CSE 3114 / CSE 3219

COMPUTER GRAPHICS

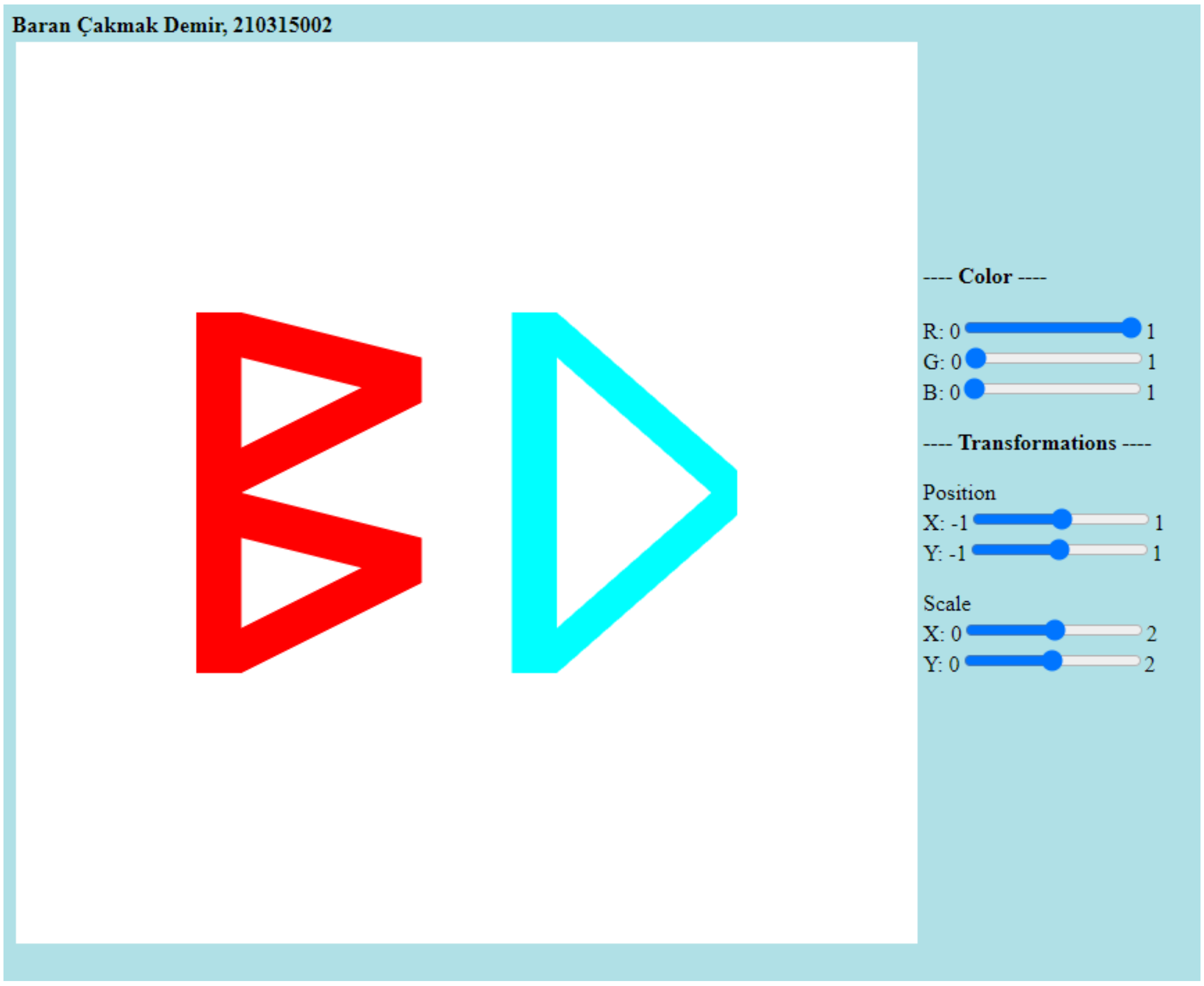
SPRING 2024

Midterm Assignment Report

Baran Çakmak Demir – 210315002

Submission Date: 5 April 2024

Program Output



Reflections

In this project, the most difficult part was the coordinate system. After completing the coordinate system it got easier. Also the color part was hard due to the contrasting colors.

