Representing an analog world on a digital computer

Computers think different

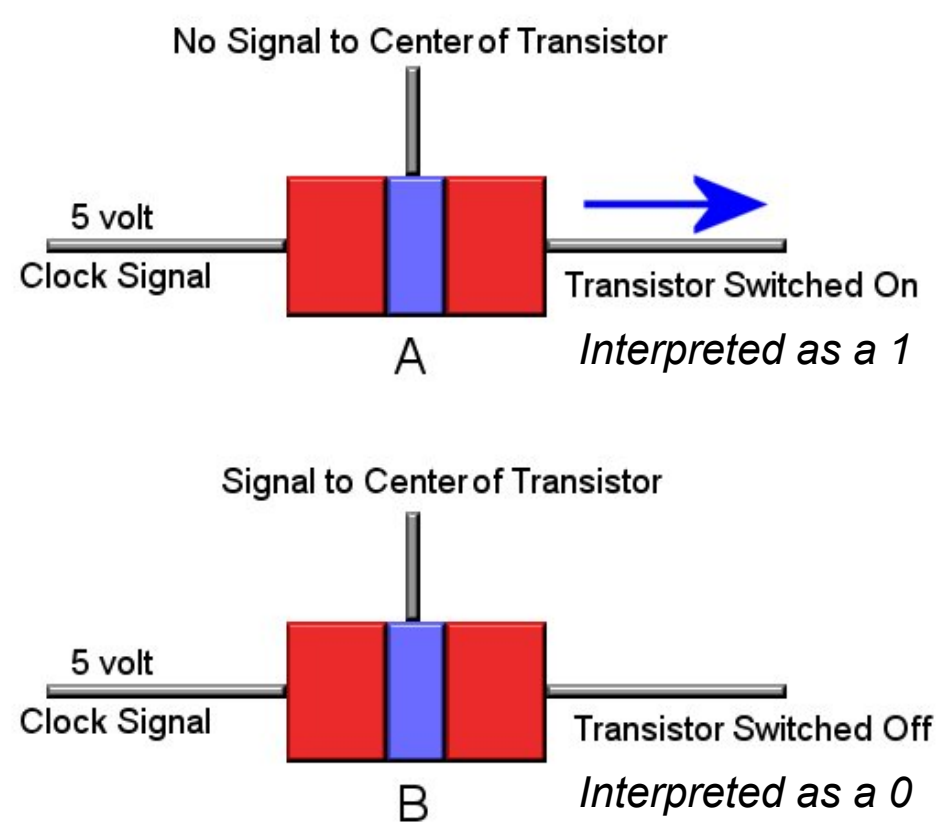
Computers are digital, the world is analog

- Analog = continuous quantities
  - Infinite resolution
- Digital = discrete values
  - Fixed resolution
- We live in an analog world
  - Continuous variables for physical dimensions
  - Shades of gray, gradients of color
  - Continuous gradations in pitch and intensity of sound
- To represent these analog traits on a computer, we need to convert them to a digital representation
- This can have important consequences to the accuracy of your data!

http://www.smithsonianmag.com/smart-news/electron-microscope-zooms-in-finds-life-on-life-on-life-30070438/?no-ist

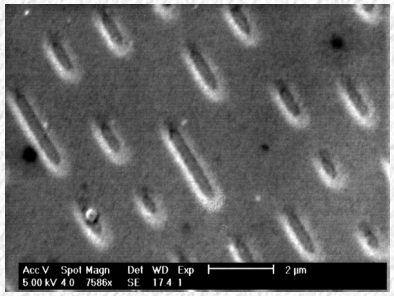
Computers only understand 0 and 1

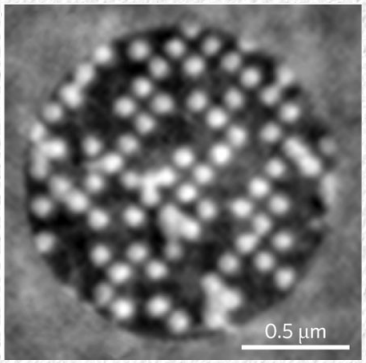
- Microprocessors are made up of many (i.e. billion +) transistors, which are like tiny switches
- Two possible states
  - On = interpreted as a 1
  - Off = interpreted as a 0
- A single transistor, set to a 0 or a 1, is called a "**bit**"
- Everything on a computer has to be represented by a series of bits



Encoding bits for storage

- Can't rely on constant electrical power for storage
- Pits in a surface – optical storage
  - CD-ROM
  - DVD
- Orientation of magnetic crystals – magnetic media
  - Hard drives
  - Magnetic tape
  - Flash media





Some analog traits

- Integer numbers
- Continuous numbers
- Time
- Color
- Sound

- How do we represent these things in a computer that only understands 0 and 1?

Decimal integers (base 10)

- We're used to decimal numbering
  - Digits from 0 to 9
  - To represent numbers bigger than 9, we string digits together
  - Positions in a number are exponents of 10
  - Consider 1,324

