CSC418/2504 Computer Graphics

CSC 418/2504: Computer Graphics

Course web site (includes course information sheet):

https://github.com/dilevin/computer-graphics-csc418

Instructors:

Prof. David I.W. Levin diwlevin@cs.toronto.edu

Gavin Barill gavin.barill@mail.utoronto.ca

Proffice Hours:

Dave – Tuesday 5-6pm, Wednesday 5-6pm

Gavin – TBD

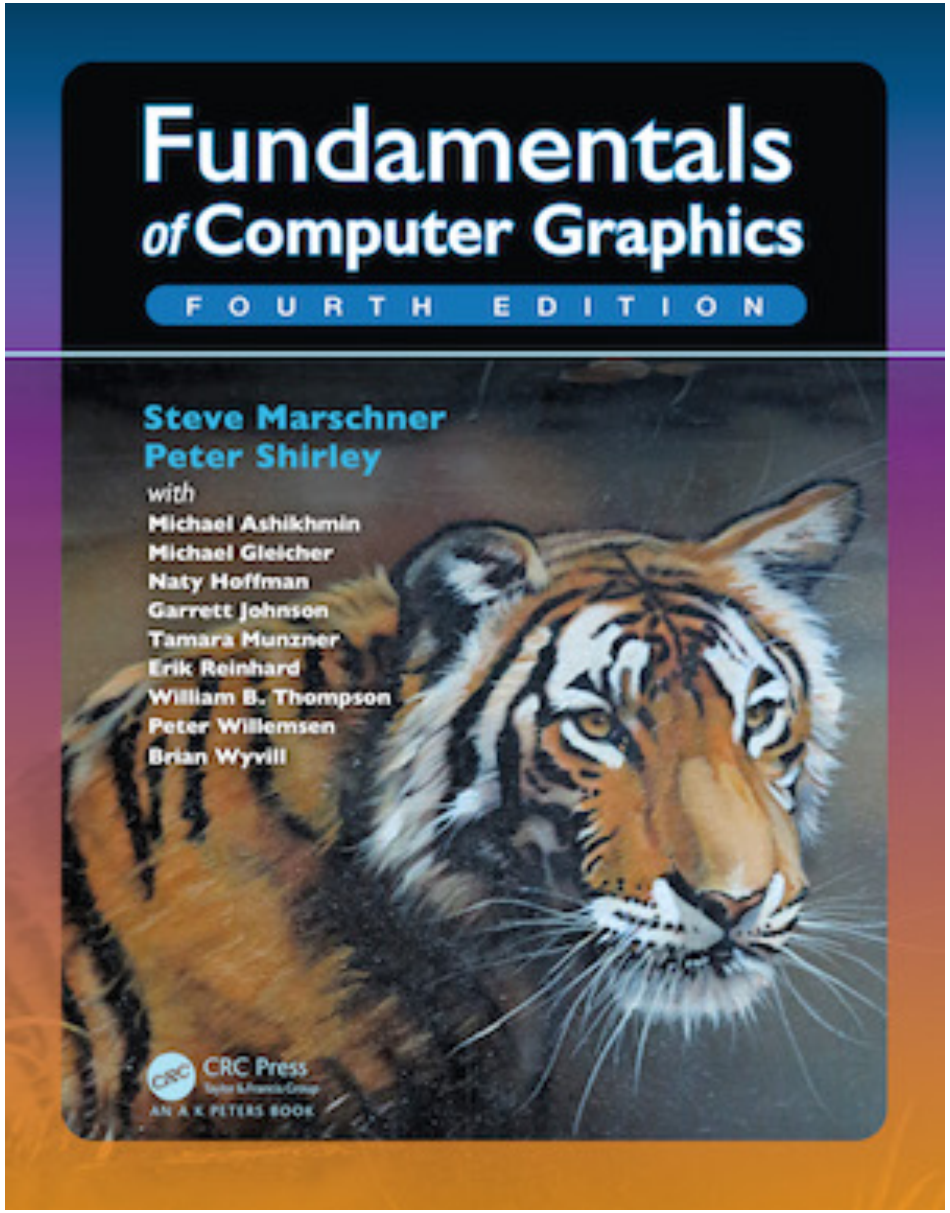
TA Office Hours: TBD

Schedule

Lectures
LEC0101 Wednesdays 15:00-17:00 in BA1190
LEC2001 Wednesdays 15:00-17:00 in BA1100
LEC0201 Tuesdays 15:00-17:00 in BA1170
LEC2201 Tuesdays 15:00-17:00 in BA1170

Tutorials
LEC0101 Monday 15:00-16:00 in BA1190
LEC2001 Monday 15:00-16:00 in BA1190
LEC0201 Monday 15:00-16:00 in BA1210
LEC2201 Monday 15:00-16:00 in BA1210

Required Textbook
(Lots of figures on slides adapted from here)



Schedule (on the webpage)

Week	Topic / Event
1	Introduction, Demos of Solutions Assignment 1 (Raster Images) due 18/01, Lecture 1
2	Assignment 2 (Ray Casting) due 25/01, Lecture 2
3	Assignment 3 (Ray Tracing) due 01/02, Lecture 3
4	Assignment 4 (Boundary Volume Hierarchy) due 08/02, Lecture 4
5	Assignment 5 (Meshes) due 15/02, Lecture 5
6	*Work on Assignment 5* , Lecture6
Reading Week February 18-22	Study for 1st exam
Monday, February 25	In-tutorial Exam (20% of grade)
7	Assignment 6 (Shader Pipeline) due 08/03, Lecture 7
8	Assignment 7 (Kinematics) due 29/03, Lecture 8
9	Final Image/Animation/Game Competition due 05/04, Lecture 9
Sunday, March 17	Drop date (consider if grade so far is <50%)
10	*Study for exam next week and work on Final Project.*
Monday, March 25	In-tutorial Exam (10% of grade)
11	Current Graphics Research and Graduate Course Preview
12	🏆 Showcase 🏆

Academic Honesty Policy

It’s on the webpage and is mandatory reading!

Administrivia

Grading:

%	Item
70%	Assignments
20%	Monday, Feb 25, in-tutorial exam
10%	Monday, March 25, in-tutorial exam

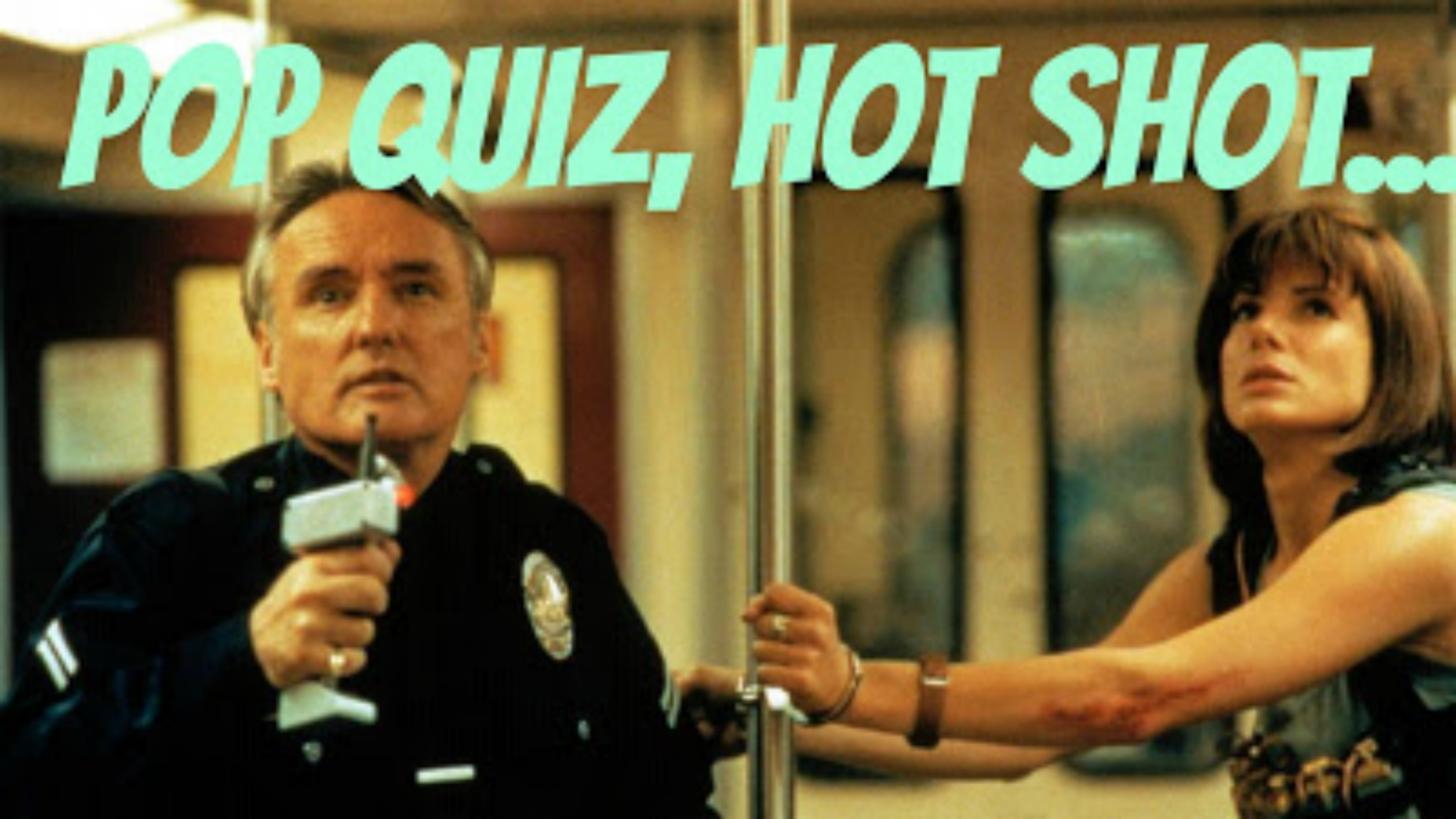
Tutorial sessions:

- Math refreshers, tutorials on graphic libraries, additional topics.
- Attendance STRONGLY encouraged since I will not be lecturing on these topics in class.

Today

1. Introduction to Computer Graphics
2. Preview of class assignments
3. Raster Images

But First ……


POP QUIZ, HOT SHOT...

Preliminary Math Quiz

DON'T PANIC IT'S NOT PART OF YOUR GRADE!

Goals:

1. Show you what kind of mathematical background is expected in this course
2. Show you what you need to brush up on. Questions about these basic math operations will not be answered by either Professors or TAs, we expect you to know this stuff.
3. Give you a sense of how ready you are to take this course.

Time: 20 minutes (should be more than enough)

DON'T PANIC IT'S NOT PART OF YOUR GRADE!

Introduction to Computer Graphics

What is Computer Graphics?

Computers:

accept, process, transform and present information.

Computer Graphics:

accept, process, transform and present information

in a visual form.

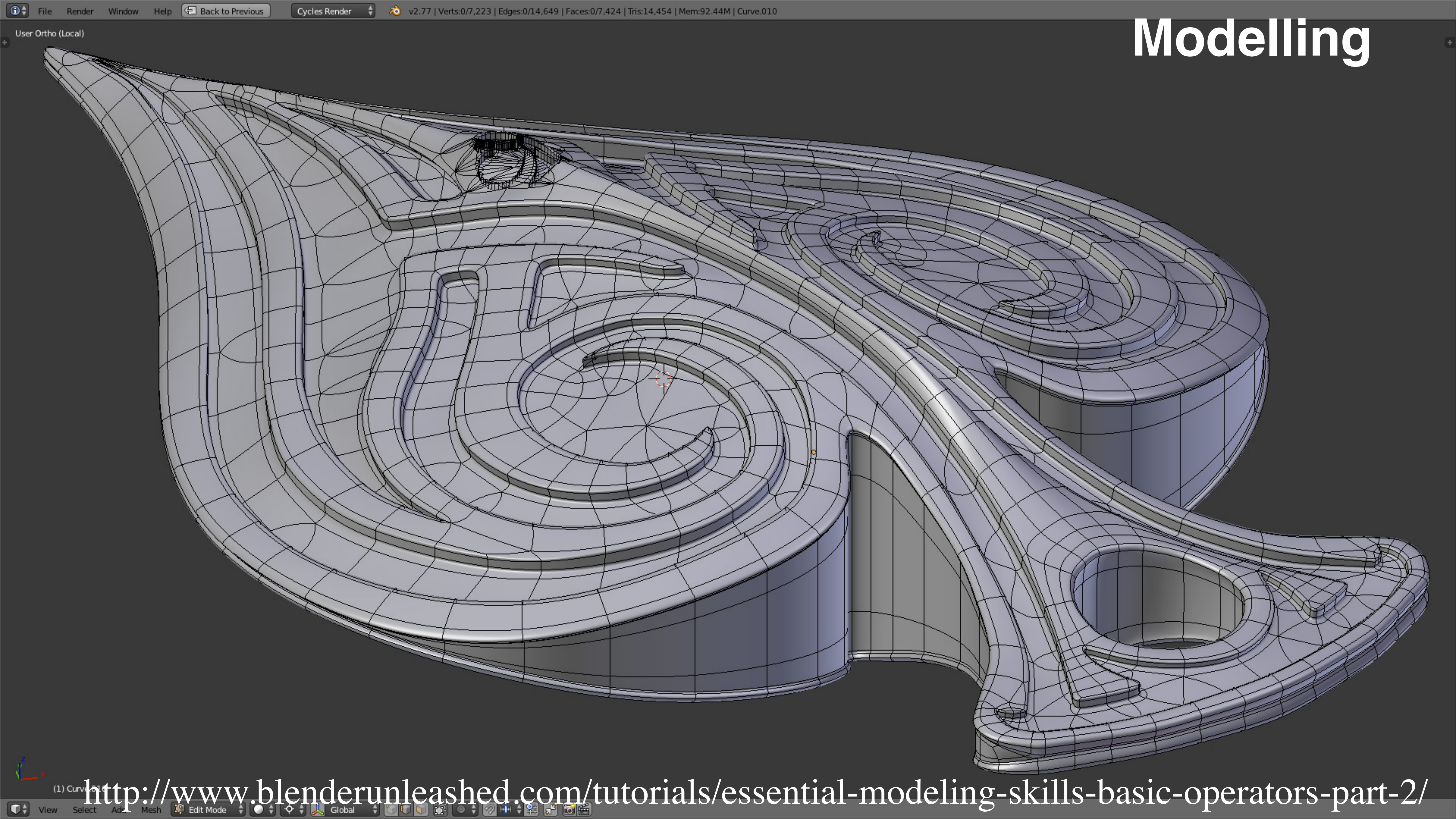
“Core” Areas of Computer Graphics

Modeling

Rendering

Animation

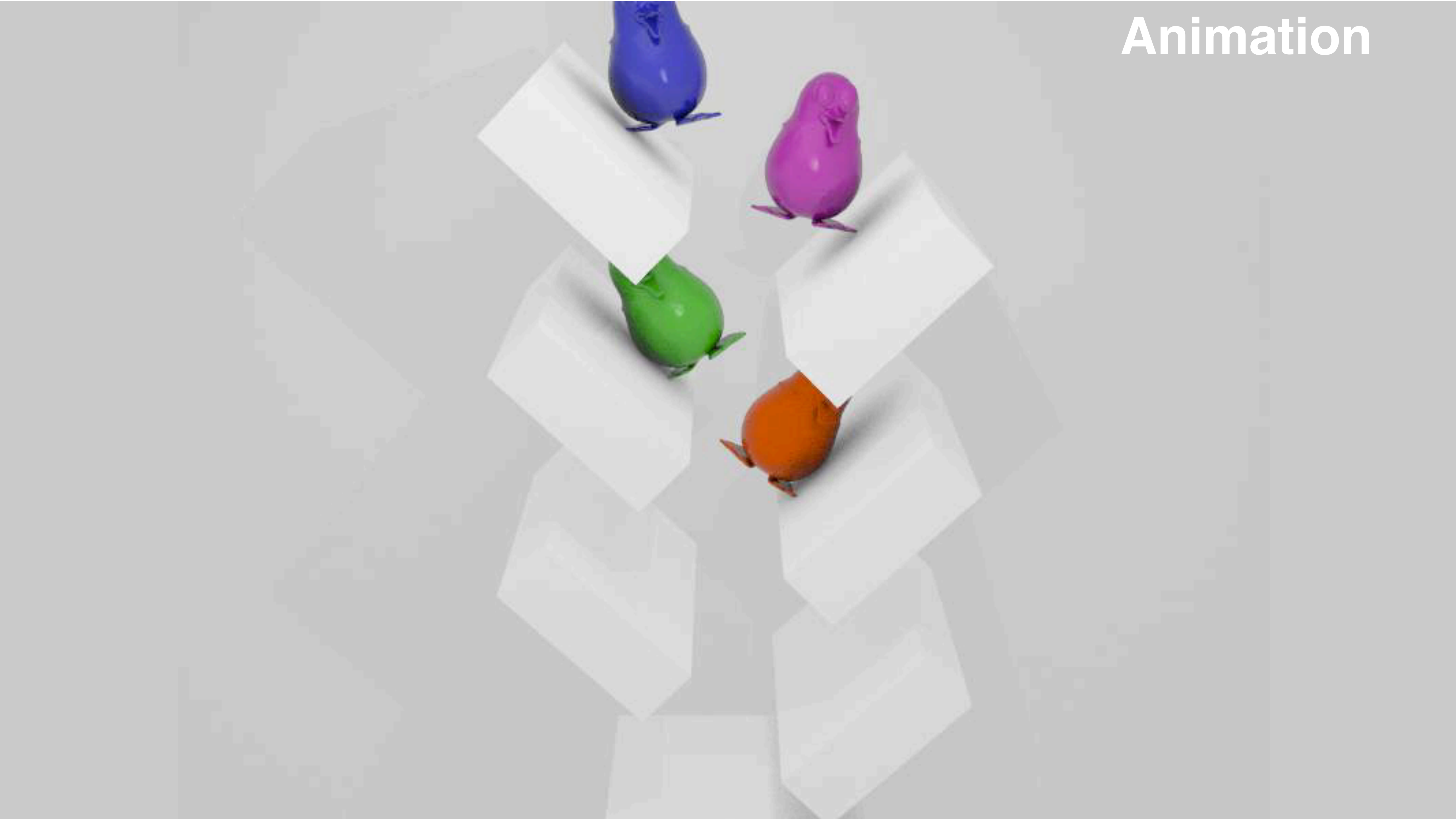
Modelling



http://www.blenderunleashed.com/tutorials/essential-modeling-skills-basic-operators-part-2/

