



CS 4300

Computer Graphics

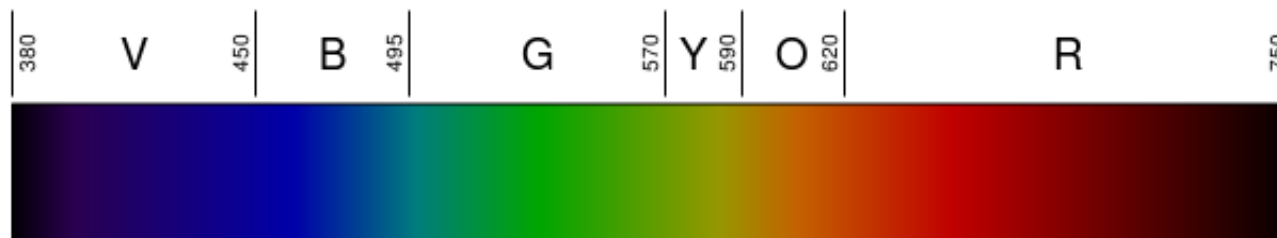
Prof. Harriet Fell
Fall 2012

Lecture 4 – September 12, 2012



What is color?

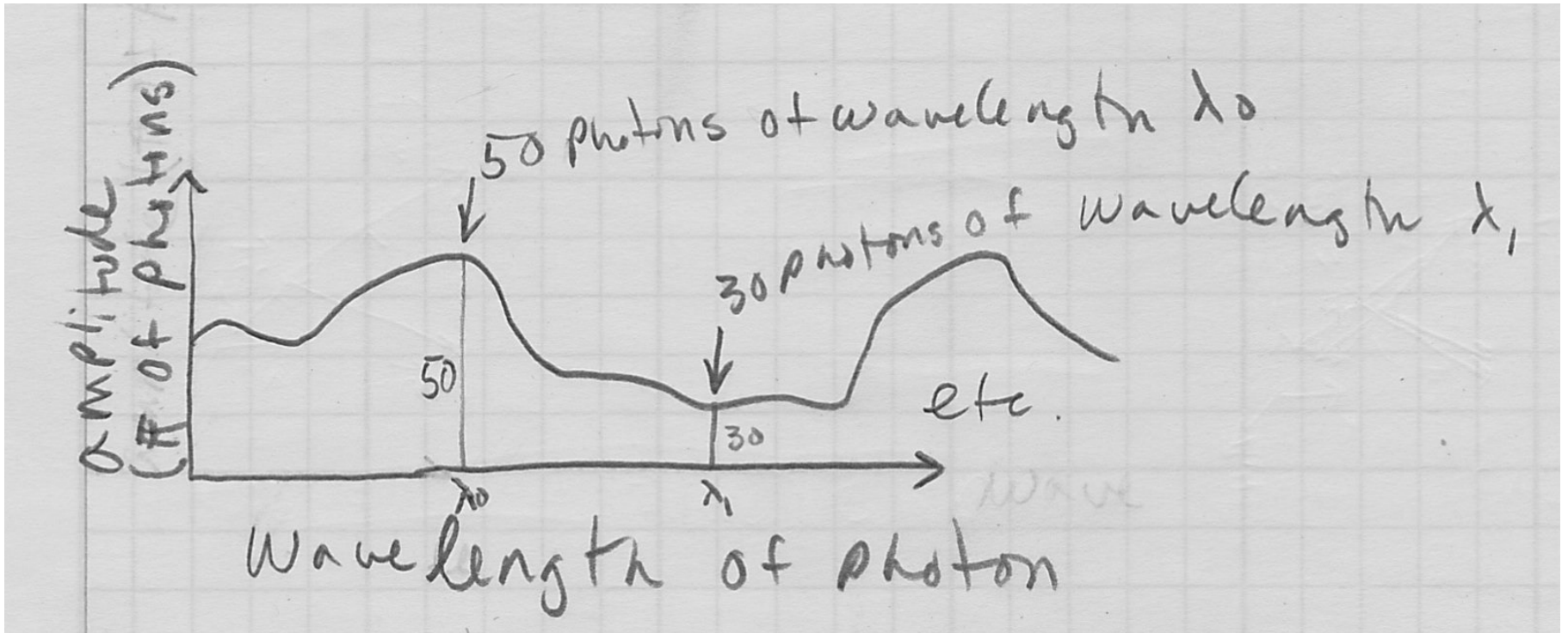
- from physics, we know that the *wavelength of a photon (typically measured in nanometers, or billionths of a meter) determines its apparent color*
- *we cannot see all wavelengths, but only the visible spectrum from around 380 to 750 nm*





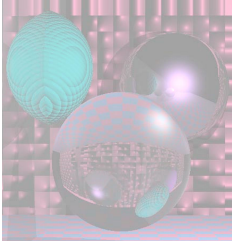
Where are the other colors?

- but where are the following colors: “brown”, “pink”, “white”, ...?
- clearly, the *color spectrum does not actually contain all colors; some colors are non-spectral*
- *generally, a large number of photons with different wavelengths are simultaneously impinging on any given location of your retina*