1000's place	100's place	10's place	1's place
10^3	10^2	10^1	10^0
1 ,	3	2	4

  - 1,324 means we have 1 @ 10^3, 3 @ 10^2, 2 @ 10^1 and 4 @ 10^0

Binary numbers

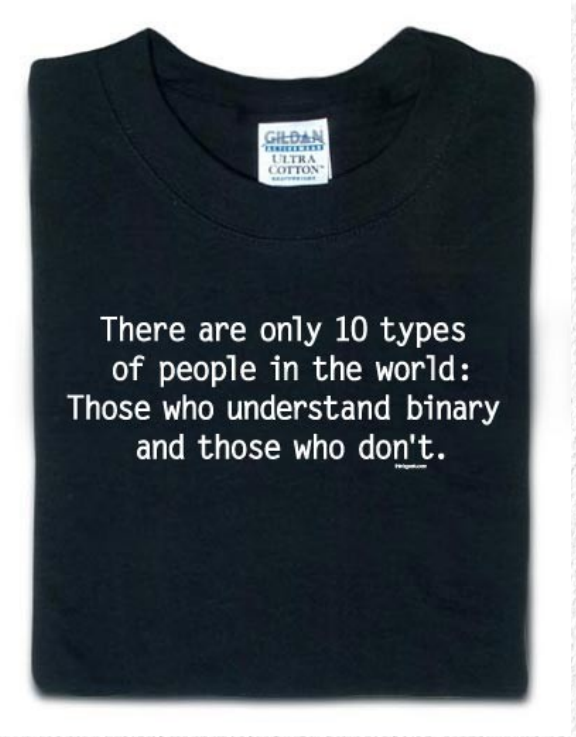
- Only two digits possible – 0 or 1
- To represent numbers bigger than 1, need to string digits together (just like with decimal numbers)
- With only 2 possible digits, each position is a power of 2
- What would 1000 be in binary?

8's place	4's place	2's place	1's place
2^3	2^2	2^1	2^0
1 ,	0	0	0

Convert a binary number to decimal

- Binary 10 means 1 @ 2^1, and 0 @ 2^0 = 2

- Binary 11 means $2^1 + 2^0$, or 3



Counting in binary

Decimal number	Binary number	Conversion
0	0	0
1	1	1
2	10	2 + 0
3	11	2 + 1
4	100	4 + 0 + 0
5	101	4 + 0 + 1
6	110	4 + 2 + 0
7		4 + 2 + 1
8		

Largest decimal number you can represent with:

8 bits = 1 byte

We can devote the left-most bit to represent the sign (+ or -)

Number of bits	Largest binary number	Decimal number
1	1	1 = 1
2	11	2+1 = 3
3	111	4+2+1 = 7
4	1111	8+4+2+1 = 15
5	11111	16+8+4+2+1 = 31
6	111111	32+16+8+4+2+1 = 63
7	1111111	64+32+16+8+4+2+1 = 127
8	11111111	128+64+32+16+8+4+2+1 = 255

So, 1 byte can represent numbers from
0 to 255 (unsigned), or from -128 to +127 (signed)

Consequence of binary representation of numbers

- Certain types of programs (e.g. databases), and programming languages require you to identify your variable type
- Need to pick a variable of the right kind, and with enough bytes, so that your numbers will fit
- If you don't use enough, numbers can be truncated (i.e. the answers will be wrong), or the program could crash (i.e. you don't get an answer at all)
- Typically, integers are either “short” (16 bit, from -32,768 to 32,767) or “long” (32 bit, from -2,147,483,648 to 2,147,483,647)
- Why not always use long? Why not use 64 bit (numbers into the quintillions)?

Quick quiz

- Which has greater precision?

$$\frac{1}{3}$$

0.333333333

Continuous numbers are recorded at a fixed precision

- Computers use a “floating point” representation for continuous numbers (with decimal places)
- Computers can only store a number to a fixed number of decimal places
  - A standard “float” variable type retains 15 significant digits
- This is *usually* not a problem for our data
  - Rare to actually have a measured digit to the 16th decimal place
  - But, calculations done on the data should retain enough decimal places that rounding doesn't cause problems
- If you are working at the limits of this fixed level of precision, you can run into problems

Floating point numbers

- Easiest to understand in comparison with fixed point decimal numbers
  - Fixed point decimals are explicit about the location of the decimal
  - If there are 5 sig figs, three after the decimal and two before, then a number smaller than 0.001 or larger than 99.999 cannot be represented
- Floating point numbers divide the decimal number into a mantissa (the recorded, significant digits) and an exponent
- Because of this, 12345, 1.2345, 12345000, 0.00012345 can all be represented with the same floating-point structure
  - 12345×10^{1}, 12345×10^{-4}, 12345×10^{3}, 12345×10^{-8}

The structure of floating point numbers

- A number stored as floating point “double” (64-bit) precision has:
  - 1 sign bit (0 for positive, 1 for negative)
  - 11 exponent bits (representing either positive or negative exponents)
  - 52 bits for the mantissa
- Spreadsheets use double-precision floating point decimal numbers, and store 15 significant digits
- What happens if you are working with numbers that are at or beyond this limit?

Examples: floating point weirdness

	A	B
1		
2	A =	0.9999999999989579000
3	B =	0.9999999999989578000
4	A - B =	0.0000000000000000999

	1*(0.5 - 0.4 - 0.1) =	-2.8E-17

	A	B	C	D	E	F
1		A =		B =		A + B =
2		1.20E+100		1.00E+100		2.200000000000000E+100
3				1.00E+99		1.300000000000000E+100
4				1.00E+98		1.210000000000000E+100
5				1.00E+97		1.201000000000000E+100
6				1.00E+96		1.200100000000000E+100
7				1.00E+95		1.200010000000000E+100
8				1.00E+94		1.200001000000000E+100
9				1.00E+93		1.200000100000000E+100
10				1.00E+92		1.200000010000000E+100
11				1.00E+91		1.200000001000000E+100
12				1.00E+90		1.200000000100000E+100
13				1.00E+89		1.200000000010000E+100
14				1.00E+88		1.200000000001000E+100
15				1.00E+87		1.200000000000100E+100
16				1.00E+86		1.200000000000010E+100
17				1.00E+85		1.200000000000000E+100
18						

