

# Lecture 9: Light, Color and Shadows

DHBW, Computer Graphics

Lovro Bosnar

22.2.2023.

# Syllabus

- 3D scene
    - Object
    - Light
    - Camera
  - Rendering
  - Image and display
- Light
    - Light and color
    - Real world light sources
    - Light models
      - Physical lights
      - Non-physical lights
      - Environment illumination
    - Shadows





Light is important because without light...

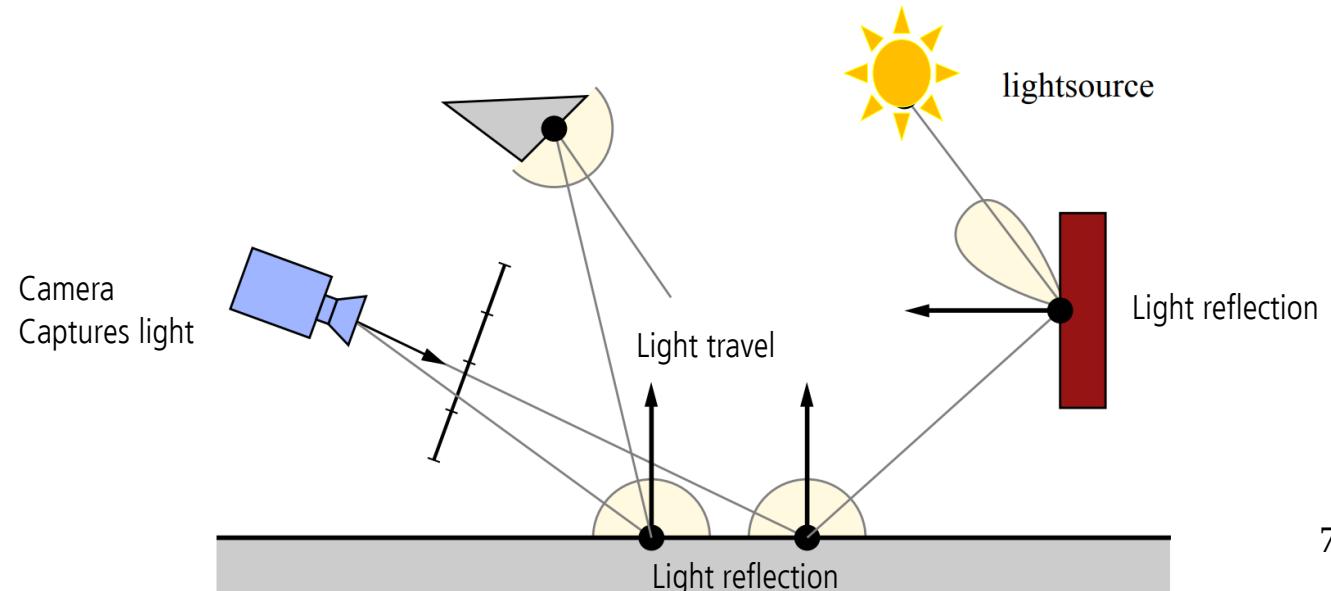


# Light and rendering

- Light is emitted from light sources, travels along rays and interacts with object in the scene (e.g., surface reflection)
- Image created when light falls on the sensor (e.g., camera film)
- This process is described with **rendering equation**

**Forward tracing:**  
tracing from light

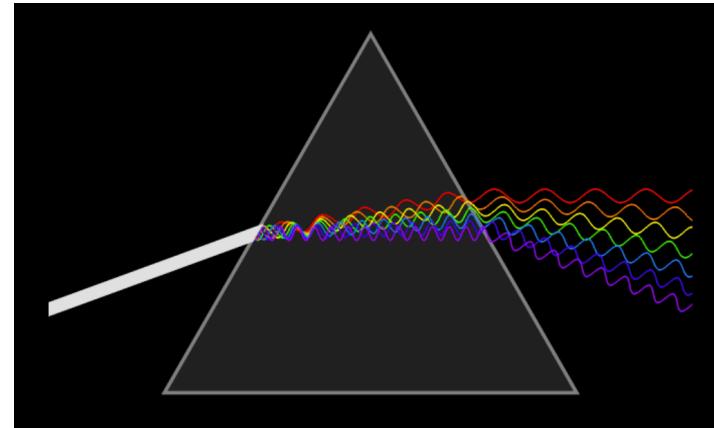
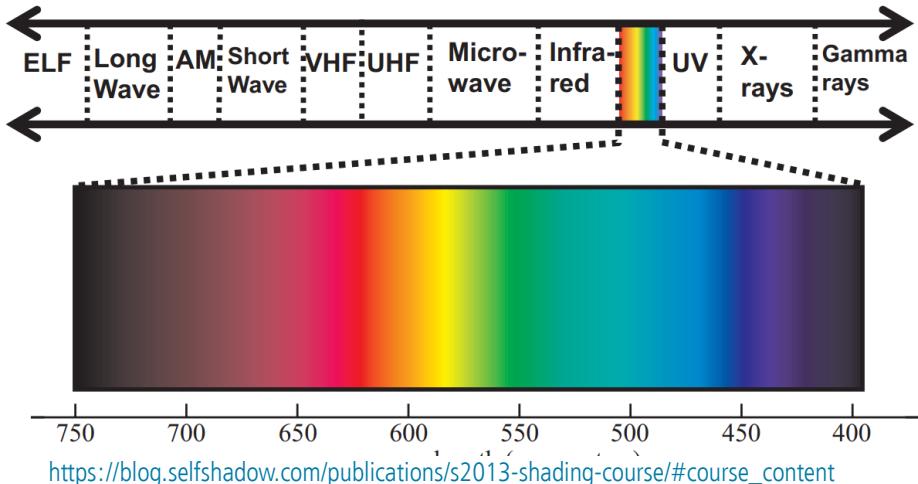
--  
**Backward tracing:**  
tracing from camera



# Light and color

# Light frequency and color

- Light is electromagnetic wave (or particle) visible by eye
  - Visible spectrum: wavelength 400 – 700 nm
- Different wavelengths (frequencies) are perceived as different color
  - color is perceptual (psychophysical) phenomena rather than physical one



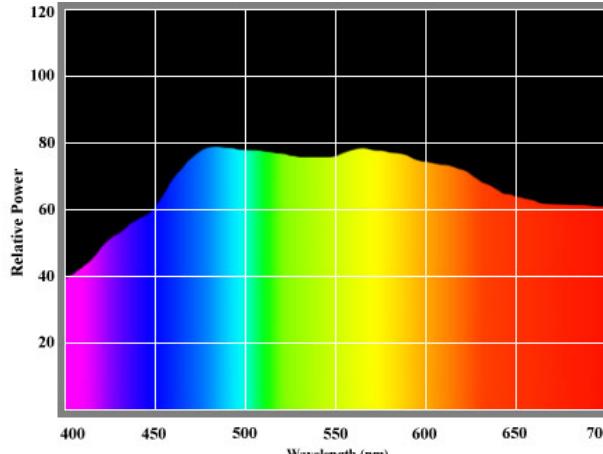
White light is made of all visible colors  
[https://en.wikipedia.org/wiki/Dispersive\\_prism](https://en.wikipedia.org/wiki/Dispersive_prism)

# Light and spectral power distribution

- Sun (midday sun Western/Northern Europe) is reference for **natural white light**
  - **Daylight illuminant - D65** by CIE - International Commission on Illumination
- Light is characterized by **spectral power distribution (SPD)**
  - Range and intensity of colors light produces at each wavelength in the visible spectrum



<https://commons.wikimedia.org/wiki/File:Sun-in-the-sky.jpg>



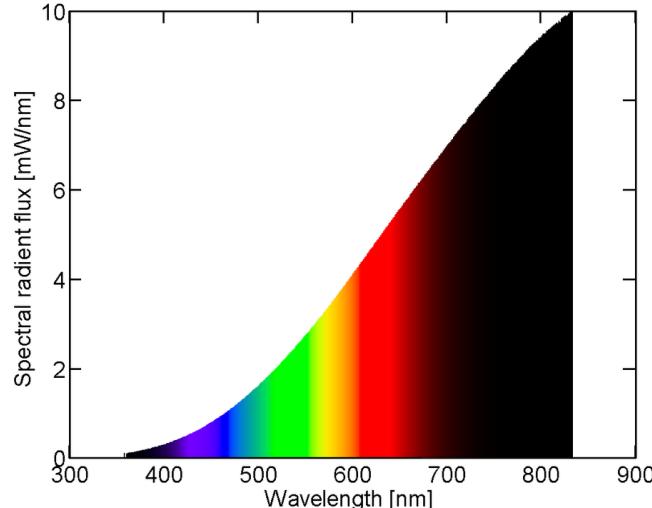
Sun and Sky - Approximately 6,000K  
<http://www.openphotographicsociety.org/photography/colour-light-and-optics/colour/203-spectral-power-distribution>

# Spectral power distribution

- Light sources can be grouped into either natural or artificial light sources.
- Color quality of artificial forms of lighting is compared to natural light
  - Color rendering index: ability of light source to reproduce color of object seen under D65 (natural light)



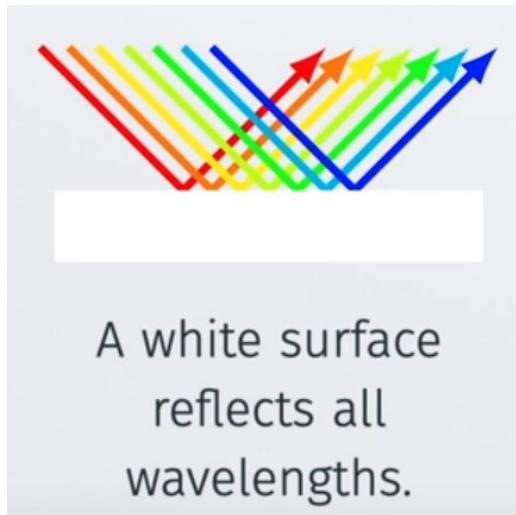
[https://en.wikipedia.org/wiki/Incandescent\\_light\\_bulb](https://en.wikipedia.org/wiki/Incandescent_light_bulb)



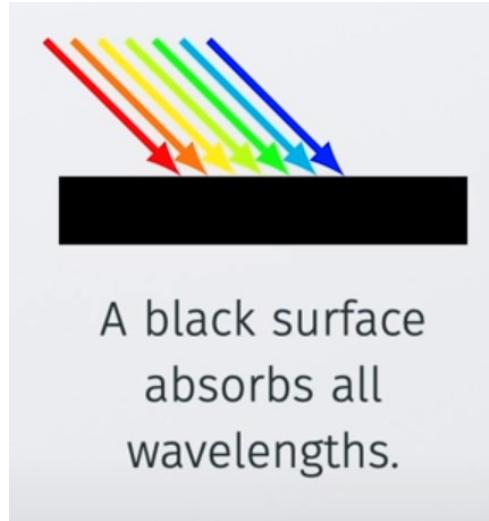
SPD of incandescent light source. Light is perceived as yellow since eye is less sensitive to red.

# Material reflectance

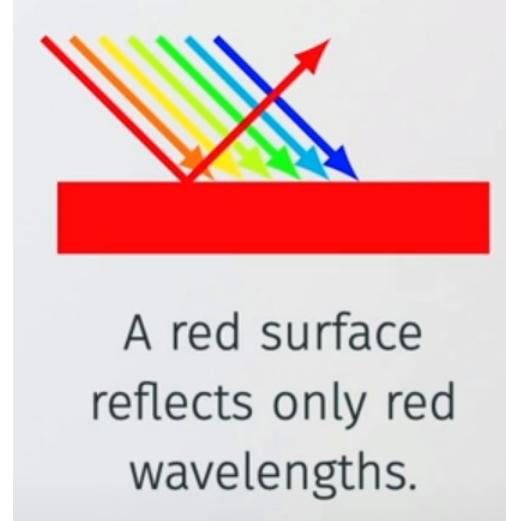
- Light falling on opaque surface will reflect and absorb
- Reflected light is perceived as color of object surface
- Different materials are characterized with different **spectral reflectance curves**



A white surface  
reflects all  
wavelengths.



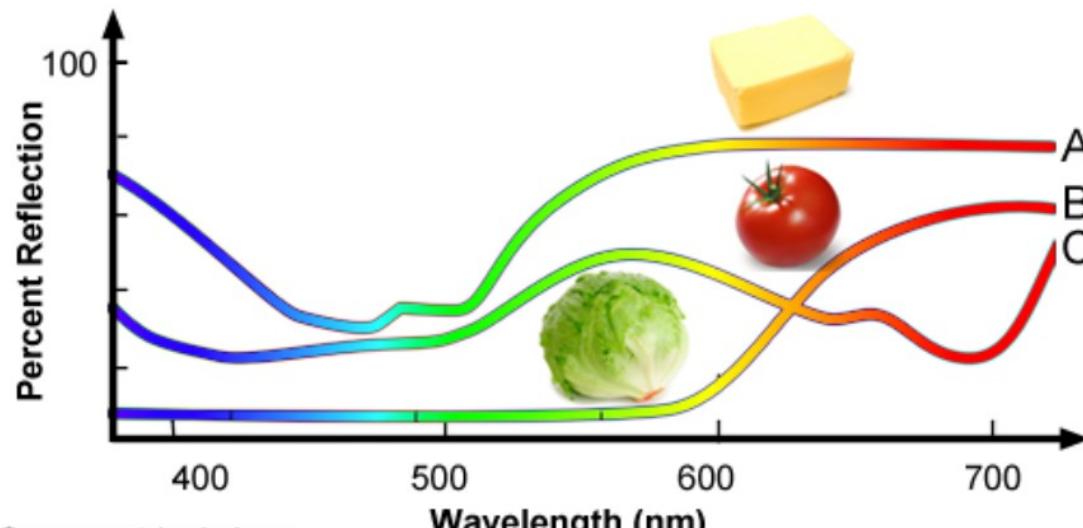
A black surface  
absorbs all  
wavelengths.



A red surface  
reflects only red  
wavelengths.

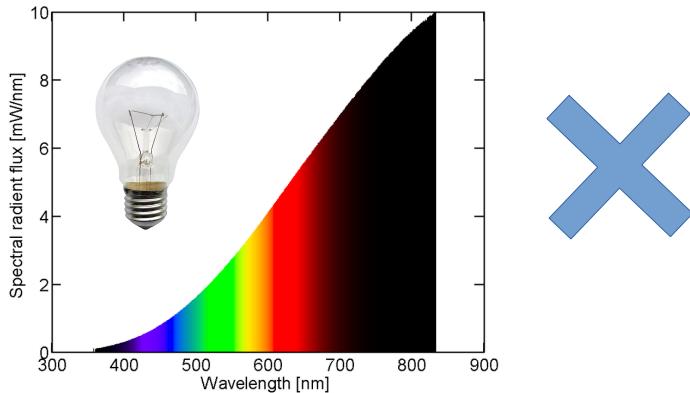
# Spectral reflectance curve

- SPD of light source defines its color and intensity
- Spectral reflectance curve: SPD describing light reflected from objects
  - Fraction of the light reflected by the surface of the object over the amount of white light (D65) illuminating the object

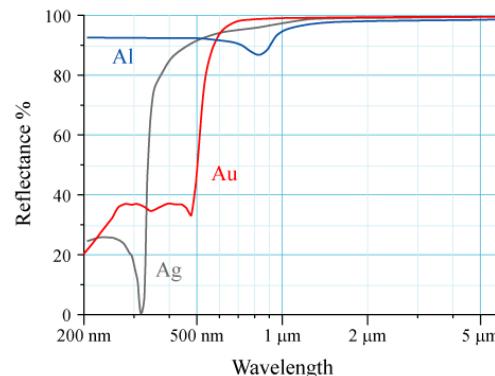


# Material reflectance

- Reflected light energy (color) depends on:
  - Incoming light energy ( $E$ )
  - Reflectance ( $R$ ) or absorption  $A = 1 - R$ ,  $A + R = 1$ 
    - $R$  and  $A$  are spectral values, but in graphics we use RGB values

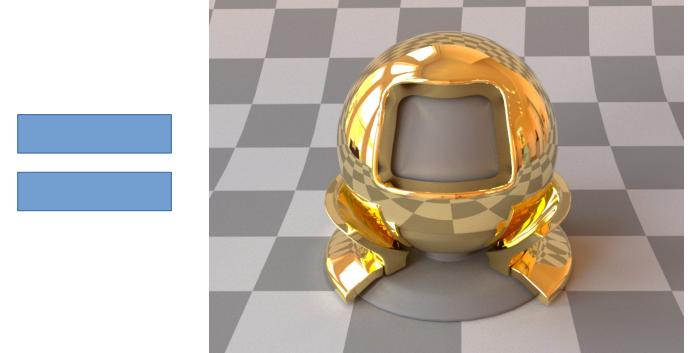


Power spectral distribution of light → incoming light energy ( $E$ )



Spectral reflectance curve → Material reflectance ( $R$ )

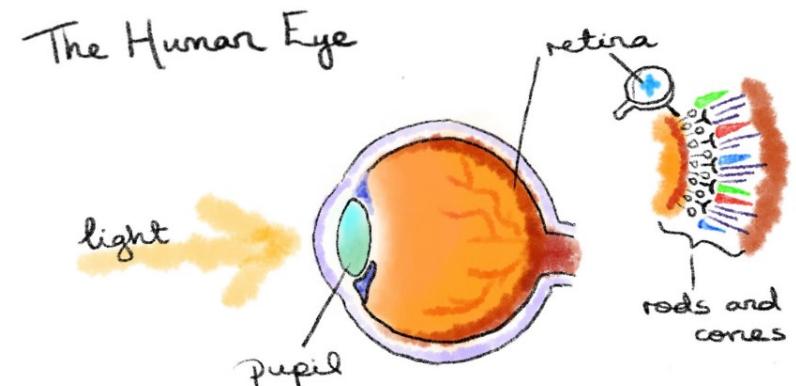
<https://en.wikipedia.org/wiki/Reflectance>



$$\text{Reflected} = E * R$$

# Light and human visual system

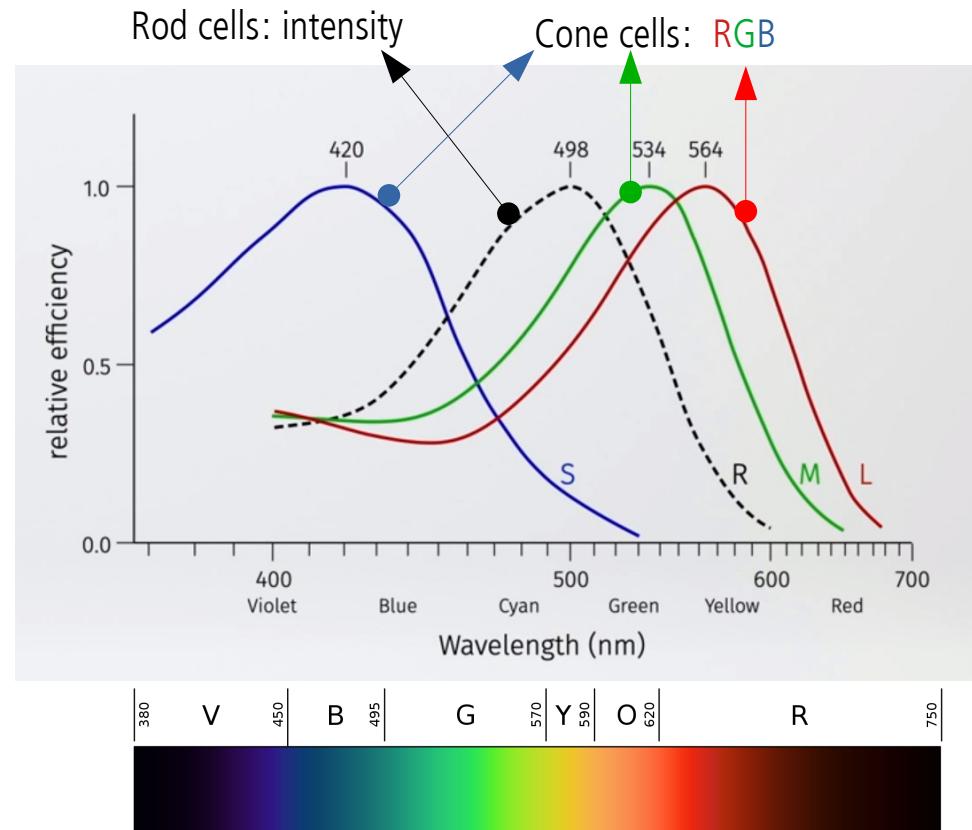
- Human visual system; back of eye (the retina) is covered with light sensitive receptors – **photo-receptive cells**
- Cone cells – responsible for trichromatic color vision
  - “Active” in well lit environments (e.g., daylight)
  - Three types sensitive to: **red, green and blue light**
- Rod cells – sensitive to smaller amount of light intensity but not reproducing colors
  - “Active” in low lit environments (e.g., night)



<http://daisymoliving.github.io/2018/07/30/design-fundamentals-colour.html>

# Light and human visual system

- Cone cells
  - Trichromatic color vision (RGB)
  - Well-lit scenes
- Rod cells
  - Intensity
  - Low-lit scenes
- Human visual system transforms light radiation into color
- In graphics we need way of transforming physical characterization of light (SPD) into color

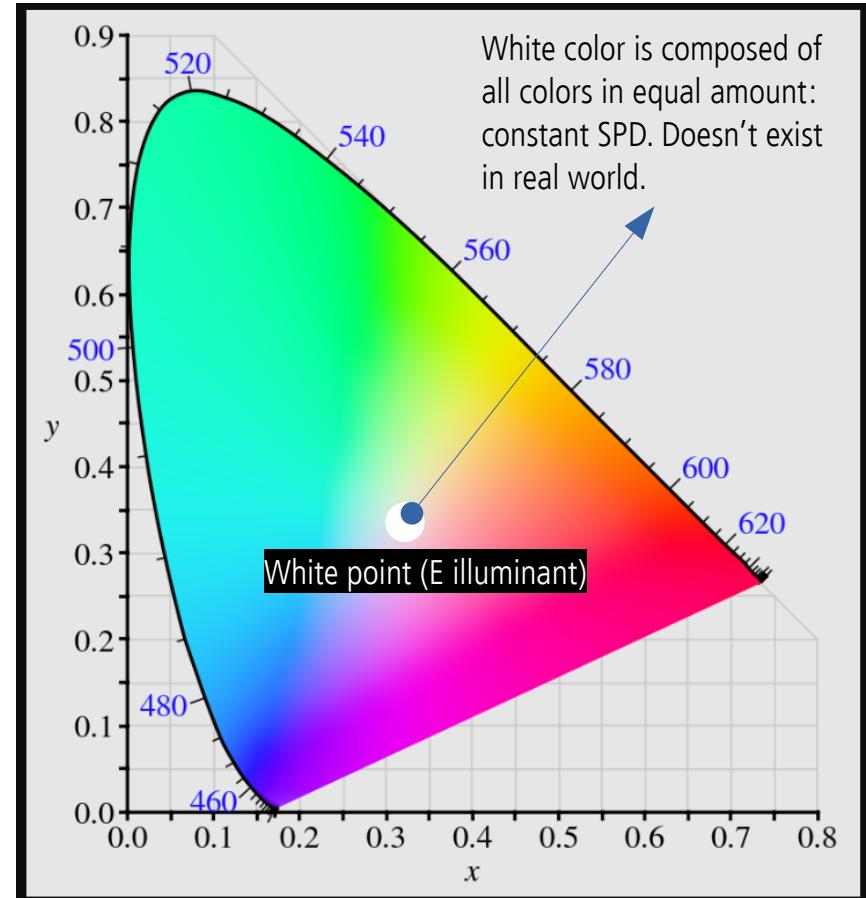


# From SPD to color

- Light from light source is characterized with spectral power distribution (SPD)
- Light reflected from object is defined with SPD; spectral reflectance curve
- Specifying colors as wavelengths is impractical and in most cases not needed
  - For rendering we will ignore wave effects such as diffraction, interference and polarization will be ignored
  - Colors must be represented so that they can be visualized to human visual system
- SPDs must be transformed in color spaces suited for trichromatic vision
  - SPD is transformed in color hue and color brightness values
  - Light is described with rays which “carry” intensity (brightness) and color (hue)

# Color space and gamut

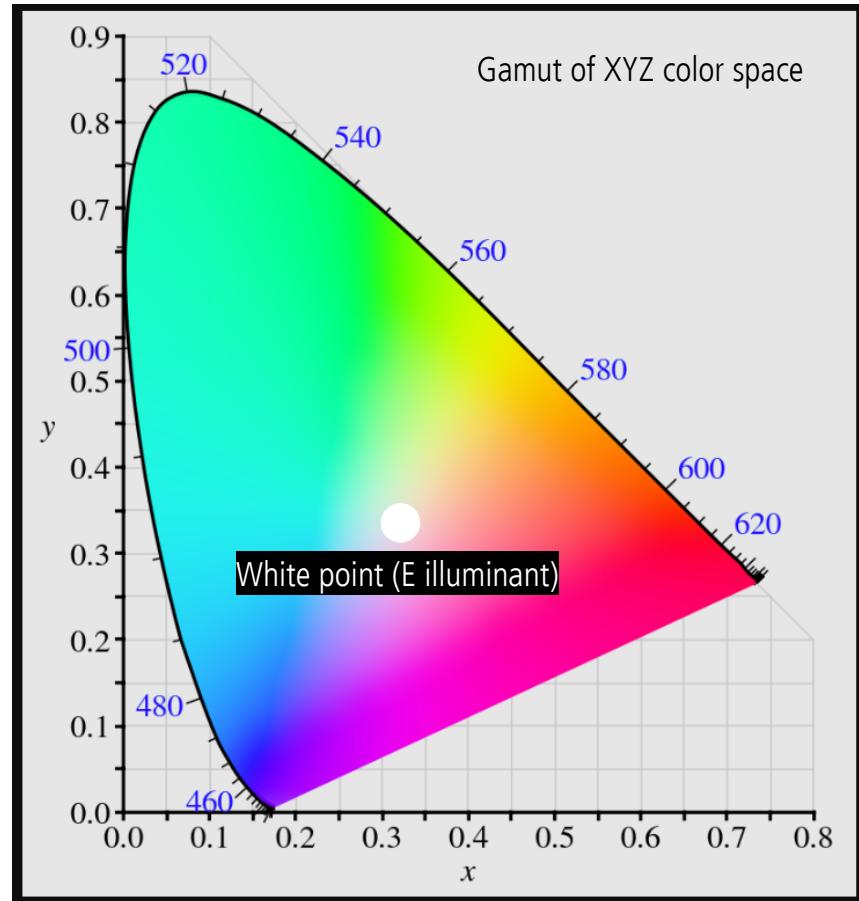
- **Colorspace**
  - Model to represent colors perceived by human visual system
  - Way of representing colors through numerical values (e.g., image pixels or calculation)
  - **CIE XYZ** is foundation for all color spaces
  - **RGB** model is most important for computer graphics
- **Gamut**
  - Range of possible colors that can be represented by particular colorspace



Gamut of XYZ color space:  
[https://en.wikipedia.org/wiki/CIE\\_1931\\_color\\_space](https://en.wikipedia.org/wiki/CIE_1931_color_space)

