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

Light Sources

- Point Light
- Directional Lights
- Spot Lights
- Ambient Light

Normal Vectors

The Phong Illumination Model

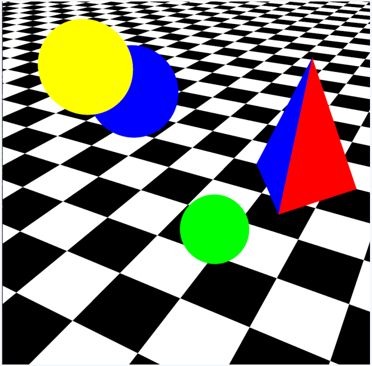
- Specular
- Blinn-Phong Model

Texturing

Motivation

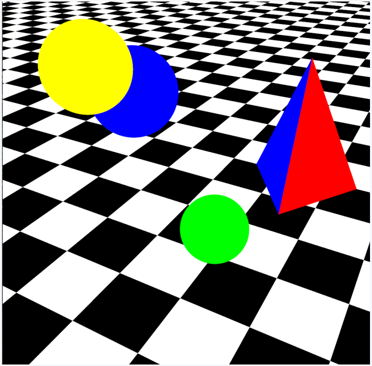
Why Lighting Matters

Why Lighting Matters

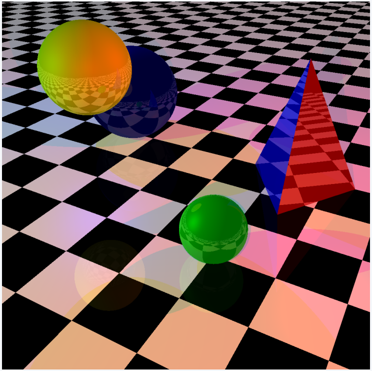


Without Lighting and Shading

Why Lighting Matters



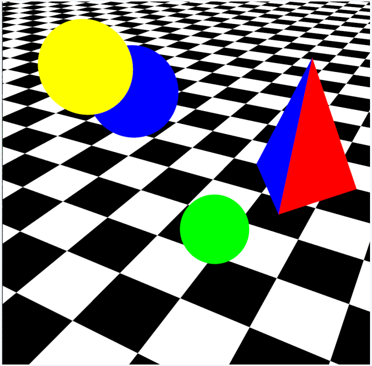
Without Lighting and Shading



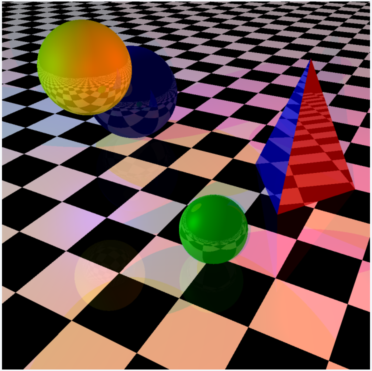
With Lighting and Shading

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

Why Lighting Matters



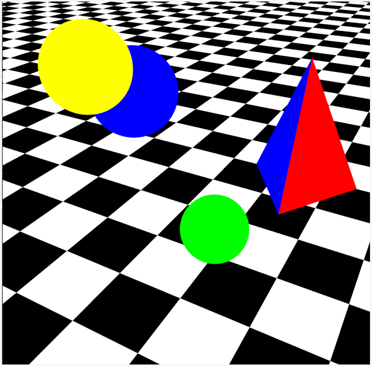
Without Lighting and Shading



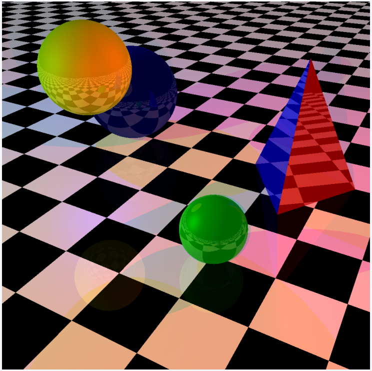
With Lighting and Shading

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

Why Lighting Matters



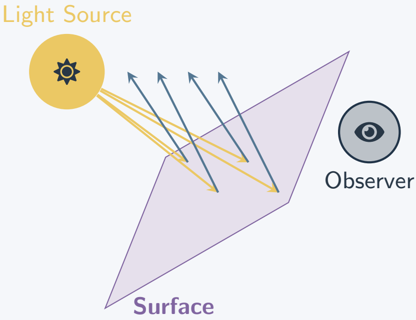
Without Lighting and Shading



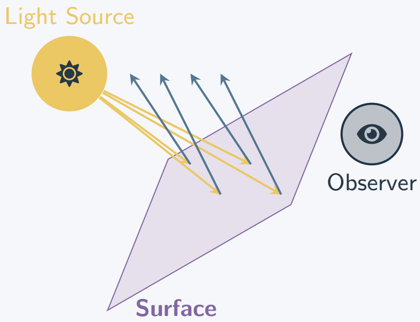
With Lighting and Shading

- **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:

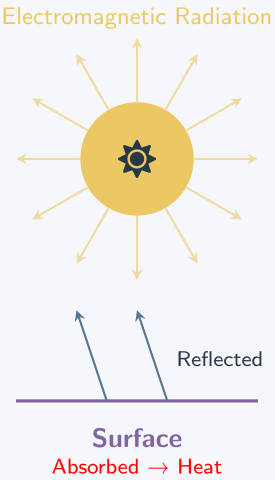
- Millions of photons per surface point
- Complex wave interactions
- Multiple scattering events
- Continuous spectrum

Computer Graphics:

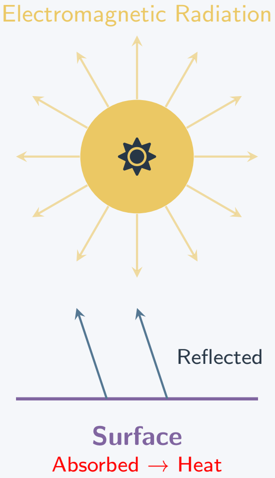
- Discrete RGB values
- Simplified mathematical models
- Local illumination approximations
- Real-time constraints

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

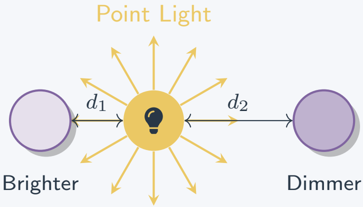
Surface interactions:

- **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



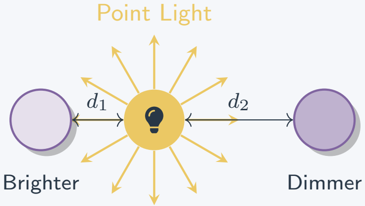
Point Light Sources - Introduction

Point Light Characteristics

Light emanating from a single point in all directions

Real-world examples:

- Light bulbs, LEDs
- Candles



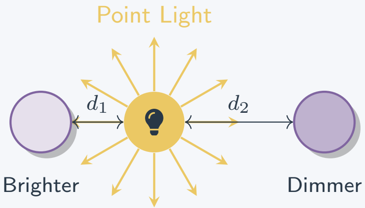
Point Light Sources - Introduction

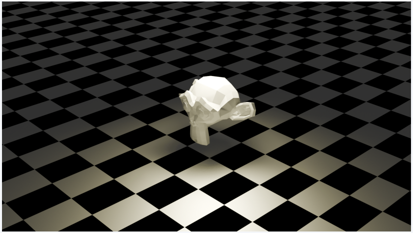
Point Light Characteristics

Light emanating from a single point in all directions

Real-world examples:

- Light bulbs, LEDs
- Candles





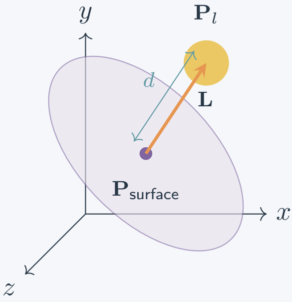
Point Light ft. Suzanne the monkey

Point Light Mathematics

Point Light Parameters

Position: $\mathbf{P}_l = (x_l, y_l, z_l)$

Intensity: $\mathbf{I}_l$ (brightness; RGB channels)



Point Light Mathematics

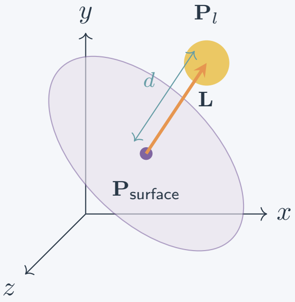
Point Light Parameters

Position: $\mathbf{P}_l = (x_l, y_l, z_l)$
Intensity: $\mathbf{I}_l$ (brightness; RGB channels)

Light direction to surface point:

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

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



Point Light Mathematics

Point Light Parameters

Position: $\mathbf{P}_l = (x_l, y_l, z_l)$
Intensity: $\mathbf{I}_l$ (brightness; RGB channels)

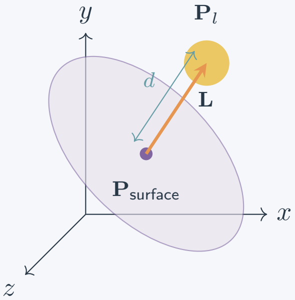
Light direction to surface point:

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

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

Distance:

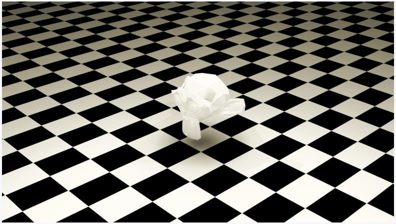
$$d = |\mathbf{P}_l - \mathbf{P}_{\text{surface}}|$$



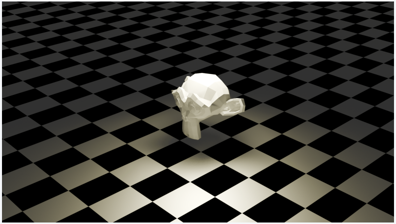
Attenuation

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

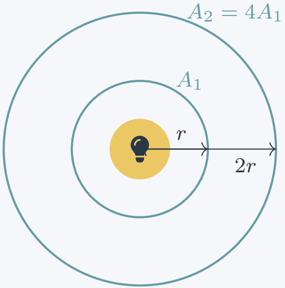


With Attenuation

Inverse Square Law - The Physics

$1/r^2$ Attenuation

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

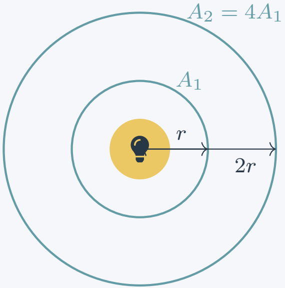


Same energy $\rightarrow$ $4\times$ area $\rightarrow$ $\frac{1}{4}$ intensity

Inverse Square Law - The Physics

$1/r^2$ Attenuation

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

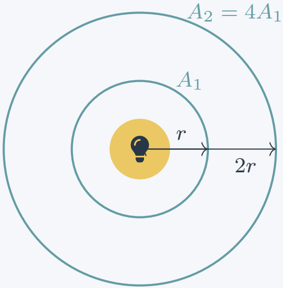


Same energy $\rightarrow$ 4$\times$ area $\rightarrow$ $\frac{1}{4}$ intensity

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.

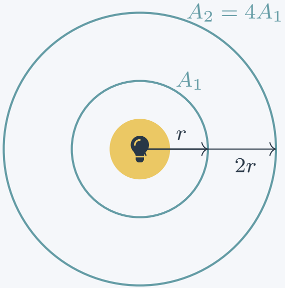


Same energy $\rightarrow$ 4$\times$ area $\rightarrow$ $\frac{1}{4}$ intensity

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:** $I \propto \frac{1}{r^2}$

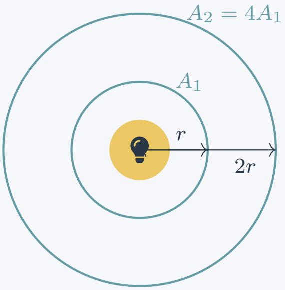


Same energy $\rightarrow$ 4$\times$ area $\rightarrow$ $\frac{1}{4}$ intensity

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

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

Point Light Implementation

Point Light Function Structure

Input parameters:

- Light position: $\mathbf{P}_l$
- Light intensity: $\mathbf{I}_l$
- Surface point: $\mathbf{P}_{\text{surface}}$

Point Light Implementation

Point Light Function Structure

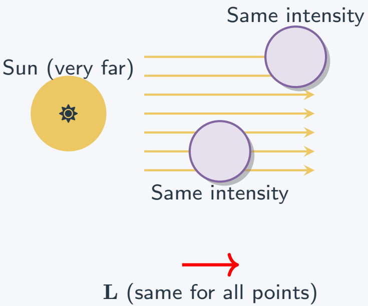
Calculation steps:

$$
\begin{aligned}
\mathbf{L} &= \mathbf{P}_l - \mathbf{P}_{\text{surface}} \\
d &= |\mathbf{L}| \\
\hat{\mathbf{L}} &= \mathbf{L}/d \\
I_{\text{final}} &= \frac{\mathbf{I}_l}{\epsilon + d^2}
\end{aligned}
$$

