Reflexions especulars

Carlos Andújar

Maig 2015

Introducció

Vector reflectit

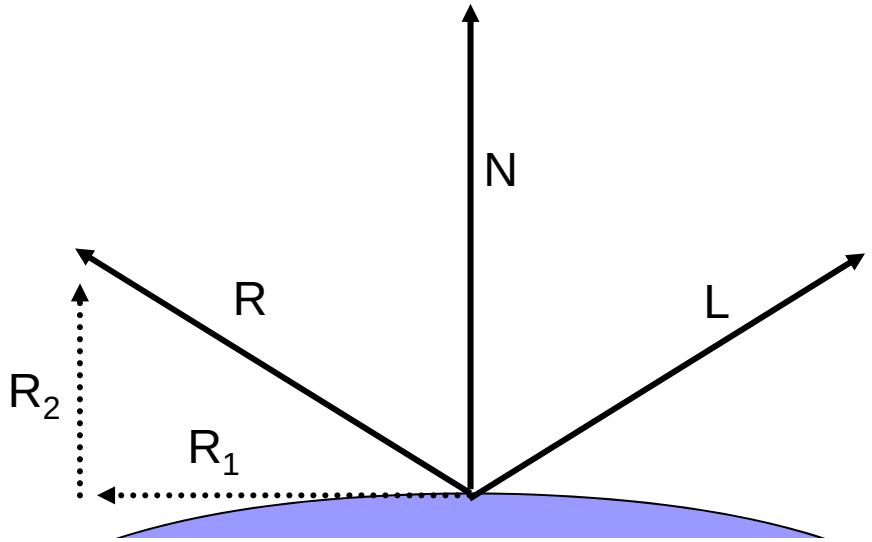
$R = R_1 + R_2$

$R_1 = -L + R_2$

$R = 2R_2 - L$

$R_2 = (N \cdot L)N$

$R = 2\,(N \cdot L)N - L$



Vector reflectit

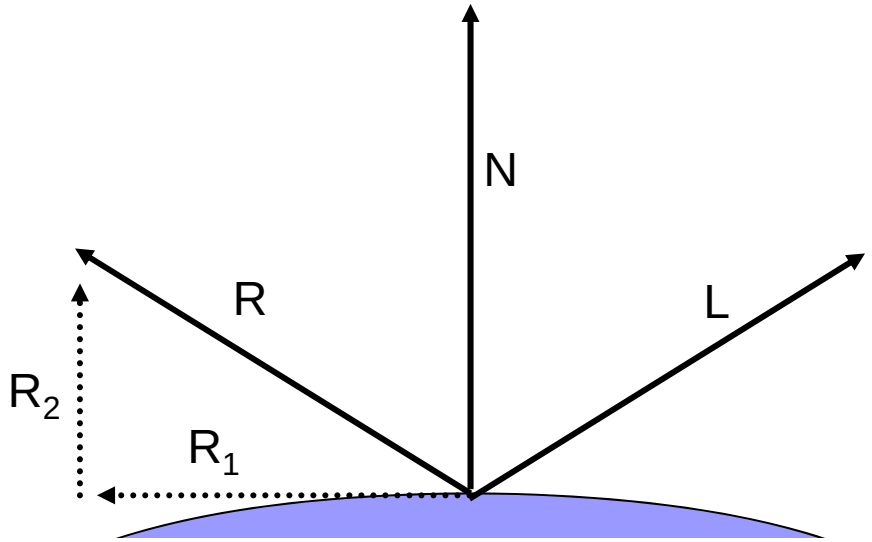
$R = R_1 + R_2$

$R_1 = -L + R_2$

$R = 2R_2 - L$

$R_2 = (N \cdot L)N$

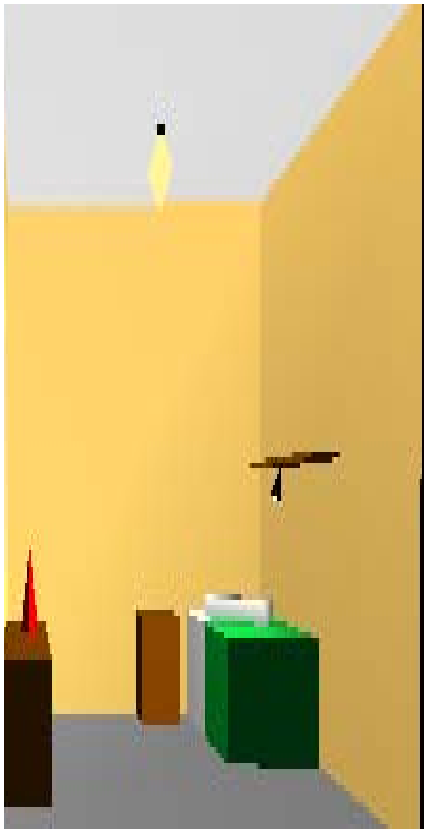
$R = 2\,(N \cdot L)N - L$

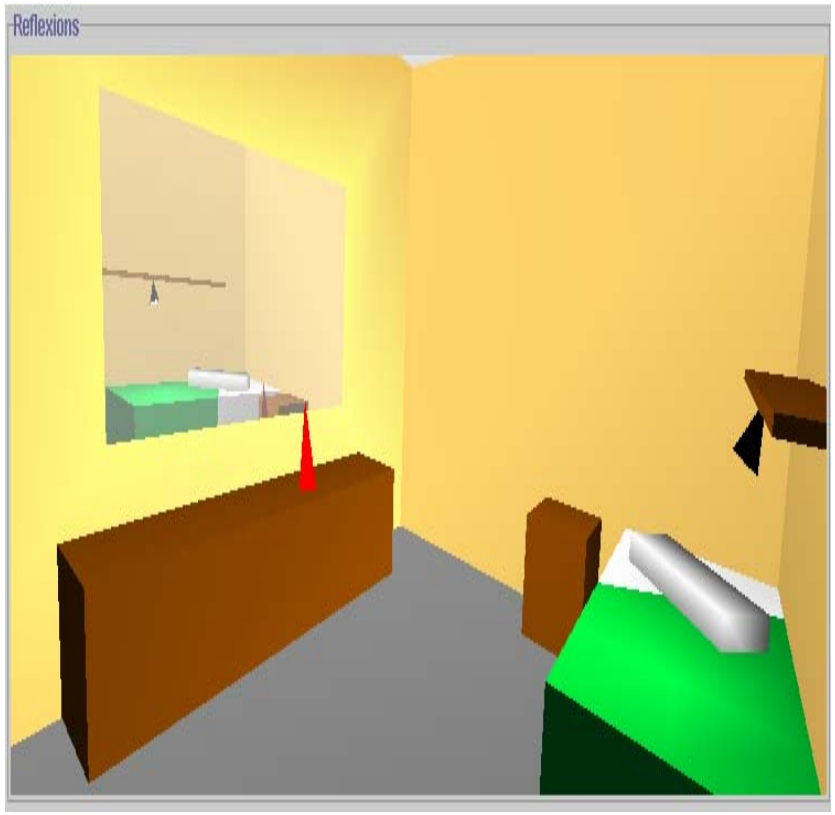


REFLEXIONS AMB OBJECTES VIRTUALS

Objectes virtuals

Objectes virtuals



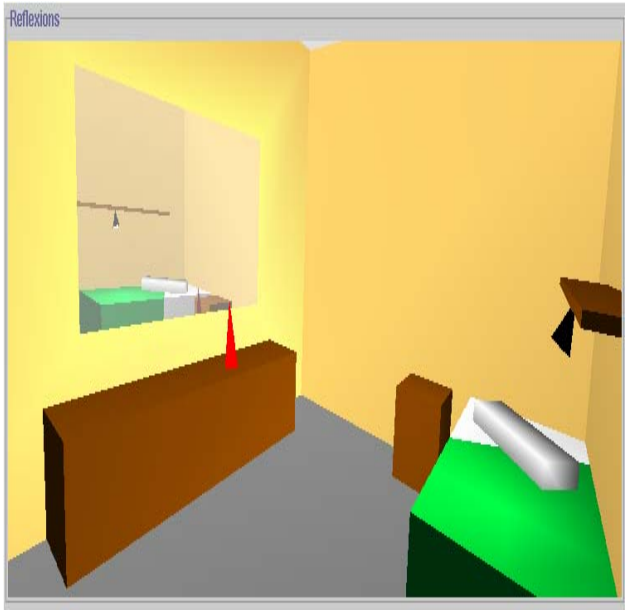


Algorisme (versió 1)

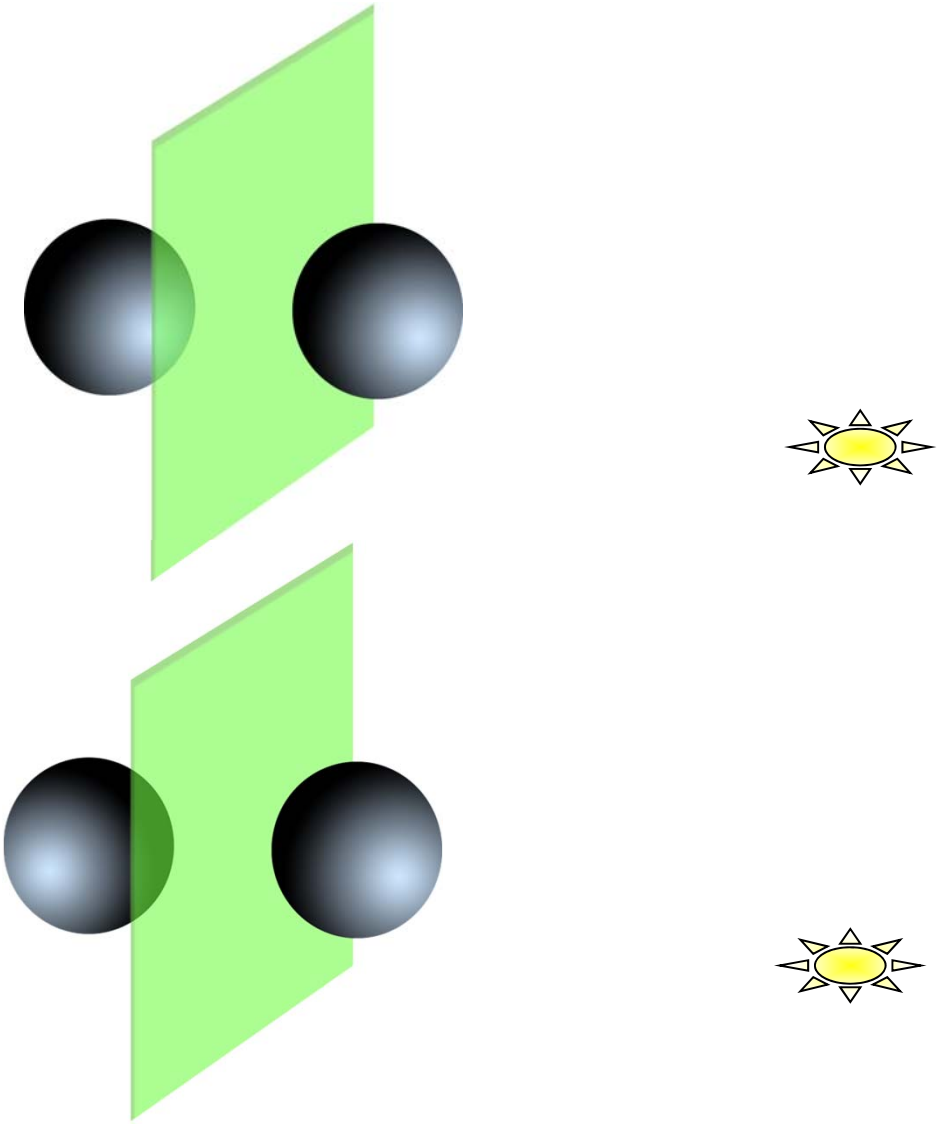
// 1. Dibuixar els objectes en posició virtual
glPushMatrix();
glMultMatrix(matriu_simetria)
glLightfv(GL_LIGHT0, **GL_POSITION**, pos);
glCullFace(GL_FRONT);
dibuixar(escena);
glPopMatrix();

