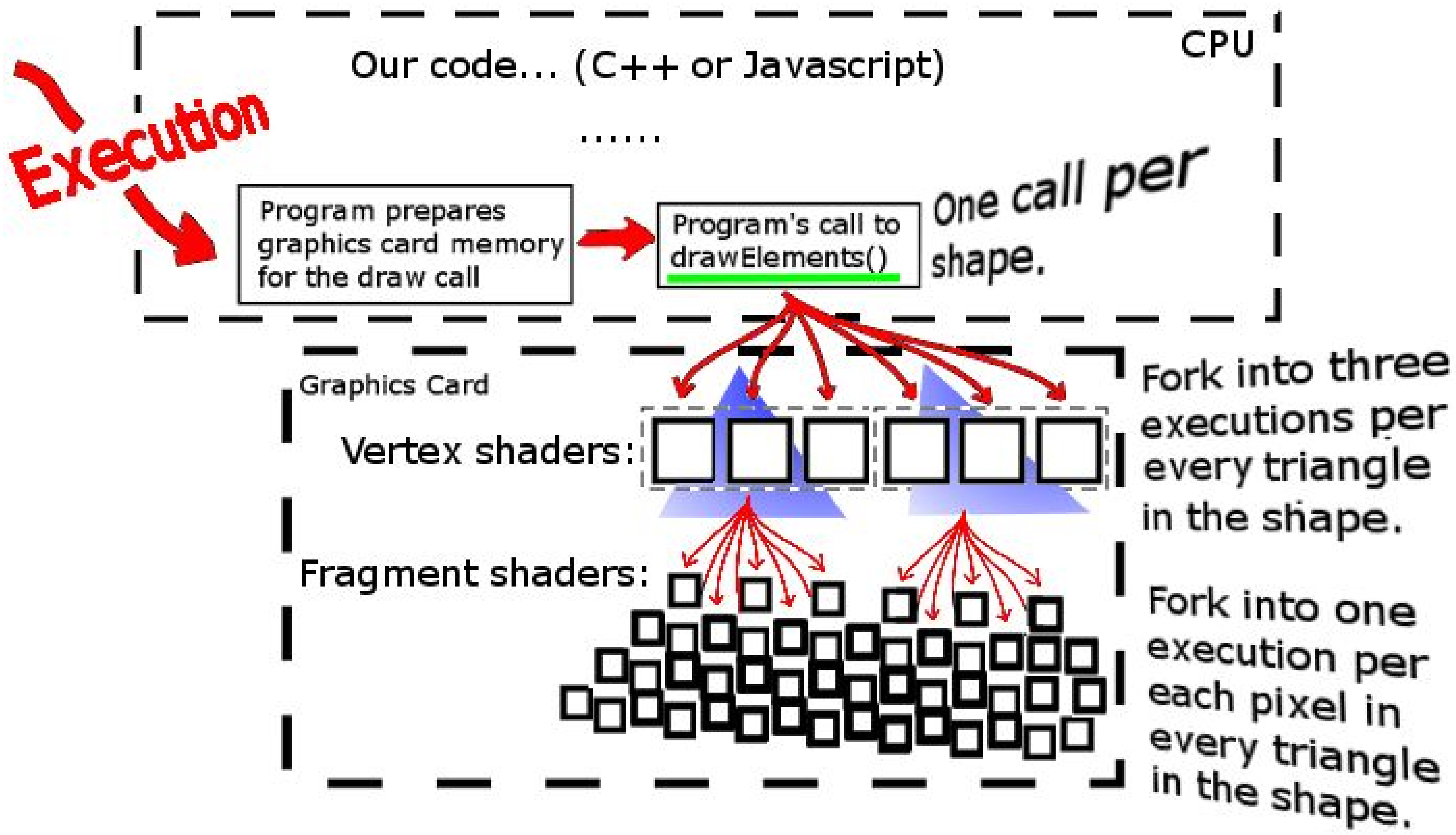
What’s a shader?



CPU

Our code... (C++ or Javascript)

......

Execution

Program prepares graphics card memory for the draw call

Program's call to drawElements()

One call per shape.

Graphics Card

Vertex shaders:

Fork into three executions per every triangle in the shape.