Directional Lights - The Sun Model

Directional Light Concept

Light source at infinite distance, so rays are effectively parallel



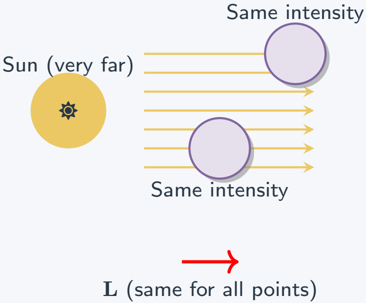
Directional Lights - The Sun Model

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

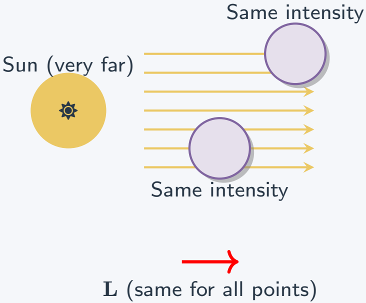


Directional Lights - The Sun Model

Directional Light Concept

Light source at infinite distance, so rays are effectively parallel

Perfect for: Sun, moon, distant lights



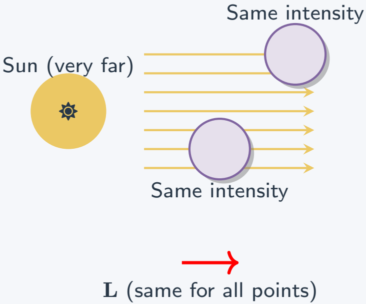
Directional Lights - The Sun Model

Directional Light Concept

Light source at infinite distance, so rays are effectively parallel

Perfect for: Sun, moon, distant lights

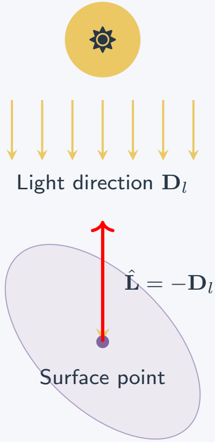
Directional Light ft. Suzanne



Directional Light Mathematics

Directional Light Parameters

Direction: $\mathbf{D}_l = (x, y, z)$
Intensity: $\mathbf{I}_l$ (constant)



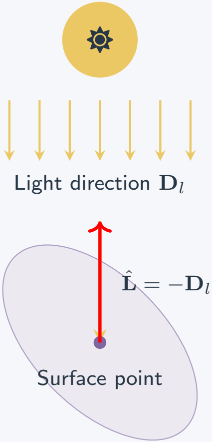
Directional Light Mathematics

Directional Light Parameters

Direction: $\mathbf{D}_l = (x, y, z)$
Intensity: $\mathbf{I}_l$ (constant) **Light direction:**

$$\hat{\mathbf{L}} = -\mathbf{D}_l$$

Note: Assuming $\mathbf{D}_l$ is normalized



Directional Light Mathematics

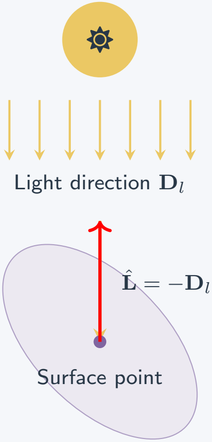
Directional Light Parameters

Direction: $\mathbf{D}_l = (x, y, z)$
Intensity: $\mathbf{I}_l$ (constant) **Light direction:**

$$\hat{\mathbf{L}} = -\mathbf{D}_l$$

Intensity at any point:

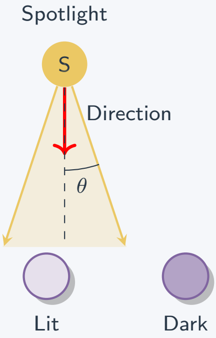
$$I_{\text{final}} = \mathbf{I}_l \quad \text{(no attenuation)}$$



Spot Lights - Introduction

Spot Light Characteristics

Light emanating from a point within a cone



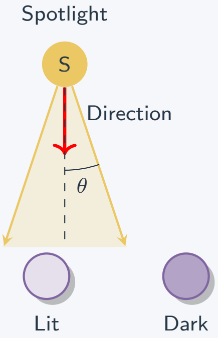
Spot Lights - Introduction

Spot Light Characteristics

Light emanating from a point within a cone

Real-world examples:

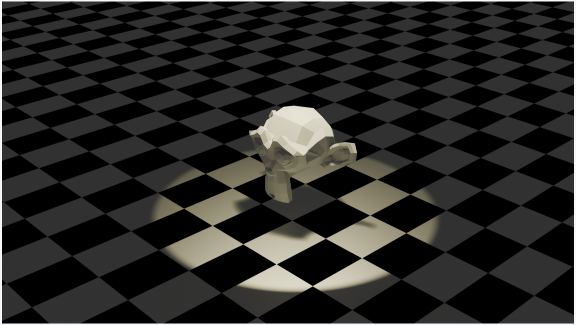
- Flashlights, headlights
- Stage spotlights
- Desk lamps



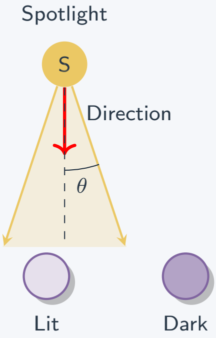
Spot Lights - Introduction

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

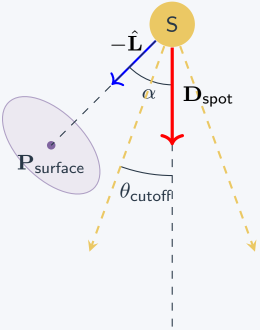
Position: $\mathbf{P}_l$

Intensity: $\mathbf{I}_l$

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

Cone angle: θ_{cutoff}

Falloff exponent: e



Spot Light Mathematics - Cone Calculation

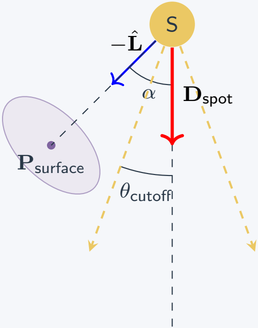
Spot Light Parameters

Position: $\mathbf{P}_l$
Intensity: $\mathbf{I}_l$
Direction: $\mathbf{D}_{\text{spot}}$
Cone angle: θ_{cutoff}
Falloff exponent: e

Step 1 - Calculate angle to surface:

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

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



Spot Light Mathematics - Cone Calculation

Spot Light Parameters

Position: $\mathbf{P}_l$

Intensity: $\mathbf{I}_l$

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

Cone angle: θ_{cutoff}

Falloff exponent: e

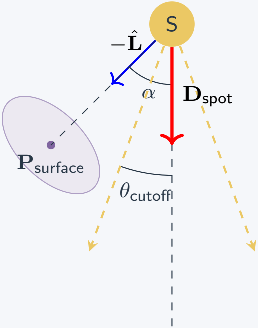
Step 1 - Calculate angle to surface:

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

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

Step 2 - Check if inside cone:

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



Spot Light Attenuation

Complete Spot Light Formula

Angular attenuation:

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

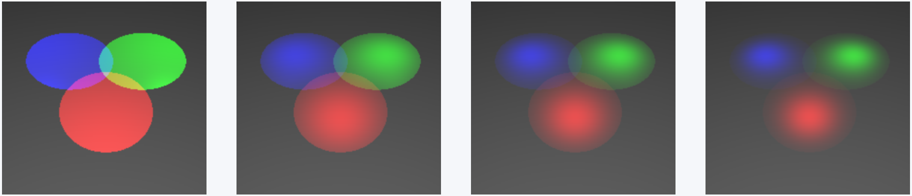
Spot Light Attenuation

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:

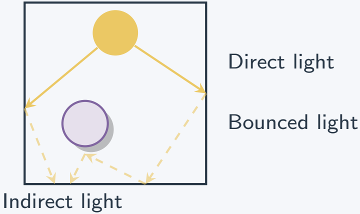
$$I_{\text{final}} = \begin{cases} \mathbf{I}_l \cdot (\cos(\alpha))^e \cdot \frac{1}{\epsilon + d^2} & \text{if } \cos(\alpha) > \cos(\theta_{\text{cutoff}}) \\ 0 & \text{otherwise} \end{cases}$$

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

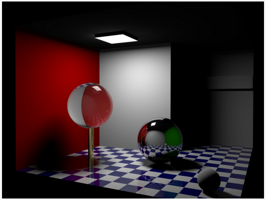


Ambient Light - Global Illumination Approximation

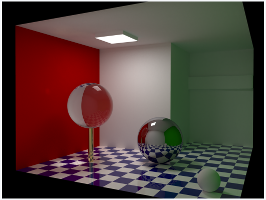
Direct Lighting Problem

Real scenes have indirect lighting

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



Only direct light



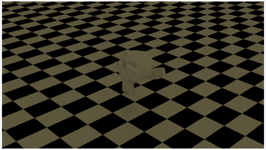
Global illumination

Ambient Light - Global Illumination Approximation

Solution: Ambient Light

Add constant ambient term

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

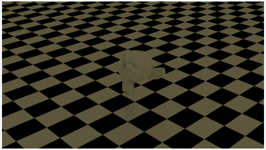


Ambient light ft. Suzanne the monkey

Ambient Light - Global Illumination Approximation

Ambient Light Parameters

Ambient intensity: I_l (constant)



Ambient light ft. Suzanne the monkey

Normal Vectors

Normal Vectors

Surface Normal Properties

Definition: Vector perpendicular to surface at a point

Normal Vectors

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 Vectors

Surface Normal Properties

Definition: Vector perpendicular to surface at a point

For triangles: Cross product of sides

$$\mathbf{N} = \frac{(\mathbf{b} - \mathbf{a}) \times (\mathbf{c} - \mathbf{a})}{|(\mathbf{b} - \mathbf{a}) \times (\mathbf{c} - \mathbf{a})|}$$

Normal Vectors

Surface Normal Properties

Definition: Vector perpendicular to surface at a point

For triangles: Cross product of sides

$$\mathbf{N} = \frac{(\mathbf{b} - \mathbf{a}) \times (\mathbf{c} - \mathbf{a})}{|(\mathbf{b} - \mathbf{a}) \times (\mathbf{c} - \mathbf{a})|}$$

For spheres: Normal at point $\mathbf{P}$ is radial assuming center at origin

$$\mathbf{N} = \frac{\mathbf{P}}{|\mathbf{P}|}$$

Normal Vectors

Surface Normal Properties

Definition: Vector perpendicular to surface at a point

For triangles: Cross product of sides

$$\mathbf{N} = \frac{(\mathbf{b} - \mathbf{a}) \times (\mathbf{c} - \mathbf{a})}{|(\mathbf{b} - \mathbf{a}) \times (\mathbf{c} - \mathbf{a})|}$$

For spheres: Normal at point $\mathbf{P}$ is radial assuming center at origin

$$\mathbf{N} = \frac{\mathbf{P}}{|\mathbf{P}|}$$

For planes: Normal defined by plane equation $Ax + By + Cz + D = 0$

$$\mathbf{N} = \frac{(A, B, C)}{\sqrt{A^2 + B^2 + C^2}}$$

Normal Vectors

Surface Normal Properties

Definition: Vector perpendicular to surface at a point

For triangles: Cross product of sides

$$\mathbf{N} = \frac{(\mathbf{b} - \mathbf{a}) \times (\mathbf{c} - \mathbf{a})}{|(\mathbf{b} - \mathbf{a}) \times (\mathbf{c} - \mathbf{a})|}$$

For spheres: Normal at point $\mathbf{P}$ is radial assuming center at origin

$$\mathbf{N} = \frac{\mathbf{P}}{|\mathbf{P}|}$$

For planes: Normal defined by plane equation $Ax + By + Cz + D = 0$

$$\mathbf{N} = \frac{(A, B, C)}{\sqrt{A^2 + B^2 + C^2}}$$

For arbitrary surfaces: Use gradient of implicit function

$$\mathbf{N} = \frac{\nabla f(x, y, z)}{|\nabla f(x, y, z)|}$$

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.

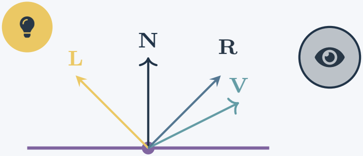
Bui Tuong Phong (1942-1975)

Phong Model Overview

Phong Formula

Break lighting into three components that can be computed independently

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

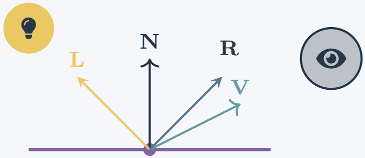


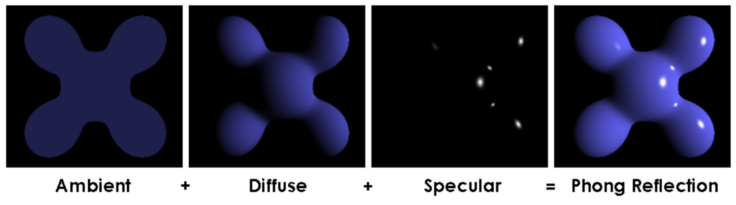
Phong Model Overview

Phong Formula

Break lighting into three components that can be computed independently

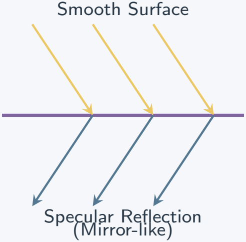
$$I = I_{\text{ambient}} + I_{\text{diffuse}} + I_{\text{specular}}$$
$$= k_a I_a + k_d \mathbf{I}_l (\mathbf{N} \cdot \mathbf{L}) + k_s \mathbf{I}_l (\mathbf{R} \cdot \mathbf{V})^n$$





