Shader input/output

From host to vertex shader

From vertex shader to fragment shader

From fragment shader to frame buffer

From host to vertex shader

Two variants:
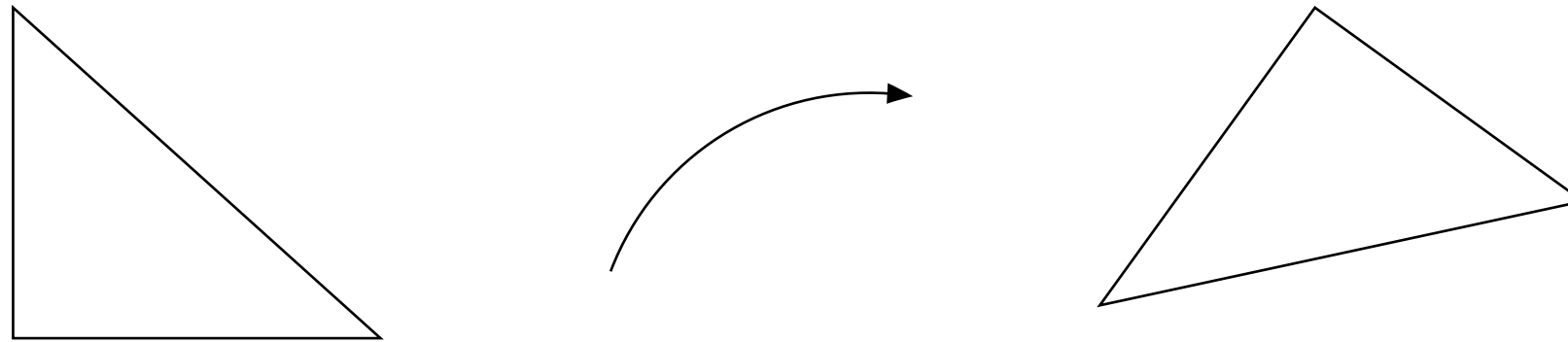
- **Uniform**

- **Attribute**

Uniform

Same for all in a primitive

Typical usage:

• Transformation matrices

• Texture units

• TIme variable

Uniform

**Same rotation
for all vertices**

Applied for entire primitive (e.g. model)

Declare as "uniform" in the shaders

Attributes

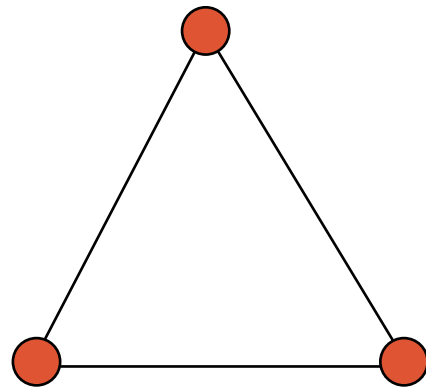**Different for every vertex**

Passed as arrays, as VBOs

Typical usage:

- **Vertices**

- **Normal vectors**

- **Texture coordinates**

Attributes

Every vertex is an attribute.

Delivered in VBOs, vertex array buffers

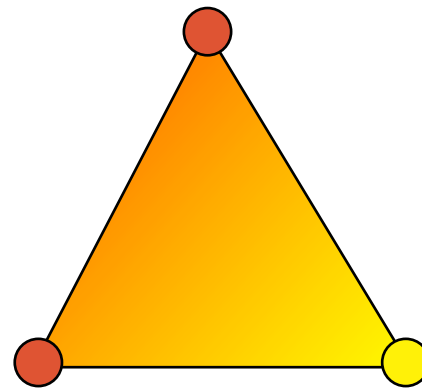**Declare as "in" in the vertex shader**

Varying

"out" from vertex shader

"in" into fragment shader

Interpolated between vertices!

