```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<script type="text/javascript">
var gl;
var shaderProgram;
var uPMatrix;
var vertexPositionBuffer;
var vertexColorBuffer;
var vertexCoordsBuffer;
var vertexNormalBuffer;
function MatrixMul(a,b) //Mnożenie macierzy
{
 let c = [
 0,0,0,0,
 0,0,0,0,
 0,0,0,0,
 0,0,0,0
 ]
 for(let i=0;i<4;i++)
 {
  for(let j=0;j<4;j++)
  {
   c[i*4+j] = 0.0;
   for(let k=0;k<4;k++)
   {
    c[i*4+j]+= a[i*4+k] * b[k*4+j];
   }
  }
 }
 return c;
}
function MatrixTransposeInverse(m)
{
 let r = [
     0, 0, 0, 0,
     0, 0, 0, 0,
     0, 0, 0, 0,
     0, 0, 0, 0
  ];
 r[0] = m[5]*m[10]*m[15] - m[5]*m[14]*m[11] - m[6]*m[9]*m[15] + m[6]*m[13]*m[11] + m[7]*m[9]*m[14] - m[7]*m[13]*m[10];
 r[1] = -m[1]*m[10]*m[15] + m[1]*m[14]*m[11] + m[2]*m[9]*m[15] - m[2]*m[13]*m[11] - m[3]*m[9]*m[14] + m[3]*m[13]*m[10];
 r[2] = m[1]*m[6]*m[15] - m[1]*m[14]*m[7] - m[2]*m[5]*m[15] + m[2]*m[13]*m[7] + m[3]*m[5]*m[14] - m[3]*m[13]*m[6];
 r[3] = -m[1]*m[6]*m[11] + m[1]*m[10]*m[7] + m[2]*m[5]*m[11] - m[2]*m[9]*m[7] - m[3]*m[5]*m[10] + m[3]*m[9]*m[6];
 r[4] = -m[4]*m[10]*m[15] + m[4]*m[14]*m[11] + m[6]*m[8]*m[15] - m[6]*m[12]*m[11] - m[7]*m[8]*m[14] + m[7]*m[12]*m[10];
 r[5] = m[0]*m[10]*m[15] - m[0]*m[14]*m[11] - m[2]*m[8]*m[15] + m[2]*m[12]*m[11] + m[3]*m[8]*m[14] - m[3]*m[12]*m[10];
 r[6] = -m[0]*m[6]*m[15] + m[0]*m[14]*m[7] + m[2]*m[4]*m[15] - m[2]*m[12]*m[7] - m[3]*m[4]*m[14] + m[3]*m[12]*m[6];
 r[7] = m[0]*m[6]*m[11] - m[0]*m[10]*m[7] - m[2]*m[4]*m[11] + m[2]*m[8]*m[7] + m[3]*m[4]*m[10] - m[3]*m[8]*m[6];
 r[8] = m[4]*m[9]*m[15] - m[4]*m[13]*m[11] - m[5]*m[8]*m[15] + m[5]*m[12]*m[11] + m[7]*m[8]*m[13] - m[7]*m[12]*m[9];
 r[9] = -m[0]*m[9]*m[15] + m[0]*m[13]*m[11] + m[1]*m[8]*m[15] - m[1]*m[12]*m[11] - m[3]*m[8]*m[13] + m[3]*m[12]*m[9];
 r[10] = m[0]*m[5]*m[15] - m[0]*m[13]*m[7] - m[1]*m[4]*m[15] + m[1]*m[12]*m[7] + m[3]*m[4]*m[13] - m[3]*m[12]*m[5];
 r[11] = -m[0]*m[5]*m[11] + m[0]*m[9]*m[7] + m[1]*m[4]*m[11] - m[1]*m[8]*m[7] - m[3]*m[4]*m[9] + m[3]*m[8]*m[5];
 r[12] = -m[4]*m[9]*m[14] + m[4]*m[13]*m[10] + m[5]*m[8]*m[14] - m[5]*m[12]*m[10] - m[6]*m[8]*m[13] + m[6]*m[12]*m[9];
 r[13] = m[0]*m[9]*m[14] - m[0]*m[13]*m[10] - m[1]*m[8]*m[14] + m[1]*m[12]*m[10] + m[2]*m[8]*m[13] - m[2]*m[12]*m[9];
 r[14] = -m[0]*m[5]*m[14] + m[0]*m[13]*m[6] + m[1]*m[4]*m[14] - m[1]*m[12]*m[6] - m[2]*m[4]*m[13] + m[2]*m[12]*m[5];
 r[15] = m[0]*m[5]*m[10] - m[0]*m[9]*m[6] - m[1]*m[4]*m[10] + m[1]*m[8]*m[6] + m[2]*m[4]*m[9] - m[2]*m[8]*m[5];
 var det = m[0]*r[0] + m[1]*r[4] + m[2]*r[8] + m[3]*r[12];
 for (var i = 0; i < 16; i++) r[i] /= det;

 let rt = [ r[0], r[4], r[8], r[12],
         r[1], r[5], r[9], r[13],
         r[2], r[6], r[10], r[14],
         r[3], r[7], r[11], r[15]
         ];

 return rt;
}
function CreateIdentytyMatrix()
{
 return [
 1,0,0,0, //Macierz jednostkowa
```

```
 0,1,0,0,
 0,0,1,0,
 0,0,0,1
 ];
}
function CreateTranslationMatrix(tx,ty,tz)
{
 return  [
 1,0,0,0,
 0,1,0,0,
 0,0,1,0,
 tx,ty,tz,1
 ];
}
function CreateScaleMatrix(sx,sy,sz)
{
 return [
 sx,0,0,0,
 0,sy,0,0,
 0,0,sz,0,
 0,0,0,1
 ];
}
function CreateRotationZMatrix(angleZ)
{
 return [
 +Math.cos(angleZ*Math.PI/180.0),+Math.sin(angleZ*Math.PI/180.0),0,0,
 -Math.sin(angleZ*Math.PI/180.0),+Math.cos(angleZ*Math.PI/180.0),0,0,
 0,0,1,0,
 0,0,0,1
 ];
}
function CreateRotationYMatrix(angleY)
{
 return [
 +Math.cos(angleY*Math.PI/180.0),0,-Math.sin(angleY*Math.PI/180.0),0,