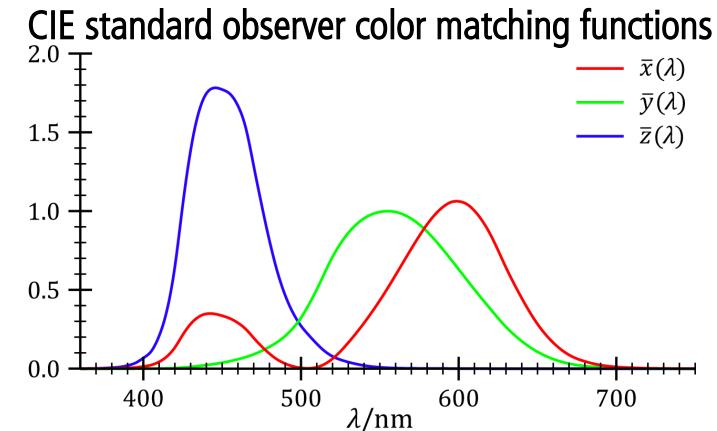
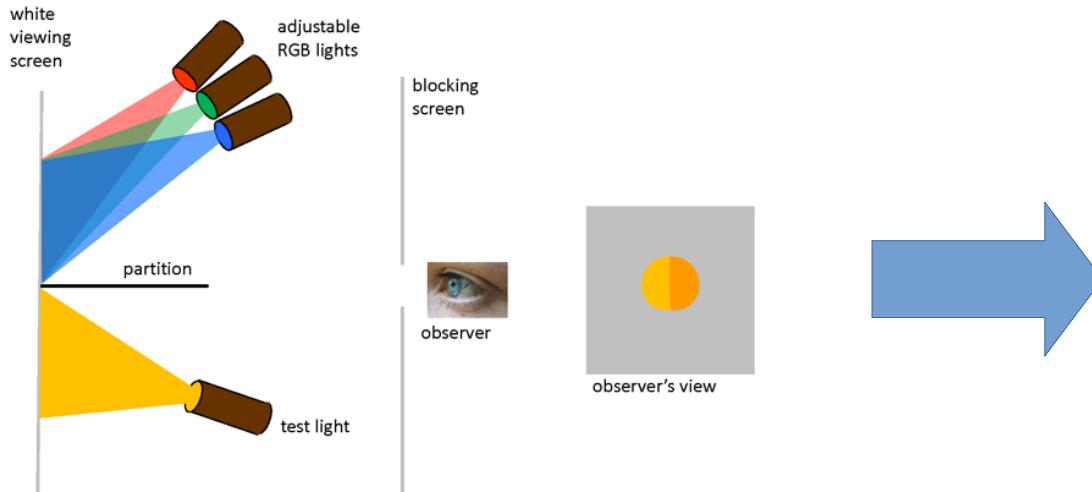
# XYZ colorspace

- Trichromatic system motivated idea that any color can be represented by combination of three values
  - Artificial additive primary colors (X, Y, Z)
- XYZ values would serve as basis for converting SPD to color spaces visible to human visual system
- **Gamut of XYZ color space** - full range of possible colors visible by human visual system
- To find transfer function from SPD to XYZ colorspace, **color matching experiment** was performed



# Color matching experiment

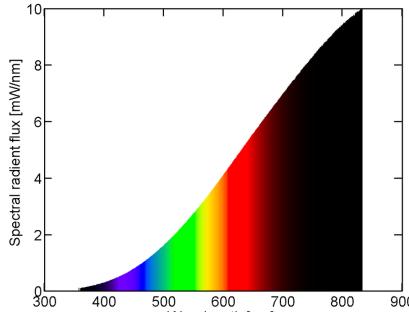
- Any color perceived by human visual system can be achieved by combining R, G, B signals (trichromatic theory of color vision)
- Colormatching experiment used trichromatic theory to form **color matching functions**



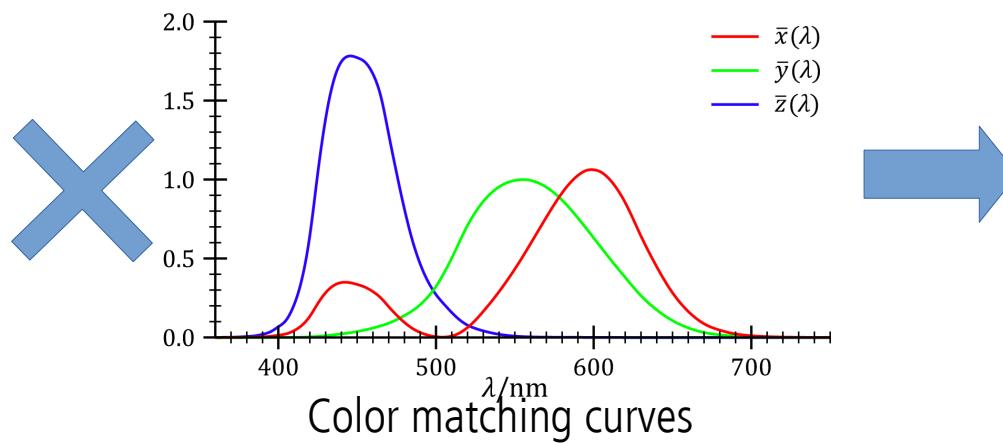
Y curve is close to luminosity function  
expressing the brightness

# From SPD of light source to XYZ

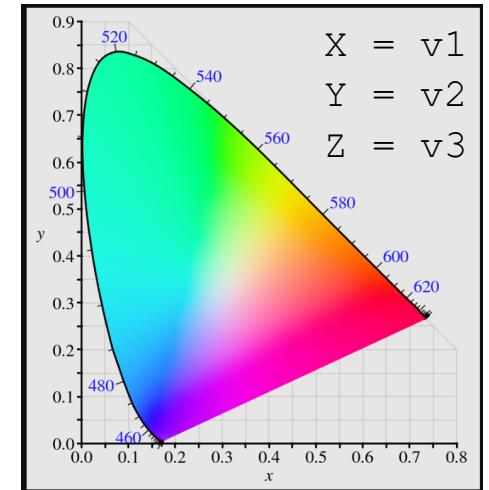
- XYZ colorspace is obtained from SPD of light source using color matching functions
  - SPD and the color-matching functions are in tabular form



SPD of light source



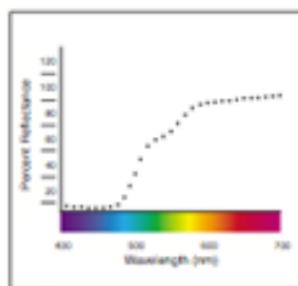
[https://en.wikipedia.org/wiki/CIE\\_1931\\_color\\_space](https://en.wikipedia.org/wiki/CIE_1931_color_space)



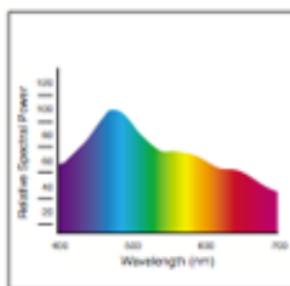
XYZ color space

# From SPD of reflected light to XYZ

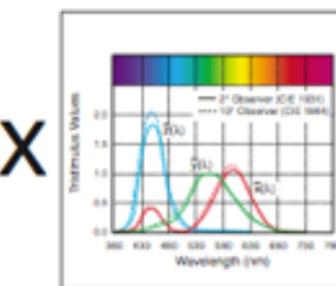
- Light reflected from object is represented as spectral reflectance curve
  - Fraction of the light reflected by the surface of the object over the amount of white light (D65) illuminating the object



X



X



=

$$\begin{aligned} X &= 62.04 \\ Y &= 69.72 \\ Z &= 7.34 \end{aligned}$$

Spectral  
Reflectance  
Curves

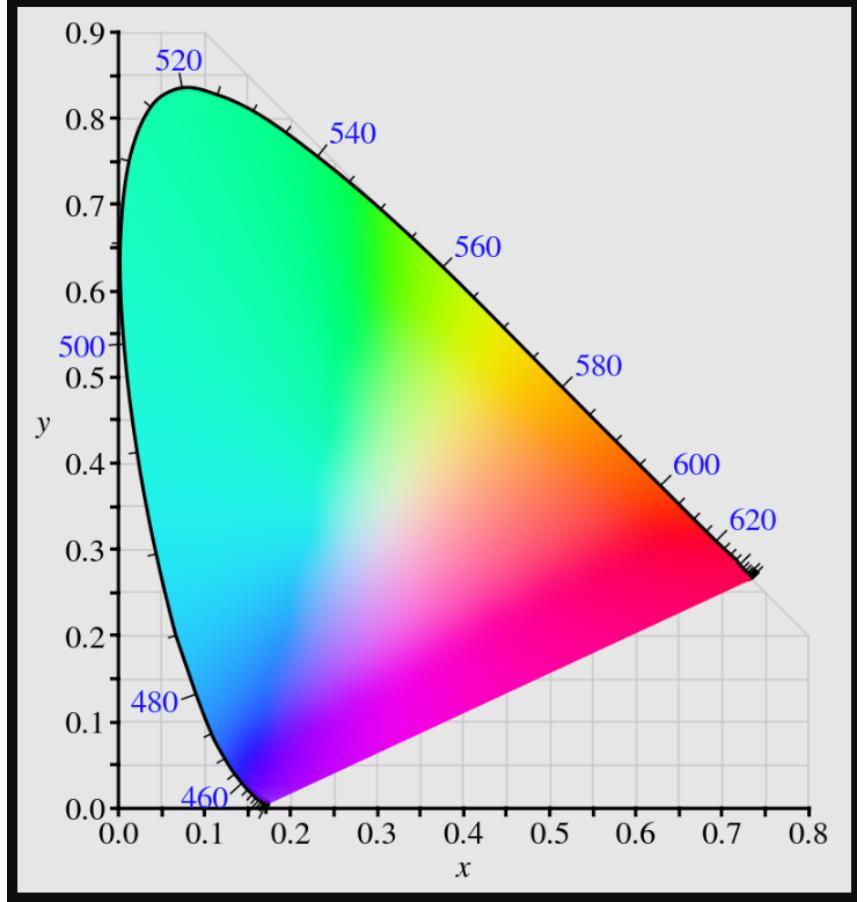
D65 Illuminant

CIE Color  
Matching Functions

Tristimulus  
Values

# XYZ color space

- International, device-independent standard for color specification
- System capable representing all colors human visual system can perceive
- Gold standard for color storage
- **XYZ have to be transformed to, e.g., RGB space to be visualized**

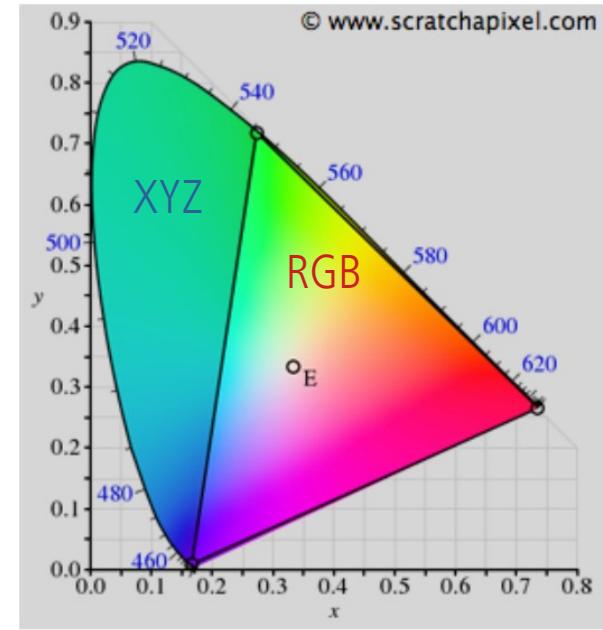
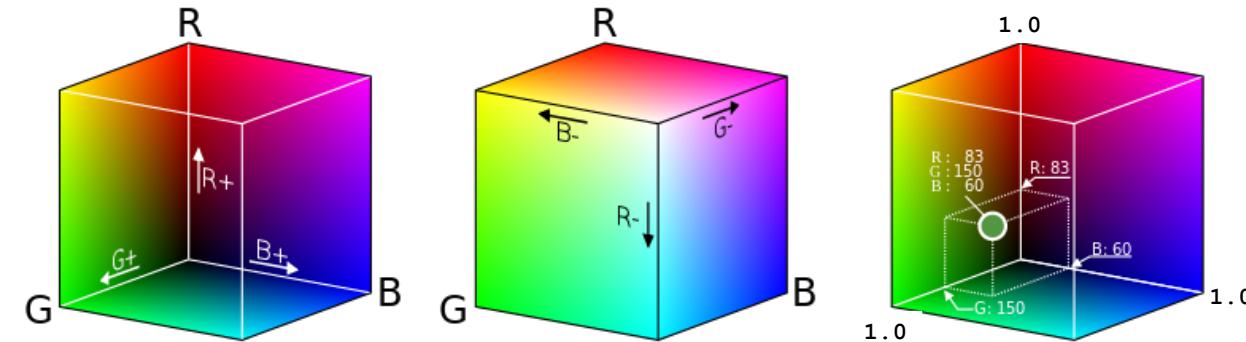


$X + Y + Z = 1$ , gamut can be visualized using:

$$x := \frac{X}{X+Y+Z}, \quad y := \frac{Y}{X+Y+Z}$$

# RGB colorspace

- Similar as human visual system: represent colors as simple sum of primary colors (**R**, **G**, **B**)
- Visualized as cube where **R**, **G**, **B** are placed on axis
  - Point inside cube represents one color
- **R**, **G**, **B** intensities are in  $[0, 1]$



XYZ vs RGB gamut

# XYZ to RGB

- XYZ and RGB are **linear color spaces** defined with three coordinates
  - Define a 3D space (e.g., cube for RGB)
- Conversion of color from XYZ to RGB can be seen as moving a point from one 3D space to another
  - **Linear transformation** performed using **3x3 matrix**
  - Color is not changed, only expressed in different colorspace

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.4124564 & 0.3575761 & 0.1804375 \\ 0.2126729 & 0.7151522 & 0.0721750 \\ 0.0193339 & 0.1191920 & 0.9503041 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

[http://www.brucelindbloom.com/index.html?Eqn\\_RGB\\_XYZ\\_Matrix.html](http://www.brucelindbloom.com/index.html?Eqn_RGB_XYZ_Matrix.html)

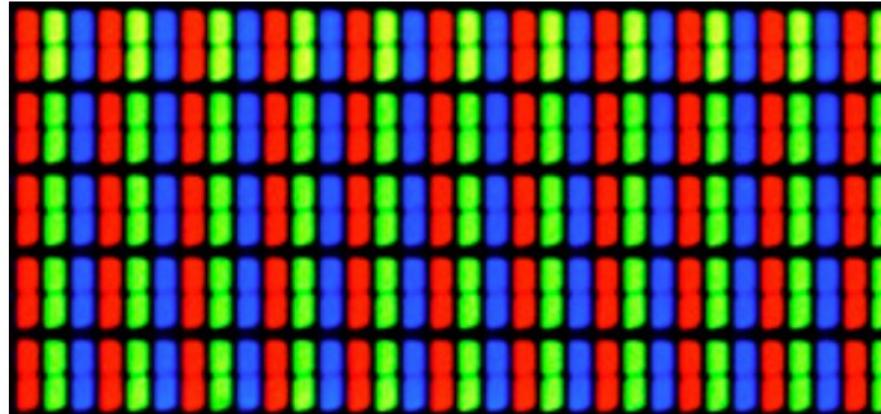
<http://www.russellcottrell.com/photo/matrixCalculator.htm>

<https://www.oceanopticsbook.info/view/photometry-and-visibility/from-xyz-to-rgb>

<https://terathon.com/blog/rgb-xyz-conversion-matrix-accuracy/>

# RGB colorspace

- Most **computer screens technology** is based on systems similar to RGB color model.
  - Additive colorspace: summing all RGB values gives white color
- **Most rendering systems work with RGB colorspace**
  - RGB is **linear** color space → linear property is required for correct rendering (shading) computation

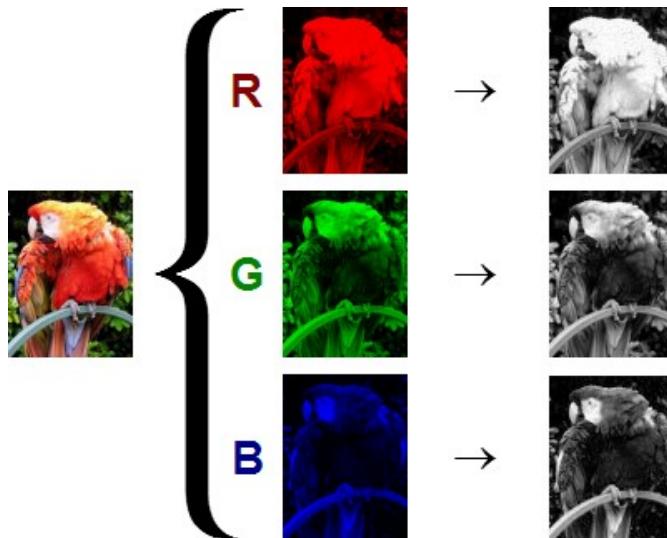


Close-up of LCD screen:

<https://jakubmarian.com/the-illusion-of-rgb-screens/>

# RGB colorspace

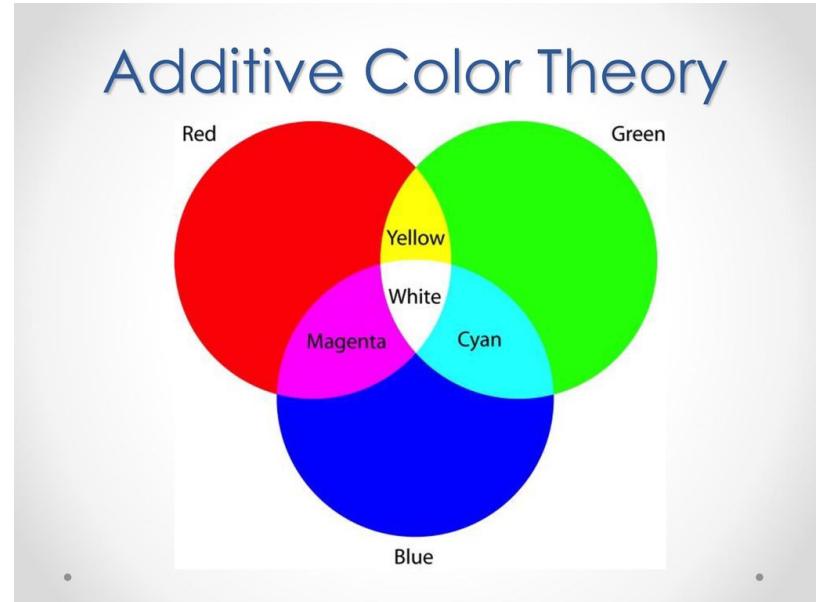
- Typical image formats store three intensities in three channels: R, G, B.
  - e.g., standard image format PNG: 8 bit per channel [0,255] range
    - $2^{(8 \times 3)} = 16.7$  million colors (term: true color)
  - e.g., HDR images: 10 or 12 bit per channel



[https://www.mathworks.com/help/matlab/creating\\_plots/image-types.html](https://www.mathworks.com/help/matlab/creating_plots/image-types.html)

# Additive colorspaces

- Any color is created by adding red, green or blue
- White color is made of red, green and blue
- Usage: display devices, computer science, lighting technology → **colors are created with light emission**
- Display device (e.g., monitor) can further increase or decrease the overall global image intensity (brightness)



<https://carsonparkdesign.com/what-makes-white-on-the-screen/>

# Subtractive colorspace

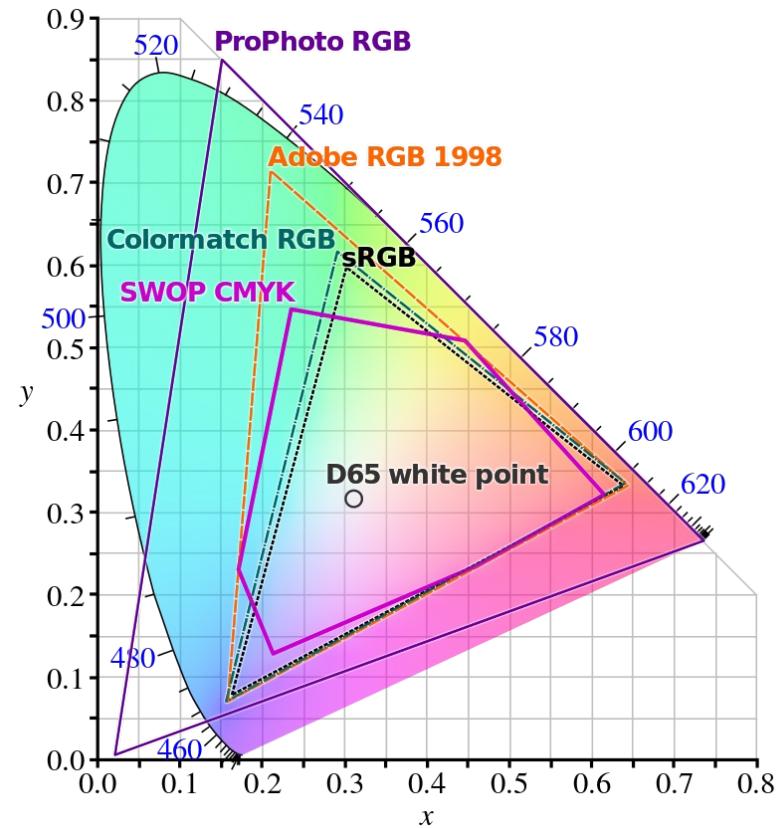
- Main colors: **cyan, magenta, yellow** - white light is made by subtracting
- Usage: art, painting, printing
- Color from canvas is not a result of light emitted from canvas, but light reflected from canvas
  - When white light hits canvas, what is not absorbed is reflected



<https://www.greenleafblueberry.com/blogs/news/modern-primary-colors>

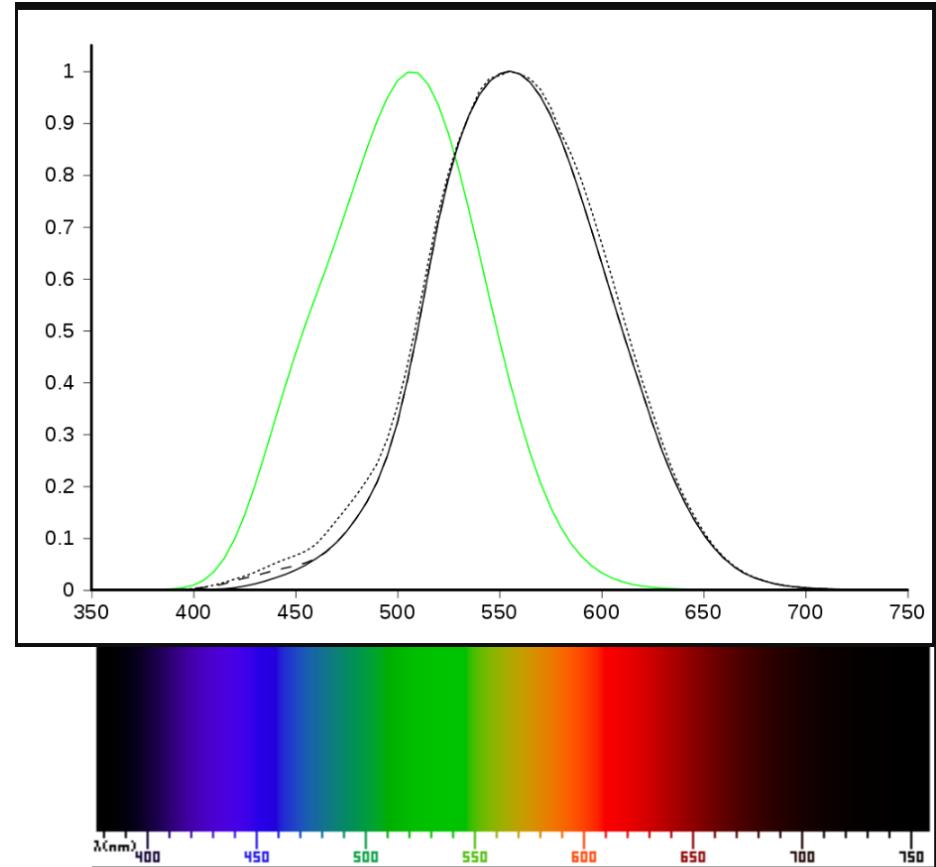
# Other colorspaces

- Different colorspaces represent **different gamut of colors**
  - RGB and CMY have small gamut → device limitations
- RGB and XYZ are **linear colorspace**s (transformed using  $3 \times 3$  matrix)
  - Not all colorspaces are linear, e.g., sRGB, HSV, HSL
  - Any non-linear colorspace must be converted to linear colorspace before  $3 \times 3$  matrix transformation



# Color brightness

- 555 to 560 nm (green towards yellow) are perceived by the human eye as being the brightest
  - Blue color: the dimmest
  - Green color: the brightest
- How bright color appear to human eye:  
**luminosity function**
  - Average sensitivity of human eye to different wavelengths for well lit conditions
- Brightness or lightness is present in **HSV** and **YUV** color models

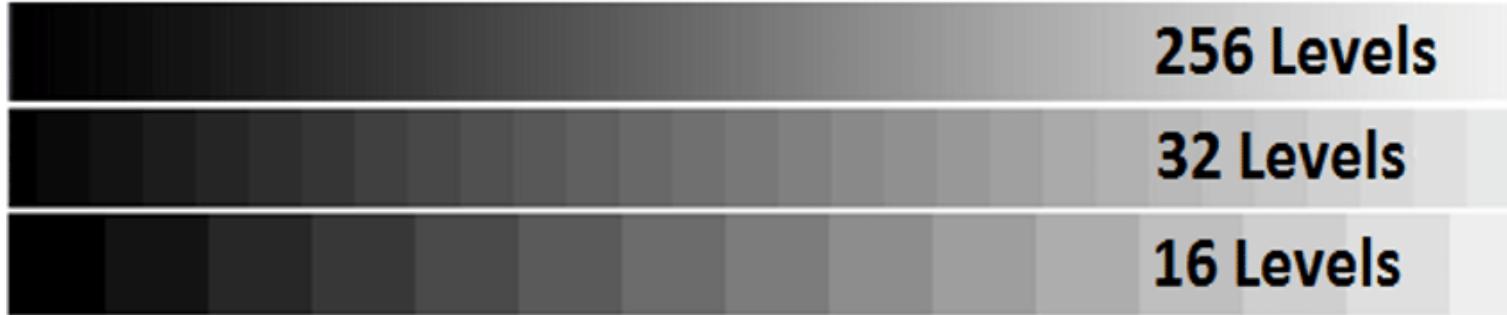


Luminosity function for photopic – green - (well lit conditions) and scotopic (low lit conditions) – black.

