# Computer Graphics

Lecture 1

**Logistics; Images**

or: I ordered an image and all I got was this grid of colored boxes

# Announcements

- Assignments out:
  - HW0 due next Friday
  - A0 due Monday, 10/3
  - I think we'll have covered everything you need by Monday.
- Please bring your name card to every class (or leave them with me and I'll bring them).

# Syllabus/Logistics: Questions?

# Logistics / Syllabus: Key Points

# Logistics / Syllabus: Key Points

- Lectures occasionally flipped

# Logistics / Syllabus: Key Points

- Lectures occasionally flipped
- Book

# Logistics / Syllabus: Key Points

- Lectures occasionally flipped
- Book
- Slip days

# Logistics / Syllabus: Key Points

- Lectures occasionally flipped
- Book
- Slip days
- Math

# Logistics / Syllabus: Key Points

- Lectures occasionally flipped
- Book
- Slip days
- Math
- Julia, Javascript

# Logistics / Syllabus: Key Points

- Lectures occasionally flipped
- Book
- Slip days
- Math
- Julia, Javascript
- Feedback

# Logistics / Syllabus: Key Points

- Lectures occasionally flipped
- Book
- Slip days
- Math
- Julia, Javascript
- Feedback
- Q&A - Discord?

# Syllabus

## True or False?

# Syllabus

## True or False?

1. You have 3 slip days that can applied to extend any deadline.

# Syllabus

## True or False?

1. You have 3 slip days that can applied to extend any deadline.

2. You **don't** need to email me to use a slip day.

# Syllabus

## True or False?

1. You have 3 slip days that can applied to extend any deadline.

2. You **don't** need to email me to use a slip day.

3. The midterm is given in week 5 of the course.

# Syllabus

## True or False?

1. You have 3 slip days that can applied to extend any deadline.

2. You **don't** need to email me to use a slip day.

3. The midterm is given in week 5 of the course.

4. Lecture slides, videos, etc. are posted on Canvas.

# Goals (meta)

- A slide like this will (should) appear at the beginning of each lecture.
- This is my way of conveying what I expect you to be able to understand or do after the class.
- In aggregate, these goals form a study guide.

# Goals

- Understand how images are represented mathematically (as a function) and computationally (as a 2D array of values sampled from that function)
- Know the different ways to represent raster images on a computer, including color.
- Know how 2D arrays are indexed in Julia

# How do we graphics?

Let's design a simple graphics system.

The goal: draw a triangle on the screen.

# How do we graphics?

Let's design a simple graphics system.

The goal: draw a triangle on the screen.

(why a triangle? more on this next time...)

# How do we graphics?

Let's design a simple graphics system.

The goal: draw a triangle on the screen.

(why a triangle? more on this next time...)

Pseudocode for graphics:

- Create a model of a scene
- Render an image of the scene

# How do we graphics?

Let's design a simple graphics system.

The goal: draw a triangle on the screen.

(why a triangle? more on this next time...)

Pseudocode for graphics:

- Create a model of a scene *"Represent" the triangle*
- Render an image of the scene

# How do we graphics?

Let's design a simple graphics system.

The goal: draw a triangle on the screen.

(why a triangle? more on this next time...)

Pseudocode for graphics:

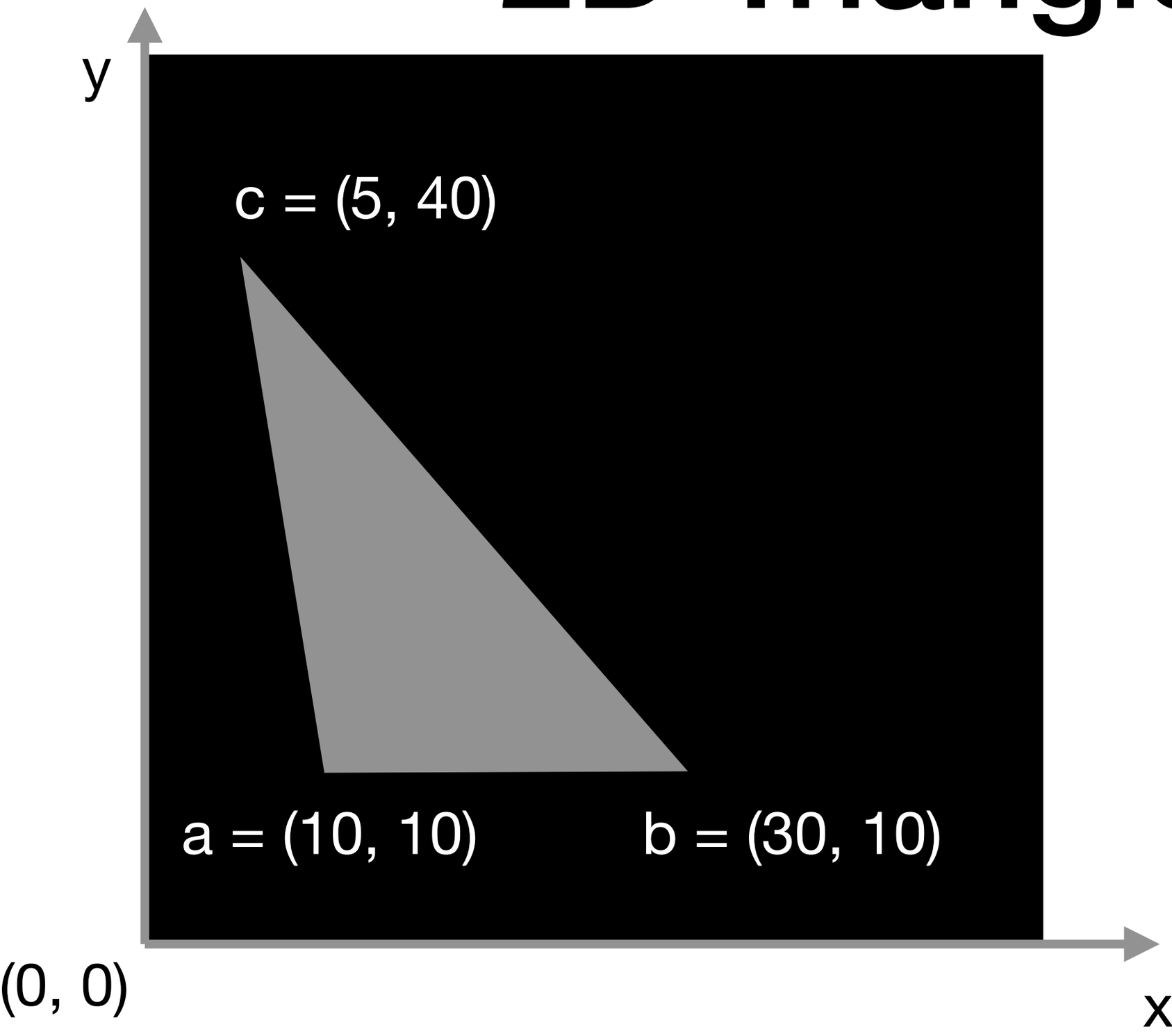
- Create a model of a scene

*"Represent" the triangle*

- Render an image of the scene

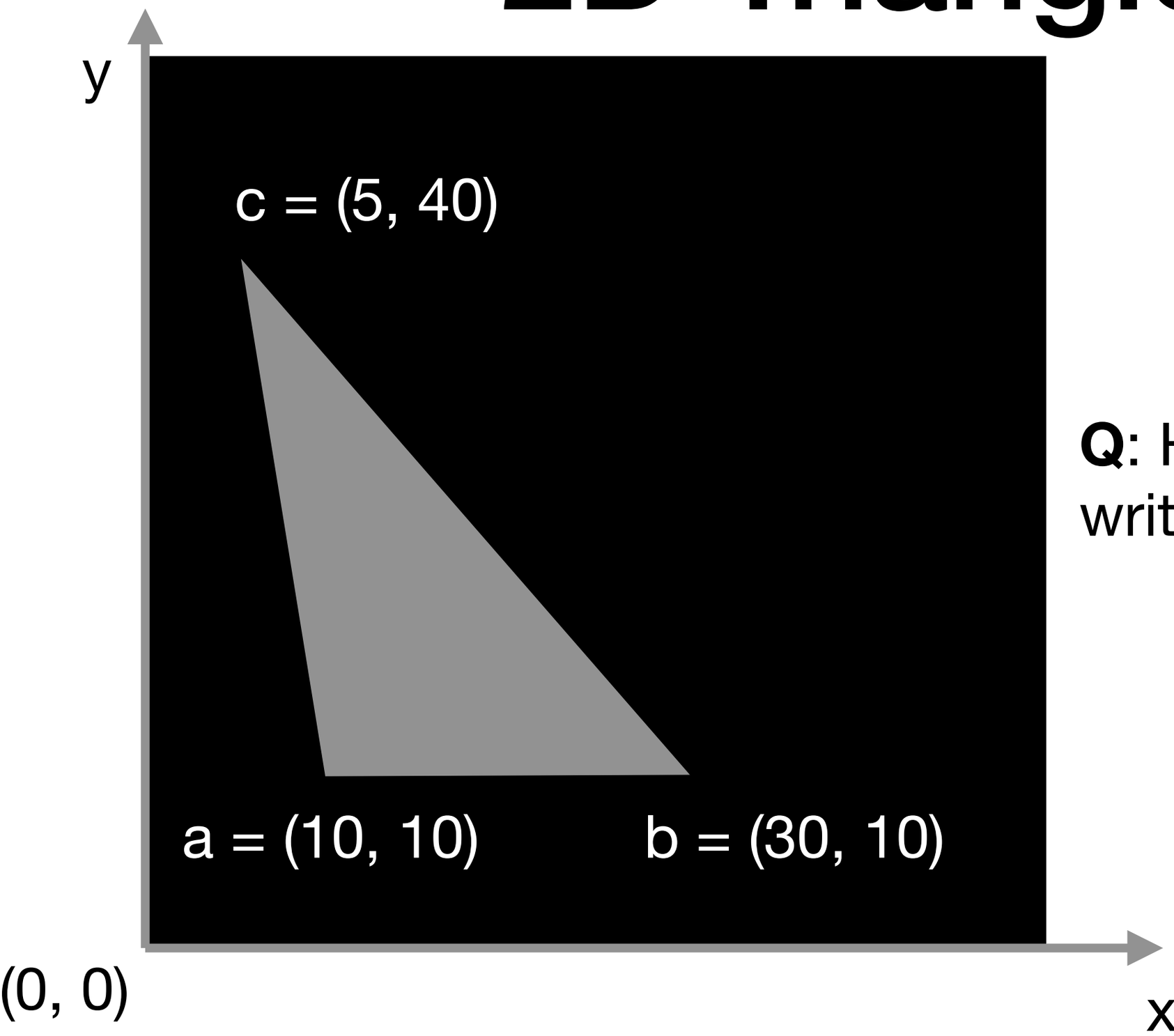
*Turn on pixels inside the triangle*

# 2D Triangles




Convention: list vertices in **counterclockwise** order.

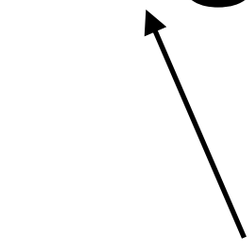
# 2D Triangles




**Q**: How many ways can I write down this triangle?

Convention: list vertices in **counterclockwise** order.

# Render an image of the model



what **is** that?

# Render an image of the model

What **is** an image anyway?

- A photographic print?
- A photographic negative?
- The screen you're watching this on?
- Some numbers in RAM?

# What is an image?

At its most formal and general: a **function** that maps *positions* in 2D to *distributions of radiant energy*

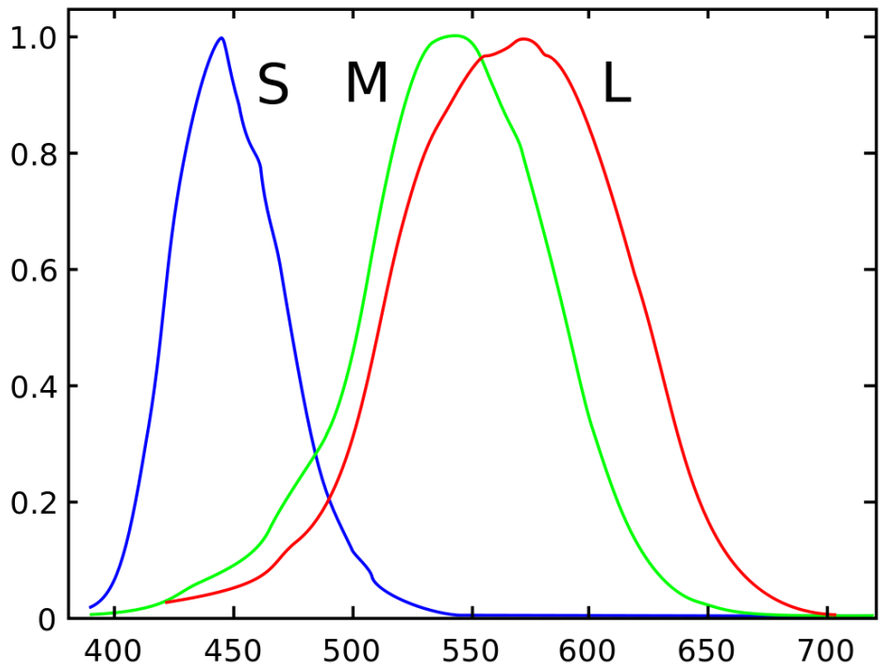
# What is an image?

