Textures

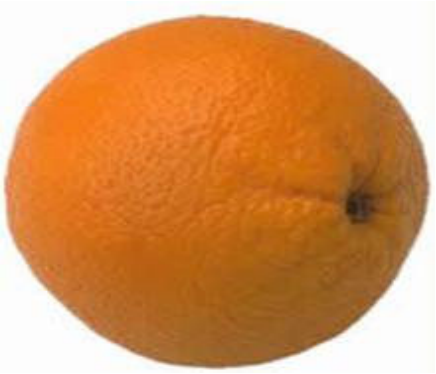
Carlos Andujar

Feb 2021

INTRODUCCIÓ

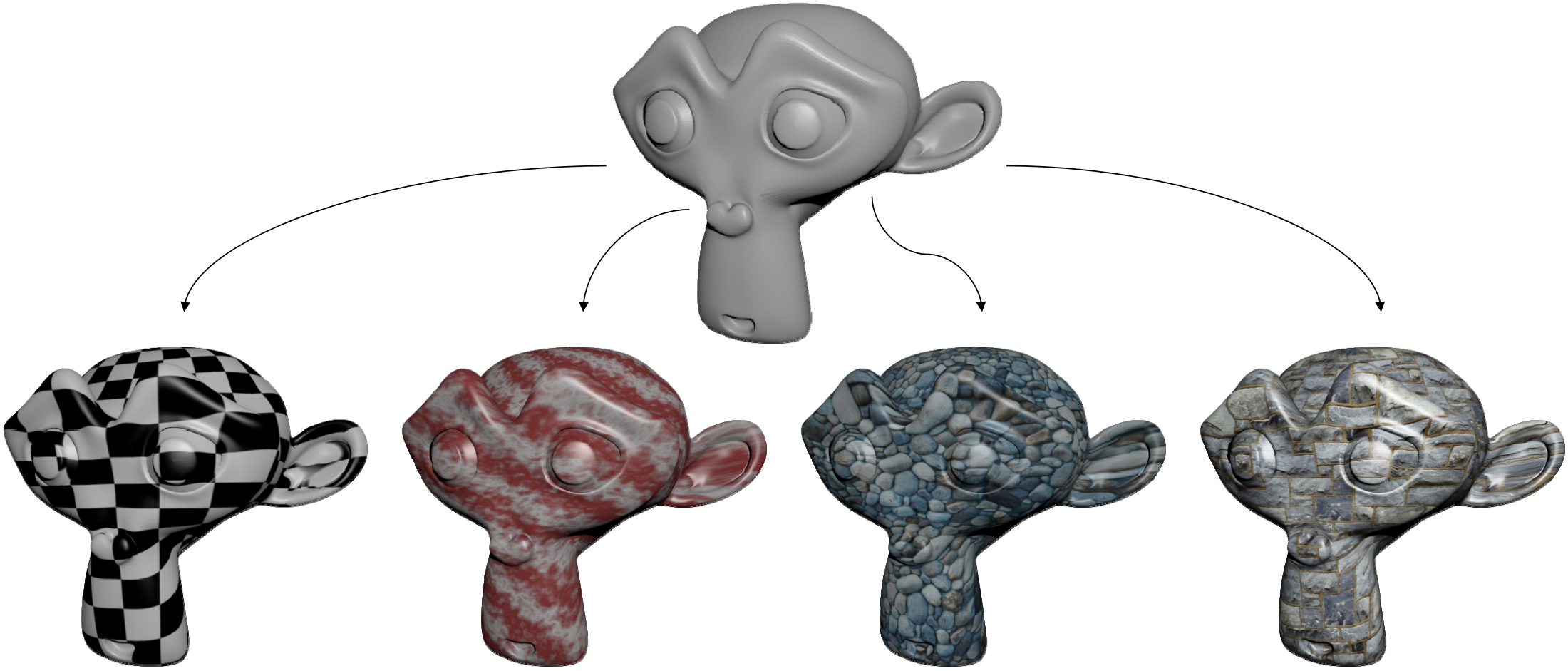
Representació de detalls superficials

Variacions de la *geometria:*



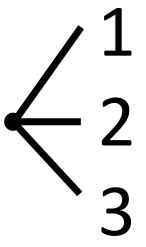
Variacions de les *propietats òptiques:*

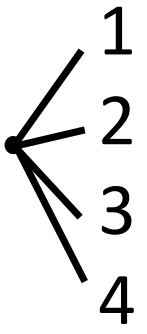
Representació de detalls superficials



DEFINICIONS

Textures

Una textura és una taula de 

dimensions, on cada cel·la enmagatzema una certa propietat amb 

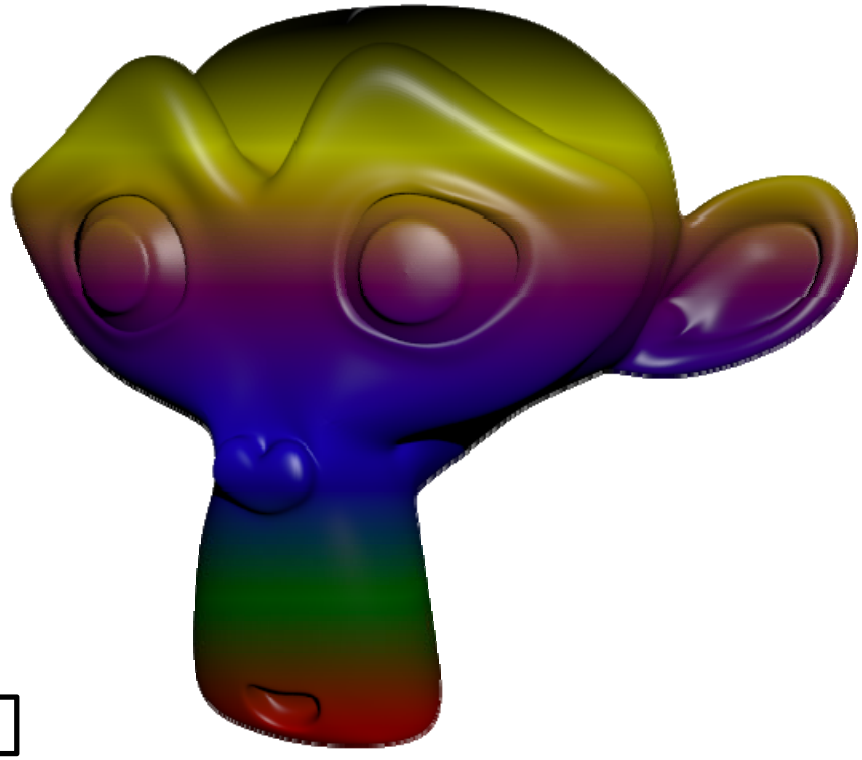
canals.

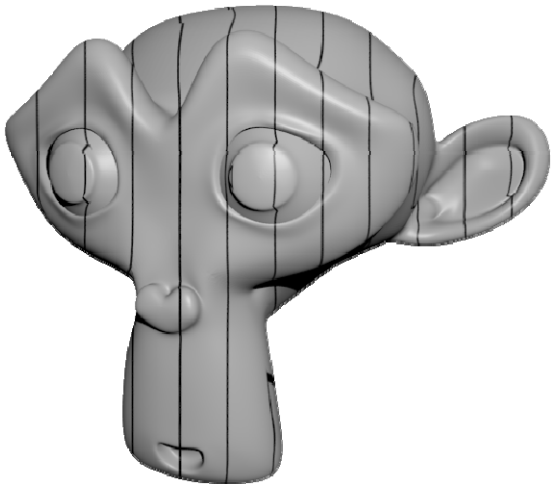
Habitualment les textures s'utilitzen al FS, tot i que també es poden usar en altres shaders.

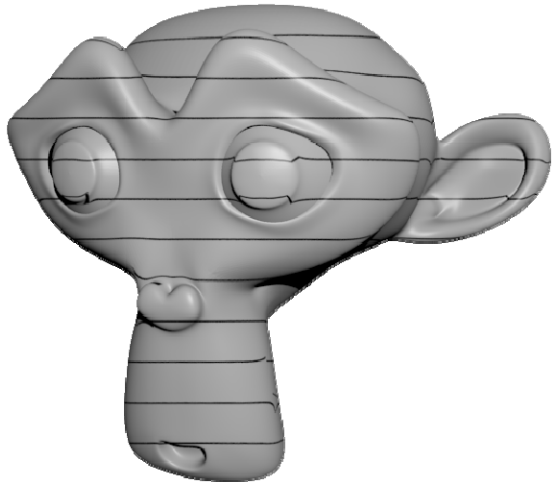
Textures

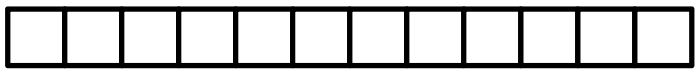
Una textura és una taula de 1 / 2 / 3 dimensions, on cada cel·la enmagatzema una certa propietat amb 1 / 2 / 3 / 4 canals.

Textures 1D



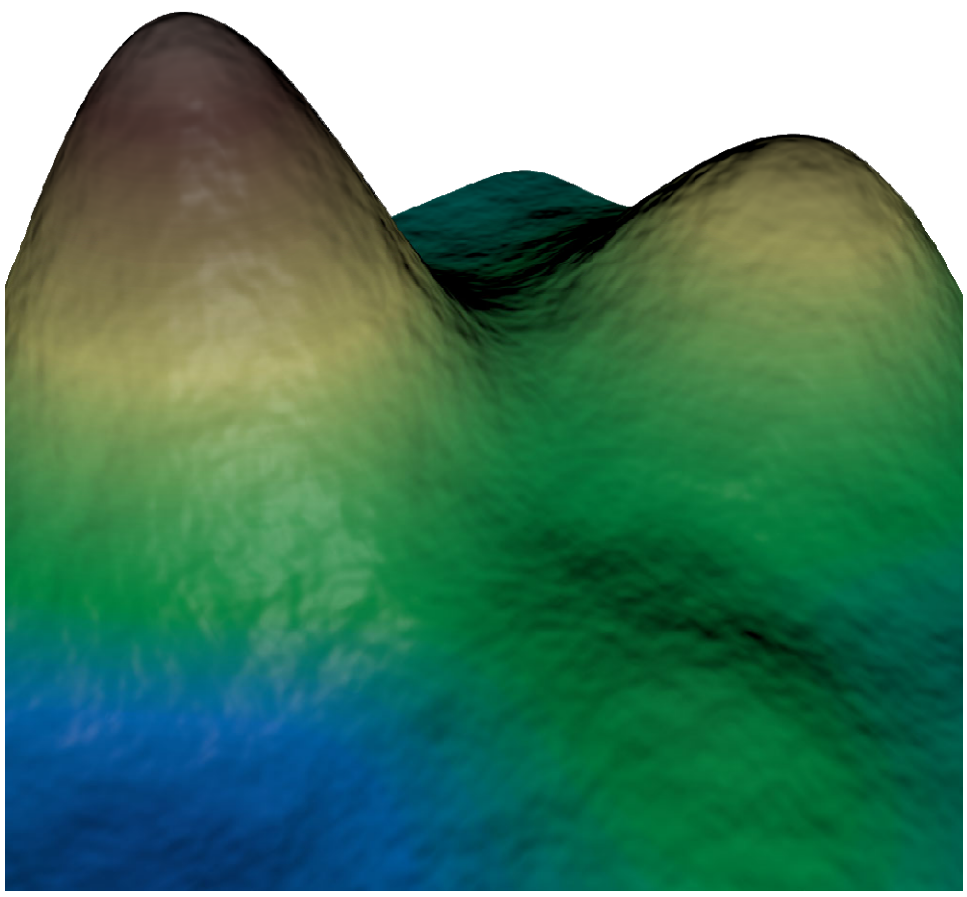


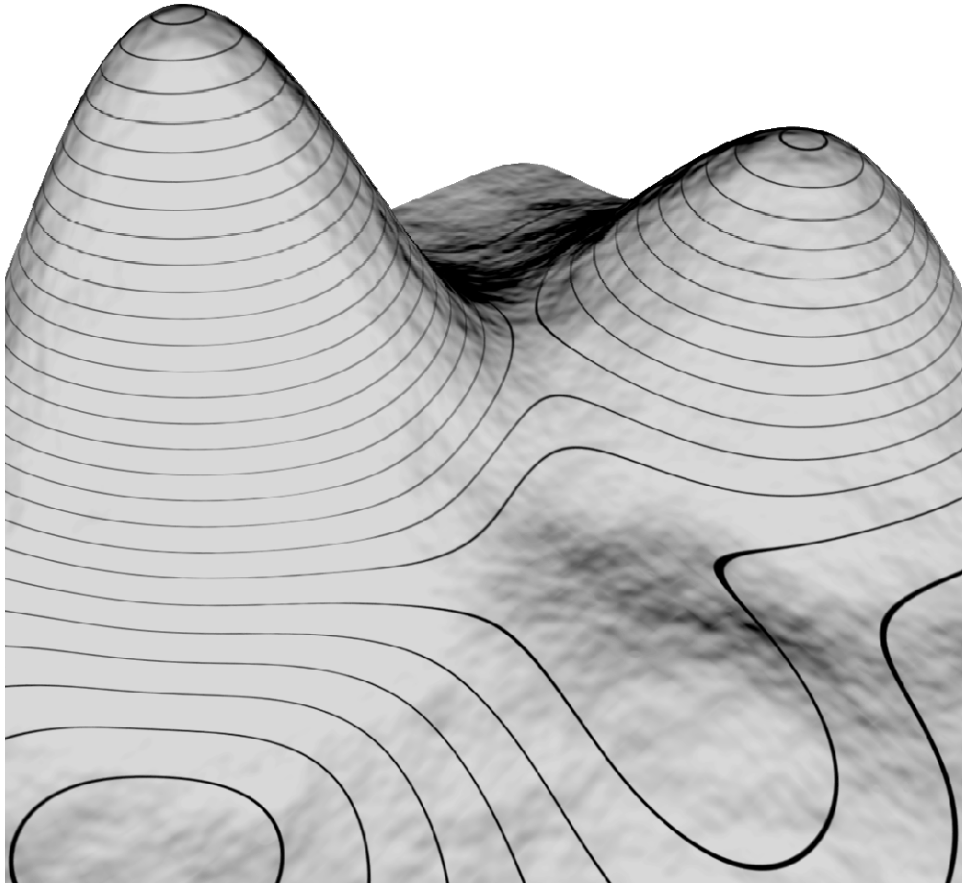




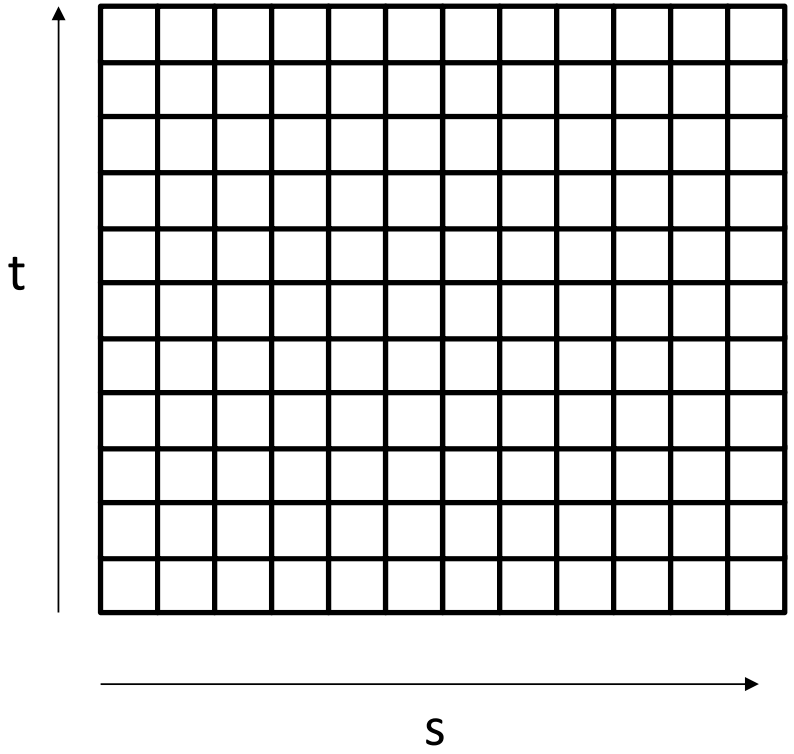
s

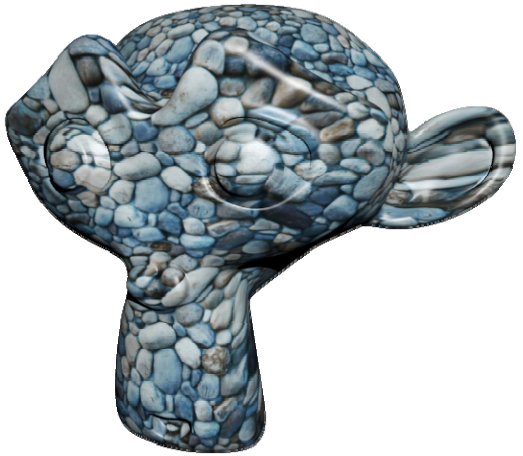
Textures 1D

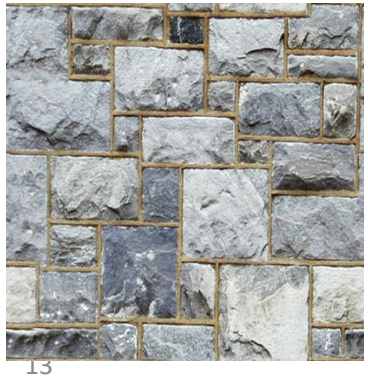




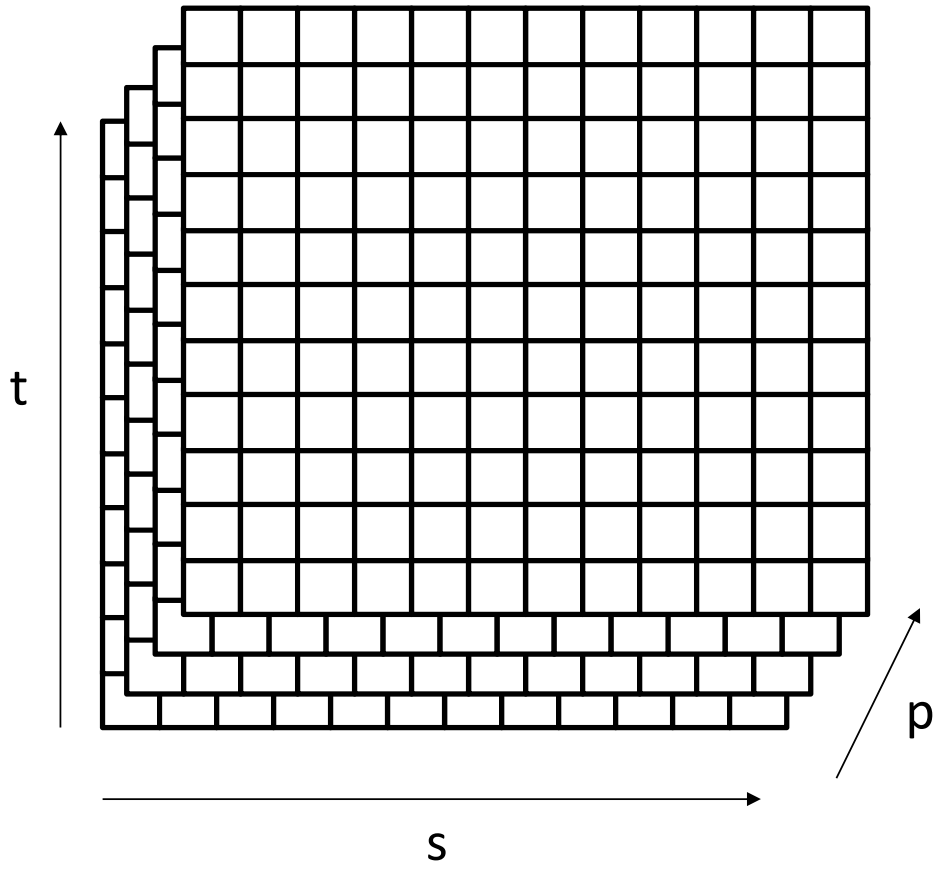
Textures 2D



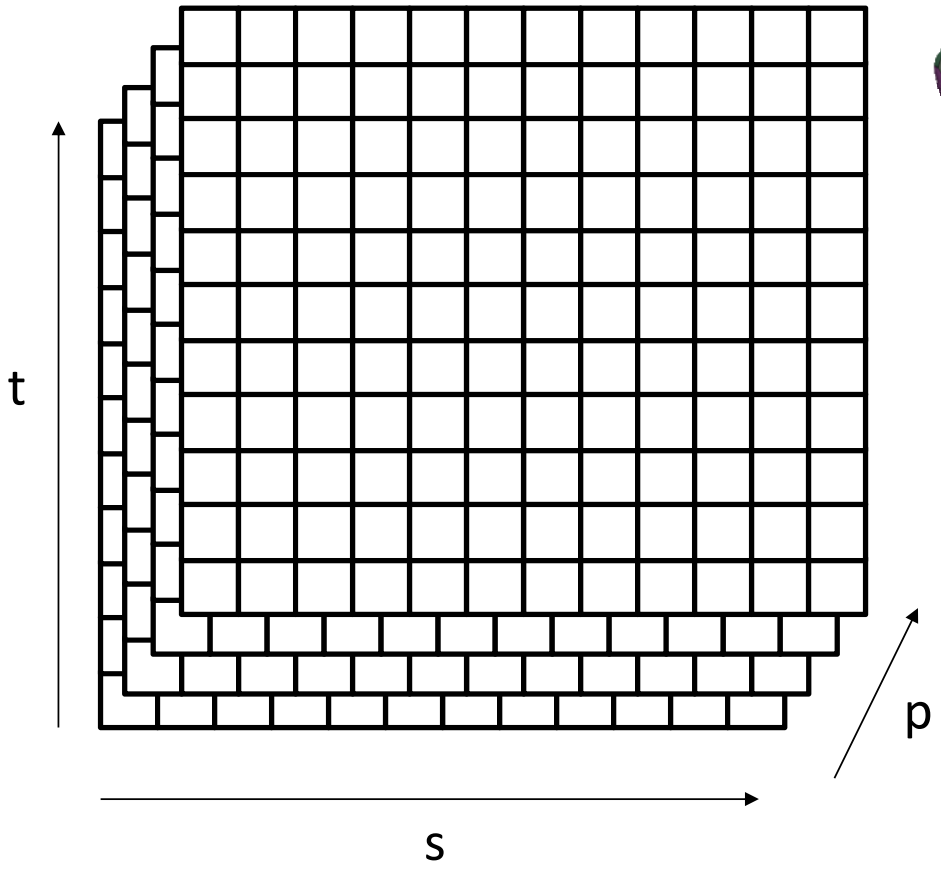


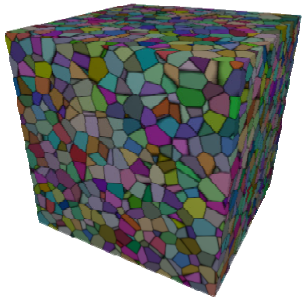


Textures 3D

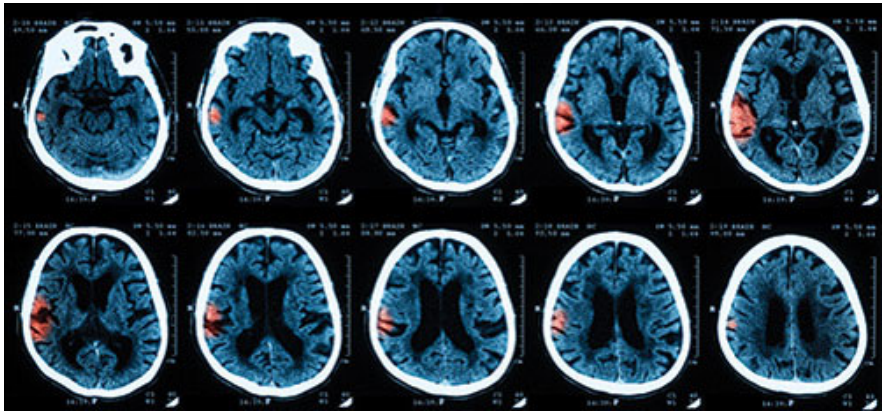


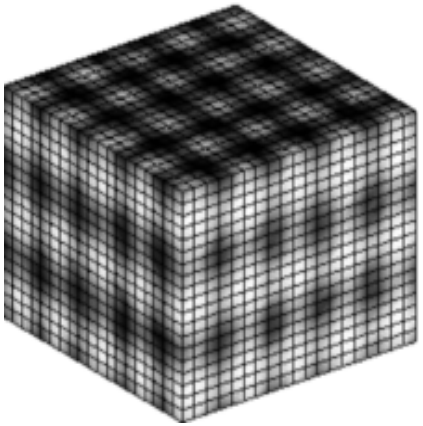
Textures 3D



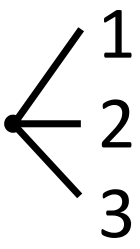


Textures 3D

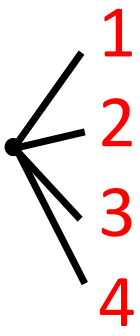




Textures

Una textura és una taula de 

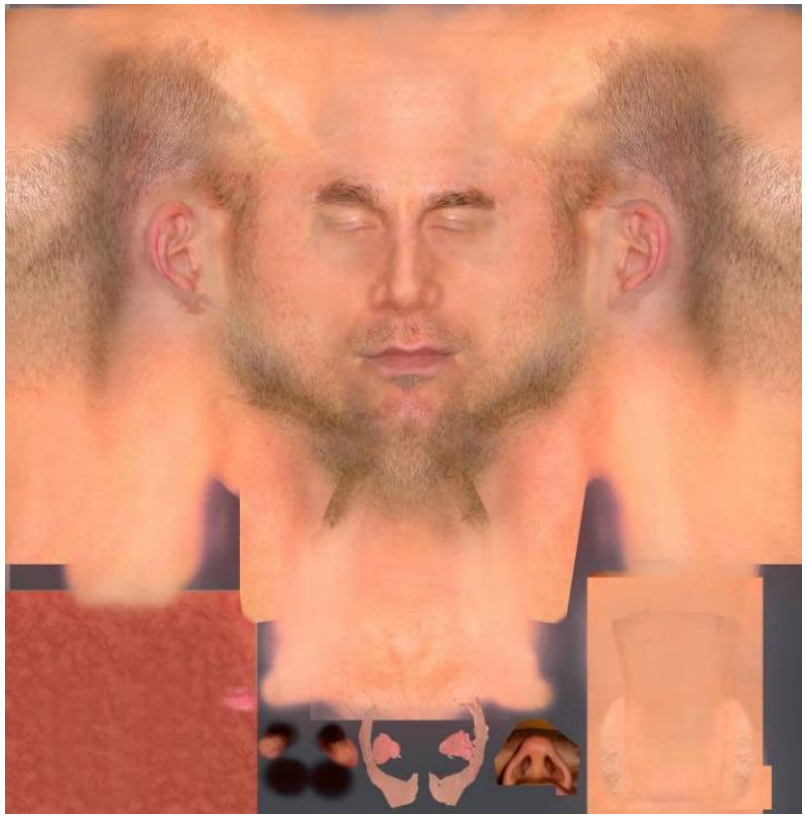
dimensions, on cada cel·la

enmagatzema una certa propietat amb 

canals.

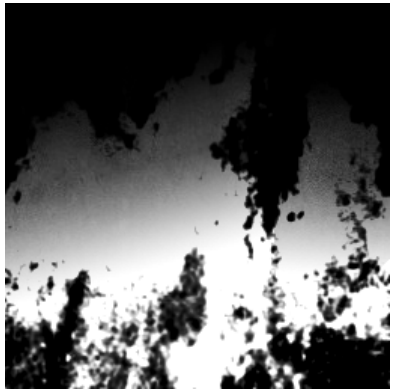
Habitualment les textures s'utilitzen al FS, tot i que també es poden usar en altres shaders.

K_d (color map)

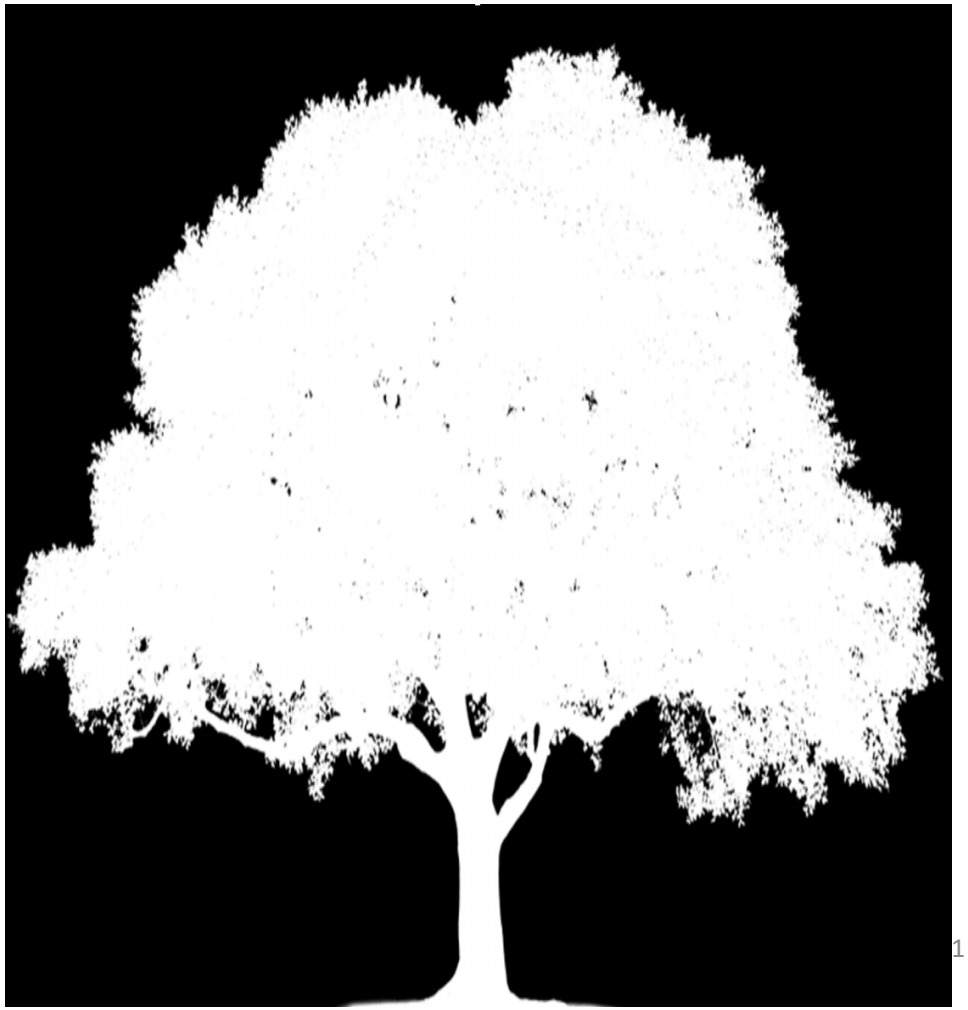


K_s (gloss map)

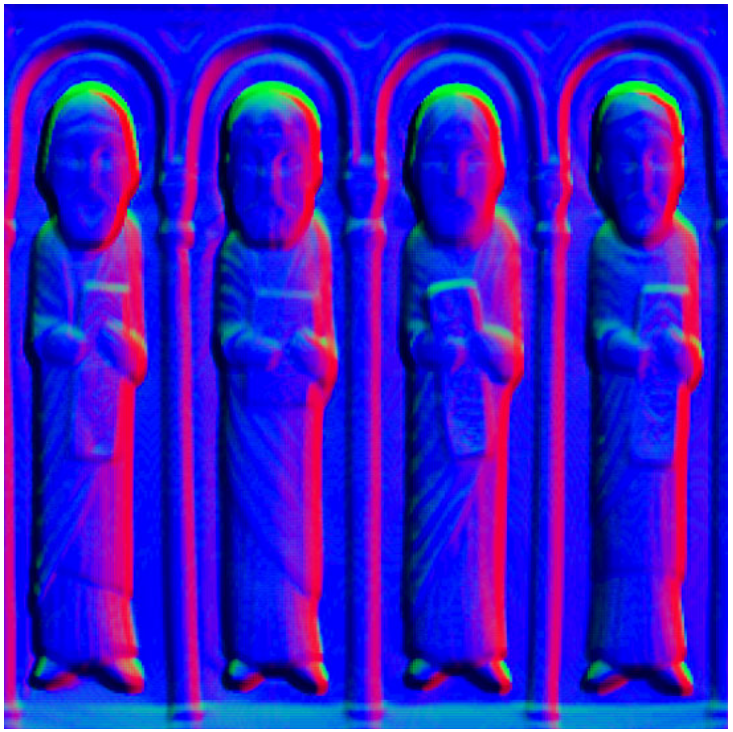
K_s (gloss map)



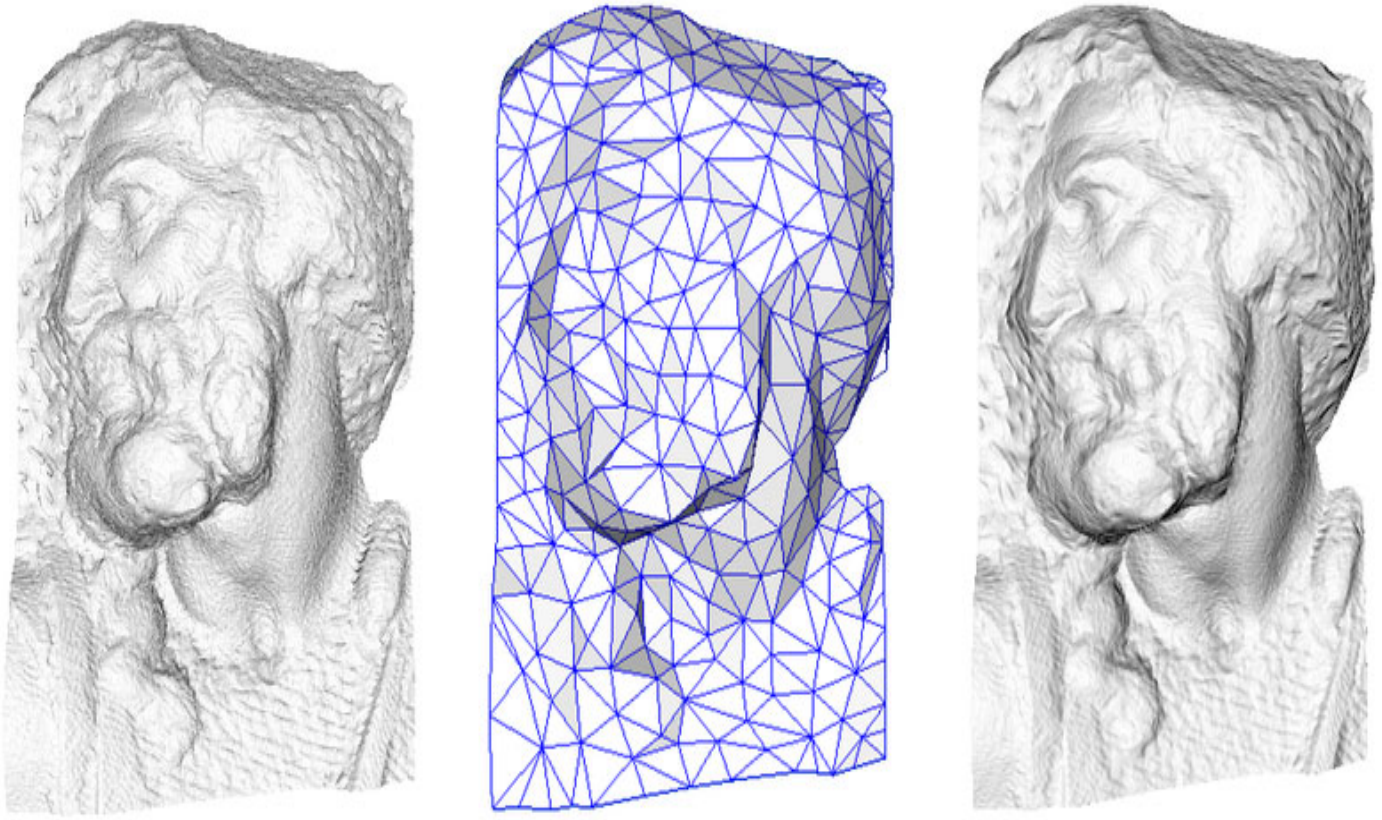
Opacitat (opacity map, alpha mask)



