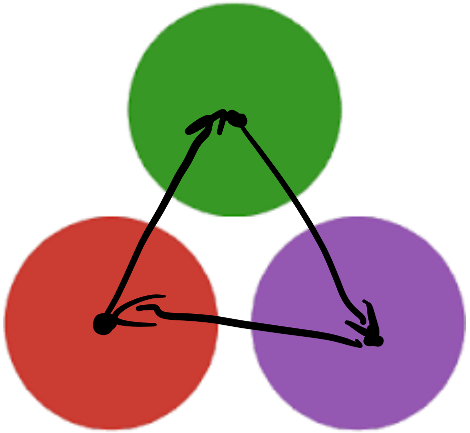
Computer Graphics

Lecture 2

Julia and Vectors



Announcements

- I made a Discord server! Link on the Syllabus page of Canvas
- HW0 due Friday!
  - Some math with vectors, to achieve point-in-triangle test
- A0 due Monday!
  - Draw a triangle! Note: code due Monday, artifact due Tuesday
- There will be videos to watch this week, for
  - L04 - Wednesday (10/2)
  - L05 - Friday (10/4)
- Also a handful more coming up: L07, L10, L11, L14
- For Friday, having one laptop per group will be useful for the in-class problems; please bring one if you can!

Goals

- Know how to manipulate the pixel values of an image in Julia
- ✓ Understand that raster images are *sampled* from an underlying ideal image.
- Be able to work with and interpret basic vector operations, including addition, subtraction, dot, and cross products.

Roadmap

- Last time: The Simplest Possible Graphics System(TM) that is A0.

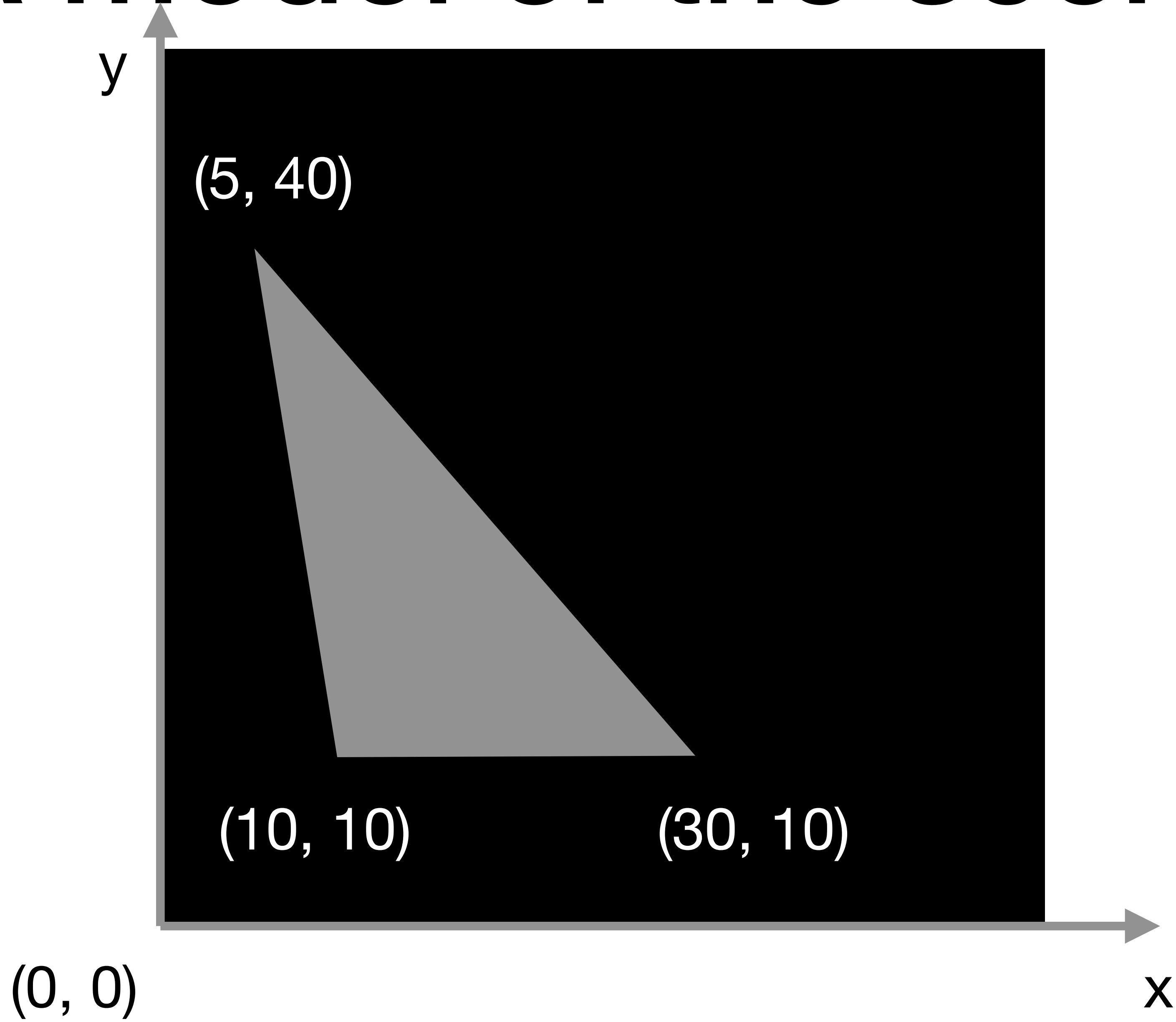
  1. Model a triangle (3 vertex positions, counter-clockwise)

  2. Draw an image of-- wait, what's an image?

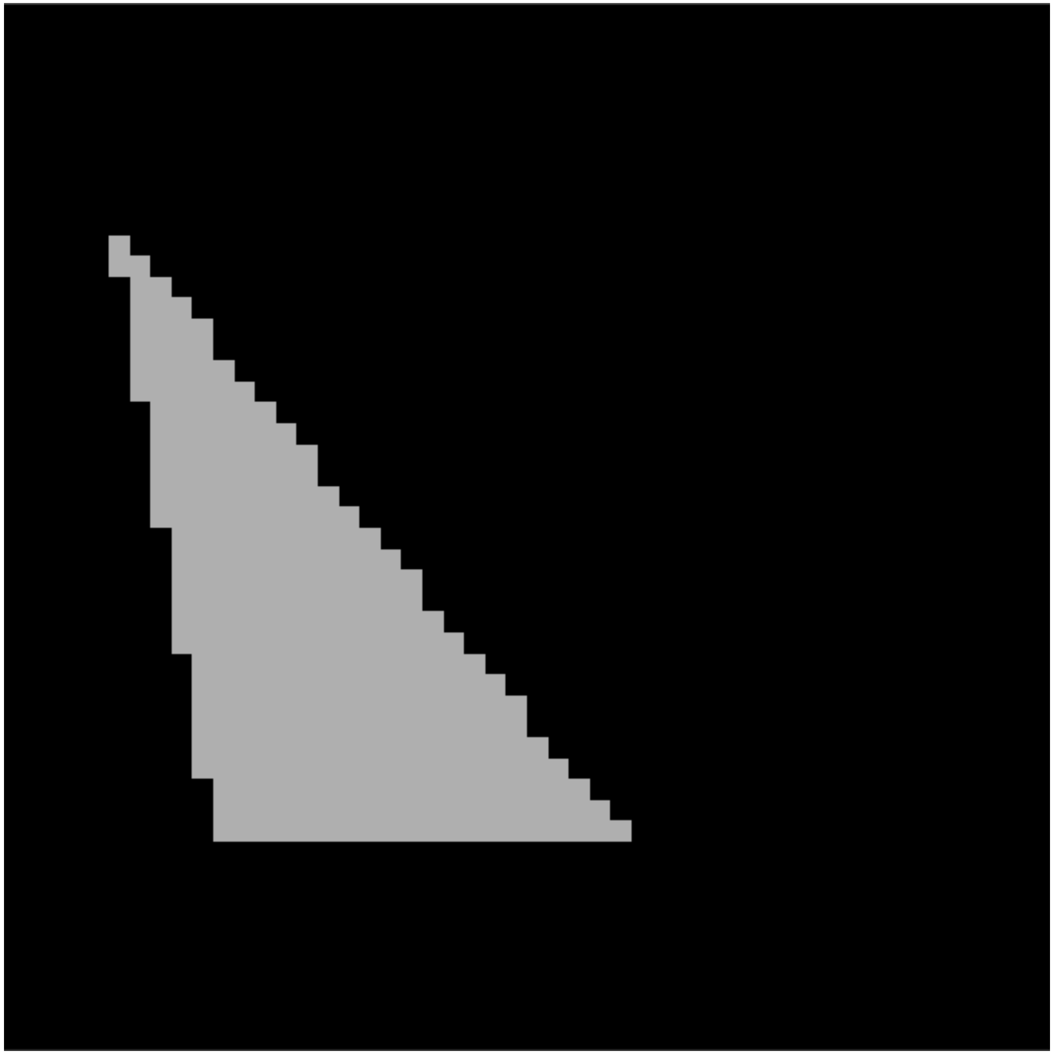
- Today: finishing up the Simplest Possible Graphics System(TM)

  ~~3~~2. Draw an image of the triangle

A model of the scene

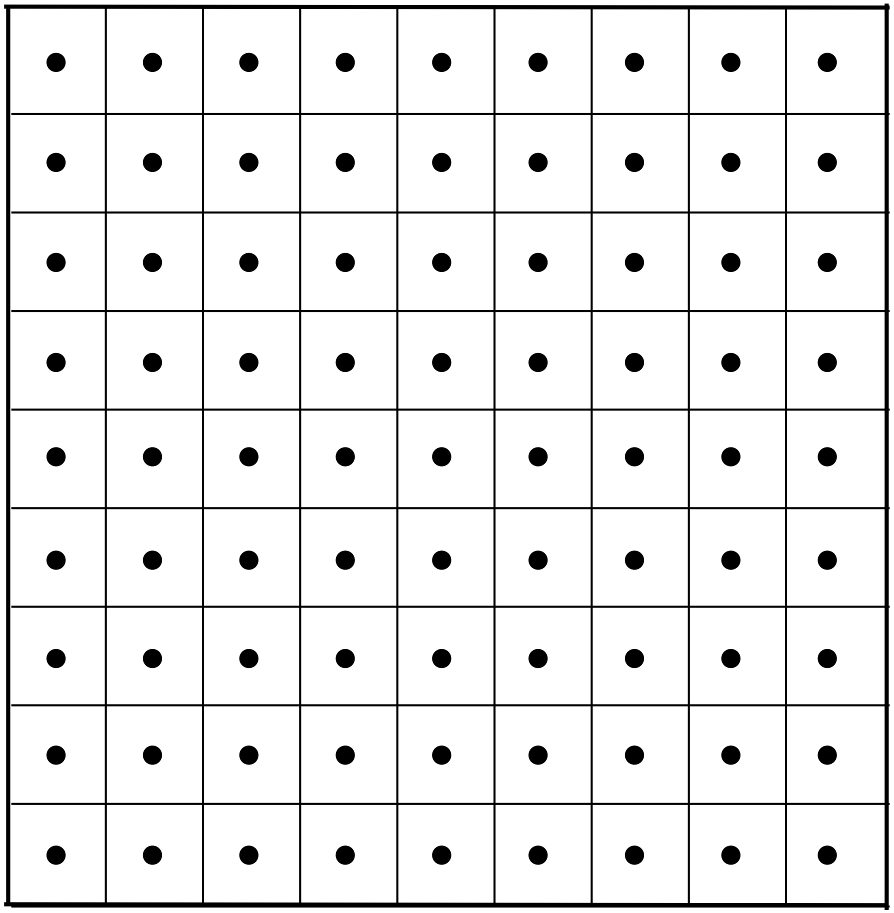


A Raster Image of the Scene



Representing Raster Images

What do pixels *mean*?



