Lighting and Shading
Today: Local Illumination

Solving the rendering equation is too expensive

First do local illumination

Then “hack in” reflections and shadows
Local Shading: Notation

$I^{in}, I$  light intensity in, light intensity out

$l, \hat{n}, \hat{v}, \hat{r}$  vector pointing to: light, normal direction, eye, reflection direction
Ignore camera and light direction completely

\[ I_a = k_a I^{in} \]

material constant
Ambient Term

Ignore camera and light direction completely

\[ I_a = k_a I^{in} \]

one eq. per color channel

material constant
Diffuse Term

Lambertian surface – constant BRDF

\[ I_d = k_d \max(\hat{l} \cdot \hat{n}, 0) I^{in} \]
Diffuse Term

Lambertian surface – constant BRDF

\[ I_d = k_d \max(\hat{l} \cdot \hat{n}, 0) I^{in} \]

ignore back faces
Specular Term

Perfect specular surface doesn’t work
Perfect specular surface doesn’t work

**Phong model:**

\[ I_s = k_s \max(\hat{\nu} \cdot \hat{r}, 0)^\alpha I^{in} \]

specularity coefficient
Specular Term

Perfect specular surface doesn’t work

**Phong** model:

\[ I_s = k_s \max(\hat{u} \cdot \hat{r}, 0)^\alpha I_{in} \]

Looks like “highlight” that moves with light & eye
Putting It Together

\[ I = \left[ k_a + k_d \max(\hat{l} \cdot \hat{n}) + k_s \max(\hat{v} \cdot \hat{r}, 0)^\alpha \right] I_{in} \]
Putting It Together

\[
I = \left[ k_a + k_d \max(\hat{l} \cdot \hat{n}) + k_s \max(\hat{v} \cdot \hat{r}, 0)^\alpha \right] I^{in}
\]

typically: \( k_a = 0; k_s = 1 \)

![Ambient](image1) + ![Diffuse](image2) + ![Specular](image3) = ![Phong Reflection](image4)
Putting It Together

\[ I = \left[ k_a + k_d \max(\hat{l} \cdot \hat{n}) + k_s \max(\hat{v} \cdot \hat{r}, 0)^\alpha \right] I^{in} \]

typically: \( k_a = 0; k_s = 1 \)

three copies of equation, one per color channel

Ambient + Diffuse + Specular = Phong Reflection
Specularity Coefficient

more specular

higher exponent
Light Attenuation

Real light attenuation: inverse square law
Light Attenuation

Real light attenuation: inverse square law

Tends to look bad: too dim or washed out

So, we cheat
Light Attenuation

\[ I_{\text{atten}} = \frac{1}{c_0 + c_1 d + c_2 d^2} I \]

\(d\) is light-to-point distance

Can tweak constant & linear term to taste
Directional Light

“Light at infinity”
• Sun

All light rays parallel

Obviously, no attenuation
Dealing with Discrete Geometry

Flat shading: use normal per face

Very obvious discontinuities at edges

Only used for stylized “chunky” effect
Gouraud Interpolation

First, compute color at vertices

Linearly interpolate color over face
Gouraud Interpolation

First, compute color at vertices

Linearly interpolate color over face

Color now continuous, but still obvious artifacts (nobody uses this anymore)
Phong Interpolation

First, linearly interpolate normals
Next, renormalize normals
(important)
Then, compute color per pixel
Phong Interpolation

First, linearly interpolate normals
Next, renormalize normals (important)
Then, compute color per pixel

Because of all of the normalizations, used to be considered decadent; now the standard local shading scheme
Local vs Global Illumination

Local:
• shade each object based only on itself, the eye, and the light sources

Global:
• take all other objects in scene into account also
• BRDFs and the rendering equation
Local vs Global Illumination

Grey Area:

• take other objects into account, without full global illumination
• adds realism without being too slow
Local vs Global Illumination

Grey Area:

- take other objects into account, without full global illumination
- adds realism without being too slow
- common techniques exist for
  - shadows
  - reflections
  - refractions
Shooting Shadow Rays
Shooting Shadow Rays

For each intersection pt:
- for each light:
  - shoot ray from point to light
  - if it hits an object:
    - do nothing (shadow)
  - else add shading contribution
Classic Bug
at what $t$ does the ray hit an object?
Classic Bug

at what $t$ does the ray hit an object?
at what $t$ does the ray hit an object?

if lucky: $\{-1.2, 0.0\}$
Classic Bug

at what $t$ does the ray hit an object?

if lucky: \{-1.2, 0.0\}
if unlucky: \{-1.2, 1e-12\}
Ignore t Near Zero?
Ignore t Near Zero?
Ignore t Near Zero?

