

Reflexions especulars

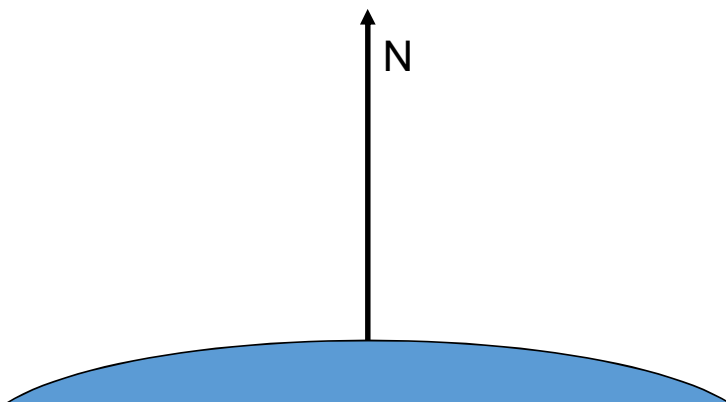
Carlos Andújar

Nov 2022

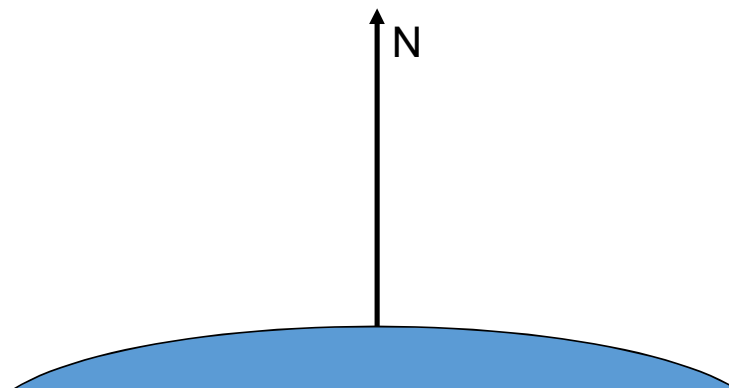
Introducció



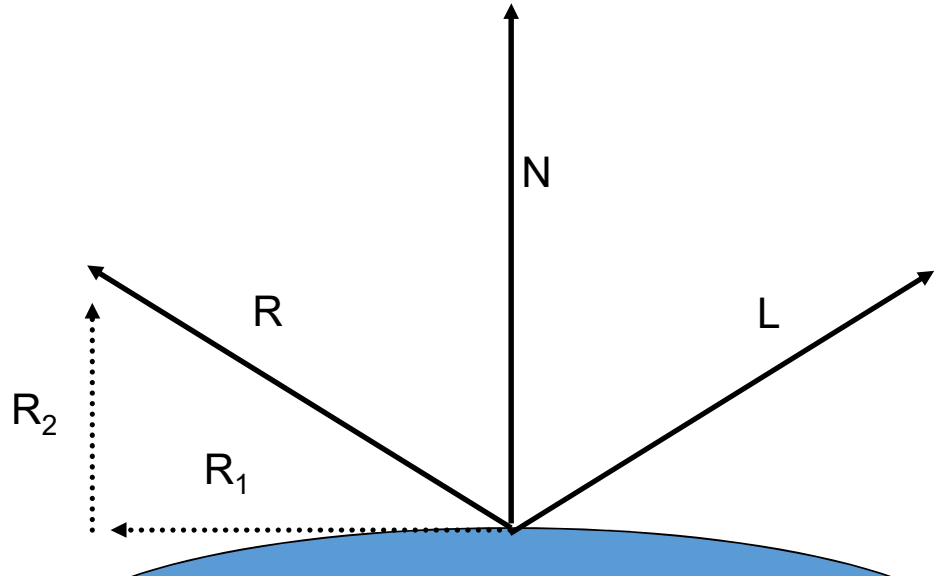
Reflexió difosa



Reflexió especular



Vector reflectit



Vector reflectit

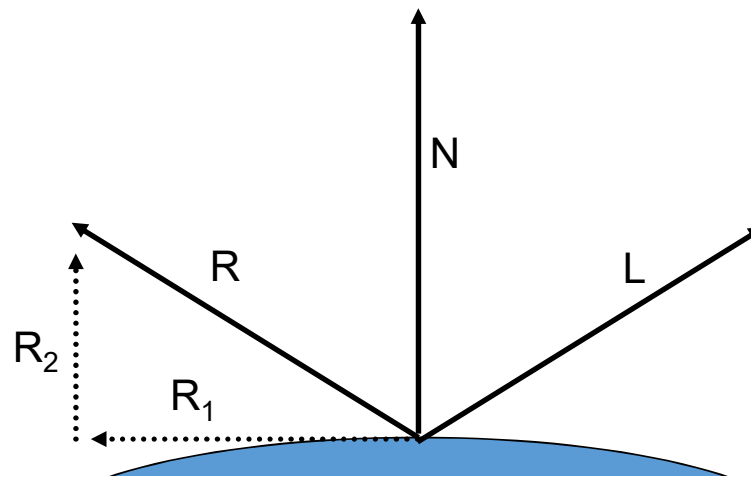
$$R = R_1 + R_2$$

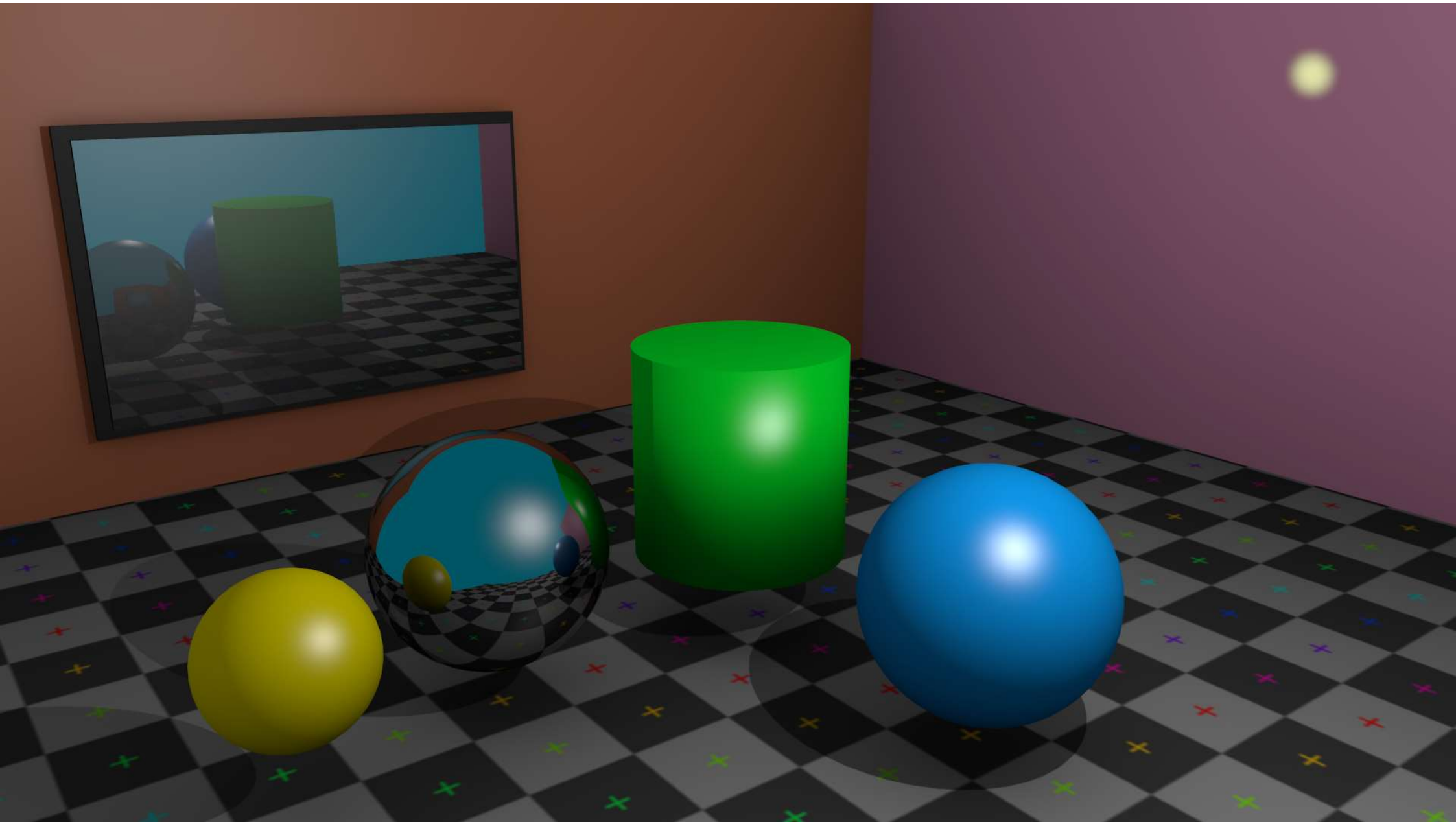
$$R_1 = -L + R_2$$

$$R = 2R_2 - L$$

$$R_2 = (N \cdot L)N$$

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





Mètodes

- Ray-tracing
- Reflexions basada en **objectes virtuals**
 - Modelats
 - Reflectits (sense stencil test)
 - Reflectits (amb stencil test)
 - Textures dinàmiques
- **Environment mapping**
 - Sphere mapping
 - Cube mapping

