

# **Visualización: Materiales**

Francisco Velasco Anguita

Dpto. Lenguajes y Sistemas Informáticos  
Universidad de Granada

Sistemas Gráficos

Grado en Ingeniería Informática  
Curso 2019-2020

# Contenidos

- 1 Introducción**
- 2 Materiales basados en un color**
- 3 Materiales con texturas**
- 4 Ejemplo: El planeta Tierra**

# Objetivos

- Entender cómo reacciona la luz con los objetos según su material
- Conocer los diferentes componentes de un material
- Saber crear materiales
  - ▶ Basados en un color
  - ▶ Basados en una textura
  - ▶ Basados en una combinación de color y varias texturas
- Saber crear un entorno mediante texturas

# Introducción

## Captando la realidad



# Modelo de iluminación

## Rendering

- Proceso de cálculo desarrollado por un ordenador destinado a generar una imagen o secuencia de imágenes.
- Influyen los elementos siguientes:

### Geometría: (*Paso de 3-D a 2-D*)

Proyección, ocultación, distorsiones de la perspectiva.

### Fotometría: (*Luz reflejada por los objetos de la escena*)

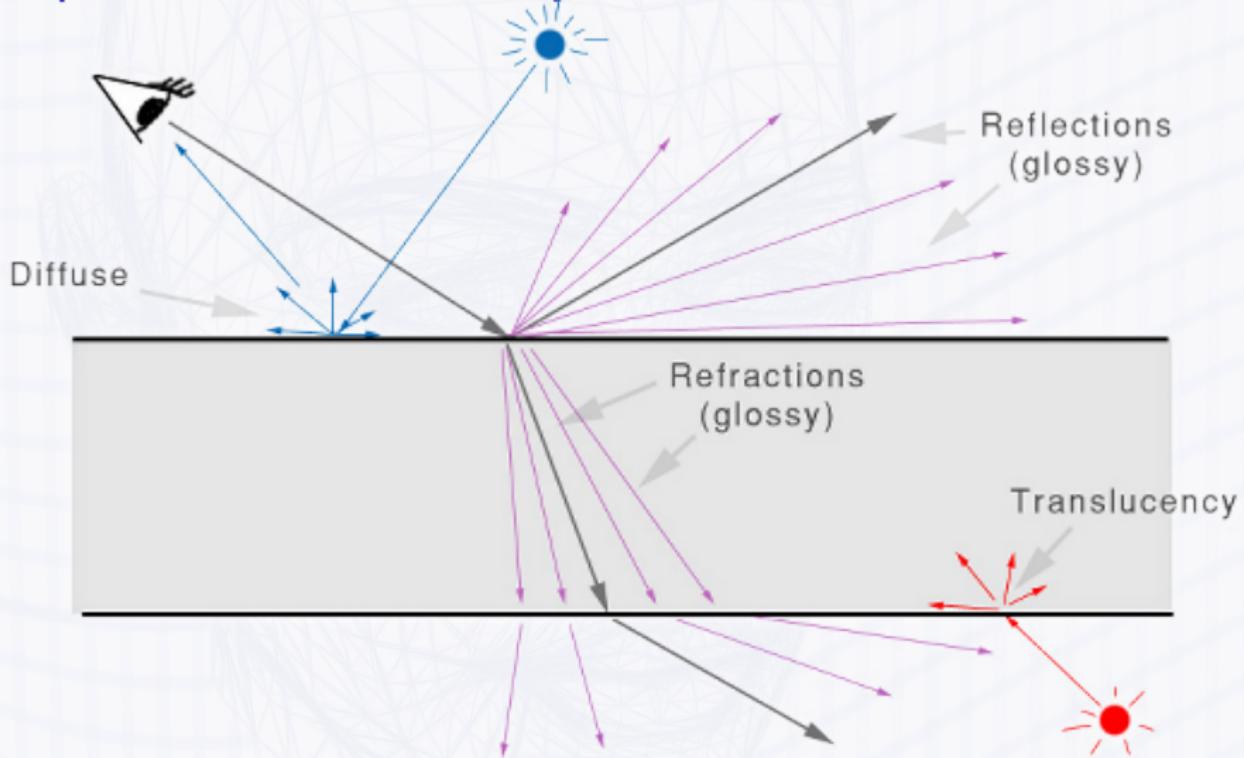
Tipo, intensidad y dirección de la iluminación, reflectancia de las superficies, etc.

