



Solar System - WebGL



WebGL / Three.js / ES6+ / JavaScript

<https://github.com/Dyzio18/solar-system-webGL>



<https://github.com/Dyzio18/solar-system-webGL>

Tech Stack

- **WebGL** - jest rozszerzeniem możliwości języka JavaScript, zapewniającym dostęp do trójwymiarowego API w przeglądarce internetowej.
- **Three.js** - biblioteka pozwalająca na korzystanie z WebGL
<https://github.com/mrdoob/three.js/>
- **JavaScript** (ES6+ / ECMAScript 2015) - nowa wersja języka JavaScript która wydawana jest od 2015 roku.
- **Webpack / babel** - transpilacja kodu JS(ES6) do starszej wersji obsługiwanej przez przeglądarki
- **NPM** - node package manager

JavaScript - ES6+

Czym jest ES6?

Jest to nowa wersja JS'a która od 2015 roku posiada coroczne wydania (release)

Nadzór nad standardem języka i jego rozwojem sprawuje organizacja ECMA
<https://tc39.github.io/ecma262/> <- Oficjalna dokumentacja

Jest to “syntax sugar” i ciągle rozszerzany żywy standard dla języka JavaScript

Dlaczego JavaScript?

Obecnie JS jest jednym z najpopularniejszych języków na świecie.

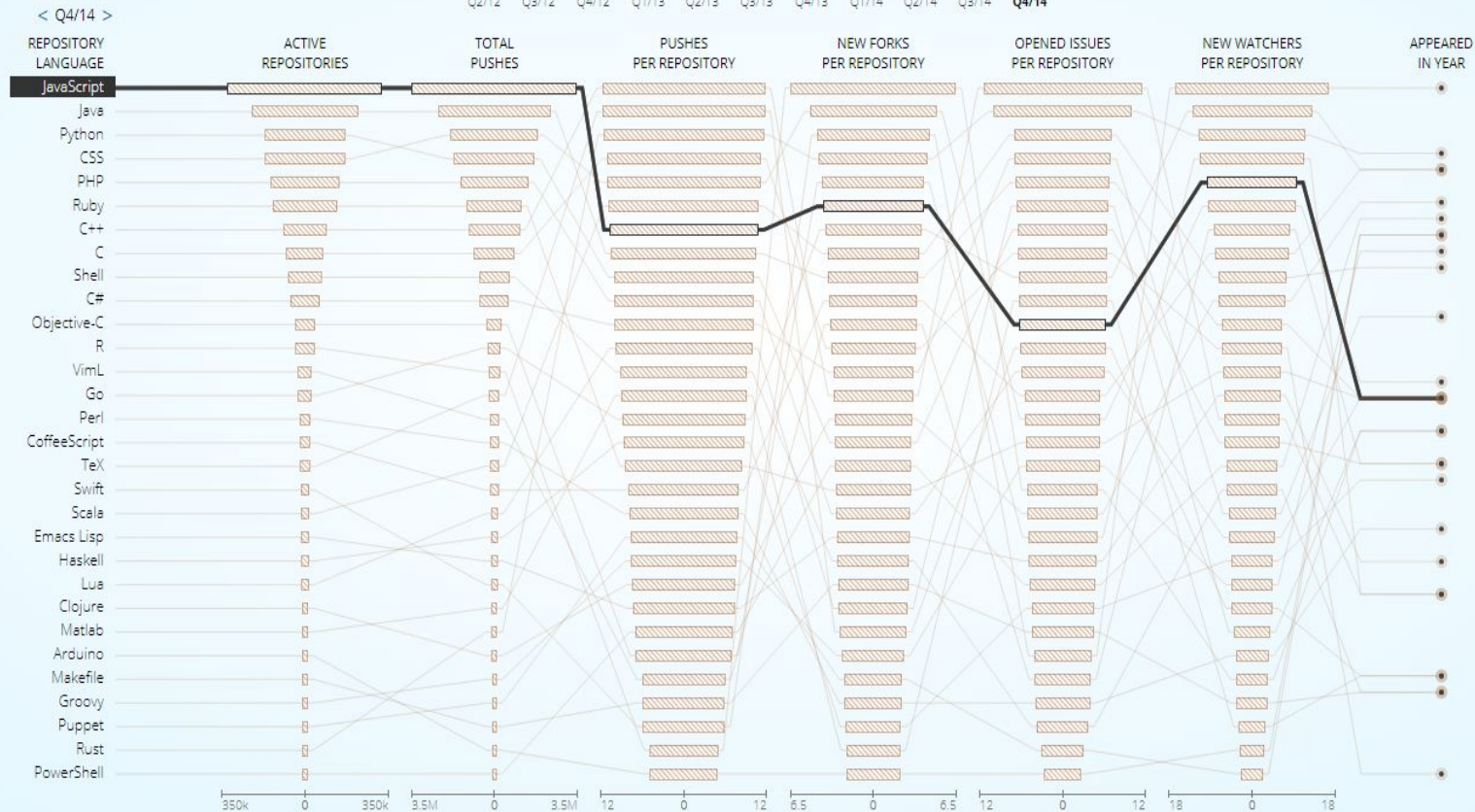
<http://github.info/> <- Najwięcej projektów opensource

