Shading

Prof. George Wolberg
Dept. of Computer Science
City College of New York

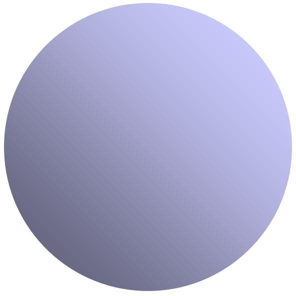
Objectives

- Learn to shade objects so their images appear three-dimensional
- Introduce the types of light-material interactions
- Build a simple reflection model (Phong model) that can be used with real-time graphics hardware

Why We Need Shading

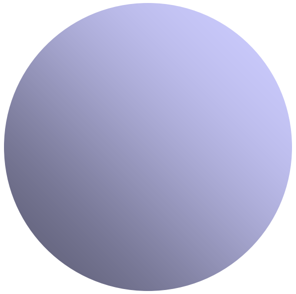
- Suppose we build a model of a sphere using many polygons and color it with **`glColor`**. We get something like

- But we want



Shading

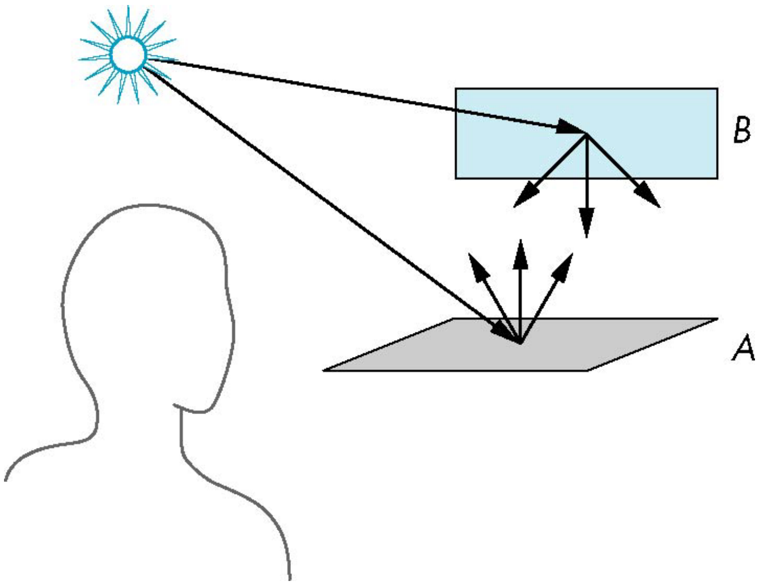
- Why does the image of a real sphere look like



- Light-material interactions cause each point to have a different color or shade
- Need to consider
  - Light sources
  - Material properties
  - Location of viewer
  - Surface orientation

Scattering

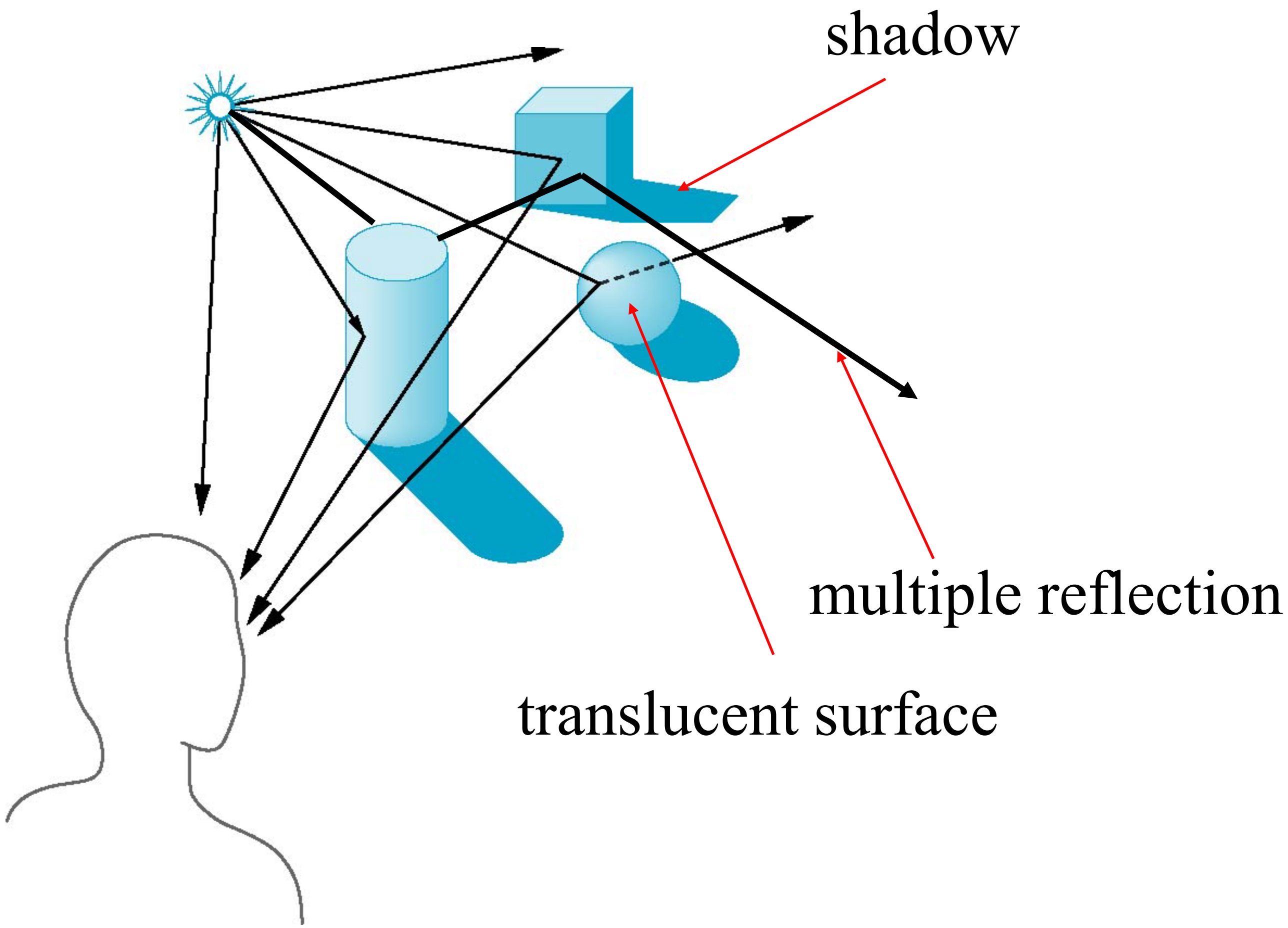
- Light strikes A
  - Some scattered
  - Some absorbed
- Some of scattered light strikes B
  - Some scattered
  - Some absorbed
- Some of this scattered light strikes A and so on



