

CSC 433/533

Computer Graphics

Alon Efrat
Credit: Joshua Levine

Today's Agenda

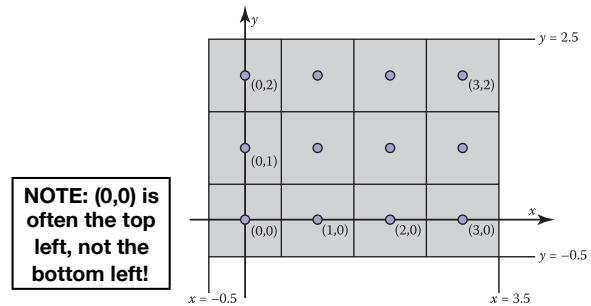
- Hw1 discussion
- **PLEASE START EARLY** so that we can debug any issues wrt git, Javascript, etc.
- Goals For Today:
- Finish discussing Javascript odds-and-ends (e.g. Accessing the DOM)
- Introduce images, displays

A Simple Mathematical Abstraction for Images

- We can abstract an image as a *function*, I
- $I: R \rightarrow V$
- The **domain**, R , is a some **continuous** rectangular area ($R \subseteq \mathbb{R}^2$) and the **range**,
- V , is a set of possible values.
- Since R is two dimensional, we can use $I(x,y)$ to represent the value of the image at a position $(x,y) \in R$

Raster Images

- We digitize $I(x,y)$ as an array of values, $I[y][x]$, called **pixels**, for **picture elements**

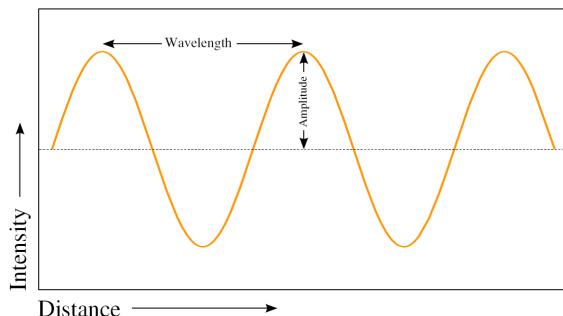


How Do We Acquire Raster Images?

Light

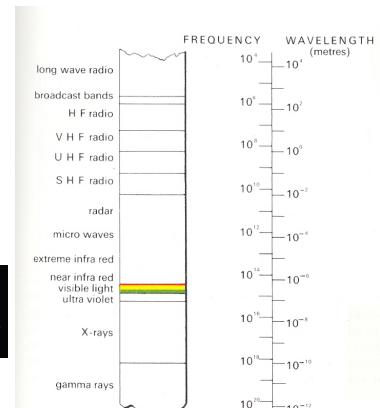
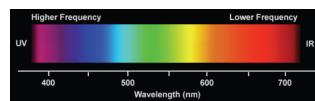
- Is both: (1) particles known as **photons** that (2) act as **waves**.
- **Amplitude** (height of wave)
- **Wavelength** (distance of which wave repeats)
 - Frequency is the inverse of wavelength
 - Relationship between wavelength (λ) and frequency (f):
 - $\lambda = c / f$
 - Where c = speed of light = 299,792,458 m / s

