

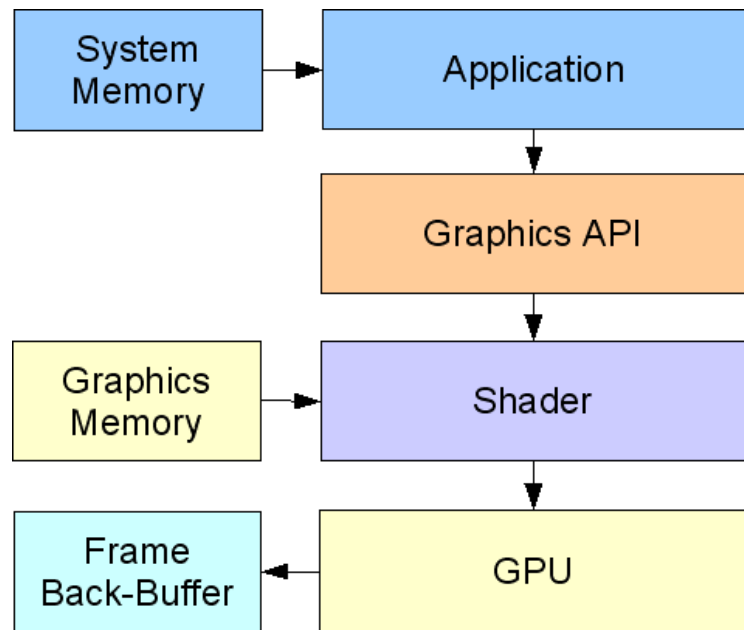
CSE4204

LAB-1 : Intro to WebGL

Mohammad Imrul Jubair

Graphics API

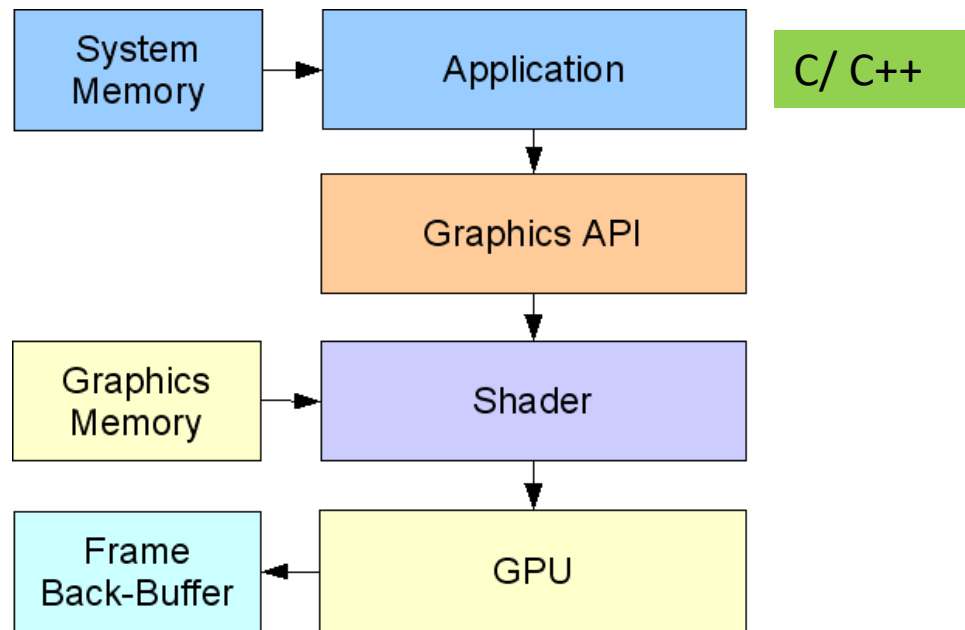
- Example: OpenGL, WebGL, Direct3D, etc.



Source: https://ict.senecacollege.ca/~chris.szalwinski/archives/gam670.071/content/shadr_p.html

OpenGL

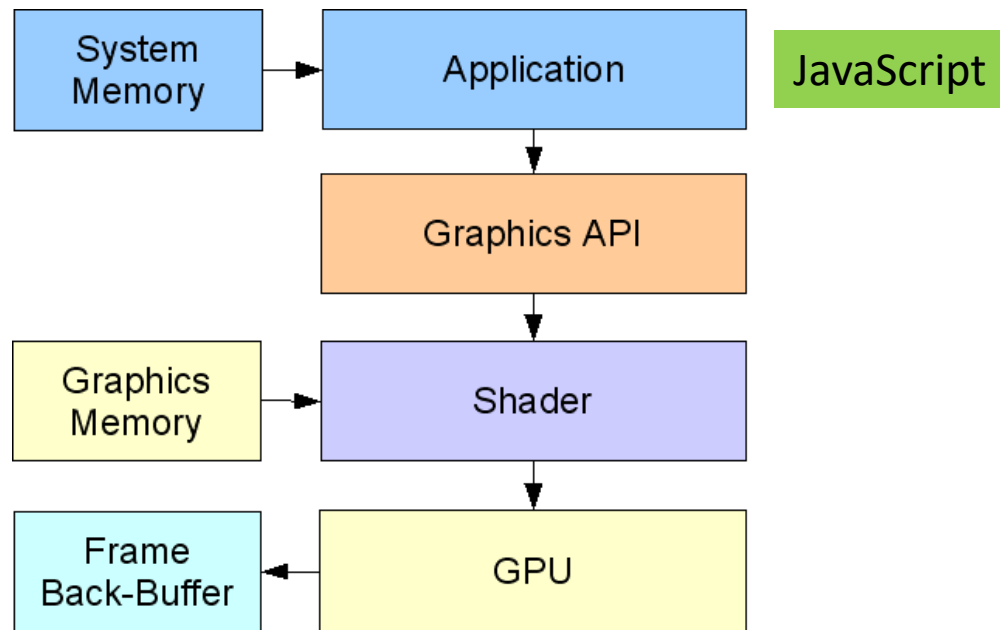
- OpenGL and OpenGL ES



Source: https://ict.senecacollege.ca/~chris.szalwinski/archives/gam670.071/content/shadr_p.html

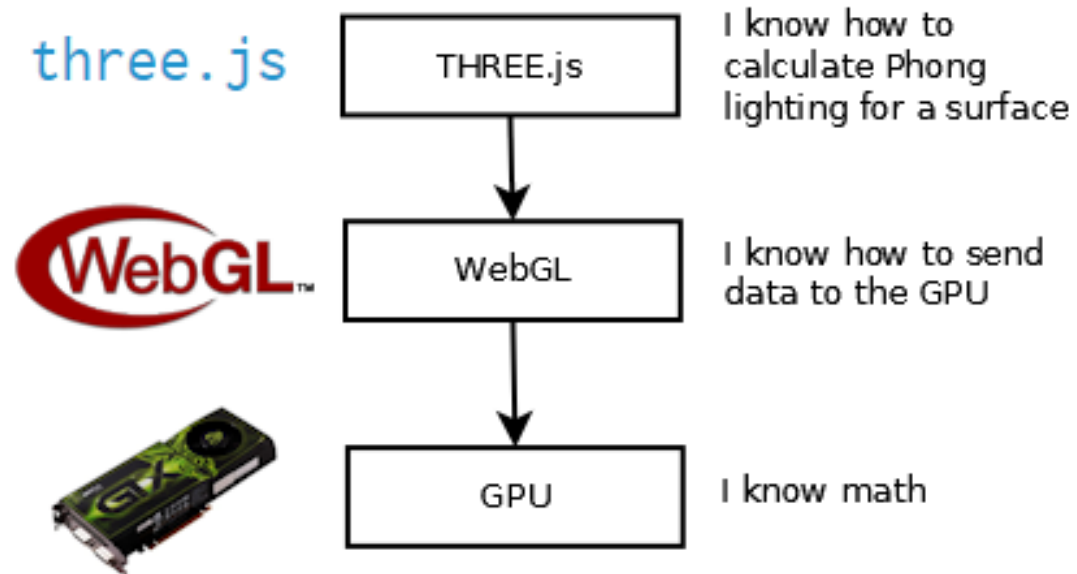
WebGL

- a JavaScript interface for OpenGL-ES-2.x API, promoted by Khronos.