Consequence of floating-point representation in a computer

- If you are working at the limits of your computer's precision, beware
- Best to pick units of measure that will not put you near these limits
- Floating point math can give you some odd results, even for simple calculations – watch for problems

Representing letters and symbols in binary

- ASCII = American Standard Code for Information Interchange
- This is sometimes called “plain text” - no formatting instructions included
- Numeric codes for 128 specified characters
- First 32 are control characters (things like carriage returns) – used to control the hardware

A selection – symbols and numbers

DEC	OCT	HEX	BIN	Symbol	HTML Number	HTML Name	Description
32	040	20	00100000		&#32;		Space
33	041	21	00100001	!	&#33;		Exclamation mark
34	042	22	00100010	"	&#34;	&quot;	Double quotes (or speech marks)
35	043	23	00100011	#	&#35;		Number
36	044	24	00100100	$	&#36;		Dollar
37	045	25	00100101	%	&#37;		Procenttecken
38	046	26	00100110	&	&#38;	&amp;	Ampersand
39	047	27	00100111	'	&#39;		Single quote
40	050	28	00101000	(	&#40;		Open parenthesis (or open bracket)
41	051	29	00101001	)	&#41;		Close parenthesis (or close bracket)
42	052	2A	00101010	*	&#42;		Asterisk
43	053	2B	00101011	+	&#43;		Plus
44	054	2C	00101100	,	&#44;		Comma
45	055	2D	00101101	-	&#45;		Hyphen
48	060	30	00110000	0	&#48;		Zero
49	061	31	00110001	1	&#49;		One
50	062	32	00110010	2	&#50;		Two
51	063	33	00110011	3	&#51;		Three
52	064	34	00110100	4	&#52;		Four
53	065	35	00110101	5	&#53;		Five
54	066	36	00110110	6	&#54;		Six
55	067	37	00110111	7	&#55;		Seven
56	070	38	00111000	8	&#56;		Eight
57	071	39	00111001	9	&#57;		Nine

ASCII upper and lower case letters

DEC	OCT	HEX	BIN	Symbol	HTML Number	HTML Name	Description
32	040	20	00100000		&#32;		Space
65	101	41	01000001	A	&#65;		Uppercase A
66	102	42	01000010	B	&#66;		Uppercase B
67	103	43	01000011	C	&#67;		Uppercase C
68	104	44	01000100	D	&#68;		Uppercase D
69	105	45	01000101	E	&#69;		Uppercase E
70	106	46	01000110	F	&#70;		Uppercase F
71	107	47	01000111	G	&#71;		Uppercase G
72	110	48	01001000	H	&#72;		Uppercase H
73	111	49	01001001	I	&#73;		Uppercase I
74	112	4A	01001010	J	&#74;		Uppercase J
97	141	61	01100001	a	&#97;		Lowercase a
98	142	62	01100010	b	&#98;		Lowercase b
99	143	63	01100011	c	&#99;		Lowercase c
100	144	64	01100100	d	&#100;		Lowercase d
101	145	65	01100101	e	&#101;		Lowercase e
102	146	66	01100110	f	&#102;		Lowercase f
103	147	67	01100111	g	&#103;		Lowercase g
104	150	68	01101000	h	&#104;		Lowercase h
105	151	69	01101001	i	&#105;		Lowercase i
106	152	6A	01101010	j	&#106;		Lowercase j

Spelling cat in ASCII

Letter	C	A	T
ASCII Decimal code:	67	65	84
Binary number:	01000011	01000001	01010100

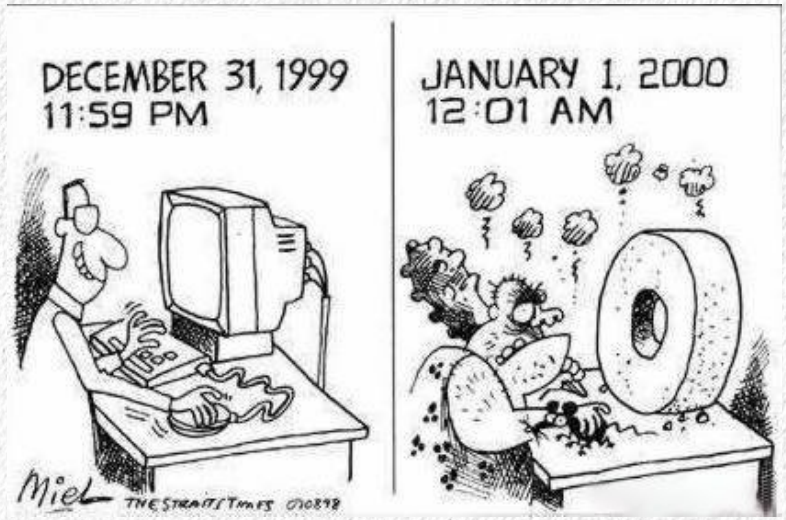
Representing time in a computer

- Time passes continuously
  - We can measure it to an arbitrary level of precision
  - Current limit to our ability to measure time in the real world is the attosecond (10^{-18} s)
  - An attosecond is to a second what a second is to 31.71 billion years
- All clocks can only measure time to a fixed resolution – computers are no different
  - Computers measure time by counting “ticks” that have elapsed since a selected starting time point, called the “epoch”
  - Ticks are 100 nanoseconds
  - This is referred to as “system time”

System time on a computer

- Different computer operating systems use different epochs
  - UNIX-derived systems use 1 January 1970 00:00:00 UT (where UT is time taken at the prime meridian) for the epoch
  - Windows NT used 1 January 1601 00:00:00 UT as its epoch
- Number of ticks before or after the epoch converted to time
  - Positive numbers of ticks are times after the epoch, negative numbers would be times previous to the epoch
- Resolution recorded is usually to the nearest millisecond
- Timekeeping by computers can drift over time, but can synchronize with atomic clock signals, or “time servers” on their network that keep even more accurate time

