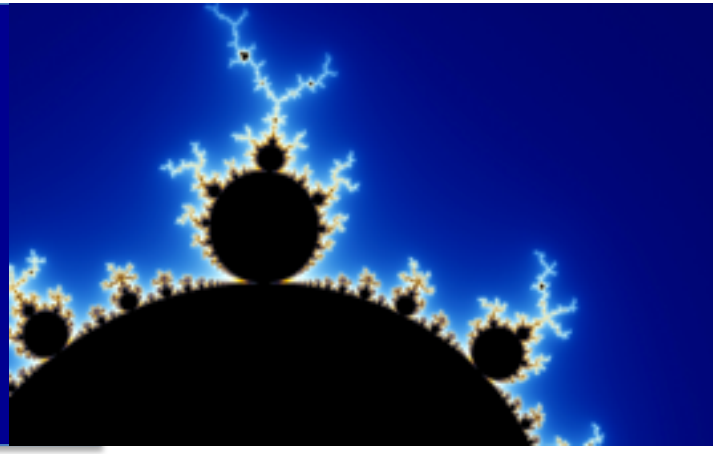
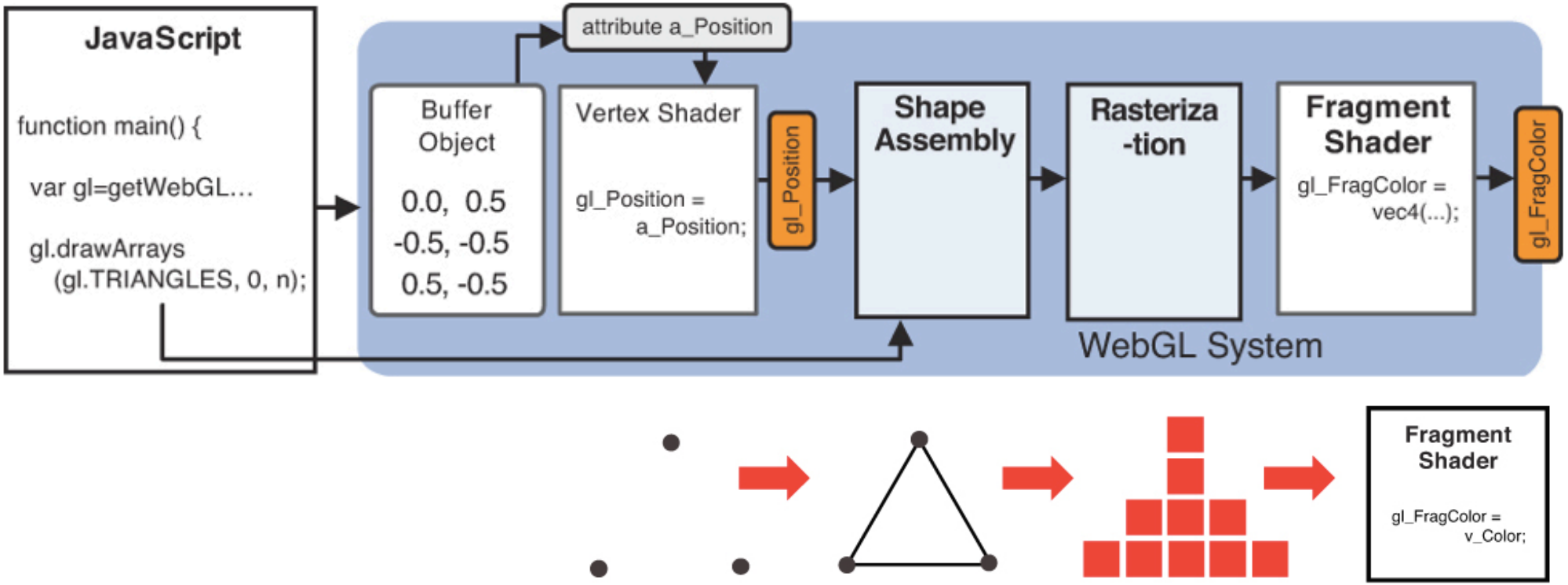
Computer Graphics



The Complete Graphics Execution Model

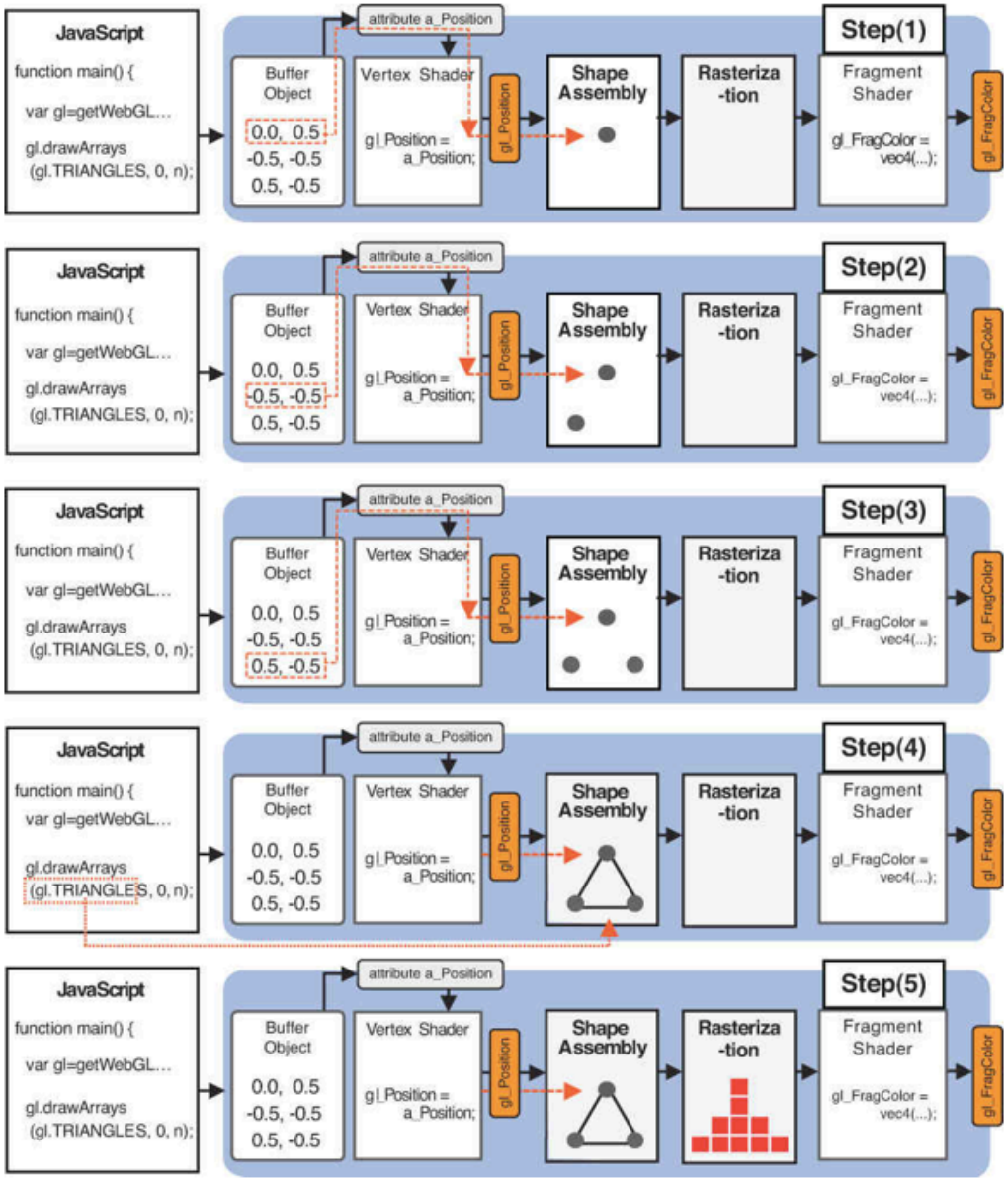
Two more steps between the vertex and fragment shaders



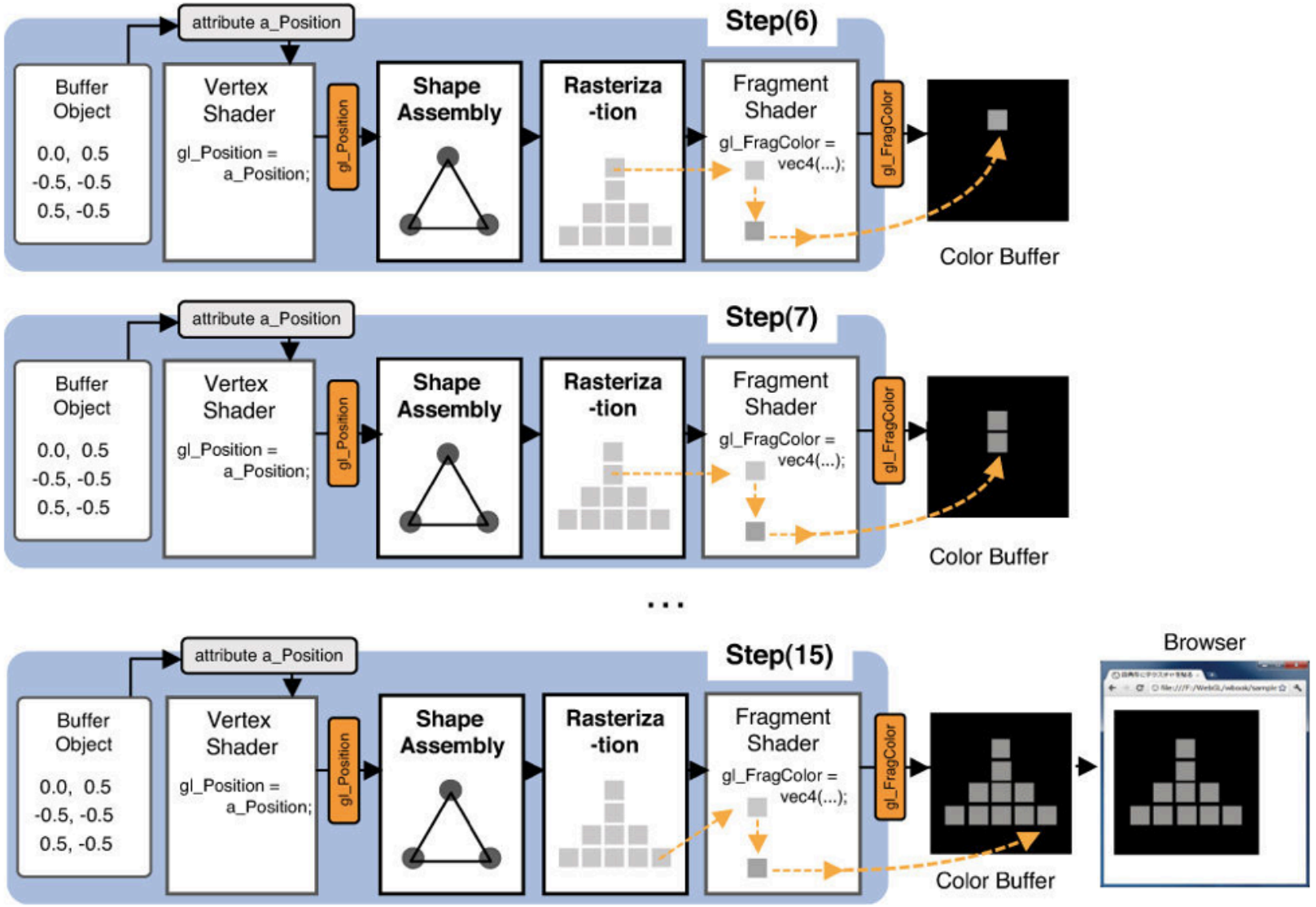
The geometric shape assembly process: In this stage, the geometric shape is assembled from the specified vertex coordinates. The first argument of `gl.drawArray()` specifies which type of shape should be assembled.

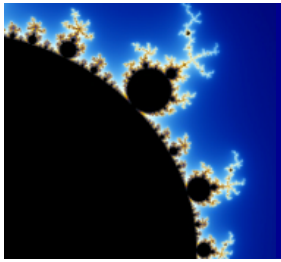