Rendering

Animation



Other Areas of Computer Graphics

User Interaction

Virtual Reality

Visualization

Image Processing

3D Scanning

Computational Photography

Assignment Previews

Raster Images

Ray Casting

Ray Tracing

Boundary Volume Hierarchies

Meshes

Shaders

Kinematics

Final Project: Image Showcase !

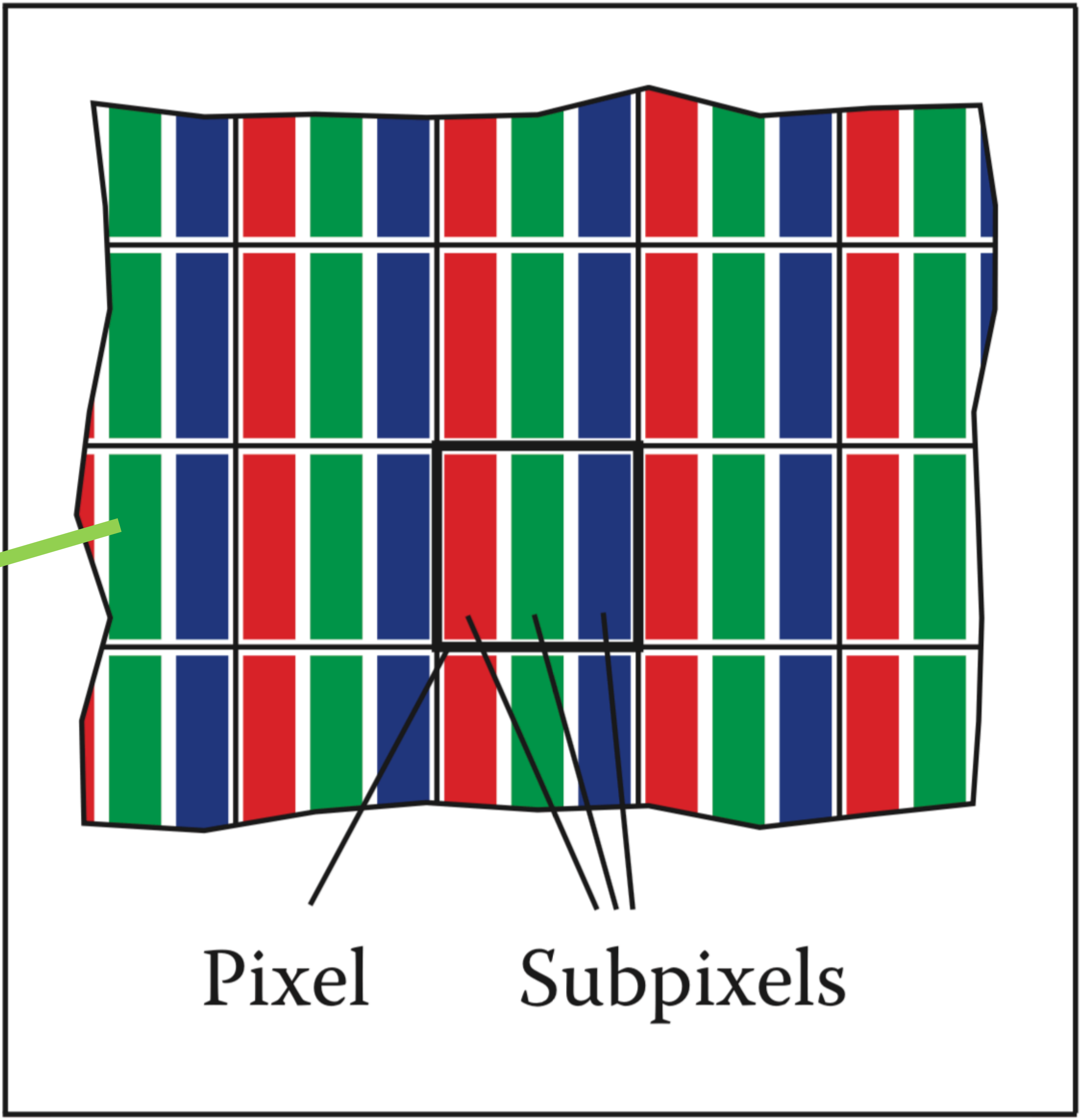
To the terminal …

Raster Images

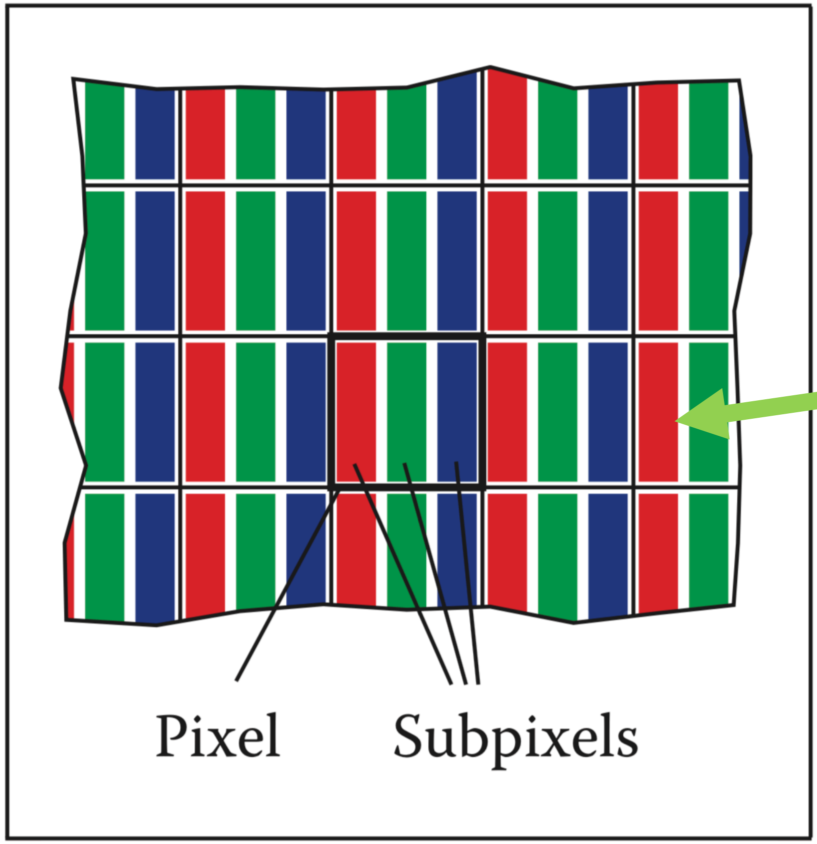
Raster Displays

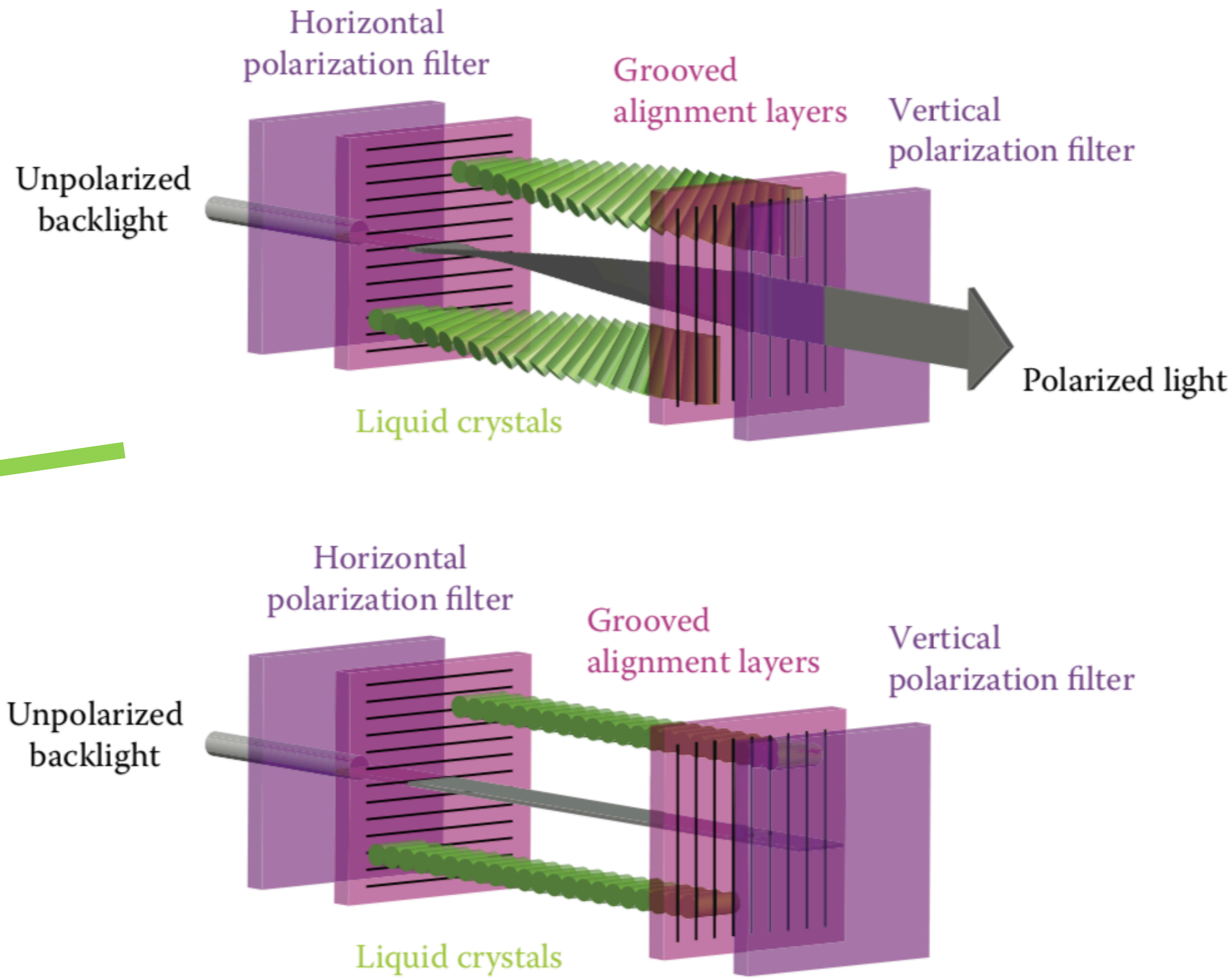
Raster Displays

