Computer Graphics

Prof. Feng Liu

Fall 2016

http://www.cs.pdx.edu/~fliu/courses/cs447/

10/03/2016

Announcements

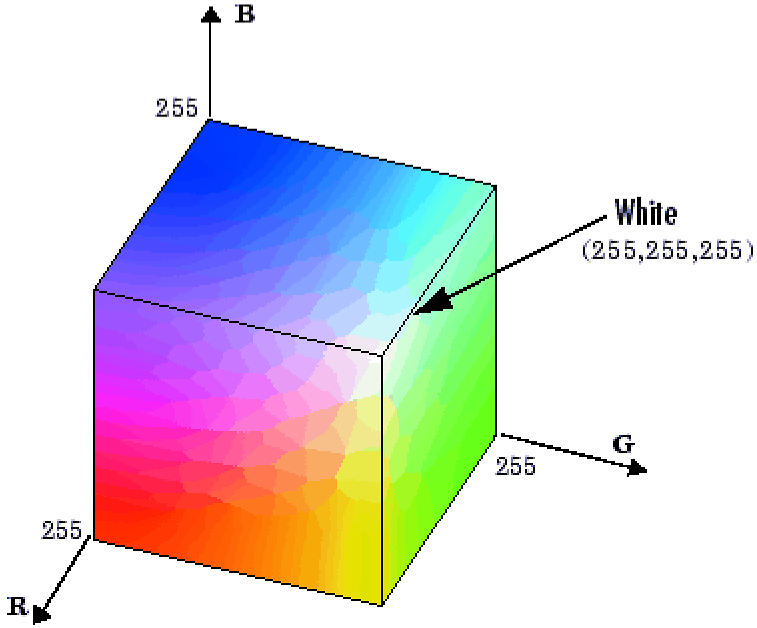
- Free Textbook: Linear Algebra
  - By **Jim Hefferon**
  - http://joshua.smcvt.edu/linalg.html/
- Homework 1 due in class on Oct. 05
- Project 1 is available on course website
  - due 5pm October 28

Last Time

- Color
  - The principle of trichromacy says that any spectrum can be matched using three primaries (but sometimes you have to subtract a primary)
  - A color system consist of primaries and color matching functions that are used to determine how much of each primary is needed to match a spectrum
  - RGB, CIE XYZ, HSV are some examples of color systems
  - Linear color spaces make it easy to convert between colors - matrix multiply
  - Color calibration is an important step to achieving accurate color
- Today
  - Perceptually linear (uniform) color spaces make distances between colors meaningful

RGB Color Space

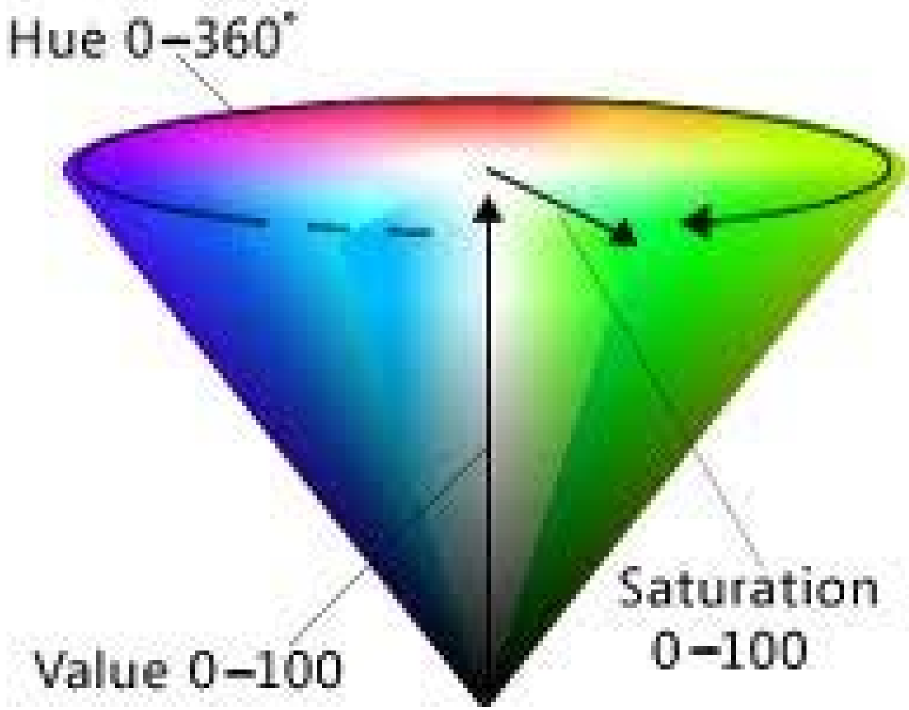
- Demo



HSV Color Space (Alvy Ray Smith, 1978)

- Hue: the color family: red, yellow, blue…
- Saturation: The purity of a color: white is totally unsaturated
- Value: The intensity of a color: white is intense, black isn't
- Space looks like a cone
  - Parts of the cone can be mapped to RGB space
- Not a linear space, so no linear transform to take RGB to HSV
  - But there is an algorithmic transform

