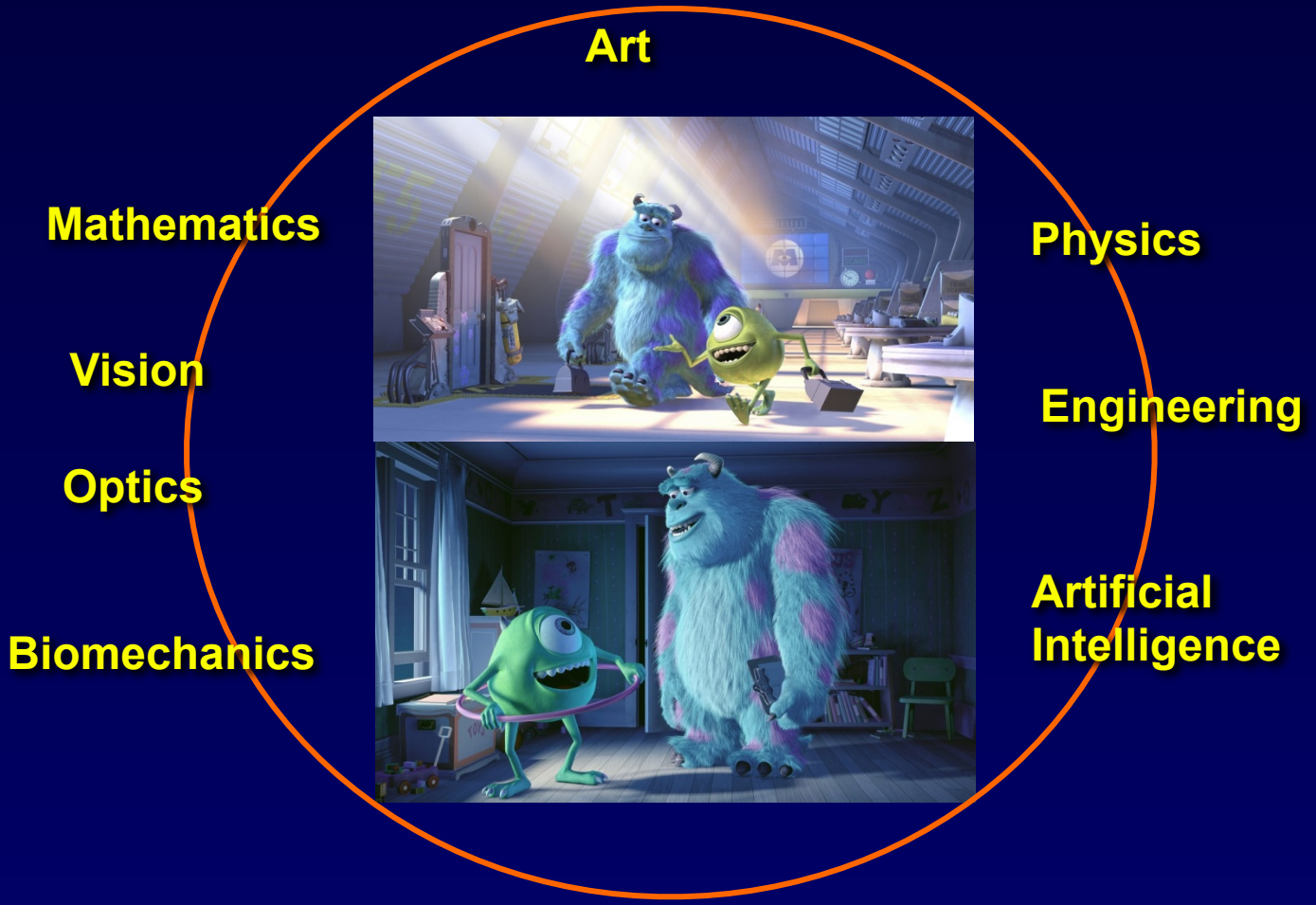
IT 360 – Fall 2023
Programming for Computer Graphics

Colors, Image Processing

Instructor: Tomer Weiss

Recap: Computer Graphics

The Art and Science of creating imagery by computer



Applications of CG

Entertainment

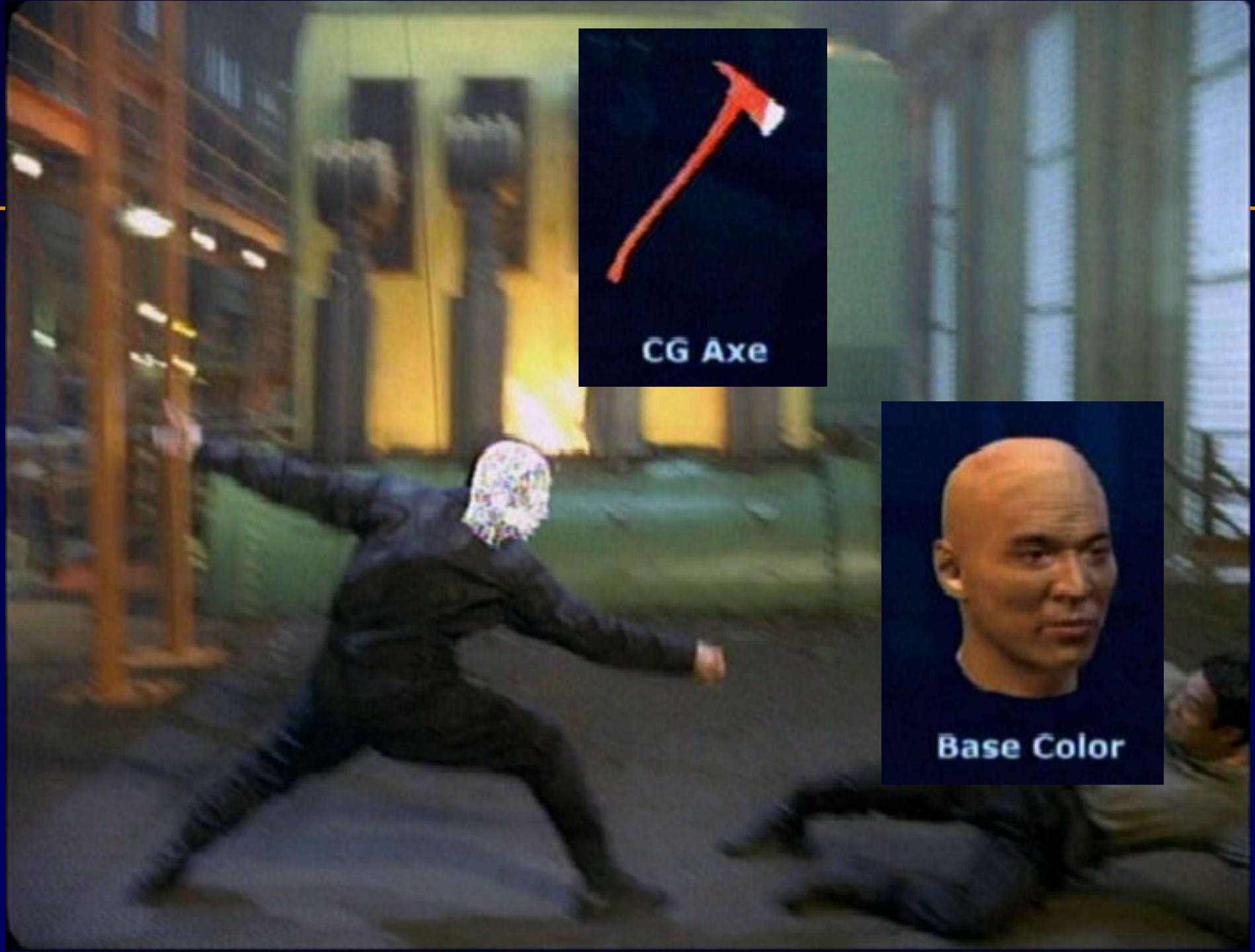
- Films
- Computer Games
- Virtual reality

Visualization

- Scientific visualization
- Medical visualization
- Flight simulation
- Architecture

Education, etc.

