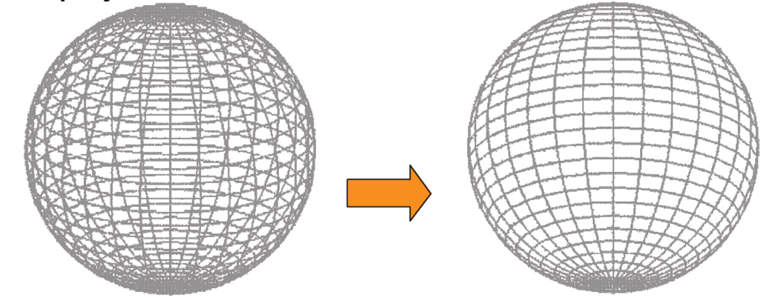
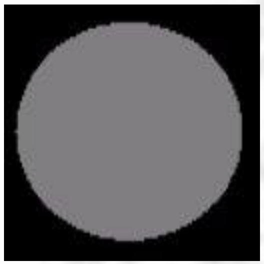
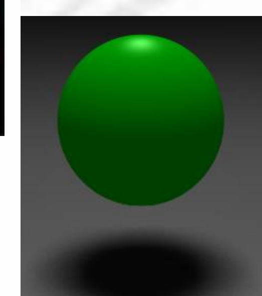
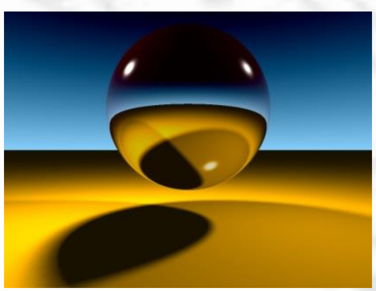
Progression in Visual Realism

