Aplicación 3D + Webdev Three.js

CC3501 – Modelación y Computación Gráfica para Ingenieros

Pablo Pizarro R. @ppizarror.com

Contenidos de hoy

- Modelación de un problema “complicado”.
- Implementación de una escena que posee elementos móviles que deben detectar colisiones.
  - Implementación con OpenGL
- Breve introducción a Javascript + la librería Three.js.

Problema de hoy

- Se pide modelar una Pokebola (esfera) ubicada inicialmente en algún punto determinado con una velocidad inicial.
- En el mundo existe un plano.
- La Pokebola está sometida a la gravedad.
- Cuando comience la simulación la Pokebola debe moverse. Los rebotes con el suelo son inelásticos. Ello es, que si Vz es la velocidad antes del impacto, $-\alpha Vz$ es la velocidad posterior al impacto. $0 < \alpha < 1$.

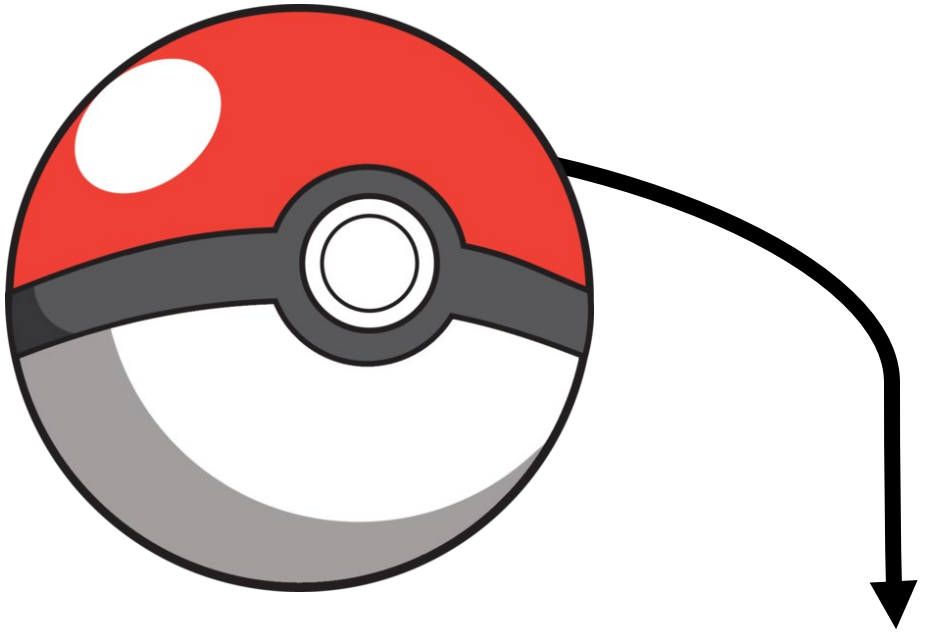
Problema de hoy - ¿Cómo modelarlo?

- Aplicar programación orientada a objetos.
- ¿Qué propiedades debe tener la Pokebola?

Problema de hoy - ¿Cómo modelarlo?

- Aplicar programación orientada a objetos.
- ¿Qué propiedades debe tener la Pokebola?
  - Posición
  - Velocidad
  - Tamaño
  - Color
  - Constante elástica α
  - Actualización de la velocidad en cada instante de tiempo (while true).
    - Utilizar cinemática para incrementar la velocidad en el eje z.
  - Actualización de la posición en cada instante de tiempo (while true).
  - Dibujar la esfera en el mundo.

Problema de hoy - ¿Cómo modelarlo?



En cada instante de tiempo:

$$Vel_{x+1} = Vel_x + a_x * \Delta t$$

$$Vel_{y+1} = Vel_y + a_y * \Delta t$$

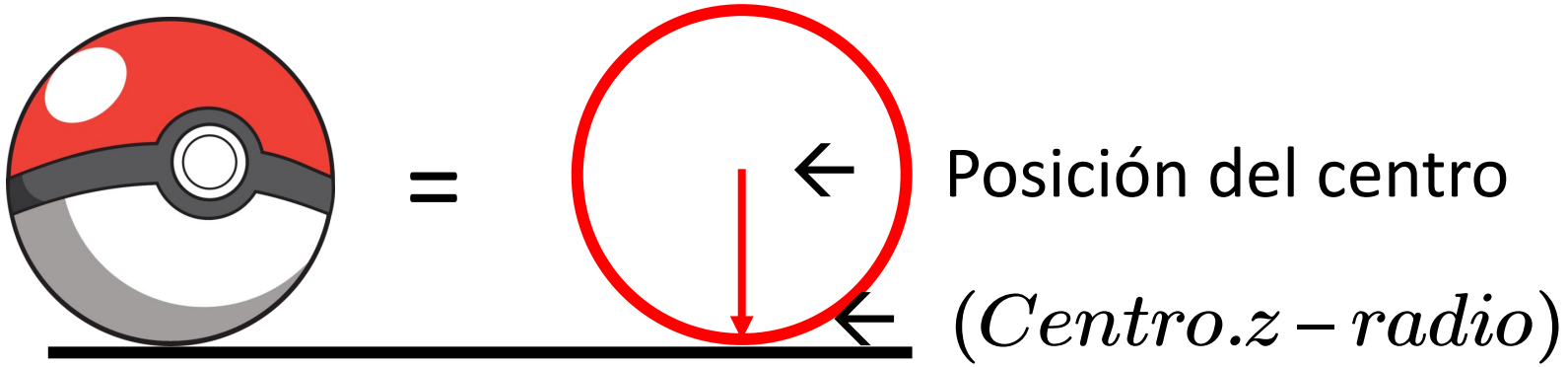
$$Vel_{z+1} = Vel_z + a_z * \Delta t$$

$$Pos_{x+1} = Pos_x + Vel_x * \Delta t$$

$$Pos_{y+1} = Pos_y + Vel_y * \Delta t$$

$$Pos_{z+1} = Pos_z + Vel_z * \Delta t$$

Problema de hoy - ¿Cómo modelar colisiones?

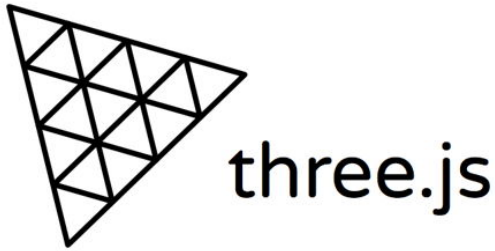


Si $(Centro.z - radio) < 0$ existe una colisión, invierte la velocidad multiplicada por el factor de elasticidad.

$Vel_z = -Vel_{z+1} * \alpha$

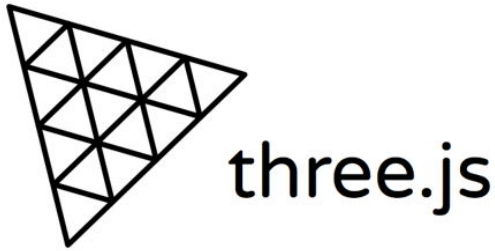
Solución en Python (PyOpenGL)

Three.js



- Librería programada en **Javascript** que permite crear complejas aplicaciones en 3D utilizando WebGL.
- Permite desarrollar aplicaciones ejecutables desde cualquier navegador web moderno.
- Extensiva documentación: https://threejs.org/docs/
- Muchos ejemplos: https://threejs.org/examples/

Javascript

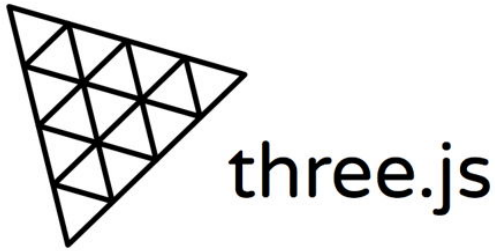


- Lenguaje de programación interpretado, orientado a objetos.
- Similar a Python.

```
function Object(){
   this.variable = 1;
   this.variable2 = “String”;
   this.lista = [1, 2, “String”, false];
   this.method = function(){
      let a = 1; // Variable global
      if (a === 1){
          return 2;
      } else {
          return false;
      }
   }
}
```

- Definición de objetos se realiza con *funciones.*
- Para declarar variables locales se utiliza **let** y globales **var**.
- Funcionamiento de listas es idéntico a Python.
- Para crear objetos utilizar **new Object(…)**

Javascript



- La estructura de objetos en Javascript se denomina JSON (Javascript object notation).

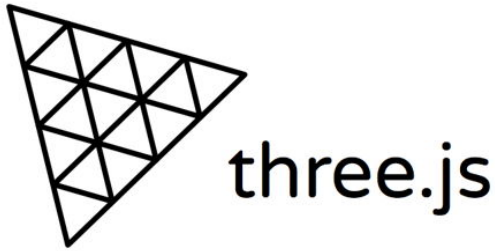
```
let modelo = {
    enabled: false,
    mesh: {
        “mesh-data”: null,
        “list-vertex”: new VertexList(),
        “list-faces”: [[1,2,3], [4,5,6]],
        “list-edges”: edges[0],
    }…
}
```

```
let a = modelo.enabled; // false
let b = modelo[“mesh”][“mesh-data”];
modelo[“mesh”][“list-edges”] = null;
```

- Objetos similares a los diccionarios de Python.

