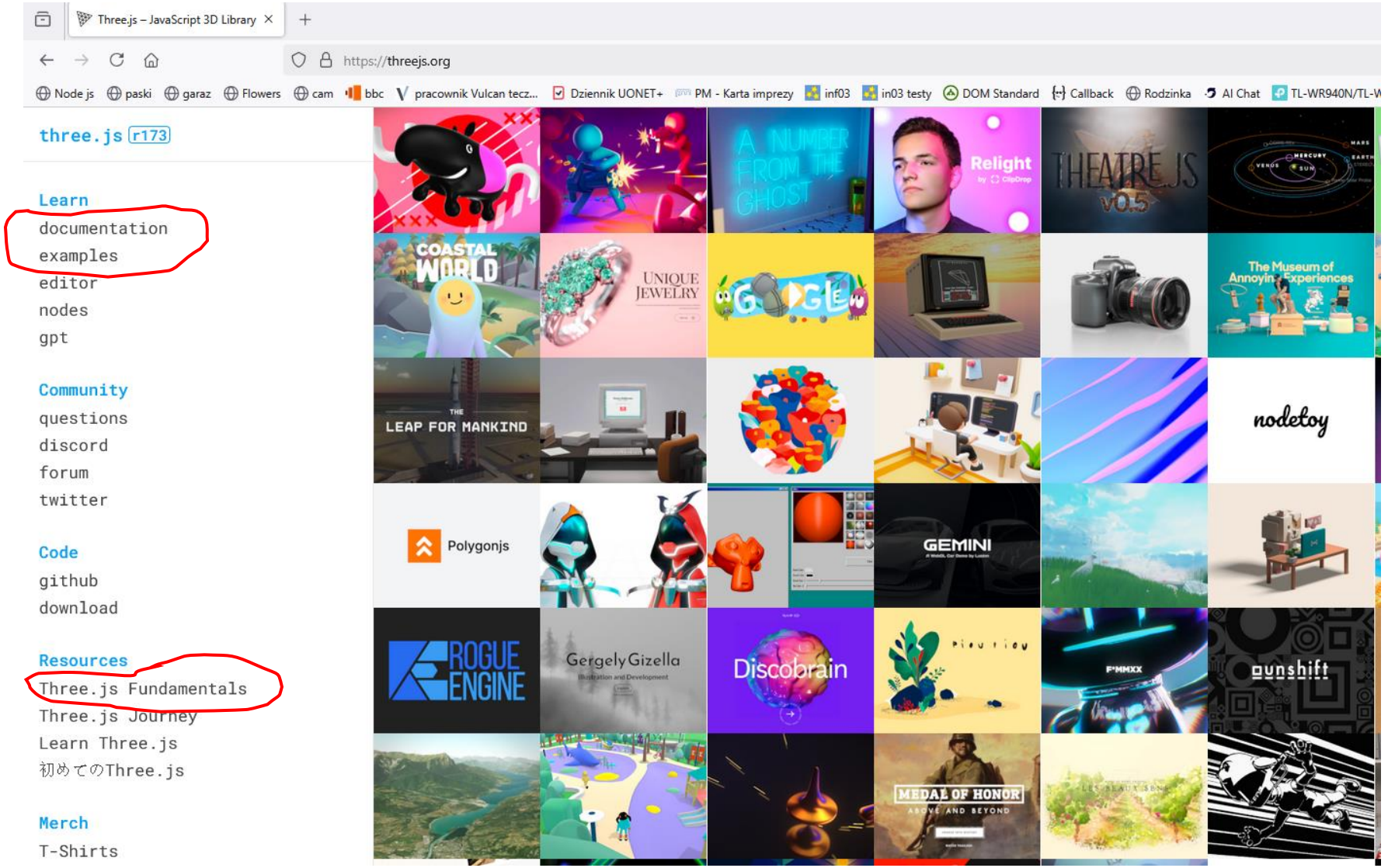
three.js

tutorial version 1.0

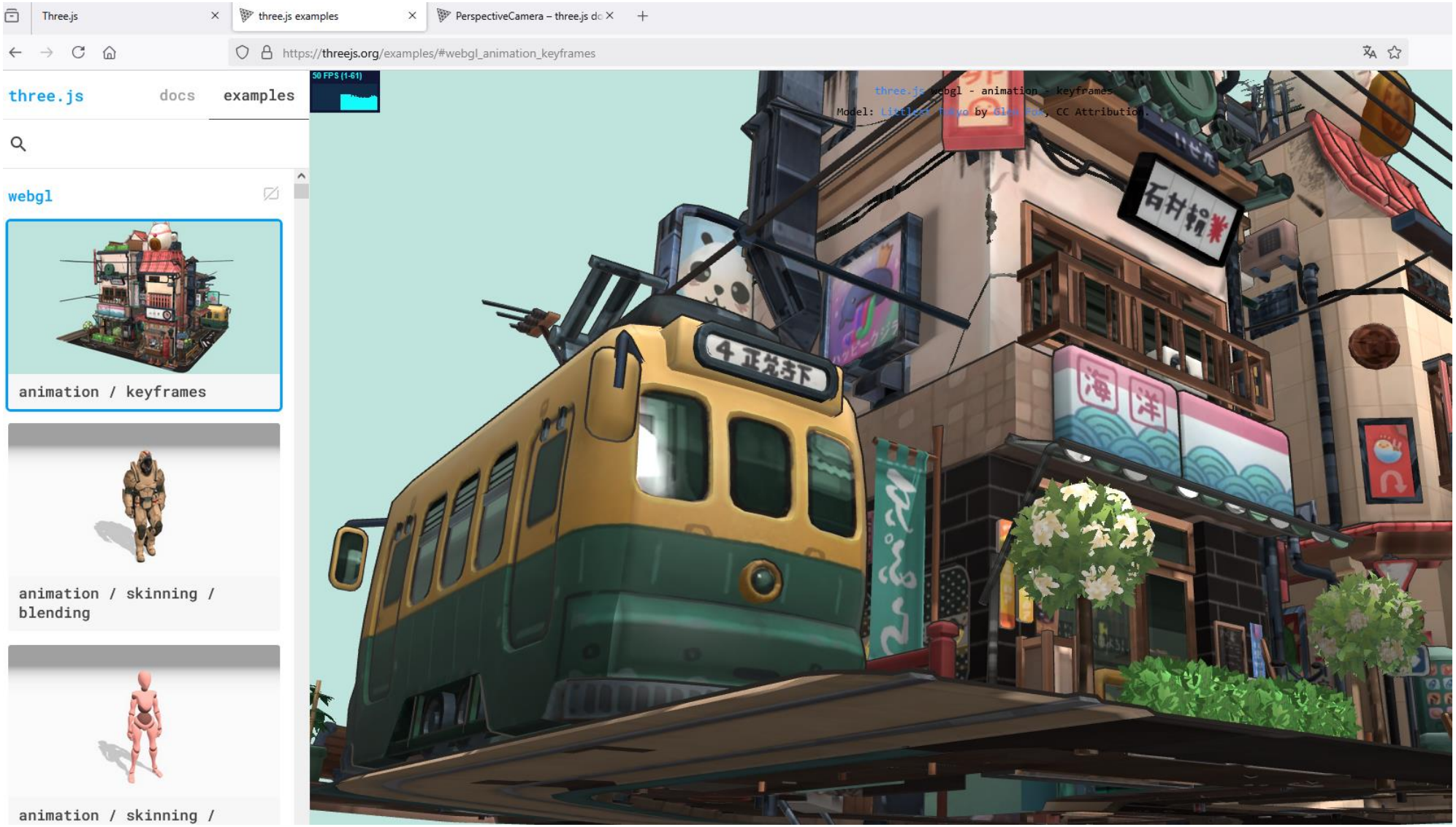
three.js

JavaScript 3D Library



https://threejs.org/

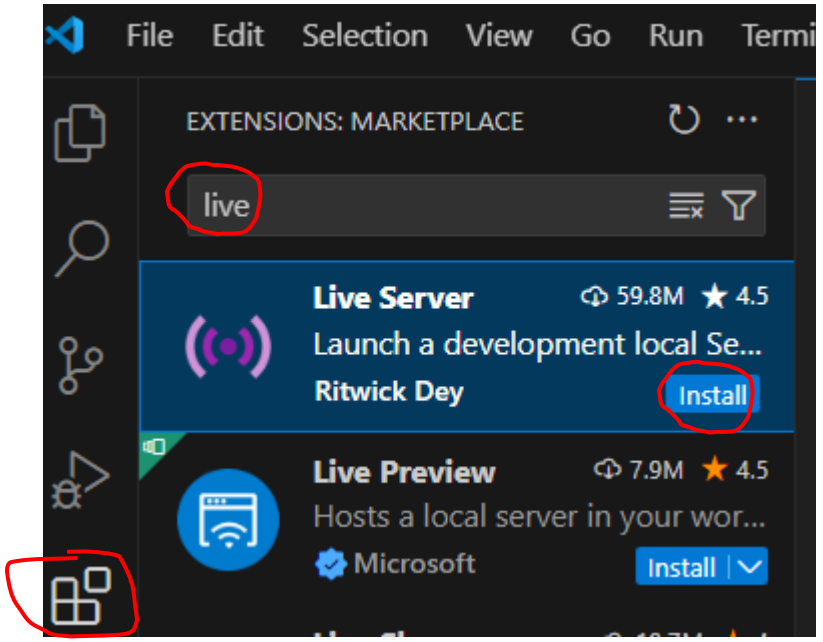
three.js examples



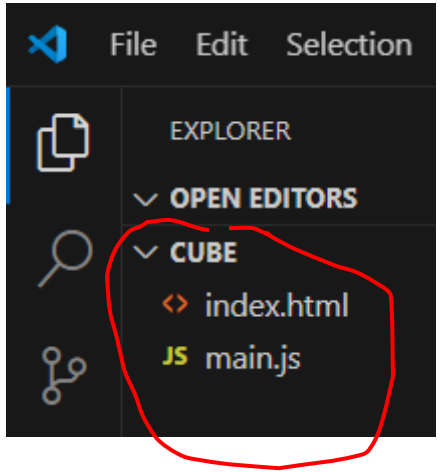
https://threejs.org/examples/#webgl_animation_keyframes

Visual Studio Code Live Server

instalacja Live Server w VSC



Cube



projekt Cube składa się z dwóch plików: index.html oraz main.js

index.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset='utf-8'>
    <title>Three.js</title>
    <script type="importmap">
        {
          "imports": {
            "three": "https://cdn.jsdelivr.net/npm/three@0.173.0/build/three.module.js",
            "three/addons/": "https://cdn.jsdelivr.net/npm/three@0.173.0/examples/jsm/"
          }
        }
      </script>
</head>
<body>
    <script type="module" src="/main.js"></script>
</body>
</html>

```

wersja biblioteki (najlepiej najnowsza)

https://threejs.org/docs/index.html#manual/en/introduction/Installation

Cube

File Edit Selection View Go Run Terminal Help — cube

main.js

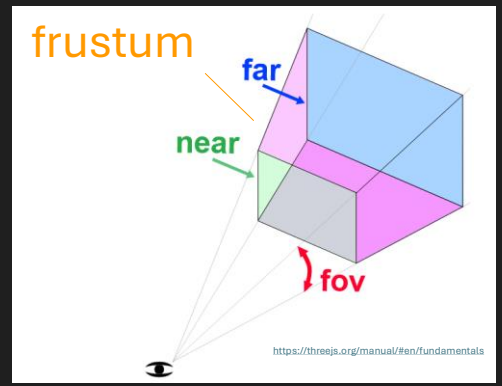
```
1  import * as THREE from 'three';
2
3  const scene = new THREE.Scene();
4  const camera = new THREE.PerspectiveCamera( 75, window.innerWidth / window.innerHeight, 0.1, 1000 );
5
6  const renderer = new THREE.WebGLRenderer();