Reflexions amb objectes virtuals

Mètodes

- Ray-tracing
- Reflexions basada en **objectes virtuals**
 - Modelats
 - Reflectits (sense stencil test)
 - Reflectits (amb stencil test)
 - Textures dinàmiques
- **Environment mapping**
 - Sphere mapping
 - Cube mapping

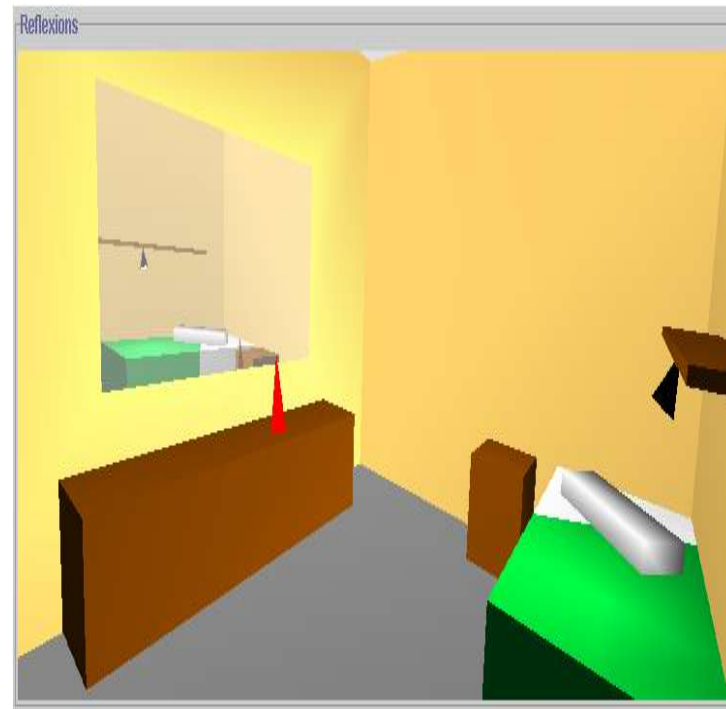
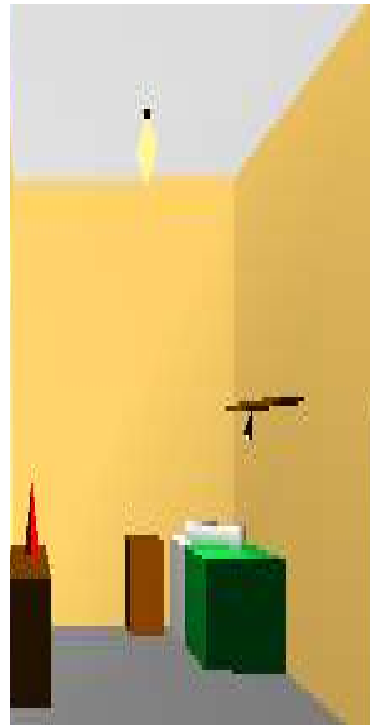
Objectes virtuals



Mètodes

- Ray-tracing
- Reflexions basada en **objectes virtuals**
 - Modelats
 - Reflectits (sense stencil test)
 - Reflectits (amb stencil test)
 - Textures dinàmiques
- **Environment mapping**
 - Sphere mapping
 - Cube mapping

Objets virtuels



Mètodes

- Ray-tracing
- Reflexions basada en **objectes virtuals**
 - Modelats
 - Reflectits (sense stencil test)
 - Reflectits (amb stencil test)
 - Textures dinàmiques
- **Environment mapping**
 - Sphere mapping
 - Cube mapping

Algorisme (versió 1)

// 1. Dibuixar els objectes en posició virtual

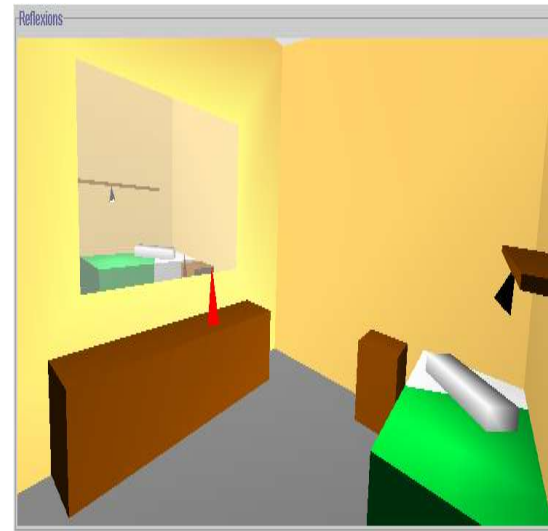
```
glPushMatrix();  
glMultMatrix(matriu_simetria)  
glLightfv(GL_LIGHT0, GL_POSITION, pos);  
glCullFace(GL_FRONT);  
dibuixar(escena);  
glPopMatrix();
```

// 2. Dibuixar el mirall semi-transparent

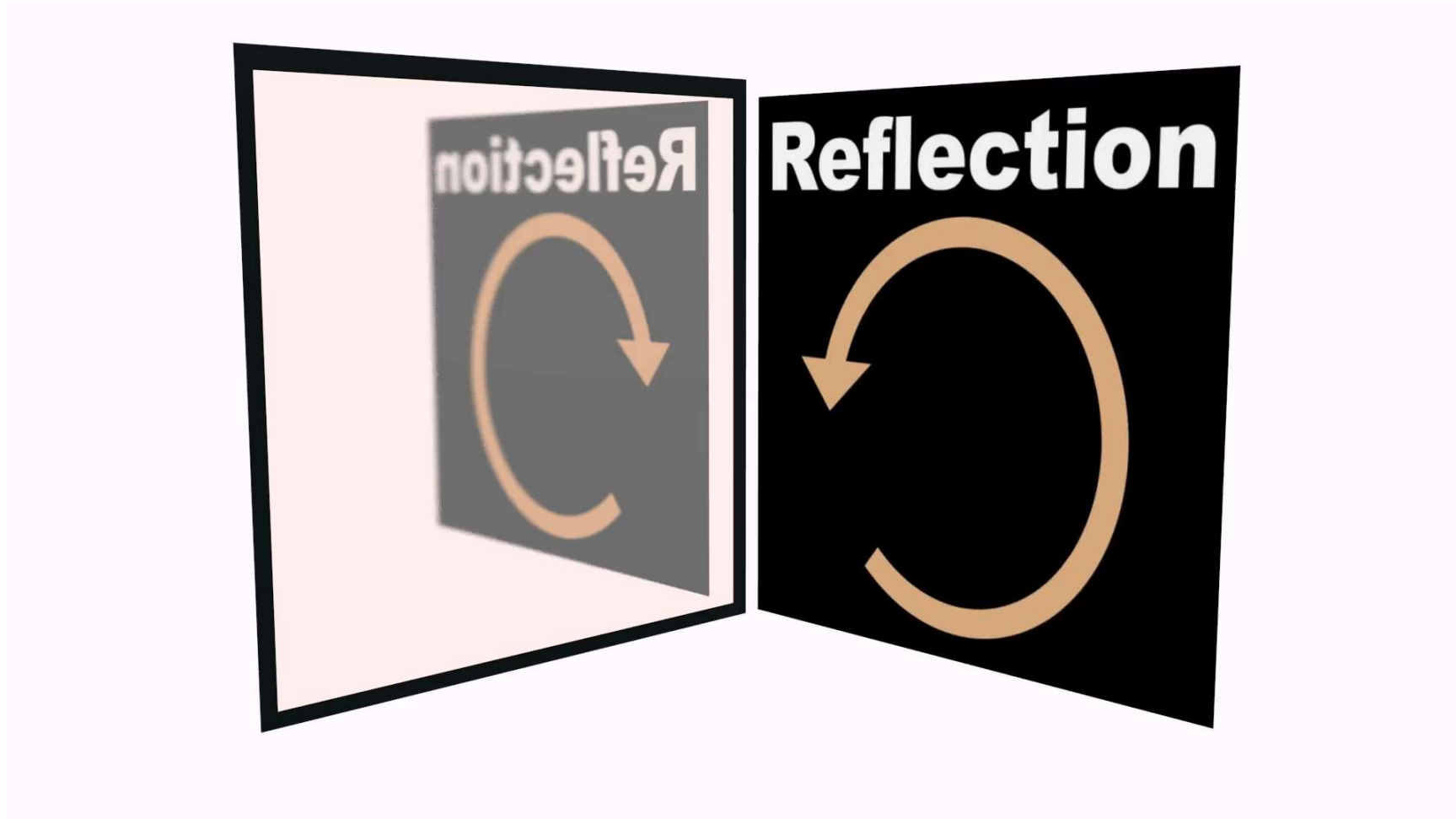
```
glLightfv(GL_LIGHT0, GL_POSITION, pos);  
glCullFace(GL_BACK);  
dibuixar(mirall);
```

