Aplicación 3D + Colisiones + Three.js

Auxiliar N°9

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. $0 < \alpha < 1$.

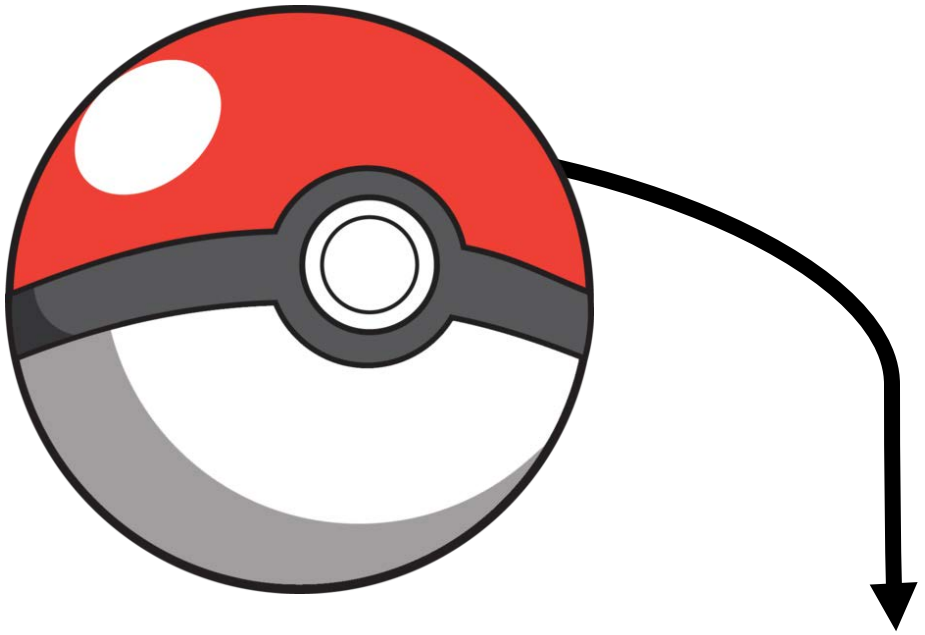
Problema de hoy - ¿Cómo modelarlo?

- 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:

