CITS3003 Graphics & Animation

Lecture 16:
Shading Models
(i.e., methods)

Objectives

- Introduce distance terms to the shading model.
- More details about the Phong model (light-material interaction).
- Introduce the Blinn lighting model (also known as the modified Phong model).
- Consider computation of the normal vectors of some simple surfaces.

Distance Terms

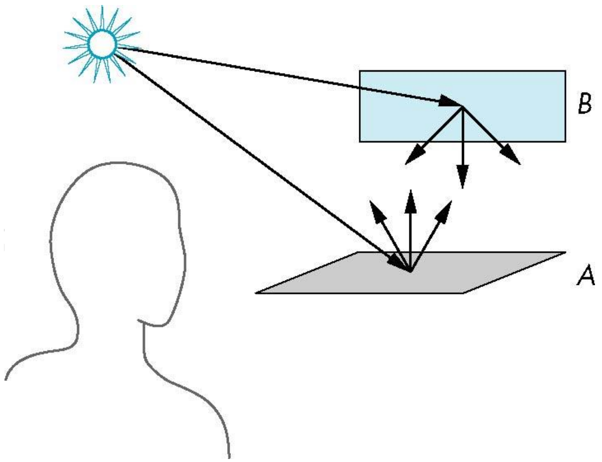
- Most shading calculations require the direction from the point on the surface to the light source position.
- As we move across a surface, calculating the intensity at each point, we should re-compute this vector repeatedly.
- However, if the light source is far from the surface, the vector does not change much as we move from point to point.
- The calculations for distant light sources are similar to the calculations for parallel projections; they replace the *location* of the light source with the *direction* of the light source.

- Point Source $P_0 = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$, Distant Source $P_0 = \begin{bmatrix} x \\ y \\ z \\ 0 \end{bmatrix}$

Distance Terms

- Light from a point source that reaches a surface is inversely proportional to the square of the distance between them

- We can add a factor of the form $1/(a + bd + cd^2)$ to the diffuse and specular terms for both point sources and



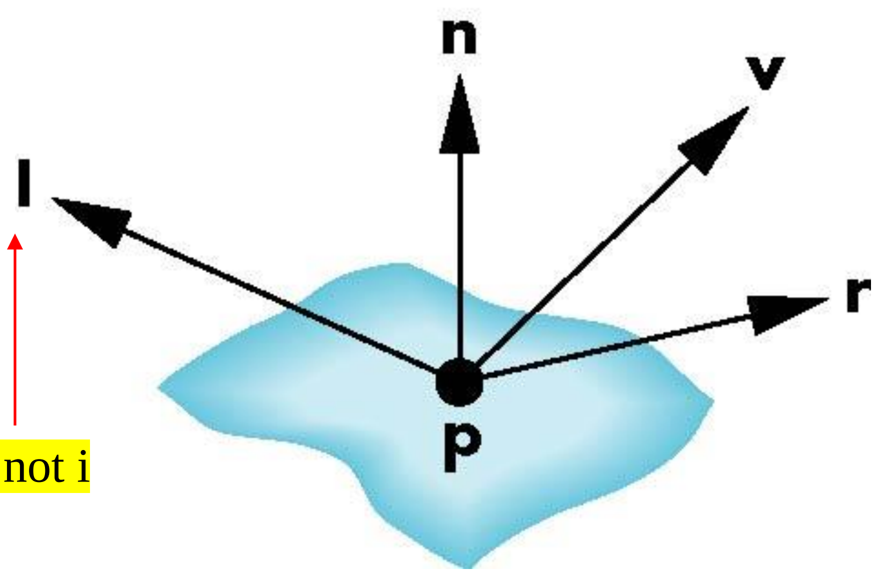
- The constant (a) and linear (bd) terms soften the effect of the point source

Note: the distance term should not be applied to the ambient term.

