Interactive Computer Graphics

- The Graphics Pipeline: *Illumination and Shading*

Some slides adopted from
F. Durand and B. Cutler, MIT

The Graphics Pipeline

Modelling Transformations ← Input:
- geometric model
- illumination model
- camera model
- viewport

Illumination (Shading)

Viewing Transformation (Perspective / Orthographic)

Clipping

Projection (to Screen Space)

Scan Conversion (Rasterization)

Visibility / Display → Output: 2D image for framebuffer display

The Graphics Pipeline

Modelling Transformations

Illumination (Shading)

Viewing Transformation (Perspective / Orthographic)

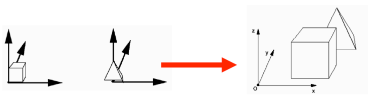
Clipping

Projection (to Screen Space)

Scan Conversion (Rasterization)

Visibility / Display

- 3D models are defined in their own coordinate system
- Modeling transformations orient the models within a common coordinate frame (world coordinates)



Object space World space

The Graphics Pipeline

Modelling Transformations

Illumination (Shading)

Viewing Transformation (Perspective / Orthographic)

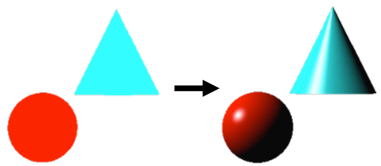
Clipping

Projection (to Screen Space)

Scan Conversion (Rasterization)

Visibility / Display

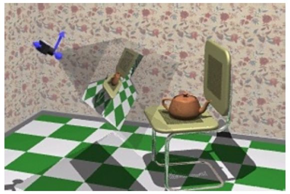
- Vertices are lit (shaded) according to material properties, surface properties and light sources
- Uses a local lighting model



The Graphics Pipeline

- Modelling Transformations
- Illumination (Shading)
- **Viewing Transformation (Perspective / Orthographic)**
- Clipping
- Projection (to Screen Space)
- Scan Conversion (Rasterization)
- Visibility / Display

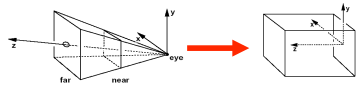
- Maps world space to eye (camera) space
- Viewing position is transformed to origin and viewing direction is oriented along some axis (typically z)



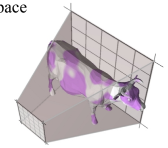
The Graphics Pipeline

- Modelling Transformations
- Illumination (Shading)
- Viewing Transformation (Perspective / Orthographic)
- **Clipping**
- Projection (to Screen Space)
- Scan Conversion (Rasterization)
- Visibility / Display

- Transforms to Normalized Device Coordinates
- Portions of the scene outside the viewing volume (view frustum) are removed (clipped)



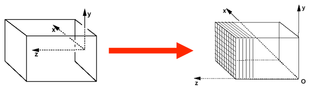
Eye space NDC



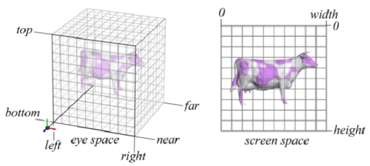
The Graphics Pipeline

- Modelling Transformations
- Illumination (Shading)
- Viewing Transformation (Perspective / Orthographic)
- Clipping
- **Projection (to Screen Space)**
- Scan Conversion (Rasterization)
- Visibility / Display

- The objects are projected to the 2D imaging plane (screen space)



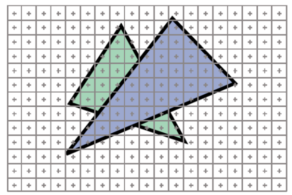
NDC Screen Space



The Graphics Pipeline

- Modelling Transformations
- Illumination (Shading)
- Viewing Transformation (Perspective / Orthographic)
- Clipping
- Projection (to Screen Space)
- **Scan Conversion (Rasterization)**
- Visibility / Display

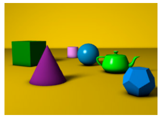
- Rasterizes objects into pixels
- Interpolate values inside objects (color, depth, etc.)