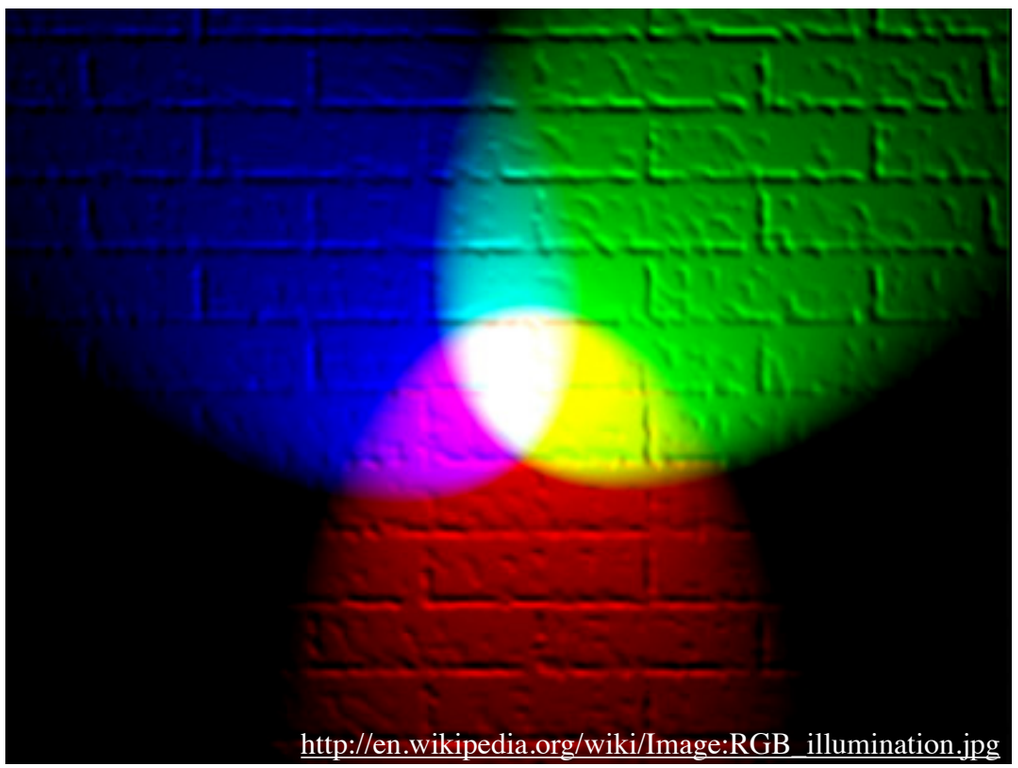
At its most formal and general: a **function** that maps *positions* in 2D to *distributions of radiant energy*

$$I : \mathbb{R}^2 \Rightarrow \; ??$$

# What about color?

- Humans are trichromatic, so we usually represent color as combinations of red, green, and blue

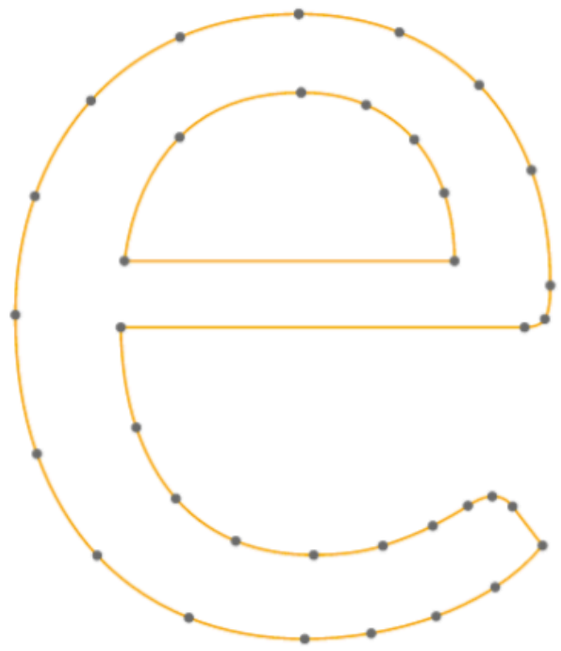


# How do we represent images?

- Raster formats - a 2D array of numbers
- Vector formats - mathematical description

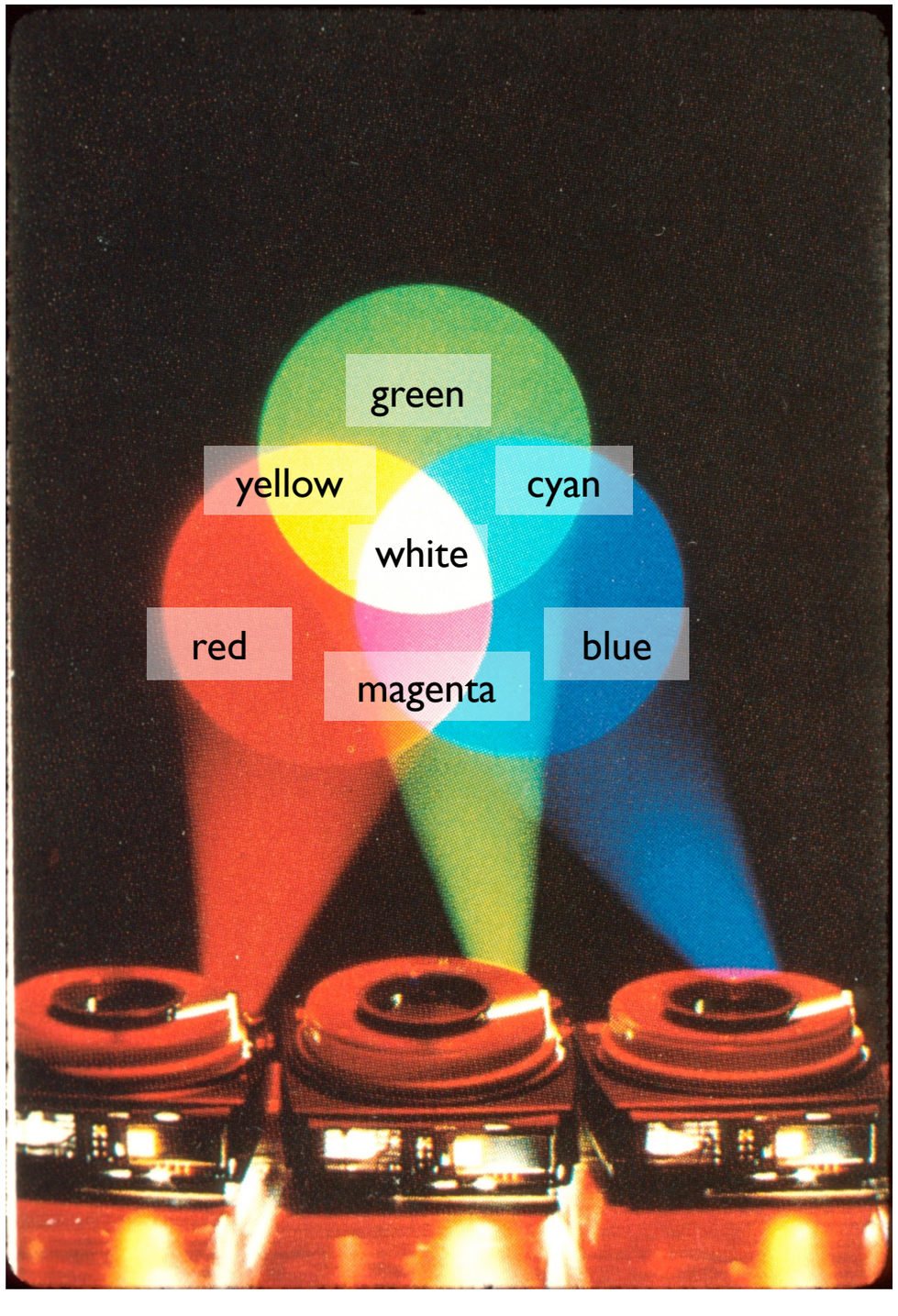
Raster Image



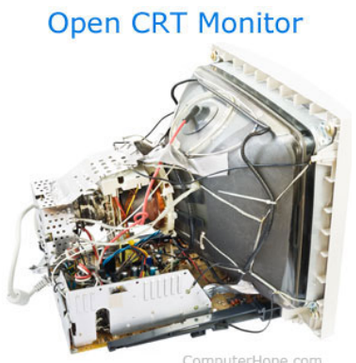

Vector Image

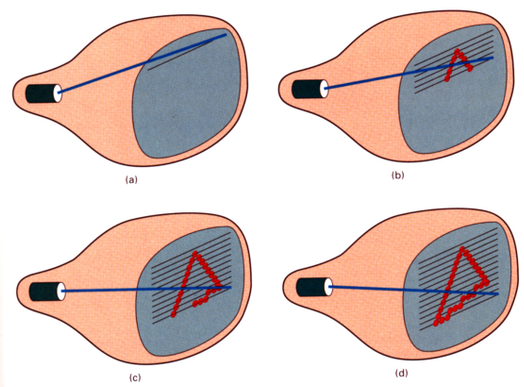
# How do we display images?
# Old School Edition

