Sección de Informática Gráfica VALENCIA | Computer Graphics Group

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Iluminación y Materiales

Gráficos 3D en la web

Iluminación y materiales

- La iluminación permite apreciar el color de los objetos
- Los objetos responden a la iluminación según su material
- El modelo de iluminación combina las características del material y las fuentes de luz para calcular el color
- La luces son un elemento más del grafo de escena
  - Les afecta la jerarquía de transformaciones
  - Afectan a todo el grafo

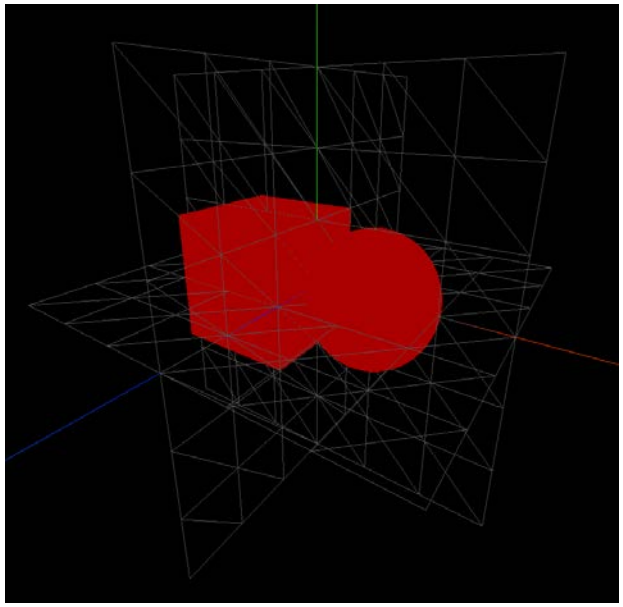
http://carvisualizer.plus360degrees.com/threejs/

Tipos de luces

- Light
  - Clase madre que deriva de Object3d
    - .position
    - .rotation
  - Propiedades
    - .color
    - .intensity [0..1]
- Clases derivadas de Light
  - AmbientLight
  - PointLight
  - DirectionalLight
  - SpotLight
  - HemisphereLight

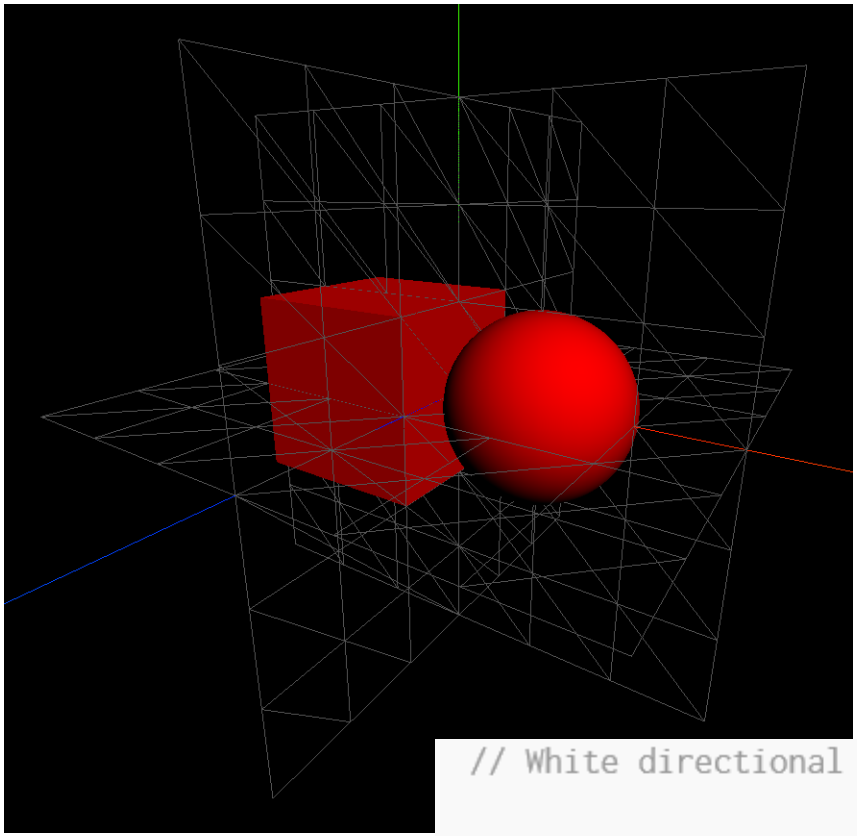
Luz ambiental

- AmbientLight(color)
  - .color
  - independiente de la posición
  - presente en todo el espacio



```
var light = new THREE.AmbientLight( 0x404040 ); // soft white light
scene.add( light );
```

