Introduction to Computer Graphics

Originally authored by:
Toshiya Hachisuka

Presented by:
Kenshi Takayama

Last Time

- Introduction to ray tracing

- Intersection algorithms with a ray

  - Sphere

  - Implicit surface

  - Triangle

- GLSL Sandbox

Last Time

```
for all pixels {
        ray = generate_camera_ray( pixel )
        for all objects {
            hit = intersect( ray, object )
            if "hit" is closer than "first_hit" {first_hit = hit}
        }
        pixel = shade( first_hit )
}
```
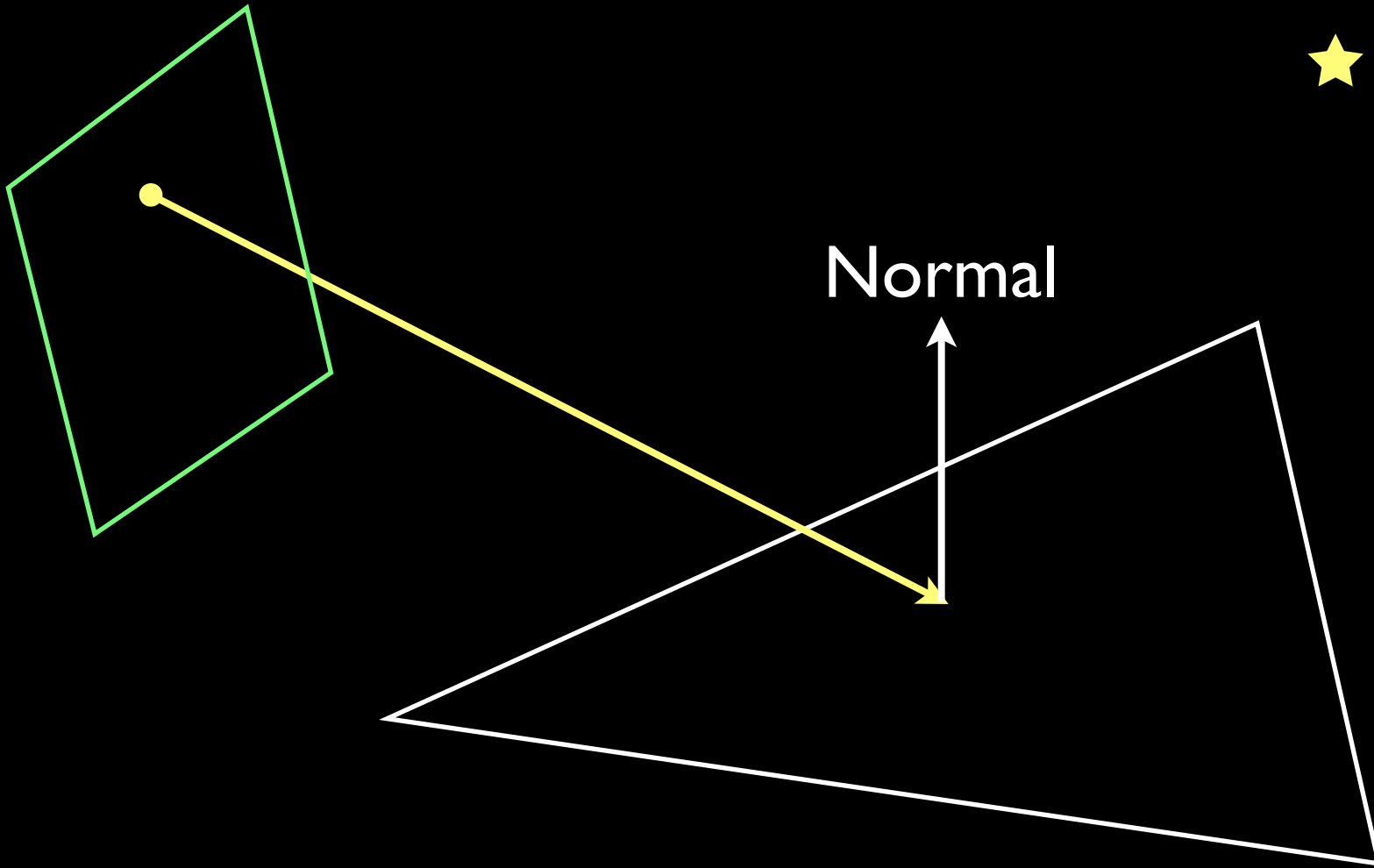
Today

```
for all pixels {
        ray = generate_camera_ray( pixel )
        for all objects {
            hit = intersect( ray, object )
            if "hit" is closer than "first_hit" {first_hit = hit}
        }
        pixel = shade( first_hit )
}
```

Today

- Shading models
  - BRDF
  - Lambertian
  - Specular
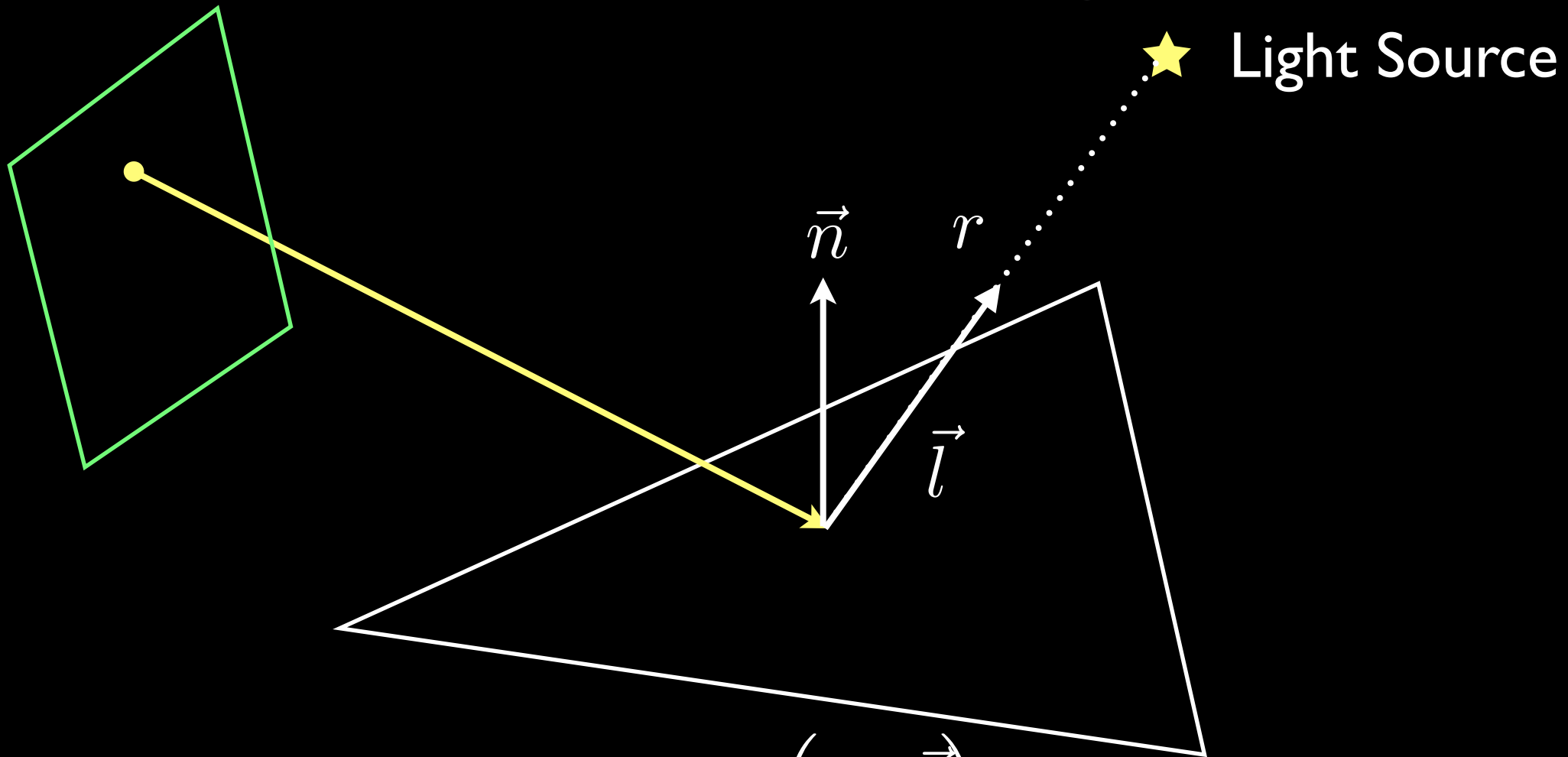- Simple lighting calculation
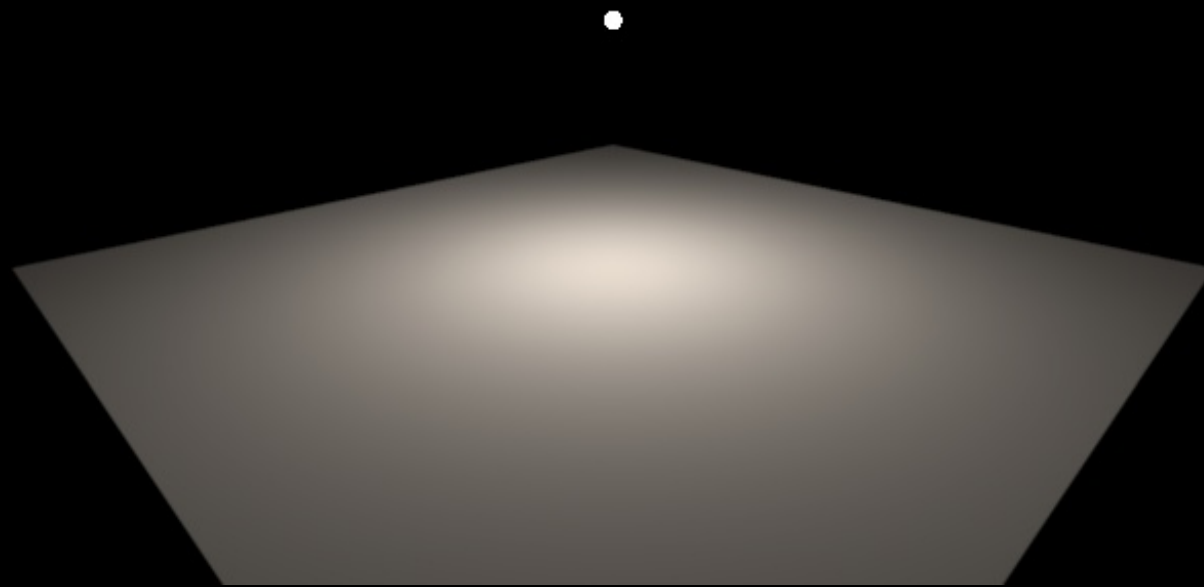- Tone mapping

Basic Shading

★ Light Source

Normal

How much light illuminate this point?

Basic Shading

Light Source

$\vec{n}$ r

$\vec{l}$

Irradiance $E = \dfrac{\Phi\left(\vec{n} \cdot \vec{l}\right)}{4\pi r^2}$
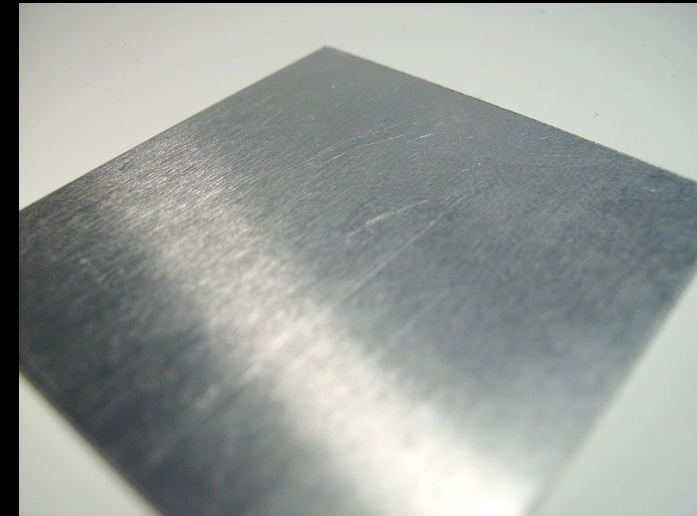
Basic Shading

