Shading

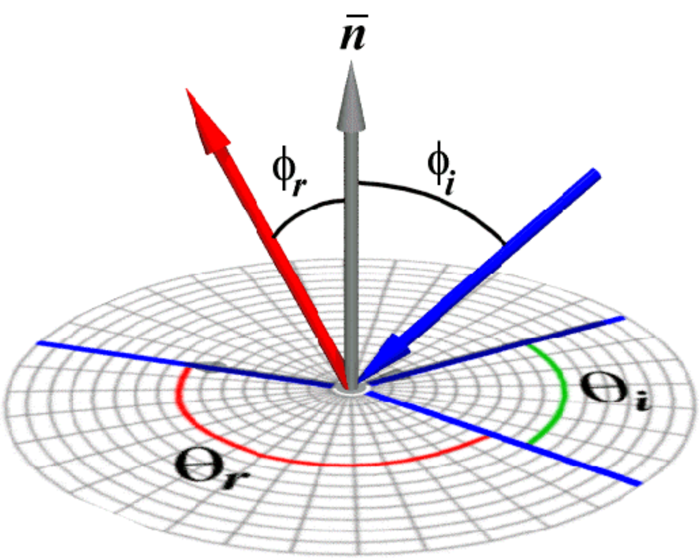
Goal of Shading

Capture light and material interactions in a scene based on camera position and orientation

The rendering equation is the physically-based model for light and material interactions

The Rendering Equation

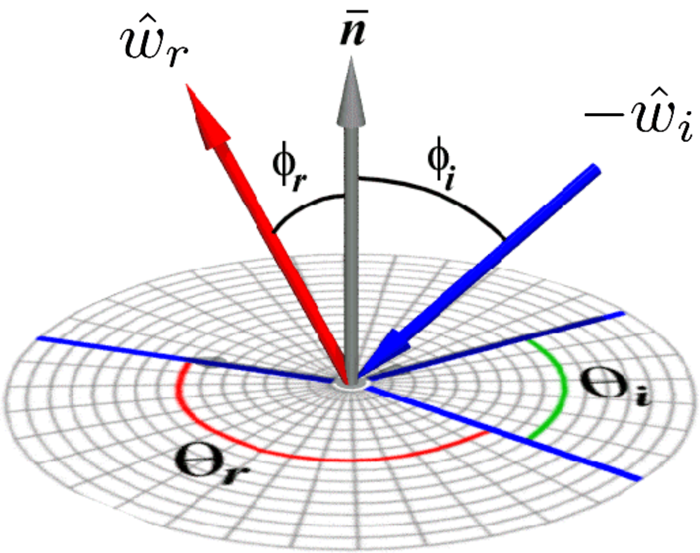
$$L_{\mathrm{out}}(\theta_r, \phi_r) = \int_{\theta_i} \int_{\phi_i} f_r(\theta_r, \phi_r, \theta_i, \phi_i) L_{\mathrm{in}}(\theta_i, \phi_i) \cos\theta_i$$



The Rendering Equation

$$L_{\text{out}}(\theta_r, \phi_r) = \int_{\theta_i} \int_{\phi_i} f_r(\theta_r, \phi_r, \theta_i, \phi_i) L_{\text{in}}(\theta_i, \phi_i) \cos \theta_i$$

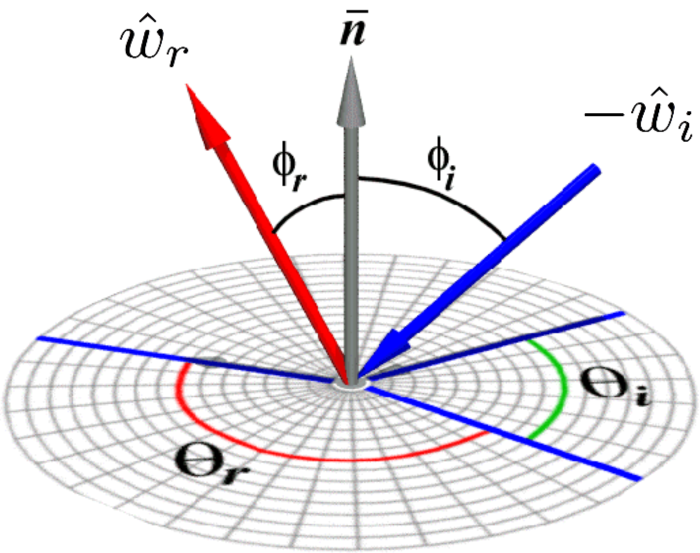
$$L_{\text{out}}(\hat{w}_r) = \int_{\hat{w}_i \in \text{hemisphere}} f_r(\hat{w}_r, \hat{w}_i)\ L_{\text{in}}(\hat{w}_i)\ \hat{w}_i \cdot \hat{n}$$