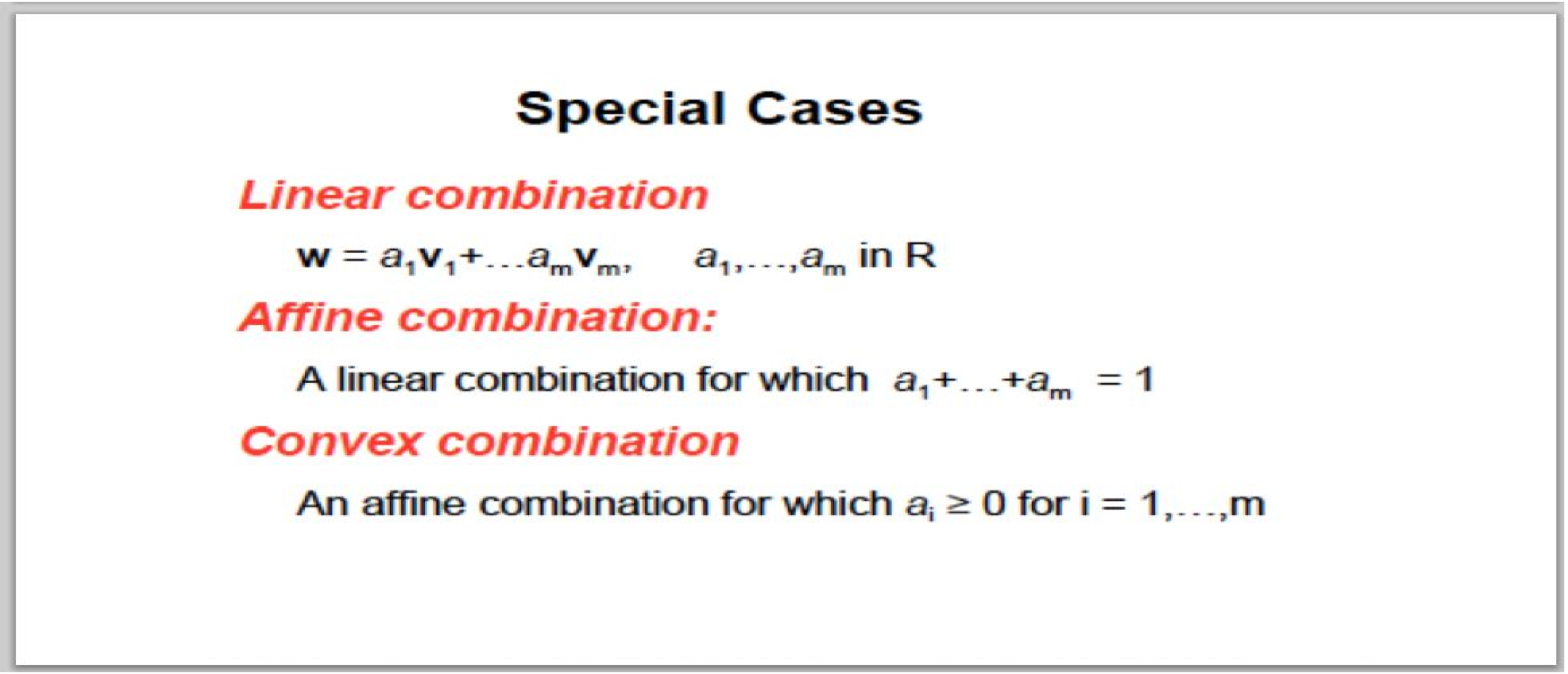
Fragment shaders:

Fork into one execution per each pixel in every triangle in the shape.

A Vertex shader places all three points in their final place.

A Fragment shader samples color-relevant data in between the three vertices that have the data, using interpolation.

Interpolation



Special Cases

Linear combination

$\mathbf{w} = a_1\mathbf{v}_1 + \ldots a_m\mathbf{v}_m, \quad a_1,\ldots,a_m$ in R

Affine combination:

A linear combination for which $a_1 + \ldots + a_m = 1$

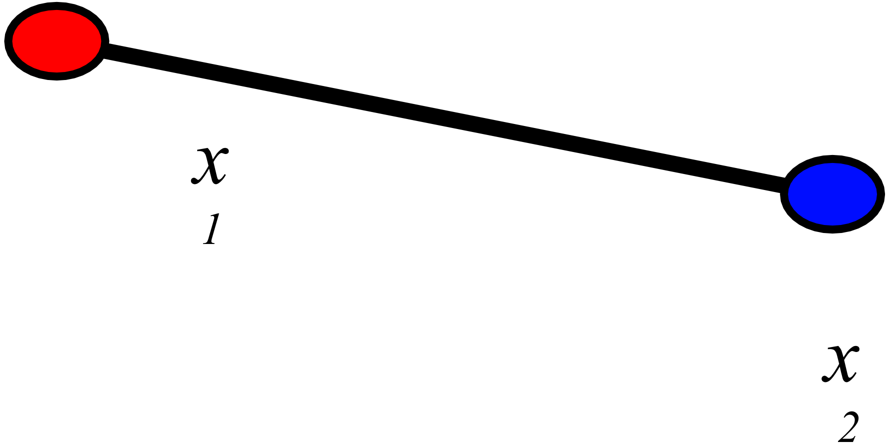
Convex combination

An affine combination for which $a_i \geq 0$ for $i = 1,\ldots,m$

Barycentric Interpolation

How do you interpolate values defined at vertices across the entire triangle?

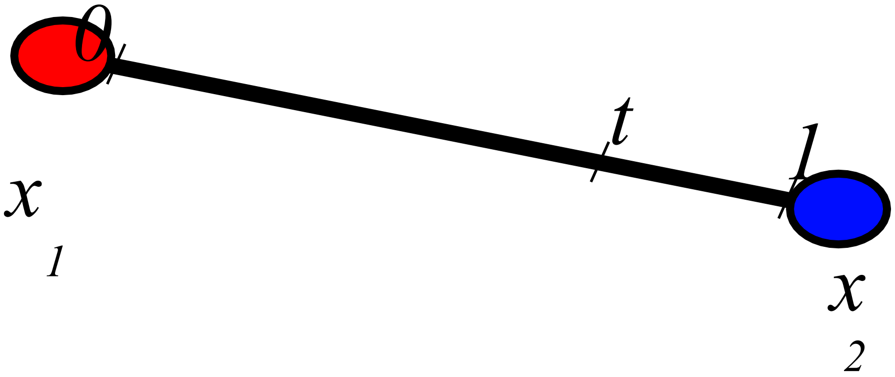
Solve a simpler 1D problem first (each vertex has a color):



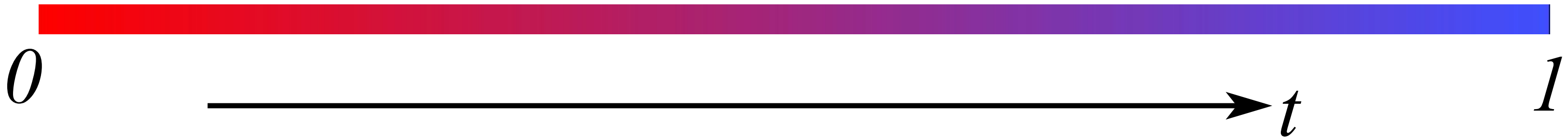
Barycentric Interpolation

How do you interpolate values defined at vertices across the entire triangle?

Solve a simpler problem first:



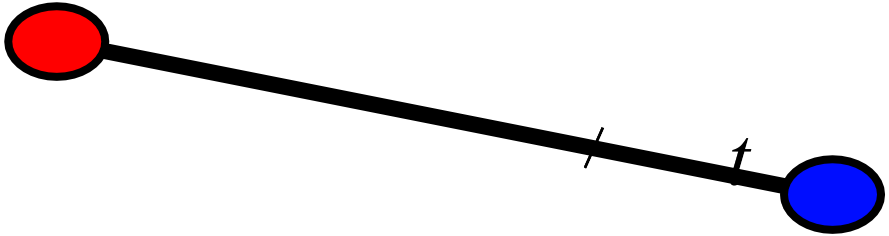
Want to define a value for every $t \, \varepsilon \, [0,1]$:



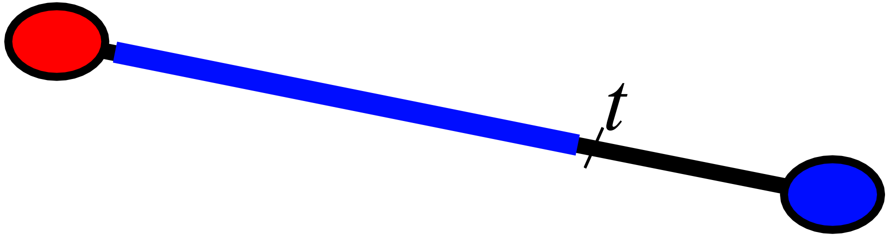
Barycentric Interpolation

How do we come up with this equation?