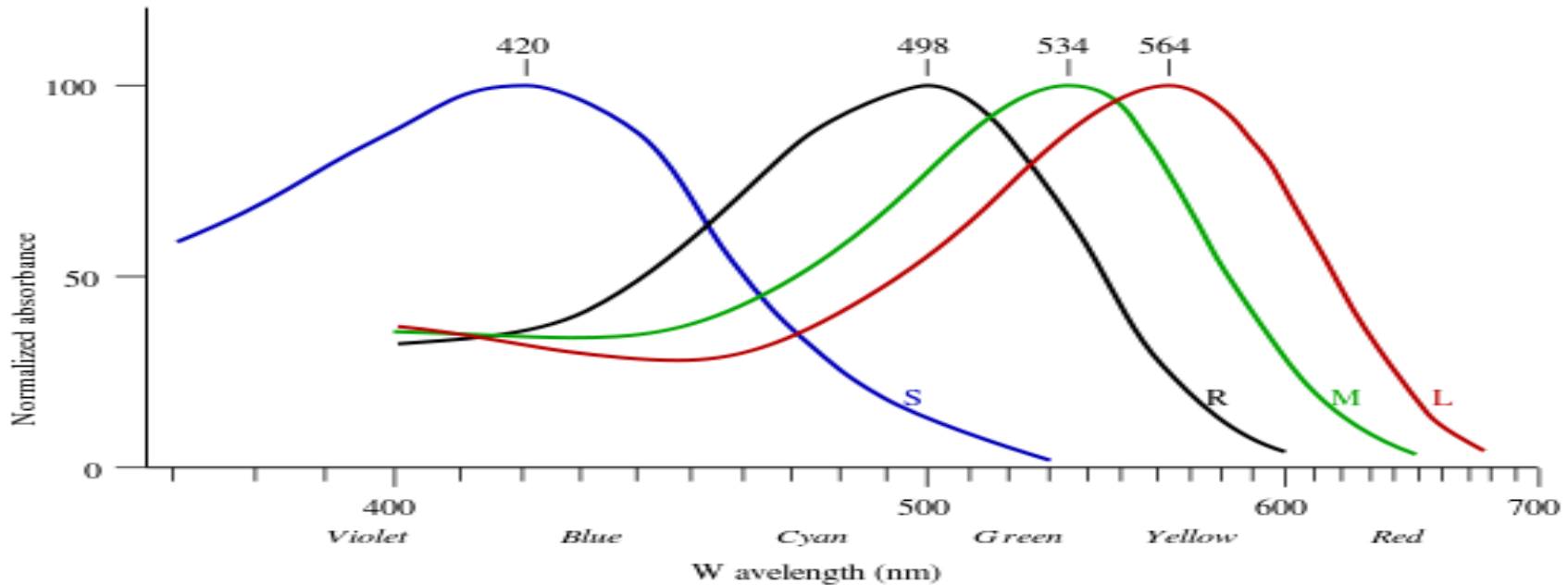
Marty Vona's sketch

- the actual incident light is not of a single wavelength, but can be described by a spectral histogram
- the histogram represents the relative quantity of photons of each wavelength



Human Perception of Color

- the human eye cannot determine the exact histogram
- in fact just representing a complete spectral histogram exactly would require an infinite amount of space because it's a continuous quantity
- the biological solution is another form of sampling
- three types of cone cells respond (with the equivalent of a single number each) to the degree to which the actual incident histogram is similar to response histograms with peaks near red, green, and blue



- so the original continuous histogram impinging on one location of your retina is reduced to three measurements
- (actually, there is a fourth *rod cell type*, which is mainly active in low light conditions)
- *color blindness is typically caused by anomalies in the types of cone cells*
- *other animals also have different cone cells*

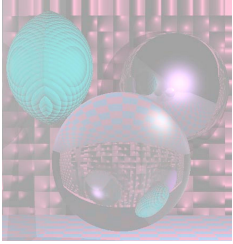
- because we have converted a continuous object into a set of discrete samples, we have to consider *aliasing*
 - *different incident histograms, called metamers, may be mapped to the same set of cone cell responses*
 - *how many distinct colors can be seen?*
 - *one way to think about it is to know that each cone cell type can distinguish between about 100 intensity levels of the associated response curve, and then to take a constructive approach*
 - *there are ~1M ways to combine cone cell responses, so an average human can distinguish roughly that many colors*
- the biology of human cone cells is the not only the reason we often use RGB to represent color; in fact, it defines color. Color is not an intrinsic property of light, but rather a result of the interaction between human cone cells and histograms of incident light.







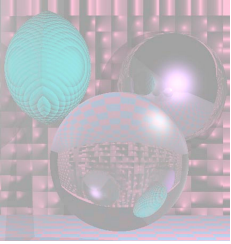




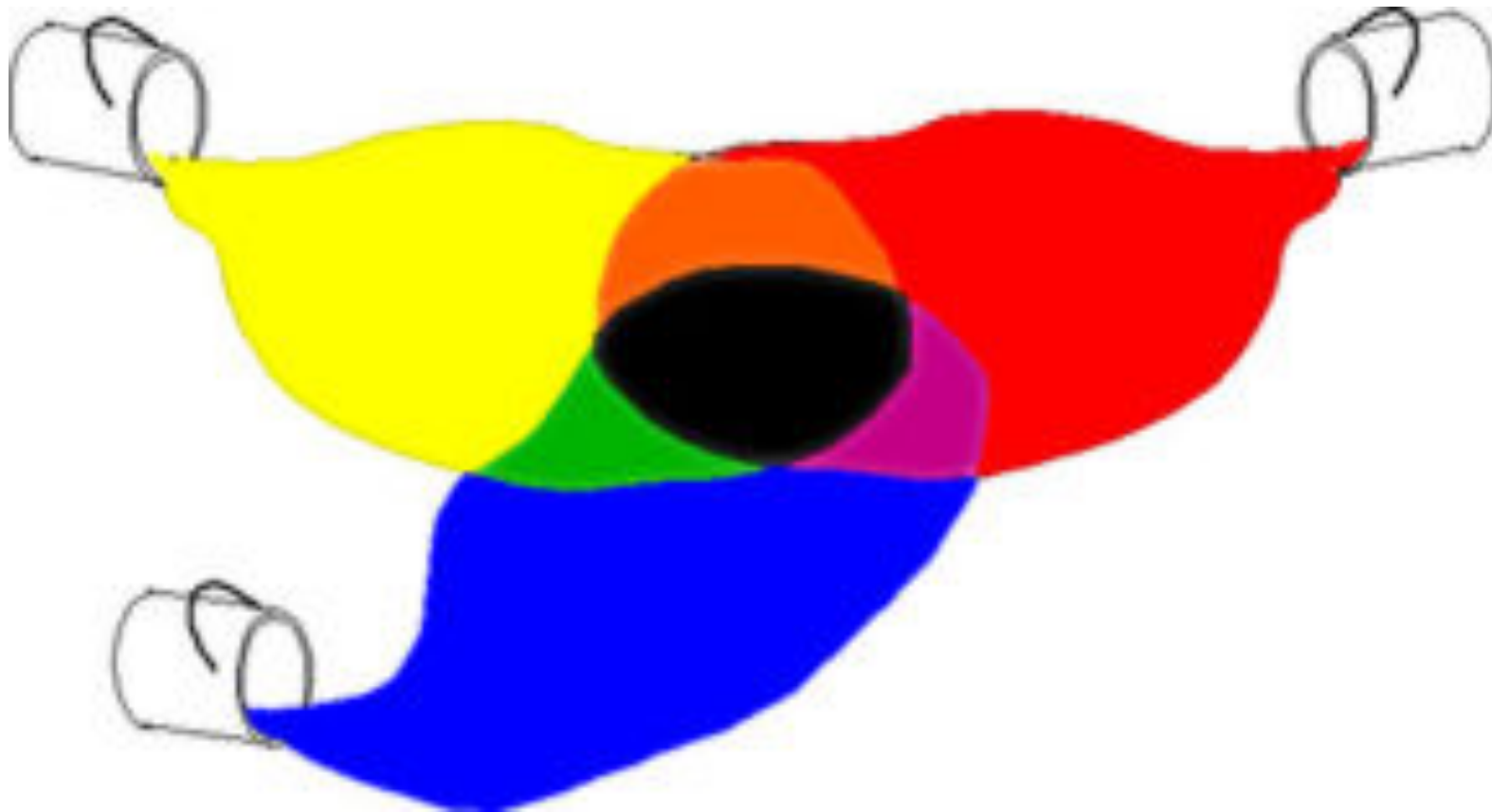
From the Hubble

[Hubble Site Link](#)





