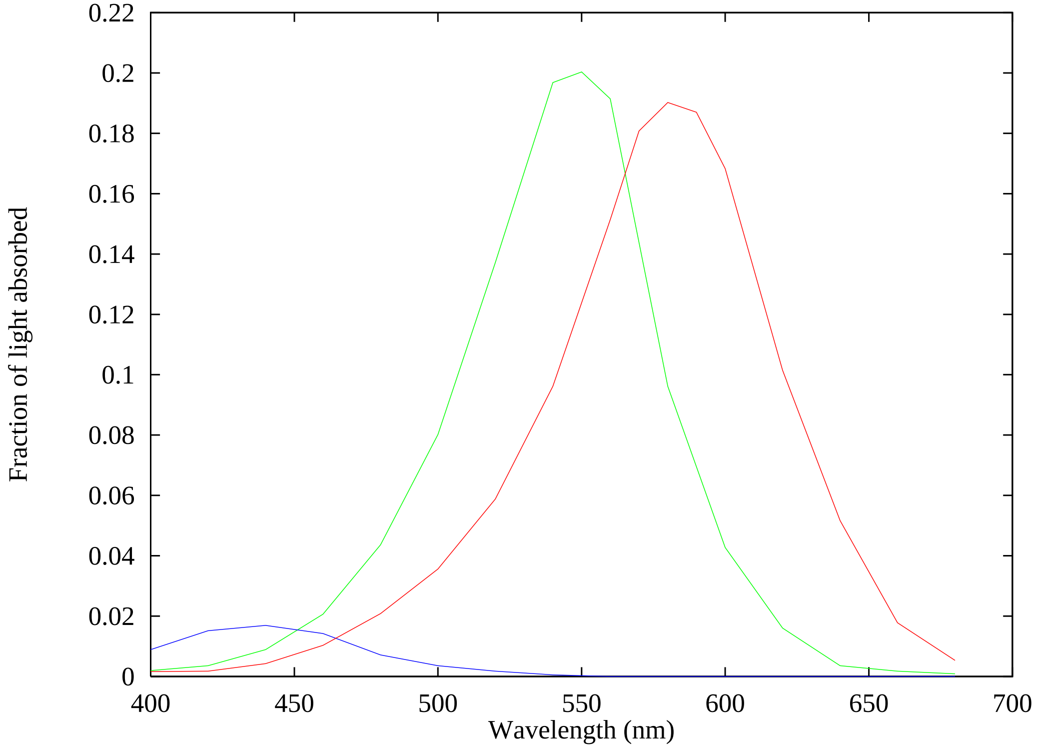
Color and Light

Displaying three-dimensional objects requires understanding color vision, light sources, and light reflection.

Colors are detected in the eye by three kinds of light-sensitive cells called **cones**. Every single wavelength triggers all three kinds of cones, but by different amounts.



Fooling the Eye

The eye is sensitive to the **ratios** of intensities of the different colors. If we shine light of wavelength 580nm (yellow), 28.6% of it will be absorbed: 19% by the red cones, 9.6% by the green cones, and none by the blue cones.

Wavelength	Red cones	Green cones	Blue cones	Description
580	0.190	0.096	0.0	Absorbance fraction
	66.4%	33.6%	0.0%	2:1:0 ratio of red to green to blue

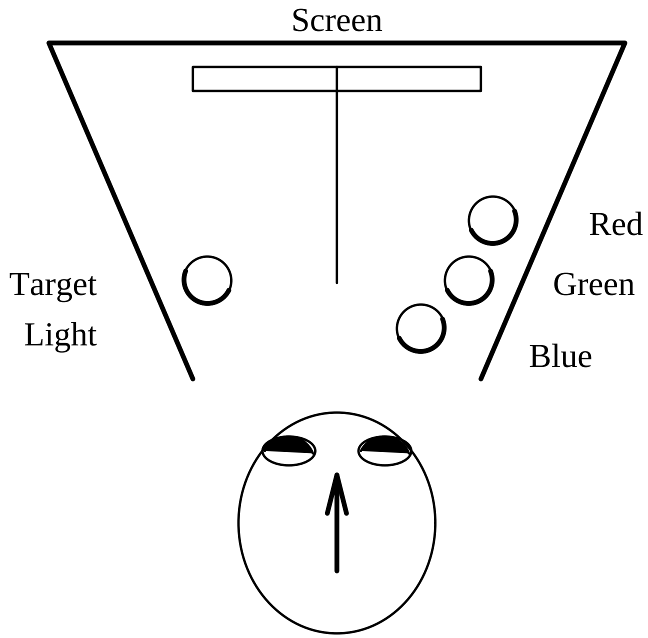
We can create the same red/green/blue cone ratio by mixing colors of other wavelengths:

Wavelength	Red cones	Green cones	Blue cones	Intensity	Description
560	0.151	0.191	0.0	0.431	greenish yellow
660	0.018	0.002	0.0	7.010	pure red

Weighted Absorbances:

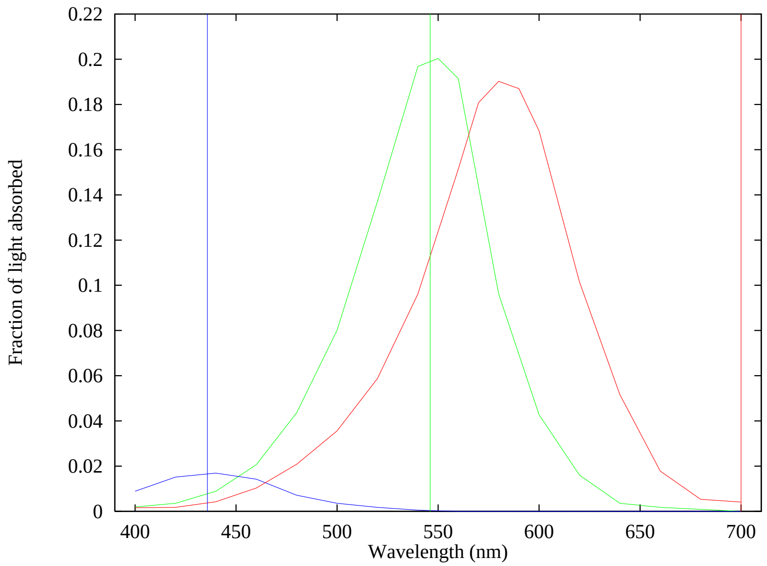
Wavelength	Red cones	Green cones	Blue cones
560	0.065	0.083	0.0
660	0.125	0.012	0.0
Sum	0.190	0.095	0.0
Percentages	66.7%	33.3%	0.0

Color Matching Experiments



Human subject adjusts intensity of red (700 nm), green (546.1 nm), and blue (435.8 nm) lights to match the color and intensity of a target light. The amounts of each color are recorded.