- Multiplatformowość

- Desktop (Electron)
- Przeglądarki (React, Angular, Vue.js ...)
- Back-end (**Node.js**, Express, Hapi ...)
- Mobile (React Native, Ionic)

<https://risingstars2016.js.org/> <- ranking najpopularniejszych na podstawie gwiazdek na githubie

- Community



Więcej na temat JS...

- <https://github.com/kamranahmedse/developer-roadmap>
Spis technologii dla WebDeveloperów
- <https://github.com/sorrycc/awesome-javascript>
Spis narzędzi i technologii powiązanych z JS
- <https://github.com/airbnb/javascript>
Style Guide od airbnb

Książki:

- <https://github.com/getify/You-Dont-Know-JS> (Składnia języka)
- <https://addyosmani.com/resources/essentialjsdesignpatterns/book/> (Wzorce projektowe w JS)

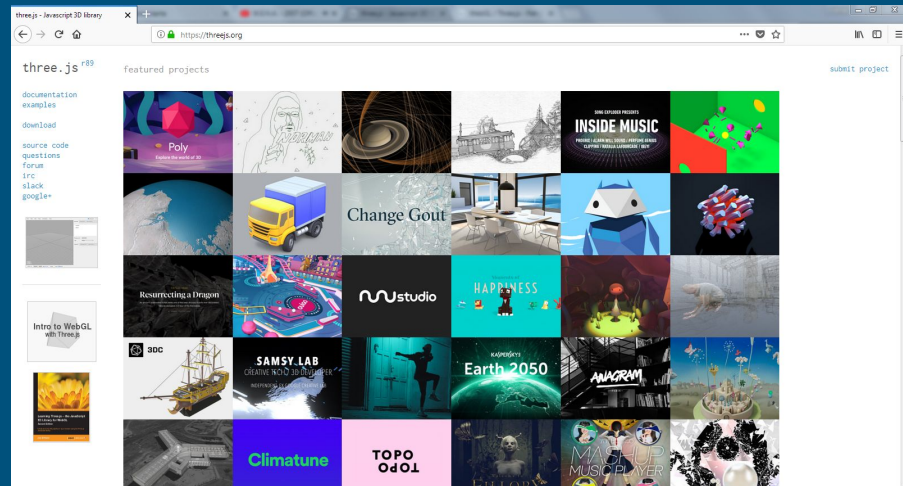
Three.js <https://threejs.org/>

Jest to biblioteka pozwalająca na korzystanie z WebGL. Stworzona została w 2010 roku przez Mr.doob (Ricardo Cabello) - obecnie jest współtworzona przez prawie tysiąc programistów. Wydana na licencji MIT.

