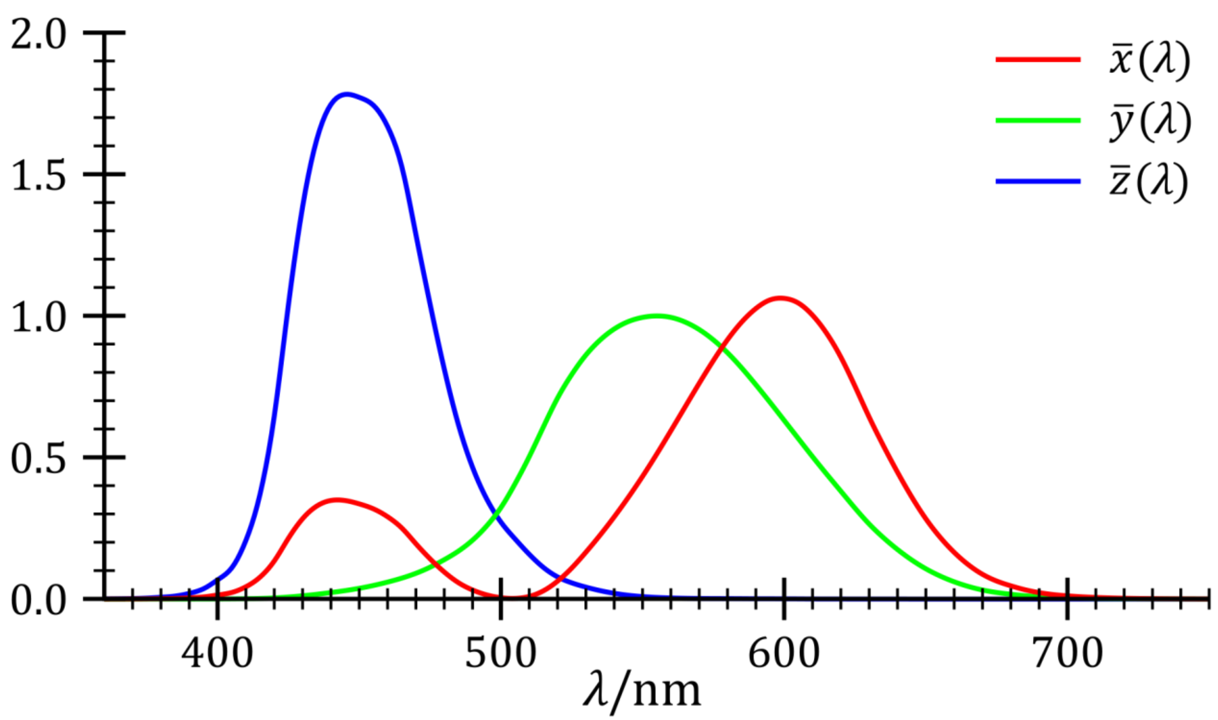
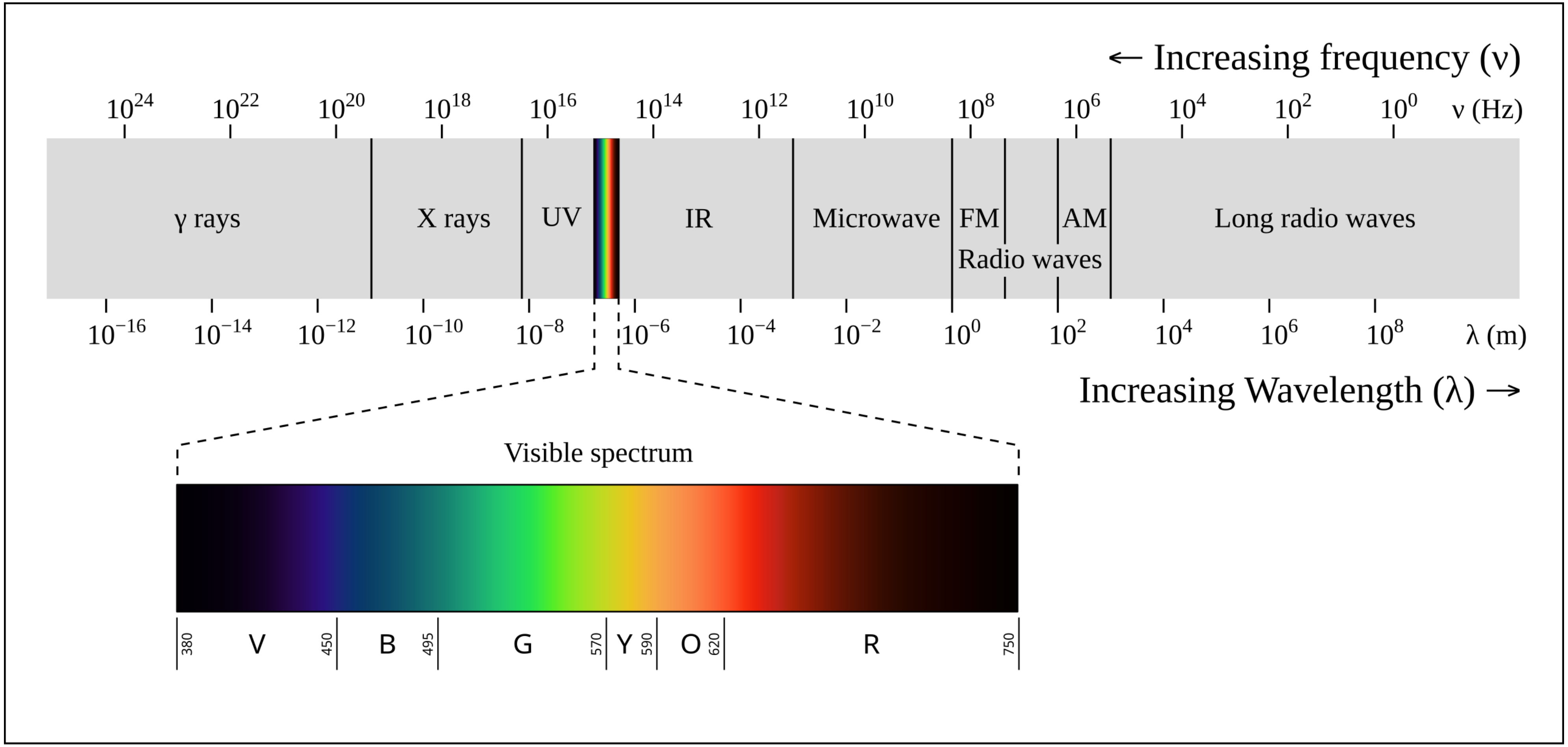


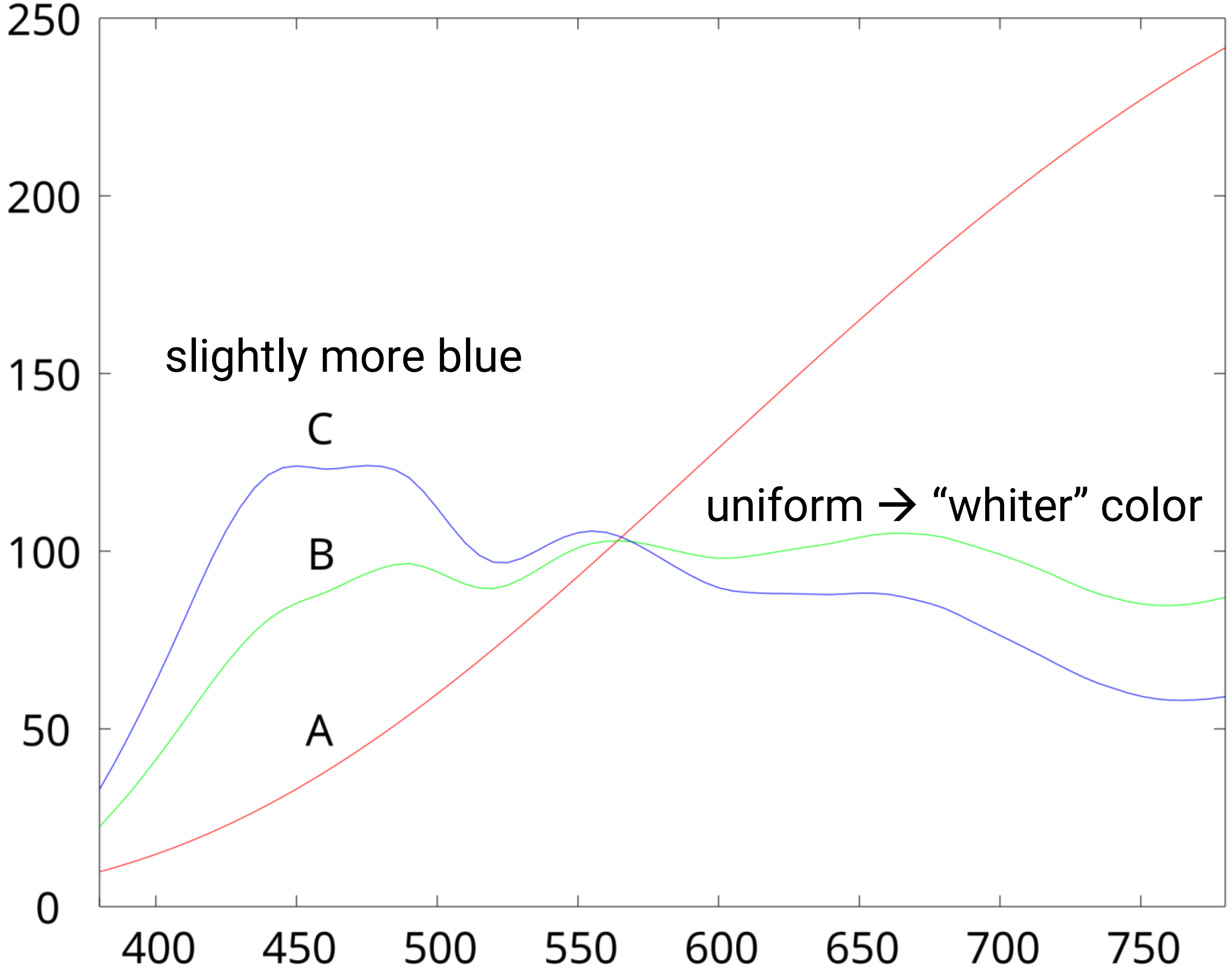
Color and HDR

Computer Graphics 24/25 – Lecture 4

Light is electromagnetic radiation; wavelength determines color



Light sources emit different wavelength distributions



more reds → "warmer" color

Here: CIE standard illuminants for:

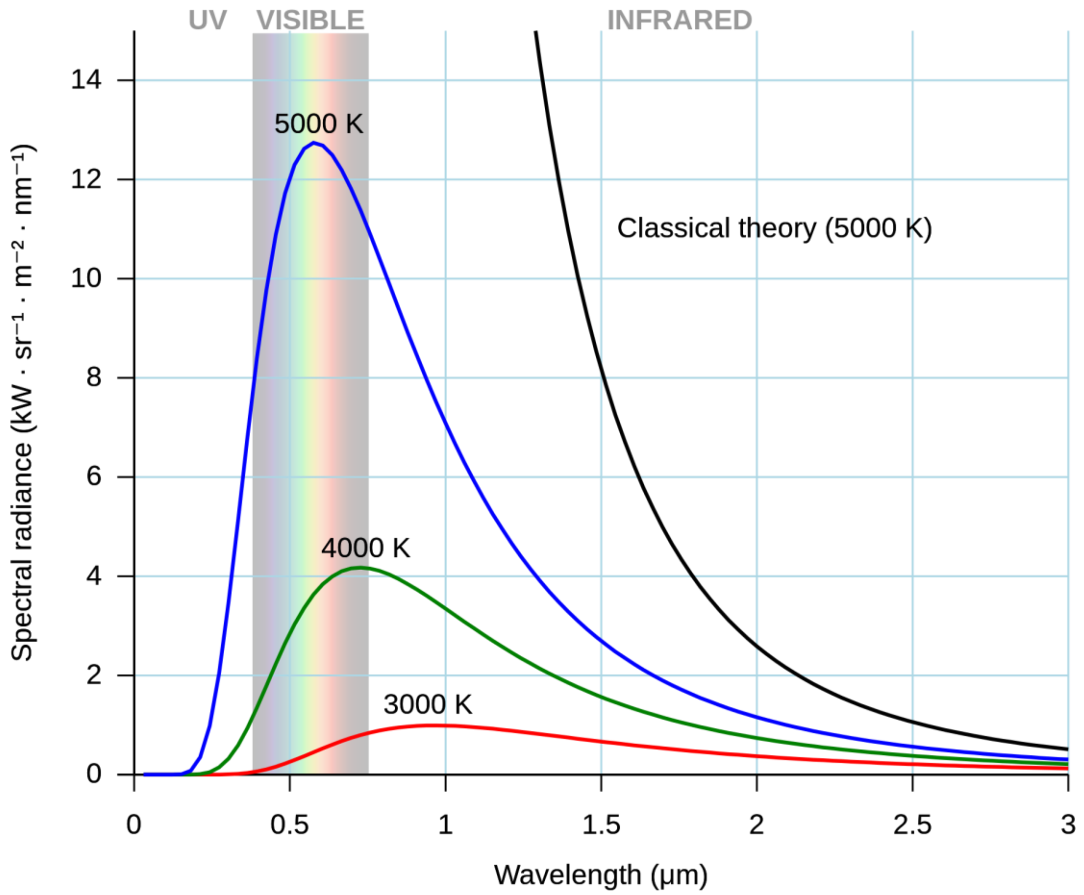
- Average incandescent light (A)
- Direct sunlight (B)
- Average daylight (C)

Black body emitters

- Idealized physical body:
  - Absorbs all light
  - In thermal equilibrium (at a constant temperature)
  - Radiates according to Planck's law:

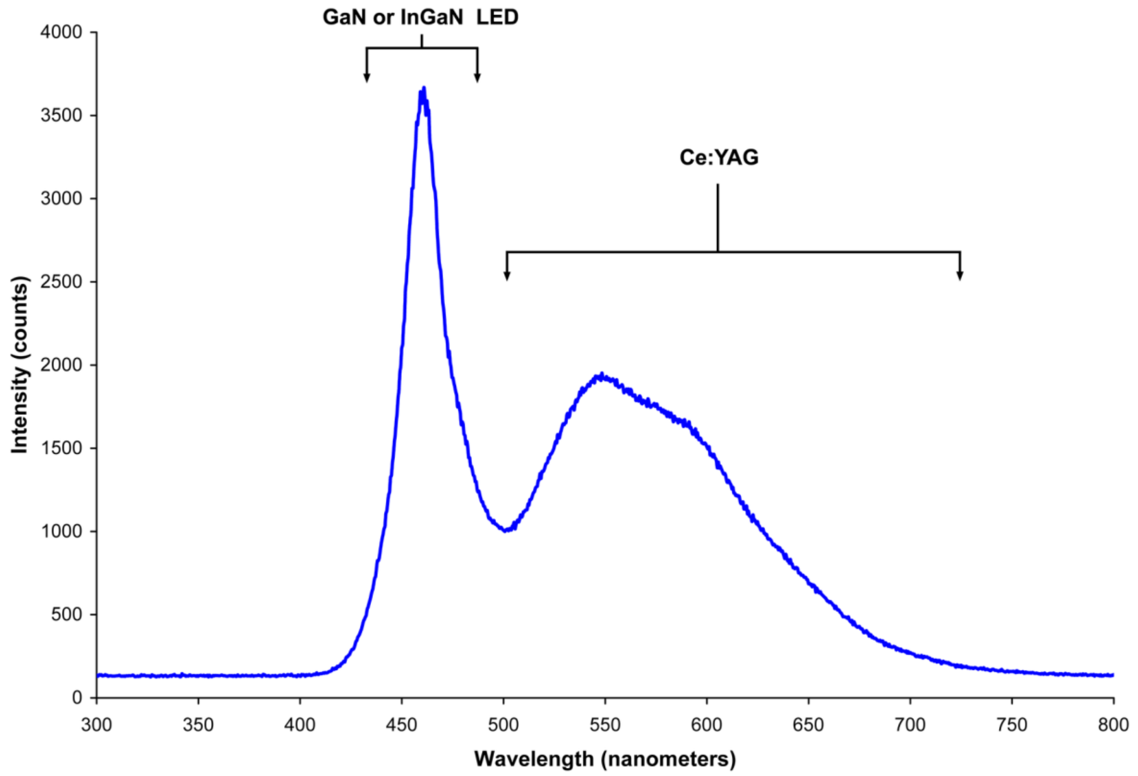
$$L_\lambda = \frac{2hv^3}{c^2} \frac{1}{\exp\left(\frac{hv}{k_B T}\right) - 1}$$

    - frequency $v = \lambda^{-1}$
    - Planck's constant h (relates frequency to energy)
    - Boltzmann's constant k_B (relates temperature to energy)
- We can use that to model lights
  - Single parameter: temperature T in Kelvin
  - Determines entire spectrum of emission



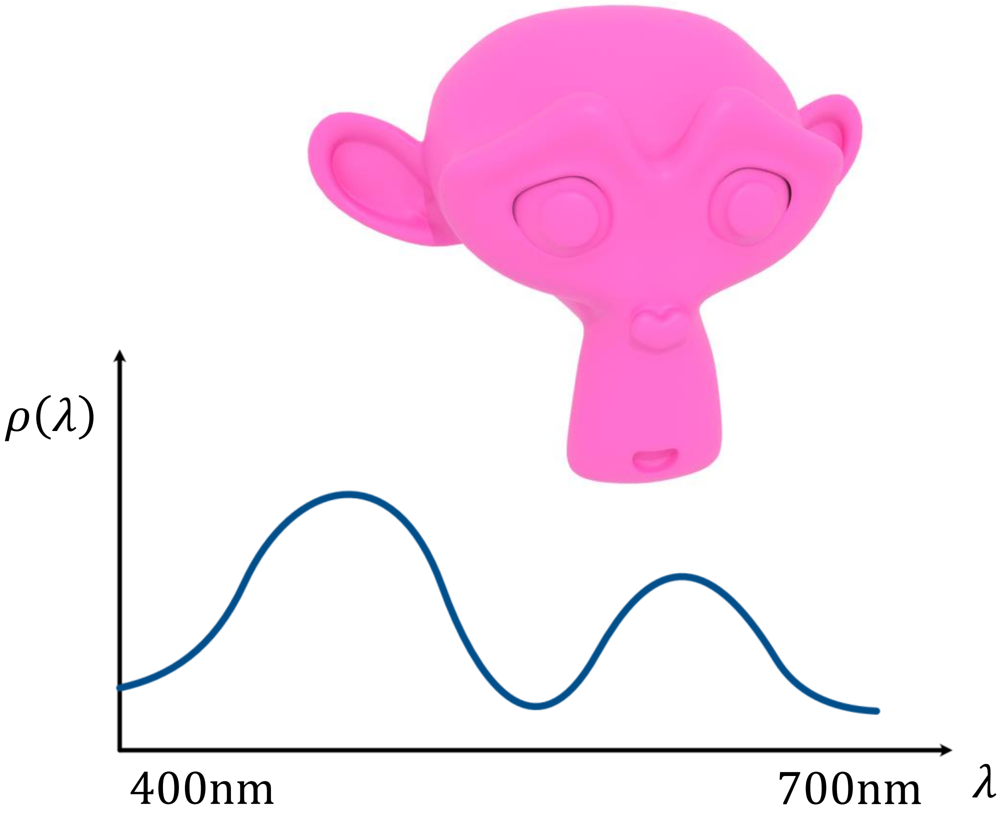
Emission spectra can be much more complex