[https://en.wikipedia.org/wiki/Luminous\\_efficiency\\_function](https://en.wikipedia.org/wiki/Luminous_efficiency_function)

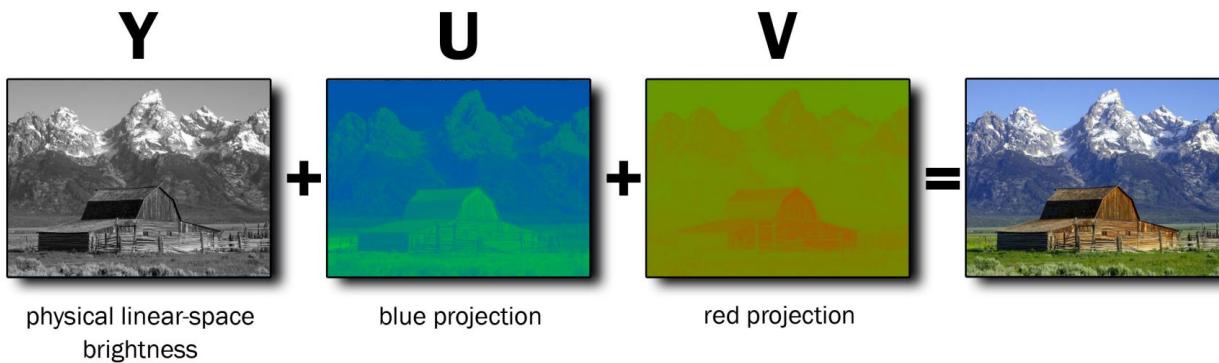
# Brightness (luminance)

- Human eye can very well see small changes in brightness
  - Many levels for brightness is needed to describe it without visual discontinuities
- **Dynamic range:** ratio between image with brightest white level and image with darkest black level
  - Very high ratios are possible in 3D scenes or real world
  - Display devices have limitations in showing such high dynamic ranges



# YUV color space

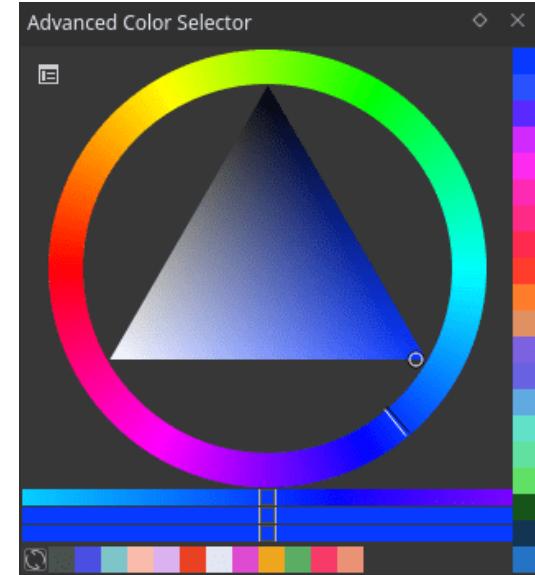
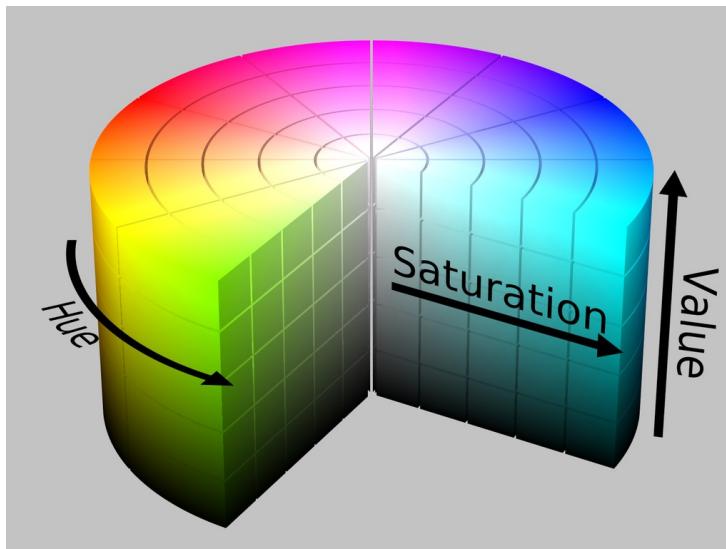
- Human eye is more sensitive to spatial variations in brightness (gray scale) than to spatial variations in color
- YUV color space can be particularly useful to highlight the importance of gray scale brightness
  - Y – brightness (luminance)
  - U and V – color (chroma)
- YUV can be obtained from RGB using linear transformations
- YUV is useful for compression: Y channel should get the most resolution (e.g., JPEG and H.264/H.265 formats)



$$\begin{pmatrix} Y \\ U \\ V \end{pmatrix} = \begin{pmatrix} .299 & .587 & .114 \\ -.14713 & -.28886 & .436 \\ .615 & -.51499 & -.10001 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

# HSV colorspace

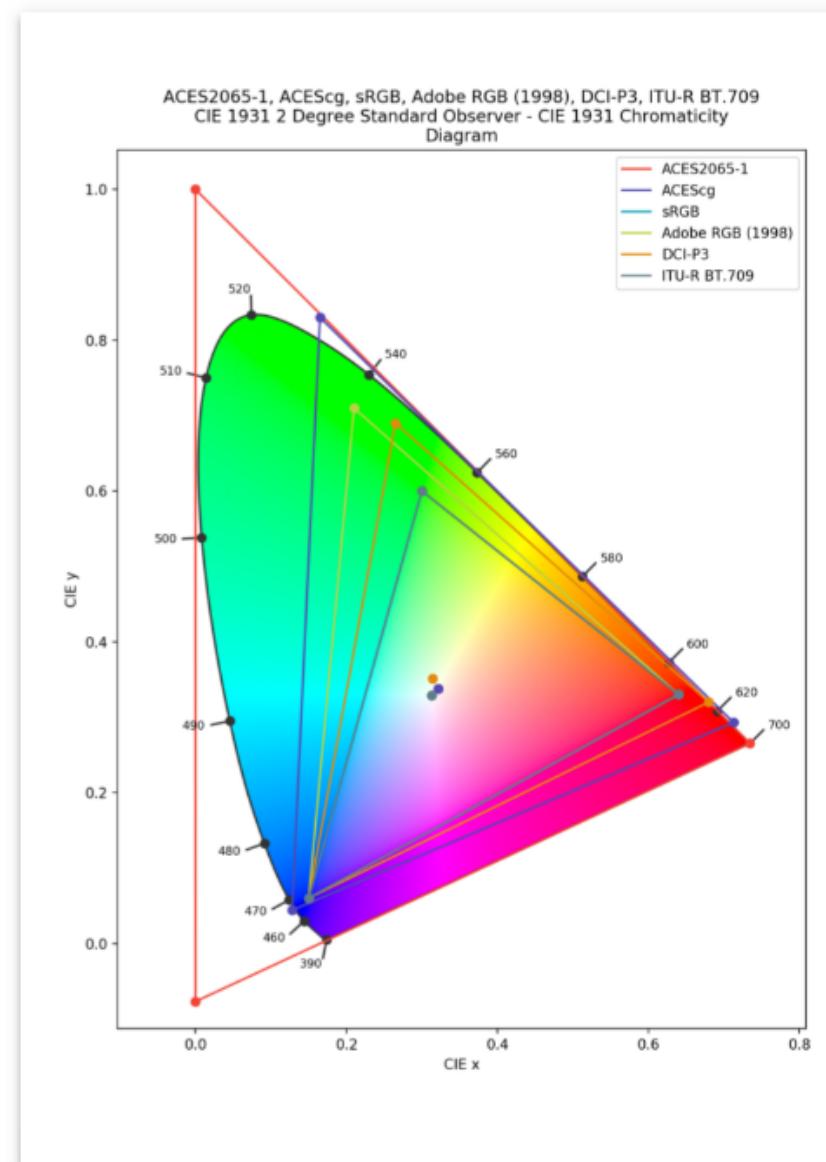
- HSV is more intuitive than RGB:
  - H – **Hue** - color (rainbow of colors), wheel, given as angle
  - S – **Saturation** - how white the color is, how much color is mixed with white
  - V – **Value** – brightness of color



HSV is often used for colorpicking in modeling environments or painting programs:  
[https://docs.krita.org/en/reference\\_manual/dockers/advanced\\_color\\_selector.html](https://docs.krita.org/en/reference_manual/dockers/advanced_color_selector.html)

# ACES colorspace

- Academy Color Encoding Specification (ACES)
  - Academy of Motion Picture Arts & Science technology committee
- Designed to cover the full gamut
- Goal: images should always look the same regardless of used camera and display device.
- Images in ACES color space can't be directly displayed to the screen. Conversion steps:
  - Reference rendering transform: from scene values to display-neutral output space
  - Output device transform: depends on display device (computer monitor, digital projector, etc.)
- Becoming de facto standard for professional production



# Color in production

- Digital drawing and image manipulation:
  - Krita: [https://docs.krita.org/en/general\\_concepts/colors/color\\_models.html](https://docs.krita.org/en/general_concepts/colors/color_models.html)
  - Gimp: <https://docs.gimp.org/2.10/en/gimp-imaging-color-management.html>
- 3D modeling:
  - Blender: [https://docs.blender.org/manual/en/latest/render/color\\_management.html](https://docs.blender.org/manual/en/latest/render/color_management.html)
  - Houdini: <https://www.sidefx.com/docs/houdini/render/linear.html>
  - Substance painter: <https://substance3d.adobe.com/documentation/spdoc/color-management-223053233.html>
- Game engines:
  - Godot: [https://docs.godotengine.org/en/3.0/tutorials/3d/high\\_dynamic\\_range.html](https://docs.godotengine.org/en/3.0/tutorials/3d/high_dynamic_range.html)
  - Unreal: <https://docs.unrealengine.com/4.26/en-US/WorkingWithMedia/OpenColorIO/>

# Light characterization

- **Radiometry** – measurement of radiation, physical transmission of light
- **Photometry** – like radiometry but related to human visual system
- **Colorimetry** – perception of color

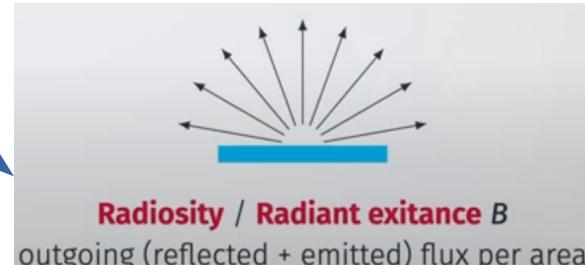
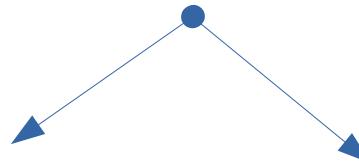
# Radiometric quantities

- Radiant energy ( $Q$ ) – **energy** carried by light (photons) [J]
- Radiant flux ( $\Phi$ ) – power carried by light - **energy over time** [ $W=Js^{-1}$ ]

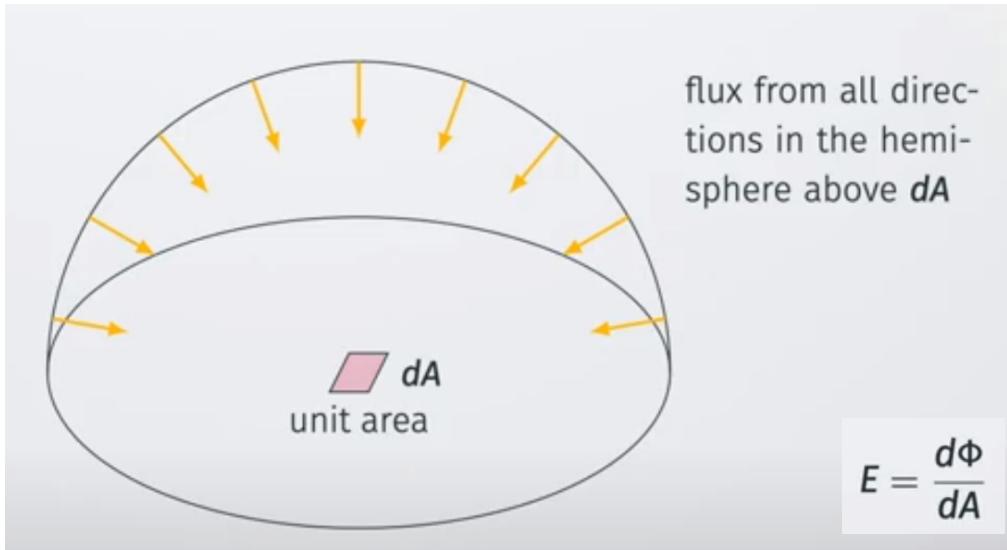
$$\Phi = \frac{dQ}{dt}$$

- Flux density ( $E$ ) – **radiant flux over area** [ $Wm^{-2}$ ]

$$E = \frac{d\Phi}{dA}$$

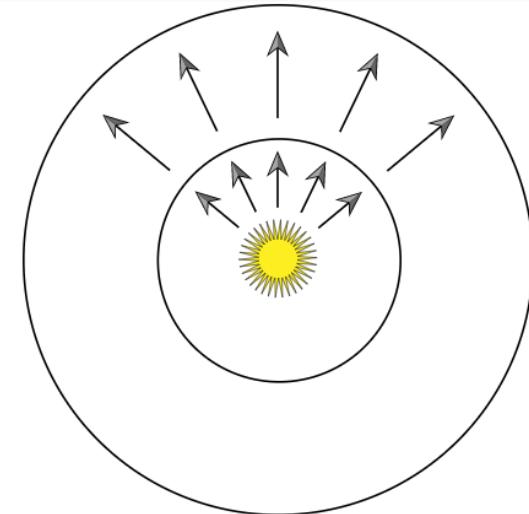


# Flux density: examples



**Irradiance** – overall radiant flux (light flow) incoming onto surface.

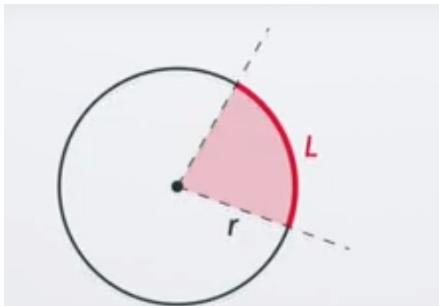
- BRDF gives information of reflected light given irradiance.



**Radiant exitance** – radiant flux (light flow) leaving a light source.

- Amount of energy received from a light falls off with the squared distance from the light.  
[https://www.pbr-book.org/3ed-2018/Color\\_and\\_Radiometry/Radiometry](https://www.pbr-book.org/3ed-2018/Color_and_Radiometry/Radiometry)

# Radiant intensity

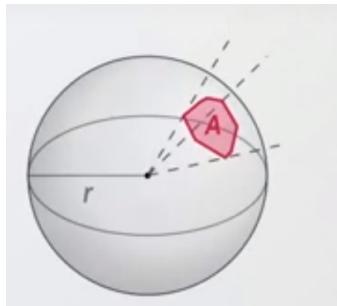


**Planar angle:** angle of segment of length  $L$  on a circle of radius  $r$

$$\alpha = \frac{L}{r}.$$

Angle of full circle:

$$\alpha = \frac{2\pi r}{r} = 2\pi \text{ [rad = radians].}$$



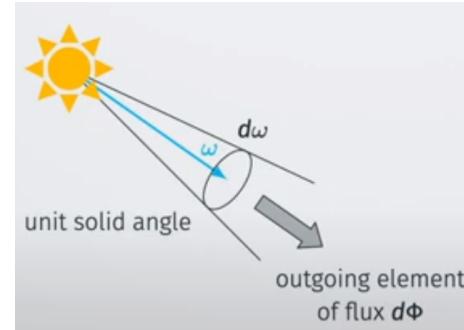
**Solid angle:** area of size  $A$  on a sphere of radius  $r$ .

$$\Omega = \frac{A}{r^2}.$$

Angle of full sphere:

$$\Omega = \frac{4\pi r^2}{r^2} = 4\pi \text{ [sr = steradians].}$$

(3D extension of the concept of plane angle)



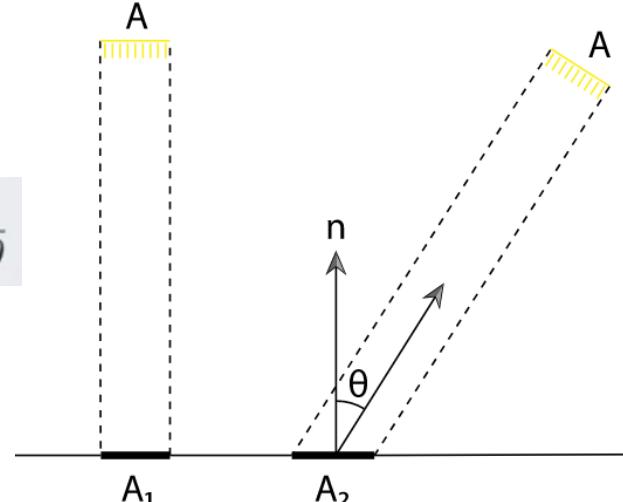
**Radiant intensity ( $I$ )** – radiant flux with respect to **solid angle** [W / sr]

- Radiant flux of light source in specific direction.
- Radiant flux reaching certain point from some point.

# Radiance

- **Radiance (L)** – radiant flux per unit solid angle, per unit projected area [W / (m<sup>2</sup> sr)]
  - Incident on surface, emerging from surface or passing through surface in certain direction
- **Electromagnetic radiation in single ray**

$$L = \frac{d^2\Phi}{d\omega dA_p} = \frac{d^2\Phi}{d\omega dA \cos \theta}$$

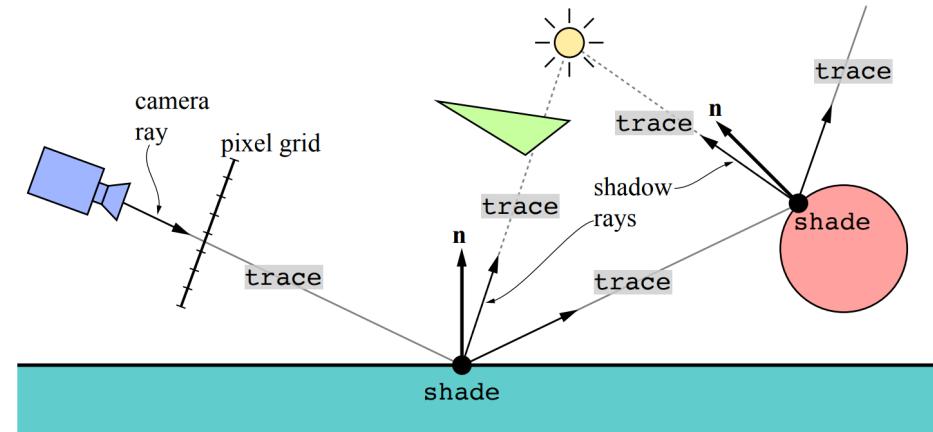
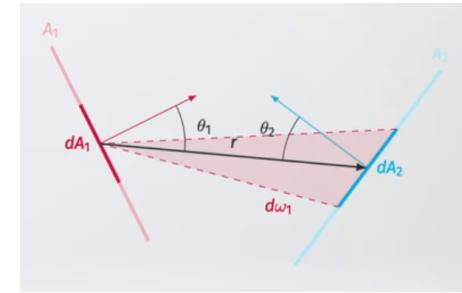


Radiance area is measured in a plane perpendicular to ray, if surface has orientation then cosine correction factor must be used.

\* Image-based rendering uses this concept, namely light field technique.

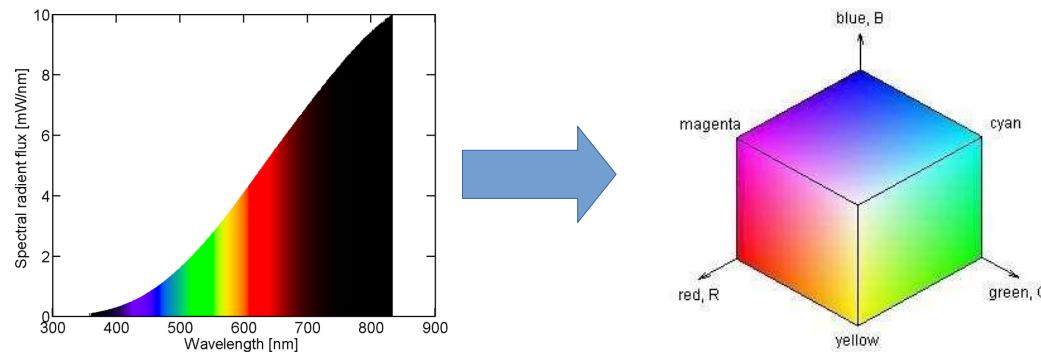
# Radiance

- Radiance is what cameras measure – prime importance for rendering
  - Rendering can be seen as evaluating radiance for each camera ray
- The purpose of shading is to calculate radiance along a camera ray
  - Radiance is simplified with color and intensity
- Conservation of radiance along rays
  - If two points on two surfaces can “see each other”, then radiance outgoing from one surface and incoming to another surface is same and vice versa → light can be traced between surfaces until it reaches the sensor
  - Radiance is not affected by distance if light is traveling in a vacuum
    - Intensity is lower for surfaces which are more distant since light covers less area



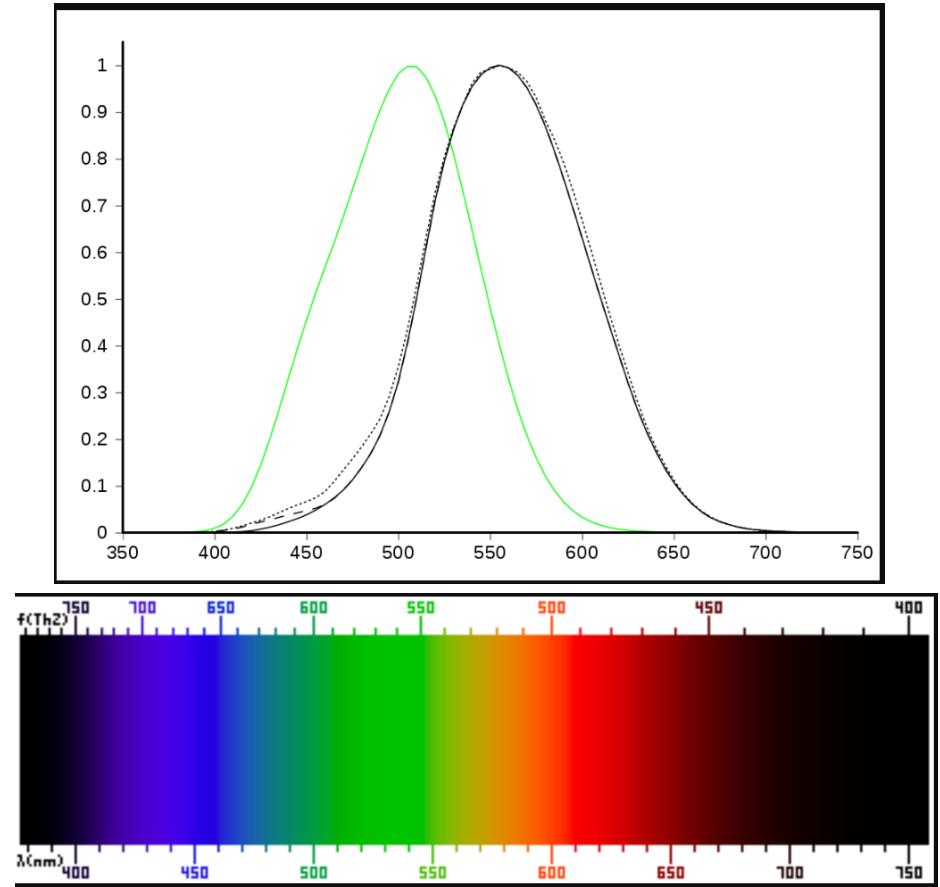
# Radiometric quantities and SPD

- Light often contains mixture of many different wavelengths – **polychromatic** (light with one wavelength is called **monochromatic**).
  - Defined using spectral power distribution (SPD)
- All radiometric quantities have spectral distributions
  - Using full SPDs for rendering is complex\*, often radiometric quantities are represented as RGB triplets.



# Photometry

- Photometry: science of measuring light (like radiometry) but taking in account human visual system
- Each radiometric quantity has equivalent photometric quantity
  - Radiant flux → **Luminous flux** (lumen - [lm])
  - Irradiance → **Illuminance** (lux - [lx])
  - Radiant intensity → **Luminous intensity** (candela - [cd])
  - Radiance → **Luminance** ([cd / m<sup>2</sup>])
- Radiometric quantities are converted to photometric by using **CIE photopic spectral luminous curve**.
  - Average sensitivity of human eye to different wavelengths for well lit conditions

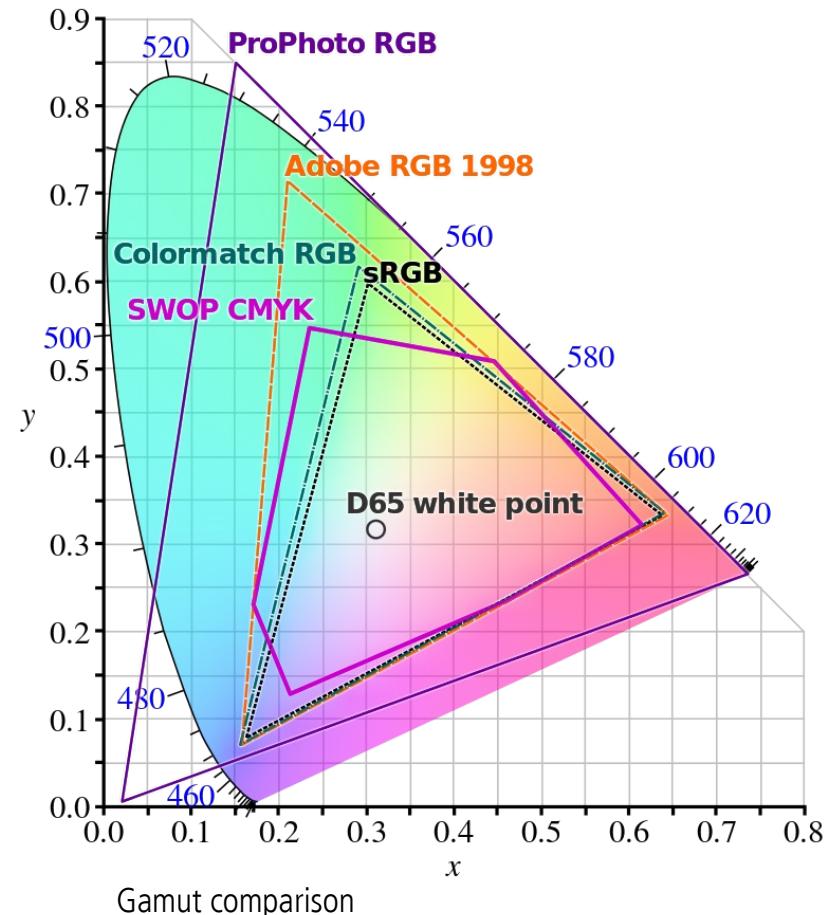
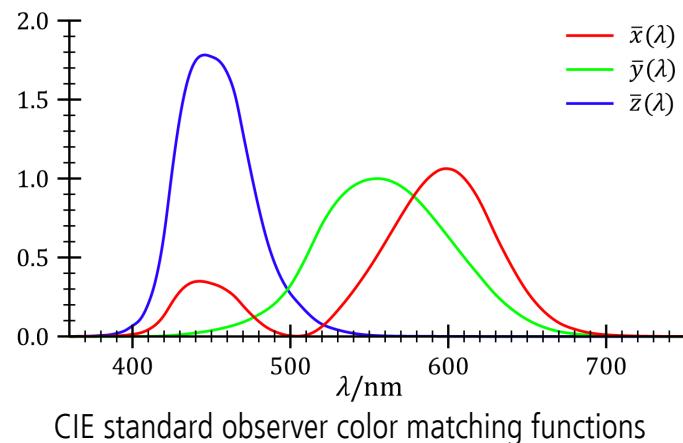


Luminosity function for photopic – green - (well lit) and scotopic (low lit) – black.

