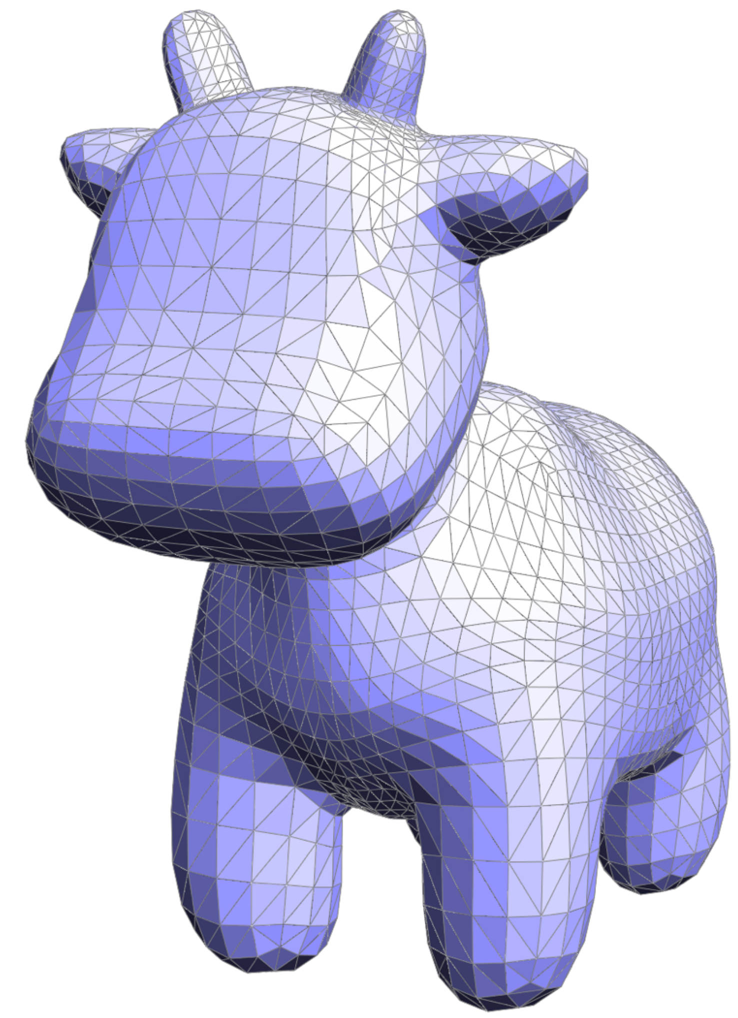
Lecture 10:

Basics of Materials and Lighting

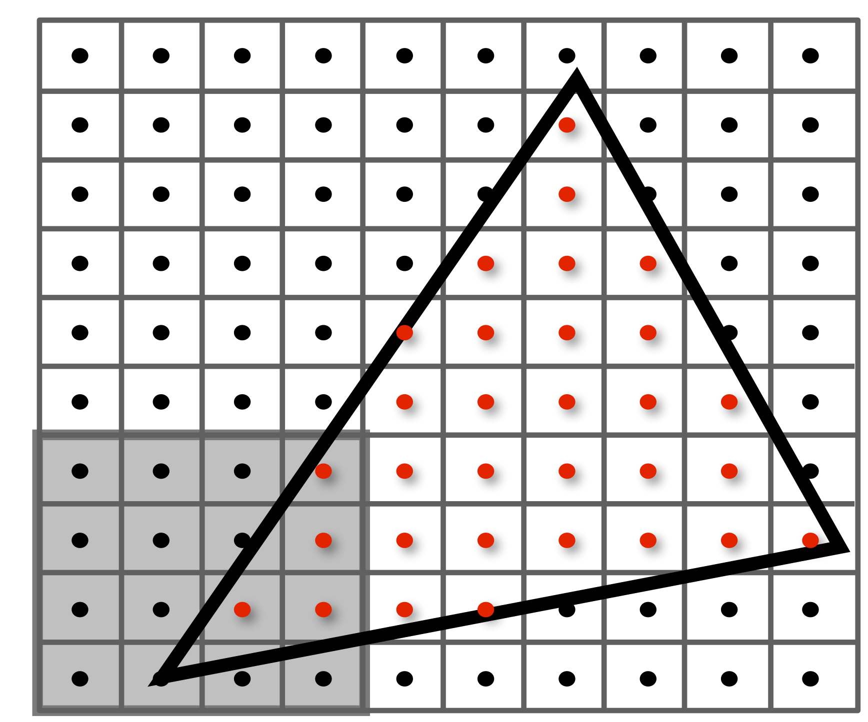
Interactive Computer Graphics
Stanford CS248, Spring 2018

Things you know so far!

- Representing geometry
  - e.g., as triangle meshes

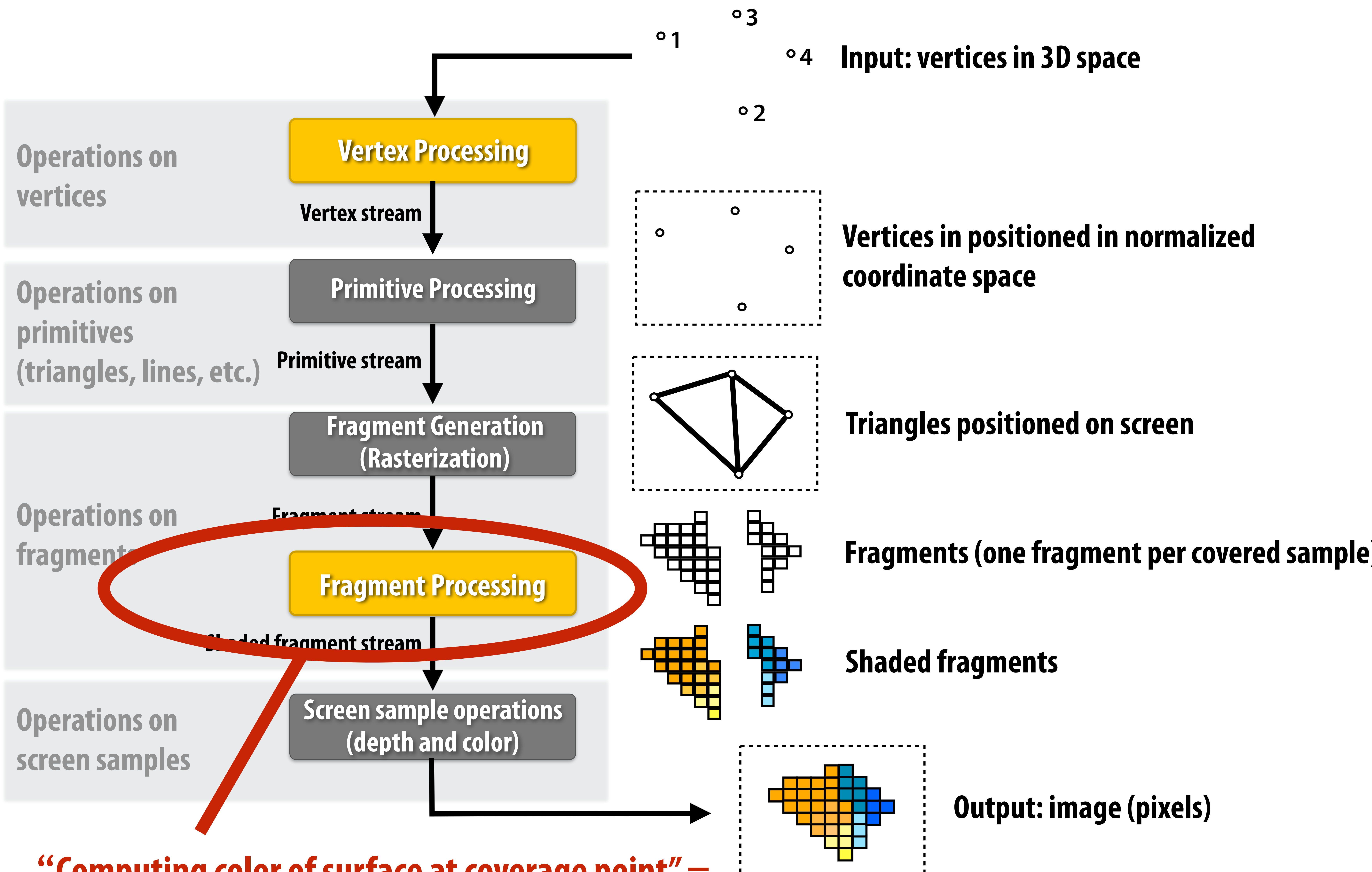


- Rasterizing geometry
  - Determining what point on what triangle covers a sample



- Today: basics of lights and materials
  - Computing the "appearance" of the surface at a point

Recall: OpenGL/Direct3D graphics pipeline

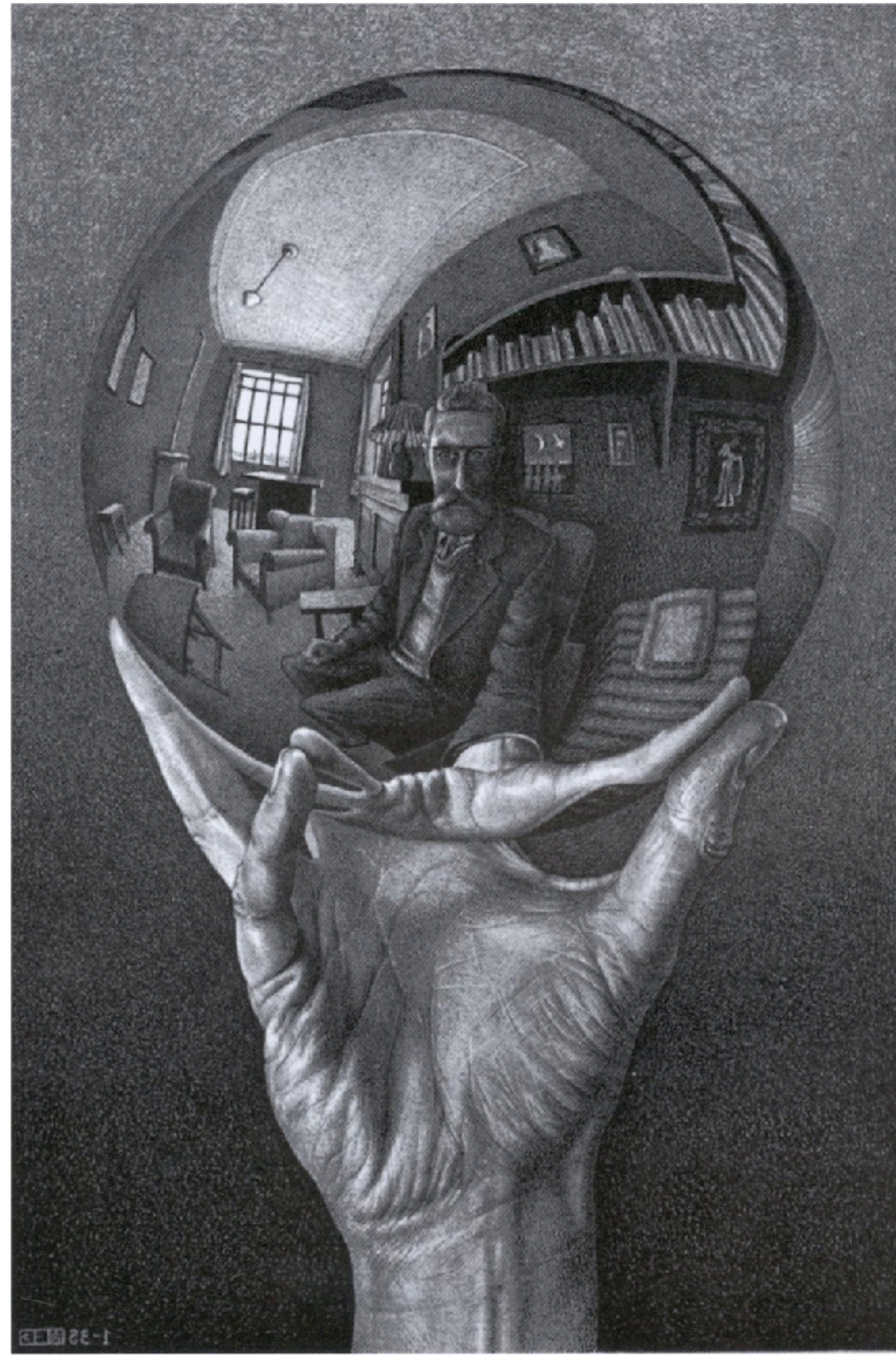


"Computing color of surface at coverage point" = simulation of lighting and materials

"Shading" in drawing

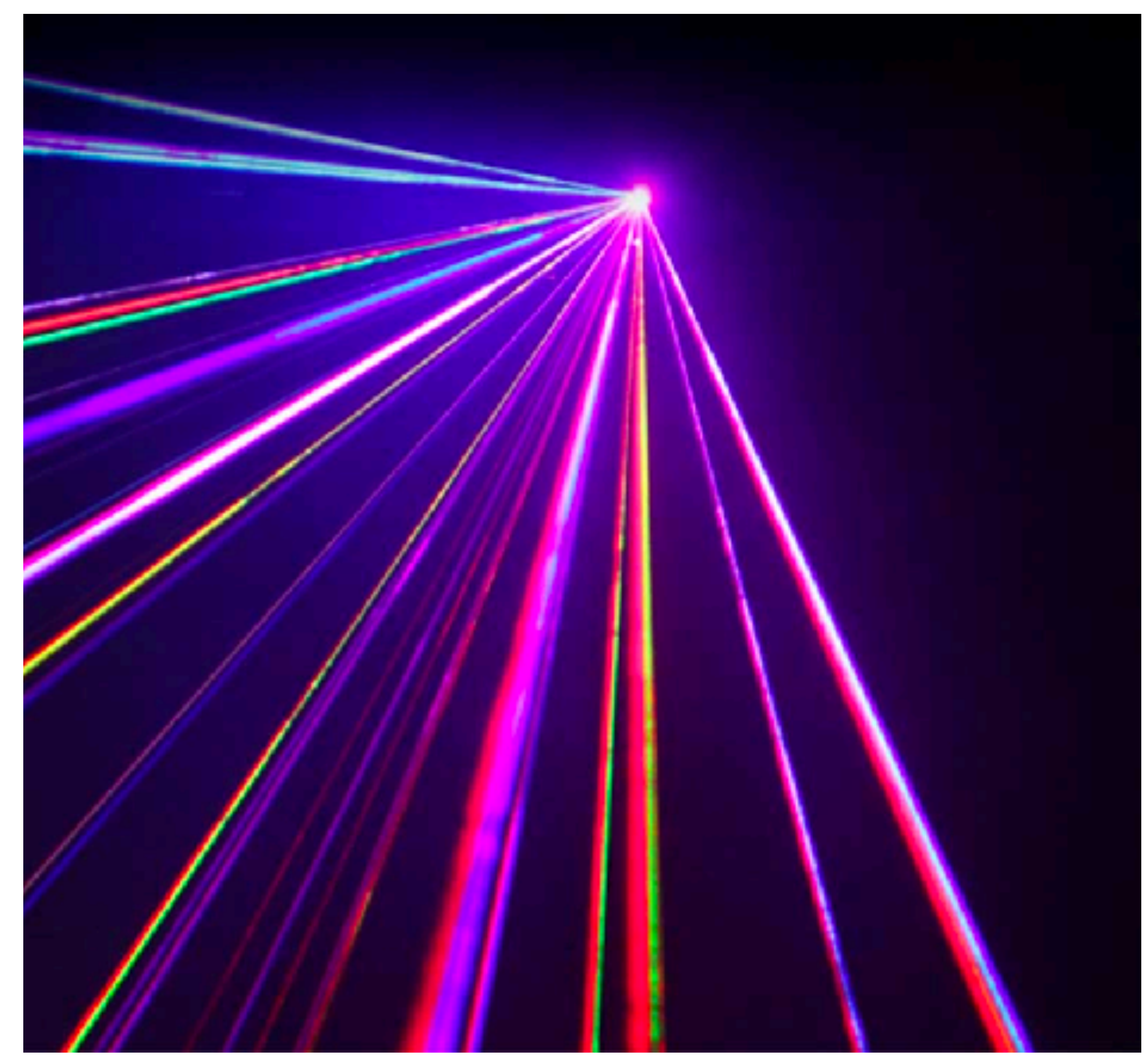
- Depicting the appearance of the surface

- Due to factors like surface material, lighting conditions



MC Escher pencil sketch

Lighting



Lighting

Lighting


Portrait Lighting Cheat Sheet
0° 45° 90° 135° 180° 225° 270° 315°
Flash @45° Down
Flash @0°
Flash @45° Up
(cc) DIYPhotography.net

Materials: diffuse

Materials: plastic

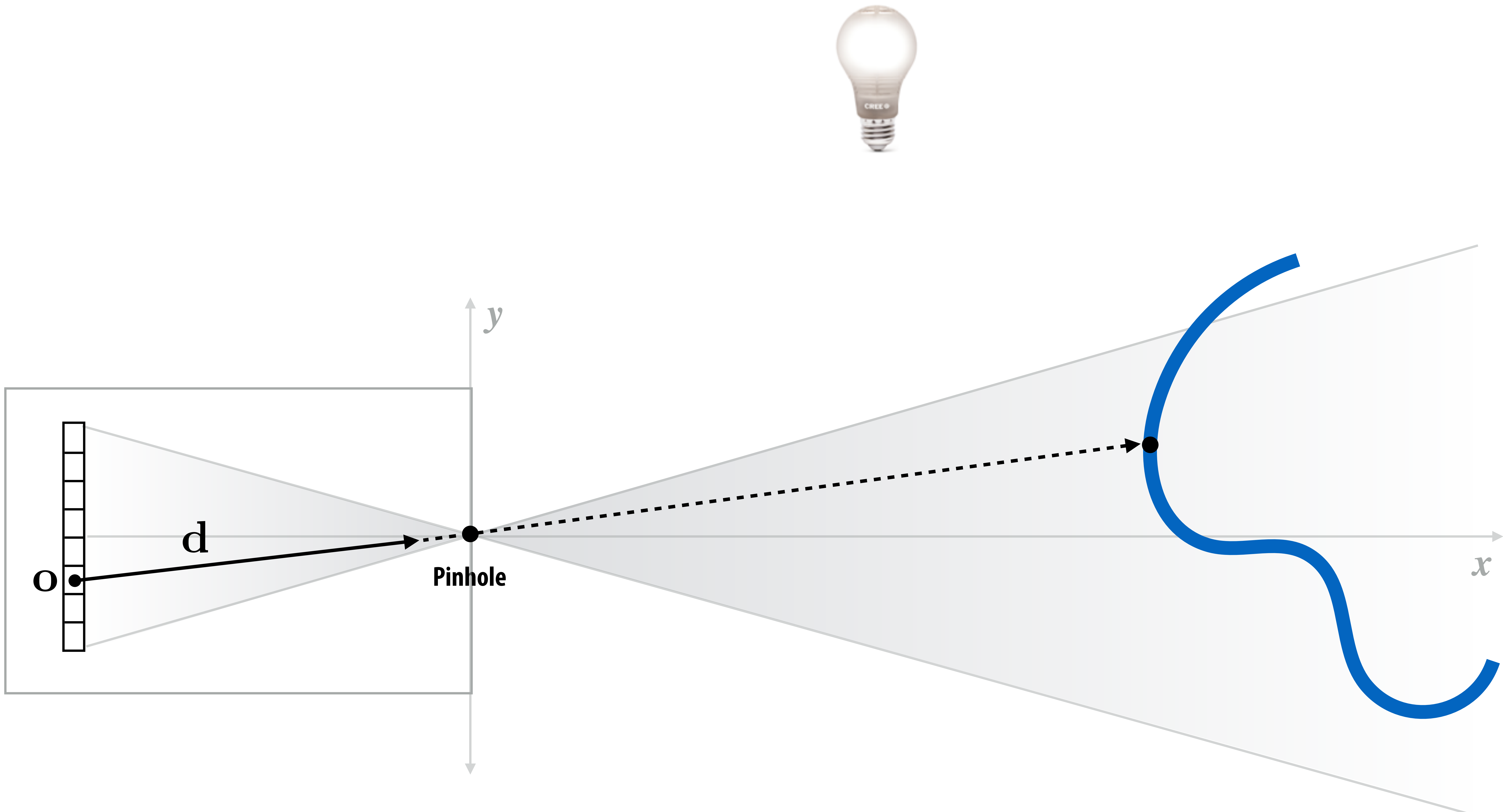
Materials: red semi-gloss paint

Materials: Ford mystic lacquer paint

Materials: mirror

Materials: gold

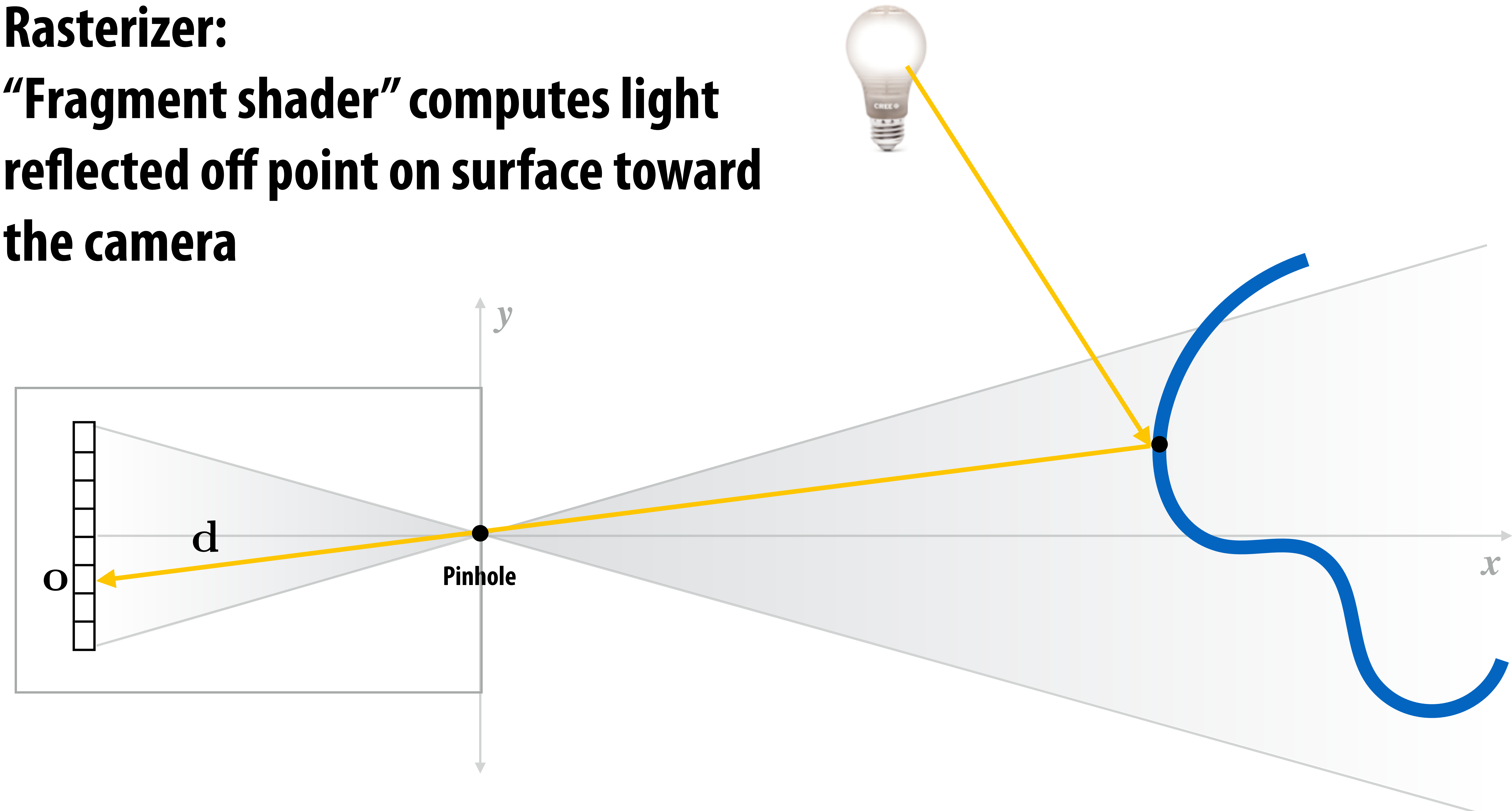
Renderer measures light energy along a ray



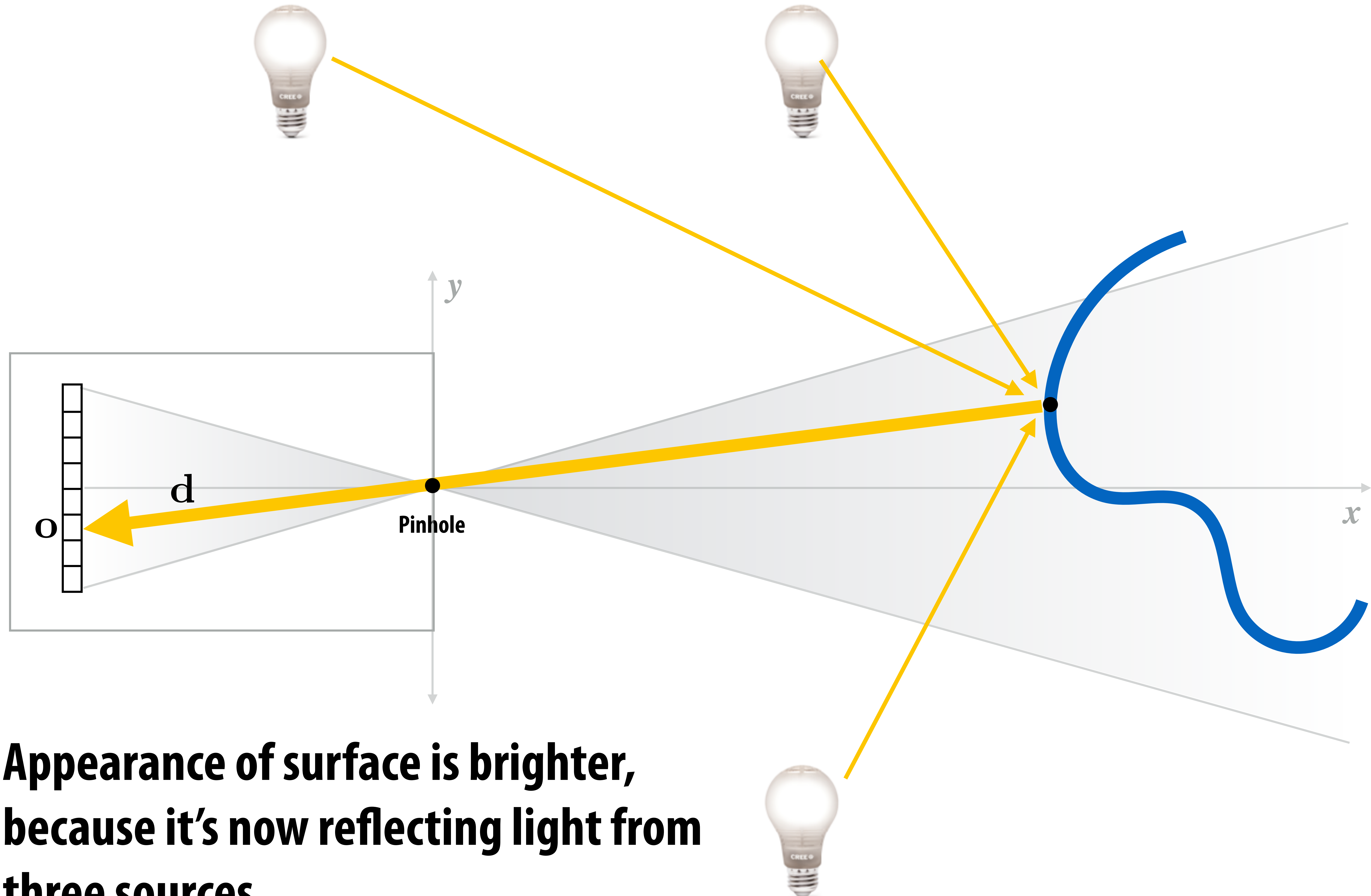
Renderer measures light energy along a ray

Rasterizer:

"Fragment shader" computes light reflected off point on surface toward the camera



Multiple light sources

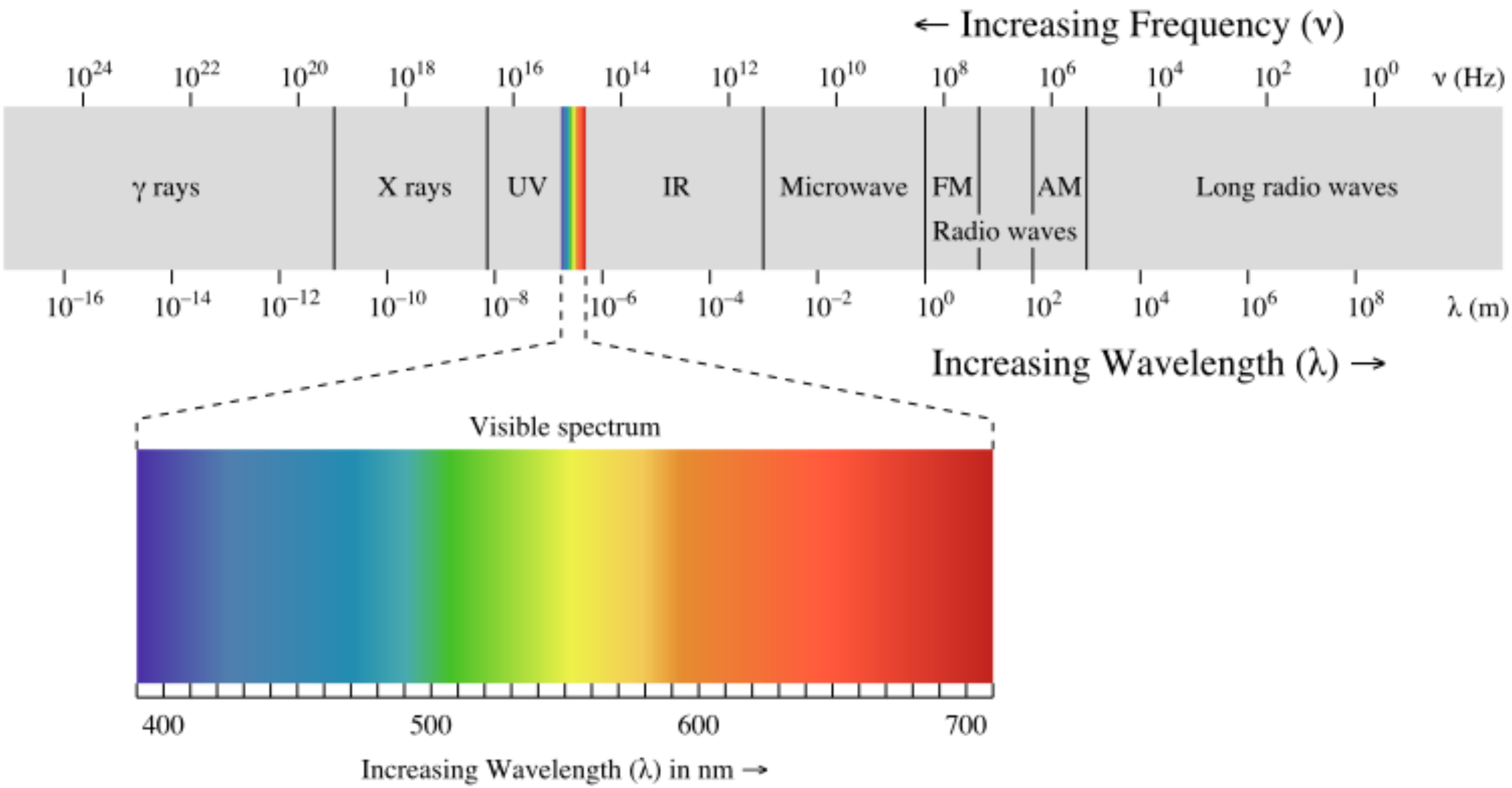


Appearance of surface is brighter, because it's now reflecting light from three sources.

Mini-tutorial on radiometry
(wait for 348B!)

Electromagnetic radiation

Light is electromagnetic radiation that is visible to eye



Lights: what do they do?

