Lighting & Shading

From Physical Reality to Beautiful Renders

Ashrafur Rahman

Adjunct Lecturer

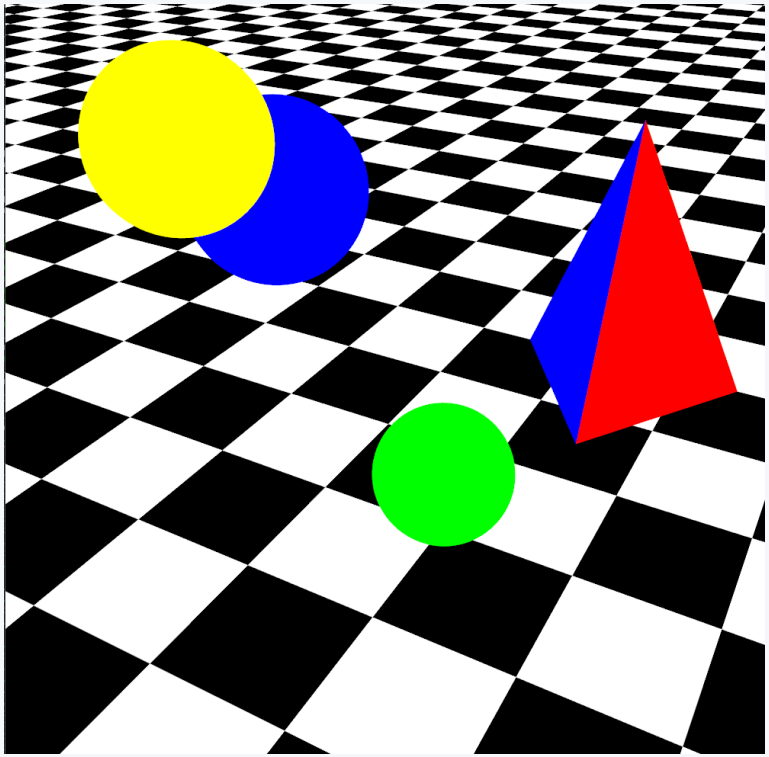
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology (BUET)

Index

Motivation

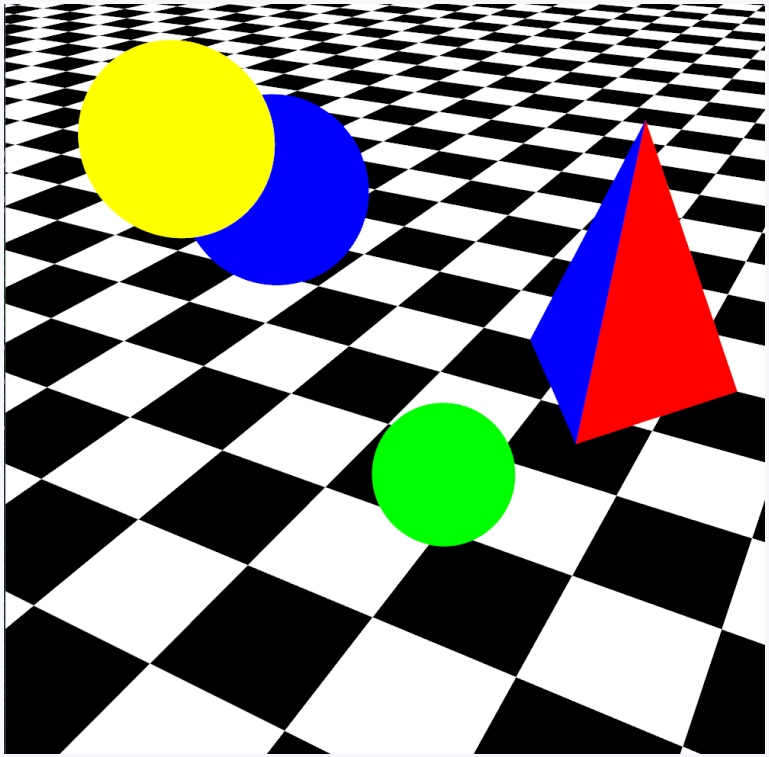
Why Lighting Matters

Why Lighting Matters

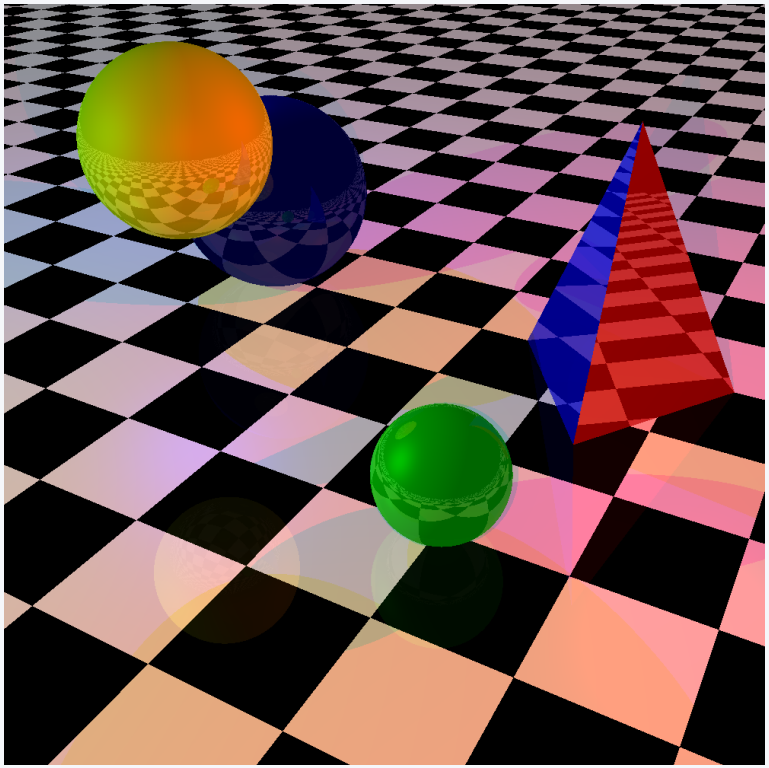


Without Lighting and Shading

Why Lighting Matters



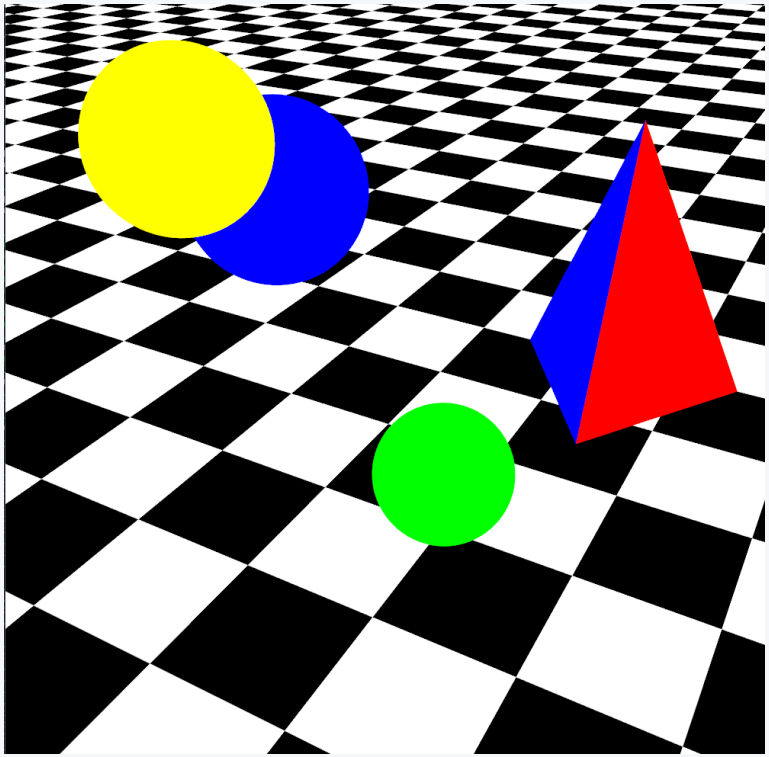
Without Lighting and Shading



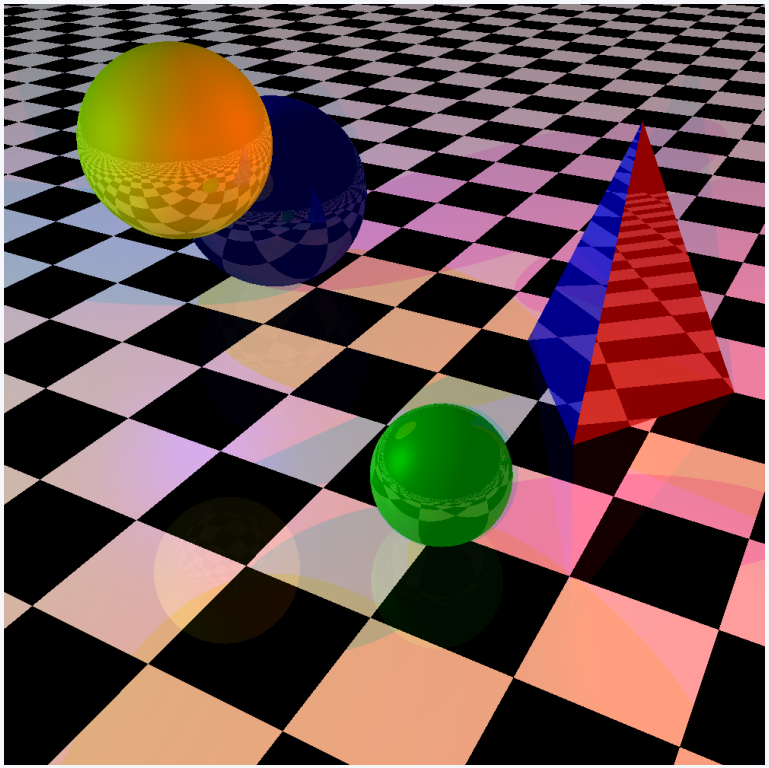
With Lighting and Shading

- **Depth perception** - Lighting reveals 3D shape and form

Why Lighting Matters



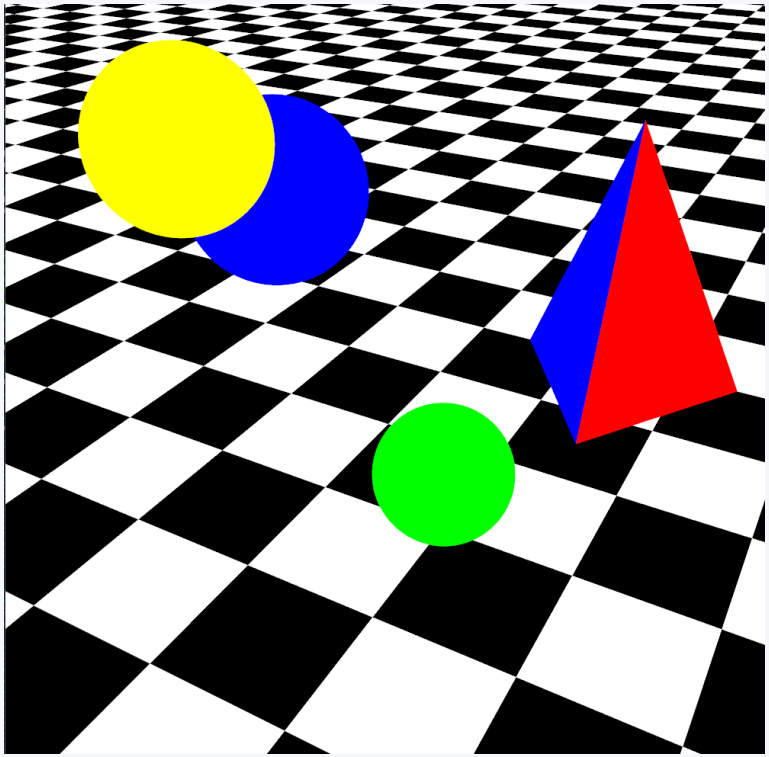
Without Lighting and Shading



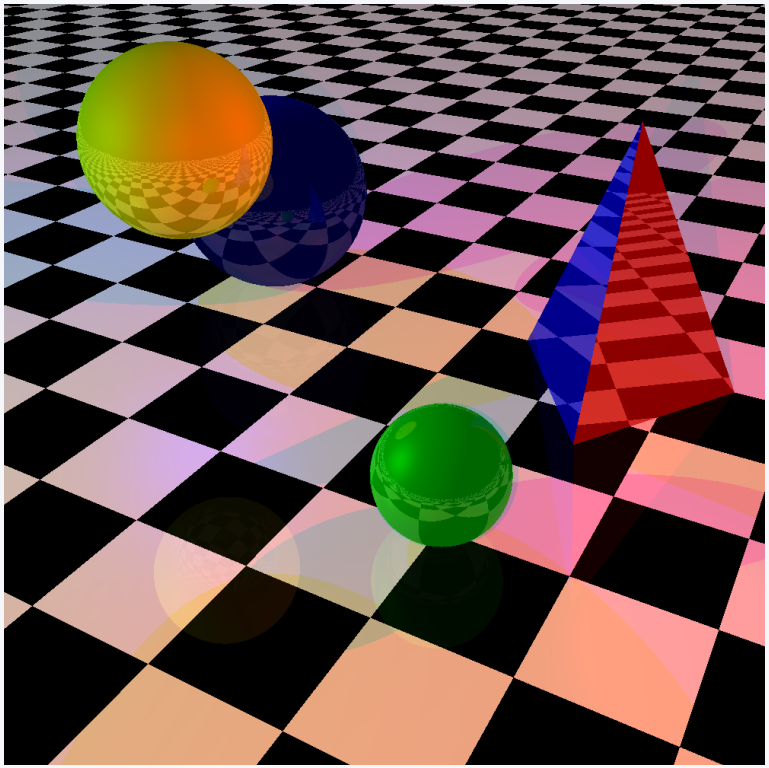
With Lighting and Shading

- **Depth perception** - Lighting reveals 3D shape and form
- **Material properties** - Distinguishes between plastic, metal, wood

Why Lighting Matters



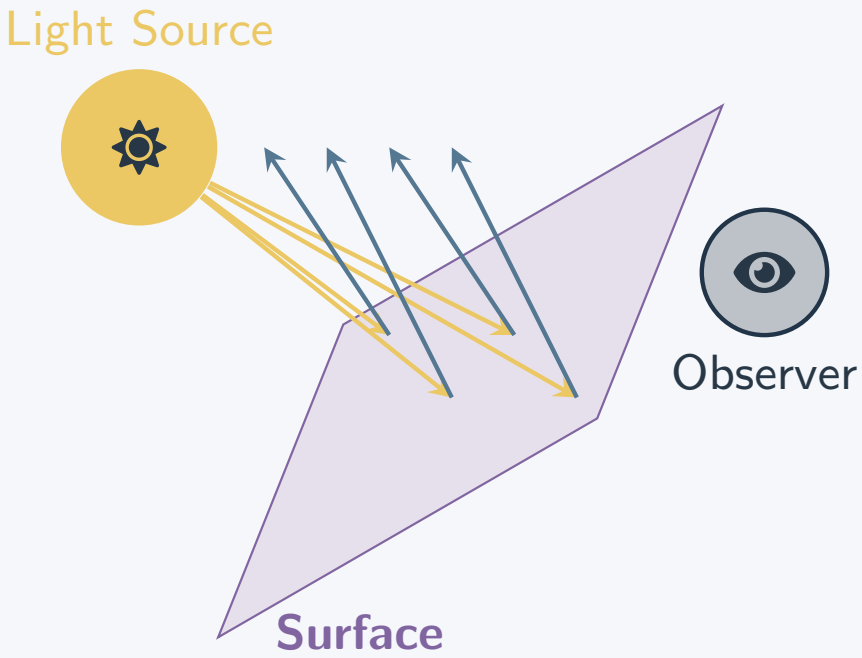
Without Lighting and Shading



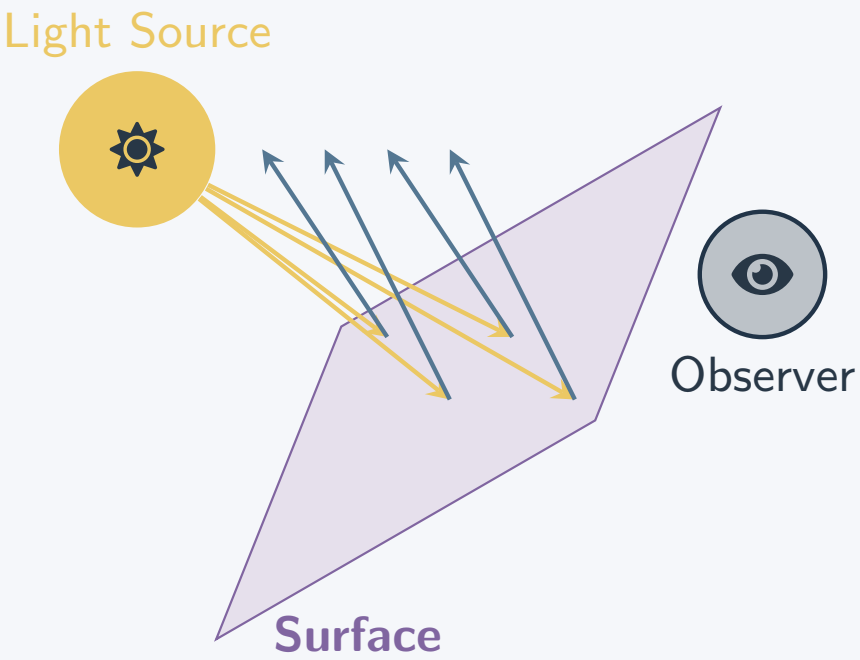
With Lighting and Shading

- **Depth perception** - Lighting reveals 3D shape and form
- **Material properties** - Distinguishes between plastic, metal, wood
- **Realism** - Makes computer graphics believable and immersive