Problems representing time on computers



- The Millennium Bug
  - Programs that stored dates with only two numbers, and always expected years to increase, were expected to crash when the year went from 99 to 00
  - Solution – reprogram to use four digit years (what will those year 9,999 people think of us?)
- The Unix 2038 problem
  - The Unix epoch is 1/1/1970 00:00:00 UTC
  - If time is stored as a 32-bit signed integer, the largest number of ticks that can be recorded is $2^{31} - 1$
  - This maximum will be reached in 2038
  - Solution: reprogram to use 64-bit integers for times

System time is used to generate random numbers

- Computers follow instructions exactly, will always return the same output given the same input
- How do you get a computer to do something spontaneous?
- Use pseudo-random numbers
  - Algorithms that take a starting “seed” number and produce unpatterned numbers from it
  - Identical sets of numbers with the same seed
  - To produce random numbers, select a different seed each time – usually use system time as the seed

Imaging

Consider this picture

What would it look like if we only used 1 bit to represent it?

1 bit image

1 bit, the only options are 0 (black) and 1 (white)

Anything above a threshold level of lightness will be assigned to 1, and will appear white

Anything below the threshold will be assigned a 0, and will appear black

Grayscale – 1 channel with 8 bits

8 bits (0 to 255) for a single channel

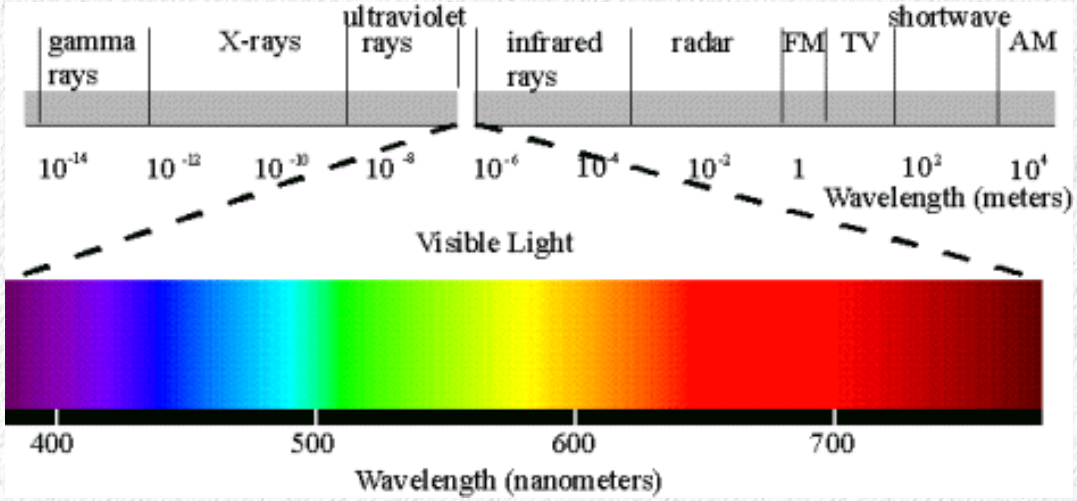
Interpreted as 256 shades of gray

Sufficient number for the transition from shade to shade to appear smooth

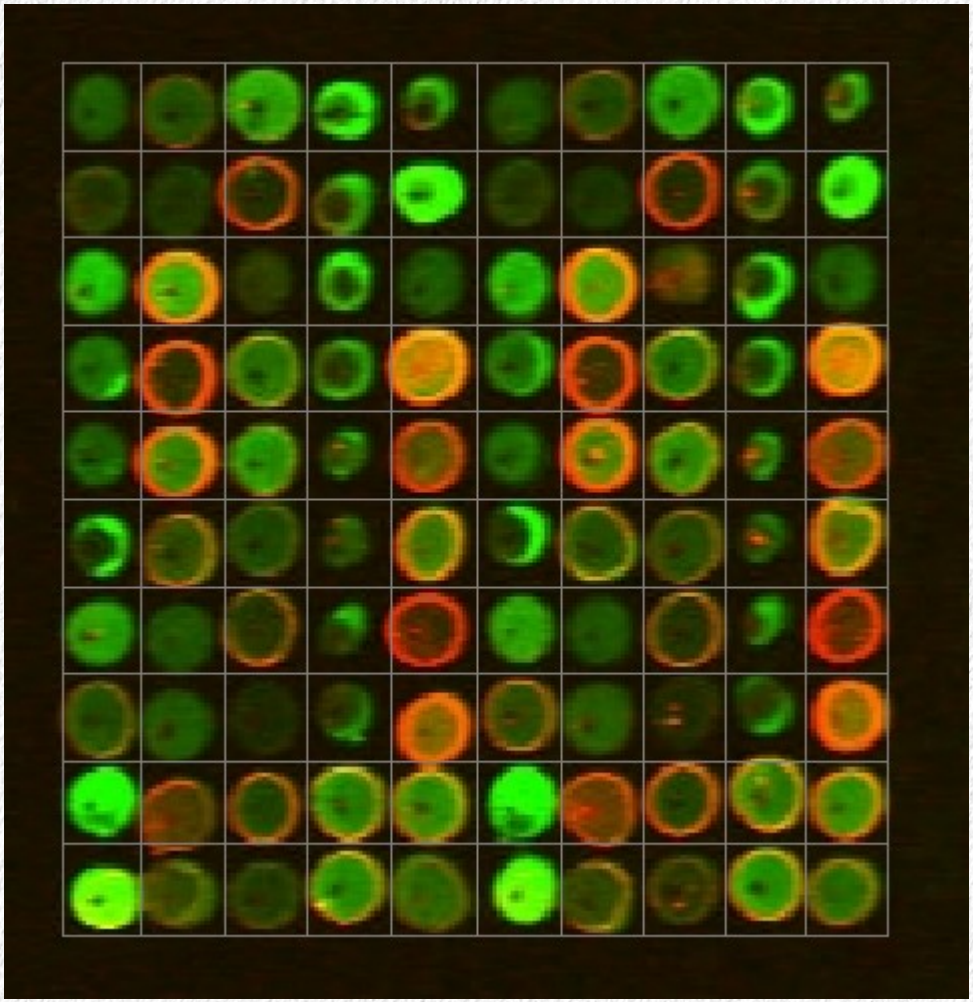
But, there are more than 256 possible shades of gray, so some get lumped together into one – some loss of detail in the image compared to the actual, analog object

