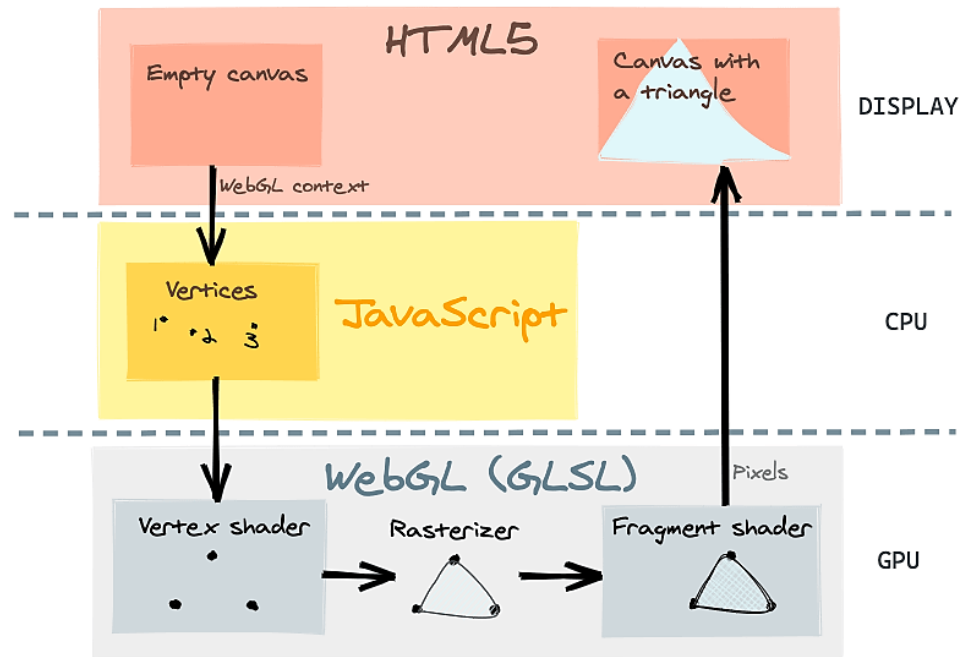


CSE4204

LAB-2 : GLSL – Attribute, Uniform, Varying and More

Mohammad Imrul Jubair

Recap



Source: <https://www.h5w3.com/44328.html>

Canvas and WebGL context

Create and Compile Shaders

Associate the shader variable

Define geometry + color and store it in buffer

Draw object

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

```
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.clearColor(0.75, 0.75, 0.75, 1.0);
gl.clear(gl.COLOR_BUFFER_BIT);
gl.drawArrays(gl.TRIANGLES, 0, 3);
```

Recap | Drawing a Triangle

Get the code:

rb.gy/zgu5ub

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

Canvas and WebGL context

Create and Compile Shaders

Associate the shader variable

Define geometry + color and store it in buffer

Draw object

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

```
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.clearColor(0.75, 0.75, 0.75, 1.0);
gl.clear(gl.COLOR_BUFFER_BIT);
gl.drawArrays(gl.TRIANGLES, 0, 3);
```

```

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);
}`;

```

Canvas and WebGL context

Create and Compile Shaders

Associate the shader variable

Define geometry + color and store it in buffer

Draw object

```

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

```

```

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.clearColor(0.75, 0.75, 0.75, 1.0);
gl.clear(gl.COLOR_BUFFER_BIT);
gl.drawArrays(gl.TRIANGLES, 0, 3);

```

```

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 );

```

Canvas and WebGL context

Create and Compile Shaders

Associate the shader variable

Define geometry + color and store it in buffer

Draw object

```

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

```

```

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.clearColor(0.75, 0.75, 0.75, 1.0);
gl.clear(gl.COLOR_BUFFER_BIT);
gl.drawArrays(gl.TRIANGLES, 0, 3);

```

```

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);

