WEBGL PROGRAMMING

2014-11-6

从CANVAS到WEBGL

123.html

```html
<!DOCTYPE HTML>
<html>
<body>

<canvas id="myCanvas" width="640px" height= "480px">your browser does not support the canvas tag </canvas>

<script type="text/javascript">

function main(){
    var canvas=document.getElementById('myCanvas');
    var ctx=canvas.getContext('2d');
    ctx.fillStyle='#000000';
    ctx.fillRect(0,0,640,480);
}

</script>

</body>
</html>
```

- Canvas使用Javascript代码，通过context调用各种绘制函数

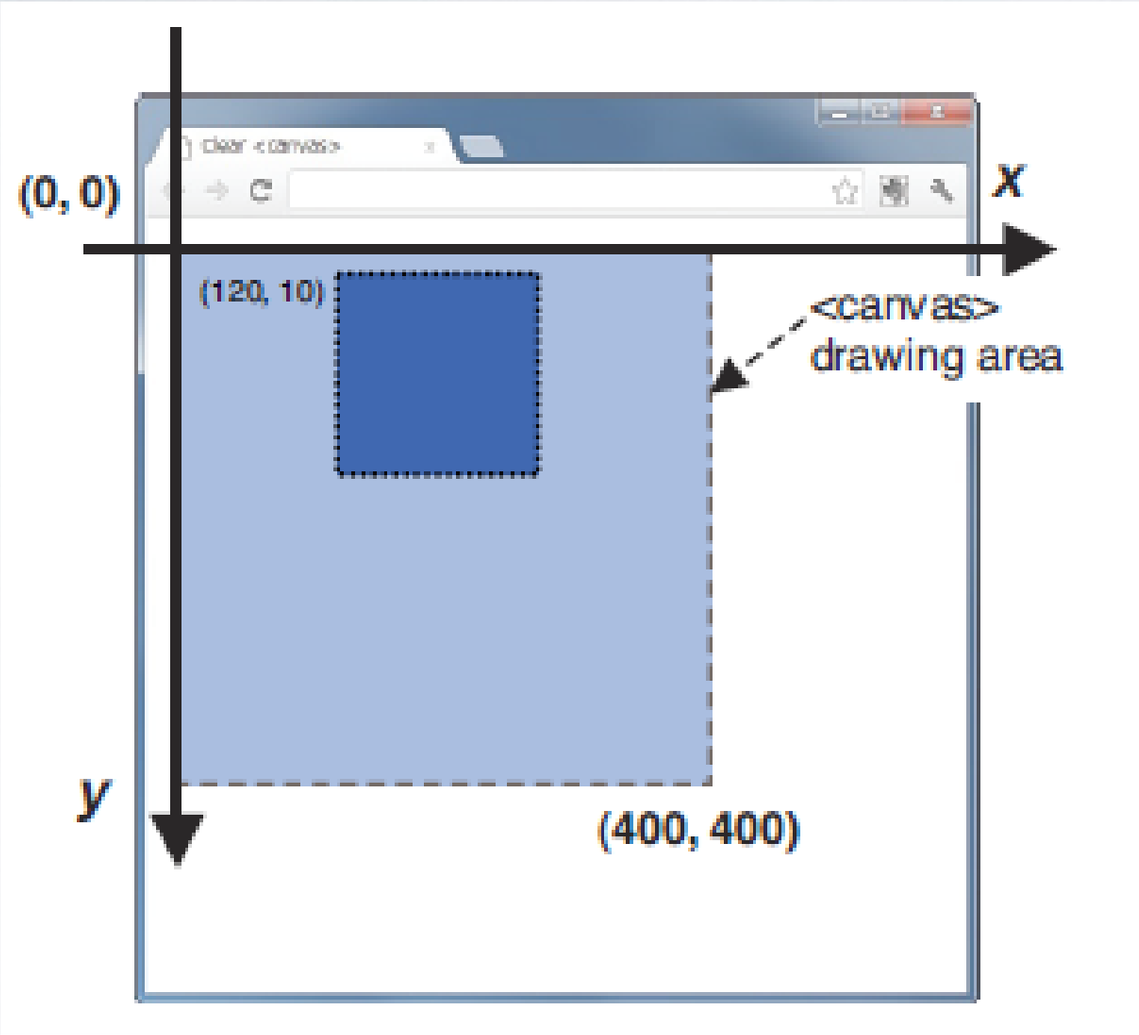
从CANVAS到WEBGL

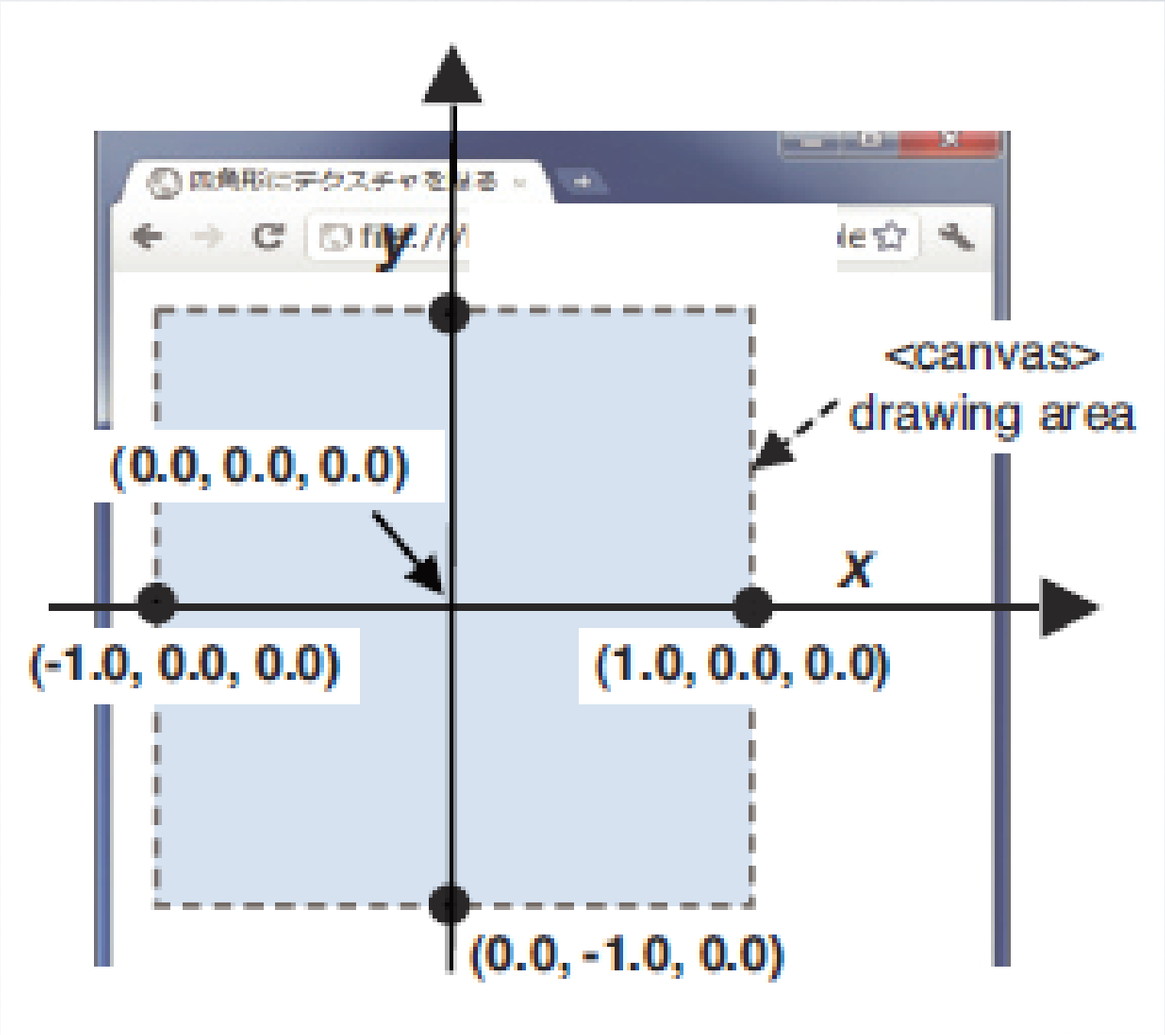
HelloCanvas.js

```
// HelloCanvas.js (c) 2012 matsuda
function main() {
  // Retrieve <canvas> element
  var canvas = document.getElementById('webgl');

  // Get the rendering context for WebGL
  var gl = getWebGLContext(canvas);
  if (!gl) {
    console.log('Failed to get the rendering context for WebGL');
    return;
  }

  // Set clear color
  gl.clearColor(0.0, 0.0, 0.0, 1.0);

  // Clear <canvas>
  gl.clear(gl.COLOR_BUFFER_BIT);
}
```