Phong model components

Types of Reflection

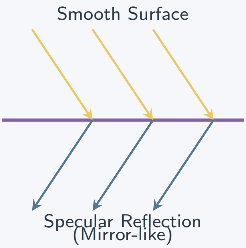


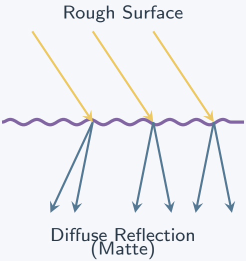
Reflection Types

Specular Reflection:

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

Types of Reflection





Reflection Types

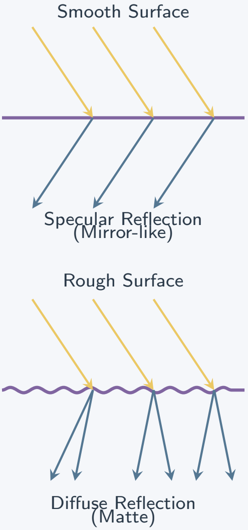
Specular Reflection:

- 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

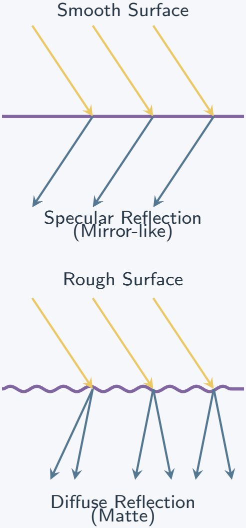
Diffuse Reflection:

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

Ambient Reflection:

- Approximation of indirect light
- View-independent

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

Ambient Reflection:

- Approximation of indirect light
- View-independent

Real surfaces: Combination of three types

Ambient Reflection - Mathematics

Ambient Component Formula

Simplest lighting component:

$$I_{\text{ambient}} = \mathbf{k}_a \odot \mathbf{I}_a$$

- $\mathbf{k}_a$ = ambient reflection coefficient of the material
- $\mathbf{I}_a$ = intensity of ambient light in the scene

Ambient Reflection - Mathematics

Ambient Component Formula

Simplest lighting component:

$$I_{\text{ambient}} = \mathbf{k}_a \odot \mathbf{I}_a$$

- $\mathbf{k}_a$ = ambient reflection coefficient of the material
- $\mathbf{I}_a$ = intensity of ambient light in the scene

Key properties:

- Independent of viewer position, light direction, or surface normal
- $\mathbf{I}_a$ is a constant for the scene or a combination of the light sources in the scene

Ambient Reflection - Mathematics

Ambient Component Formula

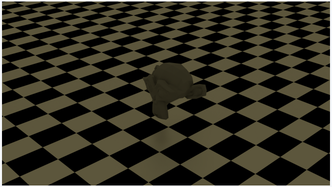
Simplest lighting component:

$$I_{\text{ambient}} = \mathbf{k}_a \odot \mathbf{I}_a$$

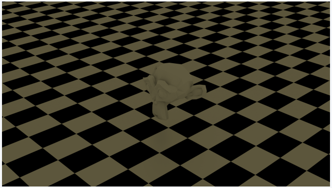
- $\mathbf{k}_a$ = ambient reflection coefficient of the material
- $\mathbf{I}_a$ = intensity of ambient light in the scene

Key properties:

- Independent of viewer position, light direction, or surface normal
- $\mathbf{I}_a$ is a constant for the scene or a combination of the light sources in the scene

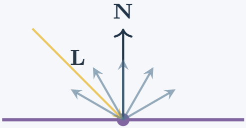


Low $\mathbf{k}_a$



High $\mathbf{k}_a$

Diffuse Reflection - Introduction



Equal brightness
all directions

Lambertian Surfaces

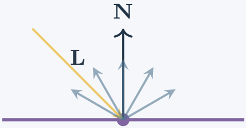
Examples:

Paper, wood, fabric

Characteristics:

- Surface appears equally bright from all viewing angles
- Light scattered uniformly in all directions
- Brightness depends only on angle of incident light

Diffuse Reflection - Introduction



Equal brightness
all directions

Lambertian Surfaces

Examples:

Paper, wood, fabric

Characteristics:

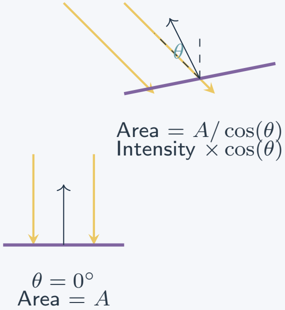
- Surface appears equally bright from all viewing angles
- Light scattered uniformly in all directions
- Brightness depends only on angle of incident light

Examples of diffuse surfaces: wood and fabric

Lambert's Cosine Law - Intuition

Why Cosine?

Consider light hitting a surface:



Lambert's Cosine Law - Intuition

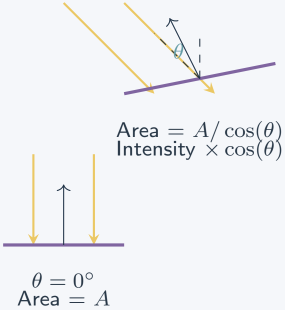
Why Cosine?

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 - Intuition

Why Cosine?

Consider light hitting a surface:

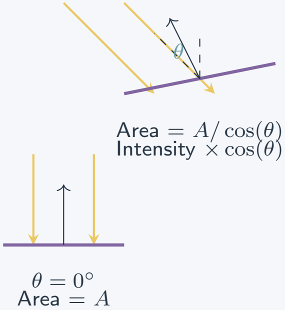
Energy per unit area depends on angle

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:

Effective intensity $\propto \cos(\theta) = \mathbf{N} \cdot \mathbf{L}$



Diffuse Reflection - Mathematics

Lambert's Law Implementation

Diffuse component formula:

$$I_{\text{diffuse}} = \mathbf{k}_d \odot \mathbf{I}_l(\mathbf{N} \cdot \mathbf{L})$$

where:

- $\mathbf{k}_d$ = diffuse reflection coefficient (material color)
- $\mathbf{I}_l$ = intensity of the light source
- $\mathbf{N}$ = surface normal
- $\mathbf{L}$ = direction to light source

Diffuse Reflection - Mathematics

Lambert's Law Implementation

Diffuse component formula:

$$I_{\text{diffuse}} = \mathbf{k}_d \odot \mathbf{I}_l (\mathbf{N} \cdot \mathbf{L})$$

where:

- $\mathbf{k}_d$ = diffuse reflection coefficient (material color)
- $\mathbf{I}_l$ = intensity of the light source
- $\mathbf{N}$ = surface normal
- $\mathbf{L}$ = direction to light source

With Clamping:

$$I_{\text{diffuse}} = \mathbf{k}_d \odot \mathbf{I}_l \max(0, \mathbf{N} \cdot \mathbf{L})$$

To avoid negative lighting.