[https://en.wikipedia.org/wiki/Luminous\\_efficiency\\_function](https://en.wikipedia.org/wiki/Luminous_efficiency_function)

# Colorimetry

- Colorimetry: relationship between SPD and perception of color
- CIE standard observer color matching functions
  - SPD → XYZ colorspace
- Define colorspaces and their gamut for visualization
- RGB color spaces are most interesting for rendering



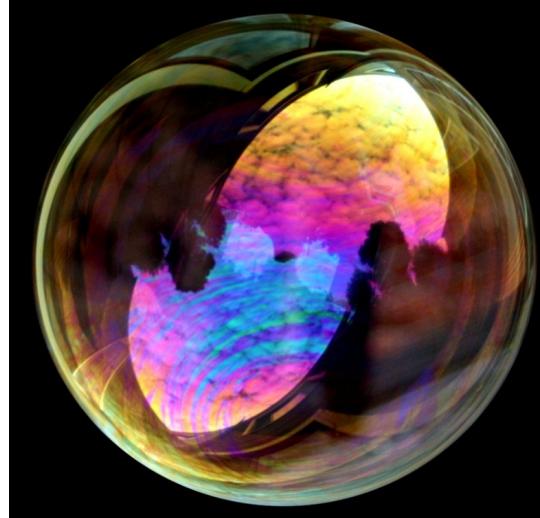
# Rendering and RGB

- For interactive and appearance-based applications (e.g., film) RGB rendering often acceptable.
  - Object surface reflection is defined using RGB in [0,1] rather than spectral reflectance curve
  - Light sources are described with RGB in [0,1] and intensity multiplier rather than SPD
- RGB is perceptual rather than physical quantity → category error in physically based rendering
  - Correct: use spectral quantities for rendering and RGB conversion for visualization
- For predictive simulations rendering spectral quantities must be used
  - SPD is used in **spectral rendering** - more correct, but expensive
  - <https://hal.inria.fr/hal-03331619/document>

# Real sources of light

# Light: microscopic level

- Described with **photons or waves**
- Some effects of light-matter interaction can be only explained (and thus modeled) using **wave optics**:
  - Interference
  - polarization
  - Diffraction
  - Iridescence
  - Pleochroism and birefringent
  - Etc.
- Microscopic level gave some foundations for shading and light transport methods\*



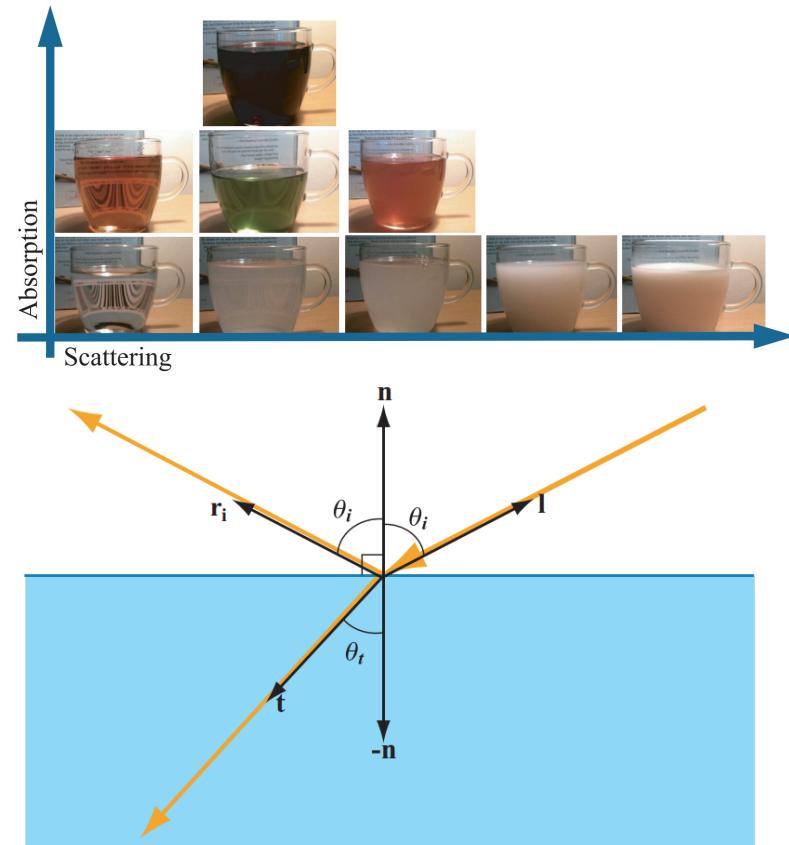
White light **interference** in a soap bubble.  
[https://en.wikipedia.org/wiki/Wave\\_interference](https://en.wikipedia.org/wiki/Wave_interference)



Color pattern of **Birefringence effect**.  
[https://en.wikipedia.org/wiki/Polarization\\_\(waves\)](https://en.wikipedia.org/wiki/Polarization_(waves))

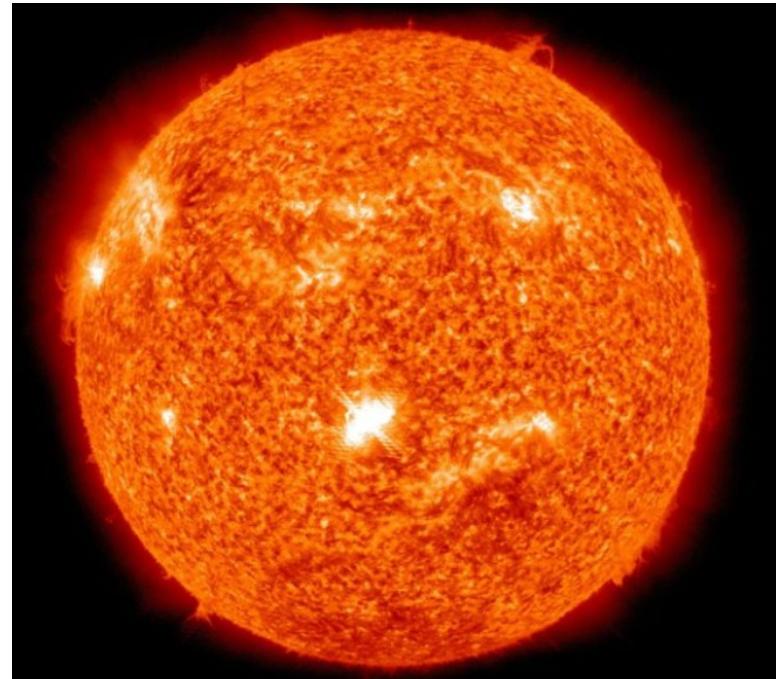
# Light: macroscopic level

- Computer graphics → geometric optics:
  - Wave effects are ignored
  - Light is described with rays
  - Microscopic light behavior is described with **index of refraction (IOR)**
- Light traveling to surface can be absorbed/scattered → participating media
  - Assume: we are using vacuum
- Light falling on surface will reflect and refract
  - Snell's law and Fresnel equations
  - Refracted light is: absorbed, transmitted or scattered inside surface



# Sources of light

- In a real world, **every** light source has a physical shape and size, e.g., Sun or very hot objects → **natural lights**
- Different way how energy is converted into electromagnetic radiation → **artificial lights**:
  - Incandescent (tungsten) lamps
  - Halogen lamps
  - Gas-discharge lamps
  - LED lights



<https://education.nationalgeographic.org/resource/sun>

# Color temperature

- Object which is heated to certain temperature emits light
  - Motion of atomic particles that hold electrical charge causes objects to emit electromagnetic radiation over a range of wavelengths (Maxwell's equations)
  - Color of emitted light depends on temperature

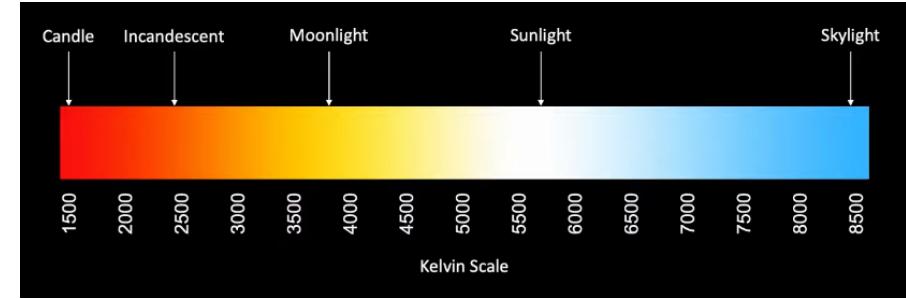
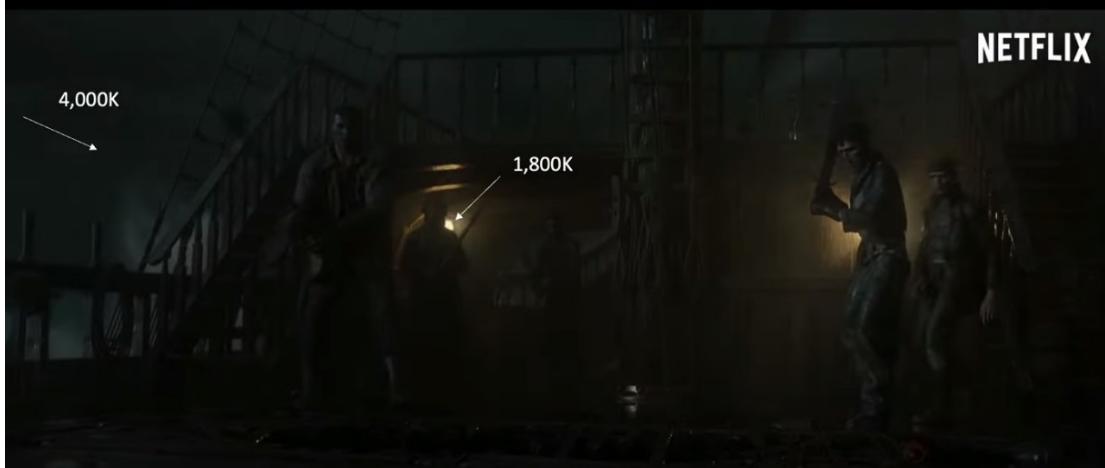


[https://docs.blender.org/manual/en/latest/render/shader\\_nodes/converter/blackbody.html](https://docs.blender.org/manual/en/latest/render/shader_nodes/converter/blackbody.html)



[https://www.inlineelectric.com/color\\_temperature](https://www.inlineelectric.com/color_temperature)

# Color temperature

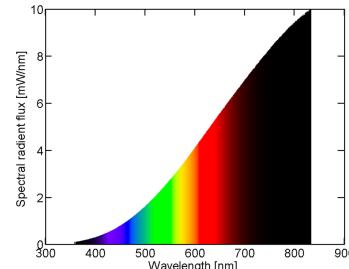


In photo-realistic rendering, light is often defined using color temperature.

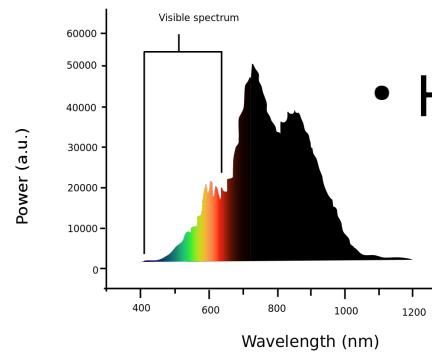
[https://www.youtube.com/watch?v=Z8AAX-ENWvQ&t=2s&ab\\_channel=Blender](https://www.youtube.com/watch?v=Z8AAX-ENWvQ&t=2s&ab_channel=Blender)



# Sources of light

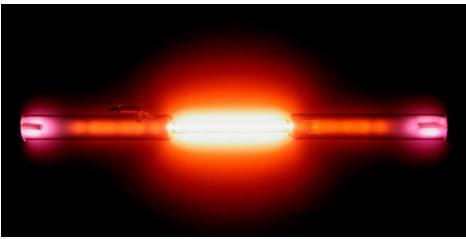


- **Incandescent (tungsten) lamps:**
  - flow of electricity through tungsten filament heats it up and causes light emission
  - Distribution of wavelengths depending on filament temperature
  - A frosted glass enclosure is often present to absorb some of the wavelengths



- **Halogen lamps:** tungsten filament is enclosed in halogen gas.

# Sources of light



Helium: [https://en.wikipedia.org/wiki/Gas-discharge\\_lamp](https://en.wikipedia.org/wiki/Gas-discharge_lamp)

- **Gas-discharge lamps:** passing electrical current through hydrogen, neon, argon, or vaporized metal gas, causes light to be emitted at specific wavelengths that depend on the particular atom in the gas.
- Fluorescent coating on the bulb's interior is often used to transform the emitted frequencies to a wider range.



LED lights: [https://en.wikipedia.org/wiki/Light-emitting\\_diode](https://en.wikipedia.org/wiki/Light-emitting_diode)

- **LED lights** are based on electroluminescence: they use materials that emit photons due to electrical current passing through them.

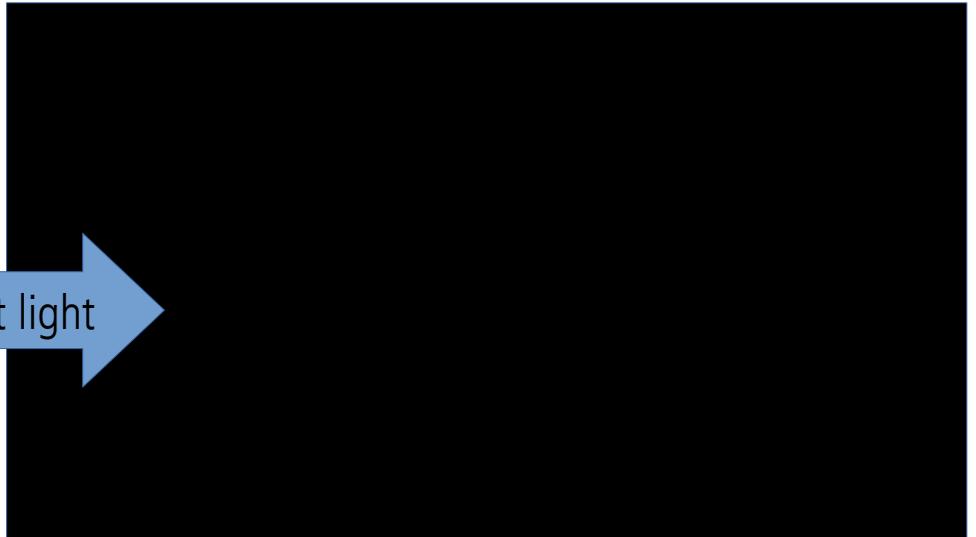
# Light model

# Rendering and light

- Light is emitted from a light source,
- Light propagates through vacuum (ignore participating media)
- Light interacts with objects (e.g., reflection, absorption) which depends on object material
- Very small portion of light enters camera, falls on film producing image

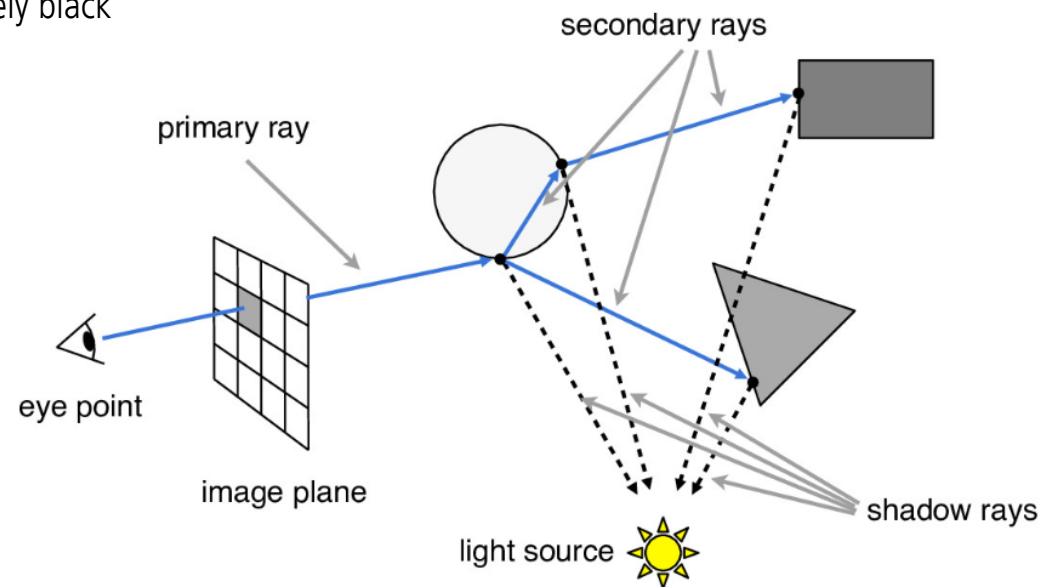


Without light



# Rendering and light

- Rendering: calculating colors of pixels on virtual image plane
  - For each pixel, camera ray is generated and tested for intersection with objects in 3D scene → **visibility solving**
  - Color of intersection point (shading point) is calculated in **shading step**
    - Shading takes in account 3D object shape (i.e., normal), object material, viewing direction and incoming light
    - Without light resulting image would be completely black



# Rendering and light

- Rendering equation: color (radiance) of shading point  $p$  in (view) direction  $\omega$

$$L_o(p, \omega_o) = L_e(p, \omega_o) + \int_{\Omega} f(p, \omega_o, \omega_i) L_i(p, \omega_i) (\omega_i \cdot n) d\omega_i$$

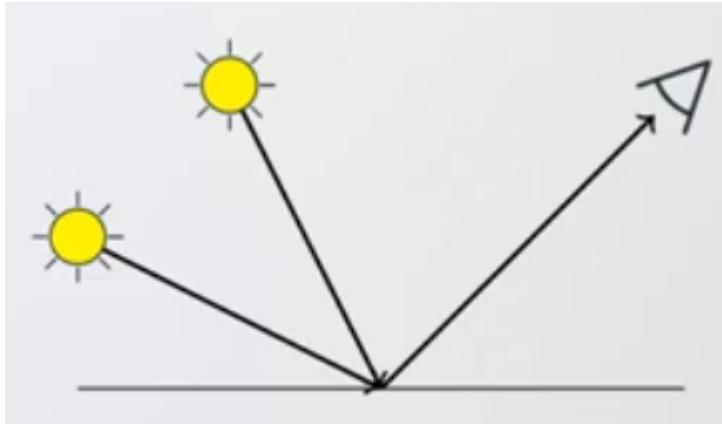
Emission Reflection

BRDF Incoming light attenuation

- Foundation of physically-based rendering
- Describes global illumination phenomena – light coming from all directions above object surface
- Recursive equation - not possible to solve analytically: approximation methods are used

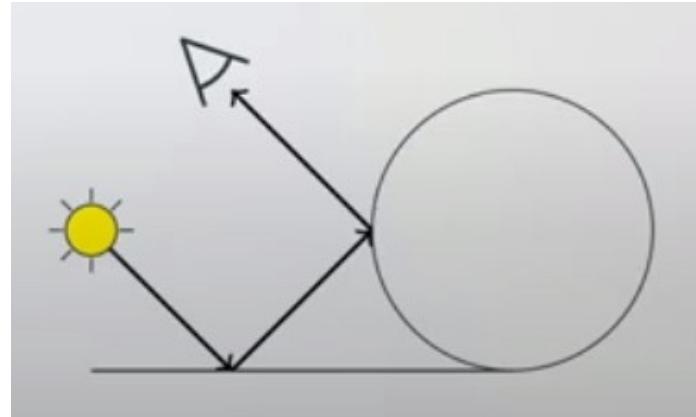
# Global and direct illumination

- Rendering equation is very general equation: different rendering approaches solve different parts of this equation



**Direct illumination:** takes in account only light from light sources (e.g., non-physical light sources; point lights)

- Often used in rasterization-based rendering
- Problem: additional work needed for shadows, reflections, etc.



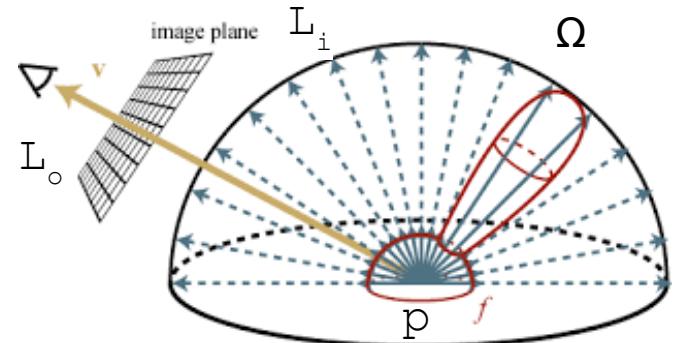
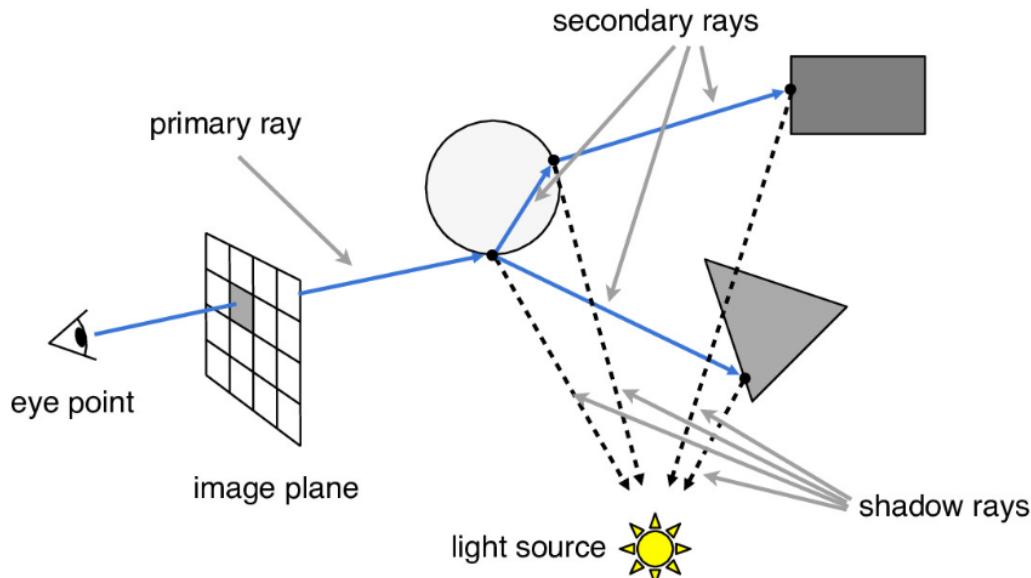
**Global illumination:** takes in account light from light sources and indirect light: reflections from other surfaces

- Often solved using ray-tracing-based rendering methods
- Multiple reflections, transmissions and shadows can be elegantly incorporated
- More advanced methods, e.g., path tracing, further build on raytracing to solve this problem more correctly

# Global illumination

- Global illumination: light can fall on surface from any direction:
  - Light sources
  - Surfaces reflecting light

$$L_o(p, \omega_o) = L_e(p, \omega_o) + \int_{\Omega} f(p, \omega_o, \omega_i) L_i(p, \omega_i)(\omega_i \cdot n) d\omega_i$$



# Local (direct) illumination

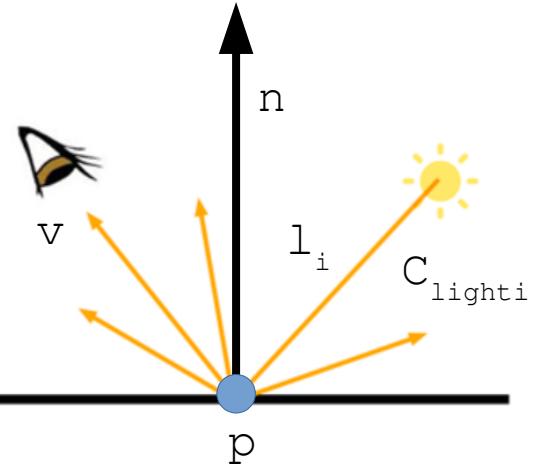
- Local illumination: shading is considering only light coming from light sources (discrete incoming light angles)
  - $l_i$  is given as color and intensity.
  - Direction of light source is given or calculated from its position

$$c_{shaded}(p, v) = \sum_{i=1}^n f(l_i, n, v) c_{light_i} (n \cdot l_i)^+$$

Light source color

BRDF

$f(l_i, n, v)$

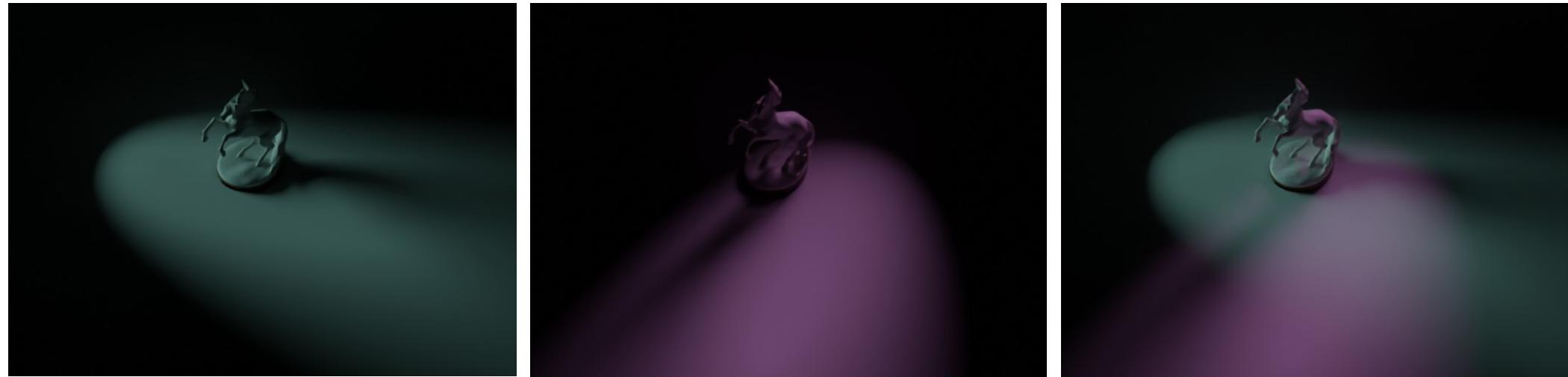


# Multiple lights

- Contribution of multiple light adds up linearly
  - Basis of photo-realistic rendering where scene contains multiple lights
  - Required for compositing of rendered images: adding multiple rendered images where each has contribution from one light

$$L_o(p, v) = \sum_{i=1}^n f(l_i, n, v) c_{light_i}(n \cdot l_i)$$

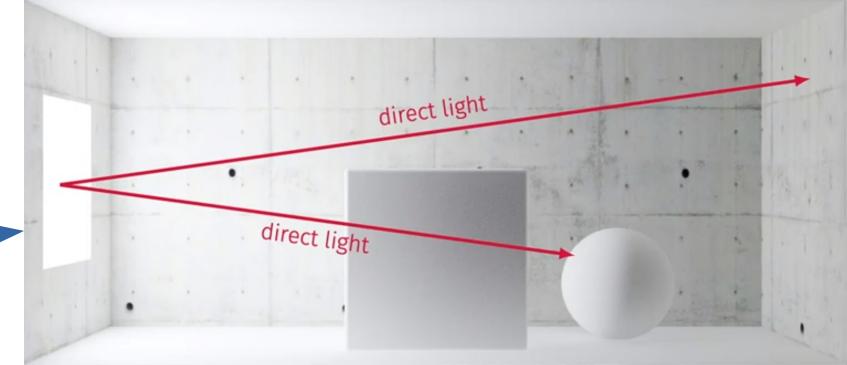
$$L_o(p, v) = \int_{l \in \omega} f(l, n, v) L_i(p, l) (n \cdot l) dl$$



# Local and global illumination

## Direct illumination

- No shadows



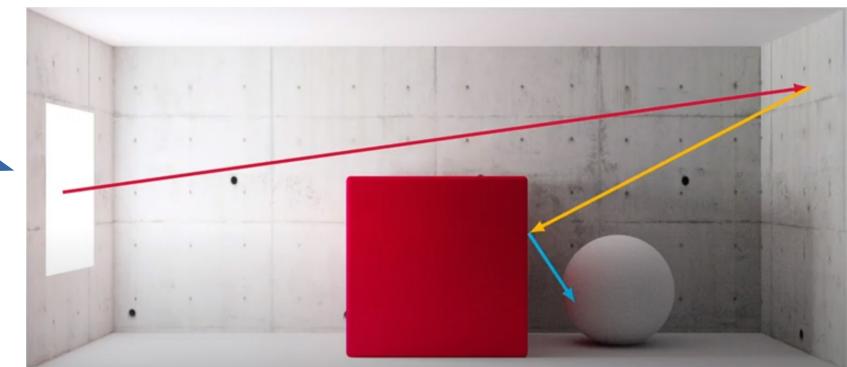
## Direct illumination with shadows

- Areas in shadow are completely black



## Global illumination

- Shadows
- Multiple reflections
- Color bleeding



# Global illumination



Indirect light causes caustics.

Blender and appleseed:

<https://www.blenderdiplom.com/en/tutorials/all-tutorials/649-rendering-caustics-in-blender-with-appleseed.html>



Light refraction can not be solved with local illumination methods:

[https://www.youtube.com/watch?v=rJghpp8RVmw&ab\\_channel=yuichiroyama](https://www.youtube.com/watch?v=rJghpp8RVmw&ab_channel=yuichiroyama)



Participating media and volumetric objects can not be accurately represented with direct illumination:

<https://forums.jangafx.com/t/blender-cycles-test-rendering-of-the-free-vdb-cloud-pack/1273/3>

# Light and rendering (shading)

- Light sources must define
  - Position or direction of light
  - Color and intensity of light
- Additionally, light sources can have:
  - Size
  - Shape



# Light models



Representation and modeling of lights with shape and size → **geometric area (physical) lights\***.

- Expensive to compute

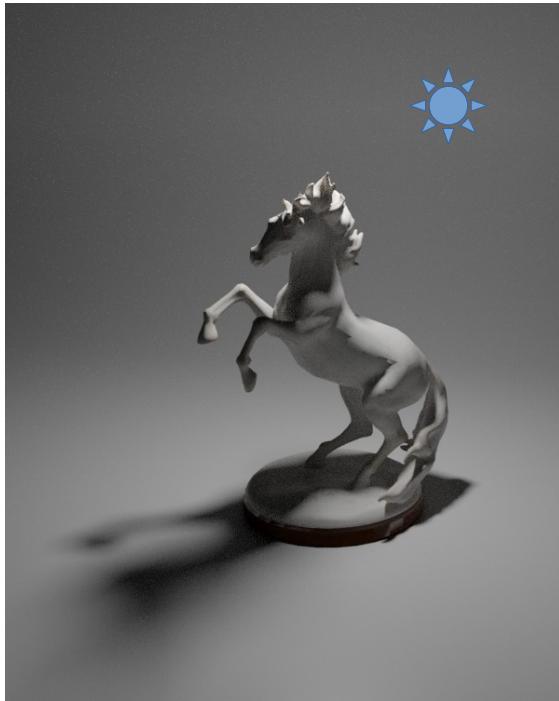
Simplification of lights with no physical size, are called **delta or non-physical lights**.