- Ver tutorial: https://www.w3schools.com/js/

Filosofía de Three.js

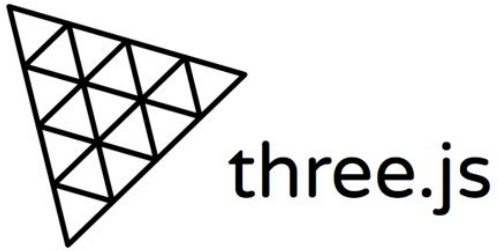


- Elementos basados en Geometrías (que heredan de *Shape*) y Materiales, los que conforman un Mesh.

```
geometry = new THREE.BoxGeometry( 0.2, 0.2, 0.2 );
material = new THREE.MeshNormalMaterial();
mesh = new THREE.Mesh( geometry, material );
```

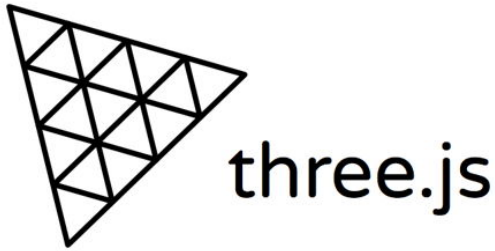
- La escena está conformada por conjuntos de Meshes.
- La cámara junto con la escena son “pasados” al renderer que despliega la escena.

Three.js en acción




```
camera = new THREE.PerspectiveCamera( 70, window.innerWidth / window.innerHeight, 0.01, 10 );
camera.position.z = 1;
scene = new THREE.Scene();

...

scene.add( mesh );
renderer = new THREE.WebGLRenderer( { antialias: true } );
renderer.setSize( window.innerWidth, window.innerHeight );
document.body.appendChild( renderer.domElement );

...

renderer.render( scene, camera );
```

Three.js en acción

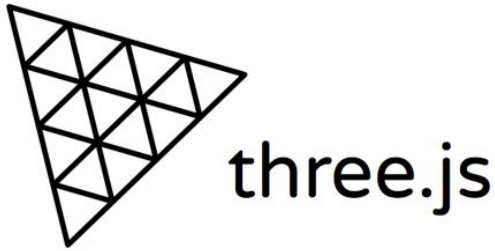


Existen múltiples maneras de crear una geometría:

1. Utilizar funciones predefinidas:
   - BoxGeometry — Cubo
   - CircleGeometry — Círculo
   - ConeGeometry — Cono
   - CylinderGeometry — Crea un cilindro
   - DodecahedronGeometry — Crea un dodecaedro
   - IcosahedronGeometry — Crea un icosaedro

   Muchos más ...

Three.js en acción

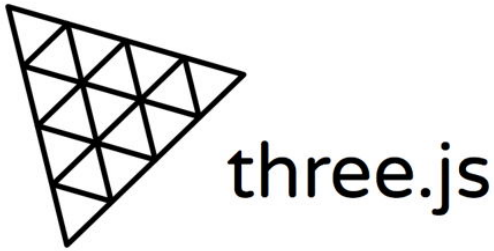


Existen múltiples maneras de crear una geometría:

2. Mediante listas de puntos:

- LatheGeometry Crea un mesh de geometría axial
- **ShapeGeometry** Crea un plano mediante una lista de puntos
  - Permite conectar puntos por rectas
  - Se pueden conectar puntos con curvas de bezier
- PlaneGeometry Crea un plano rectangular

Three.js en acción

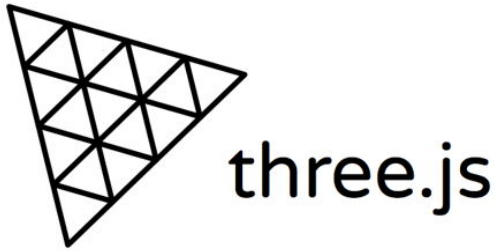


Geometrías:

Las geometrías, al ser objetos poseen múltiples campos (posición, mesh, etc). Heredan de Object3D.

https://threejs.org/docs/#api/en/core/Object3D

Three.js en acción



Geometrías:

.castShadow : Boolean

Whether the object gets rendered into shadow map. Default is false.

.children : Object3D

Array with object's children. See Group for info on manually grouping objects.