W dniu pisania aplikacji korzystałem z wersji 89.

Na stronie biblioteki znajdziemy dokumentację oraz setki przykładów jej wykorzystania.

Biblioteka ma rozbudowane community które zawsze służy pomocą, czasem na pytania odpisze sam autor Mr.doob...



<https://github.com/Dyzio18/solar-system-webGL>

Manual

Getting Started

- Creating a scene
- Import via modules
- Browser support
- WebGL compatibility check
- How to run things locally
- Drawing Lines
- Creating Text
- Migration Guide
- Code Style Guide
- FAQ
- Useful links

Next Steps

- How to update things
- Matrix transformations
- Animation System

Build Tools

Testing with NPM

Reference

Animation

- AnimationAction
- AnimationClip
- AnimationMixer
- AnimationObjectGroup
- AnimationUtils
- KeyframeTrack

Before we start

Before you can use three.js, you need somewhere to display it. Save the following HTML to a file on your computer, along with a copy of [three.js](#) in the js/ directory, and open it in your browser.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset=utf-8>
    <title>My first three.js app</title>
    <style>
      body { margin: 0; }
      canvas { width: 100%; height: 100% }
    </style>
  </head>
  <body>
    <script src="js/three.js"></script>
    <script>
      // Our Javascript will go here.
    </script>
  </body>
</html>
```

That's all. All the code below goes into the empty <script> tag.

Creating the scene

To actually be able to display anything with three.js, we need three things: scene, camera and renderer, so that we can render the scene with camera.

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

var renderer = new THREE.WebGLRenderer();
renderer.setSize( window.innerWidth, window.innerHeight );
document.body.appendChild( renderer.domElement );
```


Projekt - Solar System

Kod aplikacji dostępny jest na Githubie pod adresem:

<https://github.com/Dyzio18/solar-system-webGL>

Wersja LIVE:

<https://dyzio18.github.io/solar-system-webGL/>

Aplikacja wyświetla trójwymiarowy model układu słonecznego.

Po kliknięciu na planetę wyświetla się krótka notka na jej temat. Planety są względem siebie proporcjonalne i krążą wokół słońca z odpowiednim okresem. Dla zachowania czytelności słońce jak i odległości między planetami są mniejszych rozmiarów niż w rzeczywistości.

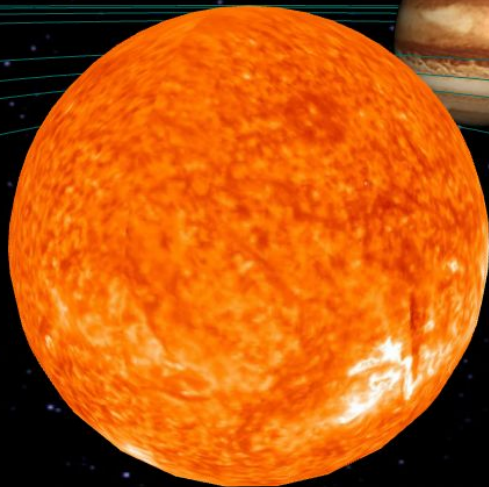
<https://github.com/Dyzio18/solar-system-webGL>

SŁOŃCE

Gwiazda centralna Układu Słonecznego, wokół której krąży Ziemia, inne planety tego układu, planety karłowate oraz małe ciała Układu Słonecznego.

Słońce uformowało się około 4,567 mld lat temu. Słońce to najjasniejszy obiekt na niebie i główne źródło energii docierającej do Ziemi. Zbudowane głównie z wodoru i helu. Ciśnienie 250 mld atmosfer. Temperatura na powierzchni ok. 6000 K.