Digital Compositing

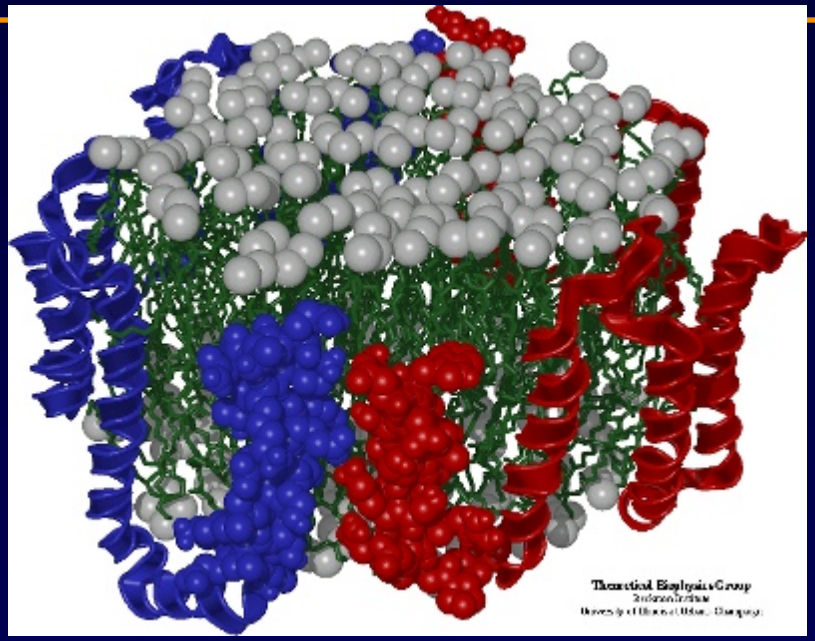

CG Axe
Base Color

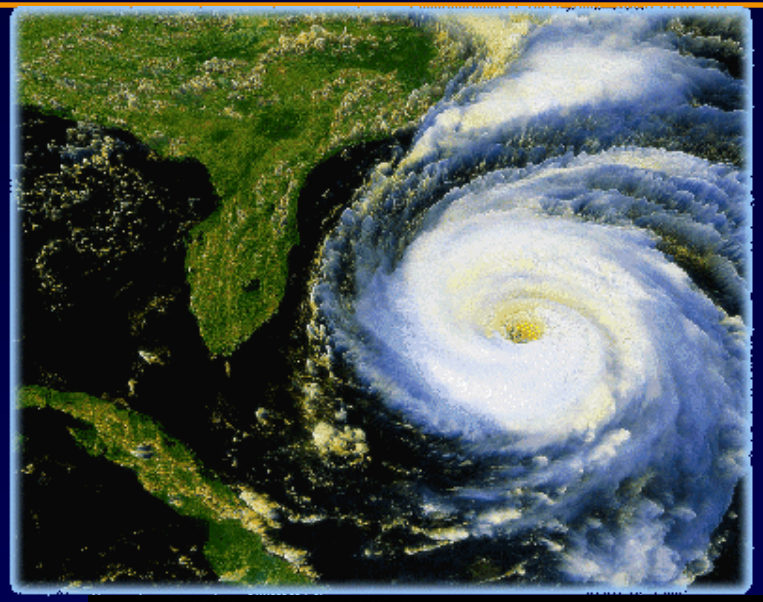
Computer-Aided Design

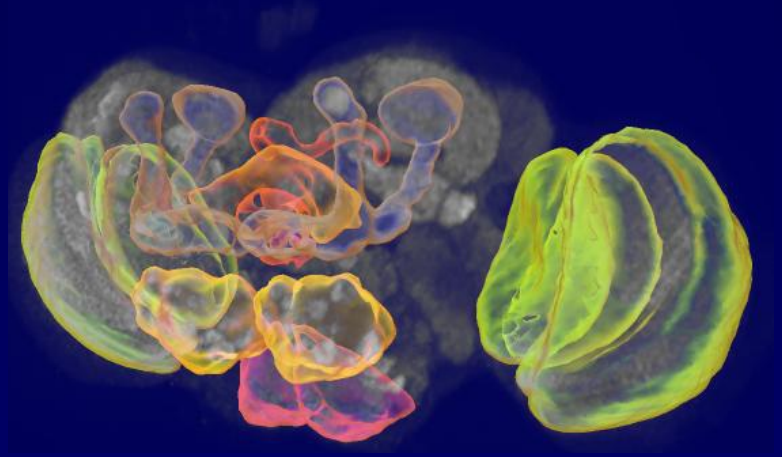
Precision modeling

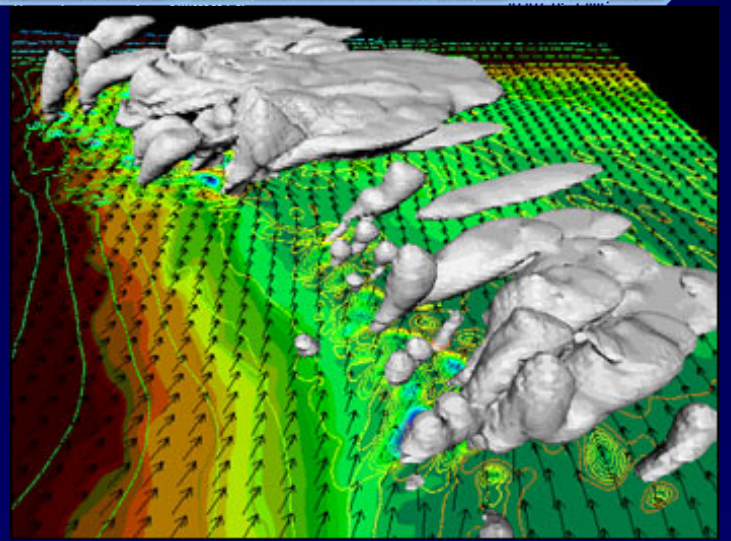
Engineering visualizatio

Visualization: Scientific









What Makes Graphics Fun?

Very interdisciplinary

- CS, Math, Physics, Art, Perception

Helps you understand why the worlds looks the way it does

You can "see" the result

You can "cheat" with reality as long as it looks good

What are we going to do

Learn the mathematical foundations of graphics

Apply them in programming projects

Implement code that demonstrates understanding

Images and Image Processing

What is an Image

- An image is a 2d rectilinear array of pixels
  ^ what do these words mean?

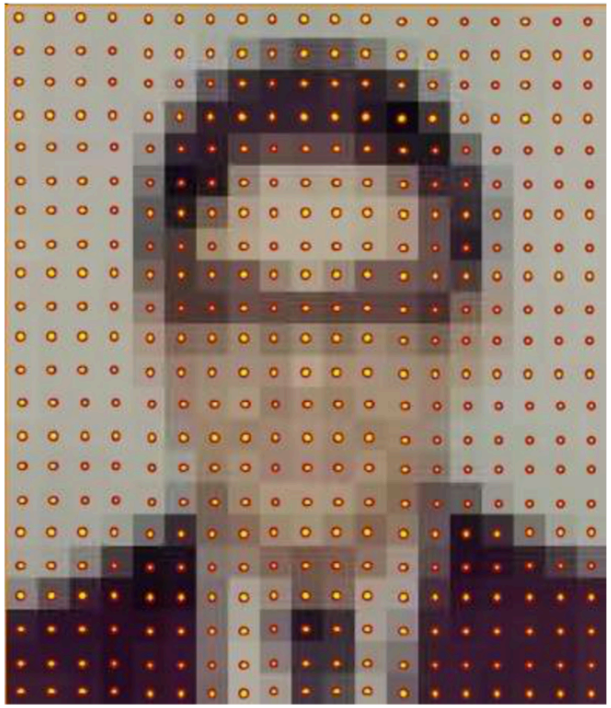
Continuous image

Digital Image

What is an Image

- An image is a 2d rectilinear array of pixels
  ^ what do these words mean?

Continuous image

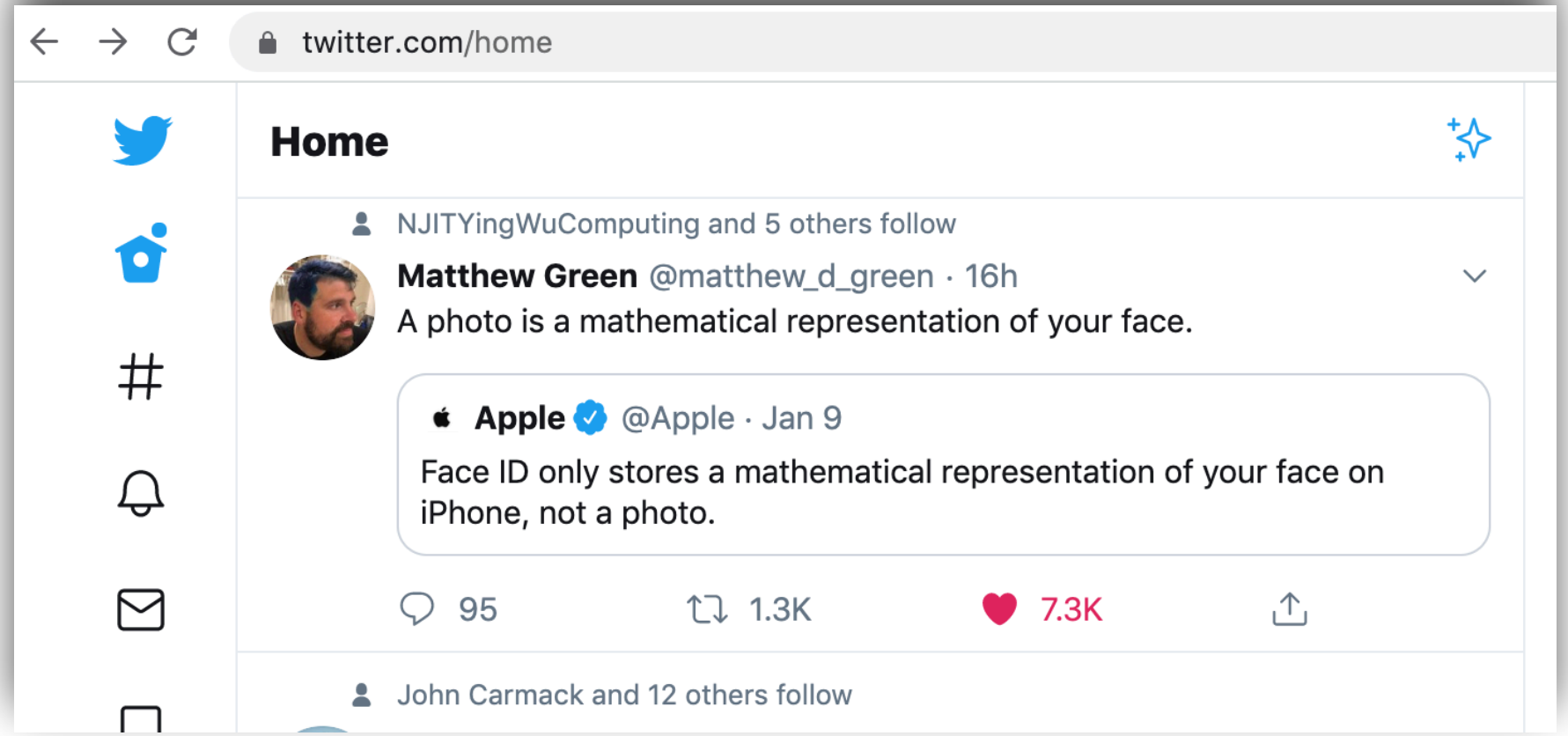


Digital Image

A pixel is a sample, not a little square…

Images: According to Twitter

- Are a mathematical representation of your face

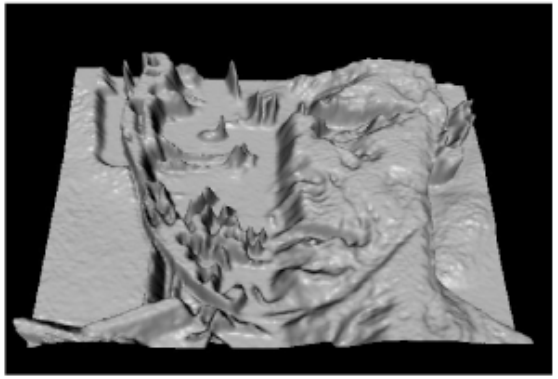


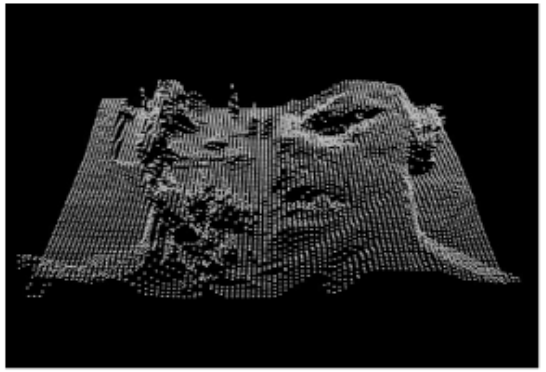
Images as Functions

Every pixel value is from 0 to 255

* 0: black
* 255: white
  * 125 -> value between black and white color wise

* 255 > higher, like a mountain
* 0> like a valley





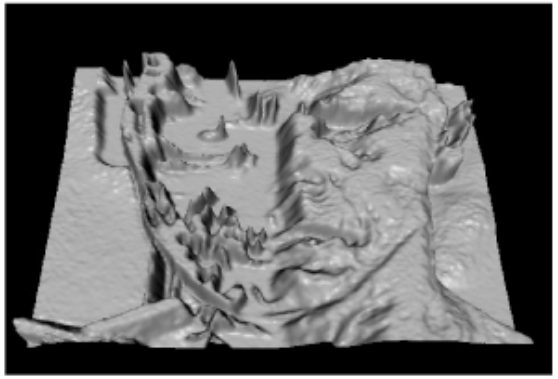
Images as Functions

Class PixelObject:
  int x (index of 2d array)
  int y
  int color (from 0 to 255)

