Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«Владимирский государственный университет

имени Александра Григорьевича и Николая Григорьевича Столетовых»

(ВлГУ)

Кафедра информационных систем и программной инженерии

Лабораторная работа № 7

по дисциплине "Программирование графических приложений"

ТЕМА РАБОТЫ:

Двухмерные геометрические преобразования

Выполнил:

студент гр. ПРИм-124

Парахин К.В.

Принял:

Жигалов И.Е.

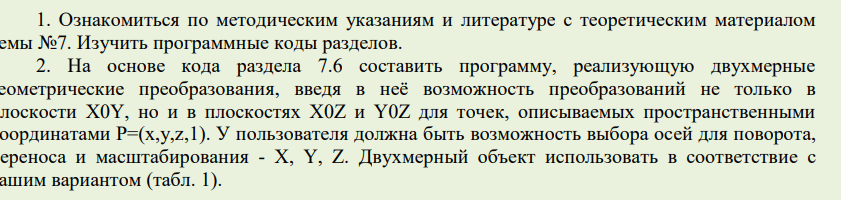
Владимир 2024 г.

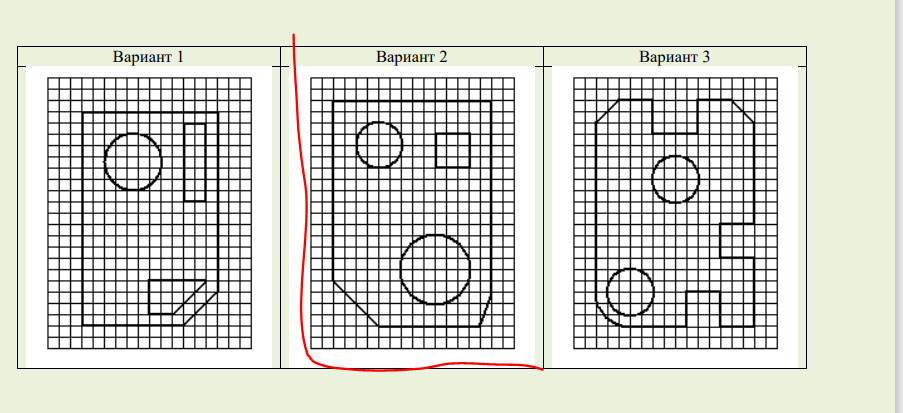
Цель работы:

Изучение способов реализации двухмерных геометрических преобразований при формировании графических сцен с использованием WebGL

Выполнение работы:

Индивидуальный вариант 2 (14 % 12 = 2)





Листинг программы:

   var vertices1;

    var vertexBuffer1;

    var vertices2 = [];

    var vertexBuffer2;

    var vertices3 = [];

    var vertexBuffer3;

    var vertices4 = [];

    var vertexBuffer4;

    function initBuffers() {

        vertices1 = [

            10, 30, 0,

            10, 110, 0,

            80, 110, 0,

            85, 25, 0,

            75, 10, 0,

            30, 10, 0,

            10, 30, 0

        ];

        var radiusLarge = 15;

        var pi = Math.PI;

        var step = 0.01;

        var centerXLarge = 55;

        var centerYLarge = 35;

        var startPoint = pi/2;

        for (i = 0; i <= 200; i+=1)

        {

            var x = Math.sin(startPoint - pi \* step \* i) \* radiusLarge + centerXLarge;

            var y = Math.cos(startPoint - pi \* step \* i) \* radiusLarge + centerYLarge;

            vertices2.push(x);

            vertices2.push(y);

            vertices2.push(0);

        }

        var radiusLarge = 10;

        var centerXLarge = 30;

        var centerYLarge = 90;

        var startPoint = pi/2;

        for (i = 0; i <= 200; i+=1)

        {

            var x = Math.sin(startPoint - pi \* step \* i) \* radiusLarge + centerXLarge;

            var y = Math.cos(startPoint - pi \* step \* i) \* radiusLarge + centerYLarge;

            vertices3.push(x);

            vertices3.push(y);

            vertices3.push(0);

        }

        vertices4.push(60);

        vertices4.push(80);

        vertices4.push(0);

        vertices4.push(60);

        vertices4.push(95);

        vertices4.push(0);

        vertices4.push(75);

        vertices4.push(95);

        vertices4.push(0);

        vertices4.push(75);

        vertices4.push(80);

        vertices4.push(0);

        vertices4.push(60);

        vertices4.push(80);

        vertices4.push(0);

        for (var i = 0; i < vertices1.length; i++)

        {

            if (i % 3 != 2)

            {

                vertices1[i] -= 50;

                vertices1[i] /= 100;

            }

        }

        for (var i = 0; i < vertices2.length; i++)