Irradiance $E = \dfrac{\Phi \left(\vec{n} \cdot \vec{l} \right)}{4\pi r^2}$

Surface Appearance

- How is light reflected by

  - Mirror
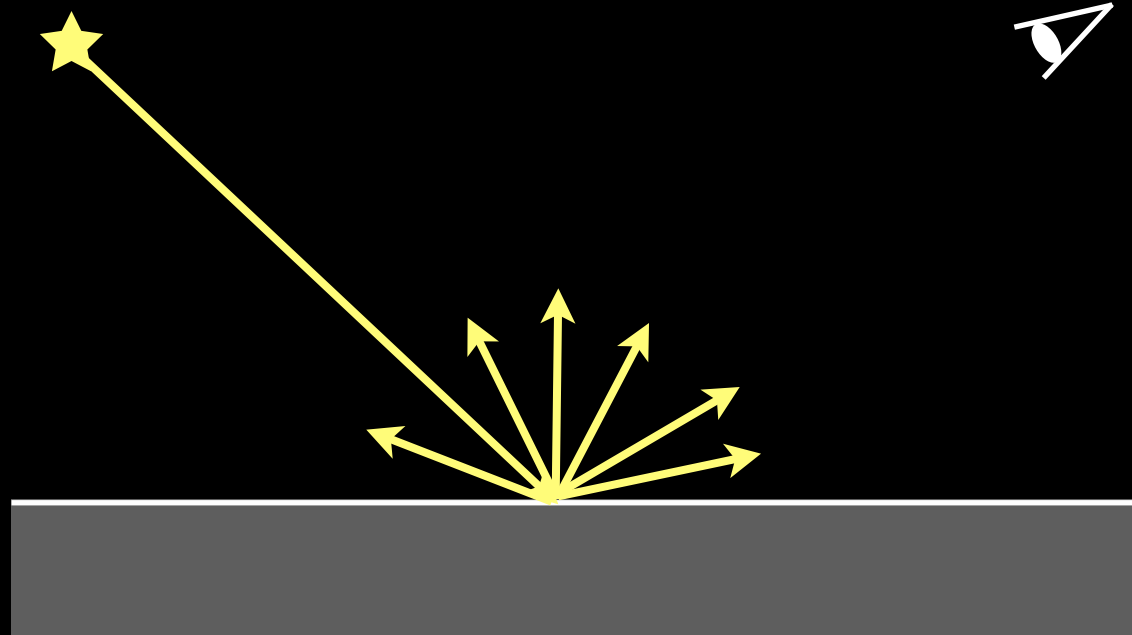
  - Paper

  - Rough metallic surface

Types of Reflections

- Diffuse: matte paint, paper

- Glossy: plastic, rough metallic surface

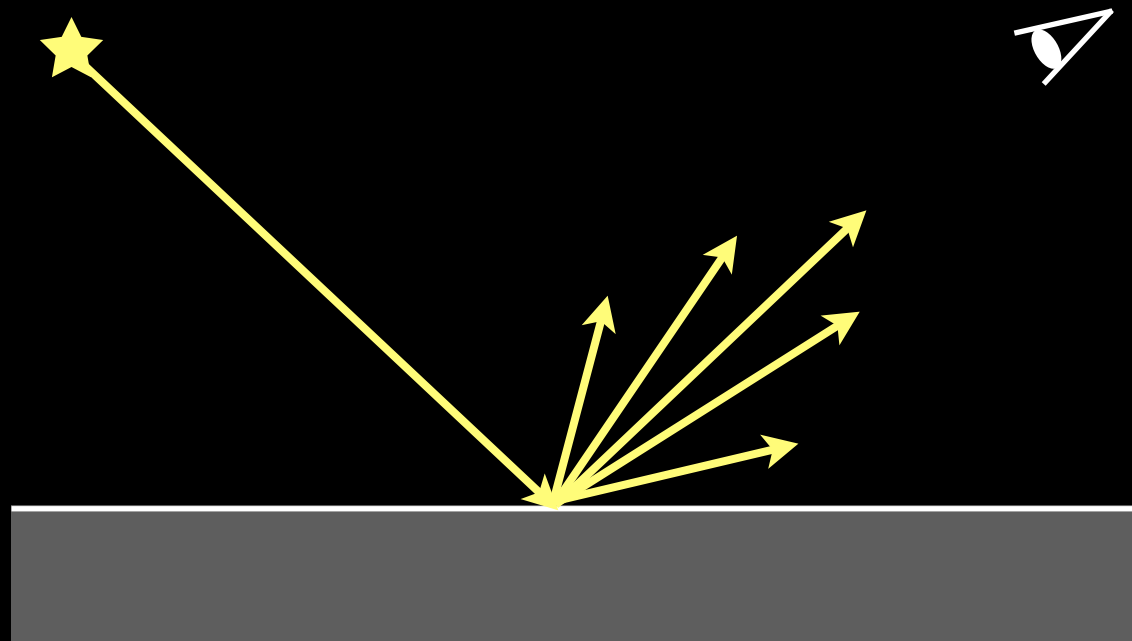- Specular: mirror

- Retro-reflective: the moon

BRDF

- **B**idirectional **R**eflectance **D**istribution **F**unction
  - How light is reflected off a surface
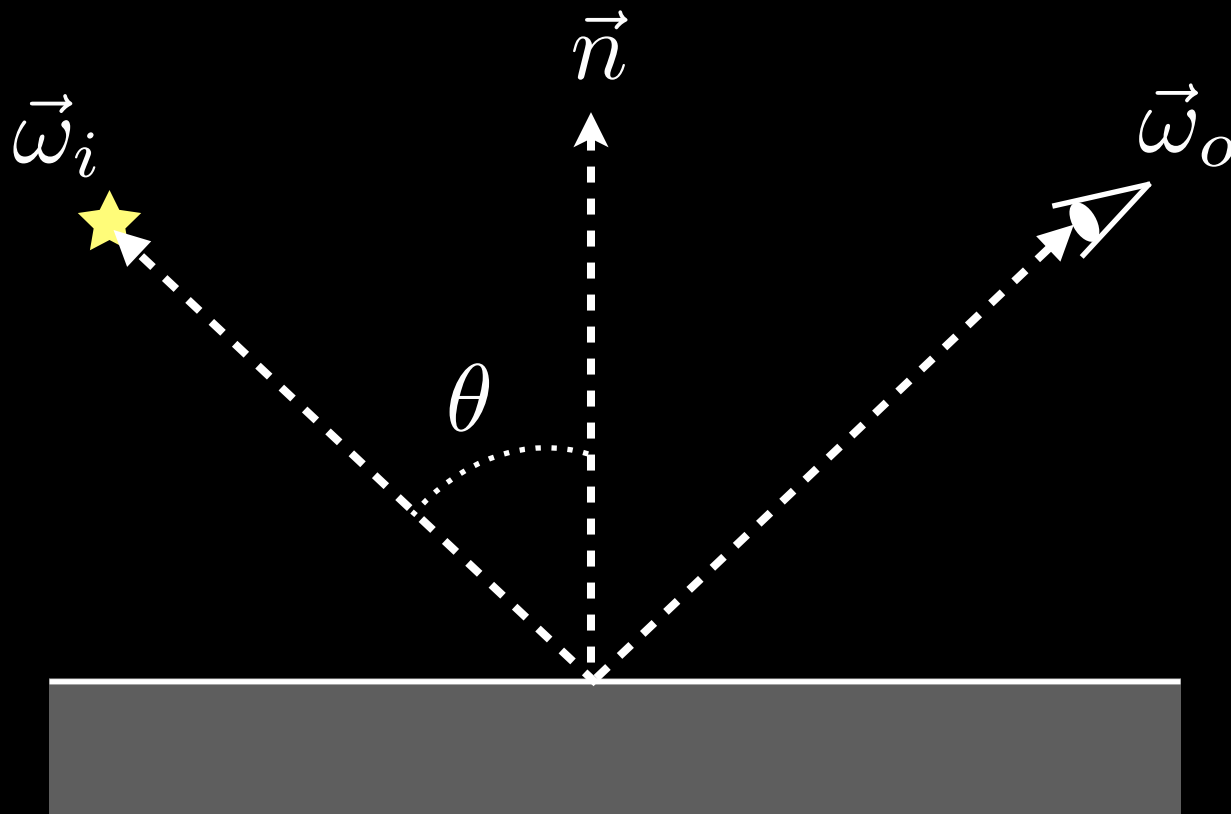 - Capture appearance of surface

BRDF

- **B**idirectional **R**eflectance **D**istribution **F**unction
  - How light is reflected off a surface
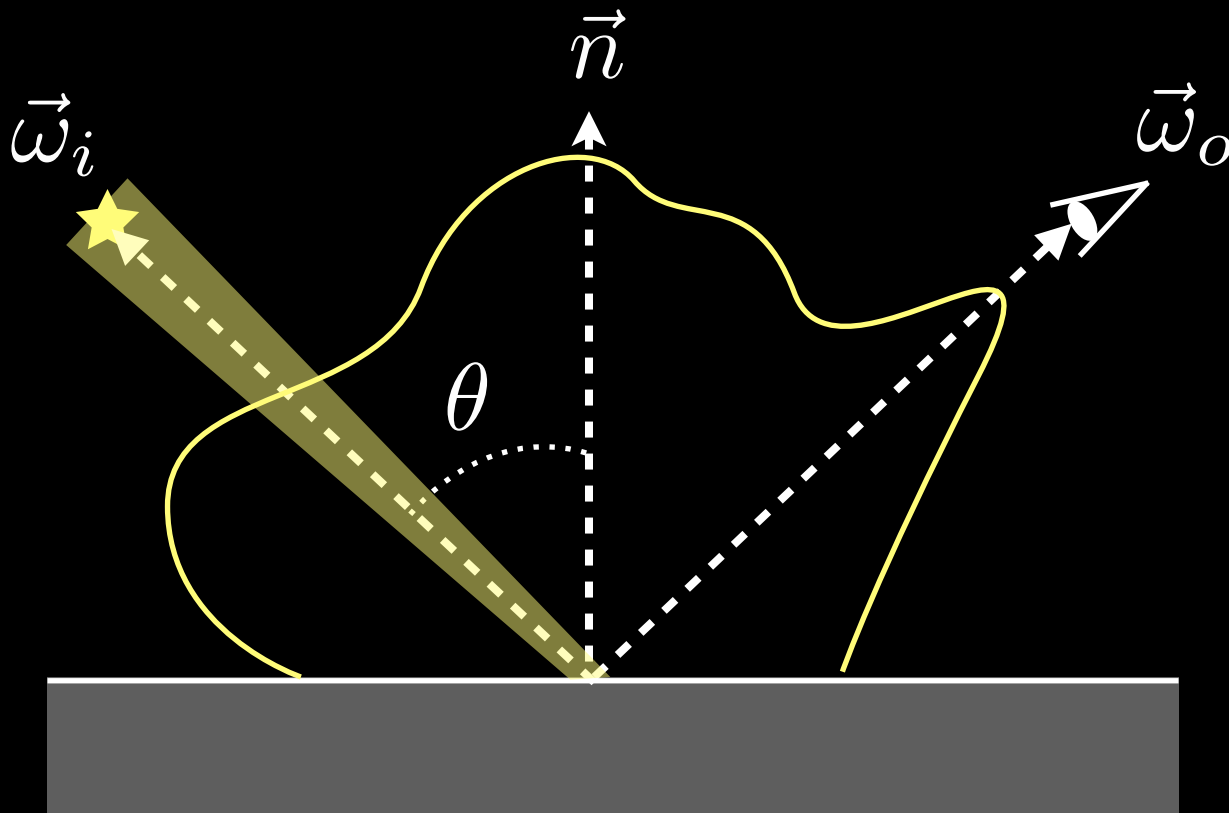 - Capture appearance of surface

BRDF

BRDF

$$f(x, \vec{\omega}_o, \vec{\omega}_i) = \frac{dL(x, \vec{\omega}_o)}{dE(x, \vec{\omega}_i)}$$

$$dE(x, \vec{\omega}_i) = L(x, \vec{\omega}_i) \cos\theta \, d\omega_i$$
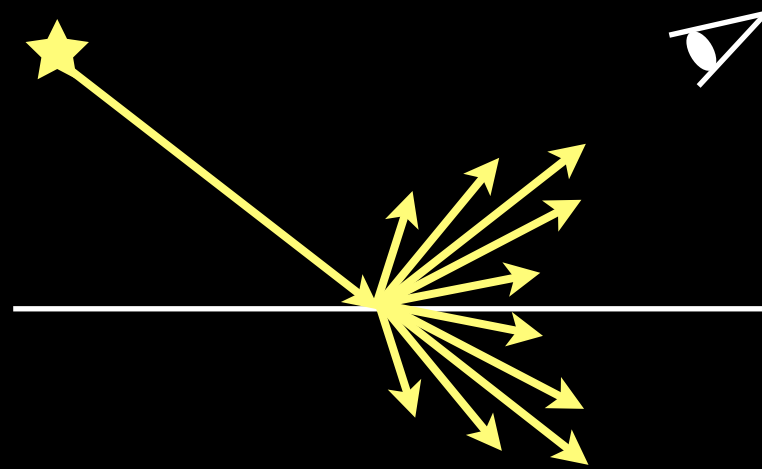
Reflected Radiance

- Sum of contributions from all directions

$$L_o(x, \vec{\omega}_o) = \int_\Omega dL_o(x, \vec{\omega}_o)$$

$$= \int_\Omega f(x, \vec{\omega}_o, \vec{\omega}_i) dE_i(x, \vec{\omega}_i)$$

$$= \int_\Omega f(x, \vec{\omega}_o, \vec{\omega}_i) L_i(x, \vec{\omega}_i) \cos\theta d\omega_i$$

BSDF

- Bidirectional Scattering Distribution Function
  - Generalization of BRDF to transparency
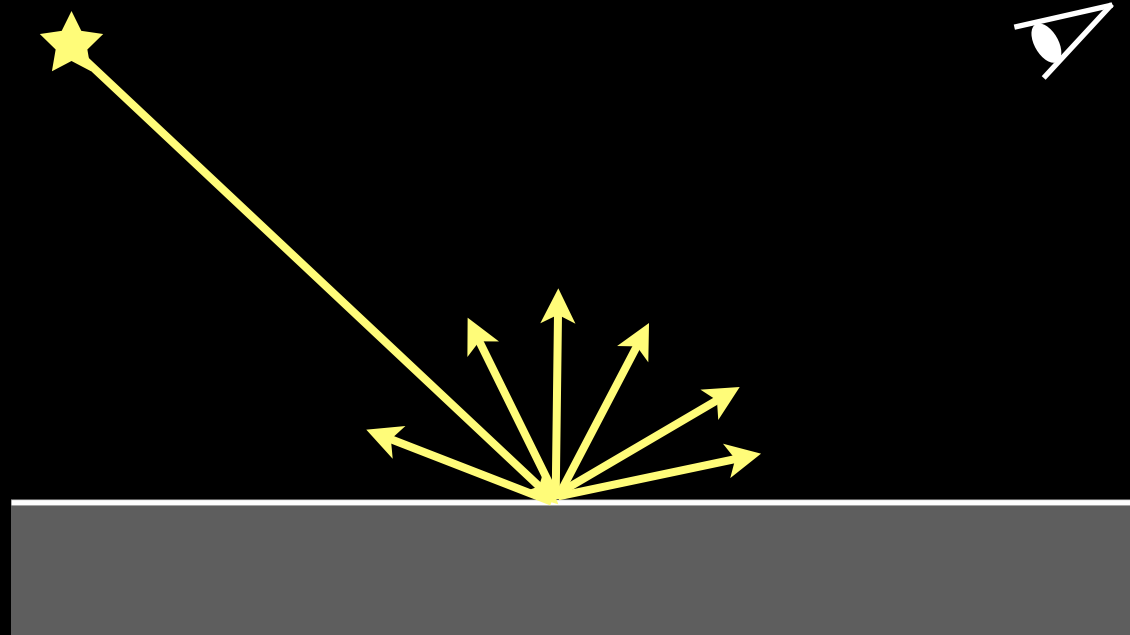 - Defined over all the directions

Defining BRDF

- Two approaches

    - Measurement

    - Analytical models

- We learn simple analytical models

Lambertian

- Uniformly reflects light over all directions
- "Matte" surfaces

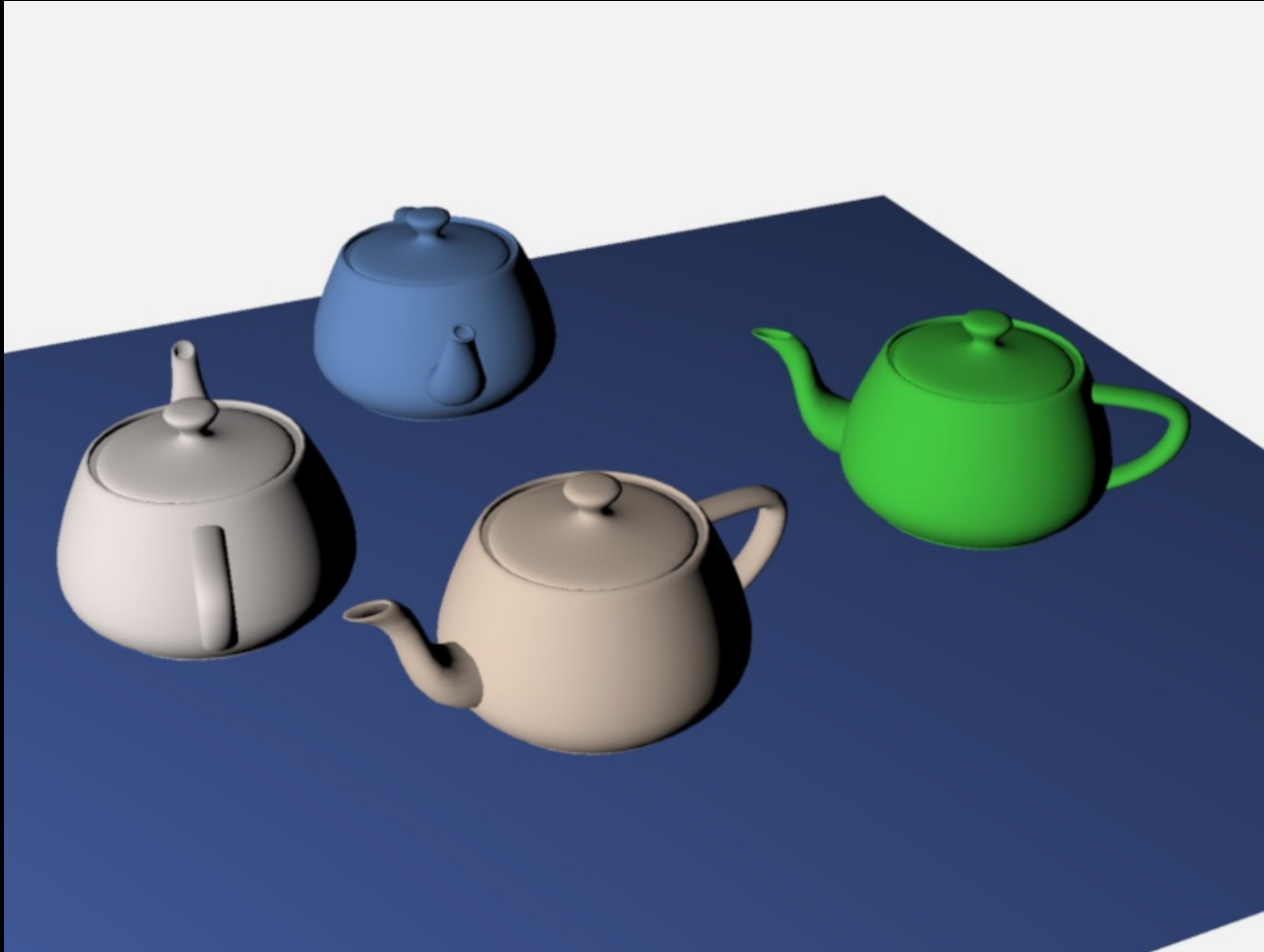$$f(\vec{\omega}_o, \vec{\omega}_i) = \frac{K_d}{\pi}$$

Lambertian

- Irradiance times BRDF = Lambertain shading

$$L_o(x, \vec{\omega}_o) = \int_\Omega dL_o(x, \vec{\omega}_o)$$

$$= \int_\Omega f(x, \vec{\omega}_o, \vec{\omega}_i) dE_i(x, \vec{\omega}_i)$$

$$= \frac{K_d}{\pi} \int_\Omega dE_i(x, \vec{\omega}_i) = \frac{K_d}{\pi} E_i$$

```
color shade (hit) {
    return (Kd / PI) * get_irradiance(hit)
}
```
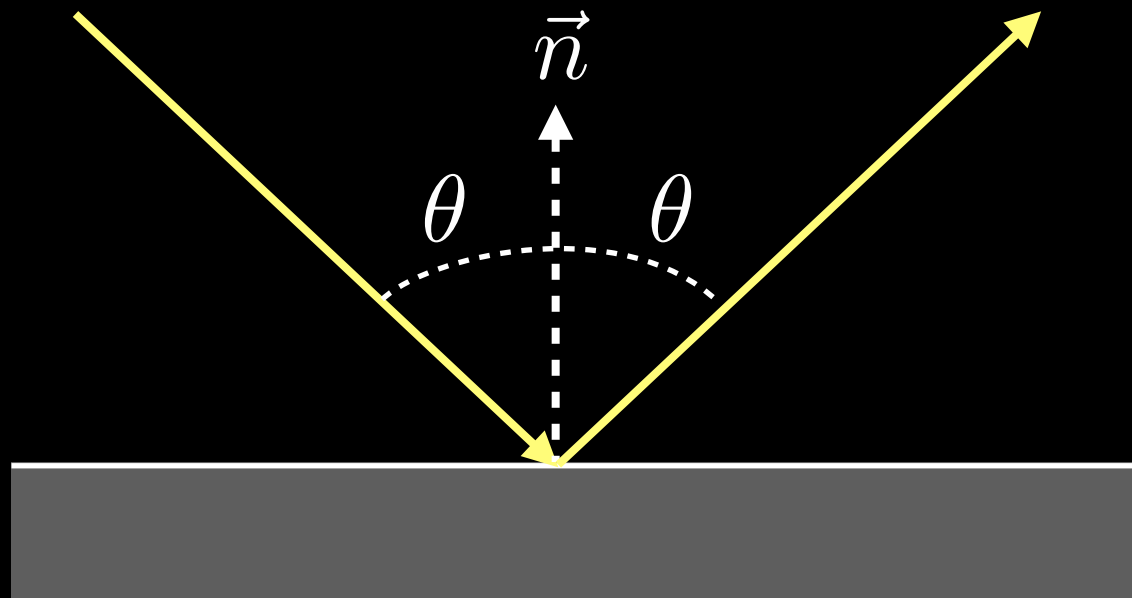
Lambertian - Example