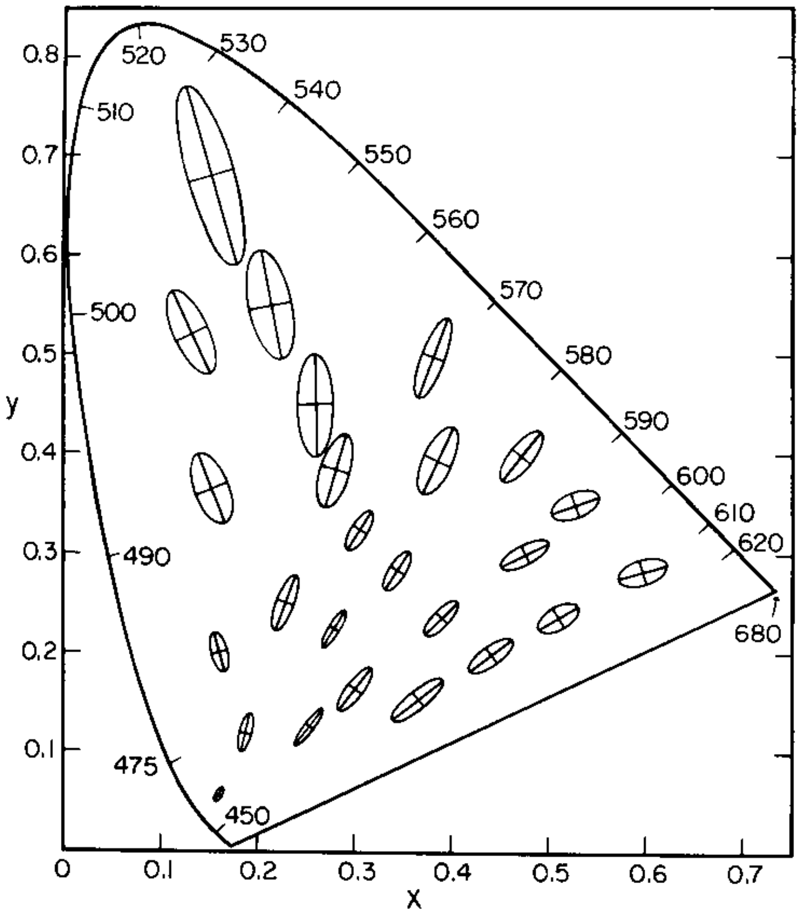
HSV Color Space



Linear Space vs. Perceptually Uniform

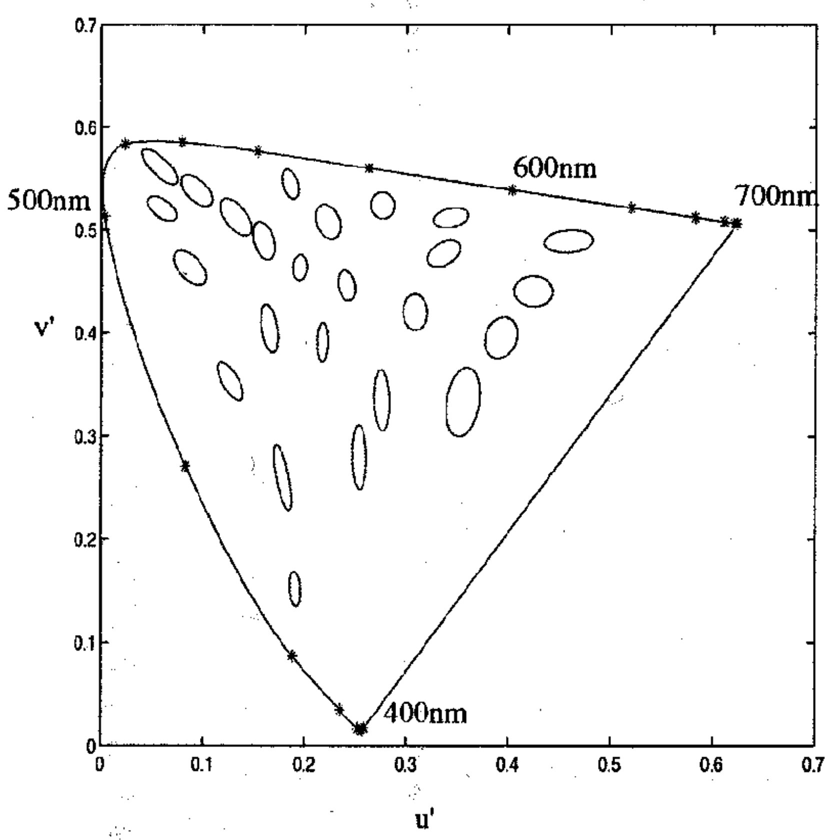
- Linear Space: RGB, CIE XYZ
  - The principle of trichromacy means that the colors displayable are all the linear combination of primaries
    - HSV is not a linear space
  - Matrix multiplication
  - Easy to convert between colors
  - Not perceptually linear
- Perceptually Uniform space
  - Computational consuming
  - Make color distance meaningful
  - CIE u'v': a good approximation