Normal (normal map)



Normal (normal mapping)



original mesh
4M triangles

simplified mesh
500 triangles

simplified mesh
and normal mapping
500 triangles

Desplaçament (bump mapping)

Desplaçament (displacement mapping)

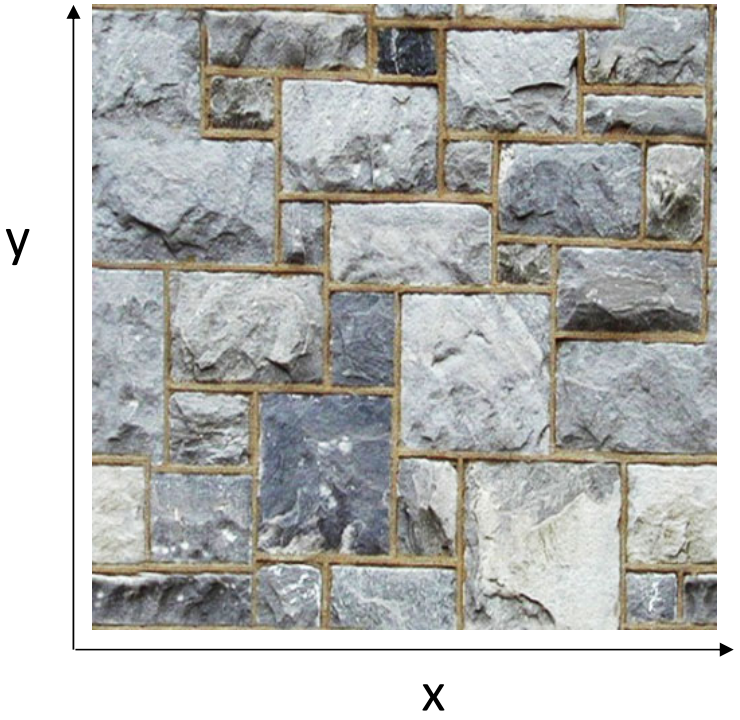
Mida d’una textura

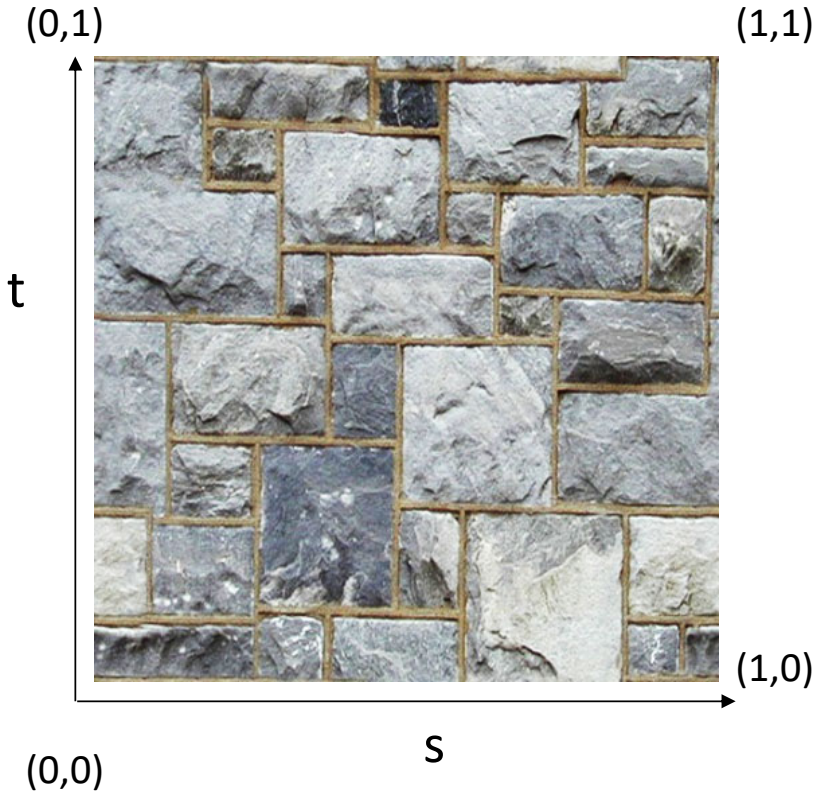
- # texels en cada dimensió
- Habitualment w, h són potència de 2.

width

height

Espai normalitzat de textura

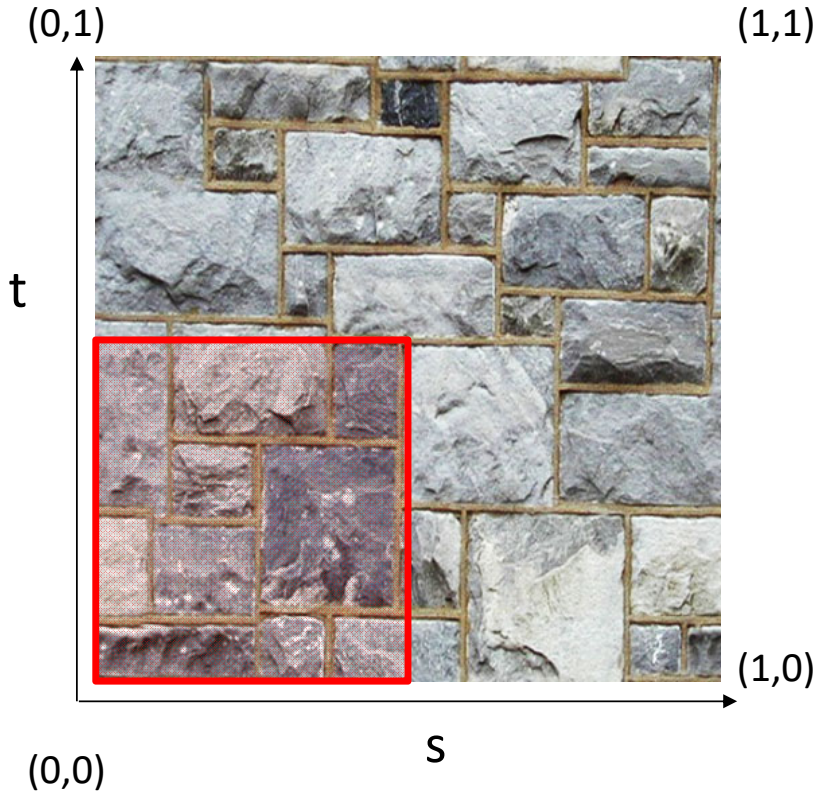




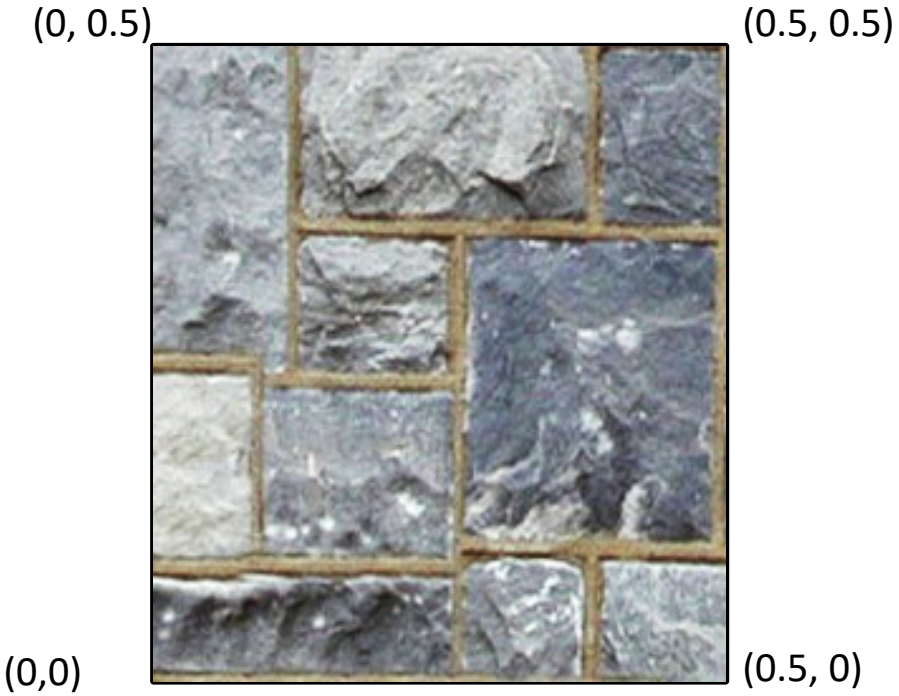
x ϵ [0, width] y ϵ [0, height]

s ϵ IR t ϵ IR

Espai normalitzat de textura

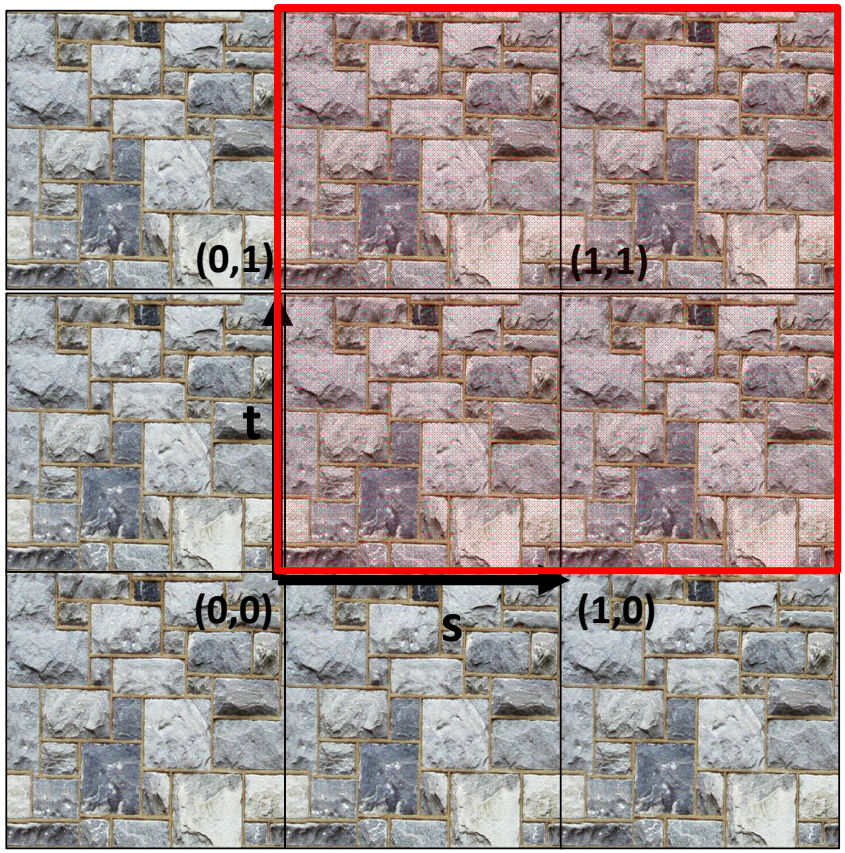


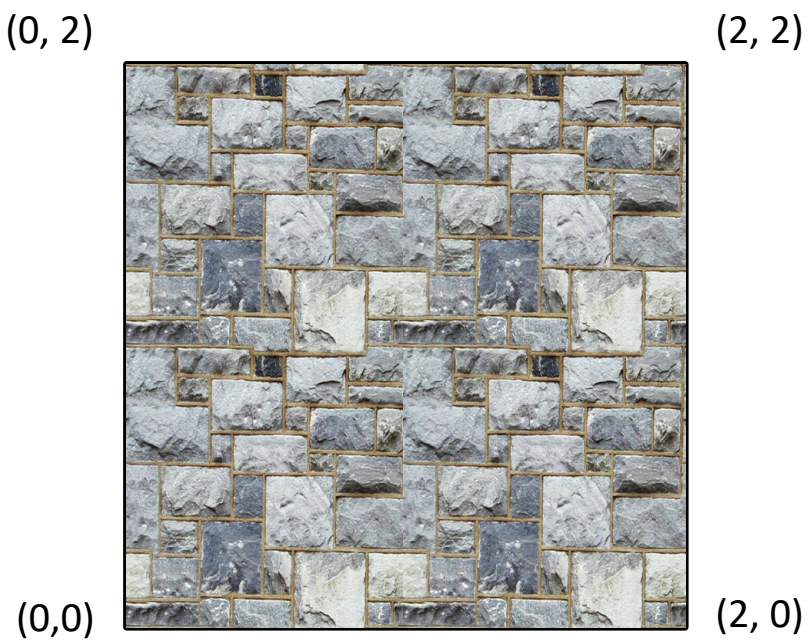
$s \in IR$ $t \in IR$



Quadrat texturat

Espai normalitzat de textura

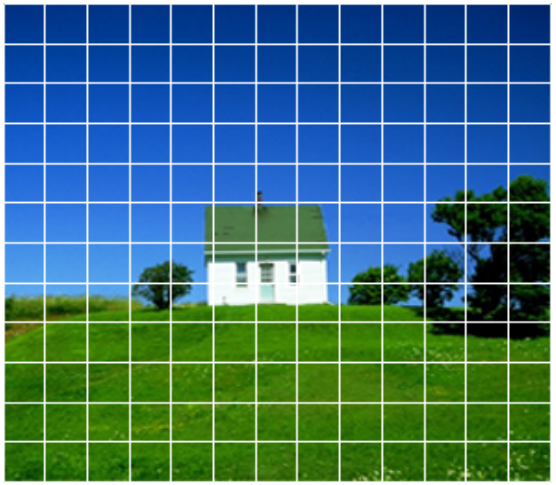


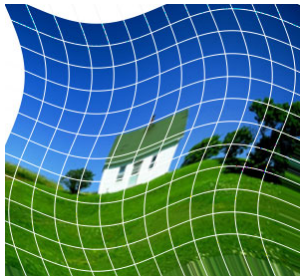


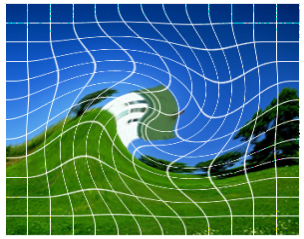
Quadrat texturat

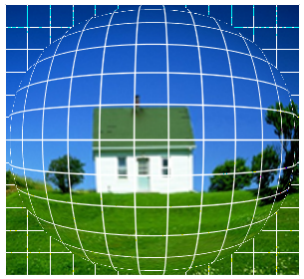
MAPPING

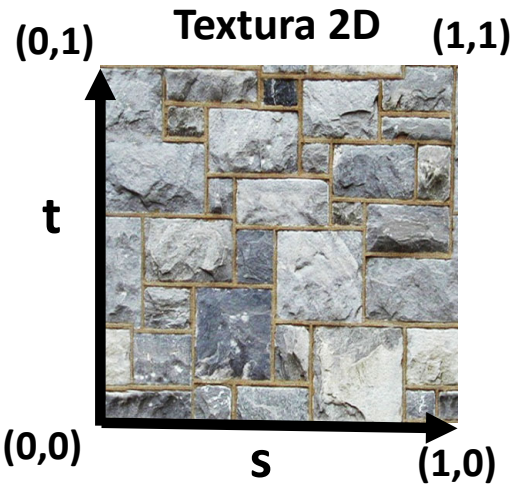
Forward / Inverse mapping

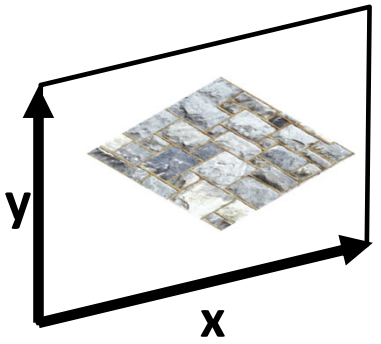






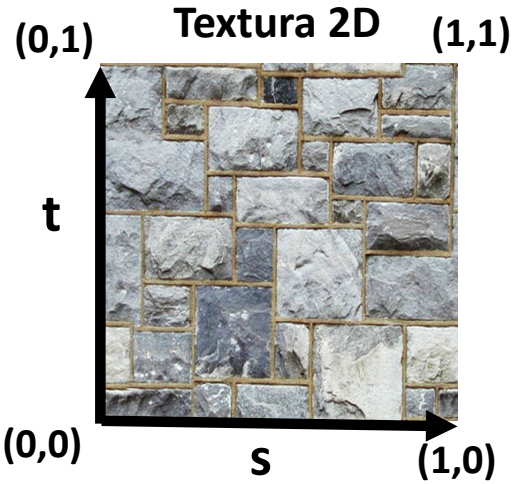


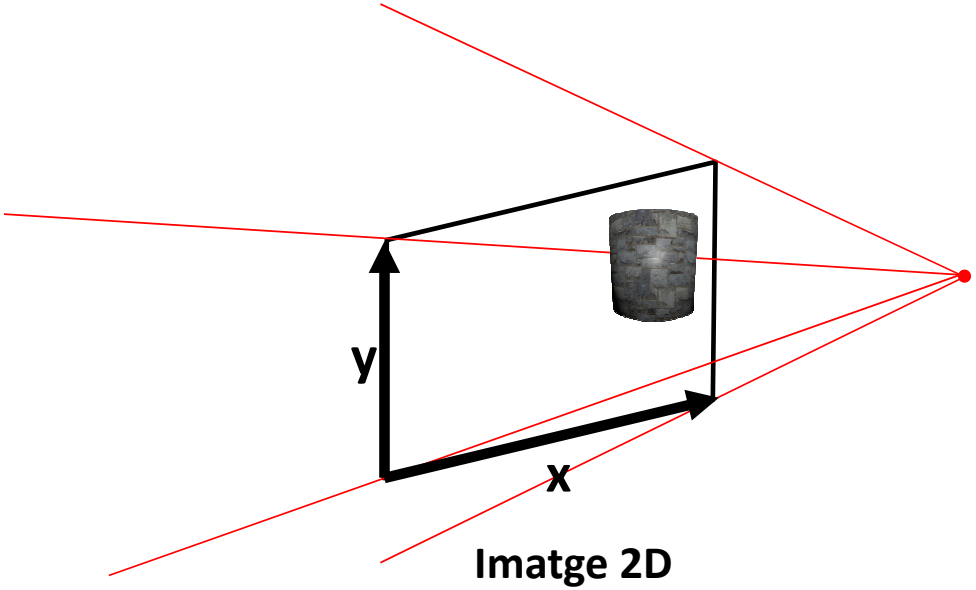




Imatge 2D

Forward / Inverse mapping

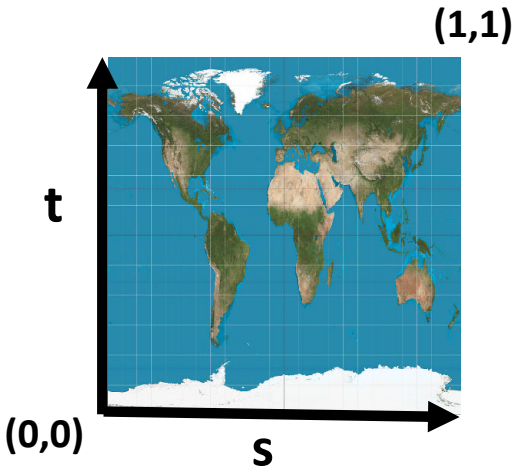


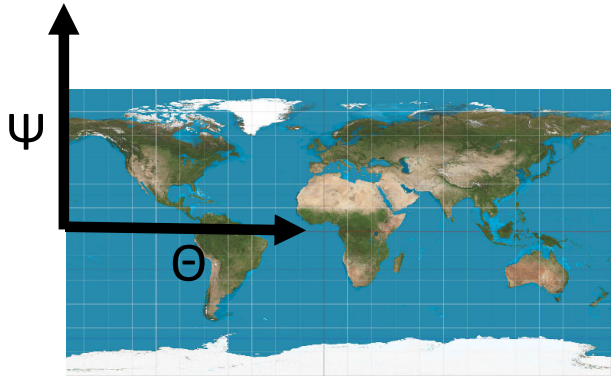


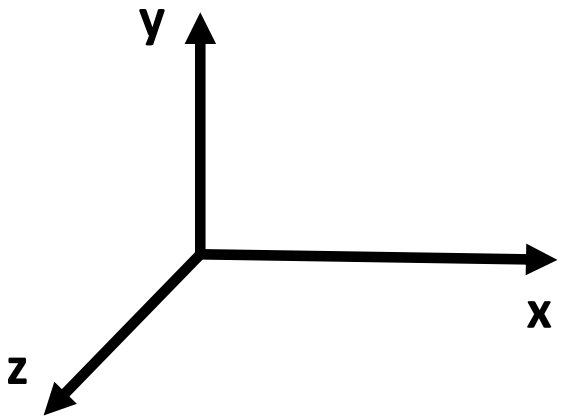
Exemple 1: Mapping esfèric

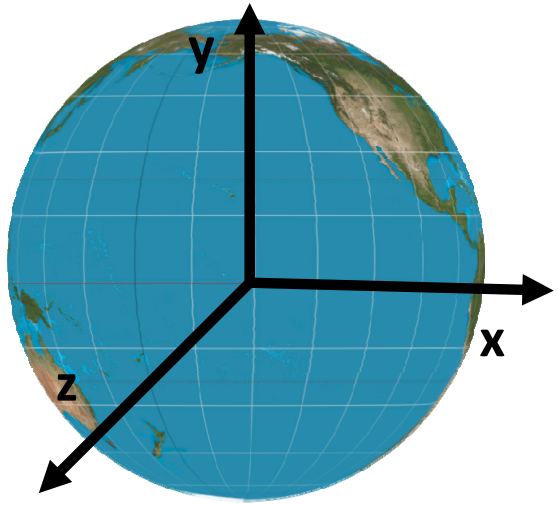
Input: s ∈ [0,1], t ∈ [0,1]

Output: x,y,z ∈ esfera unitat

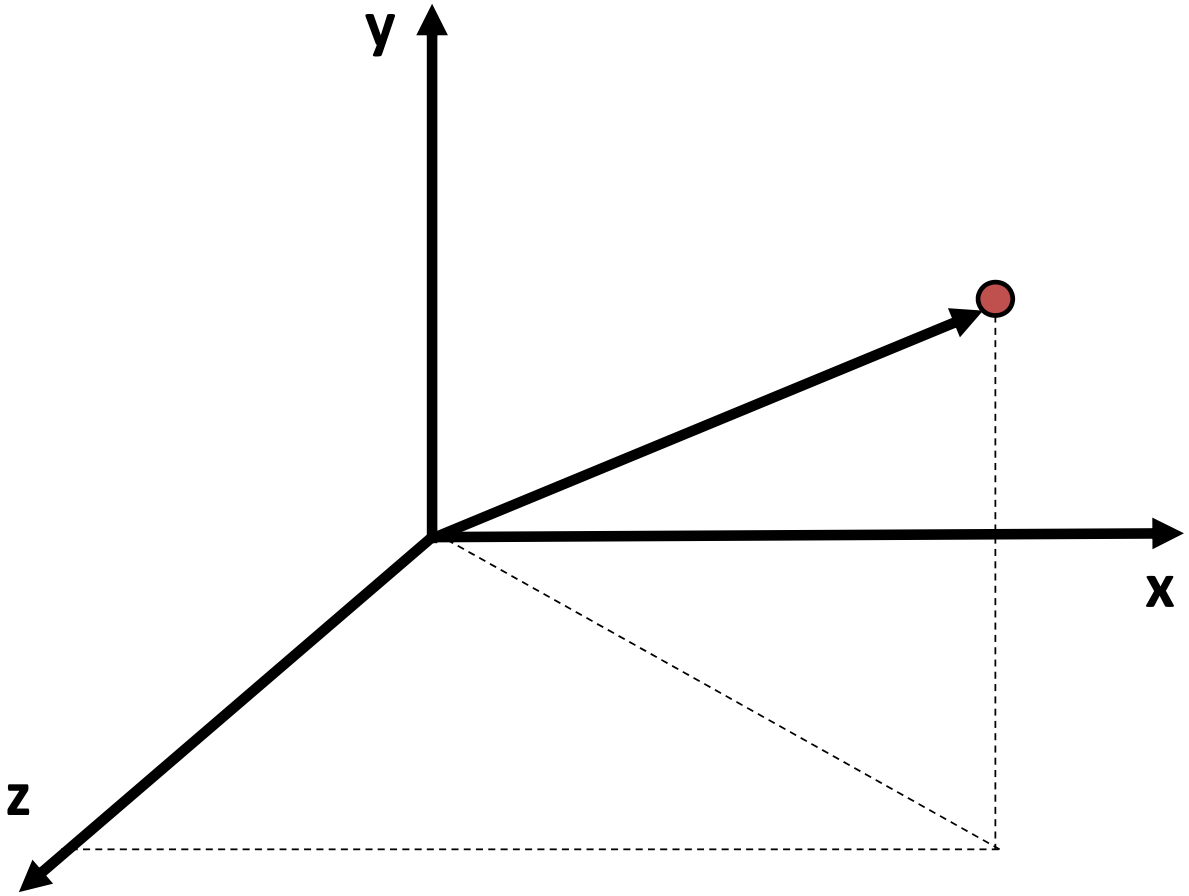


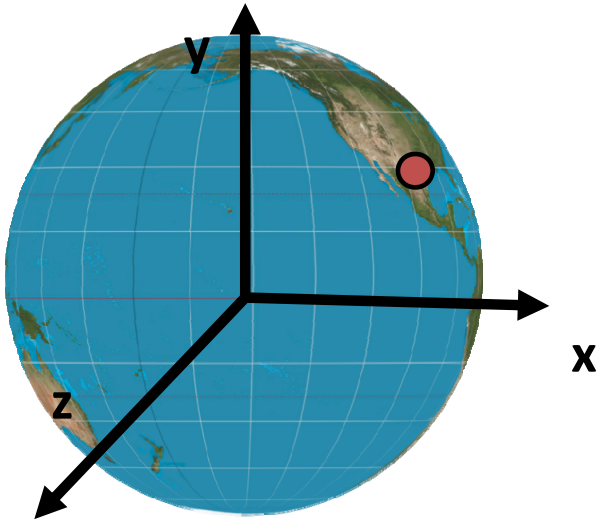






Exemple 1: Mapping esfèric



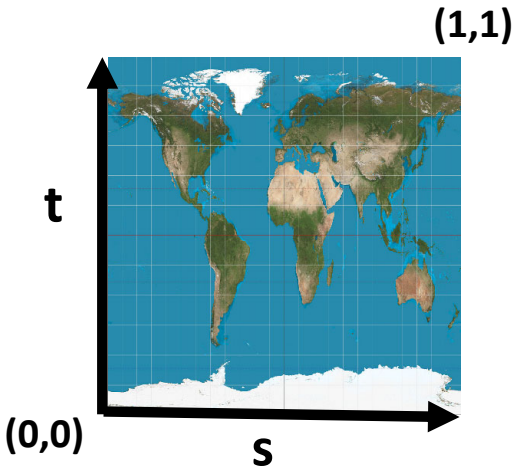


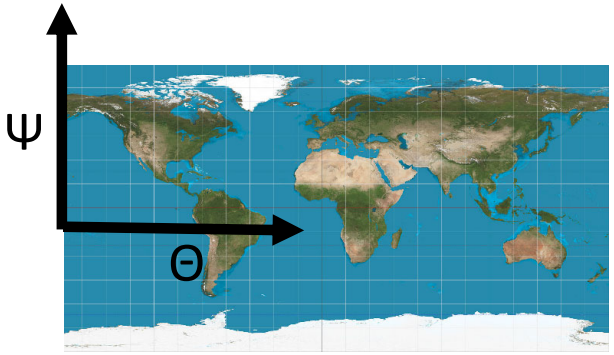
Exemple 1: Mapping esfèric

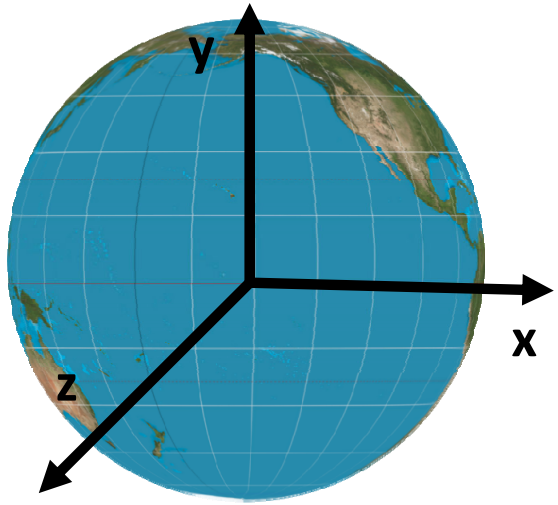
Input: s ∈ [0,1], t ∈ [0,1]
Output: x,y,z ∈ esfera unitat

// pas (s, t) → (Θ, Ψ)
Θ = 2πs;
Ψ = π(t-0.5);

// pas esfèriques → (x,y,z)
x = sin(Θ)cos(Ψ);
y = sin(Ψ);
z = cos(Θ)cos(Ψ);







Exemple 2: Mapping cilíndric

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

Output: $x,y,z \in$ cilindre r=1 sobre pla XZ

// pas (s, t) → (Θ, h)

$\Theta = 2\pi s;$

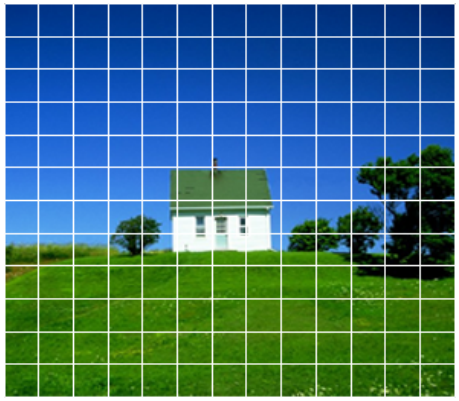
$h = t;$

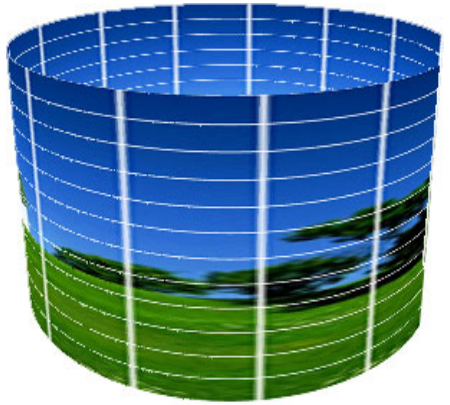
// pas cilíndriques → (x,y,z)