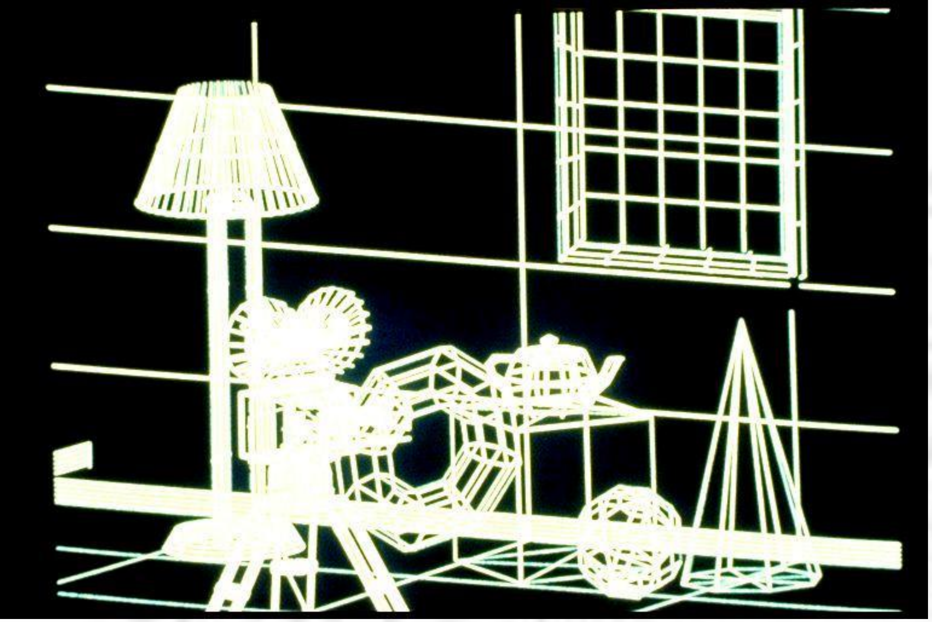


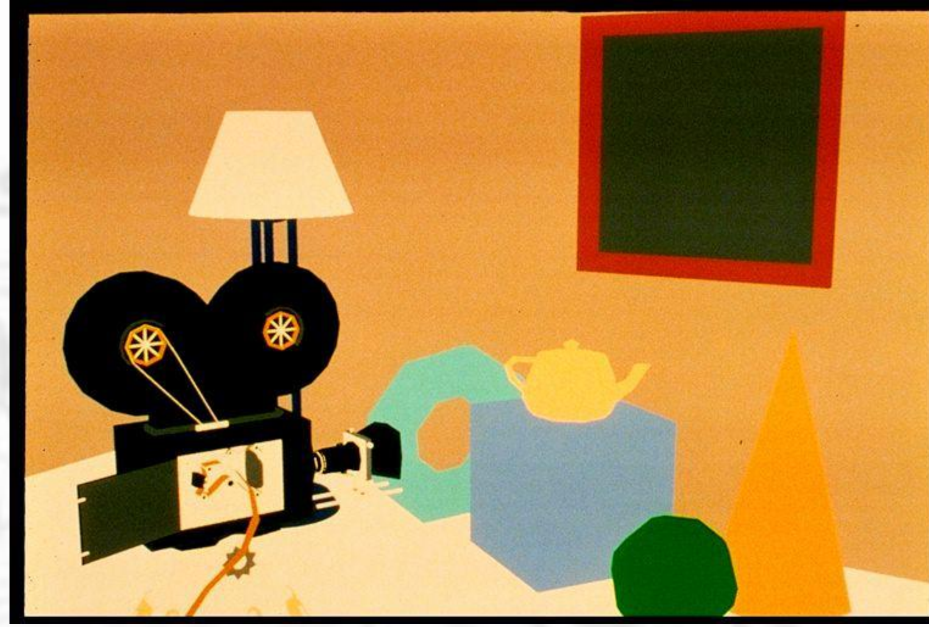


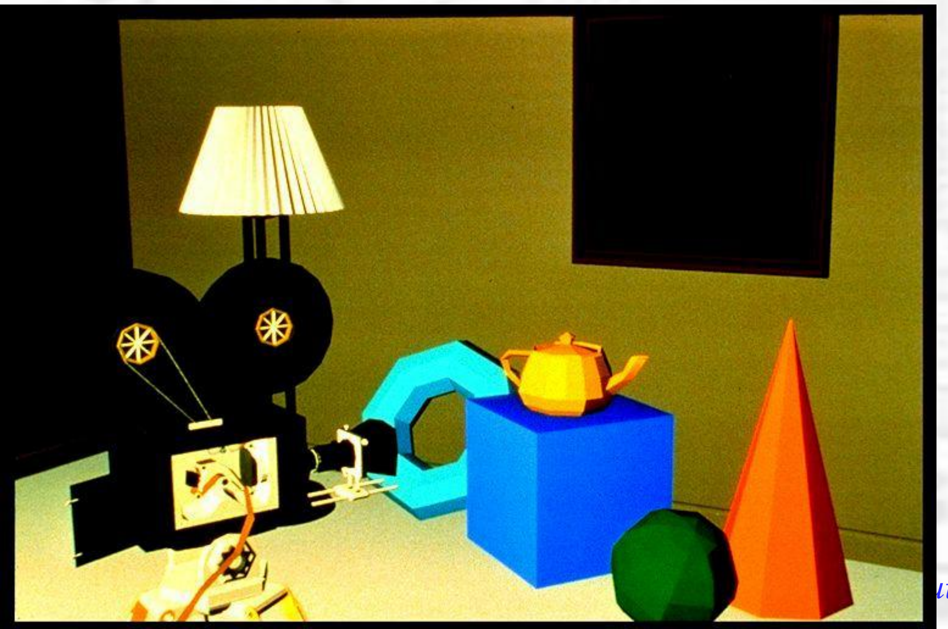
FLAT SHADING

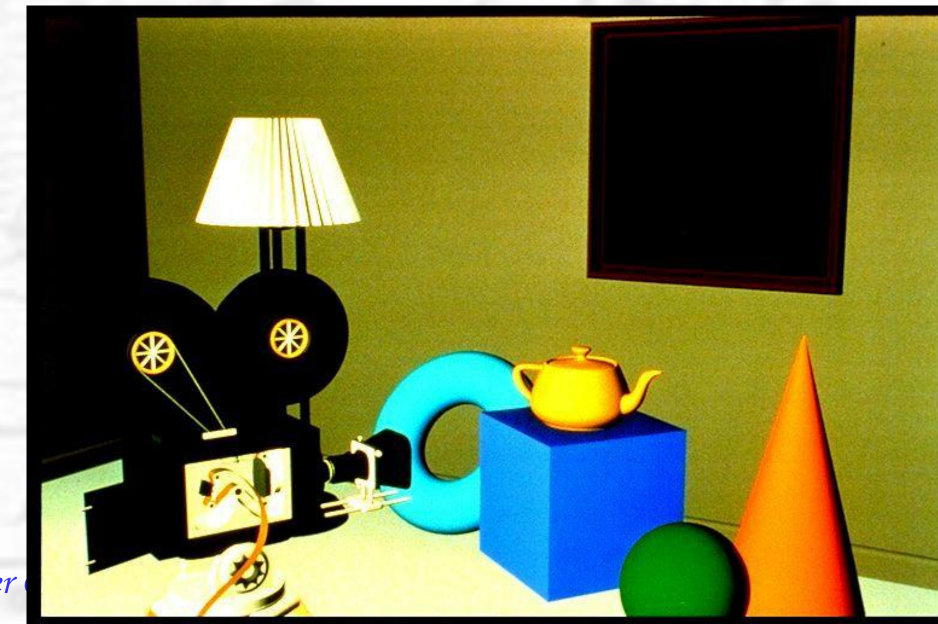


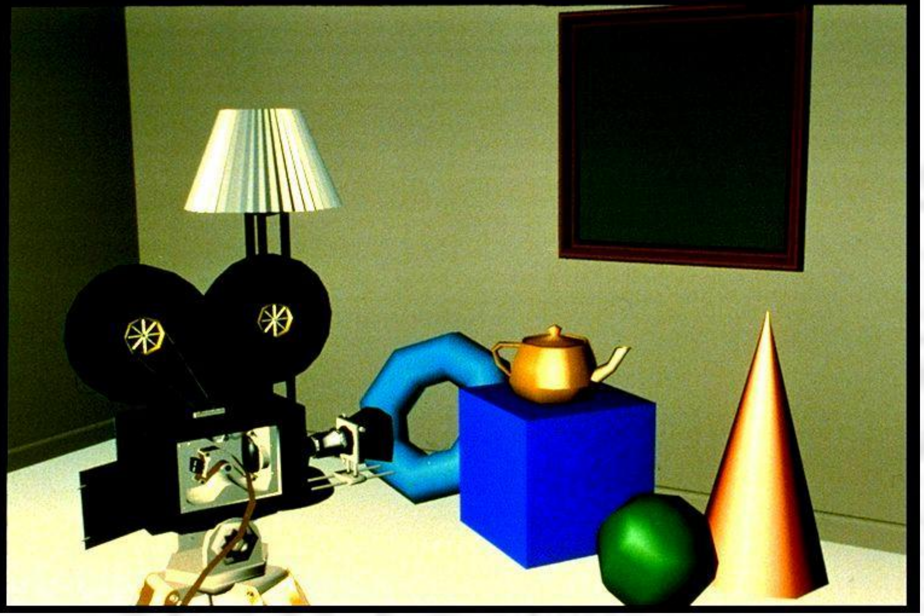


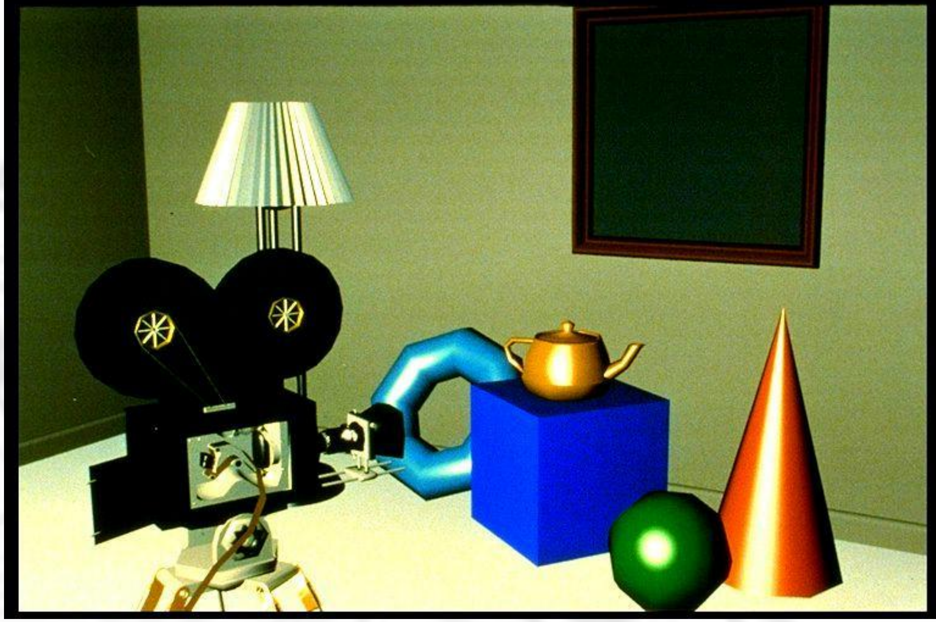


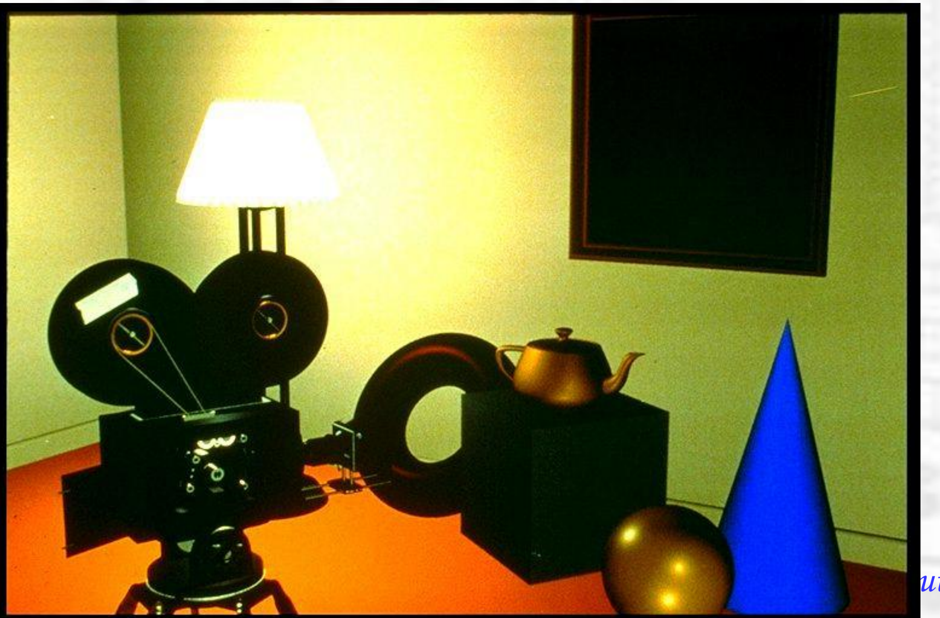


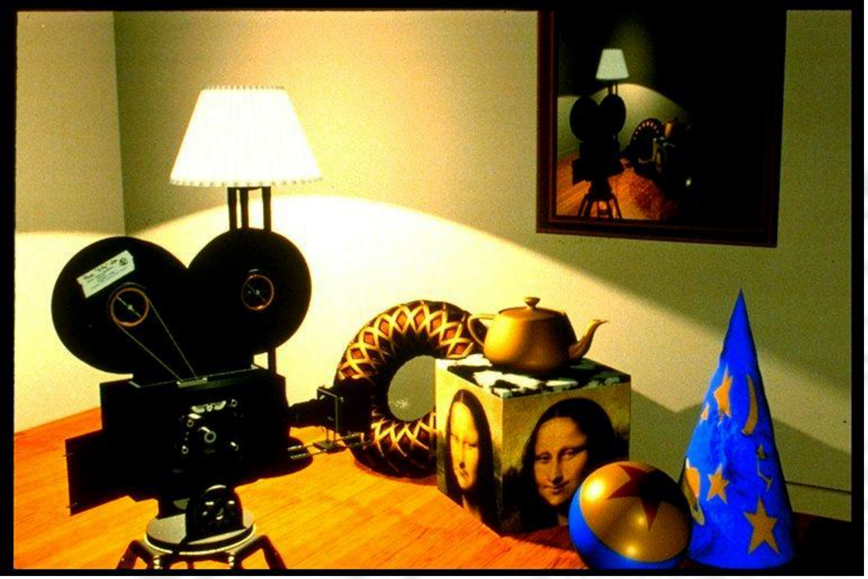


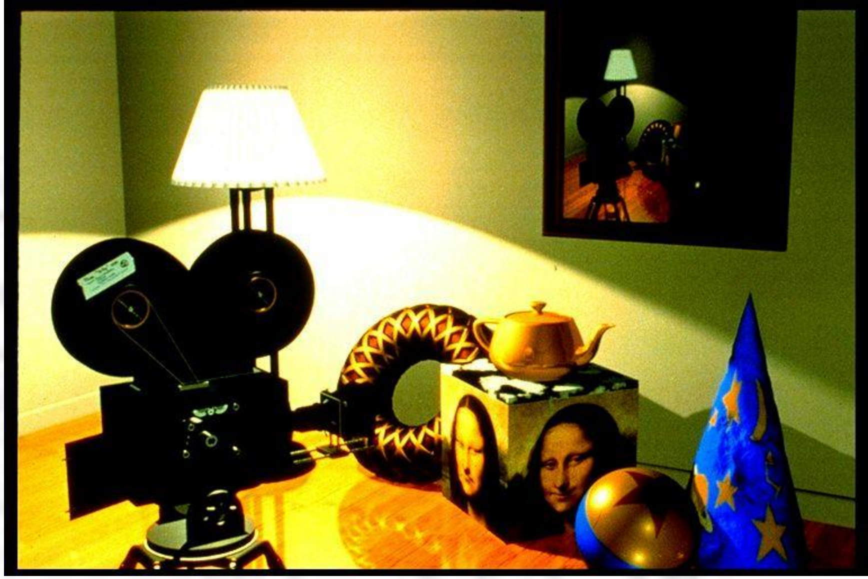













2006 RB


Christoph Hormann <chris_hormann@gmx.de>

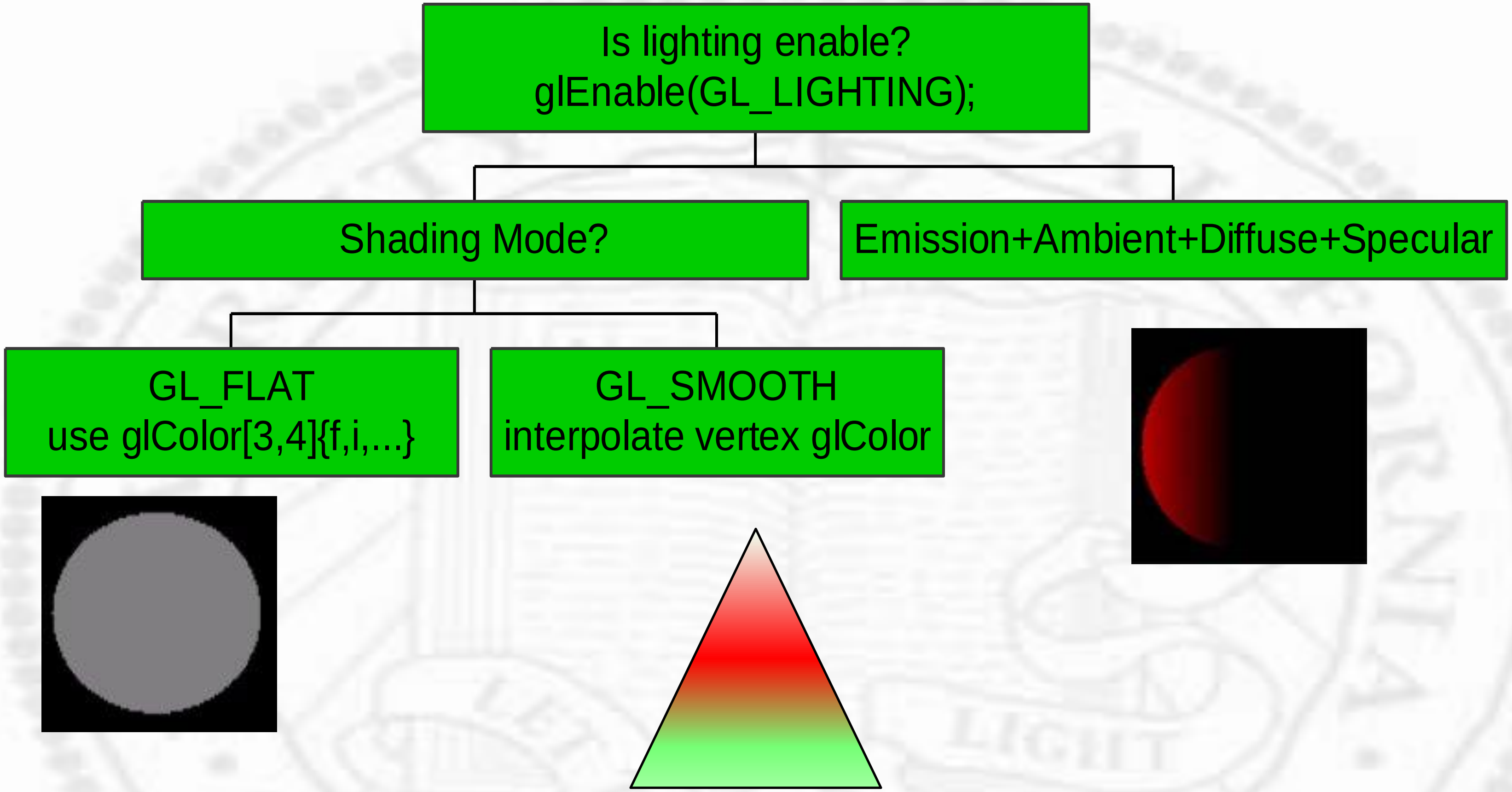
Many Things can Happen

- Hidden Surface Removal
- Flat color
- Single light
- Multiple lights
- High light
- Interpolated shading
- Texture mapping
- Shadow
- Ray tracing
- Radiosity

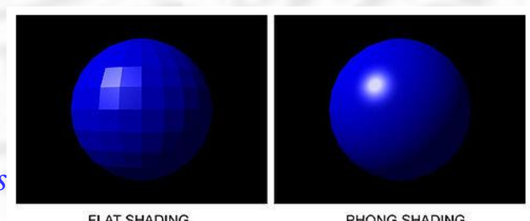
Homework #2 – Color Models

- R, G, B, (A)
- There are a lot more
  - CMY, YIQ, HSV, HLS, CIE
- Issues with half tone and dithering
  - Matching color and spatial resolution
  - For screen display and web delivery

Homework #2 –Shading Models

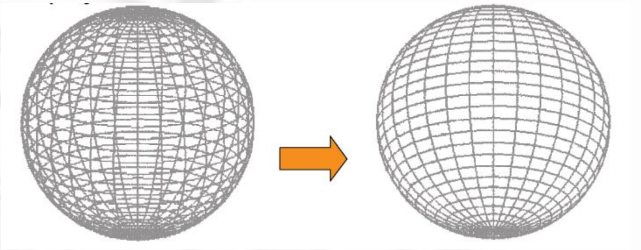


- Shading for smooth surfaces (Gouraud and Phong Shading)



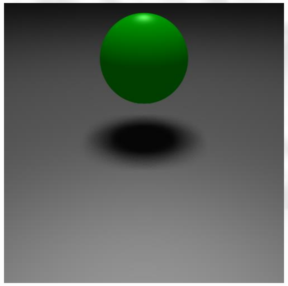
Homework #2 –HLHS Removal

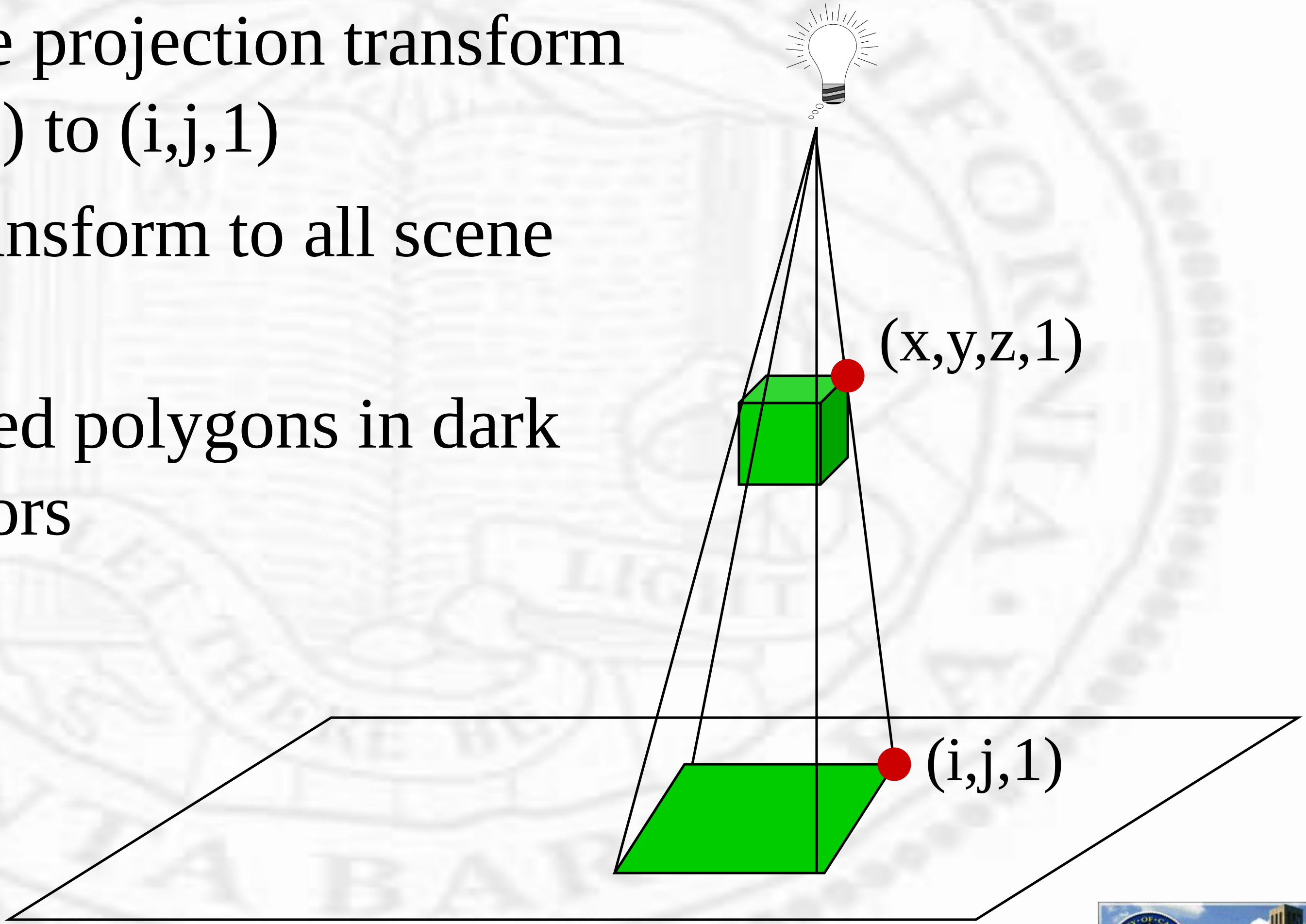
- Very simple, you need to do
  - Prepare buffer
    - glutInitDisplay(GLUT_DEPTH | …);
      - *distance* to the view point is recorded
    - glClear(GL_DEPTH_BUFFER_BIT);
      - clear to the far clipping plan distance (1.0)
  - Enable depth comparison
    - glEnable(GL_DEPTH_TEST);
  - Tell OpenGL how to do the depth comparison
    - glDepthFunc(); default is GL_LESS (in *front* of the far clipping plane)
    - Visible z values are *negative*, but distance (depth) is *positive*



Homework #2 – (Fake) Shadow

- Figure out 3D coordinates (query OpenGL)
- Figure out the projection transform From (x,y,z,1) to (i,j,1)
- Apply this transform to all scene polygons
- Draw projected polygons in dark (shadow) colors





Math

$$line \quad \begin{cases} x = l_x + t(p_x - l_x) \\ y = l_y + t(p_y - l_y) \\ z = l_z + t(p_z - l_z) \end{cases}$$

$$plane \qquad z = 0$$

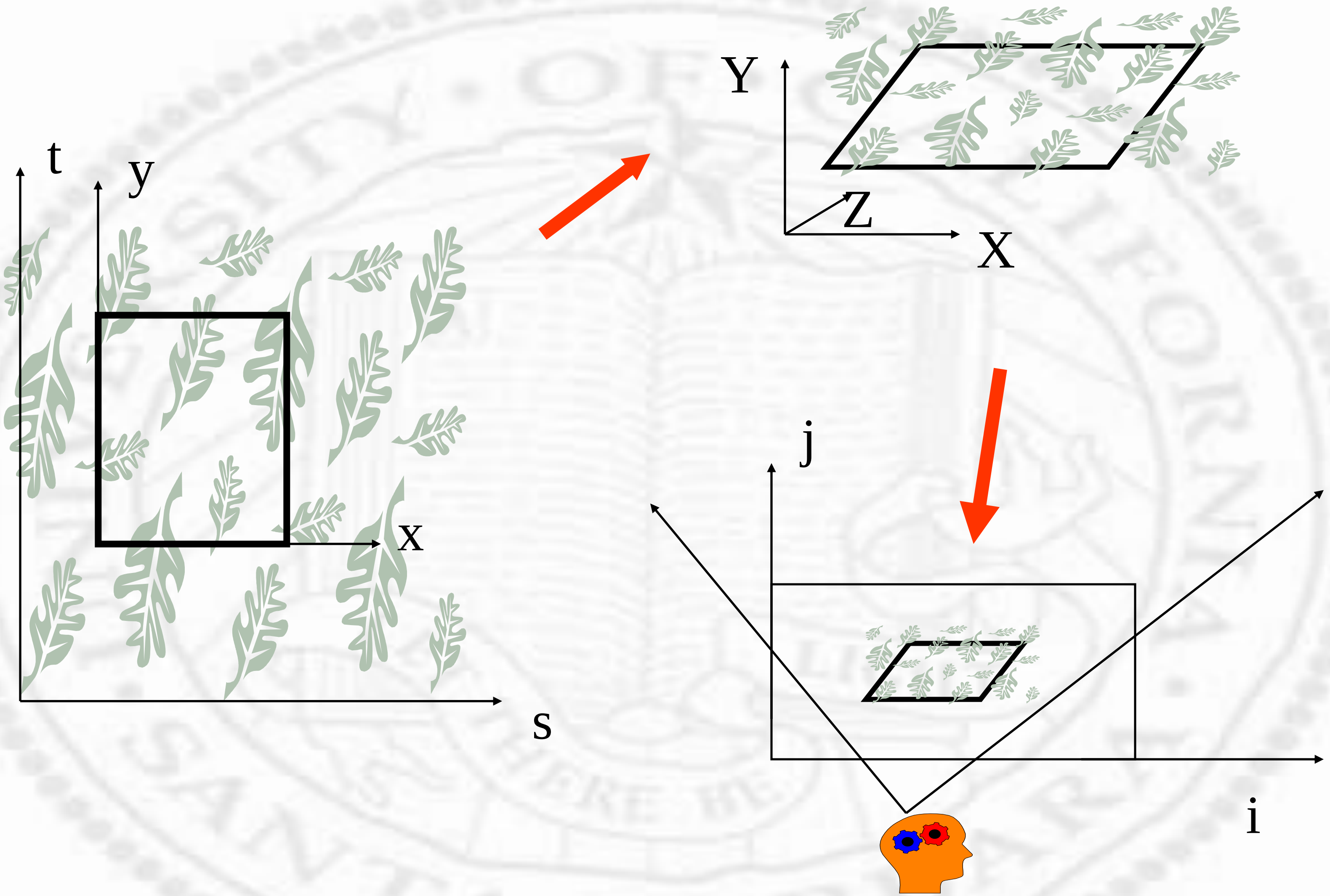
$$\Rightarrow l_z + t(p_z - l_z) = 0$$

$$\Rightarrow t = -\frac{l_z}{(p_z - l_z)}$$

$$\Rightarrow x = \frac{l_z p_x - l_x p_z}{(p_z - l_z)}, y = \frac{l_z p_y - l_y p_z}{(p_z - l_z)}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} l_z & 0 & -l_x & 0 \\ 0 & l_z & -l_y & 0 \\ 0 & 0 & 1 & -l_z \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$