Look at the picture!

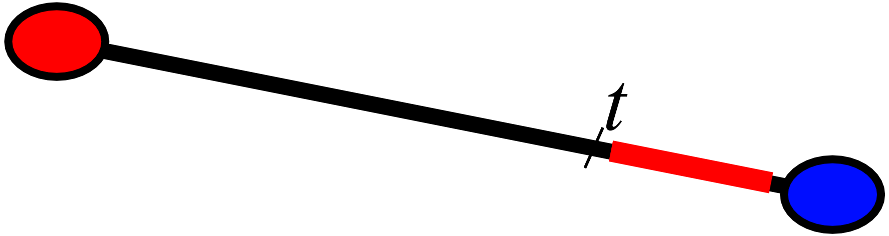


Barycentric Interpolation



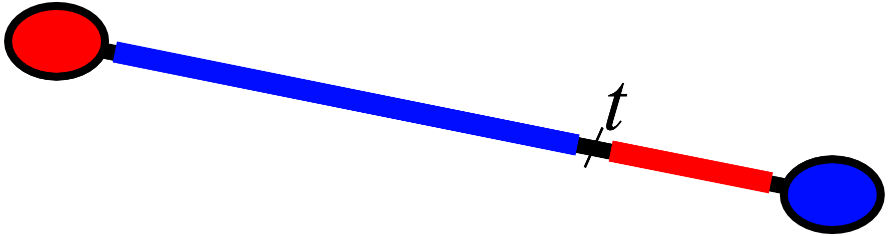
The further t is from the red point, the more blue we want.

Barycentric Interpolation



The further t is from the red point, the more blue we want. The further t is from the blue point, the more red we want.

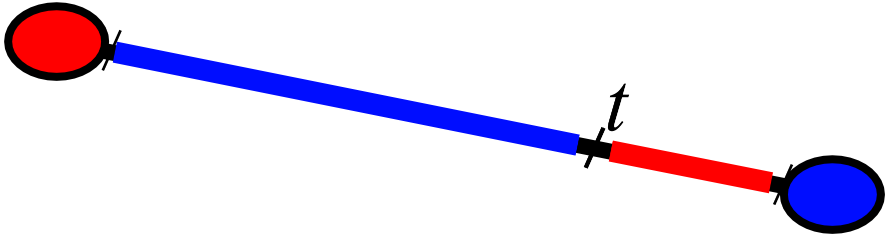
Barycentric Interpolation



The further t is from the red point, the more blue we want. The further t is from the blue point, the more red we want.

Percent blue = (length of blue segment)/(total length)

Barycentric Interpolation

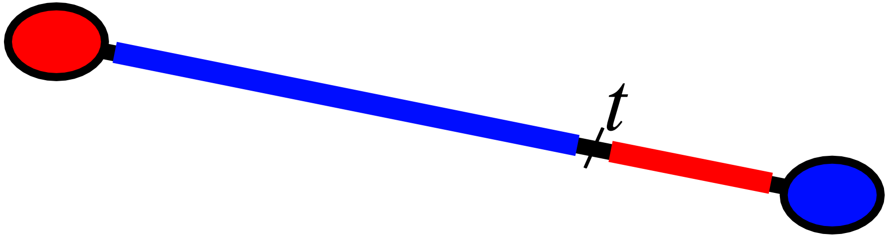


The further t is from the red point, the more blue we want. The further t is from the blue point, the more red we want.

⬇

Percent blue = (length of blue segment)/(total length) Percent red = (length of red segment)/(total length)

Barycentric Interpolation



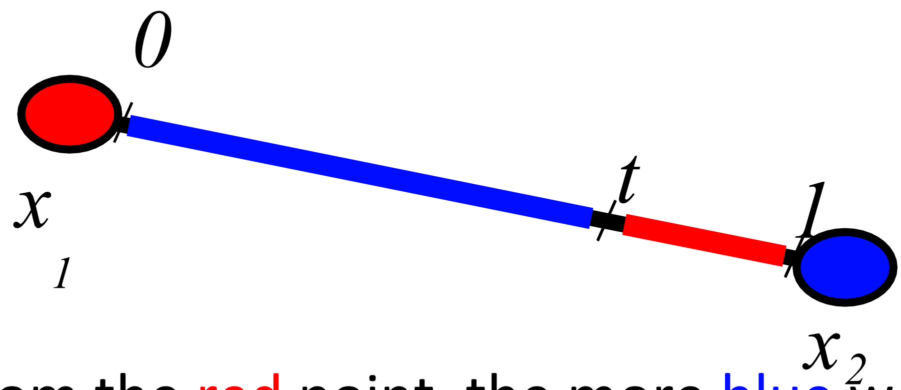
The further t is from the red point, the more blue we want. The further t is from the blue point, the more red we want.

