# SHADOWING

## THREE.JS

Elian Gonzalez

# TABLA DE CONTENIDOS

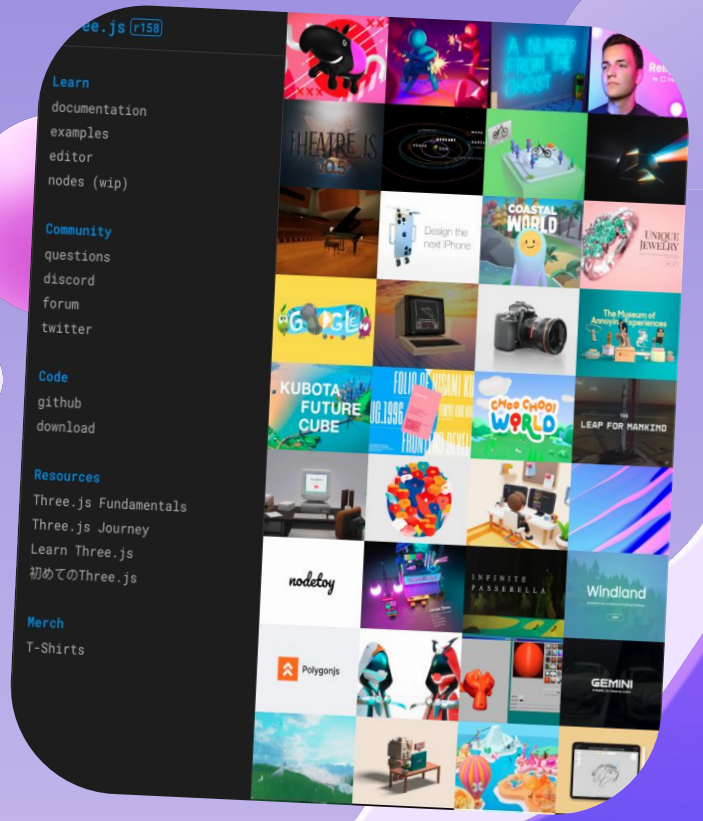
# 01

# THREE.JS

# QUE ES?

Three.js es una biblioteca JavaScript de código abierto diseñada para simplificar la creación y visualización de gráficos en 3D en navegadores web. Su objetivo es hacer que la creación de experiencias 3D sea más accesible para desarrolladores y diseñadores, aprovechando el poder de WebGL, la API de gráficos 3D de alto rendimiento.

Three.js



# COMO INSTALAR

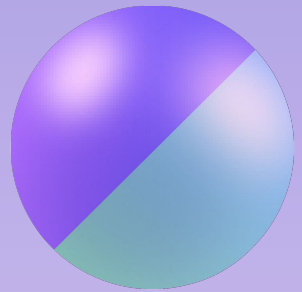
Puedes instalar Three.js desde el registro de paquetes NPM, usando node.js y una herramienta de construcción, Vite:

```
# three.js
npm install --save three

# vite
npm install --save-dev vite
```

También puedes importarlo desde un CDN:

```
<script type="importmap">
  {
    "imports": {
      "three": "https://unpkg.com/three@<version>/build/three.module.js",
      "three/addons/": "https://unpkg.com/three@<version>/examples/jsm/"
    }
  }
</script>
```



# ESTRUCTURA BASICA

Para poder mostrar cualquier cosa con three.js, necesitamos tres cosas: escena, cámara y renderizado, para poder renderizar la escena con la cámara.

- Escenas: Son el lienzo donde se desarrolla la acción.
- Cámaras: Determinan qué parte de la escena se visualiza y cómo se muestra.
- Luces: Son fuentes de iluminación que afectan la apariencia de los objetos en la escena. Pueden ser direccionales, puntuales, o tipo spot, entre otras.
- Objetos: Representan entidades en la escena, como geometría, modelos 3D, o cualquier elemento visualizable en 3D.

