Lighting and Shading

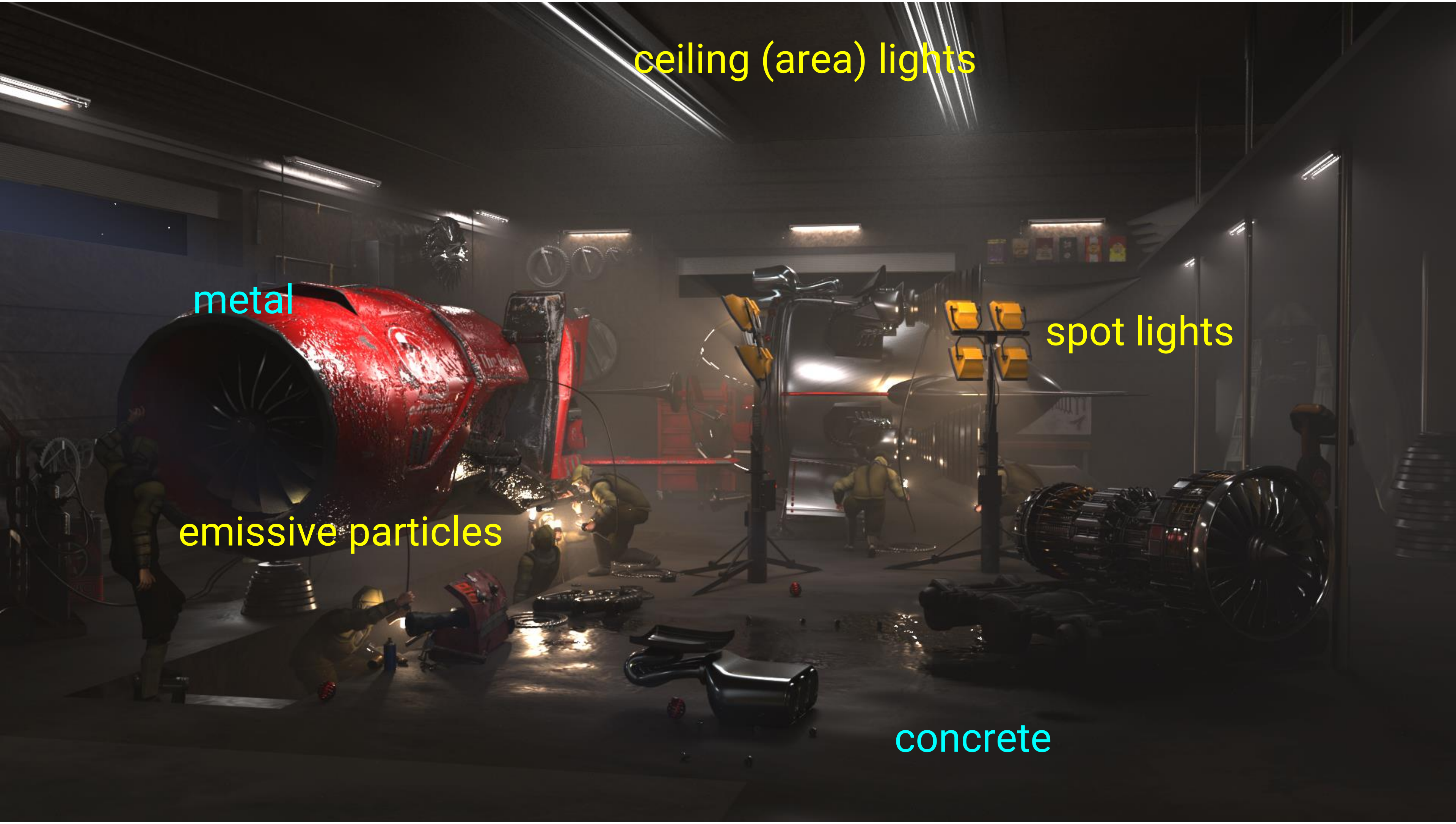
Introduction to Computer Graphics

Yu-Ting Wu

Recap.

- From week 2 to week 4, we introduced how a 3D shape shows up on the screen
- In the last week, we had a quick glance at the GPU graphics pipeline
- Next, we will talk about how to determine the fragment color
  - Lighting and shading
  - Texture mapping
  - Alpha blending for transparency objects

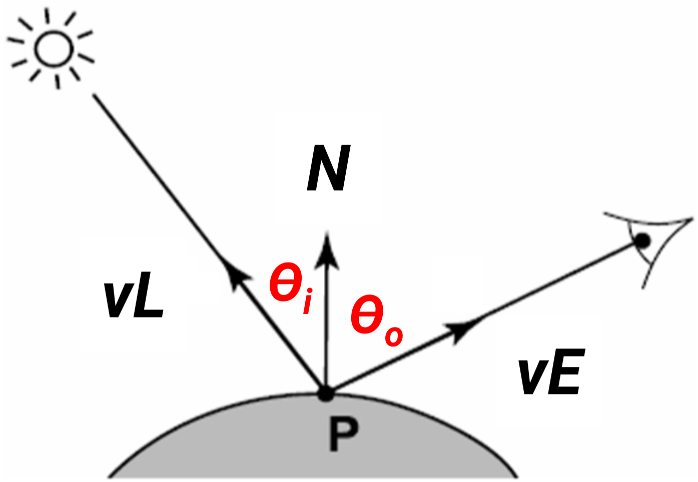
Shading: Materials and Lighting



Shading: Materials and Lighting (cont.)

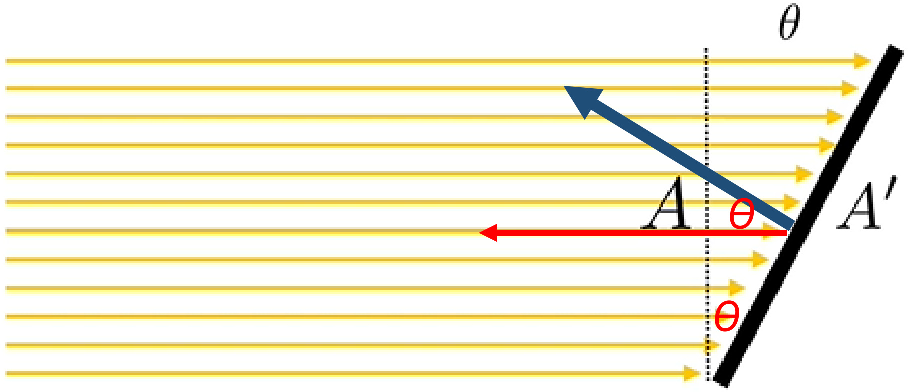
Shading

- Shading refers to the process of altering the color of an object/surface/polygon in the 3D scene
- In physically-based rendering, shading tries to approximate the **local behavior** of lights on the object's surface, based on things like
  - Surface orientation (normal) ***N***
  - Lighting direction ***vL*** (and θ_i)
  - Viewing direction ***vE*** (and θ_o)
  - Material properties
  - Participating media
  - etc.



Lambertian Cosine Law

- Illumination on an oblique surface is less than on a normal one
- Generally, illumination falls off as cosθ



