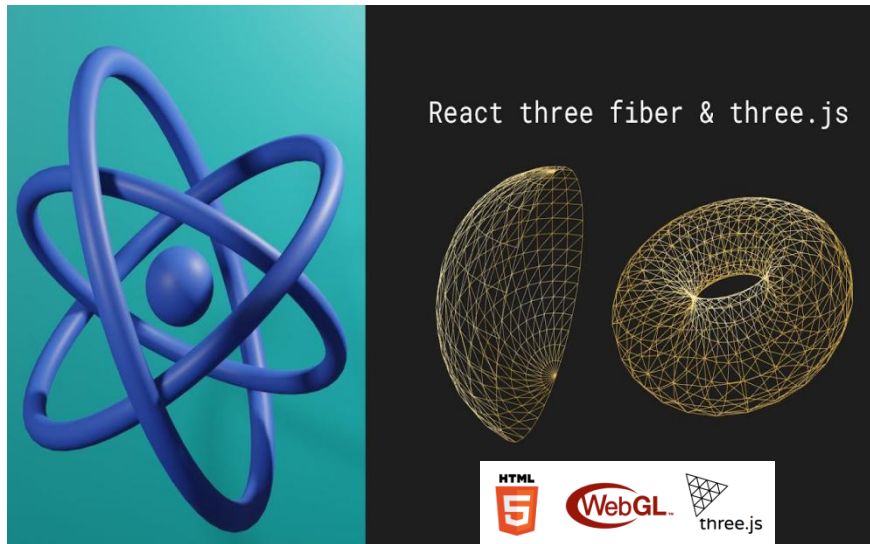


React-Three-Fiber jest to rozszerzenie popularnej biblioteki front-end'owej, którą jest React. Jest Ona nakładką na bibliotekę **Three.js**, dzięki czemu możemy tworzyć trójwymiarowe elementy graficzne wykorzystując przy tym architekturę komponentów funkcjonalnych oraz klasowych.



2. Implementacja aplikacji przy użyciu Three.js:

1. Pobranie pakietu **Three**:

```
C:\Users\PanCh\Desktop\Asystent - Praca\PAW - 5 Semestr Informatyka - AIiM\Lab10\Three.js - Project>npm install three
up to date, audited 2 packages in 751ms

found 0 vulnerabilities

C:\Users\PanCh\Desktop\Asystent - Praca\PAW - 5 Semestr Informatyka - AIiM\Lab10\Three.js - Project>npm ls
PanCh@ C:\Users\PanCh
|-- three@0.159.0

C:\Users\PanCh\Desktop\Asystent - Praca\PAW - 5 Semestr Informatyka - AIiM\Lab10\Three.js - Project>_
```

2. Utworzenie projektu składającego się z **index.html** oraz **main.js**:

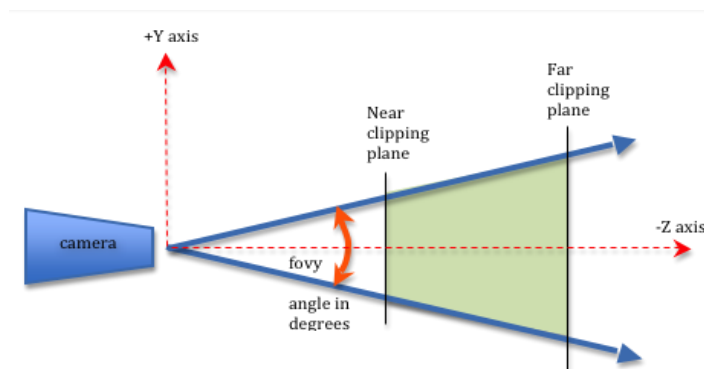
```
index.html X JS main.js
index.html > html
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="utf-8">
5     <title>Three.js Application - UMG </title>
6     <style>
7       body { margin: 0; }
8     </style>
9   </head>
10  <body>
11    <h1>Three.js Application
12    
13    </h1>
14    <script type="module" src="/main.js"></script>
15  </body>
16 </html>
```

3. Zaimportowanie biblioteki **Three**, zdefiniowanie parametrów **kamery**, utworzenie **sceny**, **kamery** oraz **renderera** graficznego:

```

index.html JS main.js X
JS main.js > ...
1  import * as THREE from 'three';
2
3  // PARAMETRY
4  var windowWidth = window.innerWidth // Szerokość okna.
5  var windowHeight = window.innerHeight // Wysokość okna.
6  var fieldOfView = 75 // Kąt widzenia kamery.
7  var aspectRatio = window.innerWidth / window.innerHeight // Współczynnik proporcji kamery.
8  var nearPlane = 0.1 // Minimalne pole widzenia kamery. ->(|)-----|
9  var farPlane = 100 // Maksymalne pole widzenia kamer. |-----(|) <-
10
11 // 1. Utworzenie sceny.
12 const scene = new THREE.Scene();
13
14 // 2. Utworzenie kamery.
15 const camera = new THREE.PerspectiveCamera(
16   fieldOfView, aspectRatio, nearPlane, farPlane
17 );
18
19 // 3. Utworzenie renderera.
20 const renderer = new THREE.WebGLRenderer();
21 renderer.setSize(windowWidth, windowHeight);
22 document.body.appendChild(renderer.domElement);

```



Dołączenie biblioteki **three** może wymagać podania pełnej ścieżki do pliku `Three.module.js`. Dla biblioteki zapisanej w katalogu bieżącego projektu będzie to:

```
import * as THREE from './node_modules/three/build/three.module.js'
```