# ESTRUCTURA BASICA

Para poder mostrar cualquier cosa con three.js, necesitamos tres cosas: escena, cámara y renderizado, para poder renderizar la escena con la cámara.

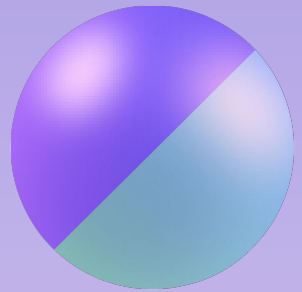
```
import * as THREE from 'three';

const scene = new THREE.Scene();
const camera = new THREE.PerspectiveCamera( 75, window.innerWidth /
window.innerHeight, 0.1, 1000 );

const renderer = new THREE.WebGLRenderer();
renderer.setSize( window.innerWidth, window.innerHeight );
document.body.appendChild( renderer.domElement );
```

# CARGAR MODELOS 3D

- Para cargar modelos 3D, three.js nos ofrece una gran variedad de “loaders” para cargar nuestro formatos, Loaders.
- Three.js nos recomienda usar el formato glTF en sus dos versiones .GLB y .GLTF ya que es el más eficiente.
- Para nuestro primer ejemplo vamos usar MTLLoader (.mtl) que nos asigna los materiales a nuestro objeto que vamos a cargar con OBJLoader(.obj).
- Usaremos castShadow y receiveShadow para proyectar la sombra del objeto.



# CARGAR MODELOS 3D

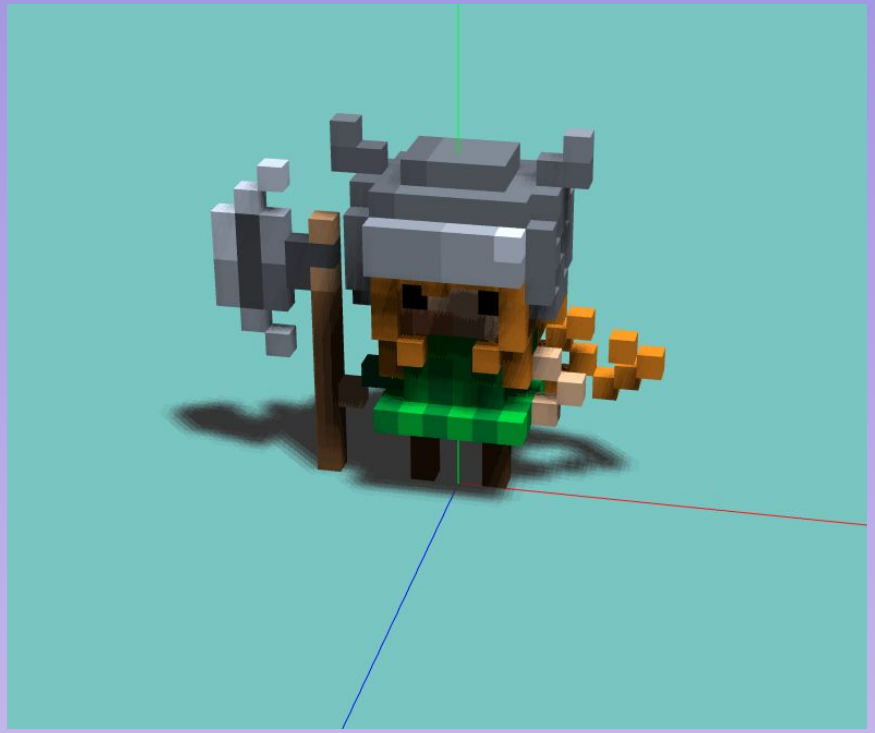
```
// Carga del modelo OBJ y MTL
const loader = new THREE.OBJLoader();
const mtlLoader = new THREE.MTLLoader();

// Cargar el archivo MTL
mtlLoader.load('modelo.mtl', function (materials) {
    materials.preload();

    // Asignar los materiales al OBJLoader
    loader.setMaterials(materials);

    // Cargar el archivo OBJ
    loader.load('modelo.obj', function (object) {
        // Añadir el modelo a la escena
        scene.add(object);
    });
});
```