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

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

Lights

Lights in Computer Graphics

- Point light
- Spot light
- Area light

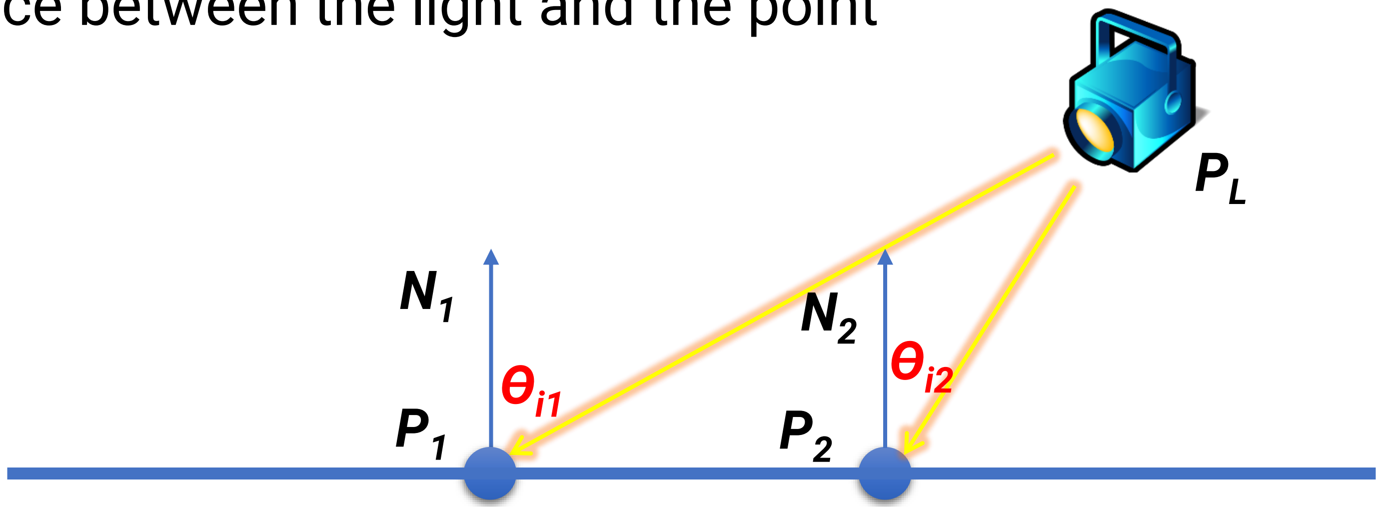
local lights

- Directional light
- Environment light

distant lights

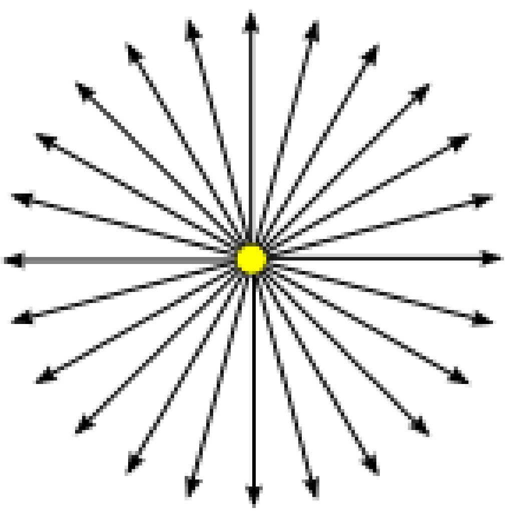
Local Light

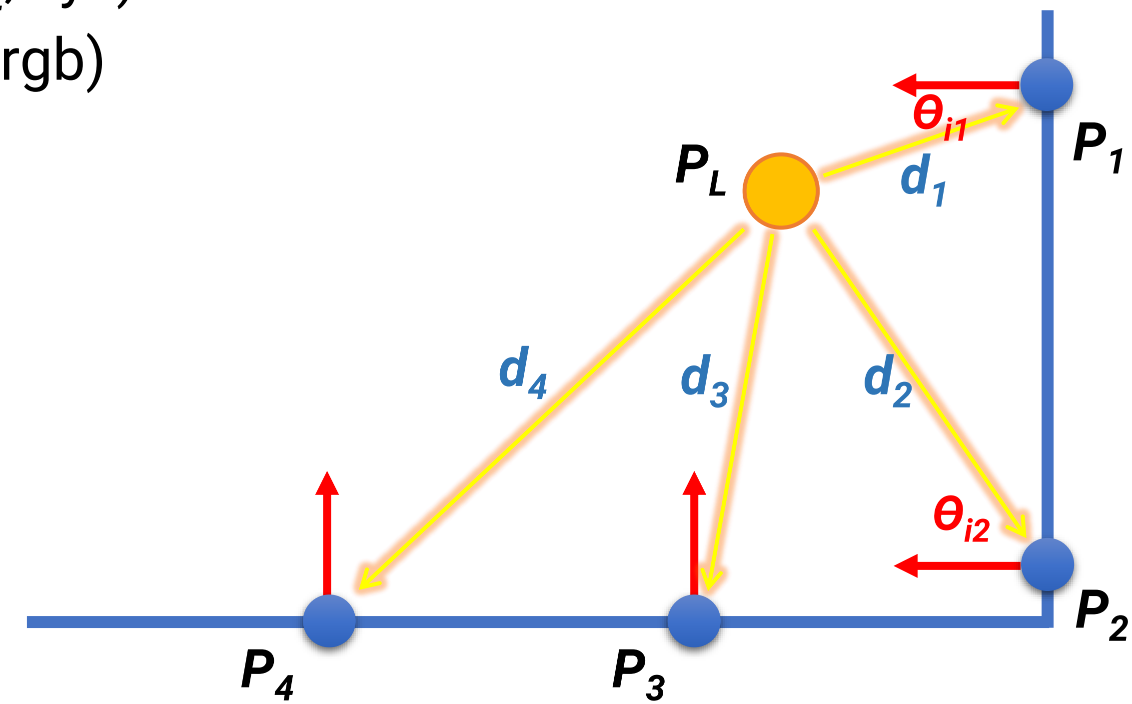
- The distance between a light and a surface is **not** long enough compared to the scene scale
- The position of light needs to be considered during shading
    - **Lighting direction $vL = |P_L - P|$**
    - **Lighting attenuation** is proportional to the square of the distance between the light and the point



Point Light

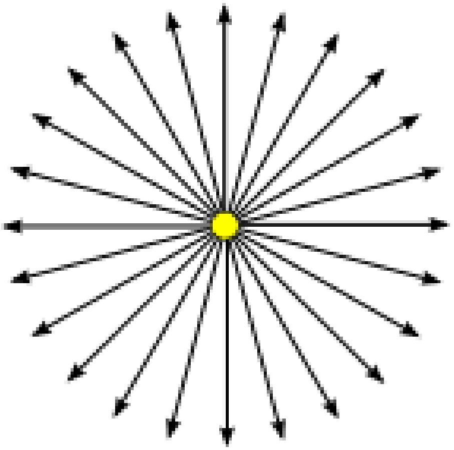
- An isotropic point light source that emits the same amount of light in all directions
- Described by
  - Light position (${\boldsymbol{P}_L}$, xyz)
  - Light intensity ($\boldsymbol{I}$, rgb)





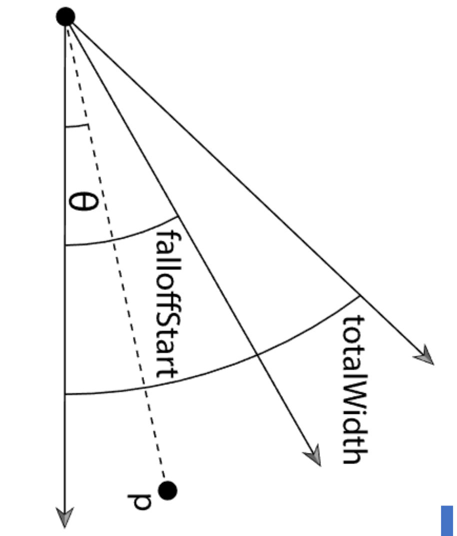
Point Light (cont.)

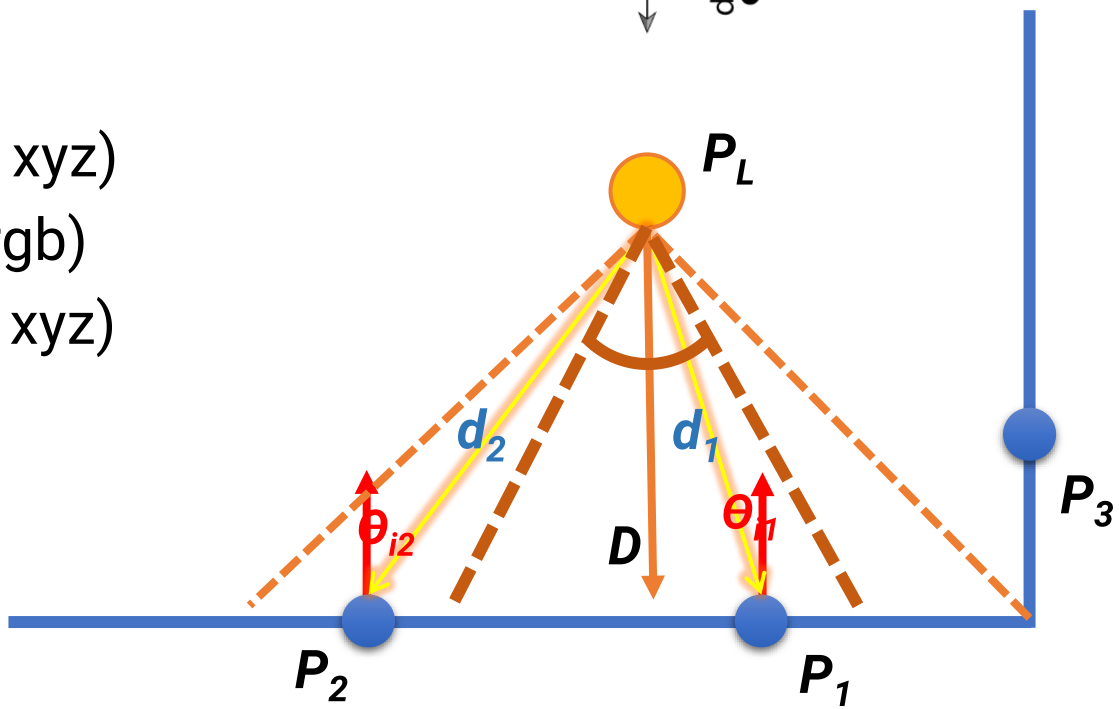
A scene illuminated by a point light



Spot Light

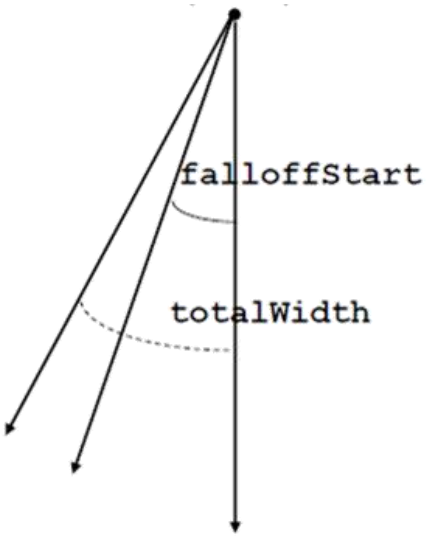
- A handy variation on point lights
- Rather than shining illumination in all directions, it emits light in a cone of directions from its position
- Described by
  - Light position (***P_L***, xyz)
  - Light intensity (***I***, rgb)
  - Light direction (***D***, xyz)
  - TotalWidth
  - FalloffStart





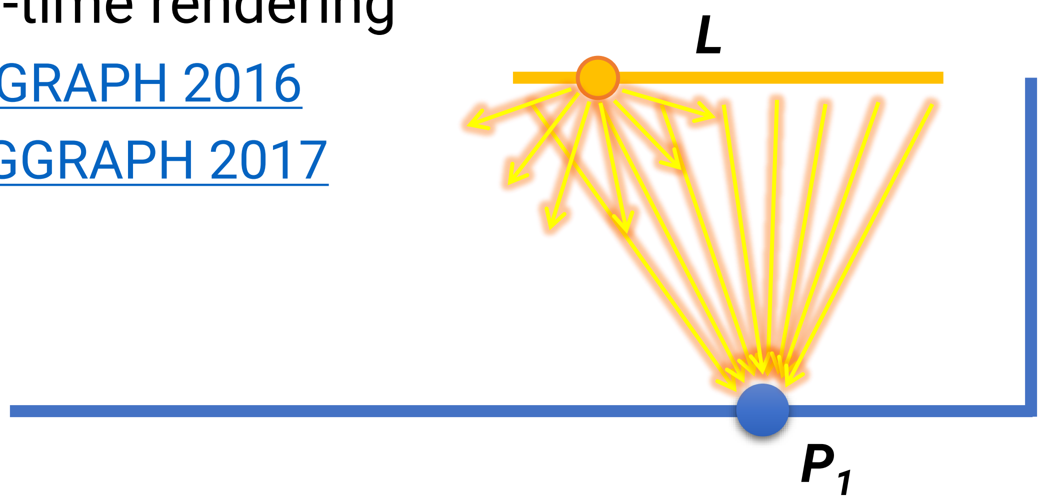
Spot Light (cont.)