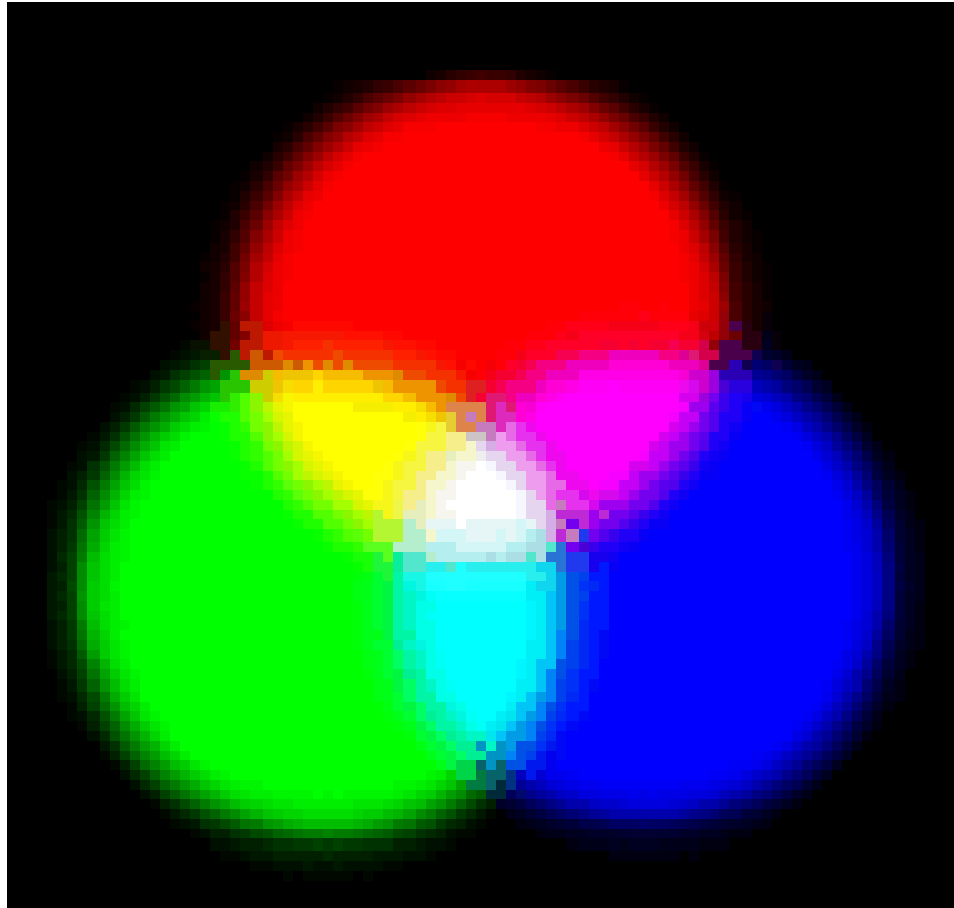
Color



www.thestagecrew.com

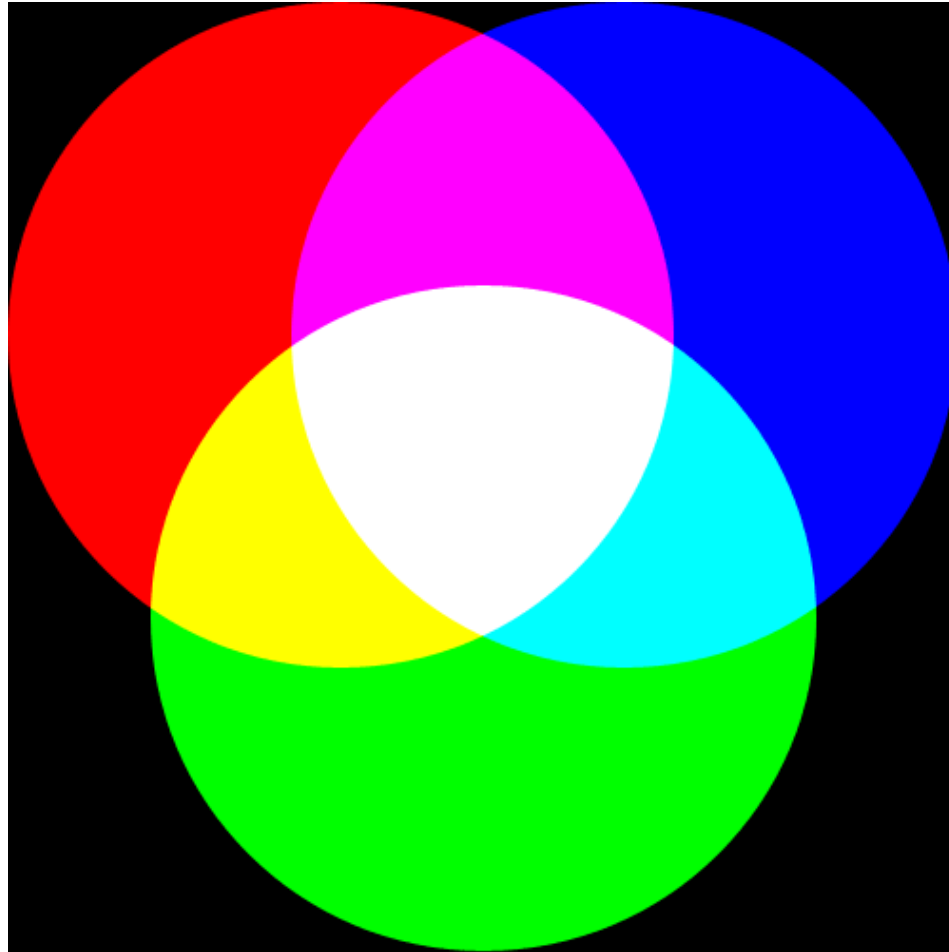


Red, Green, and Blue Light





Adding R, G, and B Values

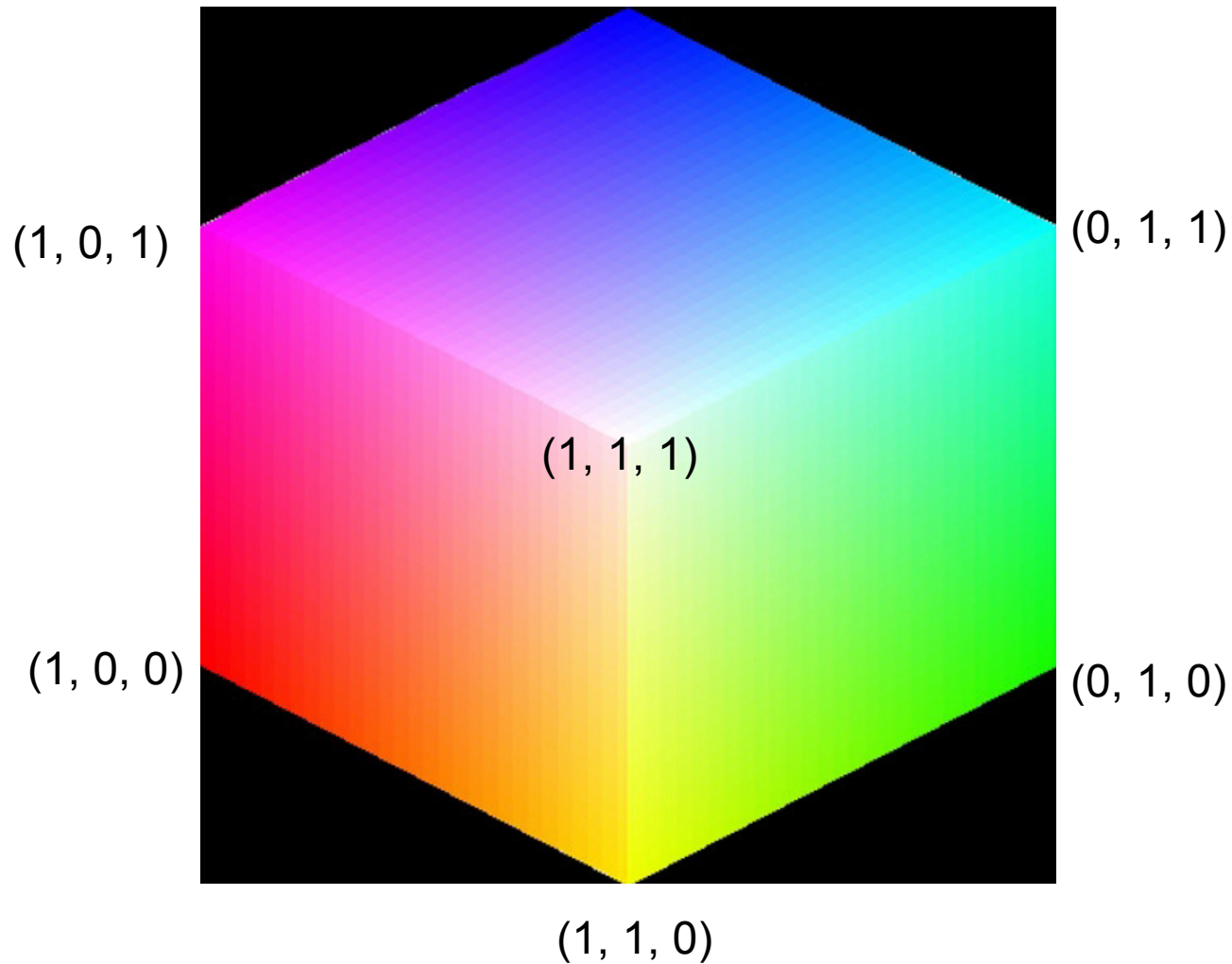


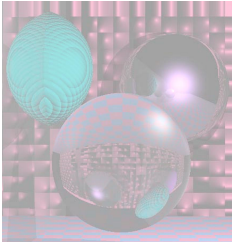
<http://en.wikipedia.org/wiki/RGB>



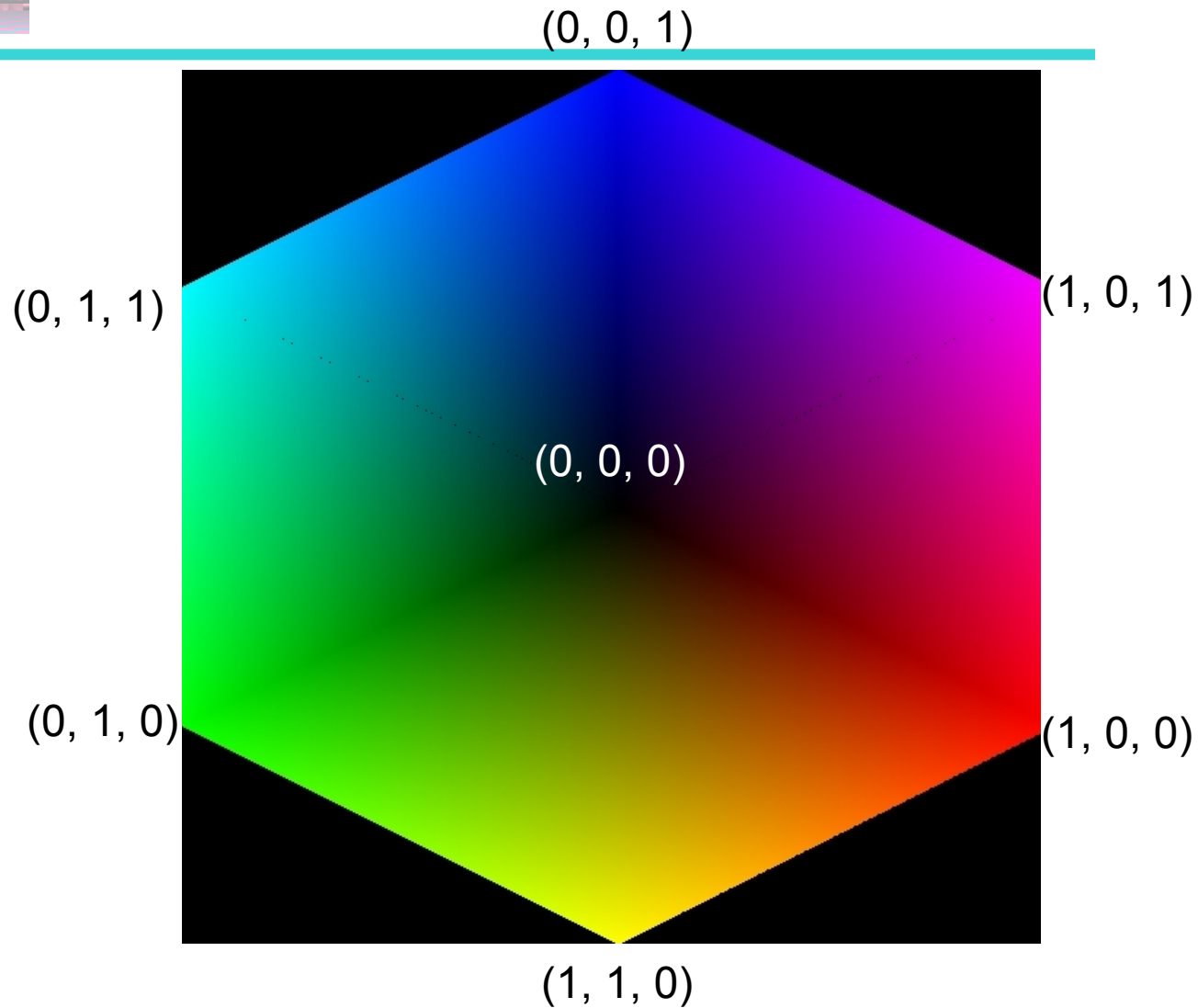
RGB Color Cube

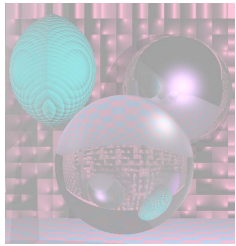
$(0, 0, 1)$





RGB Color Cube The Dark Side





Doug Jacobson's RGB Hex Triplet Color Chart

RGB Hex Triplet Color Chart
E-mail-ware...What a concept!

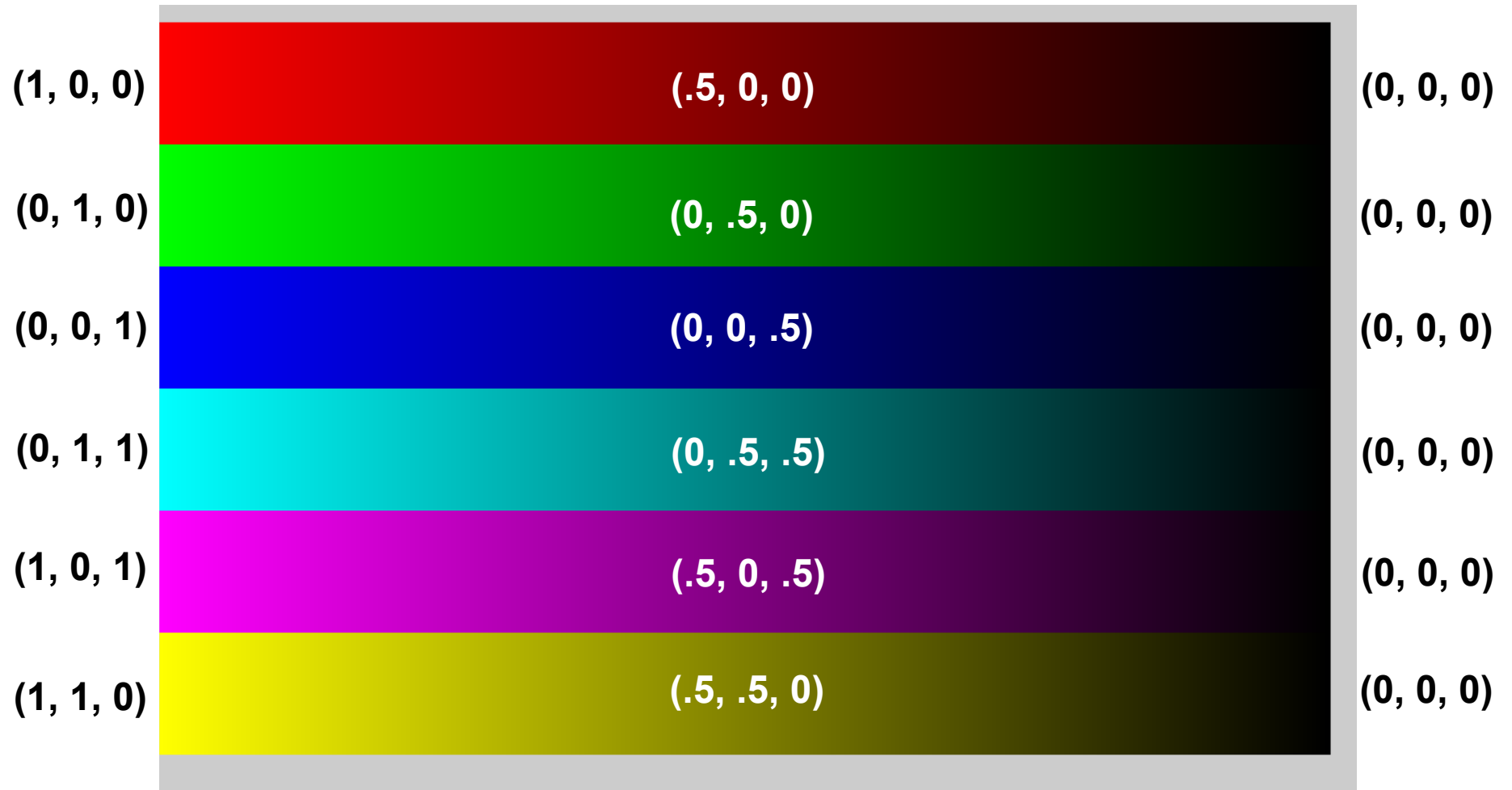
*If you find this chart helpful, send mail to Doug and say "Thanks!".
 jacobson@phoenix.net*

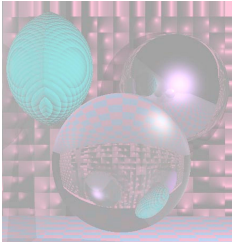
	FFFFFF	FFCCFF	FF99FF	FF66FF	FF33FF	FF00FF	
EEEEEE	FFFFCC	FFCCCC	FF99CC	FF66CC	FF33CC	FF00CC	
DDDDDD	FFFF99	FFCC99	FF9999	FF6699	FF3399	FF0099	
