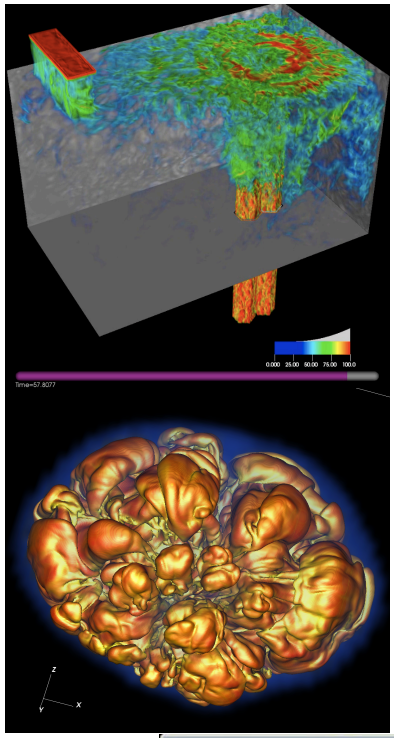
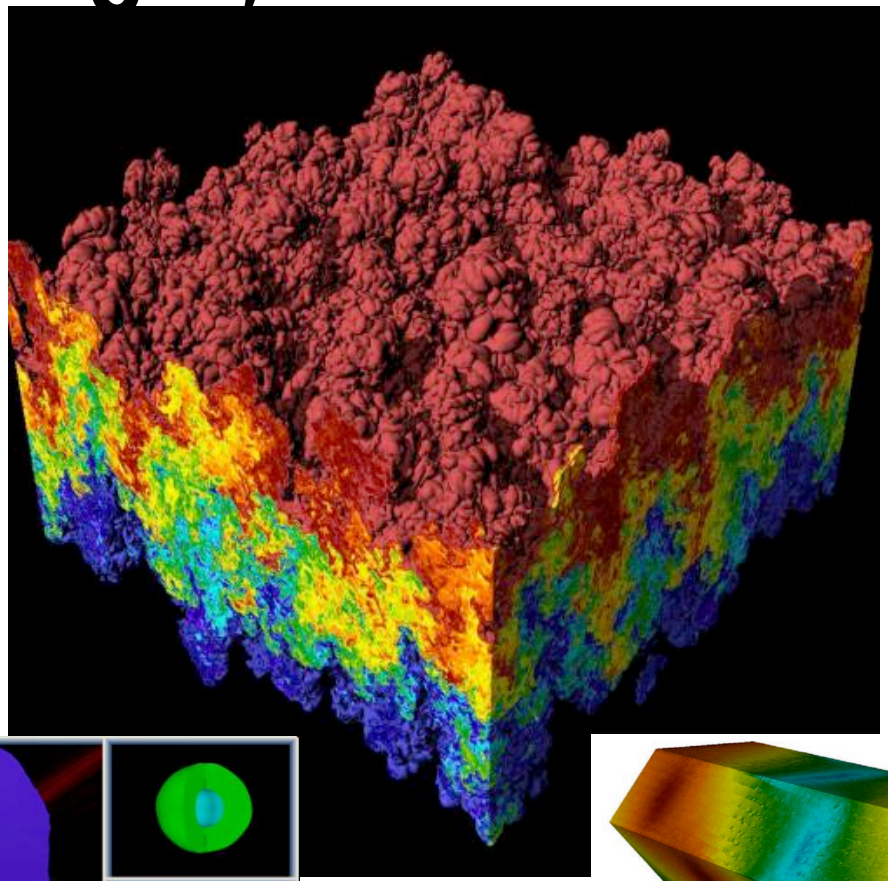
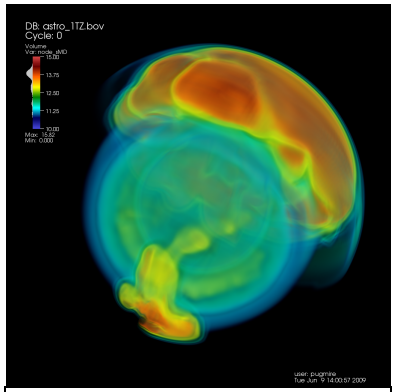
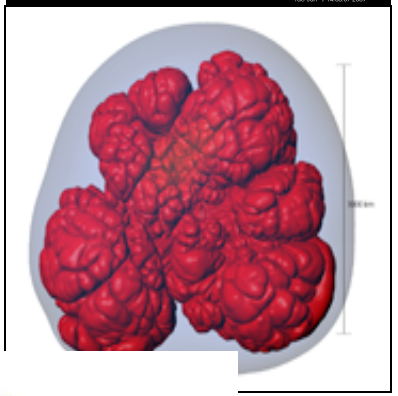
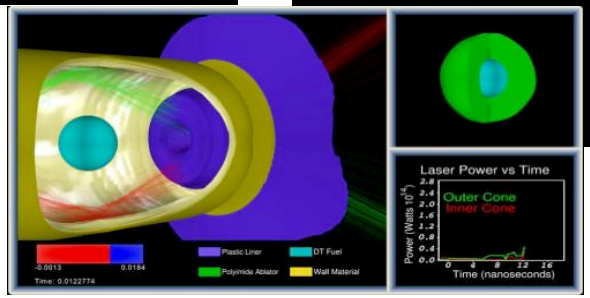
Interpolation, Cell Location, Images, and Color Maps

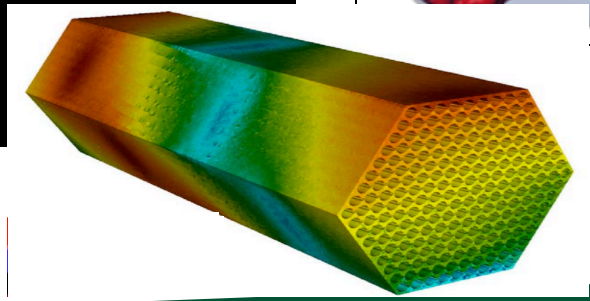












Hank Childs, University of Oregon

Notes

- Very important to me everyone has symmetric experience
  - Canvas: cuts off at 8:39am. I believe this will be 8:39:59am and not 8:39:00. If not, please put answers to me as direct message in the chat.
  - In person: pencils up at 8:39:59. If you are still writing after I call time, your score is 0.

Let’s Look at the Quiz

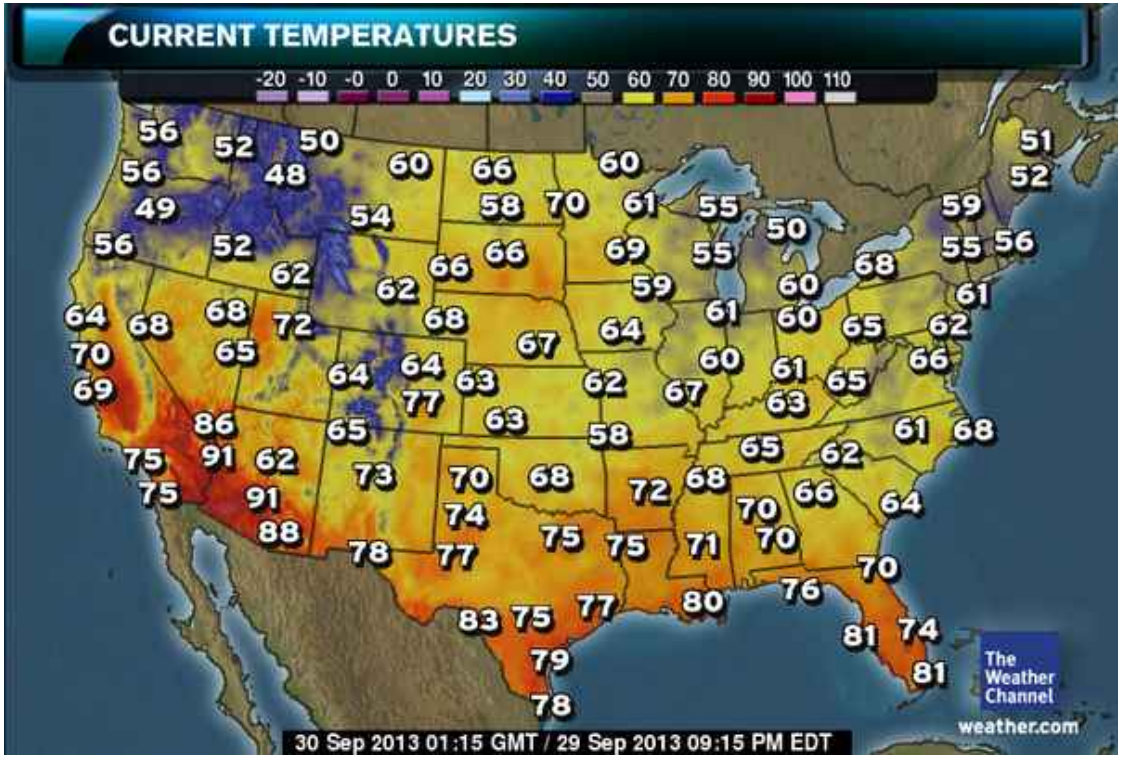
Outline

- Interpolation
- Location
- Images
- Color Maps

Outline

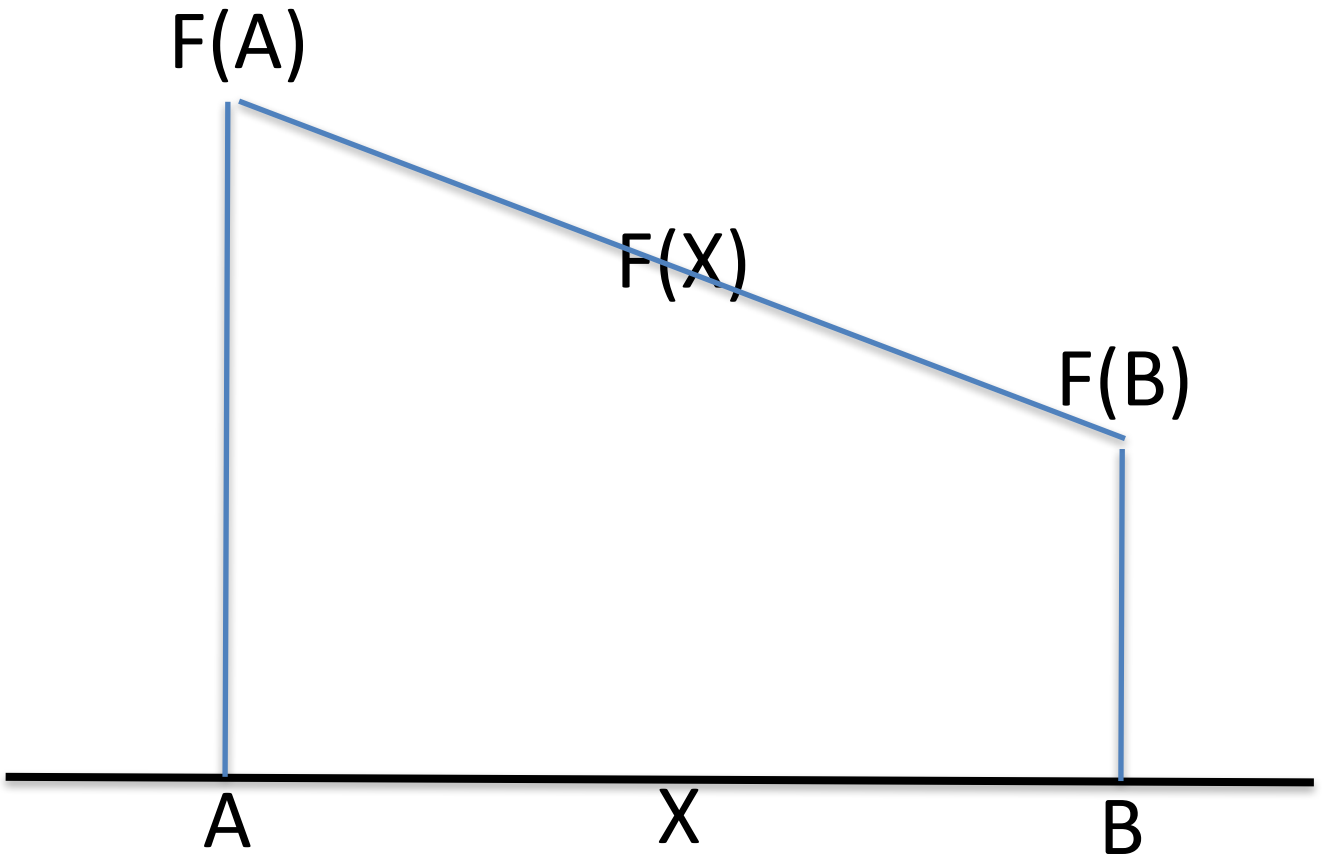
- Interpolation
- Location
- Images
- Color Maps