Rendering Equation

- The infinite scattering and absorption of light can be described by the *rendering equation*
  - Cannot be solved in general
  - Ray tracing is a special case for perfectly reflecting surfaces
- Rendering equation is global and includes
  - Shadows
  - Multiple scattering from object to object

Global Effects



Local vs Global Rendering

- Correct shading requires a global calculation involving all objects and light sources
  - Incompatible with pipeline model which shades each polygon independently (local rendering)

- However, in computer graphics, especially real time graphics, we are happy if things “look right”
  - Exist many techniques for approximating global effects

Light-Material Interaction

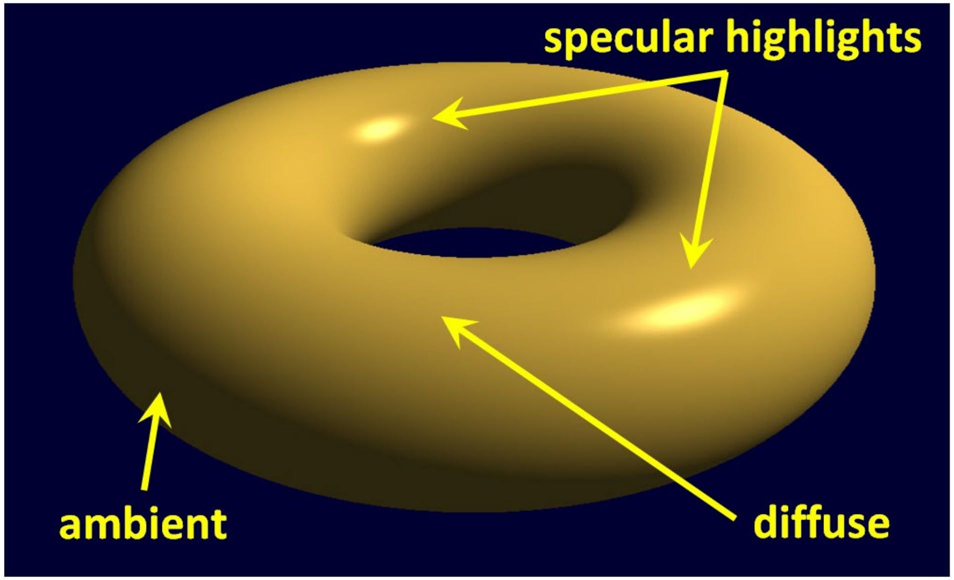
- Light that strikes an object is partially absorbed and partially scattered (reflected)
- The amount reflected determines the color and brightness of the object
  - A surface appears red under white light because the red component of the light is *reflected* and the rest is absorbed
- The reflected light is scattered in a manner that depends on the smoothness and orientation of the surface

Lighting Models

- The equation that explains how light interacts with objects and forms the image we see is called a *lighting model*
- Also known as a *shading model* or a *reflection model*
- The most common lighting models today are called ADS models because they are based on ambient, diffuse, and specular reflection
- ADS models can be used to simulate different lighting effects and a variety of materials

The “ADS” Lighting Model

- **A**<u>mbient</u> reflection simulates a low-level illumination that equally affects everything in the scene.
- **D**<u>iffuse</u> reflection brightens objects to various degree depending on the light’s angle of incidence.
- **S**<u>pecular</u> reflection conveys the shininess of an object by strategically placing a highlight of appropriate size on the object’s surface where light is reflected most directly towards our eyes.



