ĆWICZENIE 4:

Time-to-market w systemach wbudowanych, czyli wykorzystanie gotowych komponentów oprogramowania

Node.js - systemy wbudowane i JavaScript?

"Node.js to wieloplatformowe środowisko uruchomieniowe JavaScript udostępnione na licencji open-source […]"

Node.js - systemy wbudowane i JavaScript?

"Node.js to wieloplatformowe środowisko uruchomieniowe JavaScript udostępnione na licencji open-source […]"

- dostępne w systemach *Linux*, *Mac OS X*, *Windows*, *Solaris*, *FreeBSD*, *OpenBSD*, … ,

Node.js - systemy wbudowane i JavaScript?

"Node.js to wieloplatformowe środowisko uruchomieniowe JavaScript udostępnione na licencji open-source […]"

- dostępne w systemach *Linux*, *Mac OS X*, *Windows*, *Solaris*, *FreeBSD*, *OpenBSD*, … ,
- umożliwia uruchomienie kodu JavaScript poza przeglądarką internetową,

Node.js - systemy wbudowane i JavaScript?

"Node.js to wieloplatformowe środowisko uruchomieniowe JavaScript udostępnione na licencji open-source […]"

- dostępne w systemach *Linux*, *Mac OS X*, *Windows*, *Solaris*, *FreeBSD*, *OpenBSD*, … ,
- umożliwia uruchomienie kodu JavaScript poza przeglądarką internetową,
- *Node.js* nie jest serwerem, umożliwia jednak proste i szybkie utworzenie serwera oraz złożonych aplikacji internetowych,

Node.js - systemy wbudowane i JavaScript?

"Node.js to wieloplatformowe środowisko uruchomieniowe JavaScript udostępnione na licencji open-source […]"

- dostępne w systemach *Linux*, *Mac OS X*, *Windows*, *Solaris*, *FreeBSD*, *OpenBSD*, … ,
- umożliwia uruchomienie kodu JavaScript poza przeglądarką internetową,
- *Node.js* nie jest serwerem, umożliwia jednak proste i szybkie utworzenie serwera oraz złożonych aplikacji internetowych,
- ponieważ kod programu jest uruchamiany poza przeglądarką, programista ma możliwość tworzenia typowych rozwiązań "*server-side*",

Node.js - instalacja środowiska

Dla dystrybucji *Debian*, instalacja pakietu `nodejs` przebiega w sposób standardowy dla narzędzia `apt-get`:

```
root@localhost:~# apt-get install nodejs
Selecting previously unselected package nodejs.
Preparing to unpack .../nodejs_4.8.2~dfsg-1_armhf.deb ...
Unpacking nodejs (4.8.2~dfsg-1) ...
```

Node.js - instalacja środowiska

Dla dystrybucji *Debian*, instalacja pakietu `nodejs` przebiega w sposób standardowy dla narzędzia `apt-get`:

```
root@localhost:~# apt-get install nodejs
Selecting previously unselected package nodejs.
Preparing to unpack .../nodejs_4.8.2~dfsg-1_armhf.deb ...
Unpacking nodejs (4.8.2~dfsg-1) ...
```

Aby przetestować poprawność instalacji, wywołajmy komendę `nodejs -v`:

```
root@localhost:~# nodejs -v
v4.8.2
```

ĆWICZENIE 4.2:

Prosta implementacja serwera WWW z wykorzystaniem Node.js

Node.js - prosta implementacja serwera WWW

Plik */root/linux-academy/4-2/main.js*:

```
var http = require ('http');

var PORT = 8080;

var server = http.createServer (function handler (request, response) {
  response.writeHead (200, {'Content-Type': 'text/plain'});
  response.end ('Hello World!');
});

server.listen (PORT);
```