Convention: a pixel gets the color sampled at the ***center*** of the pixel.

2D Arrays in Julia

Image: A height-by-width array of pixels.

For a color float image, each pixel is 3 single-precision floats:

```
canvas = zeros(RGB{Float32}, height, width)
```

2D Arrays in Julia

Image: A height-by-width array of pixels.

For a color float image, each pixel is 3 single-precision floats:

```
canvas = zeros(RGB{Float32}, height, width)
```

Make an array of zeros…

2D Arrays in Julia

Image: A height-by-width array of pixels.

For a color float image, each pixel is 3 single-precision floats:

```
canvas = zeros(RGB{Float32}, height, width)
```

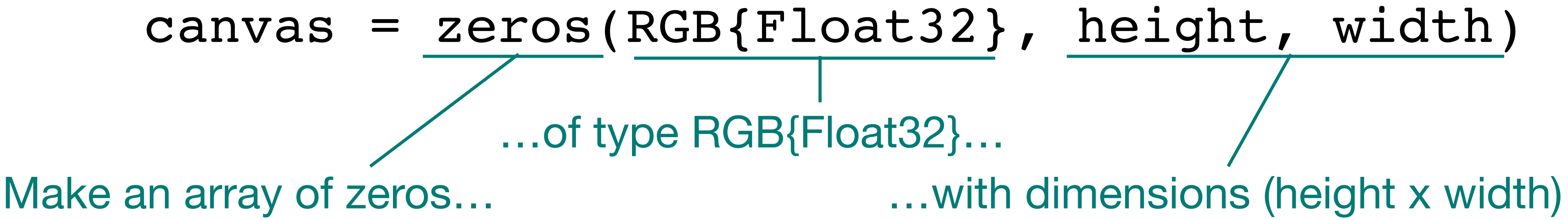
Make an array of zeros…

…with dimensions (height x width)

2D Arrays in Julia

Image: A height-by-width array of pixels.

For a color float image, each pixel is 3 single-precision floats:



```
canvas = zeros(RGB{Float32}, height, width)
```

Make an array of zeros…

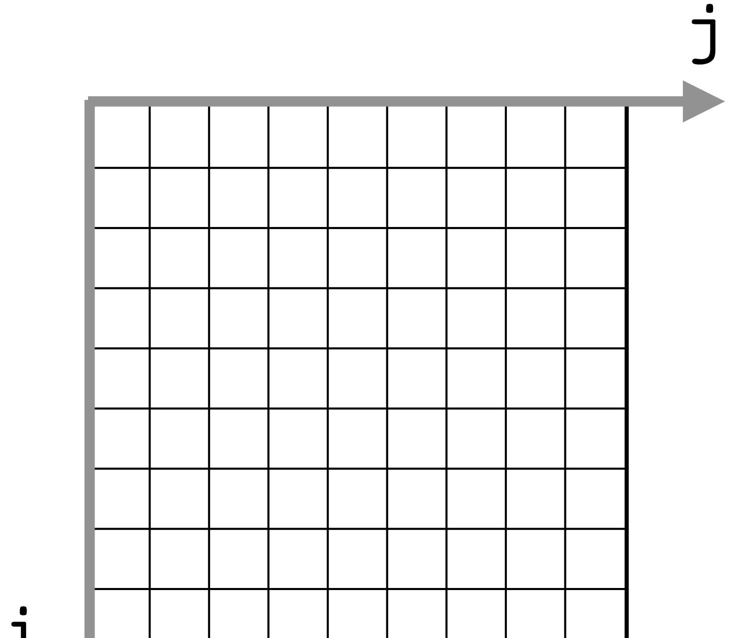
…of type RGB{Float32}…

…with dimensions (height x width)

2D Arrays in Julia

```
canvas = zeros(RGB{Float32}, height, width)
```

Matrix-style **1-based** indexing (row, column):

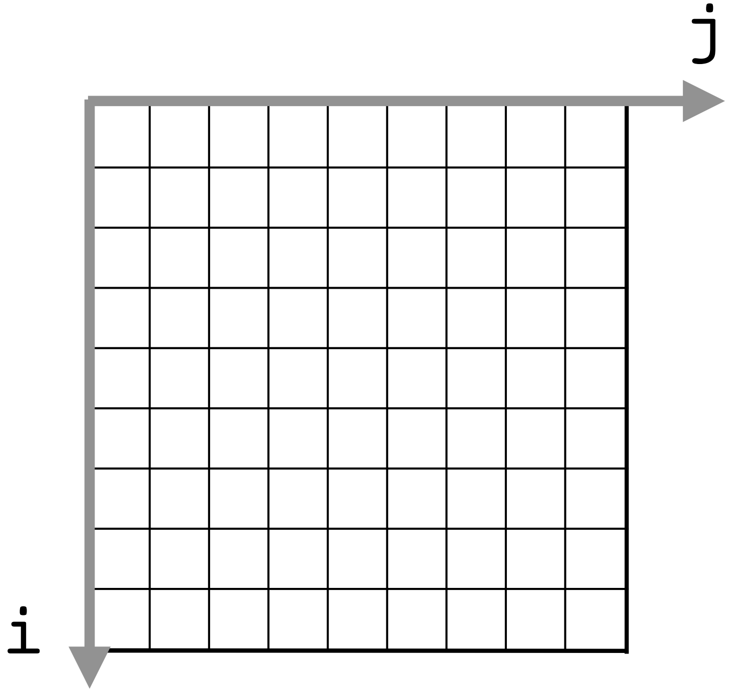


2D Arrays in Julia

```
canvas = zeros(RGB{Float32}, height, width)
```

Matrix-style **1-based** indexing (row, column):

```
canvas[i, j] # is the i'th row, j'th column
```



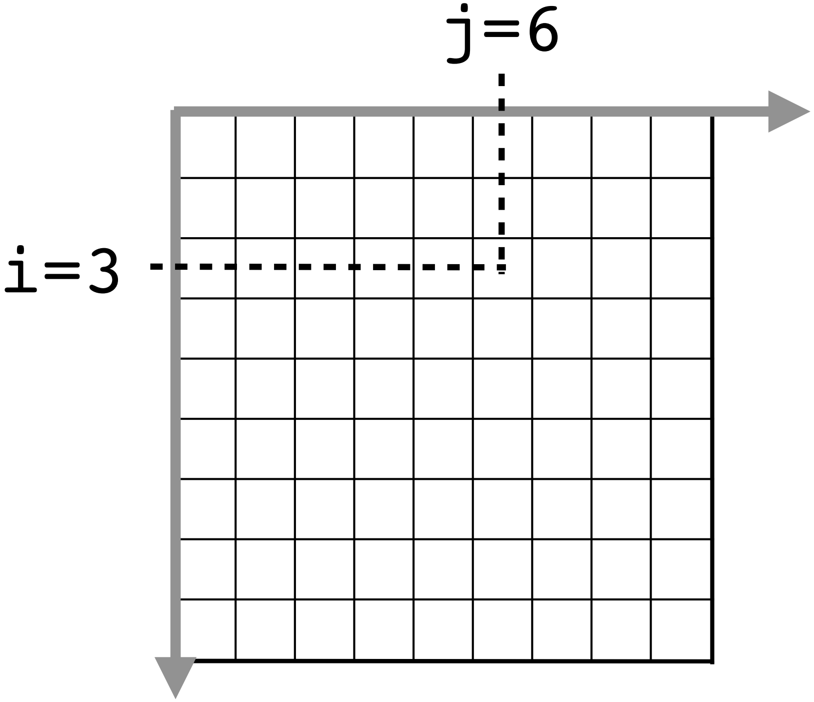
2D Arrays in Julia

```
canvas = zeros(RGB{Float32}, height, width)
```

Matrix-style **1-based** indexing (row, column):

```
canvas[i, j] # is the i'th row, j'th column
```

```
canvas[3, 6]
```



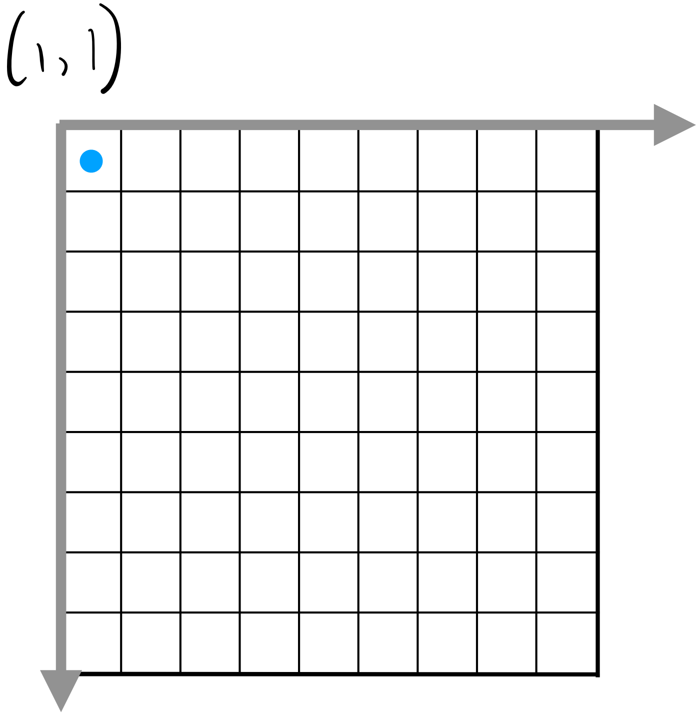
2D Arrays in Julia

```
canvas = zeros(RGB{Float32}, height, width)
```

Matrix-style **1-based** indexing (row, column):

```
canvas[i, j] # is the i'th row, j'th column
```

Q: What are the pixel coordinates of the blue point (the center of the top-left pixel)?



