

# Gràfics i Visualització de Dades

## T3: Mètodes projectius: Textures

Anna Puig

**NOTA:** Aquestes transparències es corresponen a les explicacions del tema 7.8 i 7.9 del llibre de referència bàsica

[Angel2011] Edward Angel, Dave Shreiner, **Interactive Computer Graphics: A Top-Down Approach with Shader-Based OpenGL, 6/E**, ISBN-10: 0132545233. ISBN-13: 9780132545235, Addison-Wesley, 2011

# Índex

3.1. Z-Buffer: Pipeline de visualització

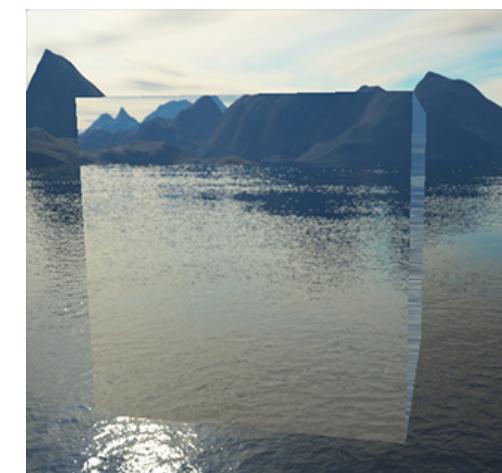
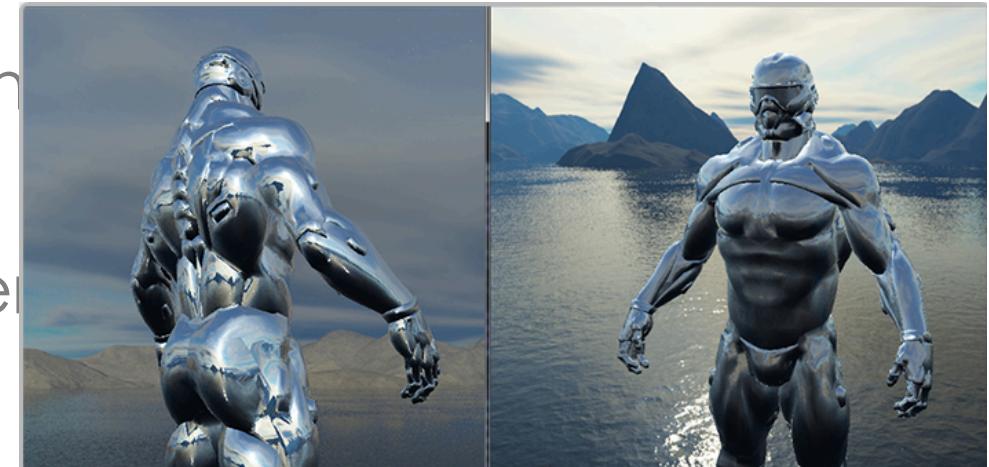
3.2. Pipeline integrat a OpenGL

3.3. Il·luminació usant shaders

3.4. Textures

3.5. **Reflexions i transparències**

3.6. Usos avançats



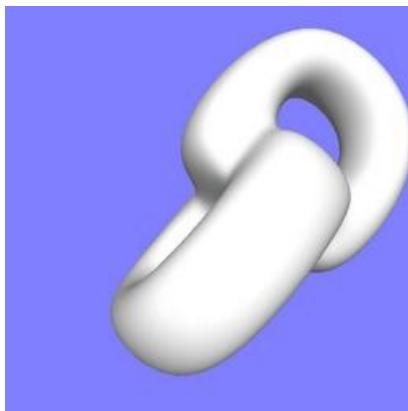
# 3.5. Reflexions: environmental mapping

**Problema:** Es vol reflectir l'entorn en un objecte:

D'on s'obté l'entorn?

1. A partir de la mateixa escena

2. L'entorn és una textura externa que simula el fons (background de l'escena)



# 3.5. Reflexions: environmental mapping

**Problema:** Es vol reflectir l'entorn en un objecte:

D'on s'obté l'entorn?

L'entorn es suposa allunyat infinitament



1. A partir de la mateixa escena

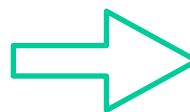
**IDEA:** Es realitzen dues visualitzacions (**shading en dos passos**):

1<sup>er</sup> pas

Visualització de l'escena sense les parts que són miralls

2<sup>on</sup> pas

Visualització només de les parts que són miralls

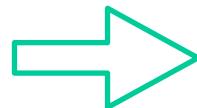


# 3.5. Reflexions: environmental mapping

**Exemple:**

Es vol aconseguir la imatge següent:

Visualització de l'escena sense les parts que són miralls



Visualització només de les parts que són miralls



1<sup>er</sup> pas



2<sup>on</sup> pas



Es visualitza l'escena sense l'objecte mirall



Sobre la imatge anterior, es visualitza la imatge vista des del mirall

# 3.5. Reflexions: environmental mapping

**Problema:** Es vol reflectir l'entorn en un objecte:



**2. L'entorn és una textura externa que simula el fons (background de l'escena): Mapping esfèric**



Textura: imatge HDR



Texture mapping indirecta (en 2 fases)