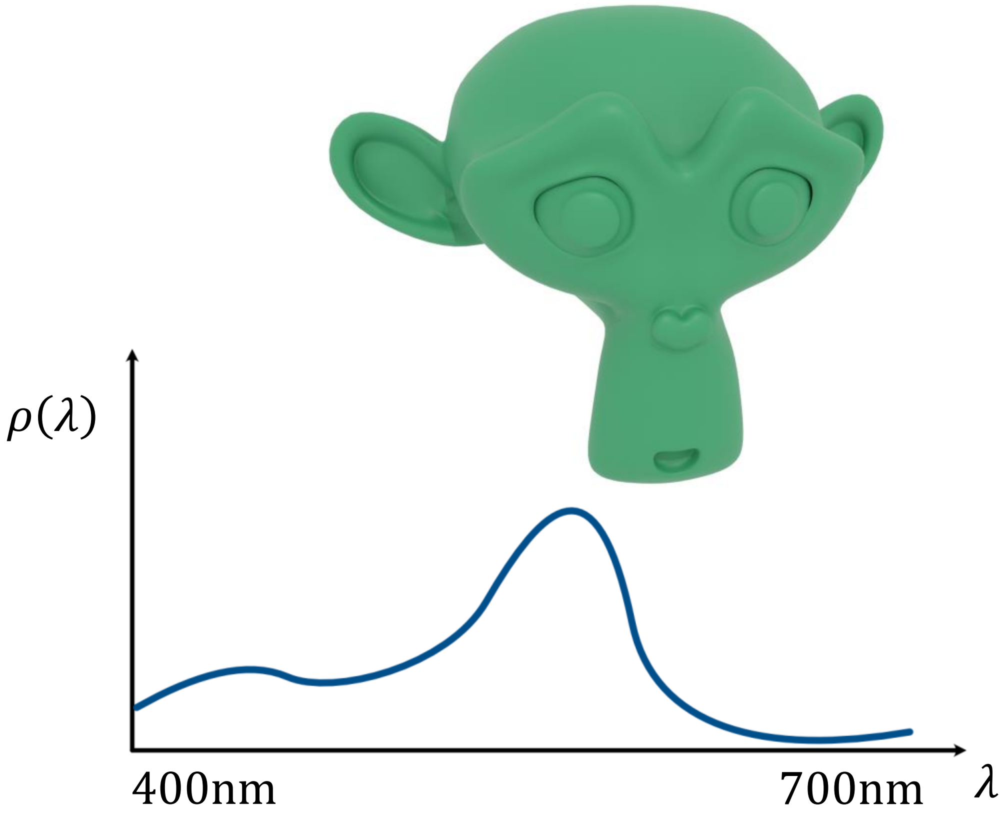
- Example: phosphor-based white LED
  - Blue LED coated with phosphors



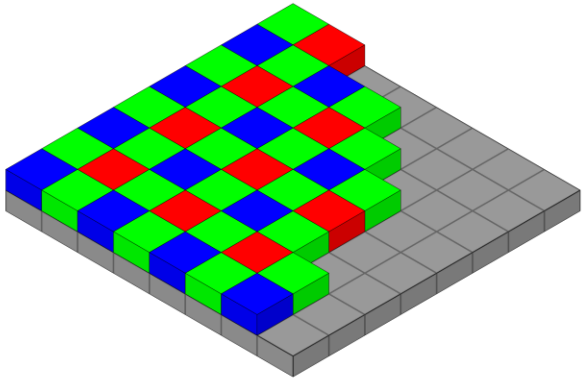
Reflectance depends on the wavelength

Example: Wavelength-dependent diffuse BRDF: $f_r(x, \omega_i, \omega_o, \lambda) = \frac{\rho(\lambda)}{\pi}$



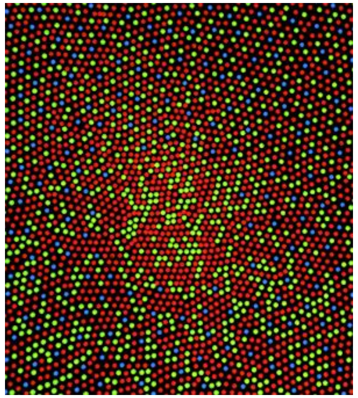


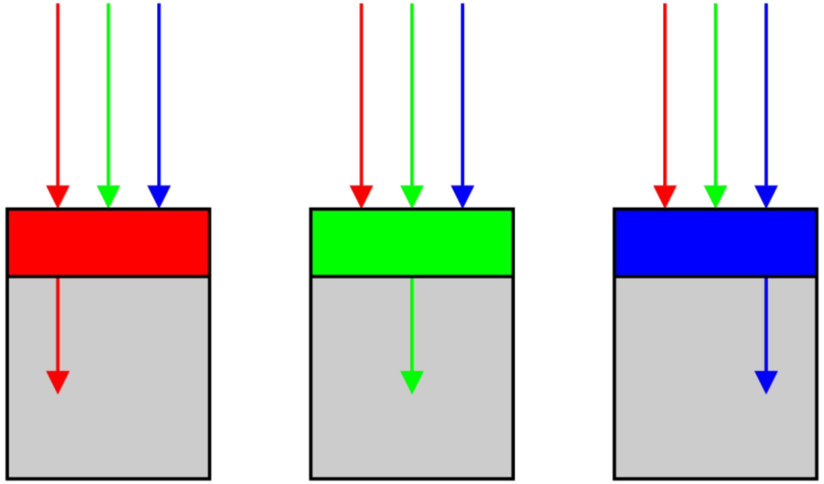
Sensors in cameras (and eyes) measure total *filtered* flux



Bayer arrangement used by cameras

Photoreceptors on the human retina





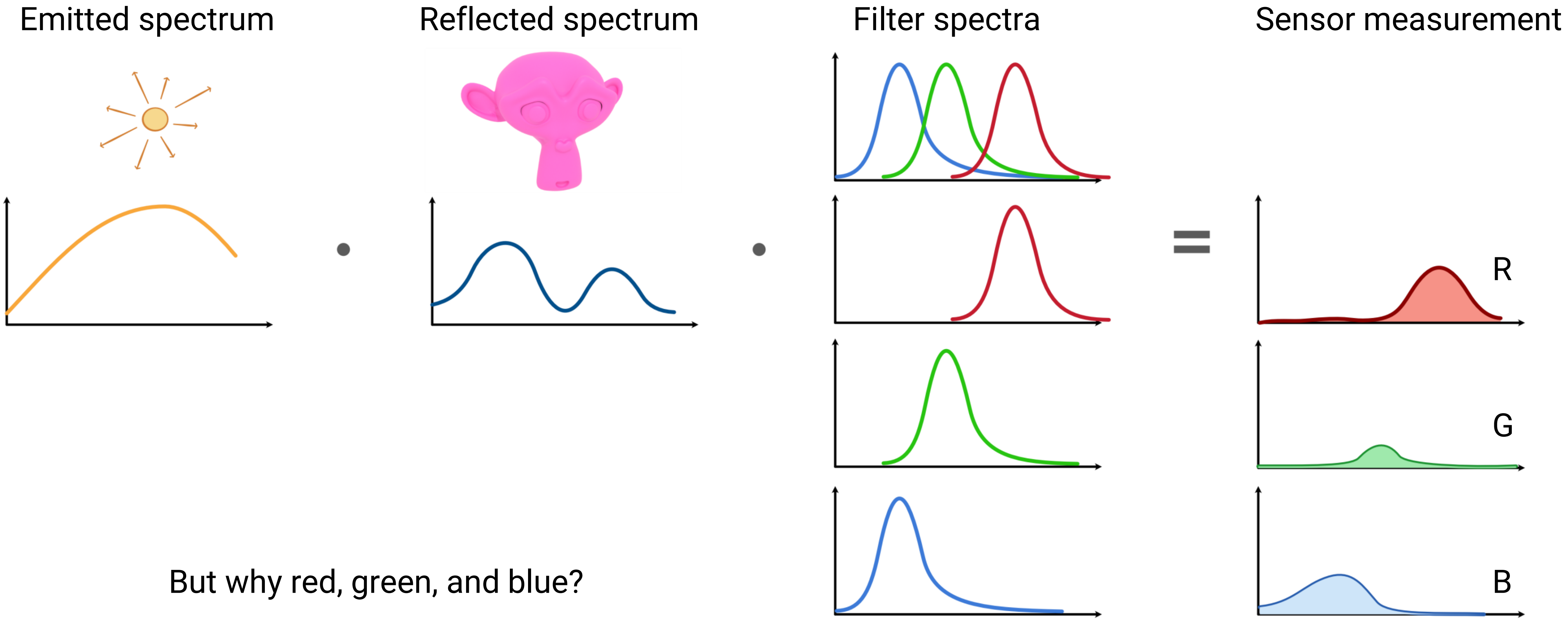
Incoming light $L(\lambda)$

Filter $r(\lambda)$, $g(\lambda)$, $b(\lambda)$

Sensor

$R = \int r(\lambda)L(\lambda)d\lambda$ $G = \int g(\lambda)L(\lambda)d\lambda$ $R = \int b(\lambda)L(\lambda)d\lambda$

The story of light: From emission to pixel value

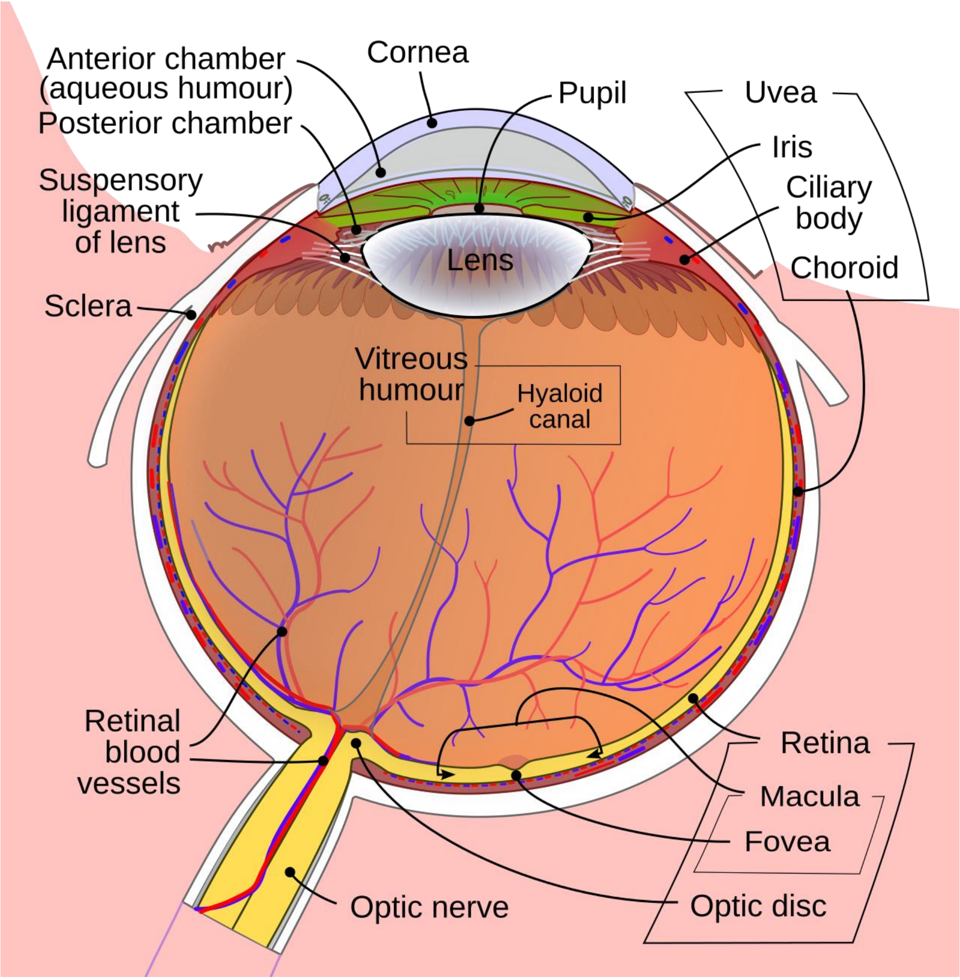


But why red, green, and blue?

