# Objeto Casero Three.js

Badrán Ramírez Kevin
*u6000258@unimilitar.edu.co*
Profesor: Gabriel Ávila

***Resumen*—En el siguiente informe se explica el procedimiento a realizar en el momento en que se desea hacer un objeto "casero" en Three.js, explicándo también tramos del código utilizado y su resultado final**

***Palabras clave*—Escribir las palabras clave que se encuentran en el informe, por ejemplo IEEE, ecuaciones, figuras, tablas.**

## I. Introducción

En este ejercicio desarrollado, el objetivo es la creación de un objeto a partir de geometrías constructivas, complementando una geometría con otra, a partir de estructura basada en Three.Js, al mismo tiempo deseamos que la estructura, a pesar de ser una composición de varias, se pueda modificar como una sola estructura, y se pueda identificar con facilidad dónde se complementan cada una de las geometrías constructivas.

## II. COMPETENCIAS A DESARROLLAR

- Desarrollo y comprensión de los objetos a crear para plasmarlos en código.
- Comprensión del manejo de las librerías a incorporar.
- Aplicación de geometrías constructivas.

## III. Desarrollo de la práctica

Para empezar el ejercicio, inicialmente es necesario identificar qué se va a plasmar durante la realización del código, esto con el principal objetivo de la incorporación de las librerías para realizar una correcta depuración del ejercicio.

Librerías incorporadas:

```
<script src="js/three.js"></script>
<script src="js/controls/OrbitControls.js"></script>
<script src="js/csg.js"></script>
<script src="js/THREE.CSG.js"></script>
<script>
```

A continuación se crean las geometrías que se utilizarán, para más adelante transformarlas en geometrías constructivas, como se en la siguiente imágen:

Creación de geometrías:

```
var sphereGeometry = new THREE.SphereGeometry( .6, 32, 32 );
var sphereGeometry2 = new THREE.SphereGeometry( .3, 32, 32 );
var sphereGeometry3 = new THREE.SphereGeometry( .9, 32, 32 );
var cilindrogeometry = new THREE.CylinderGeometry( .5, .3, 1, 32 );
```

Transformación a CSG:

```
var sphere1CSG = THREE.CSG.fromMesh( sphere1 );
var sphereCSG = THREE.CSG.fromMesh( sphere );
var sphereCSG2 = THREE.CSG.fromMesh( sphere3 );
var torsoCSG = THREE.CSG.fromMesh( torso );
var piecitoLCSG = THREE.CSG.fromMesh( piecitoL );
var piecitoRCSG = THREE.CSG.fromMesh( piecitoR );
```

Seguidamente se realizó una curva para hacer el ojo del juguete que se plasma en la escena, para que después de que la curva se cree, se pueda realizar una extrusión de la curva y general el material que corresponde al ojo izquierdo del personaje, como se ve a continuación:

*Realización de la curva uniendo los puntos a un shape y asignando resolución de la curva:

```
var curve2D = [];

    curve2D[2] = new THREE.Vector2( .4, .4 );
    curve2D[1] = new THREE.Vector2( .2, .1 );
    curve2D[0] = new THREE.Vector2( .1, .2 );


var shape = new THREE.Shape();
shape.moveTo(0,0);
shape.splineThru(curve2D);


var resolution =5;
var points = shape.getPoints( resolution );
var geometry = new THREE.BufferGeometry().setFromPoints( points );
var curveObject = new THREE.Line( geometry, material2 );
```

*Realización configuraciones de la extrusión y asignación de material:

```
//EXTRUDE
var extrudeSettings = {
    steps: 1,
    amount:0.2,
    bevelEnabled: false,
};

var geometryExt = new THREE.ExtrudeGeometry( shape, extrudeSettings );
var mesh = new THREE.Mesh( geometryExt, material2 ) ;
```

Como se puede ver se crea sólo una curva y su extrusión correspondiente, es decir, falta un ojo por crear; por ende lo que se hizo fue clonar el ya existente, tanto su curva como su extrusión, rotarlo y trasladarlo hasta que quedara junto al otro ojo; esto se hizo de la siguiente manera:

```
mesh.translateX(-.4);
mesh.translateZ(.35);
mesh.rotateY(-.2);



var mesh2;
mesh2=mesh.clone();
mesh2.rotateY(90);
mesh2.translateZ(-.15);
```

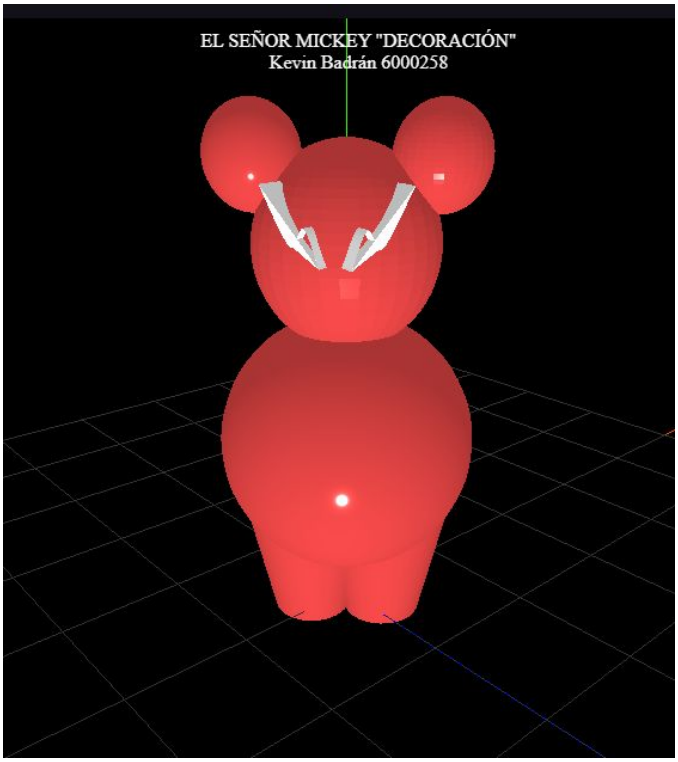
Así como se hizo con las geometrías iniciales, se transforman los ojos a CSG para que sea una unión del objeto principal y no sea un objeto aleatorio dentro del espacio.

```
var meshCSG = THREE.CSG.fromMesh( mesh );
var mesh2CSG = THREE.CSG.fromMesh( mesh2 );
```

Lo que queda por hacer con las geometrías que ya se encuentran transformadas en CSG es la asignación del material y agregarlas a la escena para que puedan ser visualizadas.

```
scene.add( spherel );
spherel.add( sphere3 );
spherel.add( torso );
spherel.add( piecitoL );
spherel.add( piecitoR );
spherel.add( mesh );
spherel.add( mesh2 );
spherel.translateY(2.6);
```

El objeto creado es el siguiente:




## IV. Conclusiones

El objeto inicial se creó sin dificultad al implementar cada una de las geometrías que componen al objeto principal, es comprensible, aunque es necesario mejorar la técnica de uso de estas funciones para realizar composiciones más complejas, pues en el ejercicio solo se realiza la unión de las piezas, mas no intersección ni sustracción, es necesario aclarar que no se incorporaron en el ejercicio debido a la poca complejidad del objeto; sin embargo, si estas operaciones (intersección y sustracción) se agregan al proyecto, este no tendrá problema en el momento de depurar, y mostrará el resultado de las geometrías.

La creación de los ojos por medio de extrusión deja claro que este método para la creación de objetos, es útil y se puede usar de diversas maneras, a pesar de que este se trabaja en 2D, es un recurso bastante viable.