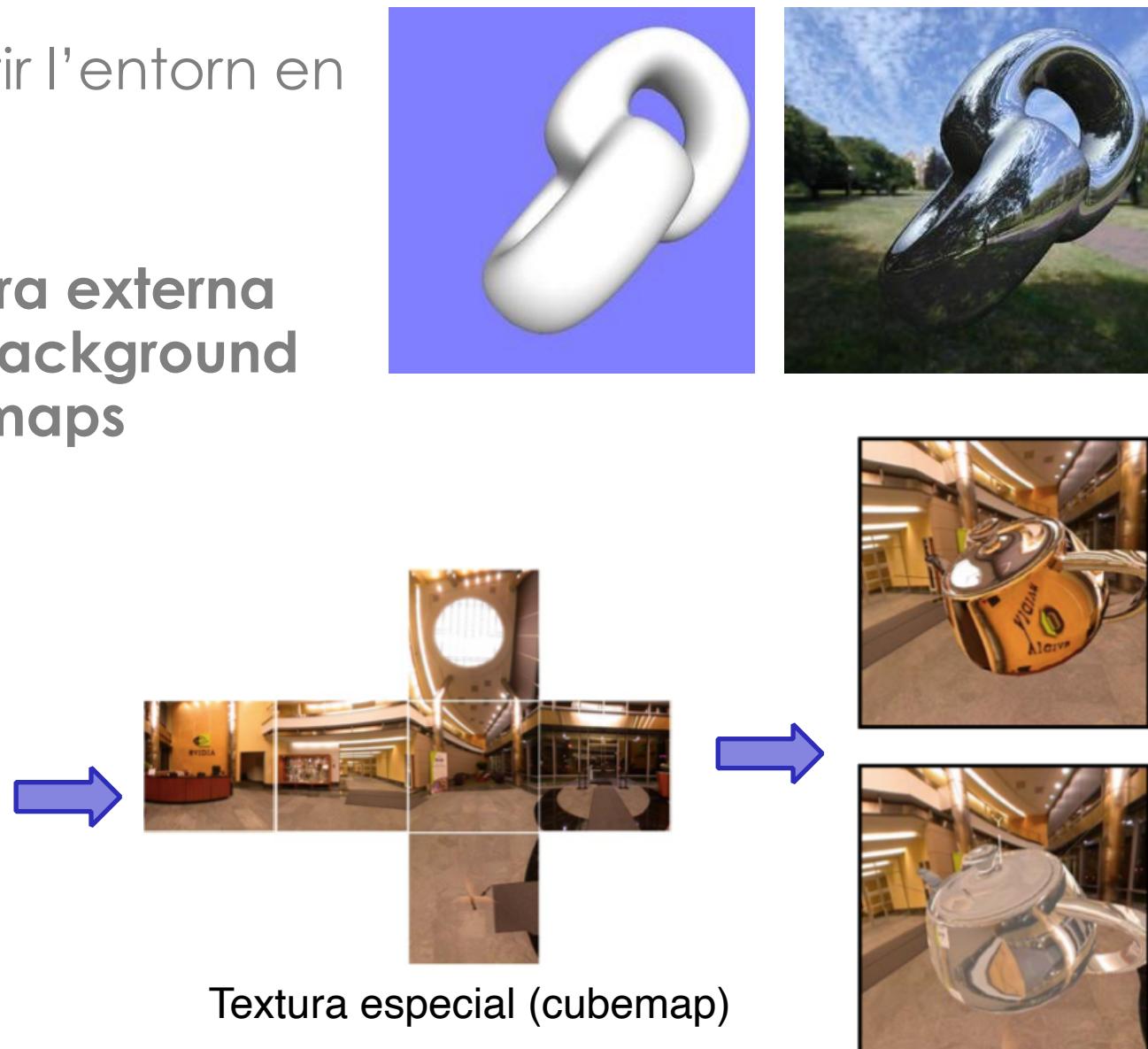
# 3.5. Reflexions: environmental mapping

**Problema:** Es vol reflectir l'entorn en un objecte:

2. L'entorn és una textura externa que simula el fons (background de l'escena): Cube maps



Cub que engloba l'escena

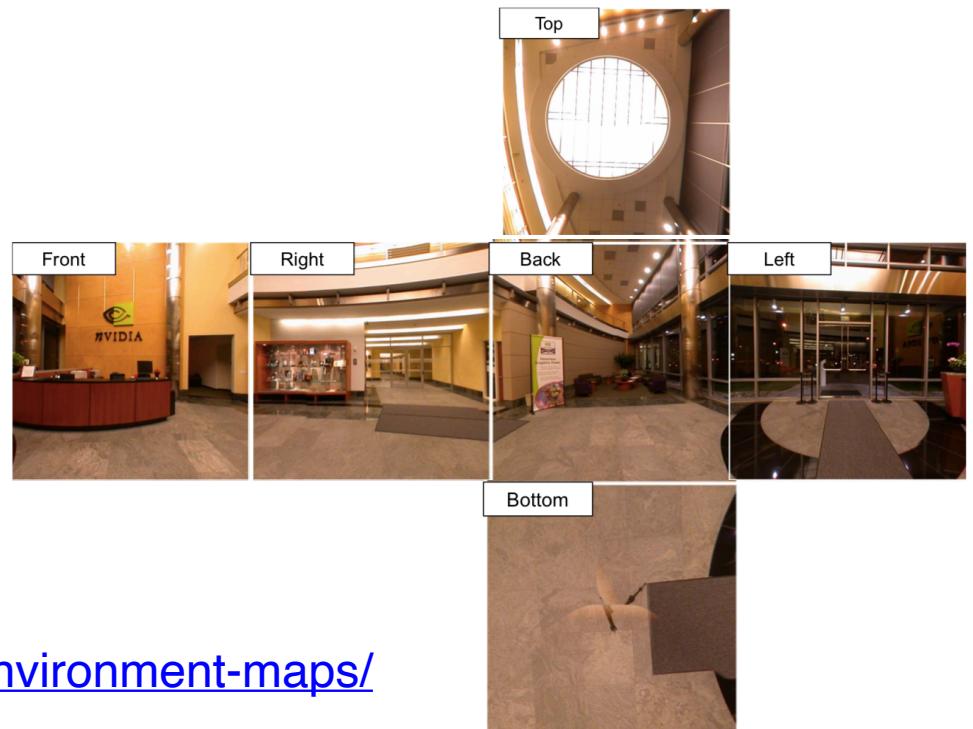
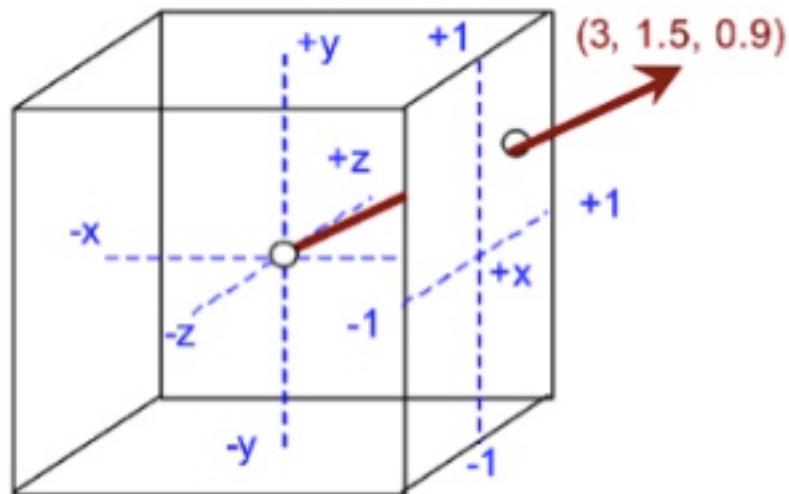


Textura especial (cubemap)

# 3.5. Reflexions: environmental mapping

- **Cub** (especial en OpenGL)

- Textures especials de GL (**CubeMaps**) i GLSL (**SamplerCube**)
- Un CubeMap és el conjunt de 6 textures (una per a cada cara del cub)
- S'accedeixen amb tres coordenades  $(s, r, t)$  en lloc de  $(s, t)$ . Ara  $(s, r, t)$  és una direcció.