7  renderer.setSize( window.innerWidth, window.innerHeight );
8  renderer.setAnimationLoop( animate );
9  document.body.appendChild( renderer.domElement );
10
11 const geometry = new THREE.BoxGeometry( 1, 1, 1 );
12 const material = new THREE.MeshBasicMaterial( { color: 0xff0000 } );
13 const cube = new THREE.Mesh( geometry, material );
14 scene.add( cube );
15
16 camera.position.z = 2;
17
18 function animate() {
19
20     cube.rotation.x += 0.01;
21     cube.rotation.y += 0.01;
22
23     renderer.render( scene, camera );
24
25 }
```

fov

aspect ratio

near

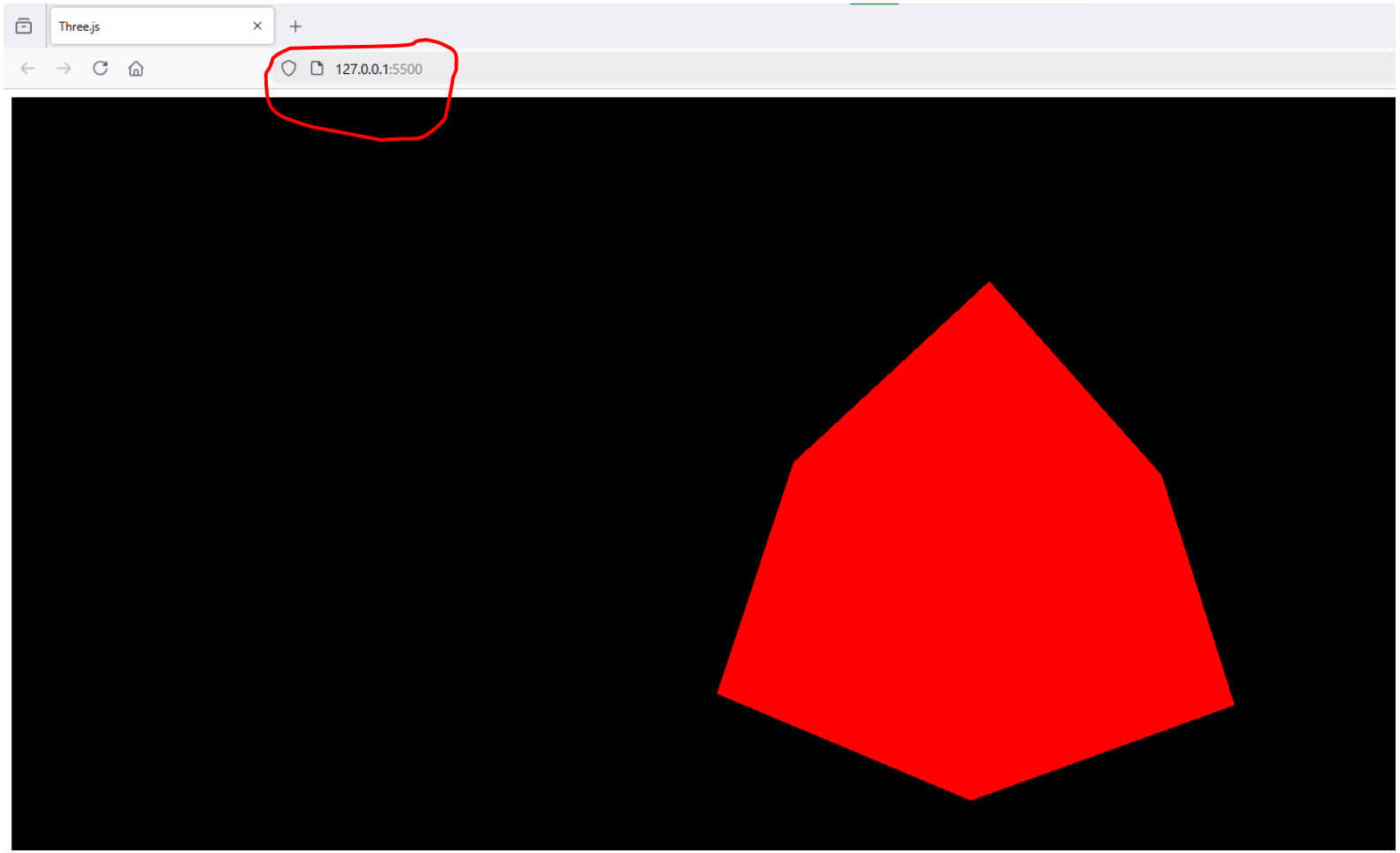
far



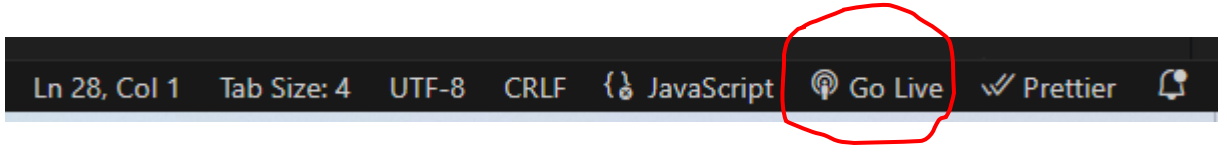
main.js

https://threejs.org/docs/index.html#manual/en/introduction/Creating-a-scene

Cube



Live Server



Cube + Capsule

three.js docs examples

en

LineCurve
LineCurve3
QuadraticBezierCurve
QuadraticBezierCurve3
SplineCurve

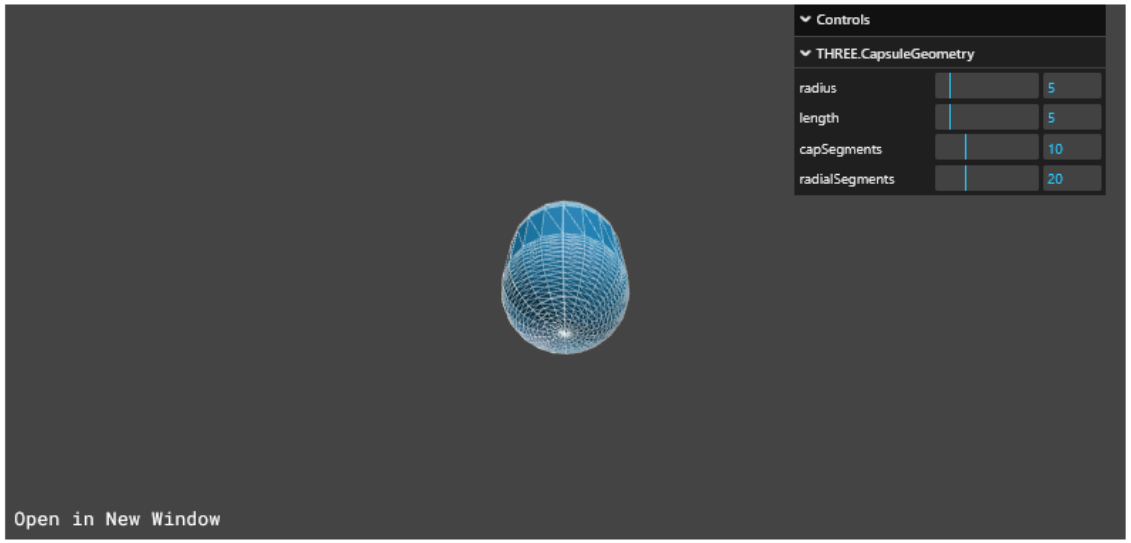
Geometries

BoxGeometry
CapsuleGeometry
CircleGeometry
ConeGeometry
CylinderGeometry
DodecahedronGeometry
EdgesGeometry
ExtrudeGeometry
IcosahedronGeometry
LatheGeometry
OctahedronGeometry
PlaneGeometry
PolyhedronGeometry
RingGeometry
ShapeGeometry
SphereGeometry
TetrahedronGeometry
TorusGeometry
TorusKnotGeometry

BufferGeometry → LatheGeometry →

CapsuleGeometry

CapsuleGeometry is a geometry class for a capsule with given radii and height. It is constructed using a lathe.



Code Example