A scene illuminated by a spot light



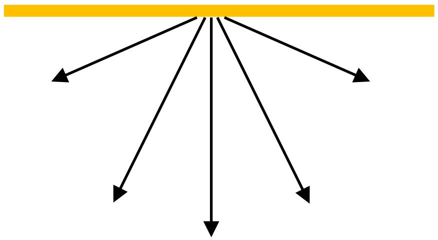
Area Light

- Defined by one or more **shapes** that emit light from their surface, with some directional distribution of energy at each point on the surface
- Require **integration** of lighting contribution across the light surface
  - In offline rendering, usually estimated by sampling
  - Expensive for real-time rendering
    - [Heitz et al., SIGGRAPH 2016]()
    - [Dupuy et al., SIGGRAPH 2017]()



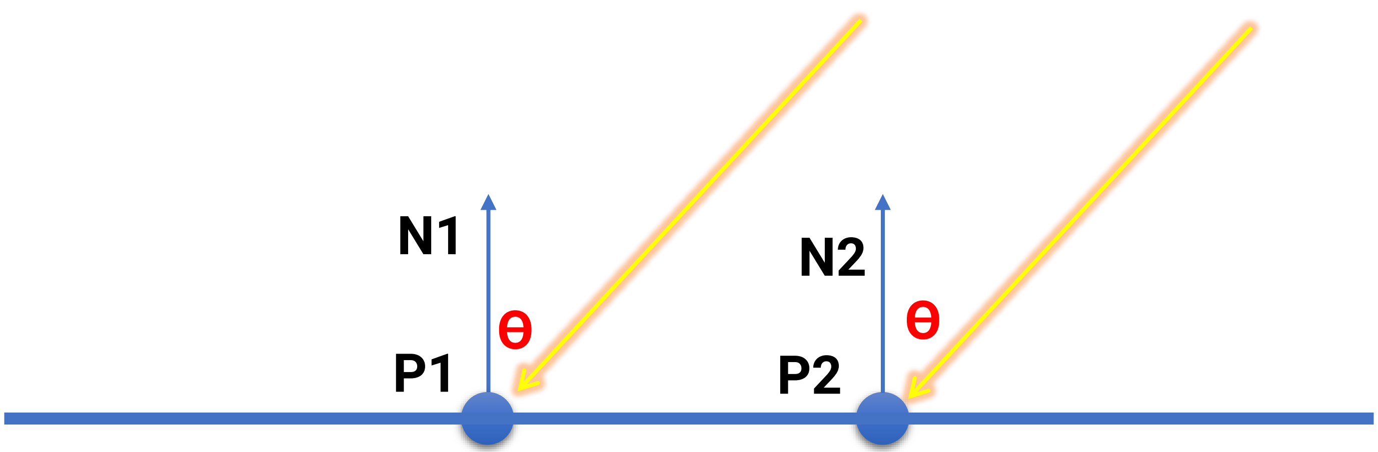
Area Light (cont.)

A scene illuminated by an area light



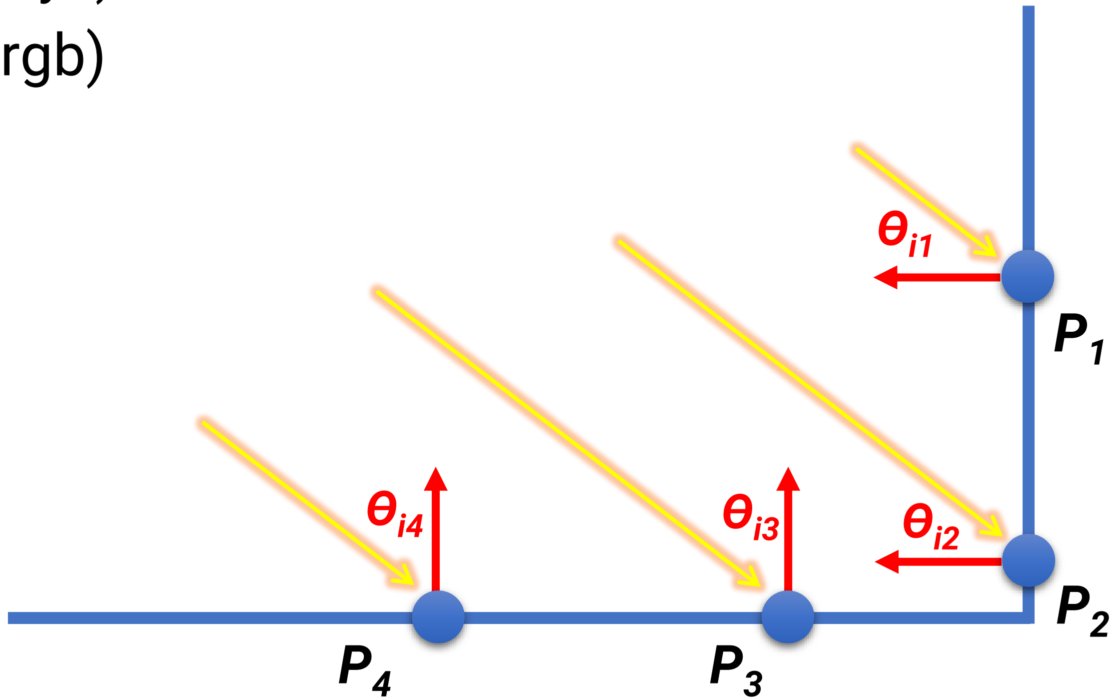
Distant Light

- The distance between a light and a surface is long enough compared to the scene scale and **can be ignored**
  - **Lighting direction** is **fixed**
  - **No lighting attenuation**
- **Directional light (sun)** is the most common distant light



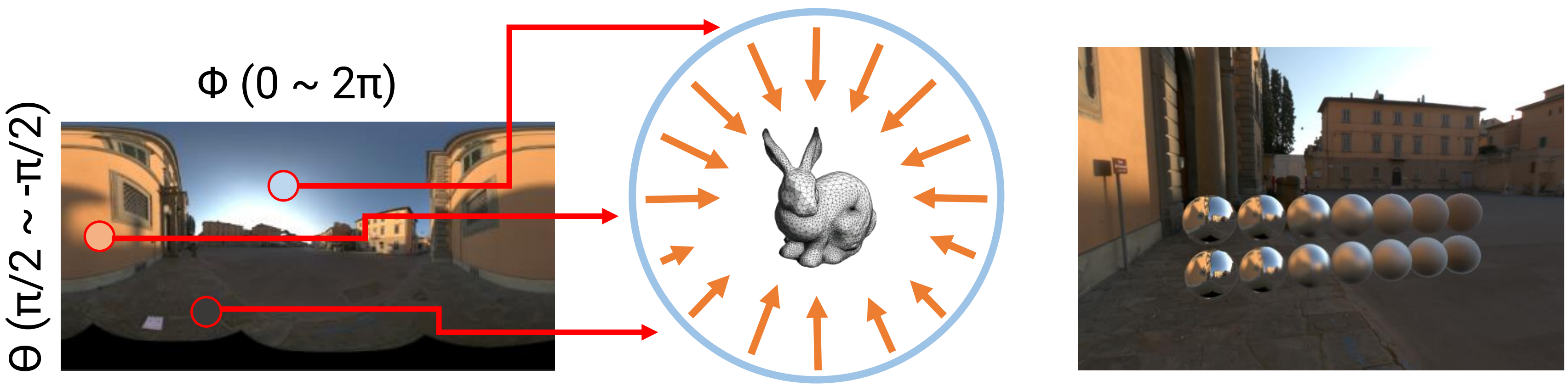
Directional Light

- Describes an emitter that deposits illumination from the **same direction** at every point in space
- Described by
  - Light direction (***D***, xyz)
  - Light radiance (***L***, rgb)



Environment Light

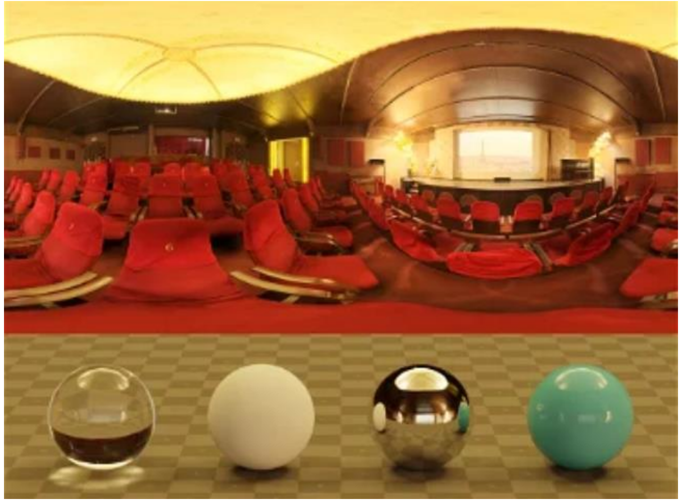
- Use a **texture** (cube map or longitude-latitude image) to represent a **spherical energy distribution**
  - Each texel maps to a spherical direction, considered as a directional light
  - The whole map illuminates the scene from a virtual sphere at an infinite distance
- Also called **image-based lighting (IBL)**

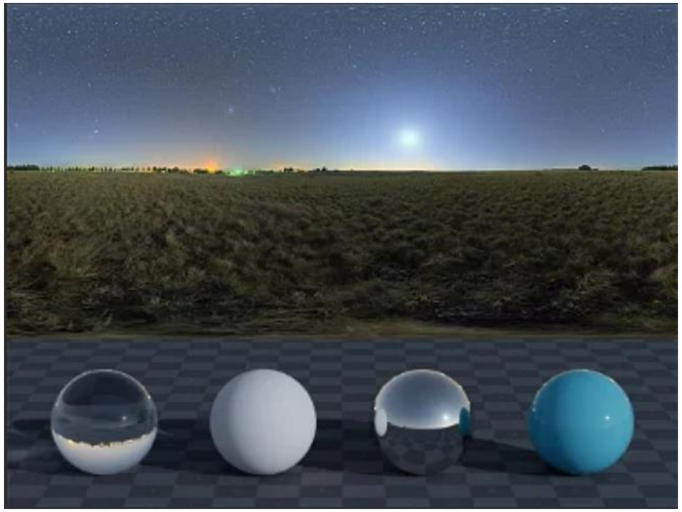


Environment Light (cont.)