The rasterization process: In this stage, the geometric shape assembled in the geometric assembly process is converted into fragments.

The processing flow of geometric shape assembly and rasterization

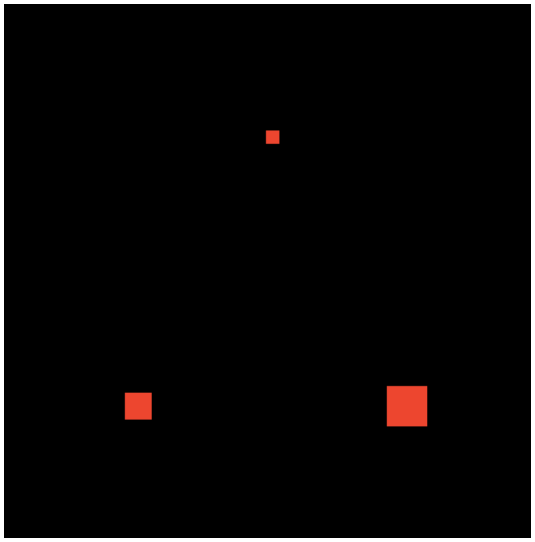


Fragment Shader Invocations



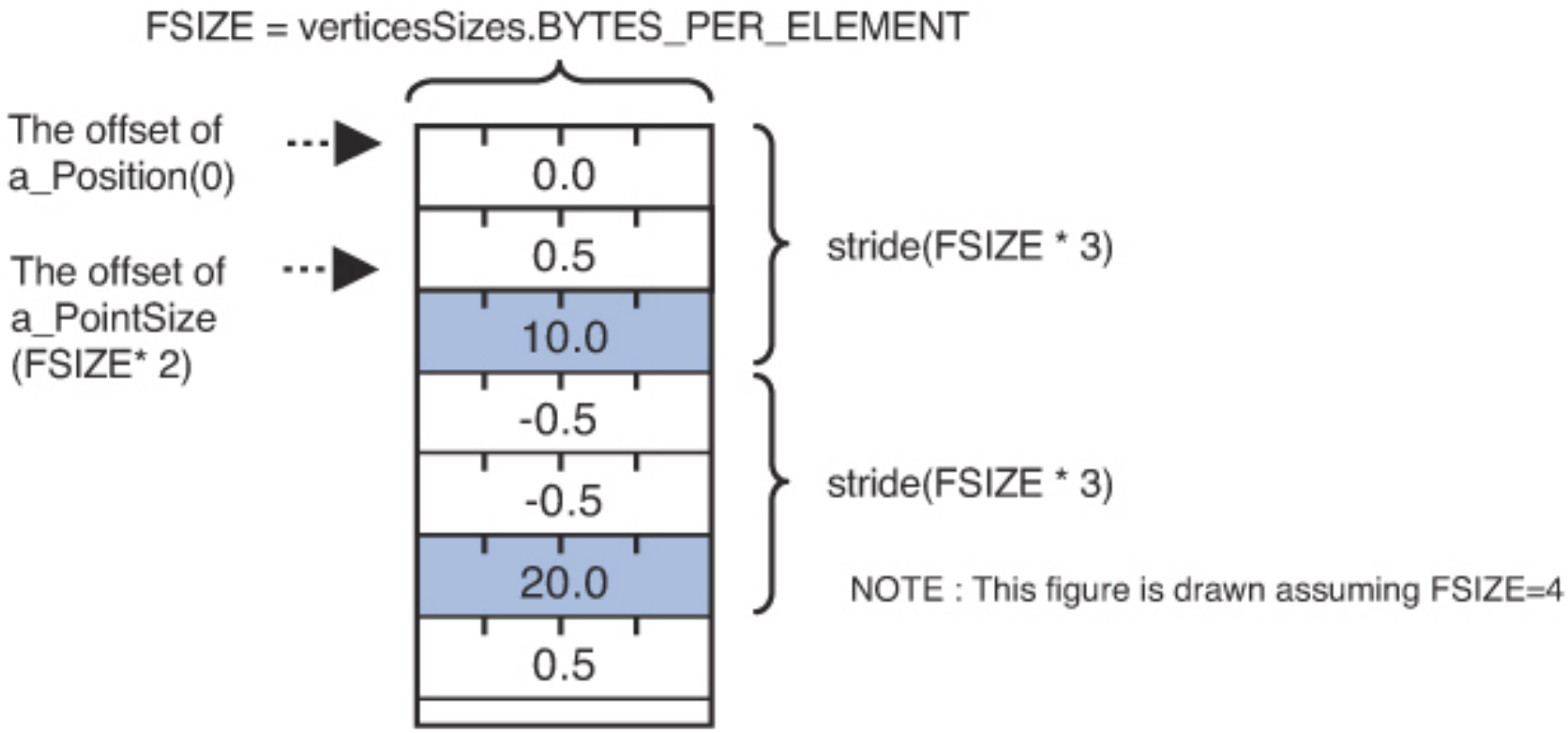


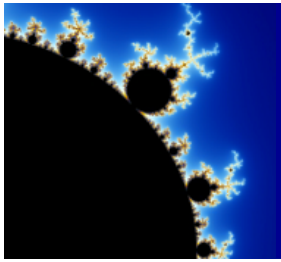
Interleaving vertex position and size



```
function initVertexBuffers(gl) {
  var verticesSizes = new Float32Array([
    // Coordinate and size of points
     0.0,  0.5,  10.0,  // the 1st point
    -0.5, -0.5,  20.0,  // the 2nd point
     0.5, -0.5,  30.0   // the 3rd point
  ]);