```
const geometry = new THREE.CapsuleGeometry( 1, 1, 4, 8 );
const material = new THREE.MeshBasicMaterial( {color: 0x00ff00} );
const capsule = new THREE.Mesh( geometry, material ); scene.add( capsule );
```

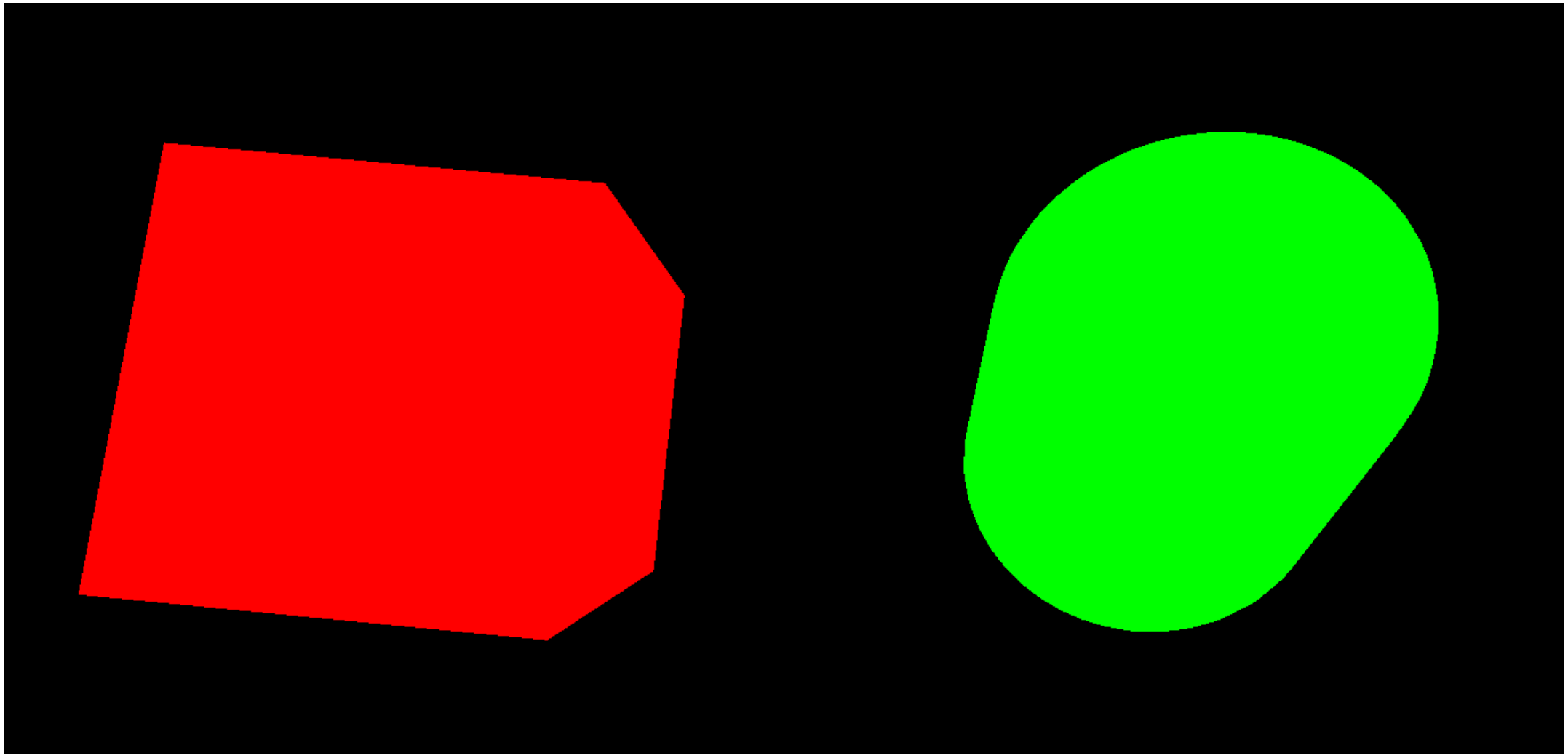
https://threejs.org/docs/index.html#api/en/geometries/CapsuleGeometry

Cube + Capsule

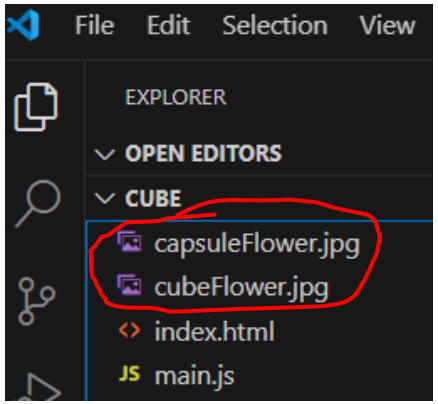
main.js