- Widely used in digital visual effects and film production

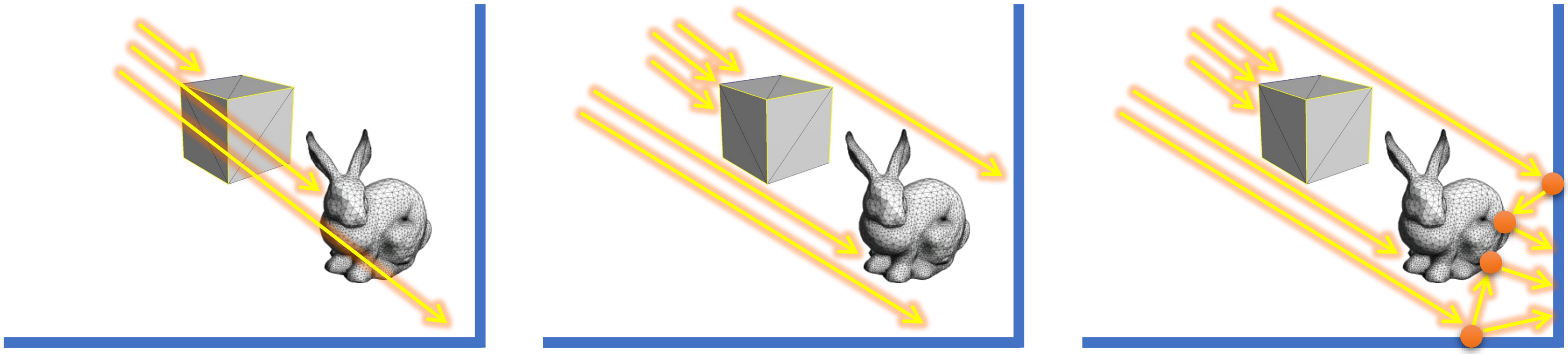
Environment Light (cont.)





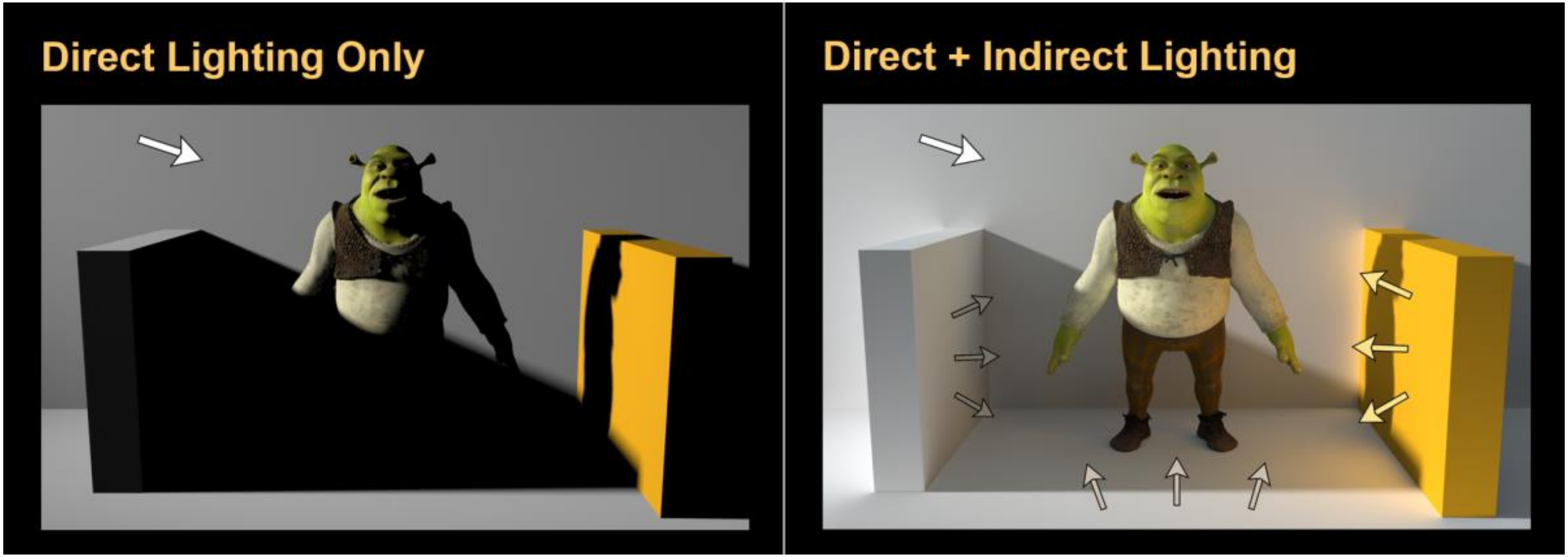

Local, Direct, and Global Illumination

- Direct illumination considers only the **direct** contribution of lights
- Local illumination can be considered as direct lighting **without occlusion** (all lights are fully visible, no shadows)
- Global illumination includes **multi-bounce** illumination reflected from other surfaces (need **recursive** computation!)



local illumination | direct illumination | global illumination

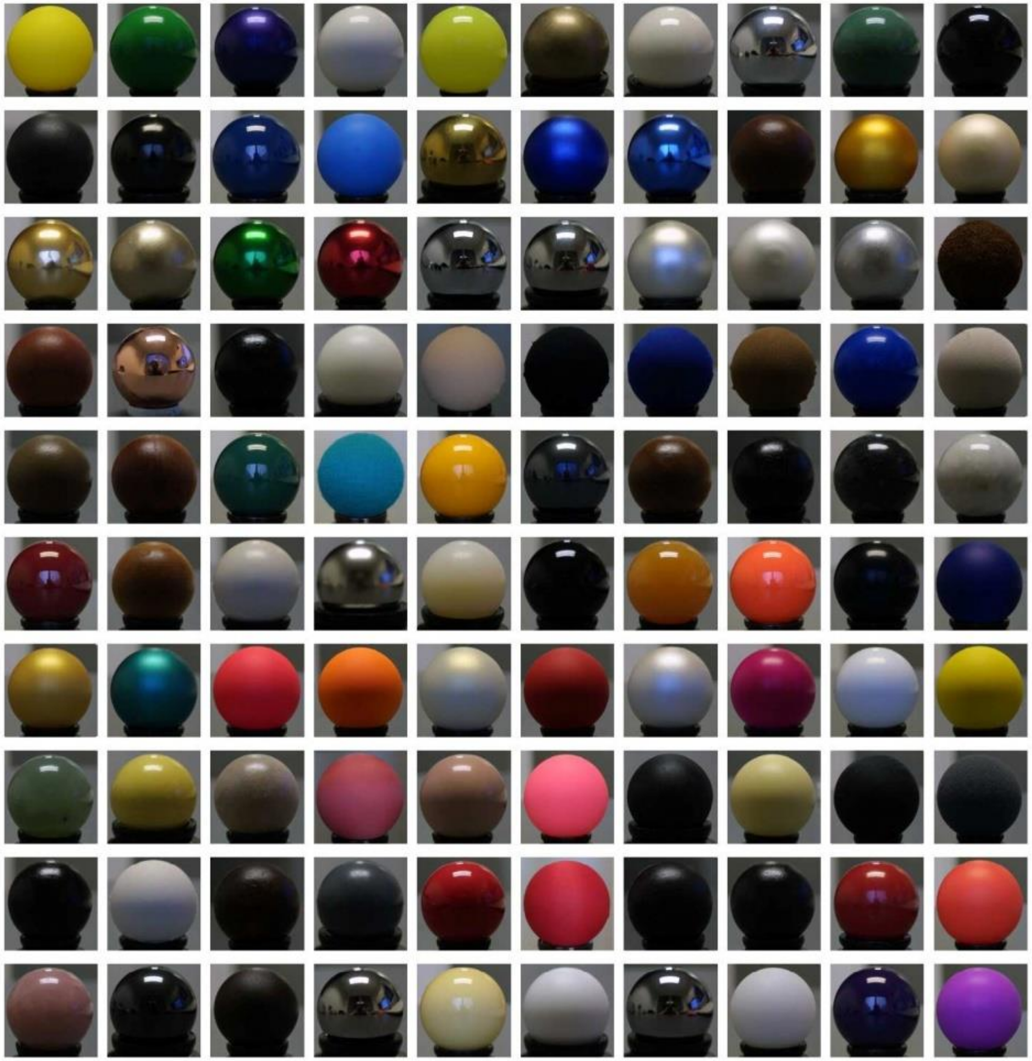
Local, Direct, and Global Illumination (cont.)



Comparison of direct and global illumination

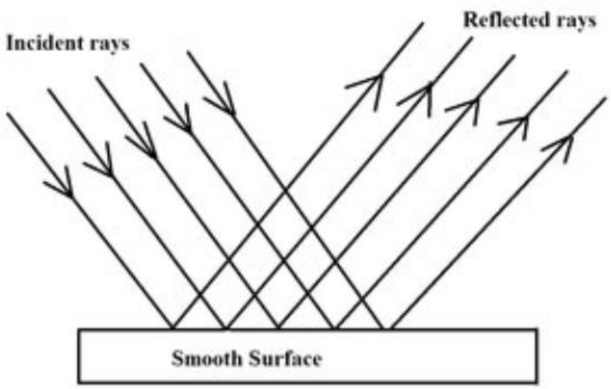
Materials

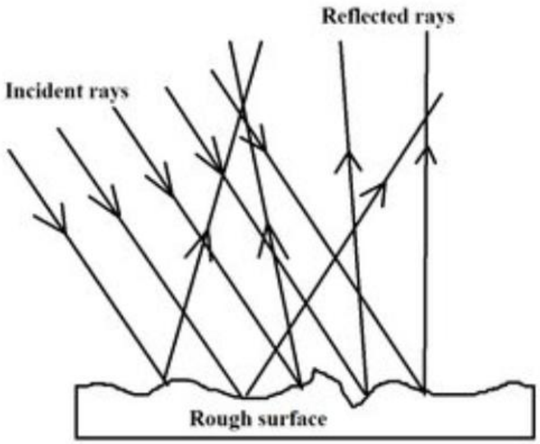
Materials



Materials (cont.)