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

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

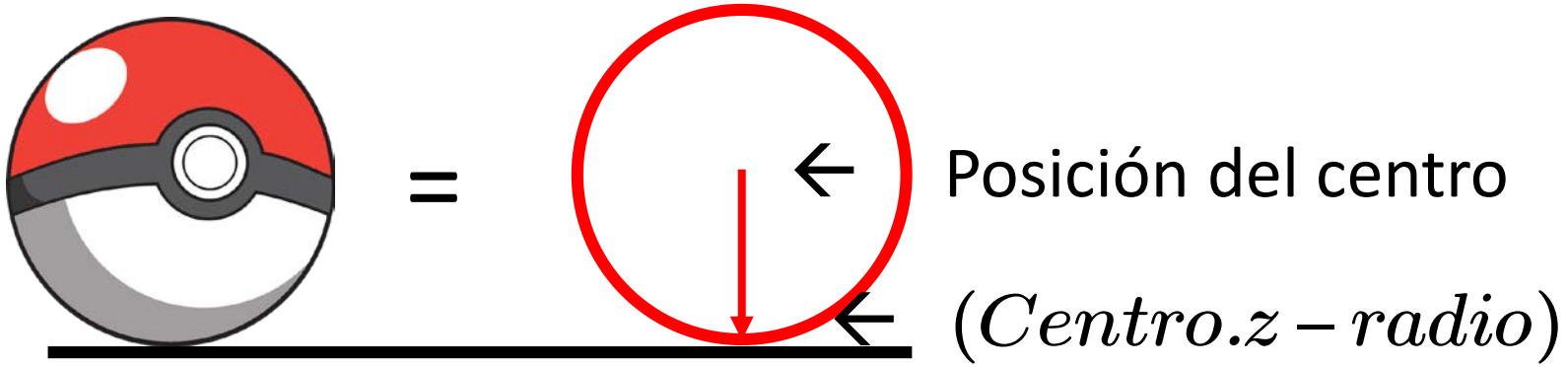
$$Vel_{z+1} = Vel_z + a_z * \Delta t$$

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

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

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

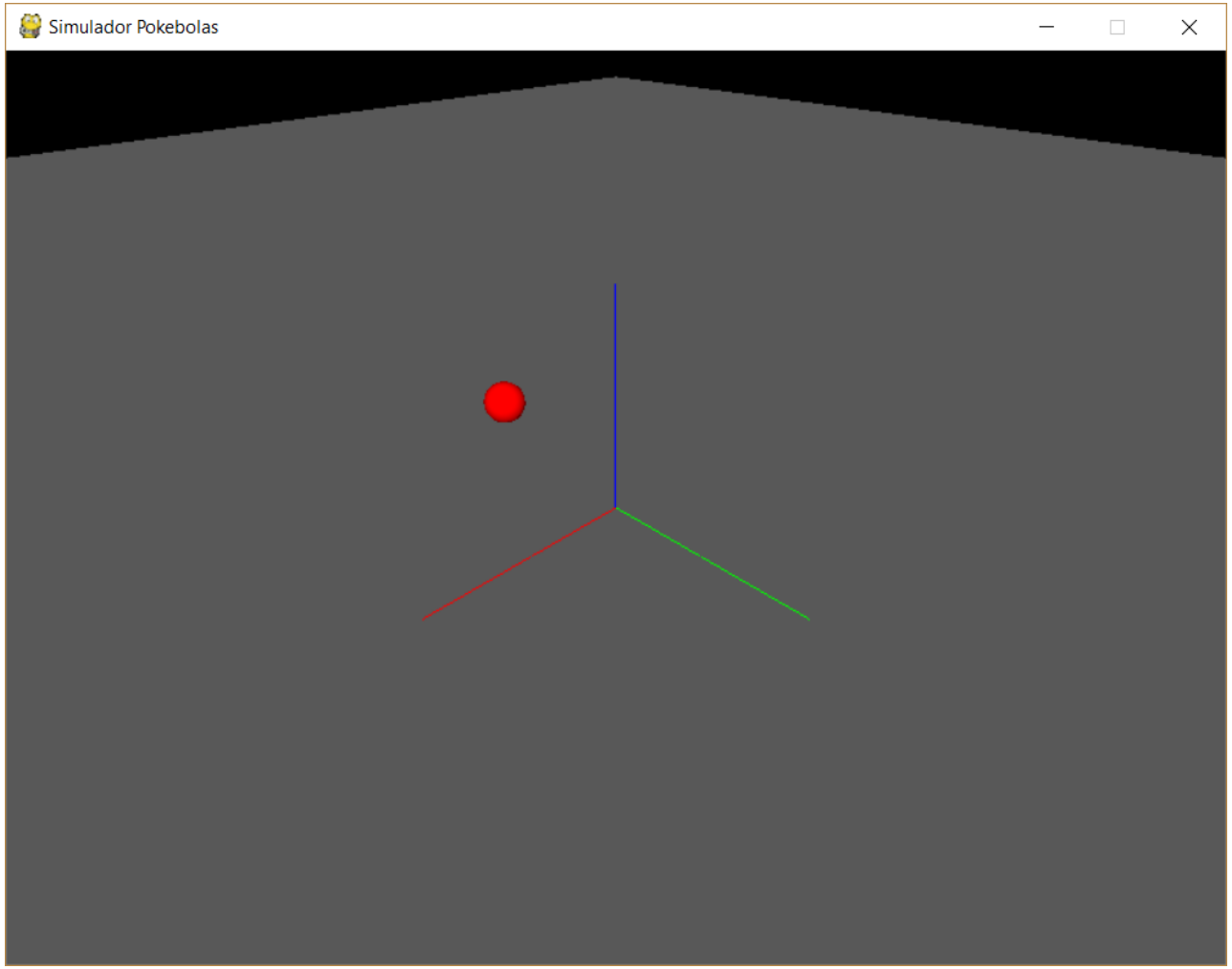
Problema de hoy - ¿Cómo modelar colisiones?



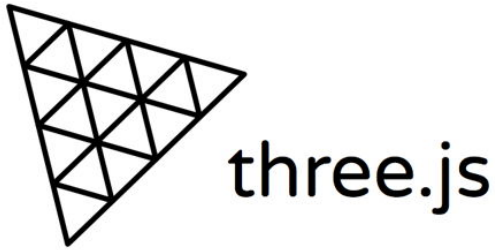
Si $(Centro.z - radio) < 0$ existe una colisión, invierte la velocidad multiplicada por el factor de elasticidad.

