CS 315 – Introduction to Computer Graphics
Assignment 1
Solution Key

Question 1

The program is based the example "gasket2.html" and "gasket2.js". The modified parts are highlighted in yellow. Three sample output images are shown after the program.

"ass1_Q1.html"

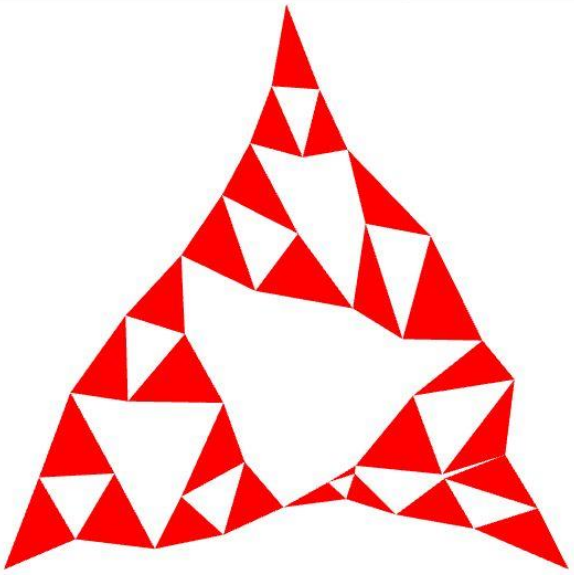
```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;charset=utf-8" >
<title>Assignment 1 Question 1</title>


<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/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="ass1_Q1.js"></script>
</head>

<body>
<canvas id="gl-canvas" width="512" height="512">
Oops ... your browser doesn't support the HTML5 canvas element
</canvas>
</body>
</html>
```

"ass1_Q1.js"

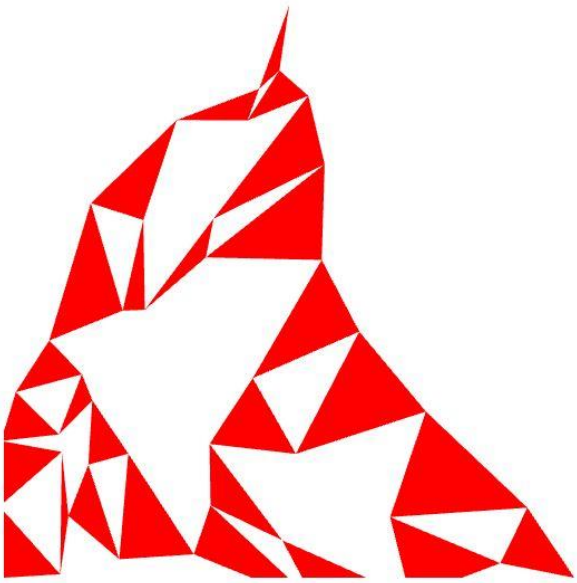
```
var canvas;
var gl;

var points = [];

var NumTimesToSubdivide = 3;
var maxD = 0.3;          // max displacement at the 1st level

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

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

    //
    //  Initialize our data for the Sierpinski Gasket
    //

    // First, initialize the corners of our gasket with three points.

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

    divideTriangle( vertices[0], vertices[1], vertices[2],
                    maxD, NumTimesToSubdivide);

    //
    //  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(points), 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 triangle( a, b, c )
{
    points.push( a, b, c );
}

function randD( D )
{
   // generate two random numbers in the range [-D, D]

   var dx = D * (Math.random()*2.0 -1.0);
   var dy = D * (Math.random()*2.0 -1.0);
   var result = vec2(dx, dy);
   return result;
}


function divideTriangle( a, b, c, D, count )
{

    // check for end of recursion

    if ( count === 0 ) {
        triangle( a, b, c );
    }
    else {

        //bisect the sides with a random displacement for the
        // middle point position

        var ab = add( mix( a, b, 0.5 ), randD(D));
        var ac = add( mix( a, c, 0.5 ), randD(D));
        var bc = add( mix( b, c, 0.5 ), randD(D));

        --count;

        // three new triangles for recursive subdivision
        // the max displacement is reduced by a half

        divideTriangle( a, ab, ac, D/2.0, count );
        divideTriangle( c, ac, bc, D/2.0, count );
        divideTriangle( b, bc, ab, D/2.0, count );
    }
}

function render()
{
    gl.clear( gl.COLOR_BUFFER_BIT );
    gl.drawArrays( gl.TRIANGLES, 0, points.length );
}
```

