

5CM507 Graphics

Lecture 05 Lighting and Shading

Dr Youbing Zhao

October 20, 2025

Contents



- ▶ Lighting: computing the luminous intensity at a 3D point on a surface
 - ▶ Phong and **Blinn-Phong**
- ▶ Shading: assigning colours to 2D pixels
 - ▶ Basic Shading Models: Flat, Gouraud, **Phong**

Lighting Models

Why do we need lighting and shading

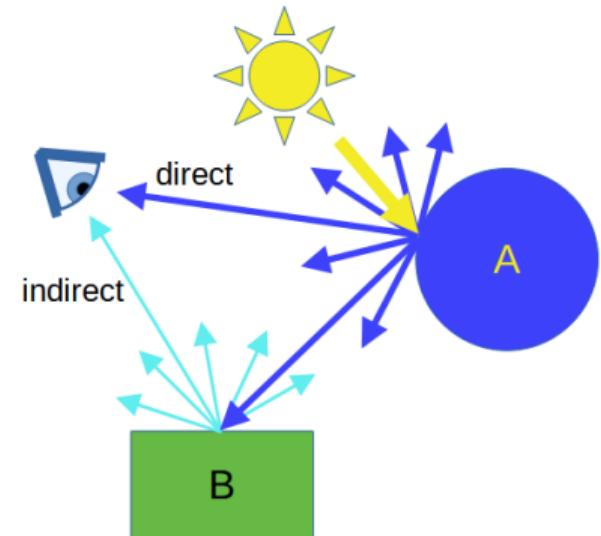
We perceive shapes through light reflected from objects



The Mousou black ink claims to absorb 99.4% of light

Light-Material Interaction

- ▶ Need to consider
 - ▶ Light sources
 - ▶ Location of viewer
 - ▶ Surface orientation
 - ▶ Material properties (colour absorption, smoothness, etc)
- ▶ Light partially absorbed and partially scattered (reflected)
 - ▶ Colour: the light reflected, the rest absorbed
 - ▶ Intensity: light direction and surface orientation
 - ▶ Scattering : smoothness of the surface

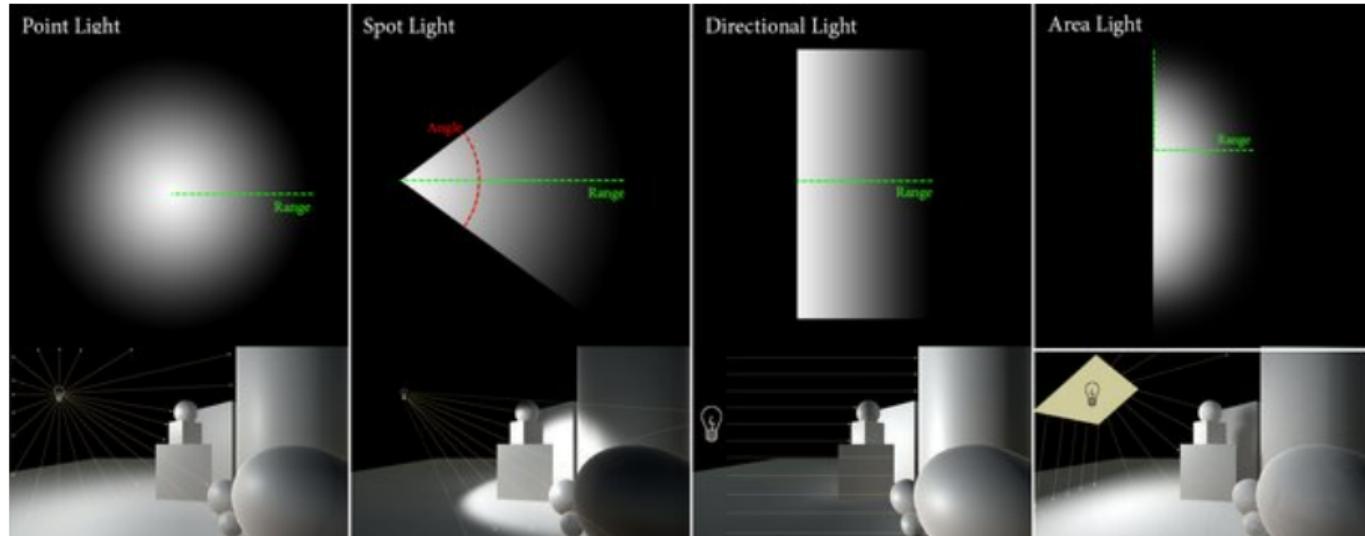


Light Sources

Basic Light Sources

Classic OpenGL : ambient, directional, point, and spotlight sources.