Some Colors Cannot be Matched by Mixing



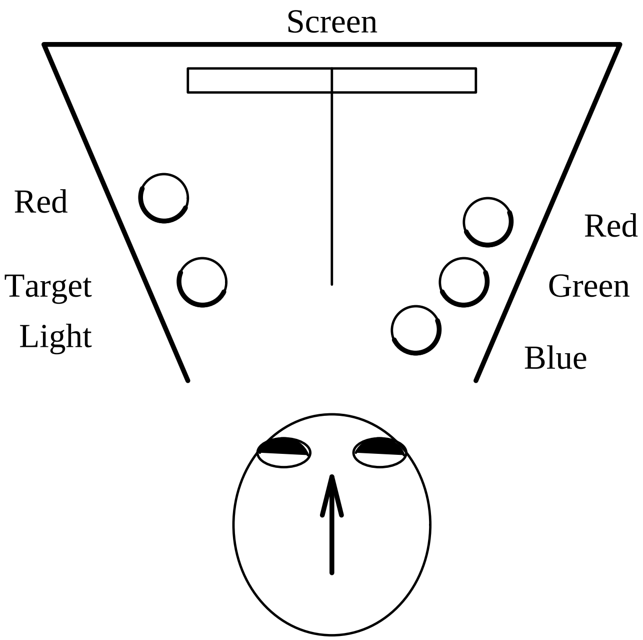
For example, at 520nm we need a ratio of red:green:blue of 30:69:1. However, our primaries have ratios of

Primary	Red intensity		Green Intensity		Blue Intensity
red (700nm)	100	:	0	:	0
green (546.1nm)	36	:	64	:	0
blue (435.8)	13	:	28	:	59

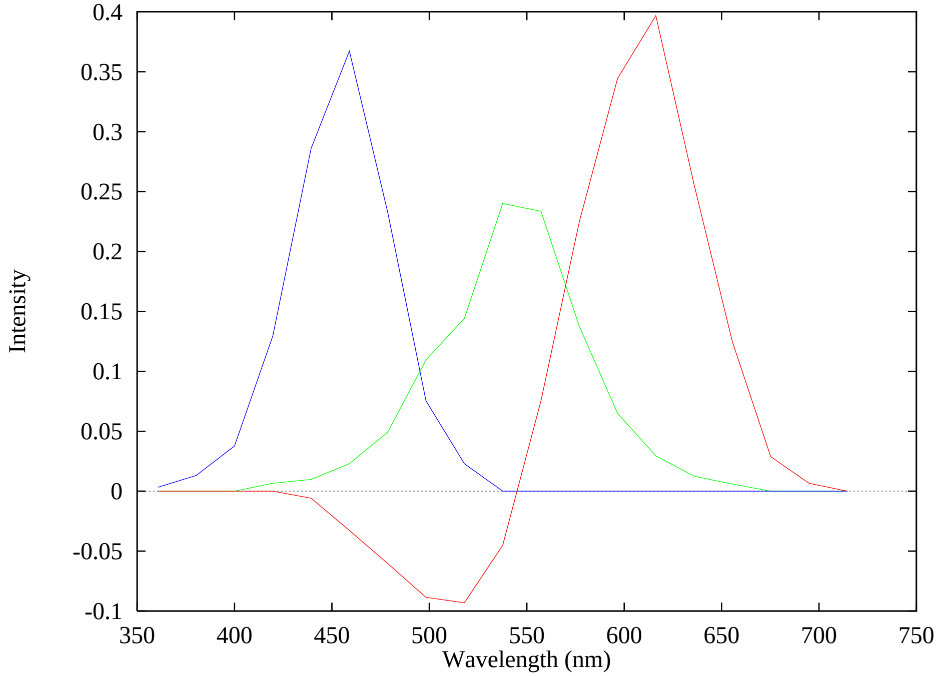
No mixtures of these will give the desired ratios. The fundamental reason is that the green primary stimulates the red cones too much, and we can't "unstimulate" them.

Color Matching by Adding Red

The target color **can** be matched if we add some red light to the target.



Color Matching Graph

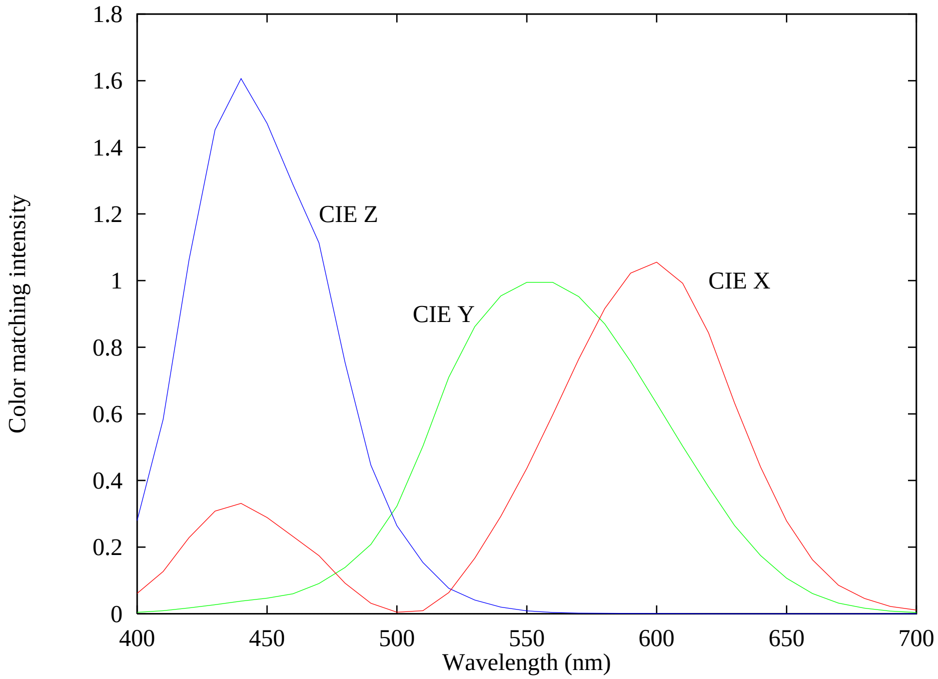


The negative amount of red shows how much red light was added to the target.

