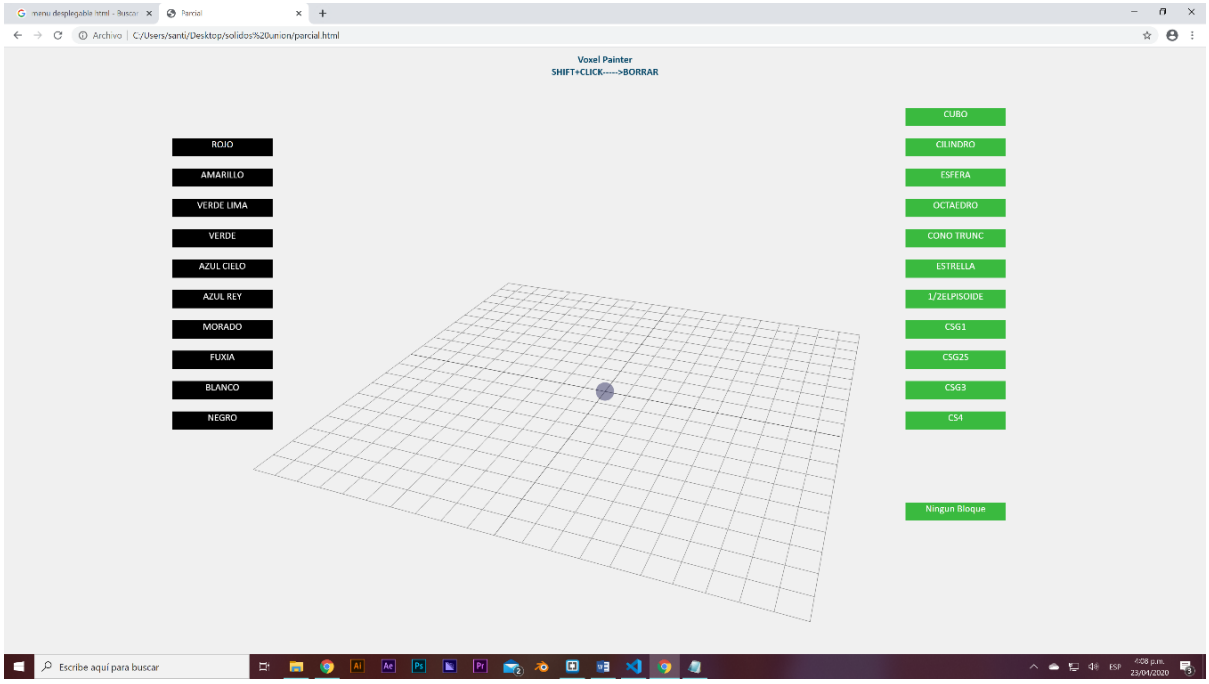
Computación Gráfica: Voxel Painter (Parcial).

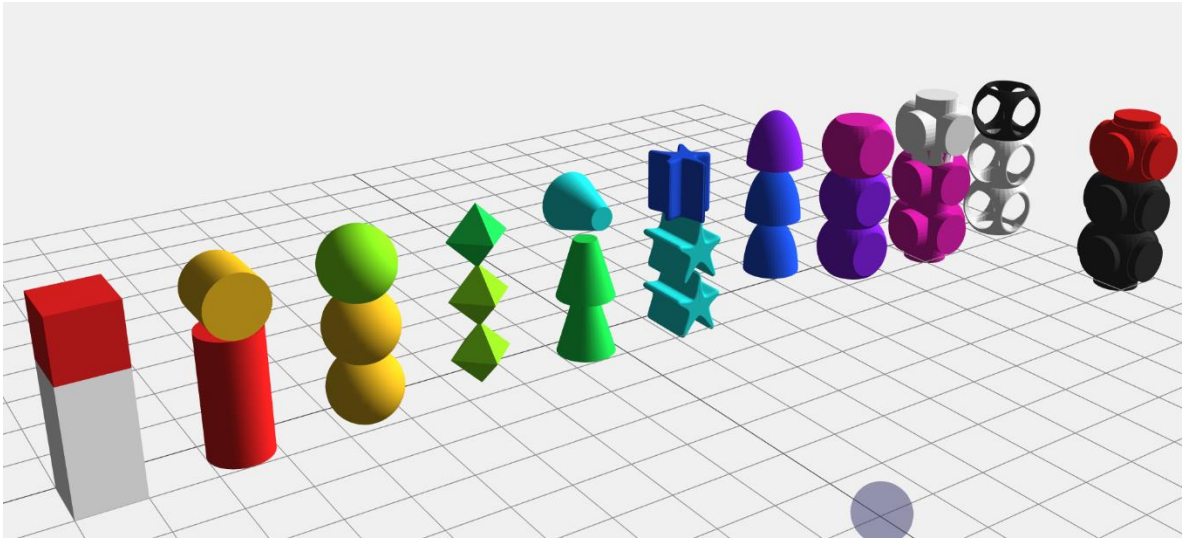
Santiago Forero Zapata
Cod:1202071
Ing. Multimedia