- WebGL以Canvas为载体，获取一个上下文gl来调用各种接口

从CANVAS到WEBGL





- WebGL以Canvas为载体在浏览器中绘制，但具有不同的坐标系

WEBGL组成

- WebGL系统包含两种编程语言：Javascript和OpenGL ES Shading Language
  - 通过Javascript进行总体控制，提供绘制内容（画什么）
  - 着色器语言（Shading Language）控制绘制过程（怎么画）
- 着色器语言的代码以字符串的形式通过Javascript传入WebGL系统，并由显卡来执行。

WEBGL——SHADER

```
3   var VSHADER_SOURCE =
4     'attribute vec4 a_Position;\n' +
5     'void main() {\n' +
6     '  gl_Position = a_Position;\n' +
7     '}\n';
8
9   // Fragment shader program
10  var FSHADER_SOURCE =
11    'void main() {\n' +
12    '  gl_FragColor = vec4(1.0, 0.0, 0.0, 1.0);\n' +
13    '}\n';
14
```

- Shader是由显卡执行的程序，由Javascript传入WebGL系统，在Javascript中以字符串的形式存在

WEBGL——SHADER

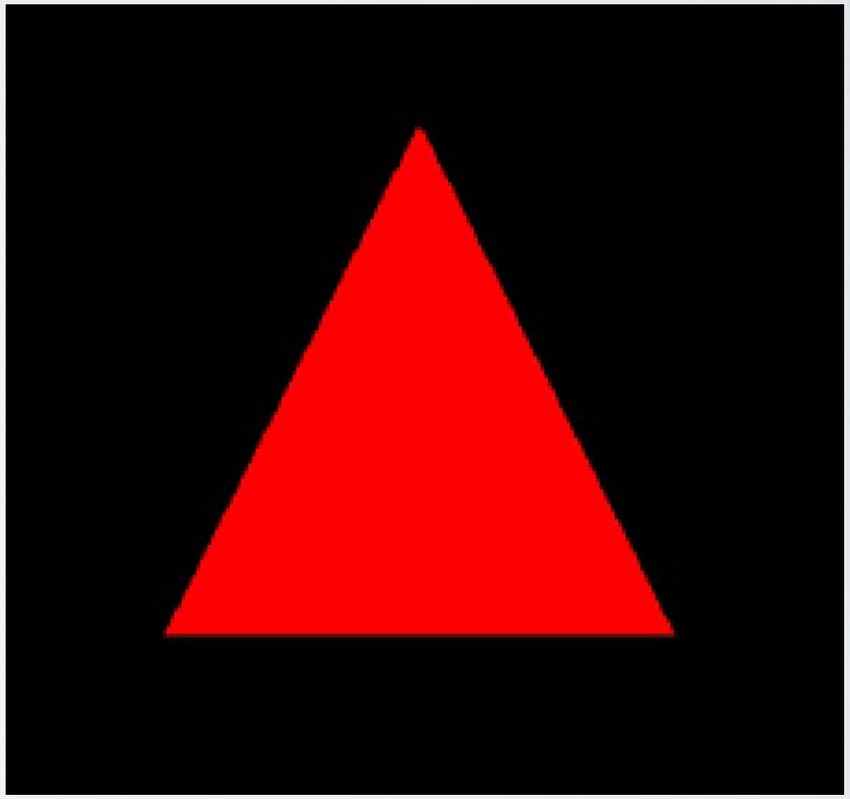
- 使用Shader之前，需要编译连接和启用，比较复杂
- 通过cuon-utils.js库，简化这一过程

```
32
33    // Initialize shaders
34    if (!initShaders(gl, VSHADER_SOURCE, FSHADER_SOURCE)) {
35      console.log('Failed to intialize shaders.');
36      return;
37    }
38
```