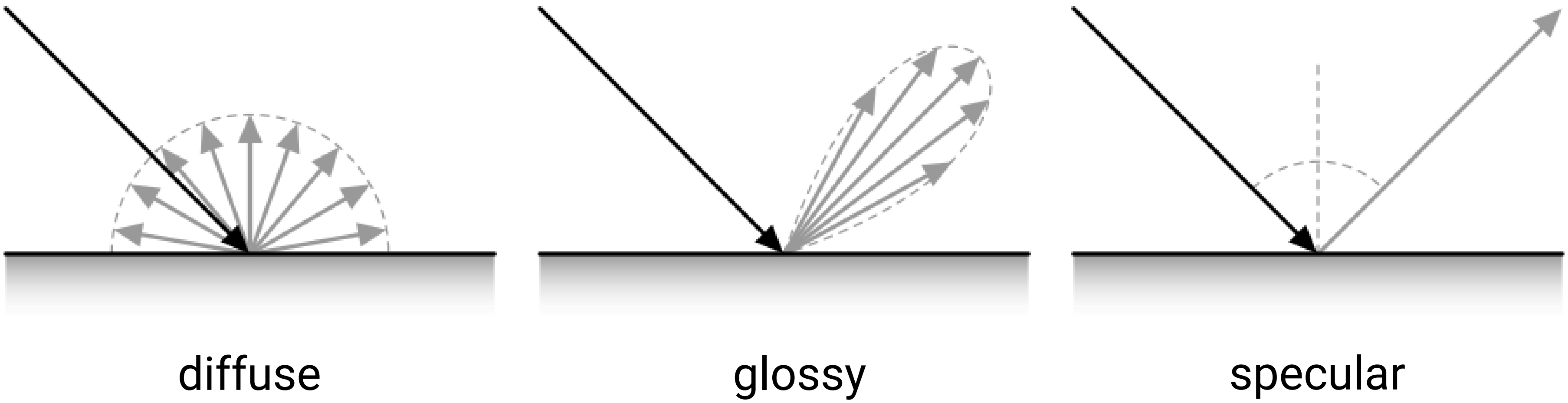
- Highly related to surface types
- The **smoother** a surface, the more reflected light is concentrated in the direction a **perfect mirror** would reflect the light





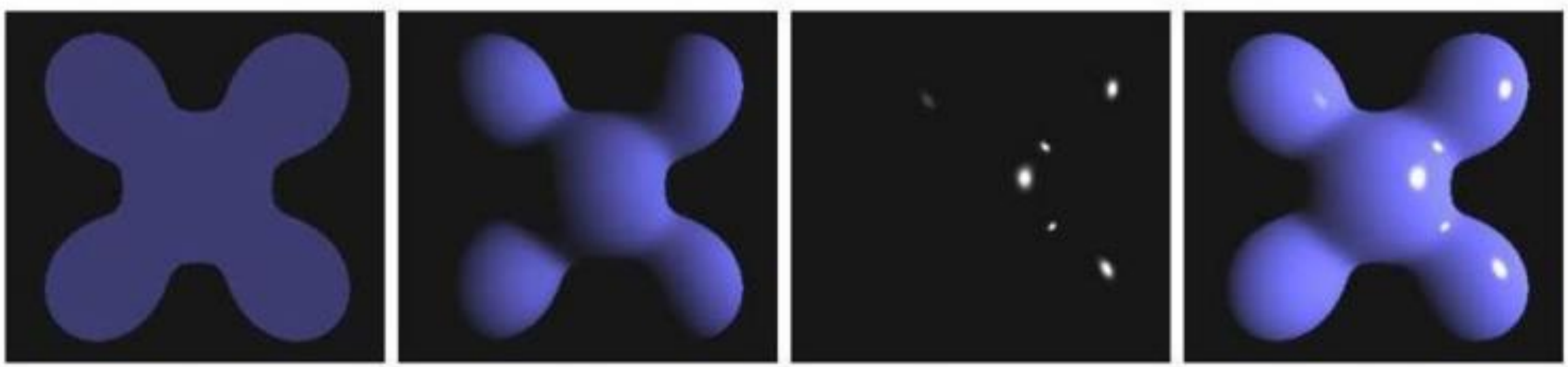
Materials (cont.)

- Highly related to surface types
- The **smoother** a surface, the more reflected light is concentrated in the direction a **perfect mirror** would reflect the light



Phong Lighting Model

- **Diffuse reflection**
  - Light goes everywhere; colored by object color
- **Specular reflection**
  - Happens only near mirror configuration; usually white
- **Ambient reflection**
  - Constant accounted for global illumination (cheap hack)



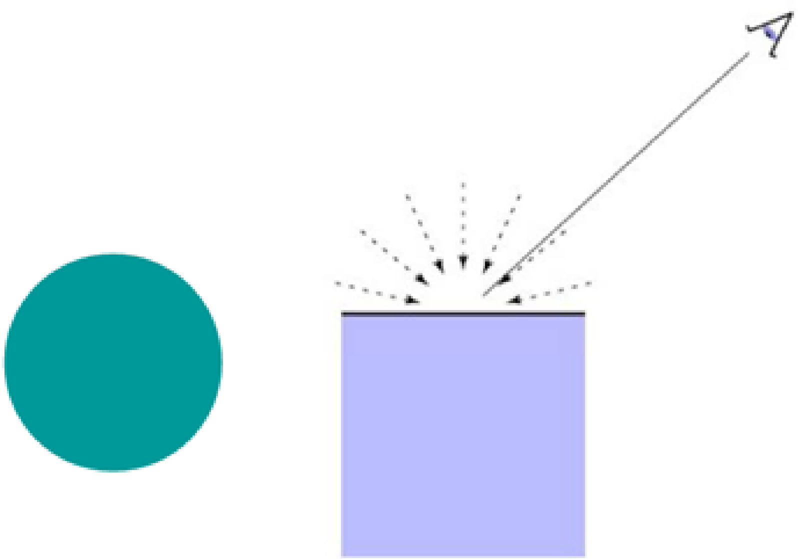
ambient diffuse specular

Ambient Shading

- Add constant color to account for disregarded illumination and fill black shadows

Ambient Shading (cont.)

- Add constant color to account for disregarded illumination and fill black shadows



the **intensity** of ambient light

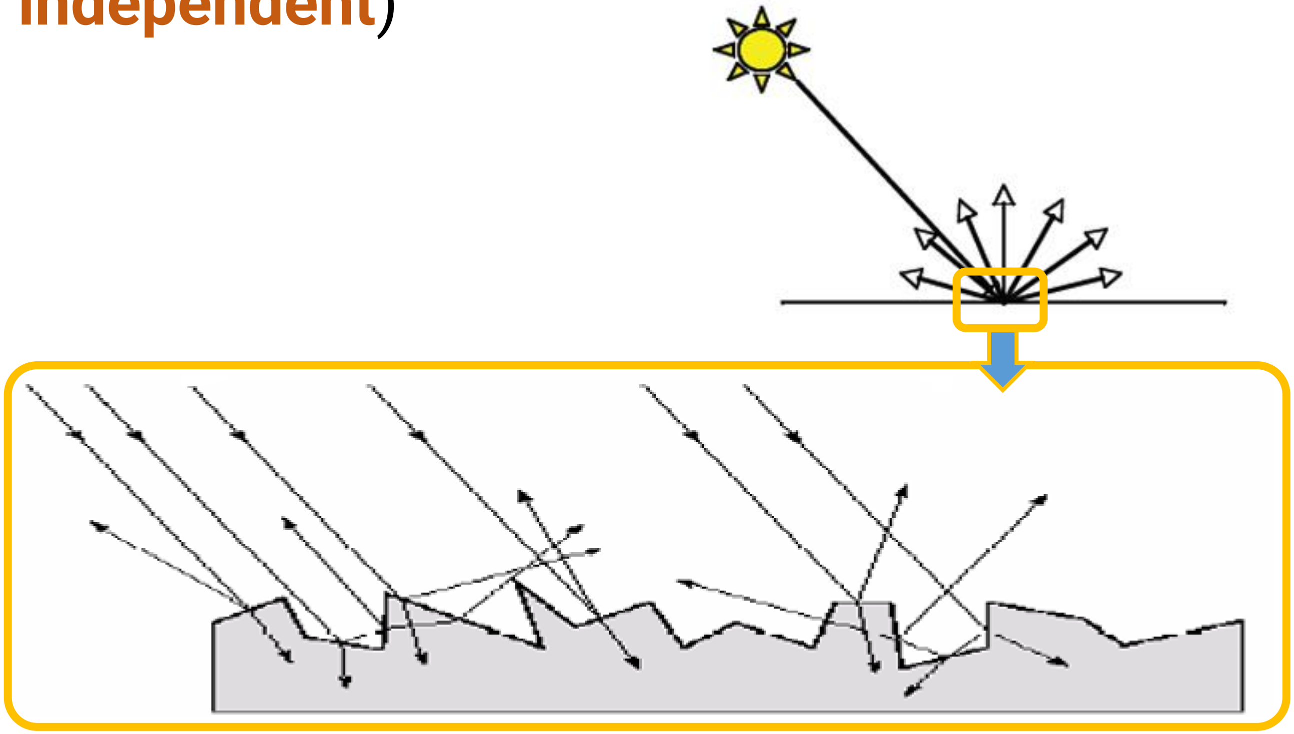
$$L_a = k_a \cdot I_a$$

ambient coefficient

reflected ambient light

Diffuse Shading

- Assume light reflects **equally in all directions**
  - The surface is rough with lots of tiny microfacets
- Therefore, the surface looks the same color from all views (**view independent**)

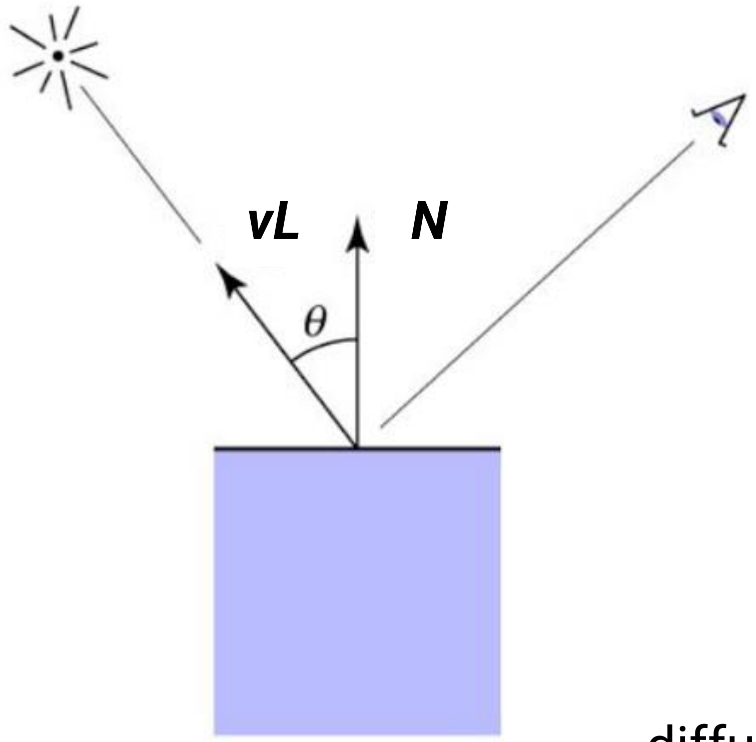


Diffuse Shading (cont.)