Source: https://ict.senecacollege.ca/~chris.szalwinski/archives/gam670.071/content/shadr_p.html

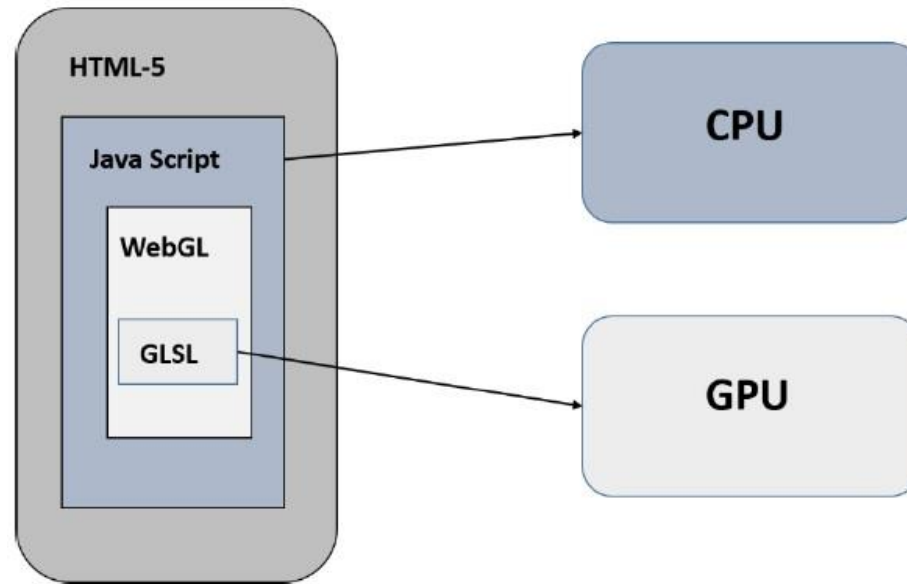
WebGL



Source: <https://cglearn.codelight.eu/pub/computer-graphics/computer-graphics>

A WebGL Program

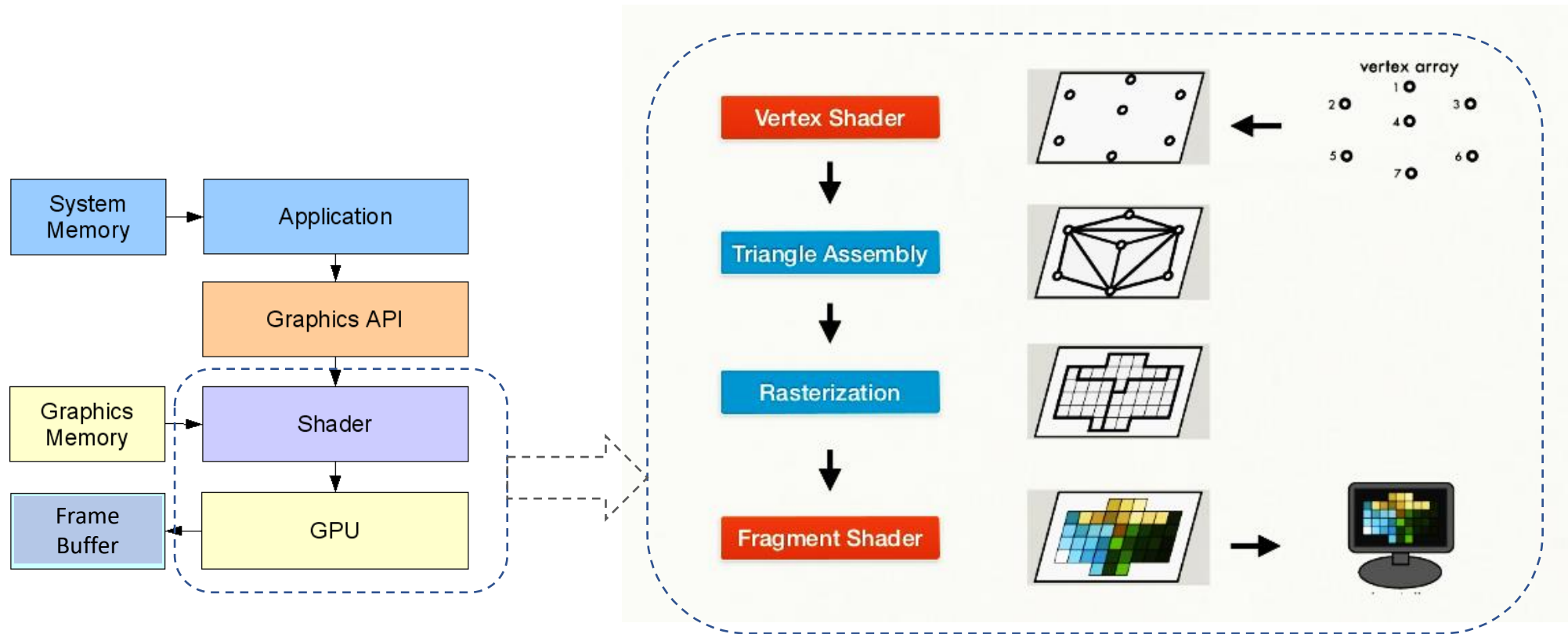
- There are two sides to any WebGL program:
 - Part – 1: written in JavaScript
 - Part – 2: written in GLSL, a language for writing "shader" programs that run on the GPU.



Source: <http://math.hws.edu/graphicsbook>

Source: https://www.tutorialspoint.com/webgl/webgl_quick_guide.htm

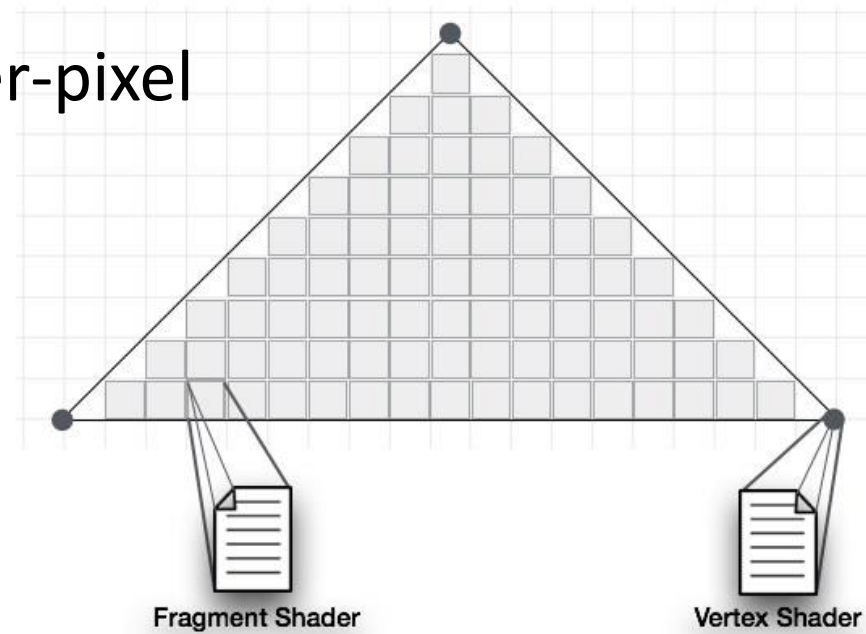
A Graphics Pipeline



Source: <https://pt.slideshare.net/senchainc/webgl-fundamentals-10013633/12>

Shaders

- Vertex Shader: Per-vertex
- Fragment Shader: Per-fragment/ per-pixel



Source: https://www.tutorialspoint.com/webgl/webgl_quick_guide.htm

A Shader Program

References:

- https://www.tutorialspoint.com/webgl/webgl_drawing_points.htm
- <http://math.hws.edu/graphicsbook/index.html>

Steps*

Get the code: <https://rb.gy/cpf3uo>

- Step 1 – Prepare the canvas and get WebGL rendering context
- Step 2 – Create and compile Shader programs
- Step 3 – Associate the shader programs with buffer objects
- Step 4 – Define the geometry and store it in buffer objects
- Step 5 – Drawing the required object

Modified from the source: https://www.tutorialspoint.com/webgl/webgl_drawing_points.htm

*Can be altered

```
var canvas = document.getElementById("webglcanvas");
var gl = canvas.getContext("webgl");

gl.clearColor(0.75, 0.75, 0.75, 1.0);
gl.clear(gl.COLOR_BUFFER_BIT);

var vertexShaderSource =
`attribute vec3 a_coords;
void main() {
    gl_Position = vec4(a_coords, 1.0);
}`;

var fragmentShaderSource =
`void main() {
    gl_FragColor = vec4(1.0, 0.0, 0.0, 1.0);
}`;

var vsh = gl.createShader( gl.VERTEX_SHADER );
gl.shaderSource( vsh, vertexShaderSource );
gl.compileShader( vsh );

var fsh = gl.createShader( gl.FRAGMENT_SHADER );
gl.shaderSource( fsh, fragmentShaderSource );
gl.compileShader( fsh );

var prog = gl.createProgram();

gl.attachShader( prog, vsh );
gl.attachShader( prog, fsh );
gl.linkProgram( prog );
gl.useProgram(prog);

var a_coords_location = gl.getAttribLocation(prog, "a_coords");

var coords = new Float32Array( [0.0, 0.0, 0.0,
                                0.0, 0.5, 0.0,
                                0.5, 0.0, 0.0] );

var a_coords_buffer = gl.createBuffer();

gl.bindBuffer(gl.ARRAY_BUFFER, a_coords_buffer);
gl.bufferData(gl.ARRAY_BUFFER, coords, gl.STATIC_DRAW);
gl.vertexAttribPointer(a_coords_location, 3, gl.FLOAT, false, 0, 0);
gl.enableVertexAttribArray(a_coords_location);
gl.drawArrays(gl.TRIANGLES, 0, 3);
```

Step – 1: Canvas and WebGL rendering context

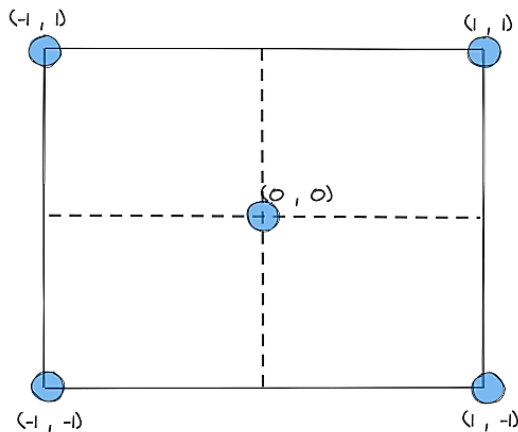
```
<canvas
```

```
id="webglcanvas" width="500" height="500">
```

```
</canvas>
```

```
var canvas = document.getElementById("webglcanvas");
```

```
var gl = canvas.getContext("webgl");
```



```
var canvas = document.getElementById("webglcanvas");
var gl = canvas.getContext("webgl");

gl.clearColor(0.75, 0.75, 0.75, 1.0);
gl.clear(gl.COLOR_BUFFER_BIT);

var vertexShaderSource =
`attribute vec3 a_coords;
void main() {
    gl_Position = vec4(a_coords, 1.0);
}`;

var fragmentShaderSource =
`void main() {
    gl_FragColor = vec4(1.0, 0.0, 0.0, 1.0);
}`;

var vsh = gl.createShader( gl.VERTEX_SHADER );
gl.shaderSource( vsh, vertexShaderSource );
gl.compileShader( vsh );

var fsh = gl.createShader( gl.FRAGMENT_SHADER );
gl.shaderSource( fsh, fragmentShaderSource );
gl.compileShader( fsh );

var prog = gl.createProgram();

gl.attachShader( prog, vsh );
gl.attachShader( prog, fsh );
gl.linkProgram( prog );
gl.useProgram( prog );

var a_coords_location = gl.getAttribLocation( prog, "a_coords" );

var coords = new Float32Array( [0.0, 0.0, 0.0,
                                0.0, 0.5, 0.0,
                                0.5, 0.0, 0.0] );

var a_coords_buffer = gl.createBuffer();

gl.bindBuffer( gl.ARRAY_BUFFER, a_coords_buffer );
gl.bufferData( gl.ARRAY_BUFFER, coords, gl.STATIC_DRAW );
gl.vertexAttribPointer( a_coords_location, 3, gl.FLOAT, false, 0, 0 );
gl.enableVertexAttribArray( a_coords_location );
gl.drawArrays( gl.TRIANGLES, 0, 3 );
```

Step – 1: Background and reset buffer

```
gl.clearColor(0.75, 0.75, 0.75, 1.0);
```

```
gl.clear(gl.COLOR_BUFFER_BIT);
```



```
var canvas = document.getElementById("webglcanvas");
var gl = canvas.getContext("webgl");

gl.clearColor(0.75, 0.75, 0.75, 1.0);
gl.clear(gl.COLOR_BUFFER_BIT);

var vertexShaderSource =
`attribute vec3 a_coords;
void main() {
    gl_Position = vec4(a_coords, 1.0);
}`;

var fragmentShaderSource =
`void main() {
    gl_FragColor = vec4(1.0, 0.0, 0.0, 1.0);
}`;

var vsh = gl.createShader( gl.VERTEX_SHADER );
gl.shaderSource( vsh, vertexShaderSource );
gl.compileShader( vsh );

var fsh = gl.createShader( gl.FRAGMENT_SHADER );
gl.shaderSource( fsh, fragmentShaderSource );
gl.compileShader( fsh );

var prog = gl.createProgram();

gl.attachShader( prog, vsh );
gl.attachShader( prog, fsh );
gl.linkProgram( prog );
gl.useProgram(prog);

var a_coords_location = gl.getAttribLocation(prog, "a_coords");

var coords = new Float32Array( [0.0, 0.0, 0.0,
                                0.0, 0.5, 0.0,
                                0.5, 0.0, 0.0] );

var a_coords_buffer = gl.createBuffer();

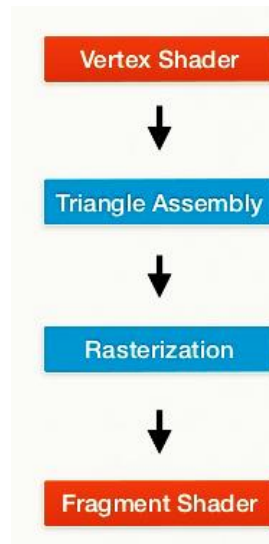
gl.bindBuffer(gl.ARRAY_BUFFER, a_coords_buffer);
gl.bufferData(gl.ARRAY_BUFFER, coords, gl.STATIC_DRAW);
gl.vertexAttribPointer(a_coords_location, 3, gl.FLOAT, false, 0, 0);
gl.enableVertexAttribArray(a_coords_location);
gl.drawArrays(gl.TRIANGLES, 0, 3);
```

Step – 2: Vertex Shader

```
var vertexShaderSource =  
  
`attribute vec3 a_coords;  
void main() {  
    gl_Position = vec4(a_coords, 1.0);  
}`;
```

The diagram shows a green box labeled 'vertices' with an arrow pointing to the variable 'a_coords' in the code. Another arrow points from 'gl_Position' to a green box labeled 'Clipping'.