Katedra Systemów Informatycznych
Uniwersytet Morski w Gdyni

4. Utworzenie obiektów do renderowania (szescian) oraz ich renderowanie:

```
// 4. Utworzenie obiektów do rysowania.

// 4.1 Utworzenie szescianu.
const geometry = new THREE.BoxGeometry(1, 2, 3); // GEOMETRIA
const material = new THREE.MeshBasicMaterial({color: 0x2a9df4}); // MATERIAŁ
const cube = new THREE.Mesh(geometry, material); // GEOMETRIA + MATERIAŁ
scene.add(cube)
camera.position.z = 5;

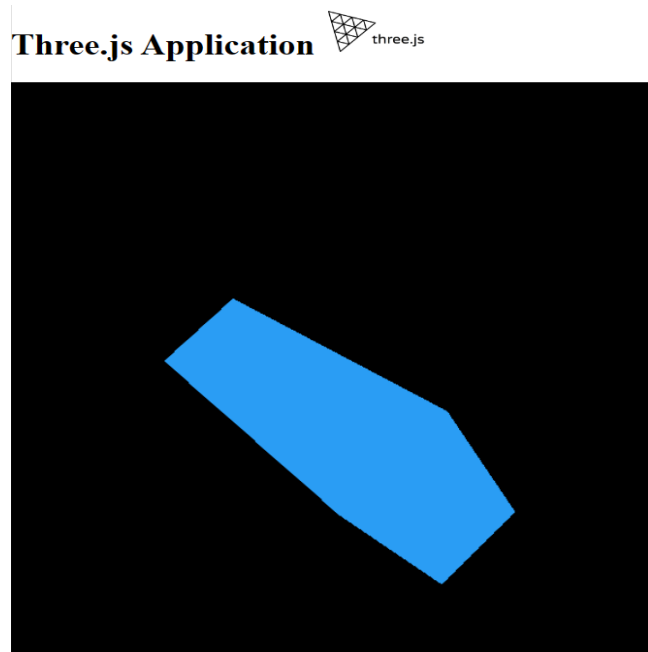
// 5. Animacja
function animateScene()
{
    cube.rotation.x += 0.01;
    cube.rotation.y += 0.01;
    cube.rotation.z += 0.01;

    requestAnimationFrame(animateScene)
    renderer.render(scene, camera)
}
animateScene()
```

5. Uruchomienie aplikacji:

```
PS C:\Users\PanCh\Desktop\Asystent - Praca\PAW - 5 Semestr Informatyka - AIIM\Lab10\Three.js - Project> npx vite
Port 5173 is in use, trying another one...

VITE v5.0.8 ready in 719 ms
  → Local:   http://localhost:5174/
  → Network: use --host to expose
  → press h + enter to show help
```

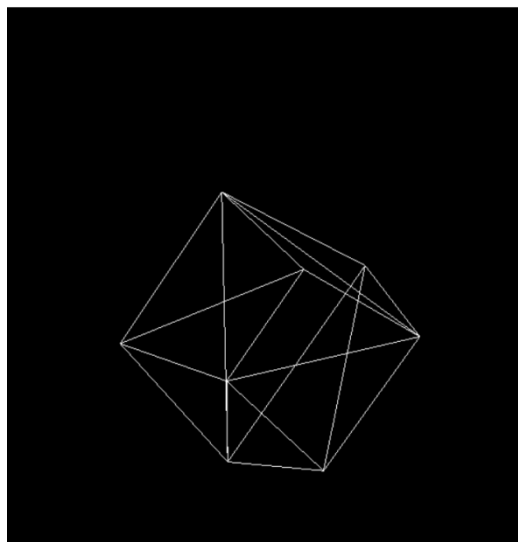




6. Ustawienie renderowania tylko siatki sześcianu:

```
// 1. Utworzenie sceny.  
const scene = new THREE.Scene();  
  
// 2. Utworzenie kamery.  
const camera = new THREE.PerspectiveCamera(  
    fieldOfView, aspectRatio, nearPlane, farPlane  
);  
  
// 3. Utworzenie renderera.  
const renderer = new THREE.WebGLRenderer();  
renderer.setSize(windowWidth, windowHeight);  
document.body.appendChild(renderer.domElement);  
  
// 4. Utworzenie obiektów do rysowania.  
const geometry = new THREE.BoxGeometry(2, 2, 2)  
const material = new THREE.MeshBasicMaterial({color: 0xffffff, wireframe: true})  
const cube = new THREE.Mesh(geometry, material)  
  
scene.add(cube)  
  
// 5. Animacja  
camera.position.z = 5;  
function animateScene()  
{  
    cube.rotation.x += 0.01;  
    cube.rotation.y += 0.01;  
    cube.rotation.z += 0.01;  
  
    requestAnimationFrame(animateScene)  
    renderer.render(scene, camera)  
}  
animateScene()
```

Three.js Application





7. Implementacja renderowania wielu sześcianów z listy:

```
// 4. Utworzenie obiektów do rysowania.

function getRandomValue(minValue, maxValue)
{
    return Math.floor(Math.random() * (maxValue - minValue + 1) ) + minValue;
}

function createCubeAttributes()
{
    var minValue = -2
    var maxValue = 2
    var randomColor = Math.floor(Math.random()*16777215);
    const cubeAttributes = {
        width:  getRandomValue(minValue, maxValue)/2,
        height: getRandomValue(minValue, maxValue)/2,
        depth:  getRandomValue(minValue, maxValue)/2,
        color: randomColor,
        position: {
            x: getRandomValue(minValue, maxValue),
            y: getRandomValue(minValue, maxValue),
            z: getRandomValue(minValue, maxValue)
        }
    }
    return cubeAttributes
}
```

```
function createCubes(cubes, numberOfCubes)
{
    for(let i=0; i<numberOfCubes; i++)
    {
        const cubeAttributes = createCubeAttributes()
        const geometry = new THREE.BoxGeometry( // GEOMETRIA
            cubeAttributes.width,
            cubeAttributes.height,
            cubeAttributes.depth
        );
        const material = new THREE.MeshBasicMaterial({color: cubeAttributes.color}); // MATERIAŁ
        const cube = new THREE.Mesh(geometry, material)
        cube.position.x = cubeAttributes.position.x
        cube.position.y = cubeAttributes.position.y
        cube.position.z = cubeAttributes.position.z
        cubes.push(cube)
    }
}

function addCubesToTheScene(cubes, numberOfCubes, scene)
{
    for(let i =0; i<numberOfCubes; i++)
    {
        scene.add(cubes[i])
    }
}
```