Node.js - prosta implementacja serwera WWW

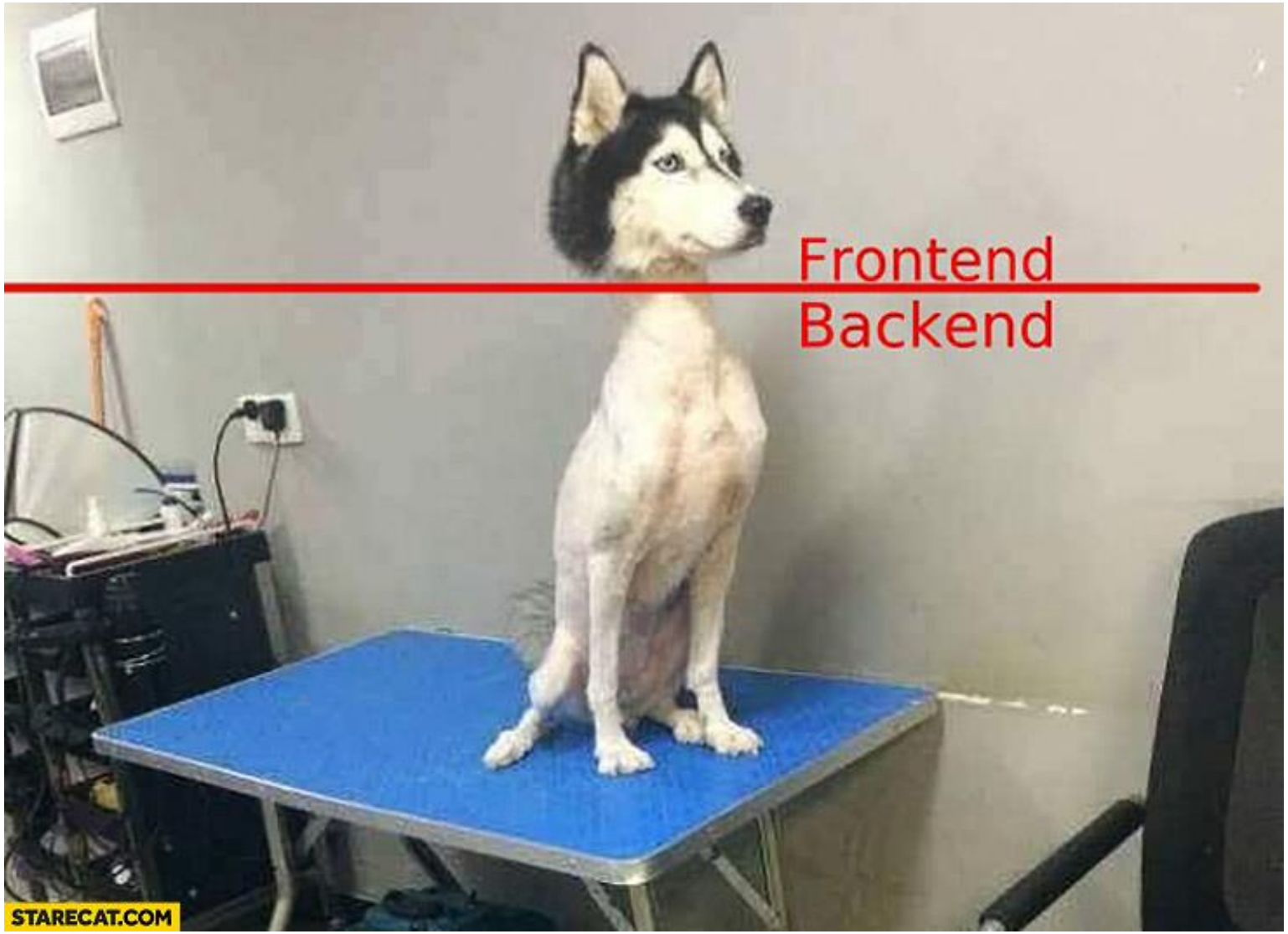
Plik */root/linux-academy/4-2/main.js*:

```
var http = require ('http');

var PORT = 8080;

var server = http.createServer (function handler (request, response) {
  response.writeHead (200, {'Content-Type': 'text/html'});
  response.end ('<!DOCTYPE html><html><head>
  <script src='/socket.io/socket.io.js'></script> <script> var socket = io();
  socket.on ('time', function (data) {document.getElementById("test").innerHTML
  = data.message; }); </script></head><body><h1>Hello World!</h1>
  <p id="test">JavaScript can change HTML content.</p></body></html>');
});

server.listen (PORT);
```

ĆWICZENIE 4.3:

Serwer WWW z podziałem na funkcje front-end oraz back-end

Serwer WWW – podział *front-end/back-end*



Serwer WWW – podział *front-end/back-end*

Plik */root/linux-academy/4-3/main.js*:

```
var http = require ('http');
var fs = require ('fs');

var index = fs.readFileSync (__dirname + '/index.html');

var PORT = 8080;

var server = http.createServer (function handler (request, response) {
  response.writeHead (200, {'Content-Type': 'text/html'});
  response.end (index);
});

server.listen (PORT);
```