Raster Displays



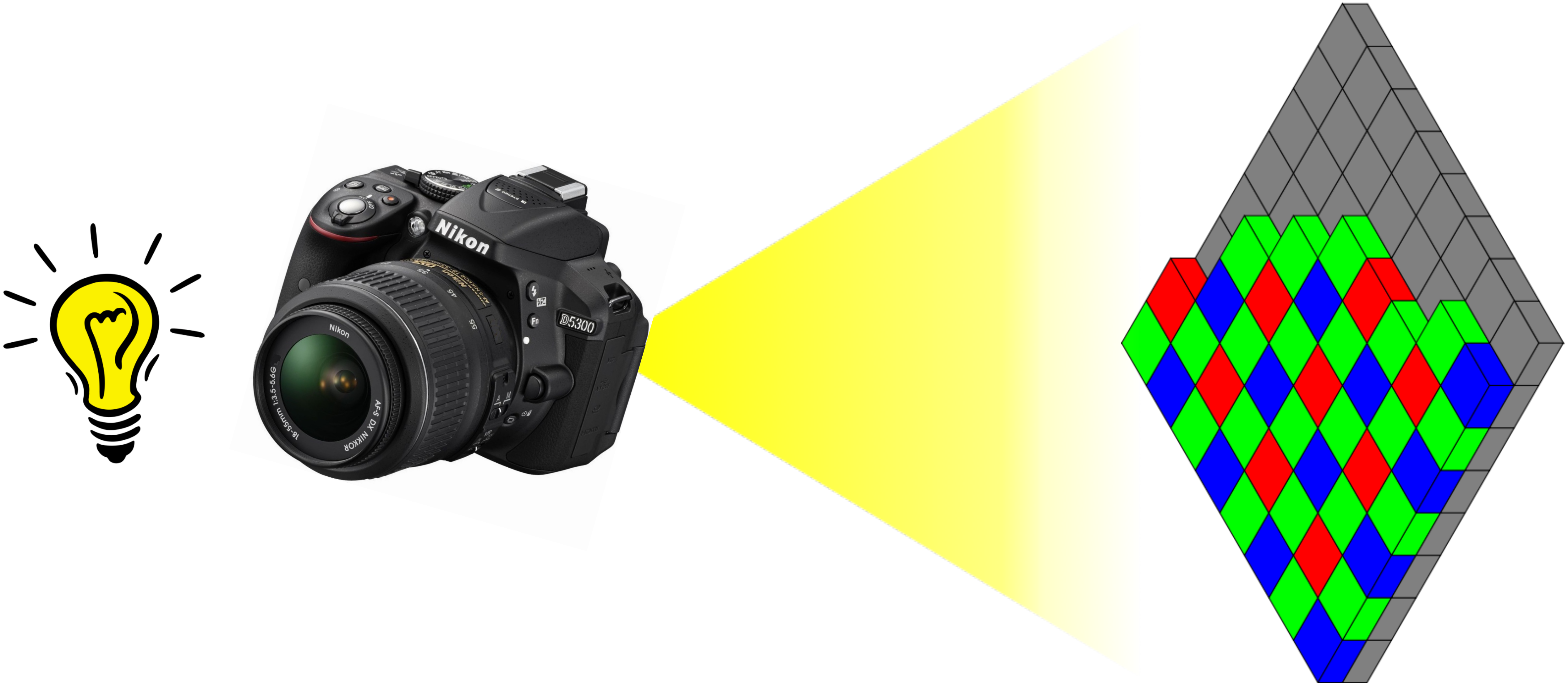
Raster Displays



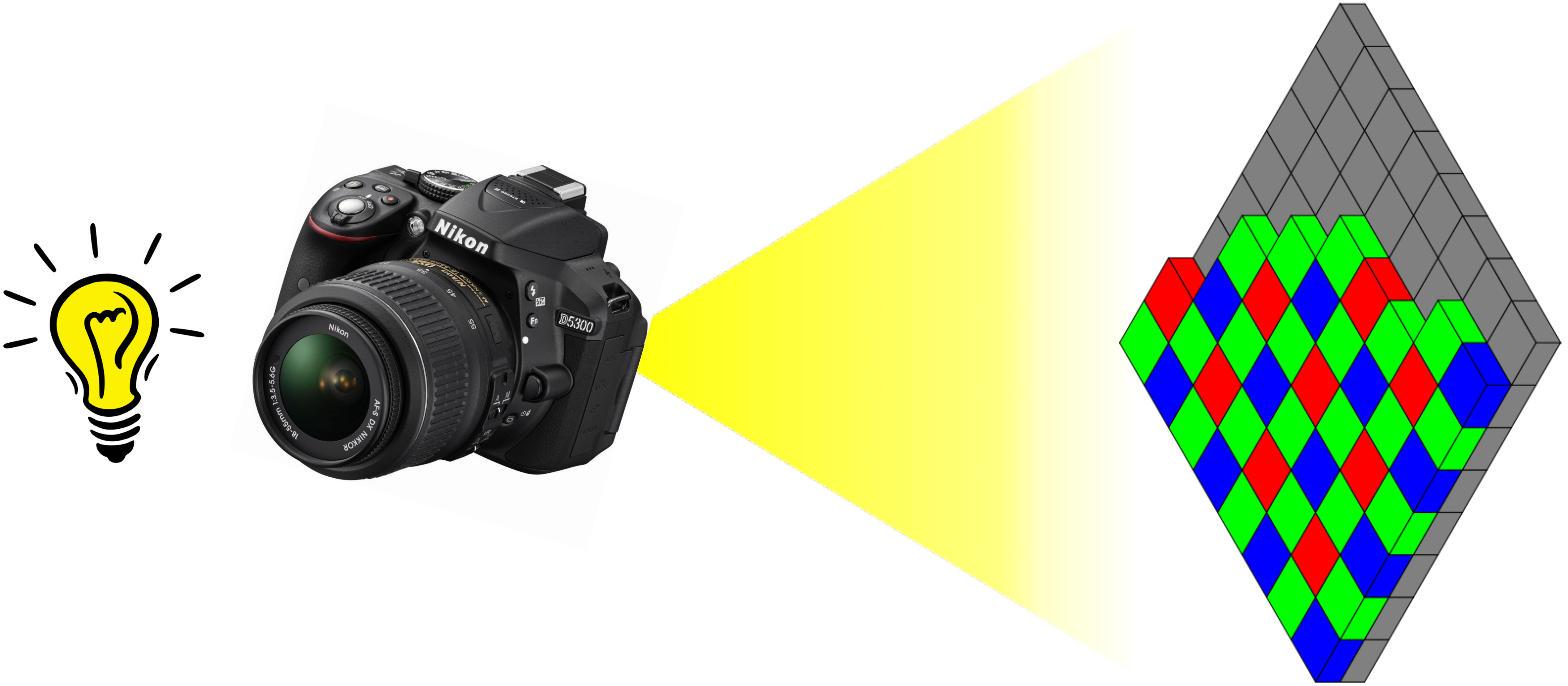


Raster Input Devices

Raster Input Devices

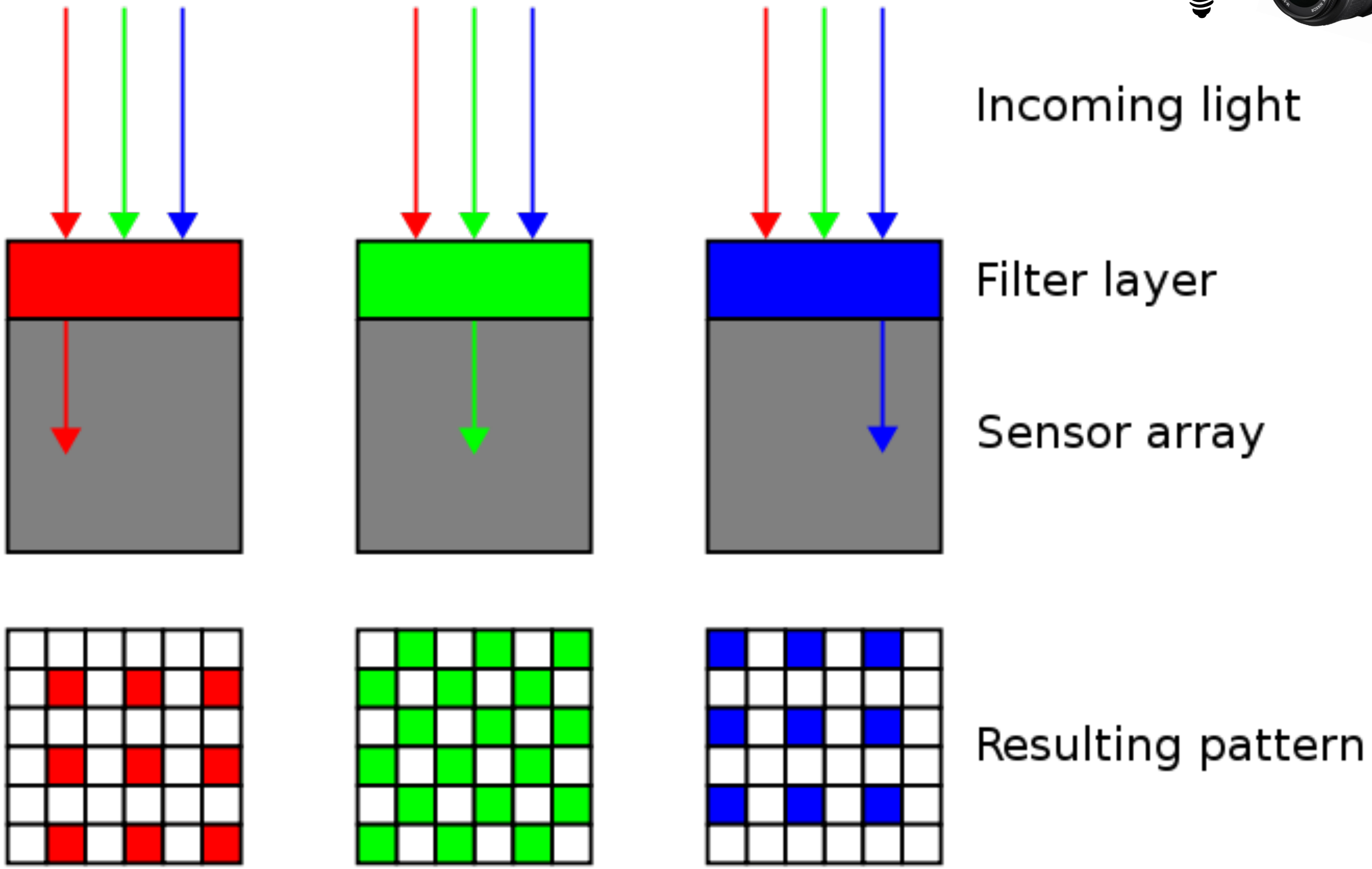


Raster Input Devices

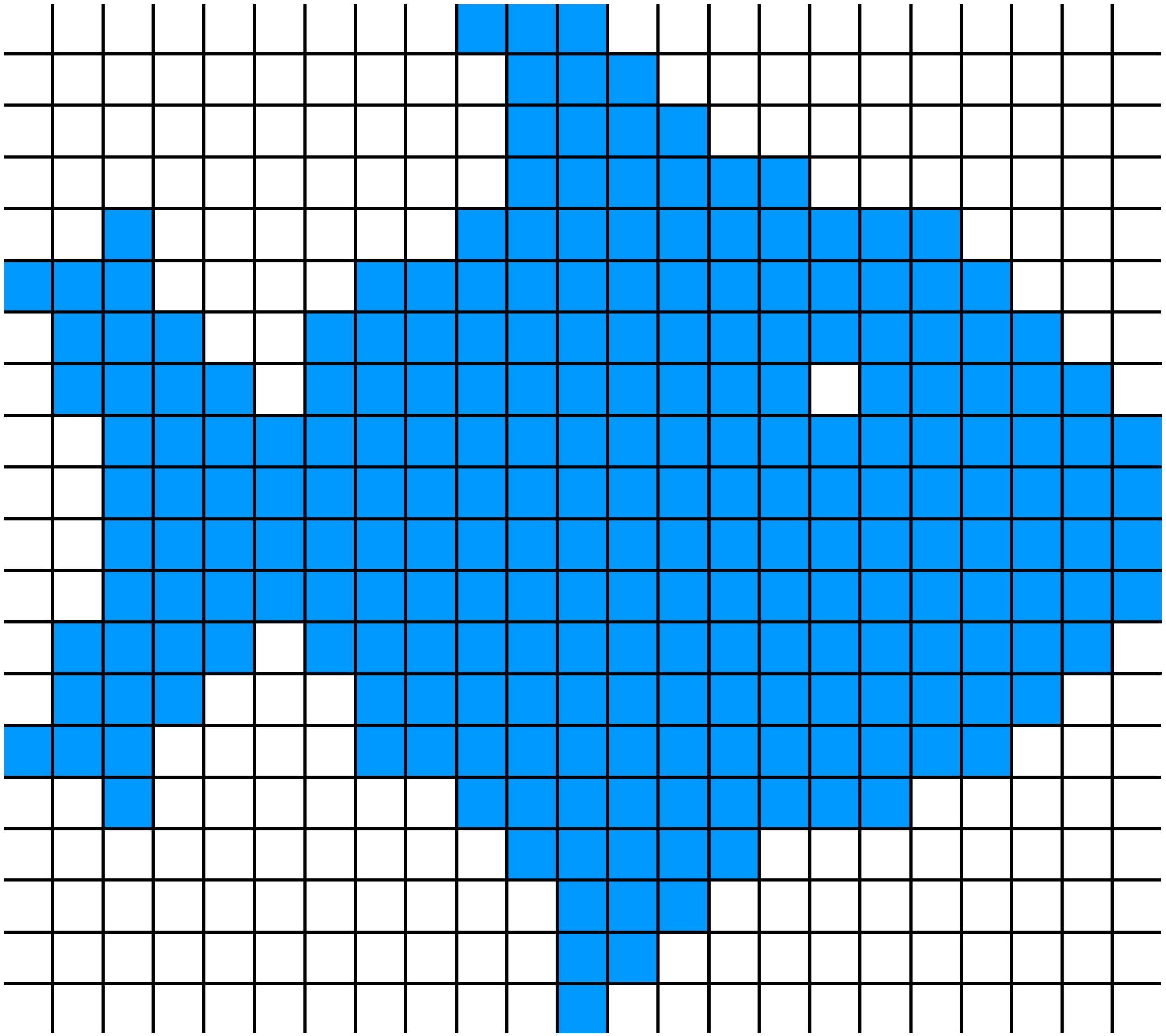


Bayer Filter

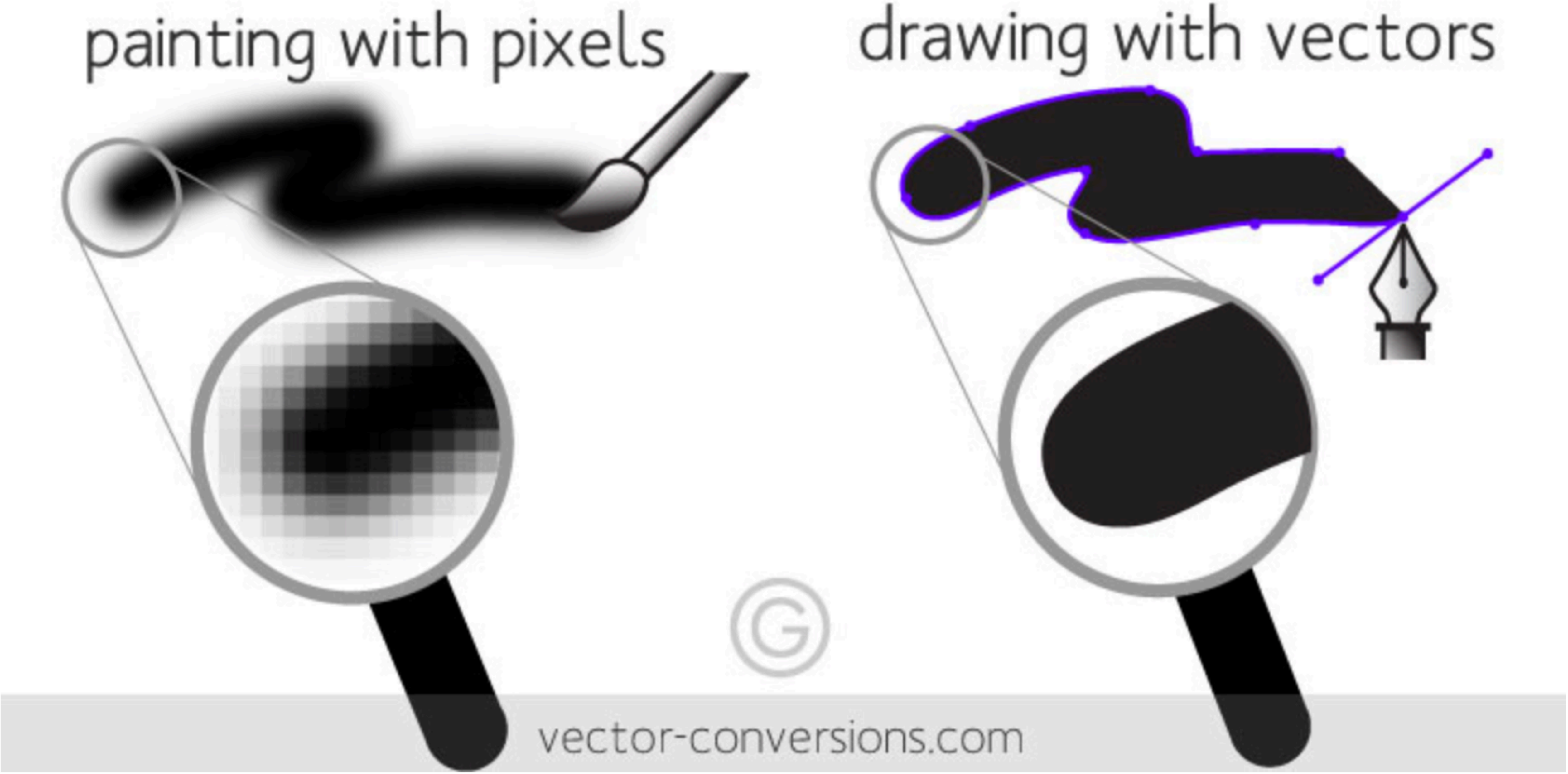
Raster Input Devices



Raster Image



Aside: More Than Just Raster Images



Images as a Function

$$I\left(x, y\right) : \mathbb{R}^2 \rightarrow \mathbb{R}^{+n}$$

Image coordinates ????