Image ->
2d array of PixelObject
e.g. 2X2 image:
[[p1,p2],
[p3,p4]]

Where p1,p2,p3,p4 are PixelObjects

[[color1,color2],
[color3,color4]]



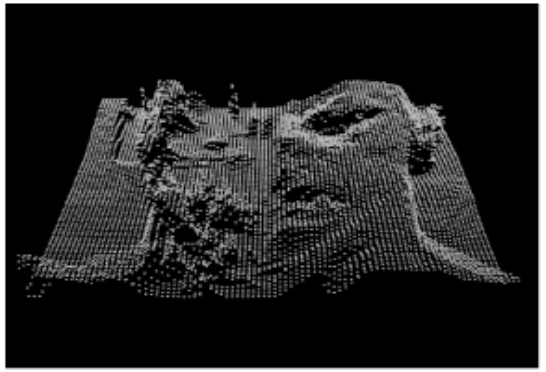


Image Acquisition

- Pixels are a sample of the real world - a continuous function
- Photoreceptors in the eye
- CCD cells in digital camera
- Rays in virtual camera
  ...more on this soon

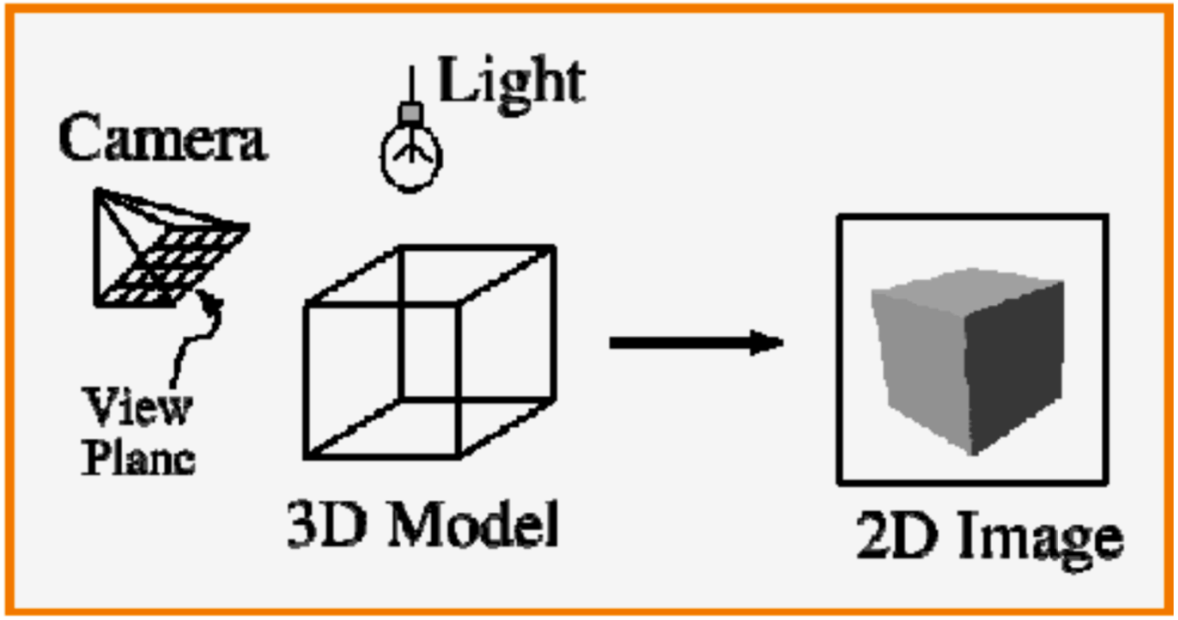


Image Display

- Recreate continuous function from samples
  - Cathode Ray Tube (CRT)

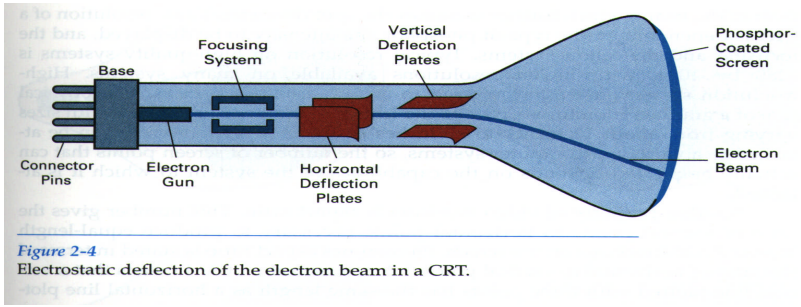
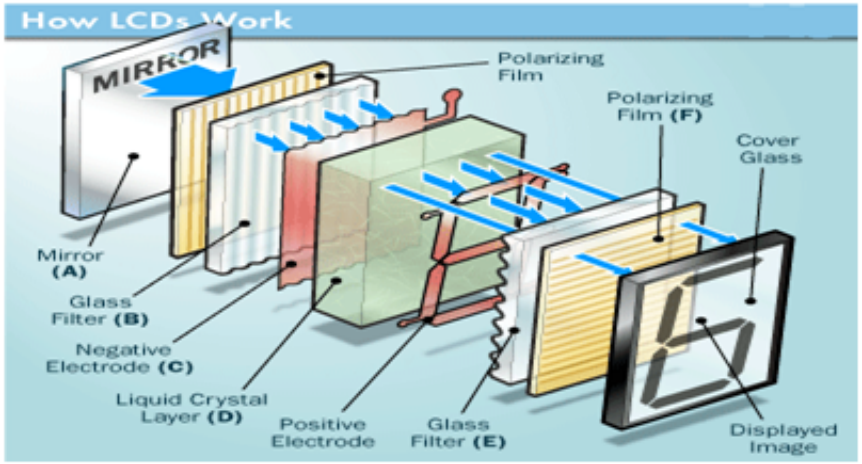


Figure 2-4
Electrostatic deflection of the electron beam in a CRT.

- Liquid Crystal Display (LCD)



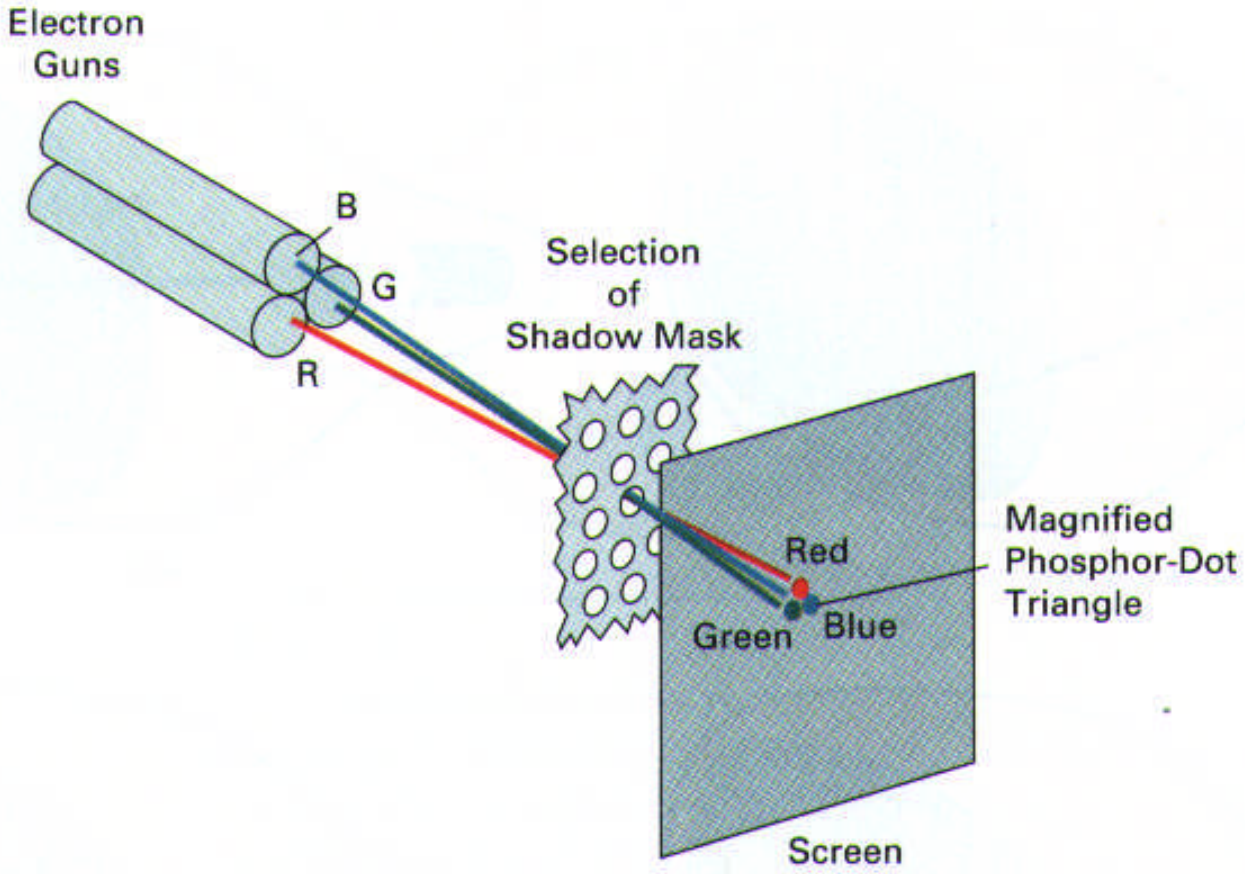
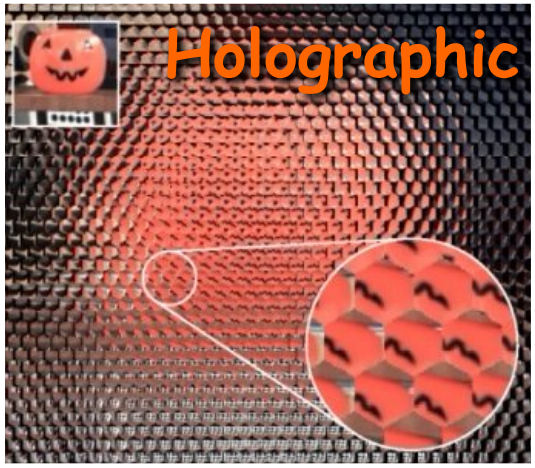


Figure 2-10

Operation of a delta-delta, shadow-mask CRT. Three electron guns, aligned with the triangular color-dot patterns on the screen, are directed to each dot triangle by a shadow mask.

