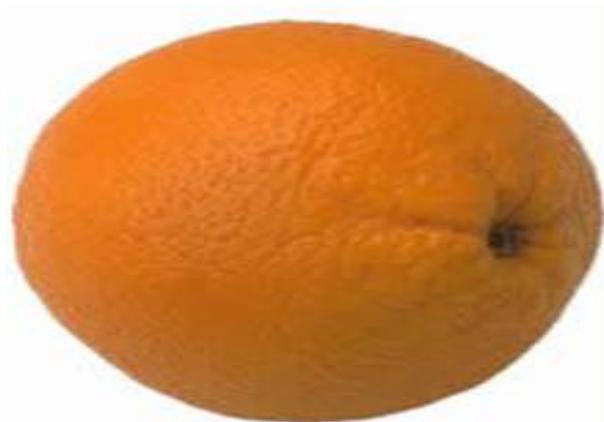


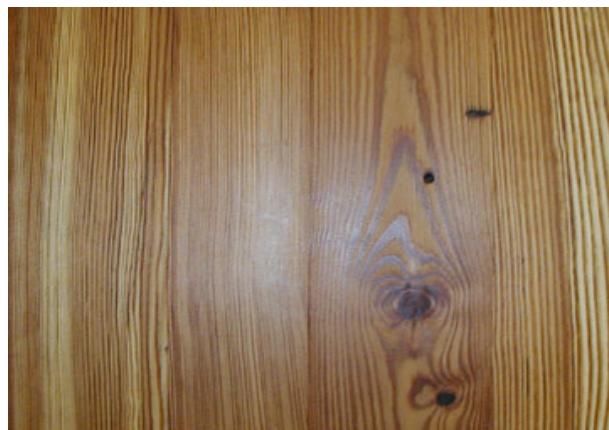
Textures

Carlos Andujar
Setembre 2013

Representació de detalls superficials

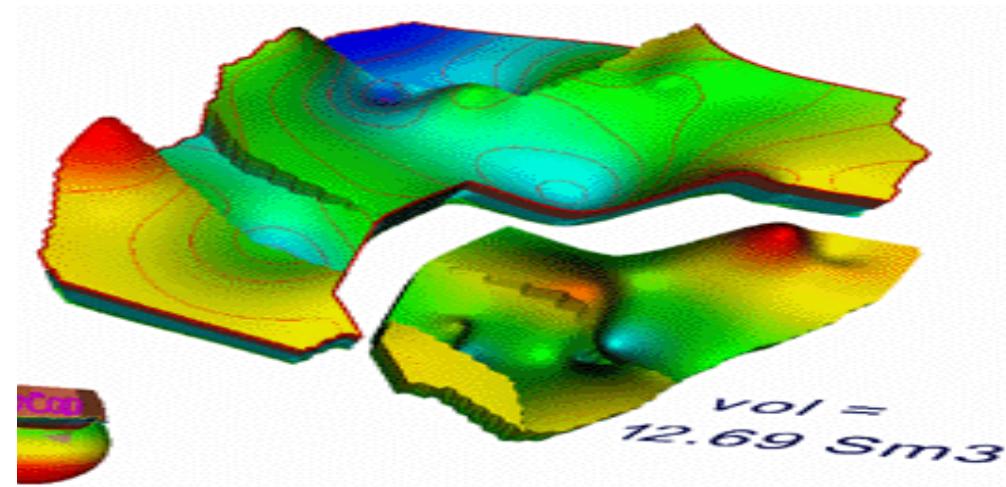
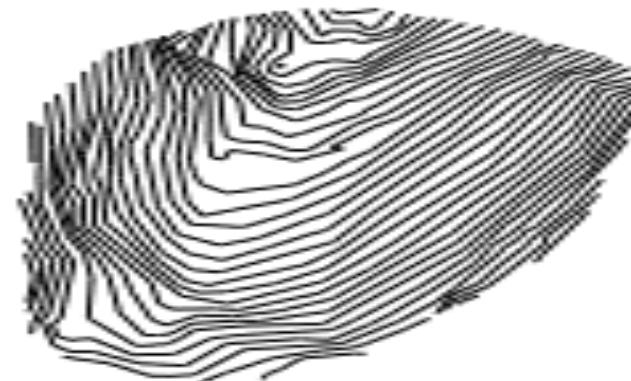


Deguts a variacions de la geometria

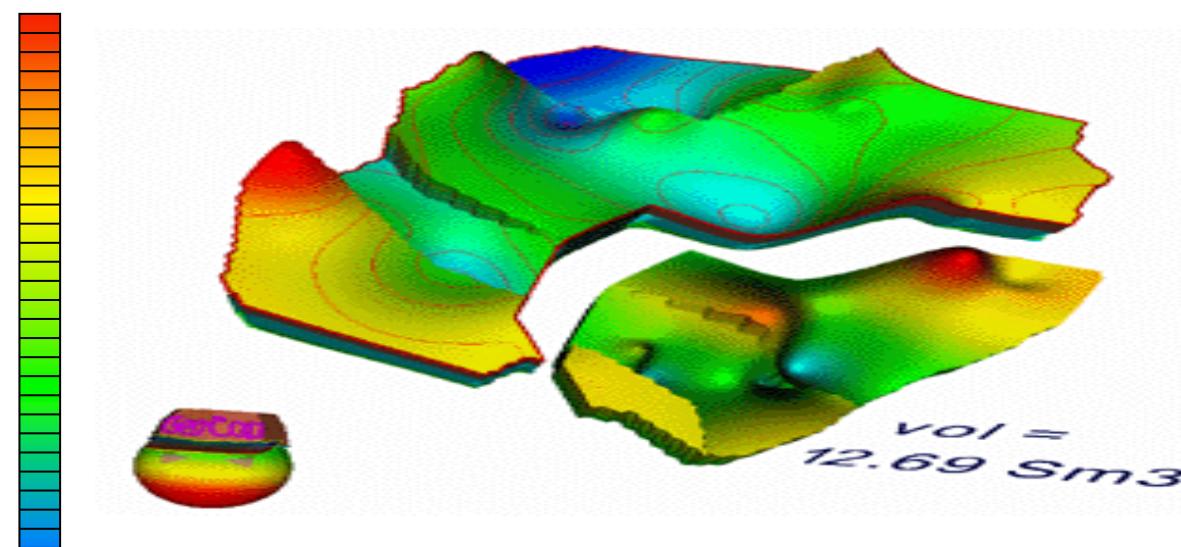
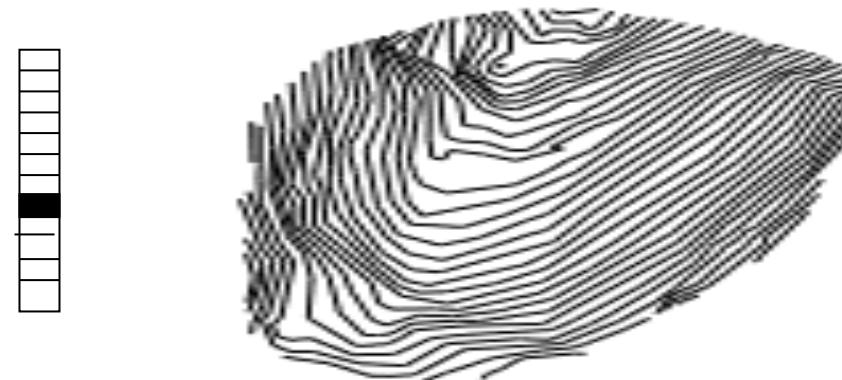


Deguts a variacions de les propietats òptiques

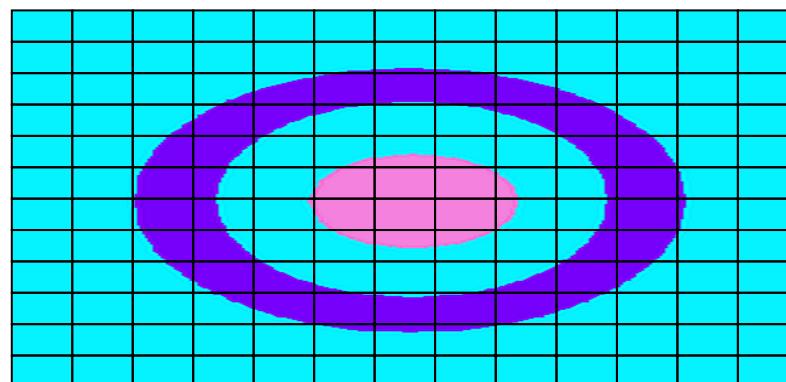
Dimensió de les textures. Textures 1D



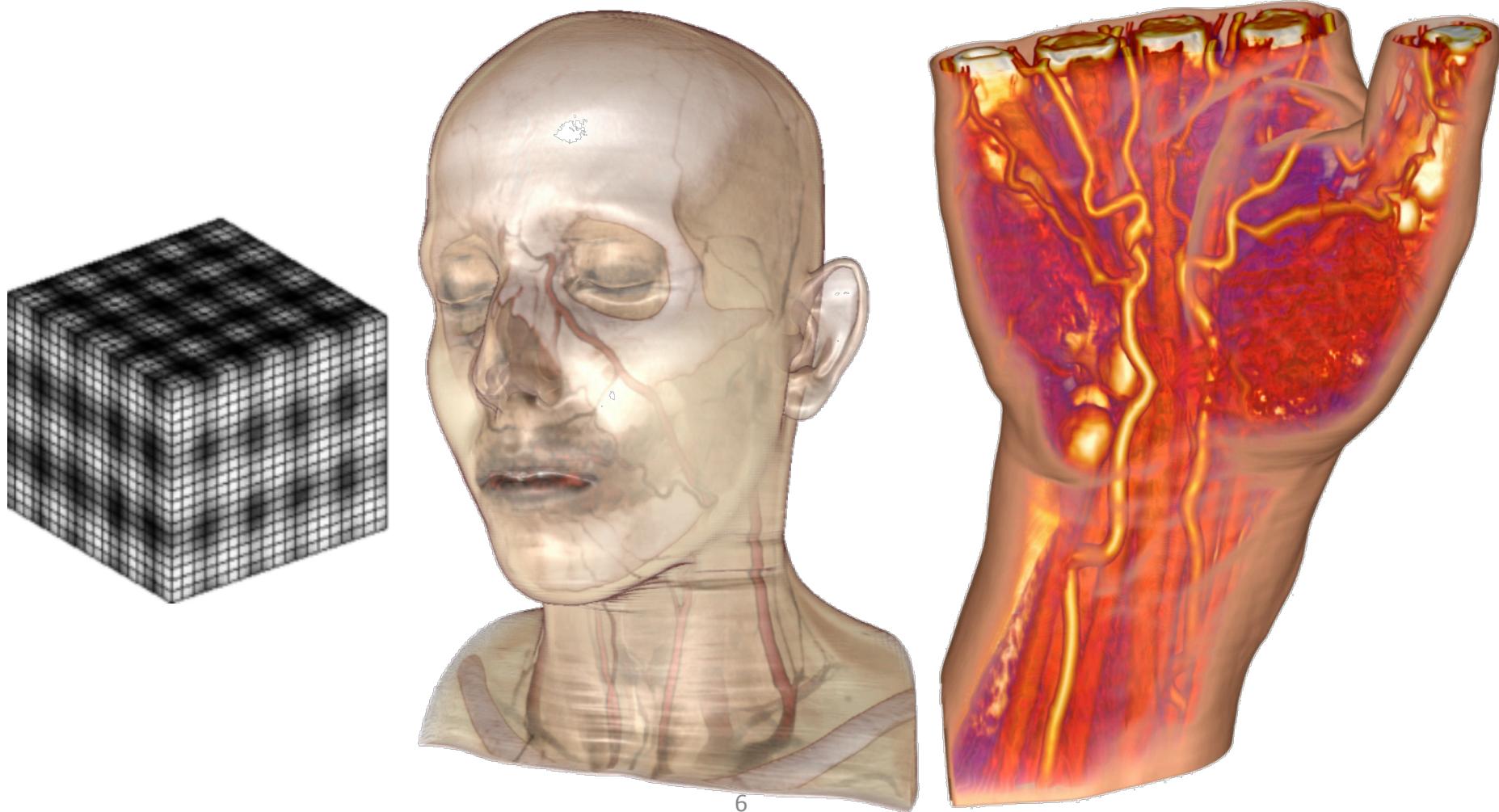
Dimensió de les textures. Textures 1D



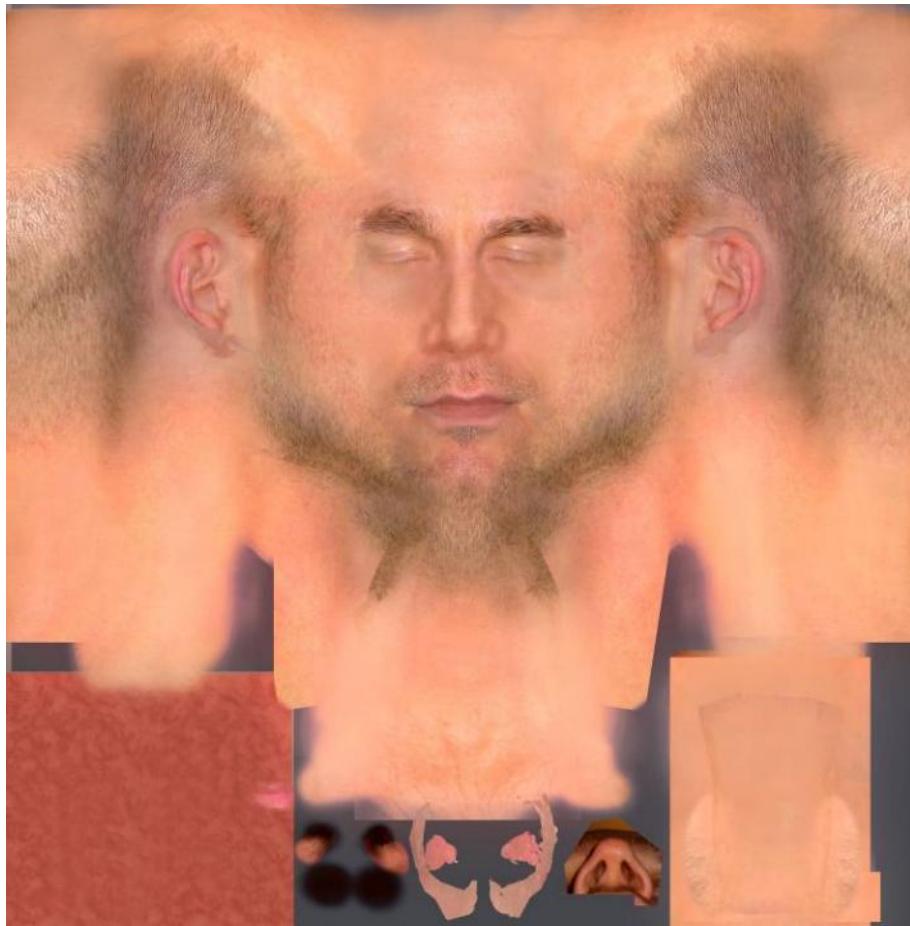
Dimensió de les textures. Textures 2D



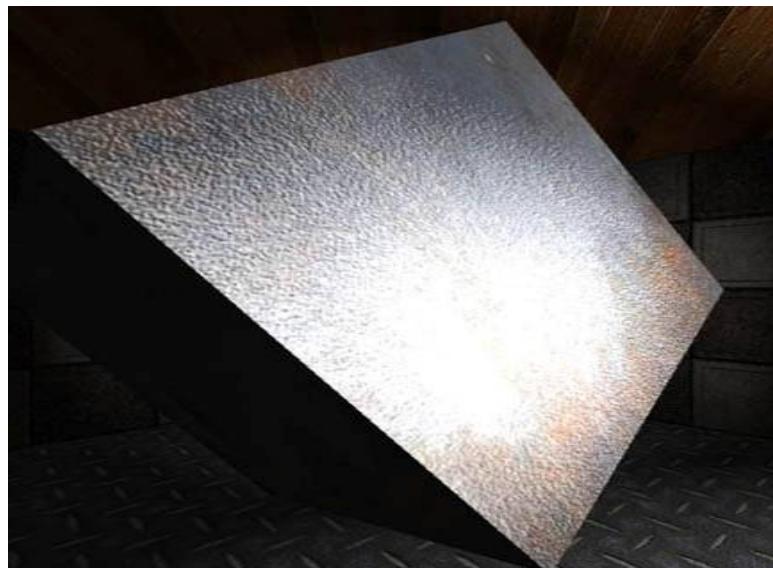
Dimensió de les textures. Textures 3D



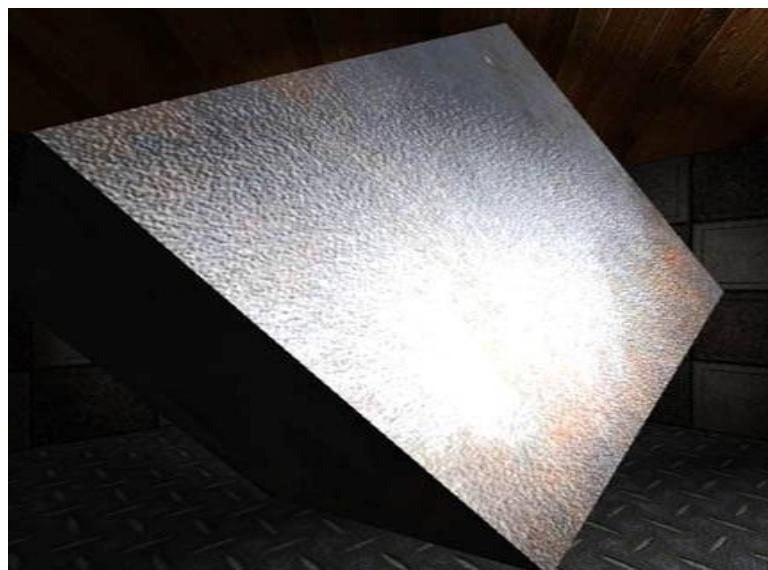
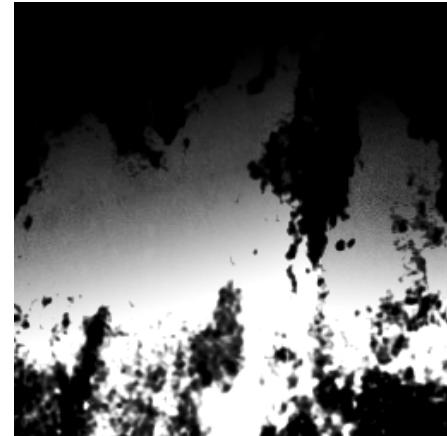
Tipus de textures: K_d (color map)