```

Canvas and WebGL context

Create and Compile Shaders

Associate the shader variable

Define geometry + color and store it in buffer

Draw object

```

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

```

```

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.clearColor(0.75, 0.75, 0.75, 1.0);
gl.clear(gl.COLOR_BUFFER_BIT);
gl.drawArrays(gl.TRIANGLES, 0, 3);

```

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

Canvas and WebGL
context

Create and Compile
Shaders

Associate the shader
variable

Define geometry +
color
and store it in buffer

Draw object

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

```
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.clearColor(0.75, 0.75, 0.75, 1.0);  
gl.clear(gl.COLOR_BUFFER_BIT);  
gl.drawArrays(gl.TRIANGLES, 0, 3);
```



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

Canvas and WebGL
context

Create and Compile
Shaders

Associate the shader
variable

Define geometry +
color
and store it in buffer

Draw object

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

```
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.clearColor(0.75, 0.75, 0.75, 1.0);
gl.clear(gl.COLOR_BUFFER_BIT);
gl.drawArrays(gl.TRIANGLES, 0, 3);
```

```
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();
```

Canvas and WebGL
context

Create and Compile
Shaders

Associate the shader
variable

Define geometry +
color
and store it in buffer

Draw object

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

```
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.clearColor(0.75, 0.75, 0.75, 1.0);
gl.clear(gl.COLOR_BUFFER_BIT);
gl.drawArrays(gl.TRIANGLES, 0, 3);
```

```

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);

```

Canvas and WebGL
context

Create and Compile
Shaders

Associate the shader
variable

Define geometry +
color
and store it in buffer

Draw object

```

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

```

```

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.clearColor(0.75, 0.75, 0.75, 1.0);
gl.clear(gl.COLOR_BUFFER_BIT);
gl.drawArrays(gl.TRIANGLES, 0, 3);

```

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

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

Canvas and WebGL
context

Create and Compile
Shaders

Associate the shader
variable

Define geometry +
color
and store it in buffer

Draw object

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

```
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.clearColor(0.75, 0.75, 0.75, 1.0);
gl.clear(gl.COLOR_BUFFER_BIT);
gl.drawArrays(gl.TRIANGLES, 0, 3);
```

Problem – 1

- We want to send color information from CPU → GPU
 - Not specified inside the shader

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



Uniform

Canvas and WebGL
context

Create and Compile
Shaders

Associate the shader
variable

Define geometry +
color
and store it in buffer

Draw object

Uniform

Canvas and WebGL
context

Create and Compile
Shaders



Associate the shader
variable

Define geometry +
color
and store it in buffer

Draw object

```
var fragmentShaderSource =  
  
    `precision mediump float;  
    uniform vec3 u_color;  
    void main() {  
        gl_FragColor = vec4(u_color, 1.0);  
    }`;
```

Uniform

Canvas and WebGL
context

Create and Compile
Shaders

Associate the shader
variable

Define geometry +
color
and store it in buffer

Draw object

```
var fragmentShaderSource =  
  
    `precision mediump float;  
    uniform vec3 u_color;  
    void main() {  
        gl_FragColor = vec4(u_color, 1.0);  
    }`;
```

```
var u_color_location = gl.getUniformLocation(prog, "u_color");
```


Uniform

Canvas and WebGL
context

Create and Compile
Shaders

Associate the shader
variable

Define geometry +
color
and store it in buffer

Draw object

```
var fragmentShaderSource =  
  
    `precision mediump float;  
    uniform vec3 u_color;  
    void main() {  
        gl_FragColor = vec4(u_color, 1.0);  
    }`;
```

```
var u_color_location = gl.getUniformLocation(prog, "u_color");
```

```
var color = new Float32Array( [0.5, 0.7, 0.3] );  
gl.uniform3fv(u_color_location, color);
```

Uniform

Canvas and WebGL
context

Create and Compile
Shaders

Associate the shader
variable

Define geometry +
color
and store it in buffer

Draw object

```
void gl.uniform1f(location, v0);  
void gl.uniform1fv(location, value);  
void gl.uniform1i(location, v0);  
void gl.uniform1iv(location, value);
```

```
var fragme
```

```
`precis
```

```
uniform
```

```
void m
```

