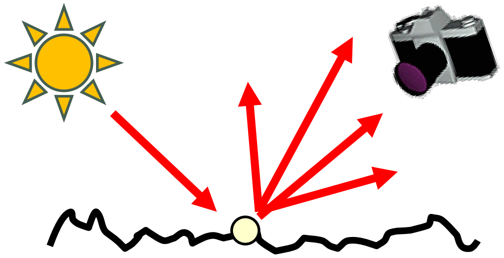
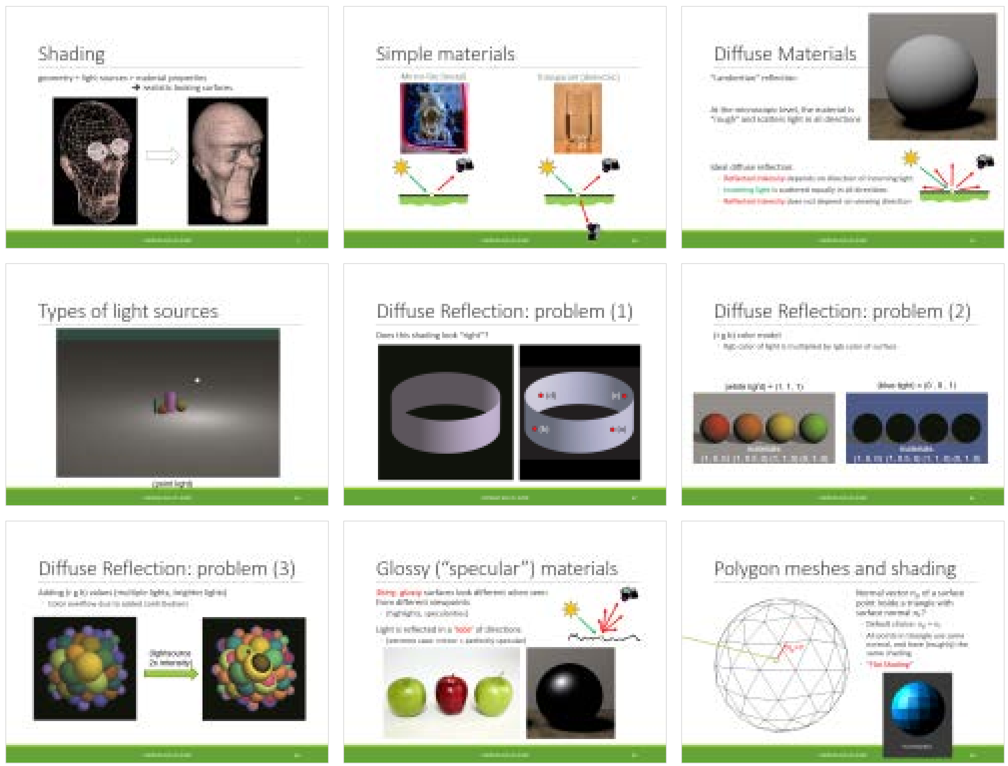
Basic Shading Models



FUNDAMENTALS OF COMPUTER GRAPHICS
PHILIP DUTRÉ
DEPARTMENT OF COMPUTER SCIENCE

Overview Lecture

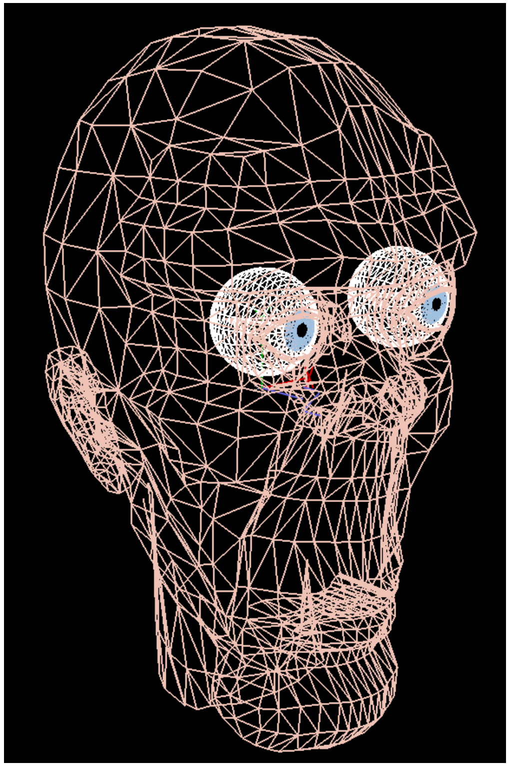


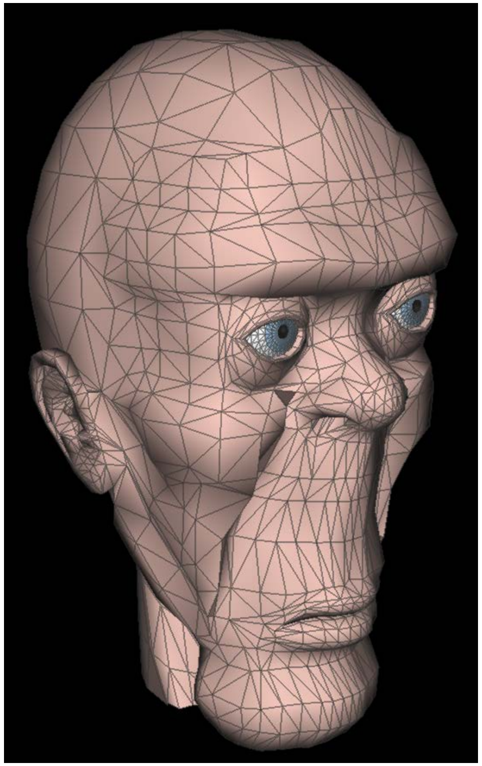
Relevant sections in book: Chapter 14, 15
(Illustrations from Ray Tracing From The Ground Up, Physically-Based Rendering, Fundamentals of Computer Graphics)
(Page numbering might skip some slides due to 'hidden' slides in my presentation.)

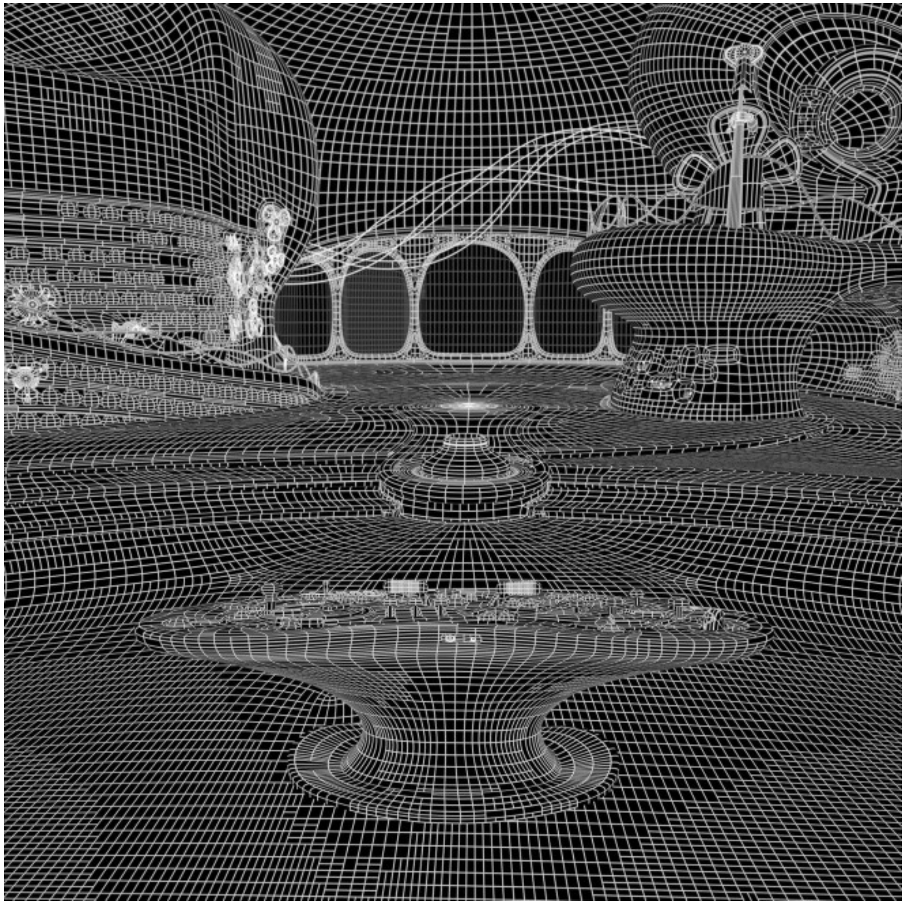
Shading

geometry + light sources + material properties

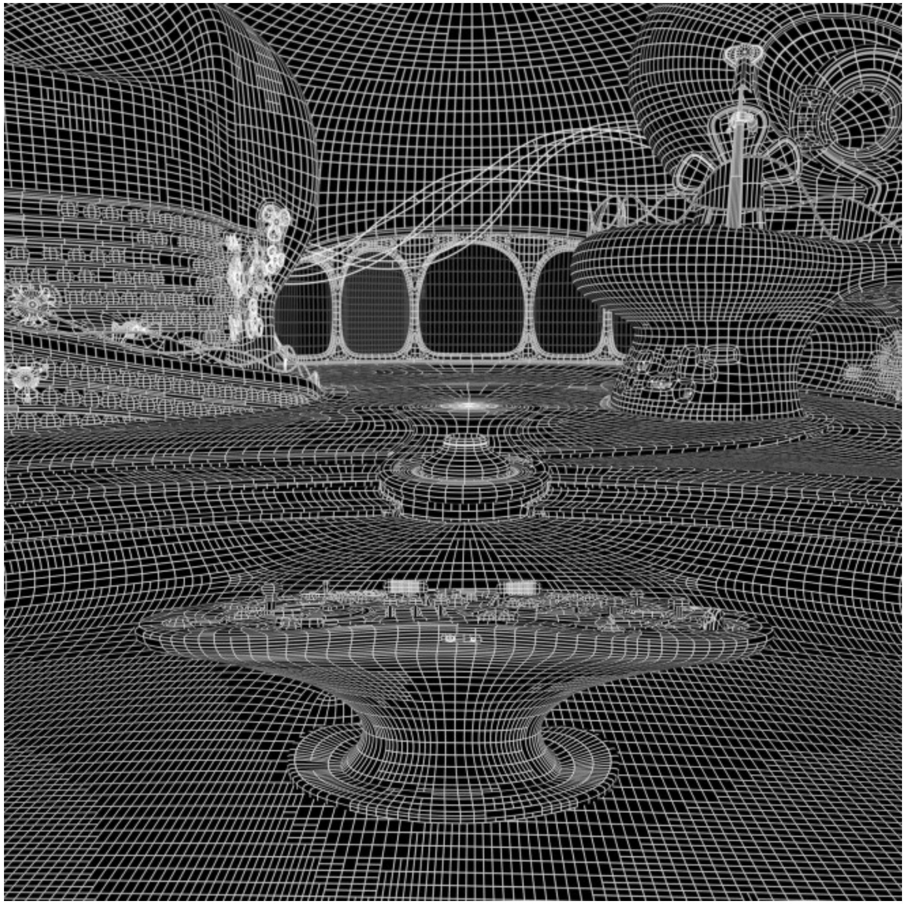
➔ realistic looking surfaces







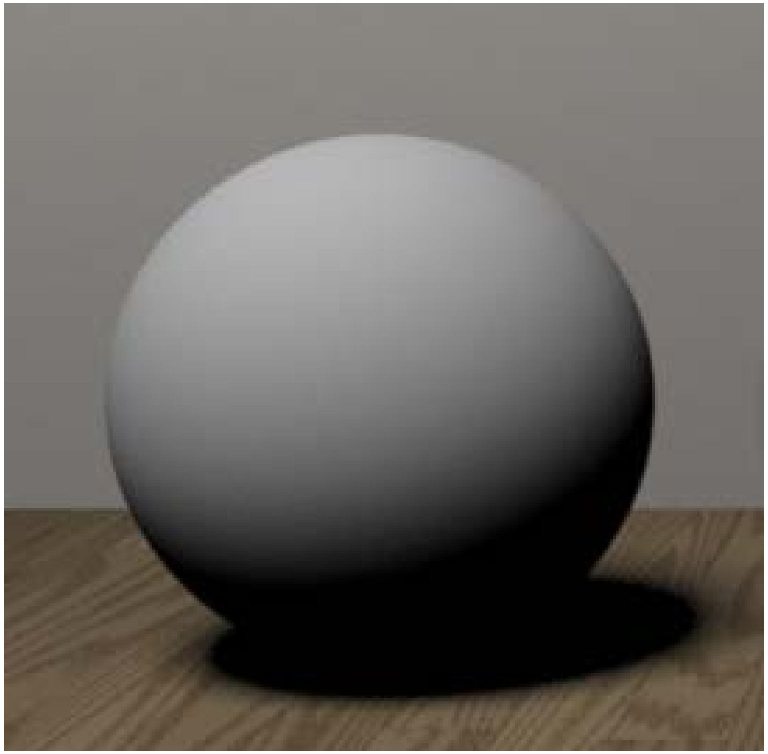
(Pixar, Inside Out, 2015)



(Pixar, Inside Out, 2015)