The Rendering Equation

$$L_{\text{out}}(\theta_r, \phi_r) = \int_{\theta_i} \int_{\phi_i} f_r(\theta_r, \phi_r, \theta_i, \phi_i) L_{\text{in}}(\theta_i, \phi_i) \cos \theta_i$$

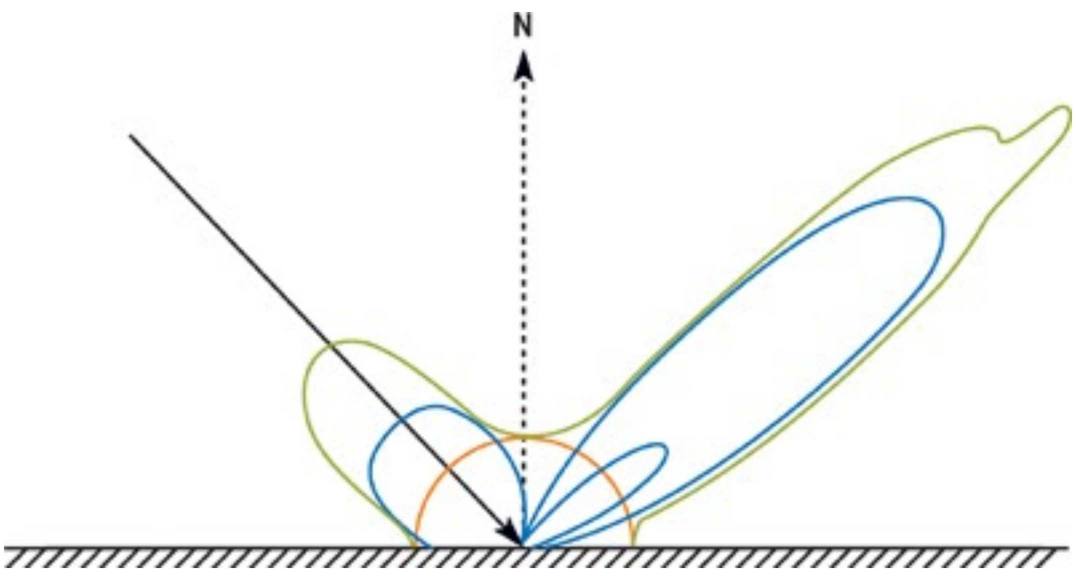
$$L_{\text{out}}(\hat{w}_r) = \int_{\hat{w}_i \in \text{hemisphere}} f_r(\hat{w}_r, \hat{w}_i) \; L_{\text{in}}(\hat{w}_i) \; \hat{w}_i \cdot \hat{n}$$



BRDF
"Bidirectional Reflectance Distribution Function"
(encodes material)