Light



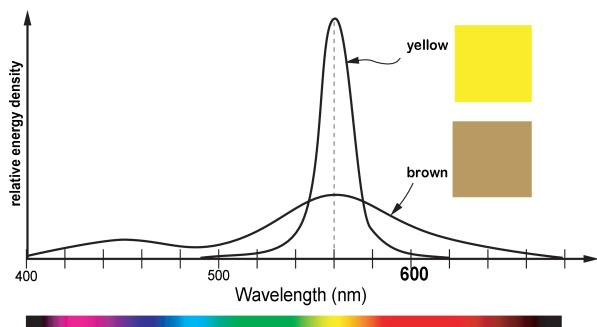
Light is Electromagnetic Radiation

- Visible spectrum is “tiny”
- Wavelength range: 380-740 nm

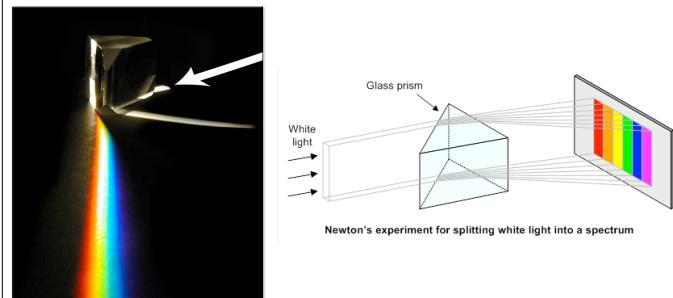


Color != Wavelength

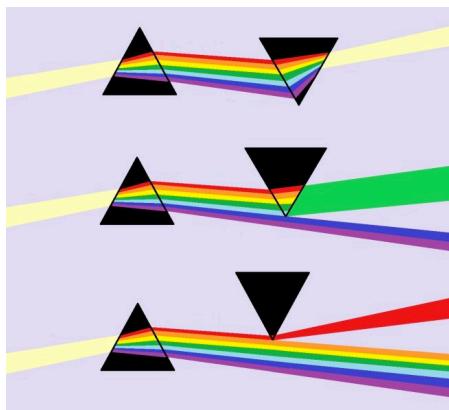
- But rather, a combination of wavelengths and energy



Isaac Newton, 1666



<http://www.webexhibits.org/colorart/bh.html>
<https://www.clivemaxfield.com/diycalculator/popup-m-cvision.shtml>

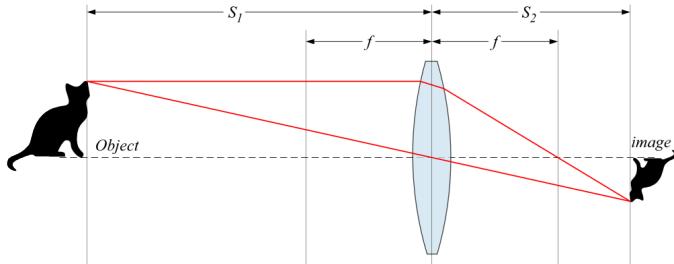


<http://www.thestargarden.co.uk/Newton's-theory-of-light.html>

Optics: Thin Lenses

- A **lens** is a transparent device that allows light to pass through while causing it to either converge or diverge.
- Given a camera, a target object, and a single converging lens:
 - Let S_1 and S_2 be the distance from the lens to the target and film
 - The **focal length**, f , is a measure of how strongly a lens converges light
 - The **magnification factor**, $m = S_2/S_1$, relates the two distances.

Thin Lens Equation

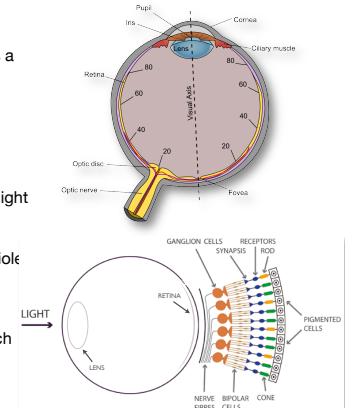


$$m = \frac{\text{image size}}{\text{object size}} = \frac{S_2}{S_1}$$

$$\frac{1}{S_1} + \frac{1}{S_2} = \frac{1}{f}$$

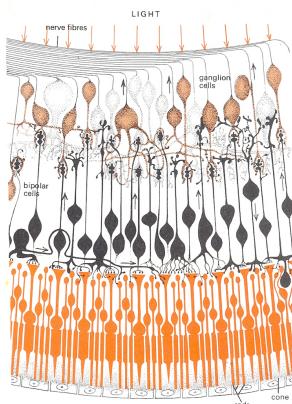
Human Optics

- In human vision, the **cornea** acts as a protective lens that roughly focuses incoming light
- Iris controls the amount of light that enters the eye
- The **lens** sharply focuses incoming light onto the retina
 - Absorbs both infrared and ultraviolet light which can damage the lens
- The **retina** is covered by **photoreceptors** (light sensors) which measure light



Photoreceptors

- Rods** (detect low-light / scotopic vision)
 - Approximately 100-150 million rods (Non-uniformly distributed across the retina)
 - Sensitive to low-light levels (scotopic vision)
- Cones** (detect day-light / photopic vision)
 - Approximately 6-7 million cones.
 - Detect color with 3 different kinds:
 - Red (L cone) : 564-580nm wavelengths (65% of all cones)
 - Green (M cone) : 534-545nm (30% of all cones)
 - Blue (S cone) : 420-440nm (5% of all cones)