The Phong Reflection Model

Recall that the Phong Model:

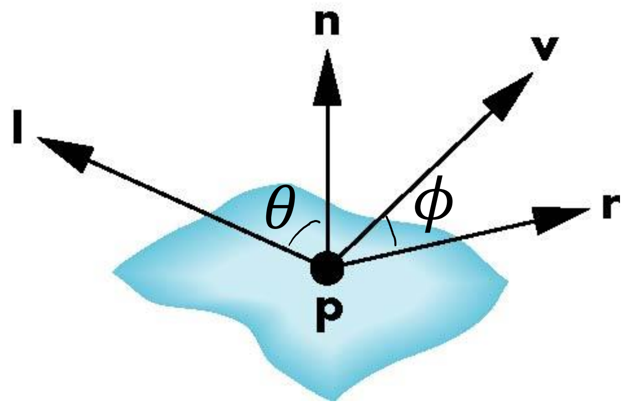
- Is a simple model that can be computed rapidly
- Has three terms
  - Diffuse term
  - Specular term
  - Ambient term



- Uses four vectors to calculate color for an arbitrary point **p** on a surface:
  - Vector **l** (to light source)
  - Vector **v** (to viewer or camera)
  - Vector **n** (Normal vector at p)
  - Vector **r** (Perfect reflector of **l** with respect to **n**)

Letter *l*, not i

The Phong Reflection Model



Recall that the Phong Model:

- Has three terms
  - Diffuse term
  - Specular term
  - Ambient term

Adding all three terms, Phong model for each light source can be written as:

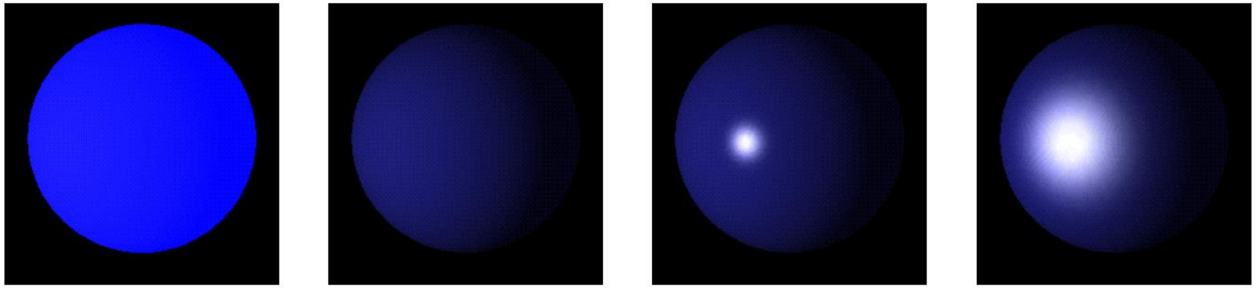


Image source

ambient diffuse specular combined

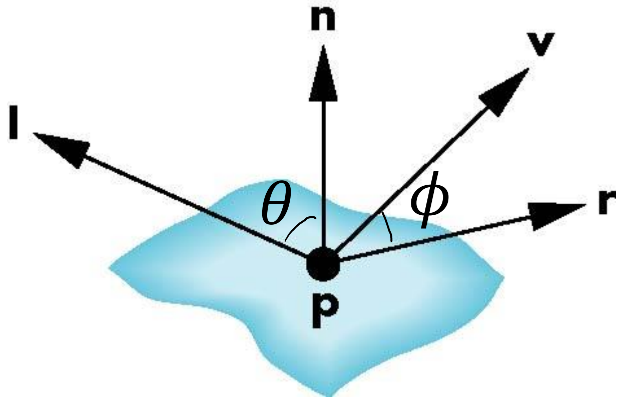
$$k_a L_a \; + \; k_d L_d \cos\theta \; + \; k_s L_s \cos^{\alpha}\phi \; = \; I$$

$\cos\theta = \mathbf{l} \cdot \mathbf{n}$; $\cos\phi = \mathbf{v} \cdot \mathbf{r}$