.customDepthMaterial : Material

Custom depth material to be used when rendering to the depth map. Can only be used in context of meshes. When shadow-casting with a DirectionalLight or SpotLight, if you are (a) modifying vertex positions in the vertex shader, (b) using a displacement map, (c) using an alpha map with alphaTest, or (d) using a transparent texture with alphaTest, you must specify a customDepthMaterial for proper shadows. Default is undefined.

.customDistanceMaterial : Material

Same as customDepthMaterial, but used with PointLight. Default is undefined.

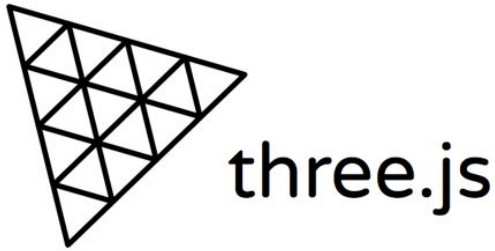
.frustumCulled : Boolean

When this is set, it checks every frame if the object is in the frustum of the camera before rendering the object. Otherwise the object gets rendered every frame even if it isn't visible. Default is true.

.id : Integer

readonly – Unique number for this object instance.

Three.js en acción



Geometrías:

.matrix : Matrix4

The local transform matrix.

.matrixAutoUpdate : Boolean

When this is set, it calculates the matrix of position, (rotation or quaternion) and scale every frame and also recalculates the matrixWorld property. Default is Object3D.DefaultMatrixAutoUpdate (true).

.matrixWorld : Matrix4

The global transform of the object. If the Object3D has no parent, then it's identical to the local transform .matrix.

.matrixWorldNeedsUpdate : Boolean

When this is set, it calculates the matrixWorld in that frame and resets this property to false. Default is false.

.modelViewMatrix : Matrix4

This is passed to the shader and used to calculate the position of the object.

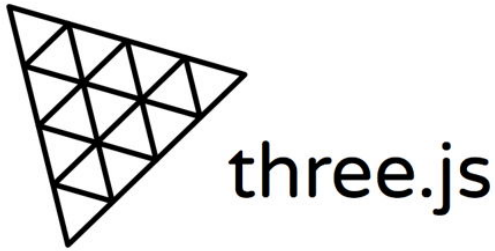
.name : String

Optional name of the object (doesn't need to be unique). Default is an empty string.

.normalMatrix : Matrix3

This is passed to the shader and used to calculate lighting for the object. It is the transpose of the inverse of the upper left 3x3 sub-matrix of this object's modelViewMatrix.

Three.js en acción



Geometrías:

.parent : Object3D

Object's parent in the scene graph. An object can have at most one parent.

.position : Vector3

A Vector3 representing the object's local position. Default is (0, 0, 0).

.quaternion : Quaternion

Object's local rotation as a Quaternion.

.receiveShadow : Boolean

Whether the material receives shadows. Default is false.

.renderOrder : Number

This value allows the default rendering order of scene graph objects to be overridden although opaque and transparent objects remain sorted independently. When this property is set for an instance of Group, all descendants objects will be sorted and rendered together. Sorting is from lowest to highest renderOrder. Default value is 0.

.rotation : Euler

Object's local rotation (see Euler angles), in radians.