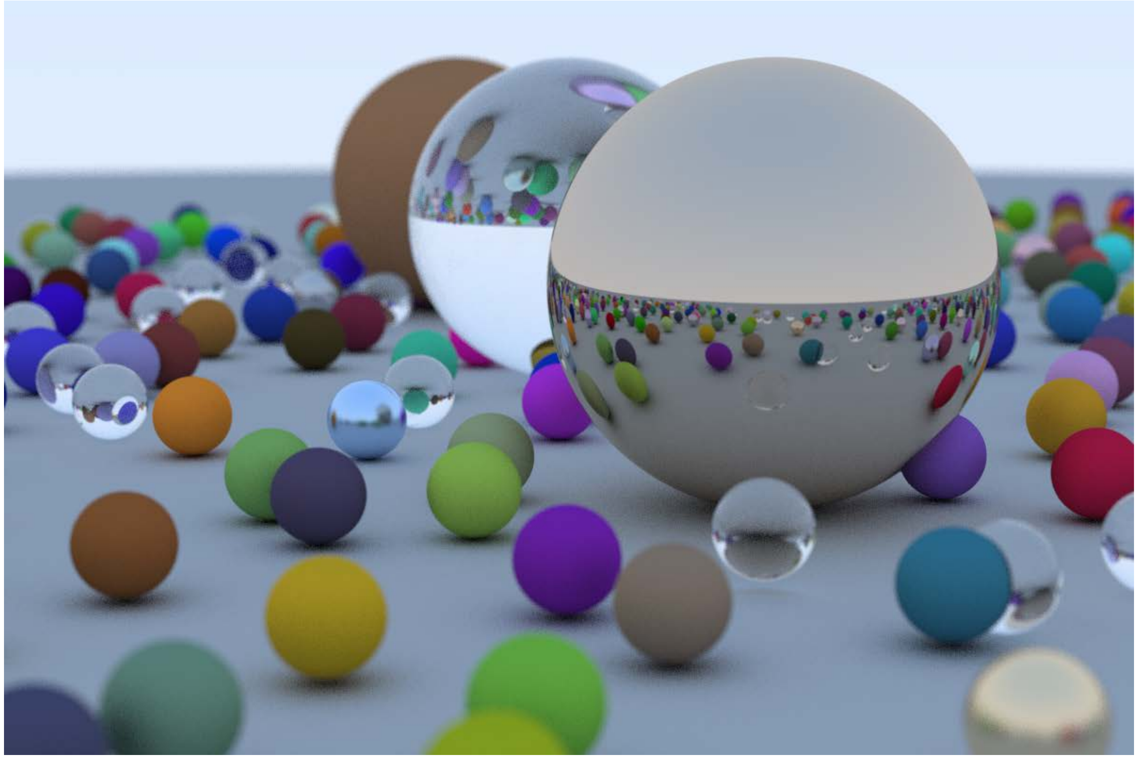
Shading: shape perception

Shading is necessary for shape perception



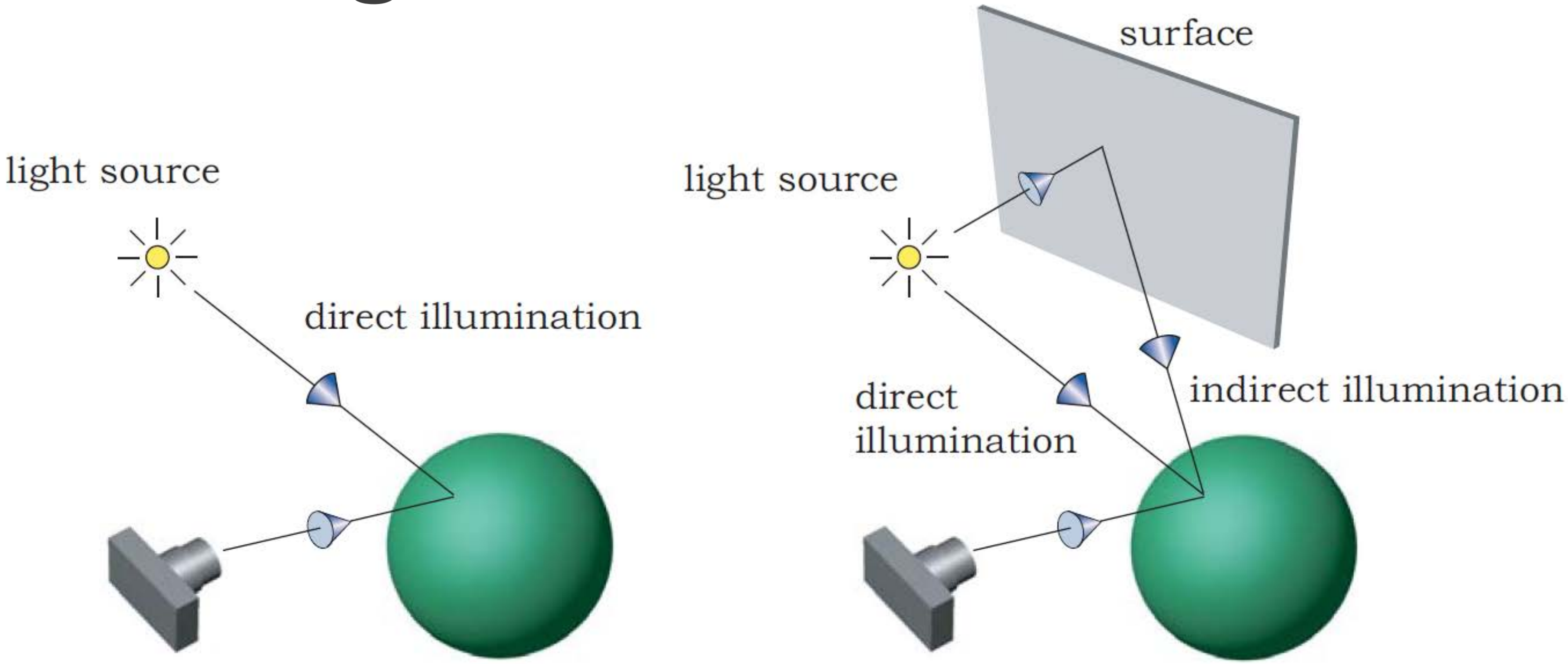
Shading: material perception

Shading is necessary for material perception



Peter Shirley, Ray Tracing in One Weekend, 2016
http://in1weekend.blogspot.com/2016/01/ray-tracing-in-one-weekend.html

Shading: direct + indirect

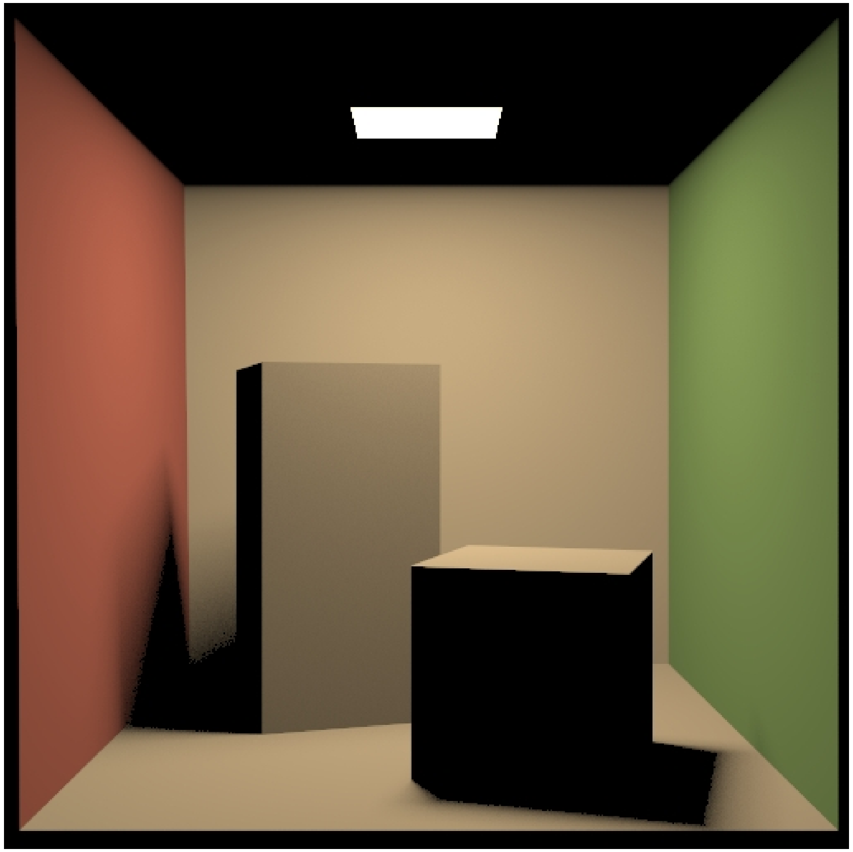


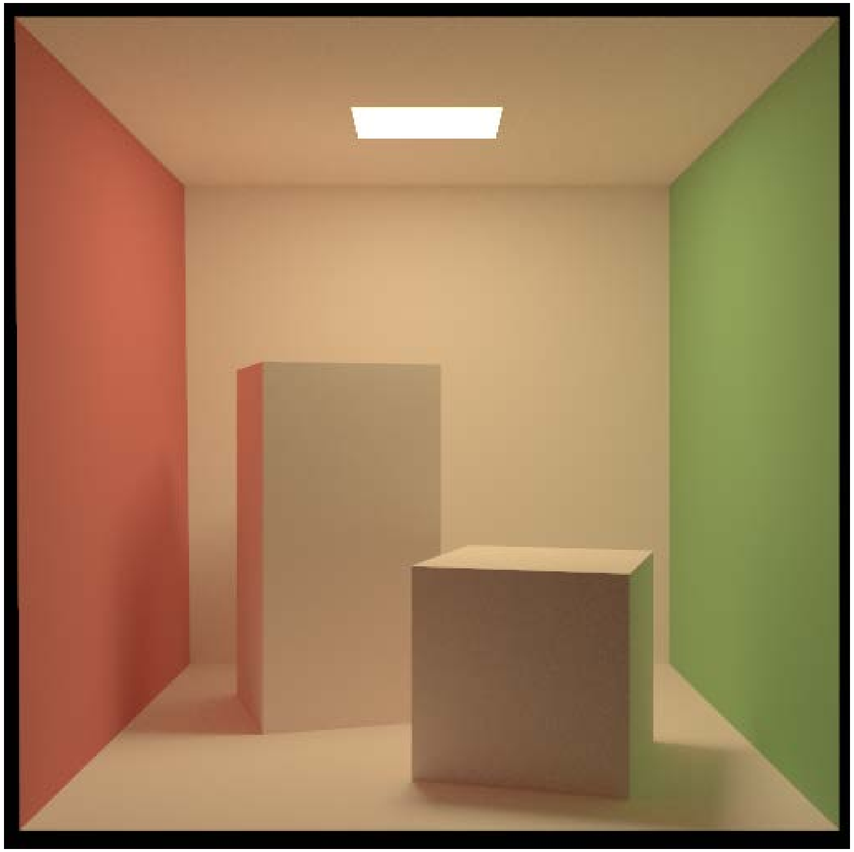
How much light is detected by the camera through each pixel?

How does light reflect at surfaces?

How can we generate all relevant light paths (direct, indirect)?

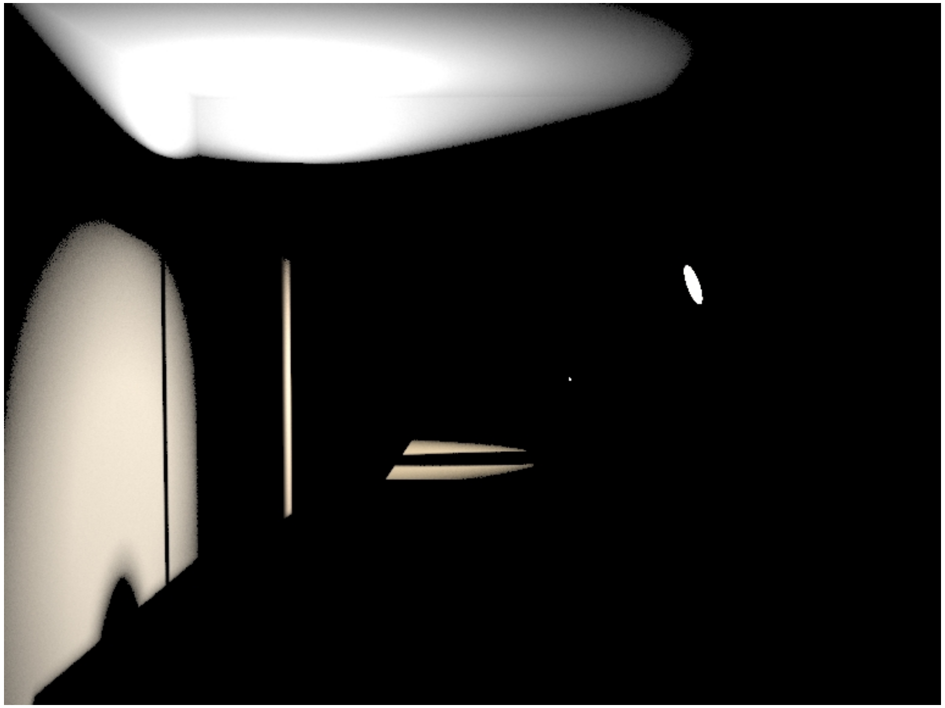
Shading: direct + indirect





Direct illumination only Global (direct+indirect) illumination
(Mitsuba Renderer)

Shading: direct + indirect



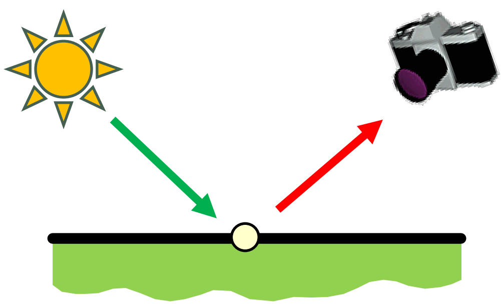
Direct illumination only

Global (direct+indirect) illumination

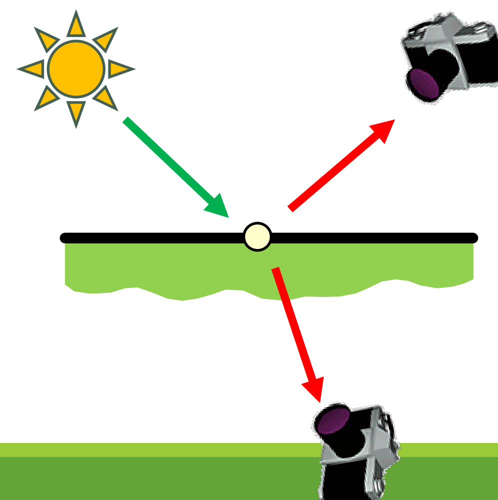
(Mitsuba Renderer)

Simple materials

Mirror-like (metal)

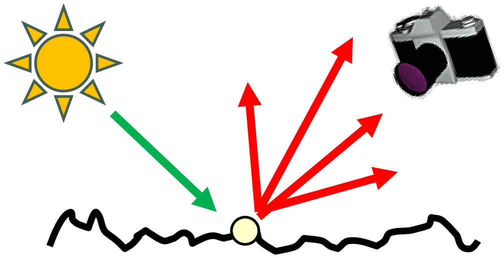


Transparant (dielectric)

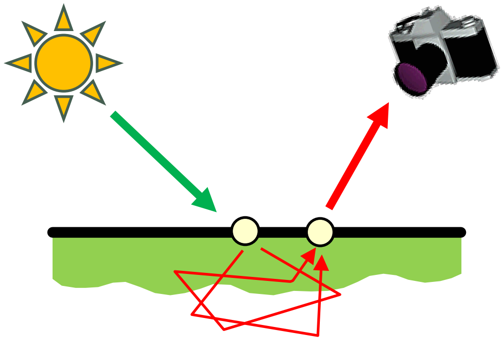


Complex materials

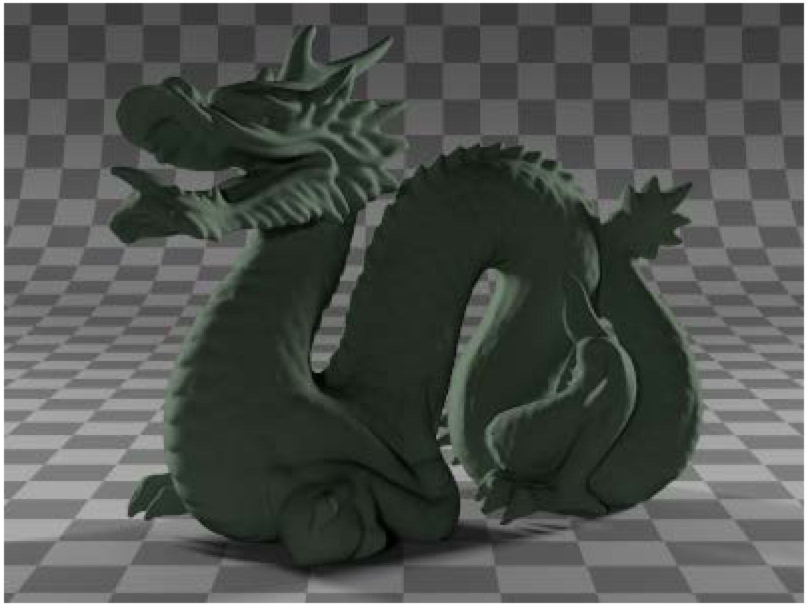
Rough (brushed)



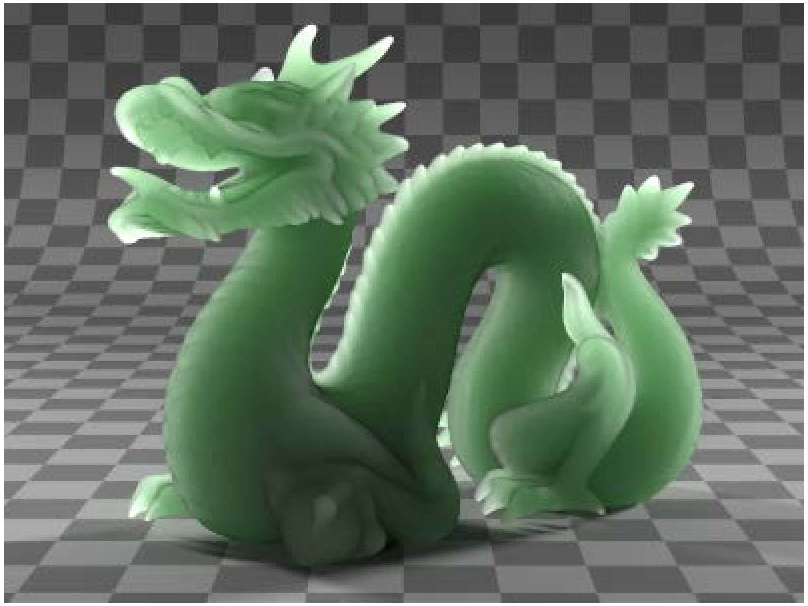
Subsurface scattering



Subsurface Scattering



Without subsurface scattering



With subsurface scattering

(A Forward Scattering Dipole Model from a Functional Integral Approximation, Frederickx & Dutré, SIGGRAPH 2017)

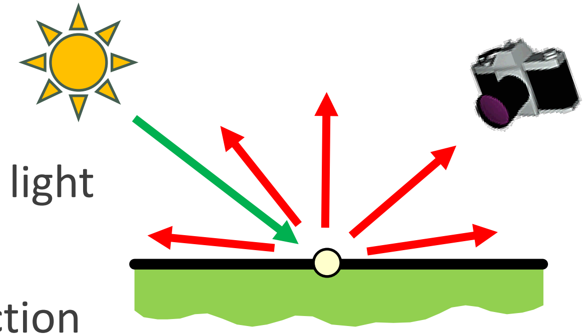
Diffuse Materials

"Lambertian" reflection

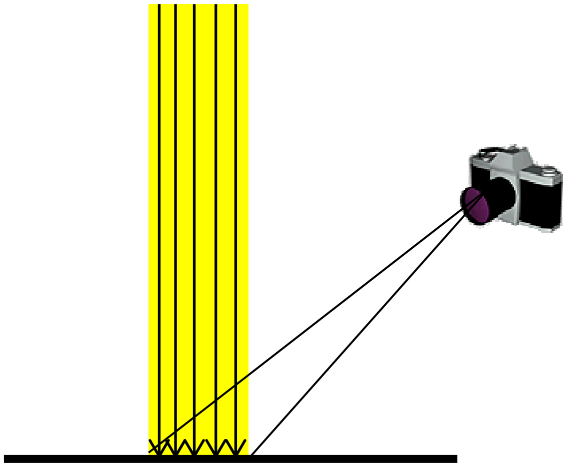
At the microscopic level, the material is "rough" and scatters light in all directions