Representing color on a computer

- Color can also be used as data
  - Colors on a microarray may indicate gene expression
  - Medical imaging
  - Remote sensing
- Color is determined by the wavelength of visible light – continuous, analog quantities
- Colors can be represented in computers using an appropriate color model



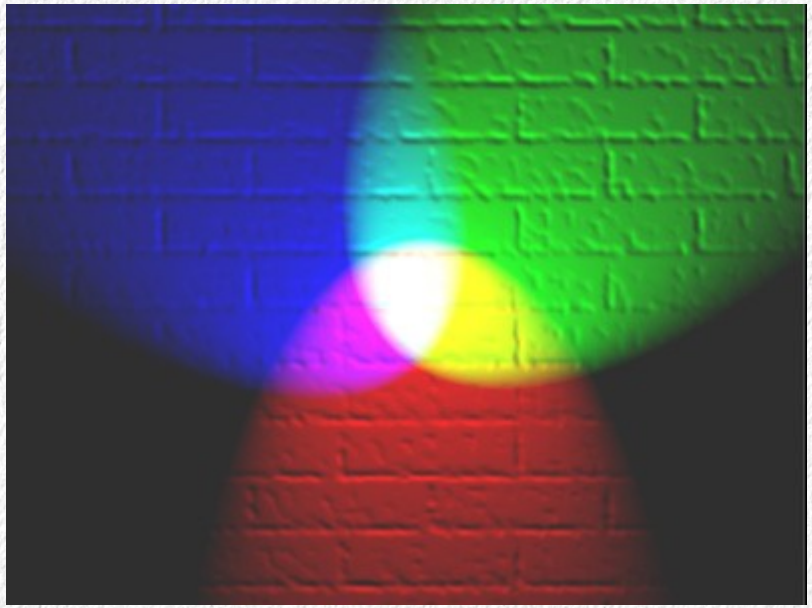
Colors into data - microarrays



- Measure expression across many genes at once in two groups
- Genes that are expressing one product have a green color, those expressing another are red, both are yellow, neither product are black
- Intensity and shade of color is used to indicate amount of expression

The RGB color model

- All colors are represented as combinations of levels of intensity of red, green, and blue light
- Computer displays can emit each color “channel” at different intensities



- Each channel records a value from 0 to 255 (8 bit unsigned integers), so 256 levels of intensity possible

24-bit color: R, G, and B channels

Red

Blue

Green

True color = three channels with 8 bits each

8 bits (0 to 255) for each color channel (R,G,B) = 24 bits

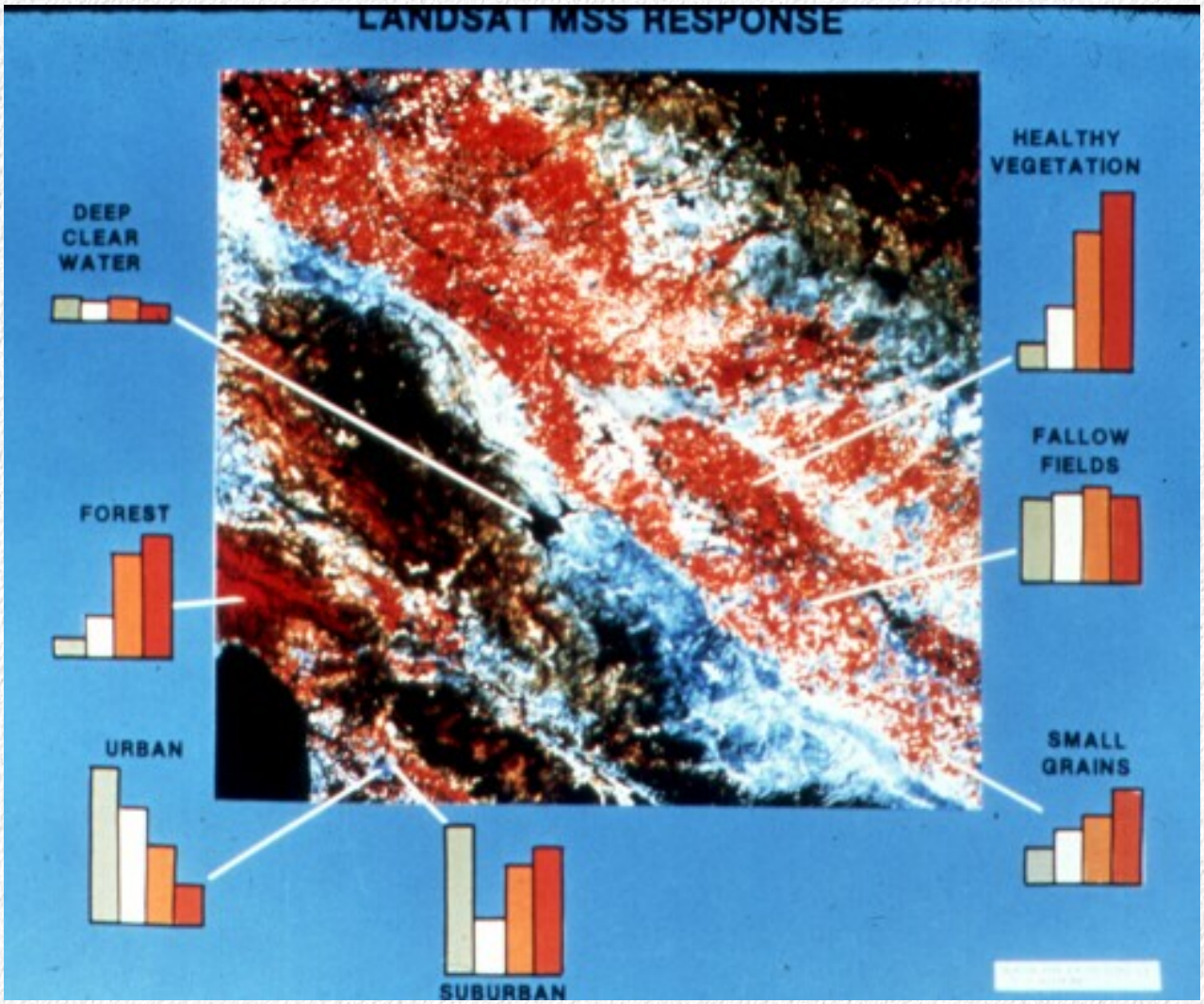
How many colors?