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

```
`};
```

```
var u_color_location = gl.getUniformLocation(prog, "u_color");
```

```
var color = new Float32Array( [0.5, 0.7, 0.3] );  
gl.uniform3fv(u_color_location, color);
```

gl.uniform*: <https://developer.mozilla.org/en-US/docs/Web/API/WebGLRenderingContext/uniform>
<https://developer.mozilla.org/en-US/docs/Web/API/WebGLRenderingContext/uniformMatrix>

Problem – 1 | Uniform variable

Get the code:

rb.gy/nbuyoz

Uniform

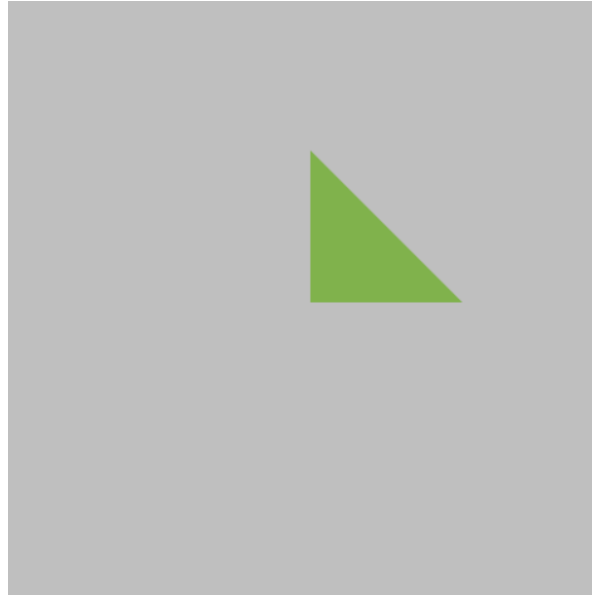
Canvas and WebGL
context

Create and Compile
Shaders

Associate the shader
variable

Define geometry +
color
and store it in buffer

Draw object



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

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

```
var fragmentShaderSource =  
`precision mediump float;  
uniform vec3 u_color;  
void main() {  
    gl_FragColor = vec4(u_color, 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);
```

```
var u_color_location = gl.getUniformLocation(prog, "u_color");  
var color = new Float32Array( [0.5, 0.7, 0.3] );  
gl.uniform3fv(u_color_location, color);
```

```
gl.clearColor(0.75, 0.75, 0.75, 1.0);  
gl.clear(gl.COLOR_BUFFER_BIT);  
gl.drawArrays(gl.TRIANGLES, 0, 3);
```

Problem – 2

- We want to shift the triangle via mouse clicking.

Clicking

Canvas and WebGL
context

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

Create and Compile
Shaders

```
var vertexShaderSource =  
`attribute vec3 a_coords;  
uniform float u_shift;  
void main() {  
    // gl_Position = vec4(a_coords, 1.0);  
    gl_Position = vec4(a_coords.x + u_shift, a_coords.y, a_coords.z, 1.0);  
};
```

Associate the shader
variable

```
var fragmentShaderSource =  
`precision mediump float;  
uniform vec3 u_color;  
void main() {  
    gl_FragColor = vec4(u_color, 1.0);  
};
```

Define geometry +
color
and store it in buffer

```
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 );
```

Draw object

```
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");
```

```
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);
```

```
var u_color_location = gl.getUniformLocation(prog, "u_color");  
var color = new Float32Array( [0.5, 0.7, 0.3] );  
gl.uniform3fv(u_color_location, color);
```

```
var u_shift_location = gl.getUniformLocation(prog, "u_shift");  
var shift = 0.0;  
gl.uniform1f(u_shift_location, shift);
```

```
gl.clearColor(0.75, 0.75, 0.75, 1.0);  
gl.clear(gl.COLOR_BUFFER_BIT);  
gl.drawArrays(gl.TRIANGLES, 0, 3);
```

```
canvas.onmousedown = function ()  
{  
    shift = shift + 0.1;  
    gl.uniform1f(u_shift_location, shift);  
    gl.clearColor(0.75, 0.75, 0.75, 1.0);  
    gl.clear(gl.COLOR_BUFFER_BIT);  
    gl.drawArrays(gl.TRIANGLES, 0, 3);  
};
```

