Sección de Informática Gráfica VALENCIA | Computer Graphics Group
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

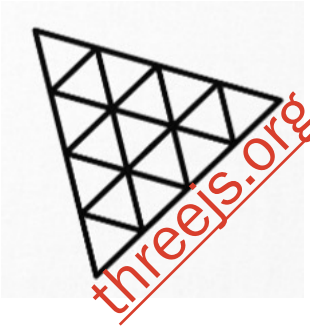
Introducción al Three.js

Gráficos 3D en la web

Three.js

- ¿Qué es Three.js?

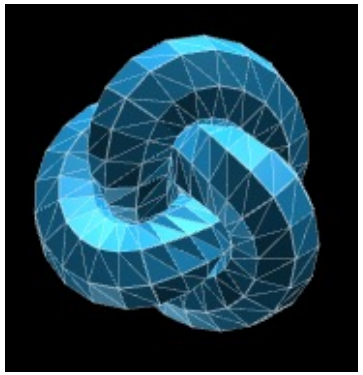
"Three.js is a library that makes WebGL - 3D in the browser - easy to use. While a simple cube in raw WebGL would turn out hundreds of lines of Javascript and shader code, a Three.js equivalent is only a fraction of that."

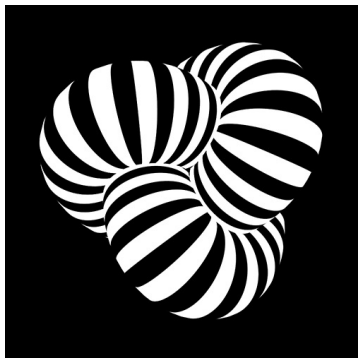


5 years of three.js

Three.js

- ¿Qué ofrece Three.js?
  - Clases predefinidas
    - Cámaras
    - Luces
    - Materiales
    - Geometría
    - Texturas
    - Cargadores ….
  - Oculta el trabajo de bajo nivel
    - Contexto de render
    - Construcción de shaders
    - Composición de transformaciones …
  - **Grafo de escena** basado en Objetc3D
    - Jerarquía de objetos
    - Transformaciones particulares a nodos





Documento esquema para Three.js

```
<!doctype html>
<html lang="es">
        <head>
                <meta charset='utf-8'>
                <title> Threejs App </title>
                <style>
                        body {
                                background-color: #000;
                                margin: 0px;
                                overflow: hidden;
                        }
                </style>
        </head>
        <body>
                <!-- Contenedor del app -->
                <div id="container"></div>

                <!-- Scripts de biblioteca -->
                <script src="lib/three.min.js"></script>

                <!-- Mi aplicacion -->
                <script>
                        // Codigo de la aplicacion
                </script>

        </body>
</html>
```

Threejs_r140.