Source Code

HTML Codes:

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;charset=utf-8" >
<title>Midterm Exam</title>

<script id="vertex-shader" type="x-shader/x-vertex">
attribute vec4 vPosition;
```

```
// TODO: get required variables


uniform float x;
uniform float y;

uniform float scaleX;
uniform float scaleY;


void main()
{
    // TODO: calculate gl_Position appropriately


    gl_Position.x = vPosition.x * scaleX + x;
    gl_Position.y = vPosition.y * scaleY + y;
    gl_Position.z = 0.0;
    gl_Position.w = 1.0;


}
</script>

<script id="fragment-shader" type="x-shader/x-fragment">

precision mediump float;

// TODO: get required variables

uniform vec4 color;

void main()
{
    // TODO: assign color

    gl_FragColor = color;
}
</script>

<script type="text/javascript" src="../Common/webgl-utils.js"></script>
<script type="text/javascript" src="../Common/initShaders.js"></script>
<script type="text/javascript" src="../Common/MV.js"></script>
<script type="text/javascript" src="midterm.js"></script>
</head>

<body style="background-color:powderblue;">

<div>
```

```html
<b>Baran Çakmak Demir, 210315002</b>
</div>
<table>
    <td>
    <canvas id="gl-canvas" width="650" height="650">
        Oops ... your browser doesn't support the HTML5 canvas element
    </canvas>
    </td>
    <td>
        <div> <strong>---- Color ----</strong> </div><br>
        <div>
        R: 0<input id="redSlider" type="range"
         min="0" max="1" step="0.05" value="1" />1
        </div>
        <div>
        G: 0<input id="greenSlider" type="range"
         min="0" max="1" step="0.05" value="0" />1
        </div>
        <div>
        B: 0<input id="blueSlider" type="range"
         min="0" max="1" step="0.05" value="0" />1
        </div>
        <br>

        <div> <strong>---- Transformations ----</strong> </div><br>

        <div>Position</div>
        <div>X: -1<input id="posX" type="range"
         min="-1" max="1" step="0.05" value="0" />1</div>
        <div>Y: -1<input id="posY" type="range"
         min="-1" max="1" step="0.05" value="0" />1</div><br>

        <div>Scale</div>
        <div>X: 0<input id="scaleX" type="range"
         min="0" max="2" step="0.05" value="1.0" />2</div>
        <div>Y: 0<input id="scaleY" type="range"
         min="0" max="2" step="0.05" value="1.0" />2</div><br>

        <br>
    </td>

</table>
<div>
</body>
</html>
```

JAVASCRIPT Codes:

```
var canvas;
var gl;
var vPosition;
var program;

var color;

var red = 1.0;
var green = 0.0;
var blue = 0.0;

var x = 0.0;
var y = 0.0;

var scaleX = 1.0;
var scaleY = 1.0;


// TODO: define any global variables you need

window.onload = function init() {
    canvas = document.getElementById("gl-canvas");

    gl = WebGLUtils.setupWebGL(canvas);
    if (!gl) { alert("WebGL isn't available"); }

    //  Configure WebGL
    gl.viewport(0, 0, canvas.width, canvas.height);
    gl.clearColor(1.0, 1.0, 1.0, 1.0);

    //  Load shaders and initialize attribute buffers
    program = initShaders(gl, "vertex-shader", "fragment-shader");
    gl.useProgram(program);

    // Create geometry data

    // TODO: create vertex coordinates for your initial letters instead of
these vertices


    // for B;
    letter1vertices = [vec2(-0.6, -0.4),
    vec2(-0.5, -0.4),
    vec2(-0.6, 0.4),
    vec2(-0.5, 0.4)];

    letter4vertices = [vec2(-0.5, -0.3),
    vec2(-0.5, -0.4),
    vec2(-0.1, -0.1),
    vec2(-0.1, -0.2)];

    letter6vertices = [vec2(-0.5, 0.0),
    vec2(-0.5, -0.1),
    vec2(-0.1, -0.1),
    vec2(-0.1, -0.2)];


    letter7vertices = [vec2(-0.5, 0.4),
    vec2(-0.5, 0.3),
    vec2(-0.1, 0.3),
    vec2(-0.1, 0.2)];

    letter8vertices = [vec2(-0.5, 0.1),
    vec2(-0.5, 0.0),
    vec2(-0.1, 0.3),
    vec2(-0.1, 0.2)];