Clicking

Canvas and WebGL
context

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

```
var vertexShaderSource =
```

```
`attribute vec3 a_coords;
```

```
uniform float u_shift;
```

```
void main() {
```

```
// gl_Position = vec4(a_coords, 1.0);
```

```
gl_Position = vec4(a_coords.x + u_shift, a_coords.y, a_coords.z, 1.0);
```

```
};
```

```
var fragmentShaderSource =  
`precision mediump float;  
uniform vec3 u_color;  
void main() {  
    gl_FragColor = vec4(u_color, 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 canvas = document.getElementById("webglcanvas");  
var gl = canvas.getContext("webgl");
```

```
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);
```

```
var u_color_location = gl.getUniformLocation(prog, "u_color");  
var color = new Float32Array( [0.5, 0.7, 0.3] );  
gl.uniform3fv(u_color_location, color);
```

```
var u_shift_location = gl.getUniformLocation(prog, "u_shift");
```

```
var shift = 0.0;
```

```
gl.uniform1f(u_shift_location, shift);
```

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

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

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

```
canvas.onmousedown = function ()  
{  
    shift = shift + 0.1;  
    gl.uniform1f(u_shift_location, shift);  
    gl.clearColor(0.75, 0.75, 0.75, 1.0);  
    gl.clear(gl.COLOR_BUFFER_BIT);  
    gl.drawArrays(gl.TRIANGLES, 0, 3);  
};
```

Define geometry +
color
and store it in buffer

Draw object

Clicking

Canvas and WebGL
context

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

Create and Compile
Shaders

```
var vertexShaderSource =  
`attribute vec3 a_coords;  
uniform float u_shift;  
void main() {  
    // gl_Position = vec4(a_coords, 1.0);  
    gl_Position = vec4(a_coords.x + u_shift, a_coords.y, a_coords.z, 1.0);  
};
```

Associate the shader
variable

```
var fragmentShaderSource =  
`precision mediump float;  
uniform vec3 u_color;  
void main() {  
    gl_FragColor = vec4(u_color, 1.0);  
};
```

Define geometry +
color
and store it in buffer

```
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 );
```

Draw object

```
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");
```

```
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);
```

```
var u_color_location = gl.getUniformLocation(prog, "u_color");  
var color = new Float32Array( [0.5, 0.7, 0.3] );  
gl.uniform3fv(u_color_location, color);
```

```
var u_shift_location = gl.getUniformLocation(prog, "u_shift");  
var shift = 0.0;  
gl.uniform1f(u_shift_location, shift);
```

```
gl.clearColor(0.75, 0.75, 0.75, 1.0);  
gl.clear(gl.COLOR_BUFFER_BIT);  
gl.drawArrays(gl.TRIANGLES, 0, 3);
```

```
canvas.onmousedown = function ()  
{  
    shift = shift + 0.1;  
    gl.uniform1f(u_shift_location, shift);  
    gl.clearColor(0.75, 0.75, 0.75, 1.0);  
    gl.clear(gl.COLOR_BUFFER_BIT);  
    gl.drawArrays(gl.TRIANGLES, 0, 3);  
};
```