## Modelo de Iluminación

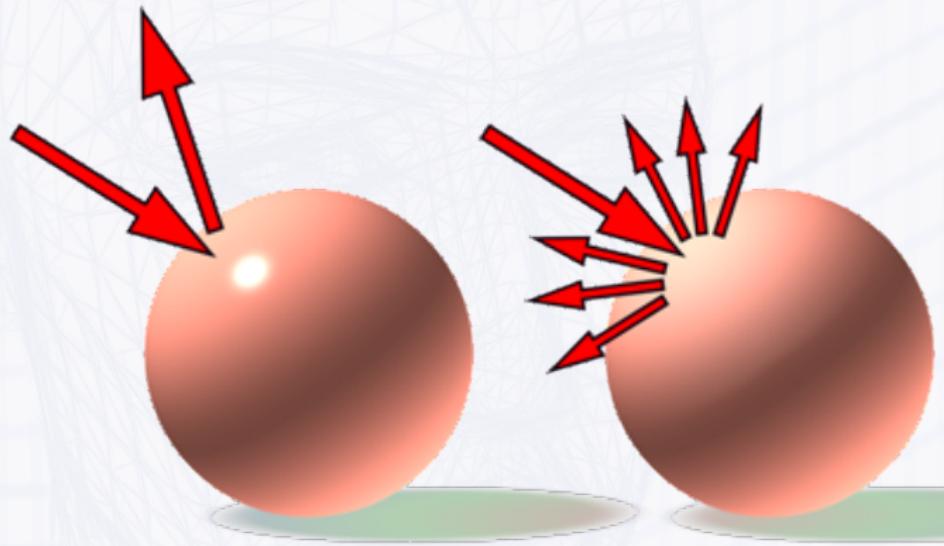
Conjunto de propiedades físicas de los objetos y de la luz que intervienen en el proceso de rendering.

# Materiales

## Comportamiento de la luz en los objetos



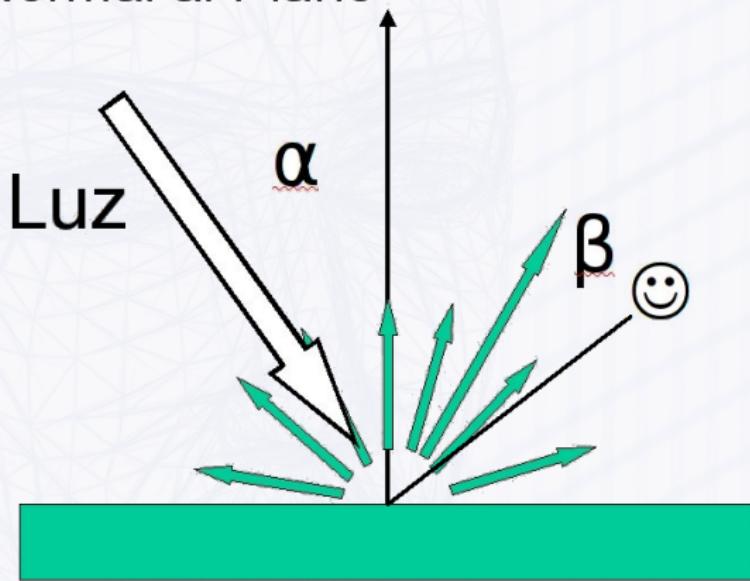
# Componente Especular vs. Difusa



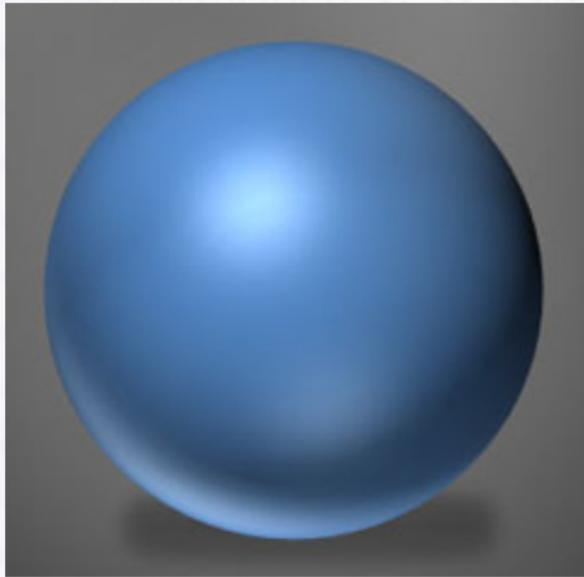
# Componente Especular vs. Difusa

Cálculo del color en un punto

Vector Normal al Plano



# Componentes *ambiental*, difusa y especular



- Componente Difusa
- Componente Especular
- Componente Ambiental

# Materiales en

# Three.js

- Three.js es una capa sobre WebGL
- El uso de programas shaders es obligatorio
- Cada shader implementa un modelo de iluminación
  - ▶ Un shader recibe:
    - ★ Los vértices de cada polígono
    - ★ Otra información necesaria según el modelo de iluminación a aplicar:
      - Vector normal
      - Parámetros del material
      - Características de las luces, etc.
  - ▶ Calcula el color que le corresponde a un determinado “pixel”
- En Three.js, cada material predefinido contiene su propio shader
- Pueden programarse shaders propios