Interfaz

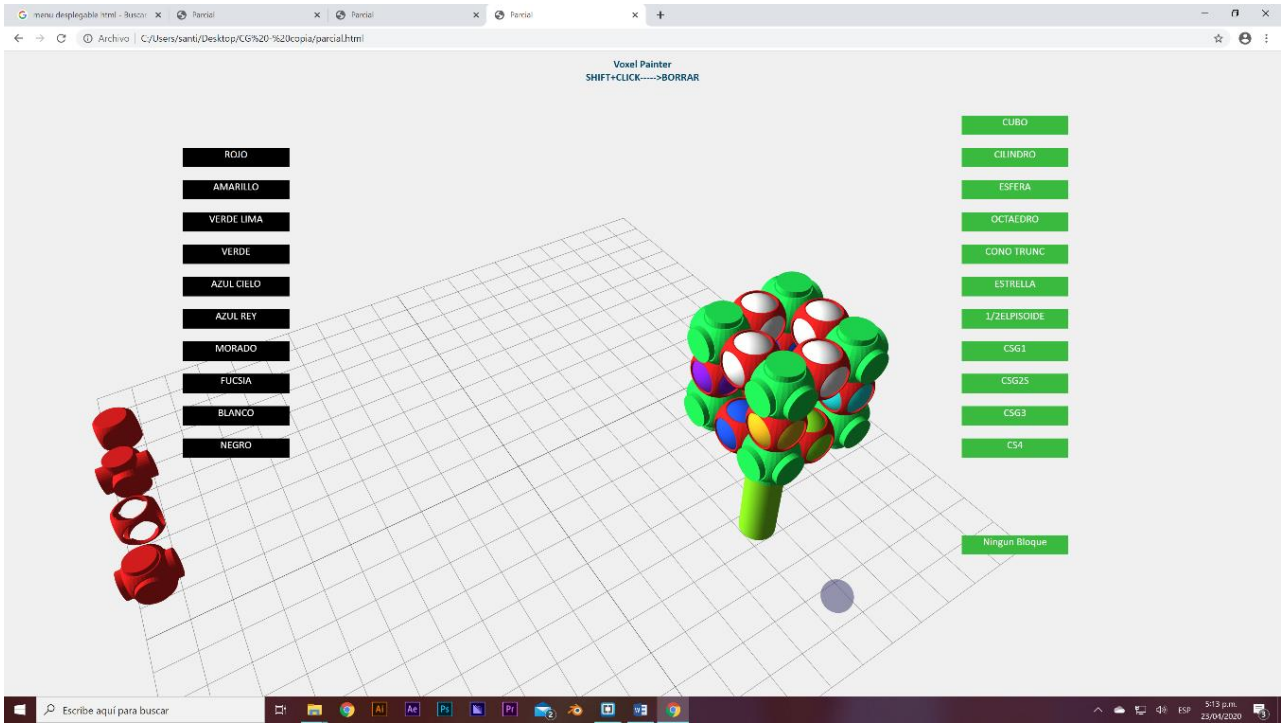
La interfaz cuenta con 10 botones en la parte izquierda del plano los cuales corresponden a los colores, en la parte derecha cuenta con 12 botones los primeros 11 correspondientes a bloques y el 12 final para ningún bloque, se debe indicar que para borrar se mantiene shift y se da click a la figura, para rotar se selecciona la figura se matiné tecla R y se da click en la posición deseada.