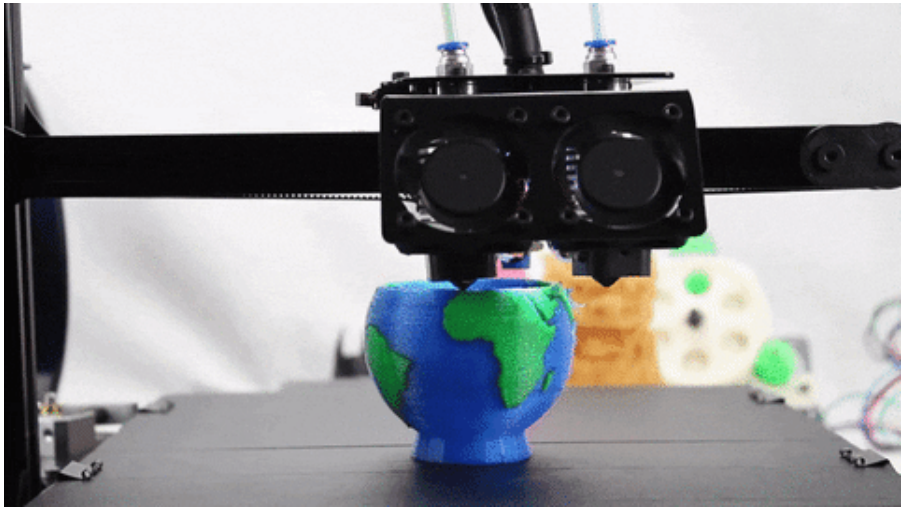
Other Displays

- Video Display



Other Displays

- Hard Copy Displays



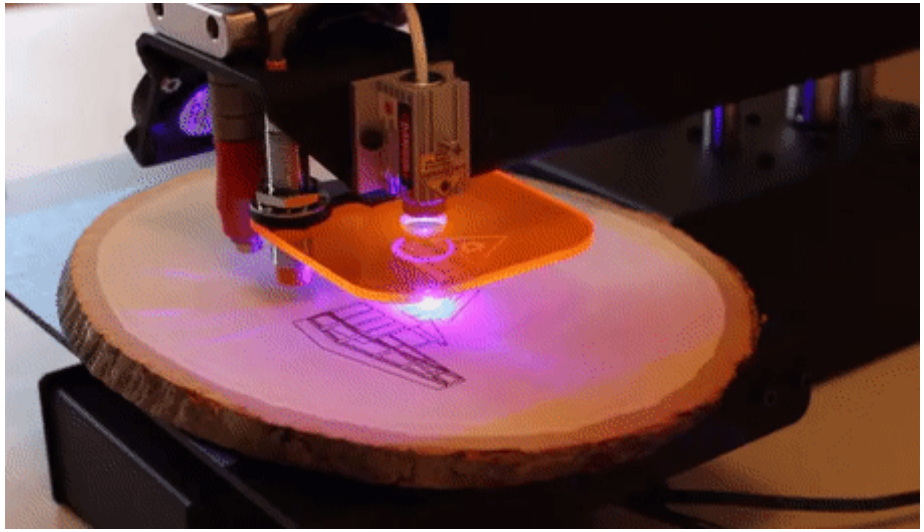


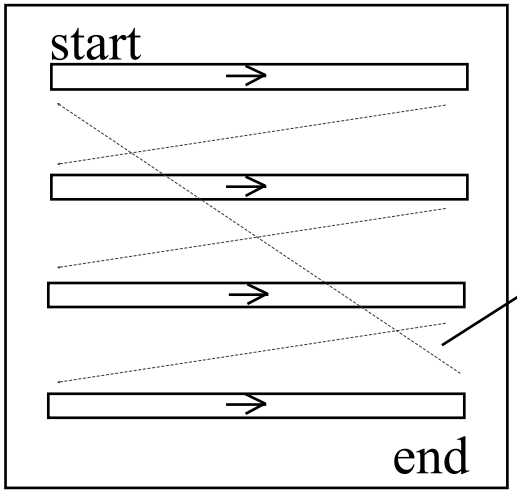
Image Resolution

- Intensity Resolution
  - Each pixel has only “depth” bits for color/intensities
- Spatial resolution
  - Image has only “Width” x “Height” pixels
- Temporal resolution
  - Monitor refreshes images at only “rate” Hz
  - For games it is typically 60HZ? -> like frames per second

Display	Width x Height	Depth	Rate (Hz)
NTSC	640 x 480	8	30
Workstation	1280 x 1024	24	75
Film	3000 x 2000	12	24
Laser Printer	6600 x 5100	1	-

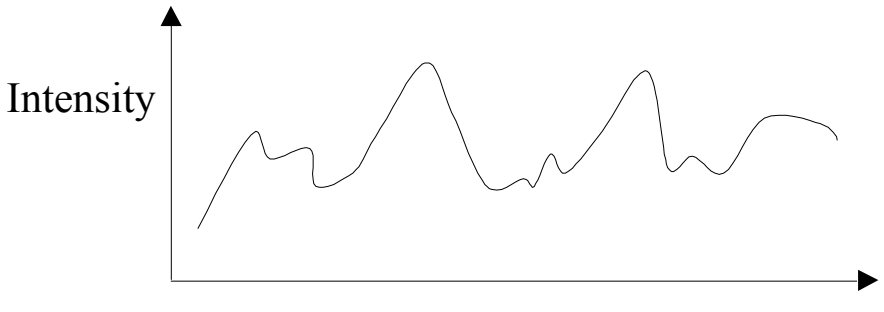
How do you make a picture?

Raster scan: Sweep screen with electron beam.



Vertical Retrace
(large delay)

Intensity on one line is like a graph



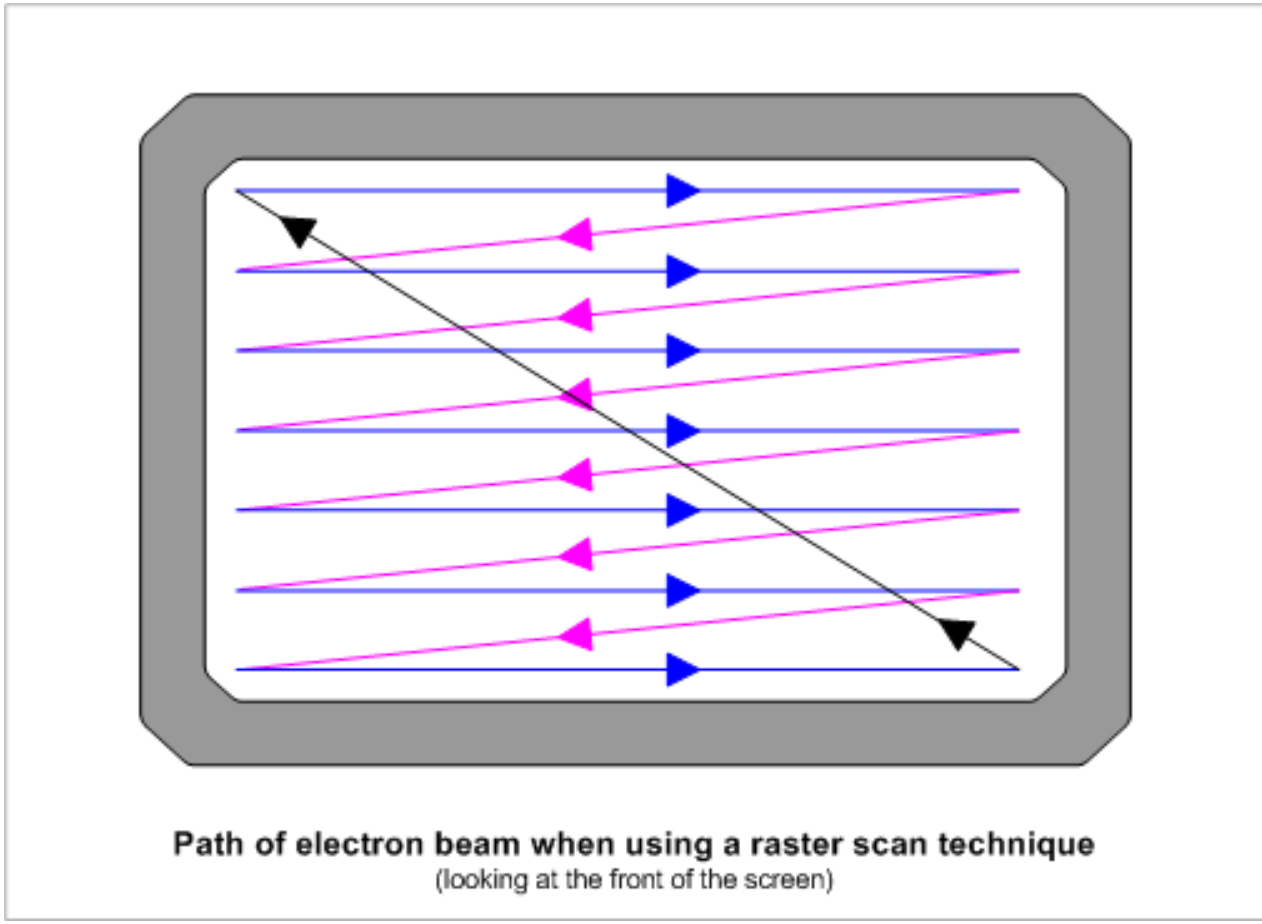
Refresh rate

– number of pictures drawable

in one second

Frame Buffer Refresh and Scan

- Electron beam traces over screen in **raster scan order**.
  - Each left-to-right trace is called a **scan line**.
  - Each spot on the screen is a **pixel**.
  - When the beam is turned off to sweep back, that is a **retrace**, or a **blanking interval.**



Path of electron beam when using a raster scan technique
(looking at the front of the screen)

Class Activity:

Change the image loaded in this code

https://editor.p5js.org/tomerwei/sketches/M9mhmQATM

Update your solution on canvas

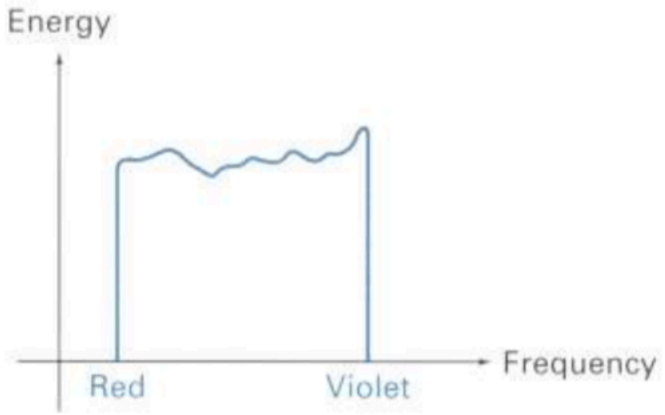
Colors

Color Models

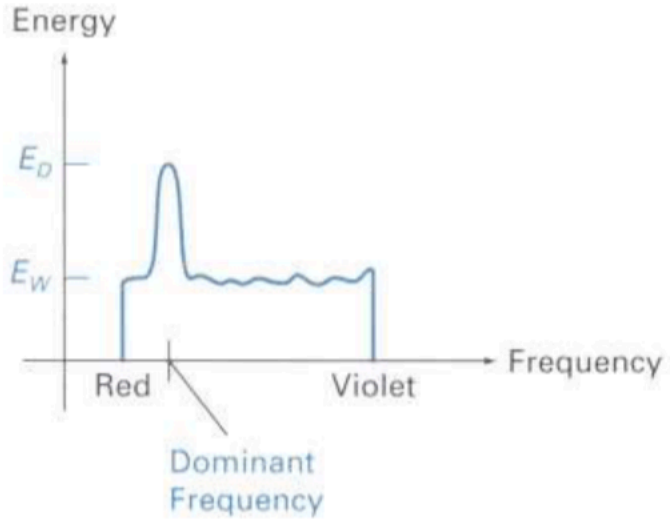
- How do we perceive colors?
  - Cones: Color
  - Rods: Light
    - NO RGB here!
- How can we describe and represent colors?