Homework #2 – Texture

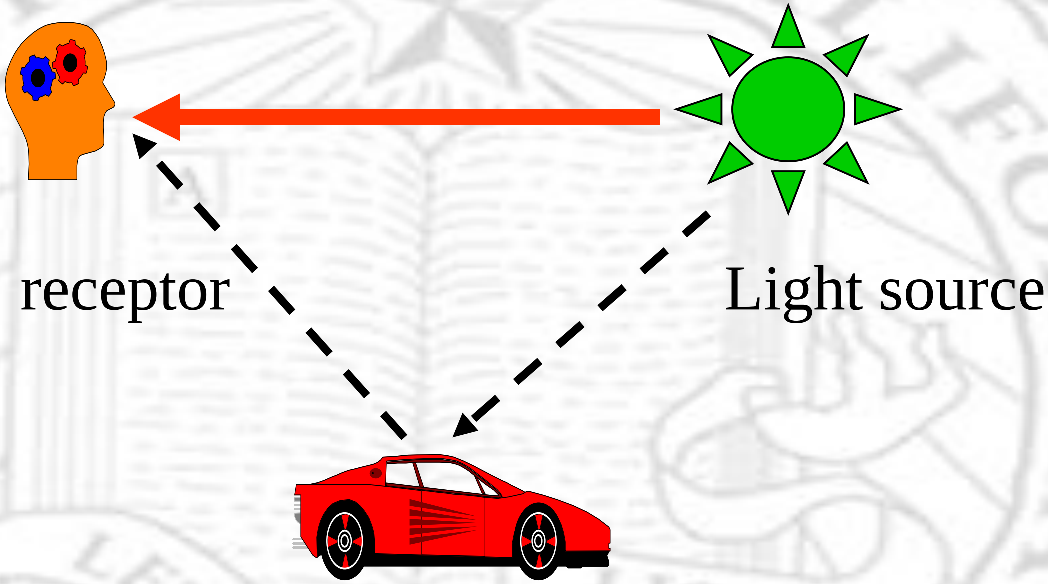


Homework #2 – Texture

- Hardest part – get images into your program (OpenGL is not for image processing)
- Taking pictures – You have a phone, right?
- Importing pictures
  - Direct (JPG) – read JPG images
  - Indirect (JPG->ppm with convert) – easy read afterward
- Nitty Gritty details

```
P3
# feep.ppm
4 4
15
 0  0  0    0  0  0    0  0  0   15  0 15
 0  0  0    0 15  7    0  0  0    0  0  0
 0  0  0    0  0  0    0 15  7    0  0  0
15  0 15    0  0  0    0  0  0    0  0  0
```

Color Model and Color Perception



Geometry and Radiometry

- In creating and interpreting images, we need to understand two things:
  - Geometry – Where scene points appear in the image (image locations)
  - Radiometry – How “bright” they are (image values)
- **Geometric** enables us to know something about the scene location of a point imaged at pixel (*u*, *v*)
- **Radiometric** enables us to know what a pixel value implies about surface lightness and illumination

Radiometry

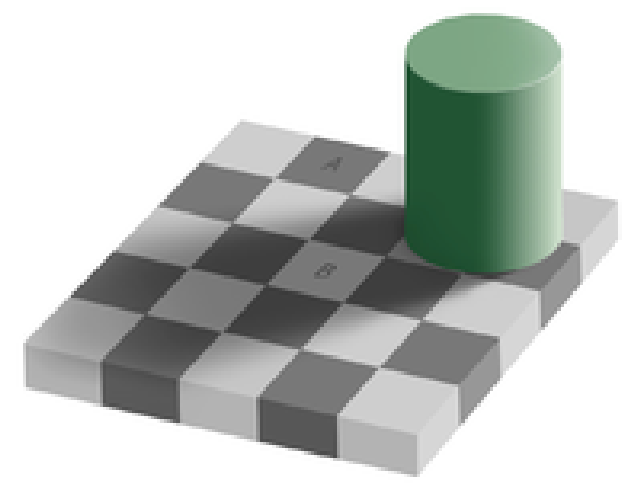
- Radiometry is the measurement of light
  - Actually, electromagnetic energy
- Imaging starts with light sources
  - Emitting photons – quanta of light energy
  - The sun, artificial lighting, candles, fire, blackbody radiators …
- Light energy interact with surfaces
  - Reflection, refraction, absorption, fluorescence…
  - Also atmospheric effects (not just solid surfaces)
- Light energy from sources and surfaces gets imaged by a camera
  - Through a lens, onto a sensor array, finally to pixel values – an image!

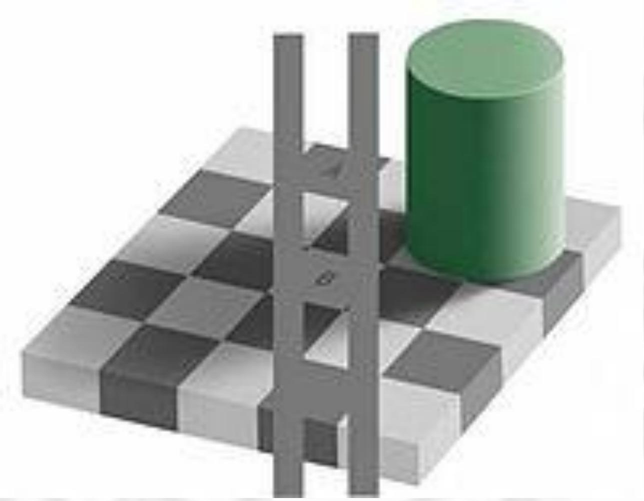
Two-Stage Approaches

- Light source + receptor
  - Various color models
  - Matching color and spatial resolution
- Light source + object + receptor
  - OpenGL model (primary ray)
  - Others (more advanced)
  - Shadow

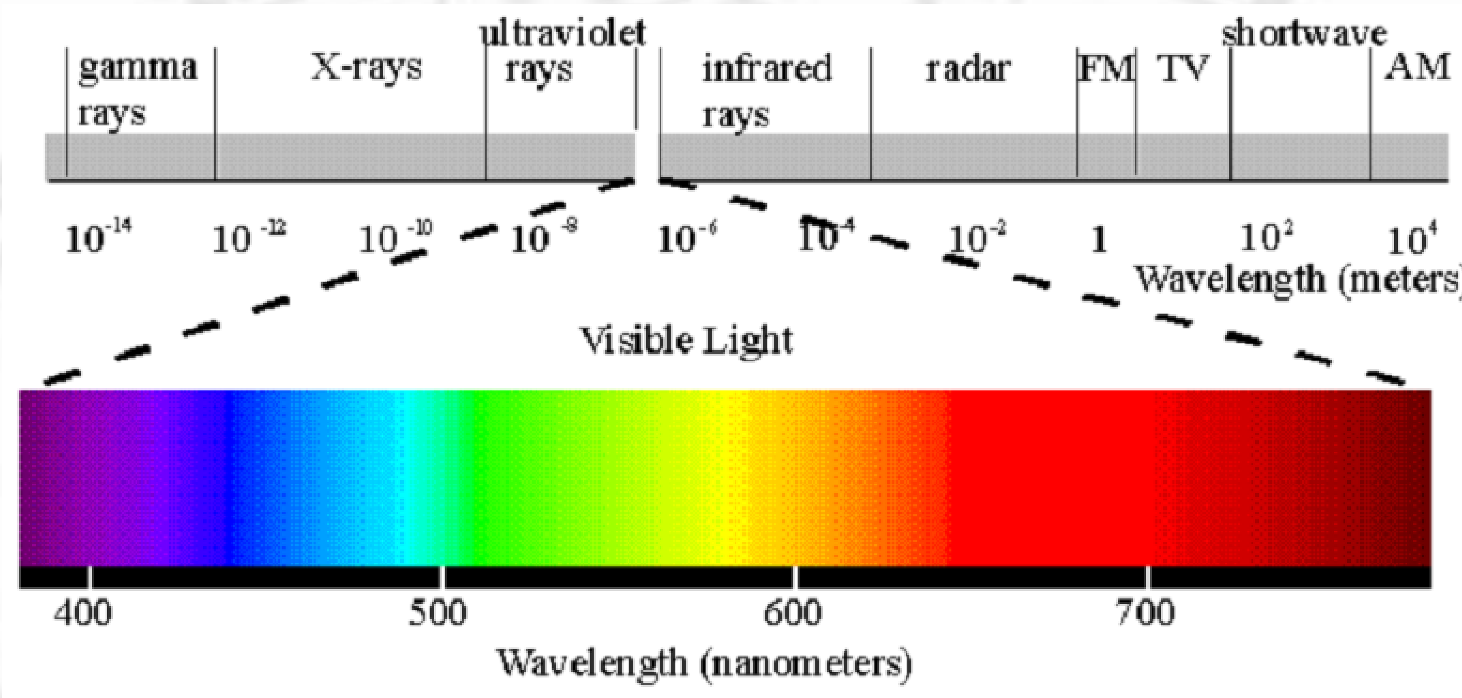
Color Perception

- Complicated
  - Nonlinear
  - Spatially variant
  - Many models and interpretations





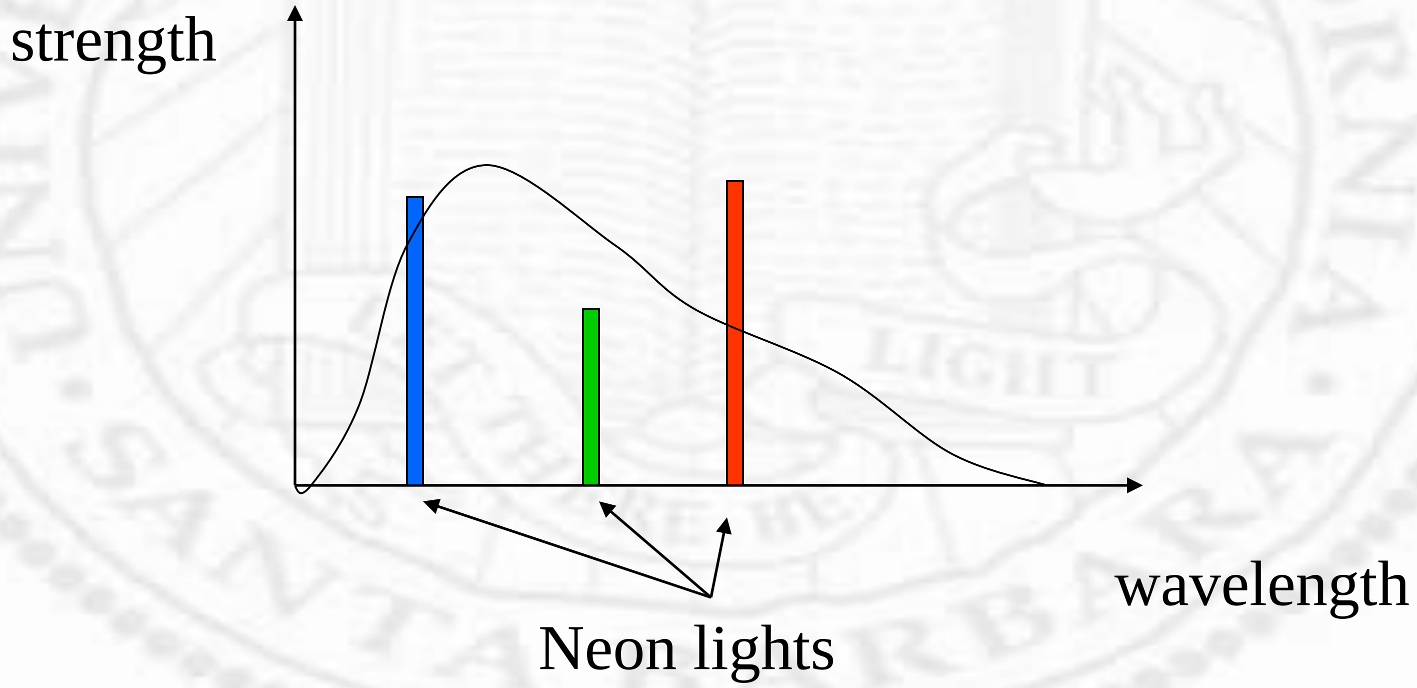
Electromagnetic (EM) Spectrum



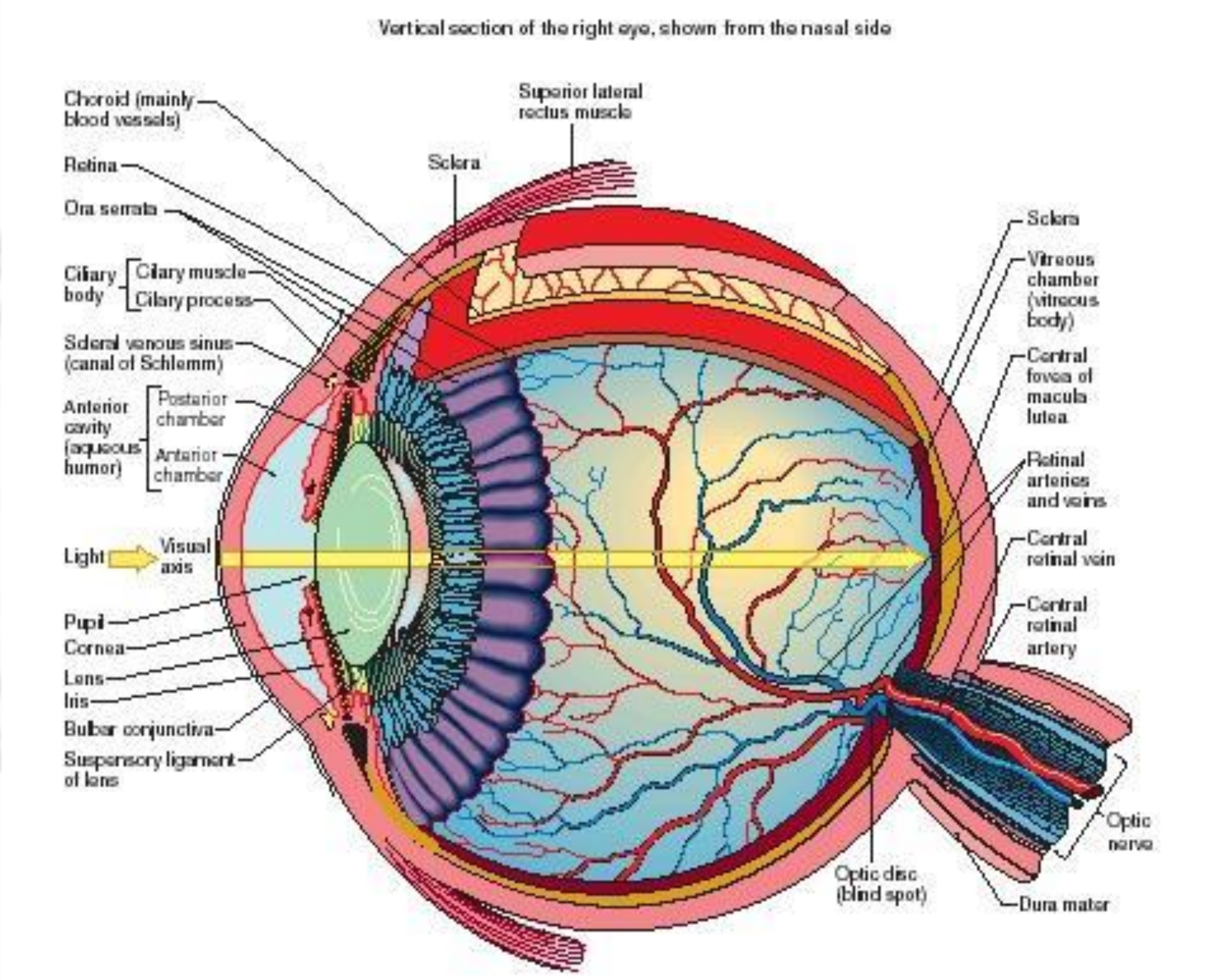
Energy, frequency, and wavelength
are related