Ideal diffuse reflection:

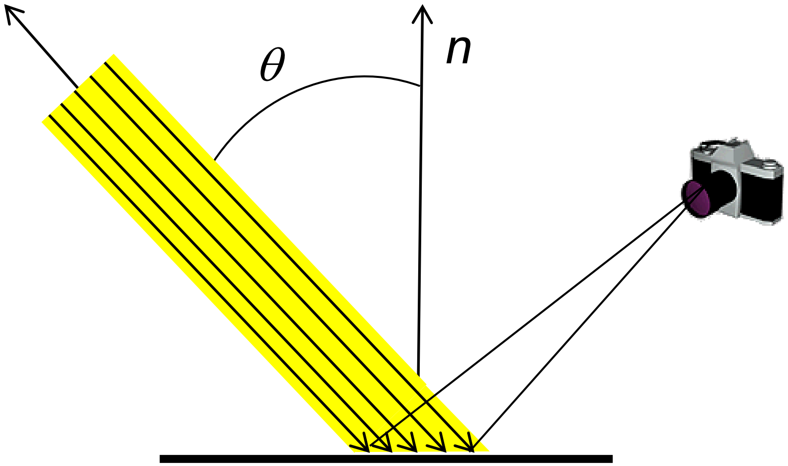
- Reflected intensity depends on direction of incoming light
- Incoming light is scattered equally in all directions
- Reflected intensity does not depend on viewing direction



Diffuse Materials



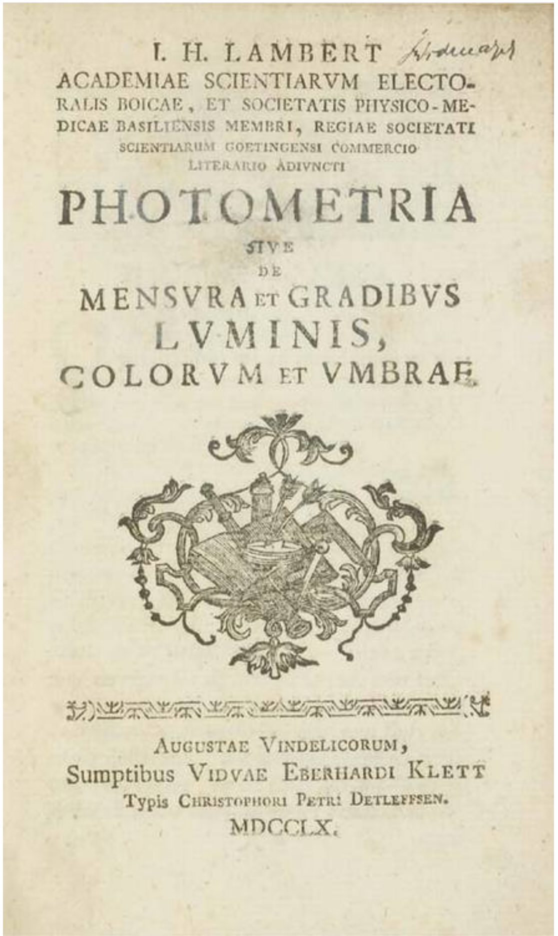
lightbeam incident on area A



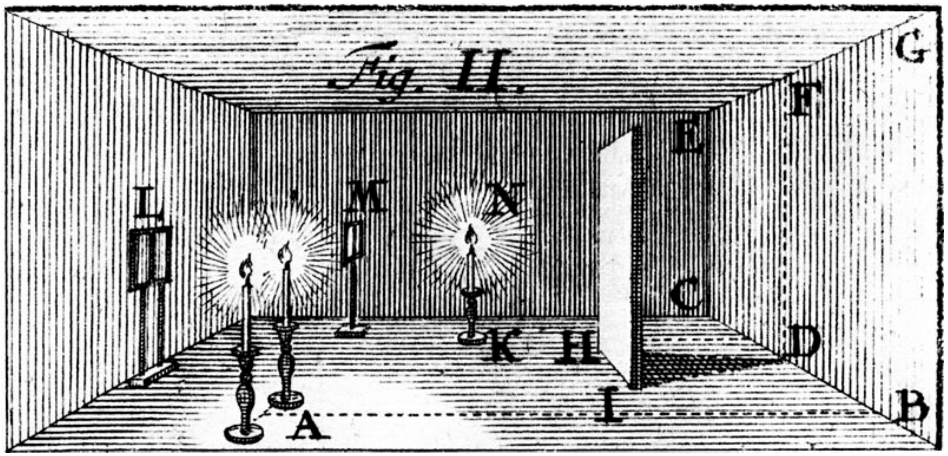
lightbeam incident on area $A/\cos\theta$

➔ average intensity per unit area $\sim\cos\theta$

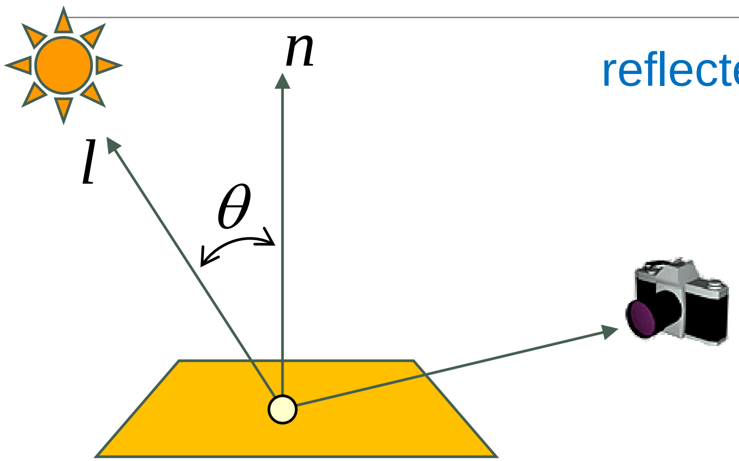
Photometria - Lambert


I. H. LAMBERT
ACADEMIAE SCIENTIARVM ELECTORALIS BOICAE, ET SOCIETATIS PHYSICO-MEDICAE BASILIENSIS MEMBRI, REGIAE SOCIETATI SCIENTIARUM GOETINGENSI COMMERCIO LITERARIO ADIVNCTI
PHOTOMETRIA
SIVE
DE
MENSVRA ET GRADIBVS LVMINIS, COLORVM ET VMBRAE.

AUGUSTAE VINDELICORUM,
Sumptibus VIDVAE EBERHARDI KLETT
Typis CHRISTOPHORI PETRI DETLEFFSEN.
MDCCLX.



Diffuse materials and ray tracing



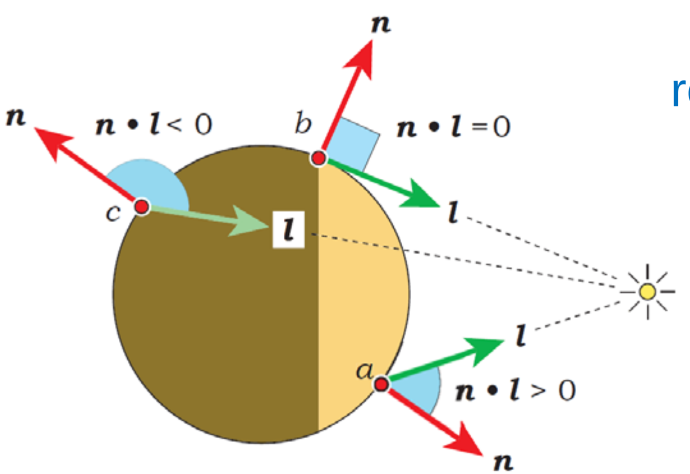
reflected intensity

$$L = \frac{k_d}{\pi} L_{light} \cos\theta$$
$$= \frac{k_d}{\pi} L_{light} \frac{(l.n)}{\|l\|.\|n\|}$$

reflectivity of surface ($0 <= k_d <=1$)

cosine-factor

intensity of light source (L)



Diffuse materials and ray tracing

How to add color?

$$L_r = \frac{k_d}{\pi} c_{d,r} L_{light}\, c_{light,r} \cos\theta$$
$$L_g = \frac{k_d}{\pi} c_{d,g} L_{light}\, c_{light,g} \cos\theta$$
$$L_b = \frac{k_d}{\pi} c_{d,b} L_{light}\, c_{light,b} \cos\theta$$

$$\Rightarrow \quad L = \frac{k_d}{\pi} c_d L_{light}\, c_{light} \cos\theta$$

reflected intensity (rgb triplet)

color of surface (rgb triplet)

color of light source (rgb triplet)

Diffuse materials and ray tracing

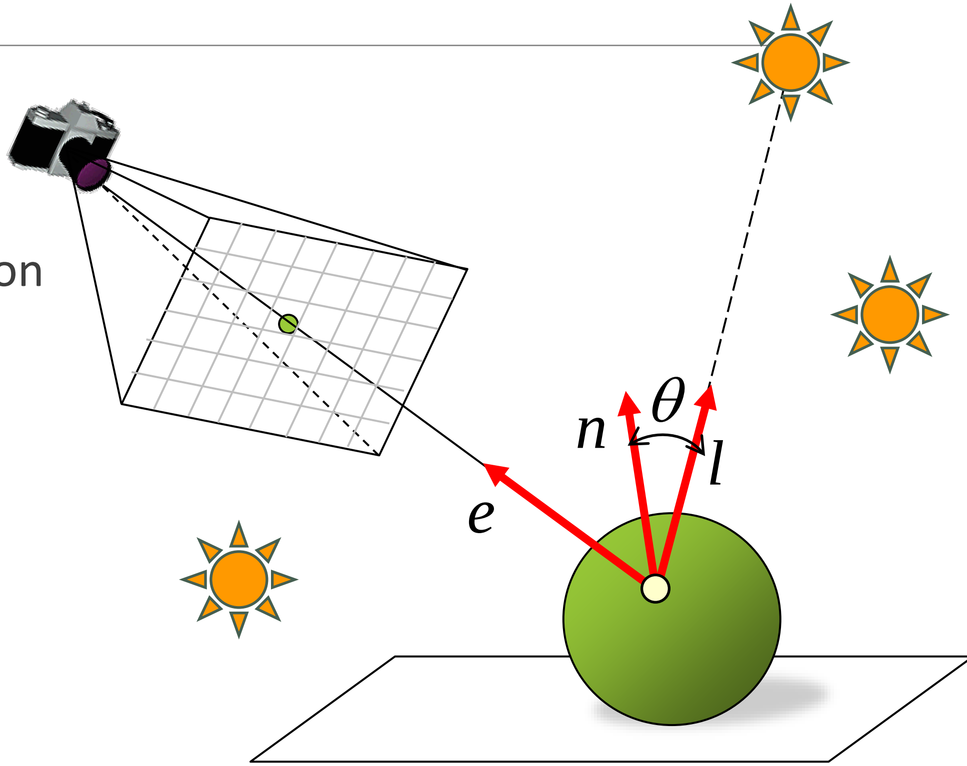
Trace a ray through pixel

Find nearest intersection point

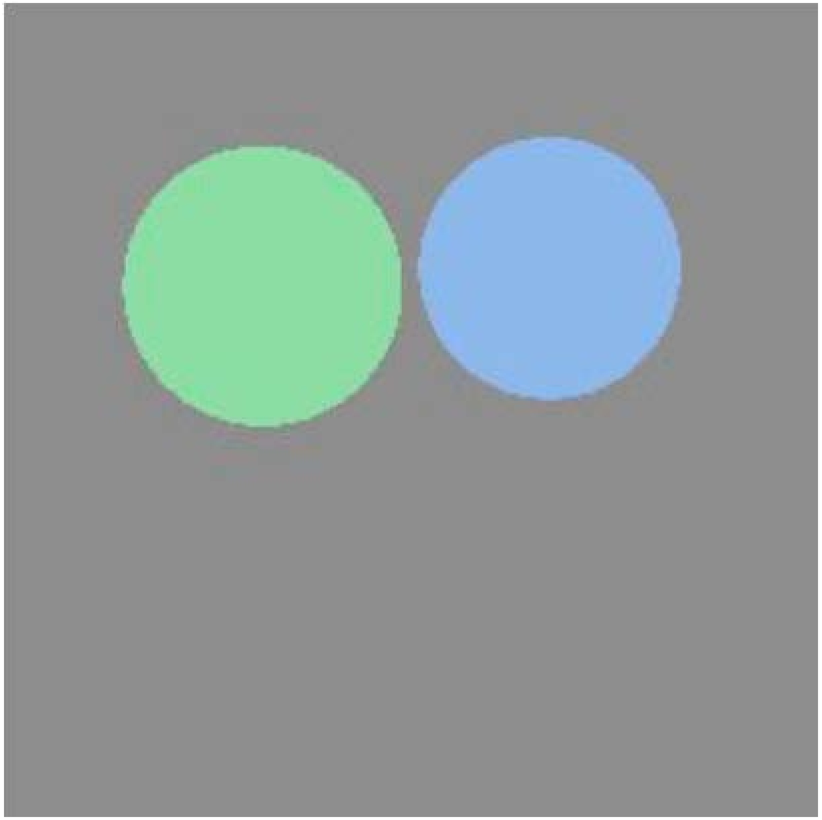
Evaluate diffuse shading at intersection point:

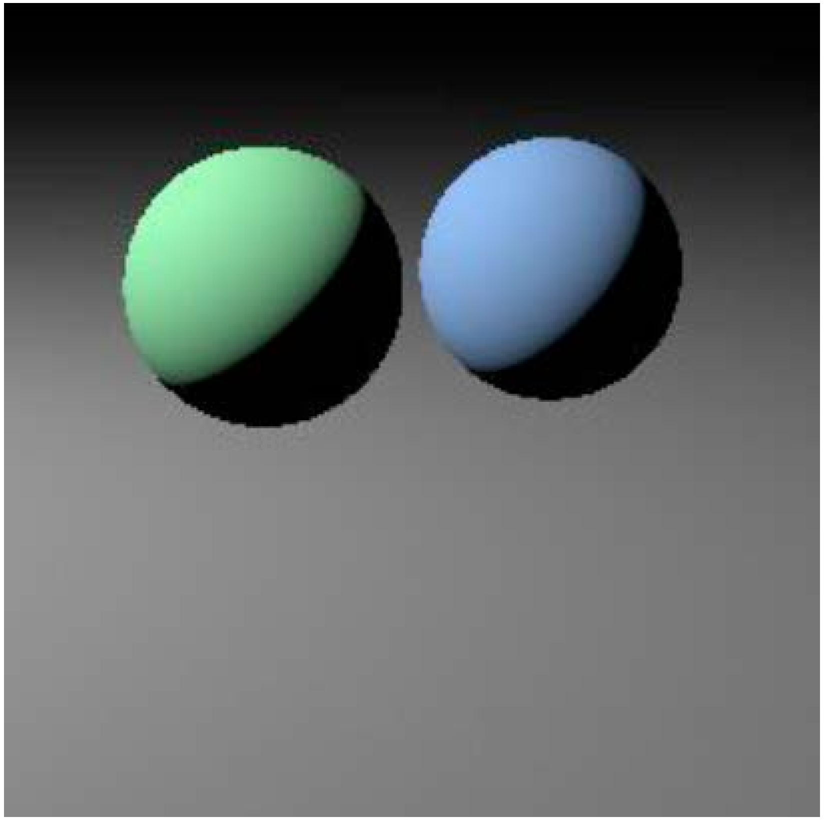
- Geometry:
  - compute normal vector n
  - direction to light source l
- Material:
  - color c_d & reflectivity k_d
- Light source:
  - color c_{light} & intensity L_{light}

(repeat for each light source, add contributions together)



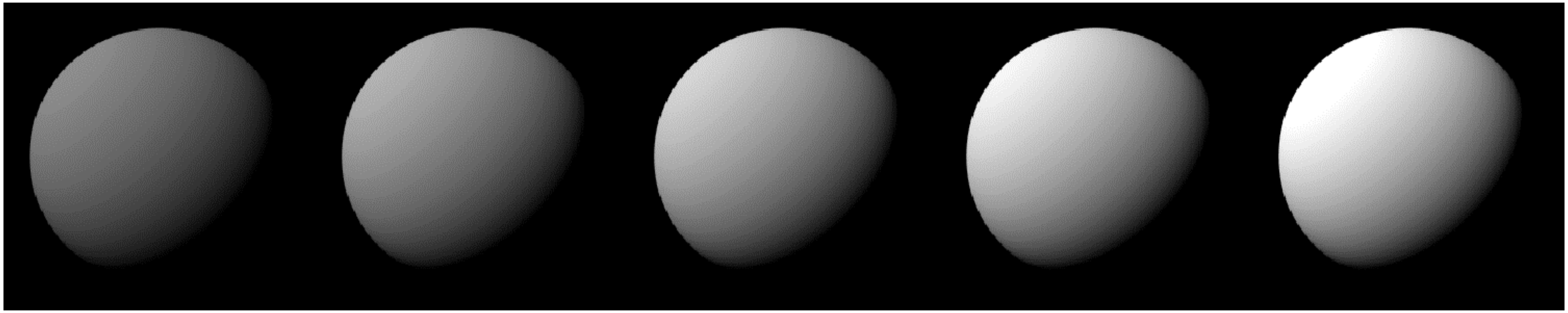
Diffuse materials and ray tracing





Diffuse materials and ray tracing

(by David Kurlander, Columbia University)



k_d = 0.4 0.55 0.7 0.85 1.0

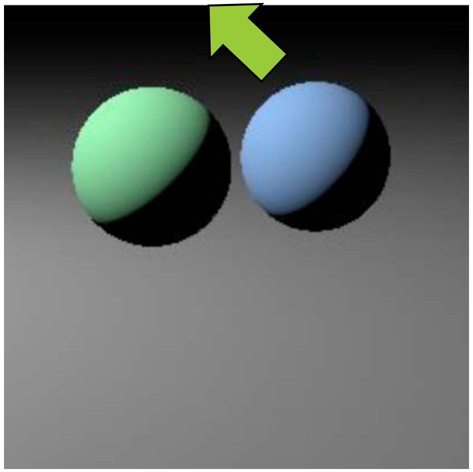
Ambient Light

Parts of the object are black?

