Gra w kości

Nagłówek

```
<html>
<head>
<title>Alan gra w kości</title>
<meta http-equiv="content-type" content="text/html;
    charset=utf-8">
<script type="text/javascript" src="glMatrix-
    0.9.5.min.js"></script>
<script type="text/javascript" src="webgl-
    utils.js"></script>
<link type="text/css" href="styles.css" rel="stylesheet" />
```

Shadery

```
<script id="shader-fs" type="x-
    shader/x-fragment">
  #ifdef GL_ES
  precision highp float;
  #endif
  varying vec2 vTextureCoord;
  uniform sampler2D uSampler;
  void main(void) {
     gl_FragColor =
    texture2D(uSampler,
    vec2(vTextureCoord.s,
    vTextureCoord.t));
  }
</script>
```

```
<script id="shader-vs" type="x-
    shader/x-vertex">
  attribute vec3 aVertexPosition;
  attribute vec2 aTextureCoord;
  uniform mat4 uMVMatrix;
  uniform mat4 uPMatrix;
  varying vec2 vTextureCoord;
void main(void) {
     gl_Position = uPMatrix *
    uMVMatrix *
    vec4(aVertexPosition, 1.0);
     vTextureCoord =
    aTextureCoord;
  }
</script>
```

Obsługa karty graficznej w JS cz.1

```
<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(„Brak obsługi webGL");
        }
    }
```

```
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;
    }
```

Obsługa karty graficznej w JS cz.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;
}
```

Obsługa karty graficznej w JS cz.3

```
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");
    }
    gl.useProgram(shaderProgram);
    shaderProgram.vertexPositionAttribute = gl.getAttribLocation(shaderProgram, "aVertexPosition");
    gl.enableVertexAttribArray(shaderProgram.vertexPositionAttribute);
    shaderProgram.textureCoordAttribute = gl.getAttribLocation(shaderProgram, "aTextureCoord");
    gl.enableVertexAttribArray(shaderProgram.textureCoordAttribute);
    shaderProgram.pMatrixUniform = gl.getUniformLocation(shaderProgram, "uPMatrix");
    shaderProgram.mvMatrixUniform = gl.getUniformLocation(shaderProgram, "uMVMatrix");
    shaderProgram.samplerUniform = gl.getUniformLocation(shaderProgram, "uSampler");
}
```

Obsługa tekstur

```
function handleLoadedTexture(textures) {
    gl.pixelStorei(gl.UNPACK_FLIP_Y_WEBGL, true);
    gl.bindTexture(gl.TEXTURE_2D, textures[0]);
    gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, gl.RGBA, gl.UNSIGNED_BYTE,
      textures[0].image);
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MAG_FILTER, gl.NEAREST);
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER, gl.NEAREST);
    gl.bindTexture(gl.TEXTURE_2D, textures[1]);
    gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, gl.RGBA, gl.UNSIGNED_BYTE,
      textures[1].image);
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MAG_FILTER, gl.LINEAR);
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER, gl.LINEAR);
    gl.bindTexture(gl.TEXTURE_2D, textures[2]);
    gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, gl.RGBA, gl.UNSIGNED_BYTE,
      textures[2].image);
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MAG_FILTER, gl.LINEAR);
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER,
      gl.LINEAR_MIPMAP_NEAREST);
    gl.generateMipmap(gl.TEXTURE_2D);
    gl.bindTexture(gl.TEXTURE_2D, null);
  }
```

```
var crateTextures = Array();
  function initTexture() {
    var crateImage = new Image();
    for (var i=0; i < 3; i++) {
      var texture = gl.createTexture();
      texture.image = crateImage;
      crateTextures.push(texture);
    }
   crateImage.onload = function () {
  handleLoadedTexture(crateTextures)
    }
    crateImage.src = "crate.gif";
  }
```

Biblioteka funkcji webGL

```
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(shaderProgra
    m.pMatrixUniform, false, pMatrix);

    gl.uniformMatrix4fv(shaderProgra
    m.mvMatrixUniform, false,
    mvMatrix);
}

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

Stan początkowy i prędkość kości

```
var xRot = Array();
    xRot.push(12);
    xRot.push(112);
    xRot.push(212);
    xRot.push(312);
    xRot.push(412);
var xSpeed = Array();
    xSpeed.push(1012);
    xSpeed.push(1112);
    xSpeed.push(1212);
    xSpeed.push(1312);
    xSpeed.push(1412);
```

```
var yRot = Array();
    yRot.push(12);
    yRot.push(112);
    yRot.push(212);
    yRot.push(312);
    yRot.push(412);
var ySpeed = Array();
    ySpeed.push(1012);
    ySpeed.push(1112);
    ySpeed.push(1212);
    ySpeed.push(1312);
    ySpeed.push(1412);
```