// 3. Dibuixar els objectes en posició real

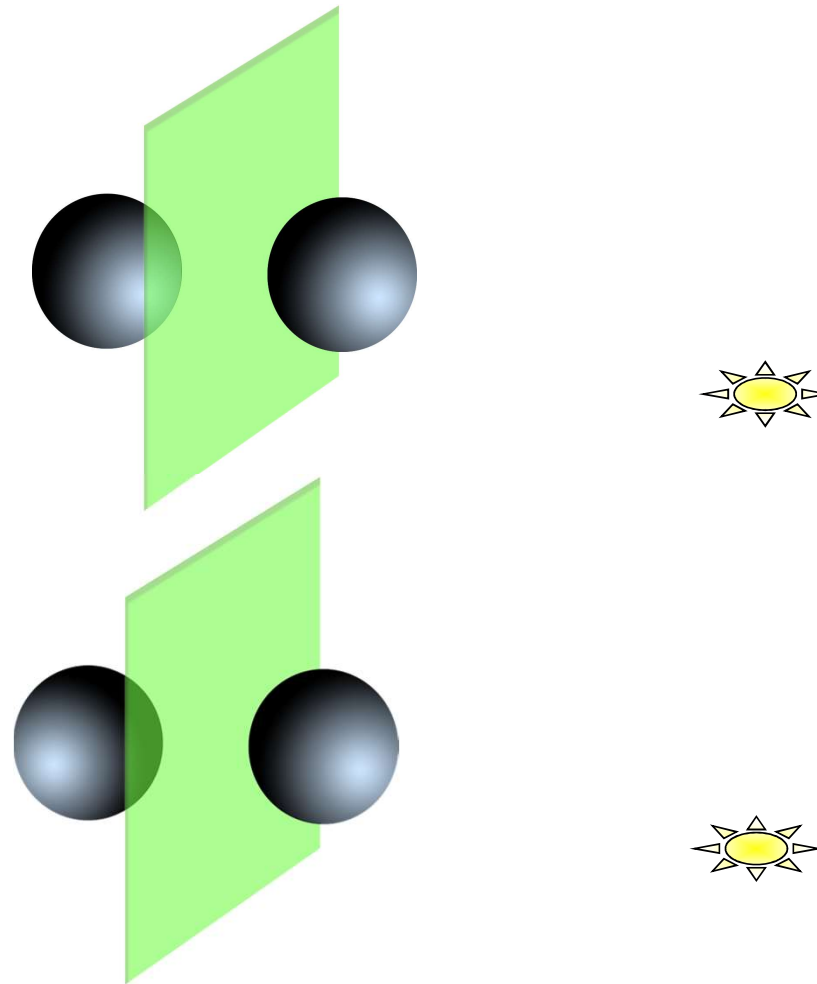
```
dibuixar(escena);
```



Inversió ordre

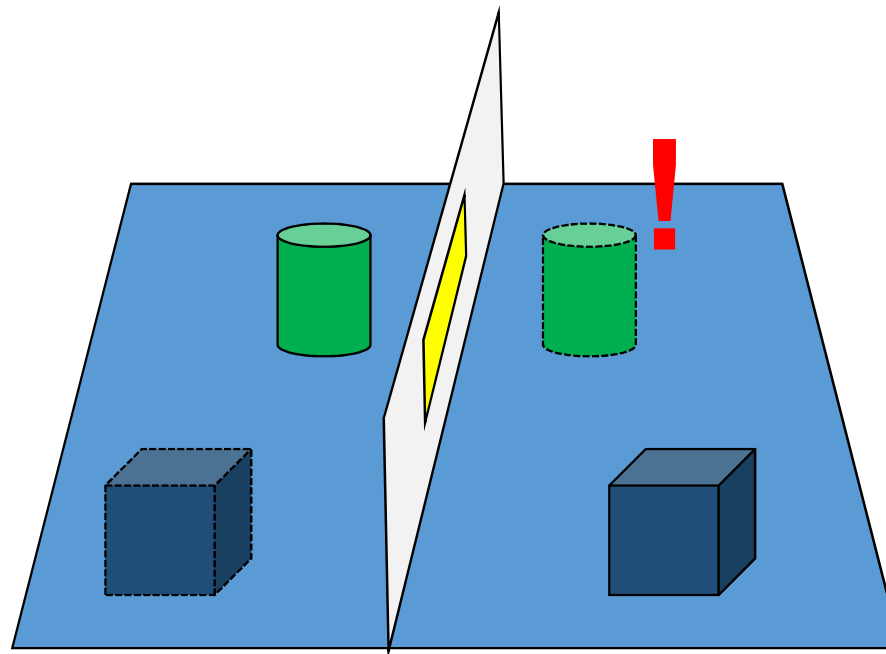


Reflexió de les fonts de llum



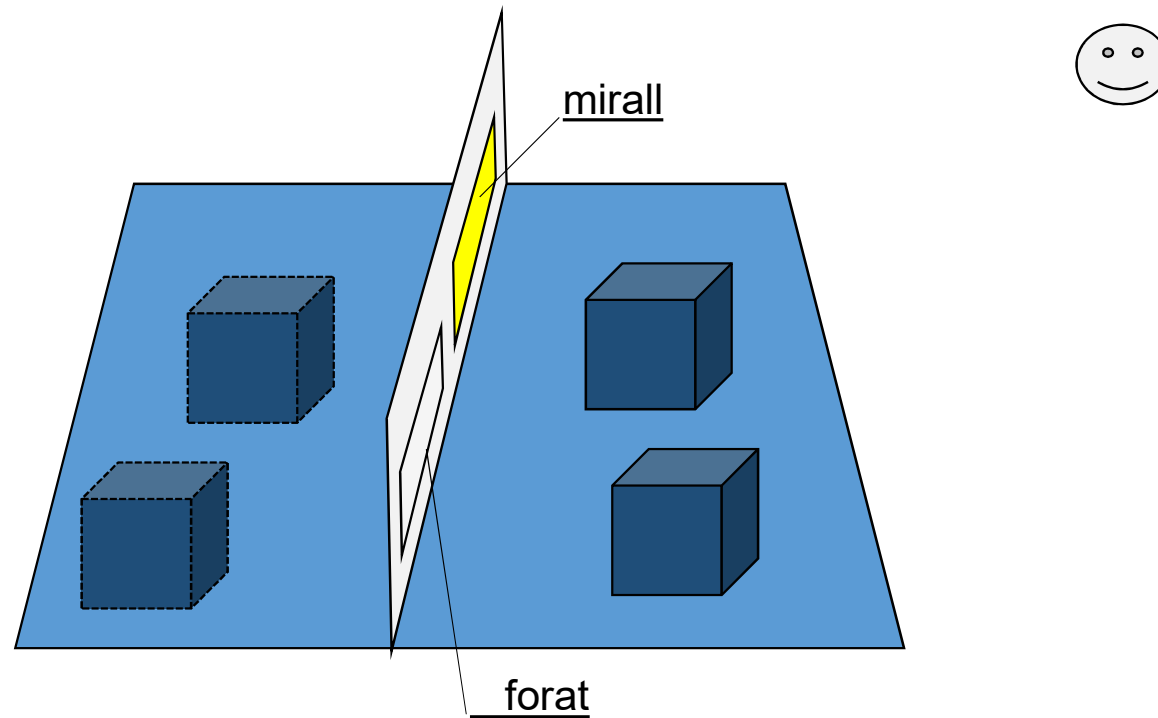
Limitacions

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



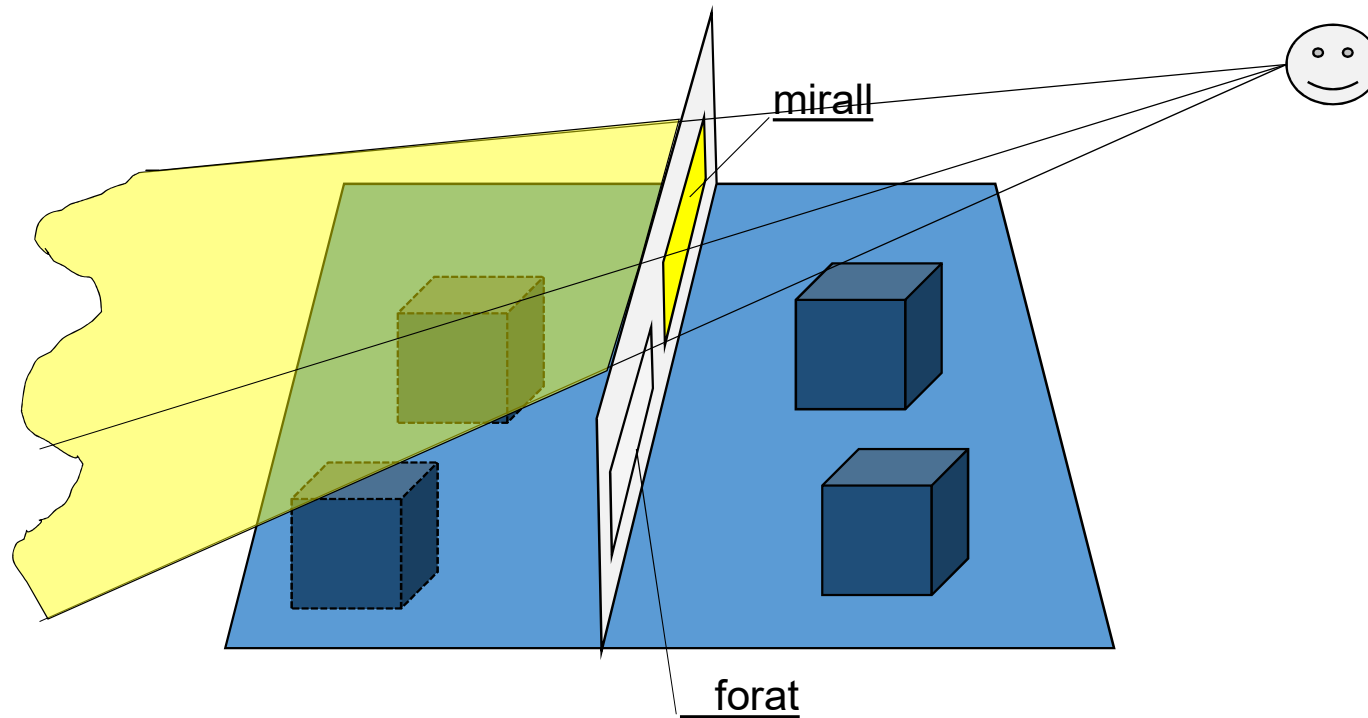
Limitacions

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

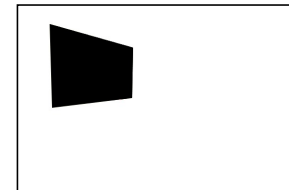
Usar stencil per limitar els objectes virtuals a la regió ocupada pel mirall.

