Human GPU #0016 – Varyings

How do we draw something more interesting then just a plain color?

How can we get some sort of variables/dynamic thing in the fragment shader so to add some logic?
With `varying`, of course!

Along to the type qualifiers `attribute` and `uniform`, we are introducing `varying` variables.
Is the only way to pass information from the vertex shader to the fragment shader.

A varying is defined per each vertex, and its valued is passed along the pipeline to the fragment shader.
This value will then be interpolated linearly along the pixels. (mindblowing, right??)

Let's say we draw a line, composed from points (vertices) A and B, that is stretched along 100 pixels width.

- vertex A has a varying with value 0.0
- vertex B has a varying with value 1.0

This will result in:

- pixel at 1px will get the varying value as 0.0
- pixel at 100px will get the varying value as 1.0
- pixel at 50px will get the varying value as 0.5
- pixel at 75px will get the varying value as 0.75

Let's give it a try and draw a triangle!

Buffers

```
{
  "position": [ 0.0, -0.8, -0.8, 0.8, 0.8, 0.8 ],
  "darkness": [ 1.0, 0.0, 0.0 ]
}
```

Attributes

```
{
  "aPosition": { "buffer": "position", "size": 2 },
  "aDarkness": { "buffer": "darkness", "size": 1 }
}
```

Vertex shader

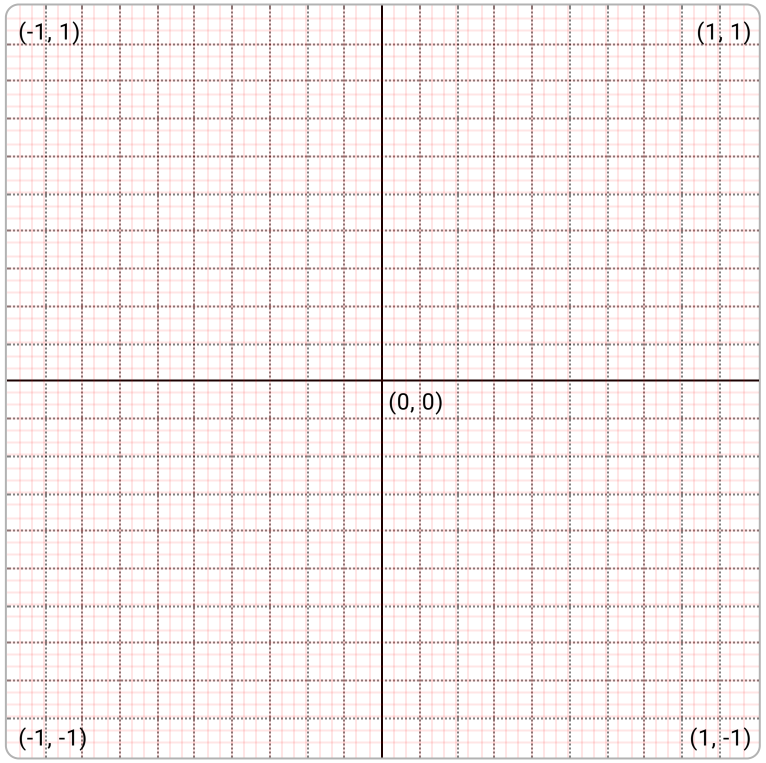
```
attribute vec2 aPosition;
attribute float aDarkness;
varying float vDark;

void main() {
  vDark = aDarkness;
  gl_Position = vec4(aPosition.xy, 0.0, 1.0);
}
```

Fragment shader

```
precision highp float;
uniform vec3 uColor;
varying float vDark;

void main() {
  gl_FragColor = vec4(vec3(1.0 - vDark), 1.0);
}
```


(-1, 1)
(1, 1)
(0, 0)
(-1, -1)
(1, -1)