Color Models

- How do we perceive colors?
  … with Visible Light
- How can we describe and represent colors?
  The color of light is characterized by…
  - Hue = dominant frequency (highest peak)
  - Saturation = excitation purity (ratio of highest to rest)
  - Lightness = luminance (area under curve)



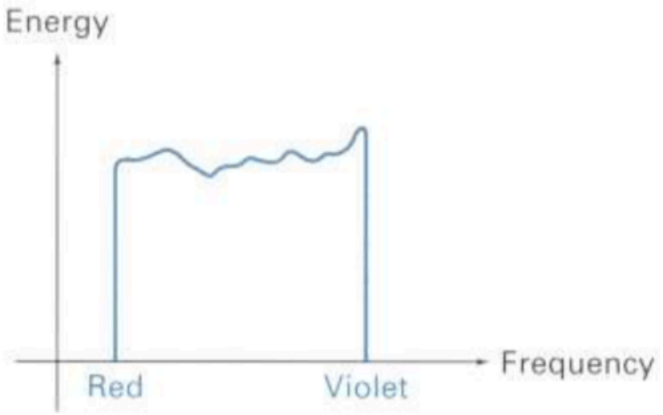
White Light



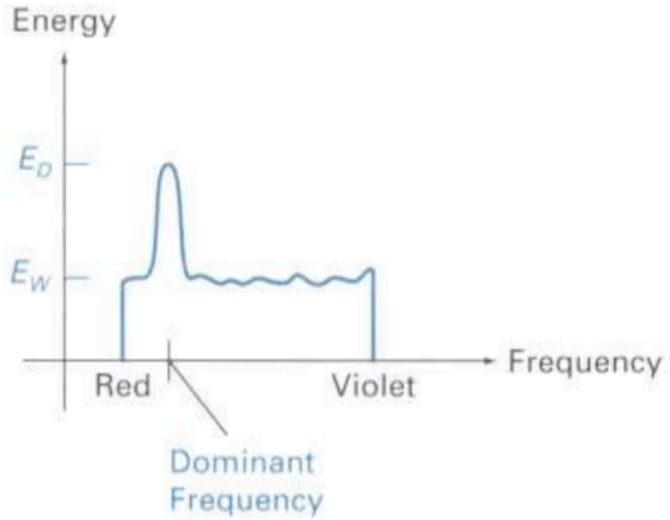
Orange Light

Color Models

- How do we perceive colors?
  … with Visible Light
- How can we describe and represent colors?
  The color of light is characterized by…
  - **Hue = the color we see (red, yellow, purple)**
  - **Saturation = How far is the color from gray (pink is less saturated than red, sky blue is less saturated than royal blue)**
  - **Lightness = How bright is the color**

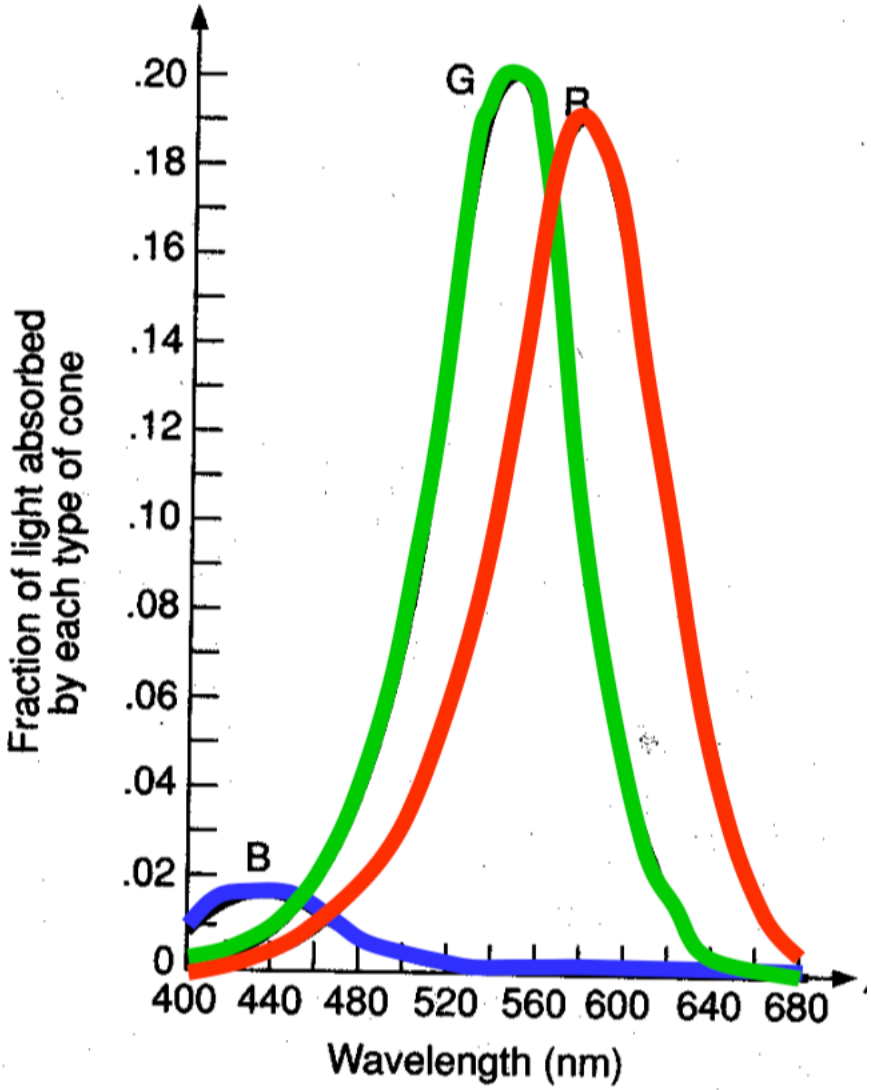


White Light



Orange Light

Color Perception



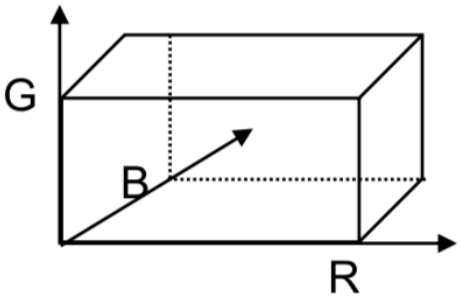
- Spectral-response functions of each of the three types of cones on the human retina

Tristimulus theory of color*

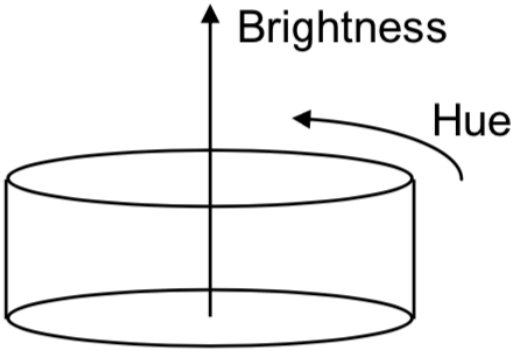
Color Spaces

- How do we define 3D color space?