$x = \sin(\Theta);$

$y = h;$

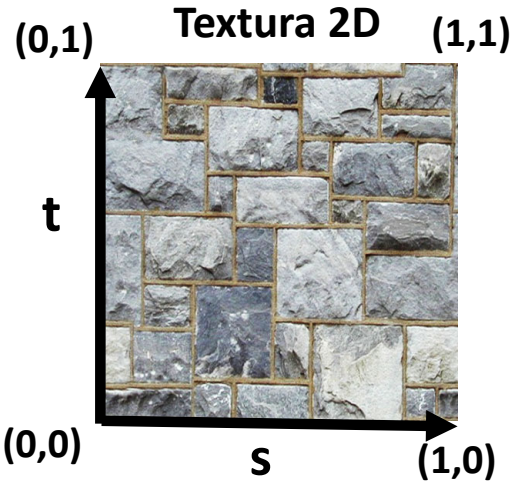
$z = \cos(\Theta);$

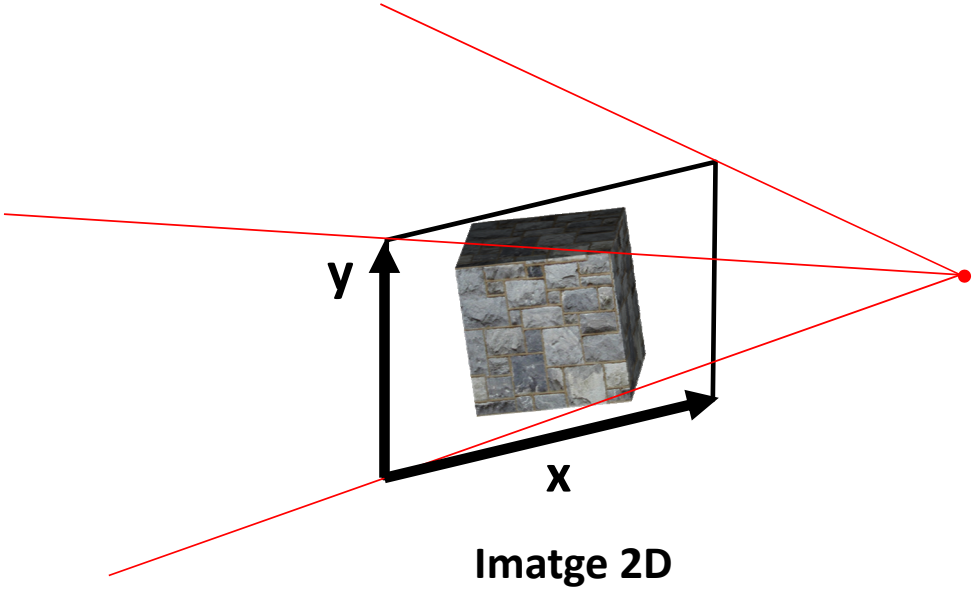




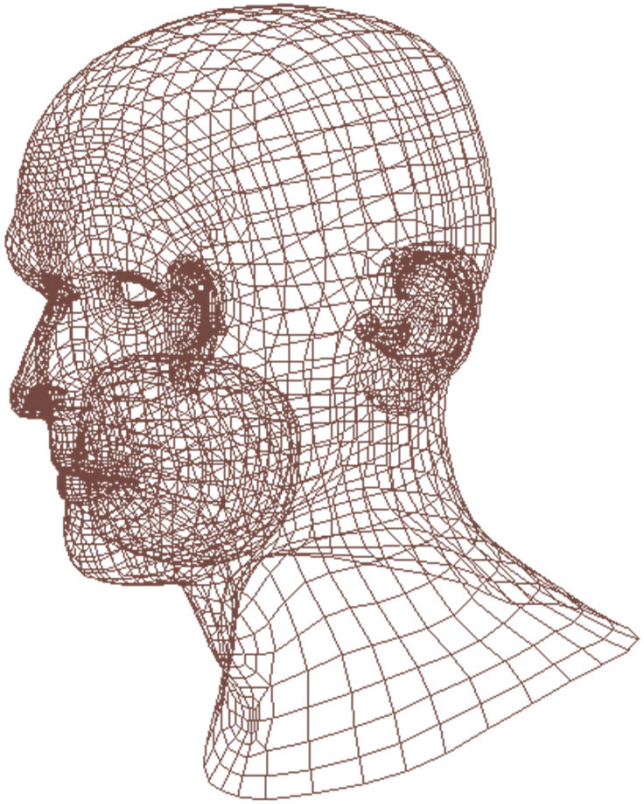
MAPPING EN OPENGL

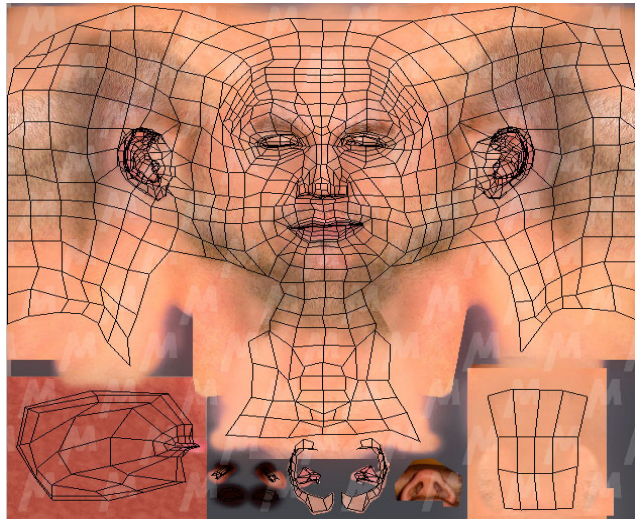
En gràfics, habitualment





En gràfics, habitualment



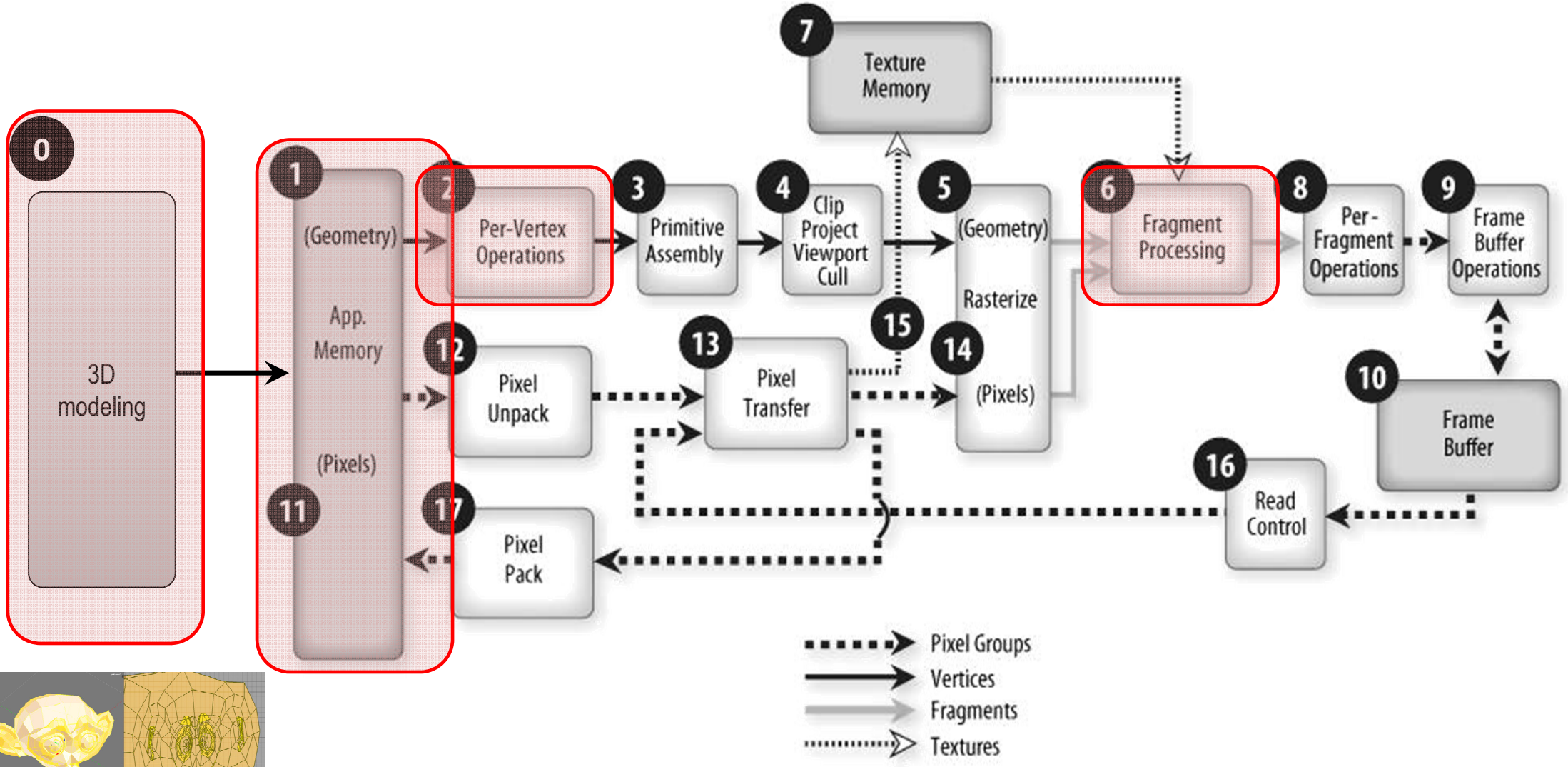


DEMO (TEXTURE MAPPING AMB BLENDER)

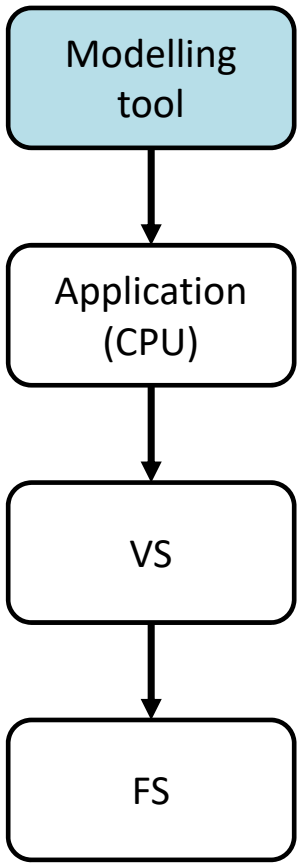
DEMO (TEXTURE MAPPING AMB OPENGL)

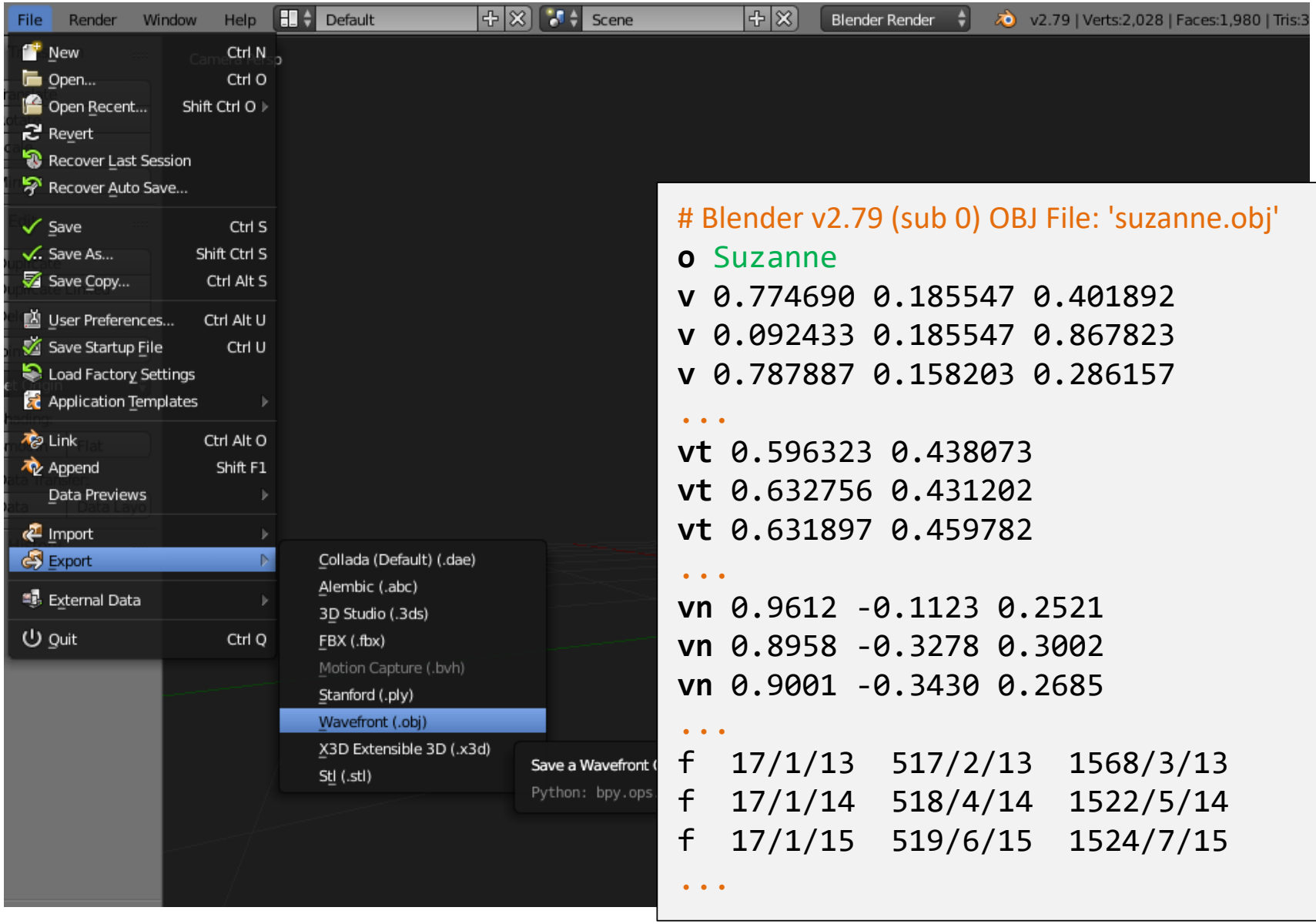
En quina etapa es generen les coordenades de textura?

COORDENADES DE TEXTURA AL PIPELINE


0
3D modeling
1
(Geometry)
App. Memory
(Pixels)
11
2
Per-Vertex Operations
3
Primitive Assembly
4
Clip Project Viewport Cull
5
(Geometry)
Rasterize
(Pixels)
6
Fragment Processing
7
Texture Memory
8
Per-Fragment Operations
9
Frame Buffer Operations
10
Frame Buffer
12
Pixel Unpack
13
Pixel Transfer
14
15
16
Read Control
17
Pixel Pack
Pixel Groups
Vertices
Fragments
Textures

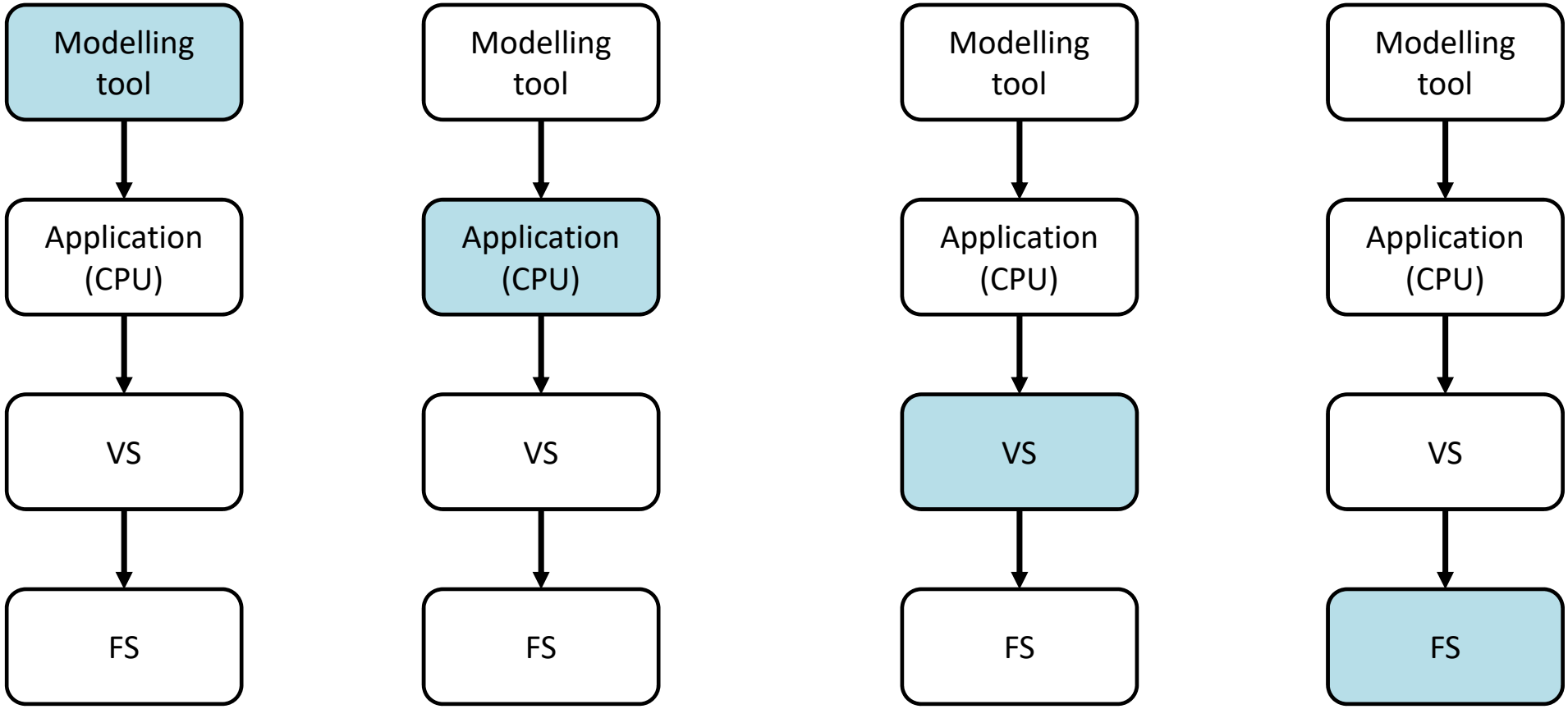
Opci






```
# Blender v2.79 (sub 0) OBJ File: 'suzanne.obj'
o Suzanne
v 0.774690 0.185547 0.401892
v 0.092433 0.185547 0.867823
v 0.787887 0.158203 0.286157
...
vt 0.596323 0.438073
vt 0.632756 0.431202
vt 0.631897 0.459782
...
vn 0.9612 -0.1123 0.2521
vn 0.8958 -0.3278 0.3002
vn 0.9001 -0.3430 0.2685
...
f  17/1/13  517/2/13  1568/3/13
f  17/1/14  518/4/14  1522/5/14
f  17/1/15  519/6/15  1524/7/15
...
```

Opcions per a generar coords de textura



MÈTODES PER GENERAR COORDS DE TEXTURA

Generació de coordenades de textura

- Bàsics
  - Amb plans S, T
  - Amb superfície auxiliar (S-mapping, O-mapping)

- Avançats (mesh parameterization)
  - Mesh partition
  - Area-preserving, Angle-preserving, Stretch-preserving...

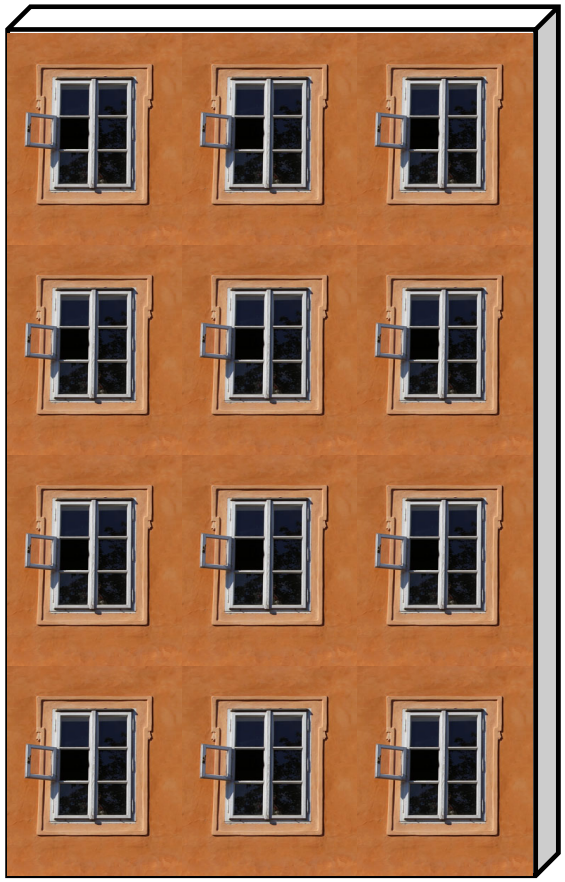
Plans S,T

Paràmetres:

- plans S,T

Càlcul s, t:

Plans S,T (exemple)

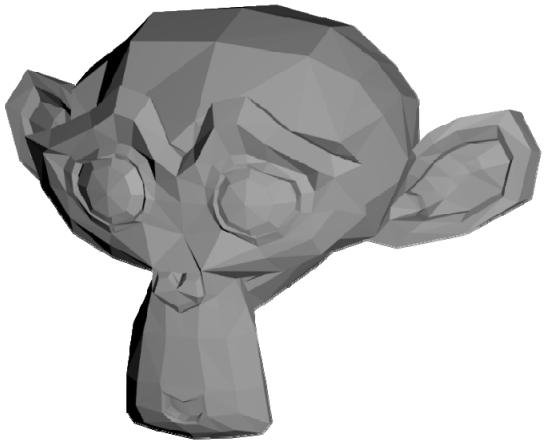


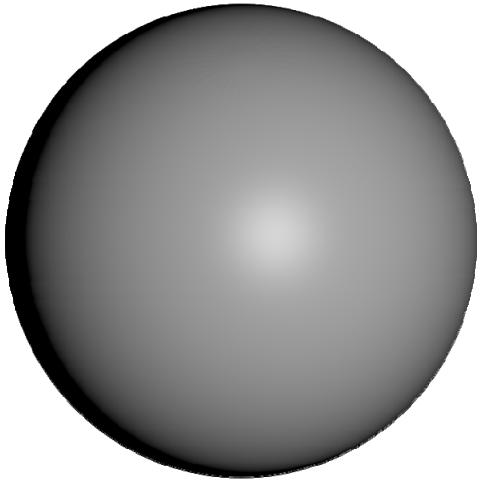
Generació de coordenades de textura

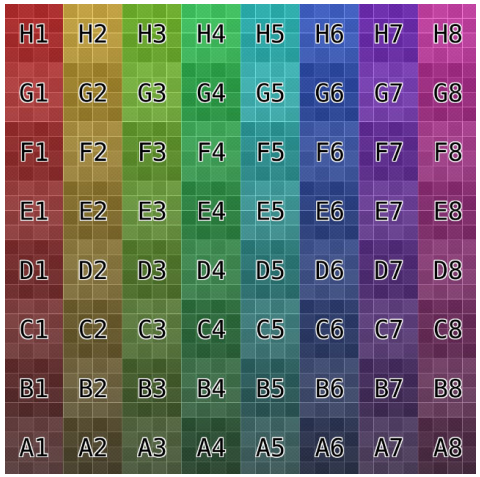
- Bàsics
  - Amb plans S, T
  - Amb superfície auxiliar (S-mapping, O-mapping)

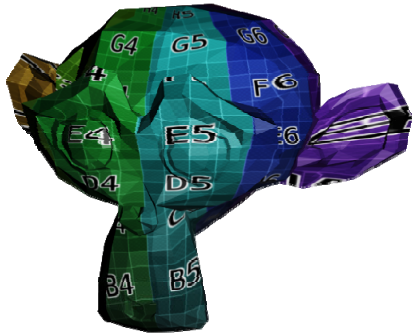
- Avançats (mesh parameterization)
  - Mesh partition
  - Area-preserving, Angle-preserving, Stretch-preserving...

Parametrització amb superfície auxiliar

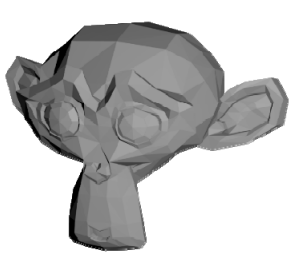


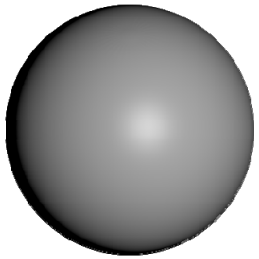


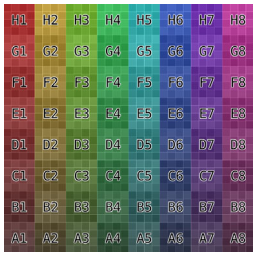




Exemples: S^{-1} mappings





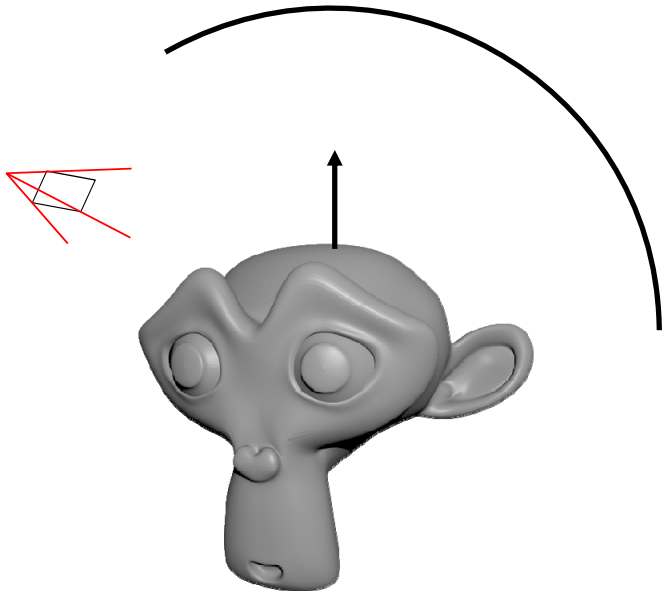


Input: $s \in [0,1]$, $t \in [0,1]$
Output: $x,y,z \in$ esfera unitat

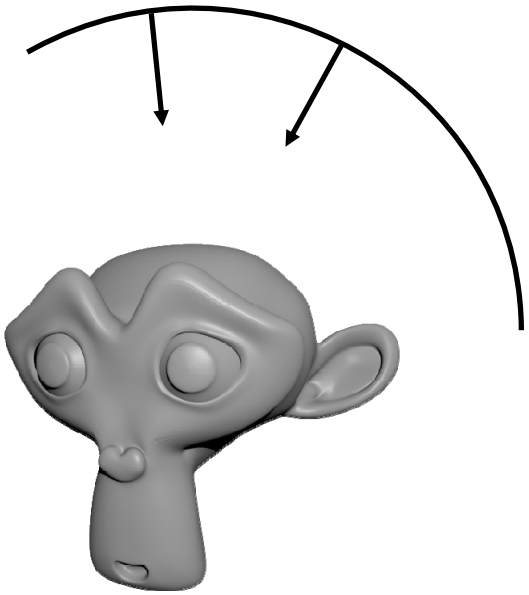
// pas (s, t) → (Θ, Ψ)
$\Theta = 2\pi s;$
$\Psi = \pi(t-0.5);$

// pas esfèriques → (x,y,z)
$x = \sin(\Theta)\cos(\Psi);$
$y = \sin(\Psi);$
$z = \cos(\Theta)\cos(\Psi);$

Exemples: O mappings

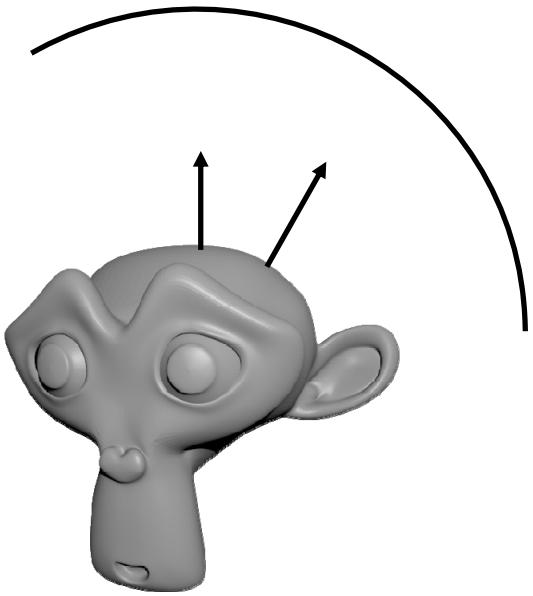


Reflected view ray

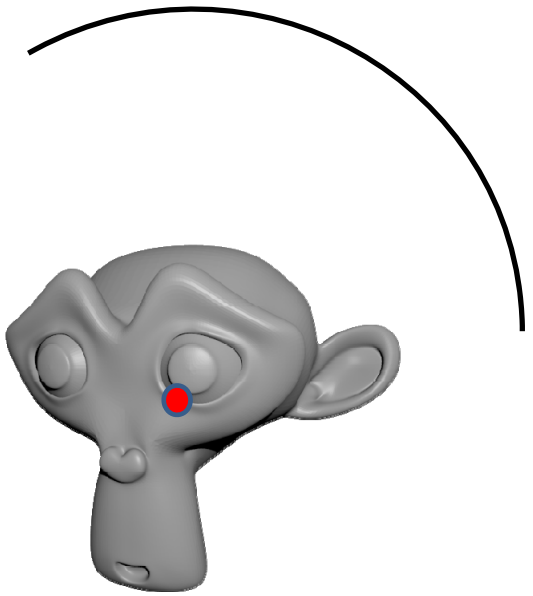


Intermediate surface normal

Exemples: O mappings

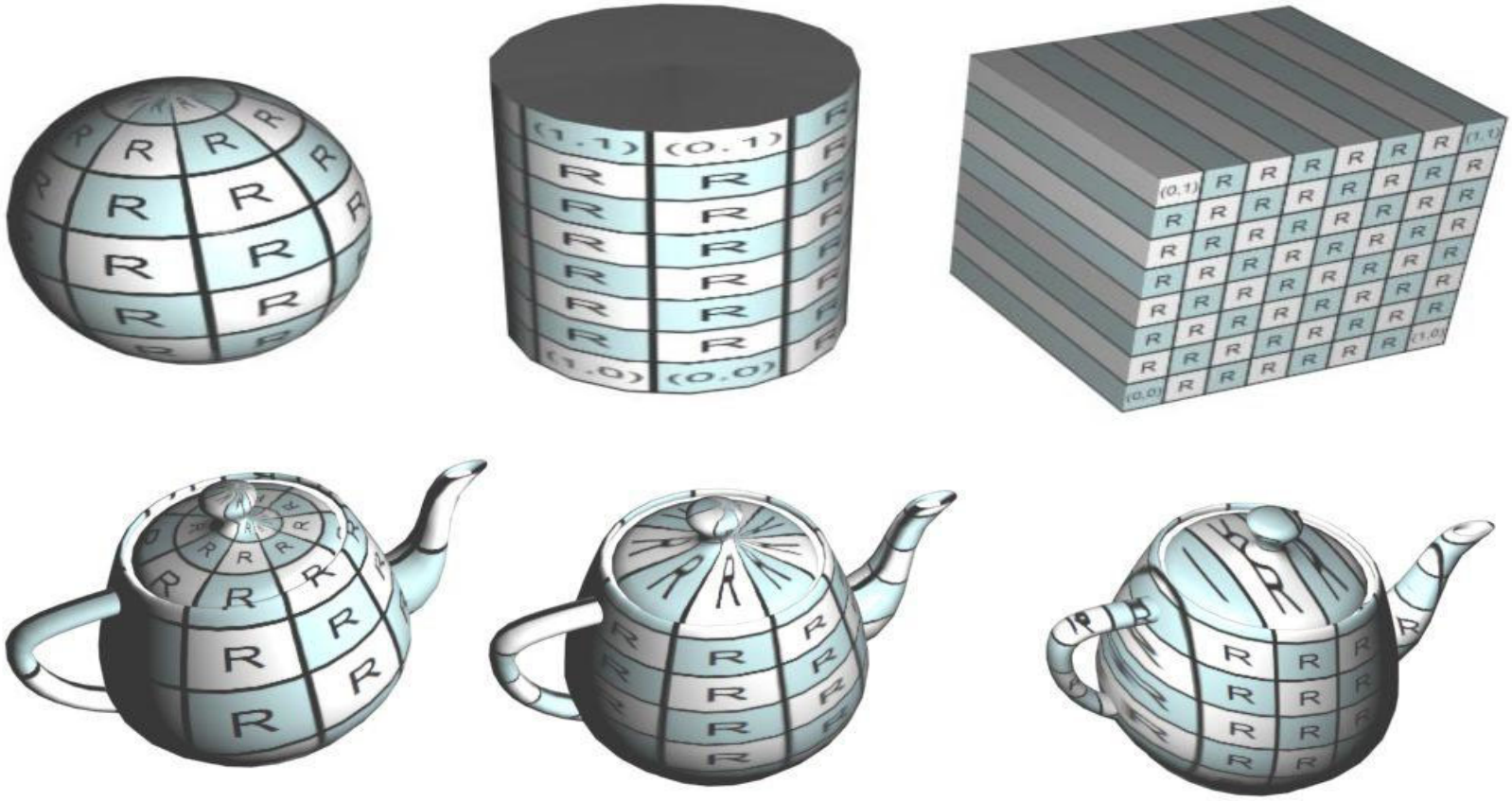


Object Normal

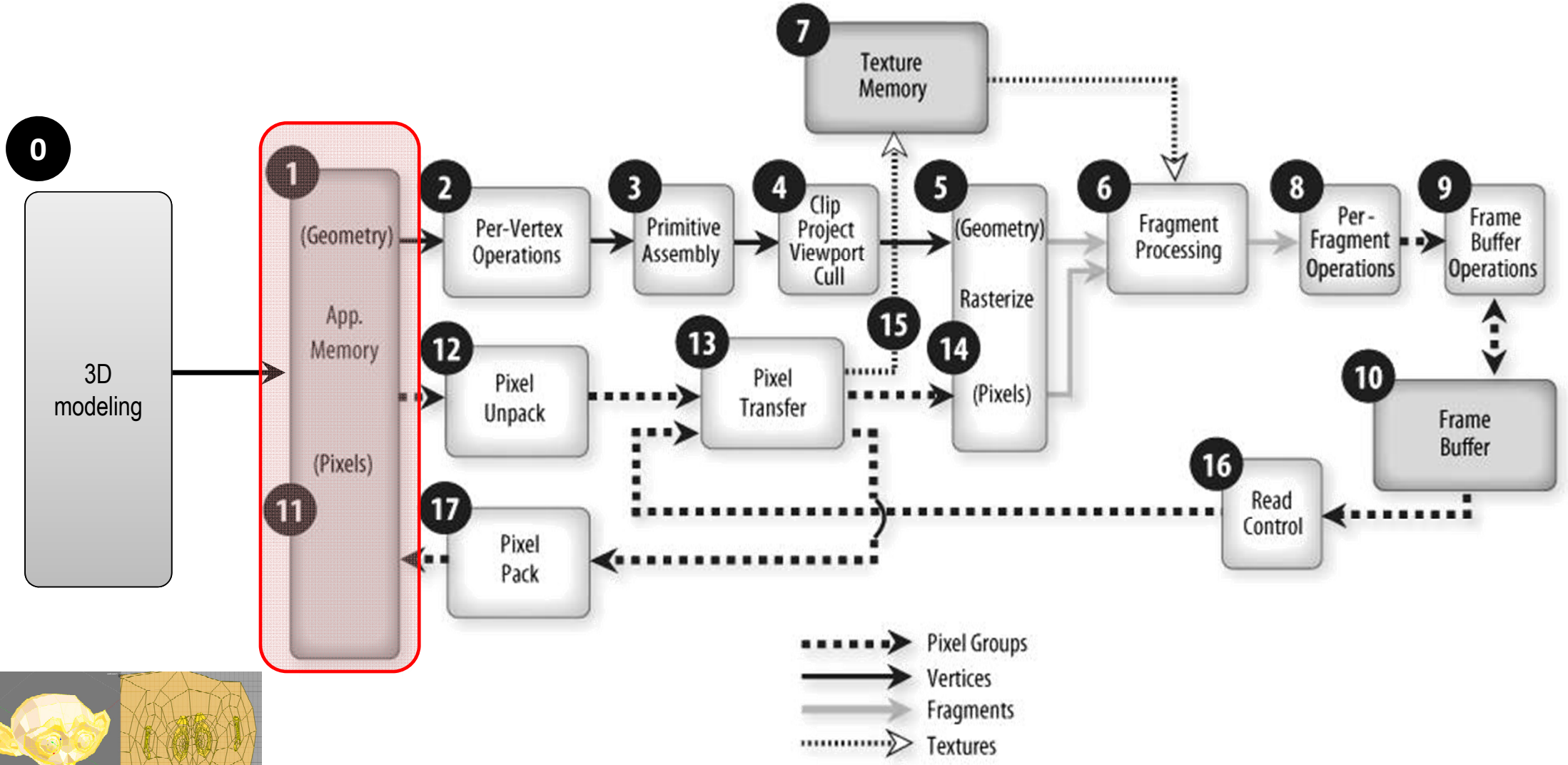


Object centroid

Projeccions esfèrica, cilíndrica i plana

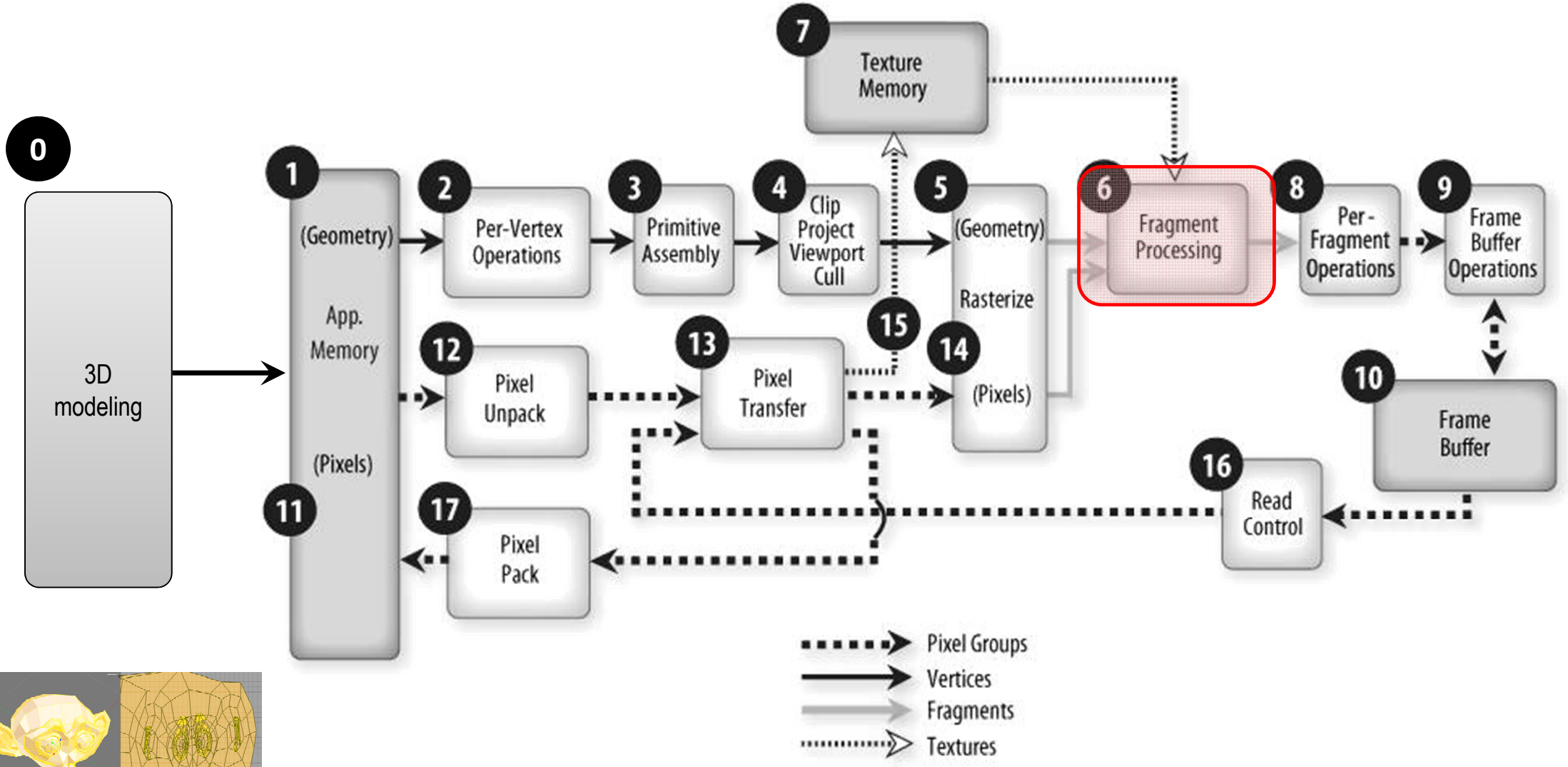


CREACIÓ DE LA TEXTURA


0
3D modeling
1
(Geometry)
App. Memory
(Pixels)
11
2
Per-Vertex Operations
3
Primitive Assembly
4
Clip Project Viewport Cull
5
(Geometry)
Rasterize
(Pixels)
6
Fragment Processing
7
Texture Memory
8
Per-Fragment Operations
9
Frame Buffer Operations
10
Frame Buffer
12
Pixel Unpack
13
Pixel Transfer
14
15
16
Read Control
17
Pixel Pack
Pixel Groups
Vertices
Fragments
Textures

```
// Load Texture (once)
QImage img0("fieldstone.png");
QImage T = img0.convertToFormat(QImage::Format_ARGB32);
glGenTextures( 1, &textureId0);
glBindTexture(GL_TEXTURE_2D, textureId0);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, T.width(), T.height(), 0,
             GL_RGBA, GL_UNSIGNED_BYTE, T.bits());

…
// Bind textures, set uniforms...
g.glActiveTexture(GL_TEXTURE0);
g.glBindTexture(GL_TEXTURE_2D, textureId0);
program->bind();
program->setUniformValue("colorMap", 0);

…
```

