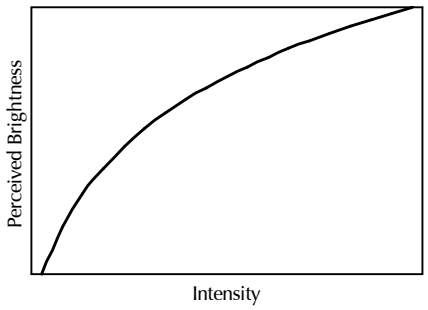
- Last week, we talked about some characteristics of the human visual system that are relevant to graphics:
  - Spatiotemporal resolution
  - Tristimulus color
- Today, we wrap up our discussion of perception, and see how the ideas are applied in computer graphics systems:
  - Sensitivity to differences in intensity
  - Gamma, quantization, dithering
  - Representation of intensity
  - Representation of color

Human Perception of Intensity

- Basic question: how many bits per pixel (bpp) do we need for computer graphics?
- Answer: it depends on what the bits represent
- Reason: human perception of intensity is non-linear
- Terminology: radiance (intensity) vs. luminance vs. lightness (perceived brightness)

Human Perception of Intensity (cont.)



- The exact form of this curve might vary depending on ambient conditions and adaptation of the eye, but the general shape remains the same

Bits per Pixel

- Number of bits needed depends on whether they represent radiance or lightness
- Experiments reveal that "Just Noticeable Difference" (JND) for humans is 1%
- Reduced in areas with rapidly varying texture
- Further reduced for most computer monitors because of ambient light

Bits per Pixel – Rules of Thumb

- Synthetic scenes: 5-7 bits in perceptually uniform space, 6-10 bits or more in linear radiance space
- Scanned pictures: 5 bits often OK – noise
- Lots of textures or lots of edges: 3 bits often enough
- What if you don't have enough bits?
  - Mach banding
  - Aliasing (jaggies)

Bits per Pixel – Synthetic Scenes

8 bits / pixel / color

7 bits / pixel / color

Bits per Pixel – Synthetic Scenes (cont.)

6 bits / pixel / color

5 bits / pixel / color

Bits per Pixel – Scanned Pictures

8 bits / pixel / color

6 bits / pixel / color

Bits per Pixel – Scanned Pictures (cont.)

5 bits / pixel / color

4 bits / pixel / color

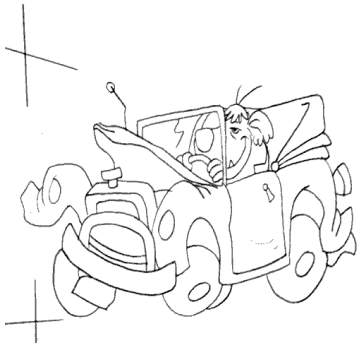
Bits per Pixel – Line Drawings

8 bits / pixel / color

4 bits / pixel / color

Bits per Pixel – Line Drawings (cont.)

3 bits / pixel / color

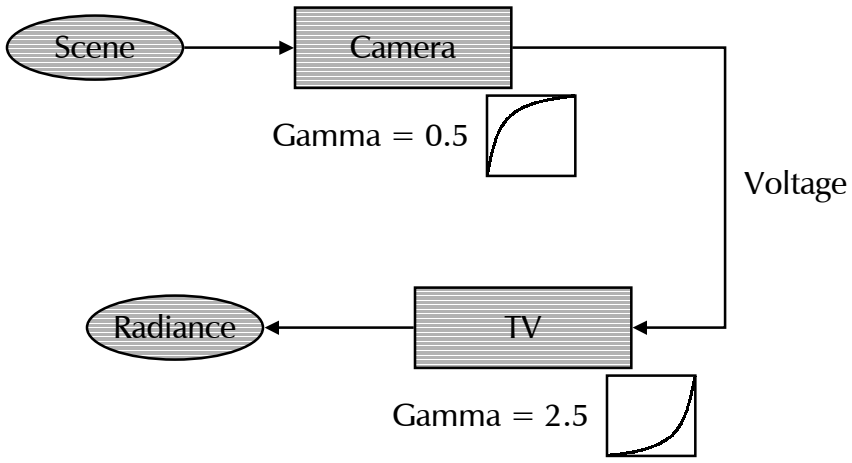


2 bits / pixel / color

Working in Perceptually Uniform Spaces

- We need fewer bits if the bits represent perceived brightness, not radiance
- A good approximation to visual system response is $B = L^{\gamma}$, with γ around 1/3 to 1/2
- So, we should (?) be working in a space where the value we store is $(\text{Intensity})^{\gamma}$
- Coincidentally, this would also compensate for response of CRT

Digression: Gamma in CRTs and Broadcast Television



Digression: Gamma in CRTs and Broadcast Television (cont.)