Obsługa klawiatury 1

```
var z = -15.0;
var filter = 0;
var stop = false;
 var visible = Array();
 visible.push(true);visible.push(true);visible.push(true);visible.push(t
 rue);visible.push(true);
var currentlyPressedKeys = {};
function handleKeyDown(event) {
    currentlyPressedKeys[event.keyCode] = true;
}
function handleKeyUp(event) {
    currentlyPressedKeys[event.keyCode] = false;
}
```

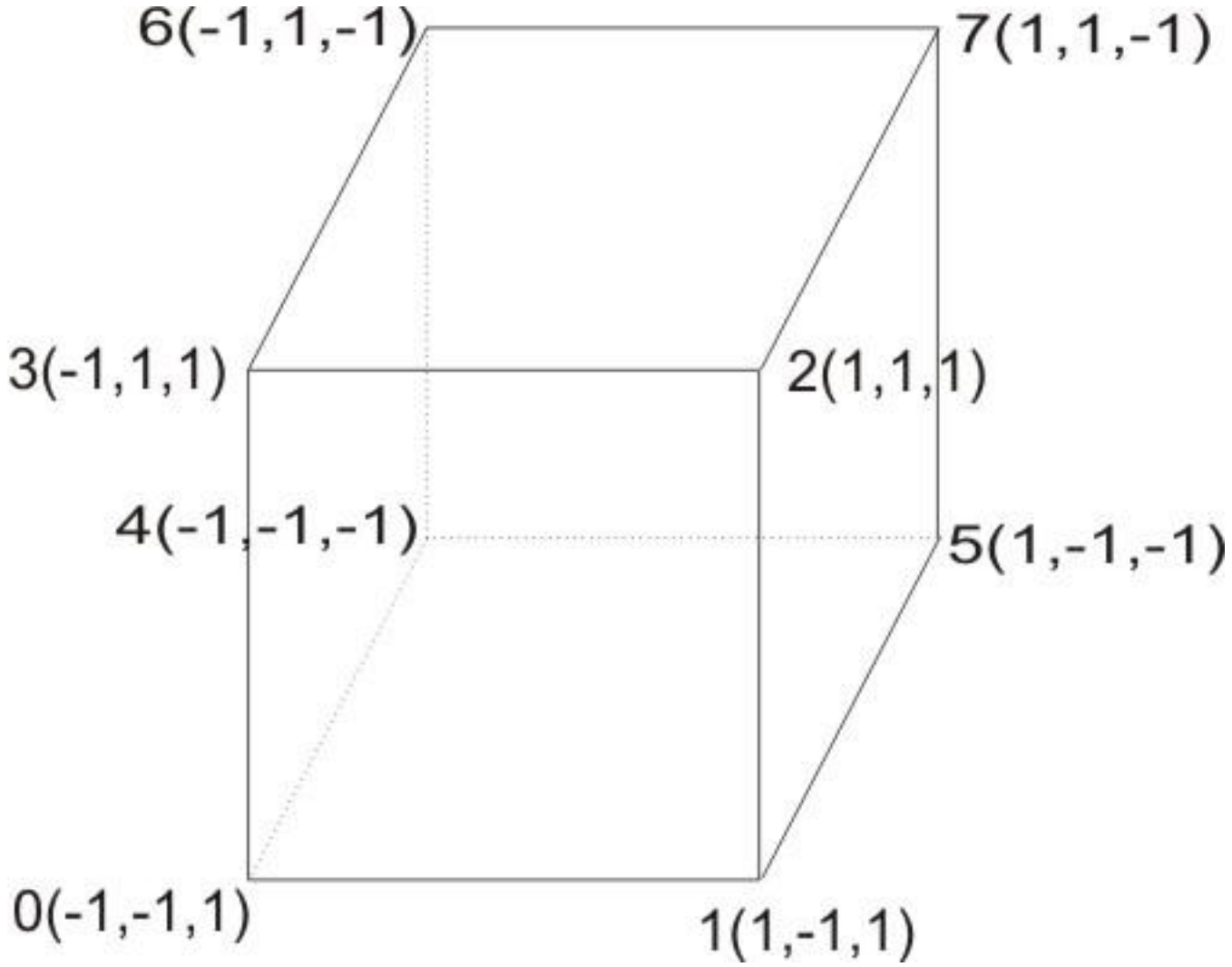
Obsługa klawiatury 2

```
function handleKeys() {
        if((level==0)||(level==1)||(level==5)||(level==9)||(level==3)||(level==7)||(level==11))
                if ((currentlyPressedKeys[32])||currentlyPressedKeys[13]) { //spacja lub ENTER
                        level++;
                        currentlyPressedKeys[32]=false;
                        currentlyPressedKeys[13]=false;
                }
        if((level==3)||(level==7)||(level==11))//gdy przygotowanie kolejnego losowania
                for (var i=0;i<5 ;i++ )
                        if (currentlyPressedKeys[49+i]) { //klawisze numeryczne
                                visible[i]=!visible[i];//uwidaczniają lub chowają kości
                                currentlyPressedKeys[49+i]=false;
                        }
  }
```

