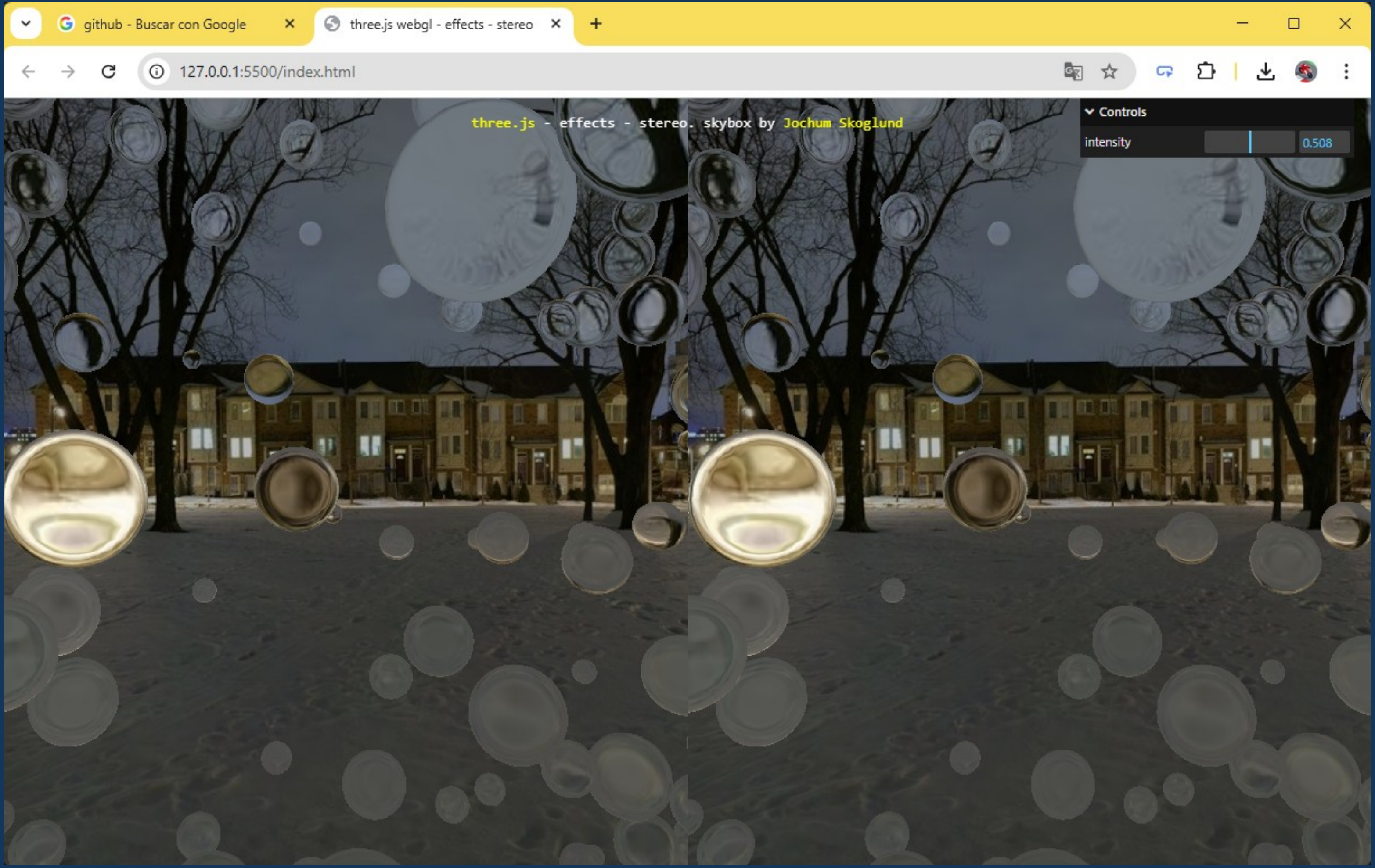
Programa Stereo

José Luis Carreño Arteaga
jcarreno53@yahoo.com.mx

Stereo

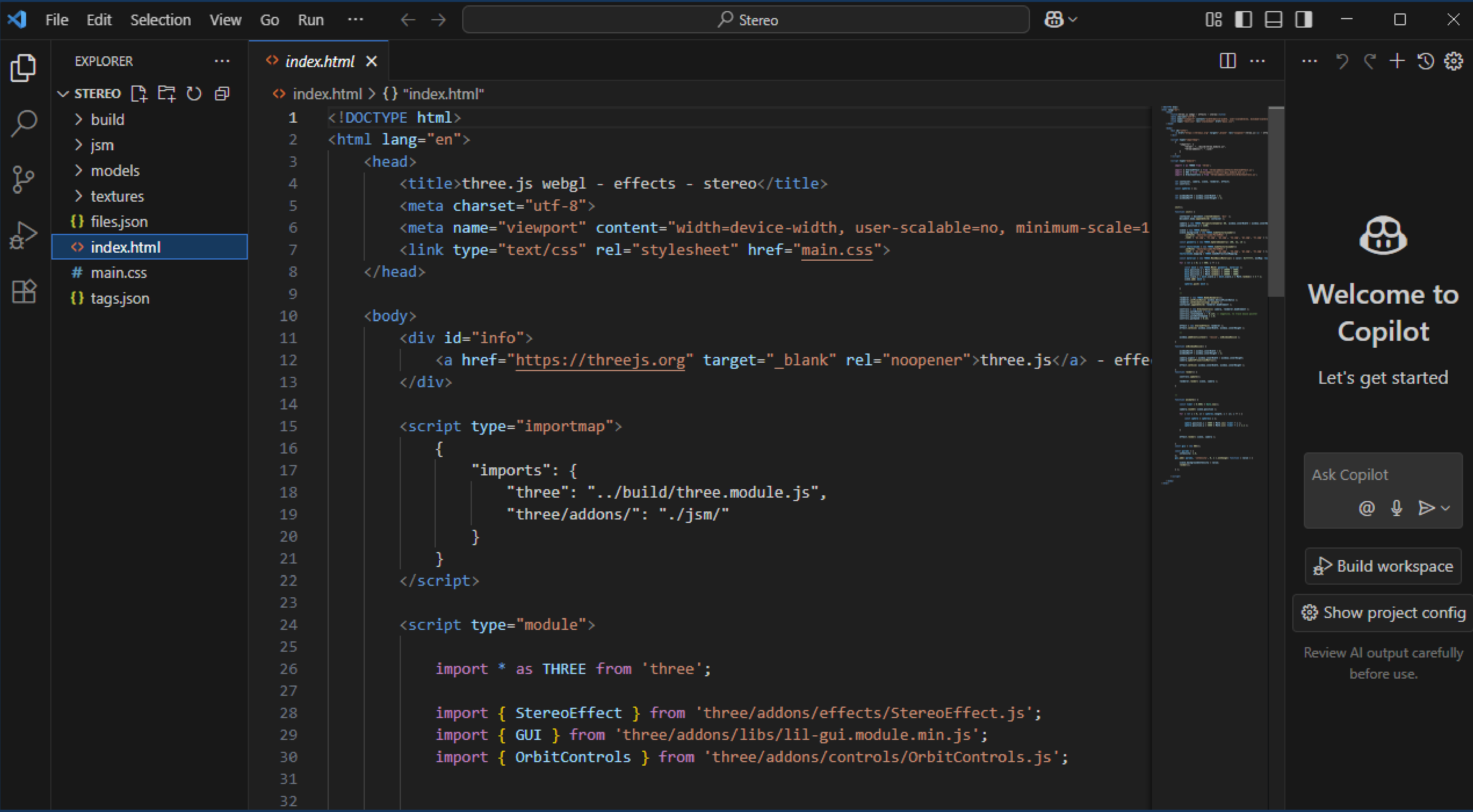
- Index.html



github - Buscar con Google
three.js webgl - effects - stereo
127.0.0.1:5500/index.html
three.js - effects - stereo. skybox by Jochum Skoglund
Controls
intensity
0.508