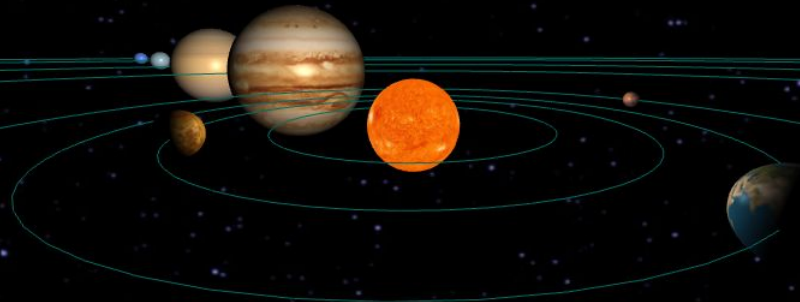
Średnica ok. 1,4 mln km. Pokryte jest ciemnymi plamami.



ZIEMIA

Trzecia według oddalenia od Słońca planeta Układu Słonecznego. Jest ona największa ze wszystkich planet wewnętrznych. Powierzchnię głównie zajmują oceany. Obieg wokół Słońca 365 dni. Obrót wokół własnej osi 24 h.

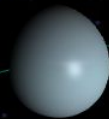
Odległość od Słońca wynosi jedną jednostkę astronomiczną 150 mln km. Temp. Powierzchni 14°C. Atmosfera właściwa, złożona z 78% azotu, 21% tlenu i 1% innych gazów. Posiada 1 naturalnego satelitę, czyli Księżyc.



URAN

Siódma według oddalenia od Słońca planeta Układu Słonecznego. Jest to planeta gazowa, zewnętrzna. Został on odkryty w 1781r. Spowity metanem. Obieg wokół Słońca 84 lata.

Obrot wokół własnej osi – 17 h.
Średnica 52 400 km. Odległość od Słońca 2 900 mln km. Temp. pow. od -211°C do -213°C . Posiada pierścienie i 15 księżyców (Miranda).



Kod Aplikacji - zależności (package.json)

```
{
  "name": "webgl_app",
  "version": "1.0.0",
  "description": "webGL app",
  "main": "./dist/index.js",
  "scripts": {
    "start": "",
    "lint": "./node_modules/.bin/eslint ./src/**/*.js --fix --cache",
    "build": "webpack --config webpack.config.js",
    "build:watch": ".\\node_modules\\.bin\\webpack --config webpack.config.js --watch"
  },
  "keywords": [
    "tree.js",
    "webGL"
  ],
  "author": "patryk.nizio@gmail.com",
  "license": "MIT",
  "dependencies": {
    "orbit-controls-es6": "^1.0.10",
    "three": "^0.89.0"
  },
  "devDependencies": {
    "babel-core": "^6.25.0",
    "babel-loader": "^7.1.1",
    "babel-plugin-transform-class-properties": "^6.24.1",
    "babel-plugin-transform-runtime": "^6.23.0",
    "babel-polyfill": "^6.23.0",
    "babel-preset-es2015": "^6.24.1",
    "babel-preset-es2016": "^6.24.1",
    "babel-preset-es2017": "^6.24.1",
    "eslint": "^4.6.1",
    "webpack": "^3.5.5",
    "webpack-node-externals": "^1.6.0"
  }
}
```

Kod aplikacji - HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title> Solar System </title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <div id="canvas" > </div>
  <div id="description" class="info__panel">
  </div>
<script src="dist/index.bundle.js"></script>
</body>
</html>
```

W pliku index.html podpinam skrypt JS.

Skrypt jest przetranspilowaną wersją ES6 i tworzy tzw. bundle czyli plik wynikowy.

W elemencie #canvas zostanie podpięta animowana scena

Kod aplikacji - main.js

```
import * as THREE from 'three';  
import OrbitControls from 'orbit-controls-es6';  
import {solarSystemCreate, solarSystemMove} from  
  './SolarSystem';  
import {dataArray} from './Helpers';  
import {Stats} from './Stats';
```

```
let camera, scene, renderer, controls, descPanel;  
let planets = {sun: {}, ...};
```