The human visual system (HVS)

(The parts that are important for us)

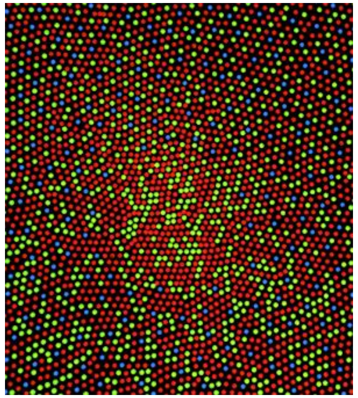
The human eye

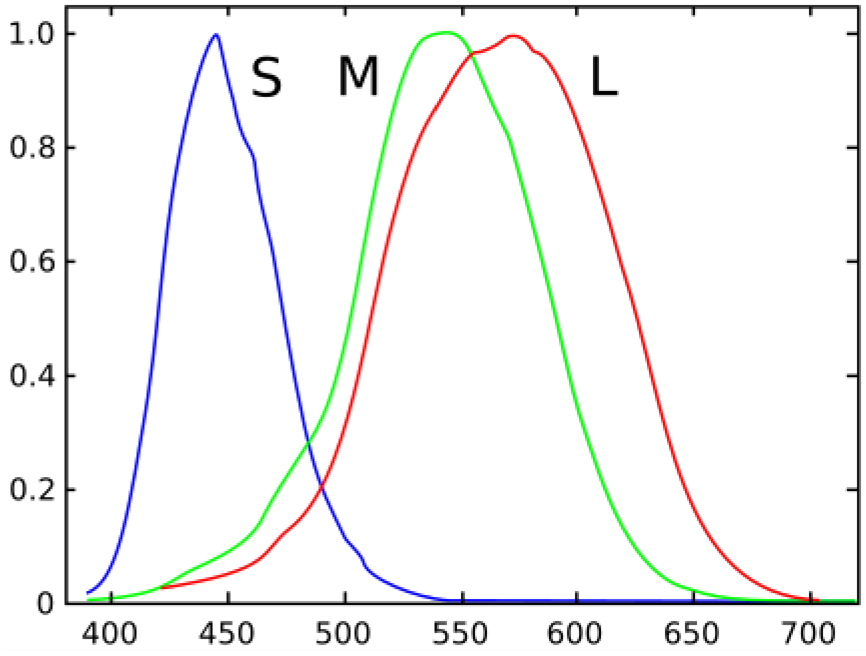


By Rhcastilhos. And Jmarchn.
https://commons.wikimedia.org/w/index.php?curid=1597930

3 types of cones for color perception

- S ("short")
  - Perceive shorter wavelengths (blueish)
  - Only few of them
- M ("medium") and L ("long")
  - Greenish and reddish wavelengths
  - More numerous
- (Exact spectral responses vary genetically)



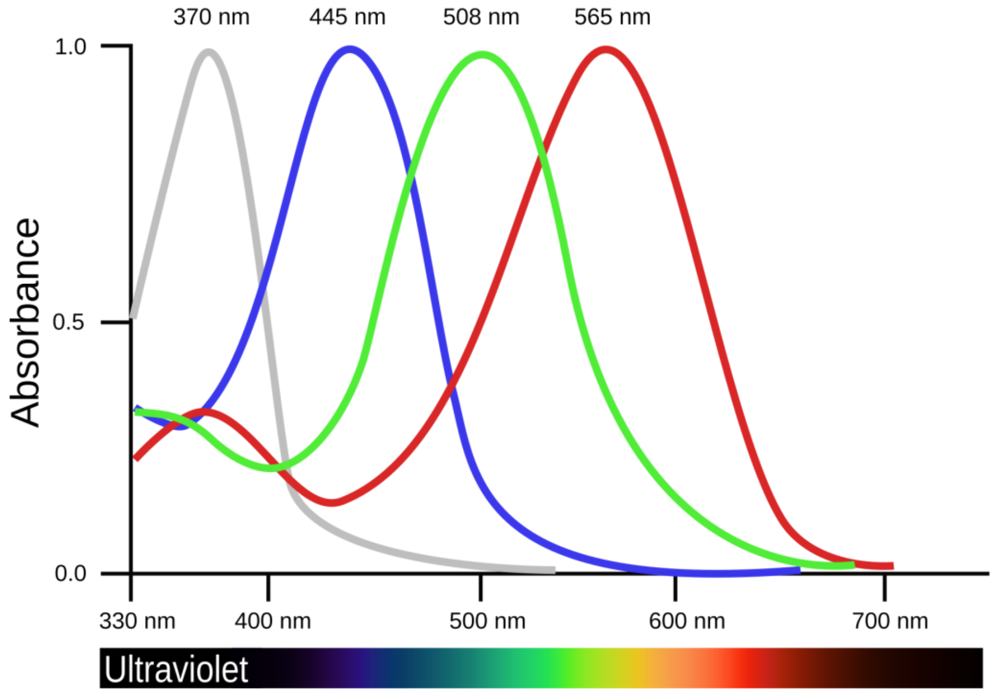


Normalized spectral sensitivities

Dichromatic and tetrachromatic vision

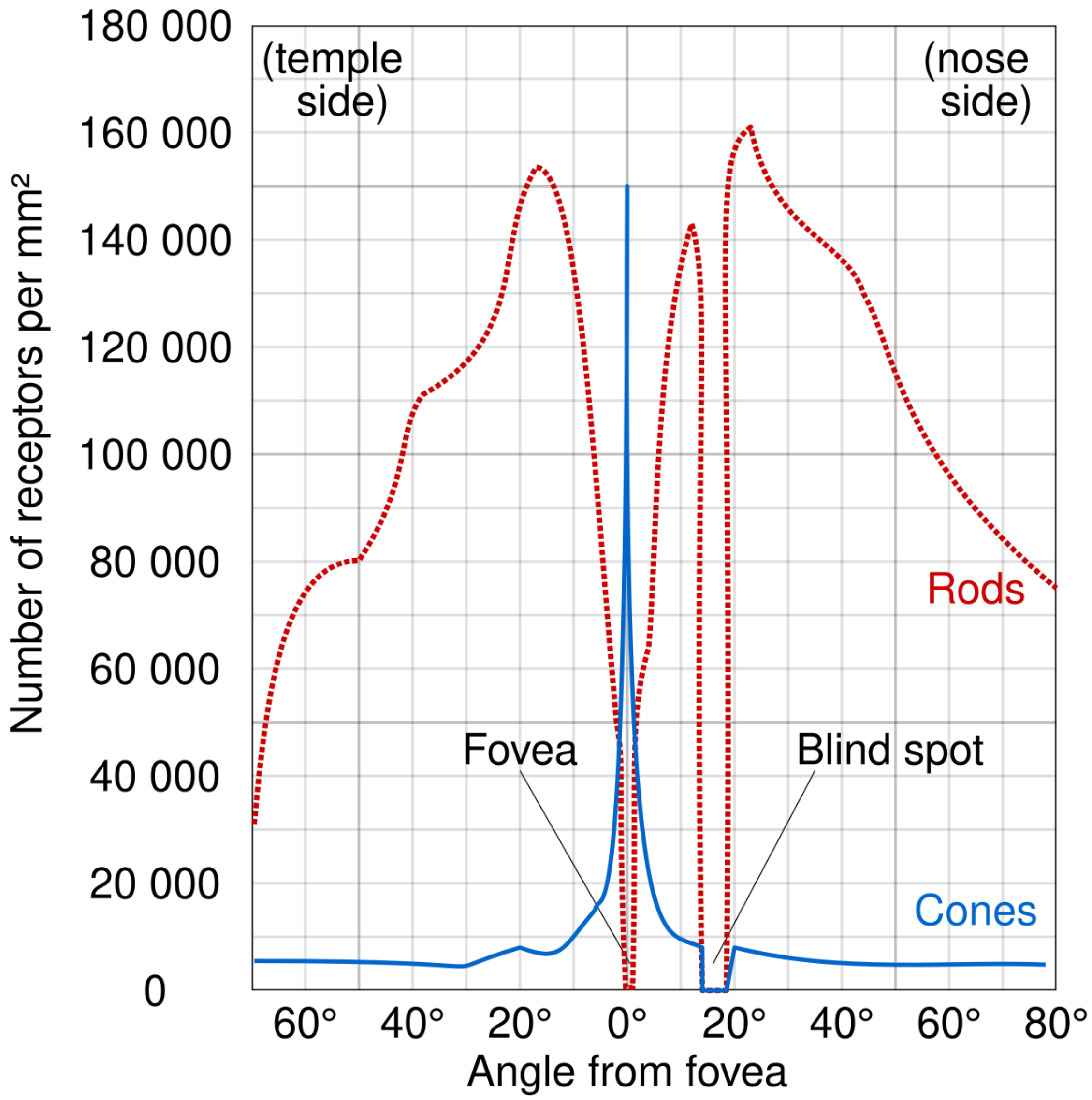
- Some birds and reptiles have 4 types of cones
  - Tetrachromacy
  - They (probably) can distinguish colors that look the same to us
  - They can see UV light
- Many mammals have only 2 types
  - Dichromacy
  - More colors will look the same to them