<https://aerotwist.com/tutorials/create-your-own-environment-maps/>

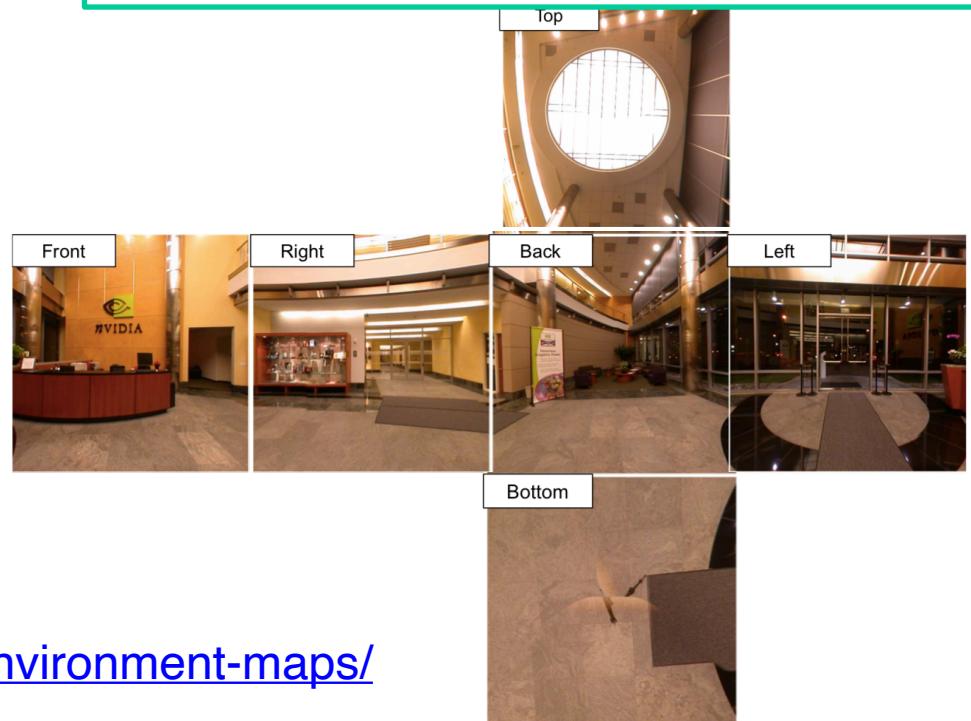
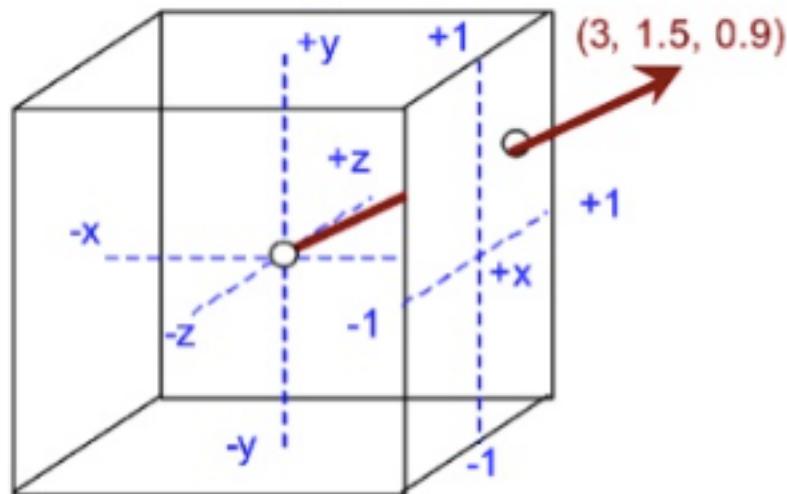
# 3.5. Reflexions: environmental mapping

## Cub (especial en OpenGL)

- Textures especials de GL (**CubeMap**)
- Un CubeMap és el conjunt de 6 textures d'un cub
- S'accedeixen amb tres coordenades que no és una direcció.

Com s'accedeix a la textura amb (s, r, t)?

- S'agafa la coordenada de màxim valor per determinar la cara
- Les altres dues coordenades dividides pel valor màxim i traslladades a l'interval (0, 1), donaran les coordenades (s,t) de textura corresponents a la cara seleccionada

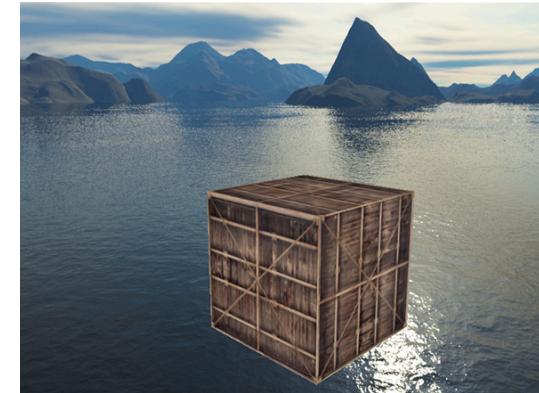
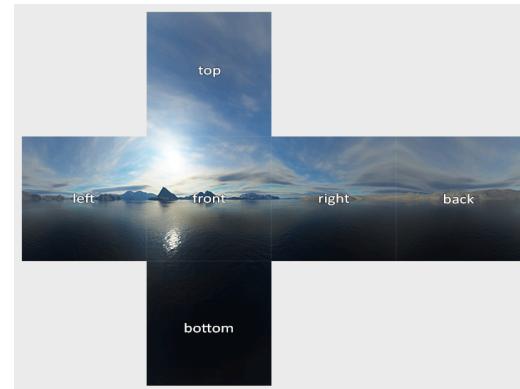


<https://aerotwist.com/tutorials/create-your-own-environment-maps/>

# 3.5. Reflexions: environmental mapping

- **Com es programa en OpenGL i glsl? Cas de visualitzar només el Cube Map en background**

1. Definició de la classe cub (amb l'**orientació de les cares cap a dins** des del make, modificant el codi del projecte CubGPUTextures)
2. Creació de la textura en la CPU com un CubeMap
3. Pas de la textura a la GPU
4. Pas o càlcul de les coordenades de textures a la GPU
5. Ús de la textura en el fragment shader



## 3.5. Reflexions: environmental mapping

- **Com es programa en OpenGL i glsl? Cas de visualitzar només el Cube Map**

2. Definició de la textura en la CPU.

1. Carregar les 6 imatges en una taula d'imatges (**faces[]**):

```
faces.push_back("//resources/textures/skybox/right.jpg");
faces.push_back("//resources/textures/skybox/left.jpg");
faces.push_back("//resources/textures/skybox/bottom.jpg");
faces.push_back("//resources/textures/skybox/top.jpg");
faces.push_back("//resources/textures/skybox/back.jpg");
faces.push_back("//resources/textures/skybox/front.jpg");
```

```
QImage image[6];

for(GLuint i = 0; i < faces.size(); i++)
{
    image[i] = QImage(faces[i]).convertToFormat(QImage::Format_RGBA8888);
}
```