```
import * as THREE from 'three';

const scene = new THREE.Scene();
const camera = new THREE.PerspectiveCamera( 75, window.innerWidth / window.innerHeight, 0.1, 1000 );

const renderer = new THREE.WebGLRenderer();
renderer.setSize( window.innerWidth, window.innerHeight );
renderer.setAnimationLoop( animate );
document.body.appendChild( renderer.domElement );

const geometry = new THREE.BoxGeometry( 1, 1, 1 );
const material = new THREE.MeshBasicMaterial( { color: 0xff0000 } );
const cube = new THREE.Mesh( geometry, material );
scene.add( cube );

const capsuleGeometry = new THREE.CapsuleGeometry( 0.5, 0.5, 10, 40 );
const capsuleMaterial = new THREE.MeshBasicMaterial( {color: 0x00ff00} );
const capsule = new THREE.Mesh( capsuleGeometry, capsuleMaterial );
scene.add( capsule );

camera.position.z = 2;

// Pozycjonowanie figur obok siebie
cube.position.x = -1; // Przesuń sześcian w lewo
capsule.position.x = 1; // Przesuń kapsułę w prawo


function animate() {

    cube.rotation.x += 0.01;
    cube.rotation.y += 0.01;

    capsule.rotation.x += 0.09;
    capsule.rotation.y += 0.03;

    renderer.render( scene, camera );

}
```

Cube + Capsule



Cube + Capsule + texture



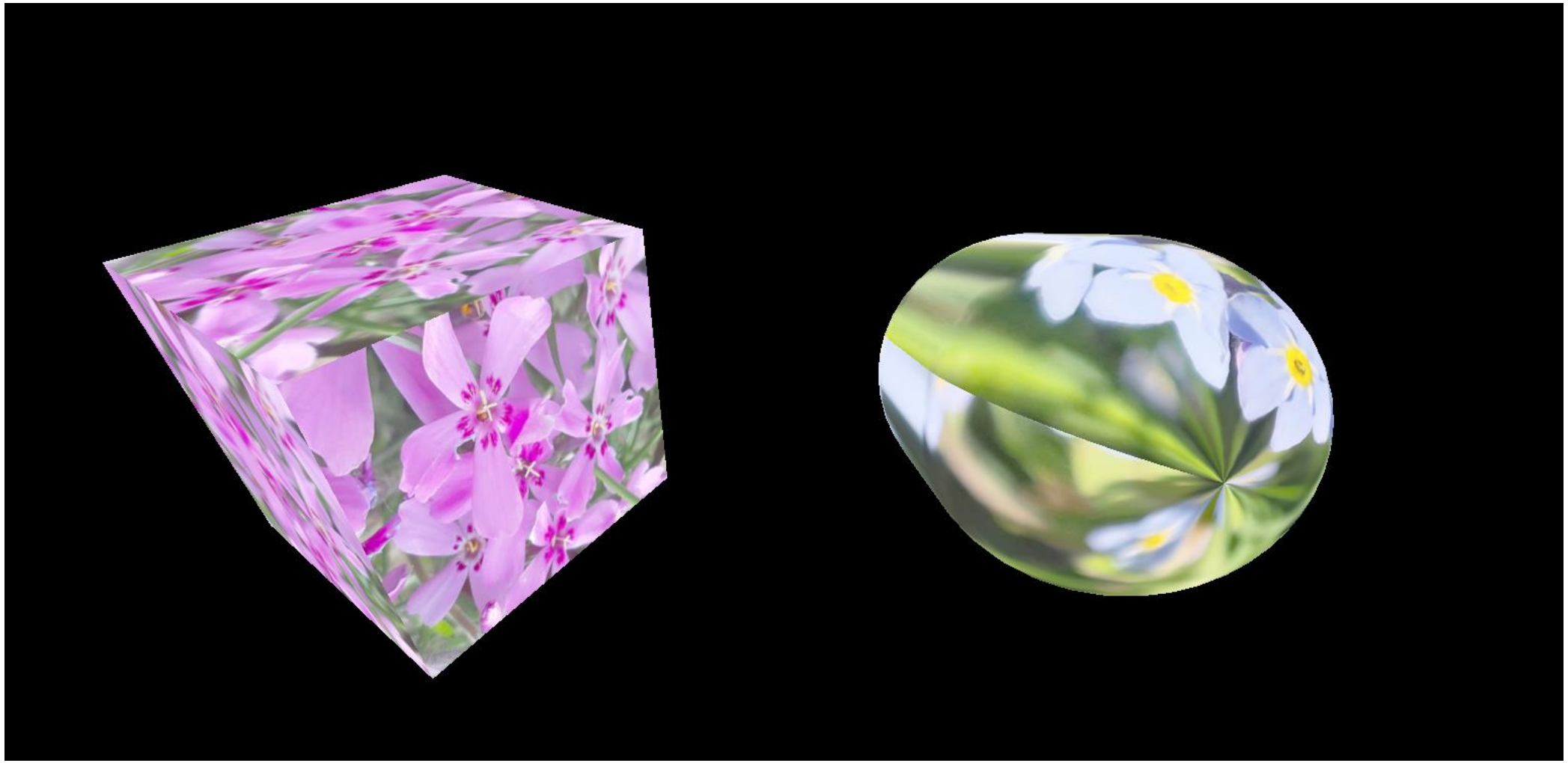
main2.js > ...