The Challenge: From Reality to Code



The Challenge: From Reality to Code



Reality vs Computation

Physical World:

- Millions of photons per surface point
- Complex wave interactions
- Multiple scattering events
- Continuous spectrum

Computer Graphics:

- Discrete RGB values
- Simplified mathematical models
- Local illumination approximations
- Real-time constraints

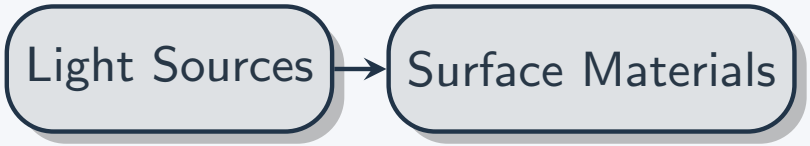
Our Journey

Our Journey

Light Sources

- Point lights
- Directional
- Spotlights
- Attenuation

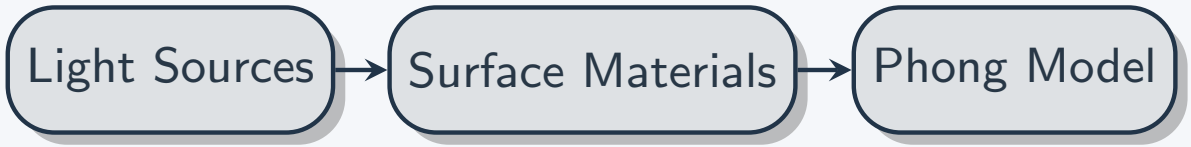
Our Journey



- Point lights
- Directional
- Spotlights
- Attenuation

- Reflection types
- Surface normals
- Materials

Our Journey

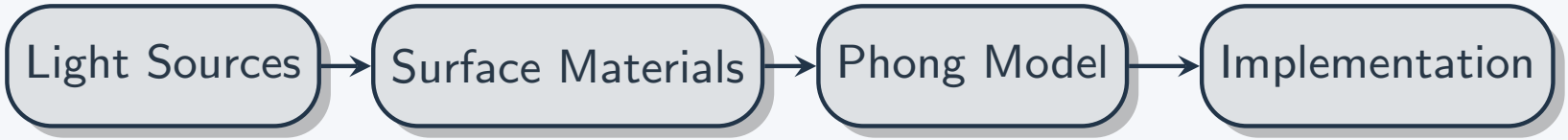


- Point lights
- Directional
- Spotlights
- Attenuation

- Reflection types
- Surface normals
- Materials

- Ambient
- Diffuse
- Specular
- Mathematics

Our Journey



- Point lights
- Directional
- Spotlights
- Attenuation

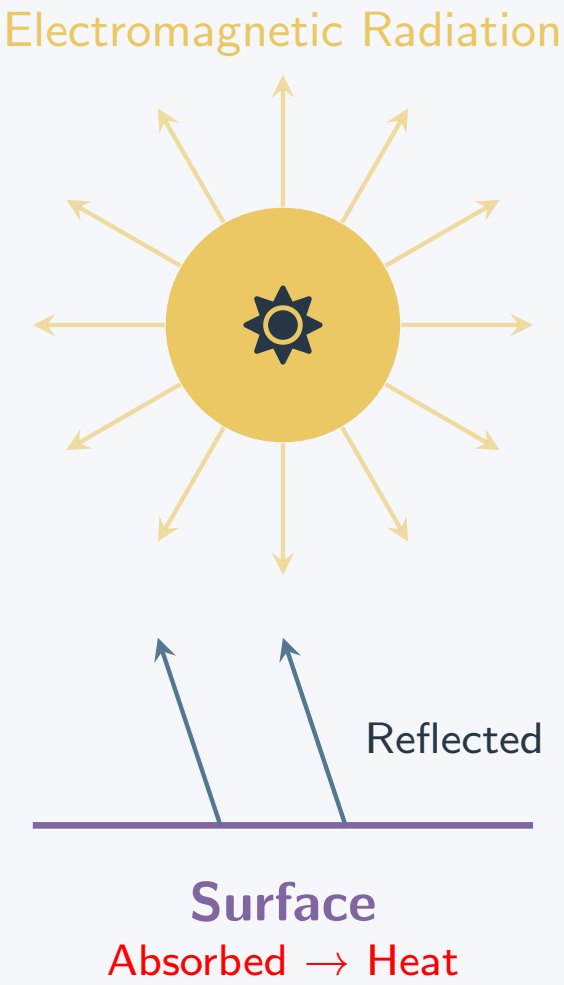
- Reflection types
- Surface normals
- Materials

- Ambient
- Diffuse
- Specular
- Mathematics

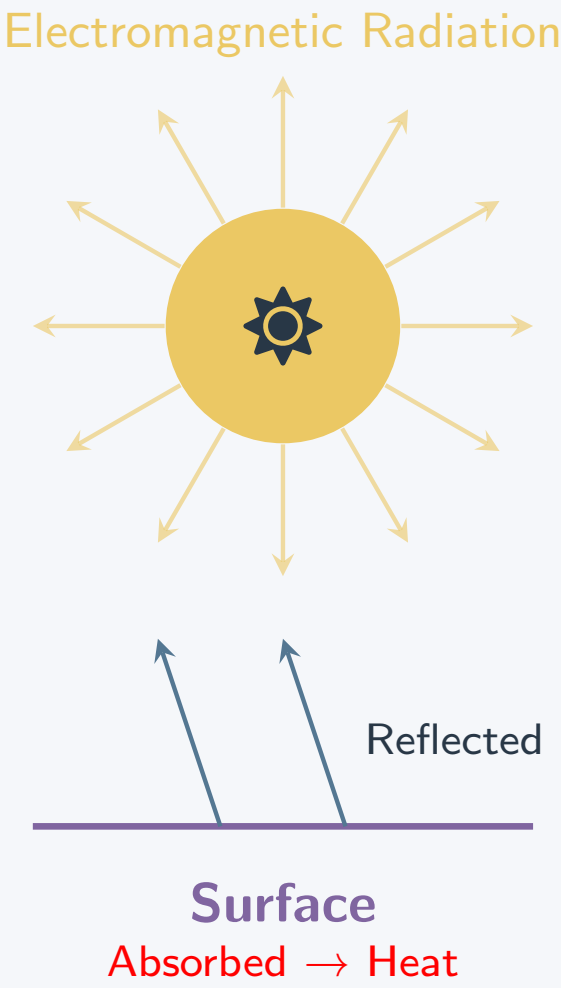
- Code structure
- Optimizations
- Edge cases

Light Sources

Light in Nature



Light in Nature



Physical Properties

Light is electromagnetic radiation:

- Wavelength determines color
- Intensity determines brightness
- Travels at speed of light (c)
- Behaves as waves and particles

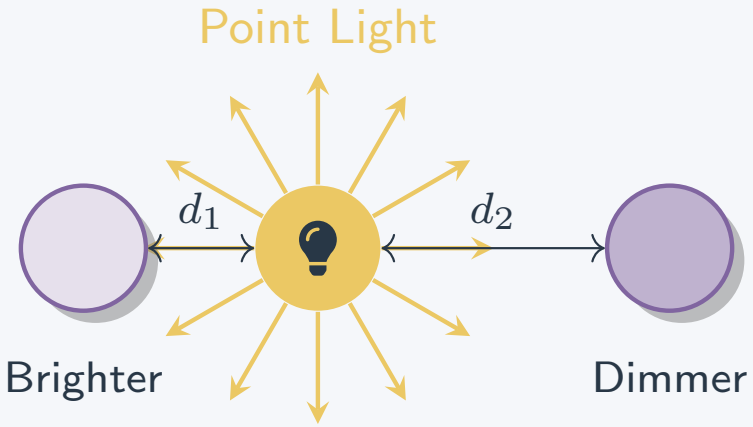
Surface interactions:

- **Reflection** (bounces off)
- Absorption (converts to heat)
- Transmission (passes through)

Point Light Sources - Introduction

Point Light Characteristics

Light emanating from a single point in all directions



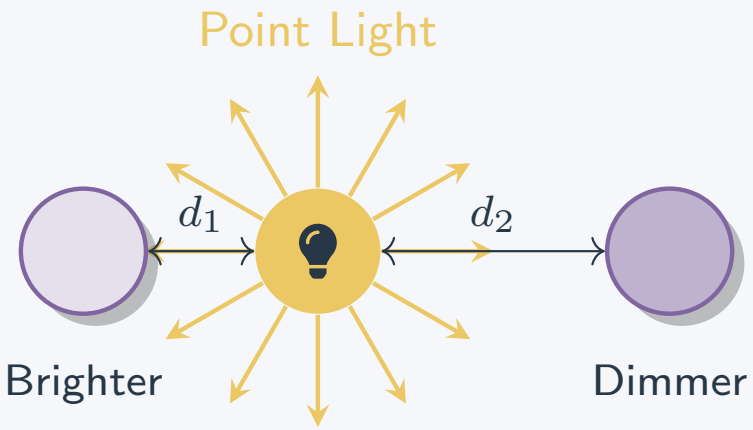
Point Light Sources - Introduction

Point Light Characteristics

Light emanating from a single point in all directions

Real-world examples:

- Light bulbs, LEDs
- Candles



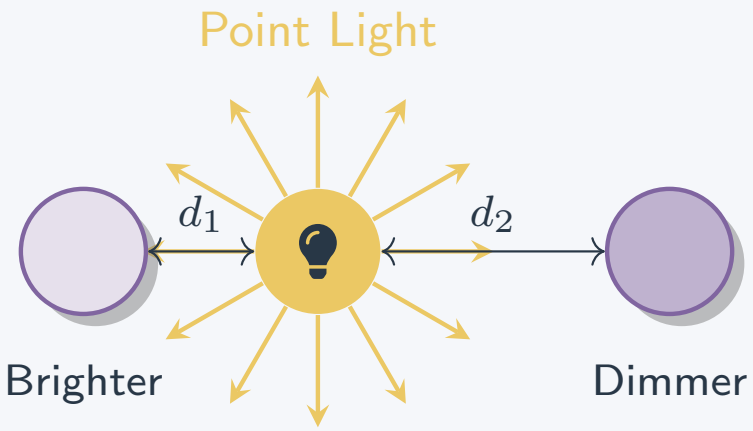
Point Light Sources - Introduction

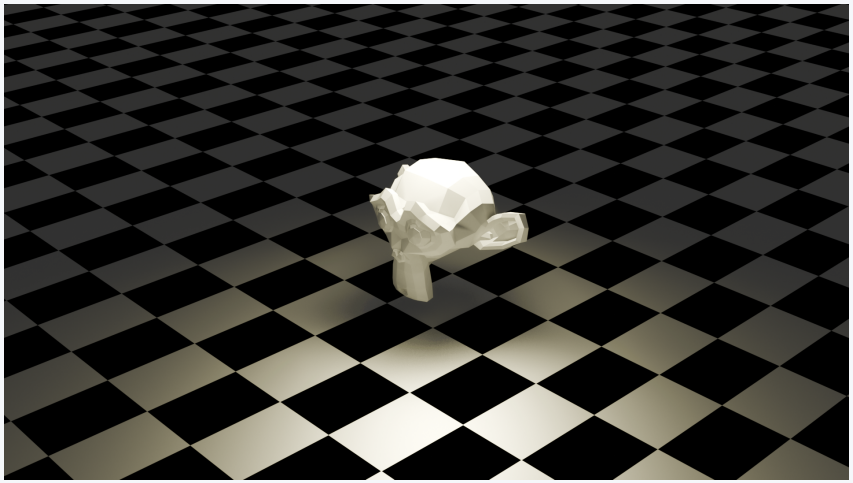
Point Light Characteristics

Light emanating from a single point in all directions

Real-world examples:

- Light bulbs, LEDs
- Candles





Point Light ft. Suzanne the monkey

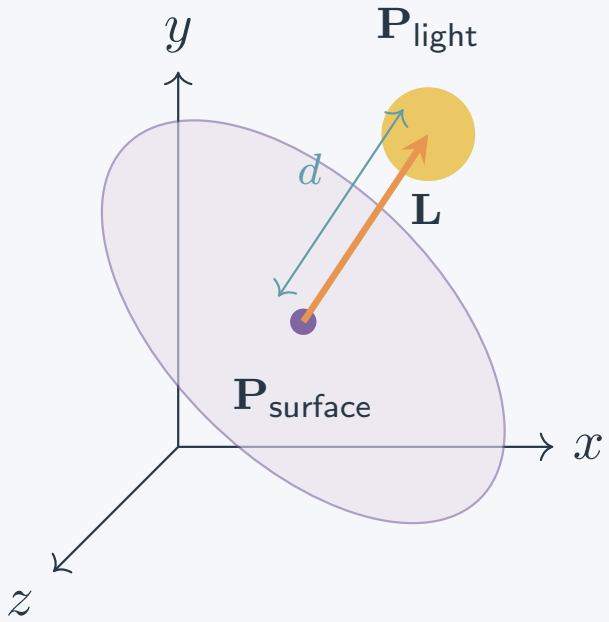
Point Light Mathematics

Point Light Parameters

Position: $\mathbf{P}_{\text{light}} = (x_l, y_l, z_l)$

Intensity: I_{light} (brightness)

Color: $\mathbf{C}_{\text{light}} = (r, g, b)$



Point Light Mathematics

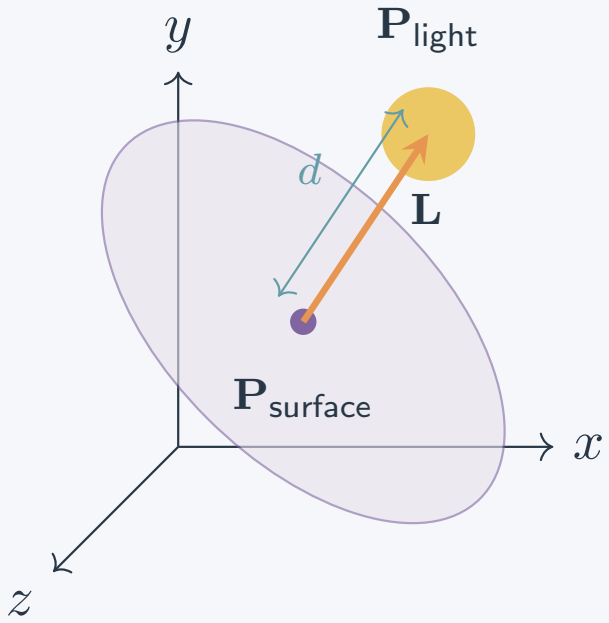
Point Light Parameters

Position: $\mathbf{P}_{\text{light}} = (x_l, y_l, z_l)$
Intensity: I_{light} (brightness)
Color: $\mathbf{C}_{\text{light}} = (r, g, b)$

Light direction to surface point:

$$\mathbf{L} = \mathbf{P}_{\text{light}} - \mathbf{P}_{\text{surface}}$$

$$\hat{\mathbf{L}} = \frac{\mathbf{L}}{|\mathbf{L}|} \quad \text{(normalized)}$$



Point Light Mathematics

Point Light Parameters

Position: $\mathbf{P}_{\text{light}} = (x_l, y_l, z_l)$
Intensity: I_{light} (brightness)
Color: $\mathbf{C}_{\text{light}} = (r, g, b)$

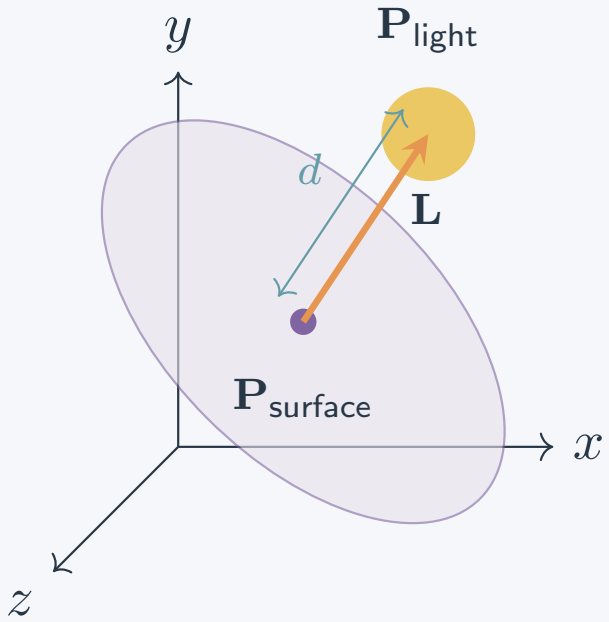
Light direction to surface point:

$$\mathbf{L} = \mathbf{P}_{\text{light}} - \mathbf{P}_{\text{surface}}$$

$$\hat{\mathbf{L}} = \frac{\mathbf{L}}{|\mathbf{L}|} \quad \text{(normalized)}$$

Distance:

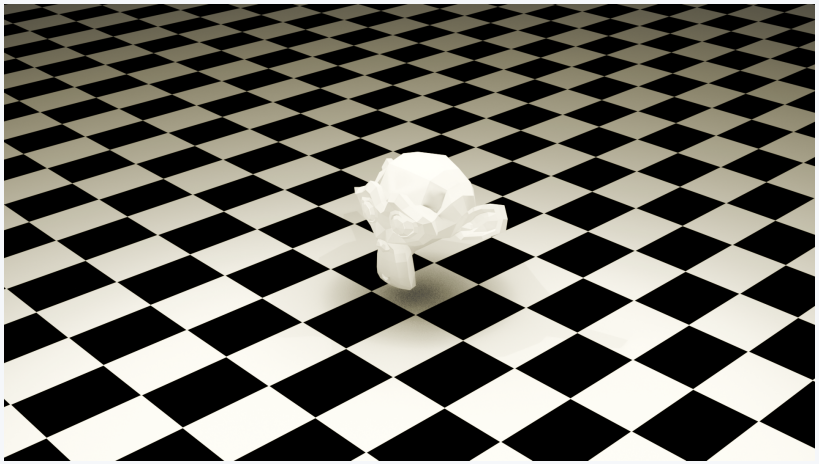
$$d = |\mathbf{P}_{\text{light}} - \mathbf{P}_{\text{surface}}|$$



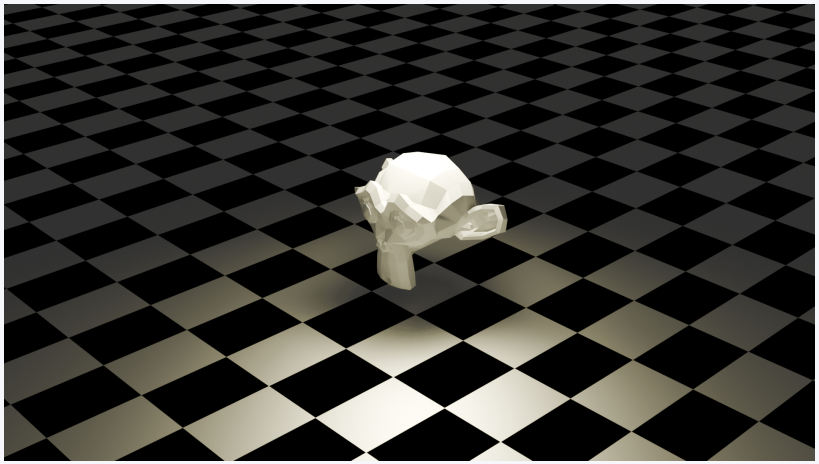
Attenuation

What is Attenuation?

Light becomes dimmer as distance increases due to the spreading of light energy over a larger area. Without this effect, distant objects would appear as bright as nearby ones, which is unrealistic.



No Attenuation

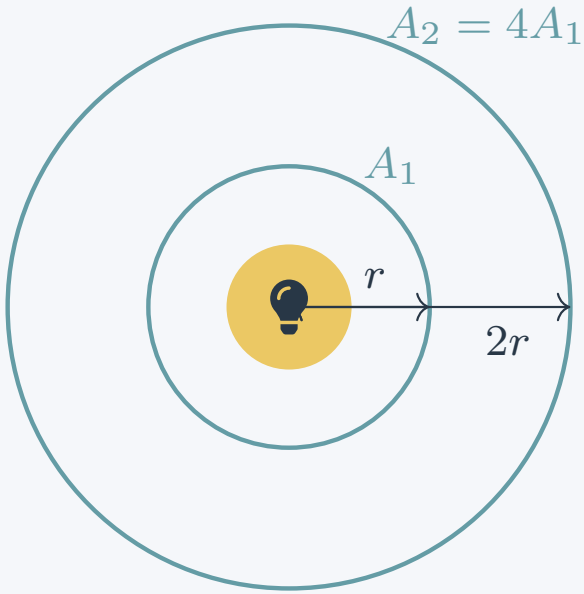


With Attenuation

Inverse Square Law - The Physics

$1/r^2$ Attenuation

- **Physical principle:** Light energy spreads over larger area as distance increases.

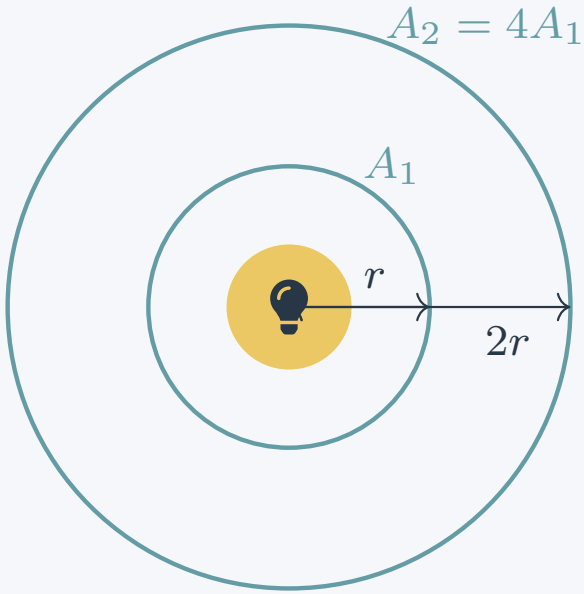


Same energy $\rightarrow$ 4$\times$ area $\rightarrow$ $\frac{1}{4}$ intensity

Inverse Square Law - The Physics

$1/r^2$ Attenuation

- **Physical principle:** Light energy spreads over larger area as distance increases.
- **Sphere surface area:** $A = 4\pi r^2$

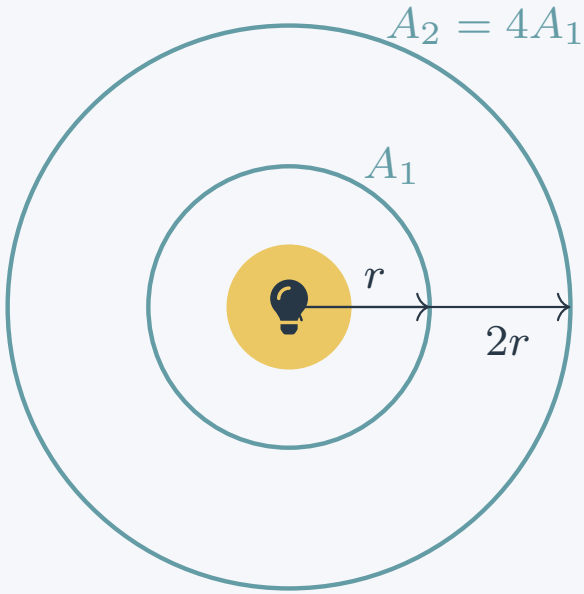


Same energy $\rightarrow$ 4$\times$ area $\rightarrow$ $\frac{1}{4}$ intensity

Inverse Square Law - The Physics

$1/r^2$ Attenuation

- **Physical principle:** Light energy spreads over larger area as distance increases.
- **Sphere surface area:** $A = 4\pi r^2$
- **Energy conservation:** Same total energy spread over larger area.

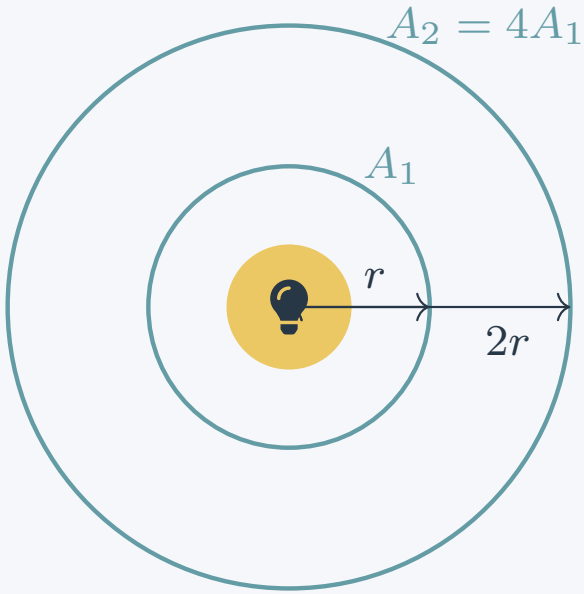


Same energy $\rightarrow$ 4$\times$ area $\rightarrow$ $\frac{1}{4}$ intensity

Inverse Square Law - The Physics

$1/r^2$ Attenuation

- **Physical principle:** Light energy spreads over larger area as distance increases.
- **Sphere surface area:** $A = 4\pi r^2$
- **Energy conservation:** Same total energy spread over larger area.
- **Intensity per unit area:** $I \propto \frac{1}{r^2}$

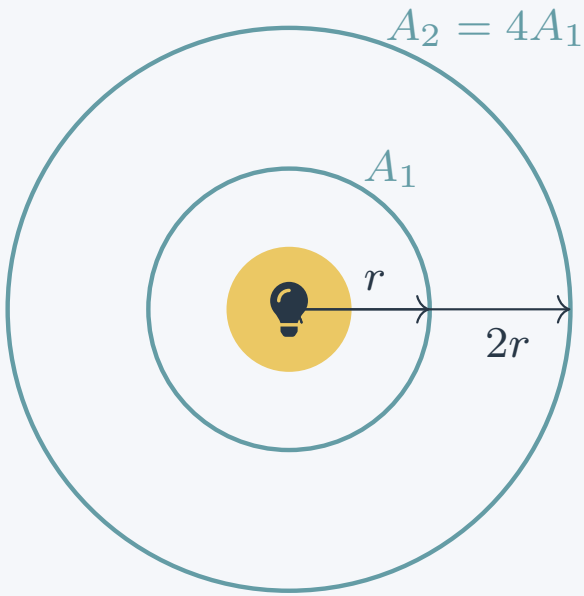


Same energy $\rightarrow$ 4$\times$ area $\rightarrow$ $\frac{1}{4}$ intensity

Inverse Square Law - The Physics

$1/r^2$ Attenuation

- **Physical principle:** Light energy spreads over larger area as distance increases.
- **Sphere surface area:** $A = 4\pi r^2$
- **Energy conservation:** Same total energy spread over larger area.
- **Intensity per unit area:** $I \propto \frac{1}{r^2}$



Same energy $\rightarrow$ 4$\times$ area $\rightarrow$ $\frac{1}{4}$ intensity

Attenuation Formula

$$I_{\text{received}} = \frac{I_{\text{light}}}{d^2} \quad \text{where } d = \text{distance to light}$$

Point Light Implementation

Point Light Function Structure

Input parameters:

- Light position: $\mathbf{P}_{\text{light}}$
- Light intensity: I_{light}
- Light color: $\mathbf{C}_{\text{light}} = (r, g, b)$
- Surface point: $\mathbf{P}_{\text{surface}}$

Point Light Implementation

Point Light Function Structure

Calculation steps:

$$\mathbf{L} = \mathbf{P}_{\text{light}} - \mathbf{P}_{\text{surface}}$$

$$d = |\mathbf{L}|$$

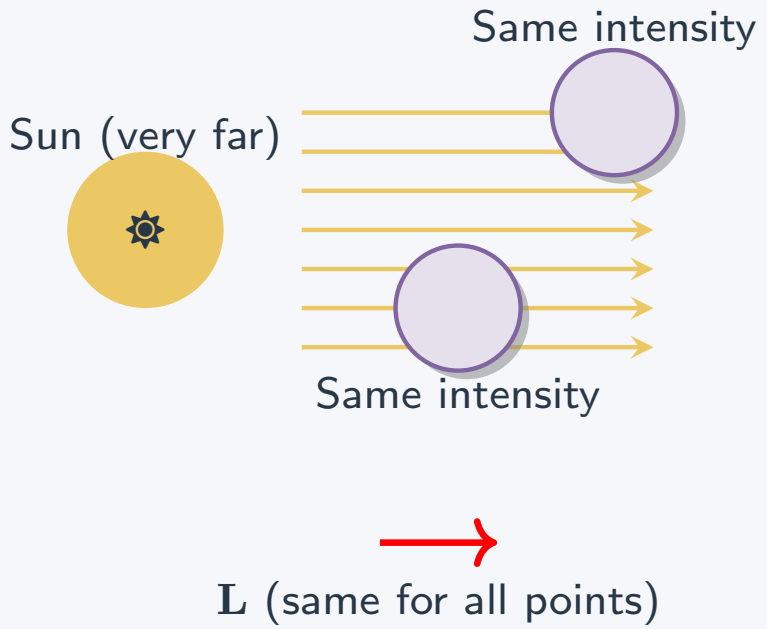
$$\hat{\mathbf{L}} = \mathbf{L}/d$$

$$I_{\text{final}} = \frac{I_{\text{light}}}{\epsilon + d^2}$$

Directional Lights - The Sun Model

Directional Light Concept

Light source at infinite distance, so rays are effectively parallel



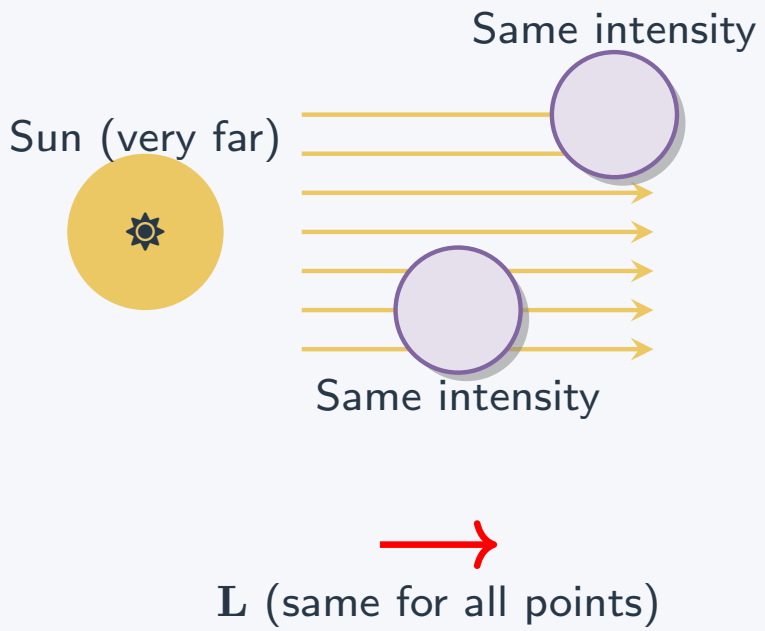
Directional Lights - The Sun Model

Directional Light Concept

Light source at infinite distance, so rays are effectively parallel

Characteristics:

- Parallel rays
- Same intensity everywhere
- No attenuation
- Defined by direction only