Modern OpenGL : no built-in lights — implemented flexibly by the user in GLSL.

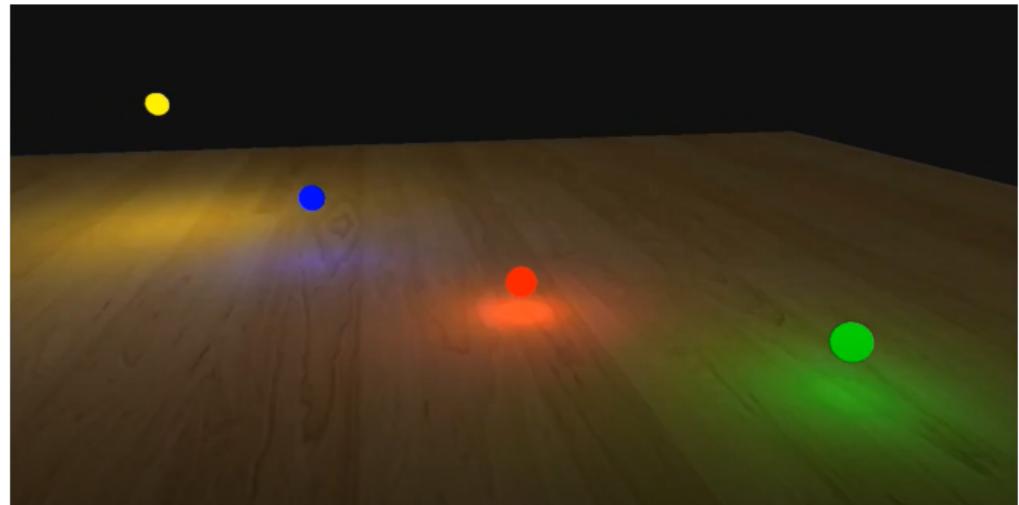
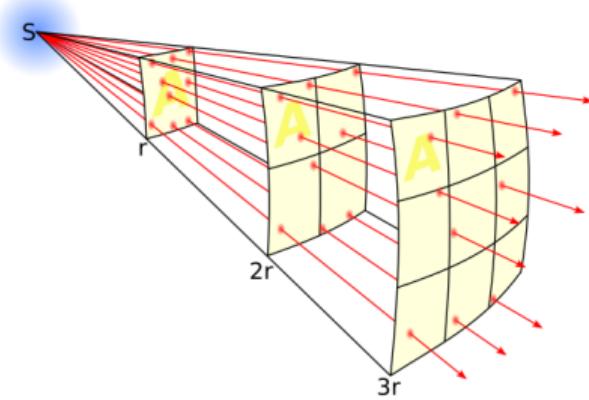


[Benjamin Kahl]

Light Attenuation - Inverse Square Falloff

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

In practice: $I = I_0 \frac{1}{a+bd+cd^2}$



The Basic Lighting Models: Phong and Blinn-Phong

Surface Reflection Types

Ambient : all-to-all directions

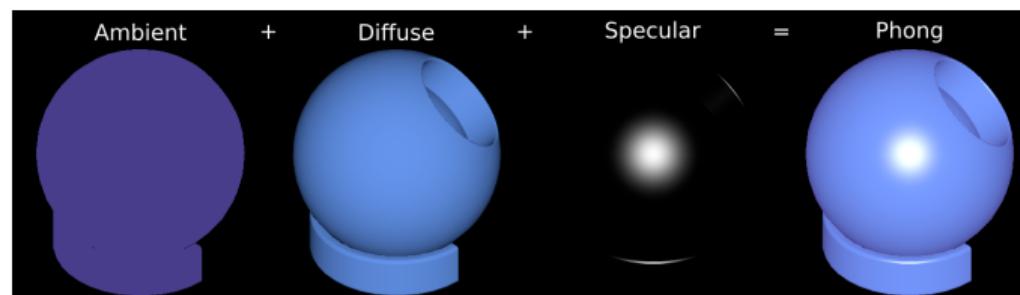
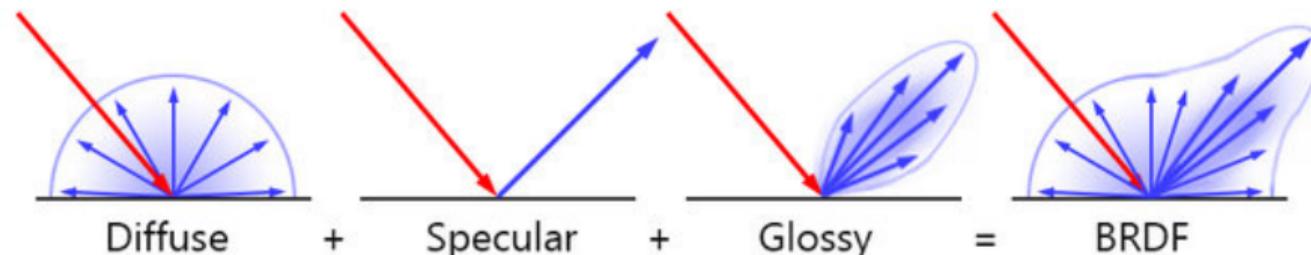
- ▶ Intensity: independent of surface normals