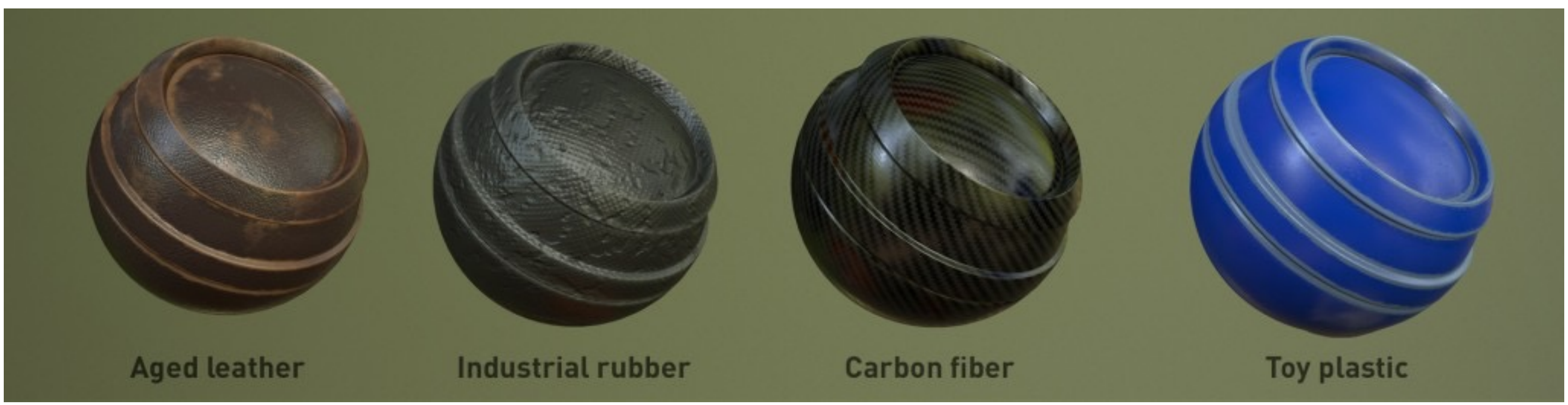
BRDFs

- Bidirectional Reflectance Distribution Function
- Captured for different materials, stored in libraries



BRDFs in Modern Graphics

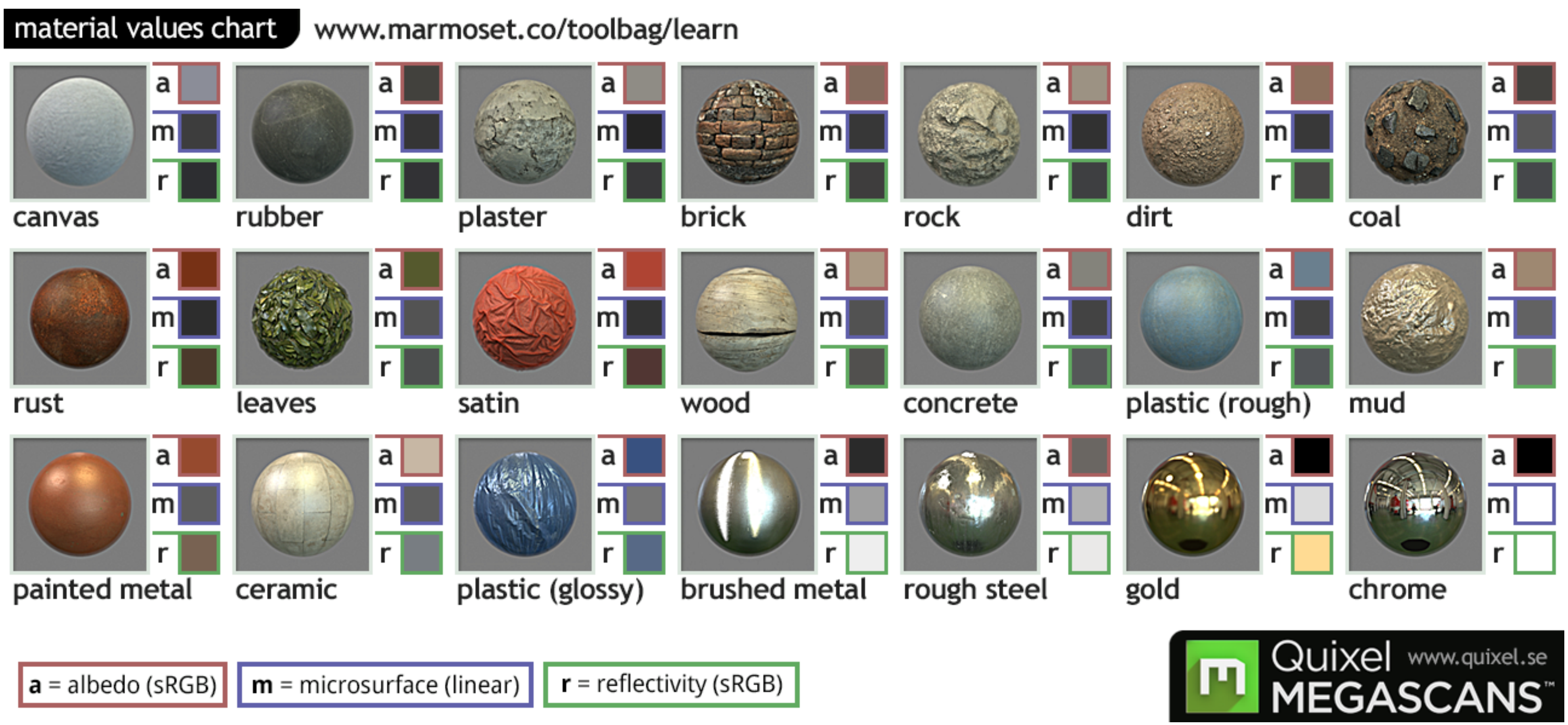
- Provide physically-based model for defining light reflectance
- Standard in modern graphics and game engines