CIE Ideal Primaries

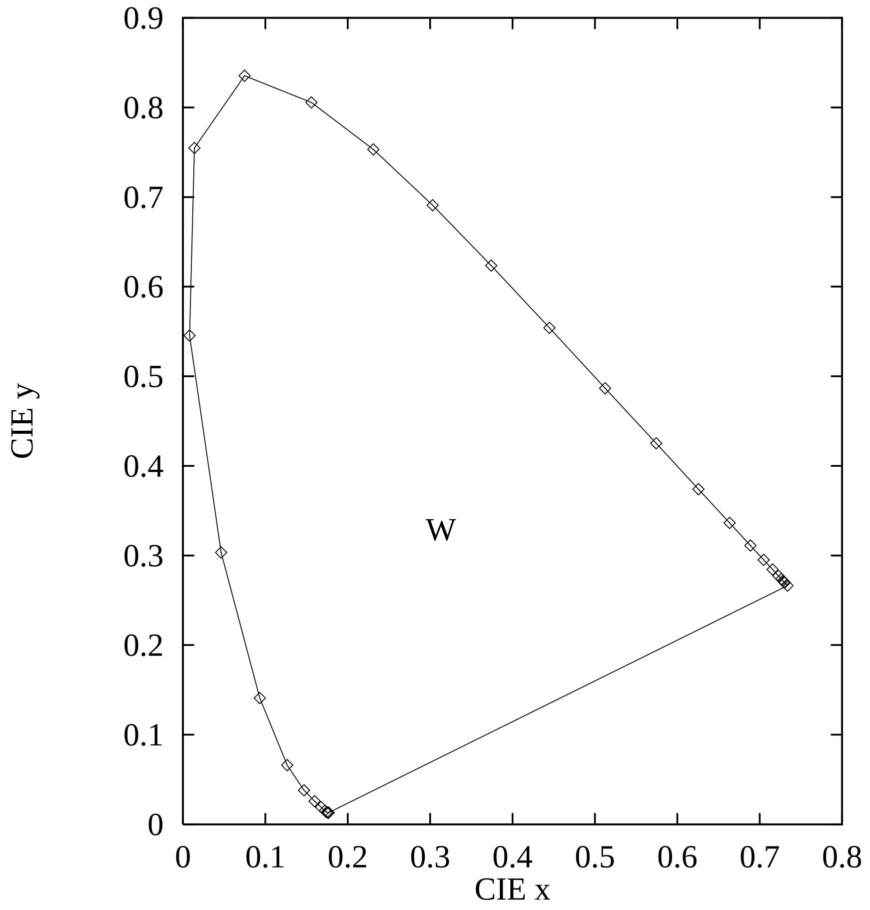
If we had three pure colors (i.e., colors that stimulated only one of the cones), we could easily produce all colors by mixing.

The CIE (Commision Internationale de l'Eclairage) defined three imaginary artificial primary colors, X, Y, and Z that can be mixed (with positive coefficients) to produce any color.



CIE Color Diagram

Every visible color (and some invisible ones too) can be mapped into a two-dimensional (x, y) space called the Chromaticity space. This captures only the color (but not intensity) of the light.



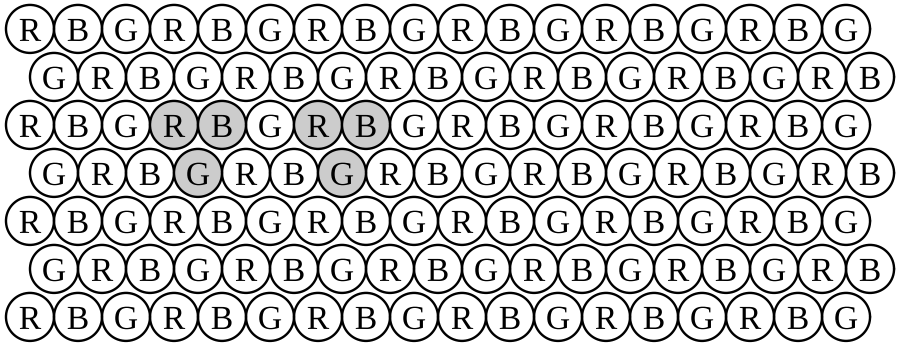
$$
\begin{aligned}
x &= X/(X+Y+Z) \\
y &= Y/(X+Y+Z) \\
z &= 1-(x+y) = Z/(X+Y+Z)
\end{aligned}
$$

Properties of the CIE Diagram

- **Spectral colors are on the edge**
- **White is near the middle**
- **Linear mixing.** Any color that results from additive mixing of two colors (x_1, y_1) and (x_2, y_2) will lie on the line segment joining these two points.
- **Color gamuts.** Any color that results from additive mixing of three colors (x_1, y_1), (x_2, y_2), and (x_3, y_3) will lie in the triangle connecting these three points.
- **Color conversion.** Switching between color systems (e.g., color monitors, printing systems) is best performed in this space.
- **Color mixing.** Mixing of colors in images should be performed in CIE space.