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

Solución en PyOpenGL

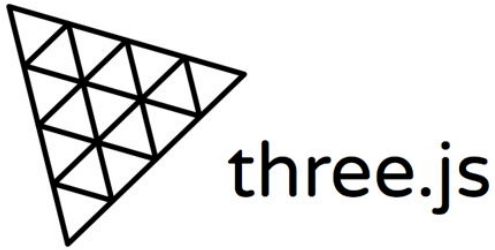


Three.js



- 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

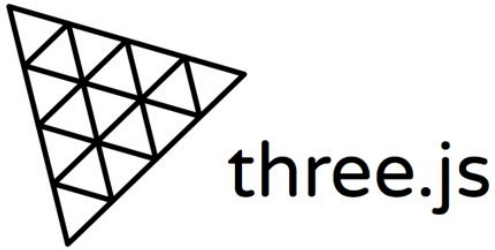


- Lenguaje de programación interpretado, orientado a objetos.
- Similar a Python.

```
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).

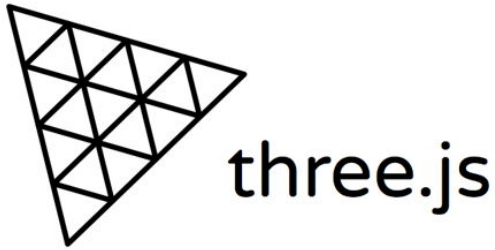
```
let modelo = {
    enabled: false,
    mesh: {
        “mesh-data”: null,
        “list-vertex”: new VertexList(),
        “list-faces”: [[1,2,3], [4,5,6]],
        “list-edges”: edges[0],
    }…
}
```

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

- Objetos similares a los diccionarios de Python.

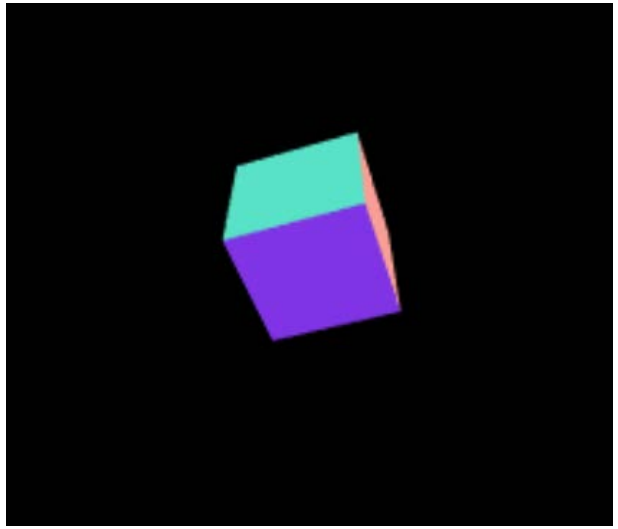
- Ver tutorial: https://www.w3schools.com/js/

Filosofía de Three.js