Luz direccional

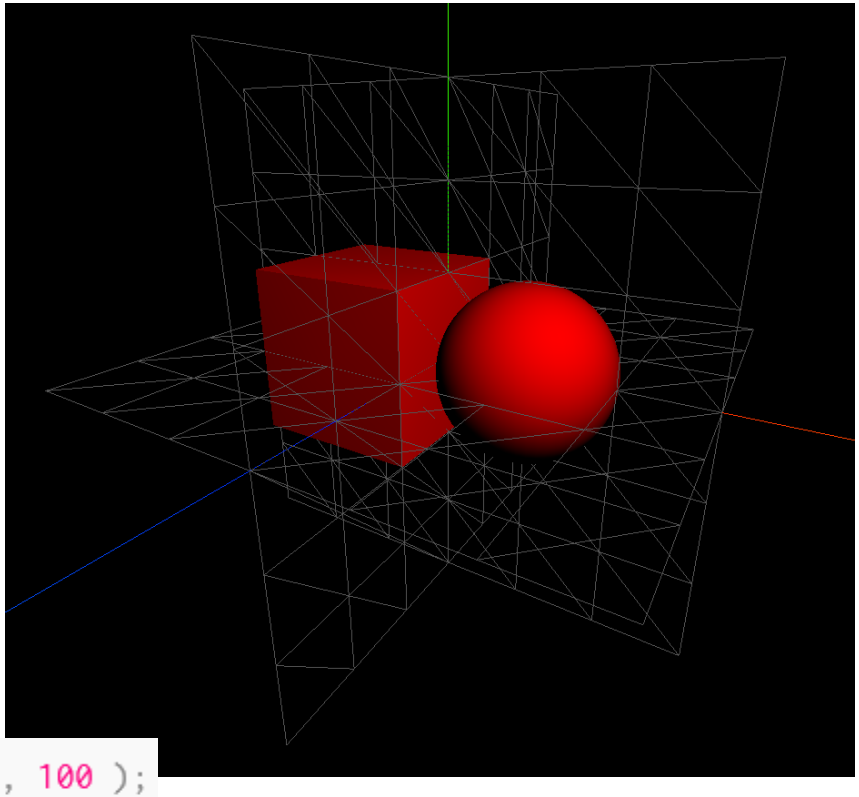


- DirectionalLight(color, intensidad)
  - .color
  - .intensity
  - .position: Vector de iluminación

```
// White directional light at half intensity shining from the top.

var directionalLight = new THREE.DirectionalLight( 0xffffff, 0.5 );
directionalLight.position.set( 0, 1, 0 );
scene.add( directionalLight );
```

Luz puntual

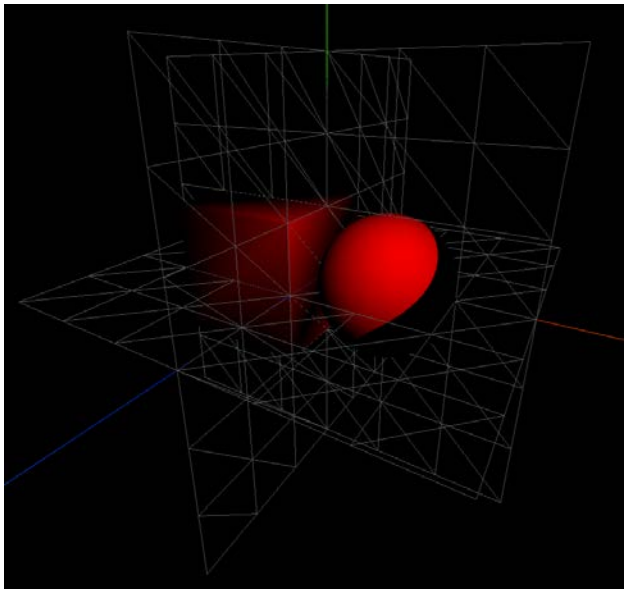
- PointLight(color, intensidad, distancia)
  - .color
  - .intensity
  - .position: posición de la luz
  - .distance: radio de alcance de la luz



```
var light = new THREE.PointLight( 0xff0000, 1, 100 );
light.position.set( 50, 50, 50 );
scene.add( light );
```