  var n = 3; // The number of vertices

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

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

  var FSIZE = verticesSizes.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 * 3, 0);
  gl.enableVertexAttribArray(a_Position);  // Enable the assignment of the buffer object

  // Get the storage location of a_PointSize
  var a_PointSize = gl.getAttribLocation(gl.program, 'a_PointSize');
  gl.vertexAttribPointer(a_PointSize, 1, gl.FLOAT, false, FSIZE * 3, FSIZE * 2);
  gl.enableVertexAttribArray(a_PointSize);  // Enable buffer allocation

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

Passing vertex position and color information together





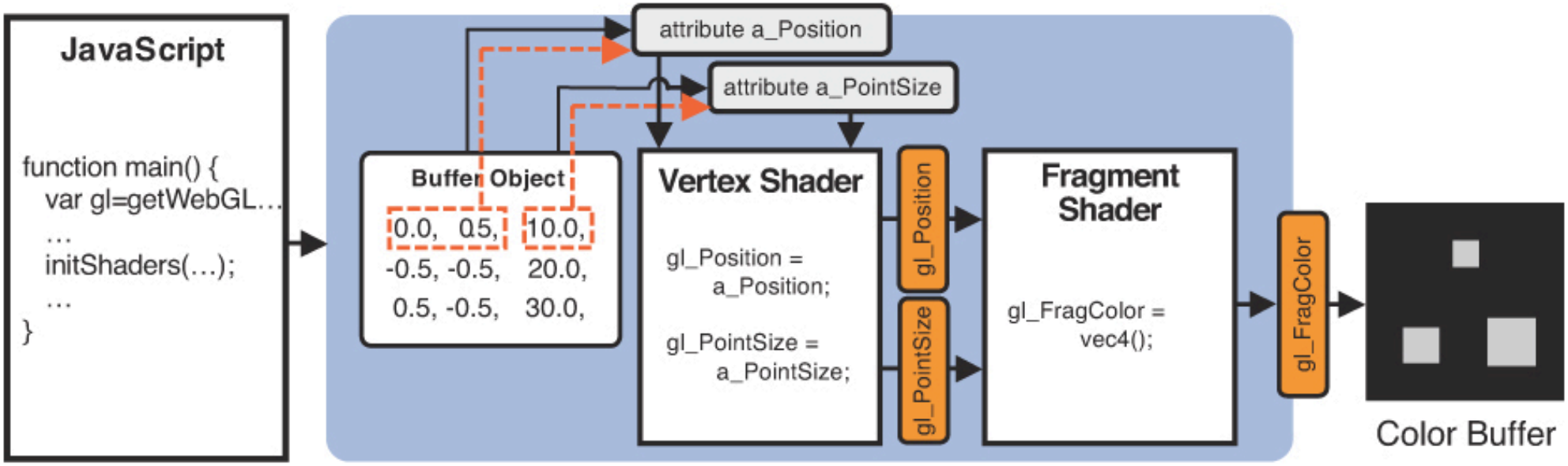
Passing vertex position and color information together

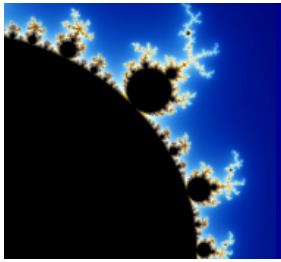
`gl.vertexAttribPointer(location, size, type, normalized, stride, offset)`

Assign the buffer object bound to `gl.ARRAY_BUFFER` to the attribute variable specified by *location*. The type and format of the data written in the buffer is also specified.

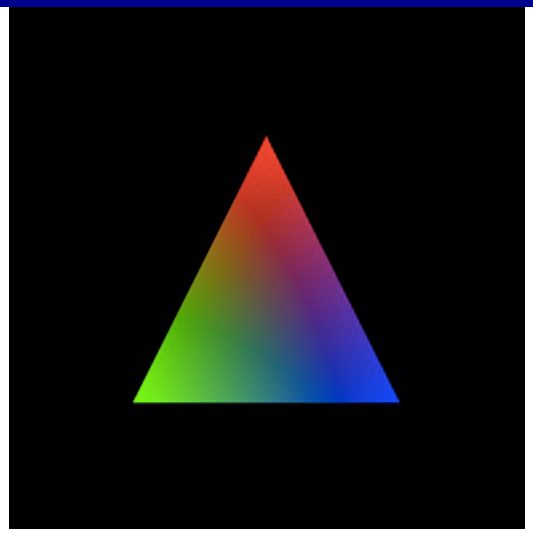
Parameters	location	Specifies the storage location of the attribute variable.
	size	Specifies the number of components per vertex in the buffer object (valid values are 1 to 4).
	type	Specifies the data format (in this case, `gl.FLOAT`)
	normalized	`true` or `false`. Used to indicate whether non-`float` data should be normalized to [0, 1] or [–1, 1].
	stride	Specifies the stride length (in bytes) to get vertex data; that is, the number of bytes between each vertex element
	offset	Specifies the offset (in bytes) in a buffer object to indicate where the vertex data is stored from. If the data is stored from the beginning. then offset is 0.