Specular Reflection

- Mirror reflection

$$f(\vec{\omega}_o, \vec{\omega}_i) = \frac{K_s \delta(\vec{\omega}_o, \vec{\omega}_r)}{\cos\theta}$$

Specular Reflection

- Recursive tracing toward the reflected dir.

$$\vec{\omega}_r = -2(\vec{\omega}_i \cdot \vec{n})\vec{n} + \vec{\omega}_i$$

```
color shade (hit) {
  case hit.material
  diffuse:
    return (Kd / PI) * get_irradiance(hit)
  mirror:
    return Ks * shade(trace(reflected_ray))
}
```
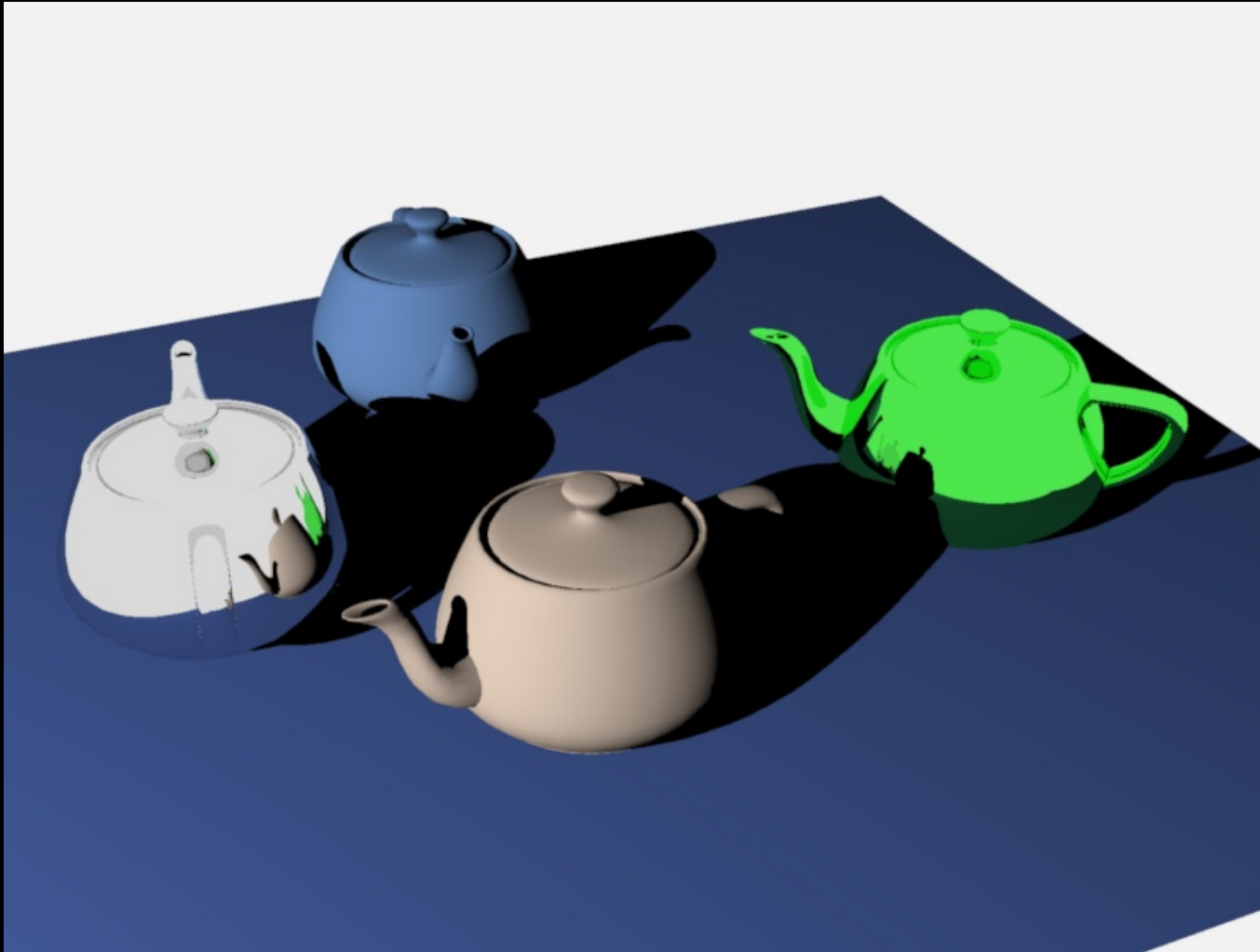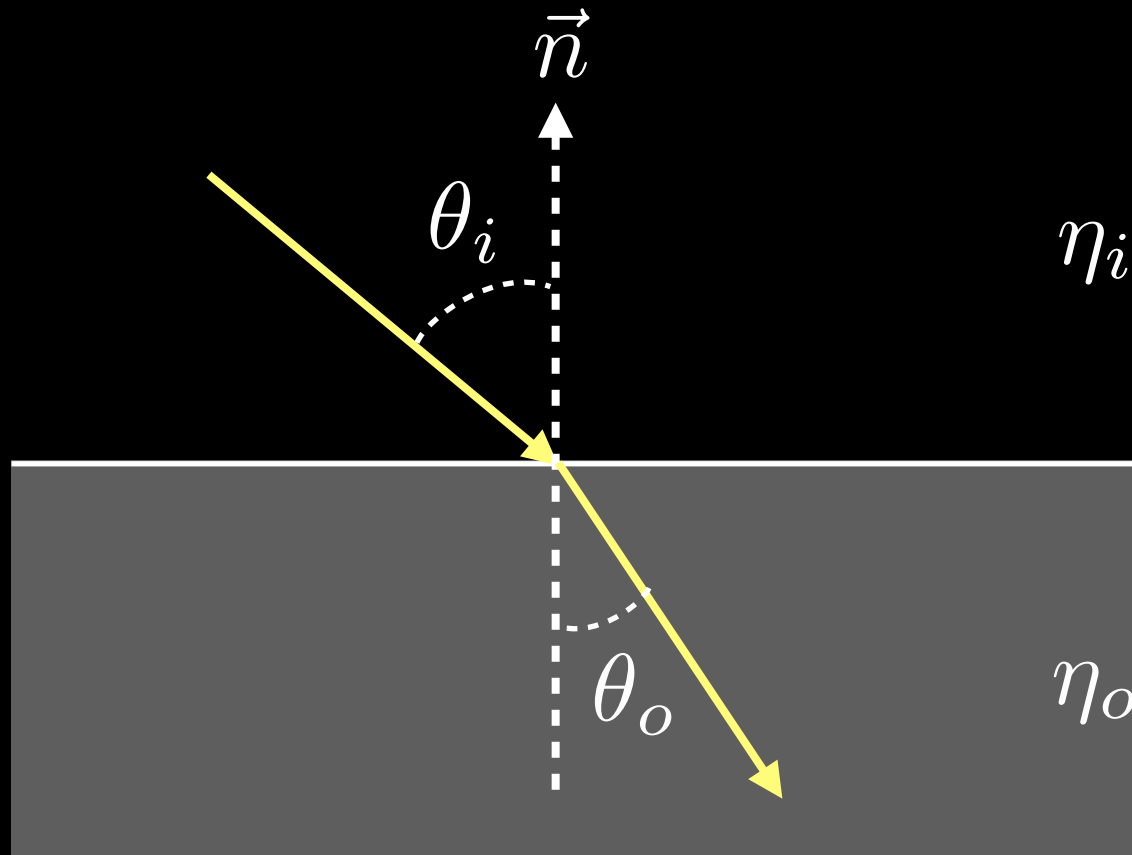
Specular Reflection - Example

Specular Refraction

- **Glass etc.** $f(\vec{\omega}_o, \vec{\omega}_i) = \dfrac{K_t \delta(\vec{\omega}_o, \vec{\omega}_t)}{\cos \theta_i}$
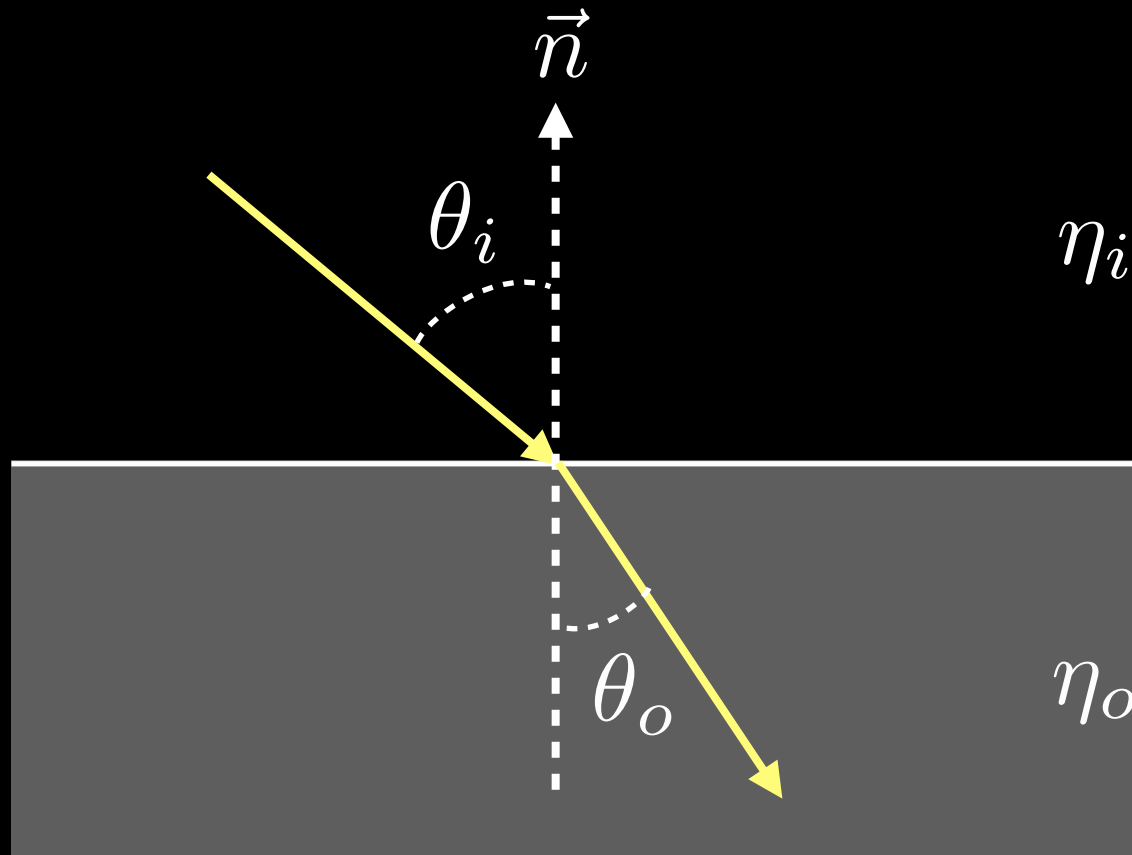
Specular Refraction

- Same as reflection: recursive tracing

```
color shade (hit) {
  case hit.material
  diffuse:
    return (Kd / PI) * get_irradiance(hit)
  mirror:
    return Ks * shade(trace(reflected_ray))
  glass:
    return Kt * shade(trace(refracted_ray))
}
```

Snell's Law

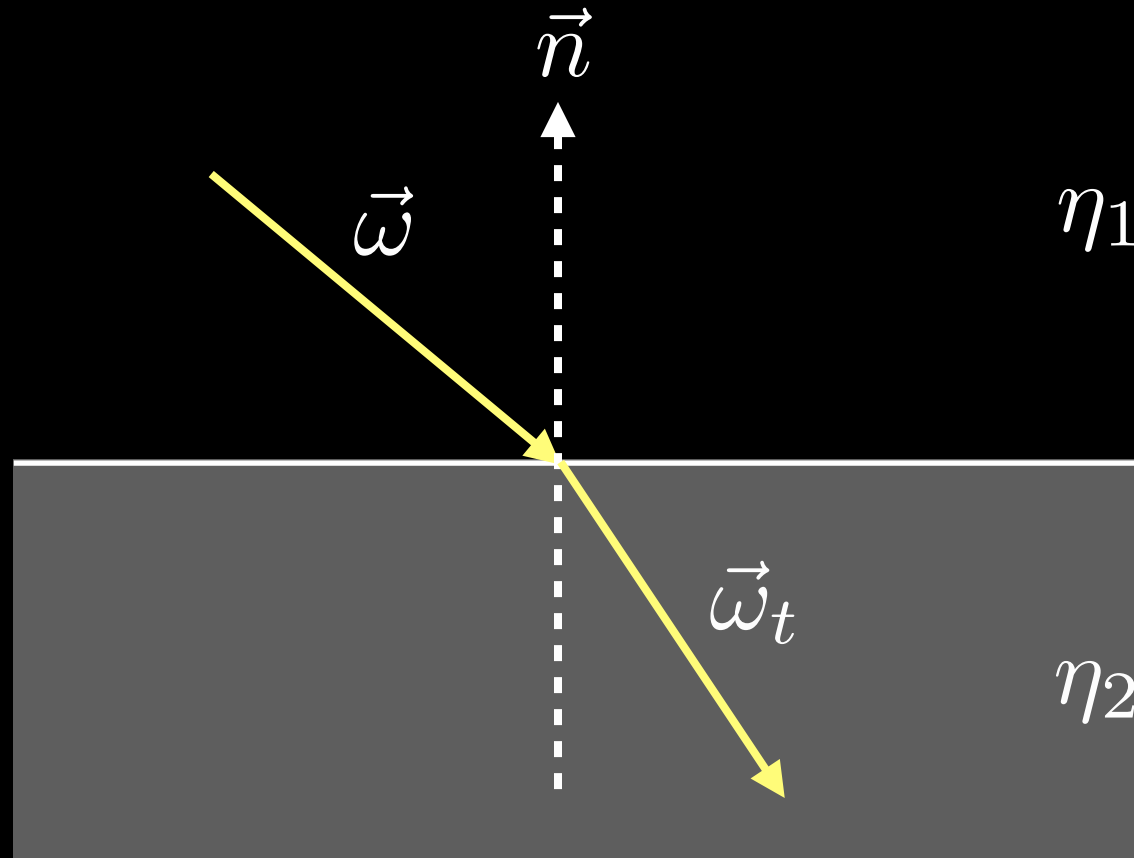$$\eta_i \sin \theta_i = \eta_o \sin \theta_o$$