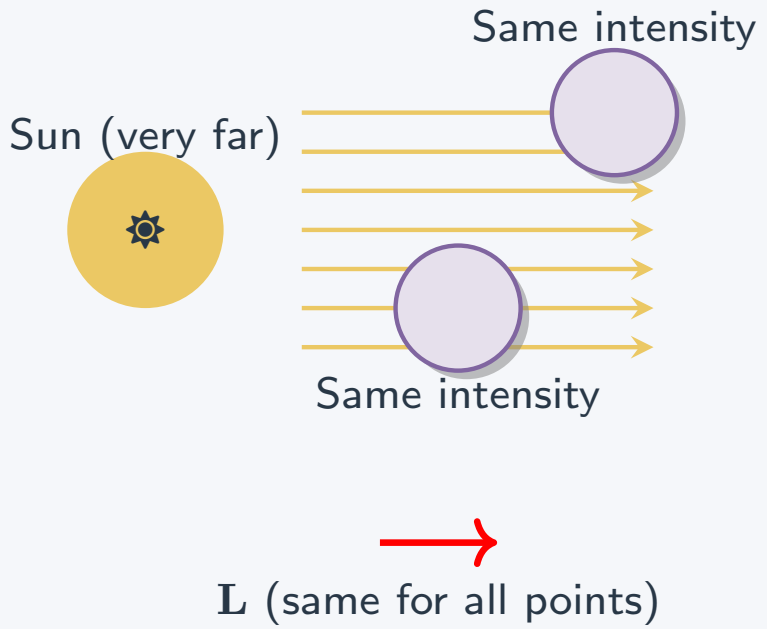


Directional Lights - The Sun Model

Directional Light Concept

Light source at infinite distance, so rays are effectively parallel

Perfect for: Sun, moon, distant lights



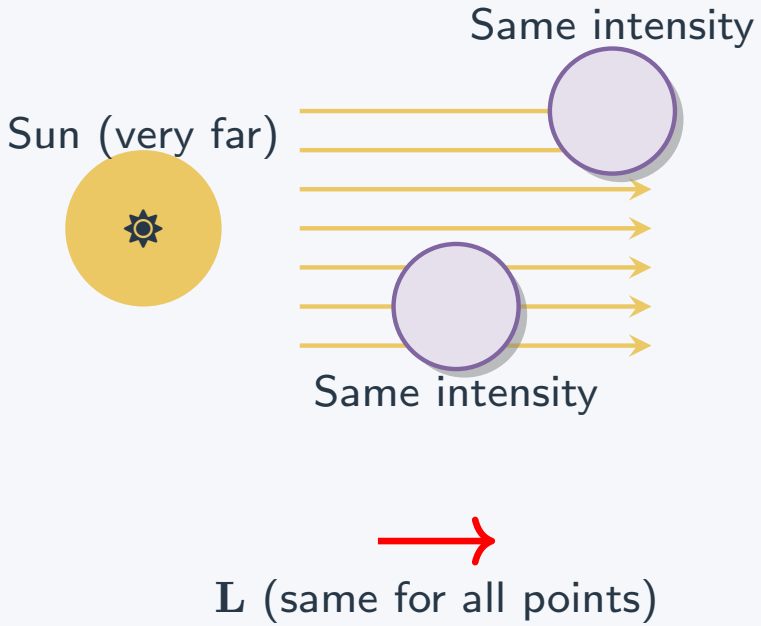
Directional Lights - The Sun Model

Directional Light Concept

Light source at infinite distance, so rays are effectively parallel

Perfect for: Sun, moon, distant lights

Directional Light ft. Suzanne



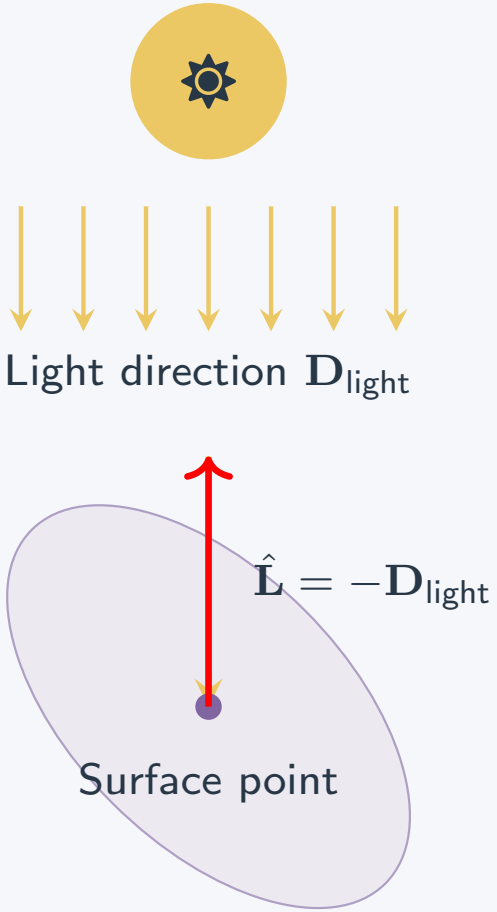
Directional Light Mathematics

Directional Light Parameters

Direction: $\mathbf{D}_{\text{light}} = (x, y, z)$

Intensity: I_{light} (constant)

Color: $\mathbf{C}_{\text{light}} = (r, g, b)$



Directional Light Mathematics

Directional Light Parameters

Direction: $\mathbf{D}_{\text{light}} = (x, y, z)$

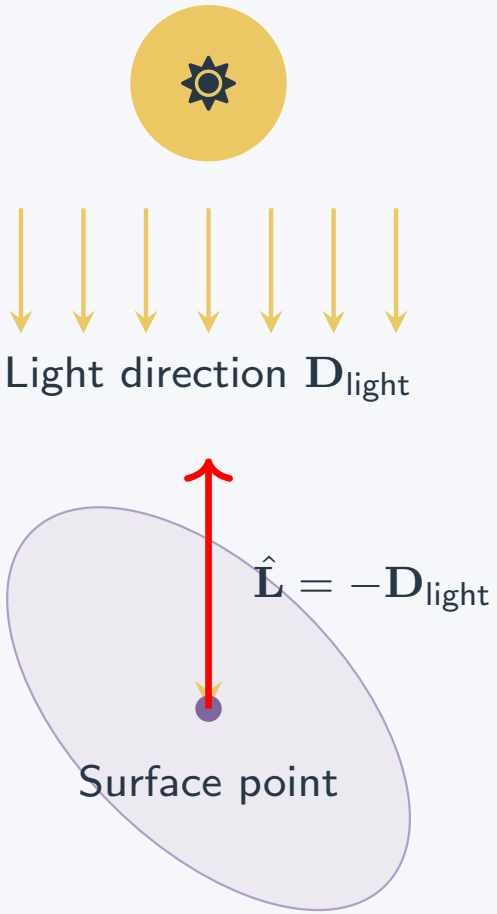
Intensity: I_{light} (constant)

Color: $\mathbf{C}_{\text{light}} = (r, g, b)$

Light direction:

$$\hat{\mathbf{L}} = -\mathbf{D}_{\text{light}}$$

Note: Assuming $\mathbf{D}_{\text{light}}$ is normalized



Directional Light Mathematics

Directional Light Parameters

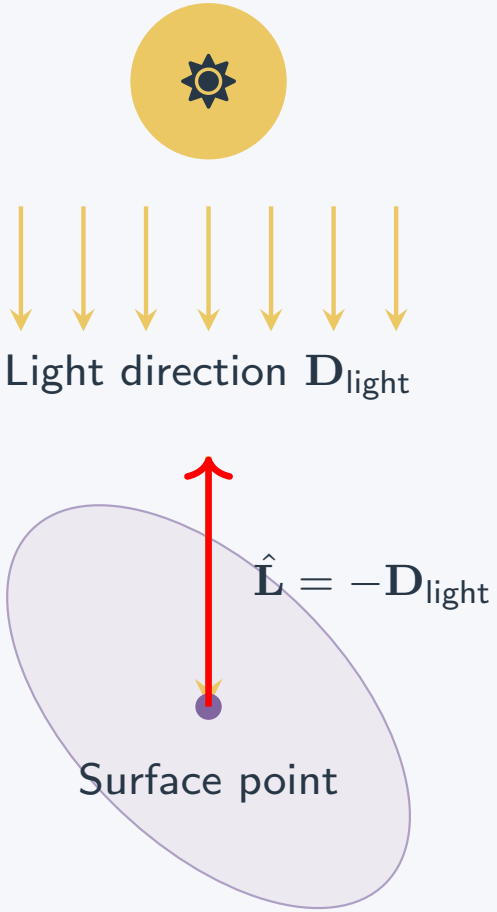
Direction: $\mathbf{D}_{\text{light}} = (x, y, z)$
Intensity: I_{light} (constant)
Color: $\mathbf{C}_{\text{light}} = (r, g, b)$

Light direction:

$$\hat{\mathbf{L}} = -\mathbf{D}_{\text{light}}$$

Intensity at any point:

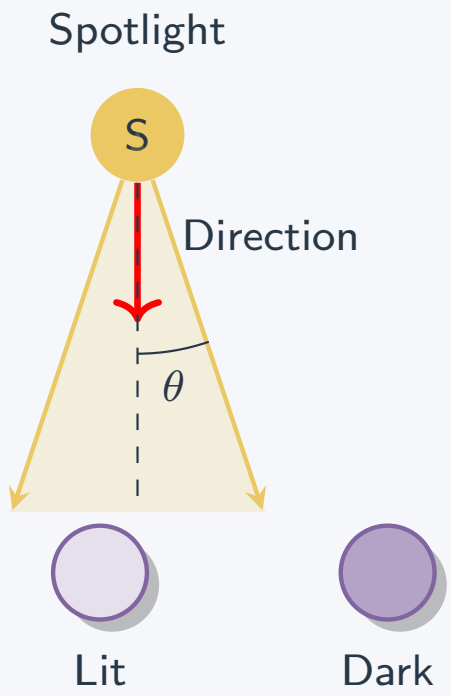
$$I_{\text{final}} = I_{\text{light}} \quad \text{(no attenuation)}$$



Spot Lights - Introduction

Spot Light Characteristics

Light emanating from a point within a cone



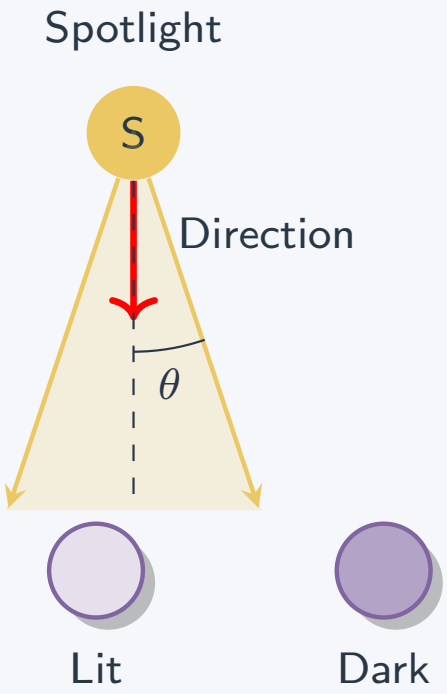
Spot Lights - Introduction

Spot Light Characteristics

Light emanating from a point within a cone

Real-world examples:

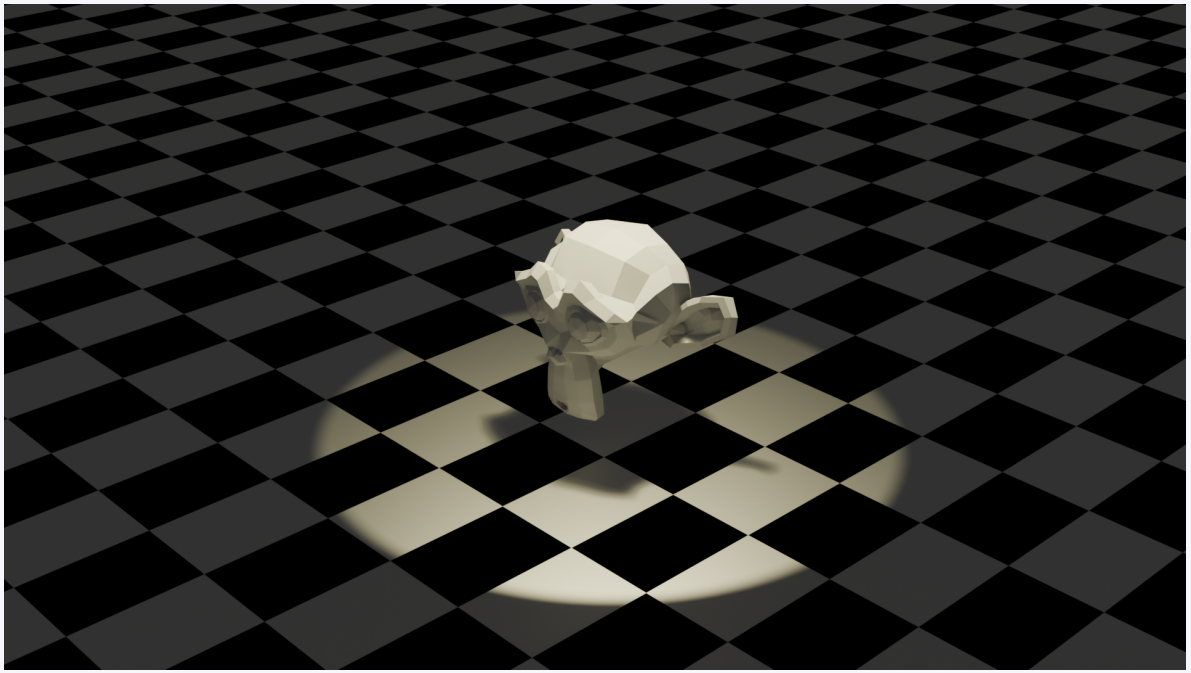
- Flashlights, headlights
- Stage spotlights
- Desk lamps



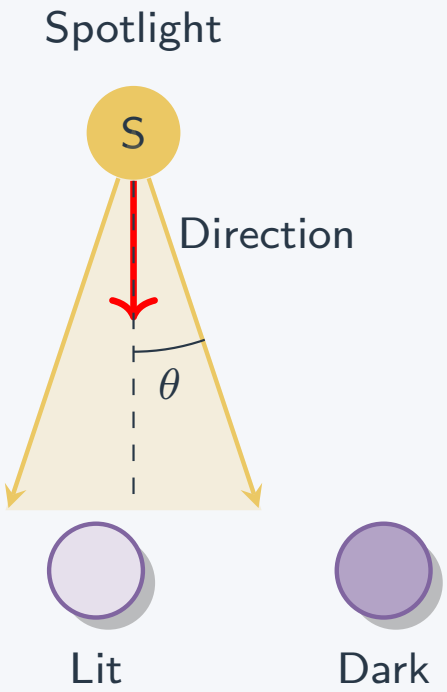
Spot Lights - Introduction

Spot Light Characteristics

Light emanating from a point within a cone



Spot Light ft. Suzanne the monkey



Spot Light Mathematics - Cone Calculation

Spot Light Parameters

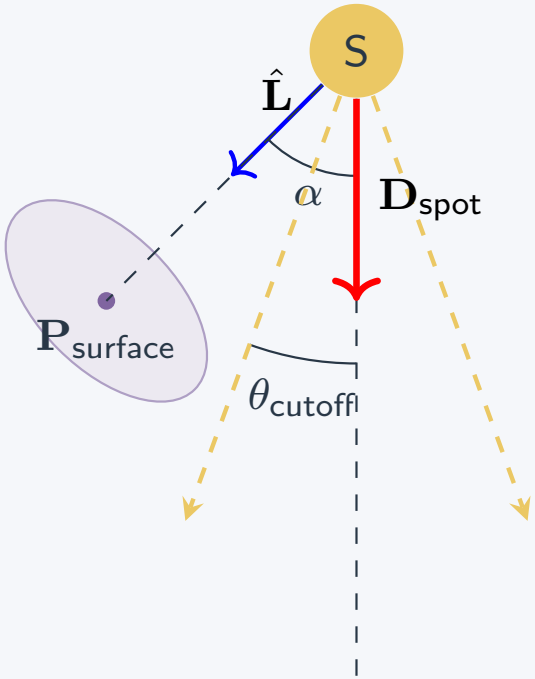
Position: $\mathbf{P}_{\text{light}}$

Direction: $\mathbf{D}_{\text{spot}}$

Color: $\mathbf{C}_{\text{light}} = (r, g, b)$

Cone angle: θ_{cutoff}

Falloff exponent: e



Spot Light Mathematics - Cone Calculation

Spot Light Parameters

Position: $\mathbf{P}_{\text{light}}$

Direction: $\mathbf{D}_{\text{spot}}$

Color: $\mathbf{C}_{\text{light}} = (r, g, b)$

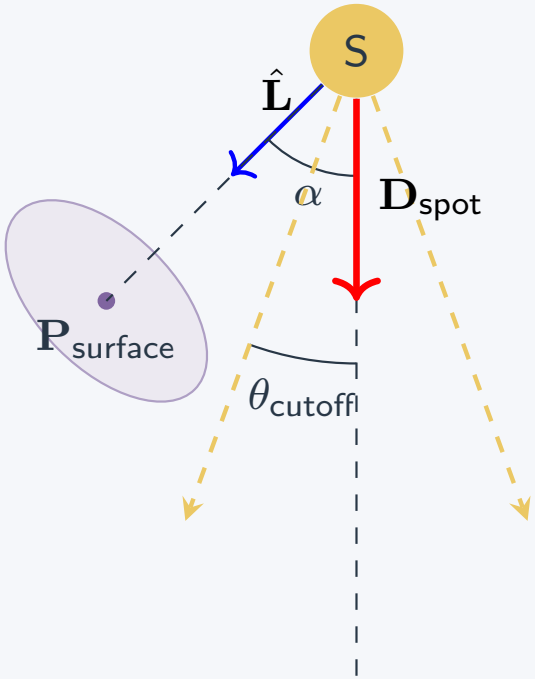
Cone angle: θ_{cutoff}

Falloff exponent: e

Step 1 - Calculate angle to surface:

$$\hat{\mathbf{L}} = \frac{\mathbf{P}_{\text{light}} - \mathbf{P}_{\text{surface}}}{|\mathbf{P}_{\text{light}} - \mathbf{P}_{\text{surface}}|}$$

$$\cos(\alpha) = \mathbf{D}_{\text{spot}} \cdot (-\hat{\mathbf{L}})$$



Spot Light Mathematics - Cone Calculation

Spot Light Parameters

Position: $\mathbf{P}_{\text{light}}$

Direction: $\mathbf{D}_{\text{spot}}$

Color: $\mathbf{C}_{\text{light}} = (r, g, b)$

Cone angle: θ_{cutoff}

Falloff exponent: e

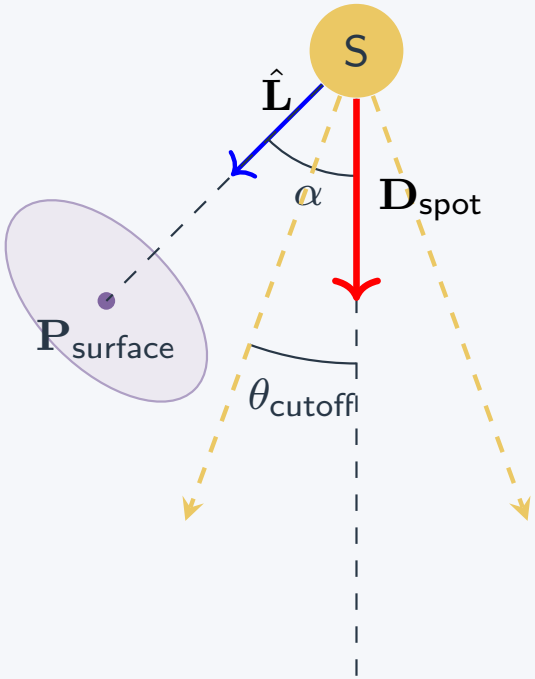
Step 1 - Calculate angle to surface:

$$\hat{\mathbf{L}} = \frac{\mathbf{P}_{\text{light}} - \mathbf{P}_{\text{surface}}}{|\mathbf{P}_{\text{light}} - \mathbf{P}_{\text{surface}}|}$$

$$\cos(\alpha) = \mathbf{D}_{\text{spot}} \cdot (-\hat{\mathbf{L}})$$

Step 2 - Check if inside cone:

$$\text{if } \cos(\alpha) > \cos(\theta_{\text{cutoff}}) \text{ then illuminate}$$