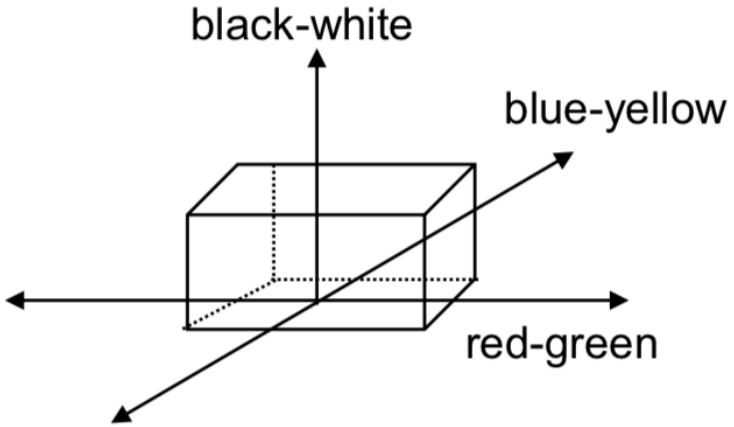
Cubic Color Spaces



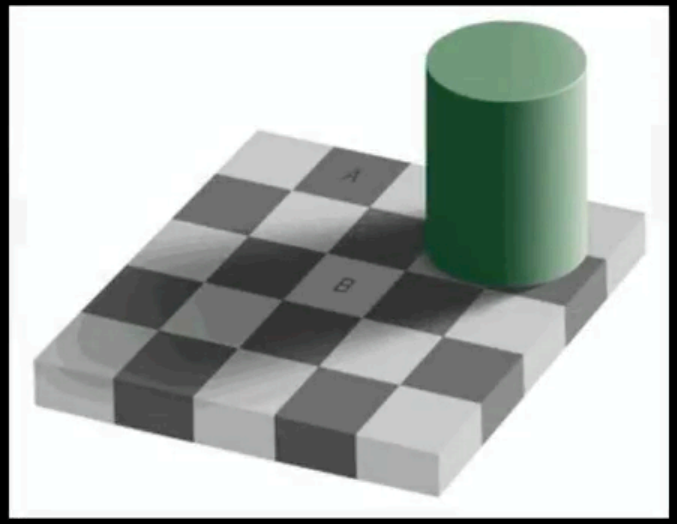
Polar Color Spaces



Opponent Color Spaces

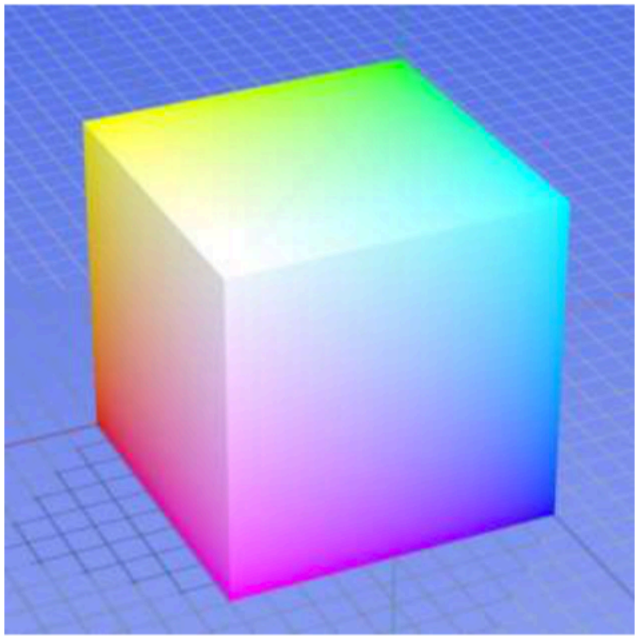


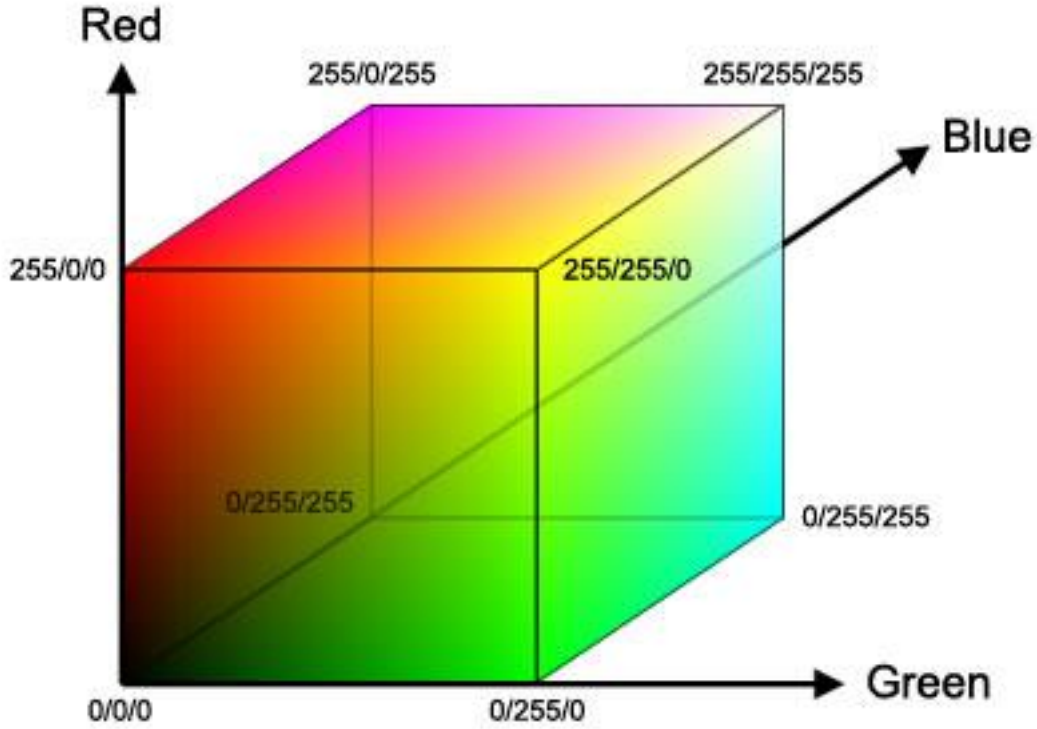
Color Spaces



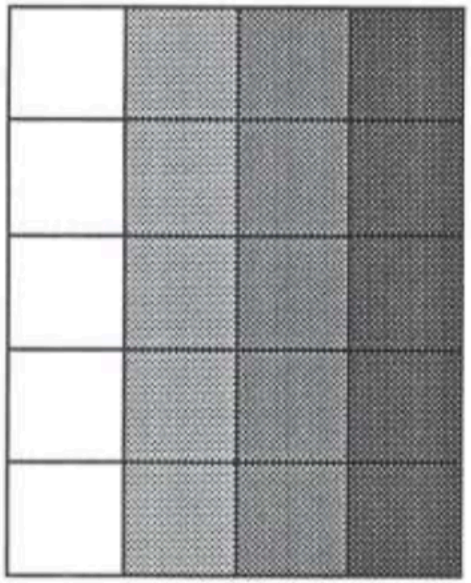
Color space: RGB

- Perhaps the most familiar color space, and the most convenient for display on a CRT.
- **Additive** color space





Color Frame Buffer



			255	150	75	0	0
		255	150	75	0	0	
	255	150	75	0	0		
255	150	75	0	0			
255	150	75	0	0			
255	150	75	0	0			
255	150	75	0	0			
255	150	75	0				

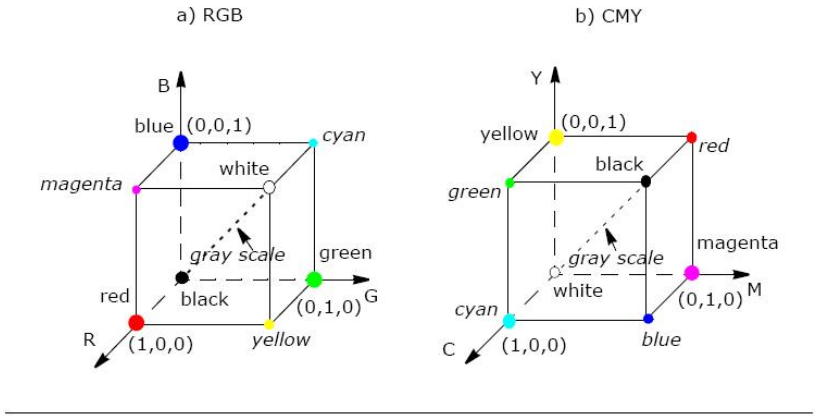
← Blue channel

← Green channel

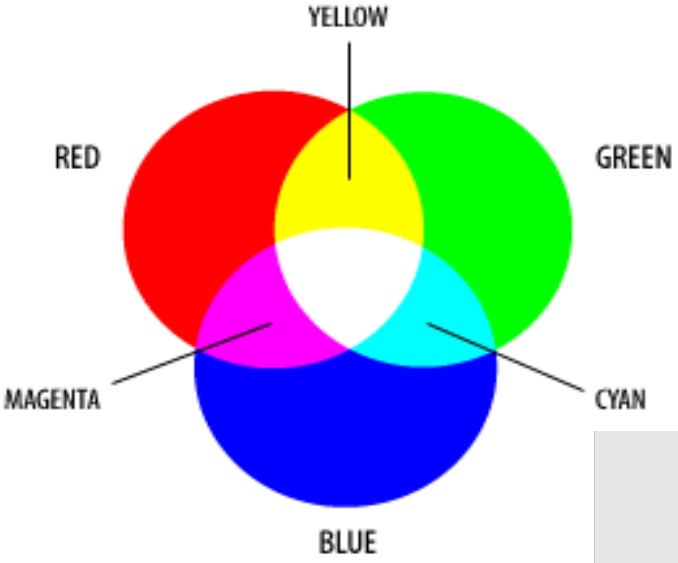
← Red channel

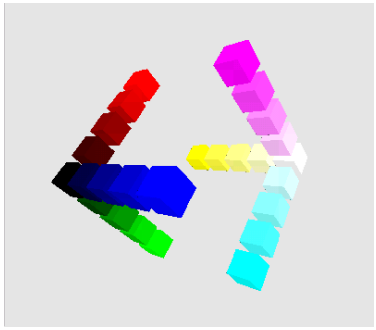
Color space: CMY

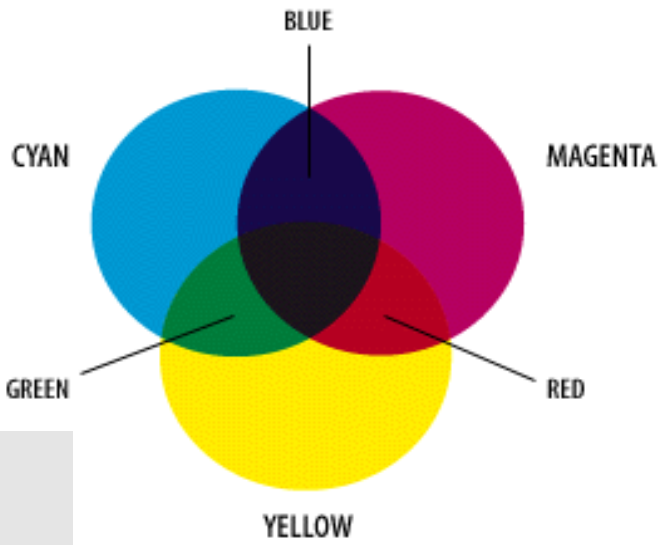
- A **subtractive** color space used for printing
- Involves three subtractive primaries:
  - **Cyan - subtracts red**
  - **Magenta - subtracts green**
  - **Yellow - subtracts blue**
- Mixing two pigments subtracts their opposites from white.
- CMY ink masks color on a white background (reduce light that would otherwise be reflected)
- CMYK adds black ink rather than using equal amounts of all three (more economic).



RGB vs. CMY

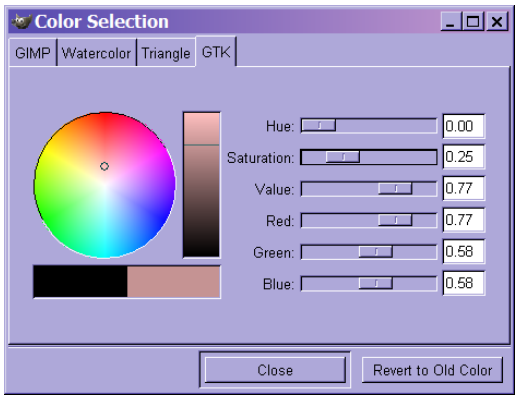

YELLOW
RED
GREEN
MAGENTA
CYAN
BLUE



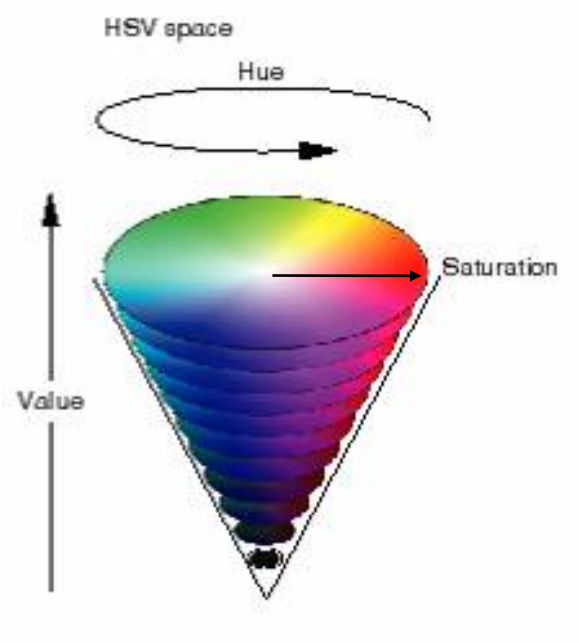

BLUE
CYAN
MAGENTA
GREEN
RED
YELLOW

Color space: HSV

- More natural for user interaction, corresponds to the **artistic concepts** of tint, shade and tone.
- The HSV space looks like a cone:
  - Hue ~ circular (color)
  - Saturation ~ distance from axis
  - Value ~ brightness



[from GIMP]