Cree 11 W LED light bulb
("60 Watt" incandescent replacement)

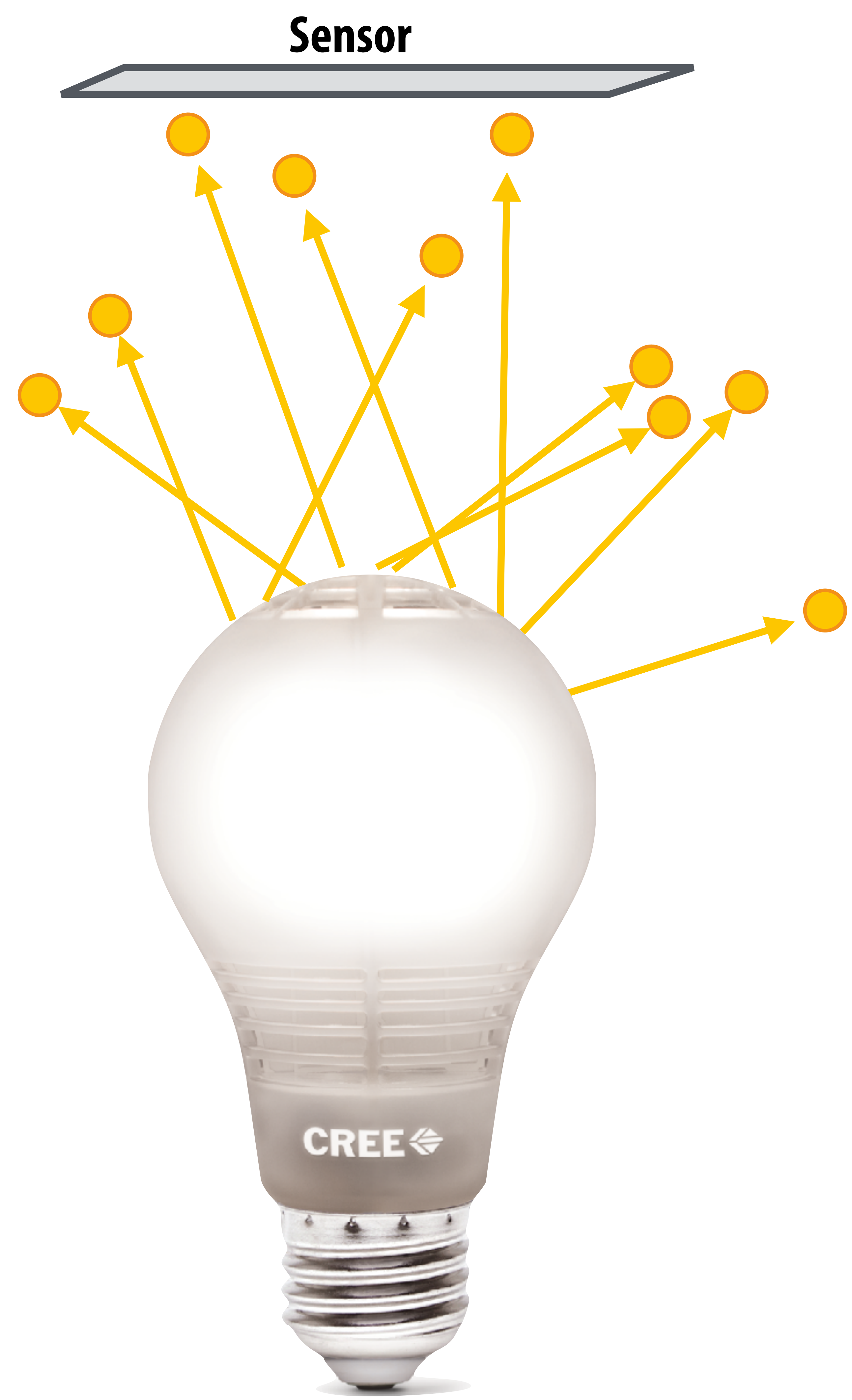
- Physical process converts input energy into photons
  - Each photon carries a small amount of energy
- Over some amount of time, light fixture consumes some amount of energy, Joules
  - Some input energy is turned into heat, some into photons
- Energy of photons hitting an object ~ exposure
  - Film, sensors, sunburn, solar panels, …
- Graphics: generally assume "*steady state*" process
  - Rate of energy consumption = power, Watts (Joules/second)

Measuring illumination: radiant flux (power)

- Given a sensor, we can count how many photons reach it
  - Over a period of time, gives the power received by the sensor
- Given a light, consider counting the number of photons emitted by it
  - Over a period of time, gives the power emitted by the light
- Energy carried by a photon:

$$Q = \frac{hc}{\lambda}$$

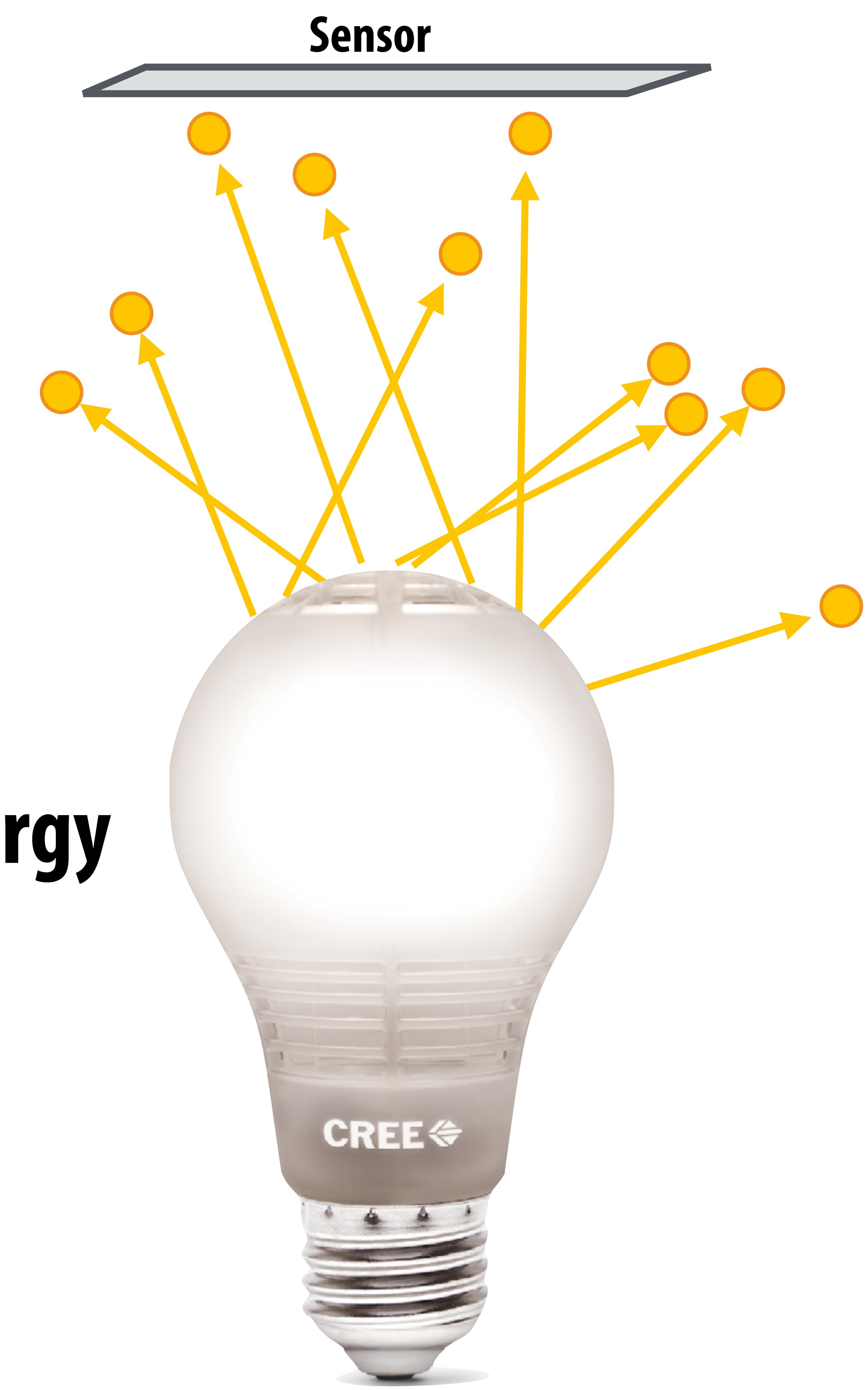
$$h \approx 6.626 \times 10^{-34}$$



Measuring illumination: radiant flux (power)

- **Flux: energy per unit time (Watts) received by the sensor (or emitted by the light)**

$$\Phi = \lim_{\Delta \to 0} \frac{\Delta Q}{\Delta t} = \frac{\mathrm{d}Q}{\mathrm{d}t} \quad \left[\frac{\mathrm{J}}{\mathrm{s}}\right]$$



- **Time integral of flux is total radiant energy**

$$Q = \int_{t_0}^{t_1} \Phi(t)\,\mathrm{d}t$$

Spectral power distribution

- **Describes distribution of energy by wavelength**

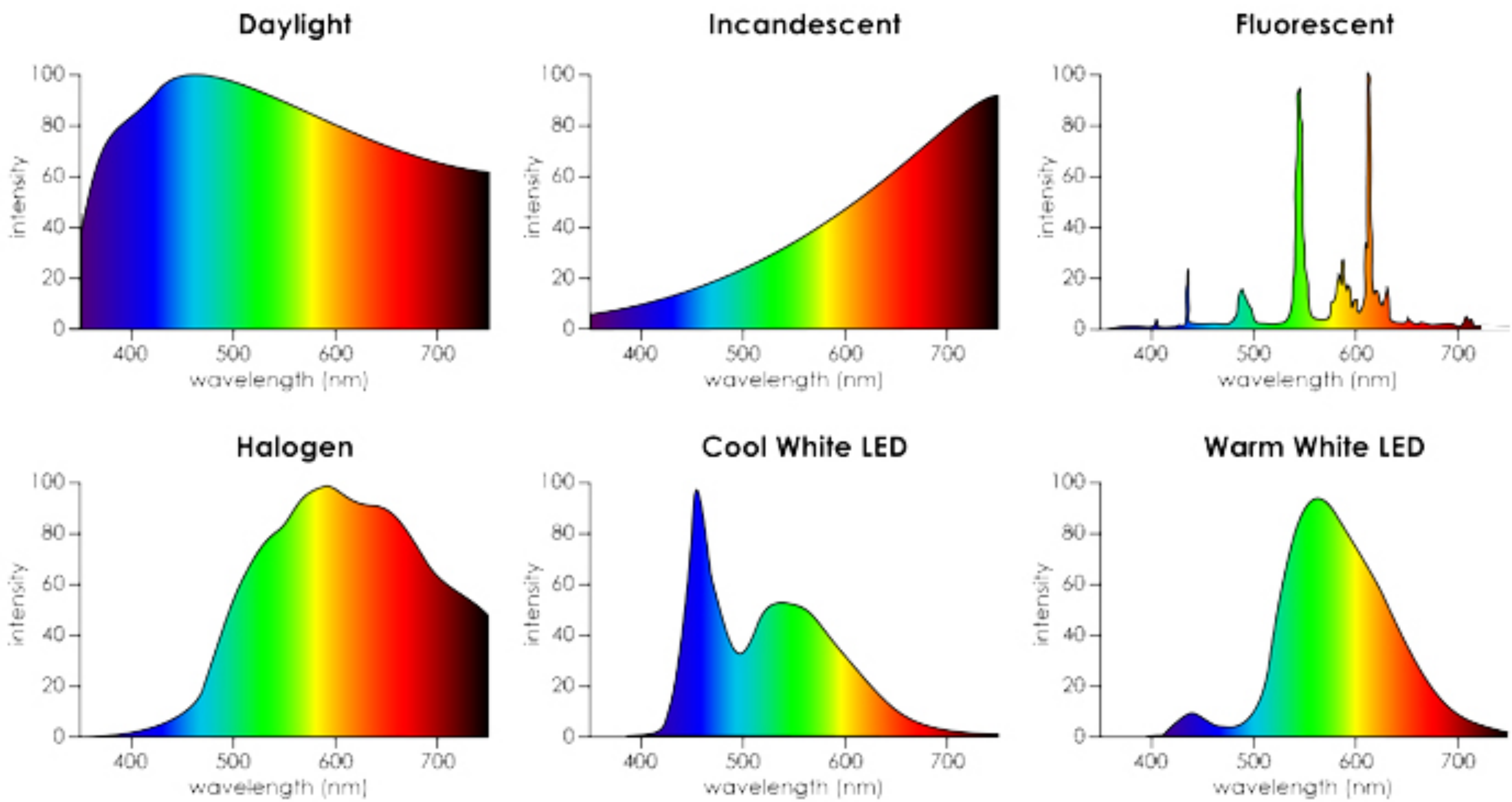
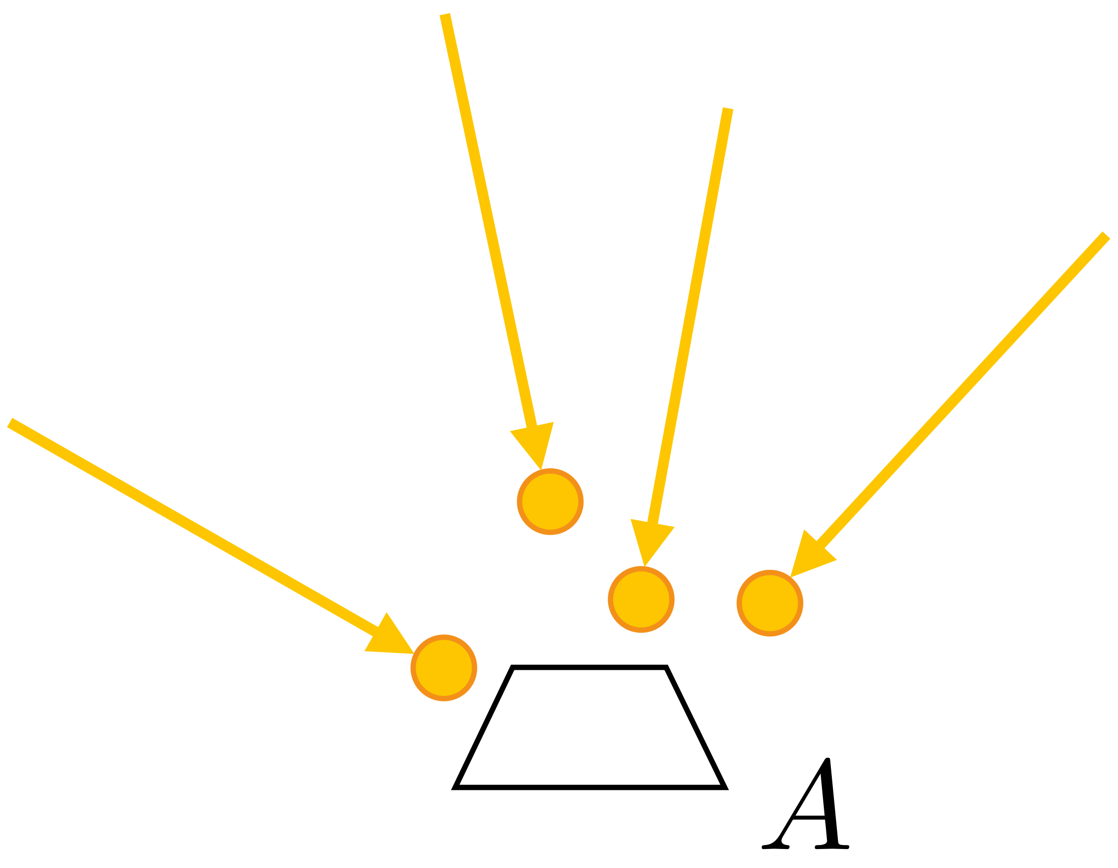


Figure credit:

admesy
ADVANCED MEASUREMENT SYSTEMS

Measuring illumination: irradiance

- Flux: time density of energy
- Irradiance: area density of flux



Given a sensor of with area A, we can consider the average flux over the entire sensor area:

$$\frac{\Phi}{A}$$

Irradiance (E) is given by taking the limit of area at a single point on the sensor:

$$E(\mathrm{p}) = \lim_{\Delta \to 0} \frac{\Delta\Phi(\mathrm{p})}{\Delta A} = \frac{\mathrm{d}\Phi(\mathrm{p})}{\mathrm{d}A} \quad \left[\frac{\mathrm{W}}{\mathrm{m}^2}\right]$$

Beam power in terms of irradiance

Consider beam with flux Φ incident on surface with area *A*

$$E = \frac{\Phi}{A}$$

$$\Phi = EA$$

Projected area

Consider beam with flux Φ incident on angled surface with area *A′*

$$A = A' \cos\theta$$

A = projected area of surface relative to direction of beam

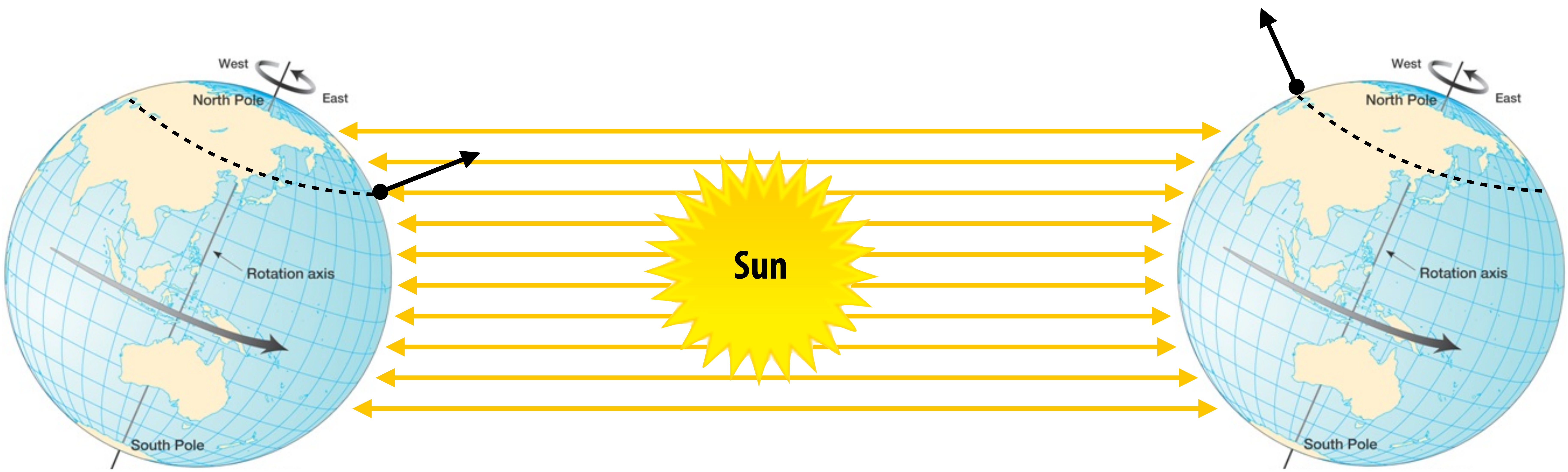
Lambert's Law

Irradiance at surface is proportional to cosine of angle between light direction and surface normal.

$$A = A' \cos\theta$$

$$E = \frac{\Phi}{A'} = \frac{\Phi\cos\theta}{A}$$

Why do we have seasons?



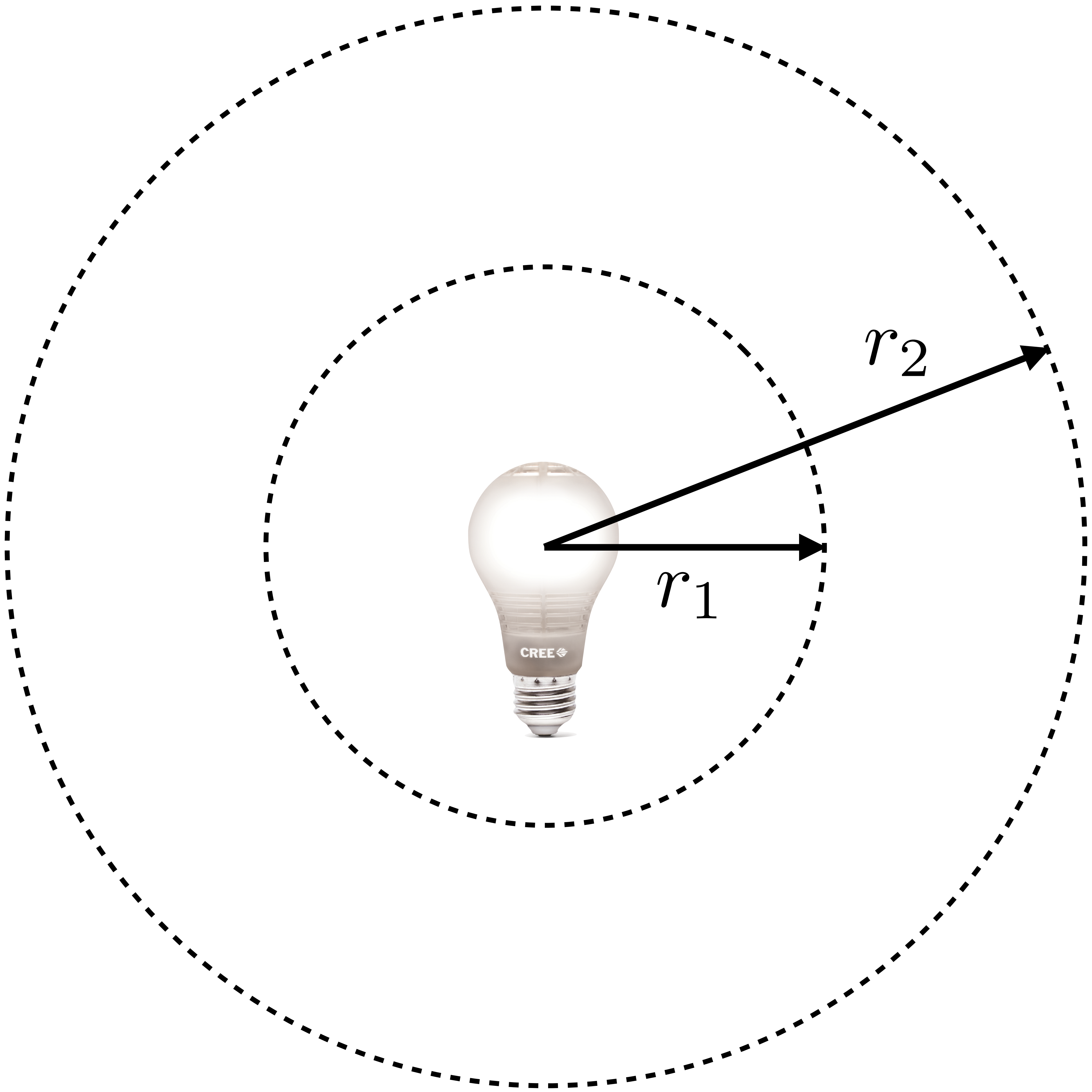
Summer
(Northern hemisphere)

Winter
(Northern hemisphere)

Earth's axis of rotation: ~23.5° off axis

[Image credit: Pearson Prentice Hall]

Irradiance falloff with distance



Assume light is emitting flux Φ in a uniform angular distribution

Compare irradiance at surface of two spheres:

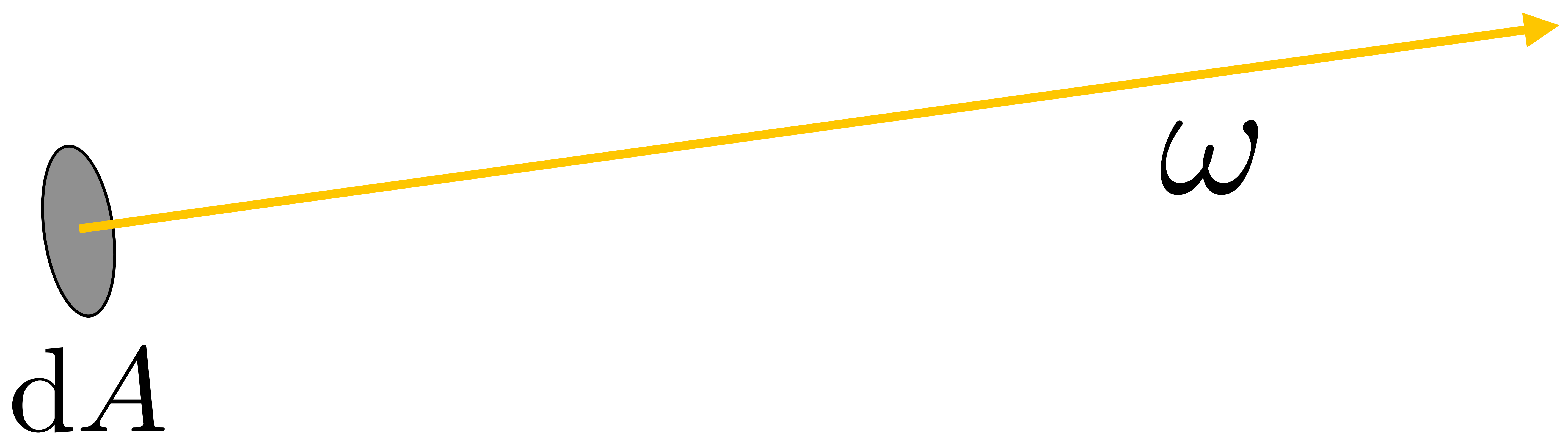
$$E_1 = \frac{\Phi}{4\pi r_1^2} \rightarrow \Phi = 4\pi r_1^2 E_1$$

$$E_2 = \frac{\Phi}{4\pi r_2^2} \rightarrow \Phi = 4\pi r_2^2 E_2$$

$$\frac{E_2}{E_1} = \frac{r_1^2}{r_2^2}$$

Measuring illumination: radiance

Radiance (L) is irradiance per unit direction.

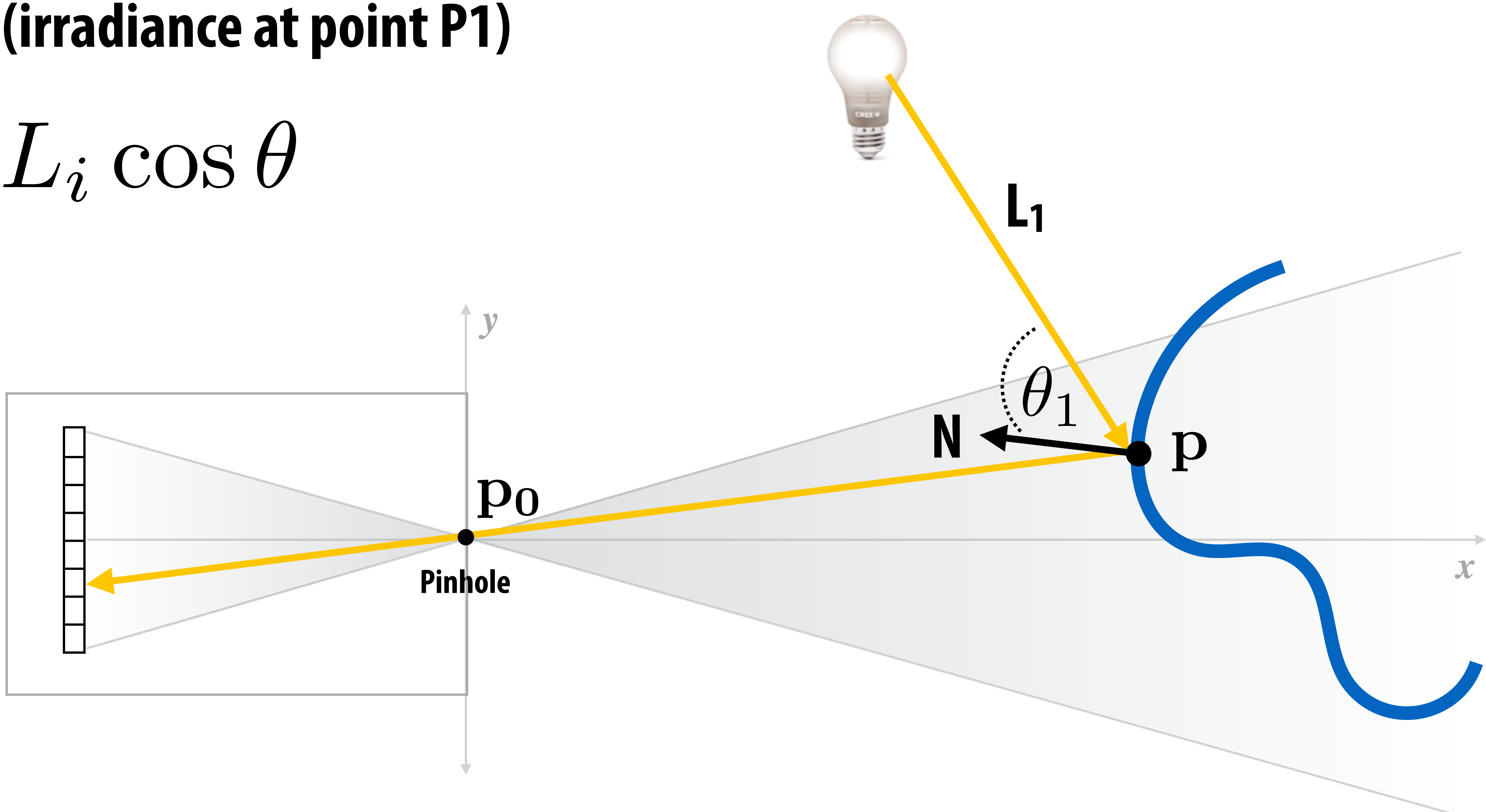


In other words, radiance is energy along a ray defined by origin point *p* and direction ω

How much light hits the surface at point p

(irradiance at point P1)