// 2. Dibuixar el mirall semi-transparent
glLightfv(GL_LIGHT0, **GL_POSITION**, pos);
glCullFace(GL_BACK);
dibuixar(mirall);

// 3. Dibuixar els objects en posició real
dibuixar(escena);

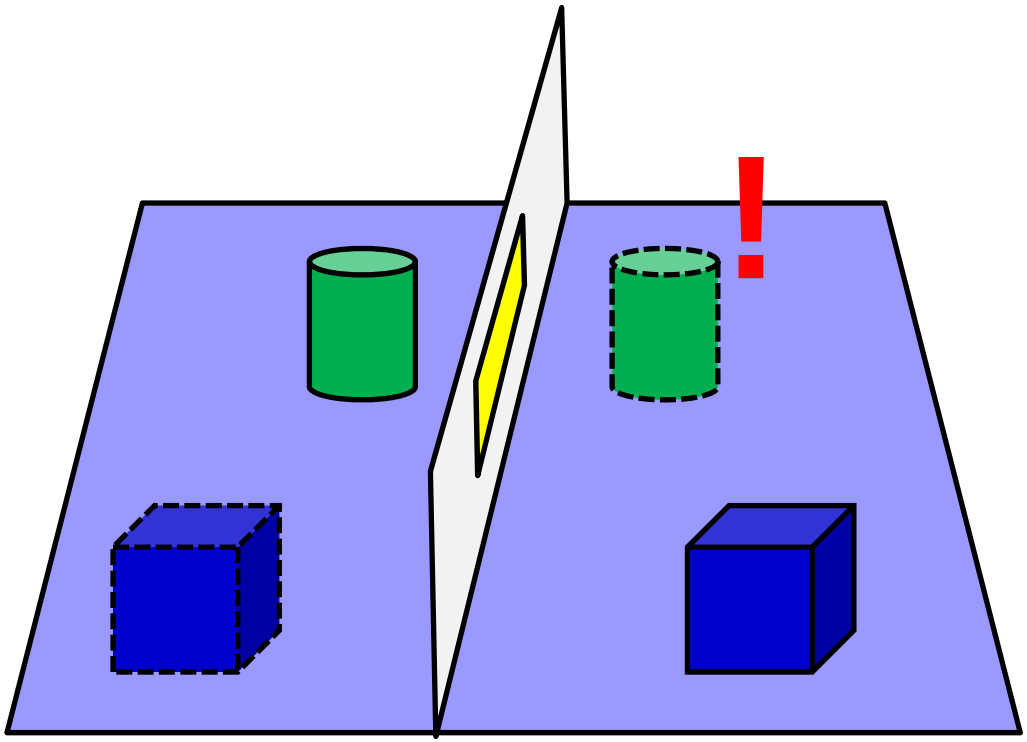


Reflexió de les fonts de llum



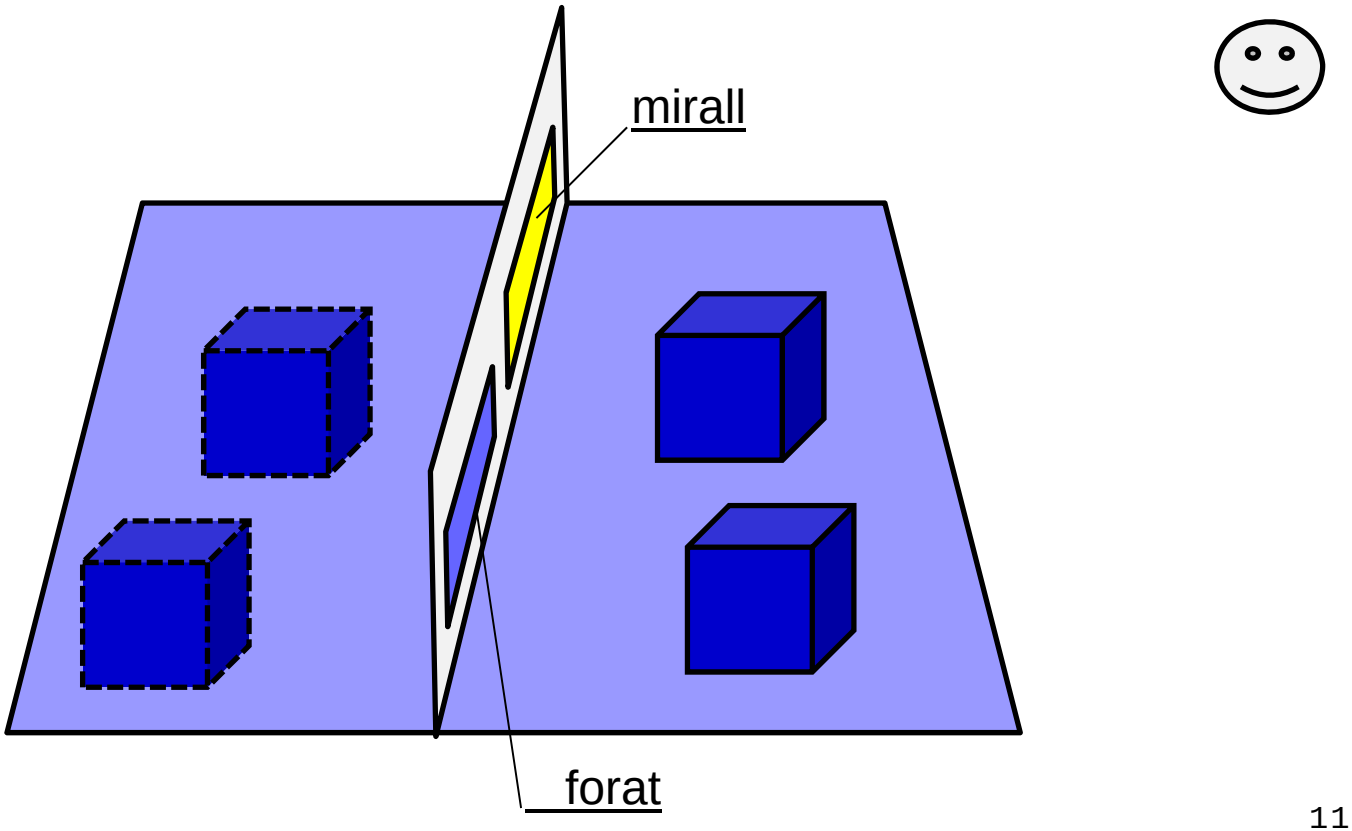
Limitacions

Assumeix que els objectes virtuals estan en el semiespai positiu del pla del mirall.



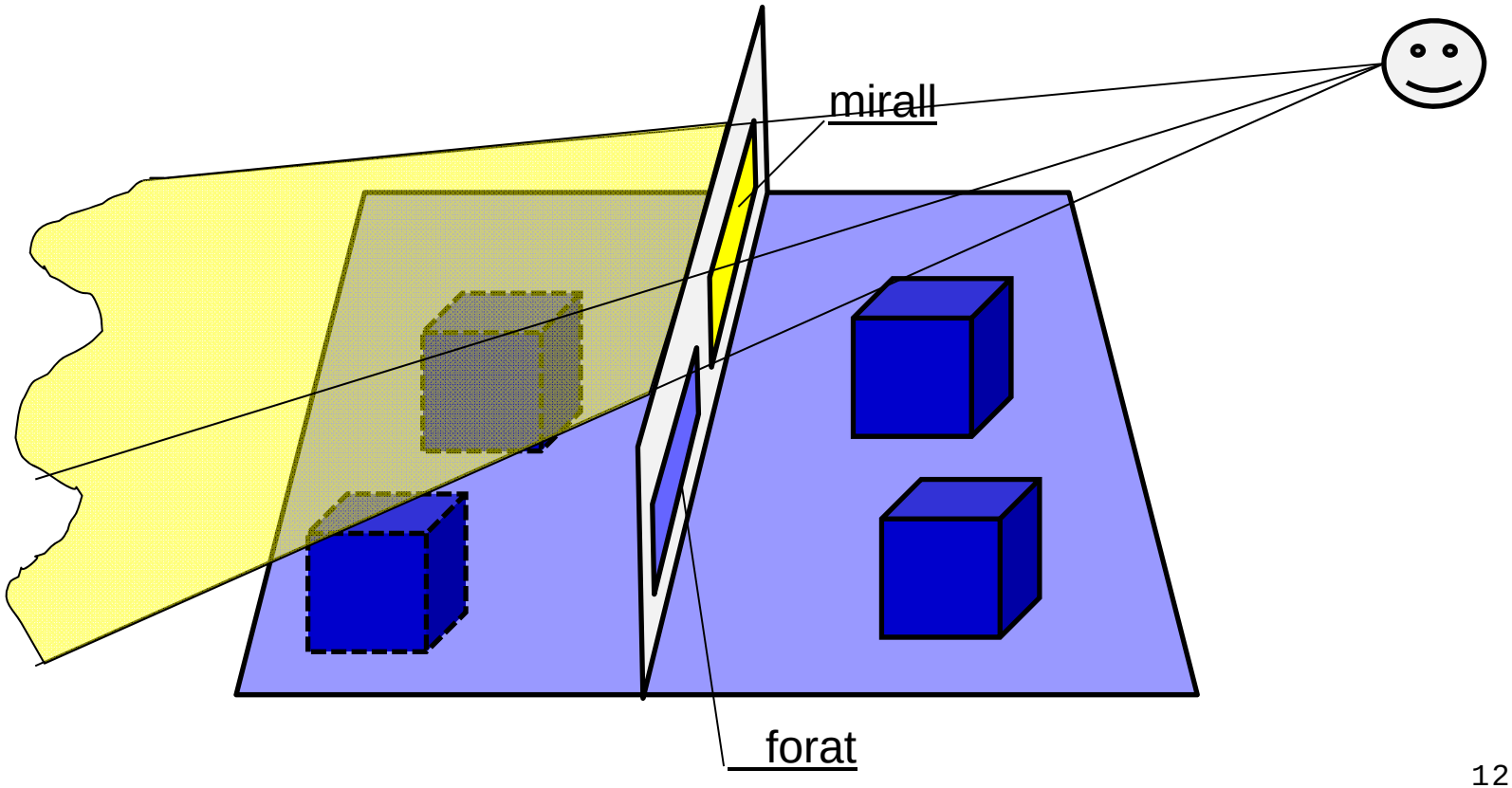
Limitacions

Assumeix que els objectes virtuals només es veuran a través del forat del mirall.