Index of Refraction

- Some values of η

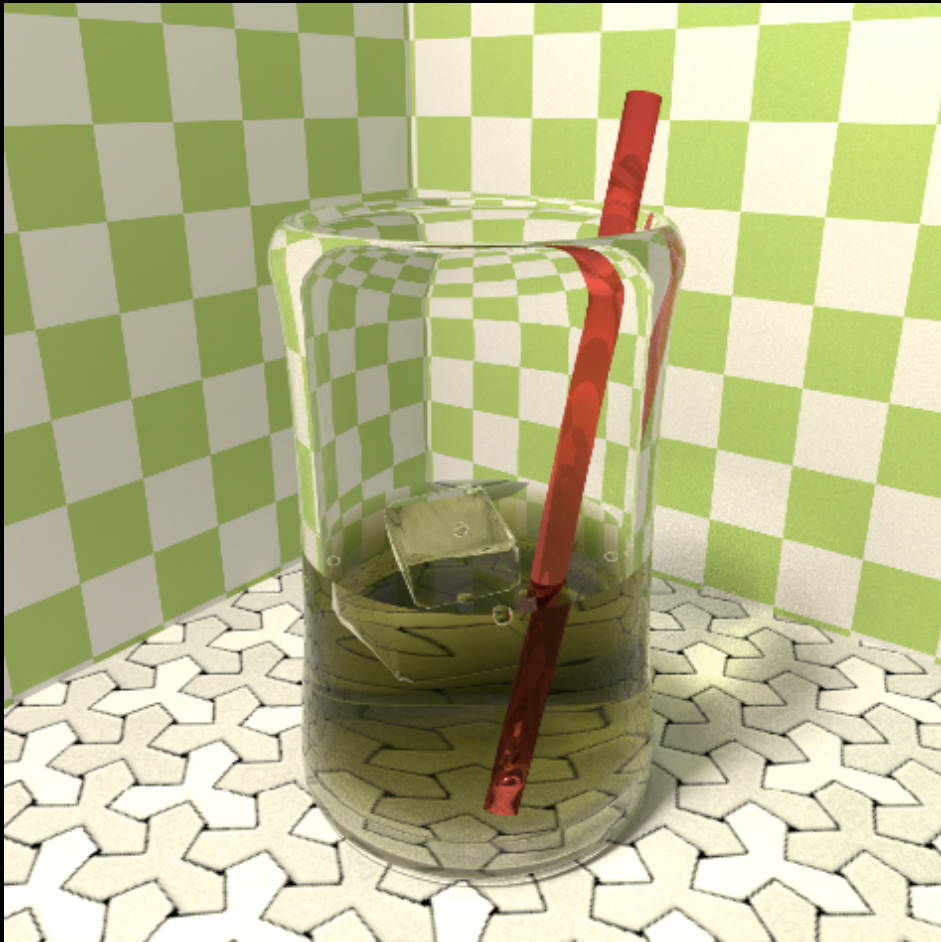|	
Vacuum	1.0
Air	1.00029
Ice	1.31
Water	1.33
Crown glass	1.52 - 1.65
Diamond	2.417

Refracted Direction

$$\vec{\omega}_t = \frac{\eta_1}{\eta_2}(\vec{\omega} - (\vec{\omega} \cdot \vec{n})\vec{n}) - \left(\sqrt{1 - \left(\frac{\eta_1}{\eta_2}\right)^2 (1 - (\vec{\omega} \cdot \vec{n})^2)} \right) \vec{n}$$

Refracted Direction

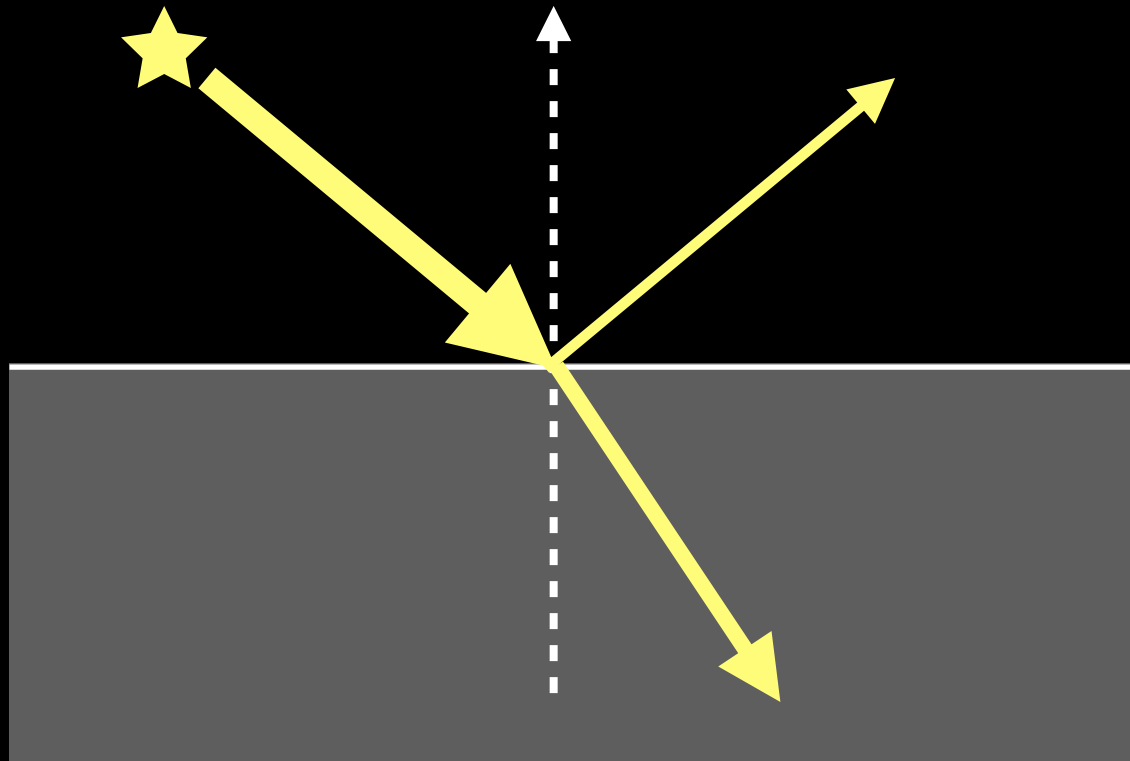Be aware of what object ray is entering and existing!

Stack-based solution

"Simple Nested Dielectrics in Ray Traced Images" [Schmidt 2002]

Fresnel Reflection

- Both reflection and refraction occur at interface

Fresnel Reflection

- Based on Fresnel equations

$$\rho_s = \frac{\eta_1 \cos\theta_i - \eta_2 \cos\theta_o}{\eta_1 \cos\theta_i + \eta_2 \cos\theta_o}$$

$$\rho_t = \frac{\eta_1 \cos\theta_o - \eta_2 \cos\theta_i}{\eta_1 \cos\theta_o + \eta_2 \cos\theta_i}$$

$$F(\theta_o, \theta_i) = \frac{1}{2}\left(\rho_s^2 + \rho_t^2\right)$$
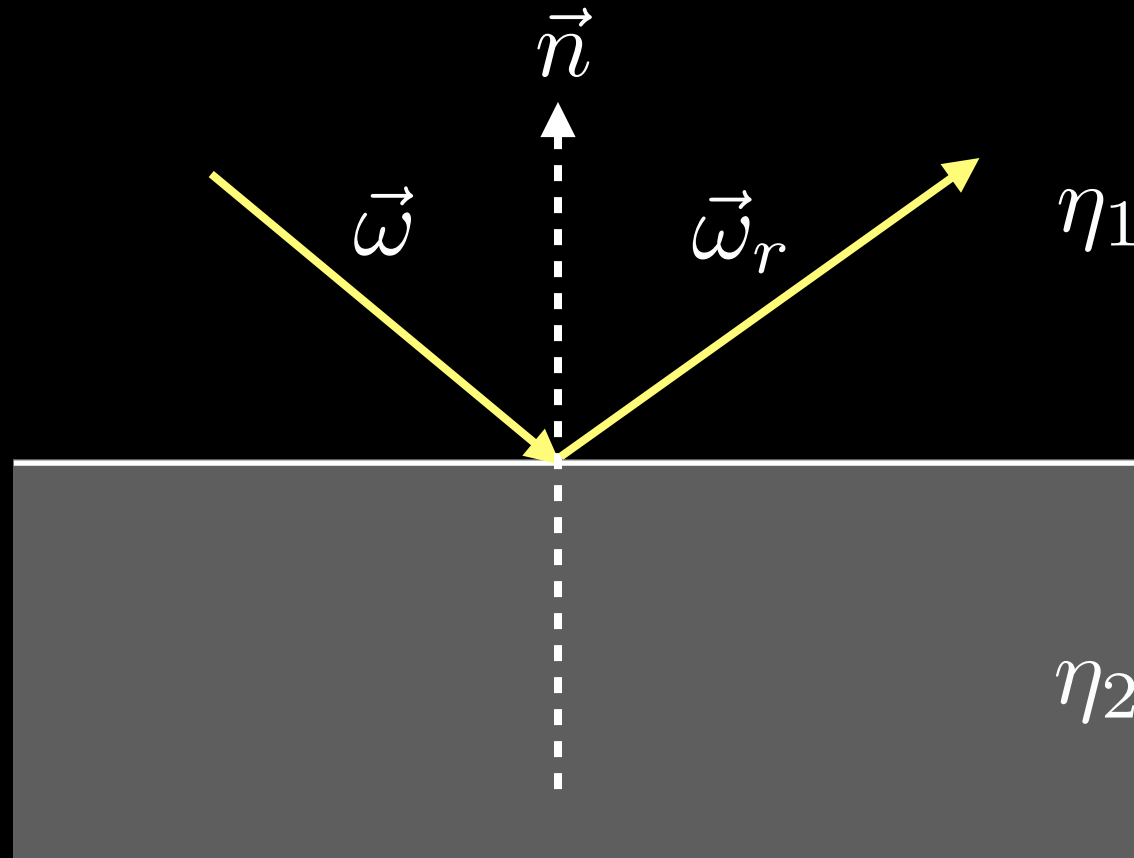
Fresnel Reflection

- Recursive call with branch

```
glass:

  R = Fresnel(hit, ray);

  return R * shade(trace(reflected_ray))
  + (1 - R) * shade(trace(refracted_ray));
```

Total Internal Reflection

$\vec{n}$

$\vec{\omega}$ $\vec{\omega}_r$ η_1

cf. optical fiber

θ_2 $\vec{\omega}_r$

η_2

Trace only reflected ray with K_s = 1 if

$$1 - \left(\frac{\eta_1}{\eta_2}\right)^2 (1 - (\vec{\omega} \cdot \vec{n})^2) < 0 \quad \text{(inside of the sqrt)}$$

Complete Glass

- Also known as dielectric materials

- Be careful about ray and normal directions

```
glass:
  if (TotalInternal) return shade(trace(reflected_ray))

  R = Fresnel(hit, ray);

  return R * shade(trace(reflected_ray))
       + (1 - R) * shade(trace(refracted_ray));
```

Microfacets

- Physically accurate modeling of rough surfaces

  - Lots of tiny mirrors at random orientations

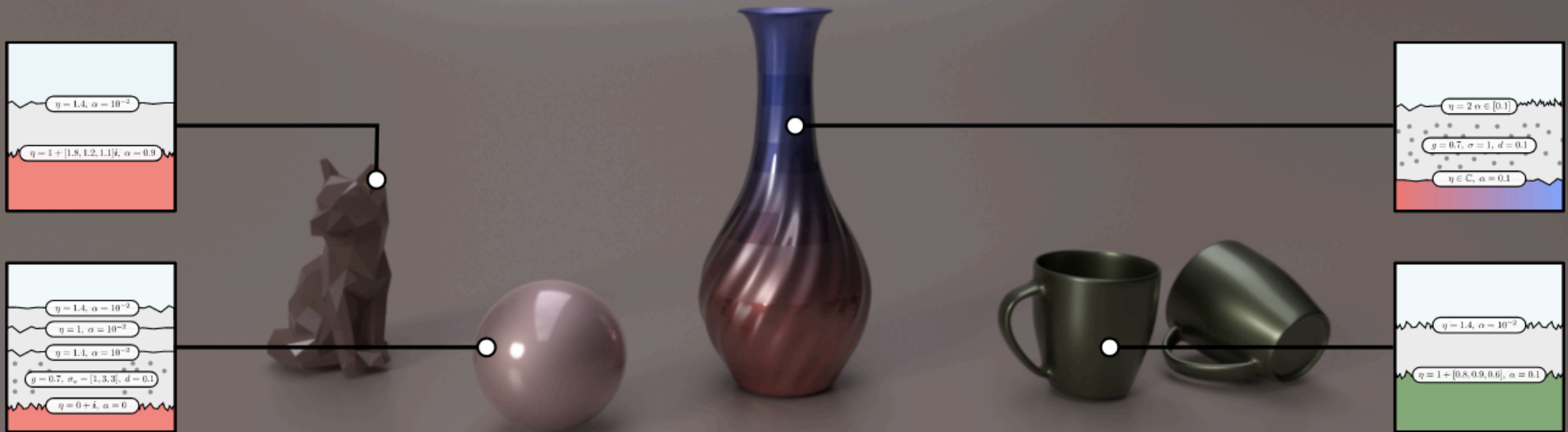 - Distribution of orientations of microfacets decides the sharpness of reflection

  - e.g., Cook-Torrance model

Specular ⟶ Glossy
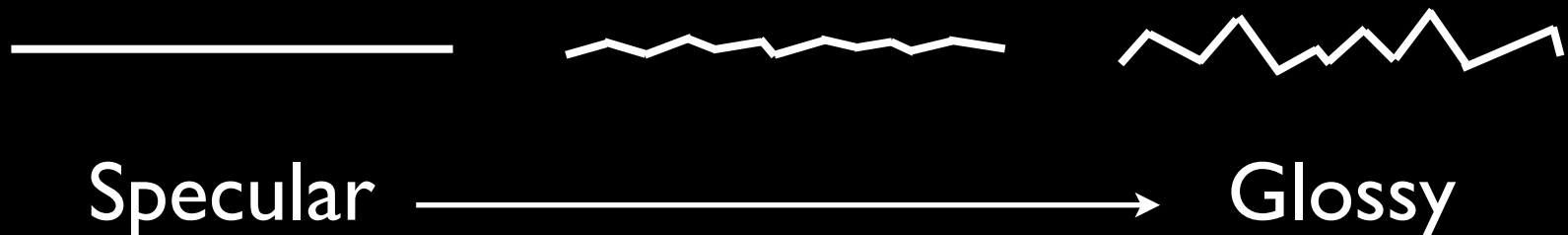
Isotropic and Anisotropic

https://google.github.io/filament/Materials.html#materialmodels/litmodel/anisotropy
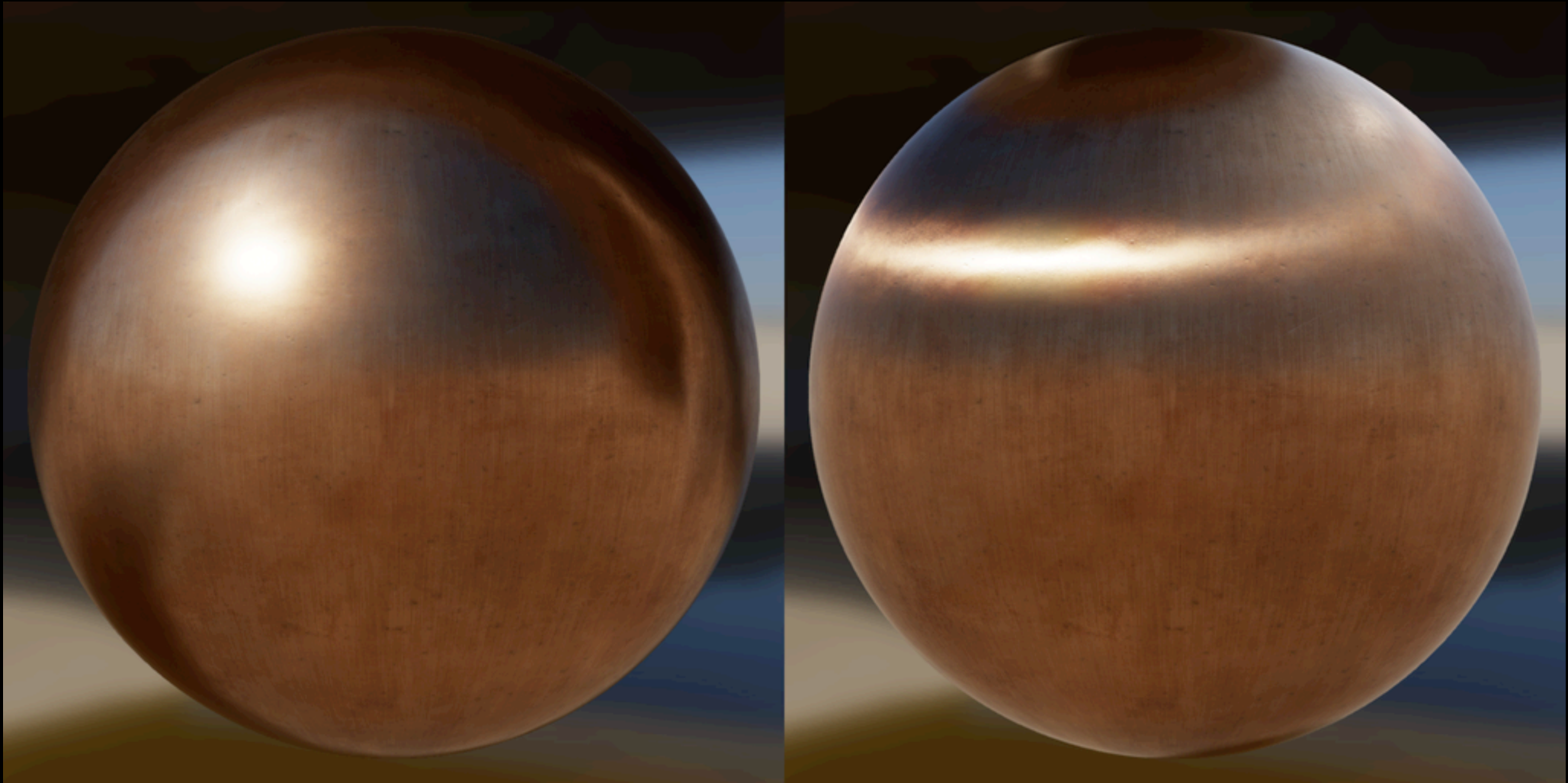
Layered Materials