2D Arrays in Julia

```
canvas = zeros(RGB{Float32}, height, width)
```

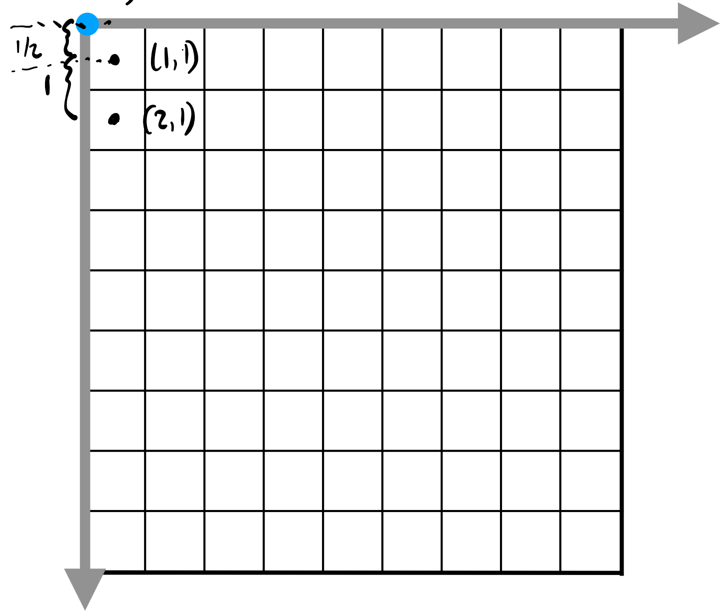
Matrix-style **1-based** indexing (row, column):

```
canvas[i, j] # is the i'th row, j'th column
```

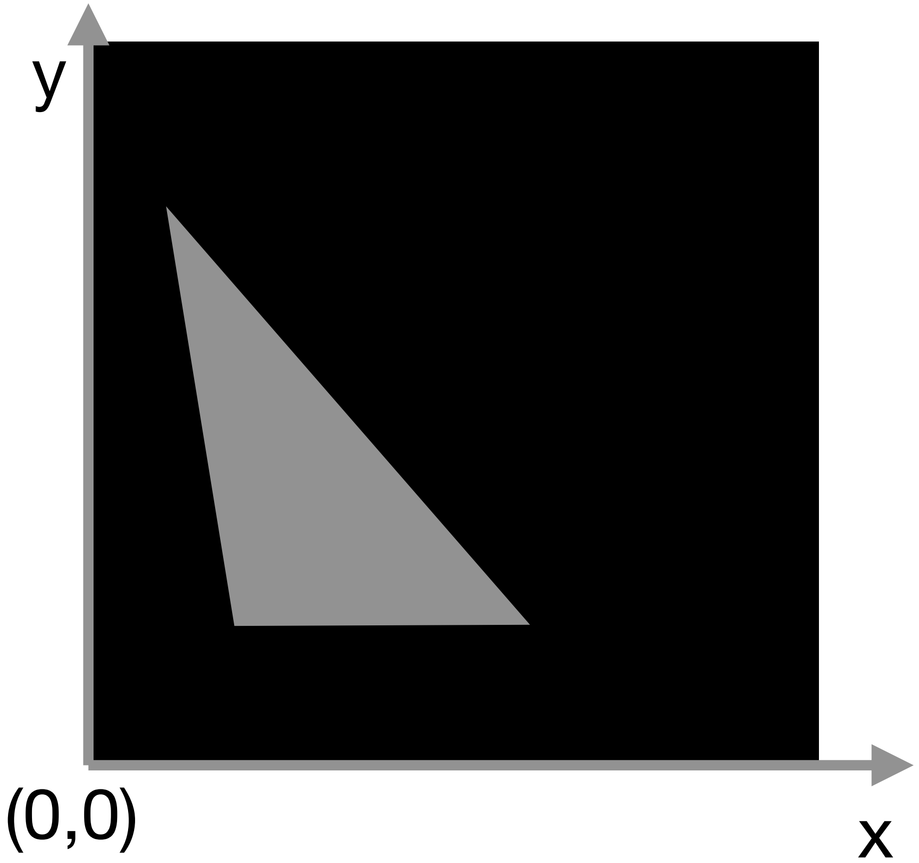
$\boxed{(0.5, 0.5)}$ (0, 0)

(1, 1)

Q: What are the pixel coordinates of the top-left corner of the image?



Raster Images: Coordinate Systems



Model coordinates

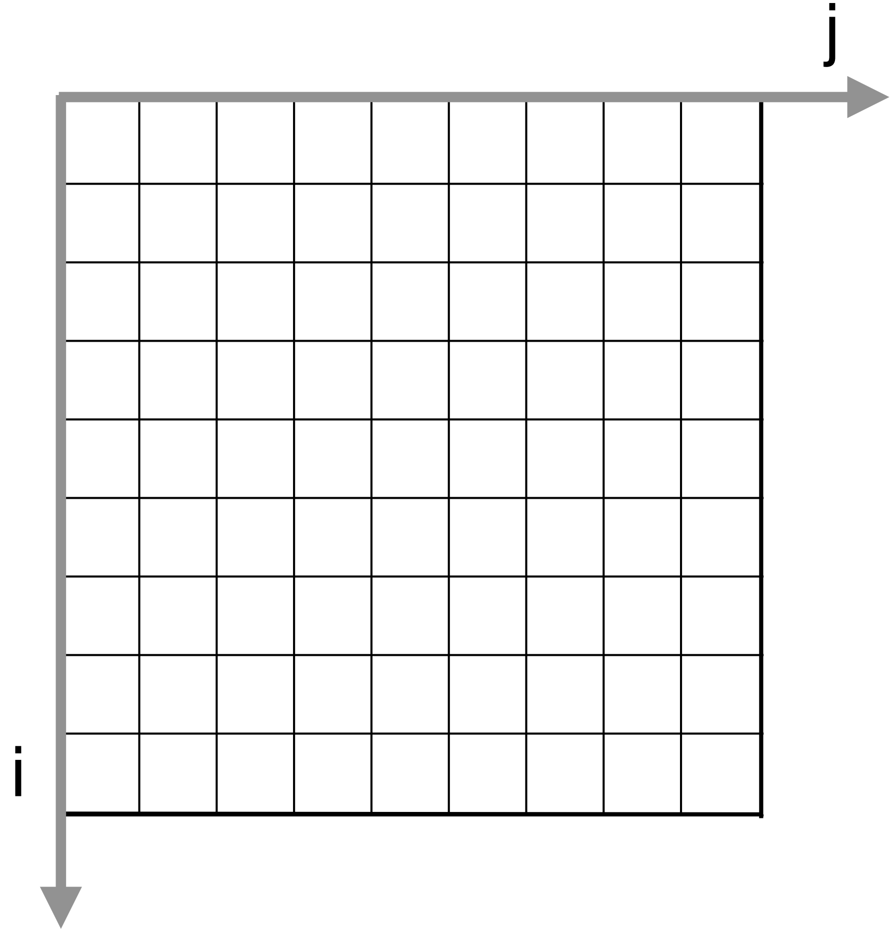
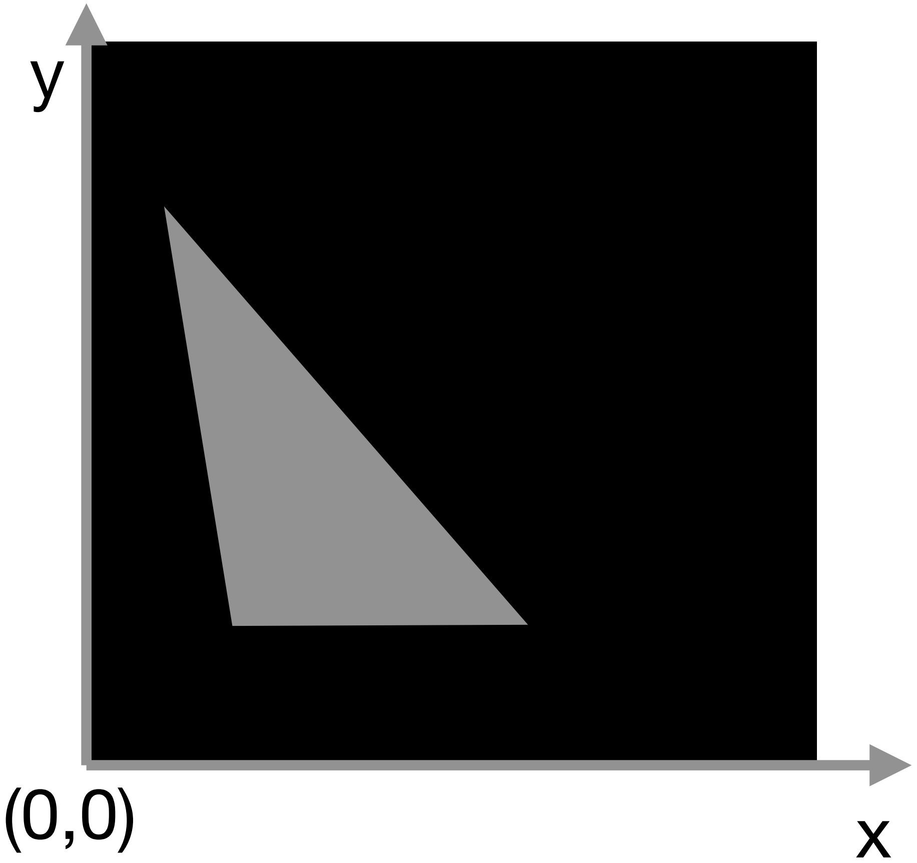