Light Sources

- Characterized by emission strength as a function of wavelength



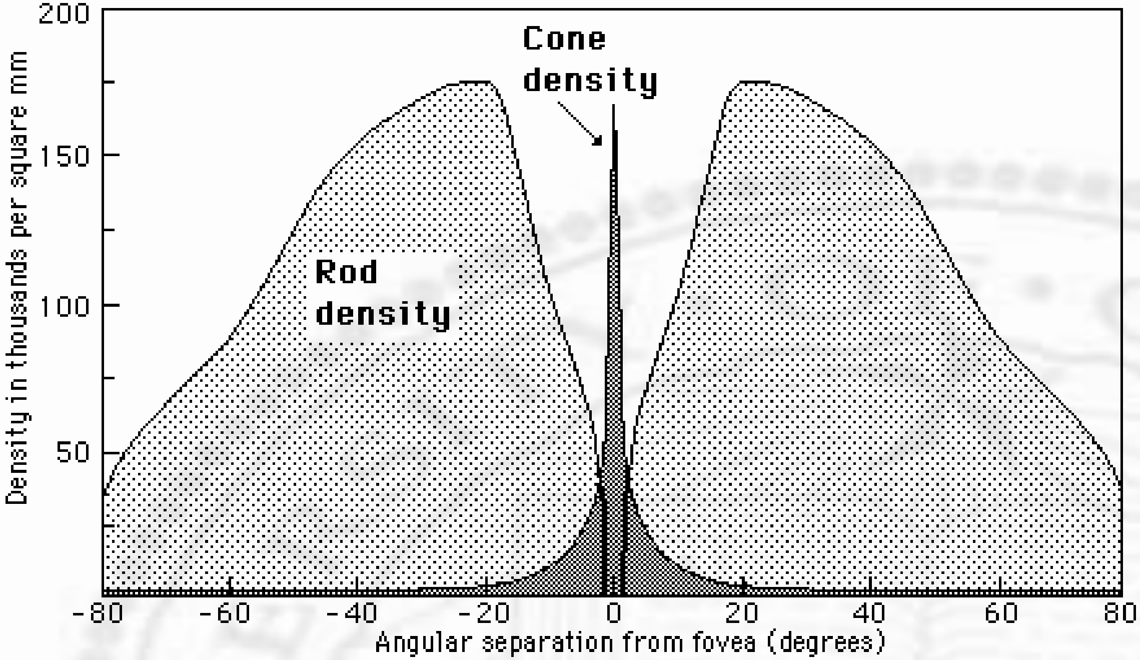
Eye

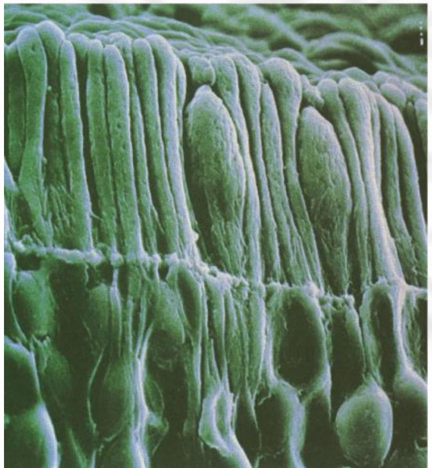

Vertical section of the right eye, shown from the nasal side
Choroid (mainly blood vessels)
Retina
Ora serrata
Ciliary body
Ciliary muscle
Ciliary process
Scleral venous sinus (canal of Schlemm)
Anterior cavity (aqueous humor)
Posterior chamber
Anterior chamber
Light
Visual axis
Pupil
Cornea
Lens
Iris
Bulbar conjunctiva
Suspensory ligament of lens
Superior lateral rectus muscle
Sclera
Sclera
Vitreous chamber (vitreous body)
Central fovea of macula lutea
Retinal arteries and veins
Central retinal vein
Central retinal artery
Optic nerve
Optic disc (blind spot)
Dura mater

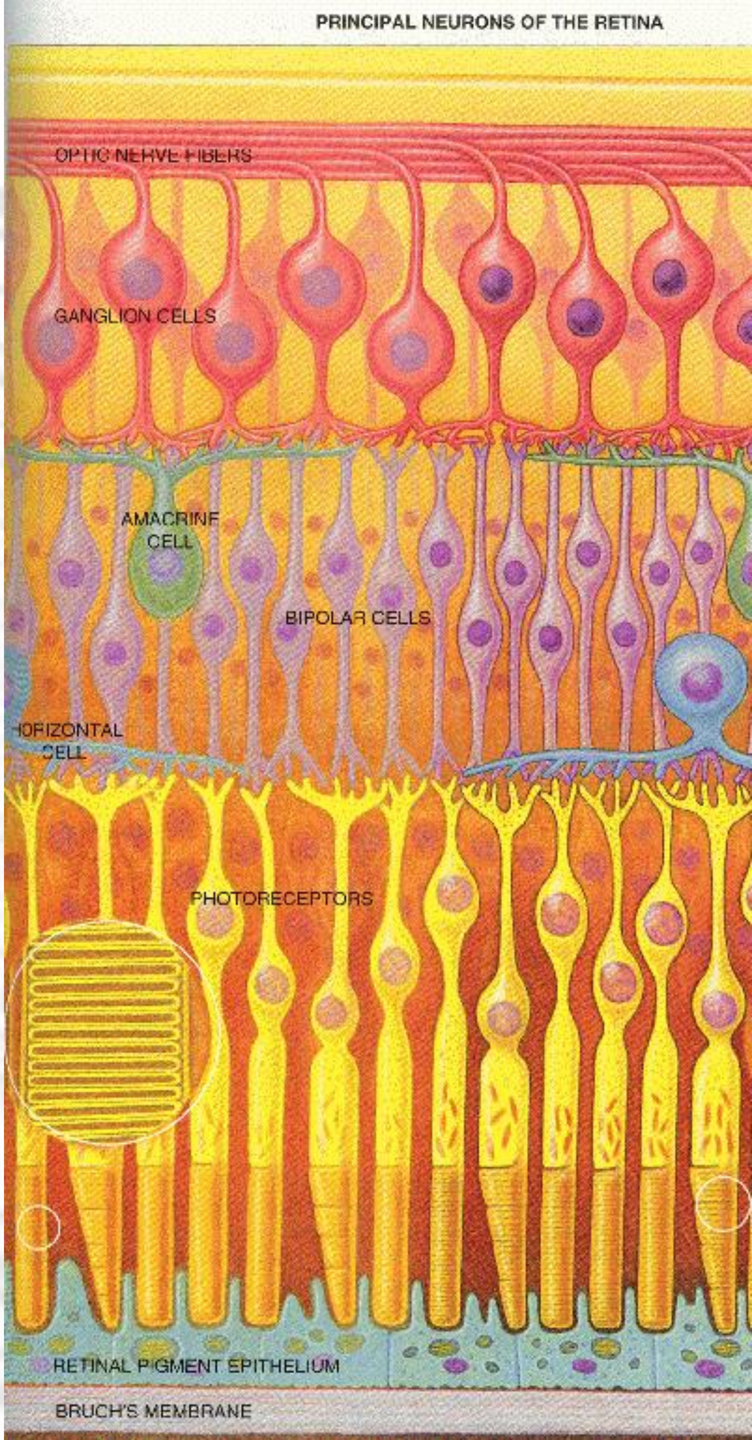
Eye

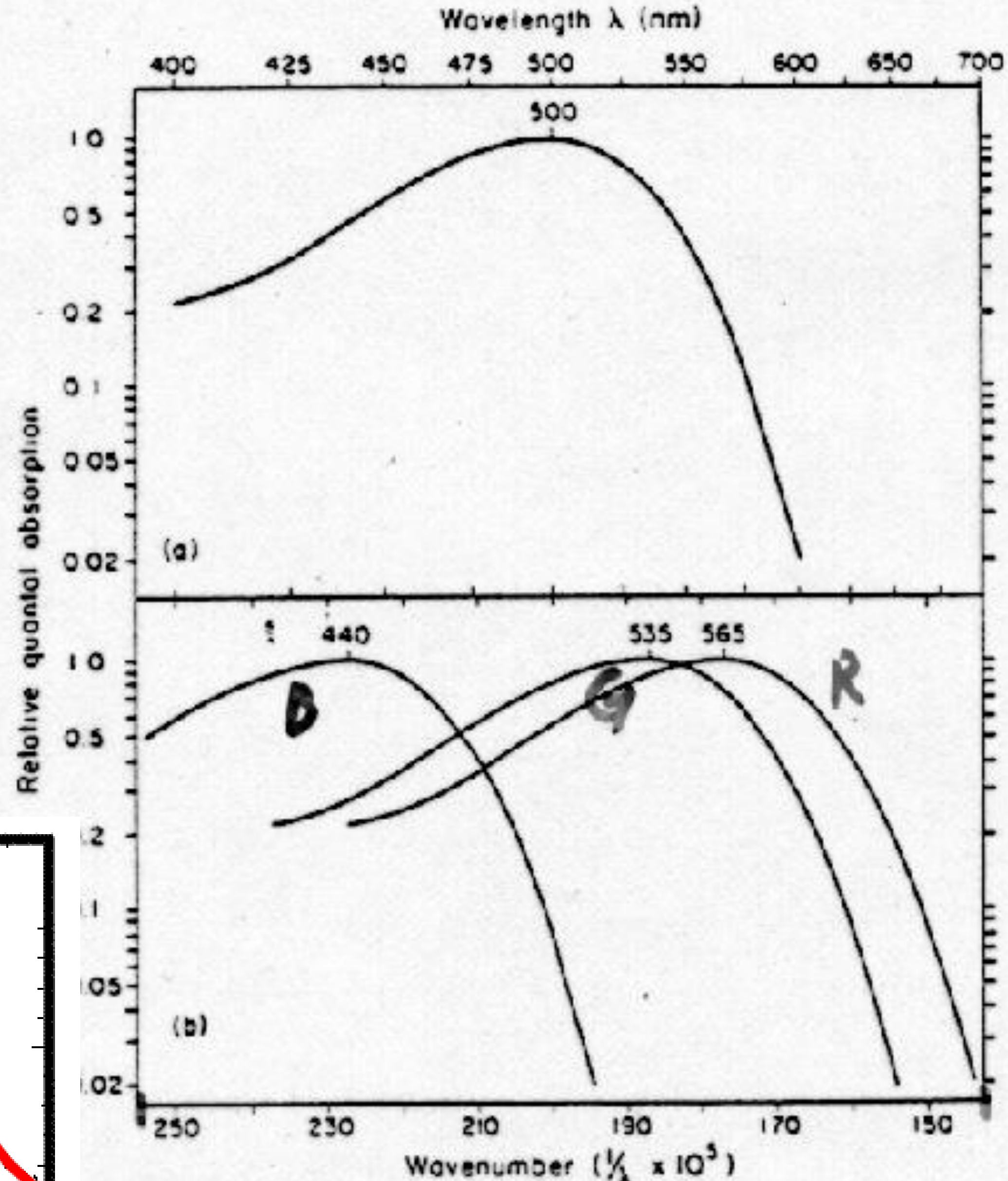
- Characterized by spectrum sensitivity functions
- Rods
  - night vision
  - achromatic (gray) perceptions
  - lack fine details
  - found mostly at periphery of retina
- Cones
  - daylight, color vision
  - found mostly at foveal pit

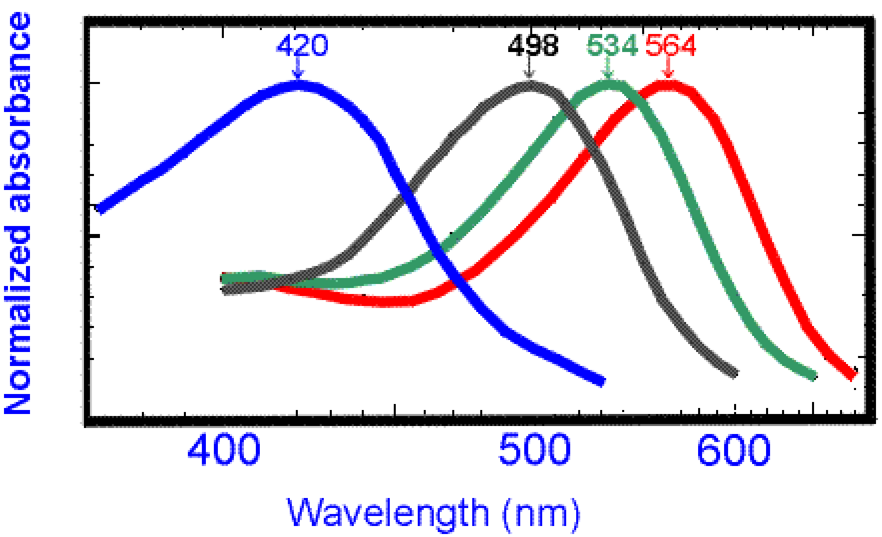
$$R,G,B = \int_{low\_band}^{high\_band} incident\_light\_strength(\lambda) \cdot eye\_sensitivity(\lambda) d\lambda$$


Cone density
Rod density
Density in thousands per square mm
200
150
100
50
-80 -60 -40 -20 0 20 40 60 80
Angular separation from fovea (degrees)



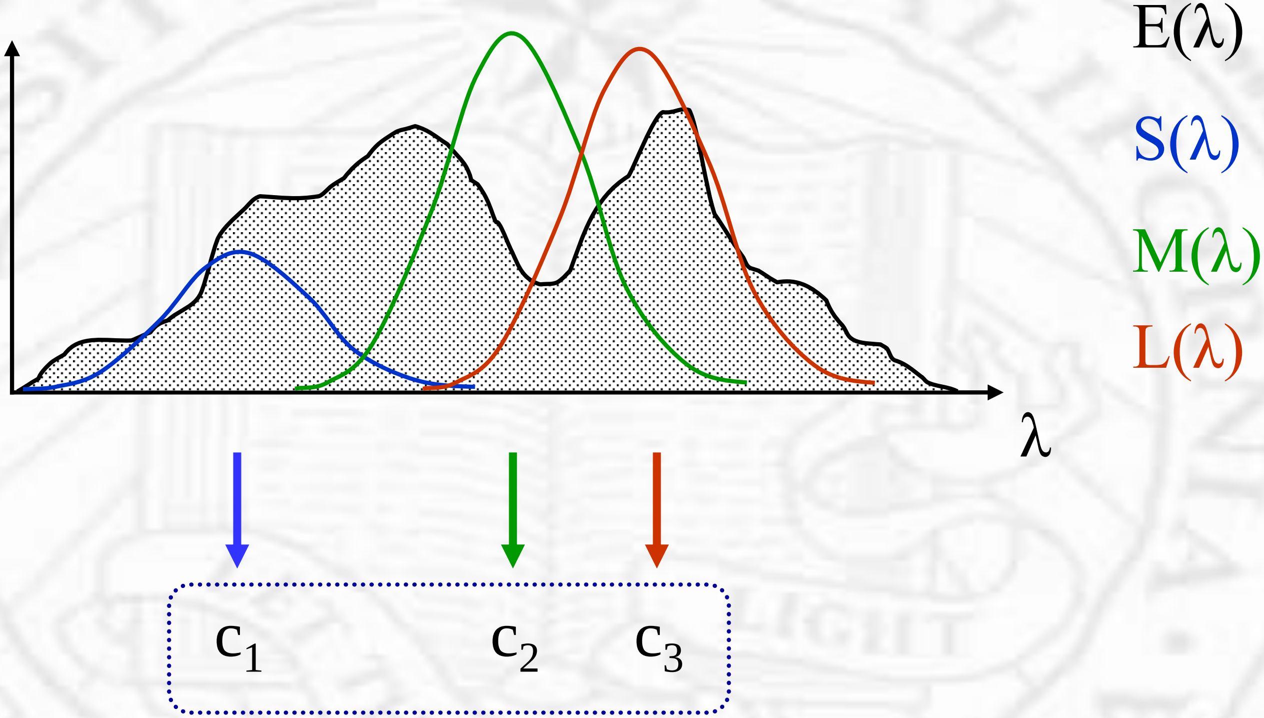

PRINCIPAL NEURONS OF THE RETINA
OPTIC NERVE FIBERS
GANGLION CELLS
AMACRINE CELL
BIPOLAR CELLS
HORIZONTAL CELL
PHOTORECEPTORS
RETINAL PIGMENT EPITHELIUM
BRUCH'S MEMBRANE