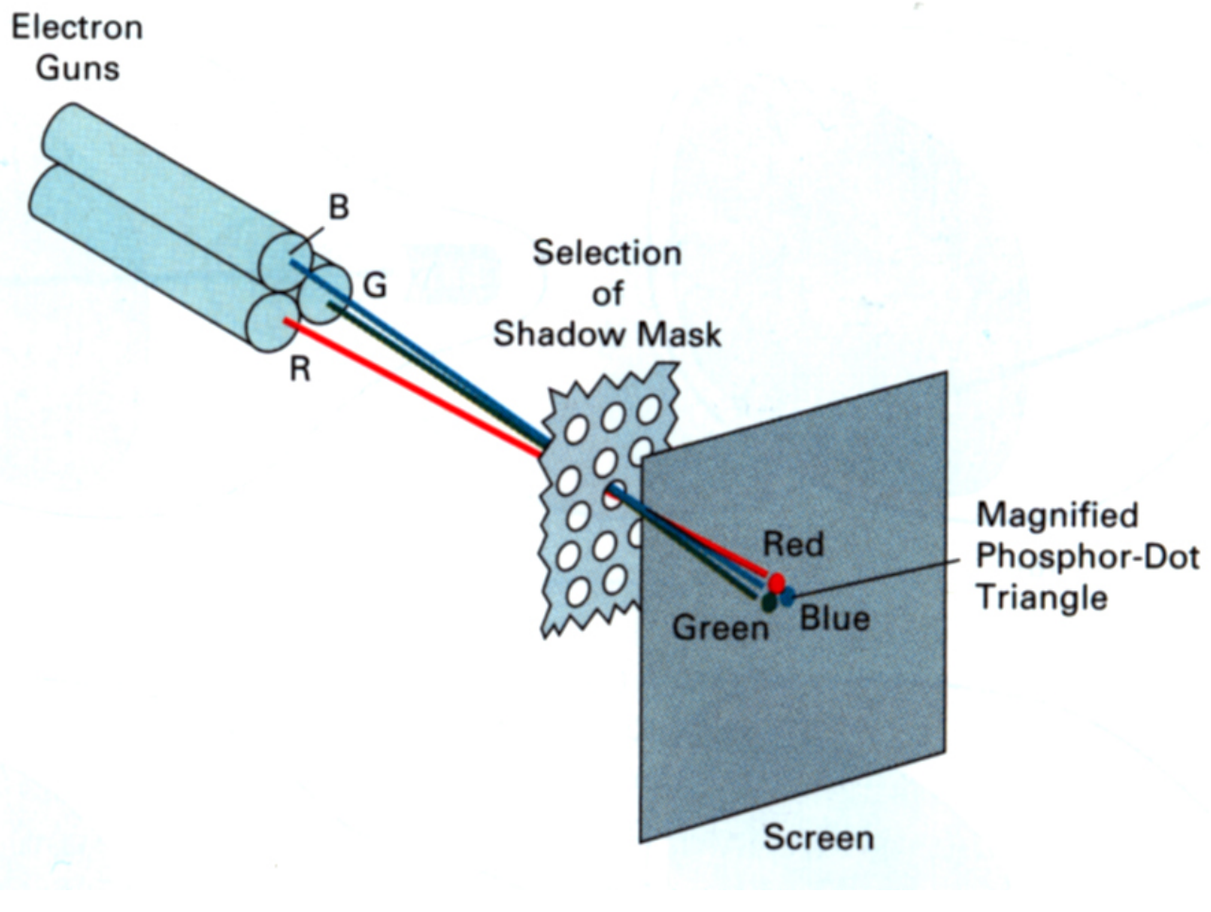
## Color Projector




## Cathode Ray Tube

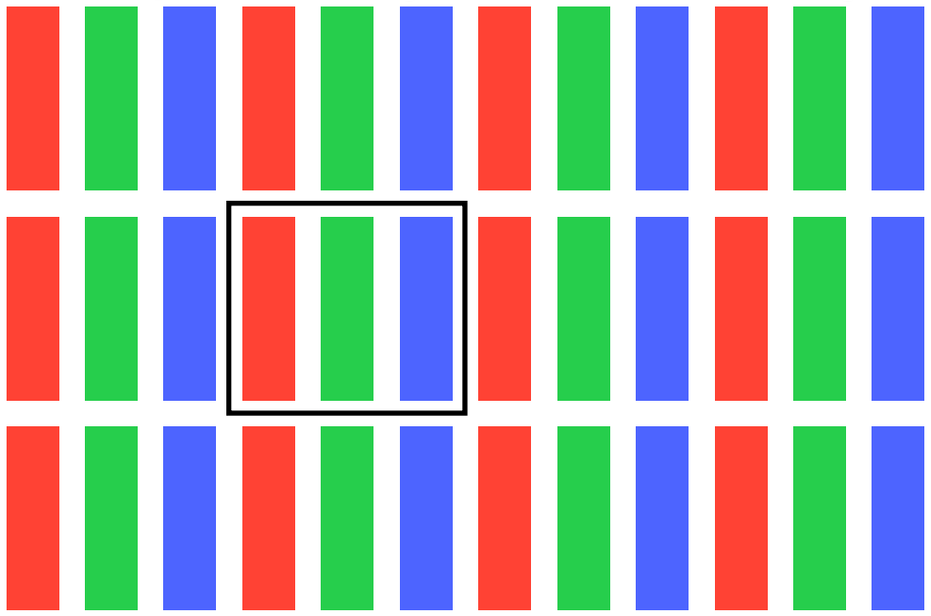


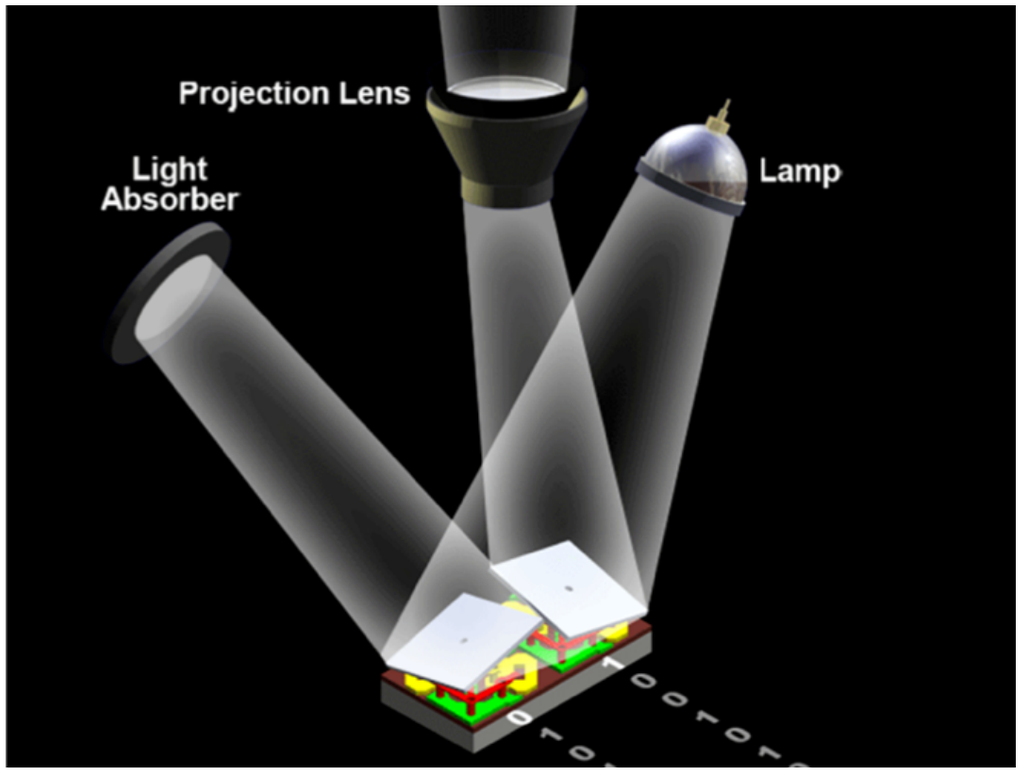



# How do we display images?
# Nowadays Edition

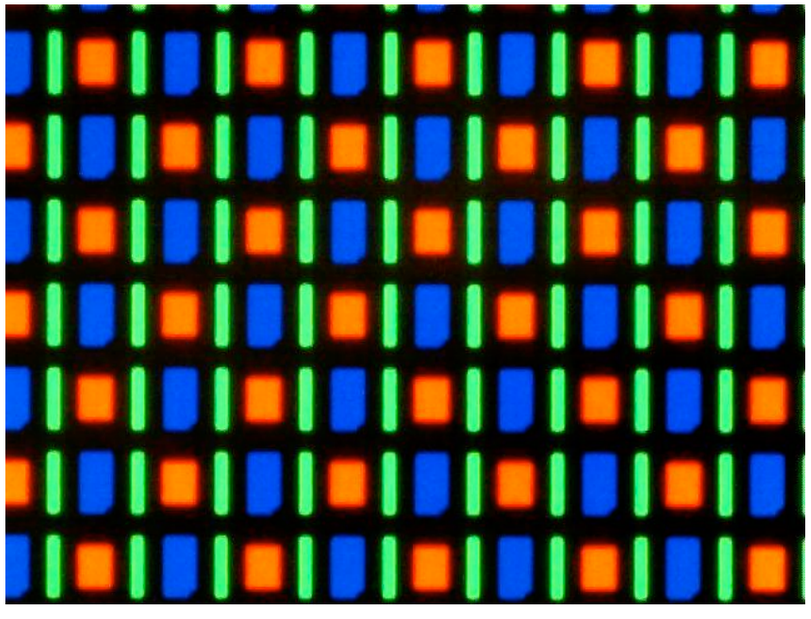
## Liquid Crystal Display



## Digital Light Processing



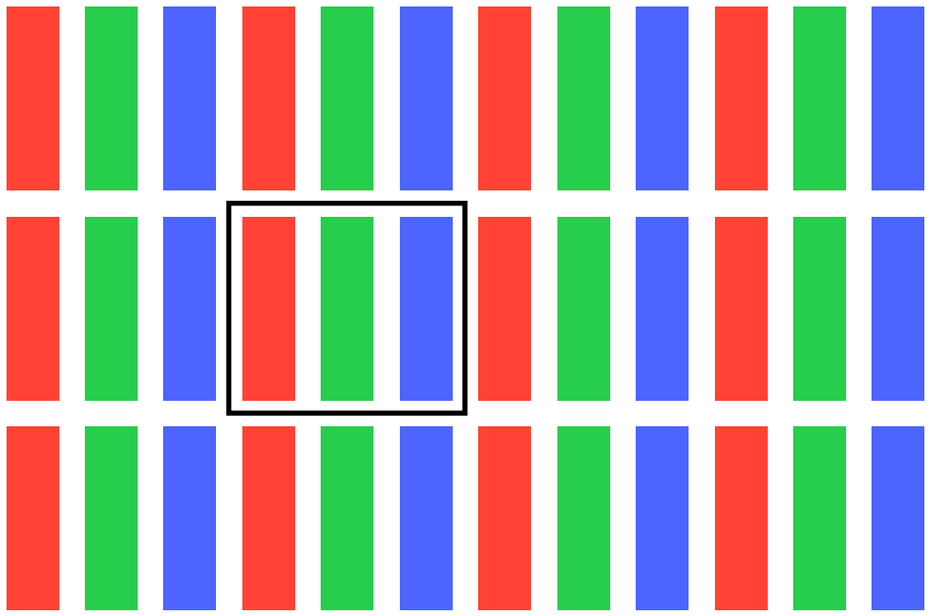

## Light Emitting Diode Display



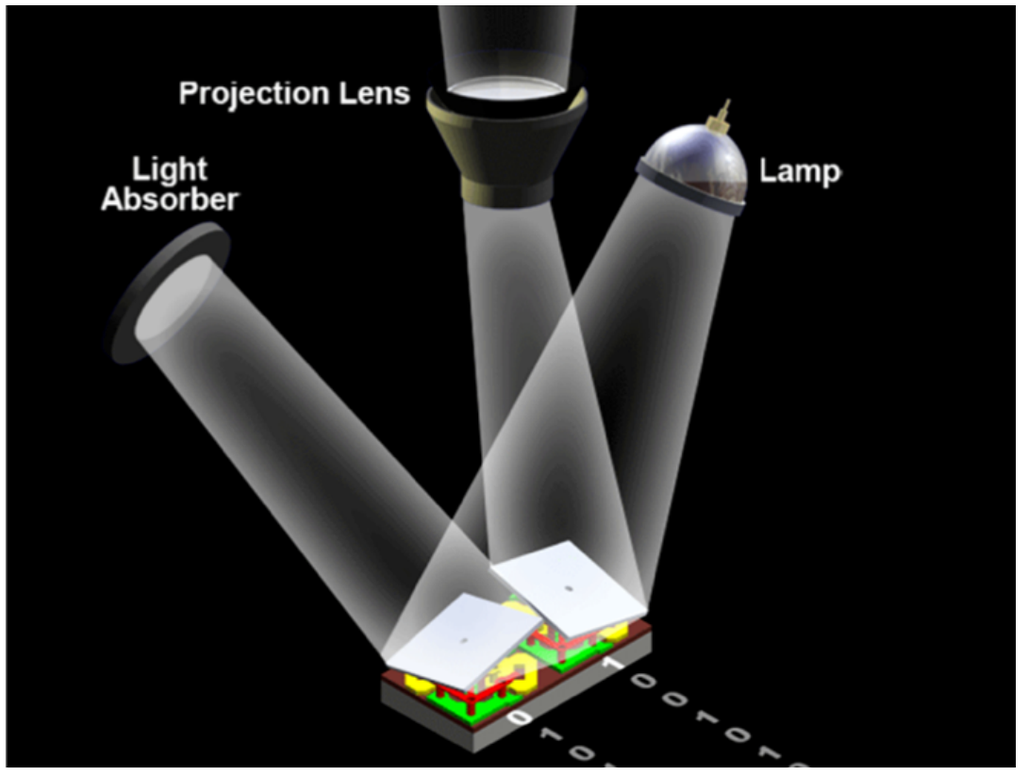
[Wikimedia Commons]

# How do we display images?
# Nowadays Edition

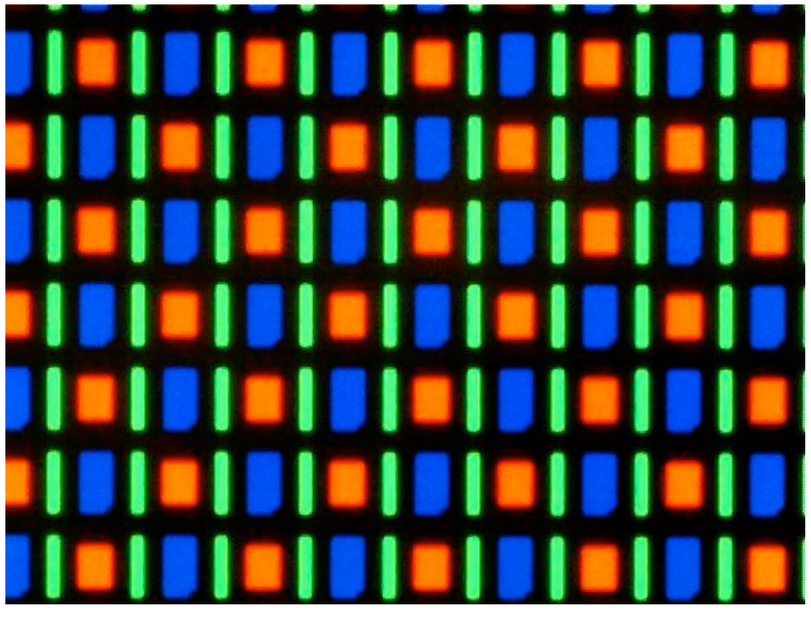
Liquid Crystal Display



Digital Light Processing



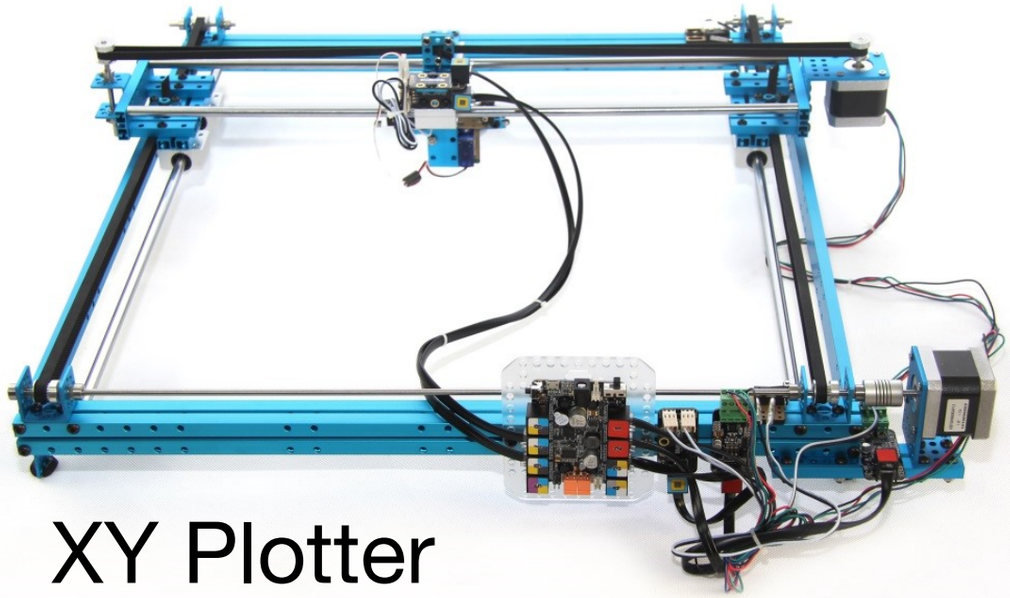

Light Emitting Diode Display



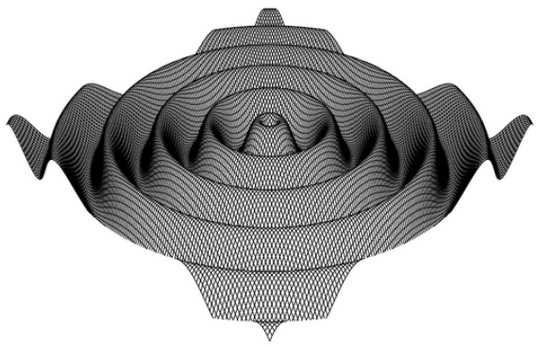
[Wikimedia Commons]