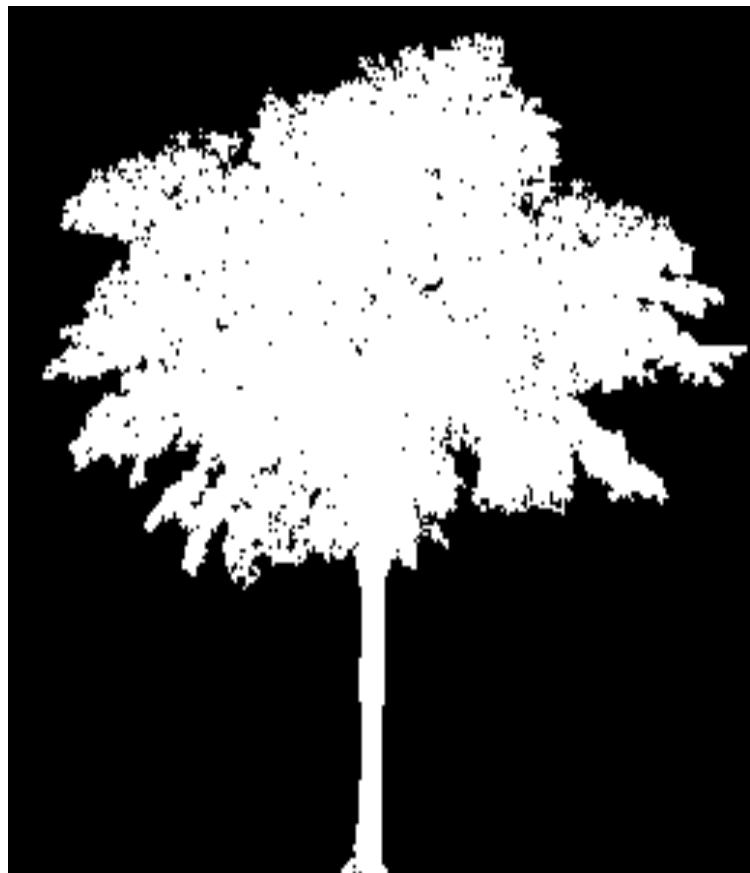
Tipus de textures: K_s (gloss map)



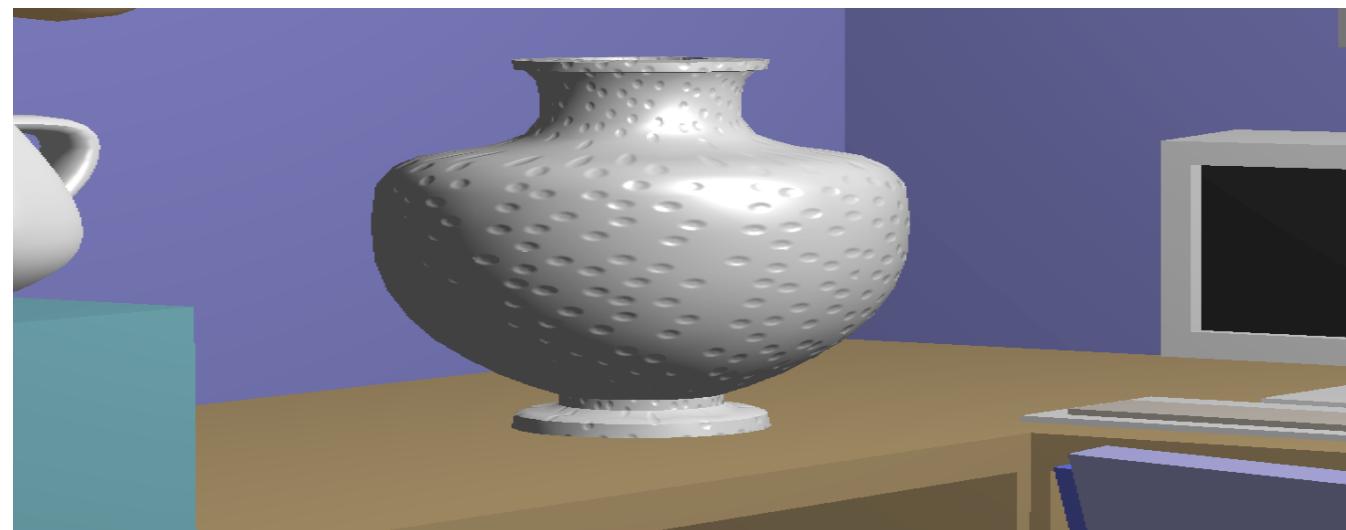
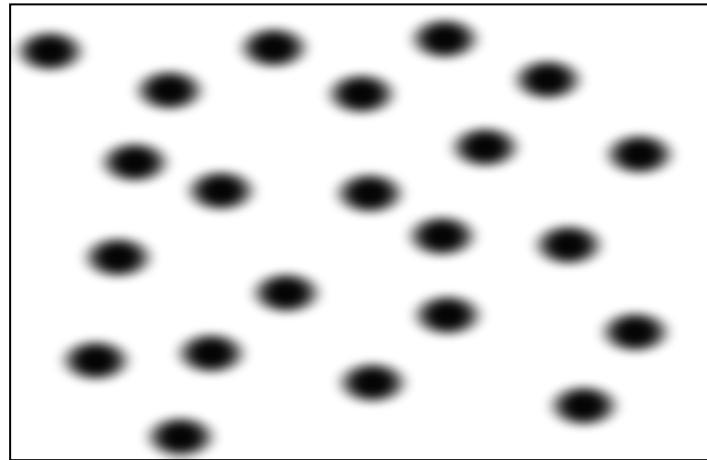
Tipus de textures: K_s (gloss map)



Tipus de textures: alpha (opacity map)



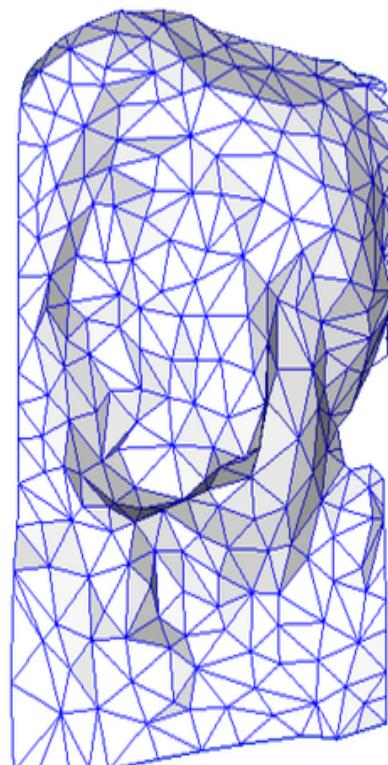
Tipus de textures: bump mapping



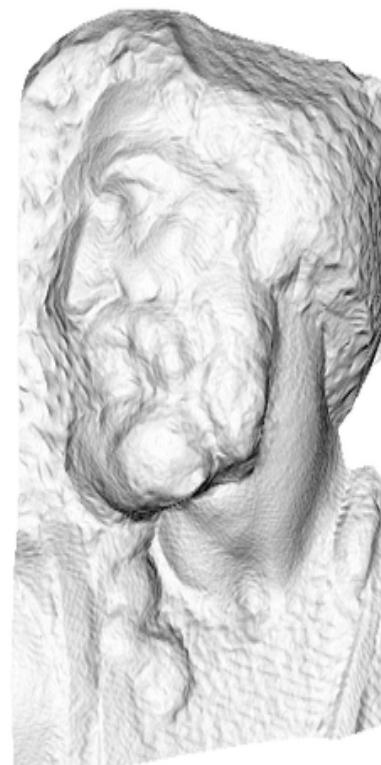
Tipus de textures: normal mapping



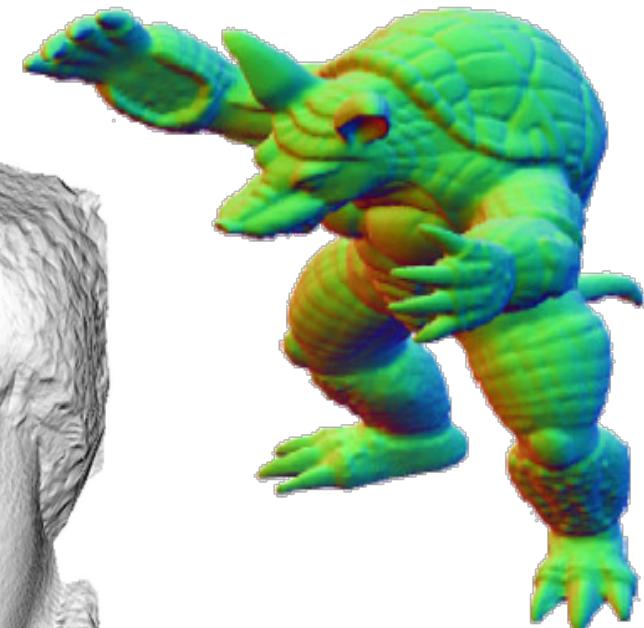
original mesh
4M triangles



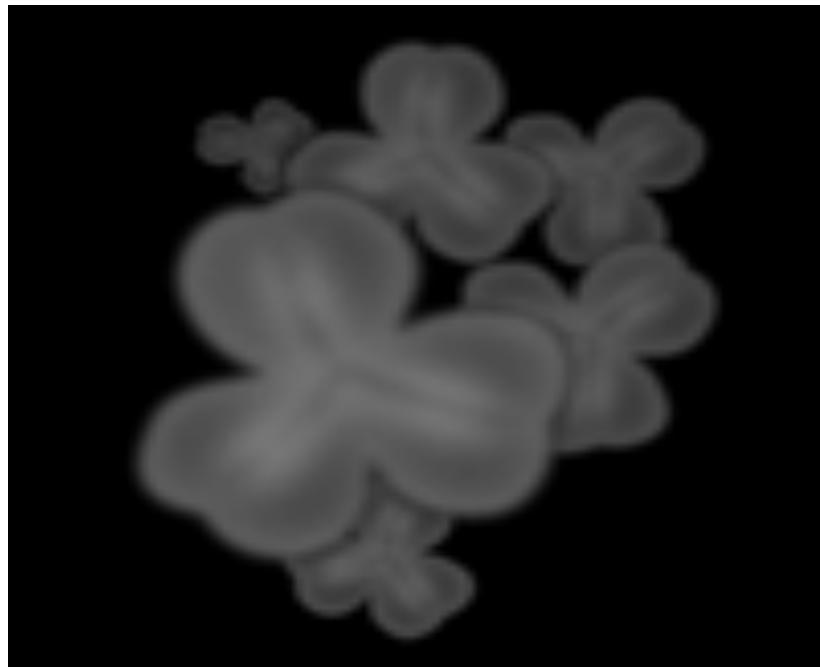
simplified mesh
500 triangles



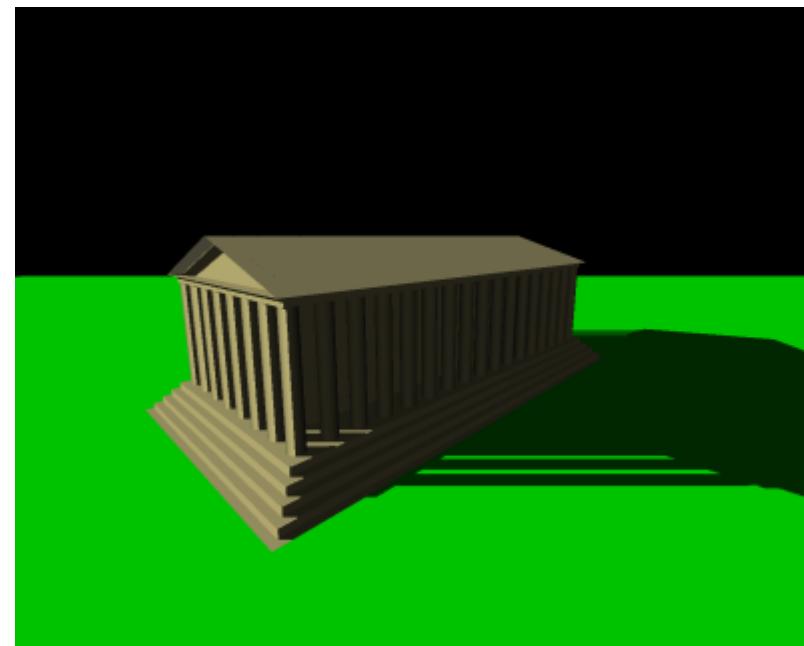
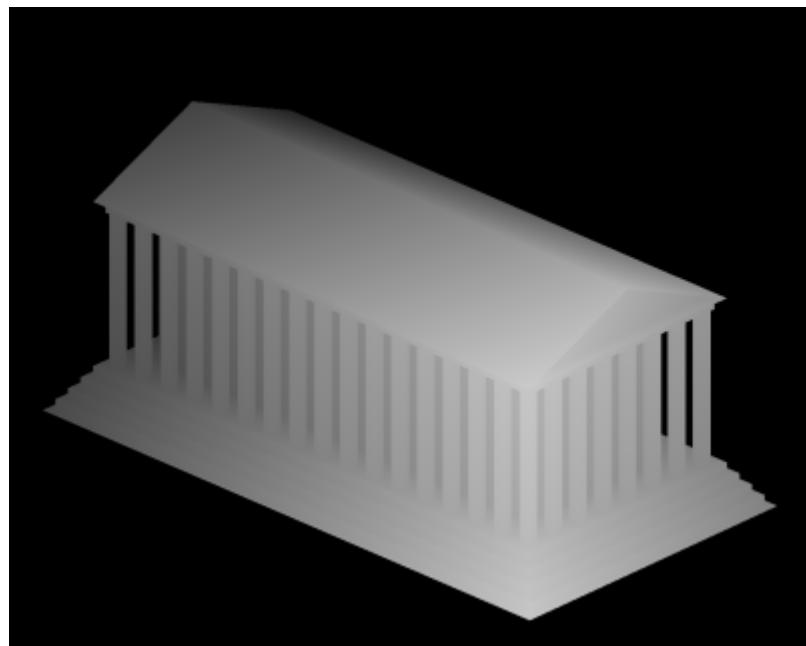
simplified mesh
and normal mapping
500 triangles



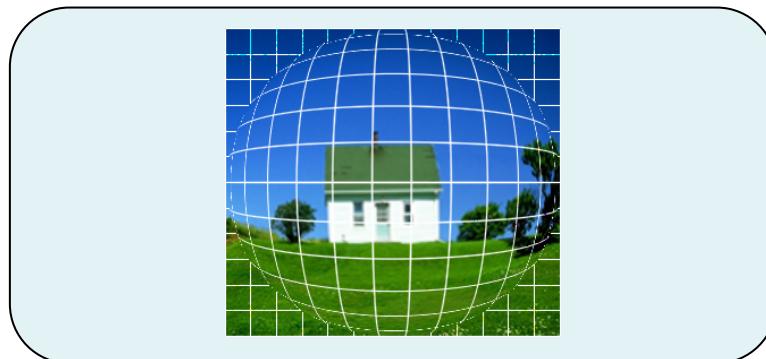
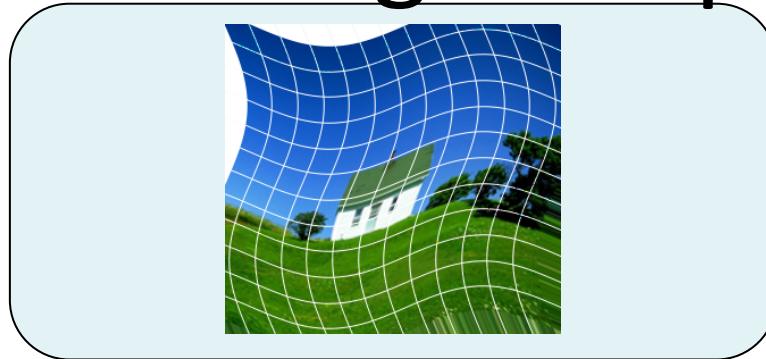
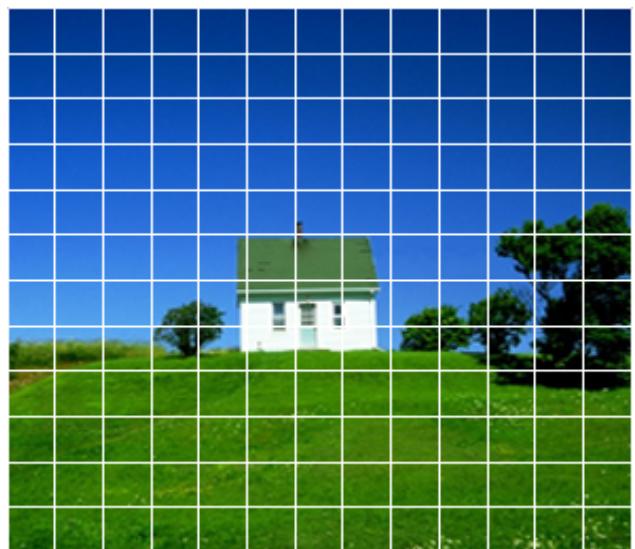
Tipus de textures: displacement map



Tipus de textures: shadow maps



Mapping directe: image warping

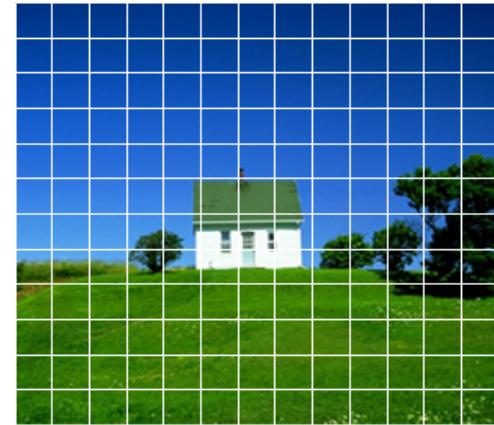


Exemple 1: Swirl

Input: $s \in [0,1]$, $t \in [0,1]$

Output: $x \in [0,1]$, $y \in [0,1]$

```
dist = distance(s, t, sc, sc) * sqrt(2);
angle = angleMax * (1-dist);
// rotació respecte (sc, sc)
s -= sc;
t -= sc;
x = s*cos(angle) - t*sin(angle) + sc;
y = s*sin(angle) + t*cos(angle) + tc;
x *= width;
y *= height;
```



Exemple 2: Mapping esfèric

Input: $s \in [0,1]$, $t \in [0,1]$

Output: $x, y, z \in$ esfera unitat

// pas $(s, t) \rightarrow (\Theta, \Psi)$

$\Theta = 2\pi s;$

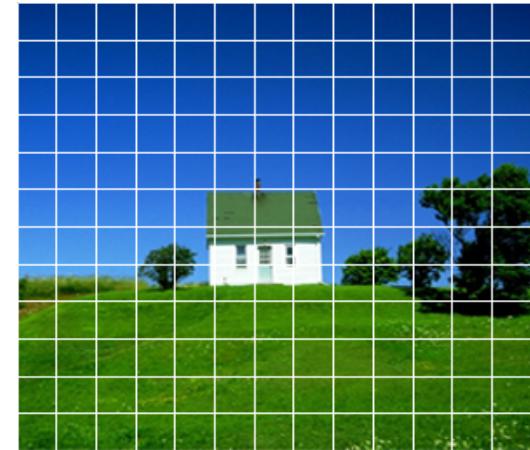
$\Psi = \pi(t-0.5);$

// pas esfèriques $\rightarrow (x, y, z)$

$x = \sin(\Theta)\cos(\Psi);$

$y = \sin(\Psi);$

$z = \cos(\Theta)\cos(\Psi);$



Exemple 3: Mapping cilíndric

Input: $s \in [0,1]$, $t \in [0,1]$

Output: $x, y, z \in$ cilindre sobre pla XZ

```
// pas  $(s, t) \rightarrow (\Theta, h)$ 
```

```
 $\Theta = 2\pi s;$ 
```