Mètodes

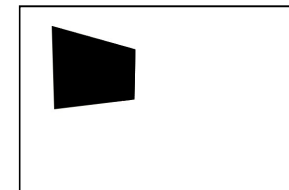
- Ray-tracing
- Reflexions basada en **objectes virtuals**
 - Modelats
 - Reflectits (sense stencil test)
 - Reflectits (amb stencil test)
 - Textures dinàmiques
- **Environment mapping**
 - Sphere mapping
 - Cube mapping

Algorisme (versió 2)

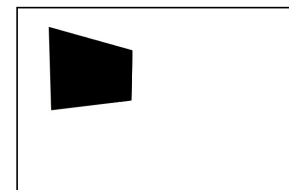
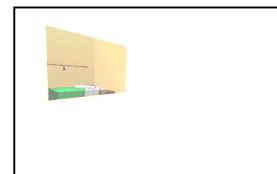
Pas 1. Dibuixar mirall al stencil buffer



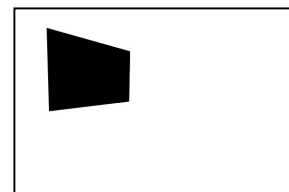
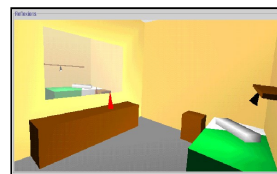
Pas 2. Dibuixar objectes en pos virtual



