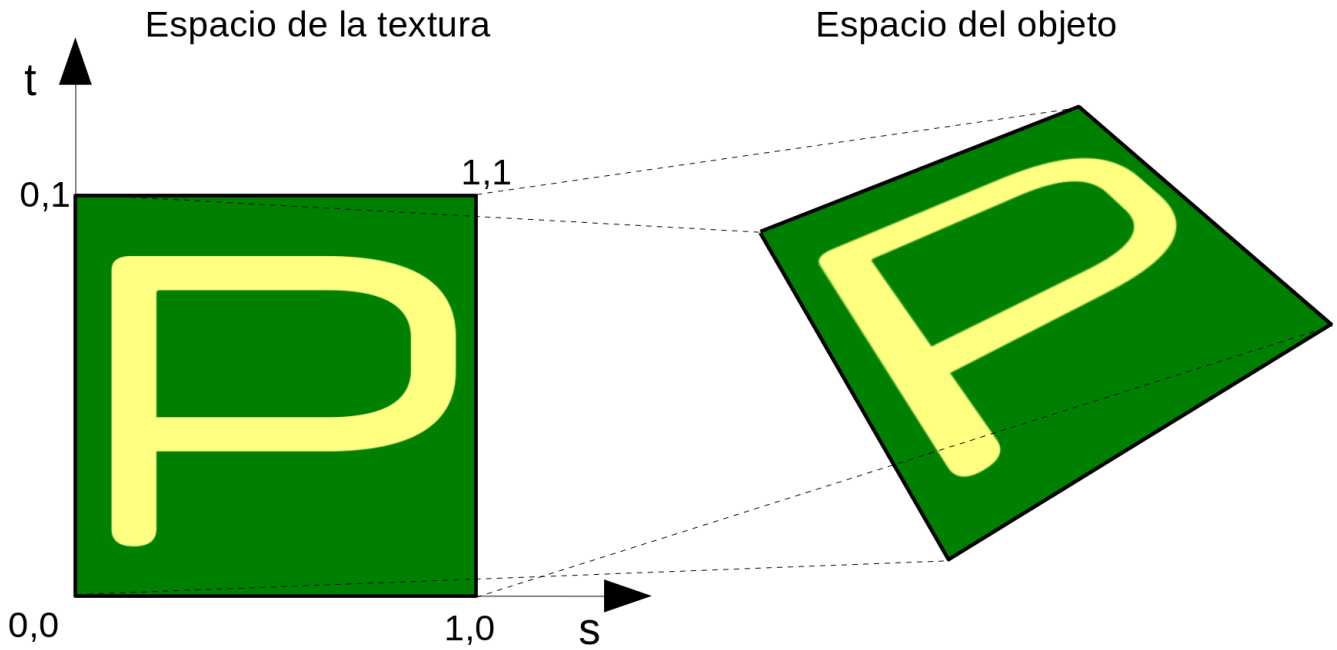
- Pero no siempre es así, por ejemplo, a veces se utiliza para modificar las normales o desplazar la geometría.

Hoy veremos...

Coordenadas de Textura

Descripción

- Las coordenadas de textura son un atributo más de los vértices.
- Es responsabilidad del programador suministrar estas coordenadas.
- El rango de coordenadas válido en el espacio de la textura es entre 0 y 1, independientemente del tamaño en píxeles de la textura.



Obtener las coordenadas de textura

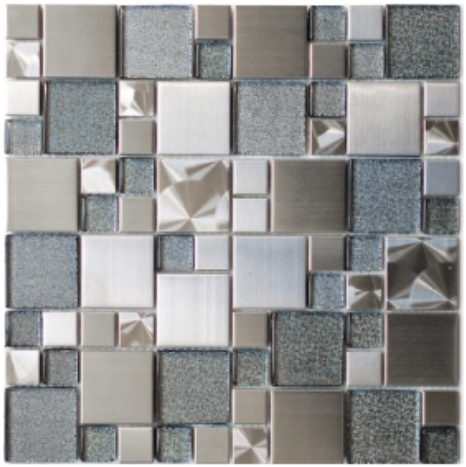
Métodos

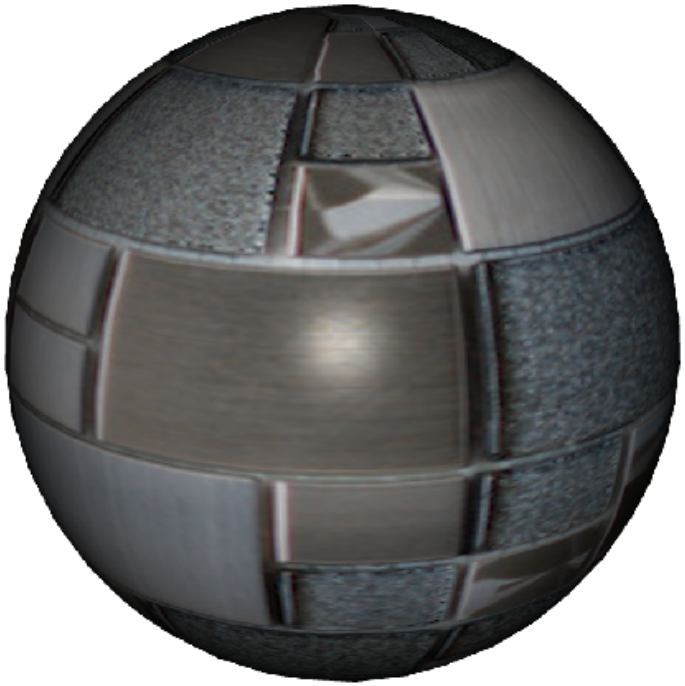
- A mano, igual que la geometría.
- Los paquetes de modelado las proporcionan y permiten ajustarlas a los artistas igual que editan la geometría.
- Las superficies paramétricas las proporcionan de manera natural.
  - Esfera:

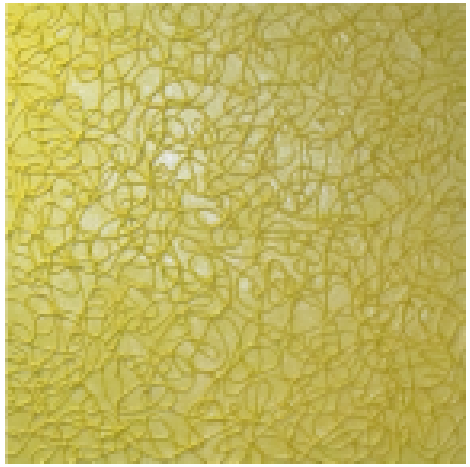
$$x = r \cdot \sin(v) \cdot cos(u) \quad (1)$$
$$y = r \cdot sin(v) \cdot sin(u) \quad (2)$$
$$z = r \cdot cos(v) \quad (3)$$

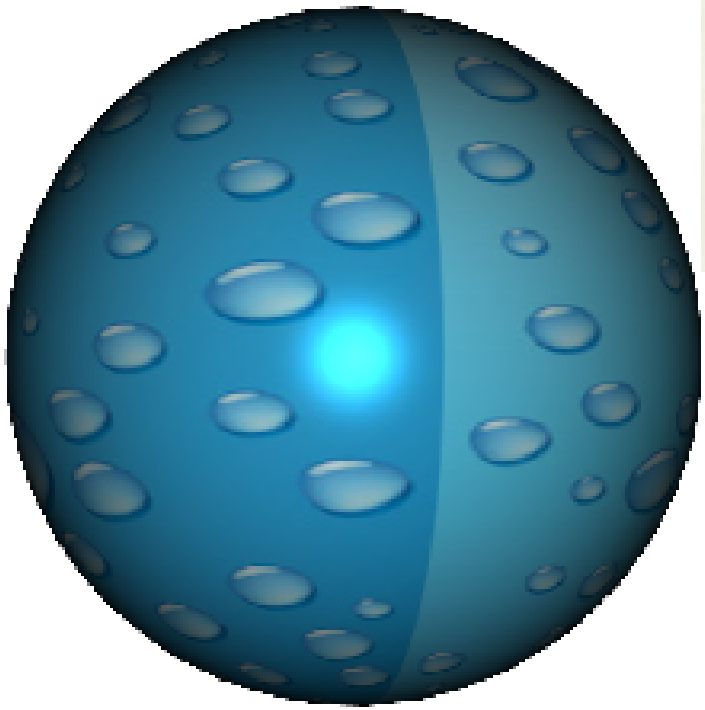
  - Toro

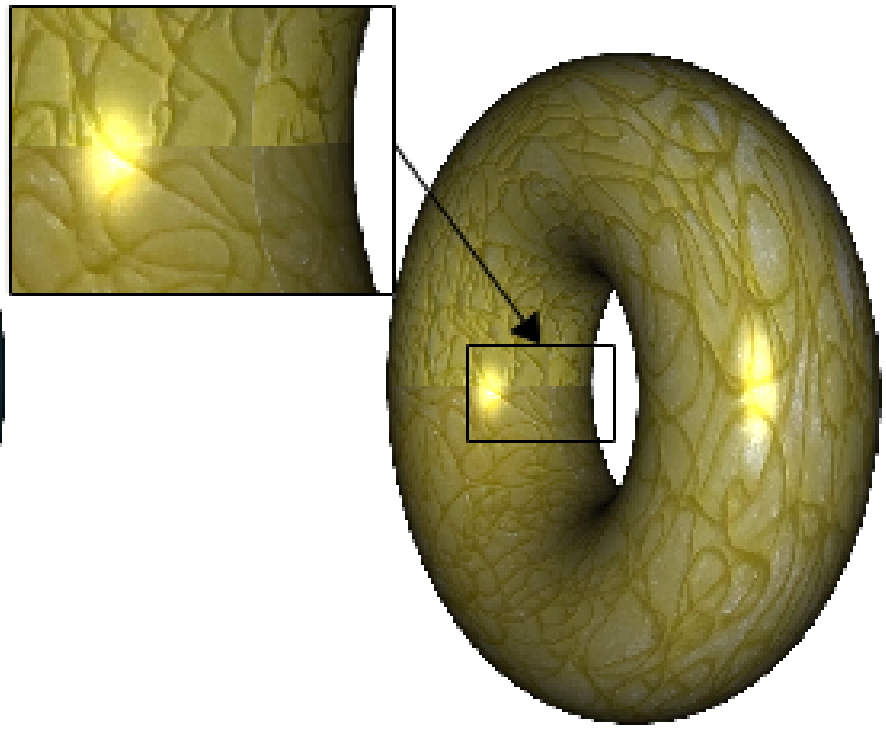
$$x = (R + r \cdot \cos(v)) \cdot cos(u) \quad (4)$$
$$y = (R + r \cdot \cos(v)) \cdot sin(u) \quad (5)$$
$$z = r \cdot sin(v) \quad (6)$$