Diffuse : scattered, one-to-all directions

- ▶ Intensity: dependent on surface normals

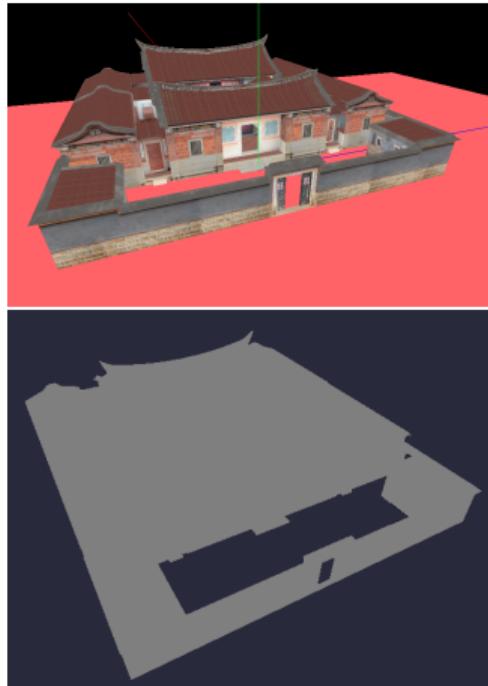
Specular : one-to-one direction (mirror-like) or spread within a range (glossy)

- ▶ Intensity: dependent on surface normal and viewing direction



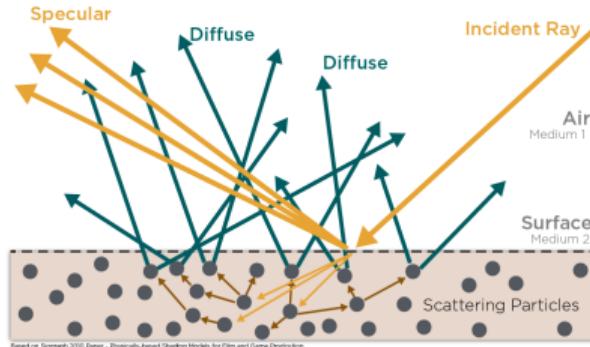
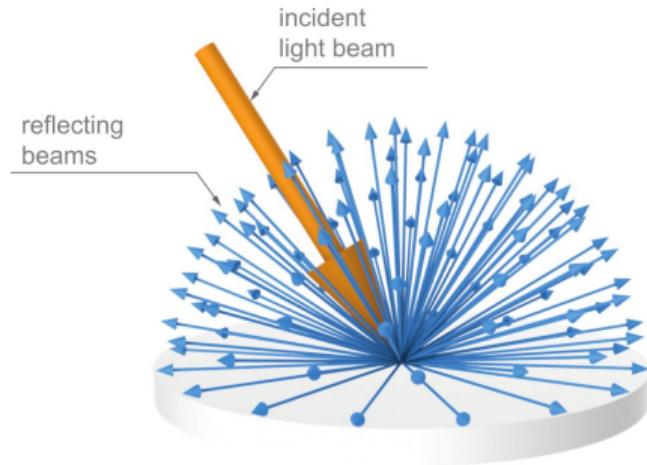
Ambient Light

- ▶ Background light without directions
- ▶ Reflection independent of surface shapes
- ▶ Approx. of an averaged lighting
- ▶ $I_{amb} = I_a k_a$
 - ▶ k_a : reflection coefficients in (R, G, B)
 - ▶ I_a : intensity of ambient light in (R, G, B)
- ▶ Rendering: flat with outlines



Diffuse Reflection

- ▶ Perfect diffuse : scattered **EQUALLY** in all directions
- ▶ Dull, matt surfaces such as paper, bricks, carpet, etc.