Passing vertex position and color information together



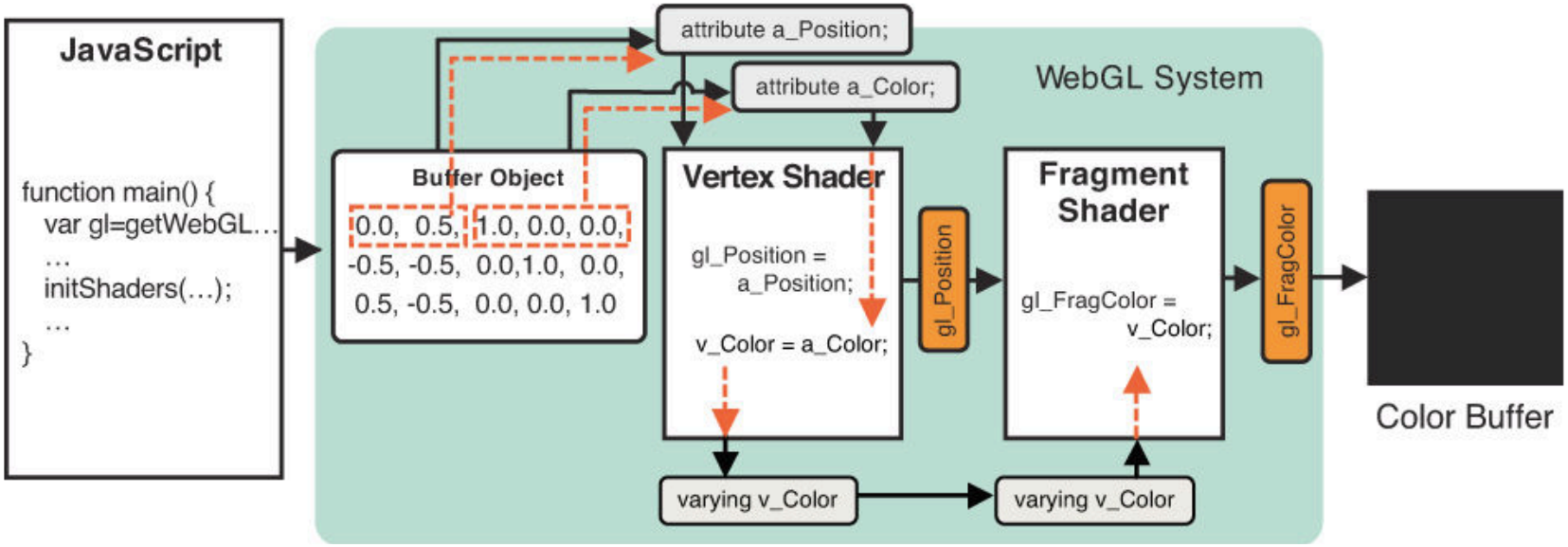


Interleaving vertex position and color



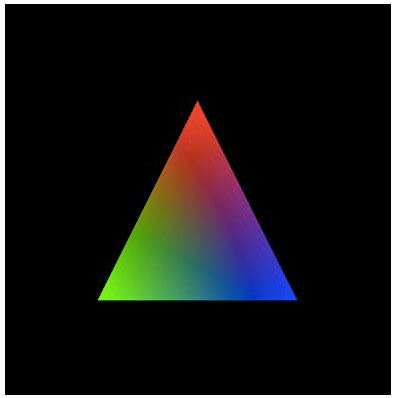
```
var verticesColors = new Float32Array([
  // Vertex coordinates and color
   0.0,  0.5,  1.0,  0.0,  0.0,
  -0.5, -0.5,  0.0,  1.0,  0.0,
   0.5, -0.5,  0.0,  0.0,  1.0,
]);
var n = 3; // The number of vertices

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