## 3.5. Reflexions: environmental mapping

- **Com es programa en OpenGL i glsl? Cas de visualitzar només el Cube Map**

2. Definició de la textura en la CPU

2. Per a fer la textura, cal activar-la, esperar a ser creada i fer el bind abans de posar-li les dades. En l'exemple, les imatges s'han carregat prèviament en una taula d'imatges:

```
glActiveTexture(GL_TEXTURE0);

texture = new QOpenGLTexture(QOpenGLTexture::TargetCubeMap);
if (!texture->isCreated())
    texture->create();

 glBindTexture(GL_TEXTURE_CUBE_MAP, texture->textureId());

texture->setFormat(QOpenGLTexture::RGBAFormat);
texture->setSize(image[0].width(), image[0].height(), image[0].depth());
texture->generateMipMaps();
texture->allocateStorage();

texture->setData(0, 0, QOpenGLTexture::CubeMapPositiveX, QOpenGLTexture::RGBA,
                 QOpenGLTexture::UInt8, (const void*)image[1].constBits(), 0);
texture->setData(0, 0, QOpenGLTexture::CubeMapPositiveY, QOpenGLTexture::RGBA,
                 QOpenGLTexture::UInt8, (const void*)image[3].constBits(), 0);
```

## 3.5. Reflexions: environmental mapping

- **Com es programa en OpenGL i glsl? Cas de visualitzar només el Cube Map**

2. Definició de la textura en la CPU:

3. Just després de crear la textura, cal posar els modes de WRAP i de mipmap:

```
texture->setWrapMode(QOpenGLTexture::ClampToEdge);
texture->setMinificationFilter(QOpenGLTexture::LinearMipMapLinear);
texture->setMagnificationFilter(QOpenGLTexture::LinearMipMapLinear);

glGenerateMipmap(GL_TEXTURE_CUBE_MAP);
```

# 3.5. Reflexions: environmental mapping

- **Com es programa en OpenGL i glsl? Cas de visualitzar només el Cube Map**

3. Pas de la textura a la GPU:

a. toGPU:

```
texture->bind(texture->textureId());
program->setUniformValue("texEnvironment", texture->textureId());

 glEnable( GL_TEXTURE_CUBE_MAP );
```

b. en el moment de fer el draw, s'activa la funció de més petit o igual en el Z-Buffer i en finalitzar es desactiva el mode de texture en cube map:

```
glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
glDepthFunc(GL_EQUAL);
glDrawArrays( GL_TRIANGLES, 0, Index );
glDepthFunc(GL_LESS);

glDisableVertexAttribArray(0);
glDisableVertexAttribArray(1);

glDisable(GL_TEXTURE_CUBE_MAP);
```

# 3.5. Reflexions: environmental mapping

- **Com es programa en OpenGL i glsl? Cas de visualitzar només el Cube Map**

4. Pas o càlcul de les coordenades de textures a la GPU (vertex shader).

```
out vec4 position;
out vec3 v_texcoord; //textura

void main()
{
    ....
    position = vPosition;

    // Pas de les coordenades de textura al fragment shader
    // El valor del color i les coordenades de textura s'interpolaran automaticament
    // en els fragments interiors a les cares dels polígons
    v_texcoord = normalize(position.xyz); //textura
}
```

# 3.5. Reflexions: environmental mapping

- **Com es programa en OpenGL i glsl? Cas de visualitzar només el Cube Map**

6. Ús de la textura en el fragment shader.

```
in vec3 v_texcoord;
out vec4 colorOut;

uniform samplerCube texEnvironment;

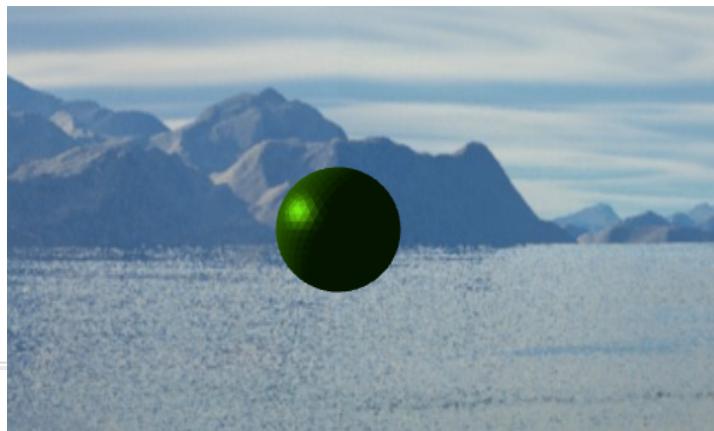
void main()
{
    colorOut = vec4(texture(texEnvironment, v_texcoord.xyz).rgb,
1.0f);
}
```