# Clase Material

## Three.js

- Clase base para el resto de materiales
- Atributos comunes
  - ▶ name
    - ★ Permite identificar el material con un nombre
  - ▶ transparent: Boolean.
    - ★ Si `false`, la figura será opaca.
    - ★ Si `true`, la figura será transparente según el atributo de opacidad
  - ▶ opacity
    - ★ Indica el nivel de transparencia
    - ★ Un valor entre 0.0 (invisible) y 1.0 (opaco)
  - ▶ visible: Boolean
    - ★ Permite hacer invisible una figura (`false`)

## Clase Material (cont.)

# Three.js



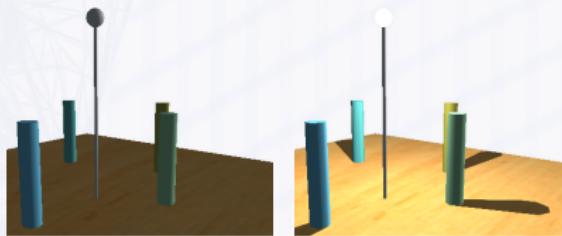
## Clase MeshNormalMaterial

- Colorea cada vértice según su normal
- Constructor: [MeshNormalMaterial\(\)](#)
- Usos:
  - ▶ Material para las figuras mientras se está desarrollando



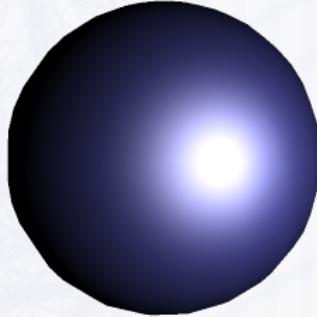
## Clase MeshLambertMaterial

- Modelo de Lambert: componentes ambiental, difusa y emisiva
- Atributos
  - ▶ `color`: Define el color difuso de la figura
  - ▶ `emissive`: Permite simular que la figura emite luz. Por defecto es negro.
  - ▶ `emissiveIntensity`: Define su intensidad, entre 0 y 1
- También dispone de los otros atributos vistos anteriormente: `shading`, `blending`, `opaque`, `transparent`, etc.
- **Usos:** Materiales mate



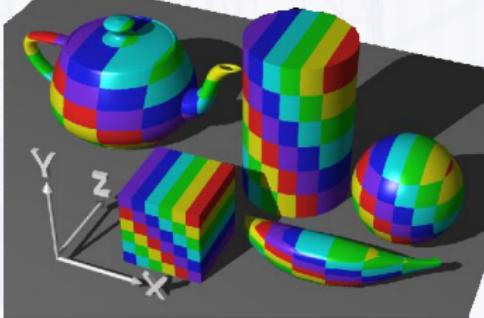
# Clase MeshPhongMaterial

- Modelo de Phong: Lambert + componente especular
- Atributos añadidos
  - ▶ specular: Define el color de los brillos
  - ▶ shininess: La intensidad del brillo. Por defecto, 30.
- **Usos:** Materiales con brillos especulares

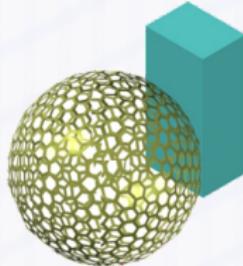
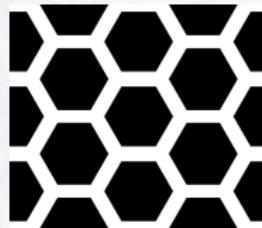
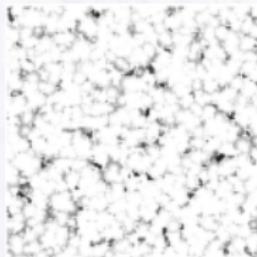


# Texturas

- Permiten usar una imagen para colorear un objeto

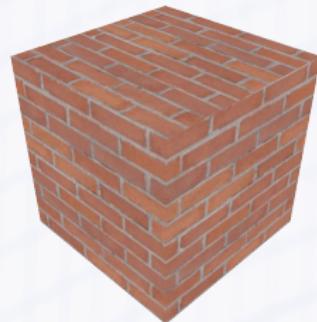


- O modificar otro tipo de características



# Texturas

## Three.js



- Se cargan con la siguiente clase

```
var loader = new THREE.TextureLoader();
var textura = loader.load ("imagen.jpg");
```