- Trick: add a term to simulate indirect light in the scene

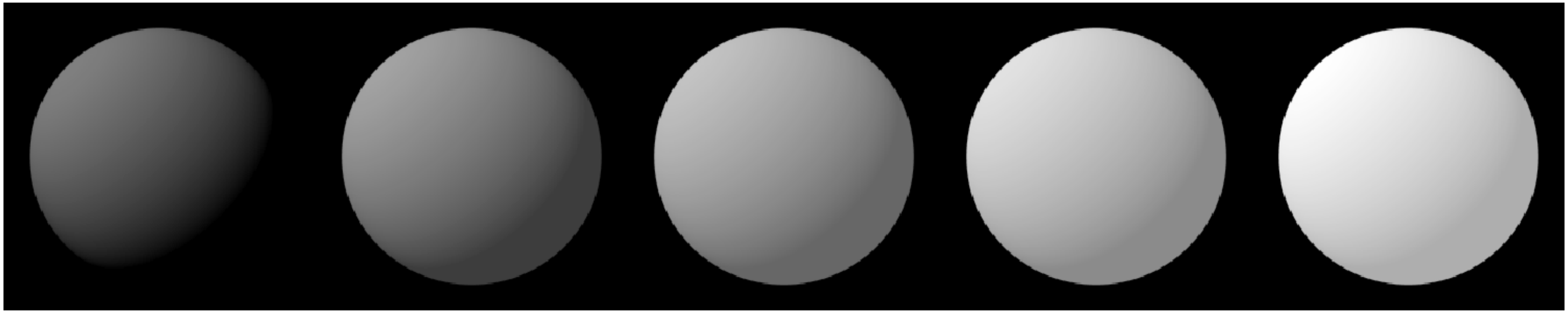
$$L = k_a c_d L_{ambient} c_{ambient} + \frac{k_d}{\pi} c_d L_{light} c_{light} \cos\theta$$

“Ambient light”
(independent of incoming direction of light)



Ambient Light

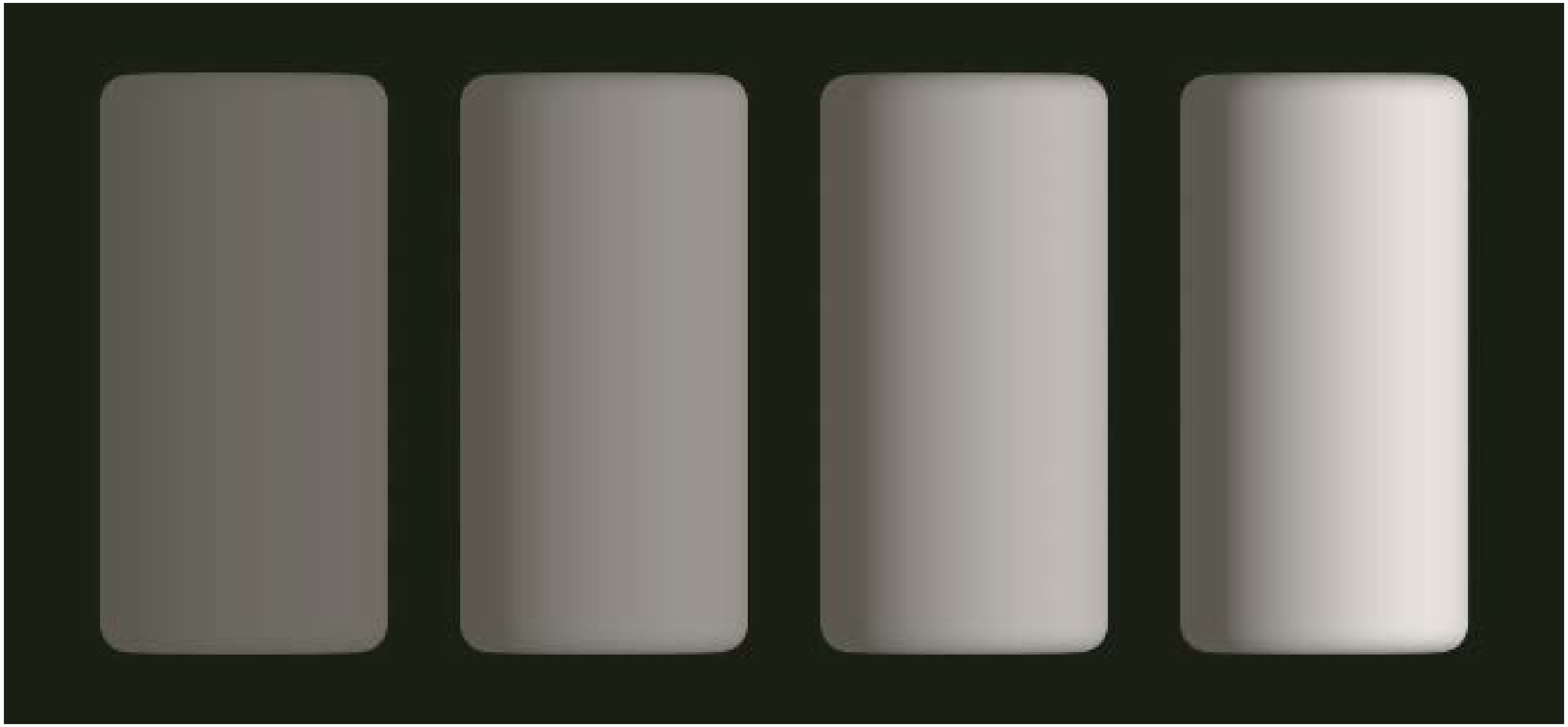
(by David Kurlander, Columbia University)



$k_a = 0.0$ 0.15 0.30 0.45 0.60

$k_d = 0.4$

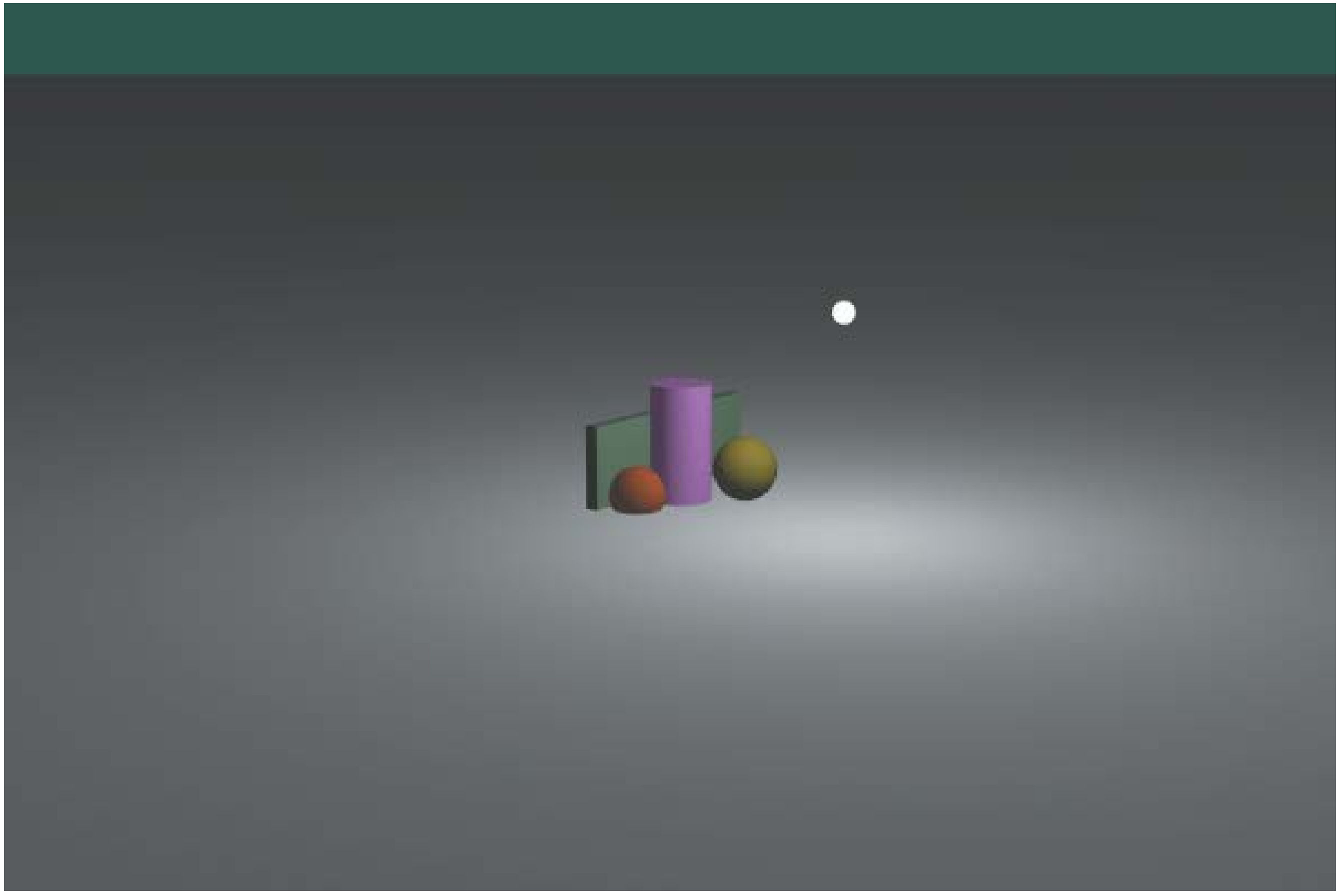
Ambient Light



$k_d =$ 0.1 0.5 0.75 1.0

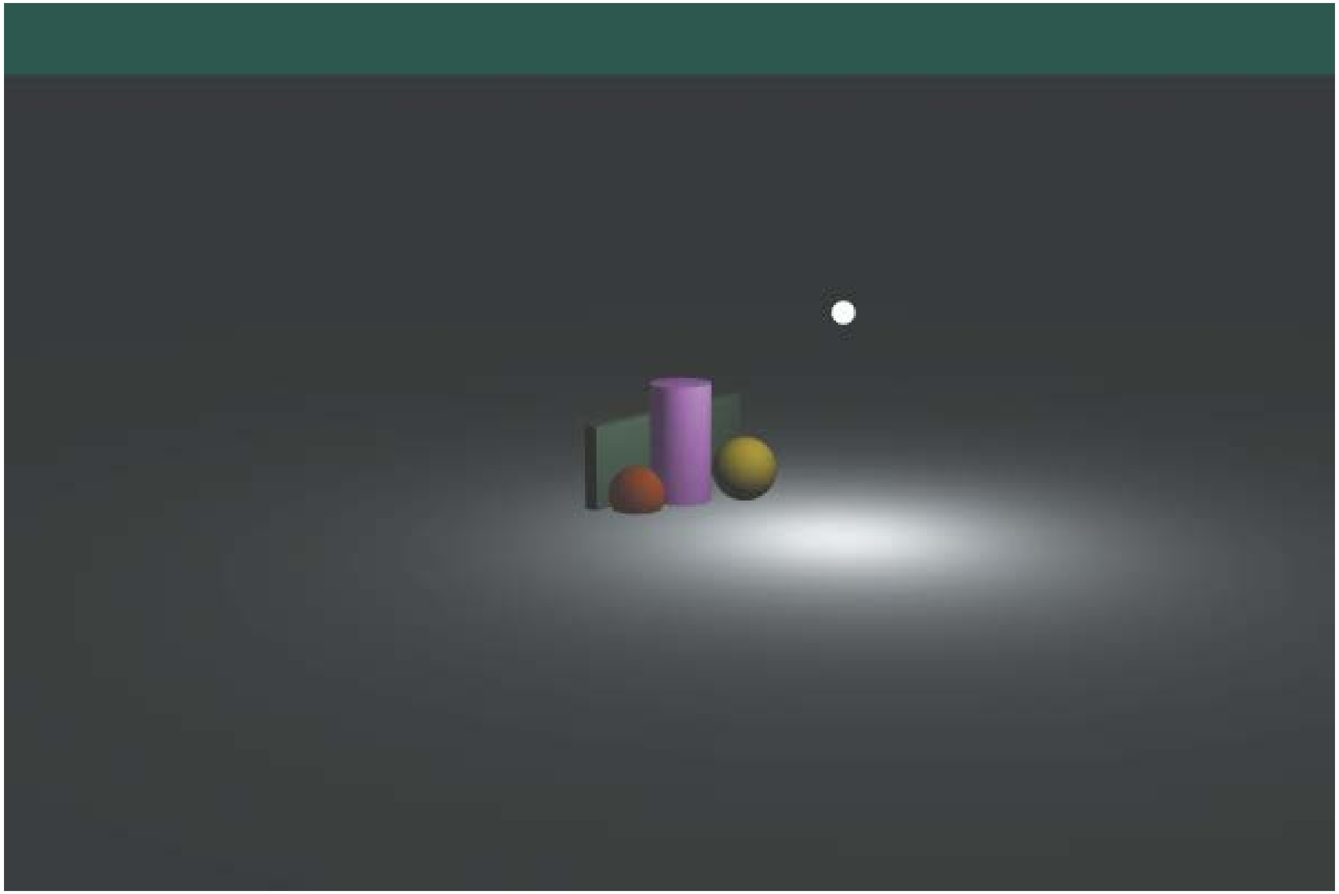
$k_a =$ 0.25

Types of light sources



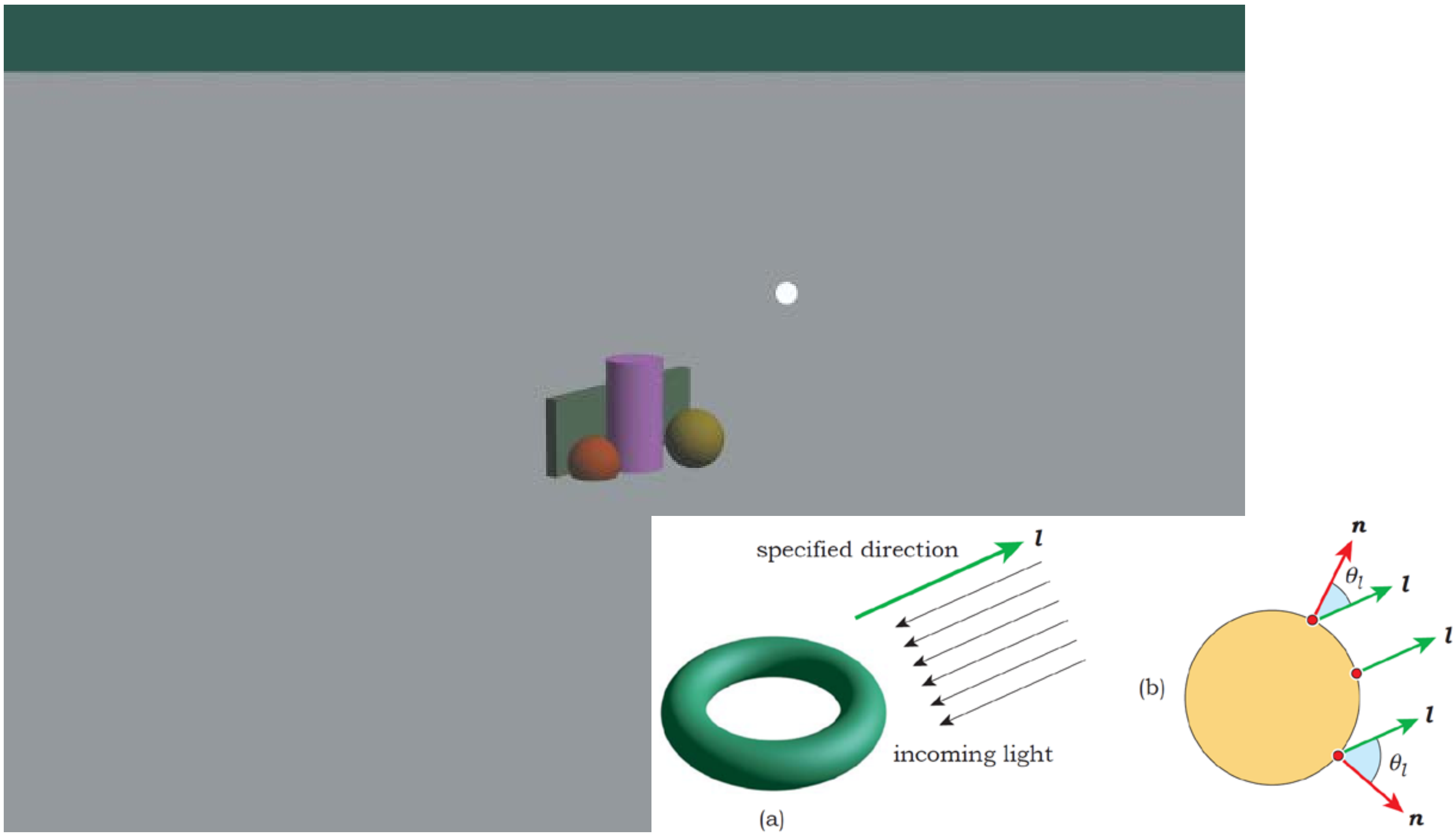
(point light)