# 3.5. Reflexions: environmental mapping

- **Com es programa en OpenGL i glsl? Cas de visualitzar només el Cube Map**

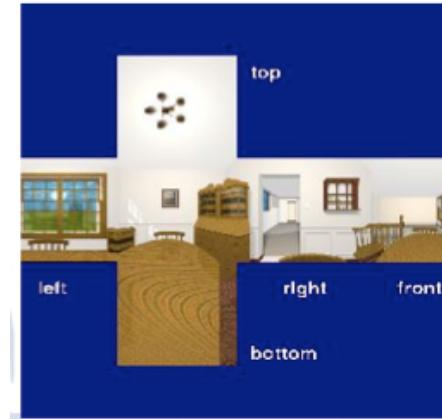
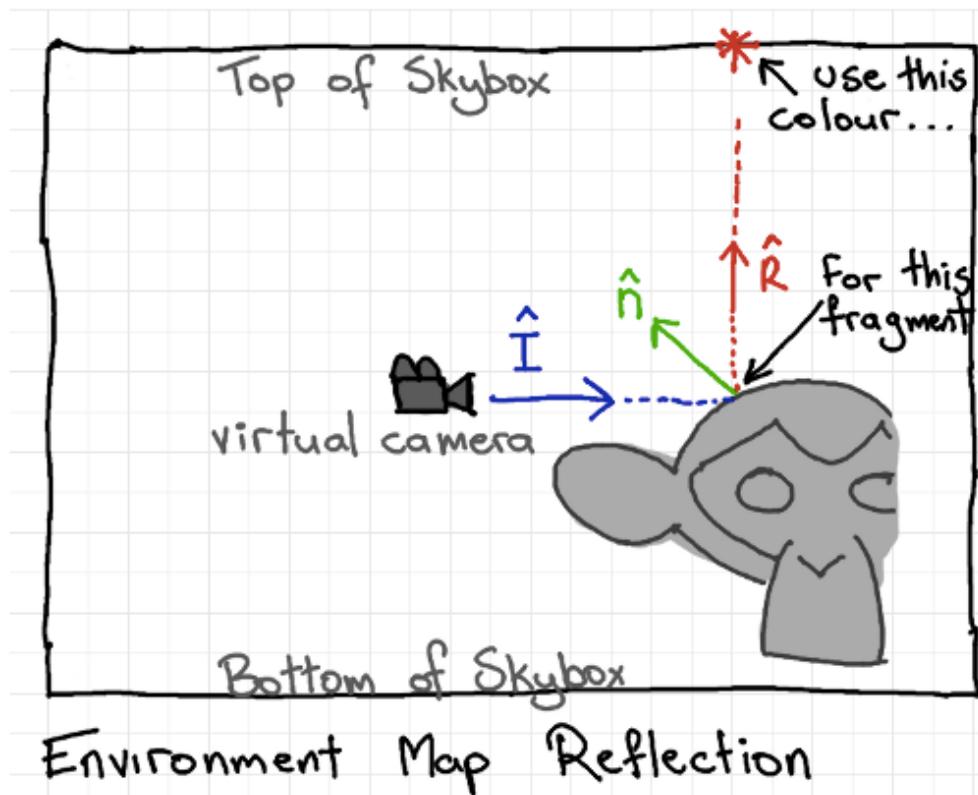
6. Com queda el paintGL()?

- Es neteja el buffer (glClear)
- S'activa el parell de shaders que pinten el cub que engloba l'escena aplicant el cubeMap
- Es pinta el Cub
- S'activa el parell de shaders amb els que es vol visualitzar l'escena
- Es pinta l'escena (però no el cub exterior)



# 3.5. Reflexions: environmental mapping

- Simulant reflexions usant CubeMaps



# 3.5. Reflexions: environmental mapping

- **Simular reflexions usant CubeMaps**

Cal activar un altre parell de shaders per a visualitzar objectes metàl·lics. Es passarà la textura de cubeMapper a calcular el color final del píxel de l'objecte:

1. Pas de l'objecte a la GPU
2. Pas de la textura a la GPU
3. Ús de la textura en el fragment shader.

```
// Calcul del raig reflectit des de vPosition amb la direcció de visió
vec3 R = normalize(reflect(-v.xyz, normalize(normal.xyz)));
colorOut = texture(texEnvironment, R);
```

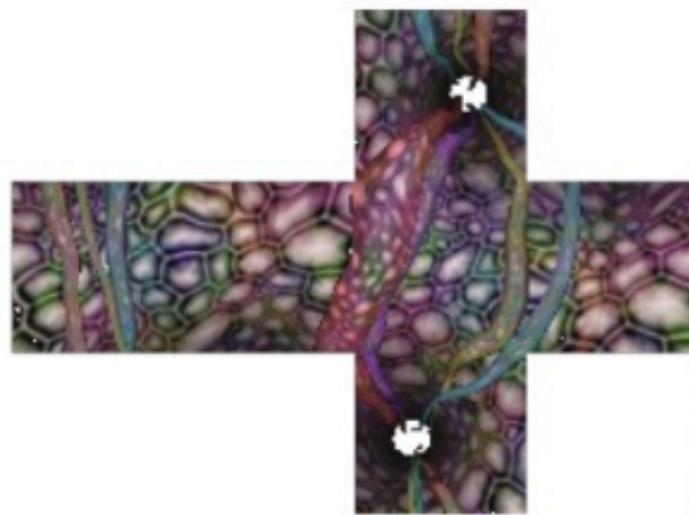
Amb quines coordenades es calcula el vector Reflectit?

- En coordenades de món?
- De càmera?

# 3.5. Reflexions: environmental mapping

- **Cube Maps dinàmics**

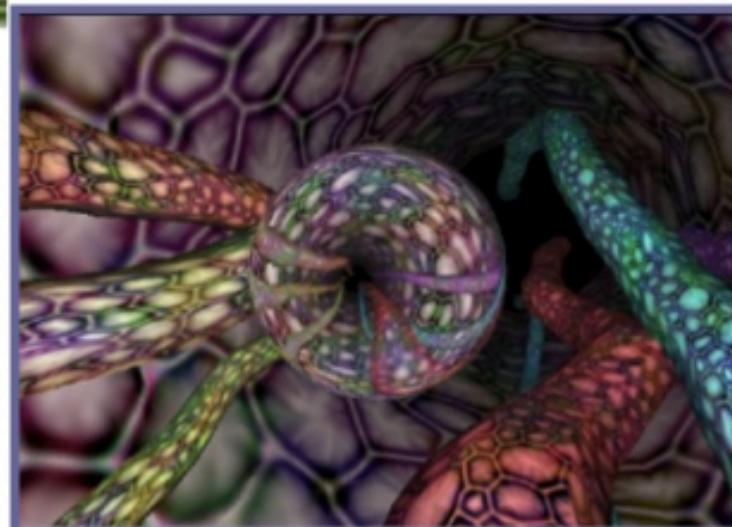
- Es calculen 6 visualitzacions des del centre del cub sense l'objecte
- Es visualitza l'escena fent environmental mapping