- Transparent coating by combining BSDFs

"Efficient Rendering of Layered Materials using an Atomic Decomposition with Statistical Operators" [Belcour 2018]
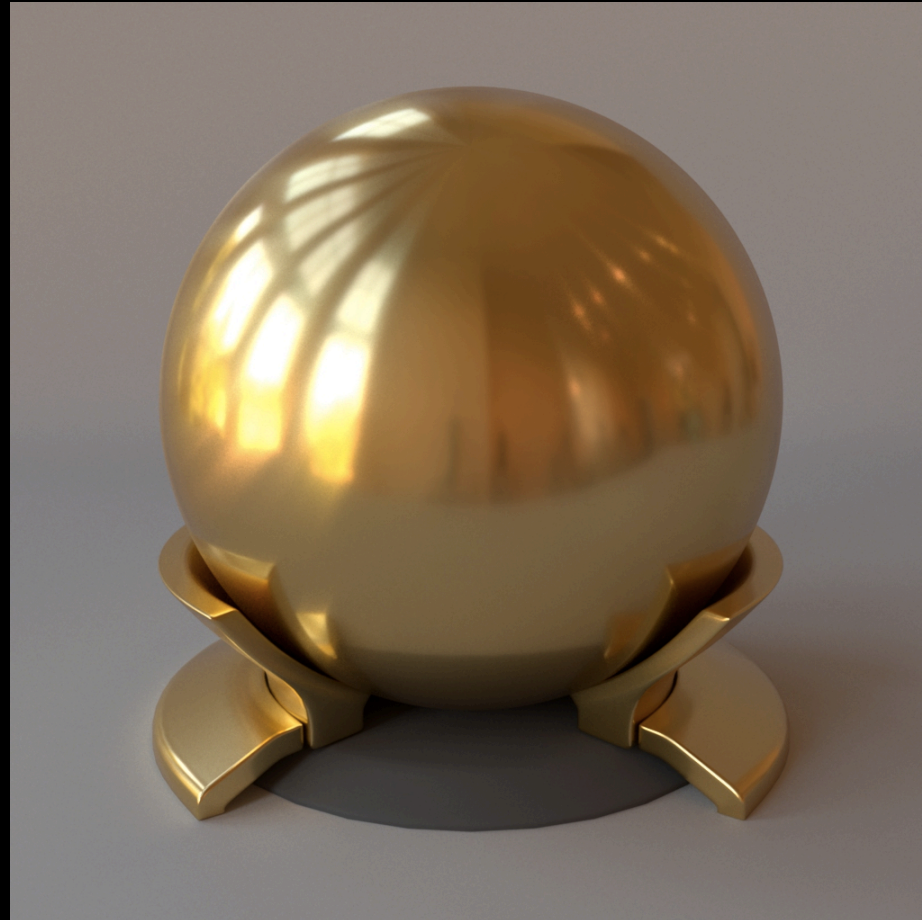
Disney BRDF
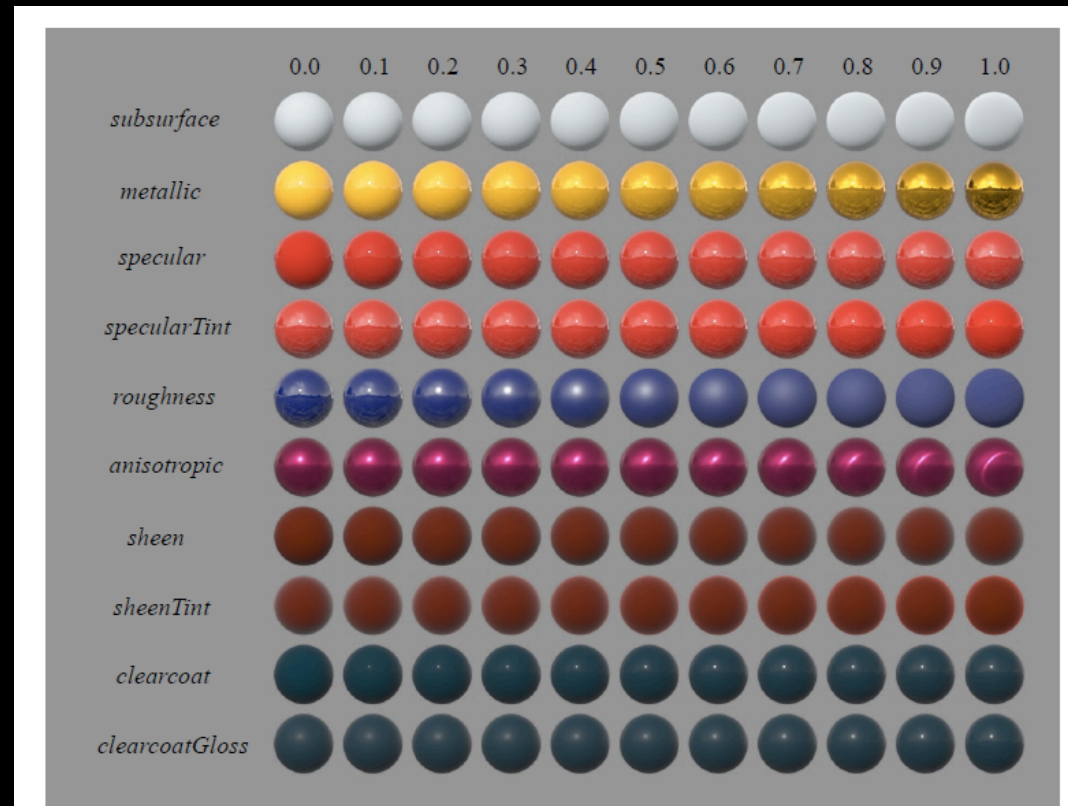
Figure 16: Examples of the effect of our BRDF parameters. Each parameter is varied across the row from zero to one with the other parameters held constant.
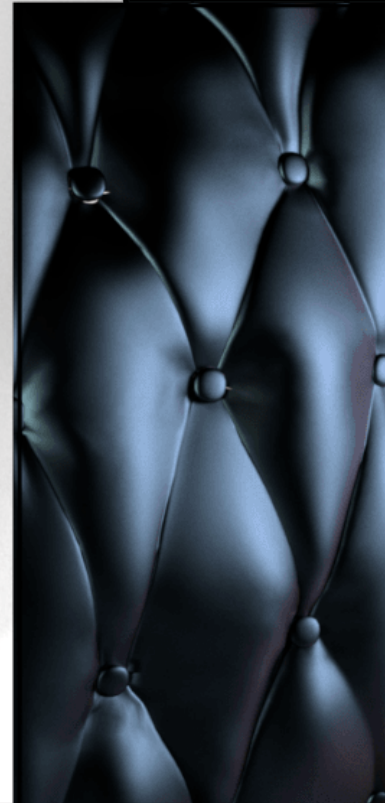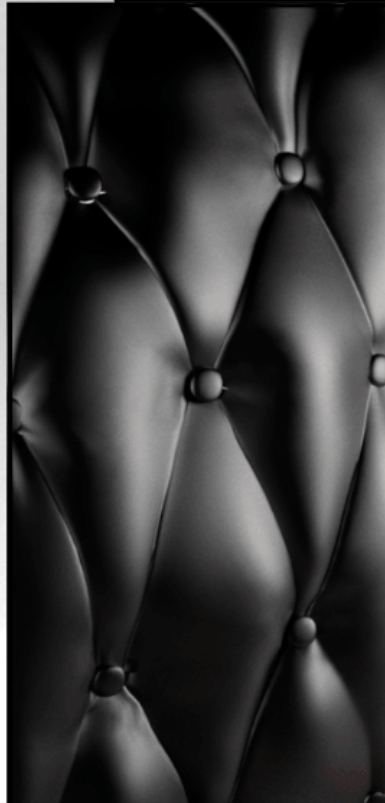
"Physically-Based Shading at Disney" [Burley 2012]

Measured BRDFs

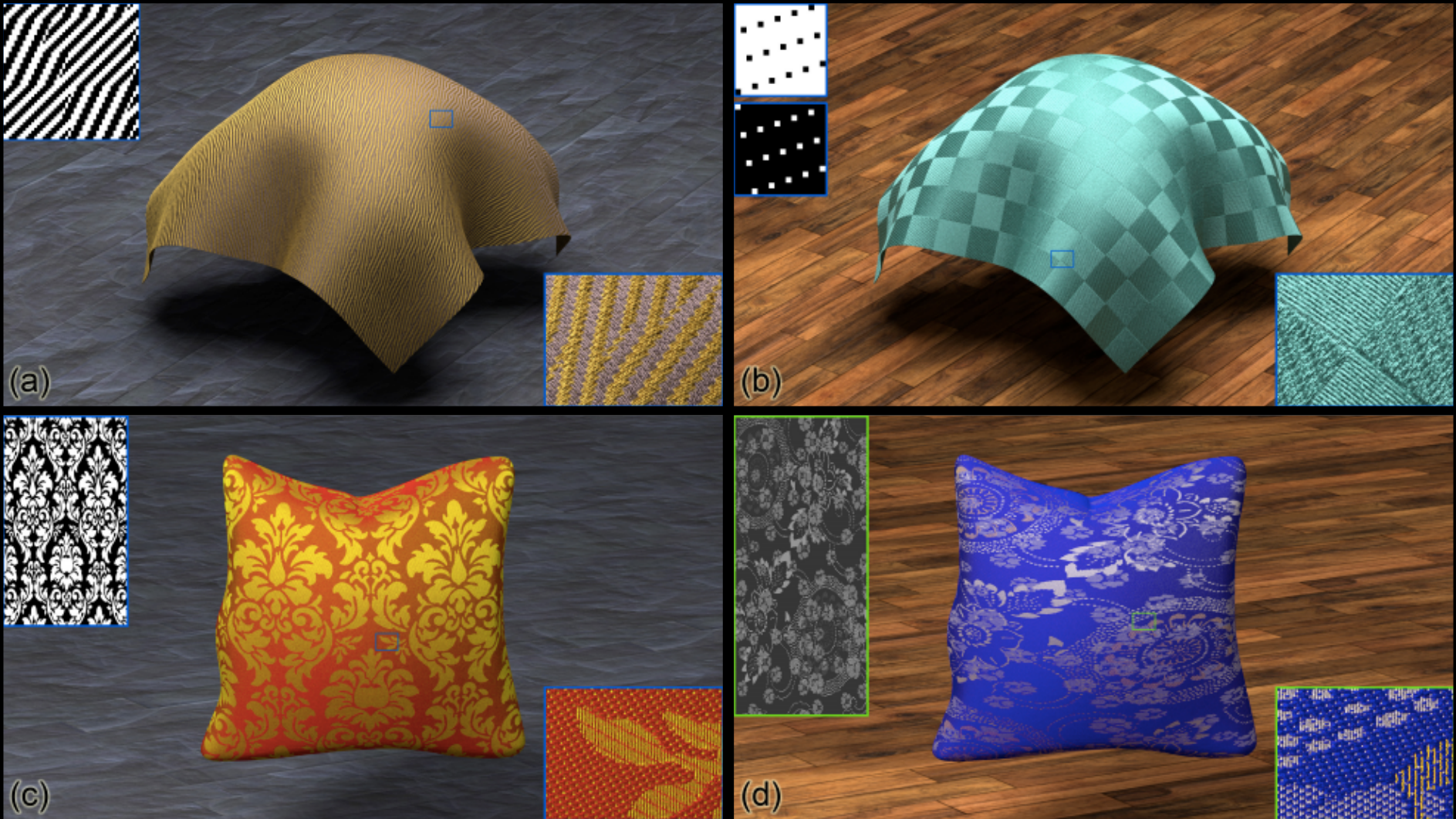Material Database at EPFL (http://rgl.epfl.ch/materials)

Iridescent Microfacets

"A Practical Extension to Microfacet Theory for the Modeling of Varying Iridescence" [Belcour 2017]

Specular Microgeometry

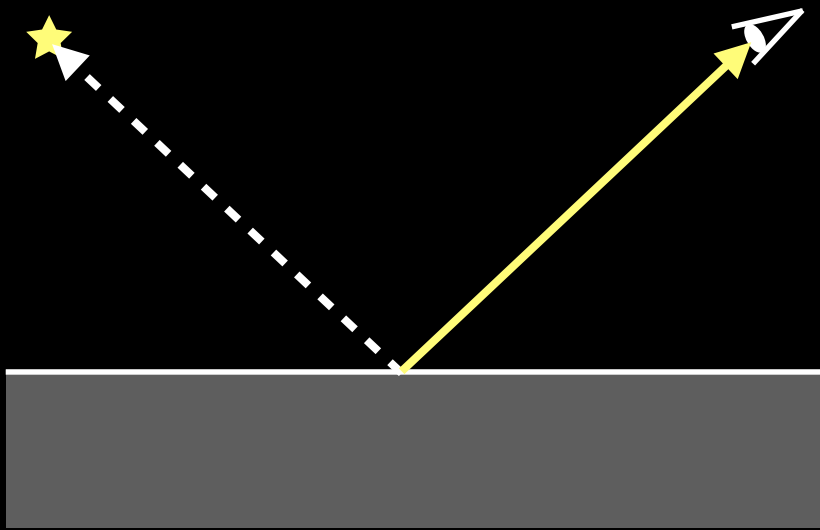"Rendering Specular Microgeometry with Wave Optics" [Yan 2018]
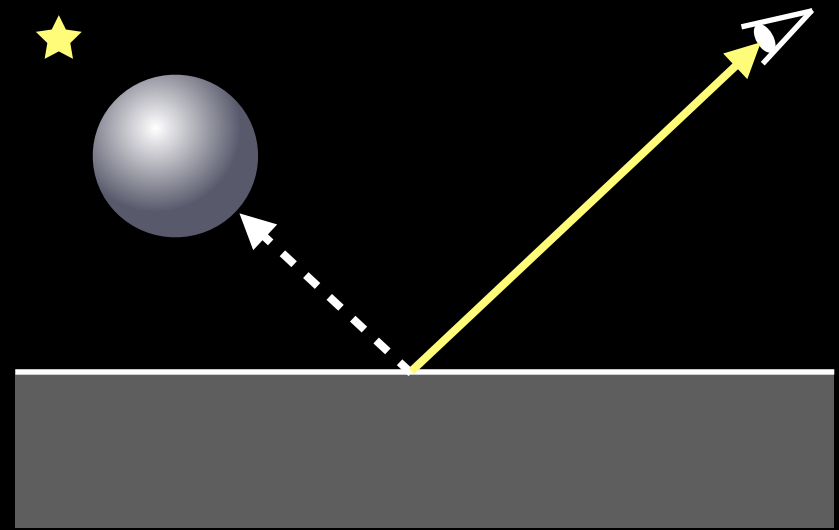
Clothes

"Structure-aware Synthesis for Predictive Woven Fabric Appearance" [Zhao 2012]