ÚS DE TEXTURA AL FRAGMENT SHADER


0
3D modeling
1
(Geometry)
App. Memory
(Pixels)
2
Per-Vertex Operations
3
Primitive Assembly
4
Clip Project Viewport Cull
5
(Geometry)
Rasterize
(Pixels)
6
Fragment Processing
7
Texture Memory
8
Per-Fragment Operations
9
Frame Buffer Operations
10
Frame Buffer
11
12
Pixel Unpack
13
Pixel Transfer
14
15
16
Read Control
17
Pixel Pack
Pixel Groups
Vertices
Fragments
Textures

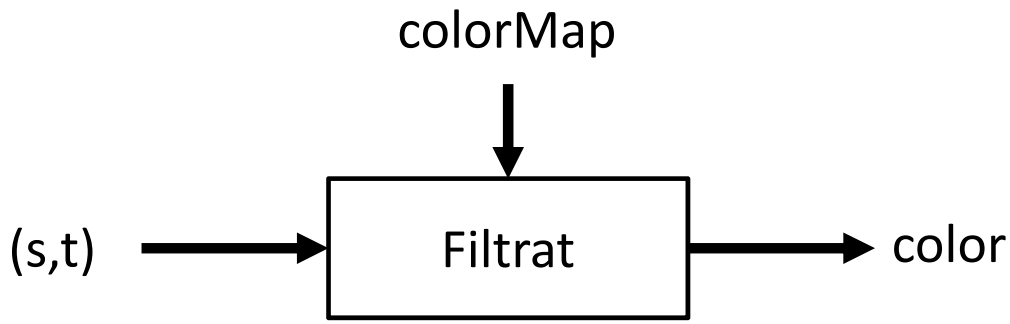
FS – exemple

```
uniform sampler2D colorMap;
in vec2 vtexcoord;
...

vec4 color = texture(colorMap, vtexcoord);
...
```

Magnification filters, Minification filters, Mipmapping

FILTRAT

Accés a textura



Necessitat del filtrat

Objeto texturado

Textura

Necessitat del filtrat

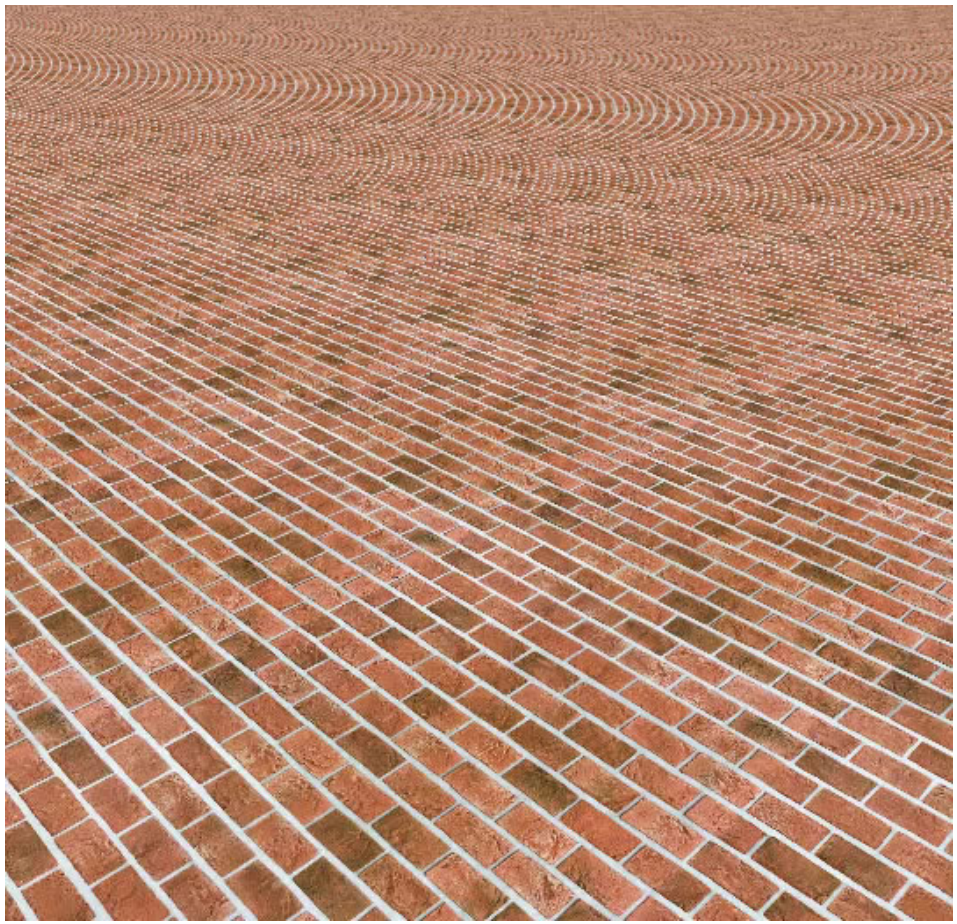
Textura

Magnification ≈ upsampling

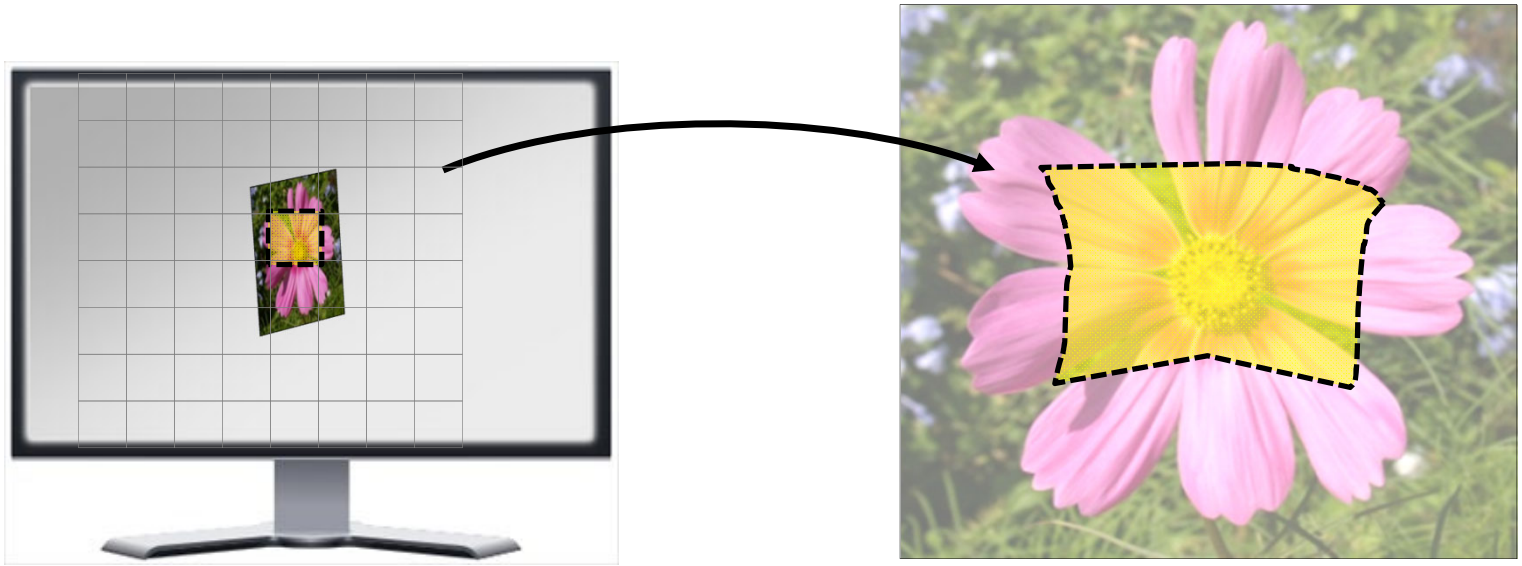
Minification ≈ downsampling

Magnification (naïve)

Minification (naïve)

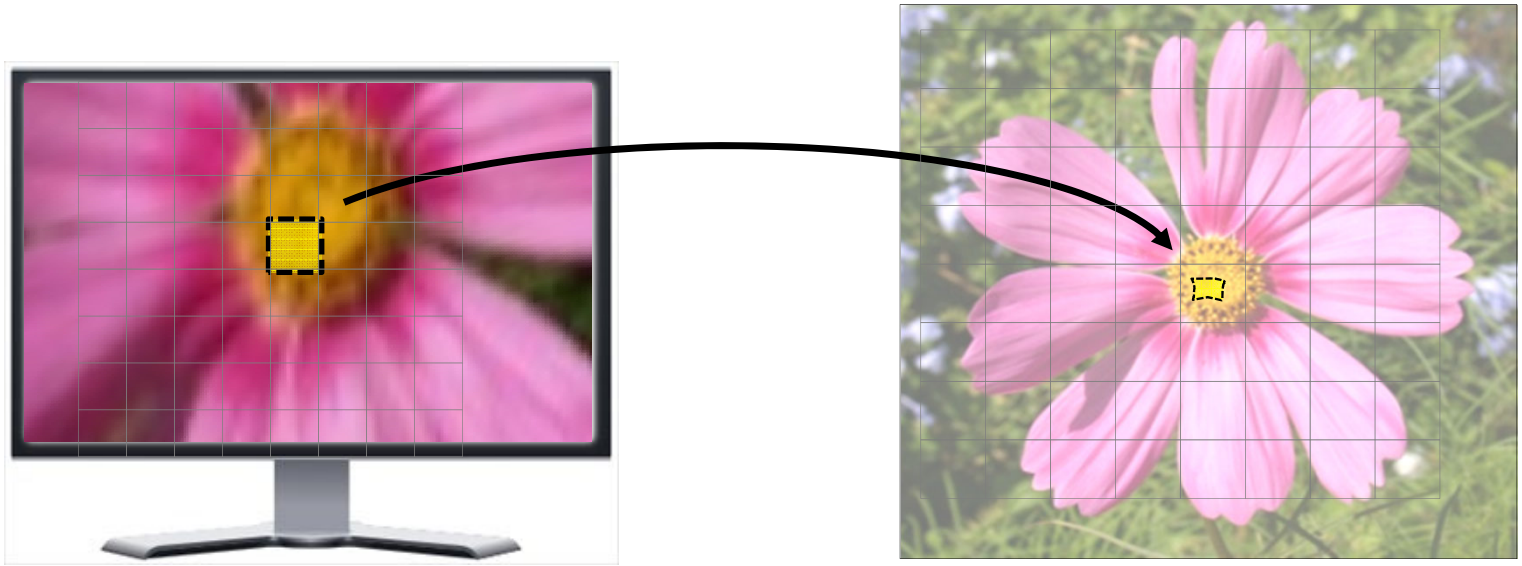


Preimatge d’un fragment



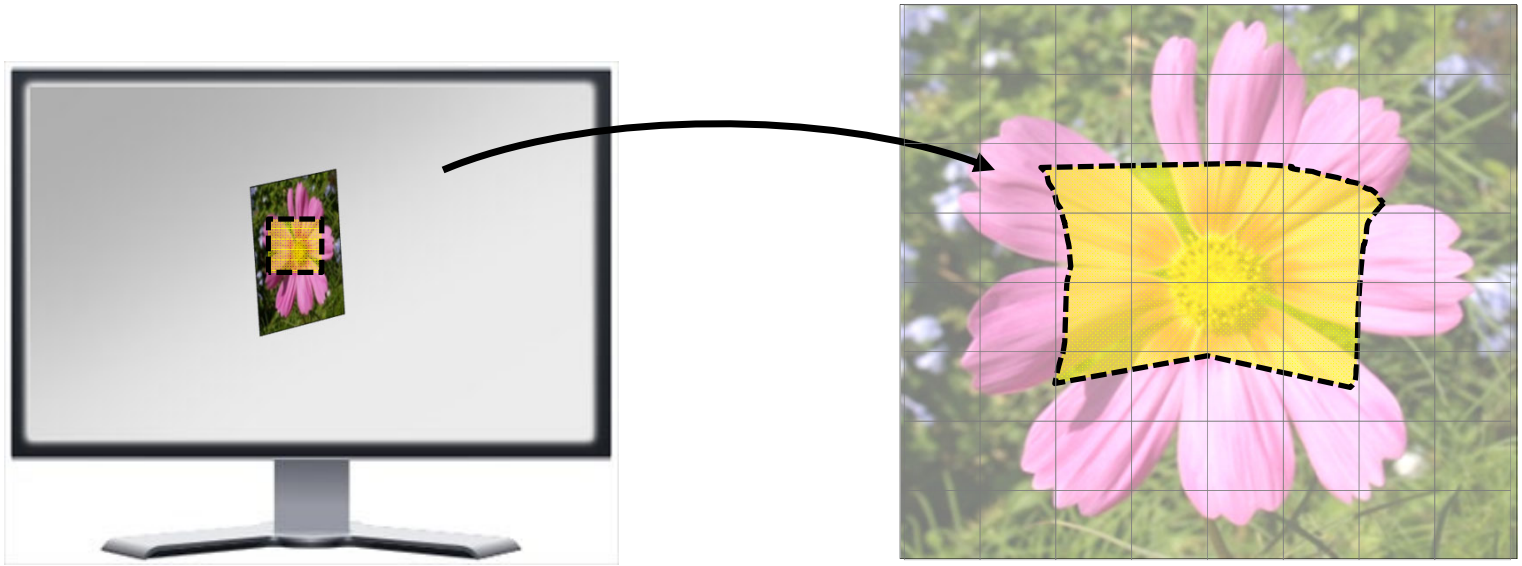
Ideal: color d’un pixel → color de la seva *preimatge* a la textura

Magnification



Magnification → la preimatge és < texel

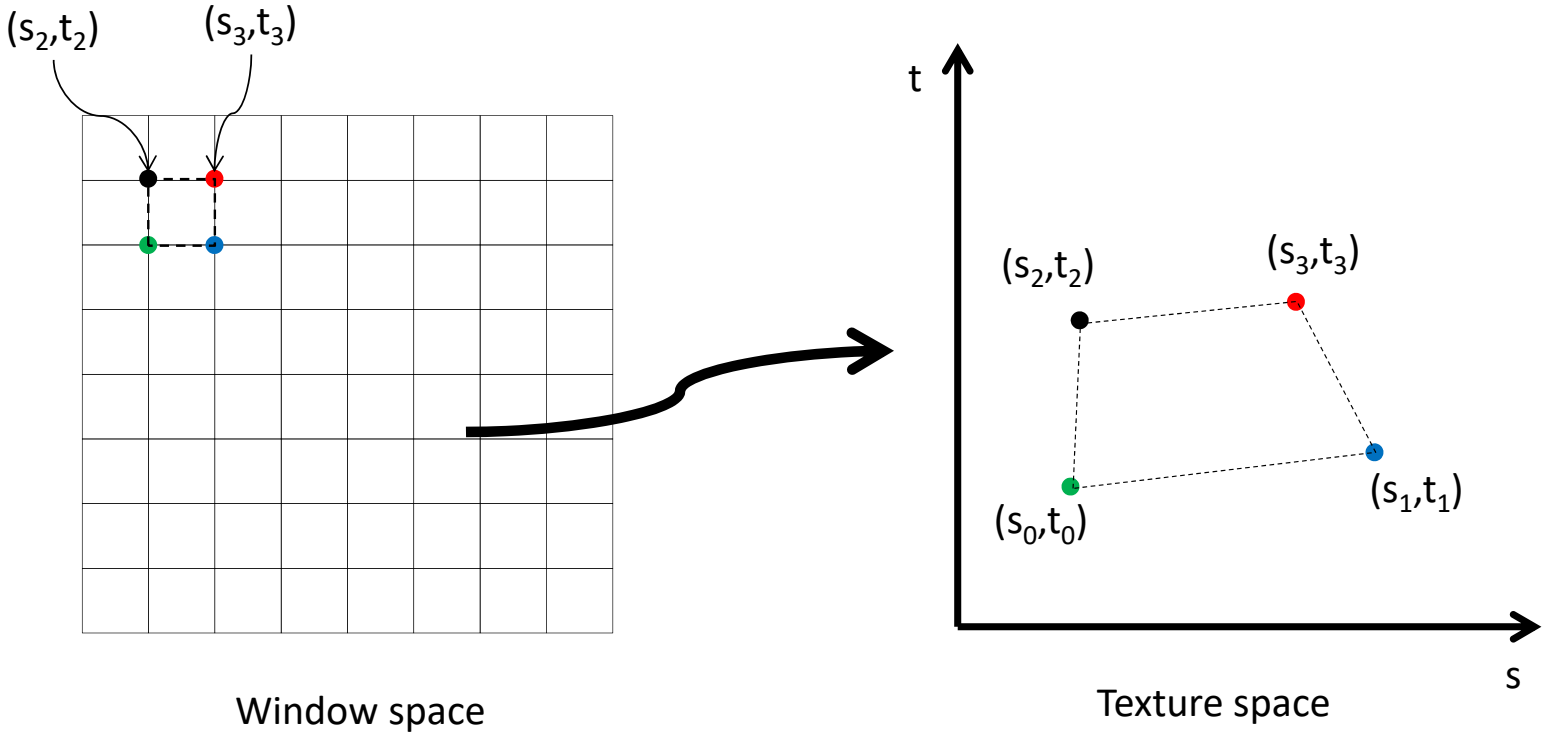
Minification



Minification → la preimatge és > texel

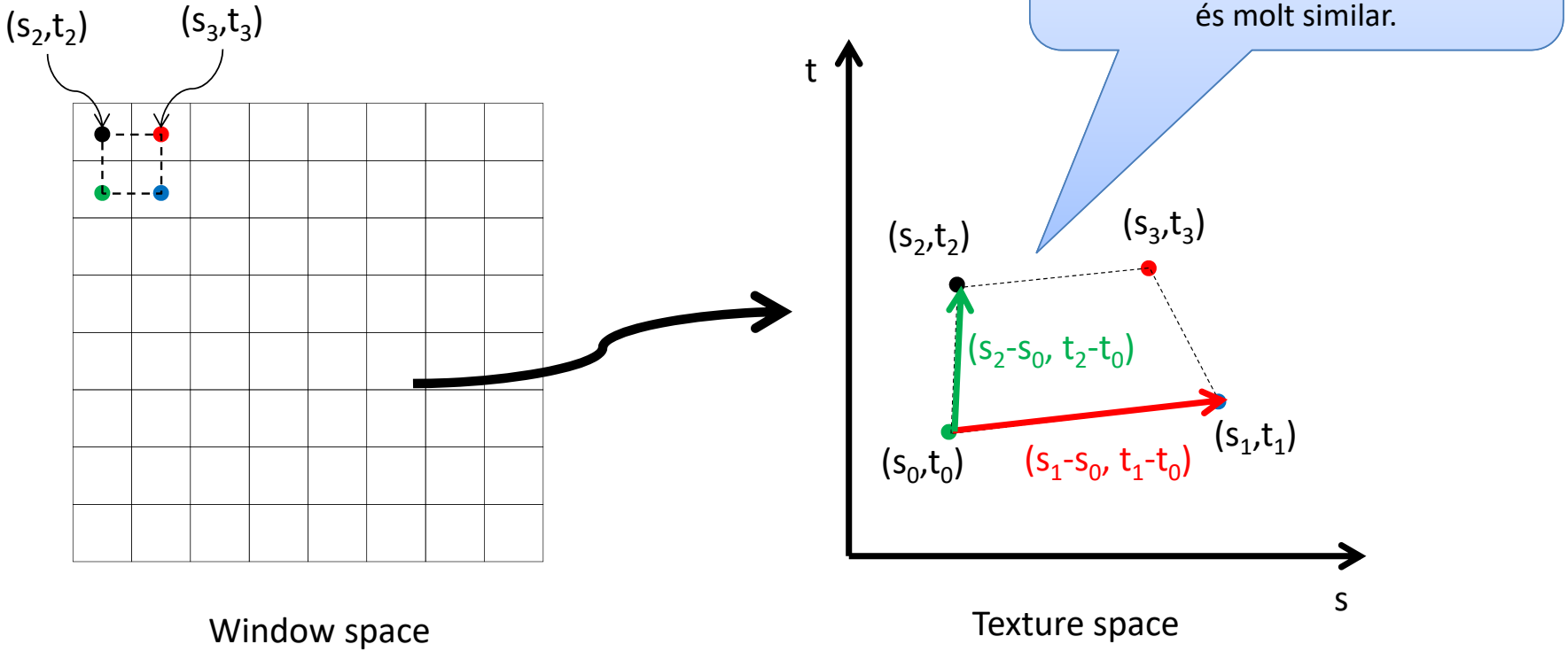
MAGNIFICATION O MINIFICATION?

Idealment



Aproximació que fa OpenGL

Usem les coords (s,t) del centre de cada fragment



Aproximació que fa OpenGL

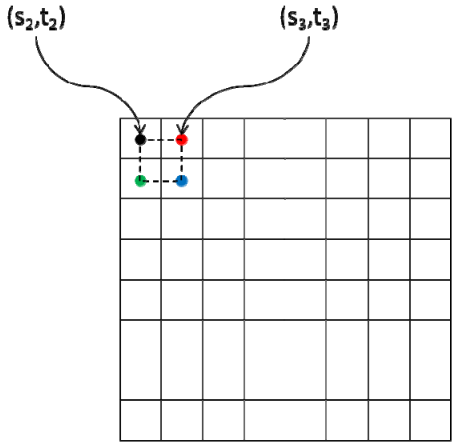
- Siguin **s**(x,y) , **t**(x,y) les coordenades s,t del fragment (x,y)
- Derivades parcials de **s**(x,y):

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

- En GLSL es poden calcular amb dFdx, dFdy:

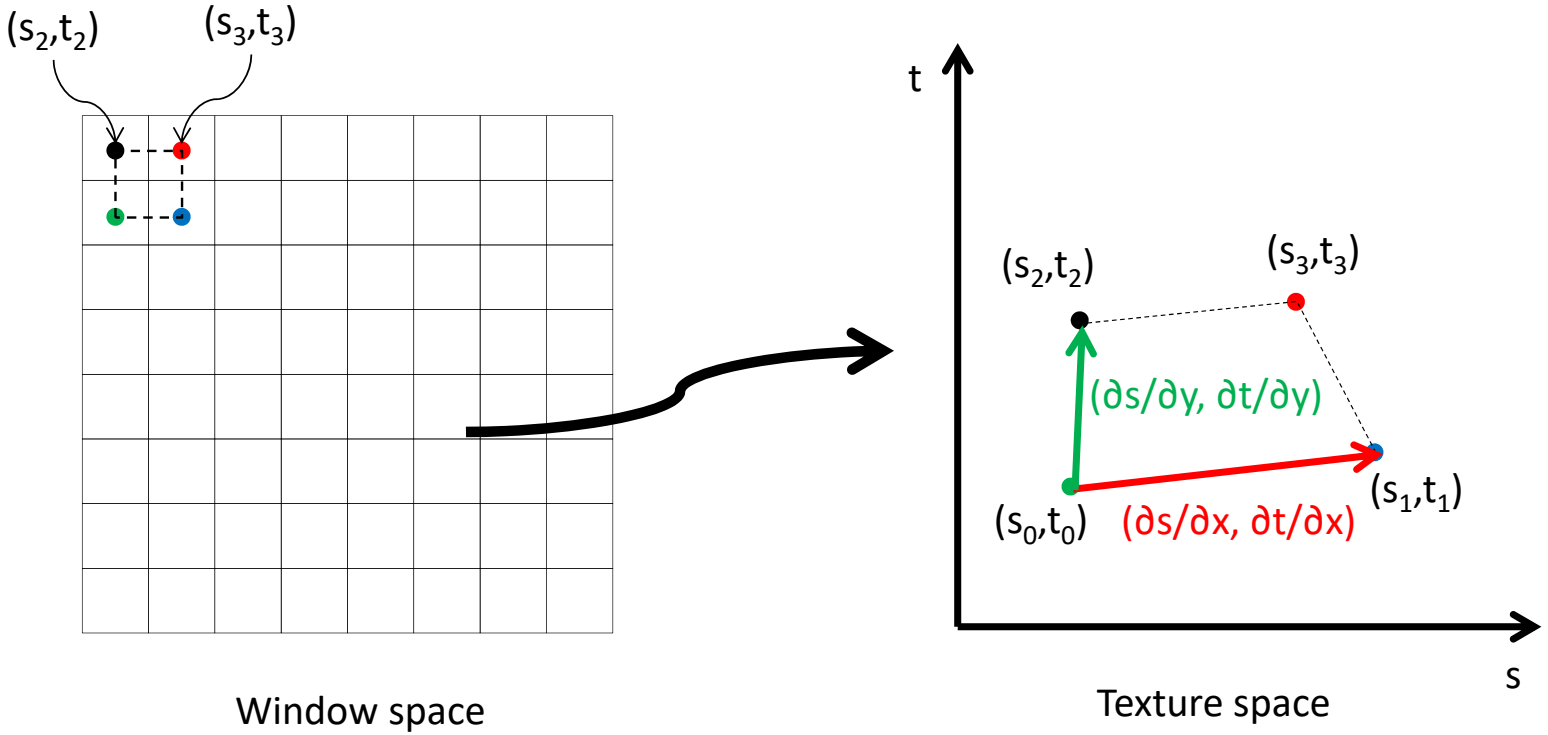
$$\partial s/\partial x \approx dFdx(texCoord.s)$$
$$\partial s/\partial y \approx dFdy(texCoord.s)$$

(anàlogament per t)



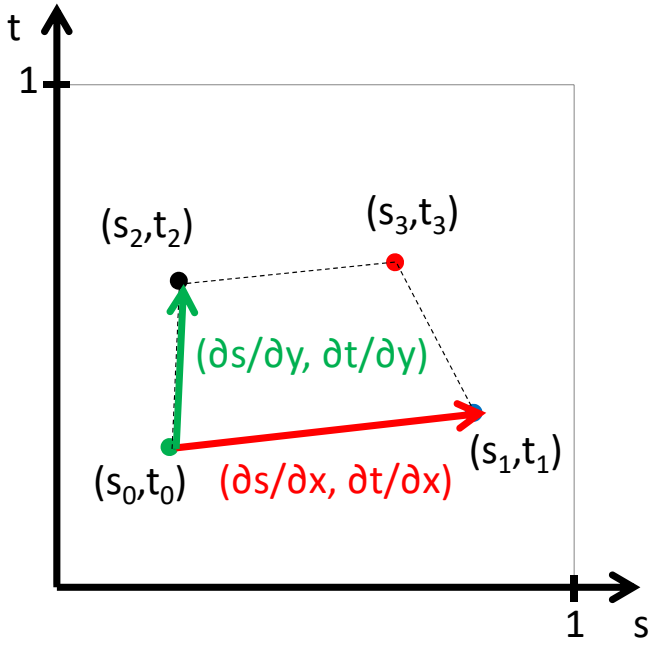
Aproximació que fa OpenGL

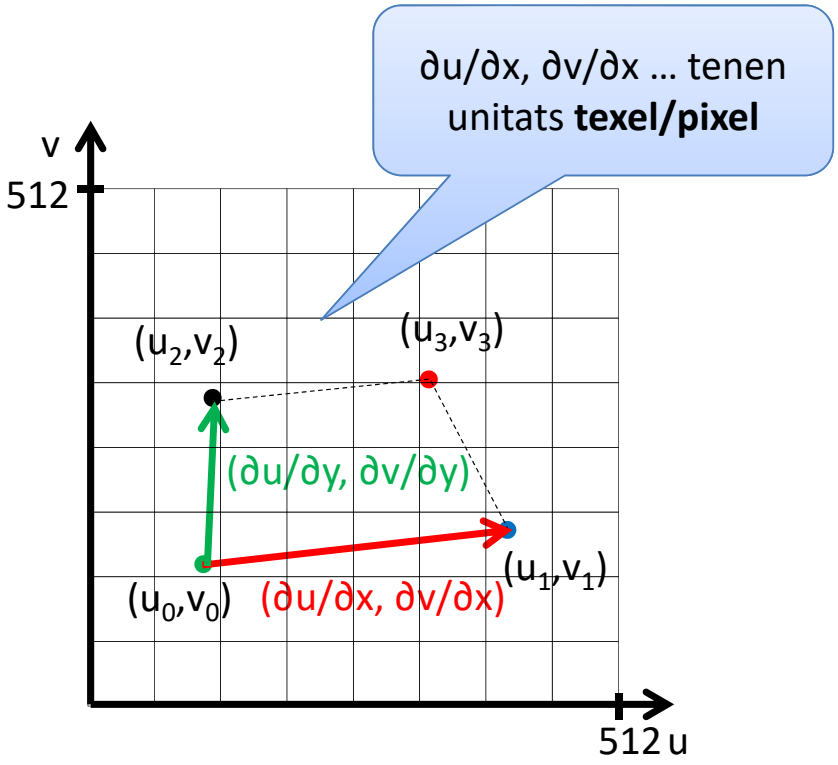
Aproximació de la mida de la pre-imatge (**en espai normalitzat de textura**)



Aproximació que fa OpenGL

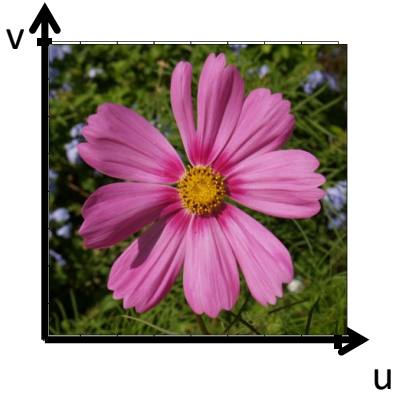
Mida de la pre-imatge (**en texels**) amb una textura 512x512





Exemple 1 (mapping 1:1)

Polígon projectat en WxH pixels



Textura WxH texels

En aquest cas un pixel correspon a un texel:

$\partial u/\partial x = \partial v/\partial y = 1 \quad \partial v/\partial x = \partial u/\partial y = 0$

Exemple 2 (magnification x2)

Polígon projectat en 2Wx2H pixels

Textura WxH texels

En aquest cas un pixel correspon a mig texel:

$\partial u/\partial x = \partial v/\partial y = ½ \quad \partial v/\partial x = \partial u/\partial y = 0$

Exemple 3 (minification x2)

Fragments veïns tenen coordenades (u,v) més separades

Textura WxH texels

Polígon projectat en ½ W x ½ H pixels

En aquest cas un pixel correspon a dos texels:

$\partial u/\partial x = \partial v/\partial y = 2 \quad \partial v/\partial x = \partial u/\partial y = 0$

Exemple 4 (anisotròpic)