```
init();  
animate();  
window.addEventListener("resize", resize);  
window.addEventListener("click", onMouseMove,  
  false);
```

Importuje zależności do projektu oraz tworzone przeze mnie moduły.

Następnie tworzę instancje obiektów globalnych takich jak scena, kontrolery, oraz planety

Wywołuje funkcję i podpinam handlers na zdarzenia resize i click

Kod aplikacji - main.js

```
/* Tworzenie sceny */
scene = new THREE.Scene();

/* Inicjalizacja kamery */
camera = new THREE.PerspectiveCamera(100, window.innerWidth / window.innerHeight, 0.01, 1000);

/* Podpięcie animacji pod element w HTML-u */
const container = document.getElementById('canvas');
document.body.appendChild(container);

    renderer = new THREE.WebGLRenderer({antialias: true, alpha:true});
    renderer.setSize(window.innerWidth * 0.99, window.innerHeight * 0.99);
    container.appendChild(renderer.domElement);
```



```
// Animacja obiektów (Układu słonecznego) -  
// plannets przekazywane przez referencje do funkcji solarSystemMove
```

```
function animate() {  
    stats.begin(); // Stats  
    setTimeout(() => {  
        controls.update();  
        solarSystemMove(planets);  
        render();  
        stats.end(); // Stats  
    }, 10);  
    requestAnimationFrame(animate);  
}
```

```
// Renderowanie sceny  
function render() {  
    renderer.render(scene, camera);  
}
```

Kod aplikacji - SolarSystem.js

```
/* Funkcja dodaje planety do sceny */  
const solarSystemCreate = (scene, planets, render) => {  
    let loader = new THREE.TextureLoader();  
    let texture, orbitCircle, orbit;  
  
    scene.background = loader.load(`https://raw.githubusercontent.com/.../img/stars.jpg`, render);  
  
    /* Mapuje wszystkie elementy tablicy solarSystemData */  
    solarSystemData.map(sphere => {  
        /* Nakładam teksturę */  
        texture = loader.load(`https://raw.... /Dyzio18/.../master/img/${sphere.name}.jpg`, render);  
        texture.wrapS = texture.wrapT = THREE.RepeatWrapping;  
        texture.matrixAutoUpdate = false;  
        ...  
    })  
}
```

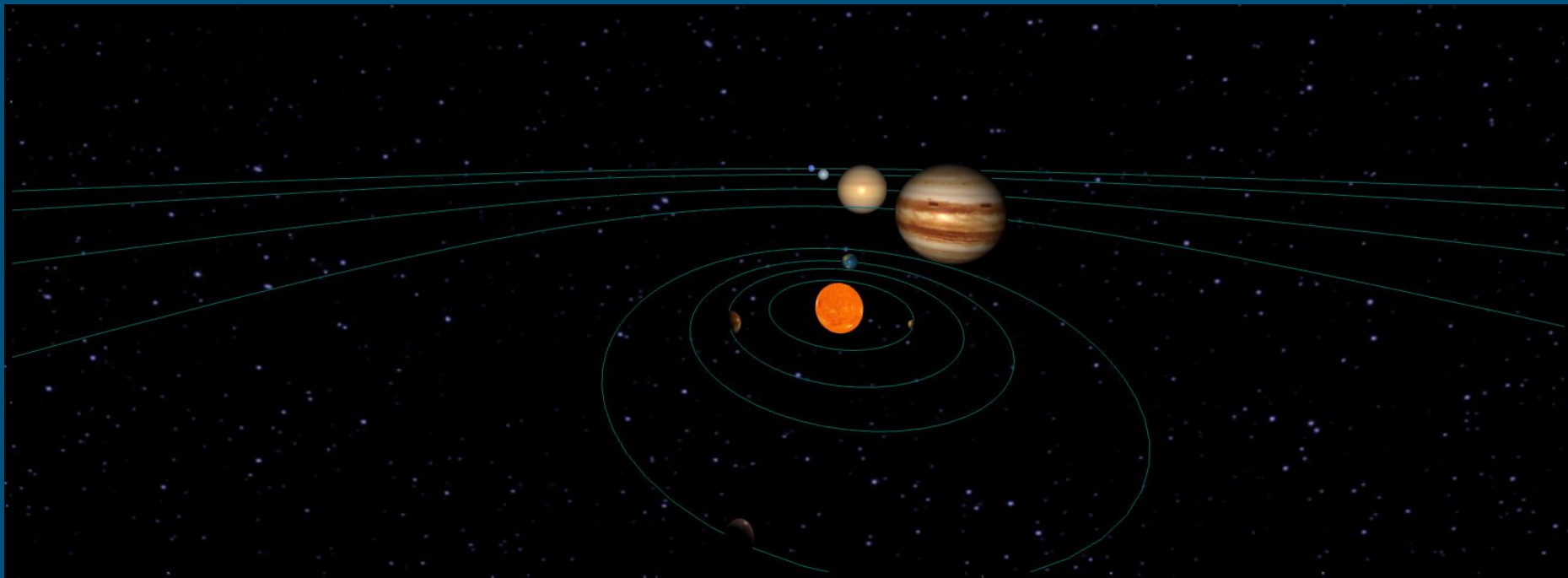
Kod aplikacji (SolarSystem.js) - tworzenie sfer