Tetrachromatic vision of some birds



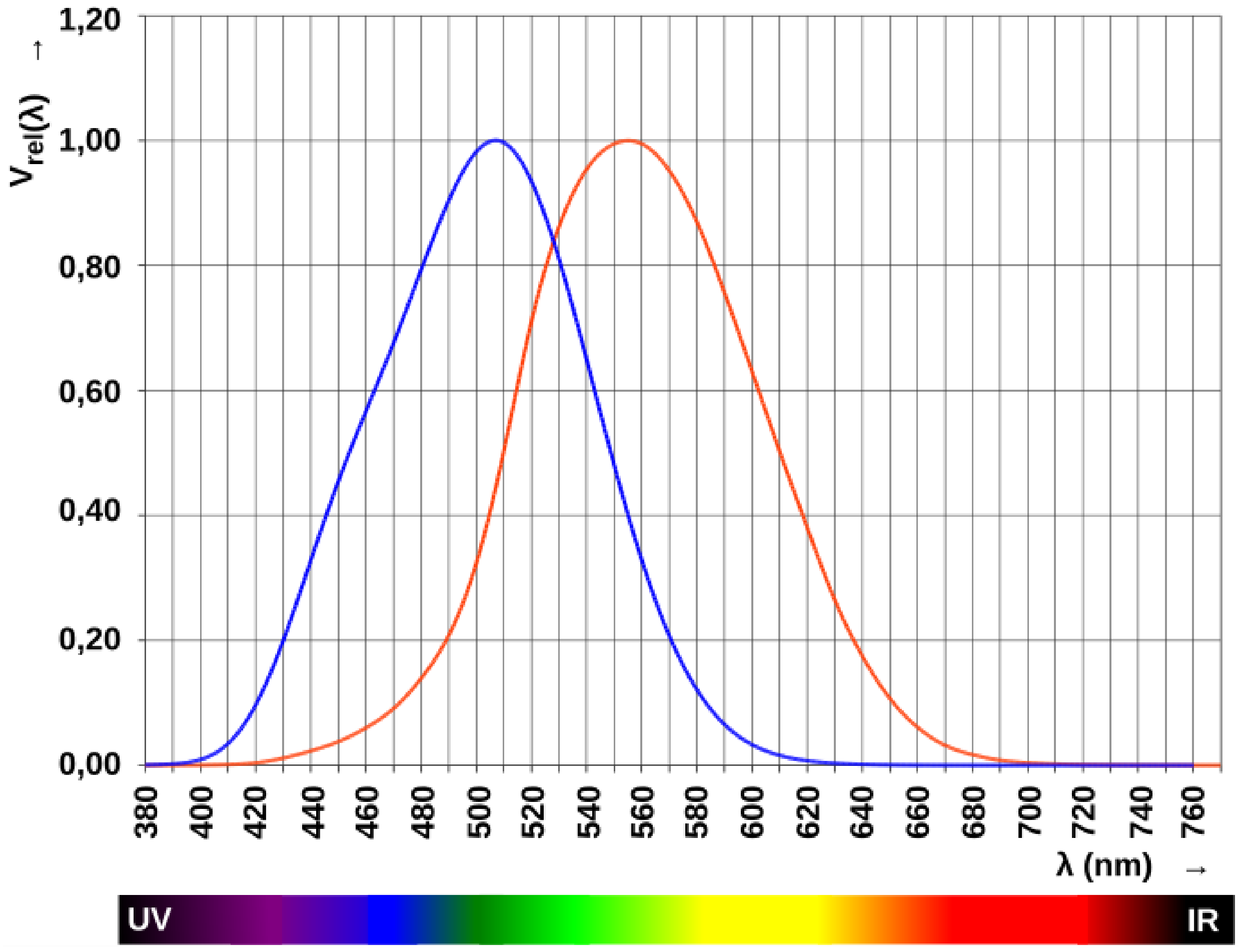
Rods for low-light vision

- Much more sensitive to light than cones
- Predominant in the periphery
- Responsible for night vision
- Most sensitive in the green-blueish region



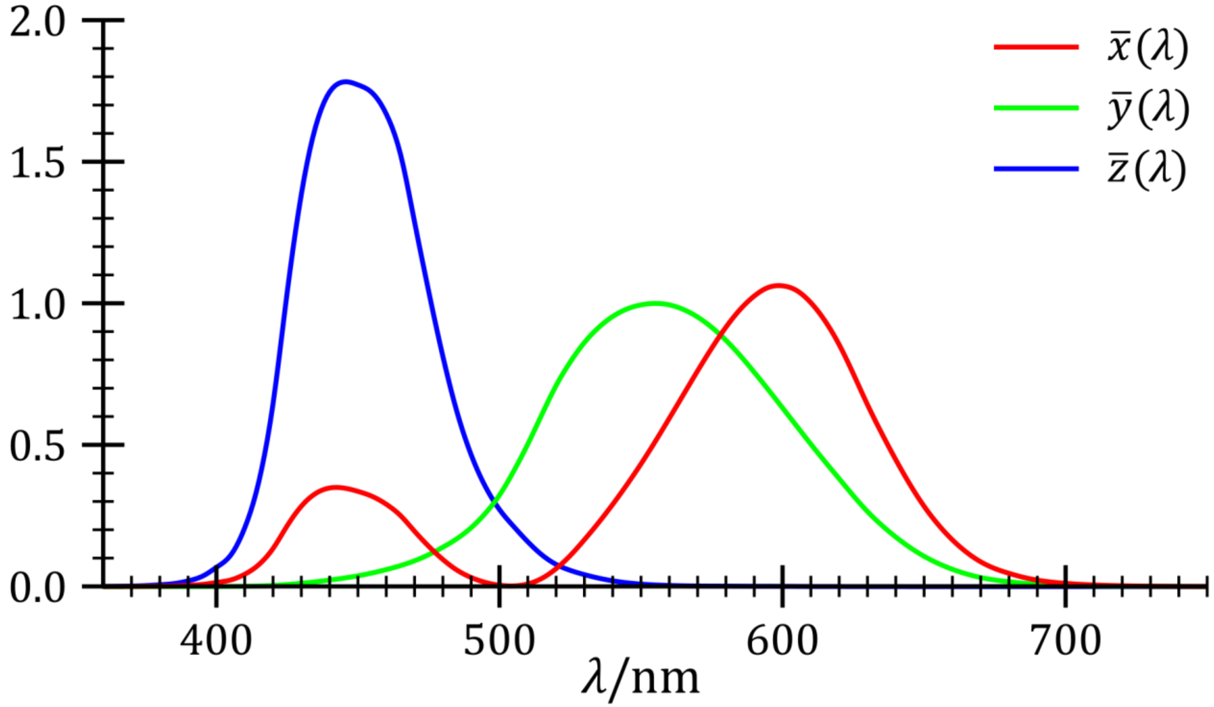
Luminous efficiency function

- Perceived brightness depends on the color
- Standardized for two cases
  - "Daytime" vision (red curve →)
    - Sufficient light to activate cones
    - Peak at 555nm (green)
  - Low-light vision (blue curve →)
    - Only rods are active, cones not sensitive enough
    - Rods are more sensitive to blue light



CIE standard observer empirically models “typical” color perception

- Uses the luminous efficiency $y(\lambda)$
  - Encodes the “brightness”
- Two more curves to model tristimulus color perception
  - $z(\lambda)$
  - $x(\lambda)$
- Values determined experimentally
  - Color matching tasks
- Published as a table



Analytic approximation of the CIE standard observer

- Using tabulated curves is expensive
- The exact response differs between humans with quite some error

➔ Analytical approximation is worthwhile and its error not critical

- A piece-wise Gaussian approximation works well
  - Wyman et al. 2013: *Simple Analytic Approximations to the CIE XYZ Color Matching Functions*
  - https://jcgt.org/published/0002/02/01/

Photometry

Radiometry vs Photometry

- Radiometry
  - Measures the "actual" radiated energy
  - Physical quantity

- Photometry
  - Weighted by *human perception*
  - Perceptual quantity

Luminous flux ⇔ Radiant flux

- Computation:
  - Multiply flux $\Phi(\lambda)$ at each wavelength by response $y(\lambda)$
  - Integrate ("sum") over all wavelengths to get total response
- Unit: lumen (lm)

$$\Phi_V = \int_\Lambda \Phi(\lambda)\, y(\lambda)\, d\lambda$$

Lumen is the right unit to quantify perceived total brightness

“How bright is my room if I use this light bulb?”

