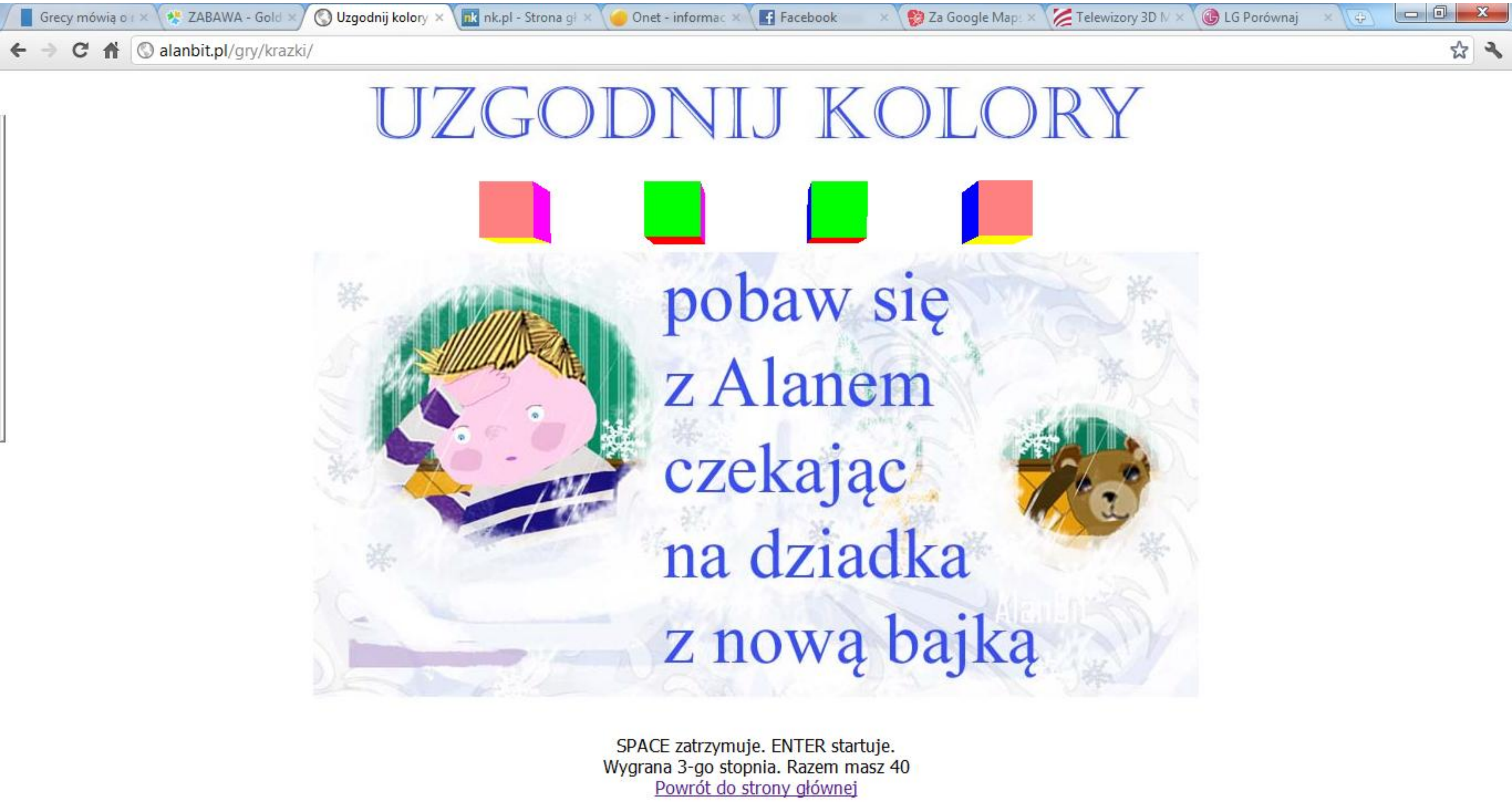
Renderowanie czystych kolorów na przykładzie jednorękiego bandyty