Solució 1

Dibuixar els objectes virtuals amb plans de retallat addicionals – glClipPlane()

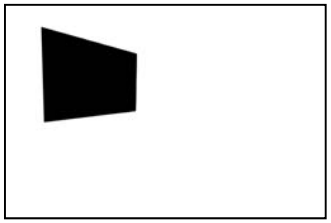


Solució 2

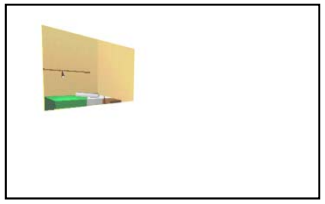
Usar stencil per limitar els objectes virtuals a la regió ocupada pel mirall.

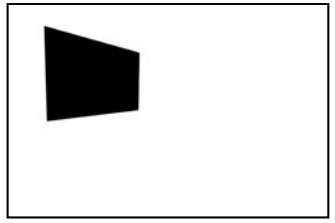
Algorisme (versió 2)

Pas 1. Dibuixar mirall al stencil buffer

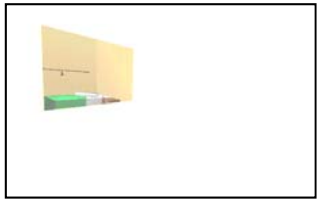


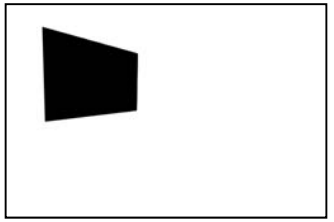
Pas 2. Dibuixar objectes en pos virtual



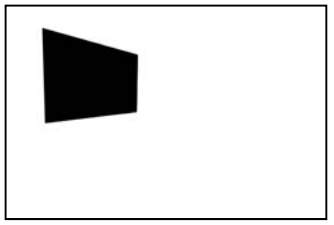


Pas 3. Dibuixar mirall semi-transparent



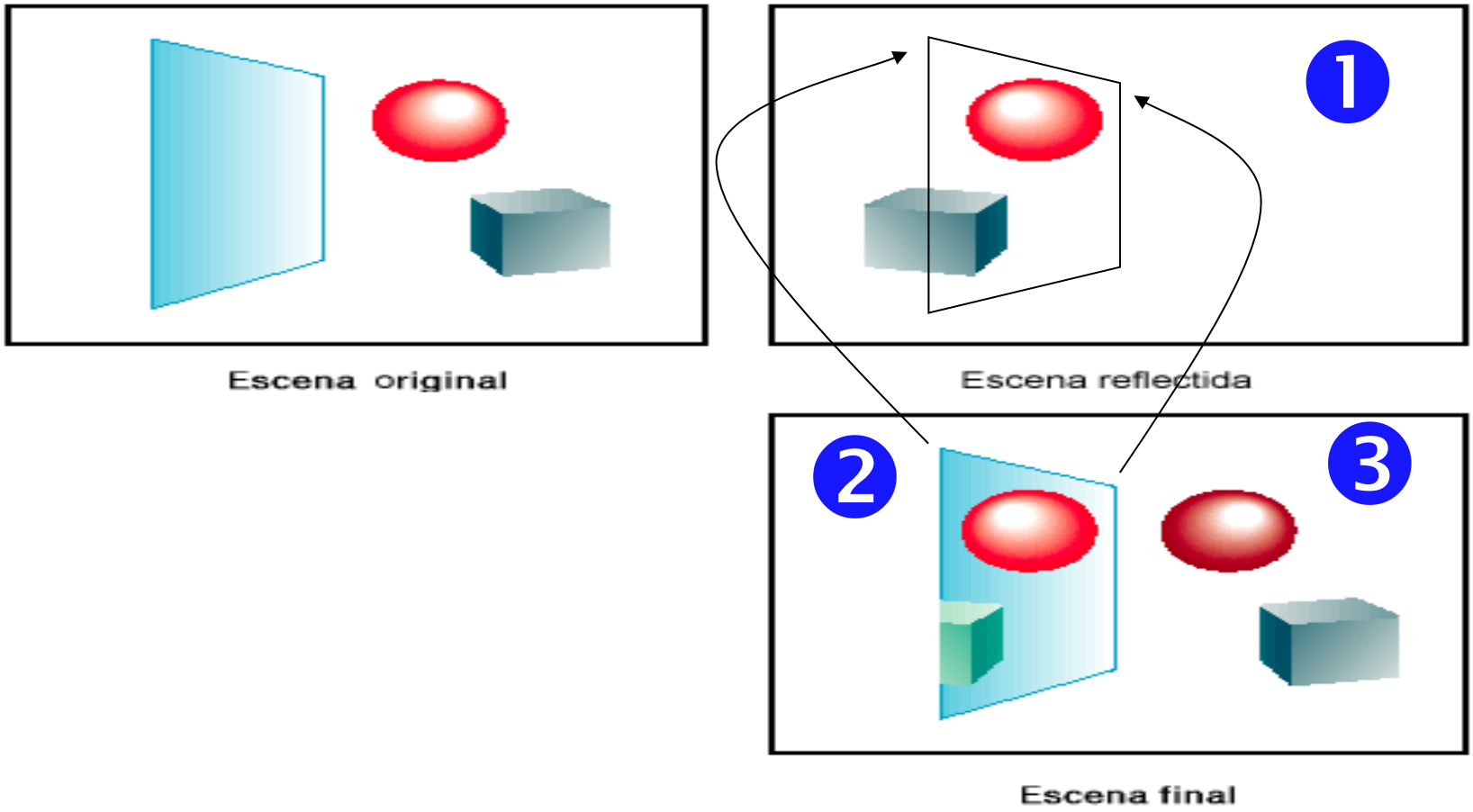


Pas 4. Dibuixar objectes en pos real

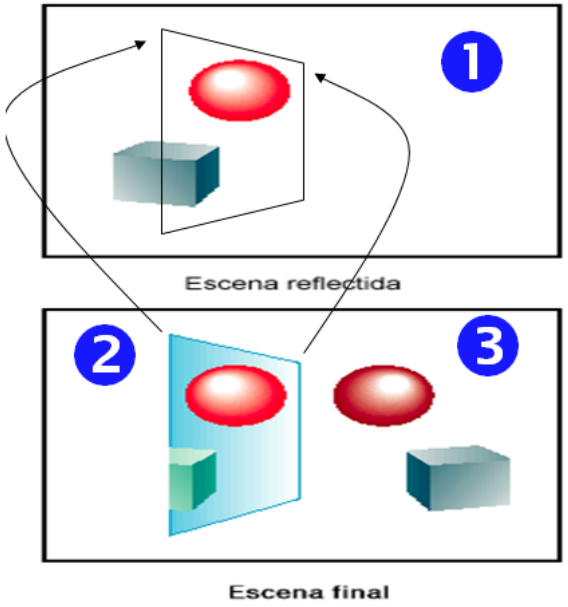


REFLEXIONS AMB TEXTURES DINÀMIQUES

Textures dinàmiques



Textures dinàmiques




```
GLdouble modelview[16];
glGetDoublev(GL_MODELVIEW_MATRIX, modelview);
GLdouble projection[16];
glGetDoublev(GL_PROJECTION_MATRIX, projection
GLint viewport[4] = {0, 0, 1, 1};
//glGetIntegerv(GL_VIEWPORT, viewport);

GLdouble s,t,r;
gluProject(x, y, z, modelview, projection, viewport, &s, &t, &r);
```

Model space

Window space

MATRIU DE REFLEXIÓ

Matriu de reflexió

Matriu de reflexió respecte un pla (a,b,c,d):

$$\begin{bmatrix} 1-2a^2 & -2ba & -2ca & -2da \\ -2ba & 1-2b^2 & -2cb & -2db \\ -2ca & -2cb & 1-2c^2 & -2dc \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

ENVIRONMENT MAPPING

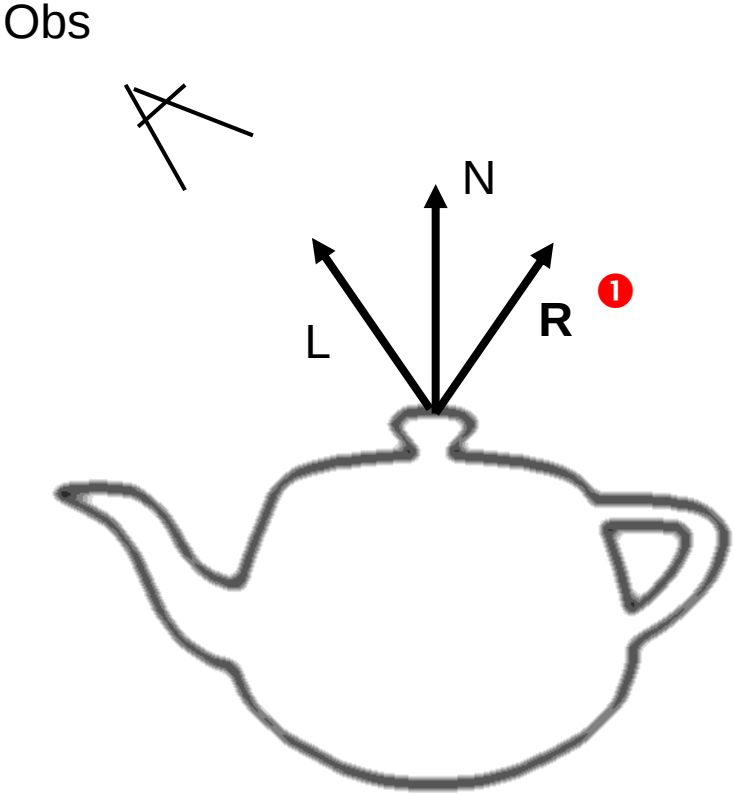
Environment map

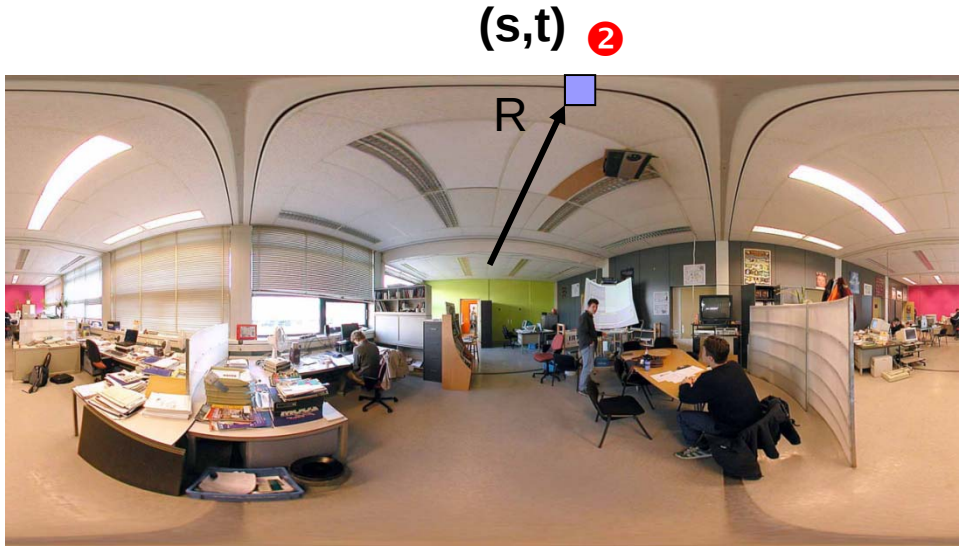
Funció (o textura) que, donada una direcció arbitrària R, ens retorna el color de l'entorn en direcció R

color = environmentMap(R)