    // for D;
    letter2vertices = [vec2(0.1, -0.4),
    vec2(0.2, -0.4),
    vec2(0.1, 0.4),
    vec2(0.2, 0.4)];

    letter3vertices = [vec2(0.2, 0.4),
    vec2(0.2, 0.3),
    vec2(0.6, 0.05),
    vec2(0.6, -0.05)];


    letter5vertices = [vec2(0.2, -0.3),
    vec2(0.2, -0.4),
    vec2(0.6, 0.05),
    vec2(0.6, -0.05)];


    // Load the data into the GPU
    buffer1 = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, buffer1);
    gl.bufferData(gl.ARRAY_BUFFER, flatten(letter1vertices), gl.STATIC_DRAW);

    buffer2 = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, buffer2);
    gl.bufferData(gl.ARRAY_BUFFER, flatten(letter2vertices), gl.STATIC_DRAW);

    buffer3 = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, buffer3);
    gl.bufferData(gl.ARRAY_BUFFER, flatten(letter3vertices), gl.STATIC_DRAW);

    buffer4 = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, buffer4);
    gl.bufferData(gl.ARRAY_BUFFER, flatten(letter4vertices), gl.STATIC_DRAW);


    buffer5 = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, buffer5);
    gl.bufferData(gl.ARRAY_BUFFER, flatten(letter5vertices), gl.STATIC_DRAW);

    buffer6 = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, buffer6);
    gl.bufferData(gl.ARRAY_BUFFER, flatten(letter6vertices), gl.STATIC_DRAW);


    buffer7 = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, buffer7);
    gl.bufferData(gl.ARRAY_BUFFER, flatten(letter7vertices), gl.STATIC_DRAW);

    buffer8 = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, buffer8);
    gl.bufferData(gl.ARRAY_BUFFER, flatten(letter8vertices), gl.STATIC_DRAW);



    xLoc = gl.getUniformLocation(program, "x");
    yLoc = gl.getUniformLocation(program, "y");

    scaleXLoc = gl.getUniformLocation(program, "scaleX");
    scaleYLoc = gl.getUniformLocation(program, "scaleY");

    colorLoc = gl.getUniformLocation(program, "color");


    document.getElementById("posX").oninput = function (event) {
        //TODO: fill here to adjust translation according to slider value
        x = event.target.value;
    };
    document.getElementById("posY").oninput = function (event) {
        //TODO: fill here to adjust translation according to slider value
        y = event.target.value;
    };
    document.getElementById("scaleX").oninput = function (event) {
        //TODO: fill here to adjust scale according to slider value
        scaleX = event.target.value;
    };
    document.getElementById("scaleY").oninput = function (event) {
        //TODO: fill here to adjust scale according to slider value
        scaleY = event.target.value;
    };
    document.getElementById("redSlider").oninput = function (event) {
        //TODO: fill here to adjust color according to slider value

        red = event.target.value;
    };
    document.getElementById("greenSlider").oninput = function (event) {
        //TODO: fill here to adjust color according to slider value

        green = event.target.value;
    };
    document.getElementById("blueSlider").oninput = function (event) {
        //TODO: fill here to adjust color according to slider value

        blue = event.target.value;
    };

    render();
};


function render() {
    gl.clear(gl.COLOR_BUFFER_BIT);

    // TODO: Send necessary uniform variables to shader and
    // perform draw calls for drawing letters

    // bind vertex buffer and associate position data with shader variables
    gl.bindBuffer(gl.ARRAY_BUFFER, buffer1);
    gl.vertexAttribPointer(vPosition, 2, gl.FLOAT, false, 0, 0);
    gl.enableVertexAttribArray(vPosition);
    // draw triangle
    color = vec4(red, green, blue, 1.0);
    gl.uniform4fv(colorLoc, color);
    gl.drawArrays(gl.TRIANGLE_STRIP, 0, letter1vertices.length);