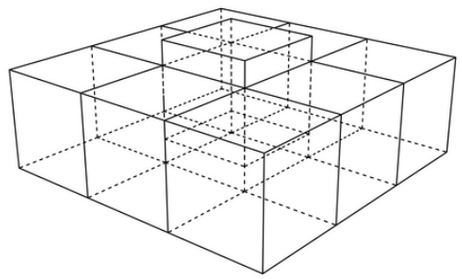
these are all examples of **raster displays**

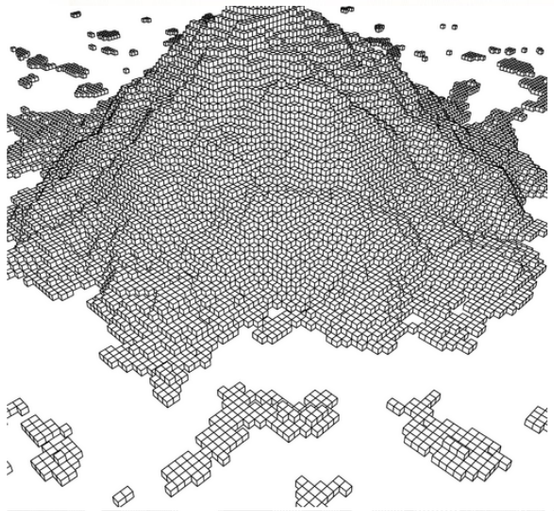
# Aside: It doesn't have to be this way...

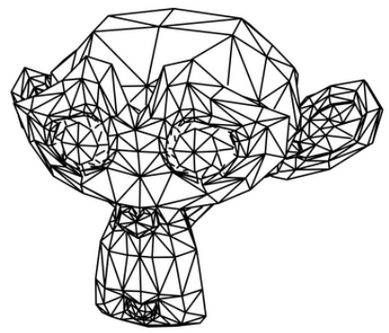


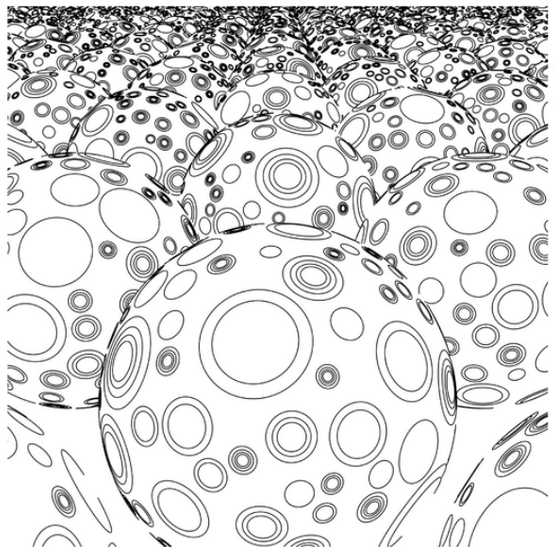
XY Plotter

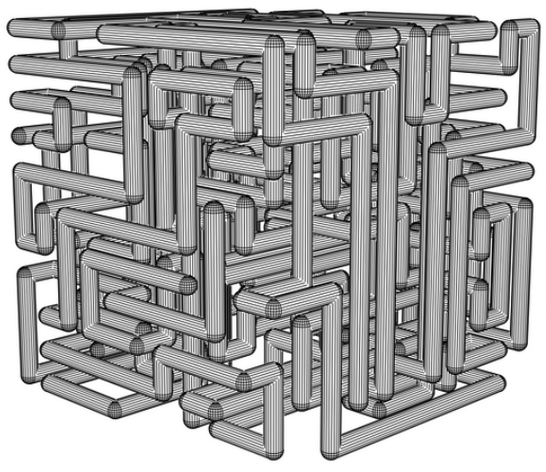


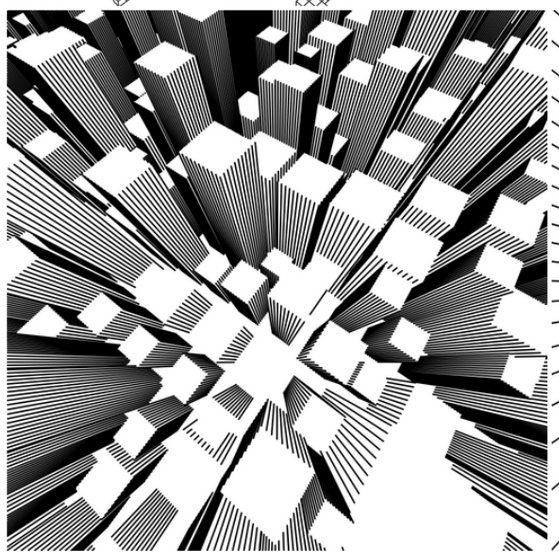


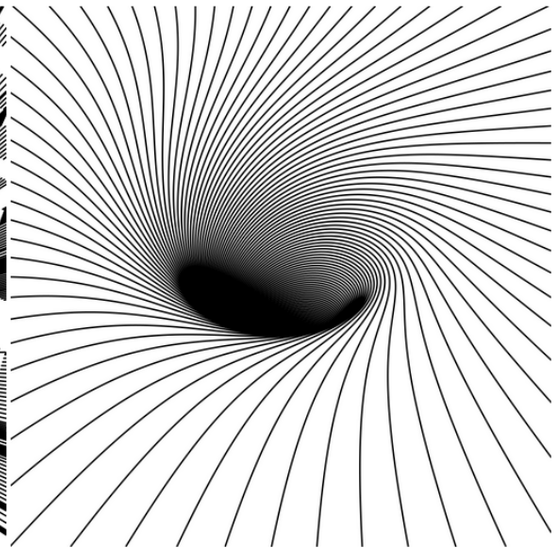










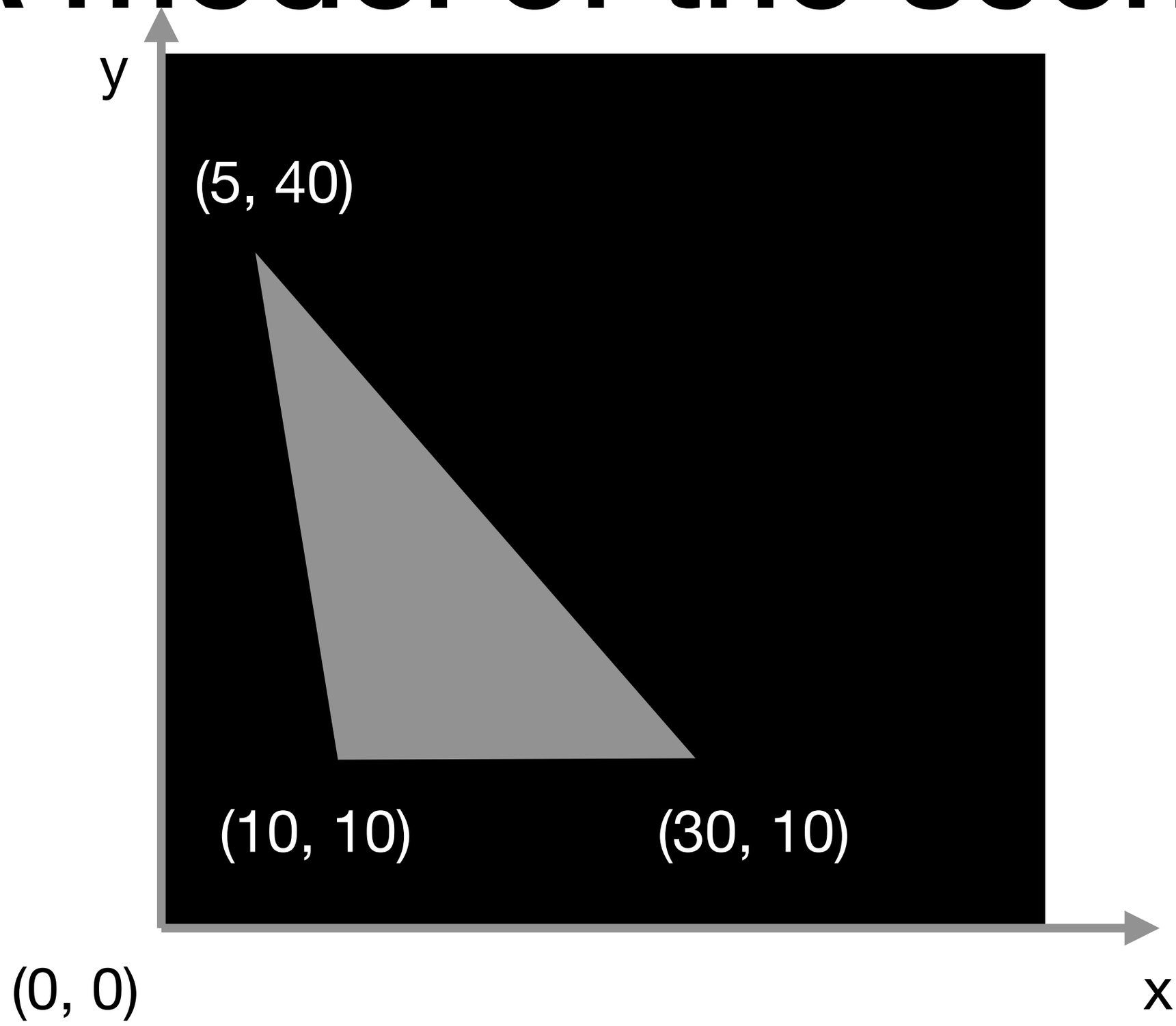


images: https://github.com/fogleman/ln

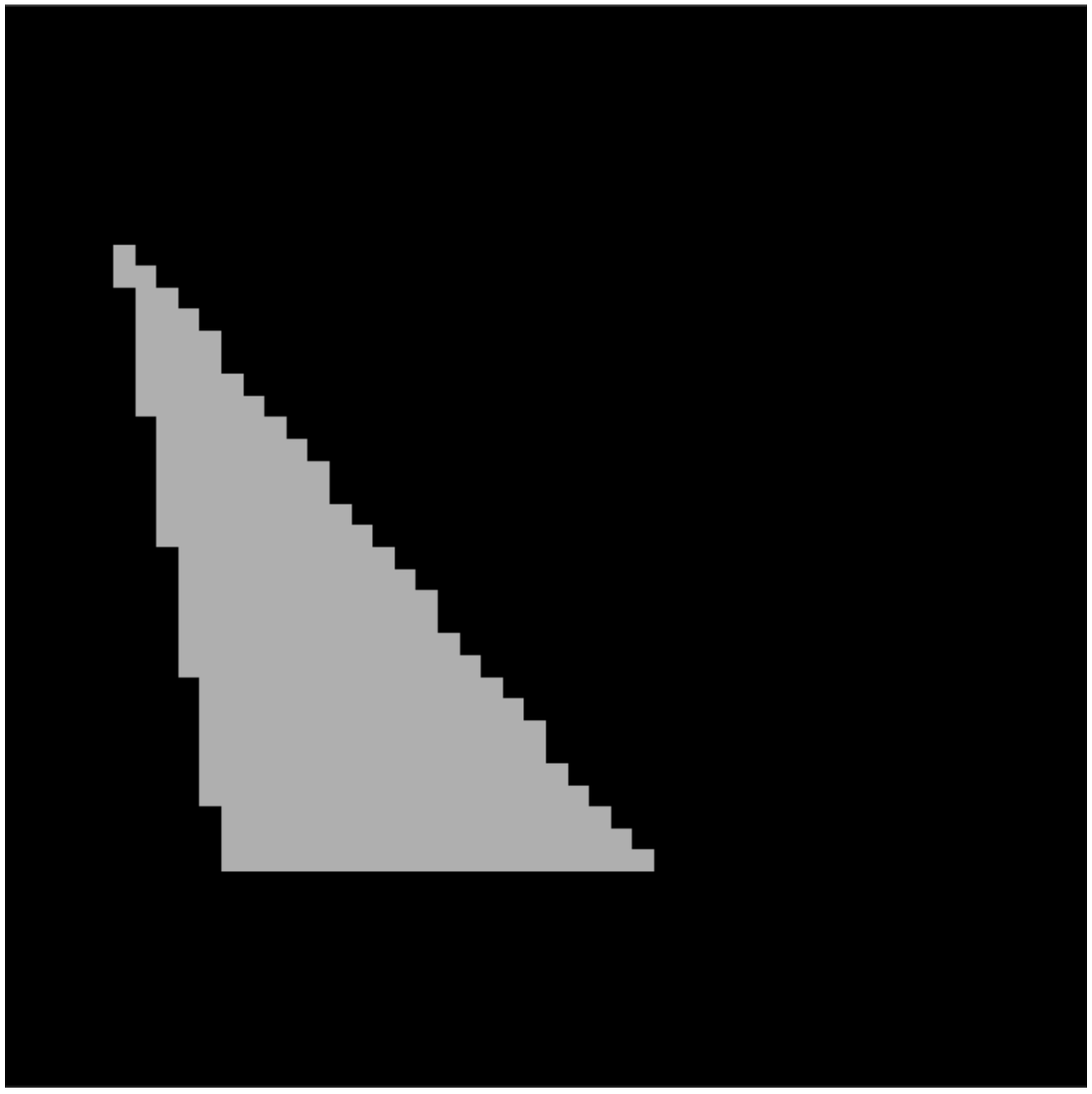
# Raster Images

- Flexible
- Display-native
- Expensive
- Not ideal
- But darn useful

# A model of the scene



# A Raster Image of the Scene



# Representing Raster Images: 2D Arrays of Numbers

- Bitmap (1 bit per pixel) $I : \mathbb{R}^2 \Rightarrow$
- Grayscale (usually 8 bpp) $I : \mathbb{R}^2 \Rightarrow$
- Color (usually 24 bpp) $I : \mathbb{R}^2 \Rightarrow$
- Floating-point (gray or color) $I : \mathbb{R}^2 \Rightarrow$
  - Bad for display, but good for processing
  - Allows **high dynamic range**
  - For LDR, values range from 0-1 by convention