MacAdam Ellipses



- Refer to the region which contains all colors which are indistinguishable
- Scaled by a factor of 10 and shown on CIE xy color space
- If you are shown two colors, one at the center of the ellipse and the other inside it, you cannot tell them apart
- Only a few ellipses are shown, but one can be defined for every point

CIE u'v' Space



- CIE u'v' is a non-linear color space where color differences are more uniform
- Note that now ellipses look more **like** circles
- The third coordinate is the original Z from XYZ

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = \frac{1}{X + 15Y + 3Z} \begin{bmatrix} 4X \\ 9Y \end{bmatrix}$$

Today

- Ink
- Image file formats
- Color quantization
- Programming tutorial 2
  - How to use FLTK within Visual Studio

Ink

- Ink is thought of as *adsorbing* particles
  - **You see the color of the paper, filtered by the ink**
  - Combining inks adsorbs more color, so subtractive color
    - White paper - red - blue = green
  - The color and texture of the paper affects the color of the image

Subtractive mixing

- Common inks: Cyan=White−Red; Magenta=White−Green; Yellow=White−Blue
  - cyan, magenta, yellow, are how the inks look when printed
- For good inks, matching is linear:
  - C+M+Y=White-White=Black
  - C+M=White-Red-Green=Blue
  - How to make a red mark?

Subtractive mixing

- Common inks: Cyan=White−Red; Magenta=White−Green; Yellow=White−Blue
  - cyan, magenta, yellow, are how the inks look when printed
- For good inks, matching is linear:
  - C+M+Y=White-White=Black
  - C+M=White-Red-Green=Blue
  - How to make a red mark?
- Usually require CMY and Black, because colored inks are more expensive, and registration is hard
  - Registration is the problem of making drops of ink line up

Calibrating a Printer

- If the inks (think of them as primaries) are linear, there exists a 3x3 matrix and an offset to take RGB to CMY
  - For example, if an RGB of (1,0,0) goes to CMY of (0,1,1); (0,1,0)→(1,0,1); and (0,0,1)→(1,1,0), then the matrix is

$$\begin{bmatrix} c \\ m \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} r \\ g \\ b \end{bmatrix}$$

- To calibrate your printer, you find out exactly what the numbers in the matrix should be
  - Print with cyan ink only and match the color with RGB, repeat with magenta and yellow, use the results to determine the matrix

Image File Formats

- How big is the image?
  - All files in some way store width and height
- How is the image data formatted?
  - Is it a black and white image, a grayscale image, a color image, an *indexed color* image?
  - How many bits per pixel?
- What other information?
  - Color tables, compression codebooks, creator information…
- All image formats are a trade-off between ease of use, size of file, and quality of reproduction

The Simplest File

- Assumes that the color depth is known and agreed on
- Store width, height, and data for every pixel in sequence
- This is how you normally store an image in memory

$0_{r,g,b}$	$1_{r,g,b}$	$2_{r,g,b}$
$3_{r,g,b}$	$4_{r,g,b}$	$5_{r,g,b}$
$6_{r,g,b}$	$7_{r,g,b}$	$8_{r,g,b}$

```
class Image {
      unsigned int width;
      unsigned int height;
      unsigned char *data;
}
```

0_r	0_g	0_b	1_r	1_g	1_b	2_r	2_g	2_b	3_r	3_g

- Unsigned because width and height are positive, and unsigned char because it is the best type for raw 8 bit data
- Note that you require some implicit scheme for laying out a rectangular array into a linear one

Indexed Color

- 24 bits per pixel (8-red, 8-green, 8-blue) are expensive to transmit and store
- It must be possible to represent all those colors, but *not in the same image*
- Solution: Indexed color
  - Assume k bits per pixel (typically 8)
  - Define a *color table* containing 2^k colors (24 bits per color)
  - Store the *index* into the table for each pixel (so store k bits for each pixel, instead of 24 bits)
  - Once common in hardware, now an artifact (256 color displays)