- Assume light reflects **equally in all directions**
  - The surface is rough with lots of tiny microfacets
- Therefore, the surface looks the same color from all views (**view independent**)

Diffuse Shading (cont.)

- Applies to diffuse or matte surface



illumination from source

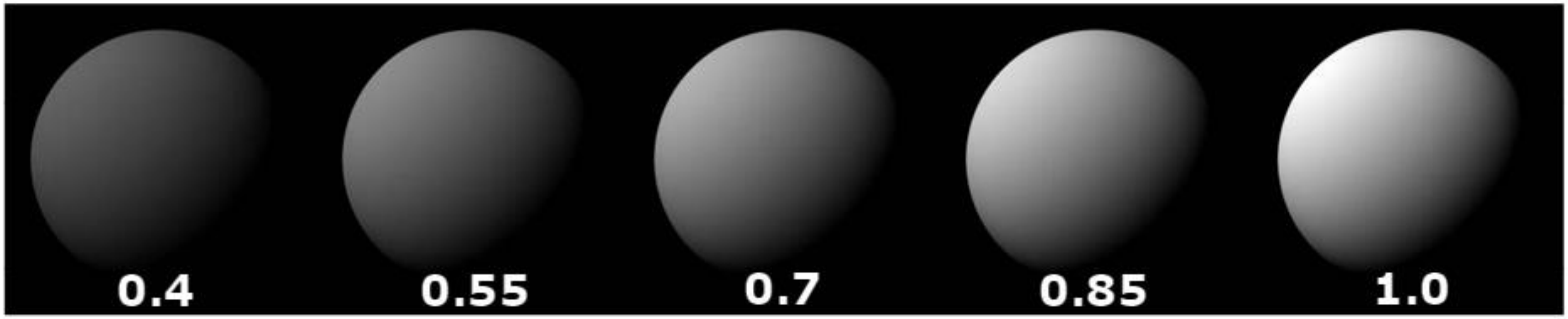
$$L_d = k_d \cdot I \cdot \max(0, N \cdot vL)$$

Lambertian law

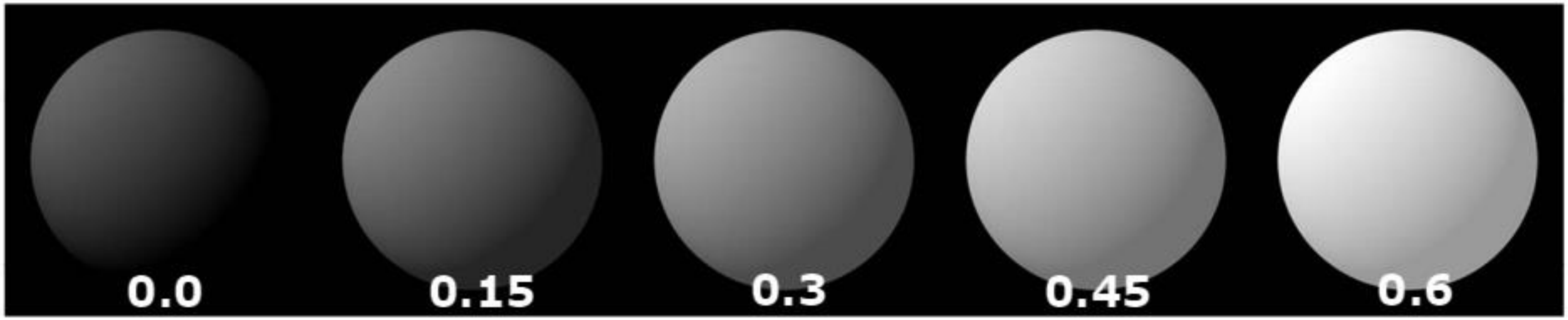
diffuse coefficient

diffusely reflected light

Diffuse Shading (cont.)



diffuse-reflection model with different k_d



ambient and diffuse-reflection model with different k_a

$I_a = 1.0 \qquad k_d = 0.4$

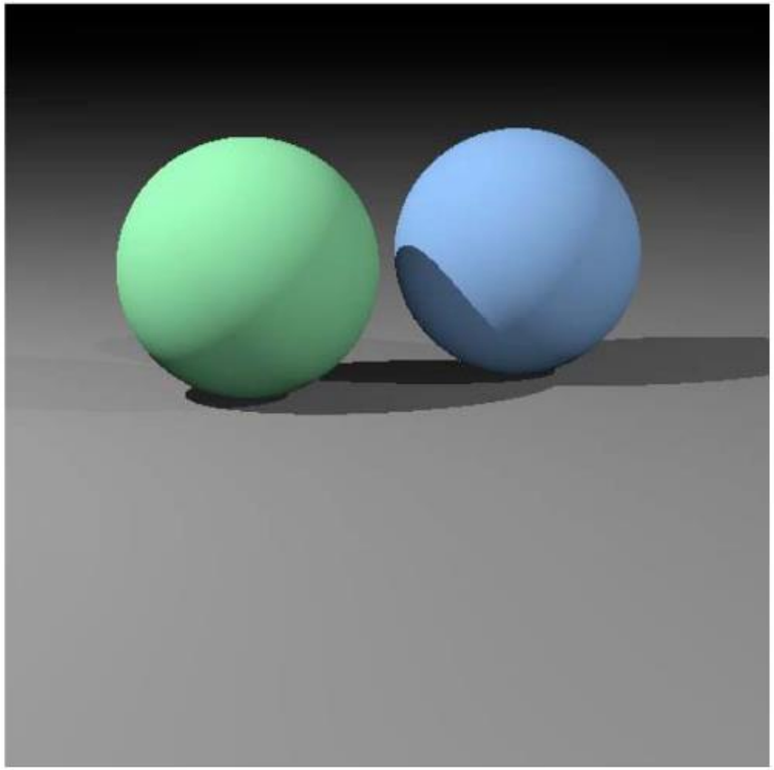
Diffuse Shading (cont.)

- For color objects, apply the formula for each color channel separately
- Light can also be non-white

Example:
white light: (0.9, 0.9, 0.9)
yellow light: (0.8, 0.8, 0.2)

$$L_d = \boxed{k_d} \cdot \boxed{I} \cdot \max(0, N \cdot vL)$$

Example:
green ball: (0.2, 0.7, 0.2)
blue ball: (0.2, 0.2, 0.7)

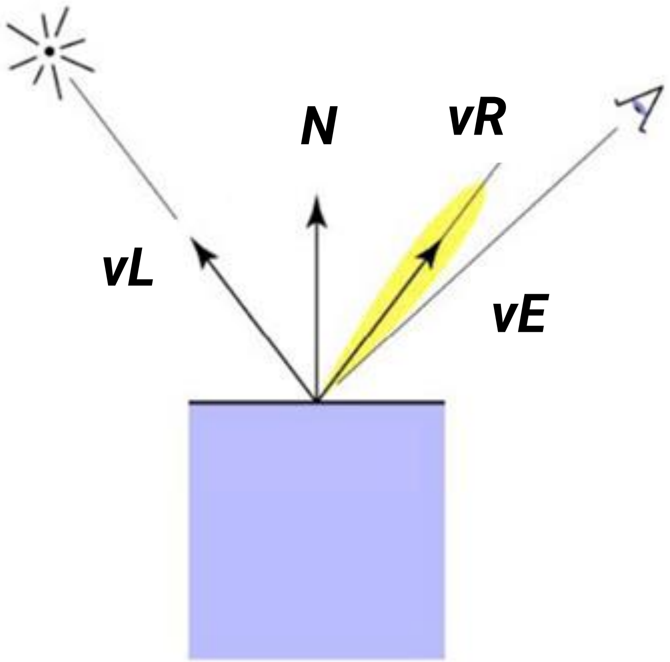


Specular Shading

- Some surfaces have highlights, mirror-like reflection
- **View direction dependent**
- Especially obvious for smooth shiny surfaces

Specular Shading (cont.)

- Phong specular model [1975]



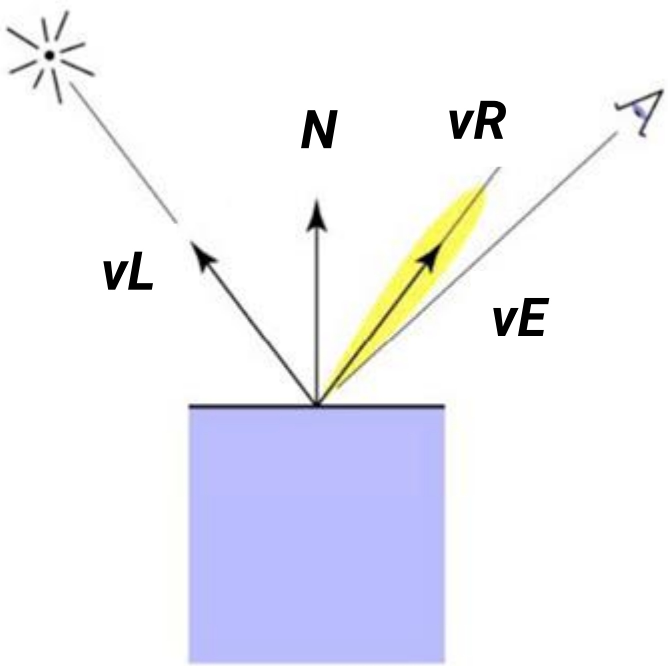
$$
\begin{aligned}
vR &= vL + 2((N \cdot vL)N - vL) \\
&= 2(N \cdot vL)N - vL
\end{aligned}
$$

perfectly reflected direction

(you can find the proof here)

Specular Shading (cont.)