Color Monitors

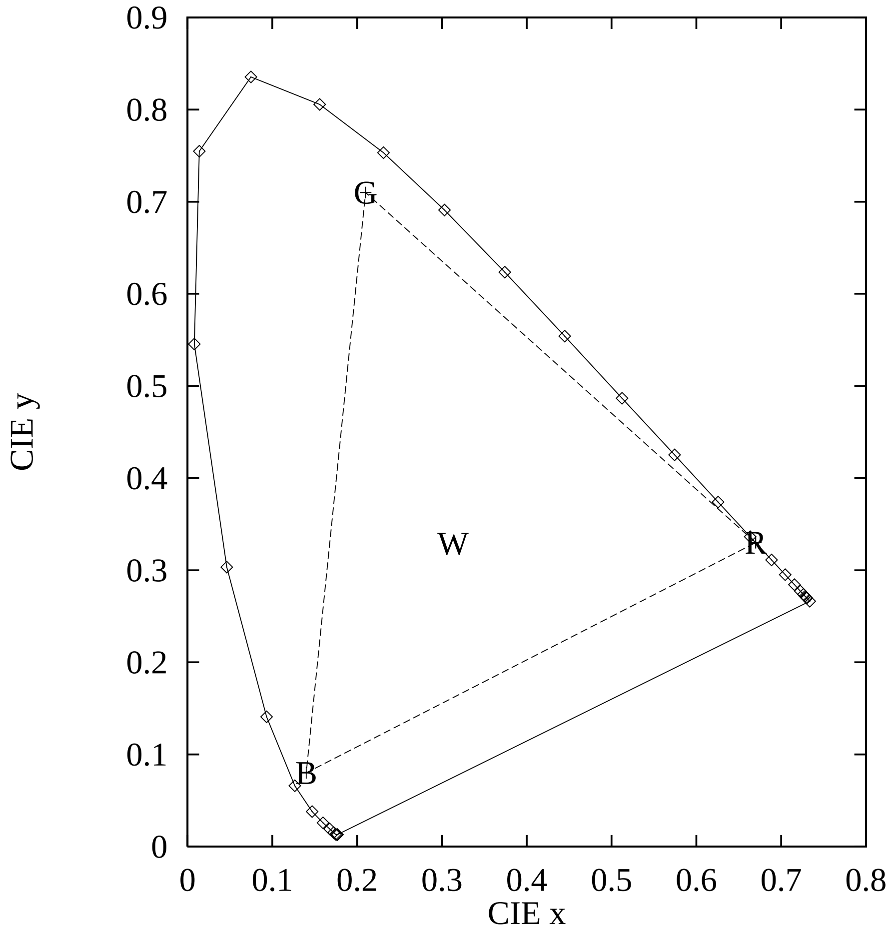
Each pixel of a color monitor is a “triad” of three phosphor dots with colors Red, Green, and Blue. The brightness of each phosphor dot can be independently controlled.



Examine your own monitor with a magnifying glass.

Color Monitor Gamut

A color monitor cannot display all perceivable colors. If we know the CIE (x, y) values for each pixel, we can compute the **gamut** of possible colors:



Color Printing and Subtractive Displays

When light hits a surface that has a red color, all wavelengths of light except red wavelengths are absorbed. Only red wavelengths are reflected.

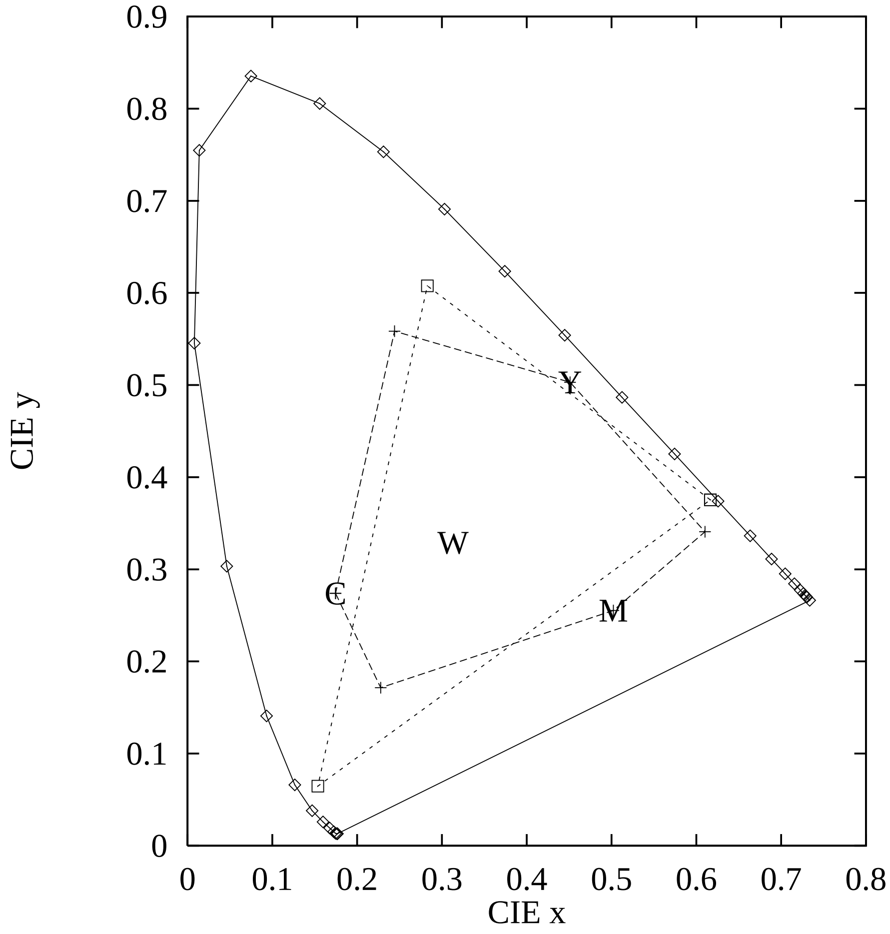
The inks used in color printing are typically Cyan, Yellow, Magenta, and Black (CYMK).

Dye Color	Absorbs Color	Reflects Colors
Cyan	Red	Blue and Green
Yellow	Blue	Green and Red
Magenta	Green	Blue and Red
Black	All	None

By mixing Cyan and Yellow, we can absorb Red and Blue so that only Green is reflected. By mixing appropriate amounts of ink, many different colors can be produced. However, most dyes interact in non-linear ways.

Color Printing Can't Print All Colors

A color printer can't print all colors a color display can display!



What the CIE Diagram Doesn't Show

The CIE diagram doesn't show the absolute luminance (total energy) of the light. However, the CIE y coordinate does provide the *relative* luminance compared to other colors in the diagram. To specify an actual color experience, we must therefore specify

- CIE x
- CIE y
- CIE Y, which is the absolute luminance value.

Converting Between Color Encodings

Given a color in (x, y, Y) coordinates, how can we convert it to intensities for RGB on a particular display or to CYMK dye intensities for a particular printer?