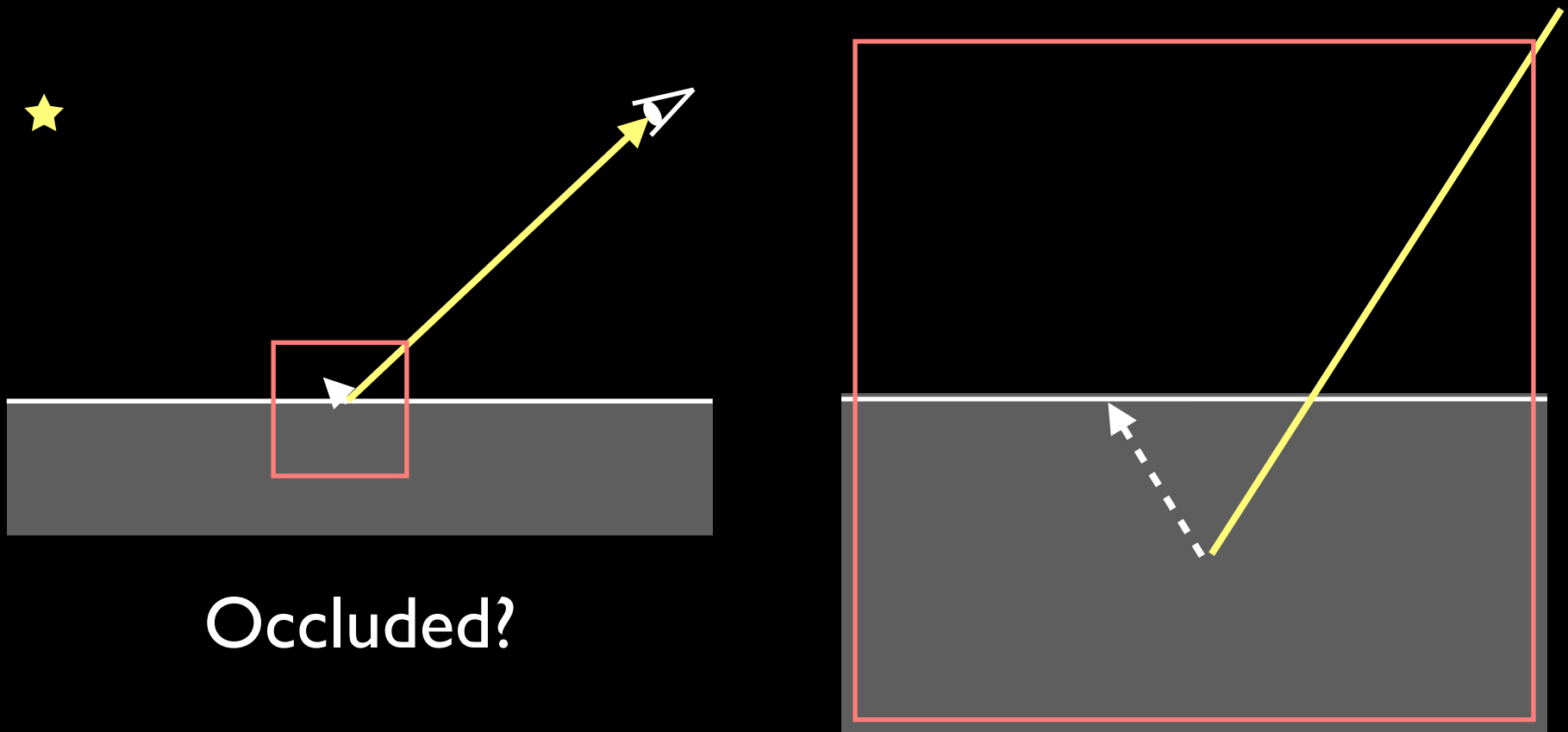# Shadows

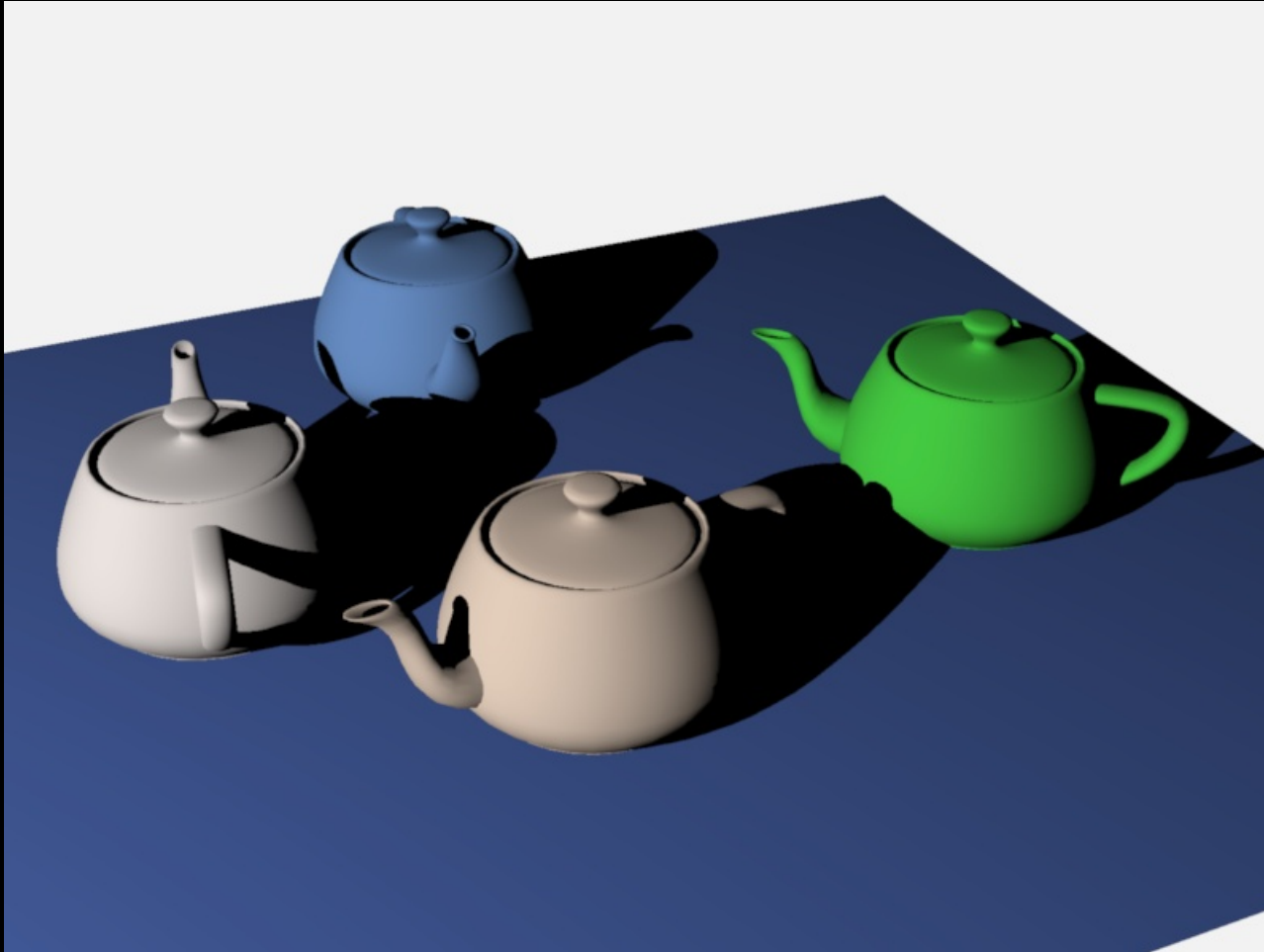- Return irradiance only if the light is visible
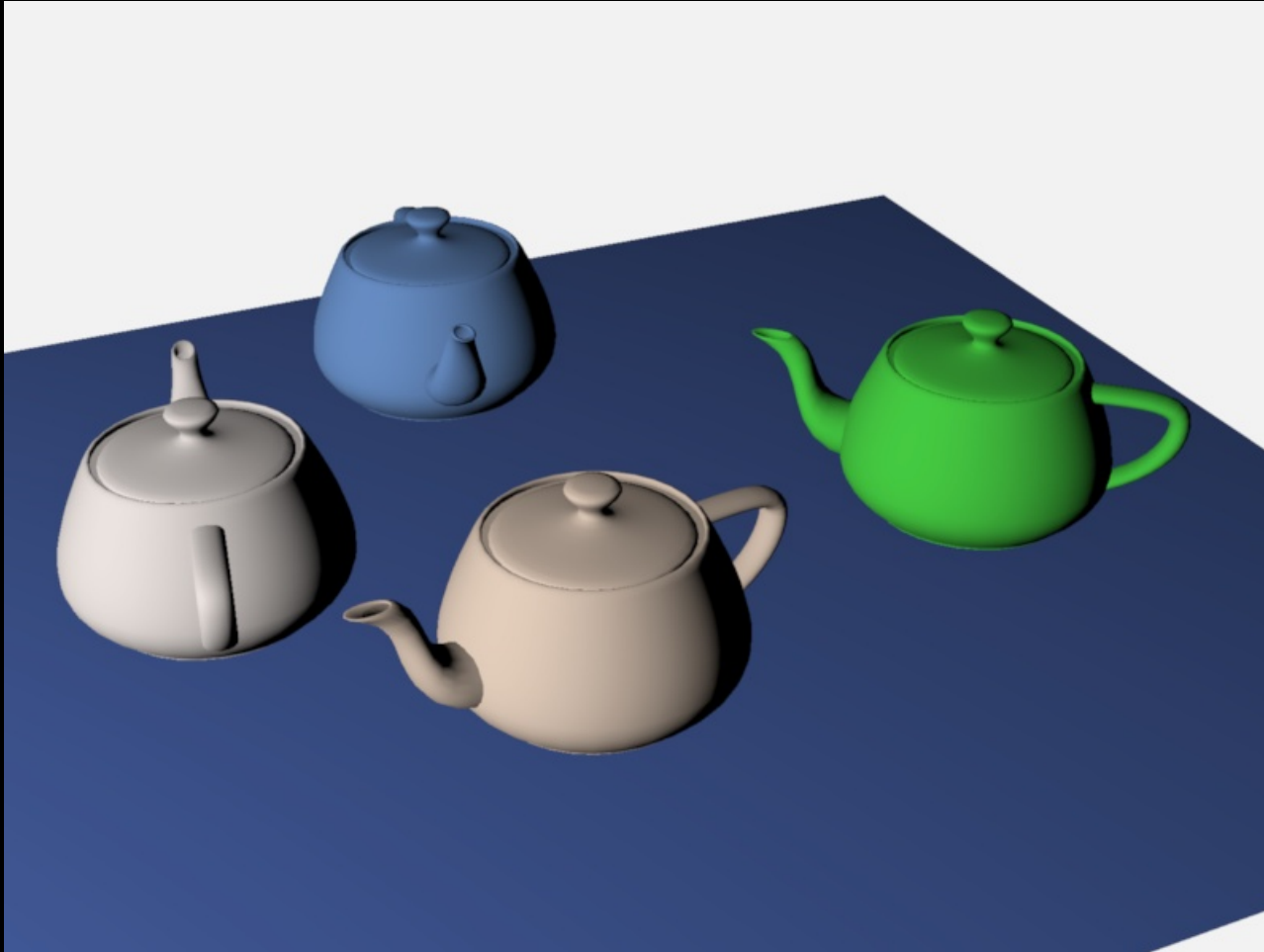

Illuminated   Occluded

Shadows

- Avoid self-intersection due to numerical error
  - Ignoring intersections that are too close etc.

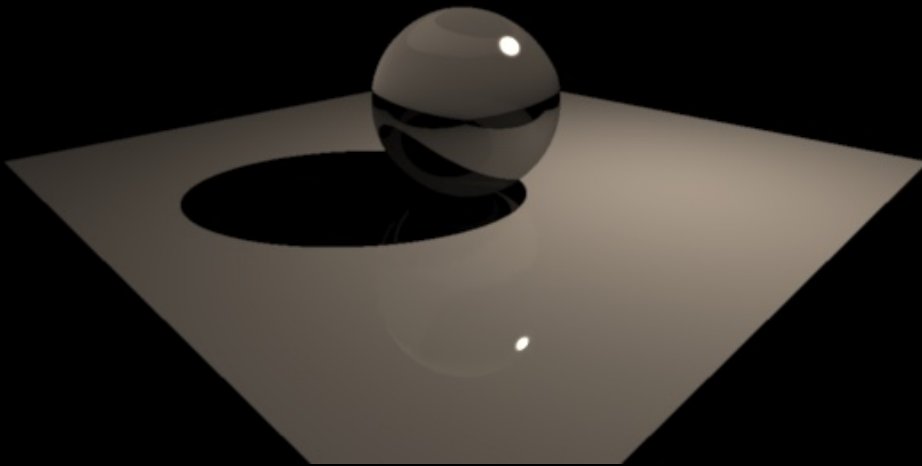Occluded?
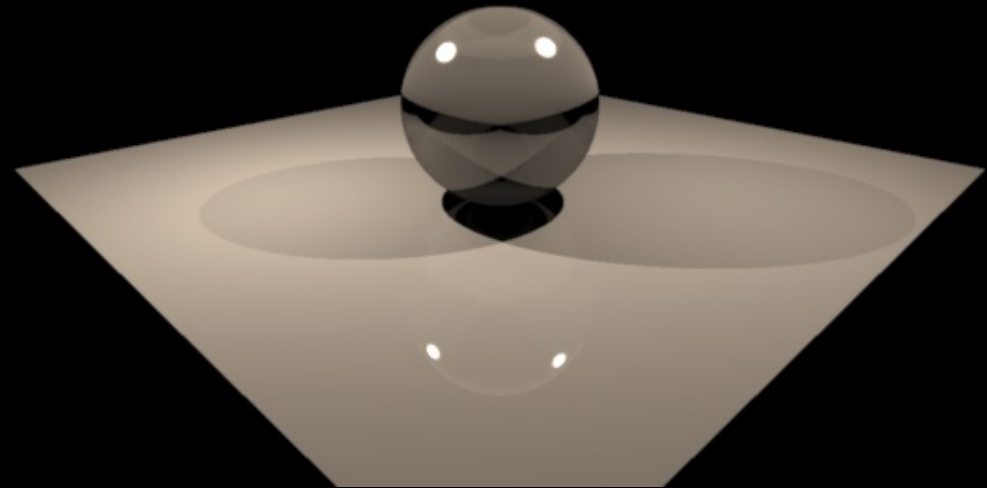
With Shadows

Without Shadows

Multiple Light Sources

- Simply add up all the contributions

- Linearity of illumination

One light source

Two light source

Image Based Lighting

- Use of pixels in the image as light sources

Input illumination

Renderings

Image Based Lighting

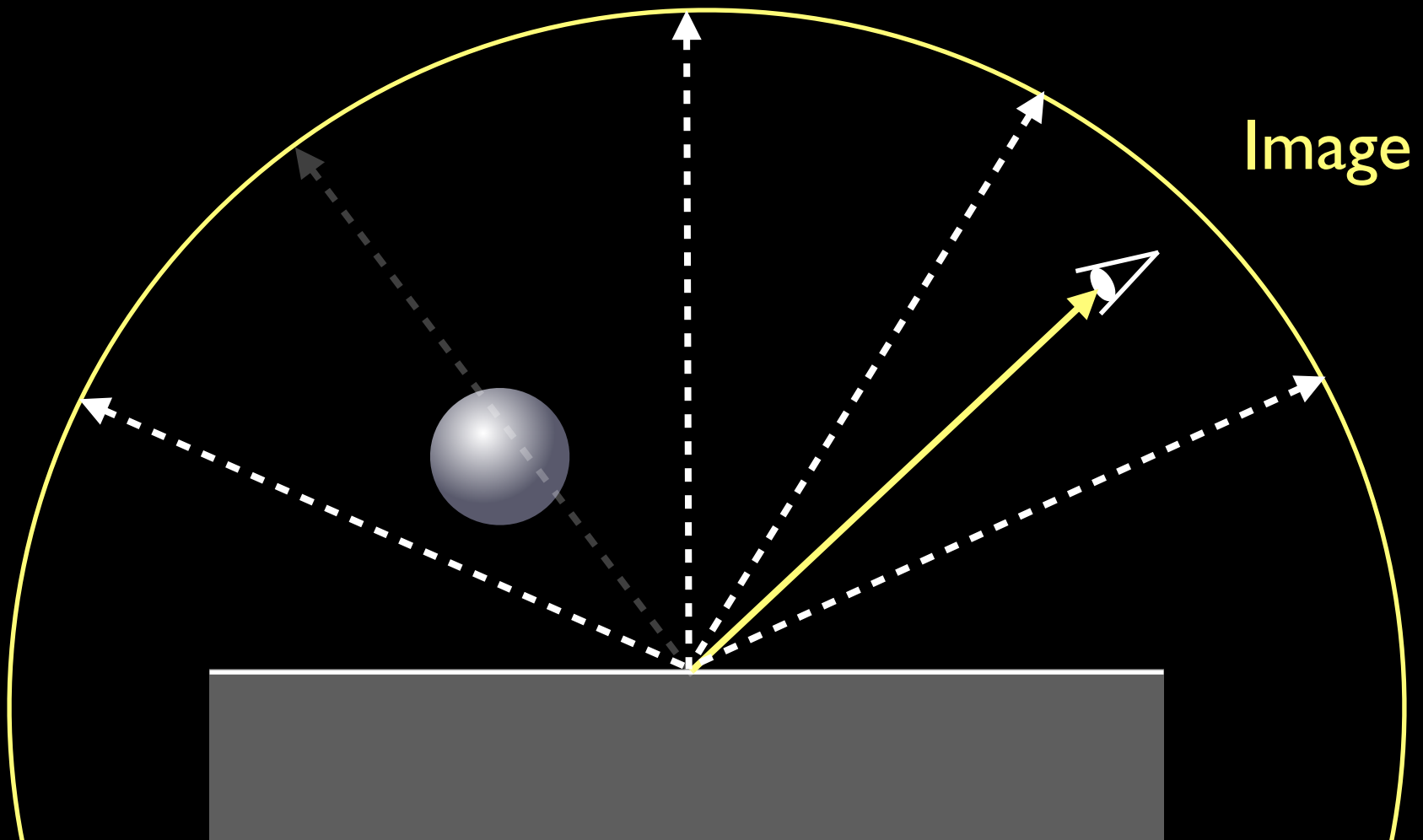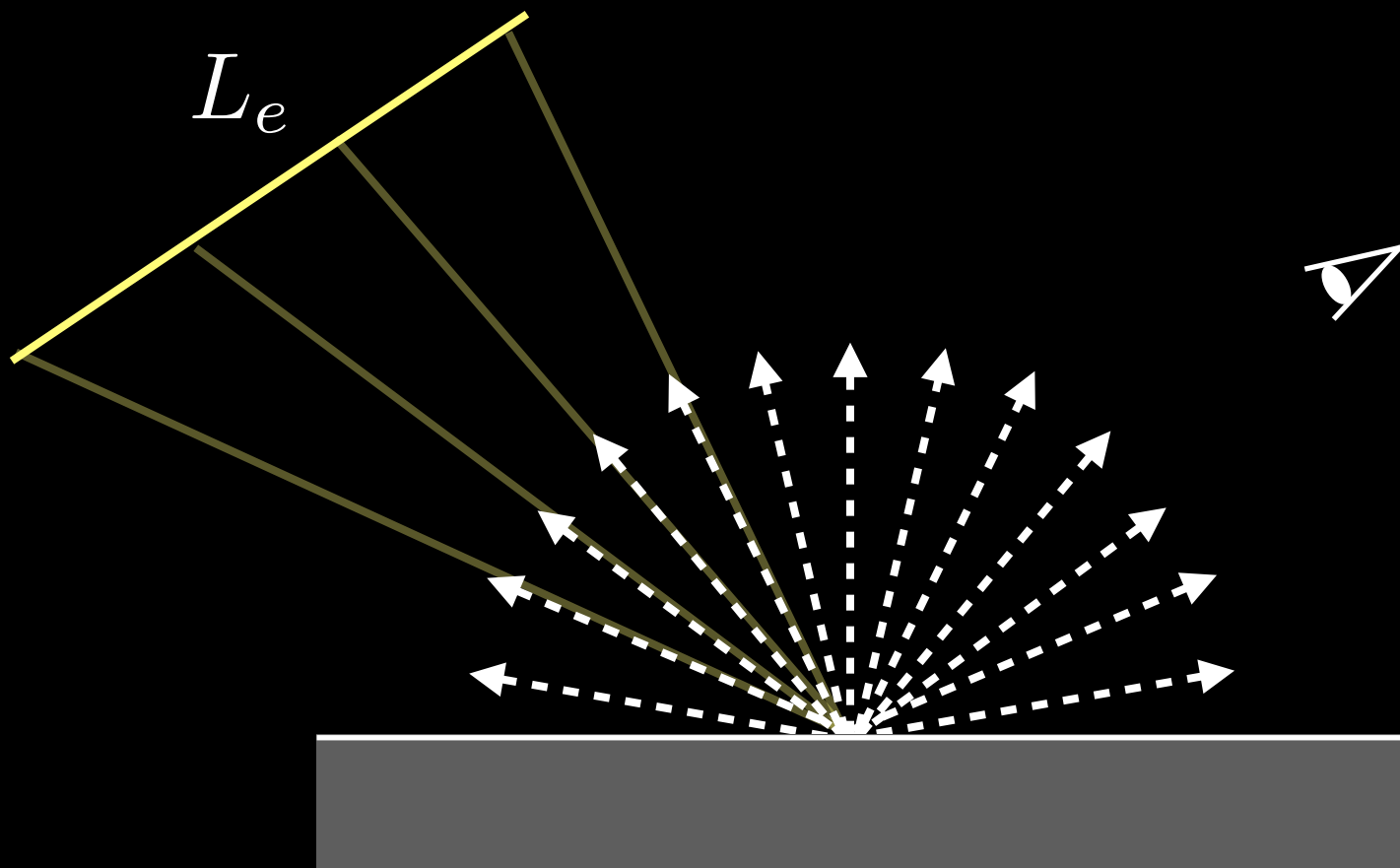- If ray hits nothing, return a value from image

