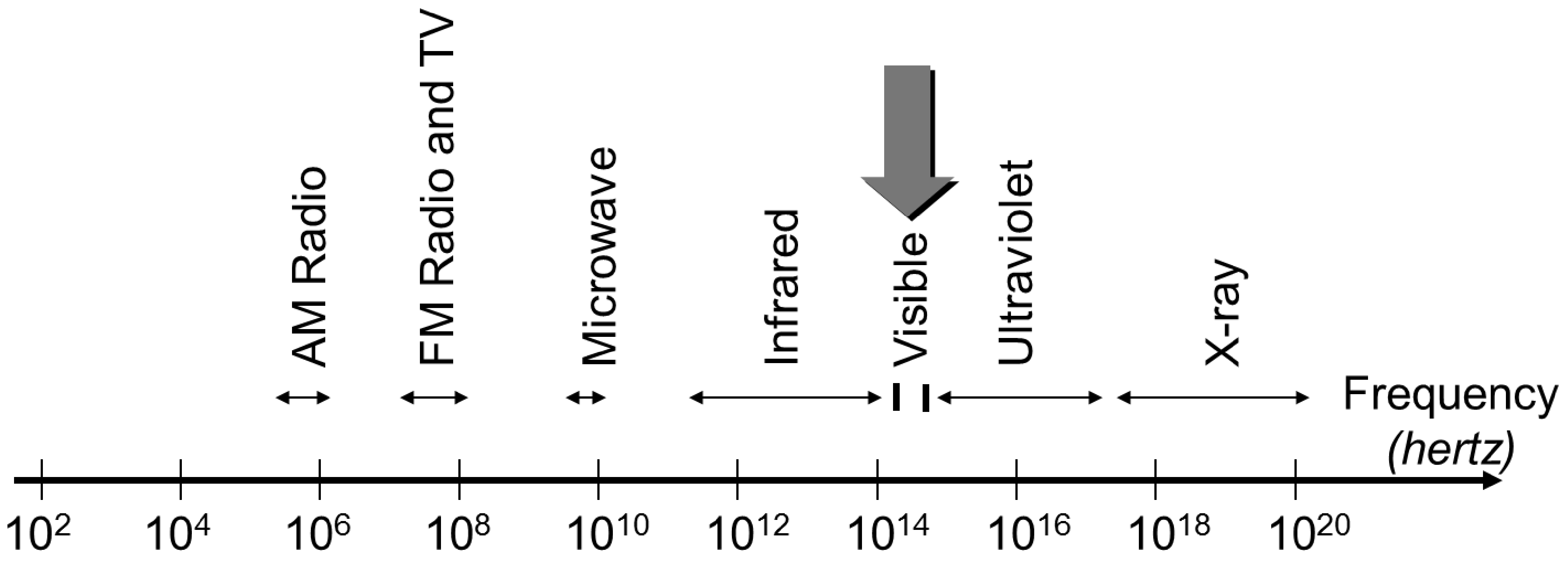
ohm

2.3 Color

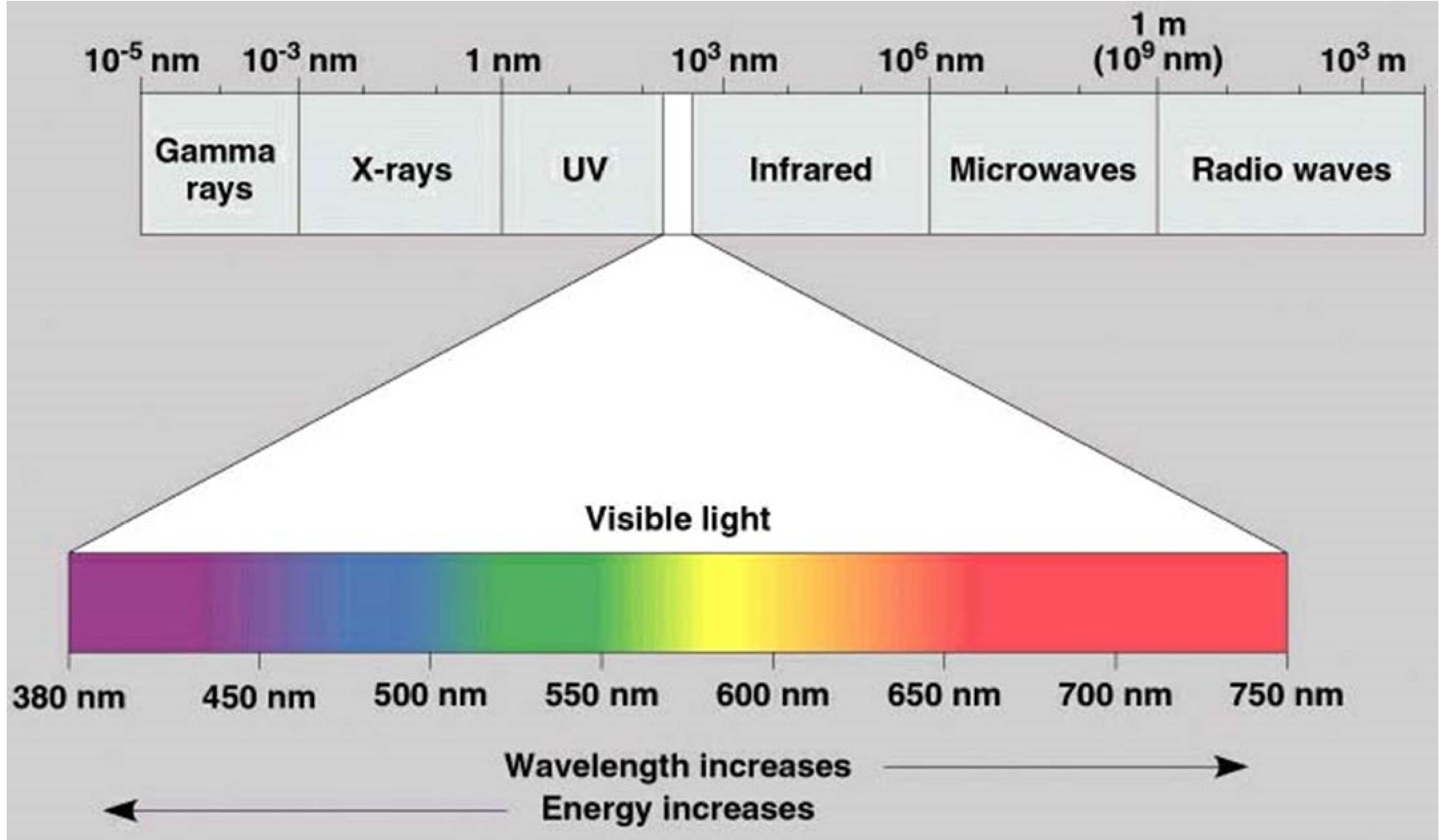
Color

- What is light?
  - Narrow frequency band of electromagnetic spectrum
  - Red color: ~4.3 * 1014 Hz
  - Violet color: ~7.5 * 1014 Hz



Color

- Light can originate from
  - Emission
  - Scattering
  - Absorption / reflection



Color

- What is color?
  - Physical
    - Spectra of wavelengths
  - Psychological
    - Stimulus sent from the optic system to the brain
    - Sensors on the retina of the eye: rods and cones
  - Computer graphics
    - Different sets of bases and coordinates
    - Depending on the type of display and application