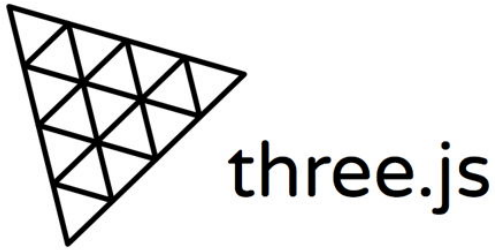
- Elementos basados en Geometrías (que heredan de *Shape*) y Materiales, los que conforman un Mesh.

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



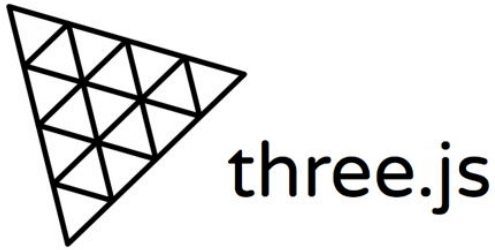
- La escena está conformada por conjuntos de Meshes.
- La cámara junto con la escena son “pasados” al renderer que despliega la escena.

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

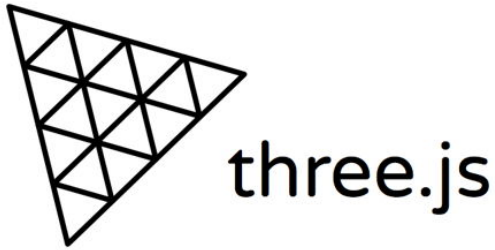


Existen múltiples maneras de crear una geometría:

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

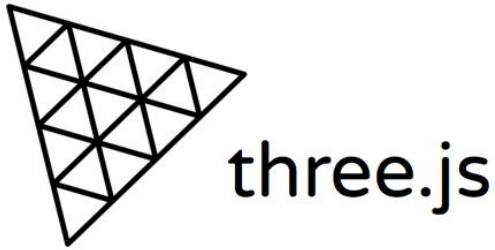


Existen múltiples maneras de crear una geometría:

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

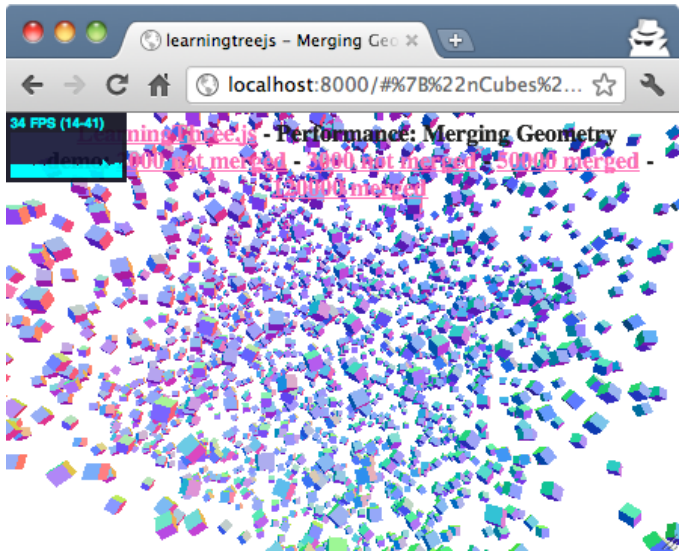
Three.js en acción

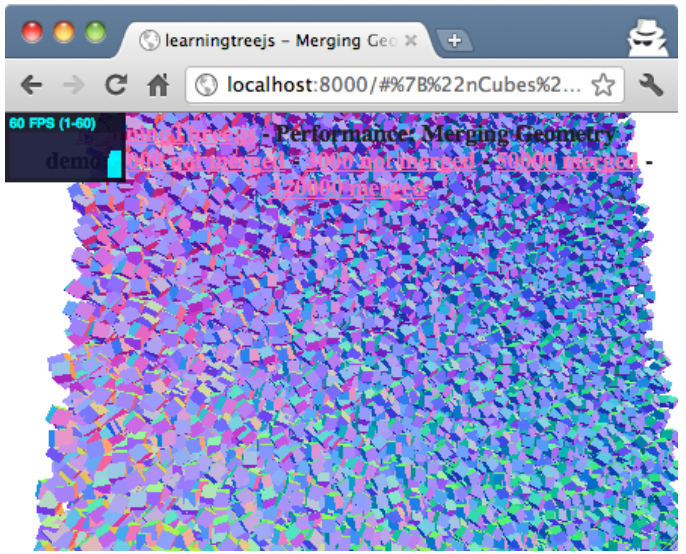