Example: Changing the shininess coefficient

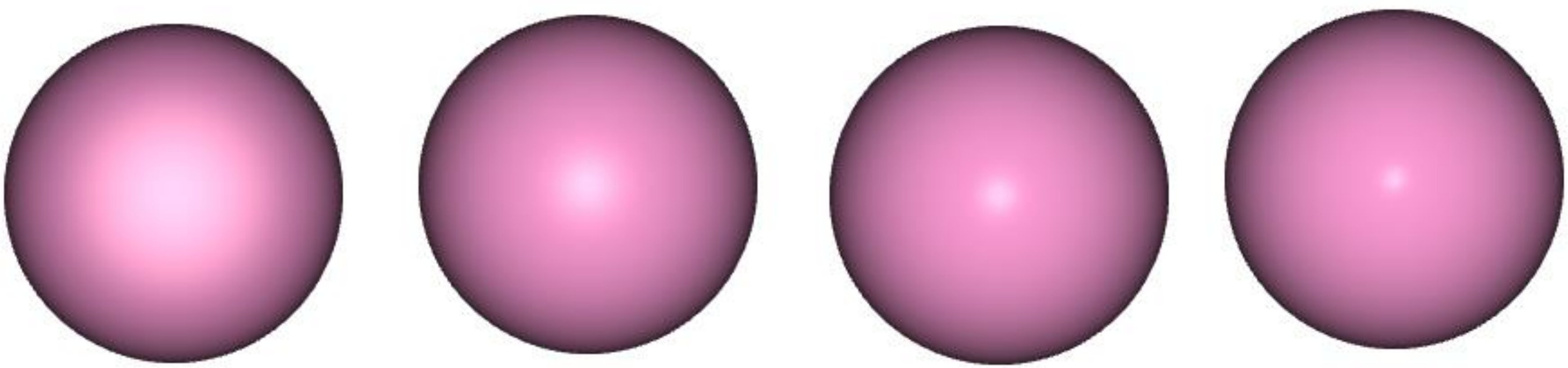
Recall that the Phong Model:

- Has three terms
  - Diffuse term
  - Specular term
  - Ambient term



$$k_s L_s \cos^{\alpha}\phi = I$$

Shininess coefficient



$\alpha = 2$ $\alpha = 16$ $\alpha = 50$ $\alpha = 128$

Demo

Shading – the Light Source Part

- In the Phong Model, we add the shading results from all the light sources together.

Total illumination for a point P = ∑(Lighting for all lights)

- 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.
  - For example, L_{id}, L_{is}, L_{ia}
- Each term has its own red, green and blue components.
  - For example, $L_{id} = L_{ird}, L_{igd}, L_{ibd}$
- Hence, there are 9 coefficients for each point source L_i :
  $L_{ird}, L_{igd}, L_{ibd}, L_{irs}, L_{igs}, L_{ibs}, L_{ira}, L_{iga}, L_{iba}$

Shading – the Light Source Part

- We can place these nine coefficients in a 3 × 3 illumination matrix for the i_{th} light source:.

$$\mathbf{L}_i = \begin{bmatrix} L_{i\mathrm{ra}} & L_{i\mathrm{ga}} & L_{i\mathrm{ba}} \\ L_{i\mathrm{rd}} & L_{i\mathrm{gd}} & L_{i\mathrm{bd}} \\ L_{i\mathrm{rs}} & L_{i\mathrm{gs}} & L_{i\mathrm{bs}} \end{bmatrix}$$

- In practice, we will use constructs such as
- **vec3 light_i_ambient, light_i_diffuse, light_i_specular;**
- Or **vec4**

Shading – the Reflection Part

- We can compute how much of each of the incident lights is reflected at the point of interest.

- For example, for the red diffuse term from source *i*, L_{ird}, we can compute a *reflection term* R_{ird}, and the latter's contribution to the intensity at ***p*** is $R_{ird}L_{ird}$.

$$\mathbf{R}_i = \begin{bmatrix} R_{i\mathrm{ra}} & R_{i\mathrm{ga}} & R_{i\mathrm{ba}} \\ R_{i\mathrm{rd}} & R_{i\mathrm{gd}} & R_{i\mathrm{bd}} \\ R_{i\mathrm{rs}} & R_{i\mathrm{gs}} & R_{i\mathrm{bs}} \end{bmatrix}$$

$$\begin{aligned} I_{i\mathrm{r}} &= R_{i\mathrm{ra}}L_{ira} + R_{i\mathrm{rd}}L_{i\mathrm{rd}} + R_{i\mathrm{rs}}L_{i\mathrm{rs}} \\ &= I_{i\mathrm{ra}} + I_{i\mathrm{rd}} + I_{i\mathrm{rs}}. \end{aligned}$$

$$I = I_\mathrm{a} + I_\mathrm{d} + I_\mathrm{s} = L_\mathrm{a}R_\mathrm{a} + L_\mathrm{d}R_\mathrm{d} + L_\mathrm{s}R_\mathrm{s},$$ omitting the subscripts *i*, r

the necessary computations are the same for each source and for each primary color