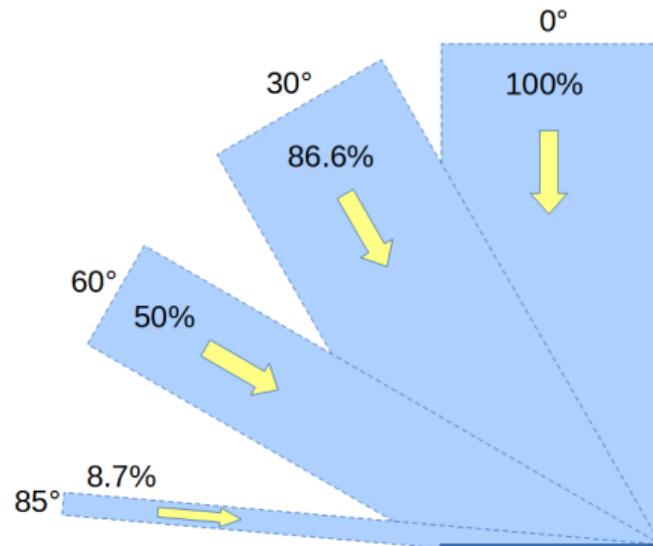
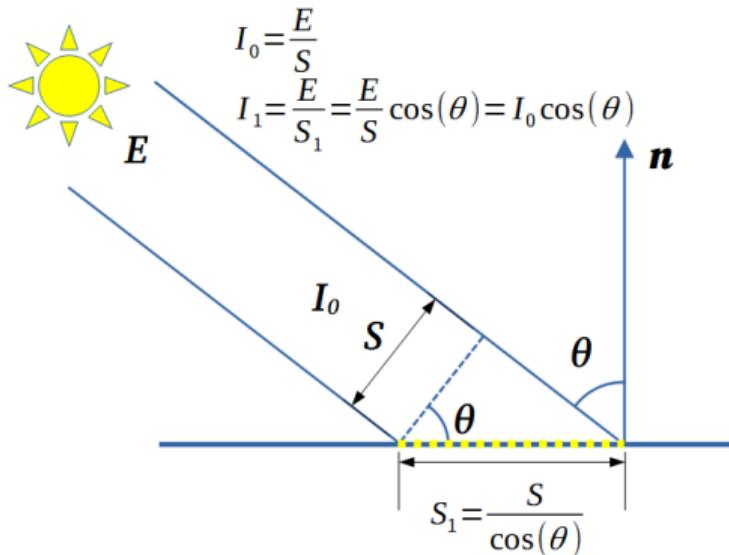
Based on Sponza 2010 Paper - Physically-based Shading Models for Film and Game Production

Diffuse Light Intensity: Lambert's Reflectance Model

the Cosine Law

Received incident light intensity on the surface:

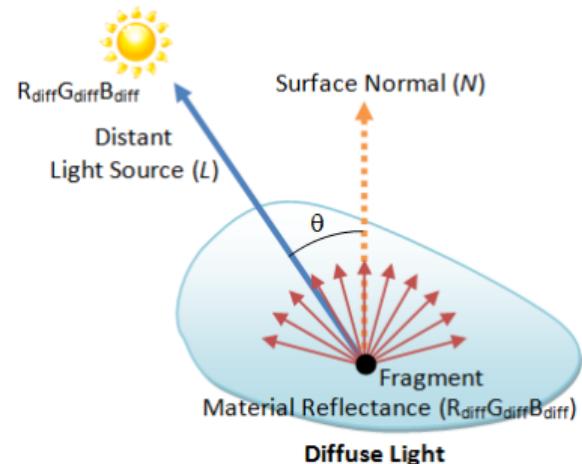
$$I_{surf} = I_0 \cos \theta \quad (\cos \theta = \frac{\mathbf{l} \cdot \mathbf{n}}{\|\mathbf{l} \cdot \mathbf{n}\|})$$



The Diffuse Light

The incident light is scattered equally in all directions

- ▶ $I_d = k_d I_{surf} = I_i k_d \cos \theta = I_i k_d (n \cdot l)$
- ▶ I_d : resulting diffuse intensity
- ▶ I_i , incoming light intensity in (R,G,B)
- ▶ k_d (diffuse) surface reflectance coefficient in (R,G,B) [0,1]
- ▶ θ : angle between normal and light direction