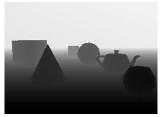


The Graphics Pipeline

- Modelling Transformations
- Illumination (Shading)
- Viewing Transformation (Perspective / Orthographic)
- Clipping
- Projection (to Screen Space)
- Scan Conversion (Rasterization)
- Visibility / Display

• Handles occlusions
• Determines which objects are closest and therefore visible





The Physics of shading

• If we look at a point on an object we perceive a colour and a shading intensity that depends on the various characteristics of the object and the light sources that illuminate it.

• For the time being we will consider only the brightness at each point. We will extend the treatment to colour later.

The Physics of shading

• Object properties:
  – Looking at a point on an object we see the reflection of the light that falls on it. This reflection is governed by:
  1. The position of the object relative to the light sources
  2. The surface normal vector
  3. The albedo of the surface (ability to adsorb light energy)
  4. The reflectivity of the surface
• Light source properties:
  – The important properties of the light source are
  1. Intensity of the emitted light
  2. The distance to the point on the surface

Radiometry

• Energy of a photon

$$e_\lambda = \frac{hc}{\lambda} \qquad h \approx 6.63 \cdot 10^{-34} J \cdot s \quad c \approx 3 \cdot 10^{8} m/s$$

• Radiant Energy of *n photons*

$$Q = \sum_{i=1}^{n} \frac{hc}{\lambda_i}$$

• Radiation flux (electromagnetic flux, radiant flux)
Units: Watts

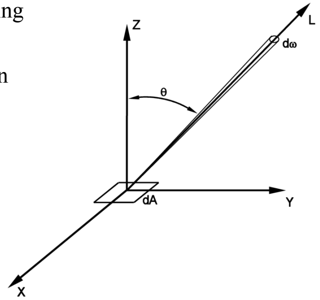
$$\Phi = \frac{dQ}{dt}$$

Radiometry

- **Radiance** – radiant flux per unit solid angle per unit projected area
  - Number of photons arriving per time at a small area from a particular direction

$$L(\omega) = \frac{d^2\Phi}{\cos\theta \; dA \; d\omega}$$

$$Units: \frac{Watt}{meter^2 \; steradian}$$



Radiometry

- **Irradiance** – differential flux falling onto differential area

$$E = \frac{d\Phi}{dA} \qquad Units: \frac{Watt}{meter^2}$$

- Irradiance can be seen as a density of the incident flux falling onto a surface.
- It can be also obtained by integrating the radiance over the solid angle.

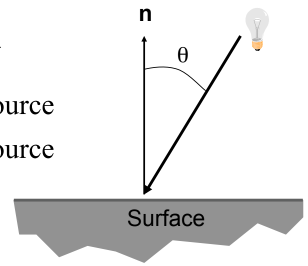
Light Emission

- Light sources: sun, fire, light bulbs etc.
- Consider a point light source that emits light uniformly in all directions

$$E = \frac{\Phi_s \cos\theta}{4\pi d^2} \qquad L = \frac{\Phi_s}{4\pi d^2}$$

Φ_s – power of the light source

d – distance to the light source



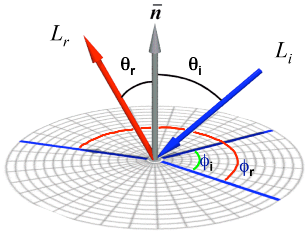
Reflection & Reflectance

- Reflection - the process by which electromagnetic flux incident on a surface leaves the surface without a change in frequency.
- Reflectance – a fraction of the incident flux that is reflected
- We do not consider:
  - absorption, transmission, fluorescence
  - diffraction

Reflectance

- Bidirectional Reflectance Distribution Function (BRDF)