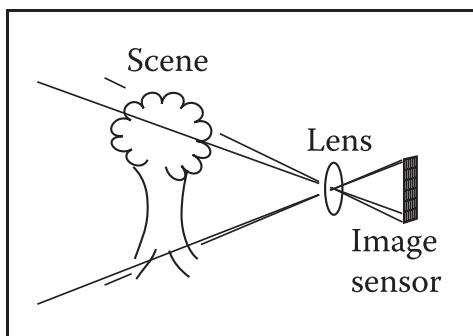
From Humans to Machines: Charge-Coupled Devices (CCDs)

- A CCD is an electronic circuit with a grid of small rectangular photocells.
- The optical lens focuses a scene onto the sensors.
- Each photocell measures the amount of light that hits it.
- The collective data of the sensors represents an image when viewed from a distance.



http://en.wikipedia.org/wiki/Charge-coupled_device

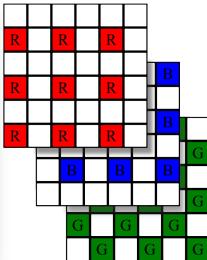
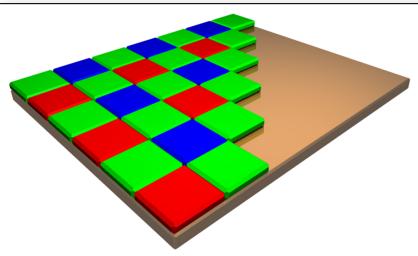
Color Image Acquisition



Color Image Acquisition

- In a single CCD color digital camera each individual photosite of the CCD is filtered to detect either red, green, OR blue light
- Most filters mimic the cone density of the human eye
- The Bayer filter uses 50% green and 25% red and blue sites.
- The 'RAW' data must be **demosaiced** (fill in the gaps) to produce a true-color image.

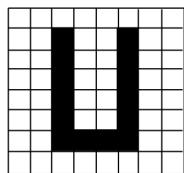
Bayer Filter



- Newer technology allows each photosite is able to discriminate and measure red, green and blue light simultaneously.

How Do We Encode Raster Images?

Bitmaps



```
1 1 1 1 1 1 1 1  
1 1 0 1 1 0 1 1  
1 1 0 1 1 0 1 1  
1 1 0 1 1 0 1 1  
1 1 0 1 1 0 1 1  
1 1 0 1 1 0 1 1  
1 1 0 0 0 0 1 1  
1 1 1 1 1 1 1 1
```

- **Bitmap:** digital image that is a 2d array of pixels which store one bit.
- Simplest digital image, a representation of a black and white image.
- Bit: ones/zeros, convention is 0 = black & 1 = white.

Digital Images Linearized

```
1 1 1 1 1 1 1 1  
1 1 0 1 1 0 1 1  
1 1 0 1 1 0 1 1  
1 1 0 1 1 0 1 1  
1 1 0 1 1 0 1 1  
1 1 0 1 1 0 1 1  
1 1 0 0 0 0 1 1  
1 1 1 1 1 1 1 1
```

11111111 | 11011011 | 11011011 | 11011011 | 11011011 | 11011011 | 11000011 | 11111111

FF DB DB DB DB DB C3 FF

- While we think of images as 2-dimensional, in memory we usually prefer to pack storage so that they are 1-dimensional.
- The same image can be represented in both binary and hexadecimal

Greyscale Images - Pixmaps

- We use 0 for black and 1 for white -- what value should we use for grey?
- Could use floating point numbers
- Instead, one convention is to use 8 bits for pixel -- how many different "shades of grey"?
- Can convert to [0.0,1.0] by dividing by 255

Javascript and Arrays

- Standard `Array` type in Javascript is *sparse*:
 - No guarantee of contiguous block of memory, memory is allocated on demand.
 - Can store mixed types
- Javascript `TypedArray` does use a contiguous block of memory
 - But, requires a fixed type

The operation Array.from

Definition and Usage

The `Array.from()` method returns an `Array` object from any object with a `length` property or an iterable object.