```
1   import * as THREE from 'three';
2
3   const scene = new THREE.Scene();
4   const camera = new THREE.PerspectiveCamera( 75, window.innerWidth / window.innerHeight, 0.1, 1000 );
5
6   const renderer = new THREE.WebGLRenderer();
7   renderer.setSize( window.innerWidth, window.innerHeight );
8   renderer.setAnimationLoop( animate );
9   document.body.appendChild( renderer.domElement );
10
11  const geometry = new THREE.BoxGeometry( 1, 1, 1 );
12  const texture = new THREE.TextureLoader().load('cubeFlower.jpg');
13  const material = new THREE.MeshBasicMaterial( { map: texture } );
14  const cube = new THREE.Mesh( geometry, material );
15  scene.add( cube );
16
17  const capsuleGeometry = new THREE.CapsuleGeometry( 0.5, 0.5, 10, 40 );
18  const capsuleTexture = new THREE.TextureLoader().load('capsuleFlower.jpg');
19  const capsuleMaterial = new THREE.MeshBasicMaterial( { map: capsuleTexture} );
20  const capsule = new THREE.Mesh( capsuleGeometry, capsuleMaterial );
21  scene.add( capsule );
22
23  camera.position.z = 2;
24
25  // Pozycjonowanie figur obok siebie
26  cube.position.x = -1; // Przesuń sześcian w lewo
27  capsule.position.x = 1; // Przesuń kapsułę w prawo
28
29
30  function animate() {
31
32      cube.rotation.x += 0.01;
33      cube.rotation.y += 0.01;
34
35      capsule.rotation.x += 0.03;
36      capsule.rotation.y += 0.03;
37
38      renderer.render( scene, camera );
39
40  }
```

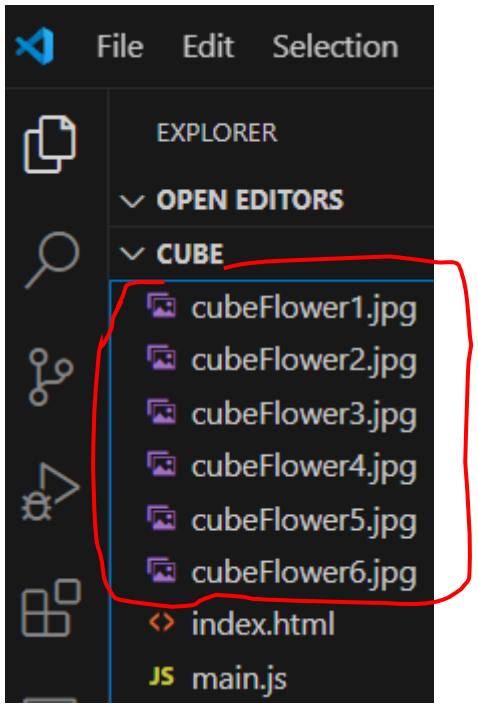
cubeFlower.jpg

capsuleFlower.jpg

Cube + Capsule + texture