index.html

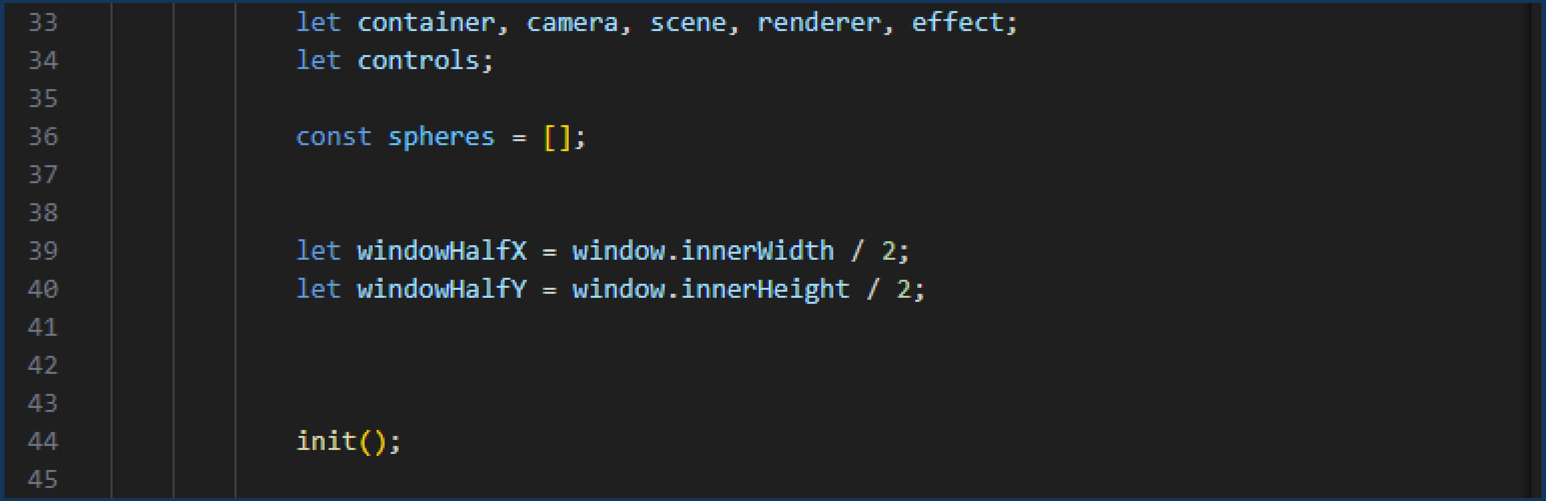
- Index.html



```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>three.js webgl - effects - stereo</title>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, user-scalable=no, minimum-scale=1
        <link type="text/css" rel="stylesheet" href="main.css">
    </head>

    <body>
        <div id="info">
            <a href="https://threejs.org" target="_blank" rel="noopener">three.js</a> - effe
        </div>

        <script type="importmap">
            {
                "imports": {
                    "three": "../build/three.module.js",
                    "three/addons/": "./jsm/"
                }
            }
        </script>

        <script type="module">

            import * as THREE from 'three';

            import { StereoEffect } from 'three/addons/effects/StereoEffect.js';
            import { GUI } from 'three/addons/libs/lil-gui.module.min.js';
            import { OrbitControls } from 'three/addons/controls/OrbitControls.js';