Create an Array from a String:

```
var myArr = Array.from("ABCDEFG");
```

Definition and Usage

The `Array.from()` method returns an `Array` object from any object with a `length` property or an iterable object.

Syntax

```
B = Array.from(object, mapFunction, thisValue)
```

`B`=`Array.from([1, 2, 3], x => x + x)`

`console.log(B);`

Output: `Array [2, 4, 6]`

Pixmap Declaration In Javascript

```
let ROWS = 8; let COLS = 8;  
//pixmap is an array of arrays  
let pixmap = [];  
for (let r = 0; r < ROWS; r++) {  
    pixmap[r] = [];  
}  
  
//pixmap2 is an array of arrays, but with lengths specified  
let pixmap2 = Array.from(Array(COLS), ()=>new Array(COLS))
```

```
//pixmap3 is utilizes a 1-d array for the whole thing  
let pixmap3 = Array(ROWS * COLS);  
  
//top left pixel (x,y) = (0,0)  
pixmap[0][0]; pixmap2[0][0]; pixmap3[0];  
  
//top right pixel (x,y) = (7,0)  
pixmap[0][7]; pixmap2[0][7]; pixmap3[7];  
  
//bottom left pixel, in general [index] = [y*COLS+x]  
pixmap[7][0]; pixmap2[7][0]; pixmap3[56];
```

Pixmap Declaration with TypedArrays

```
let ROWS = 8; let COLS = 8;  
  
//pixmap4 is utilizes a 1-d array for the whole thing  
let pixmap4 = new Uint8Array(ROWS*COLS);  
  
//Instead of storing as 0..255, can use other types  
//e.g. Uint8ClampedArray, Float32Array, etc.  
  
//Can we do multidimensional TypedArray  
//using Arrays of arrays?  
let pixmap4 = new Uint8Array(ROWS);  
pixmap4[0] = new Uint8Array(COLS); //incorrect!  
  
//for TypedArrays, we must use 1-d indexing  
//e.g. [index] = [y*COLS+x]
```

Image Allocation

```
let ROWS = 8;  
let COLS = 8;  
let pixmap2 = Array.from(Array(ROWS), () => Array(COLS));
```

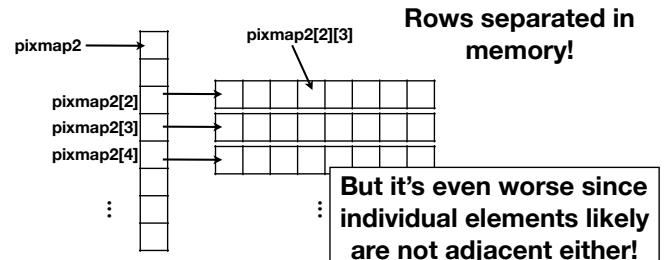


Image Allocation

```
let ROWS = 8;  
let COLS = 8;  
let pixmap4 = Uint8Array(ROWS*COLS);  
  
pixmap4 → [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]  
  
pixmap4[2] → [ ] [ ] [ ]  
pixmap4[3] → [ ] [ ] [ ]  
pixmap4[4] → [ ] [ ] [ ]
```

We don't have this
`pixmap4[2][3]`

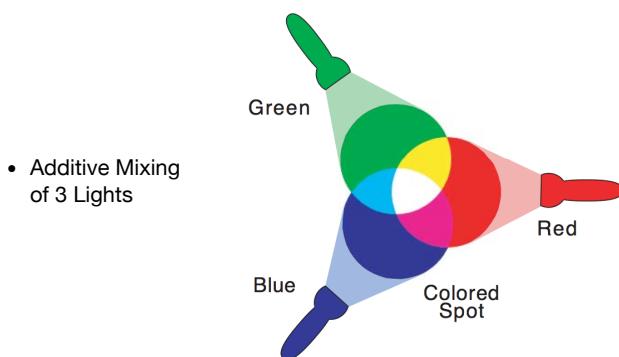
Memory contiguous, but cannot index by row / column

