COMP 370 assignment #1: Many shapes and colours

Thomas Williamson

id: 588206

2021/09/22

Many shapes and colours: make a graphics program that displays a triangle colored red green and blue on the vertices, when “1” is pressed the color changes to red, when “2” is pressed the color changes to green, when “3” is pressed the color changes to blue, “4”is pressed the color changes to a random color, “5”is pressed the color changes to yellow, “c” is pressed the shape changes to circle, “t” is pressed the shape changes to triangle, “s” is pressed the shape changes to square.

```
<!--
COMP 370 assignment #1: Many shapes and colours
Thomas Williamson
id: 588206
2021/09/22
 -->

<html>

<script id="vertex-shader" type="x-shader/x-vertex">
#version 300 es

in vec4 aPosition;
in vec4 aColor;

out vec4 vColor;

void main()
{
  gl_Position = aPosition;
  vColor = aColor;
}
</script>

<script id="fragment-shader" type="x-shader/x-fragment">
#version 300 es

precision mediump float;

in vec4 vColor;
out vec4 fColor;

void main()
{
    //fColor = vec4( 1.0, 0.0, 0.0, 1.0 );
    //fColor = vec4( 0.0, 1.0, 0.0, 1.0 );
    //fColor = vec4( 0.0, 0.0, 1.0, 1.0 );
    //fColor = vec4( 1.0, 1.0, 1.0, 1.0 );
    //fColor = vec4( 1.0, 1.0, 0.0, 1.0 );
    fColor = vColor;
}
</script>

<script type="text/javascript" src="../Common/initShaders.js"></script>
<script type="text/javascript" src="many_shapes_and_colours_a1.js"></script>