```

index.html

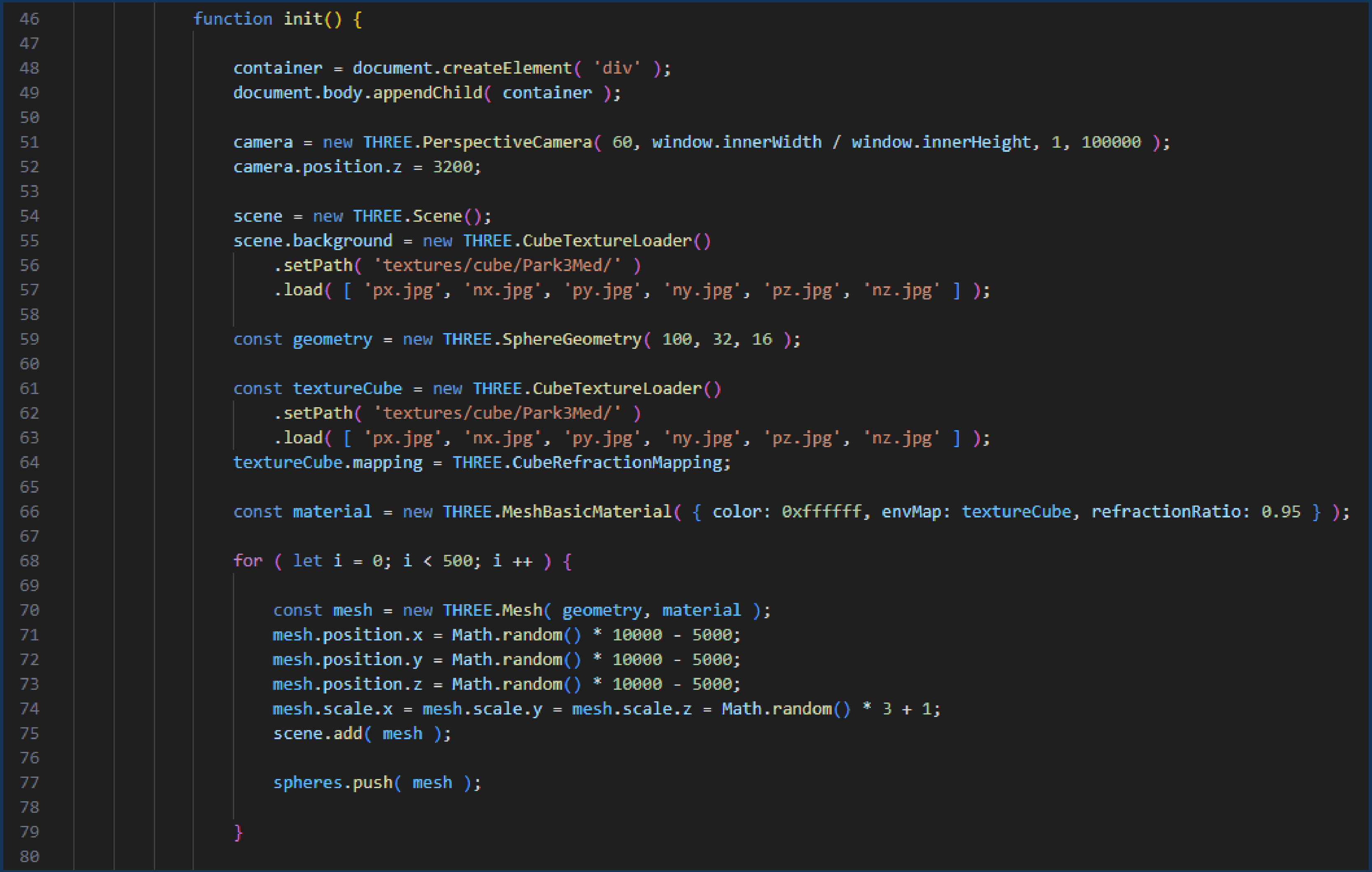
- Index.html



```
            let container, camera, scene, renderer, effect;
            let controls;

            const spheres = [];


            let windowHalfX = window.innerWidth / 2;
            let windowHalfY = window.innerHeight / 2;



            init();