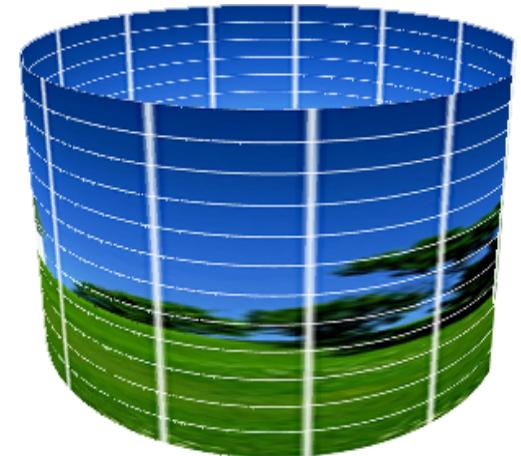
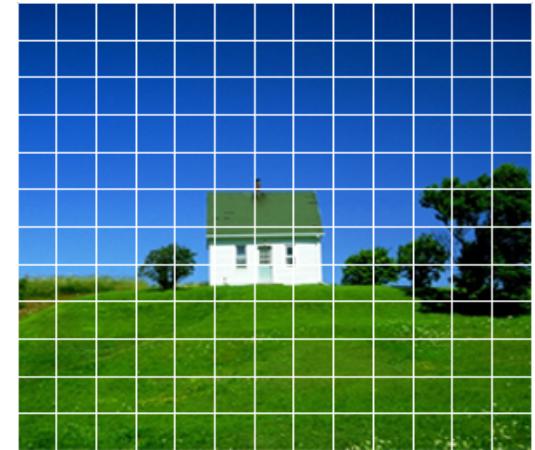
```
 $h = t;$ 
```

```
// pas cilíndriques  $\rightarrow (x, y, z)$ 
```

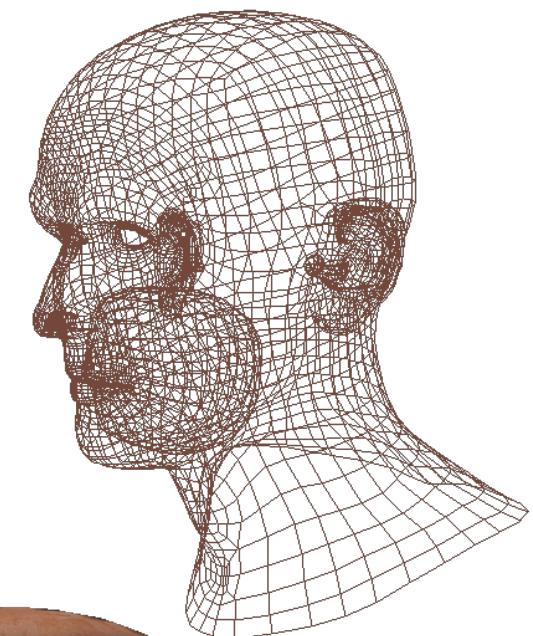
```
 $x = \sin(\Theta) * r;$ 
```

```
 $y = h;$ 
```

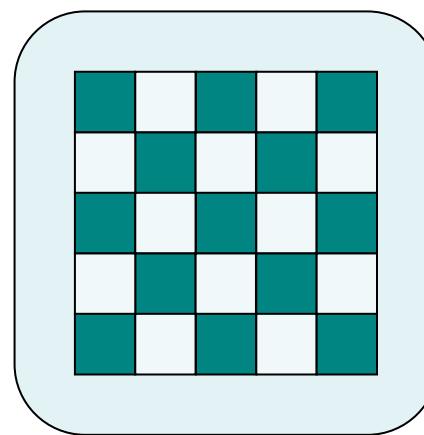
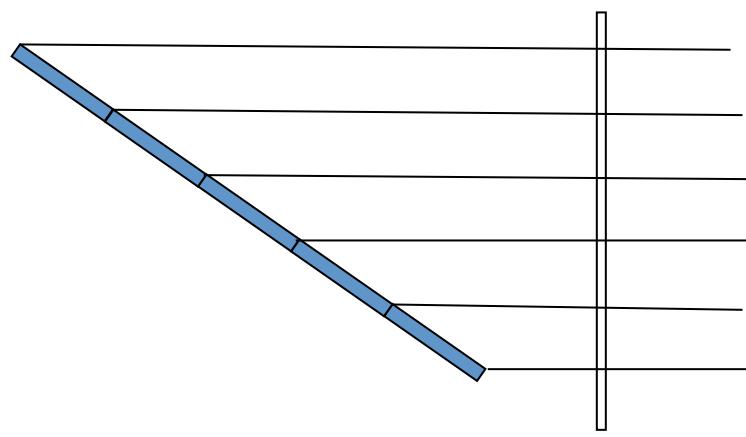
```
 $z = \cos(\Theta) * r;$ 
```



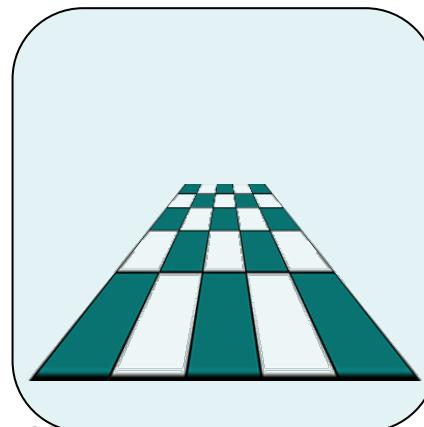
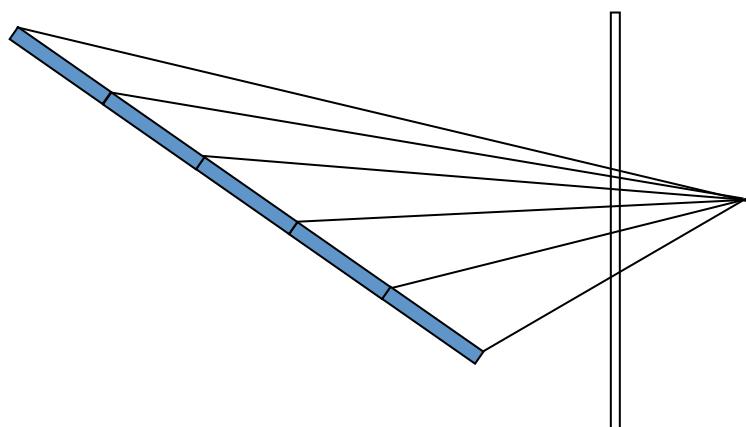
Mapping invers



Interpolació segons la projecció



Axonomètrica



Perspectiva

Magnification filters, Minification filters, Mipmapping

FILTRAT

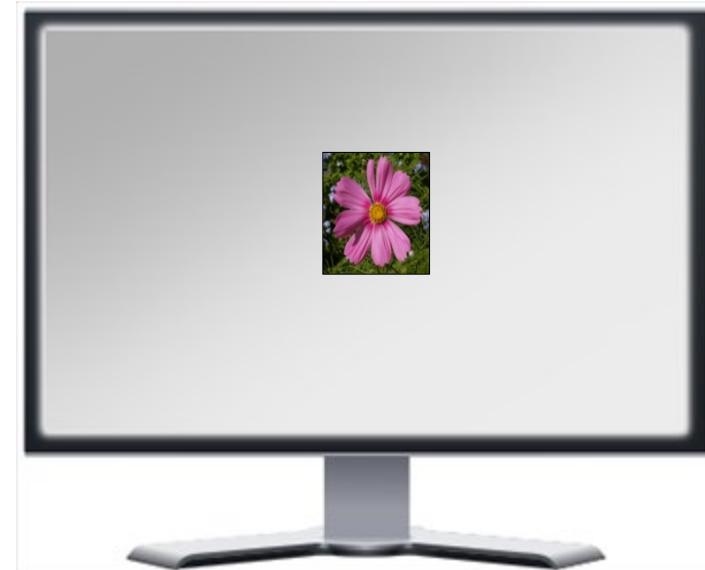
Necessitat del filtrat



Textura

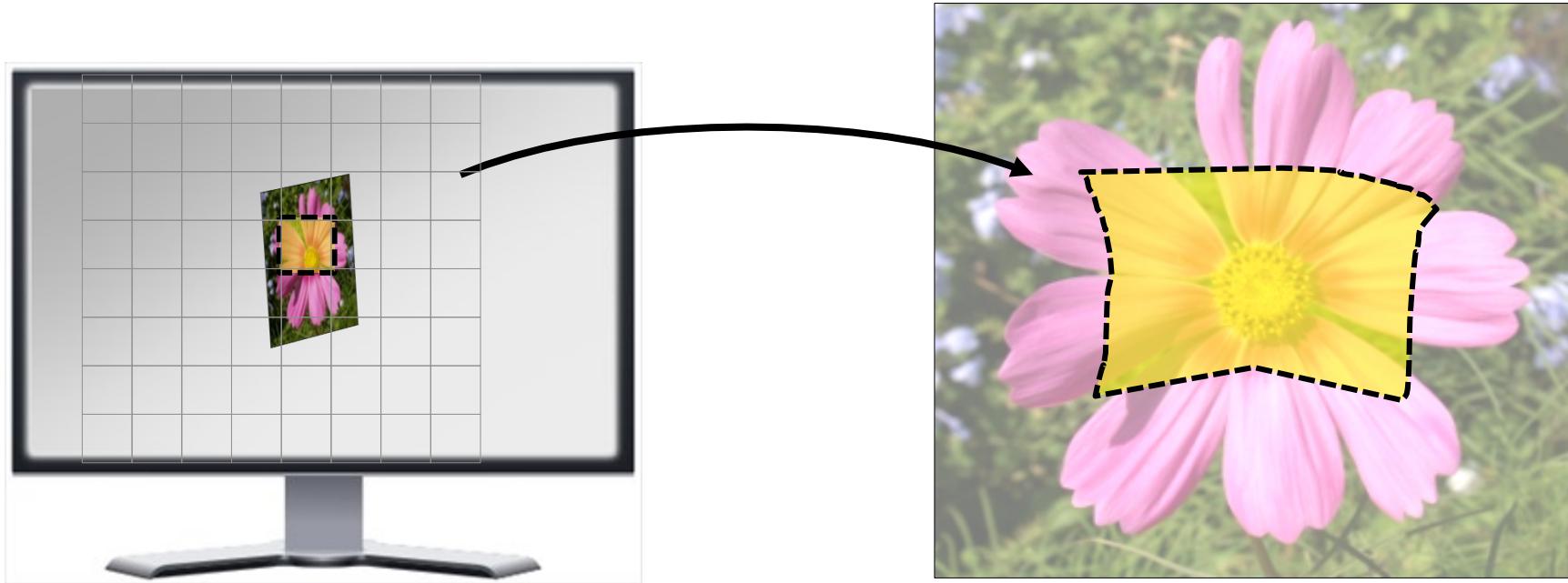


Magnification \approx upsampling



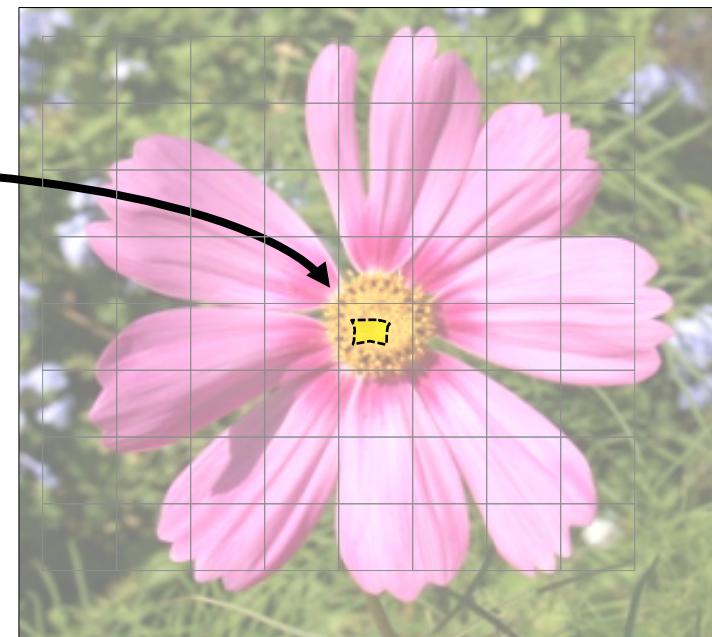
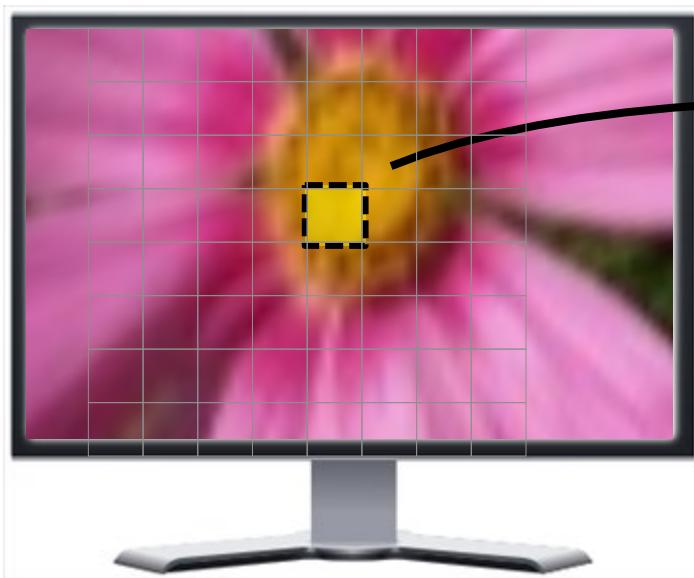
Minification \approx downsampling

Necessitat del filtrat



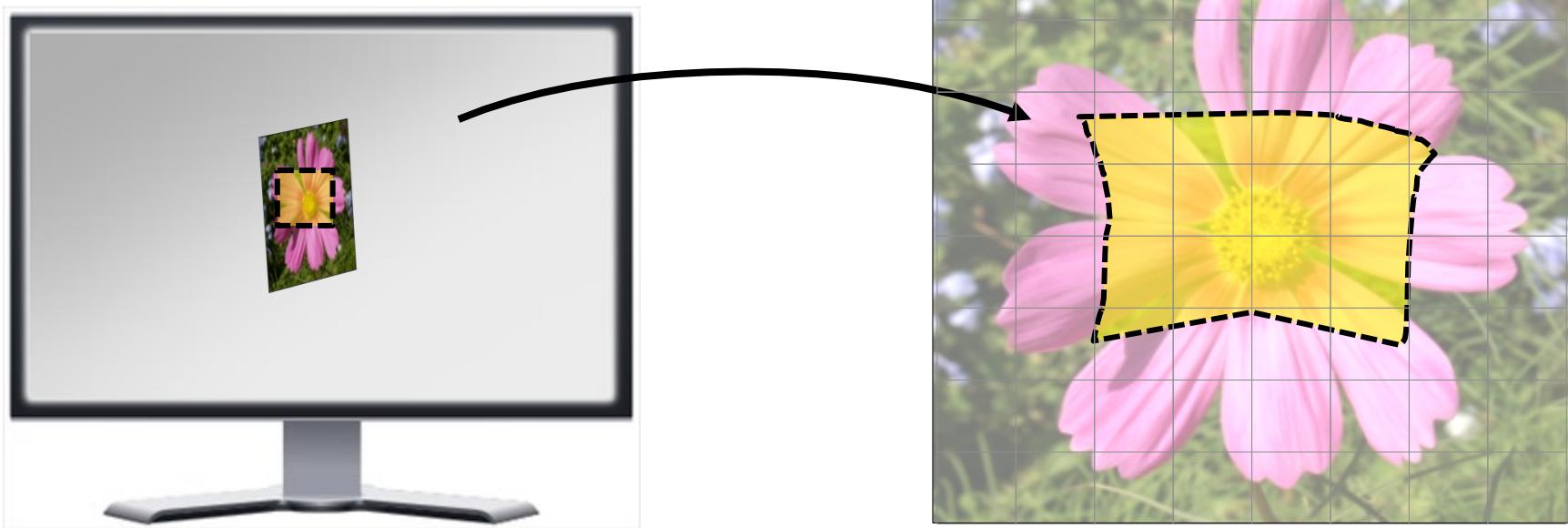
Ideal: color d'un pixel → color de la seva *preimatge* a la textura

Magnification



Magnification → la preimatge és < texel

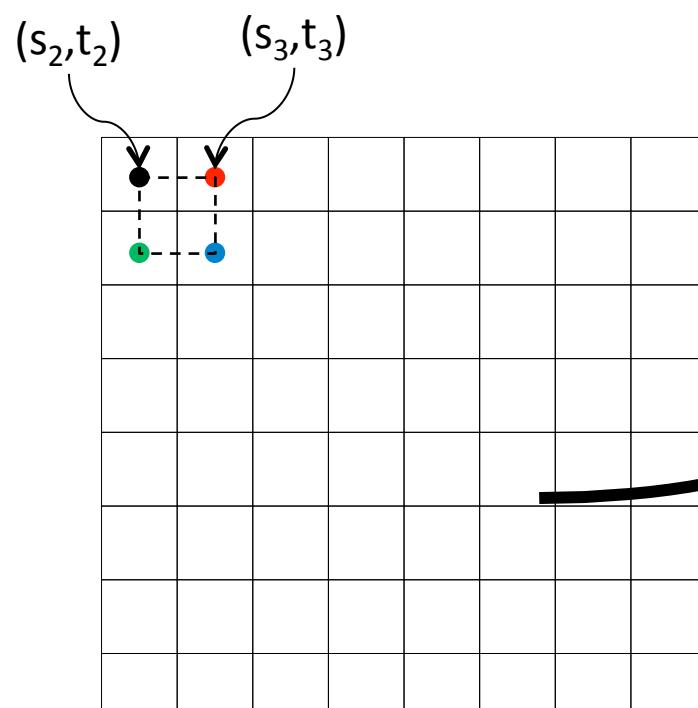
Minification



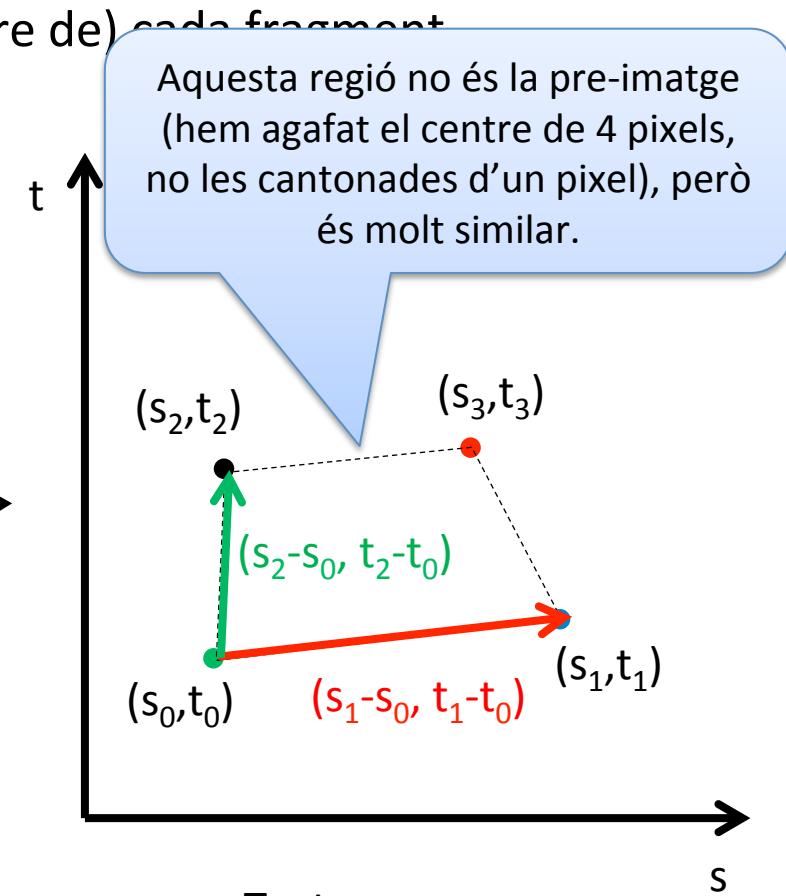
Minification → la preimatge és > texel

Aproximació que fa OpenGL

- En general no sabem la forma exacta de la pre-imatge...
- Però sí coneixem les coords (s, t) de (el centre de) cada fragment



