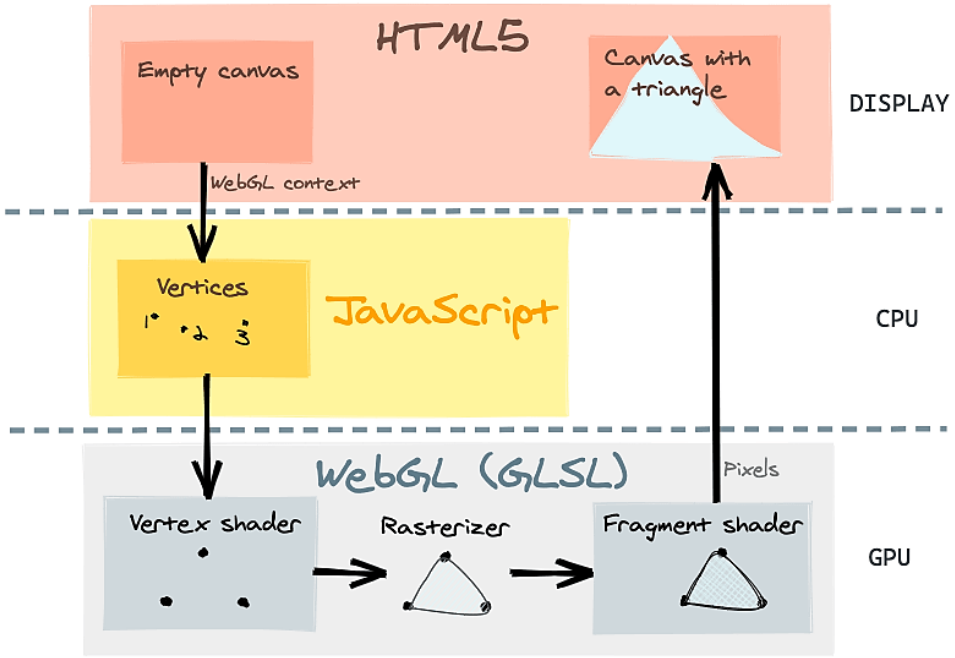
CSE4204

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

Get the materials

rb.gy/x8h51

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);
```

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

```
var fragmentShaderSource =

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

Associate the shader variable

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

Define geometry + color
and store it in buffer

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

Draw object

Uniform

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

Canvas and WebGL context

Create and Compile Shaders

```
var fragme
    `precis
    unifor
    void m
        gl_FragColor = vec4(u_color, 1.0);
    }`;
```

Associate the shader variable

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

Define geometry + color
and store it in buffer

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

Draw object

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

Uniform

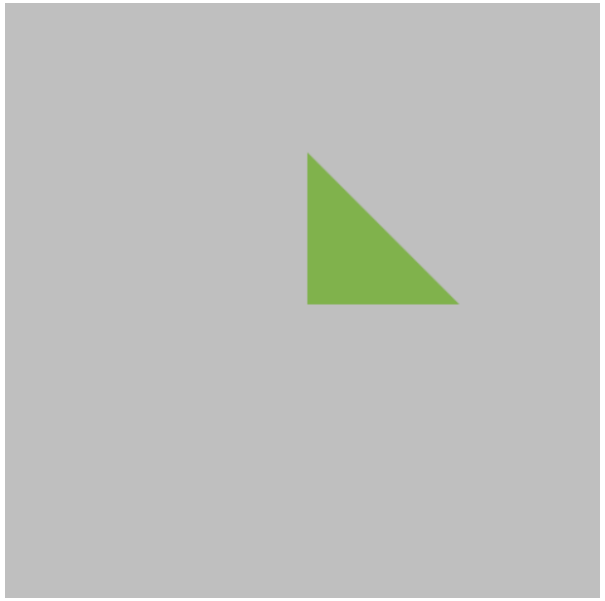
Canvas and WebGL context

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

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;
  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);
 };
```

Clicking

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;
  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);
 };
```