- **Phong specular model [1975]**
  - Fall off gradually from the perfect reflection direction



$$vR = vL + 2((N \cdot vL)N - vL)$$
$$= 2(N \cdot vL)N - vL$$

specular exponent

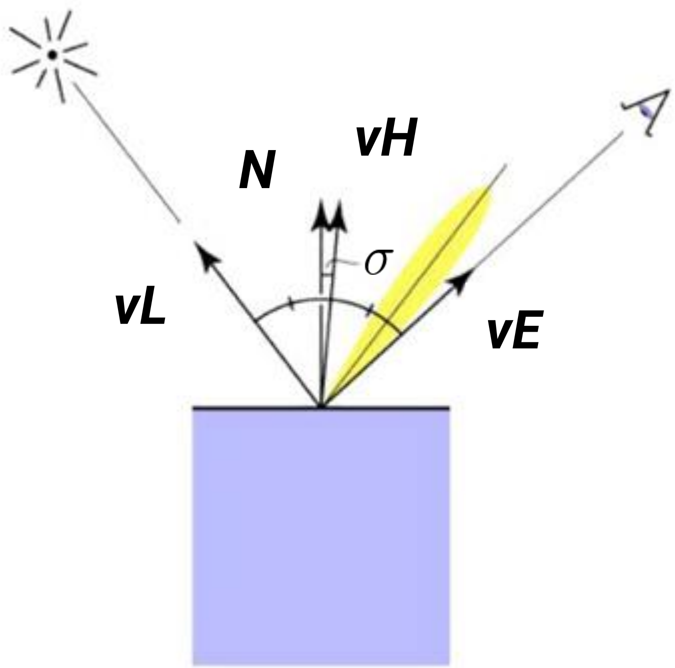
$$L_s = k_s \cdot I \cdot \max(0, \cos\sigma)^{n}$$
$$= k_s \cdot I \cdot \max(0, vE \cdot vR)^n$$

specular coefficient

specularly reflected light

Phong specular Variant: Blinn-Phong

- Rather than computing reflection directly, just compare to normal bisection property
- One can prove $\cos^n(\sigma) = \cos^{4n}(\alpha)$



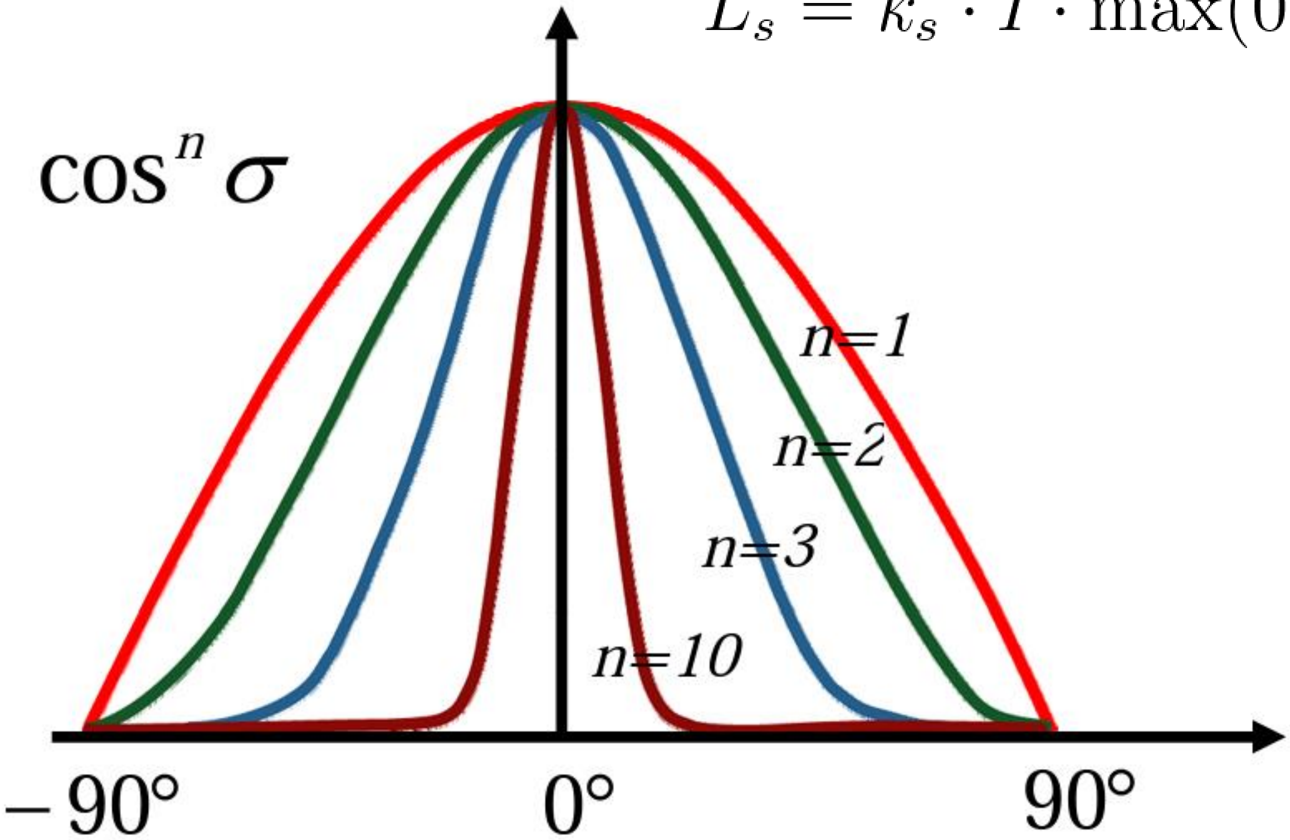
$$vH = \text{bisector}(vL, vE) = \frac{(vL + vE)}{\|vL + vE\|}$$

$$L_s = k_s \cdot I \cdot \max(0, \cos\sigma)^n = k_s \cdot I \cdot \max(0, N \cdot vH)^n$$

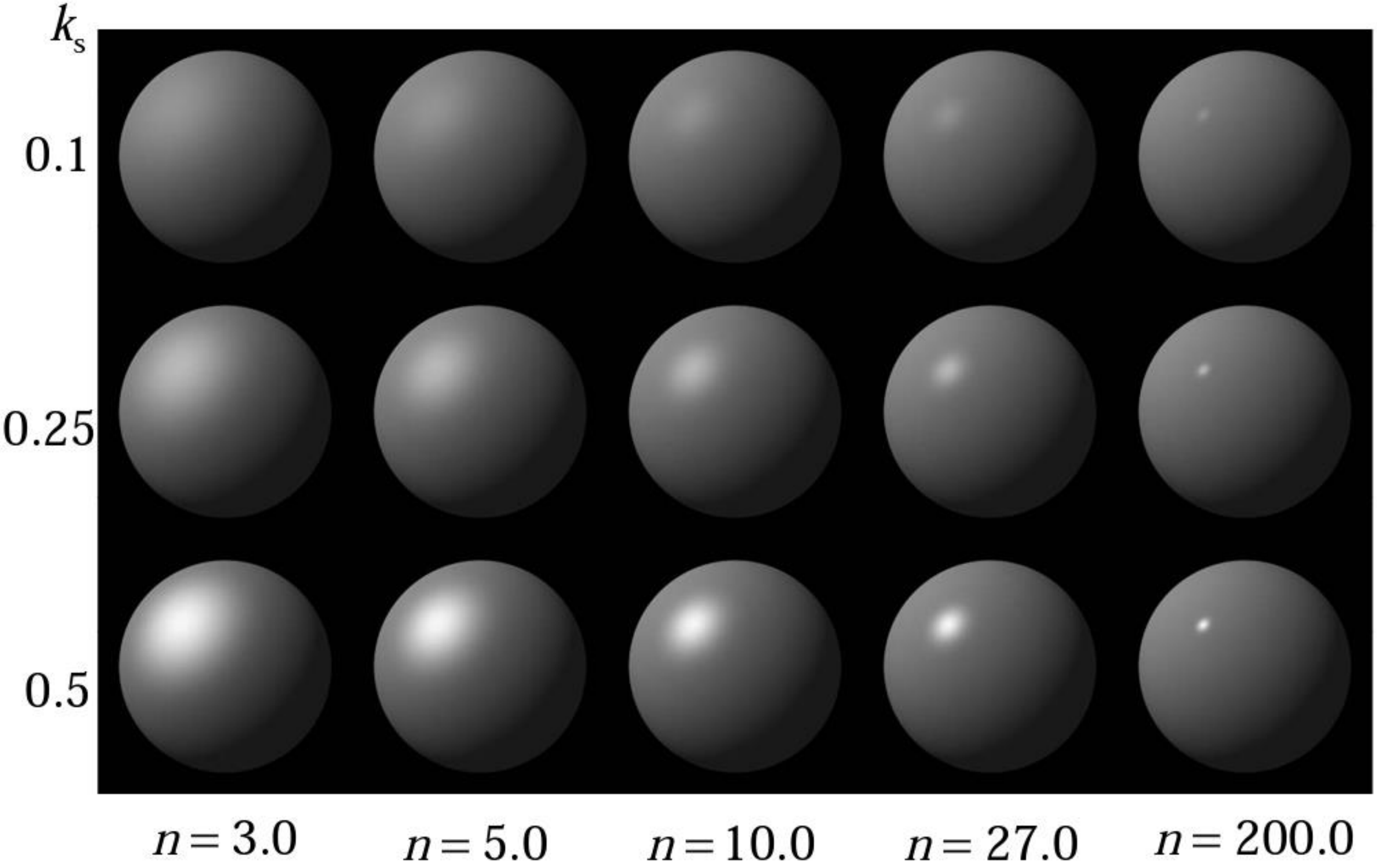
Specular Shading (cont.)

- Increase *n* narrows the lobe

$$L_s = k_s \cdot I \cdot \max(0, \cos\sigma)^n$$



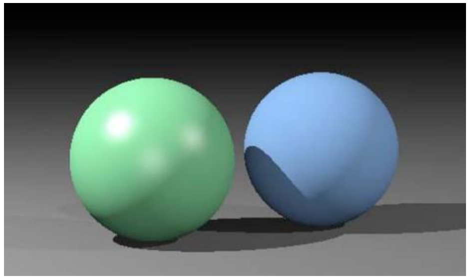
Specular Shading (cont.)