Encoding Color Images

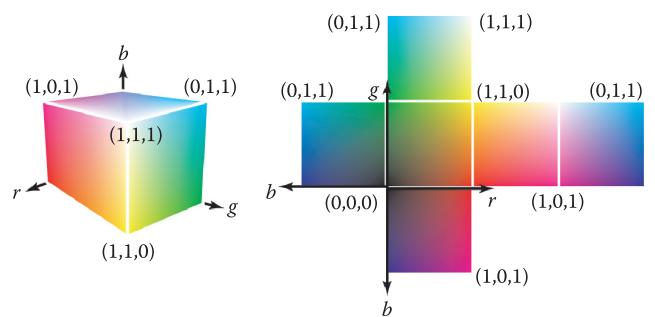
- Could encode 256 colors with an unsigned char. But what convention to use?
- One of the most common is to use 3 **channels** or bands
- Red-Green-Blue or RGB color is the most common -- based on how color is represented by lights.
- Coincidentally, this just happens to be related to how our eyes work too.

NOTE: There are many schemes to represent color, most use 3 channels. We'll come back to this next lecture

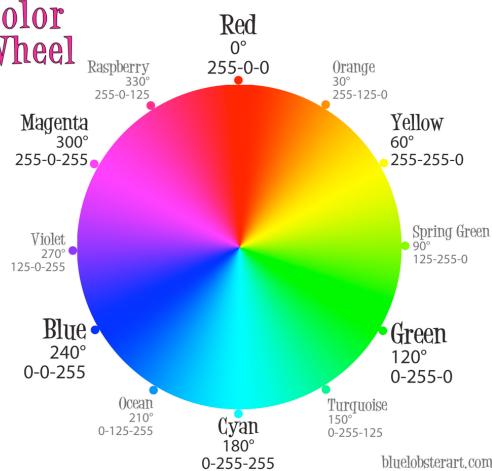
RGB Colors



RGB Color Cube



RGB Color Wheel



Encoding Color Channels

- Could use 8-bits, spread across all 3 channels (a bit ugly...)

$$59_{16} = \begin{array}{|c|c|c|} \hline 0 & 1 & 0 \\ \hline R & G & B \\ \hline \end{array} = (2/7, 6/7, 1/3) = (0.286, 0.757, 0.333)$$

- The textbook outlines a collection of other methods. The most common? 8-bit RGB images (24-bits per pixel)

```
//separate channel encoding
let red_pixmap = new Uint8Array(ROWS*COLS);
let green_pixmap = new Uint8Array(ROWS*COLS);
let blue_pixmap = new Uint8Array(ROWS*COLS);

//all together, could use an 32-bit uint,
//by standard convention we have 4 channels
let rgb_pixmap = new Uint8Array(4*ROWS*COLS);

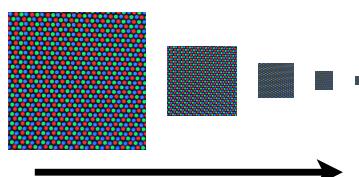
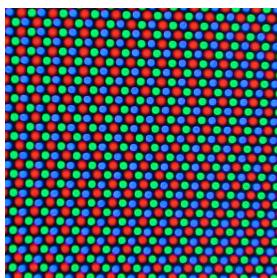
//access colors
let index = COLS*r + c;
rgb_pixmap[4*index + 0]; //red
rgb_pixmap[4*index + 1]; //green
rgb_pixmap[4*index + 2]; //blue
```

RGB Pixmap Encoding Options

How Do We Display Raster Images?

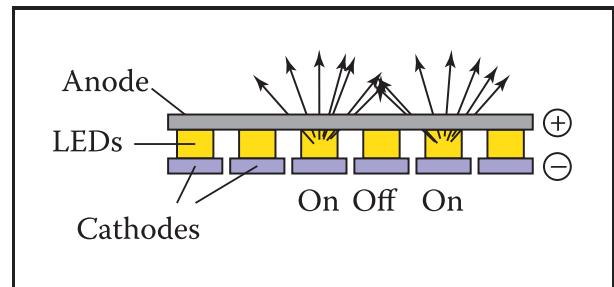
Optical Mixing

- To make (almost) any color, we combine light from three channels, Red, Green, Blue