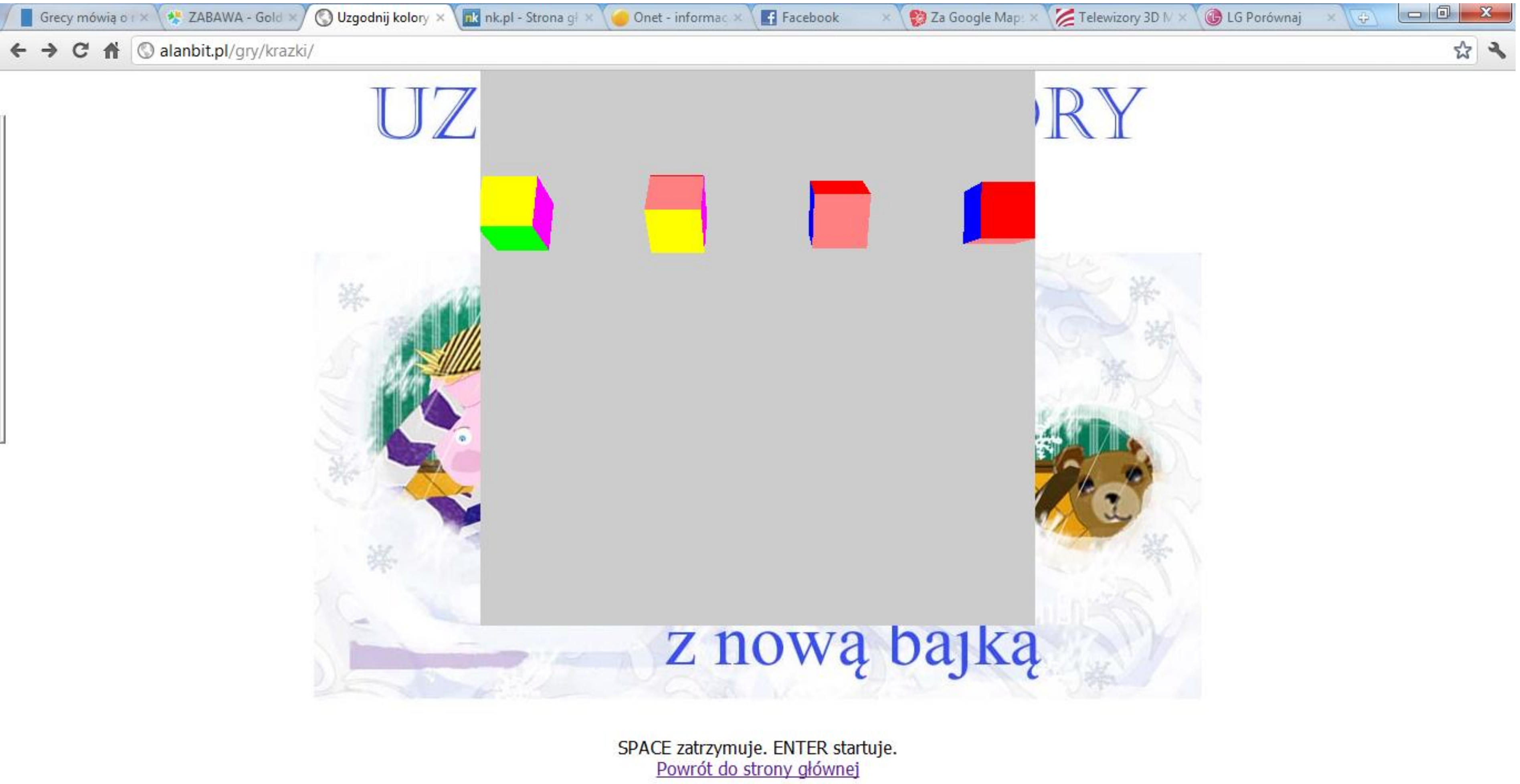
A.P.Urbański

Kolorowy jednoręki bandyta



SPACE zatrzymuje. ENTER startuje.
Wygrana 3-go stopnia. Razem masz 40
Powrót do strony głównej

Miejsce webGL w prezentacji



SPACE zatrzymuje. ENTER startuje.
Powrót do strony głównej

Segment główny

```
</head>
<body onload="webGLStart();">
		<div id="game">
	<canvas id="lesson06-canvas" style="border: none;"
	width="500" height="500"></canvas>
</div>
<center><div>SPACE zatrzymuje. ENTER startuje.</div><div
	id='alert'></div>
<a href="http://alanbit.pl">Powrót do strony
	głównej</a></center>
</body>
</html>
```

Inicjowanie webGL

```
function webGLStart() {
        var canvas = document.getElementById("lesson06-canvas");
        initGL(canvas);
        initShaders();
        initBuffers();

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

        document.onkeydown = handleKeyDown;
        document.onkeyup = handleKeyUp;

        tick();
    }

</script>
```

Inicjowanie GL

```
<script type="text/javascript">

  var gl;

  function initGL(canvas) {
    try {
      gl = canvas.getContext("experimental-webgl");
      gl.viewportWidth = canvas.width;
      gl.viewportHeight = canvas.height;
    } catch (e) {
    }
    if (!gl) {
      alert("Could not initialise WebGL, sorry :-(");
    }
  }
```