Image

Image Based Lighting

- Trace random ray above the hemisphere

```
color get_irradiance (hit) {
  Ep = ... // irradiance from point light sources
  for (i = 1...N) {
    ray = gen_random_dir(hit.n)
    if (trace(ray) = no_hit) Ei = Ei + IBL(ray, image)
  }
  return Ep + (Ei / N)
}
```

Area Light Sources

$$L_o(x, \vec{\omega}_o) = \int_\Omega f(x, \vec{\omega}_o, \vec{\omega}_i) L_i(x, \vec{\omega}_i) \cos\theta_i d\omega_i$$
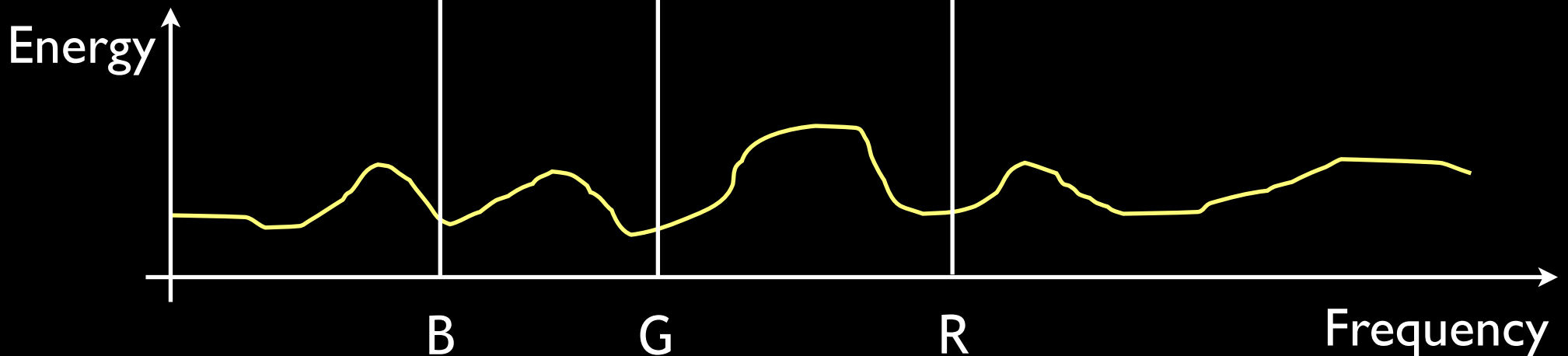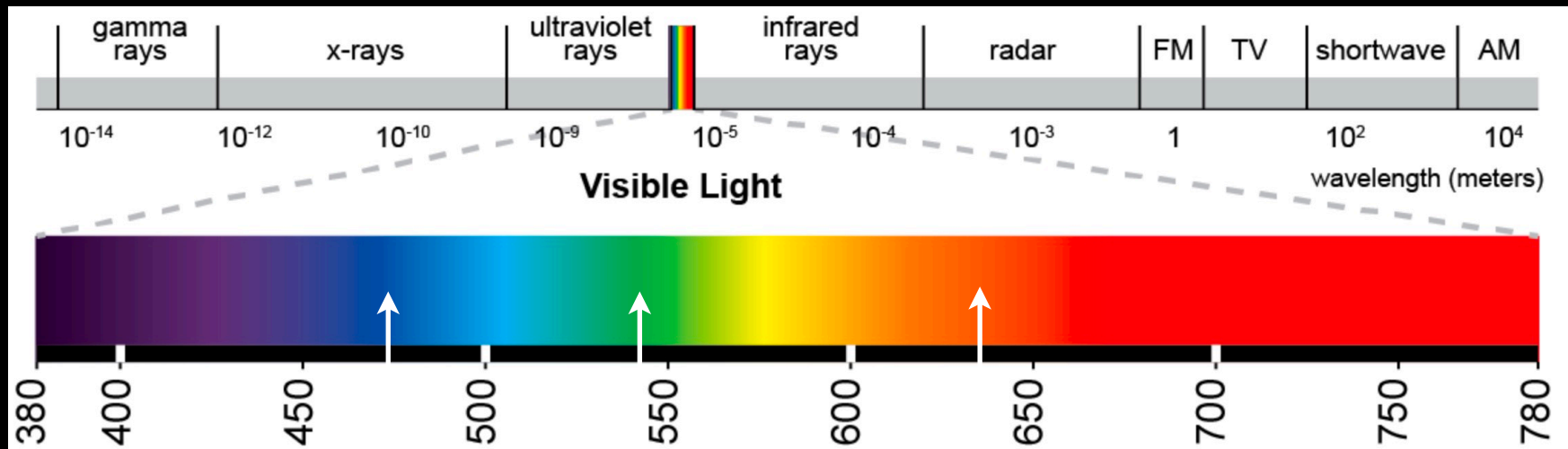

L_e

Some Details

- Generating random rays

  - http://people.cs.kuleuven.be/~philip.dutre/GI/

  - See Eqn. (34)

- Fetching pixels by rays

  - Depends on parameterization

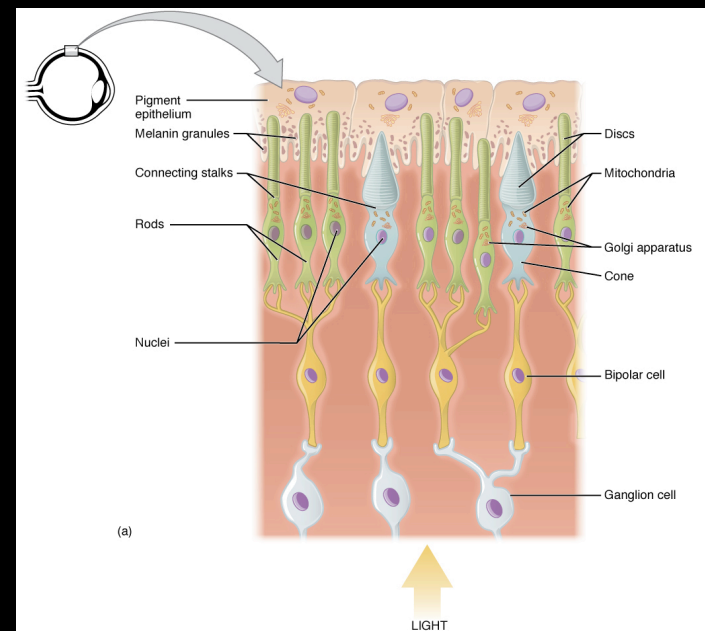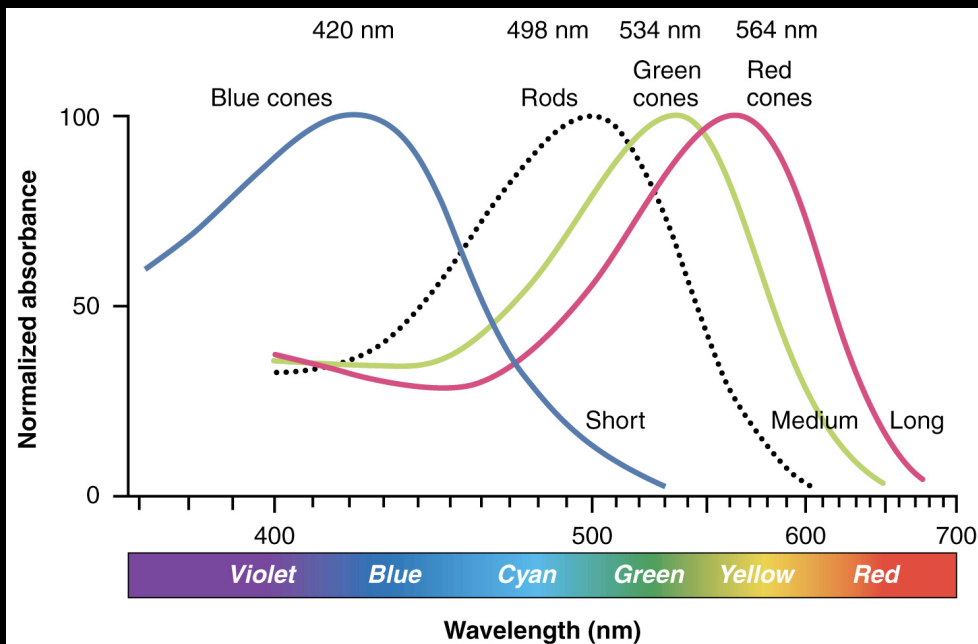 - http://www.pauldebevec.com/Probes/

Color in Computer Graphics

- Store only three values (Red, Green, Blue)

Why RGB?

- Because we see with RGB

  - Rods: Intensity sensors

  - Cones: Color sensors (typically three types)

https://en.wikipedia.org/wiki/Photoreceptor_cell

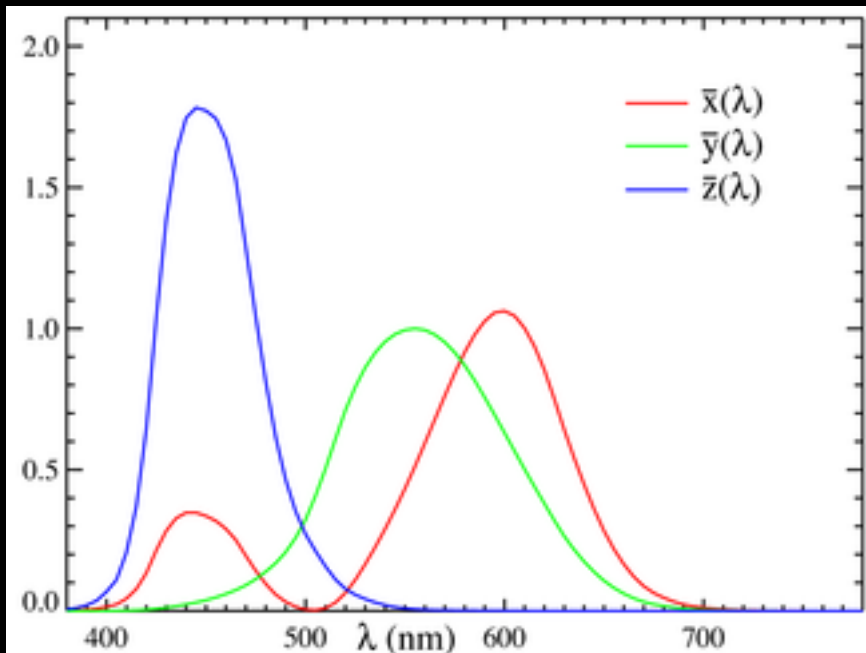# Full Spectrum Rendering

- Simulation at many wavelength ⇒ RGB
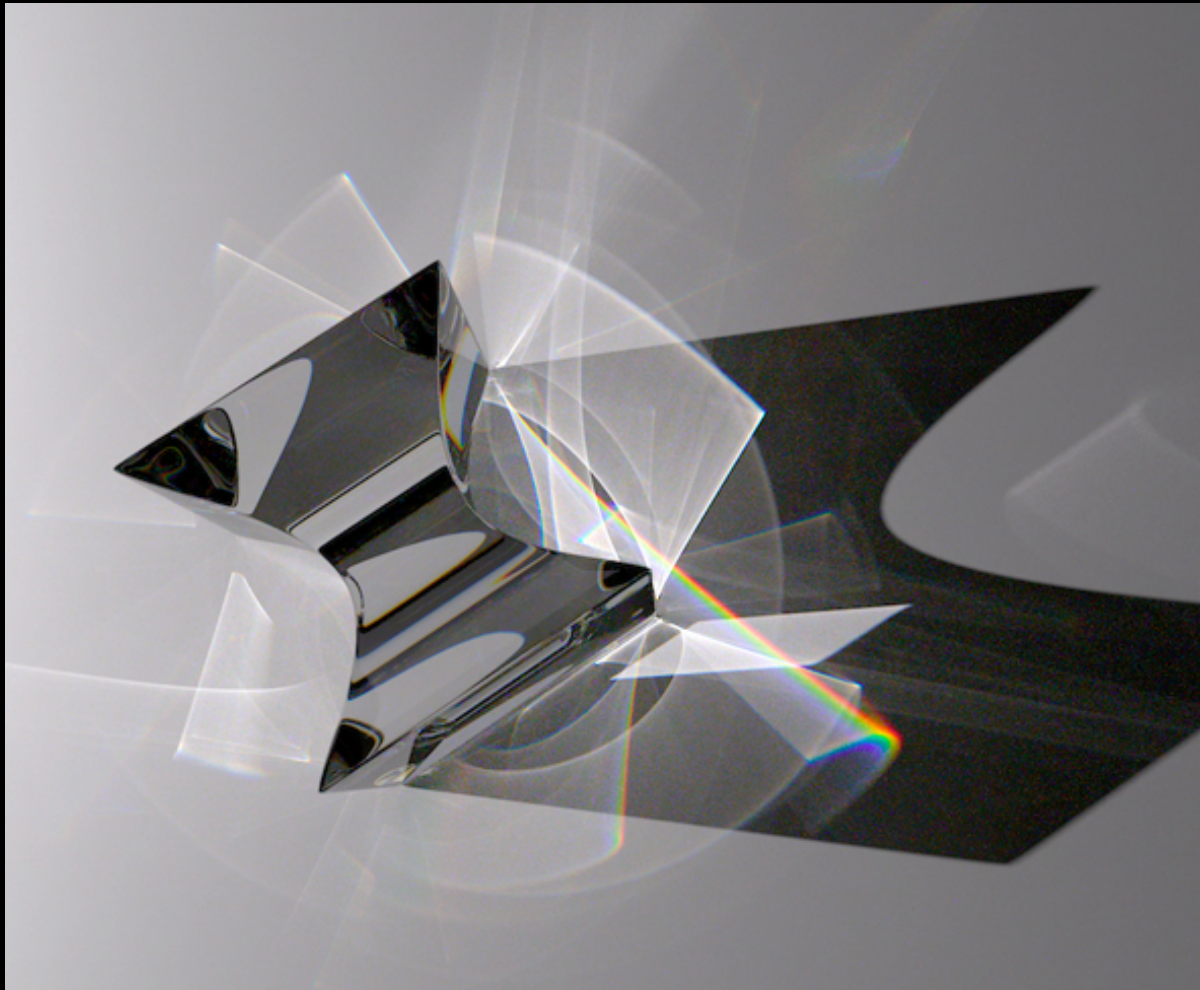
Spectrum to RGB