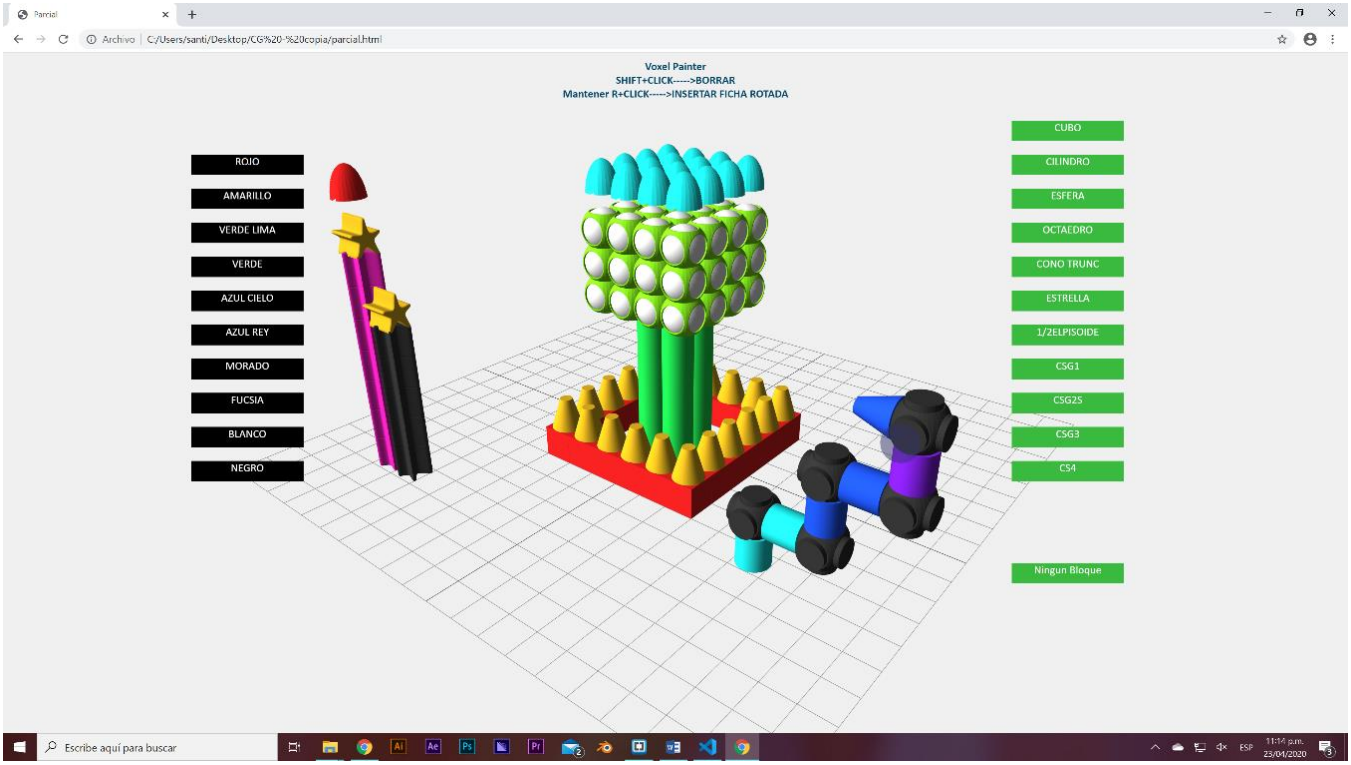


Estas son las posibles rotaciones:



En algunos casos se pueden insertar dos figuras en la misma posición como es en el caso de CS3 con la esfera:





Bloques

Para realizar cada uno de nuestros bloques usamos CSG, extrude, shapes, Fan and Strips, todo esto con three.js se elaboraron 6 bloques diferentes

- Bloques: CSG1, CSG2, CSG3, CSG4, 1/2ELIPSOIDE, ESTRELLA, GEOMETRIAS(Primitivas):