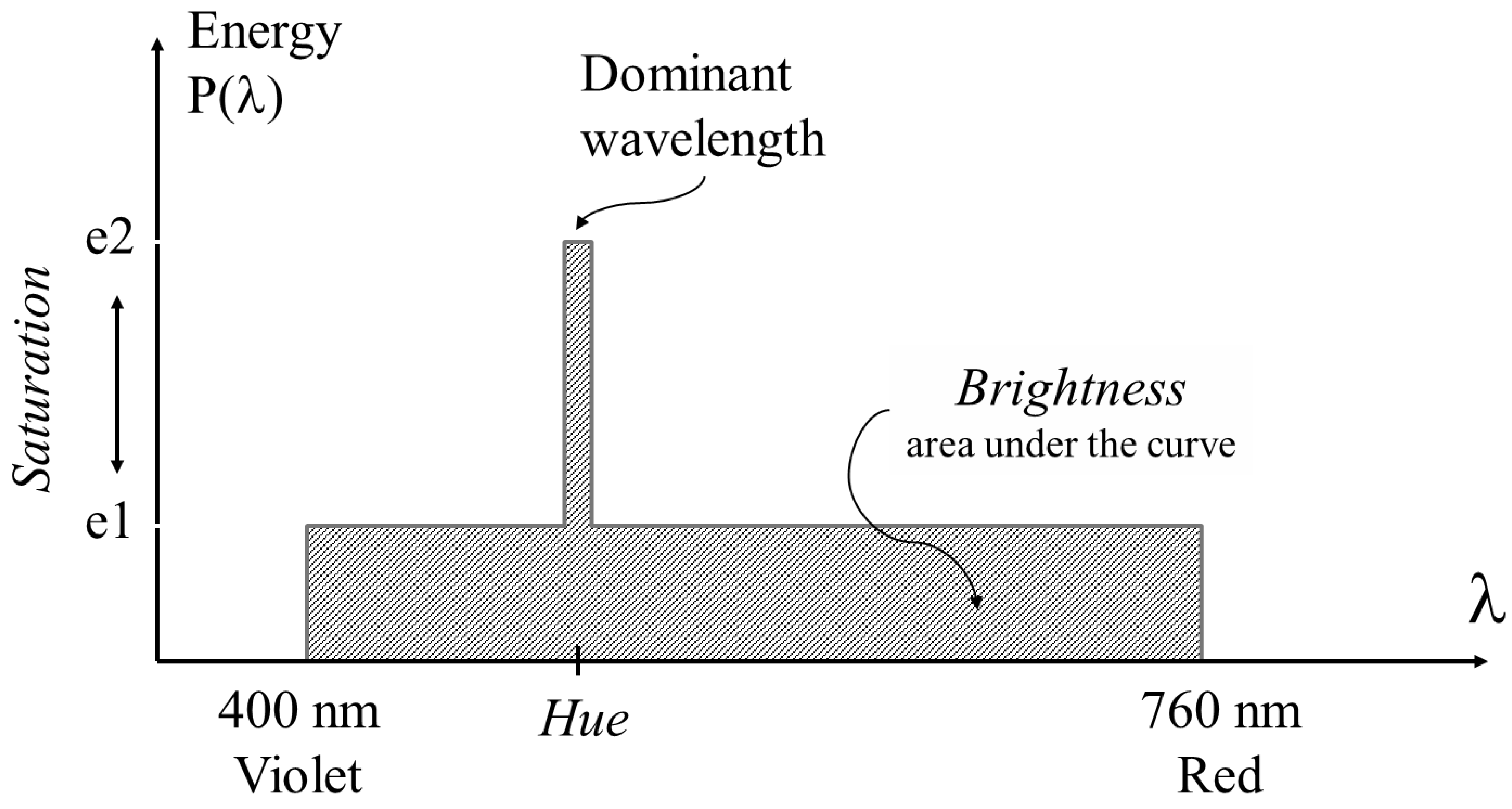
Color

- Perceptual terms
  - Hue
    - The color seen (e.g. red, blue, ...) - dominant wavelength
  - Saturation Intensität der Farbe
    - How far is the color from a grey of equal intensity (how intense is the hue?)
  - Brightness
    - Total light energy - quantified as luminance
    - Perceived intensity of a self-luminous object - emitted light
    - Lightness: refers to intensity from a reflecting object

ohm

Color

- Hue, saturation and brightness



Color

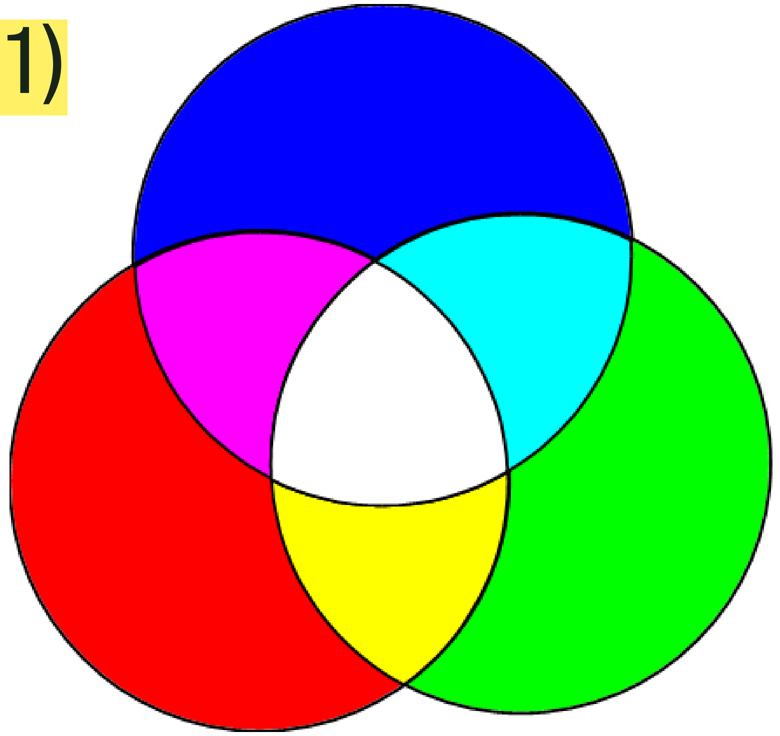
- Complementary colors
  - Mixing produces white light
    - E.g.: red + cyan, green + magenta, blue + yellow
- Primary colors
  - Base colors of color model (three colors sufficient!)
  - Other colors mixed out of primary colors
    - No finite set can produce all possible visible colors!
- Color gamut
  - Set of all colors produced from primary colors

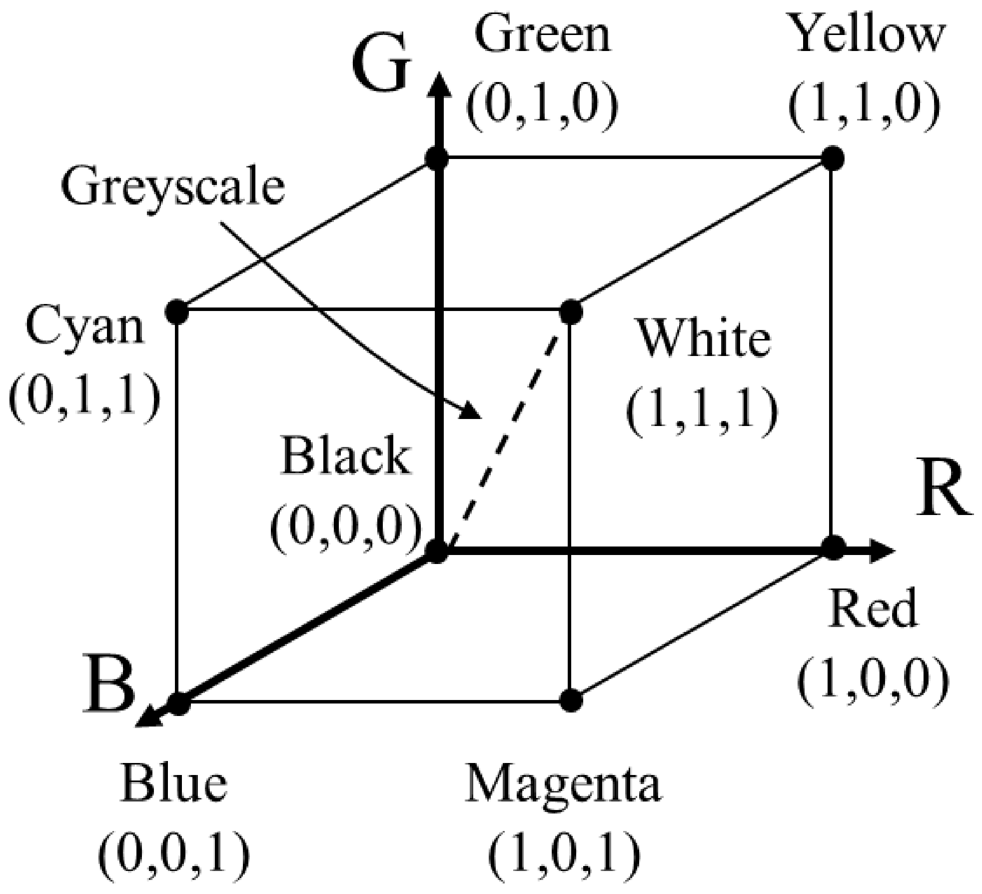
ohm

Color

RGB color model

- Red, green and blue primaries
- Used (internally) in every monitor
- Additive (colors added to a black background)
- Black = (0,0,0), White = (1,1,1)