Diffuse Reflection - Mathematics

Lambert's Law Implementation

Diffuse component formula:

$$I_{\text{diffuse}} = \mathbf{k}_d \odot \mathbf{I}_l(\mathbf{N} \cdot \mathbf{L})$$

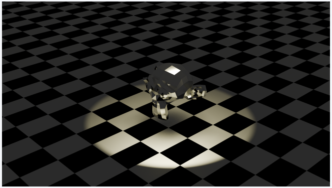
where:

- $\mathbf{k}_d$ = diffuse reflection coefficient (material color)
- $\mathbf{I}_l$ = intensity of the light source
- $\mathbf{N}$ = surface normal
- $\mathbf{L}$ = direction to light source

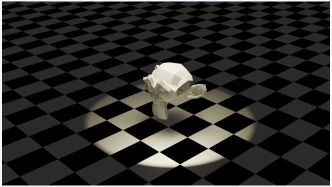
With Clamping:

$$I_{\text{diffuse}} = \mathbf{k}_d \odot \mathbf{I}_l \max(0, \mathbf{N} \cdot \mathbf{L})$$

To avoid negative lighting.



Low diffuse



High diffuse

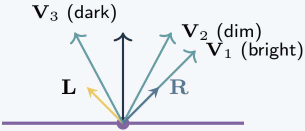
Specular Reflection - Introduction

Shiny Surfaces

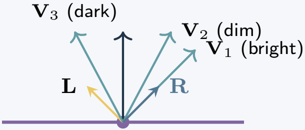
Examples:
Mirrors, metals, plastic

Characteristics:

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



Specular Reflection - Introduction



Shiny Surfaces

Examples:
Mirrors, metals, plastic

Characteristics:

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

Examples of diffuse surfaces: metal and glass

Perfect Reflection Theory

Law of Reflection

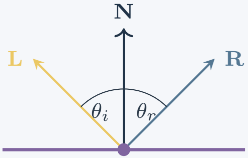
Physical principle: Angle of incidence equals angle of reflection

Vector formulation:

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

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:

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

where:

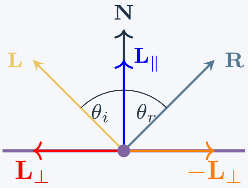
- $\mathbf{R}$ = reflection direction
- $\mathbf{N}$ = surface normal
- $\mathbf{L}$ = direction to light source

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

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

$$\mathbf{L}_{\perp} = \mathbf{L} - \mathbf{L}_{\parallel}$$

$$\mathbf{R} = \mathbf{L}_{\parallel} - \mathbf{L}_{\perp} = 2\mathbf{L}_{\parallel} - \mathbf{L}$$

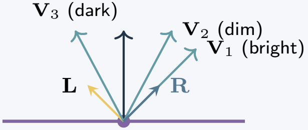


$\theta_i = \theta_r$

Phong Specular Model - Insight

Phong's Insight

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

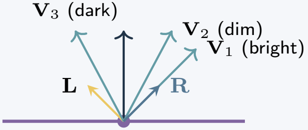


Phong Specular Model - Insight

Phong's Insight

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 - Insight

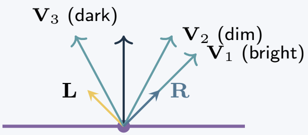
Phong's Insight

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 smooth falloff



Specular Mathematics: Intensity Calculation

Phong Specular Formula

Specular intensity:

$$I_{\text{specular}} = \mathbf{k}_s \odot \mathbf{I}_l (\mathbf{R} \cdot \mathbf{V})^n \tag{1}$$

where:

- $\mathbf{k}_s$ = specular reflection coefficient
- $\mathbf{I}_l$ = light intensity
- $\mathbf{R}$ = reflection direction
- $\mathbf{V}$ = view direction
- n = shininess exponent (controls highlight size)

Specular Mathematics: Intensity Calculation

Phong Specular Formula

Specular intensity:

$$I_{\text{specular}} = \mathbf{k}_s \odot \mathbf{I}_l (\mathbf{R} \cdot \mathbf{V})^n \quad (1)$$

where:

- $\mathbf{k}_s$ = specular reflection coefficient
- $\mathbf{I}_l$ = light intensity
- $\mathbf{R}$ = reflection direction
- $\mathbf{V}$ = view direction
- n = shininess exponent (controls highlight size)

With clamping:

$$I_{\text{specular}} = \mathbf{k}_s \odot \mathbf{I}_l \max(0, \mathbf{R} \cdot \mathbf{V})^n \quad (2)$$

To avoid negative lighting.

Specular Mathematics: Intensity Calculation

Phong Specular Formula

Specular intensity:

$$I_{\text{specular}} = \mathbf{k}_s \odot \mathbf{I}_l (\mathbf{R} \cdot \mathbf{V})^n \qquad (1)$$

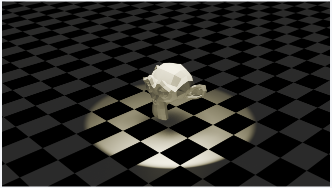
where:

- $\mathbf{k}_s$ = specular reflection coefficient
- $\mathbf{I}_l$ = light intensity
- $\mathbf{R}$ = reflection direction
- $\mathbf{V}$ = view direction
- n = shininess exponent (controls highlight size)

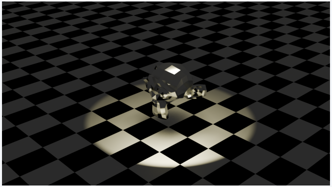
With clamping:

$$I_{\text{specular}} = \mathbf{k}_s \odot \mathbf{I}_l \max(0, \mathbf{R} \cdot \mathbf{V})^n \qquad (2)$$

To avoid negative lighting.