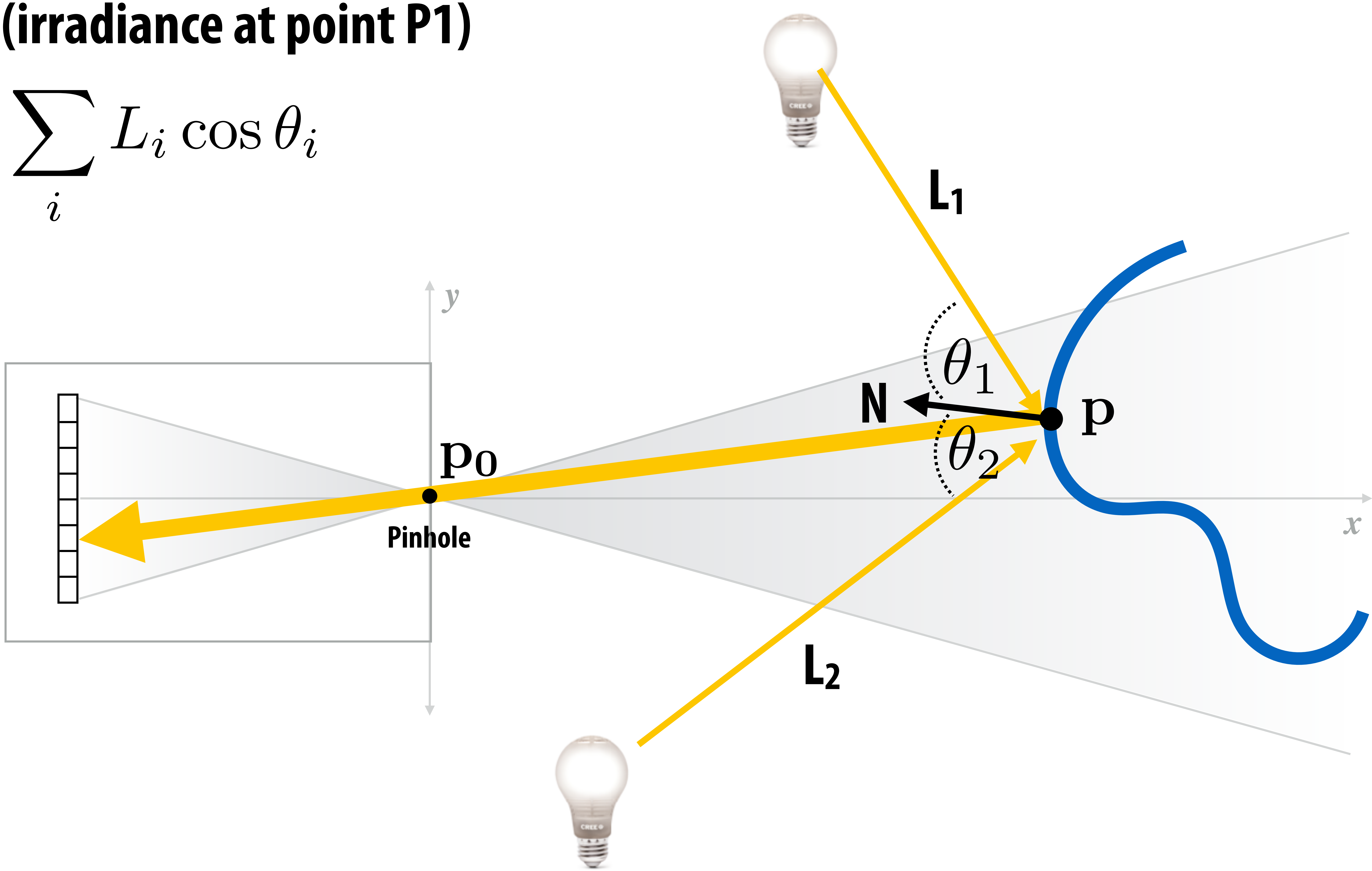
$$L_i \cos\theta$$



How much light hits the surface at point p

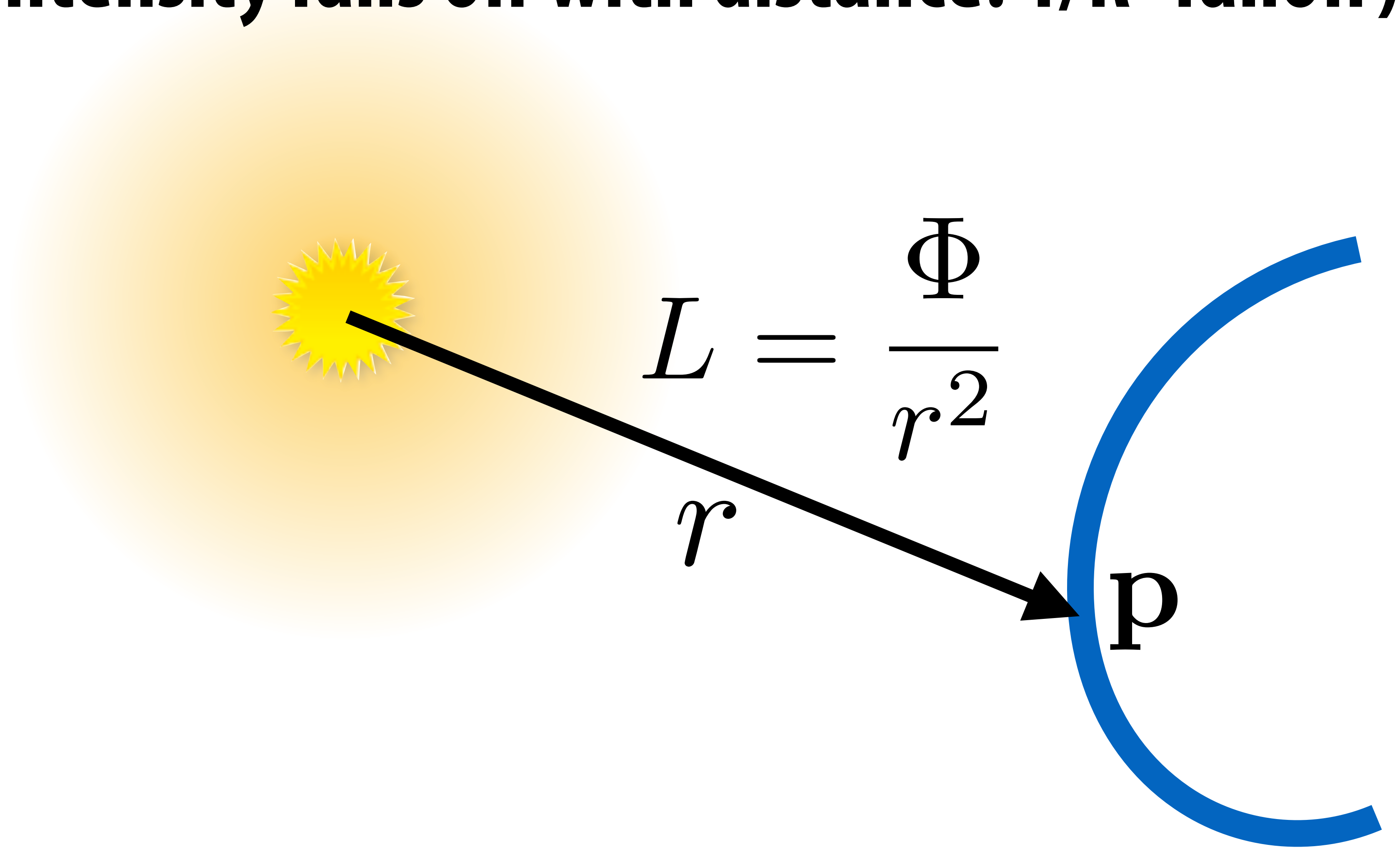
(irradiance at point P1)

$$\sum_i L_i \cos\theta_i$$

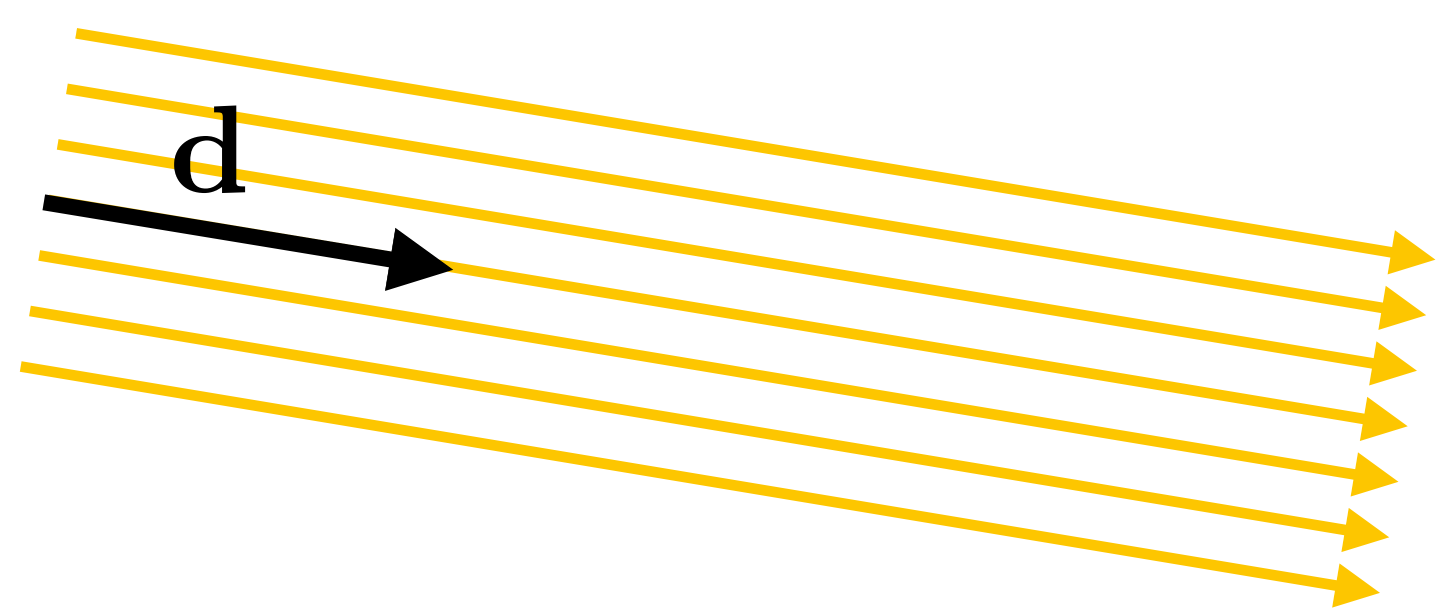


Types of lights

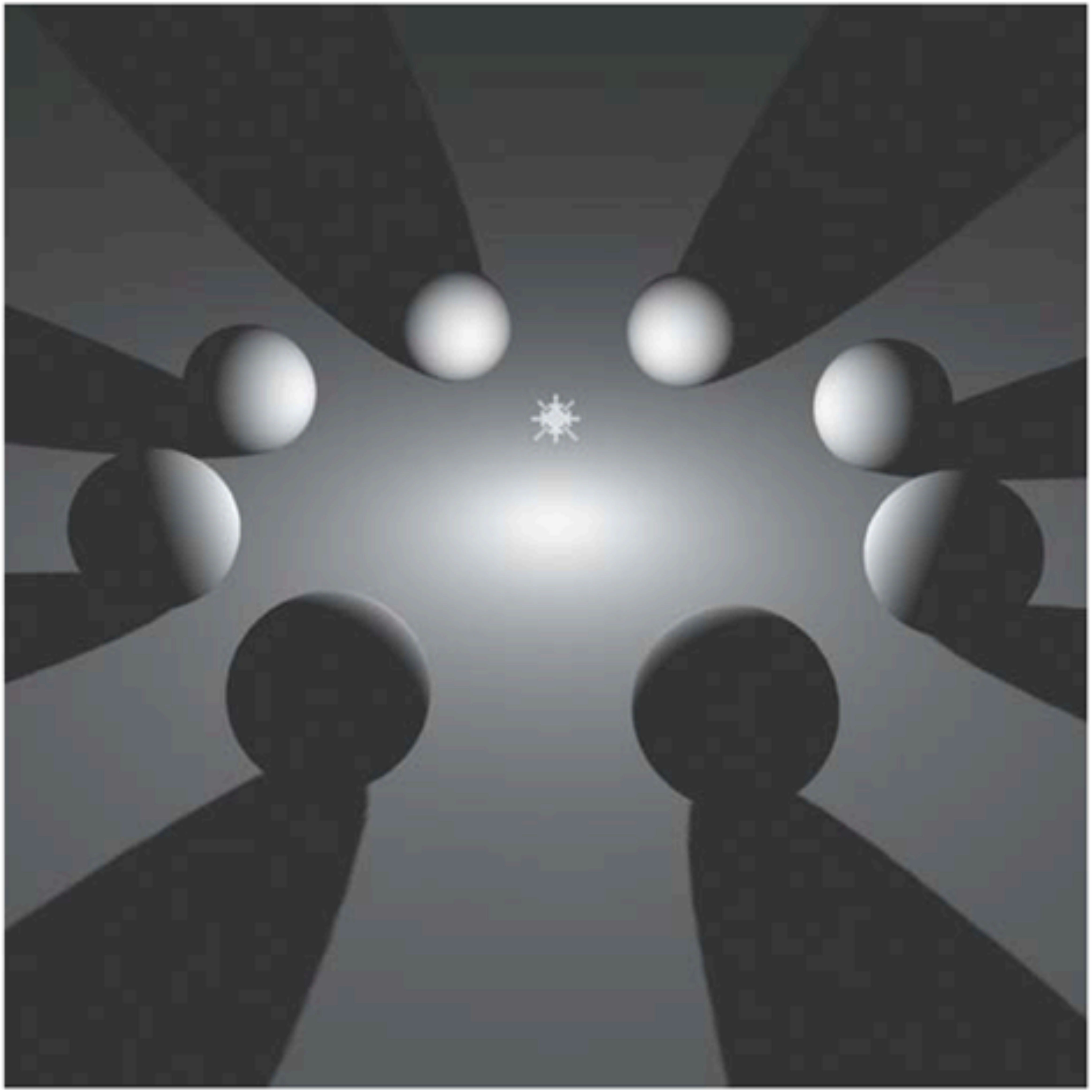
- **Attenuated omnidirectional point light**
  (emits equally in all directions, intensity falls off with distance: $1/R^2$ falloff)



- **Infinite directional light in direction *d***
  (infinitely far away, all points in scene receive light with radiance L from direction *d*

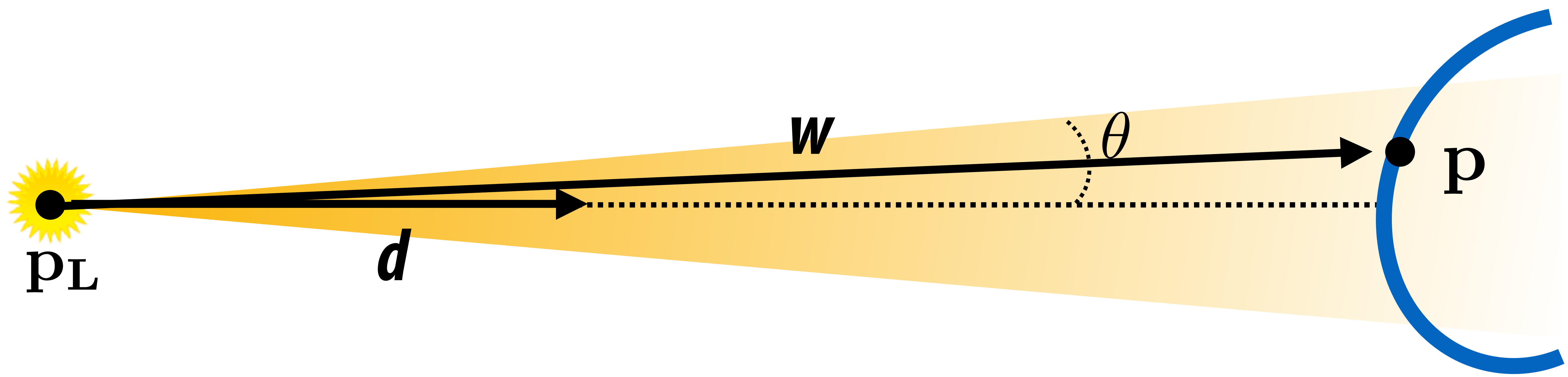


Point light with shadows



Types of lights

- **Spot light**
  (does not emit equally in all directions)



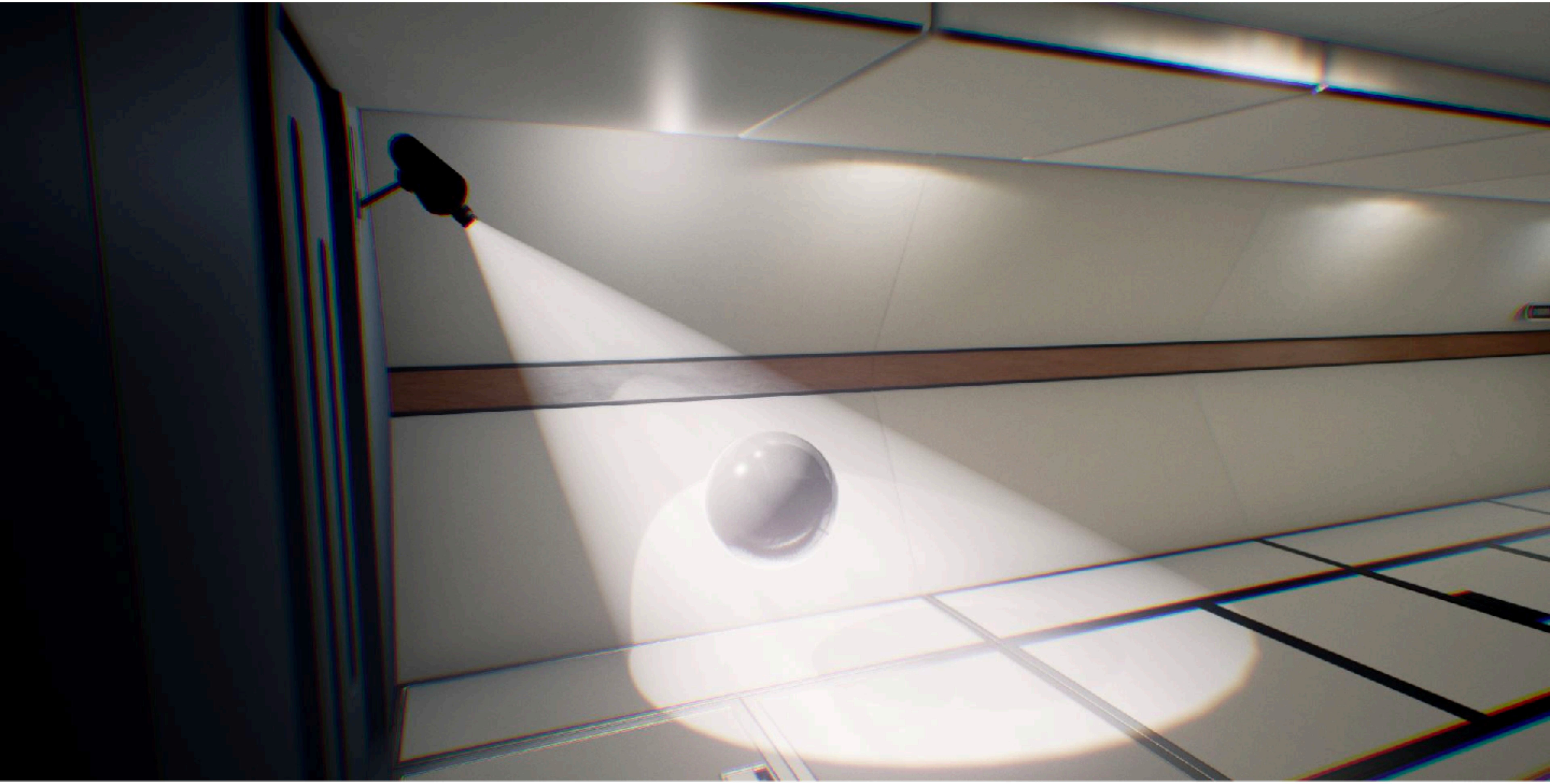
$$\mathbf{w} = \text{normalize}(\mathbf{p} - \mathbf{p_L})$$

$$L(\mathbf{w}) = 0 \quad \textbf{if} \quad \mathbf{w} \cdot \mathbf{d} > \theta$$
$$= L_0 \quad \textbf{otherwise}$$

Or, if spotlight intensity falls off from direction *d*

$$L(\mathbf{w}) \approx \mathbf{w} \cdot \mathbf{d}$$

Spot light



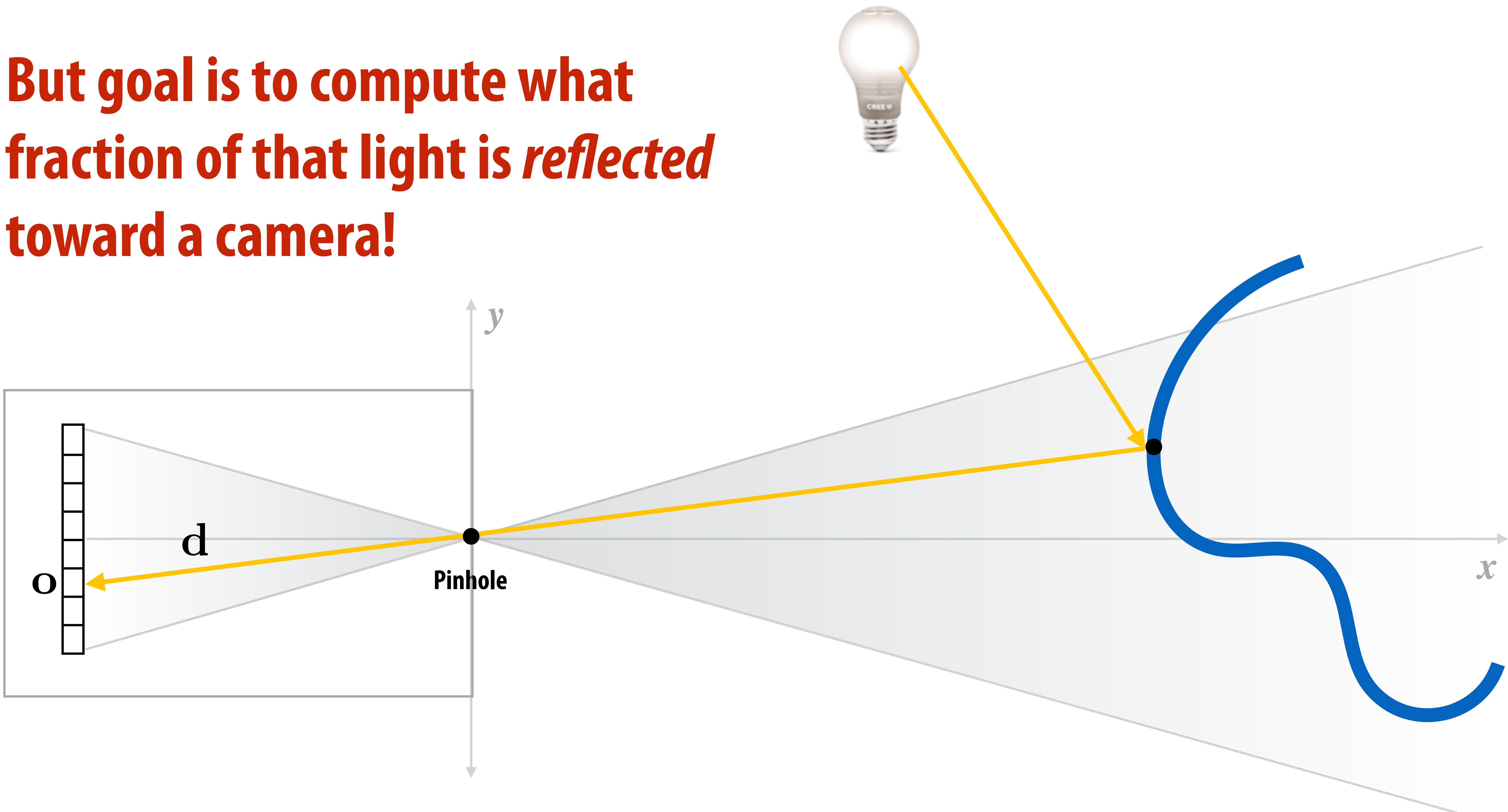
Environment map (more complicated light)

Pixel (x,y) stores radiance L from direction (ϕ, θ)

Image credit: USC High-Resolution Light Probe Image Gallery

So far… how to compute the light (radiance) arriving at a surface point

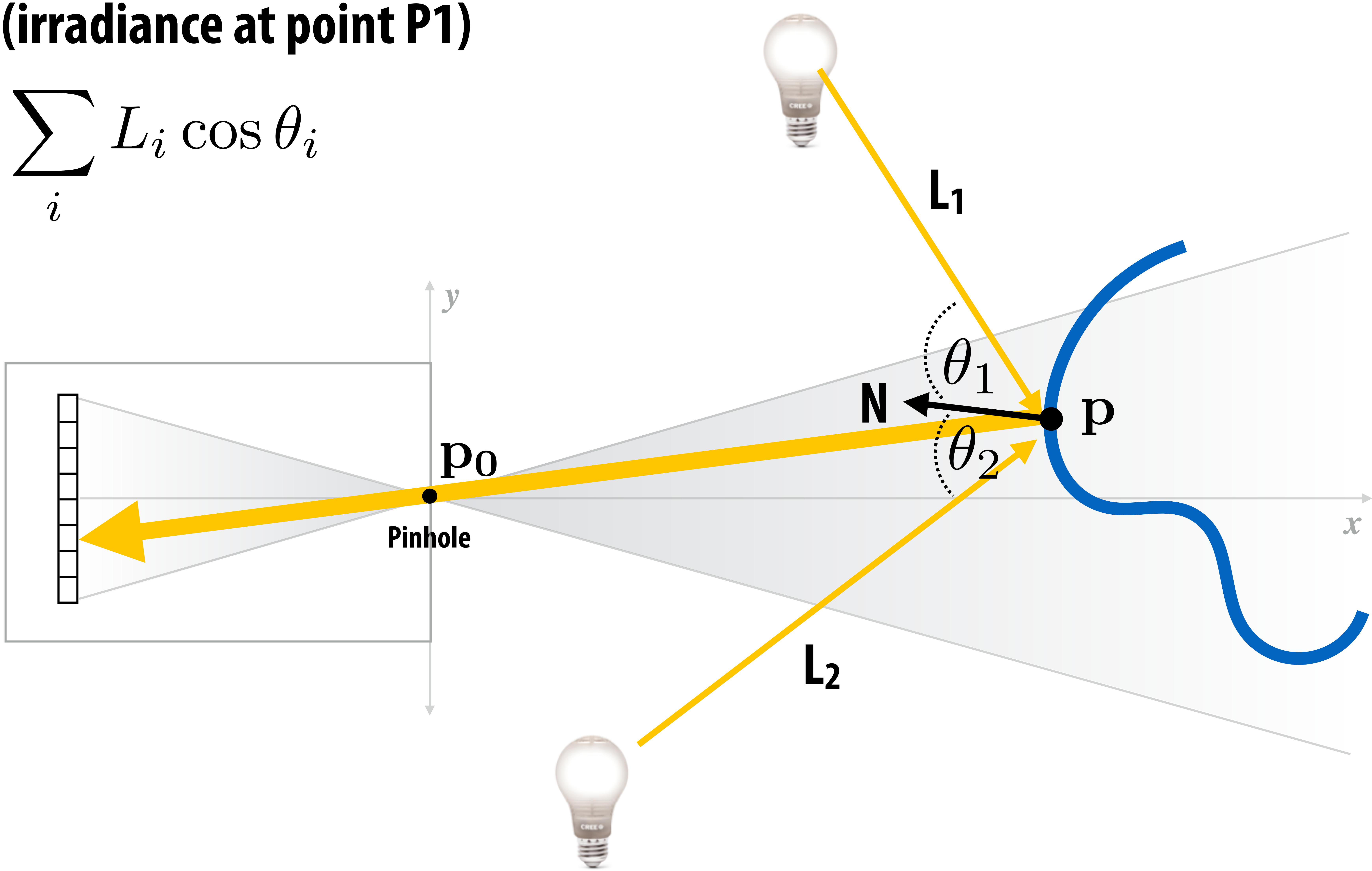
But goal is to compute what fraction of that light is *reflected* toward a camera!



How much light hits the surface at point p

(irradiance at point P1)

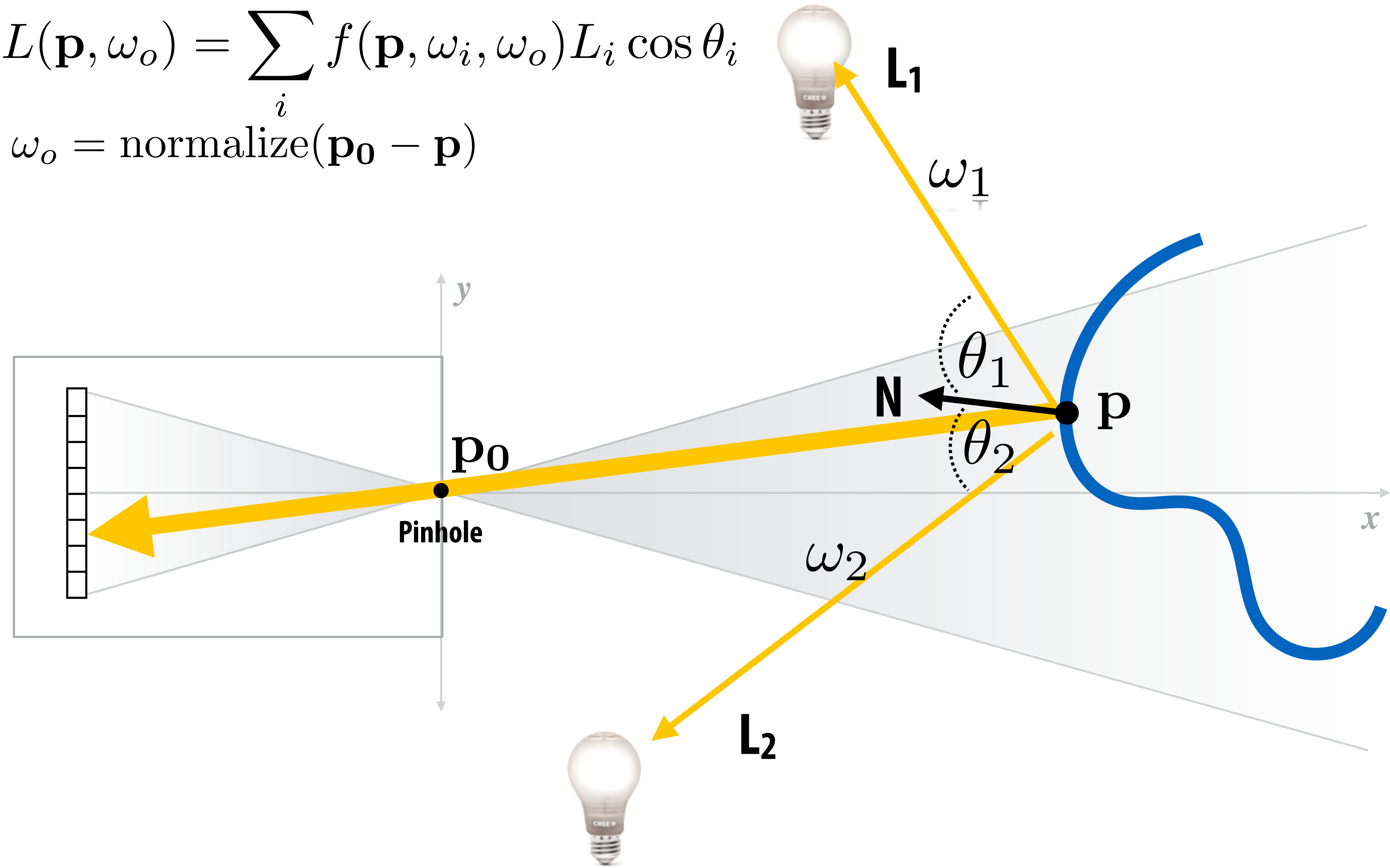
$$\sum_i L_i \cos\theta_i$$



How much light is REFLECTED from p toward p_0

$$L(\mathbf{p}, \omega_o) = \sum_i f(\mathbf{p}, \omega_i, \omega_o) L_i \cos\theta_i$$

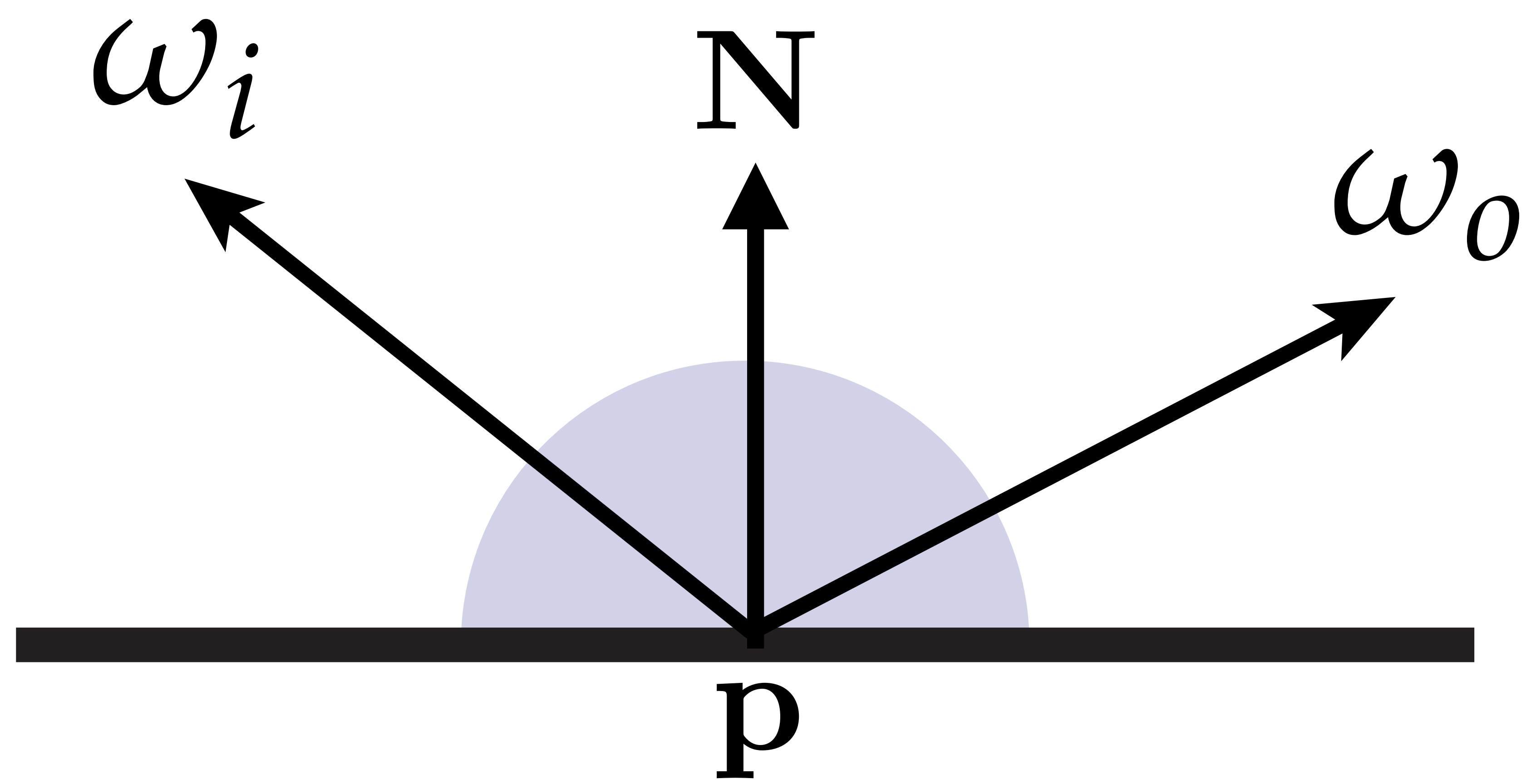
$$\omega_o = \text{normalize}(\mathbf{p_0} - \mathbf{p})$$



Bidirectional reflectance distribution function (BRDF)

- **Gives fraction of light arriving at surface point P from direction w_i is reflected in direction w_0**