Clicking

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;
  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);
```

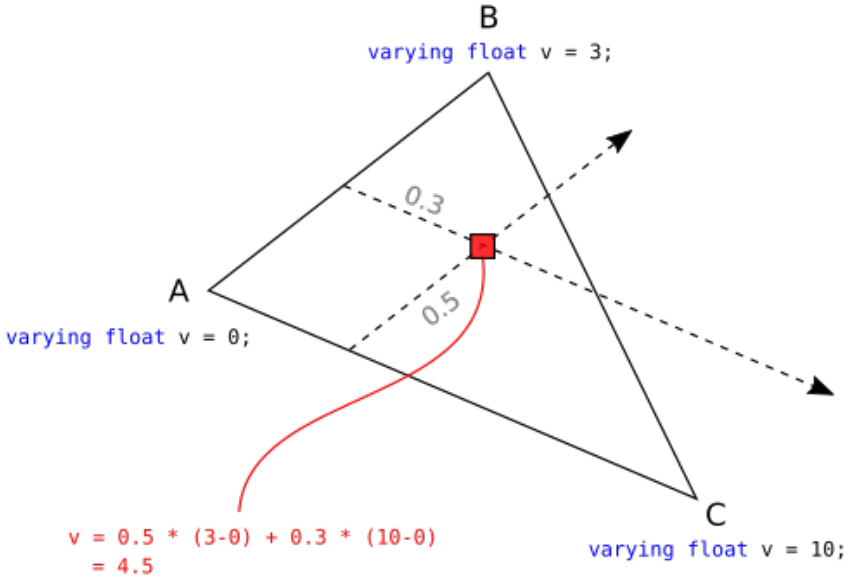
```
var canvas = document.getElementById("webglcanvas");
var  gl = canvas.getContext("webgl");

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);
};
```

Problem – 3

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

Interpolation



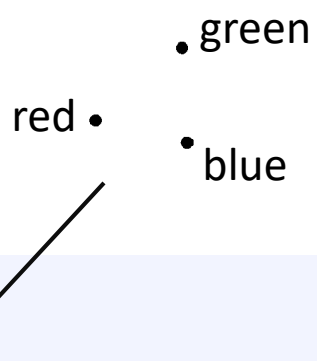
Source: https://stackoverflow.com/questions/17537879/in-webgl-what-are-the-differences-between-an-attribute-a-uniform-and-a-varying

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);
     }`;
```

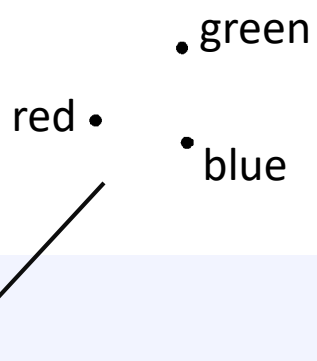
Varying




```
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);
     }`;
```

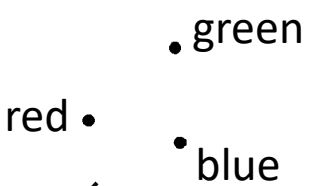
Varying




```
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