Luminance ⇔ Radiance

- Luminous flux per area and solid angle
- Unit: candela per square meter ($\frac{\mathrm{cd}}{\mathrm{m}^2}$)
  - where $1\mathrm{cd} = 1\frac{\mathrm{lm}}{\mathrm{sr}}$

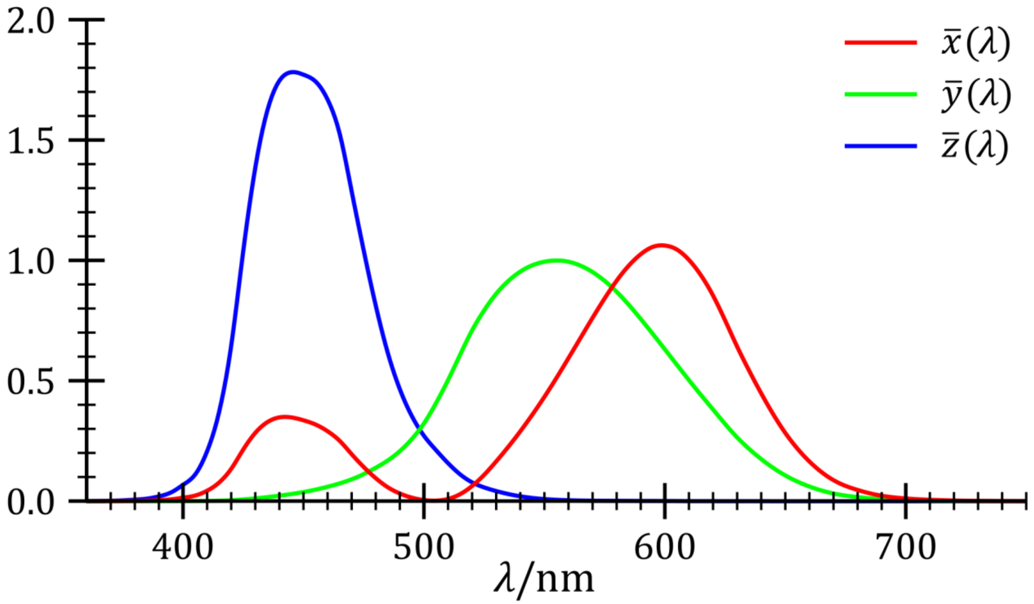
"How bright is a pixel in this display if I look at it"

Tristimulus color

CIE standard observer can be used to encode color as a triplet

- e.g., for a spectral radiance value $L(\lambda)$

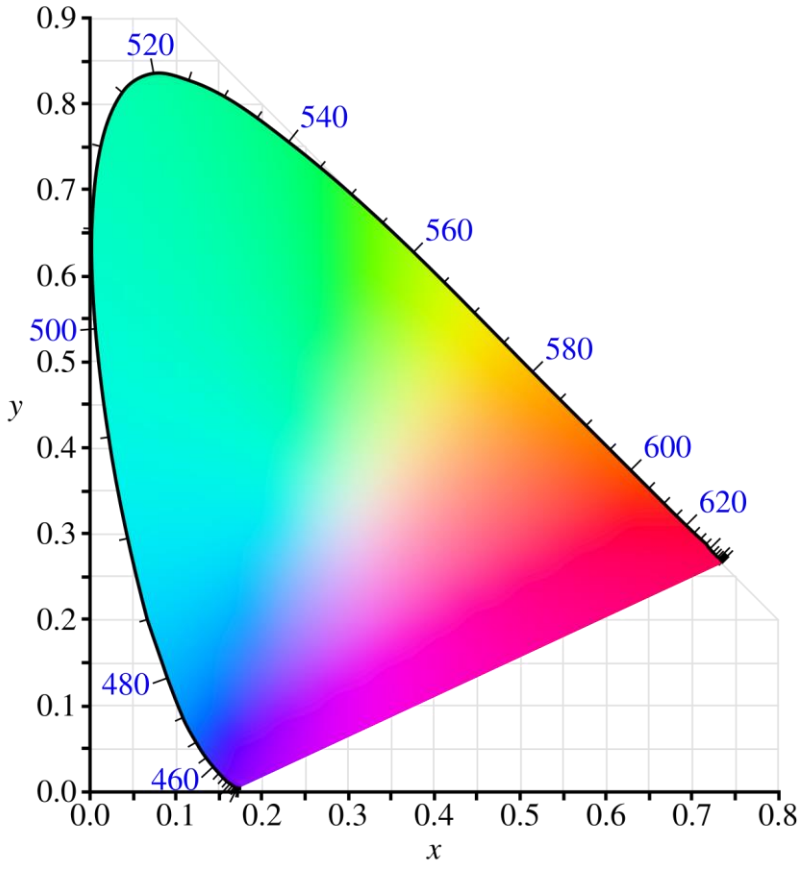
- $X = \int_\Lambda x(\lambda) L(\lambda) d\lambda$
- $Y = \int_\Lambda y(\lambda) L(\lambda) d\lambda$
- $Z = \int_\Lambda z(\lambda) L(\lambda) d\lambda$



CIE XY chromaticity diagram

- Separates the concepts of “brightness” and “color”
- Normalize values to
  - $X' = \frac{X}{X+Y+Z}$
  - $Y' = \frac{Y}{X+Y+Z}$
  - $Z' = \frac{Z}{X+Y+Z} = 1 - X' - Y'$

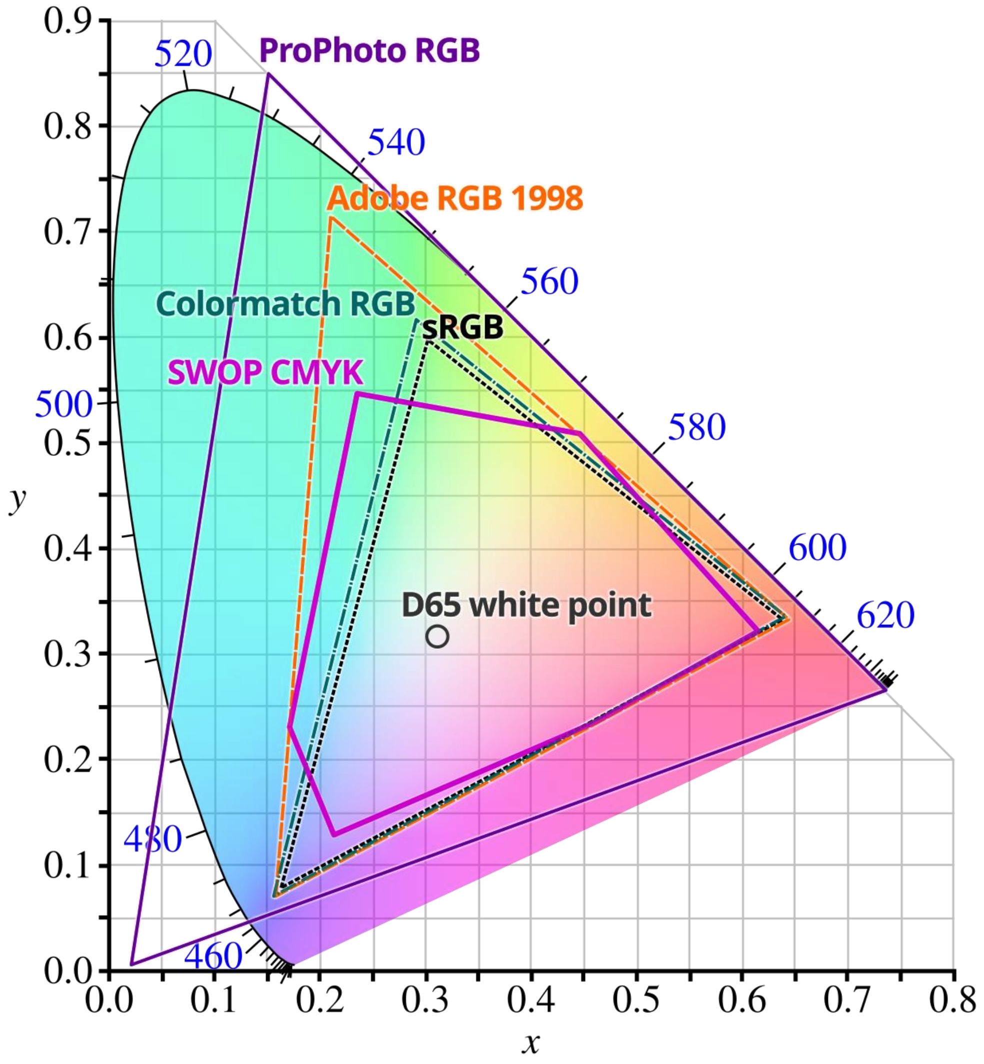
➔ 2D space of perceivable colors



(this is an illustration, impossible to display correctly)

Different RGB color spaces: triangles in the chromaticity diagram

- Define three “primary colors” in XYZ space
- Colors inside the triangle are representable
  - “Gamut”

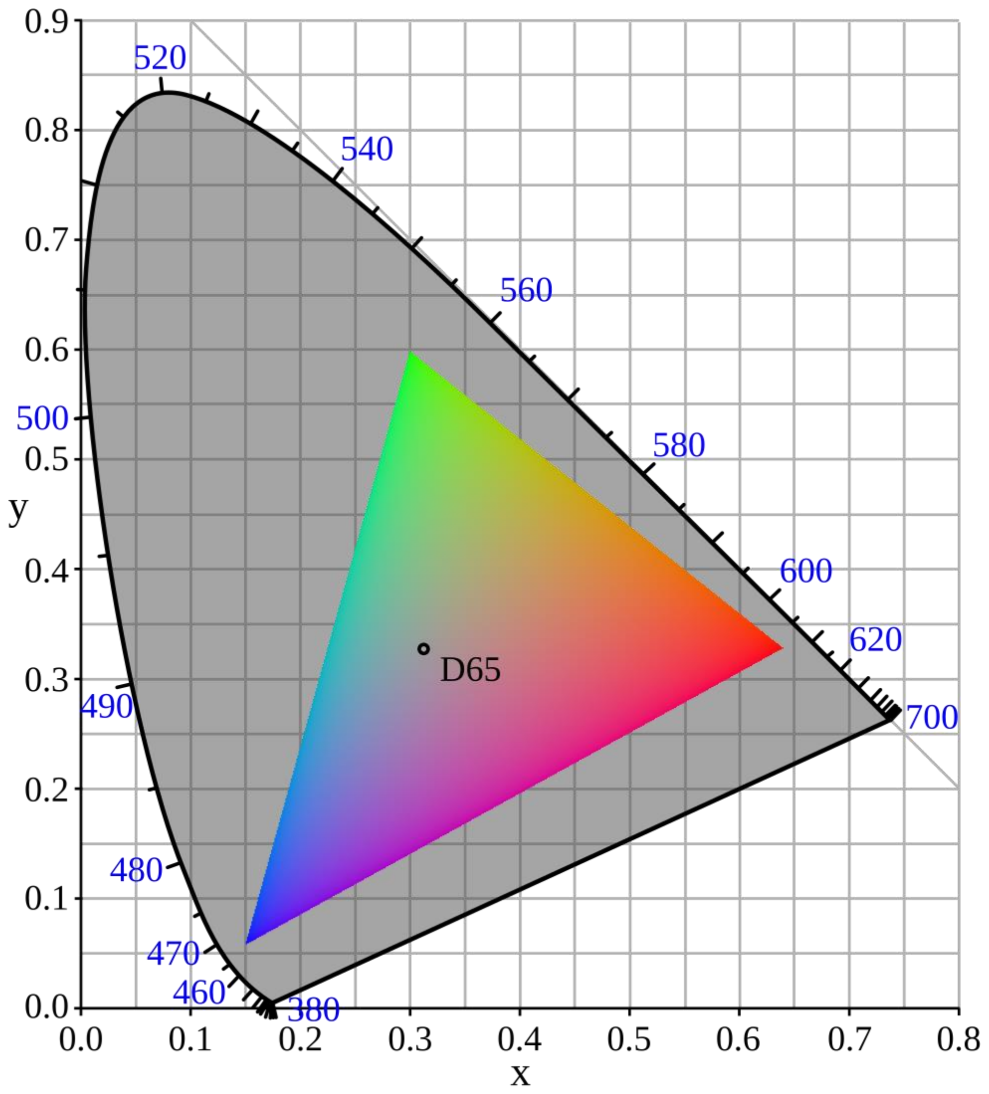


Standard RGB (sRGB)

