```
"use strict";

var gl;

var tParam = 0.0;
var tLoc;
var deltaT = 0.01;

var color = vec4(1.0,0.65,0.0,1.0);
//var vec4 color;
var Ucolor = vec4(1.0,0.65,0.0,1.0);
var Icolor = vec4(0.0,0.0,1.0,1.0);
var colorLoc;

var delay = 100;
var morph = true;

init();

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

    gl = canvas.getContext('webgl2');
    if (!gl) alert("WebGL 2.0 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

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

    var I = [
        vec2( -0.75 ,  0.75 ),
        vec2(  0.75 ,  0.75 ),
        vec2(  0.75 ,  0.50 ),
        vec2(  0.25 ,  0.50 ),
        vec2(  0.25 , -0.50 ),
        vec2(  0.75 , -0.50 ),
        vec2(  0.75 , -0.75 ),
        vec2( -0.75 , -0.75 ),
        vec2( -0.75 , -0.50 ),
        vec2( -0.25 , -0.50 ),
        vec2( -0.25 ,  0.50 ),
        vec2( -0.75 ,  0.50 ),
        vec2( -0.75 ,  0.75 )
    ];

    var U = [
        vec2( -0.75 ,  0.75 ),
        vec2( -0.38 ,  0.75 ),
        vec2( -0.38 , -0.38 ),
        vec2(  0.38 , -0.38 ),
        vec2(  0.38 ,  0.75 ),
        vec2(  0.75 ,  0.75 ),
        vec2(  0.75 ,  0.00 ),
        vec2(  0.75 , -0.38 ),
        vec2(  0.75 , -0.75 ),
        vec2( -0.75 , -0.75 ),
        vec2( -0.75 , -0.38 ),
        vec2( -0.75 ,  0.00 )
    ];


    // Load the I into the GPU
    var vBufferI = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER,
vBufferI);
    gl.bufferData(gl.ARRAY_BUFFER,
flatten(I), gl.STATIC_DRAW);

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

gl.enableVertexAttribArray(ipositionLoc);

    // Load the U into the GPU
    var vBufferU = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER,
vBufferU);
    gl.bufferData(gl.ARRAY_BUFFER,
flatten(U), gl.STATIC_DRAW);

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

gl.enableVertexAttribArray(upositionLoc);

    tLoc = gl.getUniformLocation( program,
"t" );

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

    // Initialize event handlers

document.getElementById("Morph").onclick =
function () {
        morph = !morph;
    };

    window.onkeydown = function(event) {
        var key =
String.fromCharCode(event.keyCode);
        switch(key) {
          case '1':
            morph = !morph;
            break;

          case '2':
            deltaT /= 2.0;
            break;

          case '3':
            deltaT *= 2.0;
            break;
        }
    };
    render();
};
```

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

    if (morph) tParam += deltaT;
    if (tParam>=1.0 || tParam<= 0.0)
deltaT = -deltaT;
    gl.uniform1f(tLoc, tParam);

    color = mix(Icolor,Ucolor,tParam);
    gl.uniform4fv(colorLoc, color);

    gl.drawArrays(gl.LINE_LOOP, 0, 12);

    setTimeout(
        function
(){requestAnimationFrame(render);}, delay
    );
}
```