Representació com a textura

Coords de textura d'un vèrtex

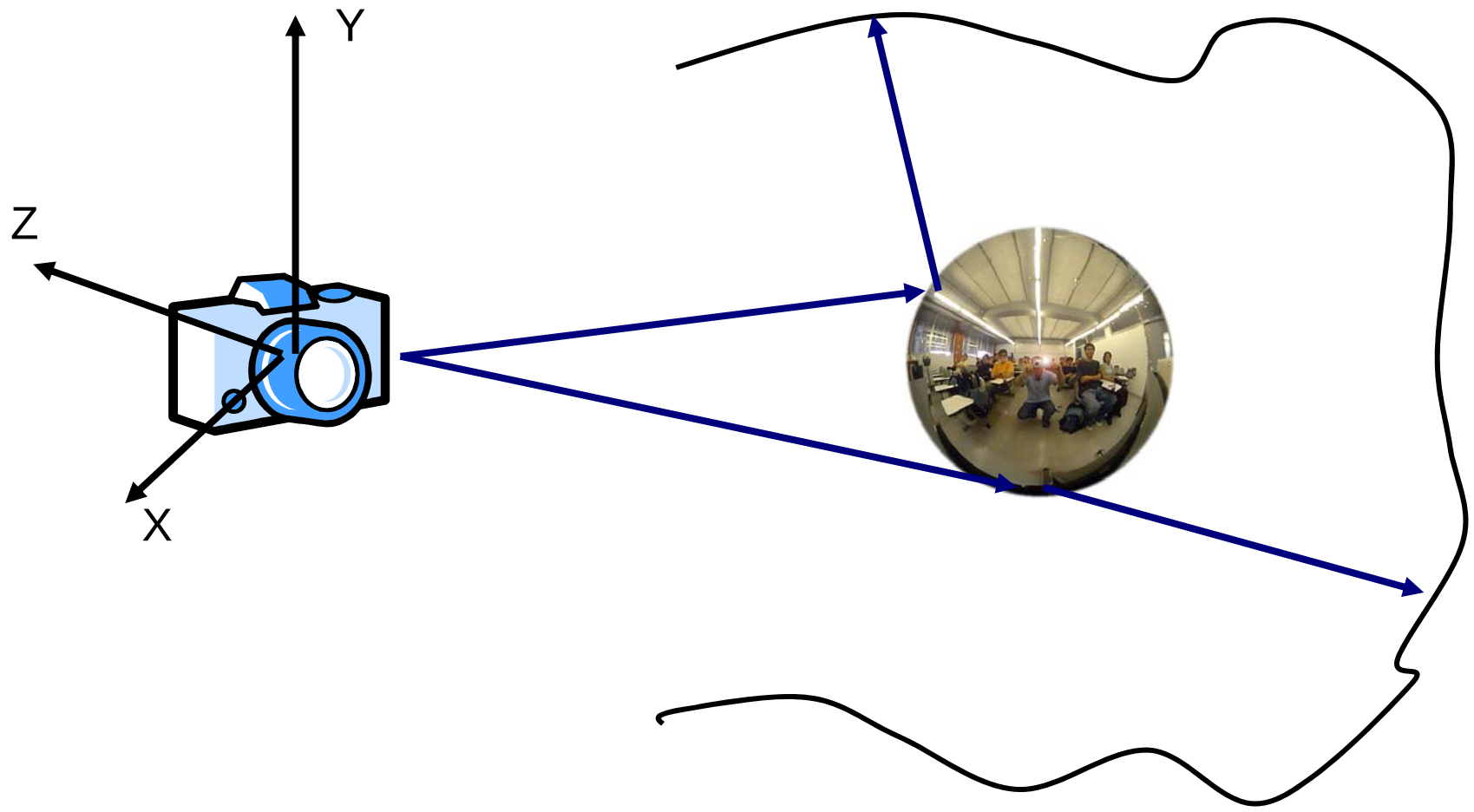




SPHERE MAPPING

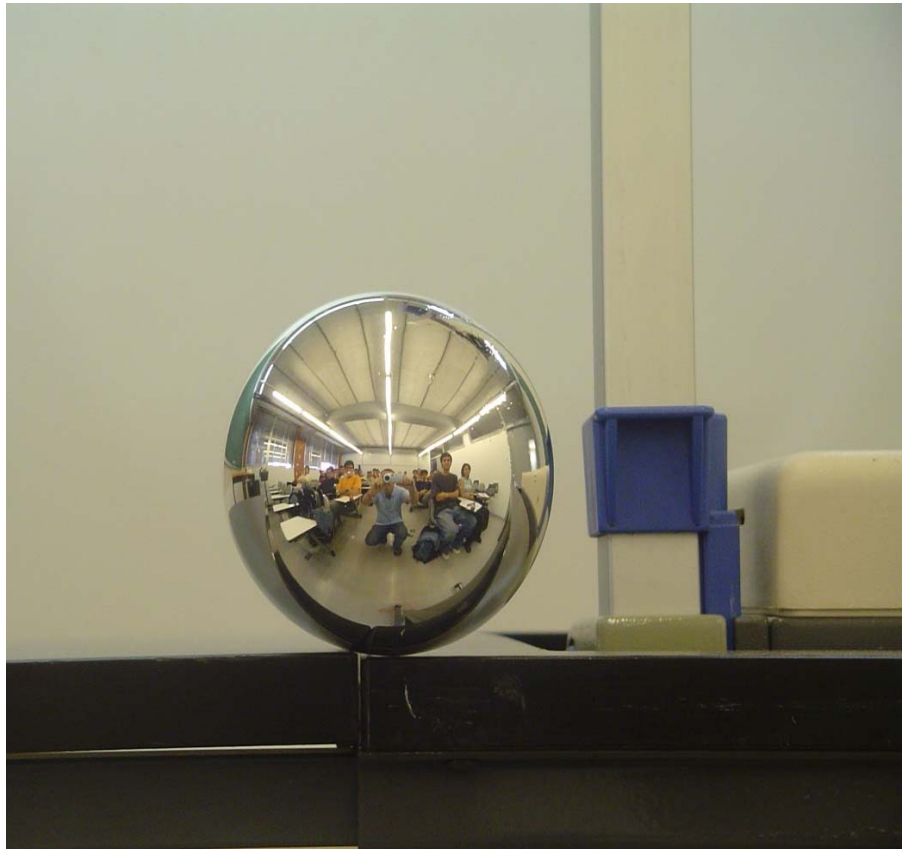
Sphere map

Construcció de sphere maps



Exemple sphere map

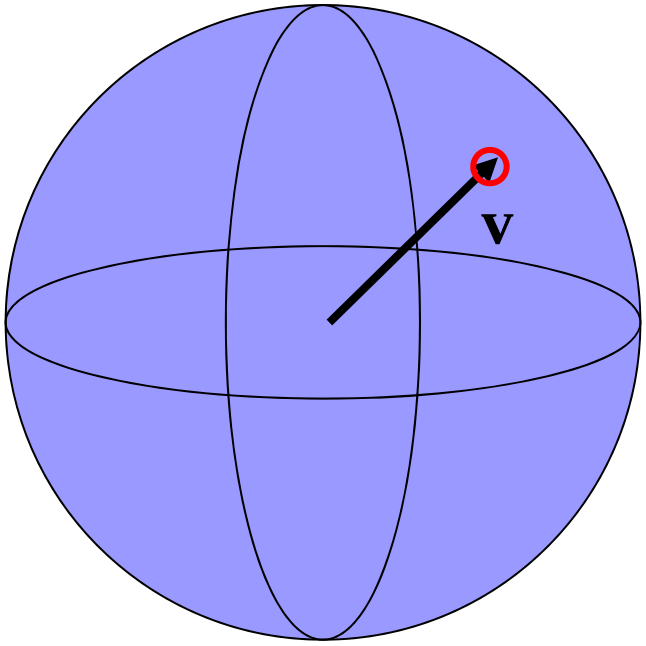
Sense retallar



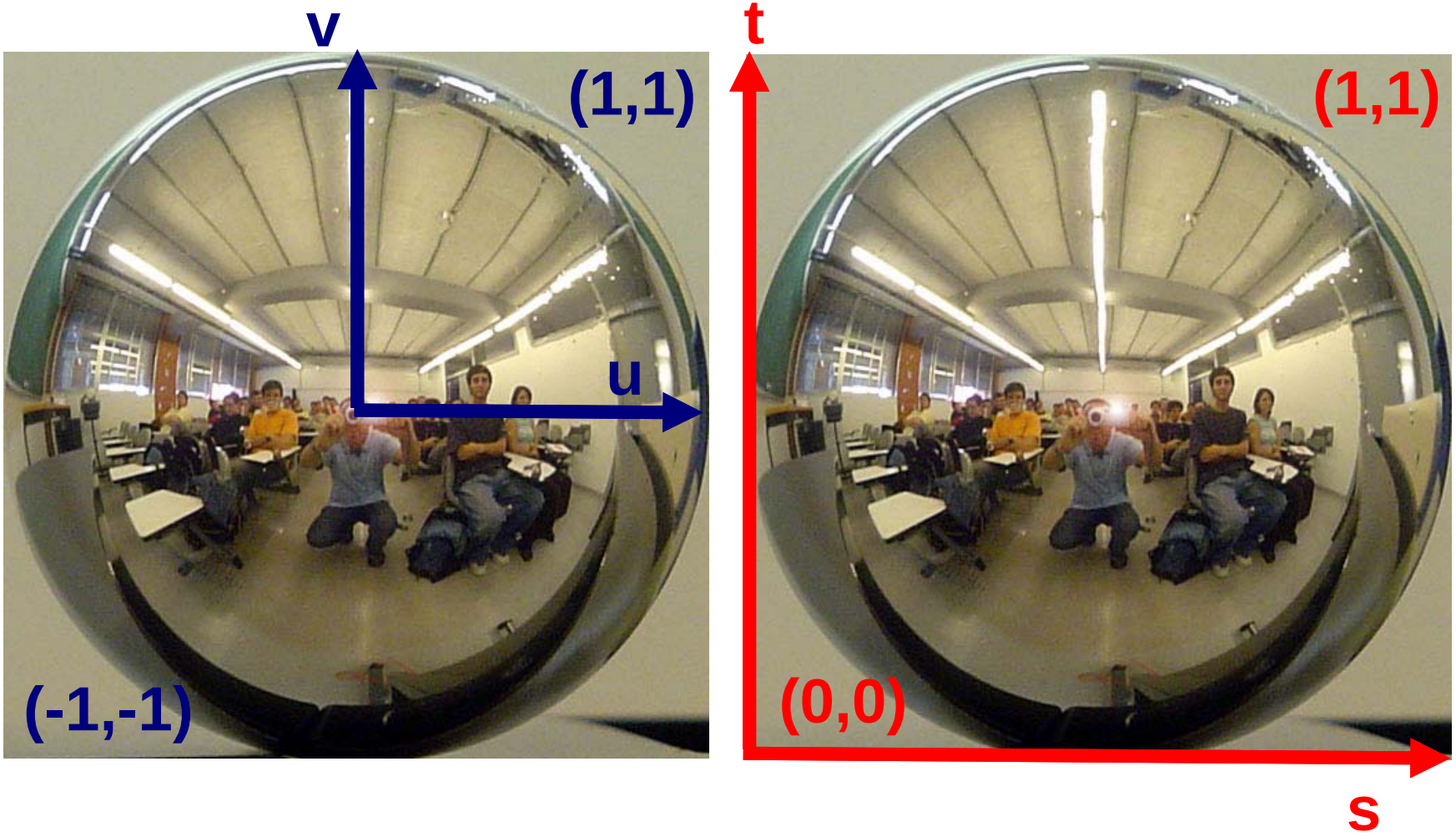
Retallat

Propietats

- De la textura només se n'aprofita el cercle inscrit
- Conté informació de aproximadament tot l'entorn (totes direccions)
- Distorsió considerable a prop de la vora del cercle

