Images and Display

What is an image?

- A photographic print
- A photographic negative?
- This projection screen
- Some numbers in RAM?

An image is:

- A 2D distribution of intensity or color
- A function defined on a two-dimensional plane

$$I : \mathbb{R}^2 \rightarrow \ldots$$

- Note: no mention of pixels yet
- To do graphics, must:
  - represent images – encode them numerically
  - display images – realize them as actual intensity distributions

Displays

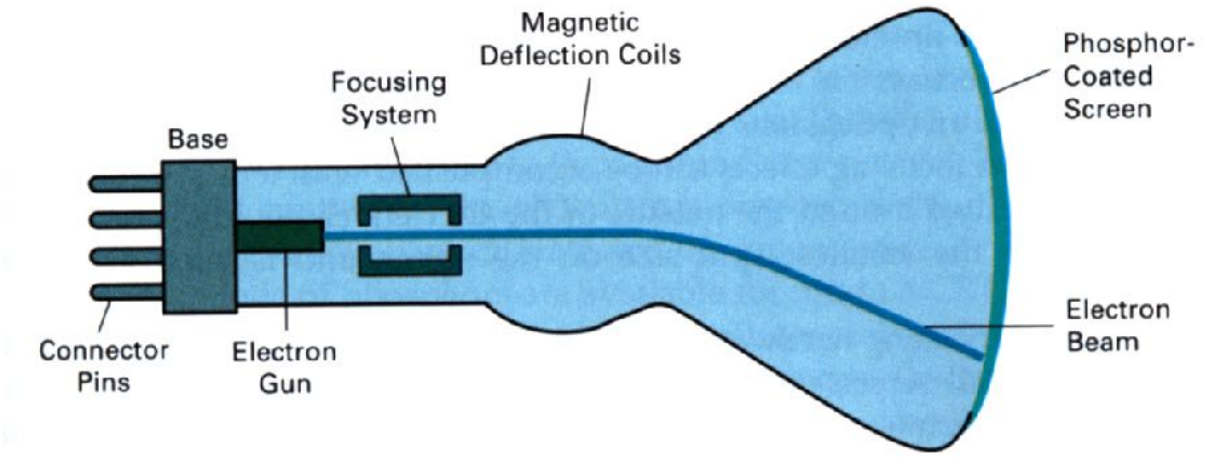
Display technologies

- Direct-view displays
  - Raster CRT display
  - LCD display
  - LED display

- Printers
  - Laser printer
  - Inkjet printer

Cathode ray tube

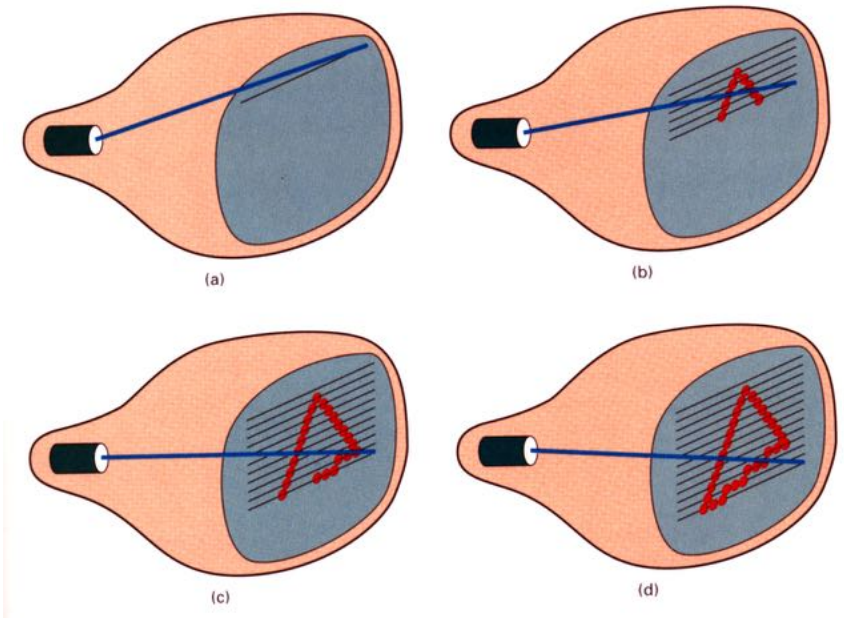
- First widely used electronic display
  - developed for TV in the 1920s–1930s



[H&B fig. 2-2]

Raster CRT display

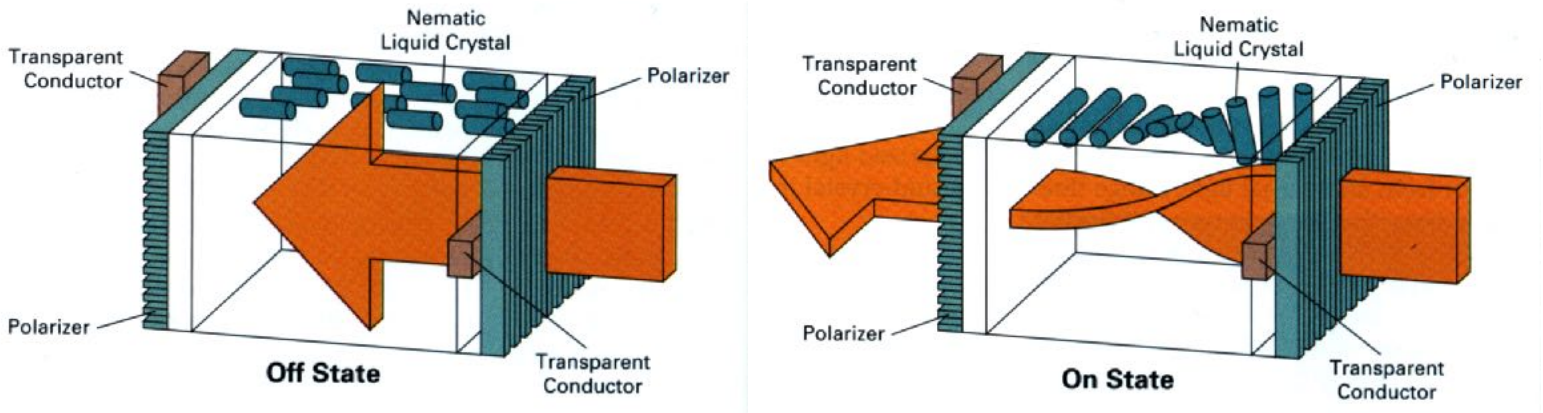
- Scan pattern fixed in display hardware
- Intensity modulated to produce image
- Originally for TV, continuous analog signal)
- For computer, intensity determined by contents of framebuffer



[H&B fig. 2-7]

LCD flat panel display

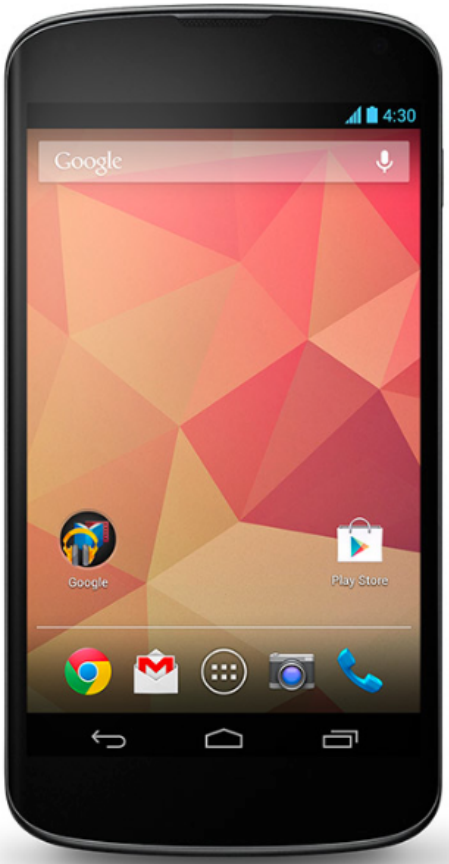
- Principle: block or transmit light by twisting its polarization
- Illumination from backlight, either fluorescent or LED
- Intermediate intensity levels possible by partial twist
- Fundamentally raster technology



[H&B fig. 2-16]

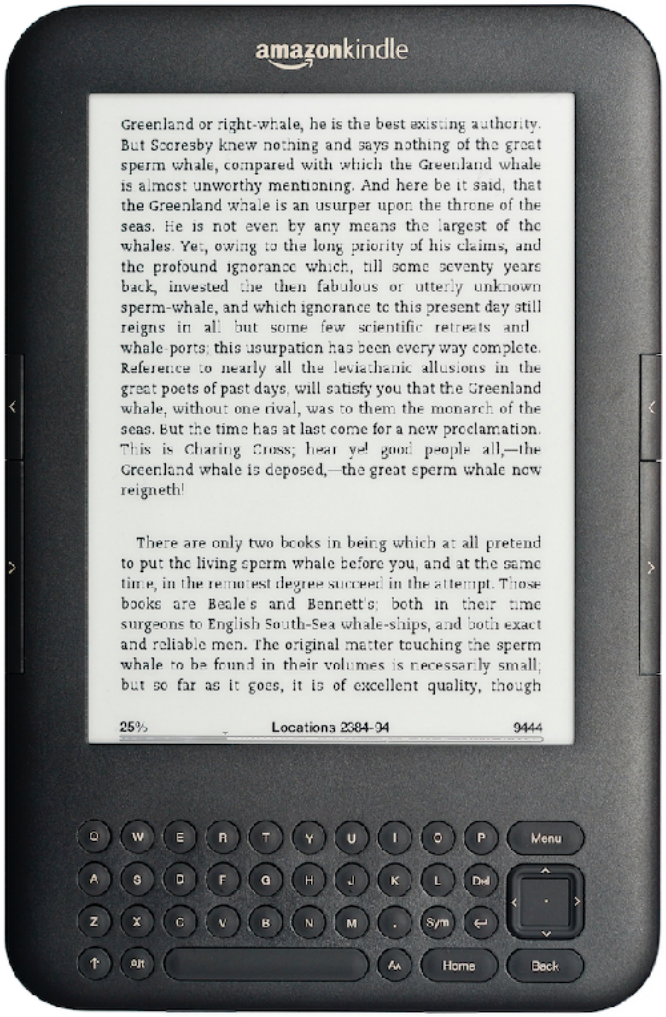
LED Displays

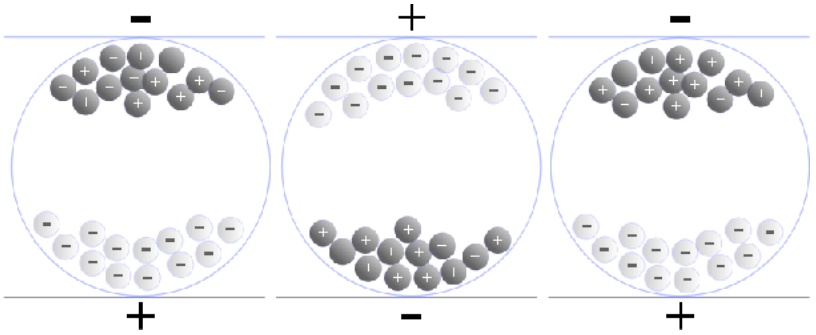
[Wikimedia Commons]



[Google—Nexus 4]

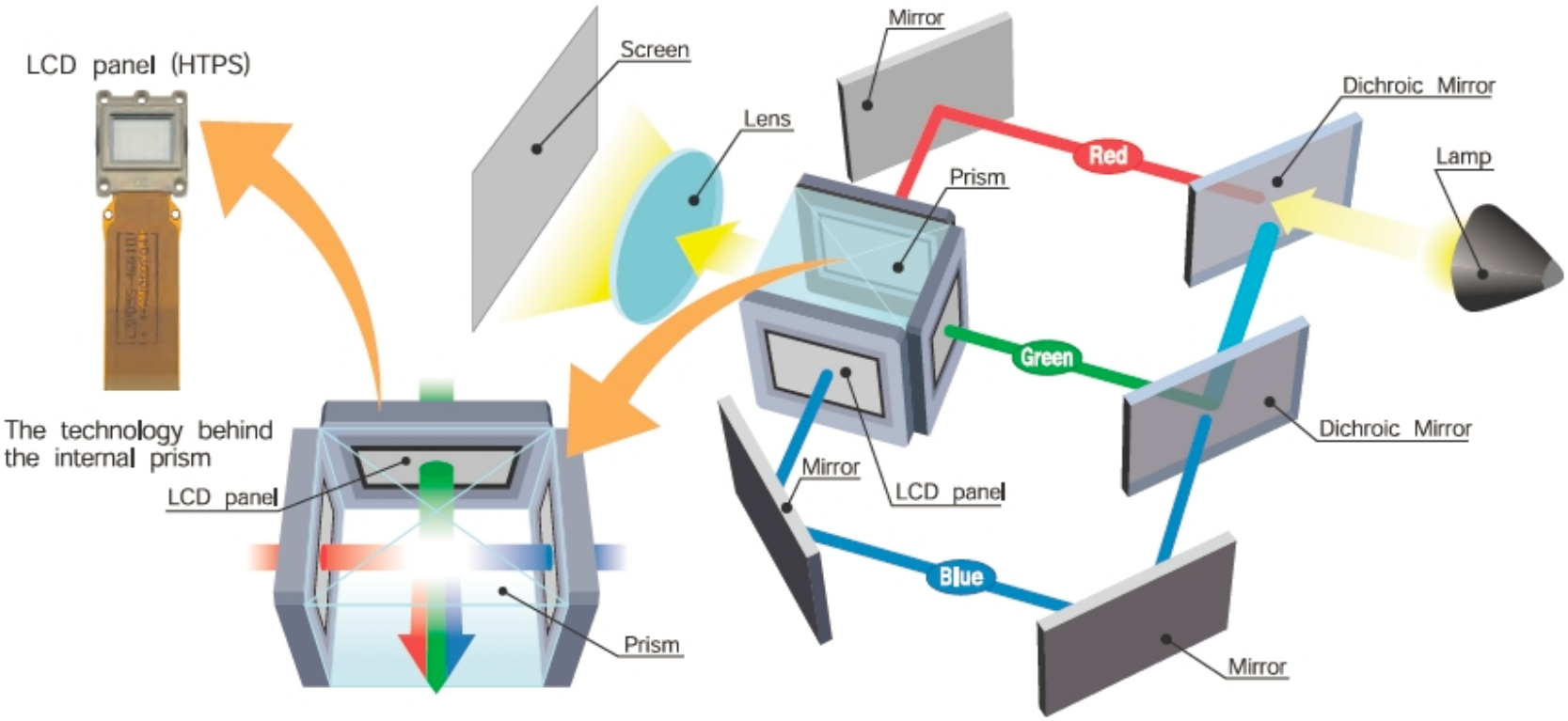
Electrophoretic (electronic ink)





