Realizacja prostej gry polegającej na symulacji jazdy samochodem

zadanie

Należy zaprogramować symulator jazdy samochodem

Sterowany kursorami odpowiadającymi odpowiednio za skręty w lewo i w prawo oraz przyspieszanie i zwalnianie

W widoku z góry na płaszczyźnie prostokąta uniemożliwiającego wydostanie się modelu poza jego boczne krawędzie

Powierzchnia do 3D

```
</script>
</head>
<body onload="webGLStart();">
    <canvas id="autko" style="border: none;"
    width="500" height="500"></canvas>
    <div id="v"></div><br />
</body>
</html>
```

Inicjalizacja webGL

```
var gl;
  function initGL(canvas) {
     try {
        gl = canvas.getContext("experimental-
    webgl");
        gl.viewportWidth = canvas.width;
        gl.viewportHeight = canvas.height;
     } catch (e) {
     }
     if (!gl) {
        alert("Użyj nowszej przeglądarki typu
    Chrome lub FireFox");
     }
  }
```

```
   function webGLStart() {
        var canvas =
document.getElementById("autko");
        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;
        tick();
              }
```

Animowanie w cyklach przez nasłuch klawiatury i każdorazowe rysowanie

```
var lastTime = 0;
  function animate() {
     var timeNow = new Date().getTime();
     if (lastTime != 0) {
        var elapsed = timeNow - lastTime;
        if(Math.abs(x+(speed * elapsed)/1000.0  * Math.sin(degToRad(-zRot)))<18
                              &&Math.abs(y+(speed * elapsed)/1000.0  * Math.cos(degToRad(-zRot)))<18){
                                    x += (speed * elapsed)/1000.0  * Math.sin(degToRad(-zRot));
                                    y += (speed * elapsed)/1000.0  * Math.cos(degToRad(-zRot));
                              }
     }
     lastTime = timeNow;
  }
  function tick() {
     requestAnimFrame(tick);
     handleKeys();
     drawScene();
     animate();
  }
```

Pojedyńczy cykl rysowania 3D

```
function drawScene() {
      document.getElementById("v").innerHTML = "speed="+speed+"x="+x+" y="+y+"zRot="+zRot+" z="+z;
    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, [x, y, z]);
    mat4.rotate(mvMatrix, degToRad(xRot), [1, 0, 0]);
    mat4.rotate(mvMatrix, degToRad(yRot), [0, 1, 0]);
    mat4.rotate(mvMatrix, degToRad(zRot), [0, 0, 1]);
    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();
    gl.drawElements(gl.TRIANGLES, cubeVertexIndexBuffer.numItems, gl.UNSIGNED_SHORT, 0);
  }
```

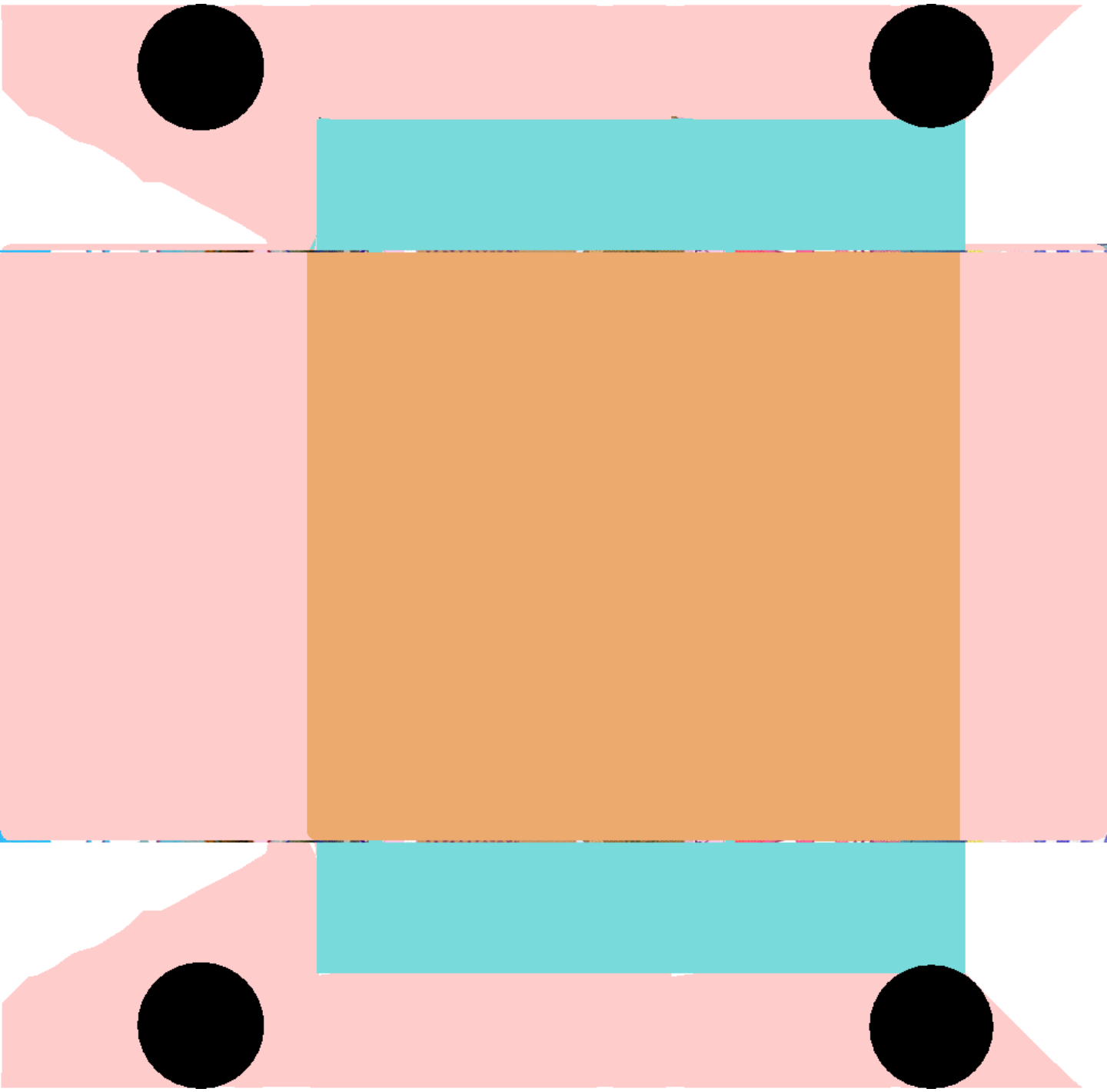
Nasze dane graficzne – wierzchołki 3D

```
var cubeVertexPositionBuffer;
  var cubeVertexTextureCoordBuffer;
  var cubeVertexIndexBuffer;
  function initBuffers() {
     cubeVertexPositionBuffer =
     gl.createBuffer();
     gl.bindBuffer(gl.ARRAY_BUFFER,
     cubeVertexPositionBuffer);
// Kolejne wierzchołki 3D
     fakturowanego autka
     vertices = [
        // Front face
        -1.0, -2.0,  1.0,
         1.0, -2.0,  1.0,
         1.0,  2.0,  1.0,
        -1.0,  2.0,  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, -2.0, -1.0,
         1.0,  2.0, -1.0,
         1.0,  2.0,  1.0,
         1.0, -2.0,  1.0,
        // Left face
        -1.0, -2.0, -1.0,
        -1.0, -2.0,  1.0,
        -1.0,  2.0,  1.0,
        -1.0,  2.0, -1.0,
     ];
  }
```