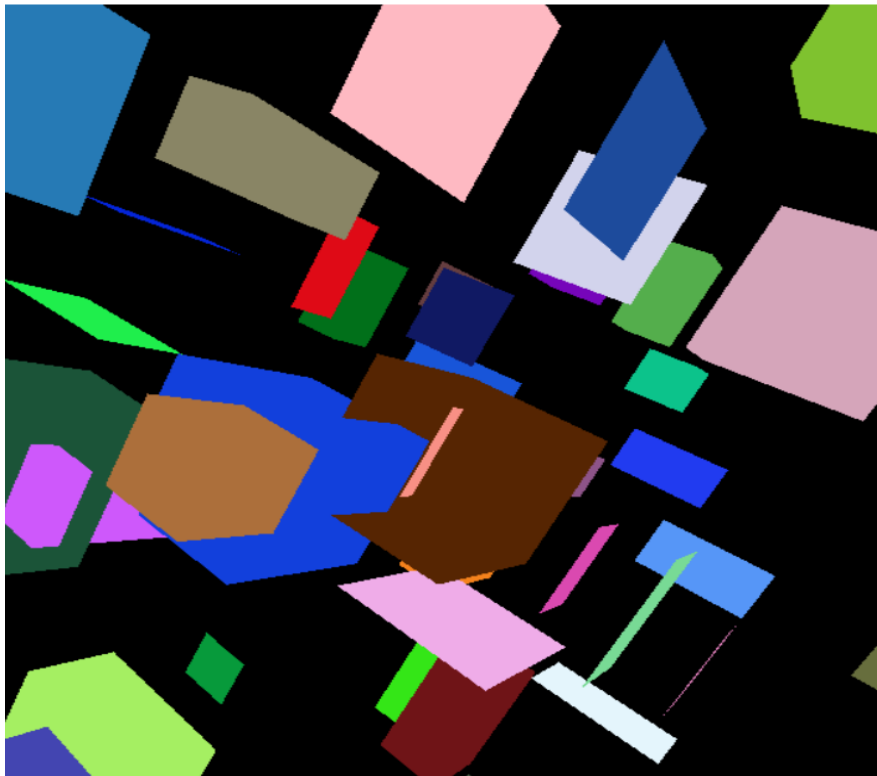
```
function addCubesToTheScene(cubes, numberOfCubes, scene)
{
    for(let i =0; i<numberOfCubes; i++)
    {
        scene.add(cubes[i])
    }
}

var cubes = []
var numberOfCubes = 50
createCubes(cubes, numberOfCubes)
addCubesToTheScene(cubes, numberOfCubes, scene)

// 5. Animacja
camera.position.z = 5;
function animateScene()
{
    for(let i =0; i<numberOfCubes; i++)
    {
        cubes[i].rotation.x += 0.01;
        cubes[i].rotation.y += 0.01;
        cubes[i].rotation.z += 0.01;
    }

    requestAnimationFrame(animateScene)
    renderer.render(scene, camera)
}
animateScene()
```

Three.js Application

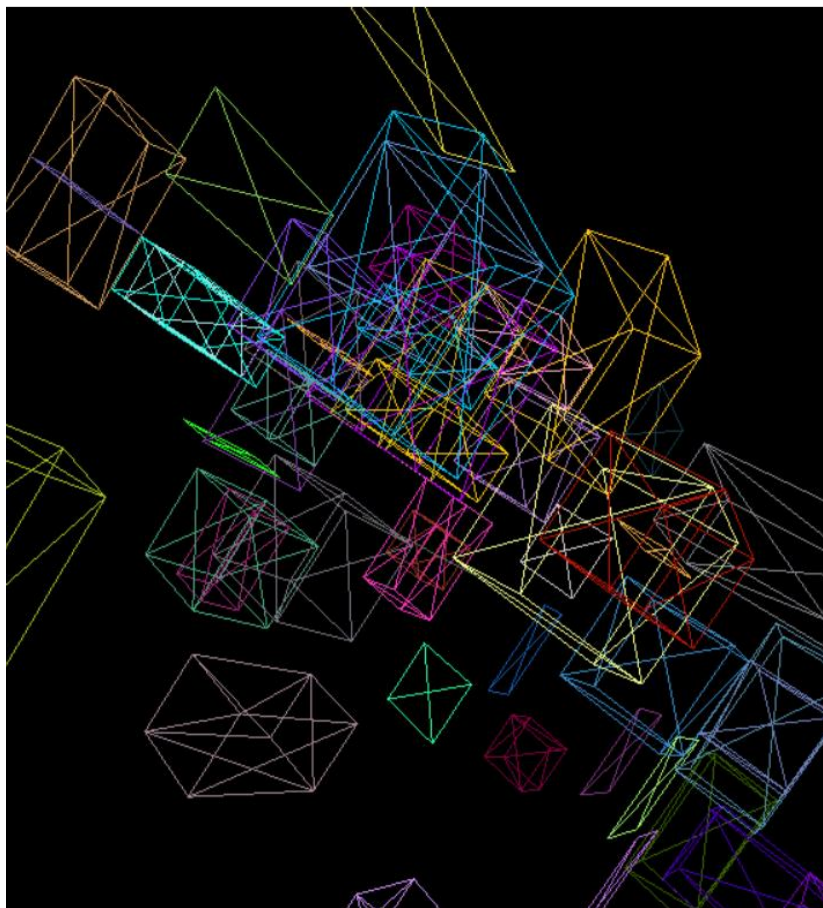




8. Renderowanie samych siatek sześciianów:

```
function createCubes(cubes, numberOfCubes)
{
  for(let i=0; i<numberOfCubes; i++)
  {
    const cubeAttributes = createCubeAttributes()
    const geometry = new THREE.BoxGeometry( // GEOMETRIA
      cubeAttributes.width,
      cubeAttributes.height,
      cubeAttributes.depth
    );
    const material = new THREE.MeshBasicMaterial({color: cubeAttributes.color, wireframe:true}); // MATERIAŁ
    const cube = new THREE.Mesh(geometry, material)
    cube.position.x = cubeAttributes.position.x
    cube.position.y = cubeAttributes.position.y
    cube.position.z = cubeAttributes.position.z
    cubes.push(cube)
  }
}
```

Three.js Application



9. Dodanie systemu świateł:

Ustawienie innego typu materiału każdego sześcianu:

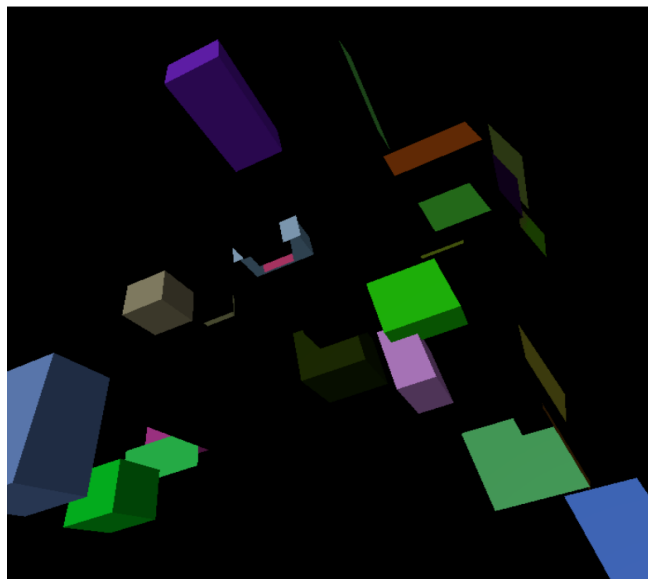
```
function createCubes(cubes, numberOfCubes)
{
  for(let i=0; i<numberOfCubes; i++)
  {
    const cubeAttributes = createCubeAttributes()
    const geometry = new THREE.BoxGeometry( // GEOMETRIA
      cubeAttributes.width,
      cubeAttributes.height,
      cubeAttributes.depth
    );
    const material = new THREE.MeshPhongMaterial({color: cubeAttributes.color}); // MATERIAŁ
    const cube = new THREE.Mesh(geometry, material)
    cube.position.x = cubeAttributes.position.x
    cube.position.y = cubeAttributes.position.y
    cube.position.z = cubeAttributes.position.z
    cubes.push(cube)
  }
}
```

Dodanie kierunkowego światła do sceny:

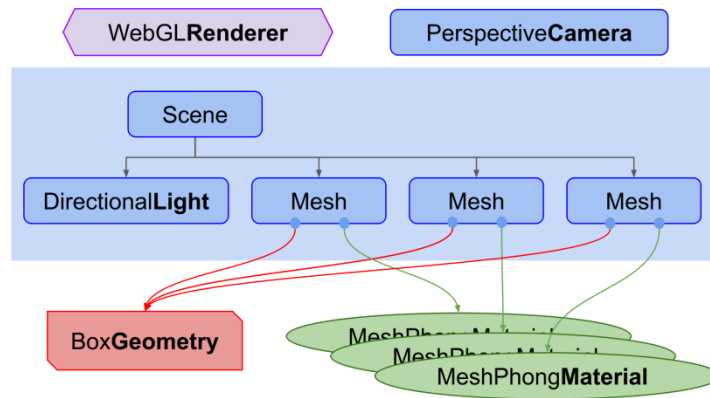
```
// Dodanie światła
const color = 0xFFFFFF;
const intensity = 1;
const light = new THREE.DirectionalLight(color, intensity);
light.position.set(-1, 2, 4);

scene.add(light);
```

Three.js Application 



10. Dodanie światła, które jest zawsze obecne:



```

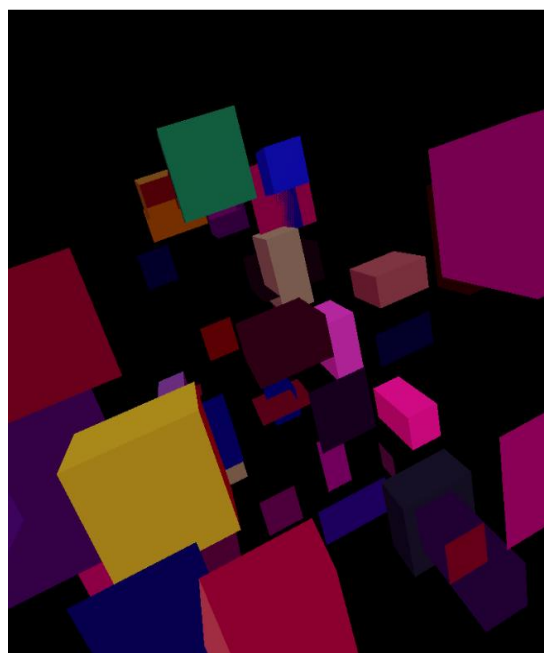
// Dodanie światła

// 1. Kierunkowego.
const directionalColor = 0xFFFFFF;
const directionalIntensity = 1.5;
const directionalLight = new THREE.DirectionalLight(directionalColor, directionalIntensity);
directionalLight.position.set(-1, 2, 4);

scene.add(directionalLight);

// 2. Zawsze obecnego.
const ambientColor = 0xFF0000AA;
const ambientIntensity = 1;
const ambientLight = new THREE.AmbientLight(ambientColor, ambientIntensity);
scene.add(ambientLight);
  
```

Three.js Application 




11. Dodanie możliwości kontroli sceny przy użyciu myszki:

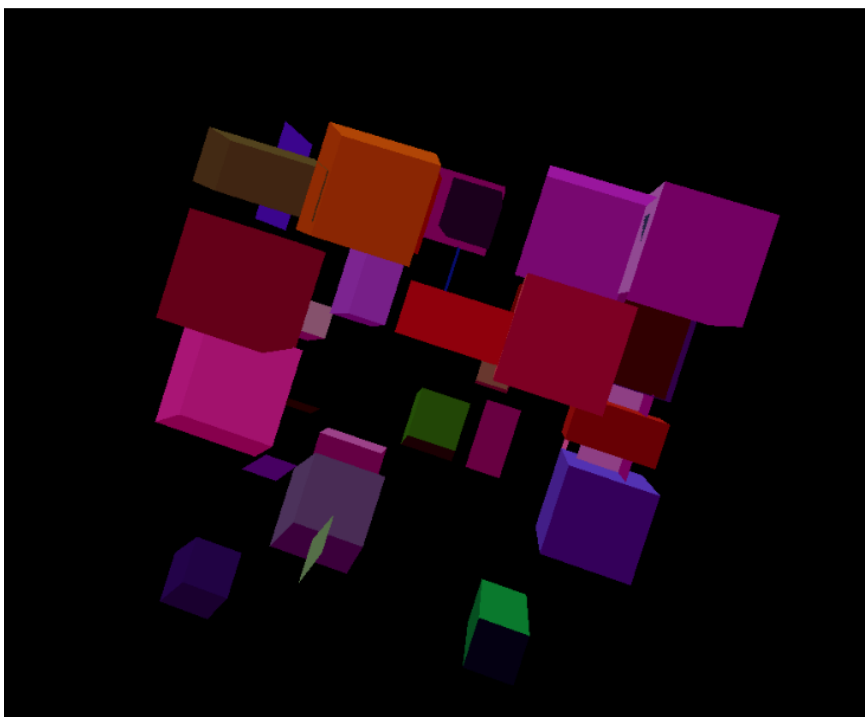
```
import * as THREE from 'three';
import {OrbitControls} from 'three/addons/controls/OrbitControls.js';

const controls = new OrbitControls(camera, renderer.domElement)
controls.target.set(0, 5, 0);

// 5. Animacja
camera.position.z = 5;
function animateScene()
{
    for(let i =0; i<numberOfCubes; i++)
    {
        cubes[i].rotation.x += 0.01;
        cubes[i].rotation.y += 0.01;
        cubes[i].rotation.z += 0.01;
    }

    requestAnimationFrame(animateScene)
    renderer.render(scene, camera)
}
animateScene()
```