Image coordinates

Raster Images: Coordinate Systems



Model coordinates

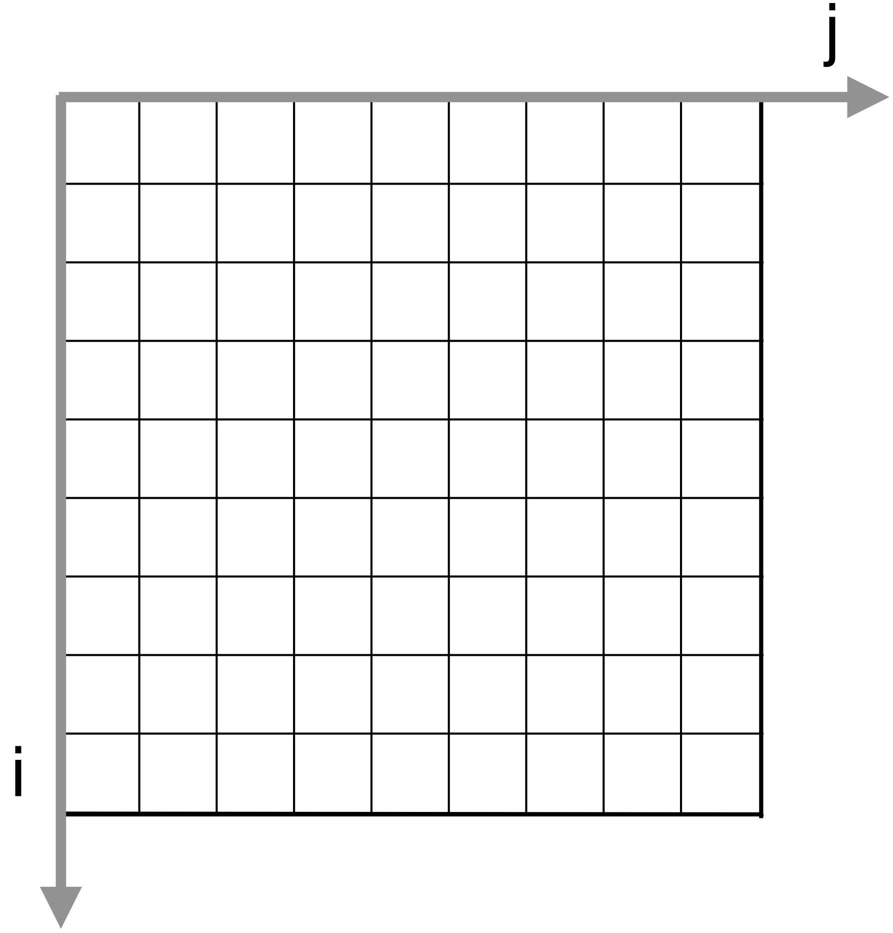
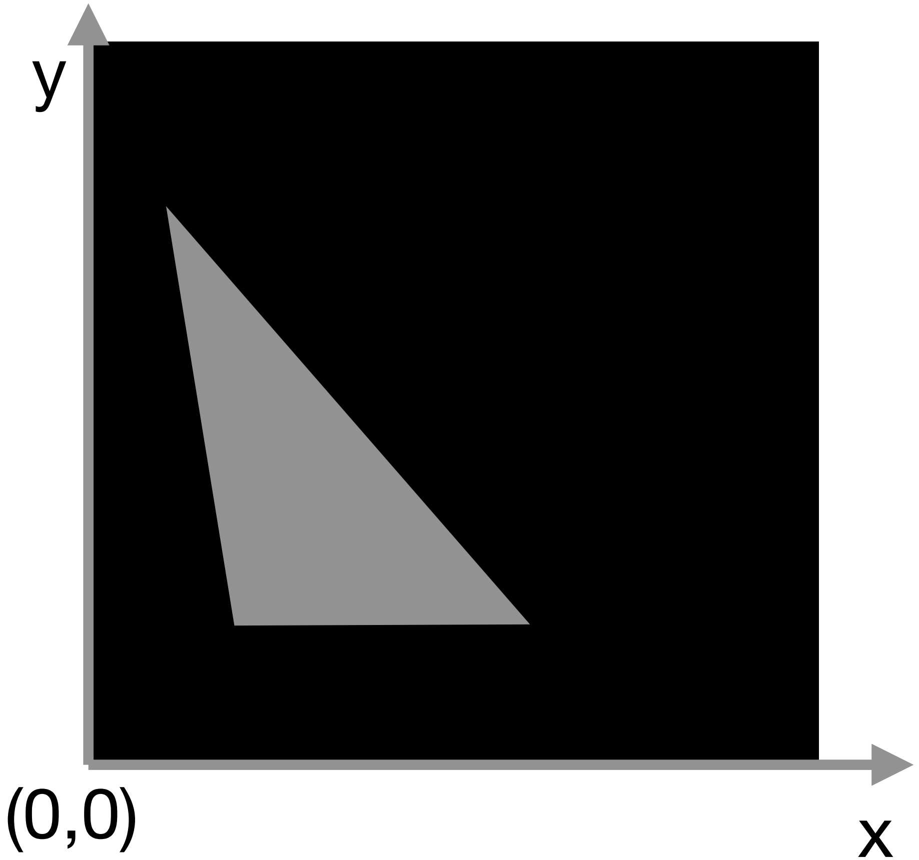


Image coordinates

We need to render it onto this.

Raster Images: Coordinate Systems



Model coordinates

We modeled our triangle like this.

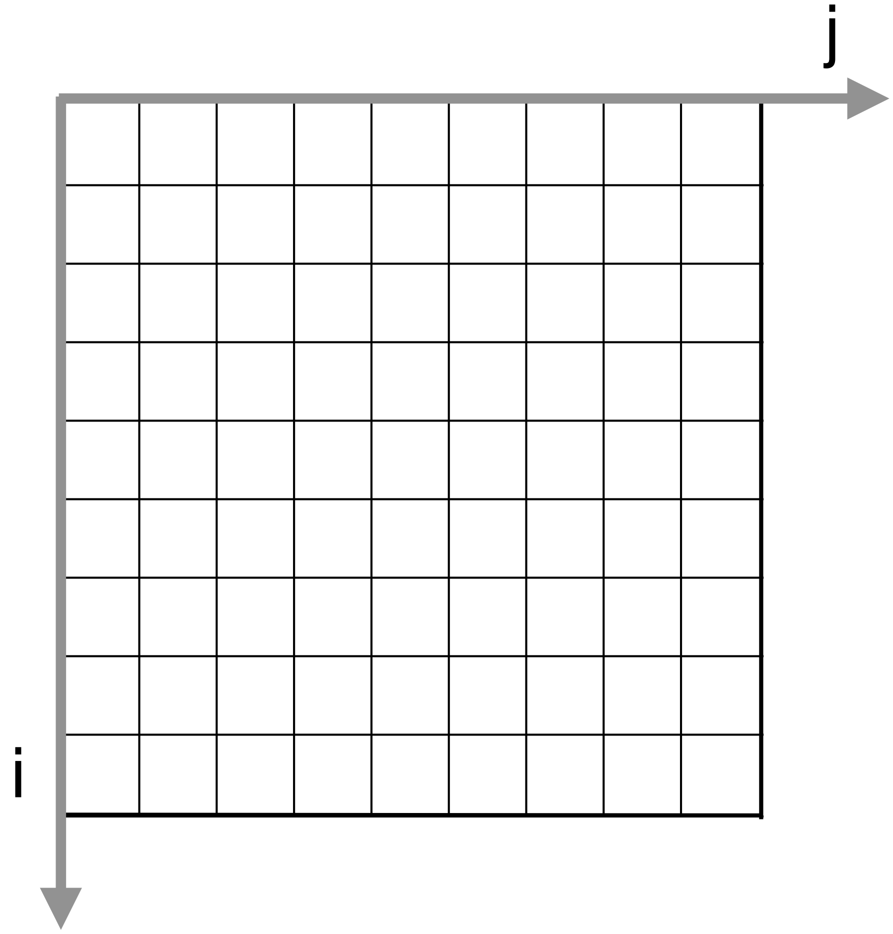
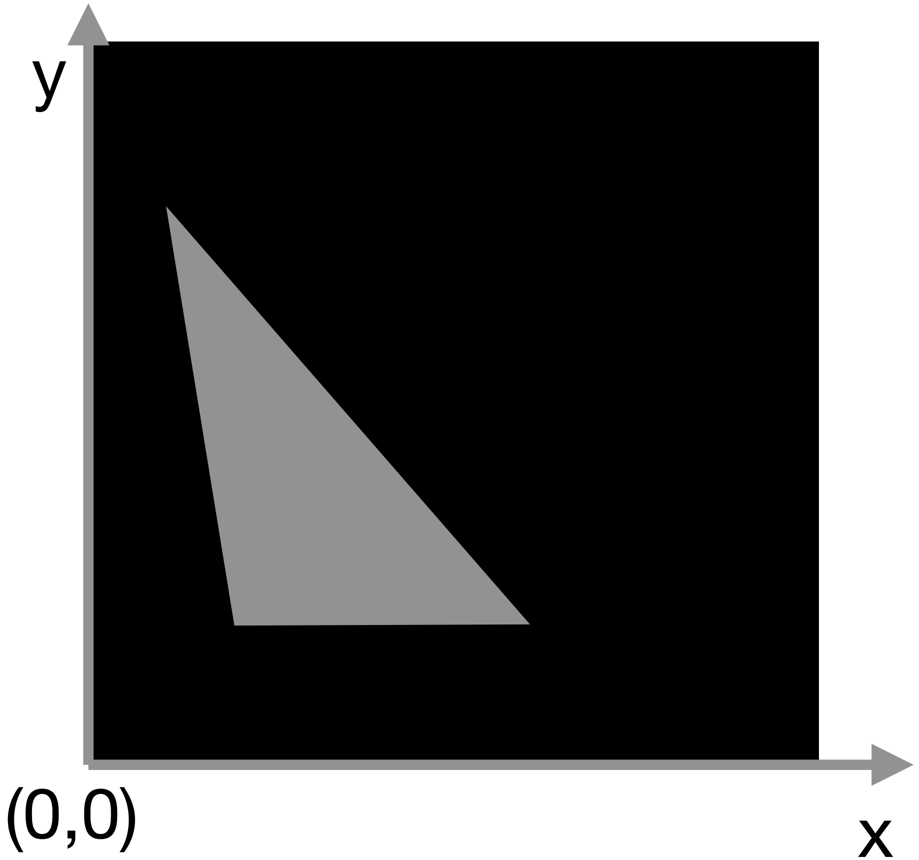


Image coordinates

We need to render it onto this.

Raster Images: Coordinate Systems



Model coordinates

We modeled our triangle like this.