```
if (sphere.name === 'sun') {  
    planets[sphere.name] = new THREE.Mesh(new THREE.SphereBufferGeometry(sphere.radius, 32, 32),  
    new THREE.MeshBasicMaterial({map: texture}));  
}  
else {  
    planets[sphere.name] = new THREE.Mesh(new THREE.SphereBufferGeometry(sphere.radius, 32, 32),  
    new THREE.MeshPhongMaterial({  
        color: 0xffffff,  
        specular: 0x050505,  
        shininess: 100,  
        map: texture  
    }));  
}
```

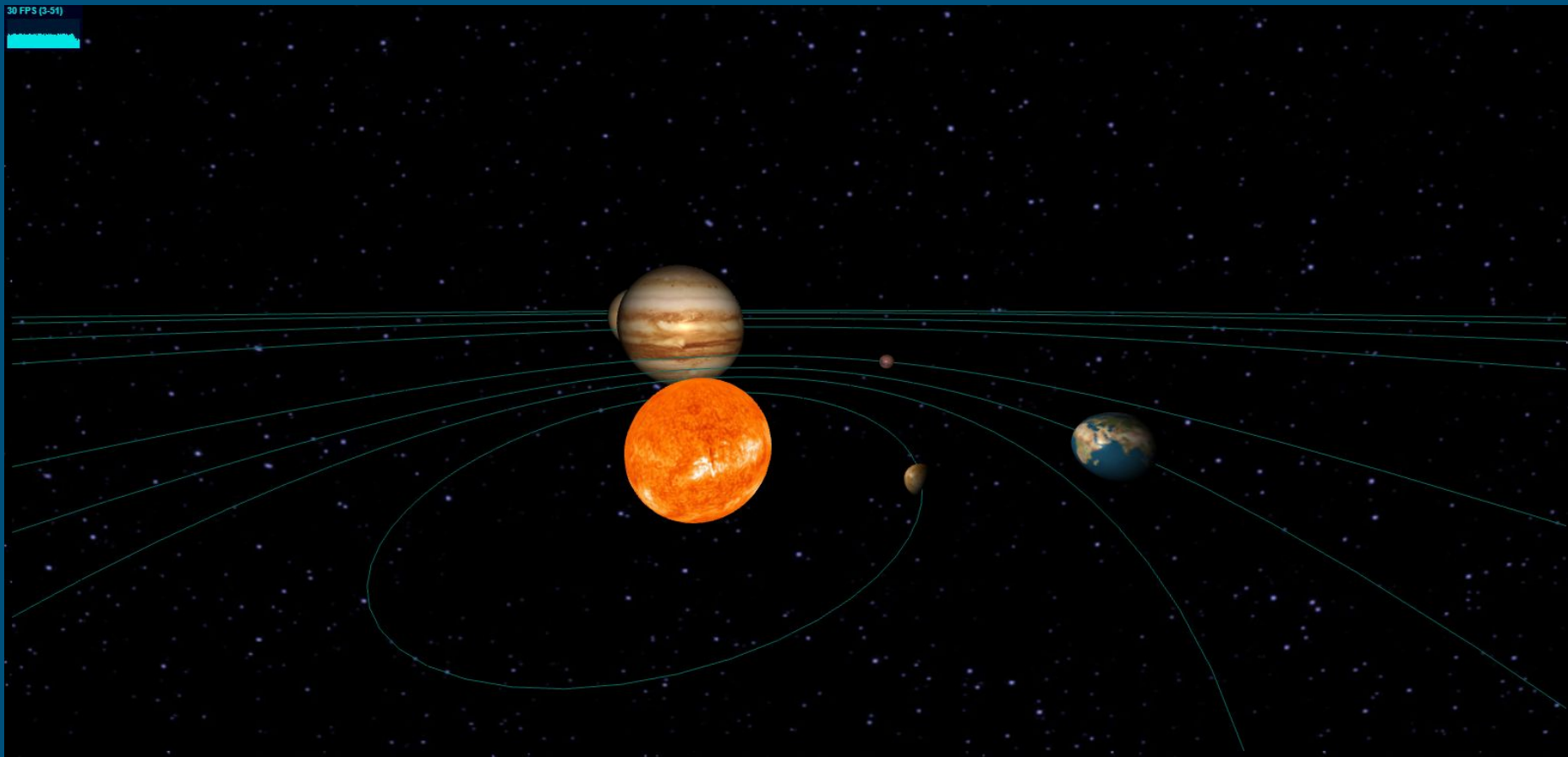
Kod aplikacji (SolarSystem.js)

animacja planet (obróć, rotacja)