Types of light sources



(point light with distance attenuation ~$1/r^2$)

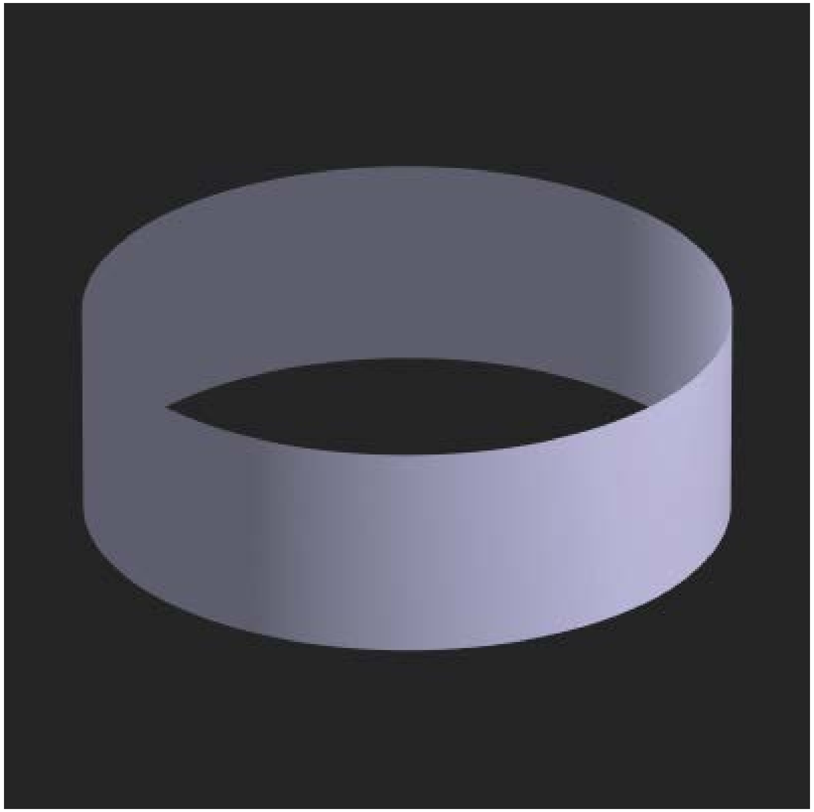
Types of light sources

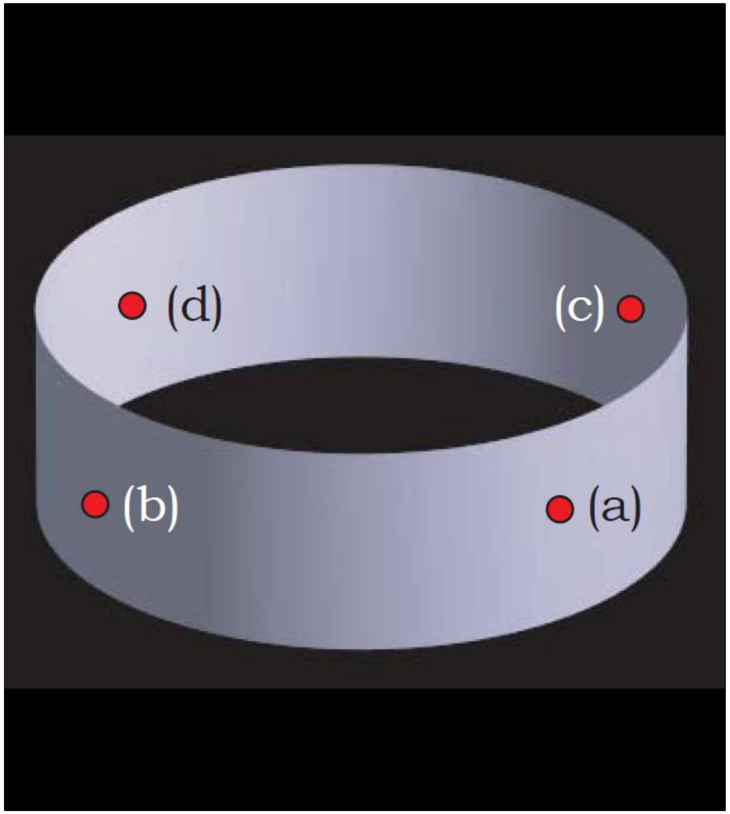


(directional light)

Diffuse Reflection: problem (1)

Does this shading look “right”?



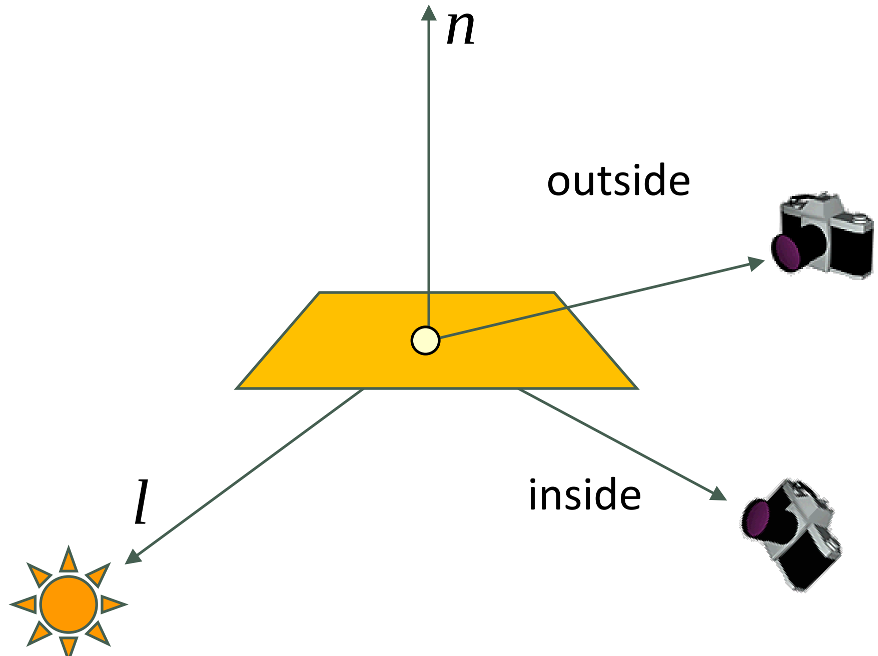


Diffuse Reflection: problem (1)

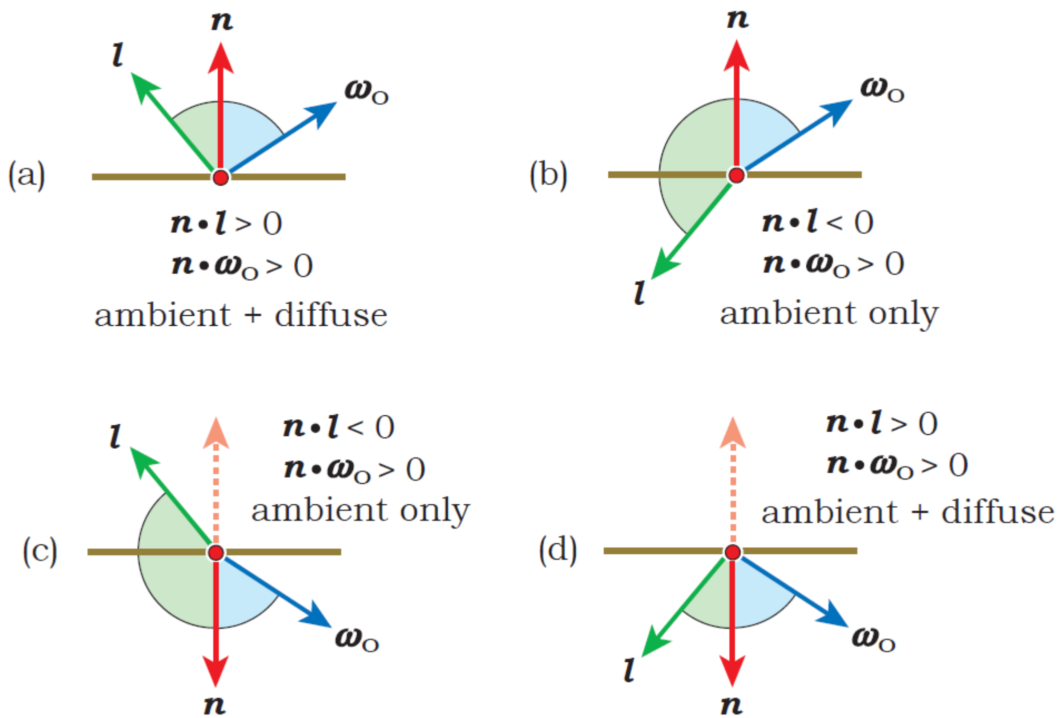
What if $\theta > \pi/2$?

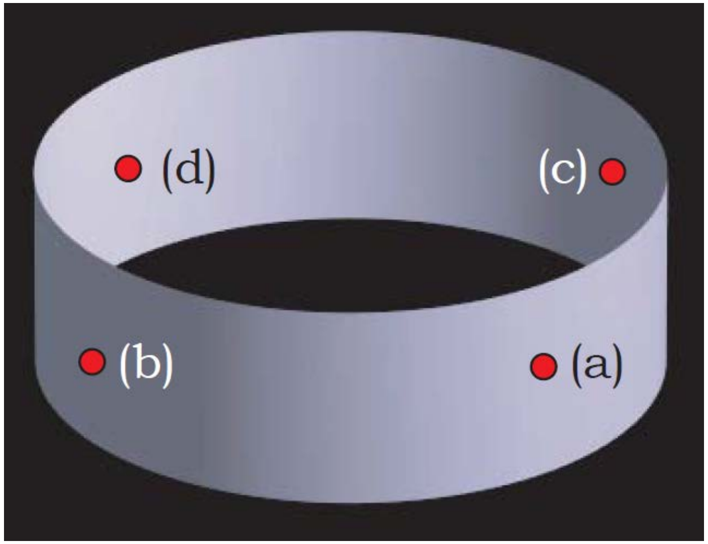
Is the surface “1-sided” or “2-sided”

- 1-sided: only the “outside” (normal vector) is shaded
- 2-sided: both outside and inside are shaded



Diffuse Reflection: problem (1)



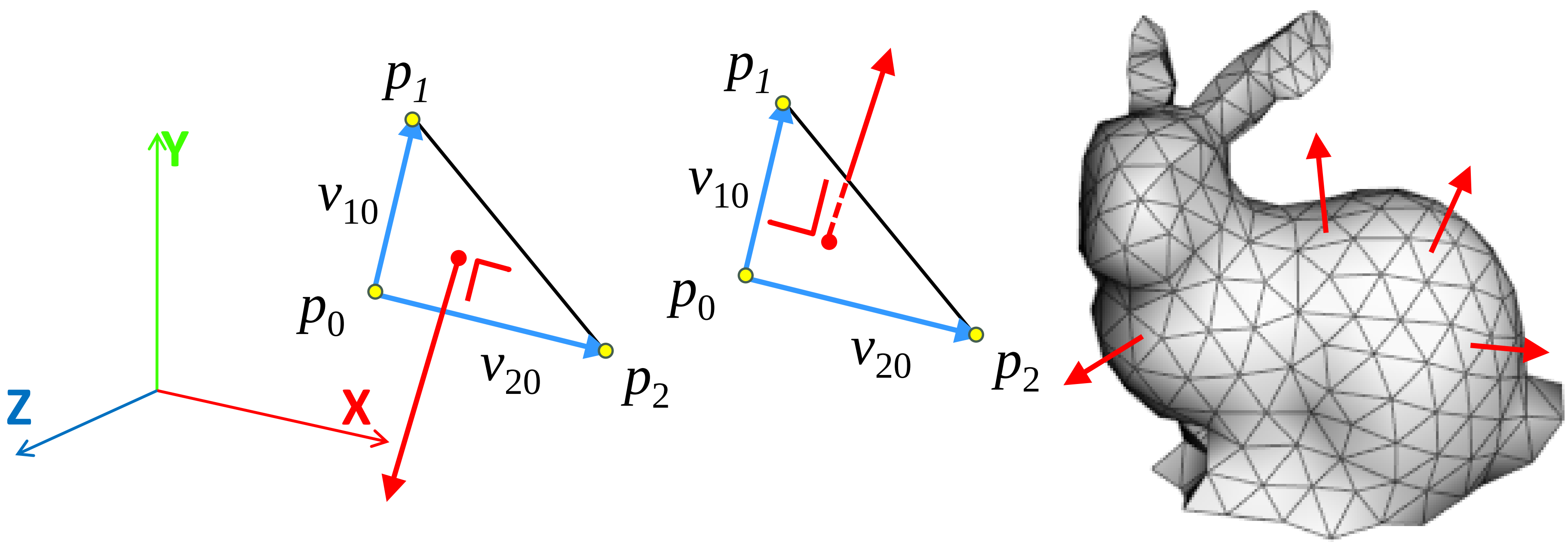


2-sided surfaces: check both light and camera direction vs normal vector

"Outside" of an object?