Shading – Material Properties

- Surfaces of objects have their material properties to compute with the light source properties, i.e.,

- There are nine absorption coefficients

$\boxed{k_{\mathrm{r}d}, k_{\mathrm{gd}}, k_{\mathrm{bd}},}$ $\boxed{k_{\mathrm{r}s}, k_{\mathrm{gs}}, k_{\mathrm{bs}},}$ $\boxed{k_{\mathrm{ra}}, k_{\mathrm{ga}}, k_{\mathrm{ba}}}$

- and a shininess coefficient α

Putting it all Together

- Instead of

$$I = k_a L_a + k_d L_d \cos\theta + k_s L_s \cos^{\alpha}\phi$$

- We can compute the lighting for RGB colors separately for each light source as:

$$I_r = k_{ra} L_{ra} + k_{rd} L_{rd} \cos\theta + k_{rs} L_{rs} \cos^{\alpha}\phi$$

$$I_g = k_{ga} L_{ga} + k_{gd} L_{gd} \cos\theta + k_{gs} L_{gs} \cos^{\alpha}\phi$$

$$I_b = k_{ba} L_{ba} + k_{bd} L_{bd} \cos\theta + k_{bs} L_{bs} \cos^{\alpha}\phi$$

- For N lights, the above calculations have to be repeated for each light

Ambient Term

- The intensity of ambient light I_a is the same at every point on the surface.

- Some of this light is absorbed and some is reflected. The amount reflected is given by the ambient reflection coefficient, $R_a = k_a$.

$$0 \leq k_a \leq 1$$

- Thus,

$$I_a = k_a L_a$$

Diffuse Term

- A perfectly diffuse reflector scatters the light that it reflects equally in all directions.

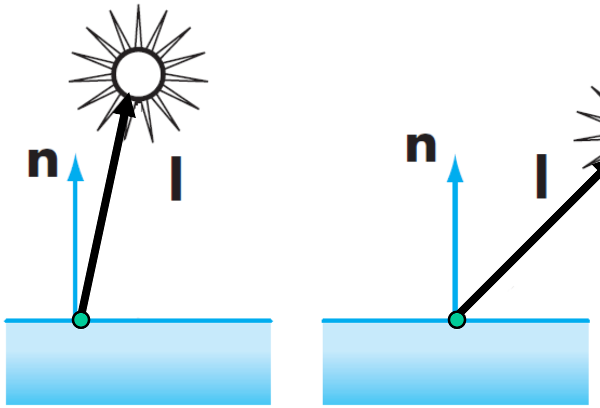
- Such a surface appears the same to all viewers. However, the amount of light reflected depends both on the material and on the position of the light source relative to the surface.

- $R_d \propto \cos\theta$
- $\cos\theta = \boldsymbol{l}.\boldsymbol{n}$
- $I_d = L_d R_d = L_d(\boldsymbol{l}.\boldsymbol{n})k_d$

