Mateusz Nawrat

241270

Śr. 11-14 TP

WebGL

1. Kod źródłowy programu

Plik style.css oraz matrix.js nie uległ zmianie w stosunku do przykładowego programu, więc nie potrzebny jest listing z ich kodu.

*Index.html:*



*Main.js:*

// zmienne globalne

var gl\_canvas;

var gl\_ctx;

var \_uv;

var \_sampler;

var \_triangleVertexBuffer;

var \_triangleFacesBuffer;

var \_position;

var \_color;

var \_PosMatrix;

var \_MovMatrix;

var \_ViewMatrix;

var \_matrixProjection;

var \_matrixMovement;

var \_matrixView;

var X, Y, Z;

var tex1, tex2, tex3;

var rotationSpeed = 0.001;

var zoomRatio = -6;

var \_cubeTexture;

var isAnimating = false;

// funkcja główna

function runWebGL () {

getRotation();

getTexture();

gl\_canvas = document.getElementById("glcanvas");

gl\_ctx = gl\_getContext(gl\_canvas);

gl\_initShaders();

gl\_initBuffers();

gl\_setMatrix();

\_cubeTexture = gl\_initTexture();

gl\_draw();

}

//tekstura

function getTexture() {

tex1 = document.getElementById('tekstura1').checked;

tex2 = document.getElementById('tekstura2').checked;

tex3 = document.getElementById('tekstura3').checked;

}

// osie obrotu

function getRotation() {

X = document.getElementById('rotateX').checked;

Y = document.getElementById('rotateY').checked;

Z = document.getElementById('rotateZ').checked;

}

// pobranie kontekstu WebGL

function gl\_getContext (canvas) {

try {

var ctx = canvas.getContext("webgl");

ctx.viewportWidth = canvas.width;

ctx.viewportHeight = canvas.height;

} catch (e) {}

if (!ctx) {

document.write('Unable to initialize WebGL context.')

}

return ctx;

}

// shadery

function gl\_initShaders () {

var vertexShader = "\n\

attribute vec3 position;\n\

uniform mat4 PosMatrix;\n\

uniform mat4 MovMatrix;\n\

uniform mat4 ViewMatrix; \n\

attribute vec2 uv;\n\

varying vec2 vUV;\n\

void main(void) {\n\

gl\_Position = PosMatrix \* ViewMatrix \* MovMatrix \* vec4(position, 1.);\n\

vUV = uv;\n\

}";

var fragmentShader = "\n\

precision mediump float;\n\

uniform sampler2D sampler;\n\

varying vec2 vUV;\n\

void main(void) {\n\

gl\_FragColor = texture2D(sampler, vUV);\n\

}";

var getShader = function(source, type, typeString) {

var shader = gl\_ctx.createShader(type);

gl\_ctx.shaderSource(shader, source);

gl\_ctx.compileShader(shader);

if (!gl\_ctx.getShaderParameter(shader, gl\_ctx.COMPILE\_STATUS)) {

alert('error in' + typeString);

return false;

}

return shader;

};

var shader\_vertex = getShader(vertexShader, gl\_ctx.VERTEX\_SHADER, "VERTEX");

var shader\_fragment = getShader(fragmentShader, gl\_ctx.FRAGMENT\_SHADER, "FRAGMENT");

var shaderProgram = gl\_ctx.createProgram();

gl\_ctx.attachShader(shaderProgram, shader\_vertex);

gl\_ctx.attachShader(shaderProgram, shader\_fragment);

gl\_ctx.linkProgram(shaderProgram);

\_PosMatrix = gl\_ctx.getUniformLocation(shaderProgram, "PosMatrix");

\_MovMatrix = gl\_ctx.getUniformLocation(shaderProgram, "MovMatrix");

\_ViewMatrix = gl\_ctx.getUniformLocation(shaderProgram, "ViewMatrix");

\_sampler = gl\_ctx.getUniformLocation(shaderProgram, "sampler");

\_uv = gl\_ctx.getAttribLocation(shaderProgram, "uv");

\_position = gl\_ctx.getAttribLocation(shaderProgram, "position");

gl\_ctx.enableVertexAttribArray(\_uv);

gl\_ctx.enableVertexAttribArray(\_position);

gl\_ctx.useProgram(shaderProgram); }

gl\_ctx.uniform1i(\_sampler, 0);

// Bufory

function gl\_initBuffers () {

var triangleVertices = [

0, 1, 0, 0.5, 1, //1 sciana 🡨--------------------------------- 1

1, 0, -1, 1, 0,

0, 0, 1, 0, 0,

0, 1, 0, 0.5, 1, //2 sciana

1, 0, -1, 1, 0,

-1, 0, -1, 0, 0,

0, 1, 0, 0.5, 1, //3 sciana

-1, 0, -1, 1, 0,

0, 0, 1, 0, 0,

0, 0, 1, 0, 0, //4 sciana

-1, 0, -1, 0, 0,

1, 0, -1, 1, 0

];

\_triangleVertexBuffer = gl\_ctx.createBuffer();

gl\_ctx.bindBuffer(gl\_ctx.ARRAY\_BUFFER, \_triangleVertexBuffer);

gl\_ctx.bufferData(gl\_ctx.ARRAY\_BUFFER,

new Float32Array(triangleVertices),

gl\_ctx.STATIC\_DRAW);

var triangleFaces = [ 🡨--------------------------------- 2

0, 1, 2,

3, 4, 5,

6, 7, 8,

9, 10, 11];