Serwer WWW – podział *front-end/back-end*

Plik */root/linux-academy/4-3/index.html*:

```
<!DOCTYPE html>
<html>

  <head>
  </head>

  <body>
    <h1>Hello World!</h1>
  </body>

</html>
```

ĆWICZENIE 4.4:

Komunikacja front-end<->back-end z wykorzystaniem socket.io

Komunikacja z wykorzystaniem biblioteki *socket.io*

Domyślnym managerem pakietów dla środowiska *Node.js* jest NPM. Jest to aplikacja wiersza poleceń, za pomocą której można instalować aplikacje dostępne w repozytorium NPM. Strona domowa aplikacji zawiera wyszukiwarkę pakietów:

https://www.npmjs.com/

Aby zainstalować pakiet wykonujemy polecenie, np.:

```
npm install socket.io
```

W aplikacji importujemy moduł poprzez wywołanie:

```
var io = require ('socket.io');
```

Komunikacja z wykorzystaniem biblioteki *socket.io*

Fragment pliku */root/linux-academy/4-4/main.js*:

```
/* ... */

var io = require ('socket.io').listen(server);

io.on ('connection', function (socket) {
  console.log ('We have new connection!');
});

function send_time() {
  io.emit ('time', {message: new Date().toISOString()});
}

setInterval (send_time, 1000);

/* ... */
```

Komunikacja z wykorzystaniem biblioteki *socket.io*

Plik */root/linux-academy/4-4/index.html*:

```
<!DOCTYPE html>
<html>
  <head>
    <script src='/socket.io/socket.io.js'></script>
    <script>
      var socket = io();
      socket.on ('time', function (data) {
        document.getElementById("test").innerHTML = data.message;
      });
    </script>
  </head>
  <body>
    <h1>Hello World!</h1>
    <p id="test">JavaScript can change HTML content.</p>
  </body>
</html>
```

ĆWICZENIE 4.5:

Serwer WWW z odczytem danych z modułu żyroskopu

Odczyt danych z modułu żyroskopu - opcja 1

Bezpośrednia implementacja obsługi żyroskopu w kodzie serwera – z wykorzystaniem operacji na plikach lub gotowych modułów *Node.js*, instalowanych poprzez menadżer pakietów NPM. Przykładem takiego modułu może być pakiet `i2c`, instalowany poleceniem:

```
npm install i2c
```

który udostępnia proste API do realizacji niskopoziomowych operacji zapisu/odczytu danych na magistrali, np.:

```
var i2c = require('i2c');
var wire = new i2c(address, {device: '/dev/i2c-1'});
wire.writeByte(byte, function(err) {});
wire.writeBytes(command, [byte0, byte1], function(err) {});
wire.readByte(function(err, res) { // result is single byte })
wire.readBytes(command, length, function(err, res) {});
```

Odczyt danych z modułu żyroskopu - opcja 2

Implementacja obsługi procesu potomnego w pliku */root/linux-academy/4-5/main.js*:

```
var spawn = require('child_process').spawn;

var child = spawn ('/tmp/gyro-i2c');

child.stdout.on ('data', function (data) {
  io.emit ('xyz', {message: data.toString().split('\n')[0]});
});

child.stderr.on ('data', function (data) {
  console.log ('stderr: ' + data);
});

child.on ('close', function (code) {
  console.log ('exit: ' + code);
});
```

Odczyt danych z modułu żyroskopu - opcja 2

Implementacja obsługi procesu potomnego w pliku */root/linux-academy/4-5/main.js*:

```
var spawn = require('child_process').spawn;

var child = spawn ('/tmp/gyro-i2c');

child.stdout.on ('data', function (data) {
  io.emit ('xyz', {message: data.toString().split('\n')[0]});
});

child.stderr.on ('data', function (data) {
  console.log ('stderr: ' + data);
});

child.on ('close', function (code) {
  console.log ('exit: ' + code);
});
```

Odczyt danych z modułu żyroskopu - opcja 2

Implementacja obsługi procesu potomnego w pliku */root/linux-academy/4-5/main.js*:

```
var spawn = require('child_process').spawn;

var child = spawn ('/tmp/gyro-i2c');

child.stdout.on ('data', function (data) {
  io.emit ('xyz', {message: data.toString().split('\n')[0]});
});

child.stderr.on ('data', function (data) {
  console.log ('stderr: ' + data);
});

child.on ('close', function (code) {
  console.log ('exit: ' + code);
});
```