# EJEMPLO OBJ Y MTL



# 02

# PCSS

Percentage closer soft shadows

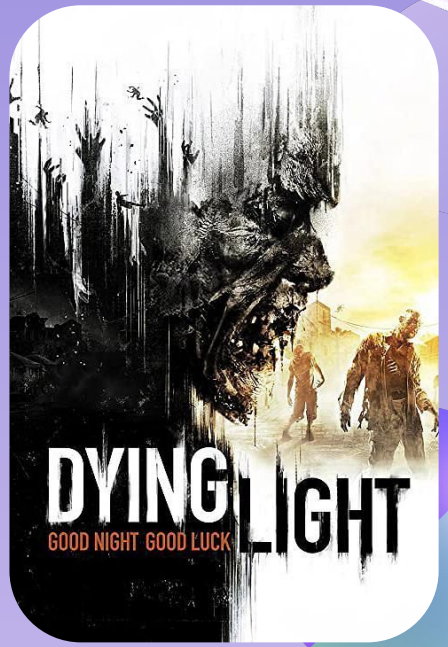
# PERCENTAGE CLOSER SOFT SHADOWS

- El algoritmo PCSS, o Sombreado de Porcentaje más Cercano y Suave, es una técnica avanzada para la generación de sombras en entornos 3D. A diferencia de las técnicas tradicionales de sombreado, PCSS simula sombras con bordes suaves y difuminados, replicando con mayor precisión la forma en que la luz se atenúa y difunde en los bordes de las sombras en el mundo real.

- Es un algoritmo de muestreo de sombras inventado por Nvidia en 2005 (documento técnico original). La intención es simular una caída más realista donde las sombras se vuelven progresivamente más suaves cuanto más se aleja el receptor del lanzador.

# DYING LIGHT

“These new and improved shadows more accurately reflect the appearance of shadows in the real world, which soften as the distance from the shadow caster increases. For example, the shadows of leaves 20 foot up a tree aren’t seen clearly on the ground with sharp outlines, and so PCSS mirrors this behavior in games.” [3]