Pas 3. Dibuixar mirall semi-transparent



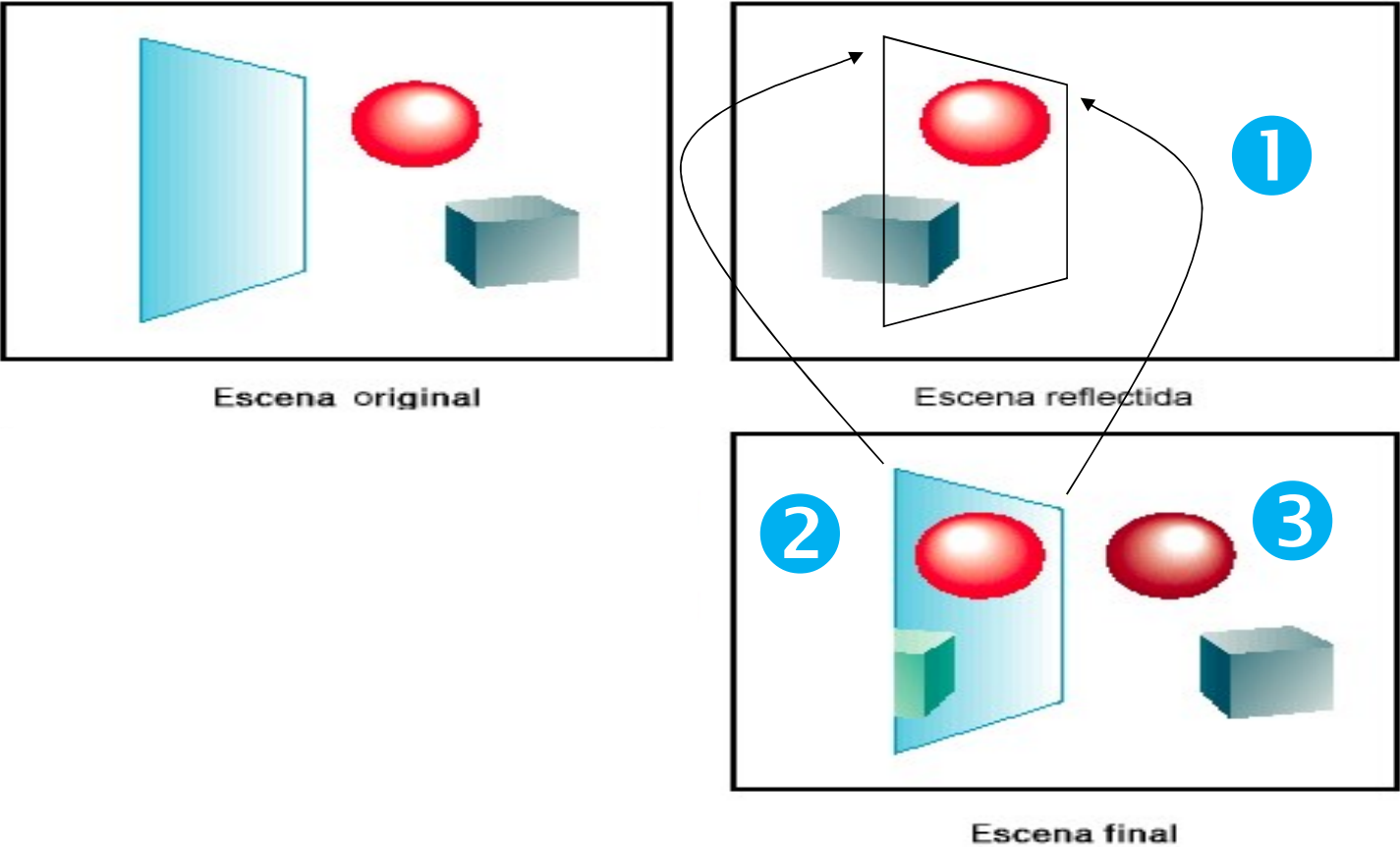
Pas 4. Dibuixar objectes en pos real



Mètodes

- Ray-tracing
- Reflexions basada en **objectes virtuals**
 - Modelats
 - Reflectits (sense stencil test)
 - Reflectits (amb stencil test)
 - Textures dinàmiques
- **Environment mapping**
 - Sphere mapping
 - Cube mapping

Textures dinàmiques



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

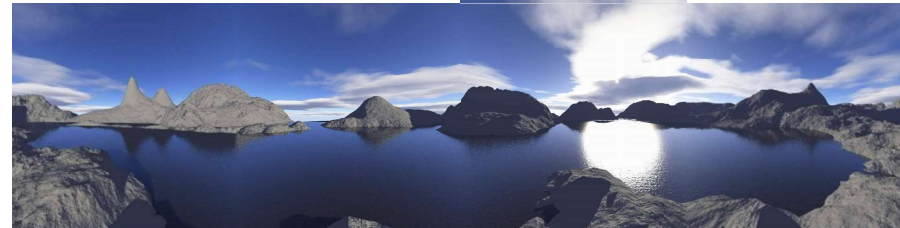
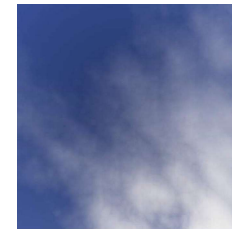
Environment map

Donat una direcció arbitrària R , ens retorna el color de l'entorn en direcció R

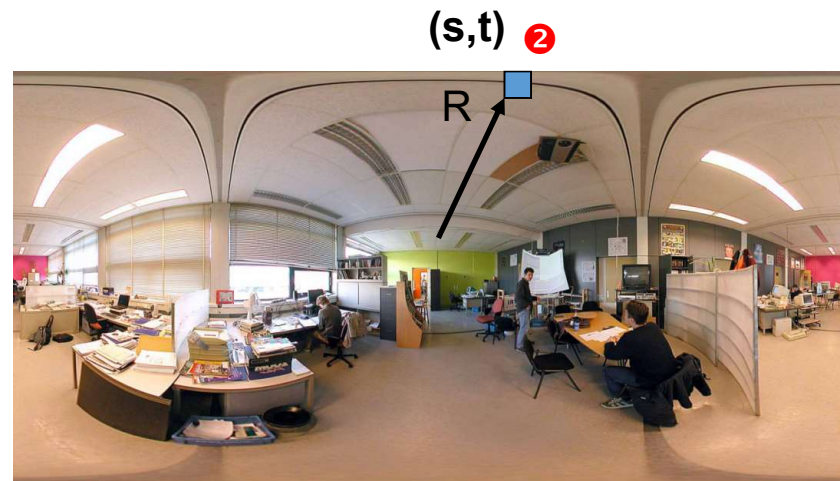
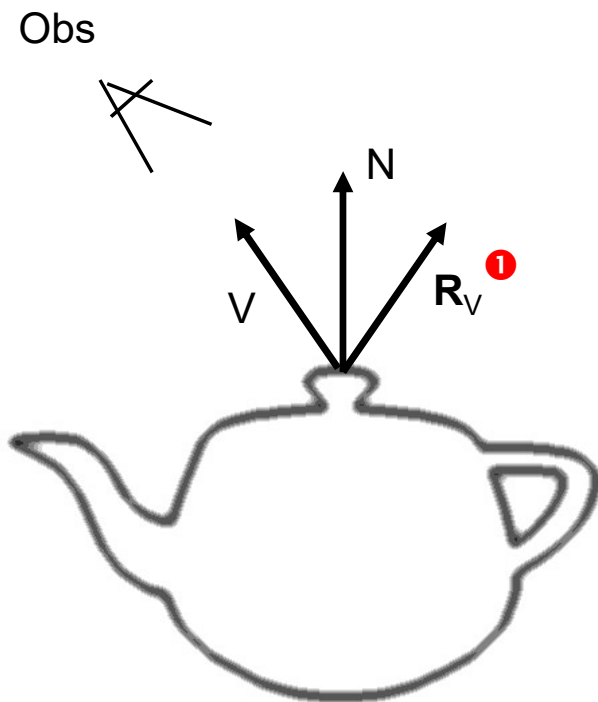
$\text{color} = \text{environmentMap}(R)$



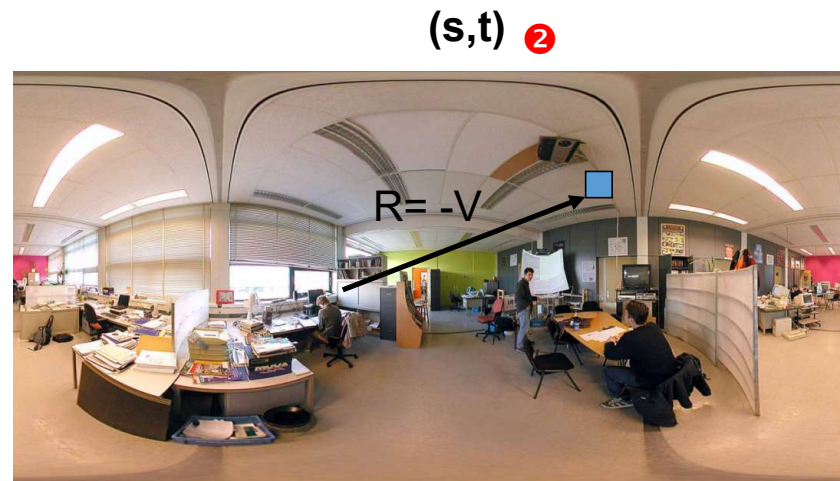
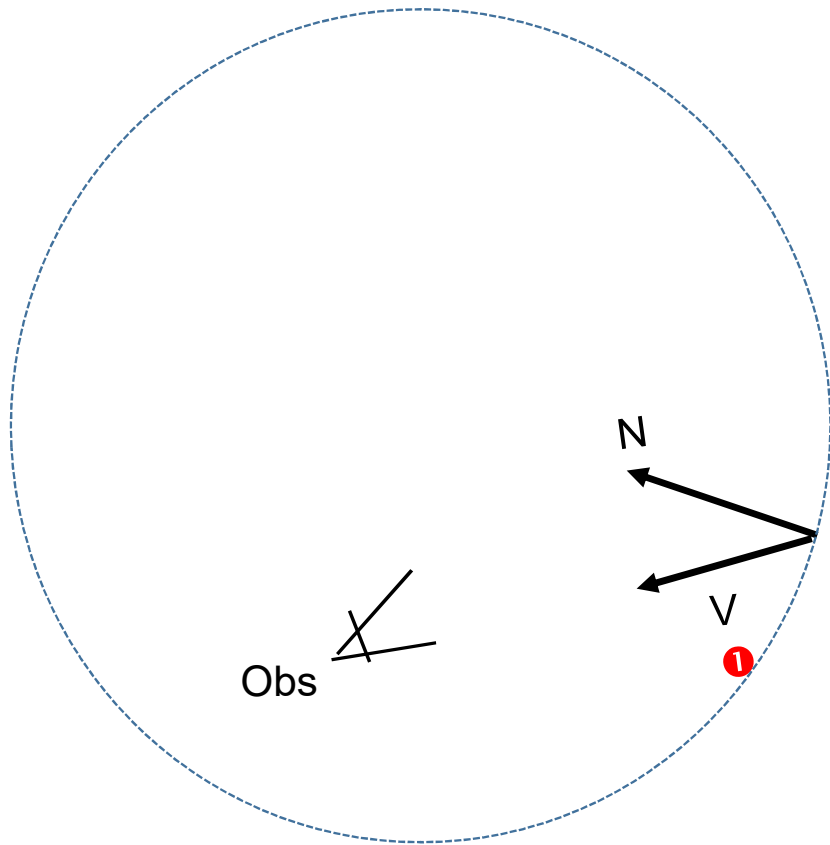
Representació com a textura