```

index.html

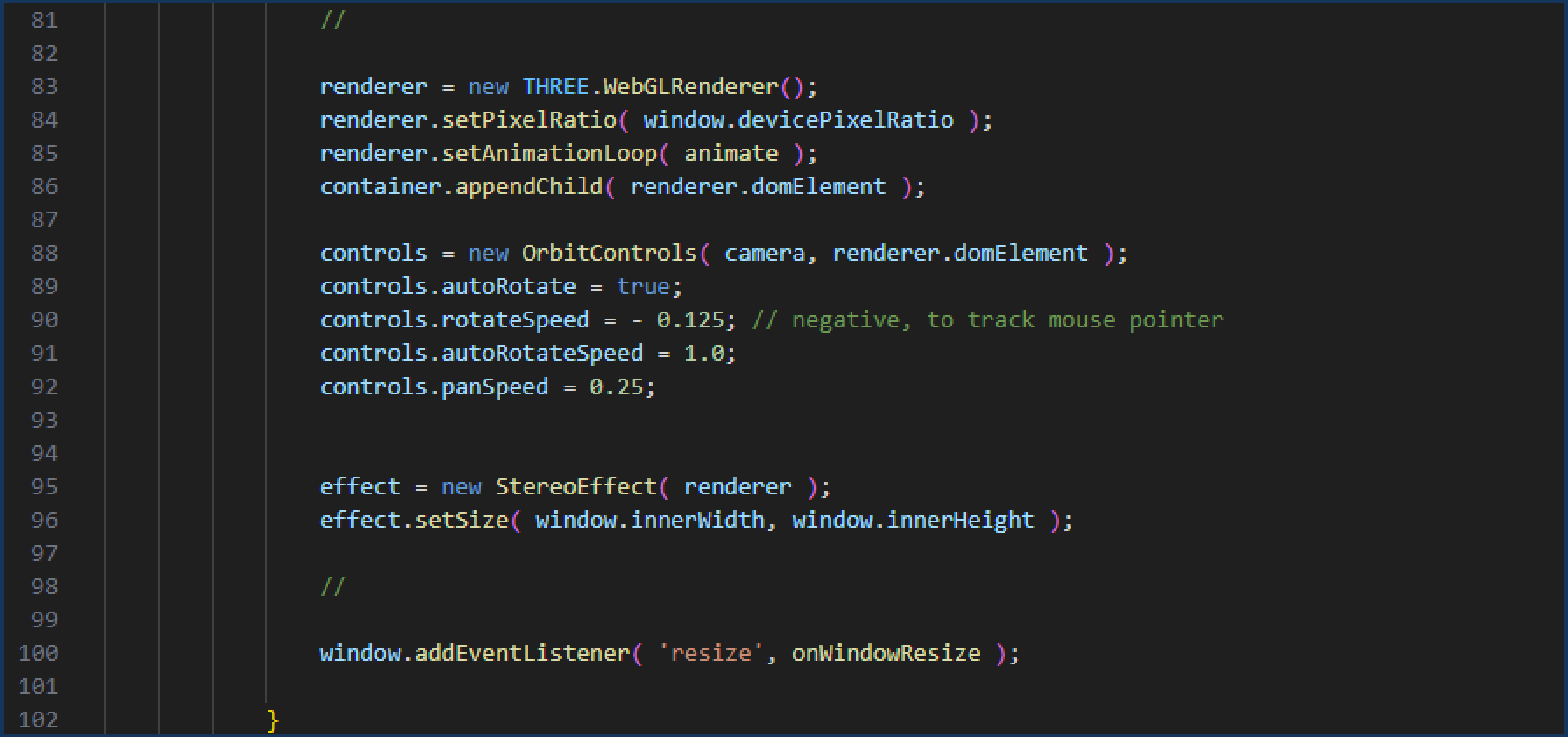
- Index.html



```
            function init() {

                container = document.createElement( 'div' );
                document.body.appendChild( container );

                camera = new THREE.PerspectiveCamera( 60, window.innerWidth / window.innerHeight, 1, 100000 );
                camera.position.z = 3200;

                scene = new THREE.Scene();
                scene.background = new THREE.CubeTextureLoader()
                    .setPath( 'textures/cube/Park3Med/' )
                    .load( [ 'px.jpg', 'nx.jpg', 'py.jpg', 'ny.jpg', 'pz.jpg', 'nz.jpg' ] );

                const geometry = new THREE.SphereGeometry( 100, 32, 16 );

                const textureCube = new THREE.CubeTextureLoader()
                    .setPath( 'textures/cube/Park3Med/' )
                    .load( [ 'px.jpg', 'nx.jpg', 'py.jpg', 'ny.jpg', 'pz.jpg', 'nz.jpg' ] );
                textureCube.mapping = THREE.CubeRefractionMapping;

                const material = new THREE.MeshBasicMaterial( { color: 0xffffff, envMap: textureCube, refractionRatio: 0.95 } );

                for ( let i = 0; i < 500; i ++ ) {

                    const mesh = new THREE.Mesh( geometry, material );
                    mesh.position.x = Math.random() * 10000 - 5000;
                    mesh.position.y = Math.random() * 10000 - 5000;
                    mesh.position.z = Math.random() * 10000 - 5000;
                    mesh.scale.x = mesh.scale.y = mesh.scale.z = Math.random() * 3 + 1;
                    scene.add( mesh );

                    spheres.push( mesh );

                }

