DATA VIS WITH SHADERS

1. Get your laptop
2. Clone this repo:

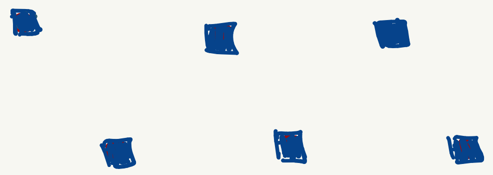
```
git@github.com:rolyatmax/webgl-learnin
```

3. npm install

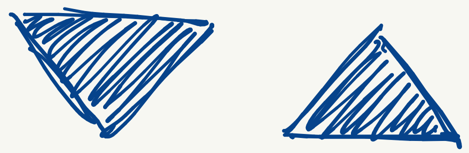
MY SHORTEST, SIMPLEST, LEAST-WRONG DESCRIPTION OF THE GRAPHICS PIPELINE.

WHAT CAN I DRAW?

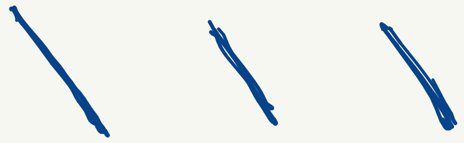
POINTS



TRIANGLES



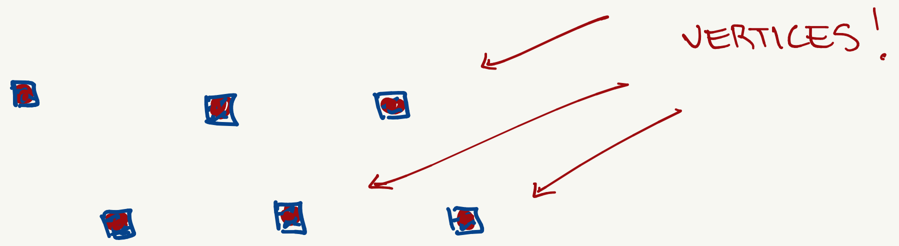
LINES



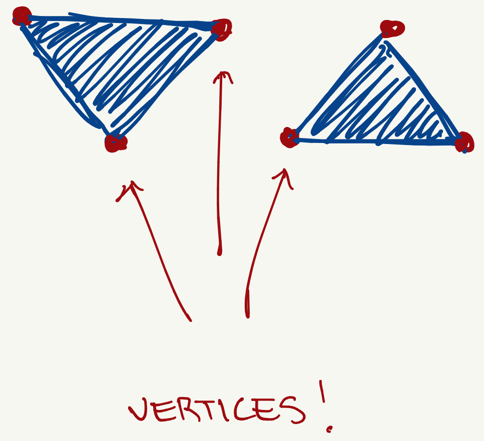
WHAT CAN I DRAW?

DEFINED BY THEIR VERTICES

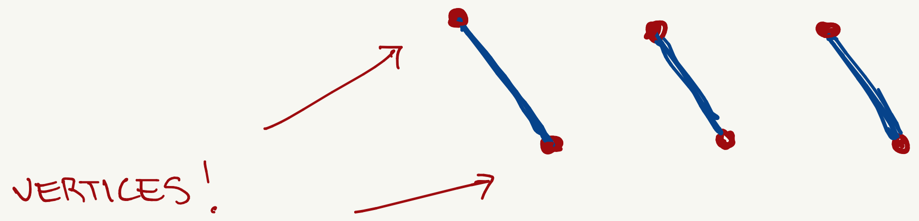
POINTS



TRIANGLES



LINES



DRAW THIS SHAPE

USING N VERTICES

WITH THIS STATE

AND THIS LOGIC.

