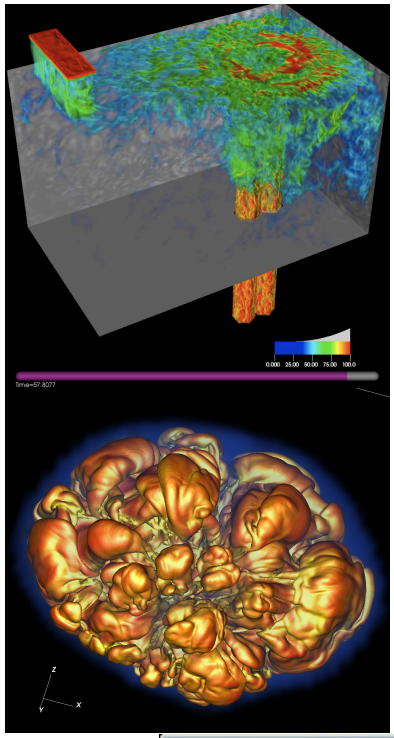
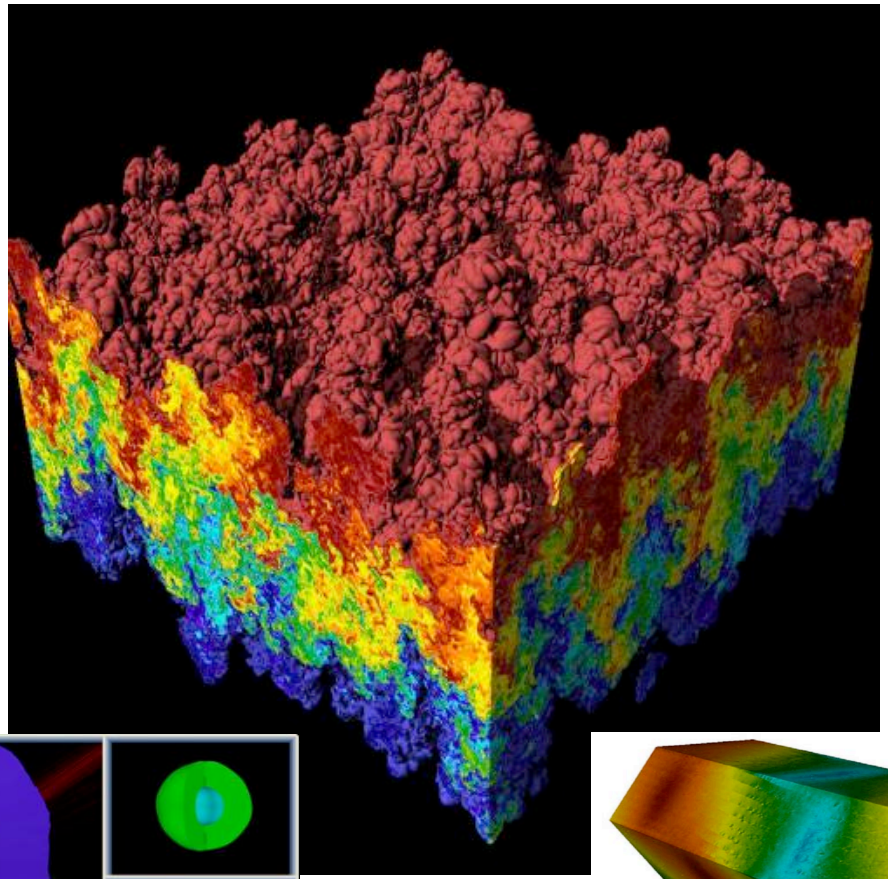
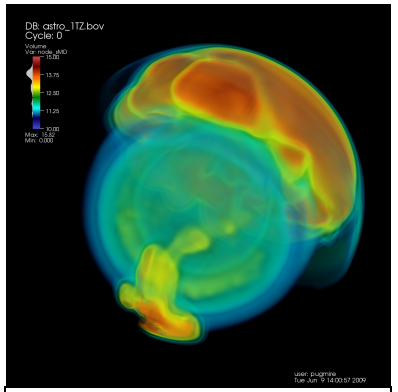
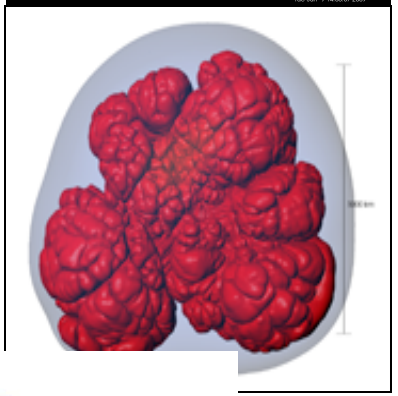
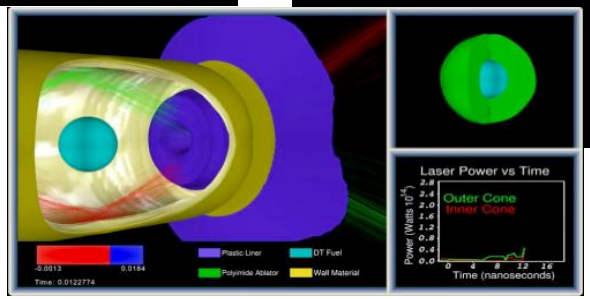
Images and Color Maps

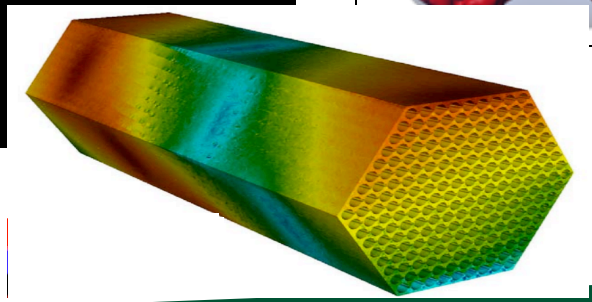












Hank Childs, University of Oregon

Quiz #2 on Thursday Jan 20

- Quiz will go 835-840
- Same format – discuss quiz for 5 minutes 830-835, then take quiz 835-840
- For Zoom: will provide hyperlink if possible
- 2 points

Quiz #2: Topic

- Will be similar to Quiz #1
- But I will make it a little trickier
- I was asked about the answer to Quiz #1 on Thursday. I rewatched the Zoom and believe that my answer was sufficient. I encourage you all to rewatch my response on Thursday's lecture if you have any questions.

Outline

- Images
- Color Maps

Outline

- Images
- Color Maps

Background on Images

- Definitions:
  - Image: 2D array of pixels
  - Pixel: A minute area of illumination on a display screen, one of many from which an image is composed.
- Pixels are made up of three colors: Red, Green, Blue (RGB)
- Amount of each color scored from 0 to 1
  - 100% Red + 100% Green + 0% Blue = Yellow
  - 100% Red + 0% Green + 100 %Blue = Purple
  - 0% Red + 100% Green + 100% Blue = Cyan
  - 100% Red + 100% Blue + 100% Green = White

Background on Images

- Colors are 0->1, but how much resolution is needed? How many bits should you use to represent the color?
  - Can your eye tell the difference between 8 bits and 32 bits?
  - → No. Human eye can distinguish ~10M colors
  - 8bits * 3 colors = 24 bits = ~16M colors
- Red = (255,0,0)
- Green = (0,255,0)
- Blue = (0,0,255)

Note on Project 3

- Colors are expressed as 0->1
- Need to multiply by 255 as you assign to unsigned chars
- Example:

```
float red[3] = { 1.0, 0.0, 0.0 };
unsigned char rgb[3];
rgb[0] = 255*red[0];
rgb[1] = 255*red[1];
rgb[2] = 255*red[2];
```

How to Organize a Struct for an Image (i.e., 3D Arrays)

- 3D array: width * height * 3 color channels
- Color:
  - Choice 1: RGB struct
    - struct rgb { unsigned char r, g, b; };
    - int npixels = width*height;
    - struct RGB *buffer = new RGB[npixels];
    - int p = 21456; // random pixel in the buffer
    - buffer[0].r = 255; buffer[0].g = 0; buffer[0].b = 0;
  - Choice 2: just 3 unsigned chars

How to organize a struct for an Image (i.e., 3D arrays)

- 3D array: width * height * 3 color channels
- Color:
  - Choice 1: RGB struct
  - Choice 2: just 3 unsigned chars
    - int npixels = width*height;
    - unsigned char *buffer = new unsigned char [3*npixels];
    - int p = 21456; // random pixel in the buffer
    - buffer[3*p+0] = 255; buffer[3*p+1] = 0; buffer[3*p+2] = 0;

Project 3 Uses Images

- For project 3, I am doing the management of the buffer
  - I do choice 2
- But you will write functions that work on one pixel
- void AssignValue (unsigned char *pixel)
  { pixel[0] = 255; pixel[1] = 0; pixel[2] = 0; };
- My code:
  - for (int i = 0 ; i < npixels ; i++) AssignValue(buffer+3*i);

Your Amazing Eyes

Crazy numbers about your eyes (with possibly some exaggeration)

- What is the pixel resolution of your eyes?