**Dynamically  
created  
cube map image**

**Image credit:**  
“Guts” GeForce 2 GTS demo,  
Thant Thessman

**Rendered scene**



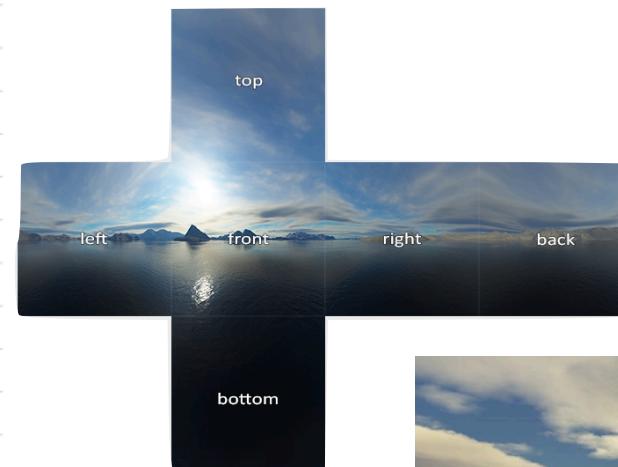
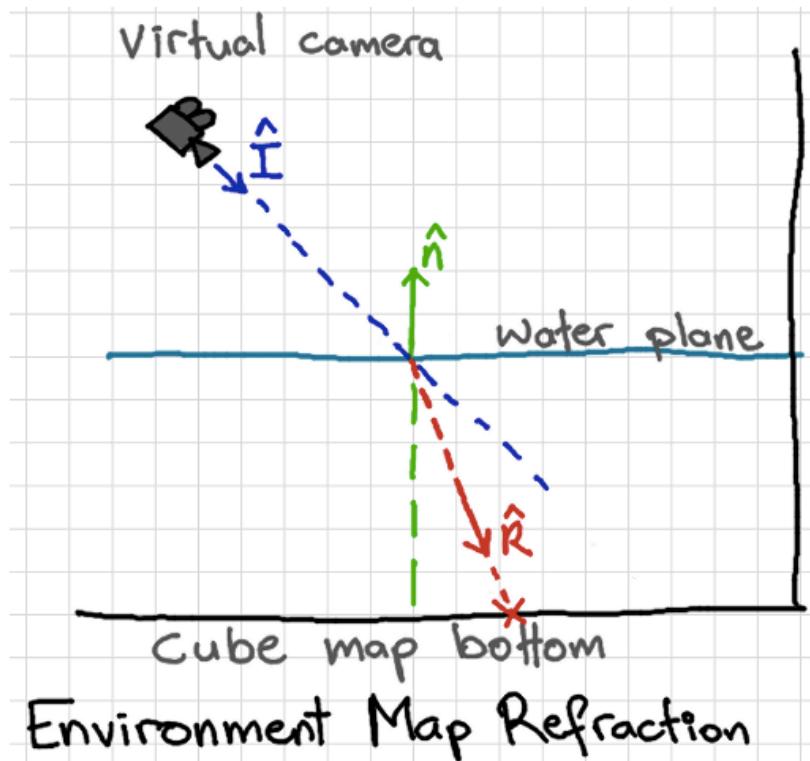
## 3.5. Transparències: environmental mapping