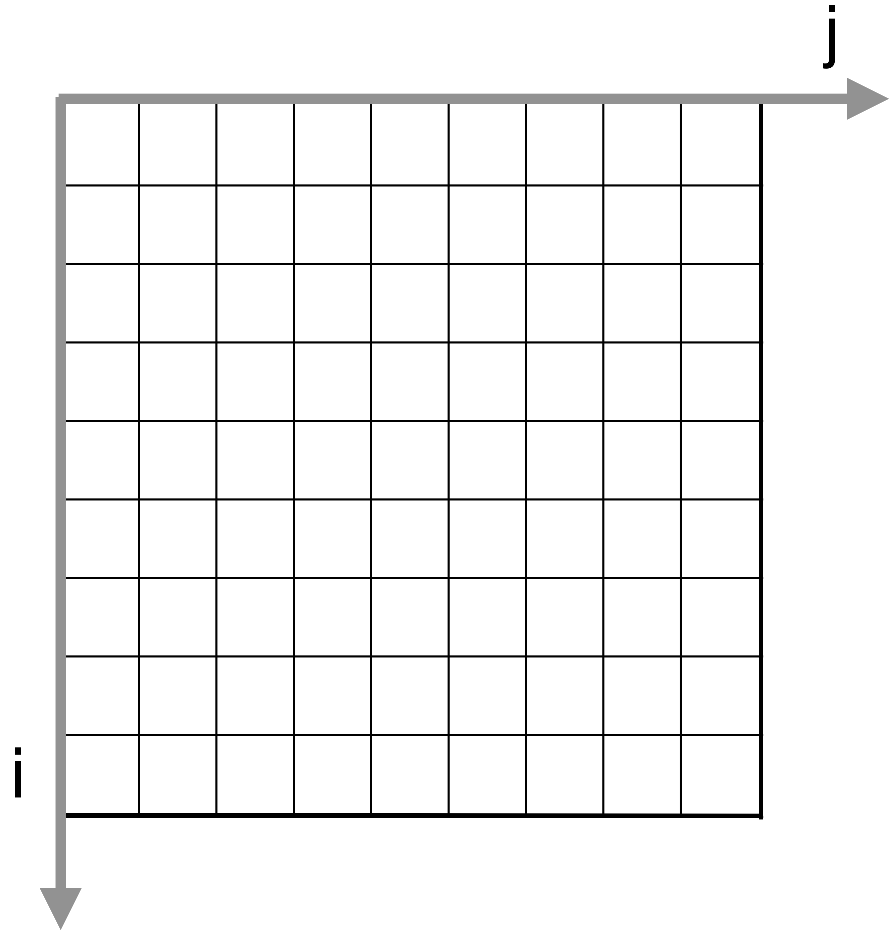
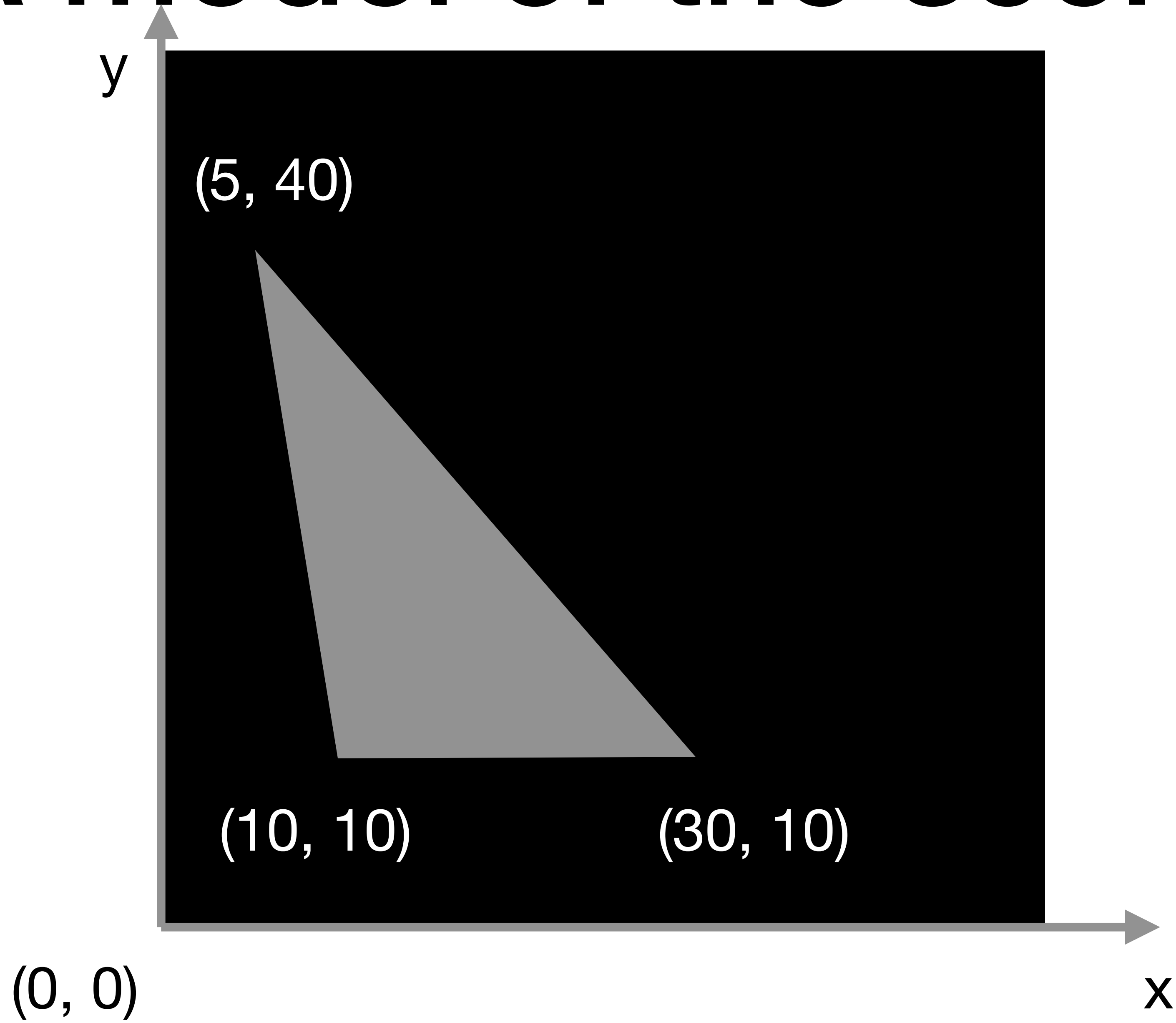


Image coordinates

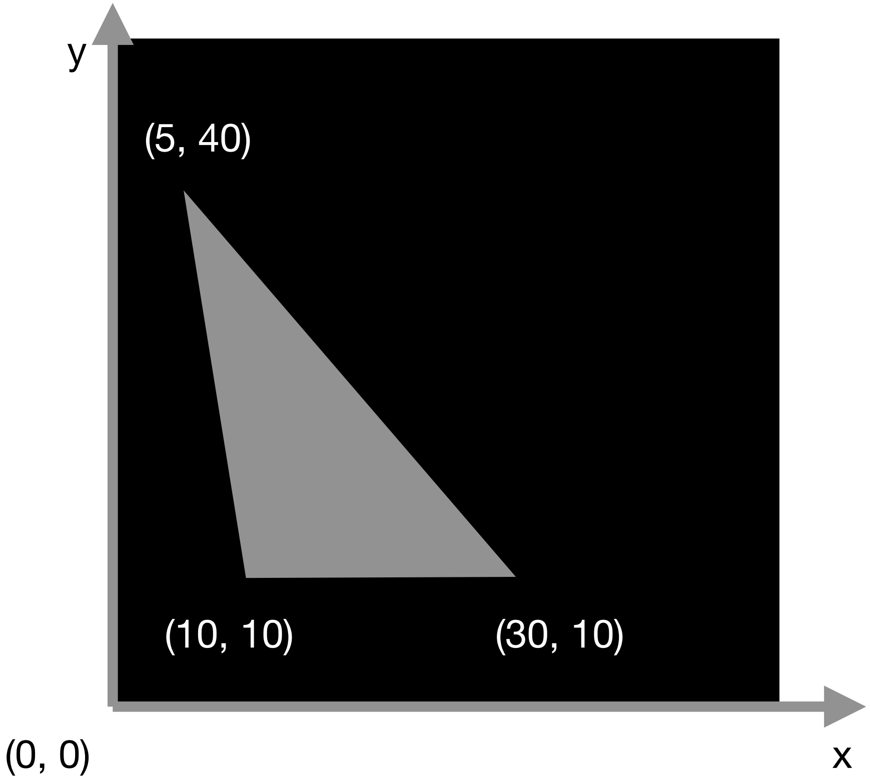
We need to render it onto this.

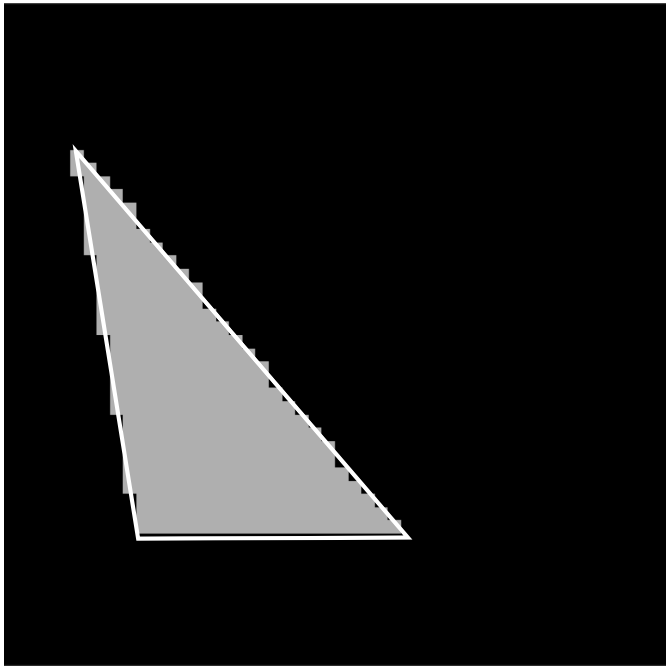
We need a coordinate transformation!

A model of the scene



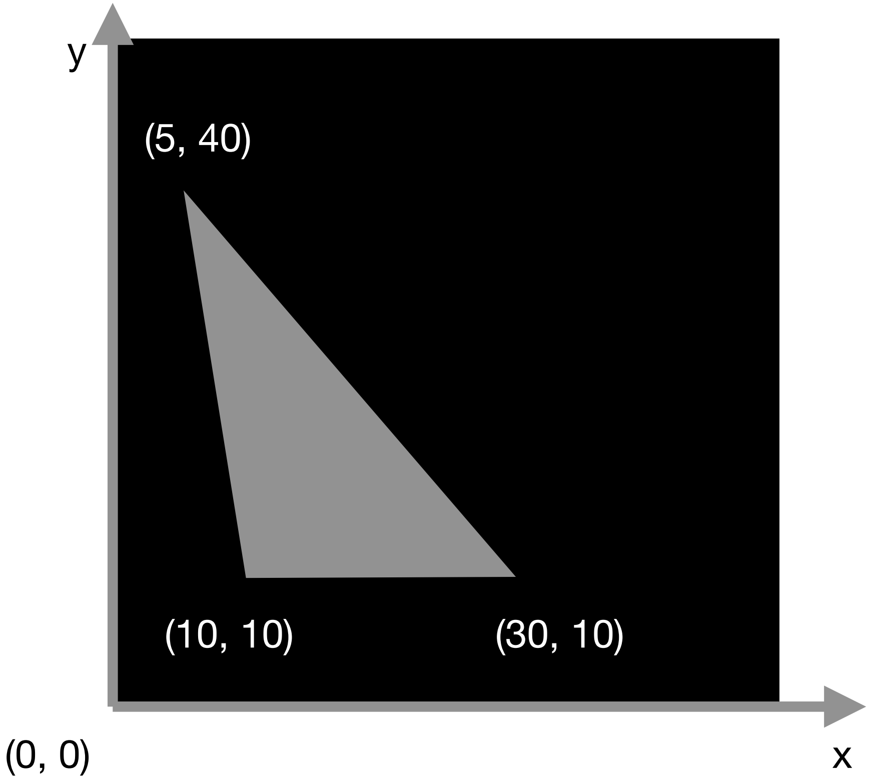
A0: Rendering (*Rasterizing*) a Triangle