Ús per reflexions especulars



Ús com a entorn (background)



Sphere mapping

Sphere map

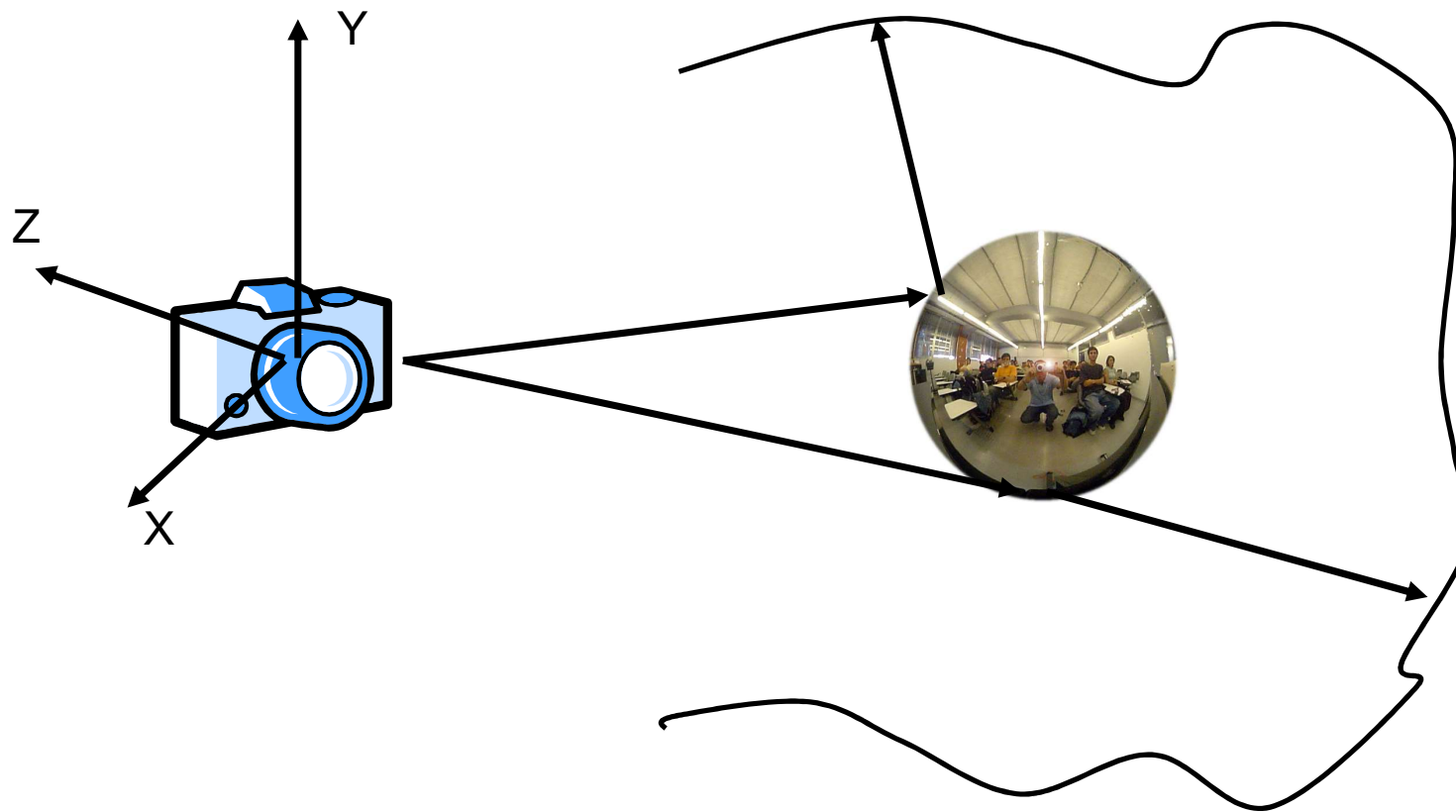


Sphere map

Hand with Reflecting Sphere by **M. C. Escher**.
Lithograph, 1935.
Official M.C. Escher website.

