

# Week 8

Need for Shading	▼
Light-Material Interaction	▼
Simple Light Sources	▼
The Phong Reflection Model	▼
Properties of Light Sources	▼
Blinn-Phong Model	▼
Computing the Normal Vector	▼

# Need for Shading

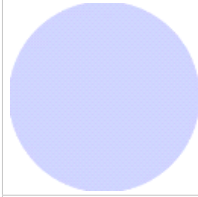
Wednesday, November 20, 2019 3:19 PM

## 3D Perception

### How do we perceive objects as 3D?

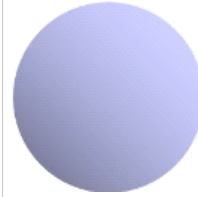
- Stereo vision is one reason but we can still perceive 3D even when we look with one eye.
- **Hidden surface removal** (i.e. occlusion) also gives us depth perception.
- **Shading is another way in which we perceive 3D objects.**
- **Light reflection** from 3D objects *is a function of*:
  - Material properties (colour)
  - The shape of the object - gives the object shading

## Need for Shading



Suppose we build a model of a sphere using many polygons and color it with glColor.

We will get uniform coloring without shading.



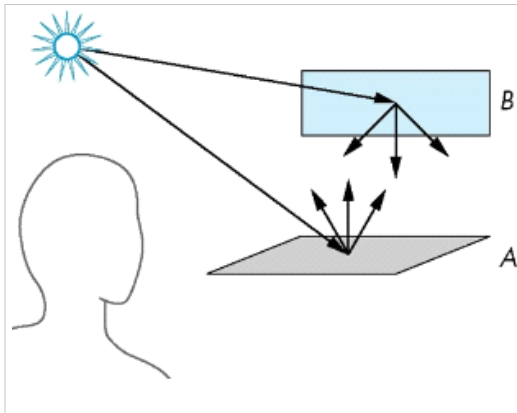
But this is what we want. There appears to be a light source

*Why does the image of a real sphere look like this?*

**Light-material interactions** *cause each point* to have a different color shade

- More reflection occurs in the top right, and this smoothly decreases with distance
- We need to consider:
  - Light sources
  - Material properties (Colour, texture, pattern)
  - Location of viewer
  - Surface orientation (3D shape)

## Scattering



Light strikes object A

→ Some scattered.

→ Some absorbed

Some of scattered light strikes object B

→ Some scattered.

→ Some absorbed.

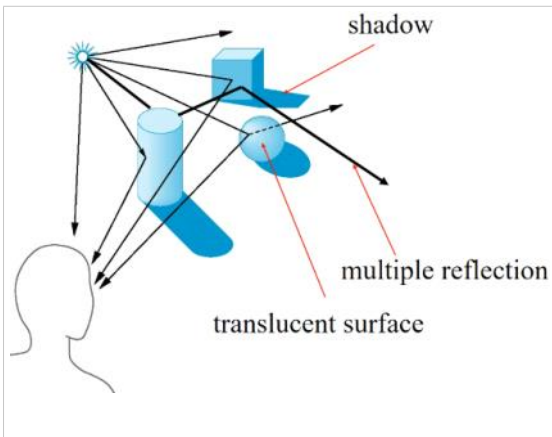
Some of this scattered light strikes A and so on.

This infinite scattering and absorption of light can be described by the **rendering equation**

- Cannot be solved in general
- Is global (used for global lighting effects)
- Includes
  - Shadows (allows for shadow production)
  - Multiple scatterings from object to object

Ray tracing is a special case for perfectly reflecting surfaces

## Global Effects



Correct shading for global effects like shadows requires a global calculation involving all objects and light sources

This is **incompatible with pipeline architecture** which shades each polygon independently (i.e. can only do local rendering)

In computer graphics, especially real time graphics, we are happy if things "look right" as opposed to perfectly real

There 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 colour and brightness of the object

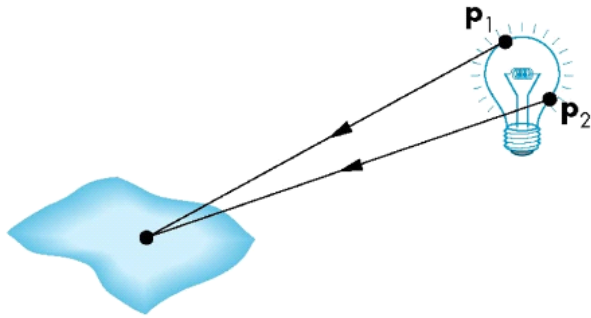
- The more reflected, the greater the brightness
- A surface appears red under white light because the red component of the light is reflected and the rest is absorbed