Low specular



High specular

Shininess Parameter (n) - Effect on Highlights

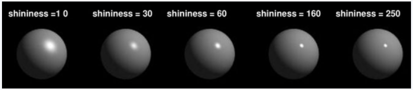
Shininess Exponent

Mathematical effect:

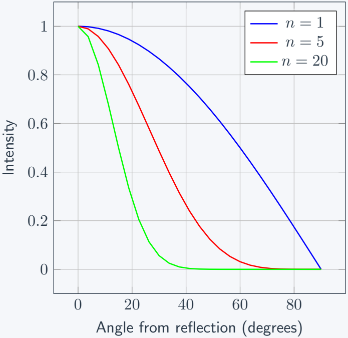
$$(\cos(\alpha))^n$$

As n increases:

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



Spheres with different shininess parameters



Complete Phong Equation Assembly

Putting It All Together

Complete Phong illumination model:

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

Complete Phong Equation Assembly

Putting It All Together

Complete Phong illumination model:

$$I = I_{\textsf{ambient}} + I_{\textsf{diffuse}} + I_{\textsf{specular}}$$

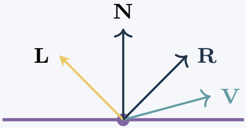
$$\mathbf{I} = \mathbf{k}_a \odot \mathbf{I_a} + \sum_{i=1}^{n} \mathbf{I_{l_i}} \odot \left[\mathbf{k}_d \max(0, \mathbf{N} \cdot \mathbf{L}_i) + \mathbf{k}_s \max(0, \mathbf{R}_i \cdot \mathbf{V})^n\right]$$

Blinn-Phong: A More Efficient Alternative

Motivation for Blinn-Phong

Problem with Phong: Computing the reflection vector $\mathbf{R}$ is expensive

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



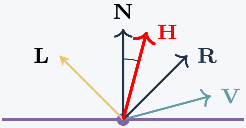
Blinn-Phong: A More Efficient Alternative

Motivation for Blinn-Phong

Problem with Phong: Computing the reflection vector $\mathbf{R}$ is expensive

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

Jim Blinn's solution (1977): Use a halfway vector instead



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

Blinn-Phong: A More Efficient Alternative

Motivation for Blinn-Phong

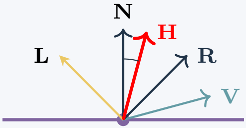
Problem with Phong: Computing the reflection vector $\mathbf{R}$ is expensive

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

Jim Blinn's solution (1977): Use a halfway vector instead

Key insight:

- When $\mathbf{V} = \mathbf{R}$, the halfway vector $\mathbf{H}$ equals the normal $\mathbf{N}$
- We can measure the angle between $\mathbf{H}$ and $\mathbf{N}$ instead
- Much cheaper to compute!



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

Blinn-Phong Mathematics and Comparison

Blinn-Phong Formula

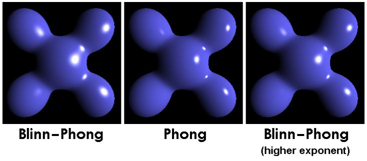
Halfway vector calculation:

$$\mathbf{H} = \frac{\mathbf{L} + \mathbf{V}}{|\mathbf{L} + \mathbf{V}|} \tag{3}$$

Specular term:

$$I_{\text{spec}} = \mathbf{k}_s \odot \mathbf{I}_l \max(0, \mathbf{N} \cdot \mathbf{H})^{n'} \tag{4}$$

where n' is typically 2-4 times larger than Phong's n



Blinn-Phong vs Phong

Blinn-Phong Mathematics and Comparison

Blinn-Phong Formula

Halfway vector calculation:

$$\mathbf{H} = \frac{\mathbf{L} + \mathbf{V}}{|\mathbf{L} + \mathbf{V}|} \tag{3}$$

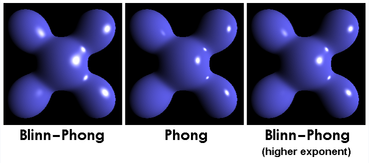
Specular term:

$$I_{\text{spec}} = \mathbf{k}_s \odot \mathbf{I}_l \max(0, \mathbf{N} \cdot \mathbf{H})^{n'} \tag{4}$$

where n' is typically 2-4 times larger than Phong's n

Performance comparison:

- **Phong:** 5 operations (2 dot products, 1 scalar multiply, 2 vector ops)
- **Blinn-Phong:** 4 operations (1 vector add, 1 normalize, 1 dot product)



Blinn-Phong vs Phong

Texturing

What Are Textures?

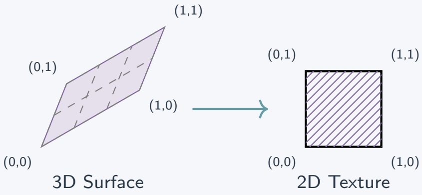
Texture Definition

Texture: A 2D image that defines surface properties as a function of position

Key idea: Map 2D texture coordinates to 3D surface points

Types of information:

- **Color** - Base surface appearance
- **Material properties** - Shininess, roughness
- **Surface details** - Bumps, scratches, patterns
- **Geometry** - Height variations



Texture Coordinates (UV Mapping)

UV Coordinate System

Parameterization: Map 3D surface to 2D texture space

Each vertex has:

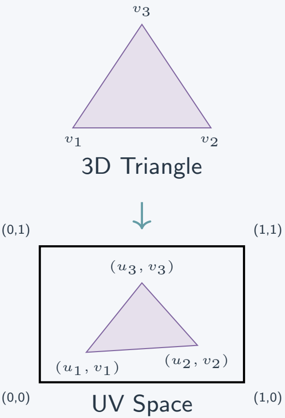
- 3D position: (x, y, z)
- 2D texture coordinates: (u, v)

Convention:

- $u, v \in [0, 1]$
- $(0, 0)$ = bottom-left of texture
- $(1, 1)$ = top-right of texture

Interpolation: Use barycentric coordinates for interior points

$$(u, v) = \alpha(u_1, v_1) + \beta(u_2, v_2) + \gamma(u_3, v_3)$$

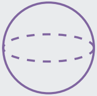


Common UV Mapping Techniques

Cube Mapping

Each face: $(0,1) \times (0,1)$

Spherical Mapping



Sphere

Spherical coordinates:

$$u = \frac{\phi}{2\pi}$$
$$v = \frac{\theta}{\pi}$$

Cylindrical Mapping

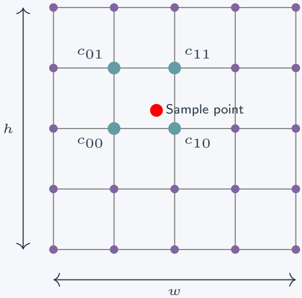
Cylindrical coordinates:

$$u = \frac{\phi}{2\pi}$$
$$v = \frac{z - z_{min}}{z_{max} - z_{min}}$$

Texture Sampling and Filtering

Texture Sampling

Problem: UV coordinates are continuous, texture pixels are discrete



Bilinear Interpolation

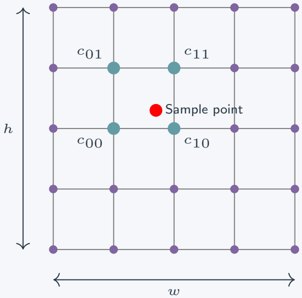
Texture Sampling and Filtering

Texture Sampling

Given texture T of resolution $W \times H$, let $w = W - 1$ and $h = H - 1$.

Nearest neighbor:

$$\text{color} = T[\text{round}(u \cdot w), \text{round}(v \cdot h)]$$



Bilinear Interpolation

Texture Sampling and Filtering

Texture Sampling

Given texture T of resolution $W \times H$, let $w = W - 1$ and $h = H - 1$.

