A Gentle Introduction

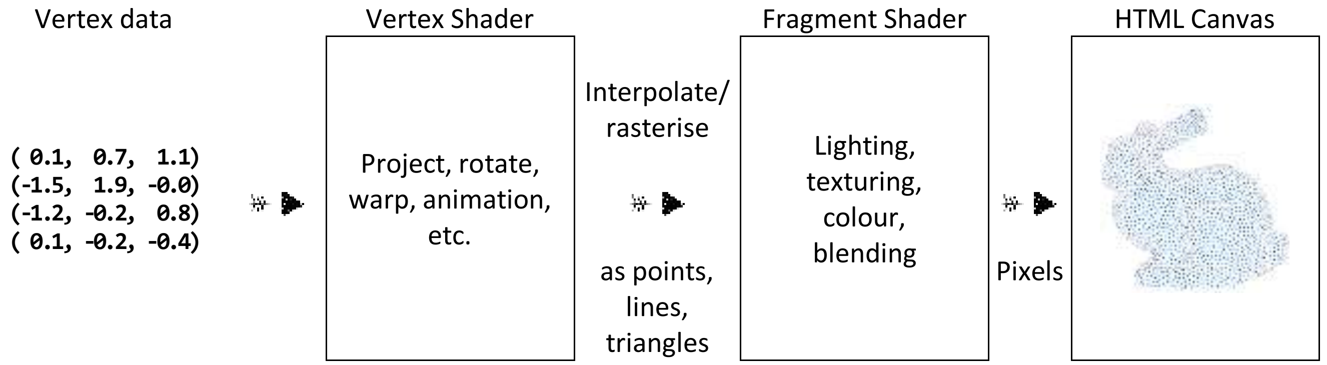
1 - Vertex Shaders

Carl Bateman
WebGL Workshop

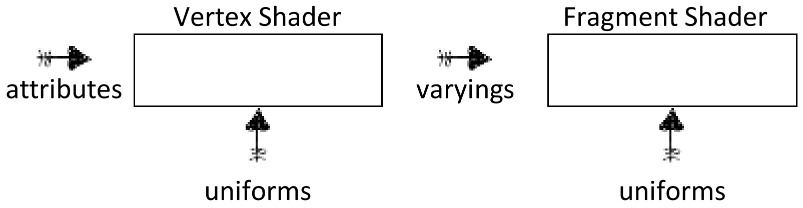
Table of Contents

Part 1. Graphics Pipeline

Simplified!



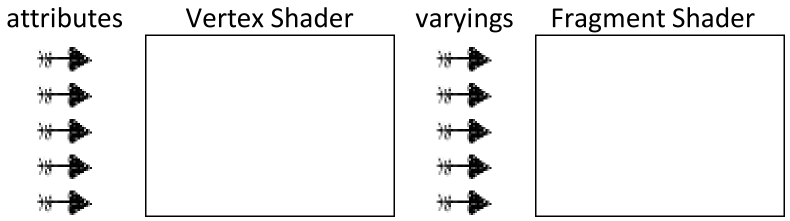
Sharing Data



SIMD

Single **I**nstruction **M**ultiple **D**ata

Shader programs run on the GPU, with the same set of instuctions (the vertex shader) working on each data item at the same time.



Programs can't access adjacent pixels or vertices (there are workrounds).

Part 2. GLSL

Khronos reference sheets

https://www.khronos.org/developers/reference-cards/

Support

https://caniuse.com/#search=webgl

WebGL (Web Graphics Library) is a JavaScript API for rendering interactive 3D and 2D graphics within any compatible web browser without the use of plug-ins. WebGL does so by introducing an API that closely conforms to OpenGL ES 2.0 that can be used in HTML5 `<canvas>` elements.

https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API

WebGL programs consist of control code written in JavaScript and shader code that is written in OpenGL ES Shading Language (GLSL ES), a language similar to C or C++, and is executed on a computer's graphics processing unit (GPU).

https://en.wikipedia.org/wiki/WebGL

WebGL1

Vertex shader

```
attribute vec3 aVertex;
attribute vec3 aColor;
varying vec3 vColor;

void main() {
  vColor = aColor;
  gl_Position = vec4(aVertex, 1.0);
}
```