# Raster Images: Storage

1 **megapixel** image - 1024x1024:

- Bitmap (1 bit per pixel) - **128 KB**
- Grayscale (8 bpp) - **1 MB**
- Color (24 bpp) - **3 MB**
- Floating-point (color) - **12MB**

# Aside: Performance

**Fact**: A 1 **megapixel** image has 1024x1024 = 1048576 = $2^{20}$ pixels.

# Aside: Performance

**Fact**: A 1 **megapixel** image has 1024x1024 = 1048576 = $2^{20}$ pixels.

**Consequence**: creating a 1 megapixel image requires making $2^{20}$ decisions.

# Aside: Performance

**Fact**: A 1 **megapixel** image has 1024x1024 = 1048576 = $2^{20}$ pixels.

**Consequence**: creating a 1 megapixel image requires making $2^{20}$ decisions.

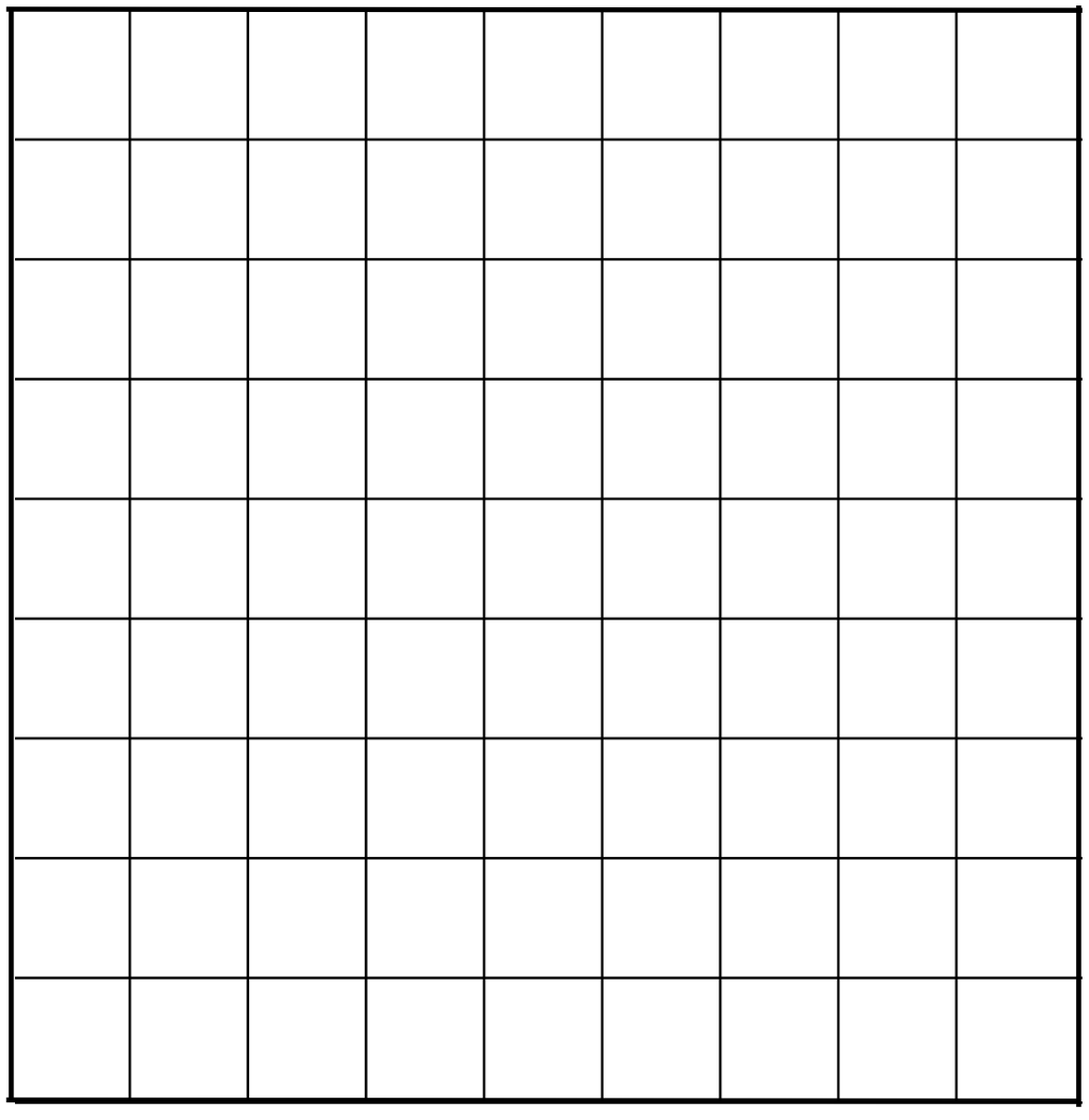
**Implication**: performance matters.

# Raster images are *sampled*

**function** that maps 2D *positions* to *distributions of radiant energy*

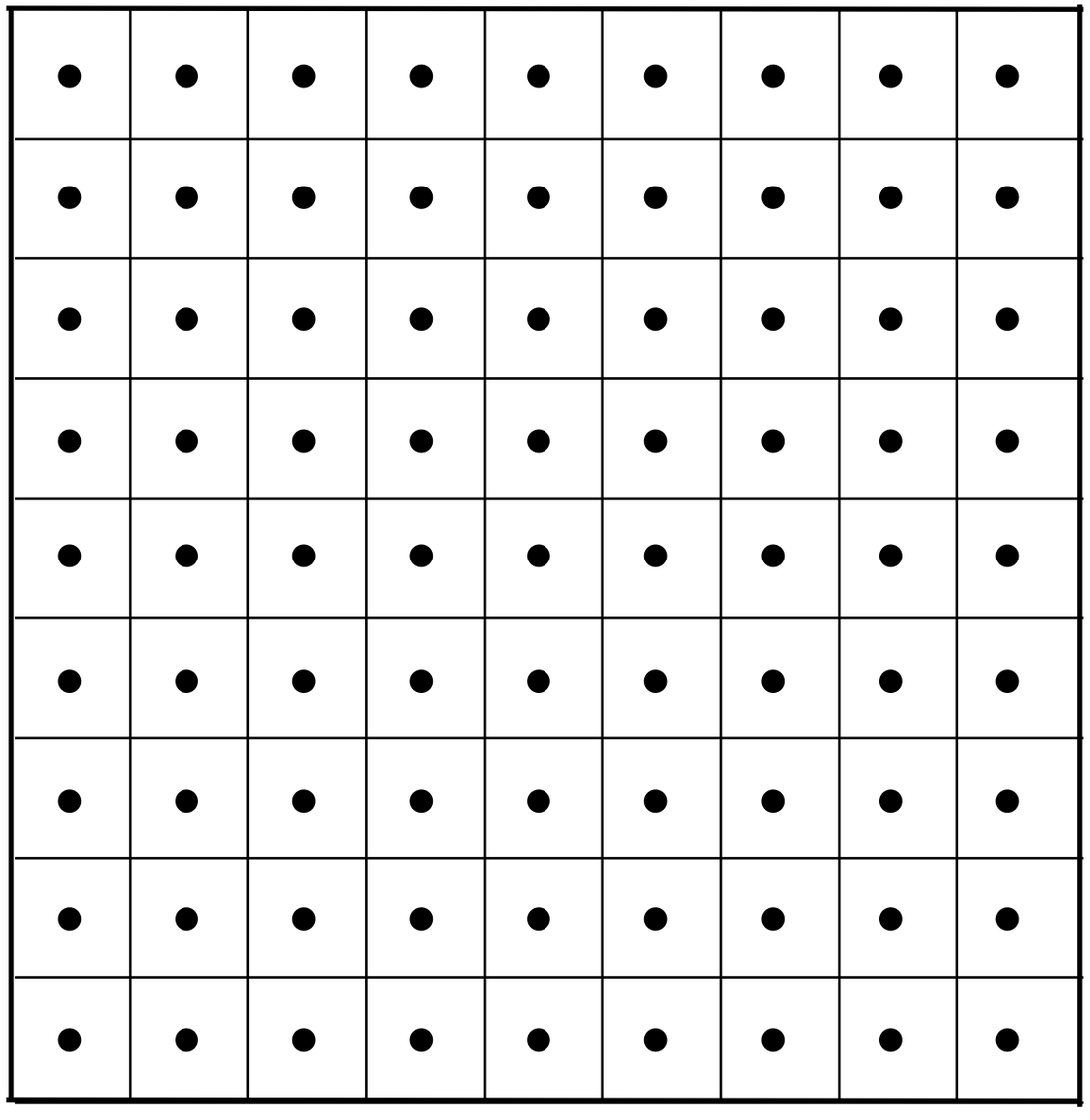
# Representing Raster Images

What do pixels *mean*?



# 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…

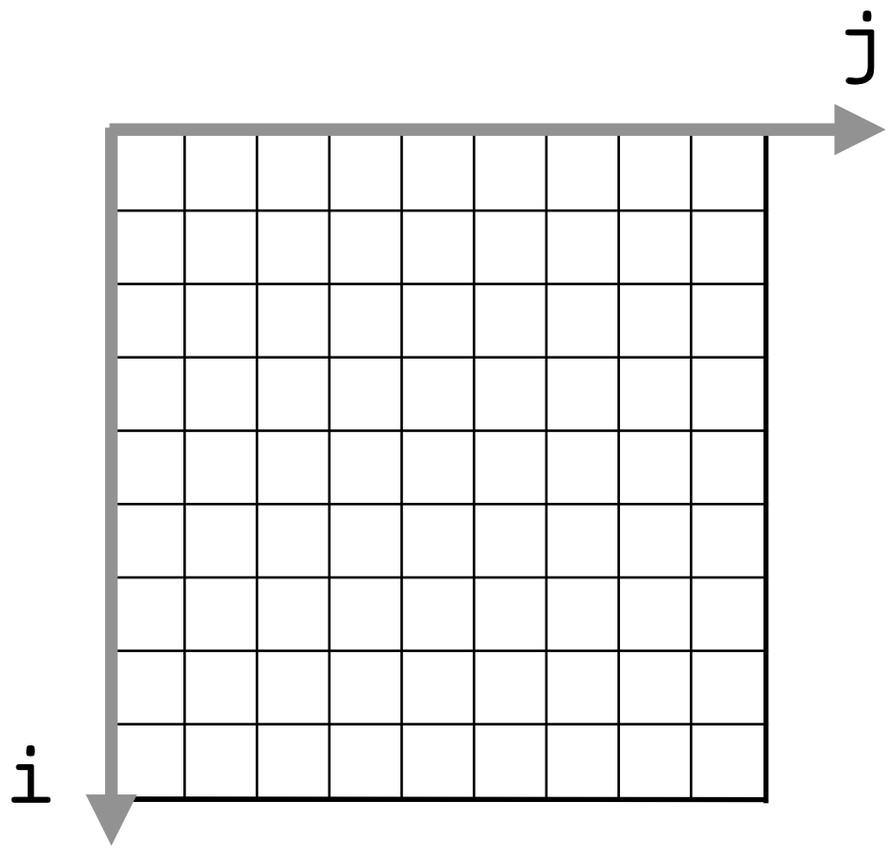
…of type RGB{Float32}…

…with dimensions (height x width)