Linear Interpolation for Scalar Field F



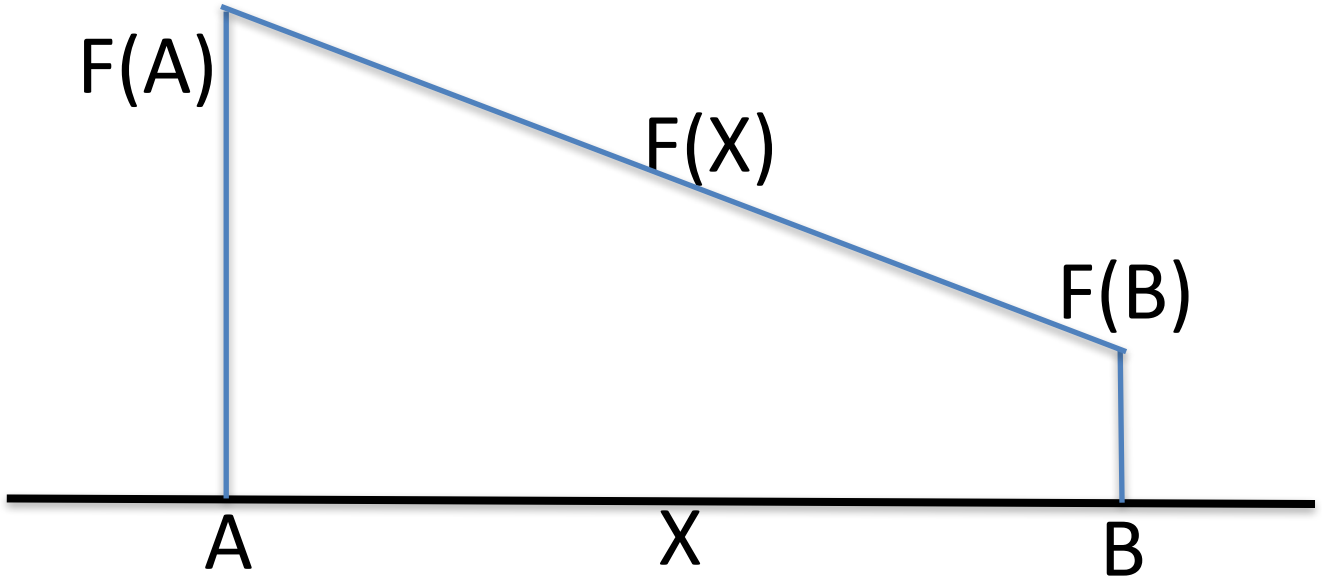
Goal: have data at some points & want to interpolate data to any location

Linear Interpolation for Scalar Field F



Linear Interpolation for Scalar Field F

- General equation to interpolate:
  - F(X) = F(A) + t*(F(B)-F(A))
- t is proportion of X between A and B
  - t = (X-A)/(B-A)

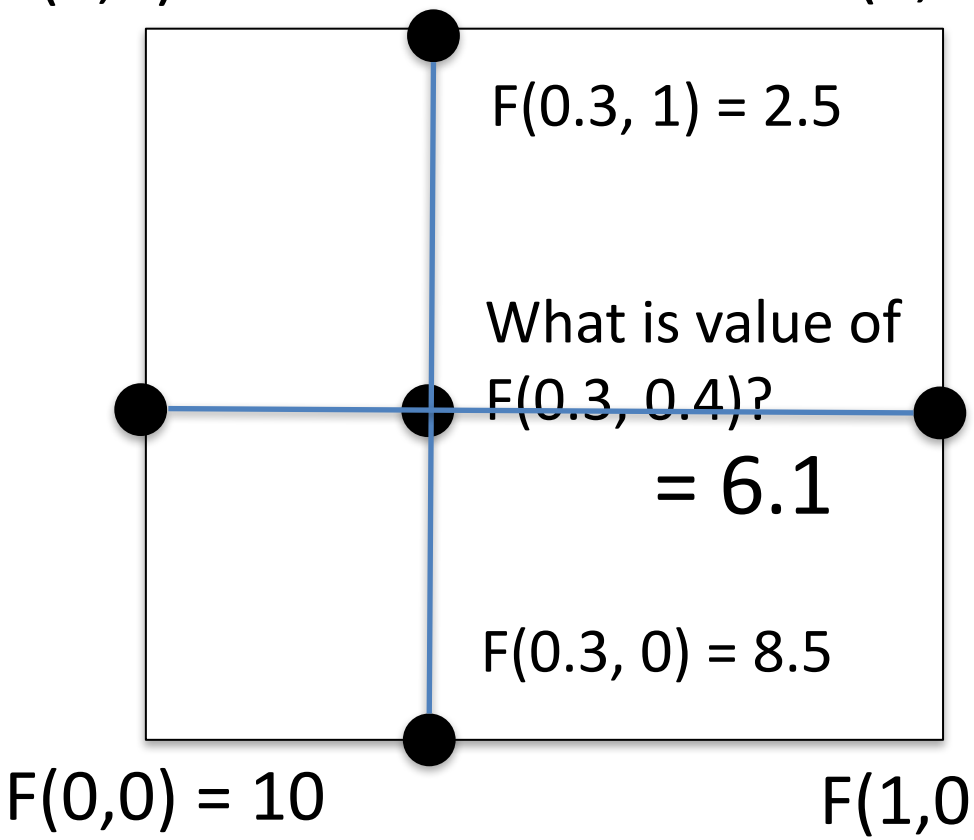


Quiz Time #4

- F(3) = 5, F(6) = 11
- What is F(4)? = 5 + (4-3)/(6-3)*(11-5) = 7

- General equation to interpolate:
  - F(X) = F(A) + t*(F(B)-F(A))
- t is proportion of X between A and B
  - t = (X-A)/(B-A)

Bilinear interpolation for Scalar Field F



Idea: we know how to interpolate along lines. Let's keep doing that and work our way to the middle.

- General equation to interpolate:

$$F(X) = F(A) + t*(F(B)-F(A))$$

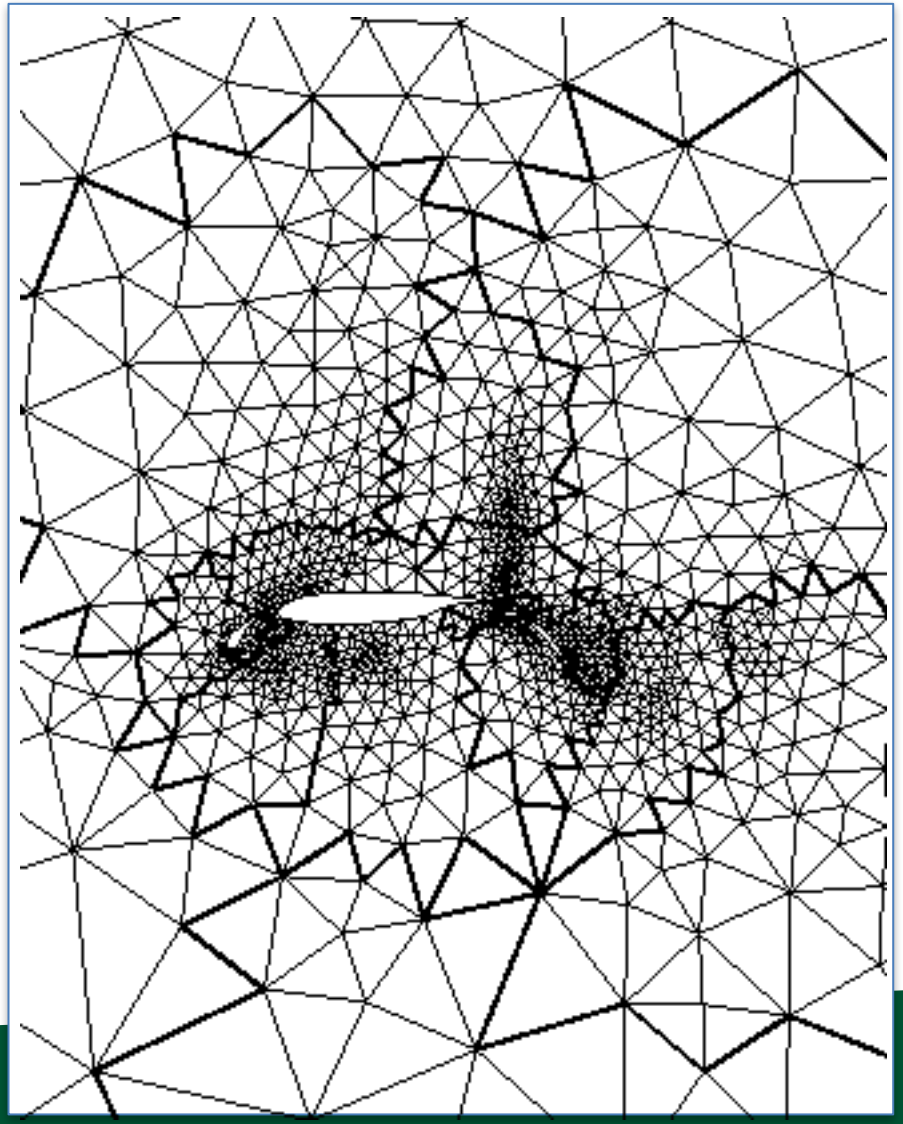
Cell Location

Cell Location

- Problem definition: you have a physical location (P). You want to identify which cell contains P.
- Solution: multiple approaches that incorporate spatial data structures.
  - Best data structure depends on nature of input data.
    - More on this later in the quarter.