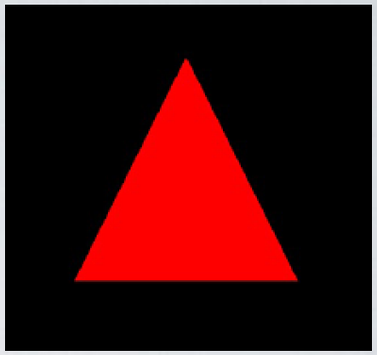
WEBGL绘制

- WebGL绘制一个红色的三角形所需信息：

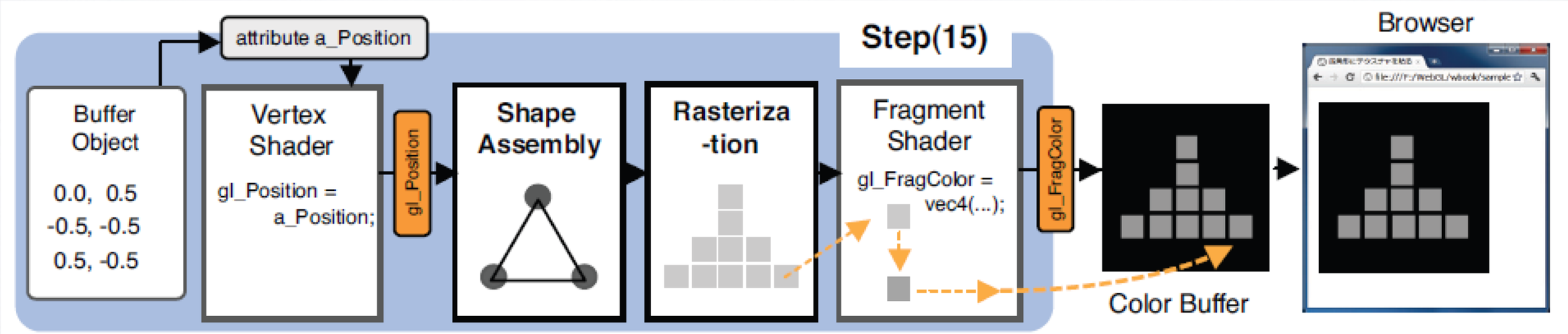
  1. 三个顶点坐标
  2. 它们组成三角形
  3. 颜色是红色

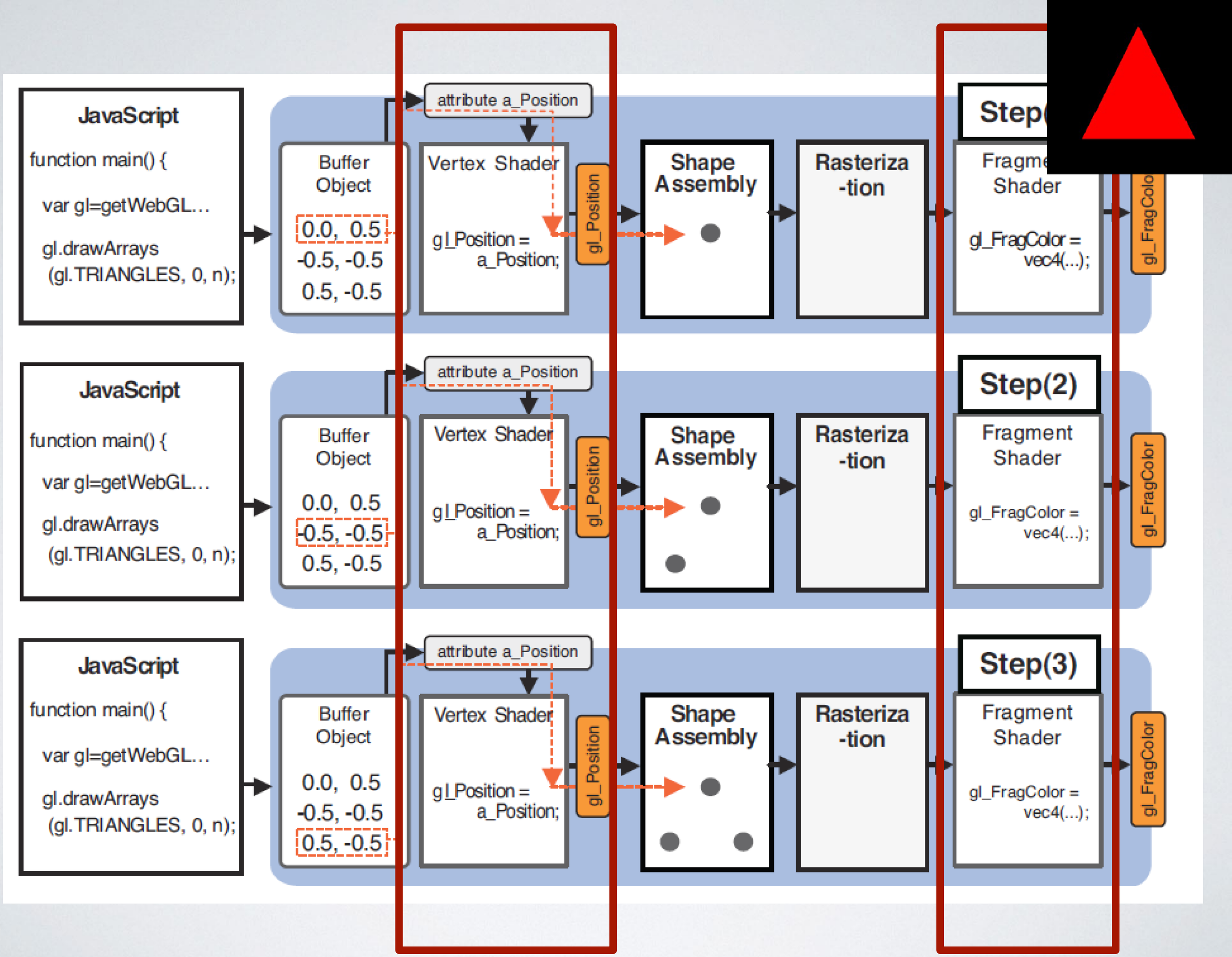


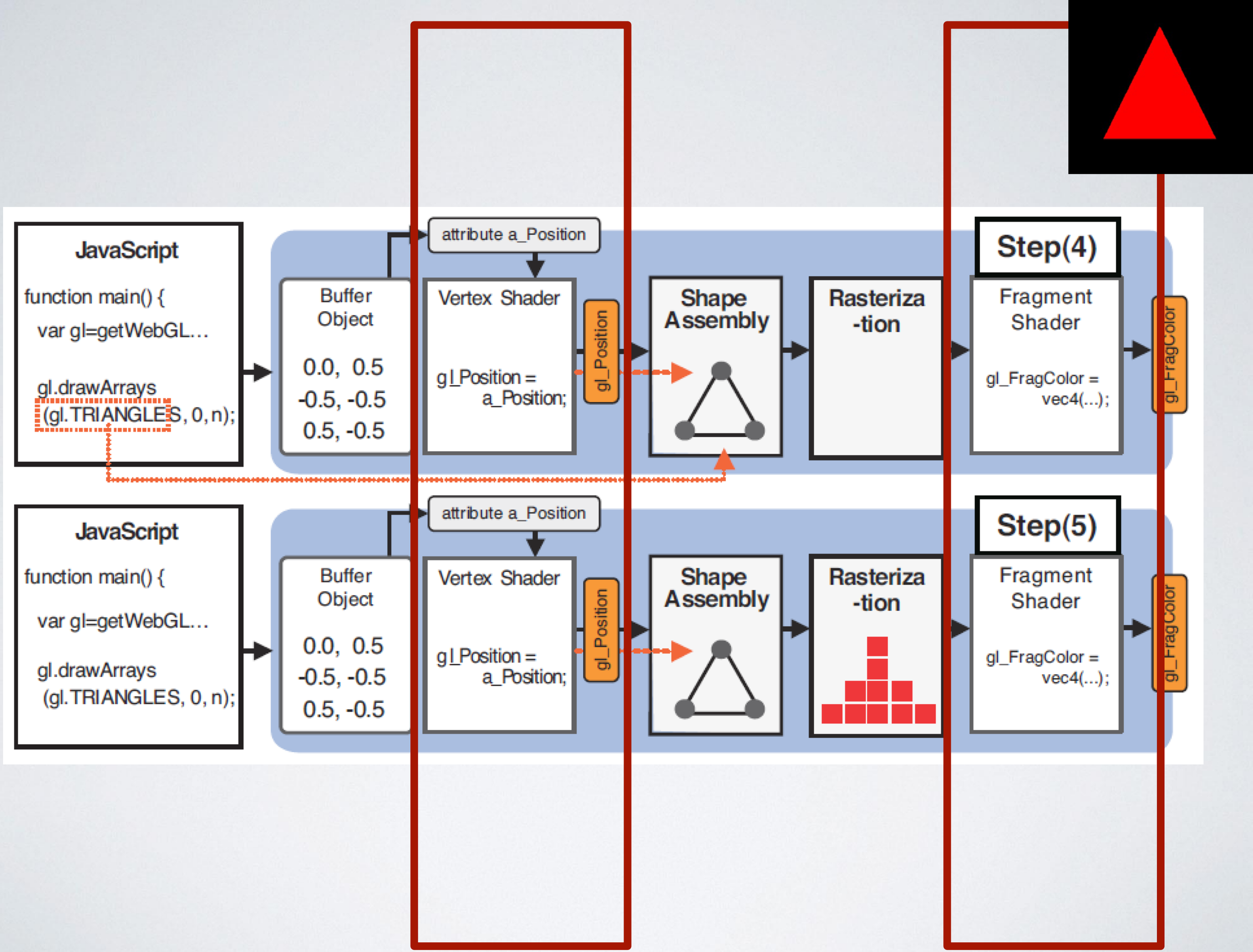
WEBGL绘制

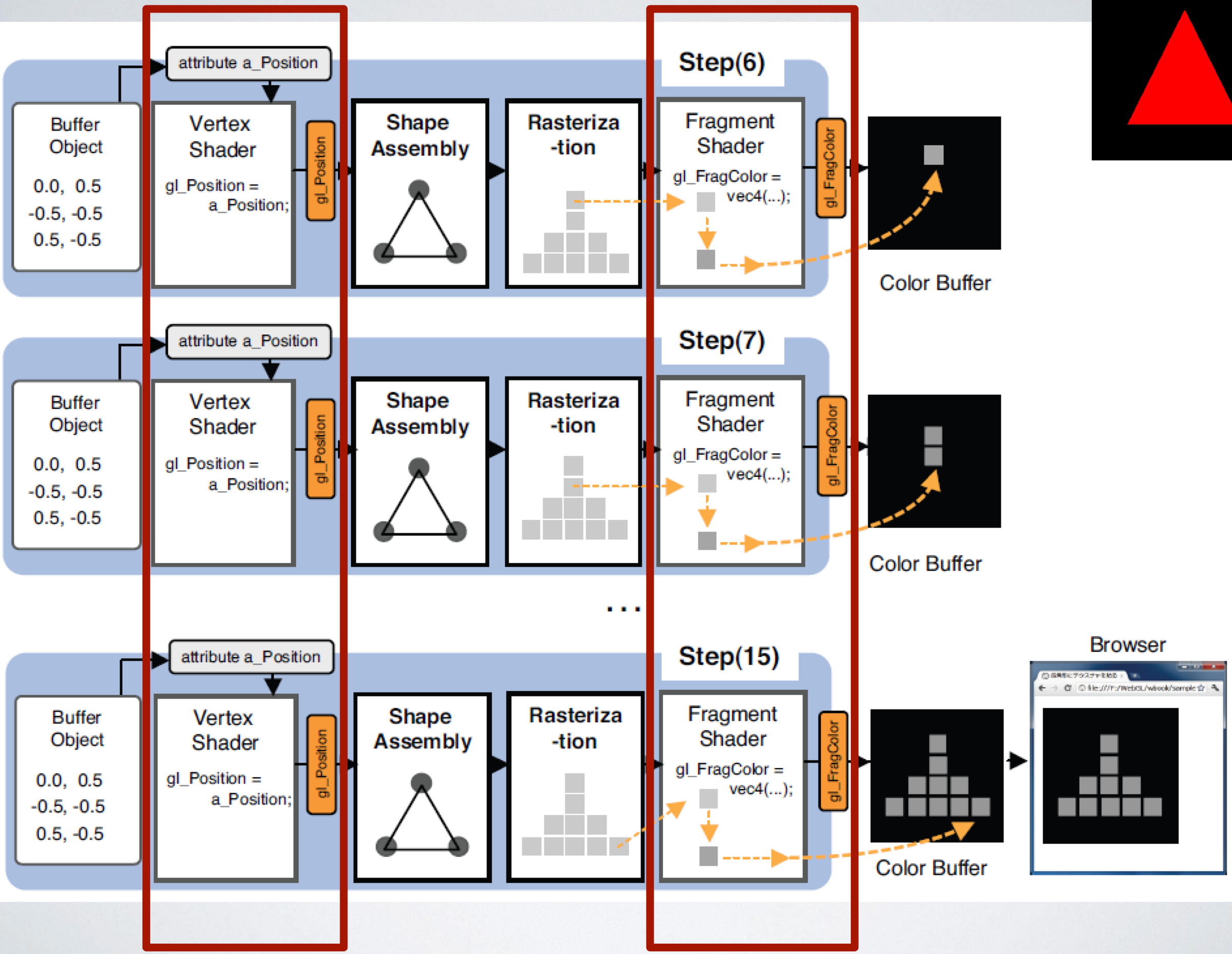


- 要绘制的图形信息**通常**由Javscript代码传递给WebGL系统并在Shader的控制下来绘制：




JavaScript
function main() {
var gl=getWebGL…
gl.drawArrays
(gl.TRIANGLES, 0, n);
Buffer Object
0.0, 0.5
-0.5, -0.5
0.5, -0.5
attribute a_Position
Vertex Shader
gl_Position =
a_Position;
gl_Position
Shape Assembly
Rasteriza -tion
Step(
Fragme
Shader
gl_FragColor =
vec4(...);
gl_FragColo
JavaScript
function main() {
var gl=getWebGL…
gl.drawArrays
(gl.TRIANGLES, 0, n);
Buffer Object
0.0, 0.5
-0.5, -0.5
0.5, -0.5
attribute a_Position
Vertex Shader
gl_Position =
a_Position;
gl_Position
Shape Assembly
Rasteriza -tion
Step(2)
Fragment Shader
gl_FragColor =
vec4(...);
gl_FragColor
JavaScript
function main() {