```
const solarSystemMove = (planets) => {  
  solarSystemData.map(sphere => {  
    sphere.orbit += sphere.lineSpeed * 0.01;  
  
    planets[sphere.name].rotateY(sphere.rotate);  
    planets[sphere.name].position.x = Math.cos(sphere.orbit) * sphere.distance;  
    planets[sphere.name].position.z = Math.sin(sphere.orbit) * sphere.distance;  
  });  
};
```



30 FPS (2-51)



GitHub - Dyzio18/solar-system-webGL

Features Business Explore Marketplace Pricing This repository Search Sign in or Sign up

Dyzio18 / solar-system-webGL

Watch 0 Star 0 Fork 0

<> Code Issues 0 Pull requests 0 Projects 0 Insights

Join GitHub today
GitHub is home to over 20 million developers working together to host and review code, manage projects, and build software together.
Sign up

Solar System - WebGL, Three.js, JS, ES6 <https://dyzio18.github.io/solar-system-webGL/>

es6 solar-system webgl threejs javascript

16 commits 2 branches 0 releases 1 contributor

Branch: master New pull request Find file Clone or download

Dyzio18 display data about sphere Latest commit 03f41a9 12 minutes ago

dist	display data about sphere	12 minutes ago
img	upgrade texture & material, add mouse control	24 days ago
src	display data about sphere	12 minutes ago
.gitignore	add to gh-pages, first live	28 days ago
README.md	upgrade Readme	24 days ago
index.html	display data about sphere	12 minutes ago
package.json	initial commit	29 days ago
style.css	display data about sphere	12 minutes ago
webpack.config.js	initial commit	29 days ago
README.md		

<https://github.com/Dyzio18/solar-system-webGL>

Dziękuję za uwagę

Patryk Nizio

INFS / Rok III / grupa I

Kod:

<https://github.com/Dyzio18/solar-system-webGL>

LIVE:

<https://dyzio18.github.io/solar-system-webGL/>