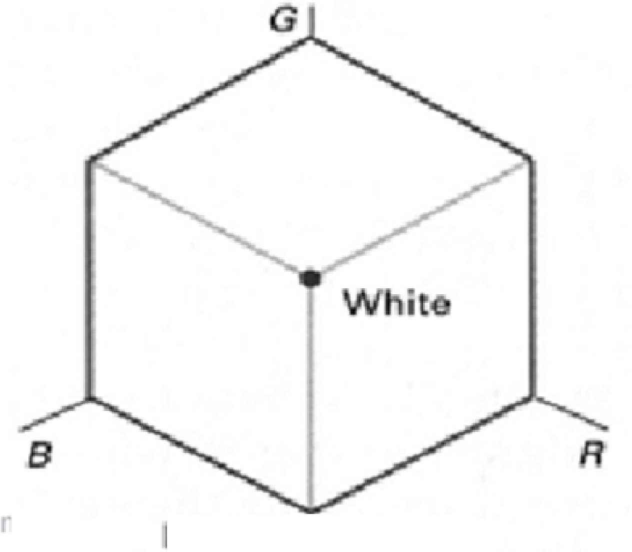




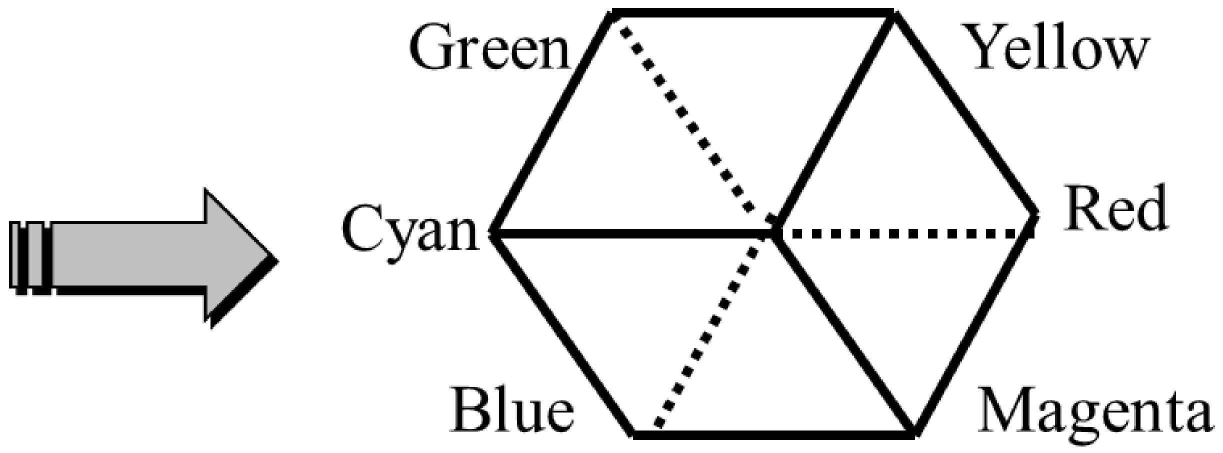
Color

HSV color model

- More intuitive color specification (hue, saturation, value = brightness)
- Derived from RGB
  - Flatten RGB color cube along the diagonal from white to black



RGB color cube



Color hexagon

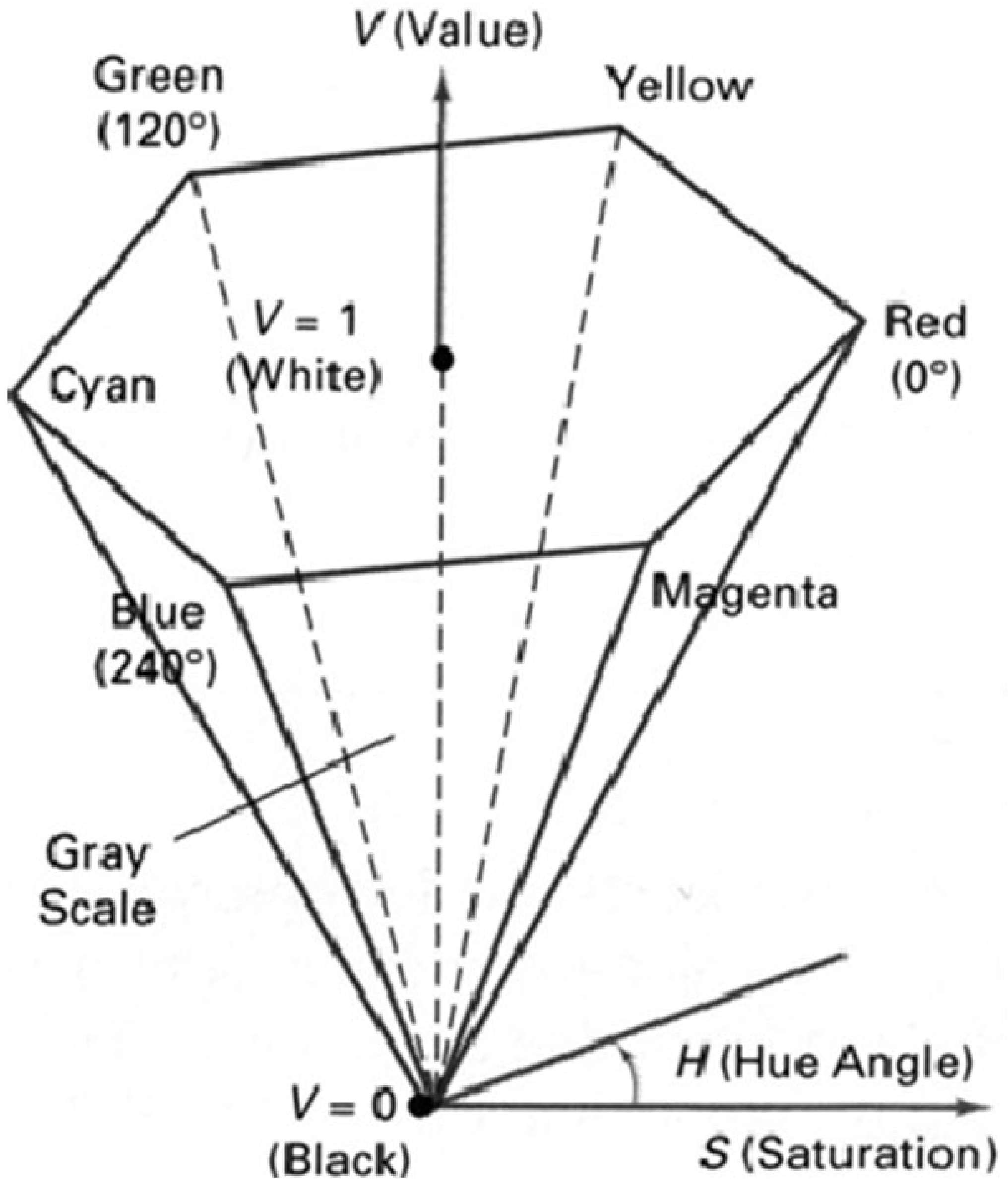
- Hue, saturation and value primaries

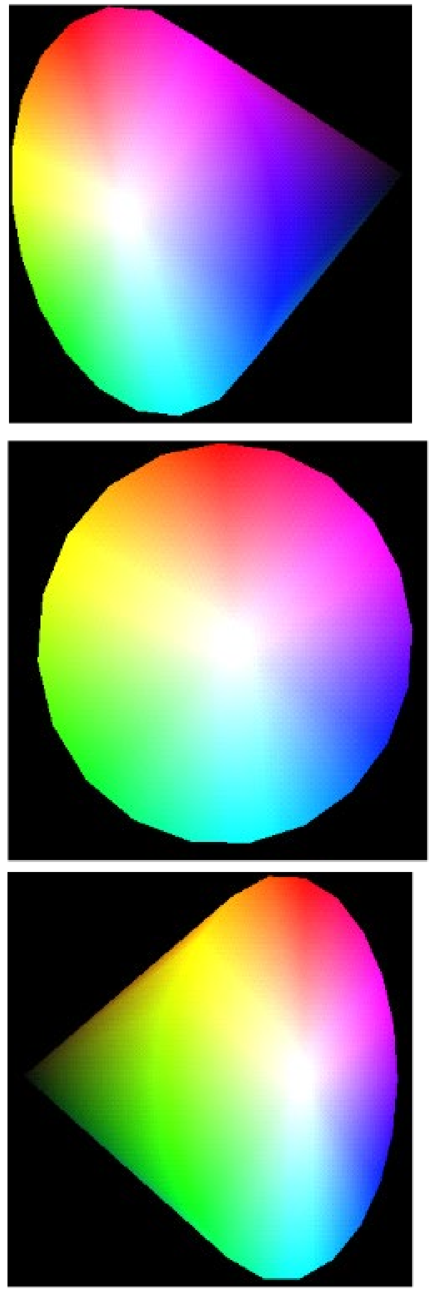
Color

HSV color model

- Components
  - Hue (H)
    range [0º, 360º]
  - Saturation (S)
    range [0, 1]
  - Value (V)
    range [0, 1]

