Programming The Pipeline

→ Review the Pipeline

Vertex Stage: ← input: world space vertices
output: NDC vertices w/ more info

Assembled / Primitives Stage: input: triangles clip // subdivide
output: triangles

Rasterization - find fragments
interpolate

Fragment ← input: fragments
output: fragments "filled in" (

Vertex ↓ Fragment
In: partially filled object
Out: fill in key slots

Language Detail (instantiates model)
① model gives variable types
uniform (once per primitive)
vertex (or input) ← call in

varying variables ← called "out" from Vertex shaders

How GLSL does it

- everything passed through variables

shaders are all:

```
void main() { }
```

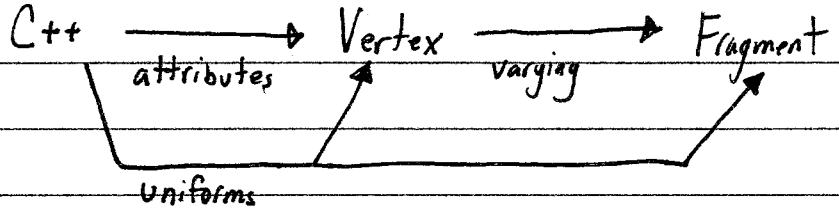
all they do is fill in the magic variables

Some variables are defined by GL

Some variables can be defined by user

↓

Variables always "connect" 2 things



built in: see Appendix I of red book

MVMatrix, MVPMatrix, NormalMatrix	uniforms
gl_Vertex, gl_Color, gl_Normal, gl_MultiTexCoord	attribute
gl_Position, gl_FrontColor	outputs
gl_FragColor, gl_FragDepth,	

11-19

Doing this:

C++ (host)

1. load in source code to shader
2. run compiler (check for errors)
3. attach shaders
4. draw polygon ⇉ send variables
5. detach shader

→ mechanism to see what variables shader wants

- name ⇉ index (per program)
- array of attributes/uniforms