Images as a Function

nD Real Numbers > 0

$$I\left(x, y\right) : \mathbb{R}^2 \rightarrow \mathbb{R}^{+n}$$

Image

coordinates

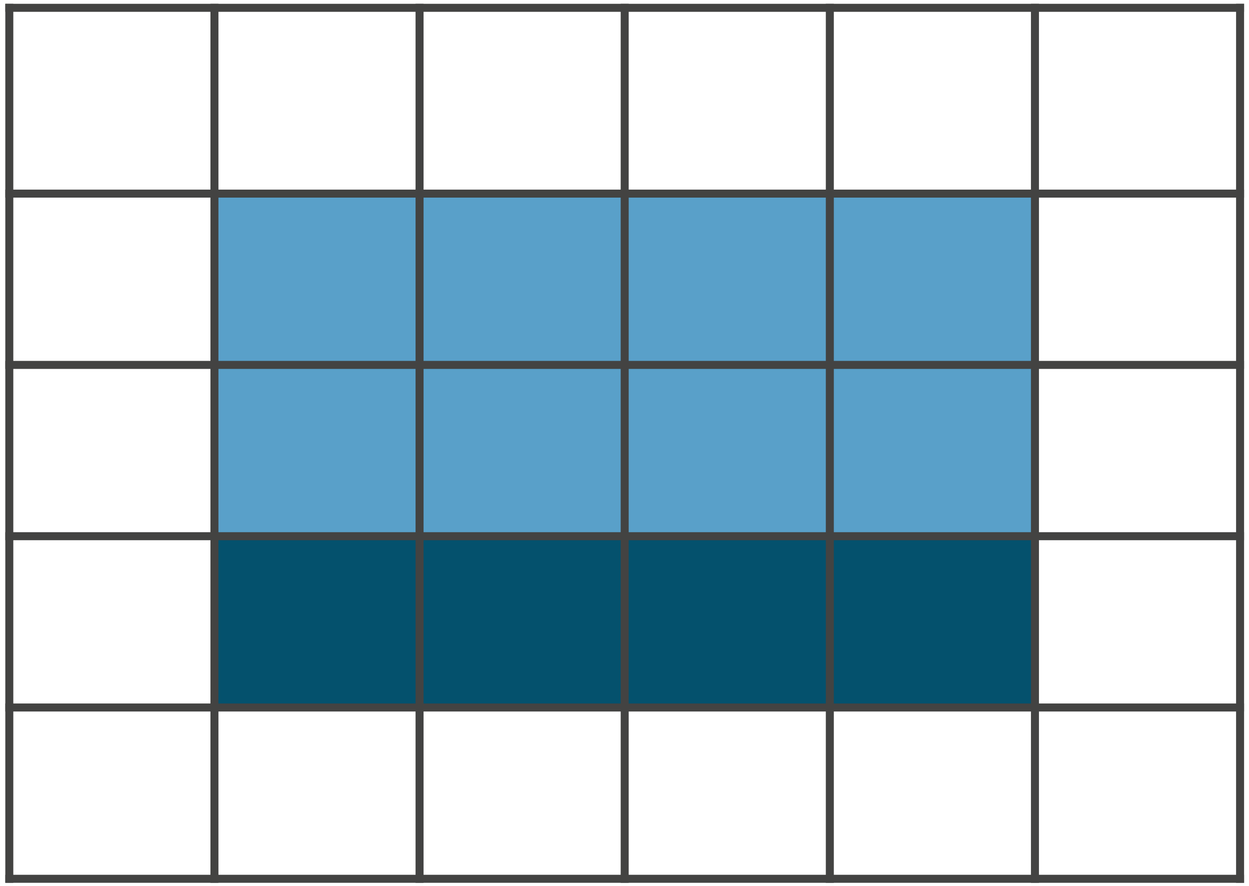
2D Real Numbers

Images as a Function

$$I\left(x,y\right):\mathbb{R}^2\rightarrow\mathbb{R}^{+n}$$

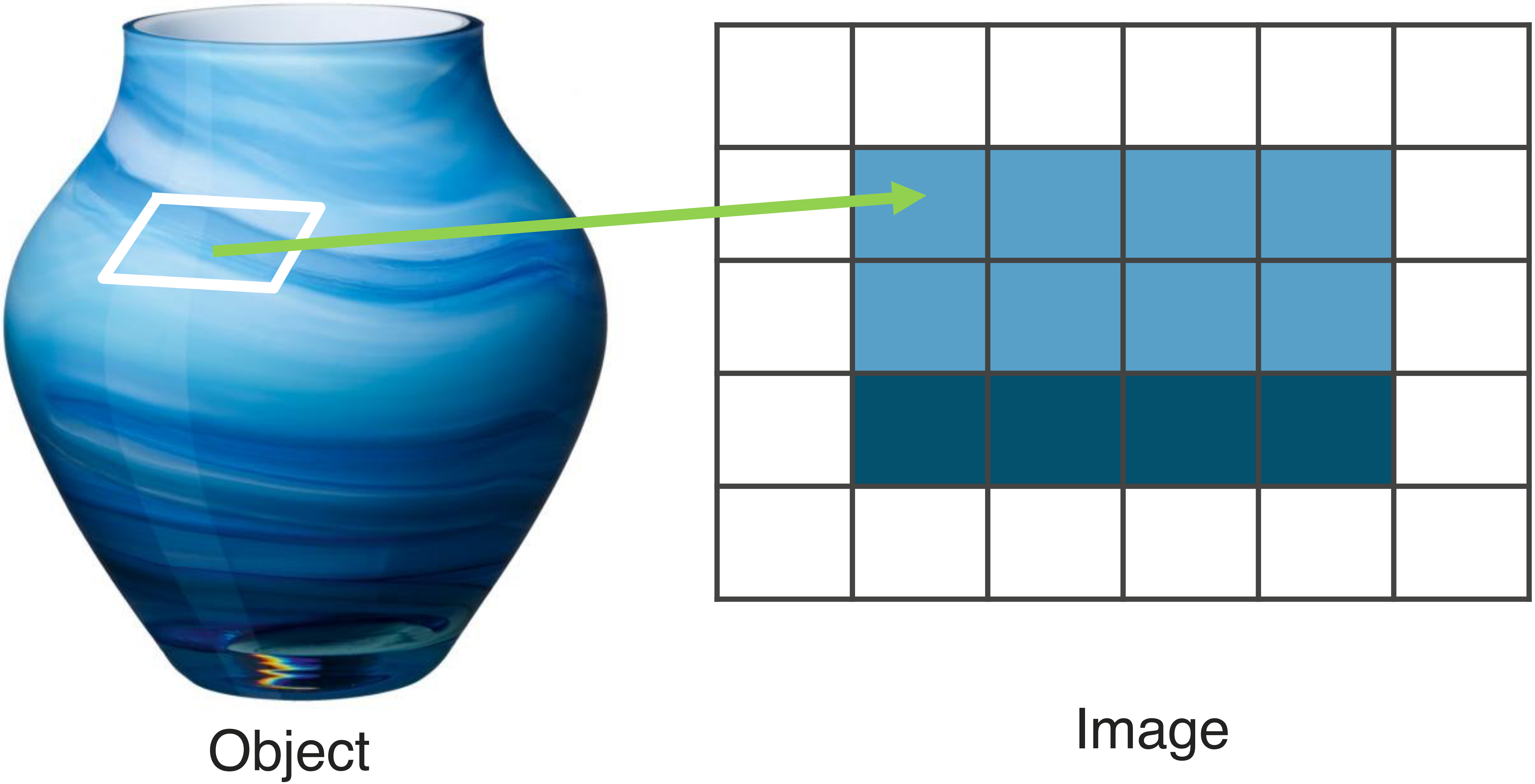
A Pixel is not a Square

Object

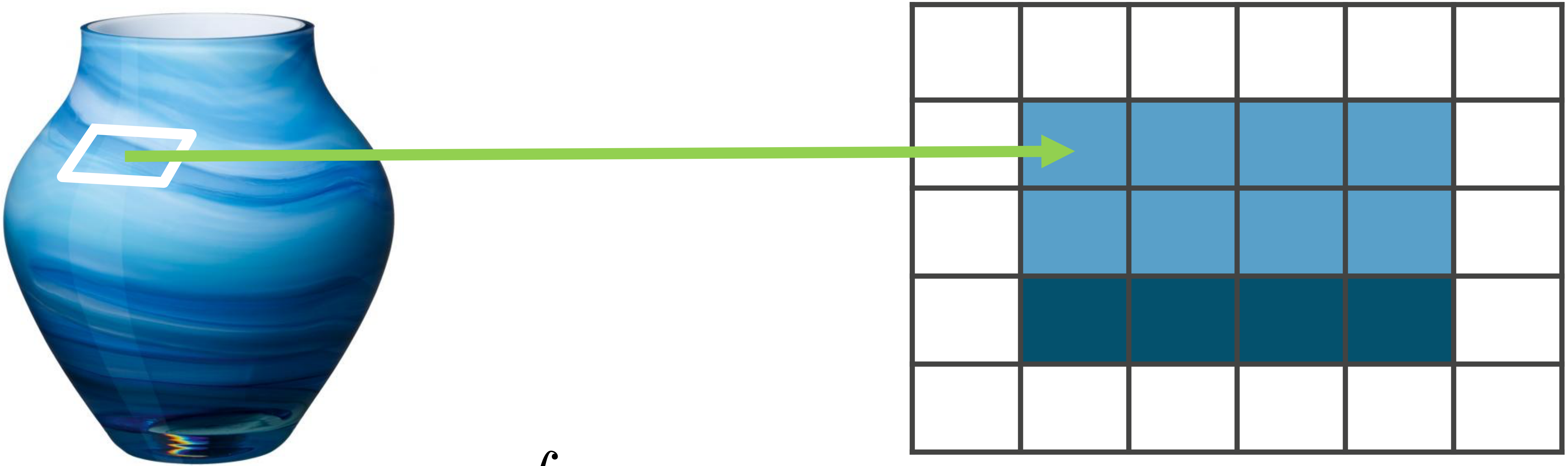


Image

A Pixel is not a Square



A Pixel is not a Square

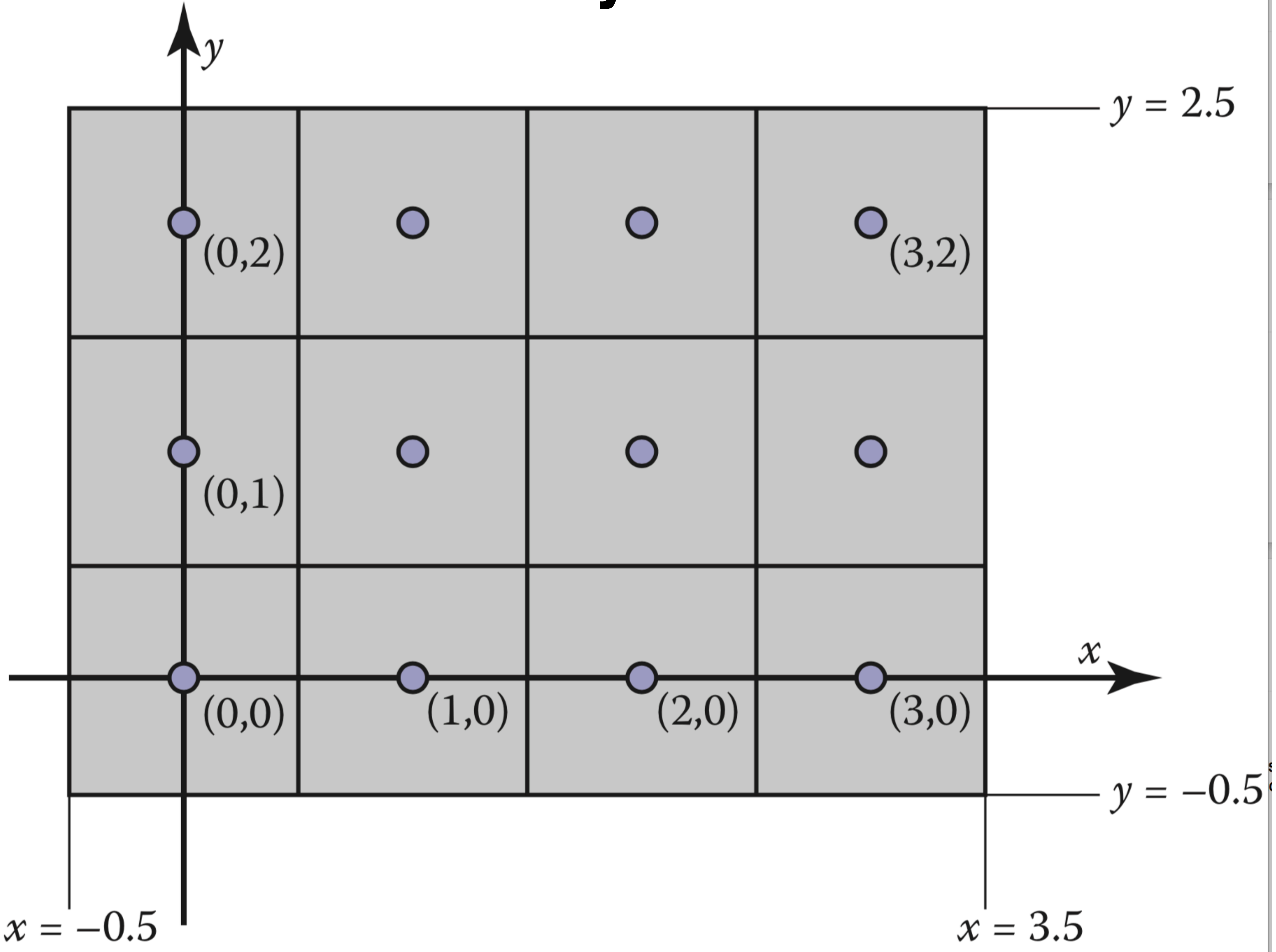


$$\text{colour} = \int_{\text{Area}} \text{Light dArea}$$

Object

Image

Standard Pixel Coordinate System

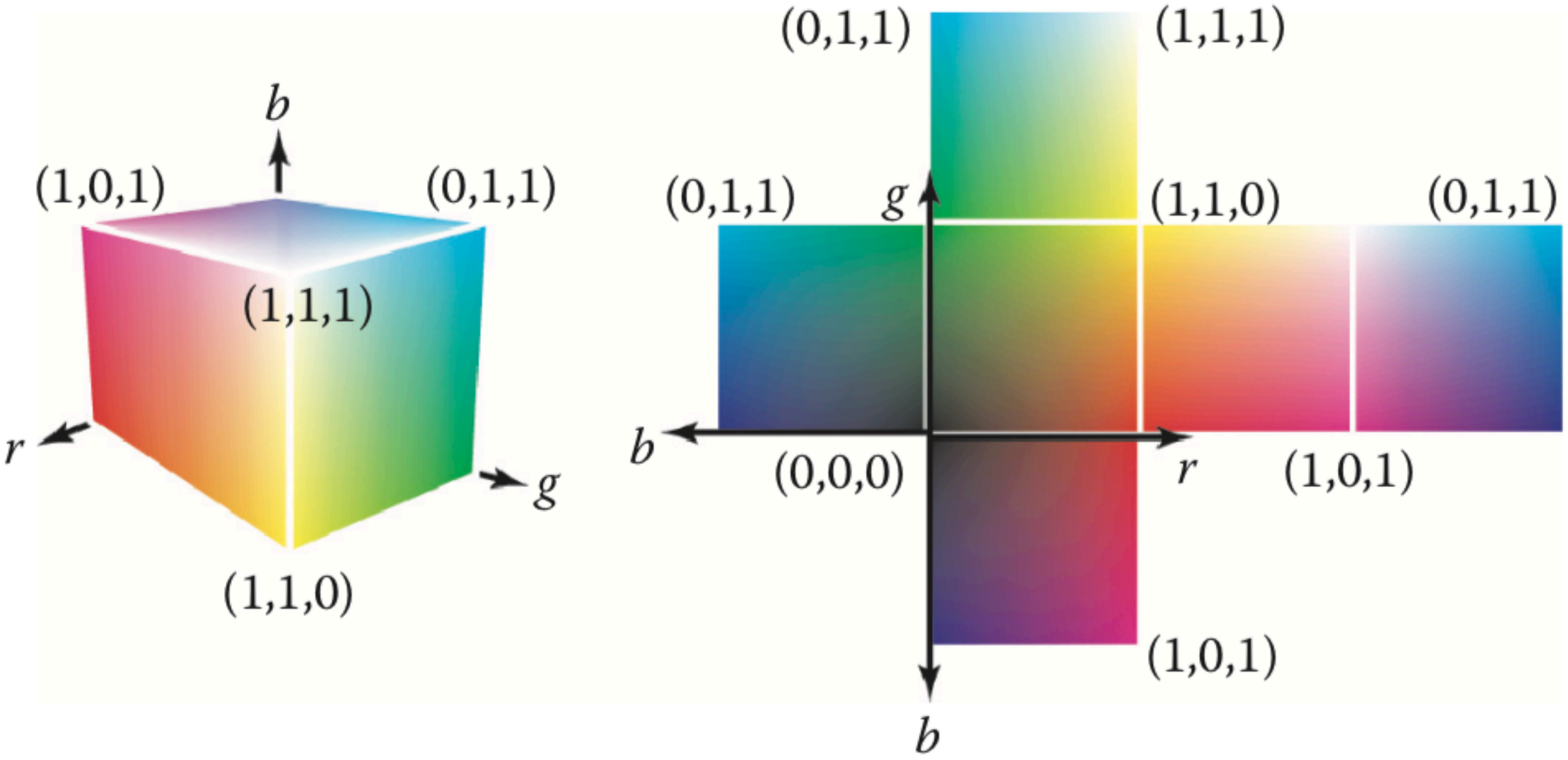


Data Types for Raster Images

Storage for 1024x1024 image (1 megapixel)