Clipping



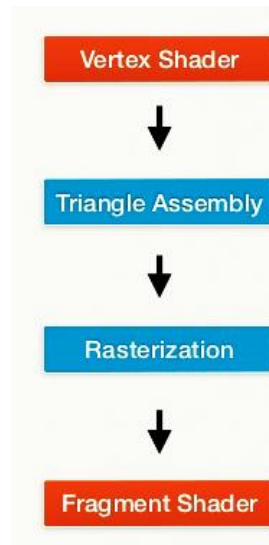
```
var canvas = document.getElementById("webglcanvas");  
var gl = canvas.getContext("webgl");  
  
gl.clearColor(0.75, 0.75, 0.75, 1.0);  
gl.clear(gl.COLOR_BUFFER_BIT);  
  
var vertexShaderSource =  
`attribute vec3 a_coords;  
void main() {  
    gl_Position = vec4(a_coords, 1.0);  
}`;  
  
var fragmentShaderSource =  
`void main() {  
    gl_FragColor = vec4(1.0, 0.0, 0.0, 1.0);  
}`;  
  
var vsh = gl.createShader( gl.VERTEX_SHADER );  
gl.shaderSource( vsh, vertexShaderSource );  
gl.compileShader( vsh );  
  
var fsh = gl.createShader( gl.FRAGMENT_SHADER );  
gl.shaderSource( fsh, fragmentShaderSource );  
gl.compileShader( fsh );  
  
var prog = gl.createProgram();  
  
gl.attachShader( prog, vsh );  
gl.attachShader( prog, fsh );  
gl.linkProgram( prog );  
gl.useProgram( prog );  
  
var a_coords_location = gl.getAttribLocation( prog, "a_coords" );  
  
var coords = new Float32Array( [0.0, 0.0, 0.0,  
                                0.0, 0.5, 0.0,  
                                0.5, 0.0, 0.0] );  
  
var a_coords_buffer = gl.createBuffer();  
  
gl.bindBuffer( gl.ARRAY_BUFFER, a_coords_buffer );  
gl.bufferData( gl.ARRAY_BUFFER, coords, gl.STATIC_DRAW );  
gl.vertexAttribPointer( a_coords_location, 3, gl.FLOAT, false, 0, 0 );  
gl.enableVertexAttribArray( a_coords_location );  
gl.drawArrays( gl.TRIANGLES, 0, 3 );
```

Step – 2: Vertex Shader

```
var vertexShaderSource =
```

```
`attribute vec3 a_coords;  
void main() {  
    gl_Position = vec4(a_coords, 1.0);  
}`;
```

Per-vertex



```
var canvas = document.getElementById("webglcanvas");  
var gl = canvas.getContext("webgl");  
  
gl.clearColor(0.75, 0.75, 0.75, 1.0);  
gl.clear(gl.COLOR_BUFFER_BIT);  
  
var vertexShaderSource =  
`attribute vec3 a_coords;  
void main() {  
    gl_Position = vec4(a_coords, 1.0);  
}`;  
  
var fragmentShaderSource =  
`void main() {  
    gl_FragColor = vec4(1.0, 0.0, 0.0, 1.0);  
}`;  
  
var vsh = gl.createShader( gl.VERTEX_SHADER );  
gl.shaderSource( vsh, vertexShaderSource );  
gl.compileShader( vsh );  
  
var fsh = gl.createShader( gl.FRAGMENT_SHADER );  
gl.shaderSource( fsh, fragmentShaderSource );  
gl.compileShader( fsh );  
  
var prog = gl.createProgram();  
  
gl.attachShader( prog, vsh );  
gl.attachShader( prog, fsh );  
gl.linkProgram( prog );  
gl.useProgram(prog);  
  
var a_coords_location = gl.getAttribLocation(prog, "a_coords");  
  
var coords = new Float32Array( [0.0, 0.0, 0.0,  
                                0.0, 0.5, 0.0,  
                                0.5, 0.0, 0.0] );  
  
var a_coords_buffer = gl.createBuffer();  
  
gl.bindBuffer(gl.ARRAY_BUFFER, a_coords_buffer);  
gl.bufferData(gl.ARRAY_BUFFER, coords, gl.STATIC_DRAW);  
gl.vertexAttribPointer(a_coords_location, 3, gl.FLOAT, false, 0, 0);  
gl.enableVertexAttribArray(a_coords_location);  
gl.drawArrays(gl.TRIANGLES, 0, 3);
```

Step – 2: Fragment Shader

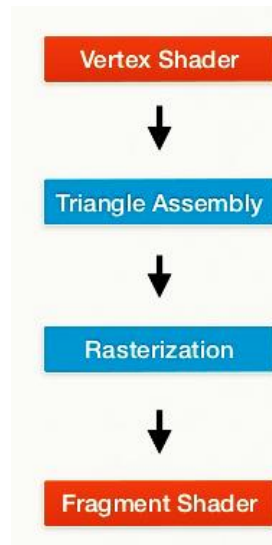
```
var fragmentShaderSource =
```

```
`void main() {
```

```
    gl_FragColor = vec4(1.0, 0.0, 0.0, 1.0);
```

```
};`
```

Colored pixels



```
var canvas = document.getElementById("webglcanvas");
var gl = canvas.getContext("webgl");

gl.clearColor(0.75, 0.75, 0.75, 1.0);
gl.clear(gl.COLOR_BUFFER_BIT);

var vertexShaderSource =
`attribute vec3 a_coords;
void main() {
    gl_Position = vec4(a_coords, 1.0);
}`;

var fragmentShaderSource =
`void main() {
    gl_FragColor = vec4(1.0, 0.0, 0.0, 1.0);
}`;

var vsh = gl.createShader( gl.VERTEX_SHADER );
gl.shaderSource( vsh, vertexShaderSource );
gl.compileShader( vsh );

var fsh = gl.createShader( gl.FRAGMENT_SHADER );
gl.shaderSource( fsh, fragmentShaderSource );
gl.compileShader( fsh );

var prog = gl.createProgram();

gl.attachShader( prog, vsh );
gl.attachShader( prog, fsh );
gl.linkProgram( prog );
gl.useProgram( prog );

var a_coords_location = gl.getAttribLocation( prog, "a_coords" );

var coords = new Float32Array( [0.0, 0.0, 0.0,
                                0.0, 0.5, 0.0,
                                0.5, 0.0, 0.0] );

var a_coords_buffer = gl.createBuffer();

gl.bindBuffer( gl.ARRAY_BUFFER, a_coords_buffer );
gl.bufferData( gl.ARRAY_BUFFER, coords, gl.STATIC_DRAW );
gl.vertexAttribPointer( a_coords_location, 3, gl.FLOAT, false, 0, 0 );
gl.enableVertexAttribArray( a_coords_location );
gl.drawArrays( gl.TRIANGLES, 0, 3 );
```