        {

            if (i % 3 != 2)

            {

                vertices2[i] -= 50;

                vertices2[i] /= 100;

            }

        }

        for (var i = 0; i < vertices3.length; i++)

        {

            if (i % 3 != 2)

            {

                vertices3[i] -= 50;

                vertices3[i] /= 100;

            }

        }

        for (var i = 0; i < vertices4.length; i++)

        {

            if (i % 3 != 2)

            {

                vertices4[i] -= 50;

                vertices4[i] /= 100;

            }

        }

        vertexBuffer1 = gl.createBuffer();

        gl.bindBuffer(gl.ARRAY\_BUFFER, vertexBuffer1);

        gl.bufferData(gl.ARRAY\_BUFFER, new Float32Array(vertices1), gl.STATIC\_DRAW);

        vertexBuffer1.itemSize = 3;

        vertexBuffer1.numItems = vertices1.length / 3;

        vertexBuffer2 = gl.createBuffer();

        gl.bindBuffer(gl.ARRAY\_BUFFER, vertexBuffer2);

        gl.bufferData(gl.ARRAY\_BUFFER, new Float32Array(vertices2), gl.STATIC\_DRAW);

        vertexBuffer2.itemSize = 3;

        vertexBuffer2.numItems = vertices2.length / 3;

        vertexBuffer3 = gl.createBuffer();

        gl.bindBuffer(gl.ARRAY\_BUFFER, vertexBuffer3);

        gl.bufferData(gl.ARRAY\_BUFFER, new Float32Array(vertices3), gl.STATIC\_DRAW);

        vertexBuffer3.itemSize = 3;

        vertexBuffer3.numItems = vertices3.length / 3;

        vertexBuffer4 = gl.createBuffer();

        gl.bindBuffer(gl.ARRAY\_BUFFER, vertexBuffer4);

        gl.bufferData(gl.ARRAY\_BUFFER, new Float32Array(vertices4), gl.STATIC\_DRAW);

        vertexBuffer4.itemSize = 3;

        vertexBuffer4.numItems = vertices4.length / 3;

    }

    function drawScene() {

        gl.viewport(0, 0, gl.viewportWidth, gl.viewportHeight);

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

        gl.bindBuffer(gl.ARRAY\_BUFFER, vertexBuffer1);

        gl.vertexAttribPointer(shaderProgram.vertexPositionAttribute, vertexBuffer1.itemSize, gl.FLOAT, false, 0, 0);

        gl.drawArrays(gl.LINE\_STRIP, 0, vertexBuffer1.numItems);

        gl.bindBuffer(gl.ARRAY\_BUFFER, vertexBuffer2);

        gl.vertexAttribPointer(shaderProgram.vertexPositionAttribute, vertexBuffer2.itemSize, gl.FLOAT, false, 0, 0);

        gl.drawArrays(gl.LINE\_STRIP, 0, vertexBuffer2.numItems);

        gl.bindBuffer(gl.ARRAY\_BUFFER, vertexBuffer3);

        gl.vertexAttribPointer(shaderProgram.vertexPositionAttribute, vertexBuffer3.itemSize, gl.FLOAT, false, 0, 0);

        gl.drawArrays(gl.LINE\_STRIP, 0, vertexBuffer3.numItems);

        gl.bindBuffer(gl.ARRAY\_BUFFER, vertexBuffer4);

        gl.vertexAttribPointer(shaderProgram.vertexPositionAttribute, vertexBuffer4.itemSize, gl.FLOAT, false, 0, 0);

        gl.drawArrays(gl.LINE\_STRIP, 0, vertexBuffer4.numItems);

    }

Измененные матрицы:

    // ������� ��������

    function createTranslationMatrix(t1, t2) {

        var selectedAxes = document.getElementById("axesSelect").value;

        if (selectedAxes == "XOY"){

            return [

                1, 0, 0, 0,

                0, 1, 0, 0,

                0, 0, 1, 1,

                t1, t2, 0, 1

            ];

        }

        else if (selectedAxes == "YOZ"){

            return [

                1, 0, 0, 0,

                0, 1, 0, 0,

                0, 0, 1, 1,

                0, t1, t2, 1

            ];

        }

        else if (selectedAxes == "XOZ"){

            return [

                1, 0, 0, 0,

                0, 1, 0, 0,

                0, 0, 1, 1,

                t1, 0, t2, 1

            ];

        }

    }

    // ������� ���������������

    function createScaleMatrix(s1, s2) {

        var selectedAxes = document.getElementById("axesSelect").value;

        if (selectedAxes == "XOY"){

            return [

                s1, 0,  0, 0,

                0,  s2, 0, 0,

                0,  0,  1, 0,

                0,  0,  0, 1

            ];