Wierzchołki sześcianu w 3D

```
var cubeVertexPositionBuffer;
var
  cubeVertexTextureCoordBuffer;
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,

        // Back 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,

        // Top 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,

        // Bottom 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,

        // Right 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,

        // Left 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
    ];
```

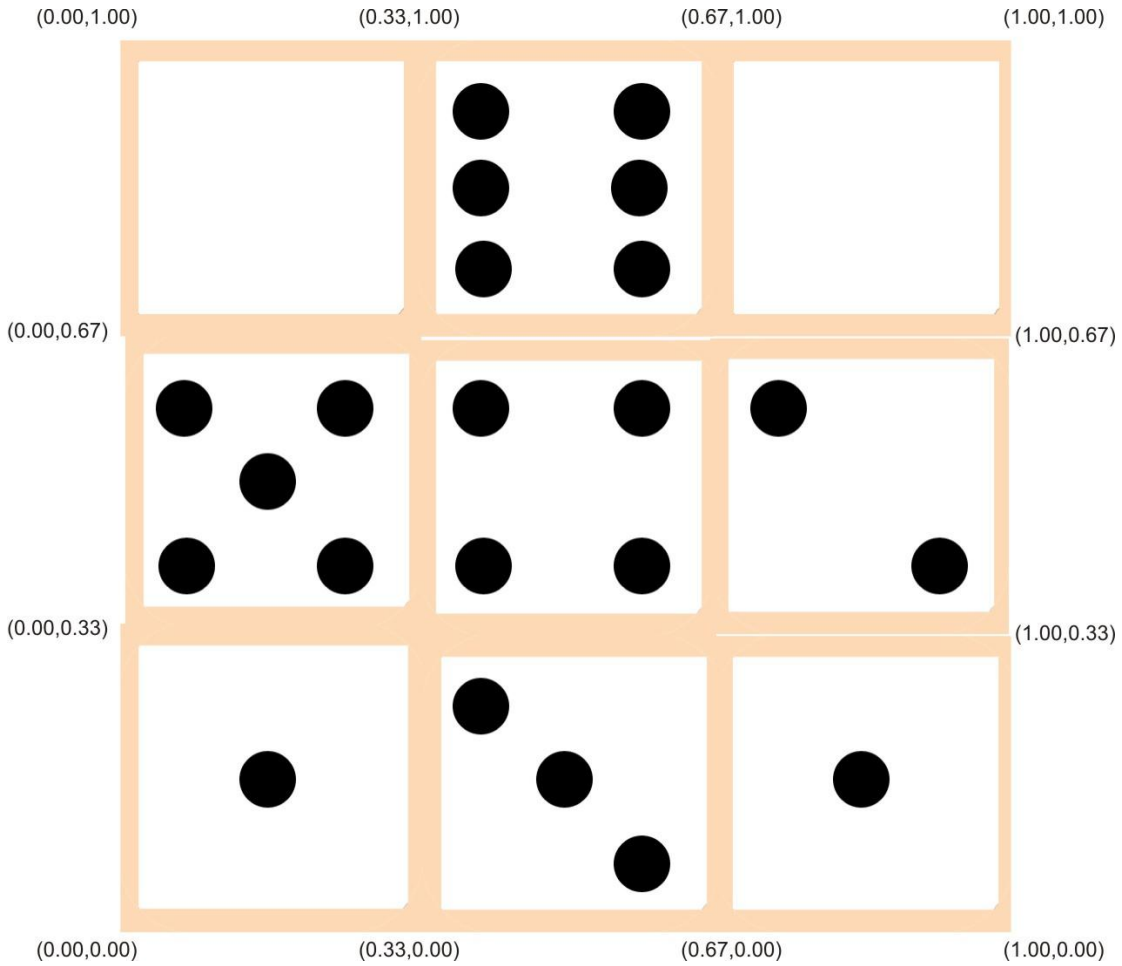
Ponumerowane wierzchołki kostki



Nakładanie tekstury na 3D

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

cubeVertexPositionBuffer.itemSize = 3;

cubeVertexPositionBuffer.numItems = 24;

  cubeVertexTextureCoordBuffer = gl.createBuffer();
  gl.bindBuffer(gl.ARRAY_BUFFER, cubeVertexTextureCoordBuffer);
  var textureCoords = [
// Front face
    0.0, 0.0,
    0.33, 0.0,
    0.33, 0.33,
    0.0, 0.33,

    // Back face
    0.67, 0.33,
    0.67, 0.67,
    0.33, 0.67,
    0.33, 0.33,

    // Top face
    0.67, 0.67,
    0.67, 0.33,
    1.0, 0.33,
    1.0, 0.67,

    // Bottom face
    0.67, 0.33,
    0.33, 0.33,
    0.33, 0.0,
    0.67, 0.0,

    // Right face
    0.33, 0.33,
    0.33, 0.67,
    0.0, 0.67,
    0.0, 0.33,

    // Left face
    0.67, 1.0,
    0.33, 1.0,
    0.33, 0.67,
    0.67, 0.67,
  ];
```

Tekstura kostki na płaszczyźnie



Złożenie tekstury z trójkątów

```
  gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(textureCoords), gl.STATIC_DRAW);
  cubeVertexTextureCoordBuffer.itemSize = 2;