```

index.html

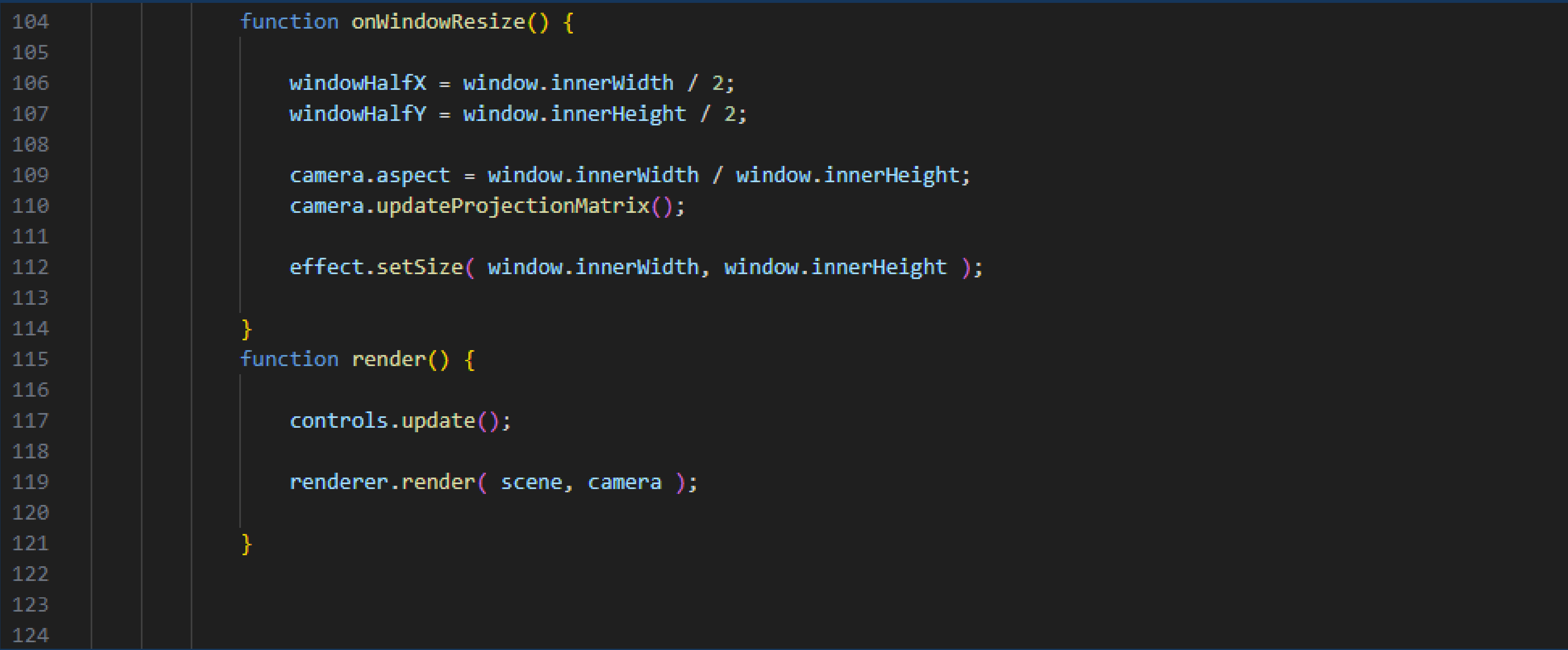
- Index.html



```
                    //

                    renderer = new THREE.WebGLRenderer();
                    renderer.setPixelRatio( window.devicePixelRatio );
                    renderer.setAnimationLoop( animate );
                    container.appendChild( renderer.domElement );

                    controls = new OrbitControls( camera, renderer.domElement );
                    controls.autoRotate = true;
                    controls.rotateSpeed = - 0.125; // negative, to track mouse pointer
                    controls.autoRotateSpeed = 1.0;
                    controls.panSpeed = 0.25;


                    effect = new StereoEffect( renderer );
                    effect.setSize( window.innerWidth, window.innerHeight );

                    //

                    window.addEventListener( 'resize', onWindowResize );

                }
```