Hue ist Winkel
Value ist y-Achse
Saturation ist x-Achse

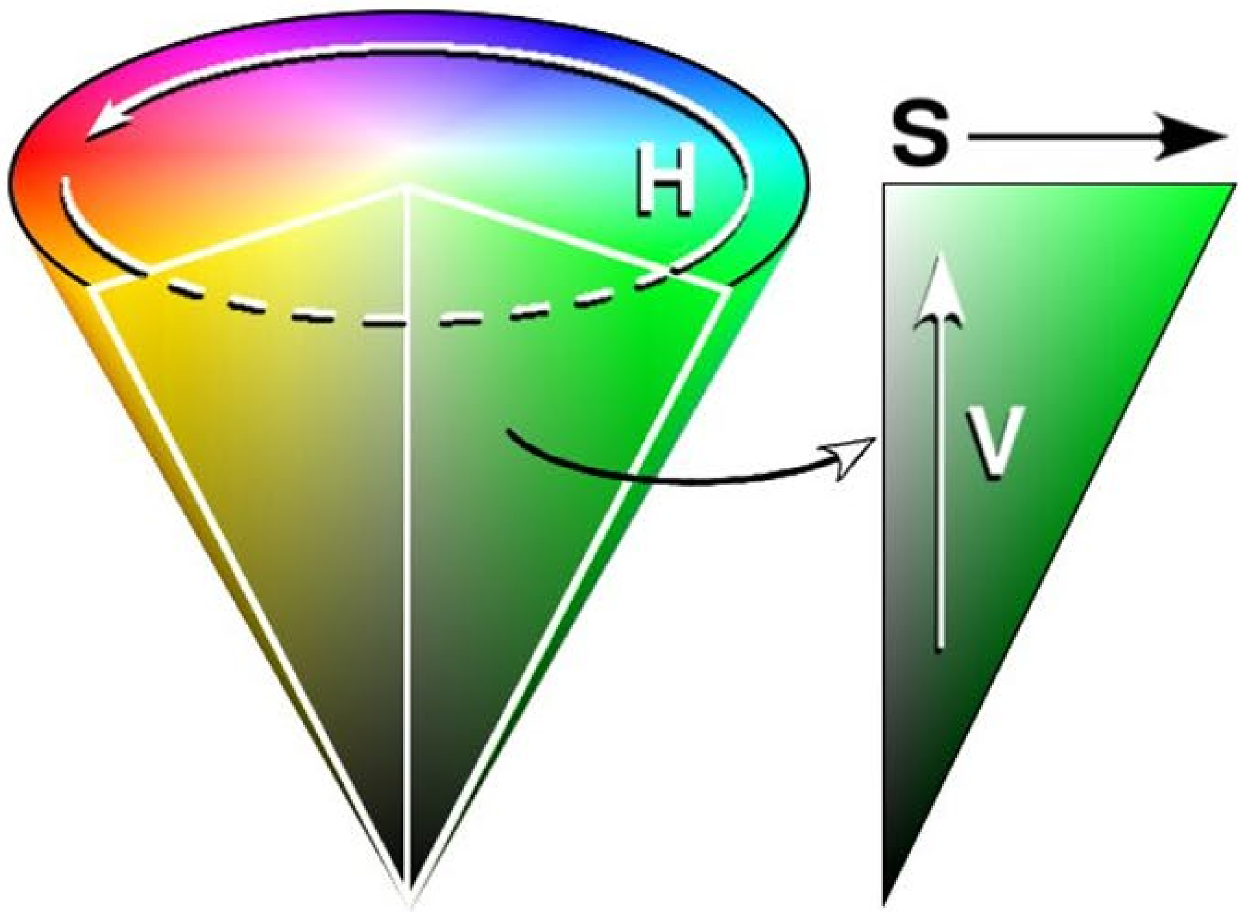


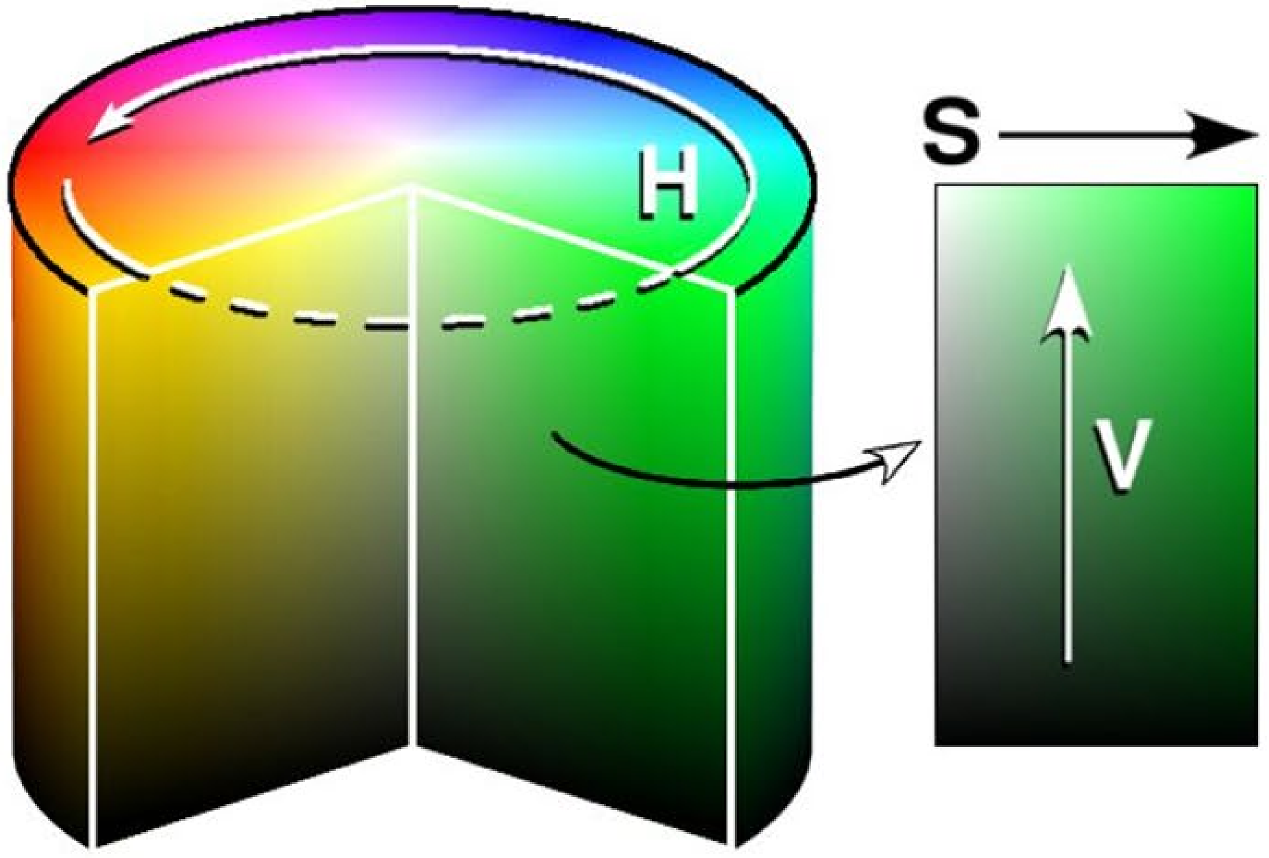


Color

Geometric model for HSV

- Circular cone or cylinder





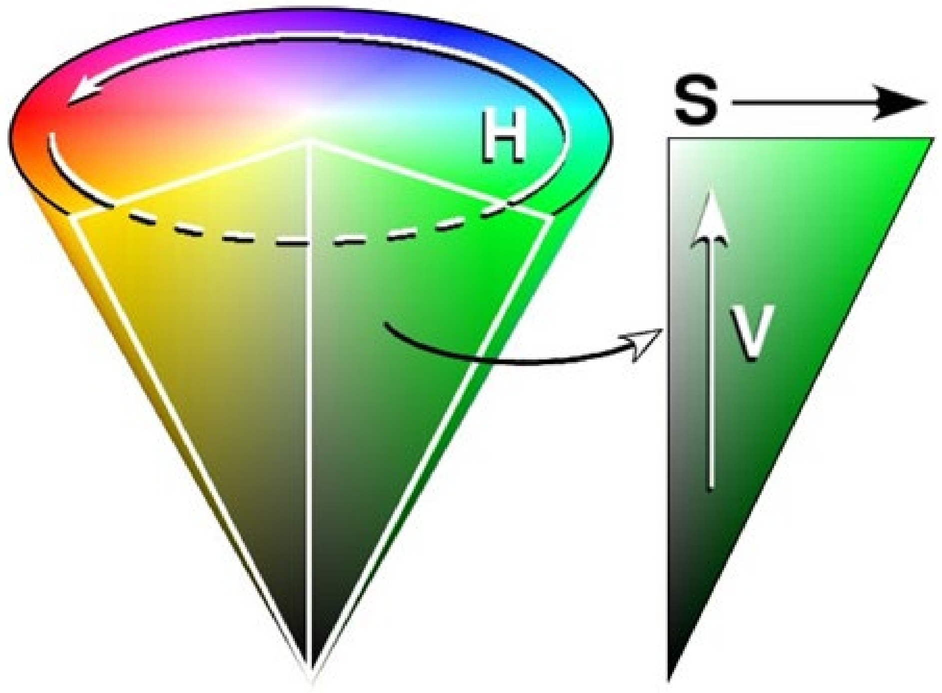
Color

- RGB to HSV (cone model)

$$V = \max(R, B, G) \quad (= \max),$$

$$S = \frac{\max - \min}{\max},$$

$$H = \begin{cases} 60 \cdot \frac{G-B}{\max - \min} & \text{if } \max\{R, G, B\} = R \\ 60 \cdot \frac{B-R}{\max - \min} + 120 & \text{if } \max\{R, G, B\} = G \\ 60 \cdot \frac{R-G}{\max - \min} + 240 & \text{if } \max\{R, G, B\} = B \end{cases}$$



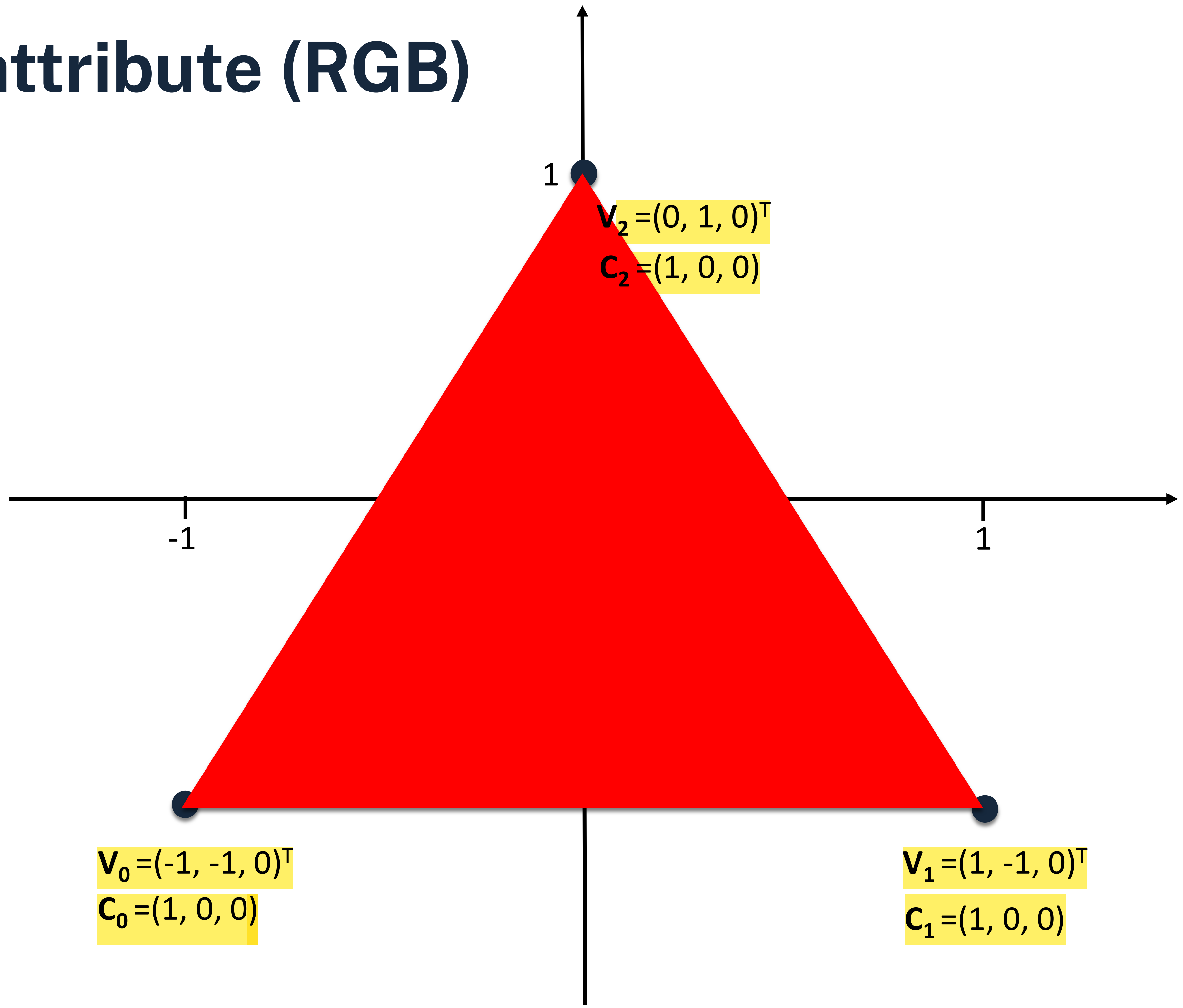
Color

Other color models

- CMY
  - Cyan, magenta and yellow primaries - used usually on hardcopy devices (printers)
  - Subtractive (white background!)
  - Complimentary to RGB (C=1-R, M=1-G, Y=1-B)
- CMYK
