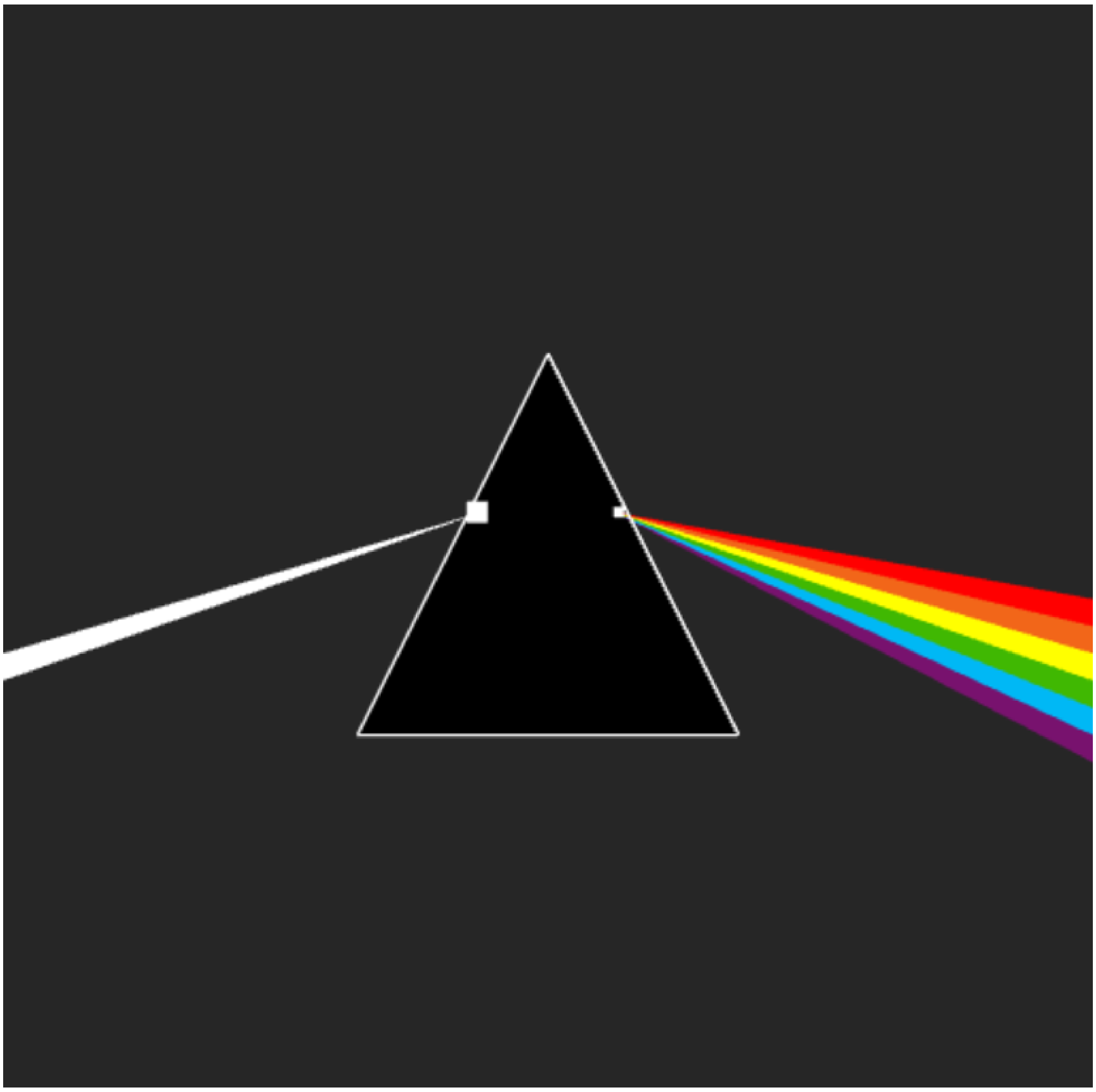
Name: Mustafa Baran Ercan
ID: 2881055520 **Section**: CMPE_360_01

PROJECT 6 REPORT



My image includes 3 lines, 2 points of different sizes and 8 triangles of different colors.

WebGL.html

```
<!DOCTYPE html>
<html>
<head>
   <title>WebGL Template</title>
   <meta http-equiv="refresh" content="3" >
    <!-- This in-line script is a vertex shader resource
        Shaders can be linked from an external file as well.
        First line must be shader language version, no spaces before.
        (Actually textbook's shader loader strips leading spaces...)
        -->
    <script id="vertex-shader" type="x-shader/x-vertex">
        #version 300 es

        // All vertex shaders require a position input.
        // The name can be changed.
        // Other inputs can be added as desired for colors and other features.
        in vec4 vPosition;
        in vec4 vColor;
        in float vPointSize;


        // This varying output is interpolated between the vertices in the
        // primitive we are drawing before being sent to an input with
        // matching name and type in the fragment shader
        out vec4 varColor;

        void main()
        {
            // gl_Position is a built-in vertex shader output.
            // Its value should always be set by the vertex shader.