Problem – 2 | Mouse Interaction

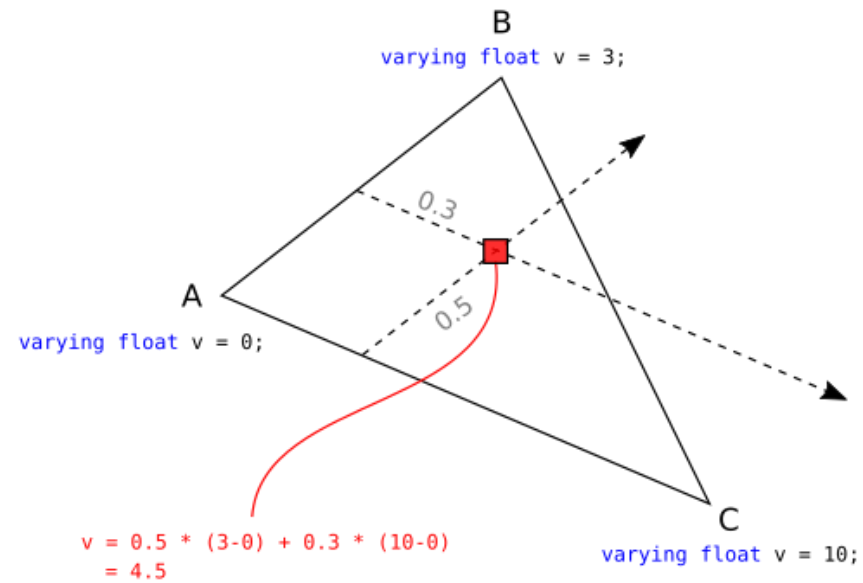
Get the code:

[rb.gy/ddbnlv](https://replit.com/@Jubair1234567890/rb-gy-ddb-nlv)

Problem – 3

- We want different color in different vertices and the color of the face will be blended accordingly.

Interpolation



Source: <https://stackoverflow.com/questions/17537879/in-webgl-what-are-the-differences-between-an-attribute-a-uniform-and-a-varying>

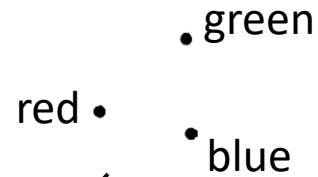
Varying

```
var vertexShaderSource =  
    `attribute vec3 a_coords;  
    attribute vec3 a_colors;  
    uniform float u_shift;  
    varying vec3 v_color;  
  
    void main() {  
        gl_Position = vec4(a_coords.x + u_shift, a_coords.y, a_coords.z, 1.0);  
        v_color = a_colors;  
    }`;
```

```
var fragmentShaderSource =  
    `precision mediump float;  
    varying vec3 v_color;  
    void main() {  
        gl_FragColor = vec4(v_color, 1.0);  
    }`;
```

Varying

• green
• blue
red •



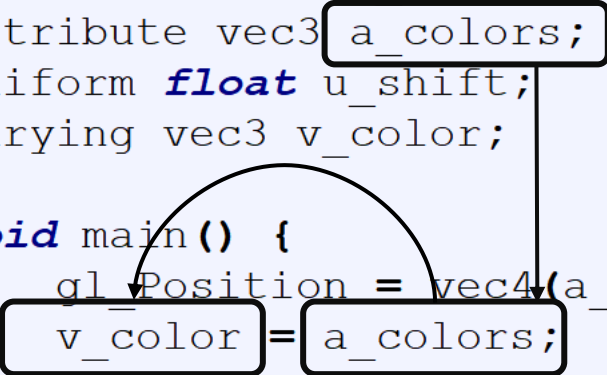
```
var vertexShaderSource =  
`attribute vec3 a_coords;  
attribute vec3 a_colors;  
uniform float u_shift;  
varying vec3 v_color;  
  
void main() {  
    gl_Position = vec4(a_coords.x + u_shift, a_coords.y, a_coords.z, 1.0);  
    v_color = a_colors;  
}`;
```

```
var fragmentShaderSource =  
`precision mediump float;  
varying vec3 v_color;  
void main() {  
    gl_FragColor = vec4(v_color, 1.0);  
}`;
```

Varying

• green
• blue
red •

```
var vertexShaderSource =  
`attribute vec3 a_coords;  
attribute vec3 a_colors;  
uniform float u_shift;  
varying vec3 v_color;  
  
void main() {  
    gl_Position = vec4(a_coords.x + u_shift, a_coords.y, a_coords.z, 1.0);  
    v_color = a_colors;  
}`;
```



```
var fragmentShaderSource =  
`precision mediump float;  
varying vec3 v_color;  
void main() {  
    gl_FragColor = vec4(v_color, 1.0);  
}`;
```