  cubeVertexTextureCoordBuffer.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;
}
        var xpos = Array();
        for ( var i=0; i<5; i++ )
                xpos.push(5.0);
```

Rysowanie pojenczej kości

```
function drawBlock(i){
        mat4.translate(mvMatrix, [3, 0.0, 0]);
        mvPushMatrix();
mat4.rotate(mvMatrix, degToRad(xRot[i]), [1, 0, 0]);
mat4.rotate(mvMatrix, degToRad(yRot[i]), [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, cubeVertexTextureCoordBuffer);
gl.vertexAttribPointer(shaderProgram.textureCoordAttribute,
cubeVertexTextureCoordBuffer.itemSize, gl.FLOAT, false, 0, 0);
gl.activeTexture(gl.TEXTURE0);
gl.bindTexture(gl.TEXTURE_2D, crateTextures[filter]);
gl.uniform1i(shaderProgram.samplerUniform, 0);
gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER, cubeVertexIndexBuffer);
setMatrixUniforms();
if(visible[i])gl.drawElements(gl.TRIANGLES, cubeVertexIndexBuffer.numItems, gl.UNSIGNED_SHORT,
0);
        mvPopMatrix();
}
```

Rysowanie całej sceny

```
function drawScene() {
   var p="";
   var r="";
   for ( var i=1; i<6; i++ ) {//numerowanie kości
            p=p+'    '+i.toString()+'    ';
            //r=r+'    '+c[i].toString()+'    ';
   }
   document.getElementById('result').innerHTML="<h1>"+p+"</h1>";//<h1>"+r+"</h1>";
   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(12, gl.viewportWidth / gl.viewportHeight, 0.1, 100.0, pMatrix);
   mat4.identity(mvMatrix);
   mat4.translate(mvMatrix, [-9, 0.0, z]);
   for ( var i=0; i<5; i++ )
            drawBlock(i);//rysowanie pojedynczych kości
   }
   var lastTime = 0;
```

Obrazowanie uwzględniające czas