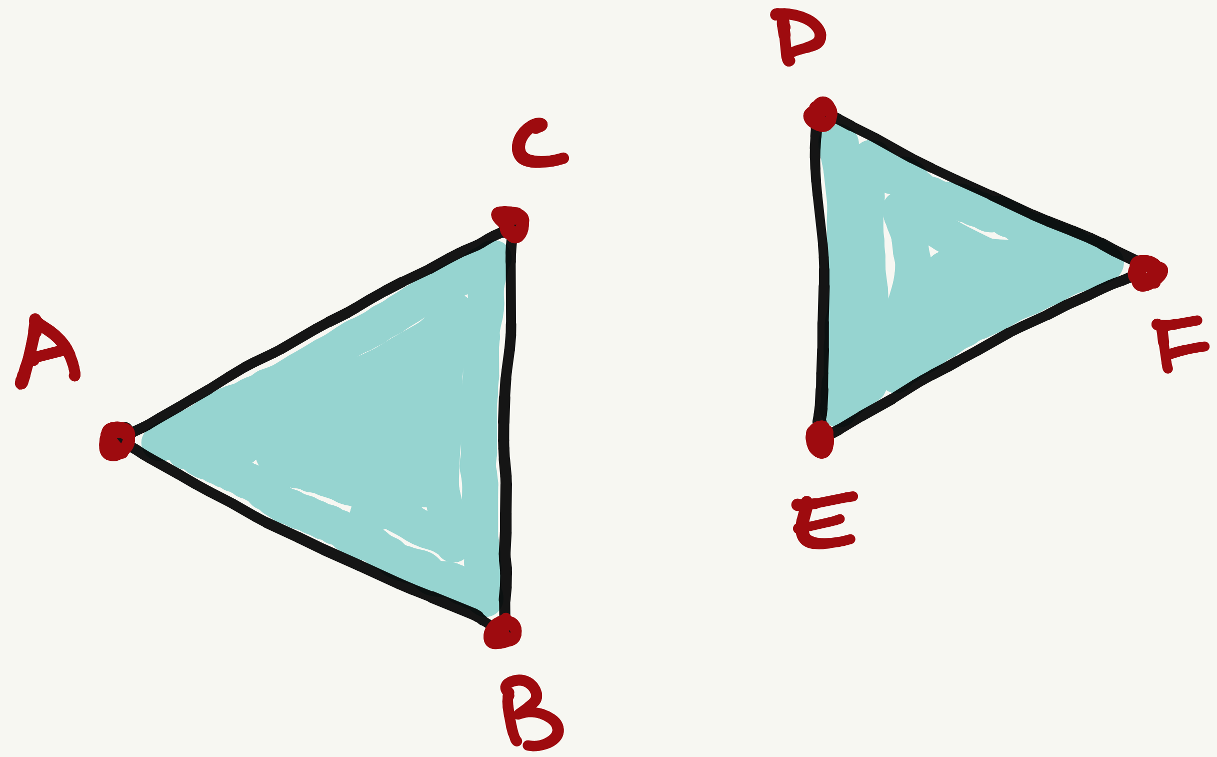
DRAW THIS SHAPE

USING N VERTICES

WITH ATTRIBUTES + UNIFORMS

AND VERTEX + FRAGMENT SHADERS.

AN EXAMPLE:



DRAW TRIANGLES

USING 6 VERTICES

(ATTRIBUTES)

WITH

A	B	C	D	E	F

AND VERTEX + FRAGMENT SHADERS.

THE VERTEX SHADER

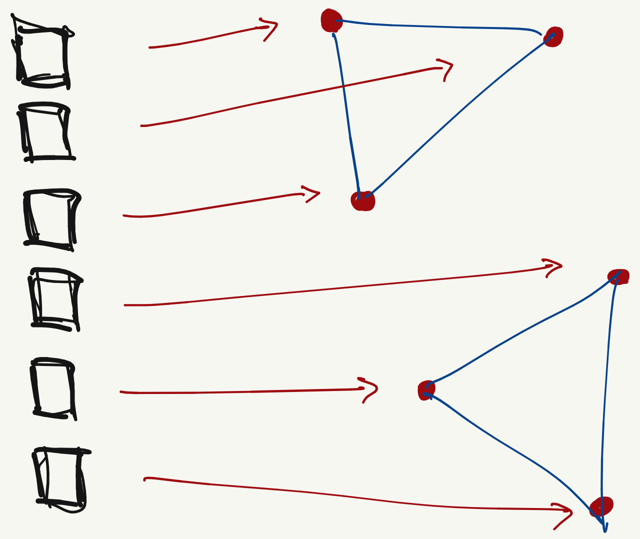
INPUT:

ATTRIBUTES
UNIFORMS

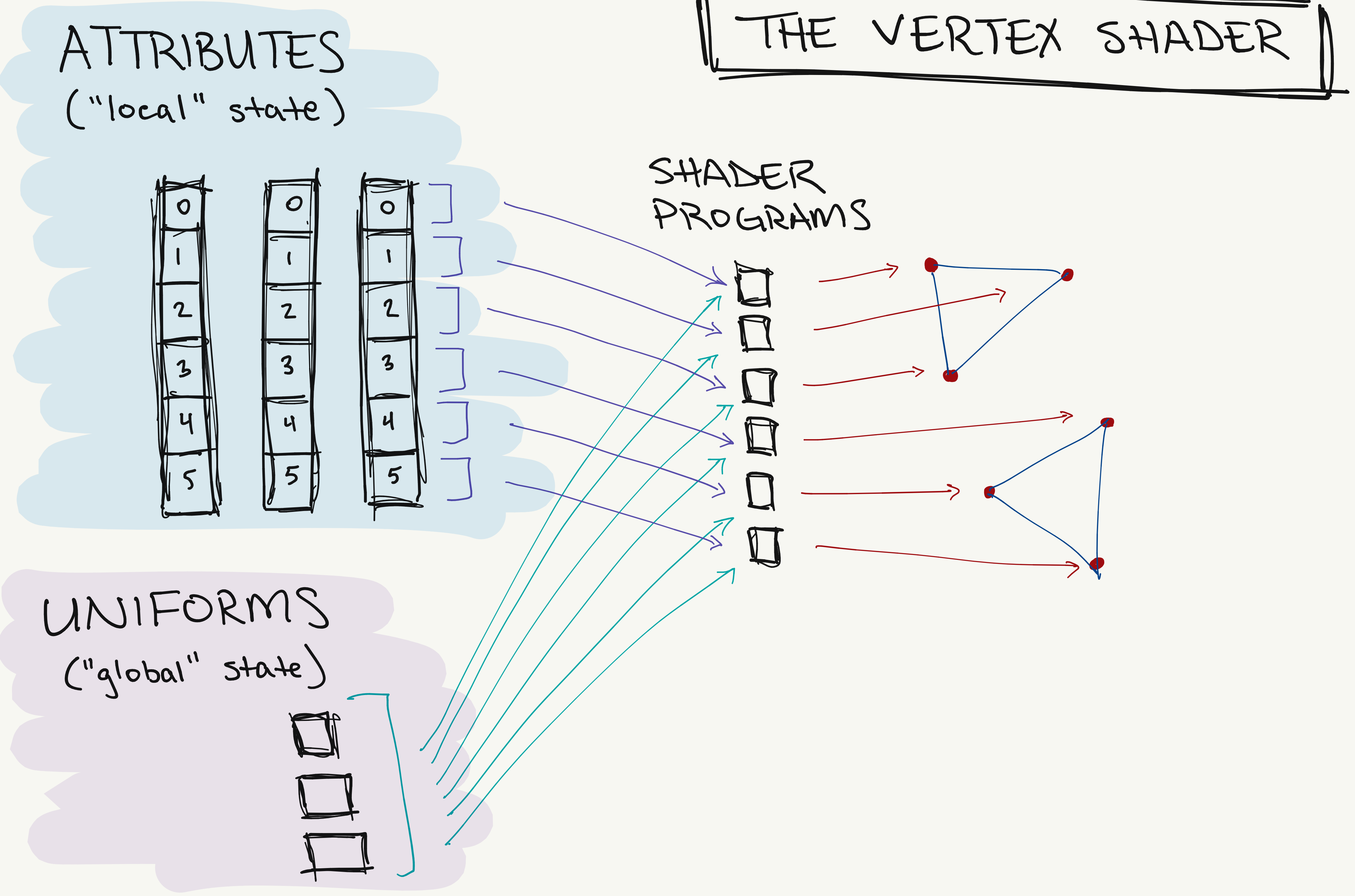
OUTPUT:

VERTEX POSITIONS

SHADER
PROGRAMS

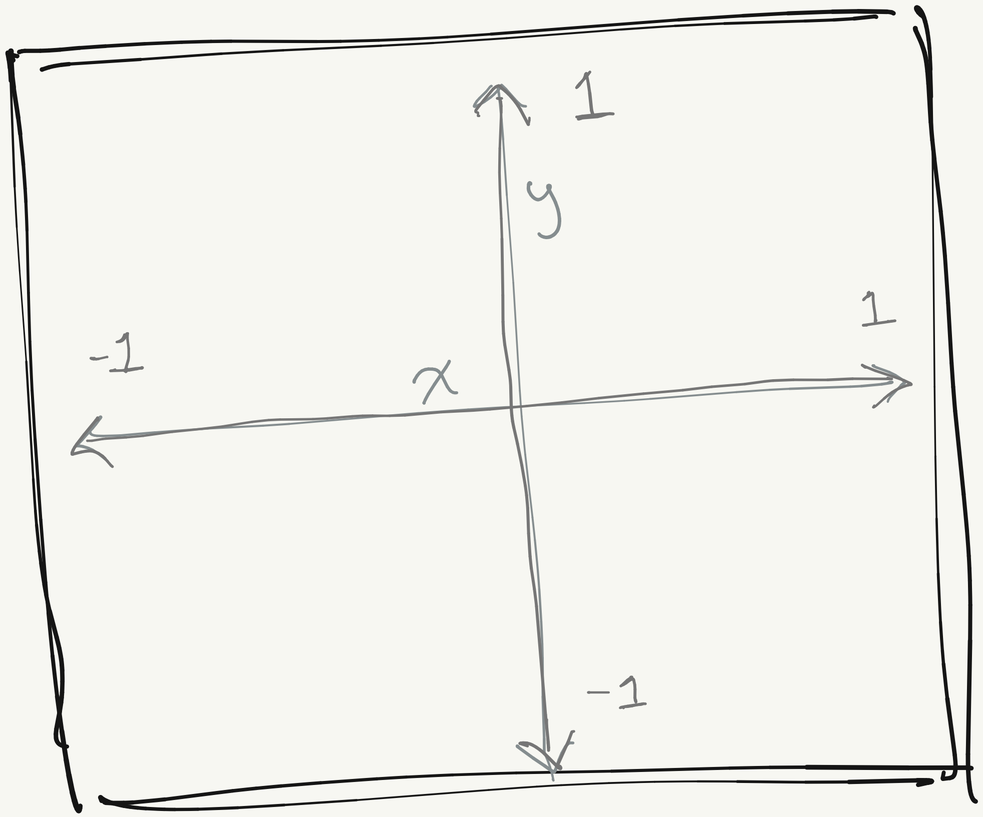


THE VERTEX SHADER



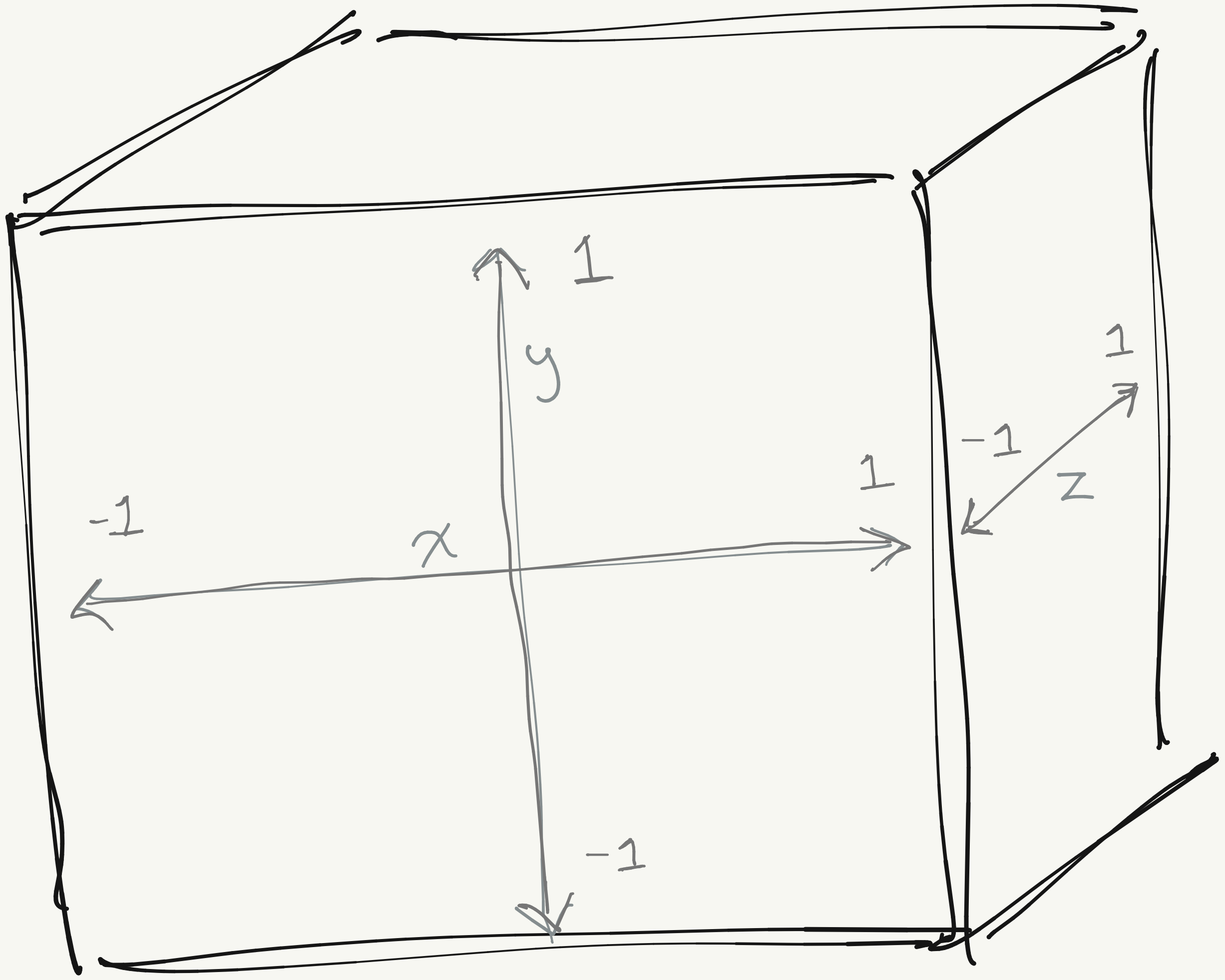
CLIPSPACE

THE COORDINATE SYSTEM USED BY WEBGL

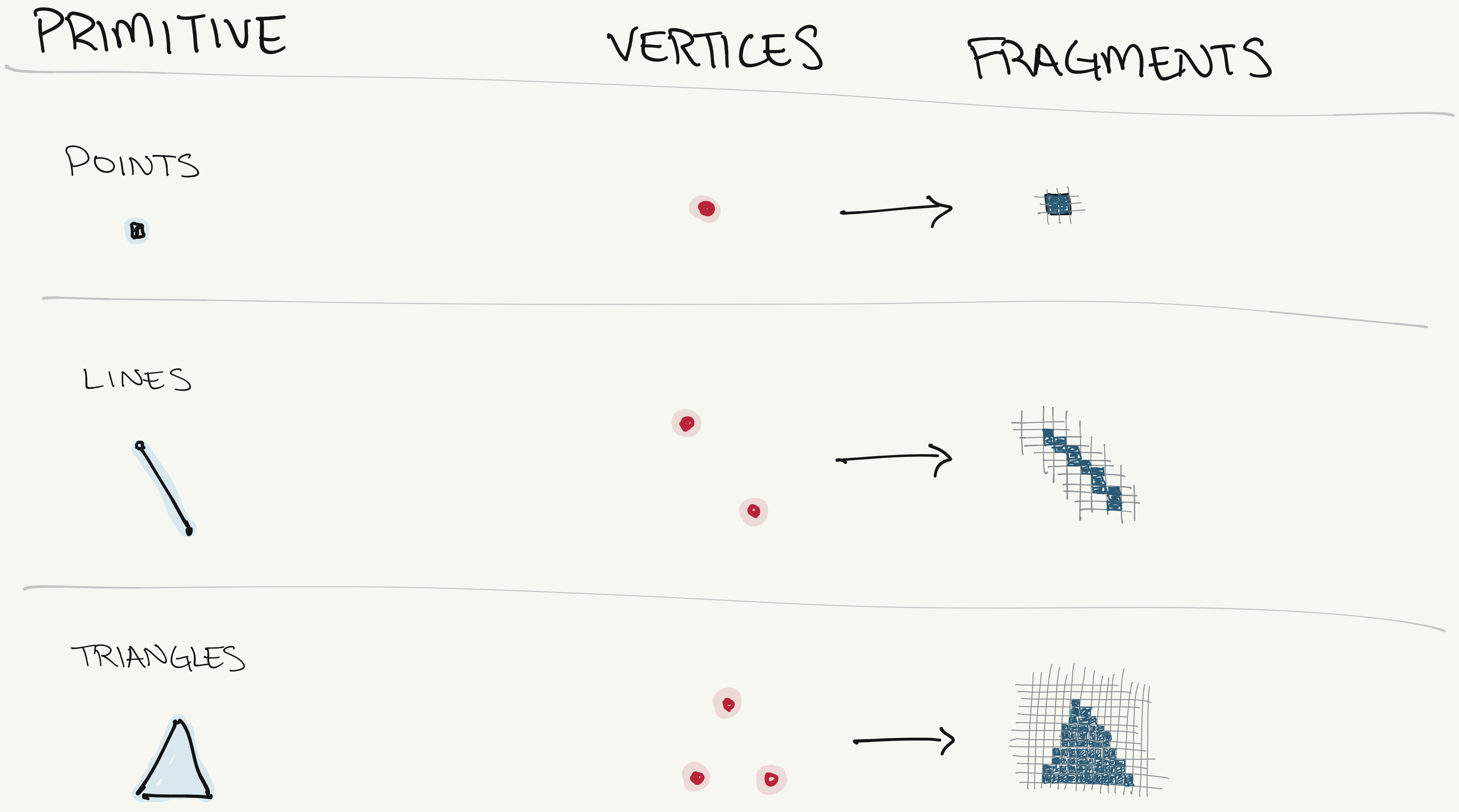