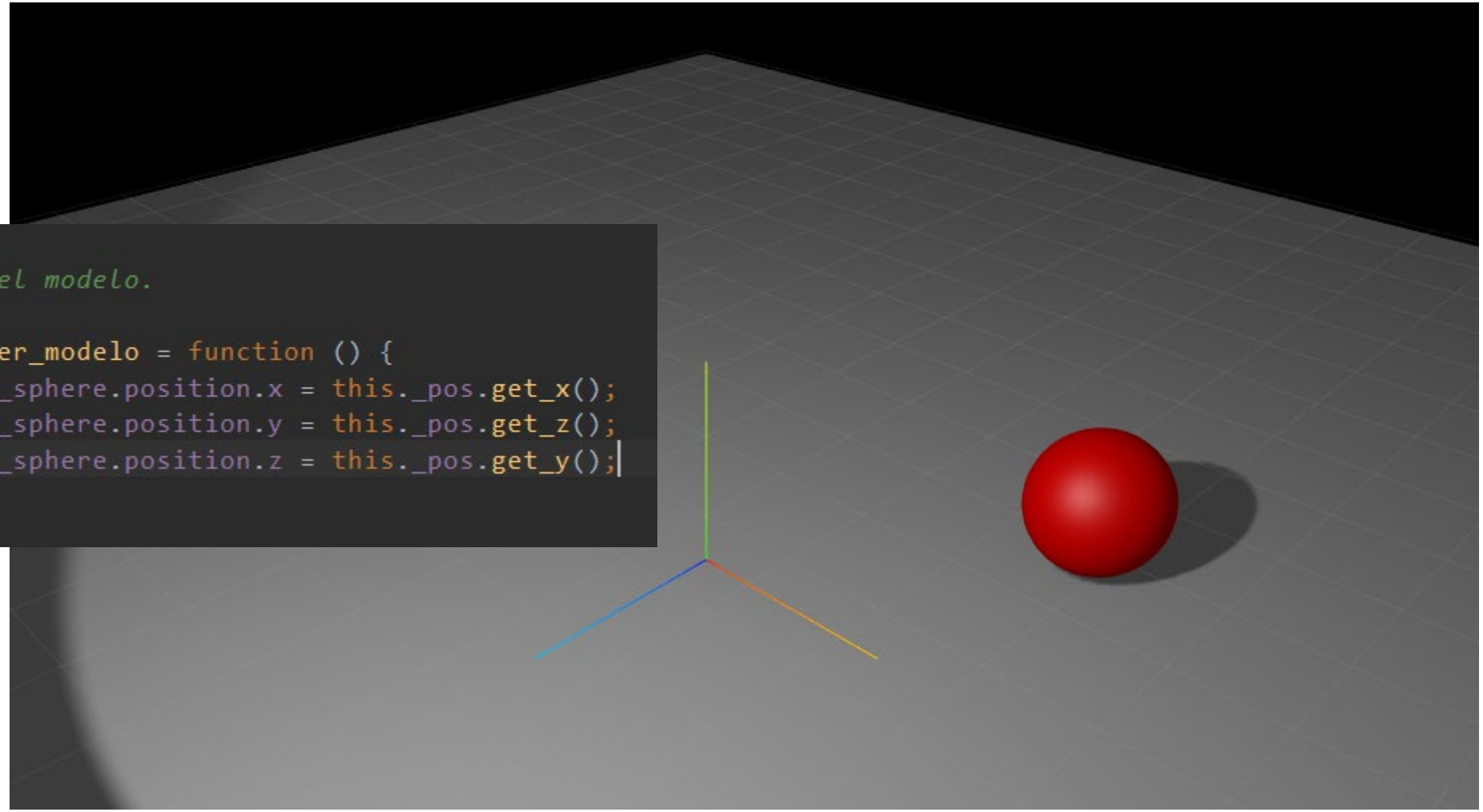
.scale : Vector3

The object's local scale. Default is Vector3(1, 1, 1).

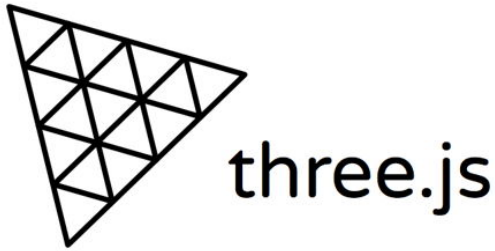
Three.js en acción

Geometrías:

```
107     /**
108      * Mueve el modelo.
109      */
110     this._mover_modelo = function () {
111         this._sphere.position.x = this._pos.get_x();
112         this._sphere.position.y = this._pos.get_z();
113         this._sphere.position.z = this._pos.get_y();
114     };
115
```