- Se pueden usar imágenes png, gif, o jpg
- Los mejores resultados se obtienen con imágenes cuadradas y resolución potencia de 2 (256x256, 512x512, etc.)
- Una vez cargada, se asigna al canal del material correspondiente
  - ▶ El canal difuso se denomina map

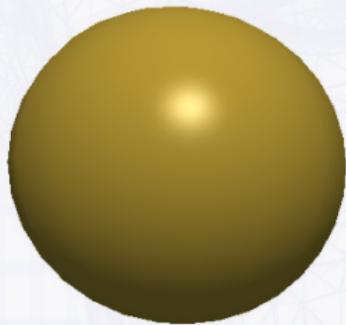
### Texturas: Uso de una textura en el canal difuso

```
var loader = new THREE.TextureLoader();
var textura = loader.load ("imagen.jpg");
var material = new THREE.MeshPhongMaterial ({map: textura});
```

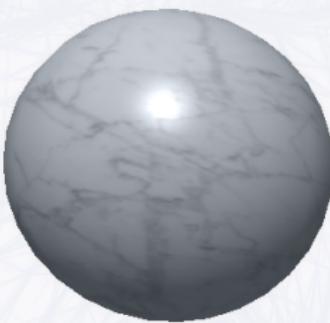
# Texturas

## Coloreando una textura

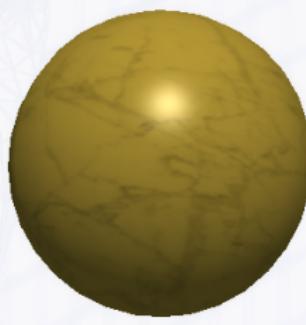
- Si a un material se le da color con el atributo `color` y también con una textura en el atributo `map` el resultado es la mezcla de ambos colores



color



map



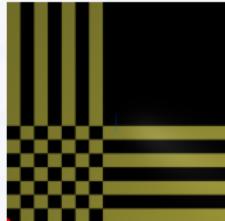
ambos

# Texturas

## Configuración de las texturas

- Repeticiones

- ▶ Se indican en los atributos `wrapS` y `wrapT`
- ▶ Puede tener 3 modos
  - ★ `THREE.RepeatWrapping`  
La textura se repite en esa dirección (S o T) tal cual
  - ★ `THREE.MirroredRepeatWrapping`  
La textura se repite en esa dirección (S o T) reflejada
  - ★ `THREE.ClampToEdgeWrapping`  
La textura no se repite, pero los píxeles de la frontera de la textura sí



- ▶ Configuración del número de repeticiones
  - ★ Se indica con el método `repeat.set (n,m)`
  - ★ A más repeticiones, menor será el tamaño de la imagen
- ▶ Un cambio de modo exige solicitar una actualización  
`material.map.needsUpdate = true;`
- ▶ Cambiar el número de repeticiones no lo requiere

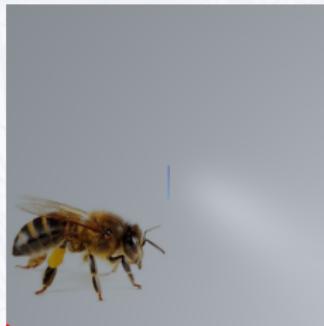
# Texturas

## Configuración de las texturas

### ● Desplazamiento

- ▶ Permite que la textura no se aplique desde el (0,0)
- ▶ Se indica con el método `offset.set (n, m)`

El desplazamiento se produce en la dirección opuesta al parámetro



offset de (0,0)



offset de (-0.5, -0.5)

En ambas hay un repeat de (2,2) en modo ClampToEdgeWrapping

# Texturas

## Configuración de las texturas

### • Giros

- ▶ Se puede situar el centro de giro
- ▶ El ángulo se da en radianes
- ▶ El centro se sitúa con `center.set (x, y)`
- ▶ El ángulo con `rotation = valor`

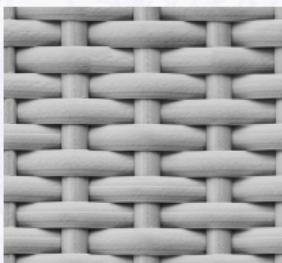


center en  $(0.5, 0.5)$  y giro de  $0.7$  radianes

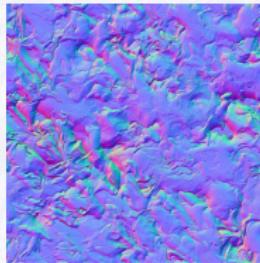
# Texturas en otros canales

## Relieve

- Se puede simular relieve mediante 2 canales
  - ▶ bumpMap, se usa una imagen en tonos de gris
    - ★ Se ajusta la magnitud con el escalar `bumpScale`