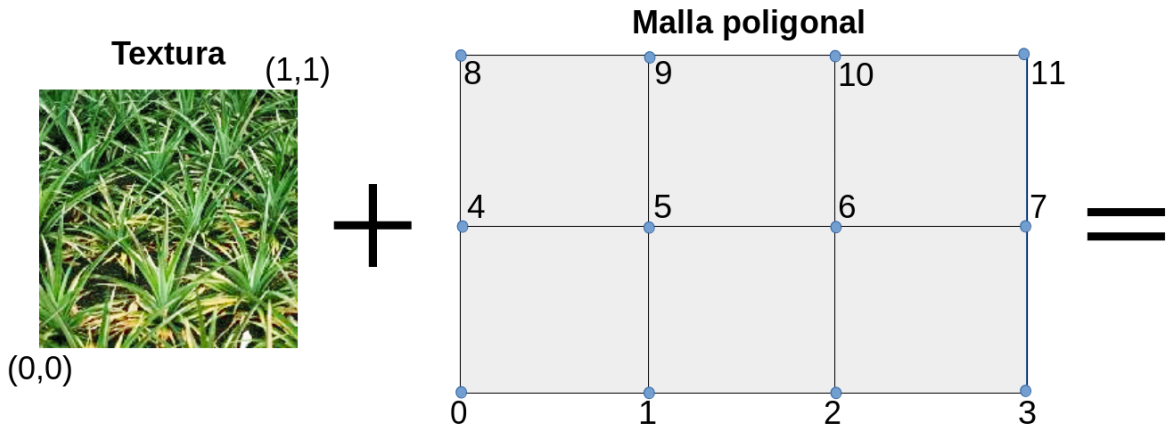






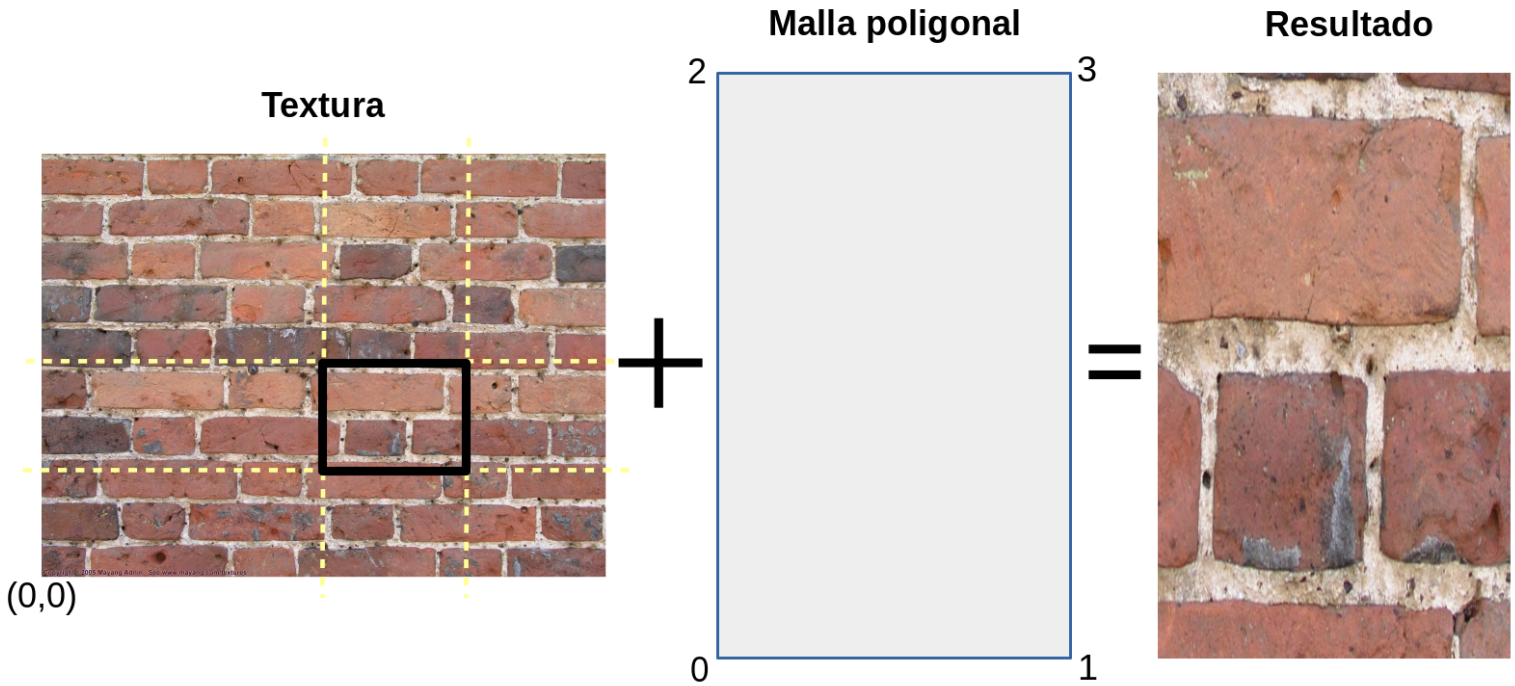
Ejercicio

Determina las coordenadas de textura de cada uno de los vértices que se encuentran numerados en la siguiente figura. Para averiguarlas, ten en cuenta el resultado que también se muestra en la figura.



Ejercicio

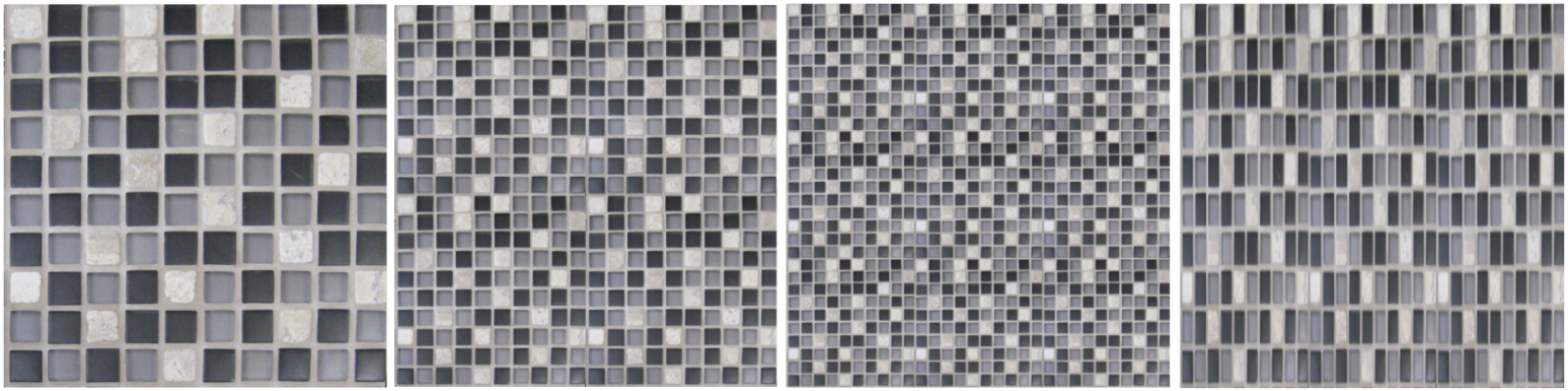
Determina las coordenadas de textura de cada uno de los vértices que se encuentran numerados en la siguiente figura. Para averiguarlas, ten en cuenta el resultado que también se muestra en la figura. En este caso, el ancho y alto del trozo de textura utilizado es de un cuarto del ancho y alto total de la textura respectivamente.



Coordenadas de textura mayor que 1

Se repite la textura ...