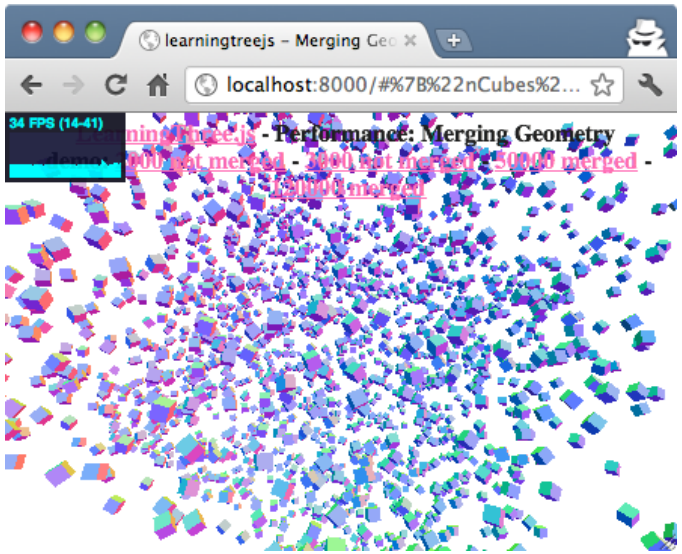
Three.js en acción

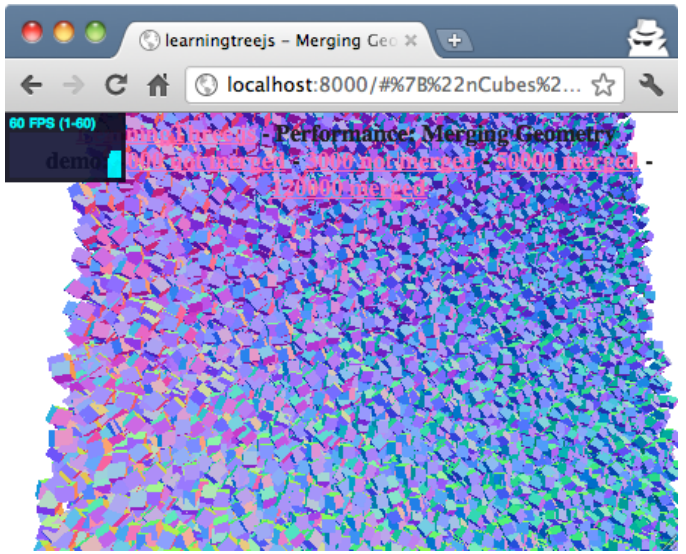


Las geometrías pueden unirse (merging) lo cual permite optimizar objetos muy complejos.

“Mientras menos datos sean intercambiados entre la CPU y la GPU mejor es el performance”

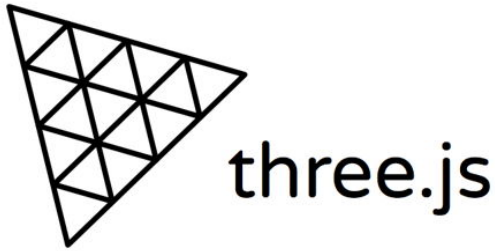
Sin merge:
34FPS





Con merge:
60FPS

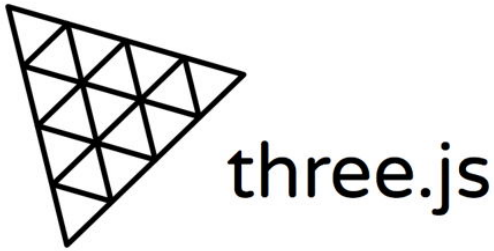
Three.js en acción



Existen múltiples materiales:

- MeshBasic
- MeshDepth
- MeshLambert
- MeshNormal
- MeshPhong
- MeshPhyscal
- MeshToon
- PointsMaterial
- RawShaderMaterial
- ShaderMaterial
- ShadowMaterial
- SpriteMaterial

Three.js en acción

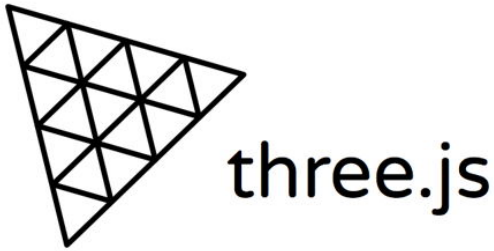


Existen múltiples materiales:

```
44      /**
45       * Crea el modelo en three.js
46       */
47      this.crear_modelo = function (scene) {
48
49          /**
50           * Crea una esfera
51           * Basado en: https://threejs.org/docs/#api/en/geometries/SphereGeometry
52           */
53          let geometry = new THREE.SphereGeometry(this._radio, widthSegments: 32, heightSegments: 32);
54          let material = new THREE.MeshPhongMaterial( parameters: {color: this._color, dithering: true});
55          this._sphere = new THREE.Mesh(geometry, material);
56          this._sphere.castShadow = true;
57          scene.add(this._sphere);
58      };
59
```

Crea la geometría (Definición de vértices y normales, topología y conectividad)

Three.js en acción

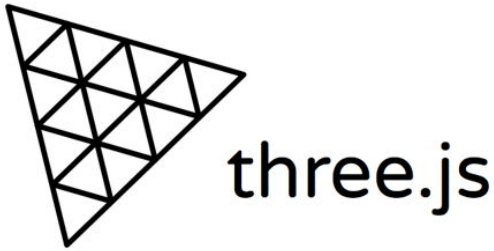


Existen múltiples materiales:

```
44    /**
45     * Crea el modelo en three.js
46     */
47    this.crear_modelo = function (scene) {
48
49        /**
50         * Crea una esfera
51         * Basado en: https://threejs.org/docs/#api/en/geometries/SphereGeometry
52         */
53        let geometry = new THREE.SphereGeometry(this._radio, 32, 32);
54        let material = new THREE.MeshPhongMaterial( parameters: {color: this._color, dithering: true});
55        this._sphere = new THREE.Mesh(geometry, material);
56        this._sphere.castShadow = true;
57        scene.add(this._sphere);
58    };
```

Crea el material, usando Shading Phong, con parámetros.