Working with BRDFs

Artists modify material constants

BRDF shaders handle lighting calculations

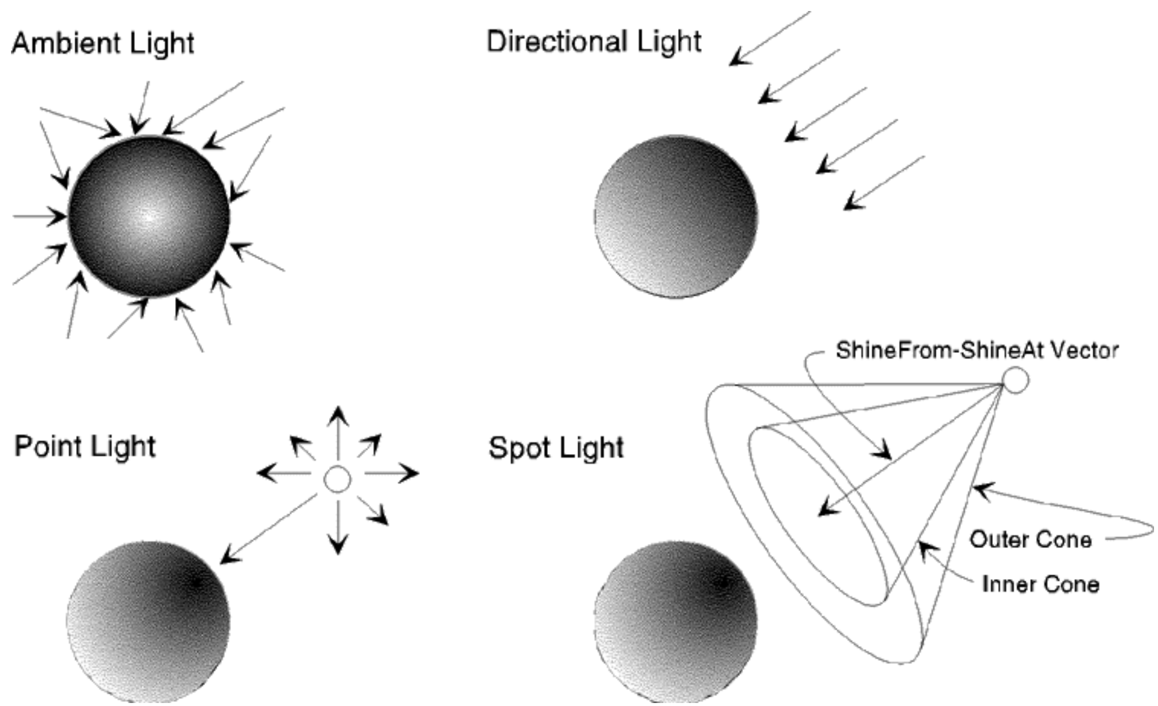


Local Illumination

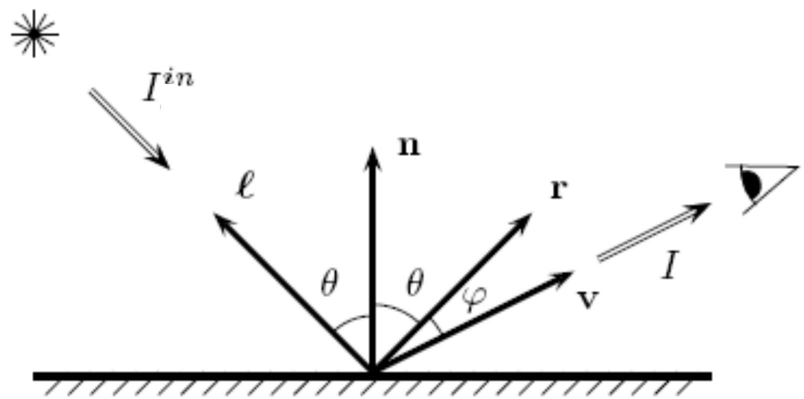
- Solving the full rendering equation is too expensive
  - Ground-truth path tracing is still not real-time
- Instead...
  - Do local illumination
  - “Hack in” reflections, shadows, color-bleed, ambient occlusion, etc