Bilinear interpolation:

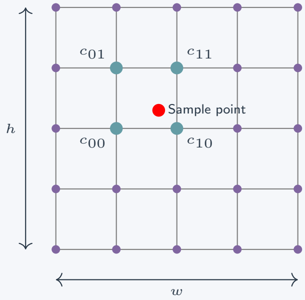
$$c_{00} = T[\lfloor u \cdot w \rfloor, \lfloor v \cdot h \rfloor]$$
$$c_{10} = T[\lfloor u \cdot w \rfloor + 1, \lfloor v \cdot h \rfloor]$$
$$c_{01} = T[\lfloor u \cdot w \rfloor, \lfloor v \cdot h \rfloor + 1]$$
$$c_{11} = T[\lfloor u \cdot w \rfloor + 1, \lfloor v \cdot h \rfloor + 1]$$

Final color:

$$\text{color} = \text{lerp}(\text{lerp}(c_{00}, c_{10}, t_u), \text{lerp}(c_{01}, c_{11}, f_u), t_v)$$

Where:

$$t_u = u \cdot w - \lfloor u \cdot w \rfloor$$
$$t_v = v \cdot h - \lfloor v \cdot h \rfloor$$



Bilinear Interpolation

lerp

Linear interpolation: Blend two values based on a factor $t \in [0, 1]$

$$\text{lerp}(a, b, t) = (1 - t) \cdot a + t \cdot b$$

Texture Sampling and Filtering

Texture Sampling

Given texture T of resolution $W \times H$, let $w = W - 1$ and $h = H - 1$.

Bilinear interpolation:

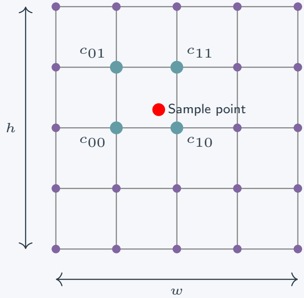
$$c_{00} = T[\lfloor u \cdot w \rfloor, \lfloor v \cdot h \rfloor]$$
$$c_{10} = T[\lfloor u \cdot w \rfloor + 1, \lfloor v \cdot h \rfloor]$$
$$c_{01} = T[\lfloor u \cdot w \rfloor, \lfloor v \cdot h \rfloor + 1]$$
$$c_{11} = T[\lfloor u \cdot w \rfloor + 1, \lfloor v \cdot h \rfloor + 1]$$

Final color:

$$\text{color} = \text{lerp}(\text{lerp}(c_{00}, c_{10}, t_u), \text{lerp}(c_{01}, c_{11}, f_u), t_v)$$

Where:

$$t_u = u \cdot w - \lfloor u \cdot w \rfloor$$
$$t_v = v \cdot h - \lfloor v \cdot h \rfloor$$



Bilinear Interpolation

Addressing Modes

Repeat: $u \bmod 1$
Clamp: $\max(0, \min(1, u))$
Mirror: Reflect at boundaries

Types of Textures

Diffuse/Albedo Maps

Purpose: Define base surface color
Usage in Phong:

$$k_d(u, v) = T_{\text{diffuse}}(u, v)$$

Effect: Varies surface color across object

Specular Maps

Purpose: Control shininess variation
Usage in Phong:

$$k_s(u, v) = T_{\text{specular}}(u, v)$$
$$n(u, v) = T_{\text{roughness}}(u, v) \cdot n_{\max}$$

Effect: Some areas shinier than others

Normal Maps

Purpose: Add surface detail without geometry
Storage: RGB values encode XYZ normal components

$$\mathbf{N}(u, v) = 2 \cdot T_{\text{normal}}(u, v) - 1$$

Effect: Bumps and surface details

Height/Displacement Maps

Purpose: Modify actual geometry
Usage:

$$\mathbf{P}'(u, v) = \mathbf{P}(u, v) + h(u, v) \cdot \mathbf{N}(u, v)$$

Effect: Actual geometric displacement

Textured Phong Model

Enhanced Phong with Textures

Original Phong:

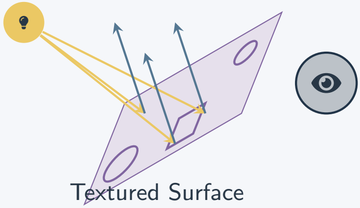
$$I = k_a I_a + k_d \mathbf{I}_l (\mathbf{N} \cdot \mathbf{L}) + k_s \mathbf{I}_l (\mathbf{R} \cdot \mathbf{V})^n$$

Textured Phong:

$$I = k_a(u,v) I_a + k_d(u,v) \mathbf{I}_l (\mathbf{N}' \cdot \mathbf{L}) \\ + k_s(u,v) \mathbf{I}_l (\mathbf{R}' \cdot \mathbf{V})^{n(u,v)}$$

Where:

- $k_d(u,v) = T_{\text{diffuse}}(u,v)$
- $k_s(u,v) = T_{\text{specular}}(u,v)$
- $\mathbf{N}' = \text{perturb}(\mathbf{N}, T_{\text{normal}}(u,v))$
- $n(u,v) = T_{\text{roughness}}(u,v) \cdot n_{\max}$



Different surface properties

Normal Mapping

Normal Map Mathematics

Tangent Space: Local coordinate system per vertex

Tangent vectors:

$$\mathbf{T} = \frac{\partial \mathbf{P}}{\partial u}$$

$$\mathbf{B} = \frac{\partial \mathbf{P}}{\partial v}$$

$$\mathbf{N} = \mathbf{T} \times \mathbf{B}$$

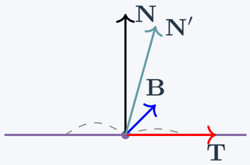
TBN Matrix:

$$\mathbf{M} = [\mathbf{T}, \mathbf{B}, \mathbf{N}]$$

Normal perturbation:

$$\mathbf{N}_{\text{map}} = 2 \cdot T_{\text{normal}}(u, v) - 1$$

$$\mathbf{N}' = \mathbf{M} \cdot \mathbf{N}_{\text{map}}$$



Bumpy surface appearance

Normal Map Colors

Blue dominant: Flat surface (pointing up)

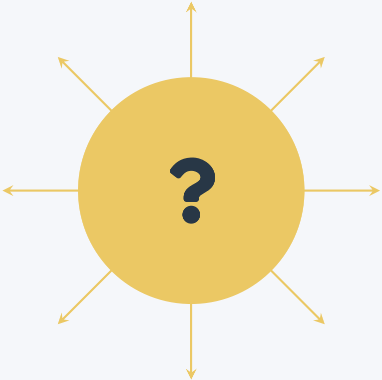
Red/Green: Surface tilted in X/Y directions

RGB → XYZ:

$(r, g, b) \to (2r - 1, 2g - 1, 2b - 1)$

Questions & Discussion

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