\* When we discussed surface material we mentioned scattering and absorption. Also, materials can be emissive. Therefore, physical lights have 3D shape and material which is emissive. Emissive material is modeled as black body meaning that it absorbs all light falling on it.

# Light models parameters

Non-physical lights:

- Color: RGB in  $[0, 1]$
- Intensity: float  $[0, 1]$   
(multiplier)
- Position → Direction



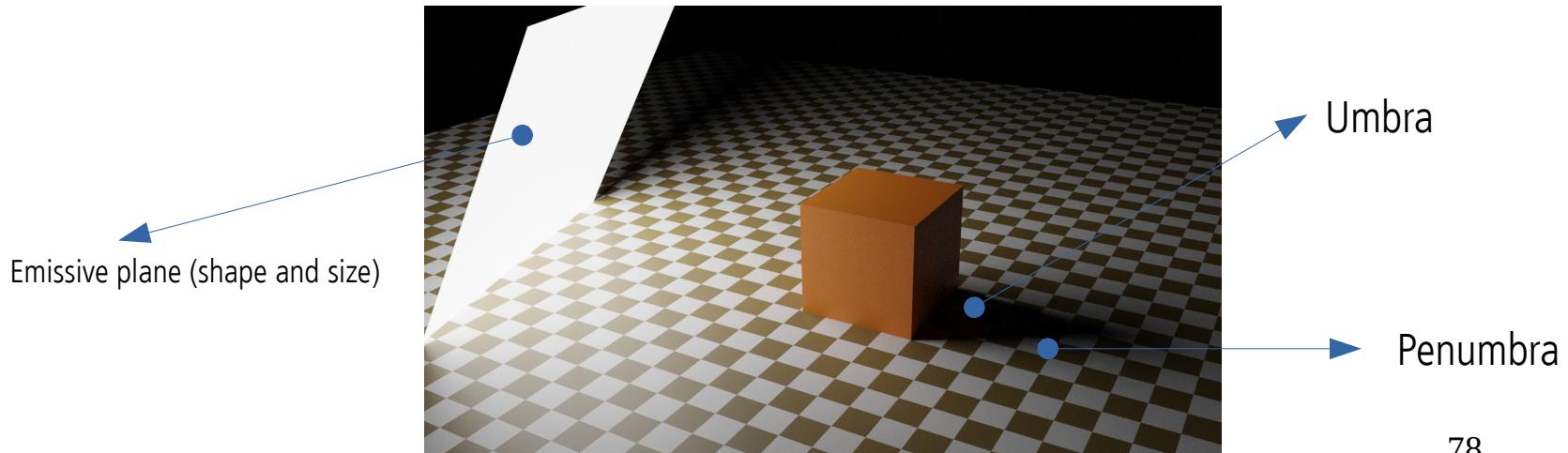
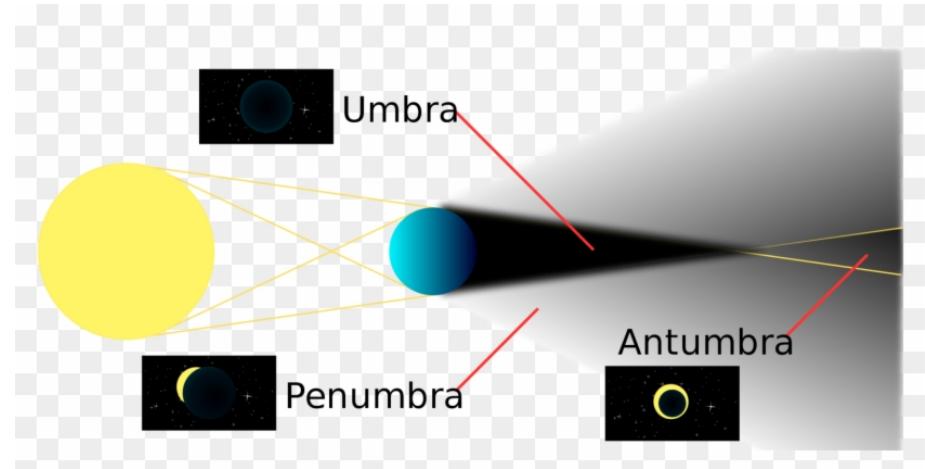
Physical Lights

- Color (RGB)
- Intensity (float)
- Position
- Shape
- Size

# Physical lights

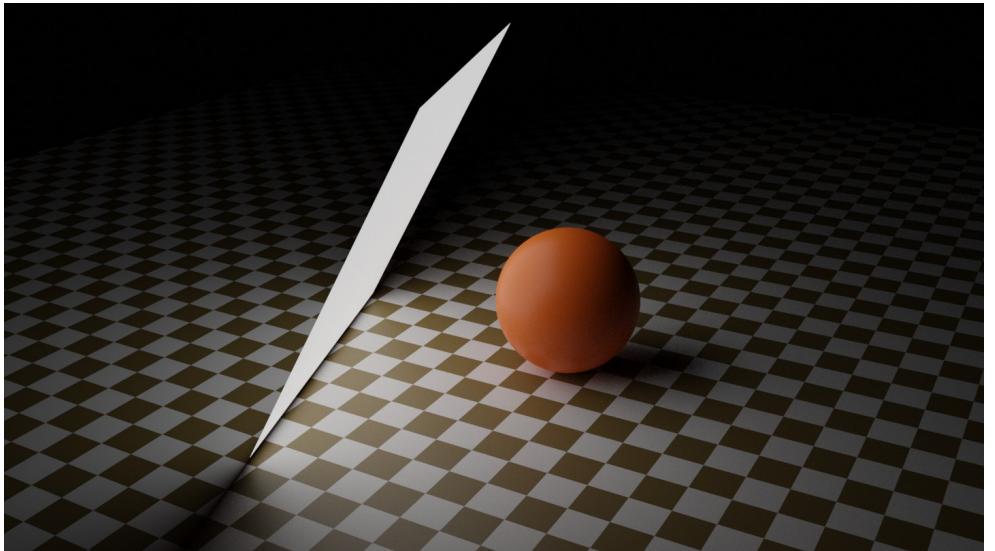
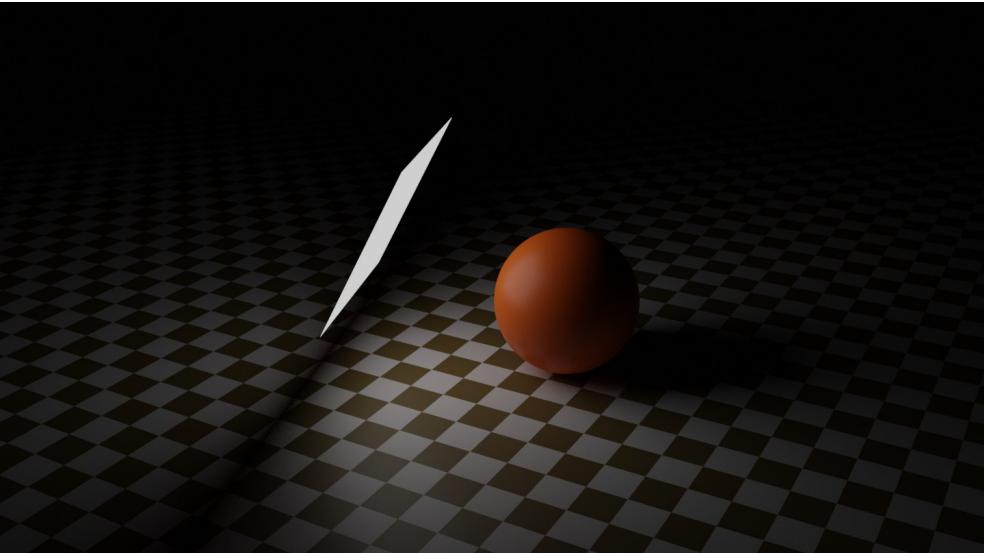
# Physical lights

- Shapes that emit light from their surface (emissive material)
  - Color and intensity
  - Position
  - shape and size



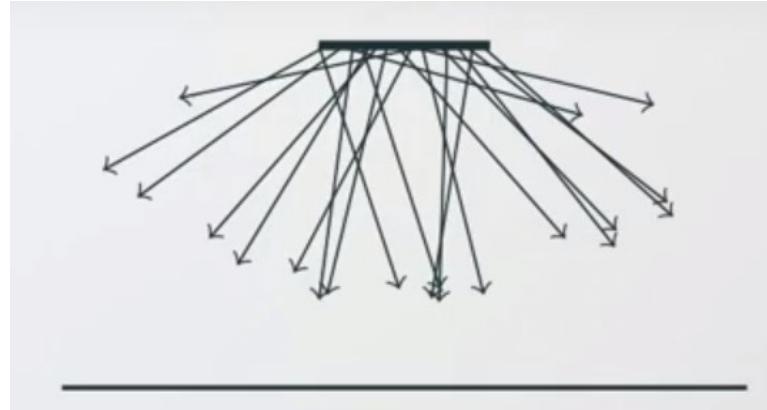
# Physical lights

- Shape and size of light matters
  - Amount of brightness
  - Softness and size of shadows



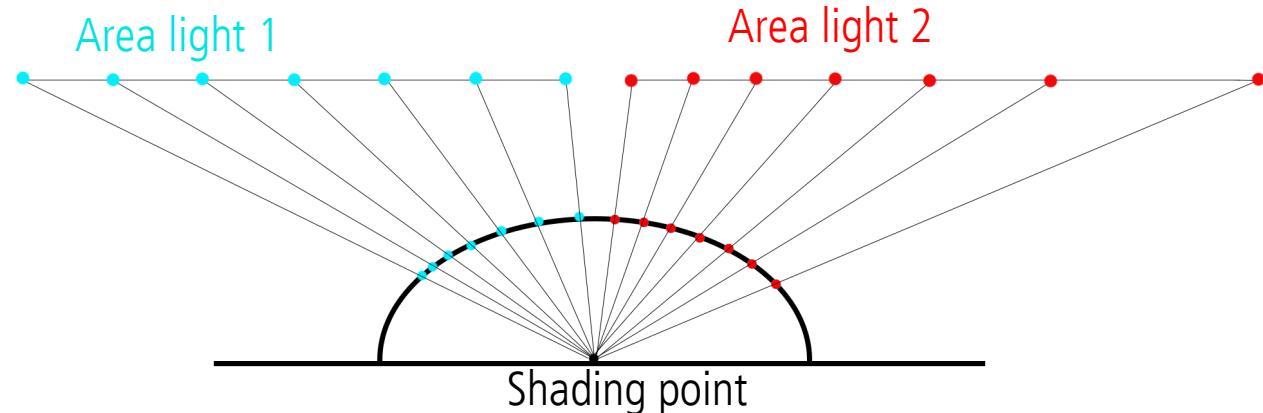
# Physical lights

- Intensity and color of emission is given as distribution over emissive surface
- Diffuse physical light: uniform spatial and directional intensity and color distribution
  - Light is emitted in direction defined with surface normal
- At each point in the scene, light from physical light source can be incident from many directions of its surface



# Physical lights

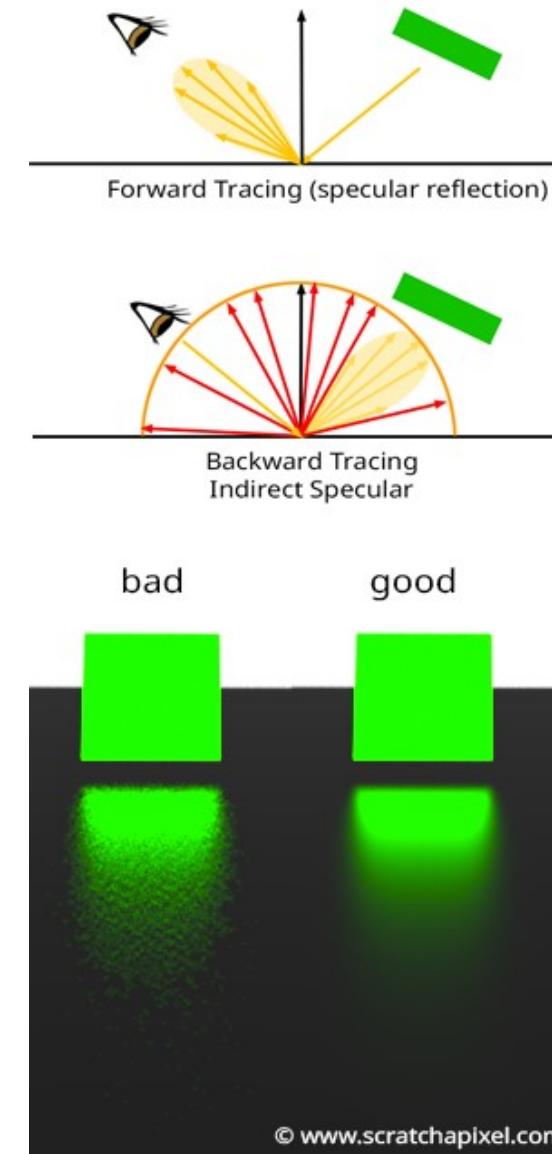
- Computing incoming light from physical light source onto shading point requires integrating (summing) contributions from points over emissive surface
  - Can not be computed in closed form → **sampling of light source is needed** which can be expensive
- In **ray-tracing based approaches** additional rays are sent from shading point to samples in physical light to approximate incoming light
- **Rasterization-based rendering** simplifies and approximates effects of physical lights



# Physical lights: ray-tracing-based rendering

- Ray-tracing-based rendering simulates physical flow of light, emission from physical lights and their effects on 3D scene (e.g., shadows) can be elegantly solved
- Physical lights have area which has to be sampled
  - Additional rays are generated for points sampled on area lights
  - Approximation-based methods such as Monte Carlo integration must be used
  - Global illumination method: path-tracing

$$L_o(p, \omega_o) = L_e(p, \omega_o) + \int_{\Omega} f(p, \omega_o, \omega_i) L_i(p, \omega_i) (\omega_i \cdot n) d\omega_i$$



# Light Sampling and Shadow/Direct Specular Noise

AA\_Samples = 1. Light samples = 2 (4 samples).

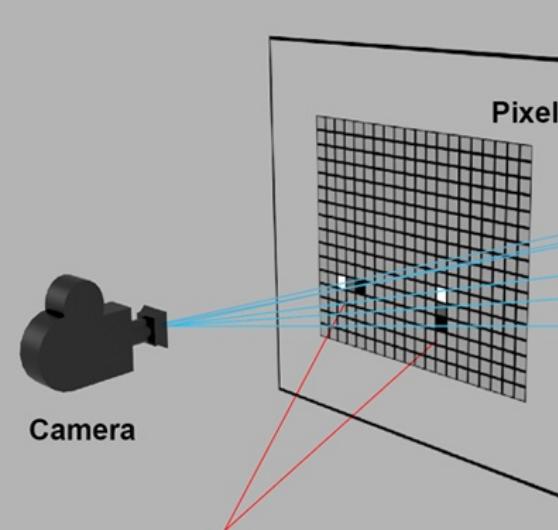
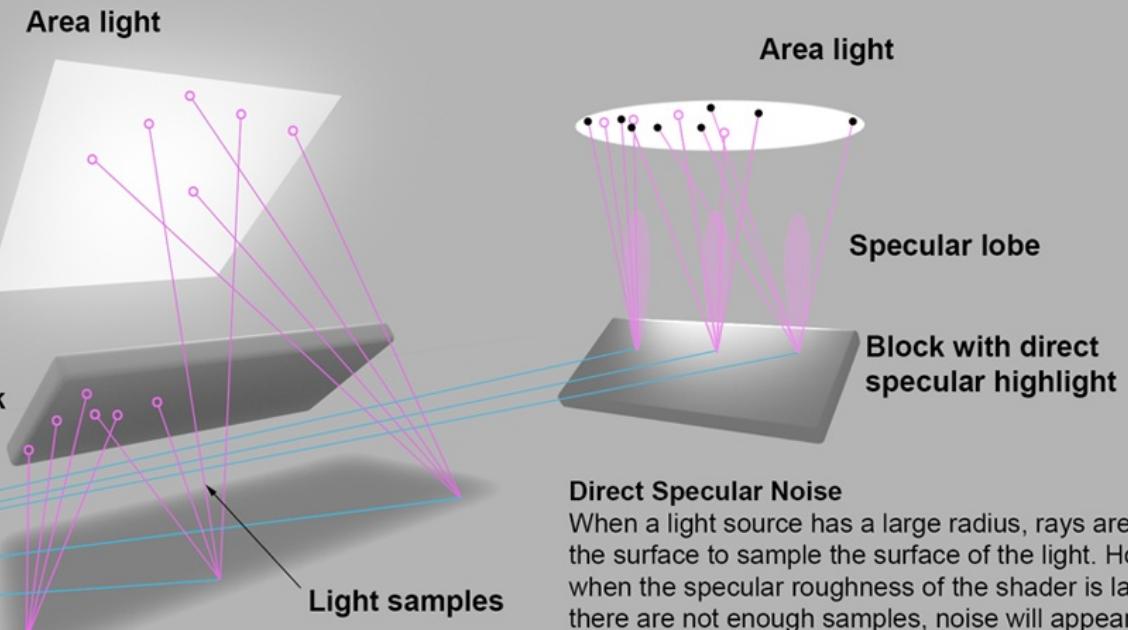


Image noise due to low sampling  
(pixel value = average subsamples).



## Shadow Noise

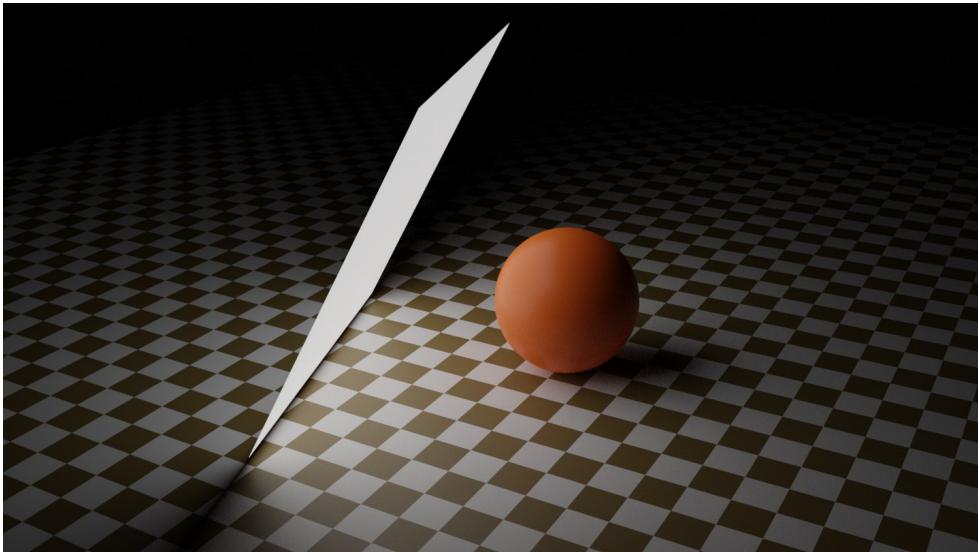
Camera rays intersecting with geometry fire shadow rays that check occlusion. If the radius of a light  $> 0$ , those rays are fired within the angle of the 'light sampling angle' (shaded area with dotted lines); shadows can be noisy as Arnold attempts to resolve the area light with insufficient rays. In this example, the 4 samples fired toward the light randomly hit either the block (resulting in a low value) or the light (resulting in a high value).

## Direct Specular Noise

When a light source has a large radius, rays are fired from the surface to sample the surface of the light. However, when the specular roughness of the shader is large and there are not enough samples, noise will appear on the surface. Noise can also appear when the light source is too close to the surface creating high intensity samples.

# Physical lights: rasterization-based rendering

- Rasterization-based rendering is not simulating natural flow of light. Therefore, physical lights and their effects require simplifications and approximations.



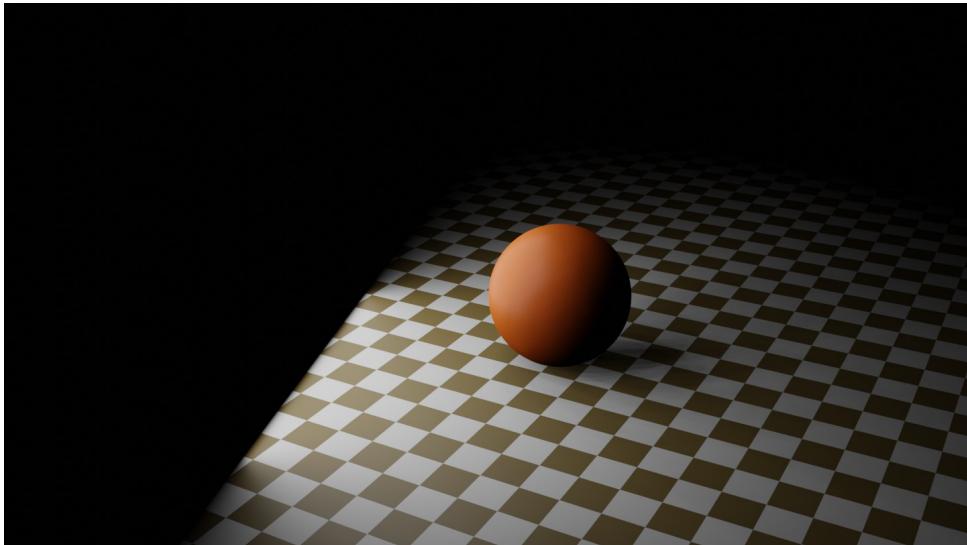
Physical lights in ray-tracing (path-tracing, Blender, Cycles)



Physical lights in rasterization (Blender, EEVEE)

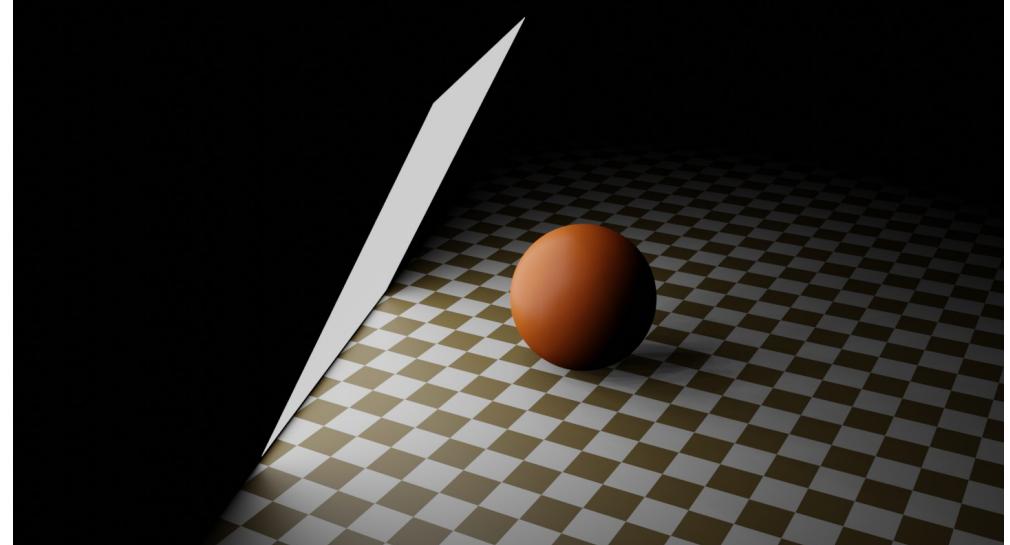
# Physical lights: rasterization-based rendering

- Special cases of physical lights - rectangular planes, can be approximated in rasterization-based renderer\*



Rectangular plane emission (Blender, EEVEE):

[https://docs.blender.org/manual/en/latest/render/lights/light\\_object.html#area-light](https://docs.blender.org/manual/en/latest/render/lights/light_object.html#area-light)



Emissive plane and rectangular plane emission approximation (Blender, EEVEE)

# Physical lights: rasterization-based rendering

- Alternative is to pre-compute effects of physical lights using path-tracing and store it in texture which is then used during shading\*
  - <https://docs.blender.org/manual/en/latest/render/cycles/baking.html>
  - <https://catlikecoding.com/unity/tutorials/rendering/part-16/>
- Effects of physical lights can be approximated using non-physical lights

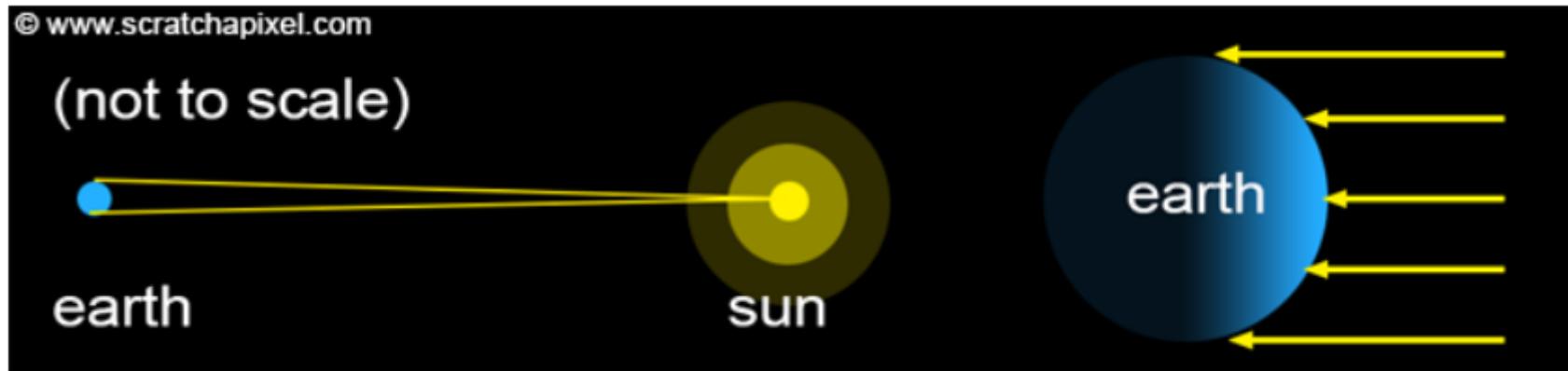
# Non-physical lights

# Non-Physical lights

- **Non-physical lights simulate effects of physical lights:** light fall-off with distance, which objects are illuminated, which objects cast shadows, etc.
- Types:
  - **Directional** (distant, parallel) lights
  - **Point** (spherical, omnidirectional) lights
    - Spot lights
    - Other variations

# Directional (parallel) lights

- Directional lights **simulate light sources which are considered so far from the objects in a 3D scene** so their emission can be represented by parallel rays.
  - We only care for the **direction** of those parallel rays

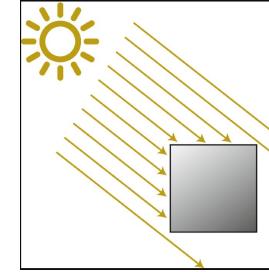


Example of directional light: is Sun light on Earth.

- Sun has spherical shape, but it is so far and Earth is so small compared to it that light rays reaching Earth can be considered parallel (small cone of directions – solid angle).
- For scenes that cover small area of Earth's surface, Sun light can be assumed parallel.