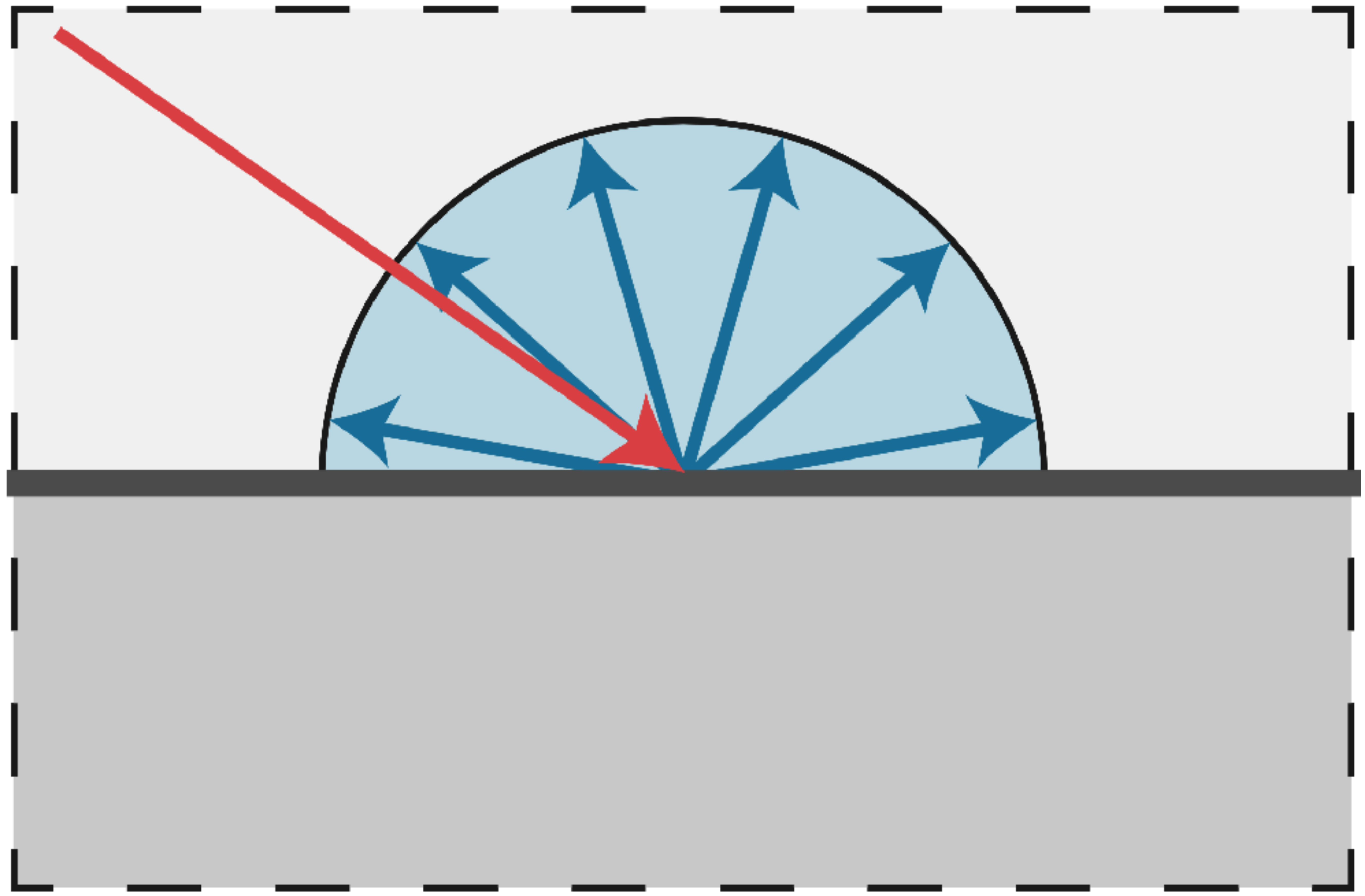
$$f(\mathbf{p}, \omega_i, \omega_o)$$



Reflection models

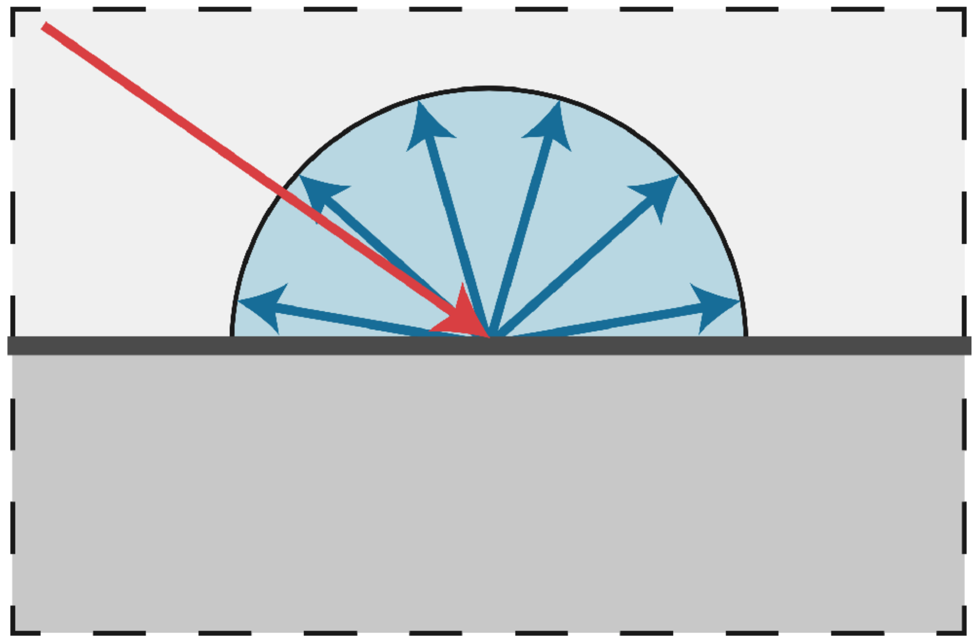
- *Reflection* is the process by which light incident on a surface interacts with the surface such that it leaves on the incident (same) side without change in frequency
- Choice of reflection function determines surface appearance

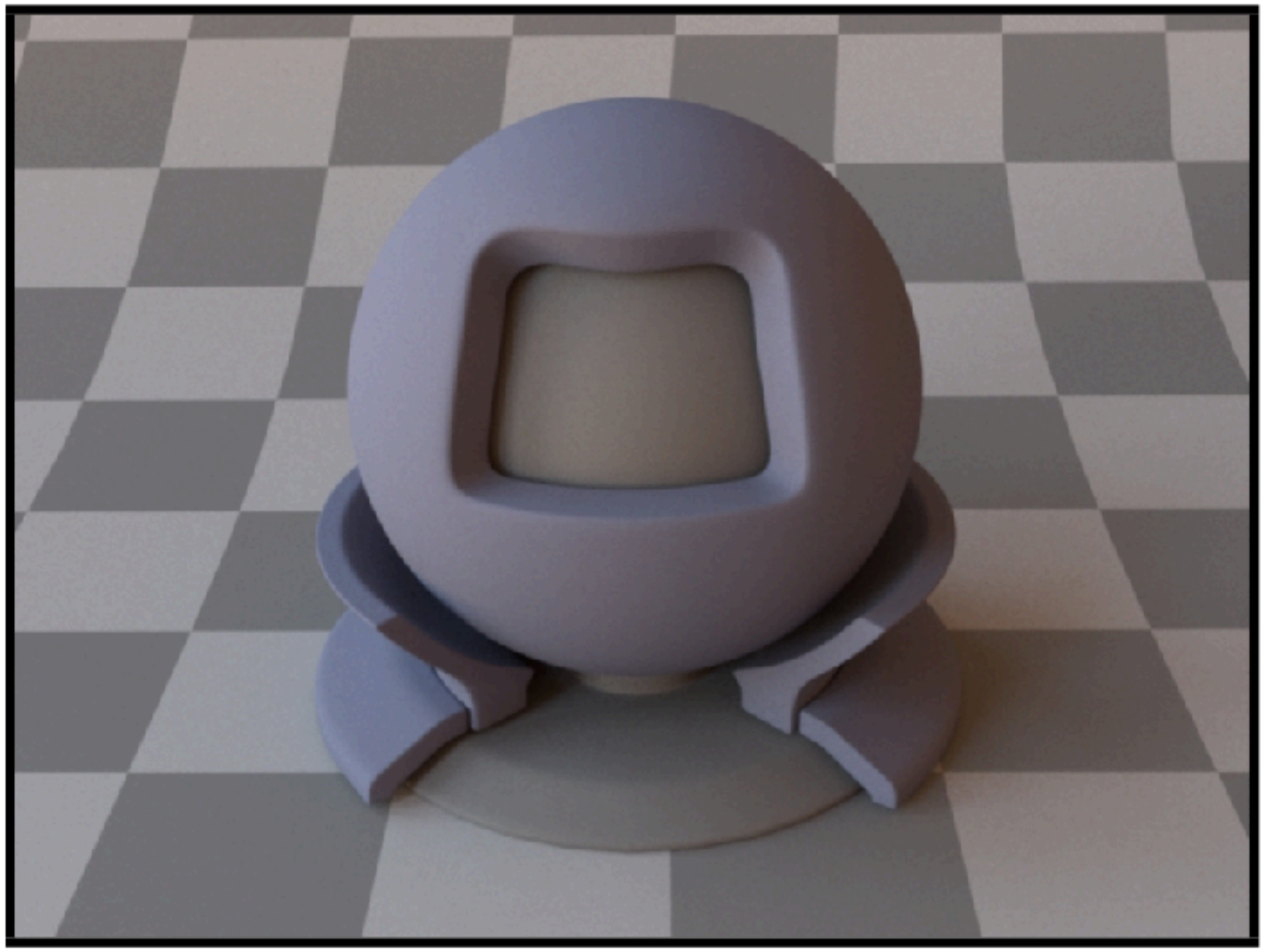
What is this material?



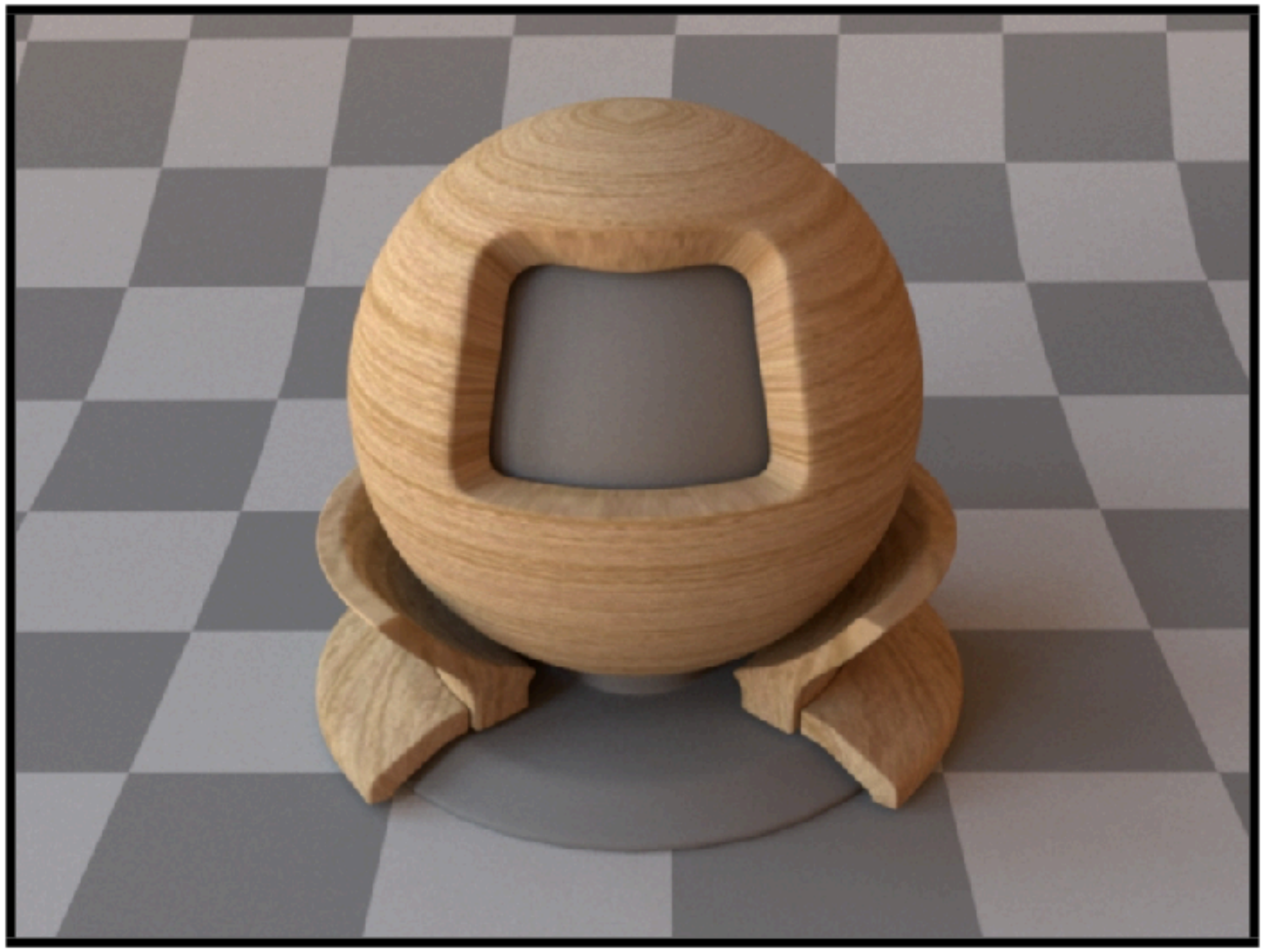
Light is scattered equally in all directions

Diffuse / Lambertian material





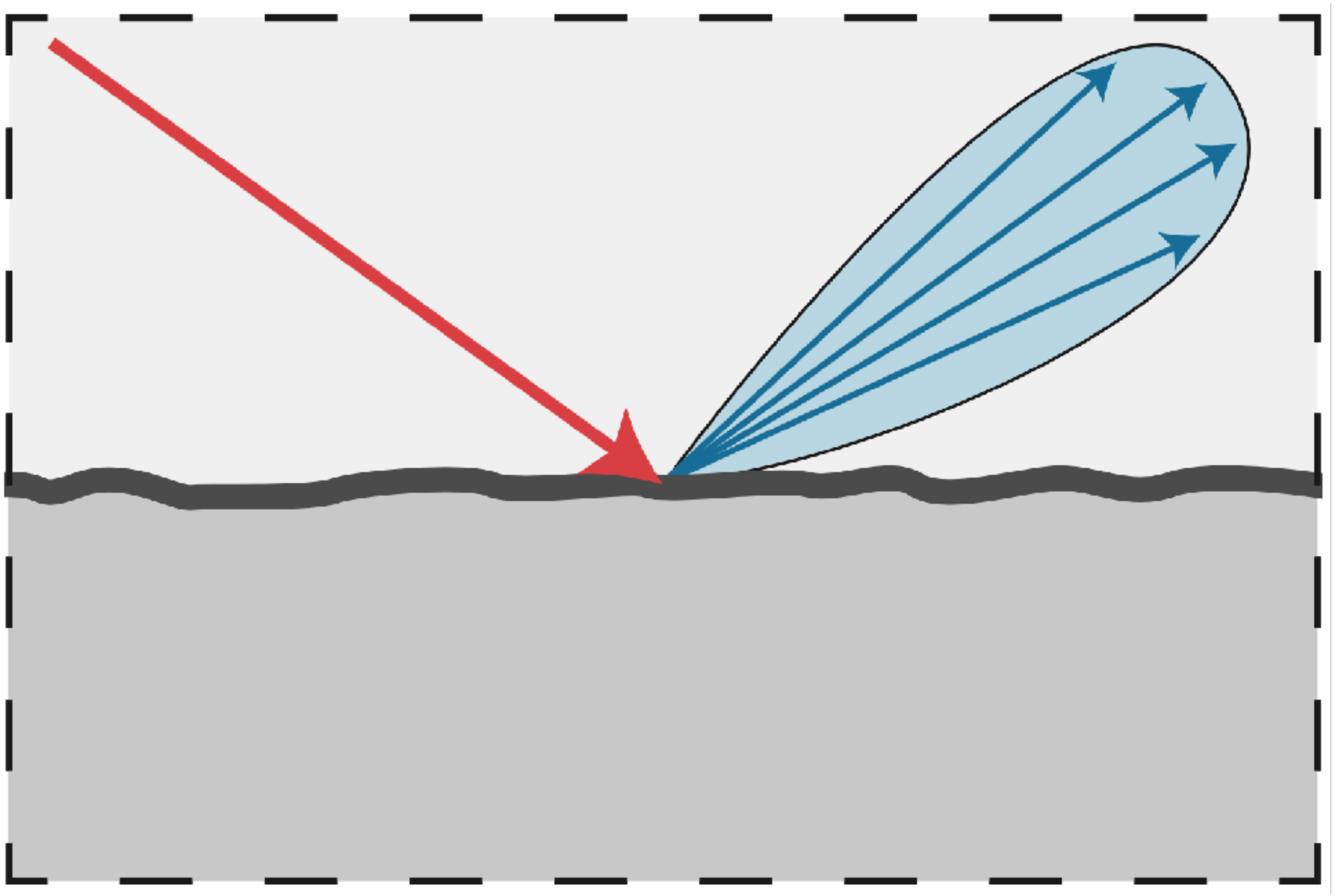
Uniform colored diffuse BRDF
Albedo (fraction of light reflected) is same for all surface points p



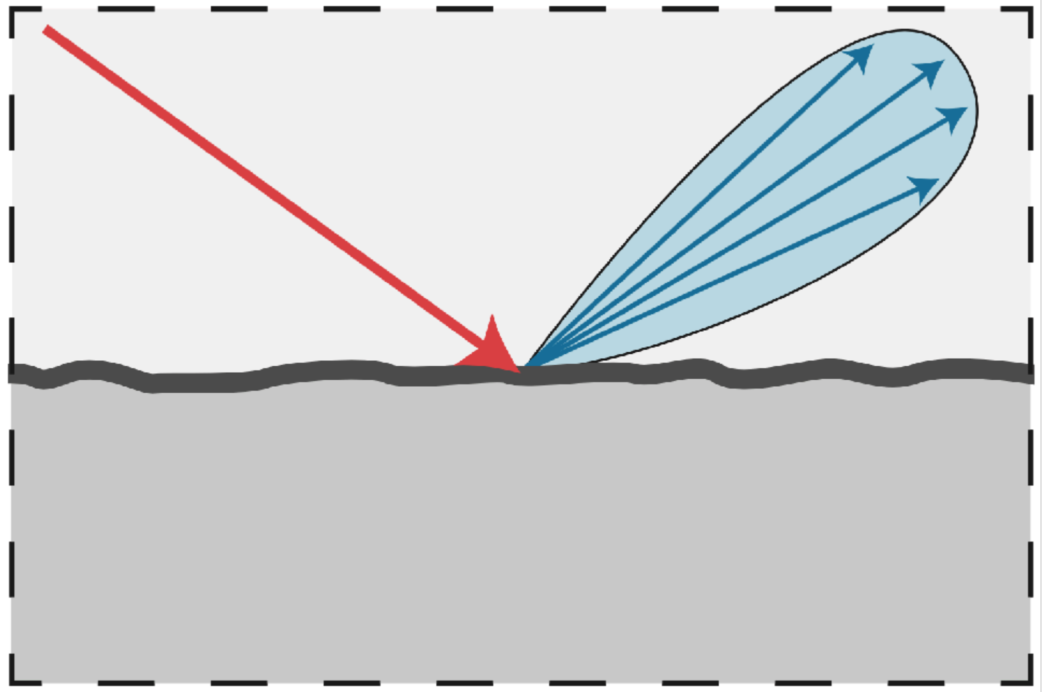
Textured diffuse BRDF
Albedo is spatially varying, and is encoded in texture map.

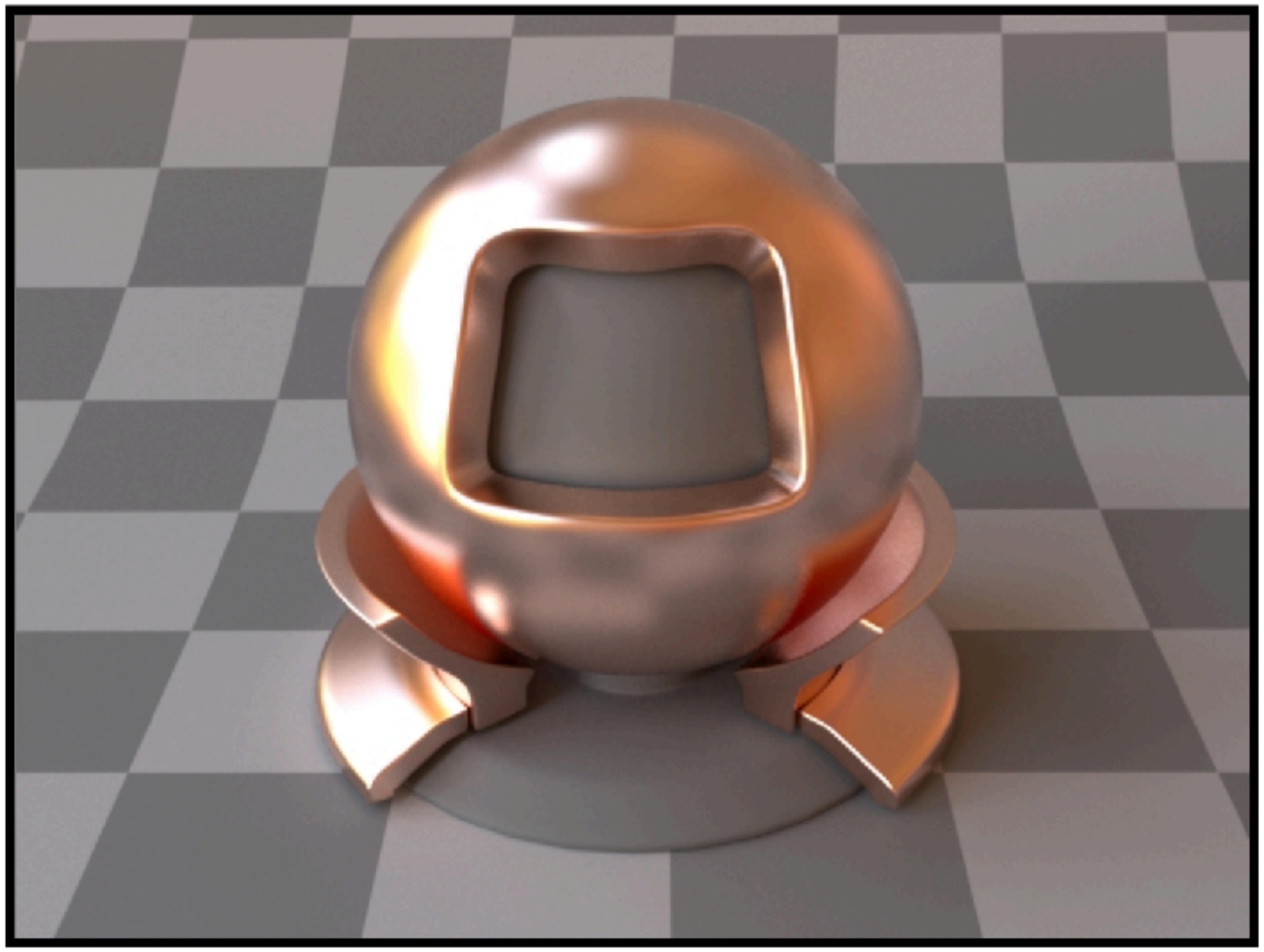
[Mitsuba renderer, Wenzel Jakob, 2010]

What is this material?

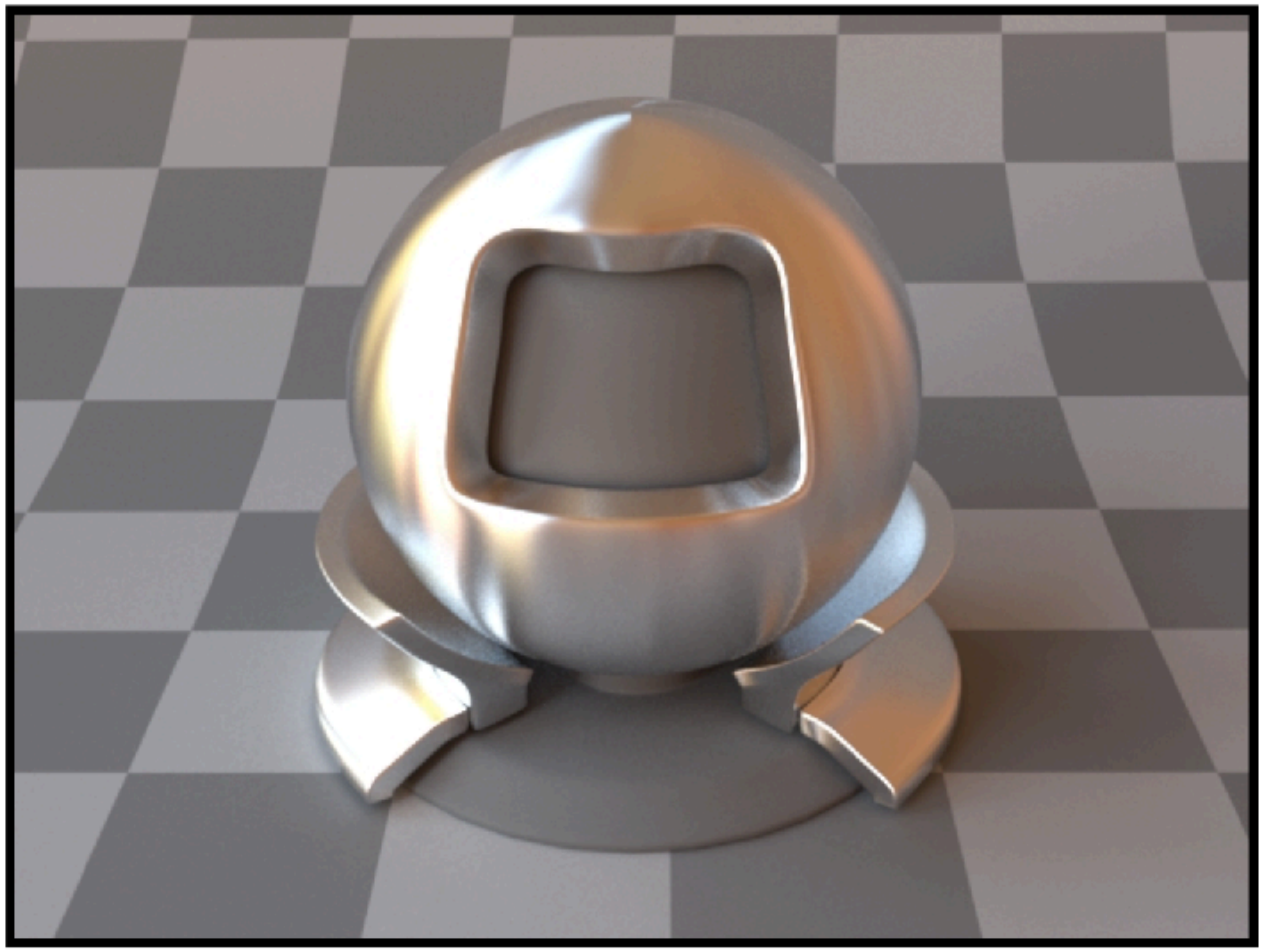


Glossy material (BRDF)





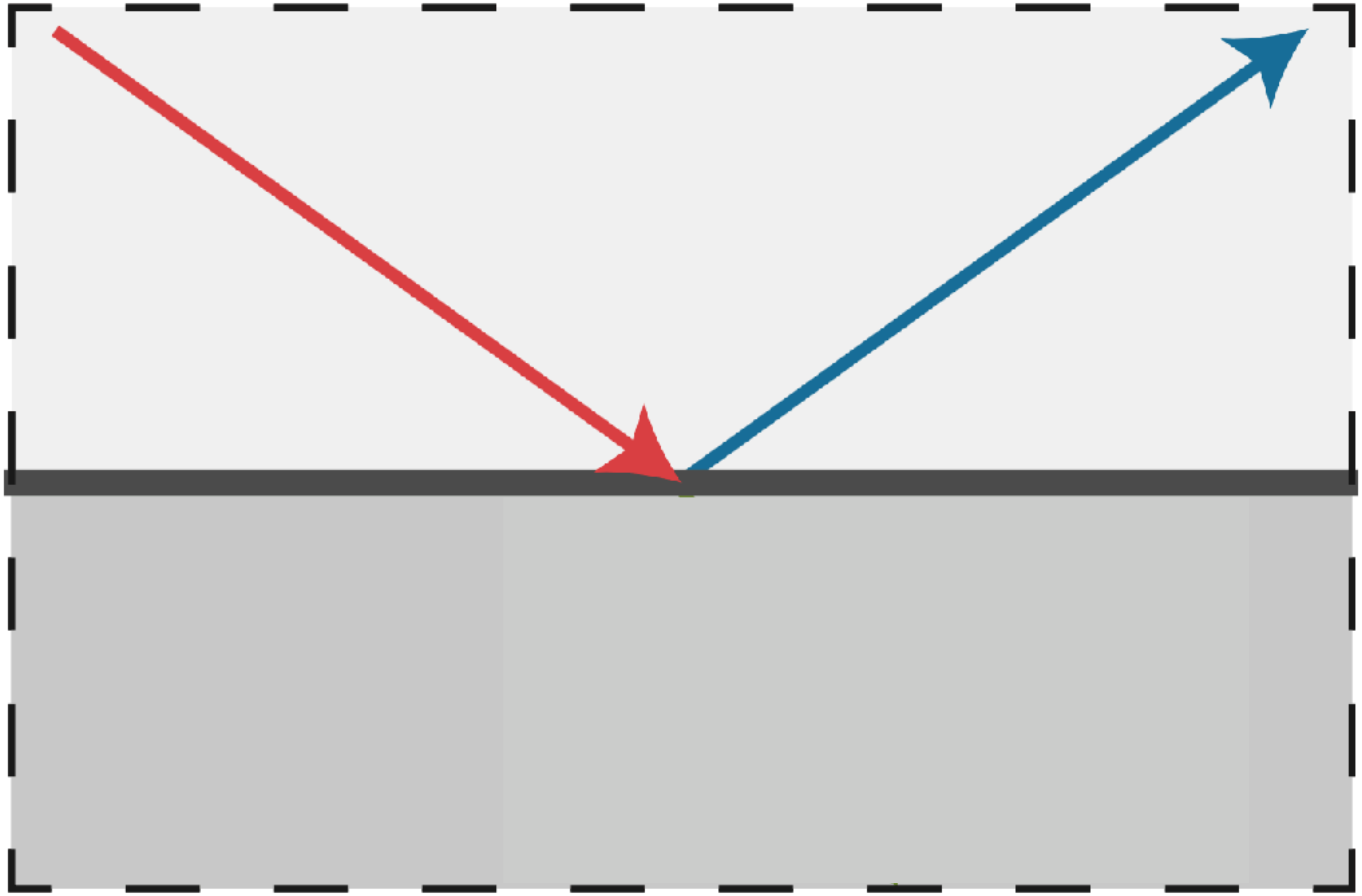
Copper



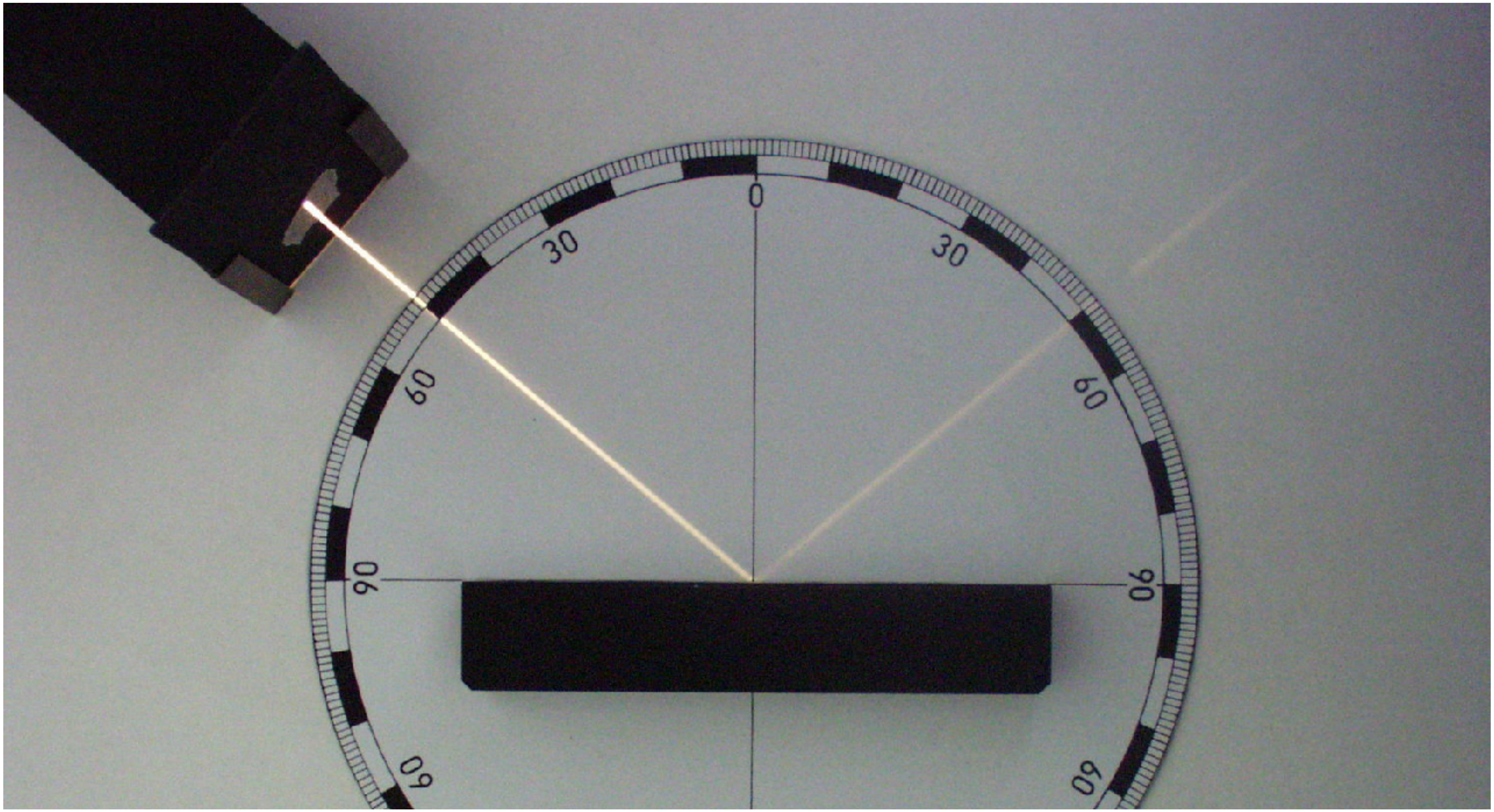
Aluminum

[Mitsuba renderer, Wenzel Jakob, 2010]

What is this material?