Biblioteki i shader fragmentów

```
<script type="text/javascript" src="glMatrix-0.9.5.min.js"></script>
<script type="text/javascript" src="webgl-utils.js"></script>
<script id="shader-fs" type="x-shader/x-fragment">
    #ifdef GL_ES
    precision highp float;
    #endif

    varying vec4 vColor;

    void main(void) {
        gl_FragColor = vColor;
    }
</script>
```

Shader wierzchołków

```
<script id="shader-vs" type="x-shader/x-vertex">
  attribute vec3 aVertexPosition;
  attribute vec4 aVertexColor;

  uniform mat4 uMVMatrix;
  uniform mat4 uPMatrix;

  varying vec4 vColor;

  void main(void) {
    gl_Position = uPMatrix * uMVMatrix * vec4(aVertexPosition, 1.0);
    vColor = aVertexColor;
  }
</script>
```

Pobieranie shaderów 1

```
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;
        }
```

Pobieranie shaderów 2

```
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;
  }
```

Inicjowanie shaderów 1

```
var shaderProgram;

  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");
    }
```

Inicjowanie shaderów 2

```
  gl.useProgram(shaderProgram);

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

  shaderProgram.vertexColorAttribute = gl.getAttribLocation(shaderProgram,
 "aVertexColor");
  gl.enableVertexAttribArray(shaderProgram.vertexColorAttribute);

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

Biblioteki przetwarzania macierzy

```
var mvMatrix = mat4.create();
    var mvMatrixStack = [];
    var pMatrix = mat4.create();

    function mvPushMatrix() {
        var copy = mat4.create();
        mat4.set(mvMatrix, copy);
        mvMatrixStack.push(copy);
    }

    function mvPopMatrix() {
        if (mvMatrixStack.length == 0) {
            throw "Invalid popMatrix!";
        }
        mvMatrix = mvMatrixStack.pop();
    }
```

```
function setMatrixUniforms() {

gl.uniformMatrix4fv(shaderProgram.pM
atrixUniform, false, pMatrix);

gl.uniformMatrix4fv(shaderProgram.mv
MatrixUniform, false, mvMatrix);
    }

    function degToRad(degrees) {
        return degrees * Math.PI / 180;
    }
```

Inicjowanie buforów 1

```
var cubeVertexPositionBuffer;
var cubeVertexColorBuffer;
var cubeVertexIndexBuffer;

function initBuffers() {
    cubeVertexPositionBuffer = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, cubeVertexPositionBuffer);
    vertices = [
        // Front face
        -1.0, -1.0,  1.0,
         1.0, -1.0,  1.0,
         1.0,  1.0,  1.0,
        -1.0,  1.0,  1.0, …
```

Inicjowanie buforów 2

```
gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(vertices), gl.STATIC_DRAW);
cubeVertexPositionBuffer.itemSize = 3;
cubeVertexPositionBuffer.numItems = 24;

cubeVertexColorBuffer = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, cubeVertexColorBuffer);
colors = [
   [1.0, 0.0, 0.0, 1.0], // Front face
   [1.0, 1.0, 0.0, 1.0], // Back face
   [0.0, 1.0, 0.0, 1.0], // Top face
   [1.0, 0.5, 0.5, 1.0], // Bottom face
   [1.0, 0.0, 1.0, 1.0], // Right face
   [0.0, 0.0, 1.0, 1.0]  // Left face
];
var unpackedColors = [];
for (var i in colors) {
   var color = colors[i];
   for (var j=0; j < 4; j++) {
      unpackedColors = unpackedColors.concat(color);
   }
}
```

Inicjowanie buforów 3

```
    gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(unpackedColors), gl.STATIC_DRAW);
    cubeVertexColorBuffer.itemSize = 4;
    cubeVertexColorBuffer.numItems = 24;

    cubeVertexIndexBuffer = gl.createBuffer();
    gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER, cubeVertexIndexBuffer);
    var cubeVertexIndices = [
      0, 1, 2,      0, 2, 3,    // Front face
      4, 5, 6,      4, 6, 7,    // Back face
      8, 9, 10,     8, 10, 11,  // Top face
      12, 13, 14,   12, 14, 15, // Bottom face
      16, 17, 18,   16, 18, 19, // Right face
      20, 21, 22,   20, 22, 23  // Left face
    ];
    gl.bufferData(gl.ELEMENT_ARRAY_BUFFER, new Uint16Array(cubeVertexIndices), gl.STATIC_DRAW);
    cubeVertexIndexBuffer.itemSize = 1;
    cubeVertexIndexBuffer.numItems = 36;
  }
```

Obroty sześcianów

```
•    var xRot = Array();
•            xRot.push(12);
•            xRot.push(112);
•            xRot.push(212);
•            xRot.push(312);
•      var xSpeed = Array();
•            xSpeed.push(1012);
•            xSpeed.push(1112);
•            xSpeed.push(1212);
•            xSpeed.push(1312);