Textura WxH texels

En direcció horitzontal → magnification
En direcció vertical → minification

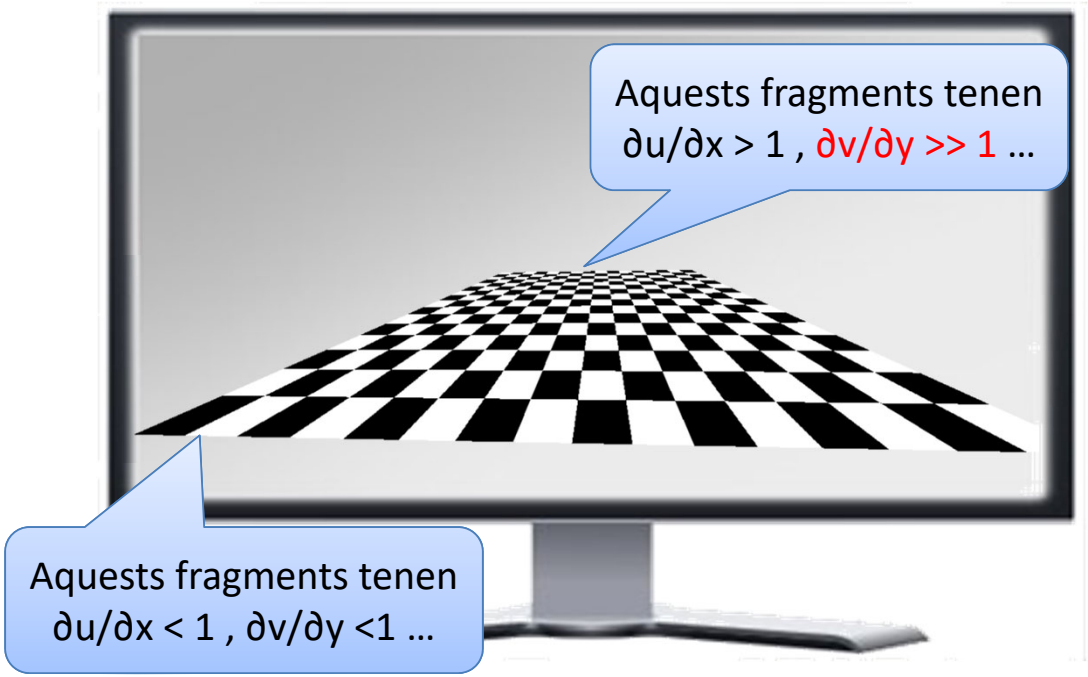
Exemple 4 (anisotròpic)

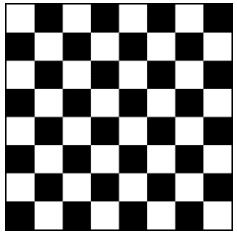
Textura WxH texels

En direcció horitzontal → magnification
En direcció vertical → minification

$\partial u/\partial x = ½ \;\; \partial v/\partial x = 0$
$\partial u/\partial y = 0 \;\; \partial v/\partial y = 2$

Exemple 5



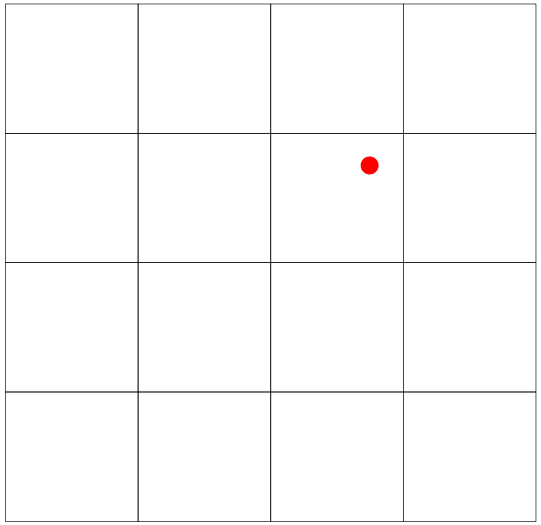


Textura

TEXTURE FILTERS

Texture filters

Determinen com s'avalua **texture**(sampler, texCoord)



Textura WxH texels

MAGNIFICATION

Magnification filters

- GL_NEARE

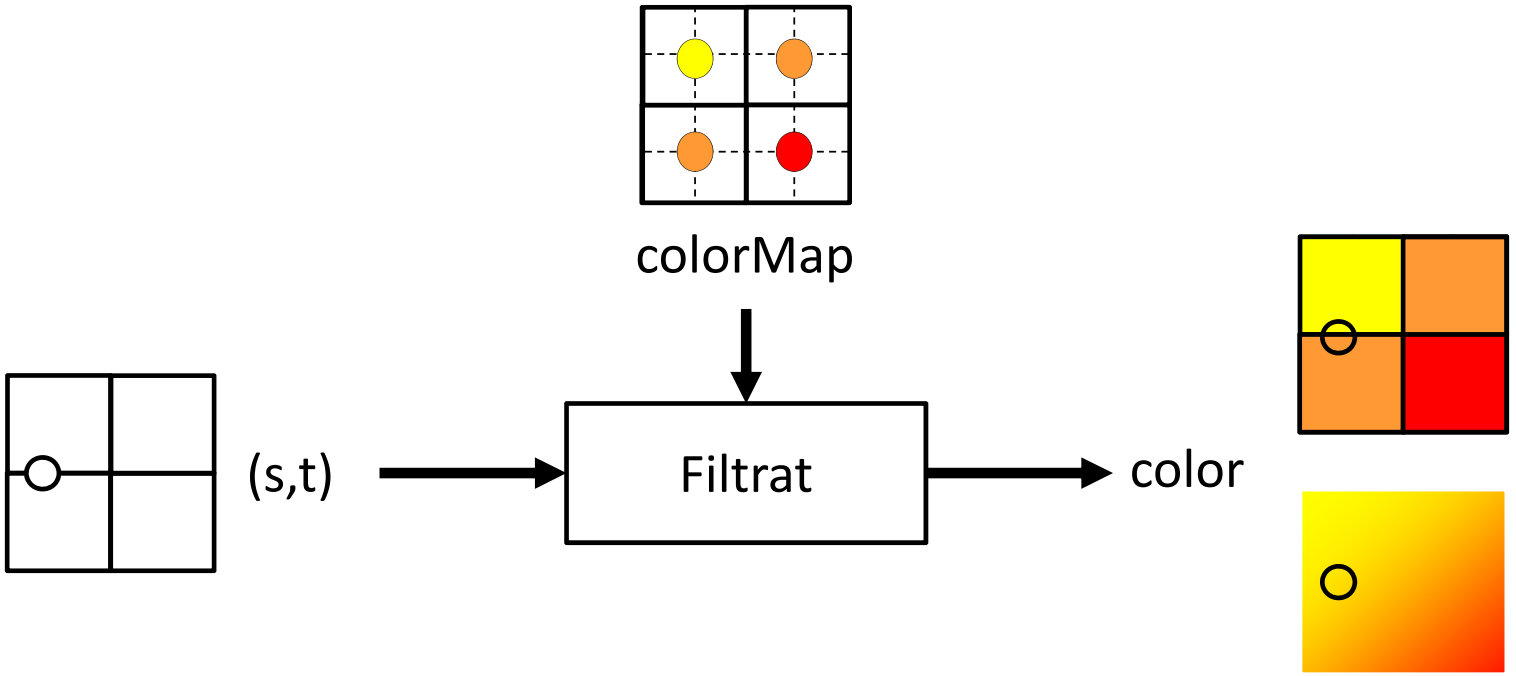
glTexParameterf(G

- GL_LINEAR

glTexParameterf(G

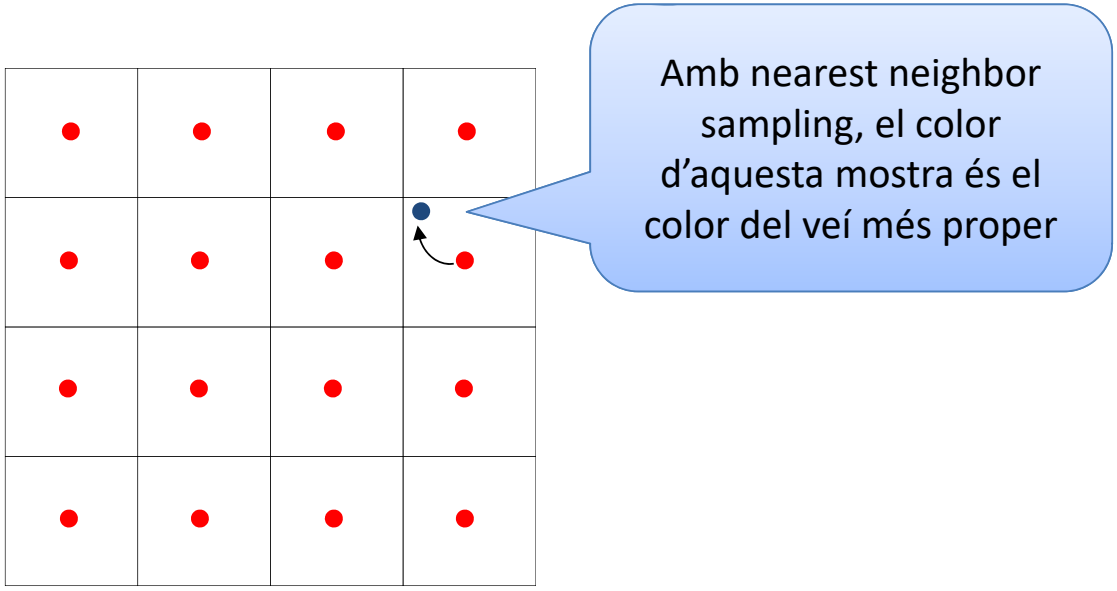
```
// Load Texture (once)
QImage img0("fieldstone.png");
QImage T = img0.convertToFormat(QImage::Format_ARGB32);
glGenTextures( 1, &textureId0);
glBindTexture(GL_TEXTURE_2D, textureId0);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, T.width(), T.height(), 0,
                    GL_RGBA, GL_UNSIGNED_BYTE, T.bits());
glTexParameterf()...
// Bind textures, set uniforms...
g.glActiveTexture(GL_TEXTURE0);
g.glBindTexture(GL_TEXTURE_2D, textureId0);
program->bind();
program->setUniformValue("colorMap", 0);

...
```

Magnification filters



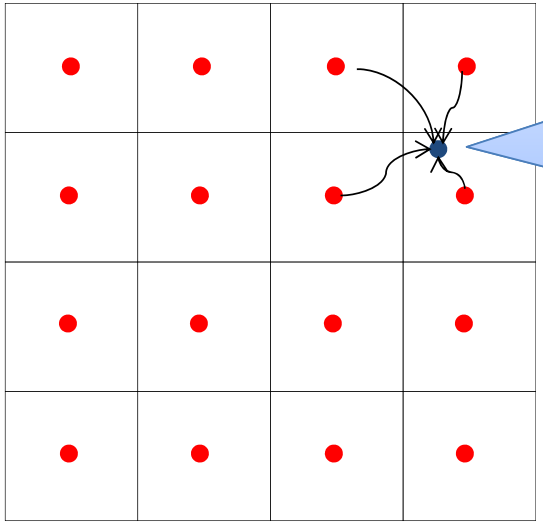
Magnification filters

GL_NEAREST: nearest neighbor sampling



Magnification filters

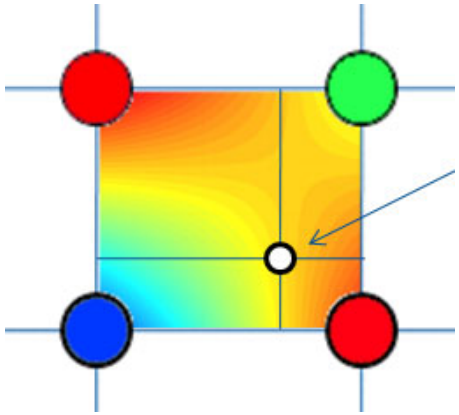
GL_LINEAR: bilinear interpolation

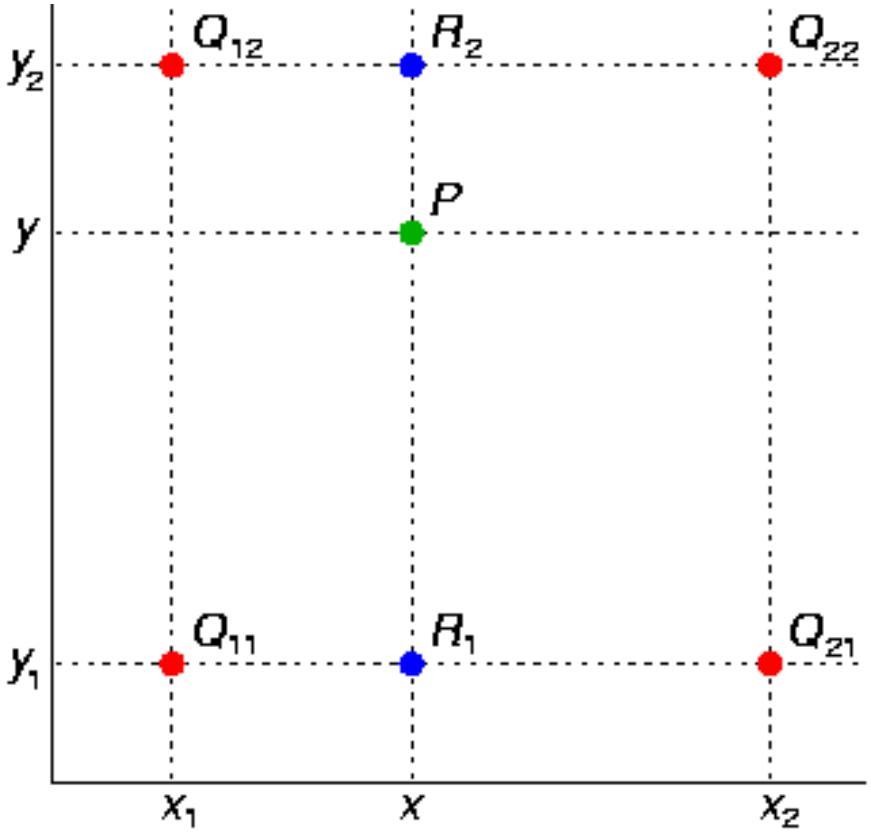


Amb interpolació bilineal, el color d'aquesta mostra és una mitjana ponderada dels colors dels quatre veïns més propers

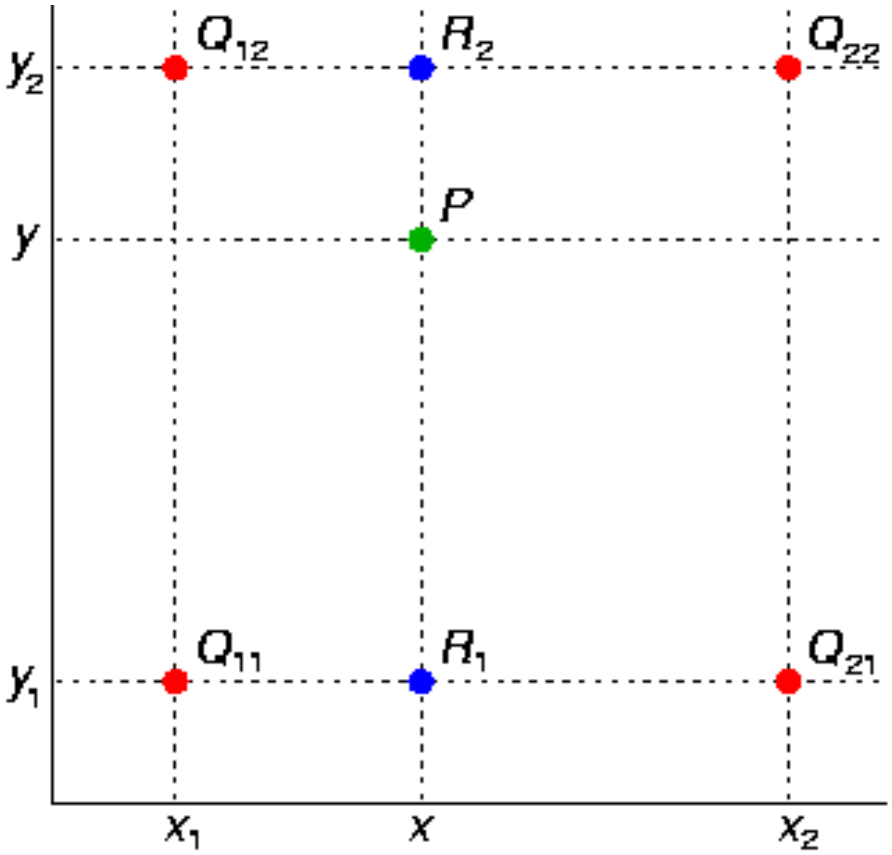
Comparació

Bilinear interpolation

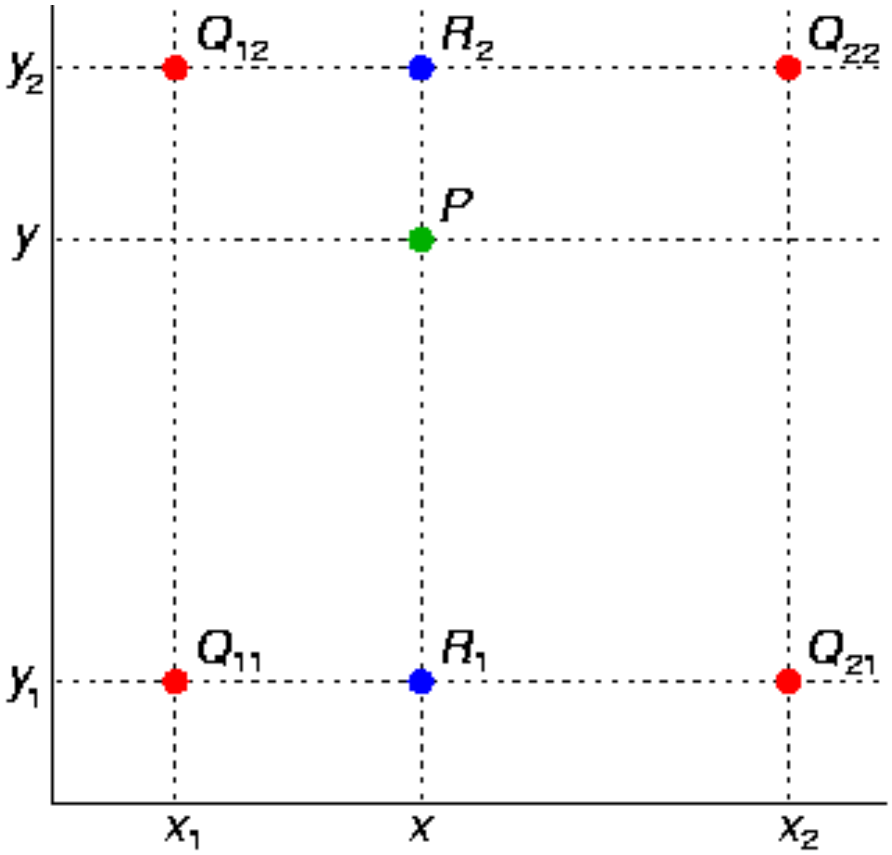




Bilinear interpolation



Bilinear interpolation



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

MINIFICATION

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!

Minification x16

Minification x8

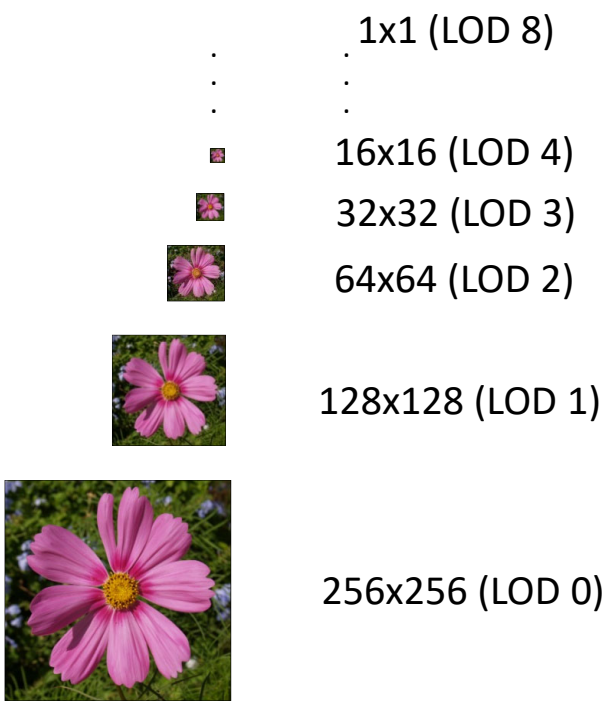
Minification x4

Minification x2

Textura

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

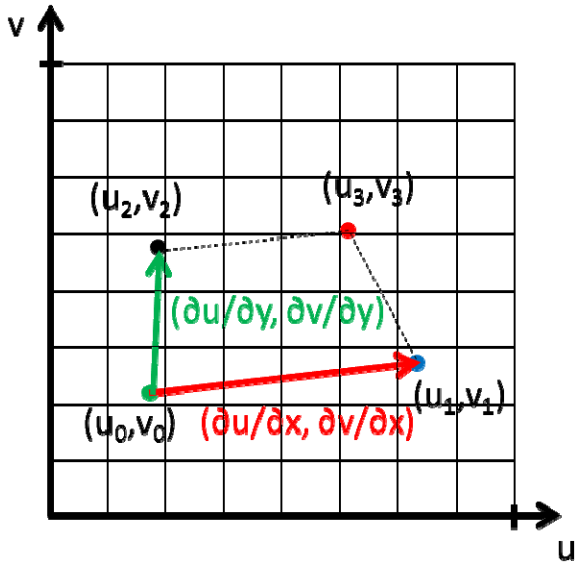
	Minification	**∂u/∂x, ∂v/∂y**	**LOD**
	1x	1	0
	2x	2	1
	4x	4	2
	8x	8	3
	2^{λ} x	2^{λ}	λ

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

Mipmapping

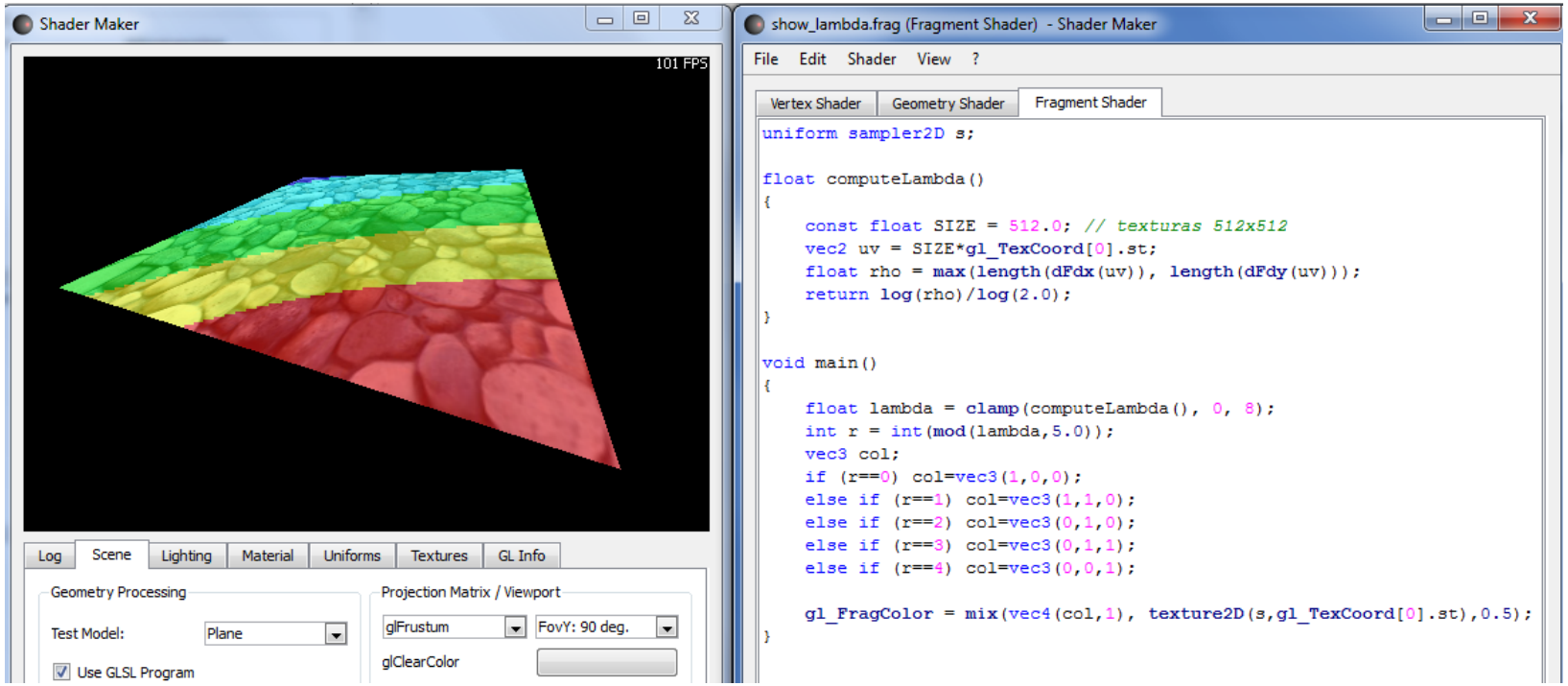
En general:

- Es calcula un valor rho=f(∂u/∂x, ∂v/∂x, ∂u/∂y, ∂v/∂y)
- Es calcula el $\lambda=\log_2(\text{rho})$



Mipmapping

Demo: show_lambda.frag



Minification filters

Without mipmapping

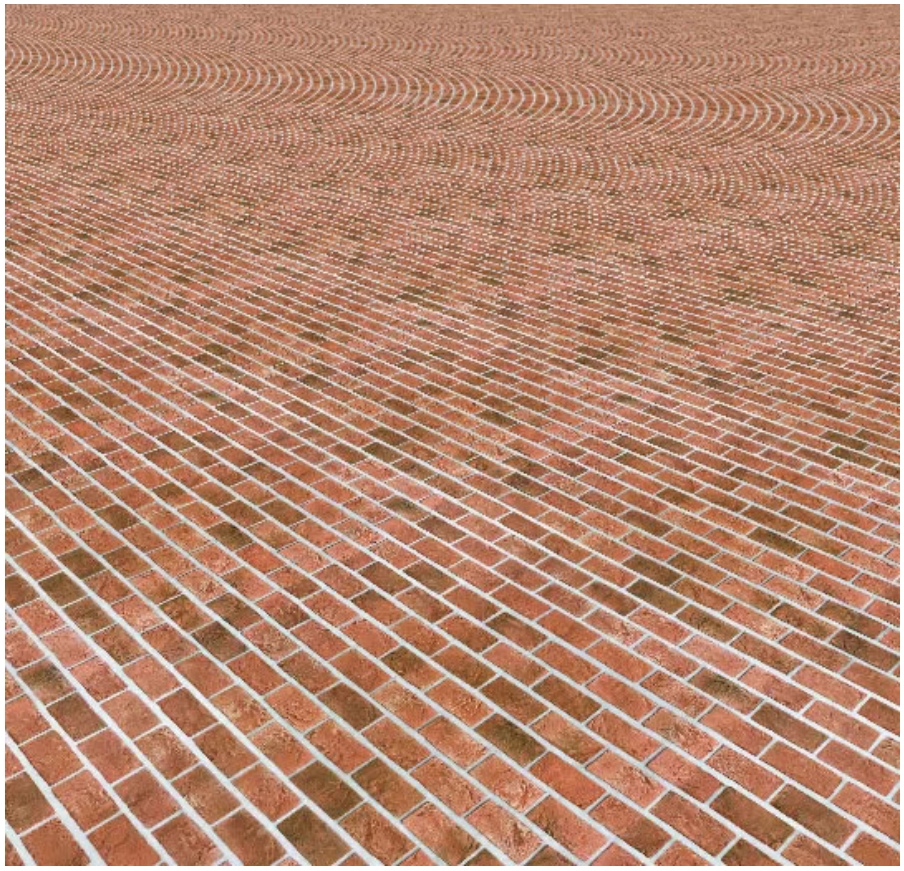
- GL_NEAREST // Nearest neighbor sampling on LOD 0
- GL_LINEAR // Bilinear interpolation on LOD 0

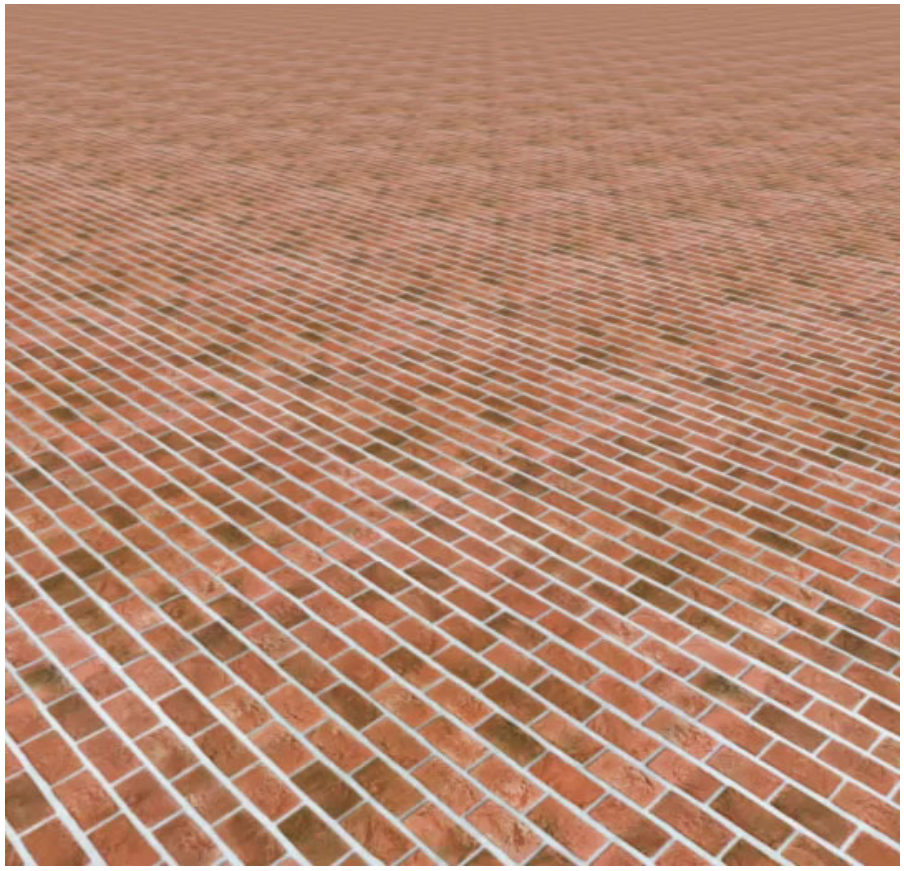
With mipmapping

- GL_NEAREST_MIPMAP_NEAREST // Nearest neighbor sampling on LOD int(λ)
- GL_LINEAR_MIPMAP_NEAREST // Bilinear sampling on LOD int(λ)
- GL_NEAREST_MIPMAP_LINEAR
  // c_0 = nearest neighbor on LOD int(λ)
  // c_1 = nearest neighbor on LOD int(λ+1)
  // mix(c_0, c_1, fract(λ))
- GL_LINEAR_MIPMAP_LINEAR
  // c_0 = bilinear sampling on LOD int(λ)
  // c_1 = bilinear sampling on LOD int(λ+1)
  // mix(c_0, c_1, fract(λ))

GL_<samplingWithinTheLODs>_MIPMAP_<oneOrTwoLODs>

Comparació

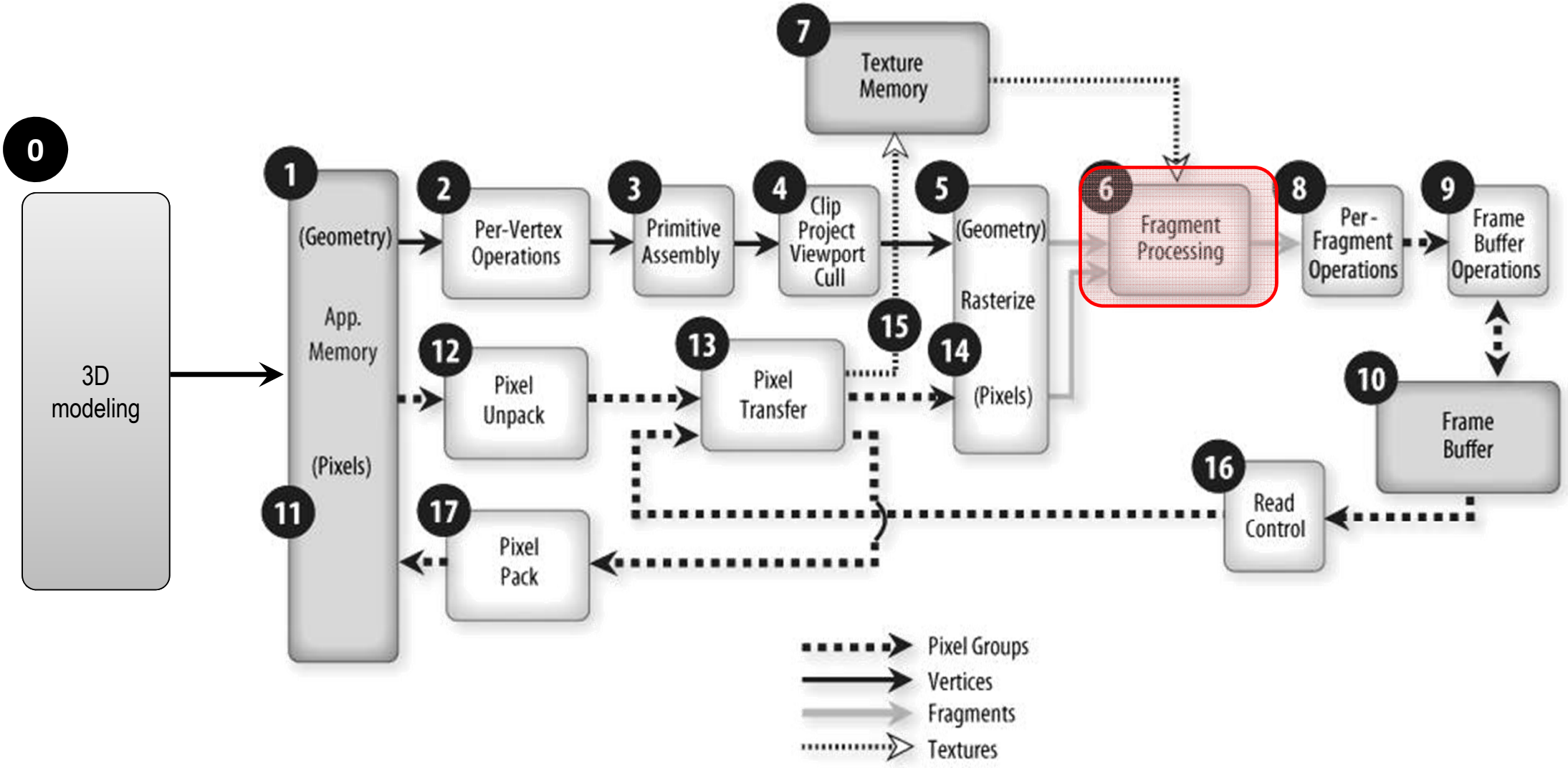




Minification filters (2)

[Demo mipmapping]

COMBINACIÓ


0
3D modeling
1
(Geometry)
App. Memory
(Pixels)
2
Per-Vertex Operations
3
Primitive Assembly
4
Clip Project Viewport Cull
5
(Geometry)
Rasterize
(Pixels)
6
Fragment Processing
7
Texture Memory
8
Per-Fragment Operations
9
Frame Buffer Operations
10
Frame Buffer
11
12
Pixel Unpack
13
Pixel Transfer
14
15
16
Read Control
17
Pixel Pack
Pixel Groups
Vertices
Fragments
Textures

FS – exemple

uniform sampler2D colorMap;
in vec2 vtexcoord;
in vec4 frontColor;

vec4 texColor = **texture**(colorMap, vtexcoord);

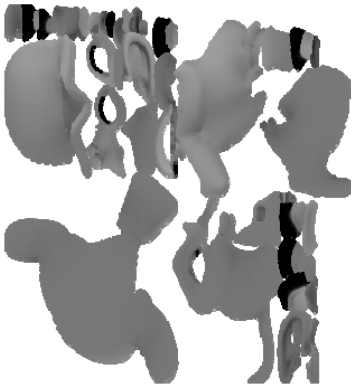
...