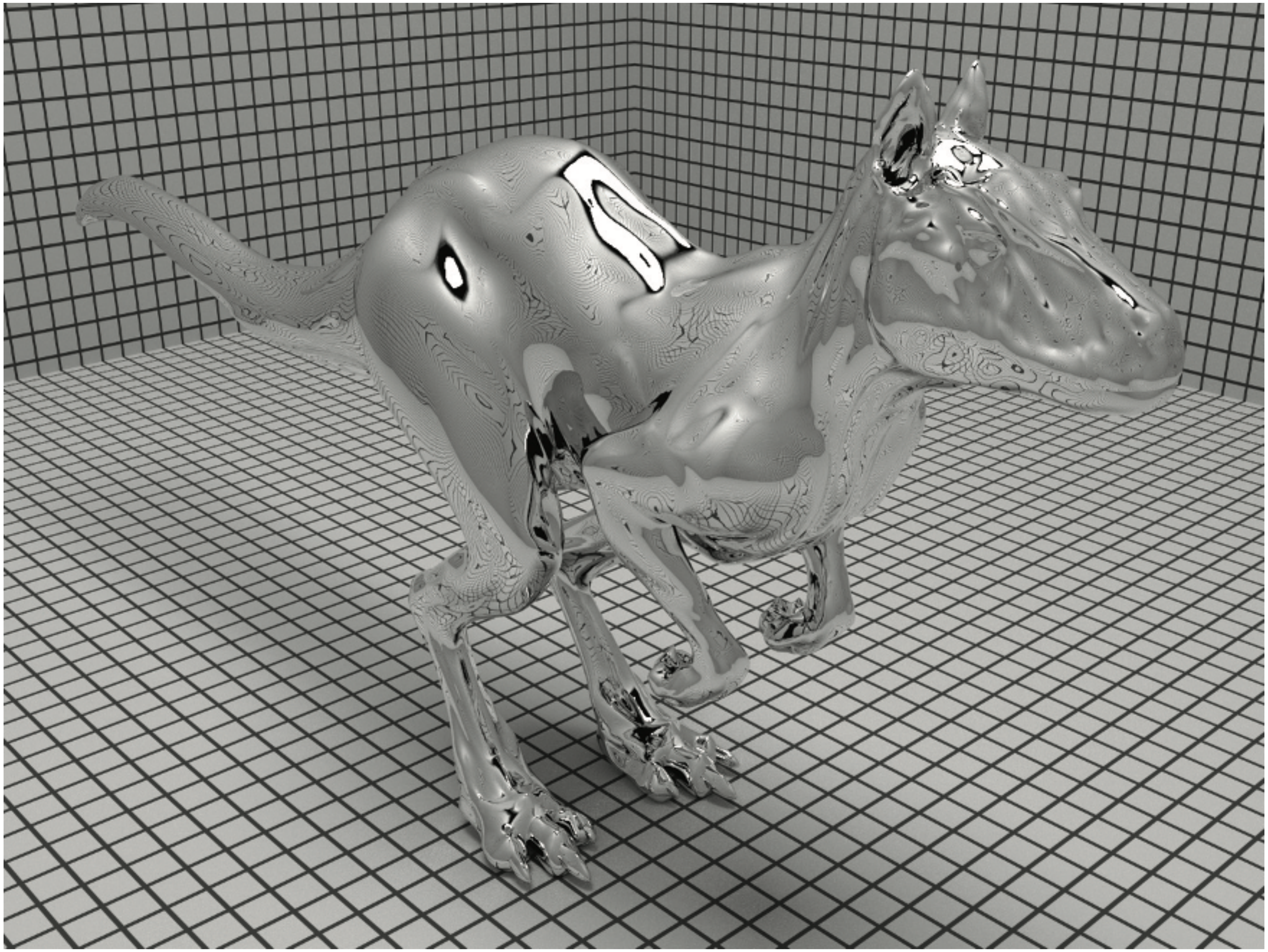


Perfect specular reflection



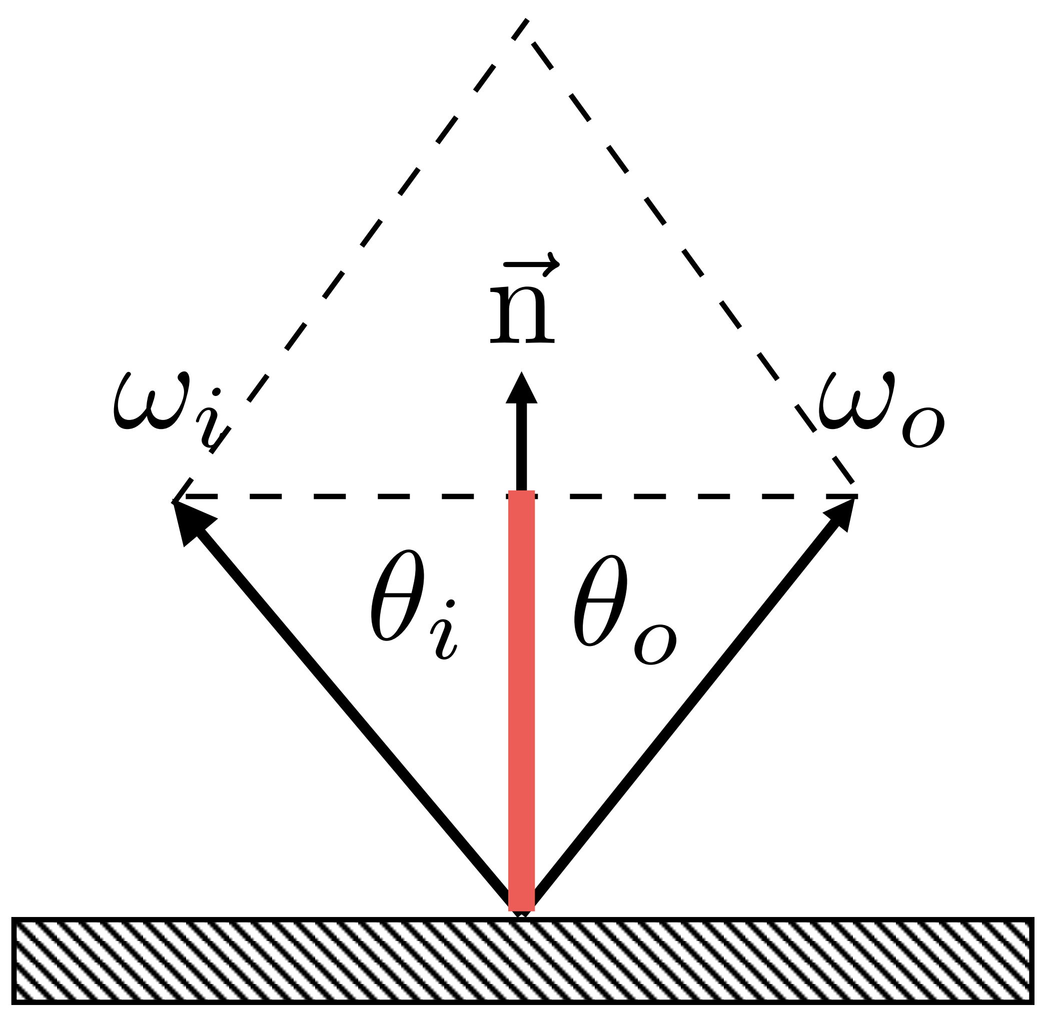
[Zátonyi Sándor]

Perfect specular reflection



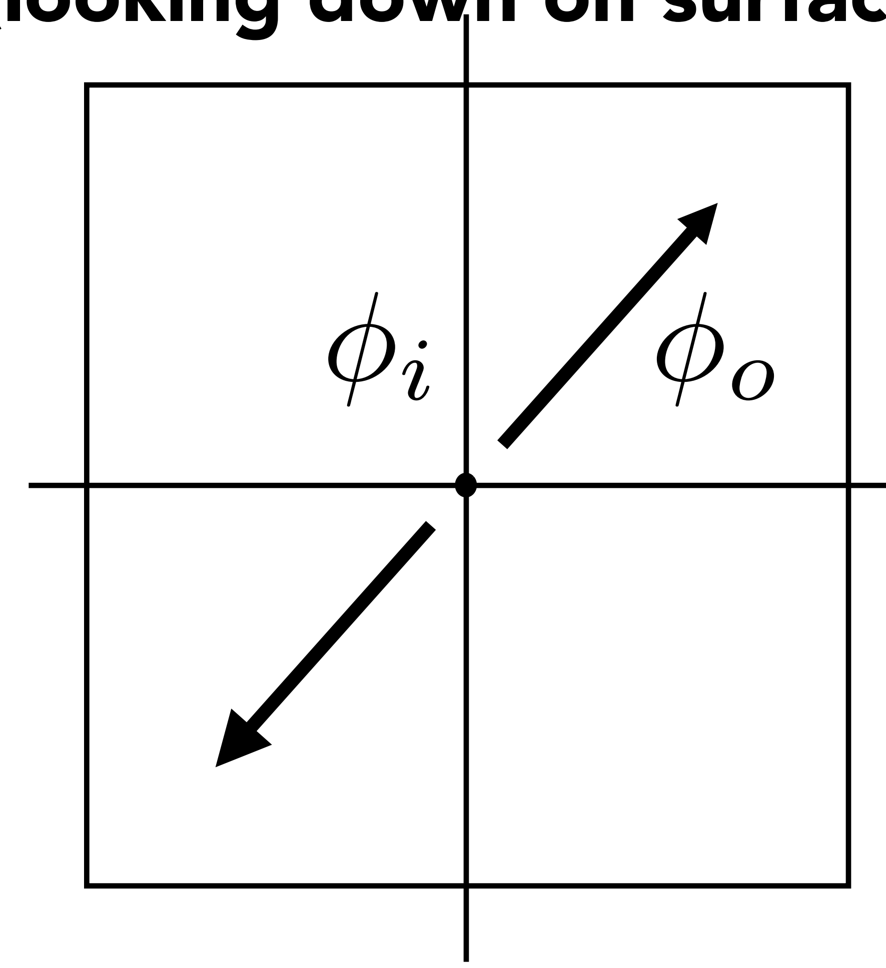
PBRT

Calculating direction of specular reflection



$$\theta = \theta_o = \theta_i$$

Top-down view
(looking down on surface)



$$\phi_o = (\phi_i + \pi) \bmod 2\pi$$

$$\omega_o + \omega_i = 2\cos\theta\,\vec{\mathrm{n}} = 2(\omega_i \cdot \vec{\mathrm{n}})\vec{\mathrm{n}}$$

$$\omega_o = -\omega_i + 2(\omega_i \cdot \vec{\mathrm{n}})\vec{\mathrm{n}}$$

How might you render a specular surface

- Compute direction from surface point *p* to camera = w_o
- Given normal at *p*, compute reflection direction w_i
- Light reflected in direction w_o is light arriving from direction w_i
- How do you measure light arriving from w_i?

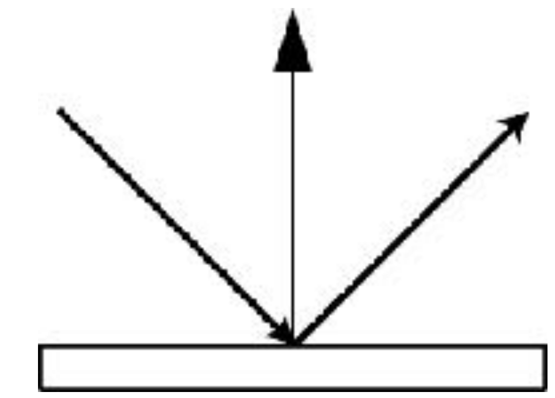
One idea…
look up amount in environment map!
(more on this later)

Pixel (x,y) stores radiance L from direction (ϕ, θ)

Some basic reflection functions

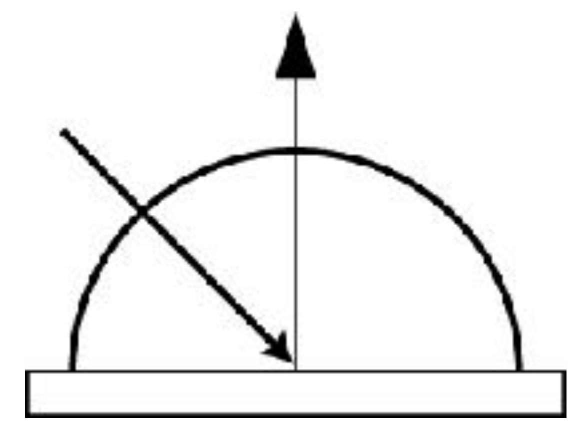
- **Ideal specular**

  Perfect mirror



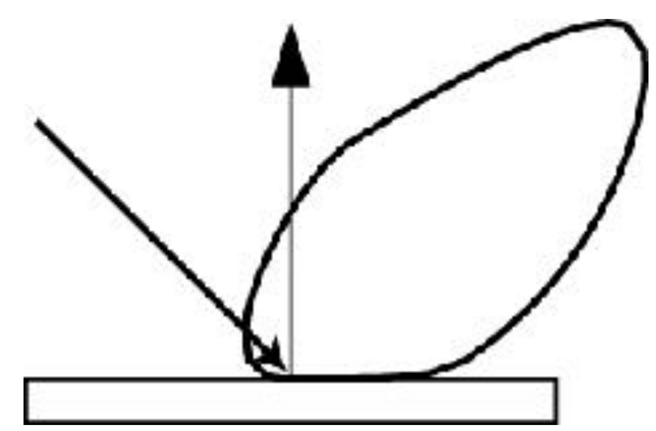
- **Ideal diffuse**

  Uniform reflection in all directions



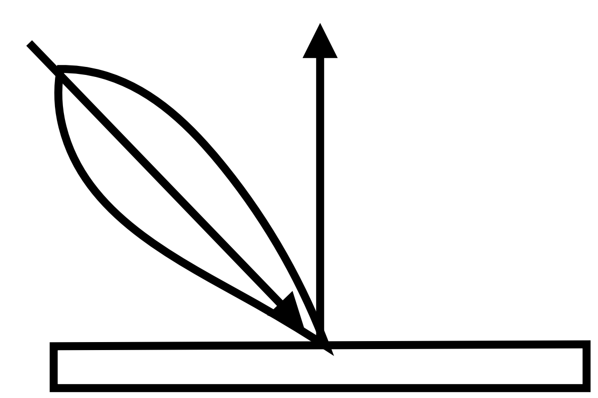

- **Glossy specular**

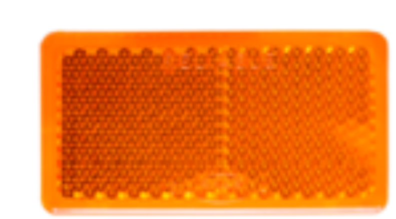
  Majority of light distributed in reflection direction



- **Retro-reflective**

  Reflects light back toward source



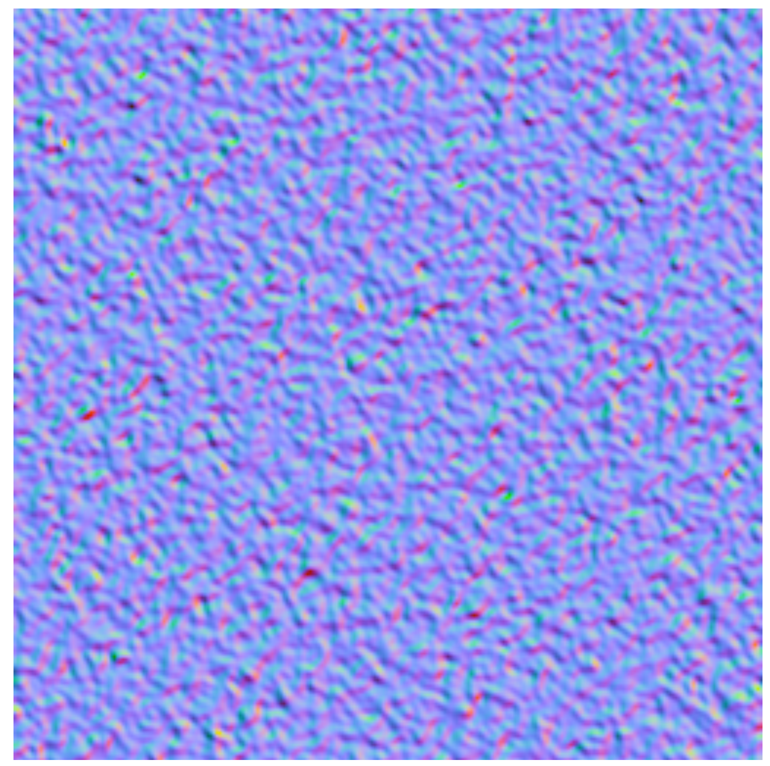
Diagrams illustrate how incoming light energy from given direction is reflected in various directions.

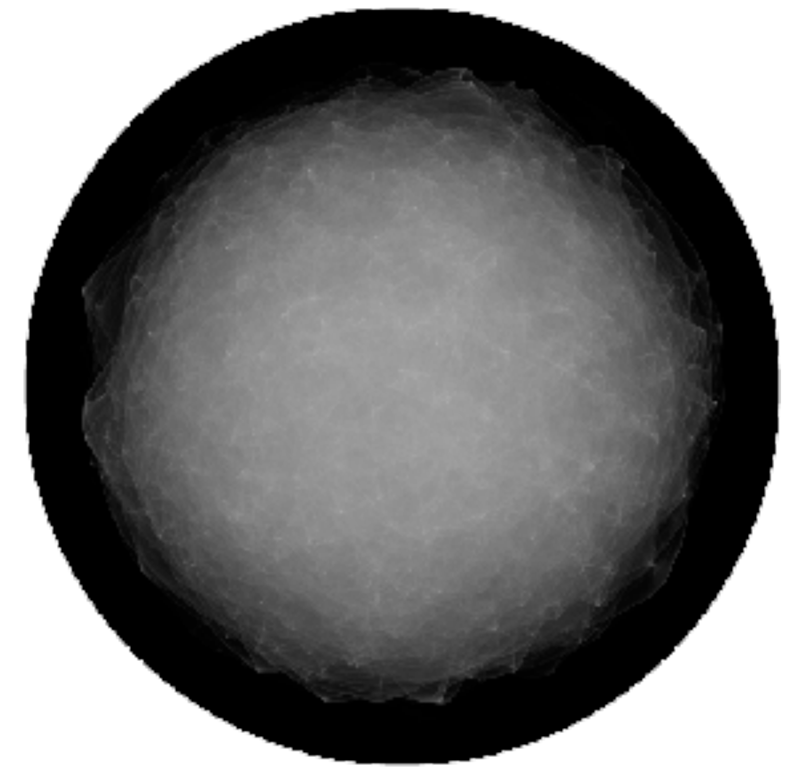
More complex materials

Isotropic / anisotropic materials (BRDFs)

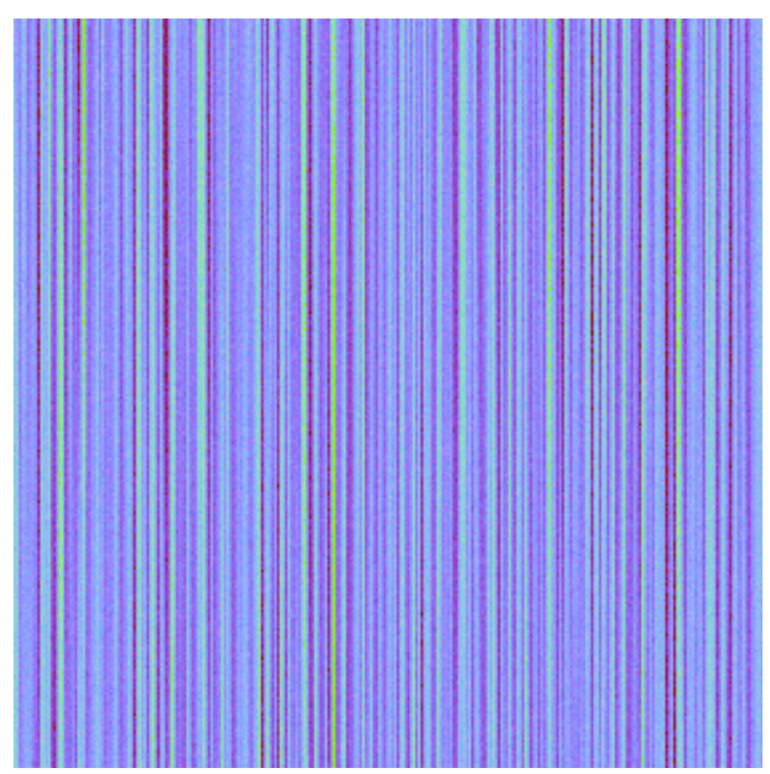
- Key: **directionality** of underlying surface

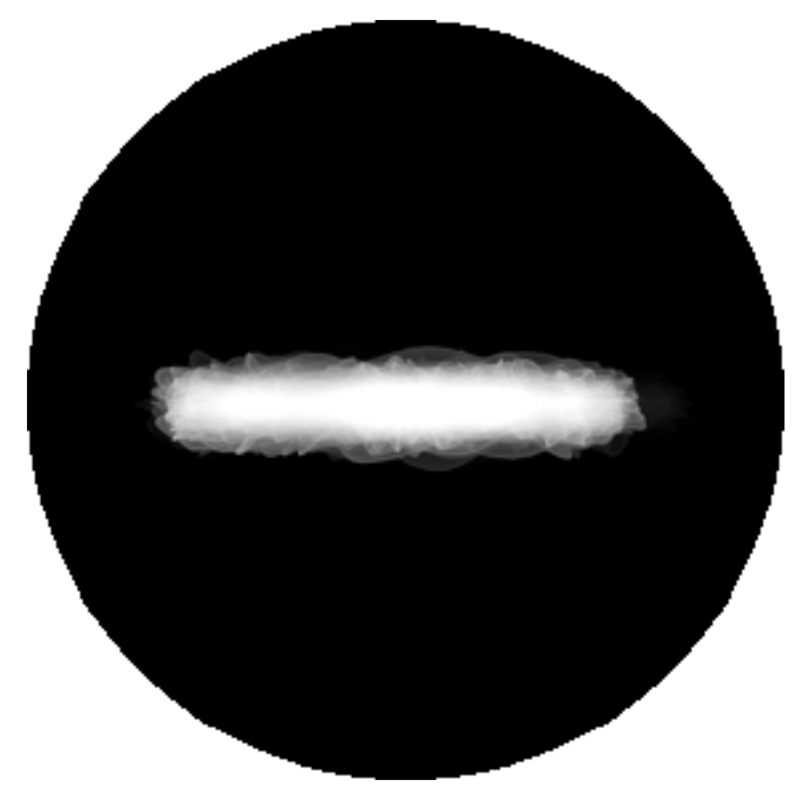
Isotropic





Anisotropic





Surface (normals)

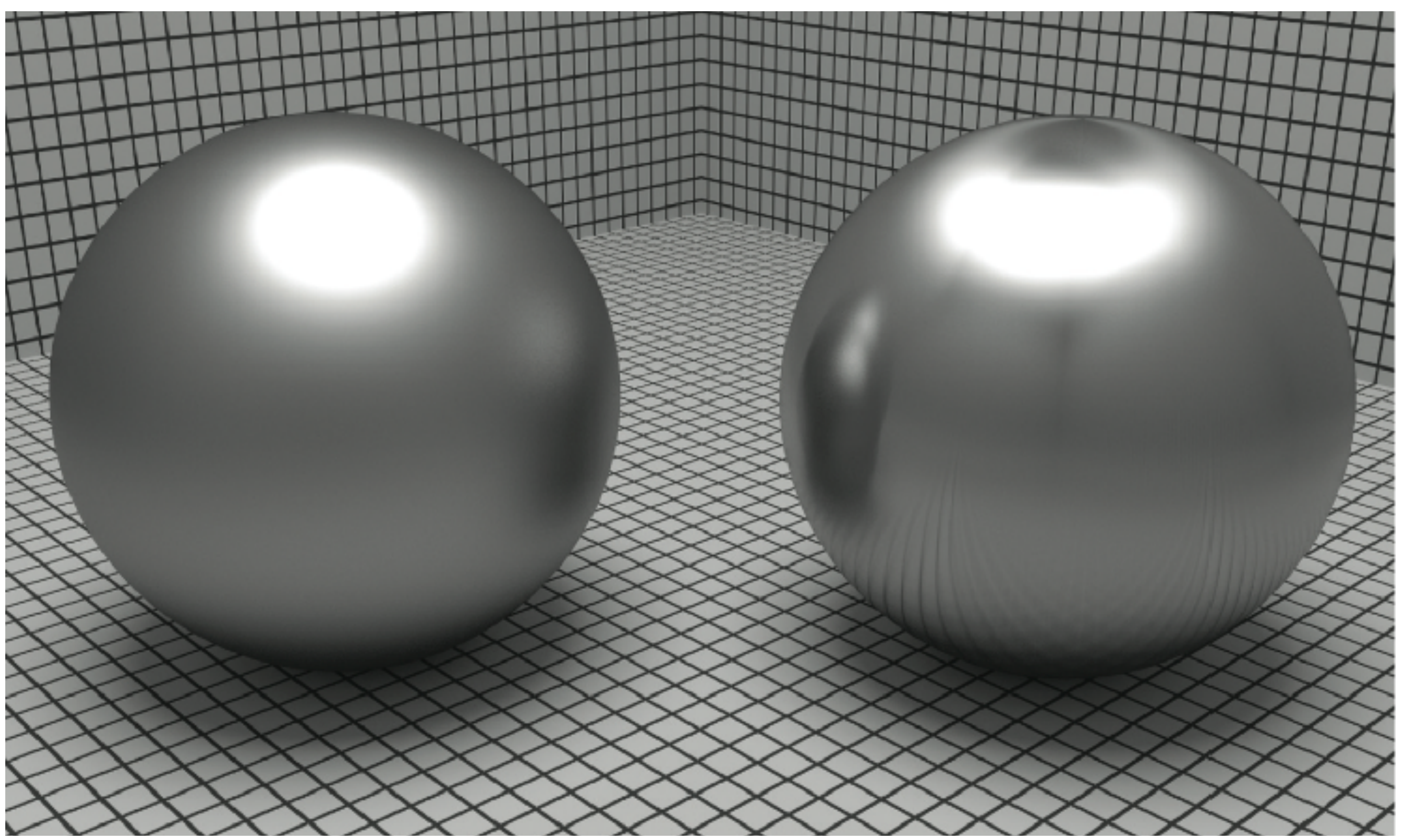
BRDF (fix wi, vary wo)

Anisotropic BRDFs

Reflection depends on azimuthal angle ϕ

$$f_r(\theta_i, \phi_i; \theta_r, \phi_r) \neq f_r(\theta_i, \theta_r, \phi_r - \phi_i)$$

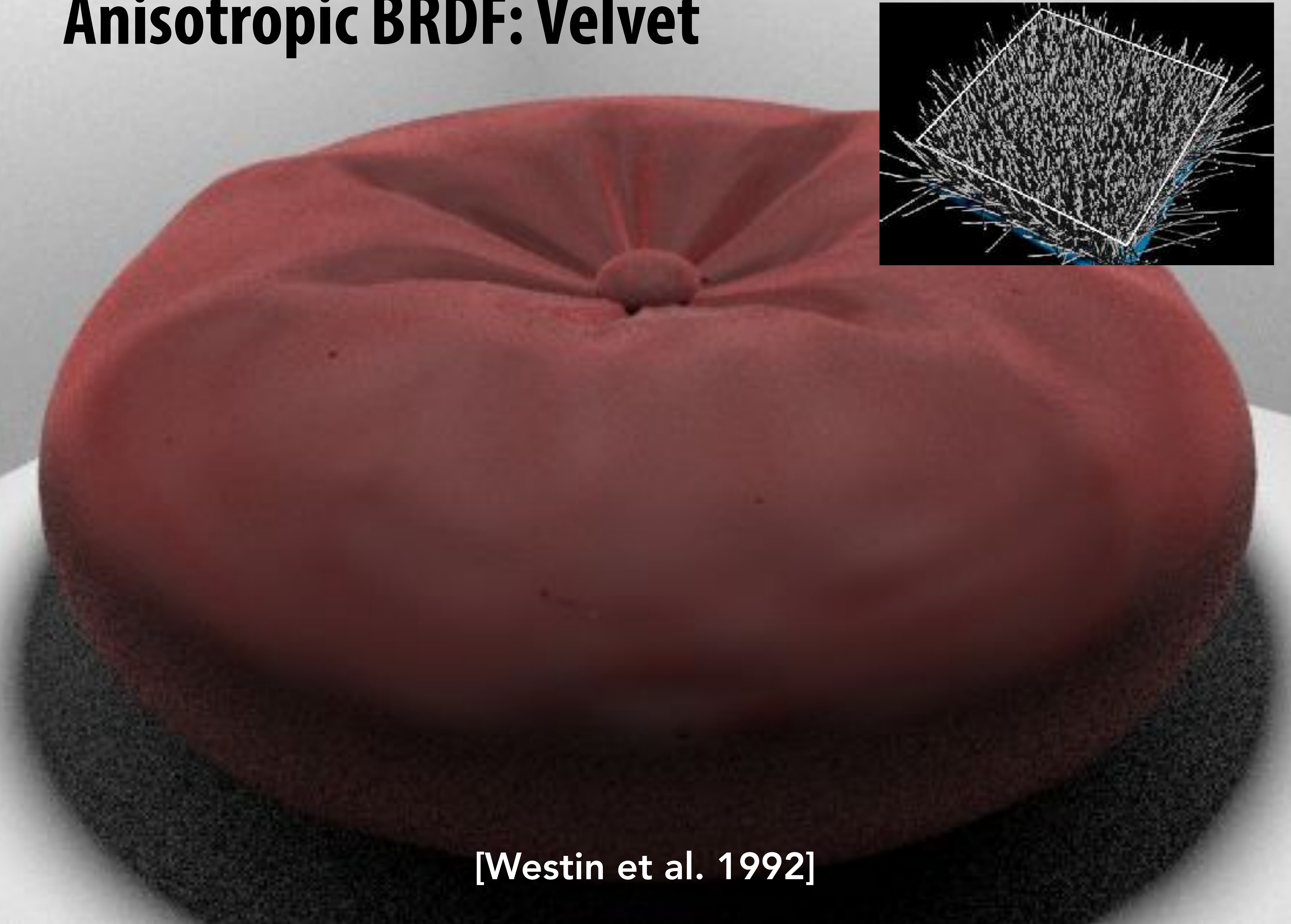
Results from oriented microstructure of surface, e.g., brushed metal