Odczyt danych z modułu żyroskopu - opcja 2

Implementacja obsługi procesu potomnego w pliku */root/linux-academy/4-5/main.js*:

```
var spawn = require('child_process').spawn;

var child = spawn ('/tmp/gyro-i2c');

child.stdout.on ('data', function (data) {
  io.emit ('xyz', {message: data.toString().split('\n')[0]});
});

child.stderr.on ('data', function (data) {
  console.log ('stderr: ' + data);
});

child.on ('close', function (code) {
  console.log ('exit: ' + code);
});
```

Odczyt danych z modułu żyroskopu - opcja 2

Implementacja obsługi procesu potomnego w pliku */root/linux-academy/4-5/main.js*:

```
var spawn = require('child_process').spawn;

var child = spawn ('/tmp/gyro-i2c');

child.stdout.on ('data', function (data) {
  io.emit ('xyz', {message: data.toString().split('\n')[0]});
});

child.stderr.on ('data', function (data) {
  console.log ('stderr: ' + data);
});

child.on ('close', function (code) {
  console.log ('exit: ' + code);
});
```

Odczyt danych z modułu żyroskopu - opcja 2

Tabela z sekcji `<body>` - plik */root/linux-academy/4-5/index.html*:

```
<table>
  <tr>
    <th>X [deg]</th>
    <td><p id="x_val">---</p></td>
  </tr>

  <tr>
    <th>Y [deg]</th>
    <td><p id="y_val">---</p></td>
  </tr>

  <tr>
    <th>Z [deg]</th>
    <td><p id="z_val">---</p></td>
  </tr>
</table>
```

Odczyt danych z modułu żyroskopu - opcja 2

Obsługa wiadomości `xyz` w sekcji `<head>` - plik */root/linux-academy/4-5/index.html*:

```
<script>

  var socket = io();

  socket.on ('xyz', function (data) {

    var arr = data.message.split(" ");

    document.getElementById("x_val").innerHTML = arr[0];
    document.getElementById("y_val").innerHTML = arr[1];
    document.getElementById("z_val").innerHTML = arr[2];
  });

</script>
```

ĆWICZENIE 4.6:

Rozbudowa interfejsu serwera WWW o elementy grafiki 3D (Three.js)

Interfejs użytkownika z elementami grafiki 3D

Gyroscope I2C

X [deg]	153.19
Y [deg]	125.43
Z [deg]	73.18

Interfejs użytkownika z elementami grafiki 3D

Gyroscope I2C

X [deg]	153.19
Y [deg]	125.43
Z [deg]	73.18

- **API WebGL** - oficjalne rozszerzenie języka HTML o interfejs grafiki 3D

Interfejs użytkownika z elementami grafiki 3D

Gyroscope I2C

X [deg]	153.19
Y [deg]	125.43
Z [deg]	73.18

- **API WebGL** - oficjalne rozszerzenie języka HTML o interfejs grafiki 3D

- **Biblioteka Three.js** - wysokopoziomowe API dla WebGL

```
(wget http://threejs.org/build/three.min.js)
```

Interfejs użytkownika z elementami grafiki 3D

- **Inicjalizacja *WebGL***
- Inicjalizacja buforów danych
- Rysowanie sceny
- *Shader* - inicjalizacja
- *Shader* - funkcje pomocnicze
- *Shader* - kod GLSC

```
<body>
  <button class="runButton" onclick="webGLStart();">Start</button>
  <canvas id="webgl_canvas" width="500" height="500"></canvas>
</body>

//snip

function webGLStart()
{
  var canvas = document.getElementById("webgl_canvas");

  initGL(canvas);
  initShaders();
  initBuffers();

  gl.clearColor(0.0, 0.0, 0.0, 1.0);
  gl.enable(gl.DEPTH_TEST);

  drawScene();
}
```

Interfejs użytkownika z elementami grafiki 3D

- Inicjalizacja *WebGL*
- **Inicjalizacja buforów danych**
- Rysowanie sceny
- *Shader* - inicjalizacja
- *Shader* - funkcje pomocnicze
- *Shader* - kod GLSC

```
var squareVertexPositionBuffer;


function initBuffers() {


  squareVertexPositionBuffer = gl.createBuffer();
  gl.bindBuffer(gl.ARRAY_BUFFER, squareVertexPositionBuffer);


  vertices = [
     1.0,  1.0,  0.0,
    -1.0,  1.0,  0.0,
     1.0, -1.0,  0.0,
    -1.0, -1.0,  0.0,
  ];

  gl.bufferData(gl.ARRAY_BUFFER,
                new Float32Array(vertices),
                gl.STATIC_DRAW);


  squareVertexPositionBuffer.itemSize = 3;
  squareVertexPositionBuffer.numItems = 4;
}
```