⬇

Percent blue = (length of blue segment)/(total length) Percent red = (length of red segment)/(total length)

Value at t = (% blue)(value at blue) + (% red)(value at red)

Barycentric Interpolation



The further *t* is from the red point, the more blue we want. The further *t* is from the blue point, the more red we want.

Percent blue = *t*

Percent red = *1-t*

Value at *t* = $tx_1 + (1\text{-}t)x_2$

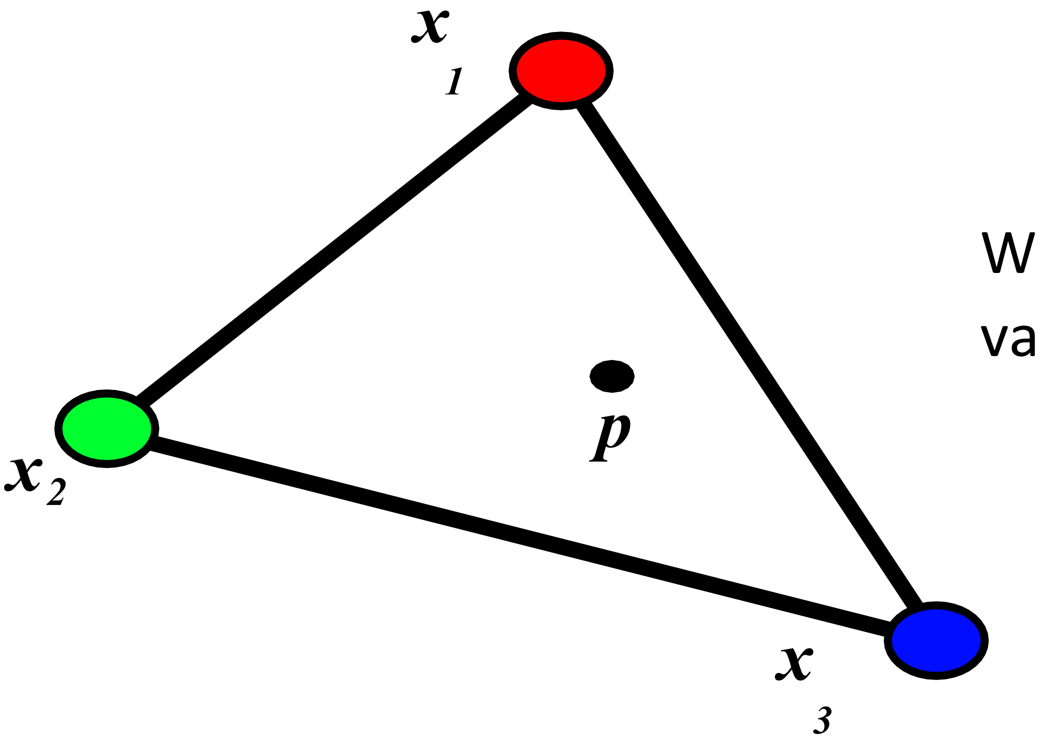
Barycentric Interpolation

Now what about triangles?