Window space



Aproximació que fa OpenGL

- Cada fragment (x,y) té les seves coordenades $(s,t) \rightarrow$
- (s,t) depenen de (x,y) i podem escriure $s(x,y), t(x,y) \rightarrow$
- Té sentit calcular les derivades parcials de $s(x,y)$ i $t(x,y)$

$$\frac{\partial s}{\partial x} \approx s(x+1,y) - s(x,y)$$

$$\frac{\partial s}{\partial y} \approx s(x,y+1) - s(x,y)$$

- Anàlogament per t :

$$\frac{\partial t}{\partial x} \approx t(x+1,y) - t(x,y)$$

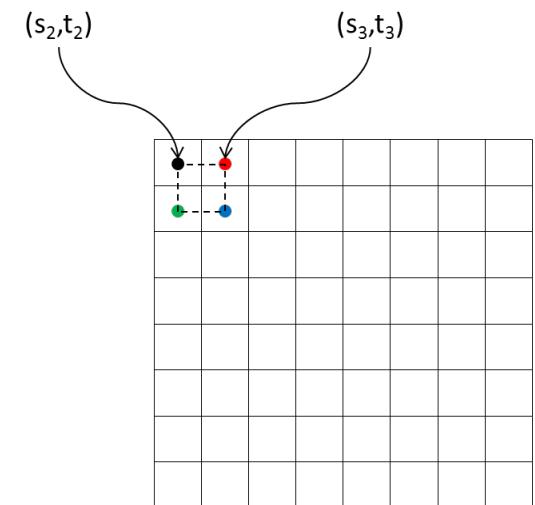
$$\frac{\partial t}{\partial y} \approx t(x,y+1) - t(x,y)$$

- En GLSL es calculen amb $dFdx, dFdy$:

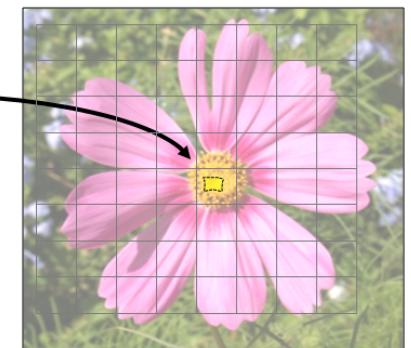
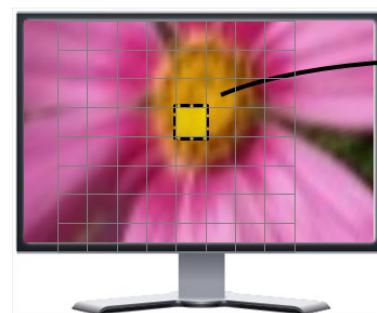
$$\frac{\partial s}{\partial x} \approx dFdx(gl_TexCoord[0].s)$$

$$\frac{\partial s}{\partial y} \approx dFdy(gl_TexCoord[0].s)$$

(anàlogament per t)



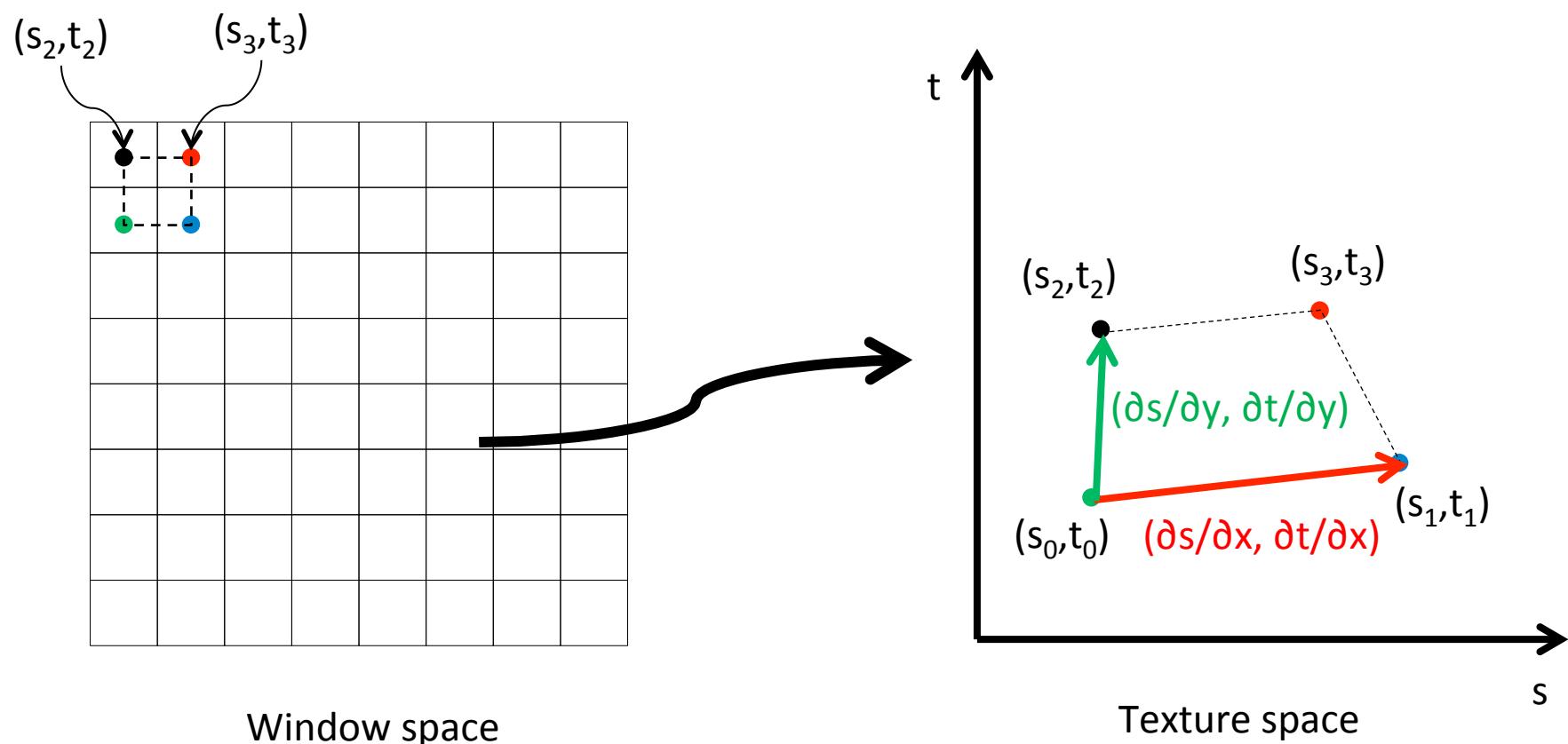
Window space



Magnification → la preimatge és < texel

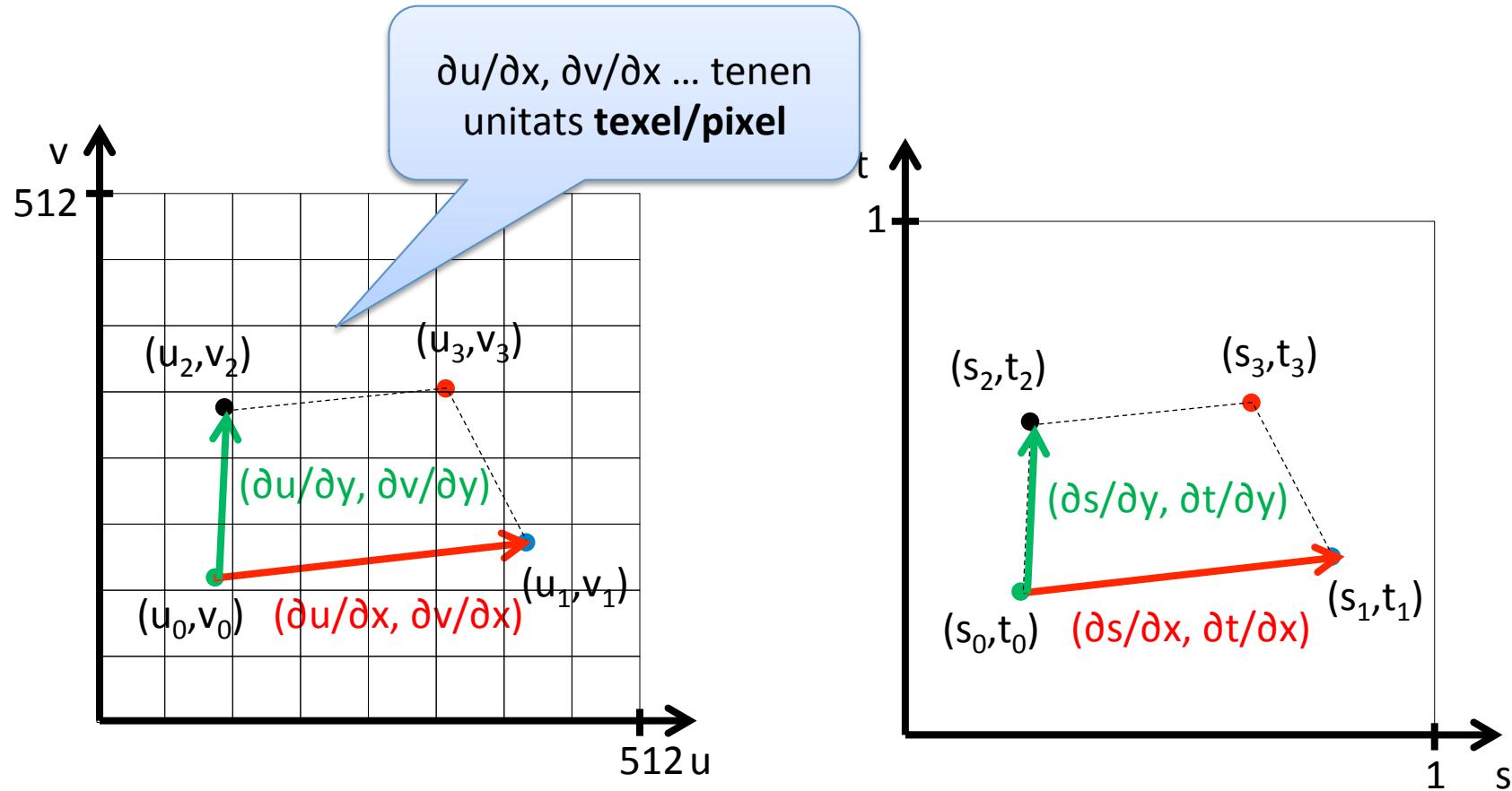
Aproximació que fa OpenGL

Aproximació de la mida de la pre-imatge (en espai normalitzat de textura)



Aproximació que fa OpenGL

Mida de la pre-imatge (**en texels**) amb una textura 512x512



Exemple 1 (mapping 1:1)



Polígon de WxH pixels

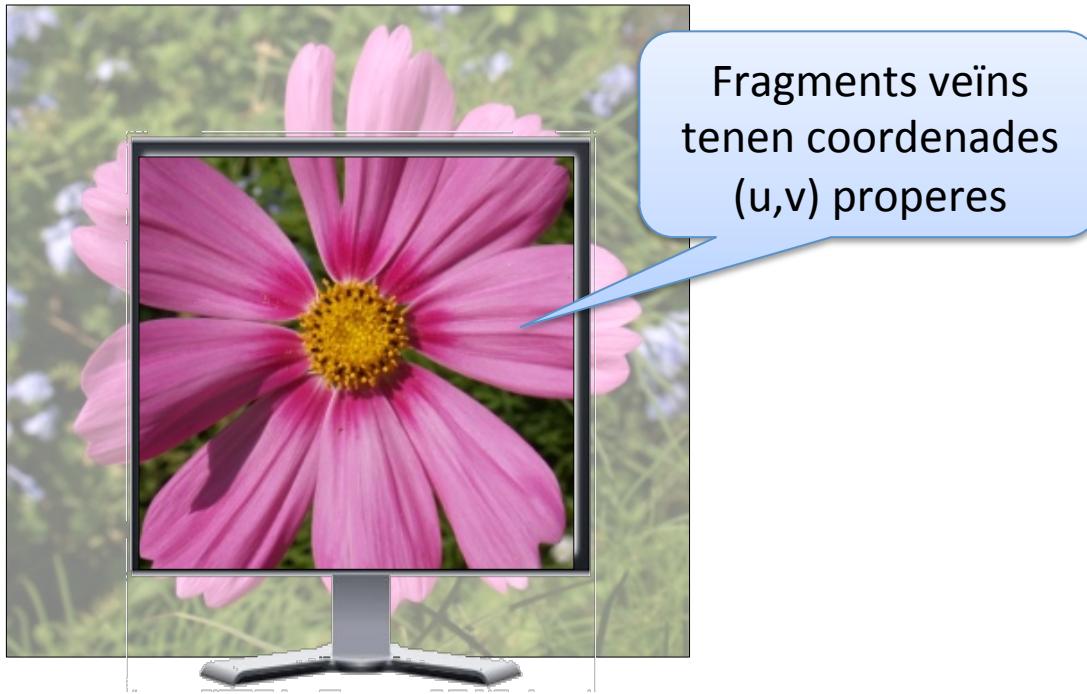


Textura WxH texels

En aquest cas un pixel correspon a un texel:

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y} = 1 \quad \frac{\partial v}{\partial x} = \frac{\partial u}{\partial y} = 0$$

Exemple 2 (magnification x2)



Textura WxH texels

Polígon projectat en $2W \times 2H$ pixels

En aquest cas un pixel correspon a mig texel:

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y} = \frac{1}{2} \quad \frac{\partial v}{\partial x} = \frac{\partial u}{\partial y} = 0$$

Exemple 3 (minification x2)



Fragments veïns
tenen coordenades
(u,v) més separades



Textura WxH texels

Polígon projectat en $\frac{1}{2} W \times \frac{1}{2} H$ pixels

En aquest cas un pixel correspon a dos texels:

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y} = 2 \quad \frac{\partial v}{\partial x} = \frac{\partial u}{\partial y} = 0$$

Exemple 4 (anisotròpic)



Textura WxH texels

En direcció horitzontal → magnification
En direcció vertical → minification

Exemple 4 (anisotròpic)

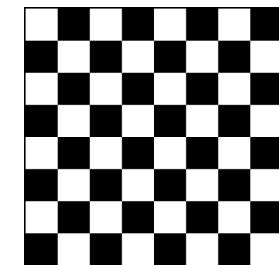
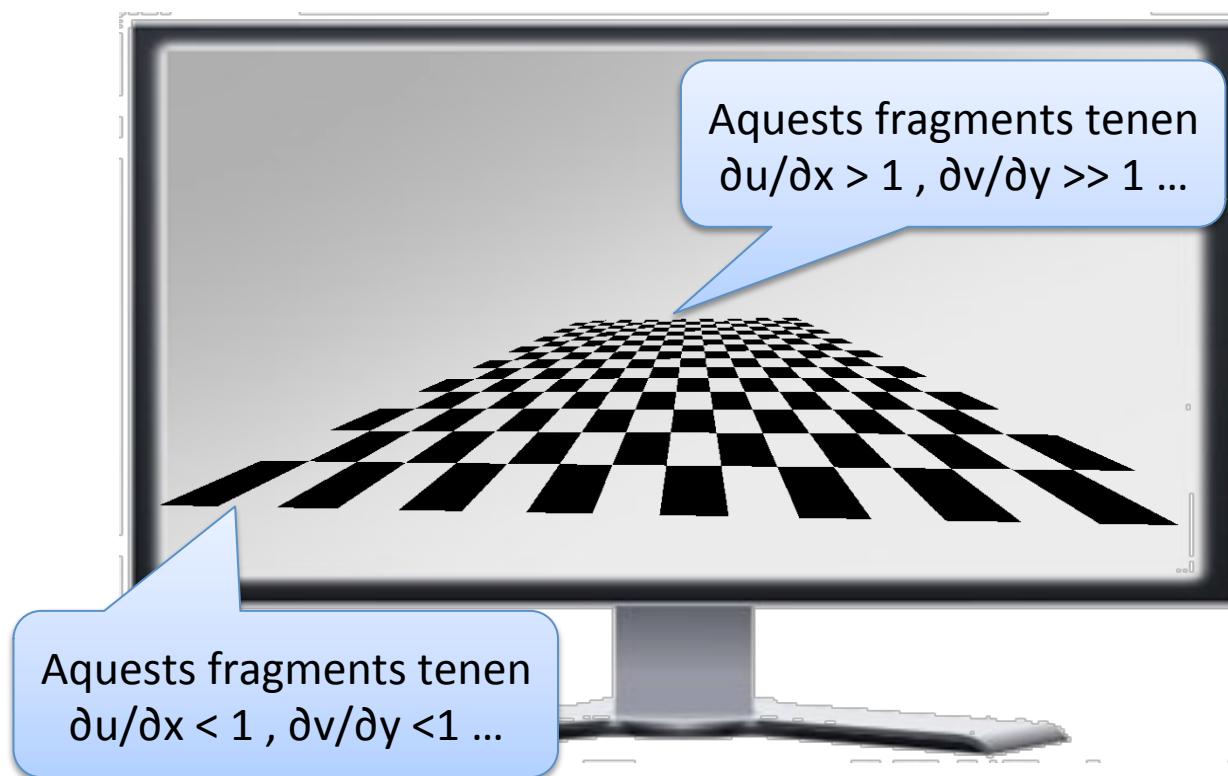


Textura WxH texels

En direcció horitzontal → magnification
En direcció vertical → minification

$$\begin{aligned}\frac{\partial u}{\partial x} &= \frac{1}{2} & \frac{\partial v}{\partial x} &= 0 \\ \frac{\partial u}{\partial y} &= 0 & \frac{\partial v}{\partial y} &= 2\end{aligned}$$

Exemple 5 (la dura realitat)

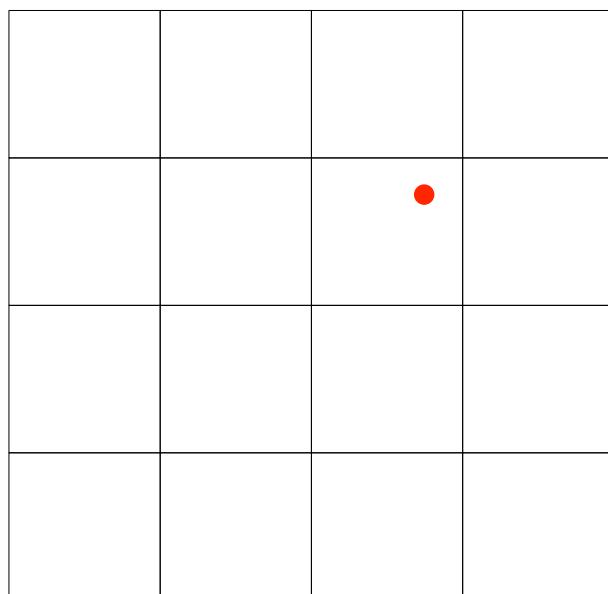


Textura

Texture filters

Determinen:

- Com es calcula el color al punt (s,t)
- Com s'avalua `texture2D(sampler, gl_TexCoord[0].st)`



Textura WxH texels

Magnification filters

- **GL_NEAREST**: nearest neighbor sampling

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

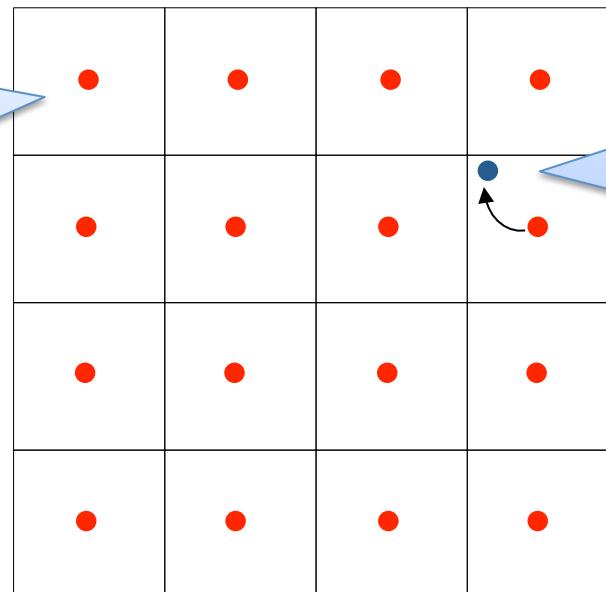
- **GL_LINEAR**: interpolació bilineal

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

Magnification filters

GL_NEAREST: nearest neighbor sampling

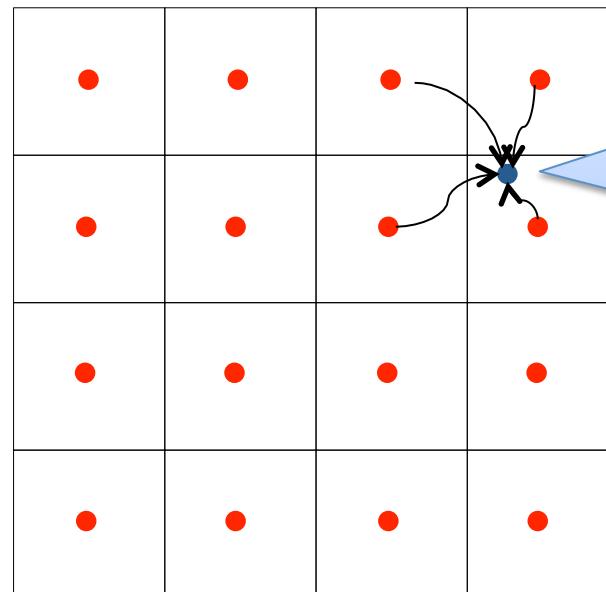
Assumirem que el color del texel és el color del seu centre



Amb nearest neighbor sampling, el color d'aquesta mostra és el color del veí més proper

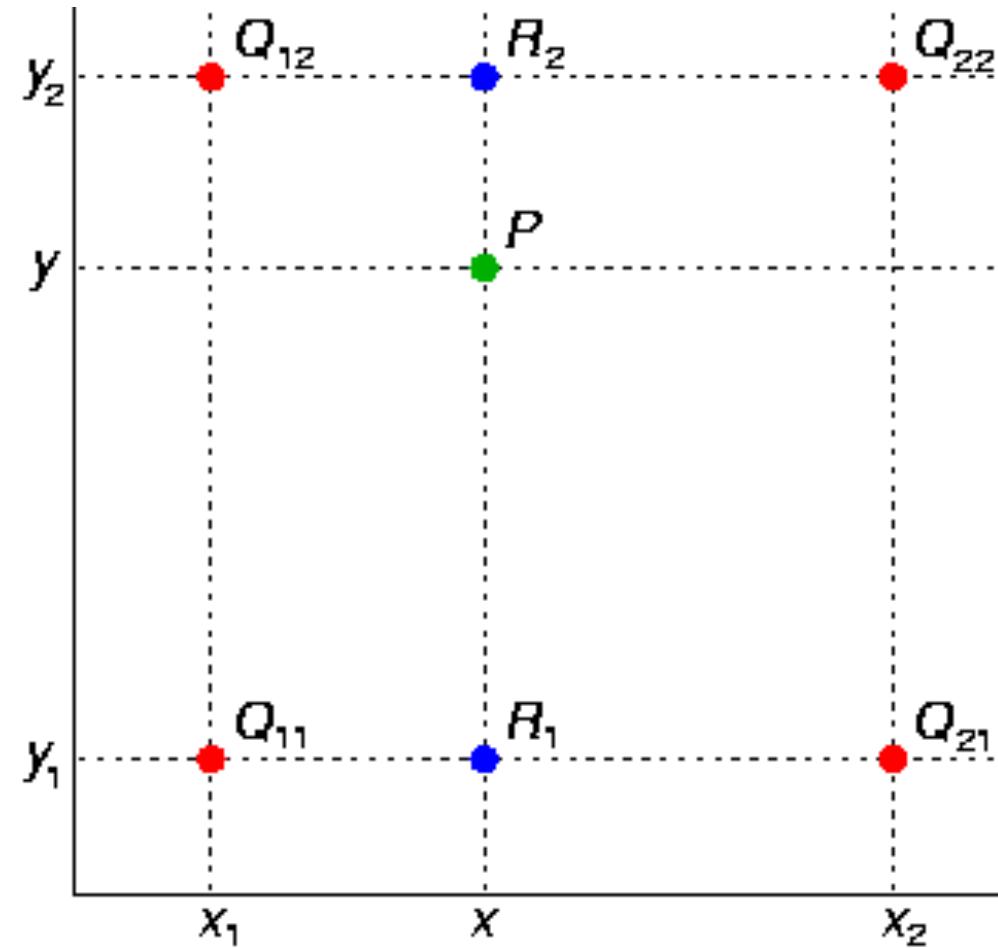
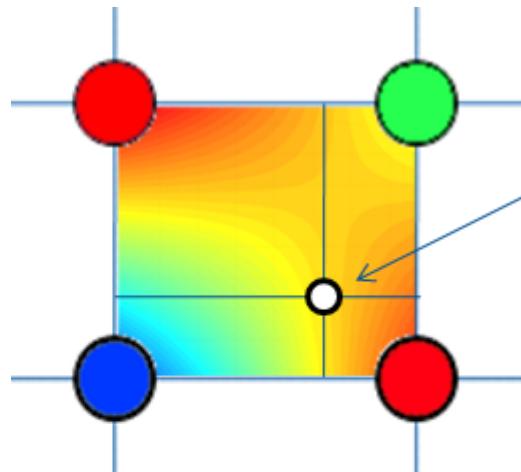
Magnification filters

GL_LINEAR: bilinear interpolation

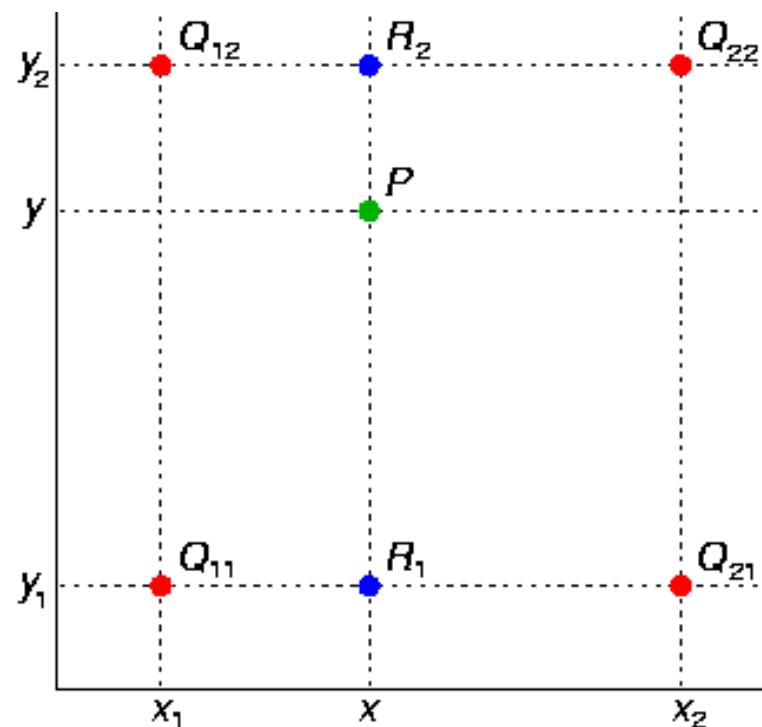


Amb interpolació bilineal,
el color d'aquesta mostra
és una mitjana ponderada
dels colors dels quatre
veïns més propers

Bilinear interpolation



Bilinear interpolation



$$f(x, y) \approx \frac{f(Q_{11})}{(x_2 - x_1)(y_2 - y_1)}(x_2 - x)(y_2 - y) + \frac{f(Q_{21})}{(x_2 - x_1)(y_2 - y_1)}(x - x_1)(y_2 - y) + \frac{f(Q_{12})}{(x_2 - x_1)(y_2 - y_1)}(x_2 - x)(y - y_1) + \frac{f(Q_{22})}{(x_2 - x_1)(y_2 - y_1)}(x - x_1)(y - y_1).$$

Magnification filters



[Demo shadermaker]

Minification filters

- **GL_NEAREST**: nearest neighbor sampling

```
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
```

- **GL_LINEAR**: interpolació bilineal

```
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
```

Minification filters

- **Magnification** → la preimatge d'un pixel en espai textura és petita → n'hi ha prou filtrant amb 2x2 texels.
- **Minification** → la preimatge d'un pixel és arbitràriament gran → no n'hi ha prou amb 2x2 texels!

Si aquest quad es projecta en 2x2 pixels, cada fragment correspon a una quarta part de la imatge!



Minification x16

Minification x8

Minification x4

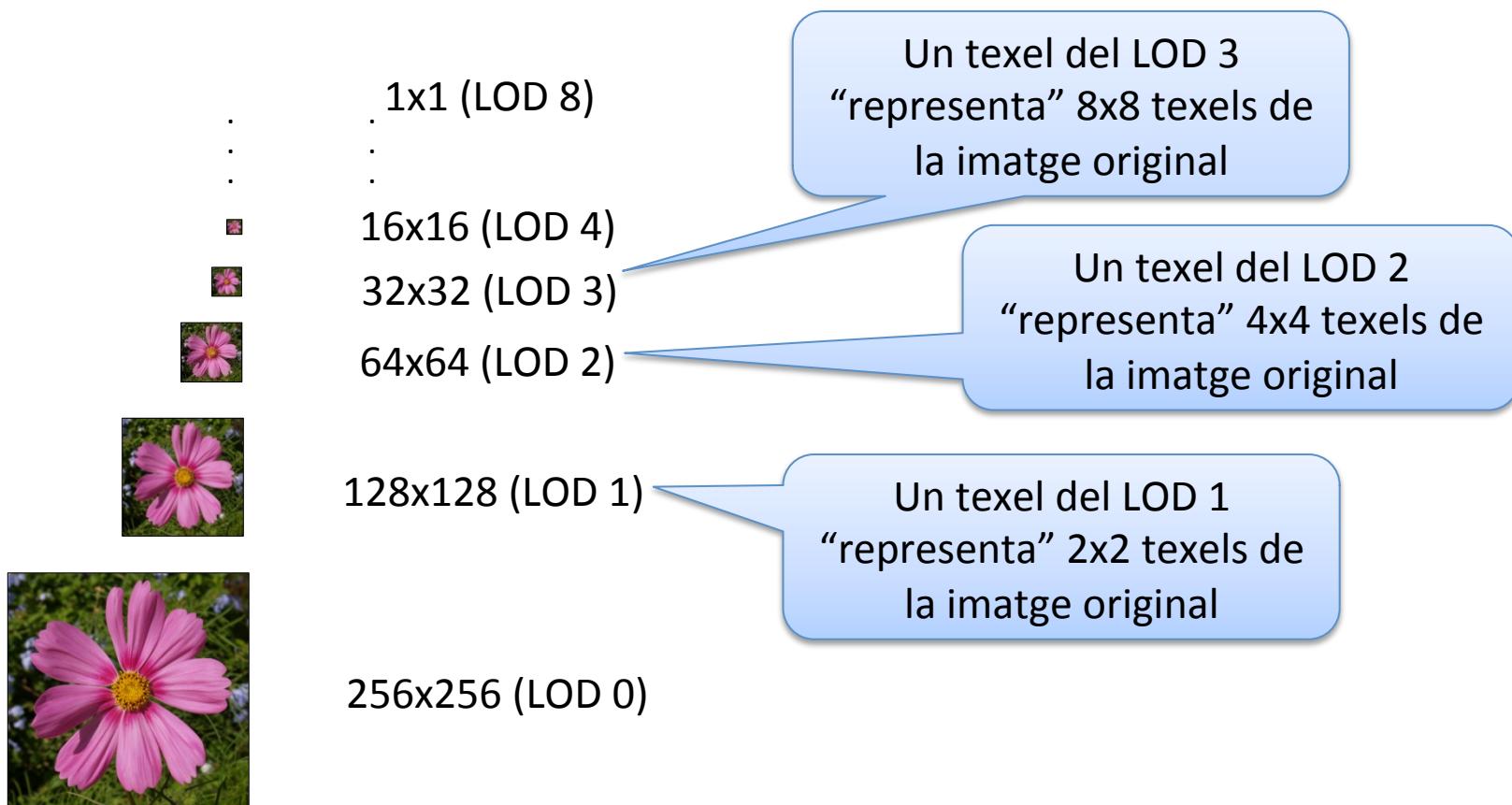
Minification x2



Textura

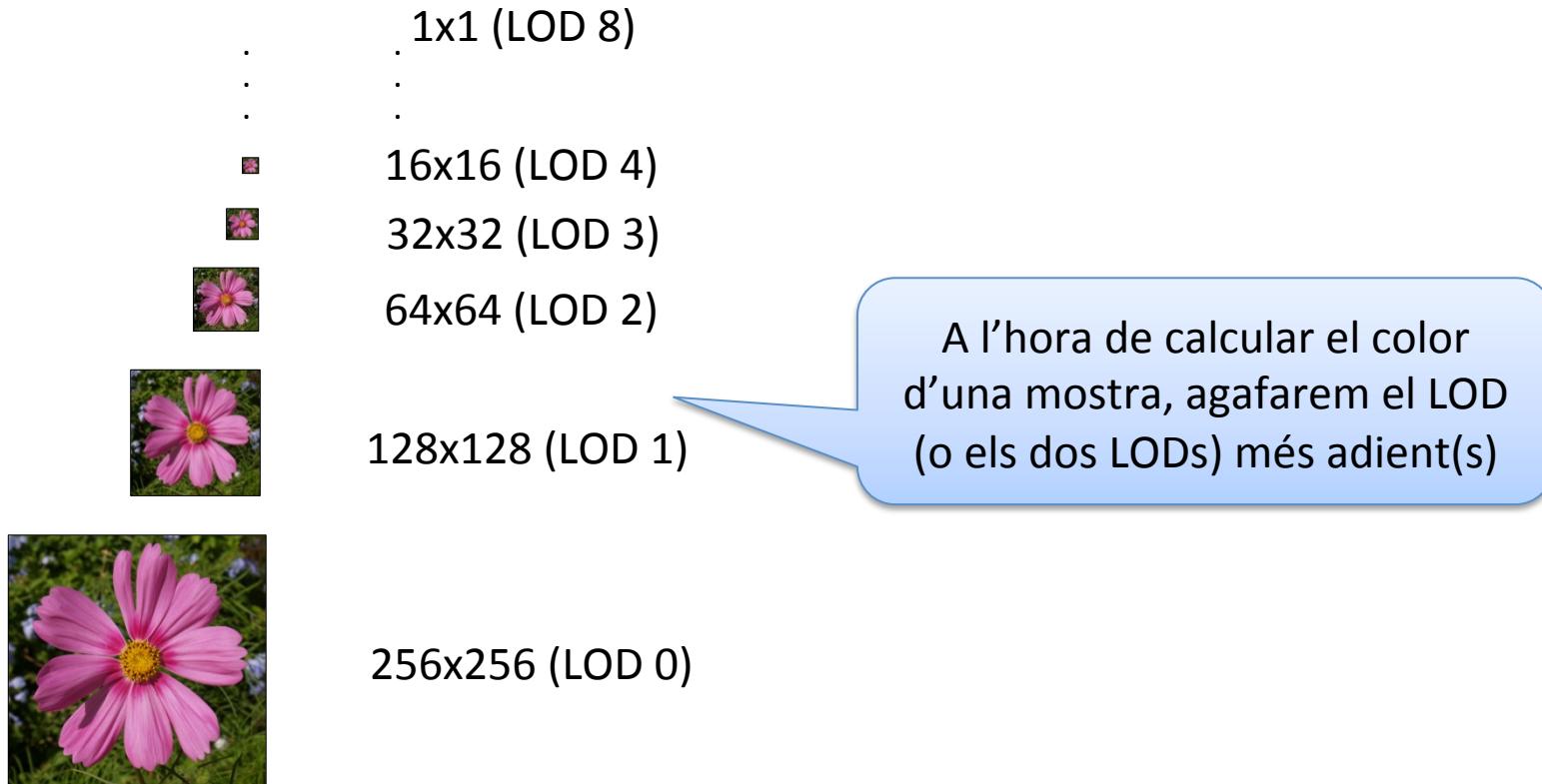
Mipmapping

Idea bàsica: cada textura està representada amb diferents resolucions (level-of-details, LODs)



Mipmapping

Idea bàsica: cada textura està representada amb diferents resolucions (level-of-details, LODs)



Mipmapping

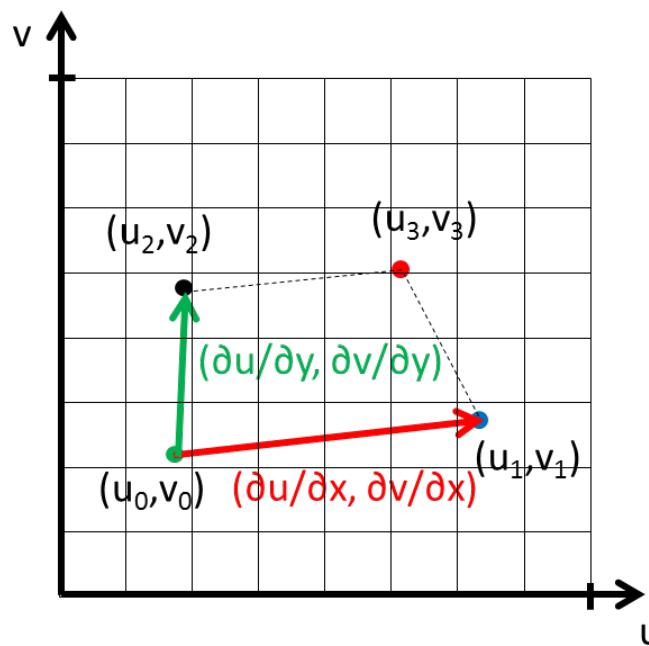
En alguns casos el LOD més adient λ és fàcil de calcular