50*MegaPixels

(* = different parts of the eye work differently; 50M pixel is an aggregation)

- What is the frequency your eyes take in information?

~20* Hertz

(* = for VR, 90Hz is sometimes needed; For video games, 30Hz almost always sufficient)

50MegaPixels x 20HZ → your eyes can take in 1GB of data per second
This is why visualization is king for understanding data

Outline

- Images
- Color Maps

Pseudocolor Plot

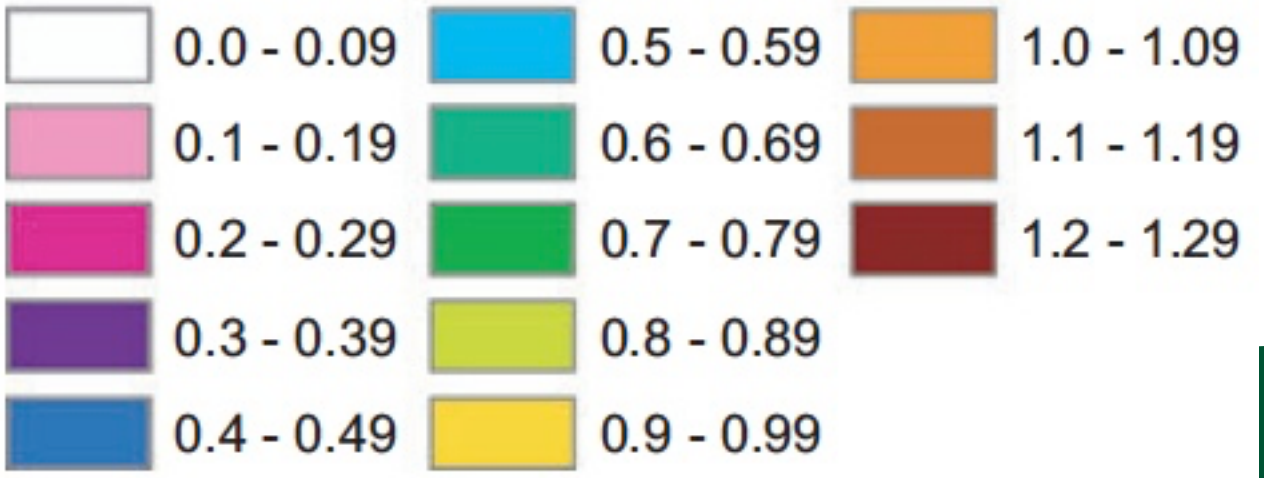
- First visualization technique we will learn
- Idea: take a scalar field on a mesh and transform it to colors
- Two elements:
  - Define a map
  - Apply the map

Pseudocolor Plot

- First visualization technique we will learn
- Idea: take a scalar field on a mesh and transform it to colors
- Two elements:
  - Define a map
  - Apply the map

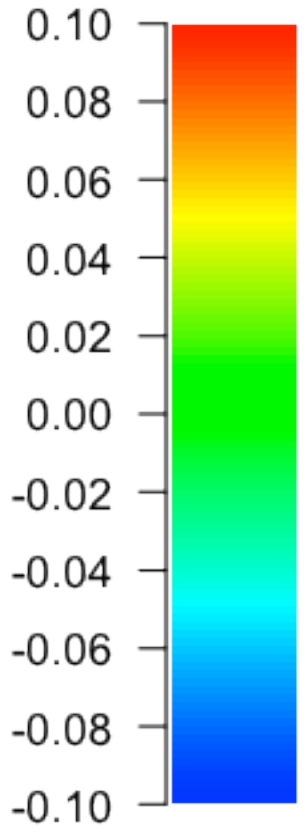
Color Map Example (“Discrete”)

- Example below defines the color for scalar values between 0 and 1.29
- Grouped in 13 ranges (“discrete”)



Color Map Example (“Continuous”)

- Example below defines the color for scalar values between -0.10 and 0.10
- No groupings (“continuous”)
- This color map appears to have a fixed number of colors, and then blends between those colors
  - Red, yellow, green, cyan, blue
    - (This is not immediately obvious)



Pseudocolor Plot

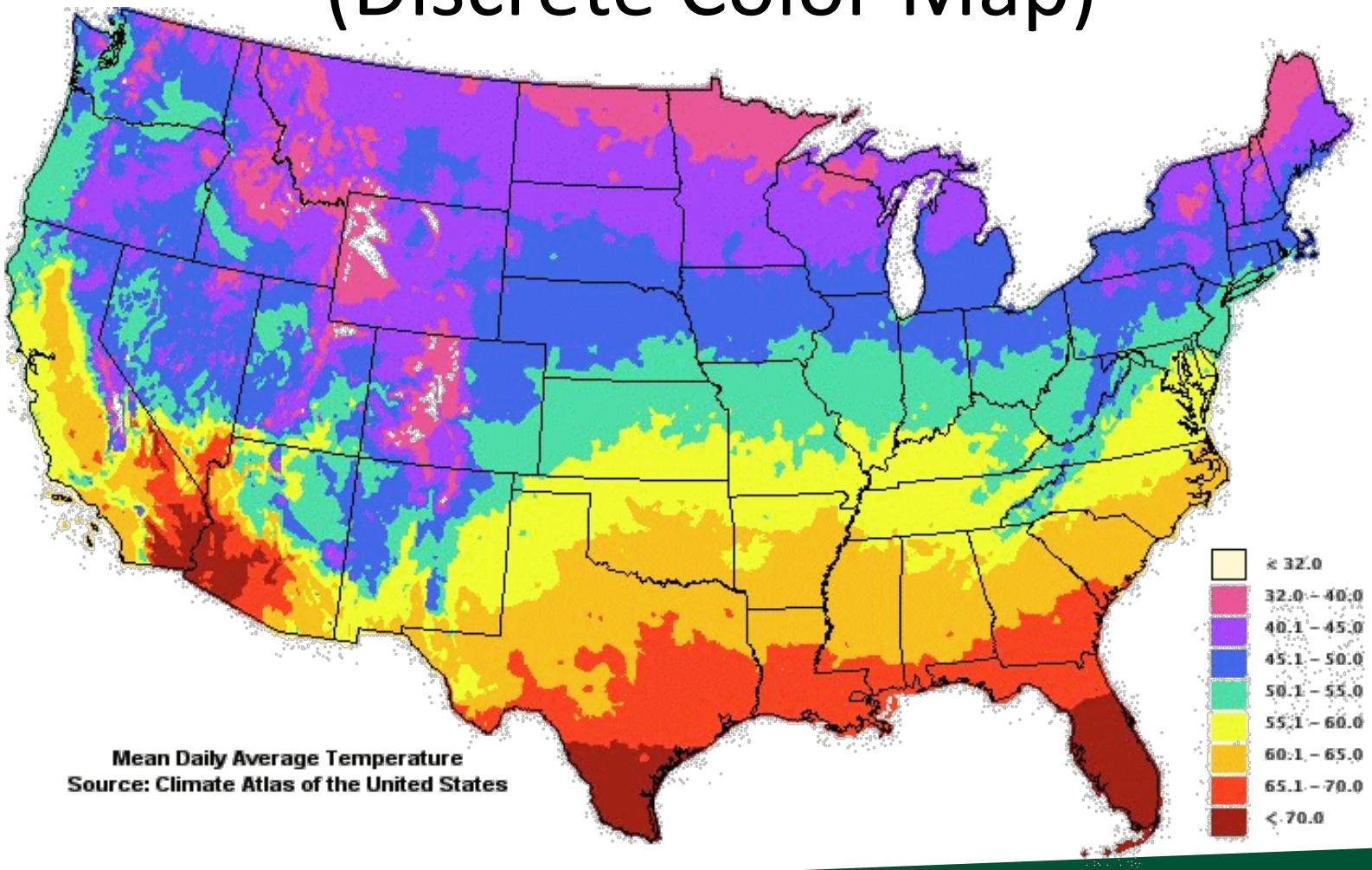
- First visualization technique we will learn
- Idea: take a scalar field on a mesh and transform it to colors
- Two elements:
  - Define a map
  - Apply the map

Pseudocolor Plot

- Defined:
  - Create a mapping, M, from scalar values to colors
  - For each pixel:
    - Evaluate V, the scalar field at that pixel location
    - Obtain color C, by applying M to V