 0,1,0,0,
 +Math.sin(angleY*Math.PI/180.0),0,+Math.cos(angleY*Math.PI/180.0),0,
 0,0,0,1
 ];
}
function CreateRotationXMatrix(angleX)
{
 return [
 1,0,0,0,
 0,+Math.cos(angleX*Math.PI/180.0),+Math.sin(angleX*Math.PI/180.0),0,
 0,-Math.sin(angleX*Math.PI/180.0),+Math.cos(angleX*Math.PI/180.0),0,
 0,0,0,1
 ];
}
function createRect(mx,my,mz,dax,day,daz,dbx,dby,dbz)
{
 p1x = mx;          p1y = my;          p1z = mz;
 p2x = mx + dax;    p2y = my + day;    p2z = mz + daz;
 p3x = mx + dbx;    p3y = my + dby;    p3z = mz + dbz;
 p4x = mx + dax + dbx; p4y = my + day + dby; p4z = mz + daz + dbz;

 let vertexPosition = [p1x,p1y,p1z, p2x,p2y,p2z, p4x,p4y,p4z,  //Pierwszy trójkąt
                       p1x,p1y,p1z, p4x,p4y,p4z, p3x,p3y,p3z]; //Drugi trójkąt

 return vertexPosition;
}
function createNormal(p1x,p1y,p1z,p2x,p2y,p2z,p3x,p3y,p3z) //Wyznaczenie wektora normalnego dla trójkąta
{
 let v1x = p2x - p1x;
 let v1y = p2y - p1y;
 let v1z = p2z - p1z;

 let v2x = p3x - p1x;
 let v2y = p3y - p1y;
 let v2z = p3z - p1z;
```

```
 let v3x =  v1y*v2z - v1z*v2y;
 let v3y =  v1z*v2x - v1x*v2z;
 let v3z =  v1x*v2y - v1y*v2x;

 vl = Math.sqrt(v3x*v3x+v3y*v3y+v3z*v3z); //Obliczenie długości wektora

 v3x/=vl; //Normalizacja na zakreś -1 1
 v3y/=vl;
 v3z/=vl;

 let vertexNormal = [v3x,v3y,v3z, v3x,v3y,v3z, v3x,v3y,v3z];
 return vertexNormal;
}
function CreateBox(x,y,z,dx,dy,dz)
{
 //Opis sceny 3D, położenie punktów w przestrzeni 3D w formacie X,Y,Z
 let vertexPosition = []; //3 punkty po 3 składowe - X1,Y1,Z1, X2,Y2,Z2, X3,Y3,Z3 - 1 trójkąt
 let vertexNormal = [];

 vertexPosition.push(...createRect(-1,-1,-1,0,2,0,2,0,0));
 vertexPosition.push(...createRect(-1,-1,-1,0,0,2,0,2,0));
 vertexPosition.push(...createRect(-1,-1,-1,2,0,0,0,0,2));

 vertexPosition.push(...createRect(1,1,1,-2,0,0,0,-2,0));
 vertexPosition.push(...createRect(1,1,1,0,-2,0,0,0,-2));
 vertexPosition.push(...createRect(1,1,1,0,0,-2,-2,0,0));
 //
 for(let i=0;i<vertexPosition.length;i=i+9)
 {

vertexNormal.push(...createNormal(vertexPosition[i+0],vertexPosition[i+1],vertexPosition[i+2],vertexPosition[i+3],vertexPosition[i+4],vertexPosition[i+5],verte
xPosition[i+6],vertexPosition[i+7],vertexPosition[i+8]));
 }

 return [vertexPosition, vertexNormal];
}
function startGL()
{
 alert("StartGL");
 let canvas = document.getElementById("canvas3D"); //wyszukanie obiektu w strukturze strony
 gl = canvas.getContext("experimental-webgl"); //pobranie kontekstu OpenGL'u z obiektu canvas
 gl.viewportWidth = canvas.width; //przypisanie wybranej przez nas rozdzielczości do systemu OpenGL
 gl.viewportHeight = canvas.height;

  //Kod shaderów
 const vertextShaderSource = ` //Znak akcentu z przycisku tyldy - na lewo od przycisku 1 na klawiaturze
  precision highp float;
  attribute vec3 aVertexPosition;
  attribute vec3 aVertexNormal;
  uniform mat4 uMMatrix;
  uniform mat4 uInvMMatrix;
  uniform mat4 uVMatrix;
  uniform mat4 uPMatrix;
  varying vec3 vPos;
  varying vec3 vNormal;
  uniform float uNormalMul;
  void main(void) {
   vPos = vec3(uMMatrix * vec4(aVertexPosition, 1.0));
   gl_Position = uPMatrix * uVMatrix * vec4(vPos,1.0); //Dokonanie transformacji położenia punktów z przestrzeni 3D do przestrzeni obrazu (2D)
   vNormal = normalize(mat3(uInvMMatrix) * uNormalMul*aVertexNormal); //Obrot wektorow normalnych
  }
 `;
 const fragmentShaderSource = `