<canvas id="gl-canvas" width="512" height="512"> </canvas>

</html>

```

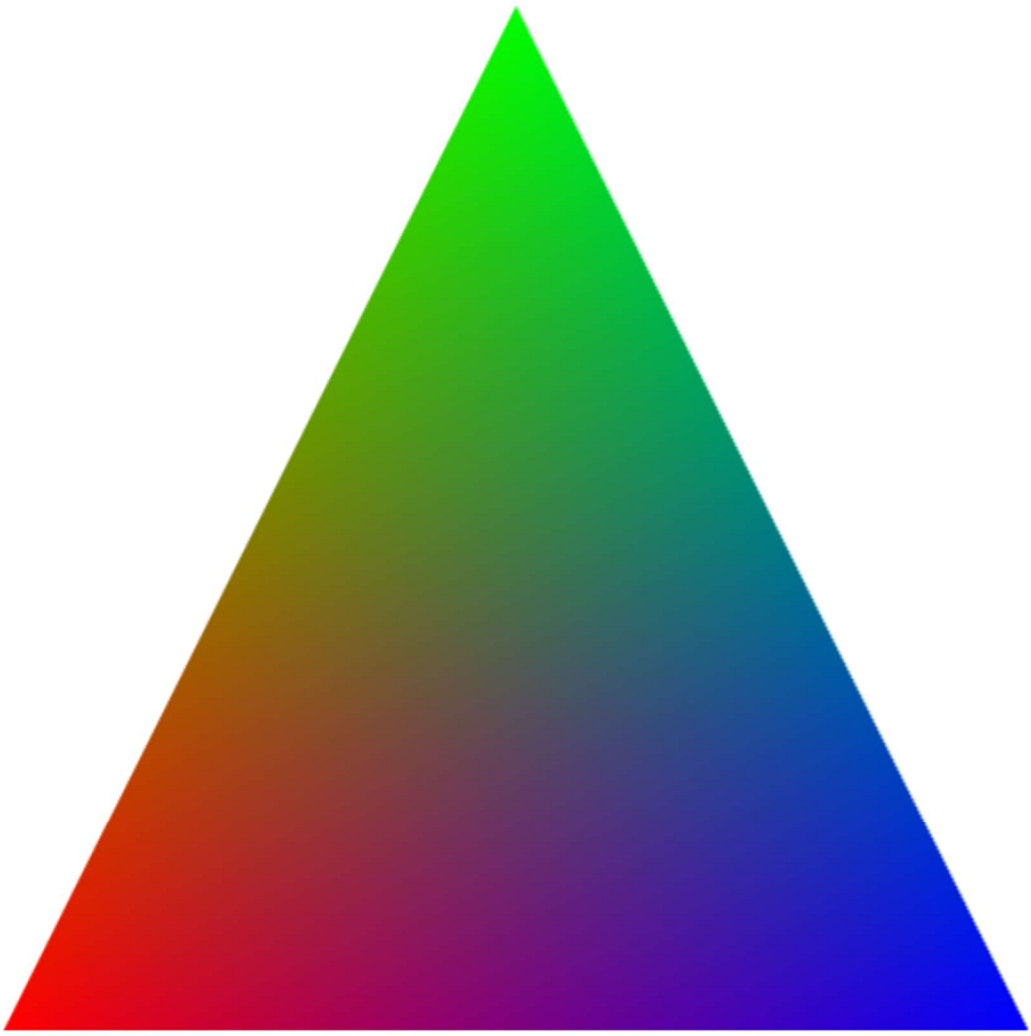
```
/*
COMP 370 assignment #1: Many shapes and colours
Thomas Williamson
id: 588206
2021/09/22

*/

"use strict";
//load variables
var gl;
var triangle;
var square;
var circle
var program;
var program2;
var program3
var t_vPosition;
var s_vPosition;
var ci_vPosition;
var tBuffer;
var sBuffer;
var ciBuffer
var t_cBuffer;
var s_cBuffer;
var ci_cBuffer
var color = 1
var shape = 1; //1: triangle, 2: square
var colorT = [1,0,0, 0,1,0, 0,0,1,];
var colorS = [1,0,0, 1,0,0, 1,0,0, 1,0,0,];
var colorC = [1,0,0, 1,0,0, 1,0,0, 1,0,0, 1,0,0, ...
var t_ColorLoc;
var s_ColorLoc;
var ci_ColorLoc;


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

    gl = canvas.getContext('webgl2');
    if (!gl) { alert( "WebGL 2.0 isn't available" ); }

    //set shape geometries
    triangle = new Float32Array([
        -1, -1 ,
         0,  1 ,
         1, -1
    ]);

    square = new Float32Array([
        -1,  1,
        -1, -1,
         1,  1,
         1, -1
    ]);
```

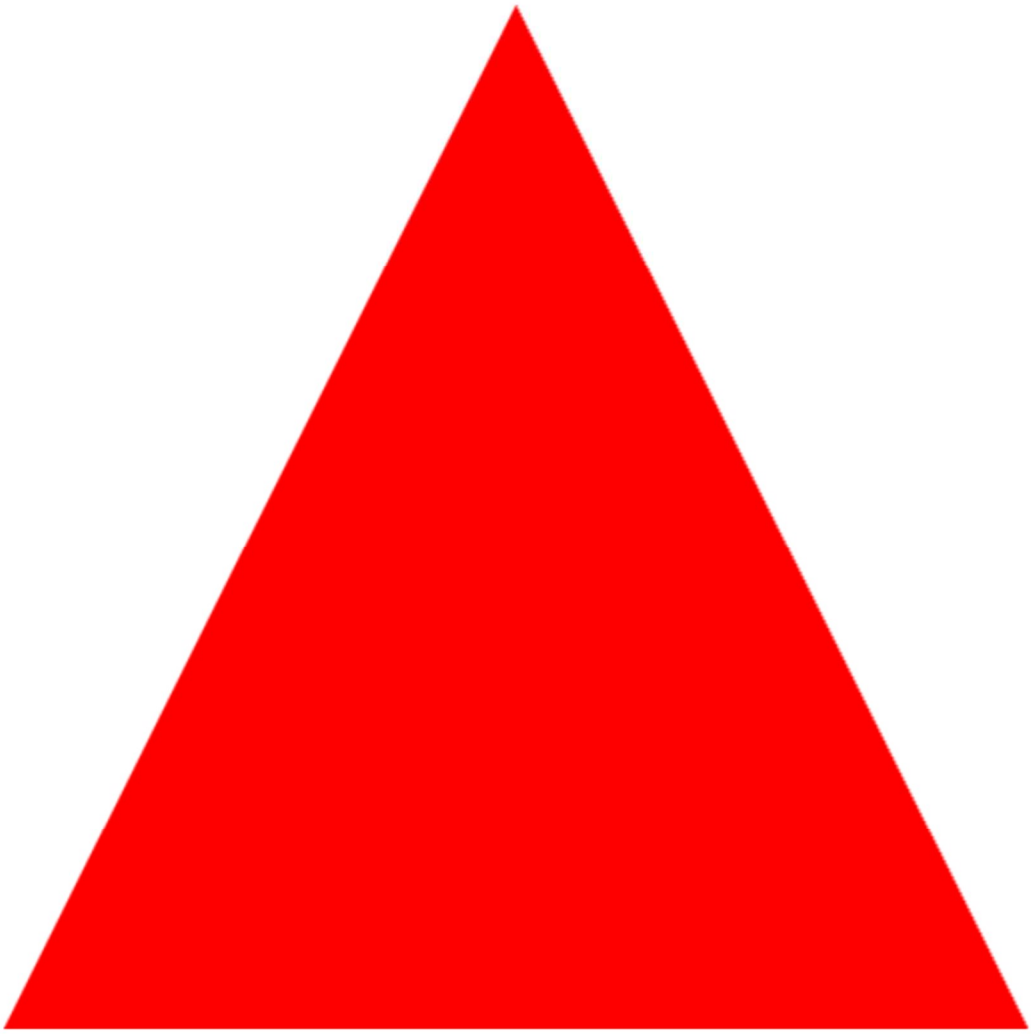
```
87
88          //python code used to obtain verticies
89          //      import math
90          // print("{")
91          // for i in range(90):
92          //      if (i%3) == 0:
93          //          #print(i)
94          //          x = round((math.cos(i*math.pi/180)*1),2)
95          //          y = round((math.sin(i*math.pi/180)*1),2)