Luz focal

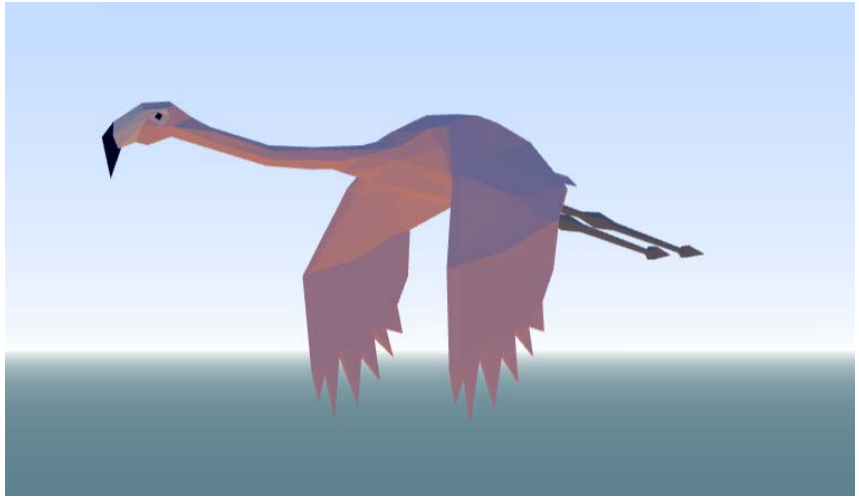
- SpotLight(color, intensidad, distancia, angulo, penumbra)
  - .color
  - .intensity
  - .distancia: radio de acción de la luz
  - .angle: ángulo del cono de luz
  - .penumbra: % del cono de luz afectado por penumbra
  - .target: objeto al que apunta el foco
- Las luces focales pueden producir sombras arrojadas
  - .shadow: información relativa a calculo de sombras por el método del doble buffer
    - .camera
      - .near
      - .far
      - .fov
    - .mapSize
      - .width
      - .height
  - .castShadow: true/false



```
var spotLight = new THREE.SpotLight( 0xffffff );
spotLight.position.set( 100, 1000, 100 );
```

Luz hemiesférica

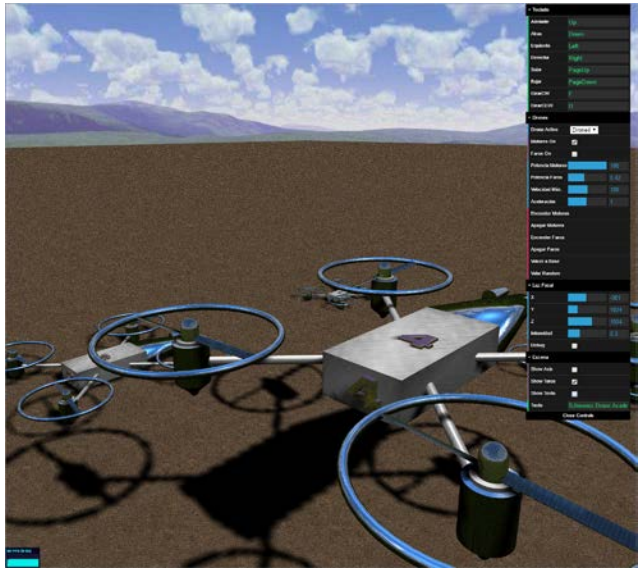
- HemisphereLight(colorCielo, colorSuelo, intensidad)
  - La luz se posiciona directamente sobre la escena
  - Los objetos se iluminan
    - por arriba del color del cielo
    - por abajo del color del suelo



```
var light = new THREE.HemisphereLight( 0xffffbb, 0x080820, 1 );
scene.add( light );
```