Interfejs użytkownika z elementami grafiki 3D

- Inicjalizacja *WebGL*
- Inicjalizacja buforów danych
- **Rysowanie sceny**
- *Shader* - inicjalizacja
- *Shader* - funkcje pomocnicze
- *Shader* - kod GLSC

```
function drawScene()
{
  gl.viewport(0, 0, gl.viewportWidth, gl.viewportHeight);
  gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);

  mat4.perspective(45, gl.viewportWidth / gl.viewportHeight,
                   0.1, 100.0, pMatrix);

  mat4.identity(mvMatrix);

  mat4.translate(mvMatrix, [3.0, 0.0, 0.0]);
  gl.bindBuffer(gl.ARRAY_BUFFER, squareVertexPositionBuffer);
  gl.vertexAttribPointer(shaderProgram.vertexPositionAttribute,
                         squareVertexPositionBuffer.itemSize,
                         gl.FLOAT, false, 0, 0);
  setMatrixUniforms();
  gl.drawArrays(gl.TRIANGLE_STRIP, 0,
                squareVertexPositionBuffer.numItems);
}
```

Interfejs użytkownika z elementami grafiki 3D

- Inicjalizacja *WebGL*
- Inicjalizacja buforów danych
- Rysowanie sceny
- ***Shader* - inicjalizacja**
- *Shader* - funkcje pomocnicze
- *Shader* - kod GLSC

```
function initShaders()
{
  var fragmentShader = getShader(gl, "shader-fs");
  var vertexShader = getShader(gl, "shader-vs");

  shaderProgram = gl.createProgram();
  gl.attachShader(shaderProgram, vertexShader);
  gl.attachShader(shaderProgram, fragmentShader);
  gl.linkProgram(shaderProgram);

  if (!gl.getProgramParameter(shaderProgram, gl.LINK_STATUS)) {
    alert("Could not initialise shaders");
  }

  gl.useProgram(shaderProgram);

  shaderProgram.vertexPositionAttribute = gl.getAttribLocation(shaderProgram,
                                                               "aVertexPosition");
  gl.enableVertexAttribArray(shaderProgram.vertexPositionAttribute);

  shaderProgram.pMatrixUniform = gl.getUniformLocation(shaderProgram, "uPMatrix");
  shaderProgram.mvMatrixUniform = gl.getUniformLocation(shaderProgram, "uMVMatrix");
}
```

Interfejs użytkownika z elementami grafiki 3D

- Inicjalizacja *WebGL*
- Inicjalizacja buforów danych
- Rysowanie sceny
- *Shader* - inicjalizacja
- ***Shader* - funkcje pomocnicze**
- *Shader* - kod GLSC

```
function getShader(gl, id)
{
  var shaderScript = document.getElementById(id);
  if (!shaderScript)
    return null;


  var str = "";
  var k = shaderScript.firstChild;
  while (k) {
    if (k.nodeType == 3)
      str += k.textContent;
    k = k.nextSibling;
  }
  var shader;
  if (shaderScript.type == "x-shader/x-fragment") {
    shader = gl.createShader(gl.FRAGMENT_SHADER);
  } else if (shaderScript.type == "x-shader/x-vertex") {
    shader = gl.createShader(gl.VERTEX_SHADER);
  } else {
    return null;
  }
  gl.shaderSource(shader, str);
  gl.compileShader(shader);
  if (!gl.getShaderParameter(shader, gl.COMPILE_STATUS)) {
    alert(gl.getShaderInfoLog(shader));
    return null;
  }
  return shader;
}
```

Interfejs użytkownika z elementami grafiki 3D

- Inicjalizacja *WebGL*
- Inicjalizacja buforów danych
- Rysowanie sceny
- *Shader* - inicjalizacja
- *Shader* - funkcje pomocnicze
- ***Shader* - kod GLSC**

```
<script id="shader-fs" type="x-shader/x-fragment">

  precision mediump float;

  void main(void) {
    gl_FragColor = vec4(1.0, 1.0, 1.0, 1.0);
  }

</script>

<script id="shader-vs" type="x-shader/x-vertex">

  attribute vec3 aVertexPosition;

  uniform mat4 uMVMatrix;
  uniform mat4 uPMatrix;

  void main(void) {
    gl_Position = uPMatrix * uMVMatrix * vec4(aVertexPosition, 1.0);
  }

</script>
```