96          //          print(str(x) + ", " + str(y) + ", ")
97          // for i in range(90):
98          //      if (i%3) == 0:
99          //          x = round(-(math.cos((90-i)*math.pi/180)*1),2)
100         //          y = round((math.sin((90-i)*math.pi/180)*1),2)
101         //          print(str(x) + ", " + str(y) + ", ")
102         // for i in range(90):
103         //      if (i%3) == 0:
104         //          x = round(-(math.cos(i*math.pi/180)*1),2)
105         //          y = round(-(math.sin(i*math.pi/180)*1),2)
106         //          print(str(x) + ", " + str(y) + ", ")
107         // for i in range(90):
108
109         //      if (i%3) == 0:
110         //          x = round((math.cos((90-i)*math.pi/180)*1),2)
111         //          y = round(-(math.sin((90-i)*math.pi/180)*1),2)
112         //          print(str(x) + ", " + str(y) + ", ")
171     circle = new Float32Array([ ...
174
175     gl.viewport( 0, 0, canvas.width, canvas.height );
176     gl.clearColor( 1, 1, 1, 1.0 );
177     gl.clear(gl.COLOR_BUFFER_BIT);
178
179 //initiate vertex and fragment - shader buffer datta
180     program = initShaders( gl, "vertex-shader", "fragment-shader" );
181     program2 = initShaders( gl, "vertex-shader", "fragment-shader" );
182     program3 = initShaders( gl, "vertex-shader", "fragment-shader");
183
184 //triangle buffer data and render
185     tBuffer = gl.createBuffer();