  - Addition of K-channel for black (saving ink)
  - Non-unique color presentation, e.g. (1,1,1,0) === (0,0,0,1)
- All of these are device dependent!
- Absolute model - device independent: CIE-XYZ
  - Used for calibration and data exchange
  - Based on human perception of colors

Color

Color as vertex attribute (RGB)



2.4 Lighting and Shading

Lighting

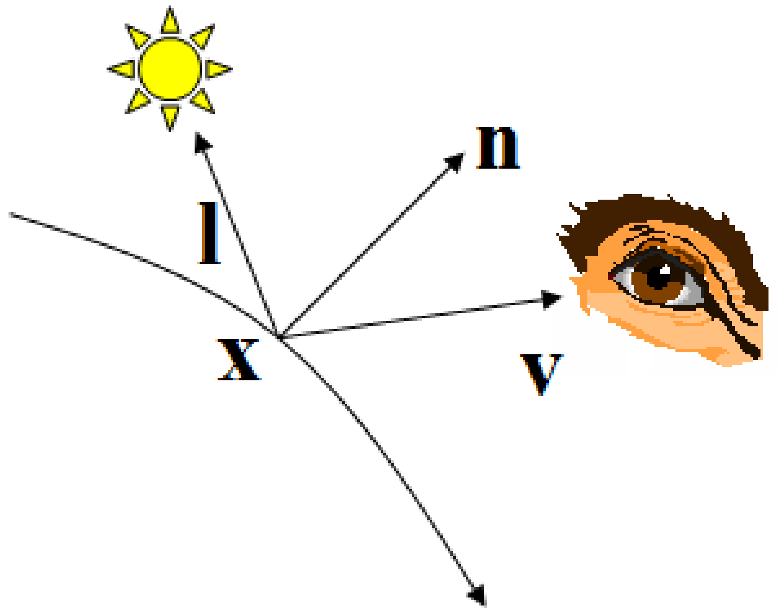
- So far
  - Given a triangle and a viewpoint in 3D, the right pixels can be set
  - Given a constant solid color, the pixels can be colorized accordingly
- Question
  - Can we create better images?
- Attempt to create a realistic image
  - Simulate lighting of the surfaces in the scene
  - Fundamentally: simulation of physics and optics
- Approach in CG
  - Use approximations
    - Physically correct calculations are too expensive for real-time graphics!