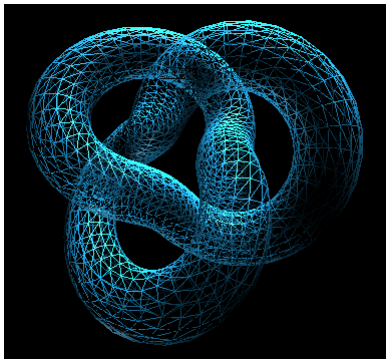
Sombras

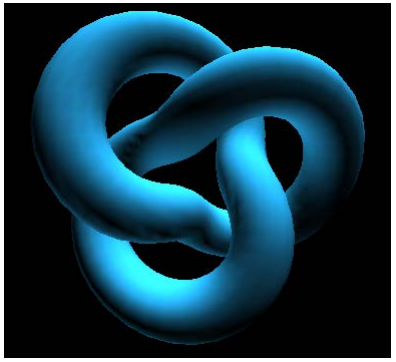
- Las luces puntuales, focales y direccionales pueden arrojar sombras
  - Se usa el algoritmo de doble buffer (visibilidad desde la fuente y el punto de vista)
  - Se deben activar las sombras en el motor, la luz y los objetos
  - La propiedad *shadow.camera* configura la vista desde la fuente



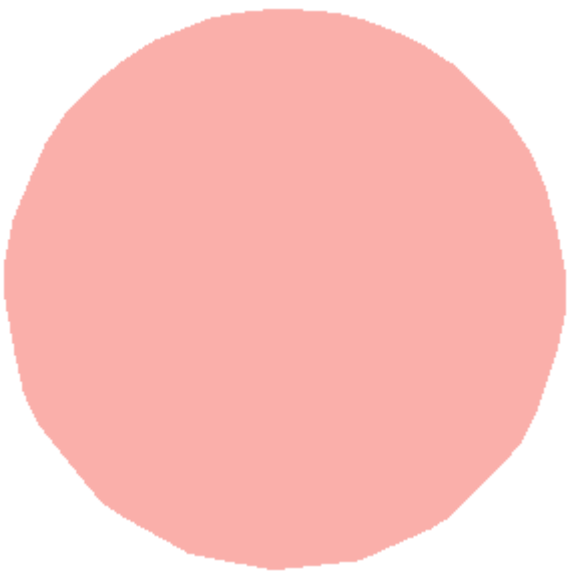
Materiales

- Material( )
  - Clase base de material
  - Propiedades
    - .opacity: 0..1
    - .transparent: true/false
    - .side: FrontSide, BackSide, DoubleSide
  - Clases derivadas
    - Para puntos
      - PointMaterial()
    - Para líneas
      - LineBasicMaterial()
      - LineDashedMaterial()
    - Mesh
      - MeshBasicMaterial()
      - MeshLambertMaterial()
      - MeshPhongMaterial()
      - MeshNormalMaterial()
      - MeshStandardMaterial()
      - ...