$$I_d = \frac{1}{a+bd+cd^2} k_d L_d \, (\boldsymbol{l}.\boldsymbol{n})$$

quadratic distance attenuation

d corresponds to distance between light source and the point on a surface



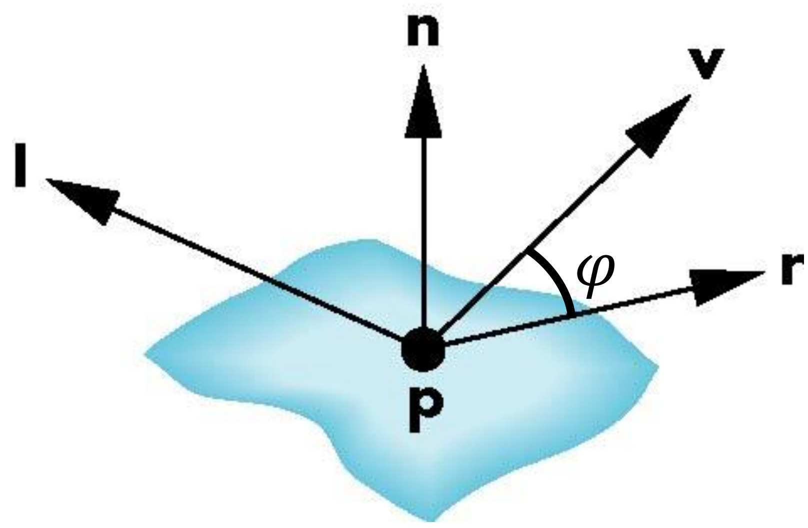
Specular Term

- Whereas a diffuse surface is rough, a specular surface is smooth. The smoother the surface is, the more it resembles a mirror.

- Phong proposed an approximate model that can be computed with only a slight increase over the work done for diffuse surfaces.

- $I_s = L_s R_s = k_s L_s \cos^{\alpha} \varphi$

$$I_s = \frac{1}{a+bd+cd^2} k_s L_s (\mathbf{r}.\,\mathbf{v})^{\alpha}$$



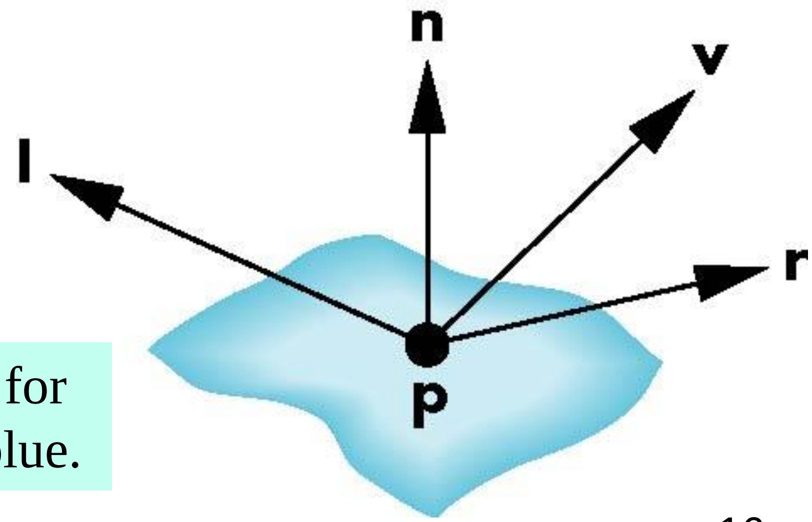
Total Shading = Adding up the Components

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

$$I = k_{\mathrm{d}} L_{\mathrm{d}} (\mathbf{l} \cdot \mathbf{n}) + k_{\mathrm{s}} L_{\mathrm{s}} (\mathbf{v} \cdot \mathbf{r})^{\alpha} + k_{\mathrm{a}} L_{\mathrm{a}}$$

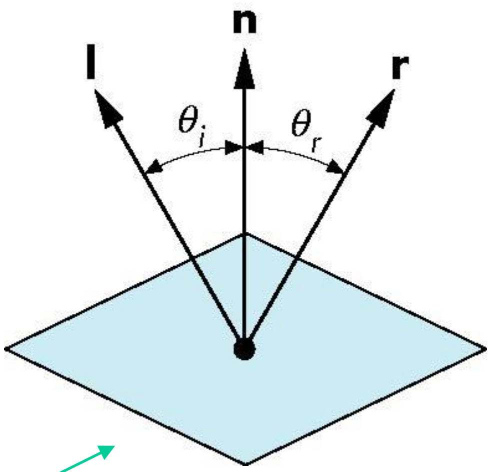
- For each colour component we add contributions from all light sources

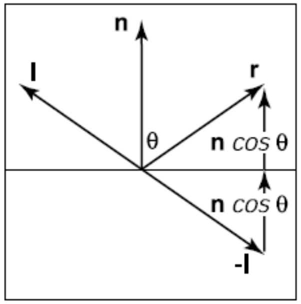
There are 3 such equations: one for red, one for green, and one for blue.