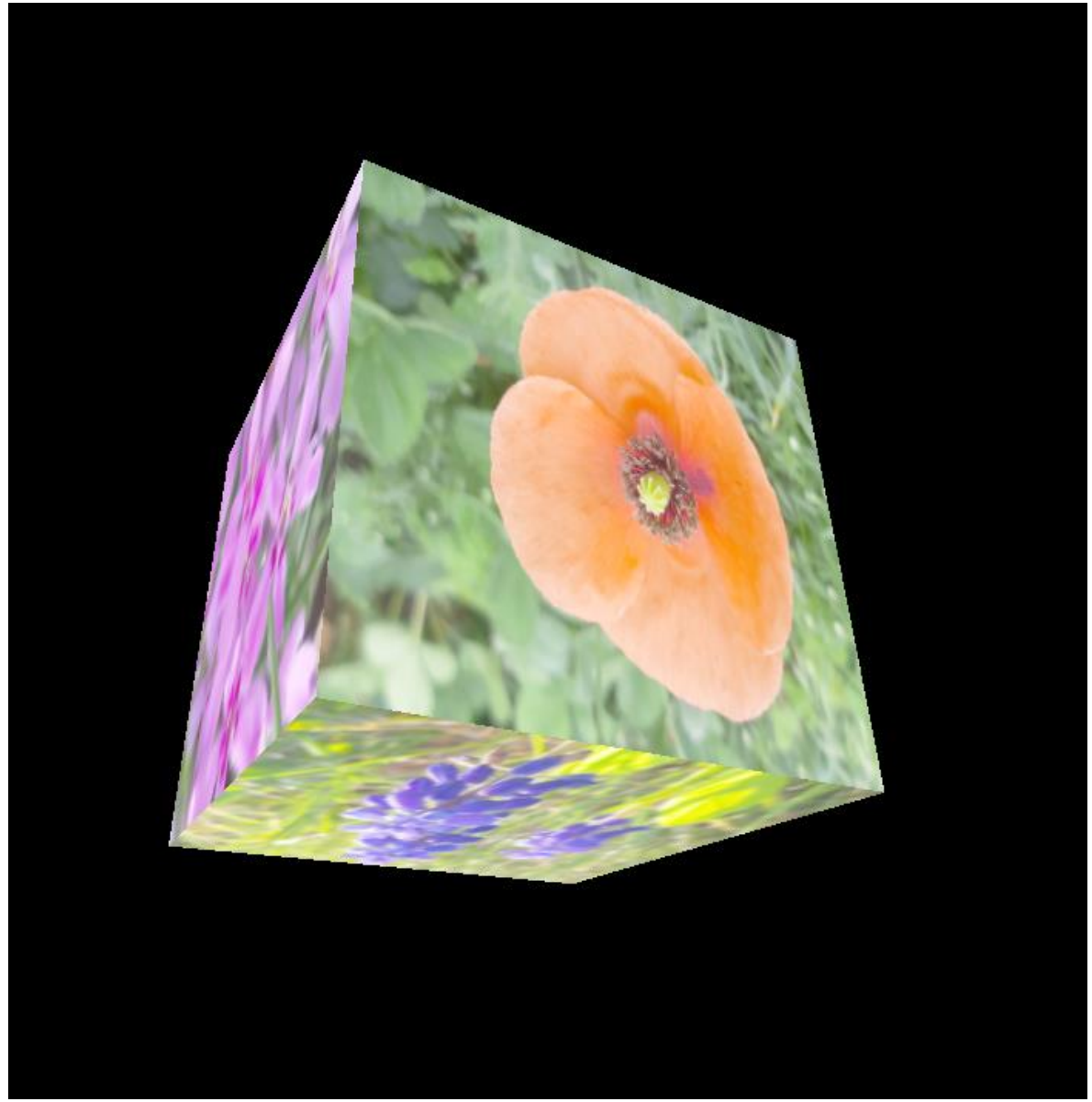
Cube + textures



main3.js > ...

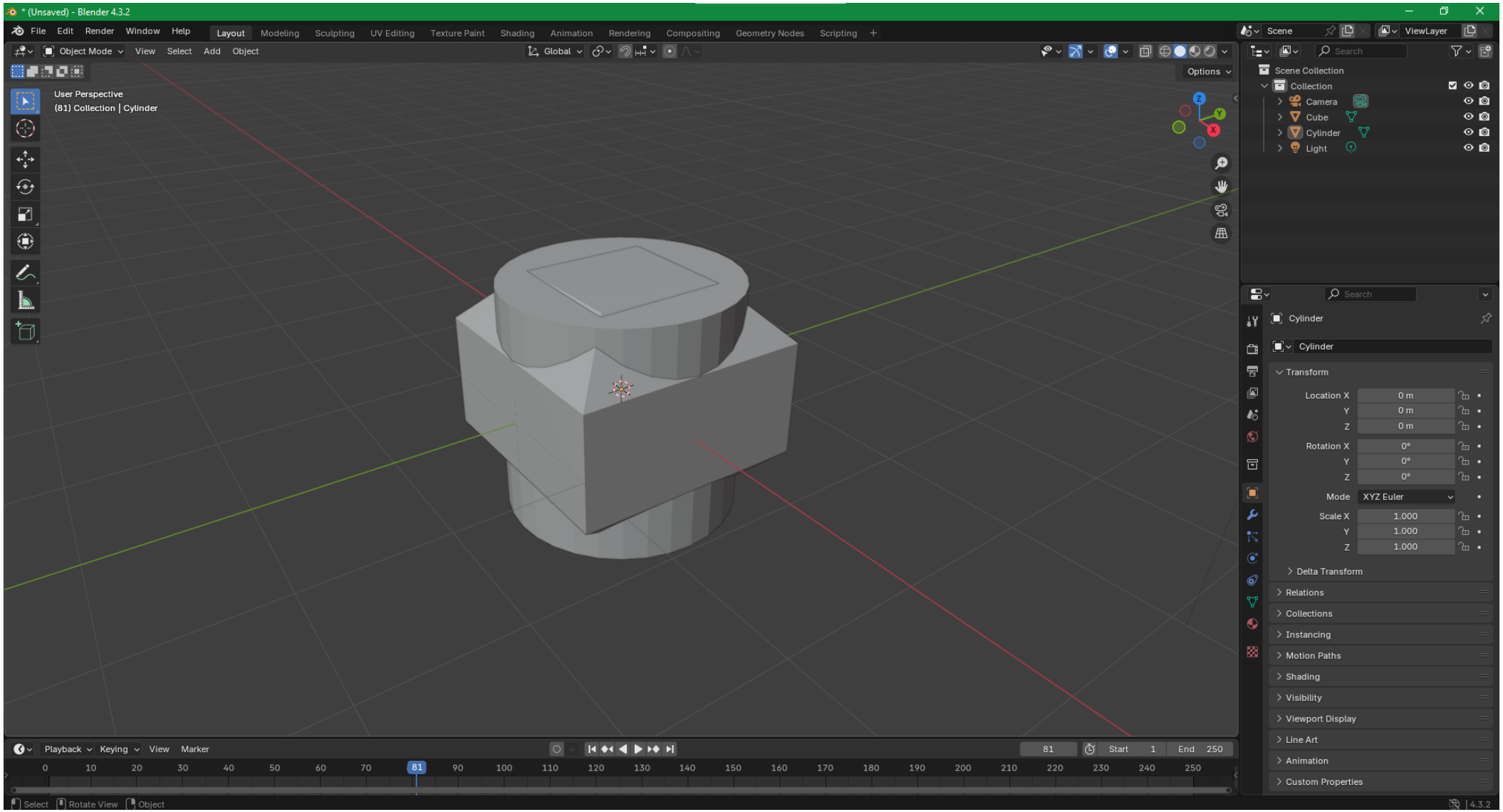
```
1   import * as THREE from 'three';
2
3   const scene = new THREE.Scene();
4   const camera = new THREE.PerspectiveCamera( 75, window.innerWidth / window.innerHeight, 0.1, 1000 );
5
6   const renderer = new THREE.WebGLRenderer();
7   renderer.setSize( window.innerWidth, window.innerHeight );
8   renderer.setAnimationLoop( animate );
9   document.body.appendChild( renderer.domElement );
10
11  // tekstury dla każdej ściany w cube
12  const texturePaths = [
13      './cubeFlower1.jpg',  // Przód
14      './cubeFlower2.jpg',  // Tył
15      './cubeFlower3.jpg',  // Góra
16      './cubeFlower4.jpg',  // Dół
17      './cubeFlower5.jpg',  // Lewo
18      './cubeFlower6.jpg',  // Prawo
19  ];
20
21  // tworzy tablicę tekstur
22  const textures = texturePaths.map(path => new THREE.TextureLoader().load(path));
23  // tworzy tablicę materiałów
24  const materials = textures.map(texture => new THREE.MeshBasicMaterial({ map: texture }));
25
26  const geometry = new THREE.BoxGeometry( 1, 1, 1 );
27  const cube = new THREE.Mesh( geometry, materials );
28  scene.add( cube );
29
30  camera.position.z = 2;
31
32  function animate() {
33
34      cube.rotation.x += 0.01;
35      cube.rotation.y += 0.01;
36
37      renderer.render( scene, camera );
38
39  }
```

Cube + textures

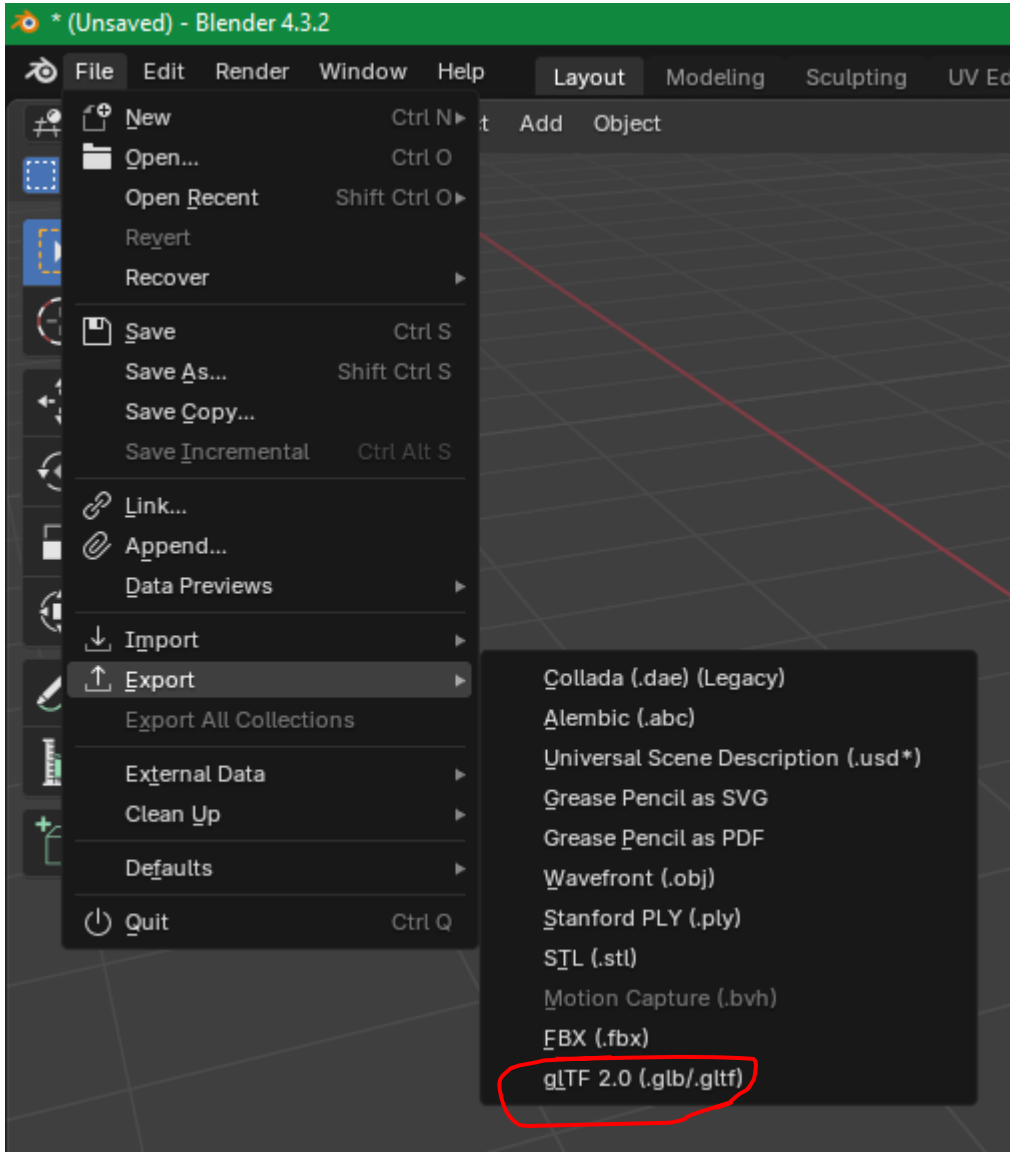


glb models

tworzymy model glb w aplikacji *Blender*



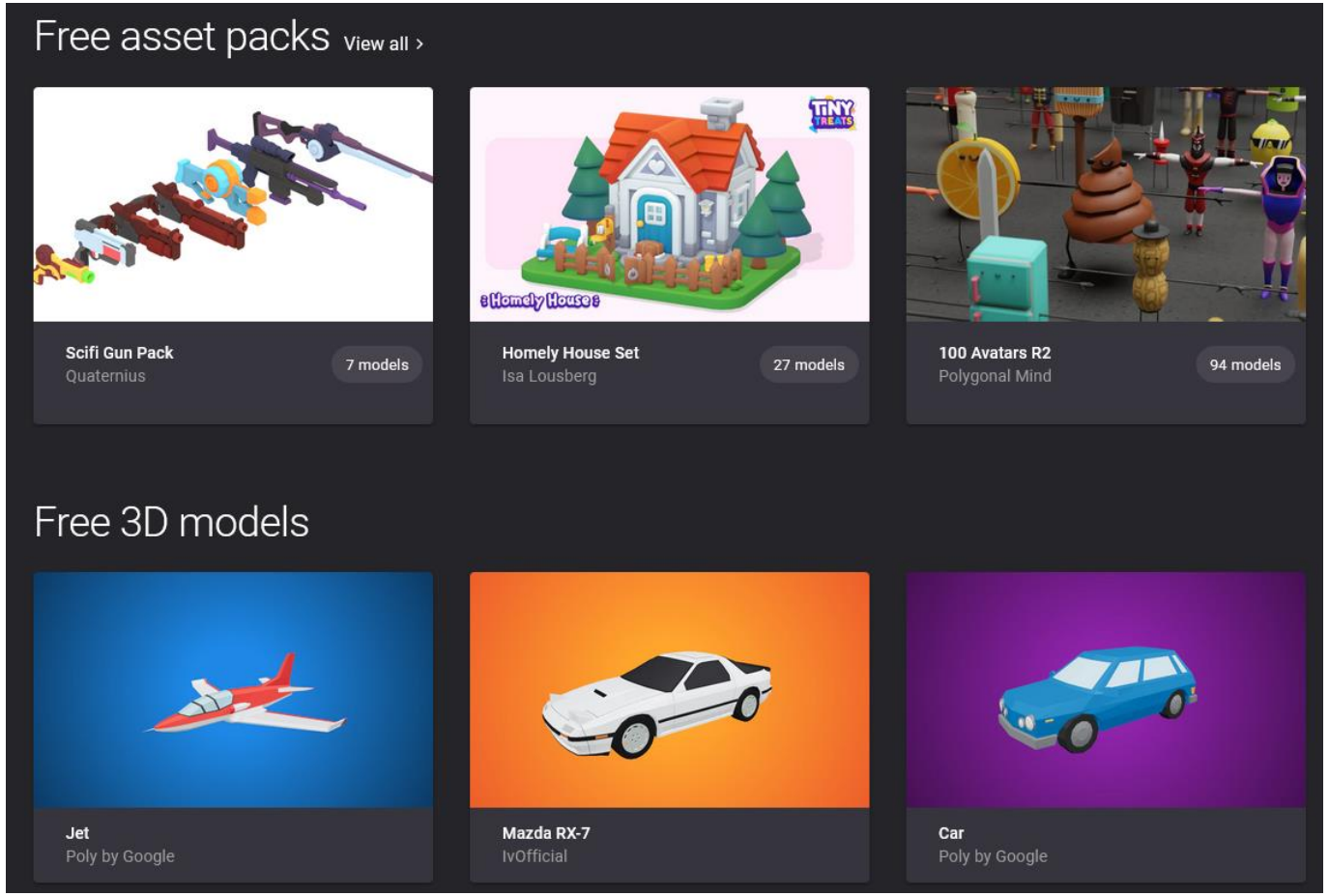