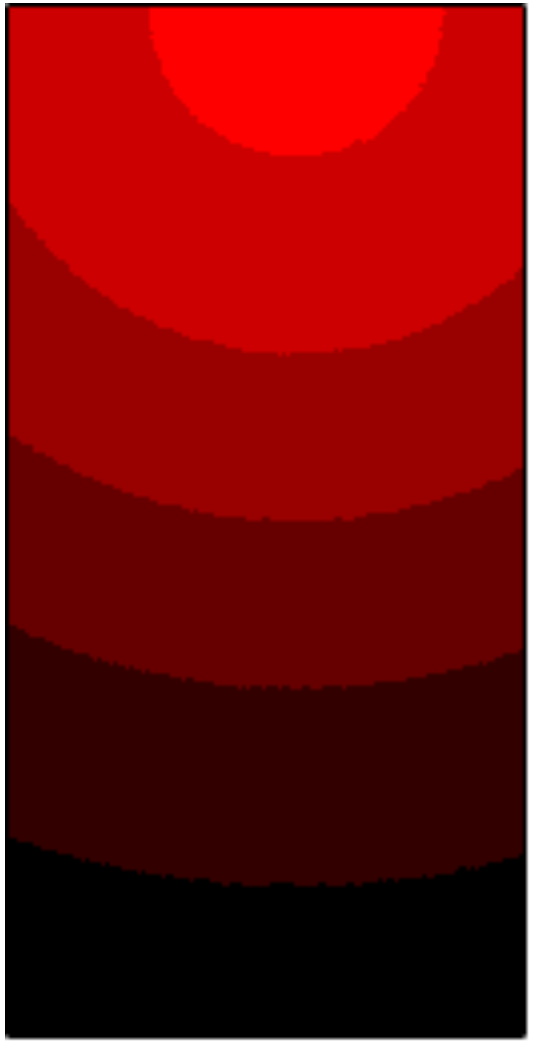
- bitmap: 128KB
- grayscale 8bpp: 1MB
- grayscale 16bpp: 2MB
- color 24bpp: 3MB
- floating-point HDR color: 12MB

RGB Images



$$
\begin{aligned}
\text{black} &= (0, 0, 0),\\
\text{red} &= (1, 0, 0),\\
\text{green} &= (0, 1, 0),\\
\text{blue} &= (0, 0, 1),\\
\text{yellow} &= (1, 1, 0),\\
\text{magenta} &= (1, 0, 1),
\end{aligned}
$$

Artifacts of Raster Images: Banding



8-bit gradient

8-bit gradient, dithered

24-bit gradient

Artifacts of Raster Images: Clipping

Original

Clipped

Gamma Correction

Display intensity is nonlinear wrt input intensity

Gamma Correction

$$\text{displayed intensity} = (\text{maximum intensity})a^{\gamma}$$

Gamma Correction

Gamma

$$\text{displayed intensity} = (\text{maximum intensity})a^{\gamma}$$

… of display

Amplitude from Image [0,1]

Gamma Correction

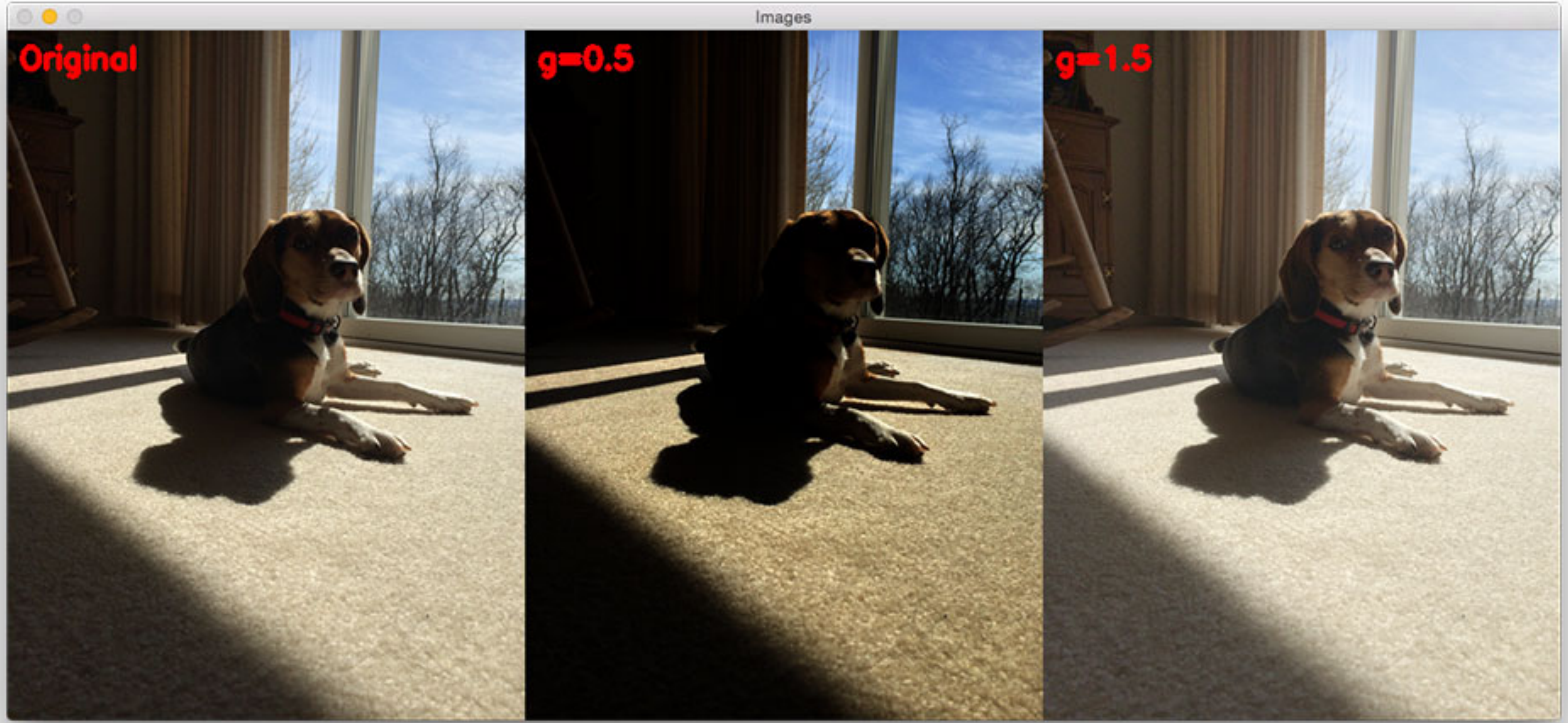
Measure: Find image amplitude that = ½ display brightness

$$0.5 = a^{\gamma}$$

Fit model

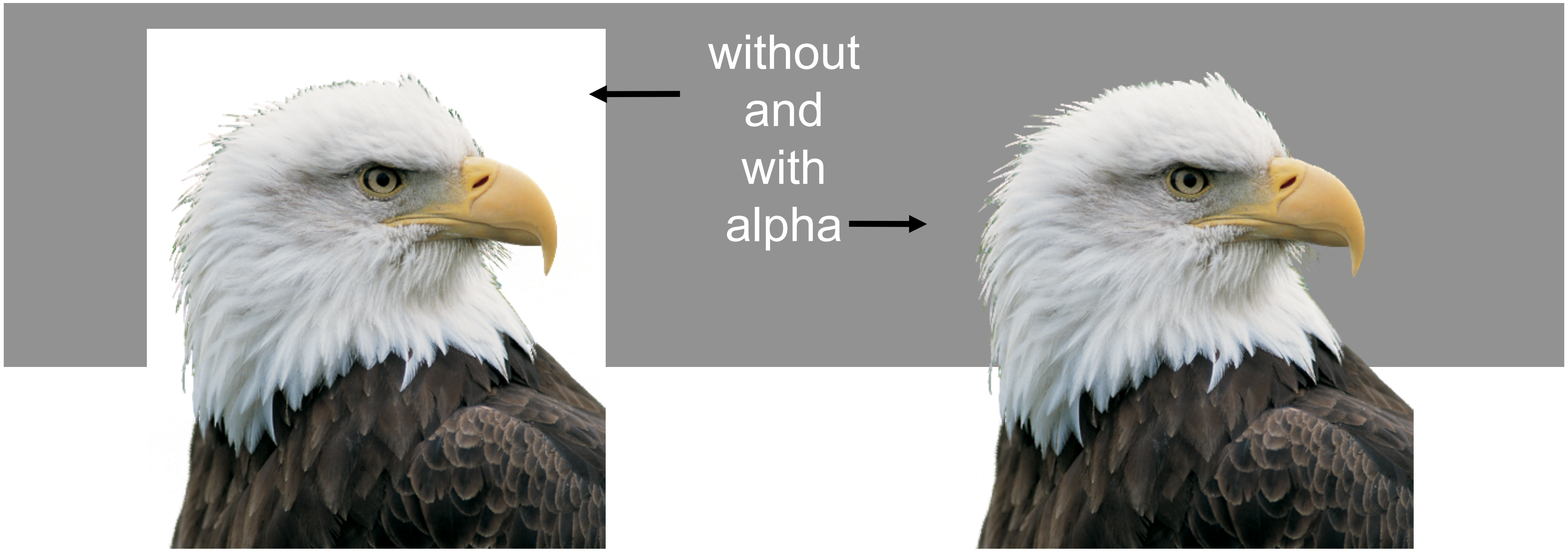
$$\gamma = \frac{\ln 0.5}{\ln a}$$

Gamma Correction



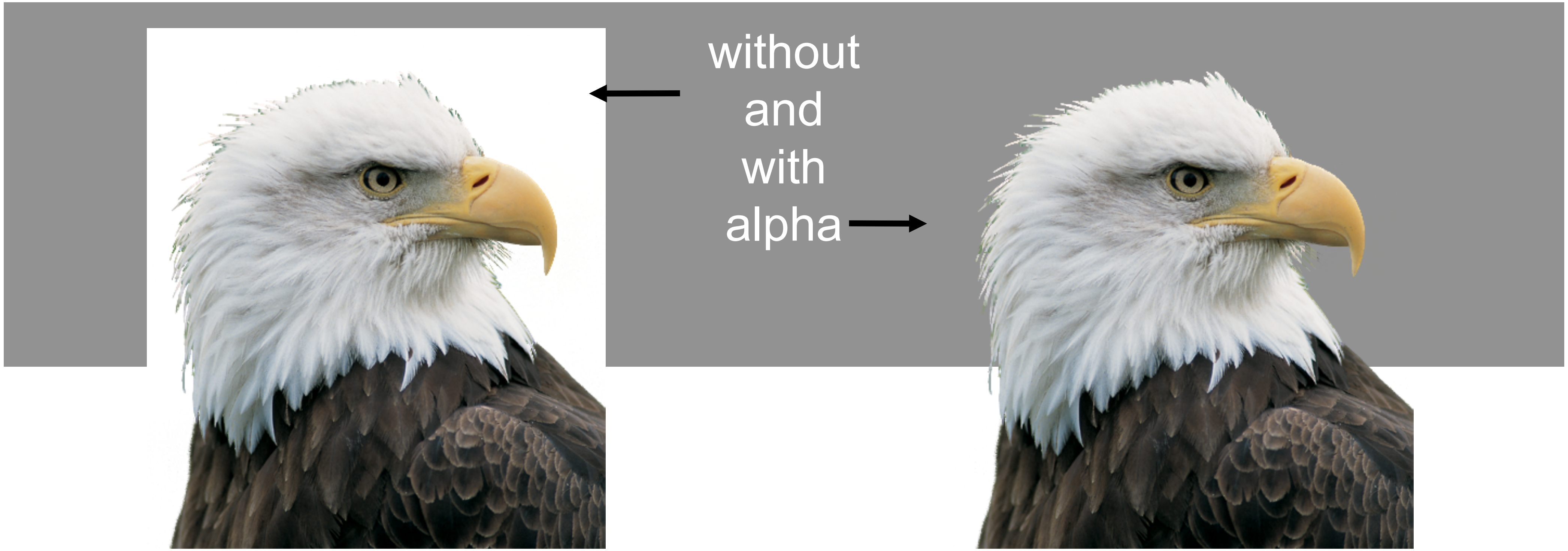
Transparency

Append (Red, Green, Blue) to be (Red, Green, Blue, Alpha)



Transparency

Append (Red, Green, Blue) to be (Red, Green, Blue, Alpha)



$$\mathbf{c} = \alpha \mathbf{c}_f + (1 - \alpha)\mathbf{c}_b.$$

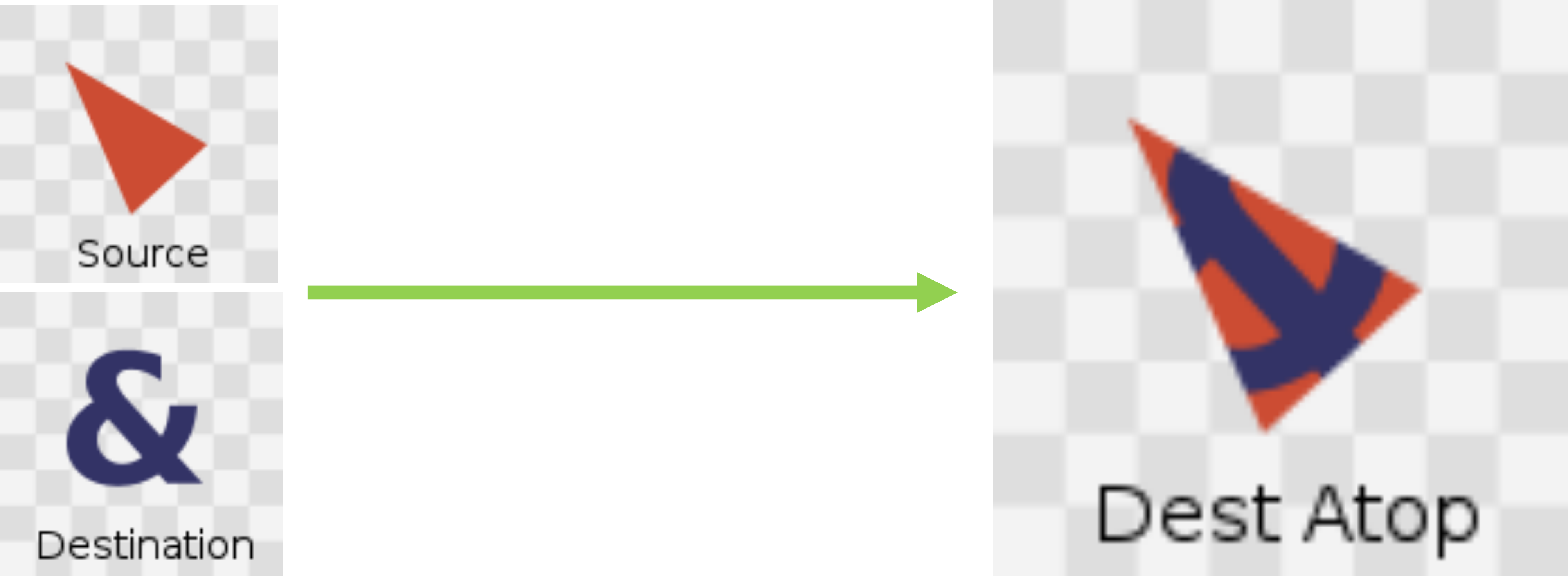
Compositing

Compositing is about layering images on top of one another

http://ssp.impulsetrain.com/porterduff.html

Compositing

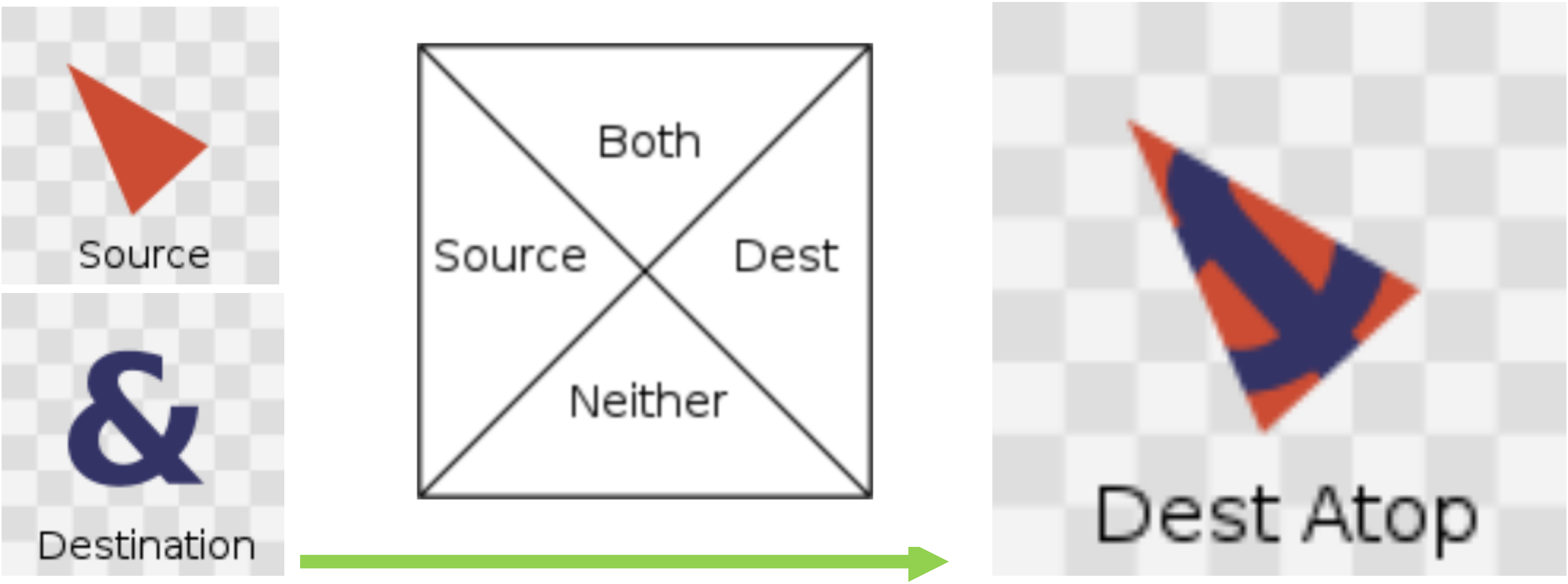
Compositing is about layering images on top of one another