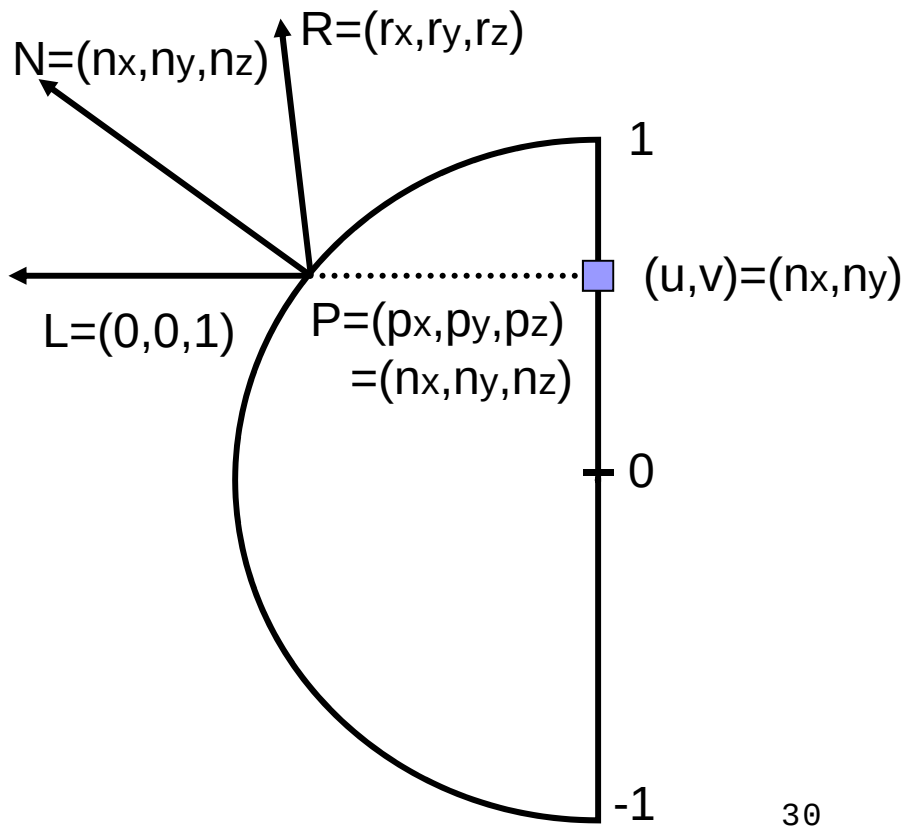


Coordenades (u,v) ←→(s,t)

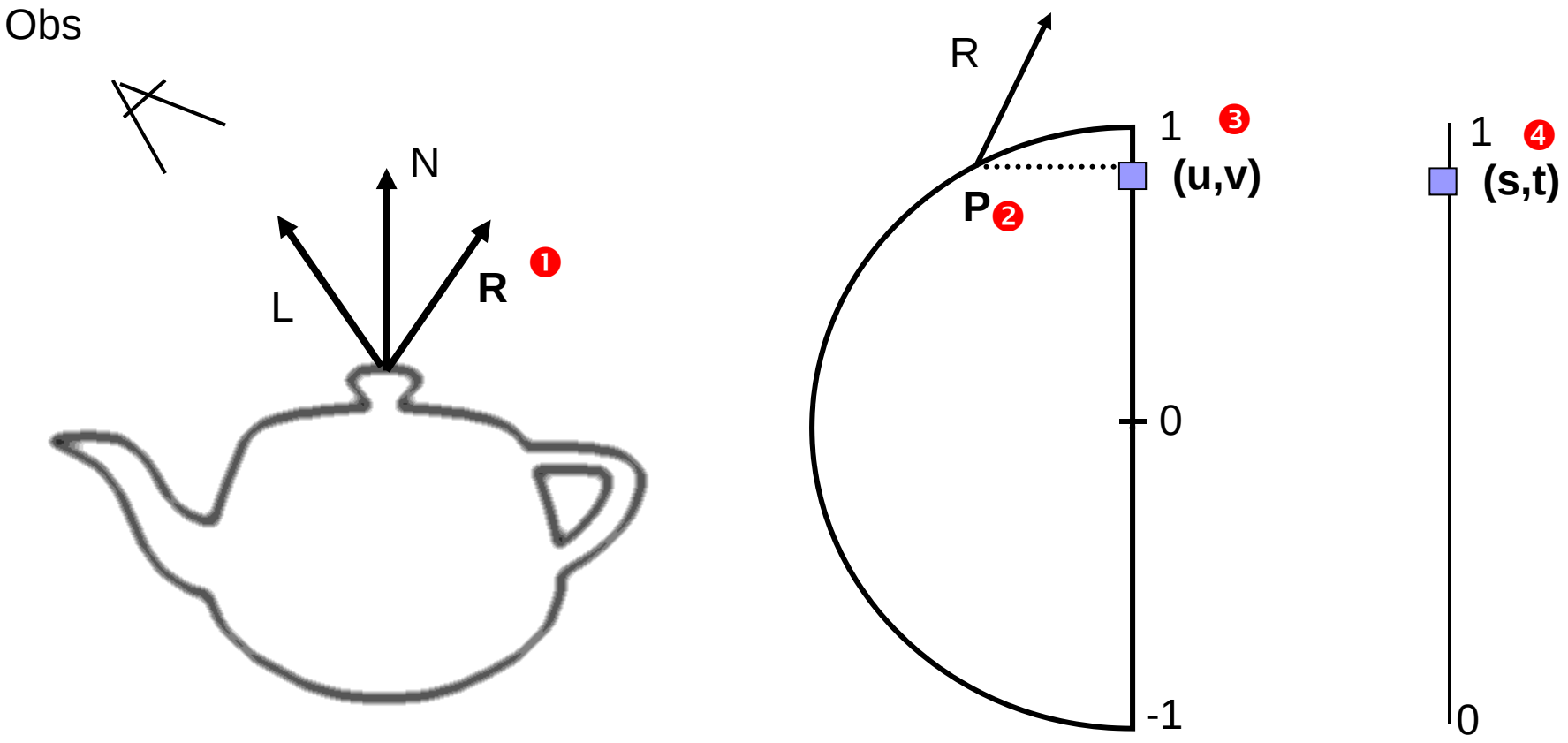


Relació entre vector R i (u,v)

$R=(2n_zn_x, 2n_zn_y, 2n_z^2-1)$



Càlcul coords de textura

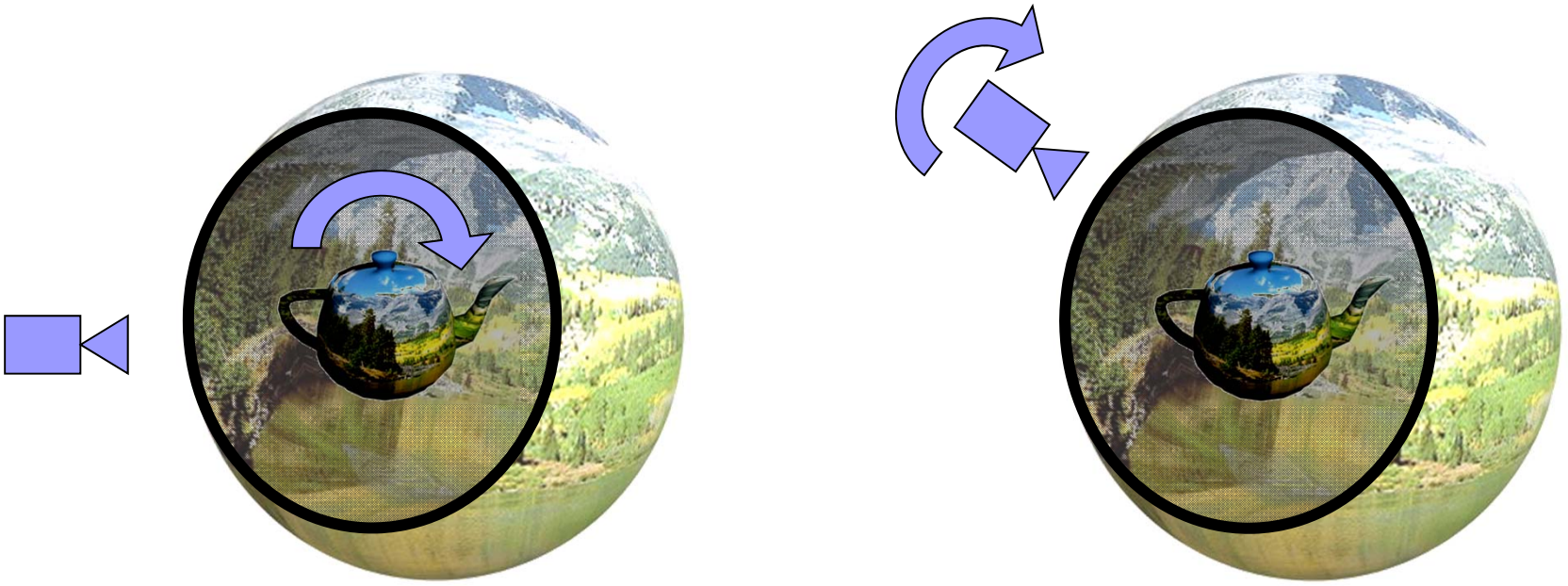


Càlcul del color donat R

```
vec4 sampleSphereMap(sampler2D sampler, vec3 R)
{
    float z = sqrt((R.z+1.0)/2.0);
    vec2 st=vec2((R.x/(2.0*z)+1.0)/2.0,(R.y/(2.0*z)+1.0)/2.0);
    return texture2D(sampler, st);
}
```