186     gl.bindBuffer( gl.ARRAY_BUFFER, tBuffer );
187     gl.bufferData( gl.ARRAY_BUFFER, triangle, gl.STATIC_DRAW );
188
189     t_vPosition = gl.getAttribLocation( program, "aPosition" );
190     gl.vertexAttribPointer( t_vPosition, 2, gl.FLOAT, false, 0, 0 );
191
192     t_cBuffer = gl.createBuffer();
193     gl.bindBuffer(gl.ARRAY_BUFFER, t_cBuffer);
194     gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(colorT), gl.STATIC_DRAW );
195
196     t_ColorLoc = gl.getAttribLocation( program, "aColor");
197     gl.vertexAttribPointer(t_ColorLoc, 3, gl.FLOAT, false, 0, 0);
198
199     gl.useProgram( program );
200     gl.enableVertexAttribArray( t_vPosition );
201
202     gl.enableVertexAttribArray(t_ColorLoc);
203
204     render();
```

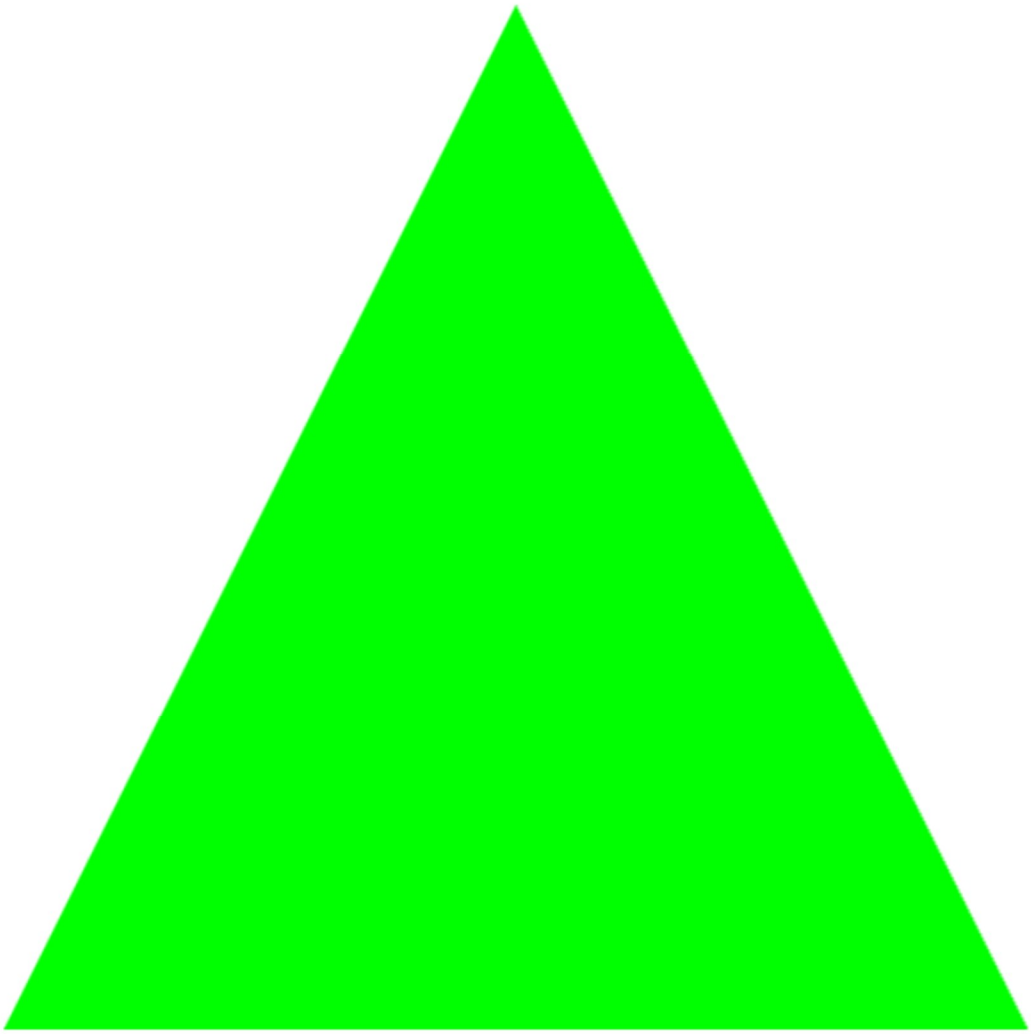
```
    //square ...
        sBuffer = gl.createBuffer();
        gl.bindBuffer( gl.ARRAY_BUFFER, sBuffer );
        gl.bufferData( gl.ARRAY_BUFFER, square, gl.STATIC_DRAW );
        s_vPosition = gl.getAttribLocation( program2, "aPosition" );
        gl.vertexAttribPointer( s_vPosition, 2, gl.FLOAT, false, 0, 0 );

        s_cBuffer = gl.createBuffer();
        gl.bindBuffer(gl.ARRAY_BUFFER, s_cBuffer);
        gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(colorS), gl.STATIC_DRAW );

        s_ColorLoc = gl.getAttribLocation( program2, "aColor");
        gl.vertexAttribPointer(s_ColorLoc, 3, gl.FLOAT, false, 0, 0);

    //circle ...
