

CS1315: Introduction to Media Computation

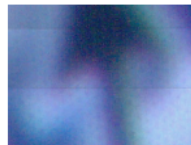
Picture encoding and manipulation

Today's class: Picture encoding

- Theory
 - Digitizing pictures into grids of pixels
 - Representing colors as numbers
- Practice
 - Manipulating pictures in JES

Digitizing pictures into dots

- We digitize pictures into many tiny dots
- Enough dots and it looks continuous to the eye
 - Our eye has **limited resolution**
 - Our **background/depth acuity** is particularly low
- Each picture element is a *pixel*



i.e. "picture element"

A Digitized picture is a *matrix* of pixels

- It's not a continuous line of elements, that is, a 1-D *array*
- A picture has two dimensions: Width and Height
- We need a 2-dimensional array: a *matrix*

	1	2	3	4
1	15	12	13	10
2	9	7		
3	6			

Just the upper left hand corner of a matrix.

Referencing a matrix

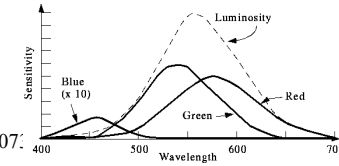
	1	2	3	4
1	15	12	13	10
2	9	7		
3	6			

- The values represent intensity:
 - 0 is the darkest
 - The maximum value is the brightest.
 - Q: What maximum value?
 - A: Usually 255.
 - Q: Why?
 - A: Later...

- We talk about positions in a matrix as (x, y), or (horizontal, vertical)
 - Element (2, 1) in the matrix at left is the value 12
 - Element (1, 3) is 6

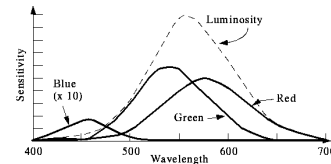
We perceive light different from how it actually is

- Color is continuous
 - Visible light is wavelengths between 370 and 730 nm
 - That's 0.00000037 and 0.00000073 meters
- But we *perceive* light with color sensors that peak around 425 nm (blue), 550 nm (green), and 560 nm (red).
 - Our brain figures out which color is which by figuring out how much of each kind of sensor is responding
 - One implication: We perceive two kinds of "orange" — any *spectral distribution* that hits our color sensors just right
 - Dogs and other simpler animals have only two kinds of sensors
 - They *do* see color. Just *less* color.



Luminance vs. Color

- We perceive borders of things, motion, depth via *luminance*
 - Luminance is *not* the amount of light, but our *perception* of the amount of light.
 - We see blue as "darker" than red, even if same amount of light.
- Much of our luminance perception is based on comparison to backgrounds, not raw values.

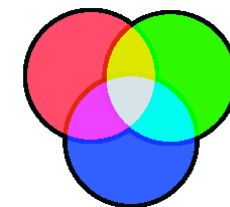
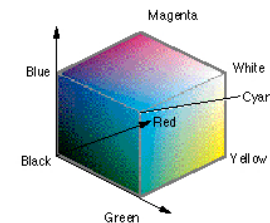


Luminance perception is *color blind*.

Different parts of the brain perceive color and luminance.

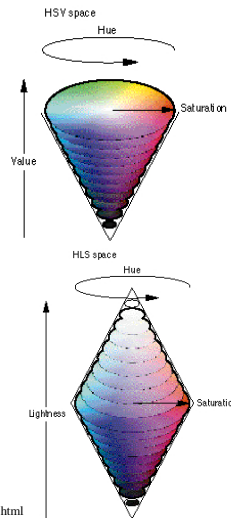
RGB: Three dimensions of color

- In RGB, each color has three component colors:
 - Amount of red
 - Amount of green
 - Amount of blue
- Each does appear as a separate dot on most devices, but our eye blends them.



There are other color models

- There are many encodings for color
 - **Printers use CMYK: Cyan, Magenta, Yellow, and black.**
 - **Spaces that aren't cubes have some advantages for computer graphics**
 - E.g. HSB (for Hue, Saturation, and Brightness)
 - **But we will use RGB exclusively. It is the simplest**



HSV, HLS, RGB images from Apple's ColorSync Developer Documentation at <http://developer.apple.com/documentation/GraphicsImaging/Conceptual/ManagingColorSync/index.html>

Why would the maximum intensity be 255?

- Let's walk through it.
 - **If we have one bit, we can represent two patterns: 0 and 1.**
 - **If we have two bits, we can represent four patterns: 00, 01, 10, and 11.**
 - **If we have three bits, we can represent eight patterns: 000, 001, 010, 011, 100, 101, 110, 111**
- The rule: In n bits, we can have 2^n patterns
 - **In 8 bits, we can have 2^8 patterns, or 256**
 - **If we make one pattern 0, then the highest value we can represent is 2^8-1 , or 255**
- In the simplest schemes for digitizing black and white pictures, a single *byte* (8 bits) of computer memory is used for each pixel
 - **So that's why intensities range from 0...255.**
 - **But what about color pictures?...**

Encoding RGB

- Each component color (red, green, and blue) is encoded as a single byte
 - **Because pixel has a combination of three colors, we need three bytes to store its color**
 - **I.e. a complete RGB color is 24 bits, 8 bits of each**
 - **Which means we can represent about 16 million distinct colors in RGB!**
- Colors go from (0, 0, 0) to (255, 255, 255)
 - **If all three components are the same, the color is in grayscale**
 - (50, 50, 50) at (2, 2)
 - **(0, 0, 0) (at position (1, 2) in example) is black**
 - **(255, 255, 255) is white**

	1	2	3
1	100,10,5	5,10,100	255,0,0
2	0,0,0	50,50,50	0,100,0

Is 256x256x256 enough colors?

- We're representing color in 24 ($3 * 8$) bits.
 - **That's 16,777,216 (2^{24}) possible colors**
 - **Our eye can discern millions of colors, so it is pretty close**
 - **But the real limitation is the physical devices: We don't get 16 million colors out of a monitor**
- Some graphics systems support 32 bits per pixel, not 24
 - **May be more pixels for color**
 - **More useful is to use the additional 8 bits to represent not color but 256 levels of *translucence***

Media jargon: 4th byte per pixel is the "Alpha channel"

Size of color images

	320 x 240 image	640 x 480 image	1024 x 768 monitor
24 bit color	1,843,200 bits 230,400 bytes	7,372,800 bits 921,600 bytes	18,874,368 bits 2,359,296 bytes
32 bit color	2,457,600 bits 307,200 bytes	9,830,400 bits 1,228,800 bytes	25,165,824 bits 3,145,728 bytes

Today's class: Picture encoding

- Theory
 - Digitizing pictures into grids of pixels
 - Representing colors as numbers
- Practice
 - **Manipulating pictures in JES**

Today's class: Picture encoding

- Theory
 - Digitizing pictures into grids of pixels
 - Picture is matrix of pixels
 - Pixels "know" where and what color they are
 - Representing colors as numbers
 - We use RGB color model
 - Each color is three values
 - Values range 0..255
 - So there are 16M colors
 - Practice
- Manipulating pictures in JES
 - getPixels
 - getting and setting colors
 - Computing distance between colors