            // The simplest shaders just copy the position attribute straight
to
            gl_Position = vPosition;
            varColor = vColor;
            gl_PointSize = vPointSize;
        }
    </script>

    <!-- This in-line script is a vertex shader resource
        Shaders can be linked from an external file as well.
        First line must be shader language version, no spaces before.
        (Actually textbook's shader loader strips the spaces...) -->
    <script id="fragment-shader" type="x-shader/x-fragment">
        #version 300 es

        // Sets default precision for floats.
        // Since fragment shaders have no default precision, you must either:
```

```
        //   - set a default before declaring types that use floating point OR
        //   - specify the precision before each floating point declaration
        // Choices are lowp, mediump, and highp.
        precision mediump float;

        in vec4 varColor;
        // The output of a fragment shader is sent to draw buffers,
        // which might be an array or the screen. The default is
        out vec4 fragColor;

        void main()
        {
            // In general, the fragment shader output should be set,
            //      but this is not required.
            // If an output is not set,
            //     there will be no output to the corresponding draw buffer.
            fragColor = varColor;
        }
    </script>

    <!-- These are external javascript files.
        The first three are the textbook libraries.
        The last one is your own javascript code. Make sure to change the name
        to match your javascript file. -->
    <script type="text/javascript" src="C:\Users\musta\Desktop\360\project
6\Common\utility.js"></script>
    <script type="text/javascript" src="C:\Users\musta\Desktop\360\project
6\Common\initShaders.js"></script>
    <script type="text/javascript" src="C:\Users\musta\Desktop\360\project
6\Common\MVnew.js"></script>
    <script type="text/javascript" src="C:\Users\musta\Desktop\360\project
6\Common\flatten.js"></script>
    <script type="text/javascript" src="WebGL.js"></script>
    </head>

    <body>
    <!-- This is the canvas - the only HTML element that can render WebGL
        graphics. You can have more than one WebGL canvas on a web page, but
        that gets tricky. Stick to one per page for now. -->
    <canvas id="gl-canvas" width="512" height="512">
        Oops ... your browser doesn't support the HTML5 canvas element
    </canvas>
</body>
</html>
```

WebGL.js

```
// This variable will store the WebGL rendering context
var gl;

window.onload = function init() {
   // Set up a WebGL Rendering Context in an HTML5 Canvas
   var canvas = document.getElementById("gl-canvas");
   gl = canvas.getContext('webgl2');
   if (!gl) alert("WebGL 2.0 isn't available");

   //  Configure WebGL
   //  eg. - set a clear color
   //      - turn on depth testing
   // This light gray clear colour will help you see your canvas
   gl.clearColor(0.15, 0.15, 0.15, 1.0);

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

   // Set up data to draw
   // Here, 2D vertex positions and RGB colours are loaded into arrays.


   var positions = [
      //white dot 1
      -0.13,  0.06,      // point 1
      //white dot 2
      0.13,  0.06,       // point 2

      // Line 1
      -0.35, -0.35,     // Line 1, point 1
      0, 0.35,       // Line 1, point 2
      // Line 2
      -0.35, -0.35,      // Line 2, point 1
      0.35, -0.35,      // Line 2, point 2
      // Line 3
      0.35, -0.35,      // Line 3, point 1
      0, 0.35,      // Line 3, point 2

      // Big black triangle on the center
      -0.35, -0.35,     // Triangle 1, point 1
      0.35, -0.35,      // Triangle 1, point 2
      0.0,  0.35,       // Triangle 1, point 3
      //red right triangle
      0.12,  0.06,      // Triangle 2, point 1
      1.0,  -0.1,       // Triangle 2, point 2
      1.0,  -0.15,      // Triangle 2, point 3
      //orange right triangle
      0.12,  0.06,      // Triangle 3, point 1
      1.0,  -0.15,      // Triangle 3, point 2
      1.0,  -0.20,      // Triangle 3, point 3
      //yellow right triangle
      0.12,  0.06,      // Triangle 4, point 1
      1.0,  -0.2,       // Triangle 4, point 2
      1.0,  -0.25,      // Triangle 4, point 3
      //green right triangle
      0.12,  0.06,      // Triangle 5, point 1
      1.0,  -0.25,      // Triangle 5, point 2
      1.0,  -0.3,       // Triangle 5, point 3
      //blue right triangle
      0.12,  0.06,      // Triangle 6, point 1
      1.0,  -0.3,       // Triangle 6, point 2
      1.0,  -0.35,      // Triangle 6, point 3
      //purple right triangle
      0.12,  0.06,      // Triangle 7, point 1
      1.0,  -0.35,      // Triangle 7, point 2
      1.0,  -0.4,       // Triangle 7, point 3
      //white left triangle