var gl=getWebGL…
gl.drawArrays
(gl.TRIANGLES, 0, n);
Buffer Object
0.0, 0.5
-0.5, -0.5
0.5, -0.5
attribute a_Position
Vertex Shader
gl_Position =
a_Position;
gl_Position
Shape Assembly
Rasteriza -tion
Step(3)
Fragment Shader
gl_FragColor =
vec4(...);
gl_FragColor


JavaScript
function main() {
var gl=getWebGL...
gl.drawArrays
(gl.TRIANGLES, 0, n);
Buffer Object
0.0, 0.5
-0.5, -0.5
0.5, -0.5
attribute a_Position
Vertex Shader
gl_Position =
a_Position;
gl_Position
Shape Assembly
Rasteriza
-tion
Step(4)
Fragment Shader
gl_FragColor =
vec4(...);
gl_FragColor
JavaScript
function main() {
var gl=getWebGL...
gl.drawArrays
(gl.TRIANGLES, 0, n);
Buffer Object
0.0, 0.5
-0.5, -0.5
0.5, -0.5
attribute a_Position
Vertex Shader
gl_Position =
a_Position;
gl_Position
Shape Assembly
Rasteriza
-tion
Step(5)
Fragment Shader
gl_FragColor =
vec4(...);
gl_FragColor

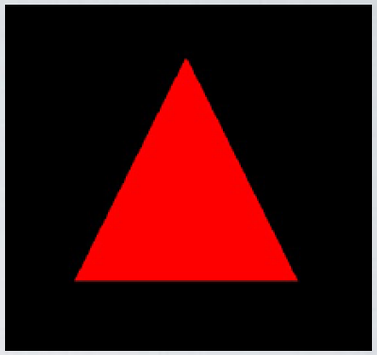

Step(6)
attribute a_Position
Buffer Object
0.0, 0.5