	Minification	$\frac{\partial u}{\partial x}, \frac{\partial v}{\partial y}$	LOD
	1x	1	0
	2x	2	1
	4x	4	2
	8x	8	3
■	$2^\lambda x$	2^λ	λ

En aquest cas $\frac{\partial u}{\partial x} = 2^\lambda$
Per tant: $\lambda = \log_2(\frac{\partial u}{\partial x})$

Mipmapping

- En general, però, $\partial u / \partial x \neq \partial v / \partial x \neq \partial u / \partial y \neq \partial v / \partial y$
- En aquests casos, es calcula un valor $\rho = f(\partial u / \partial x, \partial v / \partial x, \partial u / \partial y, \partial v / \partial y)$ i després $\lambda = \log_2(\rho)$



Minification filters (2)

Without mipmapping

- GL_NEAREST // Nearest neighbor sampling on LOD 0
- GL_LINEAR // Bilinear interpolation on LOD 0

With mipmapping

- GL_NEAREST_MIPMAP_NEAREST // Nearest neighbor sampling on LOD $\text{int}(\lambda)$
- GL_LINEAR_MIPMAP_NEAREST // Bilinear sampling on LOD $\text{int}(\lambda)$
- GL_NEAREST_MIPMAP_LINEAR // c_0 = nearest neighbor on LOD $\text{int}(\lambda)$
// c_1 = nearest neighbor on LOD $\text{int}(\lambda+1)$
// $\text{mix}(c_0, c_1, \text{fract}(\lambda))$
- GL_LINEAR_MIPMAP_LINEAR // c_0 = bilinear sampling on LOD $\text{int}(\lambda)$
// c_1 = bilinear sampling on LOD $\text{int}(\lambda+1)$
// $\text{mix}(c_0, c_1, \text{fract}(\lambda))$



GL_<samplingWithinTheImage>_MIPMAP_<oneOrTwoMipmaps>

Minification filters (2)

[Demo mipmapping]

Texture combiners

COMBINACIÓ

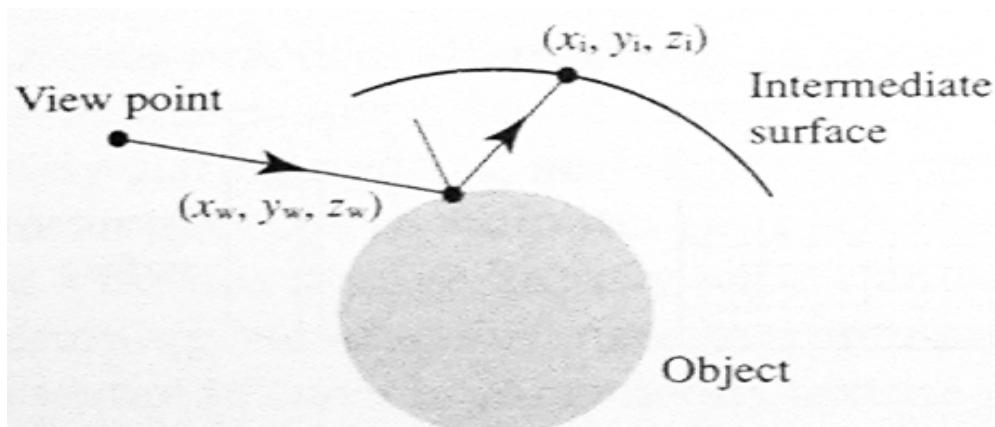
Light map/Diffuse color map



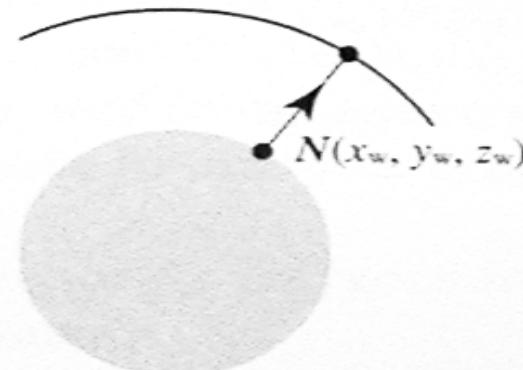
[Texture combiner demo]

GENERACIÓ DE COORDENADES DE TEXTURA

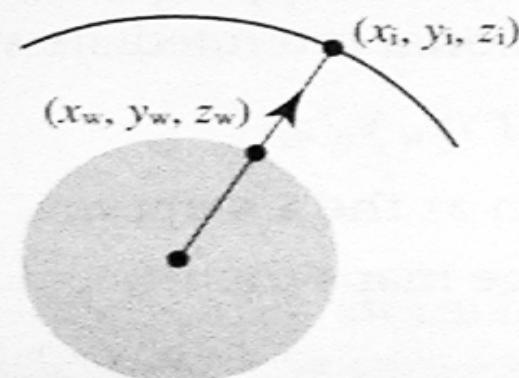
O mappings



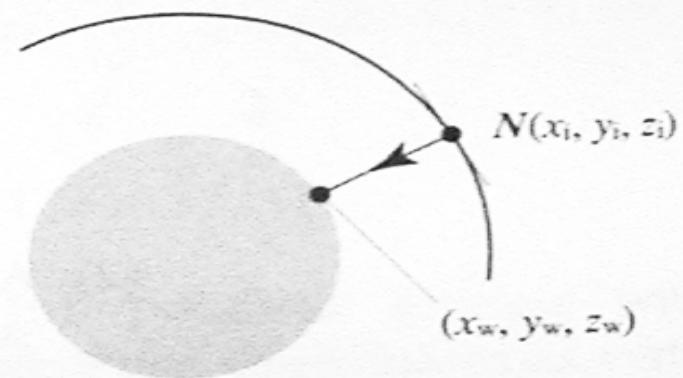
(1) Reflected ray



(2) Object normal

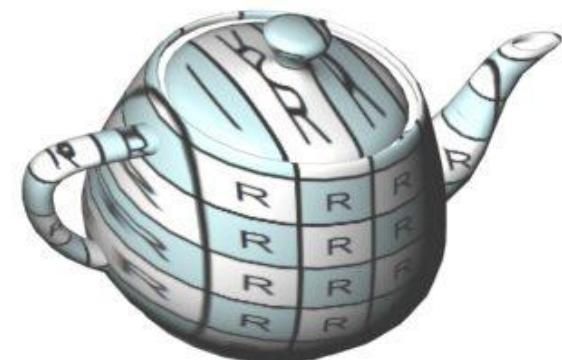
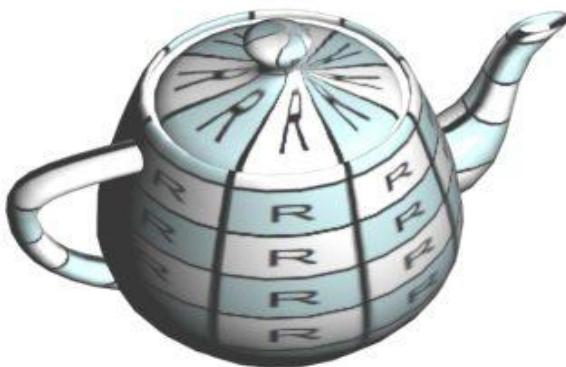
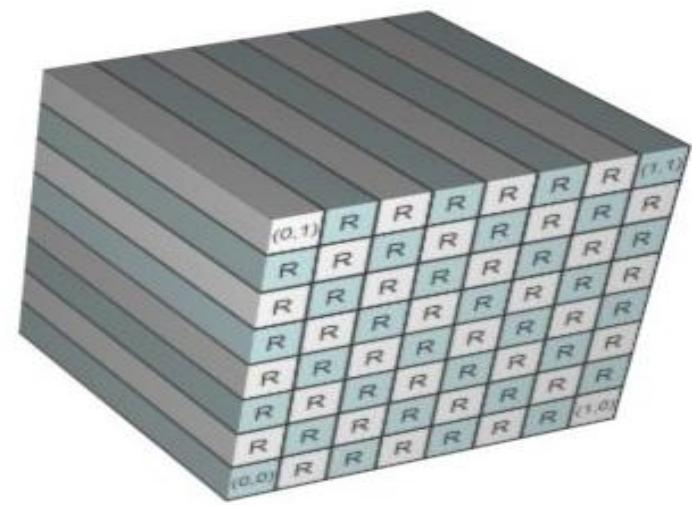
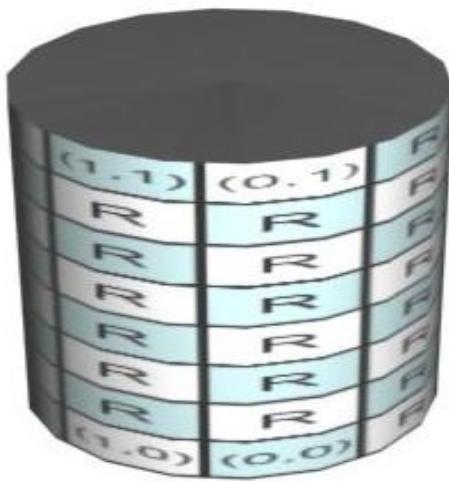


(3) Object centroid

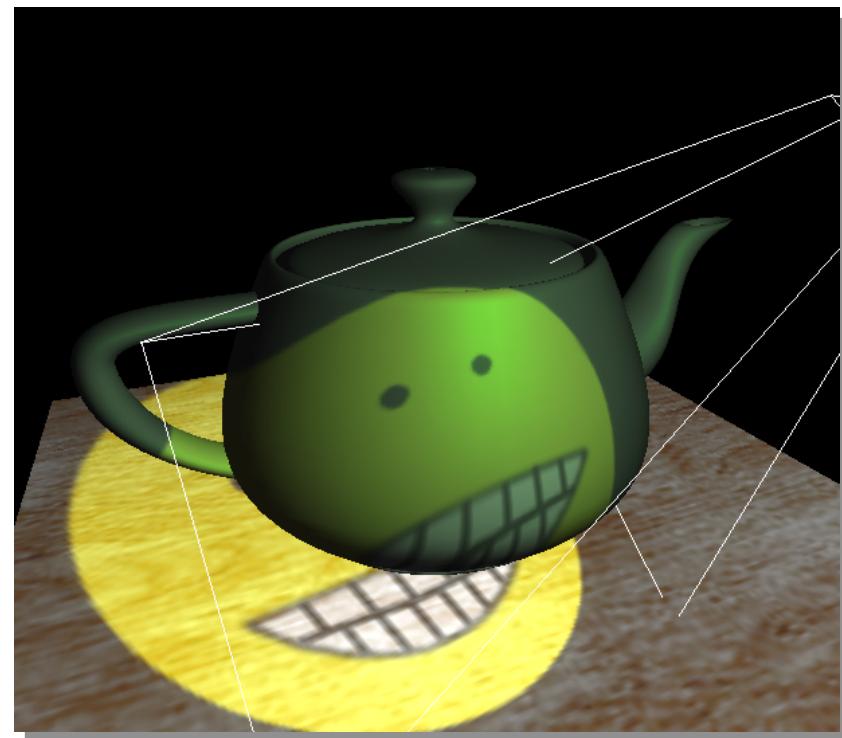
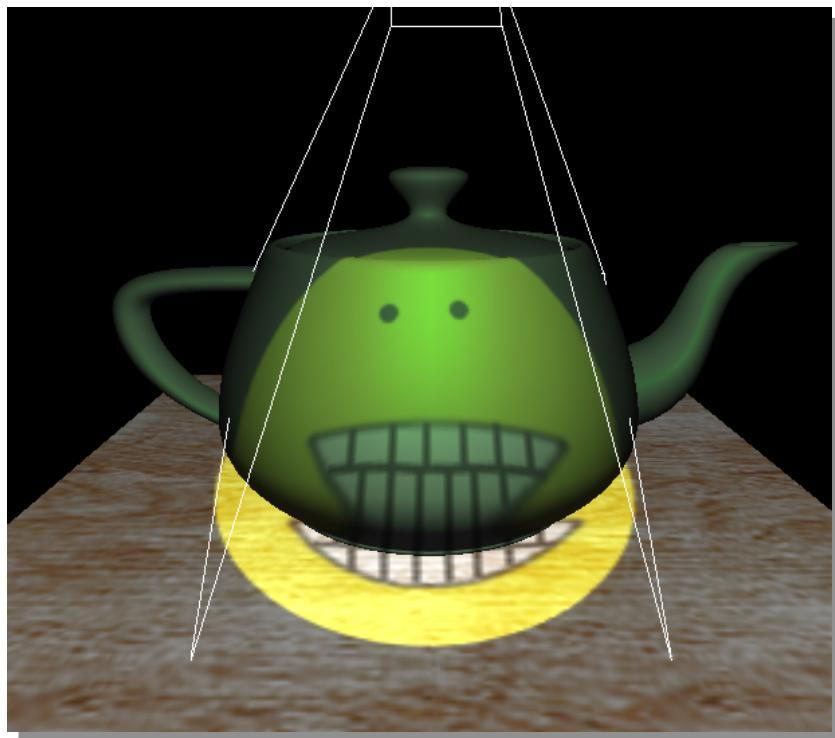


(4) Intermediate surface normal

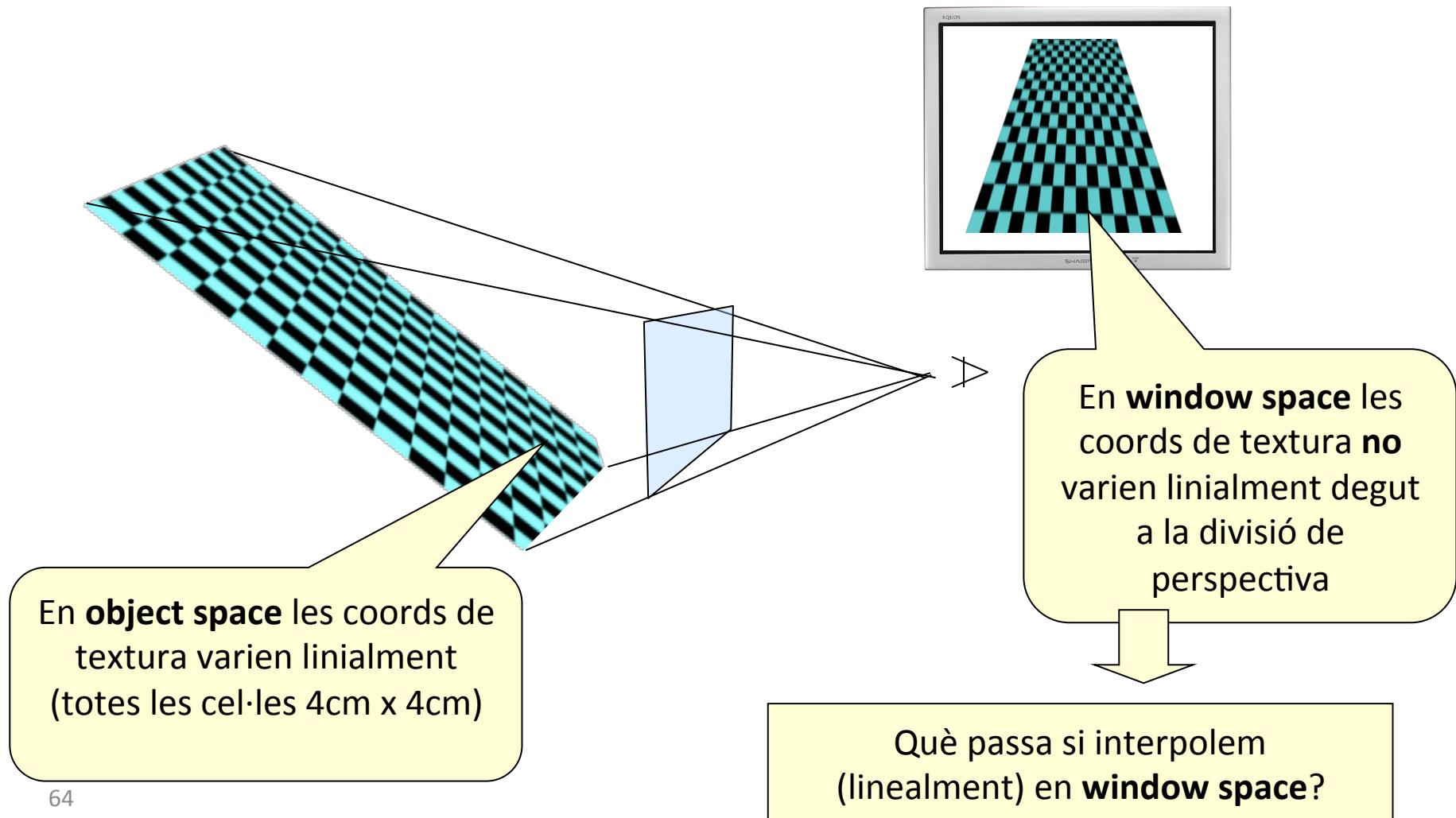
Projeccions esfèrica, cilíndrica i plana



Projective texture mapping



Perspective-correct interpolation



Perspective-correct interpolation

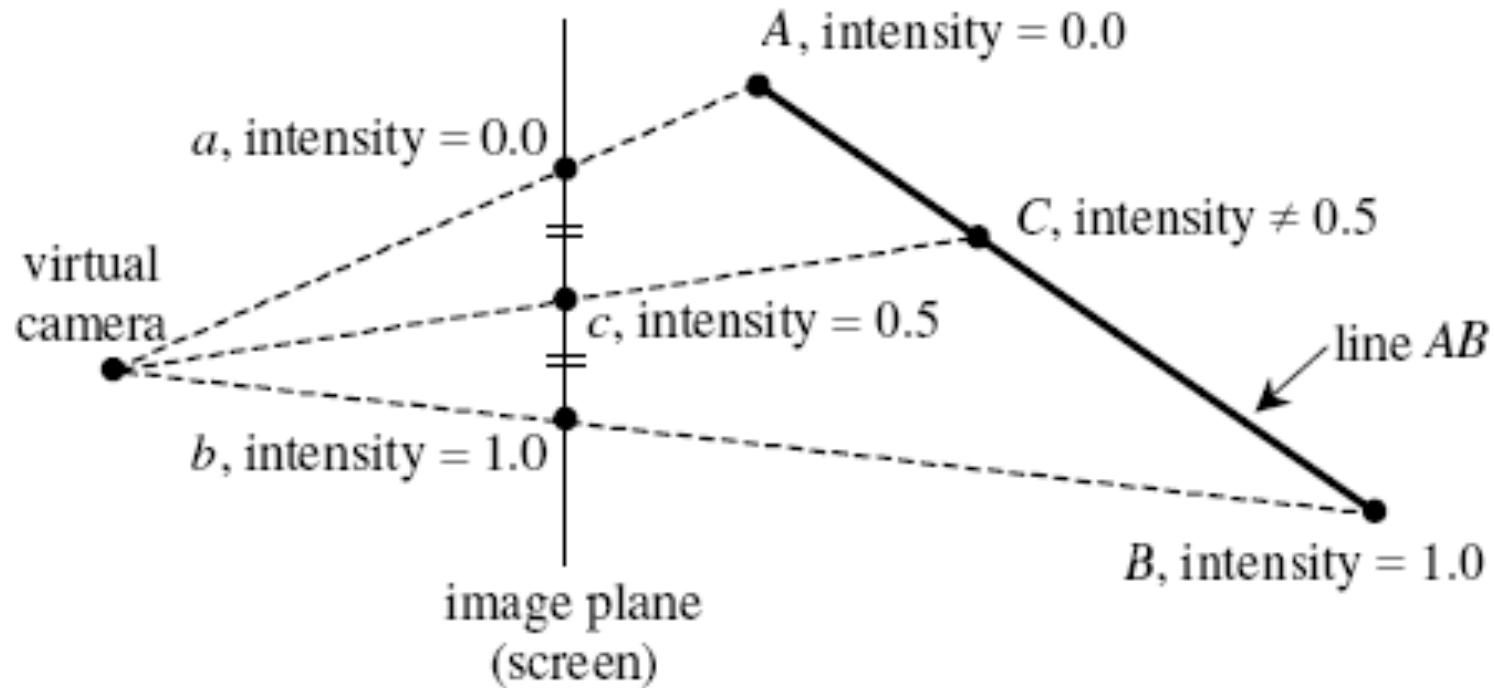


Figure 1: Straightforward linear interpolation of attribute values in the screen space (or in the image plane) does not always produce perspective-correct results.