Varying

.green

red •

•blue

```
var vertexShaderSource =  
`attribute vec3 a_coords;  
attribute vec3 a_colors;  
uniform float u_shift;  
varying vec3 v_color;  
  
void main() {  
    gl_Position = vec4(a_coords.x + u_shift, a_coords.y, a_coords.z, 1.0);  
    v_color = a_colors;  
}`;
```

Interpolation

```
var fragmentShaderSource =  
`precision mediump float;  
varying vec3 v_color;  
void main() {  
    gl_FragColor = vec4(v_color, 1.0);  
}`;
```

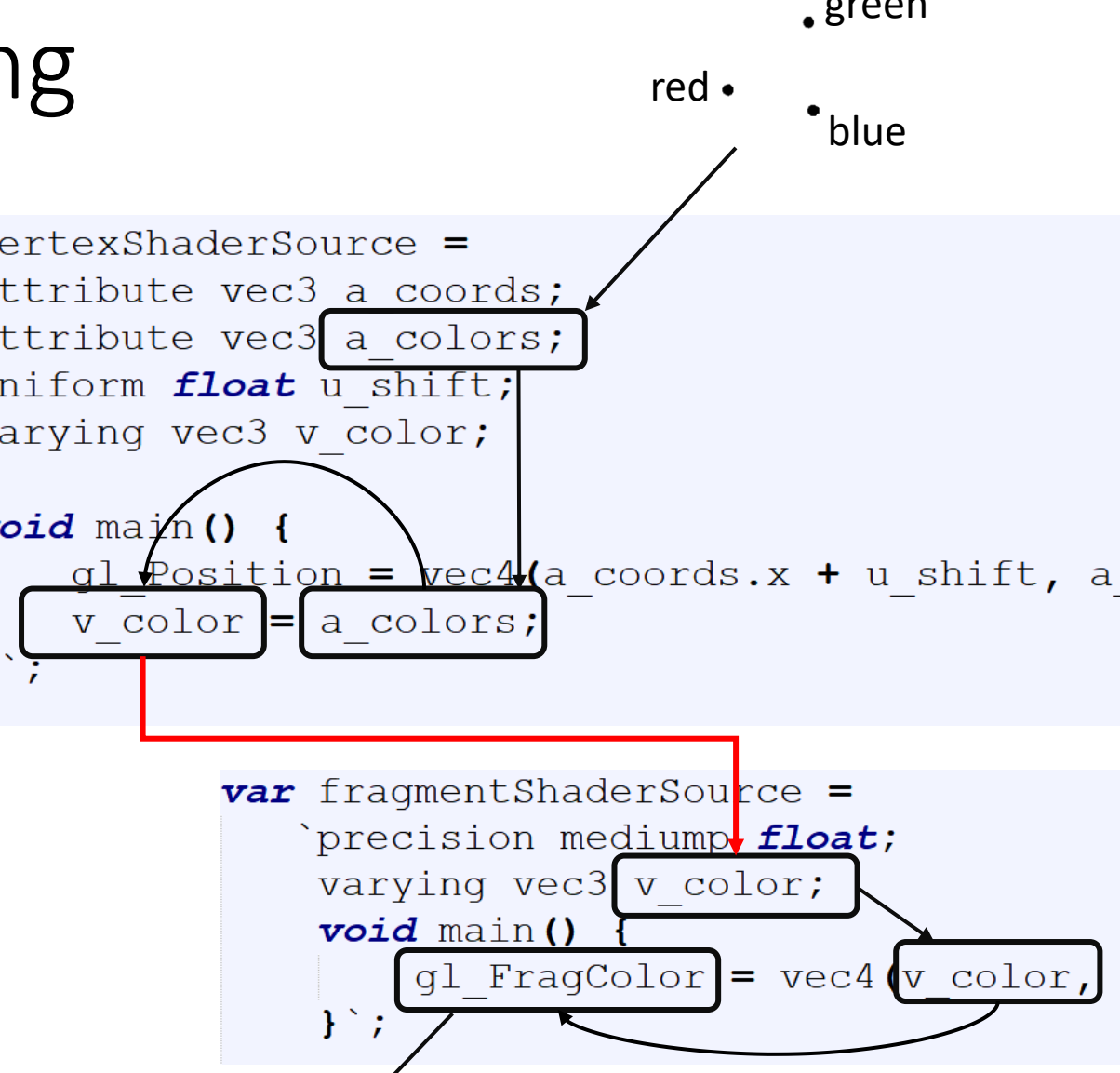
Varying

.green

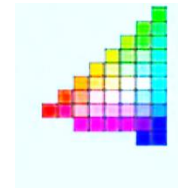
red •

•blue

```
var vertexShaderSource =  
`attribute vec3 a_coords;  
attribute vec3 a_colors;  
uniform float u_shift;  
varying vec3 v_color;  
  
void main() {  
    gl_Position = vec4(a_coords.x + u_shift, a_coords.y, a_coords.z, 1.0);  
    v_color = a_colors;  
}`;
```



```
var fragmentShaderSource =  