- Standardized and widely used color space
  - Default for the web
  - And many file formats like .png

- Linear sRGB color values are mapped to XYZ as:

$$\begin{bmatrix} X_{D65} \\ Y_{D65} \\ Z_{D65} \end{bmatrix} = \begin{bmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{bmatrix} \begin{bmatrix} R_{\text{linear}} \\ G_{\text{linear}} \\ B_{\text{linear}} \end{bmatrix}$$



Gamma-corrected sRGB

- sRGB additionally applies a *non-linear* “gamma correction”

$$C_{\mathrm{sRGB}} = \begin{cases} 12.92 C_{\mathrm{linear}}, & C_{\mathrm{linear}} \le 0.0031308 \\ 1.055(C_{\mathrm{linear}}^{1/2.4}) - 0.055, & C_{\mathrm{linear}} > 0.0031308 \end{cases}$$

- Sometimes approximated as $C_{\mathrm{sRGB}} \approx C_{\mathrm{linear}}^{\frac{1}{2.2}}$

- Why?
  - Historically: compensates for CRT brightness characteristics
  - Nowadays:
    - Typically, sRGB values are stored as 8-bit int
      - $\lfloor C_{sRGB} \cdot 255 \rfloor$
    - Gamma correction gives higher resolution for darker colors, where differences are more visible

When to use gamma correction?

- When sending image data to a display
- When storing image data in a file
  - (e.g., for PNG where the format specifies it)

When **not** to use gamma correction

- When storing float values
  - Already a piecewise-logarithmic representation space
- For any other operation
- Especially not for **interpolation**

interpolated in linear sRGB space

interpolated in gamma-correcte‌d sRGB space

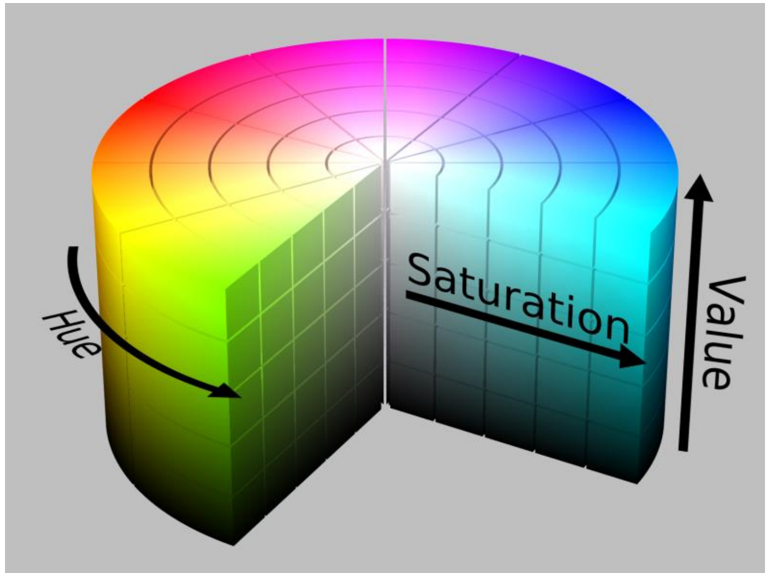
➔ When processing sRGB colors, convert them to linear first!

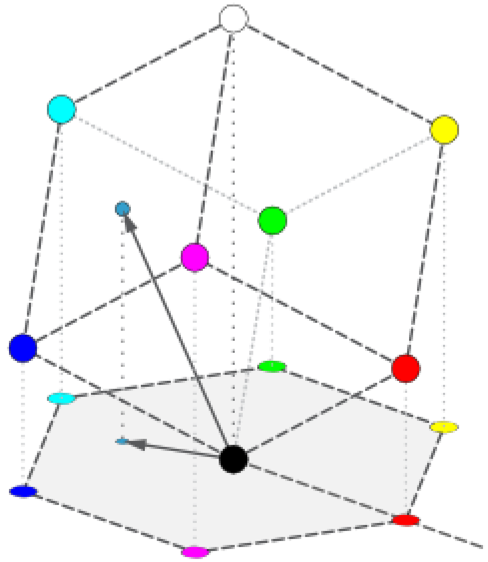
Other color spaces

Some examples

HSV – a handy color space for artistic use

- Cylindrical encoding into
  - Hue (rotation in degrees, the “color”)
  - Saturation ($S \in [0,1]$; less saturated ➔ closer to white / gray)
  - Value ($V \in [0,1]$; luminance)



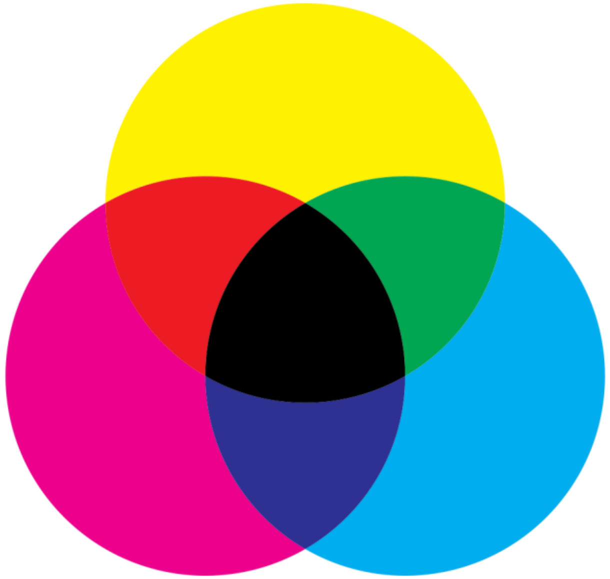


- RGB primaries span a cube of colors
- Tilting + projecting gives Hue/Saturation space

CMYK – the color of your printer

- Printers layer pigments on white paper to get color
- *Subtractive*:
  - Cyan “removes” red light
  - Magenta “removes” green light
  - Yellow “removes” blue light
- e.g., layering yellow (removes blue) and cyan (removes red)

  ➔ only green left
- Imperfect and highly susceptible to paper, inks, ...
  - Dedicated black “color” because mixing C + M + Y is only perfect in theory with ideal pigment & paper



Rendering with color

Why we shouldn’t render with tristimulus (RGB) values

- RGB model is derived from human *perception*
- We compute *emission* and *scattering* of light, perception is only the *last step*

➔ Ideally, only use tristimulus for *display*

Why we render with RGB anyway

- Cheap and easy
- Yields a reasonable approximation of the expected color
- **Input data:**
  - Textures, color values, etc.
  - Far easier to model with RGB than continuous spectrum
  - But: “Spectral upsampling” can find a matching spectrum for RGB colors

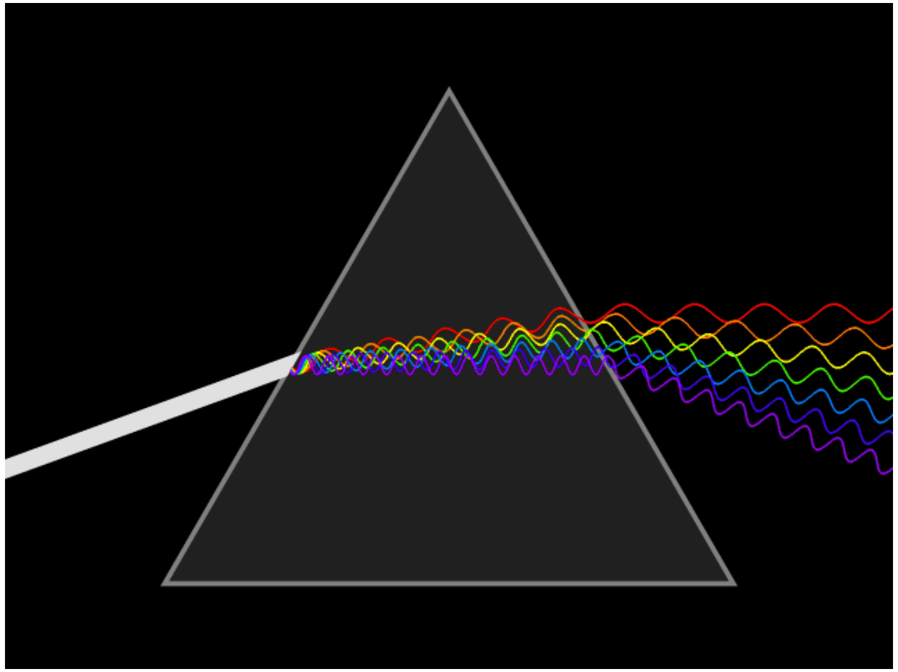
How to do “proper” spectral rendering (roughly)

- Compute per-wavelength radiance $L(x, \omega, \lambda)$
- Integrate over all wavelengths (three times) to get X, Y, and Z values
  - $X = \int_\Lambda x(\lambda) L(\lambda) d\lambda$
- Can be done with Monte Carlo:
  - Sample a random wavelength $\lambda \sim p(\lambda)$
  - Sample a ray / path as usual, but evaluate monochromatically for λ
  - Estimate $X \approx \frac{1}{n_\lambda} \sum_{i=1}^{n_\lambda} \frac{L(x,\omega,\lambda)}{p(\lambda)}$
  - We can re-use the same sample to compute Y and Z
- Compute sRGB color from XYZ

Spectral rendering enables wavelength-dependent scattering

- e.g., dispersion
- Index of refraction η depends on the wavelength
- Snell's law gives different direction for different wavelengths

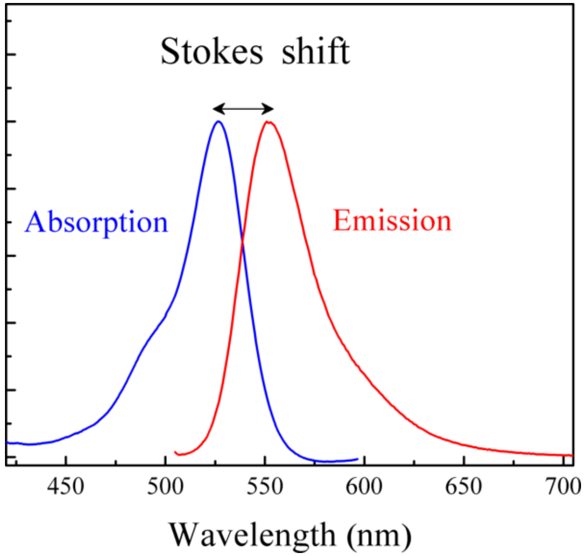
➔ Rainbows!



