

Shading and Lighting

- Shading models

- Flat
- Gouraud (smooth)
- Phong

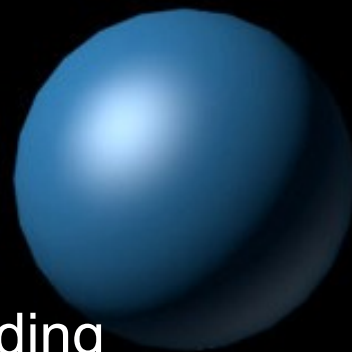
- Lighting models

- Phong
- Blinn-Phong
 - per-vertex implementation
 - per-pixel implementation

Shading

- Flat shading
 - Use triangle face normals for lighting
 - Solid color per triangle
 - `glShadeModel(GL_FLAT);`
- Gouraud
 - Use vertex normal for lighting
 - Color computed for each vertex, interpolated over triangle
 - `glShadeModel(GL_SMOOTH);`
- Phong
 - Interpolate parameters over triangle, compute color in fragment shader
 - Color computed per fragment

Shading

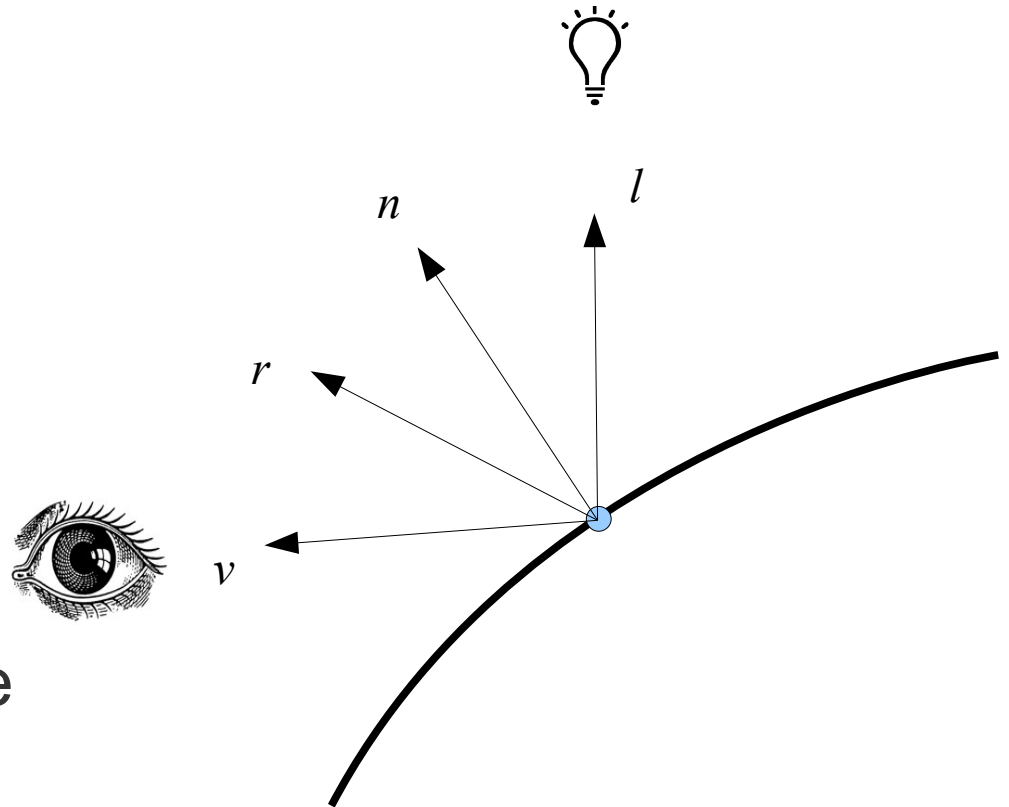


Flat, Gouraud, Phong Shading

Local lighting model

- Local light models compute appearance based on relations between each point on the surface, with the viewer, and light sources
- No volumetric scattering, no inter-reflections between objects, no shadows.

- v , view vector
- n , normal vector
- l , light vector
- r , reflection vector
- Unit vectors
- Pointing away from surface



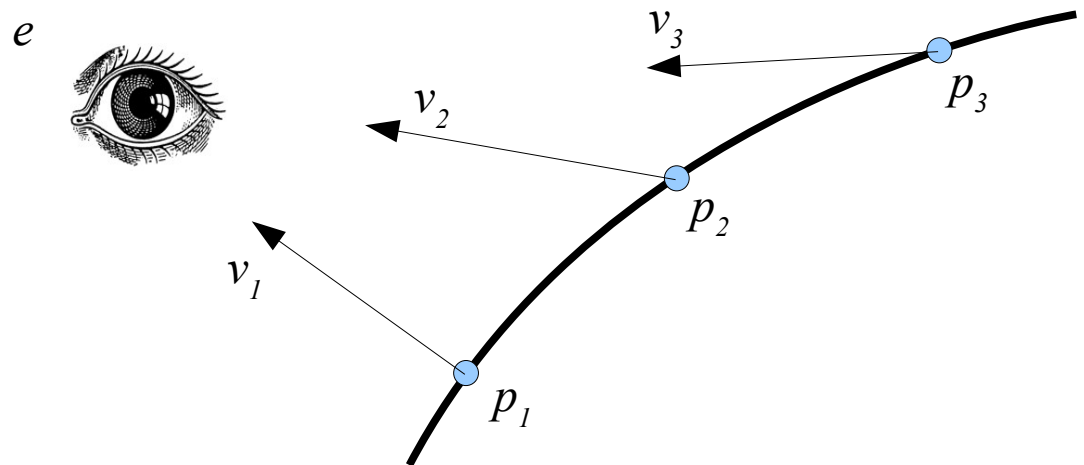
Viewer models

- Infinite viewer

- $v = -look$
- Faster since view vector is the same for all vertices
- `glLightModeli(GL_LIGHT_MODEL_LOCAL_VIEWER, GL_FALSE);`

- Local viewer

- $e = \text{eye position}, p = \text{point on surface}$
- $v = (e-p)/\|e-p\|$
- `glLightModeli(GL_LIGHT_MODEL_LOCAL_VIEWER, GL_TRUE);`



Light Sources

- Directional Light
 - Model of a light infinitely far away
 - The sun can be approximated at a directional light
 - Light vector, l , is constant
- Point Light
 - Characterized by position
 - Light vector must be computed for each computation

Coordinate system for lighting

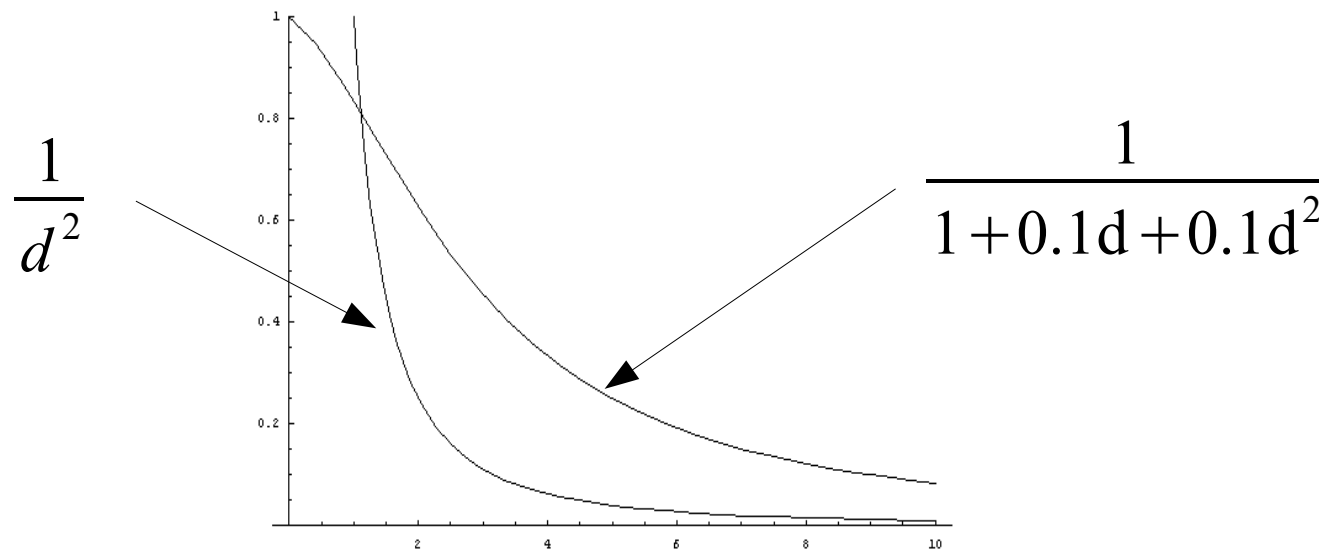
- We need all vectors to be in the same coordinate system in order to compute lighting
- Some choices:
 - Object local coordinate system
 - Tangent space
 - World coordinate system
 - Eye coordinate system

Which coordinate system to use?

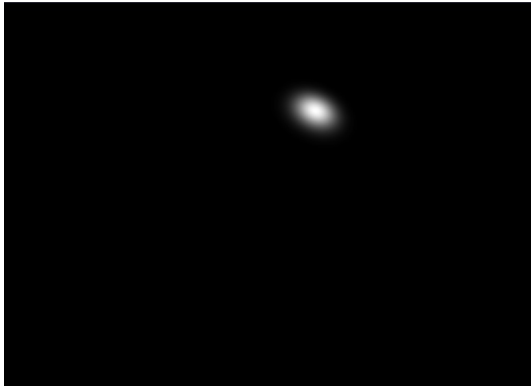
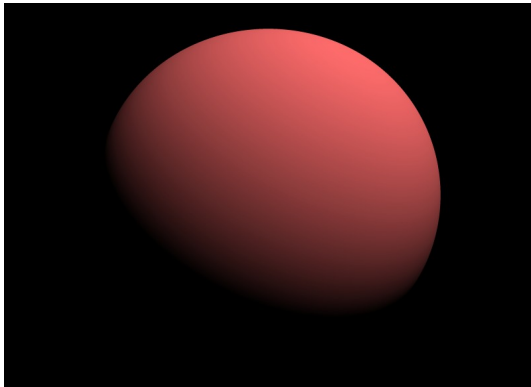
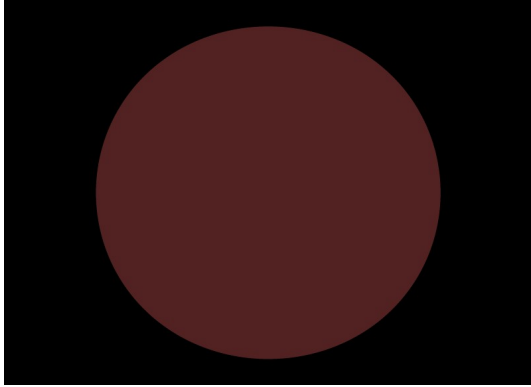
- Object local coordinate system
 - Transform lights using inverse of object modeling transformation (per-object)
- Tangent space
 - Transform lights into tangent space (per-vertex or per-fragment)
 - Tangent space normals are constant
 - Useful in normal mapping
- World coordinate system
 - Transform vertices and normals using object modeling transformation
- Eye coordinate system
 - Transform lights using view matrix
 - Transform normals using normal matrix
 - ModelView matrix inverse transpose

Light Attenuation

- Physically accurate attenuation
 - Attenuation factor = $1/d^2$
 - d is distance from light to surface
 - In practice this is very harsh – falls off too quickly
- Quadratic Attenuation with softer fall off
 - Attenuation factor = $1/(a + bd + cd^2)$

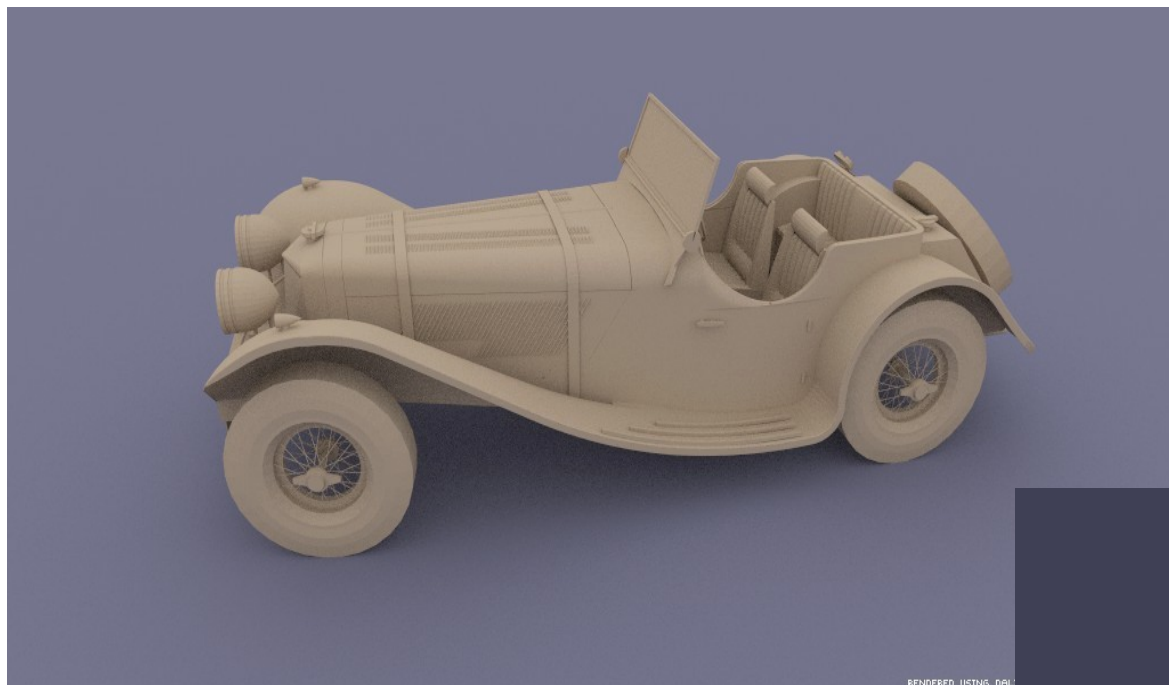


Phong Lighting



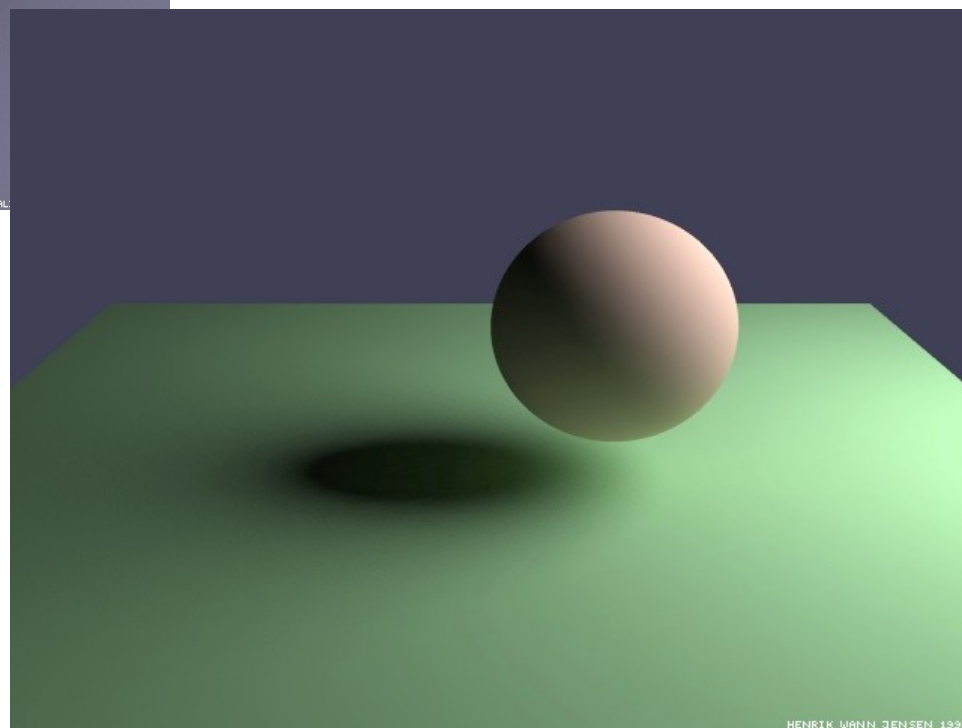
- Three terms
 - Ambient
 - Constant color
 - Bad approximation of global illumination
 - Diffuse
 - Lambertian model
 - $\max(n \cdot l, 0)$
 - Independent of view direction
 - Dependent on angle between incoming rays and surface normal
 - Specular
 - Gives highlights
 - $\max((r \cdot v)^a, 0)$
 - a : shininess

Global Illumination Effects



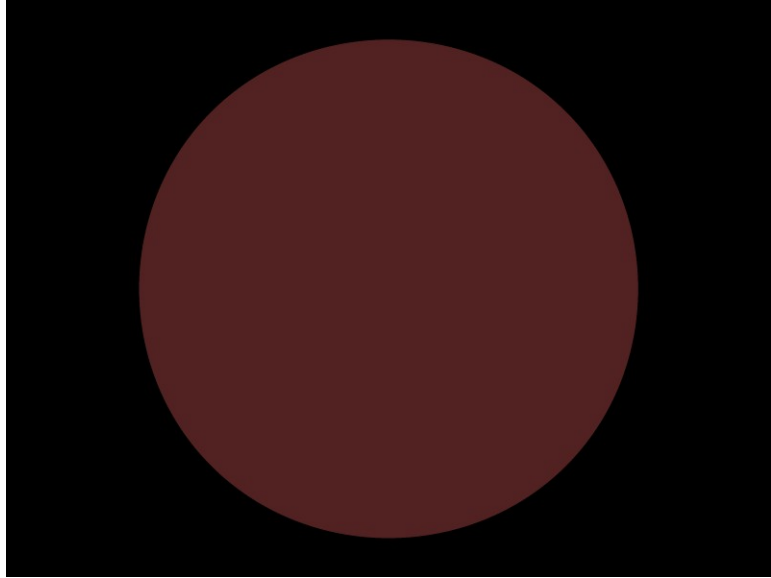
RENDERED USING DAL

- Interreflections
- Shadows
- These effects ignored or greatly simplified



HENRIK WANN JENSEN 1998

Phong Ambient Term

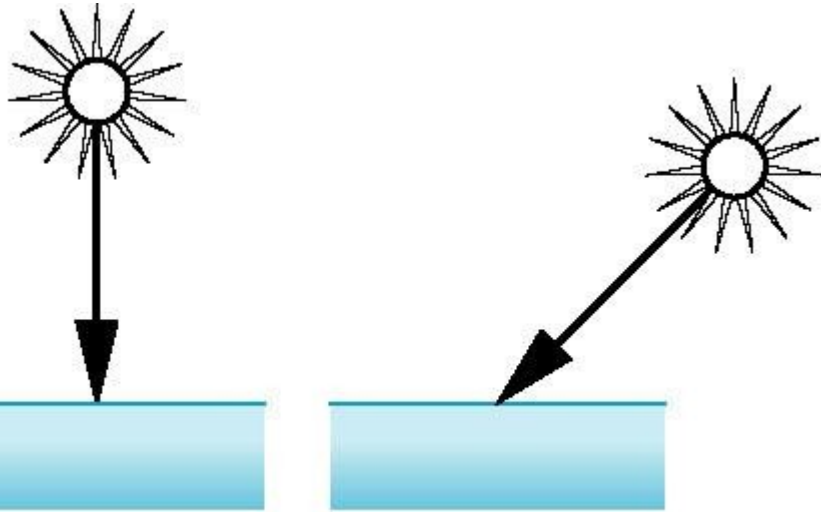


- Simulated indirect lighting
- Does not depend on position or normal

$$I_a = k_a L_a$$

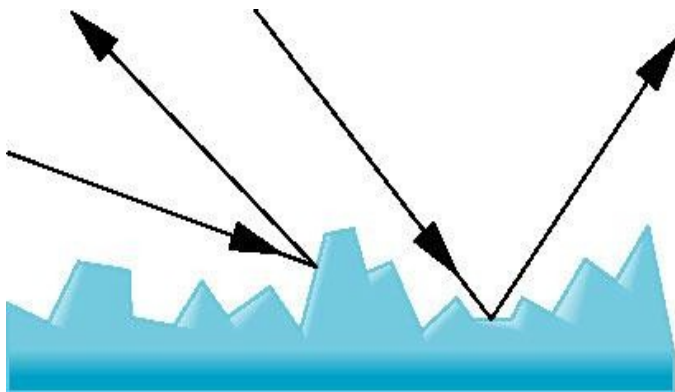
- k_a : material ambient color
- L_a : light ambient color
- I_a : intensity at surface

Lambertian reflection



Light emitted from source → light received at surface

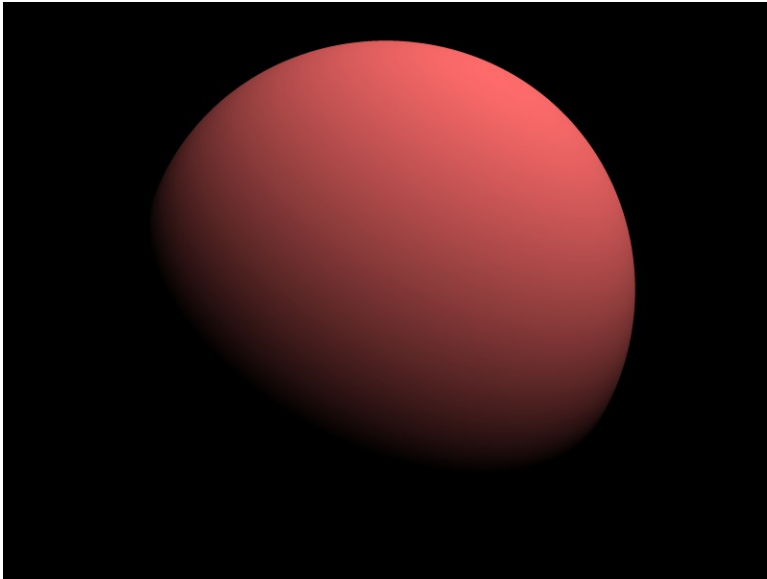
- Depends on cosine of angle between normal and light direction
- Models how light spreads over area



Light received at surface → light reflected from surface

- Does not depend on view direction
- Uniform distribution of 'microfacet' normal vectors

Phong diffuse term

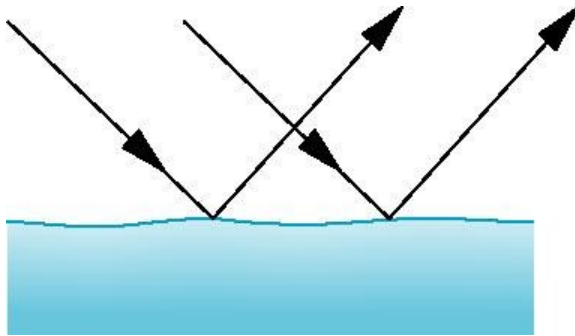
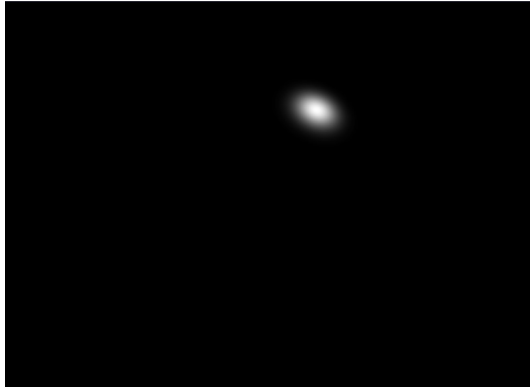


- Attenuated Lambertian reflection

$$I_d = \frac{k_d}{a + bd + cd^2} \max(n \cdot l, 0) L_d$$

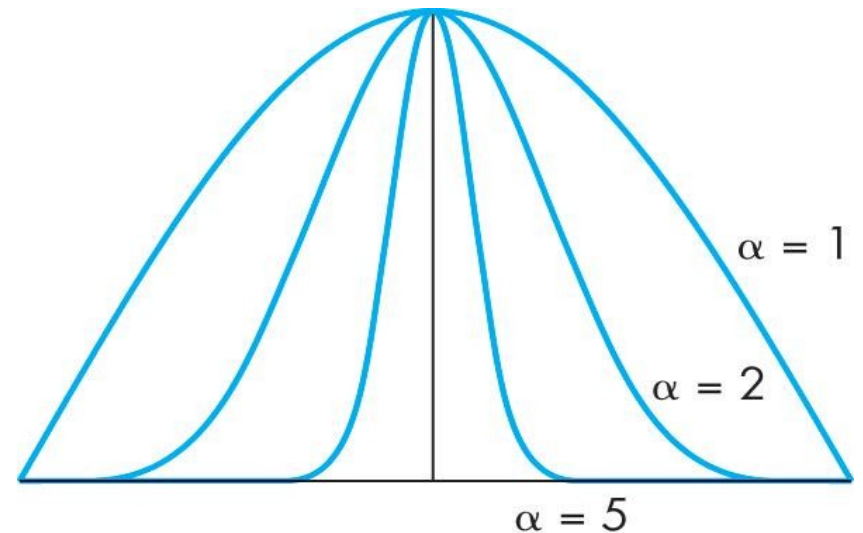
- k_d : material ambient color
- L_d : light ambient color
- I_d : intensity at surface

Phong Specular Term

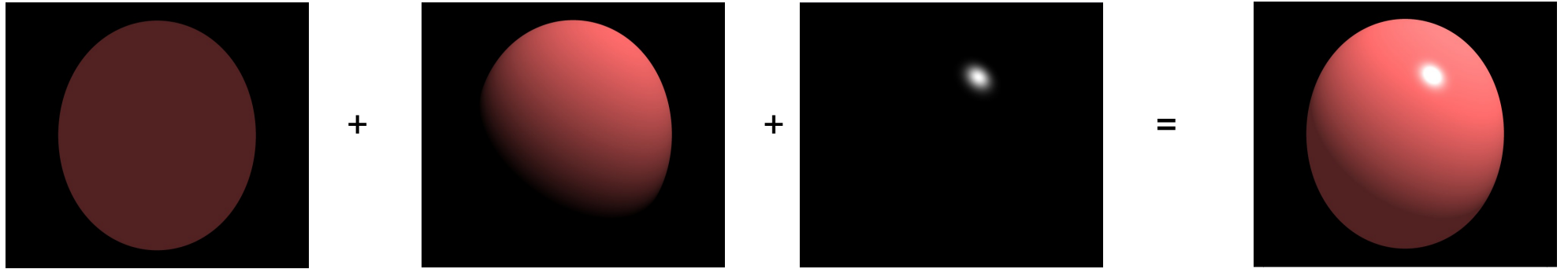


- Not physically based
- Reflection direction is geometrically accurate
- Equation simply generates a plausible highlight about the reflection direction
- α is shininess

$$I_s = \frac{k_s}{a + bd + cd^2} \max((r \cdot v)^\alpha, 0) L_s$$



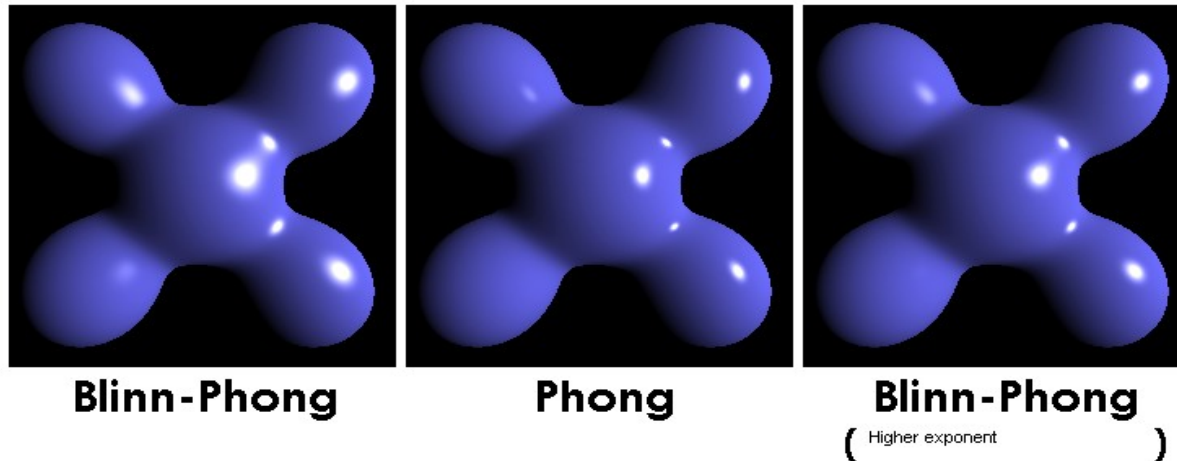
Phong lighting equation



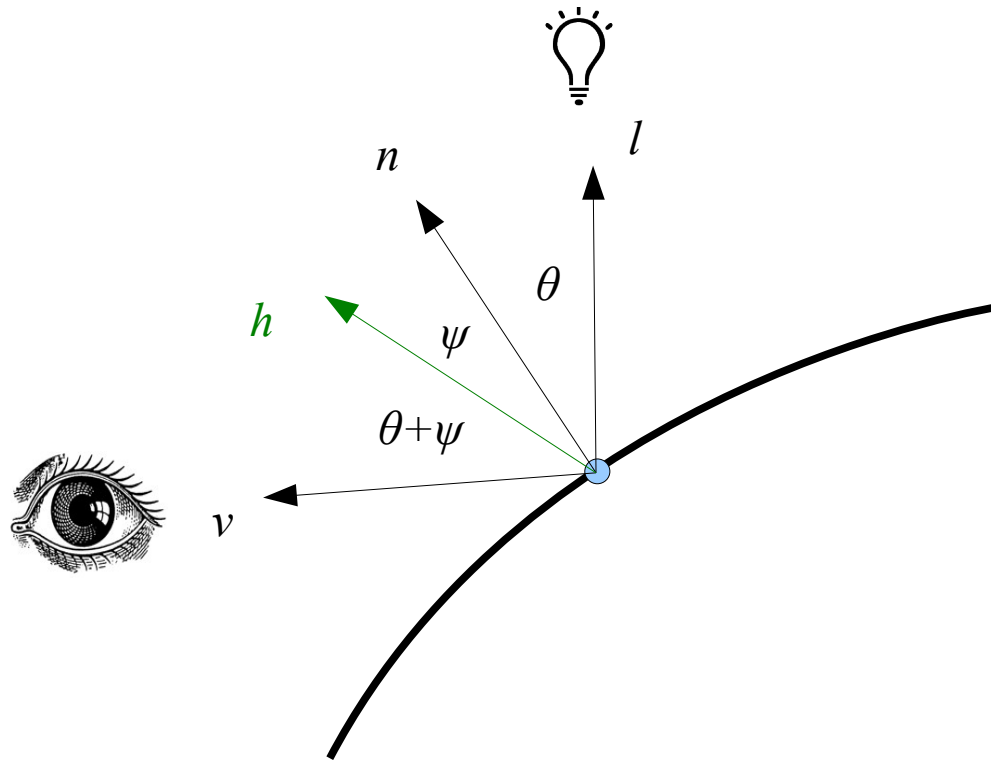
$$I = I_a k_a + \frac{1}{a + bd + cd^2} (k_d L_d \max(n \cdot l, 0) + k_s L_s \max((r \cdot v)^\alpha, 0))$$

Blinn-Phong

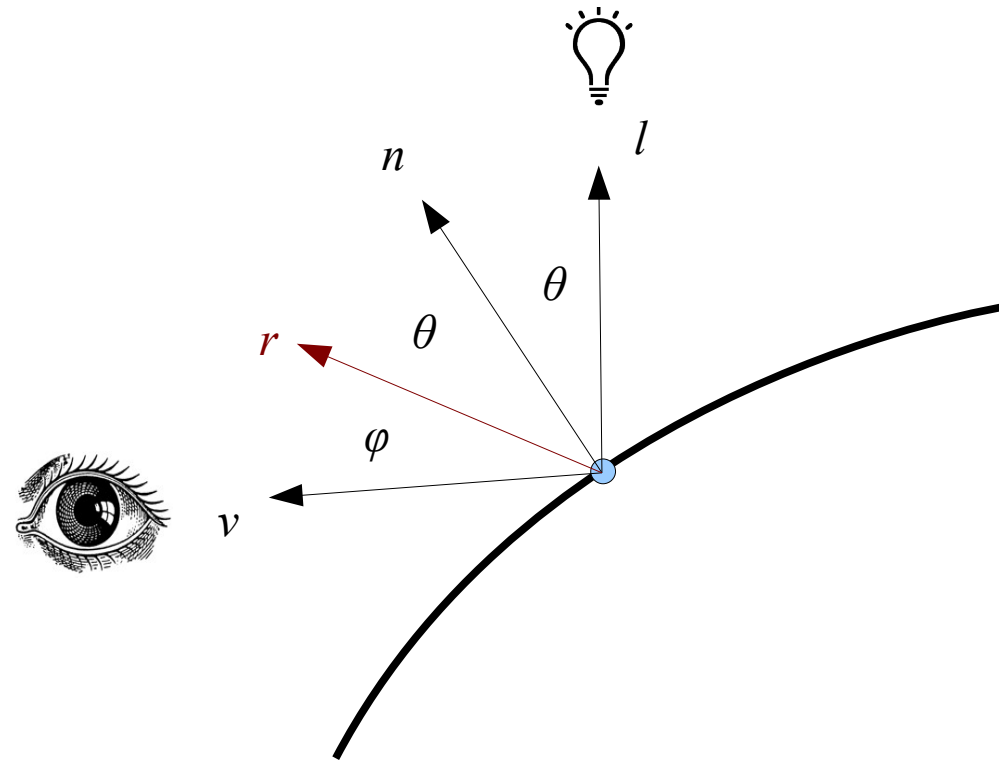
- Modified the Phong model so that the specular term is **estimated** quickly
- Less precise, but faster
 - Halfway vector: $h = \frac{1}{2}(l+v)$
 - *Replace $(r \cdot v)^\alpha$ with $(h \cdot n)^{\alpha'}$*
- This is the lighting model provided by OpenGL fixed-function pipeline.



When all vectors lie in a plane



- θ : angle between n, l
- ψ : angle between n, h
- $\theta + \psi$: angle between h, l
- Since (by def.) h is halfway between l and v , the angle between h, v is also $\theta + \psi$
- The total angle between l, v is $2(\theta + \psi)$



- θ : angle between n, l
- By ideal reflection law the angle between n and r is also θ .
- ϕ : angle between r, v
- The total angle between l, v is $2\theta + \phi$
- Angle between n, h is prop to angle between r, v
- $2\psi = \phi$