  precision highp float;
  varying vec3 vPos;
  varying vec3 vNormal;
  uniform sampler2D uSampler;
  uniform vec3 uLightPosition;
  uniform vec3 uColor;
  void main(void) {
```

```
      vec3 lightDirection = normalize(uLightPosition - vPos);
      float brightness = max(dot(vNormal,lightDirection), 0.0);
     //gl_FragColor = vec4(vColor,1.0); //Ustalenie stałego koloru wszystkich punktów sceny
     //gl_FragColor = texture2D(uSampler,vTexUV)*vec4(vColor,1.0); //Odczytanie punktu tekstury i przypisanie go jako koloru danego punktu renderowaniej
figury
     //gl_FragColor = vec4((vNormal+vec3(1.0,1.0,1.0))/2.0,1.0);
     //gl_FragColor = clamp(texture2D(uSampler,vTexUV) * vec4(brightness,brightness,brightness,1.0),0.0,1.0);
     gl_FragColor = clamp(vec4(uColor,1.0) * vec4(brightness,brightness,brightness,1.0),0.0,1.0);
   }
  `;
  let fragmentShader = gl.createShader(gl.FRAGMENT_SHADER); //Stworzenie obiektu shadera
  let vertexShader   = gl.createShader(gl.VERTEX_SHADER);
  gl.shaderSource(fragmentShader, fragmentShaderSource); //Podpięcie źródła kodu shader
  gl.shaderSource(vertexShader, vertextShaderSource);
  gl.compileShader(fragmentShader); //Kompilacja kodu shader
  gl.compileShader(vertexShader);
  if (!gl.getShaderParameter(fragmentShader, gl.COMPILE_STATUS)) { //Sprawdzenie ewentualnych błedów kompilacji
   alert(gl.getShaderInfoLog(fragmentShader));
   return null;
  }
  if (!gl.getShaderParameter(vertexShader, gl.COMPILE_STATUS)) {
   alert(gl.getShaderInfoLog(vertexShader));
   return null;
  }

  shaderProgram = gl.createProgram(); //Stworzenie obiektu programu
  gl.attachShader(shaderProgram, vertexShader); //Podpięcie obu shaderów do naszego programu wykonywanego na karcie graficznej
  gl.attachShader(shaderProgram, fragmentShader);
  gl.linkProgram(shaderProgram);
  if (!gl.getProgramParameter(shaderProgram, gl.LINK_STATUS)) alert("Could not initialise shaders");  //Sprawdzenie ewentualnych błedów

  //Opis sceny 3D, położenie punktów w przestrzeni 3D w formacie X,Y,Z
  let vertexPosition; //3 punkty po 3 składowe - X1,Y1,Z1, X2,Y2,Z2, X3,Y3,Z3 - 1 trójkąt
  let vertexNormal;

  //[vertexPosition, vertexColor, vertexCoords, vertexNormal] = CreateShpere(0,0,0,2, 6, 12);
  [vertexPosition, vertexNormal] = CreateBox(0,0,0,1,1,1);

  vertexPositionBuffer = gl.createBuffer(); //Stworzenie tablicy w pamieci karty graficznej
  gl.bindBuffer(gl.ARRAY_BUFFER, vertexPositionBuffer);
  gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(vertexPosition), gl.STATIC_DRAW);
  vertexPositionBuffer.itemSize = 3; //zdefiniowanie liczby współrzednych per wierzchołek
  vertexPositionBuffer.numItems = vertexPosition.length/9; //Zdefinoiowanie liczby trójkątów w naszym buforze

  vertexNormalBuffer = gl.createBuffer();
  gl.bindBuffer(gl.ARRAY_BUFFER, vertexNormalBuffer);
  gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(vertexNormal), gl.STATIC_DRAW);
  vertexNormalBuffer.itemSize = 3;
  vertexNormalBuffer.numItems = vertexNormal.length/9;

  //Macierze opisujące położenie wirtualnej kamery w przestrzenie 3D
  let aspect = gl.viewportWidth/gl.viewportHeight;
  let fov = 45.0 * Math.PI / 180.0; //Określenie pola widzenia kamery
  let zFar = 100.0; //Ustalenie zakresów renderowania sceny 3D (od obiektu najbliższego zNear do najdalszego zFar)
  let zNear = 0.1;
  uPMatrix = [