-0.5, -0.5
0.5, -0.5
Vertex Shader
gl_Position = a_Position;
gl_Position
Shape Assembly
Rasteriza -tion
Fragment Shader
gl_FragColor = vec4(...);
gl_FragColor
Color Buffer
Step(7)
attribute a_Position
Buffer Object
0.0, 0.5
-0.5, -0.5
0.5, -0.5
Vertex Shader
gl_Position = a_Position;
gl_Position
Shape Assembly
Rasteriza -tion
Fragment Shader
gl_FragColor = vec4(...);
gl_FragColor
Color Buffer
...
Step(15)
attribute a_Position
Buffer Object
0.0, 0.5
-0.5, -0.5
0.5, -0.5
Vertex Shader
gl_Position = a_Position;
gl_Position
Shape Assembly
Rasteriza -tion
Fragment Shader
gl_FragColor = vec4(...);
gl_FragColor
Color Buffer
Browser

WEBGL——SHADER

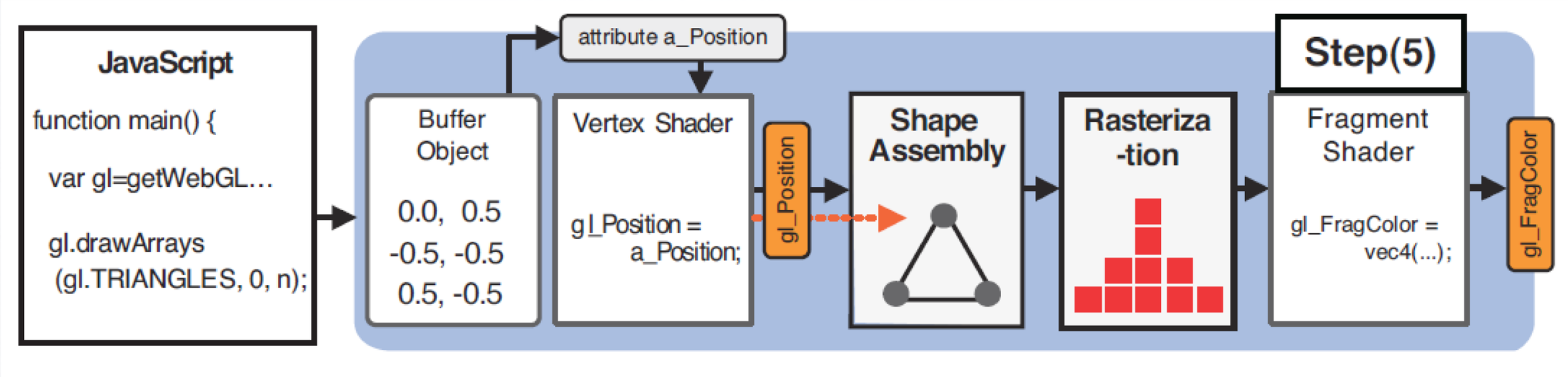
```
var VSHADER_SOURCE =
  'attribute vec4 a_Position;\n' +
  'void main() {\n' +
  '  gl_Position = a_Position;\n' +
  '}\n';

// Fragment shader program
var FSHADER_SOURCE =
  'void main() {\n' +
  '  gl_FragColor = vec4(1.0, 0.0, 0.0, 1.0);\n' +
  '}\n';
```