glb models



stworzony model eksportujemy jako plik glb

darmowe modele glb

free glb models



https://rigmodels.com/index.php?searchkeyword=glb-model

https://poly.pizza/

node.js instalacja

Run JavaScript Everywhere

Node.js® is a free, open-source, cross-platform JavaScript runtime environment that lets developers create servers, web apps, command line tools and scripts.

Download Node.js (LTS)

Downloads Node.js **v22.14.0**[1] with long-term support.
Node.js can also be installed via version managers.

Want new features sooner? Get **Node.js v23.8.0** [1] instead.

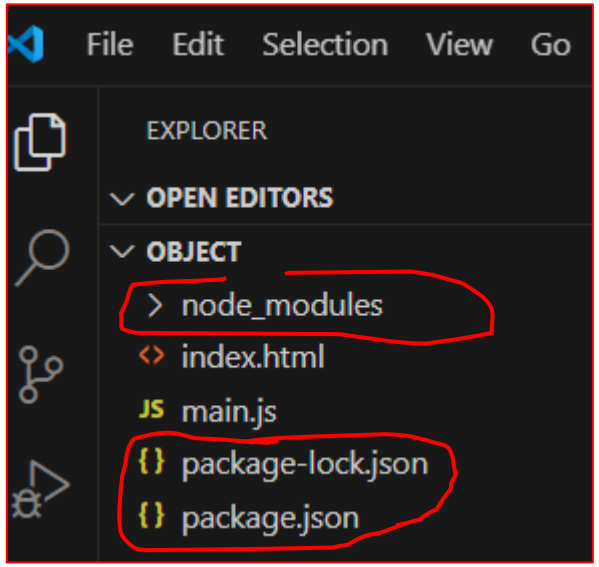
Learn more what Node.js is able to offer with our Learning materials.

https://nodejs.org/

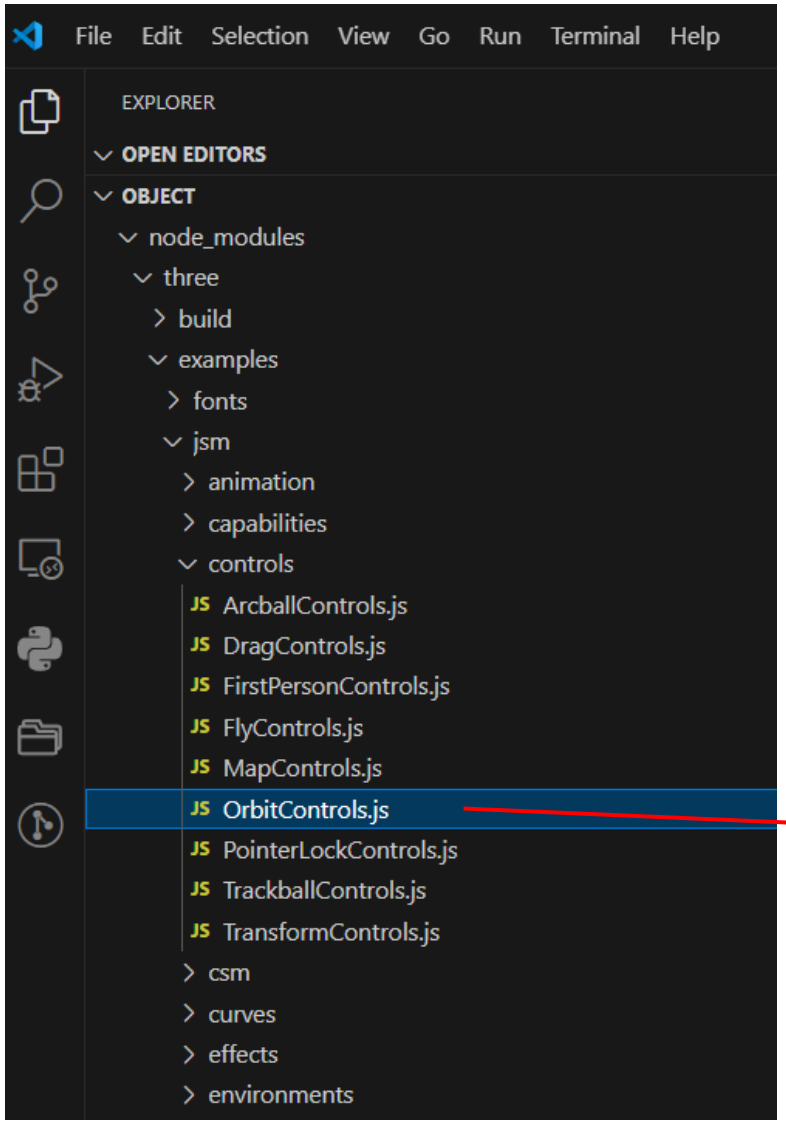
three.js instalacja

tworzymy katalog Object i instalujemy w nim three (w terminalu)

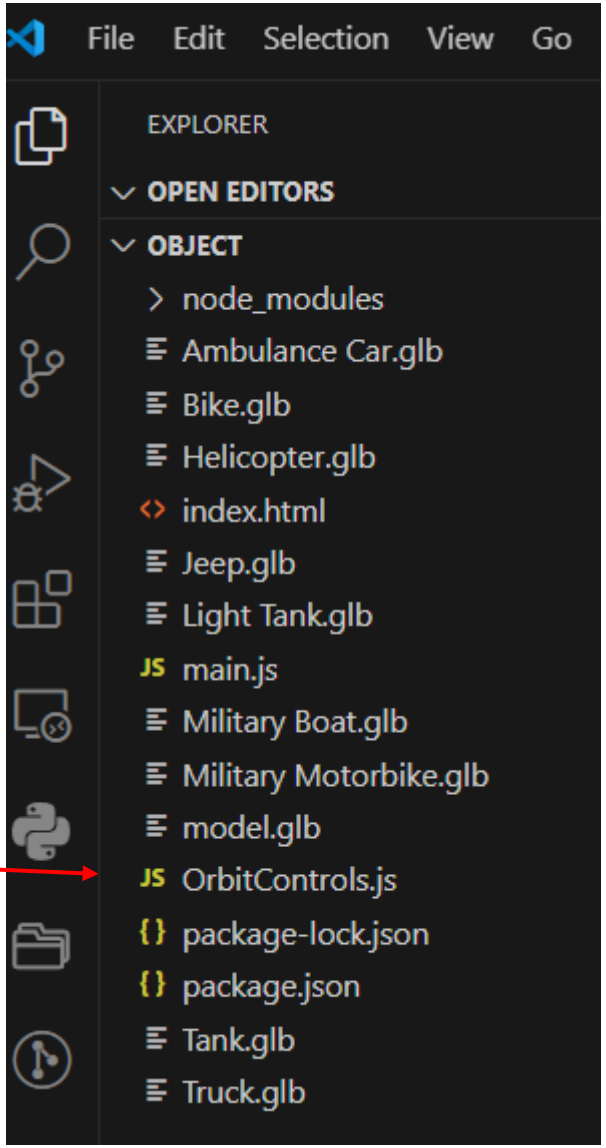
```
npm install --save three
```



glb models

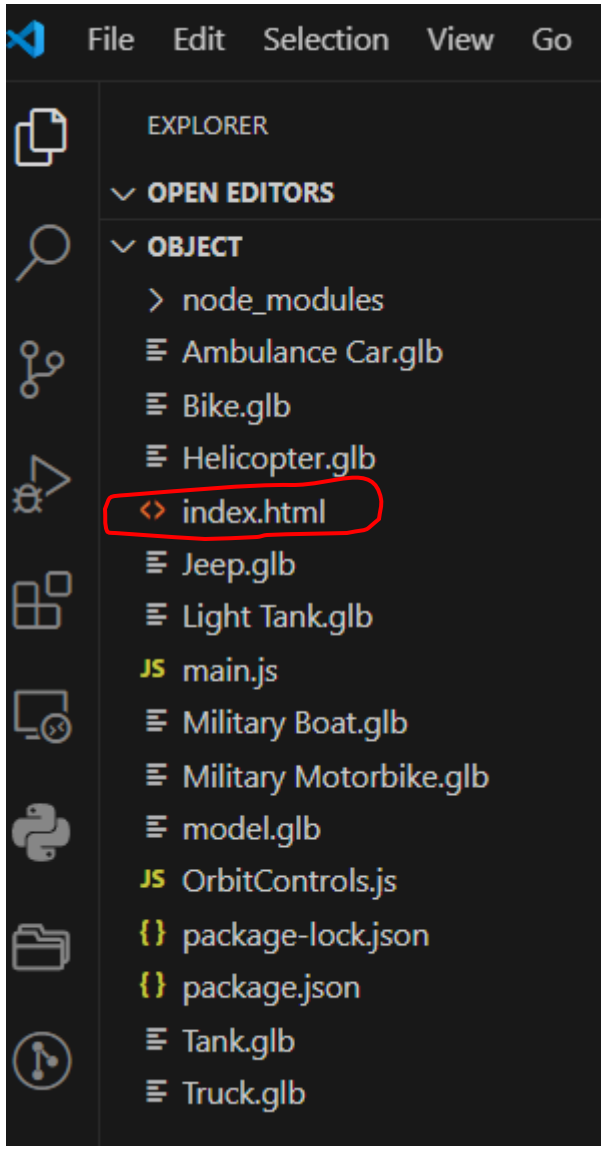


OrbitControls.js kopiujemy do katalogu projektu



glb models

index.html



index.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset='utf-8'>
    <title>Three.js</title>
    <script type="importmap">
        {
          "imports": {
            "three": "https://cdn.jsdelivr.net/npm/three@0.173.0/build/three.module.js",
            "three/addons/": "https://cdn.jsdelivr.net/npm/three@0.173.0/examples/jsm/"
          }
        }
      </script>
</head>
<body>
    <script type="module" src="/main.js"></script>
</body>
</html>