- Physics of (original) TV cameras is such that they produce signal with $\gamma = 0.5$
- CRT has $\gamma = 2.3\text{-}2.8$ – almost reciprocal
- $(L^{\gamma_1})^{\gamma_2} = I^{(\gamma_1\gamma_2)} \Rightarrow$ gammas multiply
- Small residual "system" gamma (1.2-1.4) – compensates for ambient lighting
- Computers also use CRTs with $\gamma = 2.5$
  - Output radiance not proportional to voltage sent out by graphics system

Back to Nonlinear Intensity Spaces

- So, we now have two good reasons for working in a space with $\gamma \neq 1$, but this is almost never done these days – the algorithms become too hard
- Examples: global illumination, alpha blending

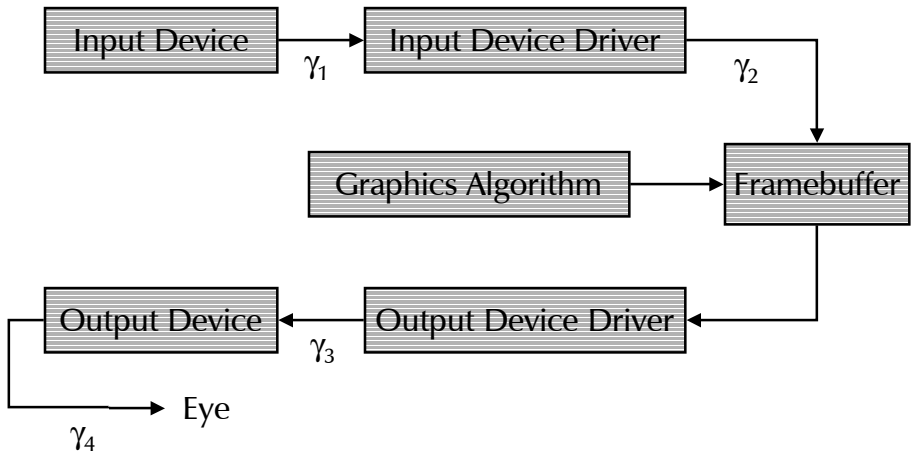
Alpha Blending with $\gamma \neq 1$

- Let $A=L_1^{\gamma}$, $B=L_2^{\gamma}$
- What value corresponds to halfway between A and B?
- Answer: $(\frac{1}{2} A^{1/\gamma} + \frac{1}{2} B^{1/\gamma})^{\gamma}$
- Result: have to do three exponentiations per pixel if $\gamma \neq 1$

Nonlinear Intensity Spaces (cont.)

- Result: in modern graphics systems, framebuffer holds radiance values ($\gamma = 1$)
- Separate "gamma correction" step right before sending values to CRT
- Beware: most pictures found on web or coming from "consumer sources" are already gamma corrected, because most PCs don't do gamma correction

Gamma in the Graphics Pipeline



Gamma in the Graphics Pipeline (cont.)

- Just as in the TV pipeline, different gamma at different stages
- The input device, its controller (hardware and software), the output device, and its controller might all apply some gamma
- Graphics algorithms right in the middle!
- Visual system has its own gamma, but you can usually ignore it – get the radiance right and it looks right

Color Images in the Framebuffer

- We know how many bits for intensity, but what about color?
- One approach: colormaps
  - Framebuffer stores index into lookup table
  - Used to be prominent, now much less used
  - Lots of research on how to pick the colors in the colormap – "optimal quantization"
  - Large body of theory, slowly becoming obsolete

Color Images in the Framebuffer (cont.)