Convention: normal vectors are pointing "outwards"

- Looking from the outside of the triangle or object, the order of vertices in which a polygon is defined should be counterclockwise
- Normal vector can be computed using a proper vector product

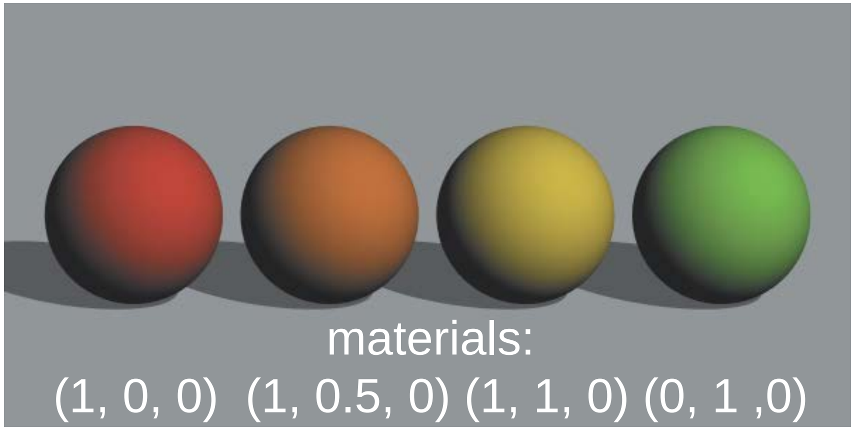


Diffuse Reflection: problem (2)

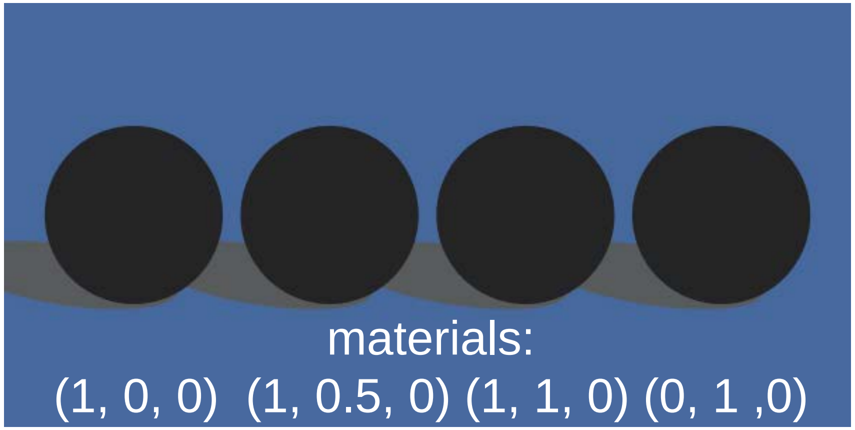
(r g b) color model

- Rgb color of light is multiplied by rgb color of surface

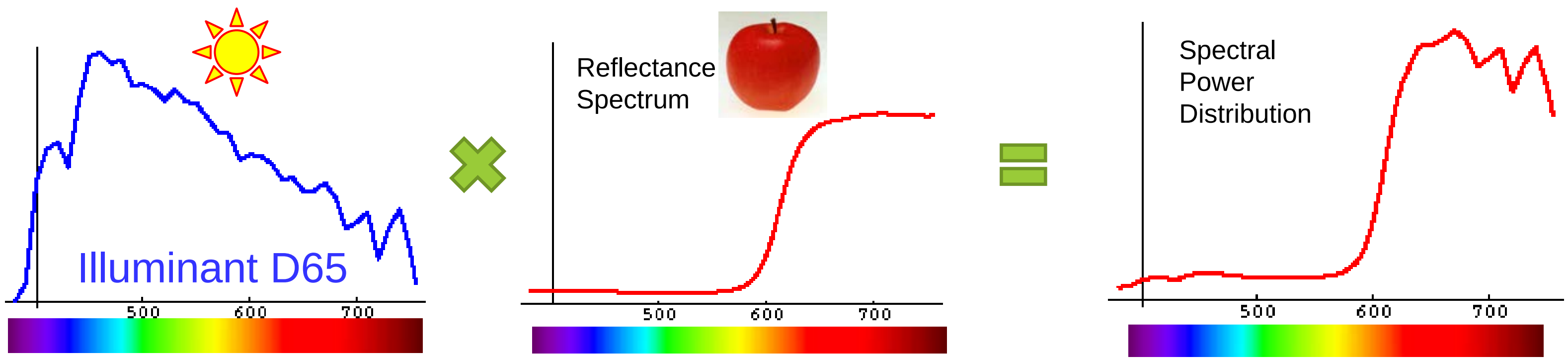
(white light) = (1, 1, 1)

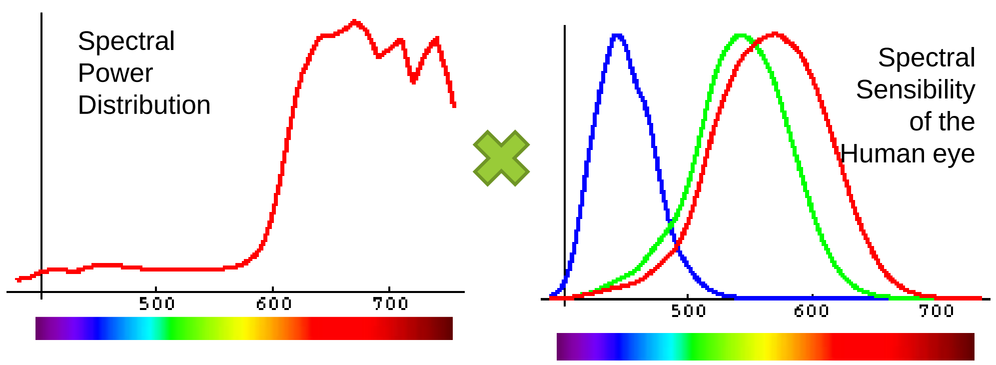


(blue light) = (0 , 0 , 1)



Diffuse Reflection: problem (2)



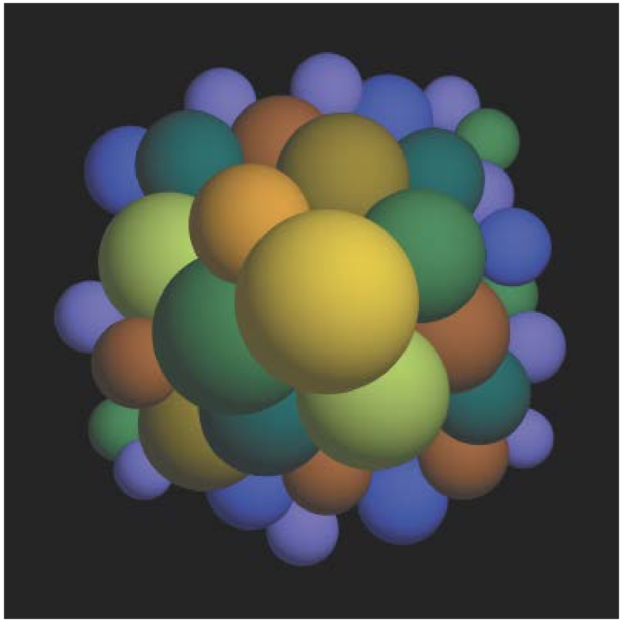


(rgb)

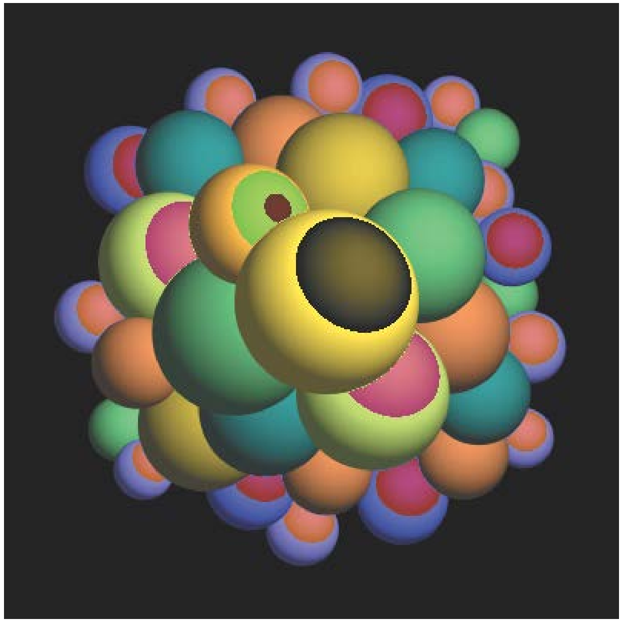
Diffuse Reflection: problem (3)

Adding (r g b) values (multiple lights, brighter lights)

- Color overflow due to added contributions



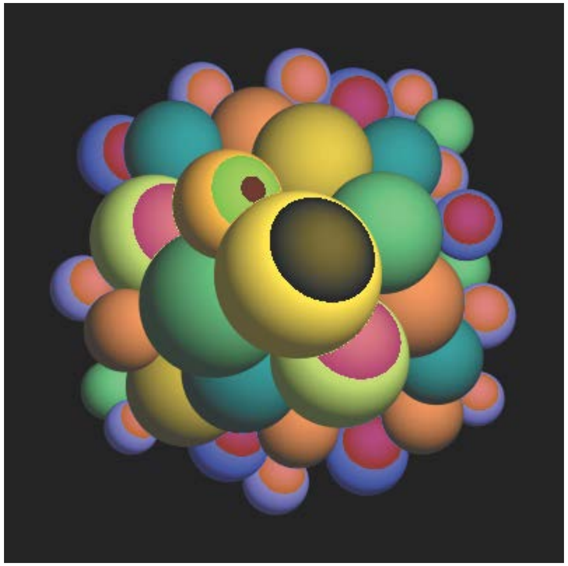
(lightsource
2x intensity)

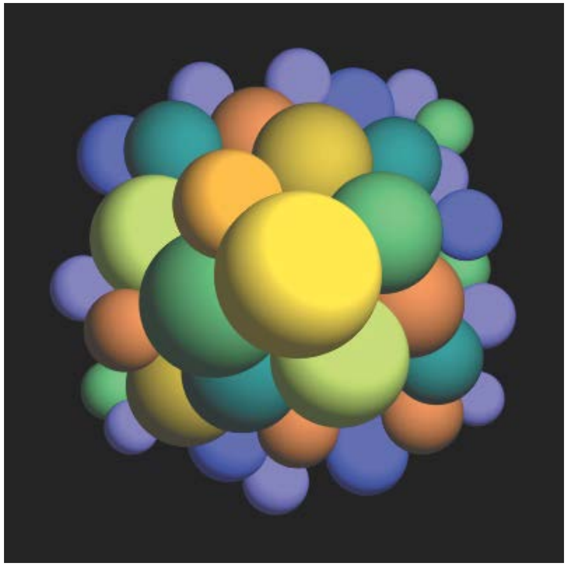


Diffuse Reflection: problem (3)

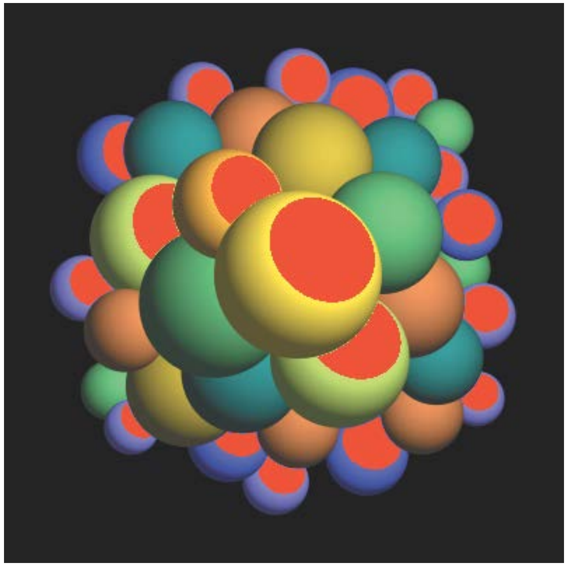
Tone -mapping operator:

- Transforms computed (color) intensities to limited valid range (e.g.[0,1])
- Simple approach: clamping





clamped image

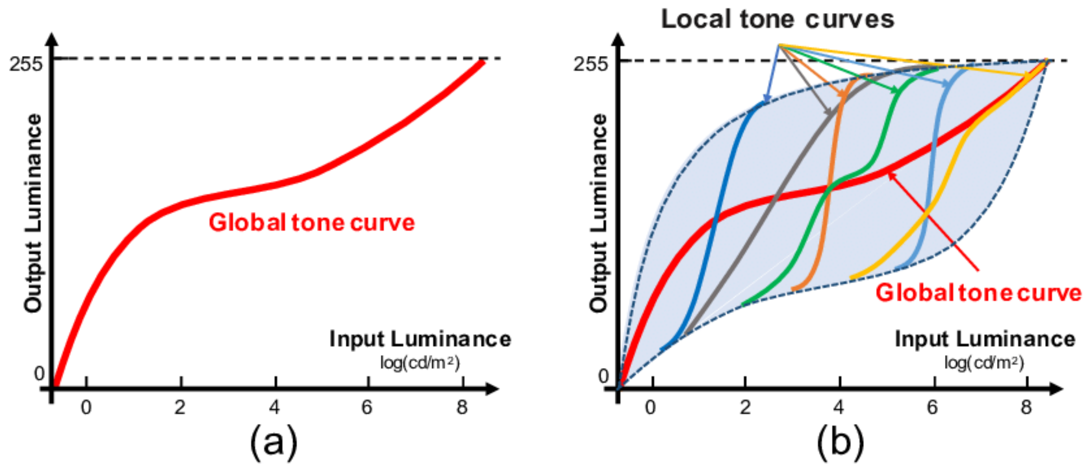


clamped regions shown in red

Diffuse Reflection: problem (3)

Tone-mapping operator:

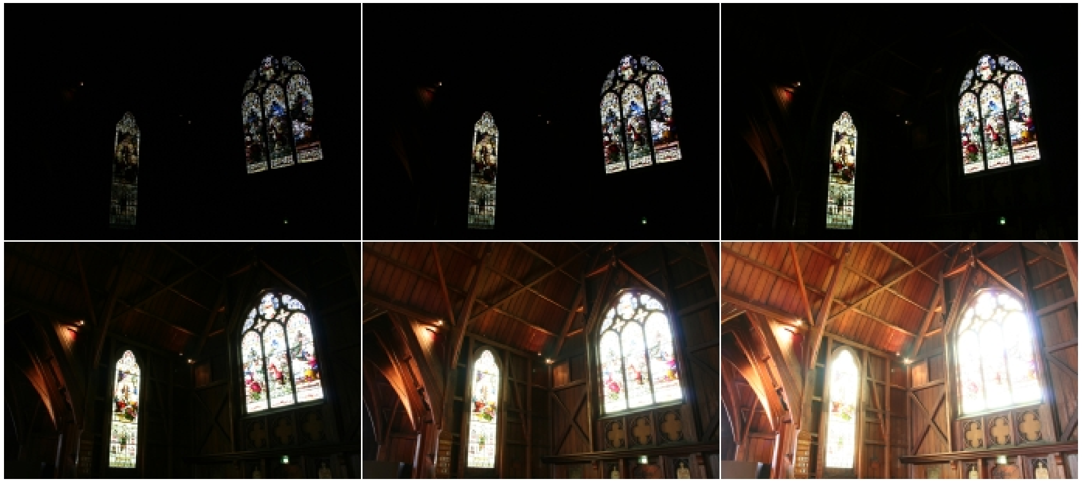
- Simple: e.g. $output = input/(input + 1)$ or $output = c.input^{\gamma}$
- Complex (local vs global)



(Real-time Tone Mapping: A State of the Art Report, March 2020)

Diffuse Reflection: problem (3)

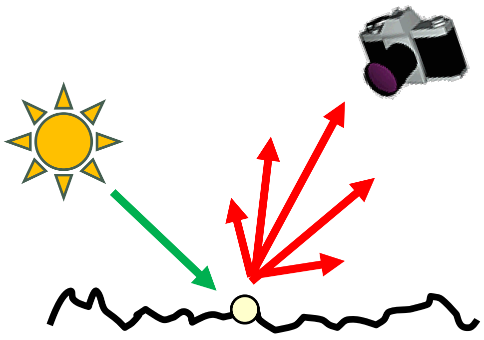
Tone -mapping operator:



Glossy ("specular") materials

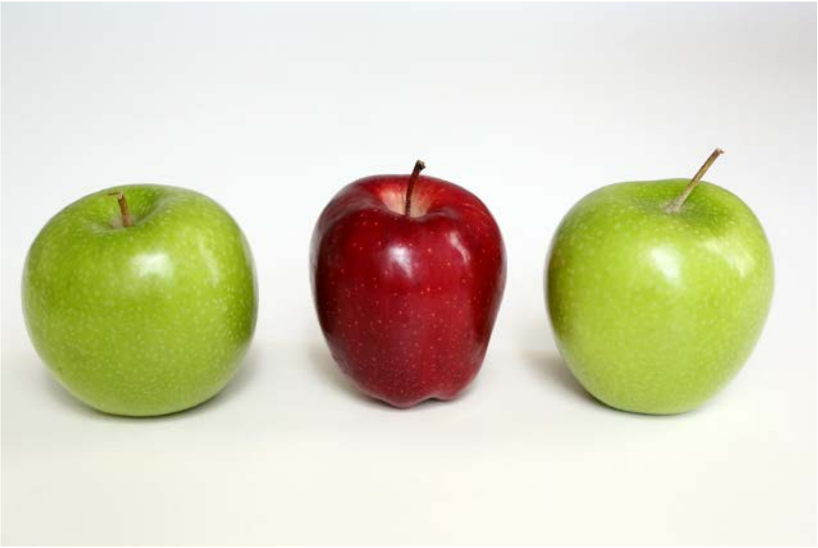
Shiny, glossy surfaces look different when seen from different viewpoints

- (highlights, specularities)