Perspective-correct interpolation

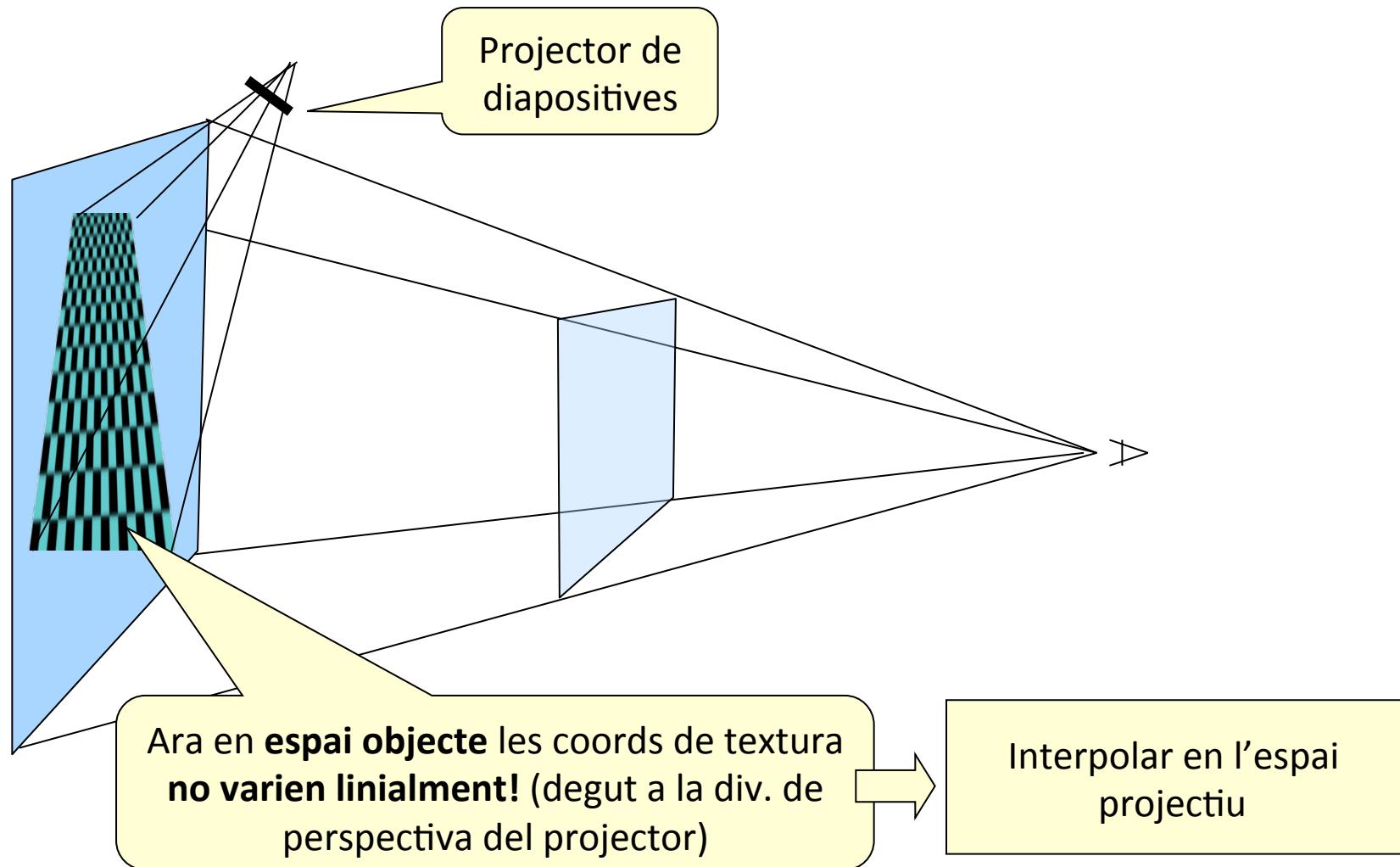
Solucions:

- a) Interpolem (s, t) en **object space** (també valdria en **world space** o **eye space**, perquè són abans de la div. de perspectiva)
- b) Interpolem (sw_d, tw_d, w_d) en **window space**, obtenint un texel (s, t, q) ; accedirem a la textura amb $(s/q, t/q)$

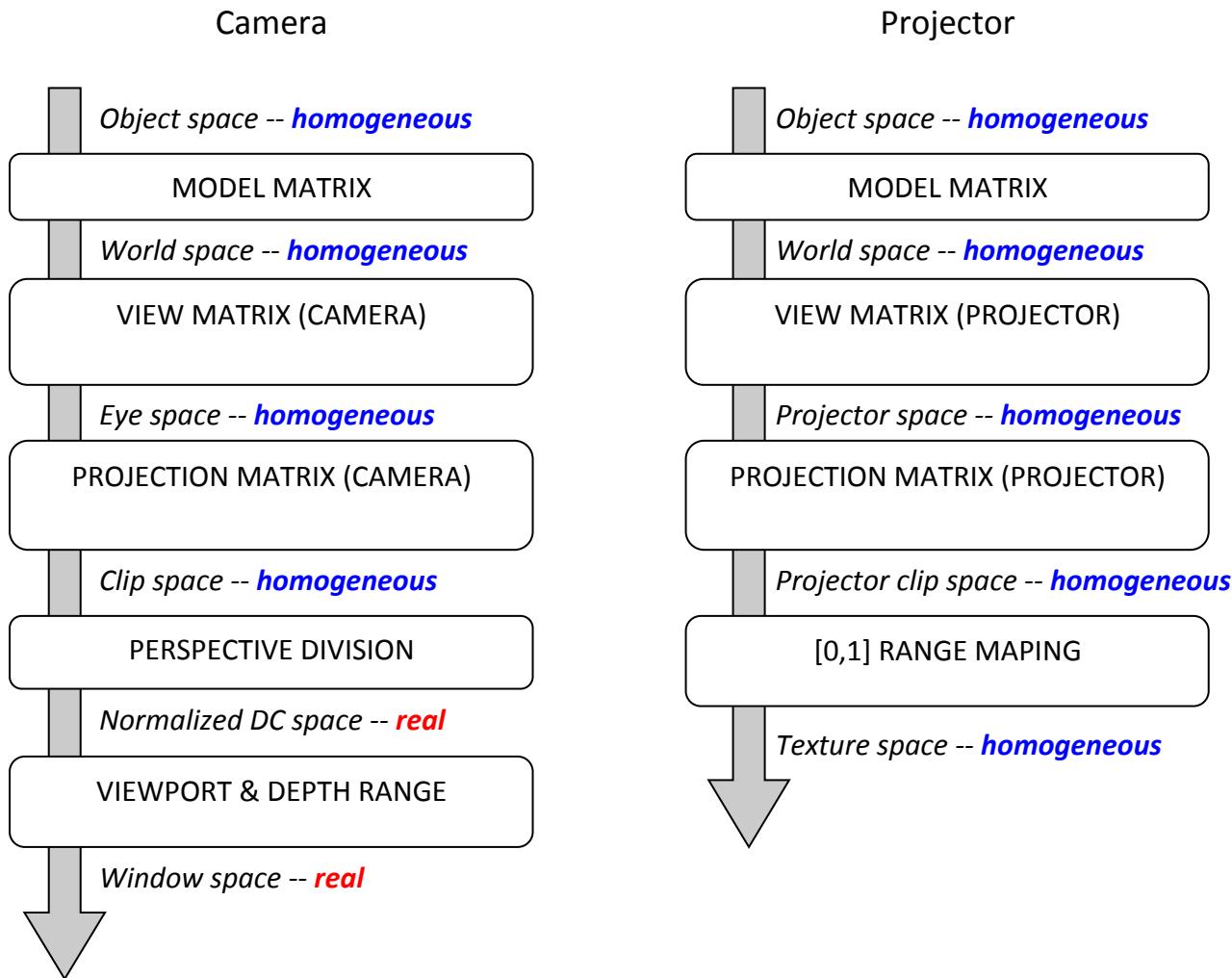
Recordeu que $w_d = 1/w_c = -1/z_e$

[Demostració de (b): consulteu per exemple l'article de Kok-Lim Low]

Projective-space interpolation



Projective texture mapping



Projective texture mapping (1/2)

```
// Pas 1: Modificar la TEXTURE MATRIX
glMatrixMode(GL_TEXTURE);
glLoadIdentity();
glTranslated(0.5, 0.5, 0.5); // T (translació)
glScaled(0.5, 0.5, 0.5);    // S (escalat)
gluPerspective(...);        //  $P_p$  (projection matrix del projector)
gluLookAt(...);             //  $V_p$  (view matrix del projector, no de càmera!)
glMultMatrixf(Model);      // M (transformació de modelat)
glMatrixMode(GL_MODELVIEW);
.../
// Pas 2: Dibuixar amb coords de textura = coords del vèrtex
...
glTexCoord3f(x, y, z);
glVertex3f(x, y, z);
...
```

Projective texture mapping (2/2)

// Pas 2: Opció (a) – Dibuixar amb coords de textura = coords del vèrtex

```
glTexCoord3f(x,y,z);  
glVertex3f(x,y,z);  
...
```

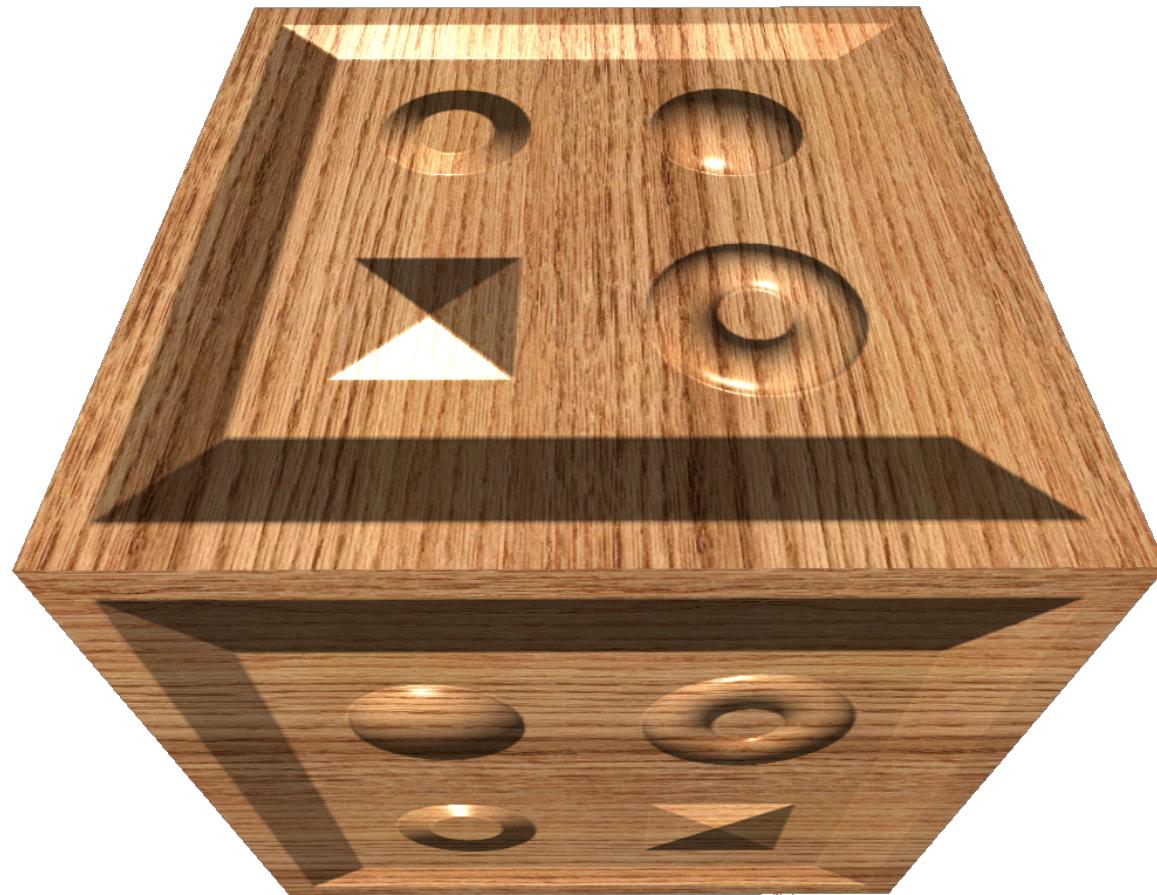
// Pas 2: Opció (b) – Utilitzar glTexGen per aconseguir el mateix efecte

```
glTexGenf(GL_S, GL_TEXTURE_GEN_MODE, GL_OBJECT_LINEAR);  
glTexGenf(GL_T, GL_TEXTURE_GEN_MODE, GL_OBJECT_LINEAR);  
glTexGenf(GL_R, GL_TEXTURE_GEN_MODE, GL_OBJECT_LINEAR);  
glTexGenf(GL_Q, GL_TEXTURE_GEN_MODE, GL_OBJECT_LINEAR);  
float s[4] ={1.0f, 0.0f, 0.0f, 0.0f};  
float t[4] ={0.0f, 1.0f, 0.0f, 0.0f};  
float r[4] ={0.0f, 0.0f, 1.0f, 0.0f};  
float q[4] ={0.0f, 0.0f, 0.0f, 1.0f};  
glTexGenf(GL_S, GL_OBJECT_PLANE, s);  
glTexGenf(GL_T, GL_OBJECT_PLANE, t);  
glTexGenf(GL_R, GL_OBJECT_PLANE, r);  
glTexGenf(GL_Q, GL_OBJECT_PLANE, q);  
dibuixarEscena()
```

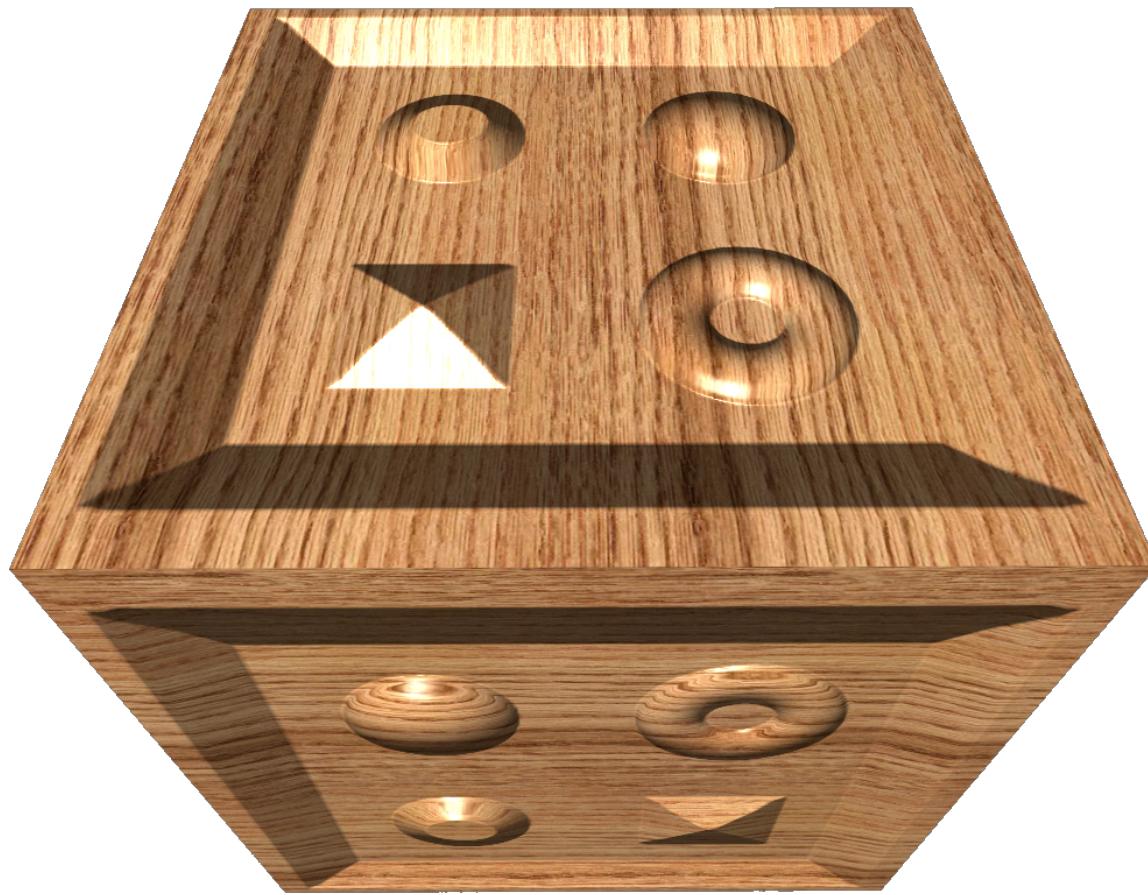
Color, bump, parallax, relief i displacement mapping