        ciBuffer = gl.createBuffer();
        gl.bindBuffer( gl.ARRAY_BUFFER, ciBuffer );
        gl.bufferData( gl.ARRAY_BUFFER, circle, gl.STATIC_DRAW );
        ci_vPosition = gl.getAttribLocation( program3, "aPosition" );
        gl.vertexAttribPointer( ci_vPosition, 2, gl.FLOAT, false, 0, 0 );

        ci_cBuffer = gl.createBuffer();
        gl.bindBuffer(gl.ARRAY_BUFFER, ci_cBuffer);
        gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(colorC), gl.STATIC_DRAW );

        ci_ColorLoc = gl.getAttribLocation( program3, "aColor");
        gl.vertexAttribPointer(ci_ColorLoc, 3, gl.FLOAT, false, 0, 0);

        window.addEventListener('keydown', this.checkKey);

    };


    function render() {

        if(shape==1){
            gl.clear( gl.COLOR_BUFFER_BIT );
            gl.drawArrays( gl.TRIANGLES, 0, 3 );
        }else if(shape==2){
            gl.clear( gl.COLOR_BUFFER_BIT );
            gl.drawArrays( gl.TRIANGLE_STRIP, 0, 4 );
        }else if(shape==3){
            //console.log("c")

            gl.clear( gl.COLOR_BUFFER_BIT );
            gl.drawArrays( gl.TRIANGLE_FAN, 0, 122 );
        }
    }

    // keyboard input

    function checkKey(e){
        switch(e.keyCode){
            // input "1" color red
            case 49:
                colorT = [1,0,0, 1,0,0, 1,0,0];
                colorS = [1,0,0, 1,0,0, 1,0,0, 1,0,0];
```

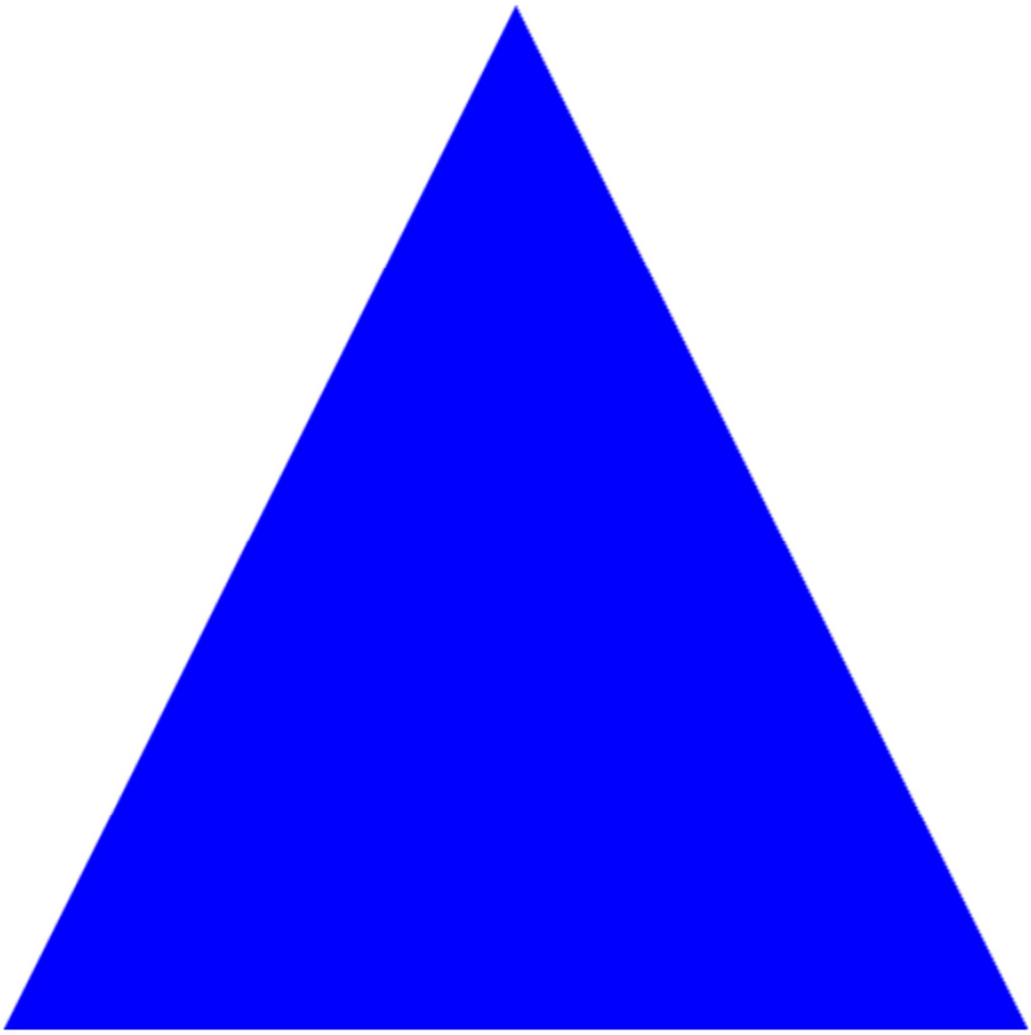
```
261             colorT = [1,0,0, 1,0,0, 1,0,0];
262             colorS = [1,0,0, 1,0,0, 1,0,0, 1,0,0];
292             colorC = [1,0,0, 1,0,0, 1,0,0, 1,0,0, 1,0,0, ...