# 2D Arrays in Julia

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

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

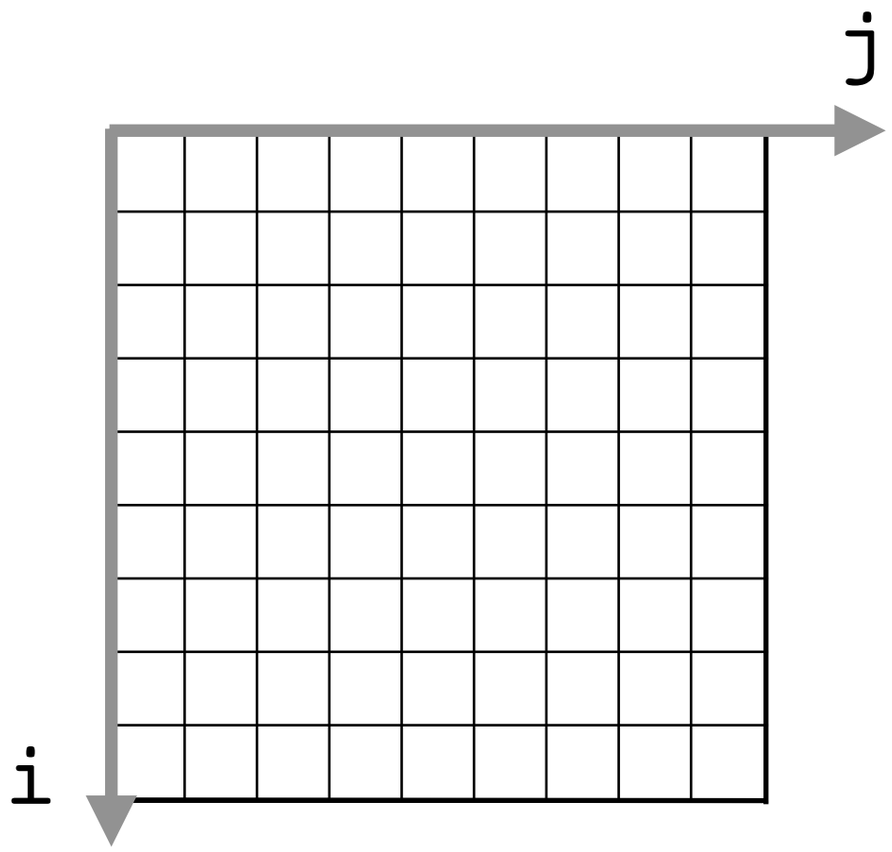


# 2D Arrays in Julia

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

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

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



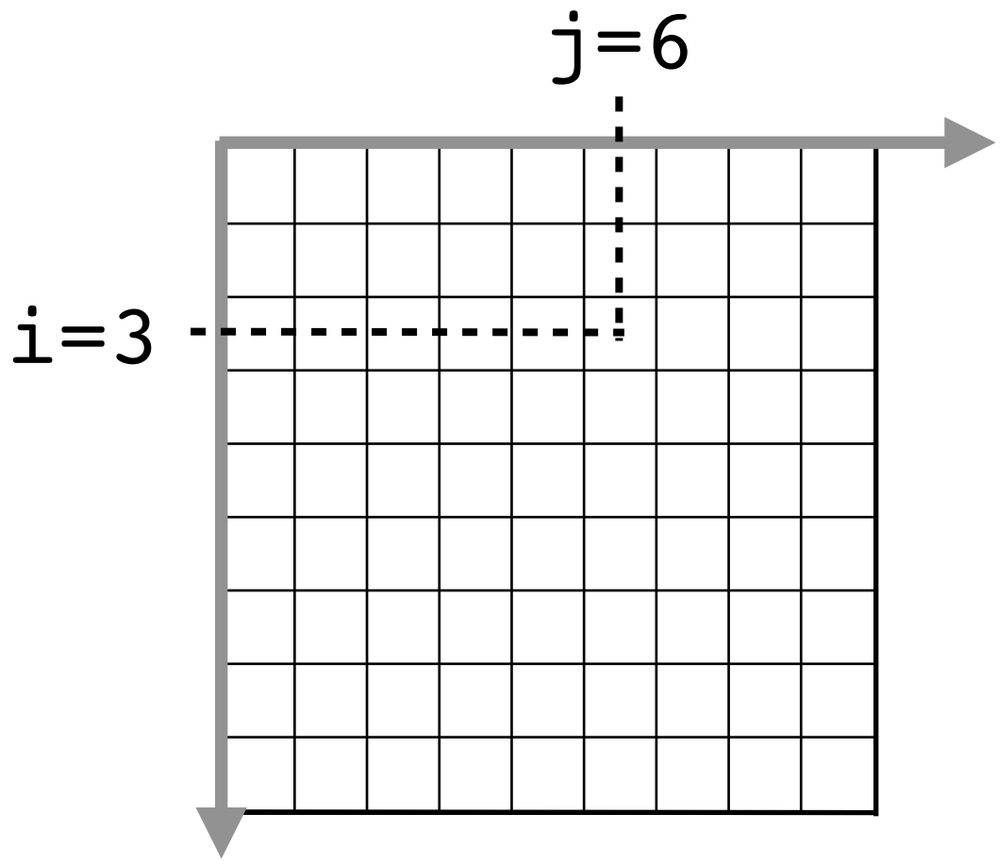
# 2D Arrays in Julia

```
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]
```



