```
import './style.css'
import * as dat from 'dat.gui'
import * as THREE from 'three'
import { OrbitControls } from 'three/examples/jsm/controls/OrbitControls.js'
import { GLTFLoader } from 'three/examples/jsm/loaders/GLTFLoader.js'
import { DRACOLoader } from 'three/examples/jsm/loaders/DRACOLoader.js'

/**
 * Base
 */
// Debug
const gui = new dat.GUI({
    width: 400
})

// Canvas
const canvas = document.querySelector('canvas.webgl')

// Scene
const scene = new THREE.Scene()

/**
 * Loaders
 */
// Texture loader
const textureLoader = new THREE.TextureLoader()

// Draco loader
const dracoLoader = new DRACOLoader()
dracoLoader.setDecoderPath('draco/')

// GLTF loader
const gltfLoader = new GLTFLoader()
gltfLoader.setDRACOLoader(dracoLoader)

/**
 * Object
 */
const cube = new THREE.Mesh(
    new THREE.BoxGeometry(1, 1, 1),
    new THREE.MeshBasicMaterial()
)

scene.add(cube)

/**
 * Sizes
 */
const sizes = {
    width: window.innerWidth,
    height: window.innerHeight
}
```

```
window.addEventListener('resize', () =>
{
    // Update sizes
    sizes.width = window.innerWidth
    sizes.height = window.innerHeight

    // Update camera
    camera.aspect = sizes.width / sizes.height
    camera.updateProjectionMatrix()

    // Update renderer
    renderer.setSize(sizes.width, sizes.height)
    renderer.setPixelRatio(Math.min(window.devicePixelRatio, 2))
})

/**
 * Camera
 */
// Base camera
const camera = new THREE.PerspectiveCamera(45, sizes.width / sizes.height, 0.1,
100)
camera.position.x = 4
camera.position.y = 2
camera.position.z = 4
scene.add(camera)

// Controls
const controls = new OrbitControls(camera, canvas)
controls.enableDamping = true

/**
 * Renderer
 */
const renderer = new THREE.WebGLRenderer({
    canvas: canvas,
    antialias: true
})
renderer.setSize(sizes.width, sizes.height)
renderer.setPixelRatio(Math.min(window.devicePixelRatio, 2))

/**
 * Animate
 */
const clock = new THREE.Clock()

const tick = () =>
{
    const elapsedTime = clock.getElapsedTime()

    // Update controls
    controls.update()
```

```
    // Render
    renderer.render(scene, camera)

    // Call tick again on the next frame
    window.requestAnimationFrame(tick)
}

tick()
```