- 一个简单的Shader范例

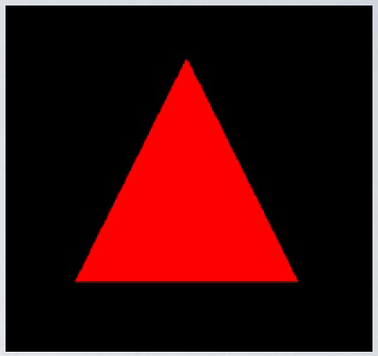
WEBGL绘制



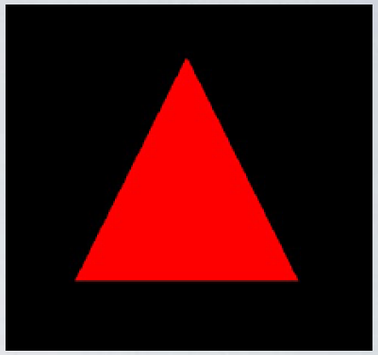
- 这些信息通常由Javscript传递给Shader并由Shader来绘制：



```
function initVertexBuffers(gl) {
  var vertices = new Float32Array([
    0, 0.5,   -0.5, -0.5,   0.5, -0.5
  ]);
  var n = 3; // The number of vertices

  // Create a buffer object
  var vertexBuffer = gl.createBuffer();
  if (!vertexBuffer) {
    console.log('Failed to create the buffer object');
    return -1;
  }

  // Bind the buffer object to target
  gl.bindBuffer(gl.ARRAY_BUFFER, vertexBuffer);
  // Write date into the buffer object
  gl.bufferData(gl.ARRAY_BUFFER, vertices, gl.STATIC_DRAW);

  var a_Position = gl.getAttribLocation(gl.program, 'a_Position');
  if (a_Position < 0) {
    console.log('Failed to get the storage location of a_Position');
    return -1;
  }
  // Assign the buffer object to a_Position variable
  gl.vertexAttribPointer(a_Position, 2, gl.FLOAT, false, 0, 0);

  // Enable the assignment to a_Position variable
  gl.enableVertexAttribArray(a_Position);

  return n;
}
```



顶点信息的传入

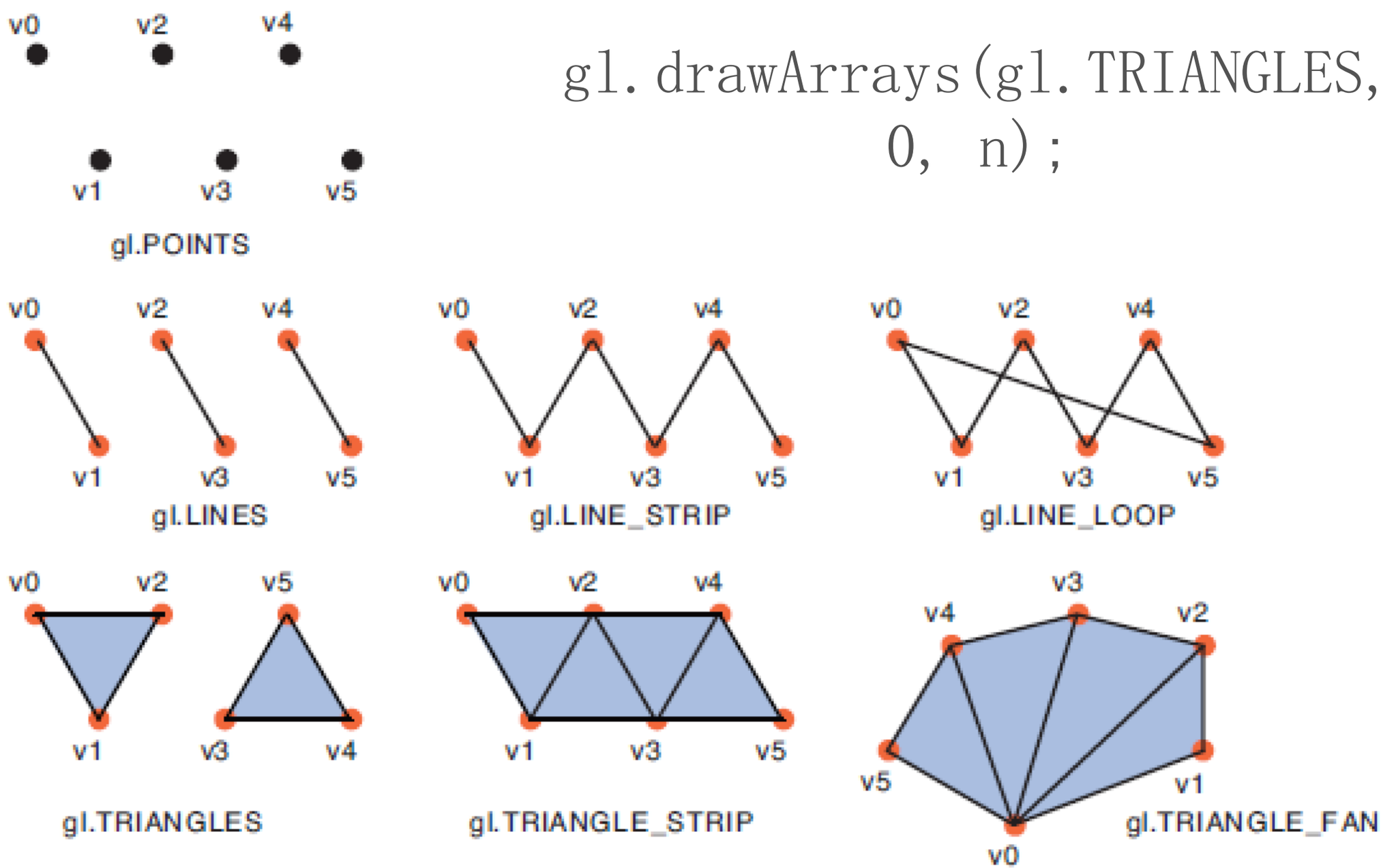


```
// Specify the color for clearing <canvas>
gl.clearColor(0, 0, 0, 1);

// Clear <canvas>
gl.clear(gl.COLOR_BUFFER_BIT);

// Draw the rectangle
gl.drawArrays(gl.TRIANGLES, 0, n);
```

“三角形”信息的传入

WEBGL基本图形与绘制

Figure 3.13 shows these basic shapes.



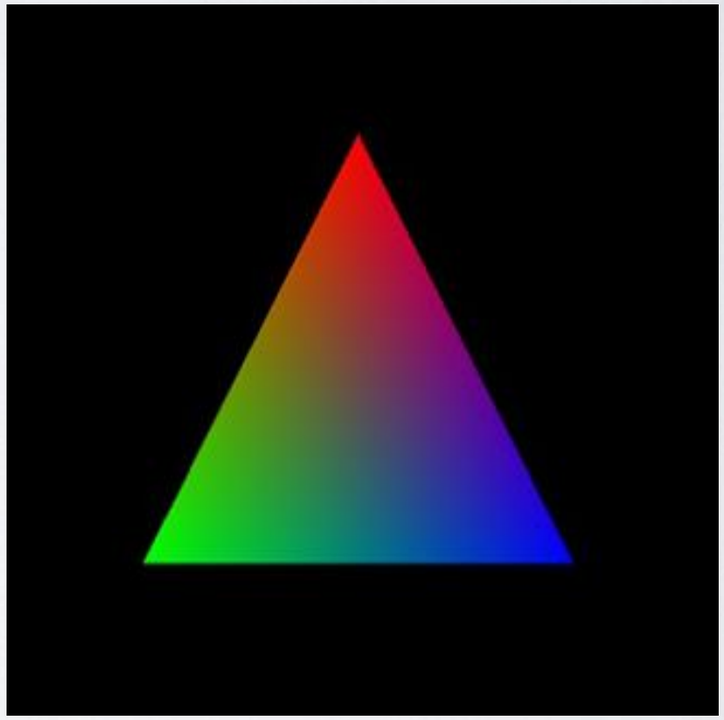

```
gl.drawArrays(gl.TRIANGLES,
0, n);
```