```
function animate() {
    var timeNow = new Date().getTime();
    if (lastTime != 0) {
        var elapsed = timeNow - lastTime;
        for ( var i=0; i<5; i++ ) {
            xRot[i] += (xSpeed[i] * elapsed) / 1000.0;
            yRot[i] += (ySpeed[i] * elapsed) / 1000.0;
        }
    }
    lastTime = timeNow;
}
```

Dane do reguł gry

```
function losuj(){
    for(var i=0;i<5;i++)
        if(visible[i]){
            xSpeed[i] = Math.random()*545+445;
            ySpeed[i] = Math.random()*545+445;
        }
}
var cx = Array(); for ( var i=0; i<5; i++ )cx.push(0);
var cy = Array(); for ( var i=0; i<5; i++ )cy.push(0);
var cy = Array(); for ( var i=0; i<5; i++ )cy.push(0);
var points = 0;
var finished=false;
var level=1;
var blad=false;
var punkty = Array();
```

```
var nazwy=[
"jedynki",
"dwójki",
"trójki",
"czwórki",
"piątki",
"szóstki",
"premia",
"suma",
"trzy jednakowe",
"cztery jednakowe",
"full",
"mały strit",
"duży strit",
"generał",
"szansa",
"suma",
"razem"];
```

Odczyt liczby oczek z położenia 1

```
var cw=[0,0,0,0,0,0];
var p='';var x;
var r='';var y;
var s=''; var w;

function wyznaczoczka(){
        for ( var i=0; i<5; i++ ) {
                cx[i]= Math.floor((xRot[i] % 360) / 90);
        }
        for ( var i=0; i<5; i++ ) {
                cy[i]= Math.floor((yRot[i] % 360) / 90);  ;
        }
        s='';
        for ( var i=0; i<5; i++ ) {
```

Odczyt liczby oczek z położenia 2

```
                x=cx[i];
                p=p+'    '+x.toString()+'    ';
                y=cy[i];
                r=r+'    '+y.toString()+'    ';
                if(cx[i]==1)w=2;else
                if(cx[i]==3)w=3;else
                if(cx[i]==0){
                        if(cy[i]==0)w=1;else
                        if(cy[i]==1)w=6;else
                        if(cy[i]==2)w=4;else
                        if(cy[i]==3)w=5;
                }else
                if(cx[i]==2){
                        if(cy[i]==0)w=4;else
                        if(cy[i]==1)w=5;else
                        if(cy[i]==2)w=1;else
                        if(cy[i]==3)w=6;
                }
                cw[i]=w;
                s+='    '+w.toString()+'    ';
        }
}
```

Wyliczanie zdobytych punktów 1

```
function wyliczpunkty(){
	var ls=0;
	for(var i=1;i<7;i++){
			var l=0;
			for(var j=0;j<5;j++){

	if(cw[j]==i)l++;
			}
			punkty.push(l*i);
			ls+=l*i;
	}
document.getElementById("alert").innerHTML='Wyznaczone';
	if(ls>=63)punkty.push(35);else punkty.push(0);
	punkty.push(ls+punkty[6]);
	var oczka = [0,0,0,0,0,0];
	for(var i=0;i<5;i++){
			oczka[cw[i]-1]++;
	}
```

```
punkty.push(0);//3 jedn.
punkty.push(0);//4 jedn.
punkty.push(0);//full
punkty.push(0);//mały street
punkty.push(0);//duży street
punkty.push(0);//generał
punkty.push(0);//szansa
punkty.push(0);//suma
punkty.push(0);//razem
for(var i=0;i<6;i++){
	if(oczka[i]==3)punkty[8]=ls;
	if(oczka[i]==4)punkty[9]=ls;
	for(var j=0;j<6;j++){
	 if((oczka[i]==3)&&(oczka[j]==2))punkty[10]=25;
}
if(oczka[i]==0)eq=0;
if(oczka[i]==5)punkty[13]=50;
}
```

Wyliczanie zdobytych punktów 2

```
var eq=1;
for(var i=0;i<5;i++){
        if(oczka[i]==0)eq=0;
}
if(eq==1)punkty[12]=40;
var eq=1;
for(var i=1;i<6;i++){
        if(oczka[i]==0)eq=0;
}
if(eq==1)punkty[12]=40;
eq=1;
for(var i=0;i<4;i++){
        if(oczka[i]==0)eq=0;
}
if(eq==1)punkty[11]=30;
eq=1;
for(var i=1;i<5;i++){
        if(oczka[i]==0)eq=0;
}
if(eq==1)punkty[11]=30;
eq=1;
for(var i=2;i<6;i++){
        if(oczka[i]==0)eq=0;
}
if(eq==1)punkty[11]=30;
punkty[14]=ls;
for(var i=8;i<15;i++)
if(punkty[15]<punkty[i])punkty[15]=punkty[i
];
punkty[16]=punkty[7]+punkty[15];
}
```

Interakcja z użytkownikiem