293             if(shape==1){
294                 triangle_Binding();
295             }else if(shape==2){
296                 square_Binding();
297             }else if(shape==3){
298                 circle_Binding();
299             }
300             render();
301             break
302
303         // input "2" color green
304          case 50:
305             colorT = [0,1,0, 0,1,0, 0,1,0];
306             colorS = [0,1,0, 0,1,0, 0,1,0, 0,1,0];
334             colorC = [0,1,0, 0,1,0, 0,1,0, 0,1,0, 0,1,0, 0,1,0, ...
337             if(shape==1){
338                 triangle_Binding();
339             }else if(shape==2){
340                 square_Binding();
341             }else if(shape==3){
342                 circle_Binding();
343             }
344             render();
345             break
346
347         // input "3" color blue
348          case 51:
349             colorT = [0,0,1, 0,0,1, 0,0,1, 0,0,1];
350             colorS = [0,0,1, 0,0,1, 0,0,1, 0,0,1];
376             colorC = [0,0,1, 0,0,1, 0,0,1, 0,0,1, 0,0,1, 0,0,1, ...
381             if(shape==1){
382                 triangle_Binding();
383             }else if(shape==2){
384                 square_Binding();
385             }else if(shape==3){
386                 circle_Binding();
387             }
388             render();
389             break
390
391          // input "4" color random
392          case 52:
393             colorT = [Math.random(),Math.random(),Math.random(), Math.random(),Math.random(),Math.random(), Math.
394             colorS = [Math.random(),Math.random(),Math.random(), Math.random(),Math.random(),Math.random(), Math.
424             colorC = [Math.random(),Math.random(),Math.random(), Math.random(),Math.random(),Math.random(), Math.
425             if(shape==1){
426                 triangle_Binding();
427             }else if(shape==2){
428                 square_Binding();
429             }else if(shape==3){
430                 circle_Binding();
431             }
432             render();
433             break
```

```
434
435             // input "5" color yellow
436             case 53:
437                 colorT = [1,1,0, 1,1,0, 1,1,0];
438                 colorS = [1,1,0, 1,1,0, 1,1,0, 1,1,0];
459                 colorC = [1,1,0, 1,1,0, 1,1,0, 1,1,0, 1,1,0, ...
469                 if(shape==1){
470                     triangle_Binding();
471                 }else if(shape==2){
472                     square_Binding();
473                 }else if(shape==3){
474                     circle_Binding();
475                 }
476                 render();
477                 break
478
479     // input "s" square
480             case 83:
481                 shape = 2;
482                 square_Binding();
483                 render();
484                 break;
485
486             // input "t" triangle
487             case 84:
488                 shape = 1;
489                 triangle_Binding();
490                 render();
491                 break;
492
493             // input "c" circle
494             case 67:
495                 shape = 3;
496                 circle_Binding();
497                 render();
498                 break;
499         }
500     }
501
502     //shape data
503     function triangle_Binding(){
504         gl.useProgram( program );
505         gl.enableVertexAttribArray( t_vPosition );
506         gl.bindBuffer( gl.ARRAY_BUFFER, tBuffer );
507         gl.vertexAttribPointer( t_vPosition, 2, gl.FLOAT, false, 0, 0 );
508         gl.bindBuffer(gl.ARRAY_BUFFER, t_cBuffer);
509         gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(colorT), gl.STATIC_DRAW );
510         t_ColorLoc = gl.getAttribLocation( program, "aColor");
511         gl.vertexAttribPointer(t_ColorLoc, 3, gl.FLOAT, false, 0, 0);
512     }
513
514     function square_Binding(){
515         gl.useProgram( program2 );
516         gl.enableVertexAttribArray( s_vPosition );
517         gl.bindBuffer( gl.ARRAY_BUFFER, sBuffer );
518         gl.vertexAttribPointer( s_vPosition, 2, gl.FLOAT, false, 0, 0 );
519         gl.bindBuffer(gl.ARRAY_BUFFER, s_cBuffer);
520         gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(colorS), gl.STATIC_DRAW );
```









444