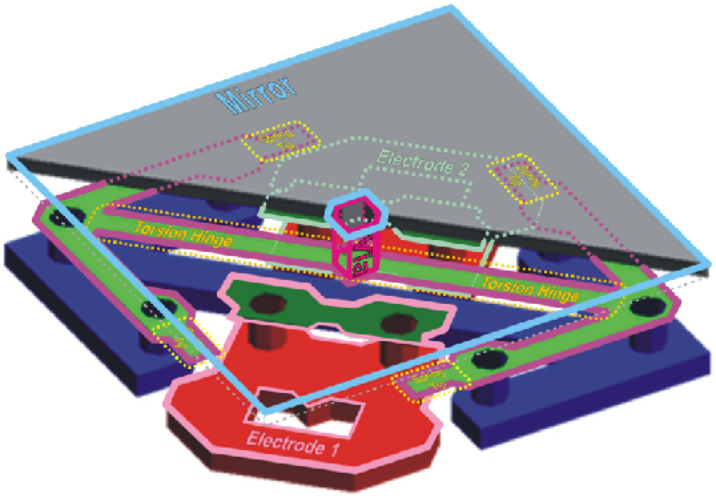
[Wikimedia Commons—Senarclens]

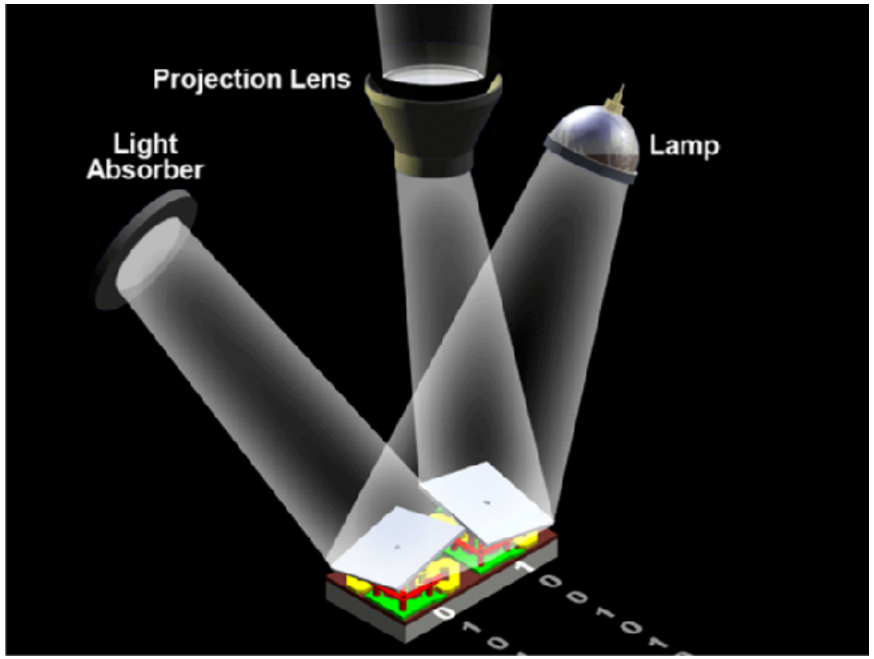
Projection displays: LCD



[Wikimedia Commons—Javachan]

Projection displays: DLP

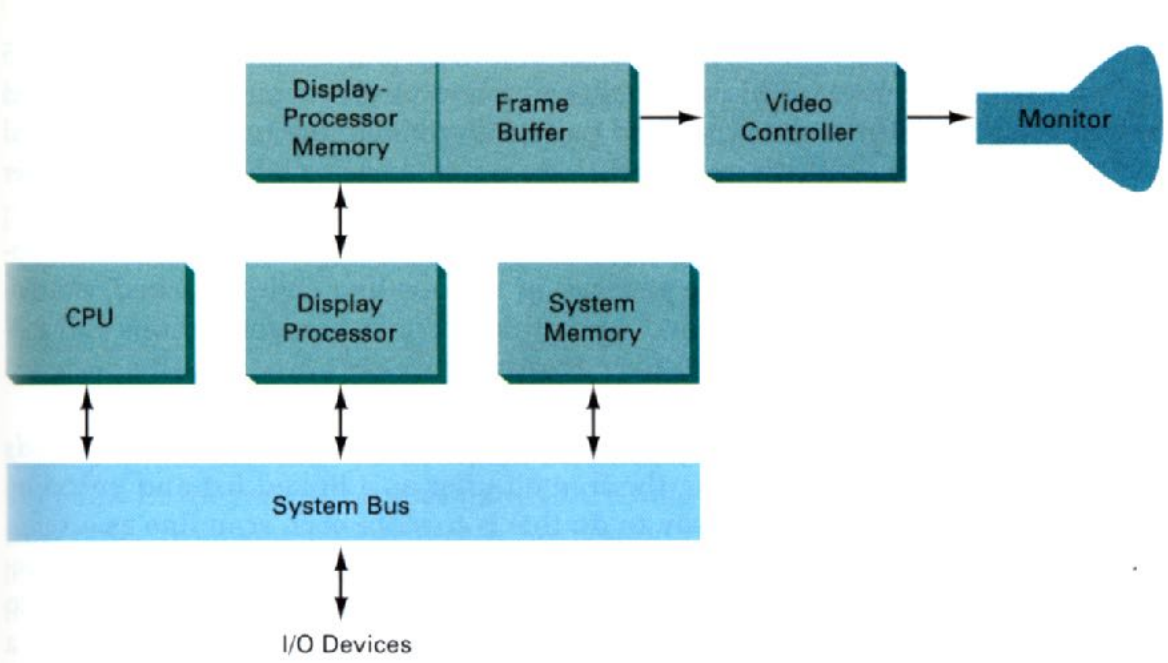




[Texas Instruments]

Raster display system

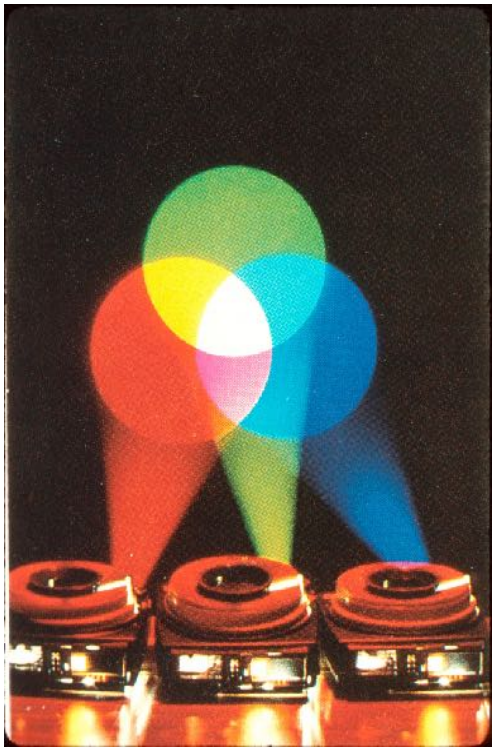
- Screen image defined by a 2D array in RAM
- In most (but not all) systems today, it's in a separate memory from the normal CPU memory
- The memory area that maps to the screen is called the frame buffer



[H&B fig. 2-29]

Color displays

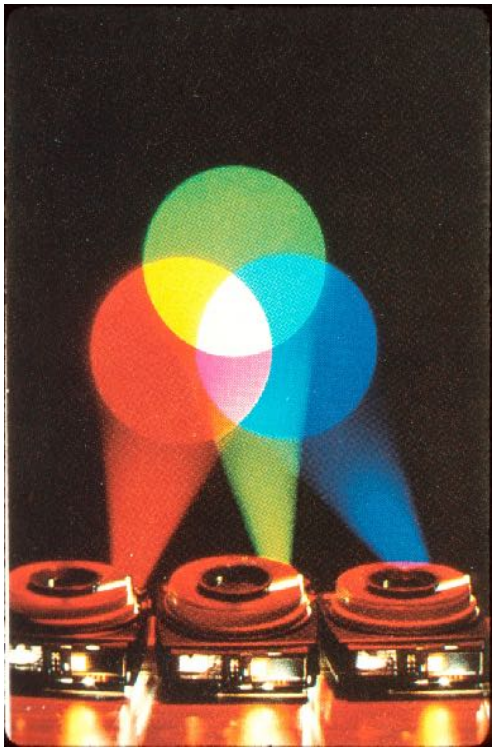
- Operating principle: humans are trichromatic
  - match any color with blend of three
  - problem reduces to producing 3 images and blending



[cs417 S02 slides]

Color displays

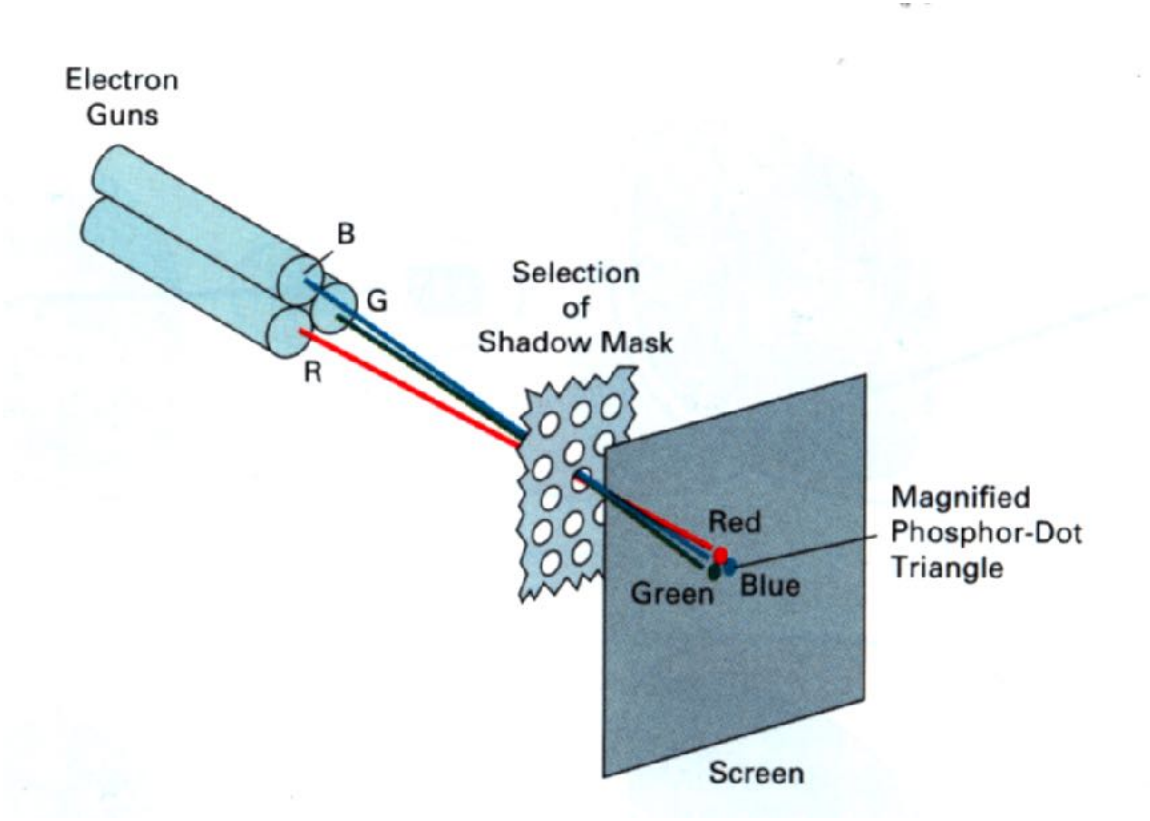
- Additive color: blend images by sum
  - e.g. overlapping projection, unresolved dots
  - R, G, B make good primaries



[cs417 S02 slides]

Color displays

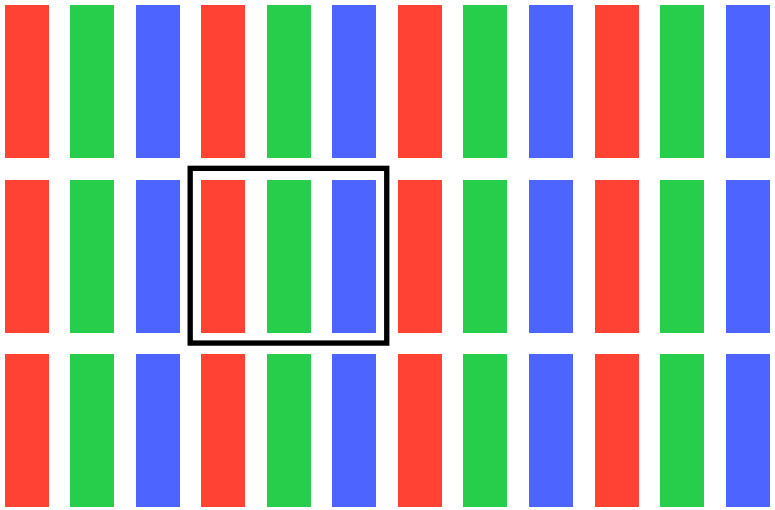
- CRT: phosphor dot pattern to produce finely interleaved color images