The reflected light is scattered in a manner that depends on the smoothness and orientation of the surface

- Smooth surfaces will reflect light like a mirror
- Rough surfaces will scatter light

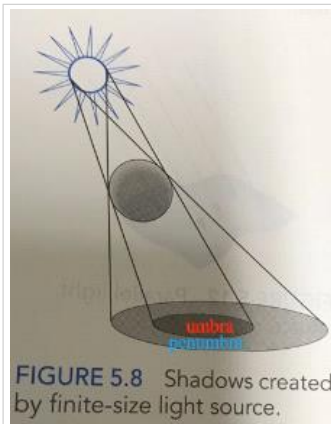
## Light Sources

Real life, general light sources are difficult to work with because we must integrate light coming from all points on the source.





## Finite Light Sources and Shadows



Real light sources have a finite size, meaning we will get **complex shadows**

Parts that see the full light sources will be the brightest (largest area outside the two circles)

Parts that do not see the light source at all will be the darkest (inner shadow circle) (umbra)

Parts that see the light source partially will have brightness values in between the two extremes depending on how much of the light source is visible (outer circle excluding the inner circle) (penumbra)

## Light Attenuation

For point light sources, light intensity decreases with the square of the distance i.e. multiply the intensity by  $\frac{1}{d^2}$  (where  $d$  is the distance from light source)

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

Point sources are easy to model but they result in *harsh (unrealistic) rendering* with very sharp drop in brightness and crisp shadows.

Hence, the light attenuation is generally modelled by a quadratic equation

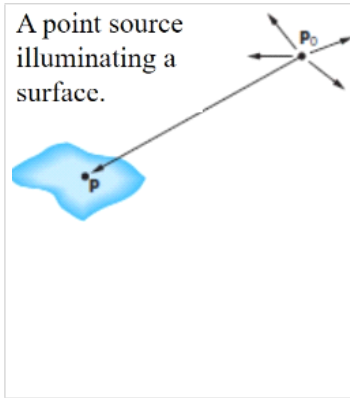
- $\frac{1}{a + bd + cd^2}$
- $a, b, c$  are user defined parameters
  - We can use them to collapse the equation into the real life equation
- $d$  is distance
- Often added to diffuse and specular terms for **point sources and spotlights**
  - But the distance term is not applied to the ambient term, as the ambient light is omnipresent and has no location
- The constant ( $a$ ) and linear ( $bd$ ) terms ***soften the effect of the point source***
- The terms cause the light intensity to ***drop off more slowly***

## Simple Light Sources

### **Simple Light Source Types:**

- Point Source
- Spotlight
- Directional Light Source
- Ambient Light

## Point Source

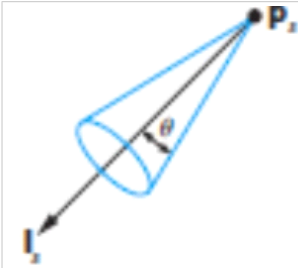


### A point source:

- Is the simplest light source.
- Illuminates equally in all directions.
- Is modelled with:
  - A position  $\mathbf{p}_0 = (x, y, z, 1)^T$
  - An illumination intensity  $\mathbf{I} = (I_r, I_g, I_b)$

For either, a distance term can be incorporated to attenuate  $\mathbf{I}$ .

## Spotlight

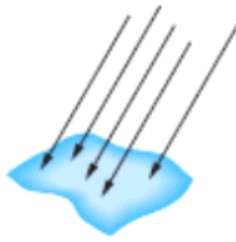


### A spotlight

- Is **modelled with a cone**
  - Apex is at  $p_0$
  - A direction  $I_s$
  - The cone angle (center to edge of cone)  $\theta$
  - An intensity  $I = (I_r, I_g, I_b)$
- Is a point source whose emitted light is restricted to a narrow range of angles.
- If  $\theta = 180^\circ$ , the spotlight becomes a point source.

For either, a distance term can be incorporated to attenuate  $I$ .