- La parte entera de las coordenadas se ignora.
- En WebGL:
  - gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_WRAP_S, gl.REPEAT);
  - gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_WRAP_T, gl.REPEAT);

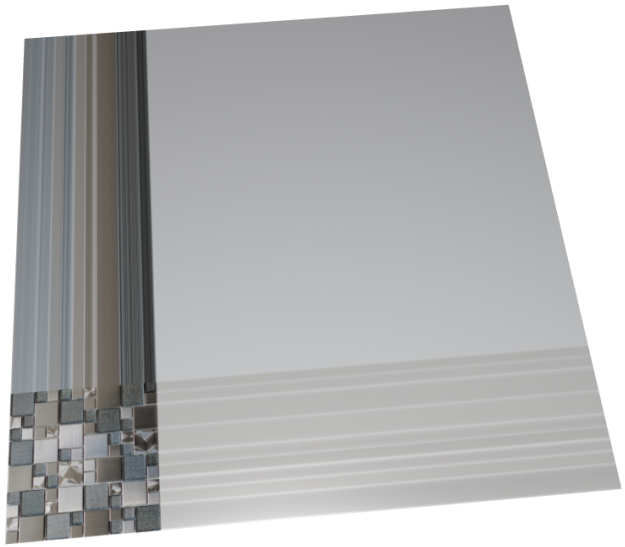


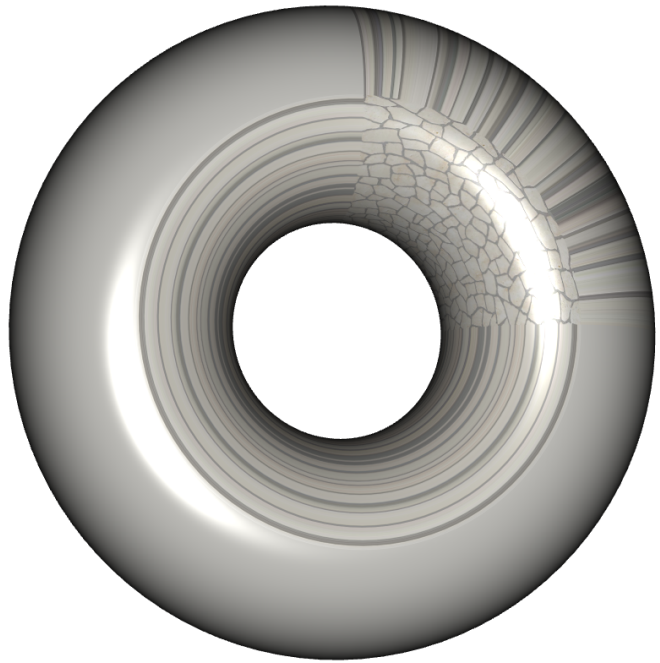
1x1 2x2 3x3 3x1

Coordenadas de textura mayor que 1

Se extienden las aristas de las texturas.

- gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_WRAP_S, gl.CLAMP_TO_EDGE);
- gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_WRAP_T, gl.CLAMP_TO_EDGE);

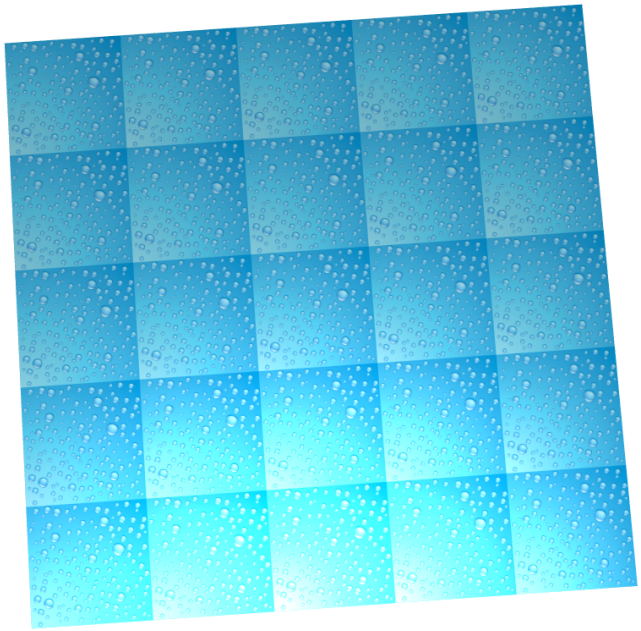


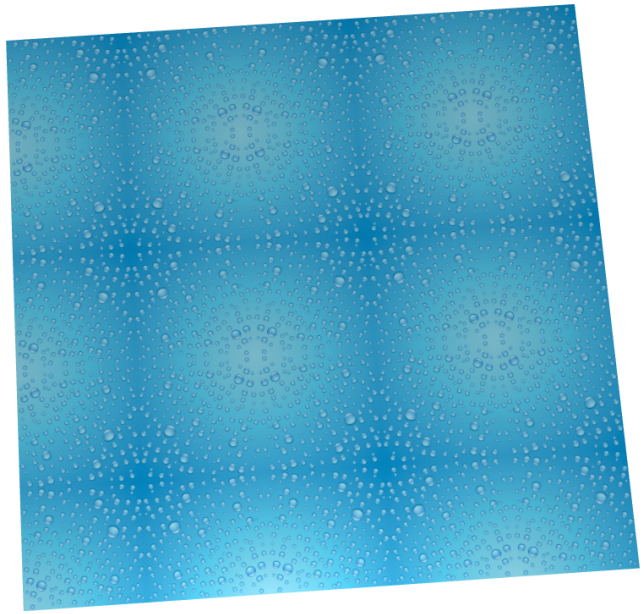


Coordenadas de textura mayor que 1

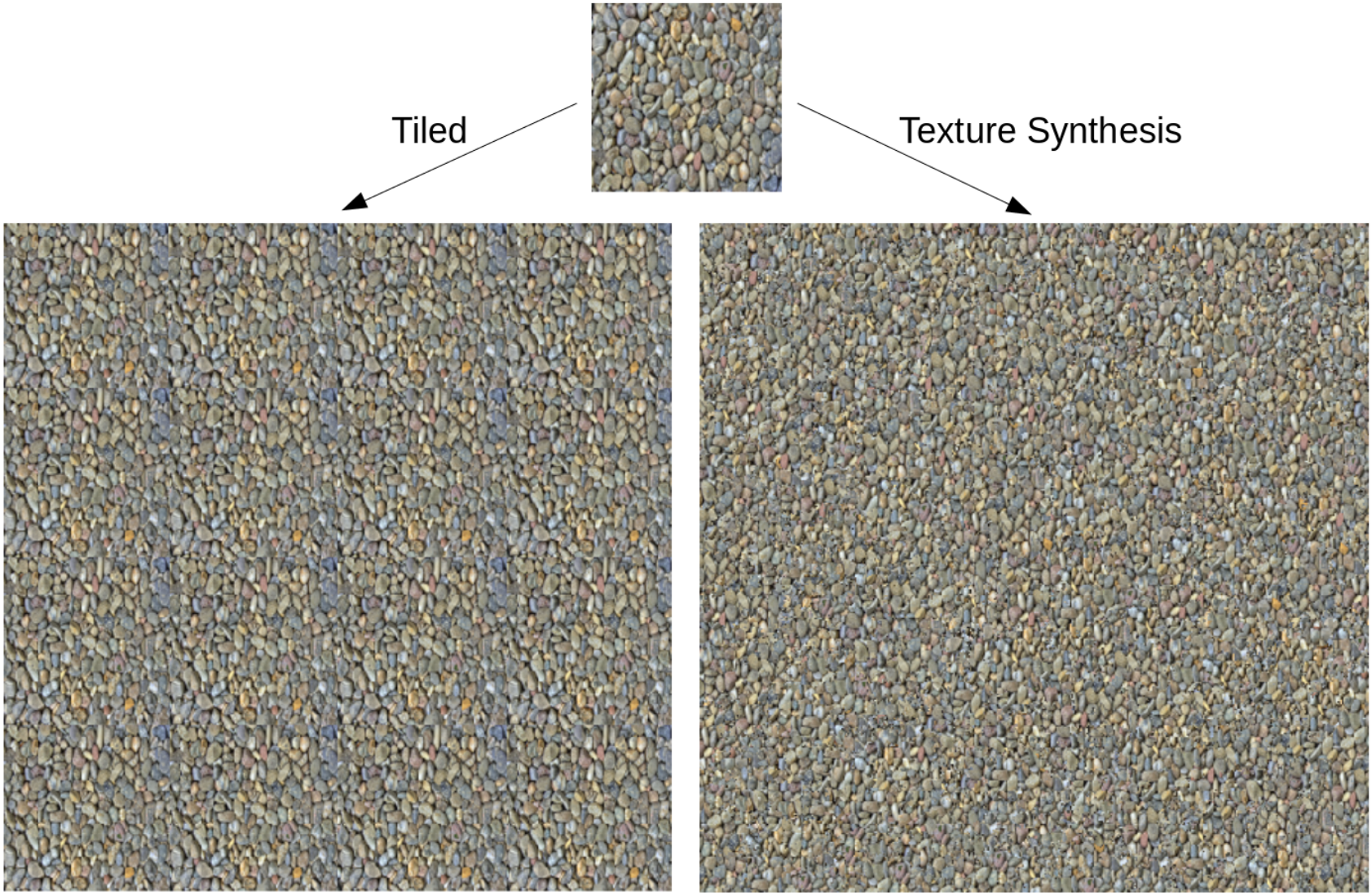
Se repiten de manera simétrica.

- gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_WRAP_S, gl.MIRRORED_REPEAT);
- gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_WRAP_T, gl.MIRRORED_REPEAT);



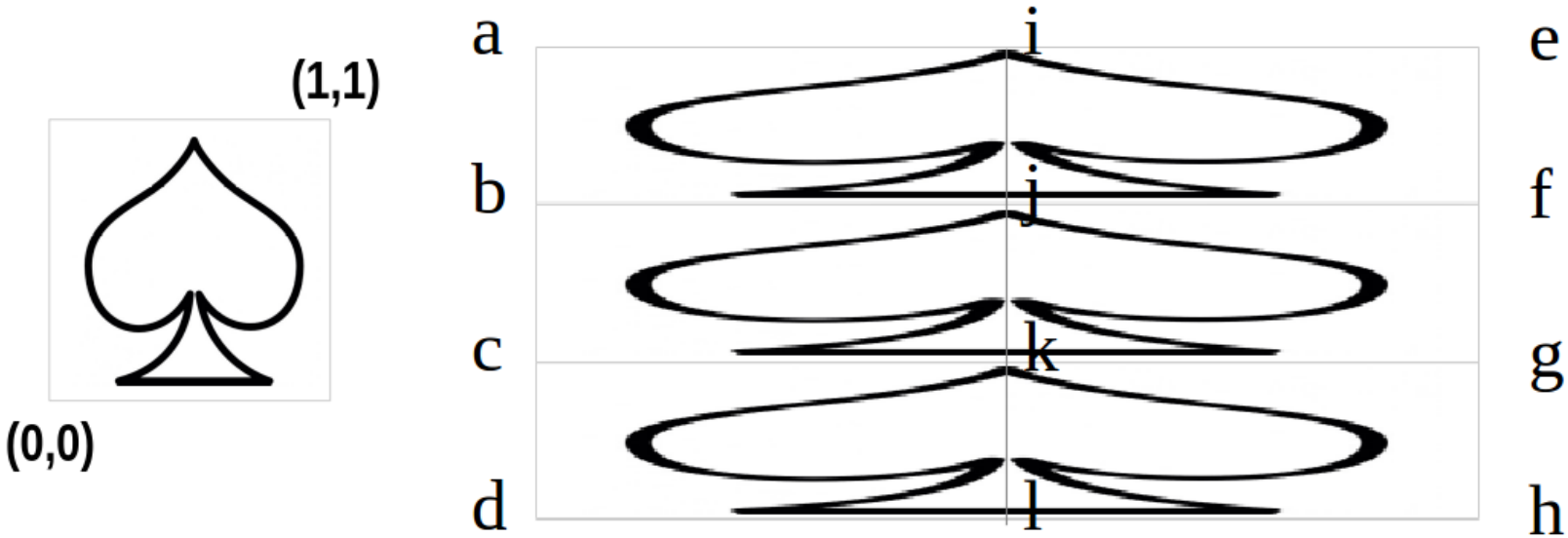


Texture Synthesis



Ejercicio

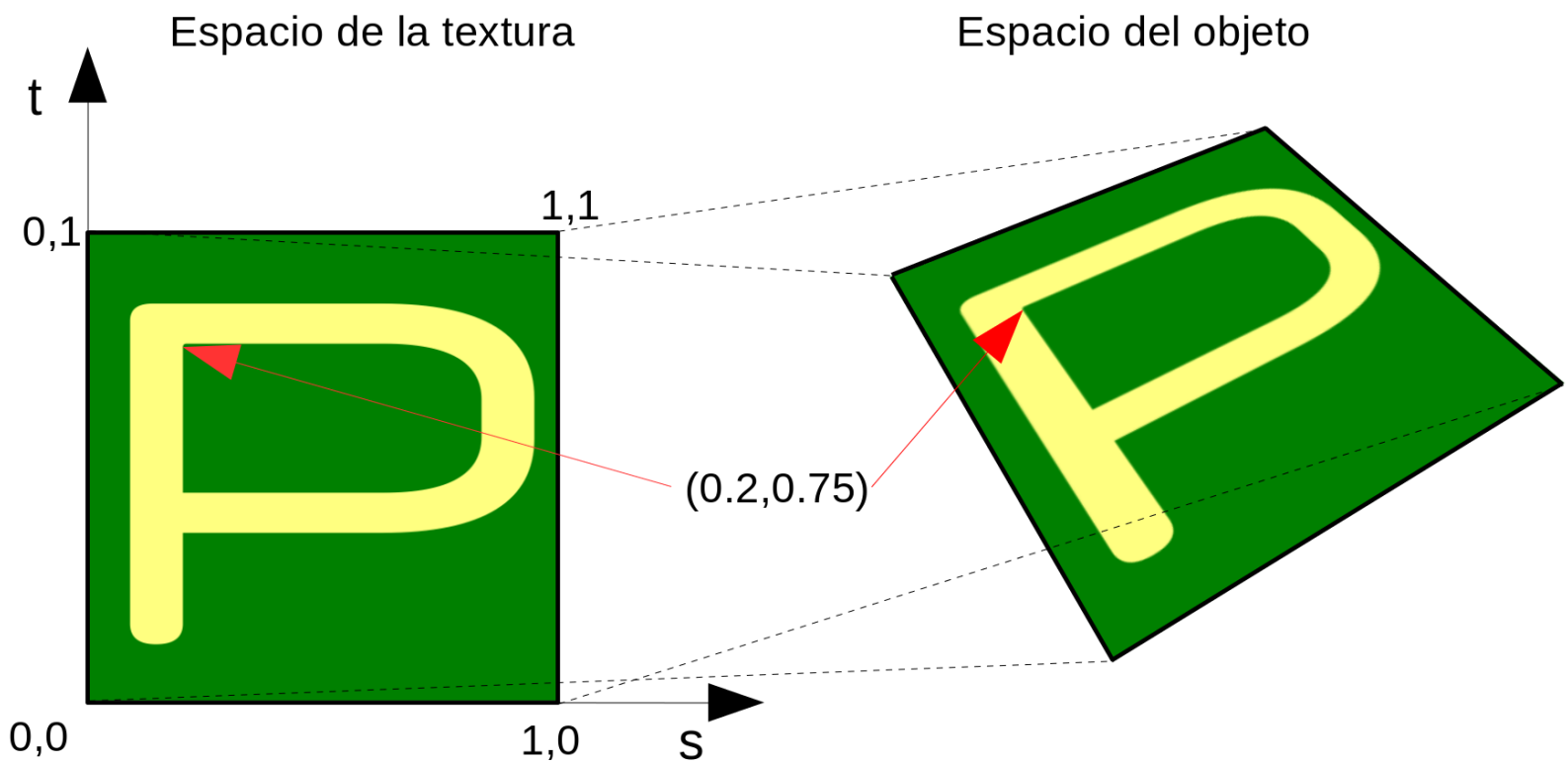
Determina las coordenadas de textura de los siguientes vértices (enumerados como a, b, c, d, e, f, g, h, i, j, k, l) de manera que la textura quede como se muestra en la figura.



Coordenadas interpoladas para cada fragmento

Descripción

- Las coordenadas de textura se proporcionan para cada vértice y son interpoladas en el *pipeline* del procesador gráfico.



El Shader

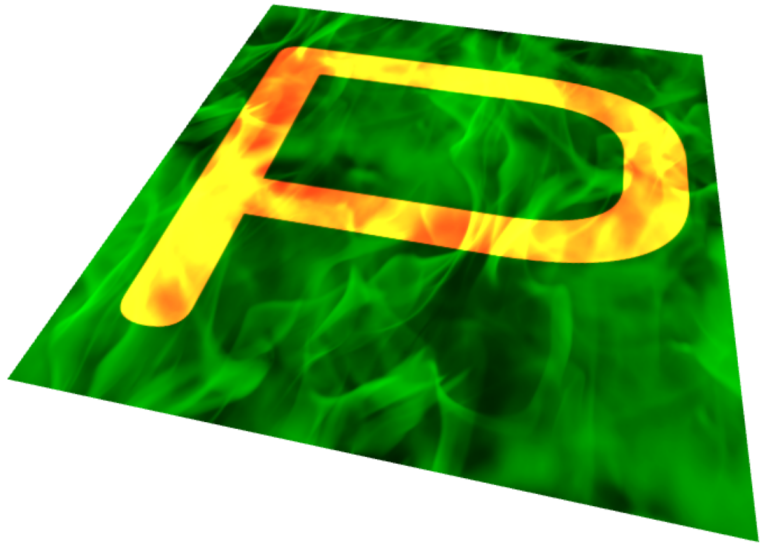
Listado 1: Shader básico para utilizar una Textura 2D

```
// Vertex shader
...
in  vec2 VertexTexcoords;  // nuevo atributo
out vec2 texCoords;

void main()  {
   ...
  // se asignan las coordenades de textura del vertice a la variable texCoords
  texCoords   = VertexTexcoords;
}

// Fragment shader
...
uniform sampler2D myTexture;     // la textura
in vec2 texCoords;               // coordenadas de textura interpoladas

void main()
{
   ...
  // acceso a la textura para obtener un valor de color RGBA
  fragmentColor = texture(myTexture, texCoords);
}
```

Multitexturas

Listado 2: Shader básico para utilizar varias Textura 2D

```
// Fragment shader
...
uniform sampler2D myTexture1, myTexture2;     // las texturas
in vec2 texCoords;     // coordenadas de textura interpoladas

void main()
{
   ...
   // acceso a la textura para obtener un valor de color RGBA
   fragmentColor = texture(myTexture1, texCoords) * texture(myTexture2, texCoords);
}
```



Hoy veremos...

Leyendo Texeles

¿Cómo obtener el valor de la textura?