http://ssp.impulsetrain.com/porterduff.html

Compositing

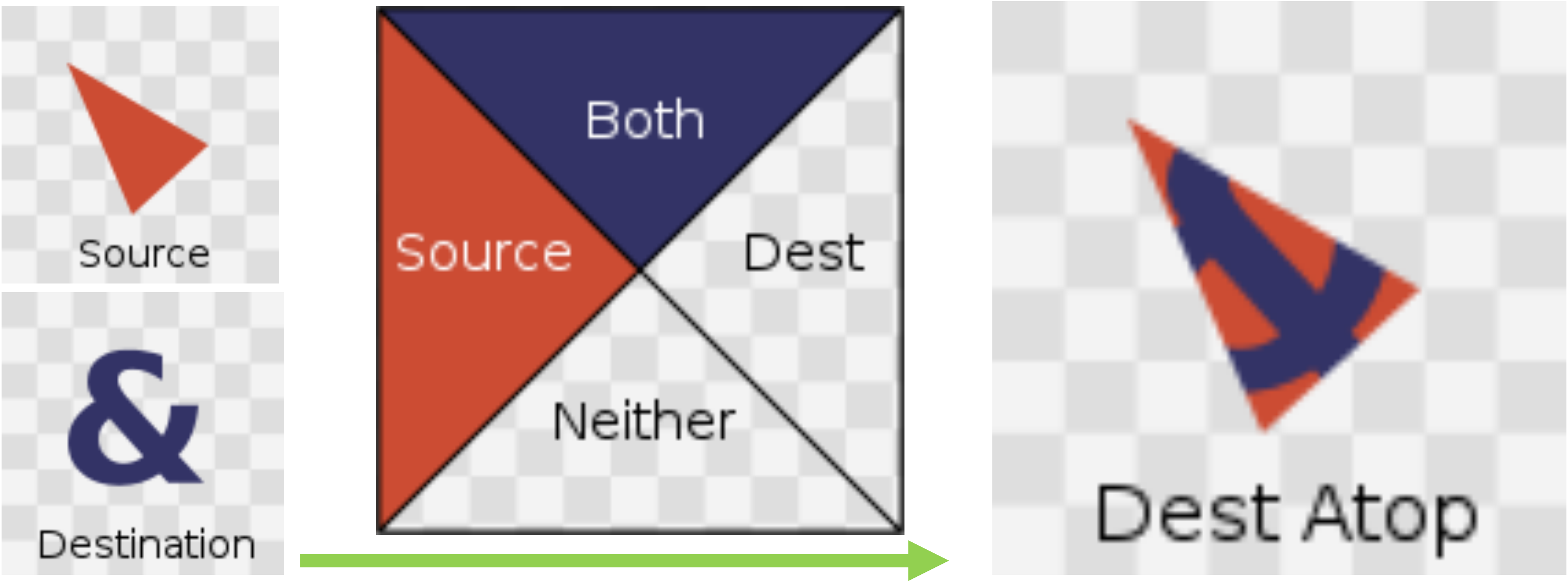
Compositing is about layering images on top of one another



http://ssp.impulsetrain.com/porterduff.html

Compositing

Compositing is about layering images on top of one another



http://ssp.impulsetrain.com/porterduff.html

Compositing

Compositing is about layering images on top of one another

$$A_{\text{src}} \cdot [s] + A_{\text{dest}} \cdot [d] + A_{\text{both}} \cdot [b]$$

http://ssp.impulsetrain.com/porterduff.html

Compositing

Compositing is about layering images on top of one another

$$A_{\text{src}} \cdot [s] + A_{\text{dest}} \cdot [d] + A_{\text{both}} \cdot [b]$$

$$A_{\text{src}} = \alpha_{\text{s}} \cdot (1 - \alpha_{\text{d}})$$
$$A_{\text{dst}} = \alpha_{\text{d}} \cdot (1 - \alpha_{\text{s}})$$
$$A_{\text{both}} = \alpha_{\text{s}} \cdot \alpha_{\text{d}}$$

http://ssp.impulsetrain.com/porterduff.html

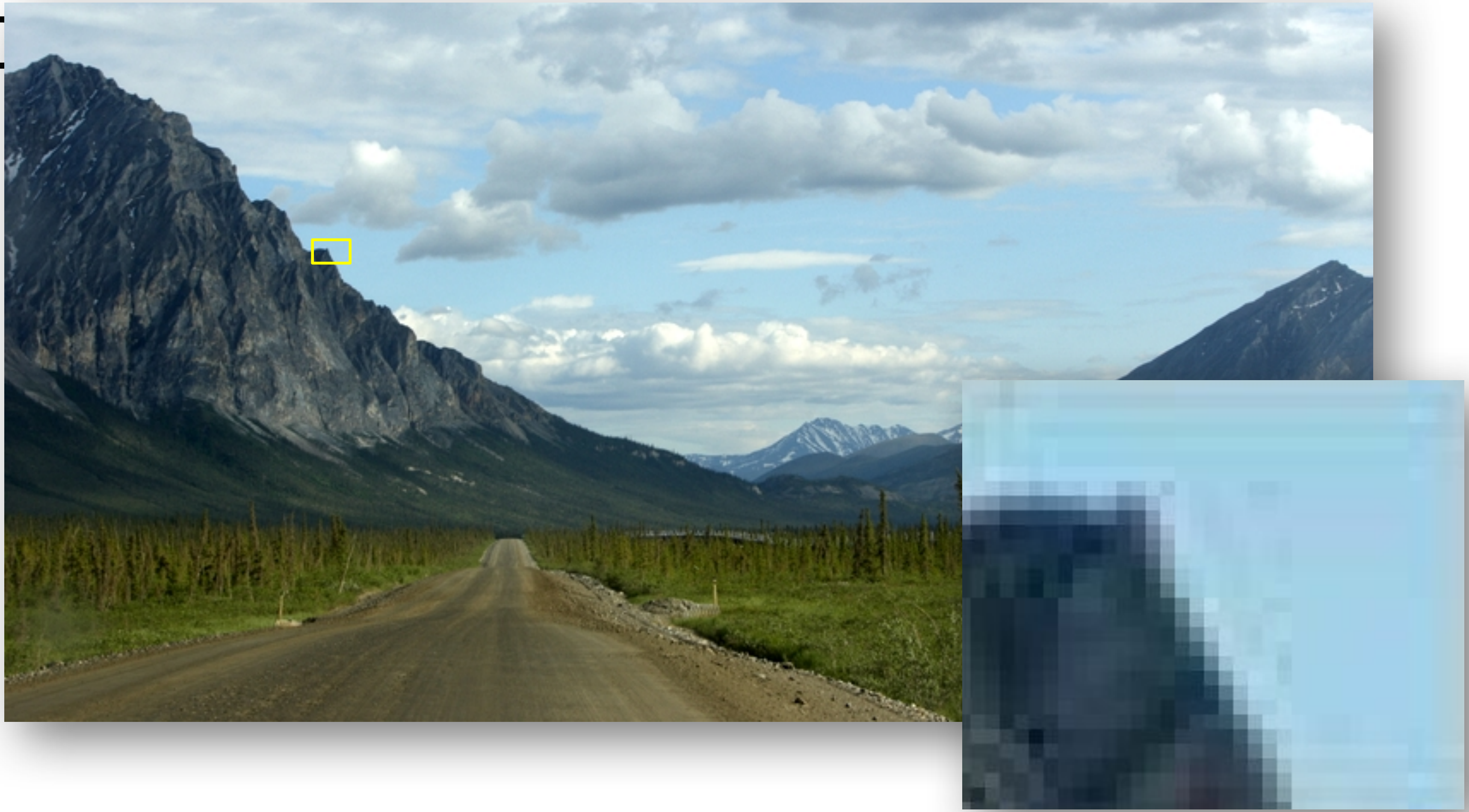
Assignment 1 Available Right Now

DUE IN TWO WEEKS (START EARLY!)

Next Week: Ray Casting

What is an Image?

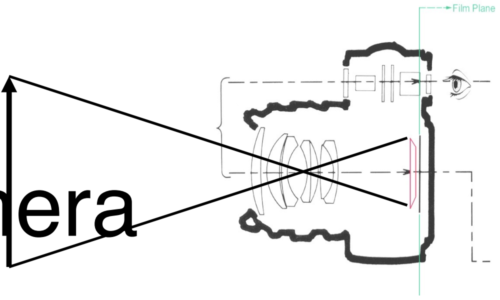
Image = distribution of light energy on 2D "film"

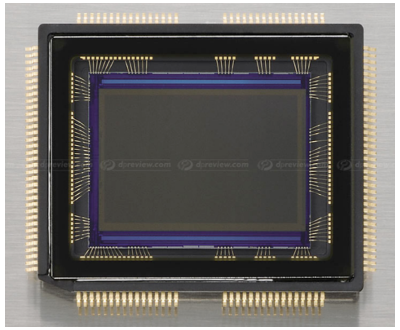


Raster Devices

Input (scanners, cameras)

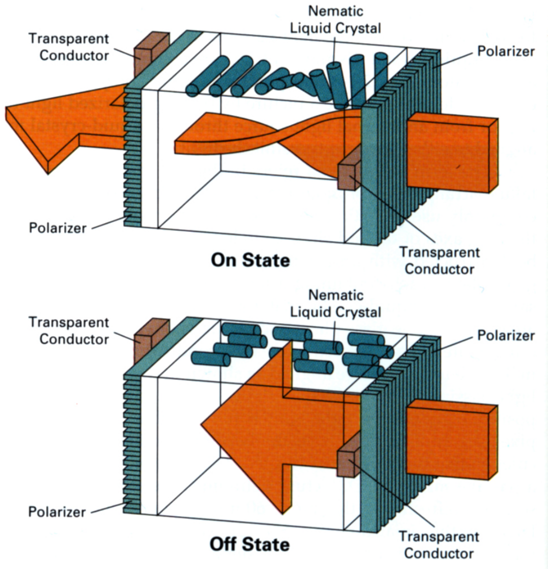
2D array sensor: digital camera





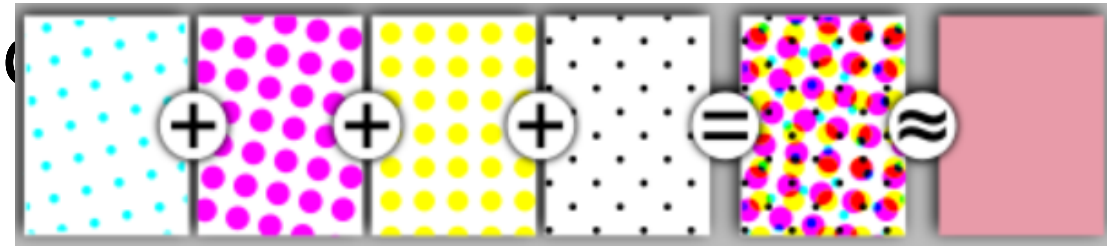
Output (printers, displays)

Emissive: light-emitting diod



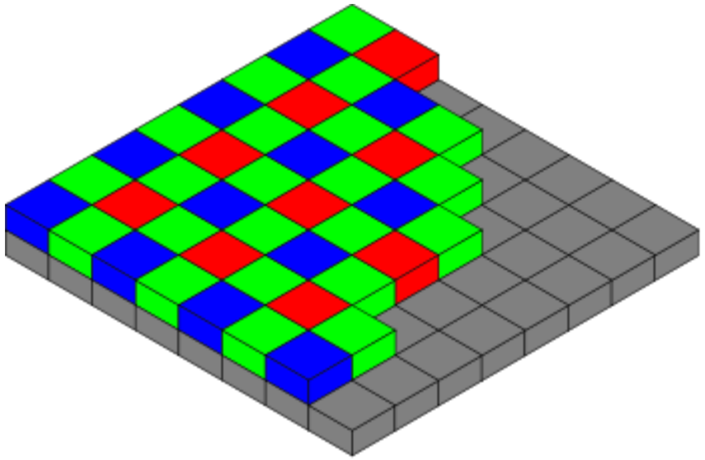
Transmissive: liquid crystal (

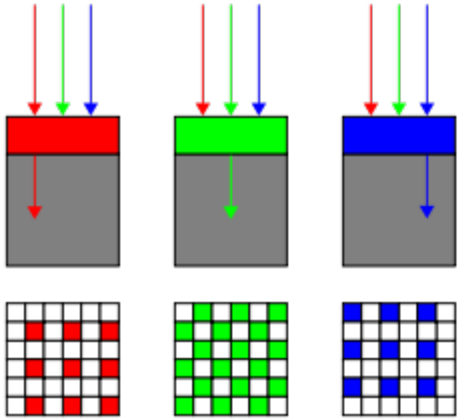
(LCD)



Camera Sensor Array

Bayer color mosaic





Raster image representation

All these devices suggest 2D arrays of numbers

Bitmaps: boolean per pixel (1 bpp):

- interp. = black and white; e.g. fax

Grayscale: integer per pixel:

- interp. = shades of gray; e.g. black-and-white print
- precision: usually byte (8 bpp); sometimes 10, 12, or 16 bpp

Color: 3 integers per pixel:

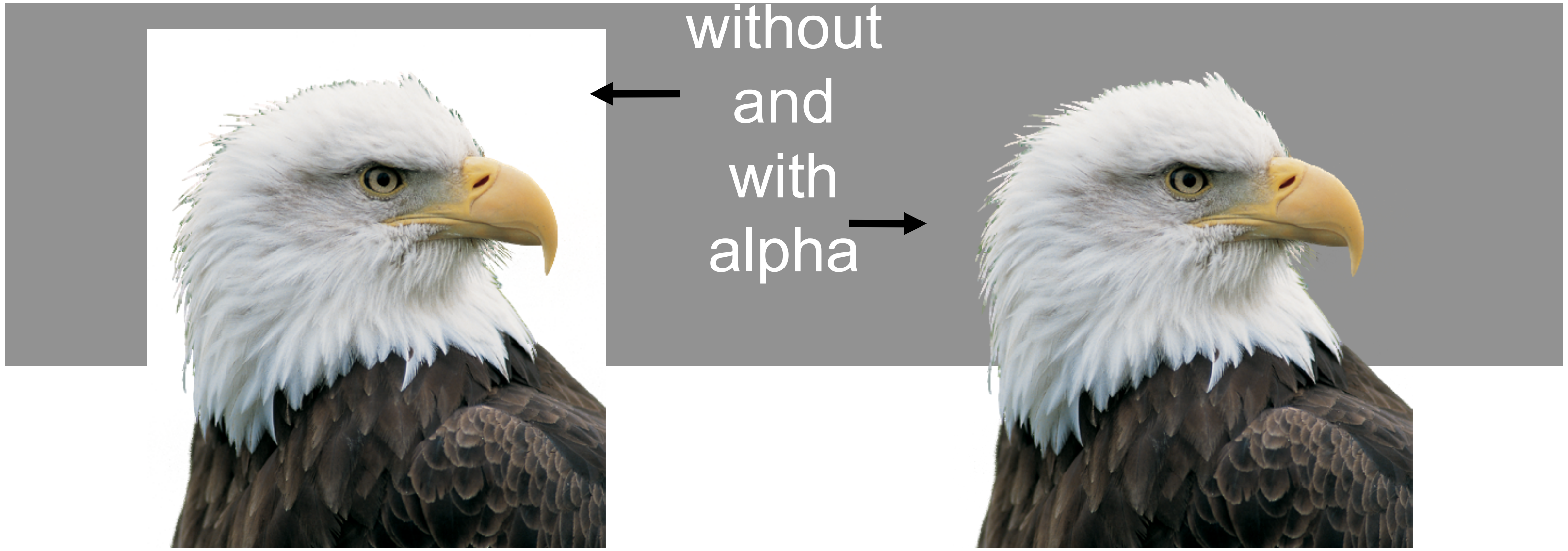
- interp. = full range of displayable color; e.g. color print
- precision: usually byte[3] (24 bpp)
- sometimes 16 (5+6+5) or 30 or 36 or 48 bpp