## Directional Light Source



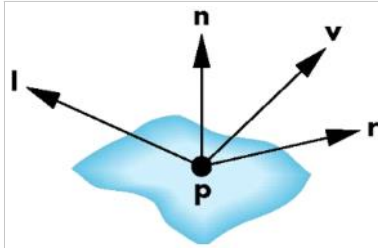
### A Directional Light Source

- Is also known as a distant light source.
- Illuminates objects with parallel rays of light
- Is modelled with:
  - A *direction*  $\mathbf{d} = (x, y, z, \mathbf{0})^T$
  - An illumination intensity  $\mathbf{I} = (I_r, I_g, I_b)$

### Ambient Light

- Models the same amount of light everywhere in the scene.
- The ambient illumination ( $\mathbf{I} = (I_r, I_g, I_b)$ ) is therefore characterized ONLY by an intensity that is identical at every point in the scene.
  - Has no point
- Can also be used to model the integration of multiple sources on a reflecting surface in the scene.

## The Phong Reflection Model

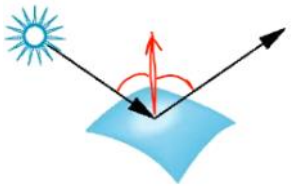



**The Phong Model** is simple and can be computed rapidly

- Has three terms:
  - *Diffuse term*
  - *Specular term*
  - *Ambient term*
- Uses **four vectors** to calculate a colour for an arbitrary point ***p*** on a surface:
  - Vector ***I*** (to light source location)
  - Vector ***v*** (direction of viewer or camera)
  - Vector ***n*** (Surface Normal vector at ***p***, orientation of surface)
  - Vector ***r*** (Perfect reflector of ***I*** with respect to ***n***, where light perfectly reflects (angle from Normal is same on each side))



## Surface Types

 <p>smooth surface</p>	<p>For smooth surfaces, the reflected light concentrates more in the direction that is a perfect mirror reflection of the incident light.</p> <p>Same angle with normal on each side</p>
 <p>rough surface</p>	<p>For rough surfaces, the reflected light scatters in all directions</p>

## Lambertian Surface

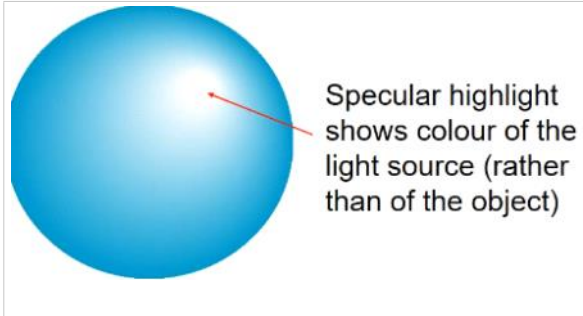
*Perfectly diffuse* reflectors

**Light scattered equally** in all directions

Amount of light reflected is proportional to the vertical component of incoming light:

- This means the reflected light is proportional to  $\cos(\theta_i)$  (vertical component)
- Where  $\cos(\theta_i) = l \cdot n$  (dot product) where  $l, n$  are normalized to unit vectors
- There are also three coefficients ( $k_r, k_g, k_b$ ) that show how much of each color component is reflected
  - These coefficients are specific to the surface.
  - They are defined by surface properties.

## Specular Surfaces



Whereas a diffuse surface is rough, a specular surface is smooth.

Smooth surfaces show **specular highlights** due to incoming light being reflected in directions concentrated close to the direction of a perfect reflection.

## Modeling Specular Reflections

Each of these terms has a red, green, and blue component. This formula repeats 3 times, one for each component.

$$I_{\text{ref}} \sim k_s I \cos^a \phi$$

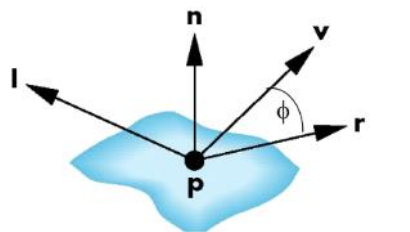
~ means "proportional to".

reflected Intensity

incoming intensity

specular reflectivity coefficient of surface

shininess coefficient of material



The diagram shows a blue irregular shape representing a surface. A point  $p$  is marked on the surface. A normal vector  $n$  points upwards from  $p$ . An incident vector  $I$  points towards  $p$ . A reflected vector  $r$  points away from  $p$ . A viewer vector  $v$  points away from  $p$ . The angle  $\phi$  is shown between the reflected vector  $r$  and the viewer vector  $v$ .