Fix: move slightly in normal direction (or backward ray direction) before shooting shadow ray
Hard Shadows

real-world doesn’t look like this
Hard Shadows

real-world doesn’t look like this

shadows usually soft

why?
Soft Shadows

- Light source
- Object
- Shadow
- Penumbra
- Umbra
- Object

Diagram showing the concept of soft shadows with different regions of light and shadow.
Other Secondary Rays

Translucent objects
Other Secondary Rays

Reflection & refraction
Reflection

Purely specular (mirrored) surface
Reflection

Purely specular (mirrored) surface:
1. Incoming ray $w_{\text{in}}$ hits surface

\[ \text{Diagram:} \quad \hat{n} \]

$w_{\text{in}}$
Reflection

Purely specular (mirrored) surface:
1. Incoming ray $w_{\text{in}}$ hits surface
2. Shoot secondary reflection ray $w_{\text{ref}}$
Reflection

Purely specular (mirrored) surface:
1. Incoming ray $w_{\text{in}}$ hits surface
2. Shoot secondary reflection ray $w_{\text{ref}}$
3. Set pixel color to color “seen” by $w_{\text{ref}}$
Choosing the Reflection Ray

Angle of reflection = angle of incidence
Choosing the Reflection Ray

Angle of reflection = angle of incidence
i.e. negate component of $w_{in}$ in normal dir
• leave **tangent** component untouched
Choosing the Reflection Ray

Angle of reflection = angle of incidence
l.e. negate component of $w_{\text{in}}$ in normal dir
• leave $\text{tangent}$ component untouched
Choosing the Reflection Ray

Angle of reflection = angle of incidence
i.e. negate component of $w_{in}$ in normal dir
• leave **tangent** component untouched

$$w_{ref} = -w_{normal} + w_{tangent}$$
Choosing the Reflection Ray

The math:

\[ w_{\text{normal}} = (w_{\text{in}} \cdot \hat{n})\hat{n} \]

\[ w_{\text{ref}} = w_{\text{in}} - 2(w_{\text{in}} \cdot \hat{n})\hat{n} = (I - 2\hat{n}\hat{n}^T)w_{\text{in}} \]

\[ w_{\text{ref}} = -w_{\text{normal}} + w_{\text{tangent}} \]
Reflection in Practice

Objects may not be perfectly mirrored
• blend reflected color with basic shading
Reflection in Practice

Objects may not be perfectly mirrored
• blend reflected color with basic shading

Objects have **base color**
• multiplies reflected color
Reflections in Practice

Reflection ray might hit reflective objects

• cast recursive reflection rays…

• stop after some maximum recursion limit
Refraction
Refraction

Light **bends** when moving between different materials.

Caused by change in speed of light.

We “see” dotted straw.
Index of Refraction

Measures speed of light in material

\[ \text{index of refraction} = \frac{\text{speed in vacuum}}{\text{speed in material}} \]
Index of Refraction

Measures speed of light in material

\[
\text{index of refraction} = \frac{\text{speed in vacuum}}{\text{speed in material}}
\]

Common values:

- Vacuum: 1.0
- Air: 1.0001
- Water: 1.33
- Glass: 1.5
Snell’s Law

\[ \frac{\sin \theta_1}{\sin \theta_2} = \frac{n_2}{n_1} \]

Special cases:

• \( n_1 = n_2 \)

• \( \theta_1 = 0 \)
Implementing Snell’s Law

Shoot refraction ray

\[ \sin \theta_1 = \frac{||w_{tangent}||}{||w_{in}||} \]

\[ w_{rfr} = \alpha w_{tangent} + w_{normal} \]

Solve for alpha

refractive index

\[ n_1 \]

\[ n_2 \]

interface

\[ \theta_2 \]

\[ \theta_1 \]
Refractions in Practice

Again, usually multiplied by a base color
Light bends when entering and leaving
• must detect both when ray-tracing
Reflection & Refraction Example