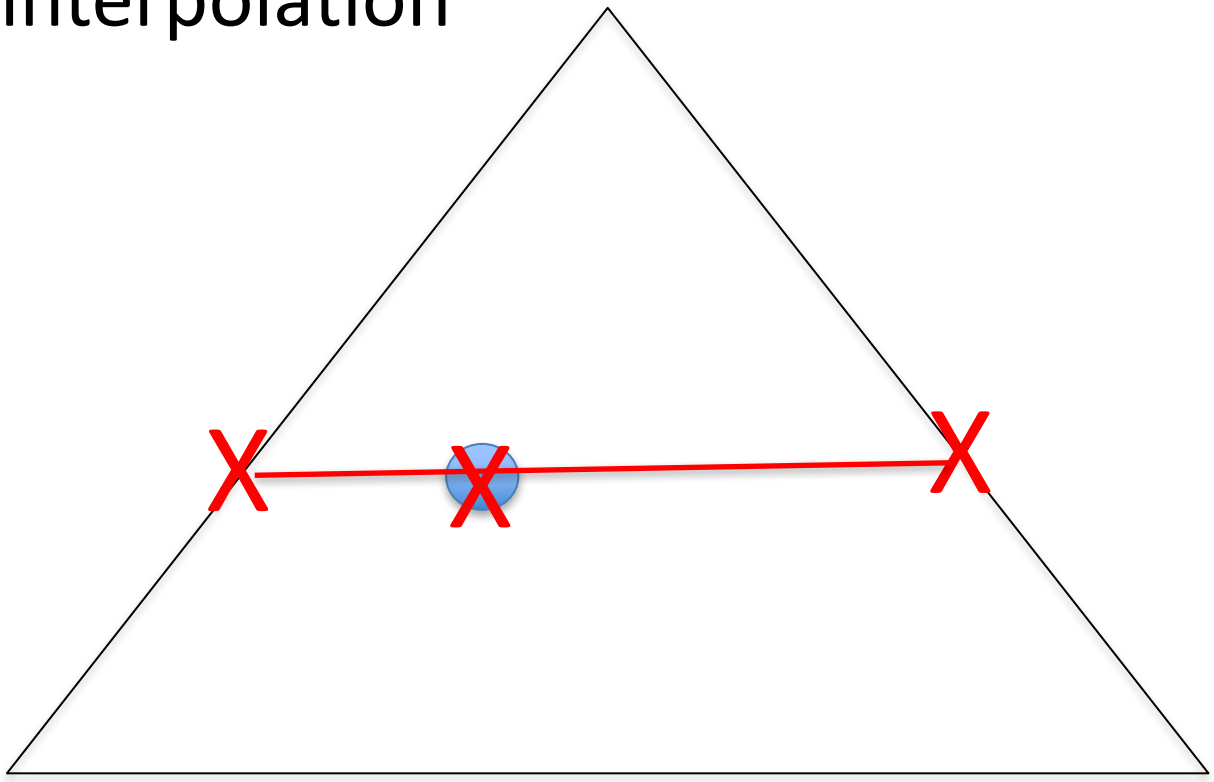
Interpolation for triangle meshes

- Two issues:
  - (1) how to locate triangle that contains point
    - (discuss in 5 slides)
  - (2) how to interpolate to value within triangle
    - (discuss now)



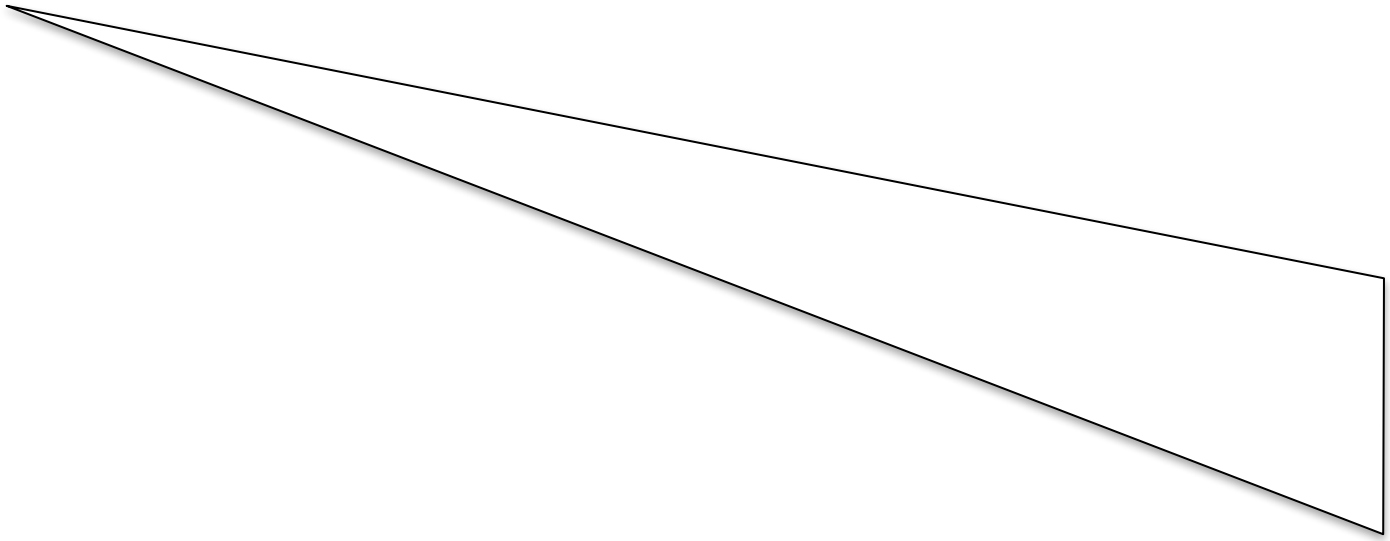
Idea #1

- More bilinear interpolation

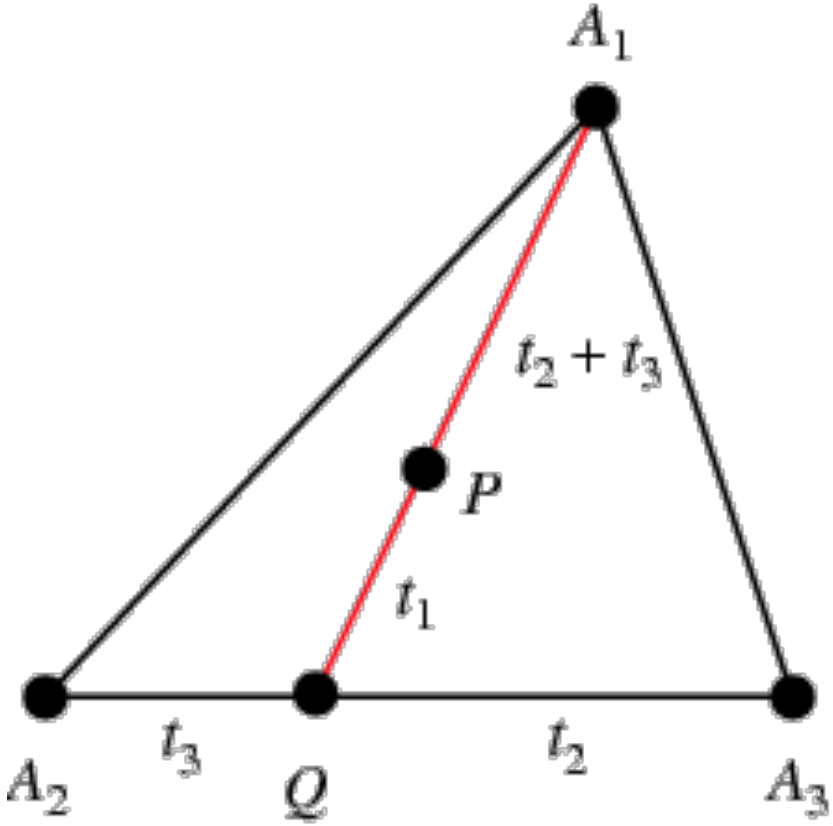


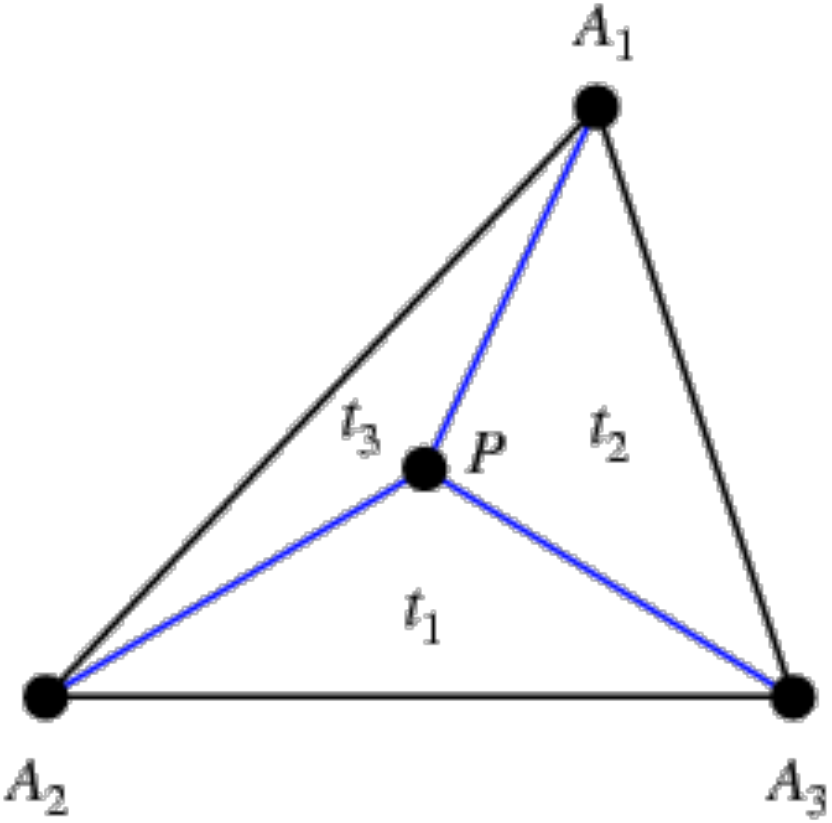
Idea #1 (cont'd)

- Different triangle, similar idea…



Idea #2: Barycentric Coordinates





V(P) = V(A1)*t1+V(A2)*t2+V(A3)*t3) / (t1+t2+t3)

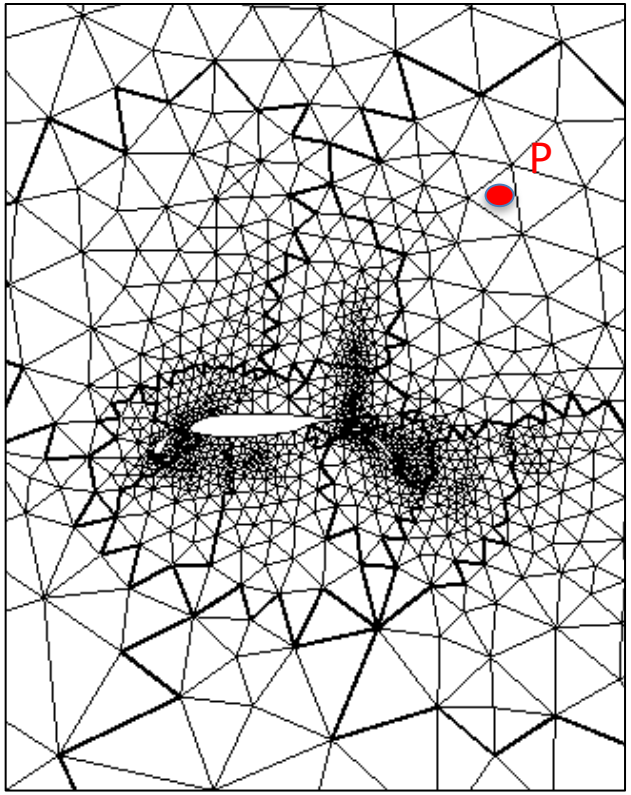
http://mathworld.wolfram.com/BarycentricCoordinates.html

Outline

- Interpolation
- Location
- Images
- Color Maps

Cell location

- Problem definition: you have a physical location (P). You want to identify which cell contains P.