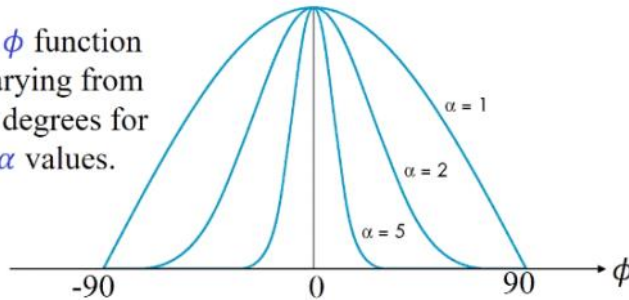
Phong proposed using a term  $\cos^a(\phi)$  that drops off as the angle between the viewer and the ideal reflection increases

## The Shininess Coefficient

Values between:

- **100 and 200** correspond to **metals**
- 5 and 10 give surfaces that look like plastic

The  $\cos^\alpha \phi$  function with  $\phi$  varying from -90 to 90 degrees for different  $\alpha$  values.



If alpha is large, all reflections are concentrated at a small angle

If alpha is 1, the reflections are scattered between -90 and 90 degrees

## Ambient Light

Ambient light is the result of *multiple interactions* between large light sources and the objects in the environment

Amount and color depend on both the colour of the light(s) and the material properties of the object

We add  $k_a I_a$  to diffuse and specular terms

- $k$  is the reflection coefficient of the surface
- $I$  is the intensity of the ambient light
- Subscripted  $a$ 's = ambient
- No distance/angle term, as there is no attenuation

# Properties of Light Sources

## 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.

$$\text{Point Source } P_0 = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \text{ Distant Source } P_0 = \begin{bmatrix} x \\ y \\ z \\ 0 \end{bmatrix} \quad \text{Point rep. to Vector rep.}$$



## Phong Model - Shading: Light Sources

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

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. Even though the ambient component of light does not have location, we add it to model how it contributes to the global lighting effect.

Each term has its own red, green and blue components.

Hence, there are **9 coefficients for each point source**:

$$L_{dr}, L_{dg}, L_{db}, L_{sr}, L_{sg}, L_{sb}, L_{ar}, L_{ag}, L_{ab}$$

- We can place these nine coefficients in a 3 x 3 illumination matrix for each ( $i^{\text{th}}$ ) light source:.

- $$\mathbf{L}_i = \begin{bmatrix} L_{ira} & L_{iga} & L_{iba} \\ L_{ird} & L_{igd} & L_{ibd} \\ L_{irs} & L_{igs} & L_{ibs} \end{bmatrix}$$

- In practice, we will use constructs such as:
  - (Vec3 OR Vec4) light\_i\_ambient, light\_i\_diffuse, light\_i\_specular;

## Phong Model - Shading: Reflection

We construct the model by assuming that we can compute how much of each of the incident lights is reflected at the point of interest.

$$R_i = \begin{bmatrix} R_{ira} & R_{iga} & R_{iba} \\ R_{ird} & R_{igd} & R_{ibd} \\ R_{irs} & R_{igs} & R_{ibs} \end{bmatrix}$$

$$I_{ir} = R_{ira}L_{ira} + R_{ird}L_{ird} + R_{irs}L_{irs} \\ = I_{ira} + I_{ird} + I_{irs}.$$

$$I_r = \sum_i (I_{ira} + I_{ird} + I_{irs}) + I_{ar},$$

$$I = I_a + I_d + I_s = L_a R_a + L_d R_d + L_s R_s,$$

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  $R_{ird}L_{ird}$

$I$  = Overall reflected Red light = Ambient + Diffuse + Specular

# Blinn-Phong Model

Tuesday, 7 April 2020 2:11 PM

Shading: Material Properties

Surfaces of objects have their material properties to interact with the light source properties

- Nine absorption/reflection coefficients:

$k_{dr}, k_{dg}, k_{db}, k_{sr}, k_{sg}, k_{sb}, k_{ar}, k_{ag}, k_{ab}$	Diffuse, Specular, Ambient X RGB
--	----------------------------------

- Shininess coefficient:  $\alpha$

## 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 ( $R_a$ )

- $R_a = k_a$
- $0 \leq k_a \leq 1$
- $I_a = k_a L_a$ 
  - $L_a$  is the ambient component of the light source

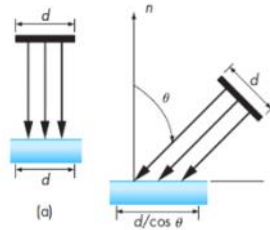
## Diffuse Term

$$R_d \propto \cos \theta$$

$$\cos \theta = l \cdot n$$

$$I_d = L_d R_d = L_d (l \cdot n) k_d$$

$$I_d = \frac{k_d}{a + b d + c d^2} (l \cdot n) L_d$$



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.