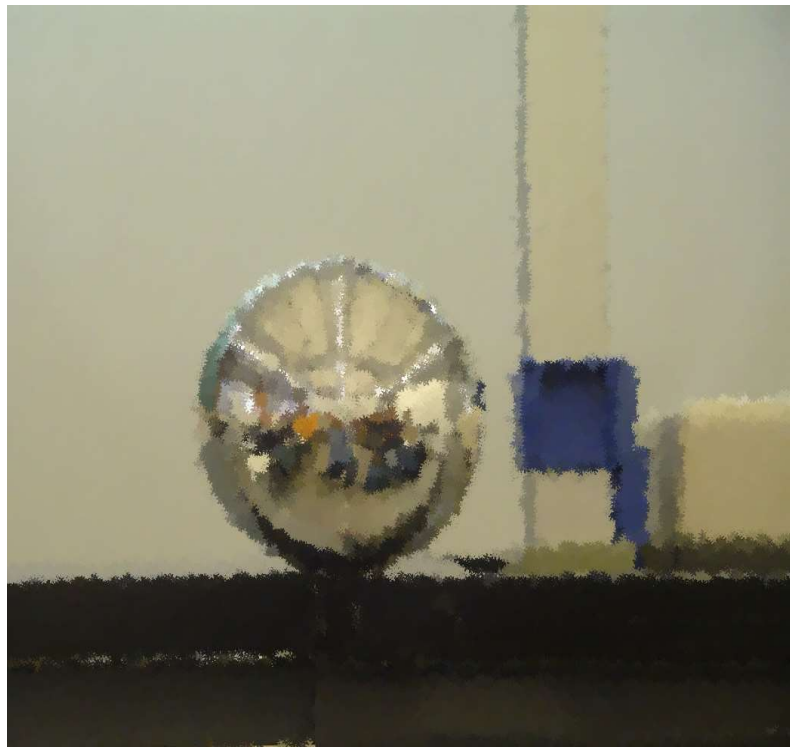


Construcció de sphere maps



Exemple sphere map

Sense retallar

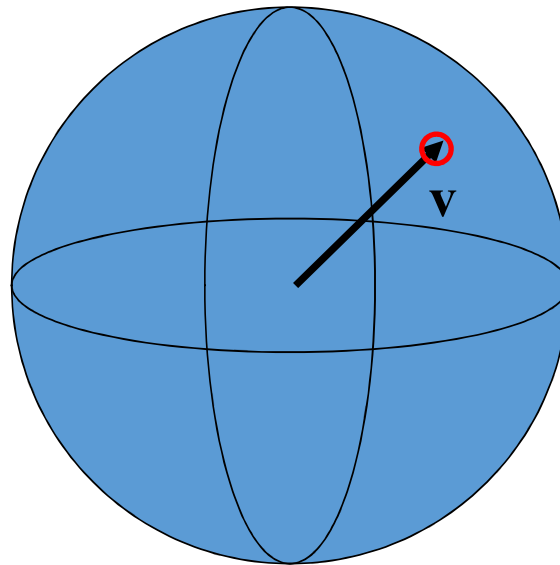


Retallat



Propietats

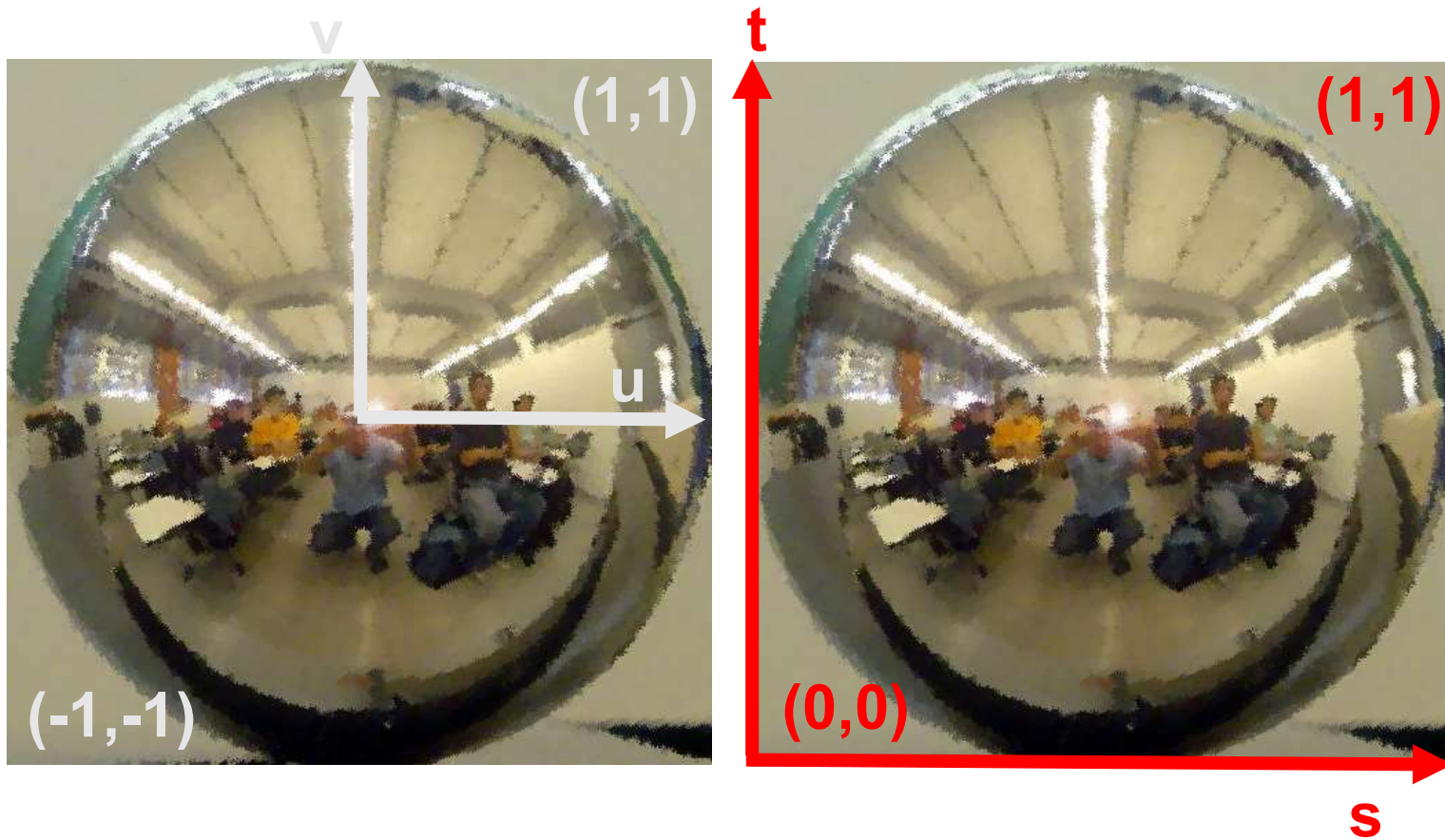
- De la textura només se n'aprofita el cercle inscrit
- Conté informació de aproximadament tot l'entorn (totes direccions)
- Distorsió considerable a prop de la vora del cercle



Exemple



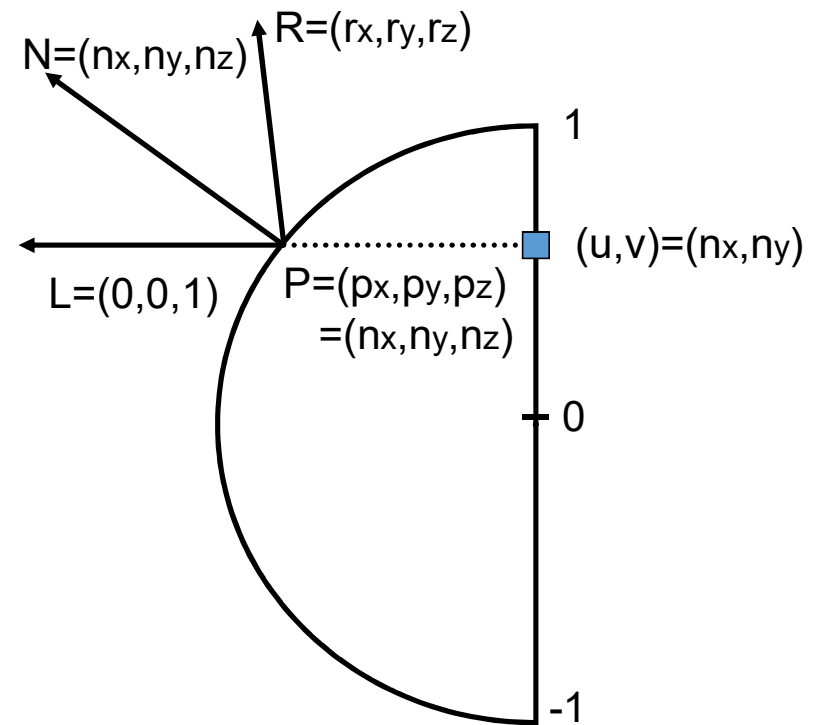
Coordenades $(u,v) \leftrightarrow (s,t)$



Relació entre vector R i (u,v)

Relació entre vector R i (u,v)

$$R=(2n_z n_x, 2n_z n_y, 2n_z^2-1)$$



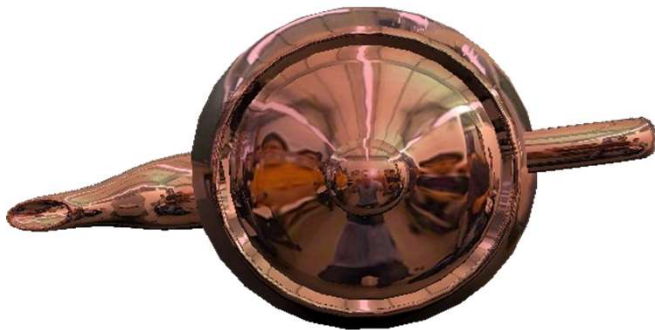
Càlcul del color donat R

```
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 texture(sampler, st);
}
```