CCCCCC	FFFF66	FFCC66	FF9966	FF6666	FF3366	FF0066	00FF00
BBBBBB	FFFF33	FFCC33	FF9933	FF6633	FF3333	FF0033	00EE00
AAAAAA	FFFF00	FFCC00	FF9900	FF6600	FF3300	FF0000	00DD00
999999	CCFFFF	CCCCFF	CC99FF	CC66FF	CC33FF	CC00FF	00CC00
888888	CCFFCC	CCFFCC	CC99CC	CC66CC	CC33CC	CC00CC	00BB00
777777	CCFF99	CCFF99	CC9999	CC6699	CC3399	CC0099	00AA00
666666	CCFF66	CCFF66	CC9966	CC6666	CC3366	CC0066	009900
555555	CCFF33	CCFF33	CC9933	CC6633	CC3333	CC0033	008800
444444	CCFF00	CCFF00	CC9900	CC6600	CC3300	CC0000	007700
333333	99FFFF	99CCFF	9999FF	9966FF	9933FF	9900FF	006600
222222	99FFCC	99CCCC	9999CC	9966CC	9933CC	9900CC	005500
111111	99FF99	99CC99	999999	996699	993399	990099	004400
000000	99FF66	99CC66	999966	996666	993366	990066	003300
FF0000	99FF33	99CC33	999933	996633	993333	990033	002200
EE0000	99FF00	99CC00	999900	996600	993300	990000	001100
DD0000	66FFFF	66CCFF	6699FF	6666FF	6633FF	6600FF	0000FF