APLICACIONS

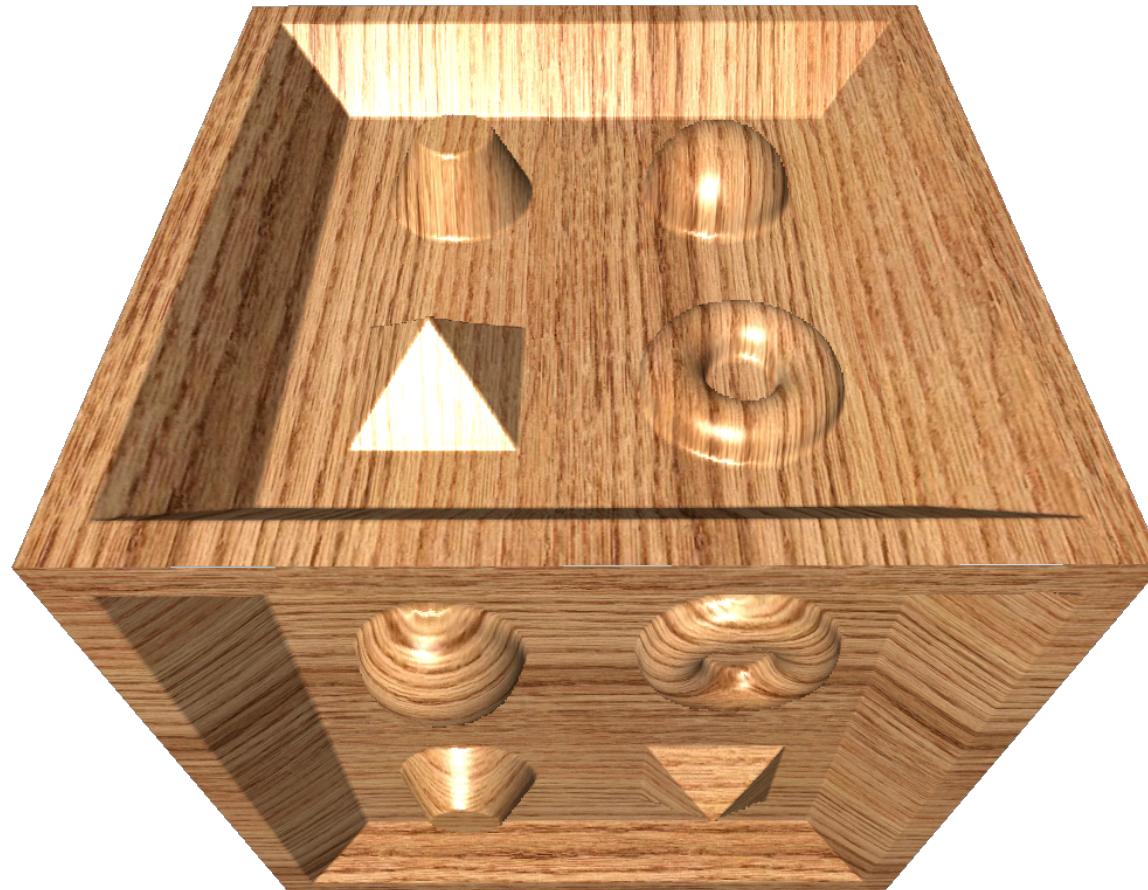
Bump mapping/Normal mapping



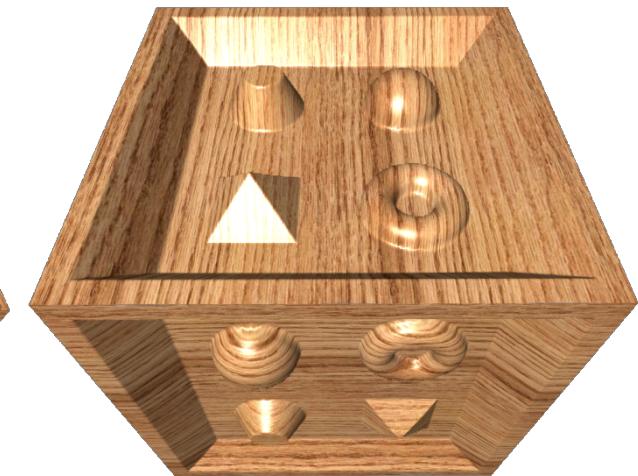
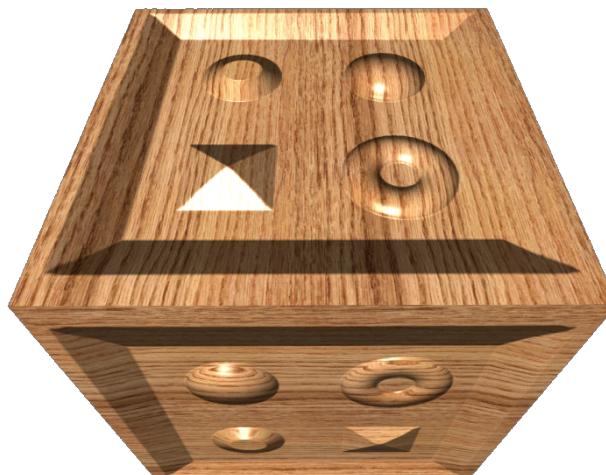
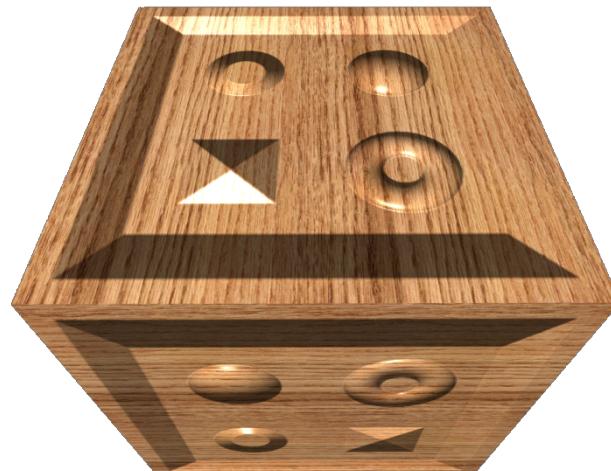
Parallax mapping



Relief mapping



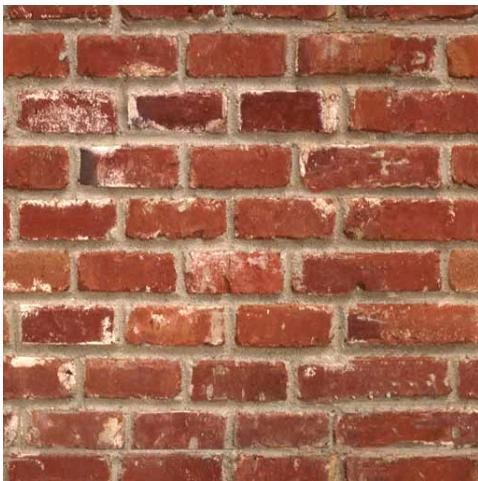
Comparació



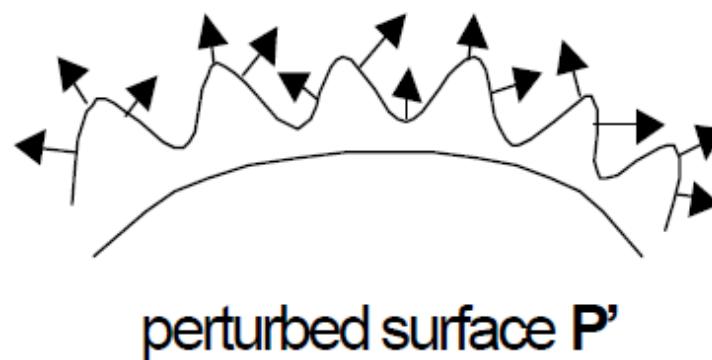
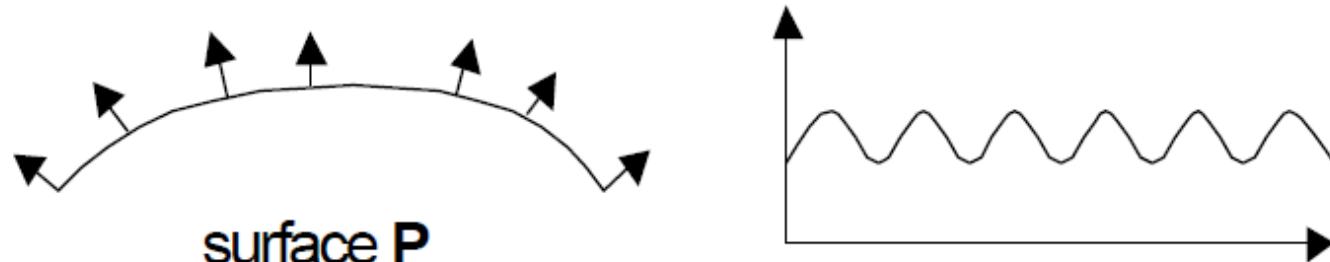
Mark Kilgard. A Practical and Robust Bump-mapping Technique for
Today's GPUs. GDC 2000

BUMP MAPPING

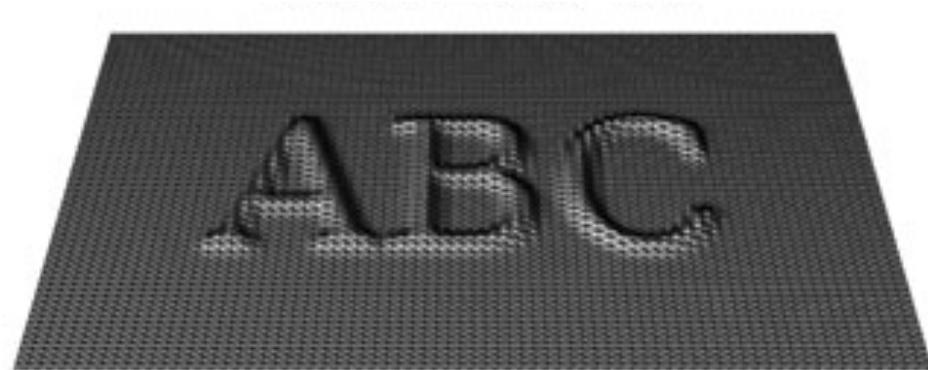
Bump mapping



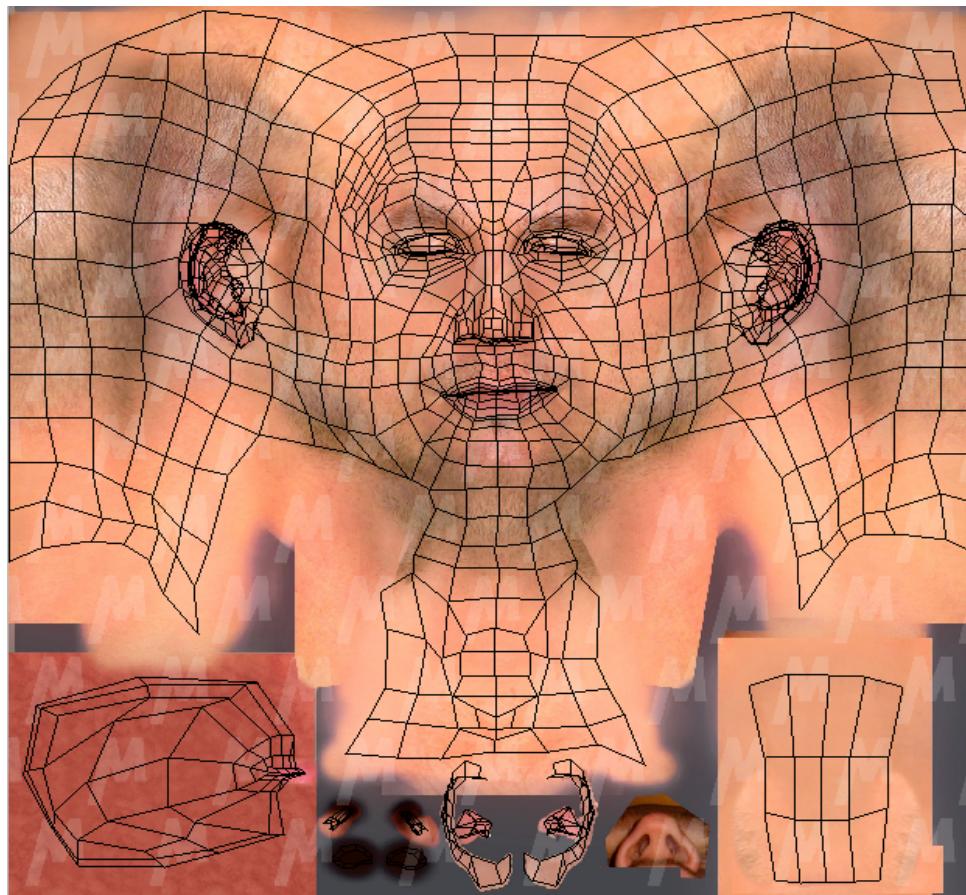
Elements bàsics



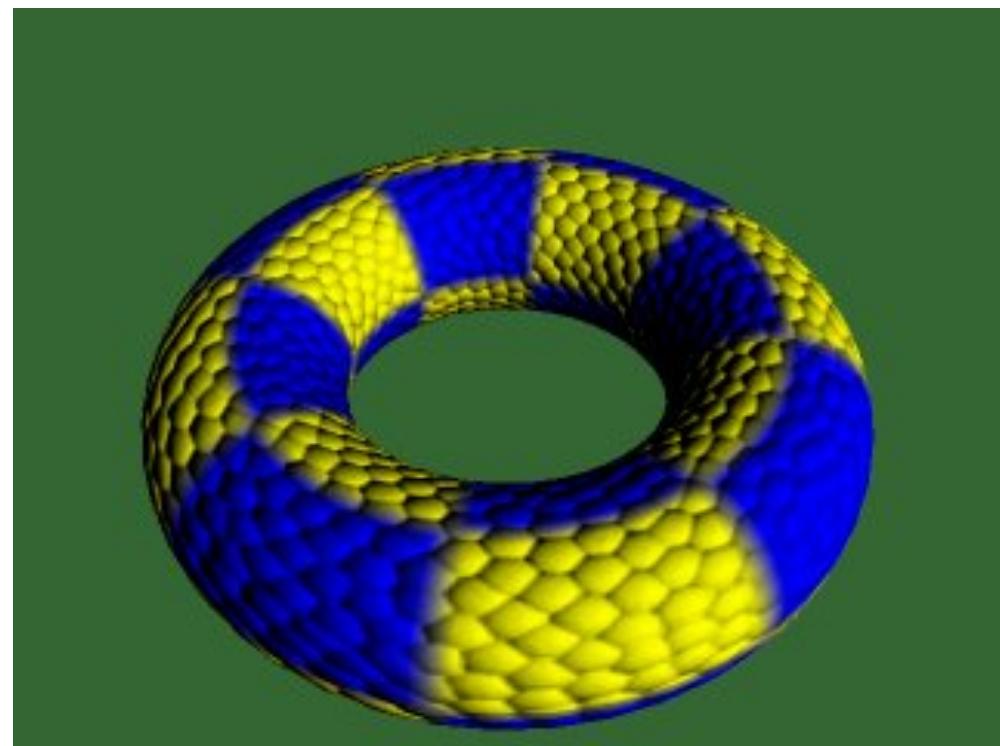
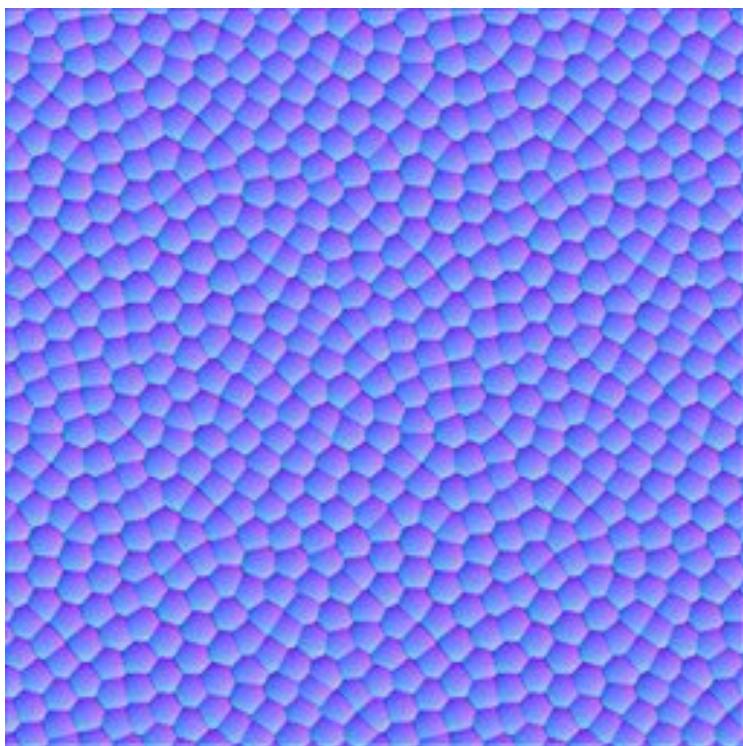
Height field



Malla amb coordenades de textura



Elements bàsics



Eqüacions

$$\mathbf{N}(u, v) = \frac{\partial \mathbf{P}(u, v)}{\partial u} \times \frac{\partial \mathbf{P}(u, v)}{\partial v}$$

$$\mathbf{N}'(u, v) = \frac{\partial \mathbf{P}'(u, v)}{\partial u} \times \frac{\partial \mathbf{P}'(u, v)}{\partial v}$$

$$\mathbf{P}'(u, v) = \mathbf{P}(u, v) + F(u, v) \frac{\mathbf{N}(u, v)}{|\mathbf{N}(u, v)|}$$

Eqüacions

$$\mathbf{N}'(u,v) = \frac{\partial \mathbf{P}'(u,v)}{\partial u} \times \frac{\partial \mathbf{P}'(u,v)}{\partial v} \quad \mathbf{P}'(u,v) = \mathbf{P}(u,v) + F(u,v) \frac{\mathbf{N}(u,v)}{|\mathbf{N}(u,v)|}$$

$$\mathbf{N}' = \left(\frac{\partial \mathbf{P}}{\partial u} + \frac{\partial F}{\partial u} \left(\frac{\mathbf{N}}{|\mathbf{N}|} \right) \right) \times \left(\frac{\partial \mathbf{P}}{\partial v} + \frac{\partial F}{\partial v} \left(\frac{\mathbf{N}}{|\mathbf{N}|} \right) \right) \quad \frac{\partial \mathbf{P}'}{\partial u} = \frac{\partial \mathbf{P}}{\partial u} + \frac{\partial F}{\partial u} \left(\frac{\mathbf{N}}{|\mathbf{N}|} \right) + F \left(\frac{\partial \frac{\mathbf{N}}{|\mathbf{N}|}}{\partial u} \right)$$

$$\mathbf{N}' = \frac{\partial \mathbf{P}}{\partial u} \times \frac{\partial \mathbf{P}}{\partial v} + \frac{\partial F}{\partial u} \left(\mathbf{N} \times \frac{\partial \mathbf{P}}{\partial v} \right) + \frac{\partial F}{\partial v} \left(\frac{\partial \mathbf{P}}{\partial u} \times \mathbf{N} \right) + \frac{\partial F}{\partial u} \frac{\partial F}{\partial v} (\mathbf{N} \times \mathbf{N})$$

$$\mathbf{N}' = \mathbf{N} + \frac{\frac{\partial F}{\partial u} \left(\mathbf{N} \times \frac{\partial \mathbf{P}}{\partial v} \right) - \frac{\partial F}{\partial v} \left(\mathbf{N} \times \frac{\partial \mathbf{P}}{\partial u} \right)}{|\mathbf{N}|}$$

Eqüacions - resum

$$\mathbf{N}'(u, v) = \frac{\partial \mathbf{P}'(u, v)}{\partial u} \times \frac{\partial \mathbf{P}'(u, v)}{\partial v}$$

$$\mathbf{P}'(u, v) = \mathbf{P}(u, v) + F(u, v) \frac{\mathbf{N}(u, v)}{|\mathbf{N}(u, v)|}$$

$$\mathbf{N}' = \mathbf{N} + \frac{\frac{\partial F}{\partial u} \left(\mathbf{N} \times \frac{\partial \mathbf{P}}{\partial v} \right) - \frac{\partial F}{\partial v} \left(\mathbf{N} \times \frac{\partial \mathbf{P}}{\partial u} \right)}{|\mathbf{N}|}$$