INB381 - Workshop 2

Configure Personal Development Environment

Windows Configuration

1. From control panel select -> Programs -> Programs and Features -> Turn Windows Features On or Off
2. Check the box for Internet Information Services
3. Create a directory on your system and give it a name
4. Start IIS and create a virtual directory give it a name and set the physical path to the directory you created in step 3
5. Also place your solution files in the directory you created in step 3

Mac Configuration

1. Apache is installed and running by default
2. To check open your browser and enter the url: localhost
    a. The response should be: It works!
3. Place your files in the directory: Library/WebServer/Documents/
4. Permission is required to modify and copy your files into this directory
5. You may configure a personal site if you wish. Details can be found on Google

Create A Triangle Using WebGL

Project Setup

1. Create directory for your workshop.
2. Download the file common.zip and extract into your project folder.
    a. The common resources are required for all future solutions.
3. Create two files on named triangle.html and the other triangle.js in the root of your project

HTML File

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Workshop 2 - Simple Triangle</title>

    <script id="vertex-shader" type="x-shader/x-vertex">
        attribute vec4 vPosition;
        void main(){
            gl_Position = vPosition;
        }
    </script>
    <script id="fragment-shader" type="x-shader/x-fragment">
        precision mediump float;
        void main(){
            gl_FragColor = vec4( 1.0, 0.0, 0.0, 1.0 );
        }
    </script>
    <script type="text/javascript" src="common/initShaders.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="triangle.js"></script>
</head>
<body>
    <canvas width="512" height="512" id="gl-canvas"></canvas>
</body>
</html>
```

JavaScript File

```
var gl;
var points;
function initWebGL(canvas) {
    gl = null;
    try {
        // Try to grab the standard context. If it fails, fallback to experimental.
        gl = canvas.getContext("webgl") || canvas.getContext("experimental-webgl");
    }
    catch(e) {}

    // If we don't have a GL context, give up now
    if (!gl) {
        alert("Unable to initialize WebGL. Your browser may not support it.");
        gl = null;
    }
    return gl;

}

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

    //gl = WebGLUtils.setupWebGL( canvas );
    gl = initWebGL(canvas);

    if ( !gl ) { alert( "WebGL isn't available" );
    }

    // Three Vertices
    var vertices = [
        vec2( -1, -1 ),
        vec2(  0,  1 ),
        vec2(  1, -1 )
    ];

    //  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

    var program = initShaders( gl, "vertex-shader", "fragment-shader" );
    gl.useProgram( program );

    // Load the data into the GPU
    var bufferId = gl.createBuffer();
    gl.bindBuffer( gl.ARRAY_BUFFER, bufferId );
    gl.bufferData( gl.ARRAY_BUFFER, flatten(vertices), gl.STATIC_DRAW );

    // Associate out shader variables with our data buffer
    var vPosition = gl.getAttribLocation( program, "vPosition" );
    gl.vertexAttribPointer( vPosition, 2, gl.FLOAT, false, 0, 0 );
    gl.enableVertexAttribArray( vPosition );
    render();

};

function render() {

    gl.clear( gl.COLOR_BUFFER_BIT );
    gl.drawArrays( gl.TRIANGLES, 0, 3 );

}
```