Floating point:

- more abstract, because no output device has infinite range
- provides *high dynamic range* (HDR)
- represent real scenes independent of display
- becoming the standard intermediate format in graphics processor

Datatypes for raster images

- Transparency (add *alpha* channel)



- Storage for 1024x1024 image (1 megapixel)
  - bitmap: 128KB
  - grayscale 8bpp: 1MB
  - grayscale 16bpp: 2MB
  - color 24bpp: 3MB
  - floating-point HDR color: 12MB

Converting pixel formats

Color to gray

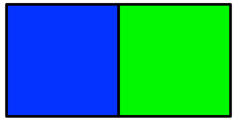
- could take one channel (blue, say)

  leads to odd choices of gray value

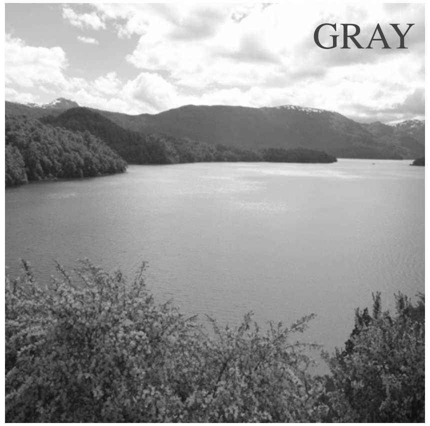
- combination of channels is better

  but different colors contribute differently to lightness

  which is lighter, full blue or full green?

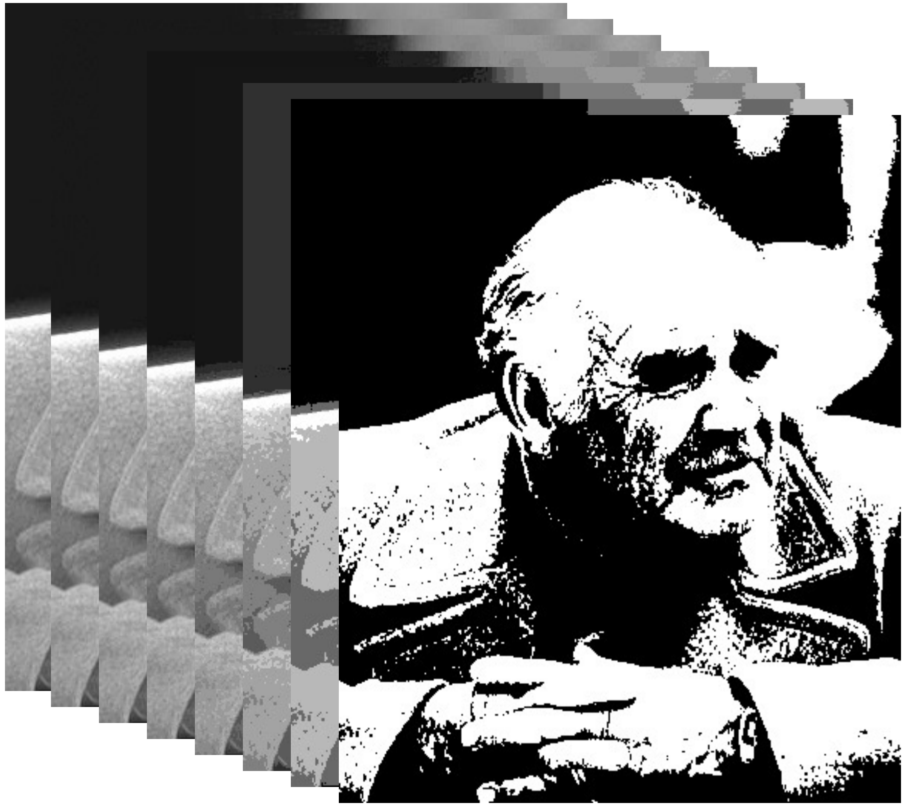


good choice: gray = 0.2 R + 0.7 G + 0.1 B



Converting pixel precision

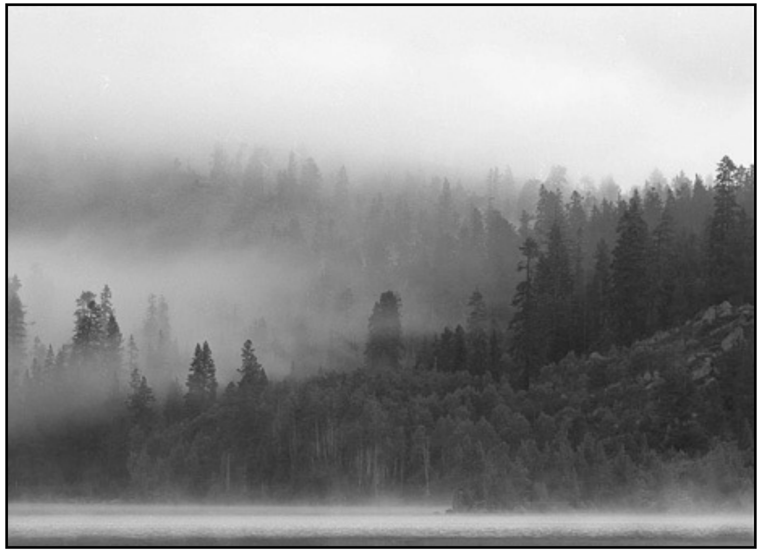
Up is easy, down loses information…

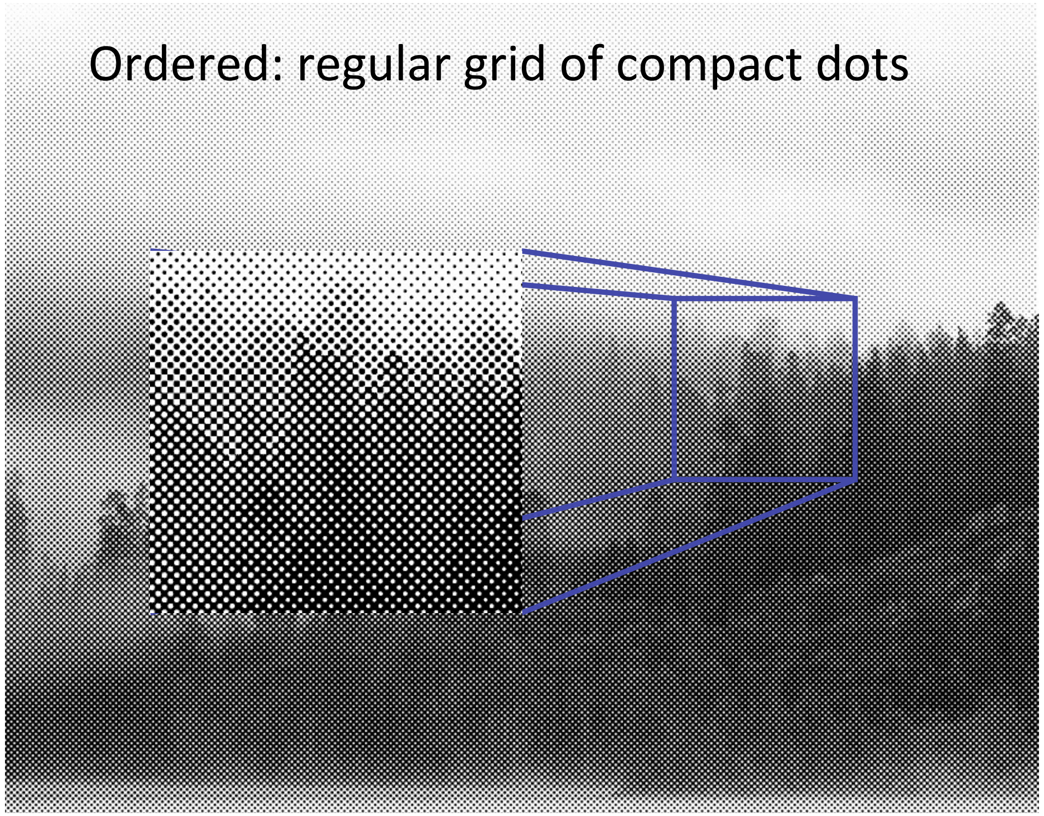


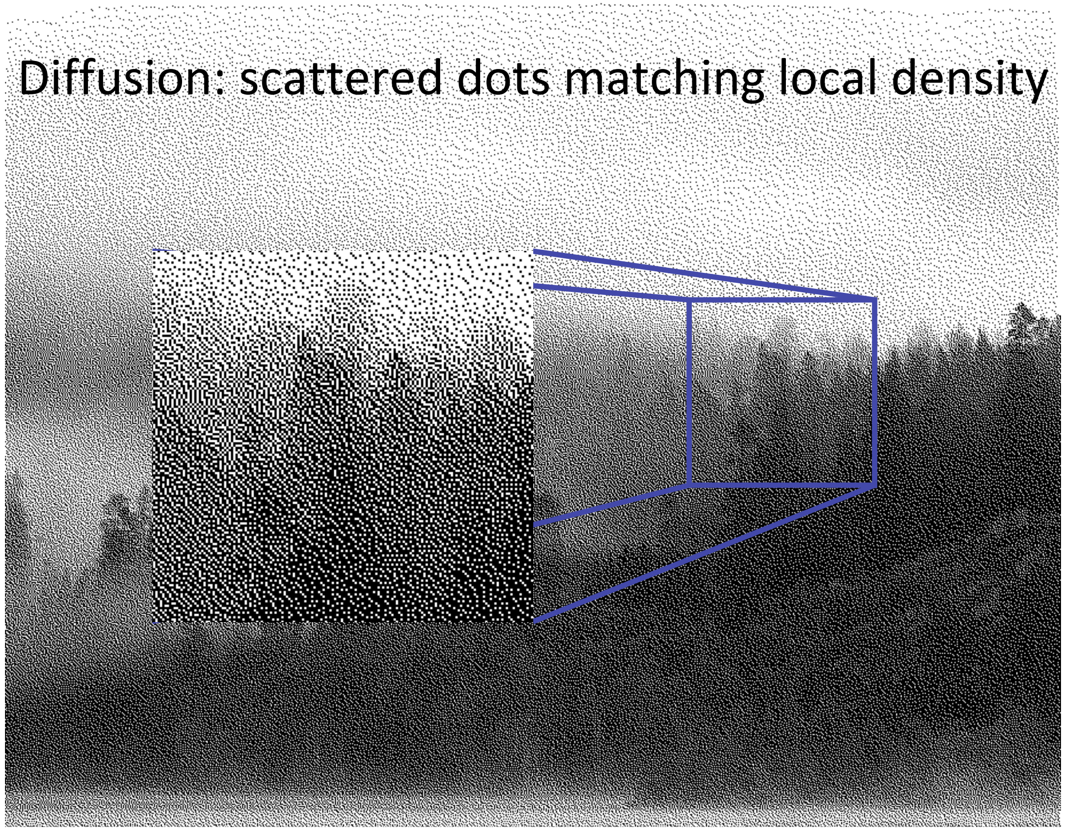
Dithering

Decreasing bpp => quantize/threshold

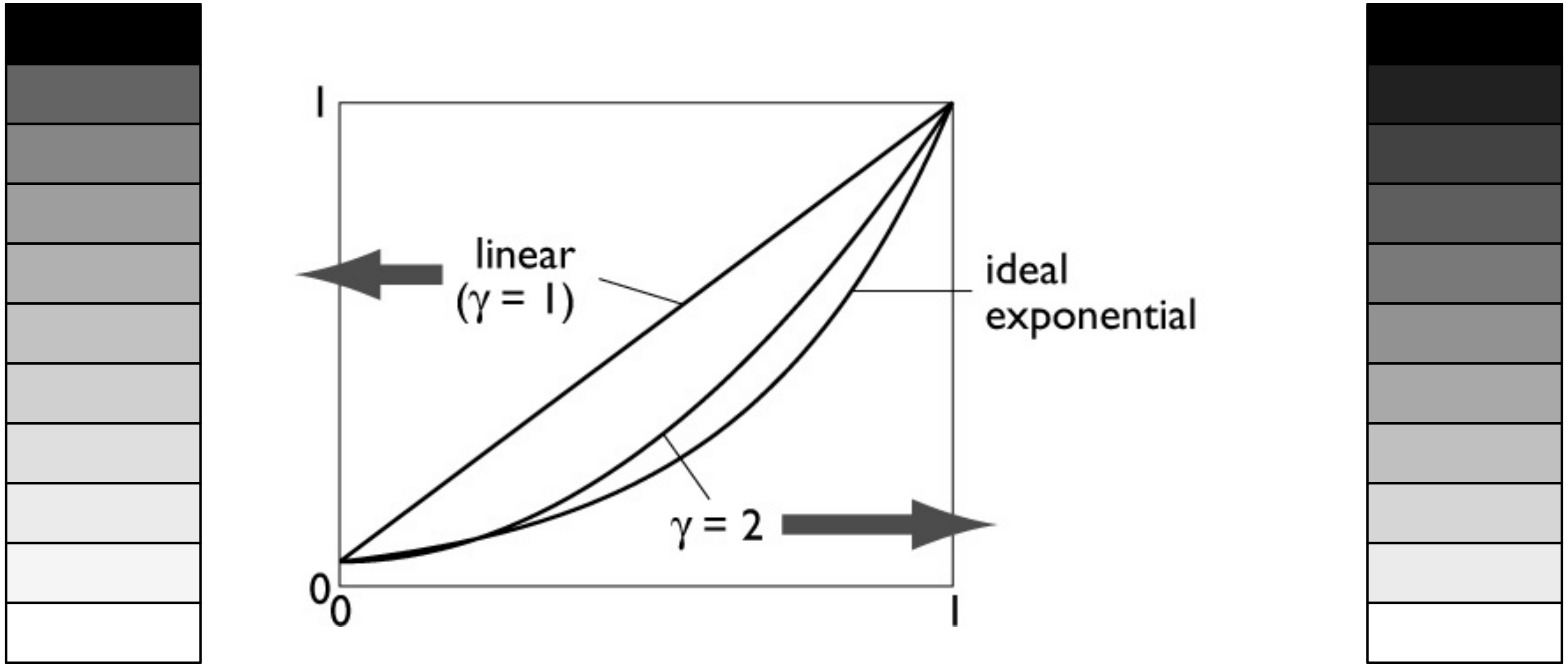
- Consistent quantization causes banding.
- Trade spatial for tonal resolution: Dither.







Gamma correction

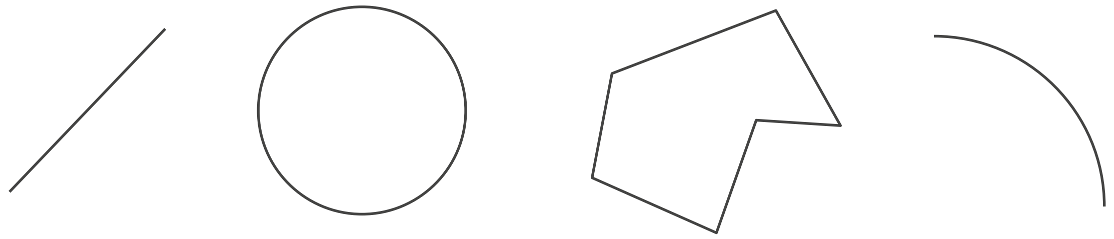


Topic 3.