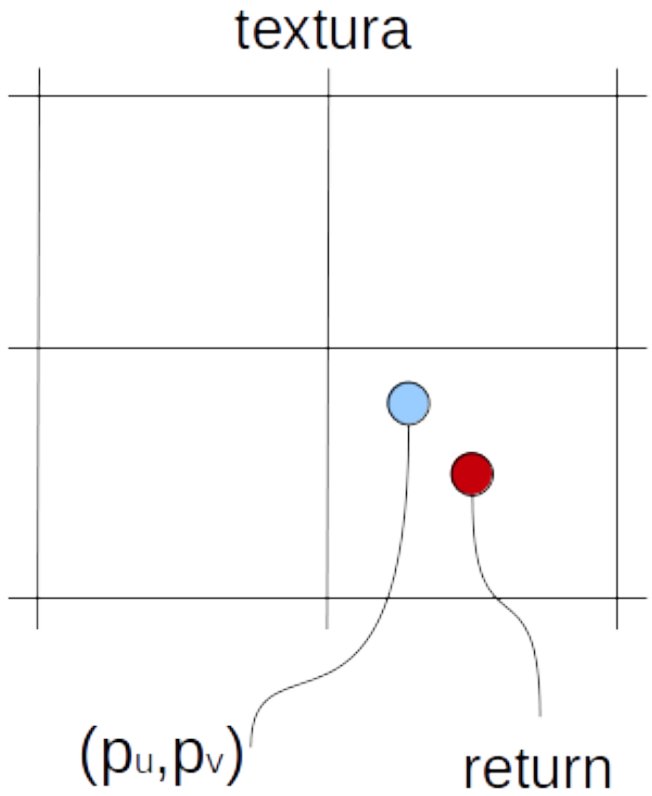
- Un Texel es un píxel de la textura.
- En una GPU moderna ($>$ 2008) se puede acceder tanto desde el procesador de vértices como desde el de fragmentos.
- Rara vez un píxel de la imagen final se corresponde con un único texel de la textura, aparecen dos problemas:
  - **Magnificación:** cuando un texel se corresponde con muchos pixeles.
  - **Minimización:** cuando un píxel se corresponde con muchos texeles.

Magnificación

Box filter

Se utiliza el *texel* más cercano. Efecto de pixelado. Muy rápido.

gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MAG_FILTER, gl.NEAREST);

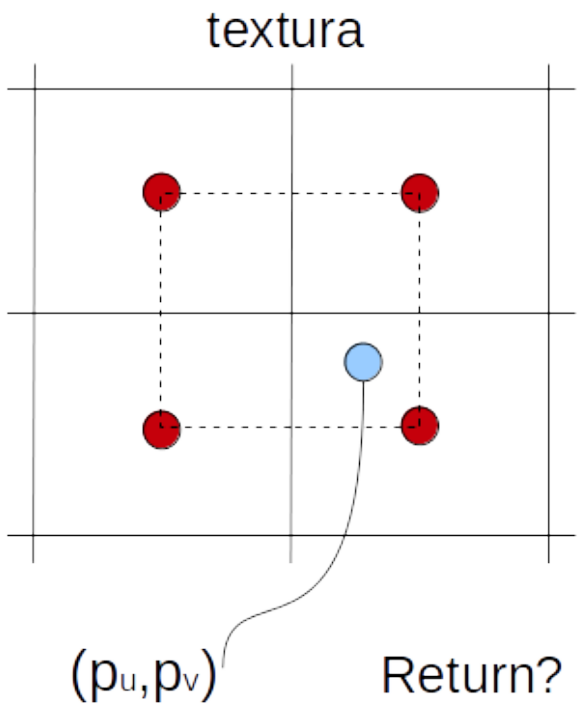


Magnificación

Bilineal filter

Utiliza cuatro *texeles* e interpola linealmente los valores. Efecto de borrosidad.

gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MAG_FILTER, gl.LINEAR);



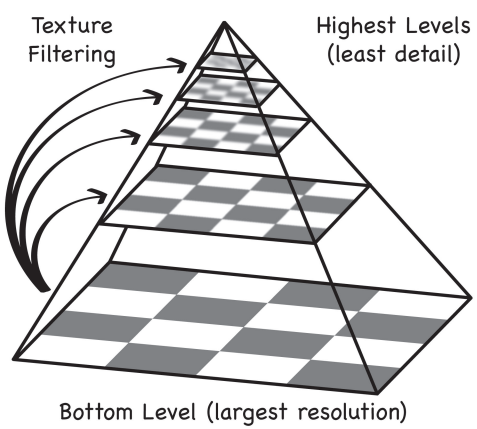
Minimización

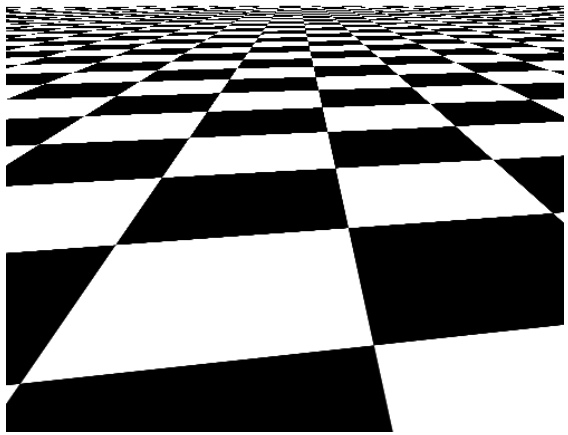
Los mismos filtros que en el problema anterior

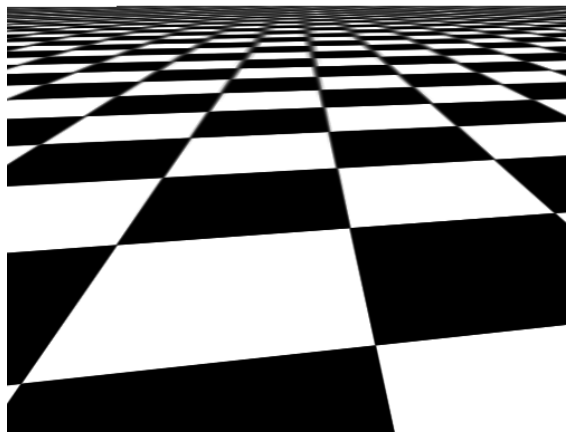
gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER, gl.NEAREST);
gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER, gl.LINEAR);

Mipmapping

Consiste en proporcionar además de la textura original un conjunto de versiones más pequeñas de la textura, cada una un cuarto más pequeña que la anterior.