It is easy to identify the cell that contains P with our eyes.
But more work to instruct a computer to do it.
What are your thoughts?

Cell location idea #1 (bad)

- Iterate over every cell
- Check if each cell contains P

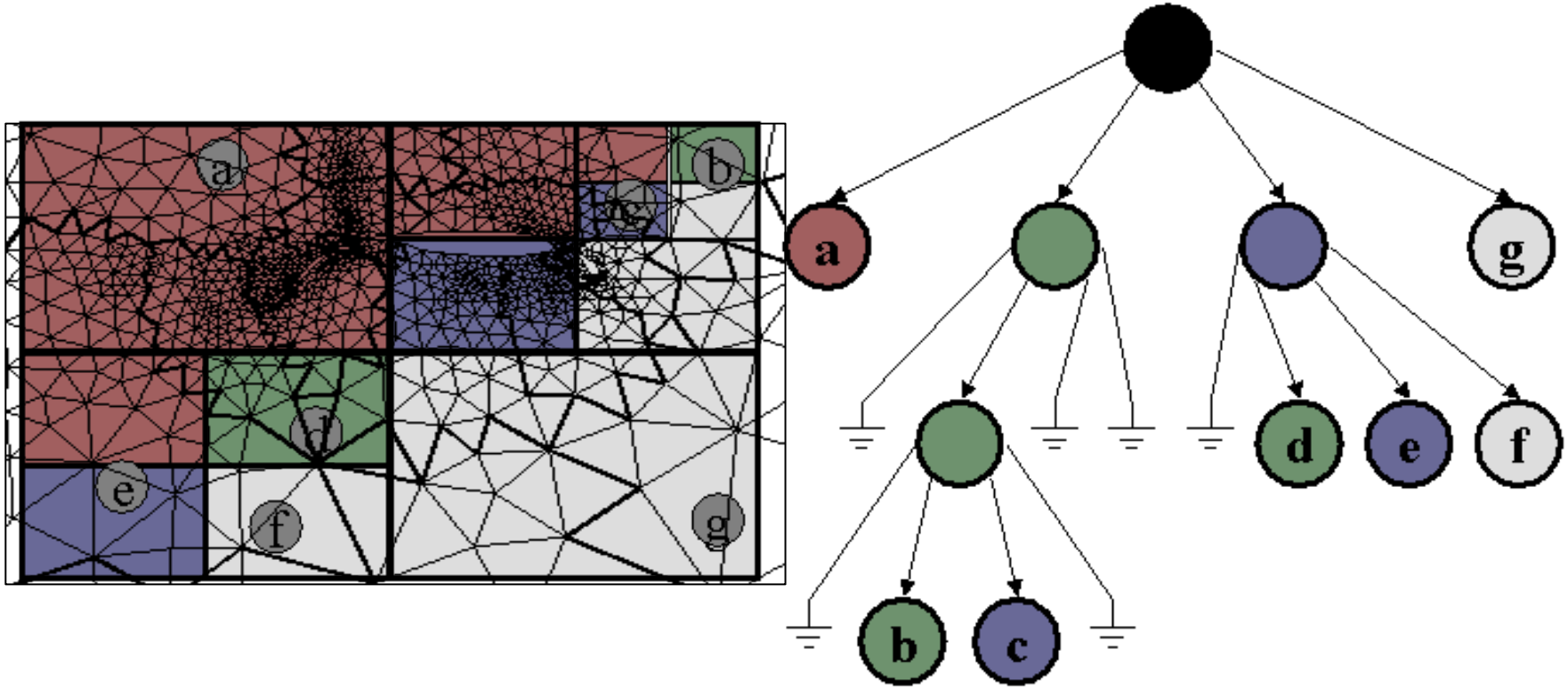
- Setup time: zero
- Search time: O(N), where N is the number of cells
- Search time for M queries: O(M*N)

Cell location idea #2 (good)

- Build “quadtree” data structure
  - (see next slide)
- Takes time to build, but then search is cheap

- Setup time: O(N log N), where N is the number of cells
- Search time: O(log N)
- Search time for M queries:
  - O(M*logN) + O(N log N)

Cell location idea #2 (good)



Credit: courses.cs.washington.edu

Comparing ideas

- Bad idea: O(M*N)
- Good idea:
  - O(N*logN) + O(M*logN)
  - = O((N+M)*logN)
- “Bad idea” is actually better if M very small

Project 2: Field Evaluation

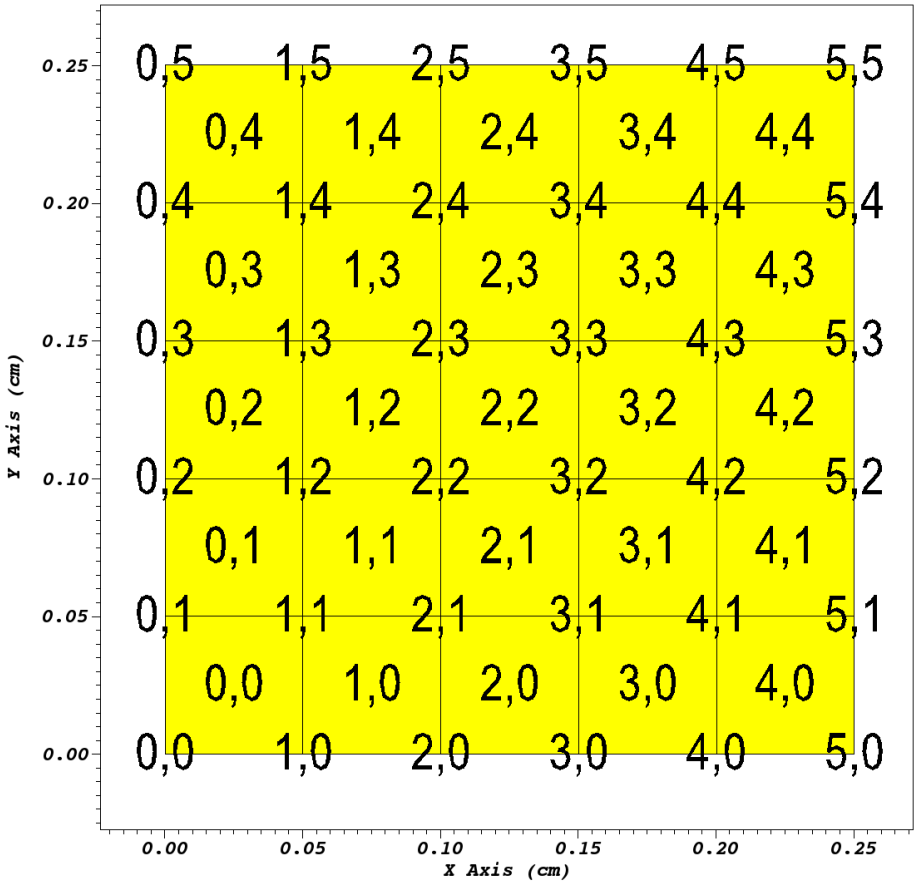
- Assigned today, prompt online
- Due Sun Jan 16 midnight (→ 6am January 17th)
- Worth 5% of your grade
- I provide:
  - Code skeleton online
  - Correct answers provided
- What you upload to Canvas? ... your source code
- Note: Project 3 coming on Thursday

Project 2: Field Evaluation

- Basic idea: for point P, find F(P)
- Important: will be on 2D rectilinear meshes, so cell location is easier
- Strategy in a nut shell:
  - Find cell C that contains P
  - Find C's 4 vertices, V0, V1, V2, and V3
  - Find F(V0), F(V1), F(V2), and F(V3)
  - Find locations of V0, V1, V2, and V3
  - Perform bilinear interpolation to location P

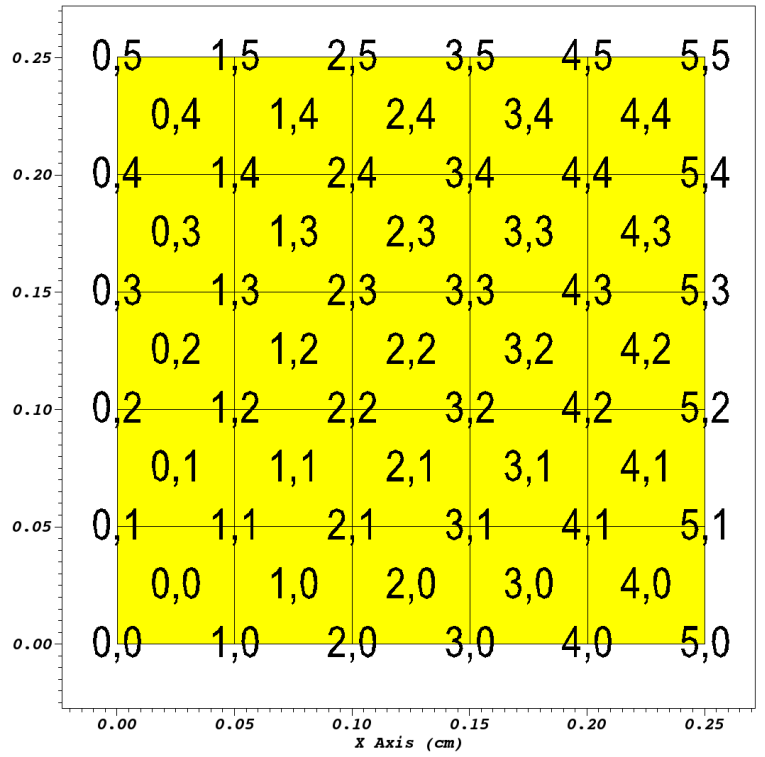
Cell location for project 2

- Traverse X and Y arrays and find the logical cell index
  - X={0, 0.05, 0.1, 0.15, 0.2, 0.25}
  - Y={0, 0.05, 0.1, 0.15, 0.2, 0.25}
- (Quiz) what cell contains (0.17,0.08)? = (3,1)