Three.js en acción

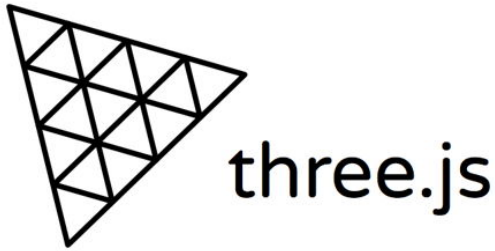


Existen múltiples materiales:

```
44    /**
45     * Crea el modelo en three.js
46     */
47    this.crear_modelo = function (scene) {
48
49        /**
50         * Crea una esfera
51         * Basado en: https://threejs.org/docs/#api/en/geometries/SphereGeometry
52         */
53        let geometry = new THREE.SphereGeometry(this._radio, widthSegments: 32, heightSegments: 32);
54        let material = new THREE.MeshPhongMaterial( parameters: {color: this._color, dithering: true});
55        this._sphere = new THREE.Mesh(geometry, material);
56        this._sphere.castShadow = true;
57        scene.add(this._sphere);
58    };
```

Crea el mesh, asigna material (shader) a cada cara de la geometría.

Three.js en acción

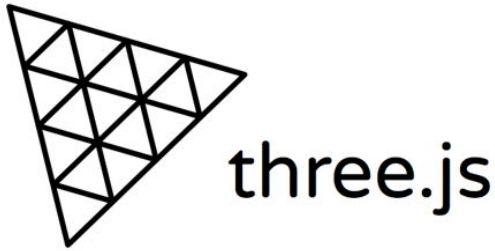


Existen múltiples materiales:

```
44      /**
45       * Crea el modelo en three.js
46       */
47      this.crear_modelo = function (scene) {
48
49          /**
50           * Crea una esfera
51           * Basado en: https://threejs.org/docs/#api/en/geometries/SphereGeometry
52           */
53          let geometry = new THREE.SphereGeometry(this._radio,  widthSegments: 32,  heightSegments: 32);
54          let material = new THREE.MeshPhongMaterial( parameters: {color: this._color, dithering: true});
55          this._sphere = new THREE.Mesh(geometry, material);
56          this._sphere.castShadow = true;
57          scene.add(this._sphere);
58      };
```

Añade el mesh (Object3D) a la escena.

Three.js en acción



Añadir luces es muy sencillo.

Three.js ofrece varias formas de crear luces, todas ellas se añaden a la escena sólo 1 vez. LUZ+MATERIAL+GEOMETRIA = Respuesta visual.

- AmbientLight
- DirectionalLight
- HemisphereLight
- PointLight
- RectAreaLight
- SpotLight

```
279
280         // noinspection JSUnusedGlobalSymbols
281         this._cameralight = new THREE.PointLight();
282         this._cameralight.color.setHex(this.objects_props.camera.light.color);
283         this._cameralight.decay = this.objects_props.camera.light.decay;
284         this._cameralight.distance = this.objects_props.camera.light.distance;
285         this._cameralight.intensity = this.objects_props.camera.light.intensity;
286         this._three_camera.add(this._cameralight);
```