      C = M(V)
    - Assign the pixel color to be C

Example Pseudocolor Plot (Discrete Color Map)



Pseudocolor in Practice

- The normal way:
  - Data set is composed of triangles
  - Apply color map to each vertex in the triangle
  - Tell graphics hardware to render each triangle (including color at each of three vertices)
  - Graphics hardware does the work of interpolating color at each pixel

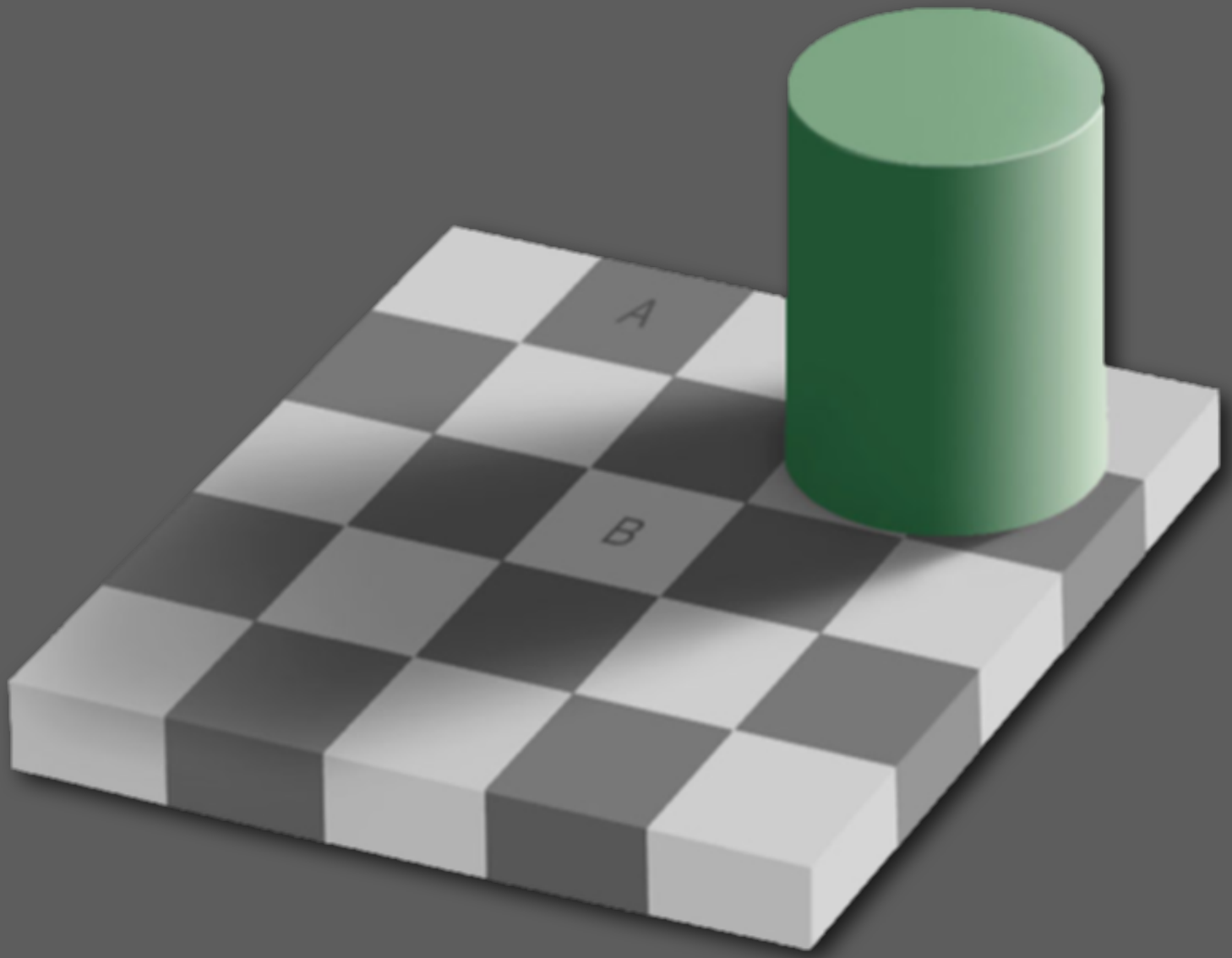
Pseudocolor NOT in Practice

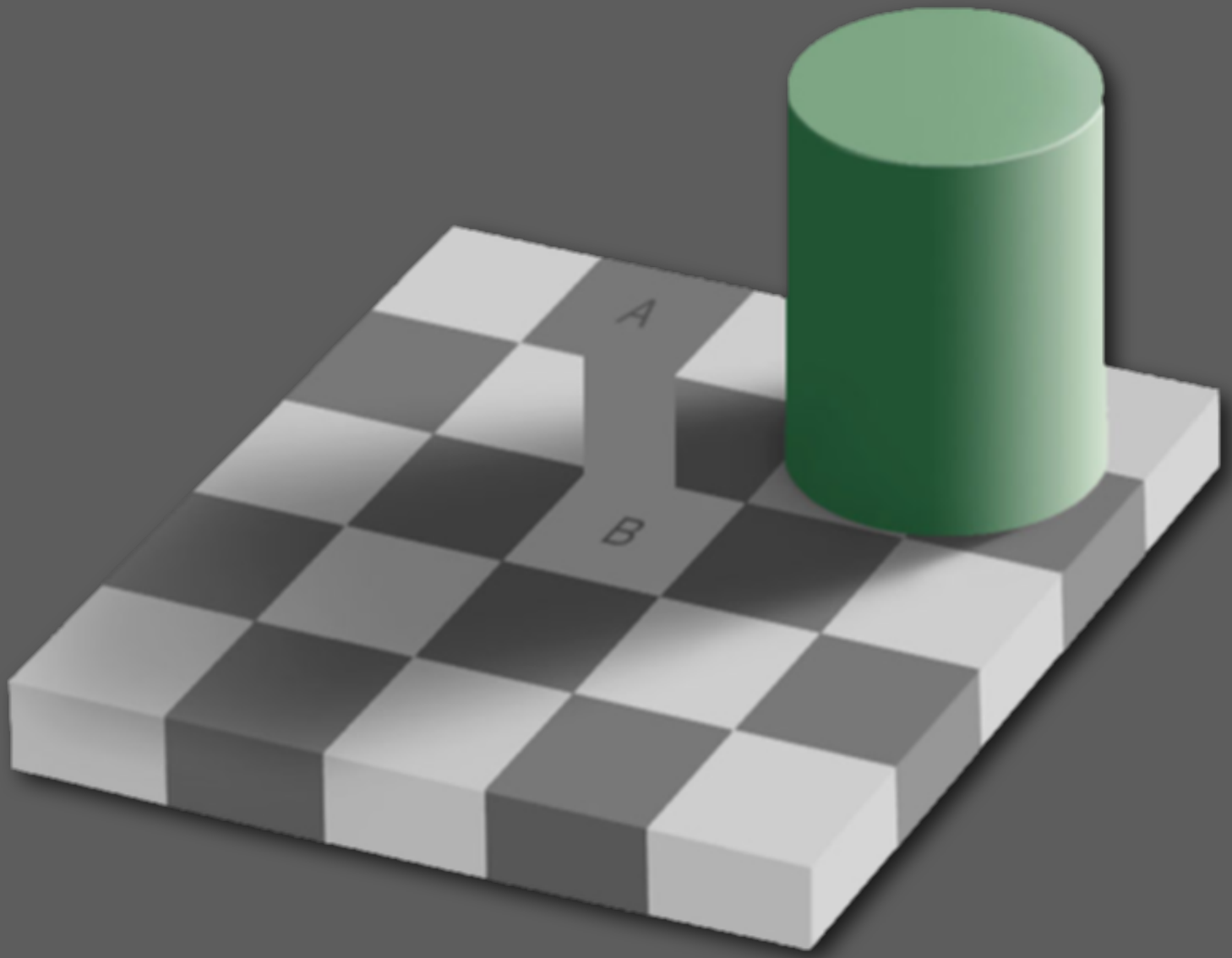
- An unusual way:
  - Assume data perfectly overlaps with image
    - Bottom left of data in bottom left pixel
    - Upper right of data in upper right pixel
  - For each pixel
    - Find corresponding spatial location
    - Evaluate scalar field
    - Apply color map
    - Assign pixel
- In effect: doing what graphics card does in software
- And: this is what we will do in Project 3