Facts about cell (3,1)

- It's cell index is 8
- It contains points (3,1), (4,1), (3,2), and (4,2)
- Facts about point (3,1):
  - Its location is (X[3], Y[1])
  - Its point index is 9
  - Its scalar value is F(9)
- Similar facts for other points
- → we have enough info to do bilinear interpolation



Big O For Cell Location in Project #2

- You should exploit the properties of rectilinear grid
  - Search along X coordinates for X-position
  - Search along Y coordinates for Y-position
- Setup time: none
- Search time: O(sqrt(N)) + O(sqrt(N)) = O(sqrt(N))
  - Note: N is total number of points, i.e., dims[0]*dims[1]

Example:

- X = { 0, 1, 3, 5, 8 };
- Y = { -1, 1, 3, 7, 9 };
- Which cell contains (2.5, 8.5)?
- Answer: cell with logical indices (1, 3)
- Because
  - X[1] < 2.5 < X[2] ← so it logical index for x is 1
  - Y[3] < 8.5 < Y[4] ← so it logical index for x is 3
- Don’t believe me? Draw a picture…

What's in the code skeleton

- Implementations for:
  - GetNumberOfPoints
  - GetNumberOfCells
  - GetPointIndex
  - GetCellIndex
  - GetLogicalPointIndex
  - GetLogicalCellIndex

} Our bijective function

  - "main": set up mesh, call functions, create output

What's not in the code skeleton

```
// pt: a two-dimensional location
// dims: an array of size two.
//          The first number is the size of the array in argument X,
//          the second the size of Y.
// X: an array (size is specified by dims).
//          This contains the X locations of a rectilinear mesh.
// Y: an array (size is specified by dims).
//          This contains the Y locations of a rectilinear mesh.
// F: a scalar field defined on the mesh.  Its size is dims[0]*dims[1].
float
EvaluationFunction(const float *pt, const int *dims,
                   const float *X, const float *Y, const float *F)
{
    return 0; // IMPLEMENT ME!!
}
```

… and a few other functions you need to implement

Outline

- Interpolation
- Location
- 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)

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

- Interpolation
- Location
- Images
- Color Maps

Pseudocolor plot

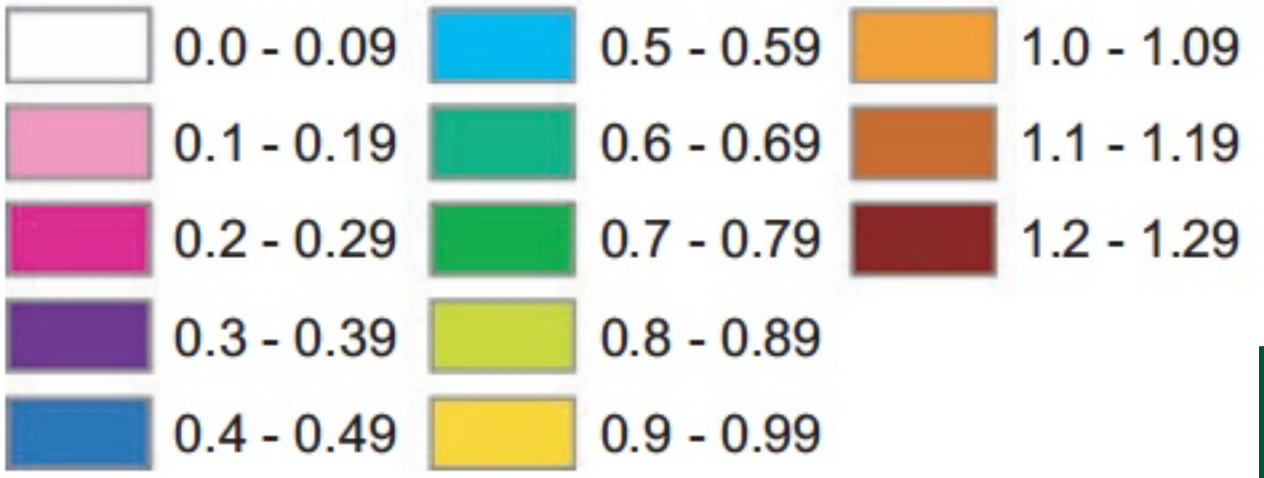
- First visualization technique we will learn
- Idea: take a scalar field on a mesh and transform it to colors
- Two elements:
  - Define a map
  - Apply the map

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

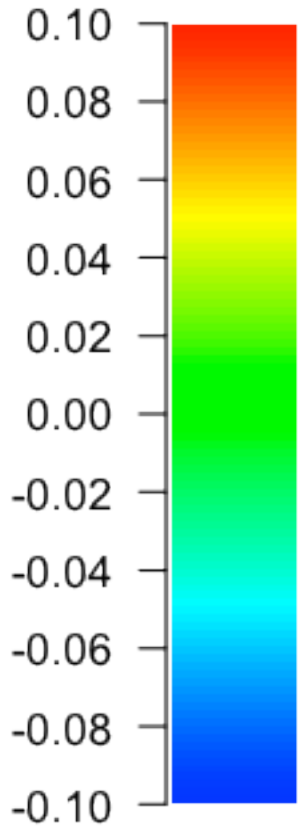
Color map example (“discrete”)

- 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)