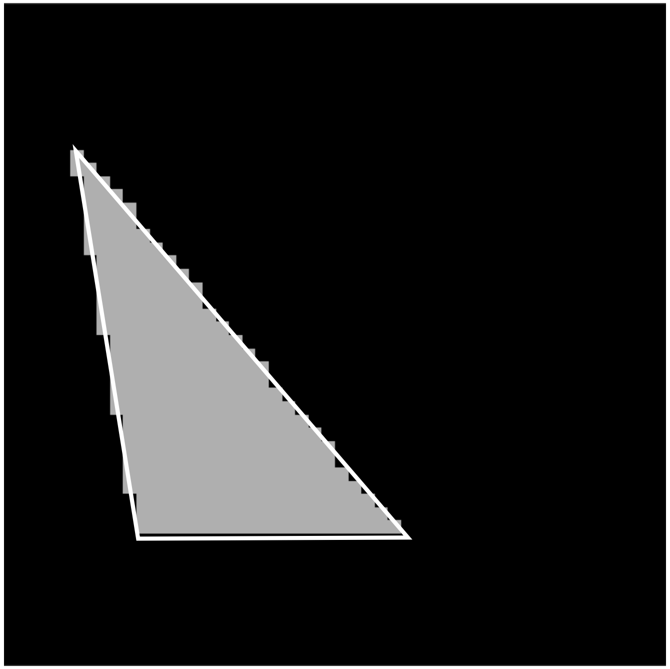




Pseudocode:

A0: Rendering (*Rasterizing*) a Triangle

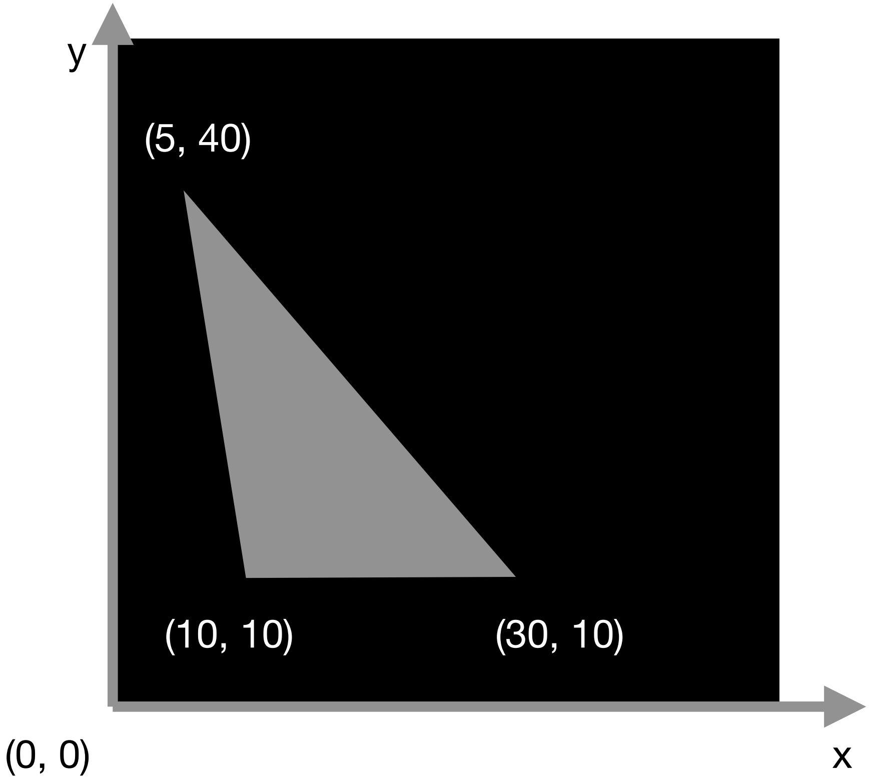


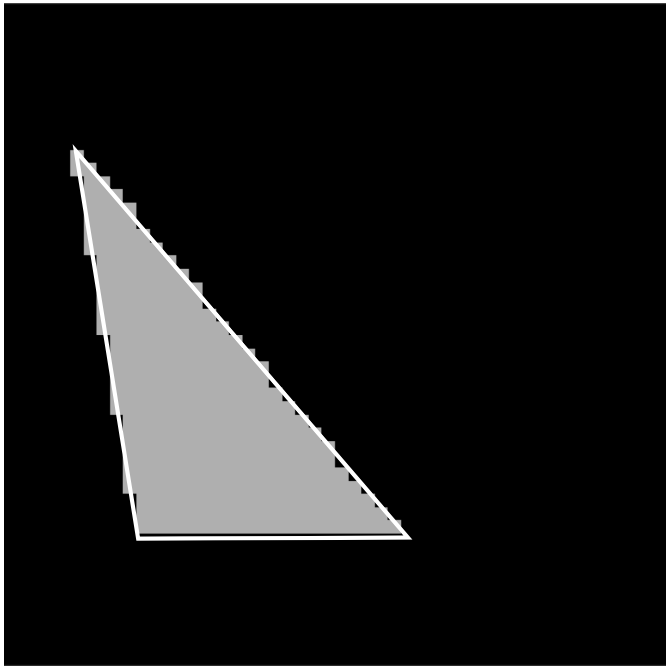


Pseudocode:

```
for each pixel p:
```

A0: Rendering (*Rasterizing*) a Triangle

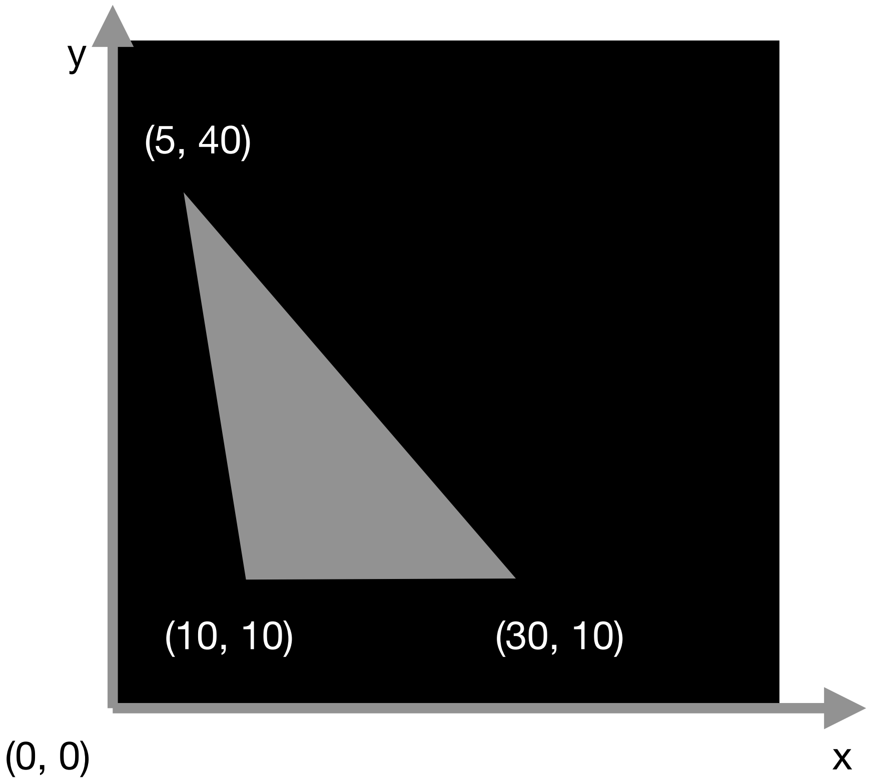


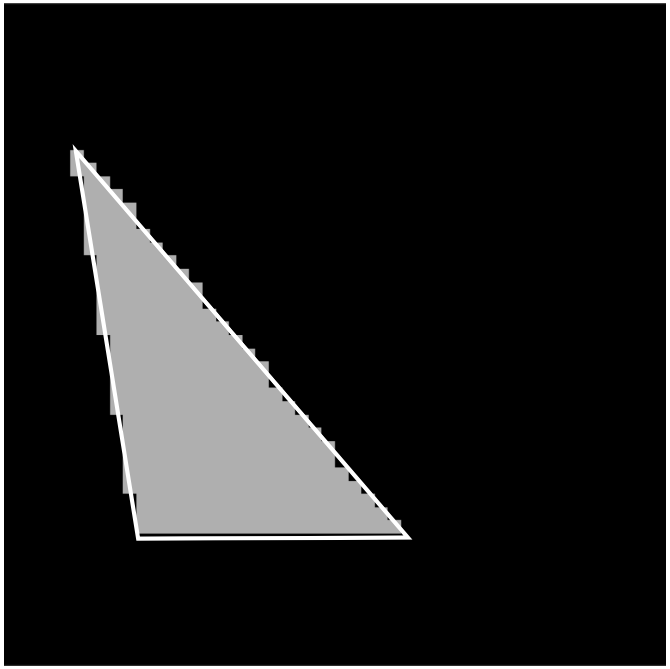


Pseudocode:

```
for each pixel p:
  if p is inside triangle:
```

A0: Rendering (*Rasterizing*) a Triangle





Pseudocode:

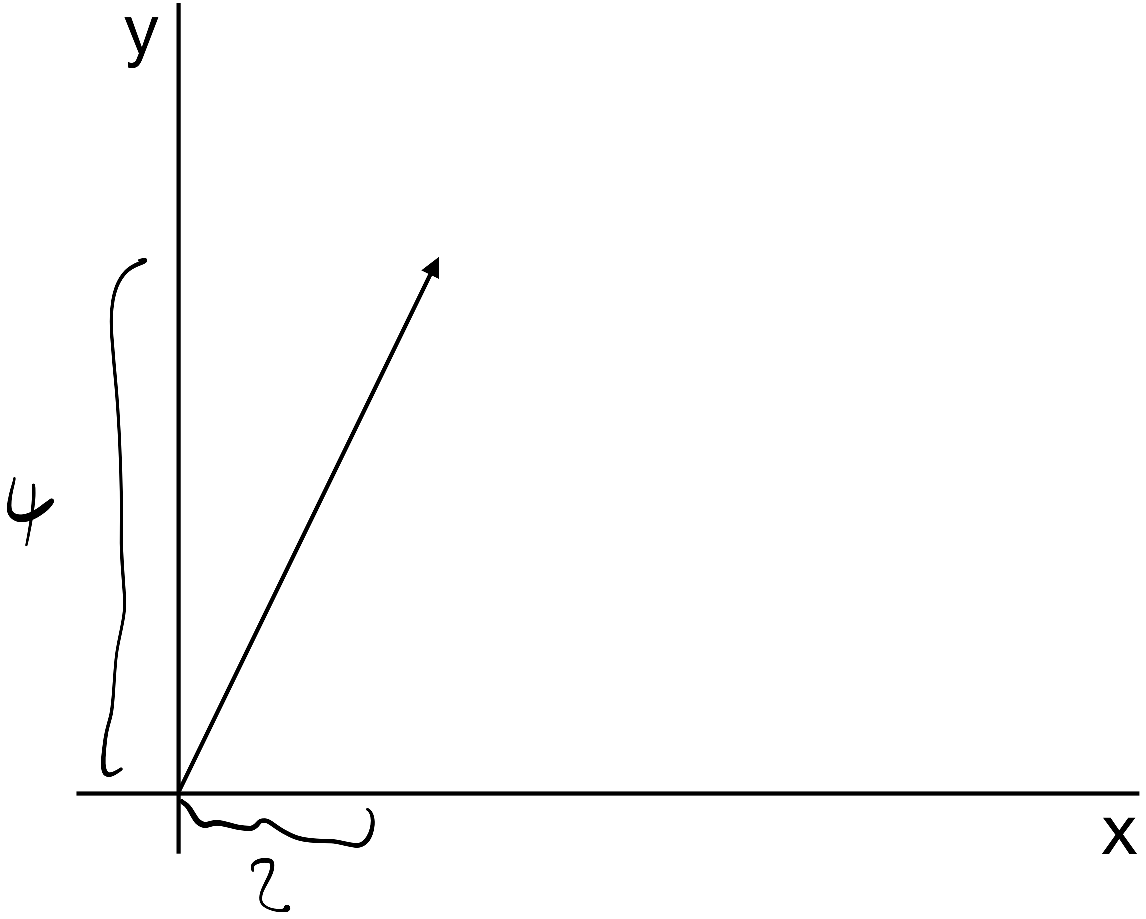
```
for each pixel p:
    if p is inside triangle:
        p = color
```