“Today’s Lecturer”: Kristi Potter

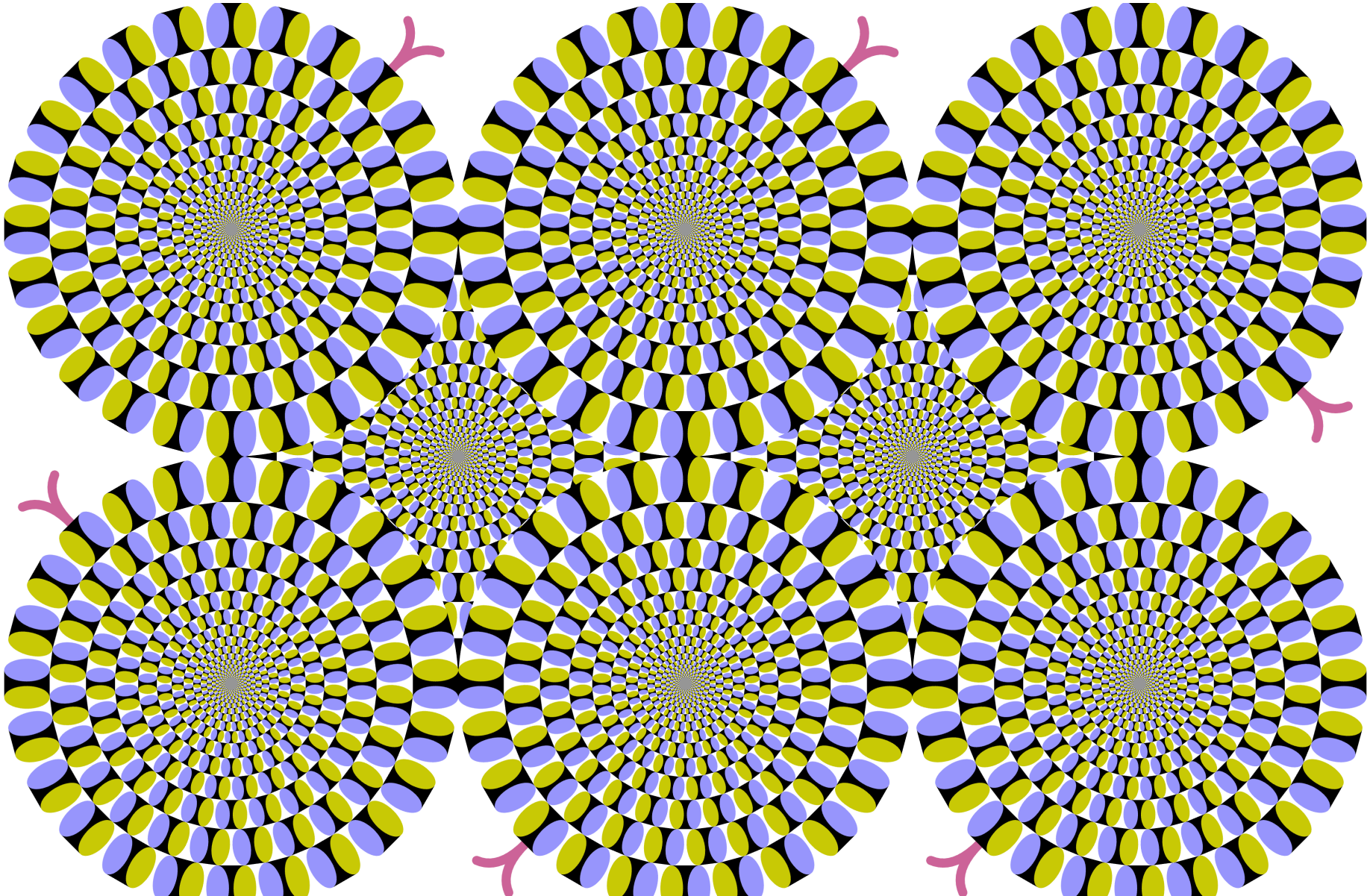
Kristin Potter is a senior scientist specializing in data visualization at the National Renewable Energy Lab’s Insight Center. Her research is focused on methods for improving visualization techniques by adding qualitative information regarding reliability and variability to the data display. This work includes researching statistical measures of uncertainty, error, and confidence levels, and translating the semantic meaning of these measures into visual metaphors. She is also interested in topics related to decision-making, performance visualization, method evaluation, and application-specific techniques. Prior to joining NREL in 2017, she worked as a research computing consultant at the University of Oregon providing visualization services, computational training and education, and other support to researchers across campus. She also was a research scientist at the University of Utah, working on projects related to the visualization of uncertainty and error in data. She received her Ph.D. in 2010 and master’s degree in 2003 from Utah; she received her bachelor’s degree in computer science and fine art in 2000 from the University of Oregon.