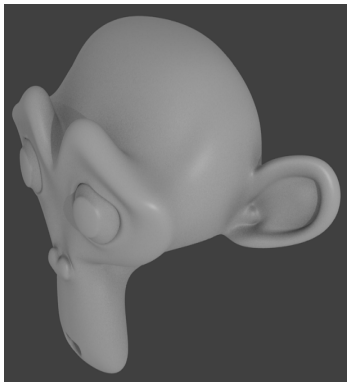
fragCoord = ???

Modes habituals

REPLACE: fragColor = texColor;

$K_d I_d (N \cdot L) + K_s I_s (R \cdot V)^s$





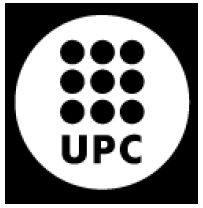
Modes habituals

MODULATE: fragColor = texColor * frontColor;

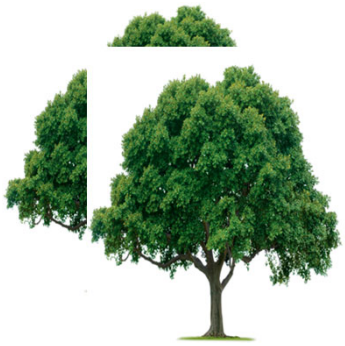
$$K_d I_d (N \cdot L) + K_s I_s (R \cdot V)^s$$

Modes habituals

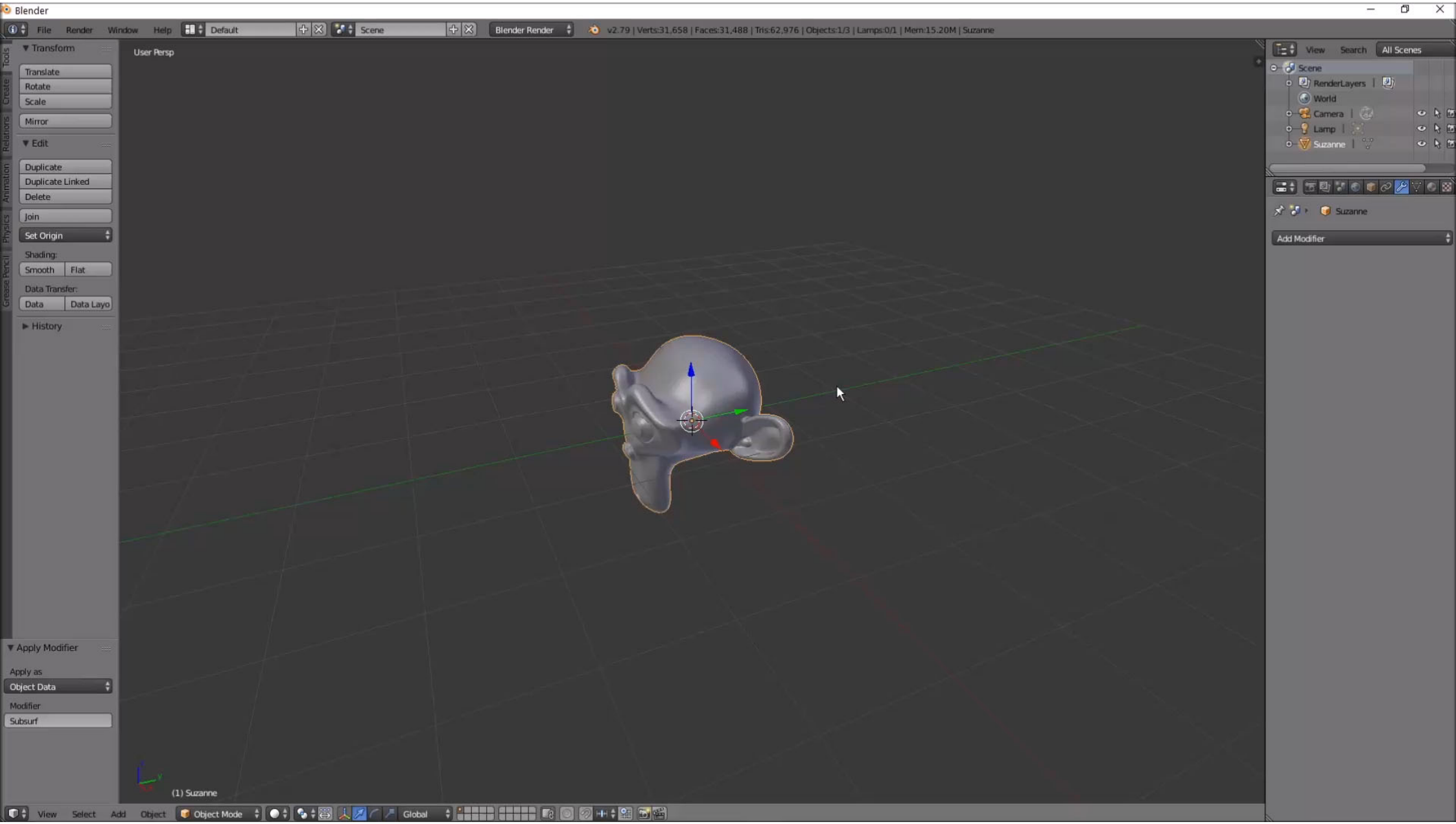
DECAL: fragColor = mix(frontColor, texColor, texColor.a);



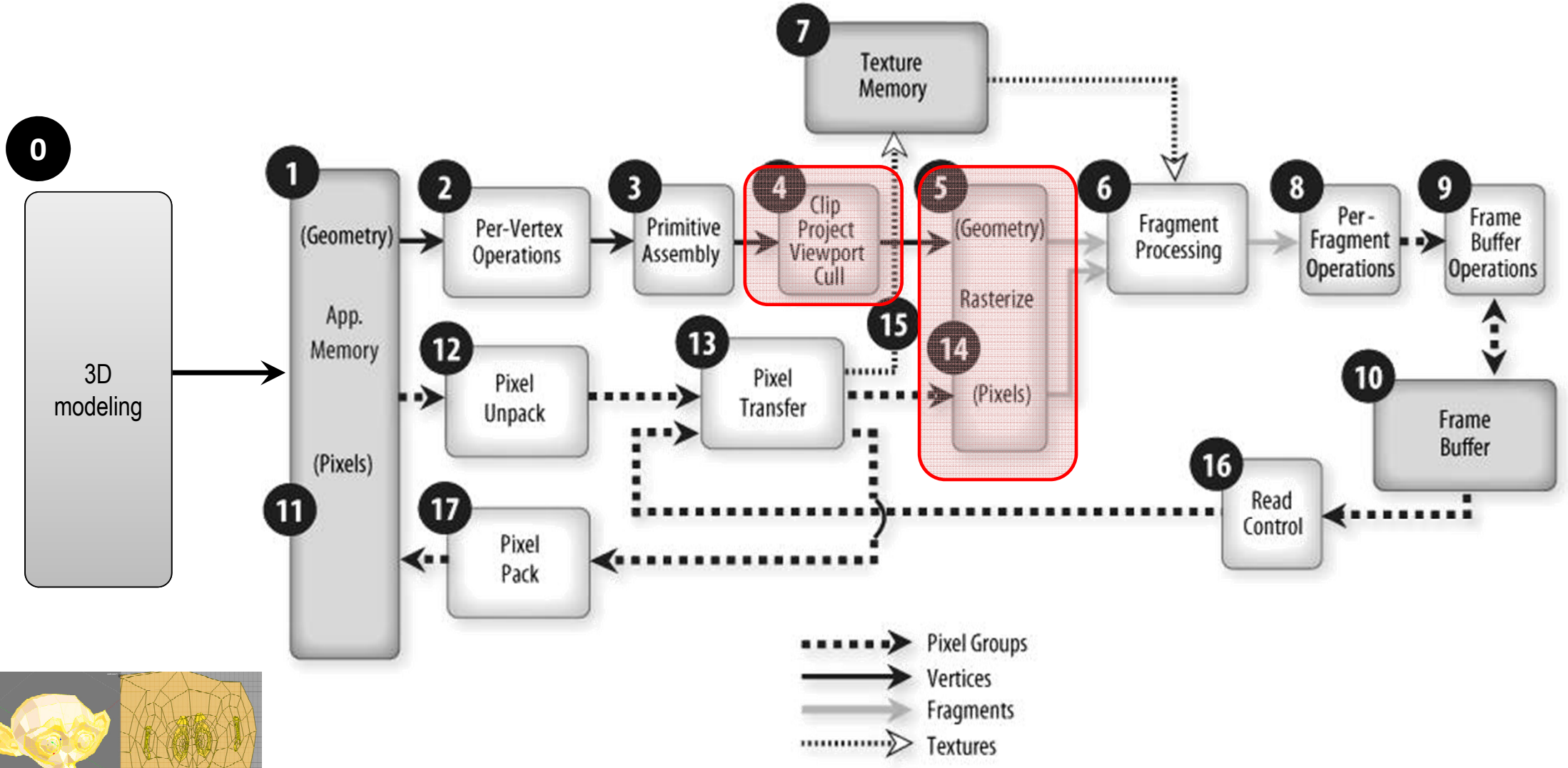

```
if (texColor.a < alphaThreshold)
    discard;
```



GENERACIÓ D’UN LIGHT MAP


Blender
File
Render
Window
Help
Default
Scene
Blender Render
v2.79 | Verts:31,658 | Faces:31,488 | Tris:62,976 | Objects:1/3 | Lamps:0/1 | Mem:15.20M | Suzanne
Transform
Translate
Rotate
Scale
Mirror
Edit
Duplicate
Duplicate Linked
Delete
Join
Set Origin
Shading:
Smooth
Flat
Data Transfer:
Data
Data Layo
History
Apply Modifier
Apply as
Object Data
Modifier
Subsurf
User Persp
(1) Suzanne
View
Select
Add
Object
Object Mode
Global
View
Search
All Scenes
Scene
RenderLayers
World
Camera
Lamp
Suzanne
Add Modifier

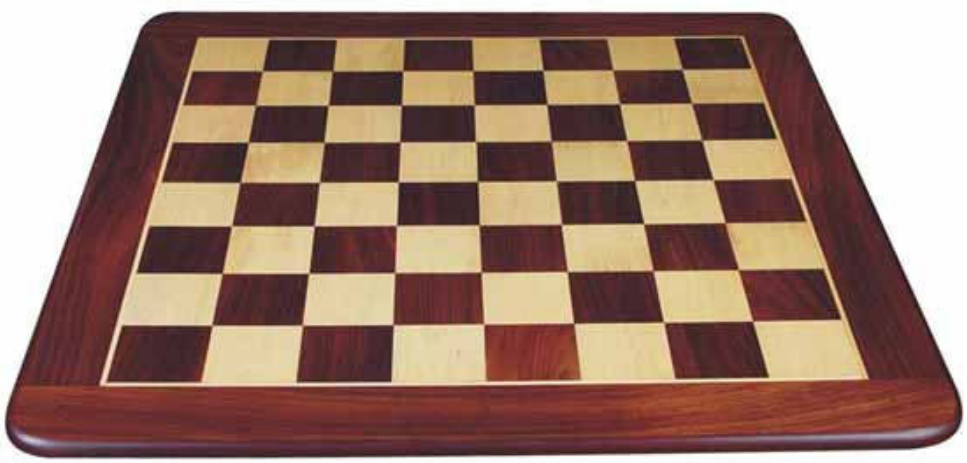
INTERPOLACIÓ DE COORDS DE TEXTURA


0
3D modeling
1
(Geometry)
App. Memory
(Pixels)
2
Per-Vertex Operations
3
Primitive Assembly
4
Clip Project Viewport Cull
5
(Geometry)
Rasterize
14
(Pixels)
6
Fragment Processing
7
Texture Memory
8
Per-Fragment Operations
9
Frame Buffer Operations
10
Frame Buffer
11
12
Pixel Unpack
13
Pixel Transfer
15
16
Read Control
17
Pixel Pack
Pixel Groups
Vertices
Fragments
Textures

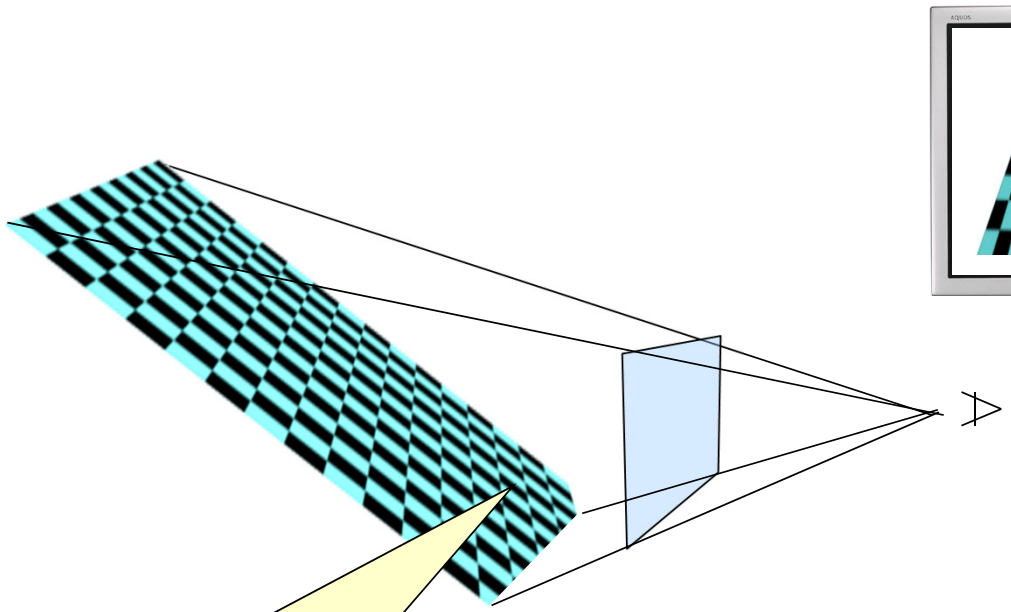
PERSPECTIVE-CORRECT INTERPOLATION

Perspective deformation

Perspective deformation



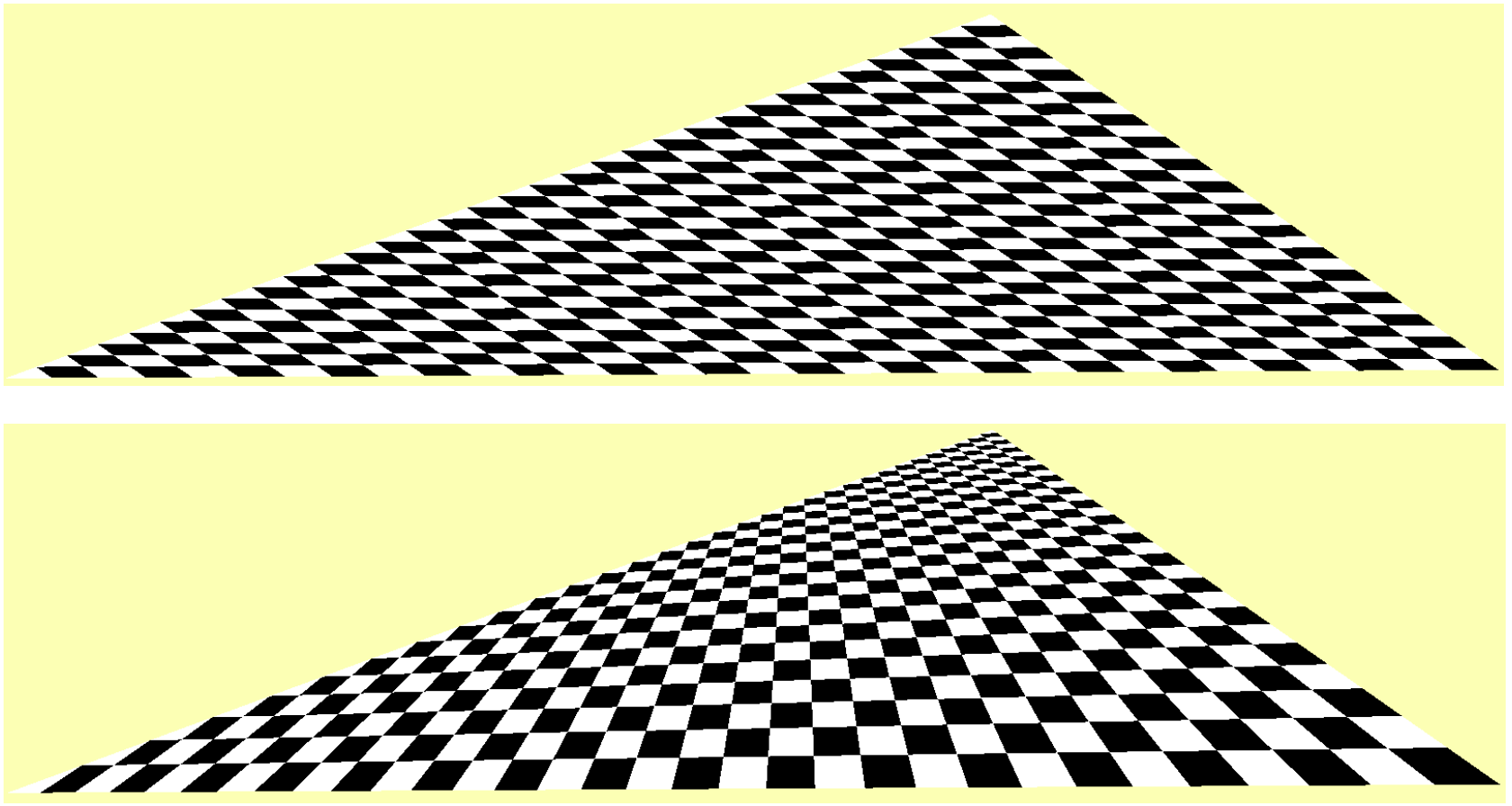
Perspective-correct interpolation



En **object space** les coords de textura varien linialment (totes les cel·les 4cm x 4cm)

En **window space** les coords de textura **no** varien linialment degut a la divisió de perspectiva

Perspective-correct interpolation



[Demo: pers.vert, pers.frag]

Perspective-correct interpolation

Solucions:

a) Interpolem (s,t) en **object space** (també valdria en **world, eye i clip space**, perquè són abans de la div. de perspectiva)

b) Interpolem (sw, tw, w) en **window space**, obtenint un texel **(s,t,q)**; accedirem a la textura amb **(s/q, t/q)**

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

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

Perspective-Correct Interpolation

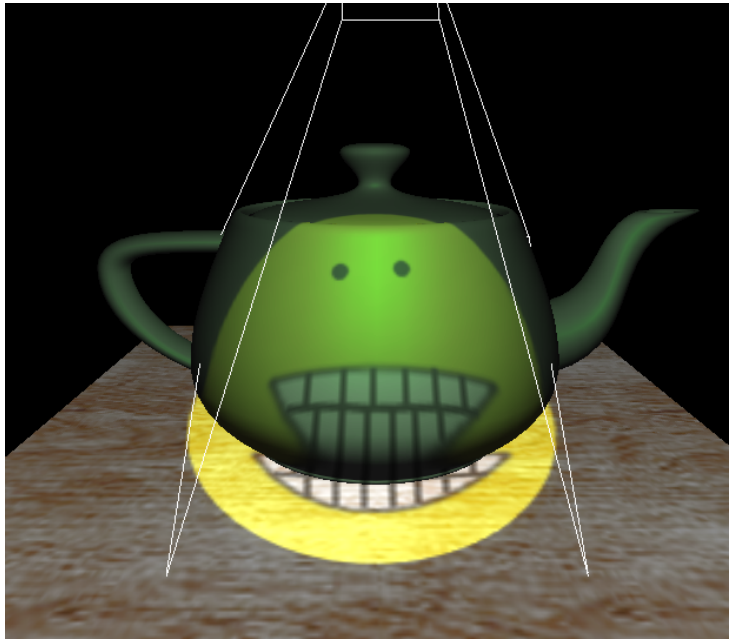
Kok-Lim Low

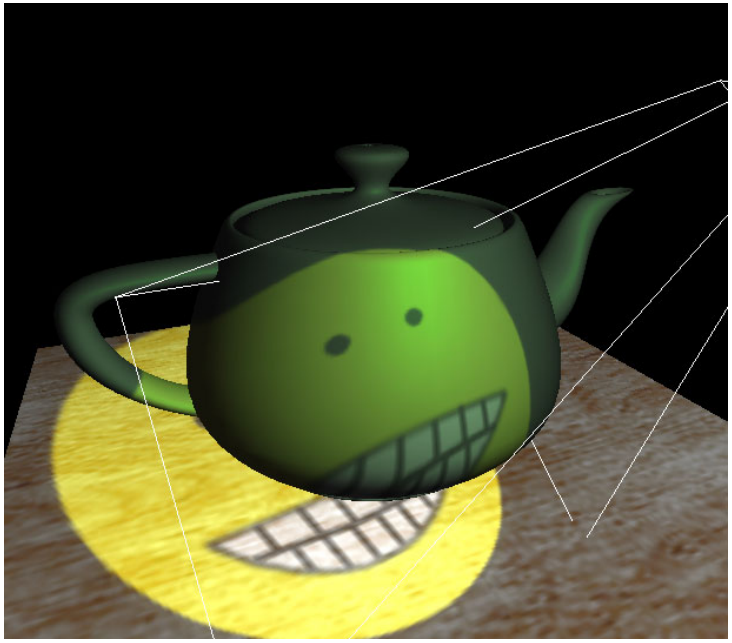
Department of Computer Science
University of North Carolina at Chapel Hill
Email: lowk@cs.unc.edu

March 12, 2002

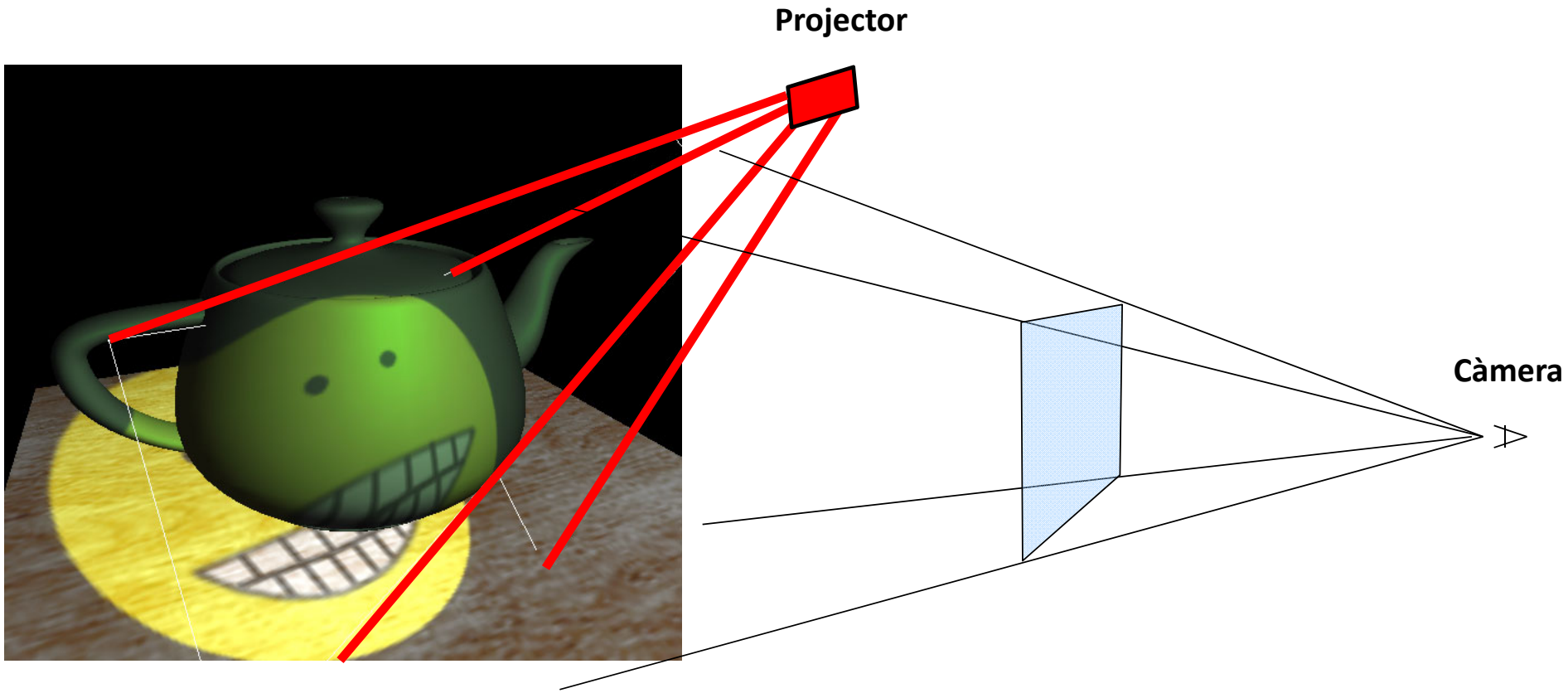
PROJECTIVE TEXTURE MAPPING

Projective texture mapping





Projective texture mapping



Opció 1 (incorrecta)

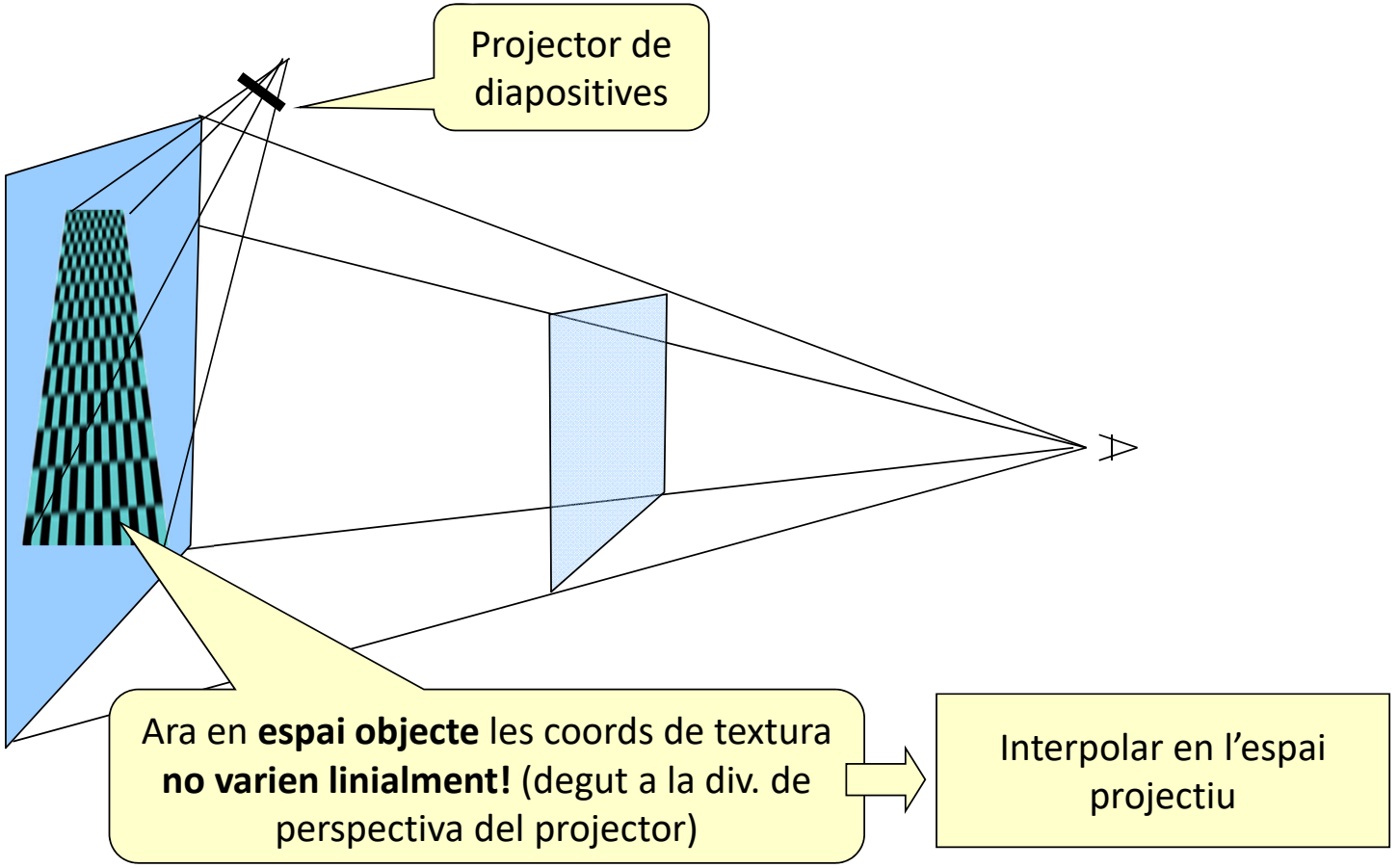
VS – generació coords de textura

Passa el vèrtex de *object space* a *window space* (viewport 1x1)
Calcula (s,t) com (x,y) del resultat

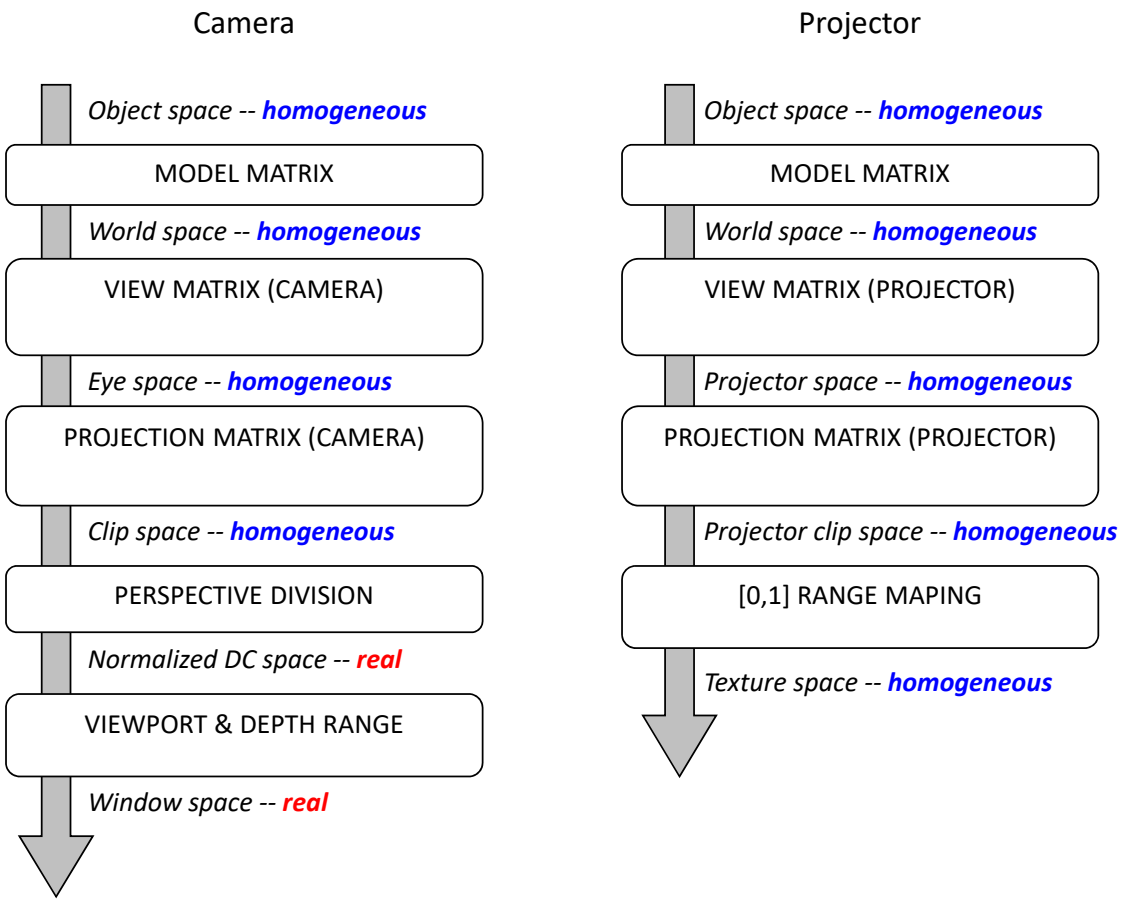
FS – accés a textura

Usa (s,t) per accedir a la textura

Projective-space interpolation



Projective texture mapping



Opció 2 (correcta)

VS – generació coords de textura

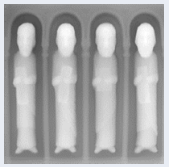
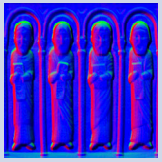
Passa el vèrtex de *object space* a *window space* (viewport 1x1)
però sense aplicar la div de perspectiva.
Calcula (s,t,p,q) com (x,y,z,w) del resultat