   1.0/(aspect*Math.tan(fov/2)),0                    ,0                  ,0                    ,
   0                  ,1.0/(Math.tan(fov/2))       ,0                  ,0                    ,
   0                  ,0                    ,-(zFar+zNear)/(zFar-zNear)  , -1,
   0                  ,0                    ,-(2*zFar*zNear)/(zFar-zNear) ,0.0,
  ];
  Tick();
}
//let angle = 45.0; //Macierz transformacji świata - określenie położenia kamery
var angleZ = 0.0;
var angleY = 0.0;
var angleX = 0.0;

var KameraPositionZ = -15.0;
var KameraPositionX =  0.0;
var KameraPositionY =  -2.0;
```

```
//korpus
   var Object1PositionX = 0.0;
   var Object1PositionY = 5.0;
   var Object1PositionZ = 0.0;

   var Object1AngleZ = -90.0;

   var Object1Sizedx = 2.0;
   var Object1Sizedy = 0.7;
   var Object1Sizedz = 1.0;


//glowa
   var Object2PositionX = 0.0;
   var Object2PositionY = 5.9;
   var Object2PositionZ = 0.0;

   var Object2AngleZ = -90.0;

   var Object2Sizedx = 0.4;
   var Object2Sizedy = 0.4;
   var Object2Sizedz = 0.4;

//lewa reka
   var Object3PositionX = 0.0;
   var Object3PositionY = 3.5;
   var Object3PositionZ = 1.0;

   var Object3AngleZ = -90.0;

   var Object3Sizedx = 0.7;
   var Object3Sizedy = 0.3;
   var Object3Sizedz = 0.2;

//lewe przedramie
   var Object4PositionX = 0.70;
   var Object4PositionY = 0.0;
   var Object4PositionZ = 0.0;

   var Object4AngleZ = -90.0;

   var Object4Sizedx = 0.7;
   var Object4Sizedy = 0.2;
   var Object4Sizedz = 0.1;


//prawa reka
   var Object5PositionX = 0.0;
   var Object5PositionY = 3.5;
   var Object5PositionZ = -1.0;

   var Object5AngleZ = -90.0;

   var Object5Sizedx = 0.7;
   var Object5Sizedy = 0.3;
   var Object5Sizedz = 0.2;

//prawe przedramie
   var Object6PositionX = 0.70;
   var Object6PositionY = 0.0;
   var Object6PositionZ = 0.0;

   var Object6AngleZ = -90.0;

   var Object6Sizedx = 0.7;
   var Object6Sizedy = 0.2;
   var Object6Sizedz = 0.1;

//lewa noga
```

```
   var Object7PositionX = 0.0;
   var Object7PositionY = 1.0;
   var Object7PositionZ = 0.6;

   var Object7AngleZ = -90.0;

   var Object7Sizedx = 1.0;
   var Object7Sizedy = 0.4;
   var Object7Sizedz = 0.2;

//lewa lydka
   var Object8PositionX = 1.0;
   var Object8PositionY = 0.0;
   var Object8PositionZ = 0.0;

   var Object8AngleZ = 45.0;

   var Object8Sizedx = 1.0;
   var Object8Sizedy = 0.3;
   var Object8Sizedz = 0.1;

//prawa noga
   var Object9PositionX = 0.0;
   var Object9PositionY = 1.0;
   var Object9PositionZ = -0.6;

   var Object9AngleZ = -90.0;

   var Object9Sizedx = 1.0;
   var Object9Sizedy = 0.4;
   var Object9Sizedz = 0.2;

//prawa lydka
   var Object10PositionX = 1.0;
   var Object10PositionY = 0.0;
   var Object10PositionZ = 0.0;

   var Object10AngleZ = 45.0;

   var Object10Sizedx = 1.0;
   var Object10Sizedy = 0.3;
   var Object10Sizedz = 0.1;

//light
var LightSize = 0.2;
var LightPositionX = -10;
var LightPositionY = 10;
var LightPositionZ = 10;

var x=10;
var y=10;
var z=5;
var v=5;

function Tick()
{
 let uMMatrix0 = CreateIdentytyMatrix();
 let uMMatrix1 = CreateIdentytyMatrix();
 let uMMatrix2 = CreateIdentytyMatrix();
 let uMMatrix3 = CreateIdentytyMatrix();
 let uMMatrix4 = CreateIdentytyMatrix();
 let uMMatrix5 = CreateIdentytyMatrix();
 let uMMatrix6 = CreateIdentytyMatrix();
 let uMMatrix7 = CreateIdentytyMatrix();
 let uMMatrix8 = CreateIdentytyMatrix();