Spot Light Attenuation

Complete Spot Light Formula

Angular attenuation:

$$\text{spot\_factor} = \begin{cases} (\cos(\alpha))^e & \text{if } \cos(\alpha) > \cos(\theta_{\text{cutoff}}) \\ 0 & \text{otherwise} \end{cases}$$

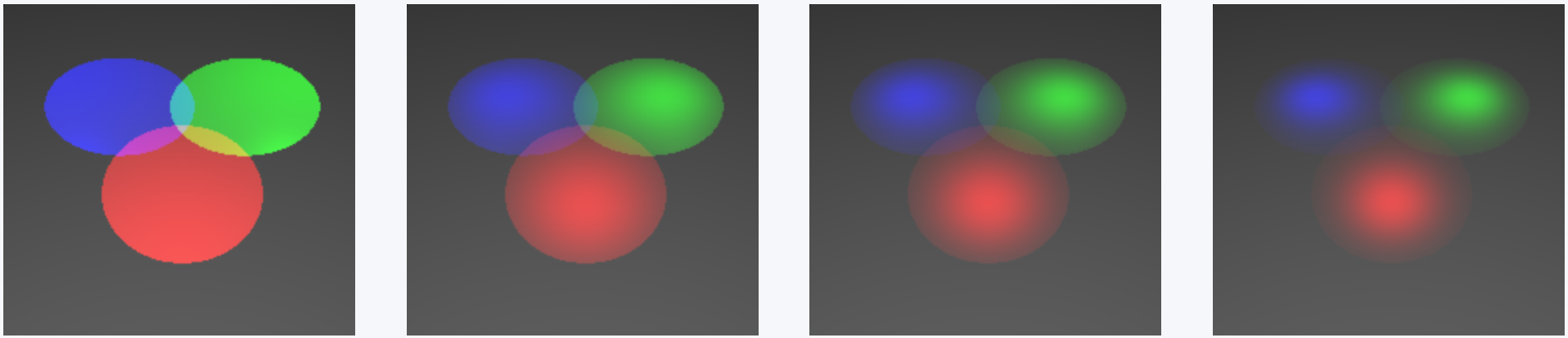
Spot Light Attenuation

Complete Spot Light Formula

Angular attenuation:

$$\text{spot\_factor} = \begin{cases} (\cos(\alpha))^e & \text{if } \cos(\alpha) > \cos(\theta_{\text{cutoff}}) \\ 0 & \text{otherwise} \end{cases}$$

The e exponent controls how much brighter the light is at the center of the cone compared to the edges.



$e = 0$ $e = 10$ $e = 20$ $e = 30$

Spot lights for different values of e

Spot Light Attenuation

Complete Spot Light Formula

Angular attenuation:

$$\text{spot\_factor} = \begin{cases} (\cos(\alpha))^e & \text{if } \cos(\alpha) > \cos(\theta_{\text{cutoff}}) \\ 0 & \text{otherwise} \end{cases}$$

Distance attenuation (same as point light):

$$\text{distance\_attenuation} = \frac{1}{\epsilon + d^2}$$

Spot Light Attenuation

Complete Spot Light Formula

Angular attenuation:

$$\text{spot\_factor} = \begin{cases} (\cos(\alpha))^e & \text{if } \cos(\alpha) > \cos(\theta_{\text{cutoff}}) \\ 0 & \text{otherwise} \end{cases}$$

Distance attenuation (same as point light):

$$\text{distance\_attenuation} = \frac{1}{\epsilon + d^2}$$

Final intensity:

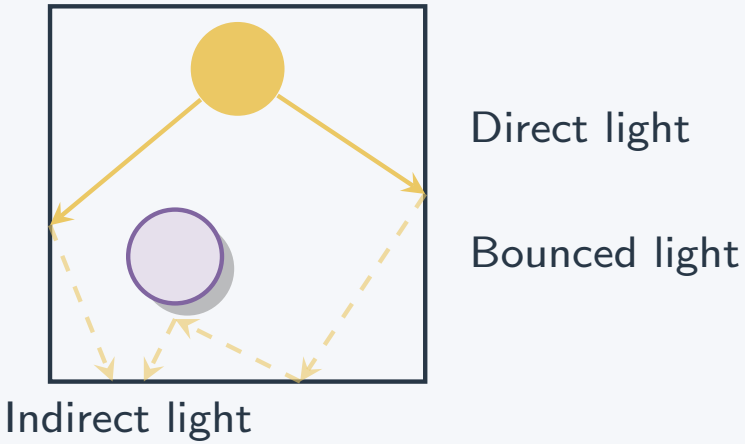
$$I_{\text{final}} = \begin{cases} I_{\text{light}} \cdot (\cos(\alpha))^e \cdot \frac{1}{\epsilon + d^2} & \text{if } \cos(\alpha) > \cos(\theta_{\text{cutoff}}) \\ 0 & \text{otherwise} \end{cases}$$

Ambient Light - Global Illumination Approximation

Direct Lighting Problem

Real scenes have indirect lighting

- Light bounces off walls, ceiling
- Reflections illuminate shadows
- Even "dark" areas receive some light

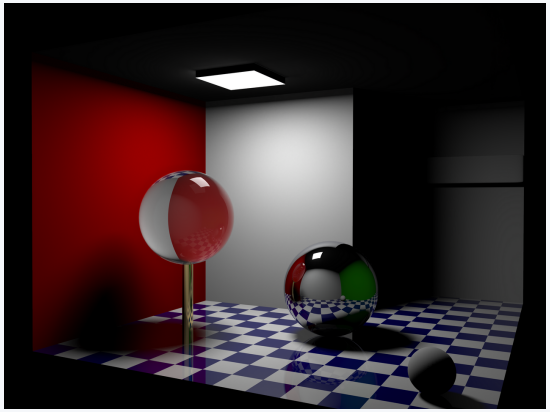


Ambient Light - Global Illumination Approximation

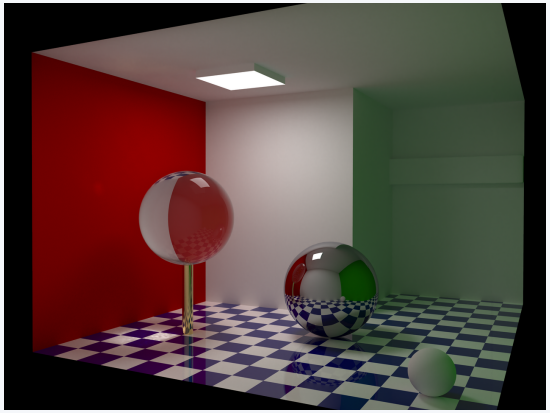
Direct Lighting Problem

Real scenes have indirect lighting

- Light bounces off walls, ceiling
- Reflections illuminate shadows
- Even "dark" areas receive some light



Only direct light



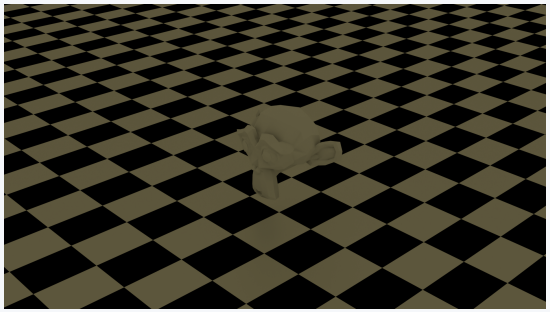
Global illumination

Ambient Light - Global Illumination Approximation

Solution: Ambient Light

Add constant ambient term

- Prevents completely black shadows
- Approximates global illumination
- Simple and fast to compute

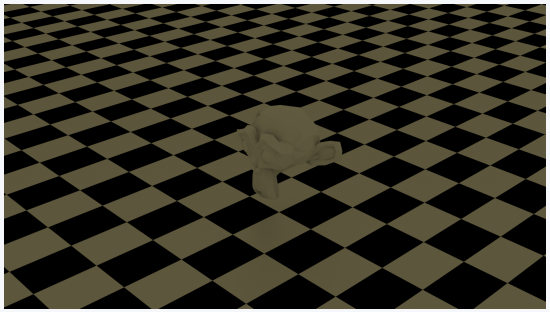


Ambient light ft. Suzanne the monkey

Ambient Light - Global Illumination Approximation

Ambient Light Parameters

Ambient intensity: I_{light} (constant)
Color: $\mathbf{C}_{\text{light}} = (r, g, b)$



Ambient light ft. Suzanne the monkey

The Phong Illumination Model

Phong Model: History

Historical Context

Developed by **Bui Tuong Phong** as his PhD dissertation at the University of Utah in 1973.
Goal: Fast, realistic-looking lighting

In Phong's words:

> *"We do not expect to be able to display the object exactly as it would appear in reality, with texture, overcast shadows, etc. We hope only to display an image that approximates the real object closely enough to provide a certain degree of realism."*

Phong completed his PhD in only **two years**, a record at the time. Yet tragically, he died of cancer only two years later.

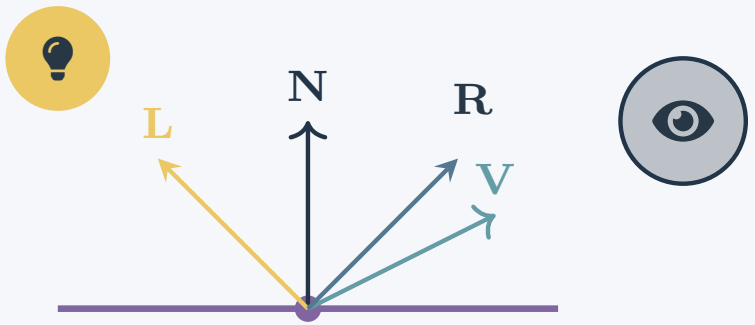
Bui Tuong Phong (1942-1975)

Phong Model Overview

Phong Formula

Break lighting into three components that can be computed independently

$$
\begin{aligned}
I &= I_{\text{ambient}} + I_{\text{diffuse}} + I_{\text{specular}} \\
&= k_a I_a + k_d I_l (\mathbf{N} \cdot \mathbf{L}) + k_s I_l (\mathbf{R} \cdot \mathbf{V})^n
\end{aligned}
$$

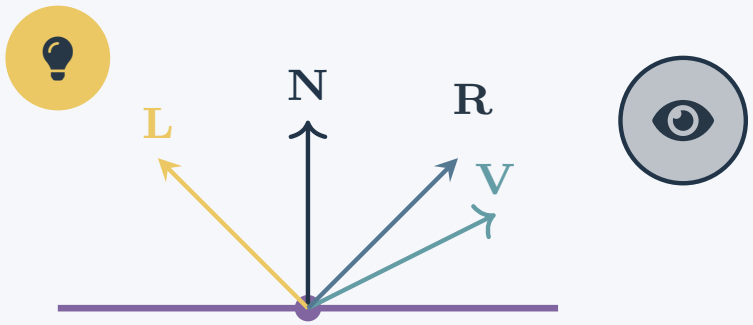


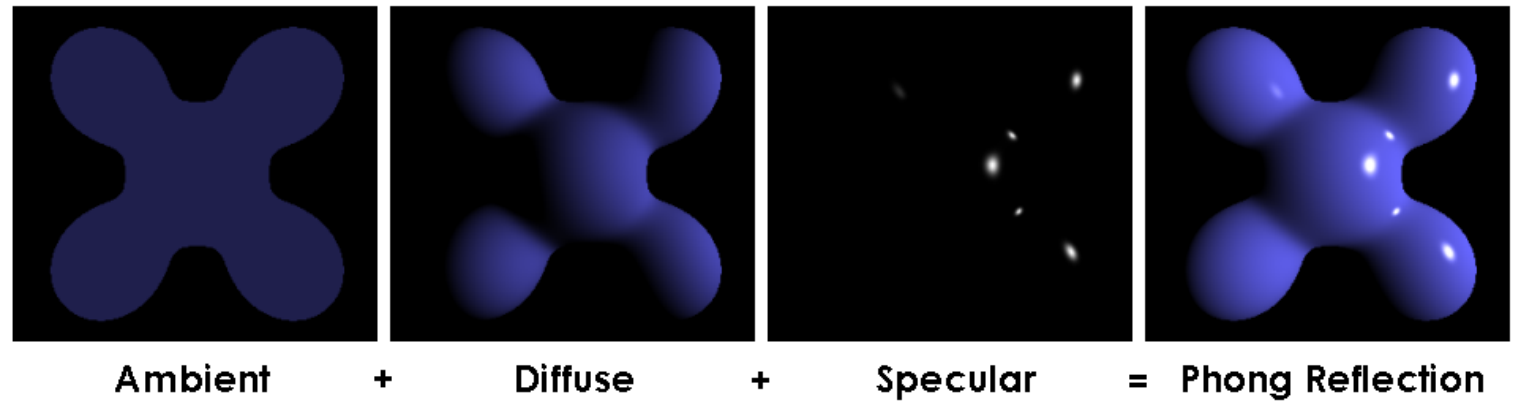
Phong Model Overview

Phong Formula

Break lighting into three components that can be computed independently

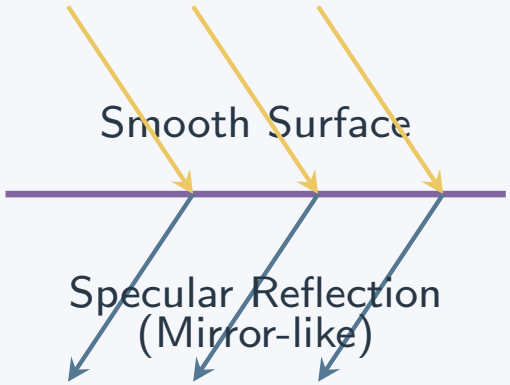
$$I = I_{\text{ambient}} + I_{\text{diffuse}} + I_{\text{specular}}$$
$$= k_a I_a + k_d I_l(\mathbf{N} \cdot \mathbf{L}) + k_s I_l(\mathbf{R} \cdot \mathbf{V})^n$$





Phong model components

Types of Reflection



Reflection Types

Specular Reflection:

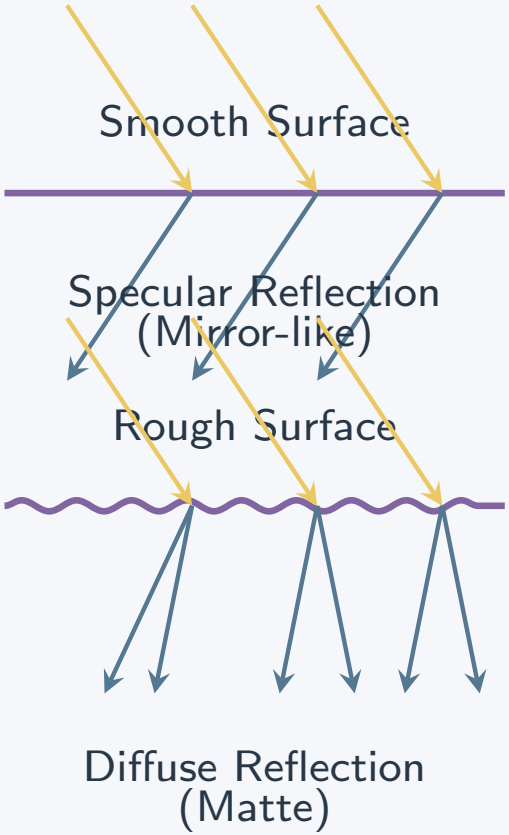
- Smooth surfaces (mirrors, metals)
- Preserves light direction
- Creates sharp highlights
- View-dependent