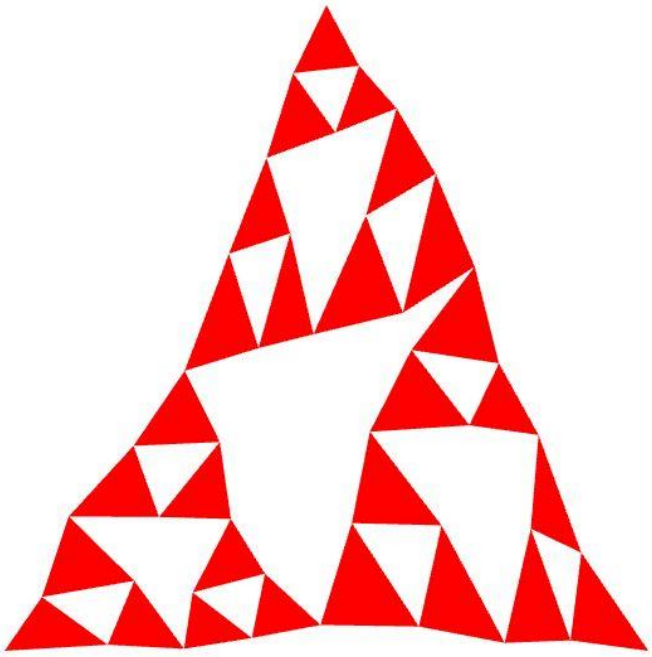
Sample Outputs



maxD = 0.3



maxD = 0.4



maxD = 0.2

Question 2

"ass1_Q2.html"

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;charset=utf-8" >
<title>Assignment 1 Question 2</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 );
}
```

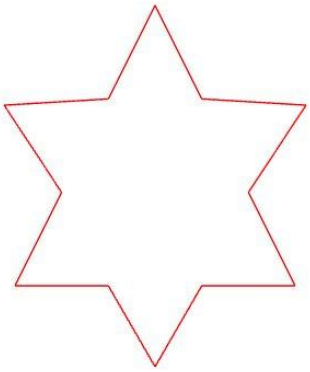
```
</script>

<script type="text/javascript" src="../Common/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="ass1_Q2.js"></script>
</head>

<body>
<canvas id="gl-canvas" width="512" height="512">
Oops ... your browser doesn't support the HTML5 canvas element
</canvas>
</body>
</html>
```

"ass1_Q2.js"

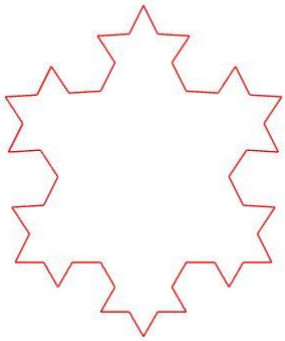
```
var canvas;
var gl;

var points = [];

var NumTimesToSubdivide = 1;

// Values of sine and cosine functions to be used for rotation
var sin60 = 0.866;      // value of sin(60)
var cos60 = 0.5;        // value of cos(60)

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

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

    //
    //  Initialize our data for the Sierpinski Gasket
    //

    // First, initialize the corners of our gasket with three points.

    var vertices = [
        vec2( -0.5, -0.5 ),    // the size of the triangle is reduced
        vec2(  0,  0.5 ),      // by a half such that the new figure
        vec2(  0.5, -0.5 )     // won't be out of clipping boundary
    ];

    divideEdge( vertices[0], vertices[1], NumTimesToSubdivide);
    divideEdge( vertices[1], vertices[2], NumTimesToSubdivide);
    divideEdge( vertices[2], vertices[0], NumTimesToSubdivide);

    //
    //  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(points), 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 edge( a, b )
{
    points.push( a, b);
}

function divideEdge( a, b, count )
{

    // check for end of recursion

    if ( count === 0 ) {
        edge( a, b );
    }
    else {

        // the line from a to b is divided into three equal segments
        // by two points c and e

        var c = mix( a, b, 1.0/3.0 );
        var e = mix( a, b, 2.0/3.0 );

        // find vector from point a to pint c
        var v1 = vec2((c[0]-a[0]), (c[1]-a[1]) );

        // rotate v1 for 60 degrees
        var v2 = vec2(v1[0]*cos60 - v1[1]*sin60,
                      v1[0]*sin60 + v1[1]*cos60);

        // points d and c and e will form a triangle with equal
        // length edges
        var d = add(c, v2);

        --count;
```

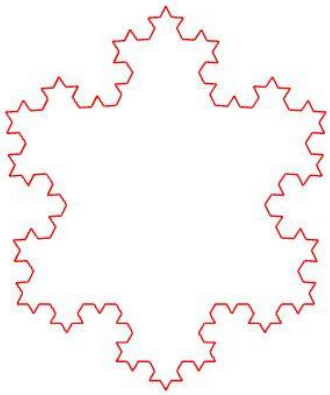
```
        // four new lines are recursively further divided

        divideEdge( a, c, count );
        divideEdge( c, d, count );
        divideEdge( d, e, count );
        divideEdge( e, b, count );

    }
}

function render()
{
    gl.clear( gl.COLOR_BUFFER_BIT );
    gl.drawArrays( gl.LINES, 0, points.length );
}
```



I level subdivision



2 levels subdicition



3 levels subdivision