Theta is the angle between the normal and the incident light.

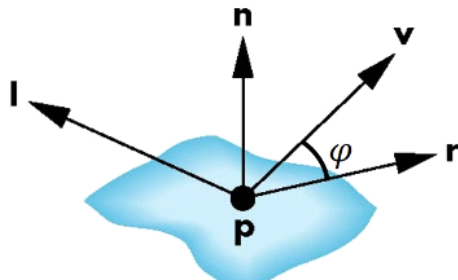
$k_d$  is material property for diffuse light

We divide it by the distance adjustment

## Specular Term

$$I_s = L_s R_s = k_s L_s \cos^\alpha \varphi$$

$$I_s = k_s L_s (\mathbf{r} \cdot \mathbf{v})^\alpha$$



Phi is the angle between  $\mathbf{r}$  and  $\mathbf{v}$   
If Phi is small, the value of the component increases, making the surface appear more specular

Specular reflection is mirror-like reflection from smooth surfaces

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

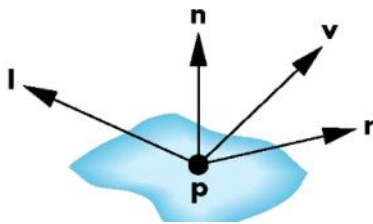
Phong proposed an approximate model that can be computed with only a slight increase over the work done for diffuse surfaces.

## Shading: Adding Components

For each light source and each color component, the Phong model can be written as:

- $I = k_d L_d (\mathbf{l} \cdot \mathbf{n}) + k_s L_s (\mathbf{v} \cdot \mathbf{r})^\alpha + k_a L_a$

- We add up the Diffuse + Specular + Ambient reflection terms
- Diffuse and specular components would normally have distance terms
- For each colour component, we add contributions from all light sources
- We must calculate for each light source
- There are 3 such equations: one for red, one for green, and one for blue.





## Computing the Perfect Reflector

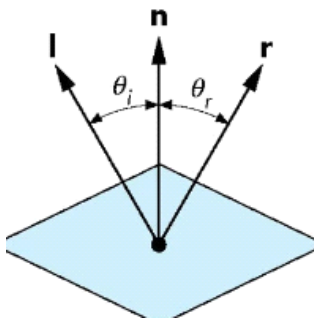
To compute the shading value  $I$  on the previous slide, we need to know those vectors in the model.

The normal vector  $\mathbf{n}$  is determined by the local orientation at point  $\mathbf{p}$ .

Vector  $\mathbf{l}$  and  $\mathbf{v}$  are specified by the application (Light source and viewer location)

We can compute  $\mathbf{r}$  from  $\mathbf{l}$  and  $\mathbf{n}$

- We want: *Angle of incidence* ( $\theta_i$ ) = *Angle of reflection* ( $\theta_r$ )
- The three vectors  $\mathbf{l}$ ,  $\mathbf{n}$  and  $\mathbf{r}$  must be coplanar.
- It is easy to verify that:  $\mathbf{r} = 2(\mathbf{l} \cdot \mathbf{n}) \mathbf{n} - \mathbf{l}$



## The Blinn-Phong Model

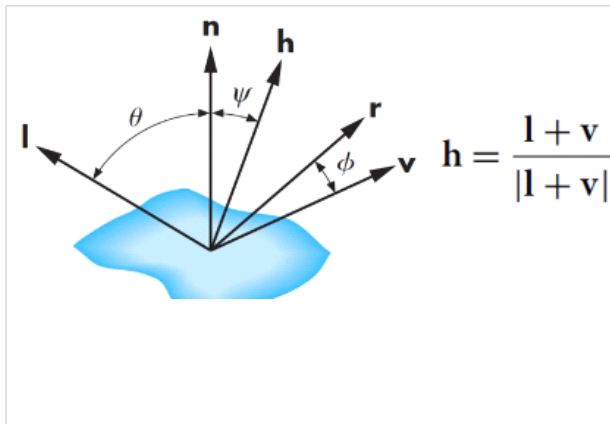
The specular term in the Phong model is problematic because it requires the calculation of a new reflection vector  $\mathbf{r}$  for each vertex.