Las geometrías pueden unirse (merging) lo cual permite optimizar objetos muy complejos.

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

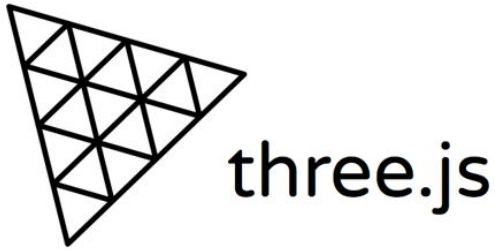
Sin merge:
34FPS





Con merge:
60FPS

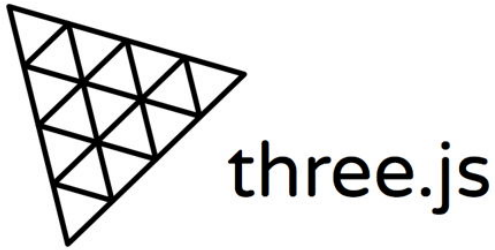
Three.js en acción



Existen múltiples materiales:

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

Three.js en acción

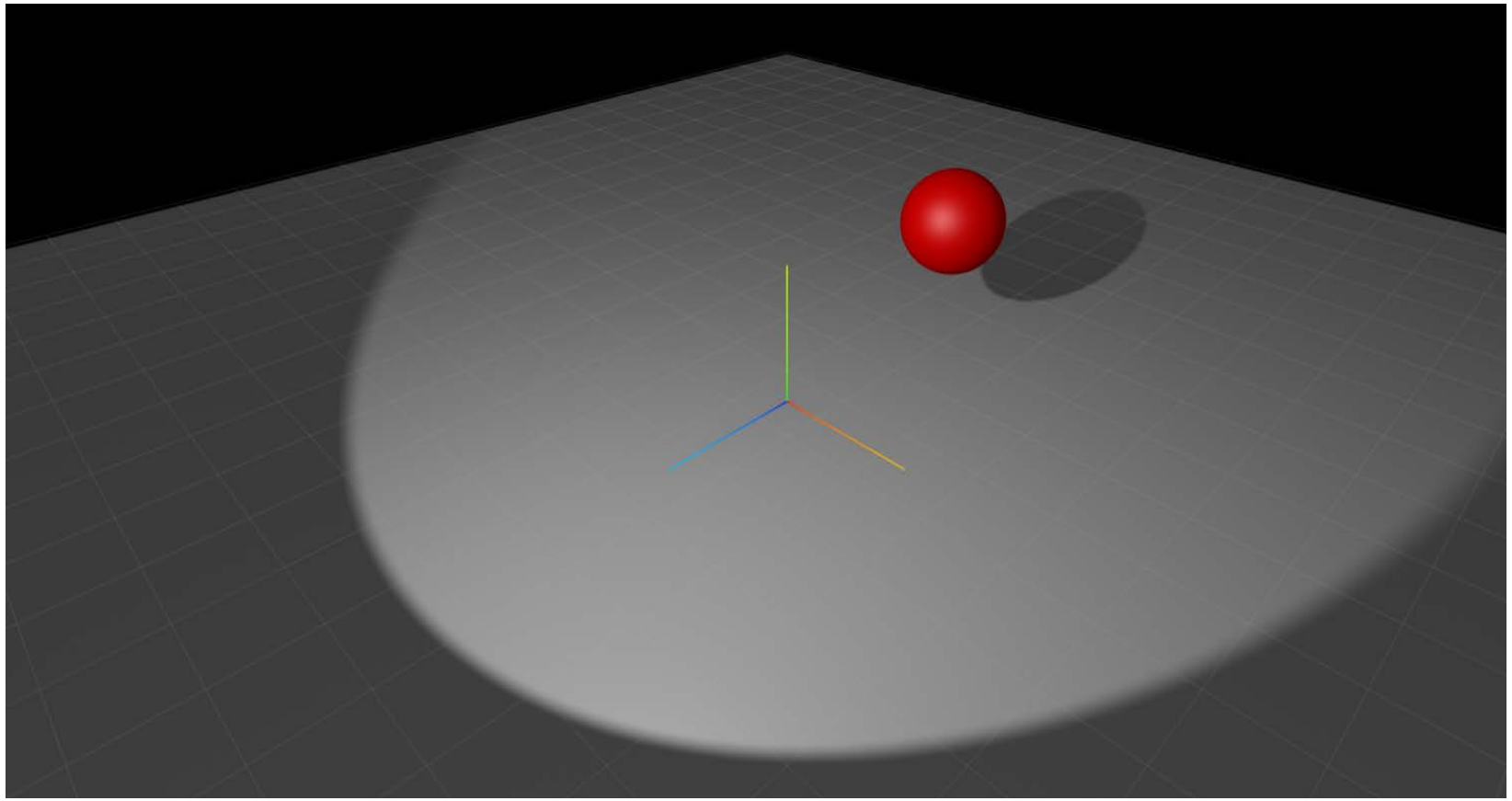


Añadir luces es muy sencillo.

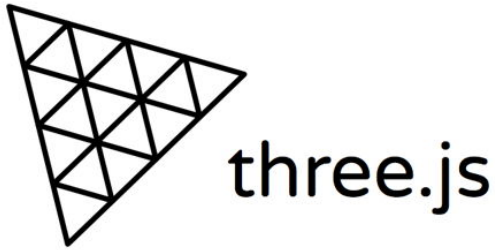
Three.js ofrece varias formas de crear luces, todas ellas se añaden a la escena sólo 1 vez. LUZ+MATERIAL+GEOMETRIA = Respuesta visual.

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

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/CC3501-2018-2/aux%209/threejs/index.html

Muchas gracias por su atención

¿Preguntas?