The Phong Shading Model – Computing the perfect reflector **r**

- To compute the shading value I on the previous slide, we need to know those vectors in the model.
- The normal vector **n** is determined by the local orientation at point **p**.
- Vector **l** and **v** are specified by the application.
- We can compute **r** from **l** and **n**
  - We want: Angle of incidence θ_i = angle of reflection θ_r.
  - The three vectors **l**, **n**, and **r** must be coplanar.
  - It is easy to verify that





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

Letter *l*, not 1.

The Modified Phong Model or Blinn Lighting Model

- The specular term in the Phong model is problematic because it requires the calculation of a new reflection vector **r** for each vertex and then the dot product with **v**.

- Blinn suggested an approximation using the halfway vector that is more efficient

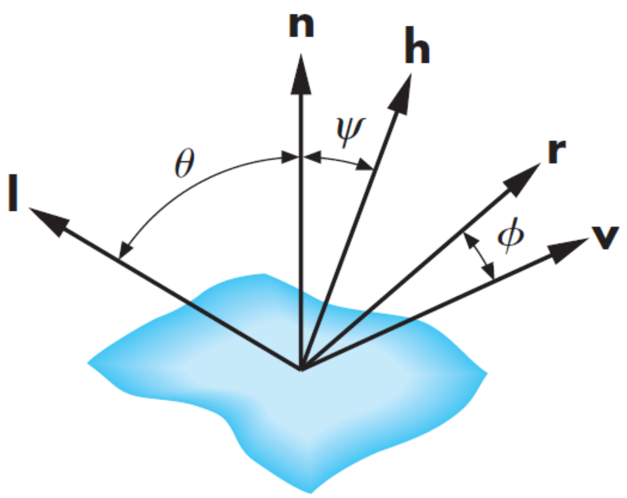
- This is referred to as the **modified Phong reflection model** or the **Blinn-Phong shading model**

  (the terminology varies in many textbooks, the term “lighting” is also used instead of “reflection” and “shading”).

Blinn-Phong Model – the Halfway Vector

- Instead of computing **r**, the **h** vector which is a normalized vector halfway between **l** and **v** is used:

$$\mathbf{h} = \frac{\mathbf{l} + \mathbf{v}}{|\mathbf{l} + \mathbf{v}|}$$



The Blinn-Phong Model – using the halfway vector

- Having got the halfway vector **h**, we replace $(\mathbf{v} \cdot \mathbf{r})^{\alpha}$ by $(\mathbf{n} \cdot \mathbf{h})^{\beta}$ where β is chosen to match the shininess of the material