FS – accés a textura

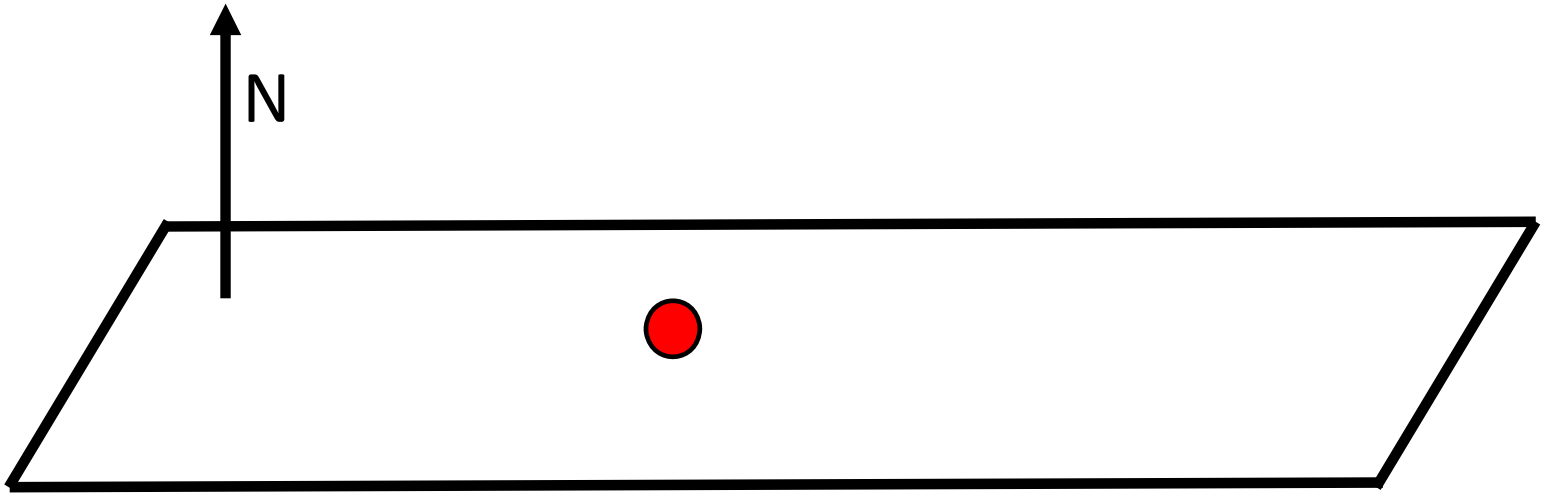
Usa (s/q, t/q) per a accedir a la textura

Color, bump, parallax, relief i displacement mapping

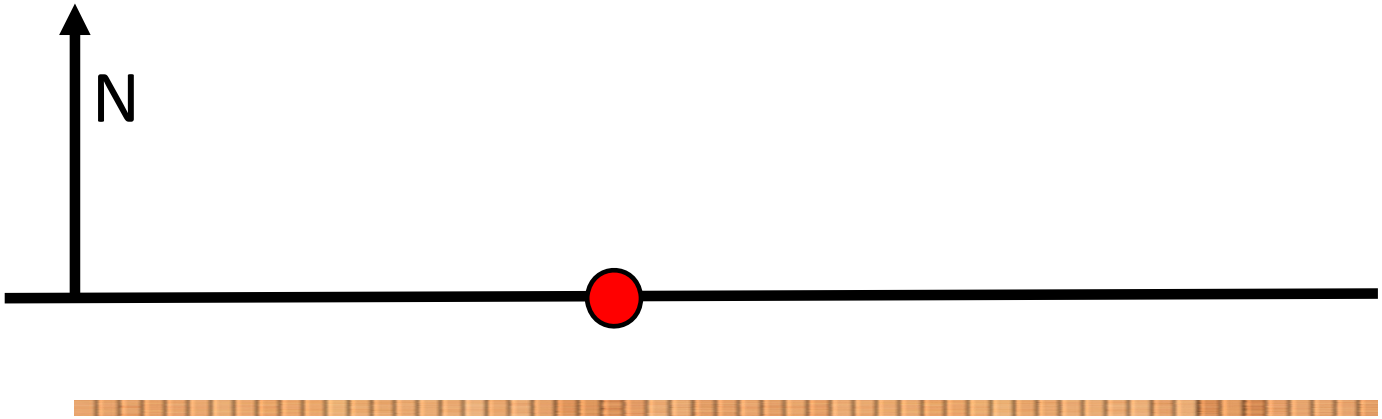
APLICACIONS

Tècnica	Info a la textura	Ús de la textura	Coords de textura	View parallax	Self-occlusion	Detailed silhouette	On s'aplica
Color mapping	RGB 3	Kd del material	(s,t)	-	-	-	FS
Bump mapping	D 1 	Modificar la normal	(s,t)	N	N	N	FS
Normal mapping	Normal 3 	Modificar la normal	(s,t)	N	N	N	FS
Parallax mapping	Normal + D 3+1 o 4	Modificar la normal	$(s+d_s, t+d_t)$	S	N	N	FS
Relief mapping	Normal + D 3+1 o 4	Modificar la normal; descartar fragments	$(s+d_s, t+d_t)$	S	S	S	FS!!!
Displacement mapping	D 1	Desplaçar els vèrtexs un cop subdividits els polígons.	(s,t)	S	S	S	CPU GS TCS+TES

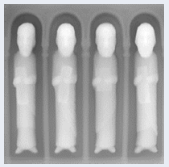
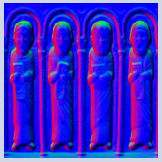
Color mapping



Color mapping

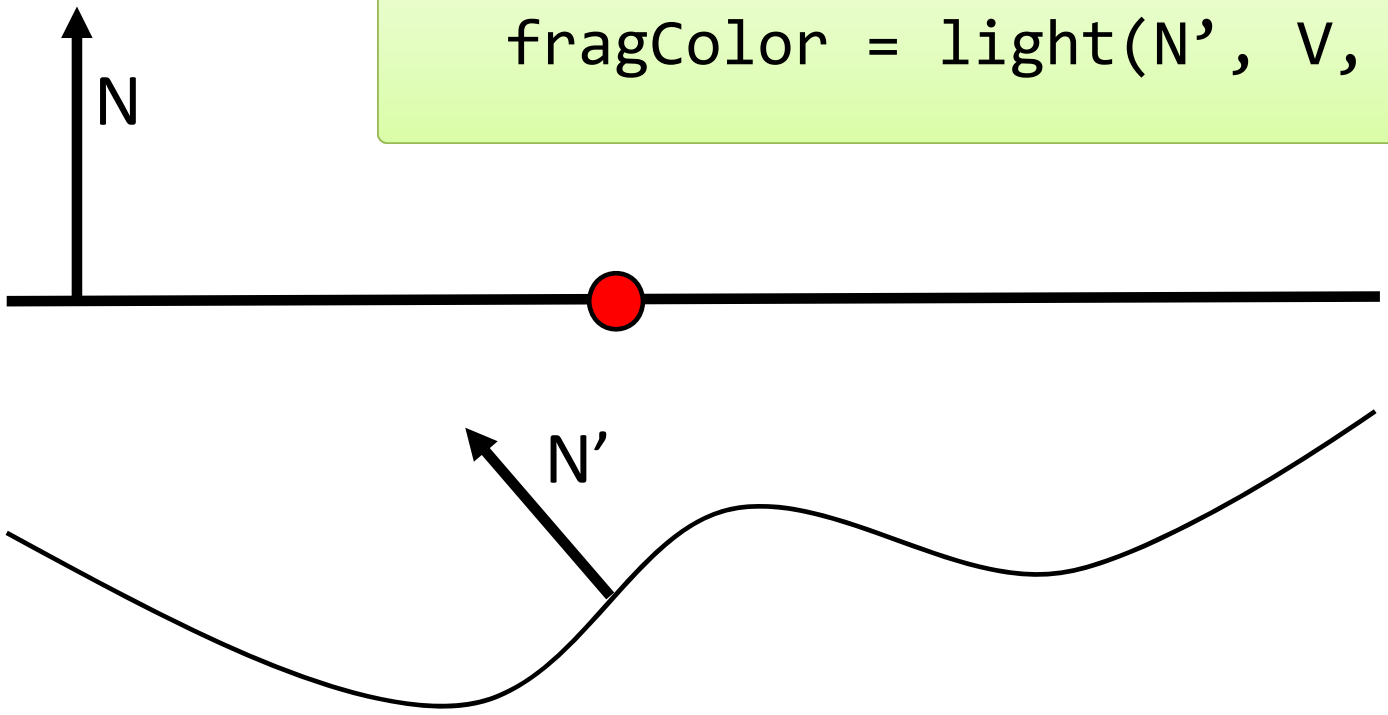


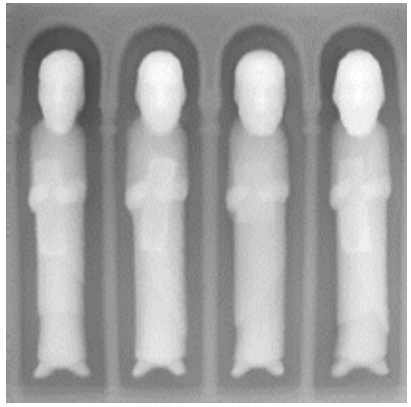
Color mapping

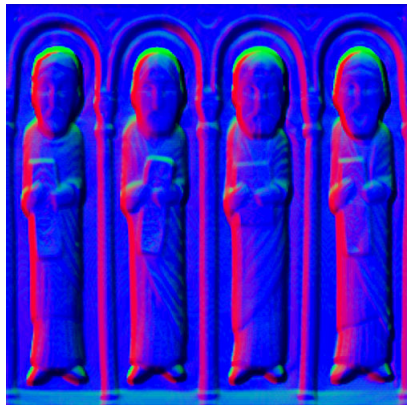
Tècnica	Info a la textura	Ús de la textura	Coords de textura	View parallax	Self-occlusion	Detailed silhouette	On s'aplica
Color mapping	RGB 3	Kd del material	(s,t)	-	-	-	FS
Bump mapping	D 1 	Modificar la normal	(s,t)	N	N	N	FS
Normal mapping	Normal 3 	Modificar la normal	(s,t)	N	N	N	FS
Parallax mapping	Normal + D 3+1 o 4	Modificar la normal	$(s+d_s, t+d_t)$	S	N	N	FS
Relief mapping	Normal + D 3+1 o 4	Modificar la normal; descartar fragments	$(s+d_s, t+d_t)$	S	S	S	FS!!!
Displacement mapping	D 1	Desplaçar els vèrtexs un cop subdividits els polígons.	(s,t)	S	S	S	CPU GS TCS+TES

Bump mapping/Normal mapping

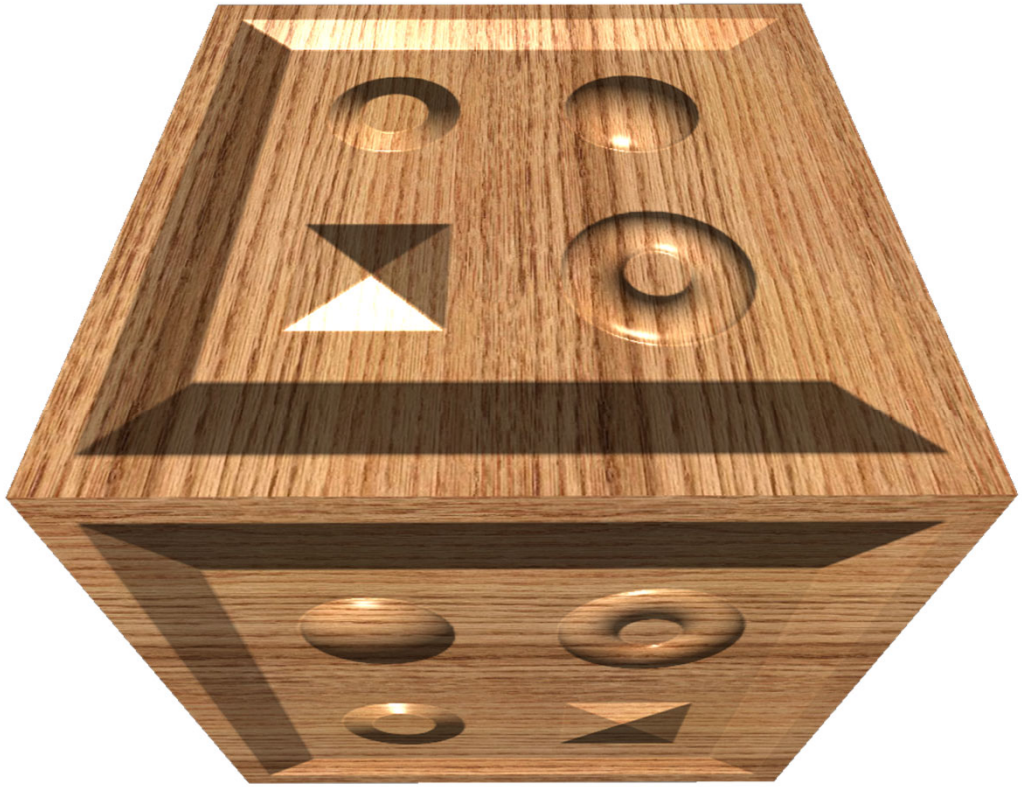
```
N’ = computeNormal(s,t);
fragColor = light(N’, V, L);
```

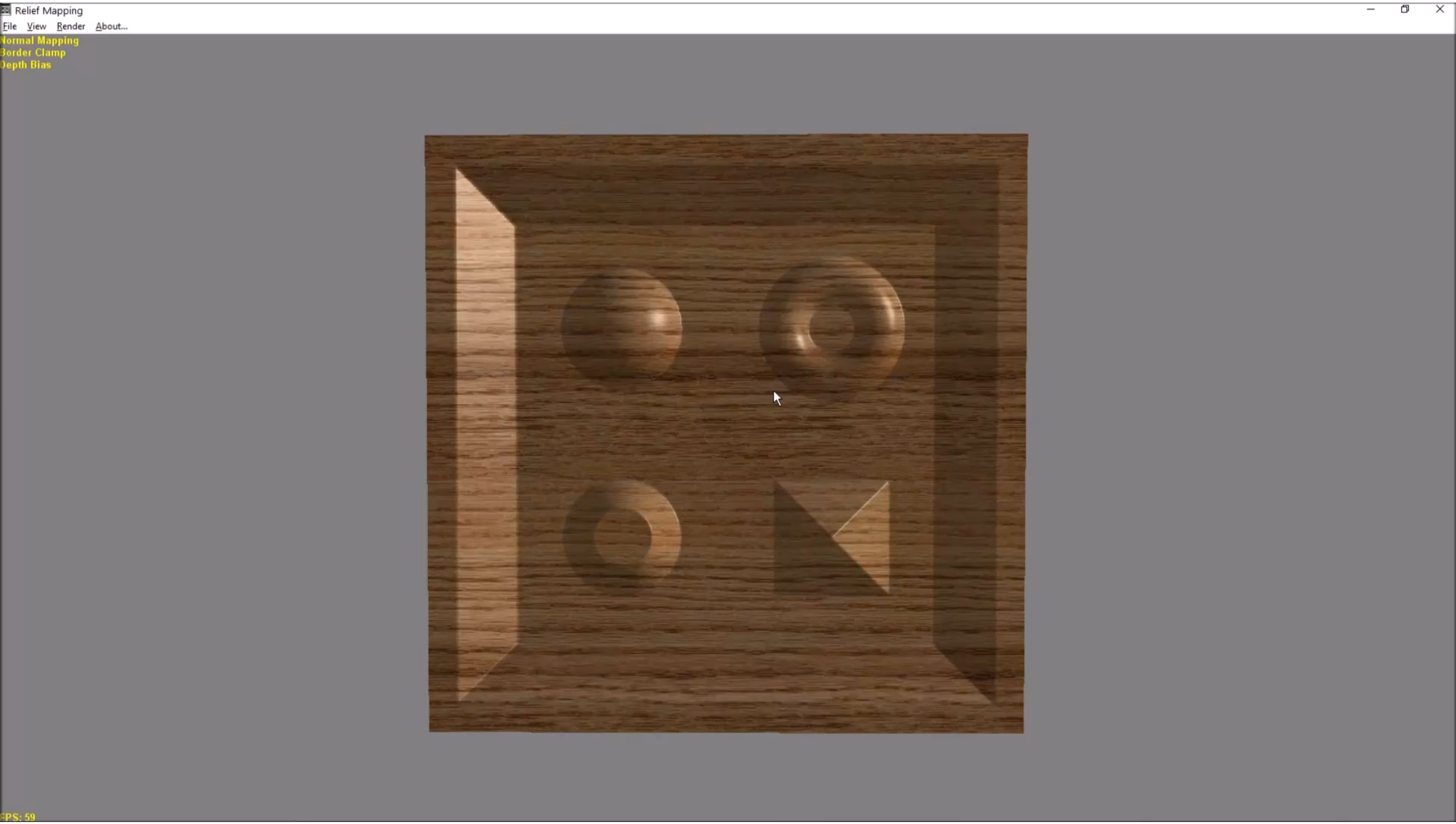




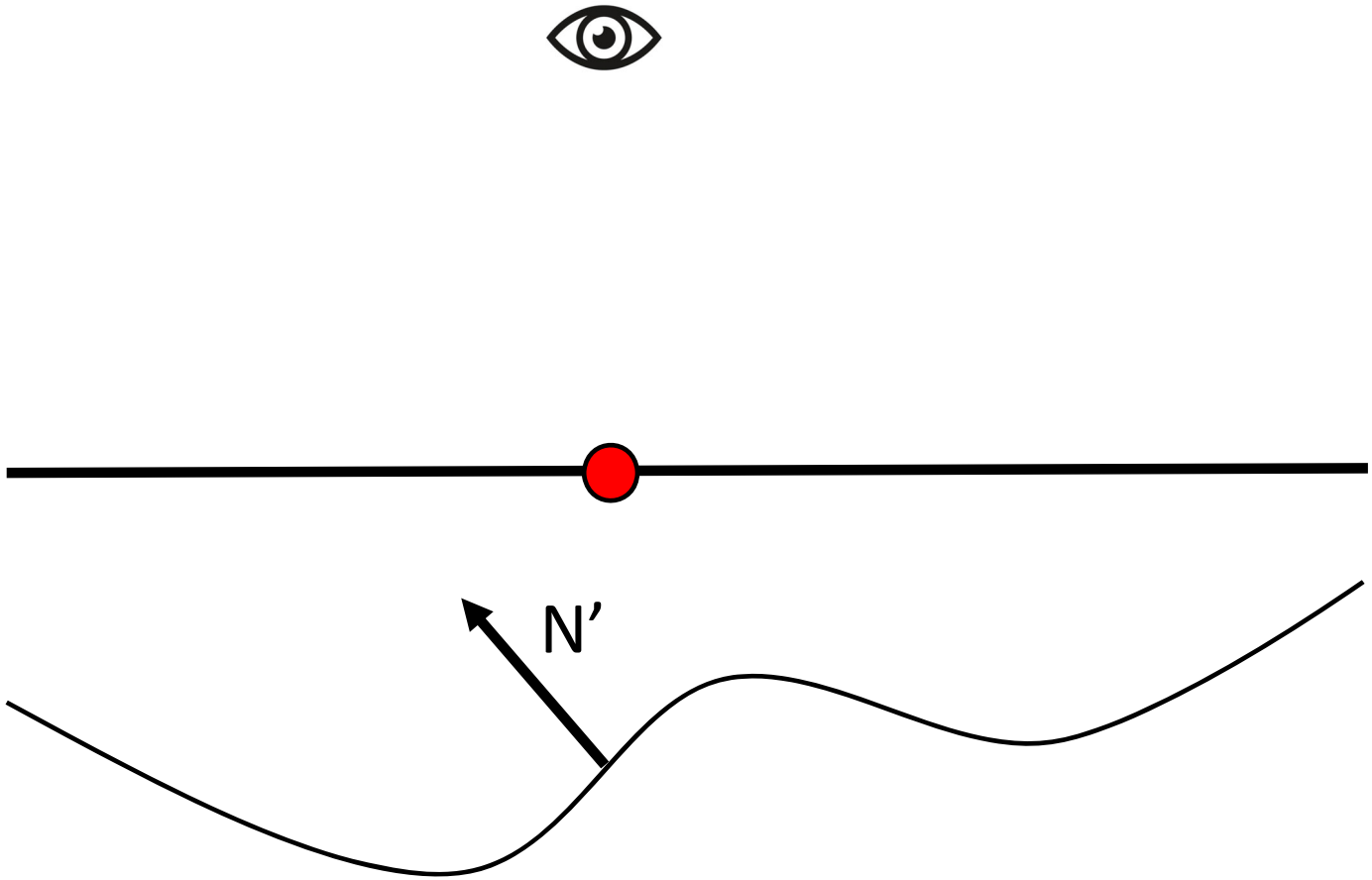


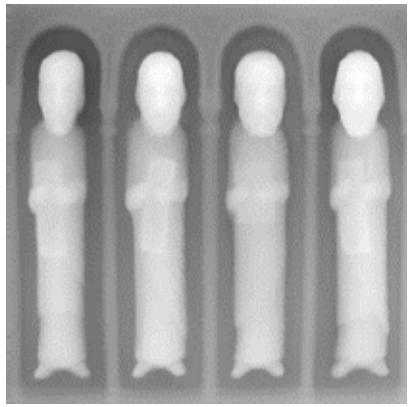
Bump mapping/Normal mapping

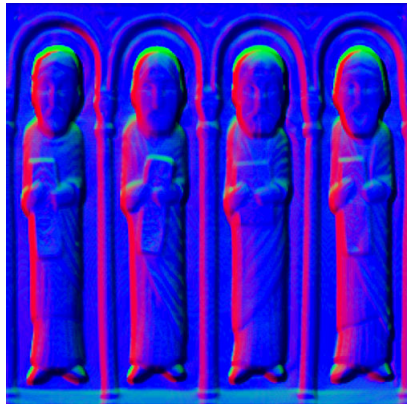



Relief Mapping
File View Render About...
Normal Mapping
Border Clamp
Depth Bias
FPS: 59

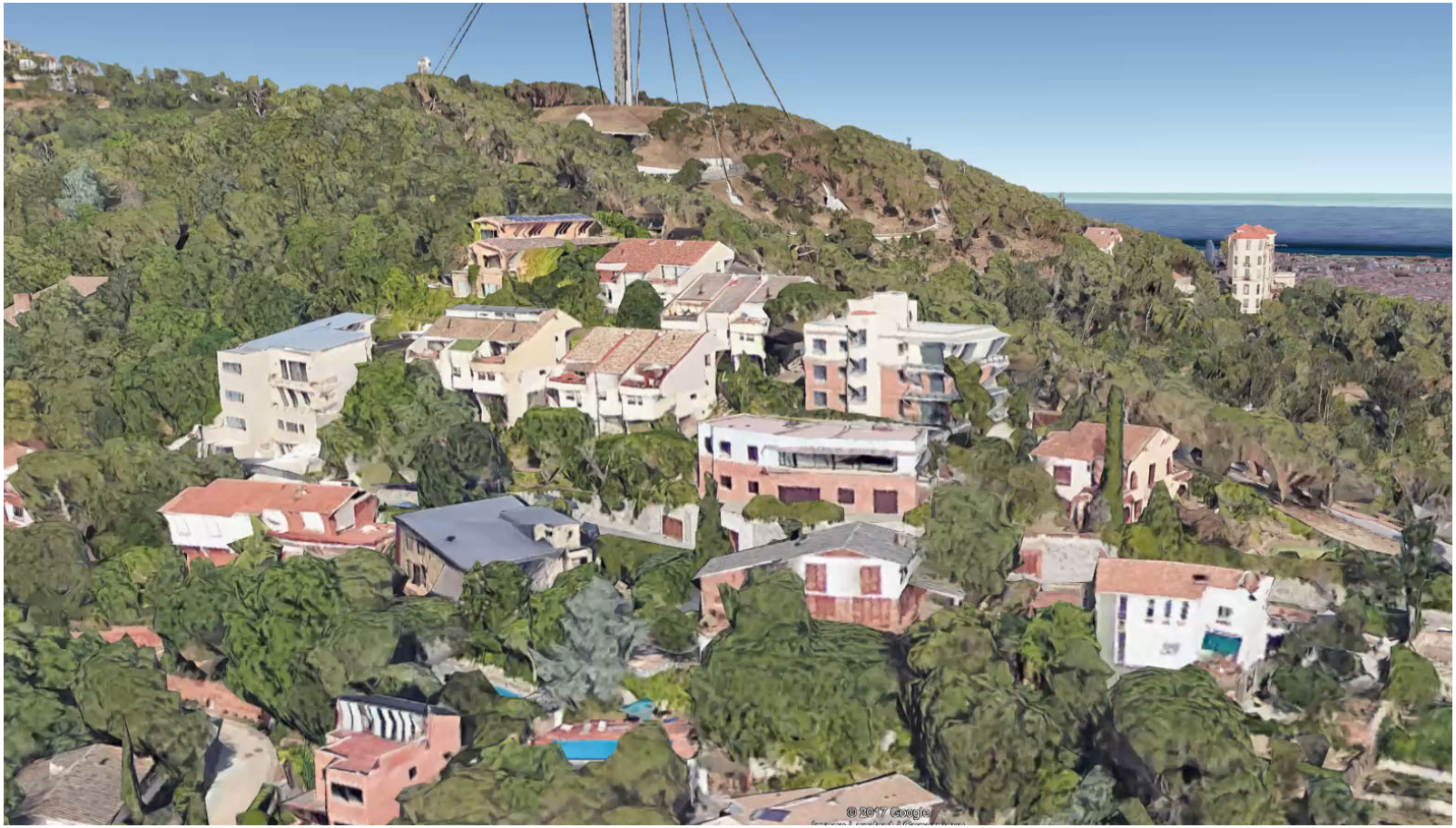
Problema Bump mapping/Normal mapping

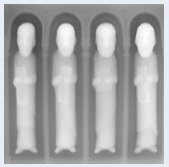
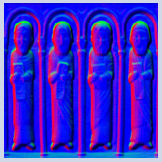




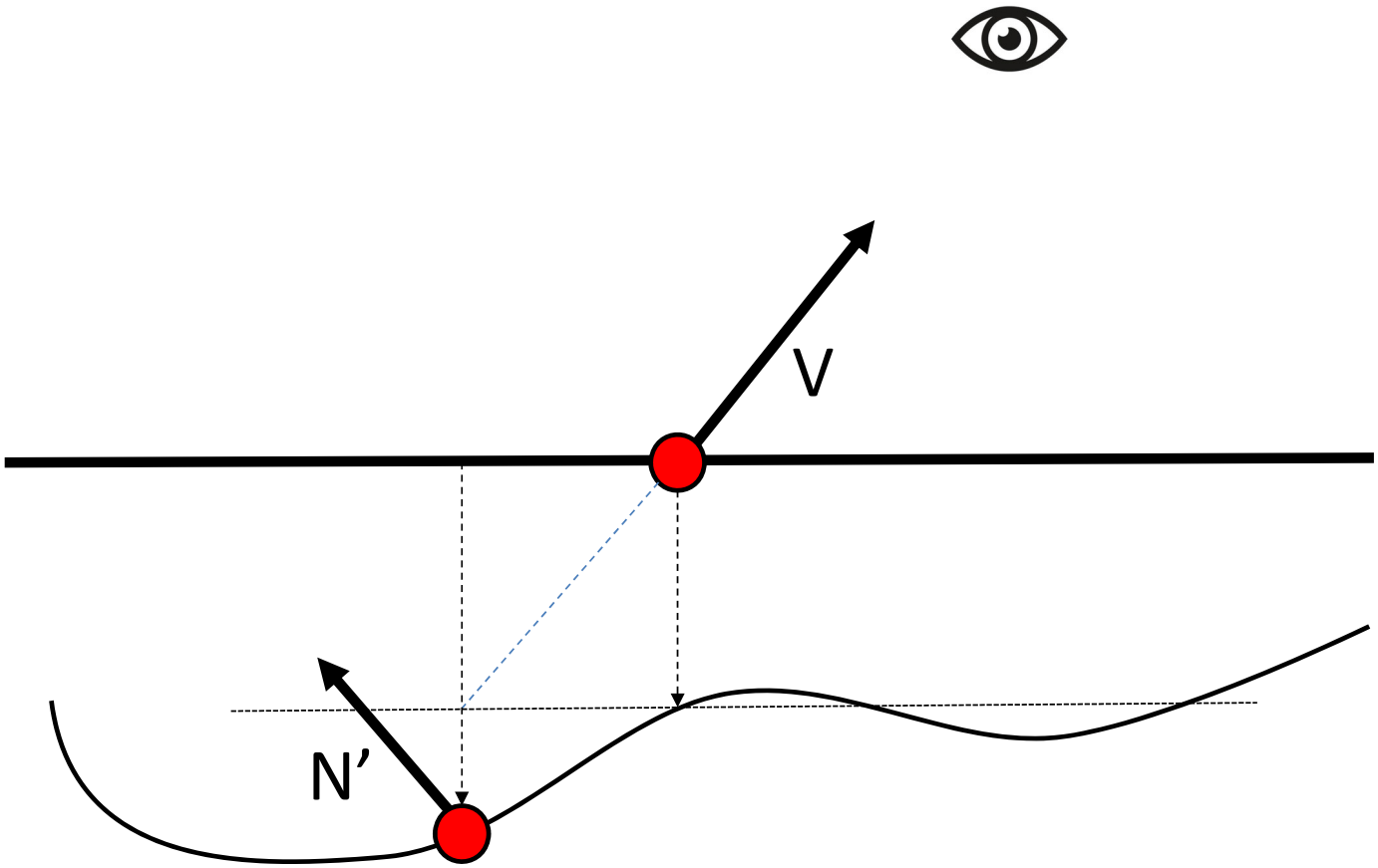


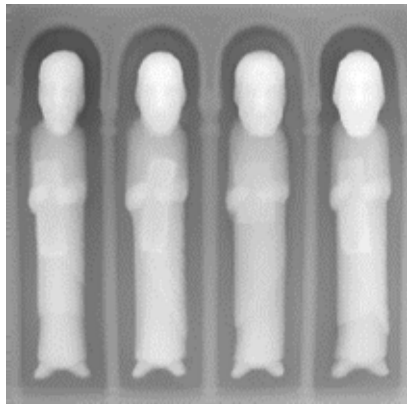
View-motion parallax

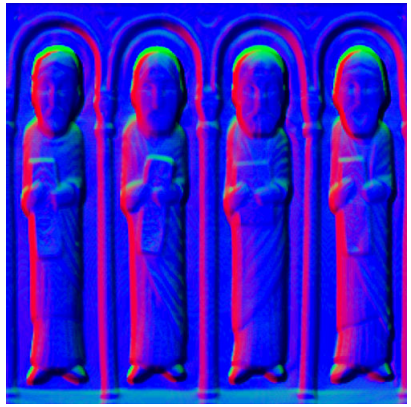


Tècnica	Info a la textura	Ús de la textura	Coords de textura	View parallax	Self-occlusion	Detailed silhouette	On s'aplica
Color mapping	RGB 3	Kd del material	(s,t)	-	-	-	FS
Bump mapping	D 1 	Modificar la normal	(s,t)	N	N	N	FS
Normal mapping	Normal 3 	Modificar la normal	(s,t)	N	N	N	FS
Parallax mapping	Normal + D 3+1 o 4	Modificar la normal	$(s+d_s, t+d_t)$	S	N	N	FS
Relief mapping	Normal + D 3+1 o 4	Modificar la normal; descartar fragments	$(s+d_s, t+d_t)$	S	S	S	FS!!!
Displacement mapping	D 1	Desplaçar els vèrtexs un cop subdividits els polígons.	(s,t)	S	S	S	CPU GS TCS+TES