Color Illusions

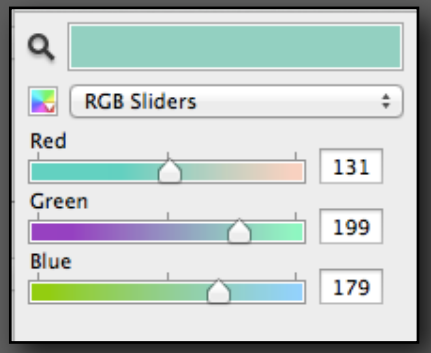

A
B

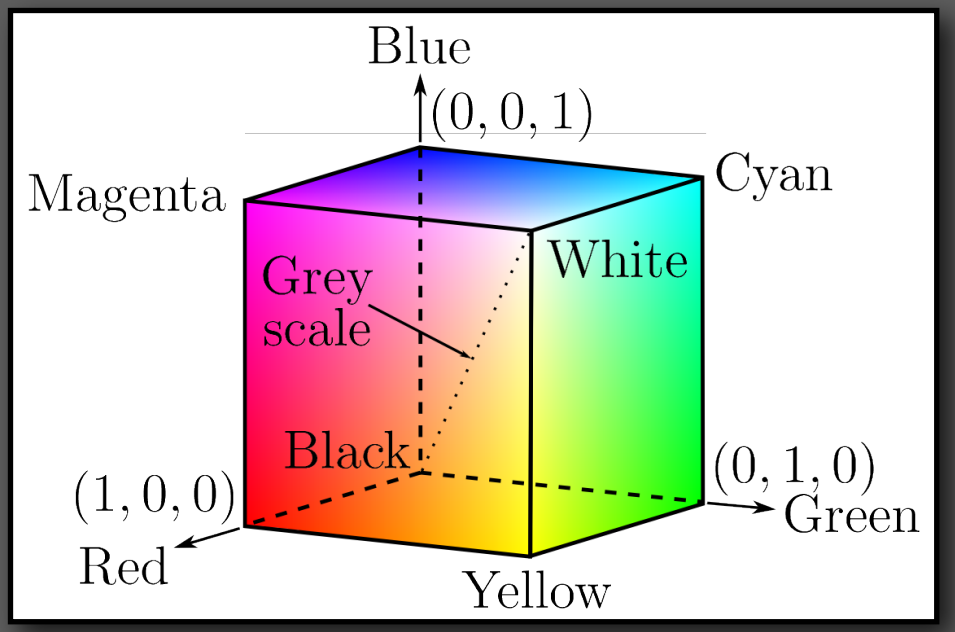

A
B

Here's the better one



Color Spaces

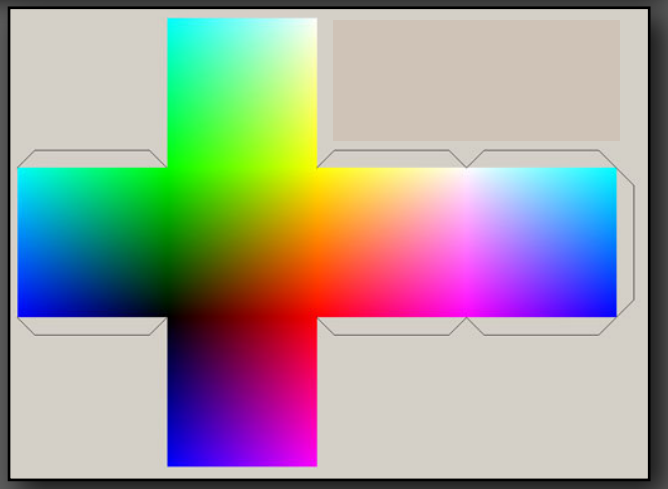


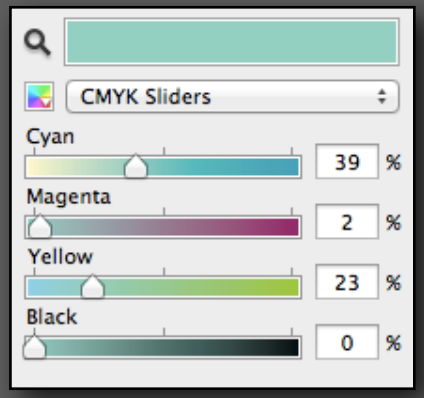


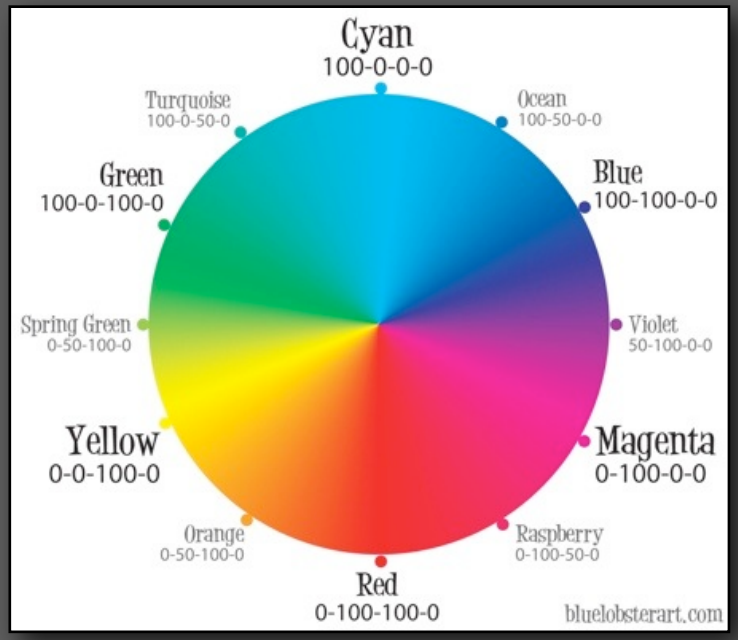
RGB

Red, Green, Blue channels

- electron guns of crt monitors



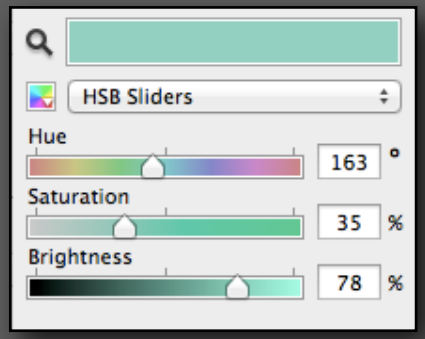


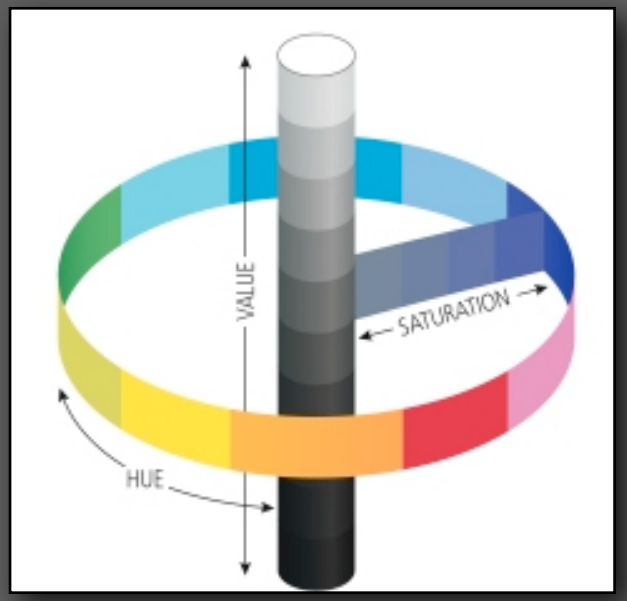


CMYK

subtractive

- **C**yan, **M**agenta, **Y**ellow, **K**ey (black)
- inks used in printing
- assumes white paper (non-existence of ink)
- often required for paper publications
- device dependent

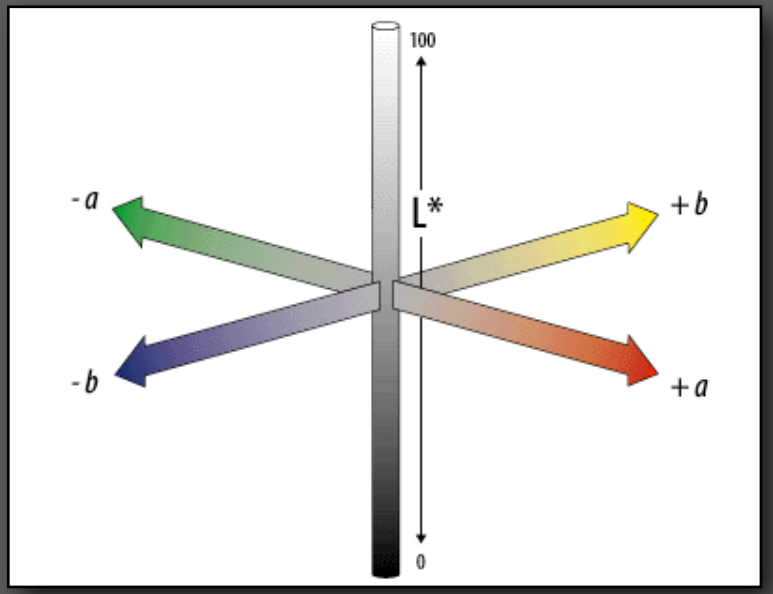




HSV [B, L, I]

additive

- **H**ue, **S**aturation, [**V**alue, **B**rightness, **L**ightness, **I**ntensity]
-polar coordinate representations of RGB space
- conical or cylindrical shaped space
- more intuitive than RGB for color tuning



CIE LAB/LUV

mathematically defined, perceptually based

- **C**ommission **I**nternationale de l'**é**clairage)
- **L**ightness, (**a**,**b**)/(**u**,**v**) color opponent dimensions

a: blue/yellow, b: green/red

lightness (0% black, 100% white)

- space includes all perceivable colors

(gamut greater than RGB, CMYK)

- device independent

RGB to HSV *C++ (ish)*

```
// in: rgb(0.0,1.0) out: h(0,360), v(0.0,1.0), s(0.0,1.0)
rgb2hsv(float r, float g, float b)
      // find the min and max of rgb
      maxColor = max(r, g, b);
      minColor = min(r, g, b);
      float delta = maxColor - minColor;
      float Value = maxColor;
      float Saturation = 0;
      float Hue = 0;
      if (delta == 0)
      Hue = 0; Saturation = 0;
      else
          Saturation = delta / maxColor;
      float dR = 60.f*(maxColor - r)/delta + 180.0;
      float dG = 60.f*(maxColor - g)/delta + 180.0;
      float dB = 60.f*(maxColor - b)/delta + 180.0;
      if (r == maxColor)
            Hue = dB - dG;
      else if (g == maxColor)
      Hue = 120 + dR - dB;
      else
      Hue = 240 + dG - dR;
      if (Hue < 0)
         Hue+=360;
      if (Hue > =360)
         Hue-=360;
```

HSV to RGB *C++ (ish)*

```
// in: h(0,360), v(0.0,1.0), s(0.0,1.0) out: rgb(0.0,1.0)
hsvToRGB(float hue, float saturation, float value)
  if( saturation == 0 ) // achromatic (grey)
     r = g = b = v;
  else{
          hue /= 60.f;
          // sector 0 to 5
          i = floor( hue );
          f = hue - i;
          // factorial part of h
          p = value * ( 1 - saturation);
          q = value * ( 1 - saturation * f );
          t = value * ( 1 - saturation * ( 1 - f ) );
          switch( i ):
            case 0: r = v; g = t; b = p;
            case 1:    r = q; g = v; b p;
            case 2:    r = p; g = v; b = t;
            case 3:    r = p; g = q; b = v;
            case 4:    r = t; g = p; b = v;
            case 5:    r = v; g = p; b = q;
}
```

Problems with Color

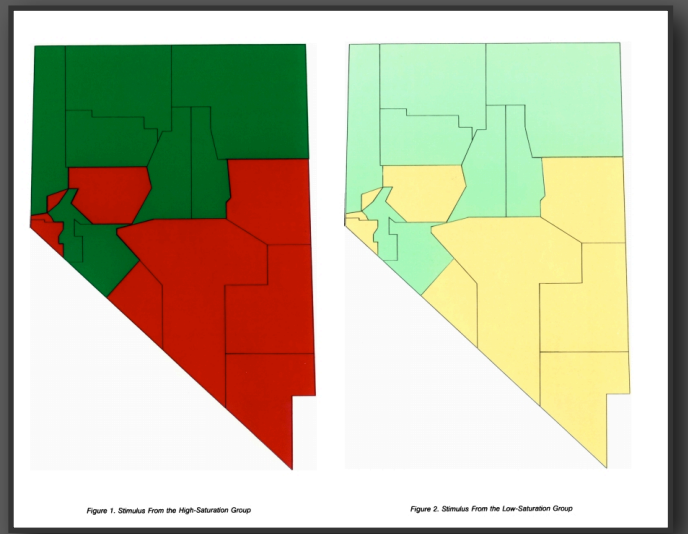
(short list)

Misused Color

"Color used poorly is worse than no color at all."