```
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

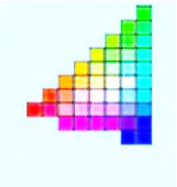
green

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

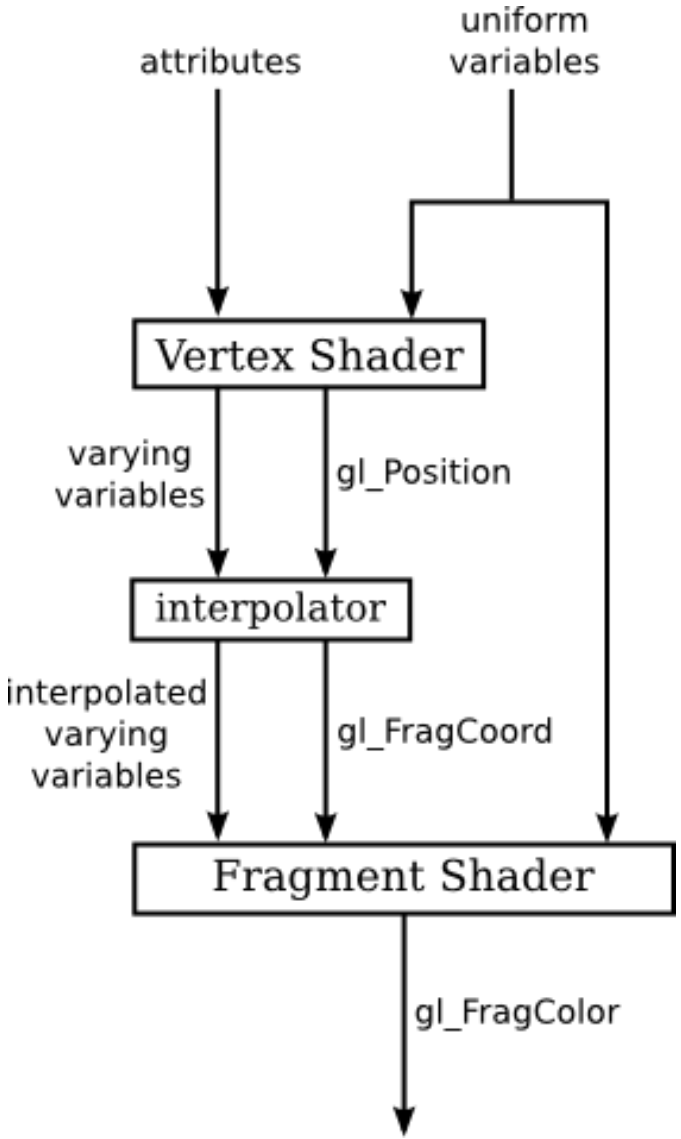
Draw object

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

```
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;
    }`;
```

Question: What will happen here?