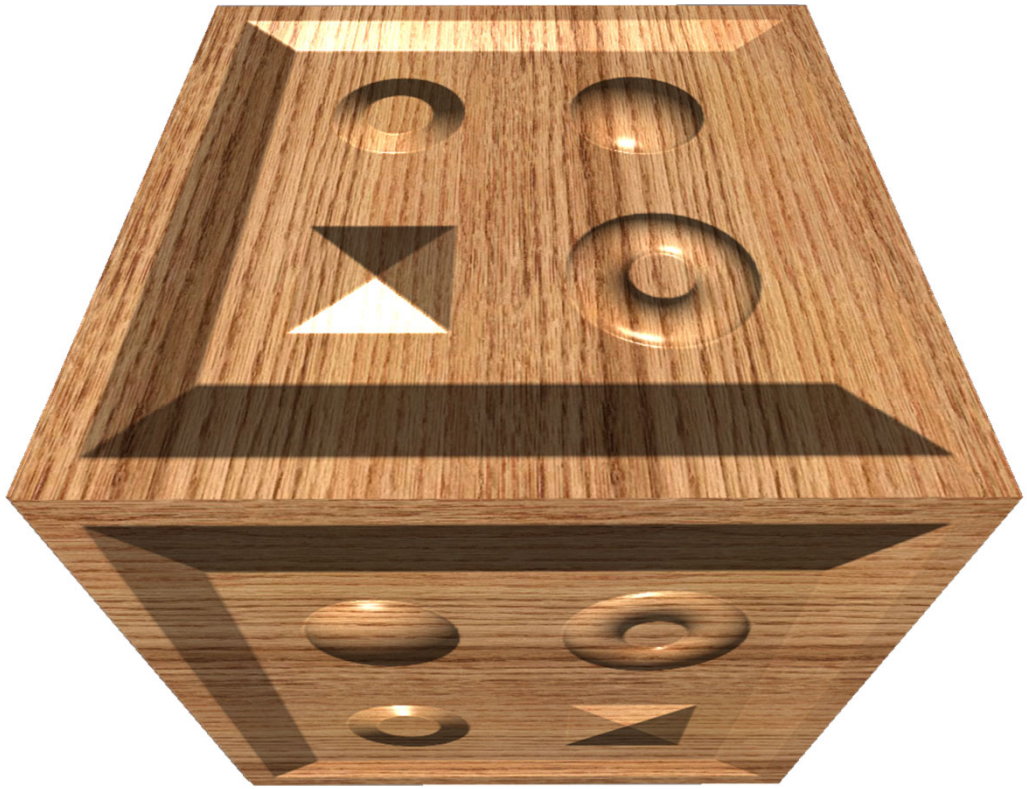
Parallax mapping

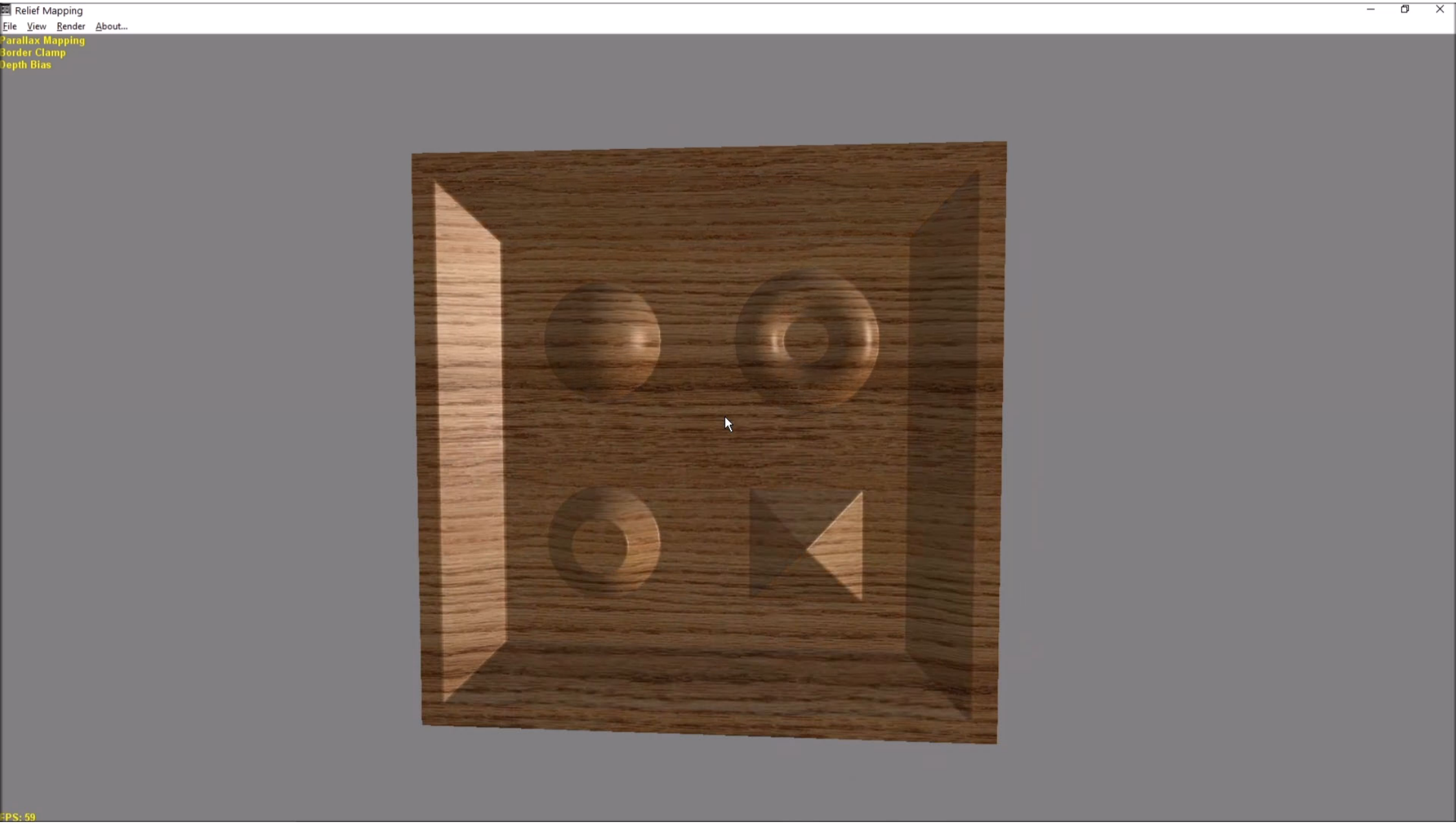




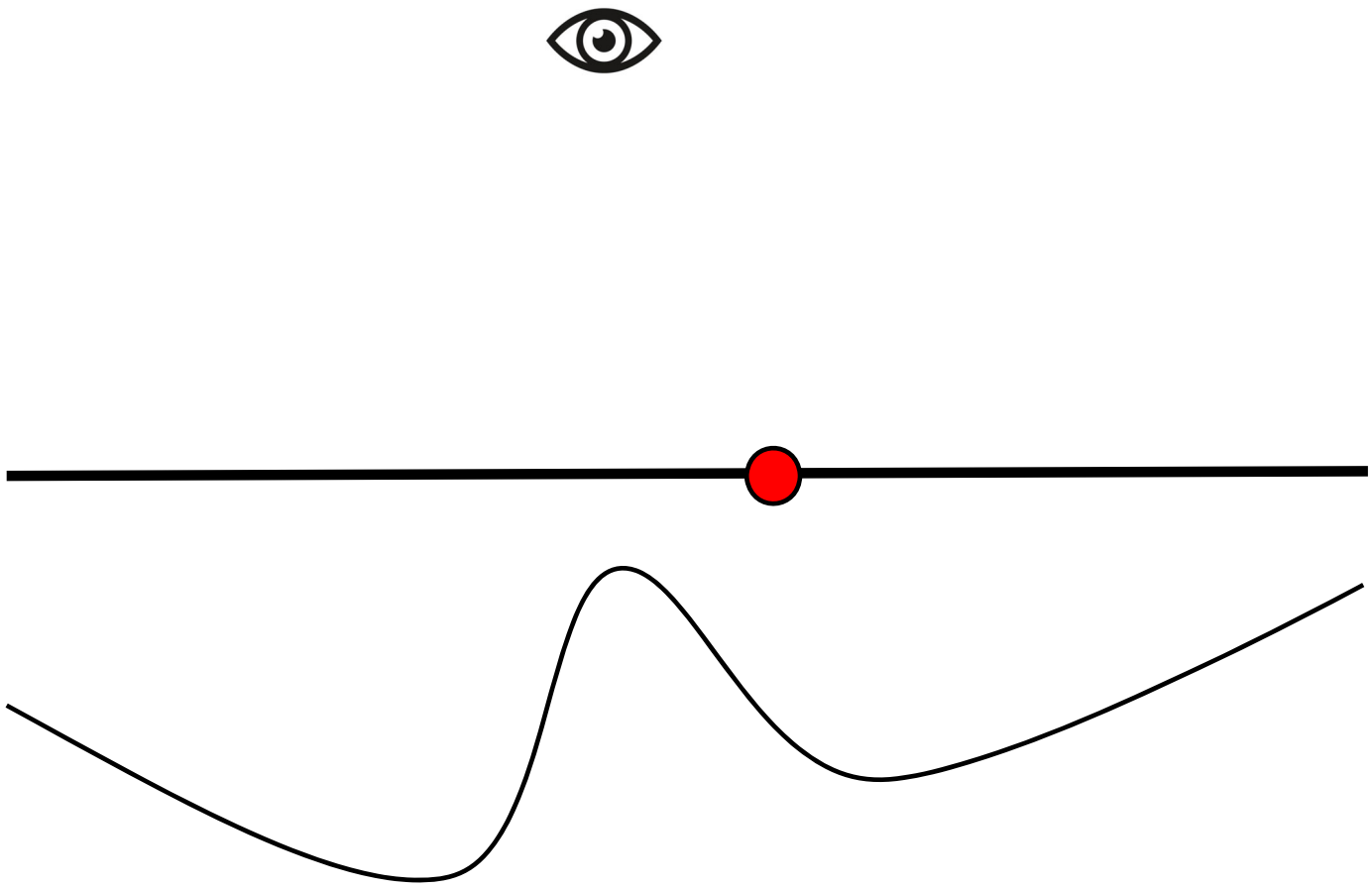


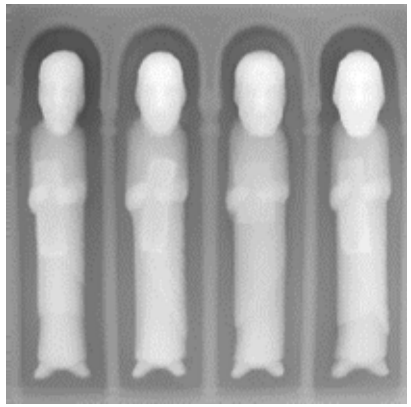
Parallax mapping

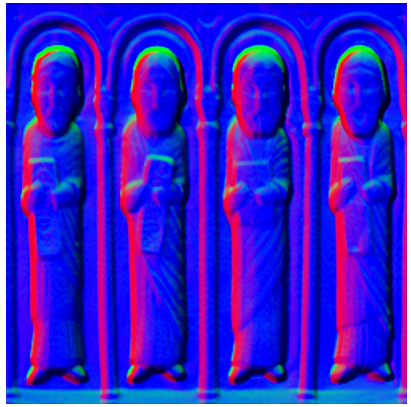


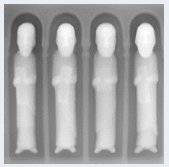
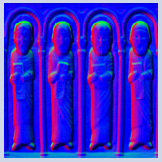

Relief Mapping
File View Render About...
Parallax Mapping
Border Clamp
Depth Bias
FPS: 59

Problema Parallax mapping

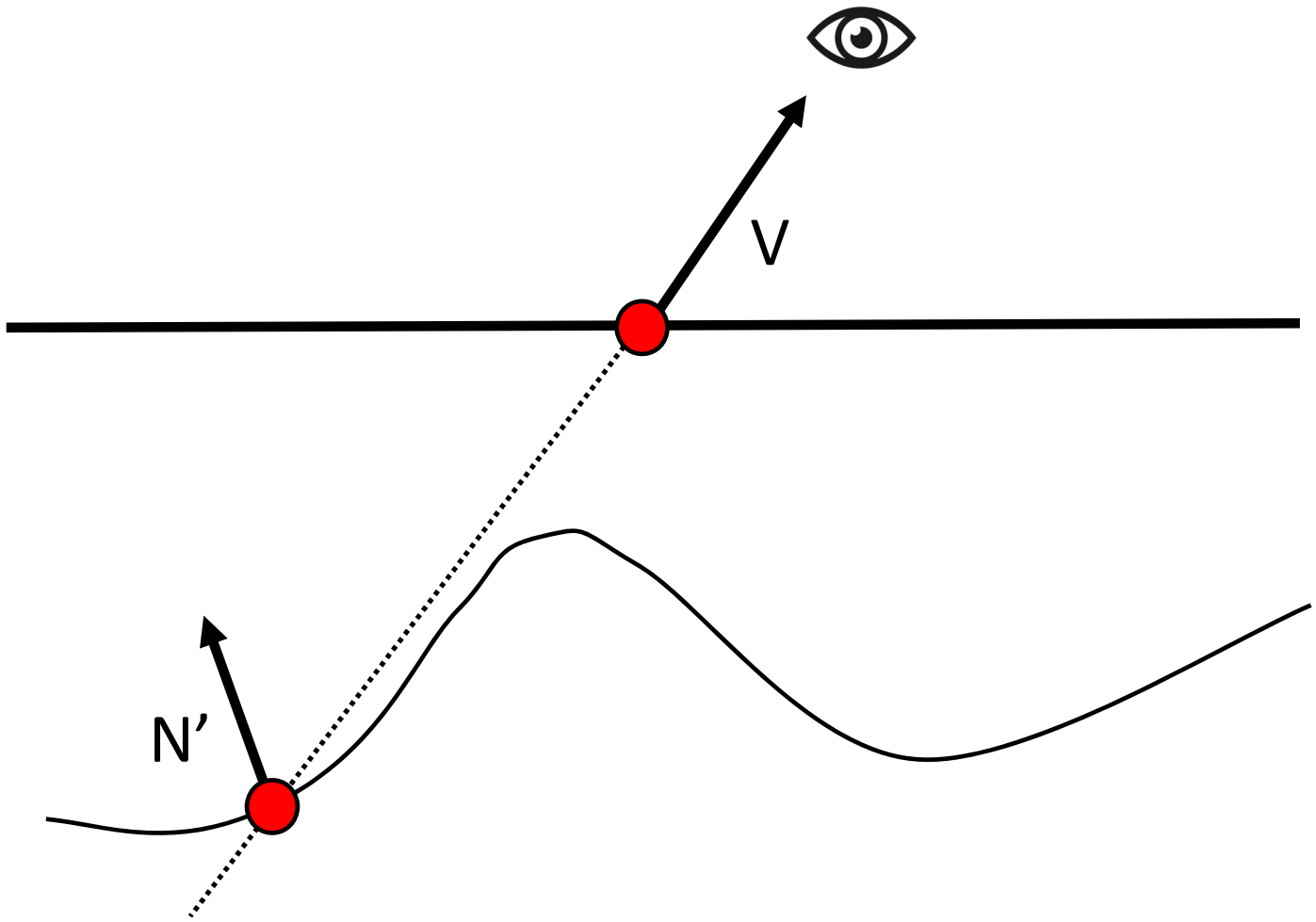


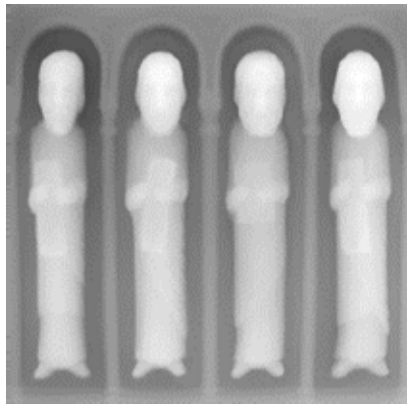


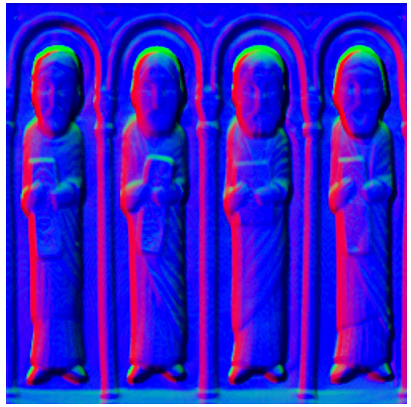


Tècnica	Info a la textura	Ús de la textura	Coords de textura	View parallax	Self-occlusion	Detailed silhouette	On s'aplica
Color mapping	RGB 3	Kd del material	(s,t)	-	-	-	FS
Bump mapping	D 1 	Modificar la normal	(s,t)	N	N	N	FS
Normal mapping	Normal 3 	Modificar la normal	(s,t)	N	N	N	FS
Parallax mapping	Normal + D 3+1 o 4	Modificar la normal	$(s+d_s, t+d_t)$	S	N	N	FS
Relief mapping	Normal + D 3+1 o 4	Modificar la normal; descartar fragments	$(s+d_s, t+d_t)$	S	S	S	FS!!!
Displacement mapping	D 1	Desplaçar els vèrtexs un cop subdividits els polígons.	(s,t)	S	S	S	CPU GS TCS+TES

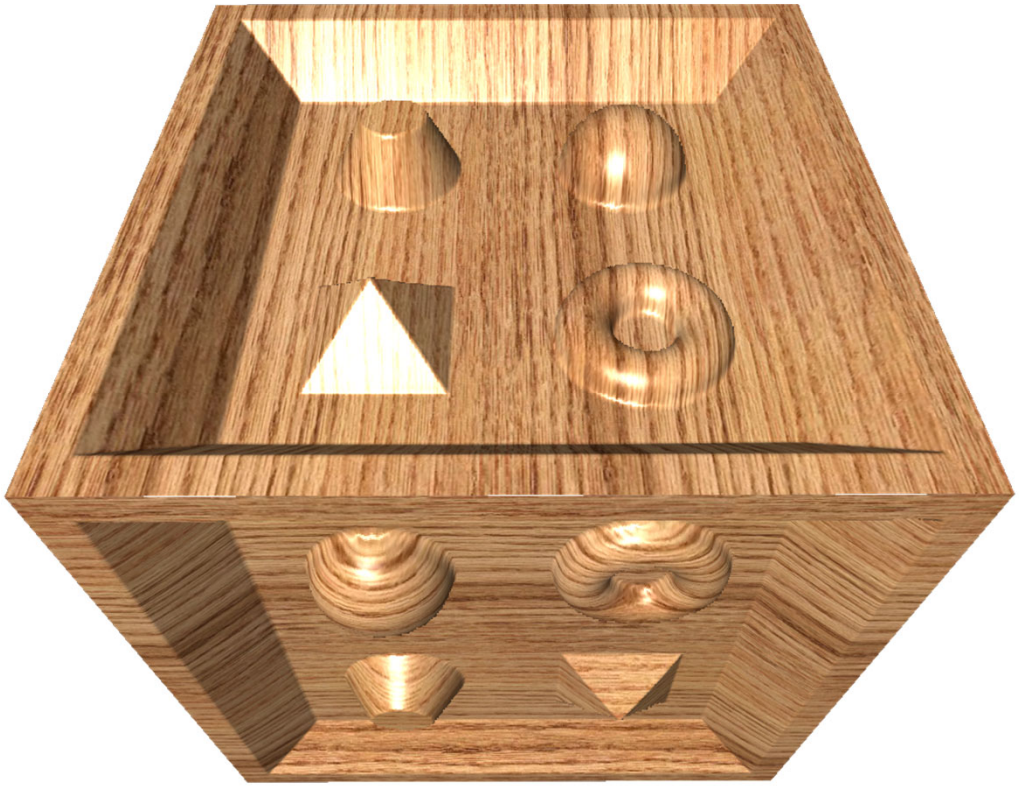
Relief mapping



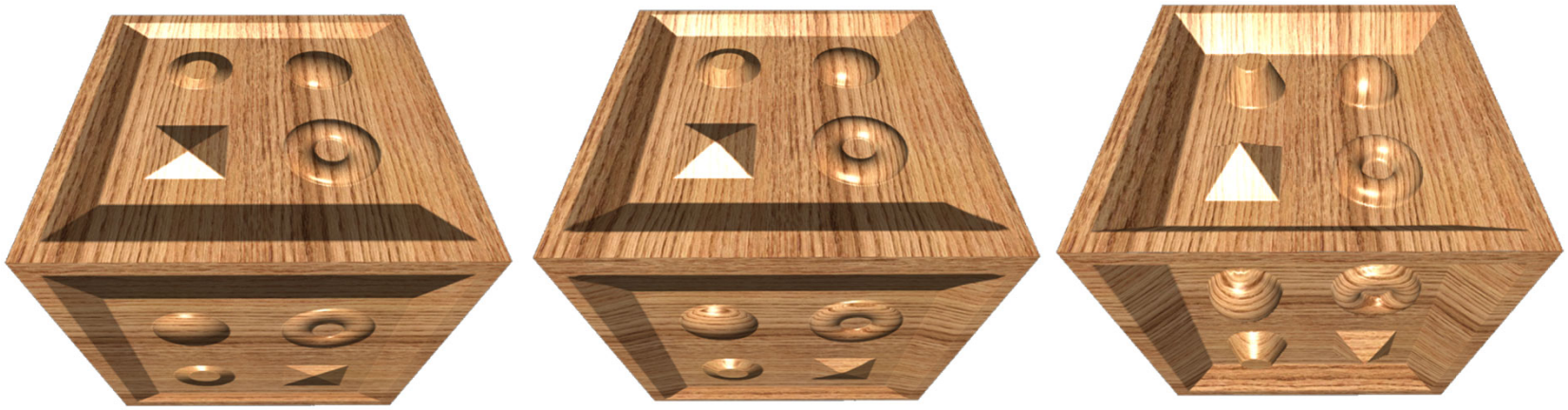


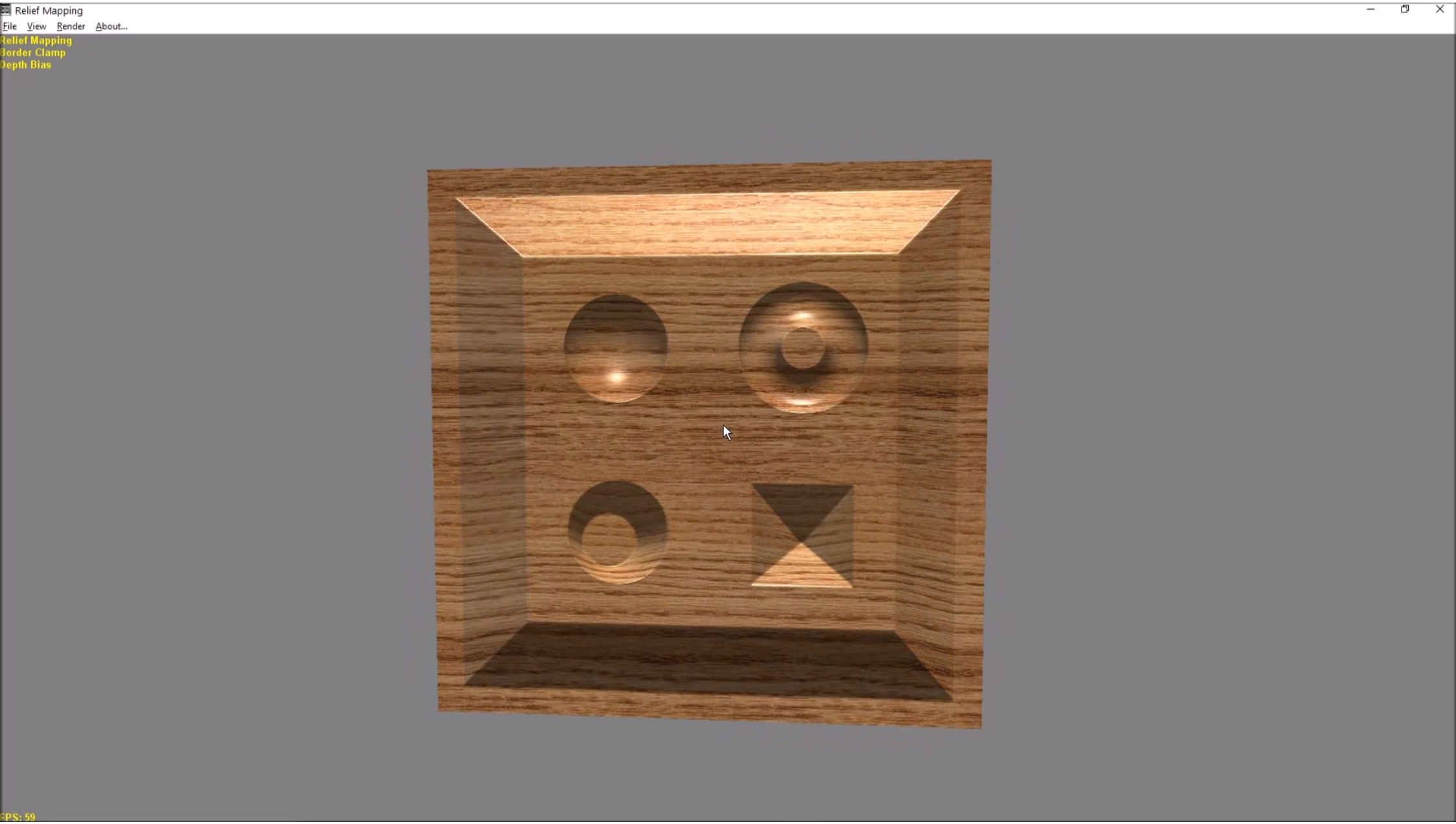


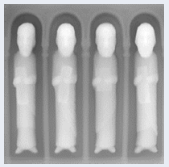
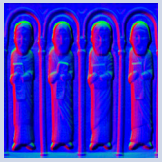
Relief mapping



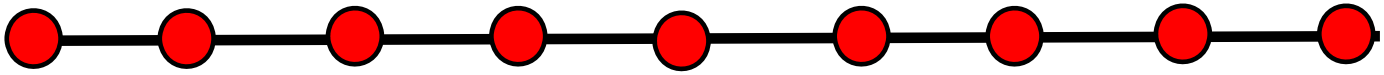
Comparació

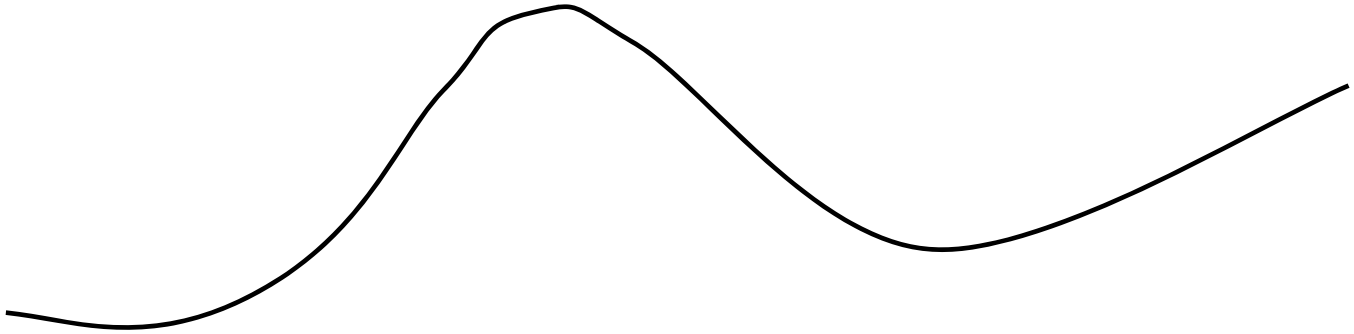


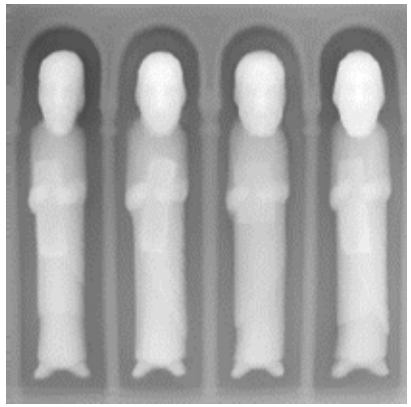

Relief Mapping
File View Render About...
Relief Mapping
Border Clamp
Depth Bias
FPS: 59

Tècnica	Info a la textura	Ús de la textura	Coords de textura	View parallax	Self-occlusion	Detailed silhouette	On s'aplica
Color mapping	RGB 3	Kd del material	(s,t)	-	-	-	FS
Bump mapping	D 1 	Modificar la normal	(s,t)	N	N	N	FS
Normal mapping	Normal 3 	Modificar la normal	(s,t)	N	N	N	FS
Parallax mapping	Normal + D 3+1 o 4	Modificar la normal	$(s+d_s, t+d_t)$	S	N	N	FS
Relief mapping	Normal + D 3+1 o 4	Modificar la normal; descartar fragments	$(s+d_s, t+d_t)$	S	S	S	FS!!!
Displacement mapping	D 1	Desplaçar els vèrtexs un cop subdividits els polígons.	(s,t)	S	S	S	CPU GS TCS+TES

Displacement mapping





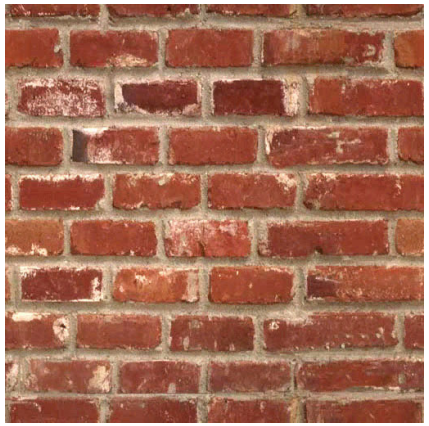


Comparació

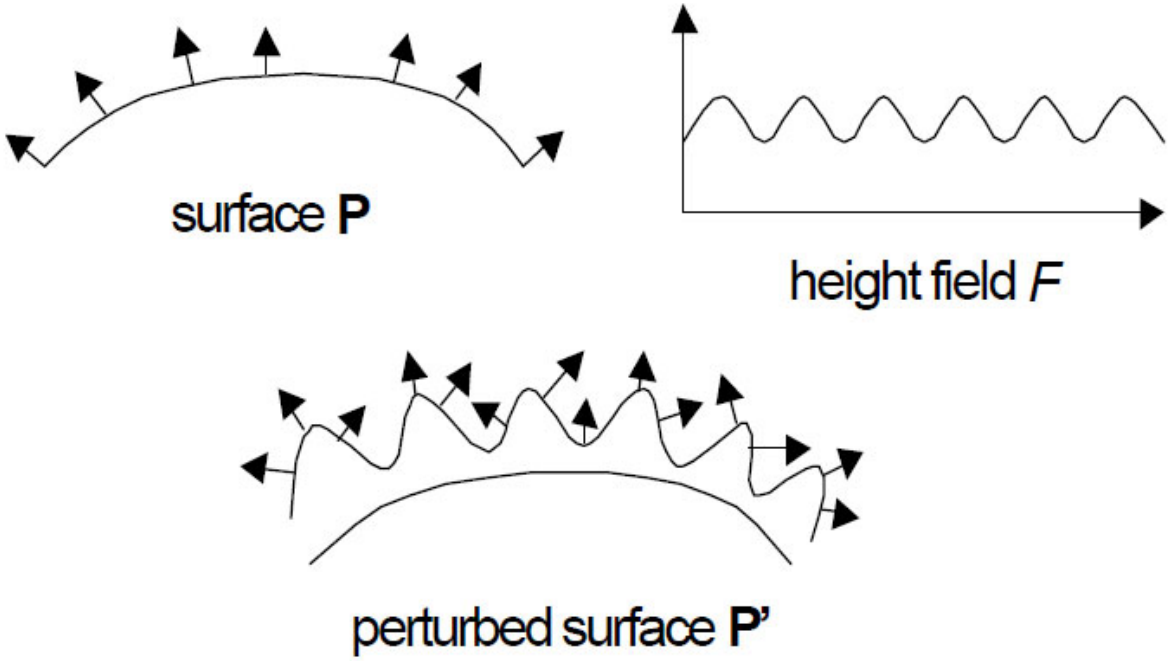
Mark Kilgard. A Practical and Robust Bump-mapping Technique for Today's GPUs. GDC 2000

BUMP MAPPING

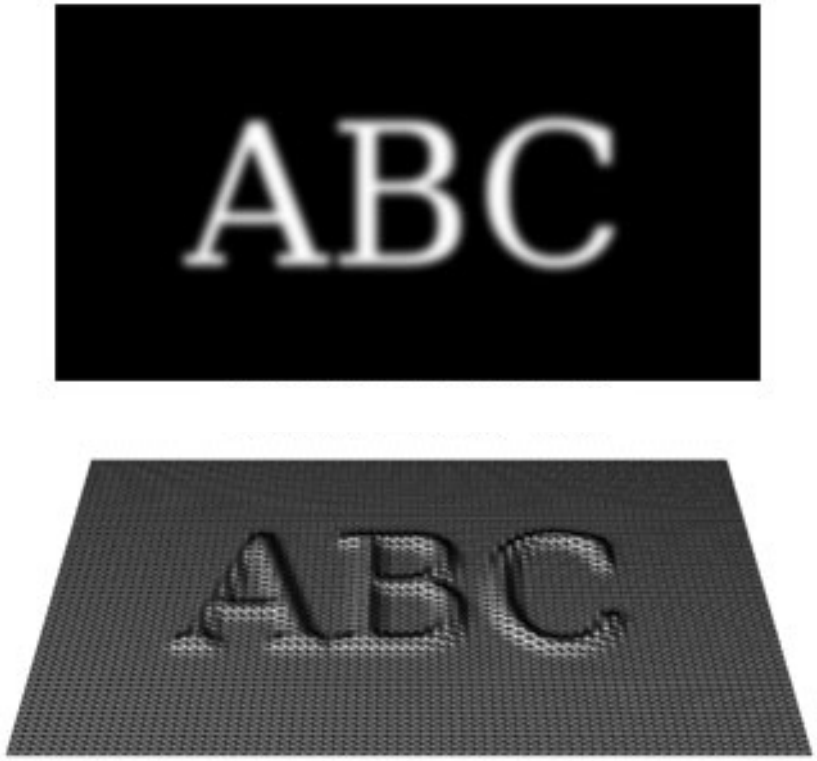
Bump mapping



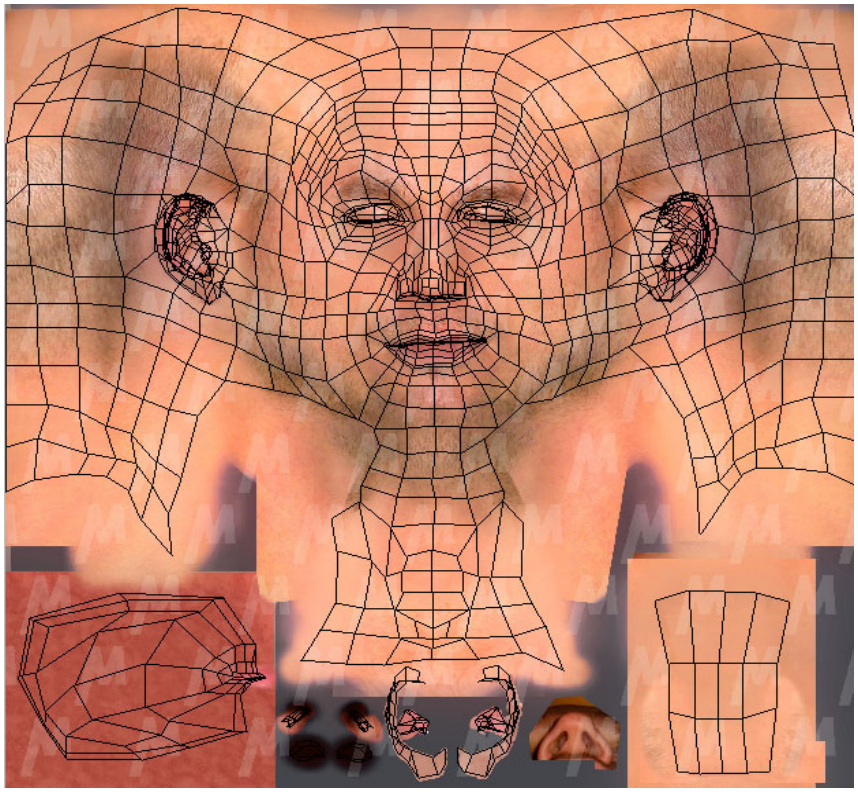
Elements bàsics

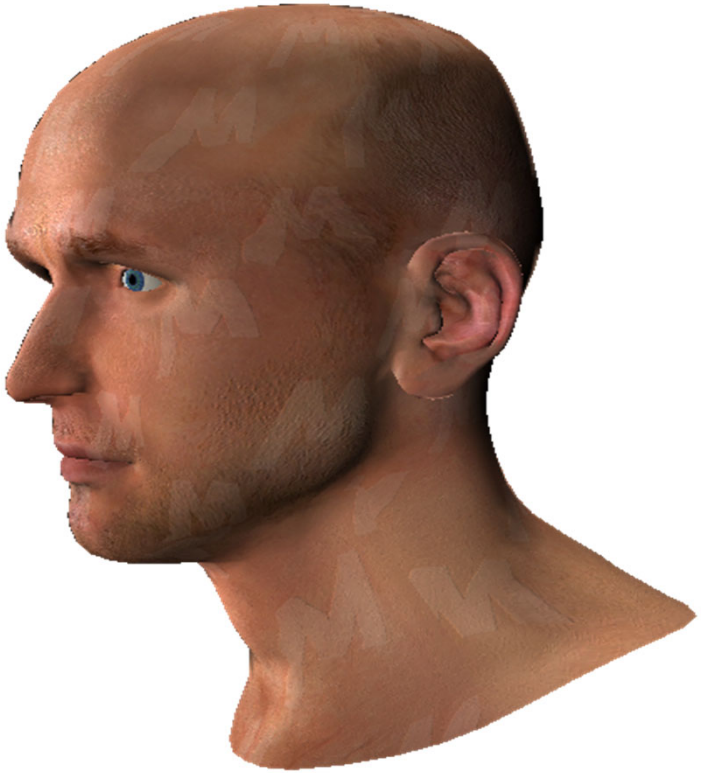


Height field



Malla amb coordenades de textura





Eqüacions

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

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

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

Eqüacions

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

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

$$\mathbf{N}' = \frac{\partial \mathbf{P}}{\partial u} \times \frac{\partial \mathbf{P}}{\partial v} + \frac{\frac{\partial F}{\partial u}\left(\mathbf{N} \times \frac{\partial \mathbf{P}}{\partial v}\right)}{|\mathbf{N}|} + \frac{\frac{\partial F}{\partial v}\left(\frac{\partial \mathbf{P}}{\partial u} \times \mathbf{N}\right)}{|\mathbf{N}|} + \frac{\frac{\partial F}{\partial u}\frac{\partial F}{\partial v}(\mathbf{N} \times \mathbf{N})}{|\mathbf{N}|^2}$$

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

Eqüacions - resum

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

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

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