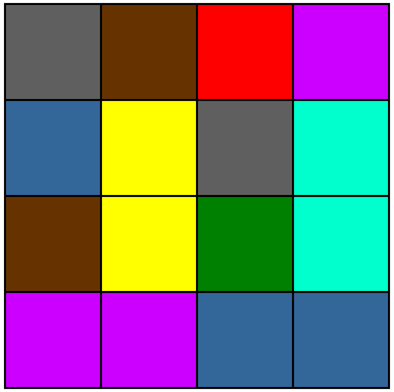
Indexed Color

Color Table

Pixel Data

4	3	0	2
1	7	4	5
3	7	6	5
2	2	1	1

Image



Only makes sense if you have lots of pixels and not many colors

Image Compression

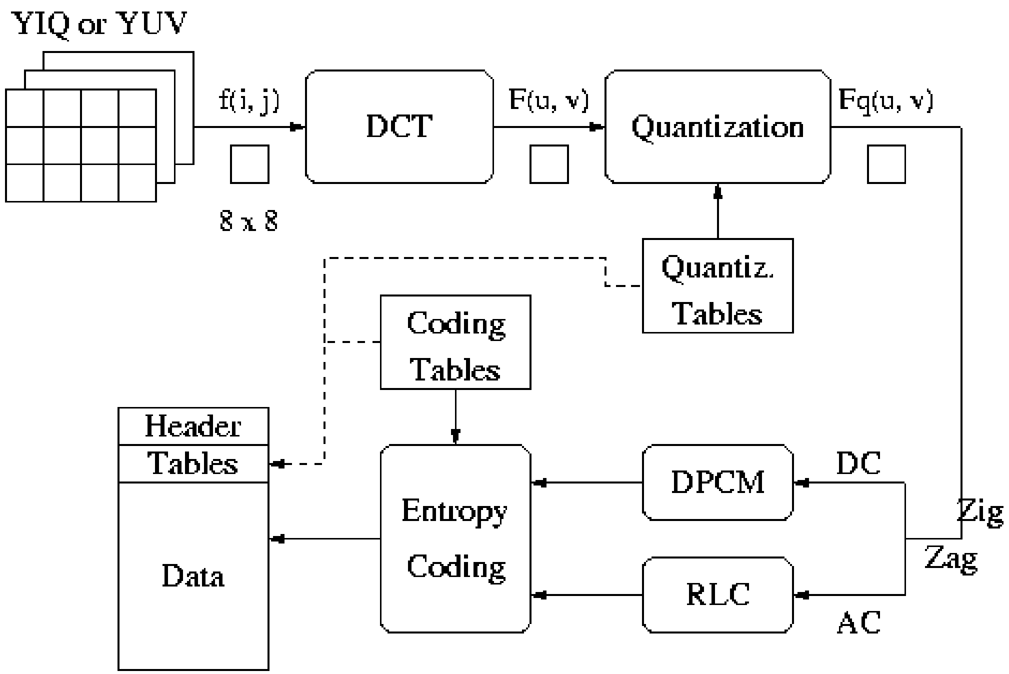
- Indexed color is one form of image compression
  - Special case of *vector quantization* – in color space, reducing the range of available colors
- Alternative 1: Store the image in a simple format and then compress with your favorite compressor
  - Doesn’t exploit image specific information
  - Doesn’t exploit perceptual shortcuts
- Two historically common compressed file formats: GIF and JPEG
  - GIF should now be replaced with PNG, because GIF is patented and the owner started enforcing the patent
    - Patent expired recently?

GIF

- Header - Color Table - Image Data - Extensions
- Header gives basic information such as size of image and size of color table
- Color table gives the colors found in the image
  - Biggest it can be is 256 colors, smallest is 2
- Image data is LZW compressed color indices
- To create a GIF:
  - Choose colors
  - Create an array of color indices
  - Compress it with LZW

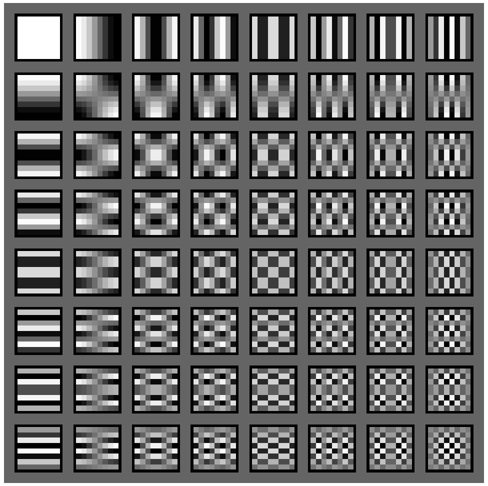
JPEG

- Multi-stage process intended to get very high compression with controllable quality degradation
- Start with YIQ color



Discrete Cosine Transform

- A transformation to convert from the *spatial* to *frequency* domain – done on 8x8 blocks
- Why? Humans have varying sensitivity to different frequencies, so it is safe to throw some of them away
- Basis functions:



Quantization

- Reduce the number of bits used to store each coefficient by dividing by a given value
  - If you have an 8 bit number (0-255) and divide it by 8, you get a number between 0-31 (5 bits = 8 bits – 3 bits)
  - Different coefficients are divided by different amounts
  - Perceptual issues come in here
- Achieves the greatest compression, but also quality loss
- “Quality” knob controls how much quantization is done

Entropy Coding

- Standard lossless compression on quantized coefficients
  - Delta encode the DC components
  - Run length encode the AC components
    - Lots of zeros, so store number of zeros then next value
  - Huffman code the encodings

Lossless JPEG With Prediction

- Predict what the value of the pixel will be based on neighbors
- Record error from prediction
  - Mostly error will be near zero
- Huffman encode the error stream
- Variation works really well for fax messages