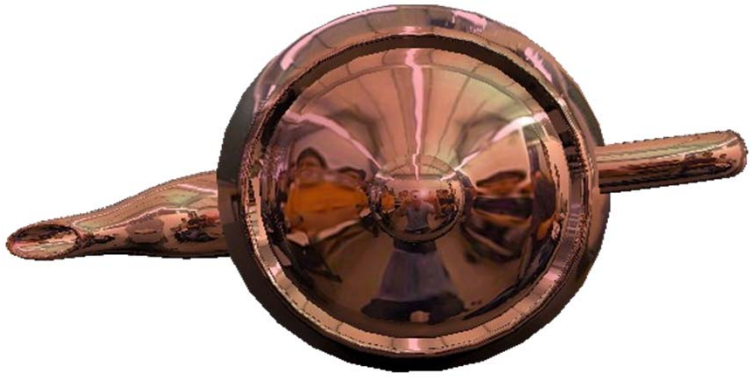
Eye/world coordinates

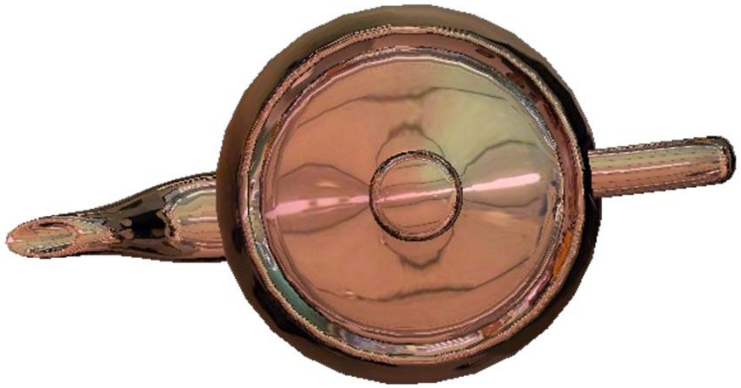
- Càlcul amb vèrtex, normal en eye coords
  - El entorn serà estàtic respecte la càmera
  - L'objecte sempre reflecteix "la mateixa part" de l'entorn
- Càlcul amb vèrtex, normal en world coords
  - El entorn serà dinàmic respecte la càmera
  - L'objecte reflecteix diferents parts de l'entorn