      -0.12,  0.06,      // Triangle 8, point 1
      -1.0,  -0.2,       // Triangle 8, point 2
      -1.0,  -0.25      // Triangle 8, point 3
   ];

var colors = [
      //white dot 1
      1, 1, 1,       // point 1
      //white dot 2
      1, 1, 1,       // point 1

      //white line 1
      1, 1, 1,       // line 1, point 1
      1, 1, 1,       // line 1, point 1
      //white line 2
      1, 1, 1,       // line 2, point 1
      1, 1, 1,       // line 2, point 2
      //white line 3
      1, 1, 1,       // line 3, point 1
      1, 1, 1,       // line 3, point 2


      // Big black triangle on the center
      0, 0, 0,          // Triangle 1, white
      0, 0, 0,          // Triangle 1, white
      0, 0, 0,          // Triangle 1, white
```

```
    //red right triangle
    1, 0, 0,           // Triangle 2, red
    1, 0, 0,           // Triangle 2, red
    1, 0, 0,           // Triangle 2, red
    //orange right triangle
    0.95, 0.4, 0.1,    // Triangle 3, orange
    0.95, 0.4, 0.1,    // Triangle 3, orange
    0.95, 0.4, 0.1,    // Triangle 3, orange
    //yellow right triangle
    1, 1, 0,           // Triangle 4, yellow
    1, 1, 0,           // Triangle 4, yellow
    1, 1, 0,           // Triangle 4, yellow
    //green right triangle
    0.25, 0.72, 0.01, // Triangle 5, green
    0.25, 0.72, 0.01, // Triangle 5, green
    0.25, 0.72, 0.01, // Triangle 5, green
    //blue right triangle
    0, 0.72, 0.96,     // Triangle 6, blue
    0, 0.72, 0.96,     // Triangle 6, blue
    0, 0.72, 0.96,     // Triangle 6, blue
    //purple right triangle
    0.47, 0.07, 0.43, // Triangle 7, purple
    0.47, 0.07, 0.43, // Triangle 7, purple
    0.47, 0.07, 0.43, // Triangle 7, purple
    //white left triangle
    1, 1, 1,           // Triangle 8, white
    1, 1, 1,           // Triangle 8, white
    1, 1, 1            // Triangle 8, white
];

var pointSizes = [
    10,  //white dot 1
    5,  //white dot 2
];


// Load the data into GPU data buffers
// The vertex positions are copied into one buffer
var vertex_buffer = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, vertex_buffer);
gl.bufferData(gl.ARRAY_BUFFER, flatten(positions), gl.STATIC_DRAW);

// The colours are copied into another buffer
var color_buffer = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, color_buffer);
gl.bufferData(gl.ARRAY_BUFFER, flatten(colors), gl.STATIC_DRAW);

var pointSize_buffer = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, pointSize_buffer);
```

```
   gl.bufferData(gl.ARRAY_BUFFER, flatten(pointSizes), gl.STATIC_DRAW);

   // Associate shader attributes with corresponding data buffers
   // Create a connection manager for the data, a Vertex Array Object
   // These are typically made global so you can swap what you draw in the
   //    render function.
   var triangleVAO = gl.createVertexArray();
   gl.bindVertexArray(triangleVAO);

   //Here we prepare the "vPosition" shader attribute entry point to
   //receive 2D float vertex positions from the vertex buffer
   var vPosition = gl.getAttribLocation(program, "vPosition");
   gl.bindBuffer(gl.ARRAY_BUFFER, vertex_buffer);
   gl.vertexAttribPointer(vPosition, 2, gl.FLOAT, false, 0, 0);
   gl.enableVertexAttribArray(vPosition);

   //Here we prepare the "vColor" shader attribute entry point to
   //receive RGB float colours from the colour buffer
   var vColor = gl.getAttribLocation(program, "vColor");
   gl.bindBuffer(gl.ARRAY_BUFFER, color_buffer);
   gl.vertexAttribPointer(vColor, 3, gl.FLOAT, false, 0, 0);
   gl.enableVertexAttribArray(vColor);

   var vPointSize = gl.getAttribLocation(program, "vPointSize");
   gl.bindBuffer(gl.ARRAY_BUFFER, pointSize_buffer);
   gl.vertexAttribPointer(vPointSize, 1, gl.FLOAT, false, 0, 0);
   gl.enableVertexAttribArray(vPointSize);

   var lineVertexBuffer = gl.createBuffer();
   gl.bindBuffer(gl.ARRAY_BUFFER, lineVertexBuffer);
   gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(positions),
gl.STATIC_DRAW);

   var vPosition = gl.getAttribLocation(program, "vPosition");
   gl.vertexAttribPointer(vPosition, 2, gl.FLOAT, false, 0, 0);
   gl.enableVertexAttribArray(vPosition);



   // Get addresses of shader uniforms
   // None in this program...

   //Either draw once as part of initialization
  render();

   //Or schedule a draw just before the next repaint event
   //requestAnimationFrame(render);
};
```

```
function render() {
   // clear the screen
   // Actually, the  WebGL automatically clears the screen before drawing.
   // This command will clear the screen to the clear color instead of white.
   gl.clear(gl.COLOR_BUFFER_BIT);

   // draw
   // Draw the data from the buffers currently associated with shader
variables
   // Our triangle has three vertices that start at the beginning of the
buffer.
   gl.drawArrays(gl.TRIANGLES, 8, 30);
   gl.drawArrays(gl.POINTS, 0, 2);
   gl.drawArrays(gl.LINES, 2, 6);

}
```