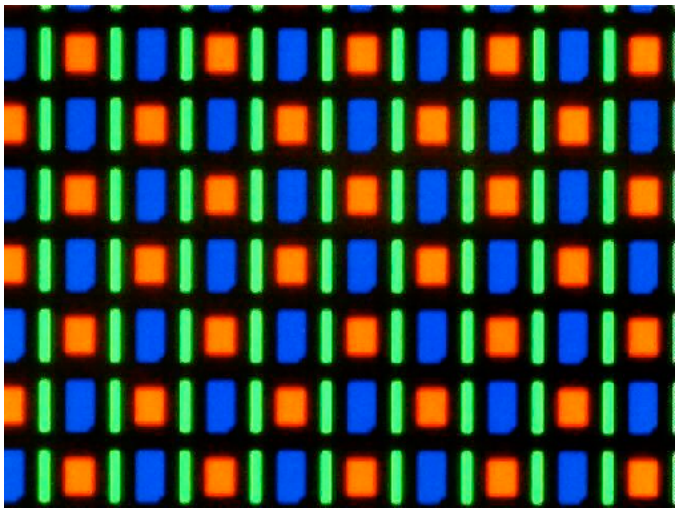


[H&B fig. 2-10]

Color displays

- LCD, LED: interleaved R,G,B pixels

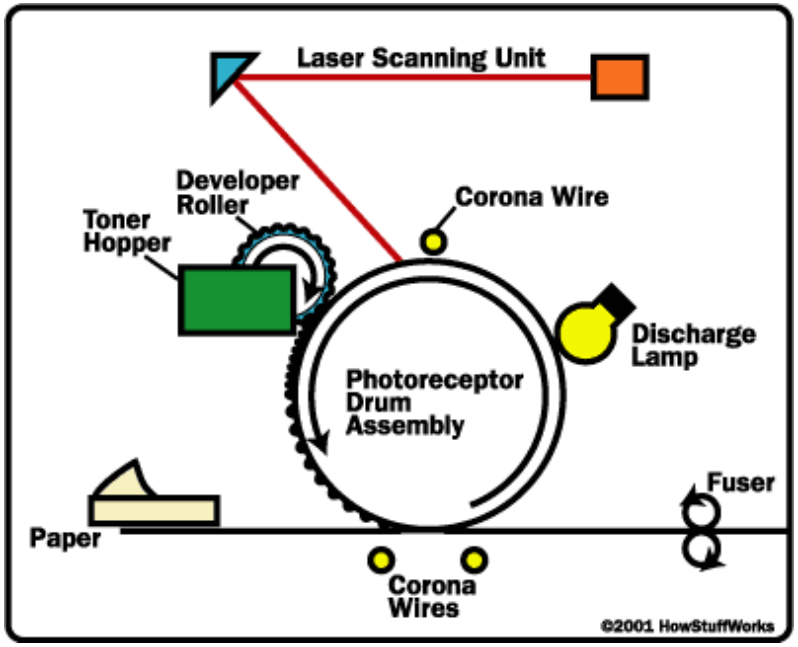




[Wikimedia Commons]

Laser printer

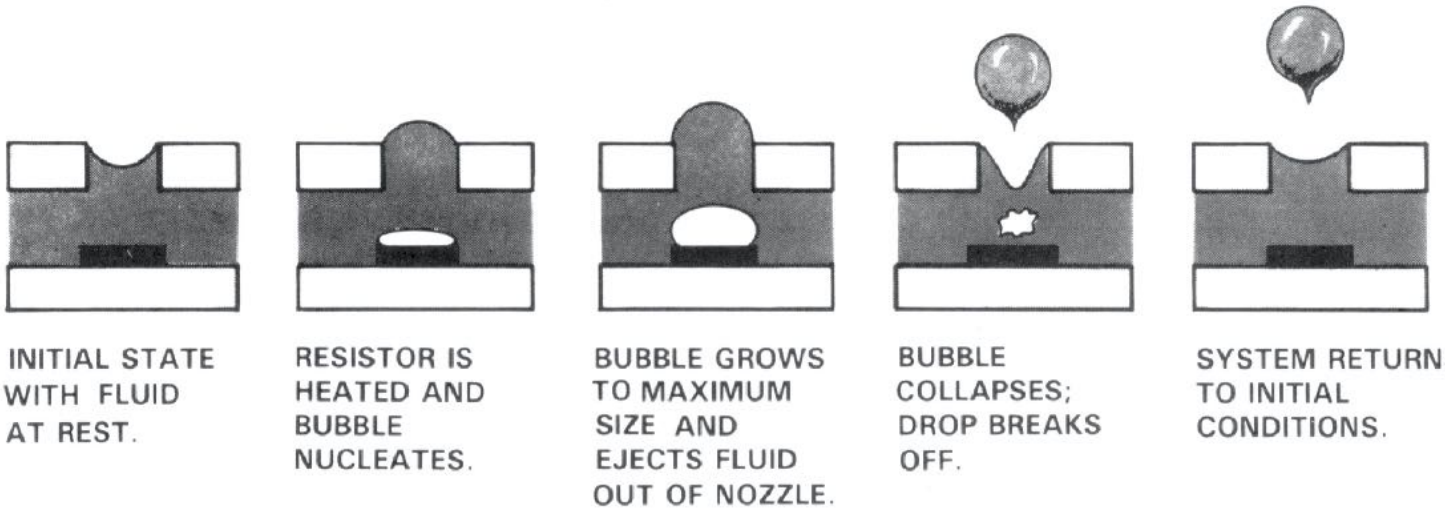
- Xerographic process
- Like a photocopier but with laser-scanned raster as source image
- Key characteristics: image is binary, resolution is high, very small, isolated dots are not possible



[howstuffworks.com]

Inkjet printer

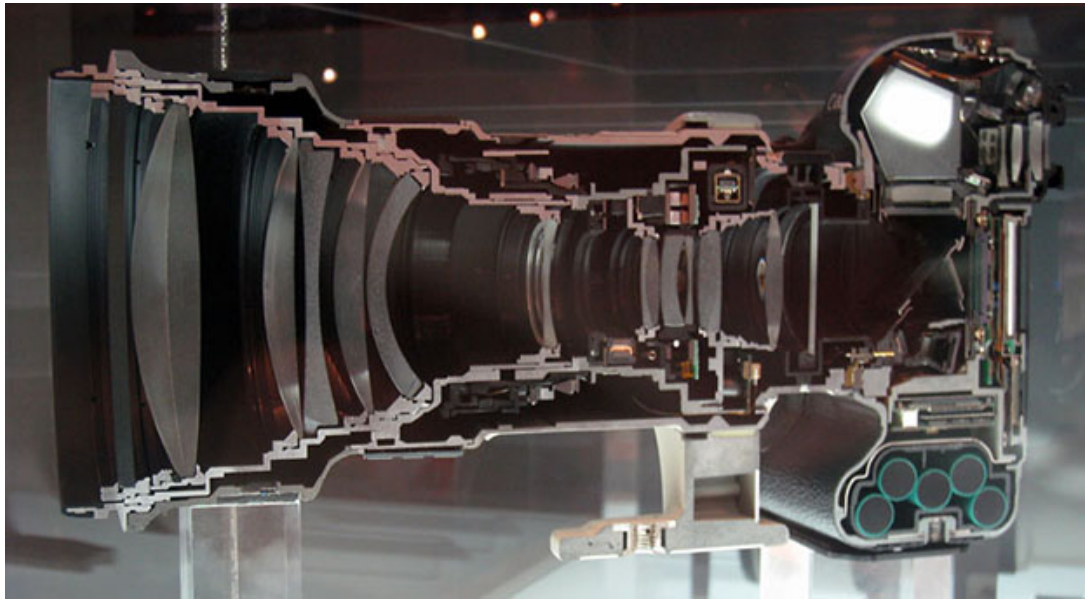
- Liquid ink sprayed in very small drops (picoliters)
- Head with many jets scans across paper
- Key characteristics: image is binary (no partial drops), isolated dots are reproduced well



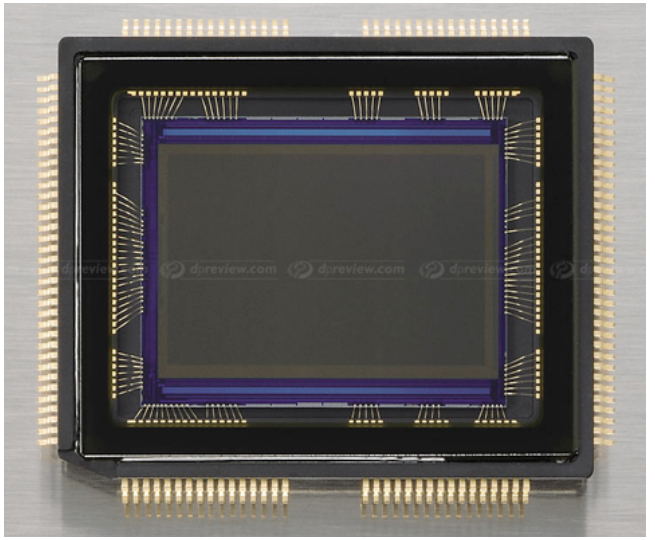
[cs417 S02 slides]

Digital camera

- A raster input device
- Image sensor contains 2D array of photosensors



[PetaPixel.com]



[dpreview.com]

Digital camera

- Color typically captured using color mosaic

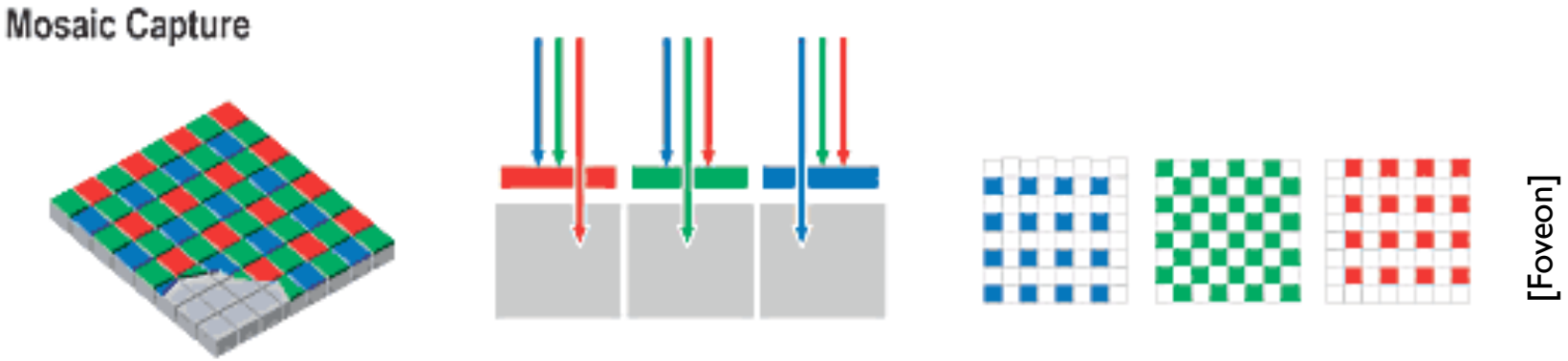


Image Representation

Raster image representation

- All these devices suggest 2D arrays of values, called pixels
- Big advantage: represent arbitrary images
- Approximate arbitrary functions with increasing resolution

Meaning of a raster image

- Meaning of a given array is a function on 2D
- Define meaning of array = result of output device?
  - that is, piecewise constant for LCD, blurry for CRT
  - but: we don't have just one output device
  - but: want to define images we can't display (e.g. too big)
- Abstracting from device, problem is reconstruction
  - image is a sampled representation
  - pixel means "this is the intensity around here"
    - LCD: intensity is constant over square regions
    - CRT: intensity varies smoothly across pixel grid

Image Data Structure

- Option 1: Store images as 2D arrays
  - Not always supported by the programming language

```
struct image {
    DataType pixels[][];
}
```

Image Data Structure

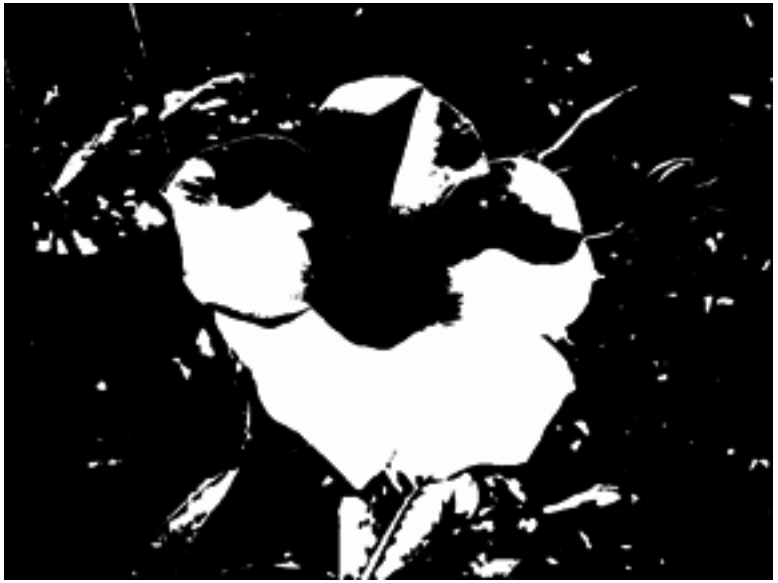
- Option 2: Store image contiguously, line-by-line
  - Most common representation
  - Access pixels manually

```
struct image {
  int width, height;
  DataType pixels[];

  DataType getPixel(int i, int j) {
    return pixels[j*width+i];
  }

  void setPixels(int i, int j, DataType v ) {
    pixels[j*width+i] = v;
  }
}
```