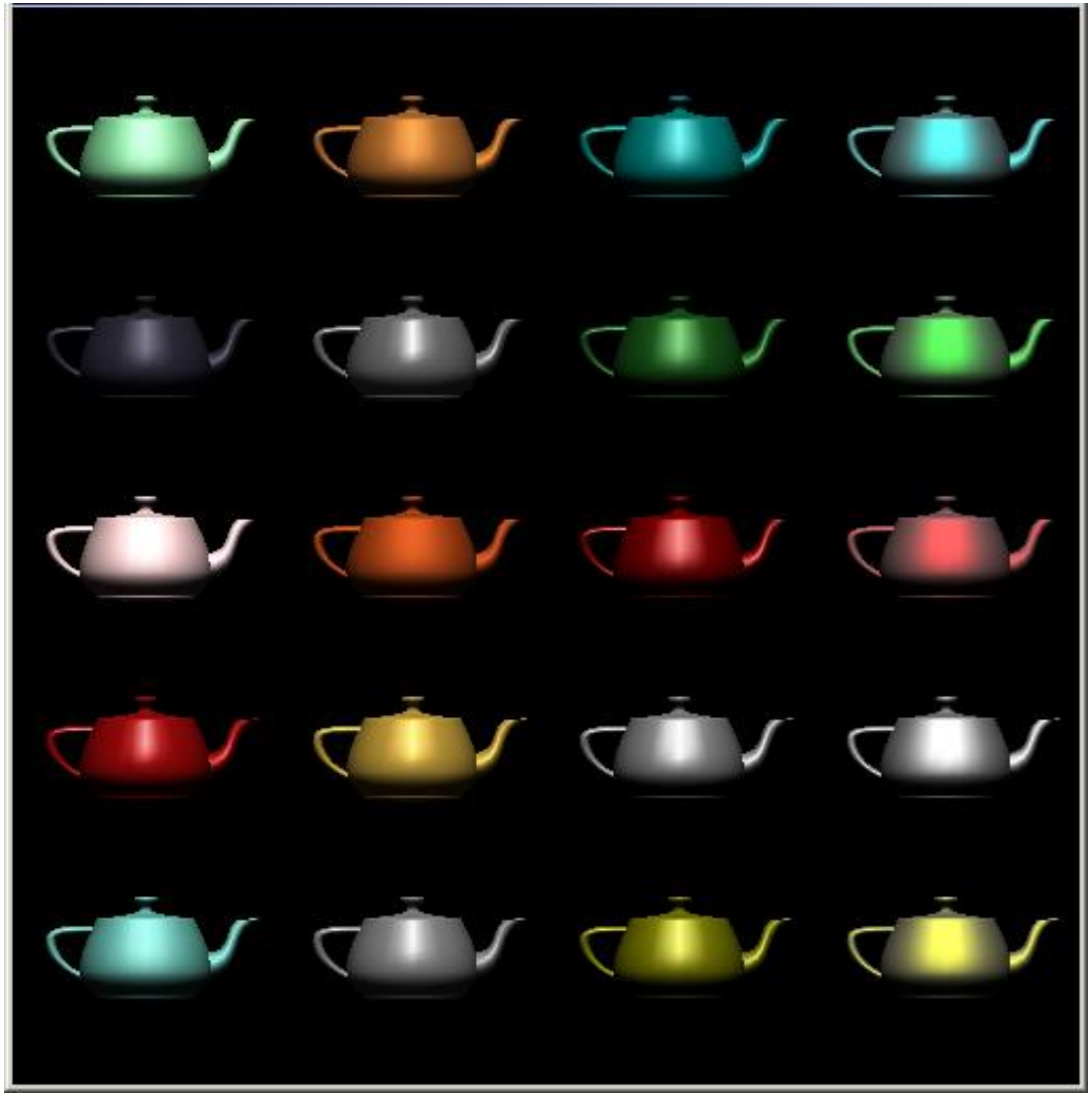
- Note that halfway angle is half of angle between **r** and **v** if vectors are coplanar

- The resulting model is known as the **modified Phong** or **Blinn lighting model**. The model can be written as (without the distance term):

$$I = k_{\mathrm{d}} L_{\mathrm{d}} (\mathbf{l} \cdot \mathbf{n}) + k_{\mathrm{s}} L_{\mathrm{s}} (\boldsymbol{n} \cdot \boldsymbol{h})^{\beta} + k_{\mathrm{a}} L_{\mathrm{a}}$$

- The Blinn lighting model is the default shading model in OpenGL

Examples



Colour plate 17 from the book: Array of Utah teapots with different material properties.

Normal vector

Computation of Normal Vector **n**

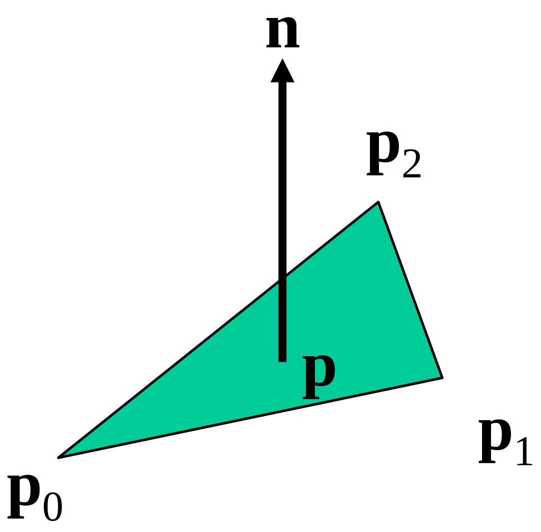
- Whether we use the Phong model or the Blinn model, we need to also determine the normal vector **n** at each point **p**.
- But how do we determine **n** in general?
- For simple surfaces like spheres there are formulas, but how we determine **n** differs depending on underlying representation of surface.
- OpenGL leaves the determination of normals to the application.