```
function dialog(){
	switch(level){
	case 0:{ document.getElementById("alert").innerHTML='Naciśnij klawisz ENTER lub SPACE by rozpocząć losowanie'; break;}
	case 1:; case 5:; case 9:{ document.getElementById("alert").innerHTML='Naciśnij klawisz ENTER lub SPACE aby zatrzymać losowanie'; break;}
	case 2:; case 6:; case 10:{
			document.getElementById("alert").innerHTML='Poczekaj aż kostki zatrzymają się!';
			for (var i=0; i<5;i++ )
				visible[i]=true;
			for ( var i=0; i<5; i++ ) {
				if((xRot[i] % 90 < 15)&&(yRot[i] % 90 < 15)){xSpeed[i]=0;ySpeed[i]=0;}
			}
```

Interakcja z użytkownikiem 2

```
        if((xSpeed[0]==0)&&(xSpeed[1]==0)&&(xSpeed[2]==0)&&(xSpeed[3]==0)&&(xSpeed[4]==0)

        &&(ySpeed[0]==0)&&(ySpeed[1]==0)&&(ySpeed[2]==0)&&(ySpeed[3]==0)&&(ySpeed[4]==
0)){
                                level++;
                        }
                        break;}
        case 3:; case 7:; case 11:{
                        if(blad)if(level==11)document.getElementById("alert").innerHTML='BŁĄD! Nie
możesz już żadnych kości chcesz rzucać ponownie. Zmień przyciskami numerycznymi i naciśnij
ENTER';
                        else document.getElementById("alert").innerHTML='BŁĄD! Zbyt wiele kości
chcesz rzucać ponownie. Można najwyżej trzy. Zmień przyciskami numerycznymi i naciśnij ENTER';
                        else document.getElementById("alert").innerHTML='Przyciskami numerycznymi
możesz(ale nie musisz) wybrać do trzech kości do kolejnego rzucania. Potem wciśnij ENTER.';break;}
```

Interakcja z użytkownikiem 3

```
case 4:; case 8:; case 12:{
        var v=0;
        for (var i=0; i<5;i++ )
                if(!visible[i])v++;
        if((v>3)||(level==12)&&(v>0)) {
                blad=true;
                level--;
        }else if(v==0){
                level=13;
        }else{
                blad=false;
                start=true;
                for (var i=0; i<5;i++ )
                        visible[i]=!visible[i];
                losuj();
                level++;
        }
```

Interakcja z użytkownikiem 4

```
        case 4:; case 8:; case 12:{
                                        case 13:{
                                        document.getElementById("alert").innerHTML='Obliczanie wyniku';
                                        wyznaczoczka();
                                        wyliczpunkty();
                                        var p='';
                                        for(var i=0;i<17;i++){
                                                        w=punkty[i];

        p+="<tr><td>"+nazwy[i]+"</td><td>"+w.toString()+"</td></tr>";
                                        }
                                        document.getElementById('alert').innerHTML=
"<h1>"+s+"</h1><center><table align='center' border='2'>"+p+"</table></center>";
                        break; }
        }
}
```

Główna pętla gry

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

Inicjalizacja gry

```
function webGLStart() {
    var canvas = document.getElementById(„rzucanie");
    initGL(canvas);
    initShaders();
    initBuffers();
    initTexture();
    gl.clearColor(0.0, 0.0, 0.0, 1.0);
    gl.enable(gl.DEPTH_TEST);
    document.onkeydown = handleKeyDown;
    document.onkeyup = handleKeyUp;
   losuj();
    tick();
}
```

```
</script>
</head>
<body onload="webGLStart();">
<center><div id='result'></div></center>
    <canvas id=„rzucanie" style="border: none;"
     width="500" height="100"></canvas>
<center><div id='alert'></div></center>
<center><a href="http://alanbit.pl">Powrót do strony
    głównej</a></center>
</body>
</html>
```