After Bowmaker & Dartnall, 1980

Responses to a source

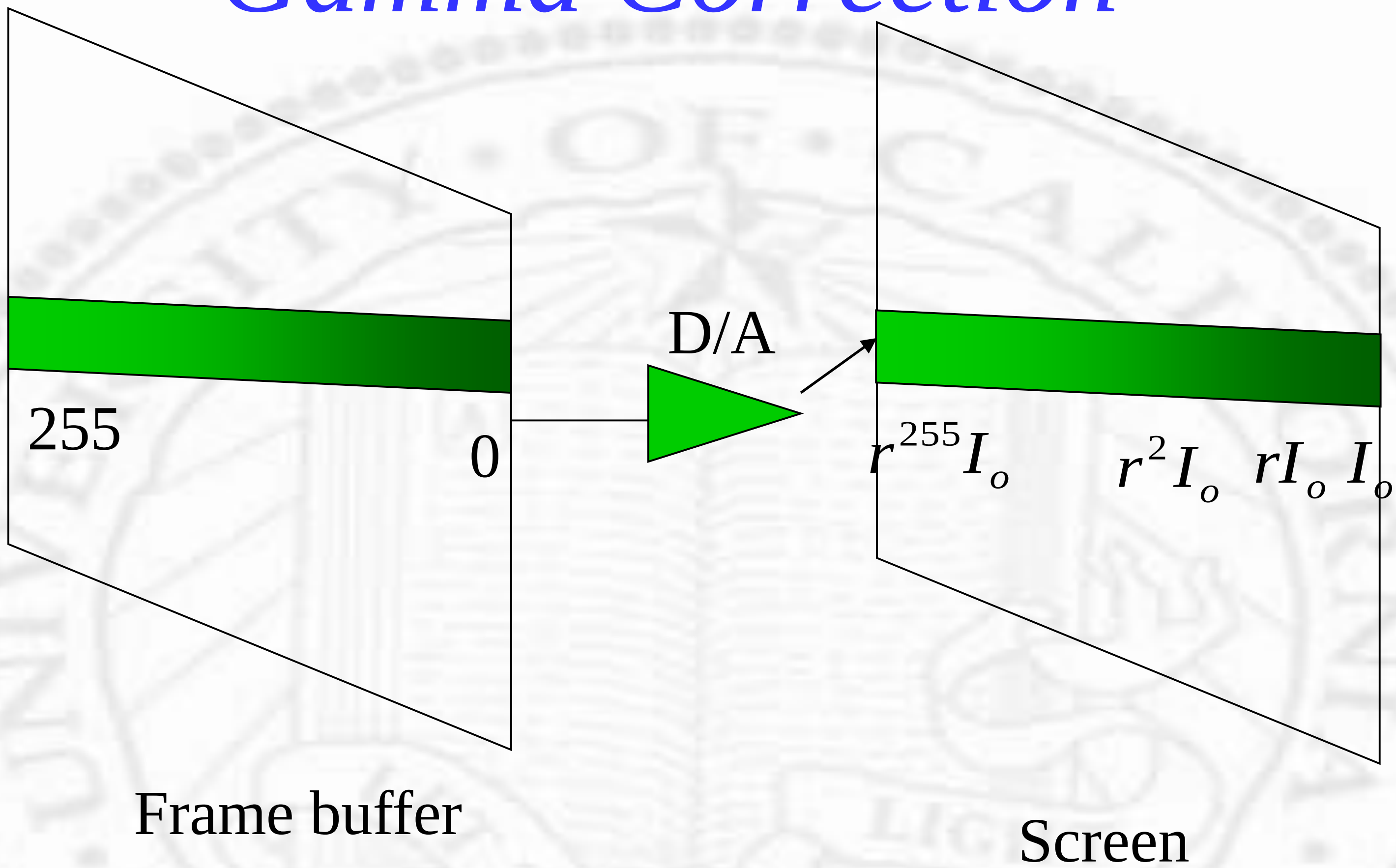


Receptor responses

Achromatic Perception

- Simple model brightness = R + G + B
- However, it is not a linear function, but a log function
- 50->100->150 does not give a linear ramp, but 50->100->200->400 does
- But when we specify intensity of a pixel, we will like to have 0 to 255 to be a linear ramp
- Need *gamma* correction

Gamma Correction



$$r^{255} I_o = 1$$

$$r = \left(\frac{1}{I_o}\right)^{\frac{1}{255}}, I_j = I_o^{\frac{225-j}{255}}$$

Gamma Correction (cont.)

- Physically, the intensity (I) of a phosphor cell is proportional to the number of incident electrons (2.2 to 2.5 for CRT)
- The number of electrons (N) in turn is proportional to control grid voltage (J)
- Control grid voltage (J) is porportional to the pixel value (V) specified

$$I = kN^{\gamma}$$

$$= k' J^{\gamma}$$

$$= k''V^{r}$$

$$I_j = k''V_j^{\,r}$$

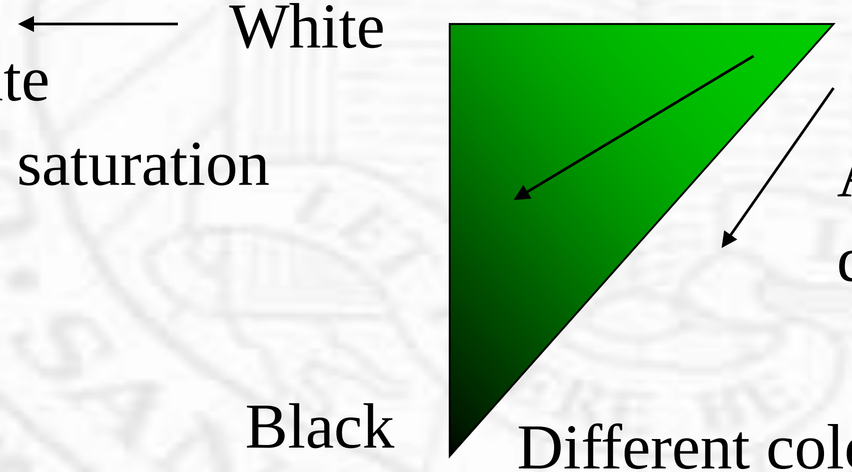
$$V_j = Round(\frac{I_j}{k''})^{\frac{1}{r}}$$

Gamma Correction (cont.)

- Instead of using pixel value directly, a look-up table with V is stored, a process called gamma correction

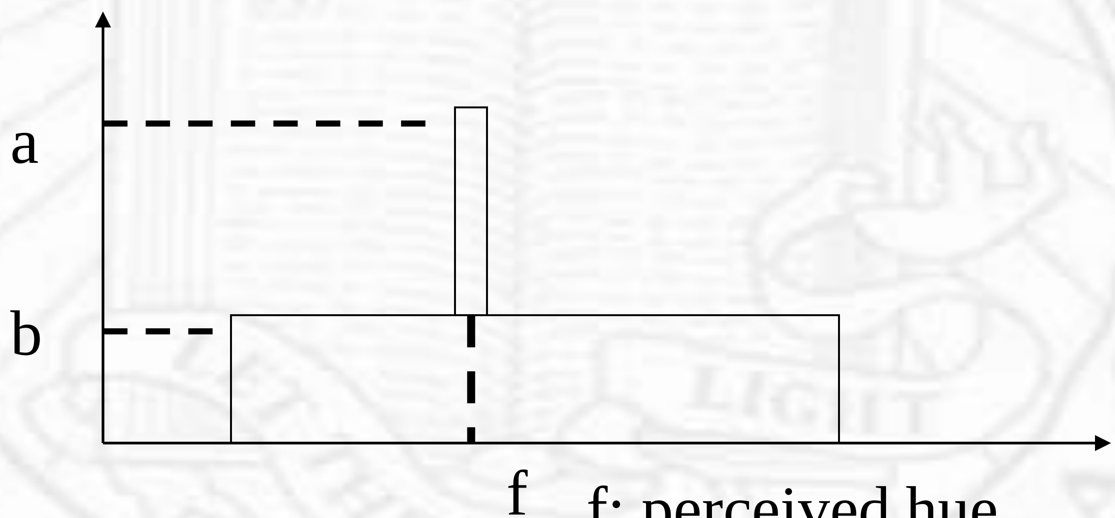
Chromatic Perception

- Many different models, old and new, motivated by different usage and conventions
- Artists and painters



Psychophysics model

- Dominant wavelength (perception of hue)
- Excitation purity (saturation)
- Luminance (amount of flight)

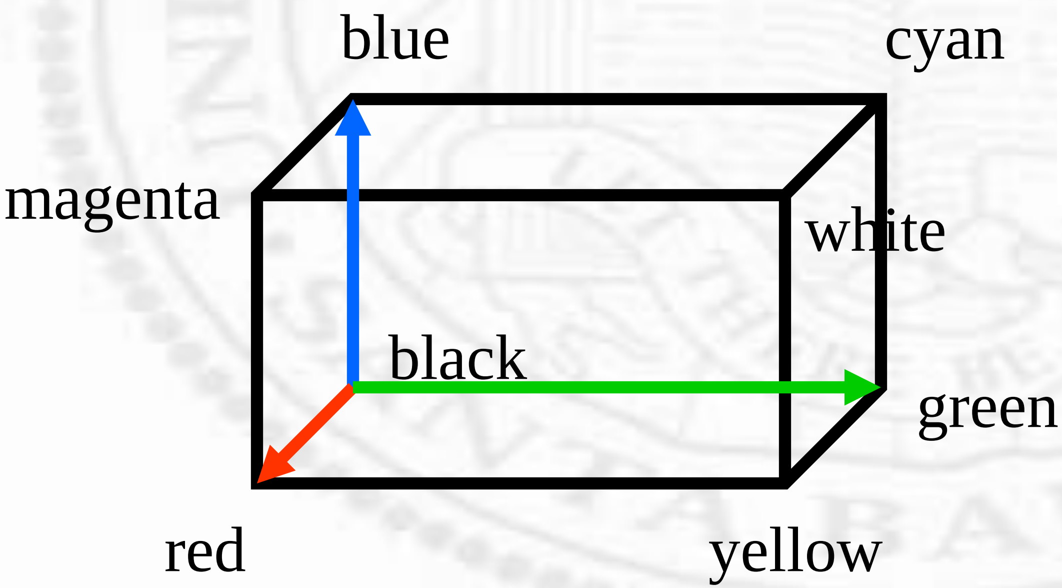


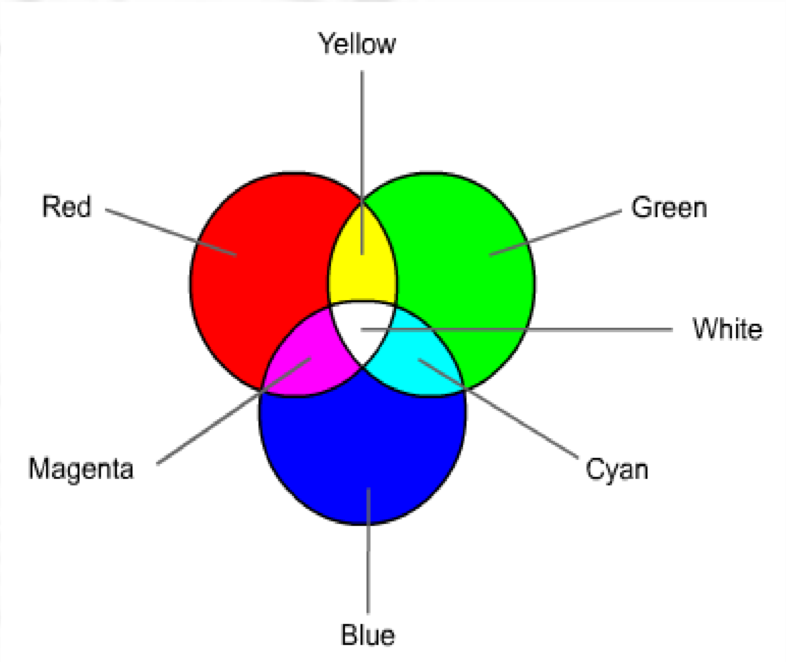
f: perceived hue
a &b: luminance

b=0 purity = 100%
a=b purity = 0%

R-G-B model

- Color CRT monitor
- Additive system

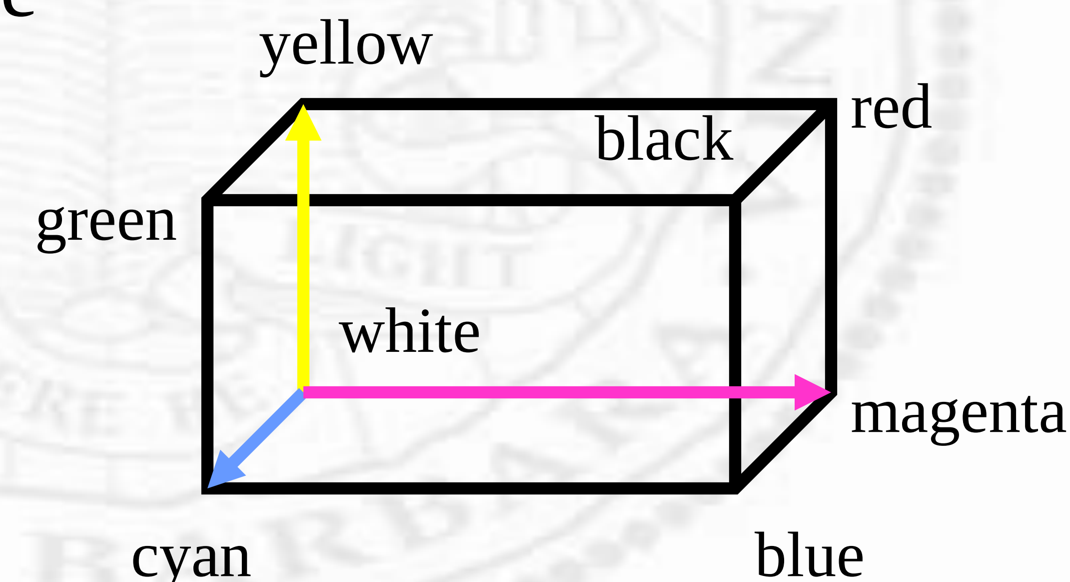


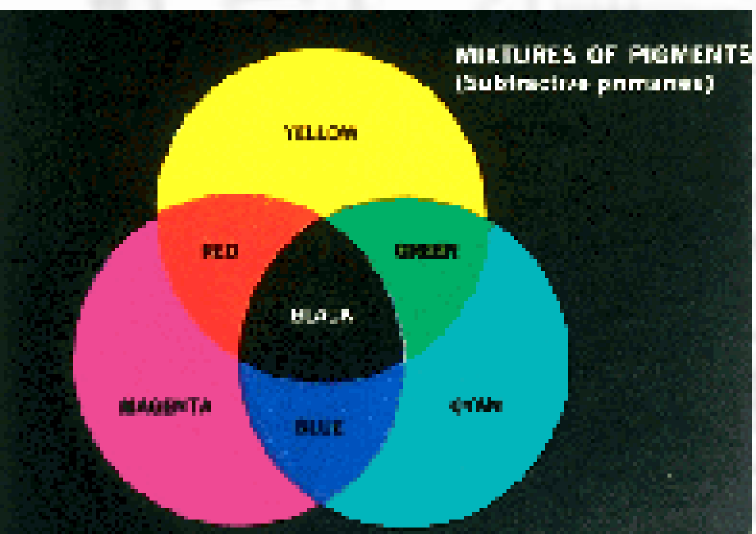


C-M-Y model

- Color printer
- Subtractive system
- cyan absorbs red
- magenta absorbs green
- yellow absorbs blue