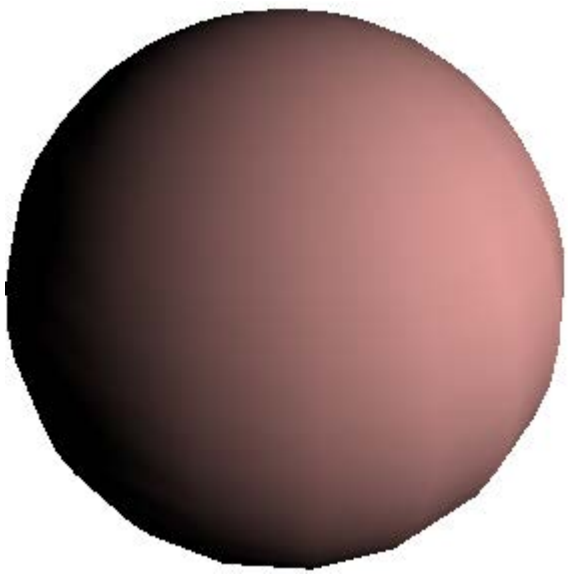




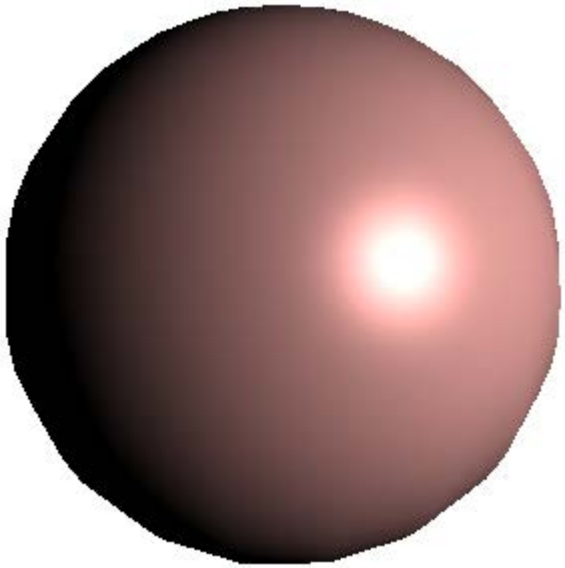
Materiales



BasicMaterial



LambertMaterial

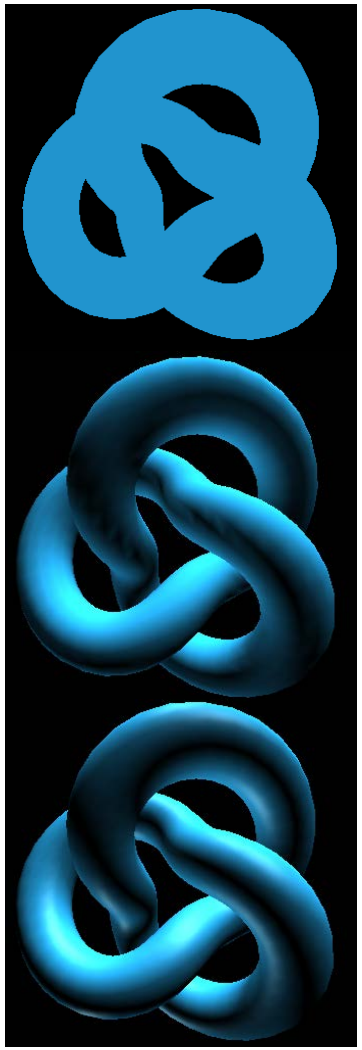


PhongMaterial

```
27    var basico = new THREE.MeshBasicMaterial( { color: 0xFAAFAA } );
28    var mate = new THREE.MeshLambertMaterial( { color: 0xFAAFAA, shading: THREE.SmoothShading } );
29    var brillante = new THREE.MeshPhongMaterial( { color: 0xFAAFAA, specular: 0xFAAFAA, shininess: 40 } );
```

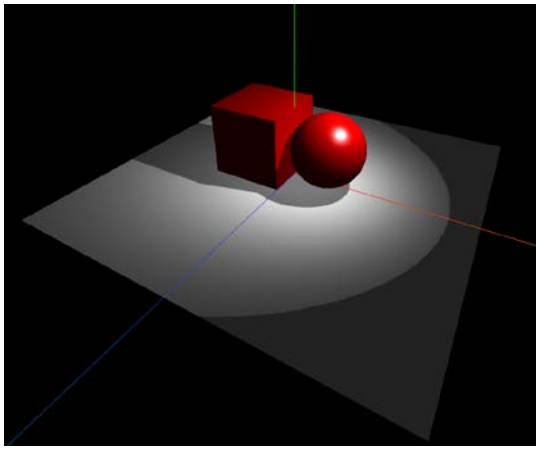
Materiales

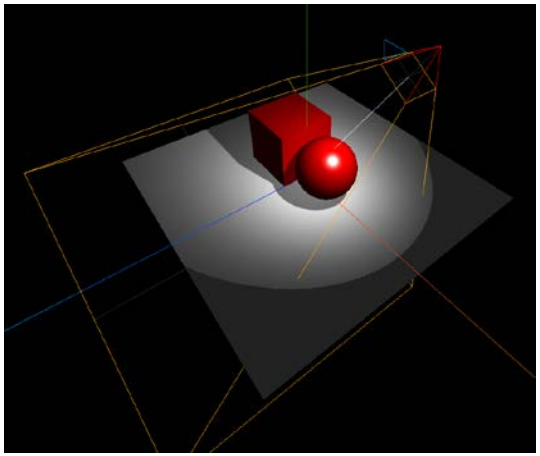
- MeshBasicMaterial({propiedades})
  - Material de color uniforme no afectado por las luces
  - .color: color del material
  - .wireframe: true/false
  - .flatShading: true/false
  - ...
- MeshLambertMaterial({propiedades})
  - Material únicamente con reflexión difusa afectado por las luces
- MeshPhongMaterial({propiedades})
  - Material con reflexión difusa y especular afectado por las luces
  - .specular: color de los brillos
  - .shininess: exponente de Phong (concentración del brillo)



Sombras (revisita)

- Las luces permiten la producción de sombras arrojadas
- Para habilitar las sombras:
  - Habilitar en el motor de render el cálculo de sombras
    - render.shadowMap.enabled= true
  - Habilitar la producción de sombras de la luz y configurar el frustum de sombra (propiedad *shadow.camera*)
    - light.castShadow = true
    - ayudante: CameraHelper()
  - Objeto que arroja sombra
    - objeto.castShadow= true
  - Objeto que recibe sombra
    - objeto.receiveShadow= true