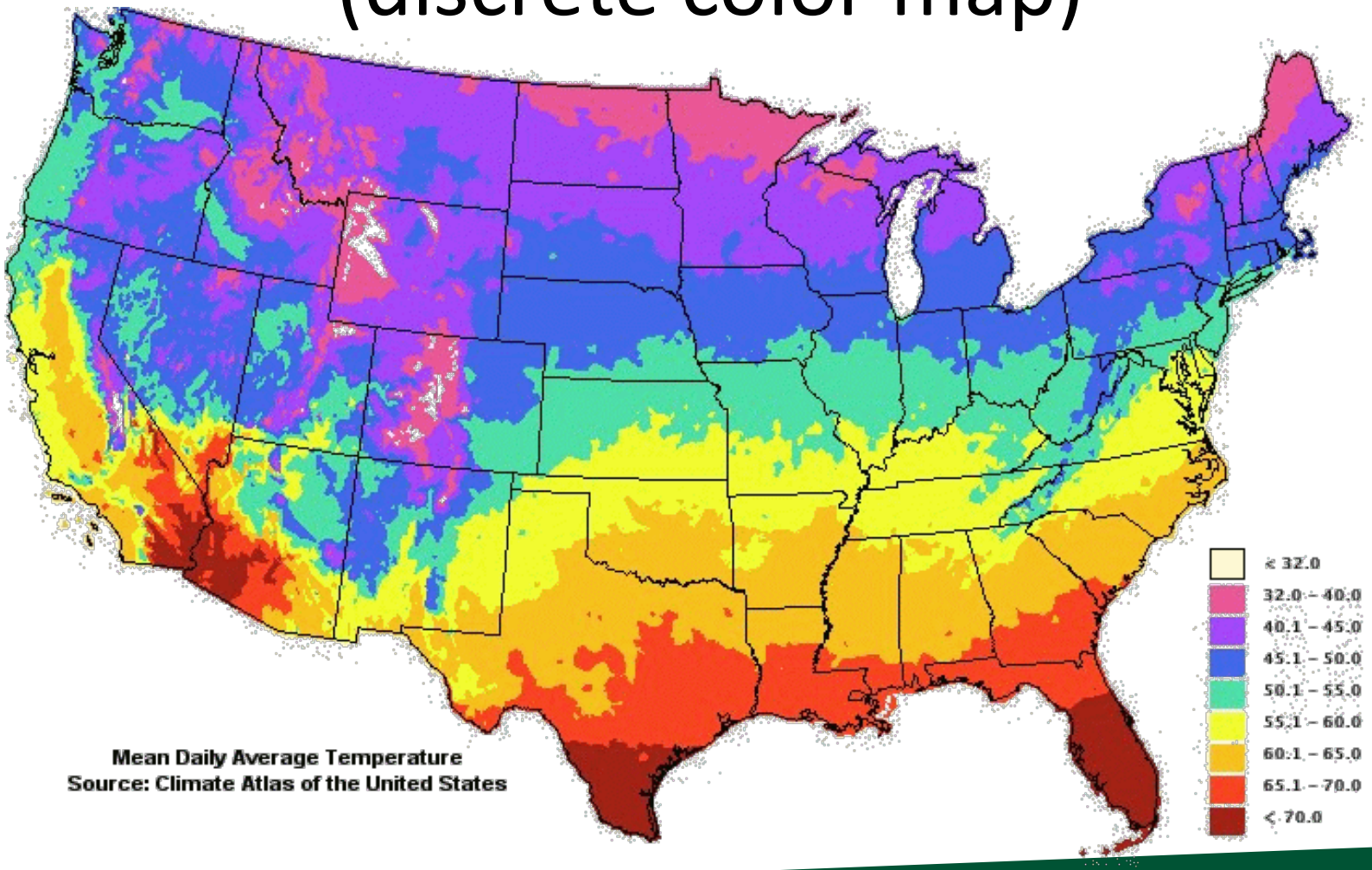
Pseudocolor plot

- 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

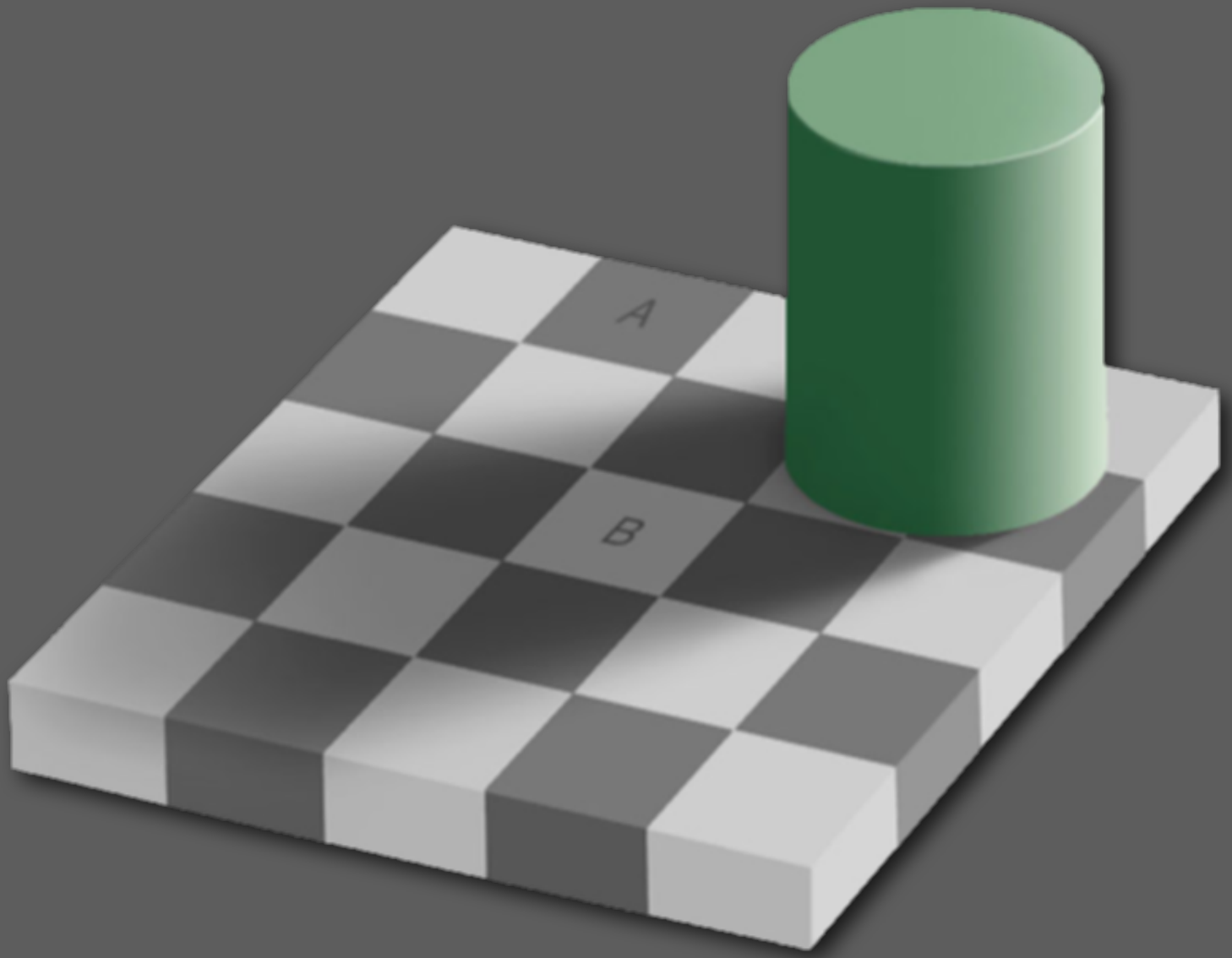
Pseudocolor NOT in practice

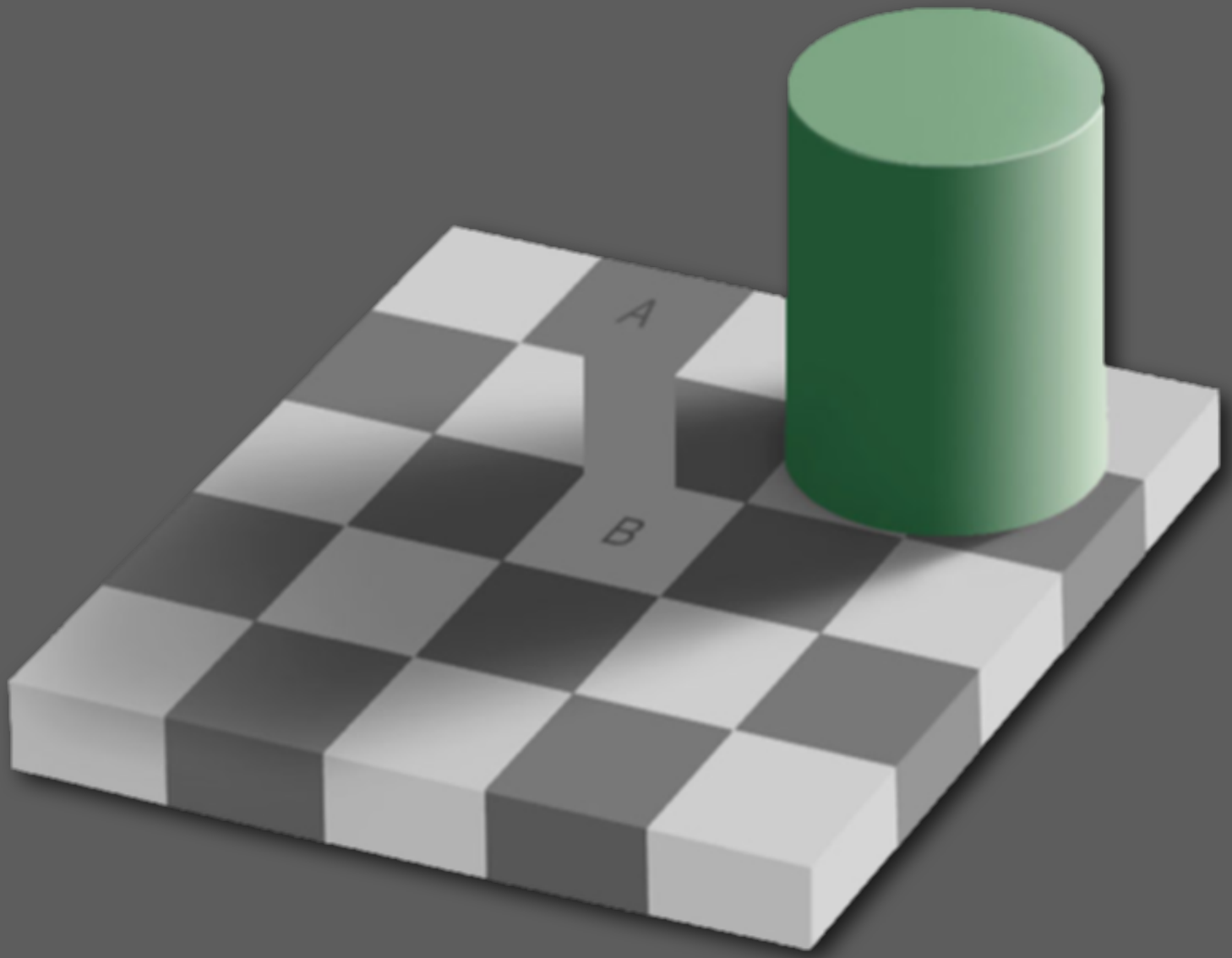
- An unusual way:
  - Assume data perfectly overlaps with image
    - Bottom left of data in bottom left pixel
    - Upper right of data in upper right pixel
  - For each pixel
    - Find corresponding spatial location
    - Evaluate scalar field
    - Apply color map
    - Assign pixel
- In effect: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

“Today’s Lecturer”: Kristi Potter

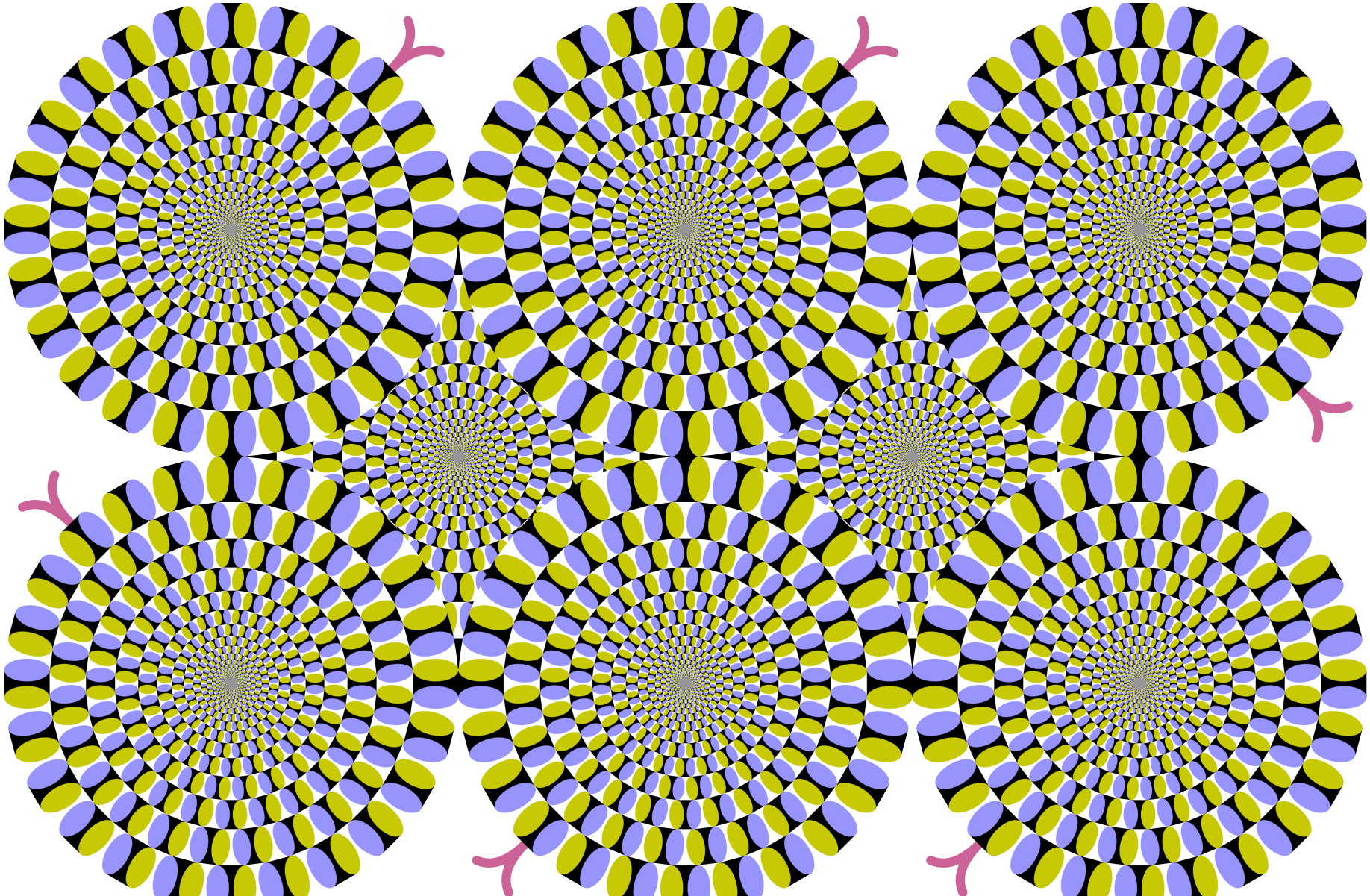
Kristin Potter is a senior scientist specializing in data visualization at the National Renewable Energy Lab’s Insight Center. Her research is focused on methods for improving visualization techniques by adding qualitative information regarding reliability and variability to the data display. This work 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