ohm

Lighting

Local illumination model

- Ambient light
  - Indirect light from surrounding
  - Modeled by constant color
- Diffuse light
  - Reflection from rough surfaces
  - Uniform in all viewing directions
- Specular light
  - Reflection from "glossy" but not perfectly mirroring surfaces
  - somewhere in "between" diffuse and mirroring



$$L_{total} = L_{ambient} + L_{diffuse} + L_{specular}$$

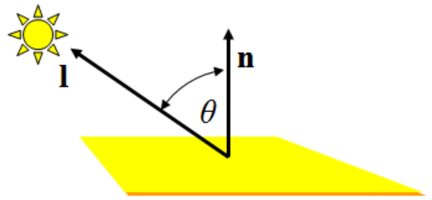
Lighting

Local illumination model

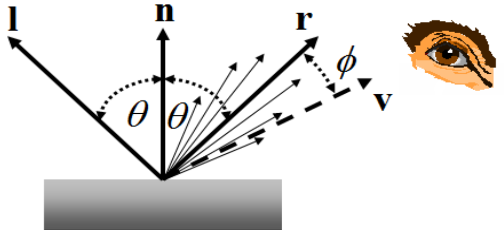
- Ambient light
  - $L_{amb} = k_{amb} I_{amb}$

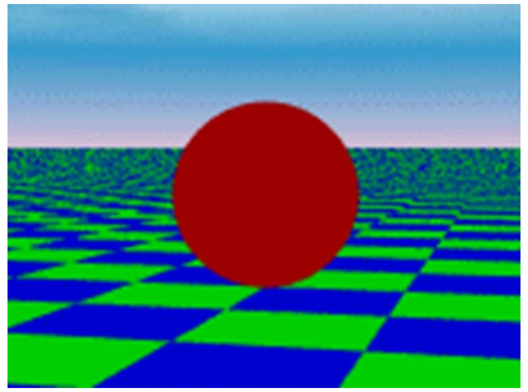
For unit-length normal vectors

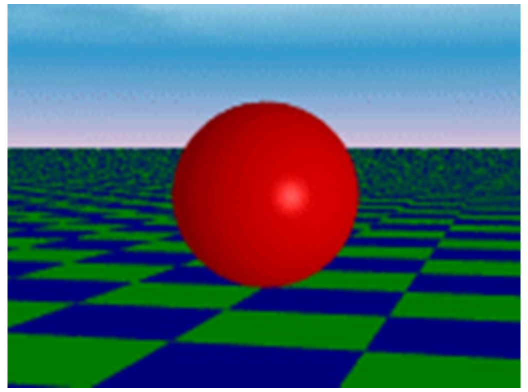
- Diffuse light
  - $L_{diff} = k_{diff} I_{in} \cos\theta$ or $L_{diff} = k_{diff} I_{in} (\mathbf{n} \bullet \mathbf{l})$



- Specular light (Phong Lighting)
  - $L_{spec} = k_{spec} I_{in} (\cos\phi)^{n_{shiny}}$
  - $L_{spec} = k_{spec} I_{in} (\mathbf{v} \bullet \mathbf{r})^{n_{shiny}}$
  - with $\mathbf{r} = (2(\mathbf{n} \bullet \mathbf{l}))\mathbf{n} - \mathbf{l}$