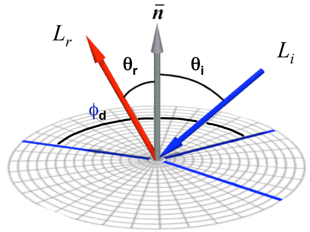
$$f_r(\theta_i,\phi_i,\theta_r,\phi_r) = \frac{dL_r(\theta_r,\phi_r)}{dE_i(\theta_i,\phi_i)} \qquad Units: \frac{1}{steradian}$$



Isotropic BRDFs

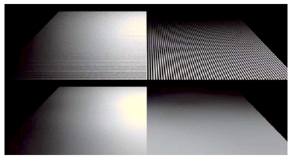
- Rotation along surface normal does not change reflectance

$$f_r(\theta_i,\theta_r,\phi_r-\phi_i) = f_r(\theta_i,\theta_r,\phi_d) = \frac{dL_r(\theta_r,\phi_d)}{dE_i(\theta_i,\phi_d)}$$



Anisotropic BRDFs

- Surfaces with strongly oriented microgeometry elements
- Examples:
  - brushed metals,
  - hair, fur, cloth, velvet



Source: Westin et.al 92

Properties of BRDFs

- Non-negativity

  $$f_r(\theta_i,\phi_i,\theta_r,\phi_r) \geq 0$$

- Energy Conservation

  $$\int_\Omega f_r(\theta_i,\phi_i,\theta_r,\phi_r)\, d\mu(\theta_r,\phi_r) \leq 1 \quad \text{for all } (\theta_i,\phi_i)$$

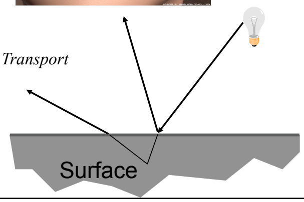
- Reciprocity

  $$f_r(\theta_i,\phi_i,\theta_r,\phi_r) = f_r(\theta_r,\phi_r,\theta_i,\phi_i)$$

Reflectance

- Bidirectional scattering-surface distribution Function (BSSRDF)

Jensen et al. SIGGRAPH 2001
A Practical Model for Subsurface Light Transport



How to compute reflected radiance?

- Continuous version

$$L_r(\omega_r) = \int_\Omega f_r(\omega_i, \omega_r) dE_i(\omega_i) =$$

$$= \int_\Omega f_r(\omega_i, \omega_r)\, dL_i(\omega_i) \cos(\omega_i \cdot n)\, d\omega_i \qquad \omega = (\theta, \phi)$$

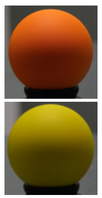
- Discrete version – n point light sources

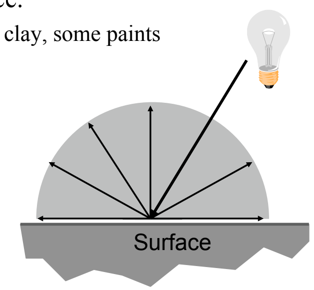
$$L_r(\omega_r) = \sum_{j=1}^{n} f_r(\omega_{ij}, \omega_r) E_j =$$

$$= \sum_{j=1}^{n} f_r(\omega_{ij}, \omega_r) \cos\theta_j \frac{\Phi_{sj}}{4\pi\, d_j^{\,2}}$$

Ideal Diffuse Reflectance

- Assume surface reflects equally in all directions.
- An ideal diffuse surface is, at the microscopic level, a very rough surface.
  - Example: chalk, clay, some paints





Ideal Diffuse Reflectance

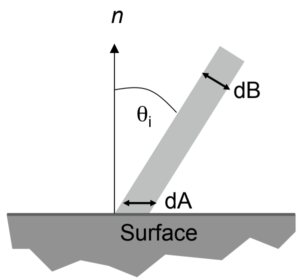
- BRDF value is constant

$$L_r(\omega_r) = \int_\Omega f_r(\omega_i, \omega_r) dE_i(\omega_i) =$$

$$= f_r \int_\Omega dE_i(\omega_i) =$$

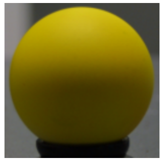
$$= f_r E_i$$

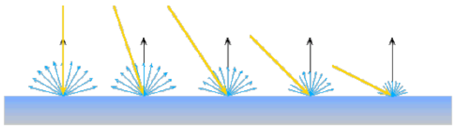
$$dB = dA \cos\theta_i$$



Ideal Diffuse Reflectance

- Ideal diffuse reflectors reflect light according to Lambert's cosine law.