More on GLSL statements: https://www.shaderific.com/glsl-statements

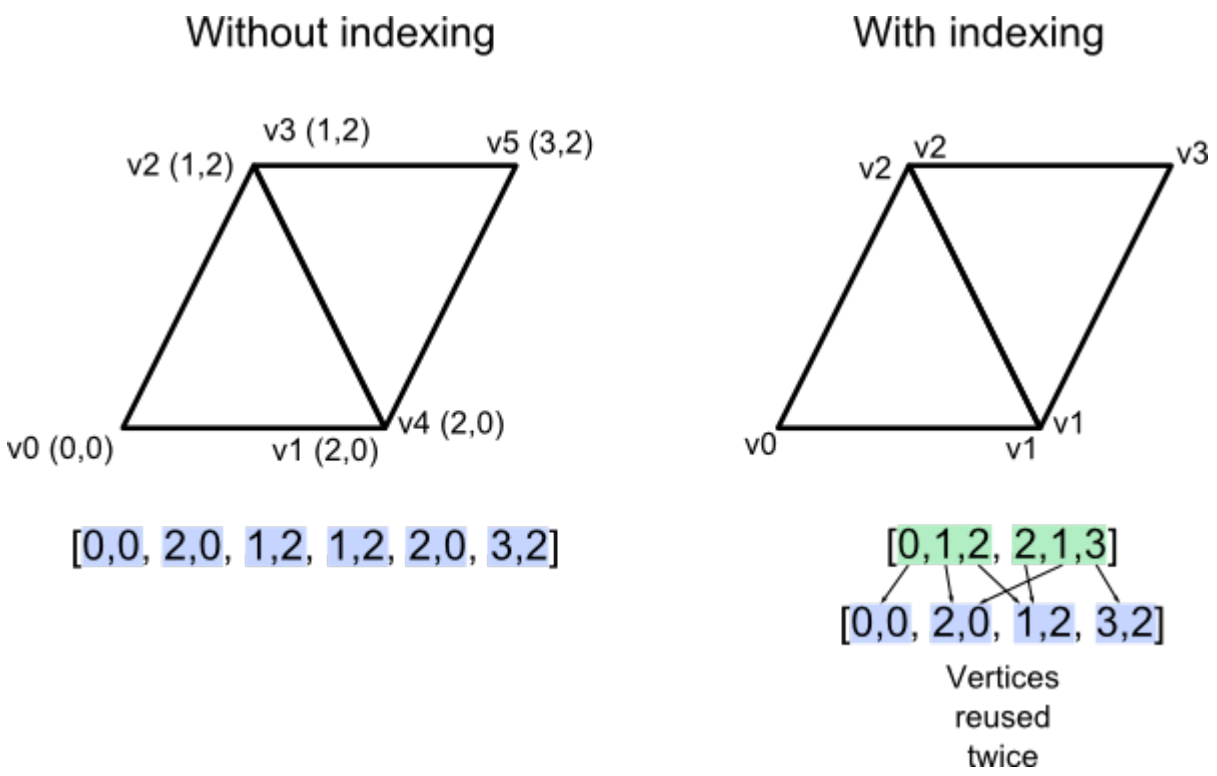
Built-in Functions in GLSL

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

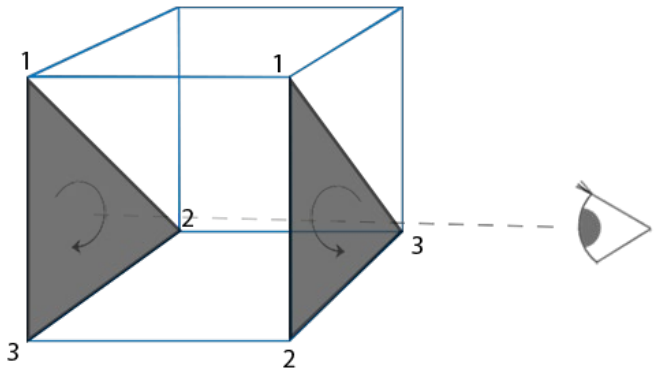
Question: What will happen here?

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

Index Buffer



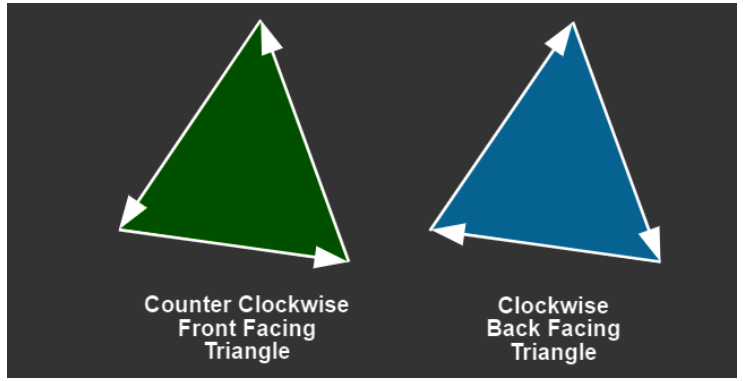
Index Buffer




```
var coords = new Float32Array( [
                -0.5, -0.5,  0.0, //v0
                 0.5, -0.5,  0.0, //v1
                 0.5,  0.5,  0.0, //v2
                -0.5,  0.5,  0.0  //v3
                                  ] );
```

```
var colors = new Float32Array( [
                1.0, 0.0, 0.0, //color at v0
                0.0, 1.0, 0.0, //color at v1
                0.0, 0.0, 1.0, //color at v2
                1.0, 1.0, 0.0 //color at v3
                              ] );
```

```
var indices = new Uint8Array([0, 1, 2,  0, 2, 3]);
```



https://webglfundamentals.org/webgl/lessons/webgl-3d-orthographic.html

Index Buffer


```
var coords = new Float32Array( [
                -0.5, -0.5,  0.0, //v0
                 0.5, -0.5,  0.0, //v1
                 0.5,  0.5,  0.0, //v2
                -0.5,  0.5,  0.0  //v3
                                ] );
```

```
var colors = new Float32Array( [
                1.0, 0.0, 0.0, //color at v0
                0.0, 1.0, 0.0, //color at v1
                0.0, 0.0, 1.0, //color at v2
                1.0, 1.0, 0.0 //color at v3
                              ] );
```

```
var indices = new Uint8Array([0, 1, 2,  0, 2, 3]);
```

```
var bufferInd = gl.createBuffer();
gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER, bufferInd);
gl.bufferData(gl.ELEMENT_ARRAY_BUFFER, indices, gl.STATIC_DRAW);
```

```
//gl.drawArrays(gl.TRIANGLES, 0, 3);
gl.drawElements(gl.TRIANGLES, 3*2, gl.UNSIGNED_BYTE, 0);
```