CLIPSPACE

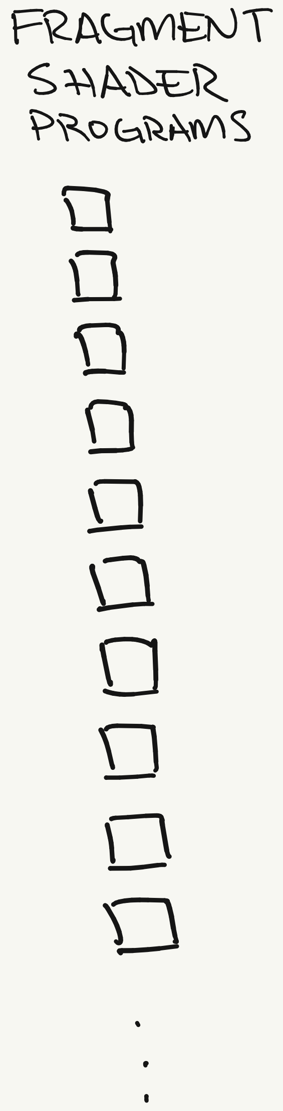
THE COORDINATE SYSTEM USED BY WEBGL



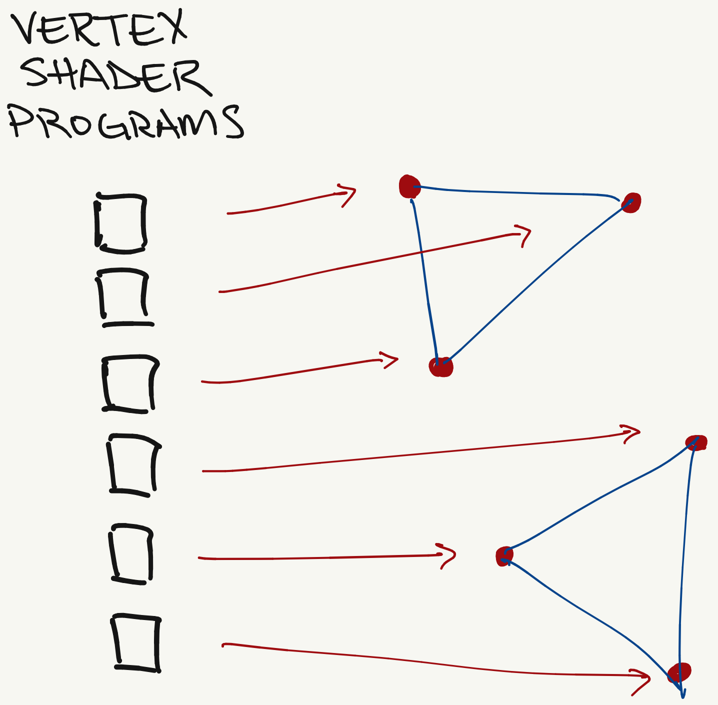
RASTERIZATION

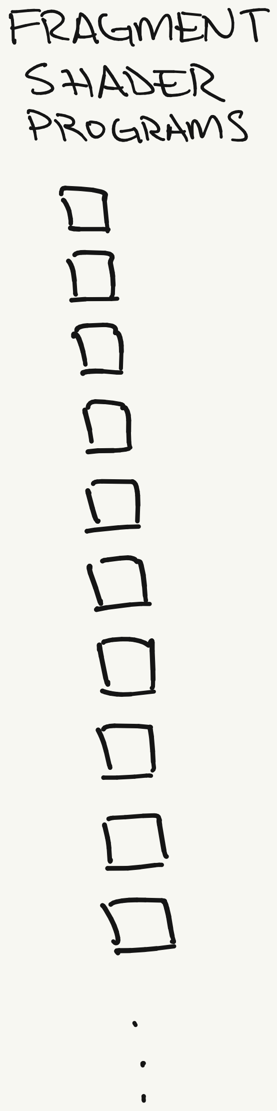