Varying

Values sent from vertex shaders are interpolated and sent to fragments

Simple usage: Set color in each vertex to get a gradient over the polygon
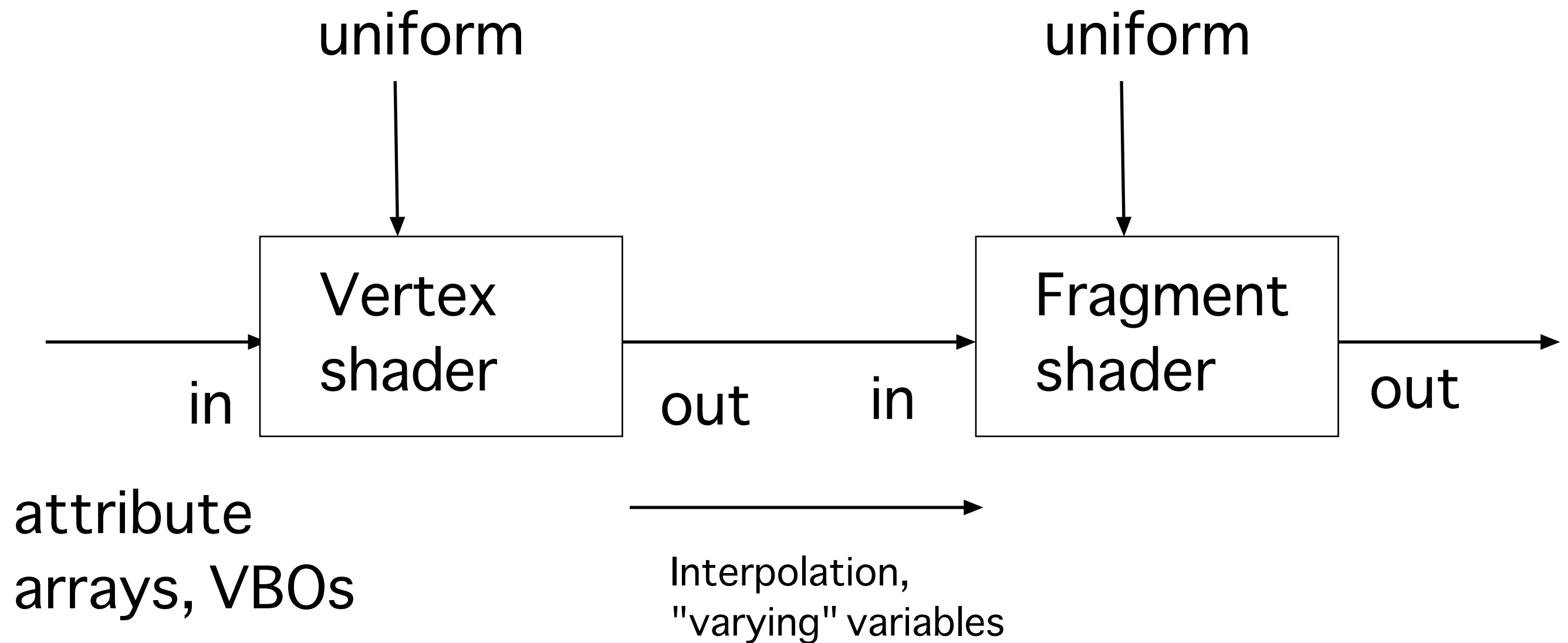
Output from fragment shader

Declared "out"

Typicaly a single output, to the frame buffer

vec3 or vec4

uniform   uniform

Vertex
shader

in   out   in

Fragment
shader

out

attribute
arrays, VBOs

Interpolation,
"varying" variables

Pass-through vertex shader

#version 150

in vec3 in_Position;

void main(void)
{
gl_Position = vec4(in_Position, 1.0);
}

Pass-through fragment shader

#version 150

out vec4 out_Color;

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

More typical vertex shader

```
#version 150

in  vec3 in_Position;
in  vec3 in_Normal;
in  vec2 in_TexCoord;
uniform mat4 mvMatrix;
uniform mat4 projMatrix;

void main(void)
{
gl_Position = projMatrix * mvMatrix * vec4(in_Position, 1.0);
exNormal = mat3(modelViewMatrix) * in_Normal;
exTexCoord = in_TexCoord;
}
```

More typical fragment shader

```
#version 150

in exTexCoord;
in exNormal;
out vec4 out_Color;

void main(void)
{
// Texture lookups, light calculations...
out_Color = ...;
}
```

Questions on shaders?

Very important concept,
not worth leaving unclear!