Script básico en Three.js

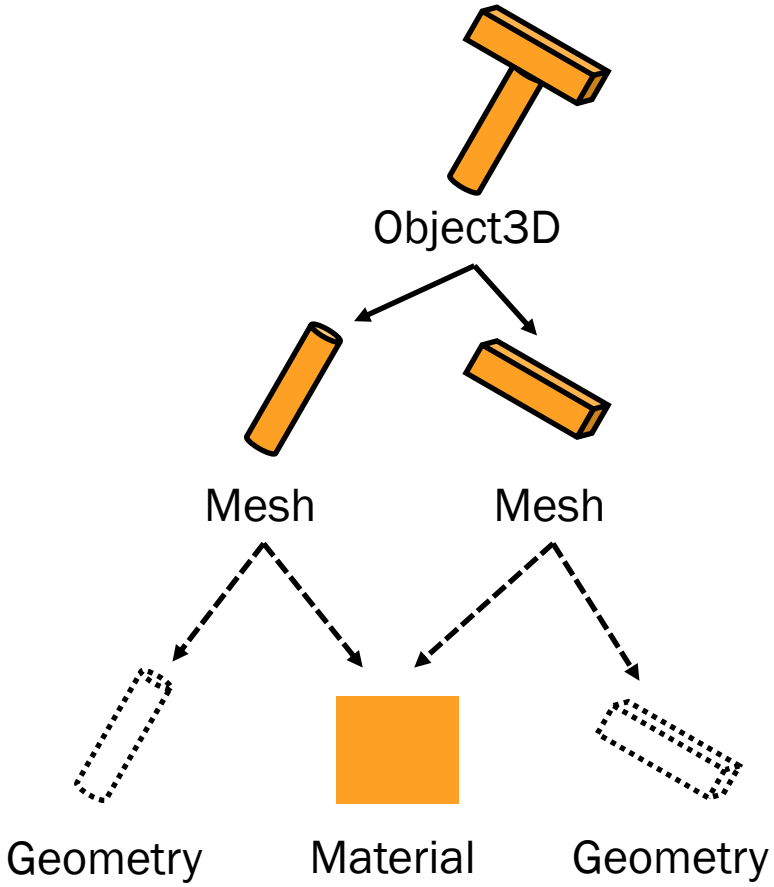
- Instanciar contexto de render, escena y cámara
- Crear el bucle de refresco

```
1  var renderer, scene, camera;
2
3  function init()
4  {
5      renderer = new THREE.WebGLRenderer();
6      renderer.setSize( window.innerWidth, window.innerHeight );
7      renderer.setClearColor( new THREE.Color(0x0000AA), 1.0 );
8      document.body.appendChild( renderer.domElement );
9
10     scene = new THREE.Scene();
11
12     var aspectRatio = window.innerWidth / window.innerHeight;
13     camera = new THREE.PerspectiveCamera( 75, aspectRatio , 0.1, 100 );
14     camera.position.set( 0, 2, 3 );
15 }
16
17 function update()
18 {
19
20 }
21
22 function render()
23 {
24     requestAnimationFrame( render );
25     update();
26     renderer.render( scene, camera );
27 }
28
29 init();
30 render();
```