- **Simular transparències amb CubMaps**

Es fa el mateix procés que en les reflexions però amb el raig refractat

- S'usa el mètode definit en GLSL

**refract(raig\_incident, normal, índice\_refracció)**



# Índex

3.1. Z-Buffer: Pipeline de visualització

3.2. Pipeline integrat a OpenGL

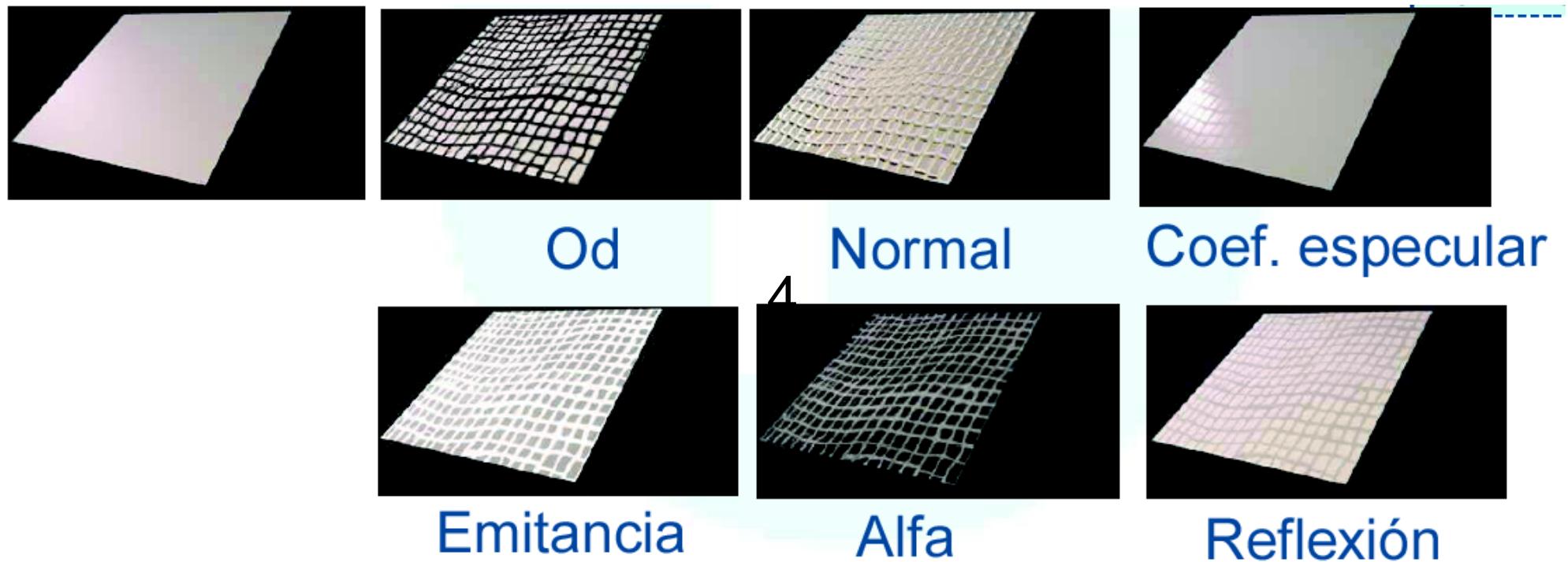
3.3. Il·luminació usant shaders

3.4. Textures

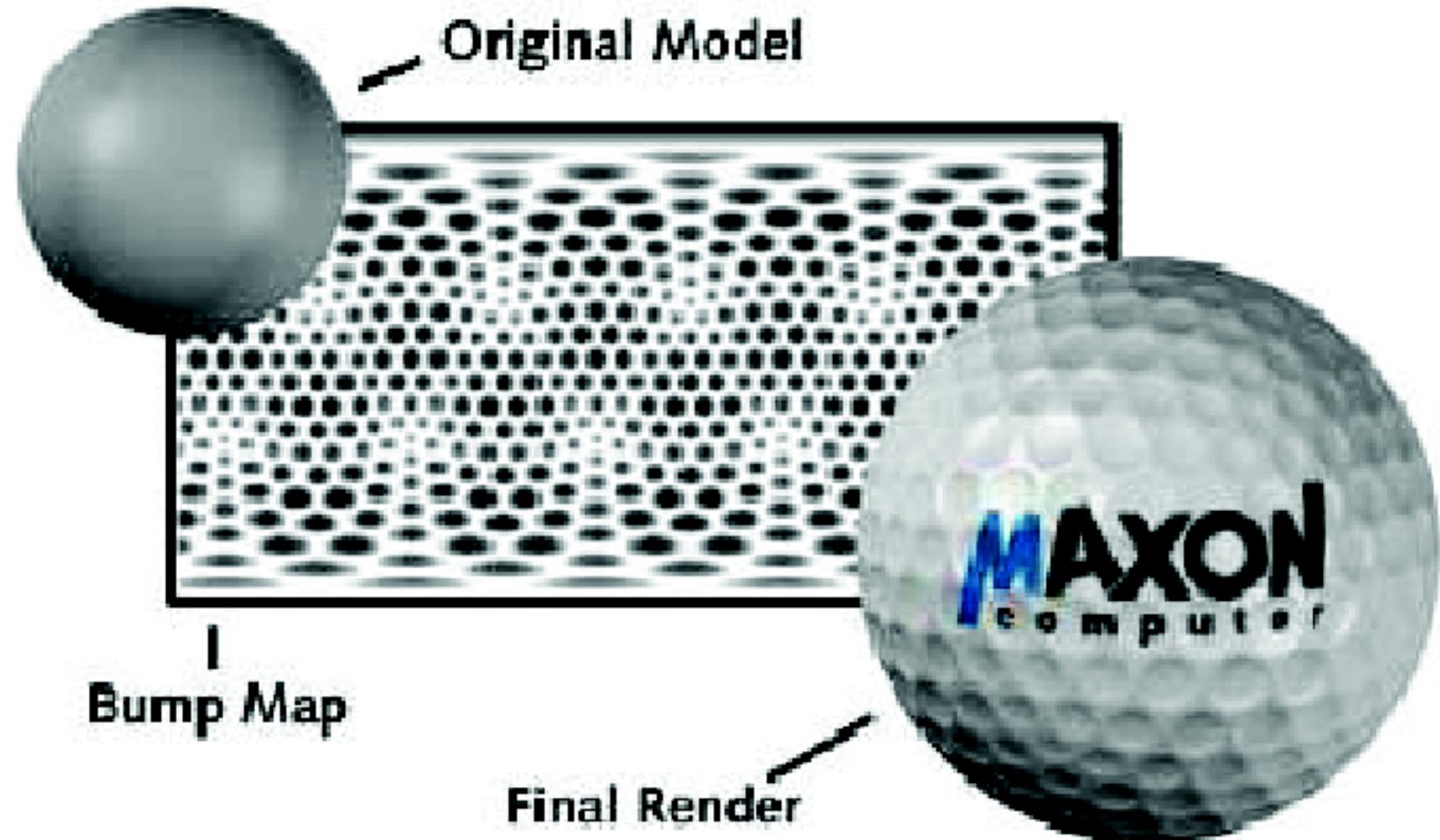
3.5. Reflexions i transparències

3.6. Usos avançats

# 3.6. Textures: usos avançats



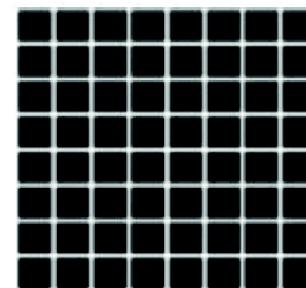
## 3.6. Textures: Bump mapping (normal)



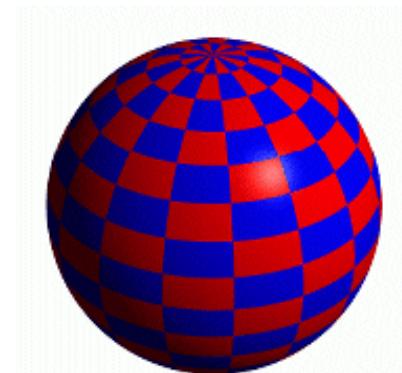
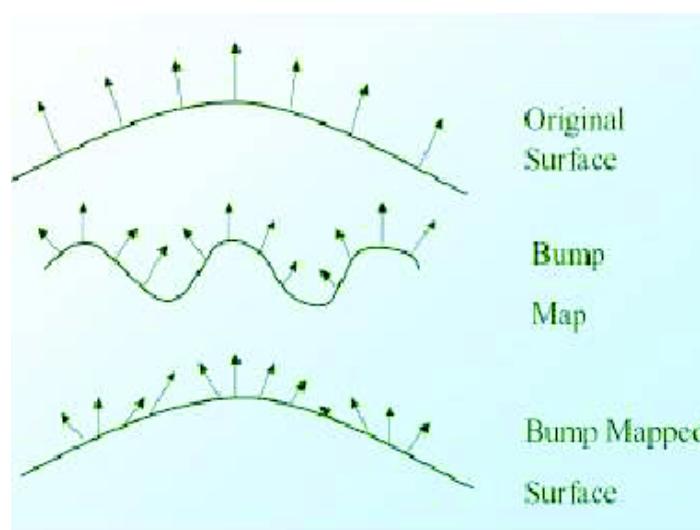
# 3.6. Textures: Bump mapping (normal)

Distorsió de la direcció de cadascuna de les coordenades de les normals en el moment de calcular a la il·luminació.