Anisotropic BRDF: Nylon

[Westin et al. 1992]

Anisotropic BRDF: Velvet

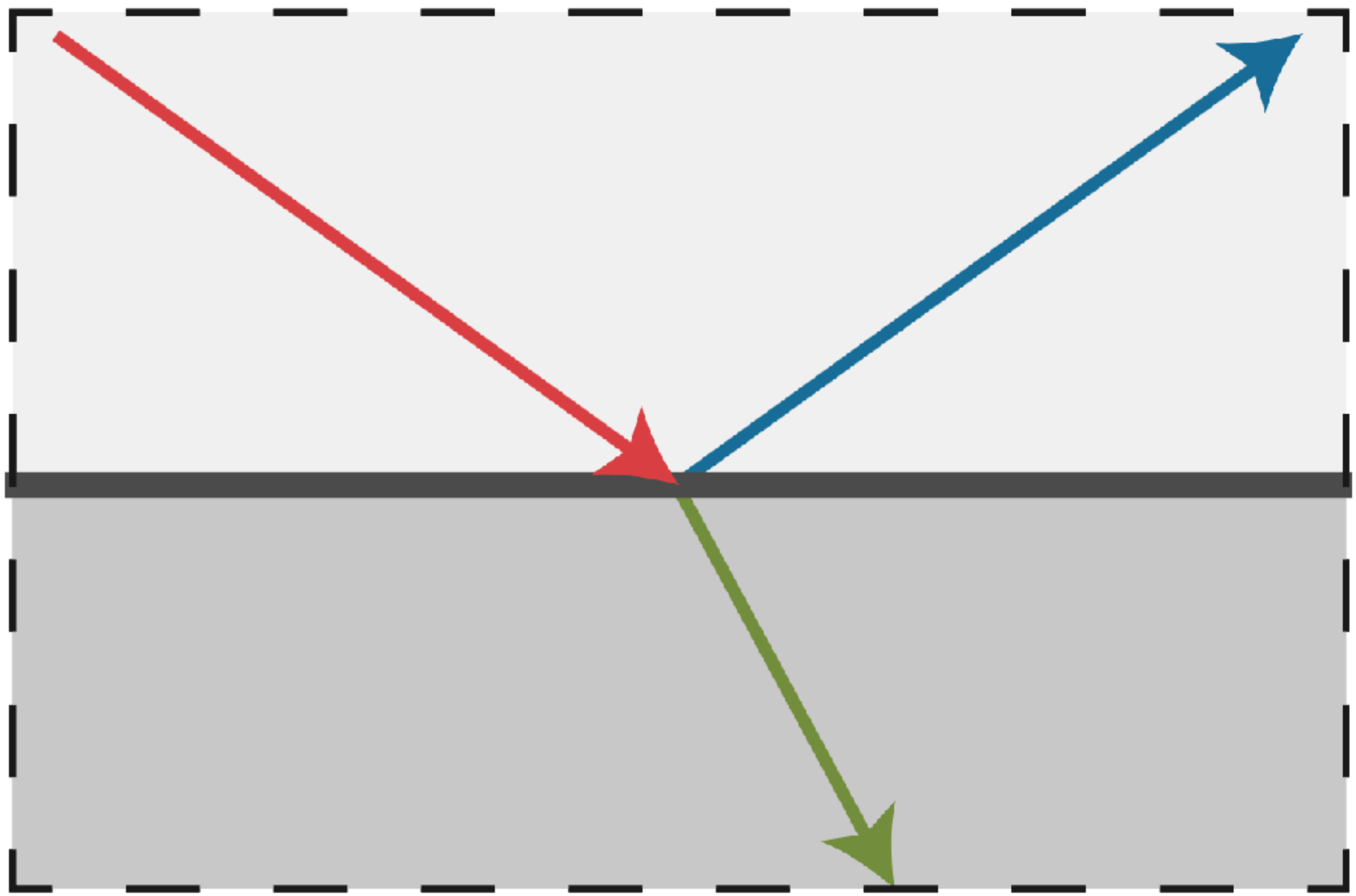


[Westin et al. 1992]

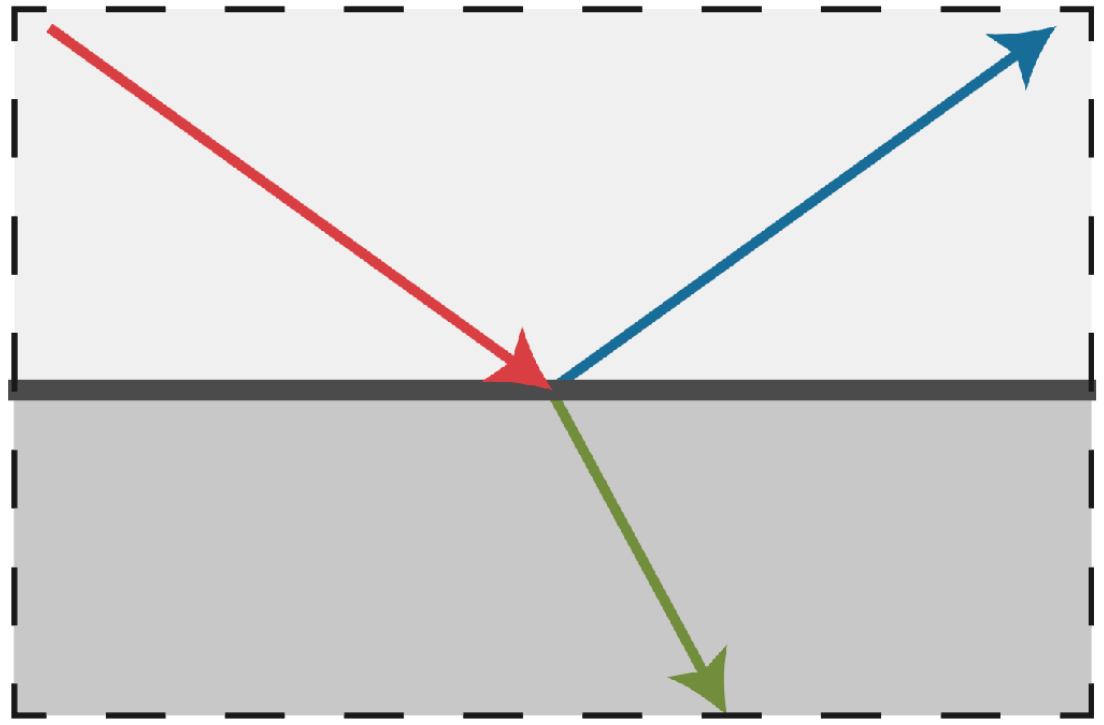
Anisotropic BRDF: Velvet

[https://www.youtube.com/watch?v=2hjoW8TYTd4]

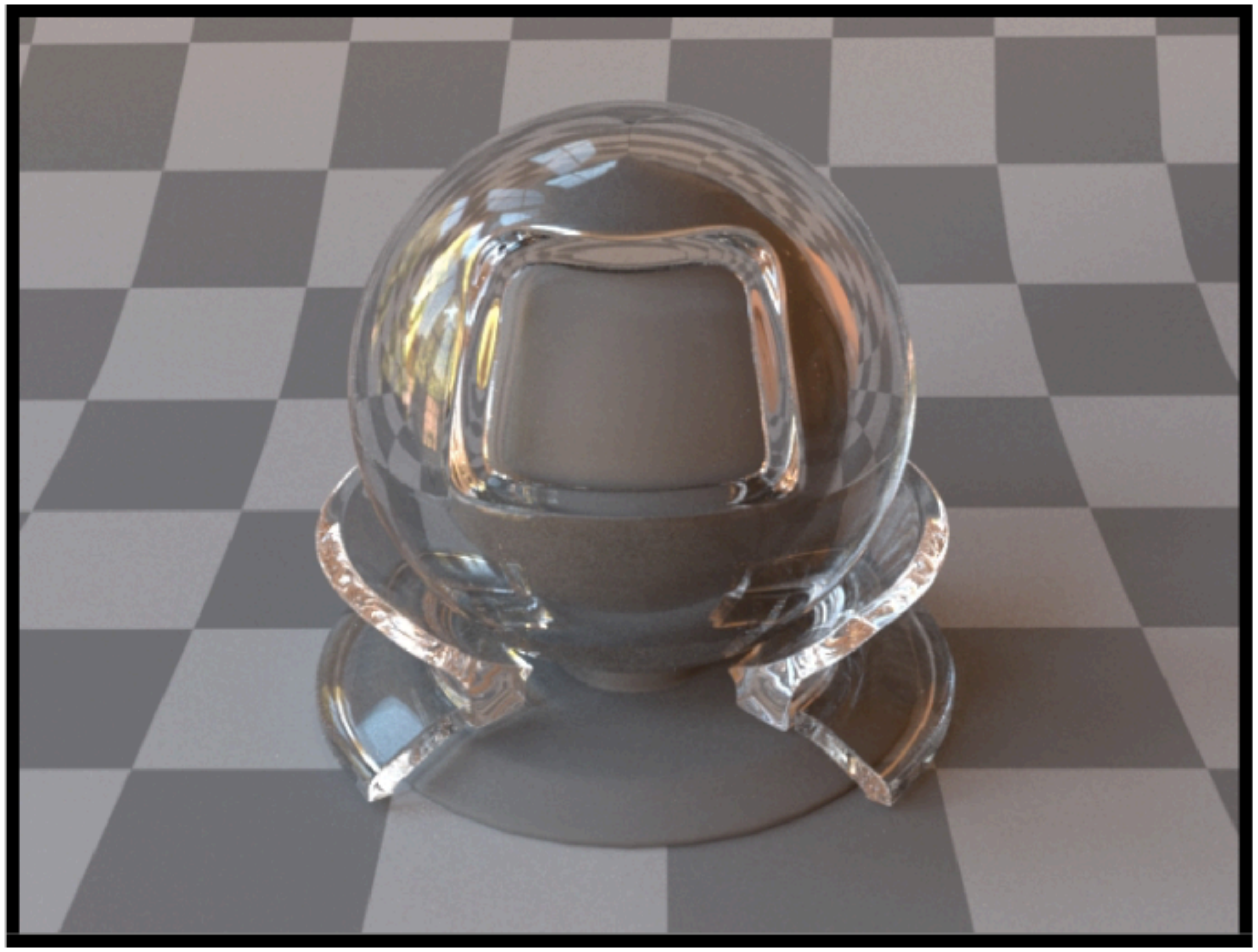
What is this material?



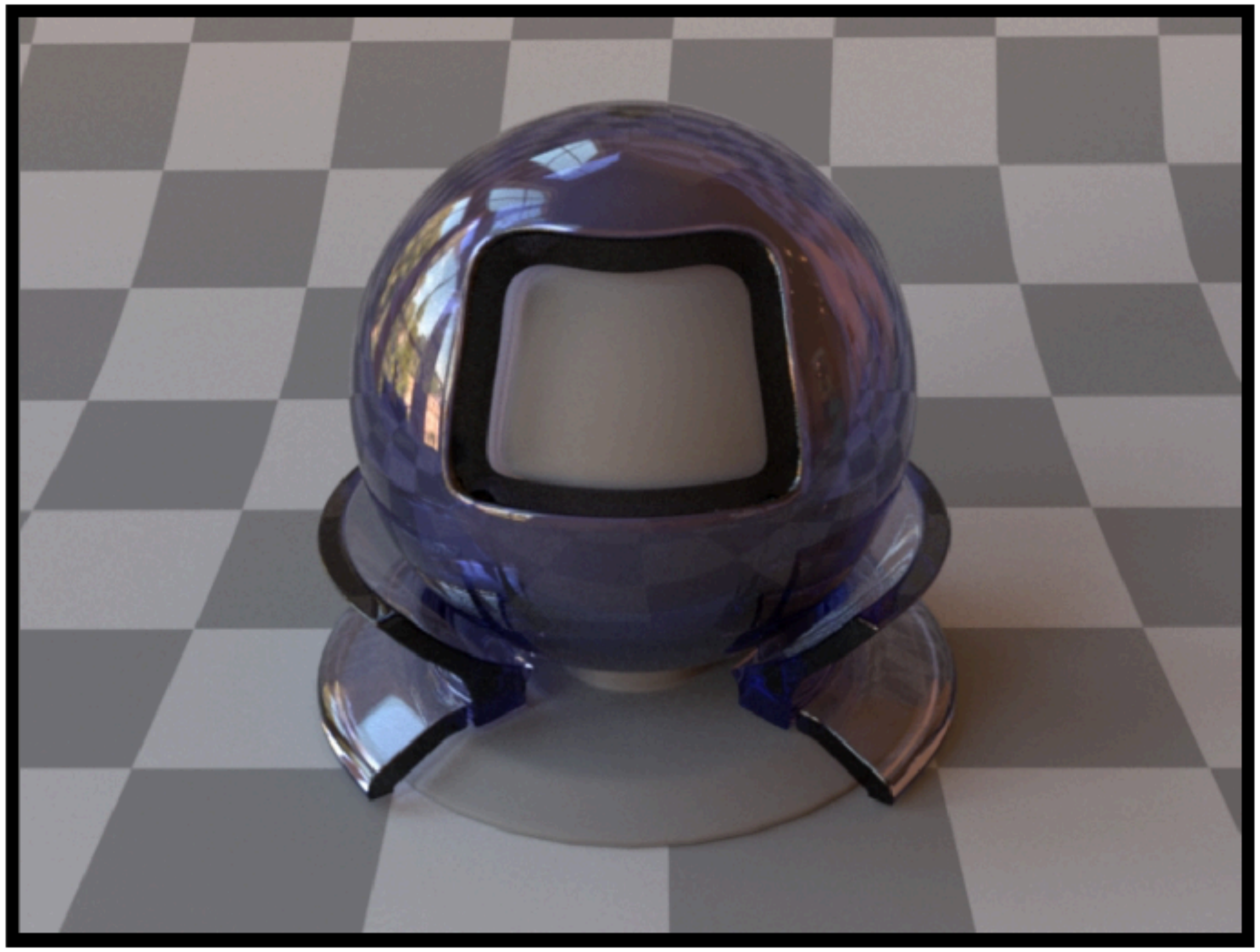
Ideal reflective / refractive material (BxDF *)



[Mitsuba renderer, Wenzel Jakob, 2010]



Air <-> water interface



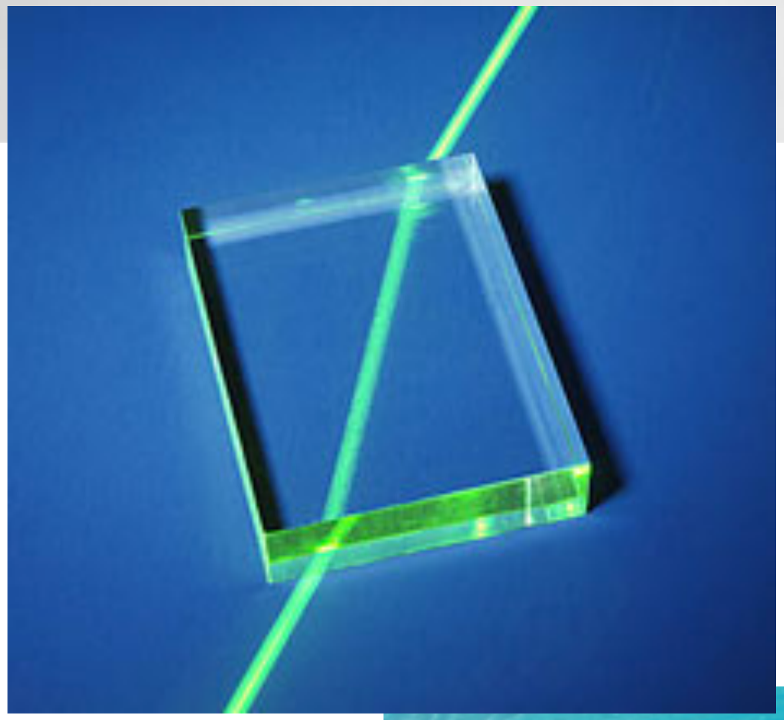
Air <-> glass interface
(with absorption)

* X stands in for reflectance "r", scattering, transmission, etc.

Transmission

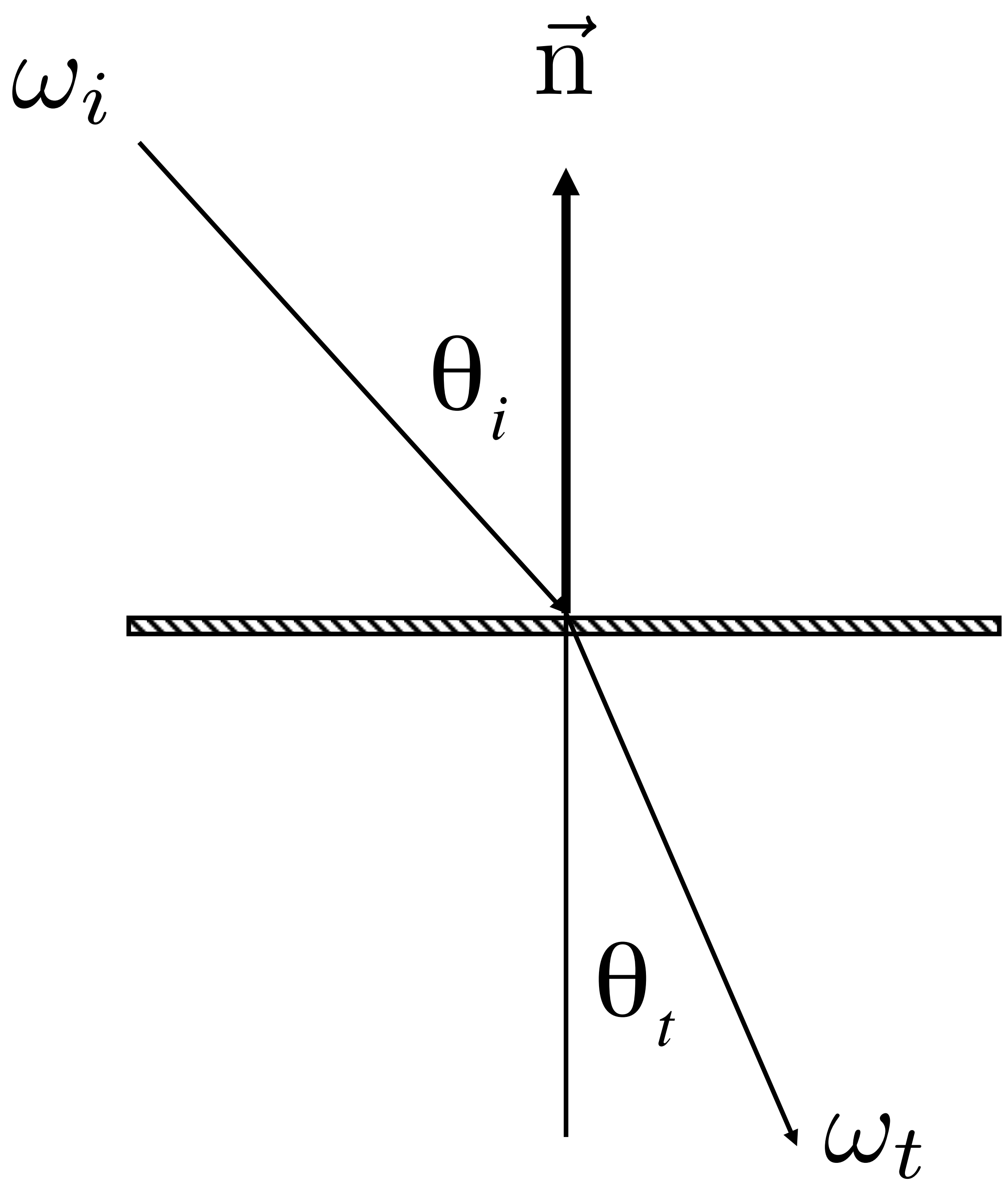
In addition to reflecting off surface, light may be transmitted through surface.

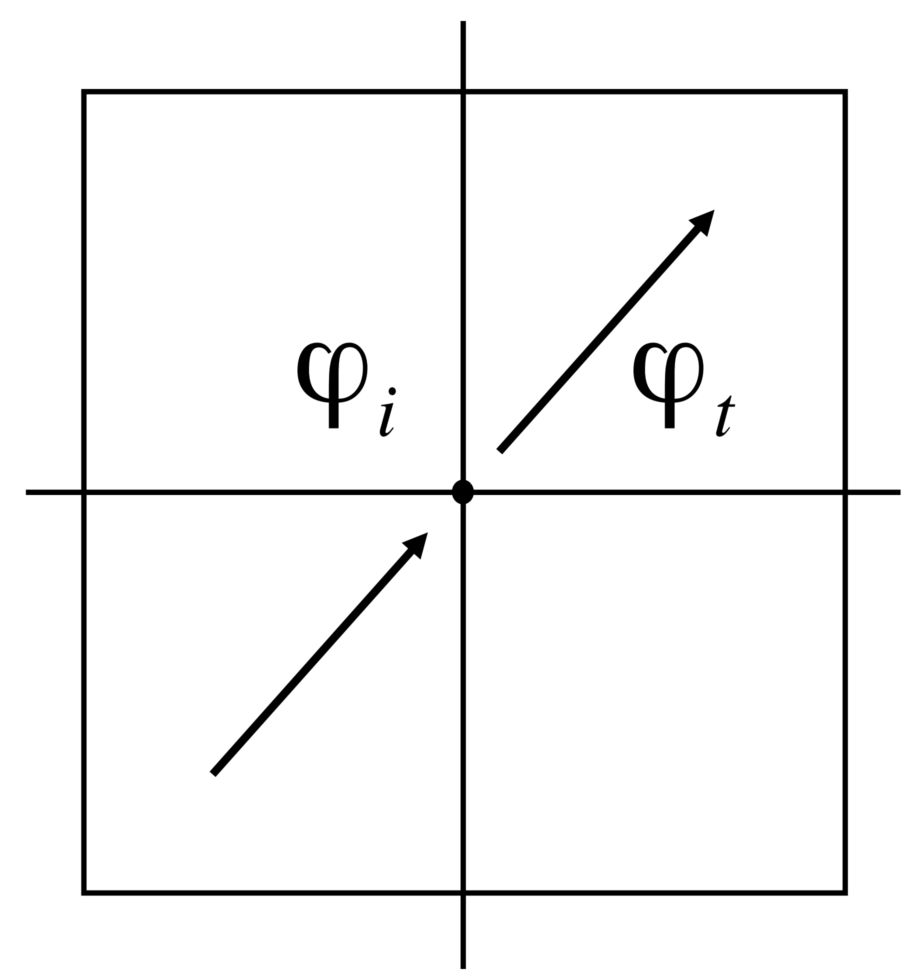
Light refracts when it enters a new medium.



Snell's Law

Transmitted angle depends on index of refraction of medium incident ray is in and index of refraction of medium light is entering.





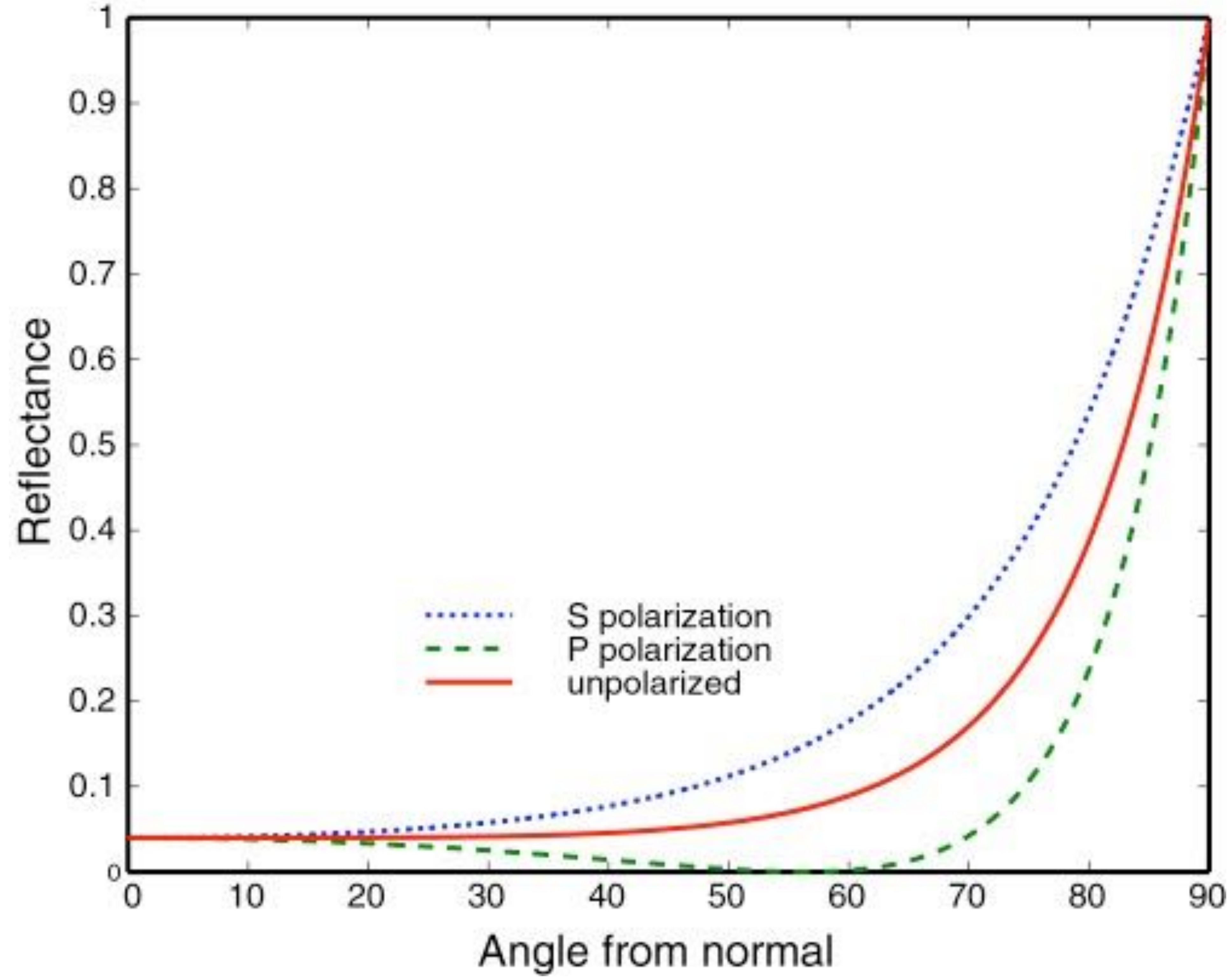
$$\eta_i \sin \theta_i = \eta_t \sin \theta_t$$

Medium	η*
Vacuum	1.0
Air (sea level)	1.00029
Water (20°C)	1.333
Glass	1.5-1.6
Diamond	2.42

* index of refraction is wavelength dependent (these are averages)

Fresnel reflection

Many real materials: reflectance increases w/ viewing angle

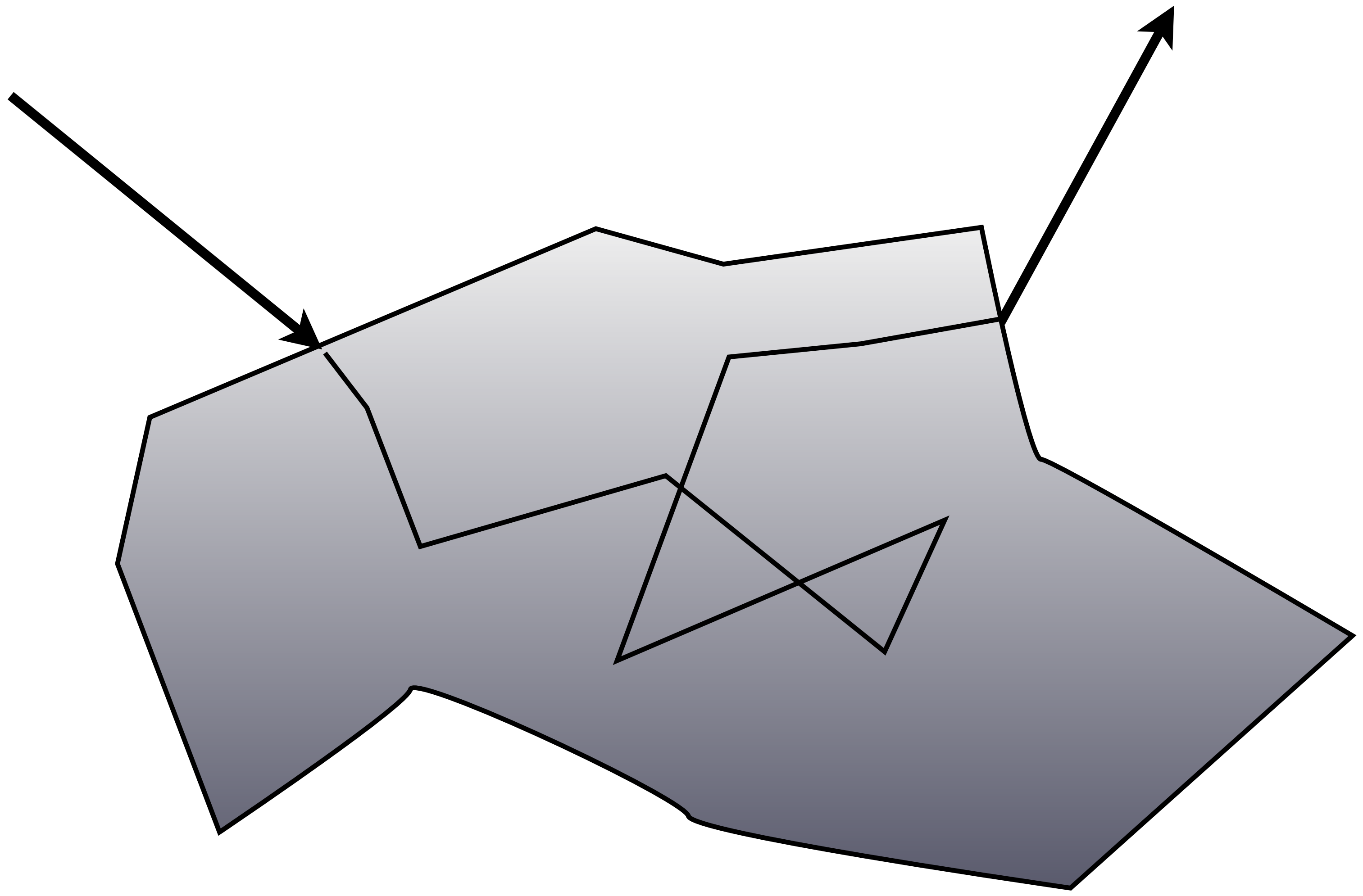


[Lafortune et al. 1997]

Snell + Fresnel: example

Subsurface scattering

- Visual characteristics of many surfaces caused by light entering at different points than it exits
  - Violates a fundamental assumption of the BRDF
  - Need to generalize scattering model (BSSRDF)



[Jensen et al 2001]

[Donner et al 2008]