Datatypes for raster images

- Bitmaps: boolean per pixel (1 bpp)
  - interp. = black and white; e.g. fax

$$I : \mathbb{R}^2 \to \{0, 1\}$$



Datatypes for raster images

- Grayscale: integer per pixel
  - interp. = shades of gray; e.g. black-and-white print
  - precision: usually byte (8 bpp); sometimes 16 bpp

$$I : \mathbb{R}^2 \to [0, 1]$$

Datatypes for raster images

- Color: 3 integers per pixel
  - interp. = full range of displayable color; e.g. color print
  - precision: usually byte[3] (24 bpp), sometimes short[3]

$$I : \mathbb{R}^2 \rightarrow [0,1]^3$$

Datatypes for raster images

- Color: can we use less then 8 bits?
  - interp. = visible color banding
  - precision: example 12 bpp

$$I : \mathbb{R}^2 \to [0, 1]^3$$

Datatypes for raster images

- Floating point: 3 floats per pixel
  - 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
  - current HDR TVs: 10–12 bits per channel
  - will discuss them later

$$I : \mathbb{R}^2 \to \mathbb{R}^3_+$$

Storage requirements

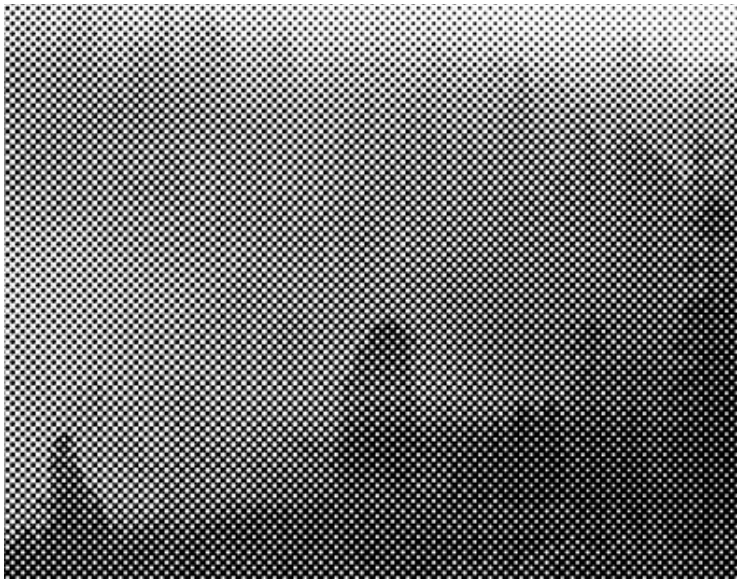
- 1024x1024 image (1 megapixel)
  - bitmap: 128KB
  - grayscale 8bpp: 1MB
  - grayscale 16bpp: 2MB
  - color 24bpp: 3MB
  - floating-point HDR color: 12MB

Dithering

- When decreasing bpp, we quantize
- Make choices consistently: banding
- Instead, be inconsistent—dither
  - turn on some pixels but not others in gray regions
  - a way of trading spatial for tonal resolution
  - choose pattern based on output device
  - laser, offset: clumped dots required (halftone)
  - inkjet, screen: dispersed dots can be used

Diffusion Dither

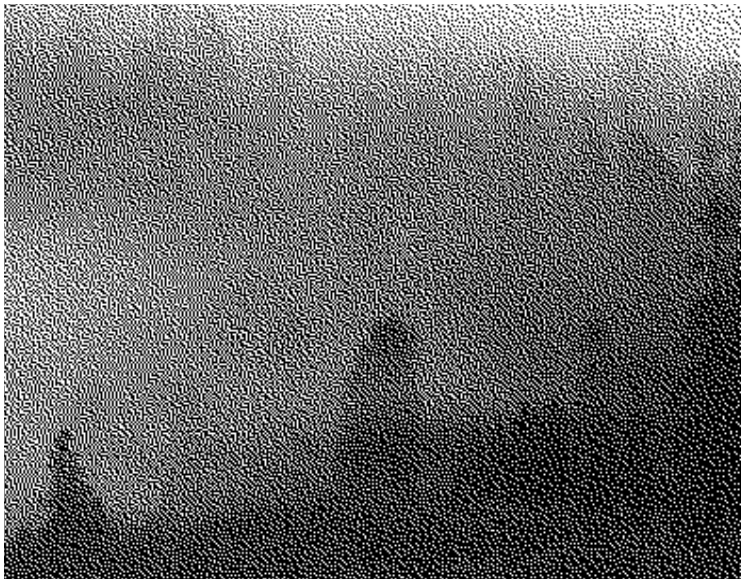
- Produces regular grid of compact dots
  - based on traditional, optically produced halftones
  - produces larger dots



[photo: Philip Greenspun]

Ordered Dither

- Produces scattered dots with the right local density
  - takes advantage of devices that can reproduce isolated dots
  - the modern winner for desktop printing



[photo: Philip Greenspun]

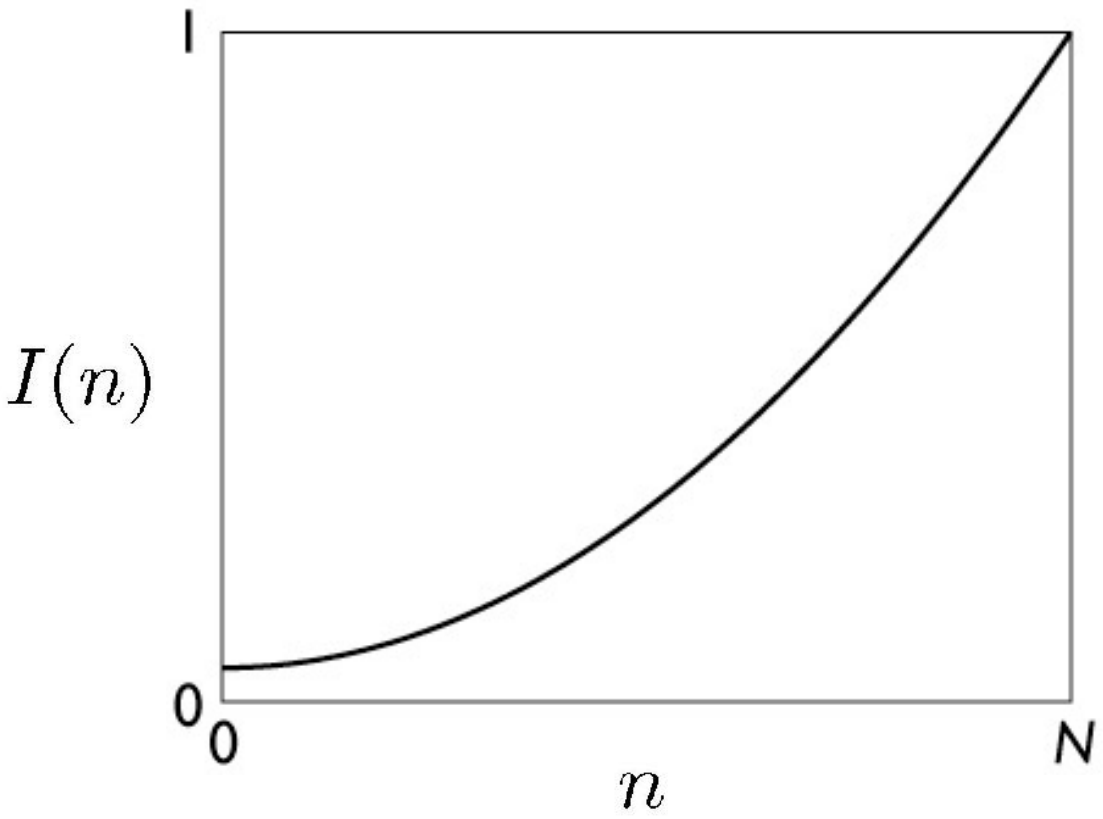
Intensity encoding in images

- What do the numbers in images (pixel values) mean?
- They determine how bright that pixel is
  - bigger numbers are brighter
- In HDR images (float), they are linearly related to the intensity
- In LDR images (integers), this mapping is not direct
  - transfer function *f*: function that maps input pixel values *n* to output luminance *I* of displayed image
  - determined by physical constraints of device and desired visual characteristics

$$I = f(n) \quad f : [0, N] \rightarrow [I_{min}, I_{max}]$$

Typical Transfer Function

$$I = f(n) \quad f : [0, N] \rightarrow [I_{min}, I_{max}]$$



Transfer function limits

- Maximum displayable intensity, Imax
  - how much power can be channeled into a pixel?
  - LCD: backlight intensity, transmission efficiency (<10%)
  - projector: lamp power, efficiency of imager and optics
- Minimum displayable intensity, Imin
  - light emitted by the display in its "off" state
  - CRT: stray electron flux,
  - LCD: polarizer quality

Transfer function limits

- Viewing flare, k: light reflected by the display
  - very important factor determining image contrast in practice
  - 5% of Imax is typical in a normal office environment [sRGB spec]
  - much effort to make very black CRT and LCD screens
  - all-black decor in movie theaters

Dynamic range

- Dynamic range: ratio between max and min display values
  - determines the degree of image contrast that can be achieved
  - a major factor in image quality

$$R_d = \frac{I_{max}}{I_{min}}$$

- Under non-ideal viewing condition, light is present in the environment, so dynamic range is reduced

$$R_d = \frac{I_{max} + I_{amb}}{I_{min} + I_{amb}}$$

Dynamic range

- Ballpark values
  - Desktop display in typical conditions: 20:1
  - Photographic print: 30:1
  - Desktop display in good conditions: 100:1
  - High-end display under ideal conditions: 1000:1
  - Digital cinema projection: 1000:1
  - Photographic transparency (directly viewed): 1000:1
  - High dynamic range display: 10,000:1

Transfer function shape

- Desirable property: the change from one pixel value to the next highest pixel value should not produce a visible contrast
  - otherwise smooth areas of images will show visible bands
- What contrasts are visible?
  - rule of thumb: under good conditions we can notice a 2% change in intensity
  - therefore we generally need smaller quantization steps in the darker tones than in the lighter tones
  - most efficient quantization is logarithmic

How many levels are needed?

- 2% steps are most efficient
  - log 1.02 is about 1/120
  - 120 steps per decade of dynamic range

$$0 \mapsto I_{min}; 1 \mapsto 1.02 I_{min}; 2 \mapsto (1.02)^2 I_{min}; ...; n \mapsto (1.02)^n I_{min}$$

- Number of steps depends on dynamic range
  - 240 for desktop display
  - 360 to print to film
  - 480 to drive HDR display

How many levels are needed?

- If we want to use linear quantization (equal steps)
  - one step must be < 2% (1/50) of lmin
  - 1500 for a print
  - 5000 for desktop display
  - 500,000 for HDR display
- Moral: 8 bits is just barely enough for low-end applications
  - but only if we are careful about quantization

Intensity quantization

- Option 1: linear quantization
  - pro: simple, convenient, amenable to arithmetic
  - con: requires more steps (wastes memory)
  - need 12 bits for any useful purpose; more than 16 for HDR

$$I(n) = (n/N)I_{max}$$

- Option 2: power-law quantization
  - pro: fairly simple, approximates ideal exponential quantization
  - con: linearize before arithmetic, must agree on exponent
  - 8 bits are OK for many applications; 12 for more critical ones

$$I(n) = (n/N)^{\gamma} I_{max}$$

Intensity quantization

- Option 3: floating-point quantization
  - floating points are also quantized to finite precision
  - pro: close to exponential; no parameters; arithmetic
  - con: takes 32 or 16 bits

$$I(x) = (x/w)I_{max}$$

Gamma Correction

- Power-law quantization, or gamma correction is most popular
- Original reason: CRTs are like that
  - intensity on screen is proportional to (roughly) voltage^2
- Continuing reason: inertia + memory savings
  - inertia: gamma correction is close enough to logarithmic that there's no sense in changing
  - memory: gamma correction makes 8 bits per pixel an acceptable option

Gamma correction

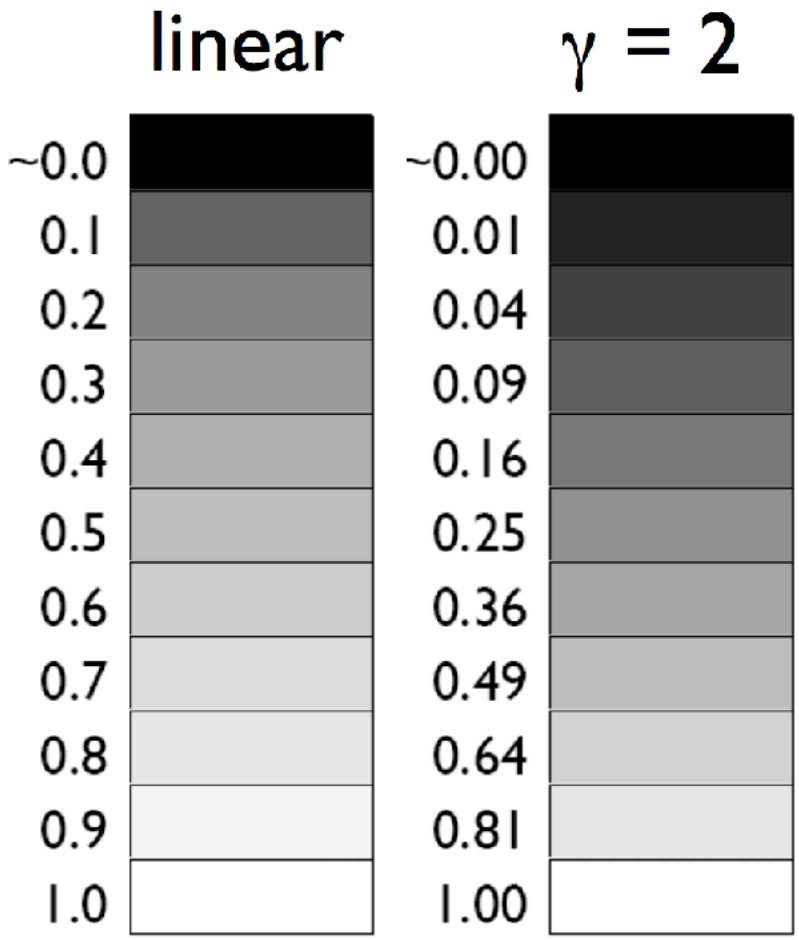
- We have computed intensities a that we want to display linearly
- In the case of an ideal monitor with zero black level and unit max