The “ADS” Lighting Model

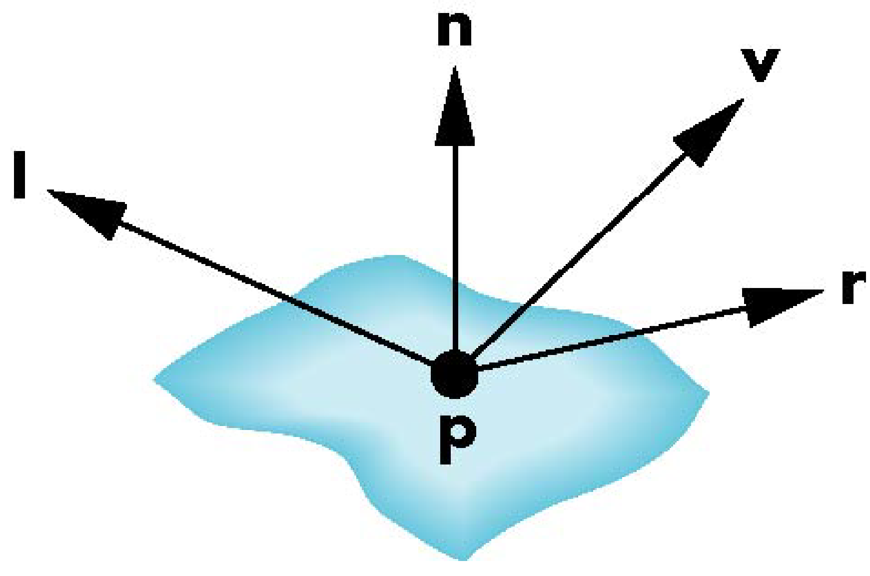
Using an ADS lighting model requires specifying contributions due to lighting on a pixel’s RGBA output value. Factors include:

- The type of light source, and its ambient, diffuse, and specular characteristics
- The object’s material’s ambient, diffuse, and specular characteristics
- The object’s material’s specified “shininess”
- The angle at which the light hits the object
- The angle from which the scene is being viewed

ADS lighting model is more commonly known as the Phong Illumination model or Phong Reflection Model

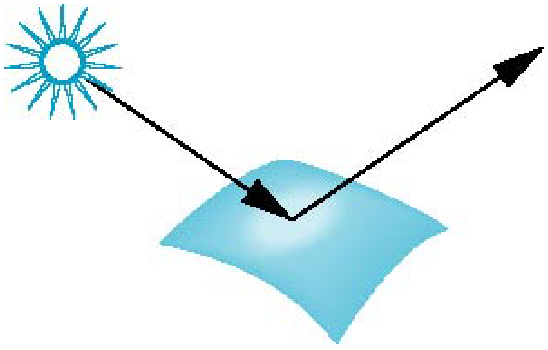
Phong Illumination Model

- A simple model that can be computed rapidly
- Has three components
  - Diffuse
  - Specular
  - Ambient
- Uses four vectors
  - To light source
  - To viewer
  - Normal
  - Perfect reflector

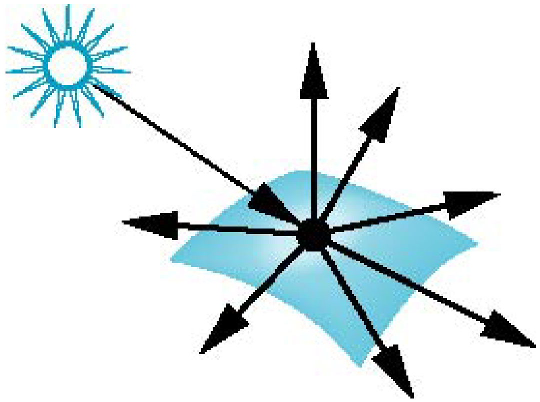


Surface Types

- The smoother a surface, the more reflected light is concentrated in the direction a perfect mirror would reflect the light
- A very rough surface scatters light in all directions



smooth surface



rough (Lambertian) surface

Types of Light Sources

- Global (or “global ambient”)
- Directional (or “distant”)
- Positional (or “point source”)
- Distributed (e.g., long fluorescent light tube)
- Spotlight

Global Ambient Light

- Models the low-level glow that reaches everywhere
- Equal everywhere
- Simplest type of light to model
- Simulates the real-world phenomenon of light that has bounced around so many times that its source and direction are undeterminable

Example:

```
float globalAmbient[4] = { 0.6, 0.6f, 0.6f, 1.0f };
```

Directional Light

- Doesn't have a source location, but has a direction
- Useful when light source is so far away that light rays are effectively parallel (e.g., sun light)
- Effect on object depends on the light's angle of impact
- Objects are brighter on the side facing a directional light than on a tangential or opposite side

Example of a red directional light pointing down the –z axis:

```
float dirLightAmbient[4] = { 0.1f, 0.0f, 0.0f, 1.0f };
float dirLightDiffuse[4] = { 1.0f, 0.0f, 0.0f, 1.0f };
float dirLightSpecular[4] = { 1.0f,0.0f, 0.0f, 1.0f };
float dirLightDirection[3] = { 0.0f, 0.0f, -1.0f };
```