256^3 = 16,777,216 different possible colors

Fine for human vision – way more than the 1 million colors we see

May not be good enough for some scientific applications

Spectral signatures – turning colors into data



Using color to identify different land uses in a landscape

Bars are reflectance in a particular range of wavelengths (three visible and infrared here)

Each land cover type has a unique set of bars

Once the signature of a cover type is known, pixels can be assigned to the cover type that most closely matches its signature

Recording digital images

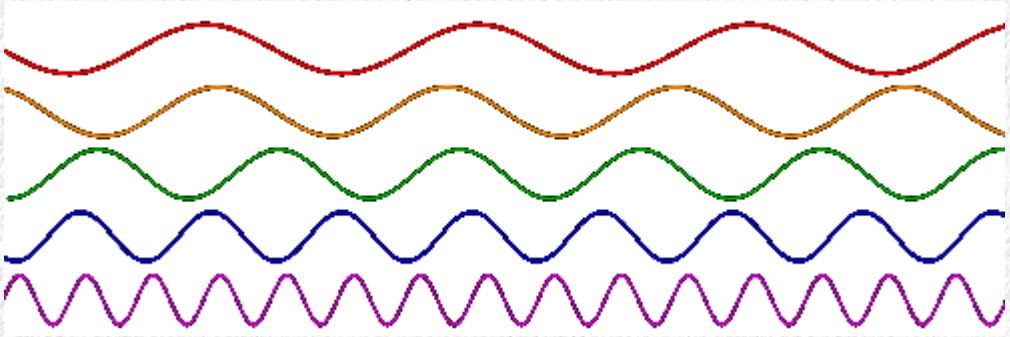
- Light sensors record intensity of particular wavelengths
  - Visible light (true color) = R,G,B
  - Can include other wavelengths (IR, UV) to produce “false color” images
- Recordings are made in a rectangular array, with some number of rows and columns of readings
- Each recorded location is a pixel on the image
- A sensor used to record 1080p high-definition video uses 1920 columns x 1080 rows of sensors
- This produces 2,073,600 pixels per image, or 2.1 megapixels per image

Resolution of an image

- To look lifelike, and avoid losing detail, the pixels in an image should not be individually visible
- Images have fixed numbers of pixels – at a large enough magnification you will see the pixels
- The appropriate resolution depends on the magnification