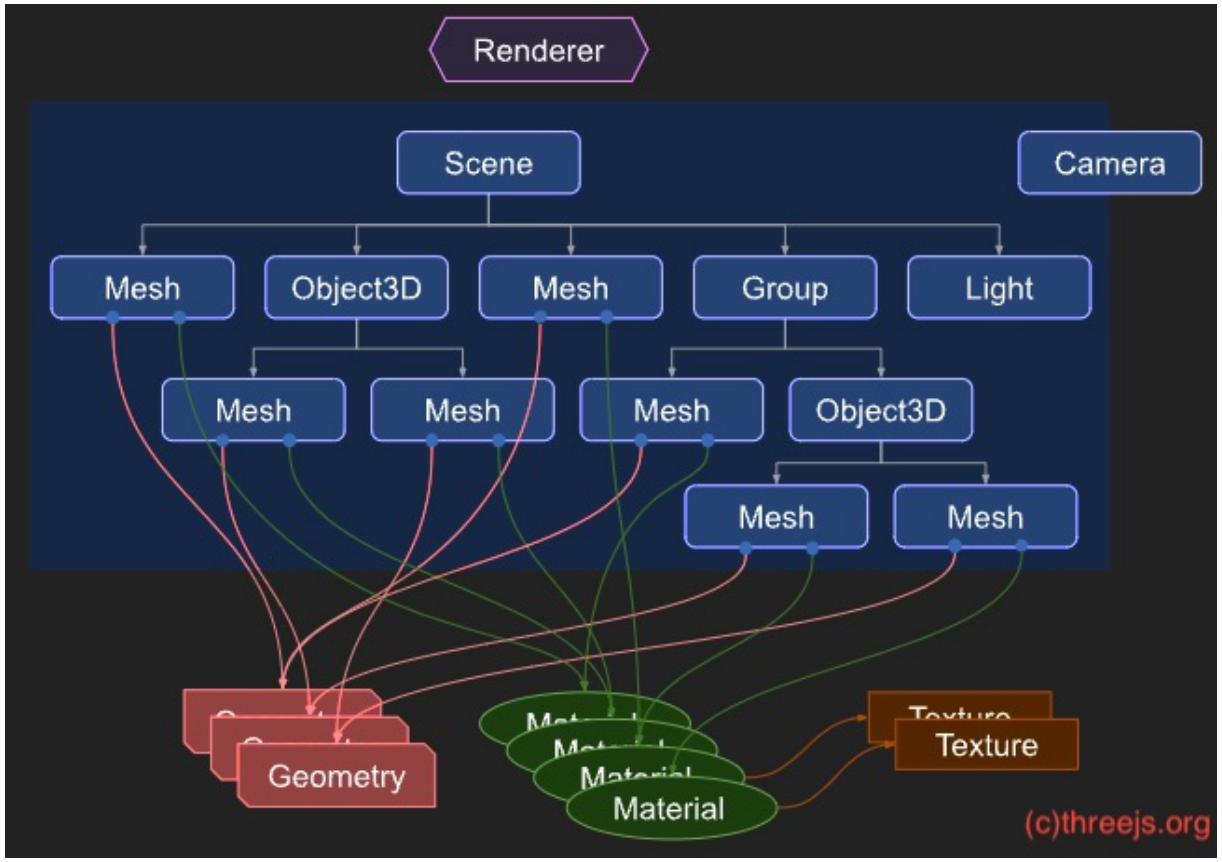
THREE.js: fovy en grados

Objetos 3D en Three.js

- Object3D es un nodo contenedor
  - formas Mesh
  - otros Object3D
  - responde al método add()
- Mesh es un nodo hoja que asocia
  - una geometría
  - un material
- Geometría
  - clase base: BufferGeometry
  - predefinidas:
    - cubo, cilindro, esfera, toro, etc (p.e. BoxGeometry(lx,ly,lz))
- Material
  - clase base: Material
  - predefinidos:
    - básico, Lambert, Phong, etc



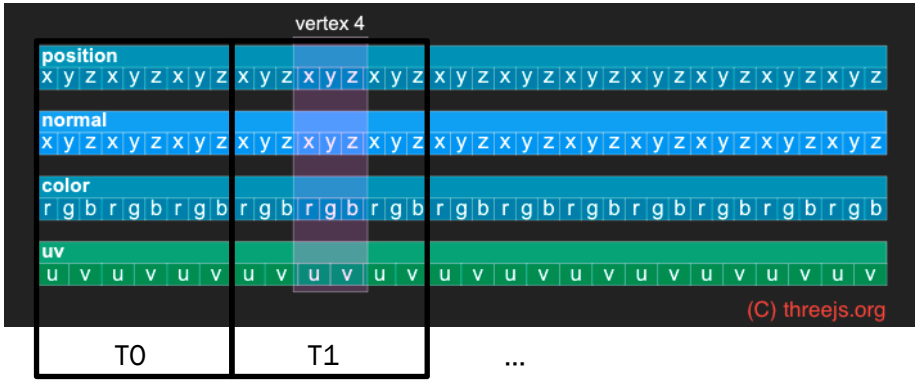
Grafo de escena



THREE.BufferGeometry : implícita en el orden

```
// Instanciar geometría propia
const geometry = new THREE.BufferGeometry();
// Construir los arrays
const position = new Float32Array( [
// x,y,z,x,y,z,...
] );
const normal = new Float32Array( [
// x,y,z,x,y,z,...
] );
const color = new Float32Array( [
// r,g,b,r,g,b,...
] );
const uv = new Float32Array( [
// u,v,u,v,,...
] );
// Construir los VBOs en la geometria
geometry.setAttribute( 'position', new THREE.BufferAttribute( position, 3 ) );
geometry.setAttribute( 'normal', new THREE.BufferAttribute( normal, 3 ) );
geometry.setAttribute( 'color', new THREE.BufferAttribute( color, 3 ) );
geometry.setAttribute( 'uv', new THREE.BufferAttribute( uv, 2 ) );
// Construir la forma
const material = new THREE.MeshBasicMaterial( );
const mesh = new THREE.Mesh( geometry, material );
```