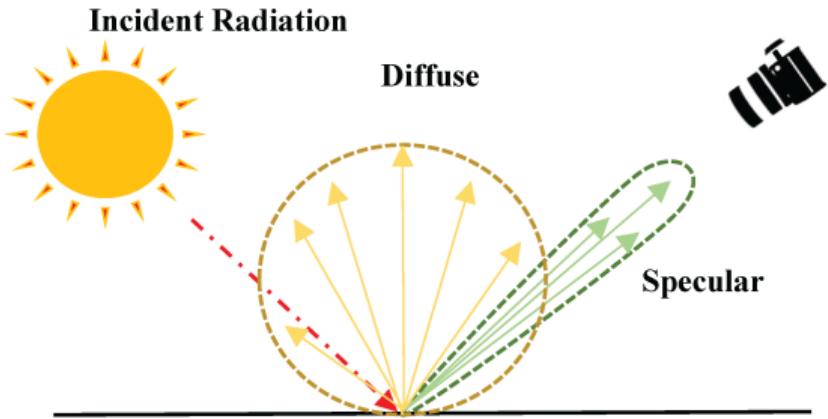


Specular Reflections - the Highlight

- ▶ Most surfaces are neither ideal diffusers nor perfectly specular (ideal reflectors)
- ▶ Specular highlights : concentrated around the direction of a perfect reflection



Less diffuse makes more metallic



Compute the Reflection Vector r

- ▶ Vectors

- ▶ \mathbf{l} : To incident light source
- ▶ \mathbf{v} Viewing direction to the viewer
- ▶ \mathbf{n} Surface normal, unit vector
- ▶ \mathbf{r} Perfect reflector

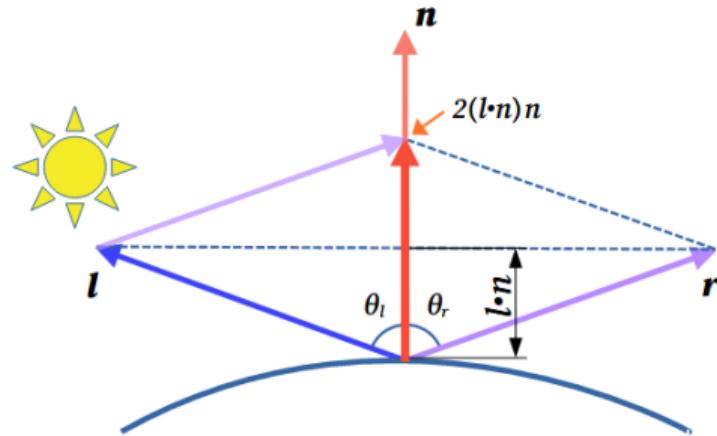
- ▶ Compute r

- ▶ $\theta_l = \theta_r$, $\mathbf{l}, \mathbf{n}, \mathbf{r}$ coplanar
- ▶ \mathbf{l} 's projection on \mathbf{n} : $\|\mathbf{l}\| \cos \theta_l = \mathbf{l} \cdot \mathbf{n}$
- ▶ Vector addition: $\mathbf{r} + \mathbf{l} = 2(\mathbf{l} \cdot \mathbf{n})\mathbf{n}$

$$\mathbf{r} = 2(\mathbf{l} \cdot \mathbf{n})\mathbf{n} - \mathbf{l}$$

- ▶ GLSL

```
vec3 reflect(vec3 I=-light_vec, vec N)
```

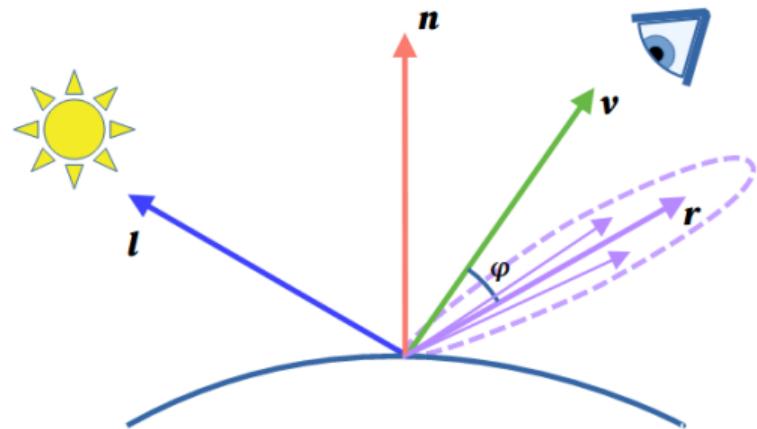


Modelling Specular Reflections

- ▶ Phong proposed using a term that dropped off as the angle between the viewer and the ideal reflection increased