Light Sources

Intensity and direction of light sources change what surfaces are affected



Local Shading: Notation

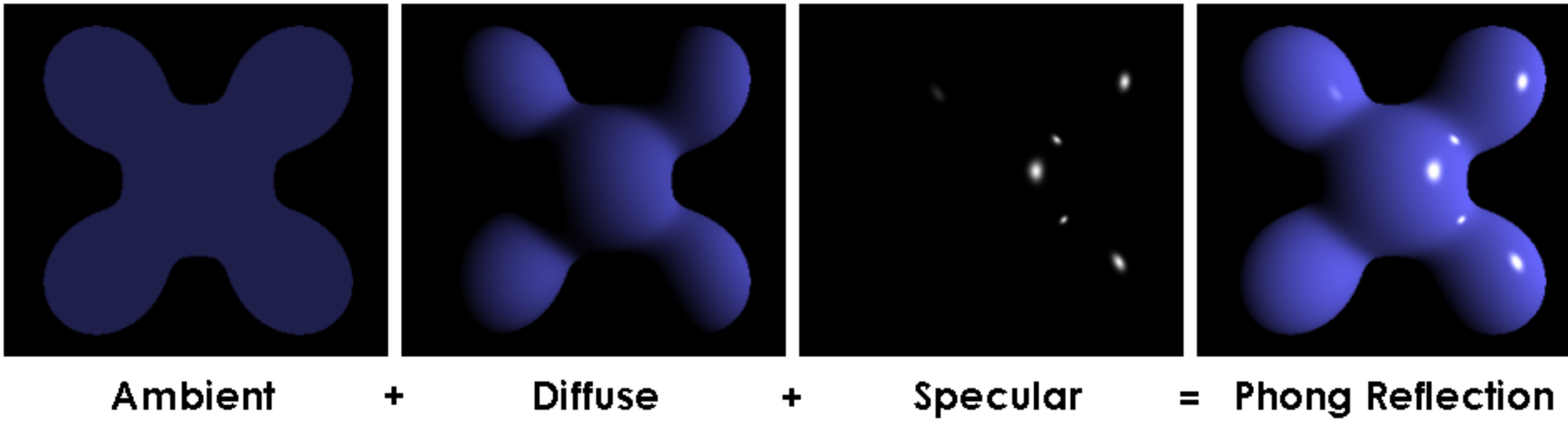


light intensity in, light intensity out

vector pointing to: light, normal direction, eye, reflection direction

Note that light intensity is related to wavelength, but we will treat intensity as a representation of RGB value

Phong Illumination



Emissive Term

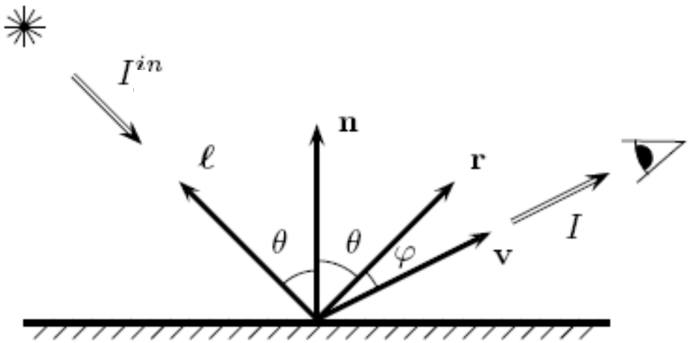
Polygon has color:

- I is resulting intensity
- k_e is emissivity

$$I = k_e$$

Often omitted as it's generally for special-purposes

Ambient Term

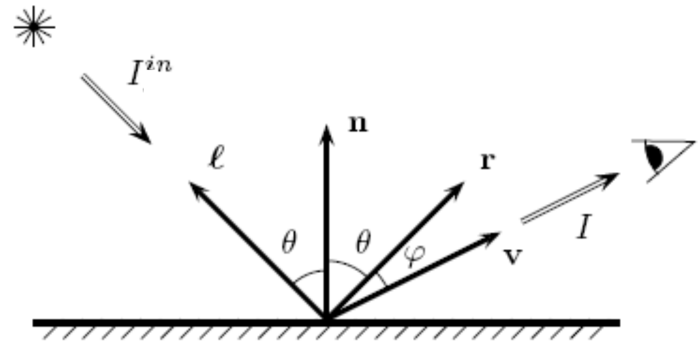


Ignore camera and light direction

- I_a is ambient intensity
- k_a is ambient reflection coefficient

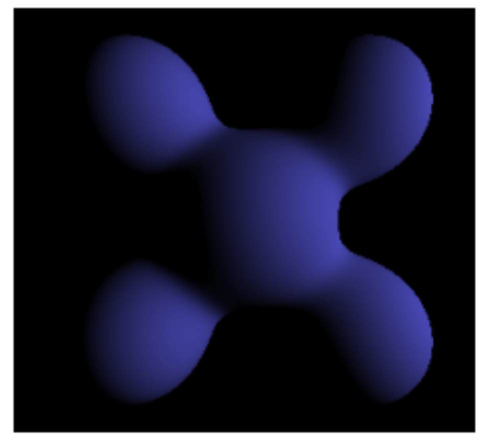
$$I = k_a I_a$$

Diffuse Term

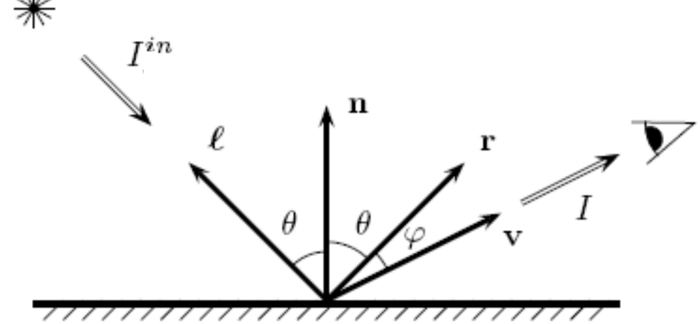


Lambertian surface – **constant** BRDF

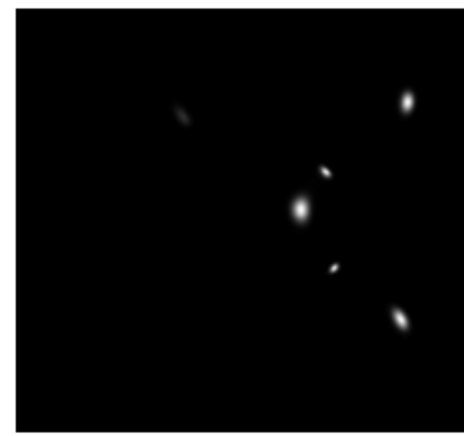
$$I = k_d I_i max(L \cdot N, 0)$$