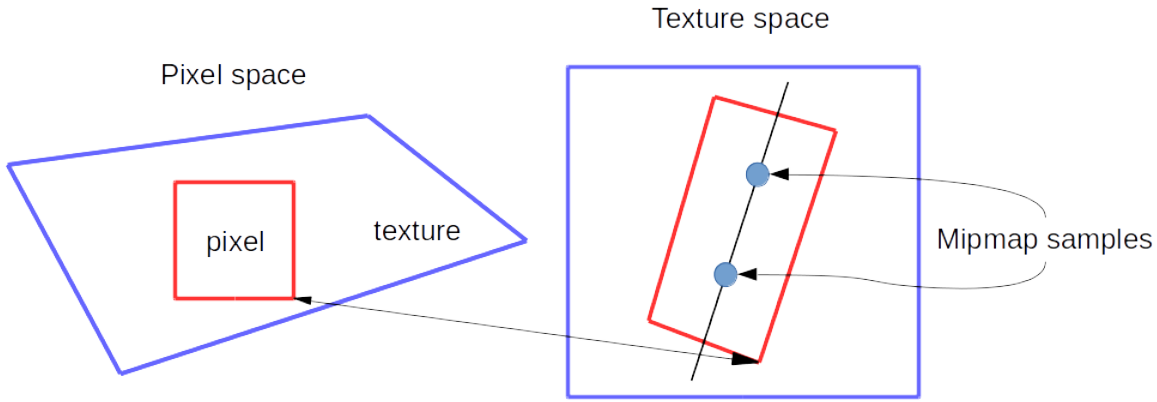
Minimización

Mipmapping

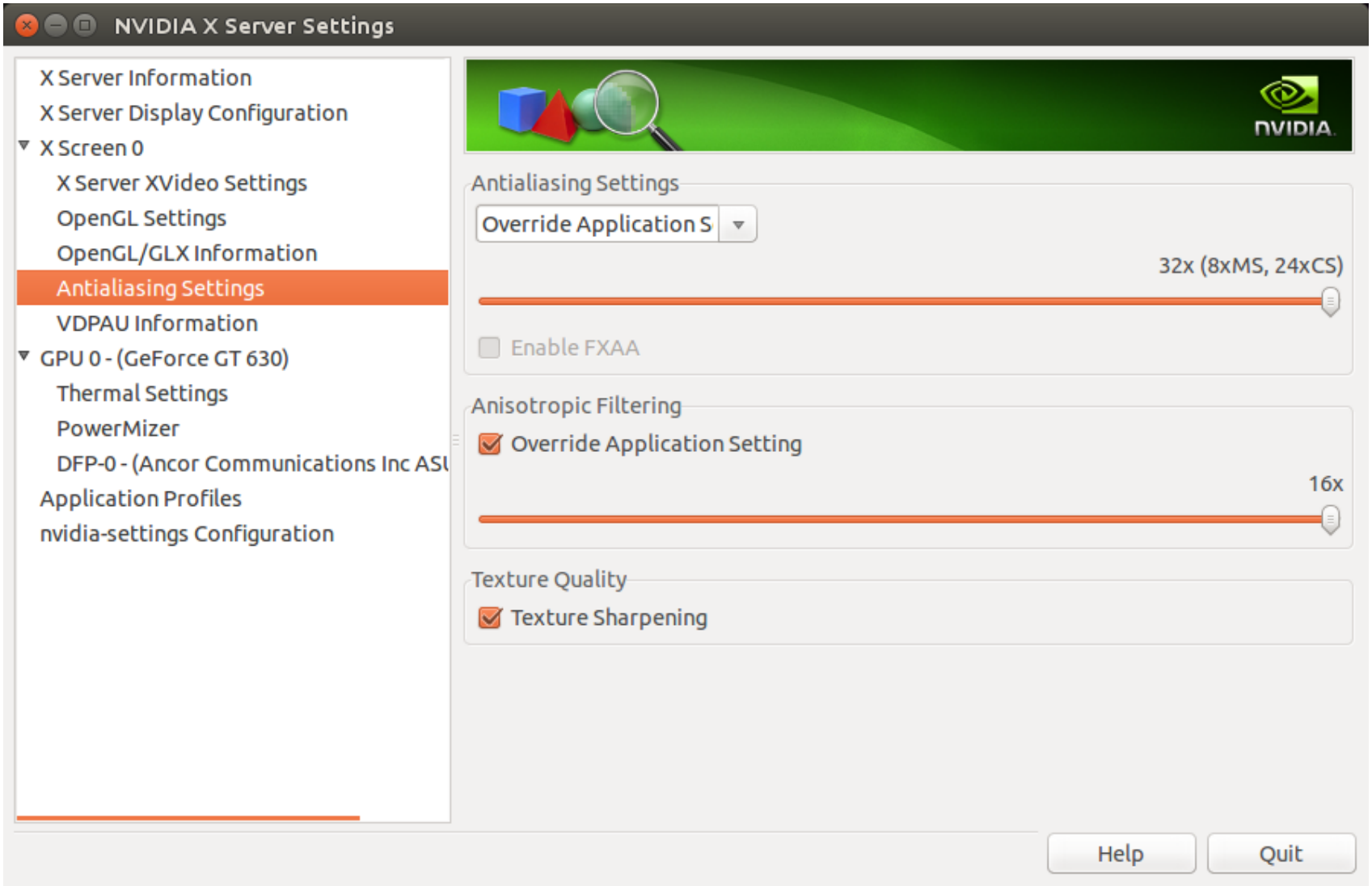
- Cada texel almacena información de cuatro texeles del siguiente nivel de más detalle.
- La GPU selecciona la textura cuyo tamaño más se acerca al tamaño de la textura en la pantalla.
- WebGL puede generar los niveles de manera automática:
  `gl.generateMipmap(gl.TEXTURE_2D);`
- Filtrado trilineal: selecciona dos texturas del mipmap, y cada una se muestrea utilizando un filtro bilineal. El color devuelto es una media ponderada de las dos muestras.
  `gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER, gl.LINEAR_MIPMAP_LINEAR);`

Filtrado Anisotrópico

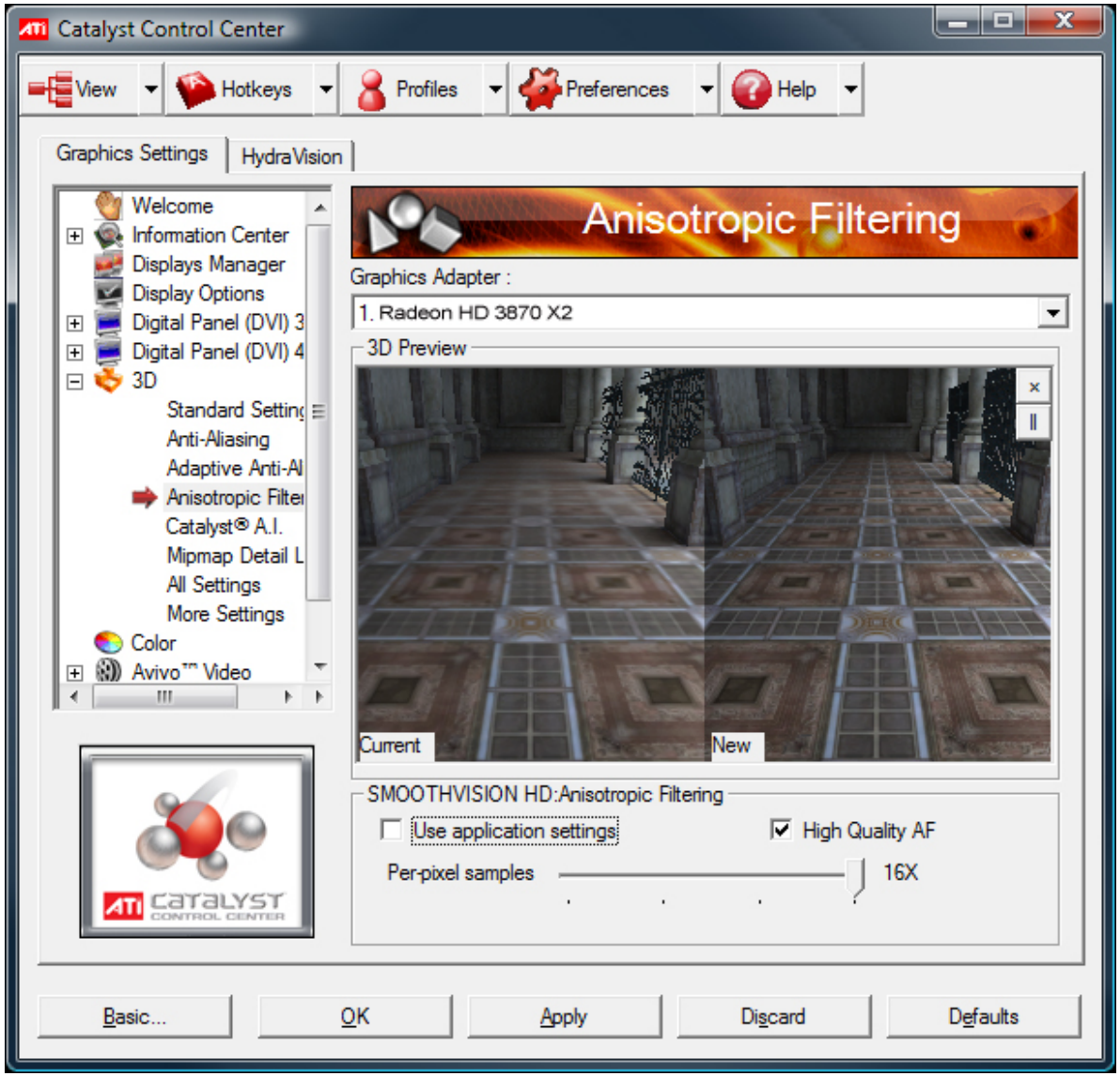
Utiliza varias muestras de un mipmap.



Nvidia Settings



Ati Settings

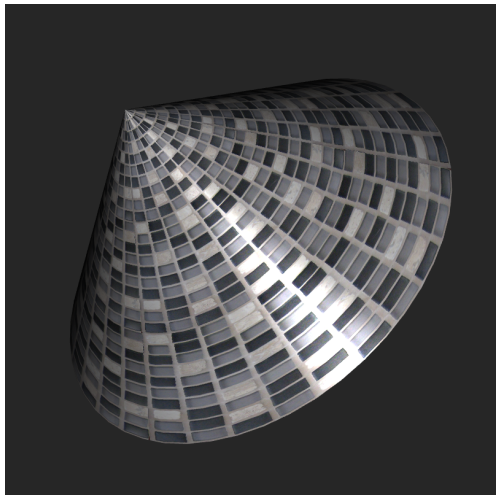
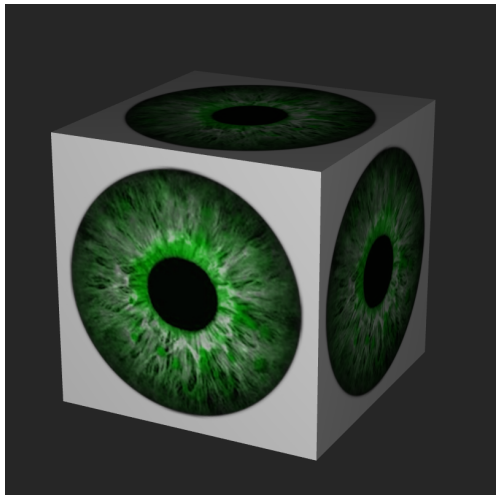
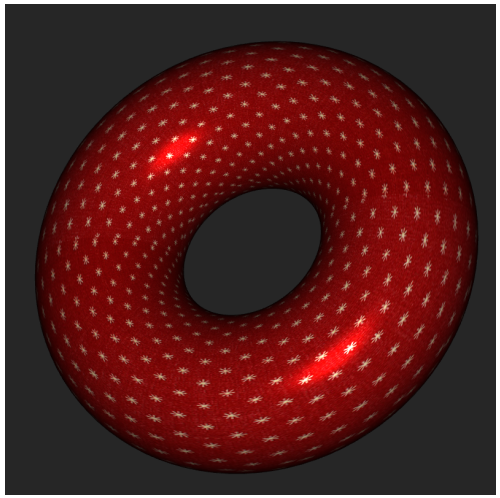


Hoy veremos...

Creación de una Textura 2D

Son tres pasos:

1. Crear un objeto textura.
2. Asignar la unidad de textura.
3. Especificar para cada vértice de la superficie sus coordenadas de textura.