- Two steps

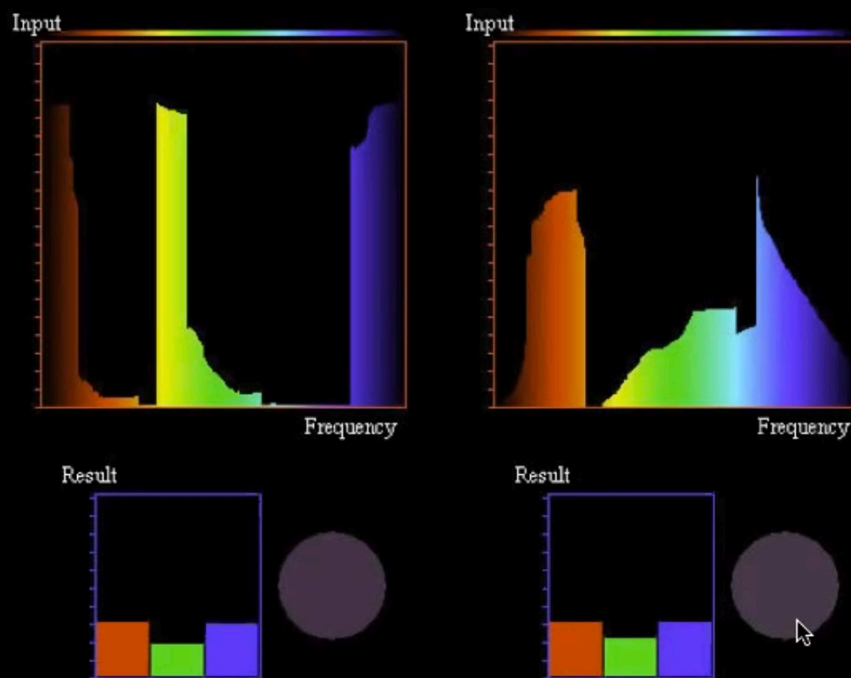 - Compute responses to spectrum as XYZ

  - Convert XYZ to RGB

$$x = \int_{400}^{700} L(\lambda)\bar{x}(\lambda)d\lambda$$

Similar for y & z

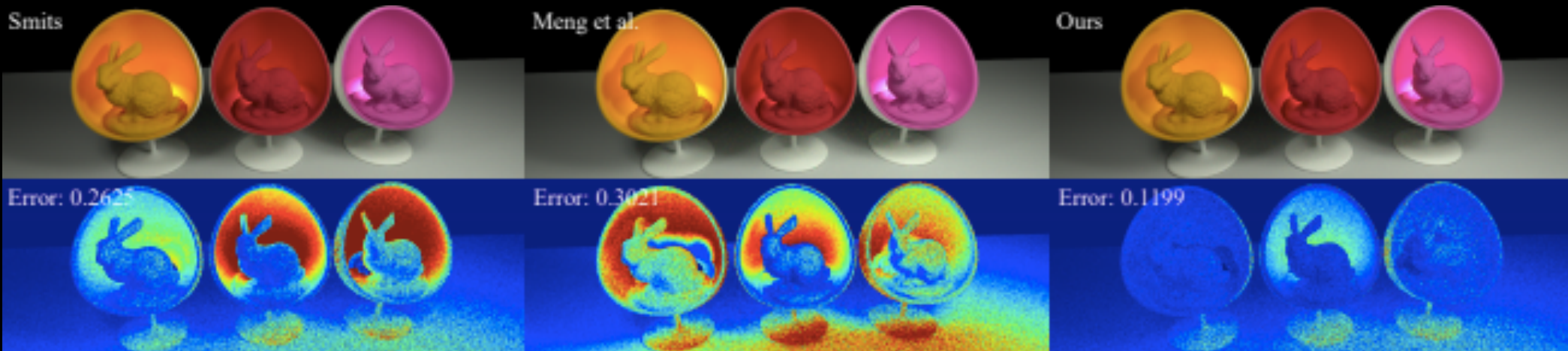$$r = c_0 x + c_1 y + c_2 z$$

RGB to Spectrum

- Ill-conditioned problem

  - Spectrum has much more info than RGB

  - "An RGB to Spectrum Conversion for Reflectances" [Smit 1999]

by Jeff Beall, Adam Doppelt and John F. Hughes
(c) 1995 Brown University and the NSF Graphics and Visualization Center

RGB to Spectrum

"Reproducing Spectral Reflectances from Tristimulus Colors" [Otsu 2018]

High Dynamic Range

- RGB values can go beyond 1.0

W/m^2

10^{-9} 10^{-6} 10^{-3} 10^{0} 10^{3}

Human Eye vs Monitor

- Human eye
  - $2^{14}:1$ brightness difference
  - 14 bits

- Monitor
  - 255:1 brightness difference
  - 8 bits

Tone Mapping

- Convert an HDR value into 0-1 (0-255)

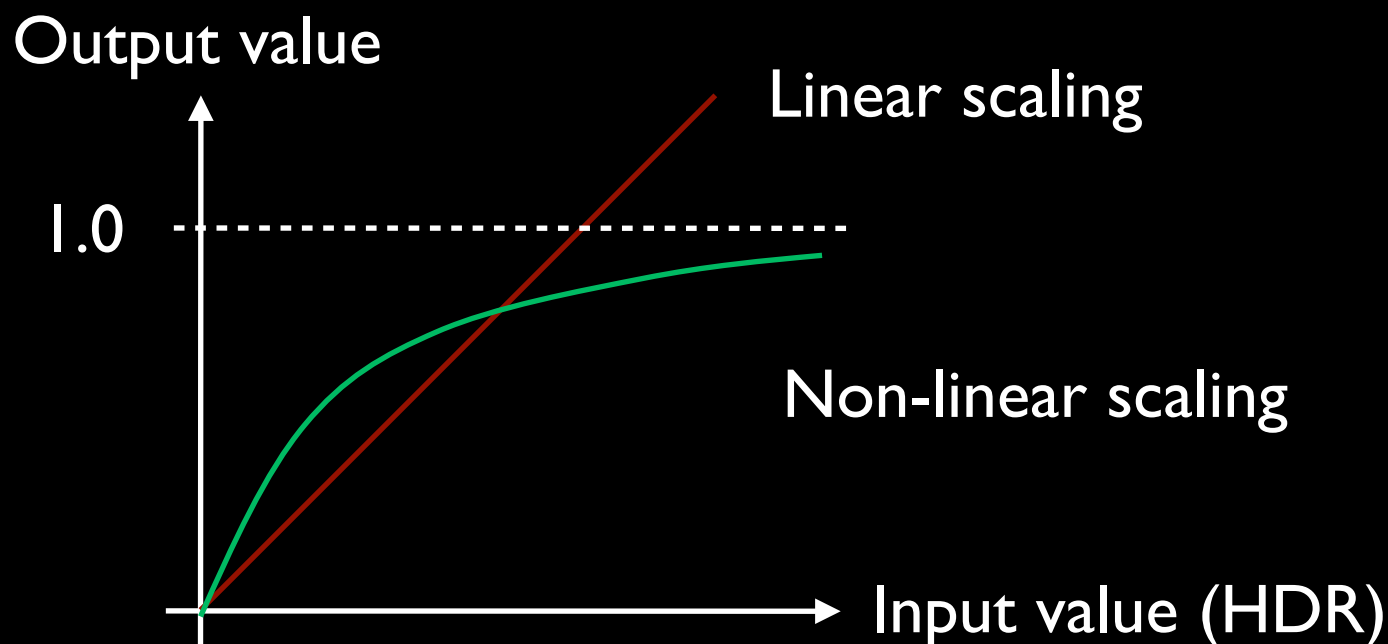# Linear Scaling

- Scale and clamp

$$l(x,y) = \min(cL(x,y), 1.0)$$

Non-linear Scaling

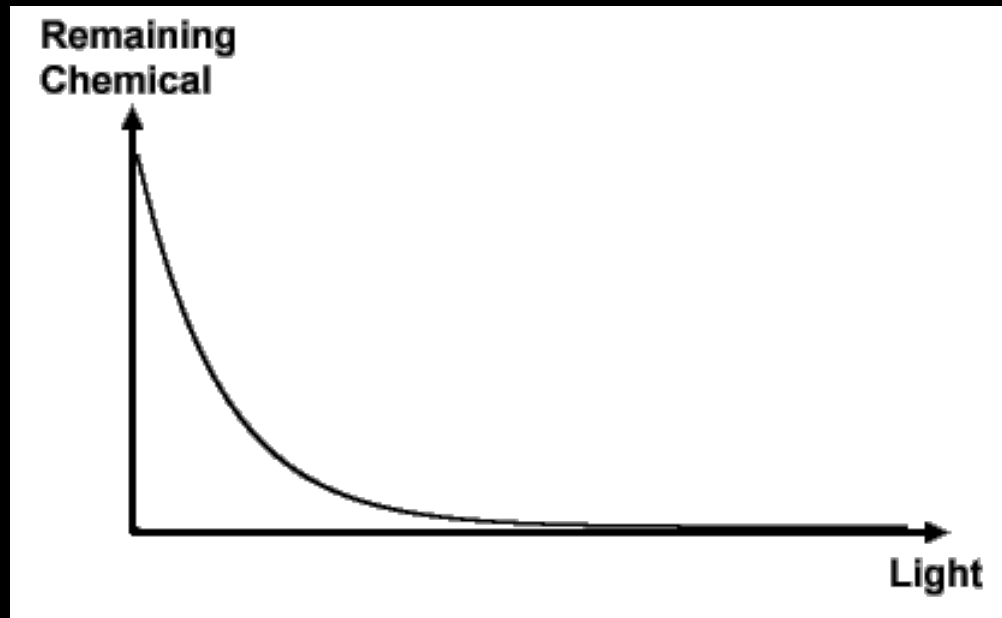- Use a function to "compress" HDR into 0-1

$$l(x, y) = f(L(x, y))$$

Film Response

- Inspired by the response curve of a film
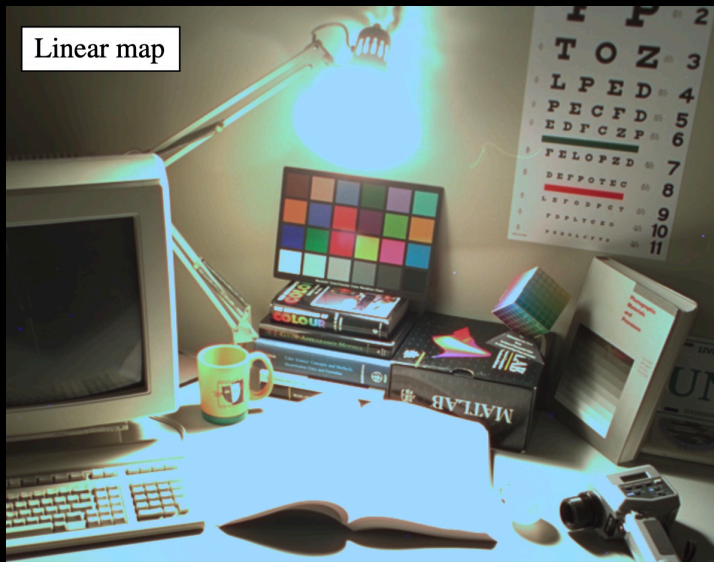
  - Simple and works well often
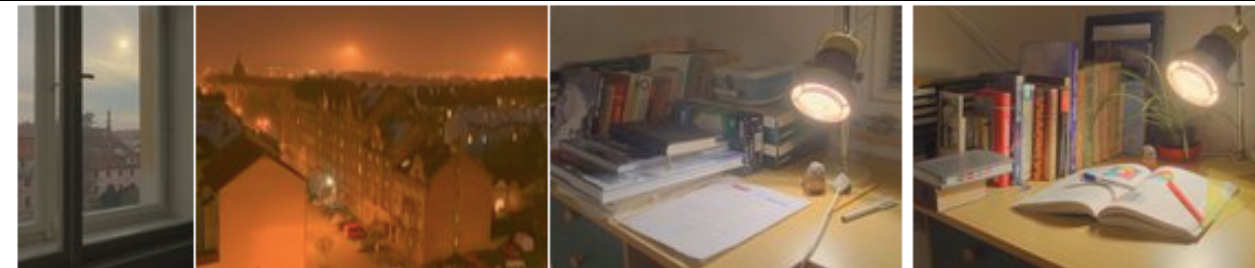
$$l(x, y) = 1.0 - e^{-cL(x,y)}$$

Photographic Tone Reproduction

- Inspired by photography

  - Often works well

  - L_{white} : smallest luminance mapped to 1.0

$$L_d(x,y) = \frac{L(x,y)\left(1 + \frac{L(x,y)}{L_{\mathrm{white}}^2}\right)}{1 + L(x,y)}$$

Linear map

New operator
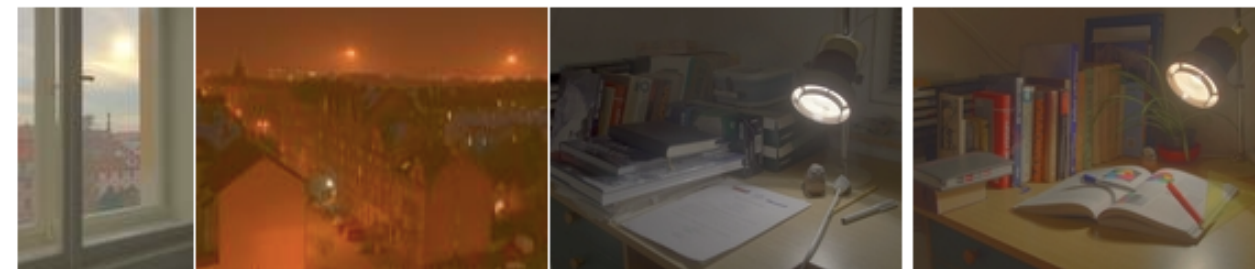
"Photographic Tone Reproduction for Digital Images" [Reinhard 2002]

Comparisons

Drago, F., Myszkowski, K., Annen, T., Chiba, N.

Adaptive Logarithmic Mapping For Displaying High Contrast Scenes
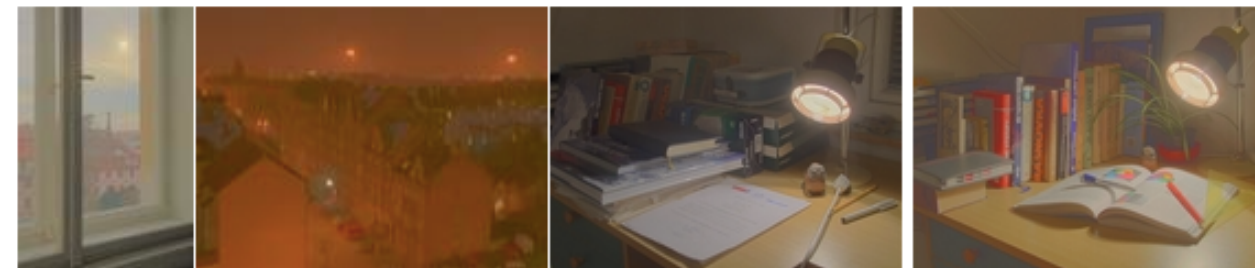
http://www-cg.cis.iwate-u.ac.jp/~frederic/logmap/
CODE AVAILABLE

Tumblin, J., Turk, G.

Low Curvature Image Simplifiers (LCIS): A Boundary Hierarchy for Detail-Preserving Contrast Reduction

http://www.cs.northwestern.edu/~jet/publications.html
CODE AVAILABLE

M. Ashikhmin

A tone mapping algorithm for high contrast images

http://www.cs.sunysb.edu/~ash/tm/

Fredo Durand, Julie Dorsey

Fast bilateral filtering for the display of high-dynamic-range images

http://graphics.csail.mit.edu/~fredo/PUBLI/Siggraph2002/
CODE AVAILABLE

Gamma Correction

- 0.5 is not necessary displayed as "0.5"
    - Nonlinear mapping on a monitor
    - Correction to input tone-mapped value
    - g is typically 1.7-2.2

$$p(x, y) = l(x, y)^{\frac{1}{g}}$$

Gamma Correction

Corrected

No correction