// Write the vertex coordinates and colors to the buffer object
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 buffer object

// 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 buffer object
```

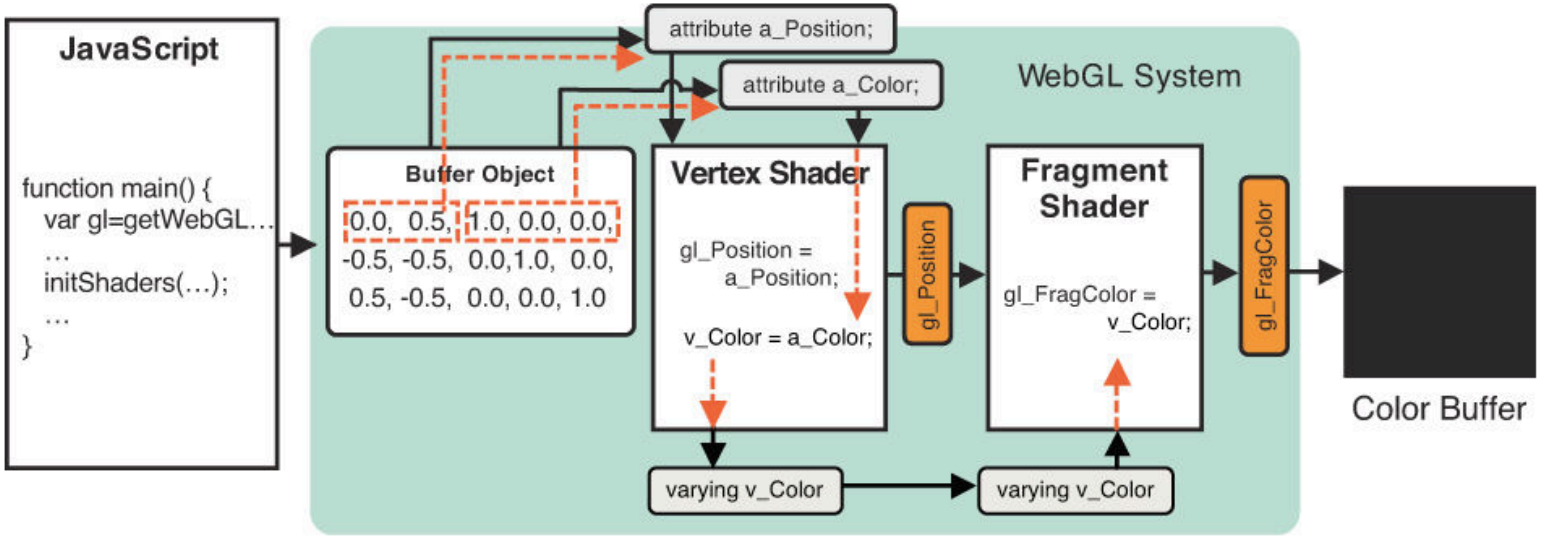
Behavior of varying variable

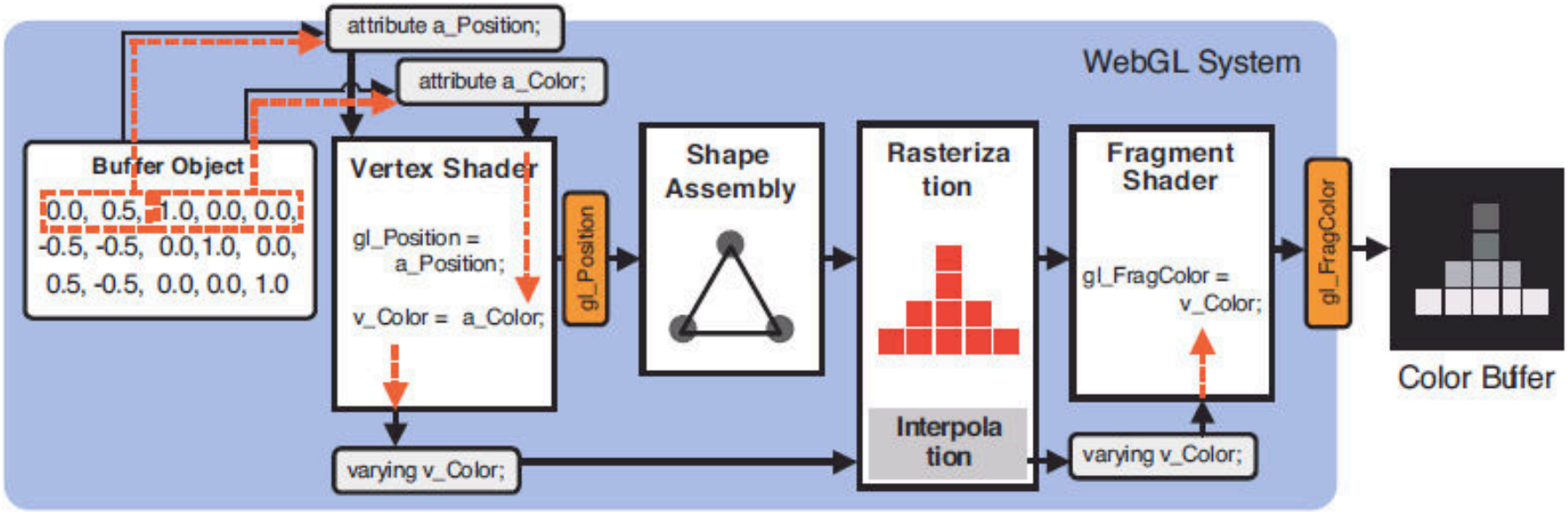



```
<script id="vertex-shader" type="x-shader/x-vertex">
attribute  vec4 a_Position;
attribute  vec4 a_Color;
varying vec4 v_Color;
void main() {
   gl_Position =  a_Position;