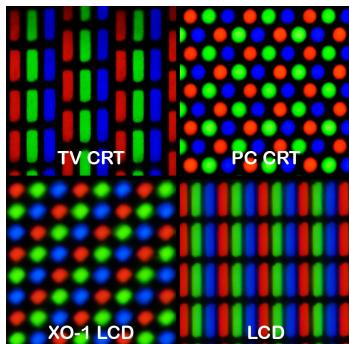


As pixels get smaller, the light blends

LEDs (Light-Emitting Diodes)



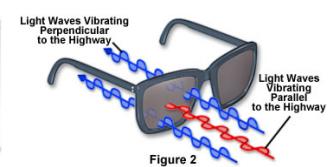
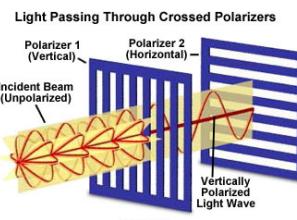
LCD Technology



- LCDs or Liquid Crystal Displays produce color by selectively blocking light through different filters
- Pixels are organized in various units of 3

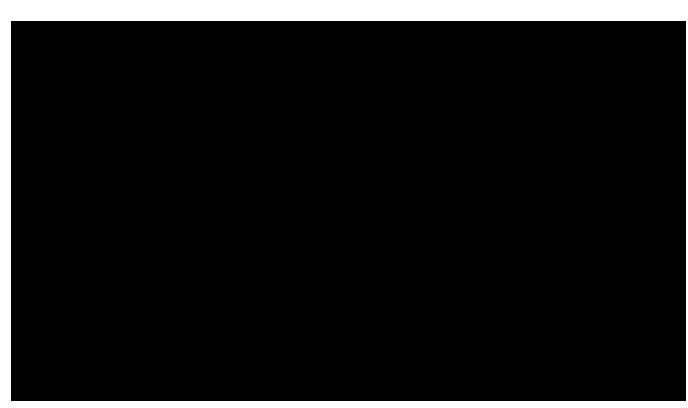
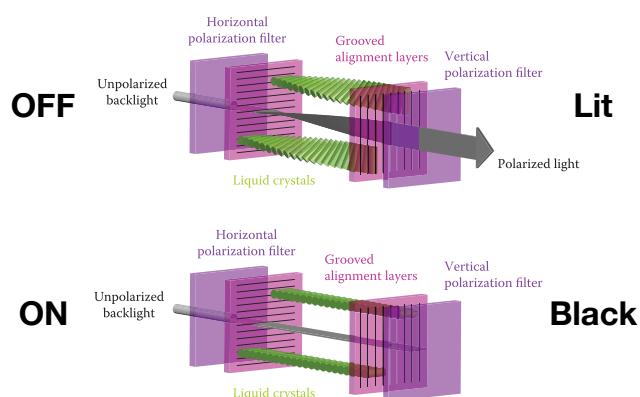
Polarized Light

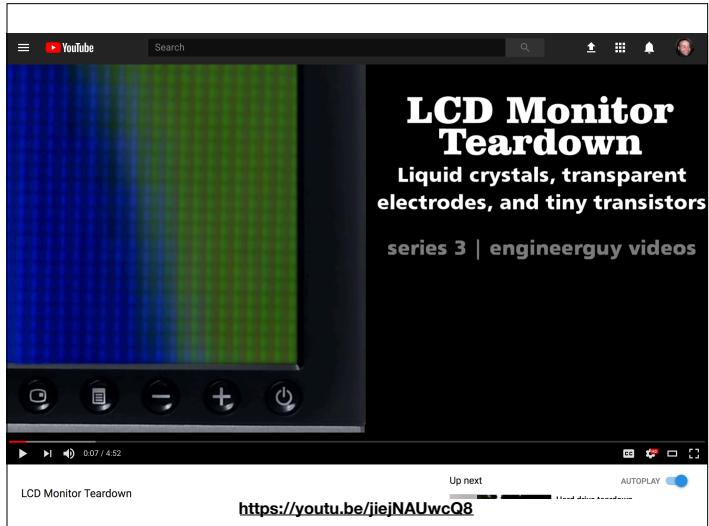
- Light oscillates as a wave in all directions perpendicular to its path. Polarizers selectively block certain oscillations



<http://www.olympusmicro.com/primer/lightandcolor/polarization.html>

Twisted Nematics





For next lecture

- FOCG, Ch. 3.3, 19
- See also optional readings!