        }

        else if (selectedAxes == "YOZ"){

            return [

                1, 0,  0, 0,

                0,  s1, 0, 0,

                0,  0,  s2, 0,

                0,  0,  0, 1

            ];

        }

        else if (selectedAxes == "XOZ"){

            return [

                s1, 0,  0, 0,

                0,  1, 0, 0,

                0,  0,  s2, 0,

                0,  0,  0, 1

            ];

        }

    }

    // ������� �������� �� ��� Z

    function createRotationMatrix(angle) {

        var angleRad = angle \* Math.PI / 180;

        var c = Math.cos(angleRad);

        var s = Math.sin(angleRad);

        var selectedAxes = document.getElementById("axesSelect").value;

        if (selectedAxes == "XOY"){

            return [

                c, -s, 0, 0,

                s,  c, 0, 0,

                0,  0, 1, 0,

                0,  0, 0, 1

            ];

        }

        else if (selectedAxes == "YOZ"){

            return [

                1,  0,  0, 0,

                0,  c, -s, 0,

                0,  s,  c, 0,

                0,  0,  0, 1

            ];

        }

        else if (selectedAxes == "XOZ"){

            return [

                c,  0, s, 0,

                0,  1, 0, 0,

                -s, 0, c, 0,

                0,  0, 0, 1

            ];

        }

    }

Меню с выбором осей:

<body onload="webGLStart();">

    </br>

    <select id="axesSelect" onchange="drawScene()">

      <option value="XOY" selected>On XOY</option>

      <option value="YOZ">On YOZ</option>

      <option value="XOZ">On XOZ</option>

    </select>

    </br>

    <canvas id="canvas" width="500" height="500"></canvas>

</body>

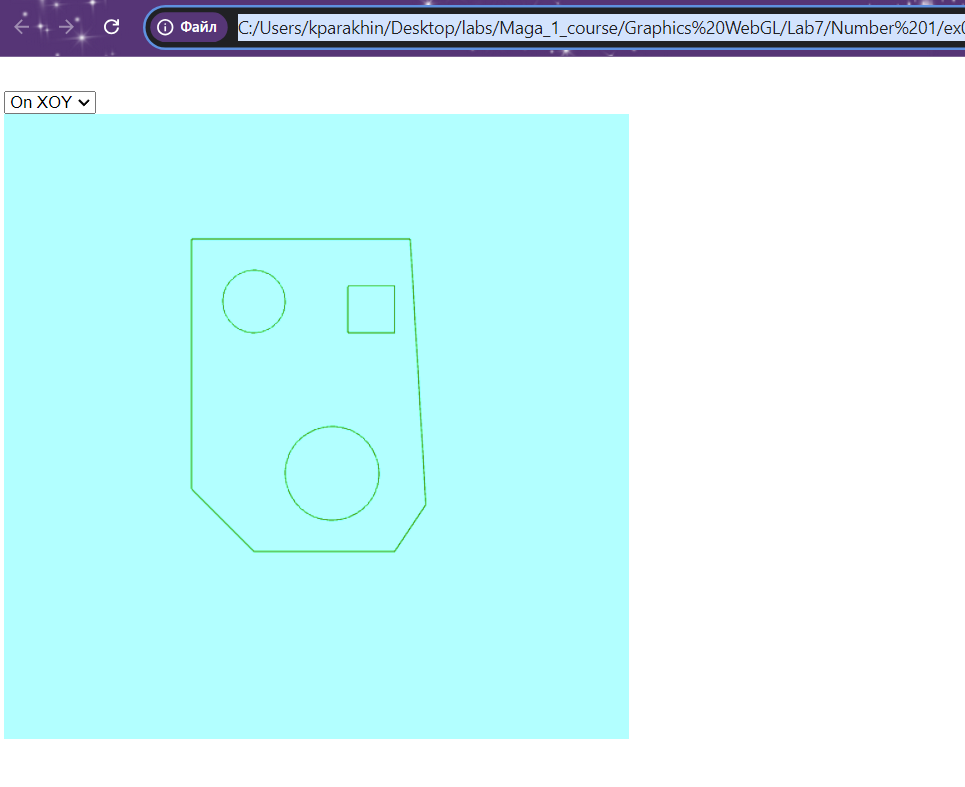


Рисунок 1. Построенное изображение на единичной матрице

В результате доступен выбор осей (XOY, YOZ и XOZ) – относительно которых можно перемещать, масштабировать и поворачивать изображенное, занесенное в буфер – относительно выбранного в меню параметра берется конкретная матрица поворота, масштабирования и поворота.

Вывод

В результате выполнения работы я провел изучение способов реализации двухмерных геометрических преобразований при формировании графических сцен с использованием WebGL