-Edward Tufte

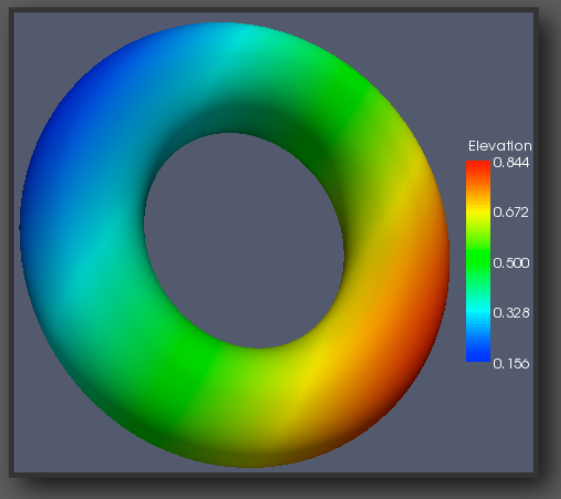
- bias a reader's perception
- cause the wrong information stand out
- hide meaningful information
- cause visual clutter and overload



Color Coding

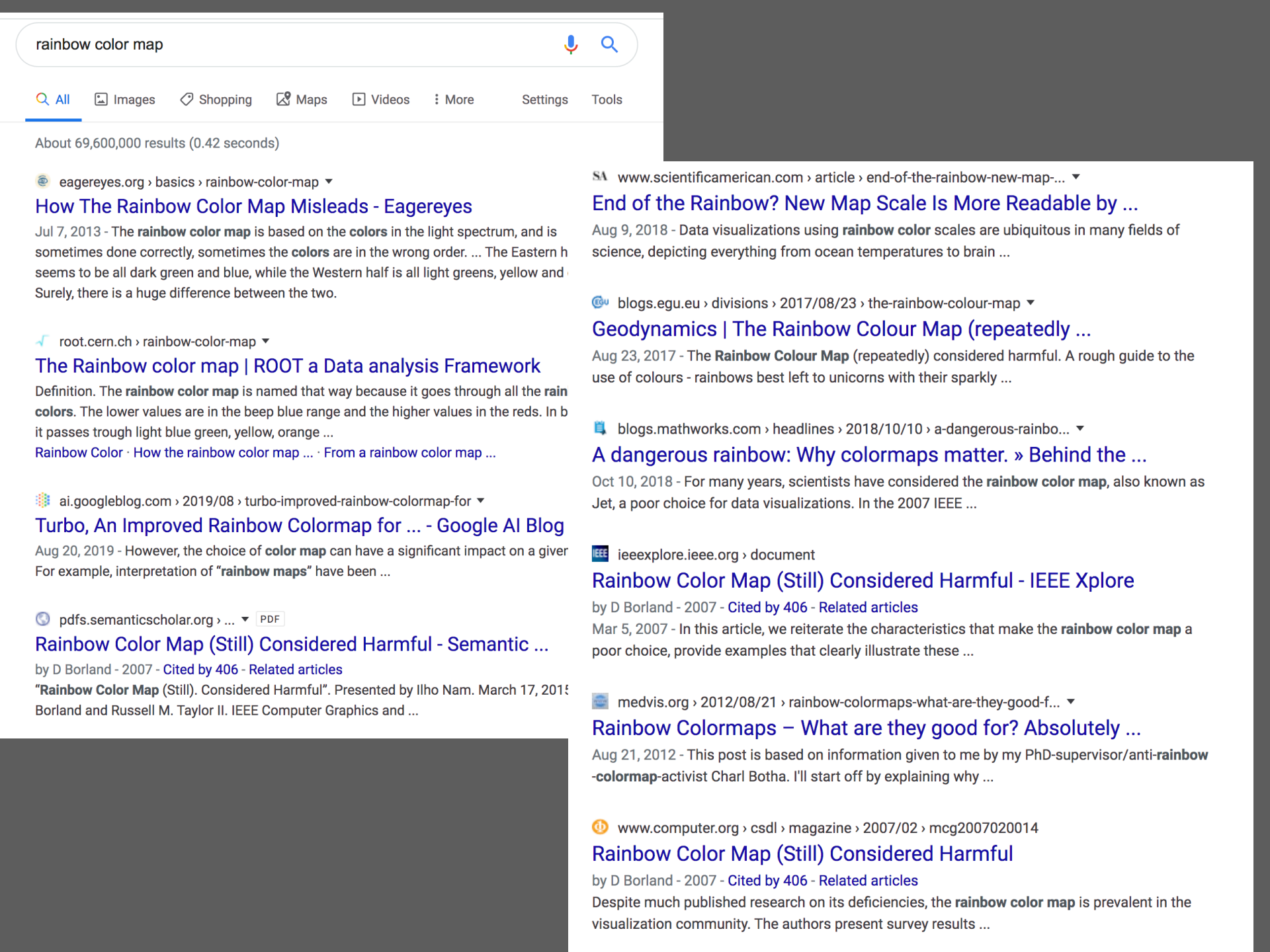
- hard to pick discernible colors
- can only get ~12 distinguishable colors
- object size, line width can influence perception

William S. Cleveland and Robert McGill.
"A Color-Caused Optical Illusion on a Statistical Graph".
In *The American Statistician,* vol. 37, no. 2, pp. 101--105, 1983.



Rainbow Colormaps

- not ideal for qualitative data
- no inherent ordering (by wavelength?)
- transitions between colors are uneven
- mapping of changing value may not equal the change we see


rainbow color map
All
Images
Shopping
Maps
Videos
More
Settings
Tools
About 69,600,000 results (0.42 seconds)
eagereyes.org › basics › rainbow-color-map
How The Rainbow Color Map Misleads - Eagereyes
Jul 7, 2013 - The rainbow color map is based on the colors in the light spectrum, and is sometimes done correctly, sometimes the colors are in the wrong order. ... The Eastern h seems to be all dark green and blue, while the Western half is all light greens, yellow and Surely, there is a huge difference between the two.
root.cern.ch › rainbow-color-map
The Rainbow color map | ROOT a Data analysis Framework
Definition. The rainbow color map is named that way because it goes through all the rain colors. The lower values are in the beep blue range and the higher values in the reds. In b it passes trough light blue green, yellow, orange ...
Rainbow Color · How the rainbow color map ... · From a rainbow color map ...
ai.googleblog.com › 2019/08 › turbo-improved-rainbow-colormap-for
Turbo, An Improved Rainbow Colormap for ... - Google AI Blog
Aug 20, 2019 - However, the choice of color map can have a significant impact on a giver For example, interpretation of "rainbow maps" have been ...
pdfs.semanticscholar.org › ... PDF
Rainbow Color Map (Still) Considered Harmful - Semantic ...
by D Borland - 2007 - Cited by 406 - Related articles
"Rainbow Color Map (Still). Considered Harmful". Presented by Ilho Nam. March 17, 201
Borland and Russell M. Taylor II. IEEE Computer Graphics and ...
www.scientificamerican.com › article › end-of-the-rainbow-new-map-...
End of the Rainbow? New Map Scale Is More Readable by ...
Aug 9, 2018 - Data visualizations using rainbow color scales are ubiquitous in many fields of science, depicting everything from ocean temperatures to brain ...
blogs.egu.eu › divisions › 2017/08/23 › the-rainbow-colour-map
Geodynamics | The Rainbow Colour Map (repeatedly ...
Aug 23, 2017 - The Rainbow Colour Map (repeatedly) considered harmful. A rough guide to the use of colours - rainbows best left to unicorns with their sparkly ...
blogs.mathworks.com › headlines › 2018/10/10 › a-dangerous-rainbo...
A dangerous rainbow: Why colormaps matter. » Behind the ...
Oct 10, 2018 - For many years, scientists have considered the rainbow color map, also known as Jet, a poor choice for data visualizations. In the 2007 IEEE ...
ieeexplore.ieee.org › document
Rainbow Color Map (Still) Considered Harmful - IEEE Xplore
by D Borland - 2007 - Cited by 406 - Related articles
Mar 5, 2007 - In this article, we reiterate the characteristics that make the rainbow color map a poor choice, provide examples that clearly illustrate these ...
medvis.org › 2012/08/21 › rainbow-colormaps-what-are-they-good-f...
Rainbow Colormaps – What are they good for? Absolutely ...
Aug 21, 2012 - This post is based on information given to me by my PhD-supervisor/anti-rainbow-colormap-activist Charl Botha. I'll start off by explaining why ...
www.computer.org › csdl › magazine › 2007/02 › mcg2007020014
Rainbow Color Map (Still) Considered Harmful
by D Borland - 2007 - Cited by 406 - Related articles
Despite much published research on its deficiencies, the rainbow color map is prevalent in the visualization community. The authors present survey results ...