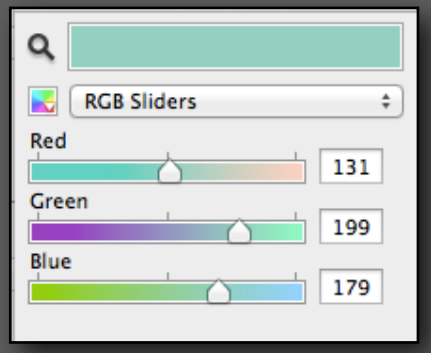

A
B

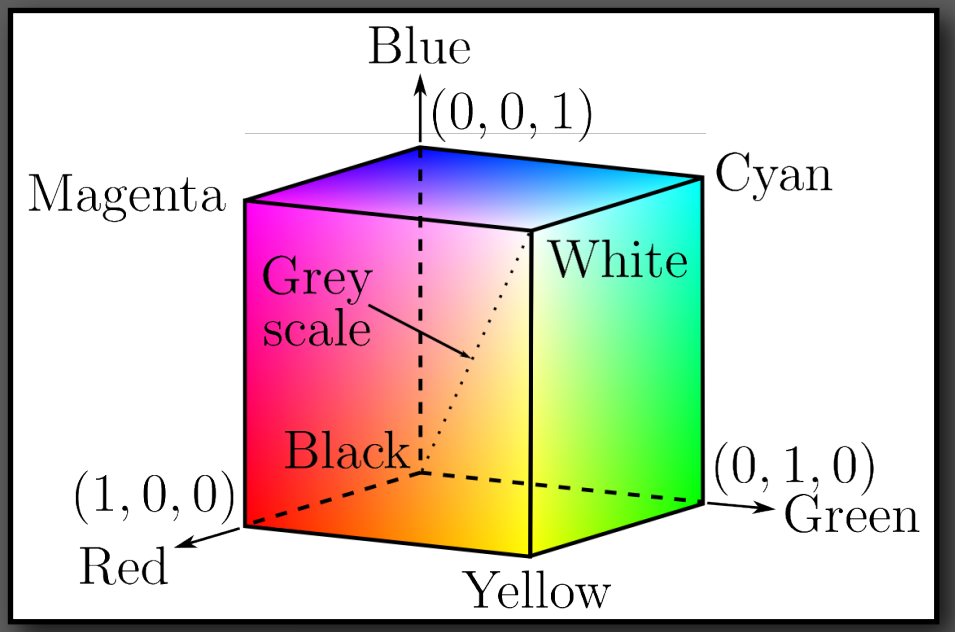

A
B

Here's the better one



Color Spaces

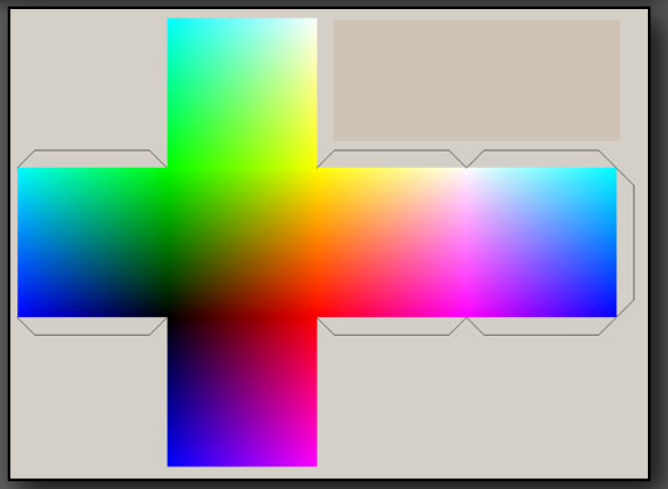


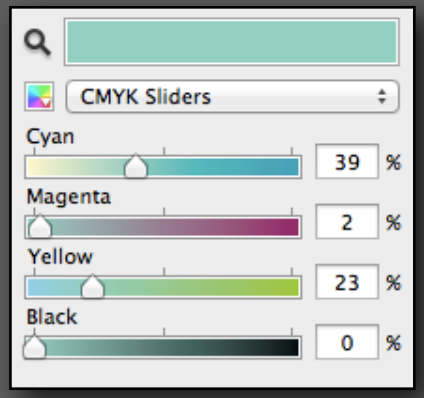


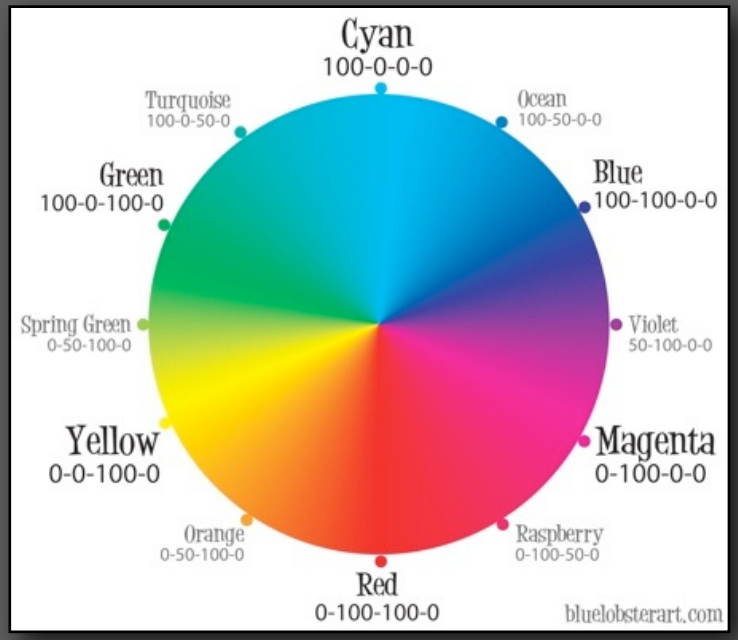
RGB

Red, Green, Blue channels

- electron guns of crt monitors



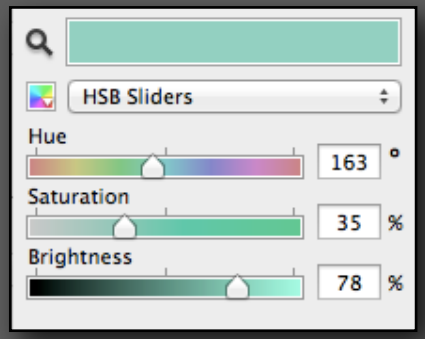


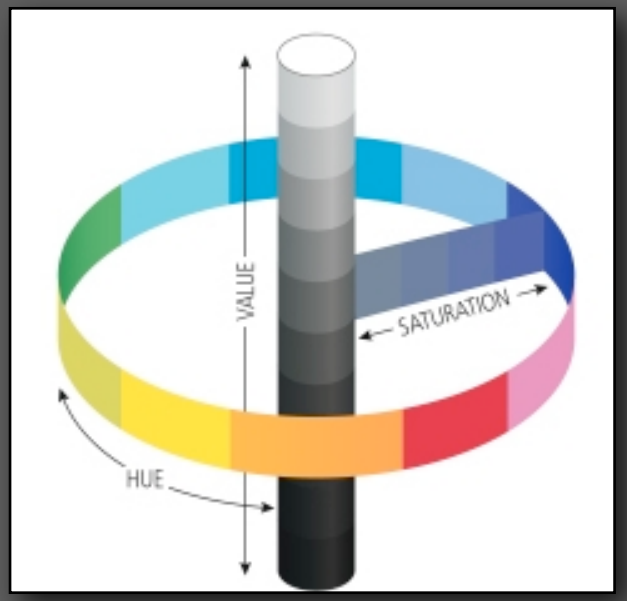


CMYK

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

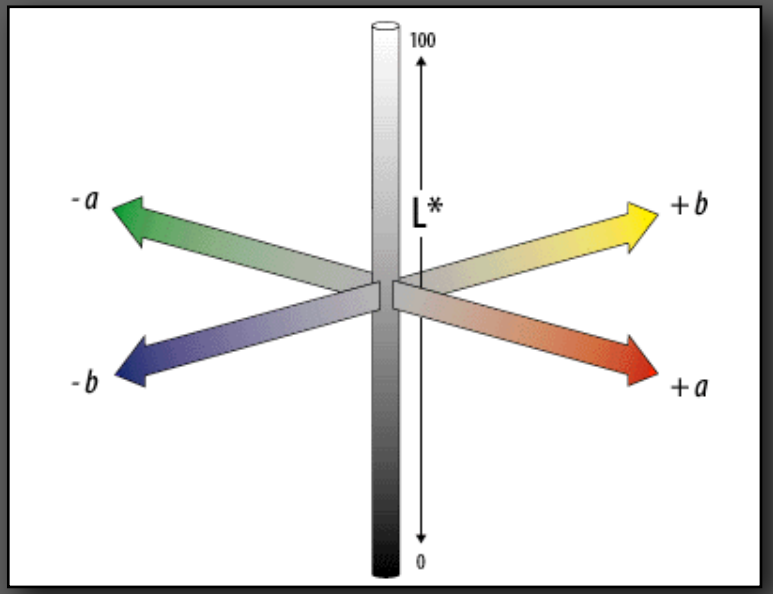




HSV [B, L, I]

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

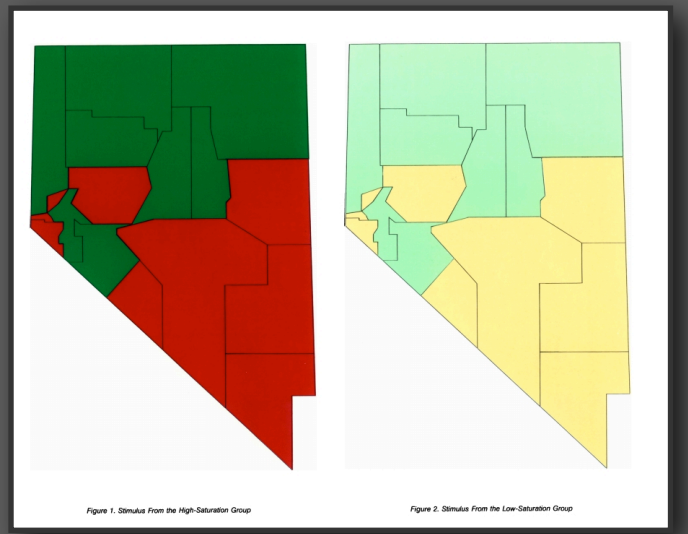
(short list)

Misused Color

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

-Edward Tufte

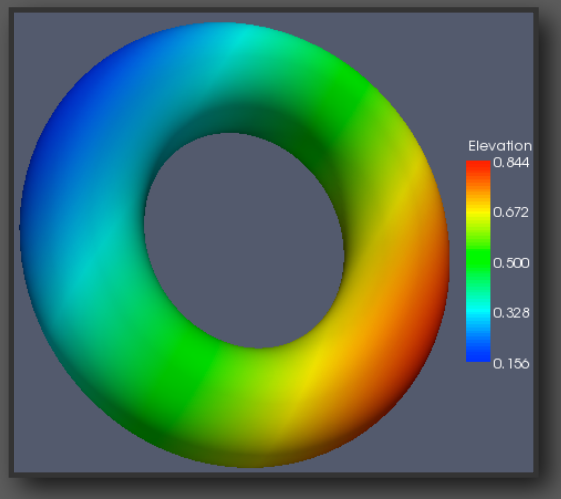
- bias a reader's perception
- cause the wrong information stand out
- hide meaningful information
- cause visual clutter and overload