Ejemplo 1

Ejemplo 2

Ejemplo 3

# EJEMPLO PCSS

# PCSS IMPLEMENTACION

Spector.js

# 03

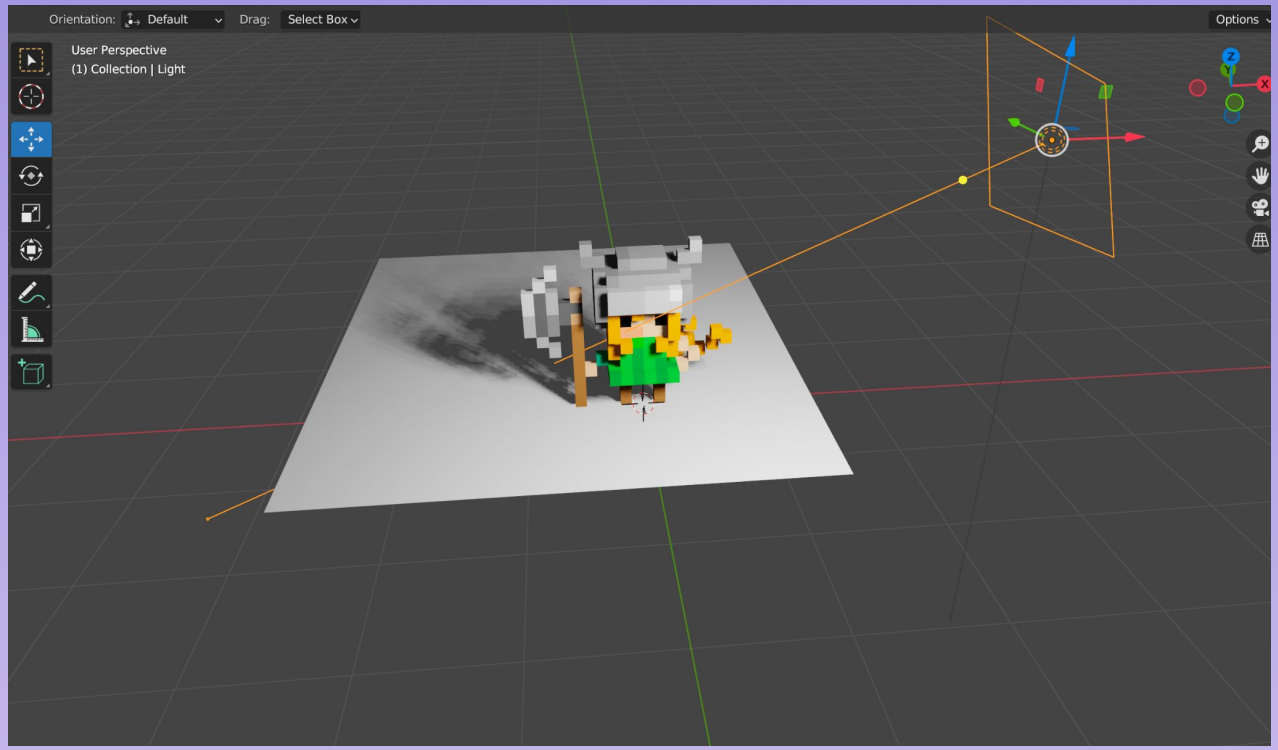
# BAKE TEXTURAS

# BAKE DE SOMBRAS BLENDER

- Consiste en pre-calcular y guardar información visual compleja, como sombras, iluminación, reflexiones o información de texturas, en una textura 2D.
- Blender lleva a cabo cálculos intensivos para capturar la información de iluminación y sombreado de una escena 3D y la proyecta sobre las superficies del modelo, almacenando esta información en una textura 2D.
- El objetivo principal del "bake textures" es reducir la carga computacional en tiempo real, al pre-calcular y almacenar información compleja en una textura.

Instructivo para bake de sombras.

# BLENDER


Orientation:
Default
Drag:
Select Box
Options
User Perspective
(1) Collection | Light

# EJEMPLO BAKE

# 04

# COMPARACION

# COMPARACIONES

- Método por defecto de Three.js: Suele ser más rápido en términos de rendimiento, ya que no requiere cálculos adicionales para suavizar los bordes de las sombras. Sin embargo, la calidad puede ser limitada.

- Algoritmo PCSS: Aunque ofrece sombras de mayor calidad, requiere más recursos computacionales debido a los cálculos adicionales para simular los bordes suaves de las sombras. Esto puede resultar en un rendimiento ligeramente más lento en comparación con el método por defecto.

- Bake sombras: Se puede llegar a obtener una escena más realista ya que lo puedes realizar en herramientas externas como blender, lo que ahorra recursos computacionales, resultando en una representación visual más eficiente en la aplicación final.

# RECURSOS

[1] Percentage-Closer Soft Shadows. Randima Fernando, NVIDIA Corporation. Extraido de https://developer.download.nvidia.com/shaderlibrary/docs/shadow_PCSS.pdf.

[2] Percentage Closer Soft Shadows Implementation. Extraido de

https://github.com/mrdoob/three.js/blob/dev/examples/webgl_shadowmap_pcss.html.

[3] dying light update introduces nvidia percentage closer soft shadows. Extraido de https://www.nvidia.com/en-us/geforce/news/gfecnt/dying-light-update-introduces-nvidia-percentage-closer-soft-shadows/.

# GRACIAS!!