* BSSRDF = bidirectional subsurface scatting reflectance distribution function

Translucent materials: Jade

Translucent materials: skin

Translucent materials: leaves

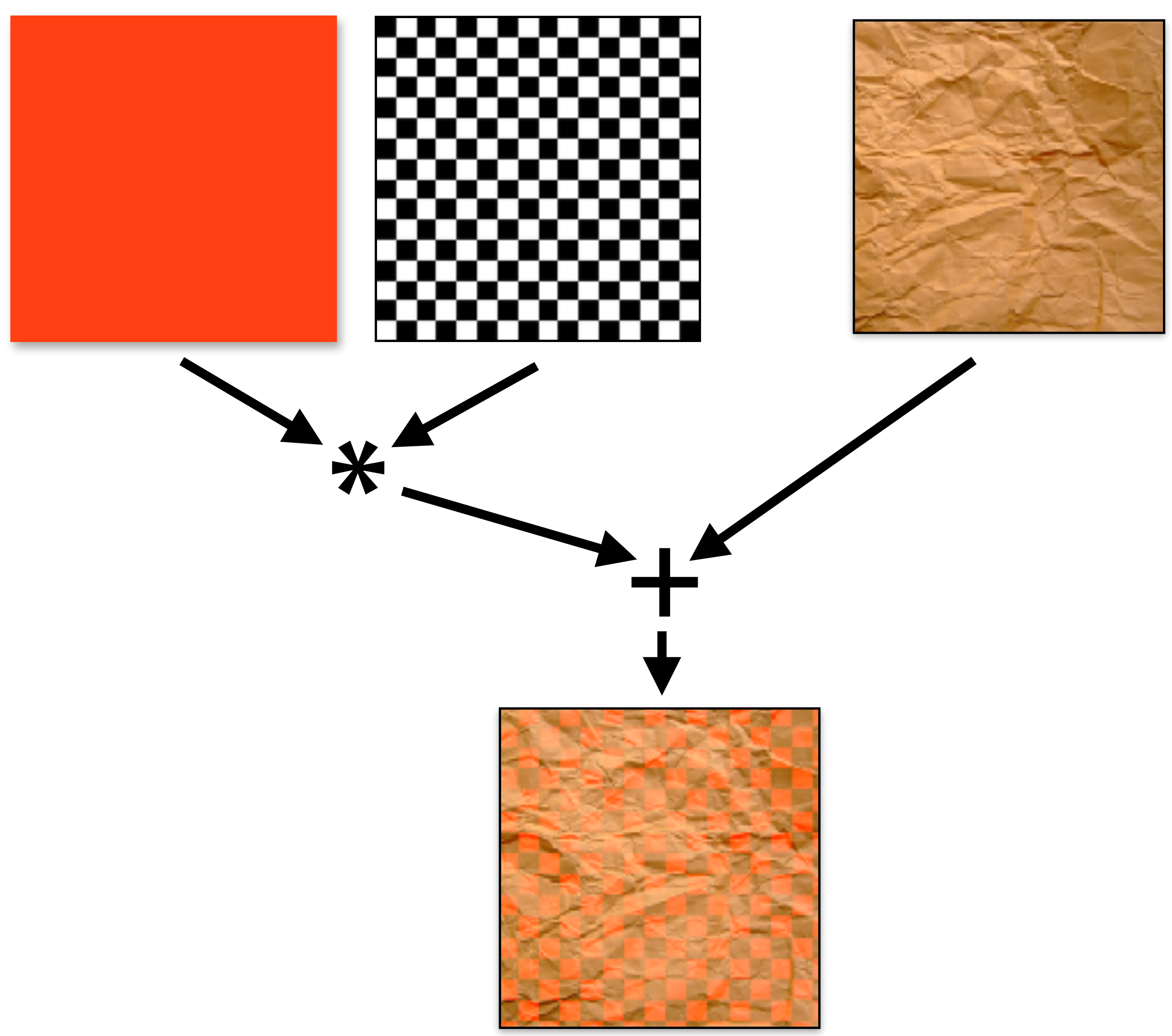
BRDF

BSSRDF

(models subsurface scattering of light)

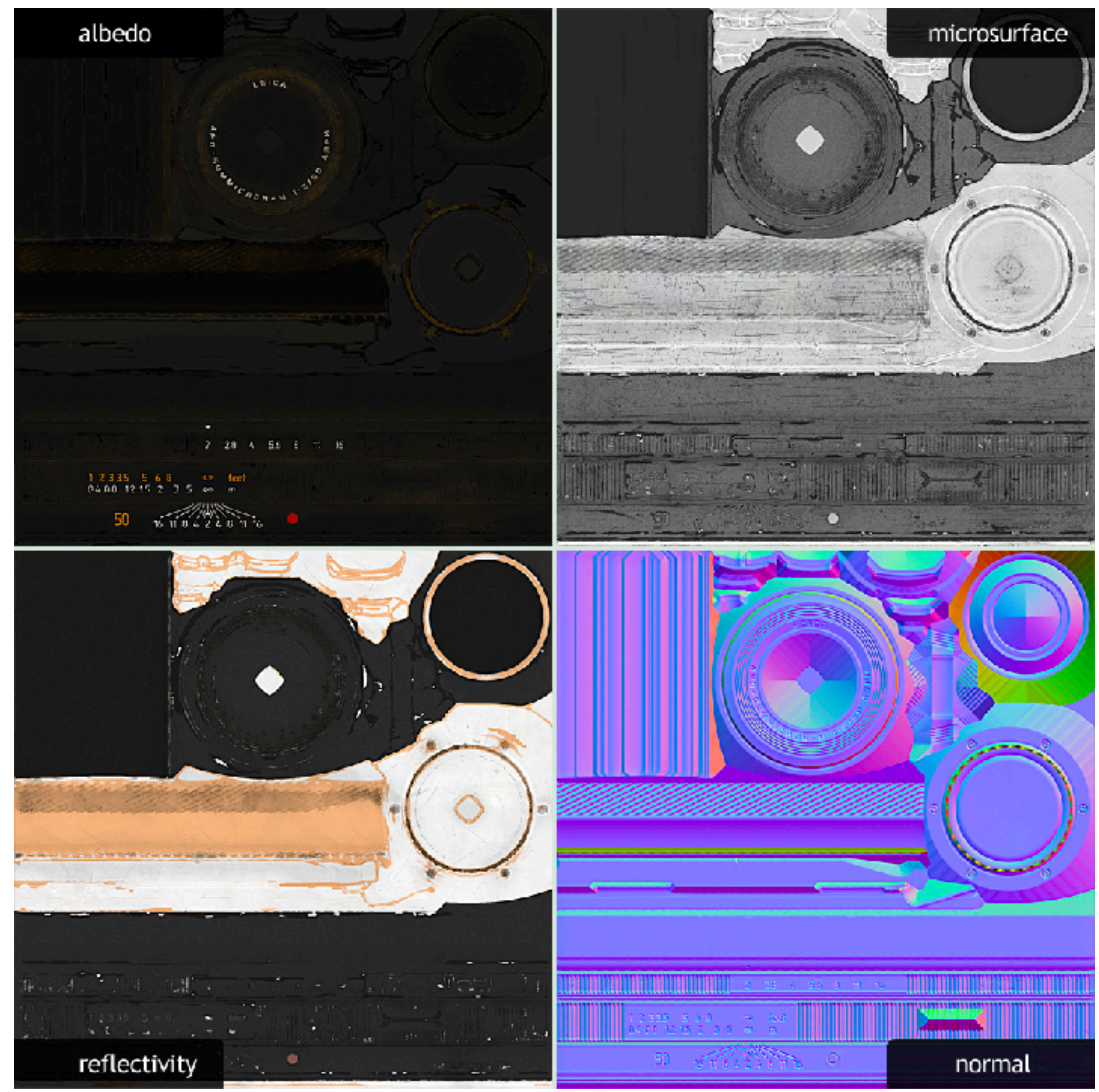
Pattern generation vs. BRDF

In practice, it is convenient to separate computation of spatially varying BRDF parameters (like albedo) from the reflectance function itself

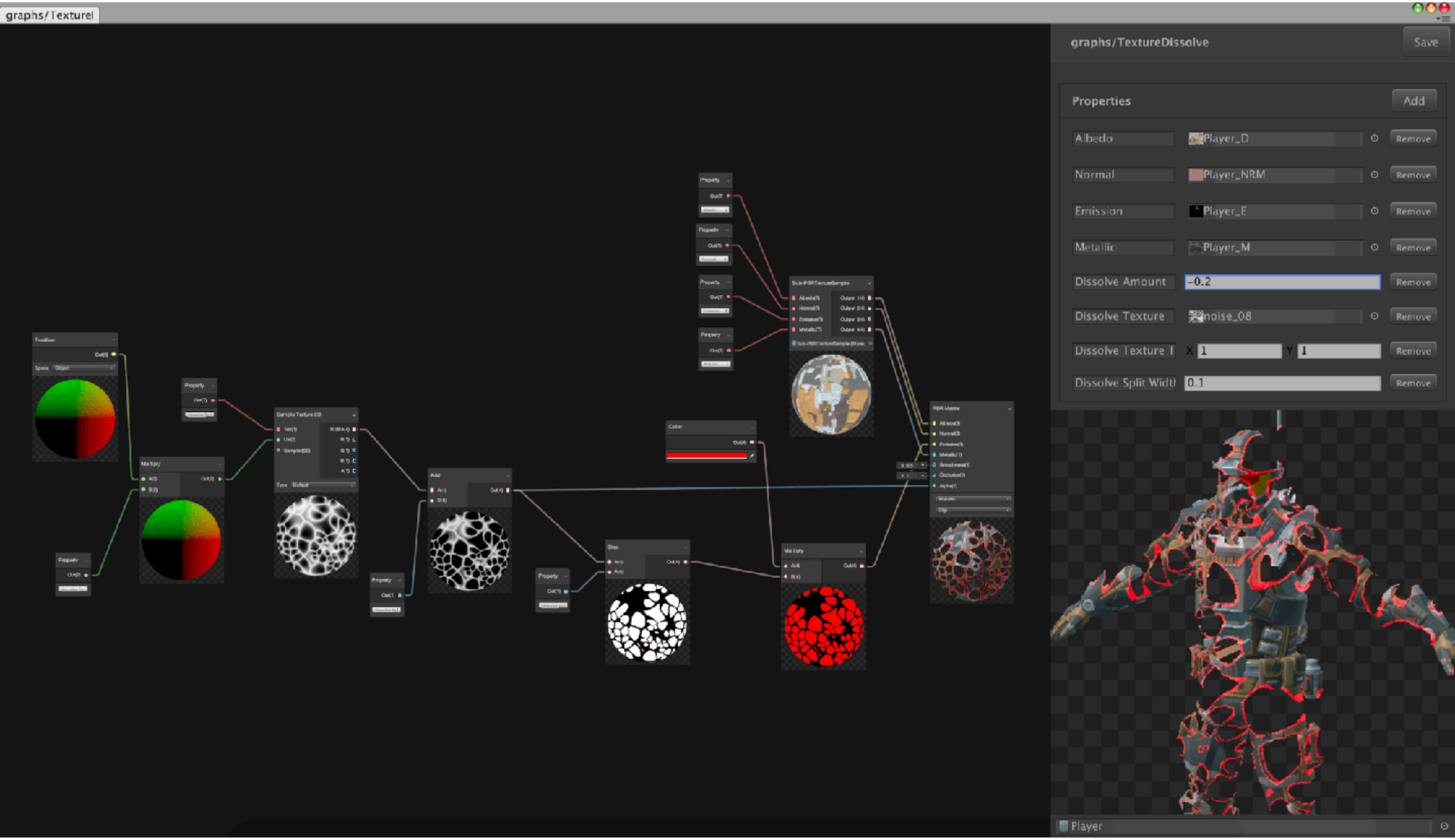


Example 1: albedo value at surface point is given by expression combining multiple textures

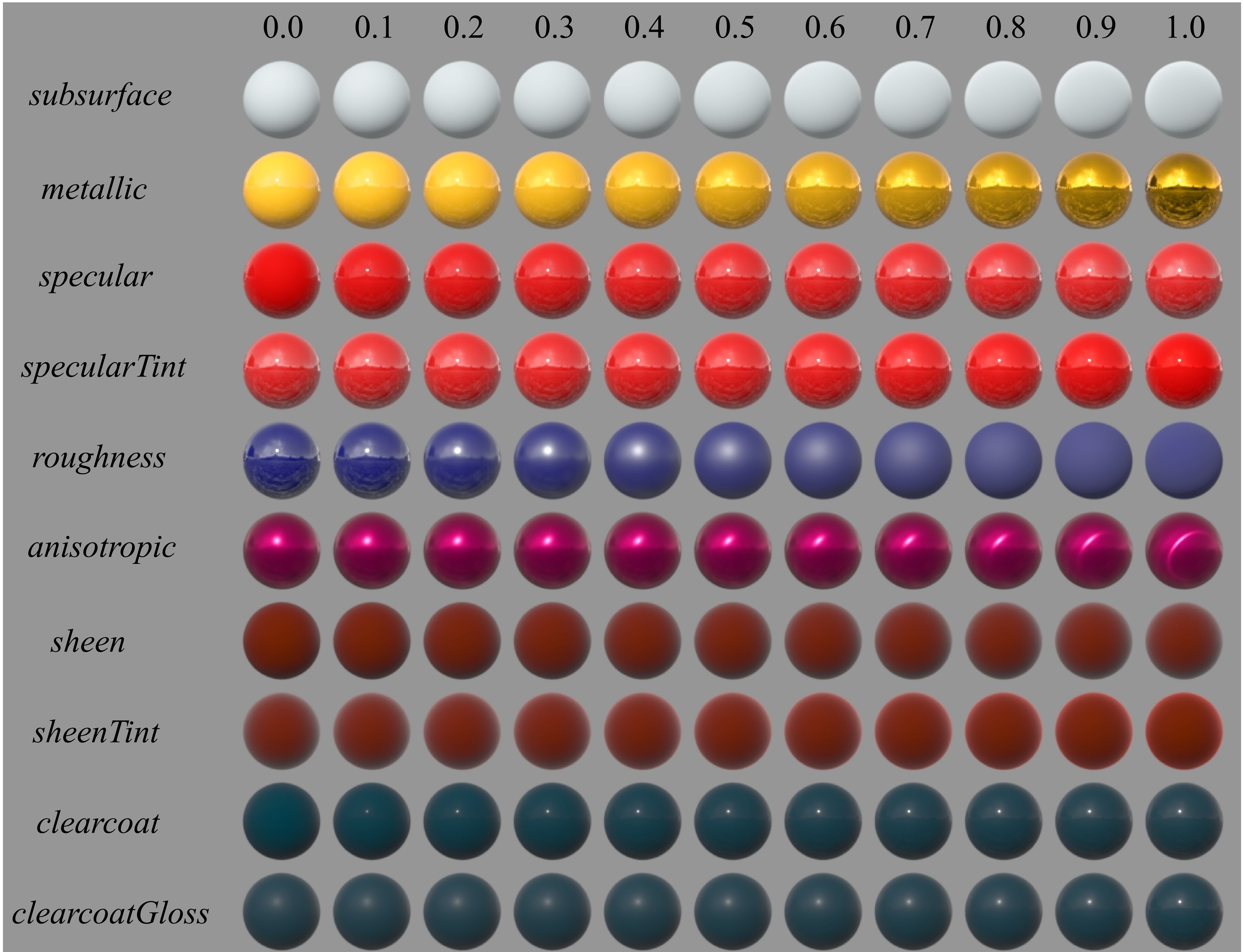
Example 2:
Different textures defining different spatially varying BRDF input parameters



Unity’s shader graph

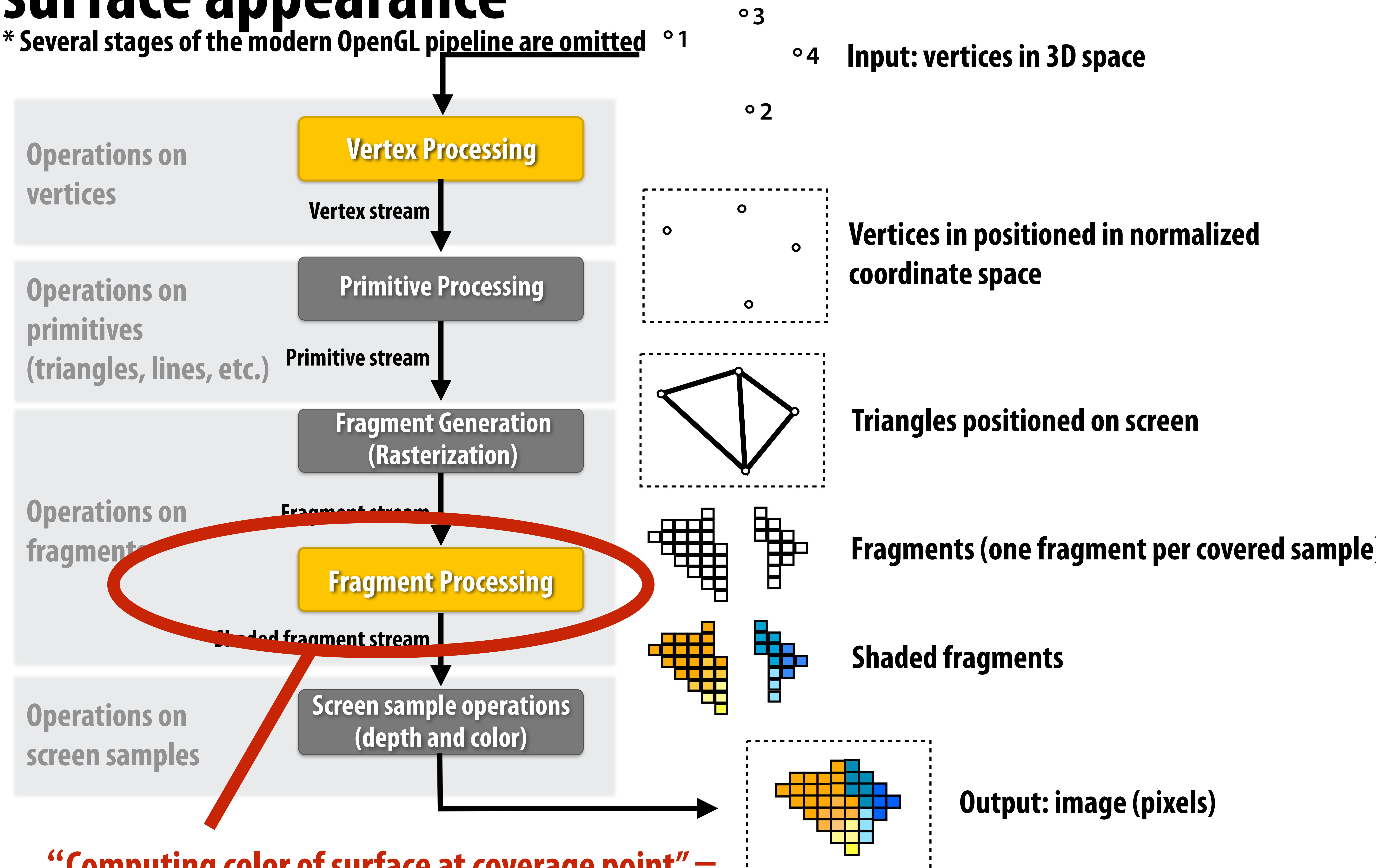


Parameters to Disney BRDF



Fragment processing stage used to evaluate surface appearance

* Several stages of the modern OpenGL pipeline are omitted



"Computing color of surface at coverage point" = simulation of lighting and materials

GLSL shader programs

Define behavior of vertex processing and fragment processing stages of pipeline

Describe operation on a single vertex (or single fragment)

Example GLSL fragment shader program

```
uniform sampler2D myTexture;
uniform vec3 lightDir;       // light direction
uniform vec3 Li;             // light intensity

varying vec2 uv;
varying vec3 norm;

void diffuseShader()
{
  vec3 kd = texture2d(myTexture, uv);
  in_light *= Li * clamp(dot(-lightDir, norm), 0.0, 1.0);
  gl_FragColor = vec4(kd * in_light, 1.0);
}
```

Program parameters

Per-fragment attributes (interpolated by rasterizer)

Sample surface albedo (reflectance color) from texture

Output color

Diffuse brdf: f(wo, wi) = kd
incoming light reflected equally in all directions
(fraction reflected = kd)

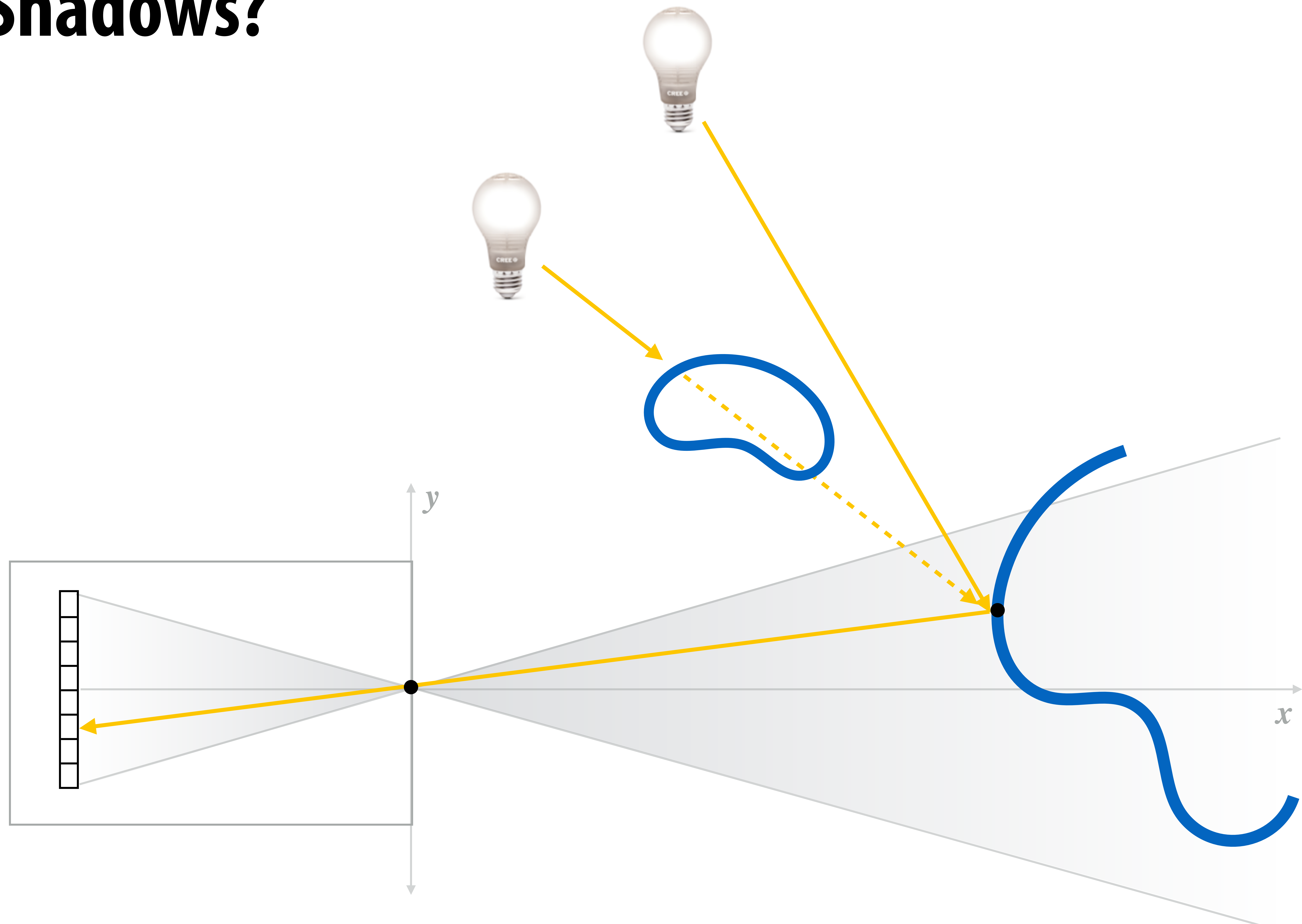
Shader function executes once per fragment.

Outputs color of surface at sample point corresponding to fragment.

(this shader performs a texture lookup to obtain the surface's material color at this point, then performs a simple lighting computation)

Real-time tricks of the trade

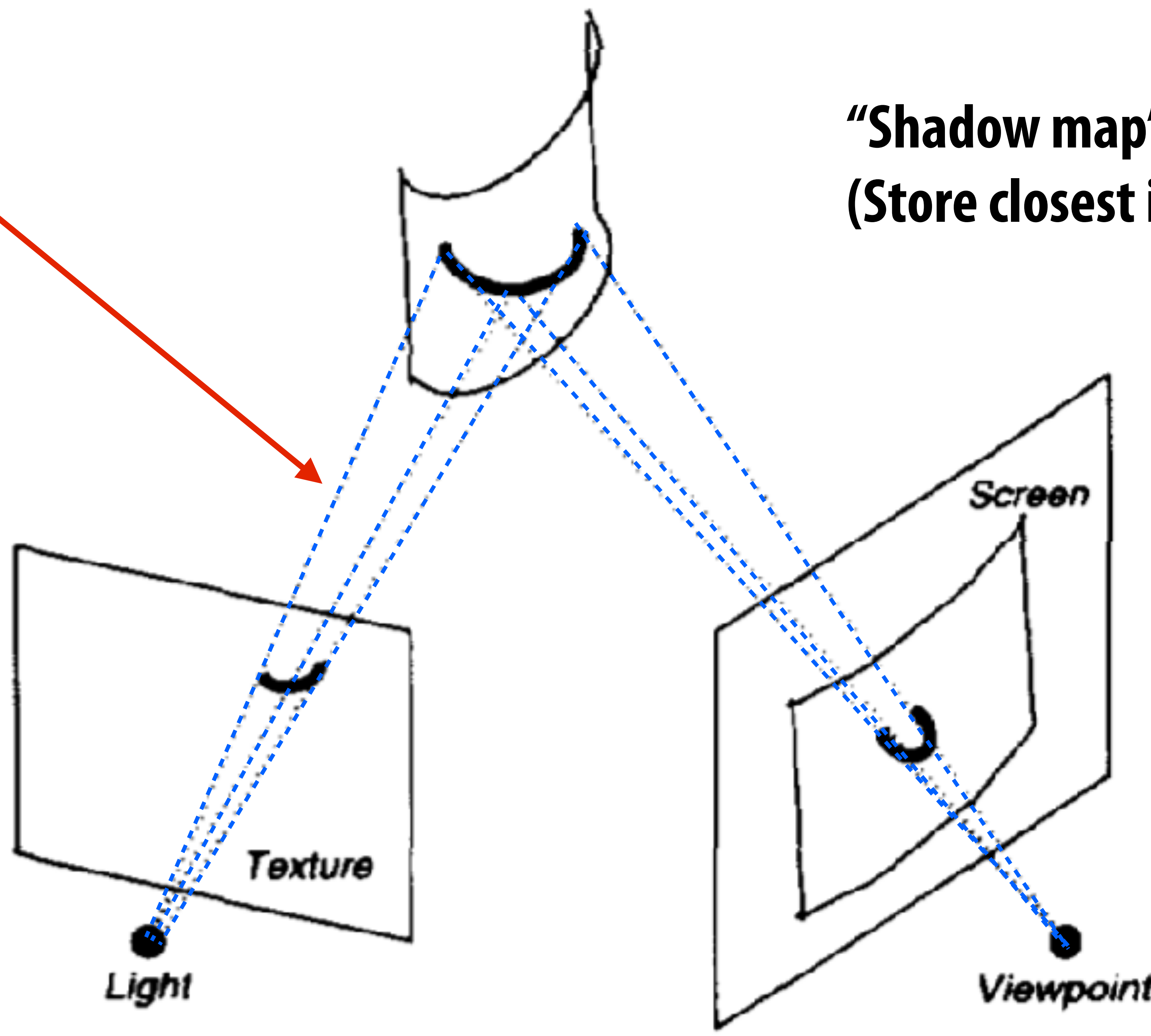
Shadows?



Shadow mapping: ray origin need not be the scene's camera position [Williams 78]

- Place ray origin at position of a point light source
- Render scene to compute depth to closest object to light along uniformly distributed "shadow rays" (answer stored in depth buffer)
- Store precomputed shadow ray intersection results in a texture

Shadow rays



"Shadow map" = depth map from perspective of a point light.
(Store closest intersection along each shadow ray in a texture map)

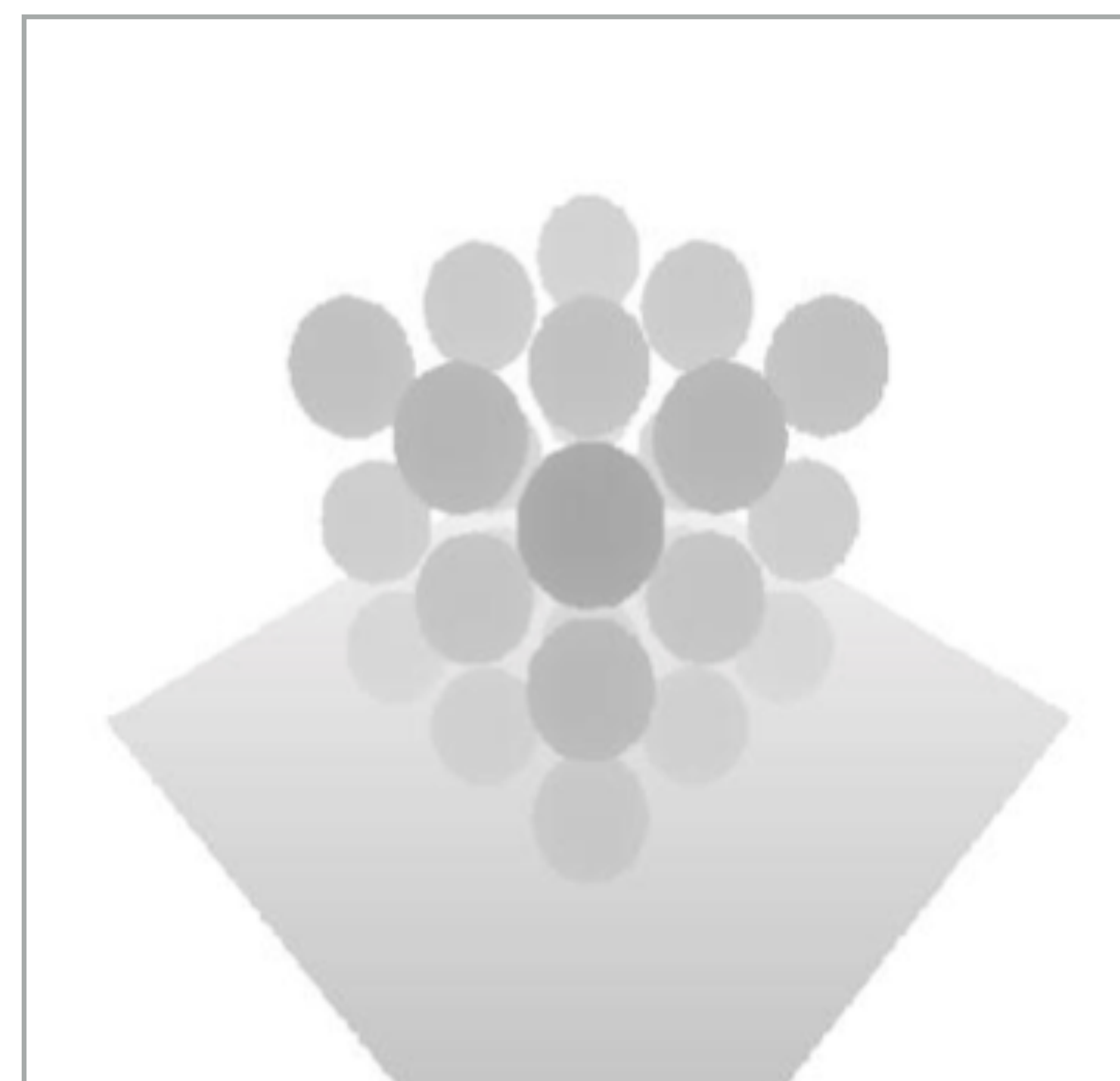
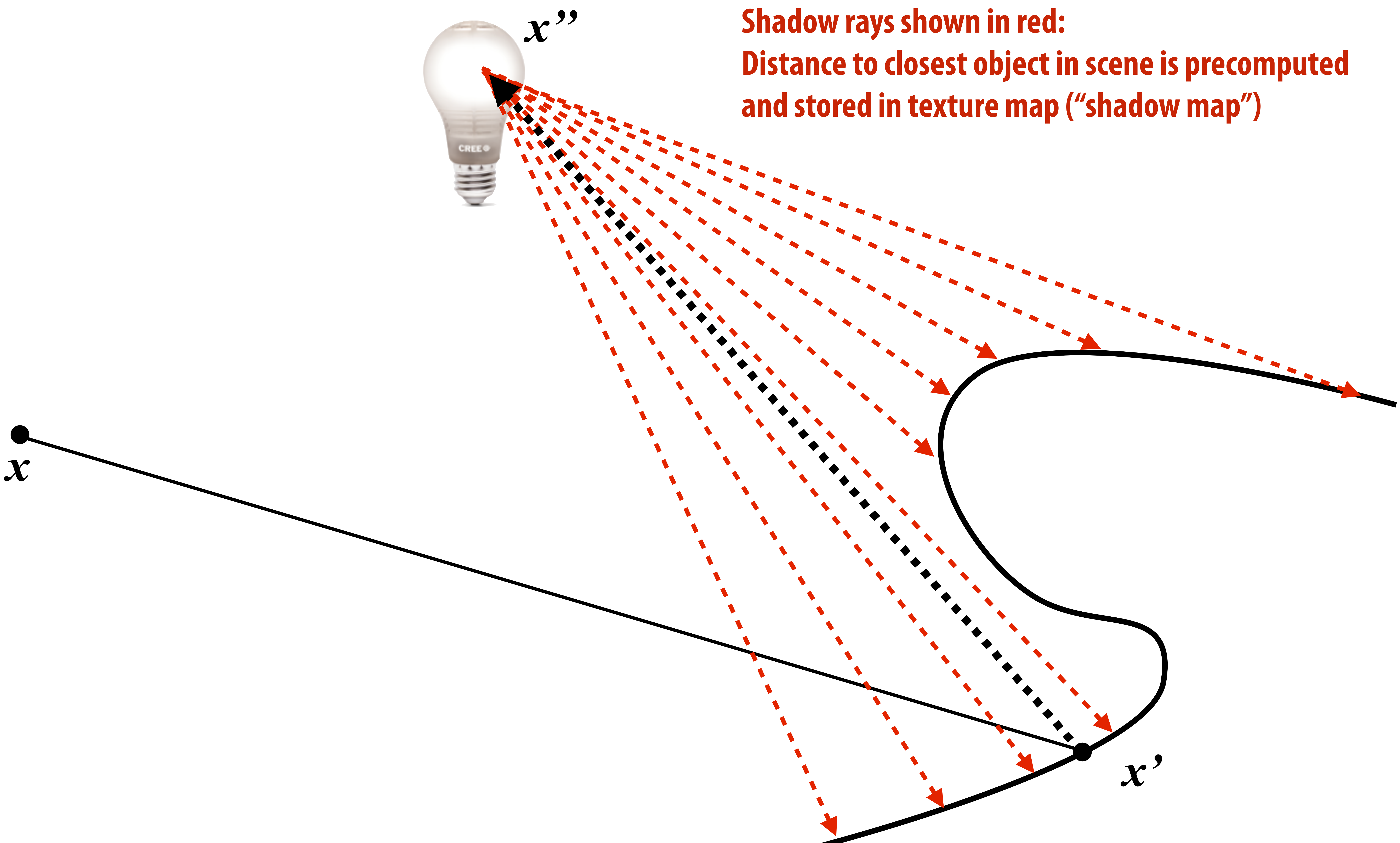
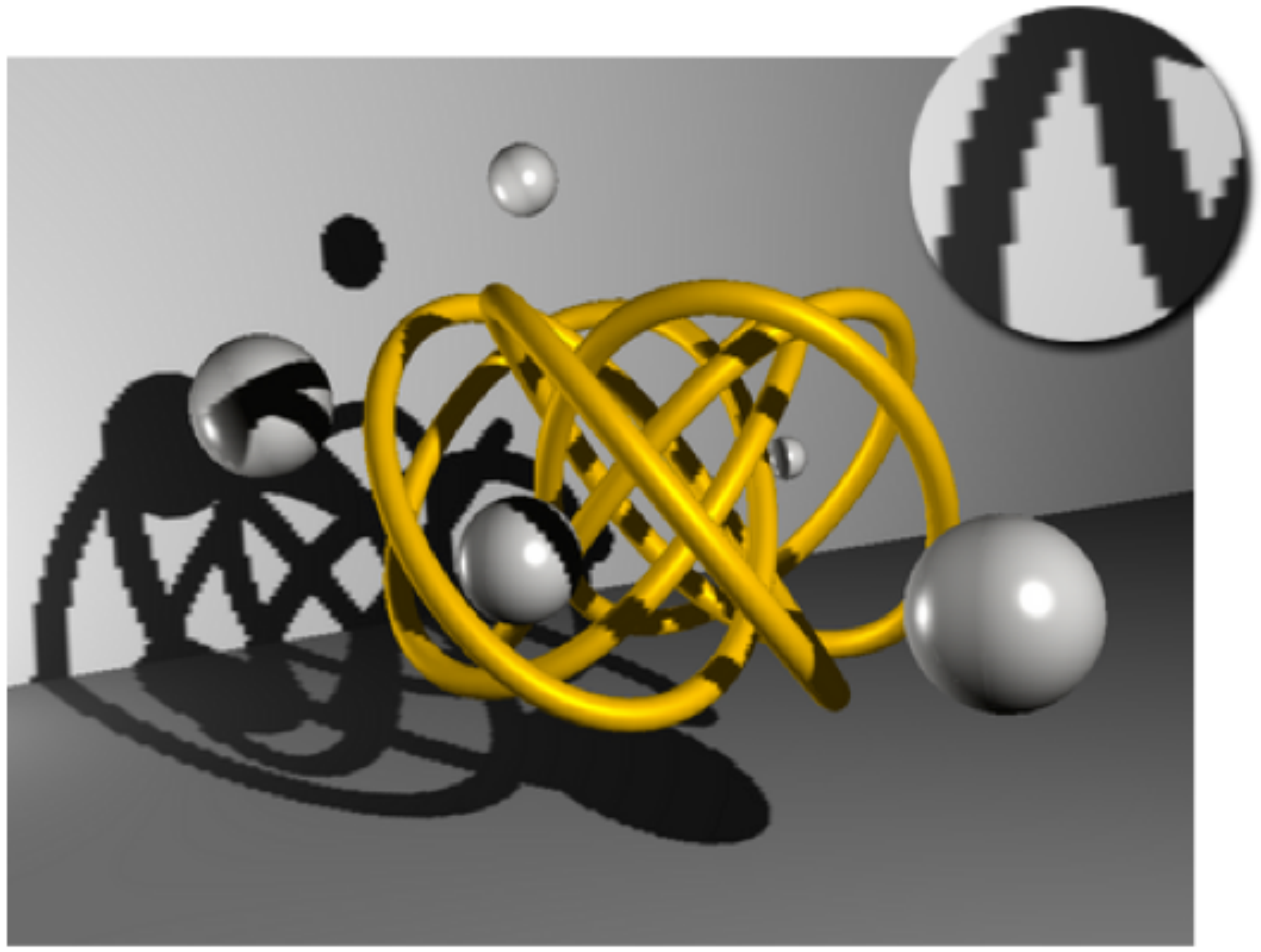