 let uMMatrix9 = CreateIdentytyMatrix();
 let uMMatrix10 = CreateIdentytyMatrix();

 let uVMatrix = CreateIdentytyMatrix();

 uVMatrix = MatrixMul(uVMatrix,CreateTranslationMatrix(KameraPositionX,KameraPositionY,KameraPositionZ));
```

```
  uVMatrix = MatrixMul(uVMatrix,CreateRotationXMatrix(angleX));
  uVMatrix = MatrixMul(uVMatrix,CreateRotationYMatrix(angleY));
  uVMatrix = MatrixMul(uVMatrix,CreateRotationZMatrix(angleZ));
 //light
  uMMatrix0 = MatrixMul(uMMatrix0,CreateScaleMatrix(LightSize,LightSize,LightSize));
  uMMatrix0 = MatrixMul(uMMatrix0,CreateTranslationMatrix(LightPositionX,LightPositionY,LightPositionZ));
 //korpus
  uMMatrix1 = MatrixMul(uMMatrix1,CreateScaleMatrix(Object1Sizedx,Object1Sizedy,Object1Sizedz));
  uMMatrix1 = MatrixMul(uMMatrix1,CreateTranslationMatrix(Object1Sizedx,0.0,0.0));
  uMMatrix1 = MatrixMul(uMMatrix1,CreateRotationZMatrix(Object1AngleZ));
  uMMatrix1 = MatrixMul(uMMatrix1,CreateTranslationMatrix(Object1PositionX,Object1PositionY,Object1PositionZ));
 //glowa
  uMMatrix2 = MatrixMul(uMMatrix2,CreateScaleMatrix(Object2Sizedx,Object2Sizedy,Object2Sizedz));
  uMMatrix2 = MatrixMul(uMMatrix2,CreateTranslationMatrix(Object2Sizedx,0.0,0.0));
  uMMatrix2 = MatrixMul(uMMatrix2,CreateRotationZMatrix(Object2AngleZ));
  uMMatrix2 = MatrixMul(uMMatrix2,CreateTranslationMatrix(Object2PositionX,Object2PositionY,Object2PositionZ));
 //lewa reka
  uMMatrix3 = MatrixMul(uMMatrix3,CreateScaleMatrix(Object3Sizedx,Object3Sizedy,Object3Sizedz));
  uMMatrix3 = MatrixMul(uMMatrix3,CreateTranslationMatrix(Object3Sizedx,0.0,0.0));
  uMMatrix3 = MatrixMul(uMMatrix3,CreateRotationZMatrix(Object3AngleZ));
  uMMatrix3 = MatrixMul(uMMatrix3,CreateTranslationMatrix(Object3PositionX,Object3PositionY,Object3PositionZ));
 //lewe przedramie
  uMMatrix4 = MatrixMul(uMMatrix4,CreateScaleMatrix(Object4Sizedx,Object4Sizedy,Object4Sizedz));
  uMMatrix4 = MatrixMul(uMMatrix4,CreateTranslationMatrix(Object4Sizedx,0.0,0.0));
  uMMatrix4 = MatrixMul(uMMatrix4,CreateRotationZMatrix(Object4AngleZ));
  uMMatrix4 = MatrixMul(uMMatrix4,CreateTranslationMatrix(Object4PositionX,Object4PositionY,Object4PositionZ));

  uMMatrix4 = MatrixMul(uMMatrix4,CreateTranslationMatrix(Object3Sizedx,0.0,0.0));
  uMMatrix4 = MatrixMul(uMMatrix4,CreateRotationZMatrix(Object3AngleZ));
  uMMatrix4 = MatrixMul(uMMatrix4,CreateTranslationMatrix(Object3PositionX,Object3PositionY,Object3PositionZ));
 //prawa reka
  uMMatrix5 = MatrixMul(uMMatrix5,CreateScaleMatrix(Object5Sizedx,Object5Sizedy,Object5Sizedz));
  uMMatrix5 = MatrixMul(uMMatrix5,CreateTranslationMatrix(Object5Sizedx,0.0,0.0));
  uMMatrix5 = MatrixMul(uMMatrix5,CreateRotationZMatrix(Object5AngleZ));
  uMMatrix5 = MatrixMul(uMMatrix5,CreateTranslationMatrix(Object5PositionX,Object5PositionY,Object5PositionZ));
 //prawe przedramie
  uMMatrix6 = MatrixMul(uMMatrix6,CreateScaleMatrix(Object6Sizedx,Object6Sizedy,Object6Sizedz));
  uMMatrix6 = MatrixMul(uMMatrix6,CreateTranslationMatrix(Object6Sizedx,0.0,0.0));
  uMMatrix6 = MatrixMul(uMMatrix6,CreateRotationZMatrix(Object6AngleZ));
  uMMatrix6 = MatrixMul(uMMatrix6,CreateTranslationMatrix(Object6PositionX,Object6PositionY,Object6PositionZ));

  uMMatrix6 = MatrixMul(uMMatrix6,CreateTranslationMatrix(Object5Sizedx,0.0,0.0));
  uMMatrix6 = MatrixMul(uMMatrix6,CreateRotationZMatrix(Object5AngleZ));