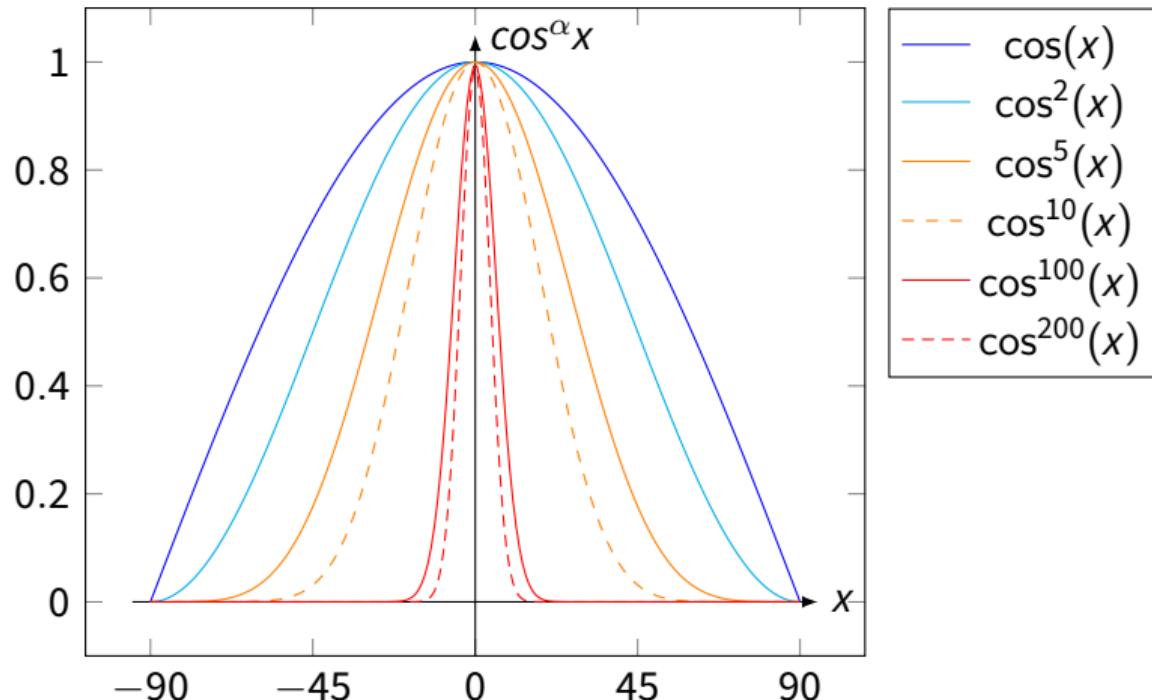
$$I_s = k_s I_i \cos^\alpha (\phi)$$

- ▶ I_s : specular light intensity
- ▶ k_s : absorption coefficient
- ▶ I_i : incoming light intensity
- ▶ $\cos \phi = \frac{r \cdot v}{\|r \cdot v\|}$
- ▶ α : shininess coefficient



The Shininess Coefficient

- ▶ Values of α between 100 and 200 correspond to metals
- ▶ Values between 5 and 10 give surface that look like plastic



Phong Model : Ambient + Diffuse + Specular

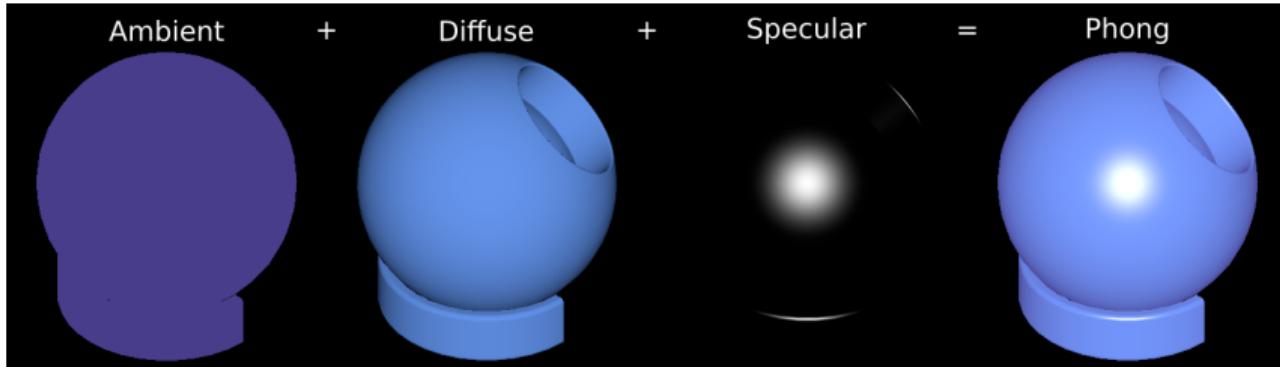
$$I_a k_a + \sum_p \frac{I}{A(d_p)} (k_d (n \cdot l_p) + k_s (r_p \cdot v)^\alpha)$$

$$I_{incoming} \propto \frac{1}{r^2}$$

Attenuation function $A(d) = a + bd + cd^2$



บุ้ย เตื่อง พีอง
Bùi Tường Phong
裴祥風
Phong shading



Blinn-Phong Model (Jim Blinn, 1977)