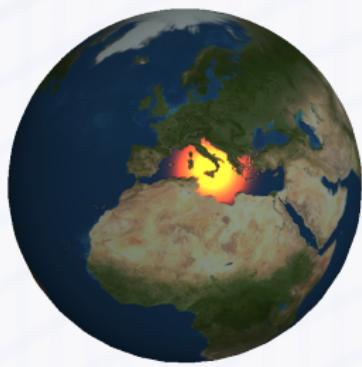
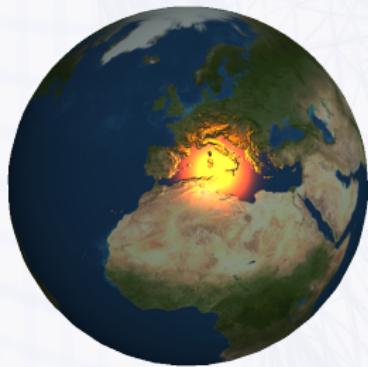
- ▶ normalMap, se usa una imagen que almacena vectores
  - ★ Se ajusta la magnitud con el vector2D `normalScale`



# Texturas en otros canales

## Especular

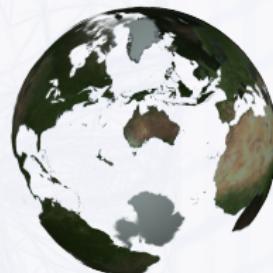
- Permite que se aplique la componente especular de manera distinta en cada parte de la geometría
- Se asigna un mapa en tonos de gris al canal `specularMap`



# Texturas en otros canales

## Transparencia

- Se asigna un mapa en tonos de gris al canal alphaMap



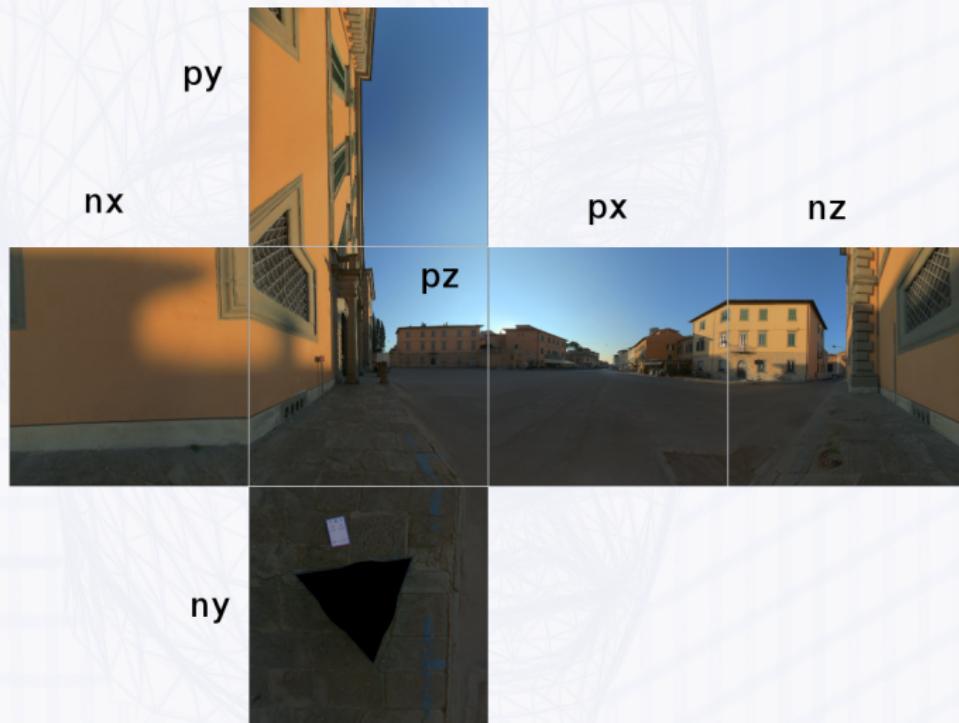
### Ejemplo: Visualización de interior y exterior con transparencias

```
var matExt = new THREE.MeshPhongMaterial () ;  
// se configura de la manera habitual y además ...  
matExt.alphaMap = unaTexturaAlfa;  
matExt.transparent = true;    matExt.side = THREE.FrontSide;  
// ahora se hace el material para el interior  
matInt = matExt.clone();      matInt.side = THREE.BackSide;  
// Se crea una figura multimaterial , el interior primero  
var figura = THREE.SceneUtils.createMultiMaterialObject (geometria ,  
[ matInt , matExt]) ;
```