Listado 3: Ejemplo de creación de una textura

```
// Crea un objeto textura
texture = gl.createTexture();
gl.bindTexture(gl.TEXTURE_2D, texture);

// Especifica la textura RGB
gl.texImage2D (gl.TEXTURE_2D, 0, gl.RGB, image.ancho, image.alto, 0, gl.RGB, gl.UNSIGNED_BYTE,
      image);

// Repite la textura tanto en s como en t
gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_WRAP_S, gl.REPEAT);
gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_WRAP_T, gl.REPEAT);

// Filtrado
gl.texParameteri (gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER, gl.LINEAR_MIPMAP_LINEAR);
gl.texParameteri (gl.TEXTURE_2D, gl.TEXTURE_MAG_FILTER, gl.LINEAR);
gl.generateMipmap(gl.TEXTURE_2D);
```

Listado 4: Asignación de objeto textura a unidad de textura

```
// Selecciona la unidad de textura 0
gl.activeTexture(gl.TEXTURE0);

// Asigna el objeto textura a la unidad de textura seleccionada
gl.bindTexture  (gl.TEXTURE_2D, texture);
```

Listado 5: Establecimiento de la unidad a la que accede el Sampler

```
// Obtiene el indice de la variable del Shader de tipo sampler2D
program.textureIndex = gl.getUniformLocation(program, 'myTexture');

// Indica que myTexture del Shader use la unidad de textura 0
gl.uniform1i(program.textureIndex, 0);
```

Hoy veremos...

Texturas 3D

Descripción

- Piensa en un bloque de material de piedra, madera, etc.

- De la misma manera, una textura 3D define un valor para cada punto en el espacio 3D.

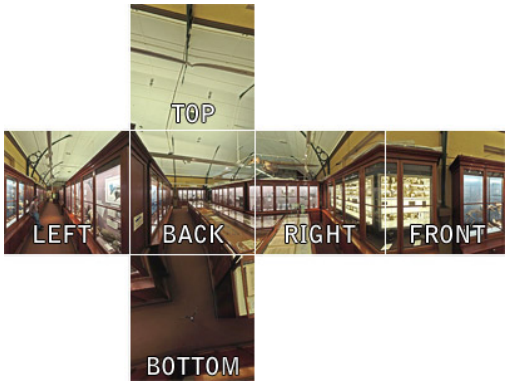
Texturas 3D

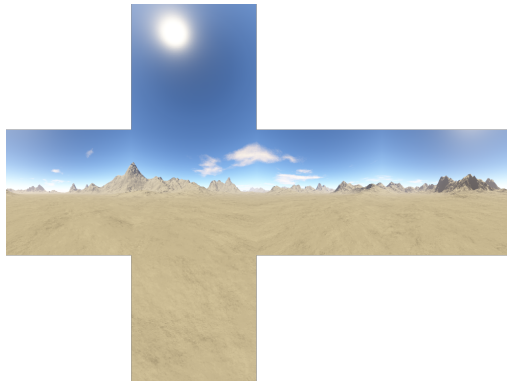
Características

- Son una extensión directa de las texturas 2D donde ahora se utilizan tres coordenadas (s, t, r).
- Ahora tenemos voxels en lugar de texels!!!
- Las propias coordenadas de los vértices se pueden utilizar como coordenadas de textura, es decir, no es necesario la parametrización de la superficie.
- Pero son caras de almacenar ... y también de filtrar (4x4 muestras).
- Ineficiente para superficies, la mayor parte de los vóxeles no se utilizan.
- WebGL 2.0 soporta este tipo de texturas así como los mismos tipos de filtrado, (pero WebGL 1.0 no).

Hoy veremos...

Cube Maps

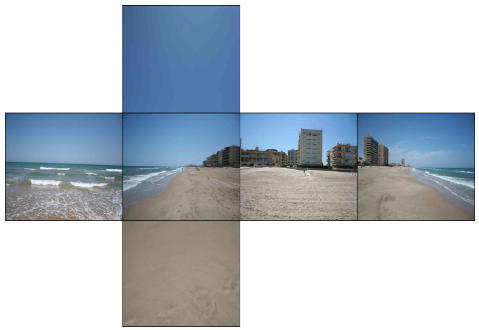


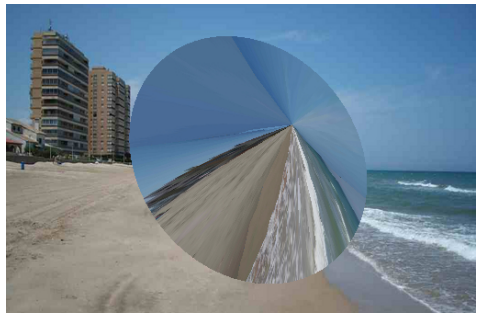


Descripción

- Seis texturas cuadradas que juntas capturan un entorno.
- Cada una se pega a una cara de un cubo.
- Soportado en WebGL.
- ¿Para qué sirve?
  - Reflection maps (environment maps)
  - Refraction maps
  - Skyboxes

Reflection mapping

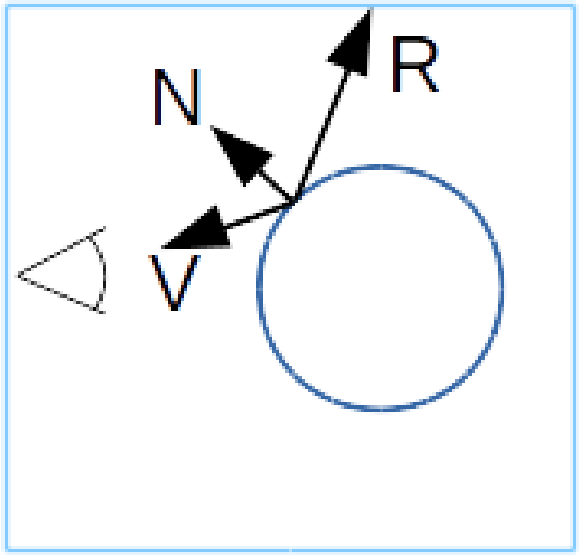




Reflection mapping

Características

- El objetivo es simular objetos que reflejan su entorno.
- Las coordenadas de textura no se suministran, se calculan!!
- Y cambian con la posición del observador.



Reflection mapping

¿Cómo se obtiene el téxel?

- Para cada punto de la superficie del objeto reflejante se obtiene el vector de reflexión respecto a la normal en ese punto de la superficie.
- ¿Cuál de las seis texturas se ha de utilizar? Se elige la coordenada de mayor magnitud. Si es la coordenada R_x, se utilizan la cara derecha o izquierda del cubo, dependiendo del signo.
- Después, hay que obtener las coordenadas s y t para acceder a la textura seleccionada.
- Ejemplo en el caso de ser R_x:

$$s = (\frac{-R_z}{|R_x|} + 1)/2 \qquad t = (\frac{-R_y}{|R_x|} + 1)/2 \tag{7}$$

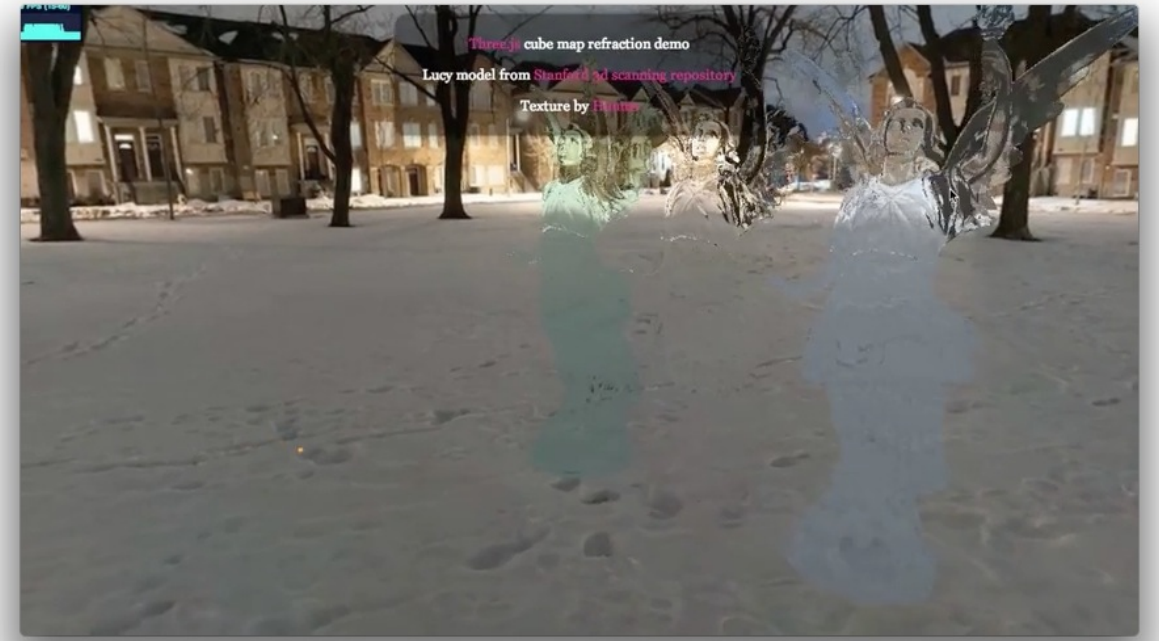
El Shader para Reflection mapping

Listado 6: Shader para Reflection Mapping

```
// Vertex shader
...
in  vec3 VertexNormal;
out vec3 R;

void main()  {
  ...
  // igual que en el modelo de iluminacion de Phong
  vec4 ecPosition = modelViewMatrix * vec4(vertexPosition,1.0);
  vec3 N          = normalize(normalMatrix * vertexNormal);
  vec3 V          = normalize(vec3(-ecPosition));

  ...
  R = reflect(-V, N);
}

// Fragment shader
...
uniform samplerCube myCubeMapTexture;     // la textura

in vec3 R;

void main()
{
  ...
  // acceso a la textura para obtener un valor de color RGBA
  fragmentColor = texture(myCubeMapTexture, R);
}
```

Video ejemplo

Refraction Mapping



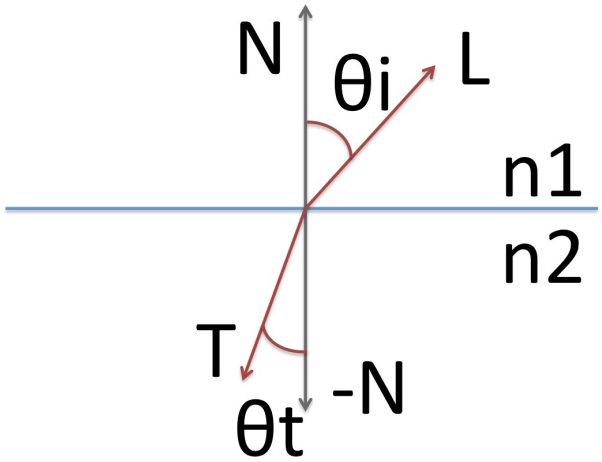
Objetivo

- Representar objetos que la luz atraviesa: hielo, agua, vidrio, esmeralda, rubí, diamente, etc.