Interfejs użytkownika z elementami grafiki 3D

three.js

- Definicja "płótna" w sekcji HEAD
- Dołączenie biblioteki Three.js
- Definicja zmiennych
- Utworzenie obiektu "sceny"
- Utworzenie obiektu "kamery"
- Utworzenie obiektu geometrycznego
- Utworzenie obiektu "materiału"
- Połączenie figury z materiałem
- Określenie obszaru renderowania

```
<canvas id="mycanvas" width="500" height="500"></canvas>

<script src='three.min.js'></script>

//snip

var camera, scene, renderer;
var geometry, material, mesh;
var x, y, z;

function init() {

  scene = new THREE.Scene();

  camera = new THREE.PerspectiveCamera (70, 500/500, 0.01, 10);
  camera.position.z = 0.5;

  geometry = new THREE.BoxGeometry (0.2, 0.2, 0.2);
  material = new THREE.MeshNormalMaterial();

  mesh = new THREE.Mesh (geometry, material);
  scene.add (mesh);

  renderer = new THREE.WebGLRenderer ({ canvas: mycanvas});
  renderer.setSize (500, 500);
  document.body.appendChild (renderer.domElement);
}
```

Interfejs użytkownika z elementami grafiki 3D

three.js

- **Definicja "płótna" w sekcji HEAD**
- Dołączenie biblioteki Three.js
- Definicja zmiennych
- Utworzenie obiektu "sceny"
- Utworzenie obiektu "kamery"
- Utworzenie obiektu geometrycznego
- Utworzenie obiektu "materiału"
- Połączenie figury z materiałem
- Określenie obszaru renderowania

```
<canvas id="mycanvas" width="500" height="500"></canvas>

<script src='three.min.js'></script>

//snip

var camera, scene, renderer;
var geometry, material, mesh;
var x, y, z;

function init() {

  scene = new THREE.Scene();

  camera = new THREE.PerspectiveCamera (70, 500/500, 0.01, 10);
  camera.position.z = 0.5;

  geometry = new THREE.BoxGeometry (0.2, 0.2, 0.2);
  material = new THREE.MeshNormalMaterial();

  mesh = new THREE.Mesh (geometry, material);
  scene.add (mesh);

  renderer = new THREE.WebGLRenderer ({ canvas: mycanvas});
  renderer.setSize (500, 500);
  document.body.appendChild (renderer.domElement);
}
```

Interfejs użytkownika z elementami grafiki 3D

three.js

- Definicja "płótna" w sekcji HEAD
- **Dołączenie biblioteki Three.js**
- Definicja zmiennych
- Utworzenie obiektu "sceny"
- Utworzenie obiektu "kamery"
- Utworzenie obiektu geometrycznego
- Utworzenie obiektu "materiału"
- Połączenie figury z materiałem
- Określenie obszaru renderowania

```
<canvas id="mycanvas" width="500" height="500"></canvas>

<script src='three.min.js'></script>

//snip

var camera, scene, renderer;
var geometry, material, mesh;
var x, y, z;

function init() {

  scene = new THREE.Scene();

  camera = new THREE.PerspectiveCamera (70, 500/500, 0.01, 10);
  camera.position.z = 0.5;

  geometry = new THREE.BoxGeometry (0.2, 0.2, 0.2);
  material = new THREE.MeshNormalMaterial();

  mesh = new THREE.Mesh (geometry, material);
  scene.add (mesh);

  renderer = new THREE.WebGLRenderer ({ canvas: mycanvas});
  renderer.setSize (500, 500);
  document.body.appendChild (renderer.domElement);
}
```

Interfejs użytkownika z elementami grafiki 3D

three.js

- Definicja "płótna" w sekcji HEAD
- Dołączenie biblioteki Three.js
- **Definicja zmiennych**
- Utworzenie obiektu "sceny"
- Utworzenie obiektu "kamery"
- Utworzenie obiektu geometrycznego
- Utworzenie obiektu "materiału"
- Połączenie figury z materiałem
- Określenie obszaru renderowania

```
<canvas id="mycanvas" width="500" height="500"></canvas>

<script src='three.min.js'></script>

//snip

var camera, scene, renderer;
var geometry, material, mesh;
var x, y, z;

function init() {

  scene = new THREE.Scene();

  camera = new THREE.PerspectiveCamera (70, 500/500, 0.01, 10);
  camera.position.z = 0.5;

  geometry = new THREE.BoxGeometry (0.2, 0.2, 0.2);
  material = new THREE.MeshNormalMaterial();

  mesh = new THREE.Mesh (geometry, material);
  scene.add (mesh);

  renderer = new THREE.WebGLRenderer ({ canvas: mycanvas});
  renderer.setSize (500, 500);
  document.body.appendChild (renderer.domElement);
}
```