\_triangleFacesBuffer = gl\_ctx.createBuffer();

gl\_ctx.bindBuffer(gl\_ctx.ELEMENT\_ARRAY\_BUFFER, \_triangleFacesBuffer);

gl\_ctx.bufferData(gl\_ctx.ELEMENT\_ARRAY\_BUFFER, new Uint16Array(triangleFaces), gl\_ctx.STATIC\_DRAW);

}

// Macierz

function gl\_setMatrix () {

\_matrixProjection = MATRIX.getProjection(40, gl\_canvas.width/gl\_canvas.height, 1, 100);

\_matrixMovement = MATRIX.getIdentityMatrix();

\_matrixView = MATRIX.getIdentityMatrix();

MATRIX.translateZ(\_matrixView, zoomRatio);

}

//tekstura

function gl\_initTexture() {

var img = new Image();

if(tex1)

img.src = 'cubeTexture.png';

if(tex2)

img.src = 'cubeTexture2.png';

if(tex3)

img.src = 'cubeTexture3.png';

img.onload = function(e) {

var texture = gl\_ctx.createTexture();

gl\_ctx.pixelStorei(gl\_ctx.UNPACK\_FLIP\_Y\_WEBGL, true);

gl\_ctx.bindTexture(gl\_ctx.TEXTURE\_2D, texture);

gl\_ctx.texParameteri(gl\_ctx.TEXTURE\_2D, gl\_ctx.TEXTURE\_MIN\_FILTER, gl\_ctx.LINEAR);

gl\_ctx.texParameteri(gl\_ctx.TEXTURE\_2D, gl\_ctx.TEXTURE\_MAG\_FILTER, gl\_ctx.LINEAR);

gl\_ctx.texImage2D(gl\_ctx.TEXTURE\_2D, 0, gl\_ctx.RGBA, gl\_ctx.RGBA, gl\_ctx.UNSIGNED\_BYTE, img);

gl\_ctx.bindTexture(gl\_ctx.TEXTURE\_2D, null);

img.webglTexture = texture;

};

return img;

}

// Rysowanie

function gl\_draw() {

gl\_ctx.clearColor(0.0, 0.0, 0.0, 0.0);

gl\_ctx.enable(gl\_ctx.DEPTH\_TEST);

gl\_ctx.depthFunc(gl\_ctx.LEQUAL);

gl\_ctx.clearDepth(1.0);

var timeOld = 0;

if (!isAnimating) { 🡨--------------------------------- 5

var animate = function (time) {

var dAngle = rotationSpeed \* (time - timeOld);

if (X)

MATRIX.rotateX(\_matrixMovement, dAngle);

if (Y)

MATRIX.rotateY(\_matrixMovement, dAngle);

if (Z)

MATRIX.rotateZ(\_matrixMovement, dAngle);

timeOld = time;

gl\_ctx.viewport(0.0, 0.0, gl\_canvas.width, gl\_canvas.height);

gl\_ctx.clear(gl\_ctx.COLOR\_BUFFER\_BIT | gl\_ctx.DEPTH\_BUFFER\_BIT);

gl\_ctx.uniformMatrix4fv(\_PosMatrix, false, \_matrixProjection);

gl\_ctx.uniformMatrix4fv(\_MovMatrix, false, \_matrixMovement);

gl\_ctx.uniformMatrix4fv(\_ViewMatrix, false, \_matrixView);

if (\_cubeTexture.webglTexture) {

gl\_ctx.activeTexture(gl\_ctx.TEXTURE0);

gl\_ctx.bindTexture(gl\_ctx.TEXTURE\_2D, \_cubeTexture.webglTexture);

}

gl\_ctx.vertexAttribPointer(\_position, 3, gl\_ctx.FLOAT, false, 4\*(3+2), 0); 🡨--------------------------------- 3

gl\_ctx.vertexAttribPointer(\_uv, 2, gl\_ctx.FLOAT, false, 4\*(3+2), 3\*4);

gl\_ctx.bindBuffer(gl\_ctx.ARRAY\_BUFFER, \_triangleVertexBuffer);

gl\_ctx.bindBuffer(gl\_ctx.ELEMENT\_ARRAY\_BUFFER, \_triangleFacesBuffer);

gl\_ctx.drawElements(gl\_ctx.TRIANGLES, 4 \* 3, gl\_ctx.UNSIGNED\_SHORT, 0); 🡨--------------------------------- 4

gl\_ctx.flush();

window.requestAnimationFrame(animate);

}

animate(0);

isAnimating = true; 🡨--------------------------------- 6

}

}

1. Realizacja zadań

W kodzie zaznaczono strzałkami oznaczonymi numerami niezbędne modyfikacje w kodzie, odpowiadające za realizację zadań odpowiednio 1, 2, 3 oraz 4 to zadanie pierwsze, natomiast linie oznaczone strzałkami 5 i 6 odpowiadają za realizację zadania drugiego.

W oznaczeniu 1 wskazana jest tablica ścian ostrosłupa, gdzie każda grupa 3 punktów odpowiada jednej ścianie a 2 dodatkowe współrzędne odpowiadają za skalowanie tekstury.

Wskazanie drugie odnosi się do trójkątów rysowanych przez funkcję, gdzie elementy wskazują na wierzchołki z pkt. 1.

Element wskazany numerem 3 pozostał bez zmian.

W pkt 4 należało zmienić ilość elementów tablicy trianglefaces na 12.

Pkt 5 i 6 odpowiadają za dodanie warunku odpowiedzialnego za stałą prędkość animacja, odpowiadają one za brak powtarzania rysowania animacji.

Do pliku index.html zostały dodane miniaturki tekstur oraz checkboxy.

1. Efekt działania programu