THE FRAG SHADER

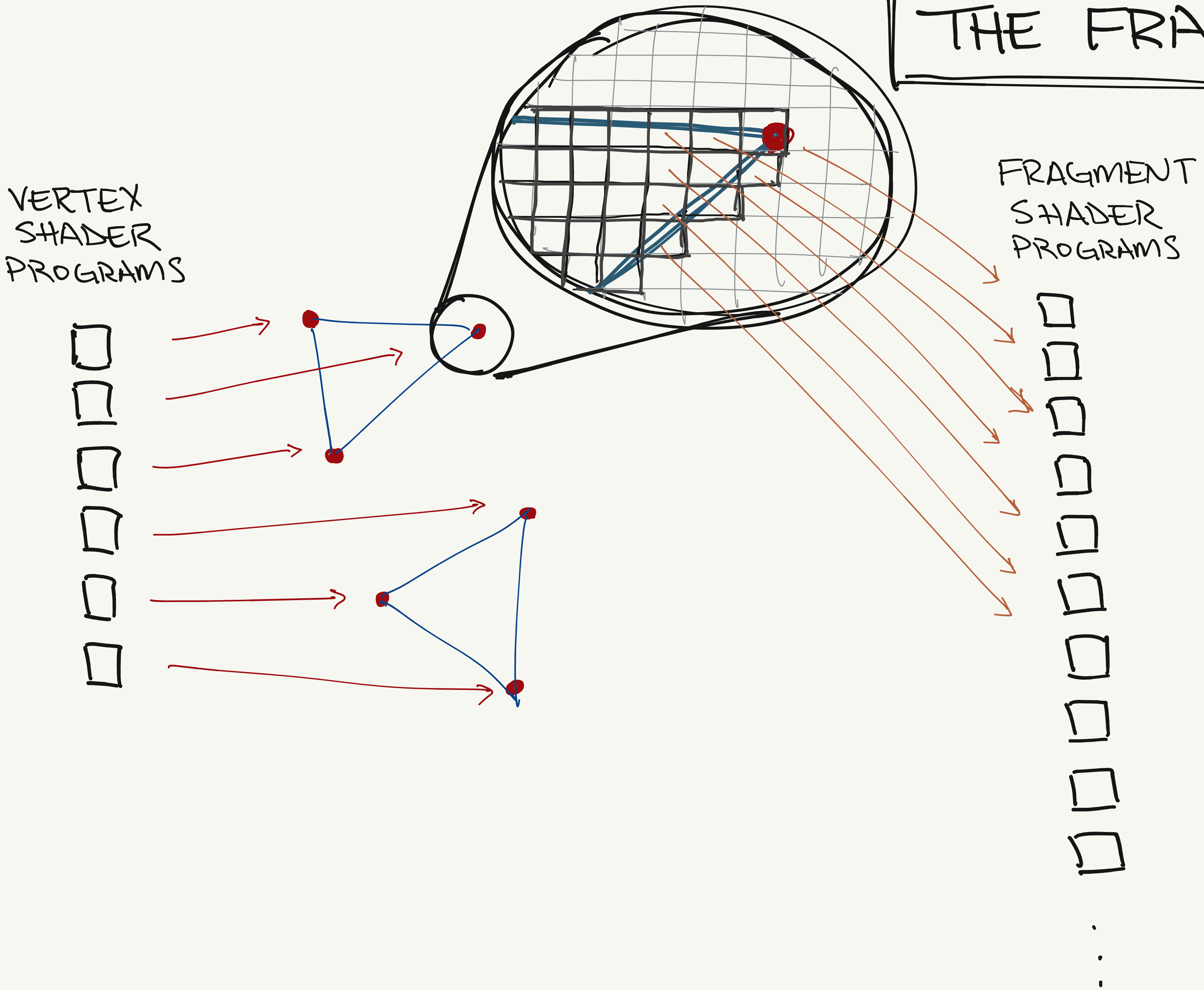


THE FRAG SHADER

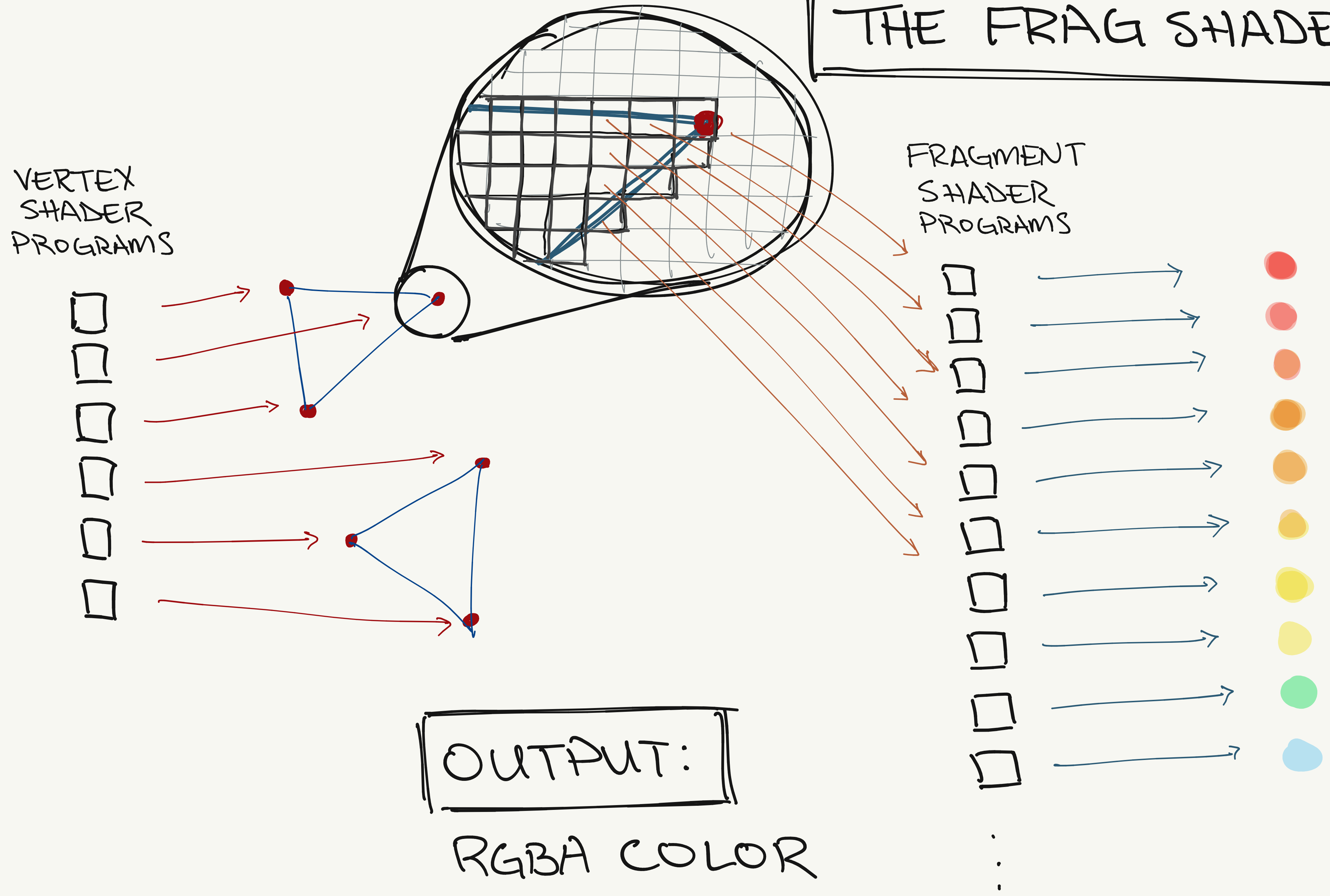




THE FRAG SHADER



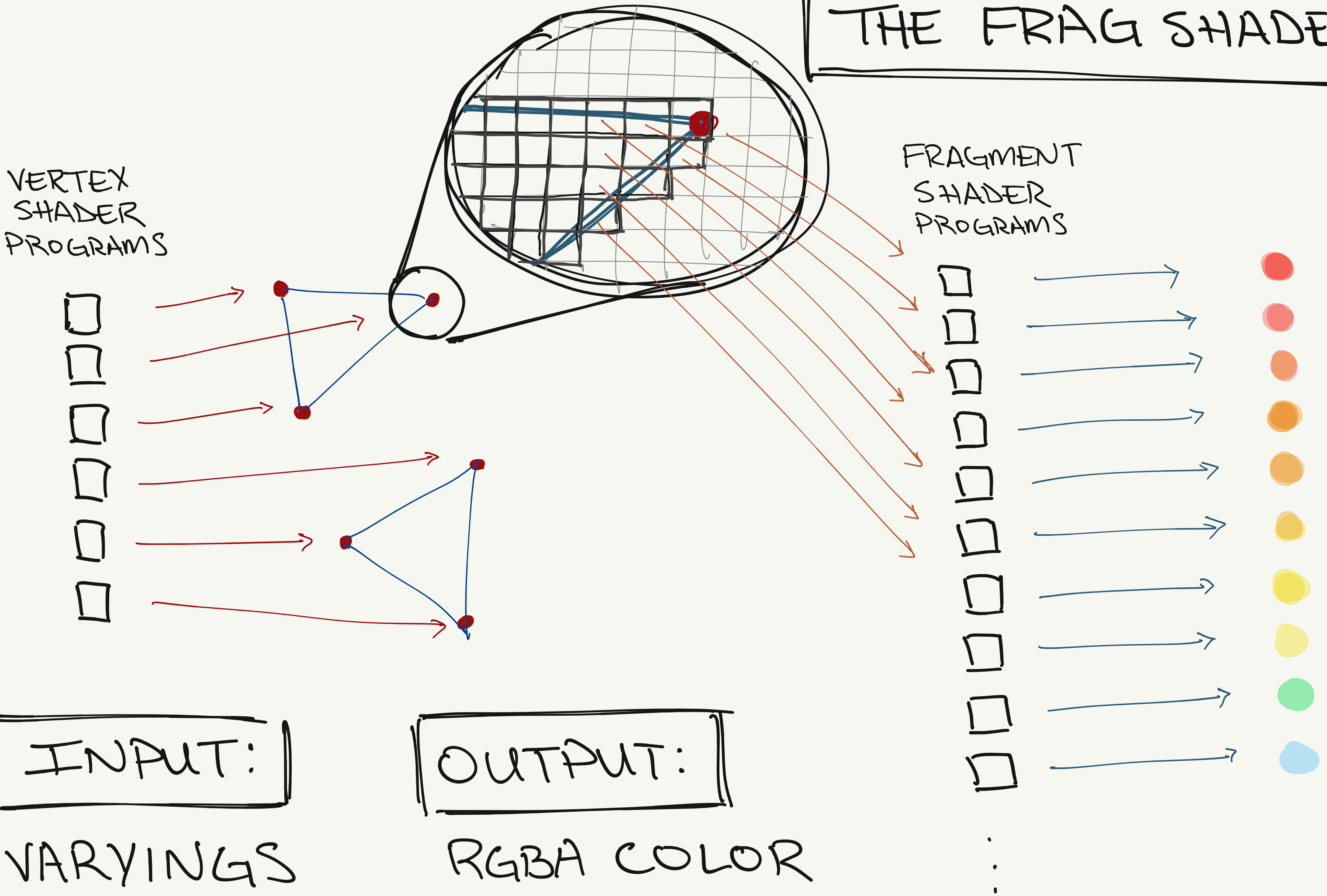
THE FRAG SHADER



OUTPUT:

RGBA COLOR

THE FRAG SHADER



INPUT:

VARYINGS
UNIFORMS

OUTPUT:

RGBA COLOR

STATE + LOGIC

BUFFERS OF BINARY DATA

COMPILED GLSL PROGRAMS (SHADERS!)

STATE + LOGIC

$$\begin{tikzcd}
\text{Uniforms} \arrow[dr] \arrow[drr, bend left] & & & \\
\text{Attributes} \arrow[r] & \text{Vertex Shader} \arrow[r, "\text{RASTERIZE}"'] & \text{Fragment Shader} \arrow[r, "\text{BLEND}"'] & \text{Colors}
\end{tikzcd}$$

LET'S
CODE!