Images in Julia: Demo

- Draw a rectangle on a canvas
- Demo colors
- See image_demo.jl

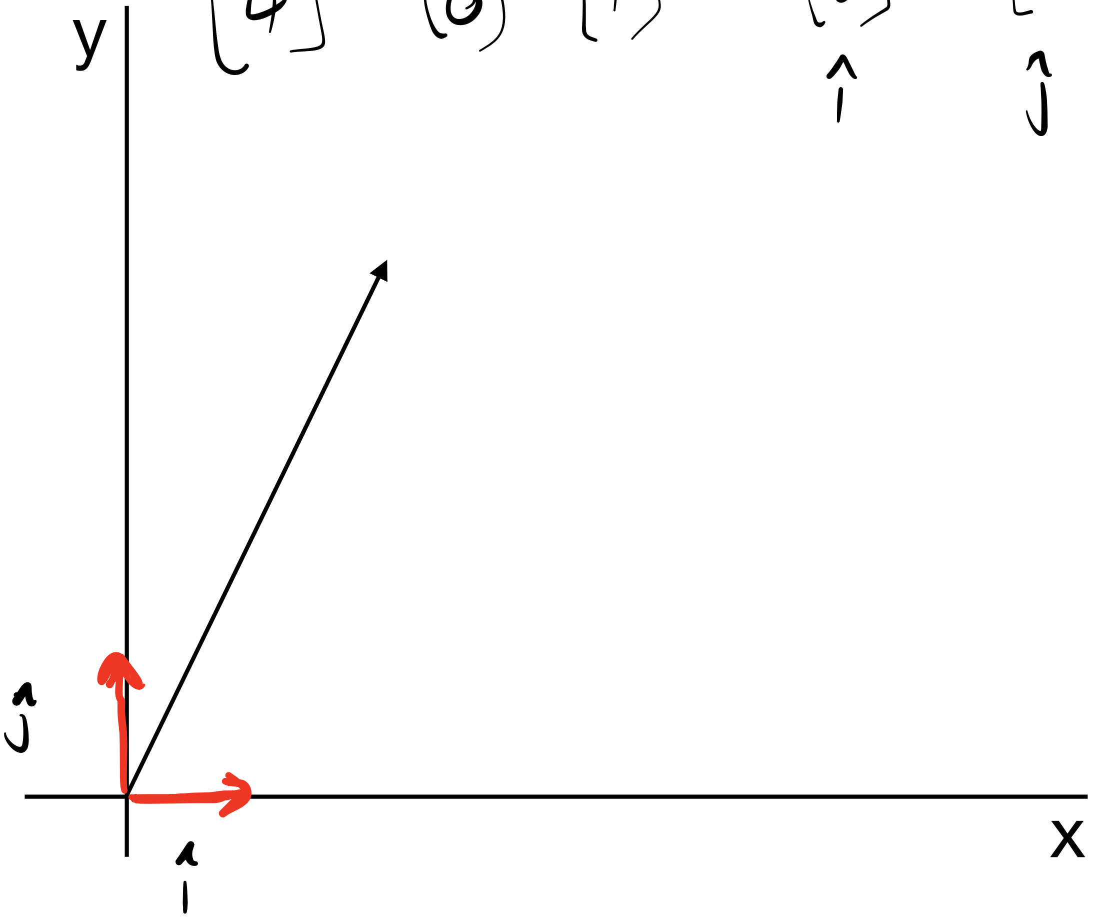
Vectors

$\begin{pmatrix}2\\4\end{pmatrix}$ or $(2,4)$



The Canonical Basis

$$\begin{bmatrix}2\\4\end{bmatrix} = \begin{bmatrix}2\\0\end{bmatrix} + \begin{bmatrix}0\\4\end{bmatrix} = 2\underset{\hat{\imath}}{\begin{bmatrix}1\\0\end{bmatrix}} + 4\underset{\hat{\jmath}}{\begin{bmatrix}0\\1\end{bmatrix}}$$

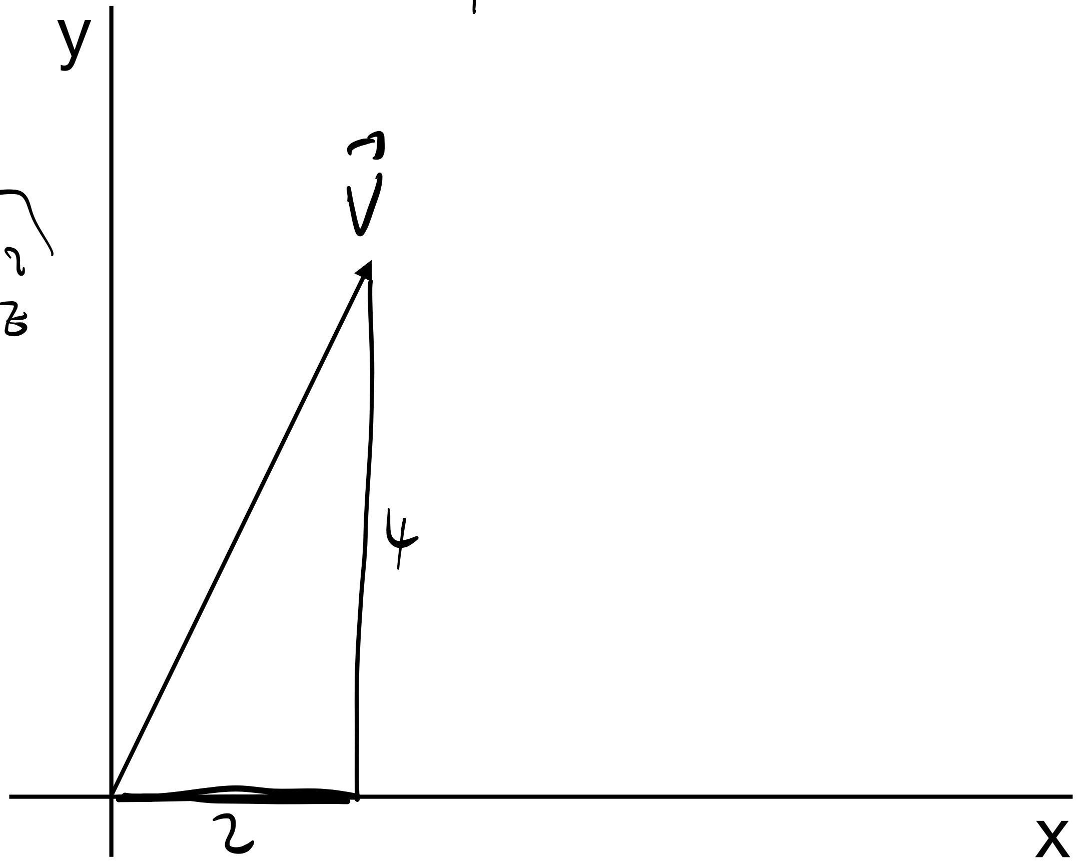


Magnitude (length)

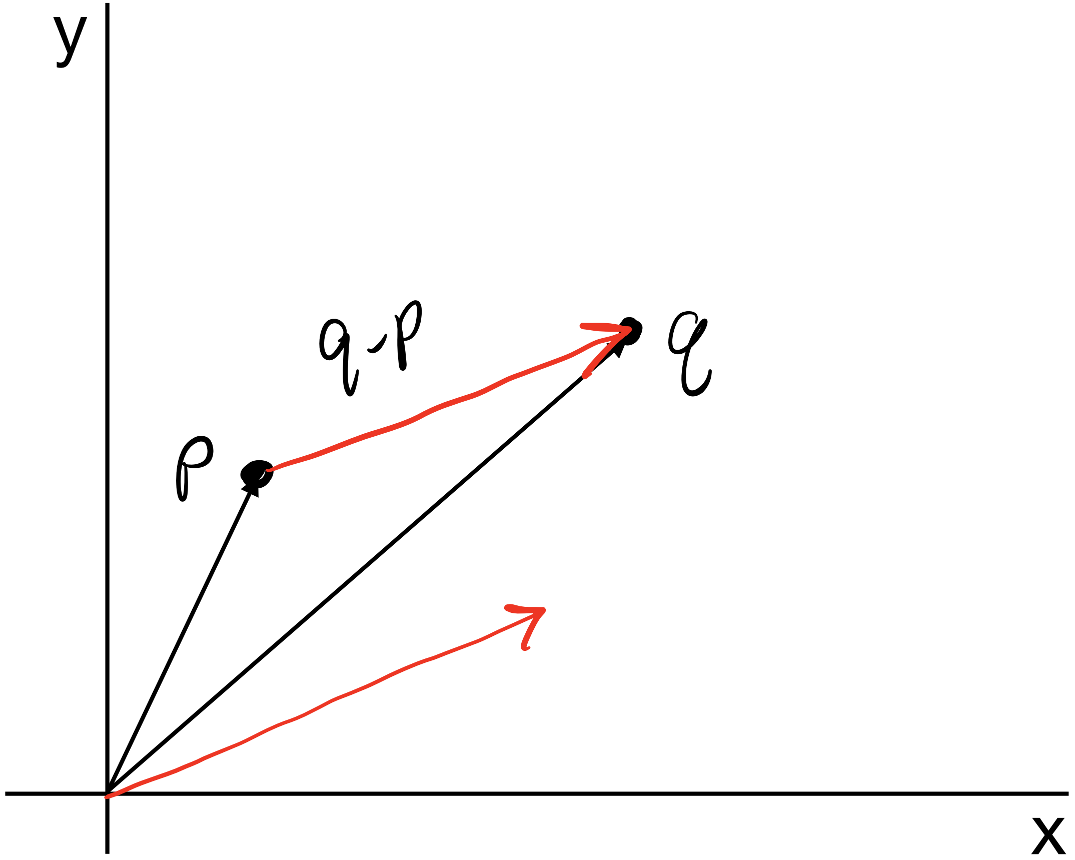
$$\|\vec{v}\| = \sqrt{v_x^2 + v_y^2}$$

3D

$$\left\|\begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix}\right\| = \sqrt{u_x^2 + u_y^2 + u_z^2}$$