Today

- Ink
- Image file formats
- Color quantization
- Programming tutorial 2
  - How to use FLTK within Visual Studio

Color Quantization

- The problem of reducing the number of colors in an image with minimal impact on appearance
  - Extreme case: 24 bit color to black and white
  - Less extreme: 24 bit color to 256 colors, or 256 grays
- Sub problems:
  - Decide which colors to use in the output (if there is a choice)
  - Decide which of those colors should be used for each input pixel

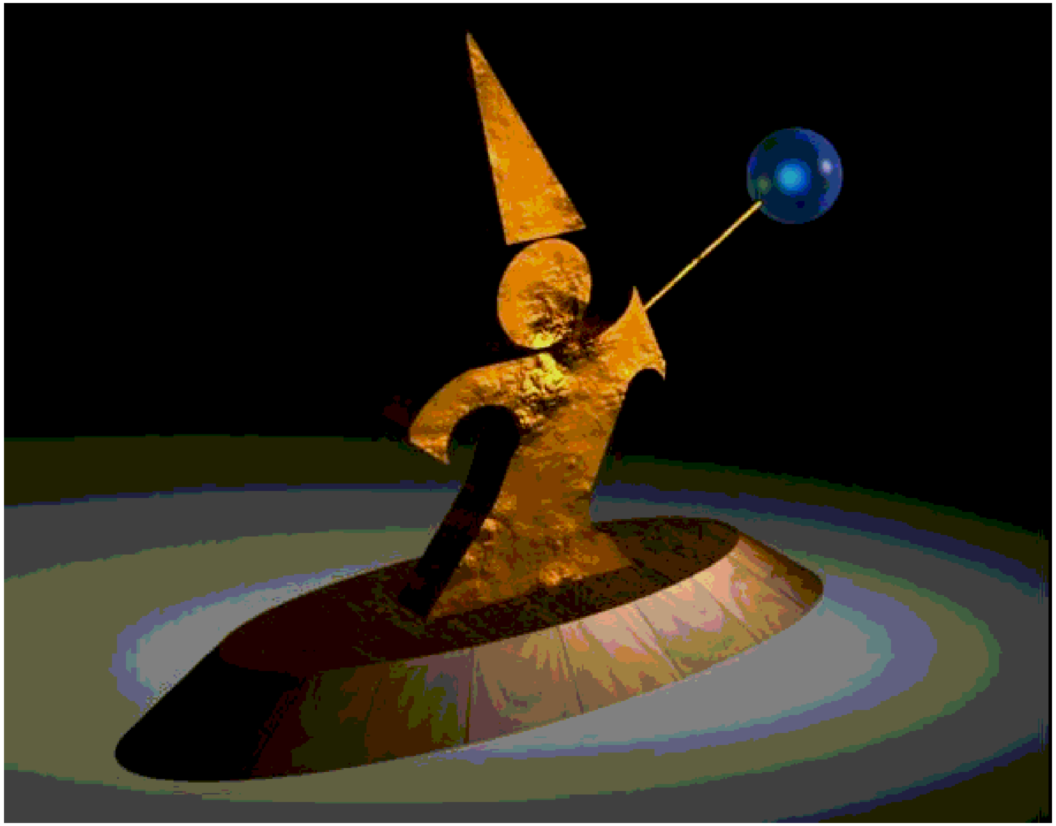
Example (24 bit color)

Uniform Quantization

- Break the color space into uniform cells
- Find the cell that each color is in, and map it to the center
- Equivalent to dividing each color by some number and taking the integer part
  - Say your original image is 24 bits color (8 red, 8 green, 8 blue)
  - Say you have 256 colors available, and you choose to use 8 reds, 8 greens and 4 blues ($8 \times 8 \times 4 = 256$)
  - Divide original red by 32, green by 32, and blue by 64
  - Some annoying details
- Generally does poorly because it fails to capture the distribution of colors
  - Some cells may be empty, and are wasted

Uniform Quantization

- 8 bits per pixel in this image
- Note that it does very poorly on smooth gradients
- Normally the hardest part to get right, because lots of similar colors appear very close together
- Does this scheme use information from the image?