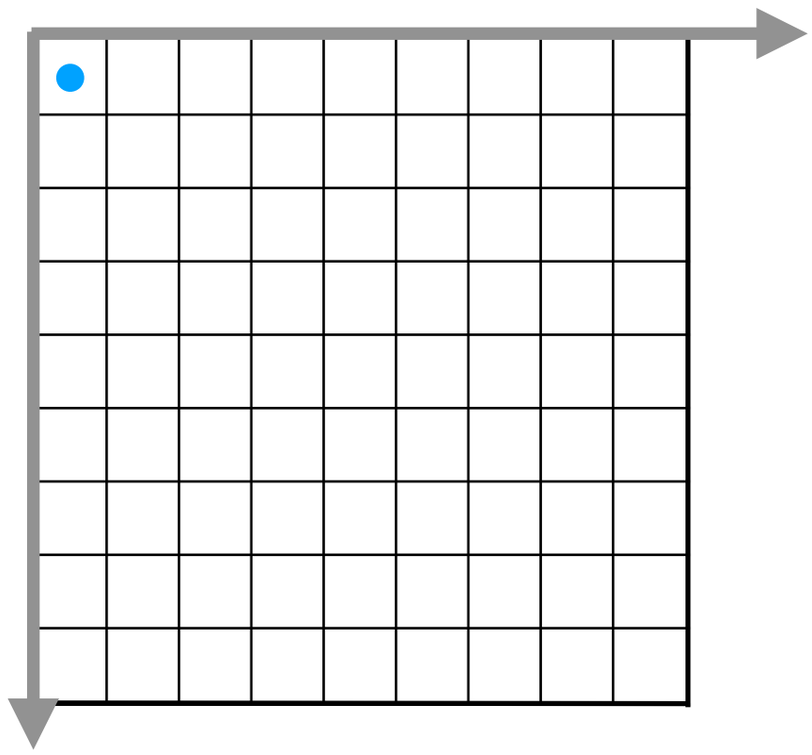
# 2D Arrays in Julia

```
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)?



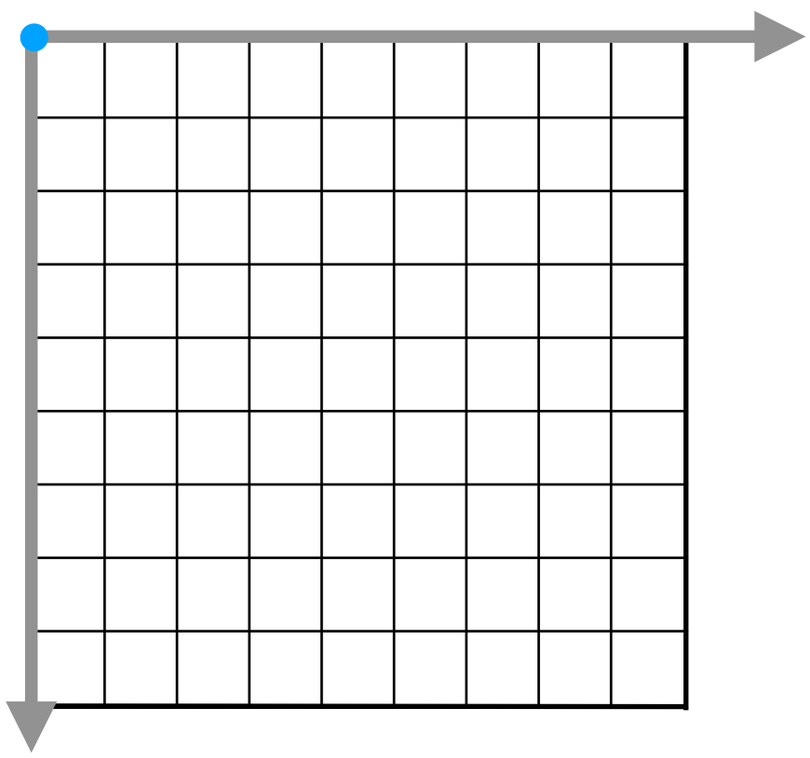
# 2D Arrays in Julia

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

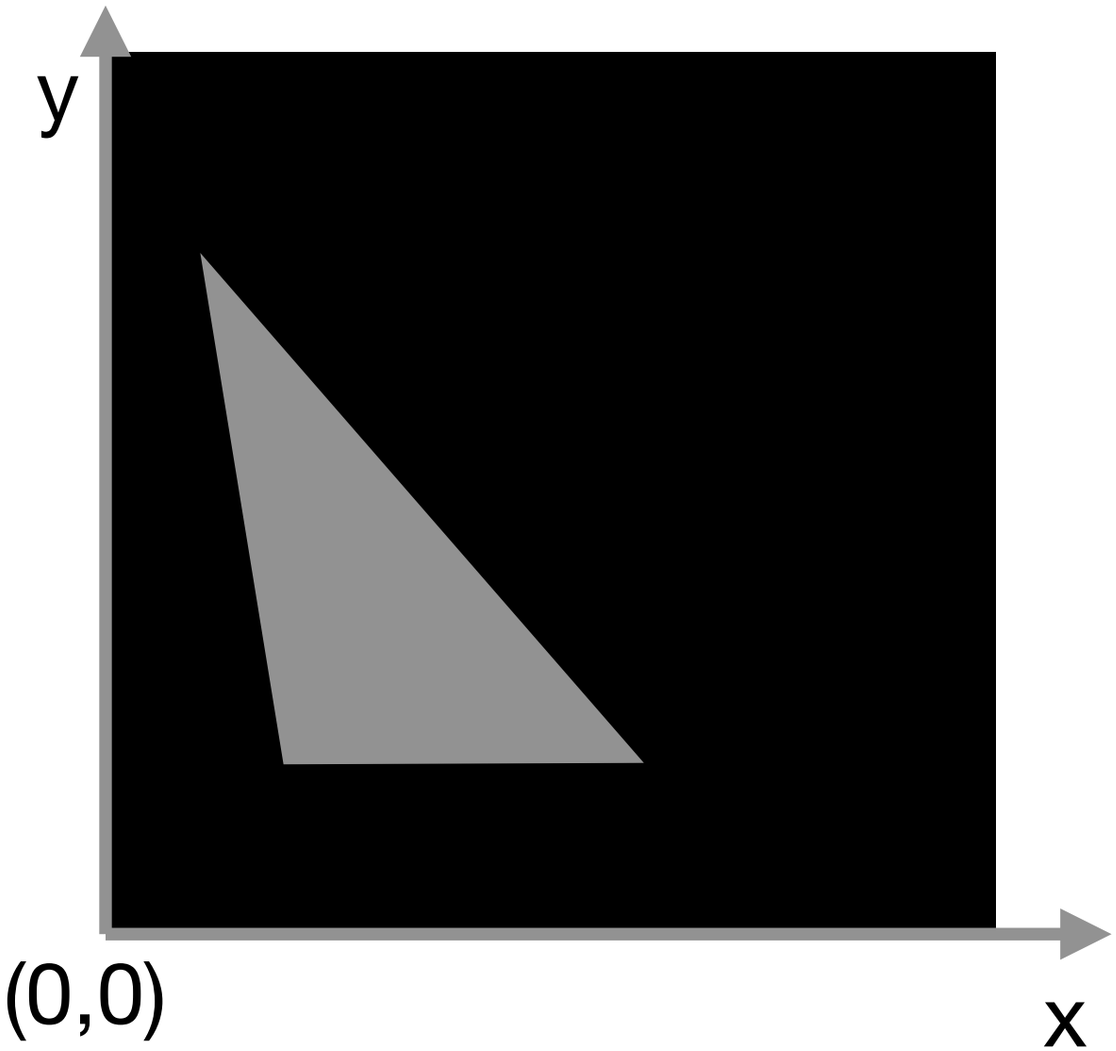
Matrix-style **1-based** indexing (row, column):

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

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



# Raster Images: Coordinate Systems




Model coordinates

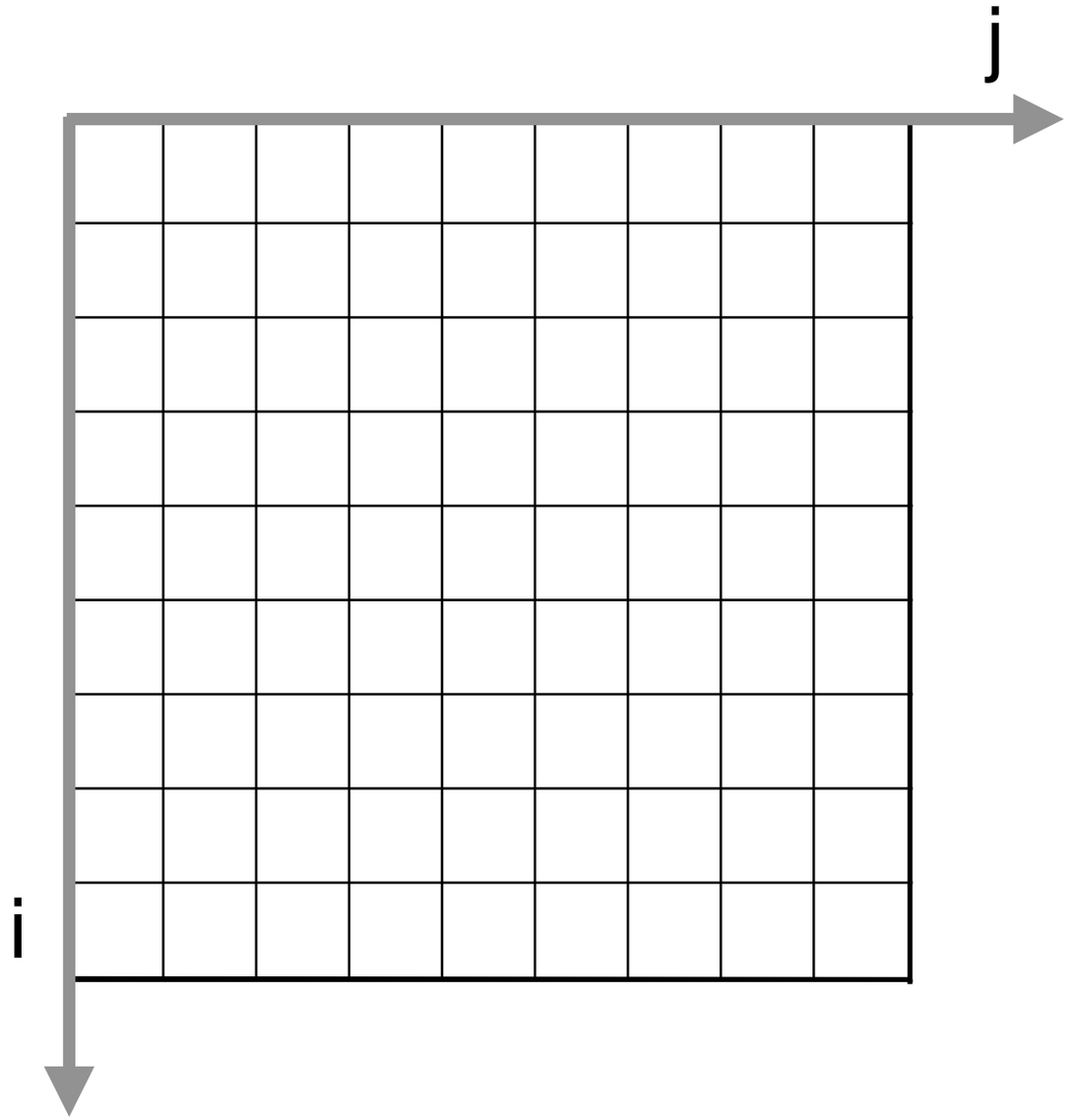
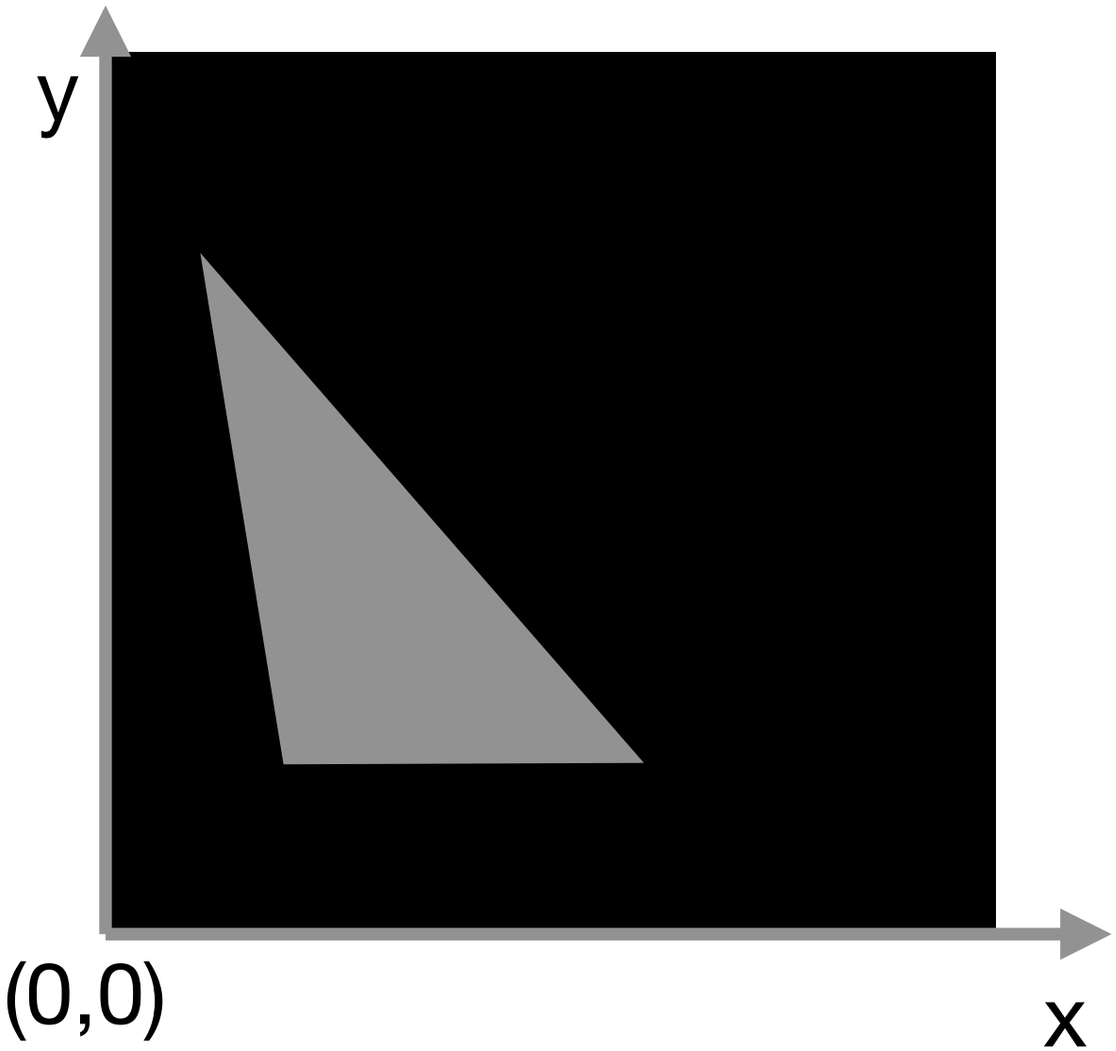



Image coordinates