Cyan+magenta = blue
cyan+yellow = green
magenta+yellow = red





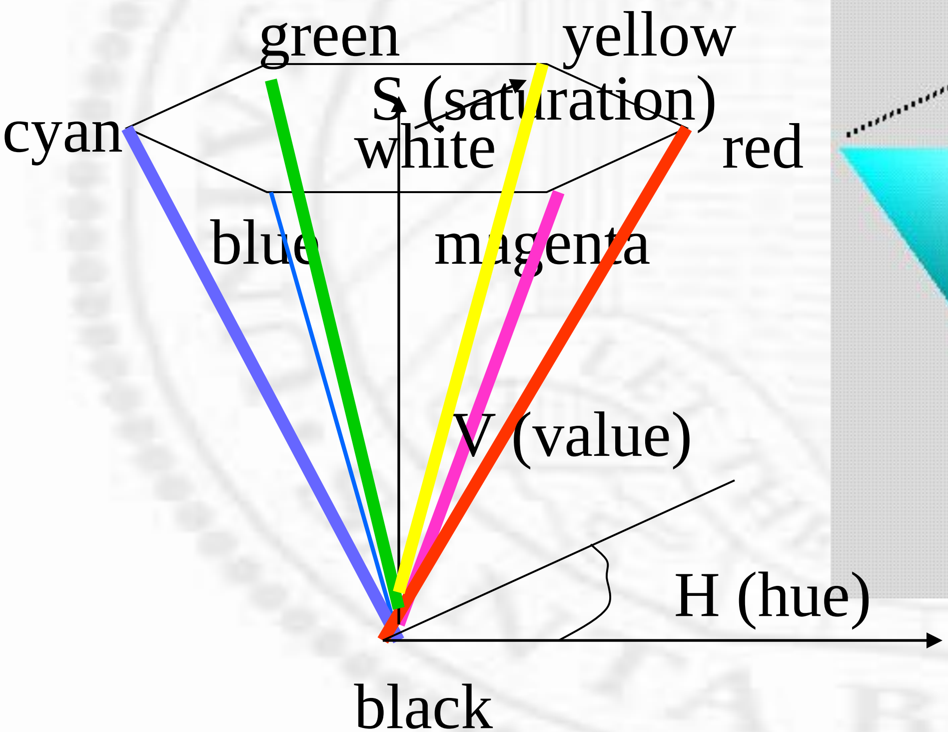
Y-I-Q model

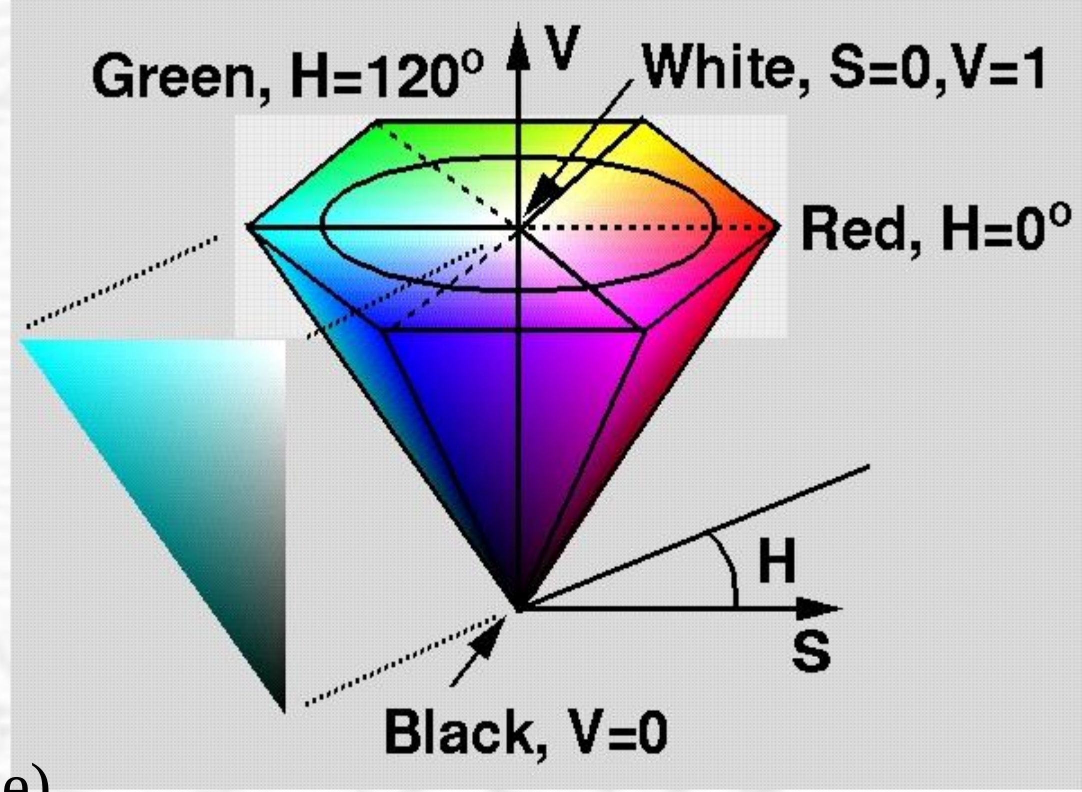
- Commercial color TV broadcasting
- Backward compatible with B/W TV
- Y uses 4MHz, I 1.5 MHz, Q 0.6 MHz because eyes are more sensitive to monochrome than color

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.3 & 0.59 & 0.11 \\ 0.6 & -0.28 & -0.32 \\ 0.21 & -0.52 & 0.31 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

H-S-V model

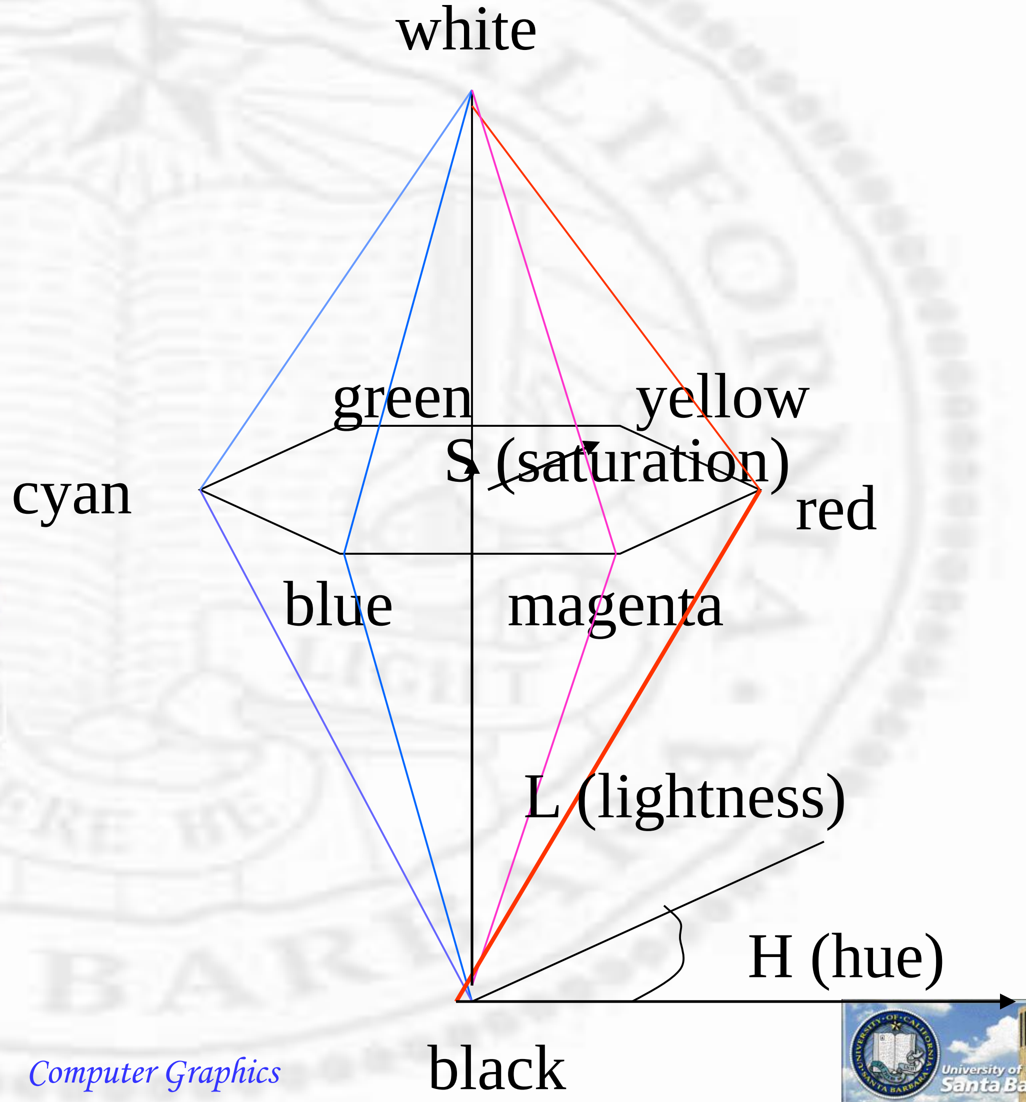
- Emulate Artists' model in 3D



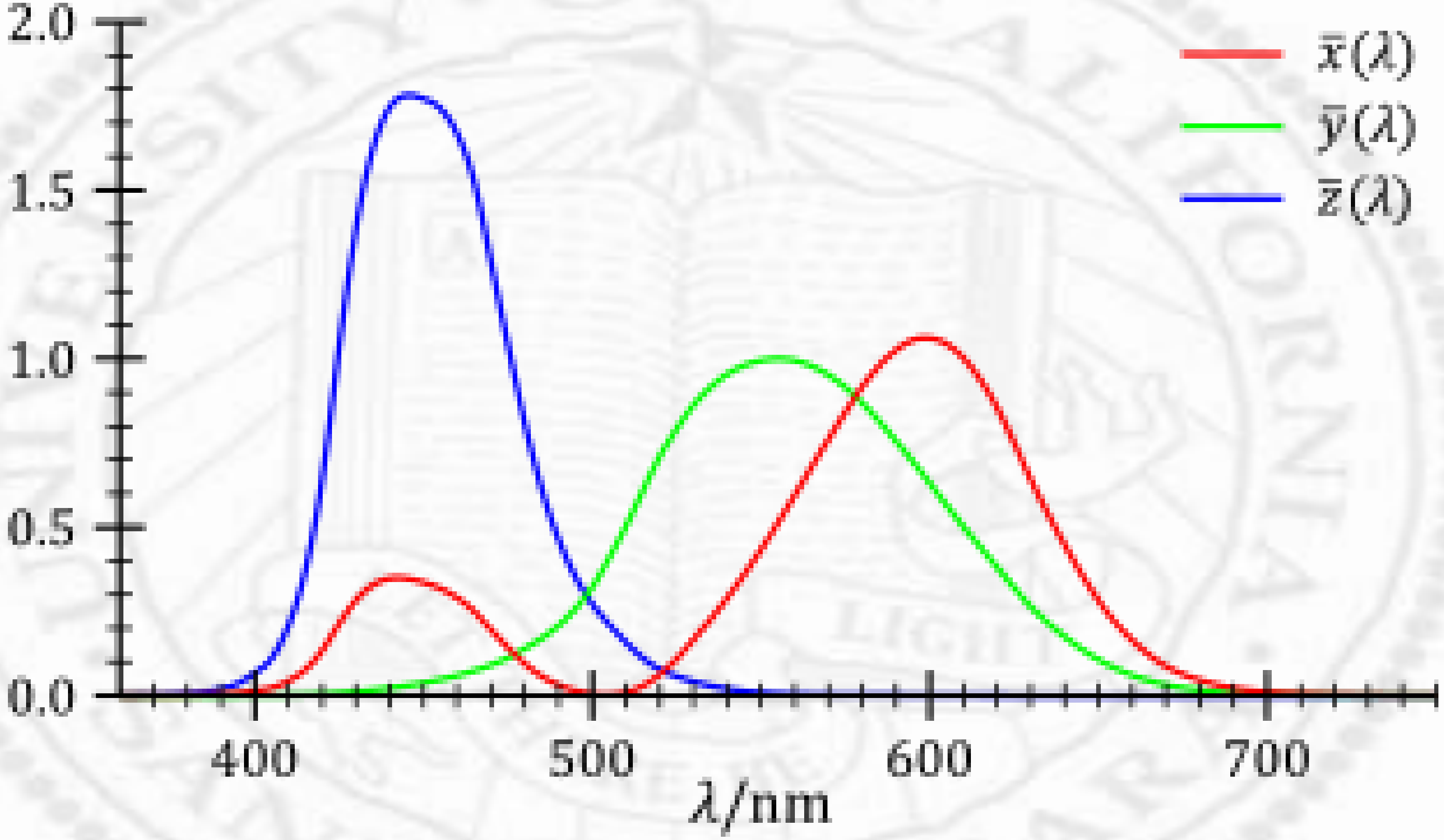


H-L-S model

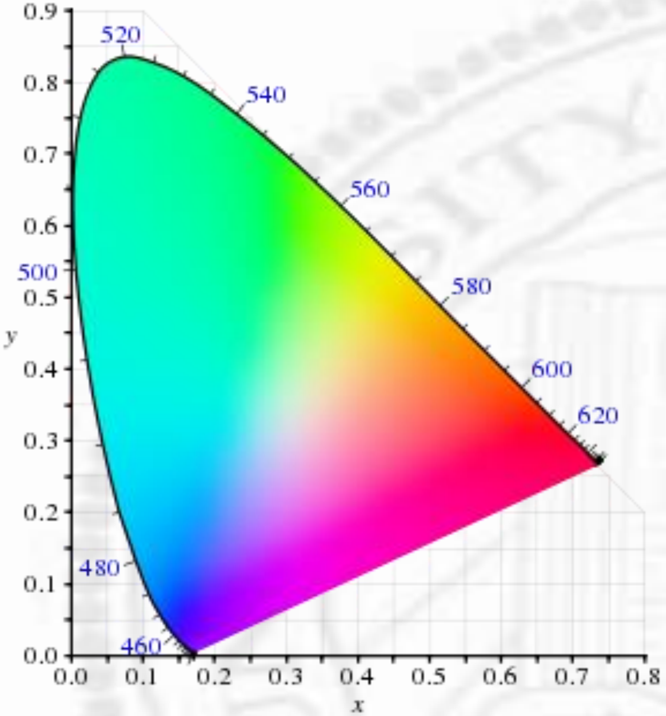
- Emulate Artists' model in 3D



CIE Standard



CIE Standard



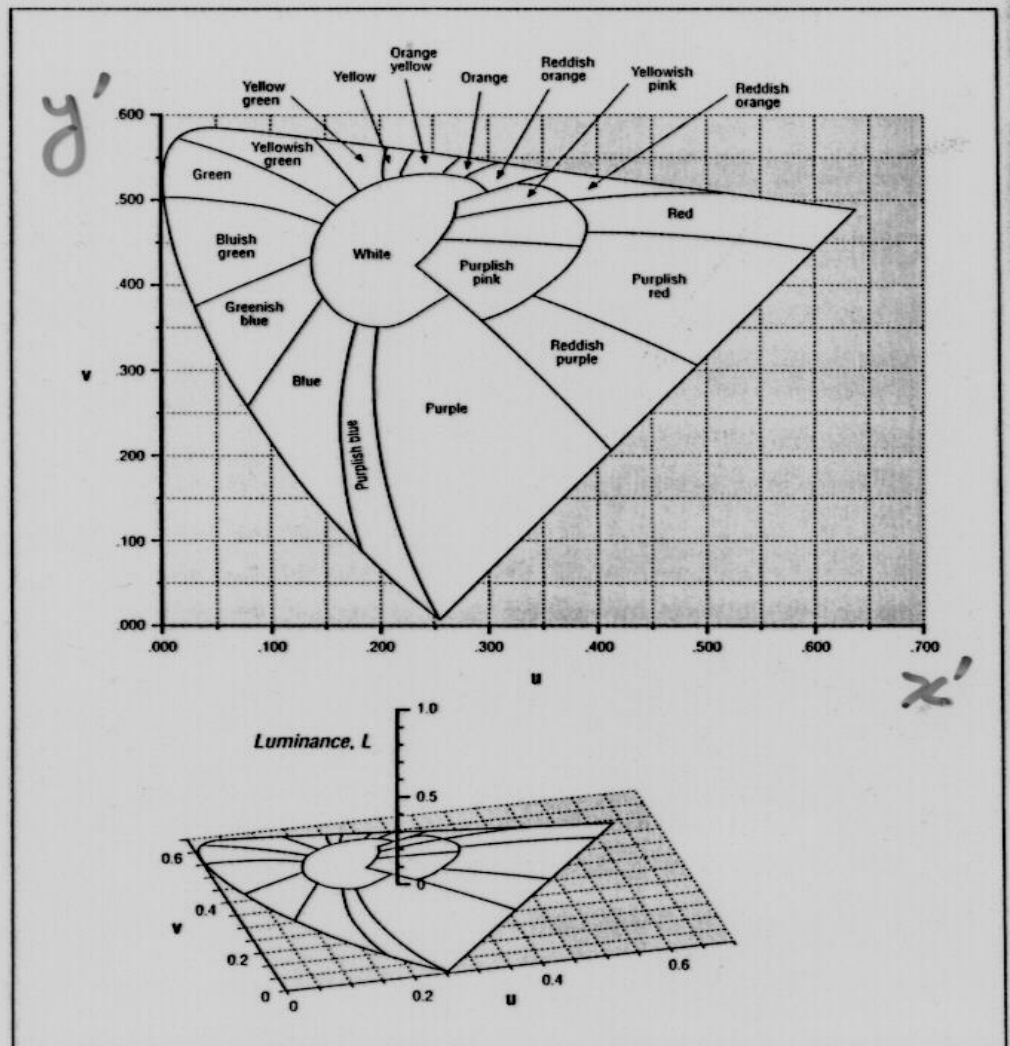
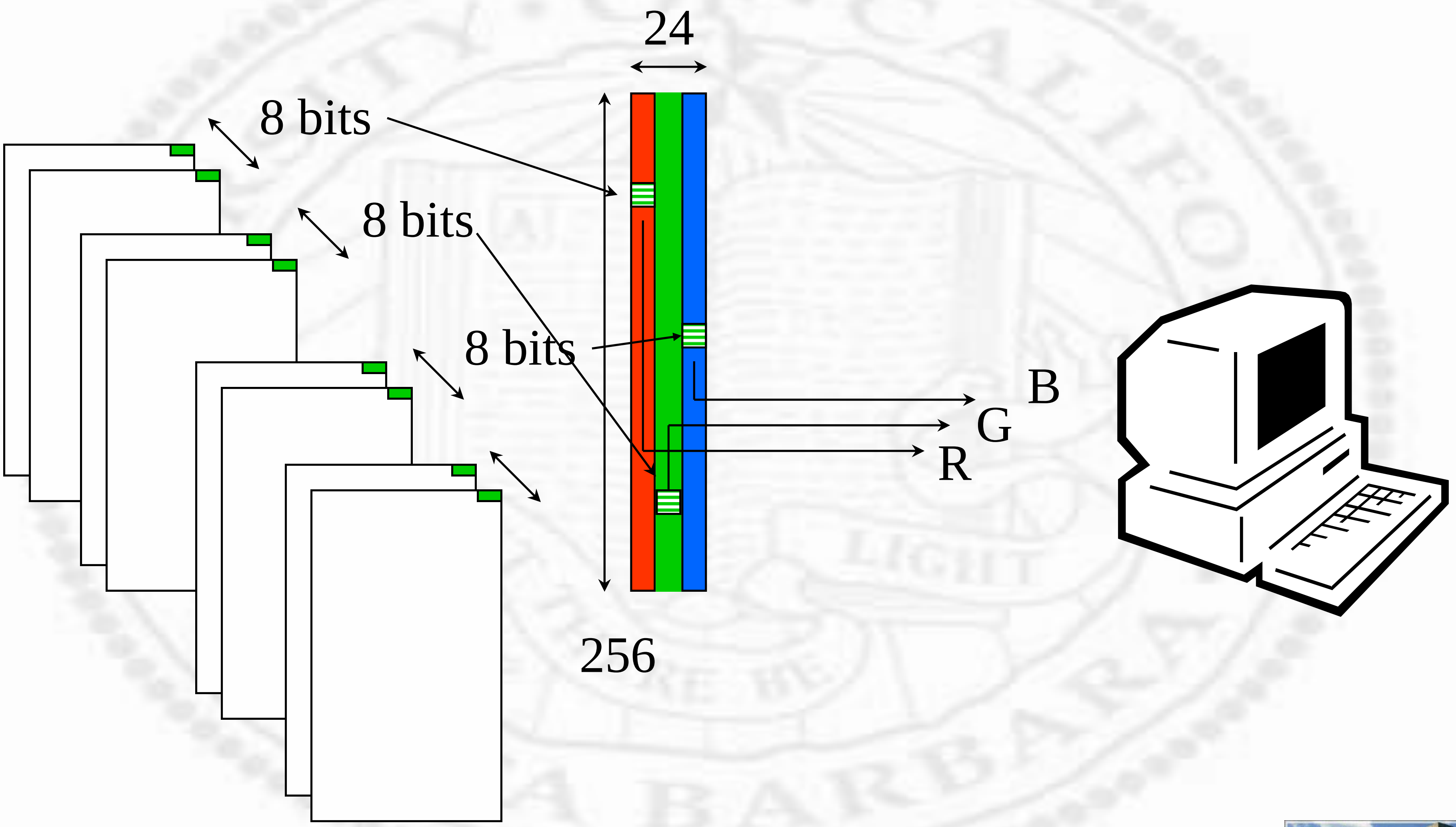