    // bind vertex buffer and associate position data with shader variables
    gl.bindBuffer(gl.ARRAY_BUFFER, buffer2);
    gl.vertexAttribPointer(vPosition, 2, gl.FLOAT, false, 0, 0);
    gl.enableVertexAttribArray(vPosition);
    // draw rectangle
    color = vec4(1 - red, 1 - green, 1 - blue, 1.0);
    gl.uniform4fv(colorLoc, color);
    gl.drawArrays(gl.TRIANGLE_STRIP, 0, letter2vertices.length);

    // bind vertex buffer and associate position data with shader variables
    gl.bindBuffer(gl.ARRAY_BUFFER, buffer3);
    gl.vertexAttribPointer(vPosition, 2, gl.FLOAT, false, 0, 0);
    gl.enableVertexAttribArray(vPosition);
    // draw rectangle
    color = vec4(1 - red, 1 - green, 1 - blue, 1.0);
    gl.uniform4fv(colorLoc, color);
    gl.drawArrays(gl.TRIANGLE_STRIP, 0, letter3vertices.length);

    // bind vertex buffer and associate position data with shader variables
    gl.bindBuffer(gl.ARRAY_BUFFER, buffer4);
    gl.vertexAttribPointer(vPosition, 2, gl.FLOAT, false, 0, 0);
    gl.enableVertexAttribArray(vPosition);
    // draw triangle
    color = vec4(red, green, blue, 1.0);
    gl.uniform4fv(colorLoc, color);
    gl.drawArrays(gl.TRIANGLE_STRIP, 0, letter4vertices.length);


    // bind vertex buffer and associate position data with shader variables
    gl.bindBuffer(gl.ARRAY_BUFFER, buffer5);
    gl.vertexAttribPointer(vPosition, 2, gl.FLOAT, false, 0, 0);
    gl.enableVertexAttribArray(vPosition);
    // draw rectangle
    color = vec4(1 - red, 1 - green, 1 - blue, 1.0);
    gl.uniform4fv(colorLoc, color);
    gl.drawArrays(gl.TRIANGLE_STRIP, 0, letter5vertices.length);

    // bind vertex buffer and associate position data with shader variables
    gl.bindBuffer(gl.ARRAY_BUFFER, buffer6);
    gl.vertexAttribPointer(vPosition, 2, gl.FLOAT, false, 0, 0);
    gl.enableVertexAttribArray(vPosition);
    // draw triangle
    color = vec4(red, green, blue, 1.0);
    gl.uniform4fv(colorLoc, color);
    gl.drawArrays(gl.TRIANGLE_STRIP, 0, letter6vertices.length);


    // bind vertex buffer and associate position data with shader variables
    gl.bindBuffer(gl.ARRAY_BUFFER, buffer7);
    gl.vertexAttribPointer(vPosition, 2, gl.FLOAT, false, 0, 0);
    gl.enableVertexAttribArray(vPosition);
    // draw rectangle
    color = vec4(red, green, blue, 1.0);
    gl.uniform4fv(colorLoc, color);
    gl.drawArrays(gl.TRIANGLE_STRIP, 0, letter7vertices.length);

    // bind vertex buffer and associate position data with shader variables
    gl.bindBuffer(gl.ARRAY_BUFFER, buffer8);
    gl.vertexAttribPointer(vPosition, 2, gl.FLOAT, false, 0, 0);
    gl.enableVertexAttribArray(vPosition);
    // draw rectangle
    color = vec4(red, green, blue, 1.0);
    gl.uniform4fv(colorLoc, color);
    gl.drawArrays(gl.TRIANGLE_STRIP, 0, letter8vertices.length);



    gl.uniform1f(xLoc, x);
    gl.uniform1f(yLoc, y);

    gl.uniform1f(scaleXLoc, scaleX);
    gl.uniform1f(scaleYLoc, scaleY);

    window.requestAnimFrame(render);
}
```