Interfejs użytkownika z elementami grafiki 3D

three.js

- Definicja "płótna" w sekcji HEAD
- Dołączenie biblioteki Three.js
- Definicja zmiennych
- **Utworzenie obiektu "sceny"**
- Utworzenie obiektu "kamery"
- Utworzenie obiektu geometrycznego
- Utworzenie obiektu "materiału"
- Połączenie figury z materiałem
- Określenie obszaru renderowania

```
<canvas id="mycanvas" width="500" height="500"></canvas>

<script src='three.min.js'></script>

//snip

var camera, scene, renderer;
var geometry, material, mesh;
var x, y, z;

function init() {

  scene = new THREE.Scene();

  camera = new THREE.PerspectiveCamera (70, 500/500, 0.01, 10);
  camera.position.z = 0.5;

  geometry = new THREE.BoxGeometry (0.2, 0.2, 0.2);
  material = new THREE.MeshNormalMaterial();

  mesh = new THREE.Mesh (geometry, material);
  scene.add (mesh);

  renderer = new THREE.WebGLRenderer ({ canvas: mycanvas});
  renderer.setSize (500, 500);
  document.body.appendChild (renderer.domElement);
}
```

Interfejs użytkownika z elementami grafiki 3D

three.js

- Definicja "płótna" w sekcji HEAD
- Dołączenie biblioteki Three.js
- Definicja zmiennych
- Utworzenie obiektu "sceny"
- **Utworzenie obiektu "kamery"**
- Utworzenie obiektu geometrycznego
- Utworzenie obiektu "materiału"
- Połączenie figury z materiałem
- Określenie obszaru renderowania

```
<canvas id="mycanvas" width="500" height="500"></canvas>

<script src='three.min.js'></script>

//snip

var camera, scene, renderer;
var geometry, material, mesh;
var x, y, z;

function init() {

  scene = new THREE.Scene();

  camera = new THREE.PerspectiveCamera (70, 500/500, 0.01, 10);
  camera.position.z = 0.5;

  geometry = new THREE.BoxGeometry (0.2, 0.2, 0.2);
  material = new THREE.MeshNormalMaterial();

  mesh = new THREE.Mesh (geometry, material);
  scene.add (mesh);

  renderer = new THREE.WebGLRenderer ({ canvas: mycanvas});
  renderer.setSize (500, 500);
  document.body.appendChild (renderer.domElement);
}
```

Interfejs użytkownika z elementami grafiki 3D

three.js

- Definicja "płótna" w sekcji HEAD
- Dołączenie biblioteki Three.js
- Definicja zmiennych
- Utworzenie obiektu "sceny"
- Utworzenie obiektu "kamery"
- **Utworzenie obiektu geometrycznego**
- Utworzenie obiektu "materiału"
- Połączenie figury z materiałem
- Określenie obszaru renderowania

```
<canvas id="mycanvas" width="500" height="500"></canvas>

<script src='three.min.js'></script>

//snip

var camera, scene, renderer;
var geometry, material, mesh;
var x, y, z;

function init() {

  scene = new THREE.Scene();

  camera = new THREE.PerspectiveCamera (70, 500/500, 0.01, 10);
  camera.position.z = 0.5;

  geometry = new THREE.BoxGeometry (0.2, 0.2, 0.2);
  material = new THREE.MeshNormalMaterial();

  mesh = new THREE.Mesh (geometry, material);
  scene.add (mesh);

  renderer = new THREE.WebGLRenderer ({ canvas: mycanvas});
  renderer.setSize (500, 500);
  document.body.appendChild (renderer.domElement);
}
```