First, we need to convert to absolute CIE primaries:

$$X = \frac{x}{y} Y \qquad Y = Y \qquad Z = \frac{z}{y} Y$$

Next, we multiply by a matrix:

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} r_x & r_y & r_z \\ g_x & g_y & g_z \\ b_x & b_y & b_z \end{pmatrix} \cdot \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

Computing the CIE-to-RGB transformation

To compute this array, we need to know the (x, y) CIE coordinates of the Red, Green, and Blue phosphors of the monitor. We also need to know the absolute CIE coordinates of White on the monitor:

$$(x_R, y_R)$$
$$(x_G, y_G)$$
$$(x_B, y_B)$$
$$(X_W, Y_W, Z_W)$$

From $z = 1 - x - y$, we can compute (x, y, z) coordinates for Red, Green, and Blue.

Let

$$C = \begin{pmatrix} x_R & x_G & x_B \\ y_R & y_G & y_B \\ z_R & z_G & z_B \end{pmatrix}$$

Computing the CIE-to-RGB transformation (2)

Compute

$$\begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} = C^{-1} \begin{pmatrix} X_W \\ Y_W \\ Z_W \end{pmatrix}$$

Finally, compute

$$T = \begin{pmatrix} r_x & r_y & r_z \\ g_x & g_y & g_z \\ b_x & b_y & b_z \end{pmatrix} = C \cdot \begin{pmatrix} v_1 & 0 & 0 \\ 0 & v_2 & 0 \\ 0 & 0 & v_3 \end{pmatrix}$$

Converting Between RGB and CYMK

Given intensity values (R, G, B) (where each value is between 0 and 1), we can convert to CYMK using the following formulas:

$$\begin{aligned} C &= \max\{R, G, B\} - R \\ Y &= \max\{R, G, B\} - B \\ M &= \max\{R, G, B\} - G \\ K &= 1 - \max\{R, G, B\} \end{aligned}$$

This is approximate. It assumes that the printed Cyan is equal to white minus the CRT Red, and this is rarely the case. High-fidelity conversions from RGB to CYMK currently require careful tweaking to compress and stretch the RGB gamut of a particular image so that it fits into the available CYMK gamut. This is an active research area.

Other Ways of Specifying Color

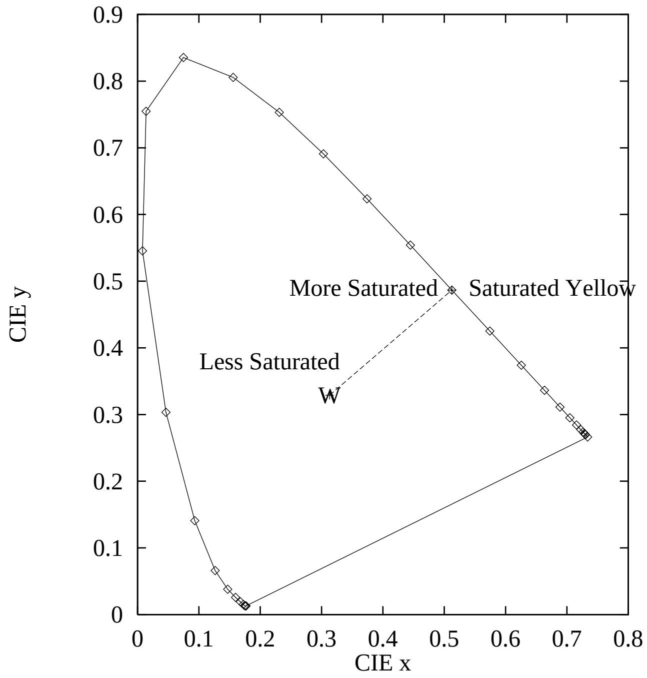
The CIE color diagram is a scientific formalism, but it doesn't provide a natural user interface for specifying colors. Several other schemes have been developed.

- **Hue, Saturation, Value**.
- **Hue, Lightness, Saturation**.

Hue, Saturation, Value

In intuitive way to describe colors is to specify their hue, saturation, and value.

- **Hue:** The dominant wavelength of the light in that color.
- **Saturation:** The purity of the color.
- **Value:** The brightness (amount of energy) of the light.

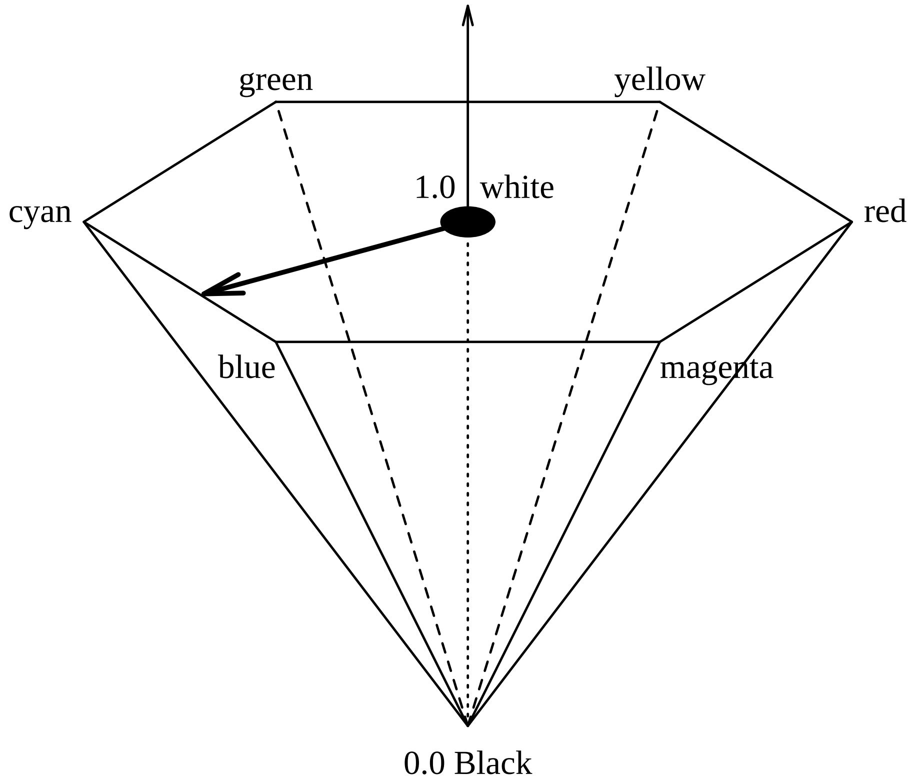


HSV Cone

Hue is measured from 0 to 360 degrees counter-clock-wise with Red = 0.