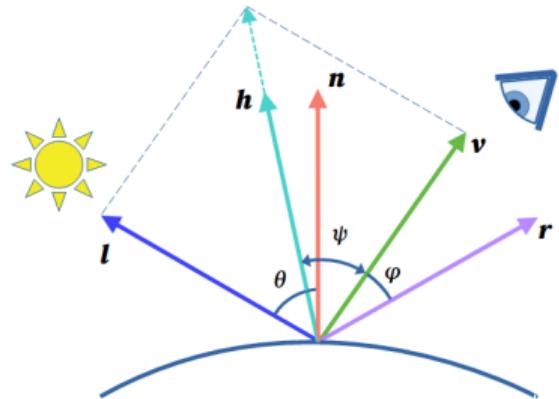
The Default Model We Use



- ▶ Change in the specular term
- ▶ Phong model: The specular term requires r for each vertex.
- ▶ Blinn-Phong model: use the halfway vector h halfway between l and v

$$h = \frac{(l+v)}{\|l+v\|}$$

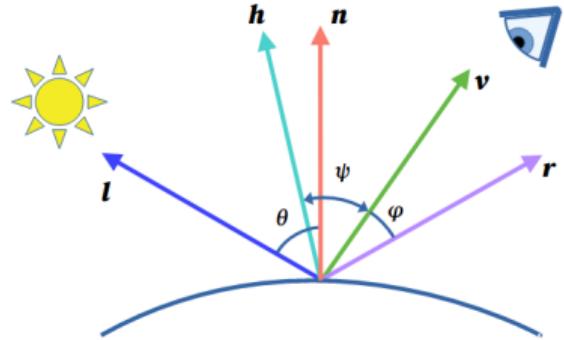
- ▶ Halfway vector represents normal of microfacets that would lead to mirror reflection to the eye
- ▶ Use cosine of angle between the half vector and the normal :
 $(v \cdot r)^\alpha \Rightarrow (n \cdot h)^\beta$ (β : shininess)
- ▶ The larger the angle between microfacet orientation and normal, the less likely



Complete Blinn-Phong Model

$$I_a k_a + \sum_p \frac{\mathbf{I}}{A(d_p)} \left(k_d (\mathbf{l}_p \cdot \mathbf{n}) + k_s (\mathbf{h} \cdot \mathbf{n})^\beta \right)$$

- ▶ All colours, reflection coefficients have separate values for R,G,B
- ▶ Usually, $k_a = k_d$ coefficient
- ▶ For metals, $k_s = k_d$
 - ▶ Highlight is colour of material
- ▶ For plastics, specular coefficient = (x,x,x)
 - ▶ Highlight is colour of light
- ▶ Each k has three r, g, b components



Vertex Shader for Both Phong and Blinn-Phong



```
// vertex shader for both Phong and Blinn-Phong models
#version 410

in layout(location=0) vec3 pos;
in layout(location=1) vec3 aNormal;

// M V P matrices
uniform mat4 model;
uniform mat4 view;
uniform mat4 projection;

out vec3 normal;
out vec3 fragPos;
```

```
void main() {
    // homogeneous coordinate
    gl_Position = projection * view * model * vec4(pos, 1.0);

    // colour_vert = colour_in;
    fragPos = vec3(model * vec4(pos, 1.0));

    // convert mat4 normal to mat3
    mat3 normalMatrix = mat3(model);

    // only correct for rigid body transforms
    normal = normalMatrix * aNormal;
}
```

Phong and Blinn-Phong in Fragment Shaders

```
// fragment shader
#version 410
out vec4 FragColor;

vec3 fragPos;
vec3 normal;
// vec2 texCoords;

uniform sampler2D floorTexture;
uniform vec3 lightPos;
uniform vec3 viewPos;
uniform bool blinn;

void main()
{
    // specify a colour for our lab
    vec3 color = vec(1.0, 0.0 0.0);

    // or use a colour from a texture
    // vec3 color = texture(floorTexture, fs_in.TexCoords).rgb;

    // ambient
    vec3 ambient = 0.05 * color;
```

```
// diffuse
vec3 lightDir = normalize(lightPos - fragPos);
vec3 normal = normalize(normal);

float diff = max(dot(lightDir, normal), 0.0);
vec3 diffuse = diff * color;
// specular
vec3 viewDir = normalize(viewPos - fragPos);
vec3 reflectDir = reflect(-lightDir, normal);
float spec = 0.0;
if(blinn) {
    vec3 halfwayDir = normalize(lightDir + viewDir);
    spec = pow(max(dot(normal, halfwayDir), 0.0), 32.0);
}
else {
    vec3 reflectDir = reflect(-lightDir, normal);
    spec = pow(max(dot(viewDir, reflectDir), 0.0), 8.0);
}
// assuming bright white light colour
vec3 specular = vec3(0.3) * spec;

// The final combined colour
fragColor = vec4(ambient + diffuse + specular, 1.0);}
```