渐变

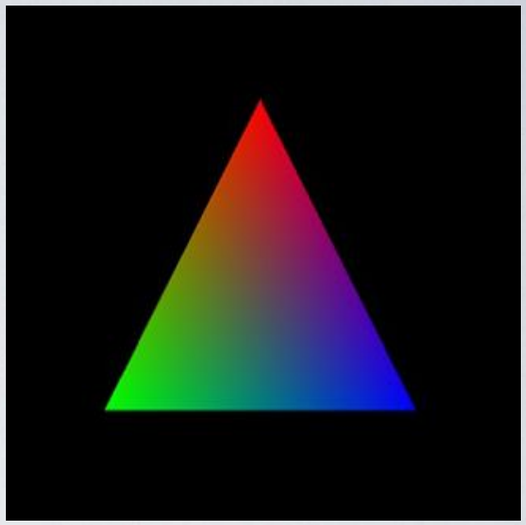
WEBGL绘制

- WebGL绘制一个渐变的三角形所需信息：

  1. 三个顶点坐标
  2. 每个顶点的颜色
  3. 组成三角形
  4. 按照渐变的方式填充三角形内部

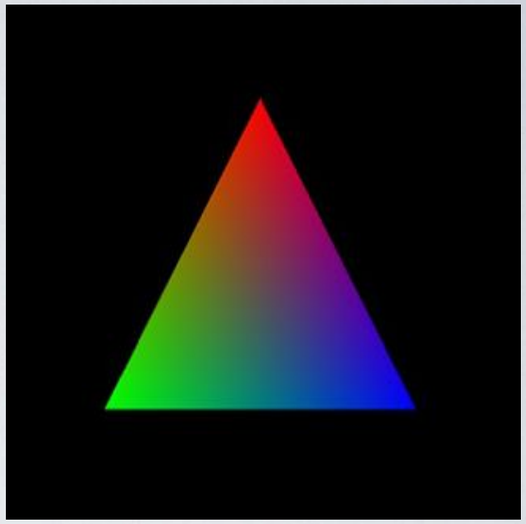


```
function initVertexBuffers(gl) {
  var verticesColors = new Float32Array([
    // Vertex coordinates and color
     0.0,  0.5,  1.0,  0.0,  0.0,
    -0.5, -0.5,  0.0,  1.0,  0.0,
     0.5, -0.5,  0.0,  0.0,  1.0,
  ]);
  var n = 3;

  // Create a buffer object
  var vertexColorBuffer = gl.createBuffer();

  // Bind the buffer object to target
  gl.bindBuffer(gl.ARRAY_BUFFER, vertexColorBuffer);
  gl.bufferData(gl.ARRAY_BUFFER, verticesColors, gl.STATIC_DRAW);

  var FSIZE = verticesColors.BYTES_PER_ELEMENT;
  //Get the storage location of a_Position, assign and enable buffer
  var a_Position = gl.getAttribLocation(gl.program, 'a_Position');
  gl.vertexAttribPointer(a_Position, 2, gl.FLOAT, false, FSIZE * 5, 0);
  gl.enableVertexAttribArray(a_Position);  // Enable the assignment of the bu

  // Get the storage location of a_Position, assign buffer and enable
  var a_Color = gl.getAttribLocation(gl.program, 'a_Color');
  gl.vertexAttribPointer(a_Color, 3, gl.FLOAT, false, FSIZE * 5, FSIZE * 2);
  gl.enableVertexAttribArray(a_Color);  // Enable the assignment of the buffe

  // Unbind the buffer object
  gl.bindBuffer(gl.ARRAY_BUFFER, null);

  return n;
}
```



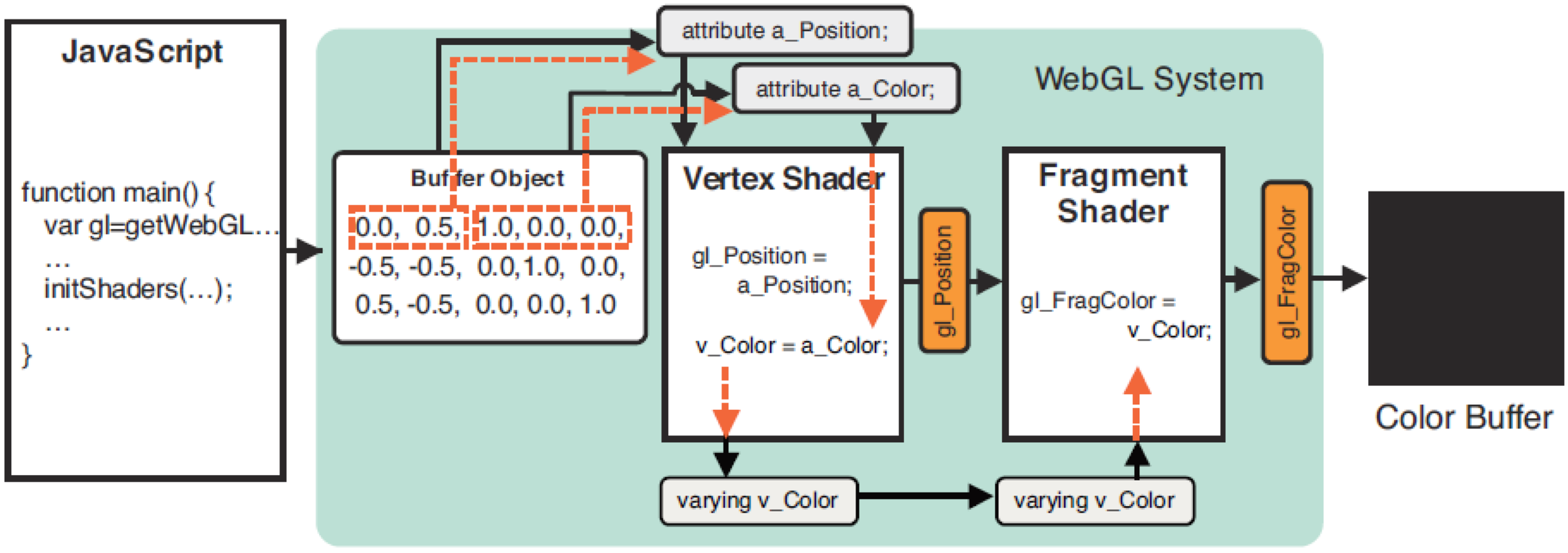
VertexBuffer
—传入更多顶点信息

```
var VSHADER_SOURCE =
  'attribute vec4 a_Position;\n' +
  'attribute vec4 a_Color;\n' +
  'varying vec4 v_Color;\n' +
  'void main() {\n' +
  '  gl_Position = a_Position;\n' +
  '  v_Color = a_Color;\n' +
  '}\n';

// Fragment shader program
var FSHADER_SOURCE =
  '#ifdef GL_ES\n' +
  'precision mediump float;\n' +
  '#endif GL_ES\n' +
  'varying vec4 v_Color;\n' +
  'void main() {\n' +
  '  gl_FragColor = v_Color;\n' +
  '}\n';
```