The vector between two points



The dot product

One version of

$\vec{u}$ times $\vec{v}$

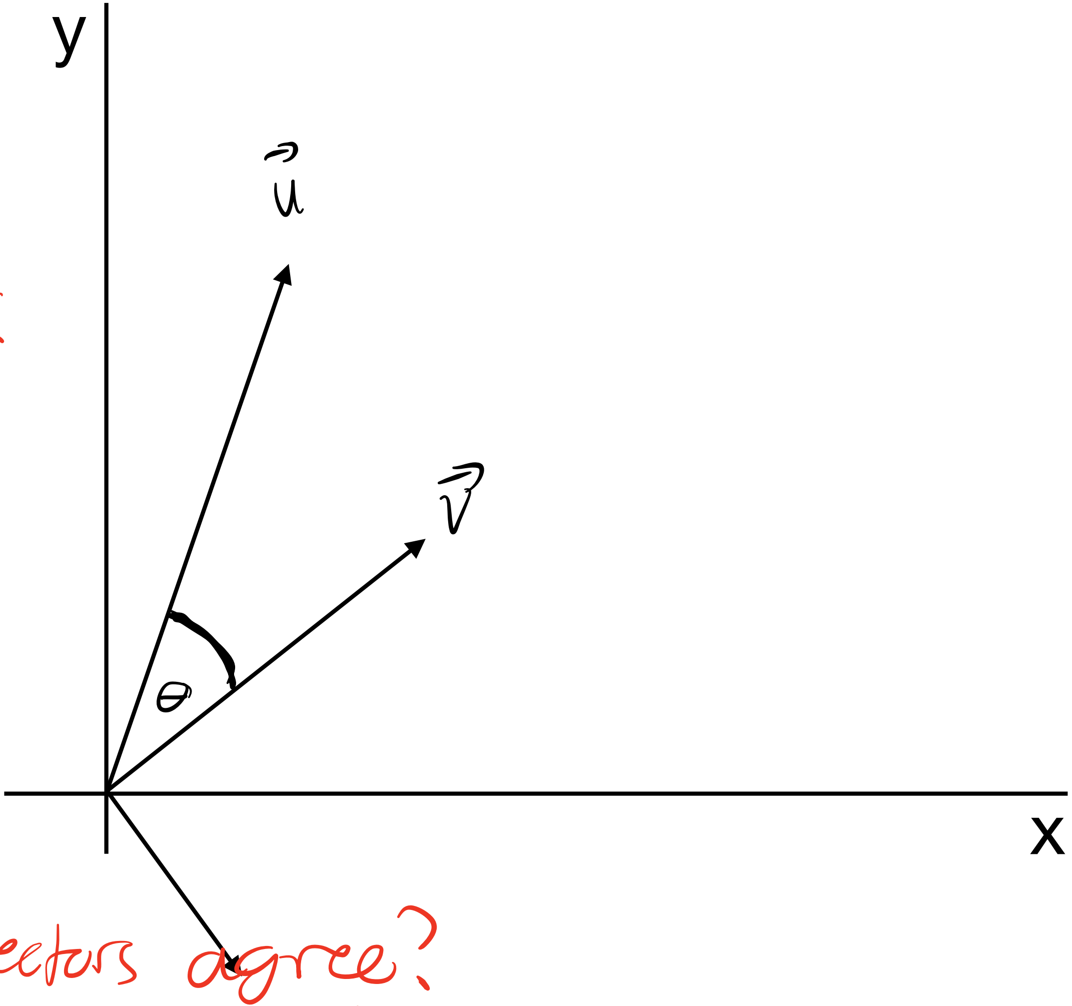
$\vec{u} = \begin{bmatrix} u_x \\ u_y \end{bmatrix}$ $\vec{v} = \begin{bmatrix} v_x \\ v_y \end{bmatrix}$

scalar

$$\vec{u} \cdot \vec{v} = \boxed{u_x v_x + u_y v_y}$$

$$= \vec{u}^T \vec{v}$$

$$= \|\vec{u}\| \|\vec{v}\| \cos\theta$$



"how much do these vectors agree?

The dot product

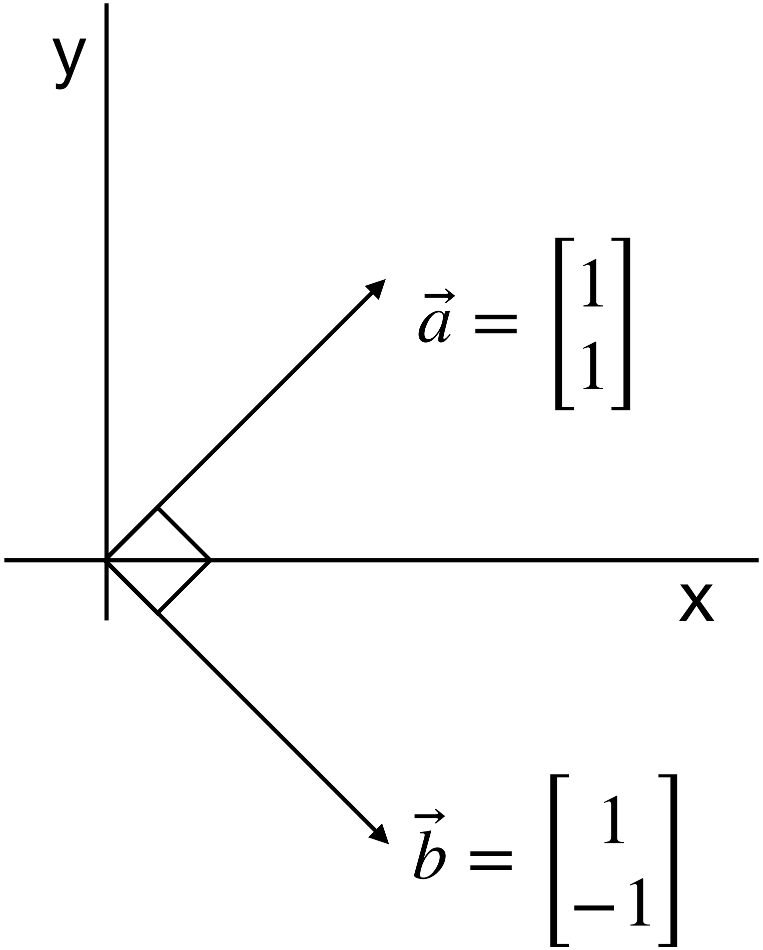
What is $\vec{a} \cdot \vec{b}$?

0

How did you know?

$\cos(90) = 0$

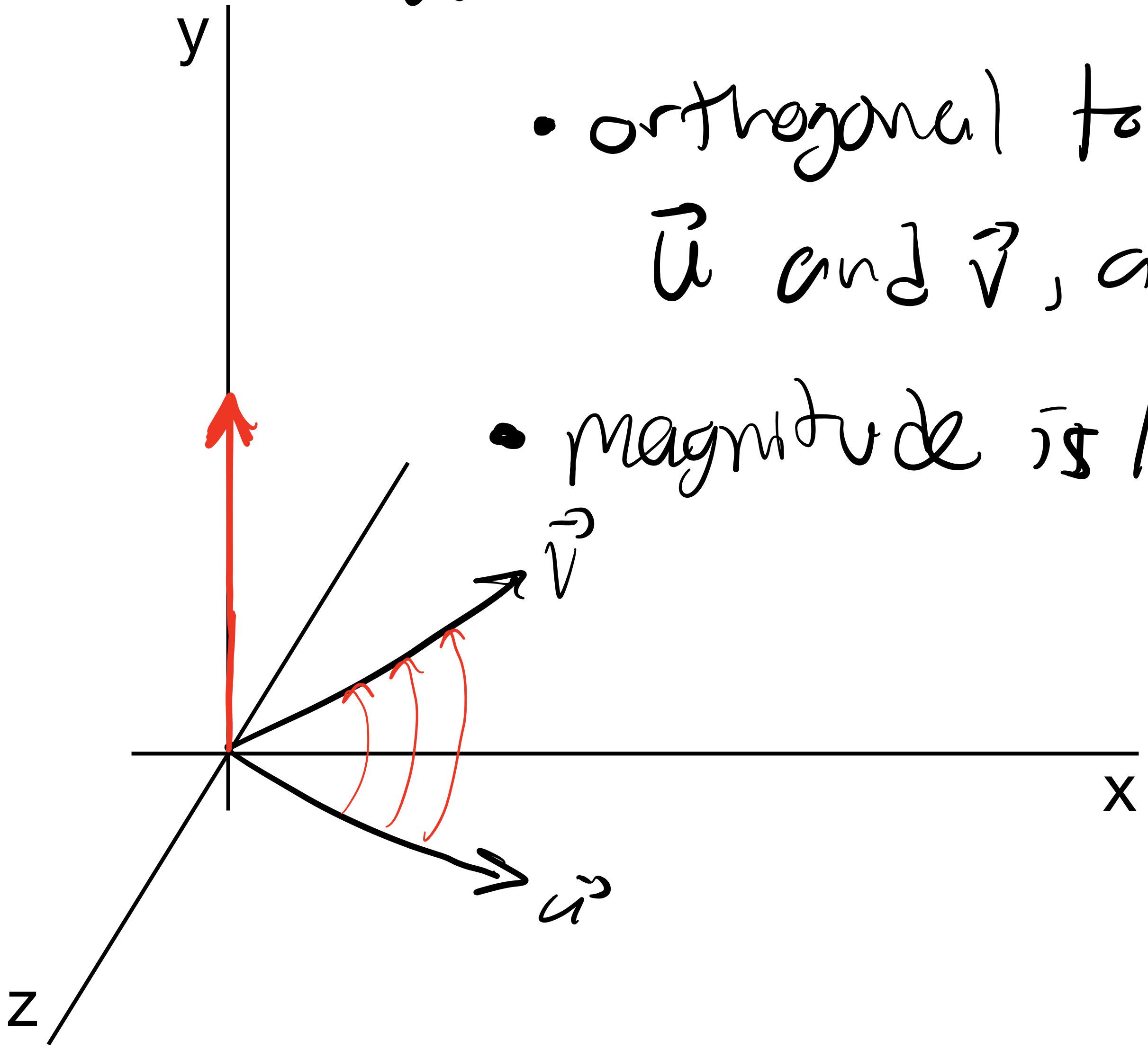
$1 \cdot 1 + 1 \cdot -1 = 0$


y
x
$\vec{a} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$
$\vec{b} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$

The cross product

$\vec{u} \times \vec{v}$ is

- orthogonal to both $\vec{u}$ and $\vec{v}$, and
- magnitude is $\|\vec{u}\|\|\vec{v}\|\sin\theta$



Point-in-Triangle