Step – 2: Create and Compile Shaders

```
var vsh = gl.createShader( gl.VERTEX_SHADER );
```

```
gl.shaderSource( vsh, vertexShaderSource );
```

```
gl.compileShader( vsh );
```

```
var canvas = document.getElementById("webglcanvas");
var gl = canvas.getContext("webgl");

gl.clearColor(0.75, 0.75, 0.75, 1.0);
gl.clear(gl.COLOR_BUFFER_BIT);

var vertexShaderSource =
`attribute vec3 a_coords;
void main() {
    gl_Position = vec4(a_coords, 1.0);
}`;

var fragmentShaderSource =
`void main() {
    gl_FragColor = vec4(1.0, 0.0, 0.0, 1.0);
}`;

var vsh = gl.createShader( gl.VERTEX_SHADER );
gl.shaderSource( vsh, vertexShaderSource );
gl.compileShader( vsh );

var fsh = gl.createShader( gl.FRAGMENT_SHADER );
gl.shaderSource( fsh, fragmentShaderSource );
gl.compileShader( fsh );

var prog = gl.createProgram();

gl.attachShader( prog, vsh );
gl.attachShader( prog, fsh );
gl.linkProgram( prog );
gl.useProgram(prog);

var a_coords_location = gl.getAttribLocation(prog, "a_coords");

var coords = new Float32Array( [0.0, 0.0, 0.0,
                                0.0, 0.5, 0.0,
                                0.5, 0.0, 0.0] );

var a_coords_buffer = gl.createBuffer();

gl.bindBuffer(gl.ARRAY_BUFFER, a_coords_buffer);
gl.bufferData(gl.ARRAY_BUFFER, coords, gl.STATIC_DRAW);
gl.vertexAttribPointer(a_coords_location, 3, gl.FLOAT, false, 0, 0);
gl.enableVertexAttribArray(a_coords_location);
gl.drawArrays(gl.TRIANGLES, 0, 3);
```


Step – 2: Create and Compile Shaders

```
var fsh = gl.createShader( gl.FRAGMENT_SHADER );
```

```
gl.shaderSource( fsh, fragmentShaderSource );
```

```
gl.compileShader( fsh );
```

```
var canvas = document.getElementById("webglcanvas");
var gl = canvas.getContext("webgl");

gl.clearColor(0.75, 0.75, 0.75, 1.0);
gl.clear(gl.COLOR_BUFFER_BIT);

var vertexShaderSource =
`attribute vec3 a_coords;
void main() {
    gl_Position = vec4(a_coords, 1.0);
}`;

var fragmentShaderSource =
`void main() {
    gl_FragColor = vec4(1.0, 0.0, 0.0, 1.0);
}`;

var vsh = gl.createShader( gl.VERTEX_SHADER );
gl.shaderSource( vsh, vertexShaderSource );
gl.compileShader( vsh );

var fsh = gl.createShader( gl.FRAGMENT_SHADER );
gl.shaderSource( fsh, fragmentShaderSource );
gl.compileShader( fsh );

var prog = gl.createProgram();

gl.attachShader( prog, vsh );
gl.attachShader( prog, fsh );
gl.linkProgram( prog );
gl.useProgram(prog);

var a_coords_location = gl.getAttribLocation(prog, "a_coords");

var coords = new Float32Array( [0.0, 0.0, 0.0,
                                0.0, 0.5, 0.0,
                                0.5, 0.0, 0.0] );

var a_coords_buffer = gl.createBuffer();

gl.bindBuffer(gl.ARRAY_BUFFER, a_coords_buffer);
gl.bufferData(gl.ARRAY_BUFFER, coords, gl.STATIC_DRAW);
gl.vertexAttribPointer(a_coords_location, 3, gl.FLOAT, false, 0, 0);
gl.enableVertexAttribArray(a_coords_location);
gl.drawArrays(gl.TRIANGLES, 0, 3);
```

Step – 2: Link shaders and use program

```
var prog = gl.createProgram();
```

```
gl.attachShader( prog, vsh );
```

```
gl.attachShader( prog, fsh );
```

```
gl.linkProgram( prog );
```

```
gl.useProgram(prog);
```

```
var canvas = document.getElementById("webglcanvas");
var gl = canvas.getContext("webgl");

gl.clearColor(0.75, 0.75, 0.75, 1.0);
gl.clear(gl.COLOR_BUFFER_BIT);

var vertexShaderSource =
`attribute vec3 a_coords;
void main() {
    gl_Position = vec4(a_coords, 1.0);
}`;

var fragmentShaderSource =
`void main() {
    gl_FragColor = vec4(1.0, 0.0, 0.0, 1.0);
}`;

var vsh = gl.createShader( gl.VERTEX_SHADER );
gl.shaderSource( vsh, vertexShaderSource );
gl.compileShader( vsh );

var fsh = gl.createShader( gl.FRAGMENT_SHADER );
gl.shaderSource( fsh, fragmentShaderSource );
gl.compileShader( fsh );

var prog = gl.createProgram();

gl.attachShader( prog, vsh );
gl.attachShader( prog, fsh );
gl.linkProgram( prog );
gl.useProgram(prog);

var a_coords_location = gl.getAttribLocation(prog, "a_coords");

var coords = new Float32Array( [0.0, 0.0, 0.0,
                                0.0, 0.5, 0.0,
                                0.5, 0.0, 0.0] );

var a_coords_buffer = gl.createBuffer();

gl.bindBuffer(gl.ARRAY_BUFFER, a_coords_buffer);
gl.bufferData(gl.ARRAY_BUFFER, coords, gl.STATIC_DRAW);
gl.vertexAttribPointer(a_coords_location, 3, gl.FLOAT, false, 0, 0);
gl.enableVertexAttribArray(a_coords_location);
gl.drawArrays(gl.TRIANGLES, 0, 3);
```