Scan Conversion

2D Drawing

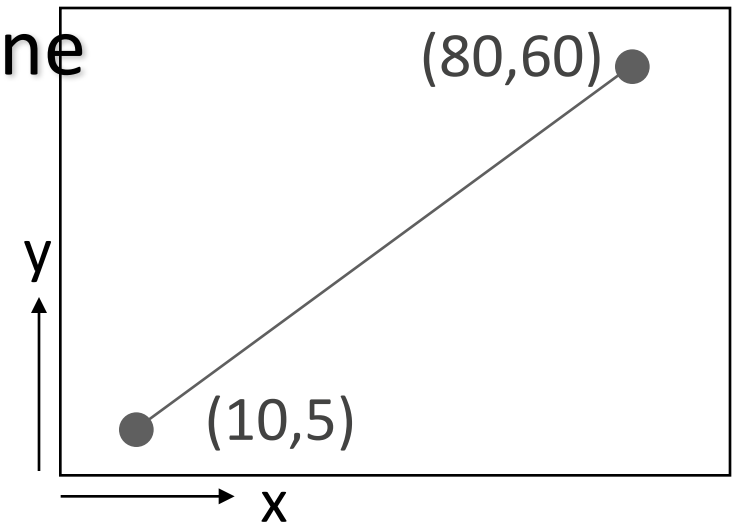
Common geometric objects:



When drawing a picture, 2D geometric shapes are specified as if they are drawn on a continuous plane

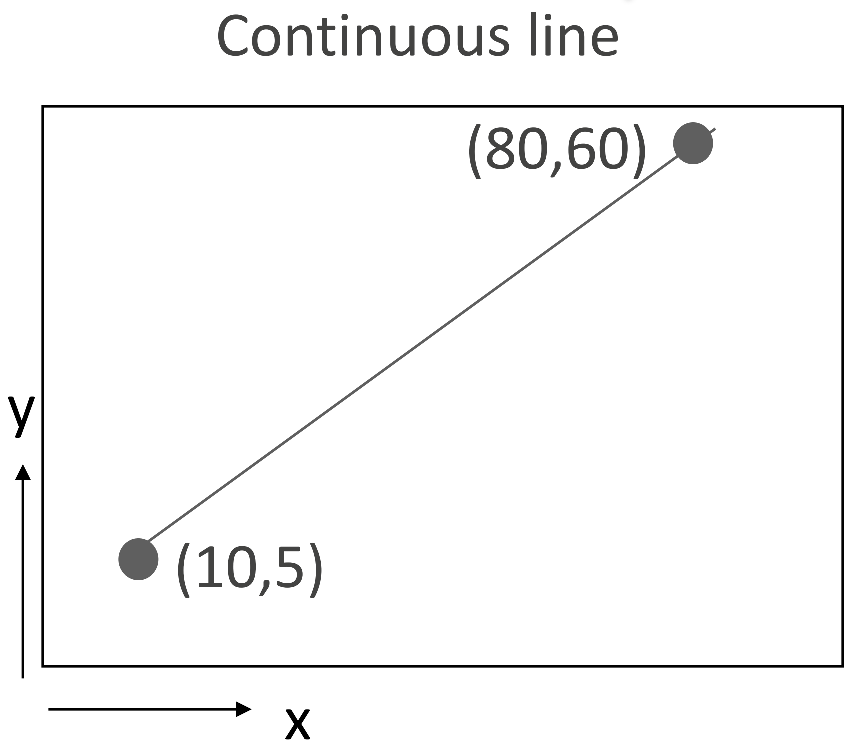
Drawing command:

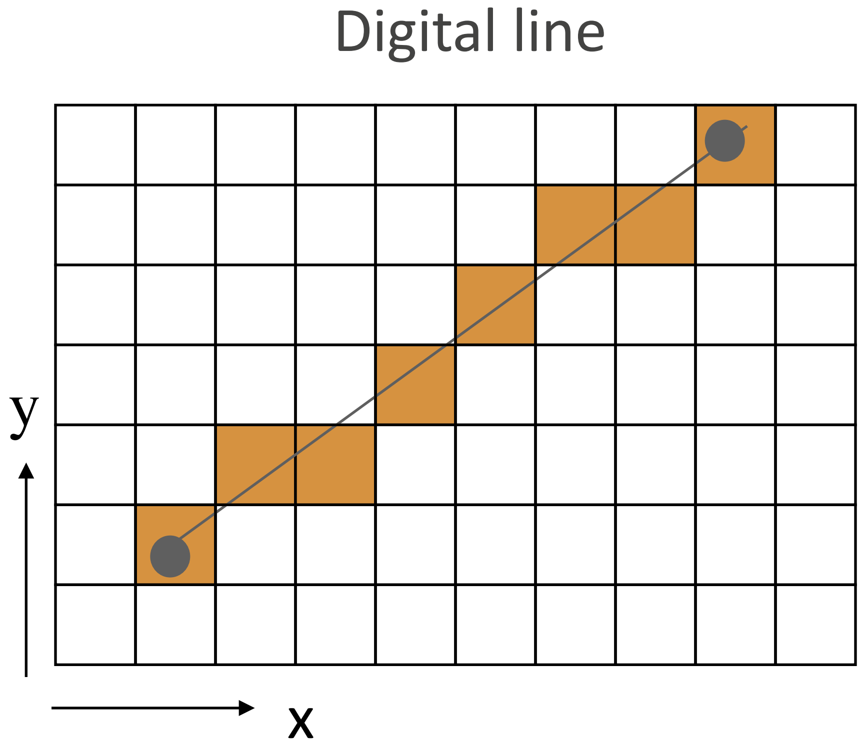
Draw a line from point (10,5) to point (80,60)



2D Drawing

In reality, computer displays are arrays of pixels, not abstract mathematical continuous planes





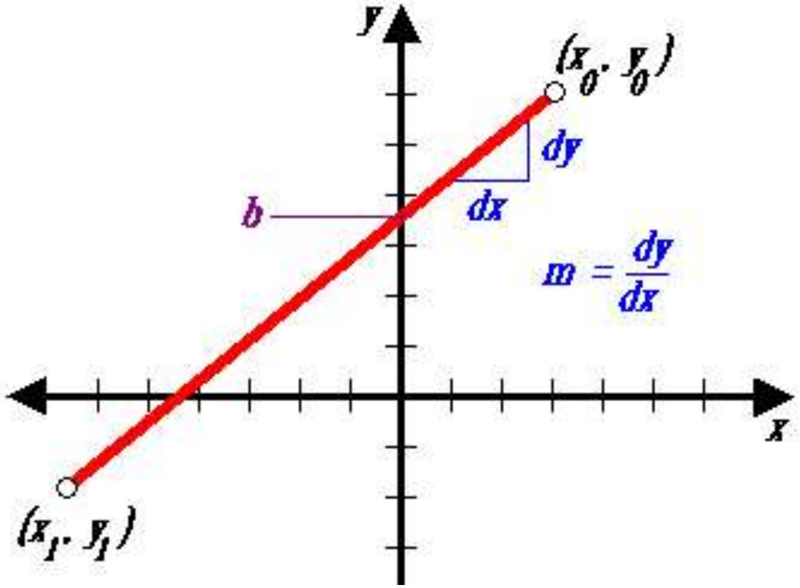
Equation of a Line

Line between (x_0, y_0) *and* (x_1, y_1)

$dx = x_1 - x_0$, $dy = y_1 - y_0$

Explicit : $y = mx + b$

$m = dy/dx$, $b = y_0$ ·



Parametric :

$$x(t) = x_0 + dx^*t$$

$$y(t) = y_0 + dy^*t$$

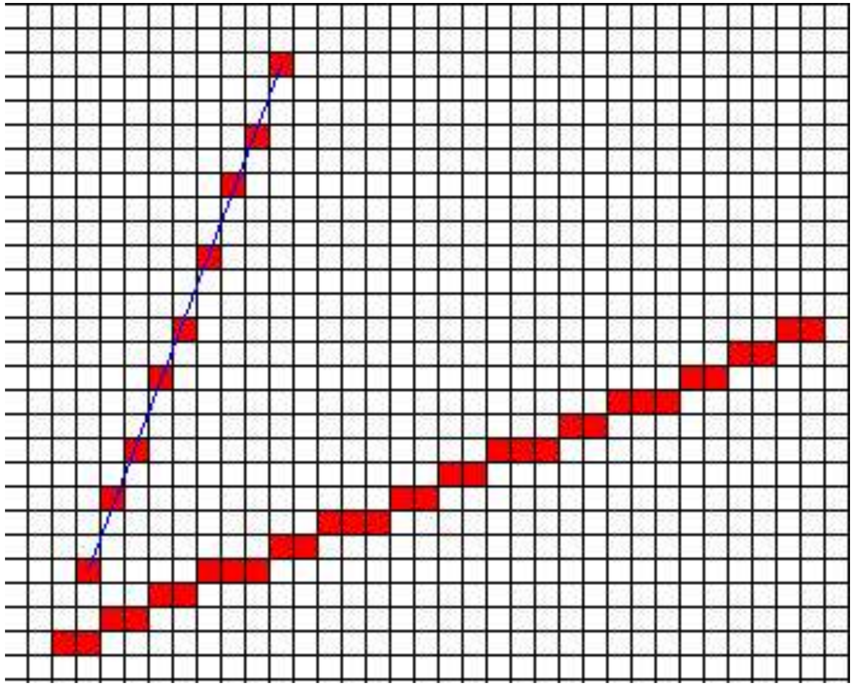
$$P = P_0 + (P_1 - P_0)^*t$$

DDA Algorithm

Explicit form:

y= dy/dx * (x-x0) + y0

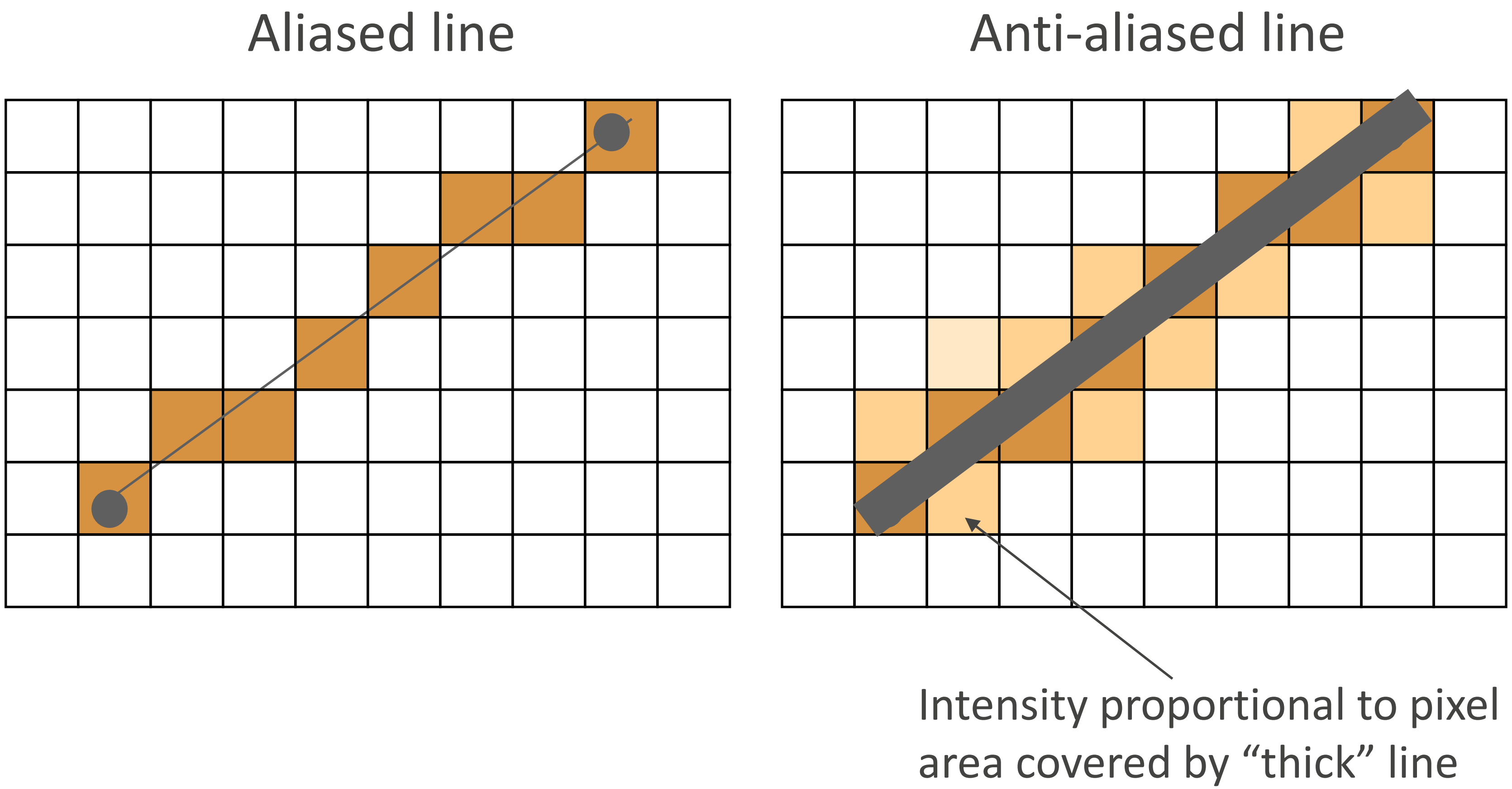
```
dx = x1-x0; dy = y1 – y0;
m = dy/dx;
for ( x=0; x<=x1-x0; x++)
{
     setpixel (x+x0, round(m*x+y0));
}
```



Anti-Aliasing

Raster line drawing can make look jaggy!

How can we make a digital line appear less jaggy?



Rasterization or Scan Conversion of triangles

Rasterize horizontal span of pixels between a pair of triangle edges.

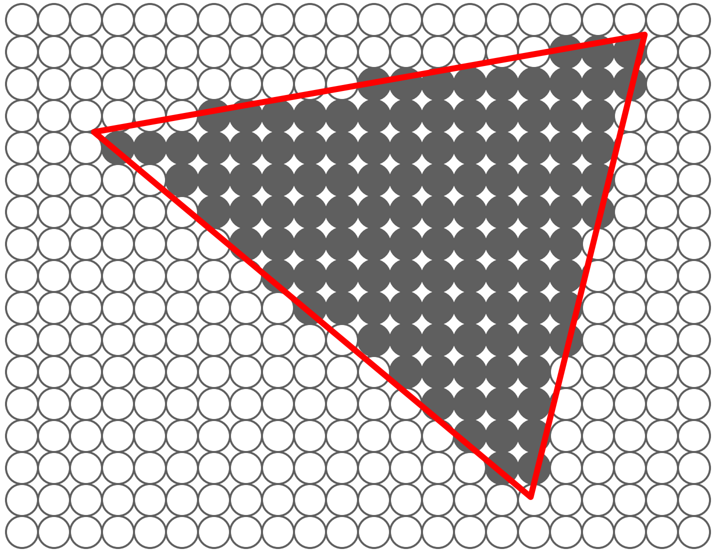


Image making algorithms

Object-Order

for-each ***object***

update the ***pixels*** the ***object*** influences;

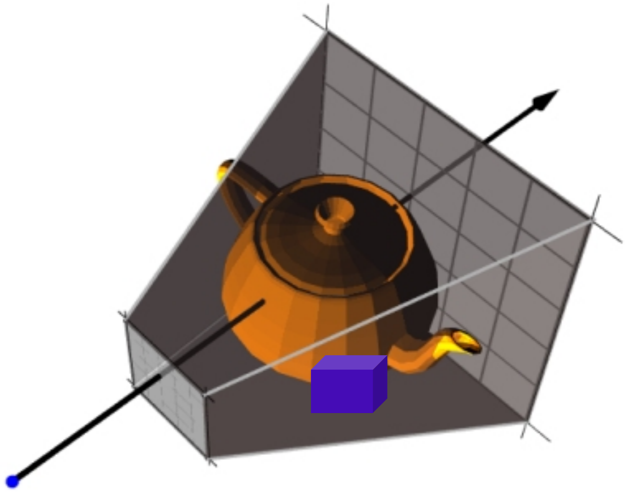


Image-Order

for-each ***pixel***

set the ***pixel*** based on the ***objects*** that influence it;