# Directional (parallel) lights

- Directional light is simplest model of light source.
  - **Direction ( $\mathbf{l}$ )** – unit vector (not affected by position!)
  - **Color ( $\mathbf{c}$ )** – RGB vector in  $[0, 1]$
  - **Intensity ( $i$ )** – float value in  $[0, \infty]$
  - Direction ( $\mathbf{l}$ ), color ( $\mathbf{c}$ ) and intensity ( $i$ ) are constant over 3D scene
- Intensity may be attenuated by shadowing and surface orientation

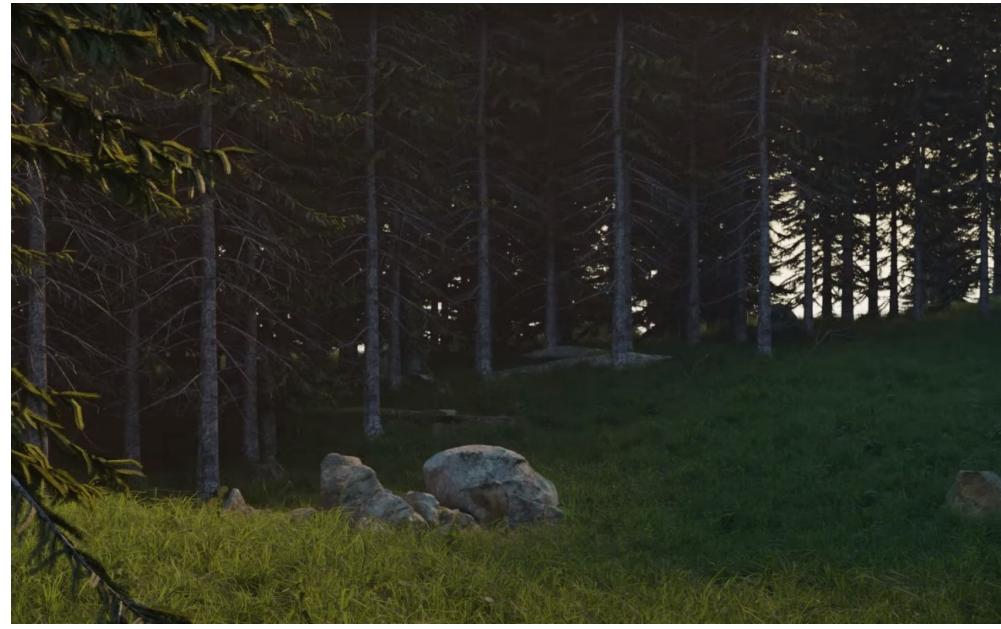


[https://link.springer.com/chapter/10.1007/978-1-4842-4457-9\\_12](https://link.springer.com/chapter/10.1007/978-1-4842-4457-9_12)



# Practical tip: modeling with lights

- Directional light sources are often used to simulate sun light and to lit environments
- Orientation of light greatly determines scene appearance



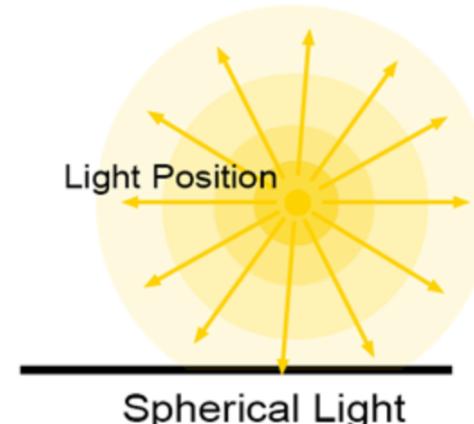
# Point lights

## Point light

- Infinitesimally small in size → no shape
  - Light is emitted uniformly in all directions (omnidirectional, isotropic)
- 
- Parameters
    - **Position** – vector determining position in world space (lights can be also transformed using 4x4 matrices)
    - **Color** – RGB vector in [0,1]
    - **Intensity** – float value in [0,inf]
    - Not affected by scale or rotation.

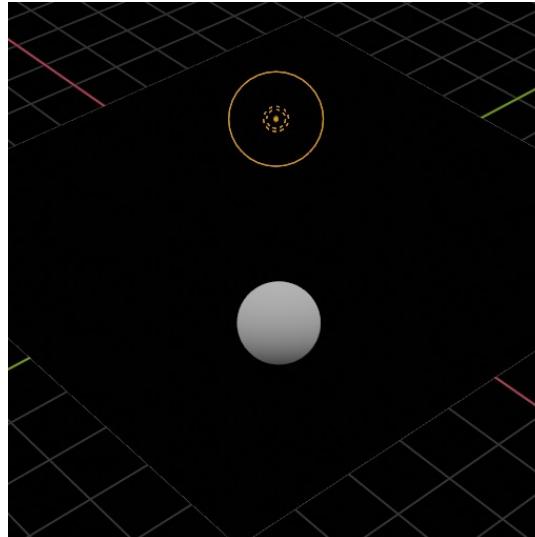
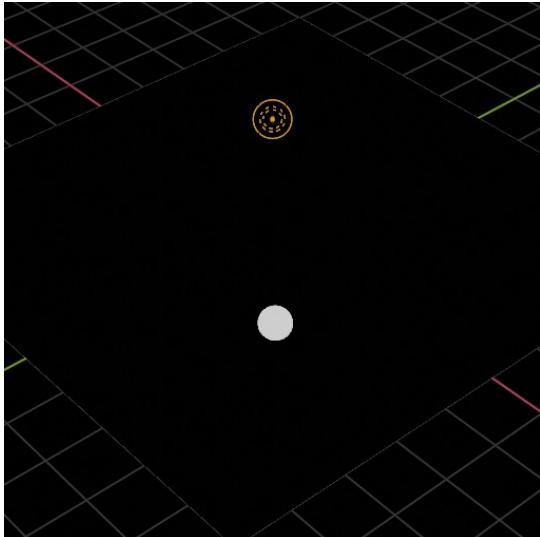


Most common light sources in nature and around us are spherical (e.g., light bulbs) or can be approximated as spherical light sources.



# Non-Physical lights: note

- For correct physically-based rendering, non-physical light should be avoided
- Example: size of reflection of object by glossy or mirror-like surface depends on its size and distance to reflective surface. If light source has no shape nor size how reflection should look like? Hack: size parameter which is only used during shading.



Point lights have parameter "size" which determines light reflection size:  
[https://docs.blender.org/manual/en/latest/render/lights/light\\_object.html#point-light](https://docs.blender.org/manual/en/latest/render/lights/light_object.html#point-light)

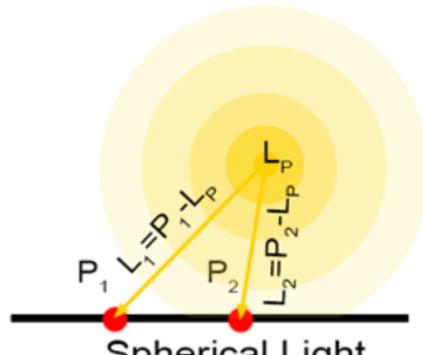
# Point lights

- Point light position determines **direction and distance of incoming light ray** for each surface point of objects in the scene

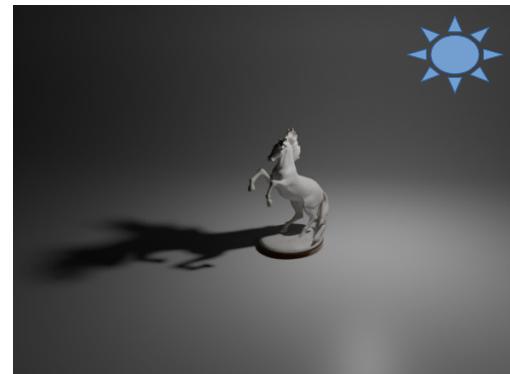
$$\text{PointToLight} = P - L_p$$

$$\text{LightDistance} = \text{length}(\text{PointToLight})$$

$$\text{LightDirection} = \text{normalize}(\text{PointToLight})$$



© www.scratchapixel.com

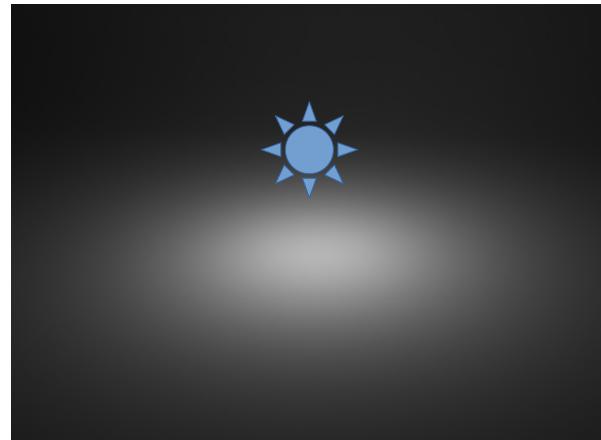
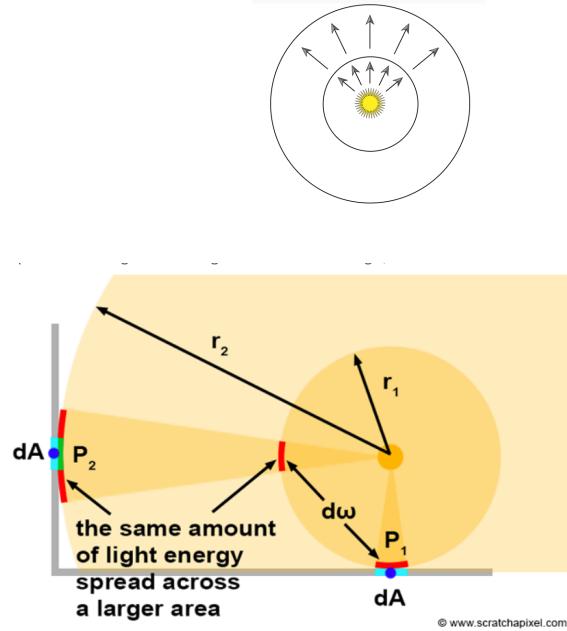


$$c_{shaded}(p, v) = \sum_{i=1}^n f(l_i, n, v) c_{light_i} (n \cdot l_i)^+$$

Point light **direction**  $l_i$  is used in BRDF and attenuation factor.

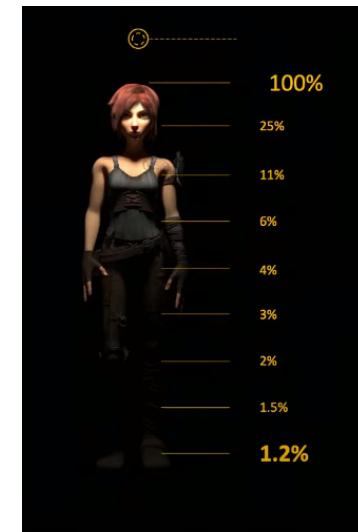
# Point lights: light falloff

- Point light sources emit light radially
  - Energy of light source is redistributed over sphere
  - As the sphere keeps expanding in the space, energy becomes spread across much larger area  
→ more distant objects in the scene receive less light → **light falloff**.
  - Distance of point light to surface point determines how much is that point illuminated  
→ **intensity of point light varies as a function of distance.**

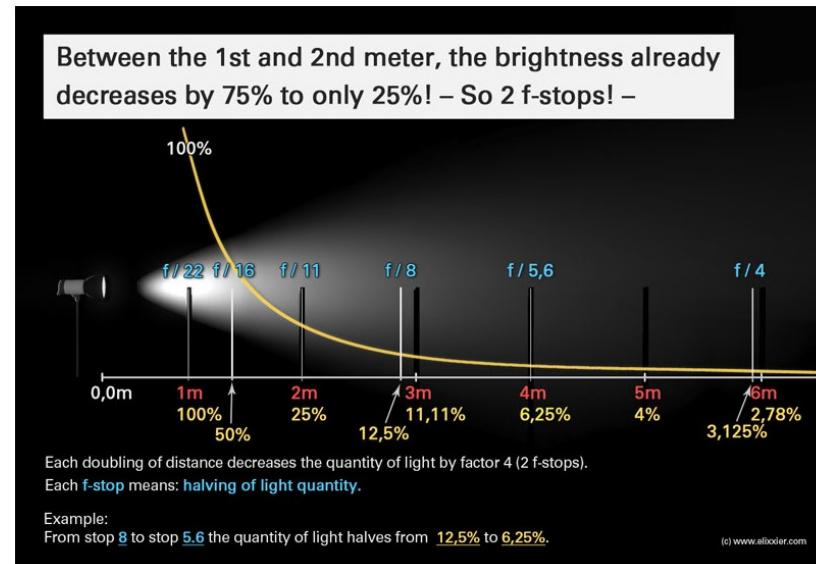


# Light falloff

- Distance from point light to surface point determines **light falloff**
- Larger distance → same energy on larger sphere → attenuated light - inverse square law falloff



$$L_i = \frac{\text{light intensity} * \text{light color}}{4\pi r^2}$$



<https://petapixel.com/inverse-square-law-light/>

# Point light - light falloff



True Grit (2010)



Toy Story 4 (2021)

[https://www.youtube.com/watch?v=Z8AAX-ENWvQ&t=2s&ab\\_channel=Blender](https://www.youtube.com/watch?v=Z8AAX-ENWvQ&t=2s&ab_channel=Blender)

# Point light - light falloff

- Problems:
  - **Small distances**: as  $r \rightarrow 0$ , light intensity  $\rightarrow$  infinity. Fix:

$$L_i = \frac{\text{light intensity} * \text{light color}}{4\pi(r + \text{epsilon})^2}$$

- **Large distance**: for efficient rendering, it is desired to light reach 0 at some point. Otherwise, light calculation should be done no matter how far the camera is from the light
- Notes:
  - Light falloff function can be any other function of distance
  - For stylized appearance different functions may be more interesting

# Practical tip: modeling with lights

- Position of light greatly determines scene appearance



# Point lights: directional light falloff

- Next to distance light falloff, real-world illumination intensity also **varies by direction**, generally:

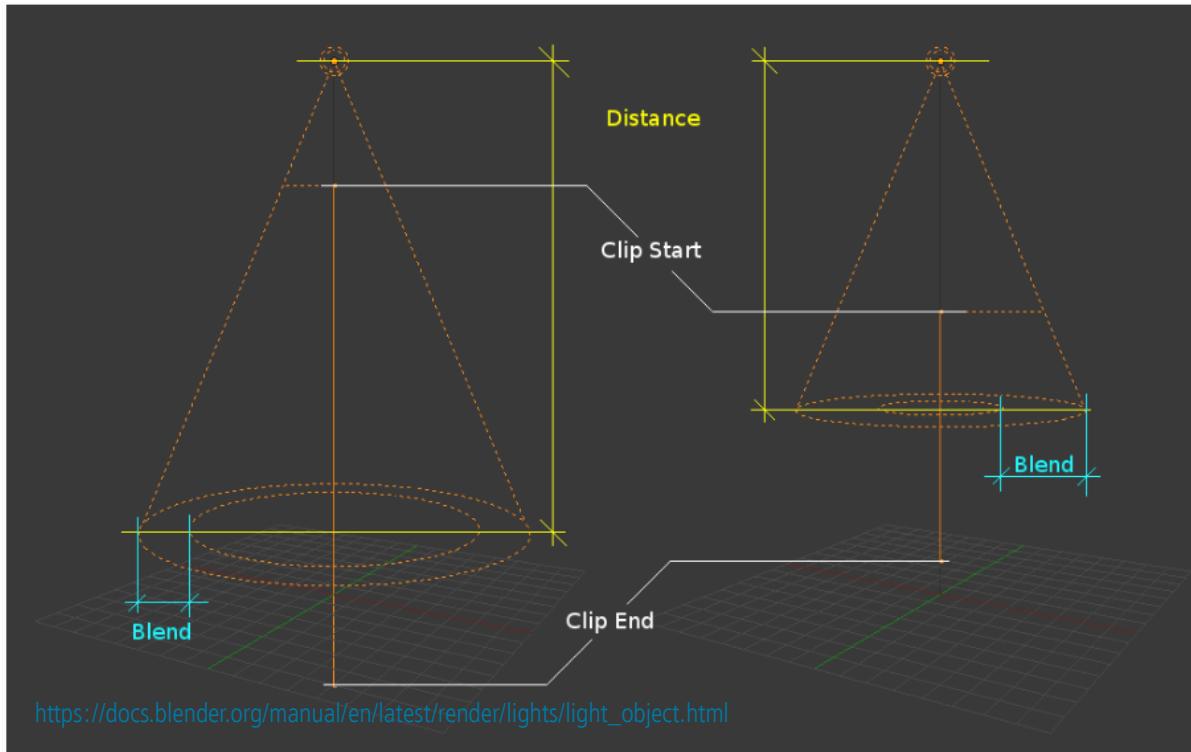
$$L_i = (\text{light intensity} * \text{light color}) f_{distance}(r) f_{direction}(l)$$

- Different choices of distance and directional light falloff functions produce different light sources:
  - **Spotlights** - emit light in a cone of directions from their position

$$L_i = (\text{light intensity} * \text{light color}) f_{distance}(r) \boxed{f_{direction}(l)}$$

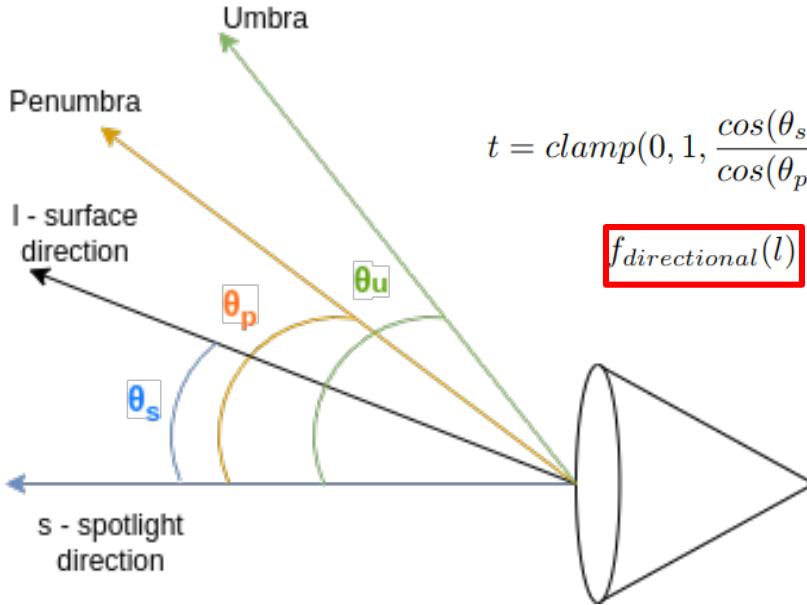
# Spotlights

- Projects light in circular cone
- Directional falloff function is rotational symmetric around a spotlight direction



# Spotlights

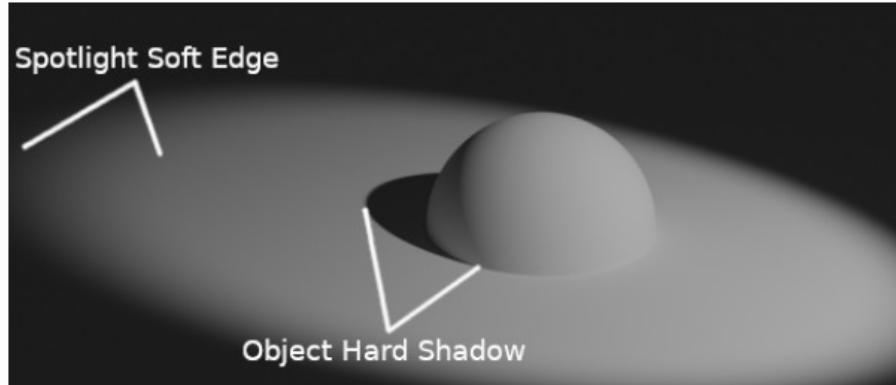
- Parameters:
  - Spotlight direction:  $s$
  - Angle of umbra  $\theta_u$ 
    - Outer cone shell
  - Angle of penumbra  $\theta_p$ 
    - Inner cone where the light is at full intensity
  - Color: RGB vector in  $[0, 1]$
  - Intensity: float in  $[0, \inf]$



$$t = \text{clamp}(0, 1, \frac{\cos(\theta_s) - \cos(\theta_u)}{\cos(\theta_p) - \cos(\theta_u)})$$

$$f_{\text{directional}}(l) = t^2$$

$$L_i = (\text{light intensity} * \text{light color}) f_{\text{distance}}(r) f_{\text{direction}}(l)$$



# Other distance and directional light falloff functions

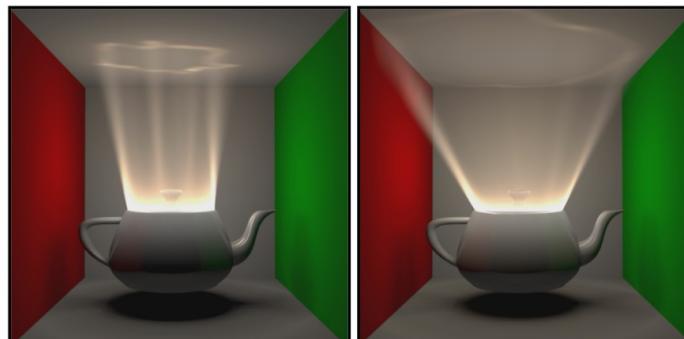
- **Textured lights**

- Use of **texture** for visual richness of light sources: intensity, color and directional/distance variations
- For **cone-type lights**: projective textures (slide projector effect, gobo/cookie lights)
- For **omni-directional-type lights**: texture for distance falloff – attenuation maps (e.g., light beams)

$$L_i = (\text{light intensity} * \text{light color}) f_{distance}(r) f_{direction}(l)$$



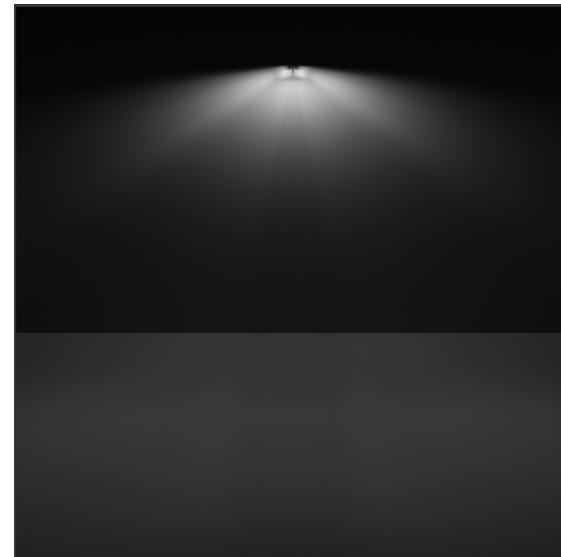
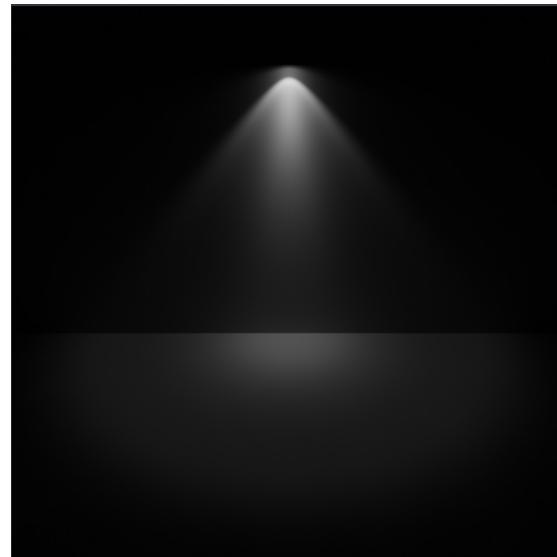
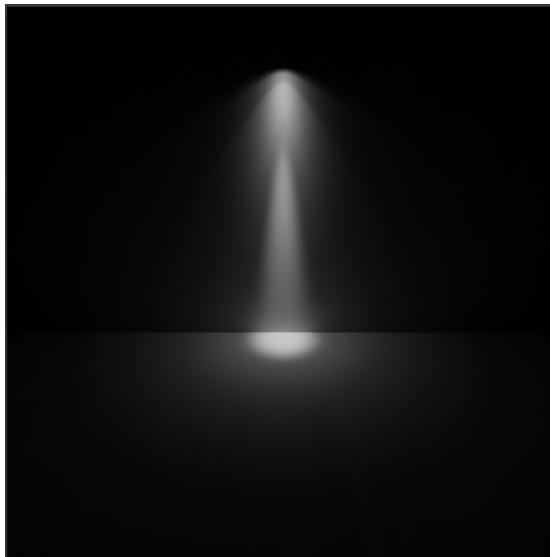
[https://developer.valvesoftware.com/wiki/Env\\_projectedtext ure](https://developer.valvesoftware.com/wiki/Env_projectedtext ure)



[https://www.researchgate.net/publication/220183756\\_A\\_Programmable\\_System\\_for\\_Artistic\\_Volumetric\\_Lighting](https://www.researchgate.net/publication/220183756_A_Programmable_System_for_Artistic_Volumetric_Lighting)

# Other distance and directional light falloff functions

- Tabulated patterns
  - Standard: Illuminating Engineering Society (IES profiles)
  - Measured from the real world using: Goniophotometer

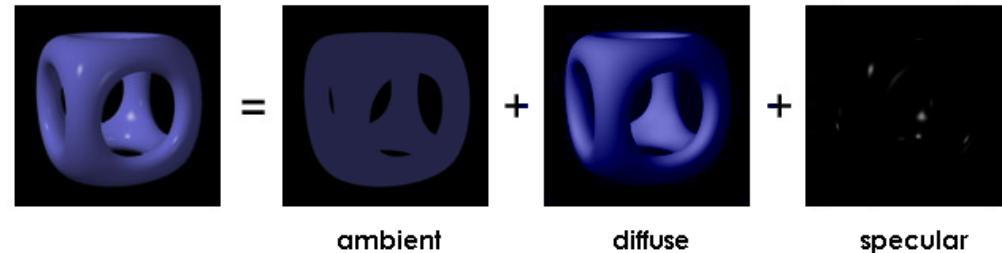


IES profiles: <https://ieslibrary.com/en/home>

# Environment illumination

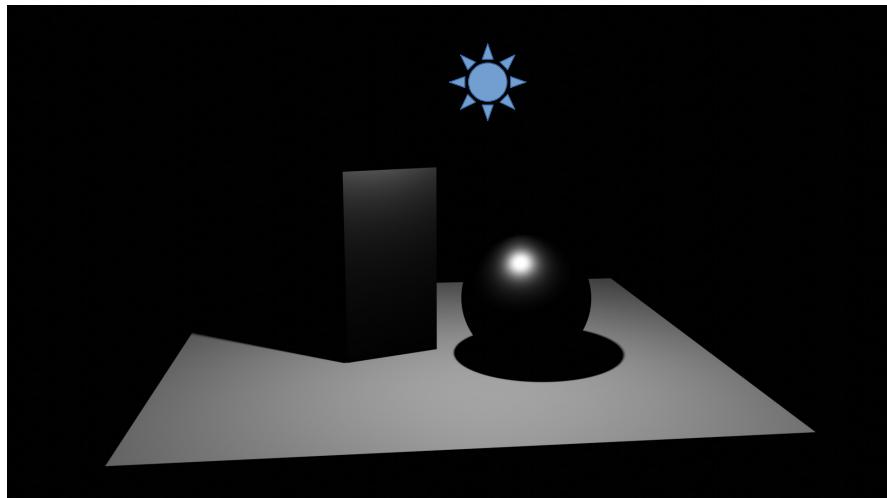
# Environment illumination

- Point and directional lights limit surfaces to receive light from small number of discrete directions
- In reality, **scenes receive light from all directions** otherwise surfaces would appear too dark or completely black if facing away from light sources
  - Example: ambient term in Phong model; constant color added to each point of surface
- To approximate light coming for different directions and large sources of light such as sky and clouds, **environment lighting** is used.