si color por vértice {vertexColor:true}

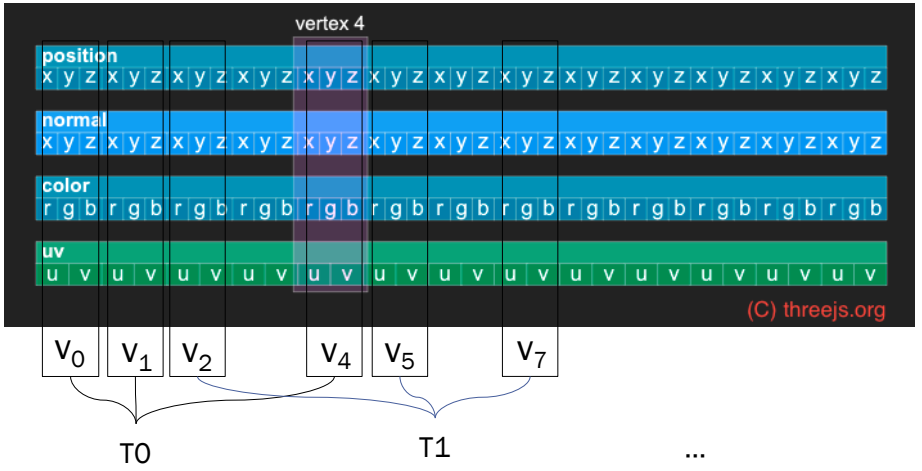


Float32arrays

THREE.BufferGeometry : explícita por índices

```
// Instanciar geometría propia
const geometry = new THREE.BufferGeometry();
// Construir los arrays
const position = new Float32Array( [
// x,y,z,x,y,z,...
] );
const normal = new Float32Array( [
// x,y,z,x,y,z,...
] );
const color = new Float32Array( [
// r,g,b,r,g,b,...
] );
const uv = new Float32Array( [
// u,v,u,v,,...
] );
// Construir los VBOs en la geometria
geometry.setAttribute( 'position', new THREE.BufferAttribute( position, 3 ) );
geometry.setAttribute( 'normal', new THREE.BufferAttribute( normal, 3 ) );
geometry.setAttribute( 'color', new THREE.BufferAttribute( color, 3 ) );
geometry.setAttribute( 'uv', new THREE.BufferAttribute( uv, 2 ) );
// Indices de triángulos
const indices = [ 0,1,4, 2,5,7, /*...*/]
geometry.setIndex(indices);
// Construir la forma
const material = new THREE.MeshBasicMaterial( );
const mesh = new THREE.Mesh( geometry, material );
```