Interfejs użytkownika z elementami grafiki 3D

three.js

- Definicja "płótna" w sekcji HEAD
- Dołączenie biblioteki Three.js
- Definicja zmiennych
- Utworzenie obiektu "sceny"
- Utworzenie obiektu "kamery"
- Utworzenie obiektu geometrycznego
- **Utworzenie obiektu "materiału"**
- Połączenie figury z materiałem
- Określenie obszaru renderowania

```
<canvas id="mycanvas" width="500" height="500"></canvas>

<script src='three.min.js'></script>

//snip

var camera, scene, renderer;
var geometry, material, mesh;
var x, y, z;

function init() {

  scene = new THREE.Scene();

  camera = new THREE.PerspectiveCamera (70, 500/500, 0.01, 10);
  camera.position.z = 0.5;

  geometry = new THREE.BoxGeometry (0.2, 0.2, 0.2);
  material = new THREE.MeshNormalMaterial();

  mesh = new THREE.Mesh (geometry, material);
  scene.add (mesh);

  renderer = new THREE.WebGLRenderer ({ canvas: mycanvas});
  renderer.setSize (500, 500);
  document.body.appendChild (renderer.domElement);
}
```

Interfejs użytkownika z elementami grafiki 3D

three.js

- Definicja "płótna" w sekcji HEAD
- Dołączenie biblioteki Three.js
- Definicja zmiennych
- Utworzenie obiektu "sceny"
- Utworzenie obiektu "kamery"
- Utworzenie obiektu geometrycznego
- Utworzenie obiektu "materiału"
- **Połączenie figury z materiałem**
- Określenie obszaru renderowania

```
<canvas id="mycanvas" width="500" height="500"></canvas>

<script src='three.min.js'></script>

//snip

var camera, scene, renderer;
var geometry, material, mesh;
var x, y, z;

function init() {

  scene = new THREE.Scene();

  camera = new THREE.PerspectiveCamera (70, 500/500, 0.01, 10);
  camera.position.z = 0.5;

  geometry = new THREE.BoxGeometry (0.2, 0.2, 0.2);
  material = new THREE.MeshNormalMaterial();

  mesh = new THREE.Mesh (geometry, material);
  scene.add (mesh);

  renderer = new THREE.WebGLRenderer ({ canvas: mycanvas});
  renderer.setSize (500, 500);
  document.body.appendChild (renderer.domElement);
}
```

Interfejs użytkownika z elementami grafiki 3D

three.js

- Definicja "płótna" w sekcji HEAD
- Dołączenie biblioteki Three.js
- Definicja zmiennych
- Utworzenie obiektu "sceny"
- Utworzenie obiektu "kamery"
- Utworzenie obiektu geometrycznego
- Utworzenie obiektu "materiału"
- Połączenie figury z materiałem
- **Określenie obszaru renderowania**

```
<canvas id="mycanvas" width="500" height="500"></canvas>

<script src='three.min.js'></script>

//snip

var camera, scene, renderer;
var geometry, material, mesh;
var x, y, z;

function init() {

  scene = new THREE.Scene();

  camera = new THREE.PerspectiveCamera (70, 500/500, 0.01, 10);
  camera.position.z = 0.5;

  geometry = new THREE.BoxGeometry (0.2, 0.2, 0.2);
  material = new THREE.MeshNormalMaterial();

  mesh = new THREE.Mesh (geometry, material);
  scene.add (mesh);

  renderer = new THREE.WebGLRenderer ({ canvas: mycanvas});
  renderer.setSize (500, 500);
  document.body.appendChild (renderer.domElement);
}
```

Interfejs użytkownika z elementami grafiki 3D

Utwórzmy również funkcję `animate()`, która dokona obrotu obiektu, zgodnie z kątem obrotu zapisanym w zmiennych x, y, z:

```
function animate() {

  requestAnimationFrame (animate);

  mesh.rotation.x = THREE.Math.degToRad(x);
  mesh.rotation.y = THREE.Math.degToRad(y);
  mesh.rotation.z = THREE.Math.degToRad(z);

  renderer.render (scene, camera);
}
```

Interfejs użytkownika z elementami grafiki 3D

Niewielkiej modyfikacji wymaga również sam kod serwera z pliku */root/linux-academy/4-6/main.js*:

```
var url = require('url');
var server = http.createServer (function handler (request, response) {

  var pathname = url.parse(request.url).pathname;
  console.log("Request for " + pathname + " received.");

  response.writeHead (200, {'Content-Type': 'text/html'});
  if(pathname == "/") {
    var index = fs.readFileSync (__dirname + '/index.html');
    response.write (index);
  } else if (pathname == "/three.min.js") {
    var script = fs.readFileSync (__dirname + '/three.min.js');
    response.write (script);
  }
  response.end();
});
```

Dziękuję za uwagę