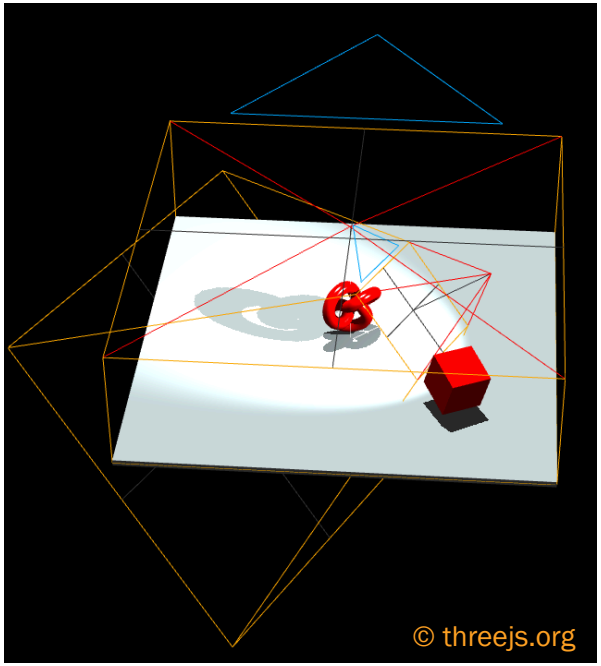




Sombras (revisita)

```
//Create a WebGLRenderer and turn on shadows in the renderer
const renderer = new THREE.WebGLRenderer();
renderer.shadowMap.enabled = true;
renderer.shadowMap.type = THREE.PCFSoftShadowMap; // default THREE.PCFShadowMap

//Create a DirectionalLight and turn on shadows for the light
const light = new THREE.DirectionalLight( 0xffffff, 1 );
light.position.set( 0, 1, 0 ); //default; light shining from top
light.castShadow = true; // default false
scene.add( light );

//Set up shadow properties for the light
light.shadow.mapSize.width = 512; // default
light.shadow.mapSize.height = 512; // default
light.shadow.camera.near = 0.5; // default
light.shadow.camera.far = 500; // default

//Create a sphere that cast shadows (but does not receive them)
const sphereGeometry = new THREE.SphereGeometry( 5, 32, 32 );
const sphereMaterial = new THREE.MeshStandardMaterial( { color: 0xff0000 } );
const sphere = new THREE.Mesh( sphereGeometry, sphereMaterial );
sphere.castShadow = true; //default is false
sphere.receiveShadow = false; //default
scene.add( sphere );

//Create a plane that receives shadows (but does not cast them)
const planeGeometry = new THREE.PlaneGeometry( 20, 20, 32, 32 );
const planeMaterial = new THREE.MeshStandardMaterial( { color: 0x00ff00 } )
const plane = new THREE.Mesh( planeGeometry, planeMaterial );
plane.receiveShadow = true;
scene.add( plane );

//Create a helper for the shadow camera (optional)
const helper = new THREE.CameraHelper( light.shadow.camera );
scene.add( helper );
```

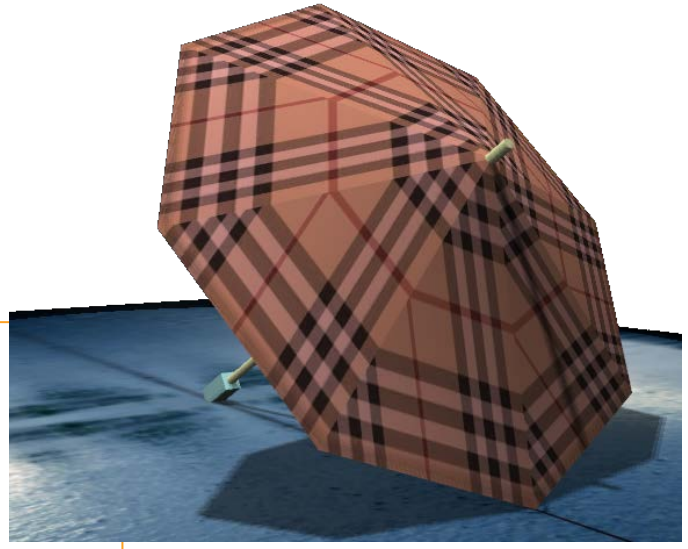
© threejs.org



Texturas de superposición

- Envuelven el objeto
- Pasos
  1. Cargar la imagen (potencia de 2)
  2. Fijar la repetición de la textura y el modo de ajuste
  3. Fijar los filtros de magnificación y minificación
  4. Incluir la textura en el material del objeto
  5. Asignar coordenadas de textura a los vértices