# Texturas en otros canales

## Entorno

- Se usa una caja con 6 texturas en el interior, una por cara



# Texturas como entorno

## Carga de las texturas

- Las texturas se cargan en una clase específica para ello
- Y se asignan al atributo `background` de la escena

### Ejemplo: Carga de texturas para el entorno

```
var path = "textures/cube/pisa/";
var format = '.png';
var urls = [
    path + 'px' + format, path + 'nx' + format,
    path + 'py' + format, path + 'ny' + format,
    path + 'pz' + format, path + 'nz' + format
];
var textureCube = new THREE.CubeTextureLoader().load( urls );
laEscena.background = textureCube;
```

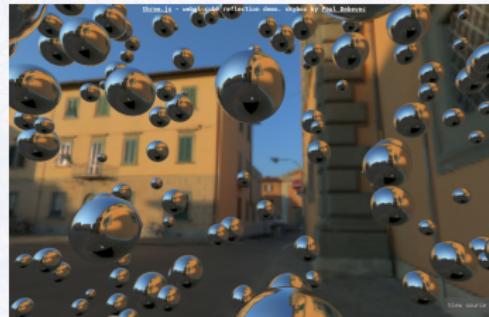
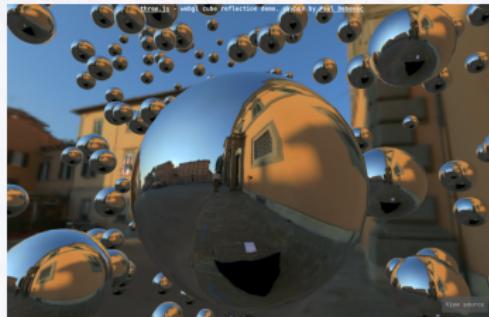
# Texturas en otros canales

## Reflexión del Entorno

- Se debe usar la CubeTexture del entorno
- Y ponerla en el canal envMap

### Ejemplo: Reflejo del entorno

```
var material = new THREE.MeshBasicMaterial( { color: 0xffffffff ,  
envMap: textureCube } );
```



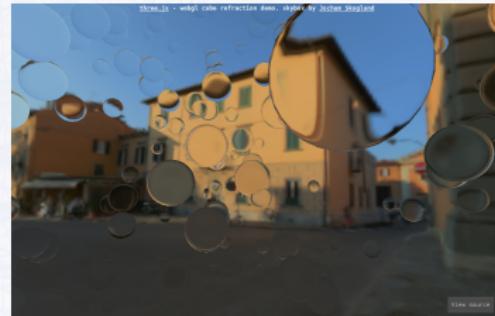
# Texturas en otros canales

## Refracción del Entorno

- Se usa también la CubeTexture del entorno
- Hay que activar el mapeo de refracción en el canal
- E indicar el índice de refracción

### Ejemplo: Reflejo del entorno

```
var material = new THREE.MeshBasicMaterial( { color: 0xffffffff ,  
    envMap: textureCube } );  
material.envMap.mapping = THREE.CubeRefractionMapping;  
material.refractionRatio = 0.95;
```



# Usar un vídeo como textura

- Se añade el vídeo en un elemento HTML
- Se toma desde ahí para asignarlo como textura a un material

## Ejemplo: Usar un vídeo como textura

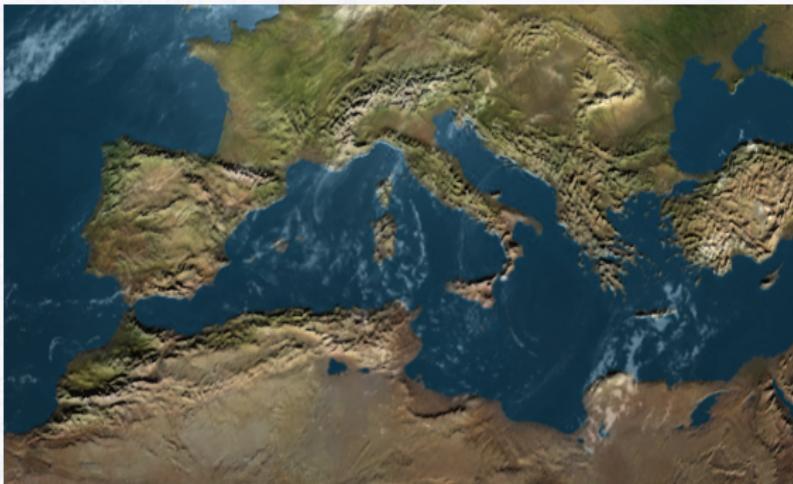
```
// La parte en HTML
<video id="video" autoplay loop style="display:none"
       src="ruta/y/fichero">
</video>

// La parte en javascript
var video = document.getElementById ('video');
texture = new THREE.VideoTexture (video);
texture.generateMipmaps = false; // si el vídeo no es cuadrado
texture.minFilter = new THREE.LinearFilter;

// La textura se usa de la manera habitual,
// asignándola al canal map de un material
```