    - CSG1:

Se procede a crear una geometria se le asignan valores a cada una de estas,

```
var boxGeometry = new THREE.BoxGeometry( 50, 50, 50 );
var sphereGeometry = new THREE.SphereGeometry( 33, 32, 32 );
var cube = new THREE.Mesh( boxGeometry );
 var sphere = new THREE.Mesh( sphereGeometry );
```

procedemos a hacer la conversion a CSG

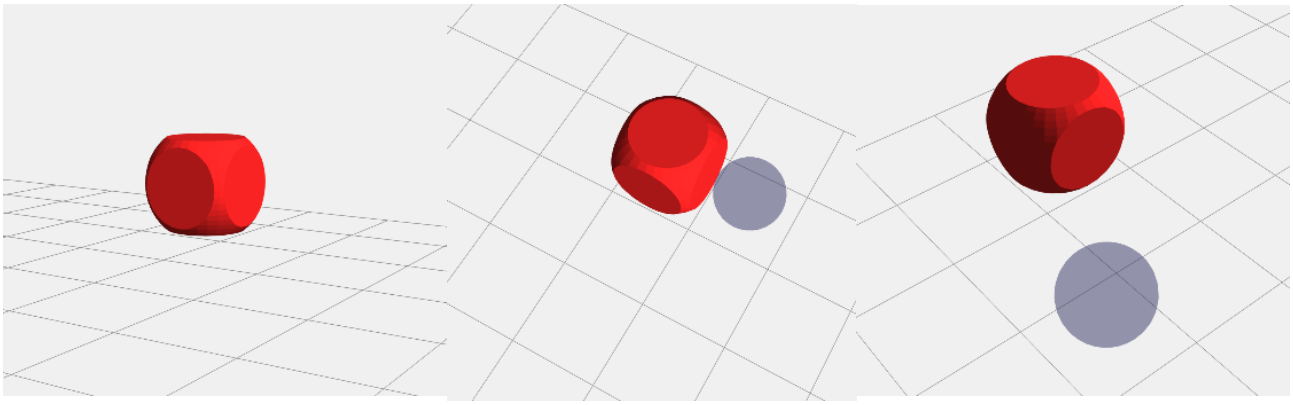
```
var boxCSG = THREE.CSG.fromMesh( cube );
            var sphereCSG = THREE.CSG.fromMesh( sphere );
```

,y luego gracias a la librería CSG se hace la interseccion de estas dos geometrias

```
var result1= boxCSG.intersect( sphereCSG );
```

como resultado de este podemos ver:

```
var C2T1 = THREE.CSG.toMesh( result1 );
```



    - CSG2:

Se procede a crear una geometria se le asignan valores a cada una de estas,

```
var geometry = new THREE.CylinderGeometry( 20, 20, 65, 32 )
var cylinder = new THREE.Mesh( geometry, material2 );
var cylinder2 = new THREE.Mesh( geometry, material2 );
var cylinder3 = new THREE.Mesh( geometry, material2 );
```

procedemos a hacer la conversion a CSG

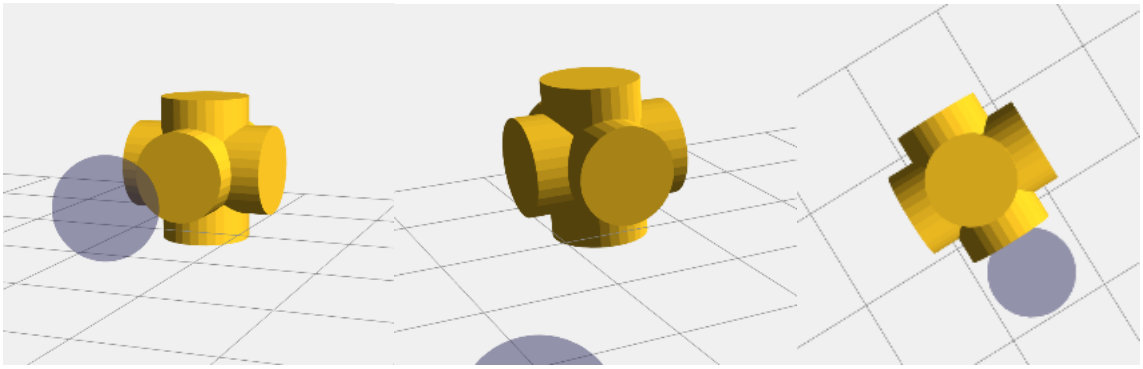
```
var cylinderCSG = THREE.CSG.fromMesh( cylinder );
var cylinder2CSG = THREE.CSG.fromMesh( cylinder2 );
var cylinder3CSG = THREE.CSG.fromMesh( cylinder3 );
```