Three.js Application  three.js



12. Dodanie podstawki wraz z teksturą:

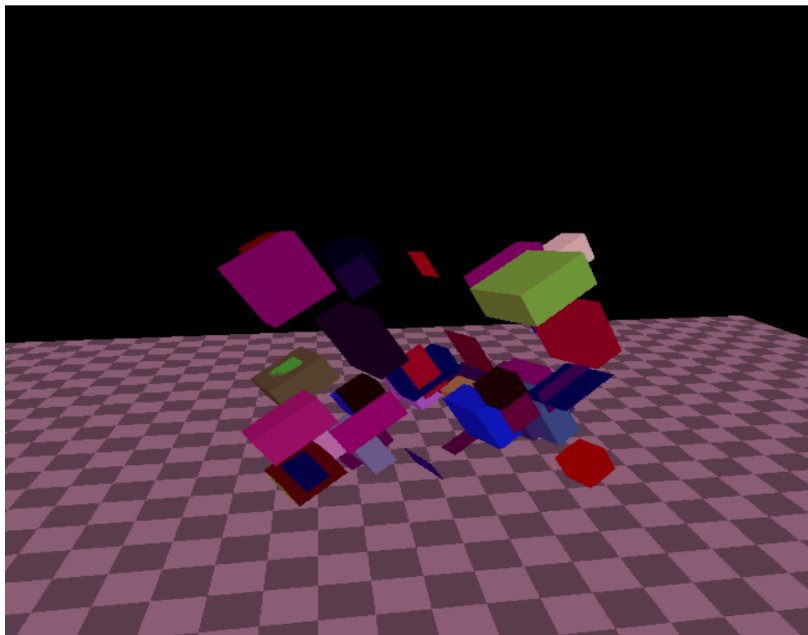
```
// DODANIE PODSTAWKI

// 1. Ustawienie Tekstury.
const planeSize = 40
const loader = new THREE.TextureLoader();
const texture = loader.load('checker.png');
texture.wrapS = THREE.RepeatWrapping;
texture.wrapT = THREE.RepeatWrapping;
texture.magFilter = THREE.NearestFilter;
texture.colorSpace = THREE.SRGBColorSpace;
const repeats = 40 / 2;
texture.repeat.set(repeats, repeats);

// 2. Utworzenie geometrii wraz z materiałem.
const planeGeo = new THREE.PlaneGeometry(planeSize, planeSize);
const planeMat = new THREE.MeshPhongMaterial({
  map: texture,
  side: THREE.DoubleSide,
});

// 3. Utworzenie siatki.
const planeMesh = new THREE.Mesh(planeGeo, planeMat);
planeMesh.rotation.x = Math.PI * -.5;
planeMesh.position.y = -5
scene.add(planeMesh)
```

Three.js Application

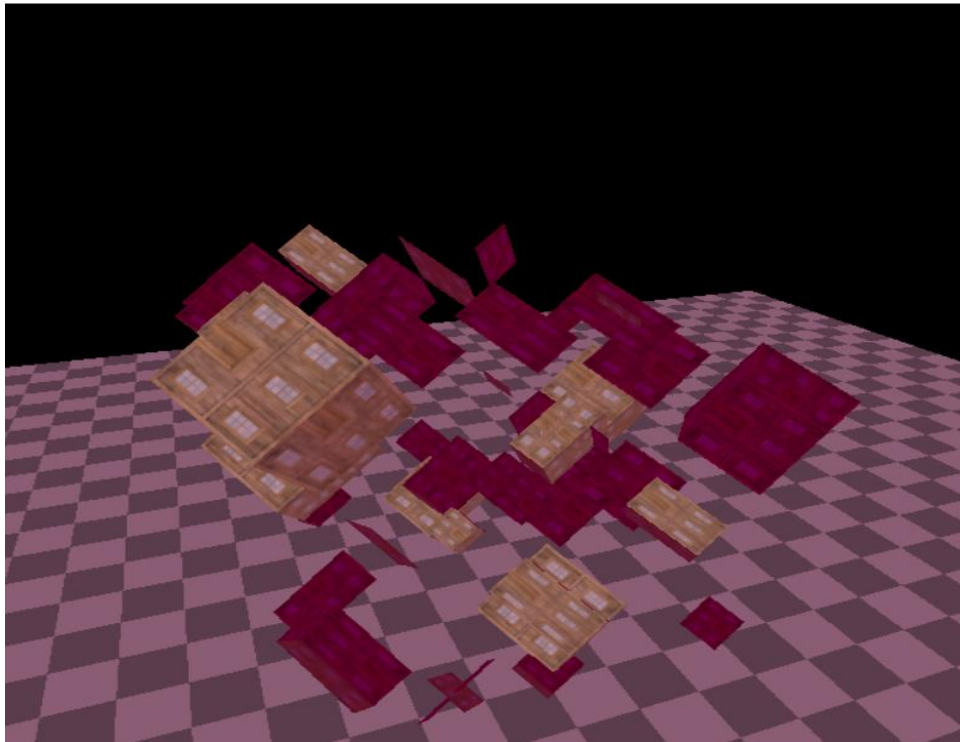




13. Dodanie tekstury każdemu sześciannowi:

```
function createCubes(cubes, numberOfCubes)
{
  for(let i=0; i<numberOfCubes; i++)
  {
    const cubeAttributes = createCubeAttributes()
    const geometry = new THREE.BoxGeometry( // GEOMETRIA
      cubeAttributes.width,
      cubeAttributes.height,
      cubeAttributes.depth
    );
    const texture = new THREE.TextureLoader().load( "wood.png" );
    const material = new THREE.MeshPhongMaterial({map: texture, depthTest:true}); // MATERIAŁ
    const cube = new THREE.Mesh(geometry, material)
    cube.position.x = cubeAttributes.position.x
    cube.position.y = cubeAttributes.position.y
    cube.position.z = cubeAttributes.position.z
    cubes.push(cube)
  }
}
```

Three.js Application



Zadanie 1

Utwórz **dowolną aplikację** wykorzystującą elementy **trójwymiarowej grafiki**, która będzie **bardziej zaawansowana** niż wyżej przedstawiona. Niech zawiera **różne bryły** np.: sfery, walce, które będą miały na sobie **teksturę**. Scena również ma zawierać **różne typy światła**.