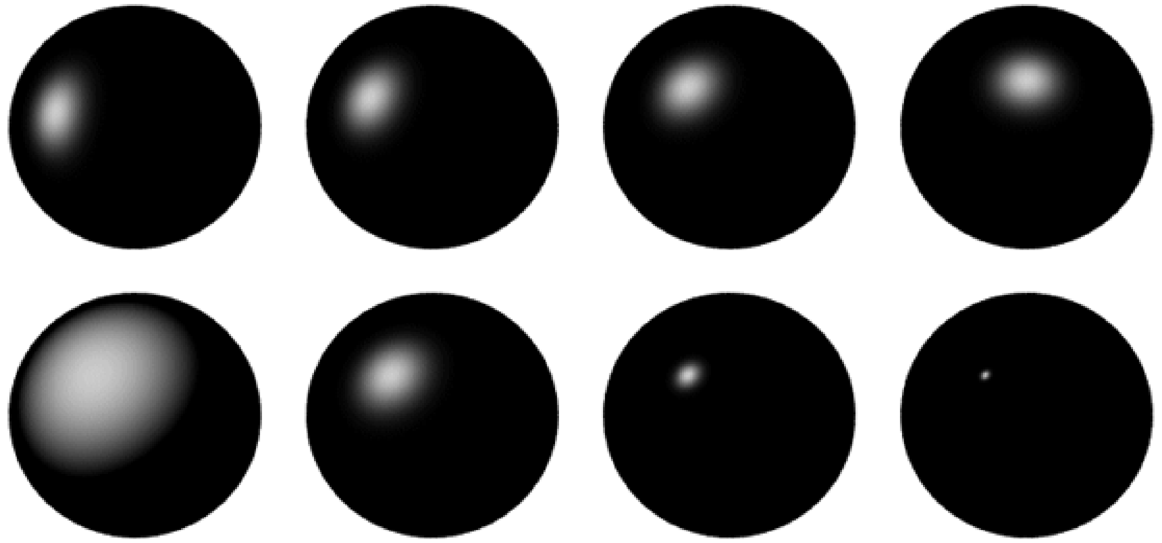






Varying $\vec{l}$

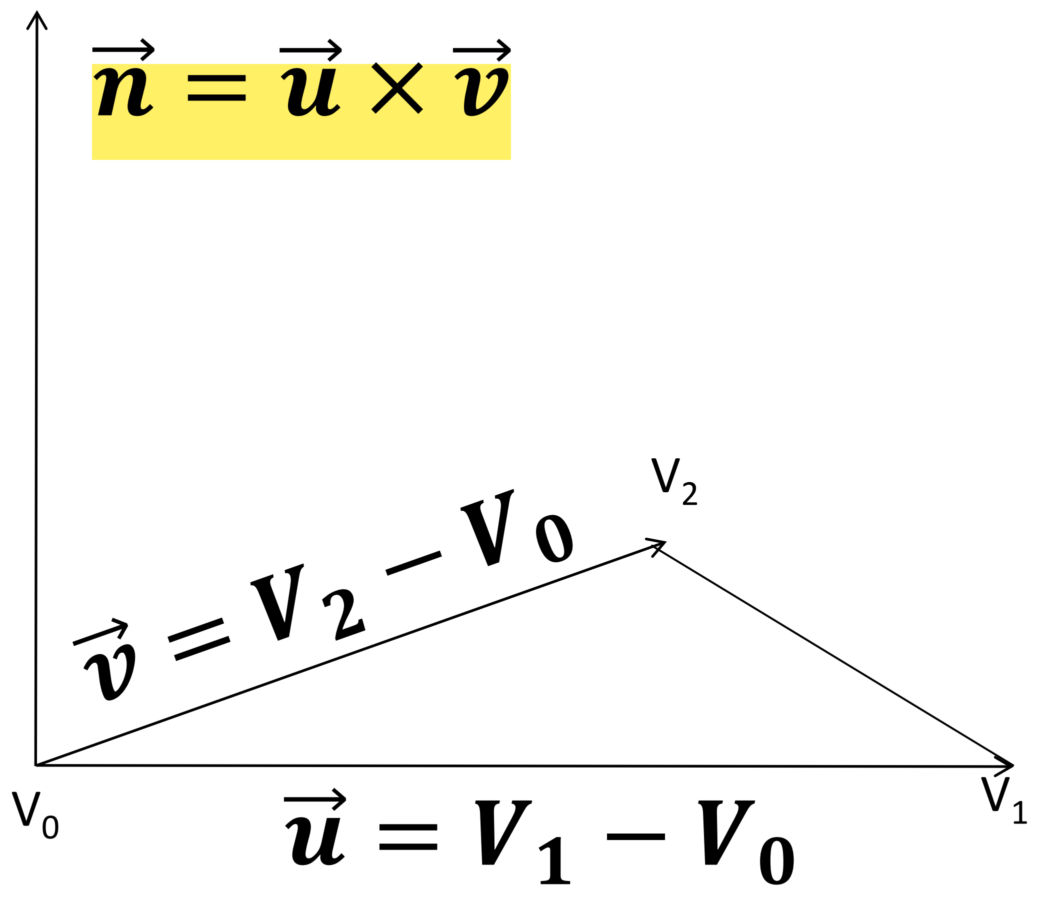
Varying n_{shi}



Lighting

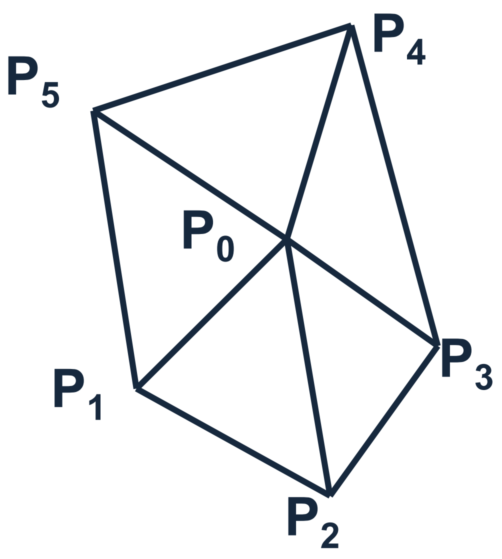
Calculation of normal vectors

- Note: direction of normal depends on vertex orientation



Lighting

Surface / mesh normals

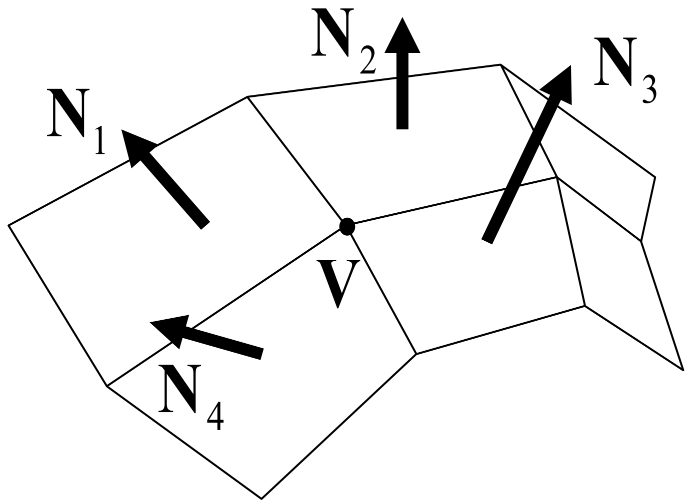


- Newell's method for polygonal meshes
  - More robust than simple cross product
  - Works also for general polygons

$$\mathbf{N}_0 = \sum_{i=1}^{n} \mathbf{P}_i \times \mathbf{P}_{i+1}, \quad \mathbf{P}_{n+1} = \mathbf{P}_1$$

- Average normals of faces to get surface point normal

Mitteln der Oberflächen Normalen, um Kanten runder ausschauen zu lassen

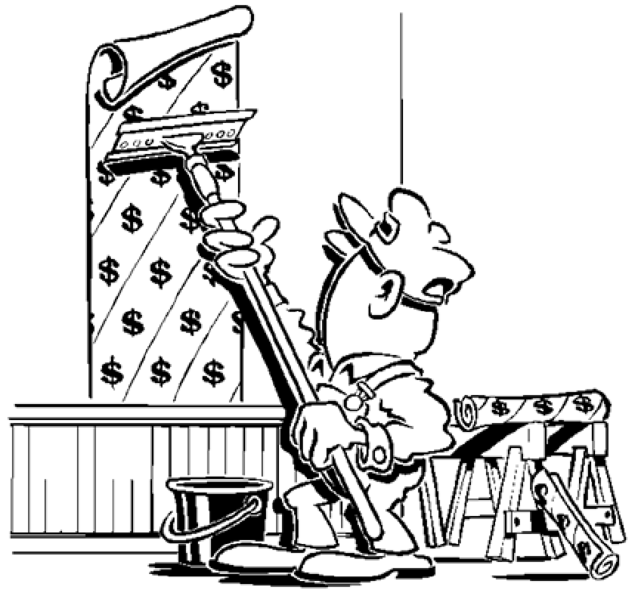


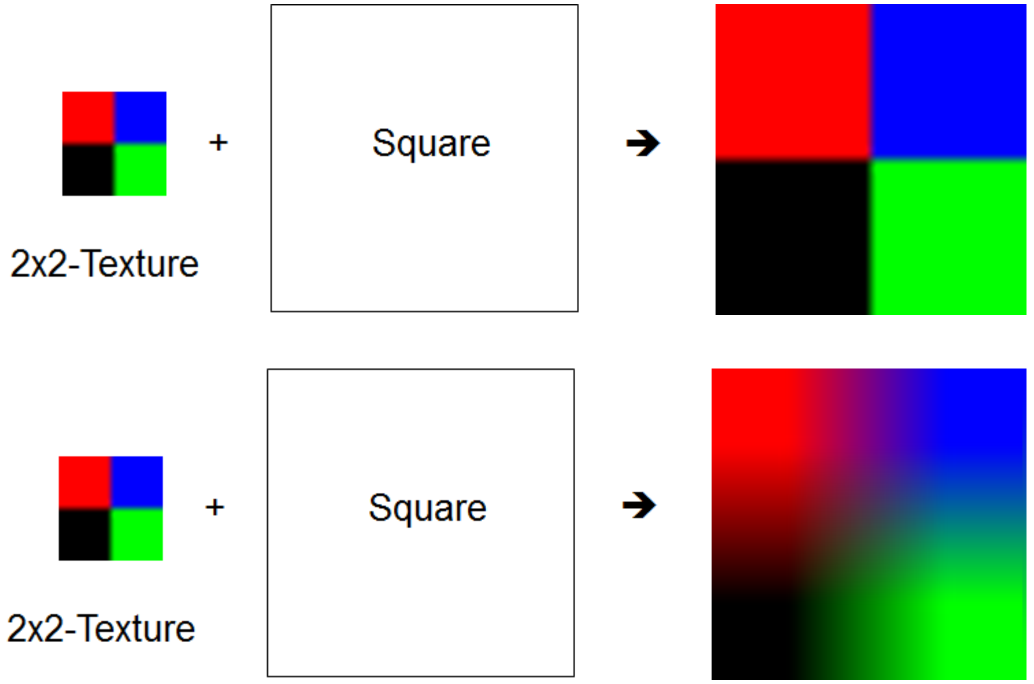
$$\mathbf{N}_V = \frac{\sum_{k=1}^{n} \mathbf{N}_k}{\left|\sum_{k=1}^{n} \mathbf{N}_k\right|}$$