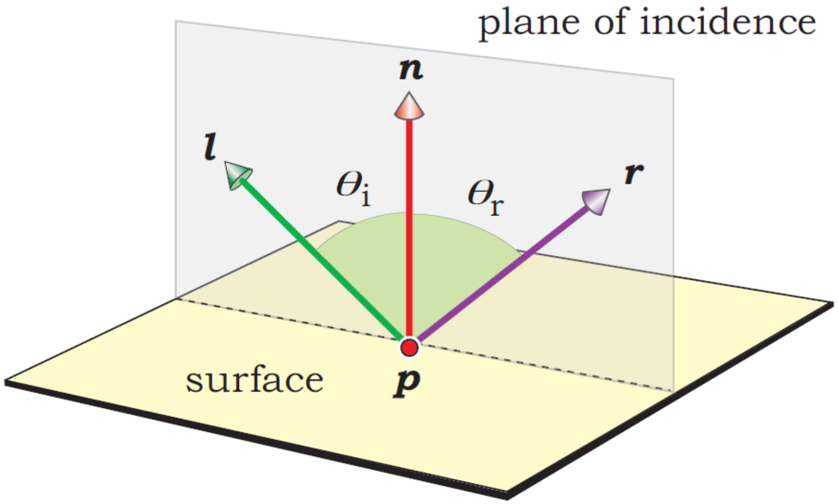
Light is reflected in a '*lobe*' of directions

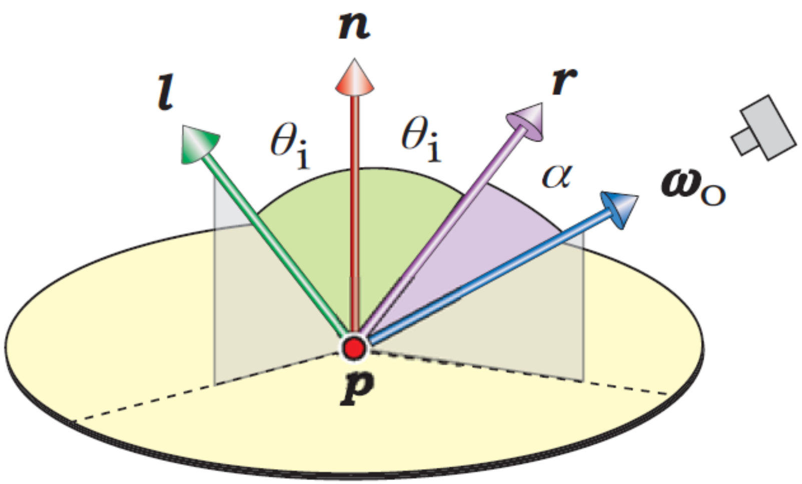
- (extreme case: mirror = perfectly specular)



Phong Material Model

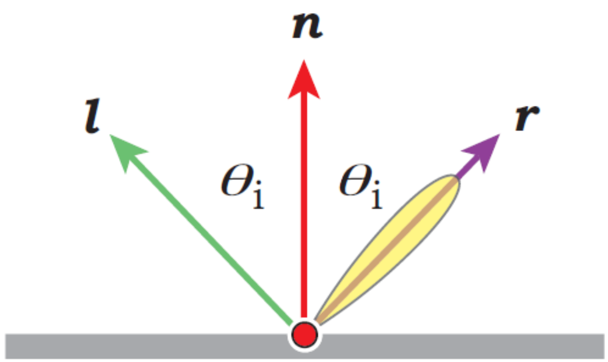
Model for glossy reflections

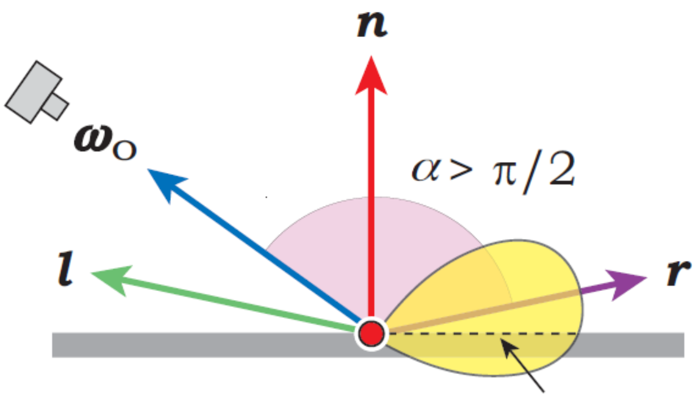


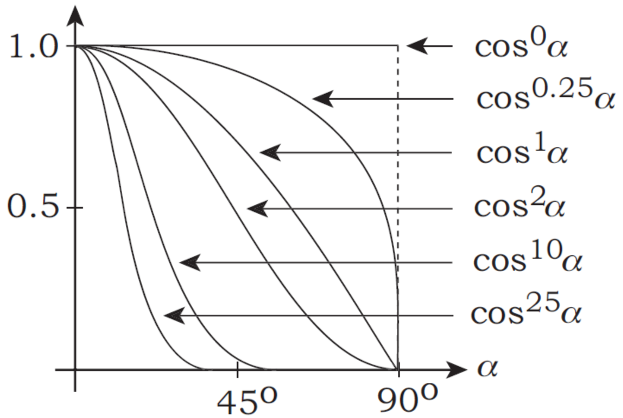


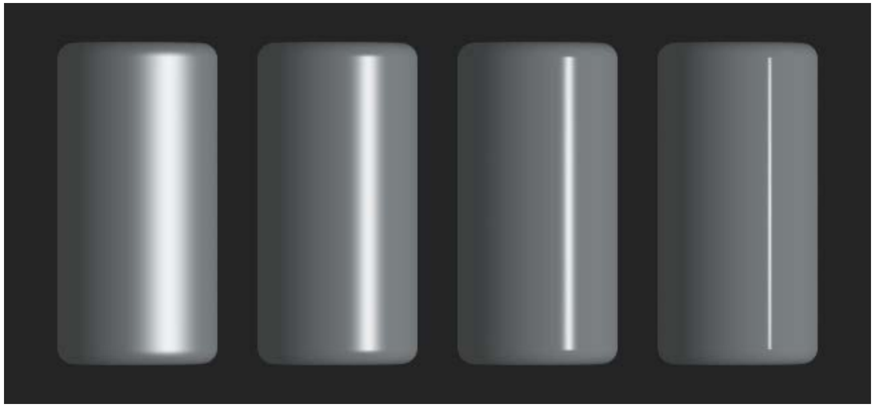
$$L = k_s(\cos\alpha)^e c_s L_{light} c_{light}$$

Phong Material Model

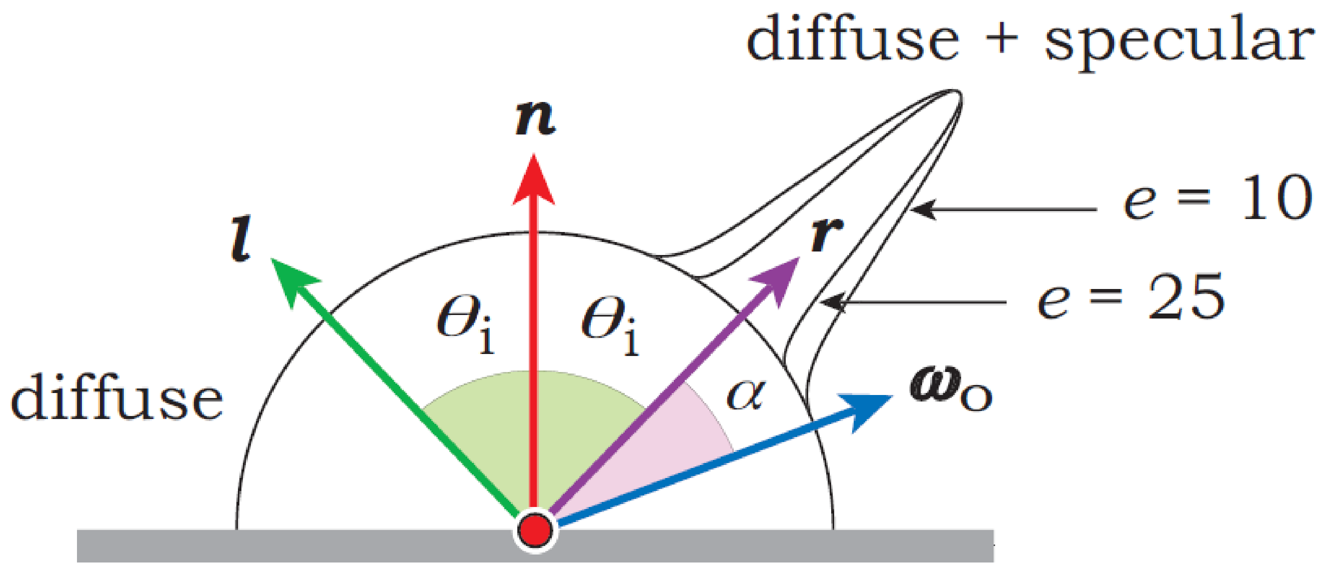






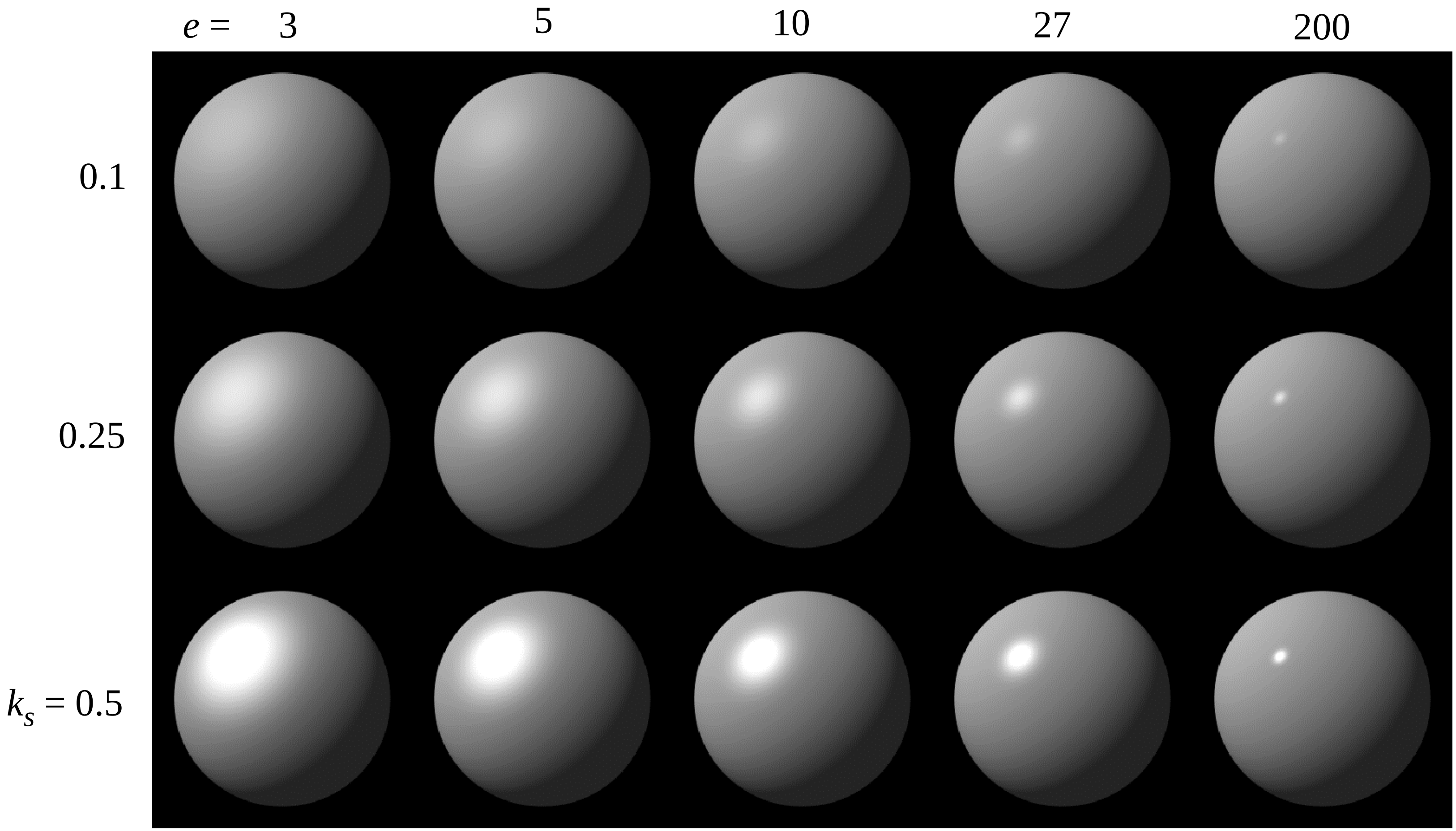


Phong Material Model

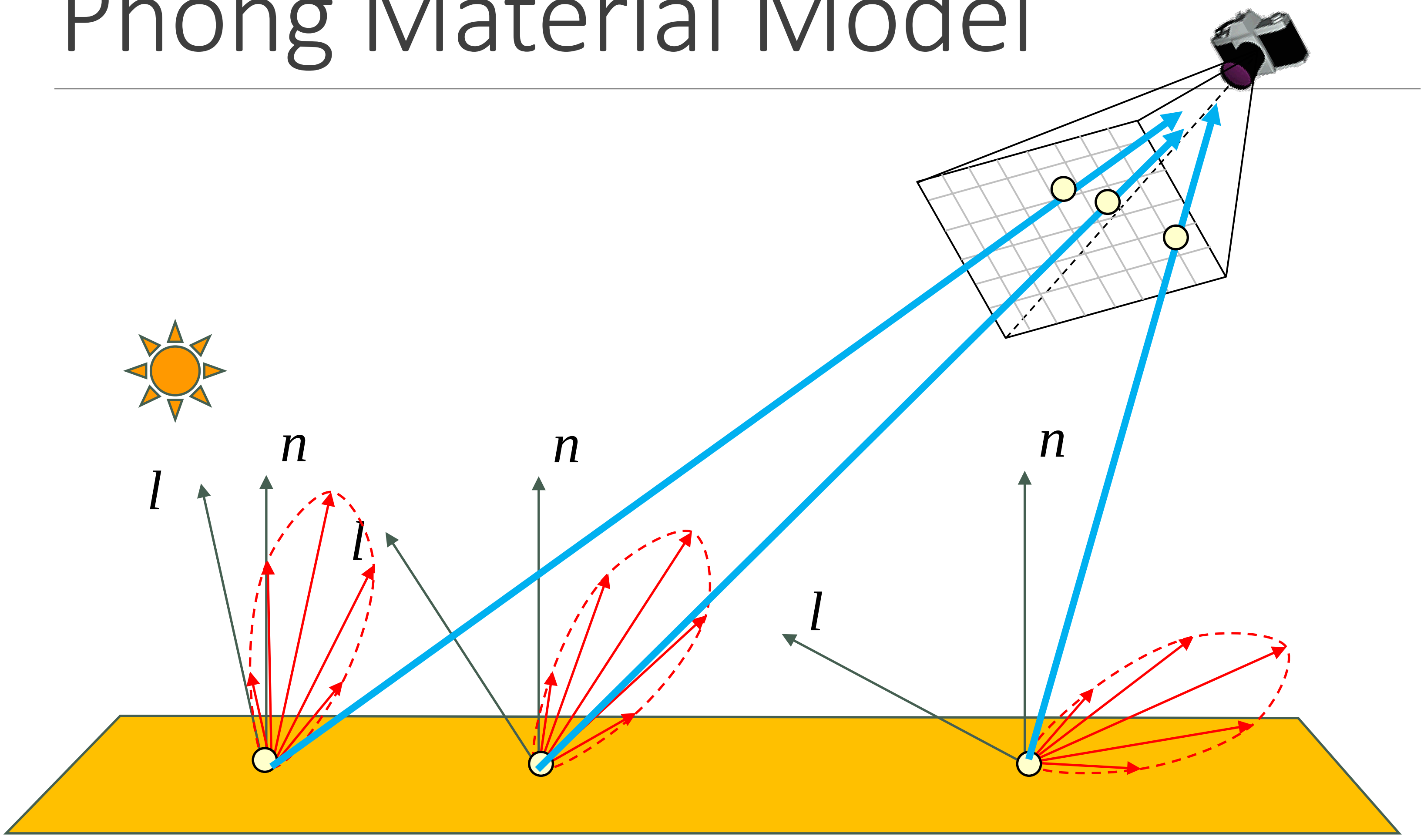


$$L = \frac{k_d}{\pi} c_d L_{light} c_{light} \cos\theta + k_s (\cos\alpha)^e c_s L_{light} c_{light}$$