Shading in OpenGL

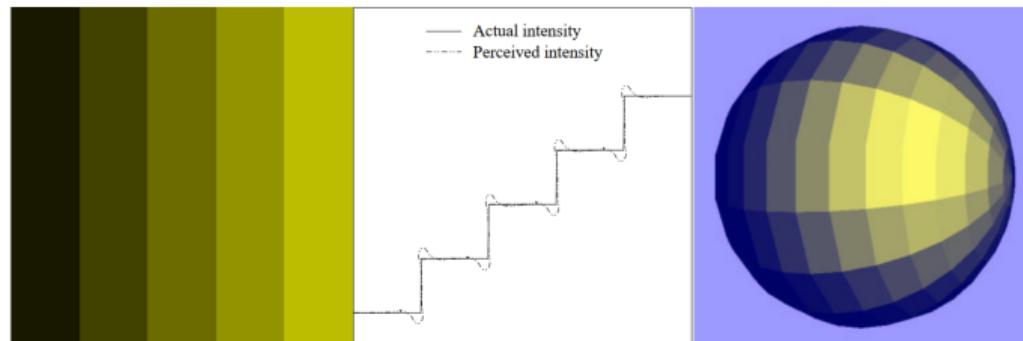
OpenGL Shading Models

- ▶ Flat shading: one normal, one colour per polygon
- ▶ Gouraud shading: vertex normals, interpolated colours per pixel
- ▶ Phong shading: interpolated normals per pixel



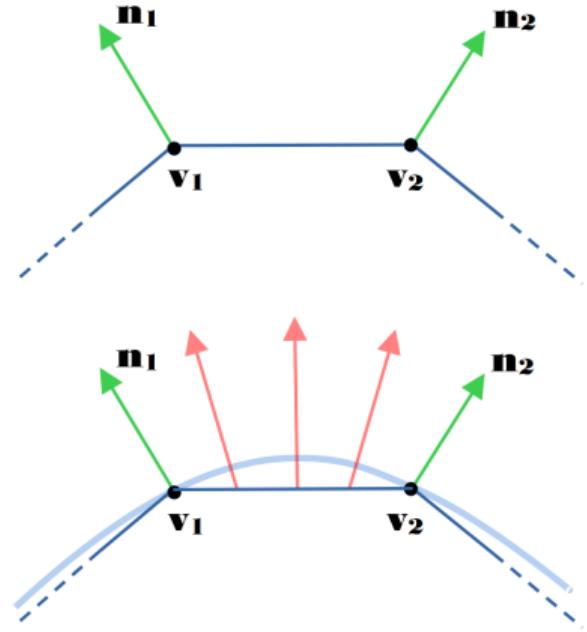
Flat Shading

- ▶ Each polygon shares the same normal vector
- ▶ Directional light
- ▶ All pixels in the same polygon are coloured by the same color
- ▶ Intensity discontinuities at polygon edges
- ▶ "flat" modifier in GLSL shaders: `flat out/in vec3 normal;`
- ▶ Mach band effect at polygon edges



Phong Shading (also by Bui Tuong Phong)

- ▶ Per-pixel shading
- ▶ Interpolating the normal vectors at the vertices
(again using barycentric coordinates)
- ▶ Applying illumination at each pixel



Interpolation of normals

Summary

- ▶ Basic Lighting Models : Ambient + Diffuse + Specular
 - ▶ Phong - reflection vector : $I_a k_a + \sum_p \frac{I}{a+b\mathbf{d}+c\mathbf{d}^2} (k_d (\mathbf{n} \cdot \mathbf{l}_p) + k_s (\mathbf{r}_p \cdot \mathbf{v})^\alpha)$
 - ▶ Blinn-Phong - half vector : $I_a k_a + \sum_p \frac{I}{a+b\mathbf{d}+c\mathbf{d}^2} (k_d (\mathbf{l}_p \cdot \mathbf{n}) + k_s (\mathbf{h} \cdot \mathbf{n})^\beta)$
- ▶ Phong shading: pixel-based, interpolation of vertex normals, normal per vertex

Pursuit of photorealism in games