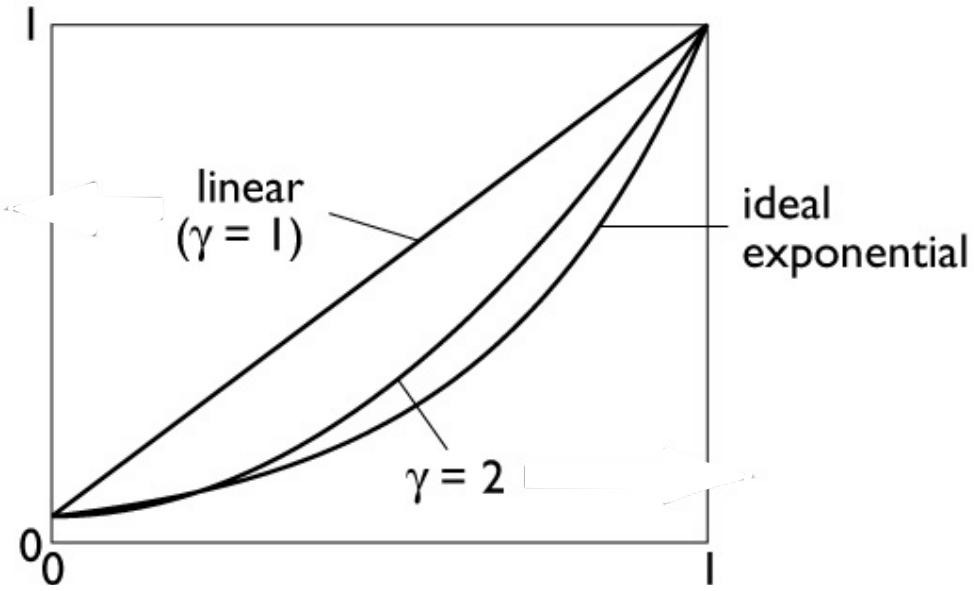
$$I(n) = (n/N)^{\gamma}$$

- This is the "gamma correction" recipe that has to be applied when computed values are converted to 8 bits for output
  - failing to do this (implicitly assuming gamma = 1) results in dark, oversaturated images
  - Typical value for gamma: 2.2

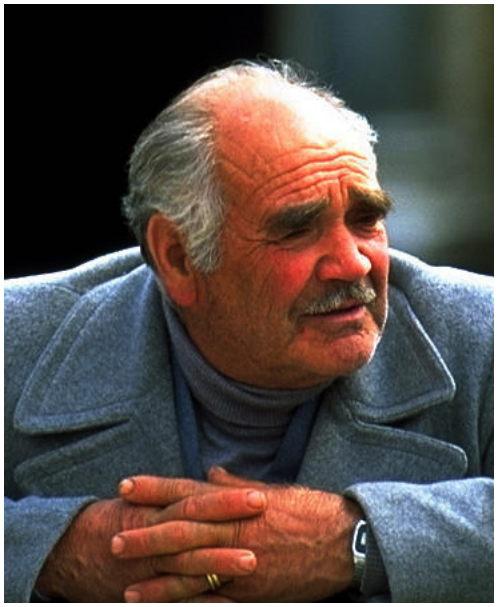
$$n(I) = NI^{\frac{1}{\gamma}}$$

Gamma quantization

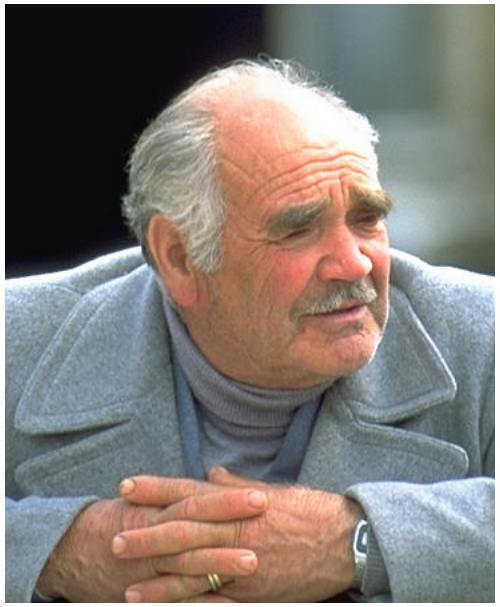




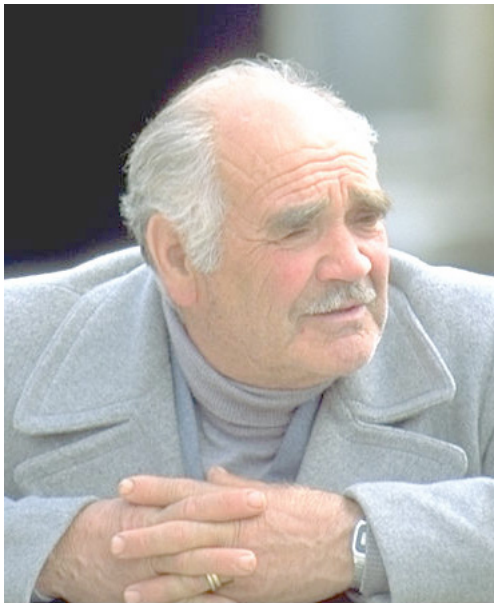
Gamma correction



γ lower than display



OK



γ higher than display

[Philip Greenspun]

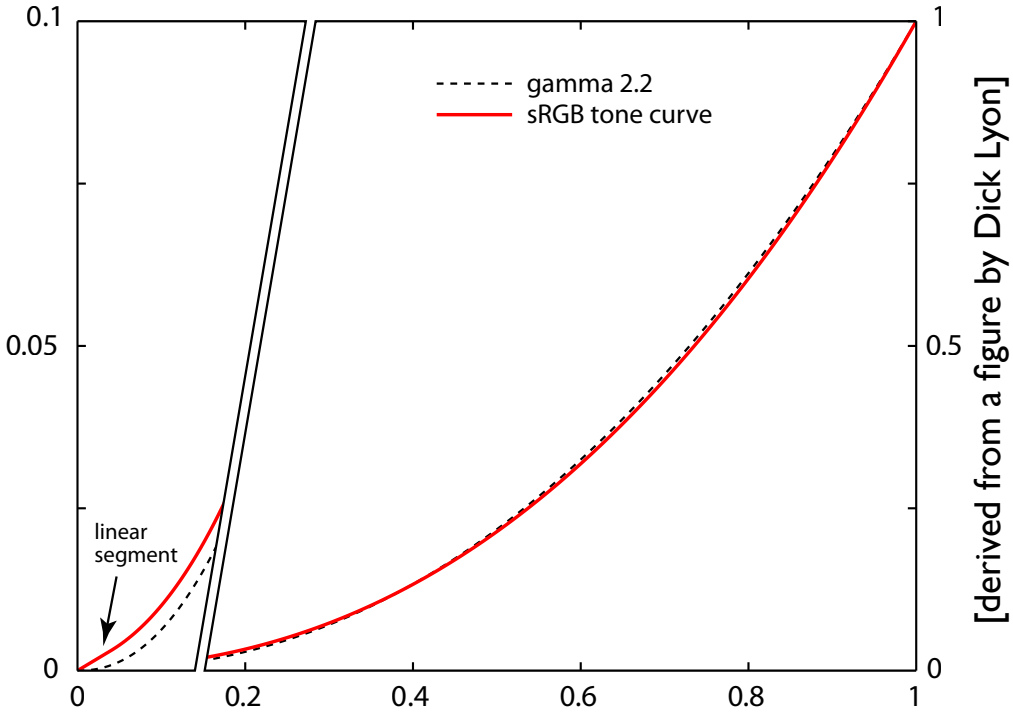
sRGB quantization curve

- The predominant standard for "casual color" in computer displays
  - backward compatible, monitors calibrated to sRGB by default
  - works well under imperfect conditions
  - approx. gamma 2.2

$$I(C) = \begin{cases} \frac{C}{12.92}, & C \leq 0.04045 \\ \left(\frac{C+a}{1+a}\right)^{2.4}, & C > 0.04045 \end{cases}$$

$$C = n/N$$

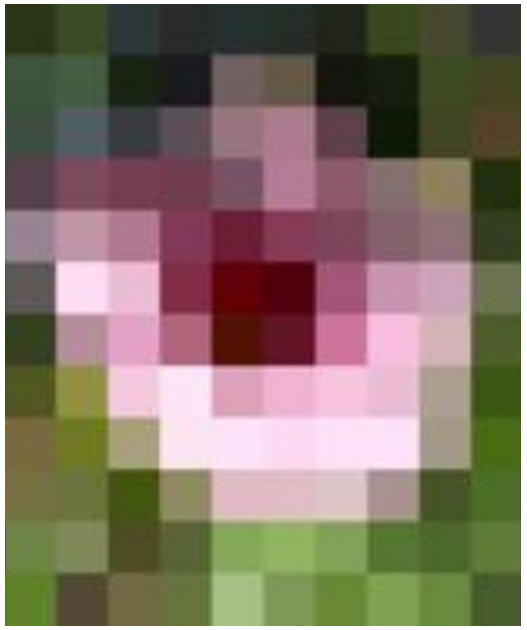
$$a = 0.055$$



What is a pixel?

- Color "around" a point, not the pixel center
- Provide better approximation of the true values

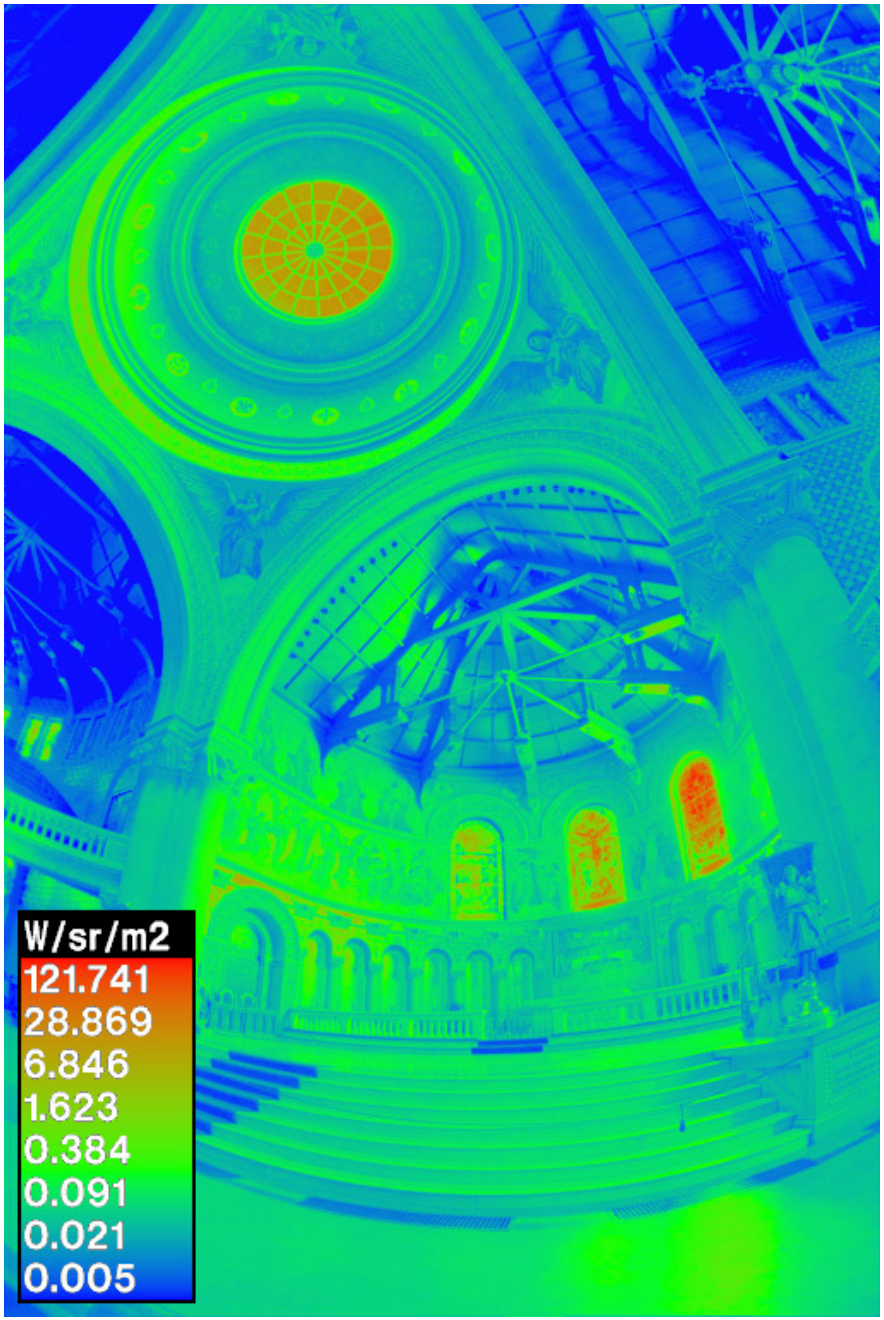
Image



Area Average

Pixel Center

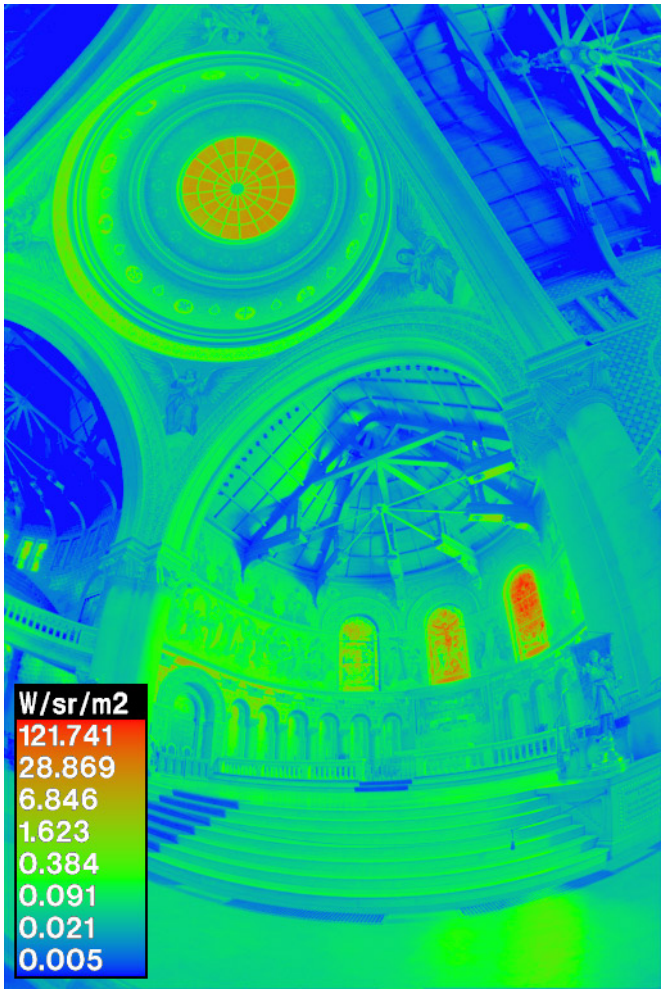
HDR Images



[Paul Debevec]