Positional (or "Point") Light

- Models nearby light sources, such as a lamp
- Has a source location, but not a direction
- Effect on object depends on the light's angle of impact
- May include optional attenuation factor to model how intensity diminishes with distance
- The constant term should be greater or equal to 1, and one other parameter > 0 for factor to be in [0..1] range

$$attenuationFactor = \frac{1}{k_c + k_l d + k_q d^2}$$

Example of a red positional light at location (5, 2, -3):

```
float posLightAmbient[4] = { 0.1f, 0.0f, 0.0f, 1.0f };
float posLightDiffuse[4] = { 1.0f, 0.0f, 0.0f, 1.0f };
float posLightSpecular[4] = { 1.0f,0.0f, 0.0f, 1.0f };
float posLightLocation[3] = { 5.0f, 2.0f, -3.0f };
```

Spotlight

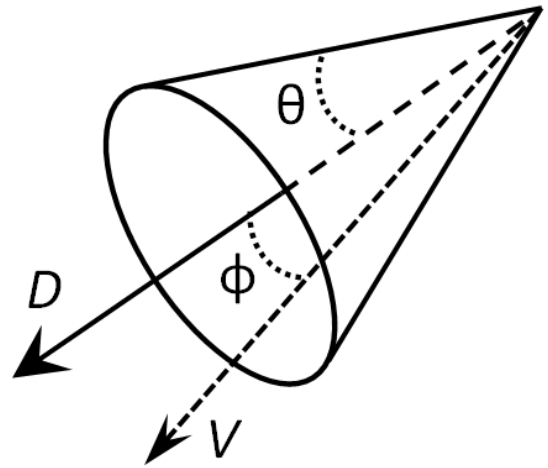
D = spotlight direction

V = direction to vertex

θ = cutoff angle

ϕ = light off-axis angle

intensityFactor = $\cos^{exp}(\phi)$



- Combines elements of positional and directional lights
- Beam is formed by half-width cutoff angle of cone
- Falloff exponent models variation of intensity across beam
- Iconic since Pixar’s animated short “Luxo, Jr.” in 1986

Spotlight

Example – red spotlight at (5,2,-3) pointing down –Z axis:

```
float spotLightAmbient[4] = { 0.1f, 0.0f, 0.0f, 1.0f };
float spotLightDiffuse[4] = { 1.0f, 0.0f, 0.0f, 1.0f };
float spotLightSpecular[4] = { 1.0f,0.0f, 0.0f, 1.0f };
float spotLightLocation[3] = { 5.0f, 2.0f, -3.0f };
float spotLightDirection[3] = { 0.0f, 0.0f, -1.0f };
float spotLightCutoff = 20.0f;
float spotLightExponent = 10.0f;
```

Materials

- Models the reflectance characteristics of surfaces
- The visible color of material is the product of the incident light and the intrinsic material color
- Usually modeled in ADS (Phong) with four terms, with the ADS terms having RGB components:
  - Ambient
  - Diffuse
  - Specular
  - Shininess (to determine size of specular highlights)

Example (for "pewter"):

```
float pewterMatAmbient[4] = { .11f, .06f, .11f, 1.0f };
float pewterMatDiffuse[4] = { .43f, .47f, .54f, 1.0f };
float pewterMatSpecular[4] = { .33f, .33f, .52f, 1.0f };
float pewterMatShininess = 9.85f;
```

Some Common Materials

material	*ambient RGBA* *diffuse RGBA* *specular RGBA*	*shininess*
Gold	0.2473, 0.1995, 0.0745, 1.0 0.7516, 0.6065, 0.2265, 1.0 0.6283, 0.5558, 0.3661, 1.0	51.200
Jade	0.1350, 0.2225, 0.1575, 0.95 0.5400, 0.8900, 0.6300, 0.95 0.3162, 0.3162, 0.3162, 0.95	12.800
Pearl	0.2500, 0.2073, 0.2073, 0.922 1.0000, 0.8290, 0.8290, 0.922 0.2966, 0.2966, 0.2966, 0.922	11.264
Silver	0.1923, 0.1923, 0.1923, 1.0 0.5075, 0.5075, 0.5075, 1.0 0.5083, 0.5083, 0.5083, 1.0	51.200

Barradeu, N., http://www.barradeau.com/nicoptere/dump/materials.html

ADS Lighting Computations

$$I_{observed} = I_{ambient} + I_{diffuse} + I_{specular}$$

- The basic ADS computation we need to perform is to determine the reflection intensity for each pixel.
- We compute the sum of the ambient, diffuse, and specular reflection contributions for each pixel, for each light source

Ambient Lighting Computation

<u>Ambient</u> computation is the simplest:

$$I_{ambient} = Light_{\ ambient} * Material_{ambient}$$

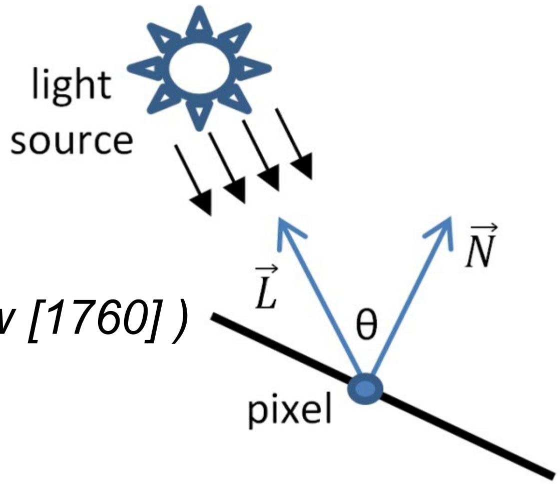
Note that each item has R, G, and B components.
So the computations actually are as follows:

$$I_{ambient}^{red} = Light_{ambient}^{red} * Material_{ambient}^{red}$$

$$I_{ambient}^{green} = Light_{ambient}^{green} * Material_{ambient}^{green}$$

$$I_{ambient}^{blue} = Light_{ambient}^{blue} * Material_{ambient}^{blue}$$

Diffuse Lighting Computation



Diffuse computation depends on the angle of incidence between the light and the surface:

(Lambert's cosine law [1760])

$$I_{diffuse} = Light_{diffuse} * Material_{diffuse} * c$$

Rightmost term determined simply using dot product:

$$I_{diffuse} = Light_{\ diffuse} * Material_{diffuse} * (\hat{N} \bullet \hat{L})$$

Only include this term if the surface is exposed to the light:

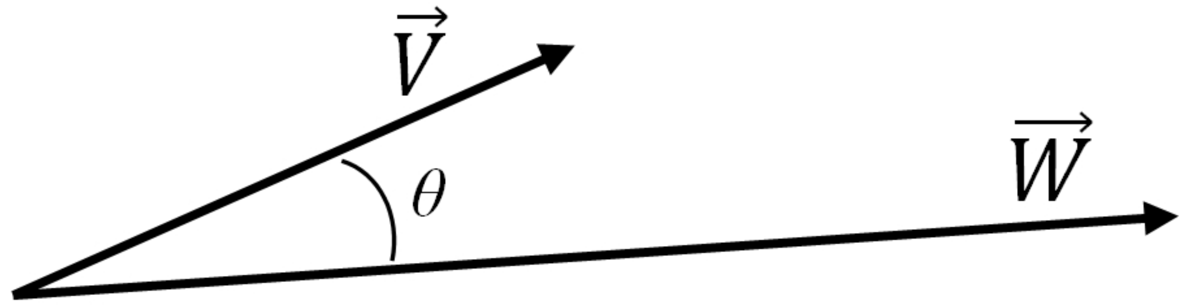
$$I_{diffuse} = Light_{\ diffuse} * Material_{diffuse} * \max((\hat{N} \bullet \hat{L}), 0)$$

As before, the diffuse material reflection coefficients have three components (RGB)

Lambertian Surface

- Perfectly diffuse reflector
- Light scatters equally in all directions
- Amount of light reflected is proportional to the vertical component of incoming light
  - reflected light ~$\cos\theta_i$
  - $\cos\theta_i = \mathbf{l} \cdot \mathbf{n}$ if vectors normalized
  - There are three coefficients, k_r, k_g, k_b that show how much of each color component is reflected
  - These terms comprise $Material_{diffuse}$
  - They are multiplied with the respective RGB components of the light

Dot Product



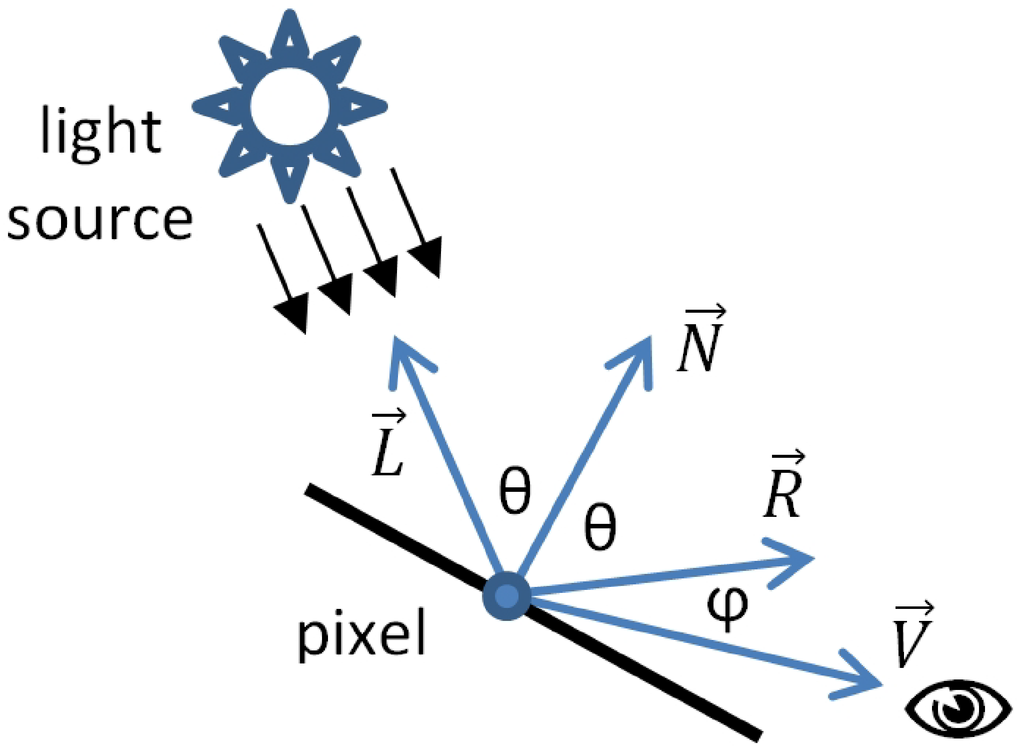
assuming V=(a,b,c) and W=(d,e,f),
V ● W = ad + be + cf

$$\vec{V} \bullet \vec{W} = \left|\vec{V}\right| * \left|\vec{W}\right| * \cos(\theta)$$

$$\theta = \arccos(\hat{V} \bullet \hat{W})$$

Specular Lighting Computation

<u>Specular </u>computation depends on the angle of reflection of the light on the surface, and the viewing angle of the eye.