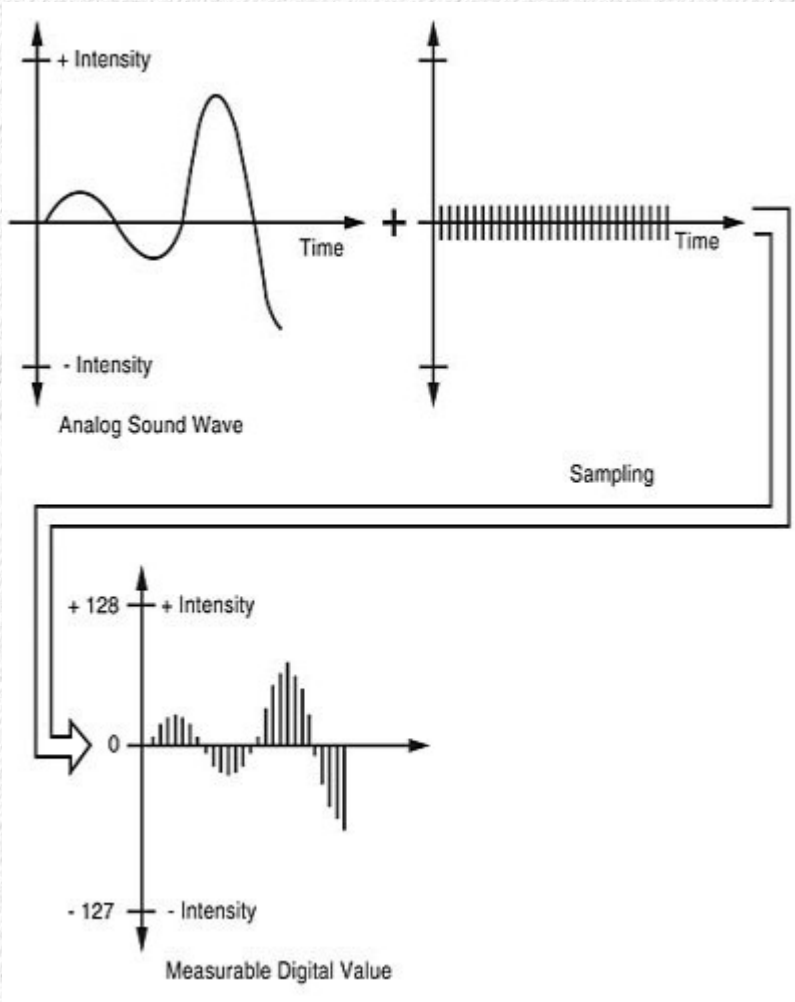
Representing sound on a computer

- Sound is a mechanical wave transmitted through the air (or a liquid or solid)
- Waves are continuous in time
- The tone we hear is determined by the wave form
  - Frequency determines pitch
  - Amplitude determines volume
- We hear sounds between 20 Hz (low pitch) and 20,000 Hz (high pitch), where Hz is cycles per second



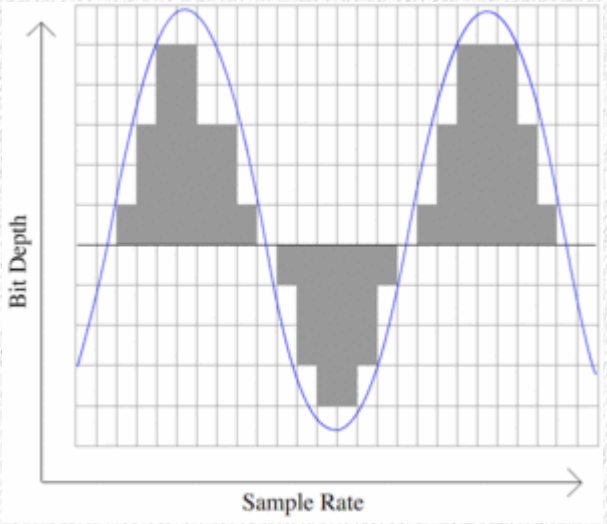
Digital representation of sound

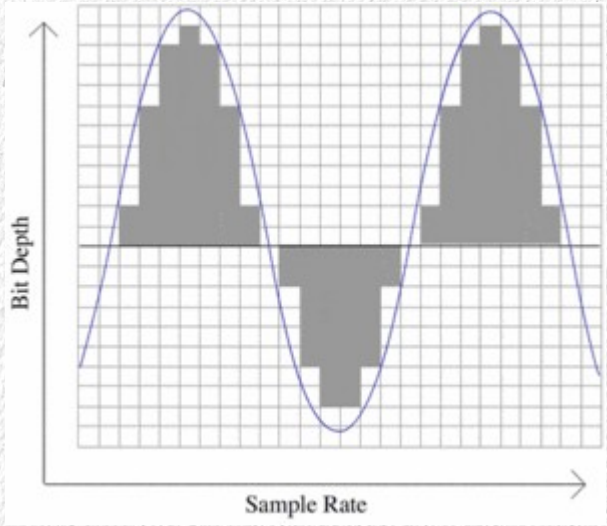
- To digitize the analog sound, samples of intensity are taken over time
- Sampling rate of 44.1 kHz is used for CD's (more than twice the highest audible frequency)
- Instead of a continuous wave, digital sound is thus a series of discrete intensities
- These intensities are passed through a digital to analog converter to produce sound