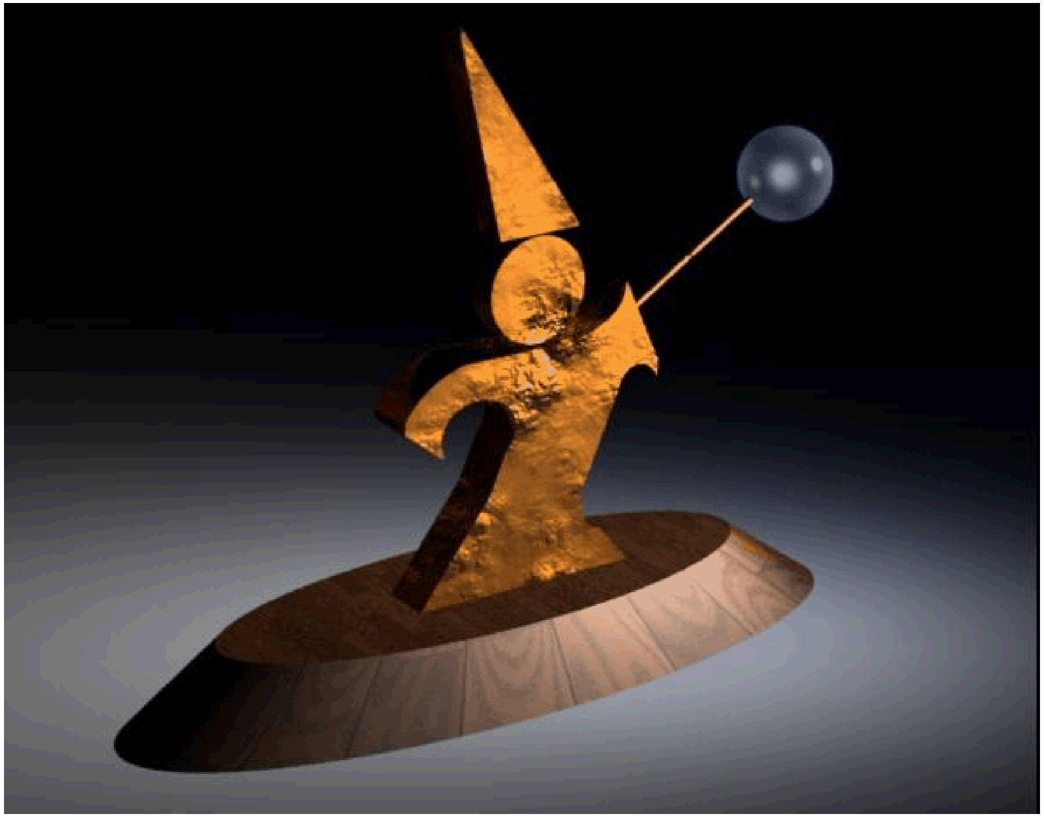


Populosity Algorithm

- Build a color histogram: count the number of times each color appears
- Choose the *n* most commonly occurring colors
  - Typically group colors into *small* cells first using uniform quantization
- Map other colors to the closest chosen color
- Problem?