# Ejemplo

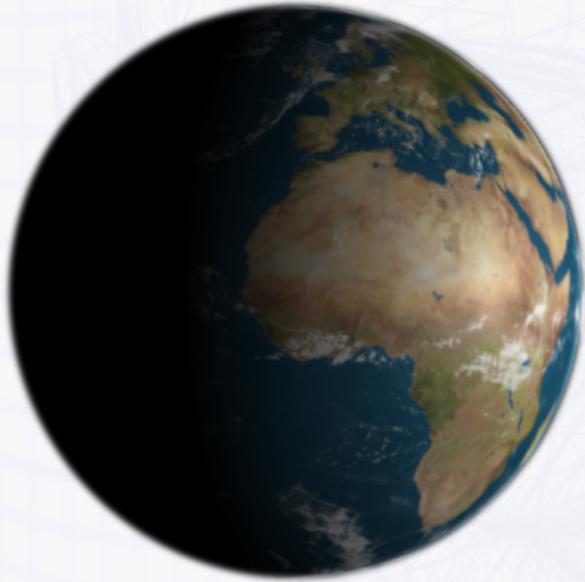
## El planeta Tierra



# El planeta Tierra

## Fuentes de luz

- Una ambiental y una direccional



# El planeta Tierra

## Textura difusa

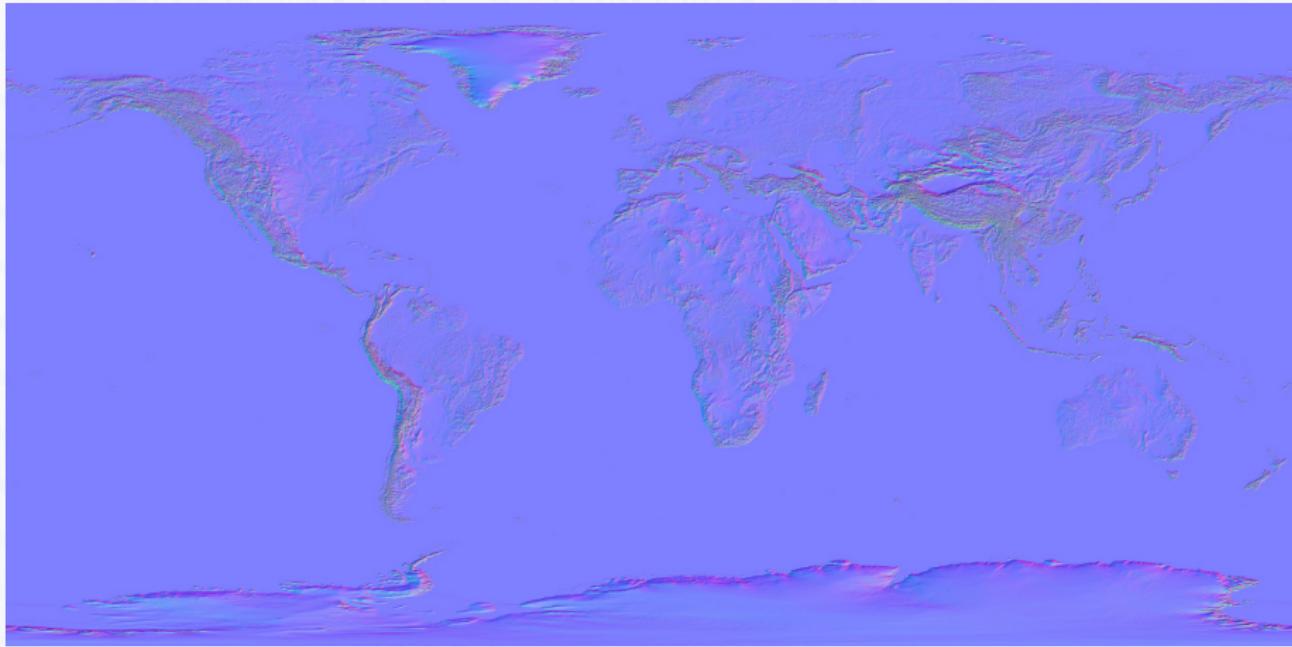
- Asignada en el atributo map



# El planeta Tierra

## Mapa de normales para el relieve

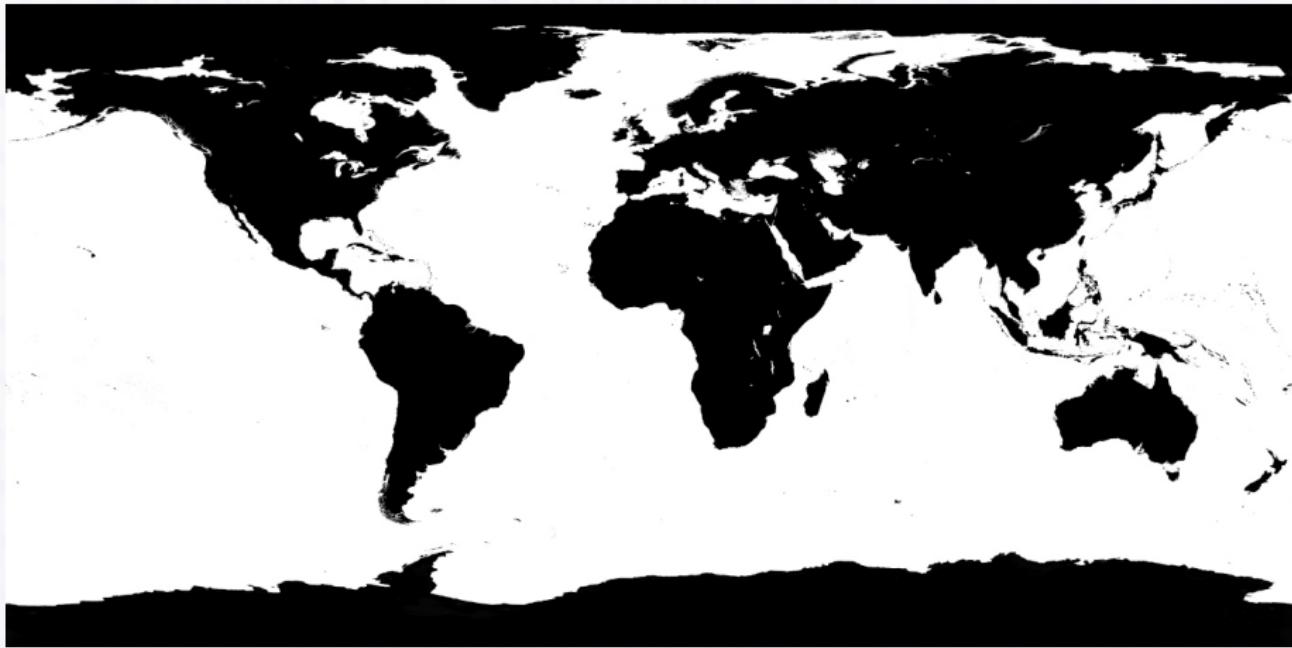
- Asignada en el atributo `normalMap`



# El planeta Tierra

Mapa en tonos de gris para la reflectividad

- Asignada en el atributo specularMap



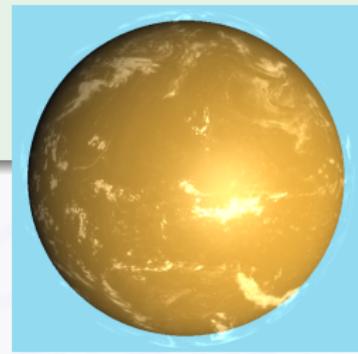
# El planeta Tierra

## Añadido de las nubes

- Se ha usado otra esfera
  - ▶ Con un radio un 15 % mayor
  - ▶ Con una velocidad de rotación un 10 % mayor
  - ▶ Con un textura difusa de nubes sobre un fondo transparente

### El planeta Tierra: Nubes

```
var cloudTexture = loader.load("ruta/nubes.png");
var cloud = new THREE.MeshPhongMaterial();
cloud.map = cloudTexture;
cloud.transparent = true;
cloud.opacity = 0.5;
cloud.blending = THREE.AdditiveBlending;
```



# El planeta Tierra



# **Visualización: Materiales**

Francisco Velasco Anguita

Dpto. Lenguajes y Sistemas Informáticos  
Universidad de Granada

Sistemas Gráficos

Grado en Ingeniería Informática  
Curso 2019-2020