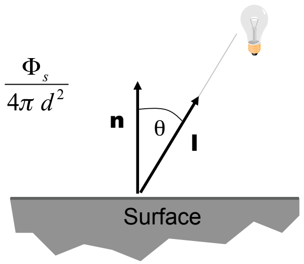




Ideal Diffuse Reflectance

- Single Point Light Source
  - k_d: The diffuse reflection coefficient.
  - **n**: Surface normal.
  - **l**: Light direction.

$$L(\omega_r) = k_d(\mathbf{n} \cdot \mathbf{l}) \frac{\Phi_s}{4\pi\, d^2}$$



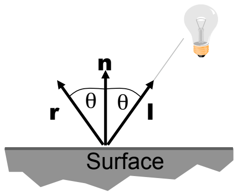
Ideal Diffuse Reflectance – More Details

- If **n** and **l** are facing away from each other, **n • l** becomes negative.
- Using max((**n • l**), 0) makes sure that the result is zero.
  - From now on, we mean max() when we write •.

Do not forget to normalize your vectors for the dot product!

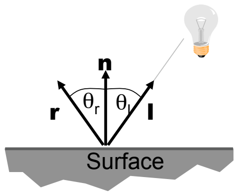
Ideal Specular Reflectance

- Reflection is only at mirror angle.
  - View dependent
  - Microscopic surface elements are usually oriented in the same direction as the surface itself.
  - Examples: mirrors, highly polished metals.



Ideal Specular Reflectance

- Special case of Snell's Law
  - The incoming ray, the surface normal, and the reflected ray all lie in a common plane.



$$n_l \sin\theta_l = n_r \sin\theta_r$$
$$n_l = n_r$$
$$\theta_l = \theta_r$$

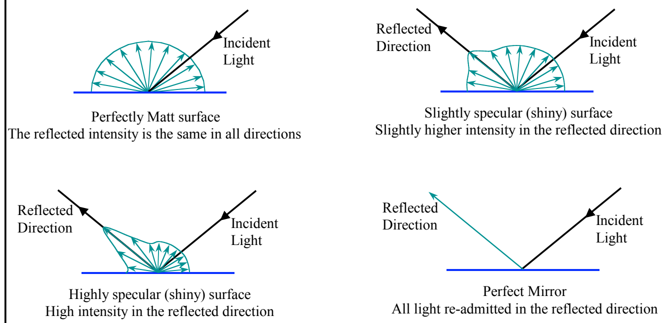
Non-ideal Reflectors

- Snell's law applies only to ideal mirror reflectors.
- Real materials tend to deviate significantly from ideal mirror reflectors.
- They are not ideal diffuse surfaces either …

Non-ideal Reflectors

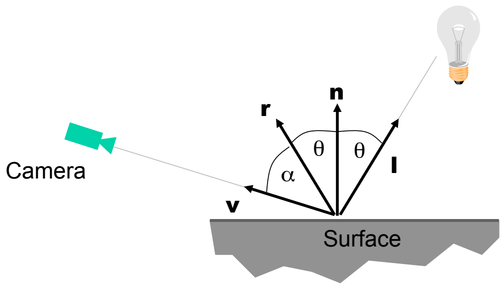
- Simple Empirical Model:
  - We expect most of the reflected light to travel in the direction of the ideal ray.
  - However, because of microscopic surface variations we might expect some of the light to be reflected just slightly offset from the ideal reflected ray.
  - As we move farther and farther, in the angular sense, from the reflected ray we expect to see less light reflected.