Chrome local --allow-file-access-from-files

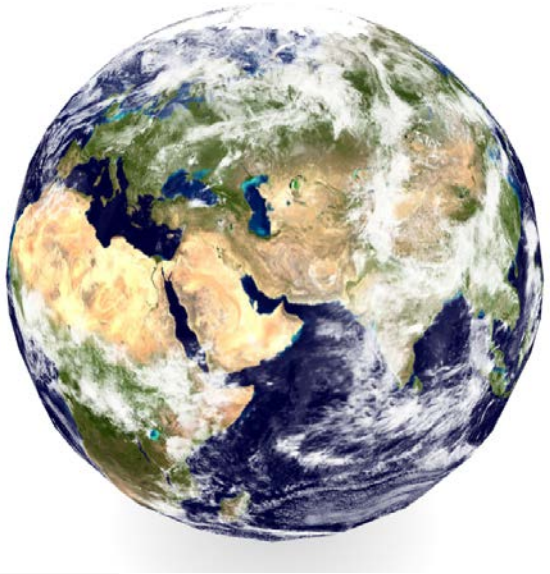


```
// CANOPY
var burberryTx = new THREE.TextureLoader().load( "./images/burberry_256.jpg");
burberryTx.repeat.set( 8, 1 );
burberryTx.wrapS = burberryTx.wrapT = THREE.RepeatWrapping;
burberryTx.magFilter = THREE.LinearFilter;
burberryTx.minFilter = THREE.LinearFilter;

var mate = new THREE.MeshBasicMaterial( { color: 0xFFFFFF,
                                          map: burberryTx } );

var tela = new THREE.Mesh( new THREE.CylinderGeometry( 0, 250.0, 100.0, 8, 1 ), mate );
tela.position.y = 450;
umbrella.add( tela );
```

Carga de textura

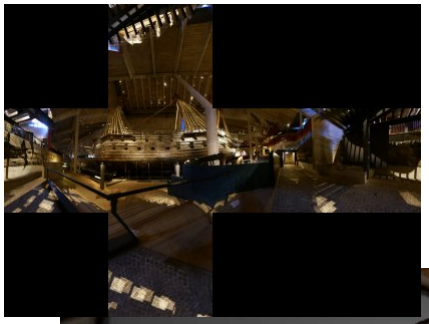
TextureLoader()

```
// instantiate a loader
var loader = new THREE.TextureLoader();

// load a resource
loader.load(
        // resource URL
        'textures/land_ocean_ice_cloud_2048.jpg',
        // Function when resource is loaded
        function ( texture ) {
                // do something with the texture
                var material = new THREE.MeshBasicMaterial( {
                        map: texture
                 } );
        },
        // Function called when download progresses
        function ( xhr ) {
                console.log( (xhr.loaded / xhr.total * 100) + '% loaded' );
        },
        // Function called when download errors
        function ( xhr ) {
                console.log( 'An error happened' );
        }
);
```



Mapas de entorno

- Mapas cúbicos formados por 6 imágenes que forman el entorno que el objeto refleja
- Se cargan con CubeTextureLoader()
- Texturas cúbicas en www.humus.name



```
var loader = new THREE.CubeTextureLoader();
loader.setPath( 'textures/cube/pisa/' );

var textureCube = loader.load( [
        'px.png', 'nx.png',
        'py.png', 'ny.png',
        'pz.png', 'nz.png'
] );

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

Habitación

- Al generar textura de mapa de entorno es conveniente generar el entorno (más realista)
- Habitación: cubo texturado por caras interiores con textura cúbica

```
const geoHabitacion = new THREE.BoxGeometry(40,40,40);
const habitacion = new THREE.Mesh(geoHabitacion, paredes);
scene.add( habitacion );
```

Array de materiales [6]

Otros ejemplos (threejs.org)

- Materiales
- Video como textura
- Sombras arrojadas
- Panorama
- Partículas
- Refraction map