  uMMatrix6 = MatrixMul(uMMatrix6,CreateTranslationMatrix(Object5PositionX,Object5PositionY,Object5PositionZ));
 //lewa noga
  uMMatrix7 = MatrixMul(uMMatrix7,CreateScaleMatrix(Object7Sizedx,Object7Sizedy,Object7Sizedz));
  uMMatrix7 = MatrixMul(uMMatrix7,CreateTranslationMatrix(Object7Sizedx,0.0,0.0));
  uMMatrix7 = MatrixMul(uMMatrix7,CreateRotationZMatrix(Object7AngleZ));
  uMMatrix7 = MatrixMul(uMMatrix7,CreateTranslationMatrix(Object7PositionX,Object7PositionY,Object7PositionZ));
 //lewa lydka
  uMMatrix8 = MatrixMul(uMMatrix8,CreateScaleMatrix(Object8Sizedx,Object8Sizedy,Object8Sizedz));
  uMMatrix8 = MatrixMul(uMMatrix8,CreateTranslationMatrix(Object8Sizedx,0.0,0.0));
  uMMatrix8 = MatrixMul(uMMatrix8,CreateRotationZMatrix(Object8AngleZ));
  uMMatrix8 = MatrixMul(uMMatrix8,CreateTranslationMatrix(Object8PositionX,Object8PositionY,Object8PositionZ));

  uMMatrix8 = MatrixMul(uMMatrix8,CreateTranslationMatrix(Object7Sizedx,0.0,0.0));
  uMMatrix8 = MatrixMul(uMMatrix8,CreateRotationZMatrix(Object7AngleZ));
  uMMatrix8 = MatrixMul(uMMatrix8,CreateTranslationMatrix(Object7PositionX,Object7PositionY,Object7PositionZ));
 //prawa noga
  uMMatrix9 = MatrixMul(uMMatrix9,CreateScaleMatrix(Object9Sizedx,Object9Sizedy,Object9Sizedz));
  uMMatrix9 = MatrixMul(uMMatrix9,CreateTranslationMatrix(Object9Sizedx,0.0,0.0));
  uMMatrix9 = MatrixMul(uMMatrix9,CreateRotationZMatrix(Object9AngleZ));
  uMMatrix9 = MatrixMul(uMMatrix9,CreateTranslationMatrix(Object9PositionX,Object9PositionY,Object9PositionZ));
 //prawa lydka
  uMMatrix10 = MatrixMul(uMMatrix10,CreateScaleMatrix(Object10Sizedx,Object10Sizedy,Object10Sizedz));
  uMMatrix10 = MatrixMul(uMMatrix10,CreateTranslationMatrix(Object10Sizedx,0.0,0.0));
  uMMatrix10 = MatrixMul(uMMatrix10,CreateRotationZMatrix(Object10AngleZ));
  uMMatrix10 = MatrixMul(uMMatrix10,CreateTranslationMatrix(Object10PositionX,Object10PositionY,Object10PositionZ));

  uMMatrix10 = MatrixMul(uMMatrix10,CreateTranslationMatrix(Object9Sizedx,0.0,0.0));
  uMMatrix10 = MatrixMul(uMMatrix10,CreateRotationZMatrix(Object9AngleZ));
```

```
 uMMatrix10 = MatrixMul(uMMatrix10,CreateTranslationMatrix(Object9PositionX,Object9PositionY,Object9PositionZ));
 //alert(uPMatrix);

 //Render Scene
 gl.viewport(0, 0, gl.viewportWidth, gl.viewportHeight);
 gl.clearColor(1.0,0.0,0.0,1.0); //Wyczyszczenie obrazu kolorem czerwonym
 gl.clearDepth(1.0);           //Wyczyścienie bufora głebi najdalszym planem
 gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
 gl.useProgram(shaderProgram)   //Użycie przygotowanego programu shaderowego

 gl.enable(gl.DEPTH_TEST);         // Włączenie testu głębi - obiekty bliższe mają przykrywać obiekty dalsze
 gl.depthFunc(gl.LEQUAL);          //

 gl.uniformMatrix4fv(gl.getUniformLocation(shaderProgram, "uPMatrix"), false, new Float32Array(uPMatrix)); //Wgranie macierzy kamery do pamięci karty
graficznej
 gl.uniformMatrix4fv(gl.getUniformLocation(shaderProgram, "uVMatrix"), false, new Float32Array(uVMatrix));
 gl.uniform1f(gl.getUniformLocation(shaderProgram, "uNormalMul"),1.0);


 gl.uniformMatrix4fv(gl.getUniformLocation(shaderProgram, "uMMatrix"), false, new Float32Array(uMMatrix1));
 gl.uniformMatrix4fv(gl.getUniformLocation(shaderProgram, "uInvMMatrix"), false, new Float32Array(MatrixTransposeInverse(uMMatrix1)));

 gl.enableVertexAttribArray(gl.getAttribLocation(shaderProgram, "aVertexPosition"));  //Przekazanie położenia