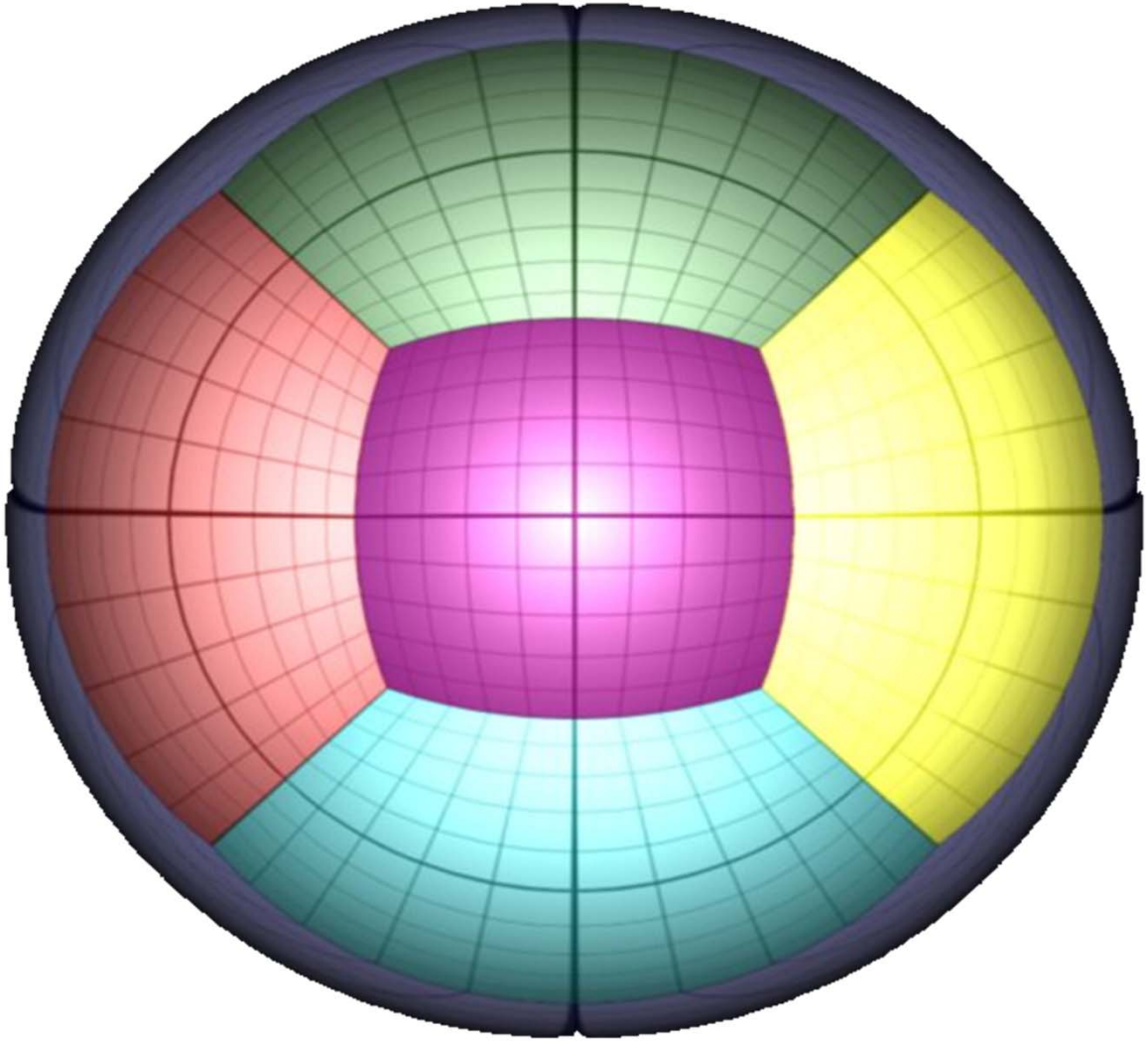
Eye/world coordinates

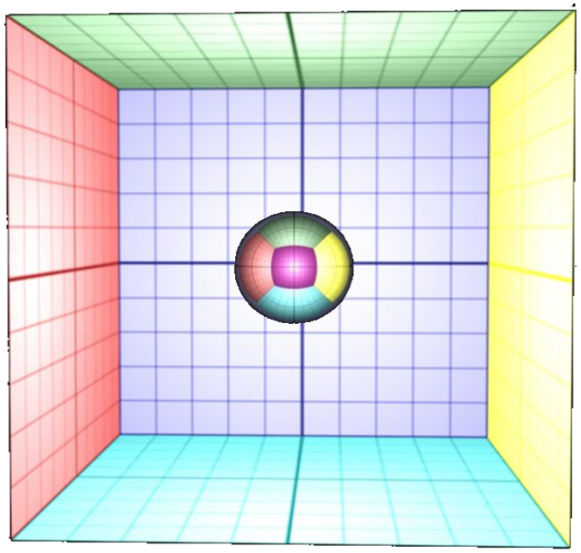
Eye/world coordinates

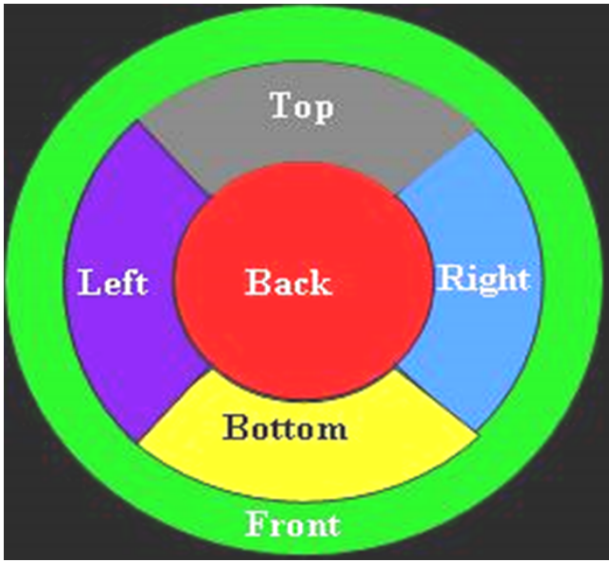




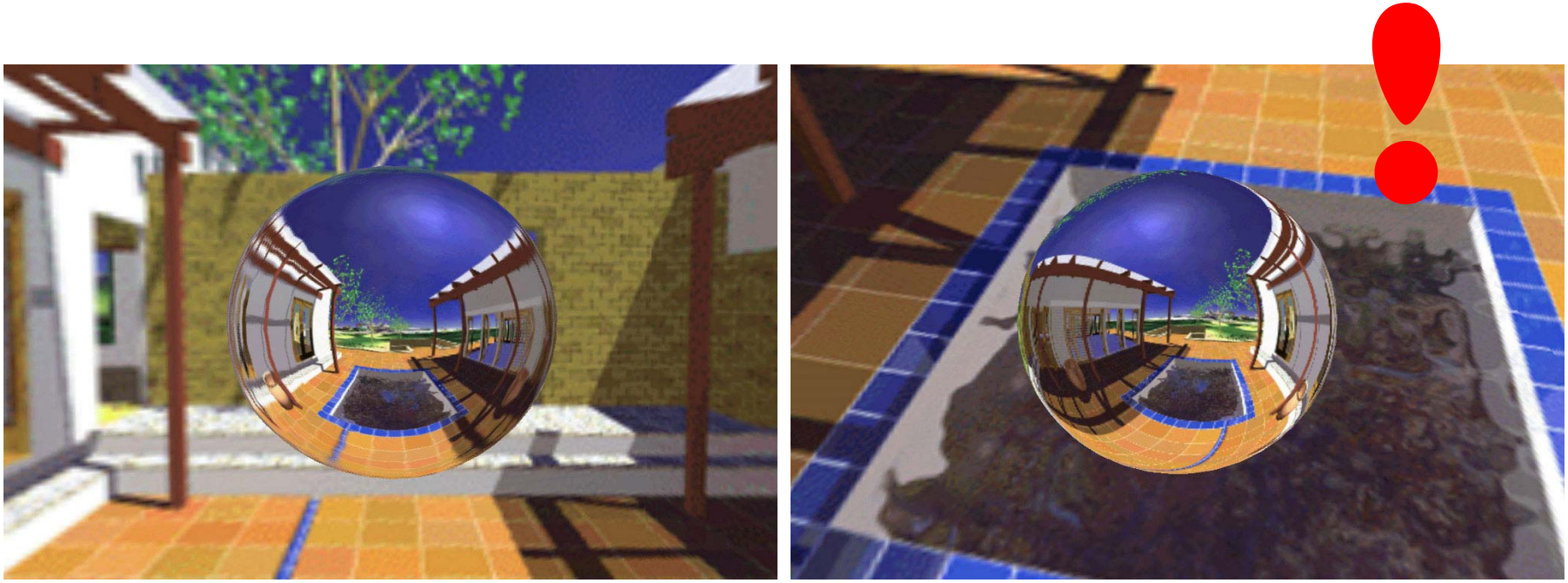
Sphere mapping: distorsió







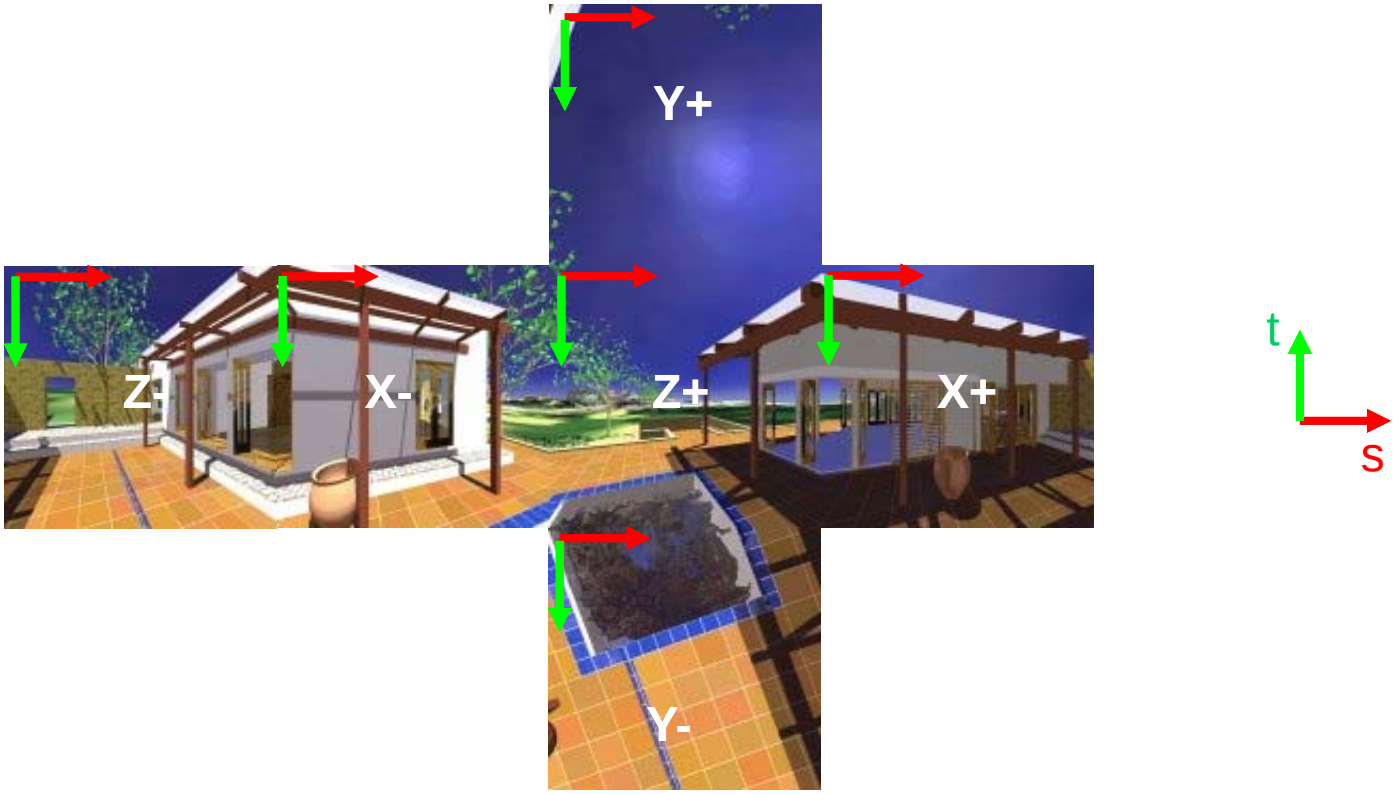
Limitacions sphere mapping



CUBE MAPPING

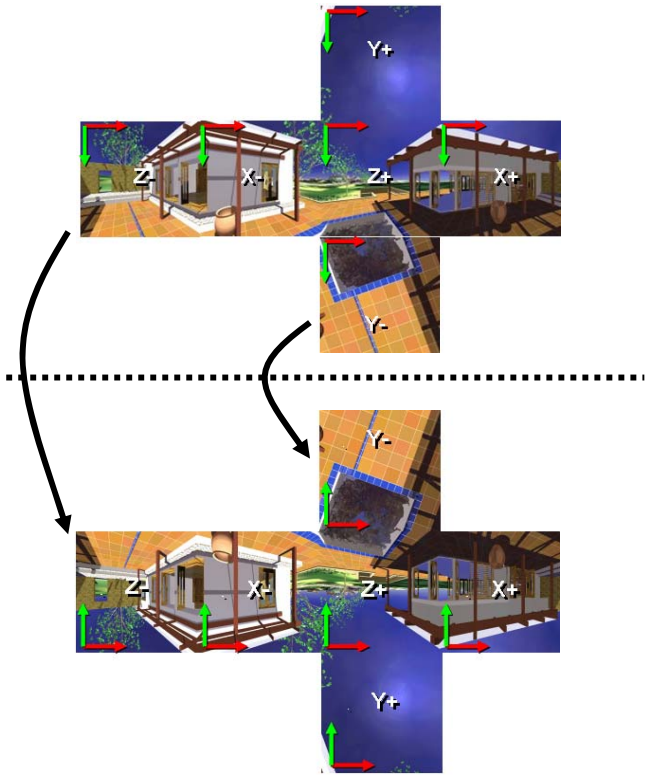
Cube mapping: orientació textures

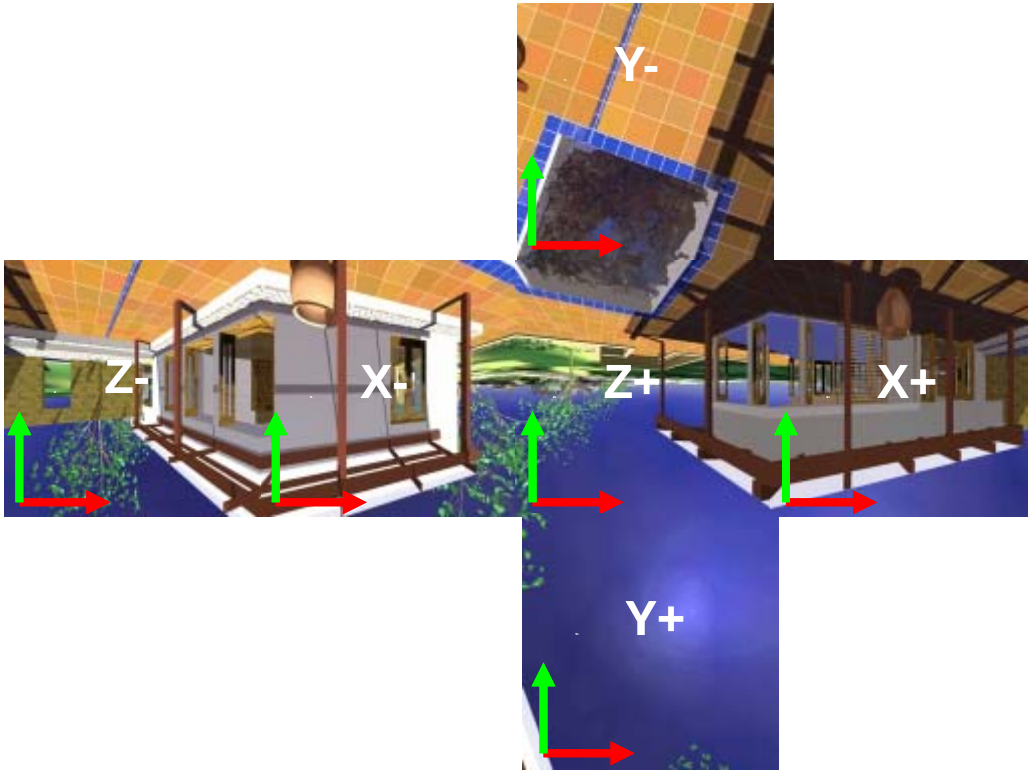
Així estan orientats els eixos (s,t) respecte cada imatge:



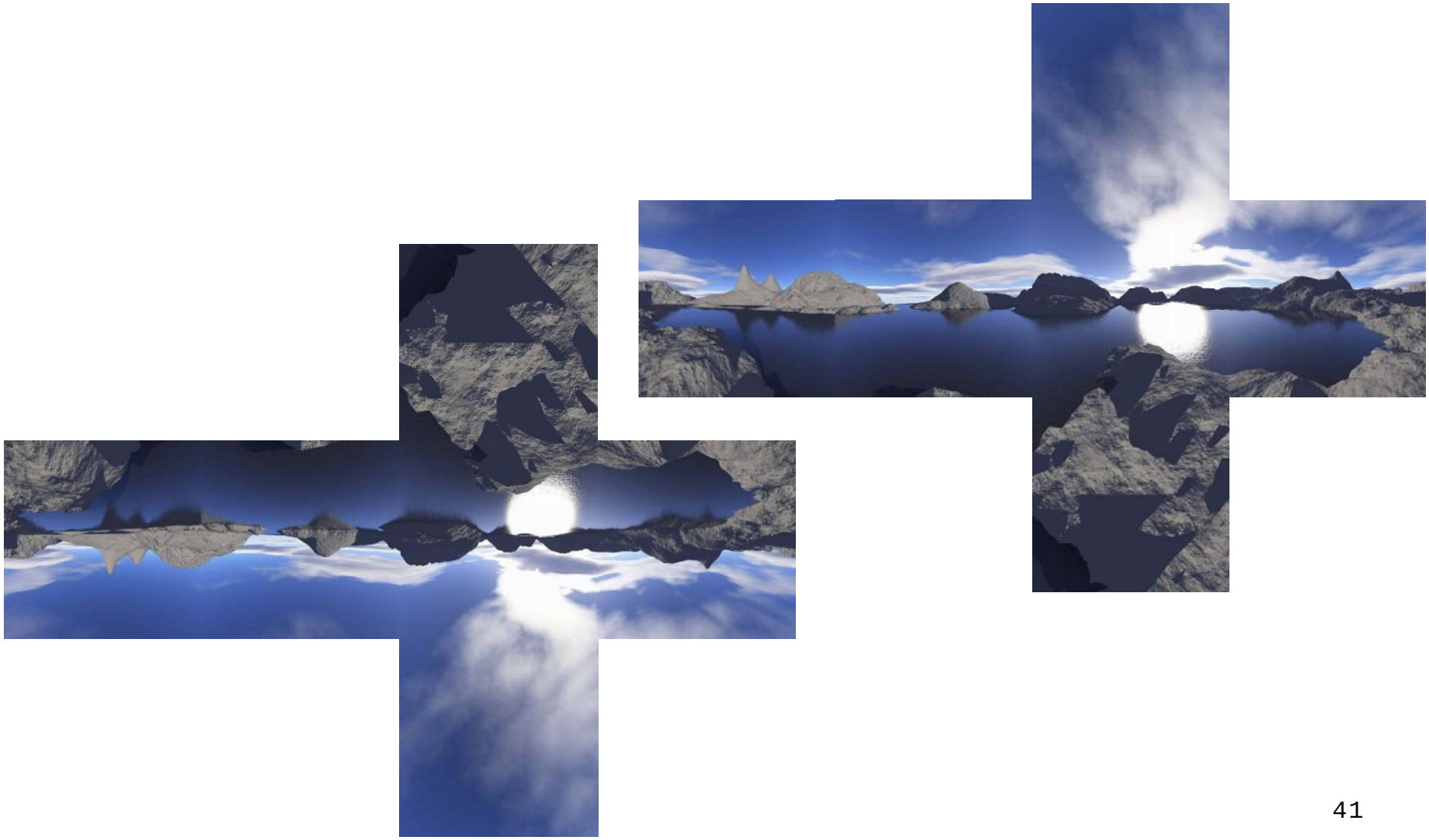
Cube mapping: orientació textures

- Per tant aquestes són les sis textures que cal definir (origen de les coords (s,t) al pixel inferior esquerra):



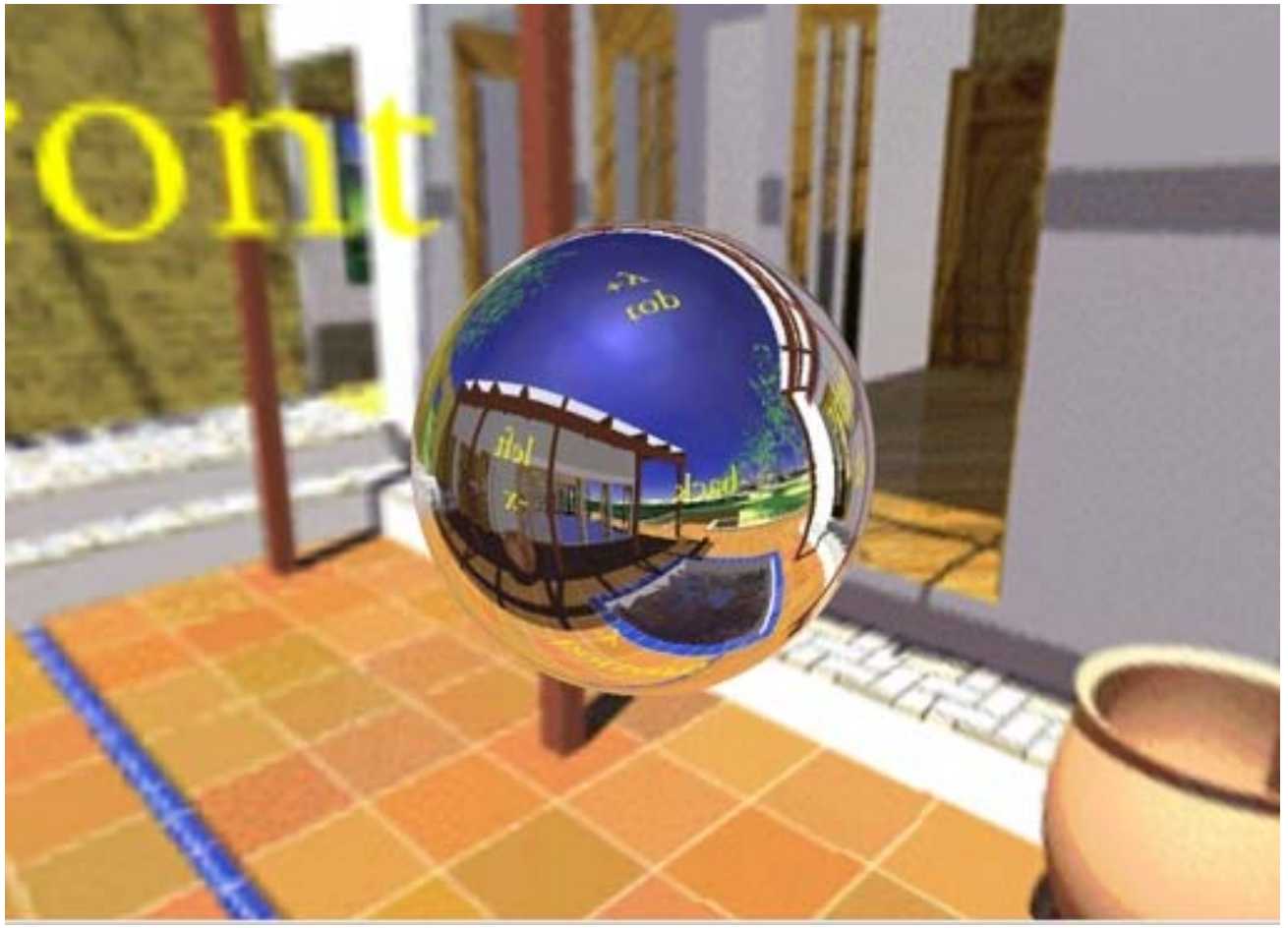


Cube mapping: orientació textures

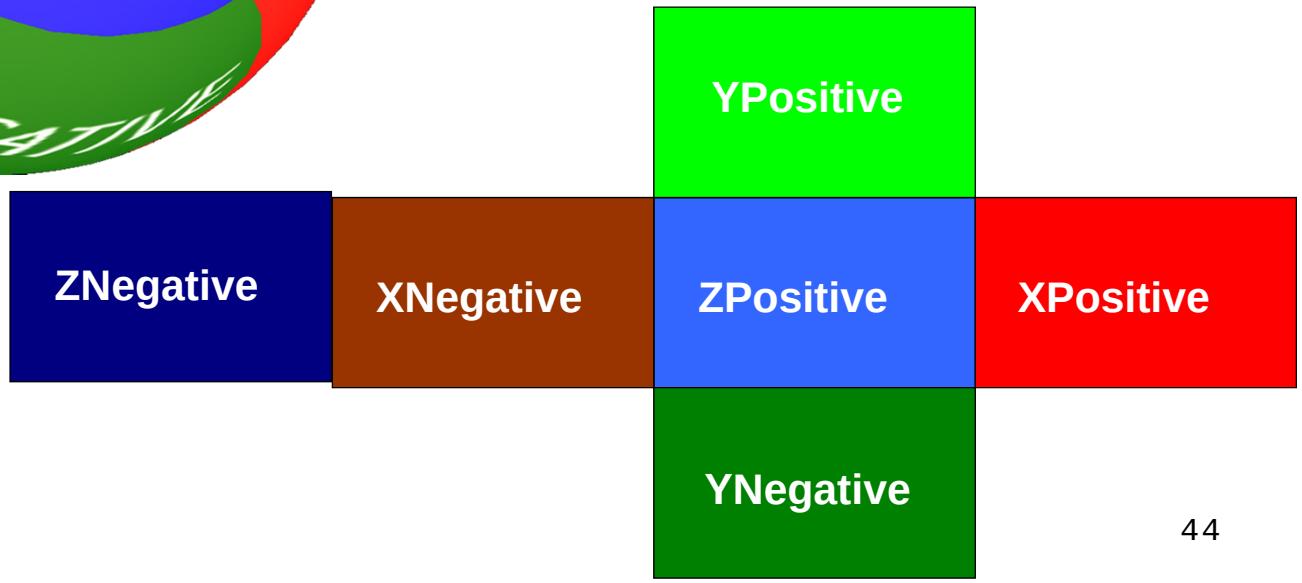


Cube mapping: orientació textures

Cube mapping: exemple



Cube mapping: exemple



Cube mapping: Pas R → (s,t)

dir	target	a	b	max
+rx	POSITIVE_X	-rz	-ry	rx
-rx	NEGATIVE_X	+rz	-ry	rx
+ry	POSITIVE_Y	+rx	+rz	ry
-ry	NEGATIVE_Y	+rx	-rz	ry
+rz	POSITIVE_Z	+rx	-ry	rz
-rz	NEGATIVE_Z	-rx	-ry	rz

$u = a/max$

$v = b/max$

$s = (u+1)/2$

$t = (v+1)/2$

Cube mapping: exemple

// 1. Creació de les sis textures

glTexImage2D(GL_TEXTURE_CUBE_MAP_**POSITIVE_X**_EXT, …);
glTexImage2D(GL_TEXTURE_CUBE_MAP_**NEGATIVE_X**_EXT, …);
glTexImage2D(GL_TEXTURE_CUBE_MAP_**POSITIVE_Y**_EXT, …);
glTexImage2D(GL_TEXTURE_CUBE_MAP_**NEGATIVE_Y**_EXT, …);
glTexImage2D(GL_TEXTURE_CUBE_MAP_**POSITIVE_Z**_EXT, …);
glTexImage2D(GL_TEXTURE_CUBE_MAP_**NEGATIVE_Z**_EXT, …);

Cube mapping: exemple

```
// 1. Creació de les sis textures
glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_X_EXT, …);
glTexImage2D(GL_TEXTURE_CUBE_MAP_NEGATIVE_X_EXT, …);
…
// 2. Activació Cube mapping
glEnable(GL_TEXTURE_CUBE_MAP_EXT);

// 3. Generació de coordenades de textura
glTexGenfv(GL_S, GL_TEXTURE_GEN_MODE,GL_REFLECTION_MAP_EXT);
glTexGenfv(GL_T, GL_TEXTURE_GEN_MODE,GL_REFLECTION_MAP_EXT);
glTexGenfv(GL_R, GL_TEXTURE_GEN_MODE,GL_REFLECTION_MAP_EXT);
glEnable(GL_TEXTURE_GEN_S);
glEnable(GL_TEXTURE_GEN_T);
glEnable(GL_TEXTURE_GEN_R);
```

Cube mapping: GLSL

```
uniform sampler2D sampler;
…
vec2 st = gl_TexCoord[0].st;
vec4 color = texture2D(sampler, st);
```

```
uniform samplerCube samplerC;
…
vec3 R;
…
vec4 color = textureCube(samplerC, R);
```