Non-ideal Reflectors: Surface Characteristics



The Phong Model

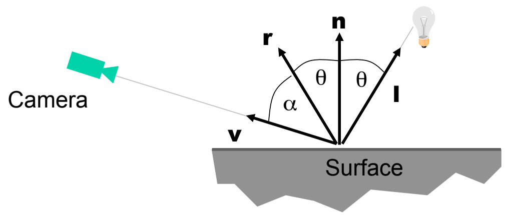
- How much light is reflected?
  - Depends on the angle between the ideal reflection direction and the viewer direction α.



The Phong Model

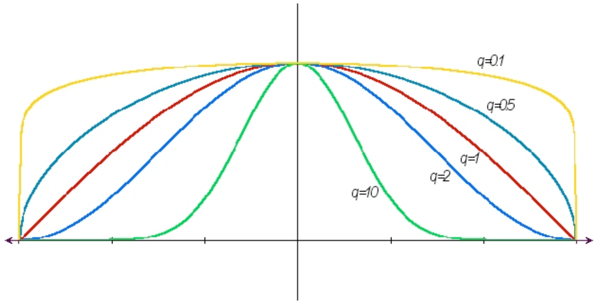
- Parameters
  - k_s: specular reflection coefficient
  - q : specular reflection exponent

$$L(\omega_r) = k_s(\cos\alpha)^q \frac{\Phi_s}{4\pi\, d^2} = k_s(\mathbf{v}\cdot\mathbf{r})^q \frac{\Phi_s}{4\pi\, d^2}$$



The Phong Model

- Effect of the q coefficient



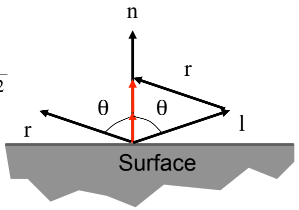
The Phong Model

$$\mathbf{r} + \mathbf{l} = 2 \quad \cos\theta\, \mathbf{n}$$

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

$$L(\omega_r) = k_s(\mathbf{v}\cdot\mathbf{r})^q \frac{\Phi_s}{4\pi\, d^2} =$$

$$= k_s(\mathbf{v}\cdot(2(\mathbf{n}\cdot\mathbf{l})\mathbf{n} - \mathbf{l}))^q \frac{\Phi_s}{4\pi\, d^2}$$

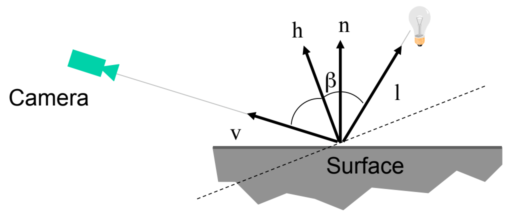


Blinn-Phong Variation

- Uses the halfway vector **h** between **l** and **v**.

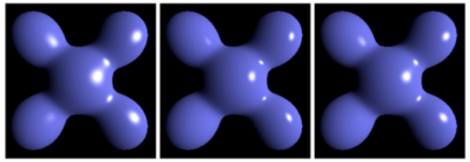
$$\mathbf{h} = \frac{\mathbf{l}+\mathbf{v}}{\|\mathbf{l}+\mathbf{v}\|}$$

$$L(\omega_r) = k_s(\cos\beta)^q \frac{\Phi_s}{4\pi d^2} = k_s(\mathbf{n}\cdot\mathbf{h})^q \frac{\Phi_s}{4\pi d^2}$$



Phong vs Blinn-Phong

- The following spheres illustrate specular reflections as the direction of the light source and the coefficient of shininess is varied.

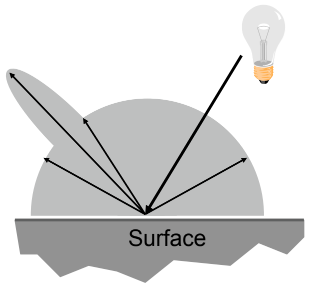


Blinn-Phong | Phong | Blinn-Phong (Lower Exponent)

The Phong Model

- Sum of three components:

  diffuse reflection + specular reflection + ambient.