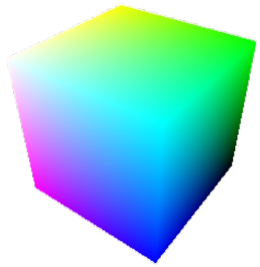
si color por vértice {vertexColor:true}



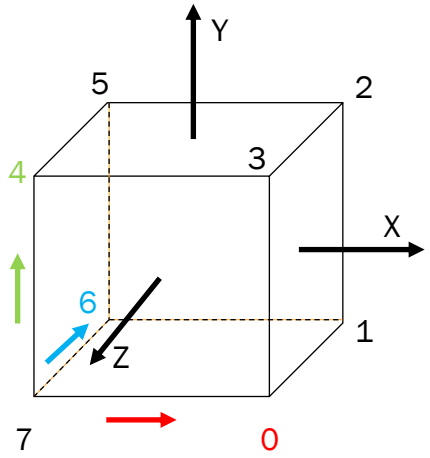
Float32arrays

Ejemplo: Cubo RGB con THREE

```
// Instancia el objeto BufferGeometry
const malla = new THREE.BufferGeometry();
// Construye la lista de coordenadas y colores por vertice
const coordenadas = [ // 8vert x3coor = 24float
  1,-1, 1,  1,-1,-1,  1, 1,-1,  1, 1, 1,
 -1, 1, 1, -1, 1,-1, -1,-1,-1, -1,-1, 1  ];
const colores = [ // 8 x 3 = 24
  1,0,0, 1,0,1, 1,1,1, 1,1,0,
  0,1,0, 0,1,1, 0,0,1, 0,0,0  ];
const indices = [ // 6caras x 2triangulos x3vertices = 36
  0,3,7, 7,3,4, 0,1,2,
  0,2,3, 4,3,2, 4,2,5,
  6,7,4, 6,4,5, 1,5,2,
  1,6,5, 7,6,1, 7,1,0     ];

// Geometria por VBO's
malla.setIndex( indices );
malla.setAttribute( 'position', new THREE.Float32BufferAttribute(coordenadas,3));
malla.setAttribute( 'color', new THREE.Float32BufferAttribute(colores,3));

// Configura un material
const material = new THREE.MeshBasicMaterial( { vertexColors: true } );

// Construye el objeto grafico
console.log(malla);   //-> Puedes consultar la estructura del objeto
cubo = new THREE.Mesh( malla, material );
```



```
/**
* CuboRGB.js
*   En sentido antihorario las caras son:
*   Delante:   7,0,3,4
*   Derecha:   0,1,2,3
*   Detras:    1,6,5,2
*   Izquierda: 6,7,4,5
*   Arriba:    3,2,5,4
+   Abajo:     0,7,6,1
* Donde se han numerado de 0..7 los vertices del cubo.
* Dado que cada vertice solo tiene un color el buffer de
* coordenadas y el de colores son de 8x3=24 floats.
* Los indices indican como formar los triangulos (6cx2tx3v=36)
*/
```

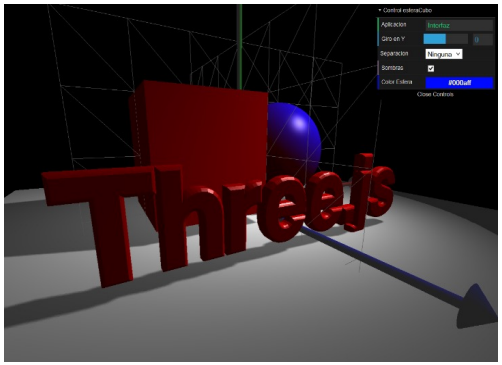


Cubo RGB con THREE: ejercicio

Se requiere asignar normales a cada vértice según la cara del cubo.
Ampliar el ejemplo para que soporte tres normales por vértice.

Texto en Three.js

- BufferGeometry -> ExtrudeGeometry -> TextGeometry(texto,atributos)
- Primero hay que cargar la fuente
- Atributos principales
  - font: fuente (shape) para el extruido
  - *size*: tamaño de la letra
  - *height*: profundidad del extruido
- La posición corresponde al extremo inferior izquierdo del texto

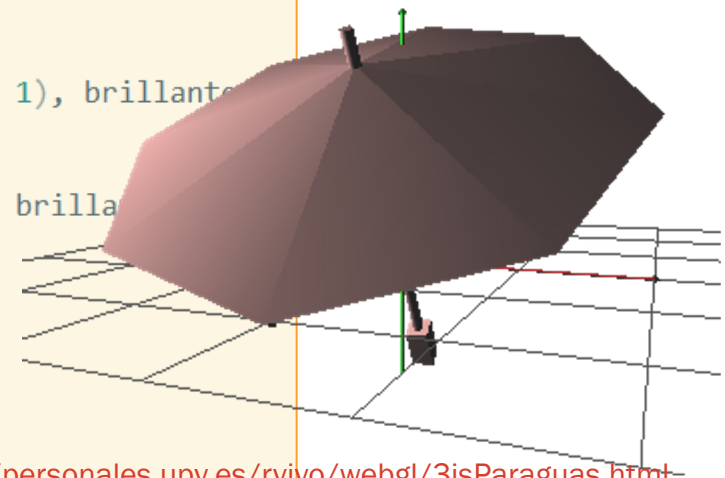



```
const loader = new FontLoader();
loader.load('fonts/droid_serif_regular.typeface.json',
  function (font)
  {
    const geometry = new TextGeometry( 'Hola', { size: 1, height: 0.1, font: font } );
    const malla = new THREE.Mesh(geometry, new THREE.MeshBasicMaterial());
    /*...*/
  }
);
```