Modeling Specular Reflections

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

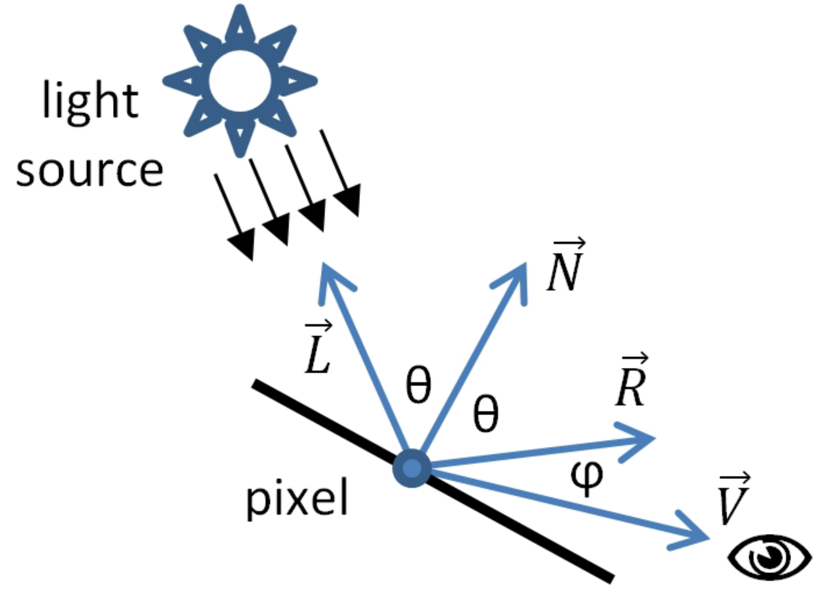
$$\mathbf{I}_r \sim k_s\, \mathbf{I} \cos^{\alpha}\phi$$

reflected intensity

absorption coef

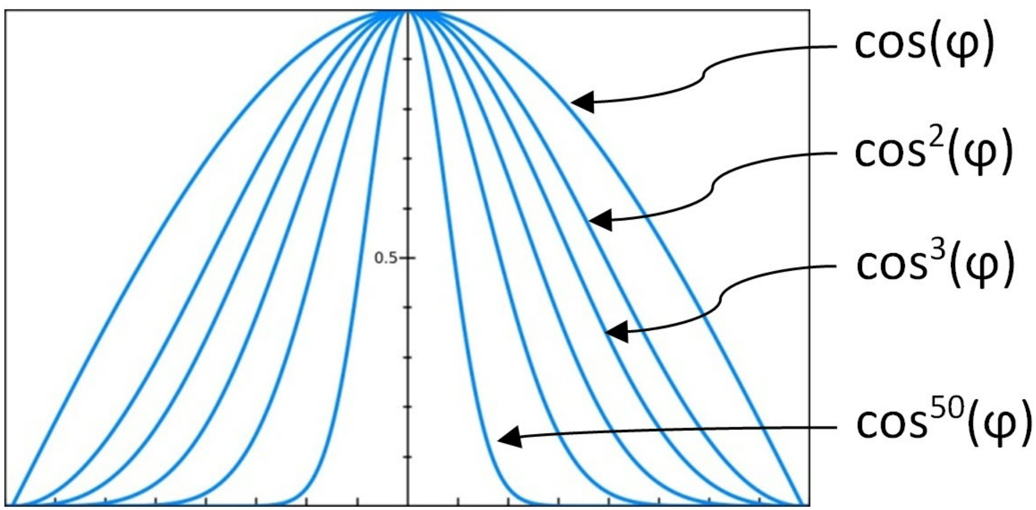
incoming intensity

shininess coef



Shininess

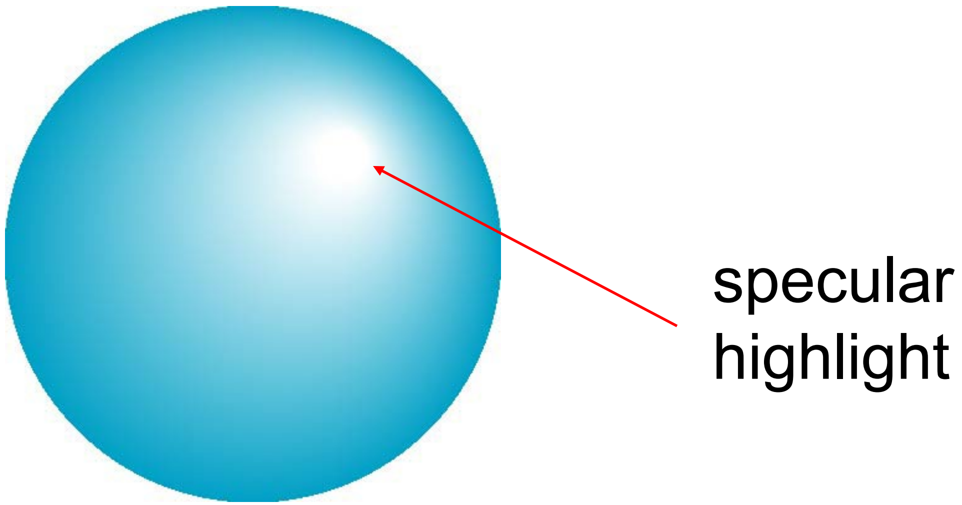
- “Shininess” modeled with a falloff function
- Expresses how quickly the specular contribution reduces to zero as the angle ϕ grows
- Exponents between 100 and 200 correspond to metals
- Exponents between 5 and 10 correspond to plastic look