3. Implementacja aplikacji przy użyciu React-Three-Fiber:

```
import { Canvas } from '@react-three/fiber'

function MyApp() {
  return (
    <Canvas>
      <group>
        <mesh>
          <meshNormalMaterial />
          <boxGeometry args={[2, 2, 2]} />
        </mesh>
      </group>
    </Canvas>
  )
}
```

Fiber

```
import * as THREE from 'three'

const scene = new THREE.Scene() // <Canvas>

const group = new THREE.Group() // <group>

const mesh = new THREE.Mesh() // <mesh />
const material = new THREE.MeshNormalMaterial() // <meshNormalMaterial />
const geometry = new THREE.BoxGeometry(2, 2, 2) // <boxGeometry />

mesh.material = material
mesh.geometry = geometry

group.add(mesh)
scene.add(group)
```

Three.js

1. Utworzenie projektu:

```
PS C:\Users\PanCh\Desktop\Asystent - Praca\PAW - 5 Semestr Informatyka - AIIM\Lab10> npx create-react-app "3d_app" ./
Creating a new React app in C:\Users\PanCh\Desktop\Asystent - Praca\PAW - 5 Semestr Informatyka - AIIM\Lab10\3d_app.
Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...

added 1463 packages in 40s
242 packages are looking for funding
run `npm fund` for details
```

2. Pobranie pakietów:

```
PS C:\Users\PanCh\Desktop\Asystent - Praca\PAW - 5 Semestr Informatyka - AIIM\Lab10\3d_app> npm install three @react-three/fiber
added 16 packages, and audited 1548 packages in 7s

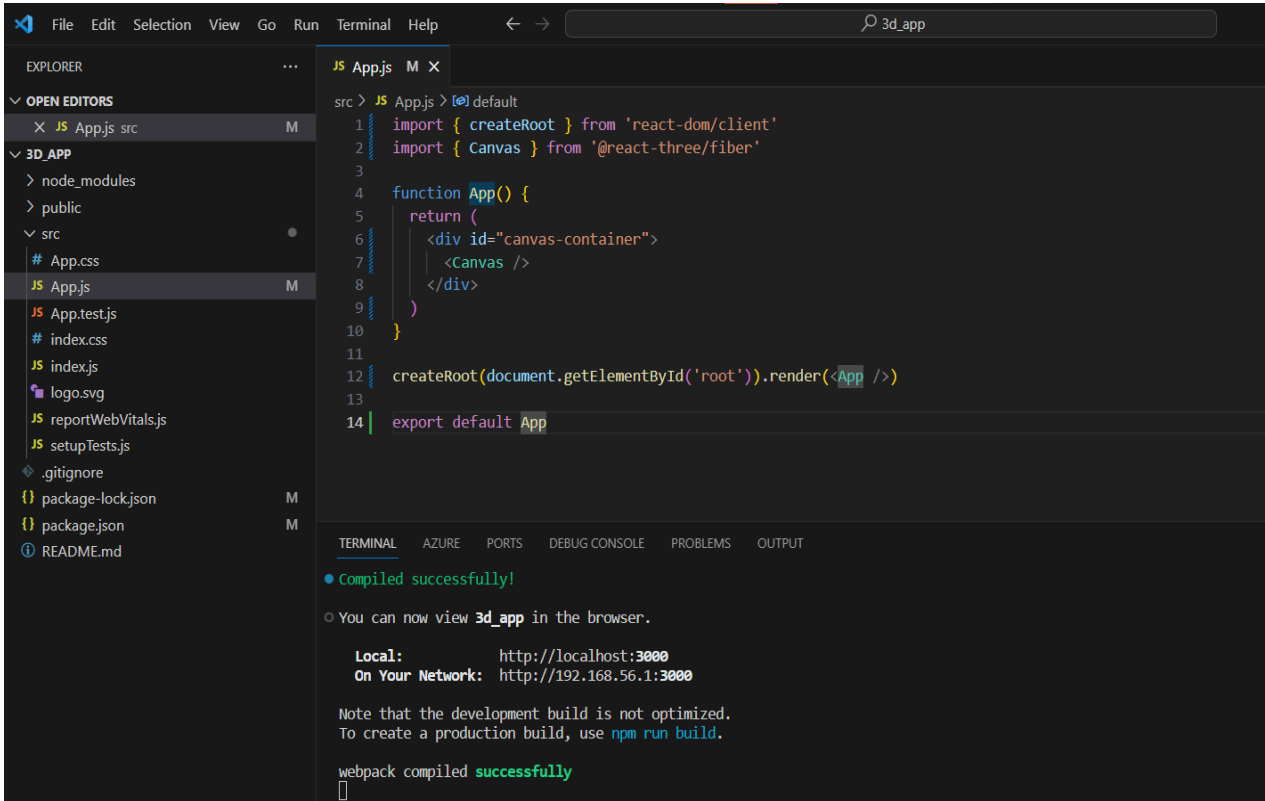
249 packages are looking for funding
run `npm fund` for details

8 vulnerabilities (2 moderate, 6 high)

To address all issues (including breaking changes), run:
npm audit fix --force

Run `npm audit` for details.
PS C:\Users\PanCh\Desktop\Asystent - Praca\PAW - 5 Semestr Informatyka - AIIM\Lab10\3d_app> █
```

3. Utworzenie projektu oraz zaimportowanie biblioteki:



```
src > JS App.js > [0] default
1 import { createRoot } from 'react-dom/client'
2 import { Canvas } from '@react-three/fiber'
3
4 function App() {
5   return (
6     <div id="canvas-container">
7       <Canvas />
8     </div>
9   )
10 }
11
12 createRoot(document.getElementById('root')).render(<App />)
13
14 export default App
```

Compiled successfully!