Transformaciones en Three.js

- Todo Object3D tiene asociada una matriz de trasformación del modelo: modelMatrix
- La modelMatrix se construye
  - Traslación: position
  - Escalado: scale
  - Giros: rotation
- El orden de aplicación es fijo por defecto, **independientemente del orden en el código**
  - O' = MM*O = T*Rx*Ry*Rz*S*O
- Cuando un objeto es hijo de otro las trasformaciones se encadenan
  - O' = MMpadre*MM*O

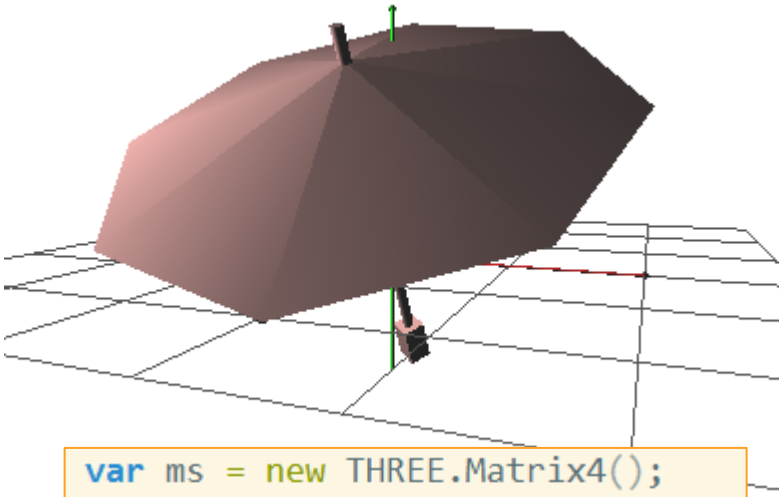
```
var tela = new THREE.Mesh( new THREE.CylinderGeometry( 0.0, 1.0, 1.0 ), mate );
tela.scale.set( 2, 0.5, 2 );
tela.position.y = 1.5;
var baston = new THREE.Mesh( new THREE.CylinderGeometry( 1, 1, 1), brillante
baston.position.y = 0.5;
baston.scale.set( 0.05, 3, 0.05 );
var mango = new THREE.Mesh( new THREE.CubeGeometry( 1, 1, 1 ), brilla
mango.scale.set( 0.2, 0.4, 0.2 );
mango.position.set( 0, -1, 0 );
mesh = new THREE.Object3D();
mesh.add( tela );
mesh.add( baston );
mesh.add( mango );
mesh.rotation.setX( Math.PI/6 );
```