Ambient Illumination

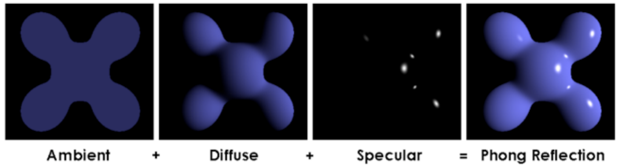
- Represents the reflection of all indirect illumination.
- This is a total hack!
- Avoids the complexity of global illumination.

$$L(\omega_r) = k_a$$

Putting it all together

- Phong Illumination Model

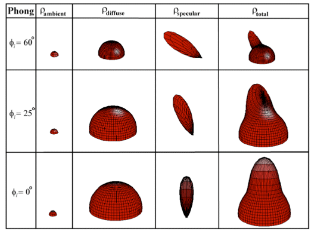
$$L(\omega_r) = k_a + \left(k_d(\mathbf{n}\cdot\mathbf{l}) + k_s(\mathbf{v}\cdot\mathbf{r})^q\right)\frac{\Phi_s}{4\pi d^2}$$



Putting it all together

- Phong Illumination Model

$$L(\omega_r) = k_a + \left(k_d(\mathbf{n}\cdot\mathbf{l}) + k_s(\mathbf{v}\cdot\mathbf{r})^q\right)\frac{\Phi_s}{4\pi d^2}$$



Adding color

- Diffuse coefficients:
  - $k_{d\text{-}red}$, $k_{d\text{-}green}$, $k_{d\text{-}blue}$
- Specular coefficients:
  - $k_{s\text{-}red}$, $k_{s\text{-}green}$, $k_{s\text{-}blue}$
- Specular exponent:

  q

Inverse Square Law

- It is well known that light falls off according to an inverse square law. Thus, if we have light sources close to our polygons we should model this effect.

$$L(\omega_r) = k_a + \left(k_d(\mathbf{n}\cdot\mathbf{l}) + k_s(\mathbf{v}\cdot\mathbf{r})^q\right)\frac{\Phi_s}{4\pi d^2}$$

where d is the distance from the light source to the object

Heuristic Law

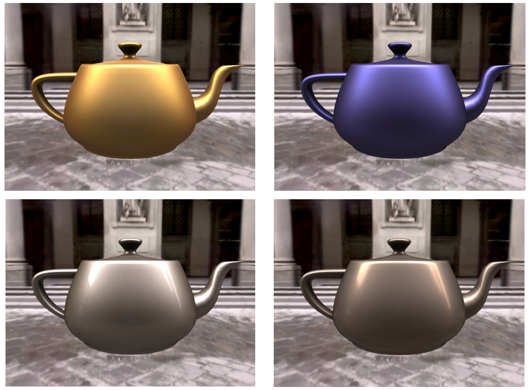
- Although physically correct the inverse square law does not produce the best results.
- Instead the following is often used:

$$L(\omega_r) = k_a + \left(k_d(\mathbf{n}\cdot\mathbf{l}) + k_s(\mathbf{v}\cdot\mathbf{r})^q\right)\frac{\Phi_s}{4\pi(d+s)}$$

where s is an heuristic constant.

- One might be tempted to think that light intensity falls off with the distance to the viewpoint, but it doesn't!
- Why not?

Questions?