Figure 5-8. The CIE-LUV color model

OpenGL Color Models

- A color monitor model with A (alpha, transparency) - RGBA model, or
- A color-index model with a colormap (set through aux-library)

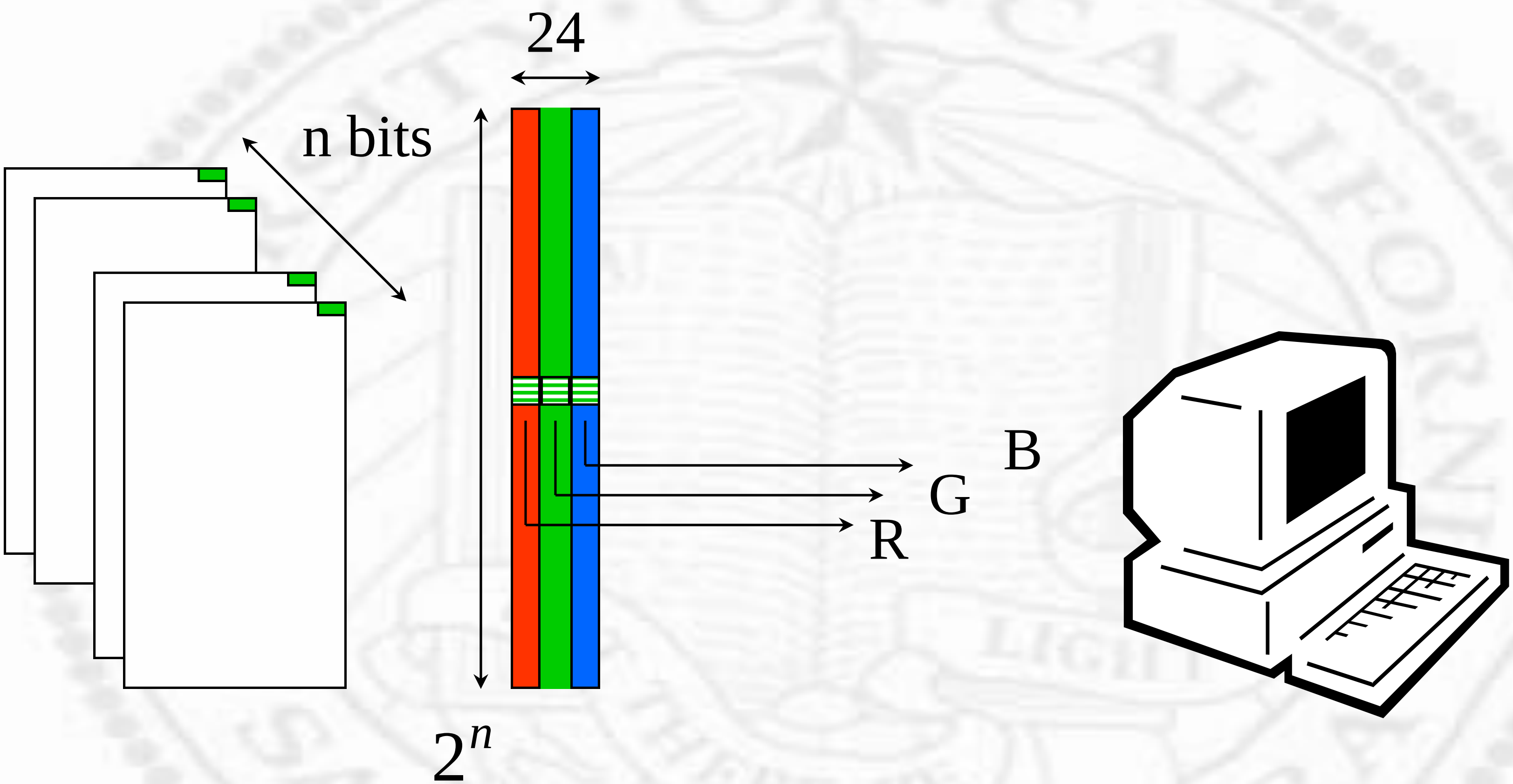
RGAB Color Mode



RGBA Color Mode

- Syntax: void glColor3f(0.0, 1.0, 0.0) or glColor4f(r, g, b, a);
  - min 0.0, max 1.0, independent of underlying hardware
- Specify object *intrinsic color* (the color when lighting was not enabled)
- Other variations: void glColor[3,4]{b,s,i,f,d,ub,us,ui} also possible

Color-Index Mode



Color-Index Mode

- Syntax: void glIndex{sifd}(TYPE c)or glIndex{sifd}v(TYPE *c);
- Specify object *intrinsic color* (the color when lighting was not enabled)
- Number of indices available is usually 2^8=256 or 2^16=65536, determined by the actual hardware (video memory in a video card and spatial resolution used)

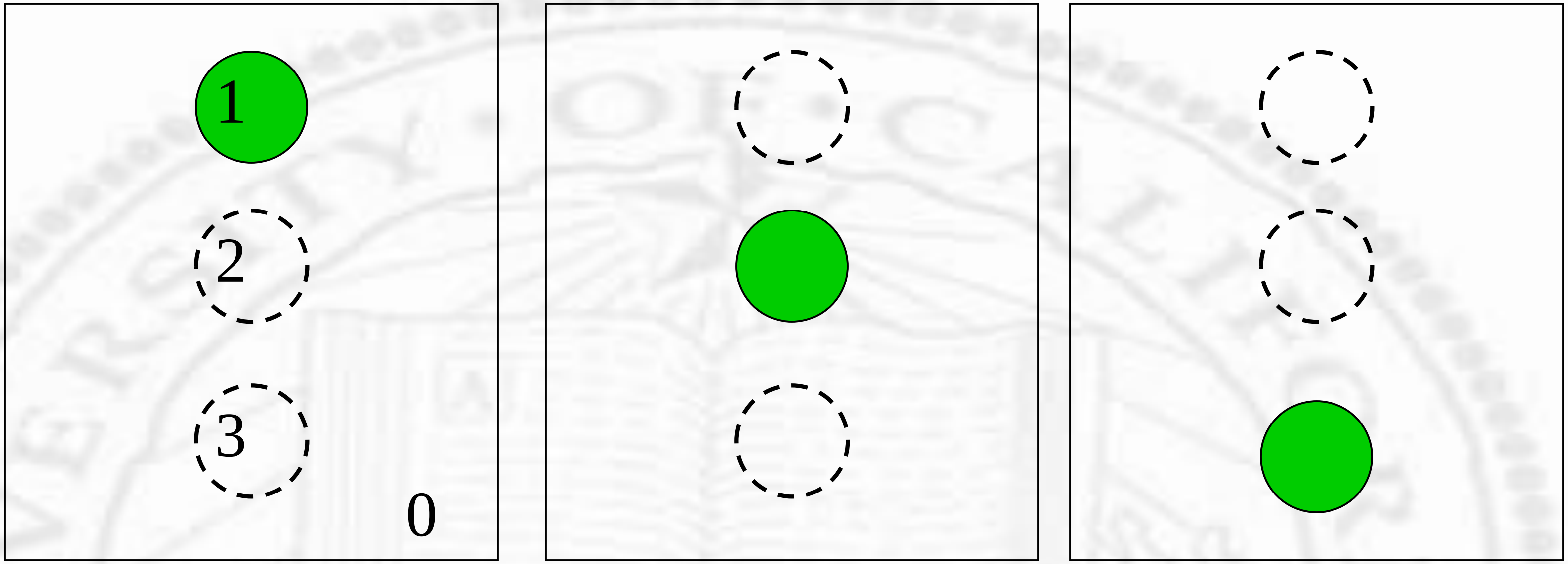
Color-Index Mode

- It might be surprising to know that OpenGL does not provide routines for manipulating color cells in a colormap
- Use aux-library void auxSetOneColor(index,r,g,b)
  - r,g,b in [0,1]

Choice btw RGBA and Index

- General rule: use RGBA
- RGBA better for shading, color, texture mapping, etc.
- For legacy application using color-index
  - Also applied to myriad of ‘less capable’ mobile devices
- Color-map can be useful for special effect such as animation

Example



0	white	white	white
1	green	white	white
2	white	green	white
3	white	white	green

Shade Model

- Void glShadeModel (GL_FLAT or GL_SMOOTH)
- Important when shade extended primitives specified by multiple vertices (e.g. polygons, triangular meshes)
- GL_FLAT: use the color of a particular vertex for the whole primitive
- GL_SMOOTH: interpolate the color of vertices
- May be troublesome in color-index mode

The "Last Mile" Issues

- Specification of color and spatial resolution was done in a device independent way
  - the device's *spatial* resolution does not matter
  - the device's *color* resolution does not matter

  To OpenGL
- In reality, the display window
  - has a fixed spatial resolution
  - a fixed color resolution (or "visual" type)
  - Printer – high spatial, low color
  - PDA, cell phone, etc. – low spatial, low color

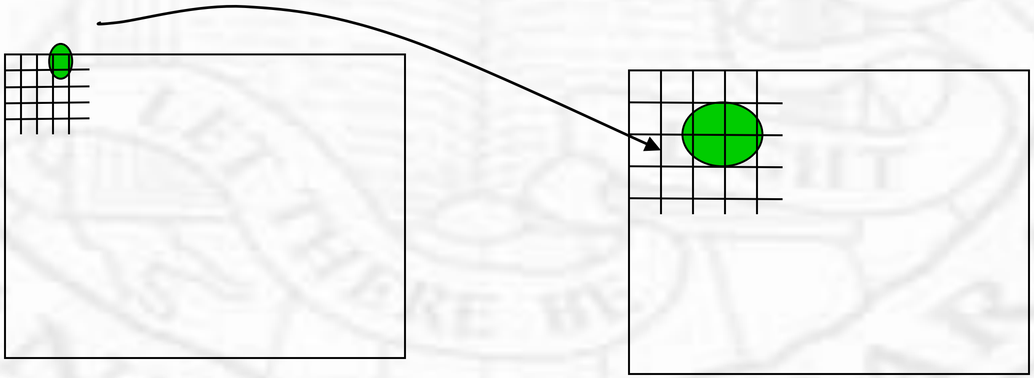
The Last Mile Issues (cont.)

- In general, taken care of by *glut* and underlying X window systems
- Not enough *spatial* resolution
  - Down sampling with interpolation
- Not enough *color* resolution
  - half toning, and more generally
  - dithering

  can be used

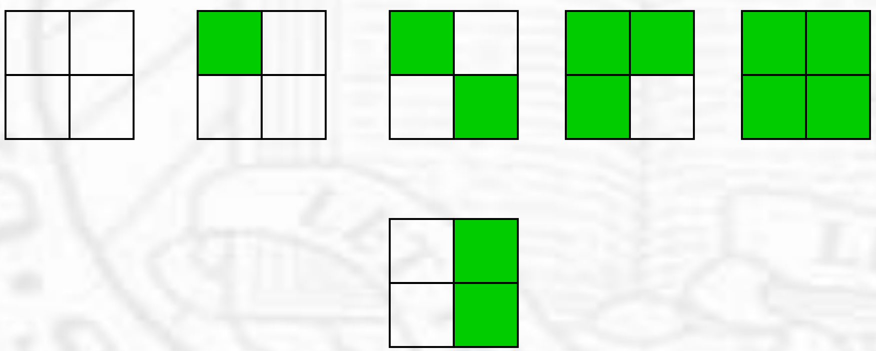
Binary Half Tone

- *High* resolution images to be produced on a low resolution device
  - e.g., a gray scale 8-bit image printed on a B/W paper
- Trade spatial resolution for color resolution



Binary Half Tone

- With a kxk square, k^2+1 intensity levels can be approximated
- Try to avoid artifacts



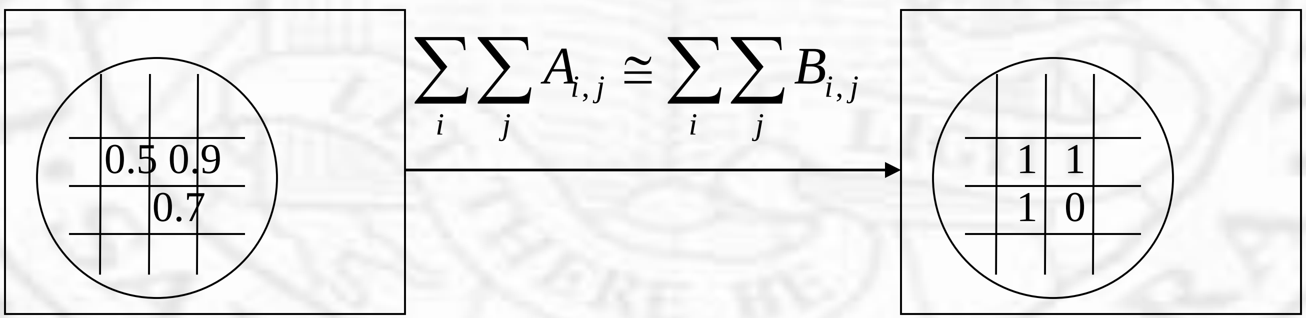
Dithering

- When trading spatial resolution for color resolution is *not* satisfactory
- Binary dithering
  - input: A[0..m-1][0..n-1] of [0..1];
  - output: B[0..m-1][0..n-1] of [0 or 1];
- Color dithering
  - input: A[0..m-1][0..n-1] of 2^m;
  - output: B[0..m-1][0..n-1] of 2^n; m>n

Binary dither

- Human perception
  - Eye integrates luminous stimuli over a solid angle of about 2 degrees
  - As long as the average intensity over a 2-deg neighborhood is similar

$$\sum_i \sum_j A_{i,j} \cong \sum_i \sum_j B_{i,j}$$



Binary Dither

- Simple thresholding

  if A(i,j)>0.5 then

      B(i,j) = 1

  else

      B(i,j)=0

- Thresholding + perturbation

  if A(i,j)+N(i,j)>0.5 then

      B(i,j) = 1

  else

      B(i,j)=0

Ordered Dither (cont.)