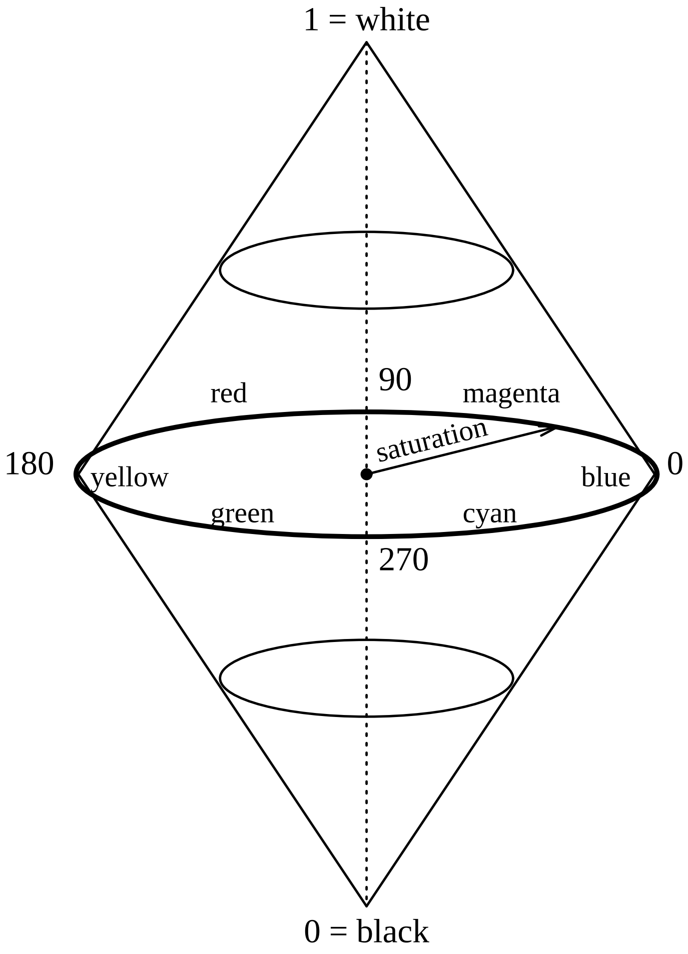
Saturation is the distance away from the center line.

Value is the vertical distance above Black.

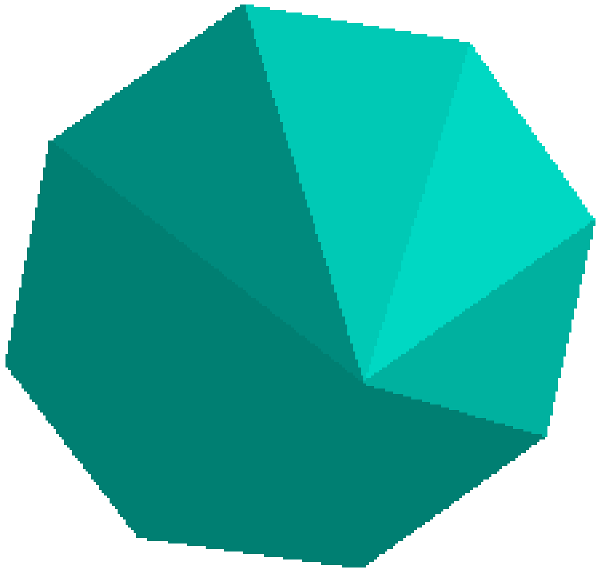


Hue, Lightness, Saturation

Developed at Tektronix. Very similar to HSV. Accounts for the fact that as light gets too bright or too dark, the range of perceivable colors narrows to only white or only black.



Illumination and Reflection



Light Sources

Emitting vs. Reflecting:

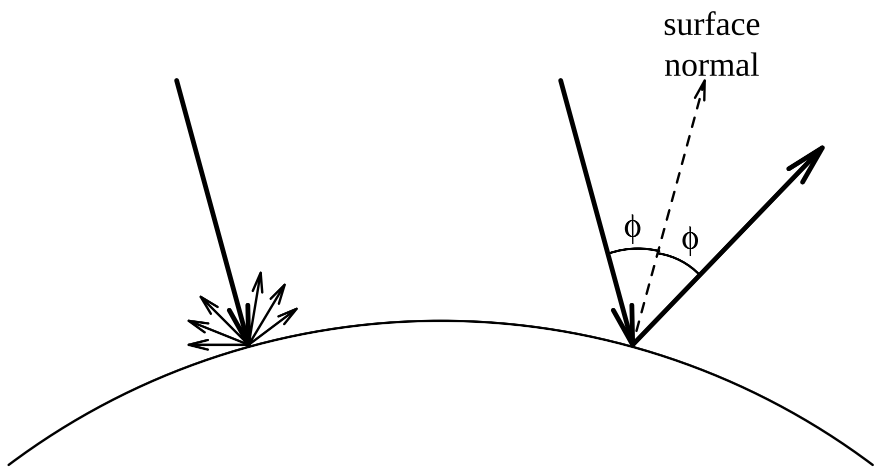
- **Light Emitting Sources**.
  Sun, light-bulbs, fluorescent lights, fire.
- **Light Reflecting Sources**.
  All illuminated surfaces: walls, floors, ceilings, other objects in scene.

Geometry:

- **Point Sources:** Sun, small light bulbs, highly-curved shiny surfaces.
- **Distributed Sources:** Fluorescent bulbs, most illuminated non-shiny surfaces.
- **Ambient Light:** Light "coming from all directions".

Reflections

- **Diffuse Reflection.** Incoming light is reflected equally in all directions.
- **Specular Reflection.** Incoming light is reflected only in one direction (with angle of incidence = angle of reflection).



Type of reflection depends on surface properties of the object.

We will find it very useful to compute a unit surface normal vector N.

Reflection of Ambient Light

We will model the properties of an object by a series of constants for diffuse and specular reflection. We will only worry about reflectivity of the primary colors (this is a major approximation!).

k_{dr} = diffuse reflectivity for red.
k_{dg} = diffuse reflectivity for green.
k_{db} = diffuse reflectivity for blue.

Each of these is a constant between 0 and 1. It tells what percentage of the light is reflected.