No canvia la superfície



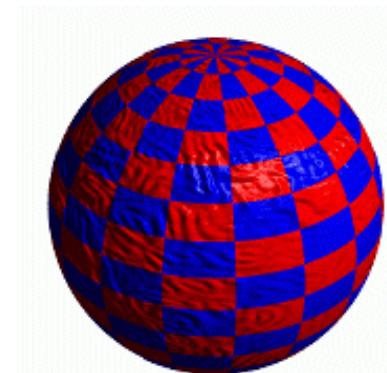
Canvia el càcul de la il·luminació



Sphere w/Texture



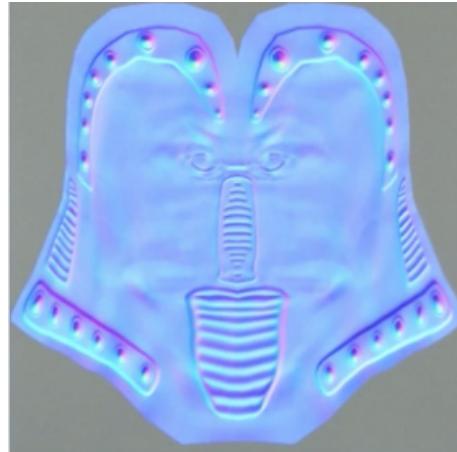
Bump Map



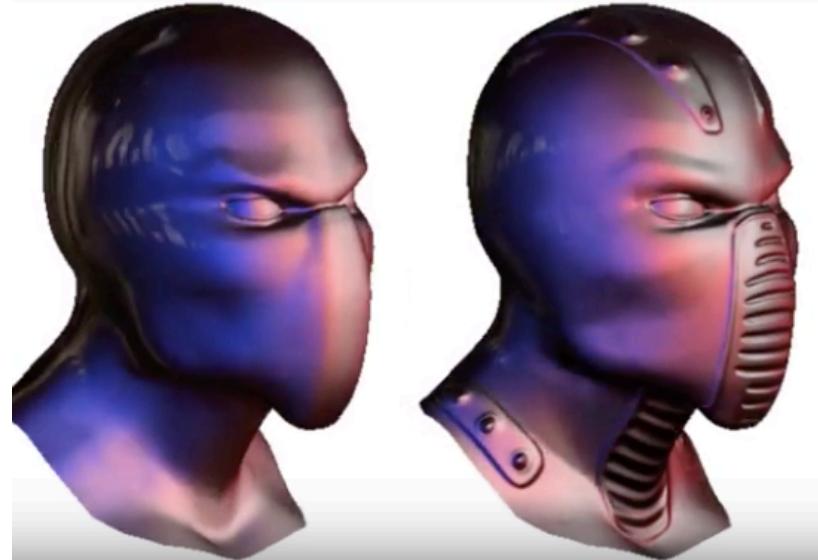
Bumpy Sphere

## 3.6. Textures: Bump mapping (normal)

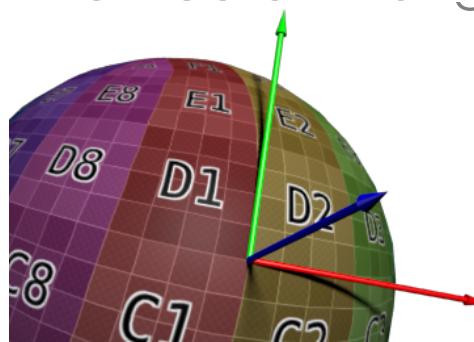
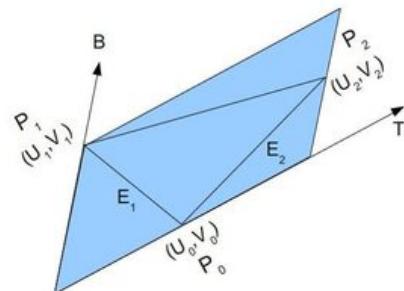
Distorsió de la direcció de cadascuna de les coordenades de les normals.



Normal (R, G, B)



Cal transformar les normals a l'espai tangent a la superfície, usant la normal, el vector Tangent i el vector Bitangent



$$\begin{bmatrix} T_x & B_x & N_x \\ T_y & B_y & N_y \\ T_z & B_z & N_z \end{bmatrix}$$

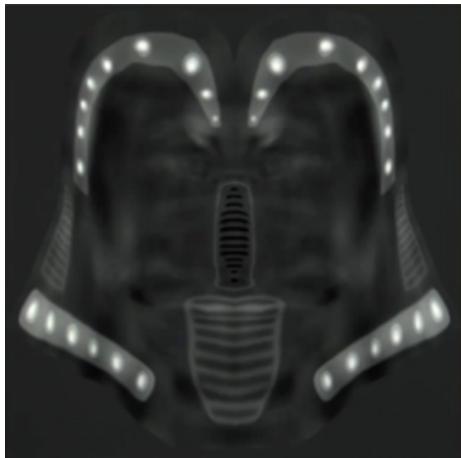
(pag. 178, de la referència [2])

## 3.6. Textures: displacement mapping

Modificació dels vèrtexs de la superfície de l'objecte en una certa direcció i un cert desplaçament

Utilització de dos mapes de textures (la direcció i el desplaçament)

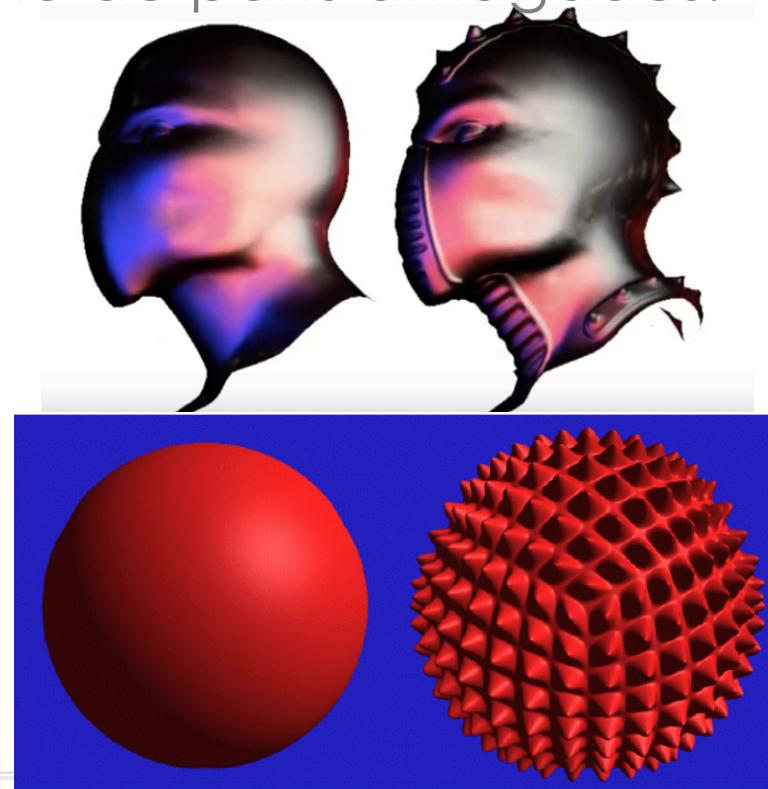
Es realitza abans de l'eliminació de parts amagades.



Displacement



Normal



Semestre Primavera  
2022-23

# Referències

- [1] Tema 5: Edward Angel, Dave Shreiner, Interactive Computer Graphics: A Top-Down Approach with Shader-Based OpenGL, 6/ E, ISBN-10: 0132545233. ISBN-13: 9780132545235, Addison-Wesley, 2011
- [2] Mathematics for 3D Game Programming and Computer Graphics, 2º Edition,  
[http://canvas.projekti.info/ebooks/  
Mathematics%20for%203D%20Game%20Programming%20and%20Computer  
r%20Graphics,%20Third%20Edition.pdf](http://canvas.projekti.info/ebooks/Mathematics%20for%203D%20Game%20Programming%20and%20Computer%20Graphics,%20Third%20Edition.pdf)
- [3] Aplicació de normal mapping en GLSL : Càlcul de l'espai tangent:  
<https://learnopengl.com/Advanced-Lighting/Normal-Mapping>
- [4] Tutorial sobre CubeMaps en GL i glsl: <https://www.keithlantz.net/2011/10/lighting-and-environment-mapping-with-glsl/>