Color Coding

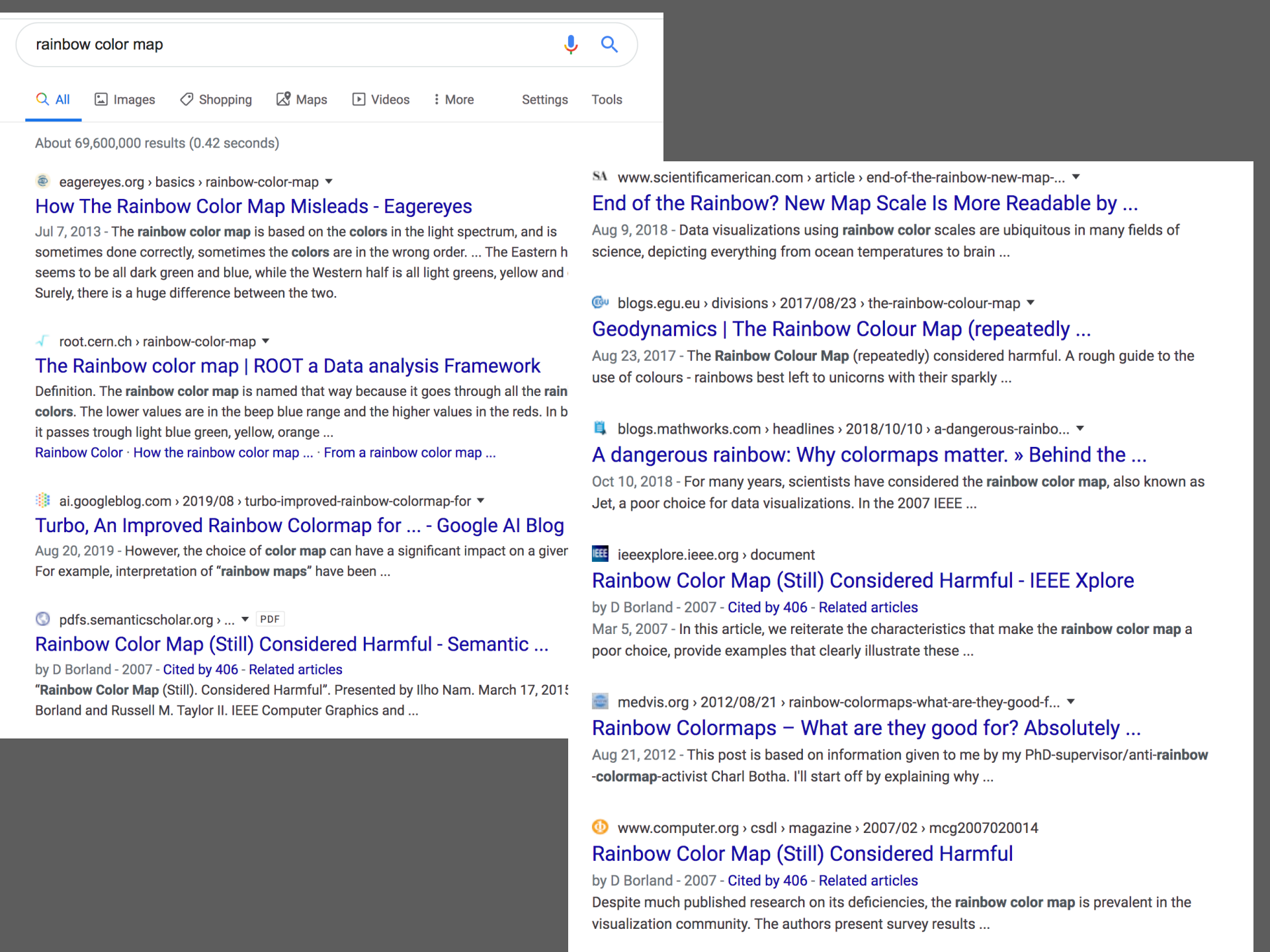
- hard to pick discernible colors
- can only get ~12 distinguishable colors
- object size, line width can influence perception

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.

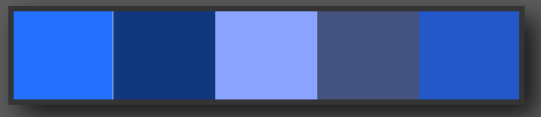
Non-data components should be displayed just visibly enough to perform their role.

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)

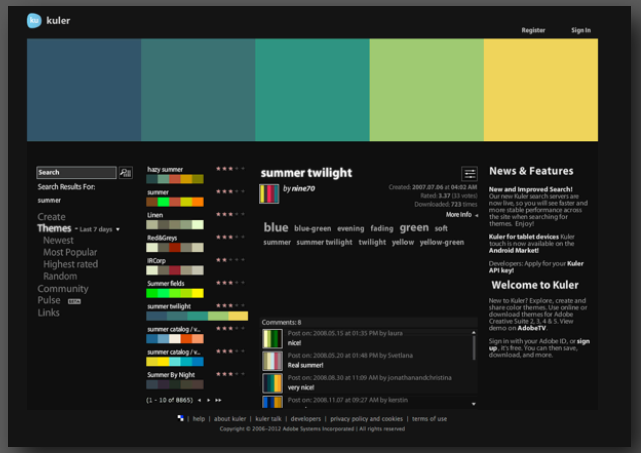
size, shape, texture, length, width,
orientation, curvature and intensity

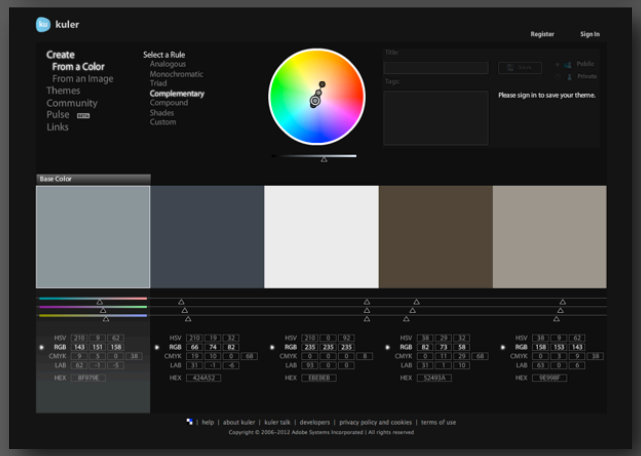
Tools for Color Scheme Creation

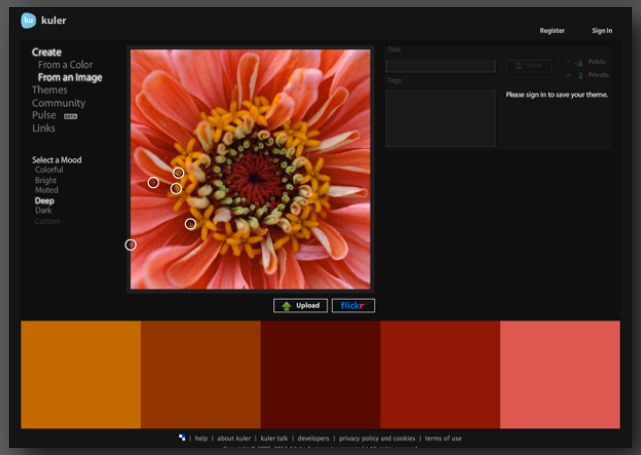
Kuhler

http://kuler.adobe.com

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



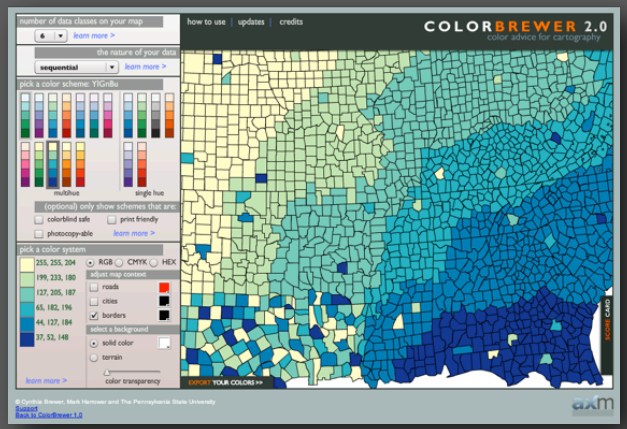


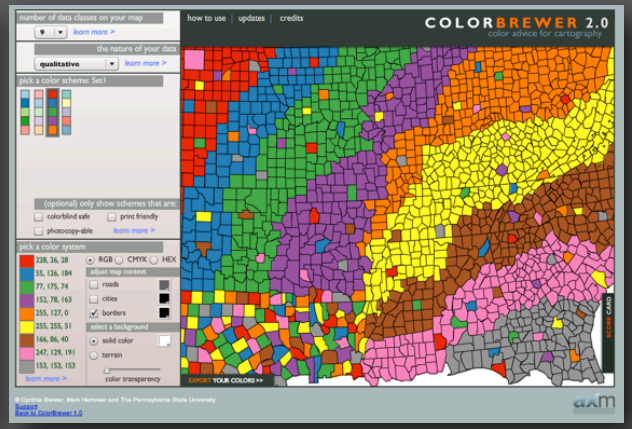


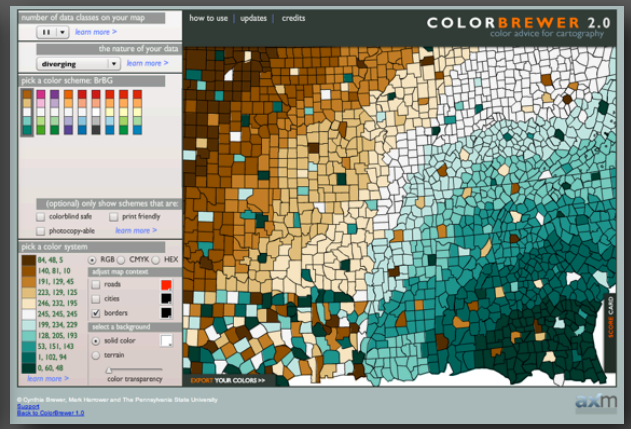
Color Scheme Designer

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

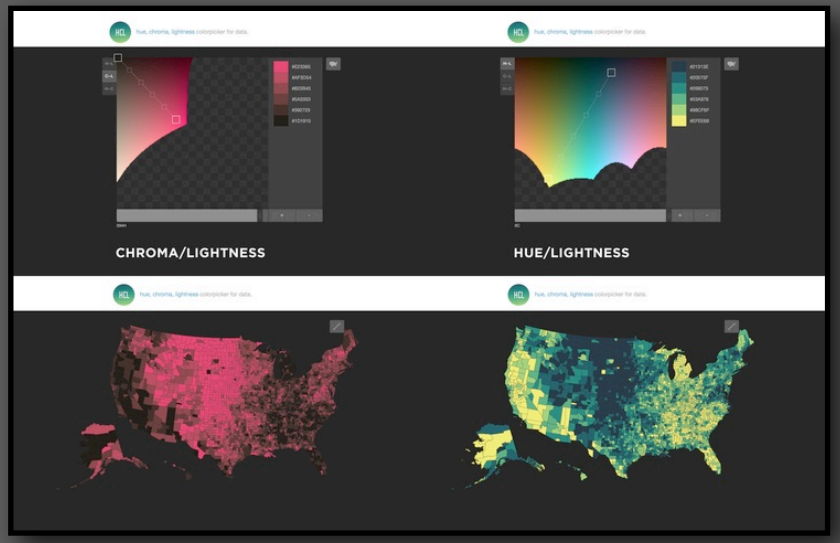
optimized for data ordered low to high

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 by tonight
- Due ???
- 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