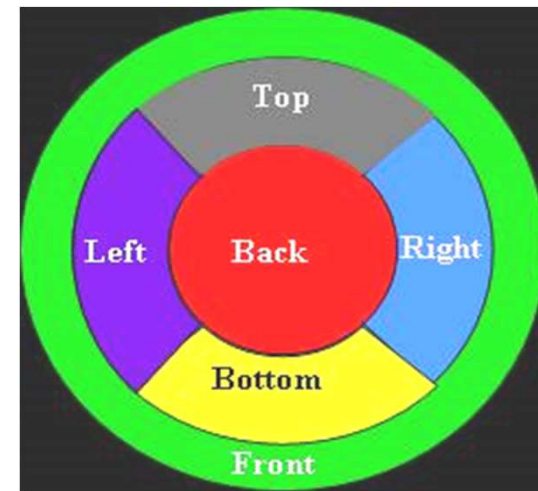
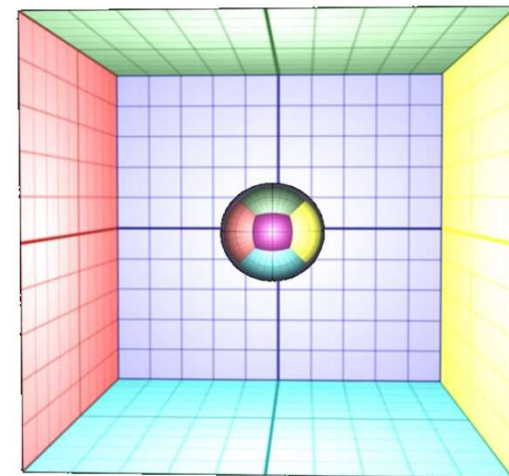
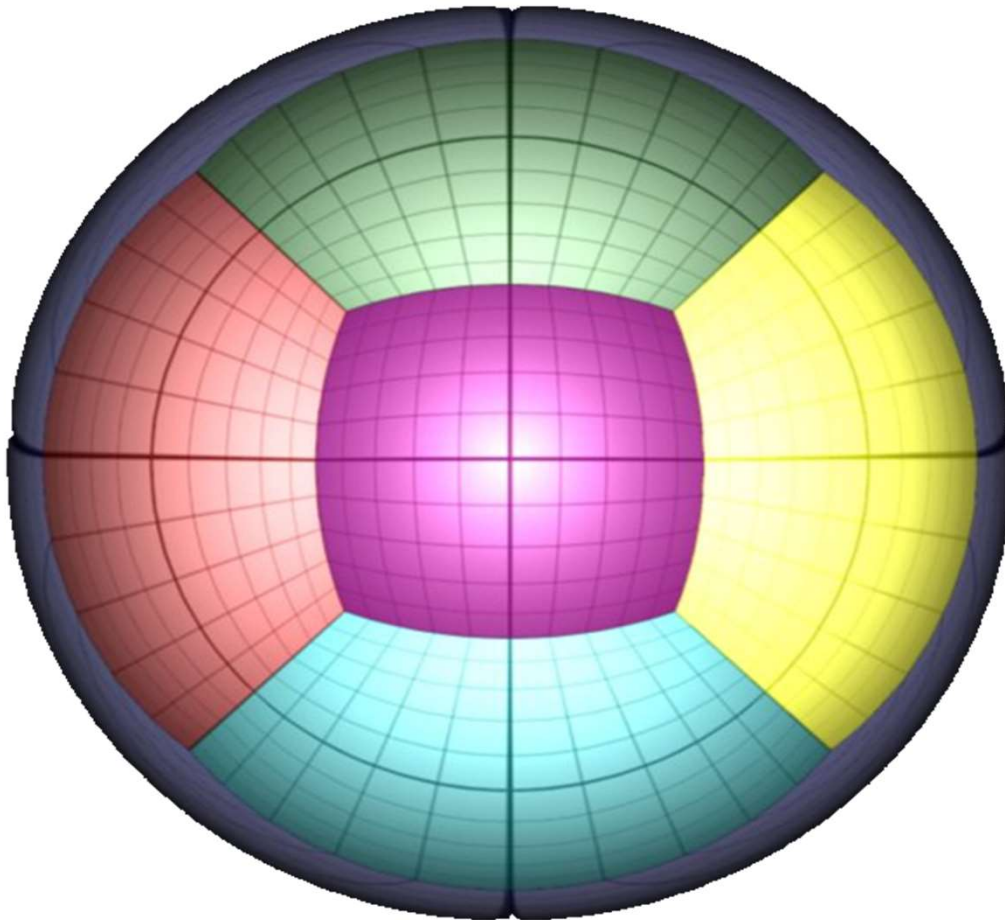
Eye/world coordinates



Eye/world coordinates



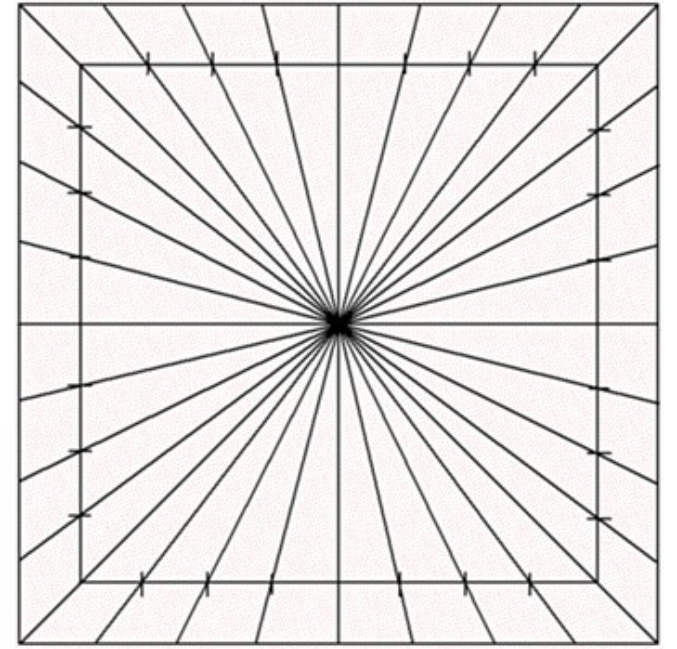
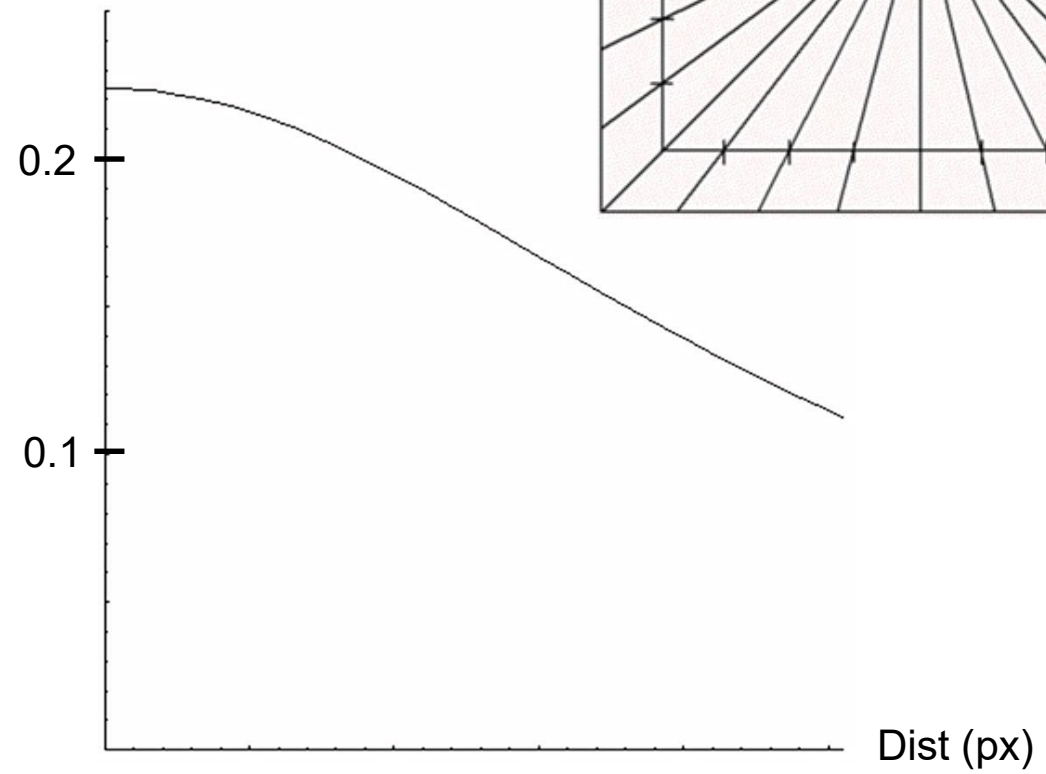
Sphere mapping: distorsió



CUBE MAPPING



Angle (deg)



Cube mapping: exemple



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: GLSL

```
uniform sampler2D sampler;  
...  
fragColor = texture(sampler, vtexCoord);
```

```
uniform samplerCube samplerC;  
...  
vec3 R;  
...  
fragColor = textureCube(samplerC, R);
```