Nasze dane graficzne – współ. tekstury

```
    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);
// Kolejne współrzędne czerpane z pliku definiującego
      teksturę autka
    var textureCoords = [
      // Front face
      0.25, 0.0,
      0.75, 0.0,
      0.75, 1.0,
      0.25, 1.0,
      // Back face
     0.0, 0.0,
      0.25, 0.0,
      0.25, 1.0,
      0.0, 1.0,
 // Top face
        0.0, 0.0,
        0.25, 0.0,
        0.25, 1.0,
        0.0, 1.0,

// Bottom face
        0.75, 1.0,
        1.0, 1.0,
        1.0, 0.0,
        0.75, 0.0,
        // Right face
        0.0, 0.0,
        0.0, 1.0,
        0.25, 1.0,
        0.25, 0.0,
       // Left face
        0.0, 0.0,
        0.25, 0.0,
        0.25, 1.0,
        0.0, 1.0,
    ];
```



Nasze dane graficzne – kolejne trójkąty

```
    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);
// Kolejne trójkąty składające się na prostokąty
    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;
```

Obsługa klawiatury

```
var currentlyPressedKeys = {};
  function handleKeyDown(event) {
    currentlyPressedKeys[event.keyCode] = true;
    if (String.fromCharCode(event.keyCode) ==
     "F") {
      filter += 1;
      if (filter == 3) {
        filter = 0;
      }
    }
  }
  function handleKeyUp(event) {
    currentlyPressedKeys[event.keyCode] =
     false;
  }
  function handleKeys() {
    if (currentlyPressedKeys[33]) {
      // Page Up
      z -= 0.05;
    }
    if (currentlyPressedKeys[34]) {
      // Page Down
      z += 0.05;
    }
    if (currentlyPressedKeys[37]) {
      // Left cursor key
      zRot += 1;
    }
    if (currentlyPressedKeys[39]) {
      // Right cursor key
      zRot -= 1;
    }
    if (currentlyPressedKeys[38]) {
      // Up cursor key
      speed += 1;
    }
    if (currentlyPressedKeys[40]) {
      // Down cursor key
      speed -= 1;
    }
  }
```

Definicje zmiennych ruchu auta

```
<script type="text/javascript">
    var xRot = 0;
    var xSpeed = 0;

    var yRot = 0;
    var ySpeed = 0;

    var zRot = 0;
    var speed = 0;

    var x = 0;
    var y = 0;
    var z = -50.0;

    var filter = 0;
```

Definicja vertex shadera

```
<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>
```

Definicja fragment shadera

```
<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>
```

Biblioteki systemowe

```
<html>

<head>
<title>Symulator jazdy autkiem</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>
<script type="text/javascript" src="shaders.js"></script>
```