CC0000	66FFCC	66CCCC	6699CC	6666CC	6633CC	6600CC	0000EE
BB0000	66FF99	66CC99	669999	666699	663399	660099	0000DD
AA0000	66FF66	66CC66	669966	666666	663366	660066	0000CC
990000	66FF33	66CC33	669933	666633	663333	660033	0000BB
880000	66FF00	66CC00	669900	666600	663300	660000	0000AA
770000	33FFFF	33CCFF	3399FF	3366FF	3333FF	3300FF	000099
660000	33FFCC	33CCCC	3399CC	3366CC	3333CC	3300CC	000088
550000	33FF99	33CC99	339999	336699	333399	330099	000077
440000	33FF66	33CC66	339966	336666	333366	330066	000066
330000	33FF33	33CC33	339933	336633	333333	330033	000055
220000	33FF00	33CC00	339900	336600	333300	330000	000044
110000	00FFFF	00CCFF	0099FF	0066FF	0033FF	0000FF	000033
	00FFCC	00CCCC	0099CC	0066CC	0033CC	0000CC	000022
	00FF99	00CC99	009999	006699	003399	000099	000011
	00FF66	00CC66	009966	006666	003366	000066	
	00FF33	00CC33	009933	006633	003333	000033	
	00FF00	00CC00	009900	006600	003300	000000	