Fragment shader

```
precision highp float;

varying vec3 vColor;

void main(void) {
  gl_FragColor = vec4(vColor, 1.0);
}
```

WebGL2

Vertex shader

```
#version 300 es

layout (location=0) in vec4 vertex;
layout (location=1) in vec3 color;

out vec3 vColor;

void main() {
  vColor = color;
  gl_Position = vertex;
}
```

Fragment shader

```
#version 300 es
precision highp float;

in vec3 vColor;
out vec4 fragColor;

void main() {
  fragColor = vec4(vColor, 1.0);
}
```

Part 3. Tasks

http://www.pleek.net/vertexlove/

Reduce number of vertices (not a code thing)

Centre cube

Change colours

Place them based on index

Move

Distort

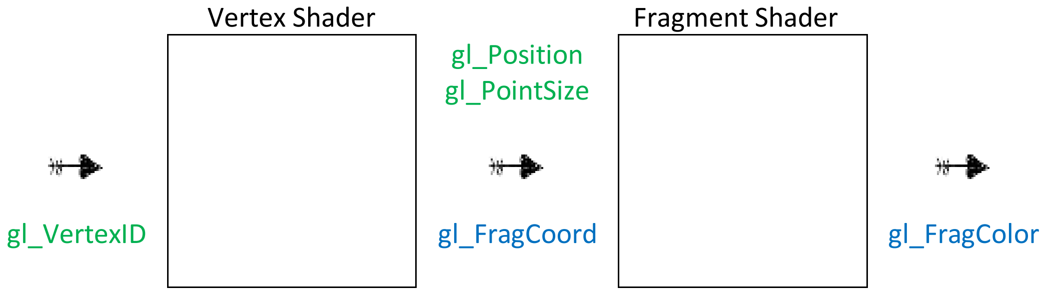
Make a sphere

Animate

Take a look at:

https://www.vertexshaderart.com/

Built-In Inputs, Outputs, and Constants



Types

GLSL is a strongly typed c-like language. Aimed at graphics it natively supports geometric functions and types.

In addition to the usual suspects: int, float, bool

vec2, vec3, vec4 (access with [0], .xyzw, .rgba)

mat2, mat3, mat4

sampler2D, sampler3D, samplerCube

Some useful(?) functions

Some you might like to try:

sin, cos, tan, pow, exp, sqrt, sign, floor, ceil, fract, mod, min, max, clamp, length, distance

Part 4. Practical Examples

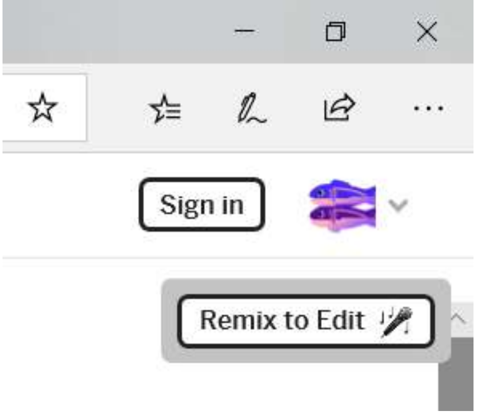
https://glitch.com/@CarlBateman/shader-intro-1-vertex

There are several examples, showing how to incorporate shaders into a web page.

Remix!

Select a project from the collection

Click on the Remix to Edit button in the top right.



This will create your own version of the project to work on.

Part 5. Resources

Vertex editor only

http://www.pleek.net/vertexlove/
https://www.vertexshaderart.com/

Vertex and fragment editor

https://shaderfrog.com/app/editor
http://shdr.bkcore.com/
http://www.kickjs.org/example/shader_editor/shader_editor.html
https://cyos.babylonjs.com/
http://bkcore.com/blog/3d/shdr-online-glsl-shader-editor-viewer-validator.html

Node editor

https://www.gsn-lib.org/index.html#projectName=public3dshader&graphName=NormalTrans
https://victhorlopez.github.io/editor/

Background info

http://www.shaderific.com/glsl
https://www.awwwards.com/inspiration/5981b0a1e13823534b28b8cb
https://medium.com/@Zadvorsky/into-vertex-shaders-594e6d8cd804

Cool

https://glitch.com/~shader-doodle-test