 gl.bindBuffer(gl.ARRAY_BUFFER, vertexPositionBuffer);
 gl.vertexAttribPointer(gl.getAttribLocation(shaderProgram, "aVertexPosition"), vertexPositionBuffer.itemSize, gl.FLOAT, false, 0, 0);

 gl.enableVertexAttribArray(gl.getAttribLocation(shaderProgram, "aVertexNormal"));  //Przekazywanie wektorów normalnych
 gl.bindBuffer(gl.ARRAY_BUFFER, vertexNormalBuffer);
 gl.vertexAttribPointer(gl.getAttribLocation(shaderProgram, "aVertexNormal"), vertexNormalBuffer.itemSize, gl.FLOAT, false, 0, 0);

 gl.uniform3f(gl.getUniformLocation(shaderProgram, "uLightPosition"),LightPositionX,LightPositionY,LightPositionZ);
 gl.uniform3f(gl.getUniformLocation(shaderProgram, "uColor"),1.0,1.0,1.0);
 gl.drawArrays(gl.TRIANGLES, 0, vertexPositionBuffer.numItems*vertexPositionBuffer.itemSize); //Faktyczne wywołanie rendrowania

 gl.uniformMatrix4fv(gl.getUniformLocation(shaderProgram, "uMMatrix"), false, new Float32Array(uMMatrix0));
 gl.uniformMatrix4fv(gl.getUniformLocation(shaderProgram, "uInvMMatrix"), false, new Float32Array(MatrixTransposeInverse(uMMatrix0)));
 gl.uniform1f(gl.getUniformLocation(shaderProgram, "uNormalMul"),1.0);
 gl.drawArrays(gl.TRIANGLES, 0, vertexPositionBuffer.numItems*vertexPositionBuffer.itemSize); //Faktyczne wywołanie rendrowania

 gl.uniformMatrix4fv(gl.getUniformLocation(shaderProgram, "uMMatrix"), false, new Float32Array(uMMatrix2));
 gl.uniformMatrix4fv(gl.getUniformLocation(shaderProgram, "uInvMMatrix"), false, new Float32Array(MatrixTransposeInverse(uMMatrix2)));
 gl.uniform3f(gl.getUniformLocation(shaderProgram, "uColor"),0.1,0.1,0.5);
 gl.drawArrays(gl.TRIANGLES, 0, vertexPositionBuffer.numItems*vertexPositionBuffer.itemSize); //Faktyczne wywołanie rendrowania

 gl.uniformMatrix4fv(gl.getUniformLocation(shaderProgram, "uMMatrix"), false, new Float32Array(uMMatrix3));
 gl.uniformMatrix4fv(gl.getUniformLocation(shaderProgram, "uInvMMatrix"), false, new Float32Array(MatrixTransposeInverse(uMMatrix3)));
 gl.uniform3f(gl.getUniformLocation(shaderProgram, "uColor"),0.1,0.5,0.1);
 gl.drawArrays(gl.TRIANGLES, 0, vertexPositionBuffer.numItems*vertexPositionBuffer.itemSize); //Faktyczne wywołanie rendrowania

 gl.uniformMatrix4fv(gl.getUniformLocation(shaderProgram, "uMMatrix"), false, new Float32Array(uMMatrix4));
 gl.uniformMatrix4fv(gl.getUniformLocation(shaderProgram, "uInvMMatrix"), false, new Float32Array(MatrixTransposeInverse(uMMatrix4)));
 gl.uniform3f(gl.getUniformLocation(shaderProgram, "uColor"),0.5,0.0,0.0);
 gl.drawArrays(gl.TRIANGLES, 0, vertexPositionBuffer.numItems*vertexPositionBuffer.itemSize); //Faktyczne wywołanie rendrowania

 gl.uniformMatrix4fv(gl.getUniformLocation(shaderProgram, "uMMatrix"), false, new Float32Array(uMMatrix5));
 gl.uniformMatrix4fv(gl.getUniformLocation(shaderProgram, "uInvMMatrix"), false, new Float32Array(MatrixTransposeInverse(uMMatrix5)));
 gl.uniform3f(gl.getUniformLocation(shaderProgram, "uColor"),1.0,0.0,1.0);
 gl.drawArrays(gl.TRIANGLES, 0, vertexPositionBuffer.numItems*vertexPositionBuffer.itemSize); //Faktyczne wywołanie rendrowania

 gl.uniformMatrix4fv(gl.getUniformLocation(shaderProgram, "uMMatrix"), false, new Float32Array(uMMatrix6));
 gl.uniformMatrix4fv(gl.getUniformLocation(shaderProgram, "uInvMMatrix"), false, new Float32Array(MatrixTransposeInverse(uMMatrix6)));
 gl.uniform3f(gl.getUniformLocation(shaderProgram, "uColor"),1.0,1.0,0.0);