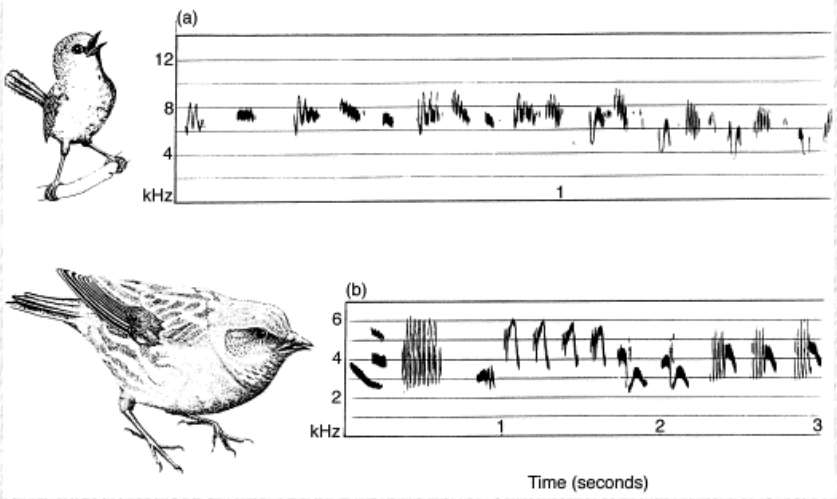
Digital sound fidelity

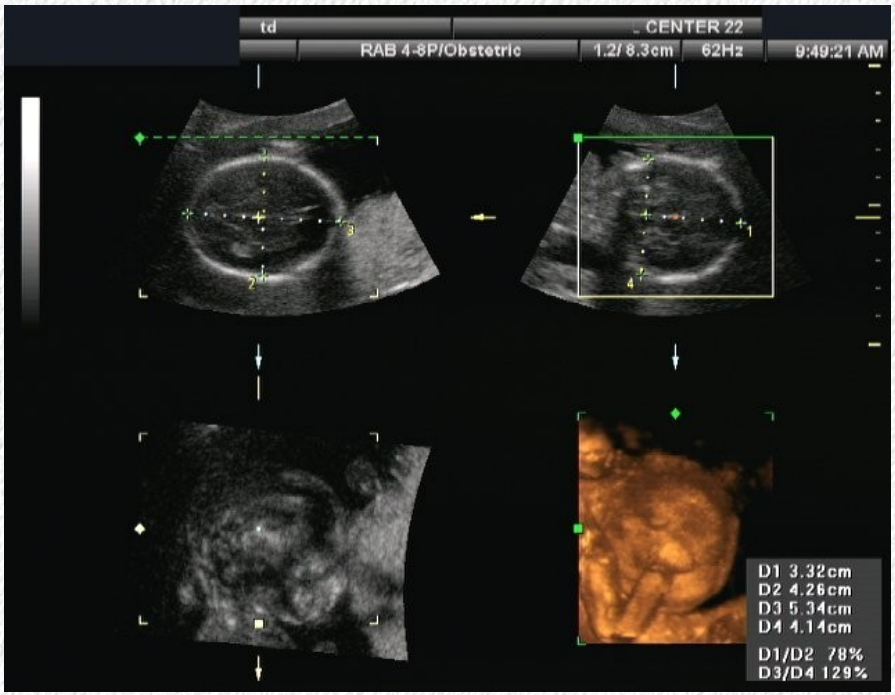
- Two characteristics
  - Sampling rate = how many times per second the sound is sampled
    - Higher sampling rates approximate the wave form at a greater number of time points.
  - Bit depth = how many bits are used to store the intensity
    - More bits means more different intensity levels can be used
- The combination of high sampling rate and large bit depth does the best job of reproducing a sound wave
- But, this also leads to the largest file sizes





Sound data in biology





- Animal vocalizations
- Ultrasound sonograms
- Sensory physiology

File size issues

- Converting analog to digital representation can result in huge amounts of digital data
- A single audio CD with 80 minutes of uncompressed audio is 700 MB
- A single 8 megapixel, 24 bit image file in an uncompressed file format is 24 MB
- Each second of uncompressed high definition video is 118 MB, each minute is 7.1 GB
- Storage space quickly becomes problematic – various methods are used to compress these files

“Lossy” and “lossless” formats

- There are a couple of major approaches to compression of audio and video:
  - Lossless: Retain all of the recorded information, but represent it in a way that requires fewer binary digits
  - Lossy: Throw out some of the recorded data
- Good lossy compression only throws out the “unimportant” parts of the data
  - MP3 compressed sound files throw out frequencies that humans can't hear
  - We'll look at jpeg compression as an example

A simple lossless compression scheme

- If you have a string of 1's and 0's that looks like this:

  `10000011111111000001110000000111111`

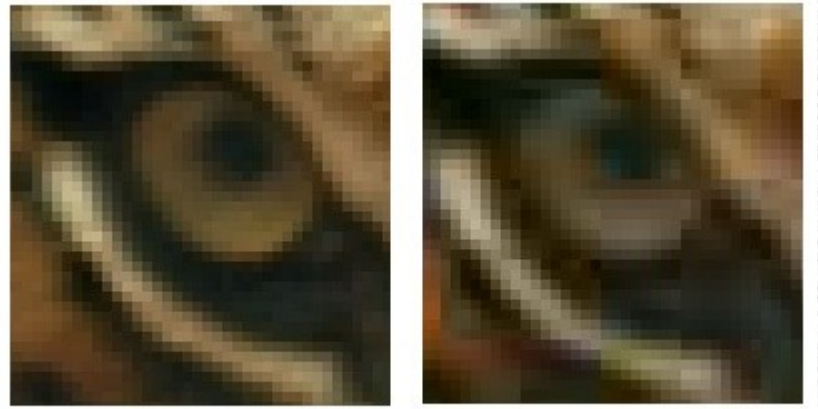
- It's more compact to write it like this:

  `1@10@51@80@51@30@71@6`

- This representation will save space to the extent that 1's and 0's tend to repeat frequently – it's not effective if they alternate frequently

Lossy compression of an image – the jpeg format

- Colors that are similar in a region are assigned to the same color
  - How similar they have to be is determined by the quality setting used
- They are then represented in a condensed way (like the lossless scheme shown before)
- The results can look fine if the image is not enlarged, look pixelated at high magnification



original image

lossy JPEG format with "artifacts"

File formats

- Two issues in choosing a file format for your data
  - Issues with using lossy formats (vs. lossless or uncompressed formats)
  - Openness of standard

Open file formats are best for scientific work

- File formats are created by people, and can be patented
- Proprietary formats = the specification for the file format is not made public
  - Only the owner of the file format can write software to read the files
  - If the maker of the software you use goes out of business, your data may become unreadable if it's in a proprietary format
- Open standards = file formats that are sufficiently well documented that any programmer can write a program to read and write the file
  - Not dependent on one company for access to your own data
- It's best to select formats for your data files that are open standards so that your data will always be available

Lossy vs. lossless in digital data

- Lossless compression is less effective at saving storage space, but retains the original, measured data
- Lossy compression may be okay, but have to know how it works – for example
  - mp3 throws out frequencies out of human audible range, but if you study vocalization in a species that can hear higher or lower frequencies it's not suitable
  - jpeg compression is designed to maintain the appearance of the image on a web site, but for medical imaging or GIS data you want each pixel to be accurate