Using Shading

- There are three levels at which shading can be applied in polygon based systems:

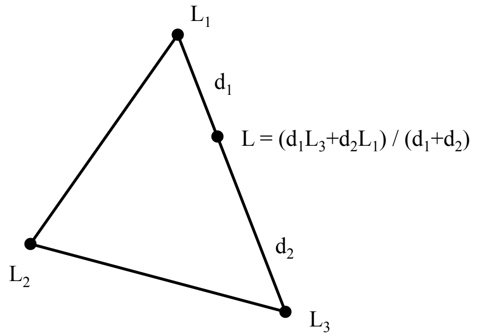
$$\begin{array}{l} \text{Flat Shading} \\ \left.\begin{array}{l}\text{Gouraud Shading} \\ \text{Phong Shading}\end{array}\right\} \text{Interpolation Shading}\end{array}$$

- They provide increasing realism at higher computational cost

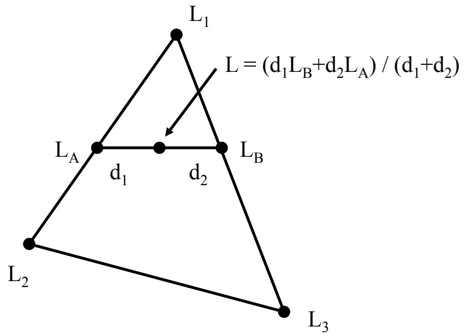
Using Shading

- Flat Shading:
  - Each polygon is shaded uniformly over its surface.
  - The shade is computed by taking a point in the centre and the surface normal vector. (Equivalent to a light source at infinity)
  - Usually only diffuse and ambient components are used.
- Interpolation Shading:
  - A more accurate way to render a shaded polygon is to compute an independent shade value at each point.
  - This is done quickly by interpolation:
  1. Compute a shade value at each vertex
  2. Interpolate to find the shade value at the boundary
  3. Interpolate to find the shade values in the middle

Calculating the shades at the edges

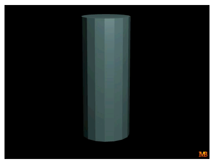


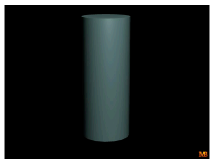
Calculating the internal shades



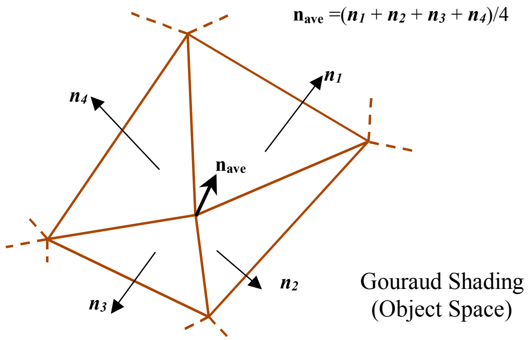
Interpolating over polygons

- In addition to interpolating shades over polygons, we can interpolate them over groups of polygons to create the impression of a smooth surface.
- The idea is to create at each vertex an averaged intensity from all the polygons that meet at that vertex.





Computing an average normal vector at a vertex



Smooth Shading

- Need to have per-vertex normals
- Gouraud Shading
  - Interpolate color across triangles
  - Fast, supported by most of the graphics accelerator cards
  - Can't model specular components accurately, since we do not have the normal vector at each point on a polygon.
- Phong Shading
  - Interpolate normals across triangles
  - More accurate modelling of specular compoents, but slower.

Interpolation of the 3D normals

- We may express any point for this facet in parametric form:

$$\mathbf{P} = \mathbf{V_1} + \mu_1(\mathbf{V_2} - \mathbf{V_1}) + \mu_2(\mathbf{V_3}-\mathbf{V_1})$$

- The average normal vector at the same point may be calculated as the vector **a**:

$$\mathbf{a} = \mathbf{n_1} + \mu_1(\mathbf{n_2} - \mathbf{n_1}) + \mu_2(\mathbf{n_3}-\mathbf{n_1})$$

and then

$$\mathbf{n_{average}} = \mathbf{a} \,/\, |\, \mathbf{a} \,|$$

2D or 3D

- The interpolation calculations may be done in either 2D or 3D
- For specular reflections the calculation of the reflected vector and viewpoint vector must be done in 3D.