18 –glsl

Simple Fragment and Vertex Shader Programs

https://webglfundamentals.org/webgl/lessons/resources/webgl-state-diagram.html?exampleId=smallest-glsl

```
const vsGLSL = `
void main() {
    gl_Position = vec4(0, 0, 0, 1);
    gl_PointSize = 100.0;
}
`;

const fsGLSL = `
precision highp float;

void main() {
    gl_FragColor = vec4(1, 0.5, 0, 1);
}
```

Setting up a Shader

```
const vertexShader = gl.createShader(gl.VERTEX_SHADER);
gl.shaderSource(vertexShader, vsGLSL);
gl.compileShader(vertexShader);

const fragmentShader = gl.createShader(gl.FRAGMENT_SHADER);
gl.shaderSource(fragmentShader, fsGLSL);
gl.compileShader(fragmentShader);

const prg = gl.createProgram();
gl.attachShader(prg, vertexShader);
gl.attachShader(prg, fragmentShader);
gl.linkProgram(prg);

gl.useProgram(prg);
```

Vertex Shader For a Cube with Two Textures

```
attribute vec4 position;
attribute vec3 normal;
attribute vec2 texcoord;
uniform mat4 projection;
uniform mat4 modelView;
varying vec3 v_normal;
varying vec2 v_texcoord;
void main() {
    gl_Position = projection * modelView * position;
    v_normal = mat3(modelView) * normal;
    v_texcoord = texcoord;
}
```

3x3

Fragment Shader For a Cube with Two Textures

```
precision highp float;
varying vec3 v_normal;
varying vec2 v_texcoord;
uniform sampler2D diffuse;
uniform sampler2D decal;
uniform vec4 diffuseMult;
uniform vec3 lightDir;
void main() {
   vec3 normal = normalize(v_normal);
   float light = dot(normal, lightDir) * 0.5 + 0.5;
   vec4 color = texture2D(diffuse, v_texcoord) * diffuseMult;
   vec4 decalColor = texture2D(decal, v_texcoord);
   decalColor.rgb *= decalColor.a;
   color = color * (1.0 - decalColor.a) + decalColor;
   gl_FragColor = vec4(color.rgb * light, color.a);
}
```

arrays in JS are objects

```
a = [1, 2]
```

$\updownarrow$

```
a = { b : "b",
      c : 2 }

a["b"]
a["c"]
```

```
{ "0" : 1,
  "1" : 2
}
```

requesting memory buffers

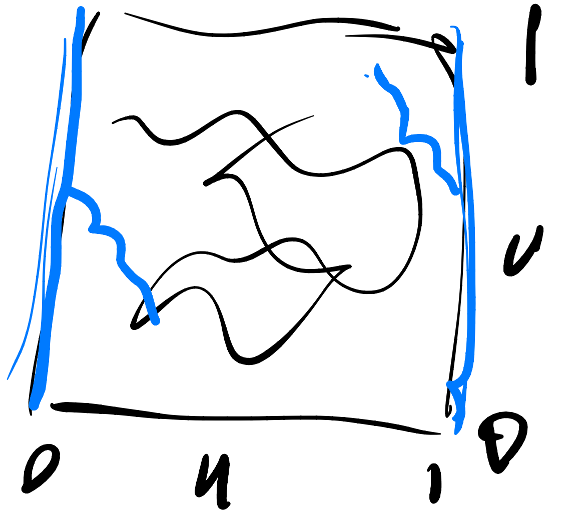
a = Float32Array ( ) $\Rightarrow$ a thing that is an array

length
data

a[1]
a.length

Typed Array Buffers

Textures have a bunch of other properties besides the data



texture2D(tex, uv)

can be any 2 floats

-0.5 , 3.2

~~[illegible]~~ transfor

Shaders in three.js

- All Materials create custom Shaders
  - three/src/renderers/WebGLRenderer.js (setProgram() line 1648)
  - three/src/renderers/webgl/WebGLProgram.js (WebGLProgram() line 378)
    - prefixVertex and prefixFragment prepended to your shader programs

- All the bits are in ShaderLib/ShaderChunk directories
  - E.g., PhongMaterial
    - /shaders/ShaderLib/meshphong_vert.glsl, /shaders/ShaderLib/meshphong_frag.glsl
  - Parameterized based on properties on material!
    - begin_vertex.glsl.js
    - project_vertex.glsl.js
    - worldpos_vertex.glsl.js

Custom Shaders: ShaderMaterial and RawShaderMaterial

- Include various parameters based on what material props you set
  - three/src/renderers/webgl/WebGLProgram.js (WebGLProgram() line 378)
    - prefixVertex and prefixFragment prepended to your shader programs

Raw gets nothing.
ShaderMaterial get a bunch of useful attribs & uniforms

Minimal Standard Parameters get Set by three

Vertex

```
uniform mat4 modelMatrix;
uniform mat4 modelViewMatrix;
uniform mat4 projectionMatrix;
uniform mat4 viewMatrix;
uniform mat3 normalMatrix;
uniform vec3 cameraPosition;
uniform bool isOrthographic;
attribute vec3 position;
attribute vec3 normal;
attribute vec2 uv;
```

Fragment

```
uniform mat4 viewMatrix;
uniform vec3 cameraPosition;
uniform bool isOrthographic;
```

Other uniforms/attributes set based on feature

- Lights
- Texture mapping (.map property)
  - Fragment: uniform sampler2D map

ex3, taken from

https://threejs.org/examples/#webgl_custom_attributes