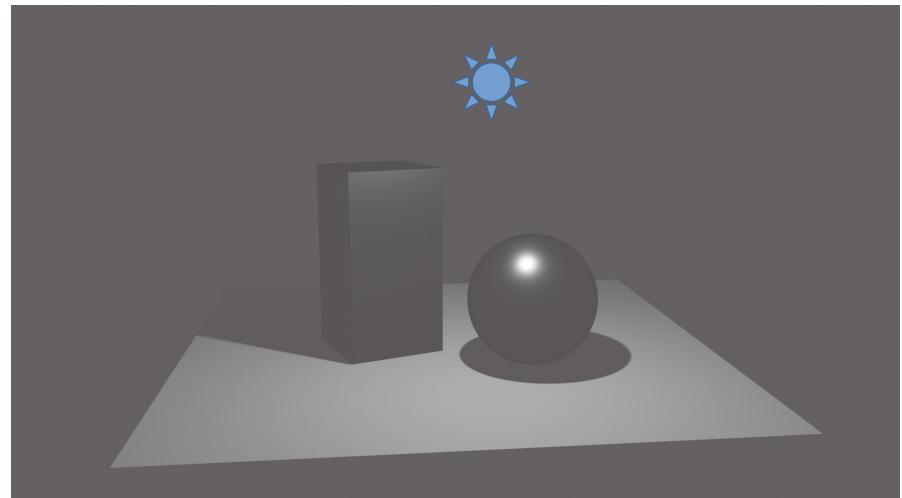


# Environment illumination

- Light coming from point or directional light sources can be described with discrete incoming light directions to surface point
- Environment illumination is described with light coming from all directions to surface point



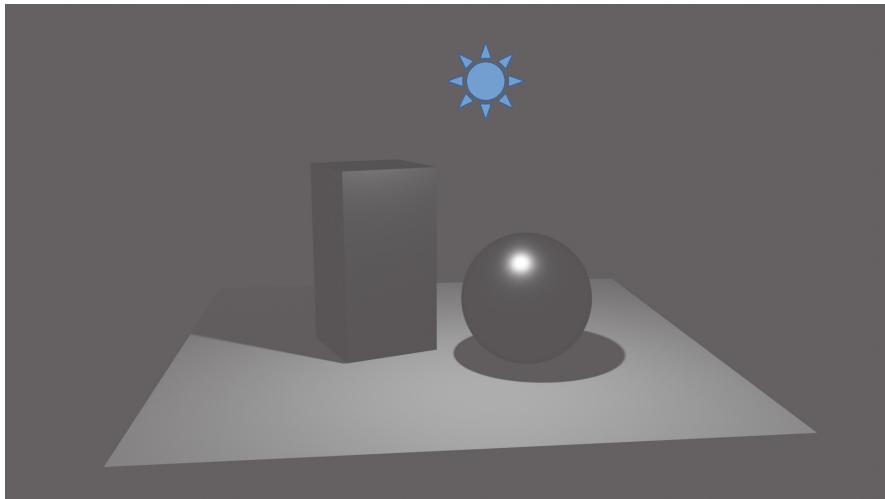
Only point light



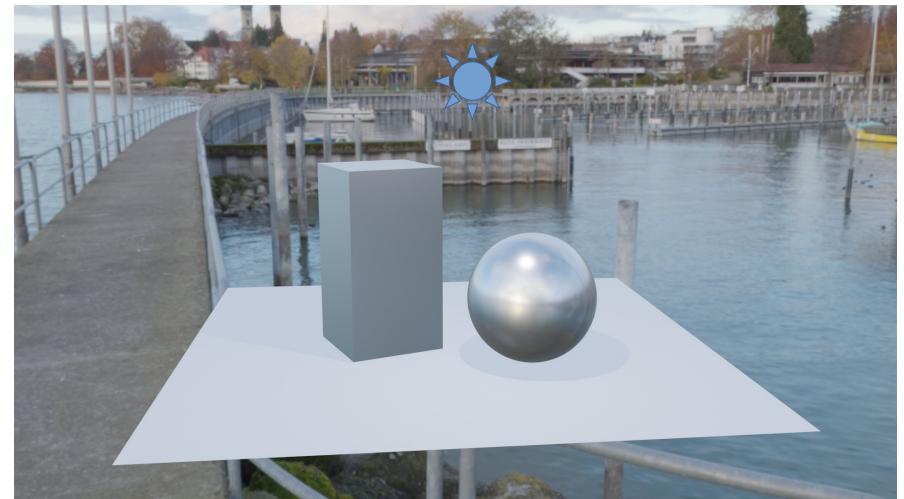
Point light and constant environment

# Environment illumination

- Environment illumination is described with light coming from all directions to surface point
- Variation in intensity from different directions can be obtained using texture



Point light and constant environment



Point light and textured environment

# Environment illumination

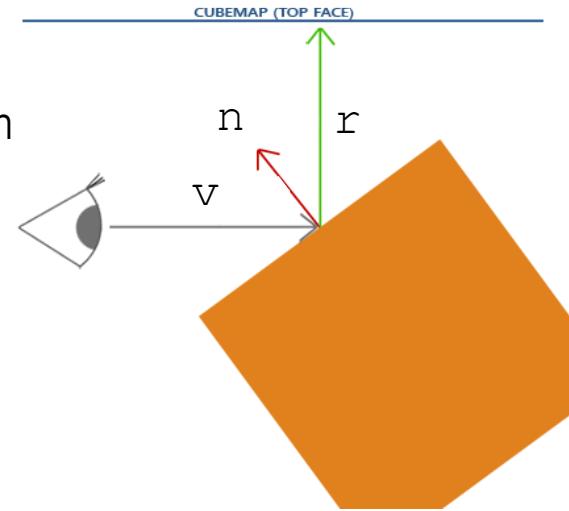
- Note that environment illumination is not global illumination
  - Knowledge of other surfaces which might reflect light is not used
  - Only more directions of light from light emitter are used
- However, this approximation can be used for simulating global illumination effects, that is, indirect light.

# Environment illumination

- Variation in intensity depends only on incoming direction and not position
- Such illumination can be represented by spherical functions
  - Defined over surface of unit sphere or space of directions
- Spherical function representations:
  - **Tabulated forms**: discrete points/directions over sphere containing intensity values which are interpolated
  - **Spherical basis**: spherical radial basis functions, spherical harmonics, etc.
  - **Environment mapping**: storing spherical function into texture, e.g., image texture
    - Most widely used, more memory costly but fast compared to other representations

# Environment mapping

- Environment illumination is stored into image texture → environment map
  - Unlike surface textures, environment textures have color intensities larger than 1 → high dynamic range
- Simplest case of environment mapping: reflection mapping
  - For each reflective material, compute normal ( $n$ ) at the location on the surface of the object
  - Compute reflection vector ( $r$ ) from view vector ( $v$ ) and normal vector ( $n$ )
$$r = 2(n \cdot v)n - v$$
  - Use reflected view vector ( $r$ ) to compute environment map texture location
  - Read value from environment map texture and use it as light intensity

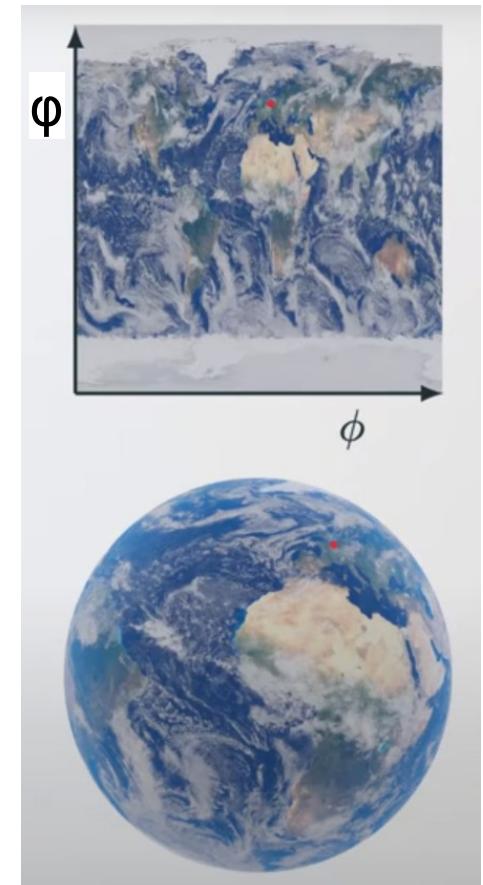


# Reflection mapping

- Variety of projector functions mapping from reflected view vector ( $r$ ) to environment texture location:
  - Latitude-longitude mapping
  - Sphere mapping
  - Cube mapping

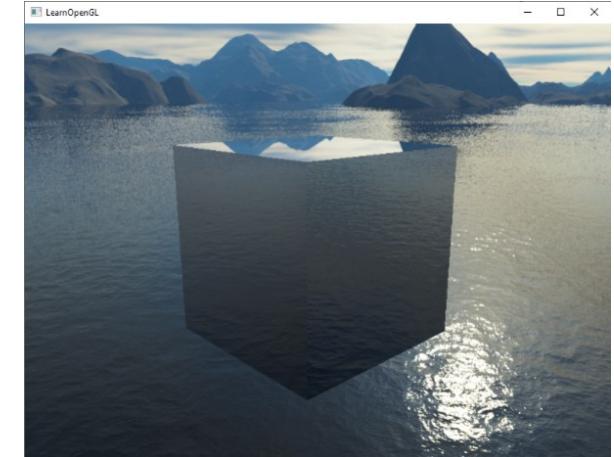
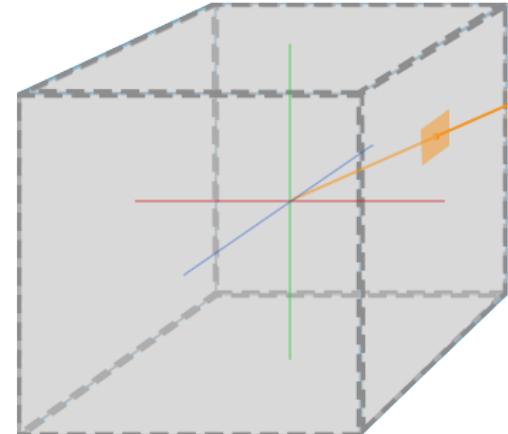
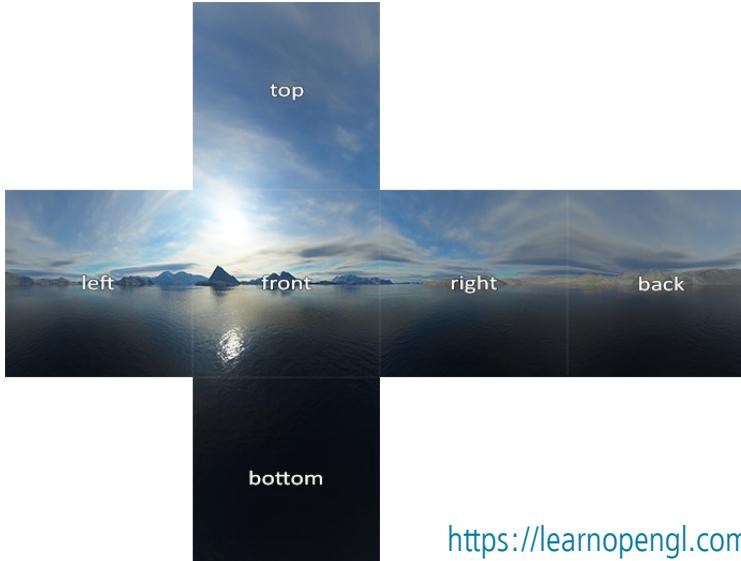
# Latitude-longitude mapping

- Blinn and Newell method, first environment mapping method, 1976
  - Environment map idea: imagine that globe is viewed from inside
- Reflected view vector  $r$  ( $r_x$ ,  $r_y$ ,  $r_z$ ) is converted to spherical coordinates:
  - Longitude  $\Phi$  in  $[0, 2\pi]$   $\rightarrow \Phi = \text{atan2}(r_y, r_x)$
  - Latitude  $\phi$  in  $[0, \pi]$   $\rightarrow \phi = \arccos(r_z)$
- Spherical coordinates are used to access the environment map

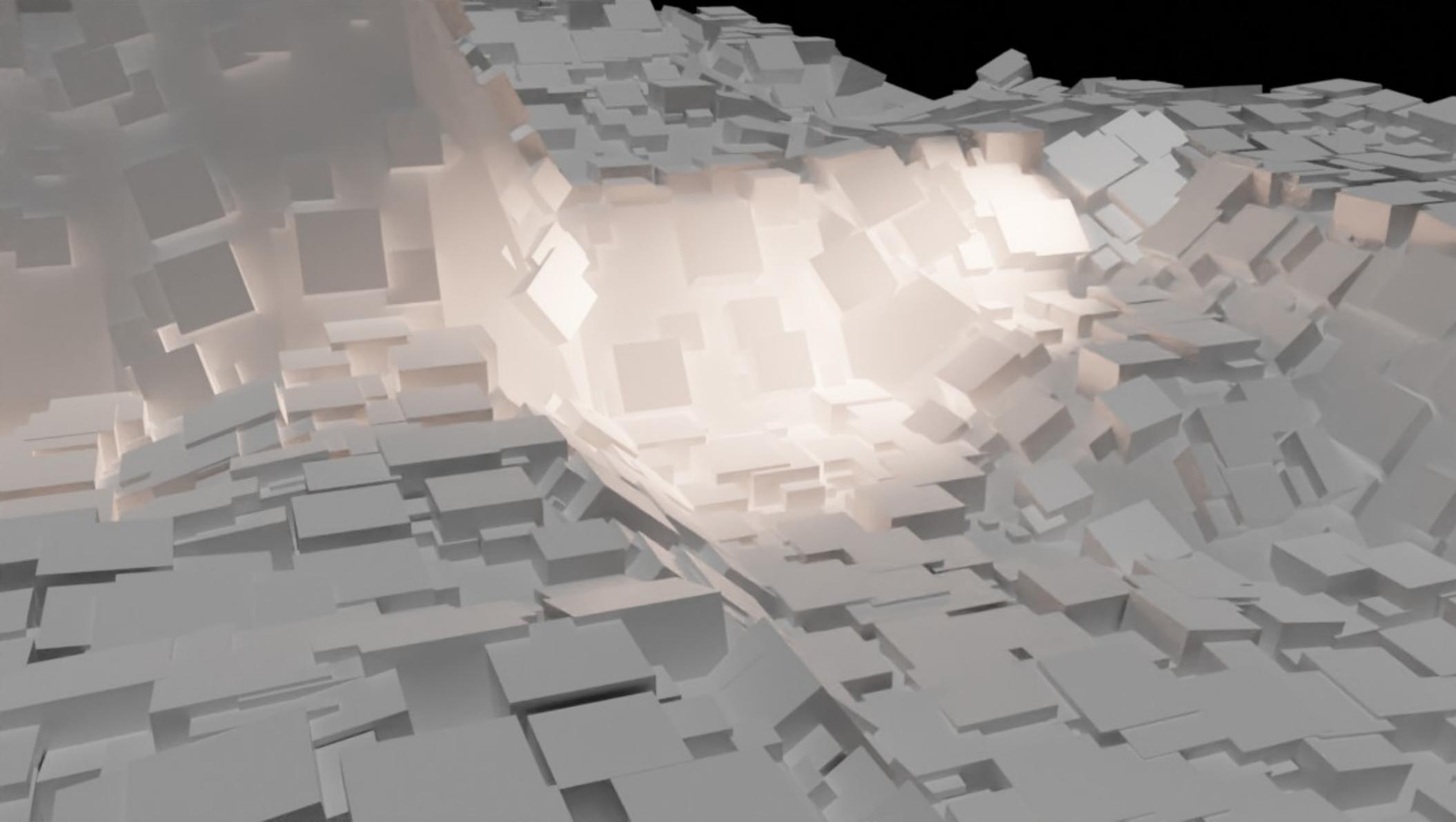


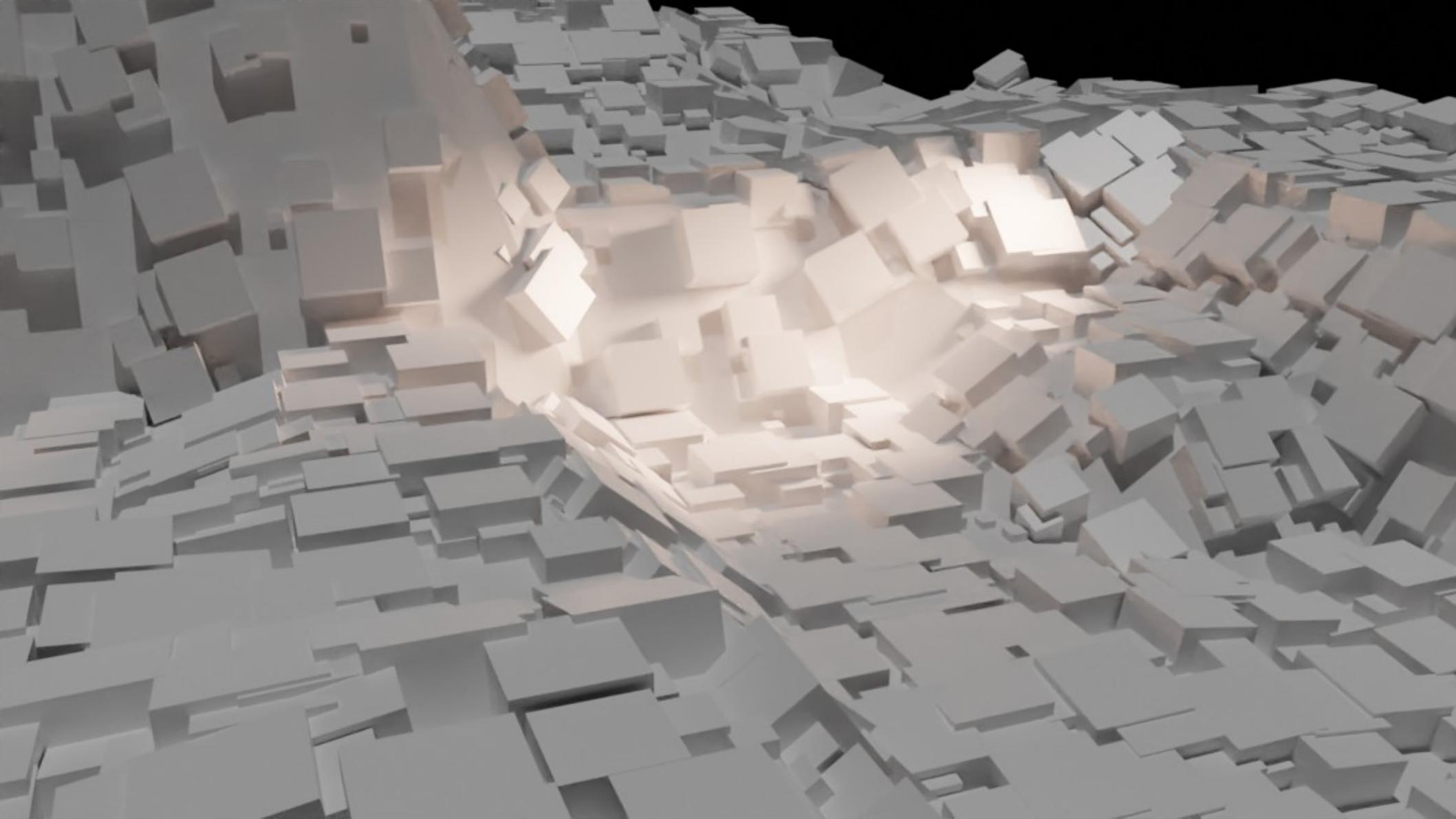
# Cube mapping

- Most widely used method, Greene 1986
- Cube map is created by projecting environment on the sides of a cube positioned with its center at camera's location
- Visualized as "cross" diagram, stored as six square textures
- Reflected view vector ( $r$ ) can be directly used as three-component texture coordinate to fetch intensity data stored in cube map



# Shadows







Scene from:

<https://casual-effects.com/data/index.html>



# Shadows

- Shadows are important for:
  - Visual understanding of **object placement and relation to other object**
  - Realistic image synthesis: needed for depicting **details and surface characteristics**



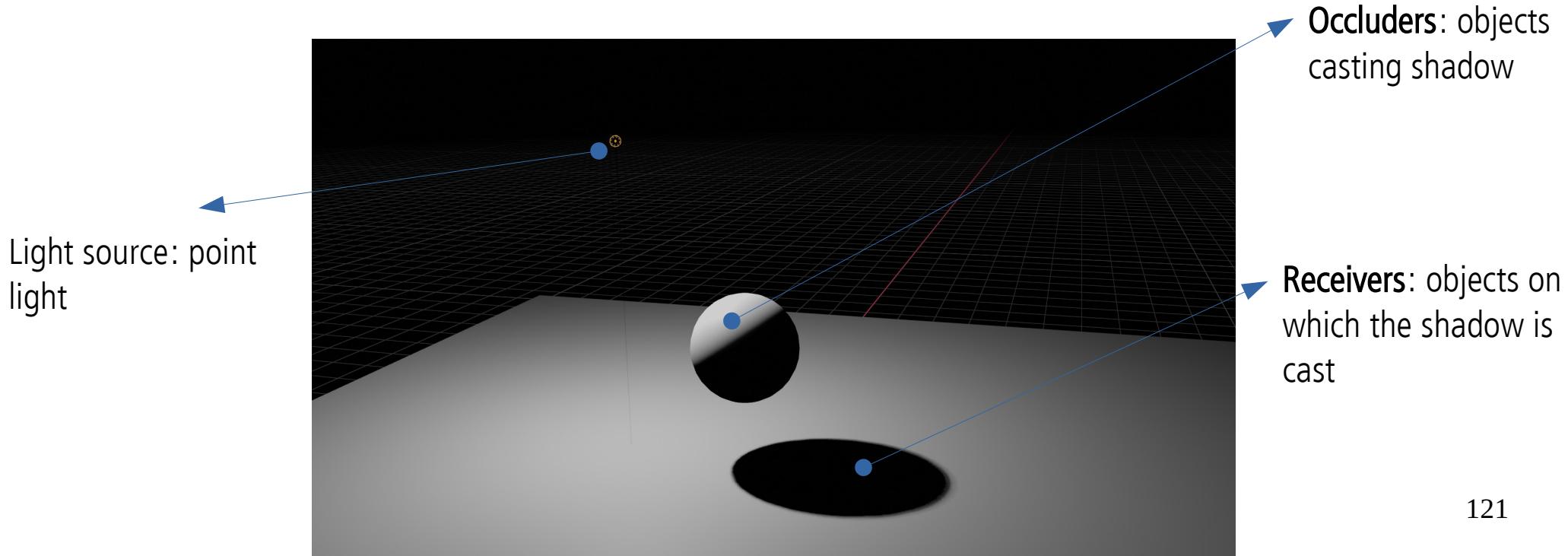
<https://creativecoding.soe.ucsc.edu/courses/cs488/reportsA/shadows.pdf>



<https://lumion.com/blog/feature-spotlight-sky-light-soft-shadows-and.html>

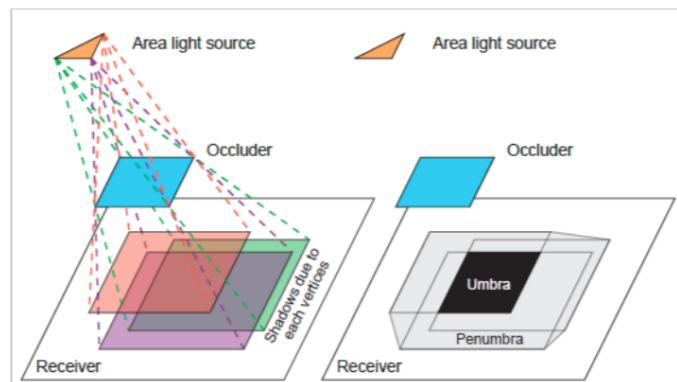
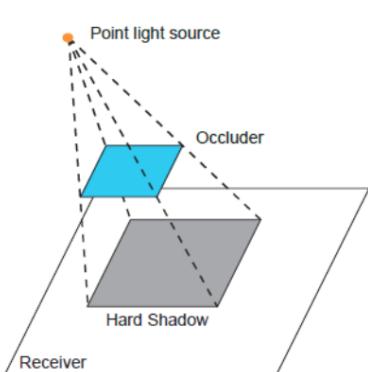
# Light and shadow

- Light falling on object causes object to cast shadow – blocking light to fall on another surface



# Sharp vs smooth shadows

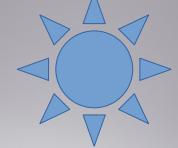
- Size of light determines shadow smoothness
  - Hard-edge shadows: sharp shadow edges
  - Soft shadows: fuzzy shadow edges
  - Softer shadow edges are more realistic, but more expensive to render



Small light sources (e.g., point lights) → hard shadows; only umbra component



Large light sources (e.g, physical lights) → soft shadows; both umbra and penumbra component

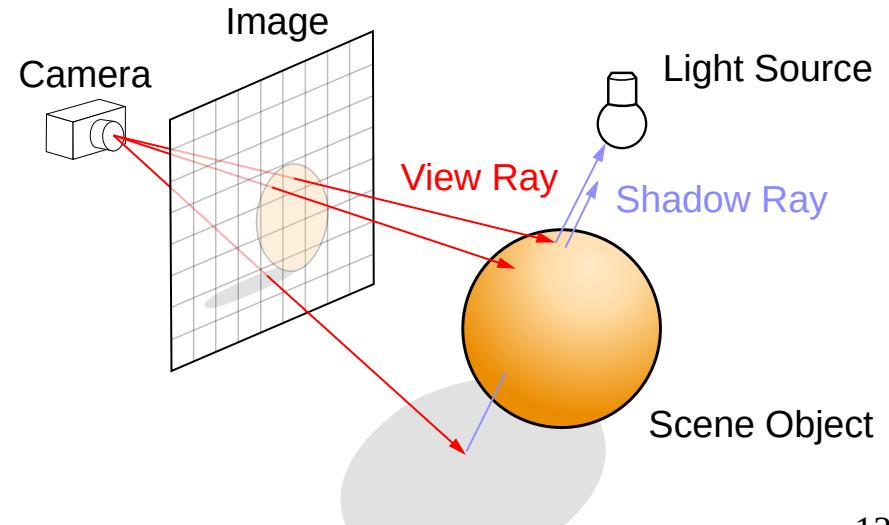
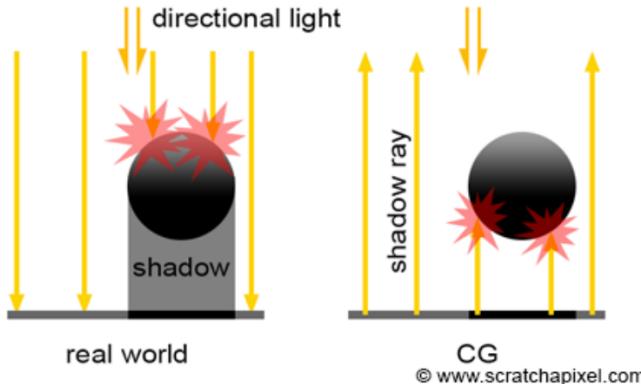


# Rendering and shadow

- Simulating shadow depends on how rendering is solving visibility problem
- In ray-tracing based rendering shadows can be calculated in single pass without complex workarounds
- In rasterization-based rendering multi-pass rendering or additional work is required to simulate/approximate shadows

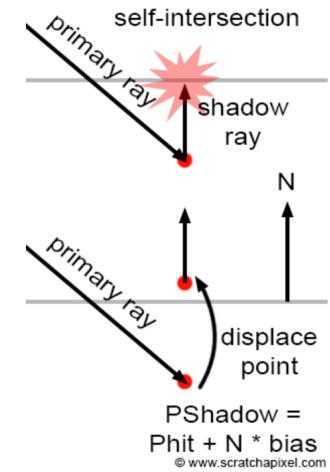
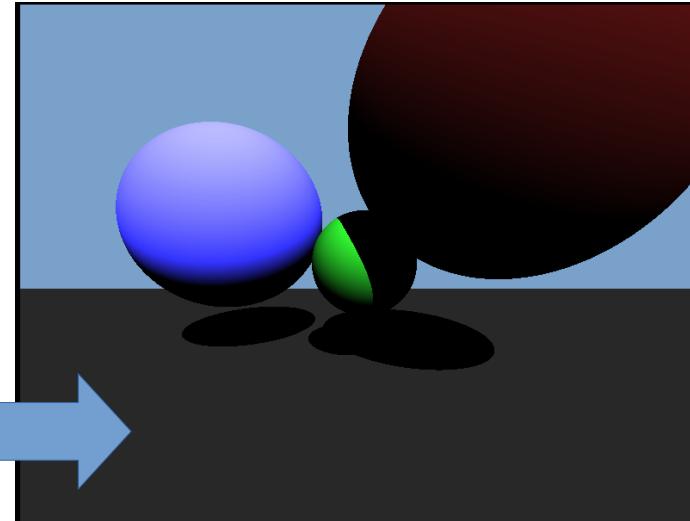
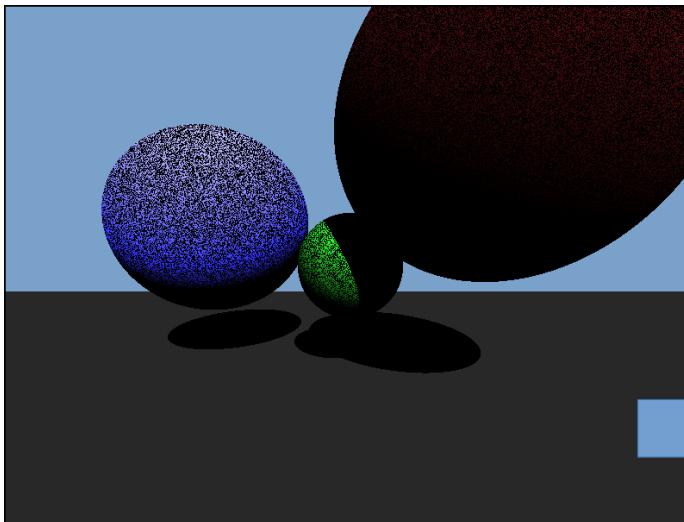
# Shadow in ray-tracing

- For each pixel of virtual image plane, a ray is generated and tested for intersection with objects in 3D scene
- For found intersection, color is calculated using shading
- During shading, additional ray is casted towards light source – **shadow ray**
  - If shadow ray intersects object on its way, then shaded surface point is in shadow
  - If object is opaque, shadow is dark
  - If object is transparent, shadow is brighter



# Shadow in ray-tracing: shadow acne

- Due to numerical limits in precision, **some intersection point positions might end up under the surface**
  - Casting shadow ray from this point causes intersection and thus light obstruction → shadow acne
  - Solution 1: use double float precision
  - Solution 2: Add small displacement to intersection point in direction of normal: **shadow bias parameter**



<https://www.scratchapixel.com/lessons/3d-basic-rendering/introduction-to-shading/light-and-shadows>

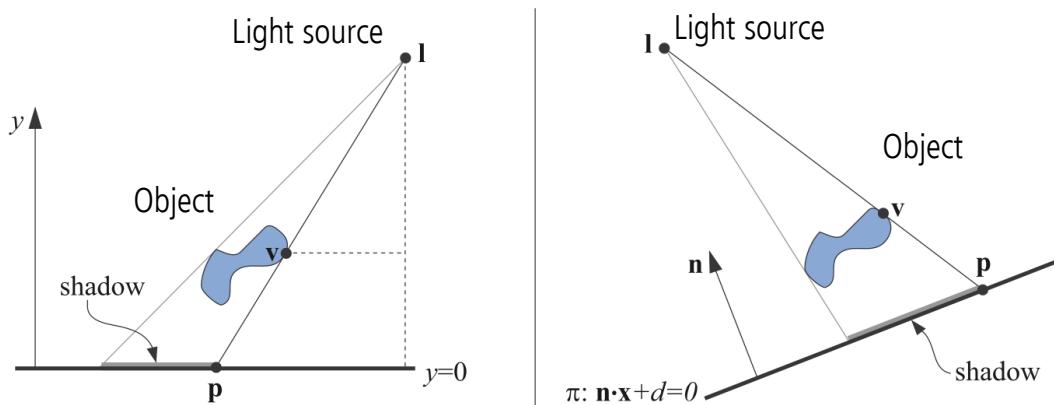
<https://bheisler.github.io/post/writing-raytracer-in-rust-part-2/>

# Shadow in rasterization

- Shadow computation in ray-tracing is elegantly solved due to nature of visibility solving which simulates physical light flow
- In rasterization, all objects in 3D scene are projected onto virtual image plane and information of 3D scene is partially lost
- Therefore, various methods are developed to simulate and introduce shadows
  - Different methods solve different special and simplified cases of shadow occurrence
    - Planar shadows
    - Volume shadows
    - Shadow mapping

# Planar shadows

- Only for objects casting shadow onto a planar surface
- Method: projection shadows



Projection matrix  $M$  is constructed to project object vertices  $v$  onto plane  $\pi$  with respect to the light source  $l$  giving shadow point  $p$ .