Specular Term



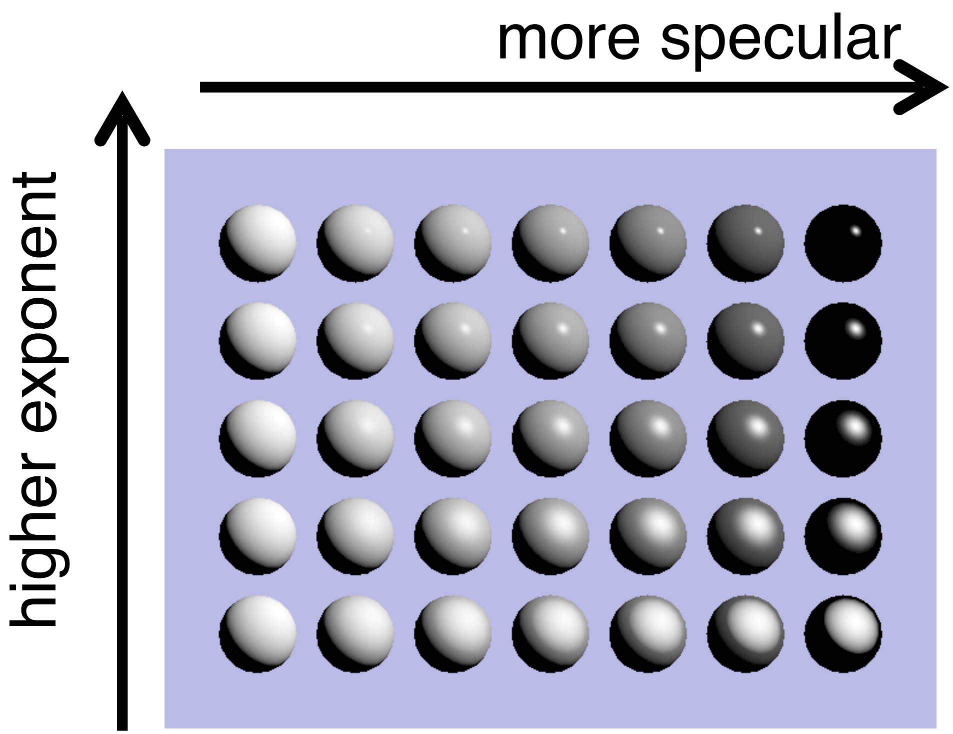
$$I = k_s I_i max(R \cdot V, 0)^n$$

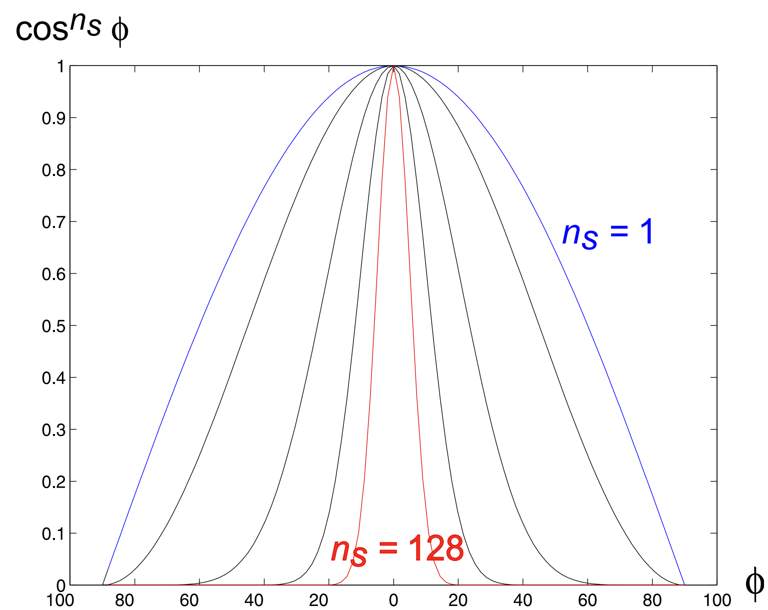


where *n* is specular coefficient

Looks like “highlight” that moves with light & eye

Specularity Coefficient





Phong Illumination Model

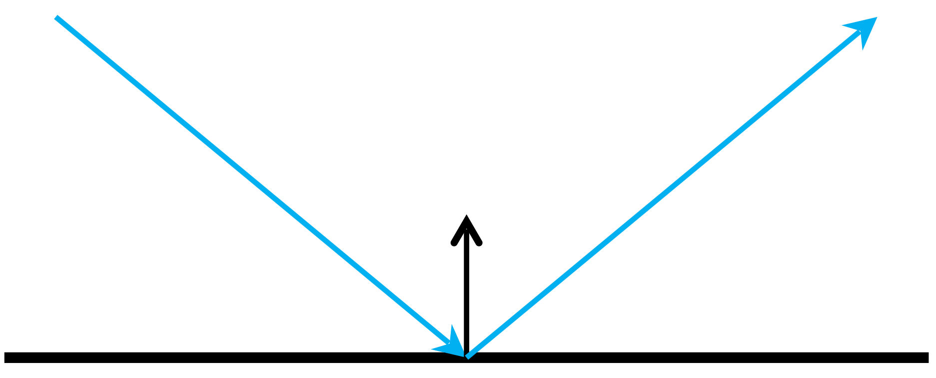
$$I = I_e + k_a I_a + \sum_{lights\ i} I_i (k_d max(L_i \cdot N, 0) + k_s max(R_i \cdot V, 0)^n)$$

Can Phong do this?

Purely specular (mirrored) surface

How To Achieve Perfect Specularity?