   v_Color = a_Color;
}
</script>

<script id="fragment-shader" type="x-shader/x-fragment">
precision mediump float;
varying vec4 v_Color;
void main() {
   gl_FragColor = v_Color;
}
</script>
```

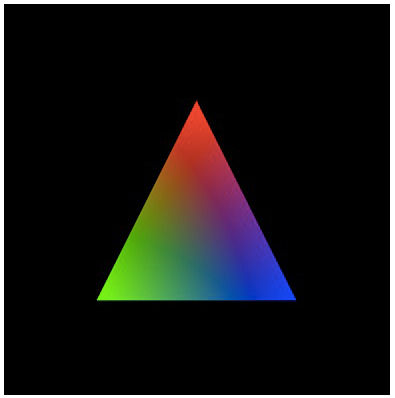
Interpolation of a varying variable

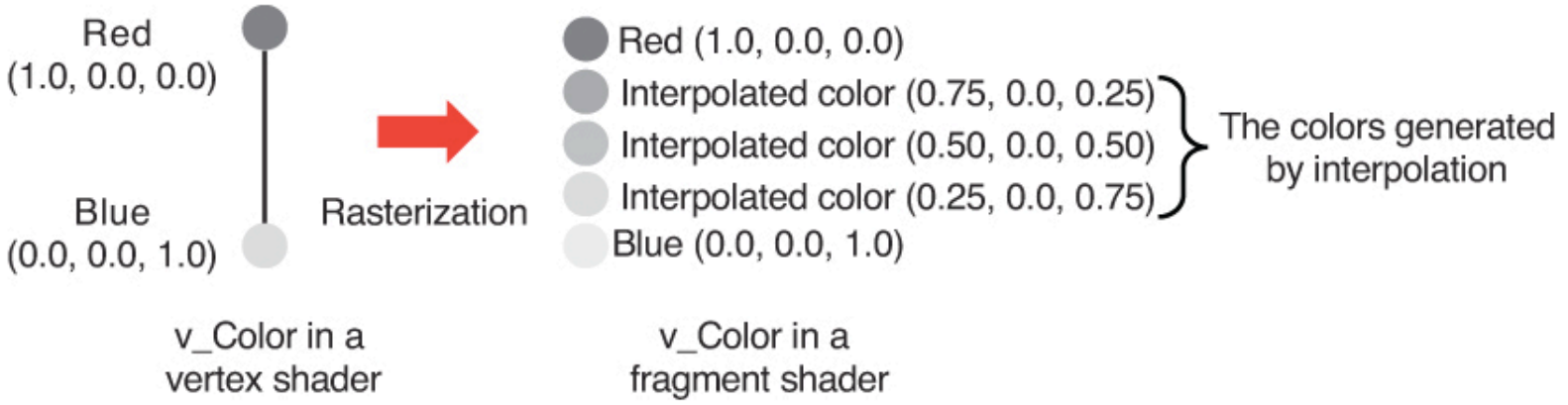


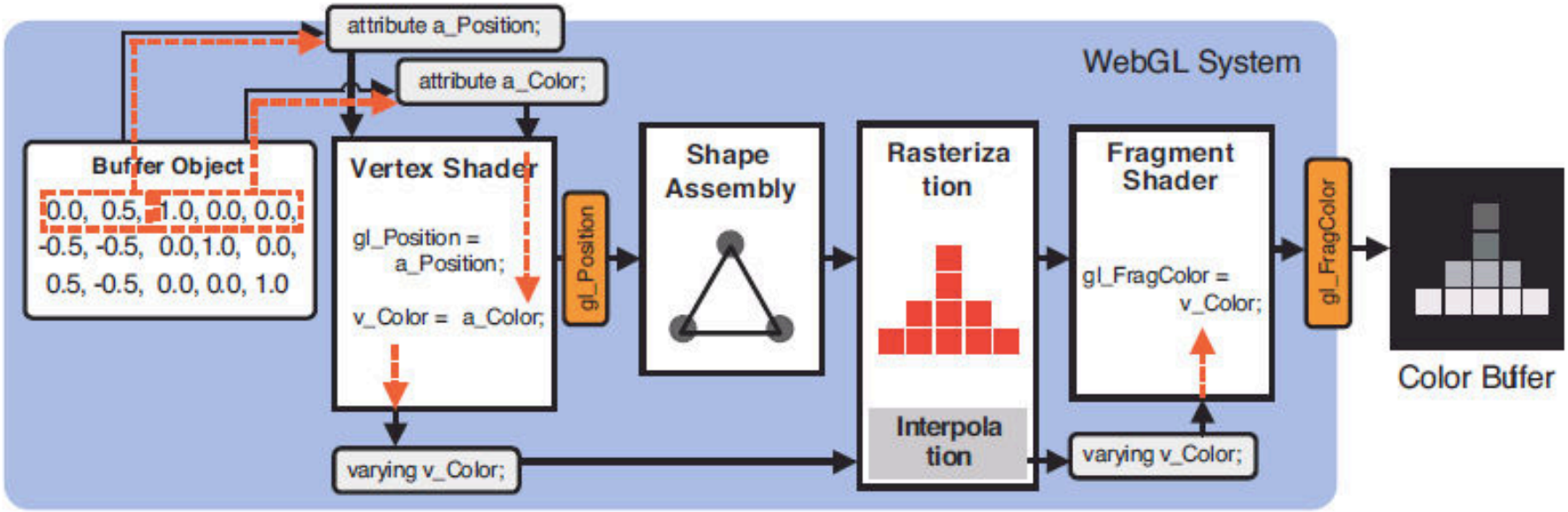


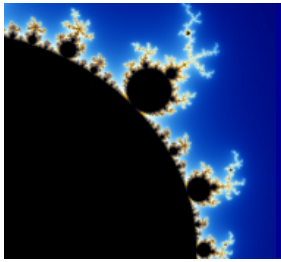


Interpolation of a varying variable









Graphics Execution Model