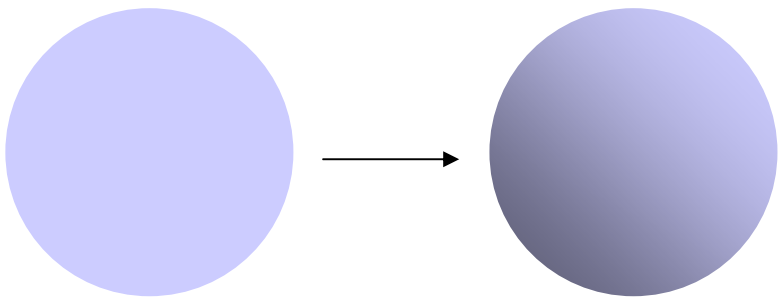
Shading

Materials and the lighting model



Reading Material

MUST read

- These slides
- OH 92-101 by Magnus Bondesson
  - Shading model, materials, flat shading vs Gouraud shading
- Cook-Torrance Shading.pdf - downloadable on course home page
- Introduktion till OpenGL
  - Sections: 11 Färg, 22 Belysning, 28 Färgblandning

May also read:

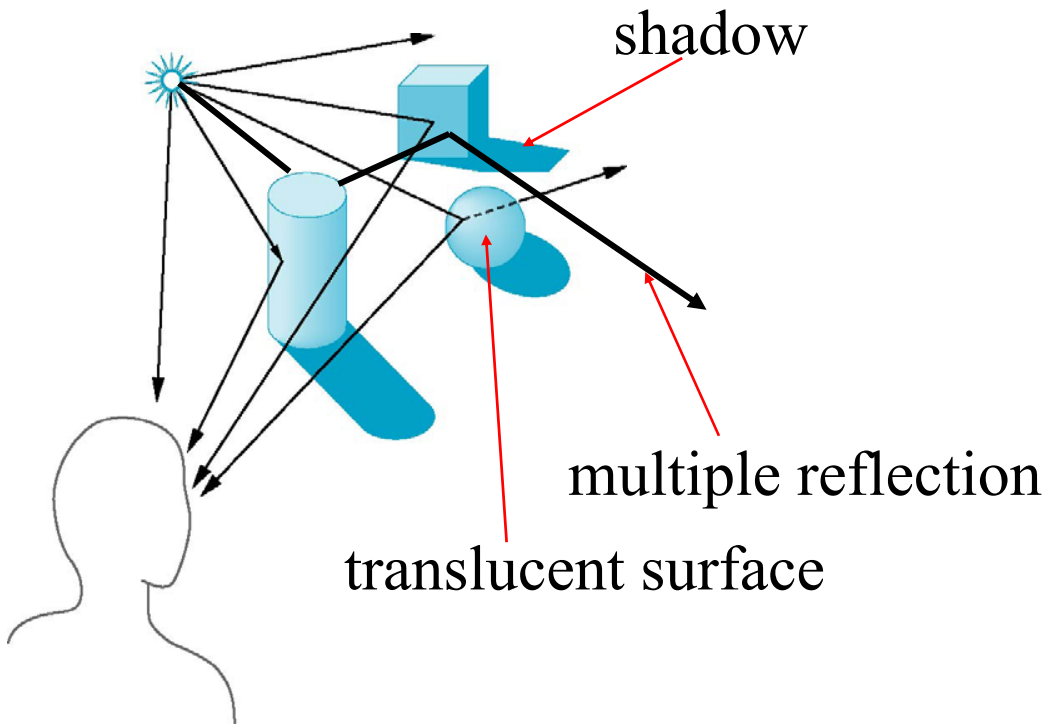
- Angel, chapter 6

Overview of today's lecture

- OpenGL's simple lighting model
- Fog
- Gamma correction
- Transparency and alpha

In Reality

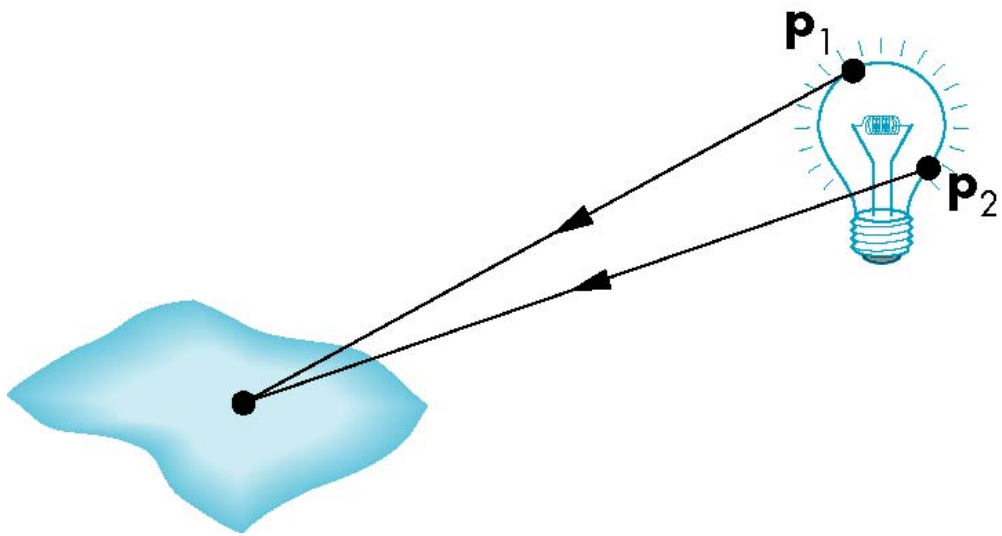
- Correct shading requires a global calculation involving all objects and light sources
  - Incompatible with pipeline model which shades each polygon independently (local rendering)
- However, in computer graphics, especially real time graphics, we are happy if things "look right"
  - Exist many techniques for approximating global effects



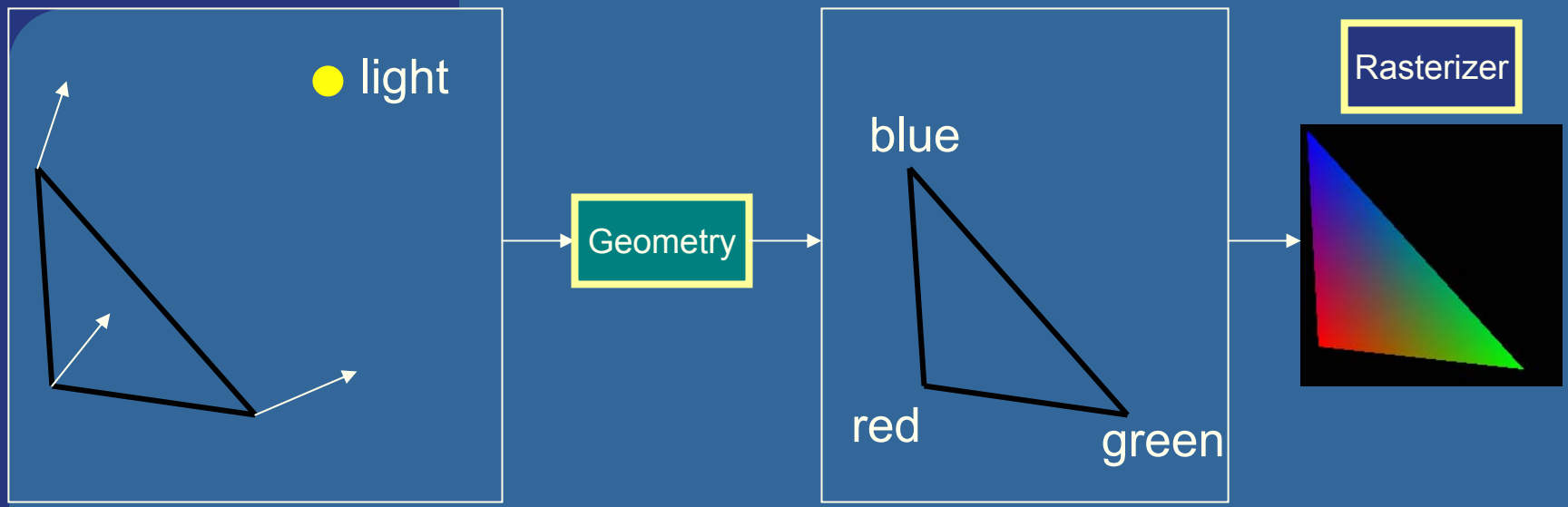
Still too expensive for real-time applications

Light Sources

General light sources are difficult to work with because we must integrate light coming from all points on the source

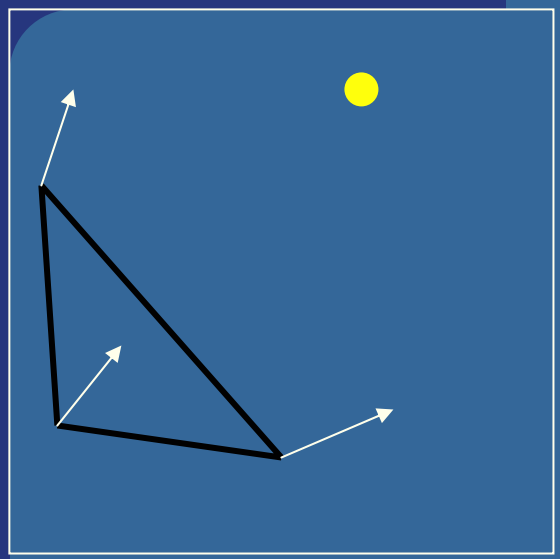


Compute lighting at vertices, then interpolate over triangle



- How compute lighting?
- We could set colors per vertex manually
- For a **little** more realism, compute lighting from
  - Light sources
  - Material properties
  - Geometrical relationships

Lighting



Light:

- Ambient (r,g,b,a)
- Diffuse (r,g,b,a)
- Specular (r,g,b,a)

Material:

- Ambient (r,g,b,a)
- Diffuse (r,g,b,a)
- Specular (r,g,b,a)
- Emission (r,g,b,a) =”självlysande färg”

Material Properties

- Material properties are also part of the OpenGL state
- Set by **`glMaterialv()`**

```
GLfloat ambient[] = {0.2, 0.2, 0.2, 1.0};
GLfloat diffuse[] = {1.0, 0.8, 0.0, 1.0};
GLfloat specular[] = {1.0, 1.0, 1.0, 1.0};
GLfloat shine = 100.0
glMaterialf(GL_FRONT, GL_AMBIENT, ambient);
glMaterialf(GL_FRONT, GL_DIFFUSE, diffuse);
glMaterialf(GL_FRONT, GL_SPECULAR, specular);
glMaterialf(GL_FRONT, GL_SHININESS, shine);
```

Defining a Point Light Source

- For each light source, we can set an RGBA for the diffuse, specular, and ambient components, and for the position

```
GL float diffuse0[]={1.0, 0.0, 0.0, 1.0};
GL float ambient0[]={1.0, 0.0, 0.0, 1.0};
GL float specular0[]={1.0, 0.0, 0.0, 1.0};
Glfloat light0_pos[]={1.0, 2.0, 3,0, 1.0};

glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
glLightv(GL_LIGHT0, GL_POSITION, light0_pos);
glLightv(GL_LIGHT0, GL_AMBIENT, ambient0);
glLightv(GL_LIGHT0, GL_DIFFUSE, diffuse0);
glLightv(GL_LIGHT0, GL_SPECULAR, specular0);
```

Ambient component: i_{amb}

- Ad-hoc – tries to account for light coming from other surfaces
- Just add a constant color:

$$\mathbf{i}_{amb} = \mathbf{m}_{amb} \otimes \mathbf{s}_{amb}$$

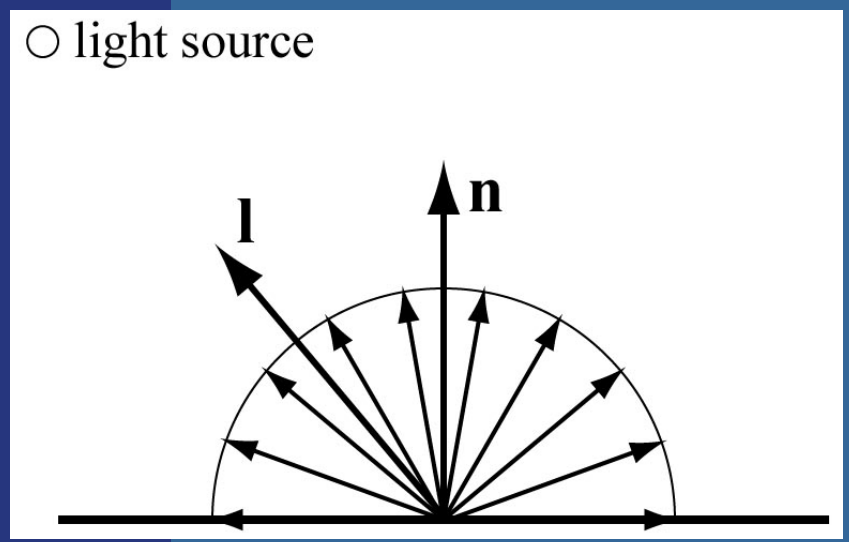
i.e., $(i_r, i_g, i_b, i_a) = (m_r, m_g, m_b, m_a)\ (l_r, l_g, l_b, l_a)$

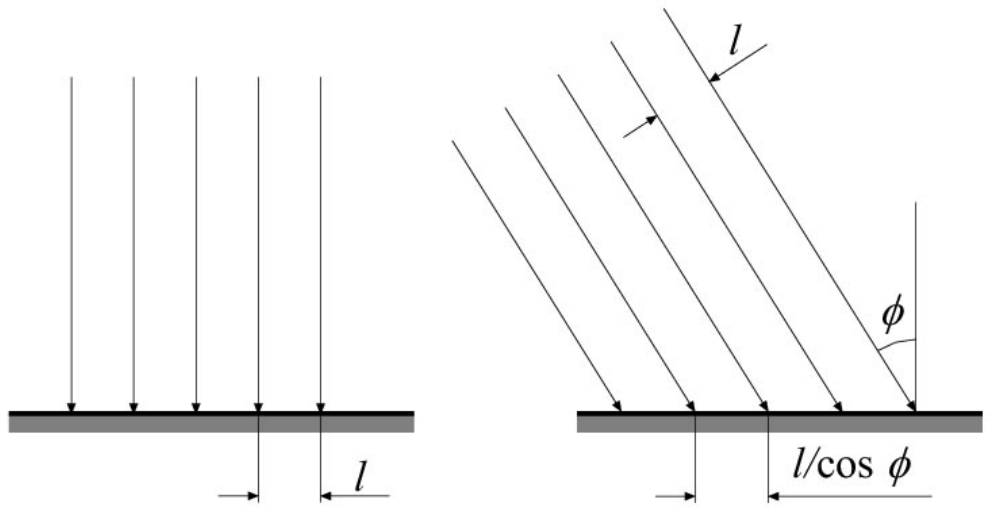
glLightModelfv(GL_LIGHT_MODEL_AMBIENT, global_ambient)

Diffuse component : i_{diff}

- $i = i_{amb} + i_{diff} + i_{spec}$
- Diffuse is Lambert's law: $i_{diff} = \mathbf{n} \cdot \mathbf{l} = \cos\phi$
- Photons are scattered equally in all directions

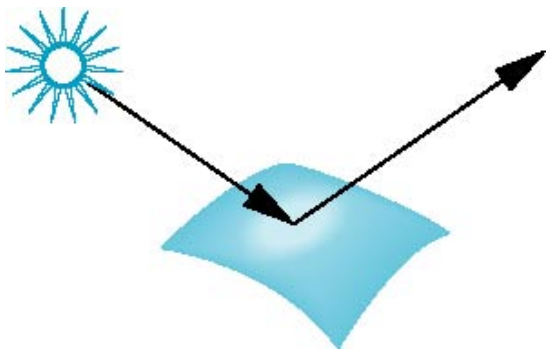
$$\mathbf{i}_{diff} = (\mathbf{n} \cdot \mathbf{l})\mathbf{m}_{diff} \otimes \mathbf{s}_{diff}$$



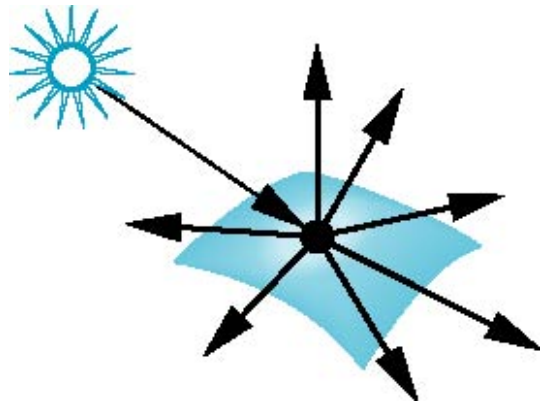


Lambertian Surfaces

- Perfectly diffuse reflector
- Light scattered equally in all directions



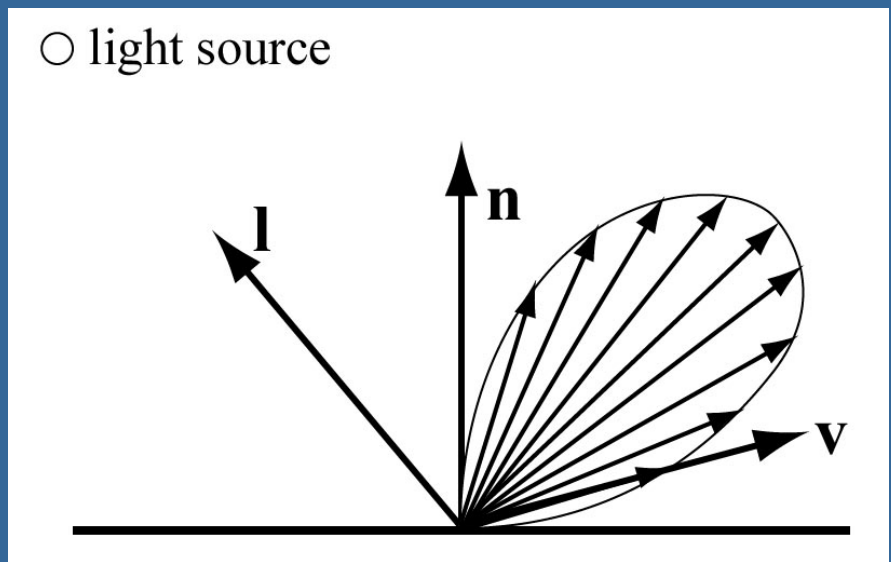
Highly reflective surface (specular)



Fully diffuse surface (Lambertian)

Lighting
Specular component : i_{spec}

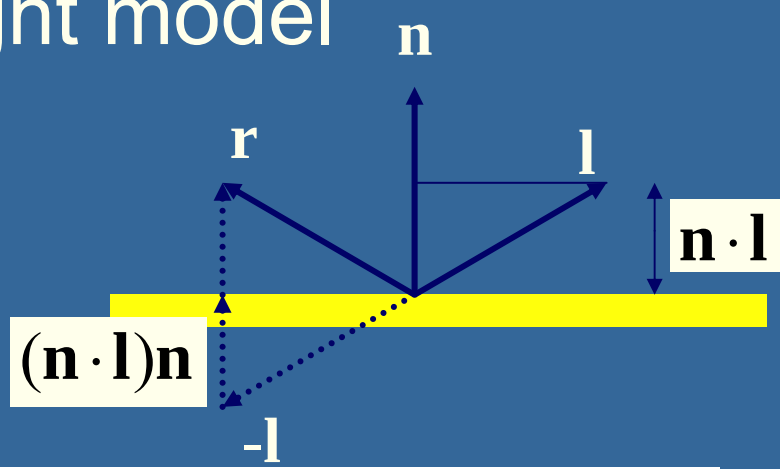
- Diffuse is dull (left)
- Specular: simulates a highlight



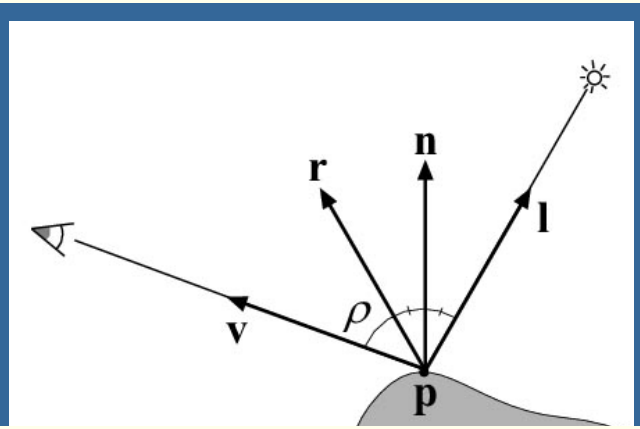
Specular component: Phong

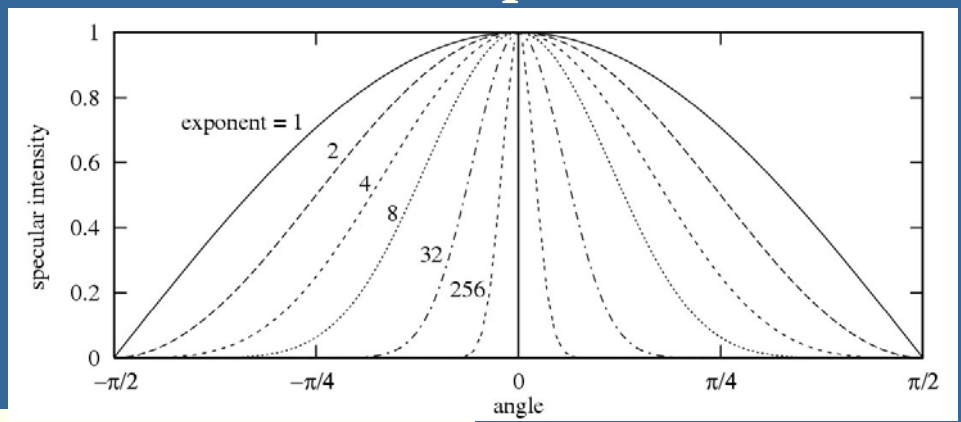
- Phong specular highlight model
- Reflect $\mathbf{l}$ around $\mathbf{n}$:

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



$$i_{spec} = (\mathbf{r} \cdot \mathbf{v})^{m_{shi}} = (\cos \rho)^{m_{shi}}$$





$$\mathbf{i}_{spec} = \max(0, (\mathbf{r} \cdot \mathbf{v})^{m_{shi}})\mathbf{m}_{spec} \otimes \mathbf{s}_{spec}$$

- Next: Blinns highlight formula: $(\mathbf{n} \cdot \mathbf{h})^{m}$

Halfway Vector

Blinn proposed replacing **v·r** by **n·h** where

h = (**l**+**v**)/|**l** + **v**|

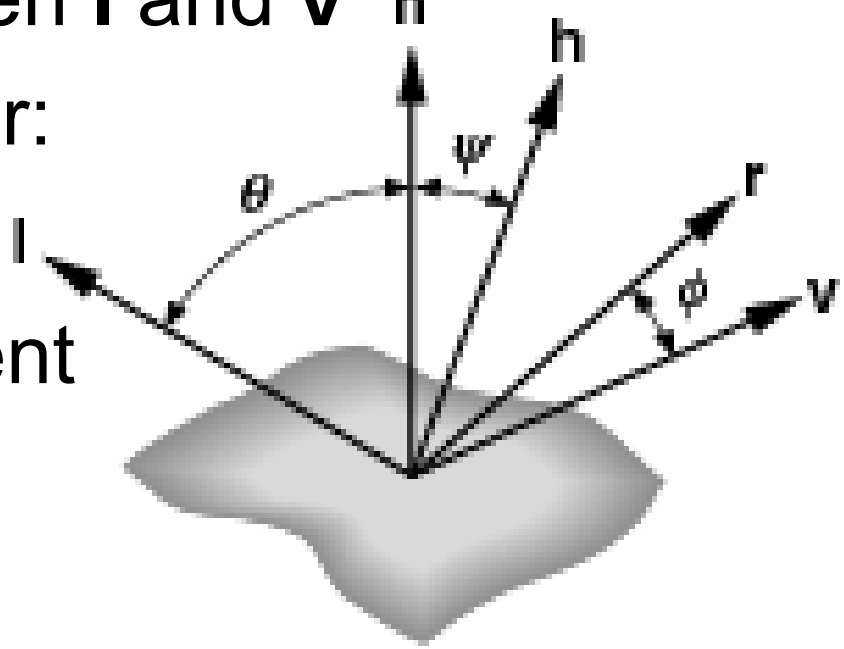
(**l**+**v**)/2 is halfway between **l** and **v**

If **n**, **l**, and **v** are coplanar:

$$\psi = \phi/2$$

Must then adjust exponent so that $(\mathbf{n}\cdot\mathbf{h})^{e'} \approx (\mathbf{r}\cdot\mathbf{v})^{e}$

$(e' \approx 4e)$

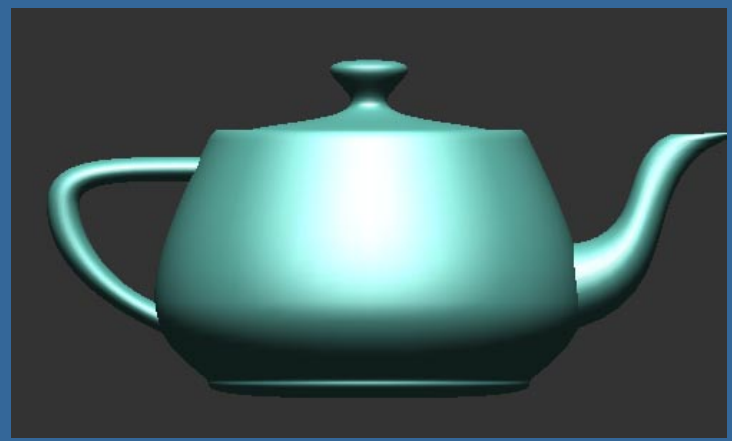


Lighting

$i = i_{amb} + i_{diff} + i_{spec}$


+

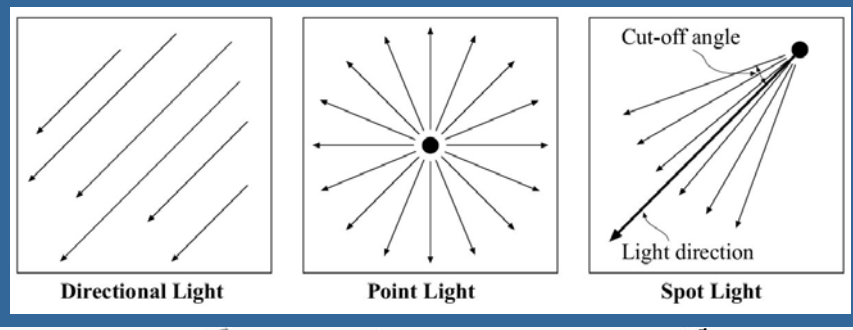
+

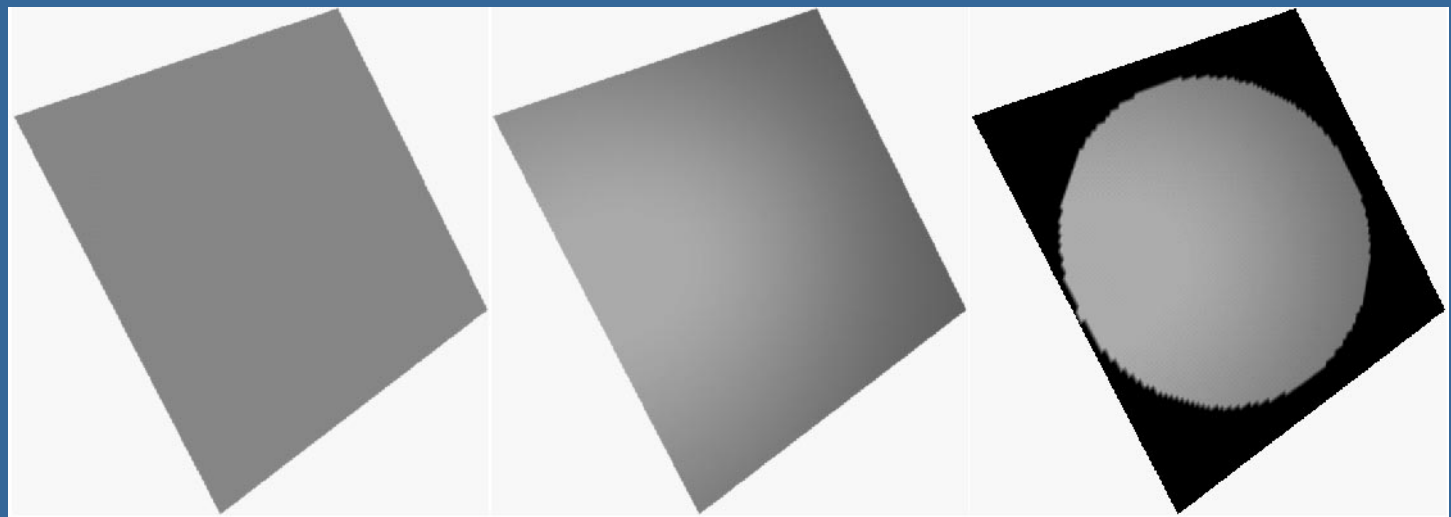
=


- This is just a hack!
- Has little to do with how reality works!

Additions to the lighting equation

- Depends on distance: $1/(a+bt+ct^2)$
- Can have more lights: just sum their respective contributions
- Different light types:





OpenGL Example

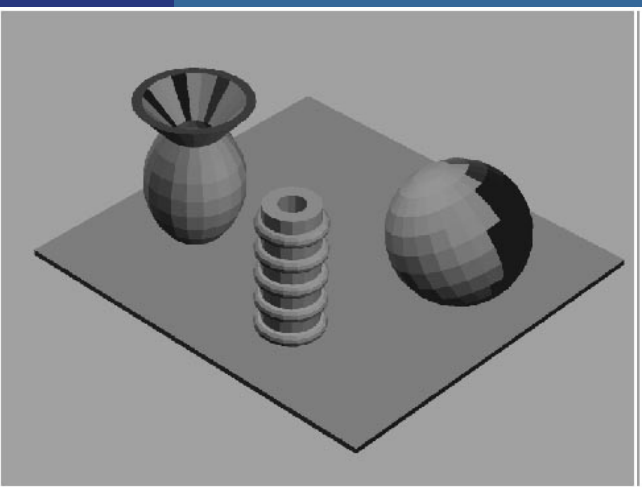
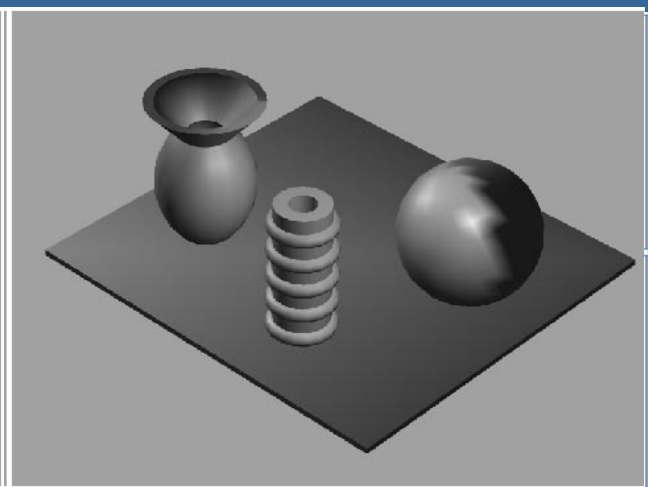
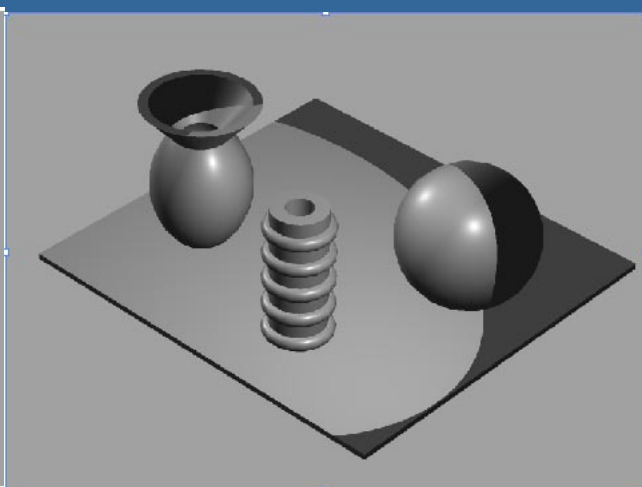
Sets constant (no) attenuation:

```
glLight(GL_LIGHT0, GL_CONSTANT_ATTENUATION, 1.0);
glLight(GL_LIGHT0, GL_LINEAR_ATTENUATION, 0.0);
glLight(GL_LIGHT0, GL_QUADRATIC_ATTENUATION, 0.0);
```

Attenuation often looks unnatural in OpenGL, due to only 8-bit colors (too small intensity range)

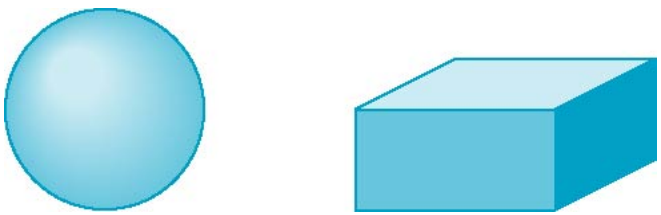
Shading

- Shading: do lighting (at e.g. vertices) and determine pixel's colors from these
- Three common types of shading:
  - Flat, Goraud, and Phong
  - glShadeModel(GL_FLAT / GL_SMOOTH);

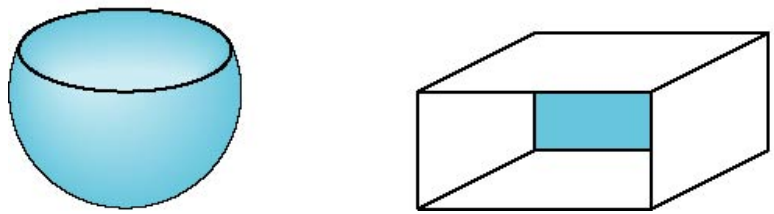




Front and Back Faces

- The default is shade only front faces which works correctly for convex objects
- If we set two sided lighting, OpenGL will shade both sides of a surface
- Each side can have its own properties which are set by using **`GL_FRONT`**, **`GL_BACK`**, or **`GL_FRONT_AND_BACK`** in **`glMaterialf`**



back faces not visible



back faces visible

Fog

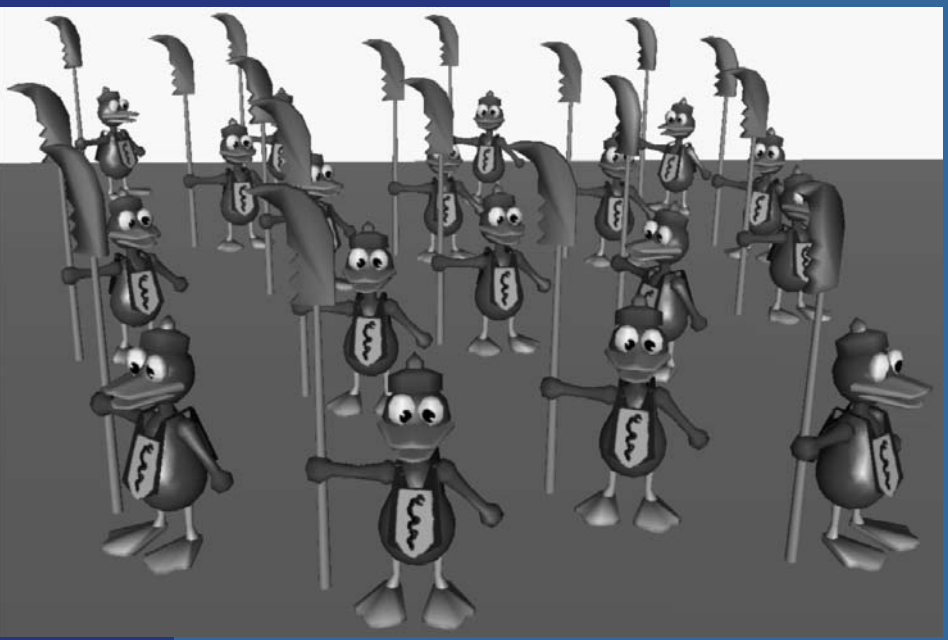
- Simple atmospheric effect
  - A little better realism
  - Help in determining distances
- Color of fog: $\mathbf{c}_f$ color of surface: $\mathbf{c}_s$

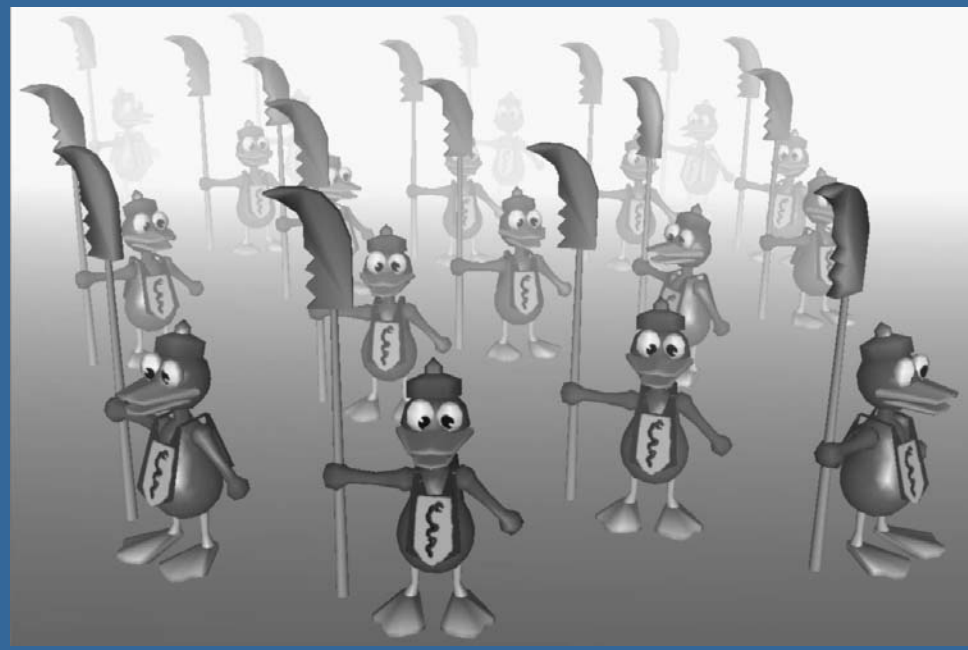
$$\mathbf{c}_p = f\mathbf{c}_f + (1-f)\mathbf{c}_s \qquad f \in [0,1]$$

- How to compute f?
- 3 ways: linear, exponential, exponential-squared
- Linear:

$$f = \frac{z_{end} - z_p}{z_{end} - z_{start}}$$

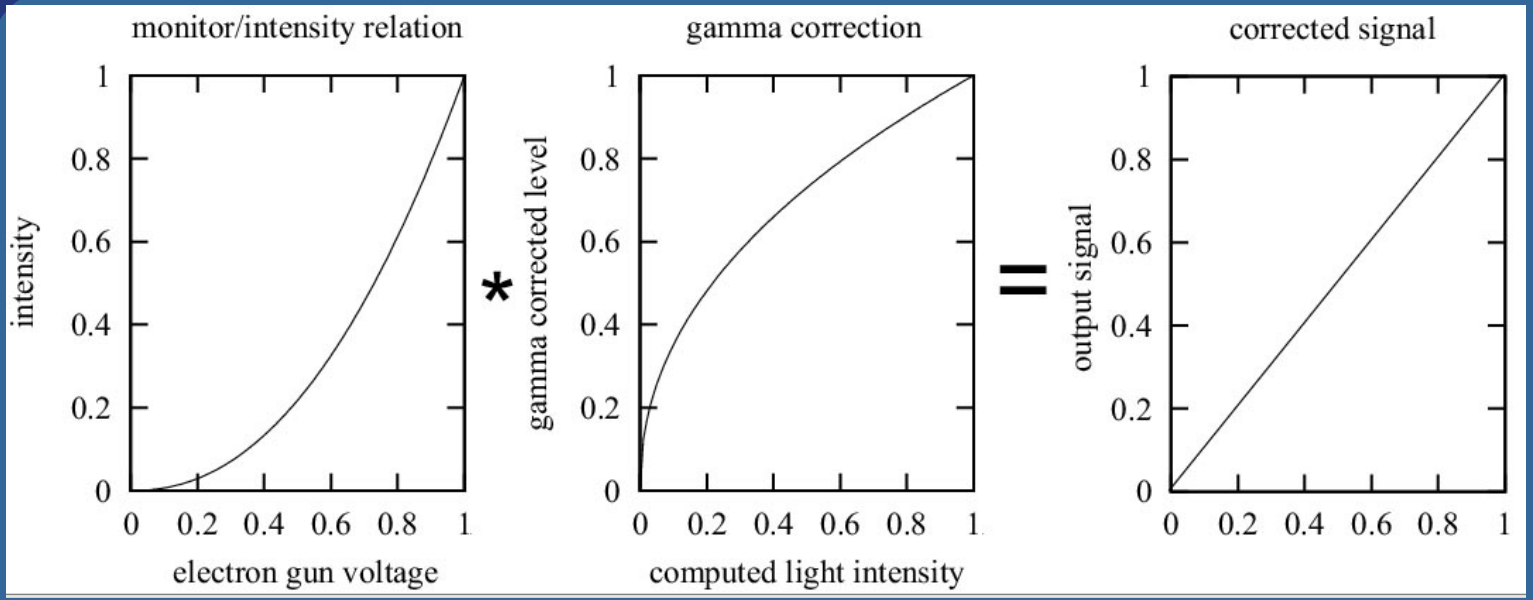
Fog example





- Often just a matter of
  - Choosing fog color
  - Choosing fog model
  - Turning it on

Gamma correction



- If input to gun is 0.5, then you don't get 0.5 as output in intensity
- Instead, gamma correct that signal: gives linear relationship

Gamma correction

$$I = a(V + \varepsilon)^{\gamma}$$

- *I*=intensity on screen
- *V*=input voltage (electron gun)
- *a*, ε, *and* γ *are* constants for each system
- Common gamma values: 2.3-2.6
- Assuming ε=0, gamma correction is:

$$c = c_i^{(1/\gamma)}$$

Why is it important to care about gamma correction?

- Portability across platforms
- Image quality
  - Texturing
  - Interpolation
- One solution is to put gamma correction in hardware…

Transparency and alpha

- Transparency
  - Very simple in real-time contexts
- The tool: alpha blending (mix two colors)
- Alpha (α) is another component in the frame buffer, or on triangle
  - Represents the opacity
  - 1.0 is totally opaque
  - 0.0 is totally transparent
- The over operator: $\mathbf{c}_o = \alpha\mathbf{c}_s + (1-\alpha)\mathbf{c}_d$

Rendered object

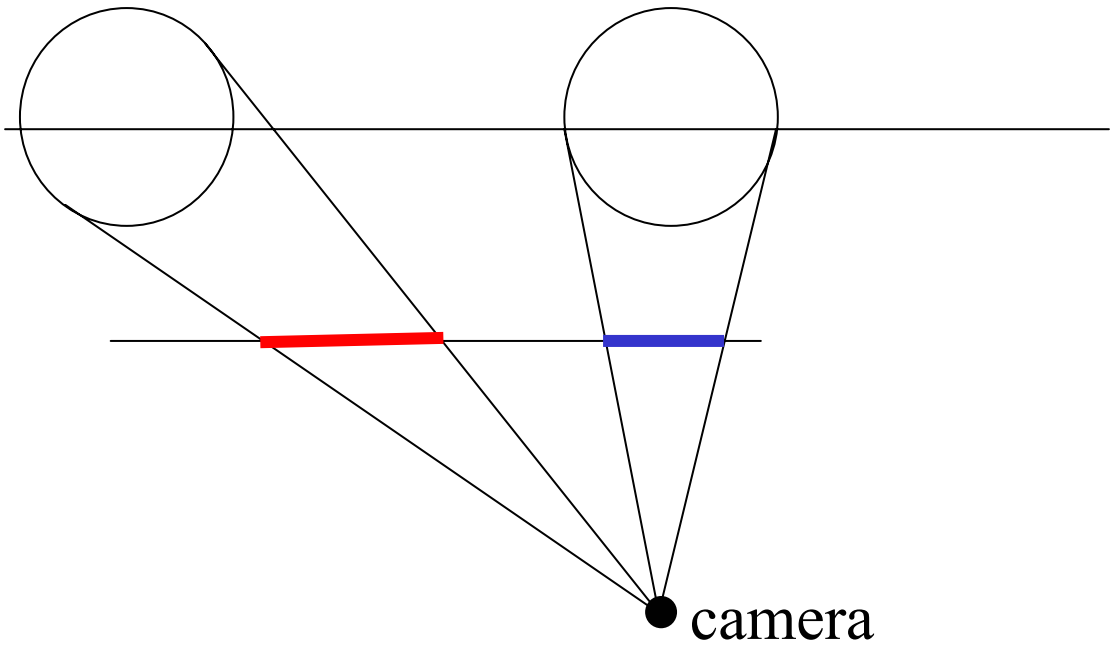
Transparency

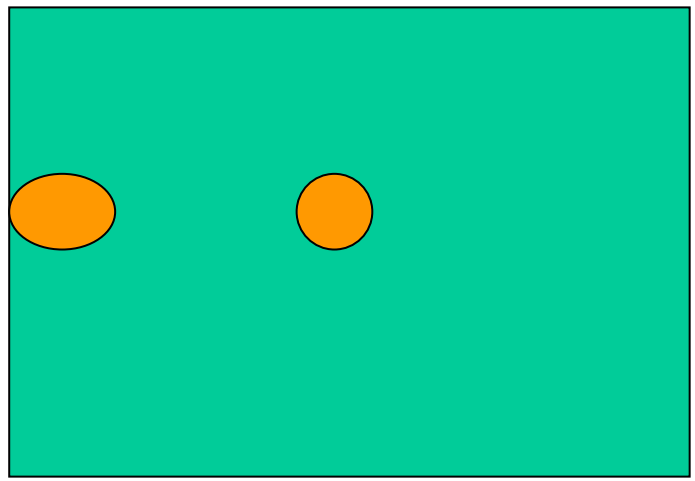
- Need to sort the transparent objects
  - Render back to front
- Lots of different other blending modes

- Can store RGBα in textures as well
- Two ways:
  - Unmultiplied
  - Premultiplied

Perspective distorsion

- Spheres often appear as ellipsoids when located in the periphery. Why?



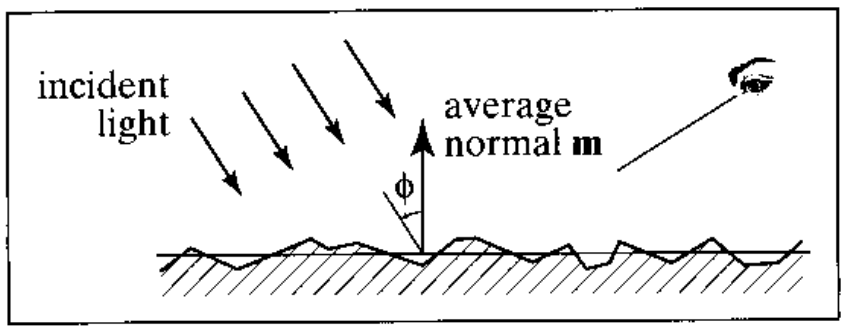


Exaggerated example

If our eye was placed at the camera position, we would not see the distorsion. We are often positioned way behind the camera.

Cook-Torrance Shading

- Phong highlights tend to give shiny plastic look
- More physically accurate models have been developed:
  - Part absorbed as heat
  - Part scattered back as diffuse light
  - Part reflected as specular light
- Torrance and Sparrow and Trowbridge and Reitz:



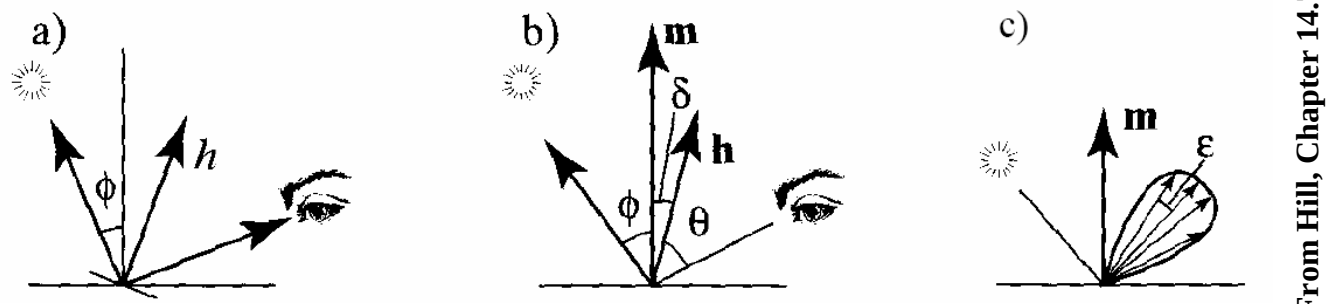
From Hill, Chapter 14.7.3

  - Rough surface is collection of shiny microfacets with an average normal **m** possibly different from macroscopic surface normal **n**

Cook-Torrance Shading

– distribution of facet orientations

a) Each microfacet act as a tiny perfect mirror



From Hill, Chapter 14.7.3

b) Thus only facets with angle (θ-Φ)/2 reflects light into the eye

c) The Beckman distribution has shown to be a good mathematical fit for many rough surfaces:

$$D(\delta) = \frac{1}{4m^2 \cos^2(\delta)} e^{-\left(\frac{\tan(\delta)}{m}\right)^2}$$

m styr lobvidden, (ytans grovhet)

δ är vinkel mellan normal **m** och halvvektorn mellan inkommande ljus och ögat

Torrance and Sparrow: Shadowing and Masking

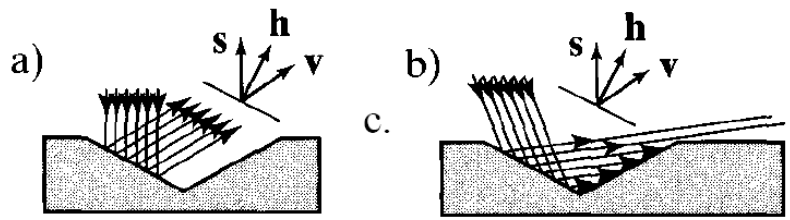
Introduce geometric factor:

$$G = \min(1, G_m, G_s)$$

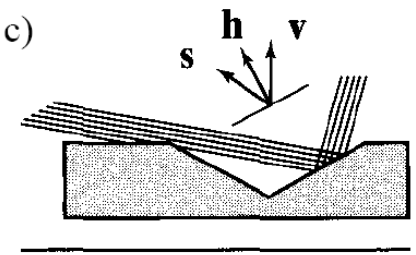
a) G=1

b) G_m (m=masked off reflection)

c) G_s (s=shadowed illumination)



From Hill, Chapter 14.7.3



$$G_m = \frac{2(\mathbf{m}\bullet\mathbf{h})(\mathbf{m}\bullet\mathbf{s})}{\mathbf{h}\bullet\mathbf{s}} \qquad G_s = \frac{2(\mathbf{m}\bullet\mathbf{h})(\mathbf{m}\bullet\mathbf{v})}{\mathbf{h}\bullet\mathbf{s}}$$

Approximativa formler

Skalar sannolikheterna med vinklar mellan **m**, **s**, **v**

s=riktning till ljuskällan

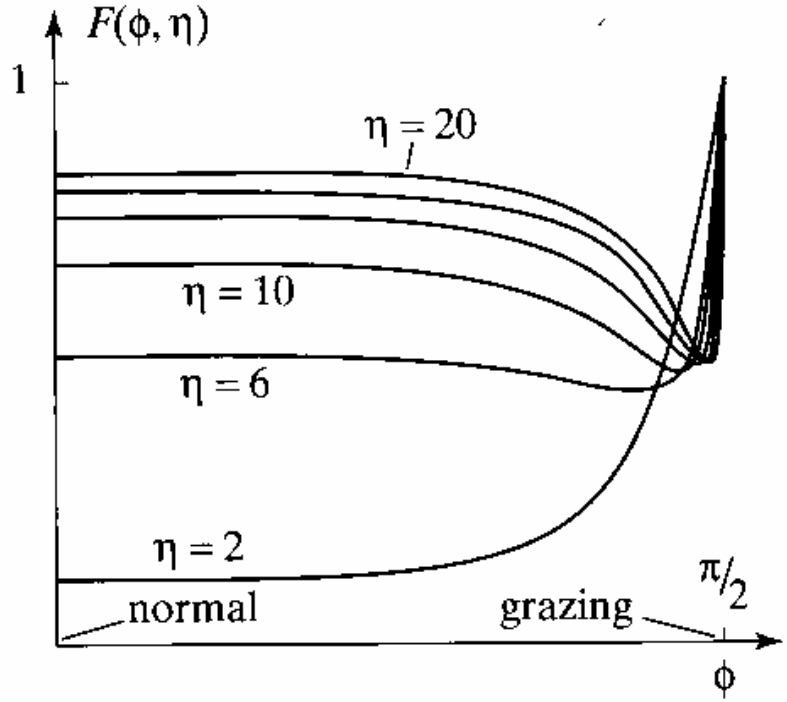
v=riktning till ögat

Torrance and Sparrow: The Fresnell Coefficient

The Fresnell effect depentens on

- wavelength and
- angle of incoming light

Causes the specular light to ”change” color (*color shift*) for different viewing angles



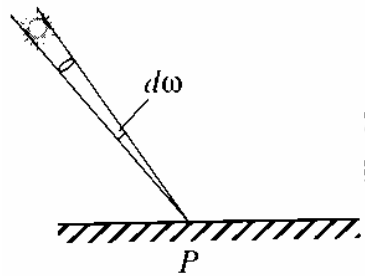
From Hill, Chapter 14.7.3

Cook-Torrance Shading - summary

- Full Cook-Torrance lighting model
  - For each r,g,b-component:

$$I_r = I_{ar} k_a F(0,\eta_r) + I_{sr} d\omega k_d F(0,\eta_r) \times lambert + I_{sr} d\omega k_s \frac{F(\phi,\eta_r) DG}{(\mathbf{m} \bullet \mathbf{v})}$$

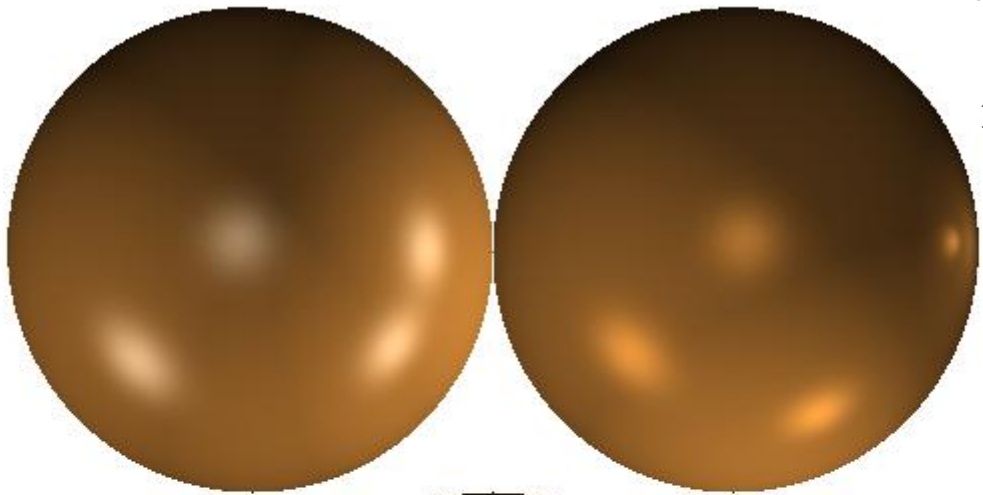
- Explanation:
  - I_{ar} = intensity of red for ambient light
  - $I_{sr} d\omega$ = intensity of red for incoming light
    - $d\omega$ is solid angle of incoming light rays
  - k_a, k_d = reflection coefficients
  - *lambert* = standard lambertian diffuse lighting, max(0,**s**•**m**)
  - $\frac{F(\phi,\eta_r) DG}{(\mathbf{m} \bullet \mathbf{v})}$ = specular reflection scaled by Fresnell (F), Beckmann (D) and Geometry (G) factors



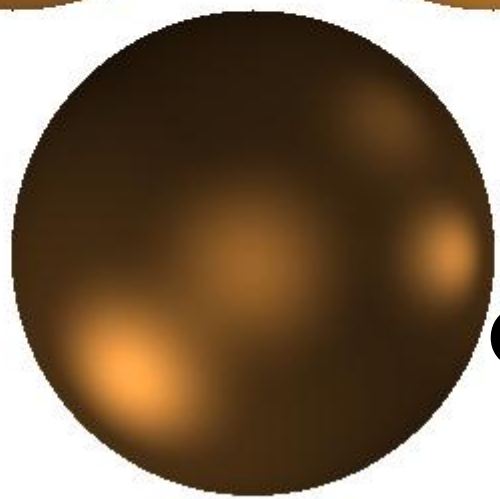
From Hill, Chapter 14.7.3

Comparison

Phong



Phong with wavelength dependent specular reflection



Cook-Torrance

From: http://www.designlab.ku.edu/~miller/Courses/773/Fall2005/Projects/Proj2Files/PNTriangleStripExamplesCT.html