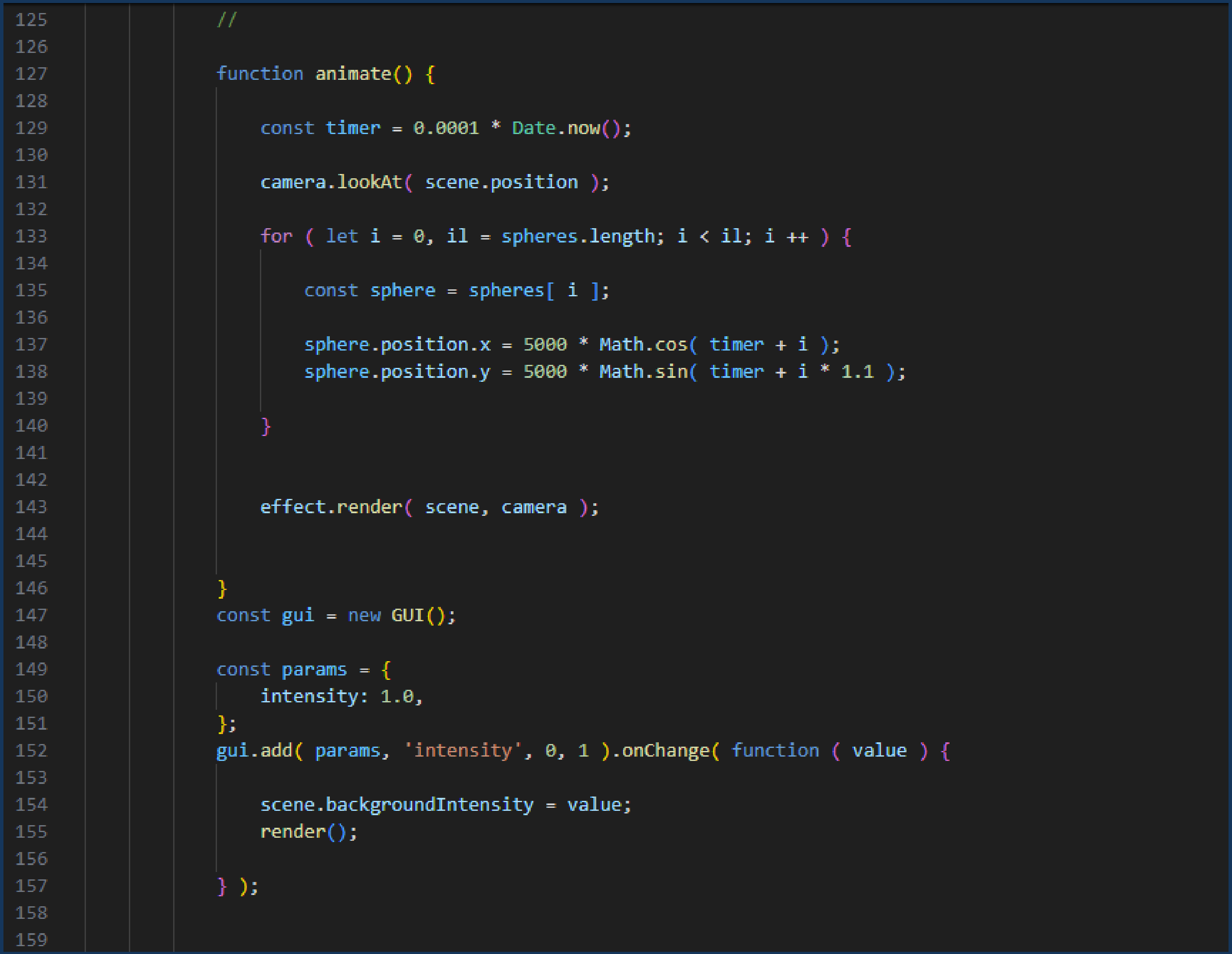
index.html

- Index.html



```
            function onWindowResize() {

                windowHalfX = window.innerWidth / 2;
                windowHalfY = window.innerHeight / 2;

                camera.aspect = window.innerWidth / window.innerHeight;
                camera.updateProjectionMatrix();

                effect.setSize( window.innerWidth, window.innerHeight );

            }
            function render() {

                controls.update();

                renderer.render( scene, camera );

            }
```

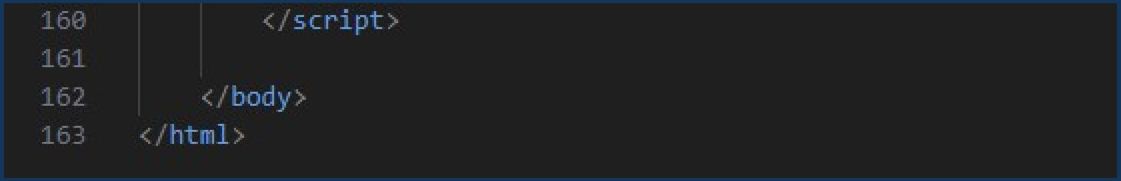
index.html



```
//

function animate() {

    const timer = 0.0001 * Date.now();

    camera.lookAt( scene.position );

    for ( let i = 0, il = spheres.length; i < il; i ++ ) {

        const sphere = spheres[ i ];

        sphere.position.x = 5000 * Math.cos( timer + i );
        sphere.position.y = 5000 * Math.sin( timer + i * 1.1 );

    }


    effect.render( scene, camera );


}
const gui = new GUI();

const params = {
    intensity: 1.0,
};
gui.add( params, 'intensity', 0, 1 ).onChange( function ( value ) {

    scene.backgroundIntensity = value;
    render();

} );
```

index.html

- Index.html



```
        </script>

    </body>
</html>
```

Referencias Bibliográficas