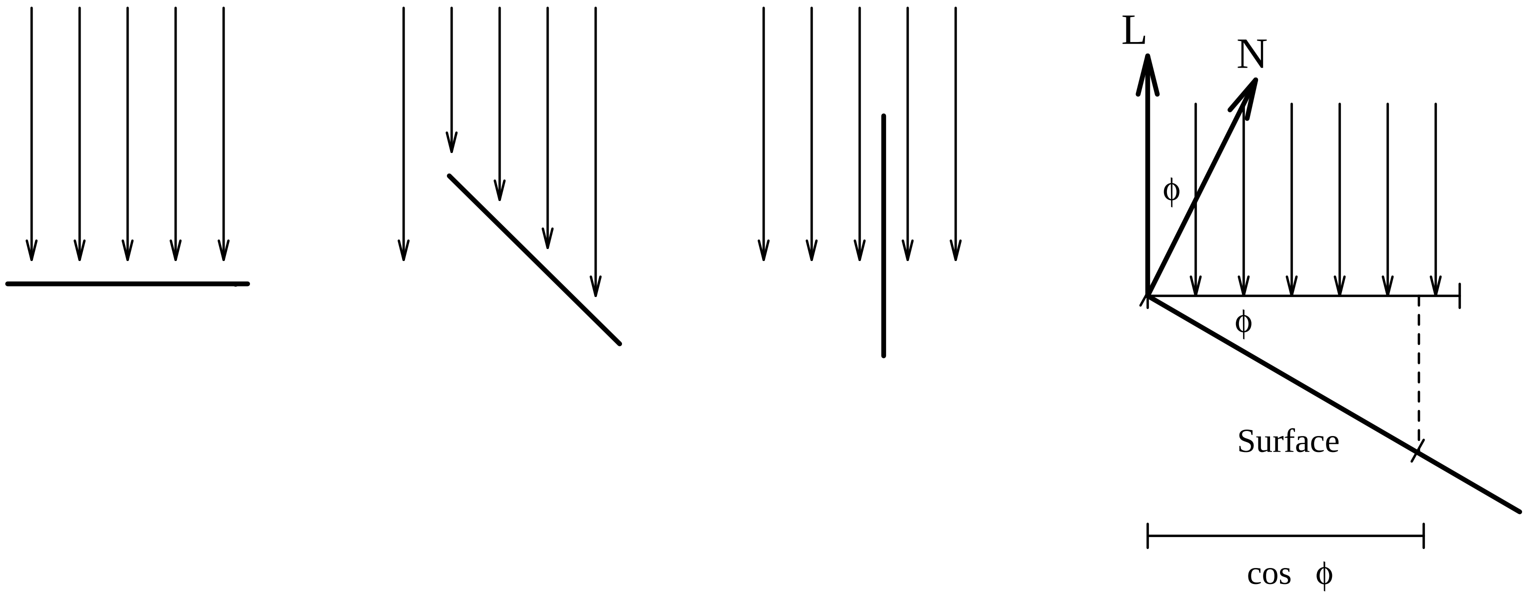
Let (I_{ar}, I_{ag}, I_{ab}) be the intensity of red, green, and blue ambient light (also specified between 0 and 1). Then the reflected light will have R,G,B intensities of

$$I_r = k_{dr} I_{ar} \qquad I_g = k_{dg} I_{ag} \qquad I_b = k_{db} I_{ab}$$

Diffuse Reflection ("Lambertian")

Assumptions: Arriving rays are parallel; Light attenuates as $\frac{1}{d}$

Amount of light hitting a unit surface area is proportional to the cosine of the angle of incidence.



If a point source at distance d has intensities (I_{pr}, I_{pg}, I_{pb}), then the diffuse reflected light has intensity

$$I_r = k_{dr}\frac{I_{pr}}{d}\cos\phi \qquad I_g = k_{dg}\frac{I_{pg}}{d}\cos\phi \qquad I_b = k_{db}\frac{I_{pb}}{d}\cos\phi$$

Specular Reflection

A point source reflected in a perfect mirror remains a point source.

However, we don't have perfect sources or perfect mirrors.

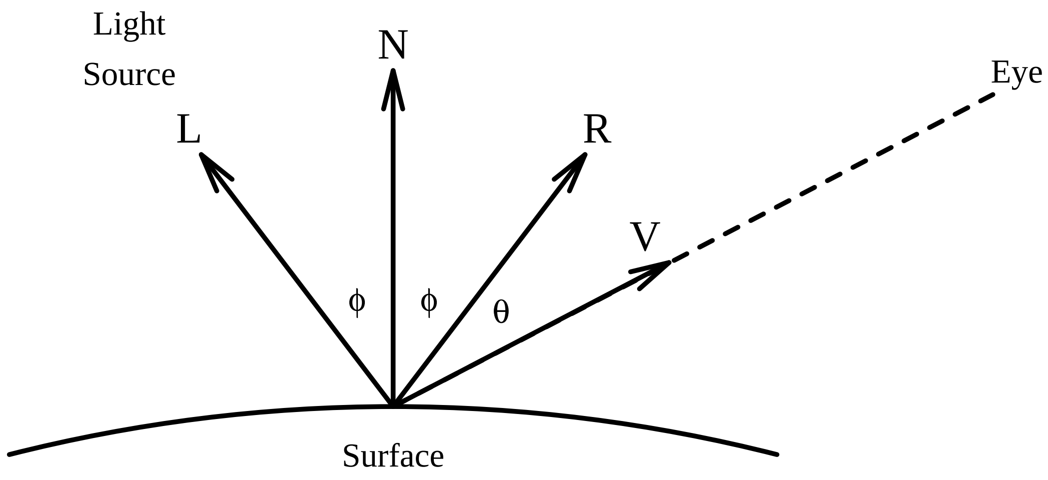
We seek a method that can give us "high lights".

We will describe a "shiny-ness" of a surface by two values:

n: Very shiny surfaces have high values of n (e.g., 200).