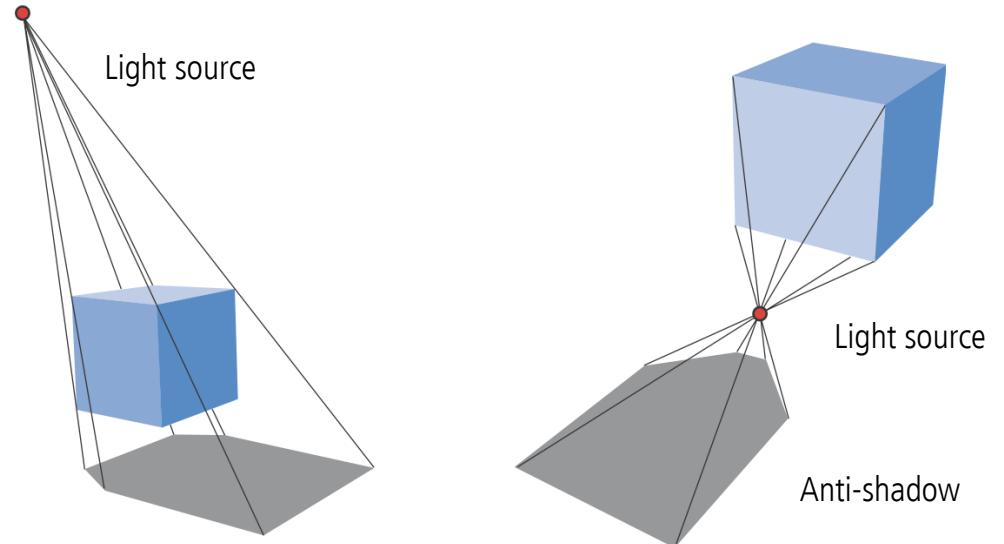
$$M = \begin{pmatrix} n \cdot l + d - l_x n_x & -l_x n_y & -l_x n_z & -l_x d \\ -l_y n_x & n \cdot l + d - l_y n_y & -l_y n_z & -l_y d \\ -l_z n_x & -l_z n_y & n \cdot l + d - l_z n_z & -l_z d \\ -n_x & -n_y & -n_z & n \cdot l \end{pmatrix}.$$

# Projection shadows

- To render shadow, projection matrix  $M$  is applied to all objects (their vertices  $v$ ) which should cast shadow on plane  $\pi$
- Projected object is obtained which is rendered with darker color and no illumination.

Notes:

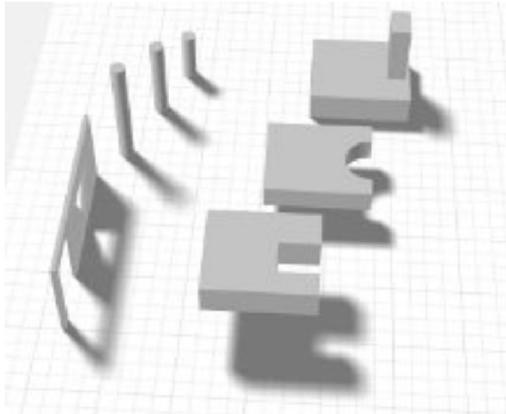
- Projected object must be above plane  $\pi$  (introduce bias parameter controlling distance)
- Shadows may fall outside of plane  $\pi$  – this breaks the illusion and clipping must be performed
- Anti-shadow can be generated if light is between object and plane  $\pi$



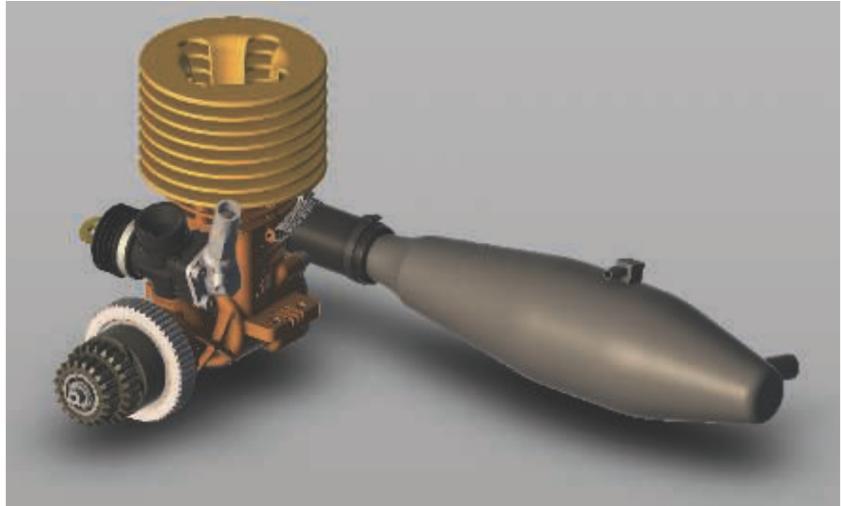
# Projection shadows

- Soft shadows can be simulated.

Simulating area light: multiple point lights are placed where area light would be placed, resulting shadows are accumulated and averaged  
(Heckbert and Herf's method)

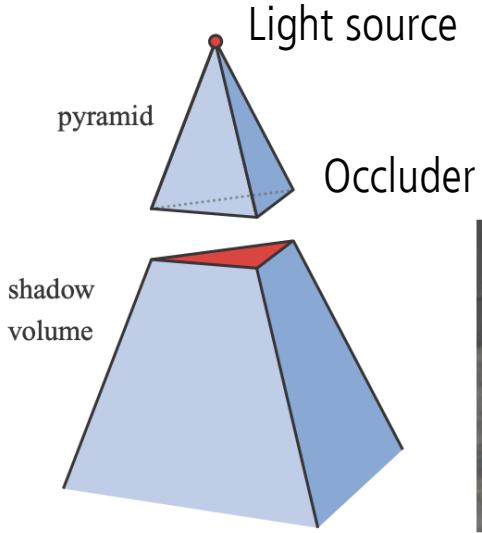
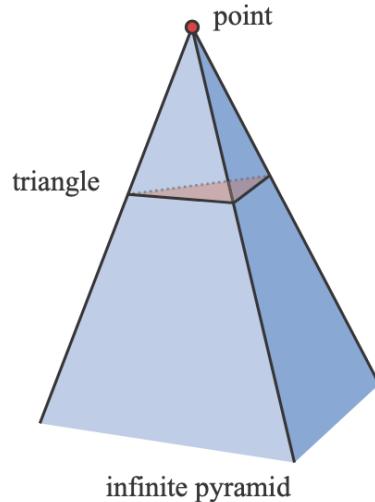


Blurring hard-shadow generated from a single point light.



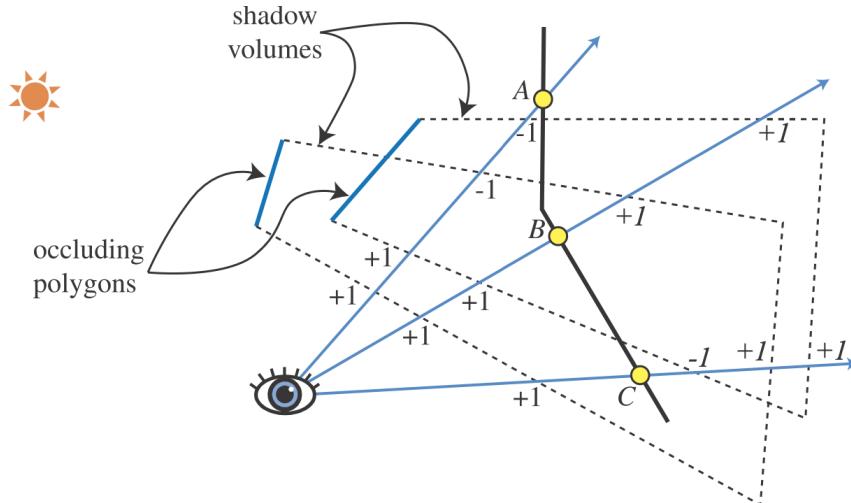
# Shadow volumes

- More general approach for receivers of different shapes
- **Volume of space is formed by occluder and light source**
  - Any object inside this volume is in shadow



# Shadow volumes

- To determine if object is in shadow, counting method is used.
  - Ray is traced from view to object
  - Every time when ray passes through front-facing triangle of shadow volume the counter is increased
  - Every time when ray passes through back-facing triangle of shadow volume the counter is decreased
  - If counter is equal to zero than object is illuminated (not in shadow)

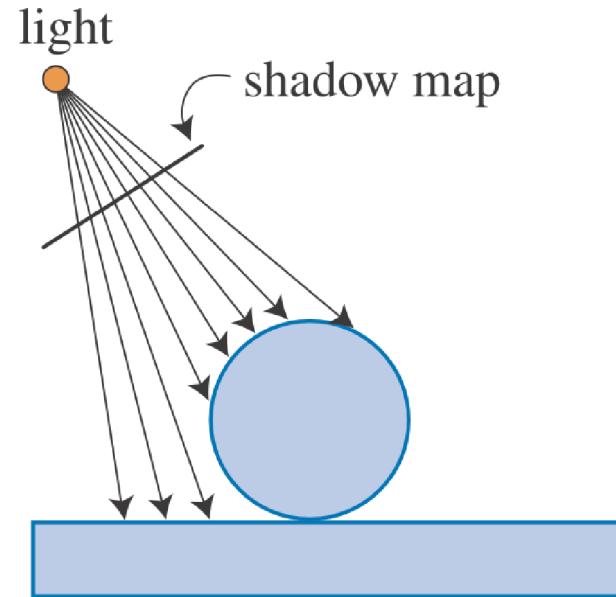


# Shadow volumes

- Drawback: extreme variability
  - High difference in computation time depending on camera, object and light position
  - If camera is in shadow volume, the shadow volume fills the whole screen resulting in high computation → slow rendering
  - Consistent frame-rate is not guaranteed
- On GPU rasterization-based rendering counting is performed using depth and stencil buffers\*

# Shadow maps

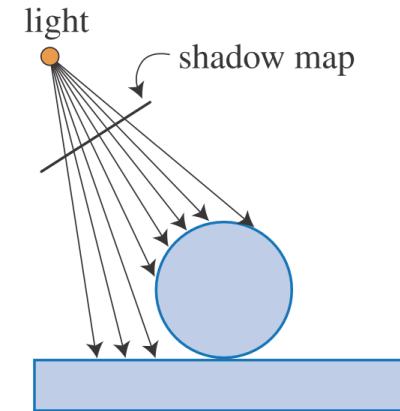
- Efficient rendering of shadows on various receivers
- Idea: solve shadow casting as visibility problem
  - Points visible to light source are illuminated
  - Points invisible to light source are in shadow
- GPU rasterization uses z-buffer to solve visibility and store distance to closest objects in depth buffer\*



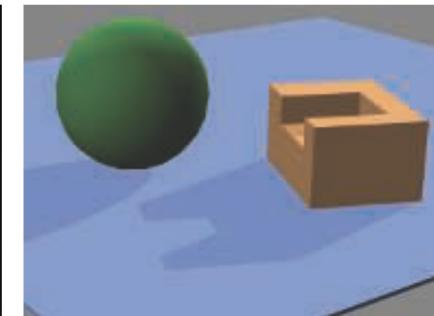
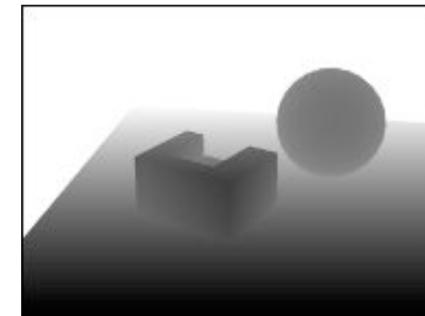
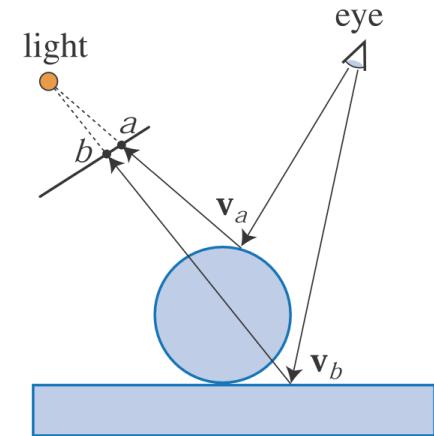
# Shadow maps

- Render the scene from position of light source that is to cast shadows
  - Store distances of visible objects in a texture → **shadow map**
- Render the scene once more, now from camera point of view → **multi-pass rendering**
  - Distance of rendered points to light source are compared to distances stored in shadow map
  - If point is farther from light source than the corresponding value in shadow map, then point is in shadow, otherwise it is not

Rendering from light source



Rendering from camera



# Shadow maps

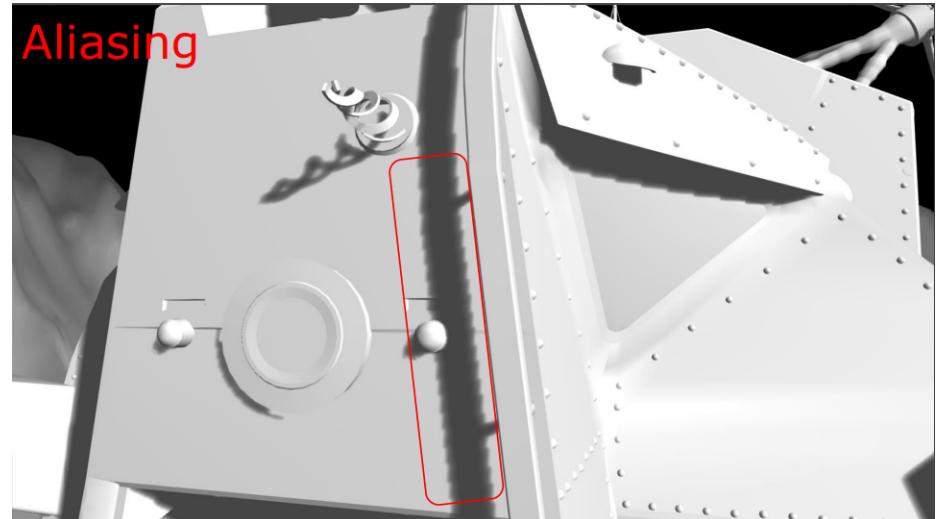
- **Shadow map creation depends on type of light:**
  - Distant lights: orthographic projection is used since light direction is parallel
  - Spotlight: frustum is defined with spotlight shape
  - Point light: omnidirectional shadow maps – six view cube (similar to environment mapping)
- **Shadow map approach is predictable**
  - Shadow map contains only objects that can cast shadows – objects inside light frustum
  - Cost of building shadow map is linear with number of triangles
  - Shadow map can be generated once and reused for all frames where light and objects are not moving

# Shadow maps

- Quality of shadows depend on shadow map resolution
  - Low shadow map resolution highlights aliasing
- Quality of shadows depends on objects distance to light source → More samples for shadow map is needed for better shadow map approximation

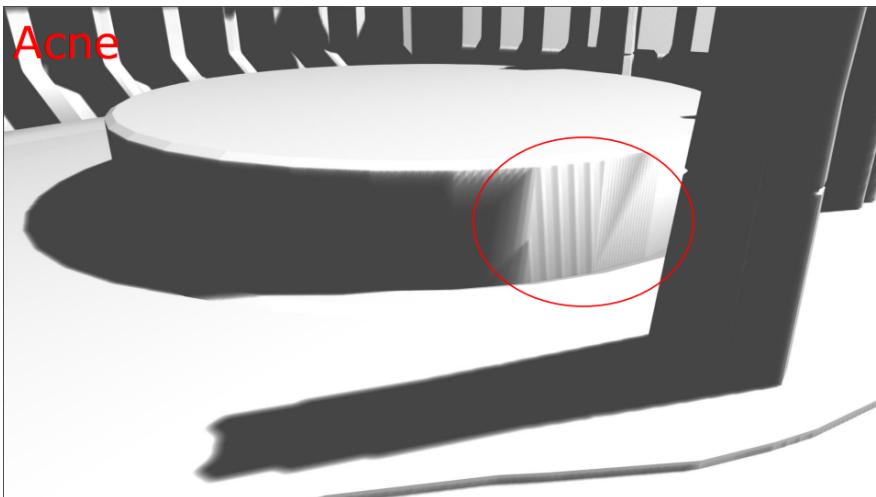


Quality of shadows depend on shadow map resolution

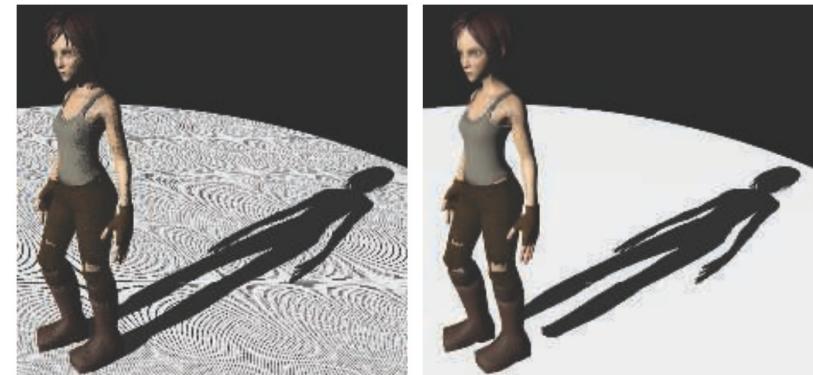


# Shadow maps

- **Self-shadow aliasing and shadow acne** becomes apparent due to numerical limits and slight differences in light and camera view values
  - **Bias parameter** must be introduced which is correcting distances in camera and light view



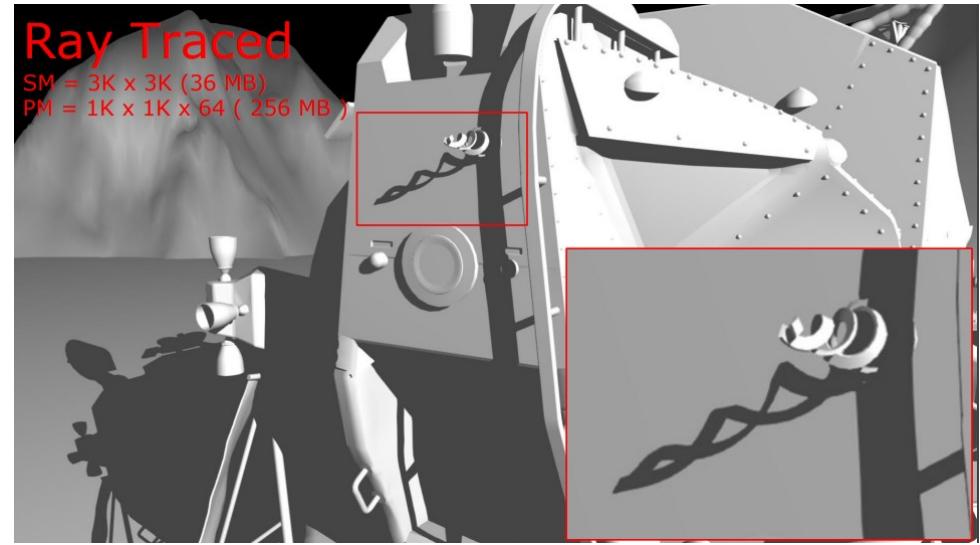
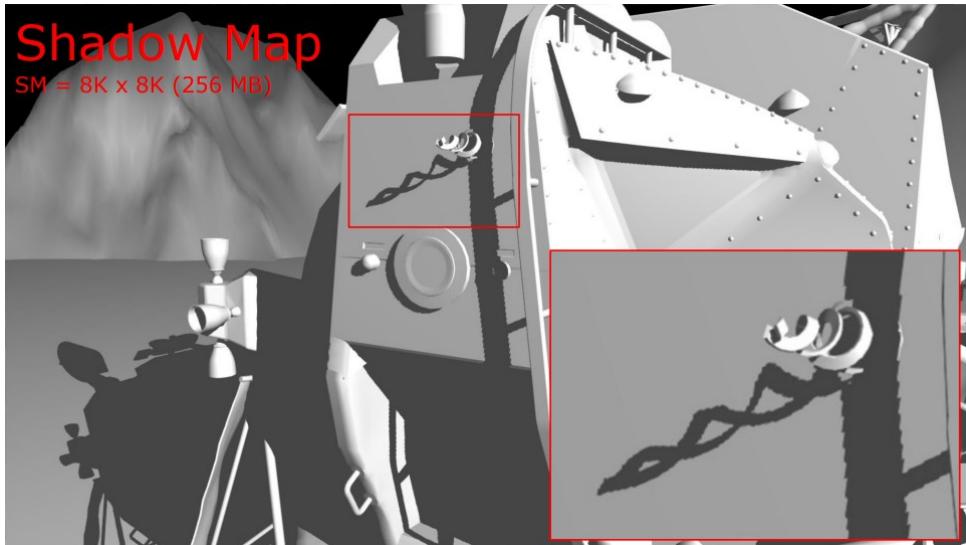
Self-shadowing (shadow acne) → moire pattern



Ground self-shadowing leads to aliasing (Moire pattern)

# Hybrid ray-traced shadows\*

- Shadow maps are most often used method and still exhibit a lot of problems: aliasing, acne, etc.
- Solution is to combine rasterization-based rendering with ray-tracing for shadow computation



# Light and Shadow: Tips and Tricks

- Global illumination gives correct light reflection and shadows effects but it is expensive for real-time applications
- Physical lights more correctly describe light color and intensity distribution but are expensive to compute in real-time
- Non-physical light sources and direct illumination can be solved fast and efficiently and can to some extent approximate shadows and global illumination effects but require additional work

# Light and Shadow: Tips and Tricks

- When creating 3D scene, artists understand the how the scene should be illuminated. Using this knowledge they place non-physical lights which are rendered with direct illumination to simulate global-illumination effects such as reflections and shadows.



<https://80.lv/articles/dishonored-interiors-lighting-props/>

# More into topic

- Lighting for artists:
  - <https://www.artstation.com/channels/lighting>
  - <https://80.lv/articles/amazing-lighting-for-simple-scenes/>
  - [https://www.youtube.com/watch?v=Ys4793edotw&ab\\_channel=BlenderGuru](https://www.youtube.com/watch?v=Ys4793edotw&ab_channel=BlenderGuru)
- Wave effects in rendering:
  - <https://developer.nvidia.com/gpugems/gpugems/part-i-natural-effects/chapter-8-simulating-diffraction>
  - [https://sites.cs.ucsb.edu/~lingqi/publications/202203\\_practical\\_plt\\_paper\\_lowres.pdf](https://sites.cs.ucsb.edu/~lingqi/publications/202203_practical_plt_paper_lowres.pdf)
- Path-tracing lights
  - [https://www.pbr-book.org/3ed-2018/Light\\_Sources/Area\\_Lights](https://www.pbr-book.org/3ed-2018/Light_Sources/Area_Lights)
  - [http://graphics.stanford.edu/papers/veach\\_thesis/](http://graphics.stanford.edu/papers/veach_thesis/)
  - <https://schuttejoe.github.io/post/arealightsampling/>
  - <https://www.scratchapixel.com/lessons/3d-basic-rendering/global-illumination-path-tracing/introduction-global-illumination-path-tracing.html>
- Area lights approximation in rasterization-based rendering (OpenGL): <https://learnopengl.com/Guest-Articles/2022/Area-Lights>
- Shadows in rasterization:
  - [https://www.realtimeshadows.com/sites/default/files/Playing%20with%20Real-Time%20Shadows\\_0.pdf](https://www.realtimeshadows.com/sites/default/files/Playing%20with%20Real-Time%20Shadows_0.pdf)
  - <https://www.cs.montana.edu/courses/spring2005/525/students/Thiesen2.pdf>
  - [https://web.cse.ohio-state.edu/~shen.94/781/Site/Slides\\_files/shadow.pdf](https://web.cse.ohio-state.edu/~shen.94/781/Site/Slides_files/shadow.pdf)
  - [https://cg.informatik.uni-freiburg.de/course\\_notes/graphics\\_07\\_shadows.pdf](https://cg.informatik.uni-freiburg.de/course_notes/graphics_07_shadows.pdf)
- Hybrid shadow computation methods (rasterization + ray-tracing):
  - <https://gpuopen.com/learn/hybrid-shadows/>
  - <https://developer.nvidia.com/content/hybrid-ray-traced-shadows>

# Summary questions

- [https://github.com/lorentzo/IntroductionToComputerGraphics/tree/main/lectures/9\\_light](https://github.com/lorentzo/IntroductionToComputerGraphics/tree/main/lectures/9_light)

# Literature

- <https://github.com/lorentzo/IntroductionToComputerGraphics>