```

glb models

main.js 1/2

main.js

```
import * as THREE from 'three';
import { GLTFLoader } from 'three/addons/loaders/GLTFLoader.js';
import { OrbitControls } from './OrbitControls.js';

const scene = new THREE.Scene();
const camera = new THREE.PerspectiveCamera( 75, window.innerWidth / window.innerHeight, 0.1, 1000 );

const renderer = new THREE.WebGLRenderer();
renderer.setSize( window.innerWidth, window.innerHeight );
renderer.setPixelRatio( window.devicePixelRatio );
document.body.appendChild( renderer.domElement );

// Dodaj światło
const ambientLight = new THREE.AmbientLight( 0xffffff, 1.5 );
scene.add( ambientLight );

const directionalLight = new THREE.DirectionalLight( 0xffffff, 1.5 );
directionalLight.position.set( 1, 1, 1 ).normalize();
scene.add( directionalLight );

const pointLight = new THREE.PointLight( 0xffffff, 1, 100 );
pointLight.position.set( 0, 2, 0 );
scene.add( pointLight );

camera.position.z = 5;

// Utwórz OrbitControls
const controls = new OrbitControls( camera, renderer.domElement );

// Opcjonalnie: Ustaw punkt docelowy kontroli
controls.target.set( 0, 0, 0 );
controls.update();
```

glb models

main.js 2/2

nazwa modelu

```
33
34  renderer.setClearColor(0xffffff, 1); // Biały kolor tła
35
36  const loader = new GLTFLoader();
37
38  // załaduj glb model
39  loader.load( 'model.glb', function ( gltf ) {
40
41      scene.add( gltf.scene );
42
43  }, undefined, function ( error ) {
44
45      console.error('Błąd ładowania modelu:', error );
46
47  } );
48
49  const animate = function () {
50      requestAnimationFrame( animate );
51
52      controls.update();  // Ważne: Uaktualnij kontrolę w każdej klatce
53      renderer.render( scene, camera );
54  };
55
56  animate();
```

glb models

model stworzony w *Blenderze* na stronie internetowej