- Alternative to colormaps: store *n* bits for each color
- Most popular layouts: 888, 565, also 555, 444, 332 (bits for R, G, B, respectively)
- Modern graphics systems often also store "alpha" as another component – layouts like 8888

Dithering

- What if you don't have enough memory to use the color depth you want? Alternatively, what if your output device has fewer intensity levels than you want?
- Trade off spatial resolution for bit depth
  - Back to psychophysics – produce least perceptible error
  - Moves low frequency error to high frequencies

Random Dither

- Scanned image needed fewer bpp because of noise
- Noise + quantization = random dither
- Need to match amount of noise to amount of quantization
- Will any noise do?
  - No – high-frequency noise is best

Random Dither – Low- vs. High-Frequency Noise

Images Copyright © Pat Hanrahan

"Pink" noise – low frequency

"Blue" noise – high frequency

Ordered Dither

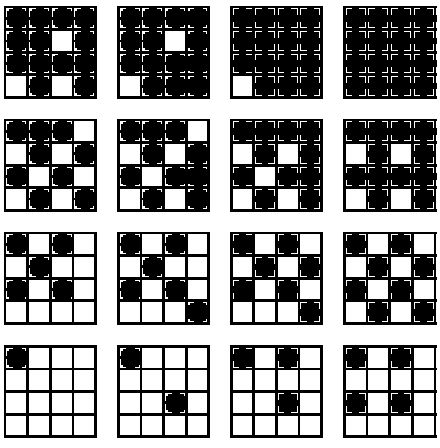
- The noise added during dithering need not be random
  - Deterministic patterns of noise are easier to implement in hardware
- Can construct a pattern that is deterministic and minimizes low-frequency content

Bayer Dither Matrices

$$\mathbf{D}_2 = \begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix}$$

$$\mathbf{D}_n = \begin{bmatrix} 4\mathbf{D}_{n/2} + 0 & 4\mathbf{D}_{n/2} + 2 \\ 4\mathbf{D}_{n/2} + 3 & 4\mathbf{D}_{n/2} + 1 \end{bmatrix}$$

$$\mathbf{D}_4 = \begin{bmatrix} 0 & 8 & 2 & 10 \\ 12 & 4 & 14 & 6 \\ 3 & 11 & 1 & 9 \\ 15 & 7 & 13 & 5 \end{bmatrix}$$



Random Dither vs. Bayer Ordered Dither

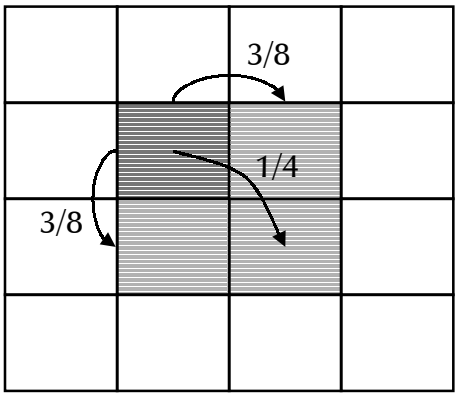
Random dither

Bayer dither

Error Diffusion in Dithering

- Bayer introduces artifacts that look like vertical and horizontal lines
  - Our visual system is very sensitive to these
- Another approach to dithering: error diffusion
- Idea: quantize a pixel, then distribute quantization error to neighbors

Floyd-Steinberg Error Diffusion Algorithm



- Image processed left-right, top-bottom
- Often found in inkjet printers

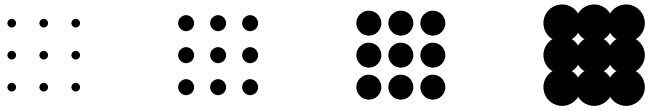
Bayer Ordered Dither vs. Floyd-Steinberg Error-Diffusion Dither

Bayer dither

Floyd-Steinberg dither

Clustered-Dot Dithering
(a.k.a. Digital Halftoning)

- Emulates classic halftoning
- Darker regions ⇒ larger clumps of pixels
- Most useful when device resolution is greater than resolution of desired image
- Often found on laser printers



Modern Computer Graphics Systems – PC Graphics Cards

- Framebuffer stores pixel values (usually 16 or 24 bits), sometimes alpha, sometimes Z
- Typical resolutions: 800x600, 1024x768, 1280x1024
- Rasterization (converting primitives to pixels) done in hardware, most other things in software

Modern Computer Graphics Systems – High-End Machines

- Framebuffer stores pixel values (usually 32 to 48 bits), alpha, Z, texture cache, other buffers (stereo, accumulation, etc.)
- Typical resolutions: 1280x1024, 1600x1200, 1920x1080 (HDTV), etc.
- Hardware does rasterization, transformation, blending, signal processing, gamma correction
- "Low-end" PC graphics cards catching up