HDR Images

- store illumination values directly
- values are linear and not clamped
  - no transfer function
  - requires floating point



[Paul Debevec]

Capturing HDR Images

- capture multiple exposure, each of which is clamped
- aligned them so that pixels corresponds
- blend the "middle" portion of the range, to avoid clamped regions

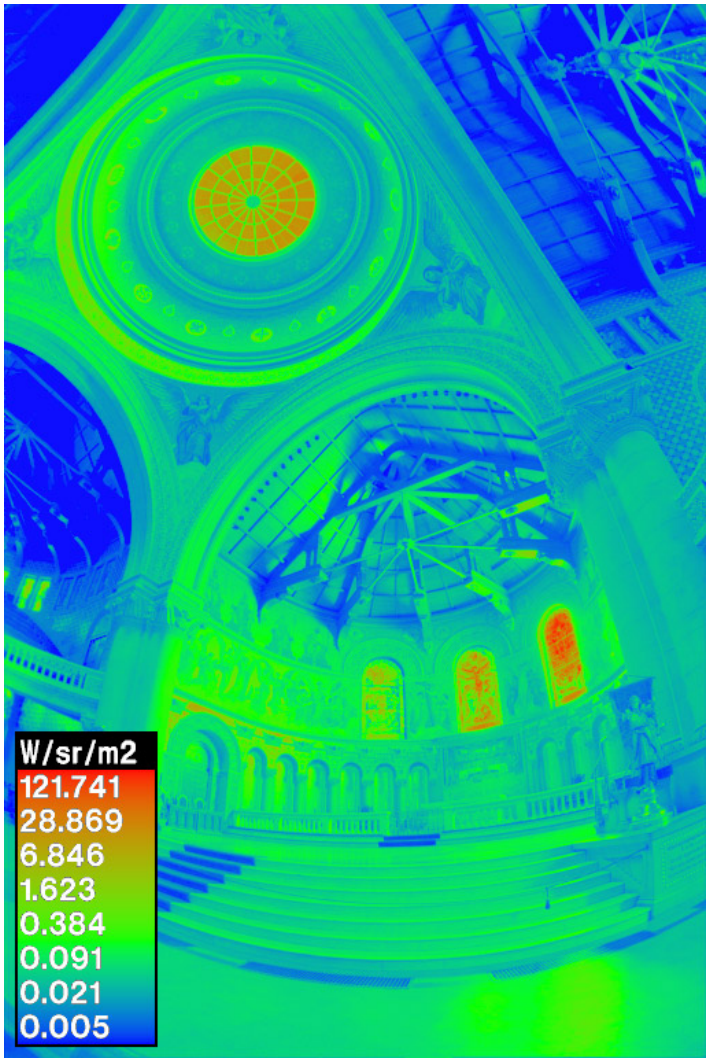
Viewing HDR images

- HDR images store all illumination values
- But displays can only reproduce between a min and max value
- Tone mapping: Reduce HDR range to display range
- Simple method: scale HDR values, apply gamma, and clamp
  - scale is often expressed in power of twos called *exposure*

$$I_{LDR} = \min(1, (sI_{HDR})^{\gamma}) \quad s = 2^{exposure}$$

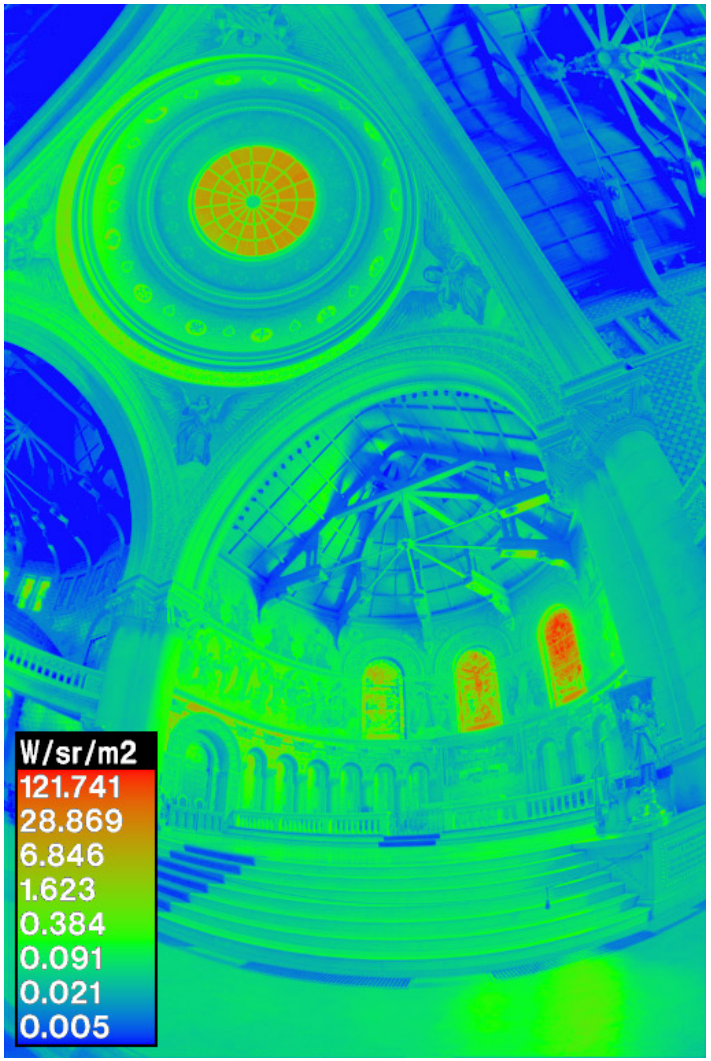
- For more artistic control: use different non-linear curve
  - control over mid-tone contrast and min/max values
- For more "correctness": simulate visual system

Linear scale



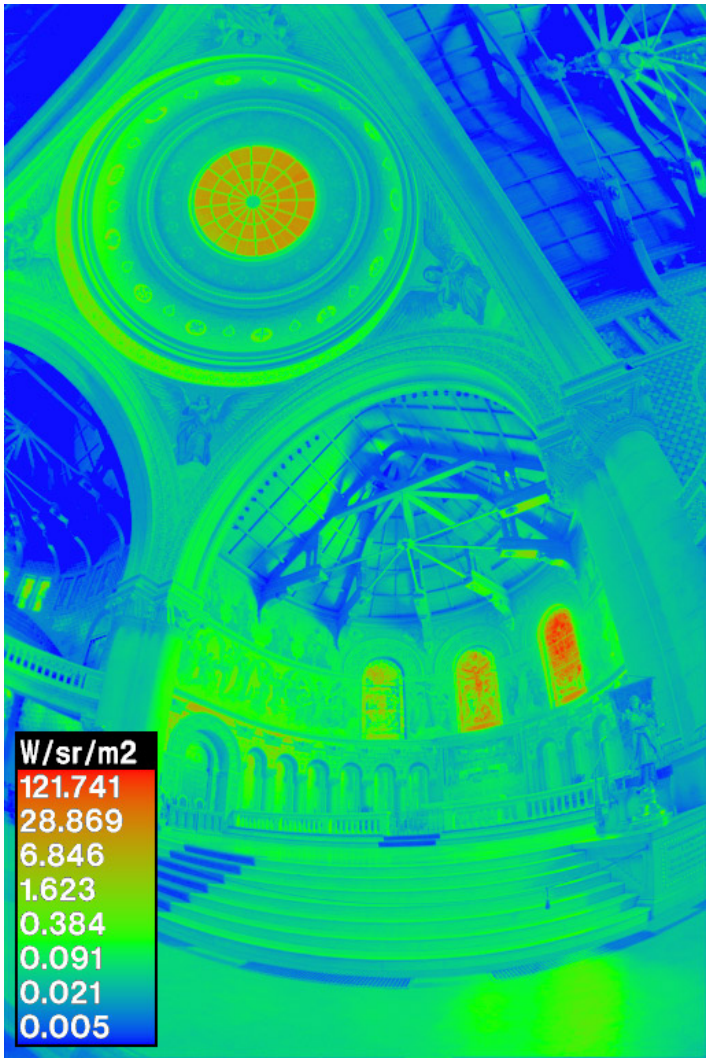
[Paul Debevec]

Select range via exposure



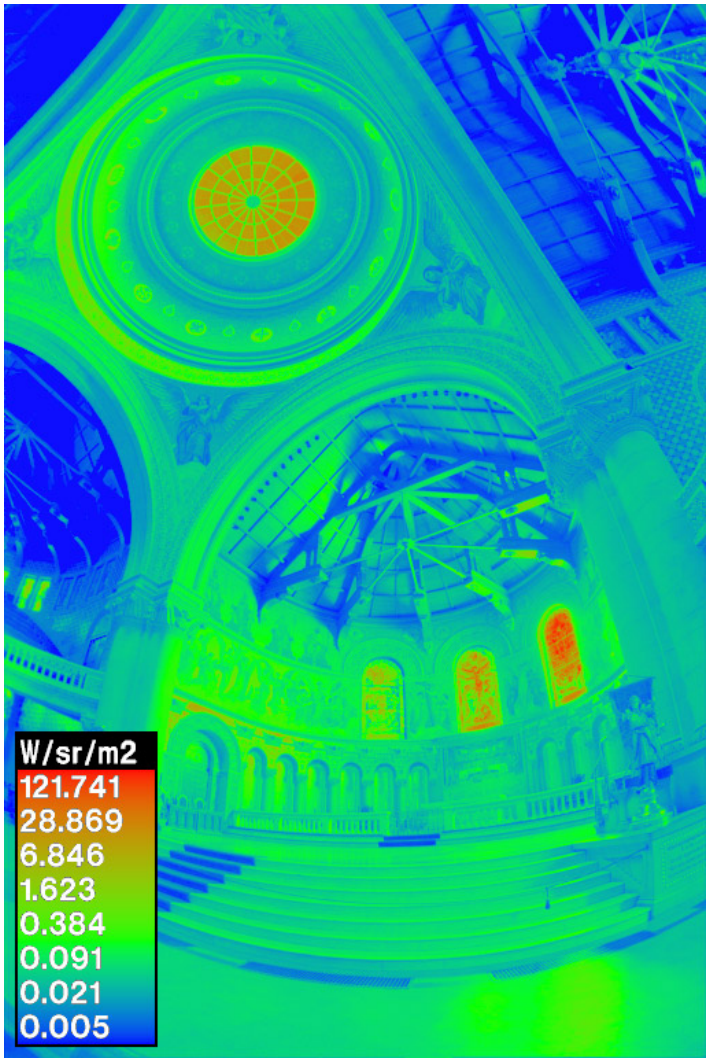
[Paul Debevec]

Non-linear correction (gamma)



[Paul Debevec]

Simulate Visual System

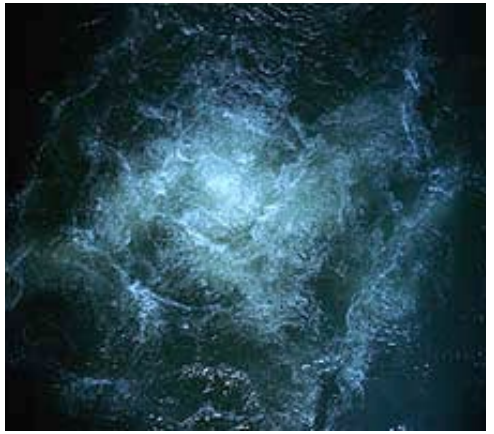
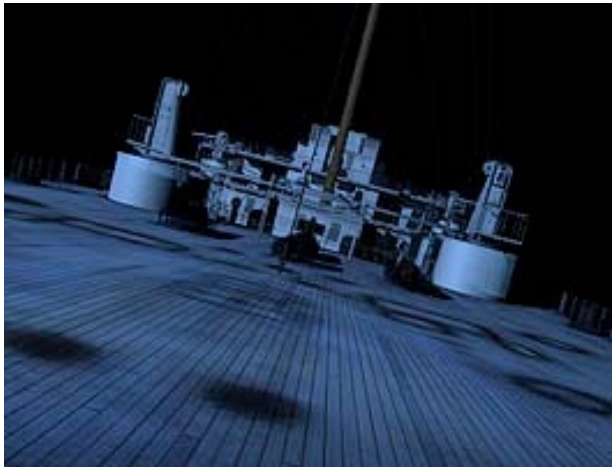


[Paul Debevec]

Compositing

Compositing



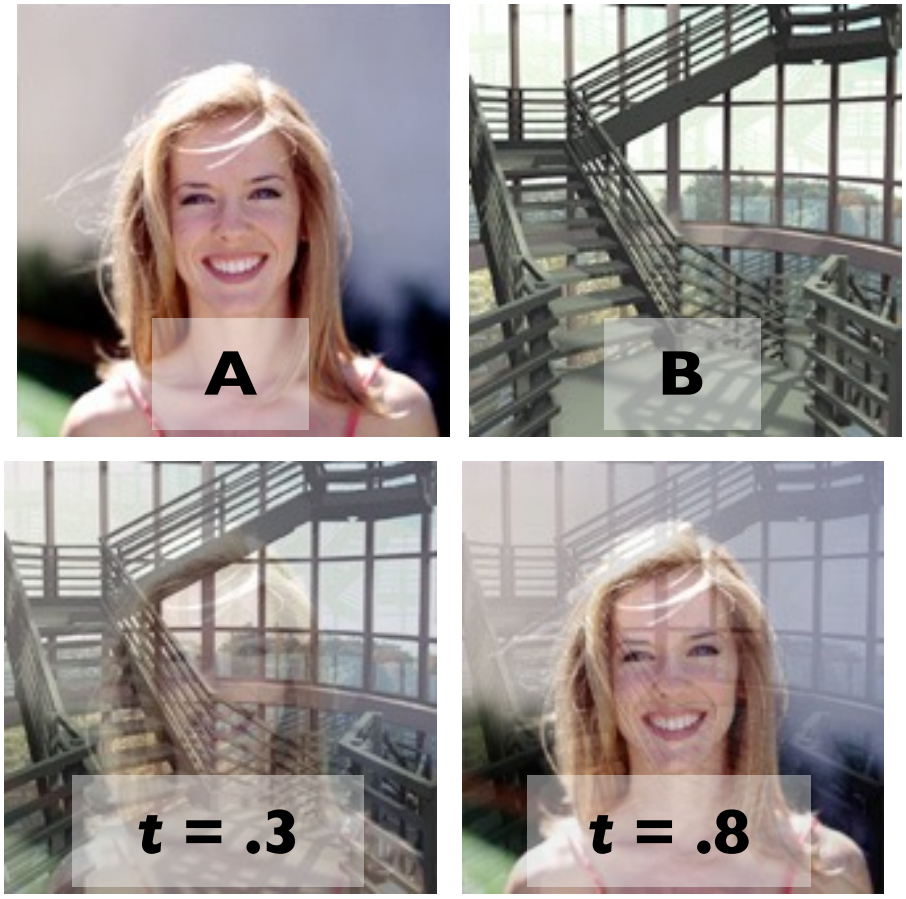

[Titanic ; DigitalDomain; vfxhq.com]