- An area based approach

x = i mode c, y = j mod c

If a(i,j)>D(x,y) then

    B(i,j) = 1

else

    B(i,j) = 0

D is called a magic square, filled with permutation of numbers from 0 to c^2-1

Error Diffusion

- a point-based approach

for i=0 to m-1 do

for j=0 to n-1 do

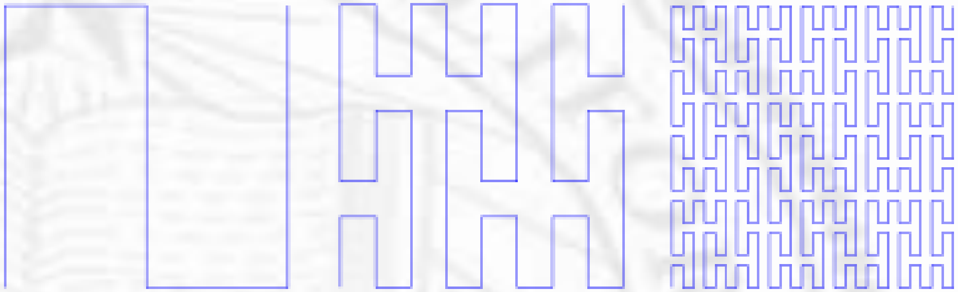
if A(i,j)>0.5 do

B(i,j) =1

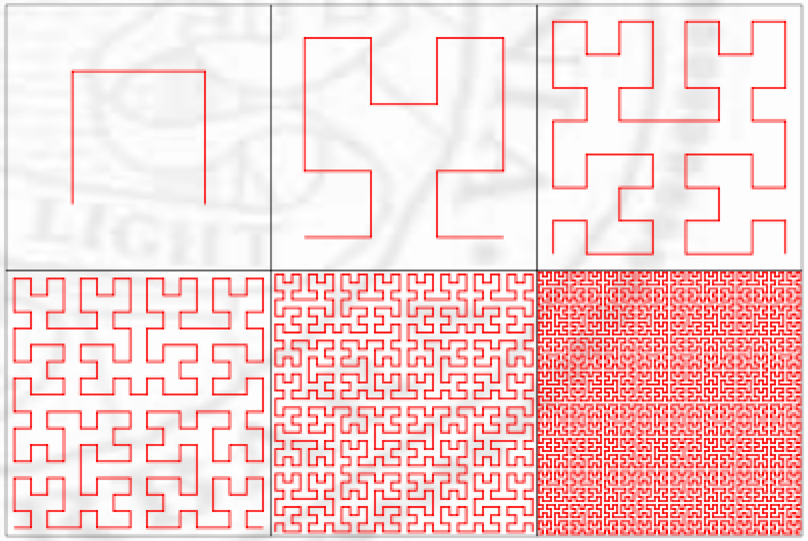
else

B(i,j) =0

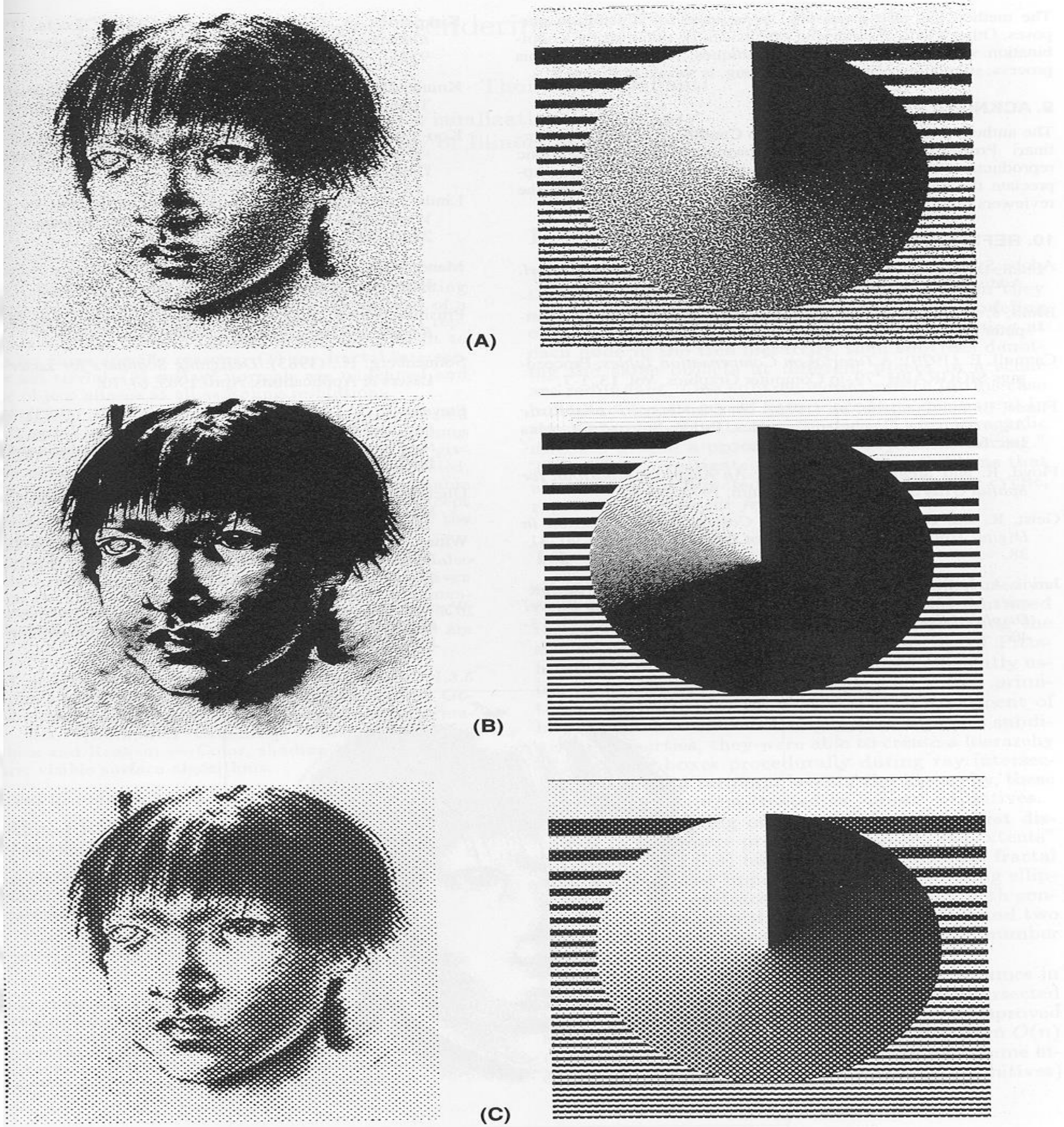
$$A(i, j+1) += \alpha \cdot (A(i, j) - B(i, j));$$
$$A(i-1, j+1) += \beta \cdot (A(i, j) - B(i, j));$$
$$A(i+1, j) += \gamma \cdot (A(i, j) - B(i, j));$$
$$A(i+1, j+1) += \delta \cdot (A(i, j) - B(i, j));$$
$$\alpha + \beta + \gamma + \delta \le 1$$



Peano curve



Hilbert curve



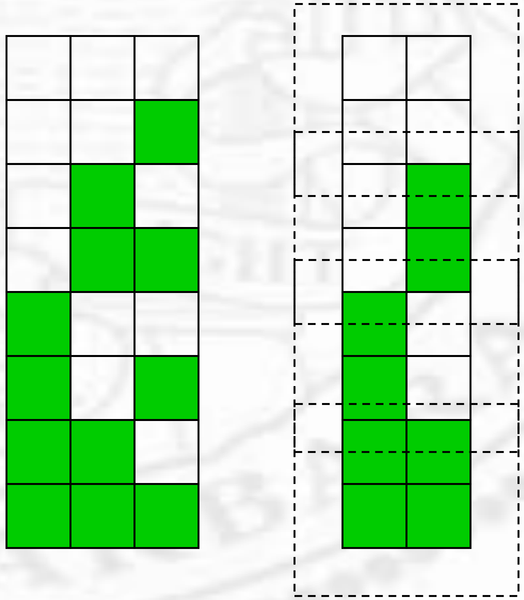
— The two test images at 75 dpi dithered with three different algorithms: (A) Space filling curve algorithm (Hilbert curve), using clusters of 32 pixels
l-Steinberg algorithm; (C) Clustered-dot ordered dither algorithm, using a matrix of order 8.

Color Dither

- Bit cut
- Median cut
- Quadtree
- etc.
  - input: A[0..m-1][0..n-1] of 2^m;
  - output: B[0..m-1][0..n-1] of 2^n; m>n

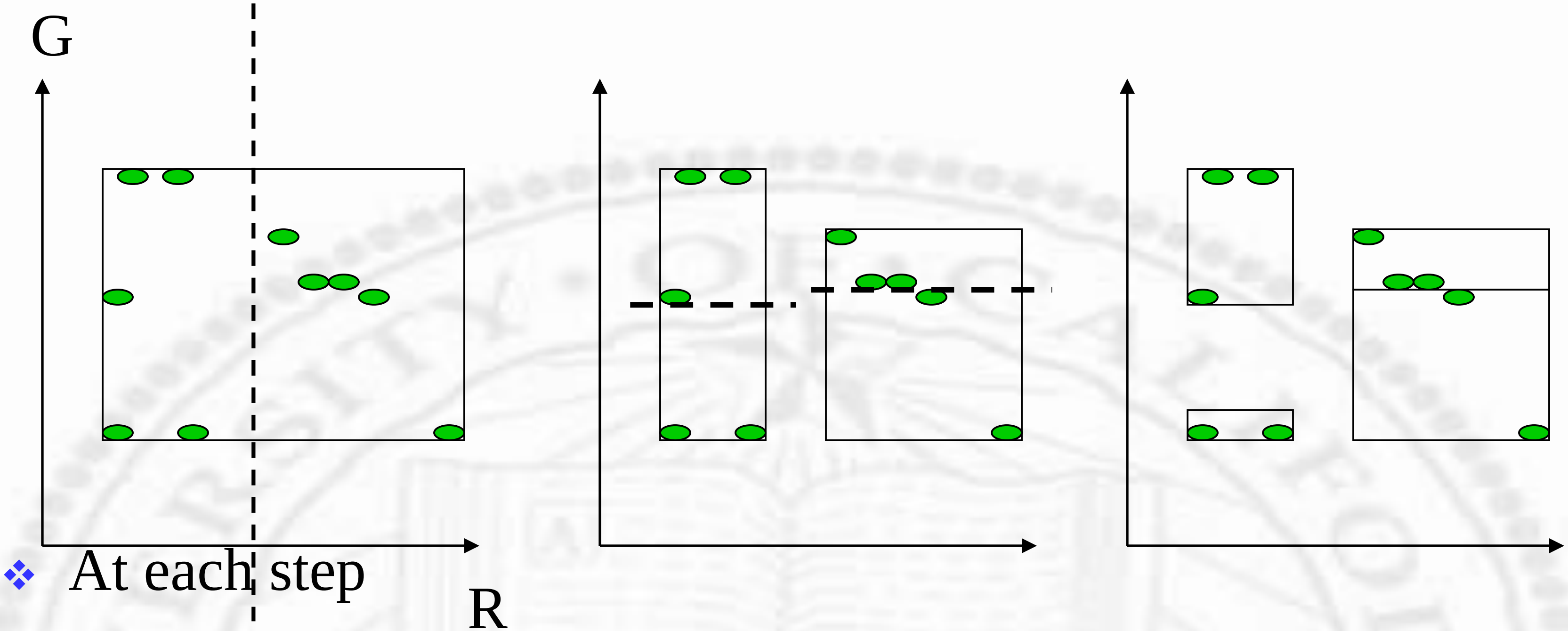
Bit cut (Uniform quantization)

- 2^m to 2^n by knocking out the lower (m-n) bits
  - do not adapt to different image contents
  - produce poor results with severe blocking and contouring effects

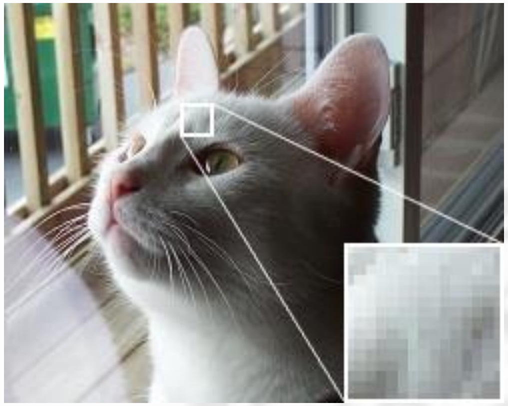


Median Cut

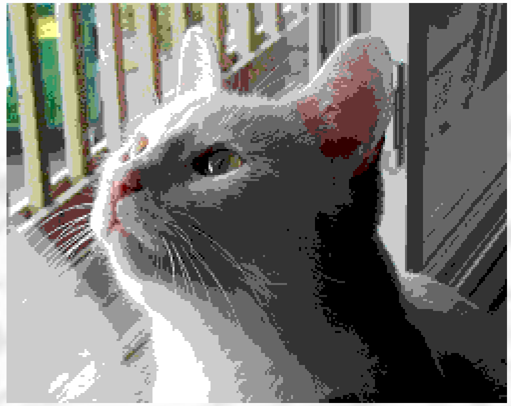
- The quantization should be adaptive depending on the image content
- Usually an image will not have pixel colors distributed uniformly over all visible spectrum
- Reserve more bits for colors which appear more frequently in an image



- At each step
  - select the axis with the largest spread
  - compute median and divide into two groups
- Recursion until C = 2^n boxes
- Use average colors in each box to build lookup table

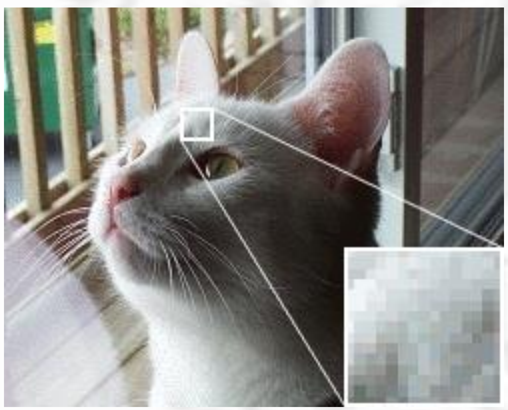


Original photo

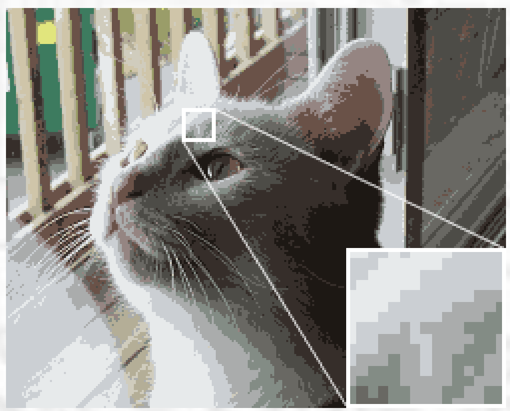


Original image using the web-safe color palette with no dithering applied. Note the large flat areas and loss of detail

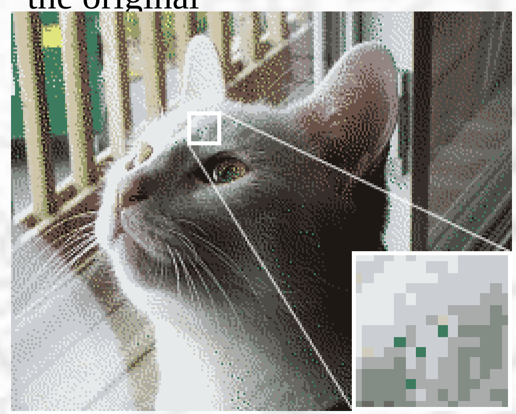
Original image using the web-safe color palette with Floyd-Steinberg dithering. Note that even though the same palette is used, the application of dithering gives a better representation of the original



Original image using the web-safe color palette with Floyd-Steinberg dithering. Note that even though the same palette is used, the application of dithering gives a better representation of the original



Depth is reduced to a 16-color optimized palette in this image, with no dithering. Colors appear muted, and color banding is pronounced



This image also uses the 16-color optimized palette, but the use of dithering helps to reduce banding.

Source: http://en.wikipedia.org/wiki/Dithering