Barycentric Interpolation

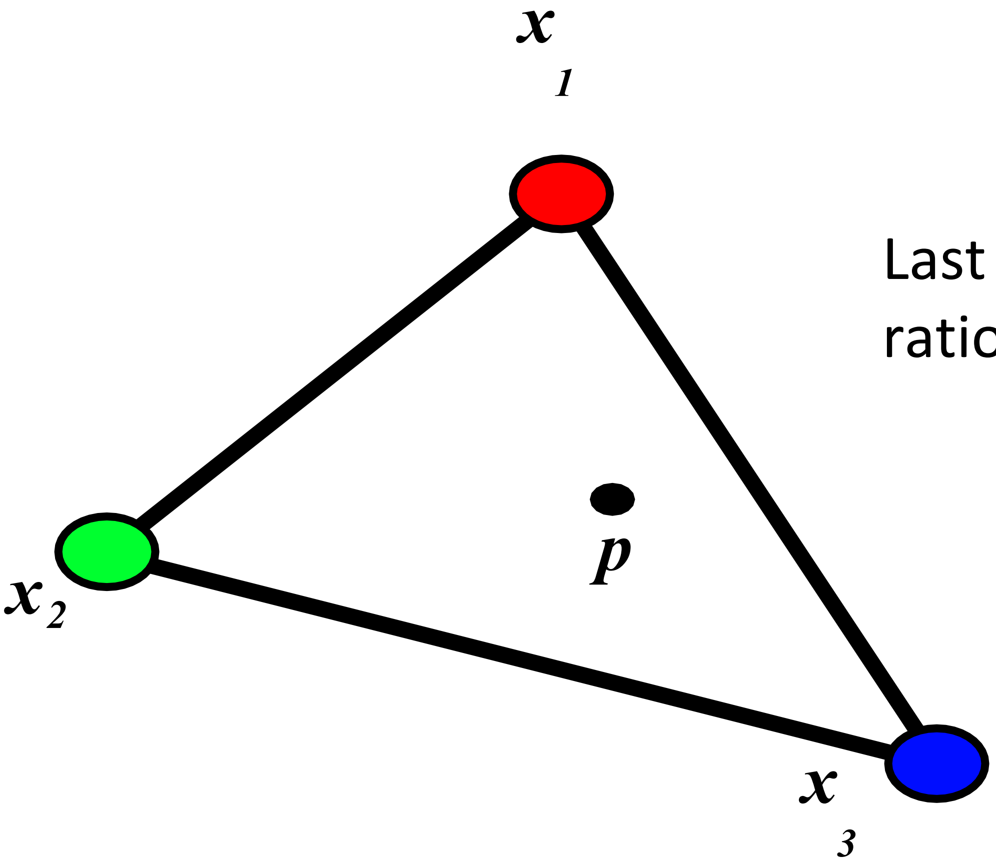
Now what about triangles?

Just consider the geometry:

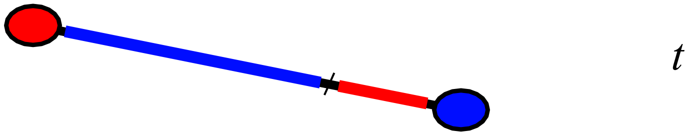


What's the interpolated value at the point p?

Barycentric Interpolation

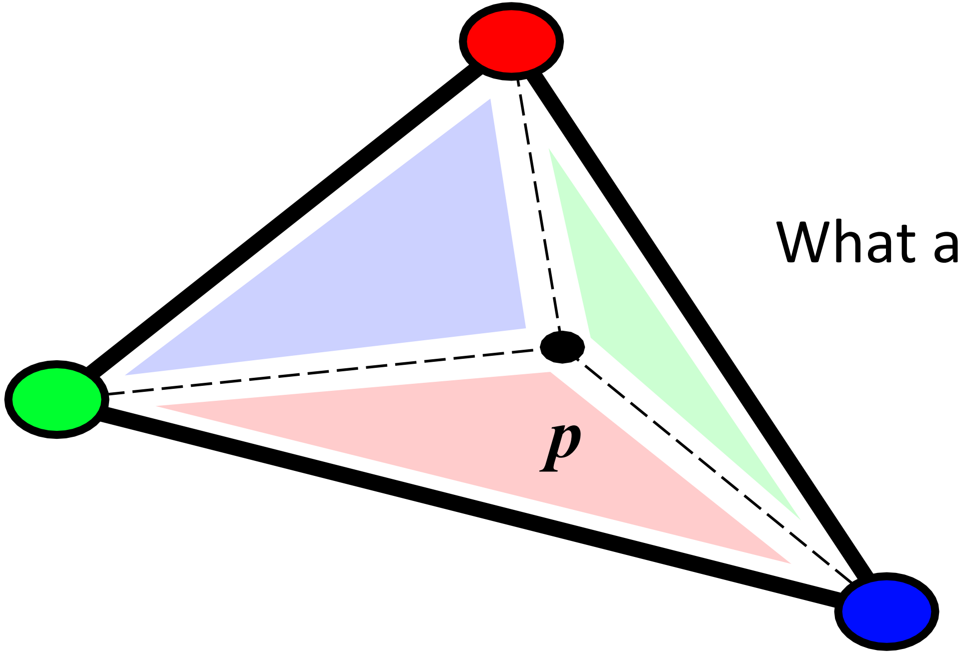


Last time (in 1D) we used ratios of lengths.