Copyright © 1995 Douglas R. Jacobson
All Rights Reserved



Making Colors Darker



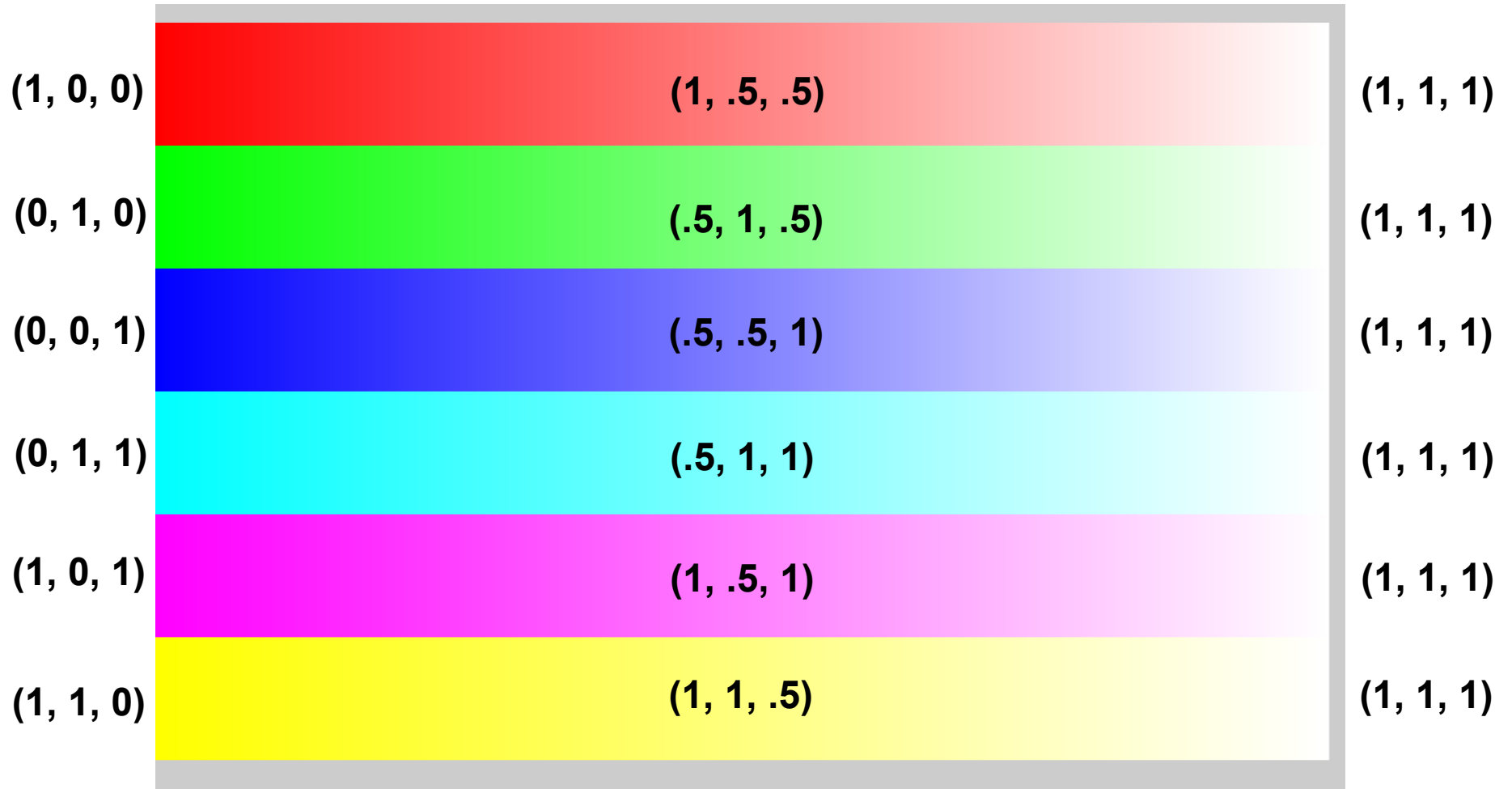


Getting Darker, Left to Right

```
for (int b = 255; b >= 0; b--){  
    c = new Color(b, 0, 0); g.setPaint(c);  
    g.fillRect(800+3*(255-b), 50, 3, 150);  
    c = new Color(0, b, 0); g.setPaint(c);  
    g.fillRect(800+3*(255-b), 200, 3, 150);  
    c = new Color(0, 0, b); g.setPaint(c);  
    g.fillRect(800+3*(255-b), 350, 3, 150);  
    c = new Color(0, b, b); g.setPaint(c);  
    g.fillRect(800+3*(255-b), 500, 3, 150);  
    c = new Color(b, 0, b); g.setPaint(c);  
    g.fillRect(800+3*(255-b), 650, 3, 150);  
    c = new Color(b, b, 0); g.setPaint(c);  
    g.fillRect(800+3*(255-b), 800, 3, 150);  
}
```