Best Practices

If you want objects to look the same color,
make background colors consistent.

If you want objects to be easily seen, use a background color that contrasts sufficiently.

Use color only when needed to serve a particular communication goal.

Use different colors only when they correspond to differences of meaning in the data.

Use soft, natural colors to display most information and bright and/or dark colors to highlight.

Stick with a monochromatic color scheme when encoding quantitative values.

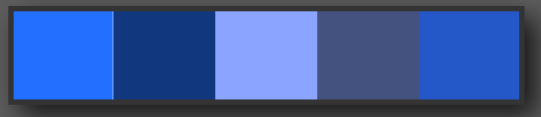
Non-data components should be displayed just visibly enough to perform their role.

To accommodate for the colorblind, avoid using a combination of red and green in the same display.

Vischeck/Daltonize
http://www.vischeck.com/

Tips

monochromatic



analogous

complementary

Color Schemes

- monochromatic

variations in lightness & saturation of a single color

- analogous

colors adjacent on the color wheel

-complementary

two colors opposite on the color wheel

Look to Nature

Simplicity

- choose one color to be used in larger amounts
- be selective about the base color
- use other colors to add interest

Avoidance of Color

- use neutrals (work with any scheme)

black, white, grey

- use diagrammatic marks (may be better encoding channels)

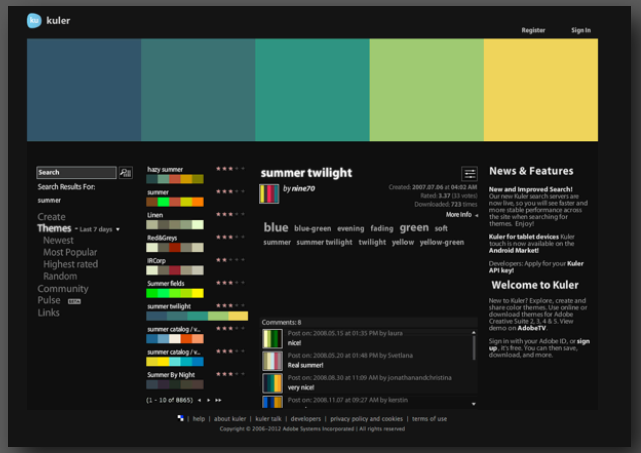
size, shape, texture, length, width,
orientation, curvature and intensity

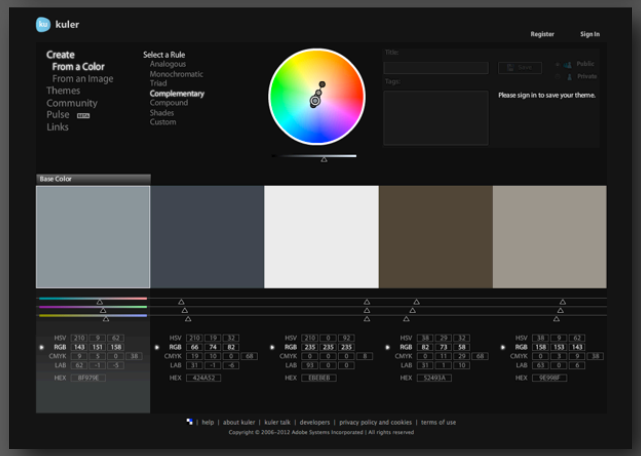
Tools for Color Scheme Creation

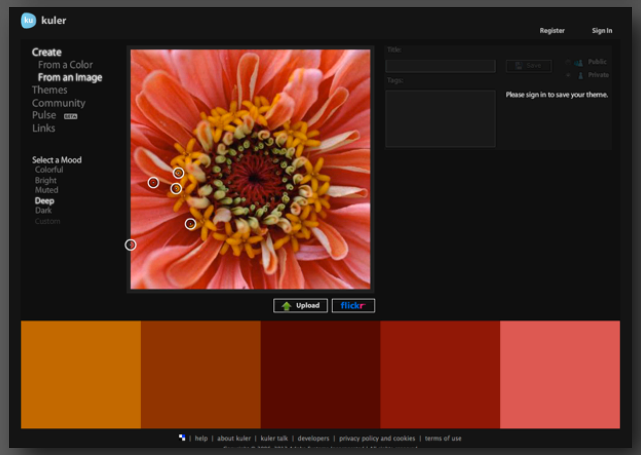
Kuhler

http://kuler.adobe.com

- pick a previously created “theme”
- create a theme from color model
- create a new theme from an image



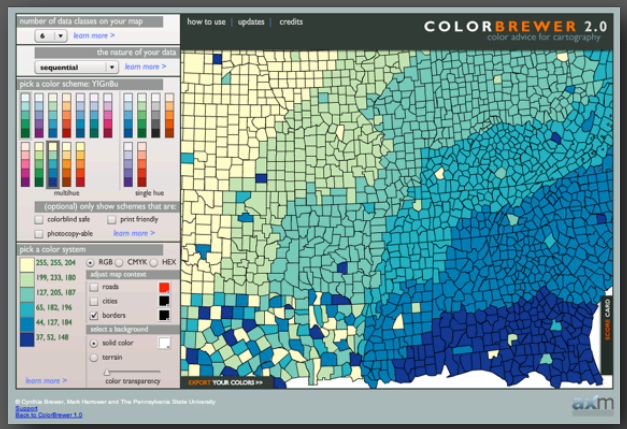


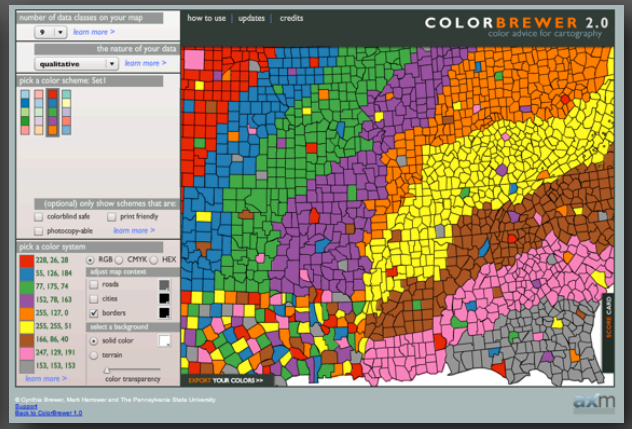


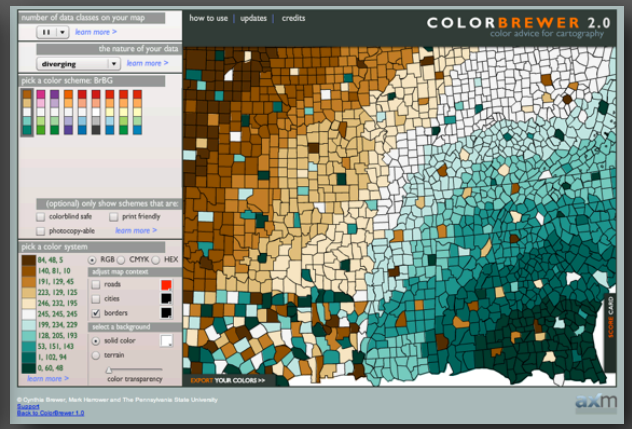
Color Scheme Designer

http://colorschemedesigner.com

- develop scheme based on a color model
- test out scheme on light and dark pages







Color Brewer

http://colorbrewer2.org/

- pre-defined color schemes for maps

sequential

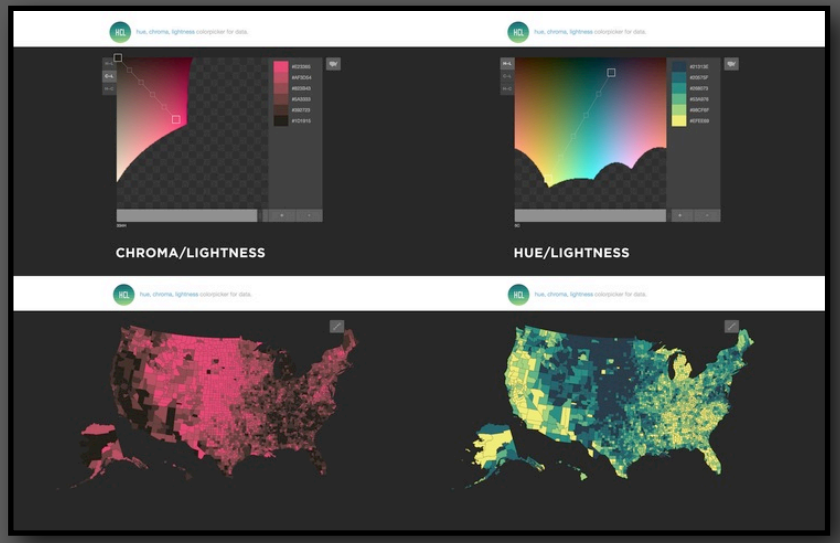
optimized for data ordered low to high

diverging

place equal emphasis on mid-range and extremes

qualitative

does not imply an order (categorical data)



HCL Color Picker

http://tristen.ca/hcl-picker/

- pick equidistant colors in the HCL colorspace
- test color scheme out on a choropleth map
- addresses issues of HSV colorspace

Questions?