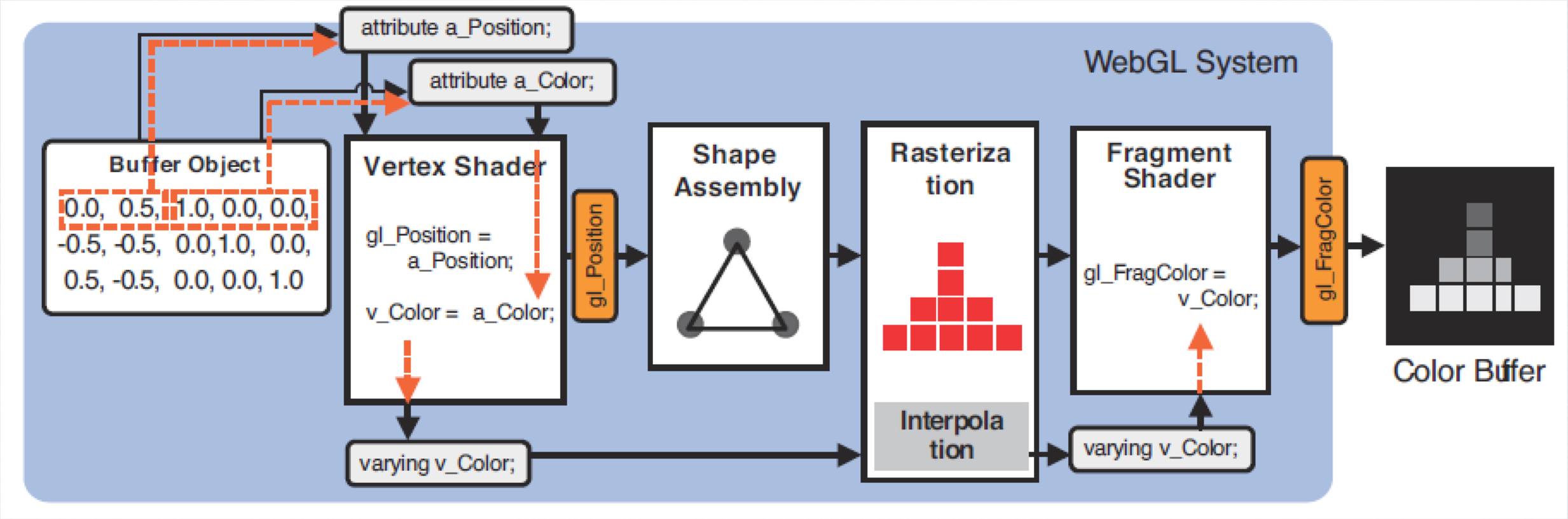


Shader
—接收更多顶
点信息

VARYING变量


JavaScript
function main() {
var gl=getWebGL...
...
initShaders(...);
...
}
attribute a_Position;
attribute a_Color;
WebGL System
Buffer Object
0.0, 0.5, 1.0, 0.0, 0.0,
-0.5, -0.5, 0.0,1.0, 0.0,
0.5, -0.5, 0.0, 0.0, 1.0
Vertex Shader
gl_Position =
a_Position;
v_Color = a_Color;
gl_Position
Fragment
Shader
gl_FragColor =
v_Color;
gl_FragColor
Color Buffer
varying v_Color
varying v_Color

插值操作（INTERPOLATION）



更多内容

WEBGL相关库的引用

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <title>Draw triangle with specification of vertex color</title>
  </head>

  <body onload="main()">
    <canvas id="webgl" width="400" height="400">
    Please use a browser that supports "canvas"
    </canvas>

    <script src="../lib/webgl-utils.js"></script>
    <script src="../lib/webgl-debug.js"></script>
    <script src="../lib/cuon-utils.js"></script>
    <script src="../lib/cuon-matrix.js"></script>
    <script src="ColoredTriangle.js"></script>
  </body>
</html>
```

MATRIX库与模型变换

```
var VSHADER_SOURCE =
  'attribute vec4 a_Position;\n' +
  'uniform mat4 u_ModelMatrix;\n' +
  'void main() {\n' +
  '  gl_Position = u_ModelMatrix * a_Position;\n' +
  '}\n';
```

```
// Create Matrix4 object for model transformation
var modelMatrix = new Matrix4();

// Calculate a model matrix
var ANGLE = 60.0; // The rotation angle
var Tx = 0.5;     // Translation distance
modelMatrix.setRotate(ANGLE, 0, 0, 1);  // Set rotation matrix
modelMatrix.translate(Tx, 0, 0);        // Multiply modelMatrix by the calcul

// Pass the model matrix to the vertex shader
var u_ModelMatrix = gl.getUniformLocation(gl.program, 'u_ModelMatrix');
if (!u_ModelMatrix) {
  console.log('Failed to get the storage location of u_xformMatrix');
  return;
}
gl.uniformMatrix4fv(u_ModelMatrix, false, modelMatrix.elements);
```

动画

```
// Start drawing
var tick = function() {
  currentAngle = animate(currentAngle);  // Update the rota
  draw(gl, n, currentAngle, modelMatrix, u_ModelMatrix);
  requestAnimationFrame(tick, canvas); // Request that the
};
tick();
}
```

requestAnimationFrame
与
setInterval (func, delay)

SHADER 切换

- initShaders(gl, VSHADER_SOURCE_1, FSHADER_SOURCE_1)
- initShaders(gl, VSHADER_SOURCE_2, FSHADER_SOURCE_2)
- 这种方法简单便捷，但是存在性能问题，详情查看cuon-util.js库中的源代码

谢谢