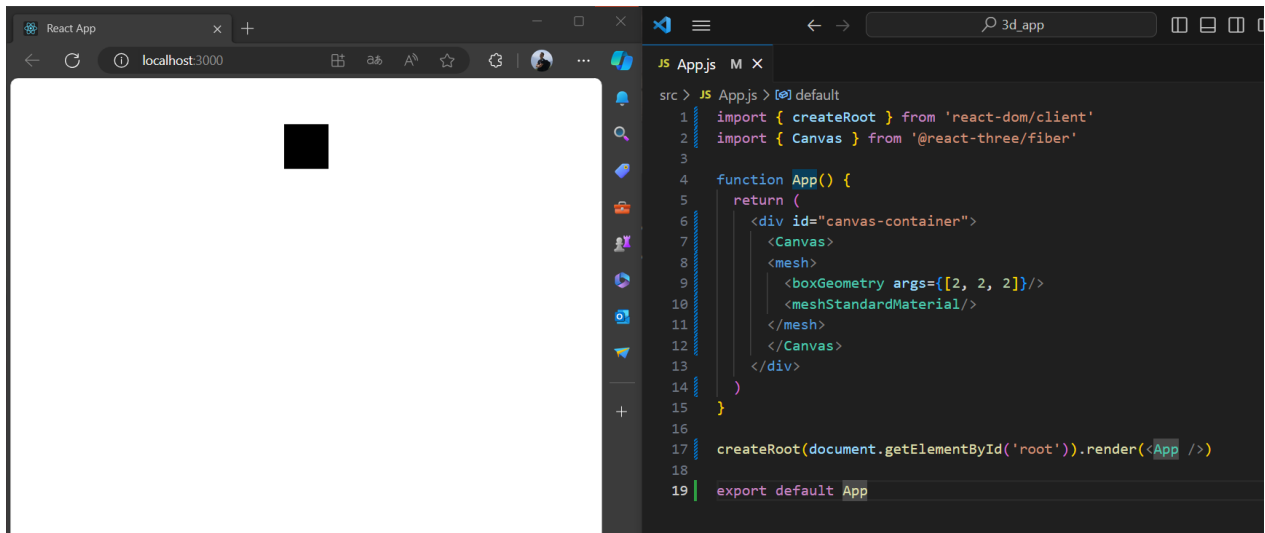
You can now view 3d_app in the browser.

Local: http://localhost:3000
On Your Network: http://192.168.56.1:3000

Note that the development build is not optimized.
To create a production build, use `npm run build`.