Diffuse Reflection:

- Rough surfaces (paper, clay, fabric)
- Scatters light uniformly
- View-independent
- Lambertian behavior

Real surfaces: Combination of both types

Types of Reflection



Reflection Types

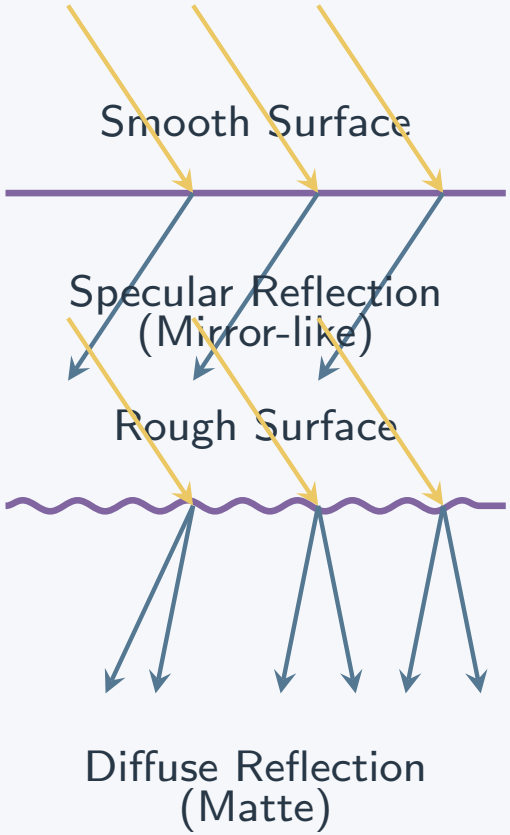
Specular Reflection:

- Smooth surfaces (mirrors, metals)
- Preserves light direction
- Creates sharp highlights
- View-dependent

Diffuse Reflection:

- Rough surfaces (paper, clay, fabric)
- Scatters light uniformly
- View-independent
- Lambertian behavior

Types of Reflection



Reflection Types

Specular Reflection:

- Smooth surfaces (mirrors, metals)
- Preserves light direction
- Creates sharp highlights
- View-dependent

Diffuse Reflection:

- Rough surfaces (paper, clay, fabric)
- Scatters light uniformly
- View-independent
- Lambertian behavior

Real surfaces: Combination of both types

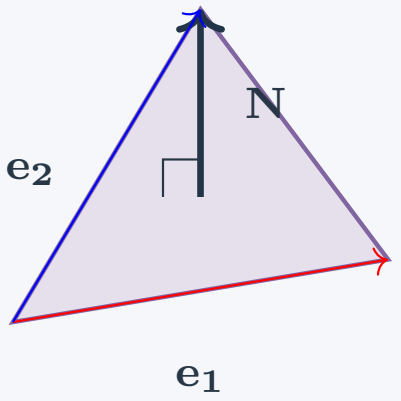
Surface Normals - The Key Vector

Surface Normal Properties

Definition: Vector perpendicular to surface at a point

Properties:

- Unit length: $|\mathbf{N}| = 1$
- Points "outward" from surface
- Determines light interaction



Normal calculation:
$\mathbf{N} = \mathbf{e_1} \times \mathbf{e_2}$

Surface Normals - The Key Vector

Surface Normal Properties

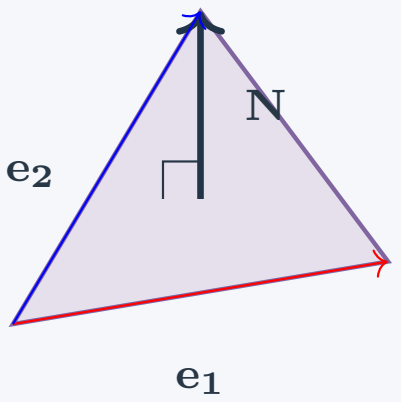
Definition: Vector perpendicular to surface at a point

Properties:

- Unit length: $|\mathbf{N}| = 1$
- Points "outward" from surface
- Determines light interaction

For polygons: Cross product of edge vectors

$$\mathbf{N} = \frac{\mathbf{e_1} \times \mathbf{e_2}}{|\mathbf{e_1} \times \mathbf{e_2}|} \tag{1}$$



Normal calculation:
$\mathbf{N} = \mathbf{e_1} \times \mathbf{e_2}$

Surface Normals - The Key Vector

Surface Normal Properties

Definition: Vector perpendicular to surface at a point

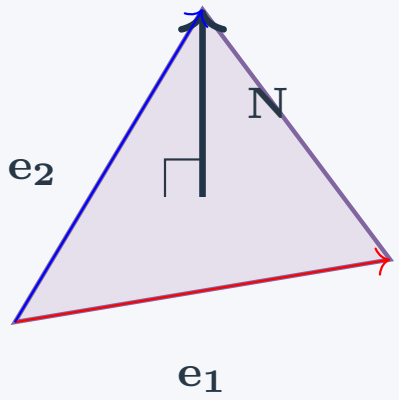
Properties:

- Unit length: $|\mathbf{N}| = 1$
- Points "outward" from surface
- Determines light interaction

For polygons: Cross product of edge vectors

$$\mathbf{N} = \frac{\mathbf{e_1} \times \mathbf{e_2}}{|\mathbf{e_1} \times \mathbf{e_2}|} \qquad (1)$$

For implicit surfaces: Gradient of surface function



Surface Normals - The Key Vector

Surface Normal Properties

Definition: Vector perpendicular to surface at a point

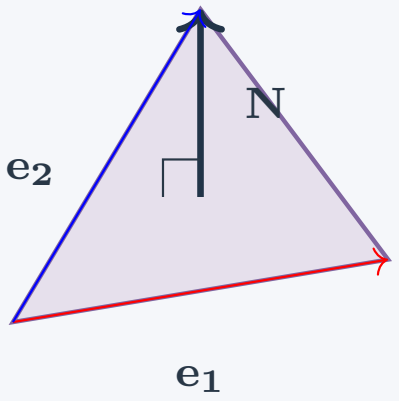
Properties:

- Unit length: $|\mathbf{N}| = 1$
- Points "outward" from surface
- Determines light interaction

For polygons: Cross product of edge vectors

$$\mathbf{N} = \frac{\mathbf{e_1} \times \mathbf{e_2}}{|\mathbf{e_1} \times \mathbf{e_2}|} \qquad (1)$$

For implicit surfaces: Gradient of surface function



Normal calculation:
$\mathbf{N} = \mathbf{e_1} \times \mathbf{e_2}$

Surface Normals - The Key Vector

Surface Normal Properties

Definition: Vector perpendicular to surface at a point

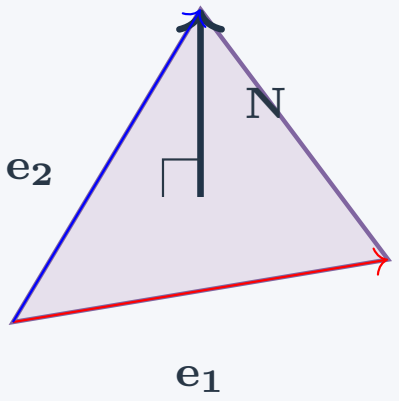
Properties:

- Unit length: $|\mathbf{N}| = 1$
- Points "outward" from surface
- Determines light interaction

For polygons: Cross product of edge vectors

$$\mathbf{N} = \frac{\mathbf{e_1} \times \mathbf{e_2}}{|\mathbf{e_1} \times \mathbf{e_2}|} \qquad (1)$$

For implicit surfaces: Gradient of surface function



Normal calculation:
$\mathbf{N} = \mathbf{e_1} \times \mathbf{e_2}$

Material Properties

Material Coefficients

Ambient coefficient: k_a

- Controls response to ambient light
- Range: $[0, 1]$
- Usually small (0.1 - 0.3)

Diffuse coefficient: k_d

- Controls Lambertian reflection
- Range: $[0, 1]$
- Represents surface color

Specular coefficient: k_s

- Controls shiny highlights
- Range: $[0, 1]$
- Higher for metallic surfaces

Shininess: n

[Material parameter

examples]

Typical Values

Matte plastic:
$k_a = 0.2, k_d = 0.8, k_s = 0.1, n = 10$

Shiny metal:
$k_a = 0.1, k_d = 0.3, k_s = 0.9, n = 100$

Rubber:

Material Properties

Material Coefficients

Ambient coefficient: k_a

- Controls response to ambient light
- Range: $[0, 1]$
- Usually small (0.1 - 0.3)

Diffuse coefficient: k_d

- Controls Lambertian reflection
- Range: $[0, 1]$
- Represents surface color

Specular coefficient: k_s

- Controls shiny highlights
- Range: $[0, 1]$
- Higher for metallic surfaces

Shininess: n

[Material parameter

examples]

Typical Values

Matte plastic:
$k_a = 0.2, k_d = 0.8, k_s = 0.1, n = 10$

Shiny metal:
$k_a = 0.1, k_d = 0.3, k_s = 0.9, n = 100$

Rubber:

Ambient Reflection - Mathematics

Ambient Component Formula

Simplest lighting component:

$$I_{\text{ambient}} = k_a \cdot I_a \tag{3}$$

where:

- k_a = ambient reflection coefficient of the material
- I_a = intensity of ambient light in the scene

Ambient Reflection - Mathematics

Ambient Component Formula

Simplest lighting component:

$$I_{\text{ambient}} = k_a \cdot I_a \tag{3}$$

where:

- k_a = ambient reflection coefficient of the material
- I_a = intensity of ambient light in the scene

Key properties:

- Independent of viewer position
- Independent of light direction
- Independent of surface normal
- Same for all points on the surface

Ambient Reflection - Mathematics

Ambient Component Formula

Simplest lighting component:

$$I_{\text{ambient}} = k_a \cdot I_a \tag{3}$$

where:

- k_a = ambient reflection coefficient of the material
- I_a = intensity of ambient light in the scene

Key properties:

- Independent of viewer position
- Independent of light direction
- Independent of surface normal
- Same for all points on the surface

Typical values:

Ambient Reflection - Mathematics

Ambient Component Formula

Simplest lighting component:

$$I_{\text{ambient}} = k_a \cdot I_a \tag{3}$$

where:

- k_a = ambient reflection coefficient of the material
- I_a = intensity of ambient light in the scene

Key properties:

- Independent of viewer position
- Independent of light direction
- Independent of surface normal
- Same for all points on the surface

Typical values:

Diffuse Reflection - Introduction

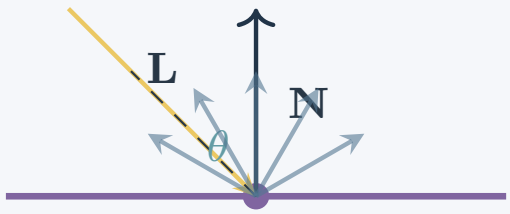
Lambertian Surfaces

Examples:

- Matte paint
- Unpolished wood
- Paper
- Clay
- Fabric

Characteristics:

- Surface appears equally bright from all viewing angles
- Light scattered uniformly in all directions



Equal brightness all directions

Diffuse Reflection - Introduction

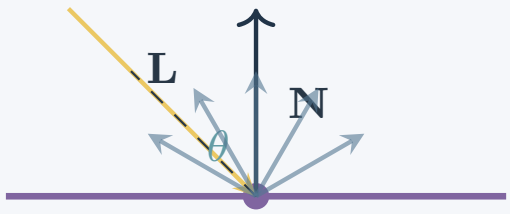
Lambertian Surfaces

Examples:

- Matte paint
- Unpolished wood
- Paper
- Clay
- Fabric

Characteristics:

- Surface appears equally bright from all viewing angles
- Light scattered uniformly in all directions

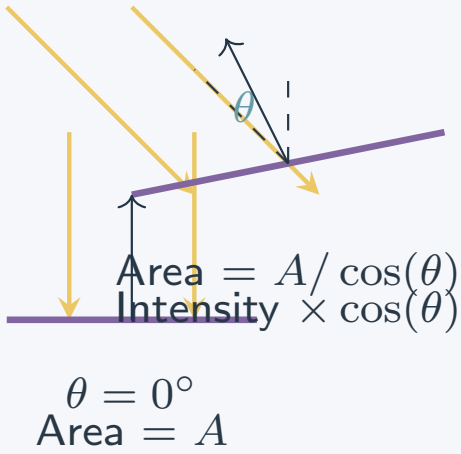


Equal brightness all directions

Lambert's Cosine Law - Geometric Derivation

Why Cosine?

Consider light hitting a surface:



Lambert's Cosine Law - Geometric Derivation

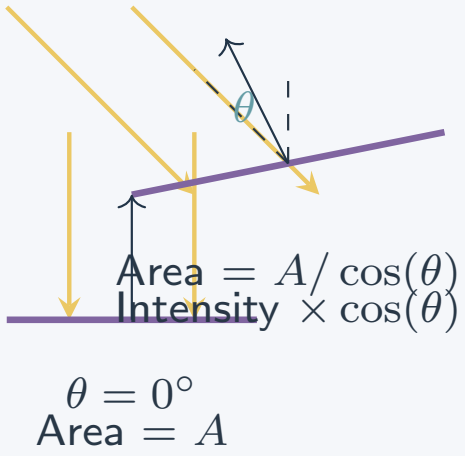
Why Cosine?

Consider light hitting a surface:

Energy per unit area depends on angle

When light hits at angle θ:

- Same light beam covers larger area
- Energy density decreases
- Area increases by factor $1/\cos(\theta)$
- Energy density decreases by factor $\cos(\theta)$



Lambert's Cosine Law - Geometric Derivation

Why Cosine?

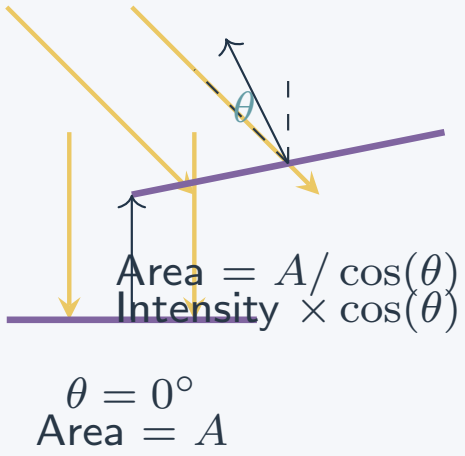
Consider light hitting a surface:

Energy per unit area depends on angle

When light hits at angle θ:

- Same light beam covers larger area
- Energy density decreases
- Area increases by factor $1/\cos(\theta)$
- Energy density decreases by factor $\cos(\theta)$

Mathematical relationship:



Diffuse Reflection - Mathematics

Lambert's Law Implementation

Diffuse component formula:

$$I_{\text{diffuse}} = k_d \cdot I_l \cdot (\mathbf{N} \cdot \mathbf{L}) \tag{5}$$

where:

- k_d = diffuse reflection coefficient (material color)
- I_l = intensity of the light source
- $\mathbf{N}$ = surface normal (unit vector)
- $\mathbf{L}$ = direction to light source (unit vector)

Diffuse Reflection - Mathematics

Lambert's Law Implementation

Diffuse component formula:

$$I_{\text{diffuse}} = k_d \cdot I_l \cdot (\mathbf{N} \cdot \mathbf{L}) \tag{5}$$

where:

- k_d = diffuse reflection coefficient (material color)
- I_l = intensity of the light source
- $\mathbf{N}$ = surface normal (unit vector)
- $\mathbf{L}$ = direction to light source (unit vector)

Important considerations:

$$\mathbf{N} \cdot \mathbf{L} = \cos(\theta) = \begin{cases} \text{positive} & \text{if light hits front of surface} \\ \text{negative} & \text{if light hits back of surface} \\ 0 & \text{if light grazes surface} \end{cases} \tag{6}$$

Diffuse Reflection - Mathematics

Lambert's Law Implementation

Diffuse component formula:

$$I_{\text{diffuse}} = k_d \cdot I_l \cdot (\mathbf{N} \cdot \mathbf{L}) \tag{5}$$

where:

- k_d = diffuse reflection coefficient (material color)
- I_l = intensity of the light source
- $\mathbf{N}$ = surface normal (unit vector)
- $\mathbf{L}$ = direction to light source (unit vector)

Important considerations:

$$\mathbf{N} \cdot \mathbf{L} = \cos(\theta) = \begin{cases} \text{positive} & \text{if light hits front of surface} \\ \text{negative} & \text{if light hits back of surface} \\ 0 & \text{if light grazes surface} \end{cases} \tag{6}$$

Diffuse Implementation Details

Step-by-Step Calculation

Input:

- Surface point $\mathbf{P}$
- Surface normal $\mathbf{N}$ (normalized)
- Light position $\mathbf{P}_{\text{light}}$
- Material diffuse coefficient k_d
- Light intensity I_l

Algorithm:

$$\mathbf{L} = \mathbf{P}_{\text{light}} - \mathbf{P} \tag{7}$$

$$\hat{\mathbf{L}} = \frac{\mathbf{L}}{|\mathbf{L}|} \tag{8}$$

$$\text{NdotL} = \max(0, \mathbf{N} \cdot \hat{\mathbf{L}}) \tag{9}$$

$$I_{\text{diffuse}} = k_d \cdot I_l \cdot \text{NdotL} \tag{10}$$

Diffuse Implementation Details

Step-by-Step Calculation

Input:

- Surface point $\mathbf{P}$
- Surface normal $\mathbf{N}$ (normalized)
- Light position $\mathbf{P}_{\text{light}}$
- Material diffuse coefficient k_d
- Light intensity I_l

Algorithm:

$$\mathbf{L} = \mathbf{P}_{\text{light}} - \mathbf{P} \tag{7}$$

$$\hat{\mathbf{L}} = \frac{\mathbf{L}}{|\mathbf{L}|} \tag{8}$$

$$\text{NdotL} = \max(0, \mathbf{N} \cdot \hat{\mathbf{L}}) \tag{9}$$

$$I_{\text{diffuse}} = k_d \cdot I_l \cdot \text{NdotL} \tag{10}$$

[Diffuse lighting at different angles]

Common Mistakes

- Forgetting to normalize $\mathbf{L}$
- Using world normal instead of surface normal
- Incorrect light direction calculation

Diffuse Implementation Details

Step-by-Step Calculation

Input:

- Surface point $\mathbf{P}$
- Surface normal $\mathbf{N}$ (normalized)
- Light position $\mathbf{P}_{\text{light}}$
- Material diffuse coefficient k_d
- Light intensity I_l

Algorithm:

$$\mathbf{L} = \mathbf{P}_{\text{light}} - \mathbf{P} \tag{7}$$

$$\hat{\mathbf{L}} = \frac{\mathbf{L}}{|\mathbf{L}|} \tag{8}$$

$$\text{NdotL} = \max(0, \mathbf{N} \cdot \hat{\mathbf{L}}) \tag{9}$$

$$I_{\text{diffuse}} = k_d \cdot I_l \cdot \text{NdotL} \tag{10}$$

[Diffuse lighting at different angles]

Common Mistakes

- Forgetting to normalize $\mathbf{L}$
- Using world normal instead of surface normal
- Incorrect light direction calculation

Specular Reflection - Introduction

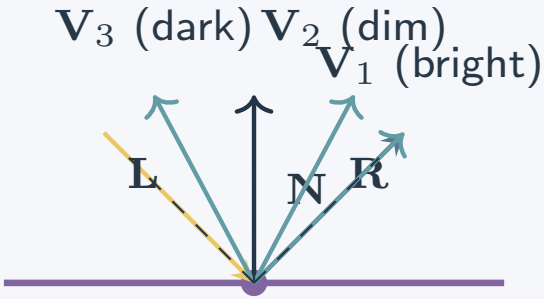
Shiny Surfaces

Examples:

- Mirrors
- Polished metals
- Glossy paint
- Plastic
- Water surface

Characteristics:

- View-dependent brightness
- Creates highlights
- Follows law of reflection
- Intensity depends on viewing angle



Specular Reflection - Introduction

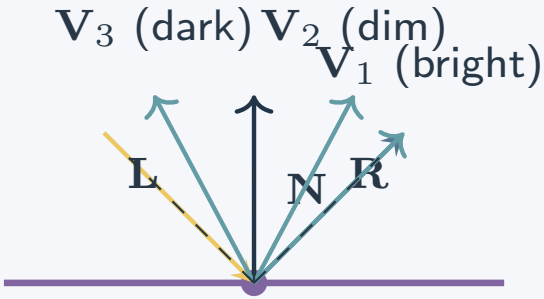
Shiny Surfaces

Examples:

- Mirrors
- Polished metals
- Glossy paint
- Plastic
- Water surface

Characteristics:

- View-dependent brightness
- Creates highlights
- Follows law of reflection
- Intensity depends on viewing angle



Perfect Reflection Theory

Law of Reflection

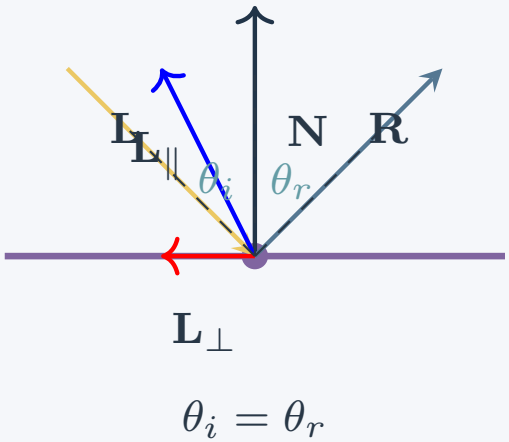
Physical principle: Angle of incidence equals angle of reflection

Vector formulation:

$$\mathbf{R} = 2(\mathbf{N} \cdot \mathbf{L})\mathbf{N} - \mathbf{L} \qquad (11)$$

where:

- $\mathbf{R}$ = reflection direction
- $\mathbf{N}$ = surface normal
- $\mathbf{L}$ = direction to light source



$\theta_i = \theta_r$

Perfect Reflection Theory

Law of Reflection

Physical principle: Angle of incidence equals angle of reflection

Vector formulation:

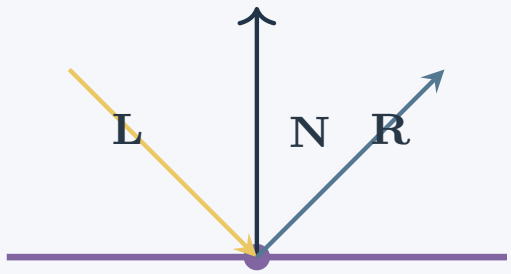
$$\mathbf{R} = 2(\mathbf{N} \cdot \mathbf{L})\mathbf{N} - \mathbf{L} \tag{11}$$

where:

- $\mathbf{R}$ = reflection direction
- $\mathbf{N}$ = surface normal
- $\mathbf{L}$ = direction to light source

Derivation: Decompose $\mathbf{L}$ into normal and tangential components

$$\mathbf{L}_{\parallel} = (\mathbf{N} \cdot \mathbf{L})\mathbf{N} \tag{12}$$



Perfect Reflection Theory

Law of Reflection

Physical principle: Angle of incidence equals angle of reflection

Vector formulation:

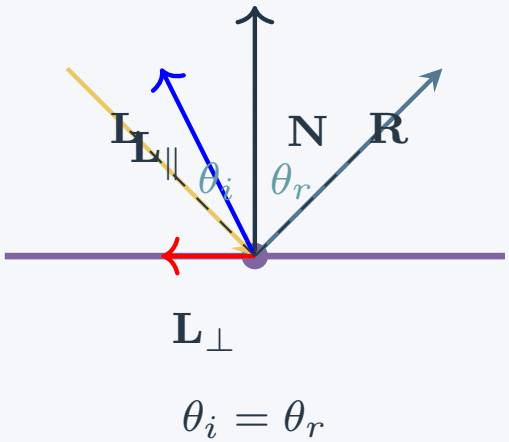
$$\mathbf{R} = 2(\mathbf{N} \cdot \mathbf{L})\mathbf{N} - \mathbf{L} \tag{11}$$

where:

- $\mathbf{R}$ = reflection direction
- $\mathbf{N}$ = surface normal
- $\mathbf{L}$ = direction to light source

Derivation: Decompose $\mathbf{L}$ into normal and tangential components

$$\mathbf{L}_{\parallel} = (\mathbf{N} \cdot \mathbf{L})\mathbf{N} \tag{12}$$



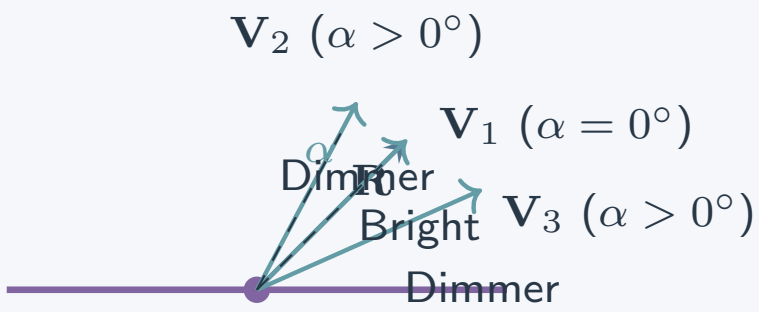
$\theta_i = \theta_r$

Phong Specular Model - Derivation

Phong's Insight

Problem: Perfect mirrors are rare in computer graphics

Most surfaces have some roughness that spreads the reflection

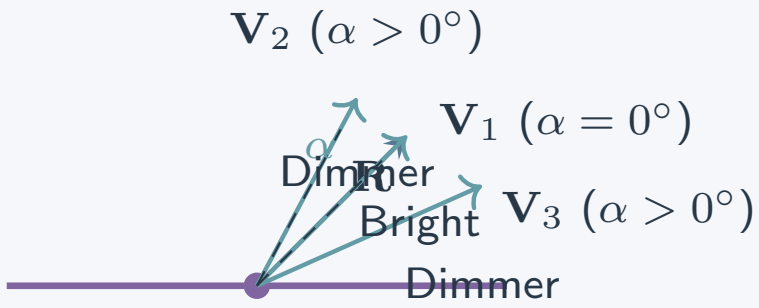


Phong Specular Model - Derivation

Phong's Insight

Problem: Perfect mirrors are rare in computer graphics
Most surfaces have some roughness that spreads the reflection

Solution: Model the spread using a power function
Brightness decreases as viewing direction deviates from perfect reflection



Phong Specular Model - Derivation

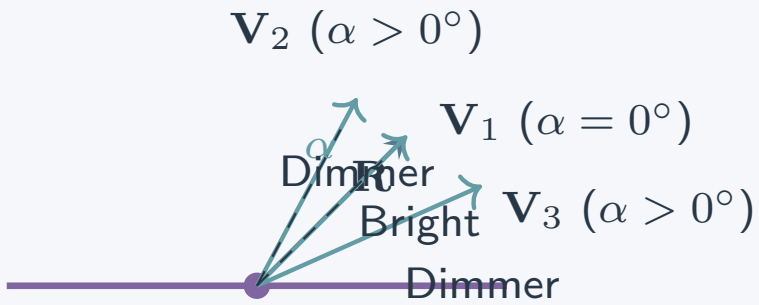
Phong's Insight

Problem: Perfect mirrors are rare in computer graphics
Most surfaces have some roughness that spreads the reflection

Solution: Model the spread using a power function
Brightness decreases as viewing direction deviates from perfect reflection

Key assumption:

- Perfect reflection at $\mathbf{V} = \mathbf{R}$
- Intensity decreases with angle between $\mathbf{V}$ and $\mathbf{R}$
- Use cosine raised to a power for



Specular Mathematics - Part 1: Reflection Vector

Calculating the Reflection Vector

Given:

- Light direction $\mathbf{L}$ (pointing toward light)
- Surface normal $\mathbf{N}$ (unit vector)

Step 1: Calculate reflection direction

$$\mathbf{R} = 2(\mathbf{N} \cdot \mathbf{L})\mathbf{N} - \mathbf{L} \tag{16}$$

Specular Mathematics - Part 1: Reflection Vector

Calculating the Reflection Vector

Given:

- Light direction $\mathbf{L}$ (pointing toward light)
- Surface normal $\mathbf{N}$ (unit vector)

Step 1: Calculate reflection direction

$$\mathbf{R} = 2(\mathbf{N} \cdot \mathbf{L})\mathbf{N} - \mathbf{L} \tag{16}$$

Step 2: Ensure $\mathbf{R}$ is normalized (should be automatic if $\mathbf{N}$ and $\mathbf{L}$ are unit vectors)

$$\hat{\mathbf{R}} = \frac{\mathbf{R}}{|\mathbf{R}|} \tag{17}$$

Specular Mathematics - Part 1: Reflection Vector

Calculating the Reflection Vector

Given:

- Light direction $\mathbf{L}$ (pointing toward light)
- Surface normal $\mathbf{N}$ (unit vector)

Step 1: Calculate reflection direction

$$\mathbf{R} = 2(\mathbf{N} \cdot \mathbf{L})\mathbf{N} - \mathbf{L} \tag{16}$$

Step 2: Ensure $\mathbf{R}$ is normalized (should be automatic if $\mathbf{N}$ and $\mathbf{L}$ are unit vectors)

$$\hat{\mathbf{R}} = \frac{\mathbf{R}}{|\mathbf{R}|} \tag{17}$$

Important: Only calculate specular if $\mathbf{N} \cdot \mathbf{L} > 0$ (light hits front of surface)

Specular Mathematics - Part 1: Reflection Vector

Calculating the Reflection Vector

Given:

- Light direction $\mathbf{L}$ (pointing toward light)
- Surface normal $\mathbf{N}$ (unit vector)

Step 1: Calculate reflection direction

$$\mathbf{R} = 2(\mathbf{N} \cdot \mathbf{L})\mathbf{N} - \mathbf{L} \tag{16}$$

Step 2: Ensure $\mathbf{R}$ is normalized (should be automatic if $\mathbf{N}$ and $\mathbf{L}$ are unit vectors)

$$\hat{\mathbf{R}} = \frac{\mathbf{R}}{|\mathbf{R}|} \tag{17}$$

Important: Only calculate specular if $\mathbf{N} \cdot \mathbf{L} > 0$ (light hits front of surface)

Specular Mathematics - Part 2: Intensity Calculation

Phong Specular Formula

Specular intensity:

$$I_{\text{specular}} = k_s \cdot I_l \cdot (\mathbf{R} \cdot \mathbf{V})^n \tag{18}$$

where:

- k_s = specular reflection coefficient (material shininess)
- I_l = light intensity
- $\mathbf{R}$ = reflection direction (unit vector)
- $\mathbf{V}$ = view direction (toward viewer, unit vector)
- n = shininess exponent (controls highlight size)

Specular Mathematics - Part 2: Intensity Calculation

Phong Specular Formula

Specular intensity:

$$I_{\text{specular}} = k_s \cdot I_l \cdot (\mathbf{R} \cdot \mathbf{V})^n \tag{18}$$

where:

- k_s = specular reflection coefficient (material shininess)
- I_l = light intensity
- $\mathbf{R}$ = reflection direction (unit vector)
- $\mathbf{V}$ = view direction (toward viewer, unit vector)
- n = shininess exponent (controls highlight size)

With clamping:

$$I_{\text{specular}} = k_s \cdot I_l \cdot \max(0, \mathbf{R} \cdot \mathbf{V})^n \tag{19}$$

Specular Mathematics - Part 2: Intensity Calculation

Phong Specular Formula

Specular intensity:

$$I_{\text{specular}} = k_s \cdot I_l \cdot (\mathbf{R} \cdot \mathbf{V})^n \tag{18}$$

where:

- k_s = specular reflection coefficient (material shininess)
- I_l = light intensity
- $\mathbf{R}$ = reflection direction (unit vector)
- $\mathbf{V}$ = view direction (toward viewer, unit vector)
- n = shininess exponent (controls highlight size)

With clamping:

$$I_{\text{specular}} = k_s \cdot I_l \cdot \max(0, \mathbf{R} \cdot \mathbf{V})^n \tag{19}$$

Complete condition:

Shininess Parameter (n) - Effect on Highlights

Shininess Exponent

Mathematical effect:

$$(\cos(\alpha))^n \qquad (21)$$

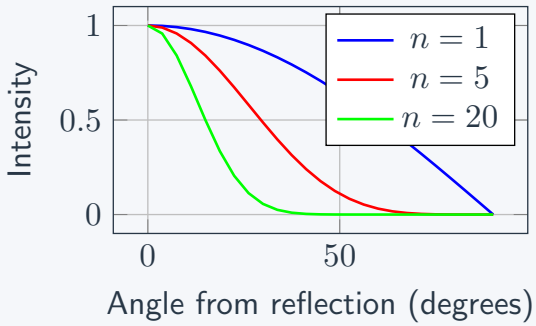
As n increases:

- Highlight becomes smaller
- Highlight becomes sharper
- Material appears shinier

Typical values:

- $n = 1$: Very wide, soft highlight
- $n = 10$: Plastic-like

[Shininess parameter comparison]



Complete Phong Equation Assembly

Putting It All Together

Complete Phong illumination model:

$$I = I_{\text{ambient}} + I_{\text{diffuse}} + I_{\text{specular}} \tag{22}$$

Complete Phong Equation Assembly

Putting It All Together

Complete Phong illumination model:

$$I = I_{\text{ambient}} + I_{\text{diffuse}} + I_{\text{specular}} \tag{22}$$

Expanded form:

$$I = k_a I_a + k_d I_l \max(0, \mathbf{N} \cdot \mathbf{L}) + k_s I_l \max(0, \mathbf{R} \cdot \mathbf{V})^n \tag{23}$$