References

- Cinematic Color, Jeremy Selan, Siggraph 2012 Course notes
- Measuring Color [Hunt, 1998]
- Color Imaging: Fundamentals and Application [Reinhard, et al. 2008]
- Color Science [Wyszecki and Stiles, 1982]
- The Science of Art (http://www.imprint.co.uk/rama/art.pdf)
- How the Retina Works (http://www.americanscientist.org/libraries/documents/200583133632_306.pdf)
- Color Theory Methods for Visualization, Theresa-Marie Rhyne, Vis 2012 Tutorial

Project 3

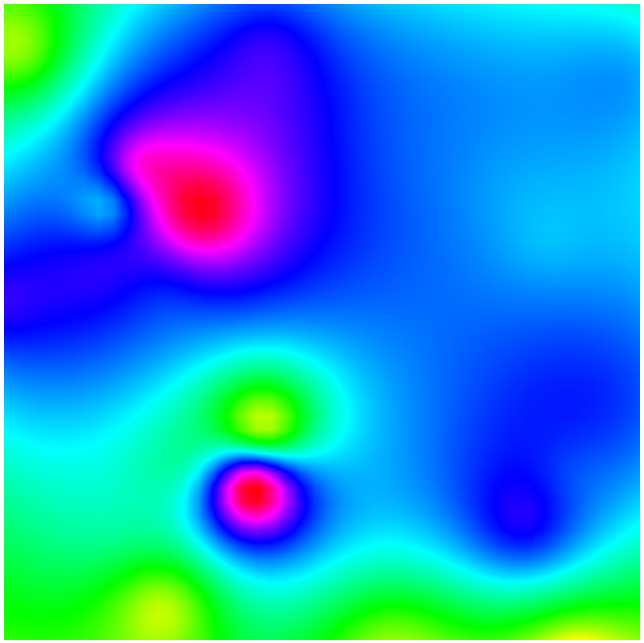
- Assigned today, prompt online
- Due Monday Jan 24th
- Worth 7% of your grade
- Provide:
  - Code skeleton online
  - Correct answers provided
- You upload
  - source code
- Wrong answers?: <1/2 credit
  - Differencer program available

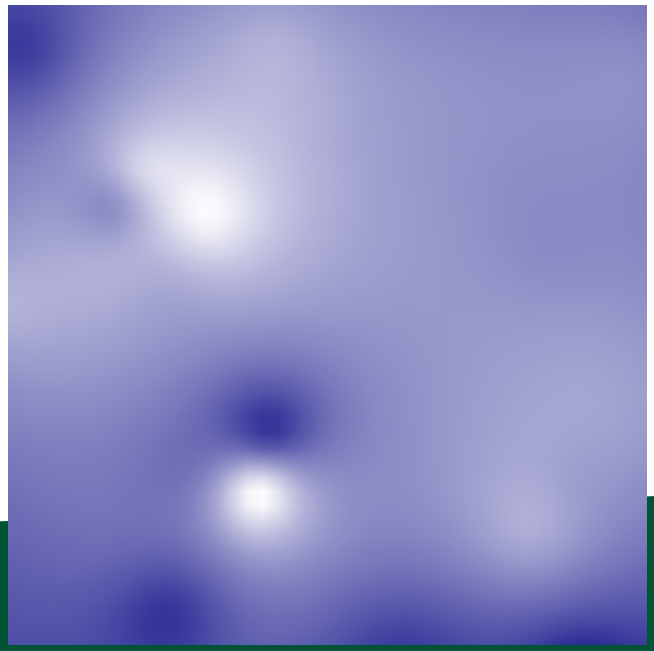
Project 3 in a nutshell

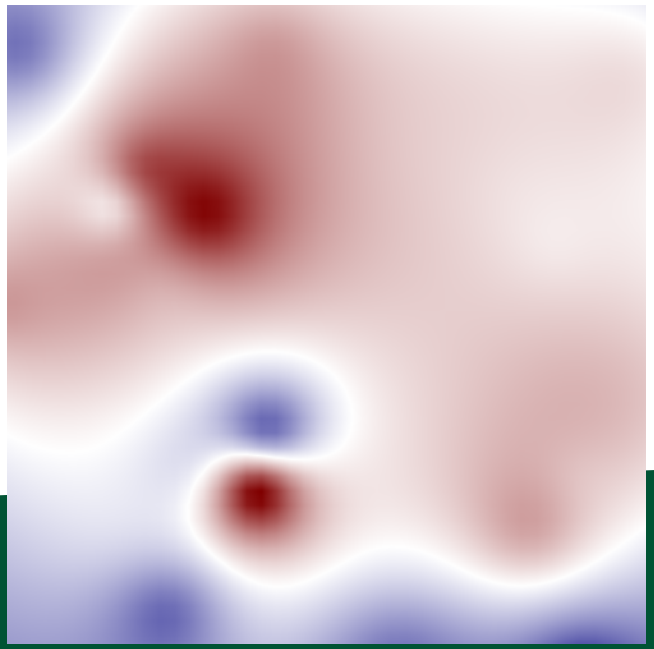
- I give you a 2D data set
- You will produce 3 images that are 500x500 pixels
- The 2D data set will be mapped on to the pixels
- Pixel (0,0): X=-9, Y=-9
- Pixel (499, 0): X=+9, Y=-9, pixel (0,499): X=-9, Y=+9
- Pixel (499,499): X=+9, Y=+9

Project 3 in a nutshell

- For each of the 250,000 pixels (500x500), you will apply 3 color maps to a scalar field







Floating-Point Differences

Differencer program / floating point differences

Hank Childs

All Sections

Jan 17 at 3:51pm

Hi Everyone,

For project #3, we will generate images with different color maps. The process to generate these images involves floating point math. While we all may be doing the same thing conceptually, we may use slightly different formulations to get there. Occasionally, this can lead to slightly different images. The rate at which these differences occurs is very low. For example, I just ran an example where two different implementations differed on 4 out of 250,000 pixel calculations. That said, we still need to tolerate a little error.

I have provided a "differencer" program that will allow you to difference your output with the reference output. If the number of pixels different is <=20 pixels, then we will declare this correct (full credit). If it is more than that, then we will declare it incorrect (less than half credit).

I include an example of running differencer below. More information on how to set up differencer are on the class webpage.

Best,
Hank

```
# compares hsv.png generated by my proj3 program with the hsv.png file I downloa
% ~/differencer/differencer.app/Contents/MacOS/differencer hsv.png ~/Downloac
Both images have 250000 pixels, good.
Difference at column = 341, row = 54
   File hsv.png has 0, 255, 201
   File /Users/hank/Downloads/hsv.png has 0, 255, 202
Difference at column = 176, row = 329
   File hsv.png has 255, 0, 132
   File /Users/hank/Downloads/hsv.png has 255, 0, 131
```

Differencer

```
# compares hsv.png generated by my proj3 program with the hsv.png file I downloaded from the class webpage
% ~/differencer/differencer.app/Contents/MacOS/differencer hsv.png ~/Downloads/hsv.png
Both images have 250000 pixels, good.
Difference at column = 341, row = 54
   File hsv.png has 0, 255, 201
   File /Users/hank/Downloads/hsv.png has 0, 255, 202
Difference at column = 176, row = 329
   File hsv.png has 255, 0, 132
   File /Users/hank/Downloads/hsv.png has 255, 0, 131
Difference at column = 204, row = 342
   File hsv.png has 207, 0, 255
   File /Users/hank/Downloads/hsv.png has 208, 0, 255
Difference at column = 210, row = 385
   File hsv.png has 131, 0, 255
   File /Users/hank/Downloads/hsv.png has 132, 0, 255
The number of different pixels is 4
Writing file differenceMap.png
```