**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").

**The Blinn Lighting model is the default shading model in OpenGL**

## The Halfway Vector



Instead of computing  $\mathbf{r}$ , the  $\mathbf{h}$  vector which is a normalized vector halfway between  $\mathbf{l}$  (light source position) and  $\mathbf{v}$  (viewer position) is used

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

Note that halfway angle is half of angle between  $\mathbf{r}$  and  $\mathbf{v}$  if vectors are coplanar

The resulting model is known as the **modified Phong/Blinn lighting model**, written as (without the distance terms):

$$I = k_d L_d (\mathbf{l} \cdot \mathbf{n}) + k_s L_s (\mathbf{n} \cdot \mathbf{h})^\beta + k_a L_a$$

## Computing the Normal Vector

The vector  $l$  can be easily computed by subtracting the light source position from a point. Similarly, the vector  $v$  can be computed by subtracting the camera position from a point.

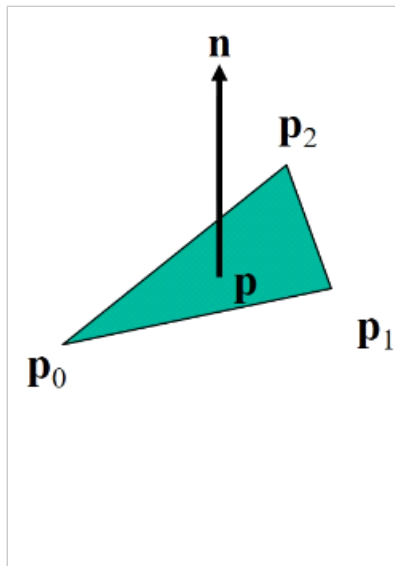
However, whether we need to use the Phong model or the Blinn model, we still need to normal vector  $n$  at each point  $p$ .

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.

## Normal for a Plane



Equation of plane:

$$ax + by + cz + d = 0$$

A plane is determined by either:

- Three points ( $p_0, p_1, p_2$ )
- A normal vector  $n$  and  $p_0$

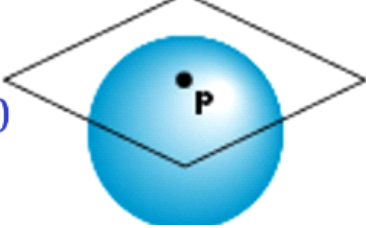
The normal vector formula:

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

We then normalize  $n$  to a unit vector

## Normal for an Implicit Sphere

Implicit representation of a sphere (with unit radius and center at the origin)

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


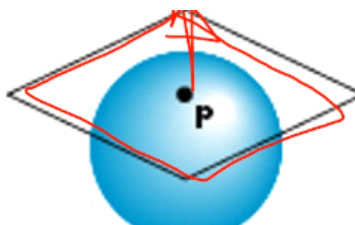
The normal vector ***n*** at a point ***p*** is given by the **gradient of *f* at *p***

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

## Normal for a Parametric Sphere

Parametric representation of a sphere:

$$\begin{aligned}x &= x(u, v) = \cos u \sin v \\y &= y(u, v) = \cos u \cos v \\z &= z(u, v) = \sin u\end{aligned}$$



The normal vector is the normal vector of the tangent plane at the point

The tangent plane is determined by the vectors:

$\frac{\partial \mathbf{p}}{\partial u} = [\partial x / \partial u, \partial y / \partial u, \partial z / \partial u]^T$	Gradient of p with respect to u (first equation) and v (second equation)
$\frac{\partial \mathbf{p}}{\partial v} = [\partial x / \partial v, \partial y / \partial v, \partial z / \partial v]^T$	

The normal vector is given by the cross product of two:

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

## Normal in General

We can also compute the normal vectors for other simple surfaces:

- Quadrics
- Parametric polynomial surfaces (e.g. Bezier surface patches (Chapter 10))

In general, the 3D surface data generated by most 3D scanners contain both vertex coordinates and vertex normals.

Everything would be represented by triangles and you would take the normal of each triangle