Complete Phong Equation Assembly

Putting It All Together

Complete Phong illumination model:

$$I = I_{\text{ambient}} + I_{\text{diffuse}} + I_{\text{specular}} \tag{22}$$

Expanded form:

$$I = k_a I_a + k_d I_l \max(0, \mathbf{N} \cdot \mathbf{L}) + k_s I_l \max(0, \mathbf{R} \cdot \mathbf{V})^n \tag{23}$$

For multiple lights:

$$I = k_a I_a + \sum_{i=1}^{n} [k_d I_{l_i} \max(0, \mathbf{N} \cdot \mathbf{L}_i) + k_s I_{l_i} \max(0, \mathbf{R}_i \cdot \mathbf{V})^n] \tag{24}$$

Complete Phong Equation Assembly

Putting It All Together

Complete Phong illumination model:

$$I = I_{\text{ambient}} + I_{\text{diffuse}} + I_{\text{specular}} \tag{22}$$

Expanded form:

$$I = k_a I_a + k_d I_l \max(0, \mathbf{N} \cdot \mathbf{L}) + k_s I_l \max(0, \mathbf{R} \cdot \mathbf{V})^n \tag{23}$$

For multiple lights:

$$I = k_a I_a + \sum_{i=1}^{n} [k_d I_{l_i} \max(0, \mathbf{N} \cdot \mathbf{L}_i) + k_s I_{l_i} \max(0, \mathbf{R}_i \cdot \mathbf{V})^n] \tag{24}$$

With RGB colors:

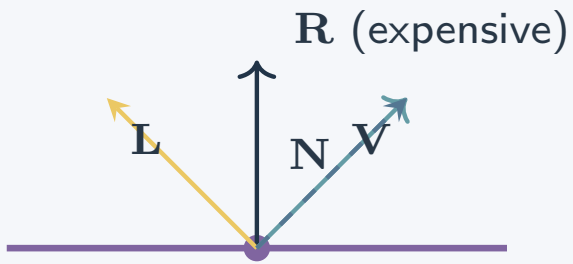
$$\mathbf{I} = k_a \mathbf{I_a} + \sum^{n} \mathbf{I_{l_i}} \odot [k_d \max(0, \mathbf{N} \cdot \mathbf{L}_i) + k_s \max(0, \mathbf{R}_i \cdot \mathbf{V})^n]$$

Phong Model Limitations & Optimization

Phong Model Issues

Computational cost:

- Reflection vector calculation expensive
- Requires 2 dot products + vector operations
- Per-light calculation needed



Expensive calculation:
$\mathbf{R} = 2(\mathbf{N} \cdot \mathbf{L})\mathbf{N} - \mathbf{L}$

Phong Model Limitations & Optimization

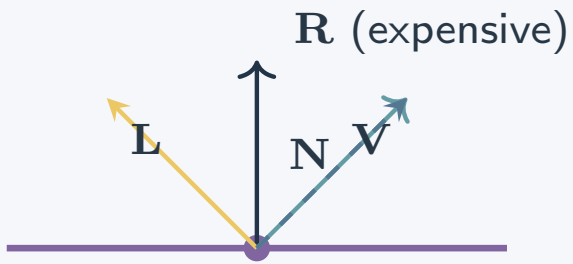
Phong Model Issues

Computational cost:

- Reflection vector calculation expensive
- Requires 2 dot products + vector operations
- Per-light calculation needed

Visual artifacts:

- Highlights can "pop" on/off abruptly
- Non-physical behavior at grazing angles
- Issues with interpolation across polygons



Expensive calculation:
$\mathbf{R} = 2(\mathbf{N} \cdot \mathbf{L})\mathbf{N} - \mathbf{L}$

Phong Model Limitations & Optimization

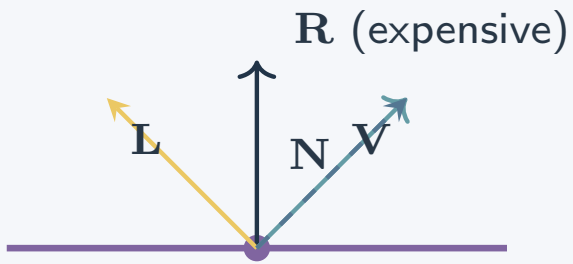
Phong Model Issues

Computational cost:

- Reflection vector calculation expensive
- Requires 2 dot products + vector operations
- Per-light calculation needed

Visual artifacts:

- Highlights can "pop" on/off abruptly
- Non-physical behavior at grazing angles
- Issues with interpolation across polygons



Phong Model Limitations & Optimization

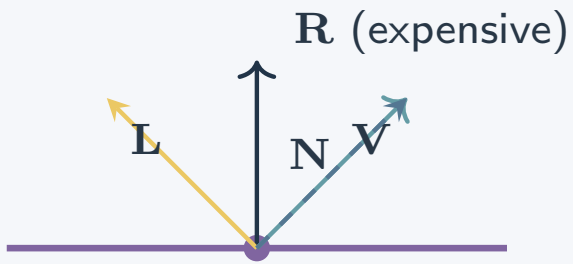
Phong Model Issues

Computational cost:

- Reflection vector calculation expensive
- Requires 2 dot products + vector operations
- Per-light calculation needed

Visual artifacts:

- Highlights can "pop" on/off abruptly
- Non-physical behavior at grazing angles
- Issues with interpolation across polygons



Expensive calculation:
$\mathbf{R} = 2(\mathbf{N} \cdot \mathbf{L})\mathbf{N} - \mathbf{L}$

Phong Model Limitations & Optimization

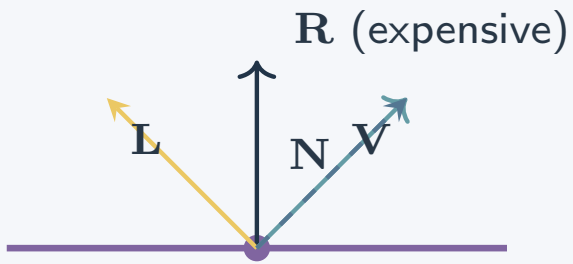
Phong Model Issues

Computational cost:

- Reflection vector calculation expensive
- Requires 2 dot products + vector operations
- Per-light calculation needed

Visual artifacts:

- Highlights can "pop" on/off abruptly
- Non-physical behavior at grazing angles
- Issues with interpolation across polygons



Expensive calculation:
$\mathbf{R} = 2(\mathbf{N} \cdot \mathbf{L})\mathbf{N} - \mathbf{L}$

Blinn-Phong Optimization

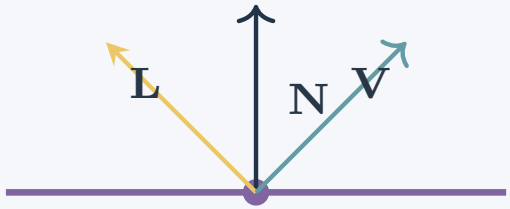
Blinn's Innovation (1977)

Key insight: Instead of using reflection vector $\mathbf{R}$, use halfway vector $\mathbf{H}$

Halfway vector: Bisects angle between light and view directions

Advantages:

- Cheaper to calculate
- More stable numerically
- Better hardware optimization
- Smoother interpolation



Blinn-Phong Optimization

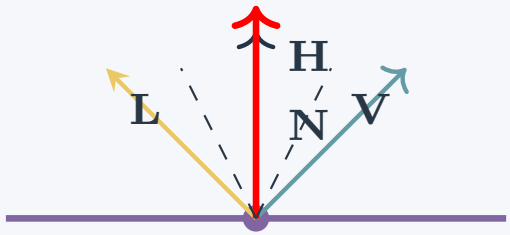
Blinn's Innovation (1977)

Key insight: Instead of using reflection vector $\mathbf{R}$, use halfway vector $\mathbf{H}$

Halfway vector: Bisects angle between light and view directions

Advantages:

- Cheaper to calculate
- More stable numerically
- Better hardware optimization
- Smoother interpolation



Halfway vector bisects $\mathbf{L}$ and $\mathbf{V}$

Blinn-Phong Optimization

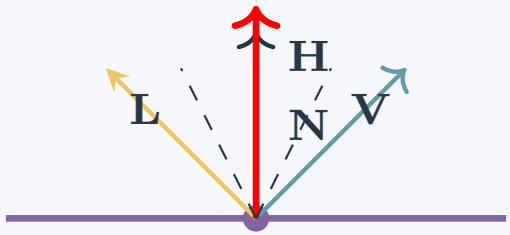
Blinn's Innovation (1977)

Key insight: Instead of using reflection vector $\mathbf{R}$, use halfway vector $\mathbf{H}$

Halfway vector: Bisects angle between light and view directions

Advantages:

- Cheaper to calculate
- More stable numerically
- Better hardware optimization
- Smoother interpolation



Halfway vector bisects $\mathbf{L}$ and $\mathbf{V}$

Half-Vector Mathematics

Halfway Vector Calculation

Definition: Vector that bisects the angle between light and view directions

Simple calculation:

$$\mathbf{H} = \frac{\mathbf{L} + \mathbf{V}}{|\mathbf{L} + \mathbf{V}|} \tag{26}$$

where $\mathbf{L}$ and $\mathbf{V}$ are unit vectors

Half-Vector Mathematics

Halfway Vector Calculation

Definition: Vector that bisects the angle between light and view directions

Simple calculation:

$$\mathbf{H} = \frac{\mathbf{L} + \mathbf{V}}{|\mathbf{L} + \mathbf{V}|} \tag{26}$$

where $\mathbf{L}$ and $\mathbf{V}$ are unit vectors

Blinn-Phong specular formula:

$$I_{\text{specular}} = k_s \cdot I_l \cdot \max(0, \mathbf{N} \cdot \mathbf{H})^{n'} \tag{27}$$

Half-Vector Mathematics

Halfway Vector Calculation

Definition: Vector that bisects the angle between light and view directions

Simple calculation:

$$\mathbf{H} = \frac{\mathbf{L} + \mathbf{V}}{|\mathbf{L} + \mathbf{V}|} \tag{26}$$

where $\mathbf{L}$ and $\mathbf{V}$ are unit vectors

Blinn-Phong specular formula:

$$I_{\text{specular}} = k_s \cdot I_l \cdot \max(0, \mathbf{N} \cdot \mathbf{H})^{n'} \tag{27}$$

Note: Shininess exponent n' in Blinn-Phong is typically 2-4 times larger than Phong's n for similar visual result

Relationship: $n' \approx 2n$ to $4n$ (depends on material)

Half-Vector Mathematics

Halfway Vector Calculation

Definition: Vector that bisects the angle between light and view directions

Simple calculation:

$$\mathbf{H} = \frac{\mathbf{L} + \mathbf{V}}{|\mathbf{L} + \mathbf{V}|} \tag{26}$$

where $\mathbf{L}$ and $\mathbf{V}$ are unit vectors

Blinn-Phong specular formula:

$$I_{\text{specular}} = k_s \cdot I_l \cdot \max(0, \mathbf{N} \cdot \mathbf{H})^{n'} \tag{27}$$

Note: Shininess exponent n' in Blinn-Phong is typically 2-4 times larger than Phong's n for similar visual result

Relationship: $n' \approx 2n$ to $4n$ (depends on material)

Blinn-Phong vs Phong Comparison

Performance Comparison

Phong reflection calculation:

$$\mathbf{R} = 2(\mathbf{N} \cdot \mathbf{L})\mathbf{N} - \mathbf{L} \quad (28)$$

$$I = (\mathbf{R} \cdot \mathbf{V})^n \quad (29)$$

Operations: 1 dot product, 1 scalar multiply, 2 vector operations, 1 power

[Performance comparison chart]

Visual Differences

Blinn-Phong highlights:

- Slightly larger
- Softer falloff
- More stable interpolation

Often preferred for real-

Blinn-Phong vs Phong Comparison

Performance Comparison

Phong reflection calculation:

$$\mathbf{R} = 2(\mathbf{N} \cdot \mathbf{L})\mathbf{N} - \mathbf{L} \quad (28)$$

$$I = (\mathbf{R} \cdot \mathbf{V})^n \quad (29)$$

Operations: 1 dot product, 1 scalar multiply, 2 vector operations, 1 power

Blinn-Phong calculation:

$$\mathbf{H} = \text{normalize}(\mathbf{L} + \mathbf{V}) \quad (30)$$

$$I = (\mathbf{N} \cdot \mathbf{H})^{n'} \quad (31)$$

[Performance comparison chart]

Visual Differences

Blinn-Phong highlights:

- Slightly larger
- Softer falloff
- More stable interpolation

Often preferred for real-

Blinn-Phong vs Phong Comparison

Performance Comparison

Phong reflection calculation:

$$\mathbf{R} = 2(\mathbf{N} \cdot \mathbf{L})\mathbf{N} - \mathbf{L} \quad (28)$$

$$I = (\mathbf{R} \cdot \mathbf{V})^n \quad (29)$$

Operations: 1 dot product, 1 scalar multiply, 2 vector operations, 1 power

Blinn-Phong calculation:

$$\mathbf{H} = \text{normalize}(\mathbf{L} + \mathbf{V}) \quad (30)$$

$$I = (\mathbf{N} \cdot \mathbf{H})^{n'} \quad (31)$$

[Performance comparison chart]

Visual Differences

Blinn-Phong highlights:

- Slightly larger
- Softer falloff
- More stable interpolation

Often preferred for real-

Blinn-Phong vs Phong Comparison

Performance Comparison

Phong reflection calculation:

$$\mathbf{R} = 2(\mathbf{N} \cdot \mathbf{L})\mathbf{N} - \mathbf{L} \quad (28)$$

$$I = (\mathbf{R} \cdot \mathbf{V})^n \quad (29)$$

Operations: 1 dot product, 1 scalar multiply, 2 vector operations, 1 power

Blinn-Phong calculation:

$$\mathbf{H} = \text{normalize}(\mathbf{L} + \mathbf{V}) \quad (30)$$

$$I = (\mathbf{N} \cdot \mathbf{H})^{n'} \quad (31)$$

[Performance comparison chart]

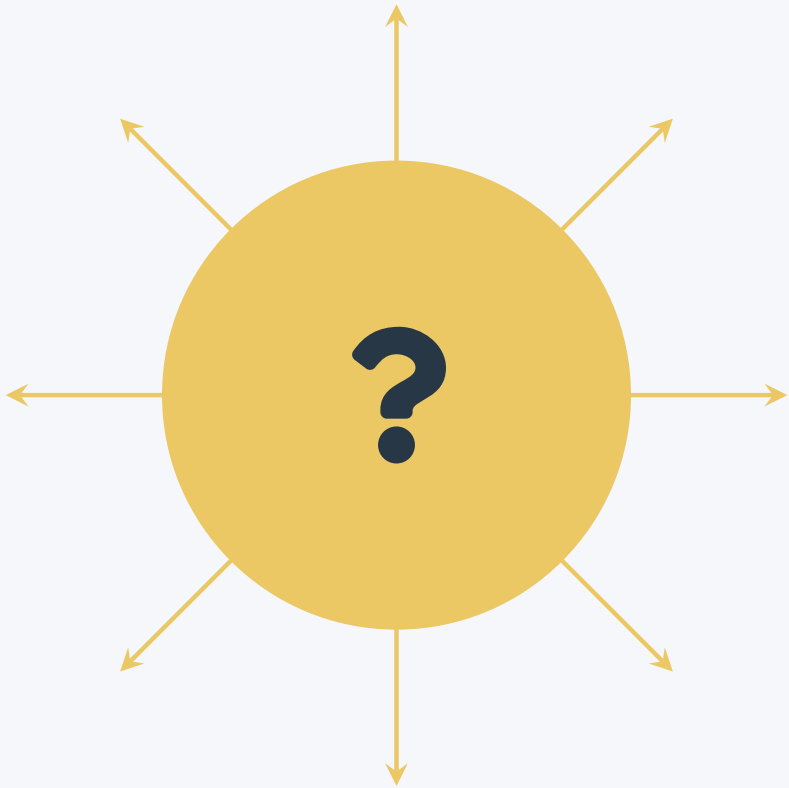
Visual Differences

Blinn-Phong highlights:

- Slightly larger
- Softer falloff
- More stable interpolation

Often preferred for real-

Questions?



References & Further Reading

- Peter Shirley and Steve Marschner et al. *Fundamentals of Computer Graphics (4th Edition)*. CRC Press, 2016.
  Available as PDF
- Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically Based Rendering: From Theory to Implementation (4th Edition)*. Morgan Kaufmann, 2023.
  Availabe online
- Peter Shirley. *Ray Tracing in One Weekend*. Self-published, 2016–2020.
  Project Website
- MIT OpenCourseWare: 6.837 Computer Graphics.
  ocw.mit.edu/6-837
- Scratchapixel: Learn Computer Graphics Programming.
  scratchapixel.com