Barycentric Interpolation

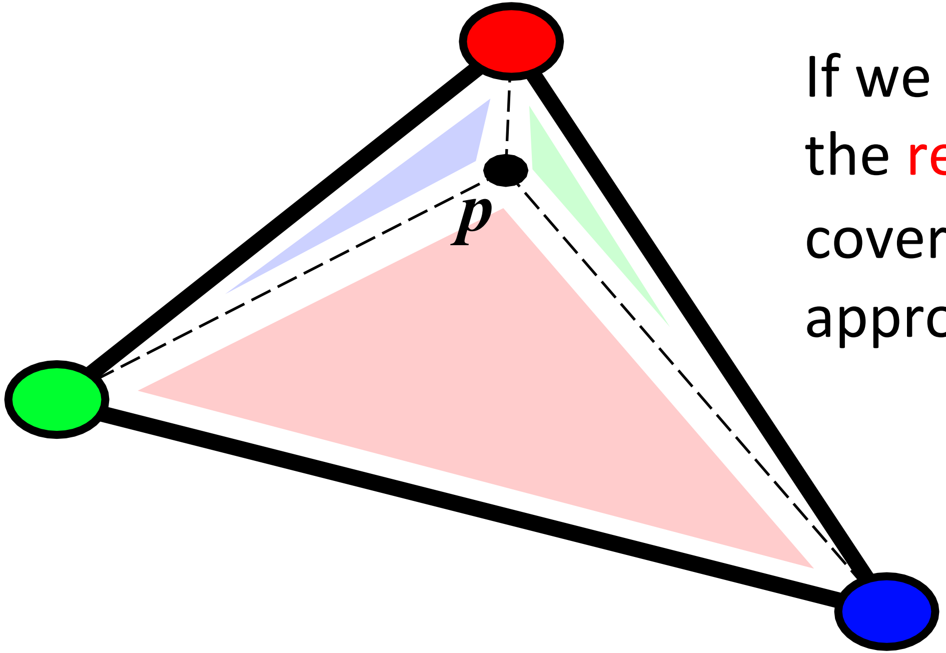
Now what about triangles? Just consider the geometry:



What about ratios of areas (2D)?

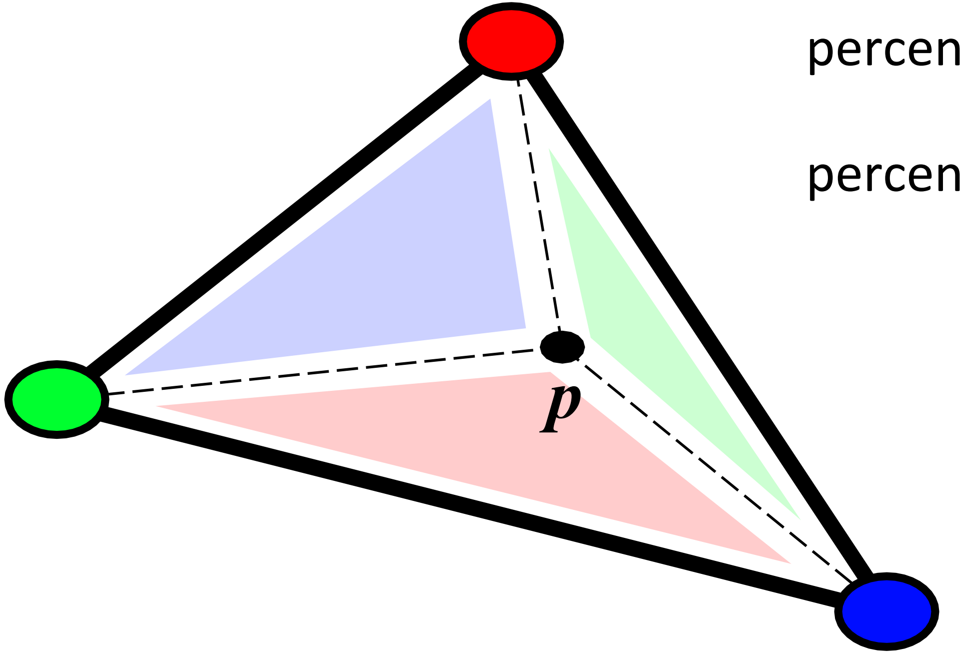
Barycentric Interpolation

Now what about triangles? Just consider the geometry:



If we color the areas carefully, the red area (for example) covers more of the triangle as p approaches the red point.