2.5 Texture Mapping

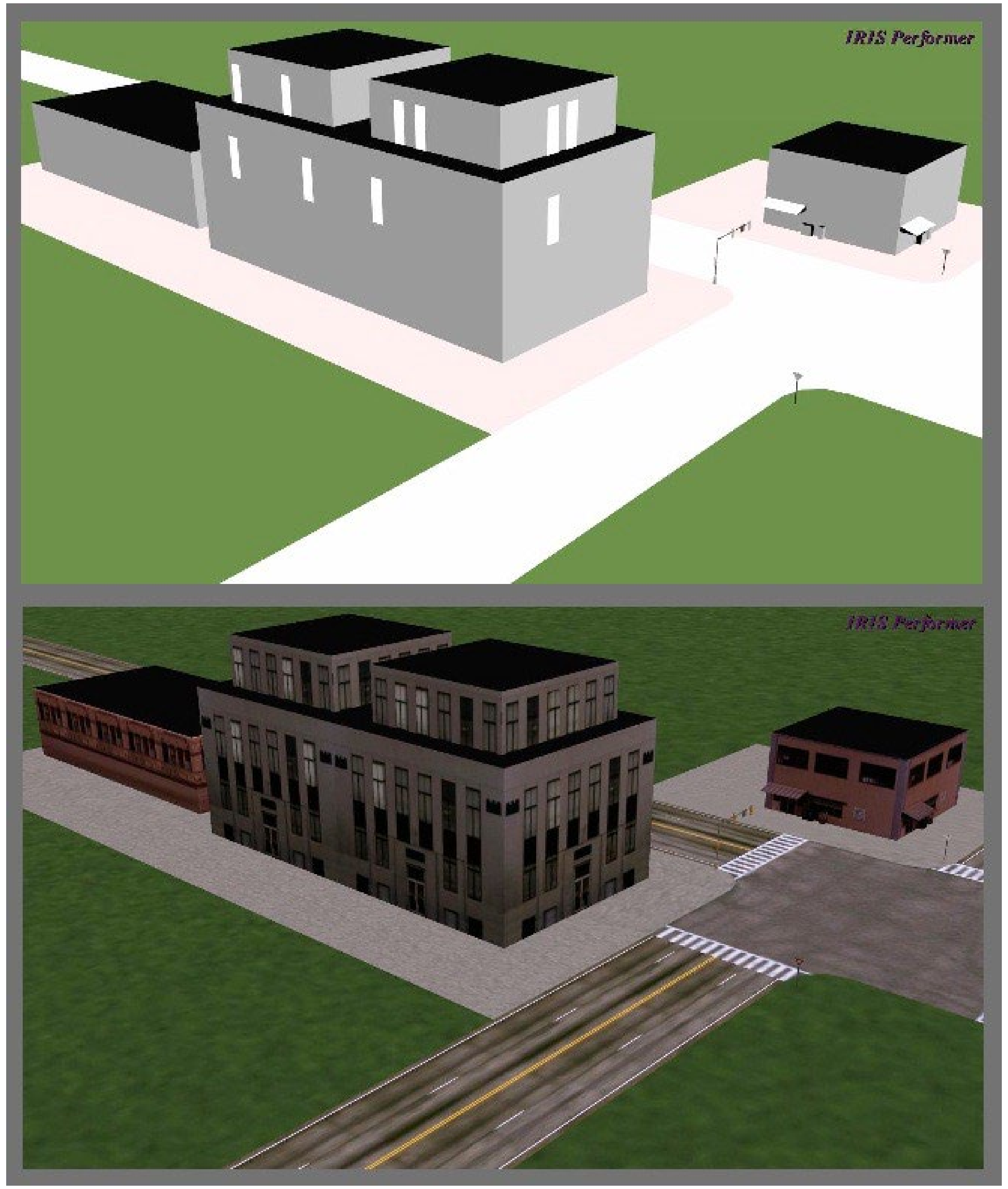
Texture Mapping

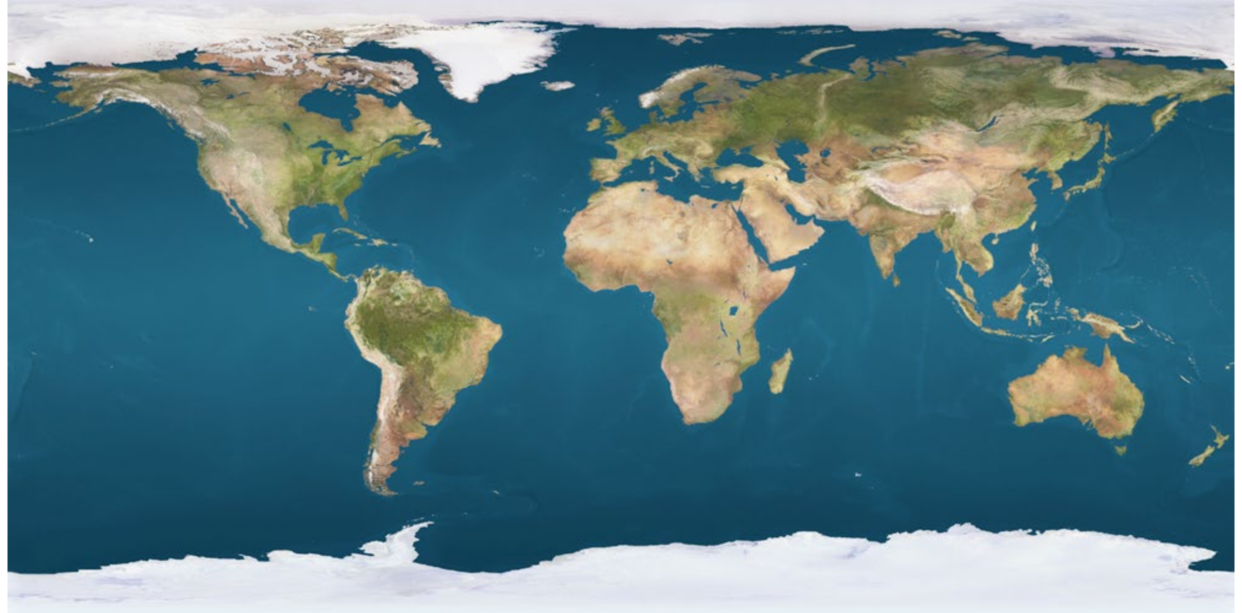
- Textures
  - Functions or pixel-based images containing reflectance information
  - Can be 1D, 2D or 3D (esp. in Vis)
  - Attach "images" to polygons to achieve highly detailed surface at low cost
  - Can be used together with lighting





Texture Mapping





ohm

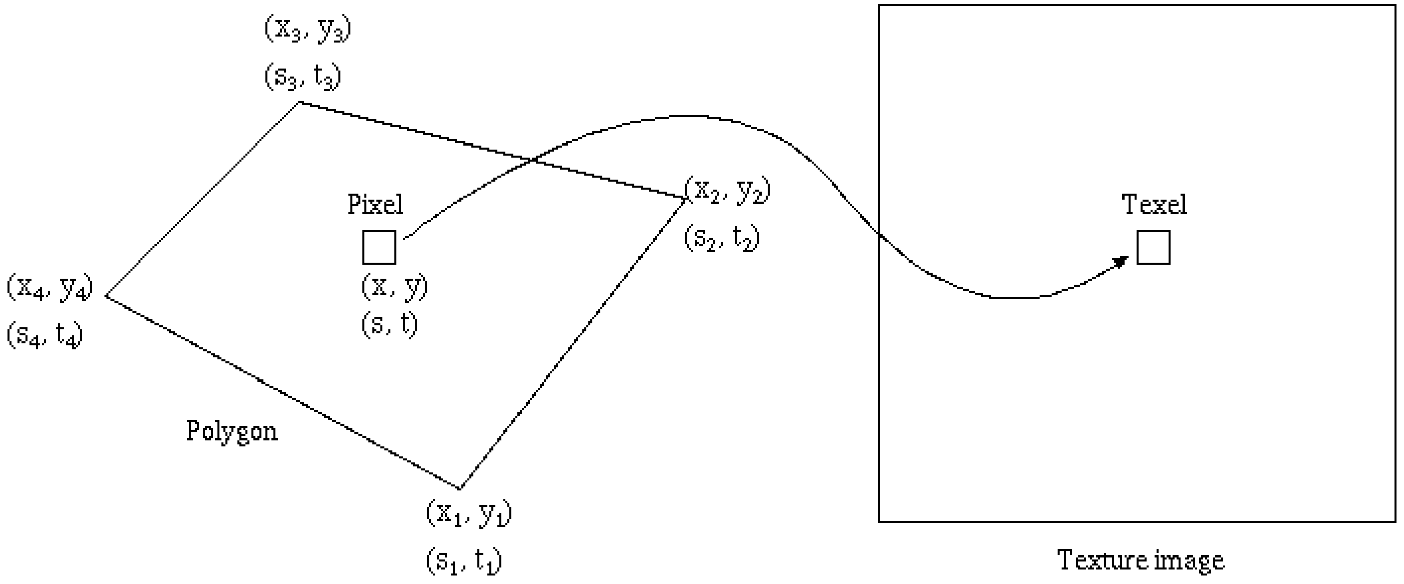
Texture Mapping

- Assign reflectance information to pixels by lookup
  - Texture: typically 2D pixel image (may be 1D or 3D)
    - Determines appearance of a surface
  - Texel
    - Pixel in a texture
  - Requires procedure to map the texture onto the surface (mapping)
    - Easy for single triangle
    - Complex for arbitrary surface in 3D

Texture Mapping

- Mapping in 2D
  - Domain of texture = $[0,1]^2$
  - Image of size n_x x n_y mapped to $[0,1]^2$
    - Usually (but not necessarily) constrained to powers of two
  - Texture coordinates u and v as vertex attributes
    - Assign to each (x,y,z) one (u,v)
  - Usually a two-step process
    - Assignment for every vertex of a mesh
    - Interpolation (later) of interior points
  - Different possibilities for mapping to $[0,1]^2$ space
    - Clamping, wrapping, periodic extension, ...
  - Texture information is read by some *interpolation* function

Texture Mapping



Source: http://idav.ucdavis.edu/~okreylos/PhDStudies/Winter2000/TextureMapping.html, http://www.flipcode.com/archives/Light_Mapping_Theory_and_Implementation.shtml

Texture Mapping

Interpolation and projection