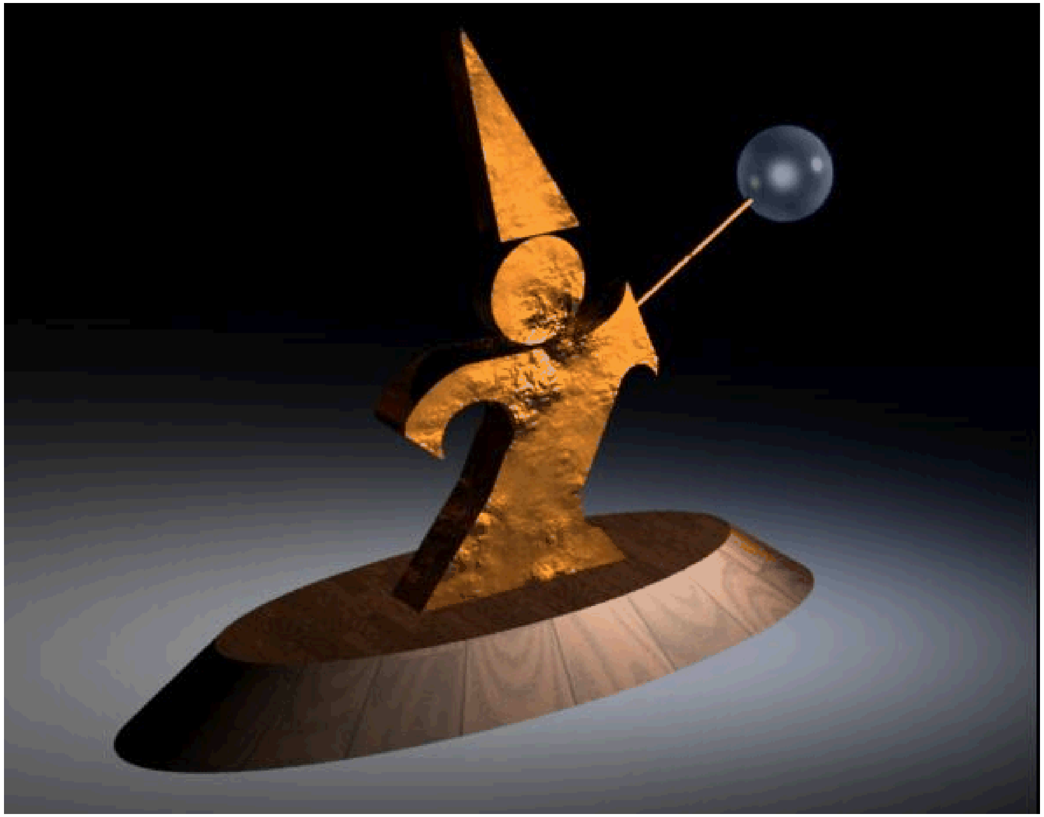
Populosity Algorithm

- 8 bit image, so the most popular 256 colors



Populosity Algorithm

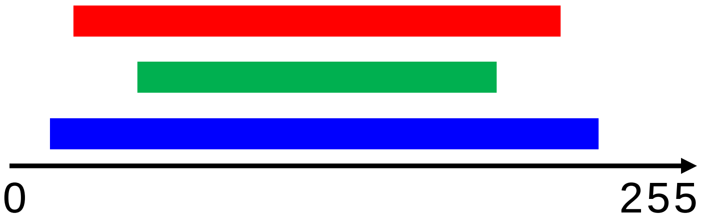
- 8 bit image, so the most popular 256 colors
- Note that blue wasn't very popular, so the crystal ball is now the same color as the floor
- Populosity ignores rare but important colors!



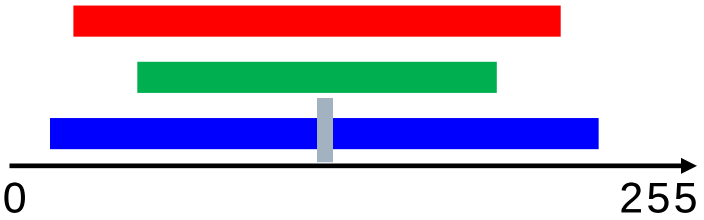
Median Cut (Clustering)

- View the problem as a *clustering* problem
  - Find groups of colors that are similar (a cluster)
  - Replace each input color with one representative of its cluster
- Many algorithms for clustering
- *Median Cut* is one: recursively
  - Find the "longest" dimension (r, g, b are dimensions)
  - Choose the median of the long dimension as a color to use
  - Split into two sub-clusters along the median plane, and recurse on both halves
- Works very well in practice

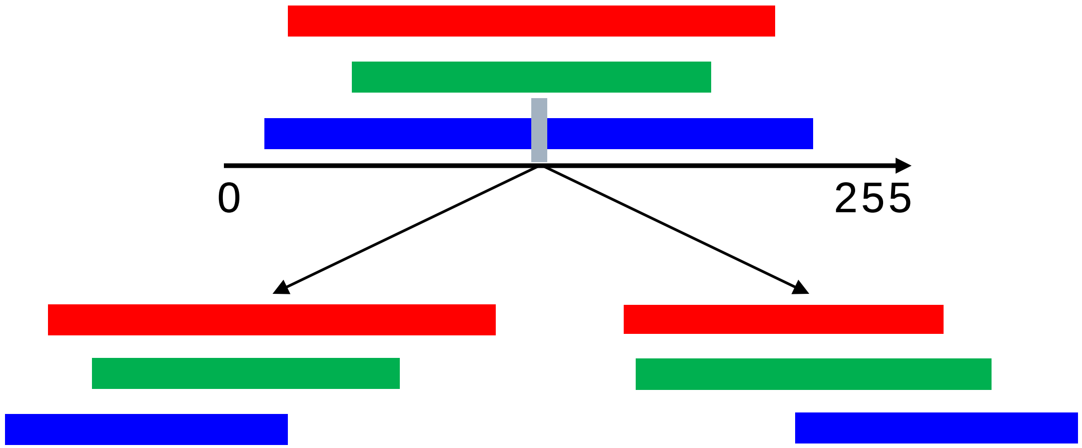
Median Cut (Clustering)



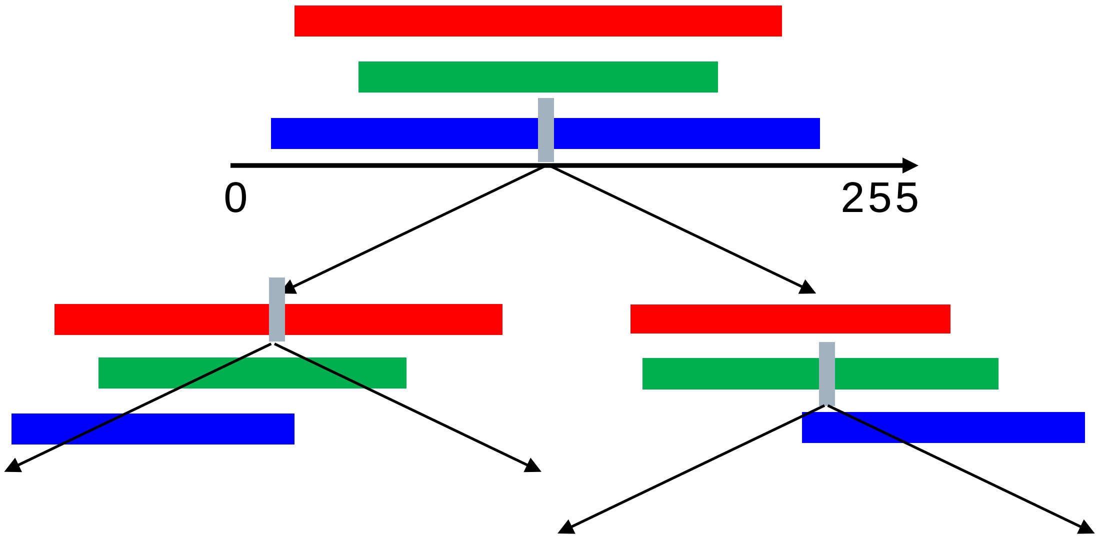
Median Cut (Clustering)