$$I_{spec} = Light_{spec} * Material_{spec} * \max(0, (\hat{R} \bullet \hat{V})^{n})$$

Specular Surfaces

- Most surfaces are neither ideal diffusers nor perfectly specular (ideal reflectors)
- Smooth surfaces show specular highlights due to incoming light being reflected in directions concentrated close to the direction of a perfect reflection



Light Sources

- In the Phong (ADS) Illumination model, we add the results from each light source
- Each light source has separate diffuse, specular, and ambient terms to allow for maximum flexibility even though this form does not have a physical justification
- Separate red, green and blue components
- Hence, 9 coefficients for each point source
  - I_{dr}, I_{dg}, I_{db}, I_{sr}, I_{sg}, I_{sb}, I_{ar}, I_{ag}, I_{ab}

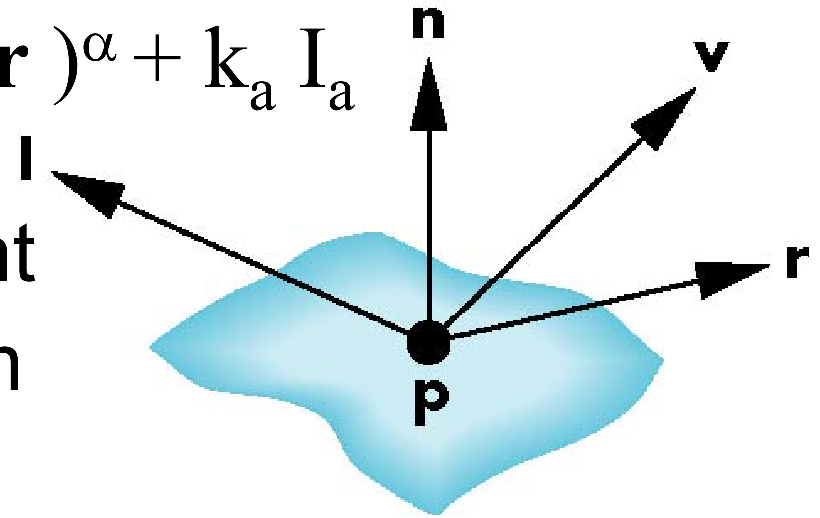
Material Properties

- Material properties match light source properties
  - Nine absorption coefficients
    - k_{dr}, k_{dg}, k_{db}, k_{sr}, k_{sg}, k_{sb}, k_{ar}, k_{ag}, k_{ab}
  - Shininess coefficient α

Adding up the Components

For each light source and each color component, the Phong (ADS) model can be written (without the distance terms) as

$$I = k_d I_d \; \mathbf{l} \cdot \mathbf{n} \; + k_s I_s (\mathbf{v} \cdot \mathbf{r})^{\alpha} + k_a I_a$$

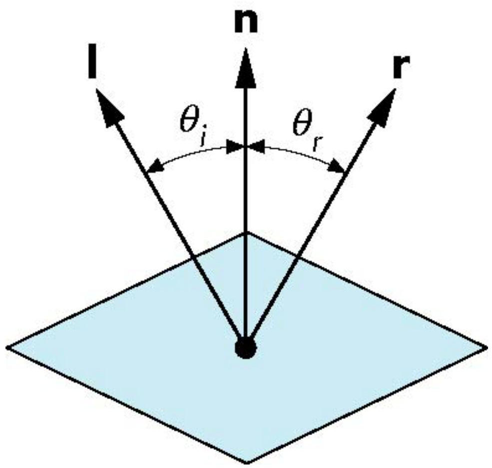


For each color component we add contributions from all sources.

Computing Reflection Vector

- Surface normal is determined by local orientation
- Angle of incidence = angle of reflection
- Normal, light, and reflection vectors must be coplanar
- We want all three to be unit length

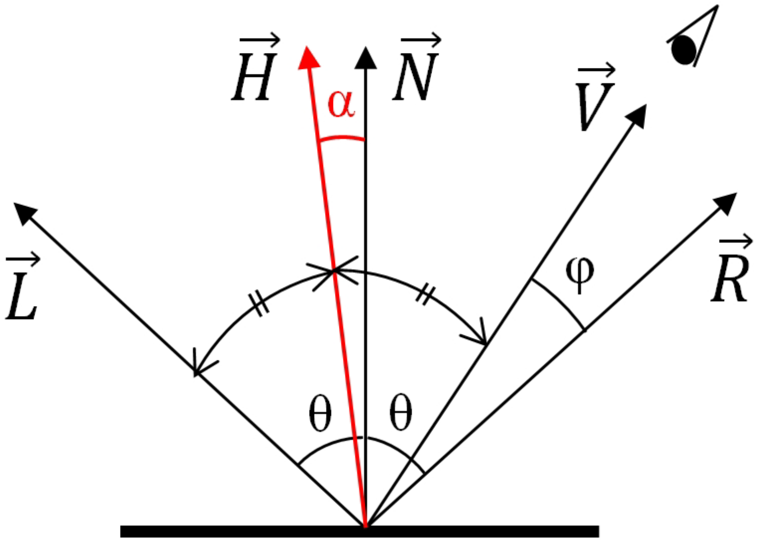
$$\mathbf{r} = 2\,(\mathbf{l} \cdot \mathbf{n}\,)\,\mathbf{n} - \mathbf{l}$$

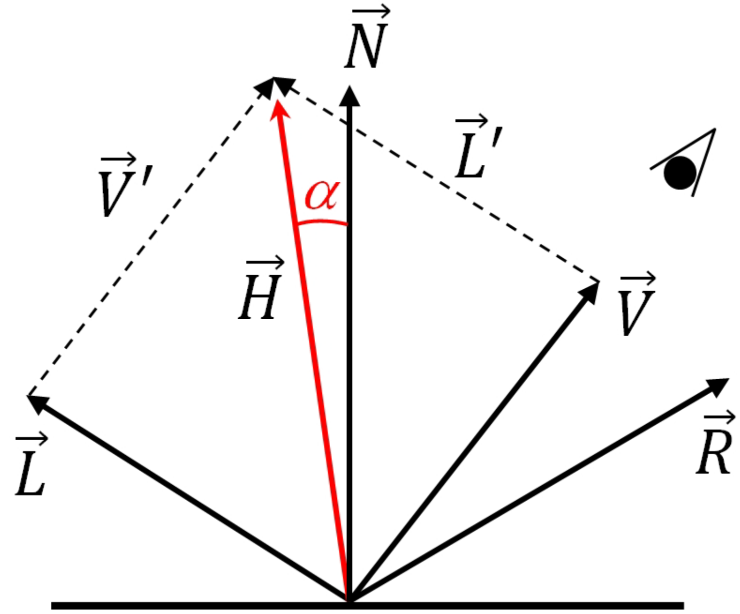


Modified Phong Model

- The specular term in the Phong model is problematic because it requires the calculation of a new reflection vector **r** for each vertex
- Blinn observed that **r** itself is not needed. It is only produced as a means of determining the angle ϕ between **v** and **r**.
- Blinn suggested an efficient approximation using the halfway vector **h** that is halfway between **l** and **v**
- He found that the angle α between **h** and **n** is approximately half of ϕ. Use α instead of ϕ
- Known as the *Blinn-Phong reflection model*

Blinn-Phong Reflection Model





Desired: φ

Can be estimated by finding α --

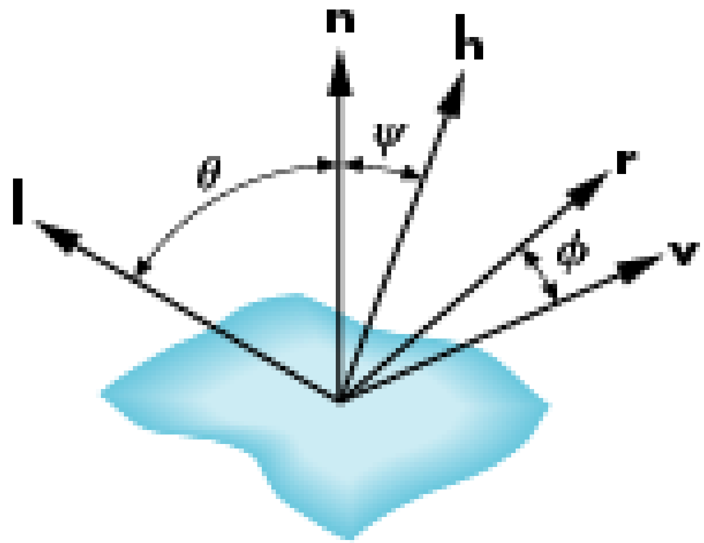
$$\alpha = \tfrac{1}{2}\varphi$$

Conveniently, $\vec{\mathrm{H}}$ is easier to compute: $\vec{H} = \vec{L} + \vec{V}$

The Halfway Vector

- **h** is normalized vector halfway between **l** and **v**

$$\mathbf{h} = (\mathbf{l} + \mathbf{v})/\,|\,\mathbf{l} + \mathbf{v}\,|$$



Using the Halfway Vector

- Replace $(\mathbf{v} \cdot \mathbf{r})^{\alpha}$ by $(\mathbf{n} \cdot \mathbf{h})^{\beta}$
- β is chosen to match shininess
- Note that halfway angle is half of angle between $\mathbf{r}$ and $\mathbf{v}$ if vectors are coplanar
- Resulting model is known as the modified Phong or Blinn-Phong lighting model
  - Specified in OpenGL standard

Example

Only differences in
these teapots are
the parameters
in the Phong model