 gl.drawArrays(gl.TRIANGLES, 0, vertexPositionBuffer.numItems*vertexPositionBuffer.itemSize); //Faktyczne wywołanie rendrowania

 gl.uniformMatrix4fv(gl.getUniformLocation(shaderProgram, "uMMatrix"), false, new Float32Array(uMMatrix7));
 gl.uniformMatrix4fv(gl.getUniformLocation(shaderProgram, "uInvMMatrix"), false, new Float32Array(MatrixTransposeInverse(uMMatrix7)));
 gl.uniform3f(gl.getUniformLocation(shaderProgram, "uColor"),0.0,1.0,1.0);
 gl.drawArrays(gl.TRIANGLES, 0, vertexPositionBuffer.numItems*vertexPositionBuffer.itemSize); //Faktyczne wywołanie rendrowania

 gl.uniformMatrix4fv(gl.getUniformLocation(shaderProgram, "uMMatrix"), false, new Float32Array(uMMatrix8));
 gl.uniformMatrix4fv(gl.getUniformLocation(shaderProgram, "uInvMMatrix"), false, new Float32Array(MatrixTransposeInverse(uMMatrix8)));
```

```
 gl.uniform3f(gl.getUniformLocation(shaderProgram, "uColor"),1.0,0.0,0.0);
 gl.drawArrays(gl.TRIANGLES, 0, vertexPositionBuffer.numItems*vertexPositionBuffer.itemSize); //Faktyczne wywołanie rendrowania

 gl.uniformMatrix4fv(gl.getUniformLocation(shaderProgram, "uMMatrix"), false, new Float32Array(uMMatrix9));
 gl.uniformMatrix4fv(gl.getUniformLocation(shaderProgram, "uInvMMatrix"), false, new Float32Array(MatrixTransposeInverse(uMMatrix9)));
 gl.uniform3f(gl.getUniformLocation(shaderProgram, "uColor"),0.0,1.0,0.0);
 gl.drawArrays(gl.TRIANGLES, 0, vertexPositionBuffer.numItems*vertexPositionBuffer.itemSize); //Faktyczne wywołanie rendrowania

 gl.uniformMatrix4fv(gl.getUniformLocation(shaderProgram, "uMMatrix"), false, new Float32Array(uMMatrix10));
 gl.uniformMatrix4fv(gl.getUniformLocation(shaderProgram, "uInvMMatrix"), false, new Float32Array(MatrixTransposeInverse(uMMatrix10)));
 gl.uniform3f(gl.getUniformLocation(shaderProgram, "uColor"),0.0,0.0,1.0);
 gl.drawArrays(gl.TRIANGLES, 0, vertexPositionBuffer.numItems*vertexPositionBuffer.itemSize); //Faktyczne wywołanie rendrowania

 setTimeout(Tick,100);

 //cialo do przodu
 Object1PositionX -= 0.1;
 Object2PositionX -= 0.1;
 Object3PositionX -= 0.1;
 Object5PositionX -= 0.1;
 Object7PositionX -= 0.1;
 Object9PositionX -= 0.1;


  //ruch czesci ciala
//rece
if(Object3AngleZ==-50){
    x=-10
    z=-5;
  }
  if(Object3AngleZ==-140)
  {
    x=10;
    z=5;
  }
  Object3AngleZ+=x;
  Object5AngleZ-=x;

  Object4AngleZ+=z;
  Object6AngleZ-=z;
//nogi
  if(Object9AngleZ==-50){
    y=-10
    v=-5;
  }
  if(Object9AngleZ==-140)
  {
    y=10;
    v=5;

  }
  Object7AngleZ-=y;
  Object9AngleZ+=y;

  Object8AngleZ-=v;
  Object10AngleZ+=v;


}
function handlekeydown(e)
{
 // Q W E A S D
 if(e.keyCode==87) angleX=angleX+1.0; //W
 if(e.keyCode==83) angleX=angleX-1.0; //S
 if(e.keyCode==68) angleY=angleY+1.0;
 if(e.keyCode==65) angleY=angleY-1.0;
 if(e.keyCode==81) angleZ=angleZ+1.0;
 if(e.keyCode==69) angleZ=angleZ-1.0;

 //U I O J K L
```

```
 if(e.keyCode==76) LightPositionX=LightPositionX+0.1;
 if(e.keyCode==74) LightPositionX=LightPositionX-0.1;
 if(e.keyCode==73) LightPositionY=LightPositionY+0.1;
 if(e.keyCode==75) LightPositionY=LightPositionY-0.1;
 if(e.keyCode==85) LightPositionZ=LightPositionZ+0.1;
 if(e.keyCode==79) LightPositionZ=LightPositionZ-0.1;

}
</script>
</head>
<body onload="startGL()" onkeydown="handlekeydown(event)">
<canvas id="canvas3D" width="640" height="480" style="border: solid black 1px"></canvas>
</body>
</html>
```