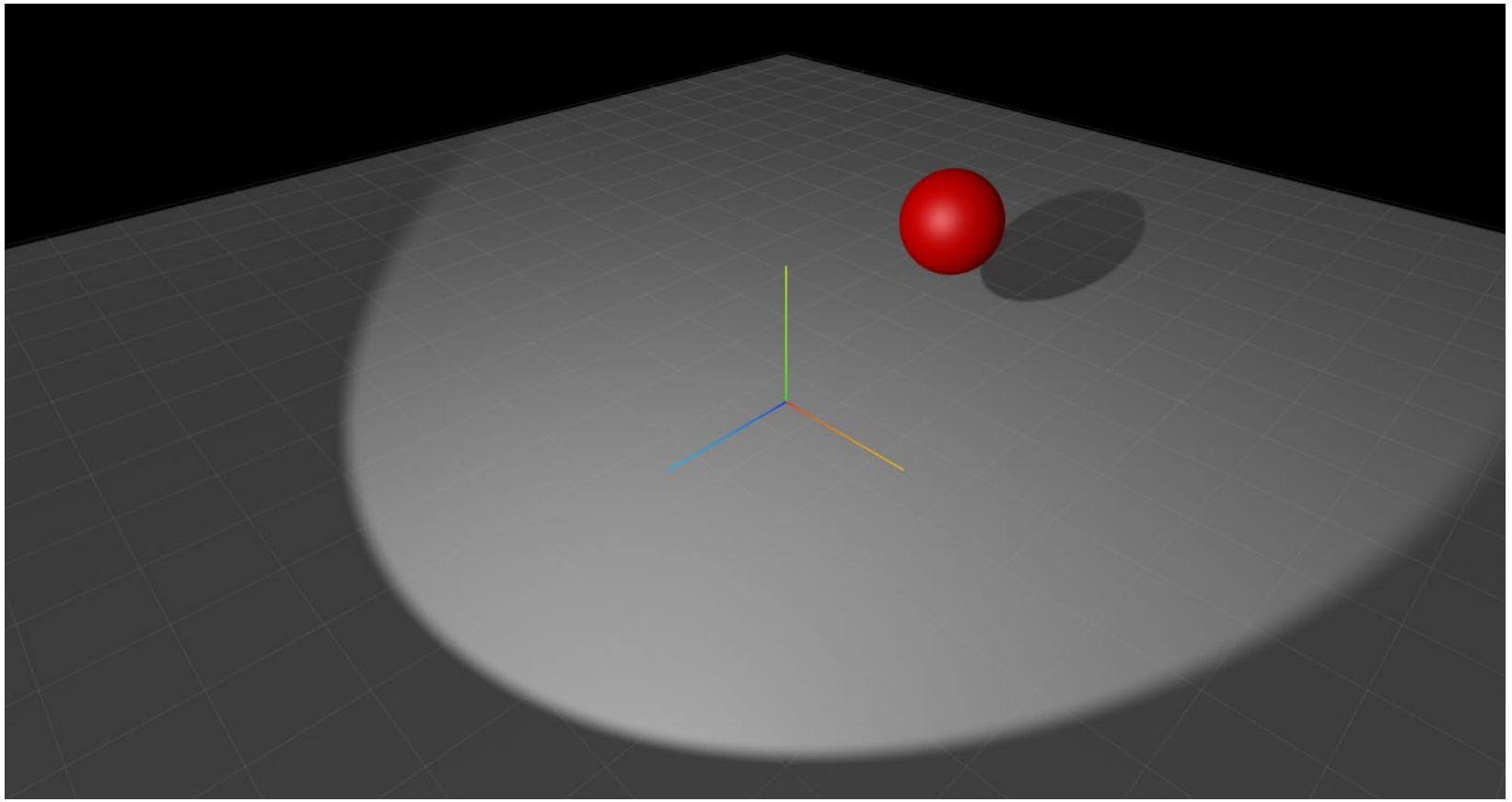
Three.js en acción

Concepto importante: Renderizador

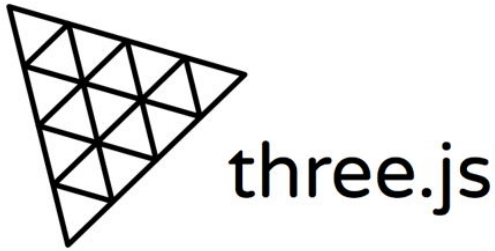
Three.js requiere definir el renderizador, ello es, la herramienta que utilizará para dibujar todo en pantalla. Se pueden usar distintos parámetros dependiendo del renderizador.

```
 * ------------------------------------------------------------------
 * Inicia el render de Three.js
 * ------------------------------------------------------------------
 */
this._renderer = new THREE.WebGLRenderer( parameters: {

    // Activa las transparencias
    alpha: false,

    // Antialias
    antialias: true,

    // Tiene un búffer de profundidad de 16 bits
    depth: true,

    // Búffer de profundidad logarítmico, usado cuando hay mucha diferencia en la escena
    logarithmicDepthBuffer: false,

    // Preferencia de WebGL, puede ser "high-performance", "low-power" ó "default"
    powerPreference: "default",

    // Precisión
    precision: 'highp',

    // Los colores ya tienen incorporado las transparencias
    premultipliedAlpha: true,

    // Para capturas, si molesta deshabilitar
    preserveDrawingBuffer: false,

    // El búffer de dibujo tiene un stencil de 8 bits
    stencil: false,

});
```

Solución en Three.js



Solución en Three.js



También puedes ejecutar la aplicación directamente desde tu navegador web (escritorio, teléfono celular, Tablet, entre otros).

https://cc3501.github.io/CC3501-2019-1/taller-threejs/threejs/

Muchas gracias por su atención

¿Preguntas?