Refraction Mapping

Cálculo

- Se utiliza la ley de Snell: $n_1 \cdot \sin\theta_i = n_2 \cdot \sin\theta_t$
- Al cociente entre n_1 y n_2 se le denomina índice de refracción.
- Por ejemplo: aire (1.0), agua (1.33), vidrio (1.52), diamante (2.42), etc.

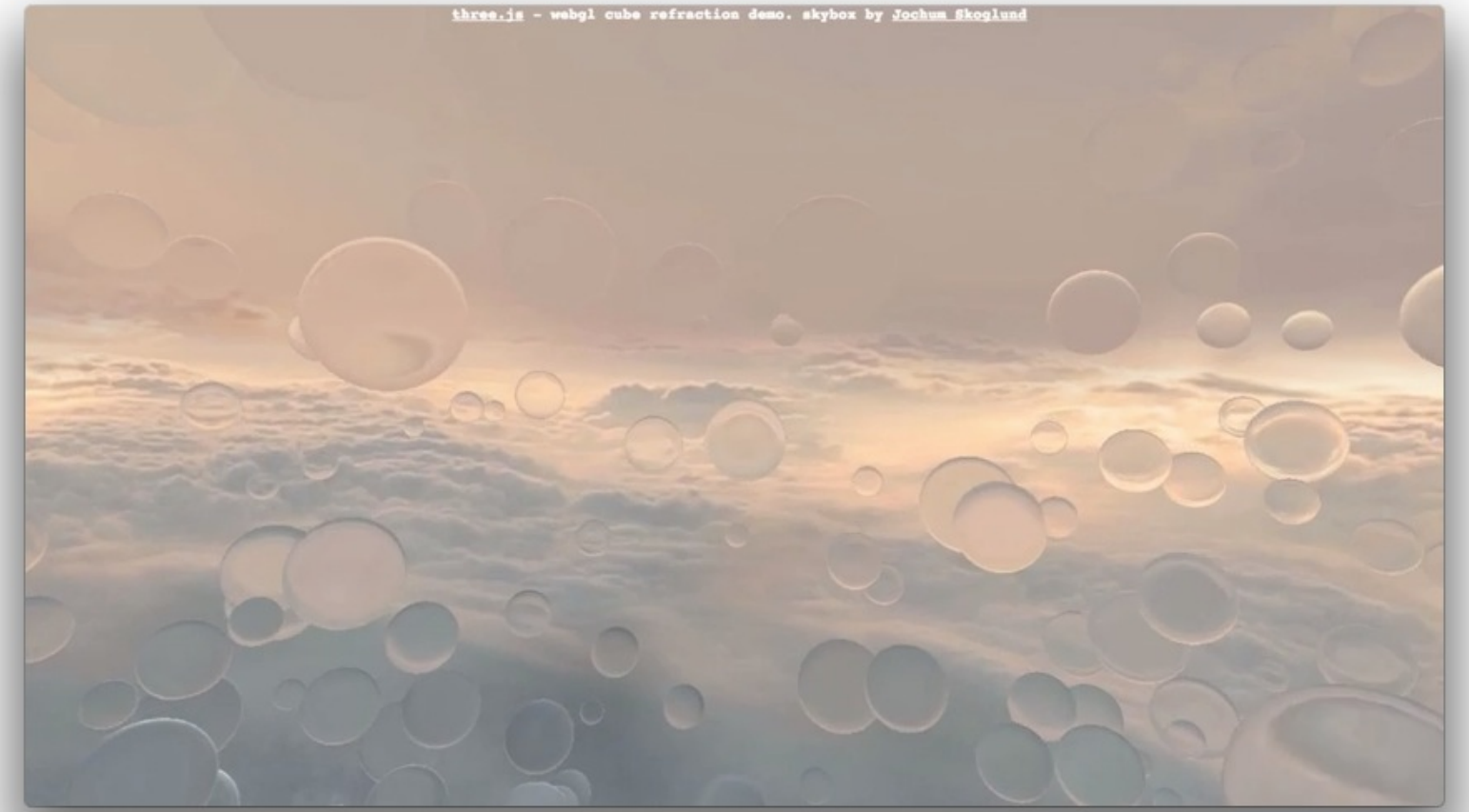


El Shader para Refraction Mapping

Listado 7: Shader para Refraction Mapping

```
// Vertex shader
...
in  vec3 VertexNormal;
out vec3 RefractDir, ReflectDir;

void main()  {
  ...
  ReflectDir = reflect(-V, N);
  RefractDir = refract(-V, N, material.indice_de_refraccion);
}

// Fragment shader
...
uniform samplerCube myCubeMapTexture;     // la textura

in vec3 RefractDir, ReflectDir;

void main()
{
  ...
  // acceso a la textura para obtener dos valores de color RGBA
  fragmentColor = mix (texture(myCubeMapTexture, RefractDir),
                       texture(myCubeMapTexture, ReflectDir),
                       material.refractionFactor);
}
```

Video ejemplo

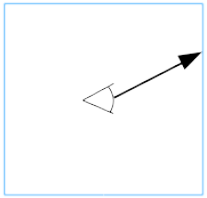


Skyboxes

Objetivo

- Representar el fondo de una escena. Contiene elementos muy distantes al observador, típicamente el sol, montañas, nubes, etc.
- Se utiliza un cubo muy grande, alrededor del observador.

El Shader para Skybox



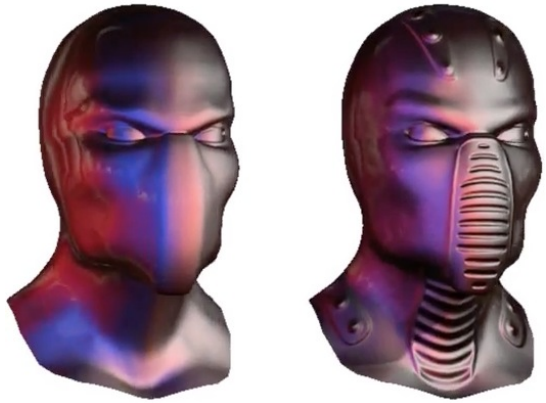
Listado 8: Shader para Skybox

```
// Vertex shader
...
in  vec3 VertexPosition;
out vec3 R;

void main()  {

  // simplemente asigna a R las coordenadas del vertice
  R = VertexPosition;
   ...
}

// Fragment shader
...
uniform samplerCube myCubeMapTexture;     // la textura
in vec3 R;

void main() {
   ...
  // acceso a la textura para obtener un valor de color RGBA
  fragmentColor = texture(myCubeMapTexture, R);
}
```

Video ejemplo

Hoy veremos...

Normal Mapping



Descripción

- Consiste en modificar la normal de la superficie para dar la ilusión de rugosidad.
- La geometría es la misma.
- Las normales modificadas se precalculan y se almacenan en un *Normal map o Bump map*.
- El *Normal map* se porporciona a la GPU como textura y contiene valores de color.

Normal Mapping

¿Cómo operamos?

- El cálculo de la iluminación se realiza en el espacio de la tangente.
- Necesitas conocer el vector tangente para cada vértice.
- En el procesador de vértices:
  - El tercer vector, llamado binormal, lo obtienes mediante el producto vectorial de la normal y la tangente.
  - Con los tres vectores creas la matriz que realiza el cambio de base de manera que la normal coincida con el Z, la tangente con el X yla binormal con el Y.
  - Operas el vector L y el vector V por esta matriz, y los resultados los interpolas para cada fragmento.
- En el procesador de fragmentos:
  - Obtienes la normal, N, del mapa de normales.
  - Y con N, L, y V aplicas el modelo de iluminación.

Hoy veremos...

Displacement Mapping

Descripción

- Consiste en aplicar un desplazamiento en cada vértice.
- El caso más sencillo es aplicarlo en la dirección de la normal.
- El desplazamiento se puede almacenar en una textura a la que se le conoce como mapa de desplazamiento.
- Desde el *vertex shader* se accede a la textura y se modifica la posición del vértice sumándole el resultado de multiplicar el despazamiento por la normal en el vértice.

Otro Ejemplo

Video ejemplo

Hoy veremos...

Alpha Mapping

Descripción

- Consiste en utilizar una textura para determinar qué partes son visibles.
- Como una plantilla por ejemplo.
- Para cada fragmento se accede a la textura y el valor devuelto te indicará si el fragmento debe continuar o no, por ejemplo.

Ejemplos