Barycentric Interpolation



Just like before:

$$\text{percent red} = \frac{\text{area of red triangle}}{\text{total area}}$$

$$\text{percent green} = \frac{\text{area of green triangle}}{\text{total area}}$$

$$\text{percent blue} = \frac{\text{area of blue triangle}}{\text{total area}}$$

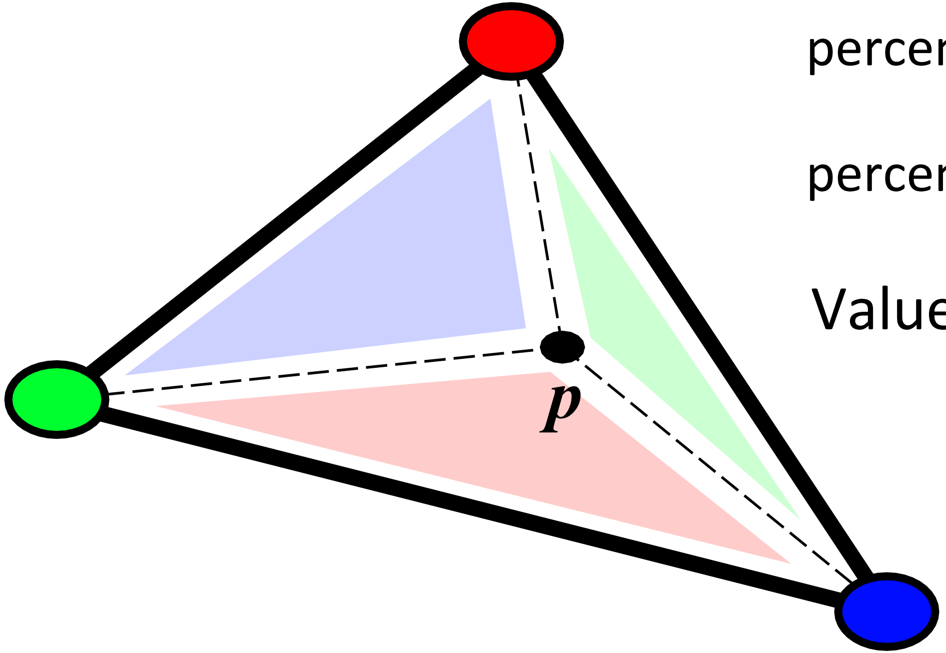
Barycentric Interpolation

Just like before:

$$\text{percent red} = \frac{\text{area of red triangle}}{\text{total area}}$$

$$\text{percent green} = \frac{\text{area of green triangle}}{\text{total area}}$$

$$\text{percent blue} = \frac{\text{area of blue triangle}}{\text{total area}}$$



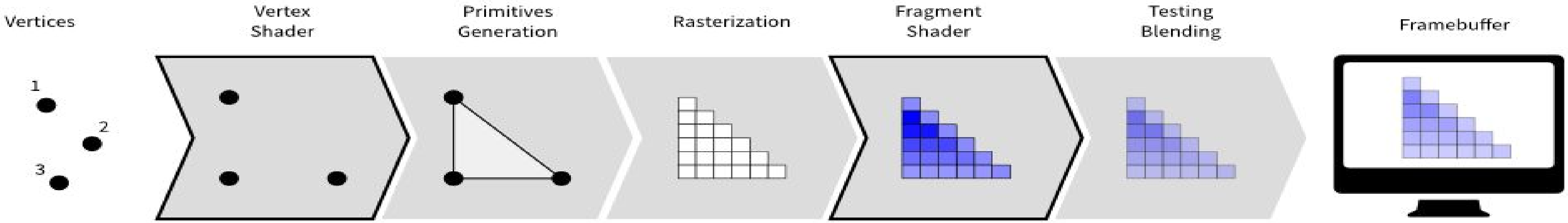
Value at ***p***:

(% red)(value at red) + (% green)(value at green) + (% blue)(value at blue)

A Fragment shader samples color-relevant data in between the three vertices that have the data, using interpolation.

- Every variable in the GLSL programs that has the qualifier "**varying**" is received by the fragment shader as already interpolated from the three extreme vertex points' values for that variable, weighted according to the fragment's position in the triangle (barycentric coordinates)

The Process


Vertices
1
2
3
Vertex Shader
Primitives Generation
Rasterization
Fragment Shader
Testing Blending
Framebuffer