Step – 3: Associate Shaders

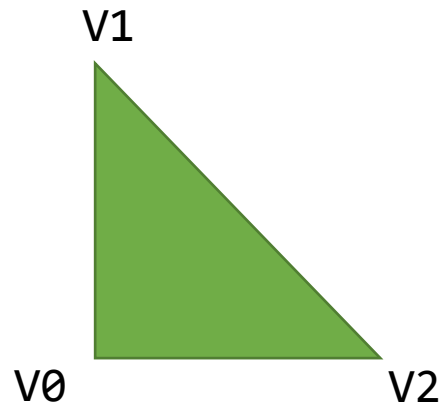
```
var a_coords_location =  
    gl.getAttribLocation(prog, "a_coords");
```

```
attribute vec3 a_coords;  
void main() {  
    gl_Position = vec4(a_coords, 1.0);  
}
```

```
var canvas = document.getElementById("webglcanvas");  
var gl = canvas.getContext("webgl");  
  
gl.clearColor(0.75, 0.75, 0.75, 1.0);  
gl.clear(gl.COLOR_BUFFER_BIT);  
  
var vertexShaderSource =  
`attribute vec3 a_coords;  
void main() {  
    gl_Position = vec4(a_coords, 1.0);  
}`;  
  
var fragmentShaderSource =  
`void main() {  
    gl_FragColor = vec4(1.0, 0.0, 0.0, 1.0);  
}`;  
  
var vsh = gl.createShader( gl.VERTEX_SHADER );  
gl.shaderSource( vsh, vertexShaderSource );  
gl.compileShader( vsh );  
  
var fsh = gl.createShader( gl.FRAGMENT_SHADER );  
gl.shaderSource( fsh, fragmentShaderSource );  
gl.compileShader( fsh );  
  
var prog = gl.createProgram();  
  
gl.attachShader( prog, vsh );  
gl.attachShader( prog, fsh );  
gl.linkProgram( prog );  
gl.useProgram(prog);  
  
var a_coords_location = gl.getAttribLocation(prog, "a_coords");  
  
var coords = new Float32Array( [0.0, 0.0, 0.0,  
                                0.0, 0.5, 0.0,  
                                0.5, 0.0, 0.0] );  
  
var a_coords_buffer = gl.createBuffer();  
  
gl.bindBuffer(gl.ARRAY_BUFFER, a_coords_buffer);  
gl.bufferData(gl.ARRAY_BUFFER, coords, gl.STATIC_DRAW);  
gl.vertexAttribPointer(a_coords_location, 3, gl.FLOAT, false, 0, 0);  
gl.enableVertexAttribArray(a_coords_location);  
gl.drawArrays(gl.TRIANGLES, 0, 3);
```

Step – 4: Define Geometry

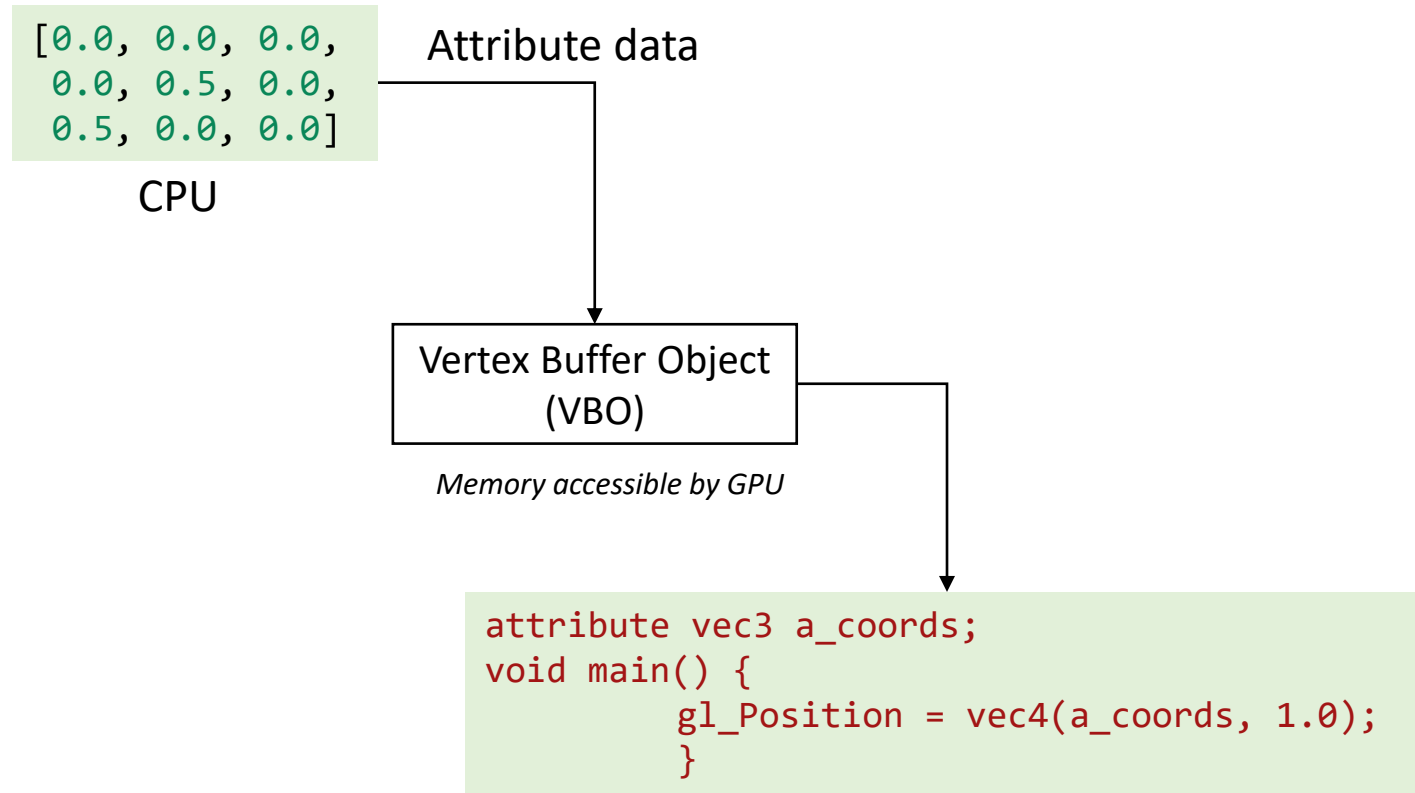
```
var coords = new Float32Array( [0.0, 0.0, 0.0, \\V0  
                                0.0, 0.5, 0.0, \\V1  
                                0.5, 0.0, 0.0] \\V2  
                                );
```



```
var canvas = document.getElementById("webglcanvas");  
var gl = canvas.getContext("webgl");  
  
gl.clearColor(0.75, 0.75, 0.75, 1.0);  
gl.clear(gl.COLOR_BUFFER_BIT);  
  
var vertexShaderSource =  
`attribute vec3 a_coords;  
void main() {  
    gl_Position = vec4(a_coords, 1.0);  
}`;  
  
var fragmentShaderSource =  
`void main() {  
    gl_FragColor = vec4(1.0, 0.0, 0.0, 1.0);  
}`;  
  
var vsh = gl.createShader( gl.VERTEX_SHADER );  
gl.shaderSource( vsh, vertexShaderSource );  
gl.compileShader( vsh );  
  
var fsh = gl.createShader( gl.FRAGMENT_SHADER );  
gl.shaderSource( fsh, fragmentShaderSource );  
gl.compileShader( fsh );  
  
var prog = gl.createProgram();  
  
gl.attachShader( prog, vsh );  
gl.attachShader( prog, fsh );  
gl.linkProgram( prog );  
gl.useProgram(prog);  
  
var a_coords_location = gl.getAttribLocation(prog, "a_coords");  
  
var coords = new Float32Array( [0.0, 0.0, 0.0,  
                                0.0, 0.5, 0.0,  
                                0.5, 0.0, 0.0] );  
  
var a_coords_buffer = gl.createBuffer();  
  
gl.bindBuffer(gl.ARRAY_BUFFER, a_coords_buffer);  
gl.bufferData(gl.ARRAY_BUFFER, coords, gl.STATIC_DRAW);  
gl.vertexAttribPointer(a_coords_location, 3, gl.FLOAT, false, 0, 0);  
gl.enableVertexAttribArray(a_coords_location);  
gl.drawArrays(gl.TRIANGLES, 0, 3);
```