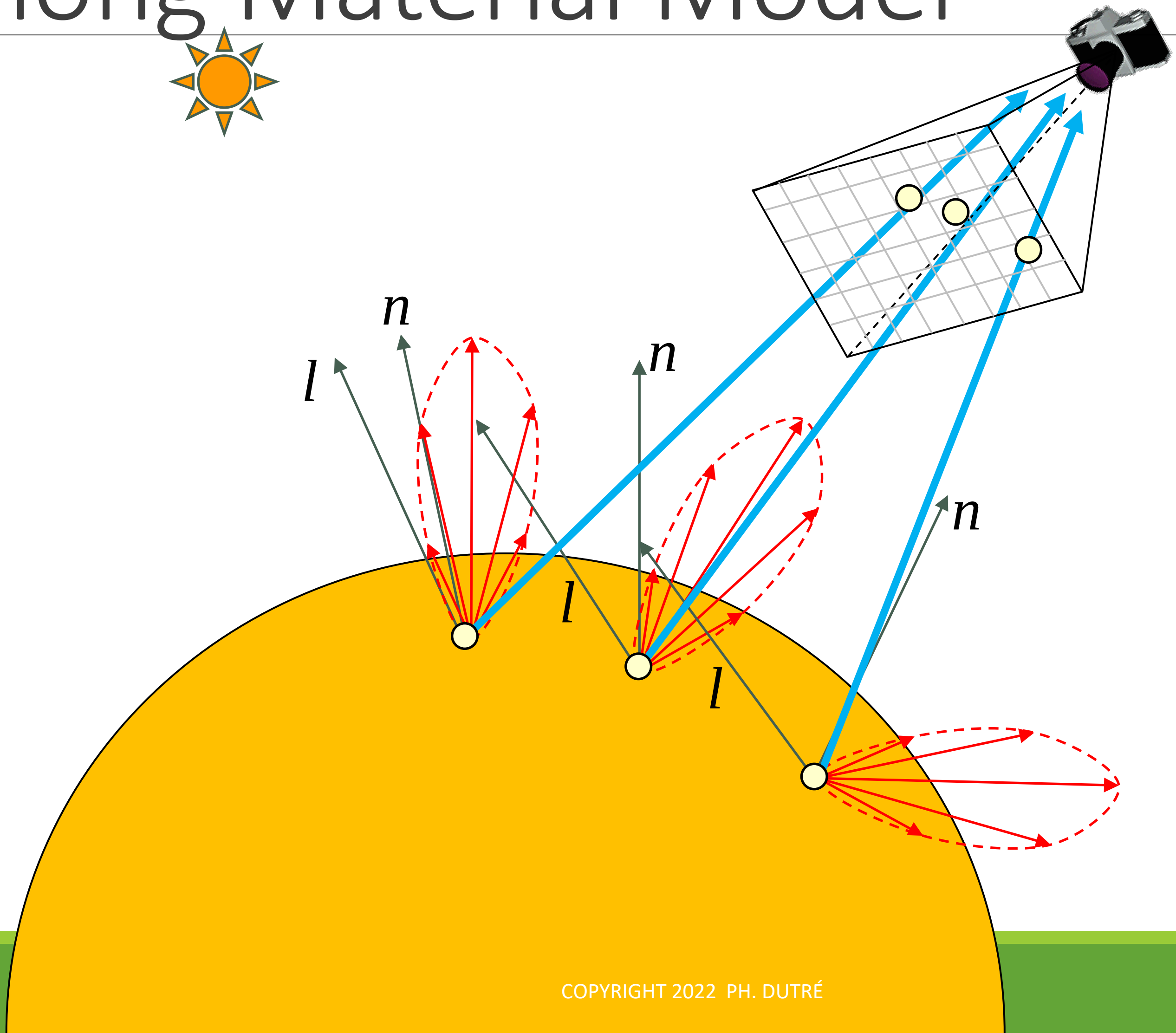
Phong Material Model



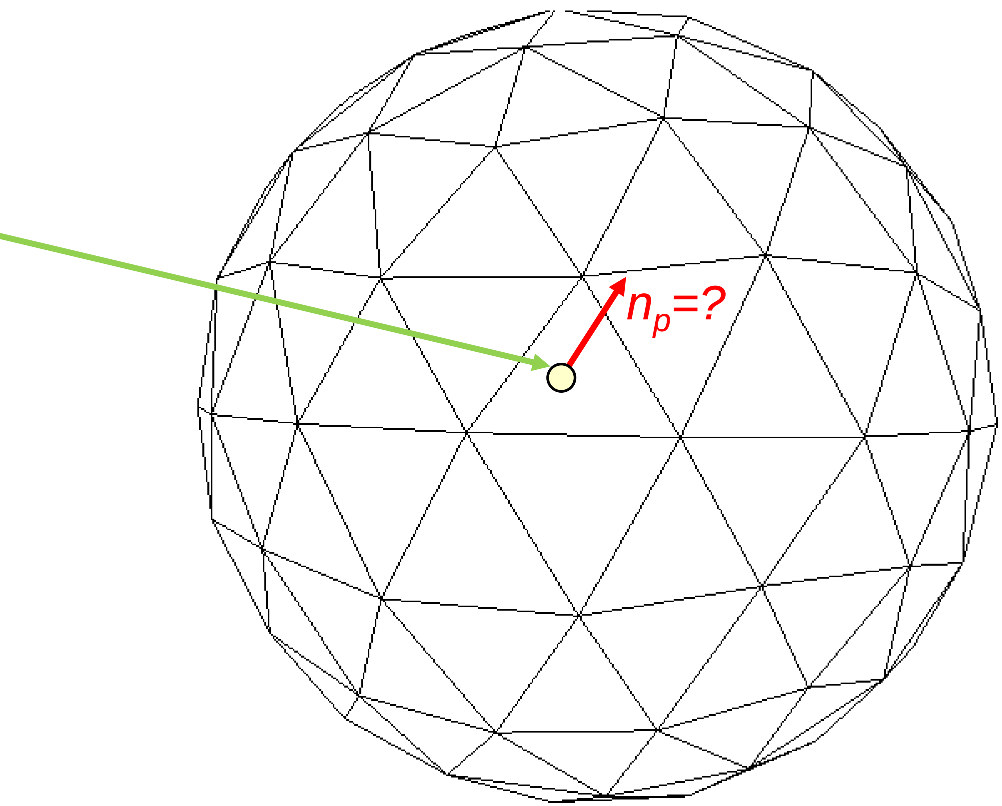
Phong Material Model



Phong Material Model



Polygon meshes and shading



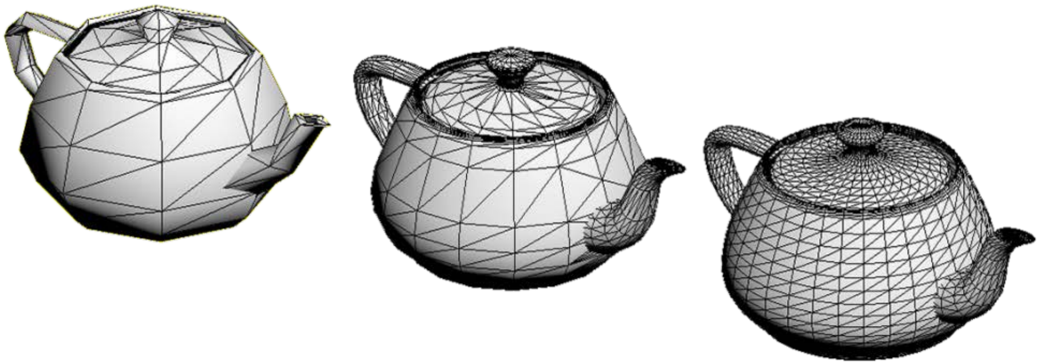
Normal vector n_p of a surface point inside a triangle with surface normal n_t?

- Default choice: $n_p = n_t$
- All points in triangle use same normal, and have (roughly) the same shading
- "Flat Shading"

Polygon meshes and shading

But:

- Some polygon meshes are modeled as an approximation of a curved surface

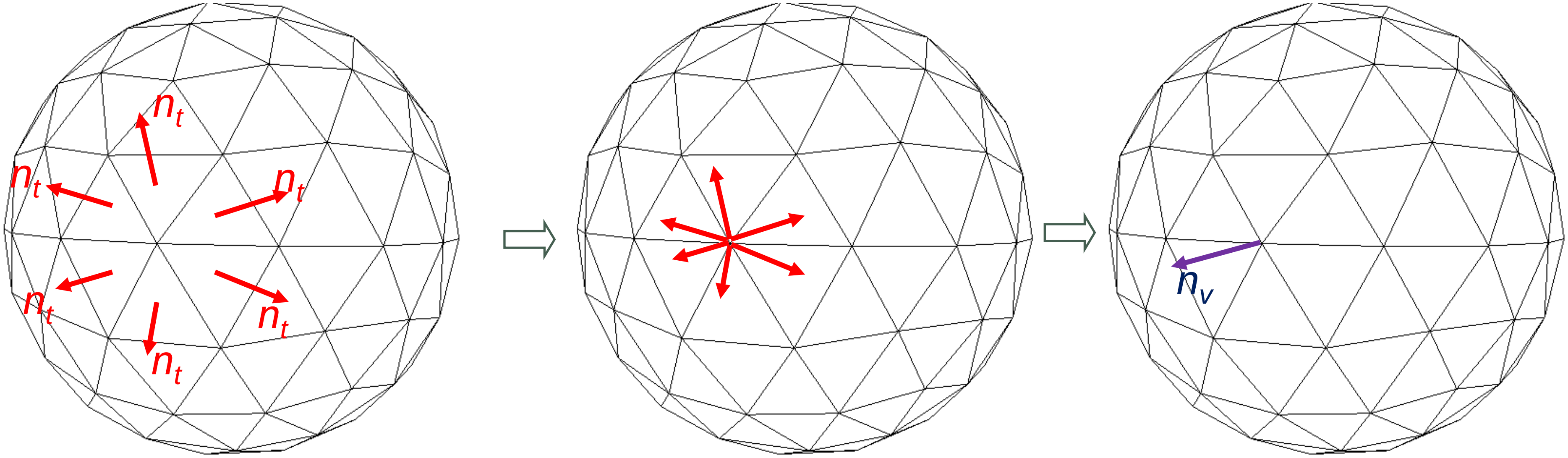


- … so geometry normal n_t on a triangle is an approximation of the "true normal" on the curved surface
- We would like to use a "shading normal" n_s in each point, that better reflects the curvature of the object
  - Shading normal is not necessarily perpendicular to the polygon mesh
  - Shading normal is used for shading, geometry normal for geometric computations

Polygon meshes and shading

Alternative A:

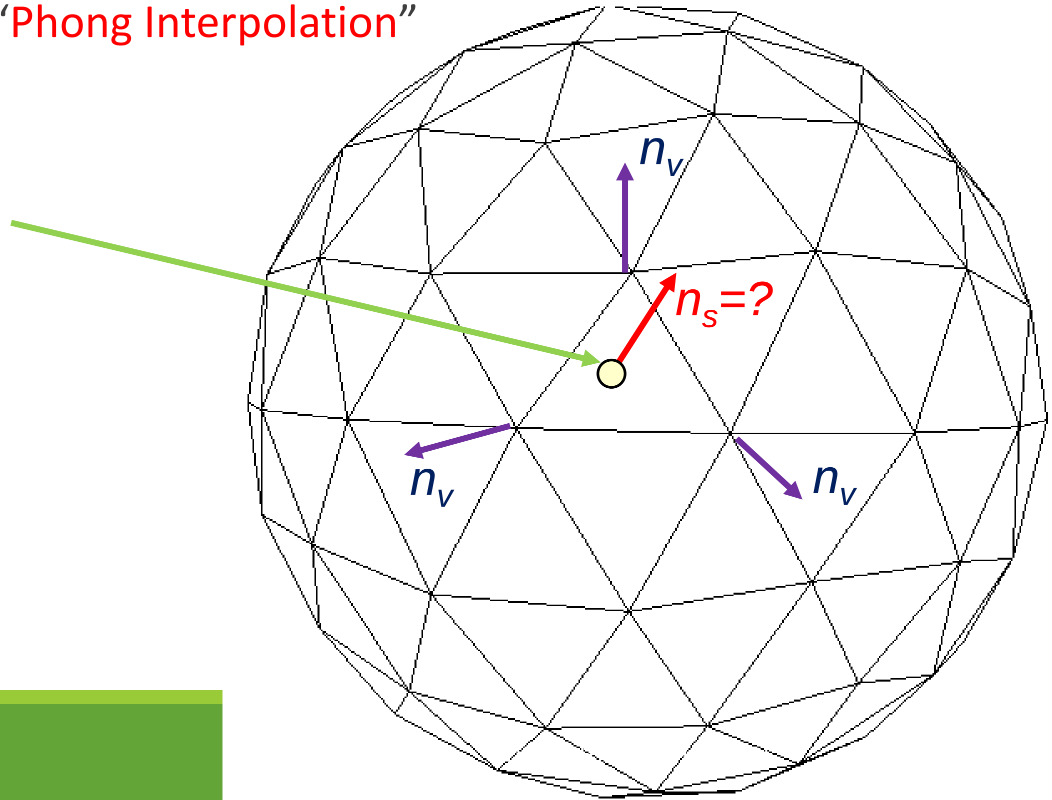
- Step 1: Compute normal n_v at each vertex
  - (weighted) average of geometry normal vectors n_t of adjoining triangles
  - makes sense if mesh is approximation of a curved surface



Polygon meshes and shading

Alternative A:

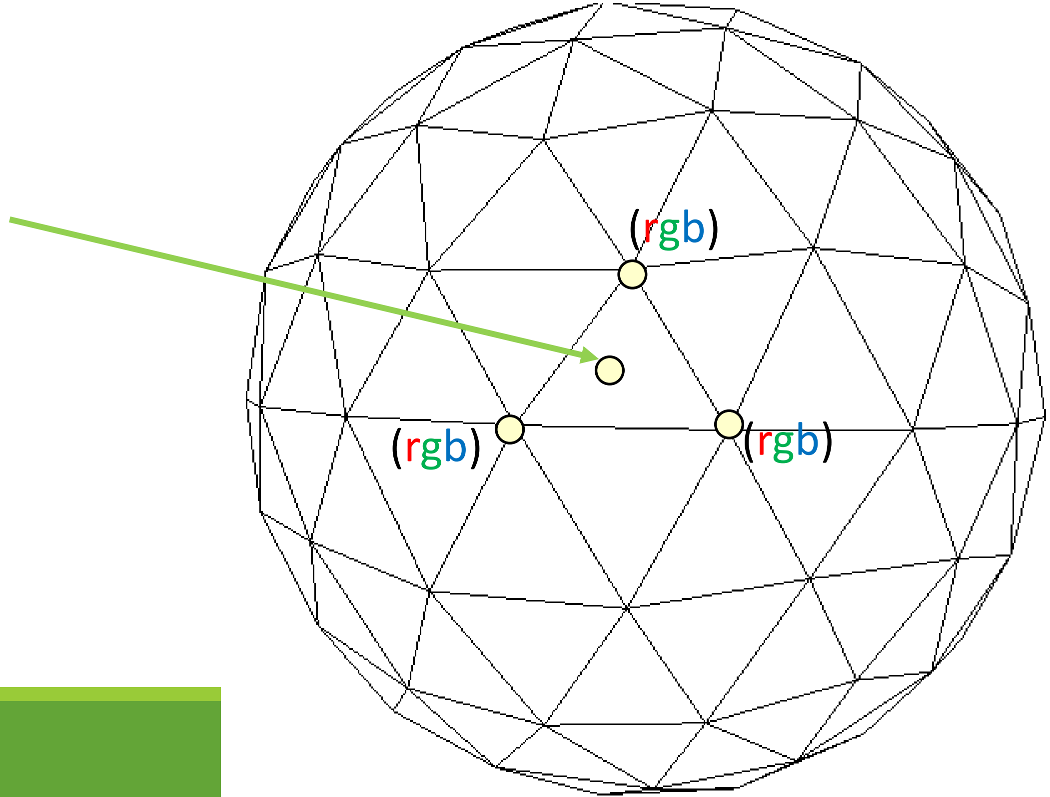
- Step 2: compute shading normal n_s as weighted average of n_v 's
- Use barycentric coordinates for interpolation (computed during ray-triangle intersection)
  ➔ "Phong Interpolation"



Polygon meshes and shading

Alternative B:

- (pre)compute shading at vertices
- Interpolate (pre)computed shading values, using barycentric coordinates
  ➔"Gouraud interpolation"



Polygon meshes and shading

Alterative A: Phong Interpolation

- Interpolate normal vectors at vertices for specific shading point, then compute shading
- More accurate (shading is computed separately at each point)
- More computation

Alternative B: Gouraud interpolation

- Compute shading at vertices, interpolate shading values for specific shading point
- Shading details not “captured” by vertices are lost
- Easier to compute
- Fits less well in ray tracing rendering engine

Many different variants exist …

Polygon meshes and shading