http://personales.upv.es/rvivo/webgl/3jsParaguas.html

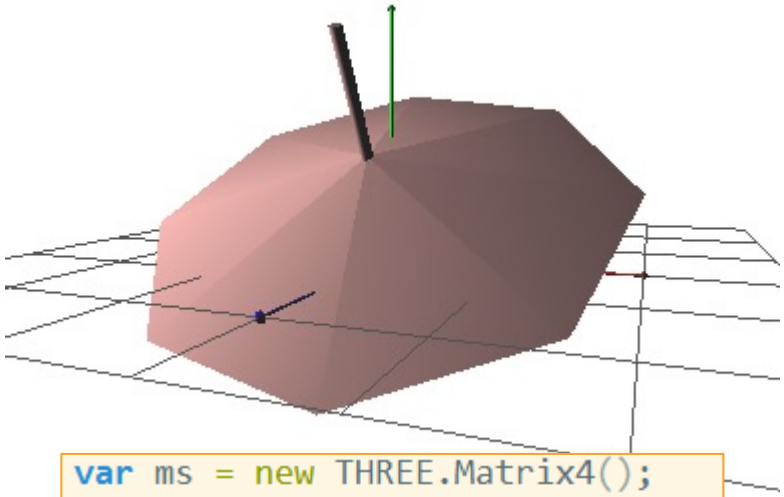
Trasformaciones en Three.js

- Es posible anular el orden fijo TRS y aplicar una modelMatrix propia
- Multitud de métodos de creación y operación para matrices
- La multiplicación (acumulación) es por la derecha



```
var ms = new THREE.Matrix4();
var mt = new THREE.Matrix4();
tela.matrixAutoUpdate = false;
mt.makeTranslation( 0, 1.5, 0 );
ms.makeScale( 2, 0.5, 2);
// modelmatrix = mt * ms
tela.matrix = mt.multiply( ms );
```

centroY = 0*msY+mtY = 1.5



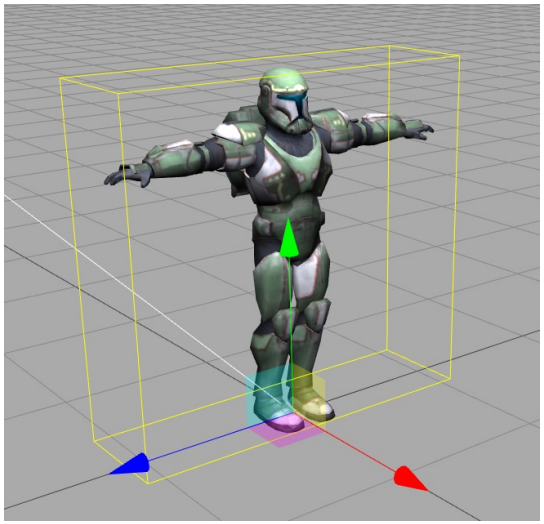
```
var ms = new THREE.Matrix4();
var mt = new THREE.Matrix4();
tela.matrixAutoUpdate = false;
mt.makeTranslation( 0, 1.5, 0 );
ms.makeScale( 2, 0.5, 2);
// modelmatrix = ms * mt
tela.matrix = ms.multiply( mt );
```

centroY = (0+mtY)*msY = 0.75

Importación de modelos

- Existen cargadores de modelos externos
  - json
    - ObjectLoader()
  - obj, gltf, collada, ...
    - Incluir la biblioteca
    - https://github.com/mrdoob/three.js/tree/master/examples/js/loaders

```
<script src = "js/OBJLoader.js"></script>
```



- Objetos cargadores
  - Instanciar el cargador
  - Cargar dando *url* y *callback* de finalización de carga
  - Manejador de carga

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

// load a resource
loader.load(
        // resource URL
        'models/skinned/UCS_config.json',
        // Function when resource is loaded
        function ( object ) {
                scene.add( object );
        }
);
```

Importación de modelos en Three.js

- Los cargadores llevan asociado un manejador con callbacks de
  - progreso
  - error
  - carga

```
LoadingManager( onLoad, onProgress, onError )

onLoad — The function that needs to be called when all loaders are done.
onProgress — The function that needs to be called when an item is complete.
onError — The function that needs to be called when an item is errors.
```

- Existe un manejador por defecto

```
OBJLoader( manager )

manager — The loadingManager for the loader to use. Default is THREE.DefaultLoadingManager.

Creates a new OBJLoader.
```

- Existen plugins en Blender para exportar JSON

Importación de modelos en Three.js

- Uso del editor
  - importar
  - exportar