# Raster Images: Coordinate Systems




Model coordinates

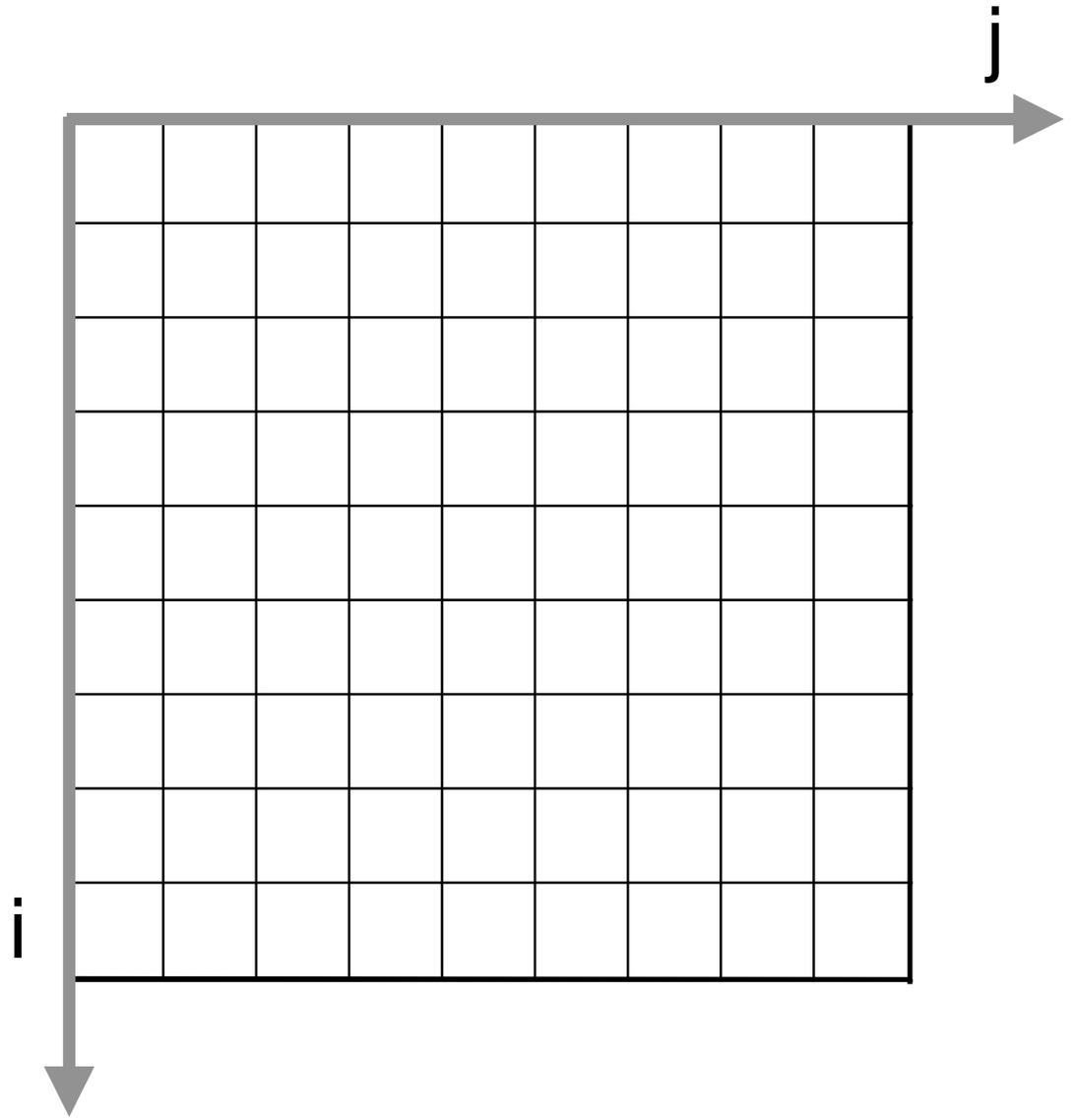
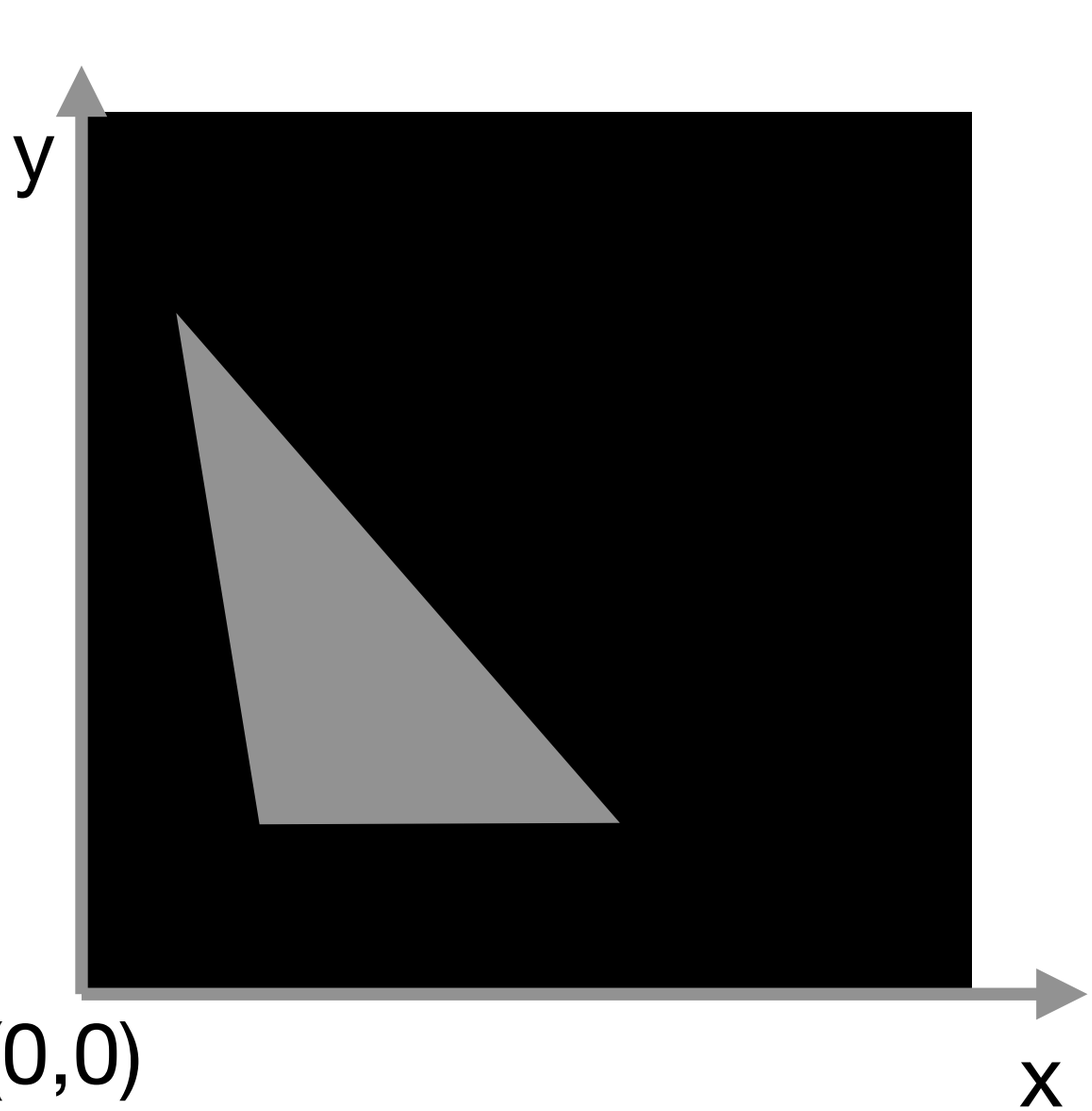



Image coordinates

We need to render it onto this.

# Raster Images: Coordinate Systems




Model coordinates

We modeled our triangle like this.

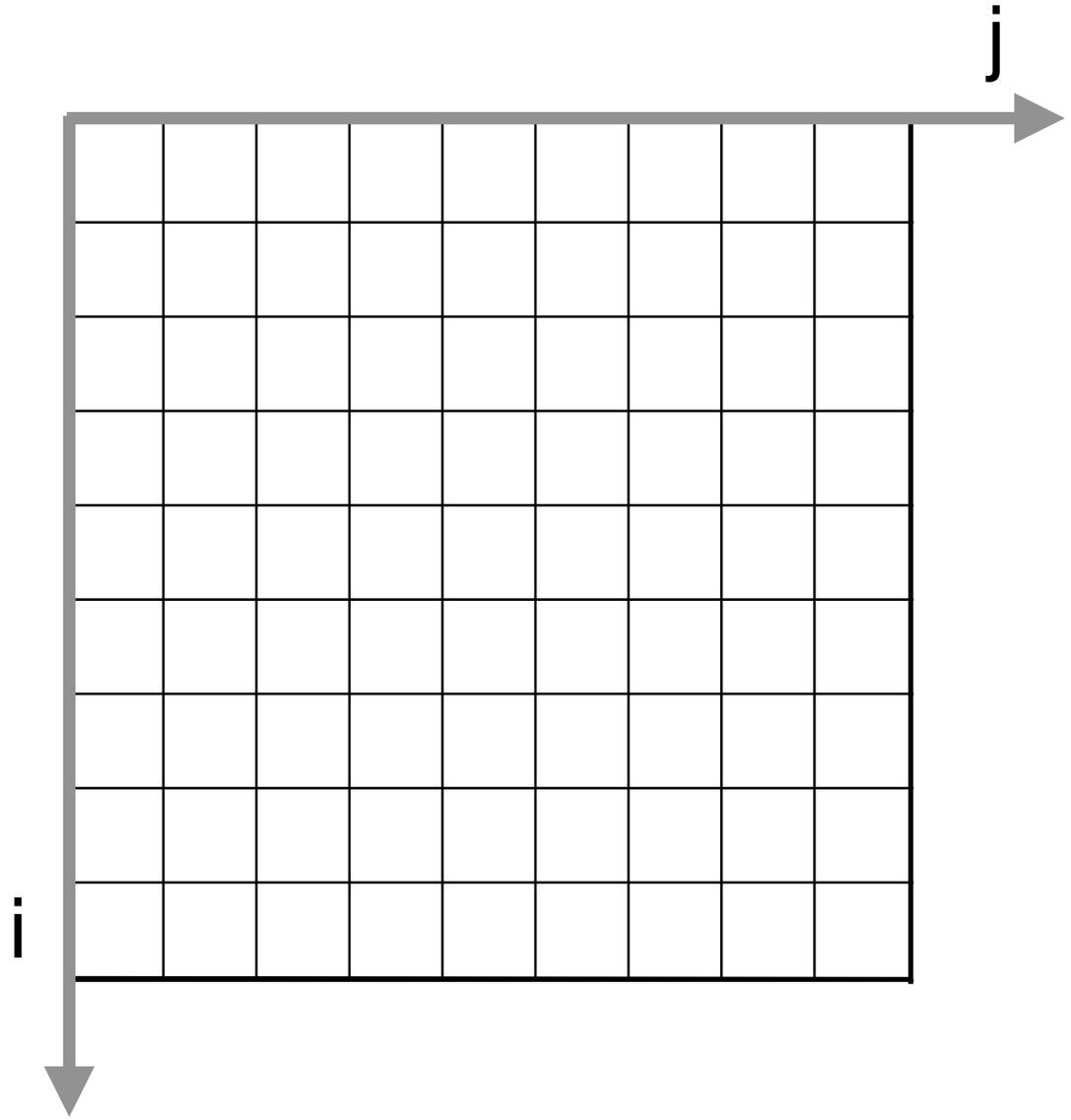
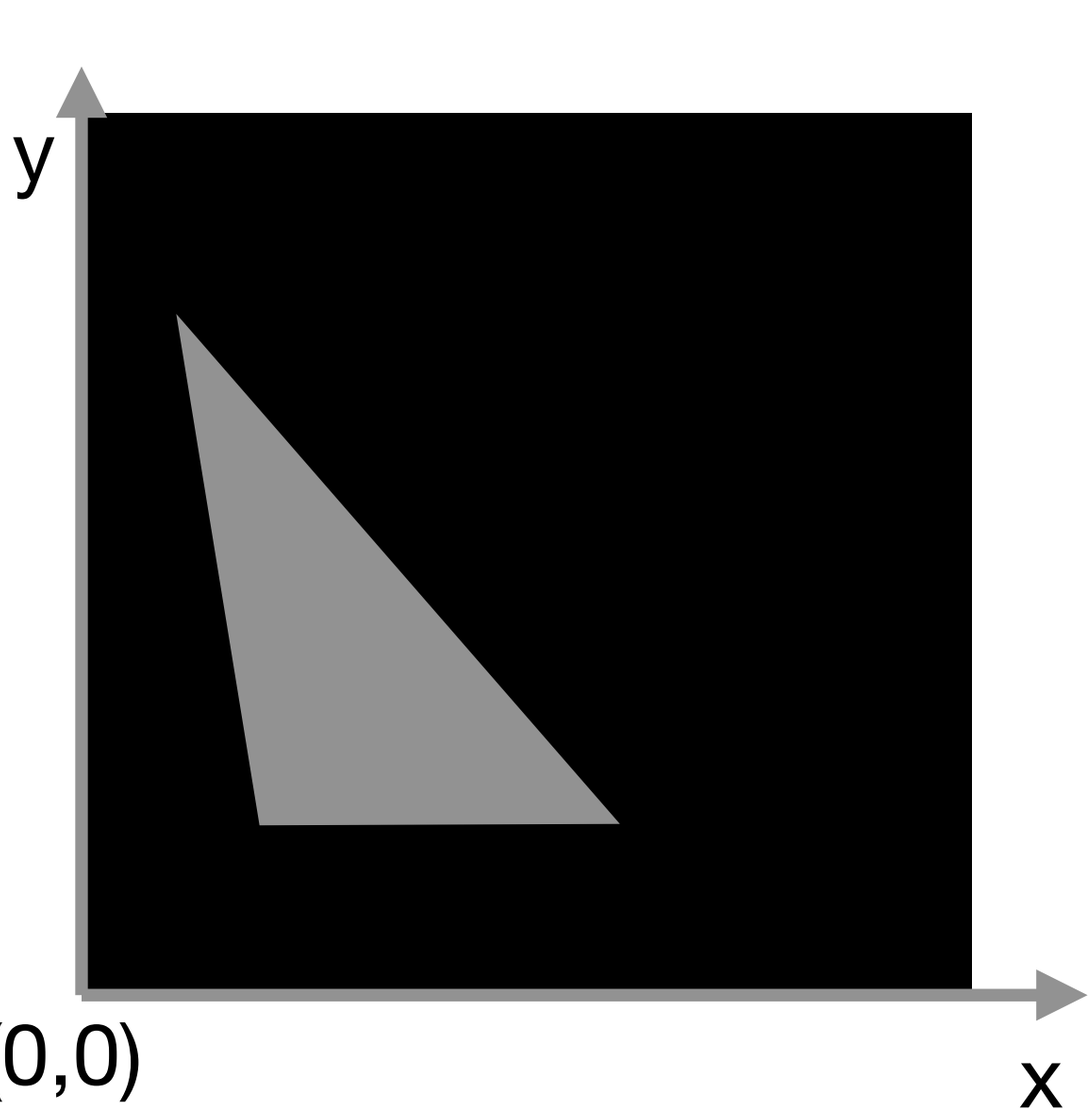



Image coordinates

We need to render it onto this.

# Raster Images: Coordinate Systems




Model coordinates

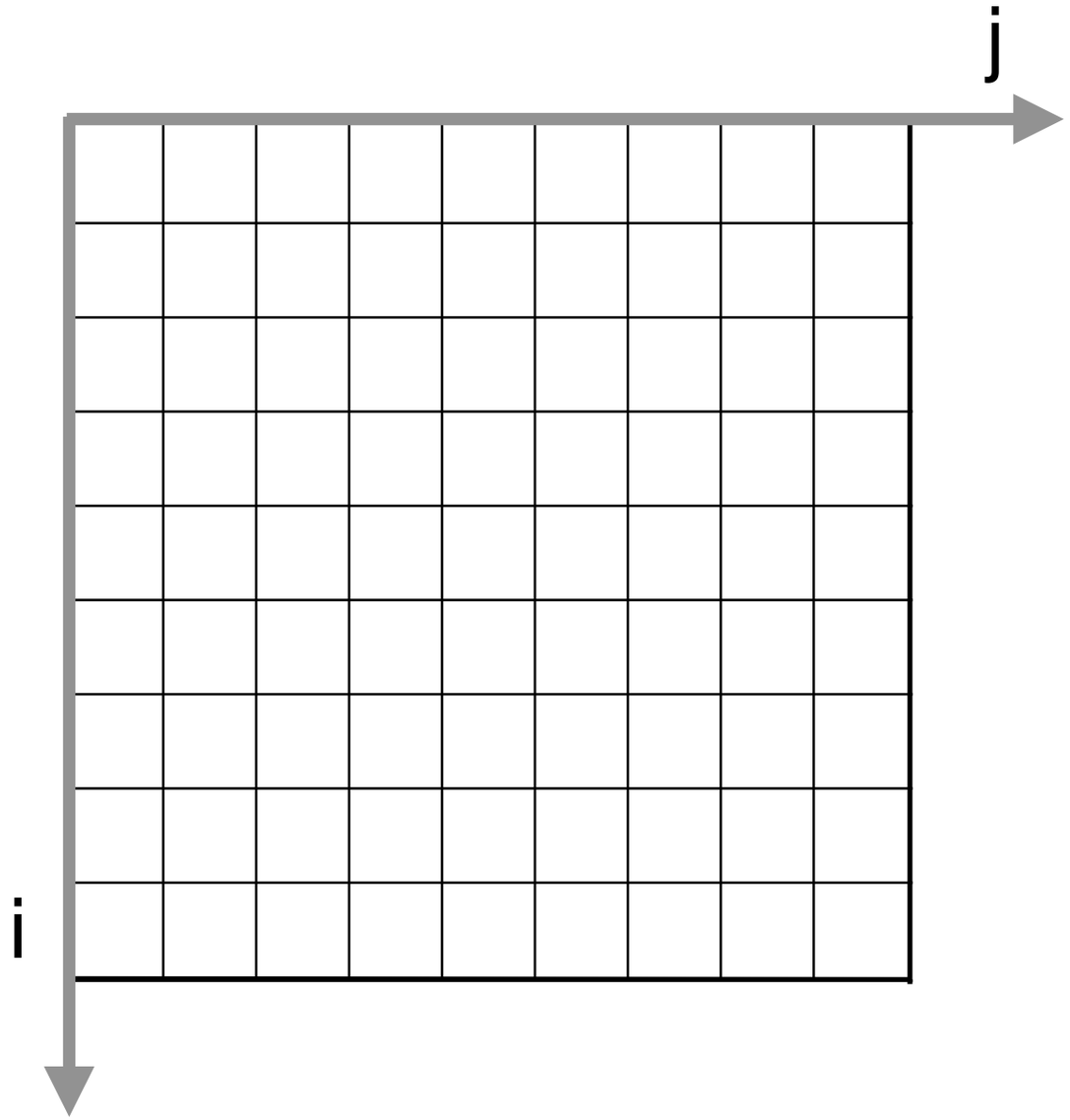



Image coordinates

We modeled our triangle like this.

We need to render it onto this.

We need a coordinate transformation!