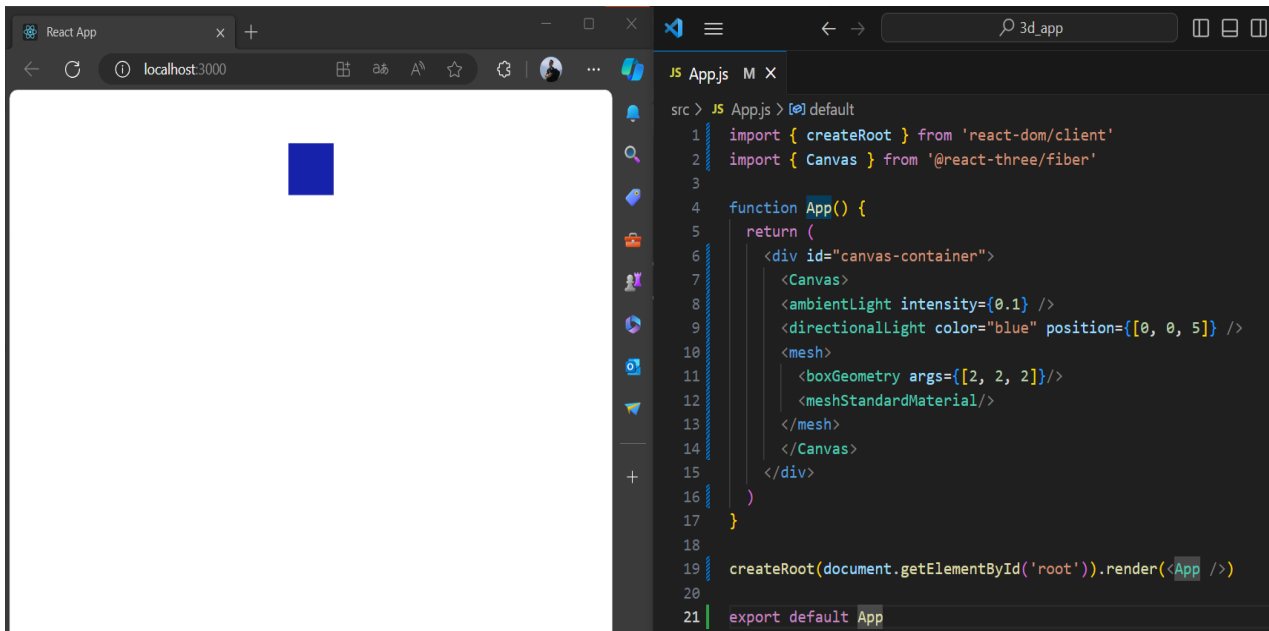
webpack compiled successfully

4. Utworzenie sześcianu:

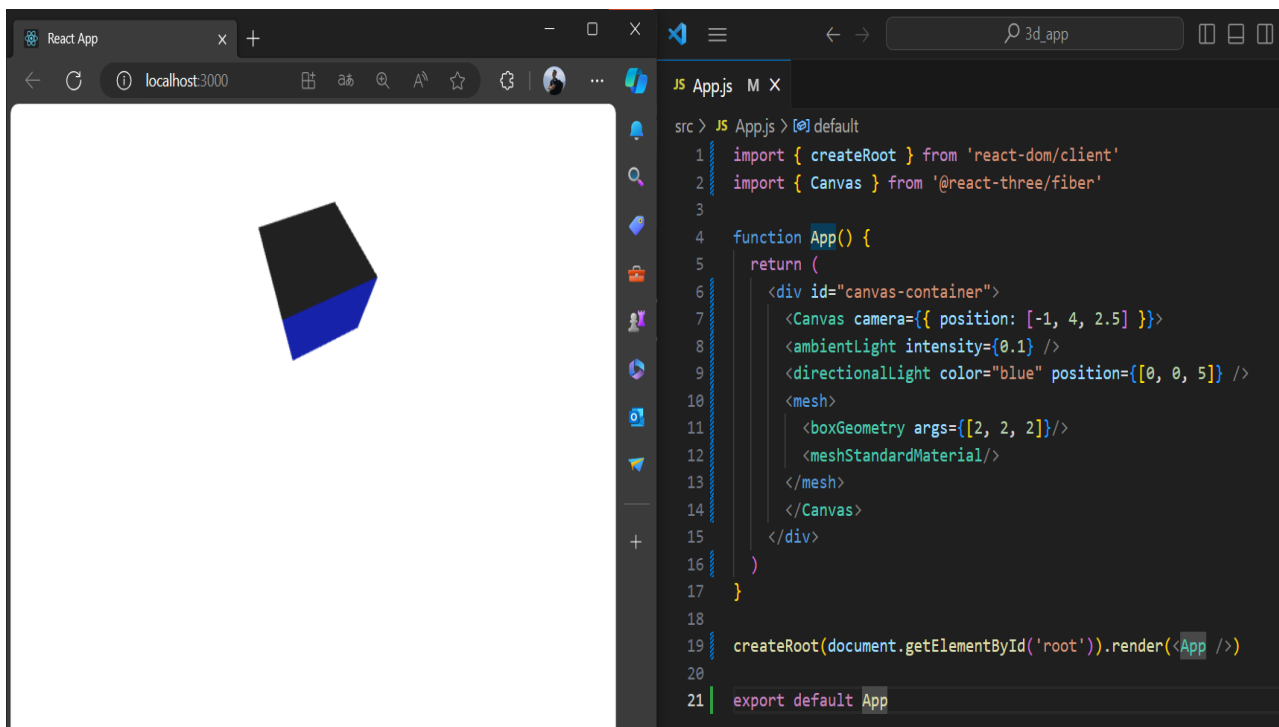


```
src > JS App.js > [0] default
1 import { createRoot } from 'react-dom/client'
2 import { Canvas } from '@react-three/fiber'
3
4 function App() {
5   return (
6     <div id="canvas-container">
7       <Canvas>
8         <mesh>
9           <boxGeometry args={[2, 2, 2]} />
10          <meshStandardMaterial />
11        </mesh>
12      </Canvas>
13    </div>
14  )
15 }
16
17 createRoot(document.getElementById('root')).render(<App />)
18
19 export default App
```

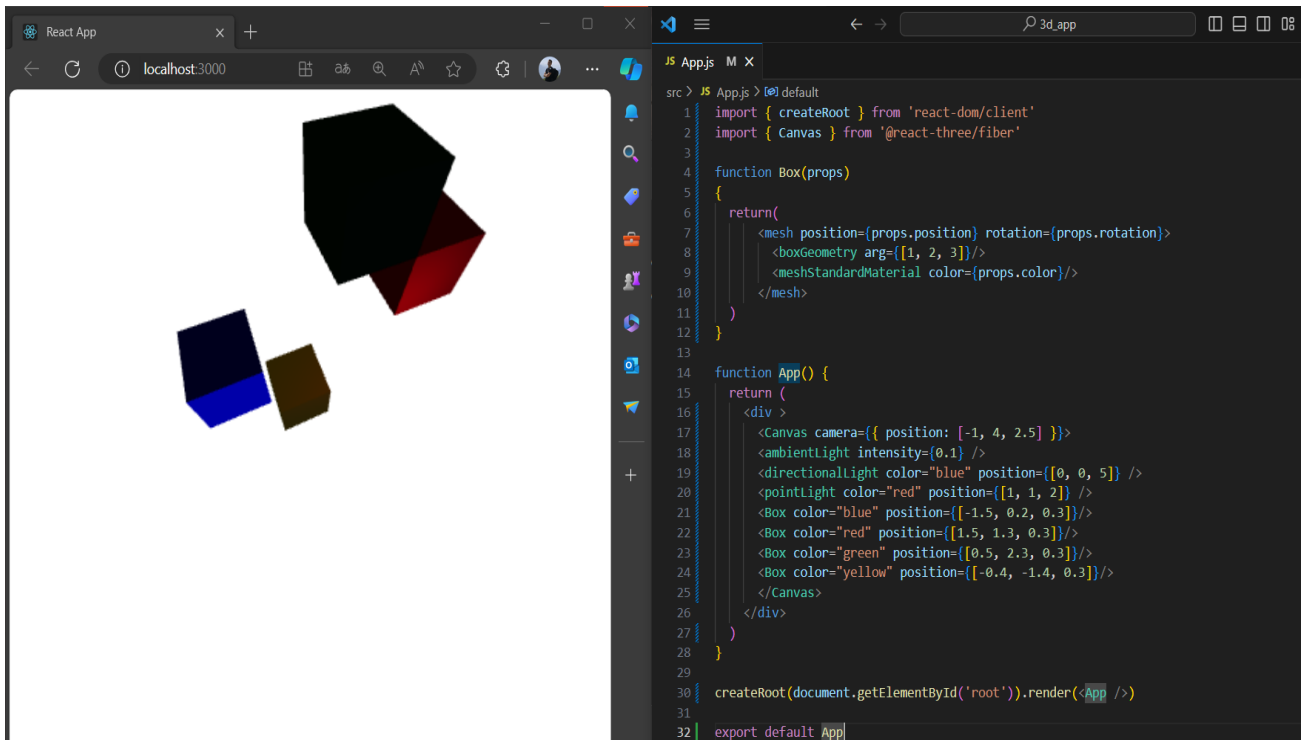
5. Dodanie świateł (kierunkowego oraz zawsze obecnego):



6. Zmiana umiejscowienia kamery:



7. Dodanie więcej sześcianów jako samodzielnych komponentów:



Zadanie 2

Utwórz taką samą aplikację jak w zadaniu 1, tylko przy użyciu biblioteki **React-Three-Fiber**.

Sprawozdanie

W sprawozdaniu wyślij linki do dwóch zadań wystawionych na serwerze foce oraz utworzone pliki źródłowe z własnymi kodami.