1. Incoming ray hits purely specular surface
2. Shoot secondary **reflection ray**
3. Set pixel color to color “seen” by reflection ray



Reflection in Practice

Objects may not be perfectly mirrored

- blend reflected color with basic shading

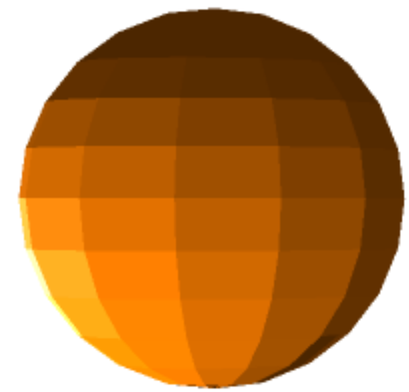
Objects have **base color**

- multiplies reflected color

Dealing with Discrete Geometry

Flat shading: use normal per face

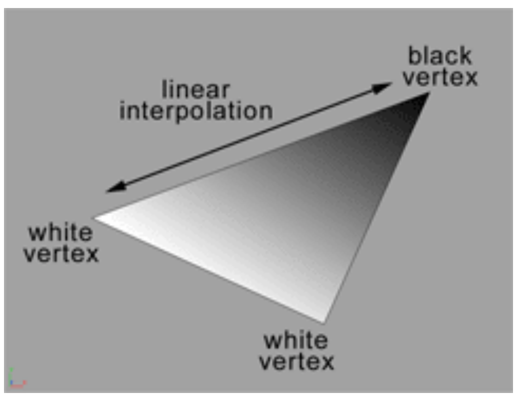
Very obvious discontinuities
at edges

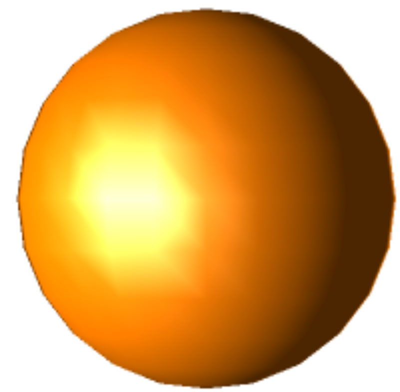


Only used for stylized “chunky” effect

Gouraud Interpolation

1. Compute color at vertices
2. Linearly interpolate over face

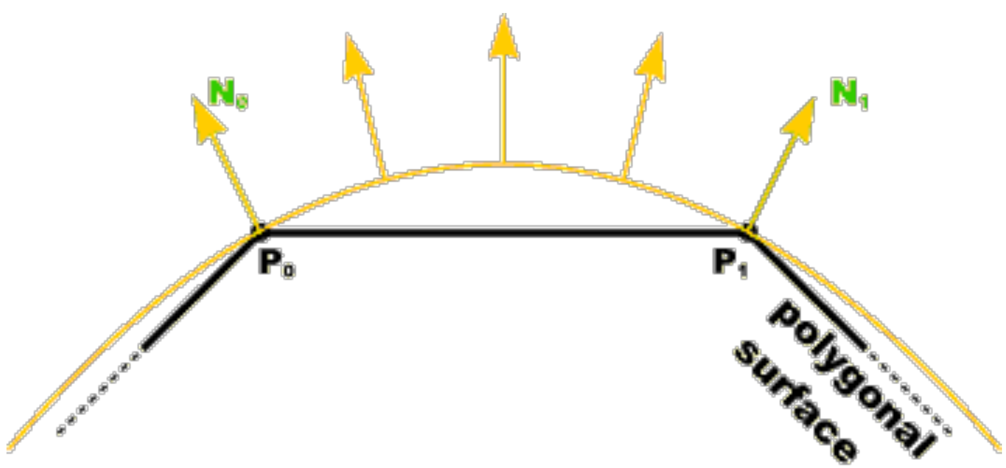


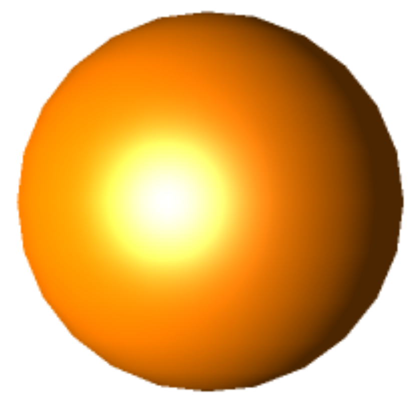


Color is continuous, but obvious artifacts
(nobody uses this anymore)

Phong Interpolation

- Linearly interpolate normals
- Renormalize normals(**important**)
- Compute color per pixel





Local vs Global Illumination Recap

Local:

- Shade each object based only on itself, the eye, and the light sources

Global:

- Take all other objects in scene into account
- Use BRDFs and the rendering equation

Ray-tracing In Practice…

- Take other objects into account, without full global illumination
- Common techniques exist for
  - Shadows
  - Reflections
  - Refractions
- Can add effects using maps, targeted ray casts, and pre-baked lighting
- Often combined with rasterization pipeline