•      var yRot = 0;
•      var ySpeed = 0;

•      var z = -25.0;

•      var filter = 0;

•      var stop = false;
```

Rysowanie 1

```
   function drawBlock(i){

                  mat4.translate(mvMatrix, [xpos[i], 0.0, 0]);

                  mvPushMatrix();

mat4.rotate(mvMatrix, degToRad(xRot[i]), [1, 0, 0]);
mat4.rotate(mvMatrix, degToRad(yRot), [0, 1, 0]);

gl.bindBuffer(gl.ARRAY_BUFFER, cubeVertexPositionBuffer);
gl.vertexAttribPointer(shaderProgram.vertexPositionAttribute, cubeVertexPositionBuffer.itemSize, gl.FLOAT, false, 0, 0);

gl.bindBuffer(gl.ARRAY_BUFFER, cubeVertexColorBuffer);
gl.vertexAttribPointer(shaderProgram.vertexColorAttribute, cubeVertexColorBuffer.itemSize, gl.FLOAT, false, 0, 0);

gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER, cubeVertexIndexBuffer);
setMatrixUniforms();

gl.drawElements(gl.TRIANGLES, cubeVertexIndexBuffer.numItems, gl.UNSIGNED_SHORT, 0);

                  mvPopMatrix();

   }
```

Rysowanie 2

```
function drawScene() {
   gl.viewport(0, 0, gl.viewportWidth, gl.viewportHeight);
          gl.clearColor(0, 0, 0, 0);
   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, [-15, 5.0, z]);

          for ( var i=0; i<4; i++ )
                    drawBlock(i);
  }
```

Animacja

```
var lastTime = 0;

function animate() {
   var timeNow = new Date().getTime();
   if (lastTime != 0) {
      var elapsed = timeNow - lastTime;
                  for ( var i=0; i<4; i++ ) {
                             xRot[i] += (xSpeed[i] * elapsed) / 1000.0;
                  }
      yRot += (ySpeed * elapsed) / 1000.0;
   }
   lastTime = timeNow;
}
```

Obsługa klawiatury

```
var currentlyPressedKeys = {};

function handleKeyDown(event) {
   currentlyPressedKeys[event.keyCode] = true;
}


function handleKeyUp(event) {
   currentlyPressedKeys[event.keyCode] = false;
}


function handleKeys() {
                        if (currentlyPressedKeys[32]) {
                                                //spacja
                                                stop=true;
                        }
   if (currentlyPressedKeys[13]) {
      // ENTER
      stop=false;
                                                finished=false;
                                                xSpeed[0] = 1345;
                                                xSpeed[1] = 1245;
                                                xSpeed[2] = 1145;
                                                xSpeed[3] = 1045;
   }
}
```

Pętla zdarzeń

```
 var c = Array(); for ( var i=0; i<4; i++ )c.push(0);
 var points = 0;
 var finished=false;

function tick() {
   stopping();
    requestAnimFrame(tick);
    handleKeys();
    drawScene();
    animate();
}
```

Zatrzymywanie

```
Function stopping(){
        if(stop&&!finished){
            for ( var i=0; i<4; i++ ) {
                if(xRot[i] % 90 < 5)xSpeed[i]=0;
            }
            if((xSpeed[0]==0)&&(xSpeed[1]==0)&&(xSpeed[2]==0)&&(xSpeed[3]==0)){
                for ( var i=0; i<4; i++ ) {
                            c[i]= (xRot[i] % 360) / 90; c[i]= c[i].toFixed(0);
            }
            if((c[0]==c[1])&&(c[1]==c[2])&&(c[2]==c[3])){
                points+=1000;
                document.getElementById("alert").innerHTML='Wygrana 1-go stopnia. Razem masz '+points;
            }else if((c[0]==c[1])&&(c[1]==c[2])||(c[1]==c[2])&&(c[2]==c[3])||(c[1]==c[0])&&(c[0]==c[3])||(c[2]==c[0])&&(c[0]==c[3])){
                points+=100;
                document.getElementById("alert").innerHTML='Wygrana 2-go stopnia. Razem masz '+points;
            }else if((c[0]==c[1])||(c[1]==c[2])||(c[2]==c[3])||(c[0]==c[3])||(c[0]==c[2])||(c[1]==c[3])){
                points+=10;
                document.getElementById("alert").innerHTML='Wygrana 3-go stopnia. Razem masz '+points;
            }else{
                document.getElementById("alert").innerHTML='Nic nie wygrałeś. Razem masz '+points;
            }
            finished=true;
            }
            }
}
```