Image credits: Segal et al. 92, Cass Everitt

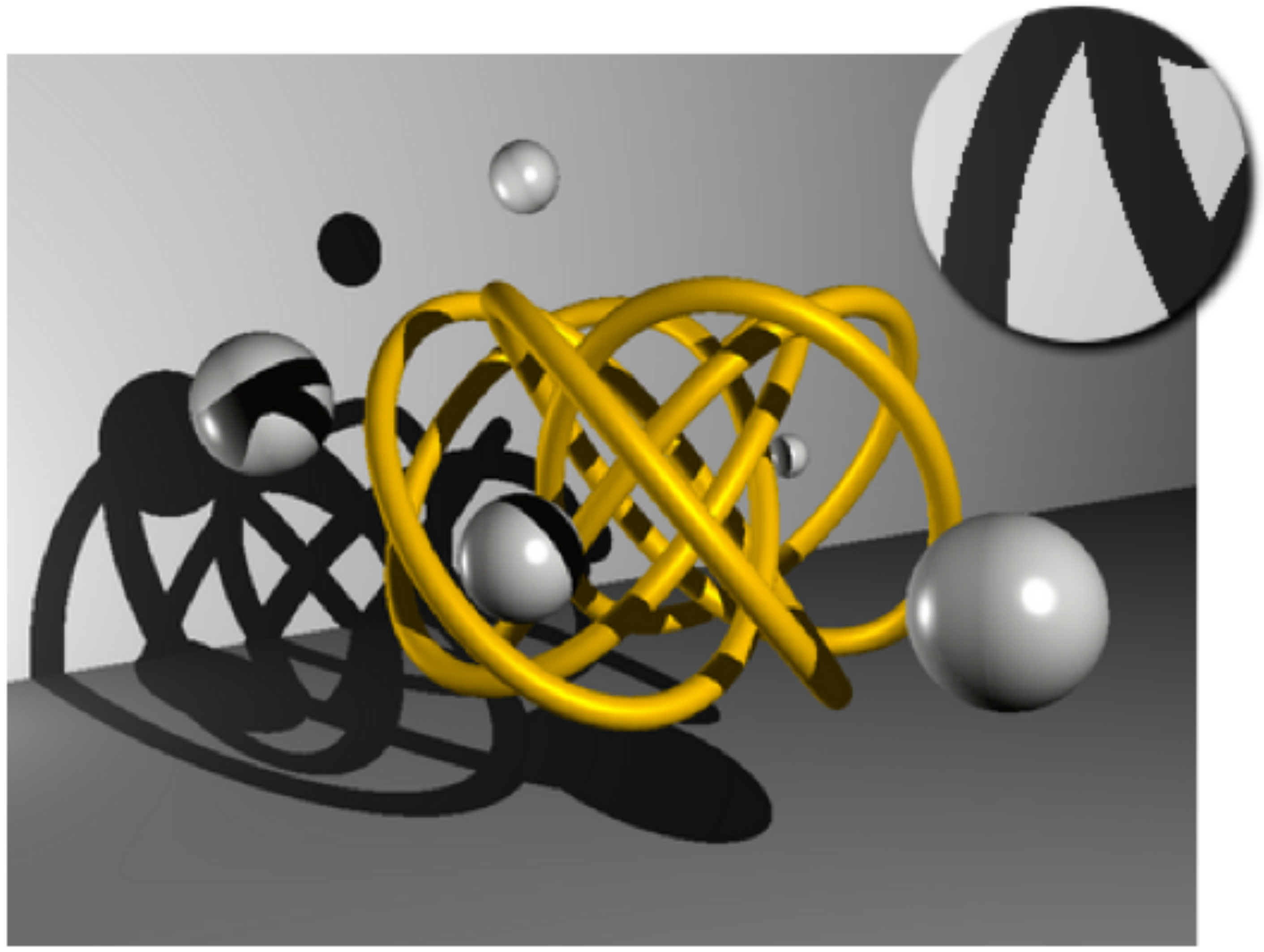
Result of shadow texture lookup approximates visibility result when shading fragment at x’



Shadow aliasing due to shadow map undersampling



Shadows computed using shadow map



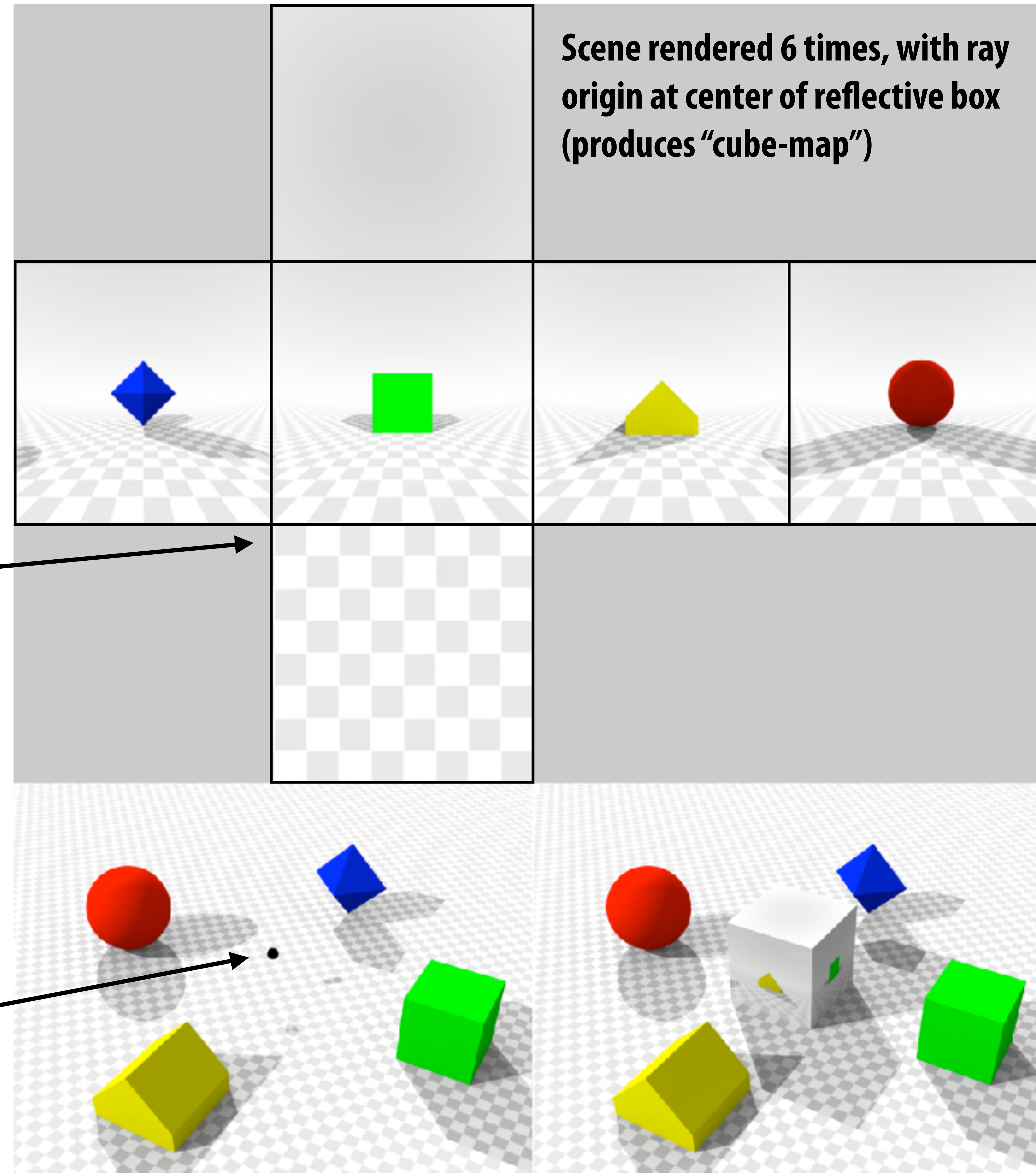
Correct hard shadows
(result from computing v(x′,x″) directly using ray tracing)

Image credit: Johnson et al. TOG 2005

Rasterization: ray origin need not be camera position

Environment mapping:
place ray origin at reflective object

Yields <u>approximation</u> to true reflection results. Why?



Cube map:
stores results of approximate mirror reflection rays

(Question: how can a glossy surface be rendered using the cube-map)

Center of projection

Image credit: http://en.wikipedia.org/wiki/Cube_mapping

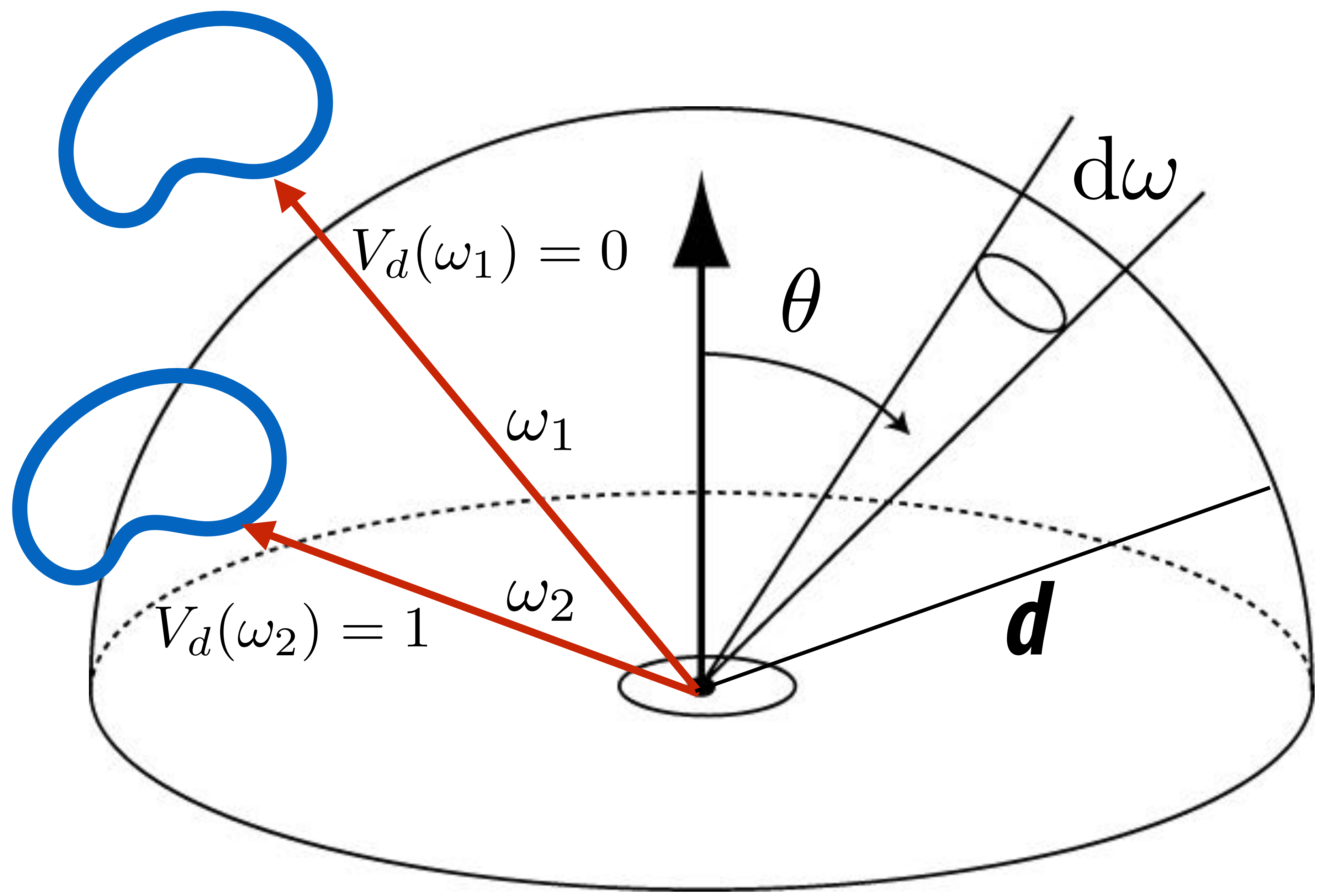
Ambient occlusion

Ambient occlusion

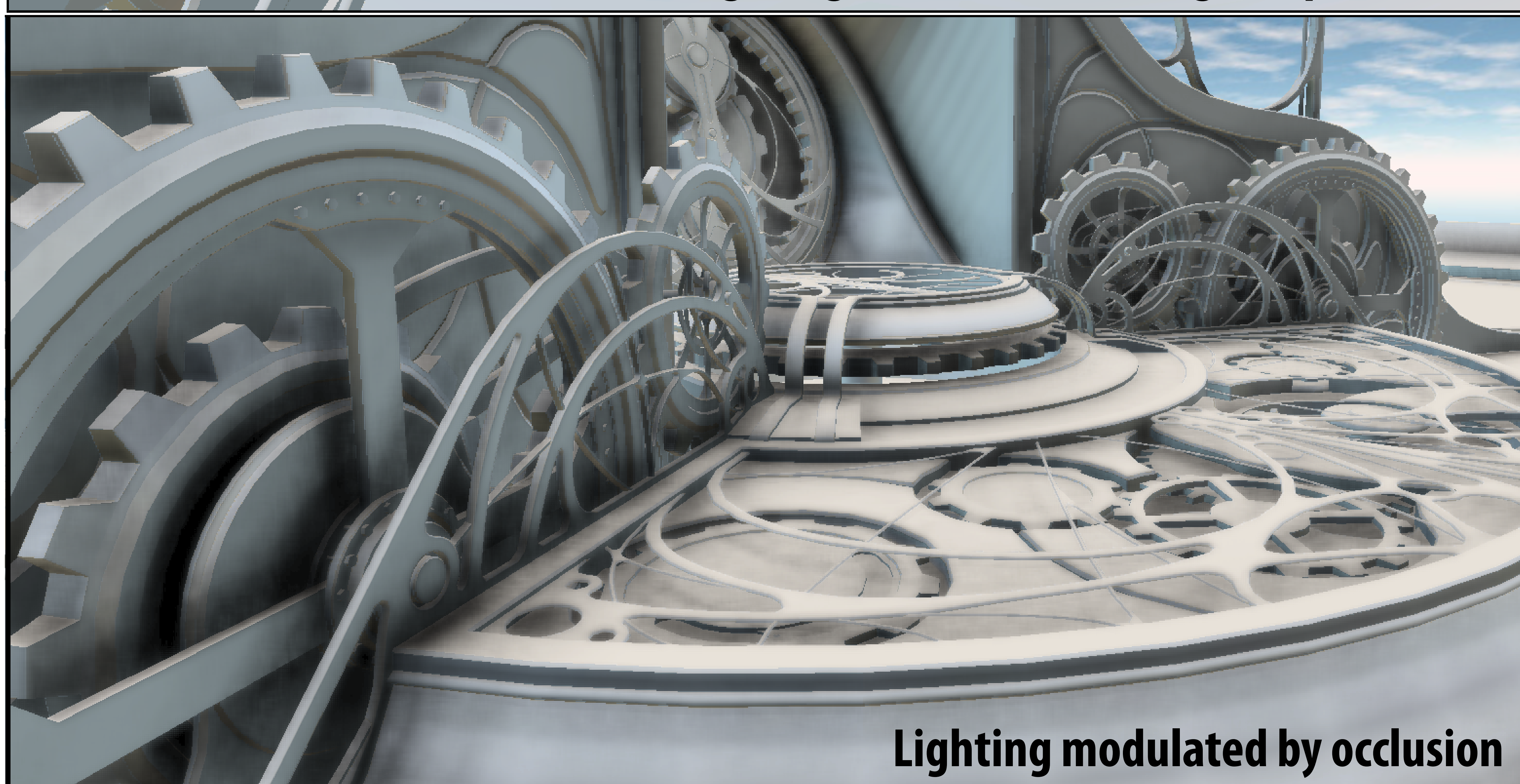
Idea:

Precompute "amount of hemisphere" that is occluded within distance *d* from a point.

When shading, attenuate environment lighting by this amount.

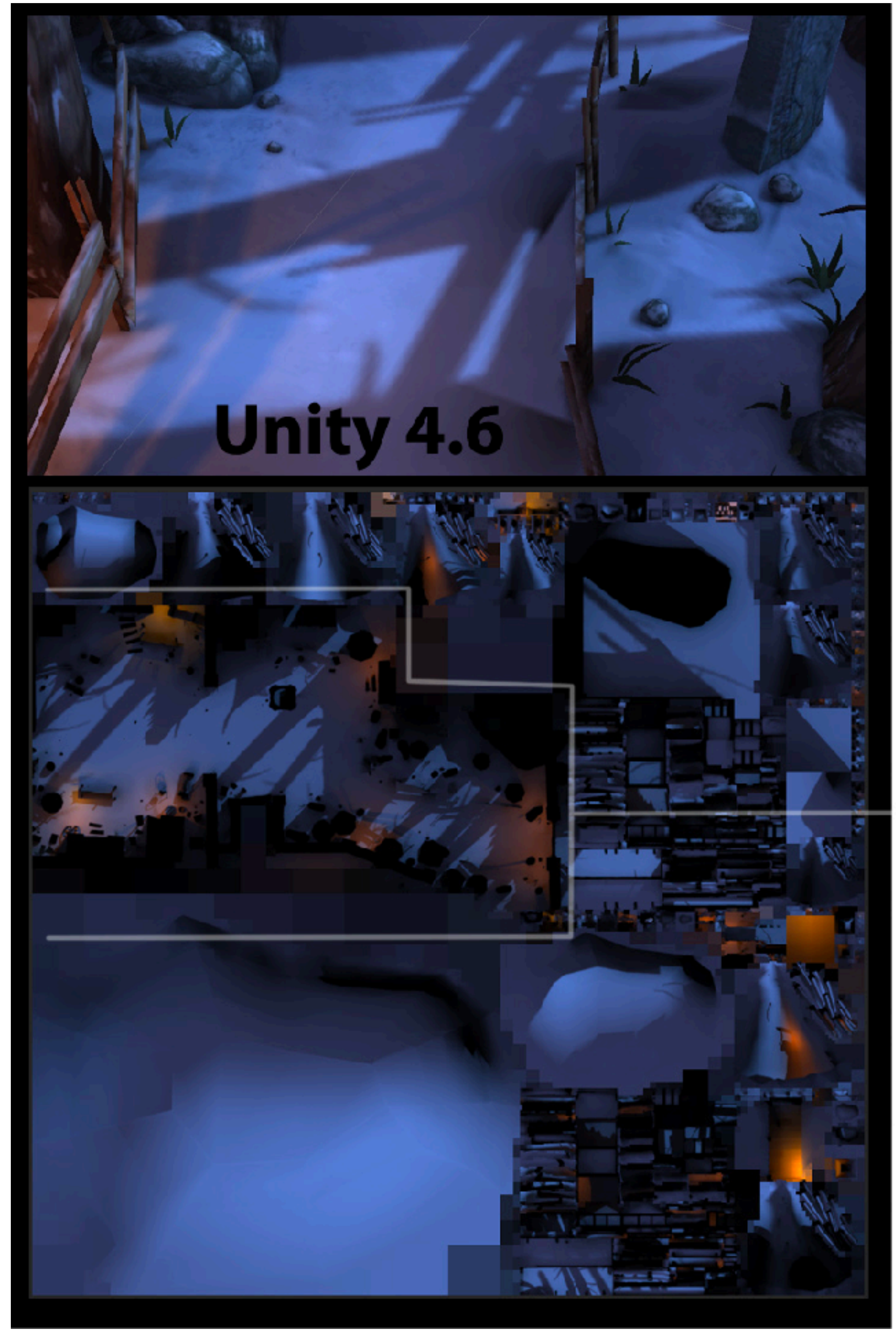


Ambient occlusion



Precomputed lighting

- Precompute lighting for a scene offline
  - Offline computations can perform advanced shadowing, inter reflection computations
- "Bake" results of lighting into texture map



Why there is interest in ray tracing

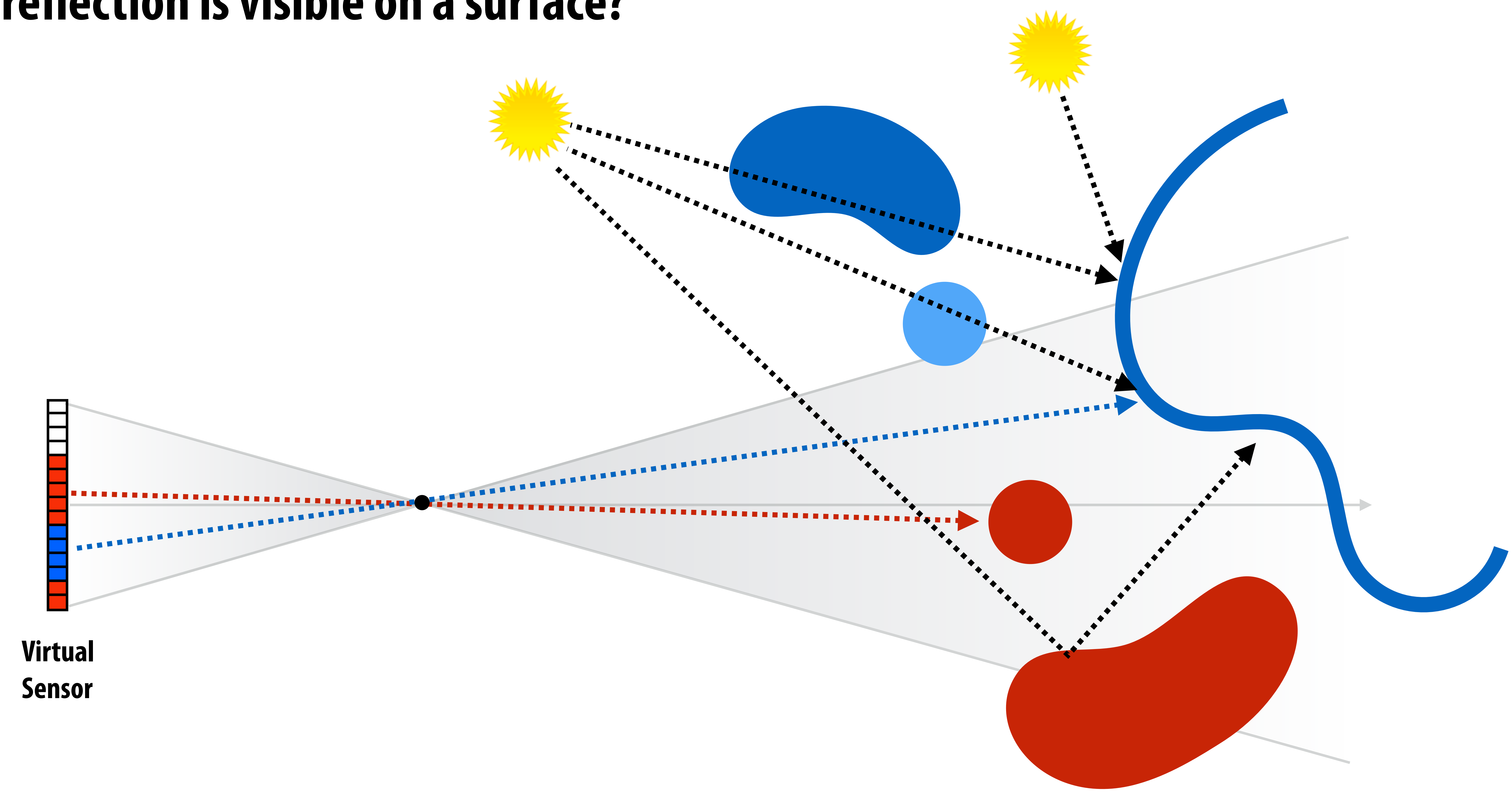
(it is general solution to many types of illumination effects)

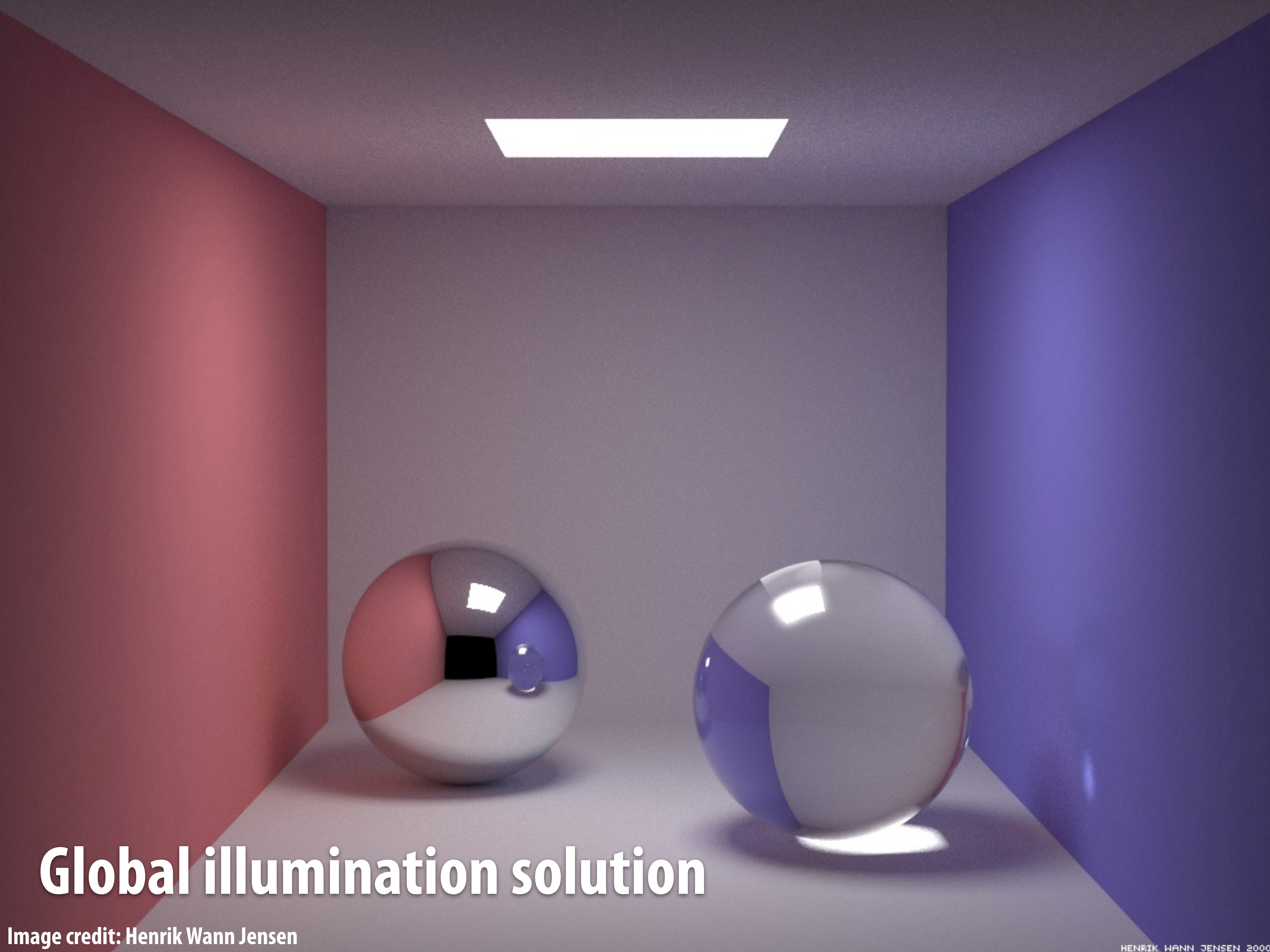
Generality of ray-scene queries

What object is visible to the camera?

What light sources are visible from a point on a surface (Is a surface in shadow?)

What reflection is visible on a surface?





Global illumination solution

Image credit: Henrik Wann Jensen

Image credit: NVIDIA (this ray traced image can be rendered at interactive rates on modern GPUs)

Acknowledgements

- Thanks to Keenan Crane, Ren Ng, Pat Hanrahan and Matt Pharr for presentation resources