Combining images

- Trivial example: video crossfade
  - smooth transition from one to another by *linear interpolation*
  - note that weights sum to 1.0
  - no brightening or darkening
  - no out-of-range values
- Written in vector notation as

$$\mathbf{c}_C = t\mathbf{c}_A + (1-t)\mathbf{c}_B \qquad \begin{bmatrix} r_C \\ g_C \\ b_C \end{bmatrix} = \begin{bmatrix} tr_A + (1-t)r_B \\ tr_A + (1-t)g_B \\ tr_A + (1-t)b_B \end{bmatrix}$$

Combining images



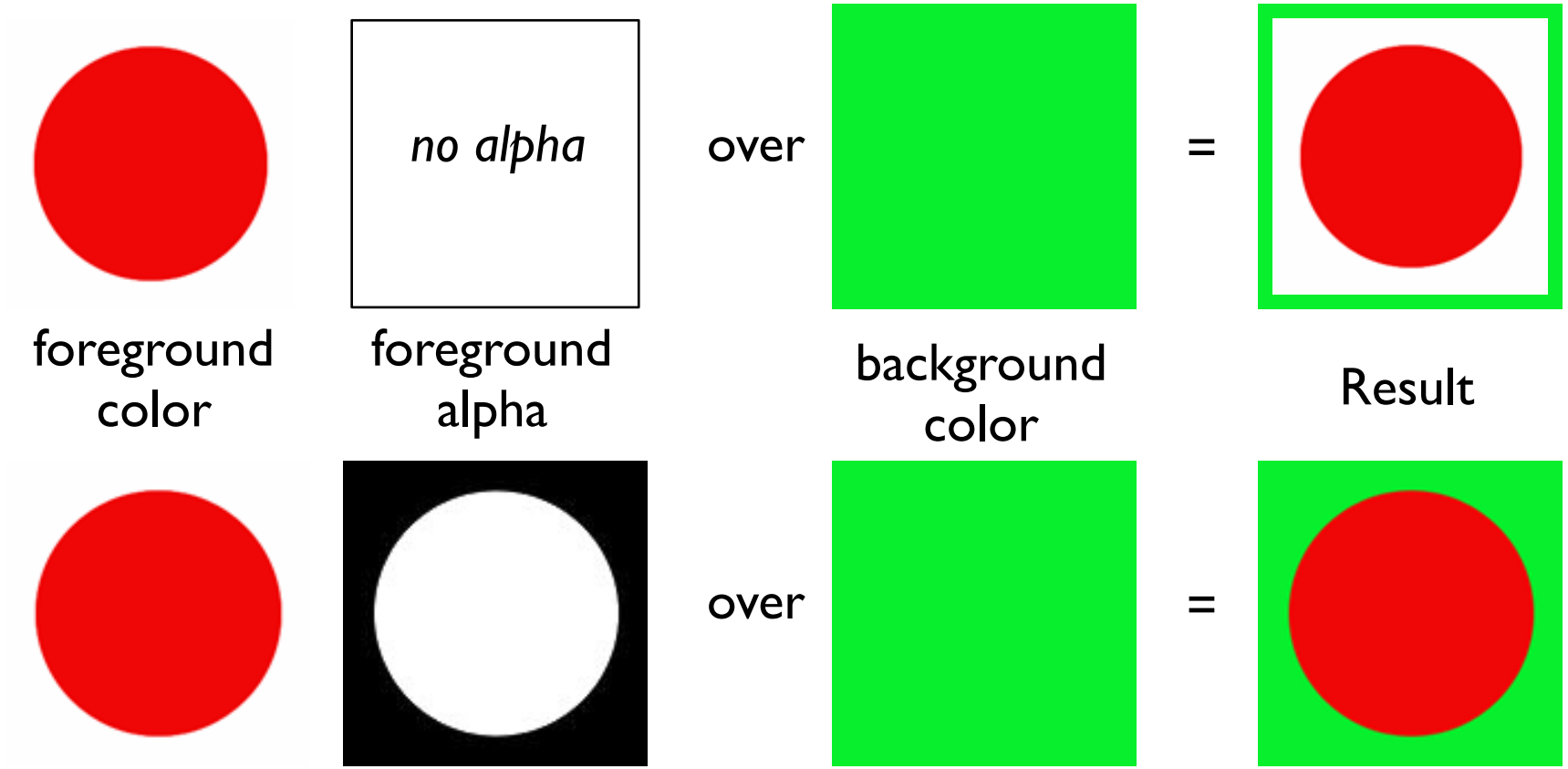
[Chuang et al. / Corel]

Foreground and background

- In many cases just adding is not enough
- Example: compositing in film production
  - shoot foreground and background separately
  - also include CG elements
  - this kind of thing has been done in analog for decades
  - how should we do it digitally?

Compositing Images

- encode transparency for each pixel: *alpha* channel



Binary image mask

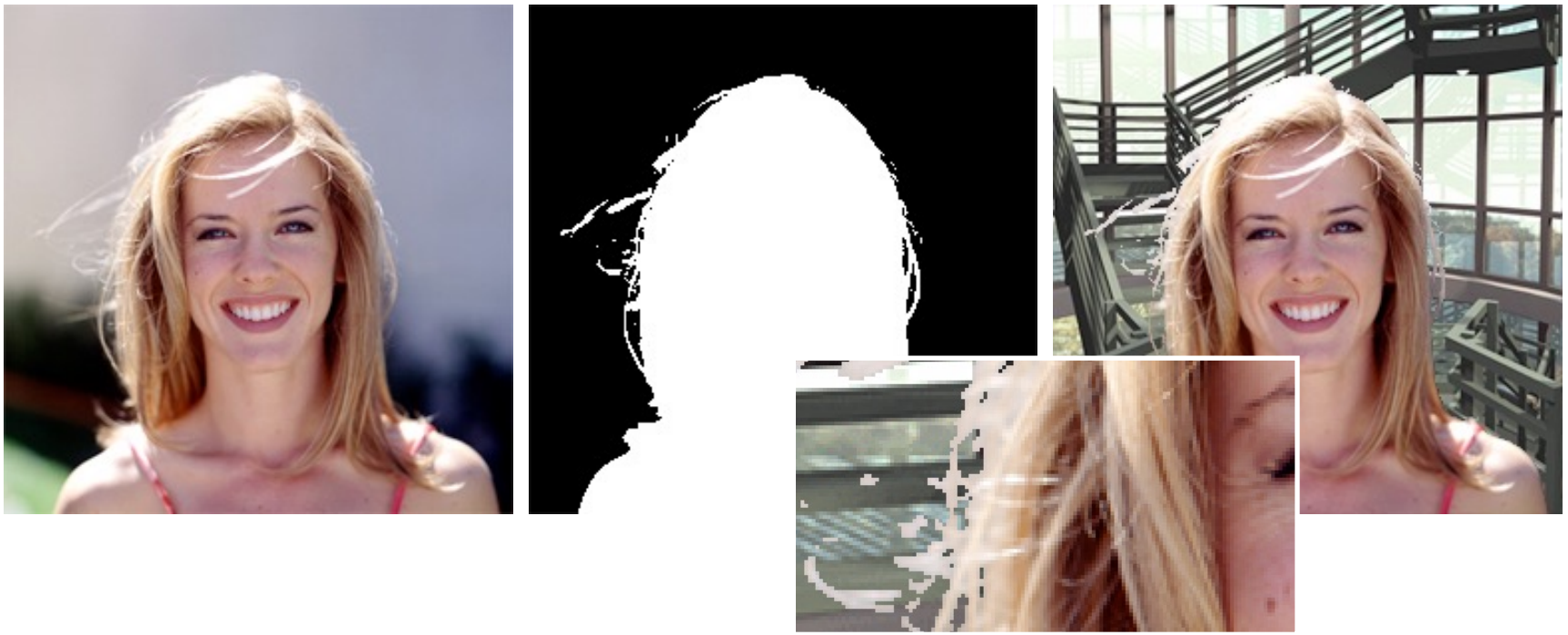
- First idea: store one bit per pixel
  - answers question "is this pixel part of the foreground?"
- Switch between images based on the bit

$$E = A \textbf{ over } B$$

$$\mathbf{c}_E = \begin{cases} \mathbf{c}_A & \alpha > 0 \\ \mathbf{c}_B & \alpha = 0 \end{cases}$$

Binary image mask

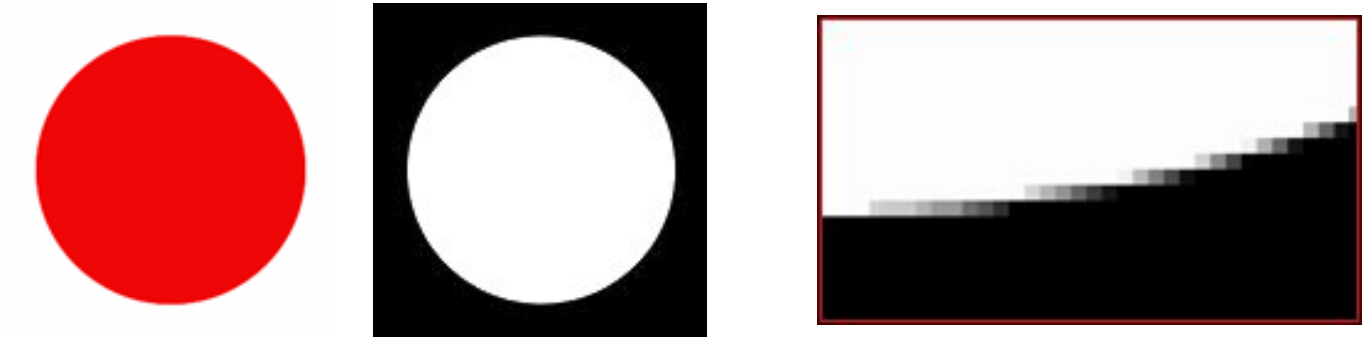
- Causes "jaggies"



[Chuang et al. / Corel]

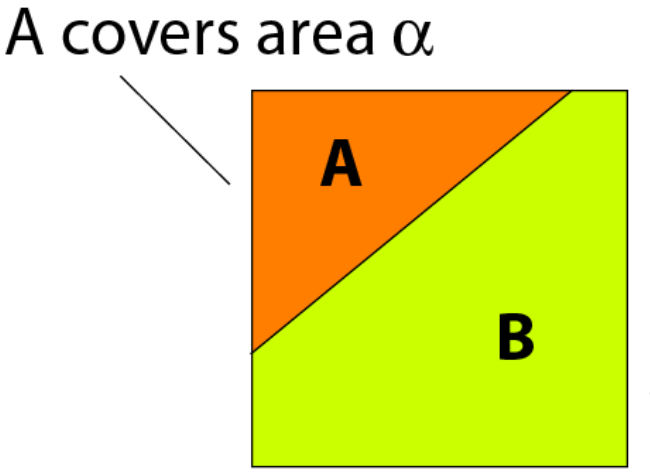
Partial pixel coverage

- pixels near boundary are not strictly foreground or background
  - interpolate boundary pixels between the fg. and bg. colors
  - store fractional alpha



Alpha compositing

- Formalized in 1984 by Porter & Duff
- Linearly interpolate based on fractional alpha
- Efficient: 8 more bits (total 32), 2 multiplies + 1 add per pixel
- Assume A and B cover the whole pixel

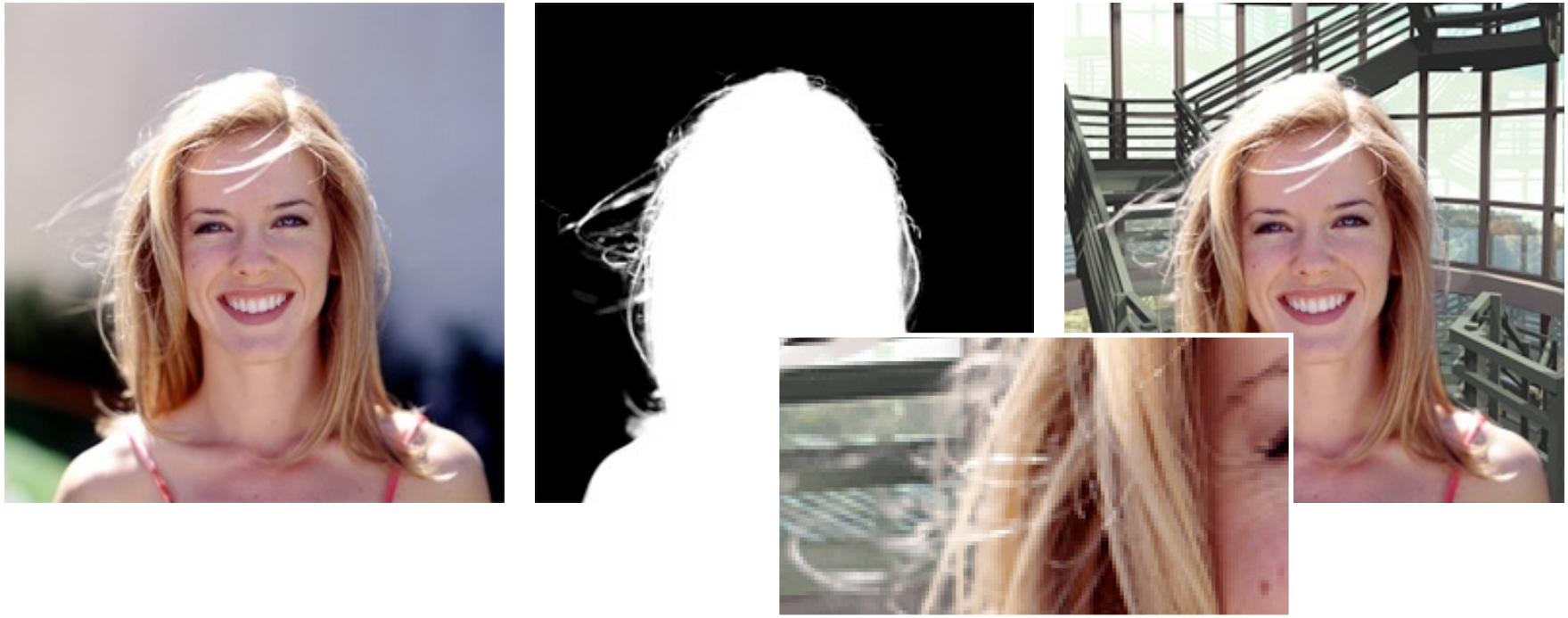


$$E = A \textbf{ over } B$$

$$\mathbf{c}_E = \alpha_A \mathbf{c}_A + (1 - \alpha_A)\mathbf{c}_B$$

Alpha compositing

- Smooth transition around edges



[Chuang et al. / Corel]