Complete Phong Lighting Model

- Compute the contribution from a light to a point by including **ambient**, **diffuse**, and **specular** components

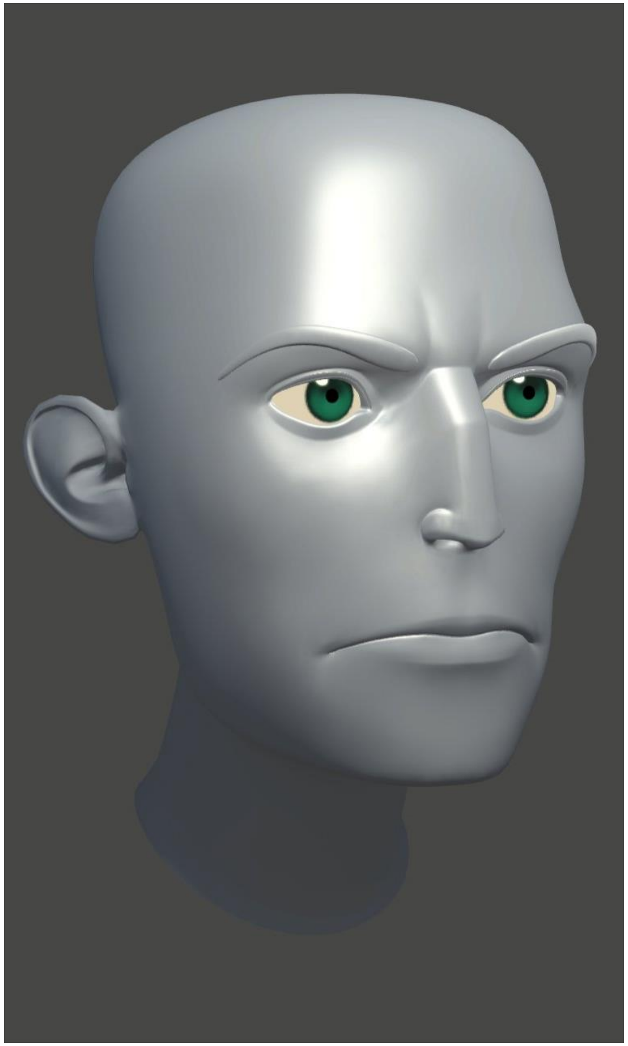
$$
\begin{aligned}
L &= L_a + L_d + L_s \\
&= k_a \cdot I_a + I(k_d \cdot \max(0, N \cdot vL) + k_s \cdot \max(0, N \cdot vH)^n)
\end{aligned}
$$



- If there are ***s*** lights, just sum over all the lights because the lighting is **linear**

$$L = k_a \cdot I_a + \sum_i^s (I_i(k_d \cdot \max(0, N \cdot vL_i) + k_s \cdot \max(0, N \cdot vH_i)^n))$$

Some Results with Phong Lighting Model



Material File Format

Material Template Library

- A material template library (*.mtl) file defines the materials of a *.obj model

cube.obj - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明

```
# Unit-volume cube with the same texture coordinates on each face.
#
# Created by Morgan McGuire and released into the Public Domain on
# July 16, 2011.
#
# http://graphics.cs.williams.edu/data

mtllib default.mtl

v -0.5 0.5 -0.5
v -0.5 0.5 0.5
v 0.5 0.5 0.5
v 0.5 0.5 -0.5
v -0.5 -0.5 -0.5
v -0.5 -0.5 0.5
v 0.5 -0.5 0.5
v 0.5 -0.5 -0.5

vt 0 1
vt 0 0
vt 1 0
vt 1 1

vn 0 1 0
vn -1 0 0
vn 1 0 0
vn 0 0 -1
vn 0 0 1
vn 0 -1 0
```

specify material file

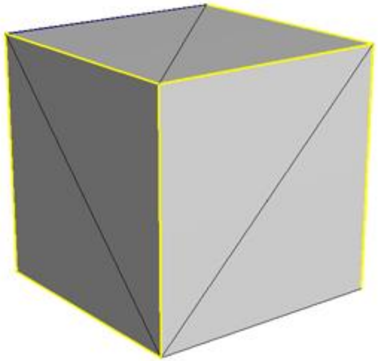
```
g cube
usemtl default

f -8/-4/-6 -7/-3/-6 -6/-2/-6
f -8/-4/-6 -6/-2/-6 -5/-1/-6
f -8/-4/-5 -4/-3/-5 -3/-2/-5
f -8/-4/-5 -3/-2/-5 -7/-1/-5
f -6/-4/-4 -2/-3/-4 -1/-2/-4
f -6/-4/-4 -1/-2/-4 -5/-1/-4
f -5/-4/-3 -1/-3/-3 -4/-2/-3
f -5/-4/-3 -4/-2/-3 -8/-1/-3
f -7/-4/-2 -3/-3/-2 -2/-2/-2
f -7/-4/-2 -2/-2/-2 -6/-1/-2
f -3/-4/-1 -4/-3/-1 -1/-2/-1
f -3/-4/-1 -1/-2/-1 -2/-1/-1
```

第 20 列，第 7 行 160% Unix (LF) UTF-8

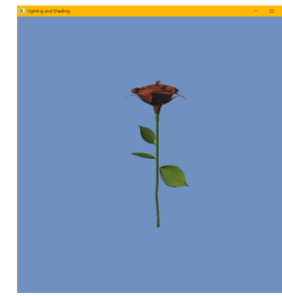
declare a new group (submesh) called “cube” that use “default” material

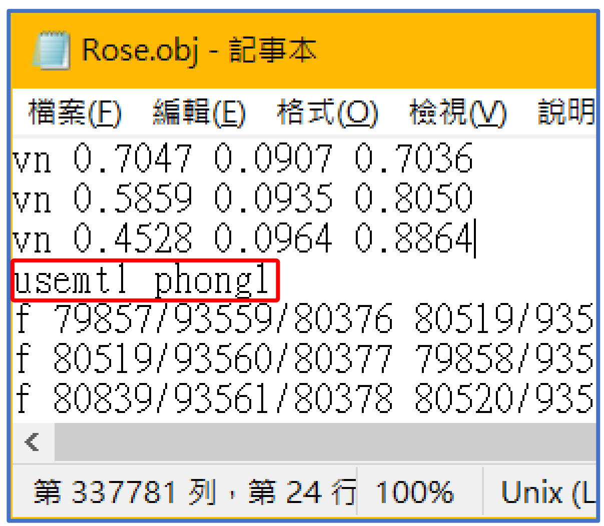
these faces are in the “cube” group and use the “default” material

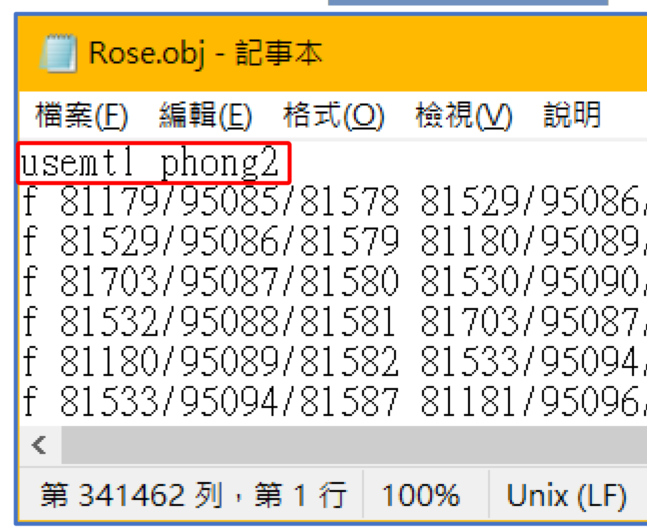


Material Template Library (cont.)

- A model can have multiple groups (sub-meshes)
- The faces in the same group have the same material properties







Material Template Library (cont.)

- The material template library (*.mtl) used by a Wavefront OBJ (*.obj) file describes material properties using
  - Phong lighting model (Ka, Kd, Ks, Ns)
  - Texture maps (mapKa, mapKd, mapKs, mapNs …)
  - Transparency (d, Tr, Ni)
  - … etc

- You can refer to the wiki page for more information https://en.wikipedia.org/wiki/Wavefront_.obj_file

Material Template Library (cont.)

Rose.obj

Rose.obj - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明

```
vn 0.7047 0.0907 0.7036
vn 0.5859 0.0935 0.8050
vn 0.4528 0.0964 0.8864
usemtl phong1
f 79857/93559/80376 80519/935
f 80519/93560/80377 79858/935
f 80839/93561/80378 80520/935
```

第 337781 列，第 24 行 100% Unix (L

Rose.obj - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明

```
usemtl phong2
f 81179/95085/81578 81529/95086/
f 81529/95086/81579 81180/95089/
f 81703/95087/81580 81530/95090/
f 81532/95088/81581 81703/95087/
f 81180/95089/81582 81533/95094/
f 81533/95094/81587 81181/95096/
```

第 341462 列，第 1 行 100% Unix (LF)

Rose.obj - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明

```
vn 0.0164 -0.9999 0.0000
usemtl phongE1
f 1/1/1 29/2/2 32/3/3 2/4/4
f 2/4/4 32/3/3 33/5/5 3/6/6
f 3/6/6 33/5/5 34/7/7 4/8/8
f 4/8/8 34/7/7 3344/9/9 3345/
f 29/2/2 30/11/11 35/12/12 32
```

第 253798 列，第 34 行 100% Unix (L

Rose.mtl

Rose.mtl - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明

```
# Blender MTL File: 'None'
# Material Count: 3

newmtl phong1
Ns 179.999996
Ka 0.500000 0.500000 0.500000
Kd 0.420000 0.620000 0.058000
Ks 0.500000 0.500000 0.500000

newmtl phong2
Ns 18.000005
Ka 0.149351 0.149351 0.149351
Kd 0.478000 0.651000 0.318000
Ks 0.500000 0.500000 0.500000

newmtl phongE1
Ns 179.999996
Ka 0.500000 0.500000 0.500000
Kd 0.700000 0.240000 0.240000
Ks 0.300000 0.300000 0.300000
```

第 1 列，第 1 行 100% Unix (LF)

Any Questions?