- Implementation:
  - When computing $L_o(\lambda, x, \omega_o)$
  - BSDF is a function of the wavelength $f_r(x, \omega_i, \omega_o, \lambda)$
  - Uses $\eta(\lambda)$ and Snell's law
  - Otherwise the same

Fluorescence: wavelength shift in the reflection

- Modelled as part of the BSDF: $f(x, \omega_i, \omega_o, \lambda_i, \lambda_o)$
- To compute $L_o(\lambda_o, \omega_o)$
  - Determine λ_i based on λ_o
  - Evaluate BSDF and proceed as usual, but with new wavelength



Blue laser turning green in beer

Mineral collection with and without UV light (Australian Museum, Sydney)

HDR images and tone mapping

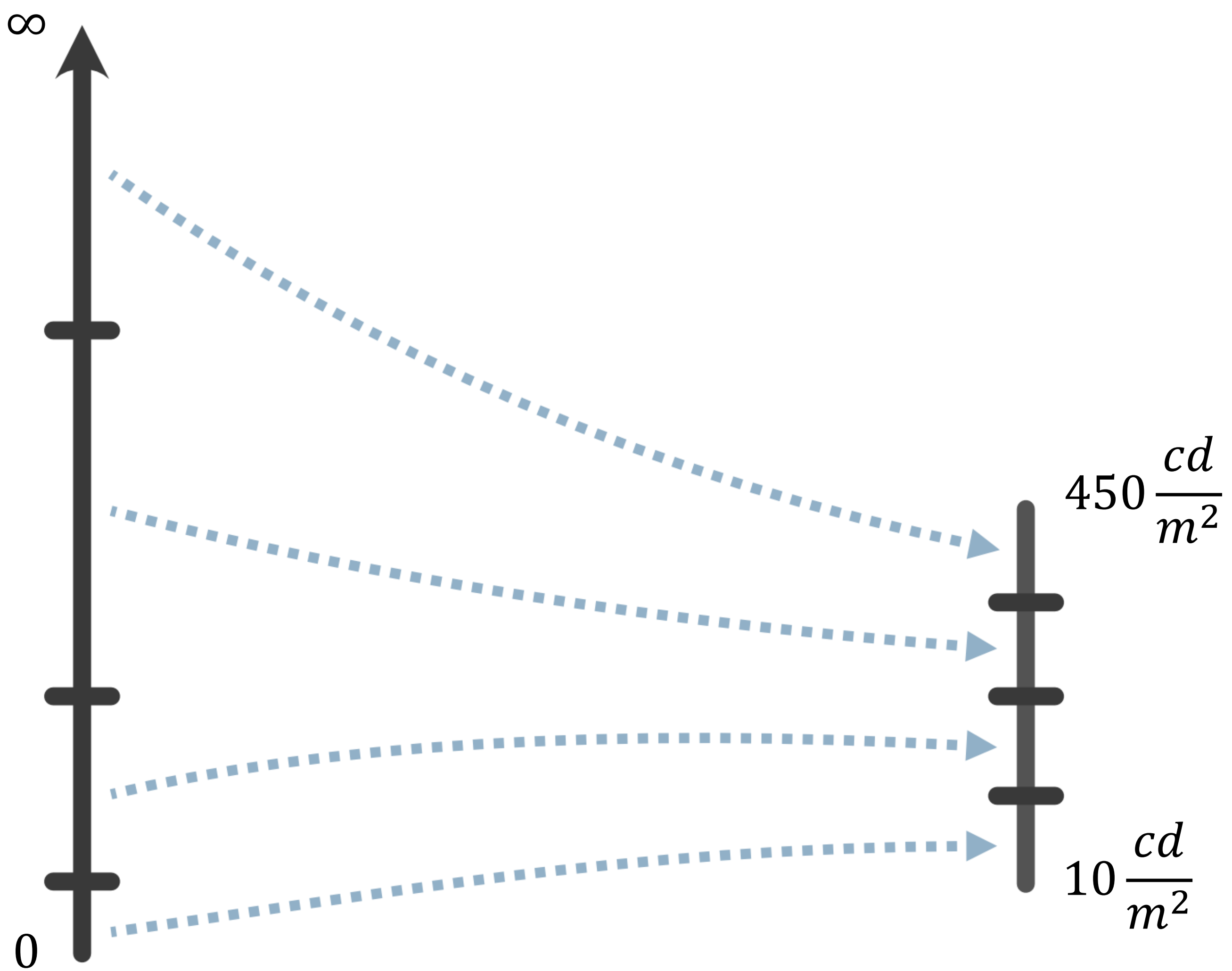
In the real world, light has a high dynamic range

Displays have a low dynamic range

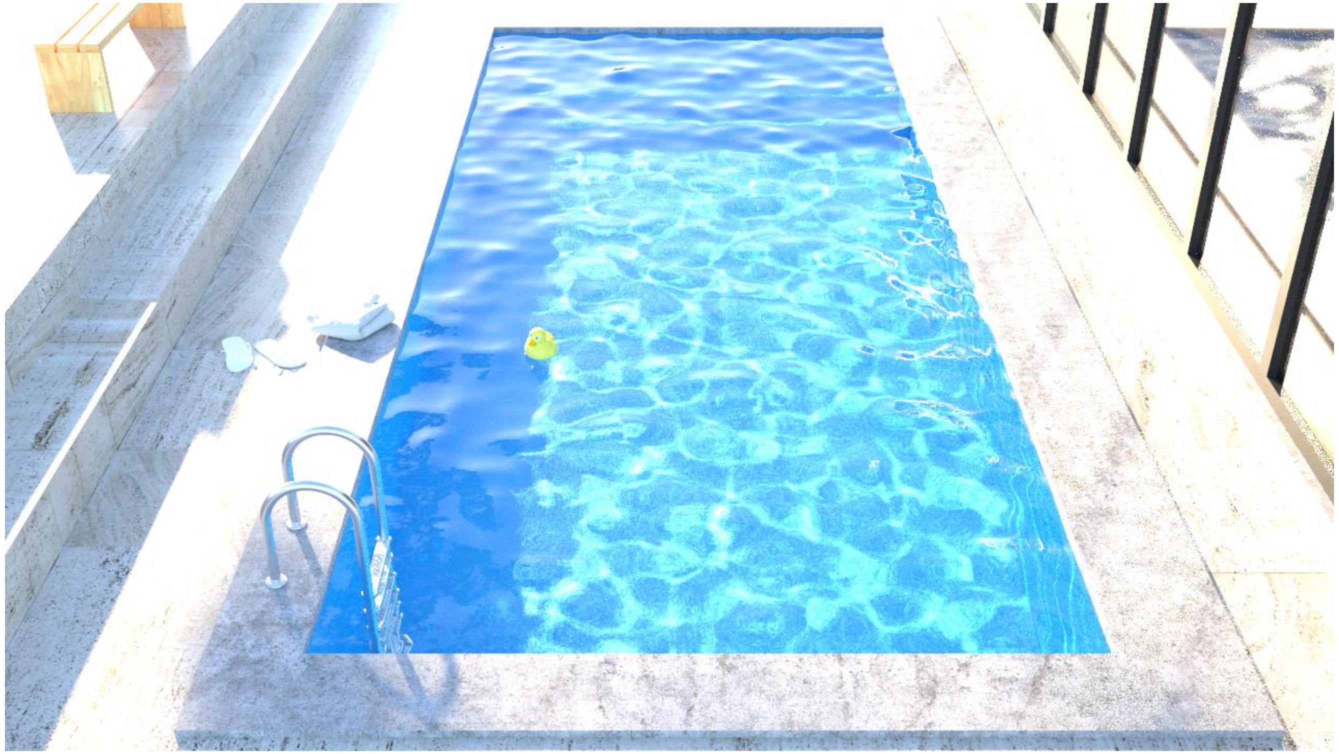
e.g., my “HDR10” screen supports up to $\sim 400 \frac{\text{cd}}{\text{m}^2}$

(and “black” is $> 0 \frac{\text{cd}}{\text{m}^2}$)

Tone-mapping compresses the dynamic range for display



The simplest “tone-mapper”: clamp



```
R = min(R, 1)
G = min(G, 1)
B = min(B, 1)
```

➔ Bright images look washed out, dark ones too dark

Reinhard tone mapping – simple

```
R = R / (R + 1.0)
```

➔ Simple, reveals more detail, but can look "washed out"

Reinhard tone mapping with maximum luminance parameter M

```
Y = 0.2126 * R + 0.7152 * G + 0.0722 * B;     // compute pixel luminance
R = R * (1 + Y / (M * M)) / (1 + Y)   // scale so that Y > M is white
```

➔ More vivid colors

Reading materials

- *Erik Reinhard; Wolfgang Heidrich; Paul Debevec; Sumanta Pattanaik; Greg Ward; Karol Myszkowski (2010). High Dynamic Range Imaging: Acquisition, Display, and Image-Based Lighting. Morgan Kaufmann. p. 82. ISBN 9780080957111.*

Handling color and dynamic range in your RGB renderer (simplified)