Compositing composites

- in real applications we have *n* layers
  - *Titanic* example
- compositing foregrounds to create new foregrounds
  - what to do with α?
- desirable property: associativity
  - can composite top-down and bottom-up

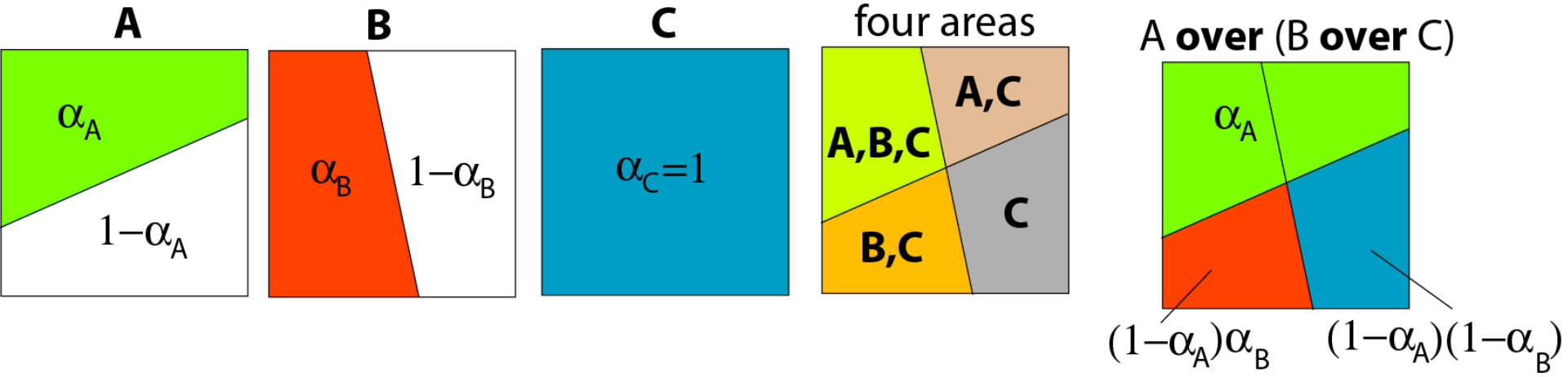
$$A \textbf{ over } (B \textbf{ over } C) = (A \textbf{ over } B) \textbf{ over } C$$

Compositing composites

- Compute compositing taking into account that areas are covered by both A and B and that C covers the whole pixel

$$D = A \textbf{ over } (B \textbf{ over } C)$$

$$\begin{aligned}\mathbf{c}_D &= \alpha_A \mathbf{c}_A + (1-\alpha_A)[\alpha_B \mathbf{c}_B + (1-\alpha_B)\mathbf{c}_C] = \\ &= \alpha_A \mathbf{c}_A + (1-\alpha_A)\alpha_B \mathbf{c}_B + (1-\alpha_A)(1-\alpha_B)\mathbf{c}_C\end{aligned}$$



Compositing composites

- Compute coverage of (A **over** B)

$$\alpha_{(A \textbf{ over } B)} = 1 - (1 - \alpha_A)(1 - \alpha_B) = \\ = \alpha_A + (1 - \alpha_A)\alpha_B$$

- but combing colors becomes complex in D = (A **over** B) **over** C

$$\mathbf{c}_D = \alpha_A \mathbf{c}_A + (1 - \alpha_A)\alpha_B \mathbf{c}_B + (1 - \alpha_A)(1 - \alpha_B)\mathbf{c}_C = \\ = \alpha_{(A \textbf{ over } B)}(...) + (1 - \alpha_{(A \textbf{ over } B)})\mathbf{c}_C$$

Premultiplied Alpha

- Compositing equation again for $E = A \textbf{ over } B$

$$\mathbf{c}_E = \alpha_A \mathbf{c}_A + (1 - \alpha_A)\alpha_B \mathbf{c}_B$$

- Note c_A appears only in the product $\alpha_A c_A$
- Multiply it ahead of time: *premultiplied alpha*:
  - store pixel value (*r'*, *g'*, *b'*, α) where *c'* = α*c*

$$\mathbf{c}'_E = \mathbf{c}'_A + (1 - \alpha_A)\mathbf{c}'_B \text{ with } \mathbf{c}' = \alpha\mathbf{c}$$

- Generalizes to the case of C not covering the whole pixel with

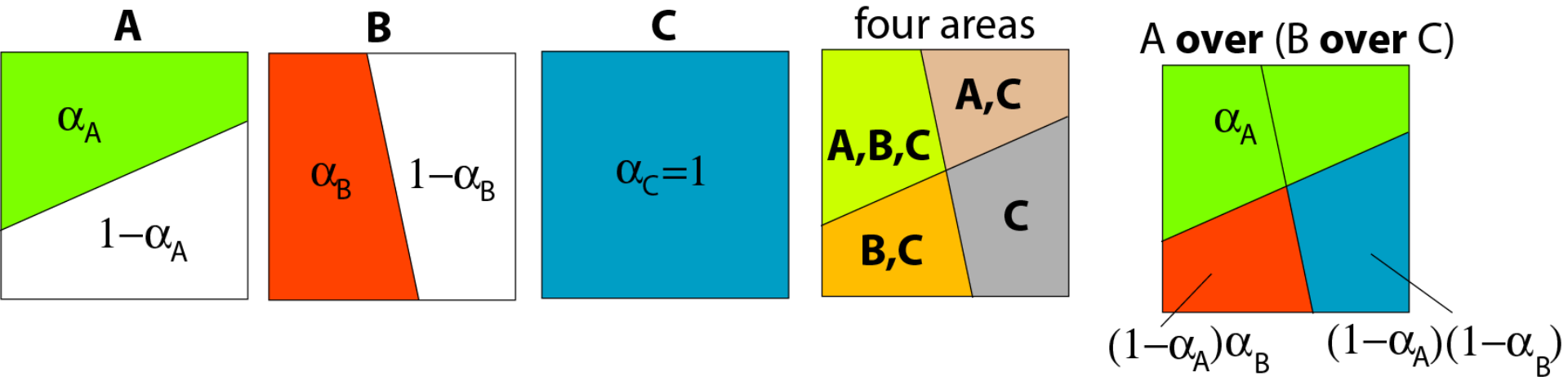
$$\alpha_E = \alpha_A + (1 - \alpha_A)\alpha_B$$

  - we will not prove this

Associativity

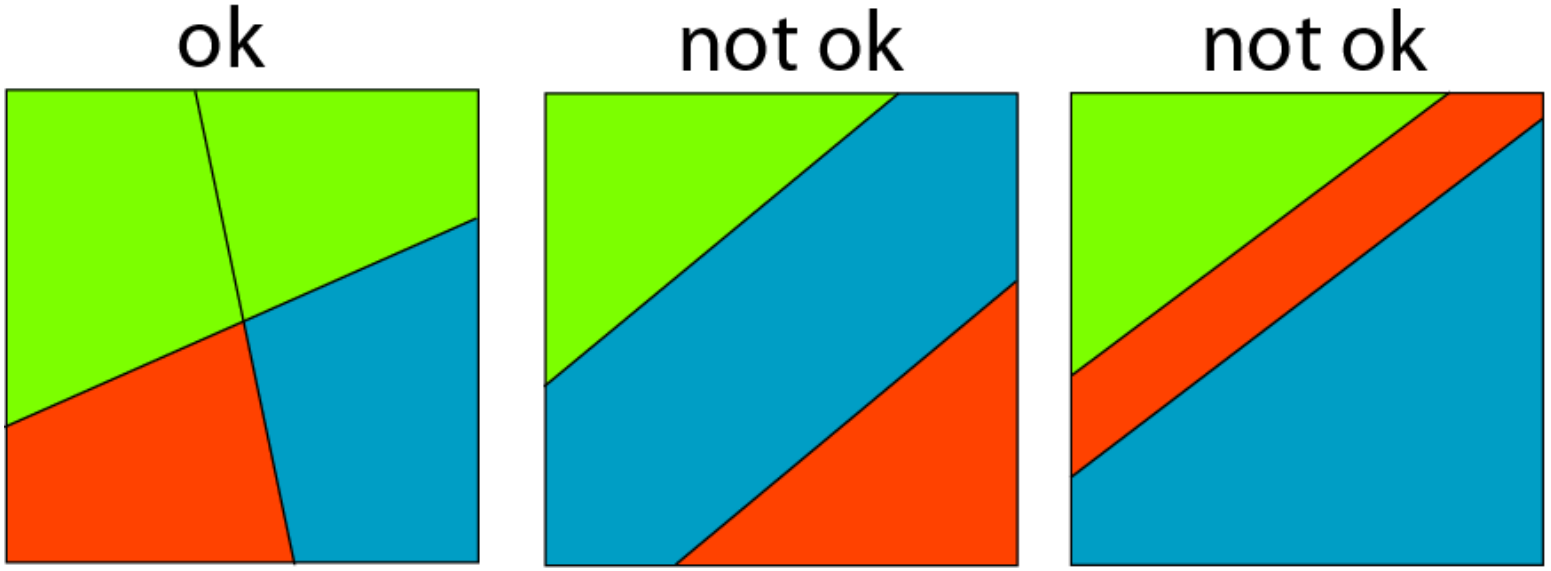
$$
\begin{aligned}
\mathbf{c}'_D &= \mathbf{c}'_A + (1-\alpha_A)\mathbf{c}'_{(B\ \textbf{over}\ C)} = \\
&= \mathbf{c}'_A + (1-\alpha_A)[\mathbf{c}'_B + (1-\alpha_B)\mathbf{c}'_C] = \\
&= \mathbf{c}'_A + (1-\alpha_A)\mathbf{c}'_B + (1-\alpha_A)(1-\alpha_B)\mathbf{c}'_C = \\
&= [\mathbf{c}'_A + (1-\alpha_A)\mathbf{c}'_B] + (1-\alpha_A)(1-\alpha_B)\mathbf{c}'_C = \\
&= \mathbf{c}'_{(A\ \textbf{over}\ B)} + (1-\alpha_{(A\ \textbf{over}\ B)})\mathbf{c}'_C
\end{aligned}
$$

$$
\Rightarrow\ (A\ \textbf{over}\ B)\ \textbf{over}\ C = A\ \textbf{over}\ (B\ \textbf{over}\ C)
$$



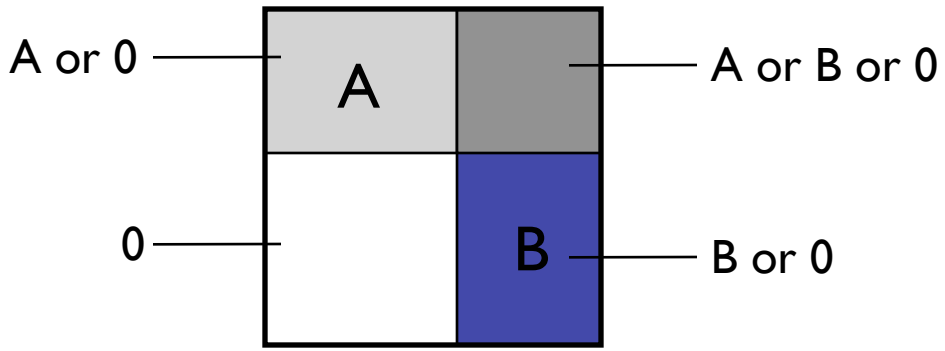
Independent coverage

- Why is it reasonable to blend α like a color?
- Simplifying assumption: covered areas are independent
  - that is, uncorrelated in the statistical sense
- Hold in most, but not all cases



Compositing Algebra

$$\alpha_E = A \ \mathbf{op} \ B$$
$$c'_E = F_A c'_A + F_B c'_B$$



1 x 2 x 3 x 2 = 12 reasonable choices

operation	quadruple	diagram	F_A	F_B
clear	(0,0,0,0)		0	0
A	(0,A,0,A)		1	0
B	(0,0,B,B)		0	1
A **over** *B*	(0,A,B,A)		1	$1-\alpha_A$
B **over** *A*	(0,A,B,B)		$1-\alpha_B$	1
A **in** *B*	(0,0,0,A)		α_B	0
B **in** *A*	(0,0,0,B)		0	α_A
A **out** *B*	(0,A,0,0)		$1-\alpha_B$	0
B **out** *A*	(0,0,B,0)		0	$1-\alpha_A$
A **atop** *B*	(0,0,B,A)		α_B	$1-\alpha_A$
B **atop** *A*	(0,A,0,B)		$1-\alpha_B$	α_A
A **xor** *B*	(0,A,B,0)		$1-\alpha_B$	$1-\alpha_A$

[Porter & Duff 84]

Compositing Graphs

- Large compositing graphs are common
- Associativity allows to cache partial results
- In turn this means that we can have large graph without paying cost and at low engineering cost
  - so compositing is used everywhere
  - Photoshop
  - Web
  - PDF
  - UIs