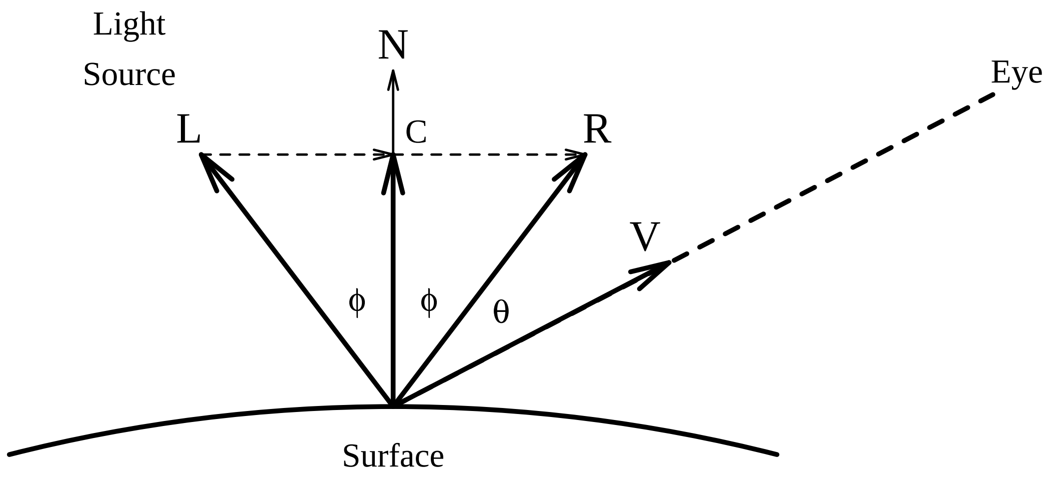
k_s: This tells how much of the incident light is reflected.

Phong's Rule



$$I_r = I_{pr} k_s (\cos\theta)^n \qquad I_g = I_{pg} k_s (\cos\theta)^n \qquad I_b = I_{pb} k_s (\cos\theta)^n$$

Computing R



The vector C can be computed as $(L \cdot N)N$.

The vector connecting L to C is $C - L$.

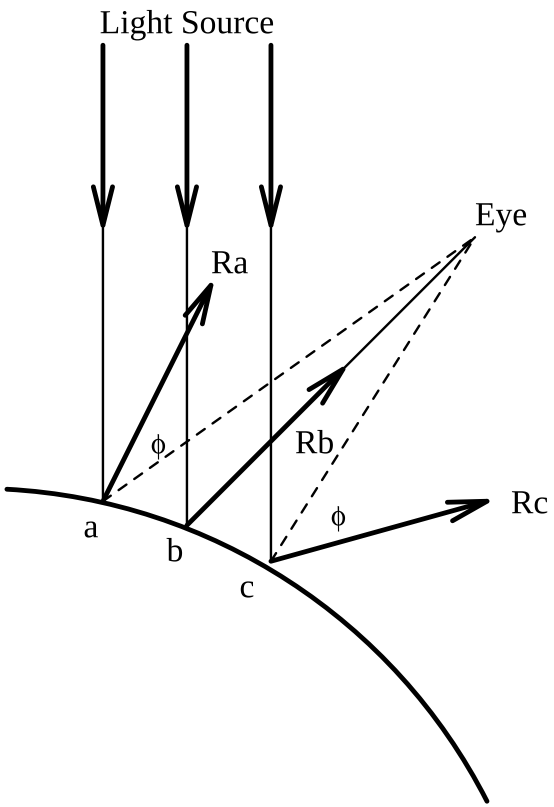
$$
\begin{aligned}
R &= L + 2(C - L) \\
&= L + 2[(L \cdot N)N - L] \\
&= L + 2(L \cdot N)N - 2L \\
&= 2(L \cdot N)N - L
\end{aligned}
$$

$$
\begin{aligned}
cos(\phi) &= L \cdot N \\
cos(\theta) &= R \cdot V
\end{aligned}
$$

Combined Reflection Model

$$
\begin{aligned}
I_r &= I_{ar}k_{df} + \frac{I_{pr}}{d}(k_{df}(L \cdot N) + k_s(R \cdot V)^n) \\
I_g &= I_{ag}k_{df} + \frac{I_{pg}}{d}(k_{df}(L \cdot N) + k_s(R \cdot V)^n) \\
I_b &= I_{ab}k_{df} + \frac{I_{pb}}{d}(k_{df}(L \cdot N) + k_s(R \cdot V)^n)
\end{aligned}
$$

How a Soft Highlight Results



Displaying a 3D scene

```
void scene3_t::solid() const
{
  int loc = 0 ;                          // lookup table location
  for (prim_t* p = prim; p; p = p→givnext())
    ((tmesh_t*)p)→setcolors(this,loc);
  scene_t::solid();
}

void tmesh_t::setcolors(const scene_t* s,int& loc) const
{
  for (int i = 0; i <facnum; i++)
    if (facet[i].givp().frontface(              // frontface test
          point3_t(s→givwv().givcen(),−s→givwv().givd())))
      facet[i].setcolor(((scene3_t*)s)→givlight(),rgb,loc);
}
```

Setting `triangle_t` Facet Color

```
void triangle_t::setcolor(light_t* light,const rgb_t& meshrgb,int& loc)
{
  rgb_t amb = light→givrgb() ;                    //  ambient(r,g,b)
  rgb_t ptl = light→givnext()→givrgb();       // point(r,g,b)
  point3_t dir = light→givnext()→givpos().normalize() ;       // L
  double dot = dir*p.givnorm() ;                  // N
  if (dot < 0) dot = 0;
  rgb = (meshrgb*(amb + dot*ptl)).trim_to_1() ;
  loc++;                                         // make lookup table entry
  setlook(int(rgb.givr()*255), int(rgb.givg()*255), int(rgb.givb()*255), loc);
  lutloc = loc;
}
```

$$
\begin{aligned}
I_r &= k_{dfr}\,(I_{ar} + I_{pr} L \cdot N) \\
I_g &= k_{dfg}\,(I_{ag} + I_{pg} L \cdot N) \\
I_b &= k_{dfb}\,(I_{ab} + I_{pb} L \cdot N)
\end{aligned}
$$

No highlights, no distance effects, over-writes colors $1 \ldots 255$.

Displaying the Faces

```
void tmesh_t::solid(const scene_t* s) const
{
  for (int i = 0; i < facnum; i++)
    if (facet[i].givp().frontface(                    // frontface test
           point3_t(s→givwv().givcen(),−s→givwv().givd())))
      facet[i].solid(s);
}

void triangle_t::solid(const scene_t* s) const
{
  col(lutloc);                        // set lut color number
  convex_t tri ;                      // make a 2d polygon
  tri += *vertex[v[0]];
  tri += *vertex[v[1]];
  tri += *vertex[v[2]];
  tri.solid(s);                       // and fill it
}
```