Step – 4: Vertex Buffer Objects

```
var a_coords_buffer = gl.createBuffer();
```



```
var canvas = document.getElementById("webglcanvas");
var gl = canvas.getContext("webgl");

gl.clearColor(0.75, 0.75, 0.75, 1.0);
gl.clear(gl.COLOR_BUFFER_BIT);

var vertexShaderSource =
`attribute vec3 a_coords;
void main() {
    gl_Position = vec4(a_coords, 1.0);
}`;

var fragmentShaderSource =
`void main() {
    gl_FragColor = vec4(1.0, 0.0, 0.0, 1.0);
}`;

var vsh = gl.createShader( gl.VERTEX_SHADER );
gl.shaderSource( vsh, vertexShaderSource );
gl.compileShader( vsh );

var fsh = gl.createShader( gl.FRAGMENT_SHADER );
gl.shaderSource( fsh, fragmentShaderSource );
gl.compileShader( fsh );

var prog = gl.createProgram();

gl.attachShader( prog, vsh );
gl.attachShader( prog, fsh );
gl.linkProgram( prog );
gl.useProgram(prog);

var a_coords_location = gl.getAttribLocation(prog, "a_coords");

var coords = new Float32Array( [0.0, 0.0, 0.0,
                               0.0, 0.5, 0.0,
                               0.5, 0.0, 0.0] );

var a_coords_buffer = gl.createBuffer();

gl.bindBuffer(gl.ARRAY_BUFFER, a_coords_buffer);
gl.bufferData(gl.ARRAY_BUFFER, coords, gl.STATIC_DRAW);
gl.vertexAttribPointer(a_coords_location, 3, gl.FLOAT, false, 0, 0);
gl.enableVertexAttribArray(a_coords_location);
gl.drawArrays(gl.TRIANGLES, 0, 3);
```

Step – 4: Vertex Buffer Objects (VBOs)

```
gl.bindBuffer(gl.ARRAY_BUFFER, a_coords_buffer);
```

will be used for attribute

It specifies how the VBO will be used.

```
gl.bufferData(gl.ARRAY_BUFFER, coords, gl.STATIC_DRAW);
```

same data will be used

```
gl.vertexAttribPointer(a_coords_location, 3,  
gl.FLOAT, false, 0, 0);
```

For 3D data (x,y,z)

```
gl.enableVertexAttribArray(a_coords_location);
```

```
var canvas = document.getElementById("webglcanvas");  
var gl = canvas.getContext("webgl");  
  
gl.clearColor(0.75, 0.75, 0.75, 1.0);  
gl.clear(gl.COLOR_BUFFER_BIT);  
  
var vertexShaderSource =  
`attribute vec3 a_coords;  
void main() {  
    gl_Position = vec4(a_coords, 1.0);  
}`;  
  
var fragmentShaderSource =  
`void main() {  
    gl_FragColor = vec4(1.0, 0.0, 0.0, 1.0);  
}`;  
  
var vsh = gl.createShader( gl.VERTEX_SHADER );  
gl.shaderSource( vsh, vertexShaderSource );  
gl.compileShader( vsh );  
  
var fsh = gl.createShader( gl.FRAGMENT_SHADER );  
gl.shaderSource( fsh, fragmentShaderSource );  
gl.compileShader( fsh );  
  
var prog = gl.createProgram();  
  
gl.attachShader( prog, vsh );  
gl.attachShader( prog, fsh );  
gl.linkProgram( prog );  
gl.useProgram(prog);  
  
var a_coords_location = gl.getAttribLocation(prog, "a_coords");  
  
var coords = new Float32Array( [0.0, 0.0, 0.0,  
                                0.0, 0.5, 0.0,  
                                0.5, 0.0, 0.0] );  
  
var a_coords_buffer = gl.createBuffer();  
  
gl.bindBuffer(gl.ARRAY_BUFFER, a_coords_buffer);  
gl.bufferData(gl.ARRAY_BUFFER, coords, gl.STATIC_DRAW);  
gl.vertexAttribPointer(a_coords_location, 3, gl.FLOAT, false, 0, 0);  
gl.enableVertexAttribArray(a_coords_location);  
gl.drawArrays(gl.TRIANGLES, 0, 3);
```

Step – 5: Draw Required Objects

```
gl.drawArrays(gl.TRIANGLES, 0, 3);
```

primitives

Start vertex

Number of vertices

```
[0.0, 0.0, 0.0, → 0
```

```
0.0, 0.5, 0.0, → 1
```

```
0.5, 0.0, 0.0] → 3
```

```
var canvas = document.getElementById("webglcanvas");
var gl = canvas.getContext("webgl");

gl.clearColor(0.75, 0.75, 0.75, 1.0);
gl.clear(gl.COLOR_BUFFER_BIT);

var vertexShaderSource =
`attribute vec3 a_coords;
void main() {
    gl_Position = vec4(a_coords, 1.0);
}`;

var fragmentShaderSource =
`void main() {
    gl_FragColor = vec4(1.0, 0.0, 0.0, 1.0);
}`;

var vsh = gl.createShader( gl.VERTEX_SHADER );
gl.shaderSource( vsh, vertexShaderSource );
gl.compileShader( vsh );

var fsh = gl.createShader( gl.FRAGMENT_SHADER );
gl.shaderSource( fsh, fragmentShaderSource );
gl.compileShader( fsh );

var prog = gl.createProgram();

gl.attachShader( prog, vsh );
gl.attachShader( prog, fsh );
gl.linkProgram( prog );
gl.useProgram(prog);

var a_coords_location = gl.getAttribLocation(prog, "a_coords");

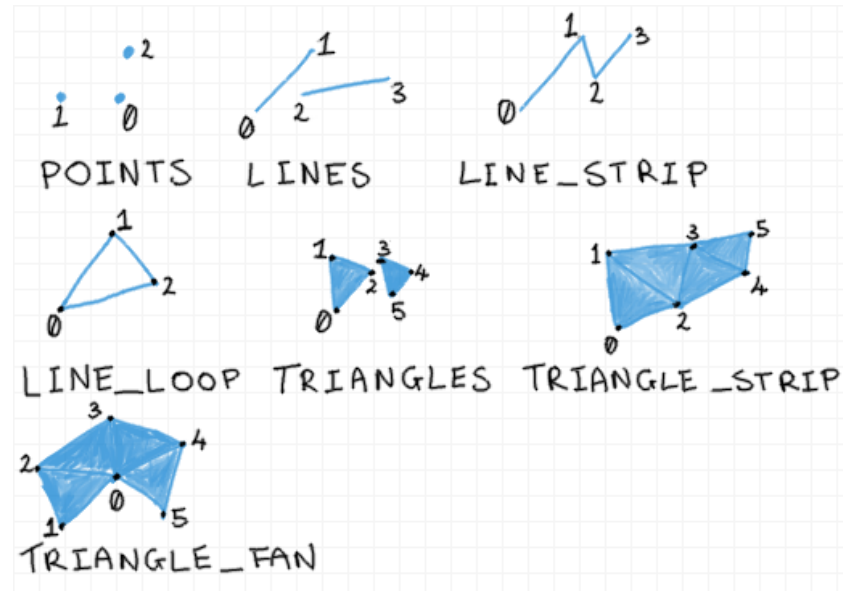
var coords = new Float32Array( [0.0, 0.0, 0.0,
                                0.0, 0.5, 0.0,
                                0.5, 0.0, 0.0] );

var a_coords_buffer = gl.createBuffer();

gl.bindBuffer(gl.ARRAY_BUFFER, a_coords_buffer);
gl.bufferData(gl.ARRAY_BUFFER, coords, gl.STATIC_DRAW);
gl.vertexAttribPointer(a_coords_location, 3, gl.FLOAT, false, 0, 0);
gl.enableVertexAttribArray(a_coords_location);
gl.drawArrays(gl.TRIANGLES, 0, 3);
```