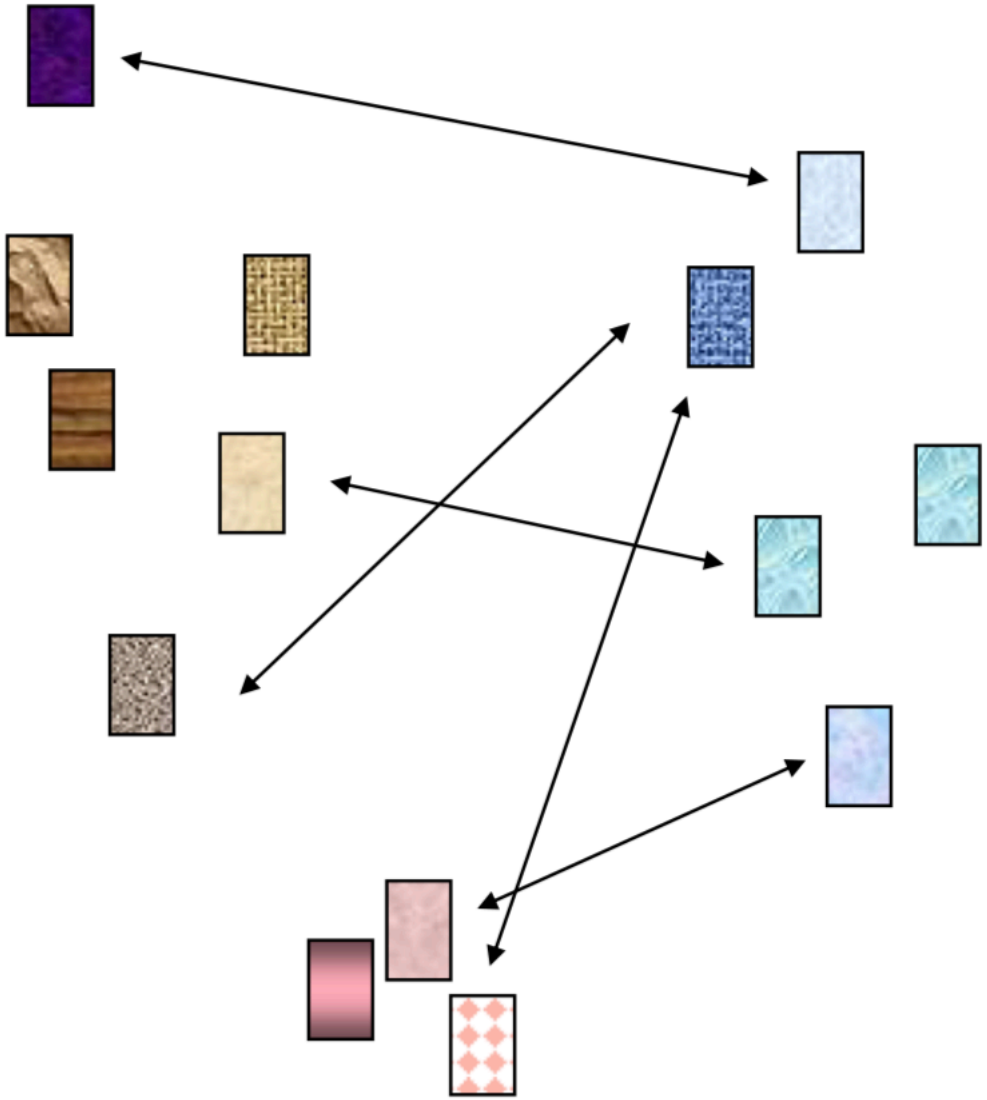


Demo: http://math.hws.edu/graphicsbook/demos/c2/rgb-hsv.html

Critiquing Game Visuals: www.youtube.com/watch?v=Dm24XyRPMwg

Distance between colors

- Distances are not linear in any color space
- Measuring differences between pixels is more useful in perceptual color spaces



Color - Summary

* Light →Eye (Cones,Rods) →[l,m,s] →Color
* Many 3D color models (RGB,CMY,…)
* We will focus on RGB
    * Because all the software\API\tools - python to Unreal Engine… use RGB
* RGB(A) - kind like RGB + Opacity

Class Activity:

Make the image reddish

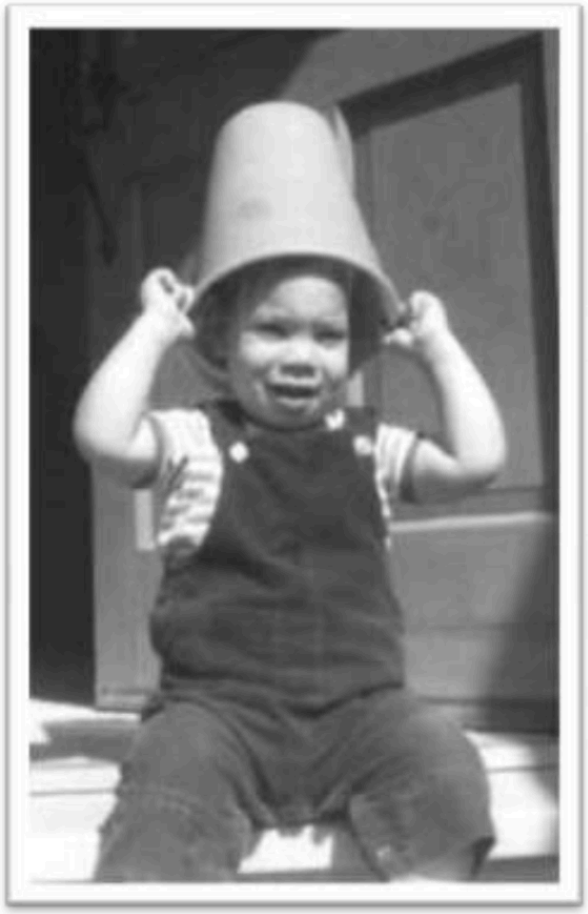
https://editor.p5js.org/tomerwei/sketches/M9mhmQATM

Update your solution on canvas

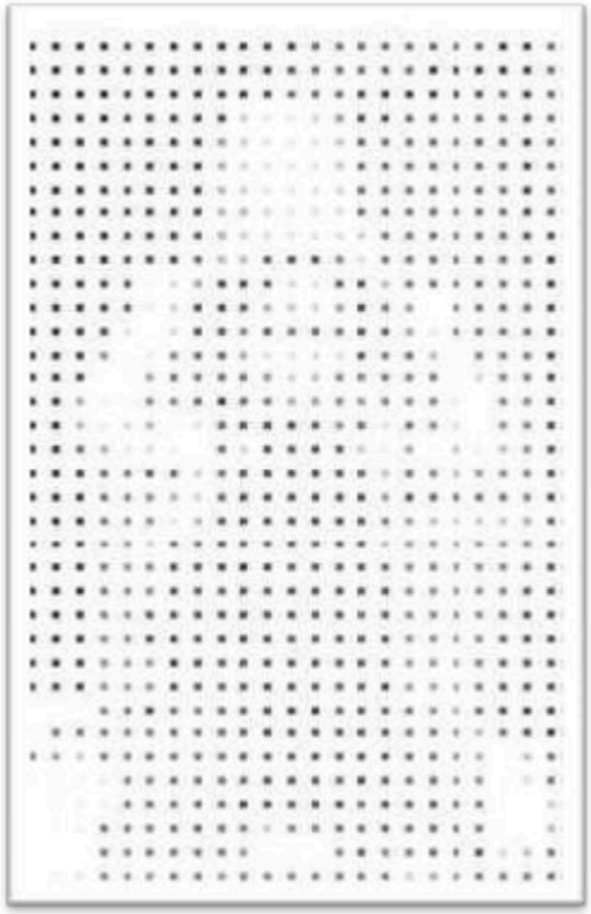
Images and Image Processing

Images: Recap

- An image is a discrete array of samples representing a continuous 2d function



Continuous function



Discrete samples

Images

- In computer graphics, we usually operate on **digital** (**discrete**) images
  - **Quantize** space into units (pixels)
  - A kind of step function
  - f : {0 … m -1}x{0 … n -1}
- An image processing operation typically defines an image f ’ in terms of an existing image f

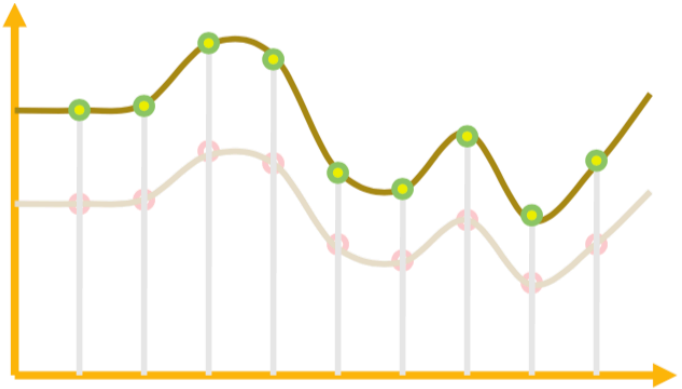
f: original

f’: sharpened

Adjusting Brightness

- Simply scale pixel components
  - Typically RGB is 0 to 255
  - Sometimes it will be 0 to 1.0 (basically 0->0, 1.0 -> 255)
  - Must clamp to range (eg. 0 to 1)?
    - Lets assume our pixel has value 255. Now we brighten it -> 255 + x —> 265
      assuming x=10 is the brightening scaler
      is 265 legit for a pixel?

https://www.html5rocks.com/en/tutorials/canvas/imagefilters/



Original

Brighter

- 255+10 = 265 bad
- oldVal -> previous pixel value
- newVal = min(255, oldVal+ brig

Adjusting Brightness

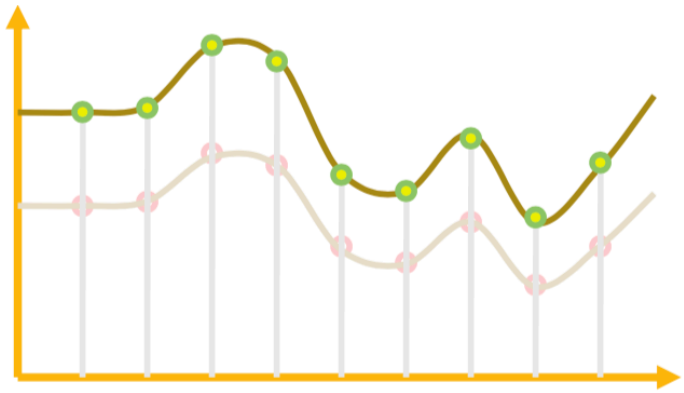
- [[1,2],[3,4]] —> [1,2,3,4]

Pixel 1	Pixel 2
Pixel3	Pixel4

https://www.html5rocks.com/en/tutorials/canvas/imagefilters/

Original

Brighter



Class Activity:

Complete the brightness code

https://editor.p5js.org/tomerwei/sketches/M9mhmQATM

Update your solution on canvas

Noise

- Common types of noise

Original

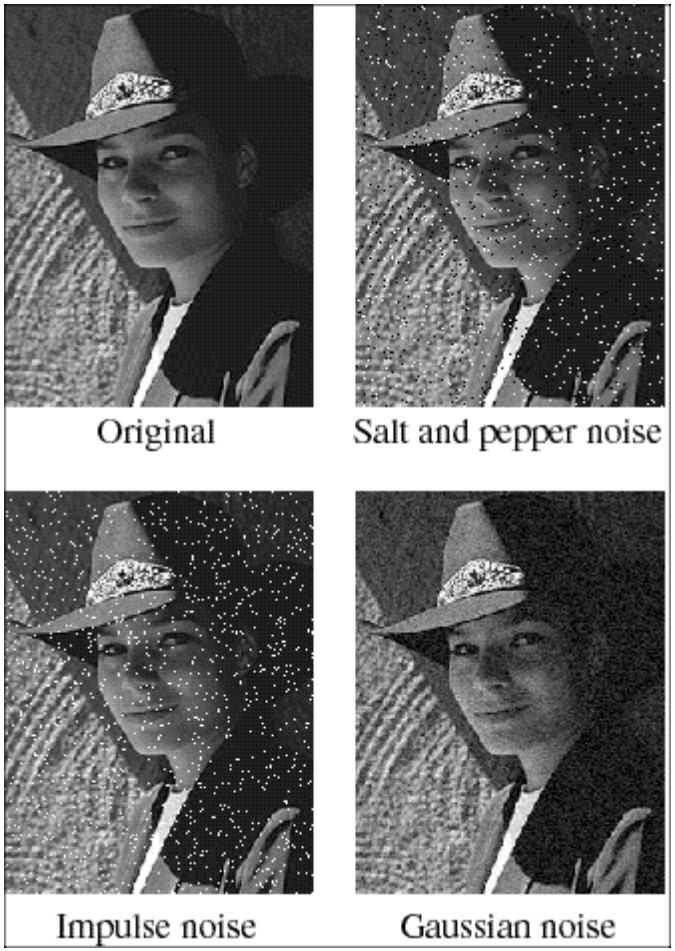
Salt and pepper noise

Impulse noise

Gaussian noise

Noise Reduction

- Is there a way to “smooth” out the noise?



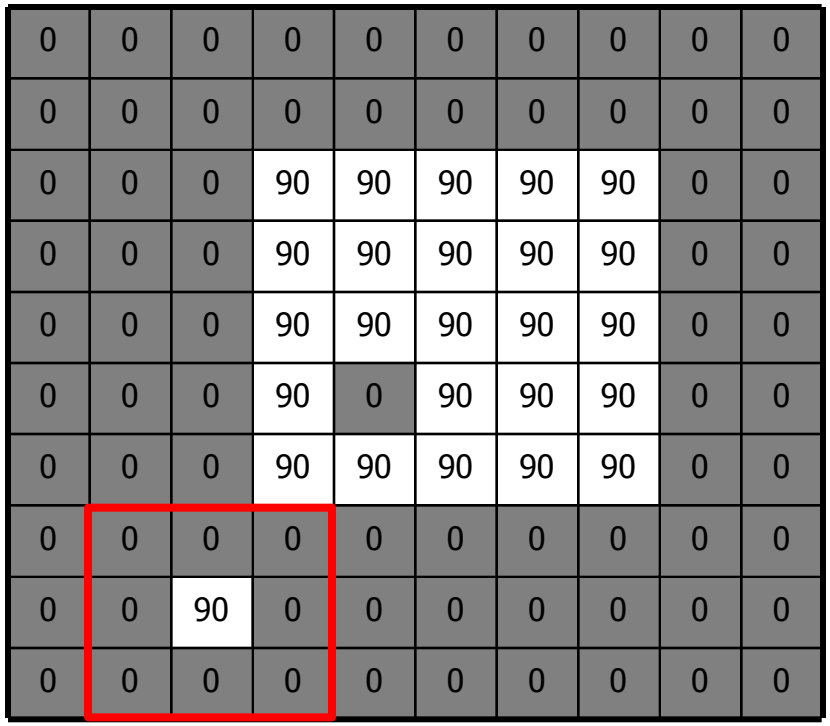
- Common types of noise:
  - **Salt and pepper noise**: random black and white pixels
  - **Impulse noise:** random white pixels
  - **Gaussian noise**: variations in intensity drawn from a Gaussian (normal) distribution

Reducing Noise

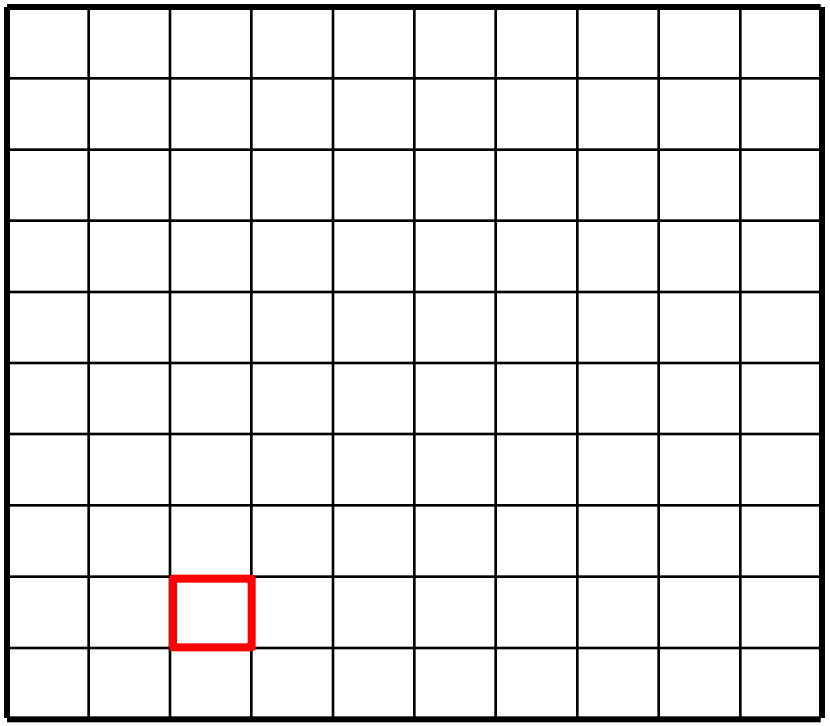
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

- Filtering
  - look at the neighborhood N around each pixel
  - replace each pixel with new value as a function of pixels in N
  - The behavior of $g(i,j) = h(f,N)$ d f

Mean filtering



$F[x, y]$



$G[x, y]$

- Replace each pixel with an average of the pixels in the k×k box around it
  - 3×3 case: $G[x, y] = \frac{1}{9} \sum_{u=0}^{2} \sum_{v=0}^{2} F[x + u - 1, y + v - 1]$

Mean filtering

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$F[x, y]$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

$G[x, y]$

- Replace each pixel with an average of the pixels in the k×k box around it
  - 3×3 case: $G[x,y] = \frac{1}{9}\sum_{u=0}^{2}\sum_{v=0}^{2} F[x+u-1, y+v-1]$

What about border pixels?

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$F[x, y]$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

$G[x, y]$

- Some options
  - don't evaluate—image gets smaller each time a filter is applied
  - pad the image with more rows and columns on the top, bottom, left, and right
    - option 1: copy the border pixels: add [0 0 0] to F in above case
    - option 2: reflect the image about the border: add [0 90 0] to F in above case

Class Activity:

Complete the filter code

https://editor.p5js.org/tomerwei/sketches/M9mhmQATM

Update your solution on canvas

Effect of filter size

- What happens if we
- use a larger mean filter?
- 5×5? 7×7?

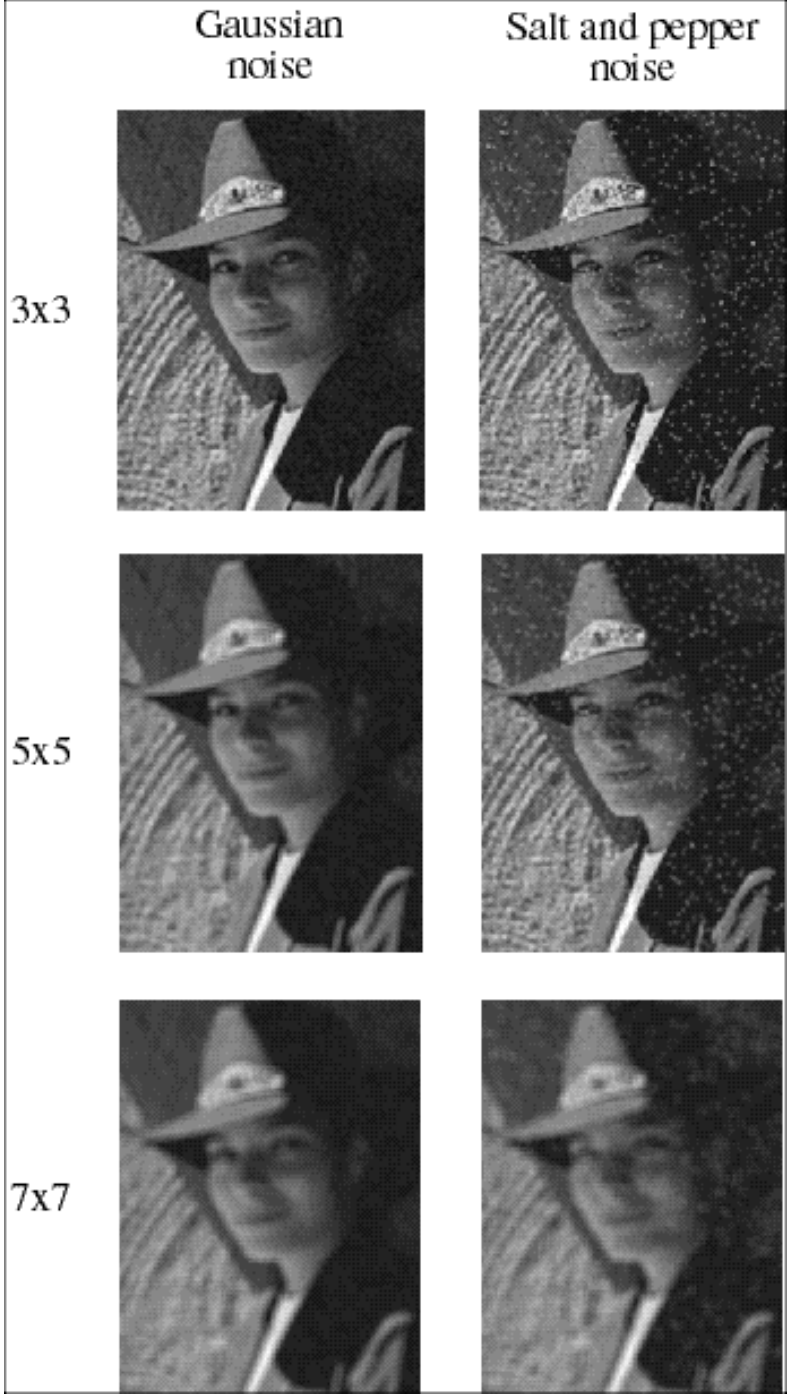


Image Processing

- Pixel operations
  - Add random noise
  - Luminance
  - Contrast
  - Saturation
- Quantization
  - Uniform Quantization
  - Floyd-Steinberg dither
- Filtering
  - Blur
  - Detect Edges
  - Sharpen
  - Emboss
  - Median
- Warping
  - Scale
  - Rotate
  - Warps
- Combining
  - Composite
  - Morph