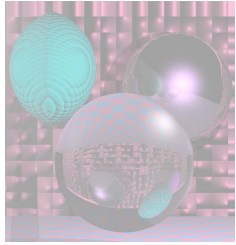
Making Pale Colors





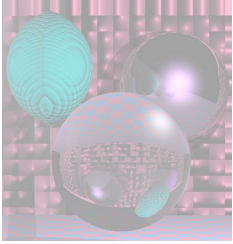
Getting Paler, Left to Right

```
for (int w = 0; w < 256; w++){  
    c = new Color(255, w, w); g.setPaint(c);  
    g.fillRect(3*w, 50, 3, 150);  
    c = new Color(w, 255, w); g.setPaint(c);  
    g.fillRect(3*w, 200, 3, 150);  
    c = new Color(w, w, 255); g.setPaint(c);  
    g.fillRect(3*w, 350, 3, 150);  
    c = new Color(w, 255, 255); g.setPaint(c);  
    g.fillRect(3*w, 500, 3, 150);  
    c = new Color(255,w, 255); g.setPaint(c);  
    g.fillRect(3*w, 650, 3, 150);  
    c = new Color(255, 255, w); g.setPaint(c);  
    g.fillRect(3*w, 800, 3, 150);  
}
```



Additive and Subtractive Color Space

- sometimes RGB are considered “additive” colors because they form a basis for the color space relative to black
- CMY can similarly be considered “subtractive” colors because, effectively
 - cyan+red = white
 - magenta+green = white
 - yellow+blue = white



Display vs. Print

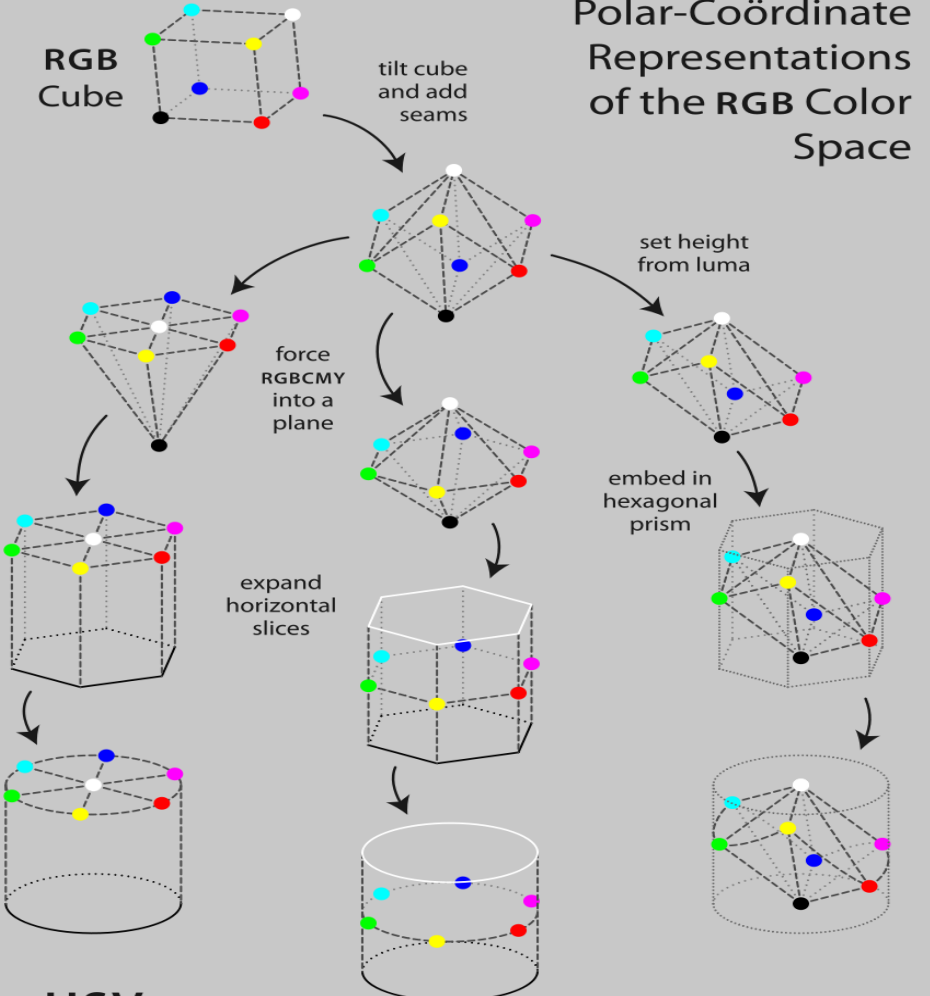
- additive colors typically used when light is generated by an output device (e.g. CRT, LCD)
- subtractive colors typically used when printing on white paper
- sometimes RGB and CMY are considered distinct color spaces



HSV Color Space

- ***hue***: the basic color, or chromaticity
- ***saturation***: how “deep” the color is (vs “pastel”)
- ***value***: the brightness of the color