`precision mediump float;  
varying vec3 v_color;  
void main() {  
    gl_FragColor = vec4(v_color, 1.0);  
}`;
```



Varying

```
a_colors_location = gl.getAttribLocation(prog, "a_colors");  
var colors = new Float32Array( [1.0, 0.0, 0.0,  
                                0.0, 1.0, 0.0,  
                                0.0, 0.0, 1.0] );  
  
a_colors_buffer = gl.createBuffer();  
gl.bindBuffer(gl.ARRAY_BUFFER, a_colors_buffer);  
gl.bufferData(gl.ARRAY_BUFFER, colors, gl.STATIC_DRAW);  
gl.vertexAttribPointer(a_colors_location, 3, gl.FLOAT, false, 0, 0);  
gl.enableVertexAttribArray(a_colors_location);
```

Canvas and WebGL
context

Create and Compile
Shaders

Associate the shader
variable

Define geometry +
color
and store it in buffer

Draw object

Problem – 3 | Varying Variable

Get the code:

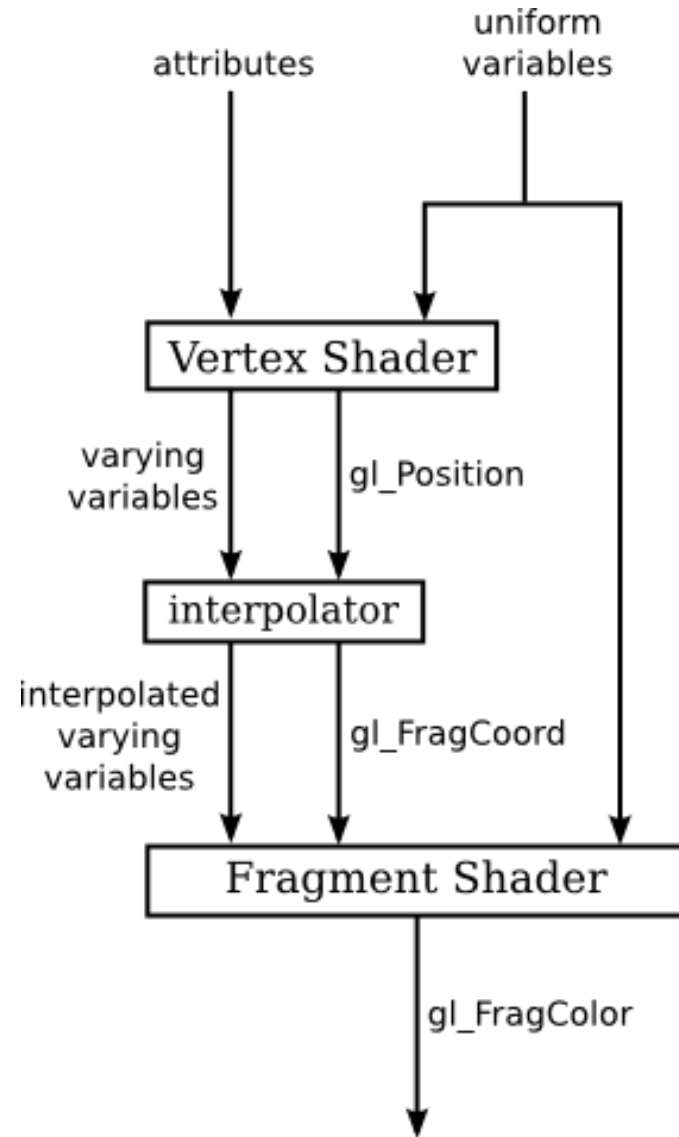
[rb.gy/p7u46l](https://replit.com/@p7u46l)