,y luego gracias a la librería CSG se hace la UNION de estas dos geometrias

```
 var result2= cylinderCSG.union( cylinder2CSG ).union(cylinder3CSG);
```

como resultado de este podemos ver:

```
var C2T2 = THREE.CSG.toMesh( result2 );
```



- CSG3:

Se procede a usar las geometrias ya creadas y hacer su respectiva conversion a CSG

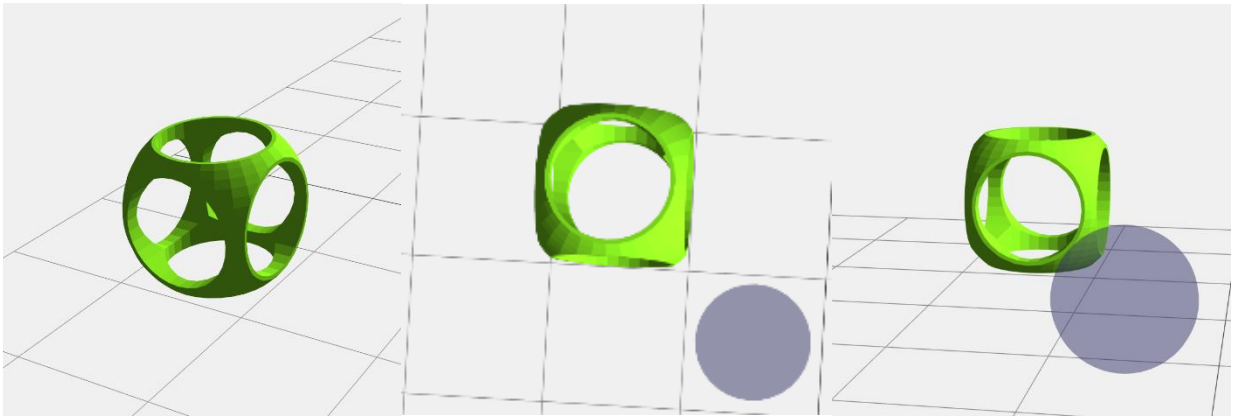
```
var UnionCSG = THREE.CSG.fromMesh( C2T2 );
var interseccionCSG = THREE.CSG.fromMesh( C2T1 );
```

,y luego gracias a la librería CSG se hace la SUBSTRACCION de estas dos geometrias

```
var result3= interseccionCSG.subtract( UnionCSG )
```

como resultado de este podemos ver:

```
var ELquees = THREE.CSG.toMesh( result3 );
```



- CSG4:

Se procede a usar las geometrias ya creadas y hacer su respectiva conversion a CSG