Primitives

```
gl.drawArrays(gl.TRIANGLES, 0, 3);
```



Source: <https://antongerdelan.net/opengl/vertexbuffers.html>

```
var canvas = document.getElementById("webglcanvas");
var gl = canvas.getContext("webgl");

gl.clearColor(0.75, 0.75, 0.75, 1.0);
gl.clear(gl.COLOR_BUFFER_BIT);

var vertexShaderSource =
`attribute vec3 a_coords;
void main() {
    gl_Position = vec4(a_coords, 1.0);
}`;

var fragmentShaderSource =
`void main() {
    gl_FragColor = vec4(1.0, 0.0, 0.0, 1.0);
}`;

var vsh = gl.createShader( gl.VERTEX_SHADER );
gl.shaderSource( vsh, vertexShaderSource );
gl.compileShader( vsh );

var fsh = gl.createShader( gl.FRAGMENT_SHADER );
gl.shaderSource( fsh, fragmentShaderSource );
gl.compileShader( fsh );

var prog = gl.createProgram();

gl.attachShader( prog, vsh );
gl.attachShader( prog, fsh );
gl.linkProgram( prog );
gl.useProgram(prog);

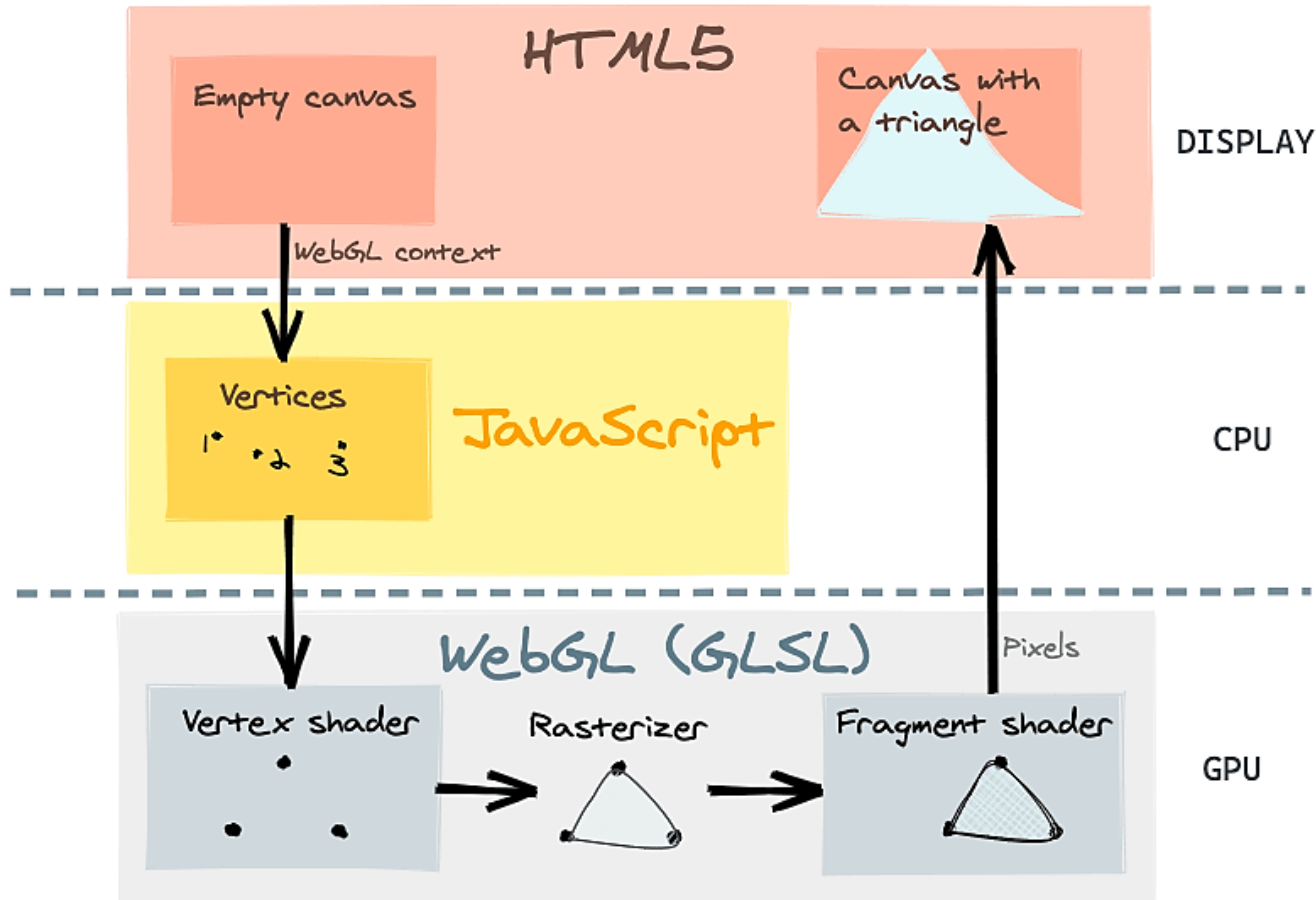
var a_coords_location = gl.getAttribLocation(prog, "a_coords");

var coords = new Float32Array( [0.0, 0.0, 0.0,
                                0.0, 0.5, 0.0,
                                0.5, 0.0, 0.0] );

var a_coords_buffer = gl.createBuffer();

gl.bindBuffer(gl.ARRAY_BUFFER, a_coords_buffer);
gl.bufferData(gl.ARRAY_BUFFER, coords, gl.STATIC_DRAW);
gl.vertexAttribPointer(a_coords_location, 3, gl.FLOAT, false, 0, 0);
gl.enableVertexAttribArray(a_coords_location);
gl.drawArrays(gl.TRIANGLES, 0, 3);
```


Full Code



```
var canvas = document.getElementById("webglcanvas");
var gl = canvas.getContext("webgl");

gl.clearColor(0.75, 0.75, 0.75, 1.0);
gl.clear(gl.COLOR_BUFFER_BIT);

var vertexShaderSource =
`attribute vec3 a_coords;
void main() {
    gl_Position = vec4(a_coords, 1.0);
}`;

var fragmentShaderSource =
`void main() {
    gl_FragColor = vec4(1.0, 0.0, 0.0, 1.0);
}`;

var vsh = gl.createShader( gl.VERTEX_SHADER );
gl.shaderSource( vsh, vertexShaderSource );
gl.compileShader( vsh );

var fsh = gl.createShader( gl.FRAGMENT_SHADER );
gl.shaderSource( fsh, fragmentShaderSource );
gl.compileShader( fsh );

var prog = gl.createProgram();

gl.attachShader( prog, vsh );
gl.attachShader( prog, fsh );
gl.linkProgram( prog );
gl.useProgram(prog);

var a_coords_location = gl.getAttribLocation(prog, "a_coords");

var coords = new Float32Array( [0.0, 0.0, 0.0,
                                0.0, 0.5, 0.0,
                                0.5, 0.0, 0.0] );

var a_coords_buffer = gl.createBuffer();

gl.bindBuffer(gl.ARRAY_BUFFER, a_coords_buffer);
gl.bufferData(gl.ARRAY_BUFFER, coords, gl.STATIC_DRAW);
gl.vertexAttribPointer(a_coords_location, 3, gl.FLOAT, false, 0, 0);
gl.enableVertexAttribArray(a_coords_location);
gl.drawArrays(gl.TRIANGLES, 0, 3);
```