Median Cut (Clustering)

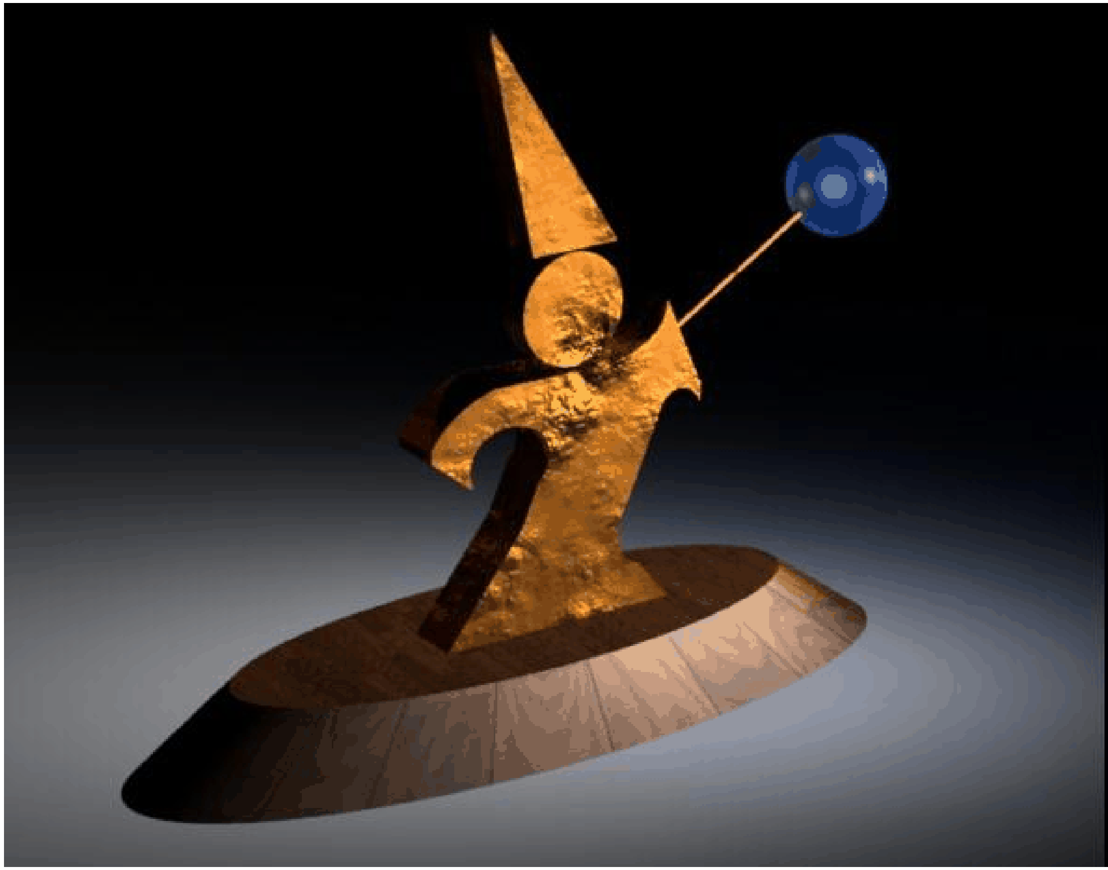


Median Cut (Clustering)



Median Cut

- 8 bit image, so 256 colors
- Now we get the blue
- Median cut works so well because it divides up the color space in the "most useful" way



Optimization Algorithms

- The quantization problem can be phrased as optimization
  - Find the set of colors and map that result in the lowest quantization error
- Several methods to solve the problem, but of limited use unless the number of colors to be chosen is small
  - It's expensive to compute the optimum
  - It's also a poorly behaved optimization

Perceptual Problems

- While a good quantization may get close colors, humans still perceive the quantization
- Biggest problem: *Mach bands*
  - The difference between two colors is more pronounced when they are side by side and the boundary is smooth
  - This emphasizes boundaries between colors, even if the color difference is small
  - Rough boundaries are “averaged” by our vision system to give smooth variation

Mach Bands in Reality

The floor appears banded

Mach Bands in Reality

Still some banding even in this 24 bit image (the floor in the background)

Dithering (Digital Halftoning)

- Mach bands can be removed by adding noise along the boundary lines
- General perceptive principle: replaced structured errors with noisy ones and people complain less
- Old industry dating to the late 1800's
  - Methods for producing grayscale images in newspapers and books

Next Time

- Dithering
- Sampling
- Signal Processing