Uniform vs Attribute vs Varying

- uniform are per-primitive parameters
 - constant during an entire draw call
- attribute are per-vertex parameters
 - typically : positions, normals, colors, UVs, ...
- varying are per-fragment (or per-pixel) parameters
 - they vary from pixels to pixels

Source: <https://stackoverflow.com/questions/17537879/in-webgl-what-are-the-differences-between-an-attribute-a-uniform-and-a-varying>

Flow of data



Source: <http://math.hws.edu/graphicsbook/c6/s1.html>

Notes

- Attribute can only be used in vertex shader. *[why?]*
- Uniform can be used in both vertex and fragment shaders. *[why?]*
- Varying must be used in both vertex and fragment shaders with the same name.
- Uniform, attributes and varying must be declared globally in the shaders.
- It is a convention to use –
 - **a_** before the name of the attribute variable
 - **u_** before the name of the uniform variable
 - **v_** before the name of the varying variable

Control Statements in GLSL

Question: What will
happen here?

```
var vertexShaderSource =
`attribute vec3 a_coords;
attribute vec3 a_colors;
uniform float u_shift;
varying vec3 v_color;

void main() {
    if (u_shift < 0.7)
        gl_Position = vec4(a_coords.x - u_shift,
                           a_coords.y,
                           a_coords.z,
                           1.0);
    else
        gl_Position = vec4(a_coords.x,
                           a_coords.y,
                           a_coords.z,
                           1.0);
    v_color = a_colors;
}`;
```

More on GLSL statements: <https://www.shaderific.com/gsl-statements>

GLSL If Else

Get the code:

rb.gy/qdtslu

Built-in Functions in GLSL

```
void main() {  
    gl_Position = vec4(clamp(a_coords.x - u_shift, -0.5, 1.0),  
                          a_coords.y,  
                          a_coords.z,  
                          1.0);  
}
```

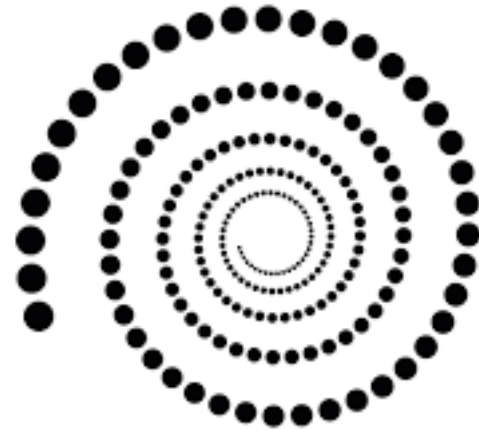
Question: What will happen here?

More GLSL built-in functions: <https://www.shaderific.com/glsl-functions>

Assignment - 1

Part A

- For each click, a 2D spiral will keep increasing.
 - The every dot will be a pixel.
 - The outer dots will be bigger than the inner ones depending on the distance from the center.



Assignment - 1

Part B

- Create a 2D scenario [use your imagination]
 - Minimum 20 triangles
- Apply color [using varying]
- Have a keyboard interaction
 - With control statements and/or built-in function

Assignment – 1

- Viva