Polar-Coordinate Representations of the RGB Color Space



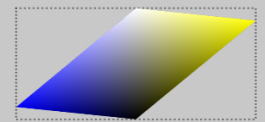
HSV
"Hexcone"
Model



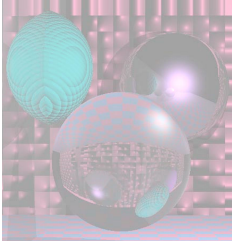
HSL
"Double Hexcone"
Model



**Luma/Chroma/
Hue Model**



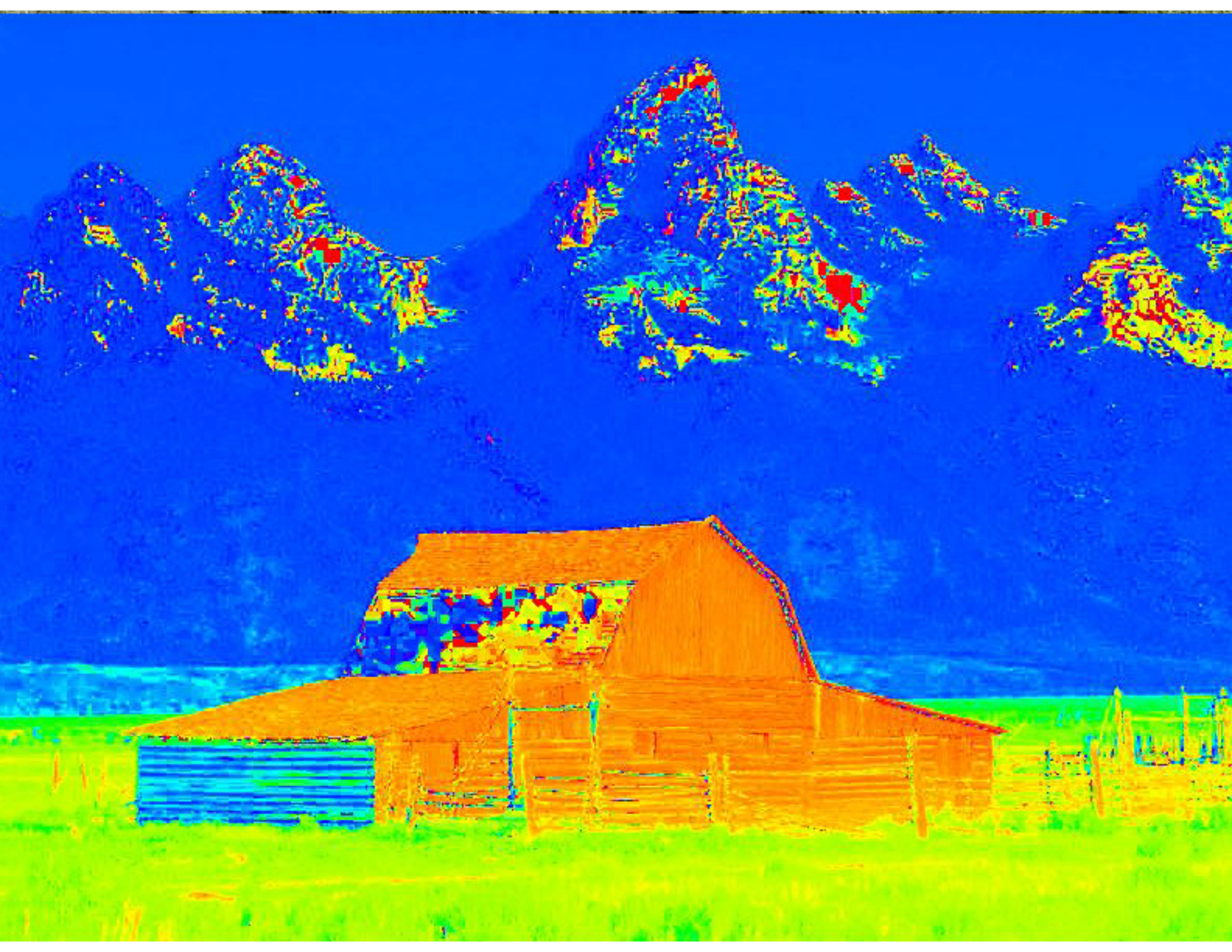
vertical cross-sections

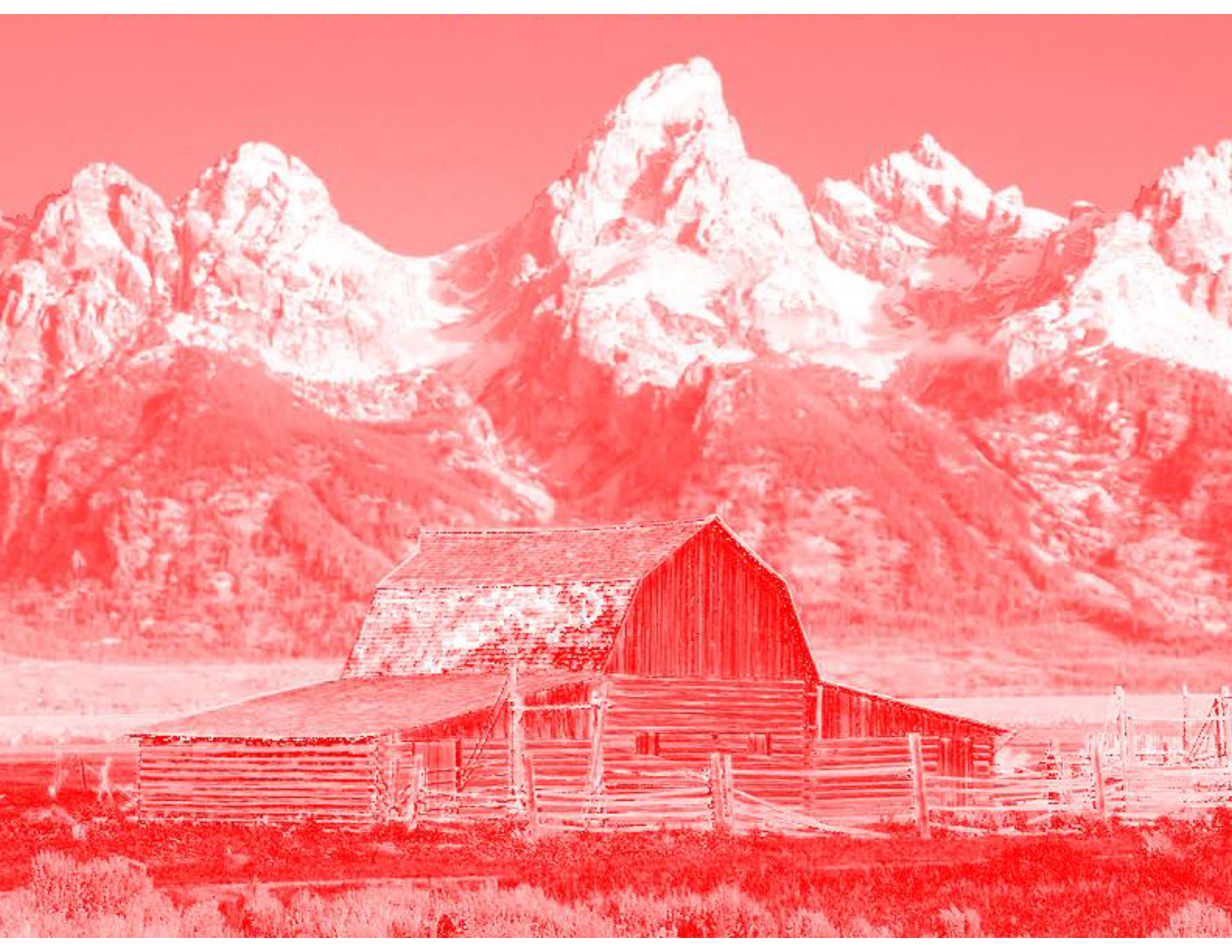


RGB to HSV

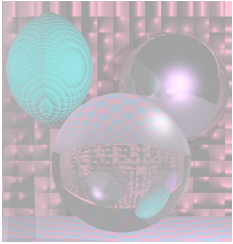
- HSV is again a 3 dimensional space, but it is typically considered to use *cylindrical coordinates*
 - *this is mainly a construction to decompose the three dimensional color space in a way that is more useful to human designers*
 - *also often useful in machine vision algorithms, which simulate our theories of (aspects of) human vision*
 - *can visualize HSV space as a “morph” of RGB space*
 - *“stretch” the white and black vertices up and down*
 - *“line up” the remaining six vertices along a common horizontal plane*
 - *for HSV, put the white vertex back onto plane*
 - *(a variation, HSL, keeps white and black symmetrically above and below)*











Try the color picker