Computing the Normal Vector **n** for a Plane

- Equation of plane: $ax + by + cz + d = 0$

- From Chapter 3 we know that a plane is determined by
  - three points $\mathbf{p}_0$, $\mathbf{p}_1$, $\mathbf{p}_2$ or by
  - a normal $\mathbf{n}$ and $\mathbf{p}_0$

- The normal vector can be obtained by:



$$\mathbf{n} = (\mathbf{p}_2 - \mathbf{p}_0) \times (\mathbf{p}_1 - \mathbf{p}_0)$$

- We then normalize $\mathbf{n}$ to a unit vector.

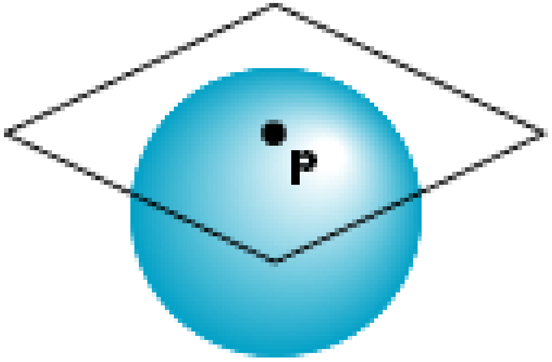
Computing the Normal Vector **n** – for a Sphere

- If we have an implicit representation of a sphere (with unit radius and centre at the origin):

$$f(x, y, z) = 0$$
$$x^2 + y^2 + z^2 - 1 = 0$$

- The normal vector **n** at a point **p** is given by gradient of f at **p** , i.e.,

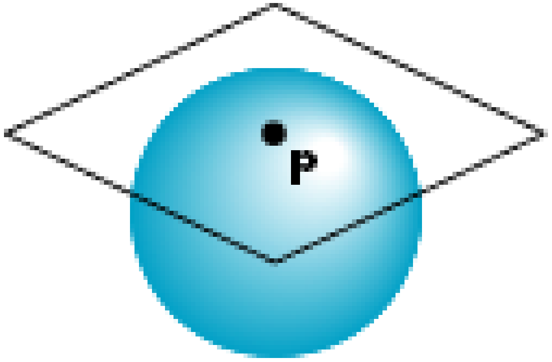
$$\mathbf{n} = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right]^{\mathrm{T}}$$
$$= [2x, 2y, 2z]^{\mathrm{T}} = 2\mathbf{p}$$



Computing the Normal Vector $\mathbf{n}$ – for a Sphere

- If we have a parametric representation of a sphere:

$$x = x(u,v) = \cos u \sin v$$
$$y = y(u,v) = \cos u \cos v$$
$$z = z(u,v) = \sin u$$



- The tangent plane is determined by the vectors

$$\partial \mathbf{p}/\partial u = [\partial x/\partial u, \partial y/\partial u, \partial z/\partial u]^{\mathrm{T}}$$
$$\partial \mathbf{p}/\partial v = [\partial x/\partial v, \partial y/\partial v, \partial z/\partial v]^{\mathrm{T}}$$

- The normal vector is given by the cross product

$$\mathbf{n} = \partial \mathbf{p}/\partial u \times \partial \mathbf{p}/\partial v$$

Computing the Normal Vector **n** – the General Case

- We can also compute the normal vectors of for other simple surfaces:
  - Quadrics
  - Parametric polynomial surfaces
    - E.g., Bezier surface patches (Chapter 10)

Further Reading

"Interactive Computer Graphics – A Top-Down Approach with Shader-Based OpenGL" by Edward Angel and Dave Shreiner, 6^{th} Ed, 2012

- Sec. 5.3. The Phong Reflection Model
- Sec. 5.4. Computation of Vectors

Below is a useful link on lighting concepts

https://learnopengl.com/Lighting/Basic-Lighting