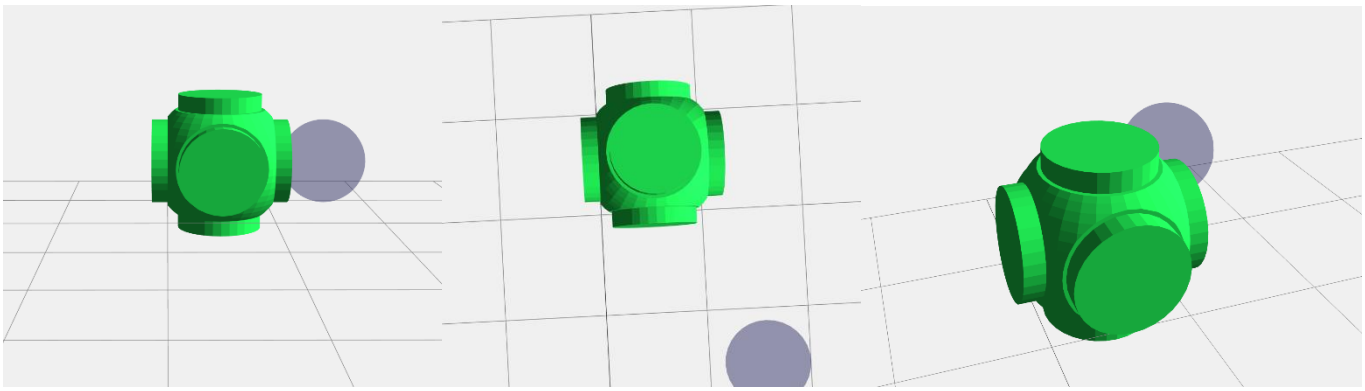
```
var UnionCSG = THREE.CSG.fromMesh( C2T2 );
var interseccionCSG = THREE.CSG.fromMesh( C2T1 );
```

,y luego gracias a la librería CSG se hace la UNION de estas dos geometrias

```
var result4=interseccionCSG.union(UnionCSG);
```

como resultado de este podemos ver:

```
var ELquees2 = THREE.CSG.toMesh( result4 );
```



- ½ ELIPSOIDE:

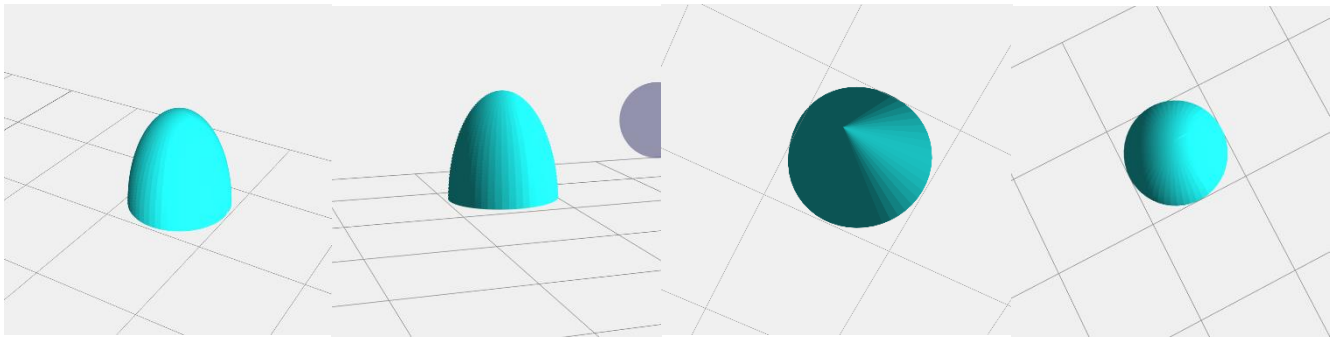
Se procede a crear una geometria la cual la llamo geoEsfera se crea un ciclo anidado esto con la intencio de recolectar todos los vertices(puntos) que componen la mitad de un elipsoide, estos valores se modificaron para satisfacer la idea:

```
var geoEsfera = new THREE.Geometry();
            var zTemp = 0;
            var res = 50.0;
            for( var i = 0; i < res; i++){
                for( var j = 0; j < res; j++){
                    var punto = new THREE.Vector3();
                    punto.x = 25 * Math.cos( ( j * 2 * Math.PI )/res) * Math.sin( ( i *
 Math.PI / 2 )/res)+125 ;
                    punto.z = 25 * Math.sin( ( j * 2 * Math.PI )/res) * Math.sin( ( i *
 Math.PI / 2 )/res)-425 ;
                    punto.y = 50 * Math.cos( ( i * Math.PI / 2 )/res) ;
                    geoEsfera.vertices.push( punto );
                }
            }
```

Al tener todos los vertices llamo a la clase TRIANGLE_FAN para que proceda a rellenar

```
fan = new TRIANGLE_FAN(geoEsfera);
fan.draw()
```

como resultado de este podemos ver:



- ESTRELLA:

Se creó un arreglo con los vértices correspondientes a una estrella de 5 puntas

```
 var curve2D = [];

curve2D[0] =new THREE.Vector2 (   0,  50 );
curve2D[1] =new THREE.Vector2 (  10,  10 ) ;
curve2D[2] =new THREE.Vector2 (  40,  10 ) ;
curve2D[3] = new THREE.Vector2 (  20, -10 );
curve2D[4] =new THREE.Vector2 (  30, -50 ) ;
curve2D[5] =new THREE.Vector2 (   0, -20 ) ;
curve2D[6] =new THREE.Vector2 ( -30, -50 ) ;
curve2D[7] =new THREE.Vector2 ( -20, -10 ) ;
curve2D[8] =new THREE.Vector2 ( -40,  10 ) ;
curve2D[9] =new THREE.Vector2 ( -10,  10 ) ;
curve2D[10] =new THREE.Vector2 (   0,  50 );
```

se crea un objeto shape, y se le asigna como argumento cada uno de los vértices del arreglo una variable resolution la cual se encarga de almacenar la cantidad de puntos entre vértices esto gracias a getPoints y se procede a crear la geometría

```
            var shape = new THREE.Shape();
            shape.splineThru(curve2D);
            var resolution = 2;
            var points = shape.getPoints( resolution );
        var geometry = new THREE.BufferGeometry().setFromPoints( points );
```

se crean parámetros para la extruccion

```
            var extrudeSettings = {
                steps: 5,
                amount: 90,
                bevelEnabled: true,
            };
```

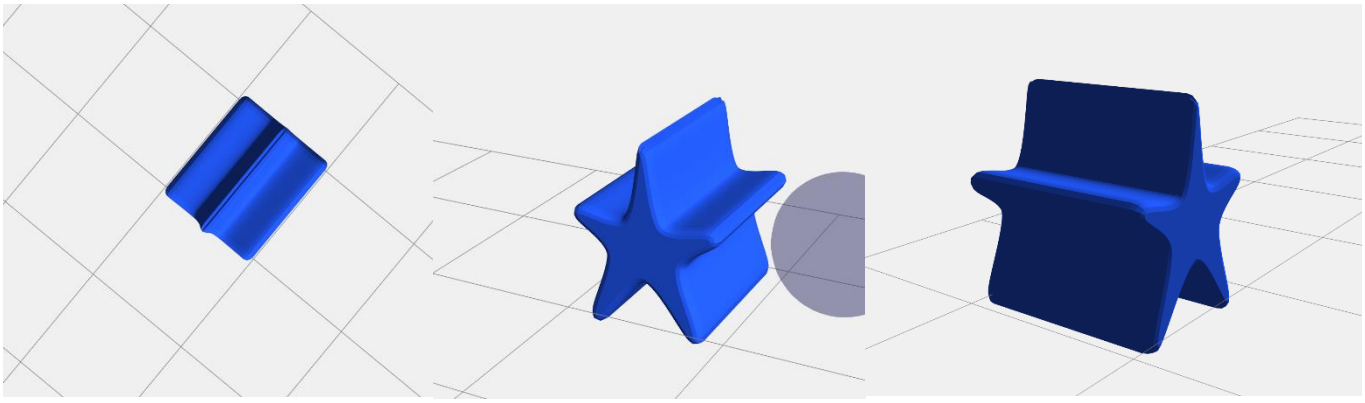
Se crea la geometría de extruccion c de los puntos obtenidos del shape y los parámetros de la extruccion

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

se crea la malla con la geometría de extruccion.

```
       var mesh = new THREE.Mesh( geometryExt, MaterialColor ) ;
```

como resultado de este podemos ver:



- GEOMETRIAS:

Se llamaron 5 geometrías y se le modifico sus respectivos valores:

```
cubeGeo = new THREE.BoxBufferGeometry( 50, 50, 50 );
ConeGeo=new THREE.CylinderBufferGeometry( 10, 25, 50, 32 );
CylinderGeo = new THREE.CylinderBufferGeometry( 25, 25, 50, 32 );
SphereGeo = new THREE.SphereBufferGeometry( 30, 32, 32 );
TorusGeo = new THREE.OctahedronBufferGeometry(27,0)
```

como resultado de este podemos ver: