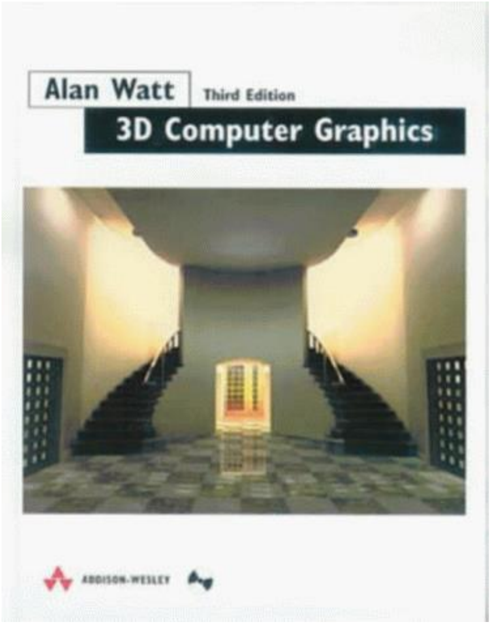
The University Of Sheffield.

COM3503/4503/6503: 3D Computer Graphics

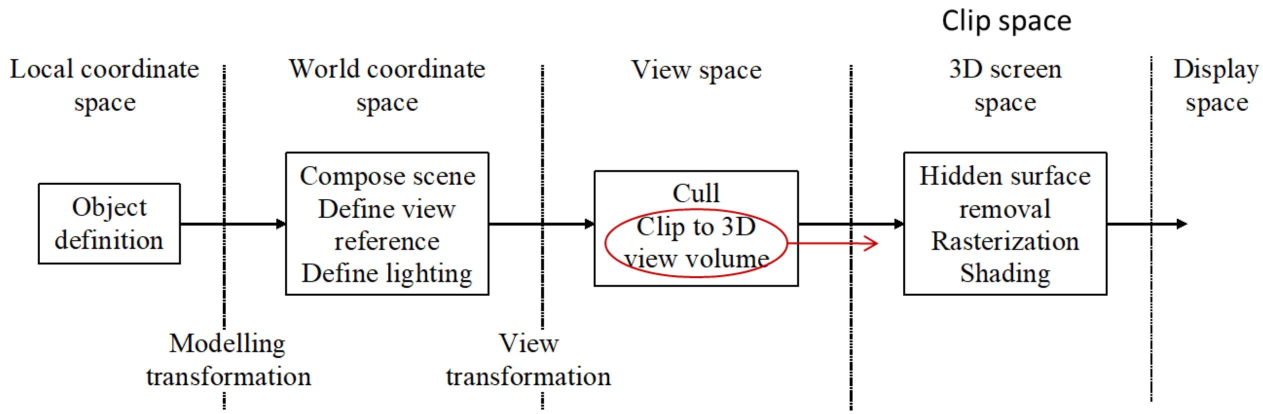
Lecture 7: Rendering: Part 1



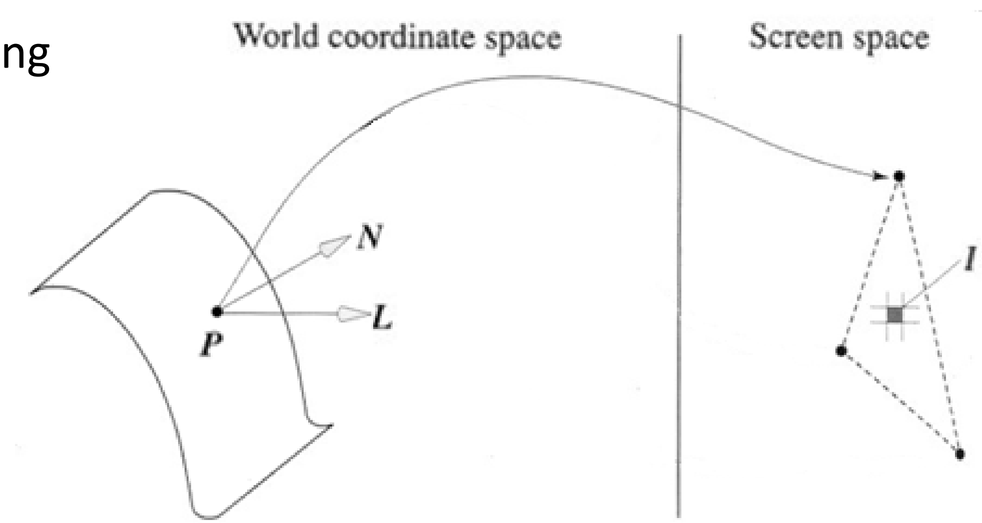
Dr. Steve Maddock
s.maddock@dcs.shef.ac.uk

1. Introduction

- We have looked at the geometrical aspects of the viewing pipeline.



- Today: Illumination and shading



1.1 Local versus global illumination

- Illumination model:
  - Nature of light from a source
- Reflection model:
  - Interaction of light with a surface in terms of the properties of the surface and nature of incident light.
- Today's lecture: local reflection models.
- Later lectures: 'Global' techniques, e.g. ray tracing.

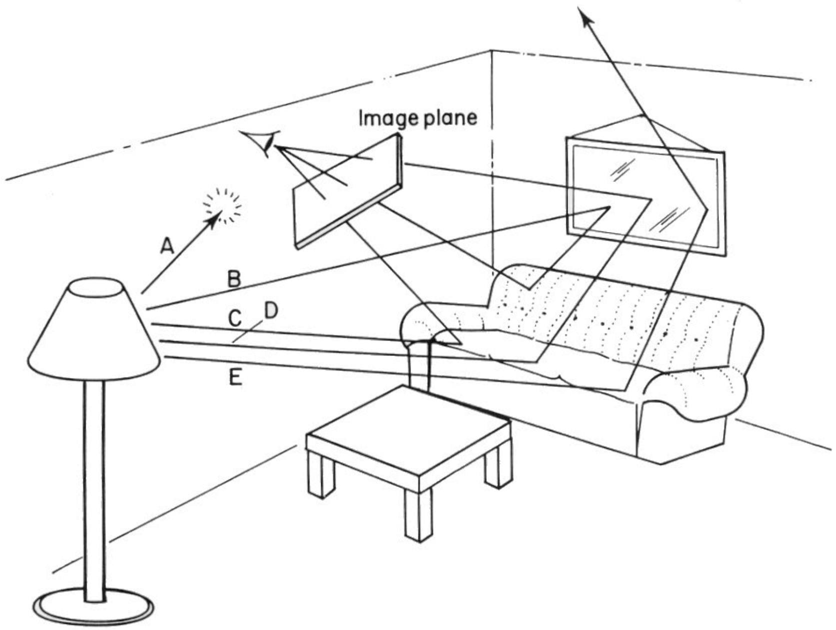
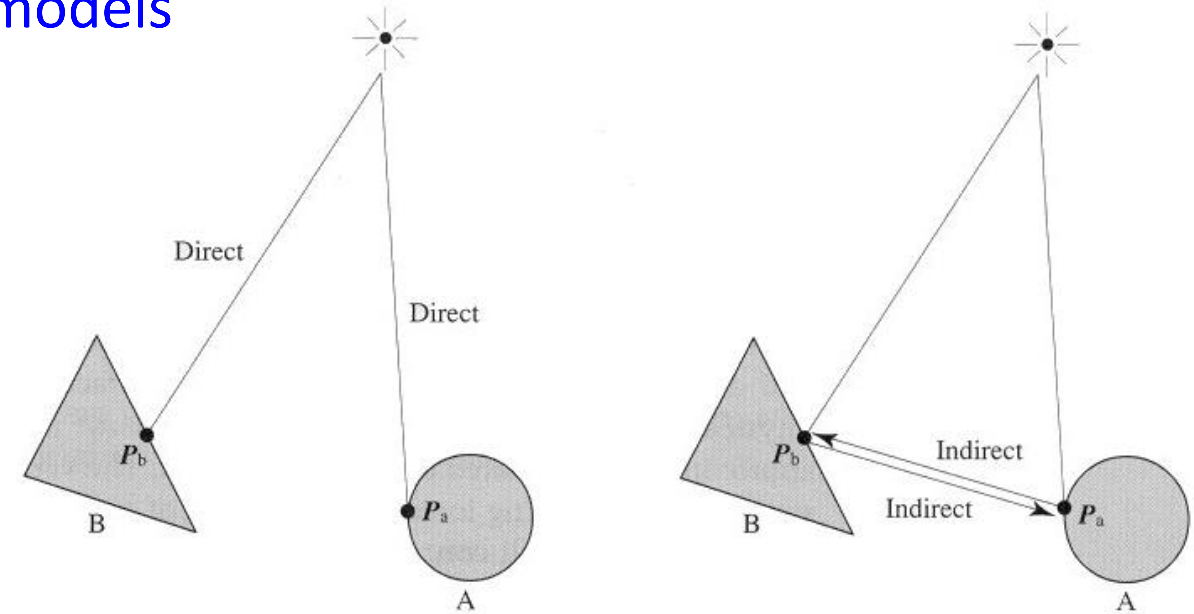


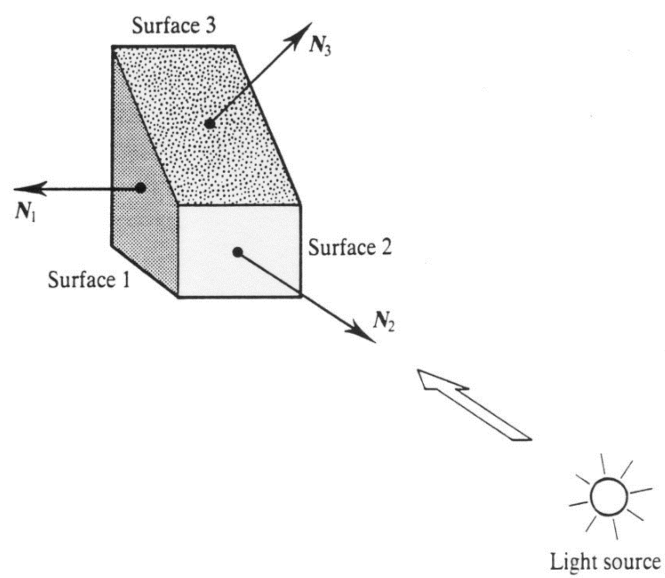
Fig. 5. Some light rays (like A and E) never reach the image plane at all. Others follow simple or complicated routes.

1.2 Local reflection models

- We consider only direct illumination of a surface by a light source

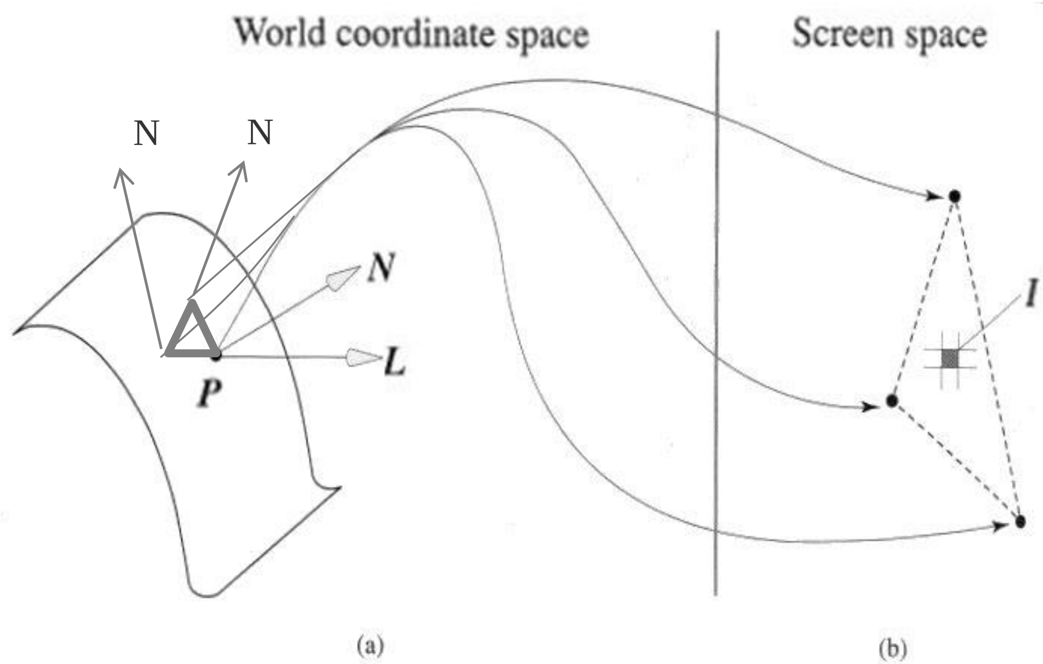


- Surfaces angled towards the light source will be brighter



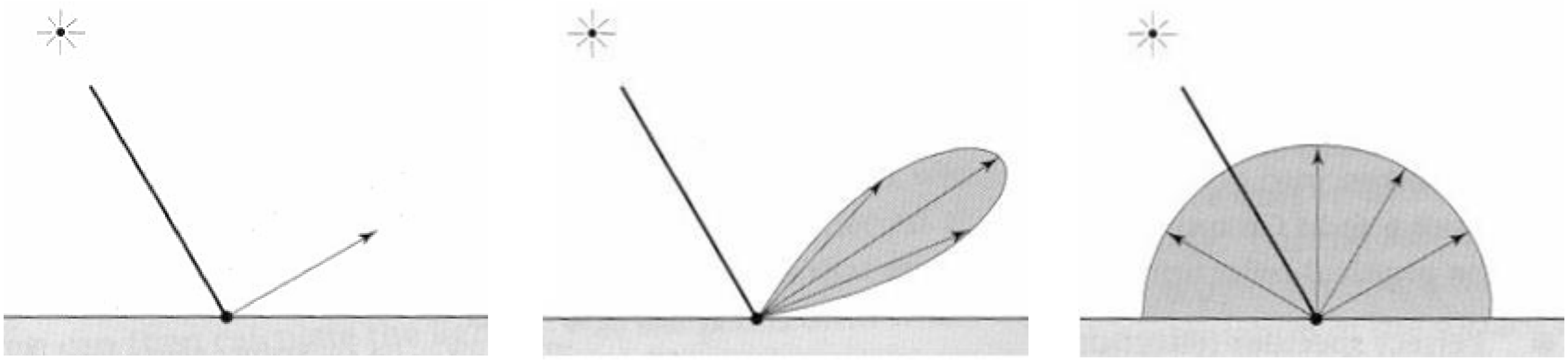
1.3 Overview

- A polygon's vertices are projected from world space into screen space
- The polygon overlaps a collection of pixels
- An intensity is calculated for each pixel
- Thus, the shading algorithm works from screen space
    - Incremental algorithm
- But, the lighting calculations are done in world space



2. Phong illumination (1975)

- Only direct illumination.
- No shadows and no inter-reflection between objects.
- Physical reflection phenomena:
  - Perfect specular reflection
  - Imperfect specular reflection
  - Perfect diffuse reflection (Gouraud, 71)



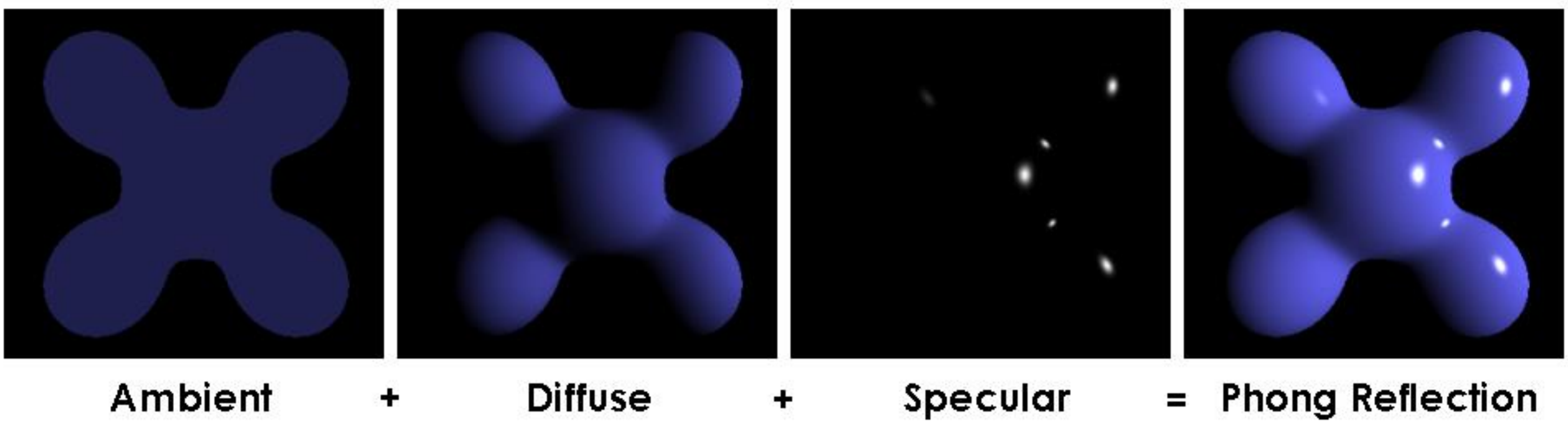
2. Phong illumination (1975)

- (Gouraud, 71: ambient + diffuse only)
- Linear combination of 3 components:

Reflected light = ambient + diffuse + specular

$$I = k_a I_a + k_d I_d + k_s I_s$$

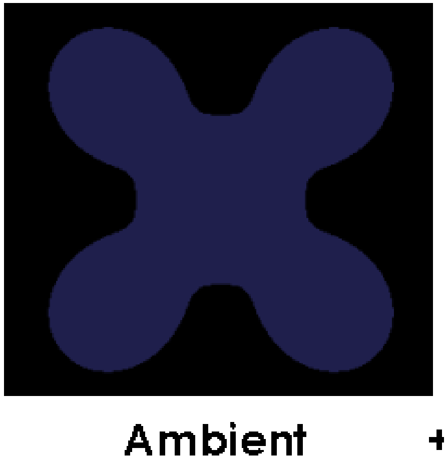
such that $k_a+k_d+k_s=1$



http://en.wikipedia.org/wiki/Phong_shading

2.1 Ambient

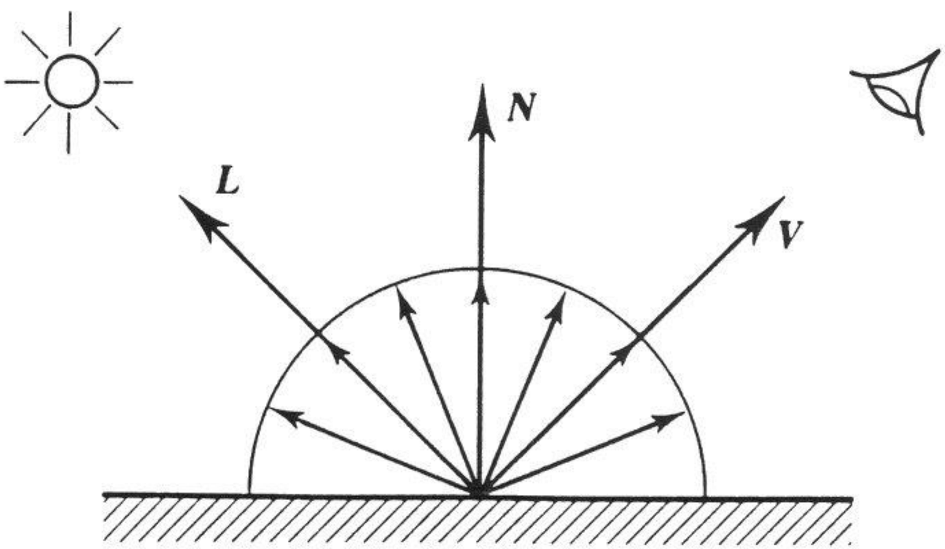
- $I = k_a I_a + k_d I_d + k_s I_s$
- A constant that simulates (approximates) global or indirect illumination.
- Illuminates all surfaces
  - All surfaces visible from the viewpoint, but maybe not the light source.
- Without the ambient term, parts of an object that couldn't see the light would be rendered as black.
- I_a – ambient light intensity
- K_a – fraction of ambient light reflected; essentially defines colour of surface

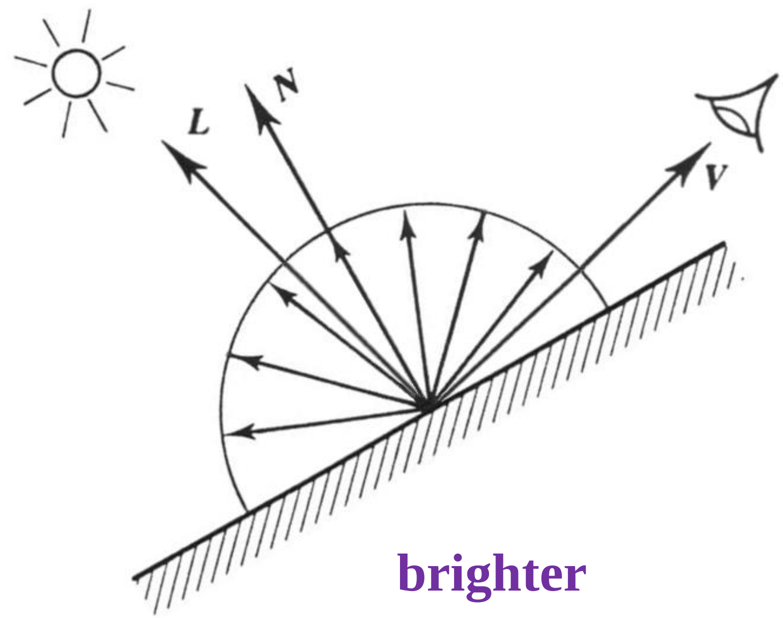


en.wikipedia.org/wiki/Phong_shading

2.2 Diffuse (Lambertian)

- $I = k_a I_a + \mathbf{k_d I_d} + k_s I_s$
- Dull matt surfaces such as chalk appear equally bright from all directions
- The brightness of the surface depends on the normal of the surface





2.2 Diffuse (Lambertian)

- $I = k_a I_a + k_d I_d + k_s I_s$

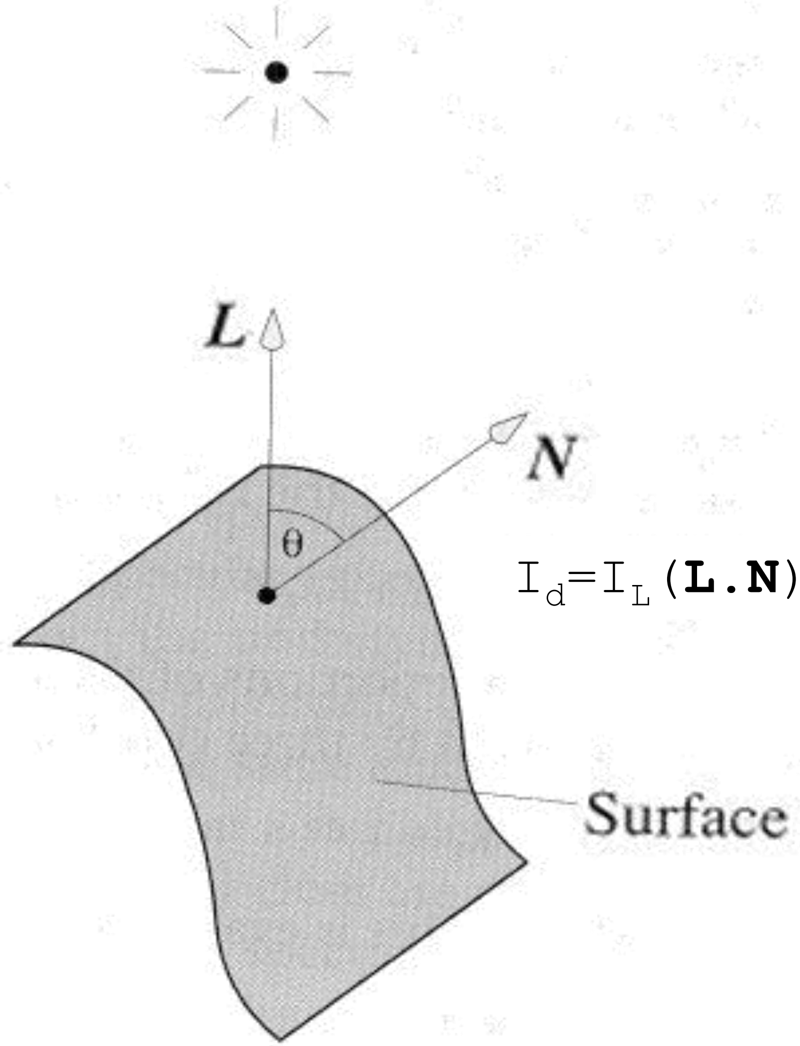
$$I_d = I_L \cos\theta = I_L\ (\mathbf{L.N})$$

$0 \leq \theta \leq \pi$

where

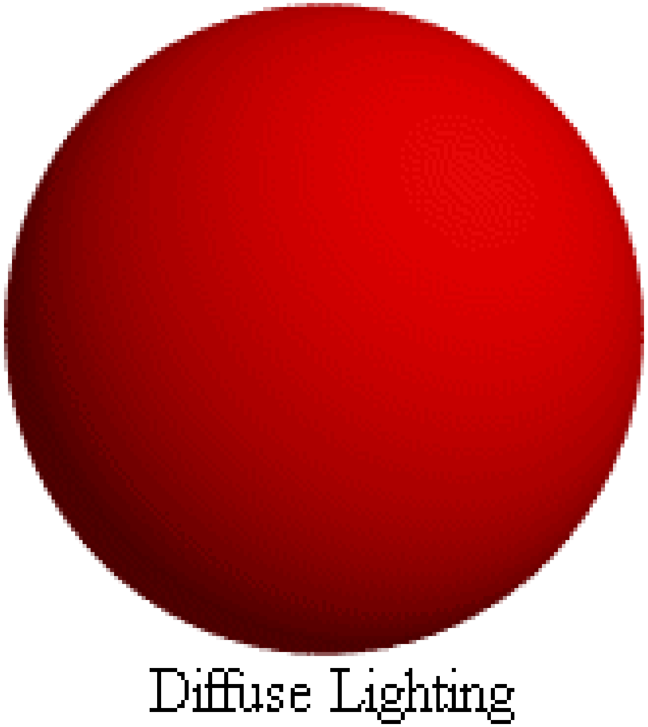
I_L is the intensity of the incident light

θ is the angle between the surface normal **N** at the point of interest and the direction of the light source **L**



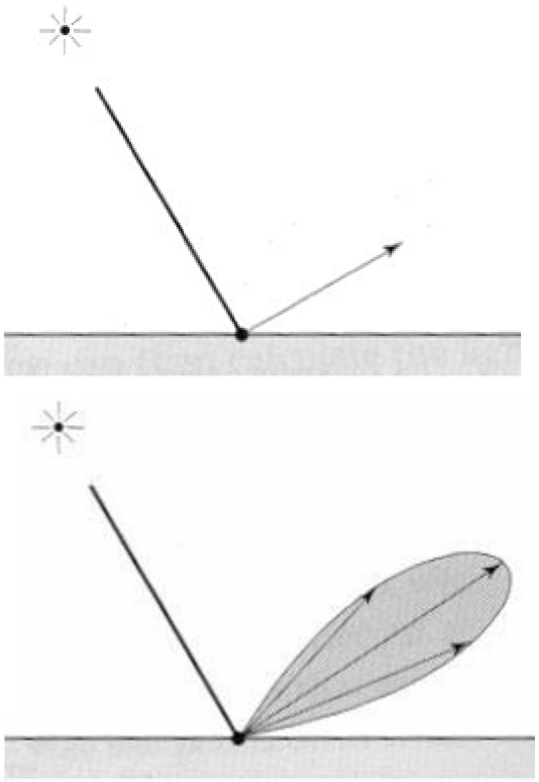
$I_d = I_L\ (\mathbf{L.N})$

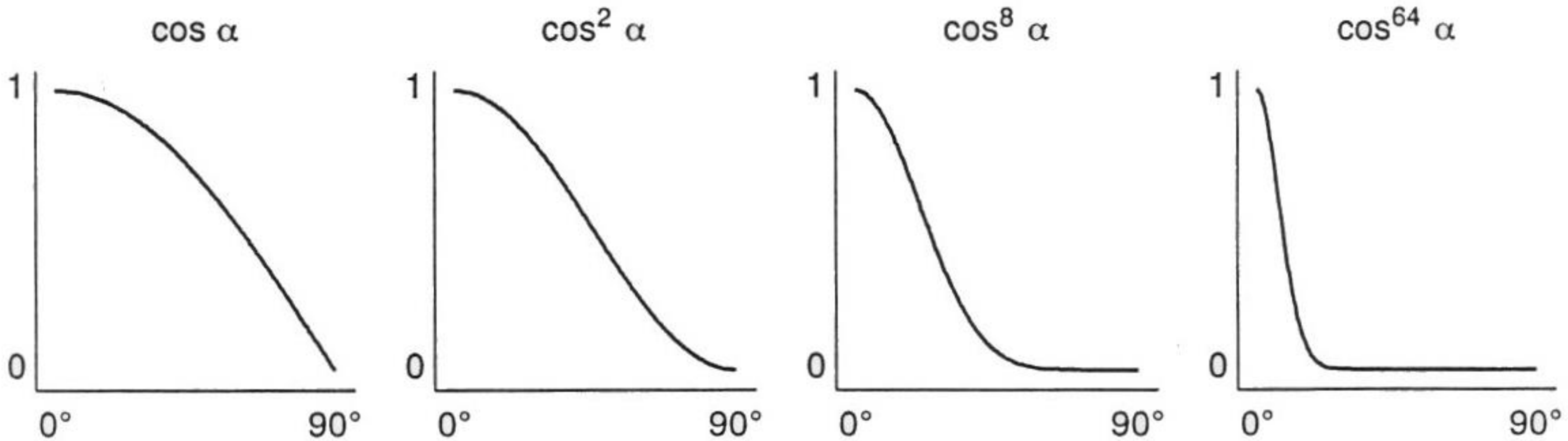
Diffuse Example: Where is the light source?



2.3 Specular

- Perfectly smooth shiny objects reflect light in the mirror direction
- However, most objects are not ideal, so light is reflected in the vicinity of the mirror direction
    - 'Specular lobe'
- The Phong model approximates this using $\cos^n\Omega$





2.3 Specular

$I = k_a I_a + k_d I_d + \color{red}{\mathbf{k_s I_s}}$

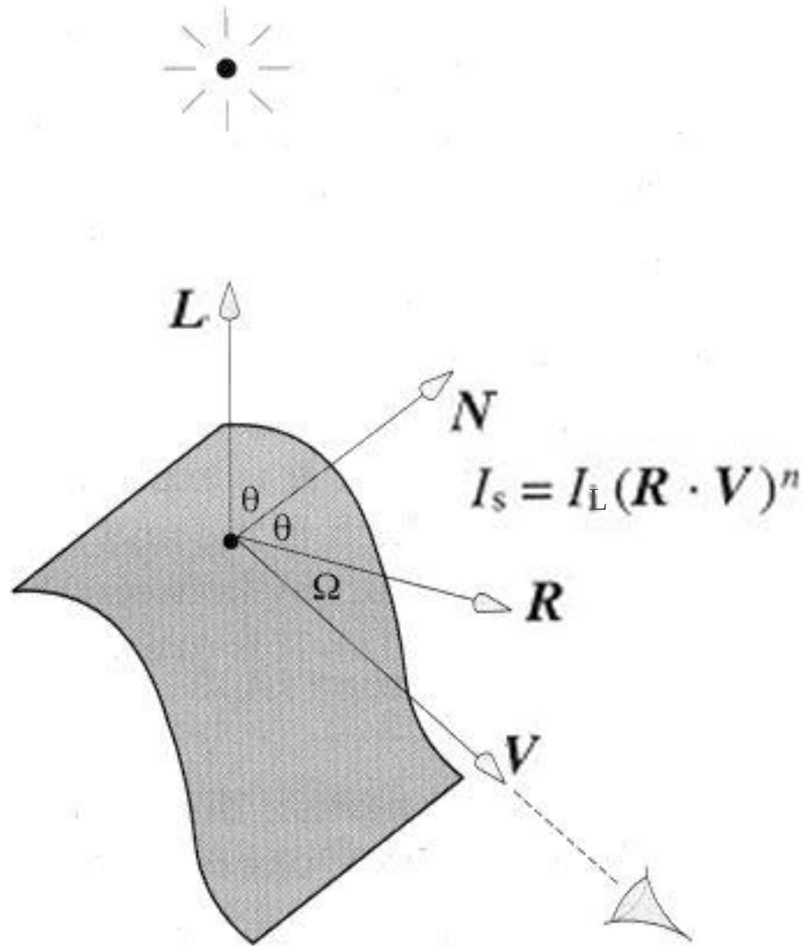
$I_s = I_L \cos^n \Omega = I_L(\mathbf{R.V})^n$

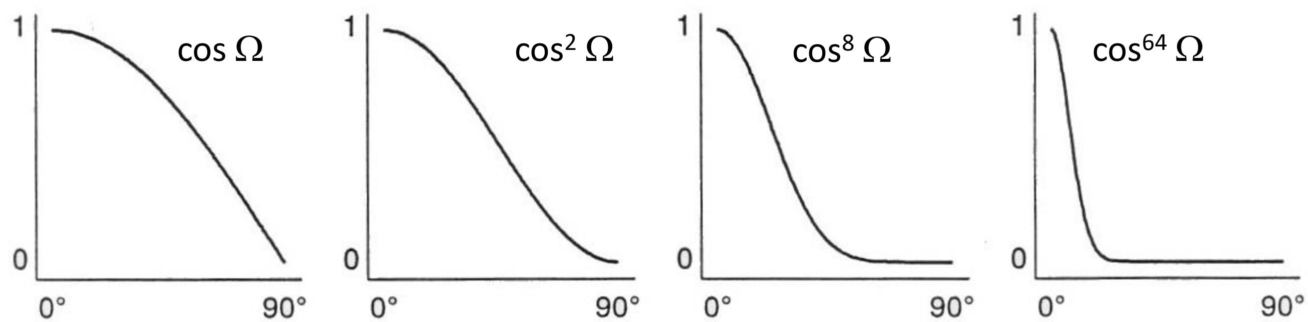
where

I_L is the intensity of the incident light

Ω is the angle between the viewing direction **V** and the mirror direction **R**.

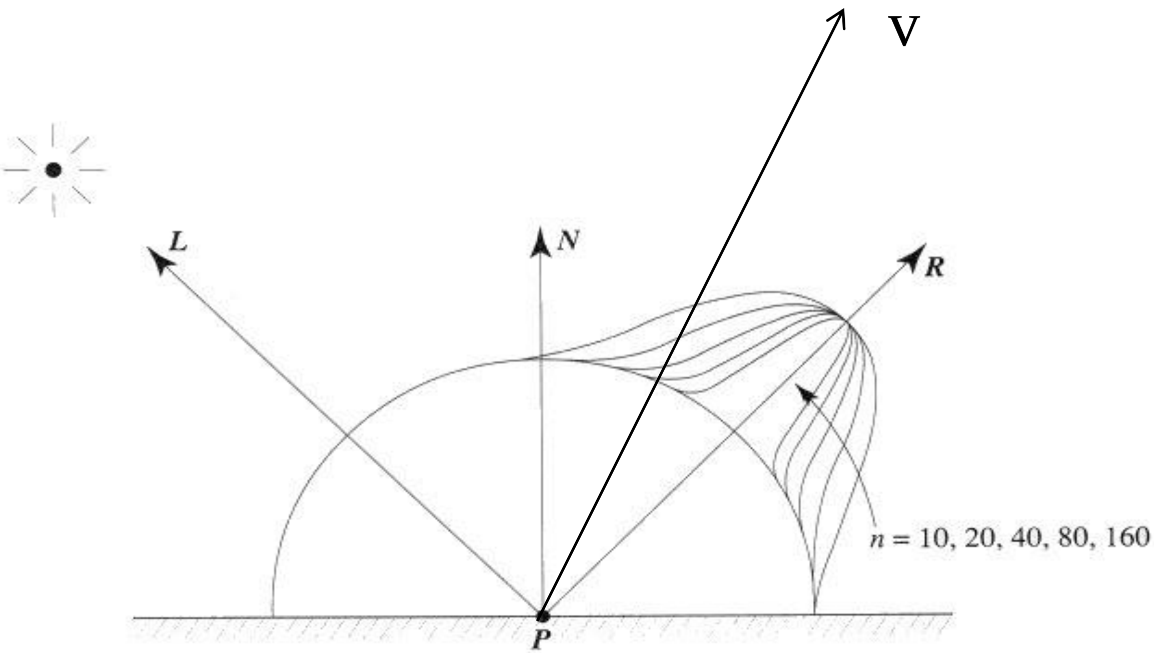
n is an index that simulates degree of imperfection of a surface - decay factor

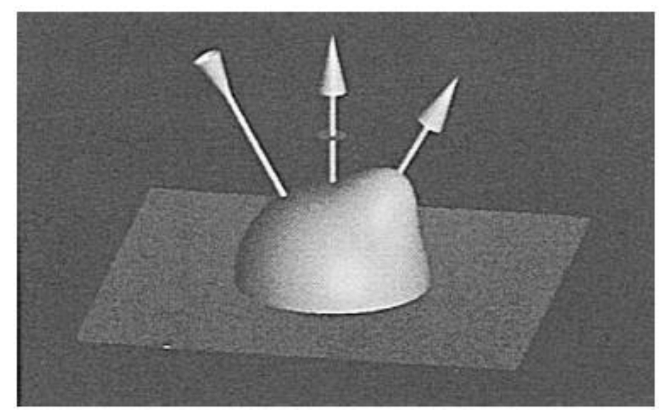




2.4 All three terms

$I = k_a I_a + I_L \left(k_d(\mathbf{L.N}) + k_s(\mathbf{R.V})^n\right)$

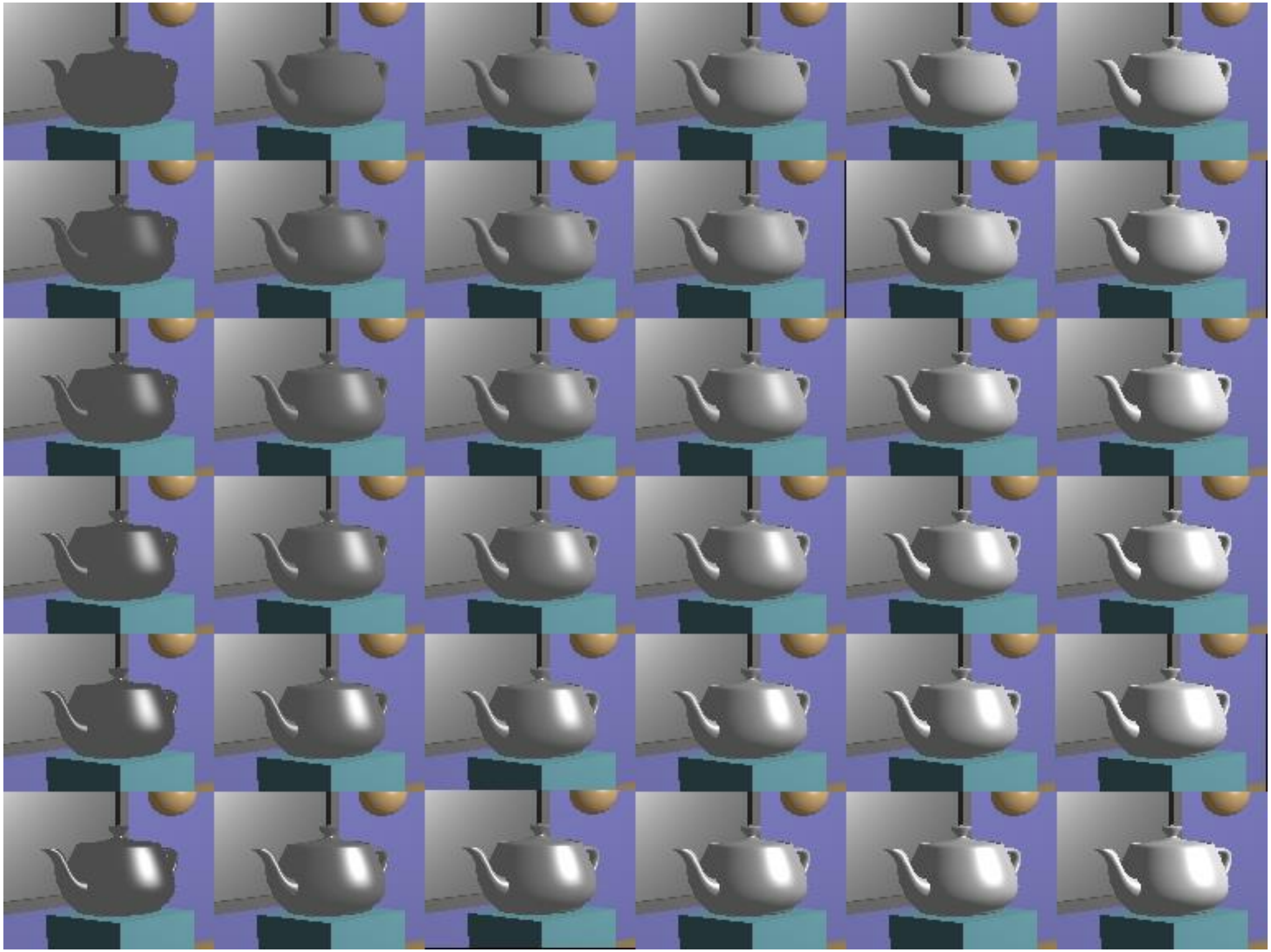




Blinn, J.F., "Models of light reflection for computer synthesized pictures", Proc. Siggraph'77, pp.192-198

$K_d \rightarrow$

$K_s \downarrow$



6x6 array of images showing a Phong shaded teapot with different K_s and K_d ranging from 0.0 to 1.0 in steps of 0.2 (K_a=0.7, n = 10.0 in all images)

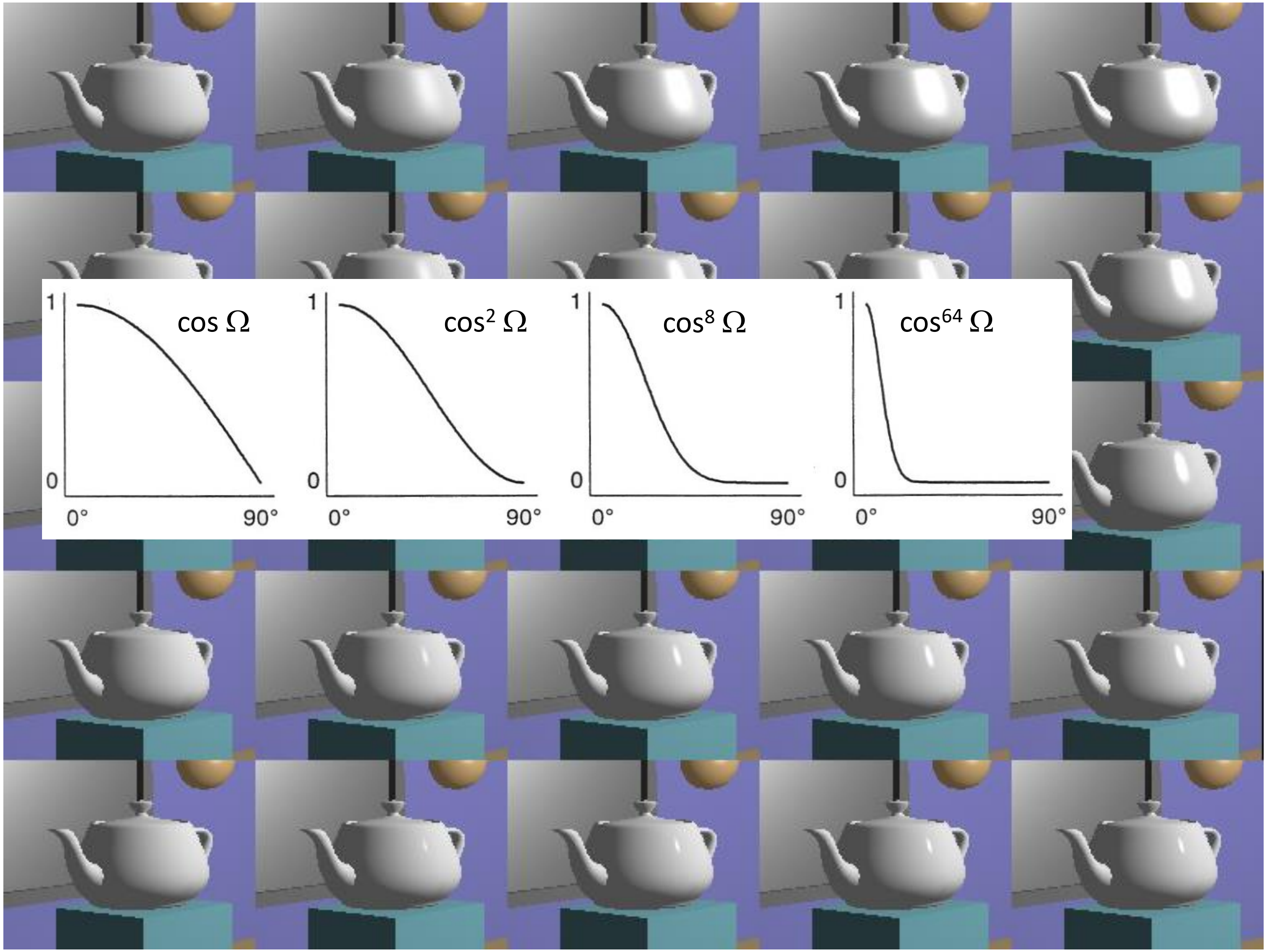
$K_s \rightarrow$

n
$\downarrow$

5x5 array of images showing different K_s and n values (with K_a=0.7, K_d=1.0)

$K_s \rightarrow$

$n \downarrow$



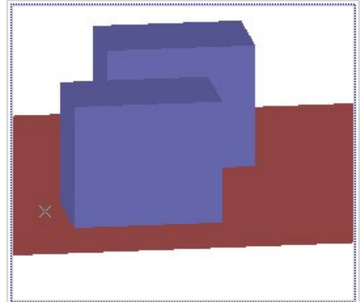
5x5 array of images showing different K_s and n values (with K_a=0.7, K_d=1.0)

2.5 Other

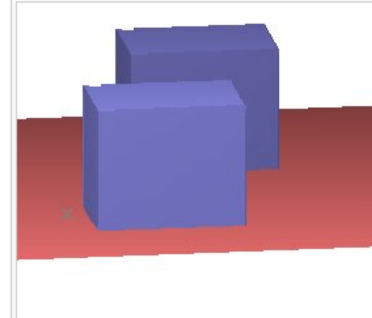
- Heuristic atmospheric attenuation:

$$I = k_a I_a + f_{att} I_L \left(k_d(\mathbf{L.N}) + k_s(\mathbf{R.V})^n\right)$$

where $f_{att}=1/(distFromSurfaceToLight^2)$



Two boxes rendered with an OpenGL renderer. Note that the colors of the two front faces are the same even though one box is further away.



The same model rendered using ARRIS CAD which implements "Distance Falloff" to make surfaces which are closer to the eye appear brighter.

http://en.wikipedia.org/wiki/Flat_shading

- Multiple lights:

$$I = k_a I_a + \sum_{\mathbf{L}} I_{\mathbf{L}} \left(k_d(\mathbf{L_L.N}) + k_s(\mathbf{R_L.V})^n\right)$$

  - Possibility of overflow, i.e. maximum light intensity is breached – need to clamp the value.

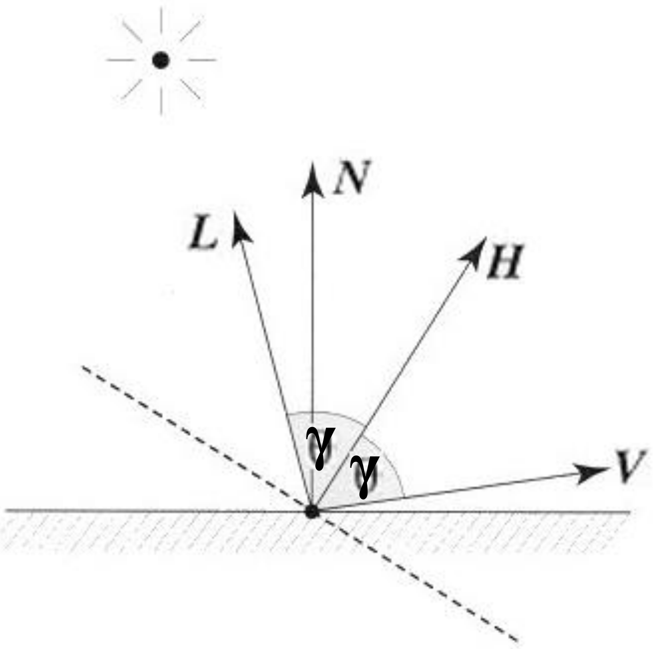
2.6 Practical considerations

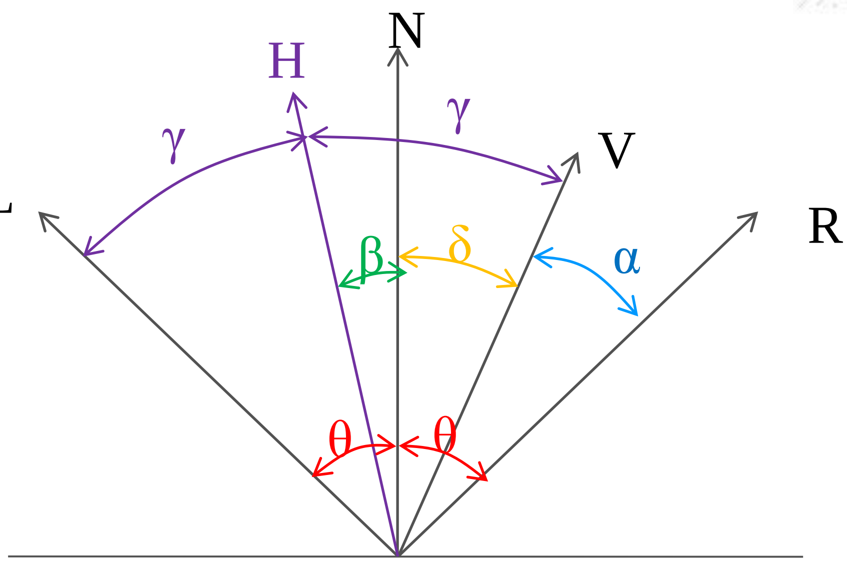
- $I = k_a I_a + I_L (k_d(\mathbf{L}.\mathbf{N}) + k_s(\mathbf{R}.\mathbf{V})^n)$
- Reduce expense of the Phong shading equation by making some geometric simplifications:
  - If light source is at infinity, then **L** is constant
  - If viewpoint is at infinity, then **V** is constant
- Given these simplifications:
  - For geometric transformation: finite viewpoint position
  - For light calculations: infinite viewpoint
  - But it works in practice!!
- Still need to calculate R, which is expensive
  - Solution: Use the halfway vector (Blinn, 77)

James F. Blinn (1977). "Models of light reflection for computer synthesized pictures". Proc. 4th annual conference on computer graphics and interactive techniques: 192.

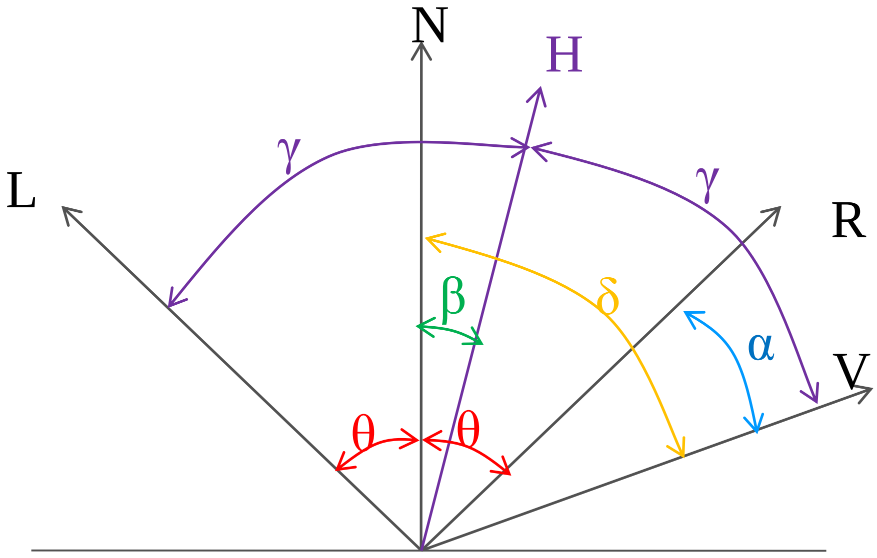
2.6.1 The halfway vector H

- $I = k_a I_a + I_L (k_d(\mathbf{L.N}) + k_s(\mathbf{R.V})^n)$
- Define: $\mathbf{H} = (\mathbf{L}+\mathbf{V})/2$
- The term (**N.H**) varies in the same manner as (**R.V**)





$$\theta = \gamma+\beta = \delta+\alpha$$
$$\gamma = \beta + \delta$$
$$\alpha = 2\beta$$

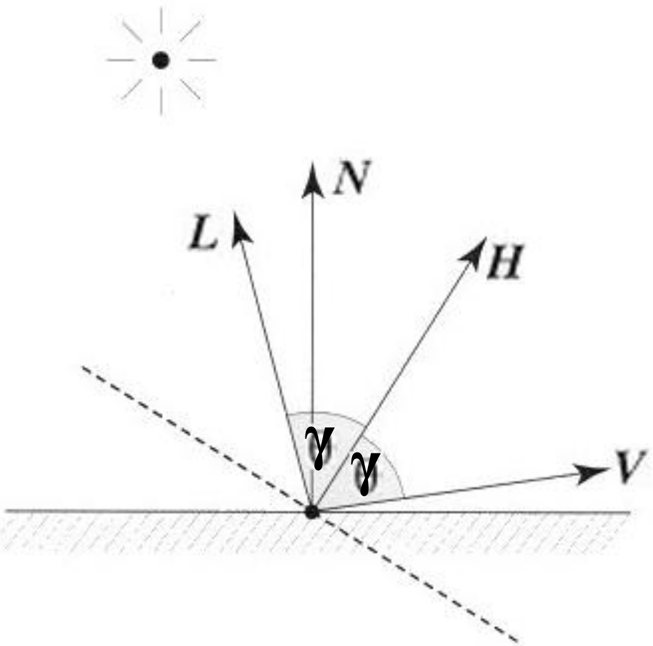


$\theta = \gamma - \beta = \delta - \alpha$

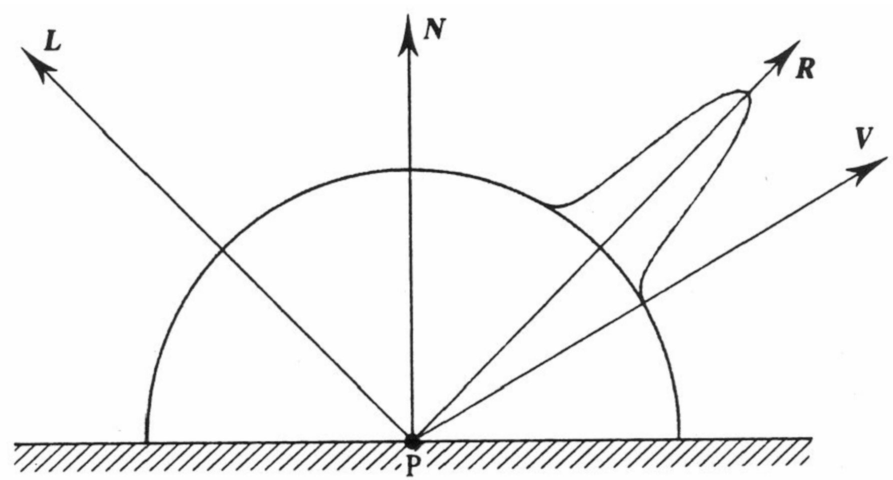
$\gamma = \delta - \beta$

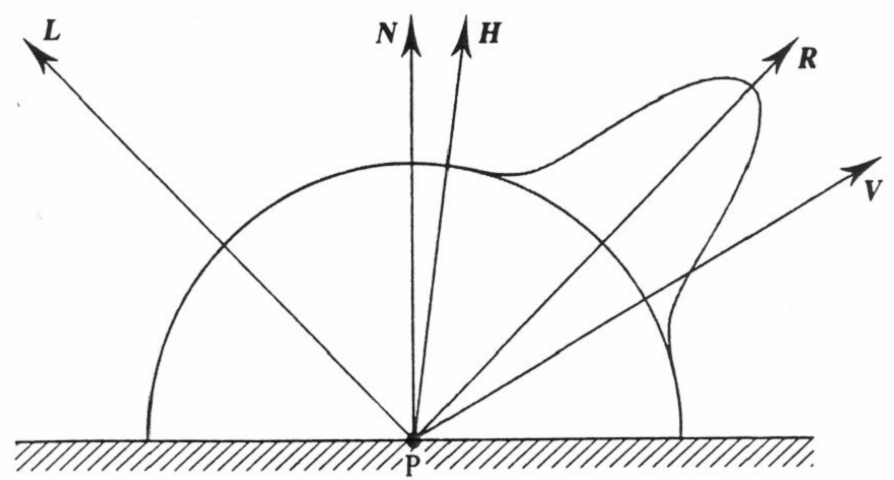
$\alpha = 2\beta$

2.6.1 The halfway vector H



- $I = k_a I_a + I_L (k_d(\mathbf{L}.\mathbf{N}) + k_s(\mathbf{R}.\mathbf{V})^n)$
- Define: $\mathbf{H} = (\mathbf{L}+\mathbf{V})/2$
- The term (**N**.**H**) varies in the same manner as (**R**.**V**)
- Rewrite: $I = k_a I_a + I_L (k_d(\mathbf{L}.\mathbf{N}) + k_s(\mathbf{N}.\mathbf{H})^n)$
- These simplifications mean that $I = f(\mathbf{N})$





2.7 Colour

- We use three colour components of the intensity: I_r, I_g, I_b
- For a white light, k_s is equal in all three equations:

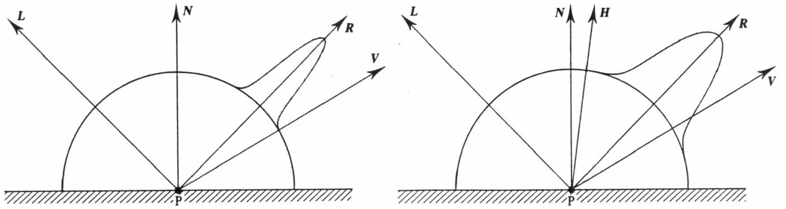
$$I_r = k_{ar}I_a + I_L\ (k_{dr}(\mathbf{L.N}) + k_s(\mathbf{N.H})^n)$$

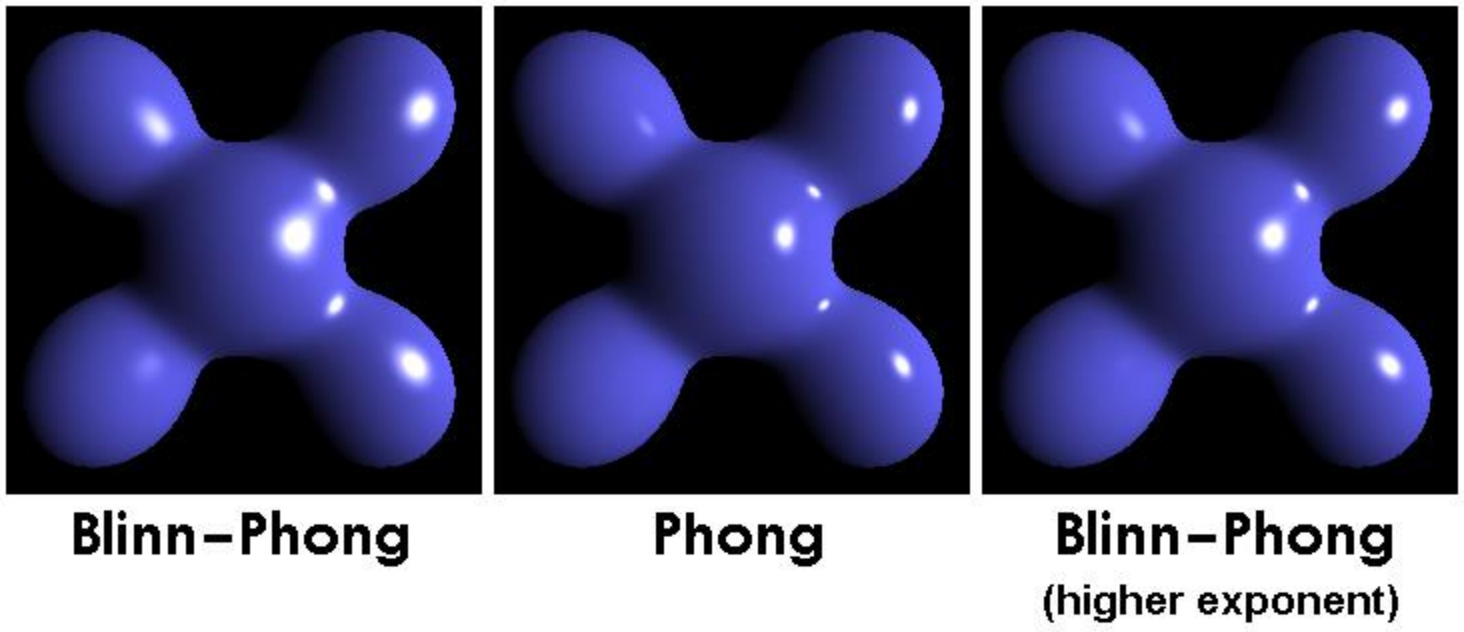
$$I_g = k_{ag}I_a + I_L\ (k_{dg}(\mathbf{L.N}) + k_s(\mathbf{N.H})^n)$$

$$I_b = k_{ab}I_a + I_L\ (k_{db}(\mathbf{L.N}) + k_s(\mathbf{N.H})^n)$$

2.8 Phong and Blinn-Phong

- $I = k_a I_a + I_L (k_d(\mathbf{L.N}) + k_s(\mathbf{R.V})^n)$
- $I = k_a I_a + I_L (k_d(\mathbf{L.N}) + k_s(\mathbf{N.H})^n)$



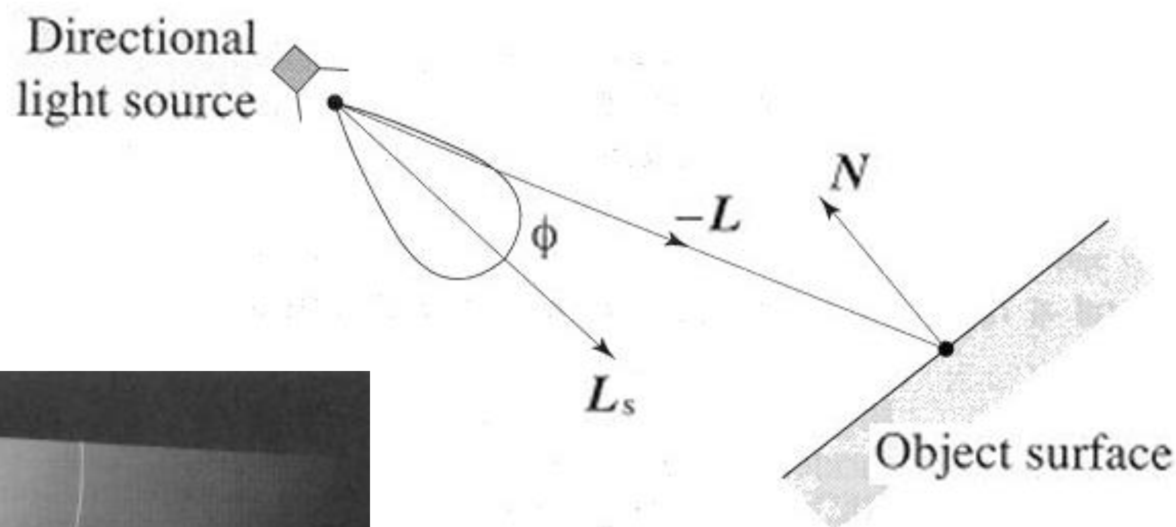


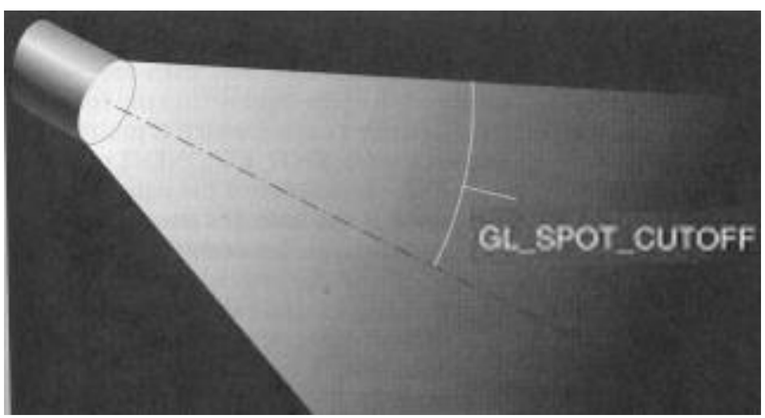
Demonstration of Blinn-Phong versus Phong reflection models, created by the uploader Brad Smith, August 7, 2006.http://en.wikipedia.org/wiki/Blinn%E2%80%93Phong_shading_model

3. Light source considerations

- (Warn, 83) provides us with a simple directional light source to simulate a spotlight:

  $I_L = I_s(-L.L_s)^m$

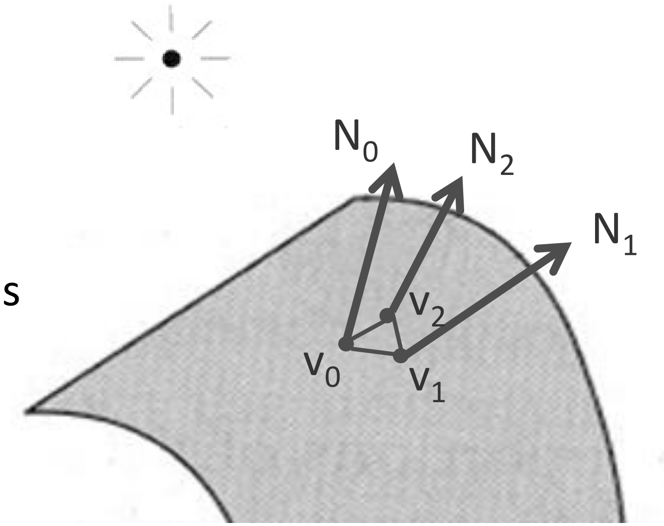


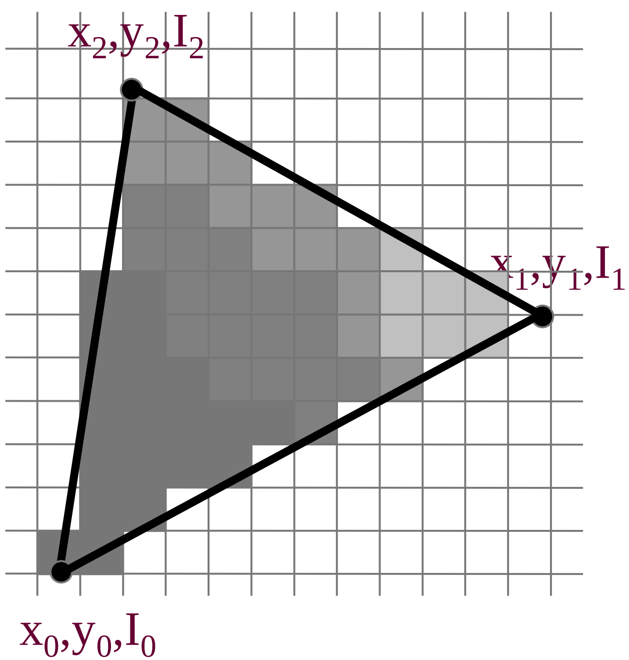


Warn,D.R. (1983). Lighting controls for synthetic images. Computer Graphics, 17(3), pp.13-21

4. Gouraud interpolative shading

- Lighting calculations done in world space
  - Need a normal at each vertex (see previous lectures)
  - I = f(N)
  - 1971: Gouraud: ambient and diffuse
  - 1975 onwards: Phong: ambient, diffuse and specular
- Each vertex now has an intensity
- Each vertex is projected to screen space
- Now need to calculate which pixels are covered by the triangle and what the intensity of each is





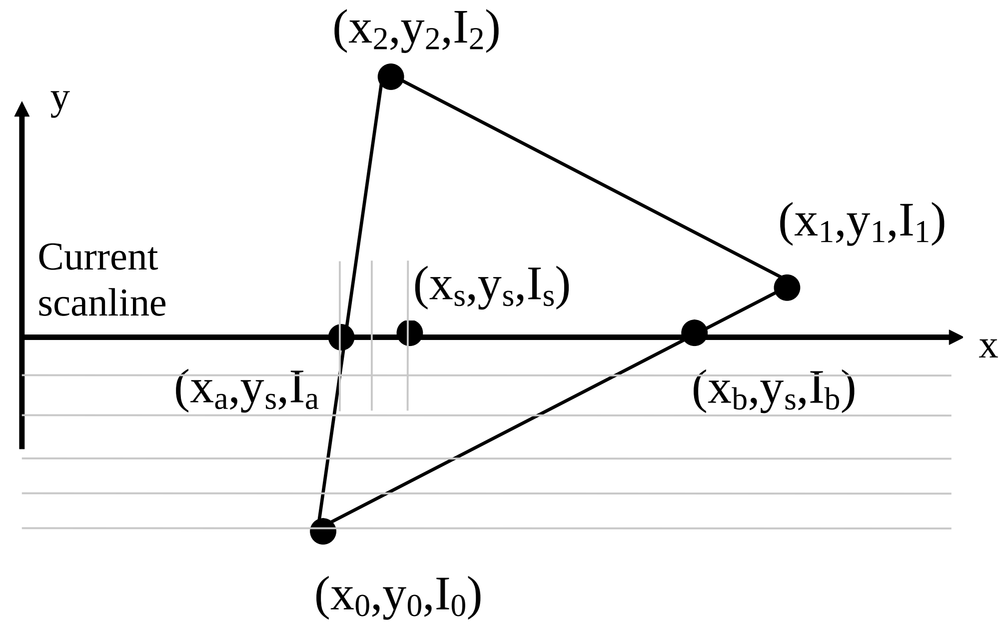
4. Gouraud interpolative shading

- Bilinearly interpolate vertex intensities over the polygon to give the intensity at each pixel
- Weighted averages:

$$I_a = \frac{1}{(y_2 - y_0)}\left(I_2(y_s - y_0) + I_0(y_2 - y_s)\right)$$

$$I_b = \frac{1}{(y_1 - y_0)}\left(I_1(y_s - y_0) + I_0(y_1 - y_s)\right)$$

$$I_s = \frac{1}{(x_b - x_a)}\left(I_a(x_b - x_s) + I_b(x_s - x_a)\right)$$

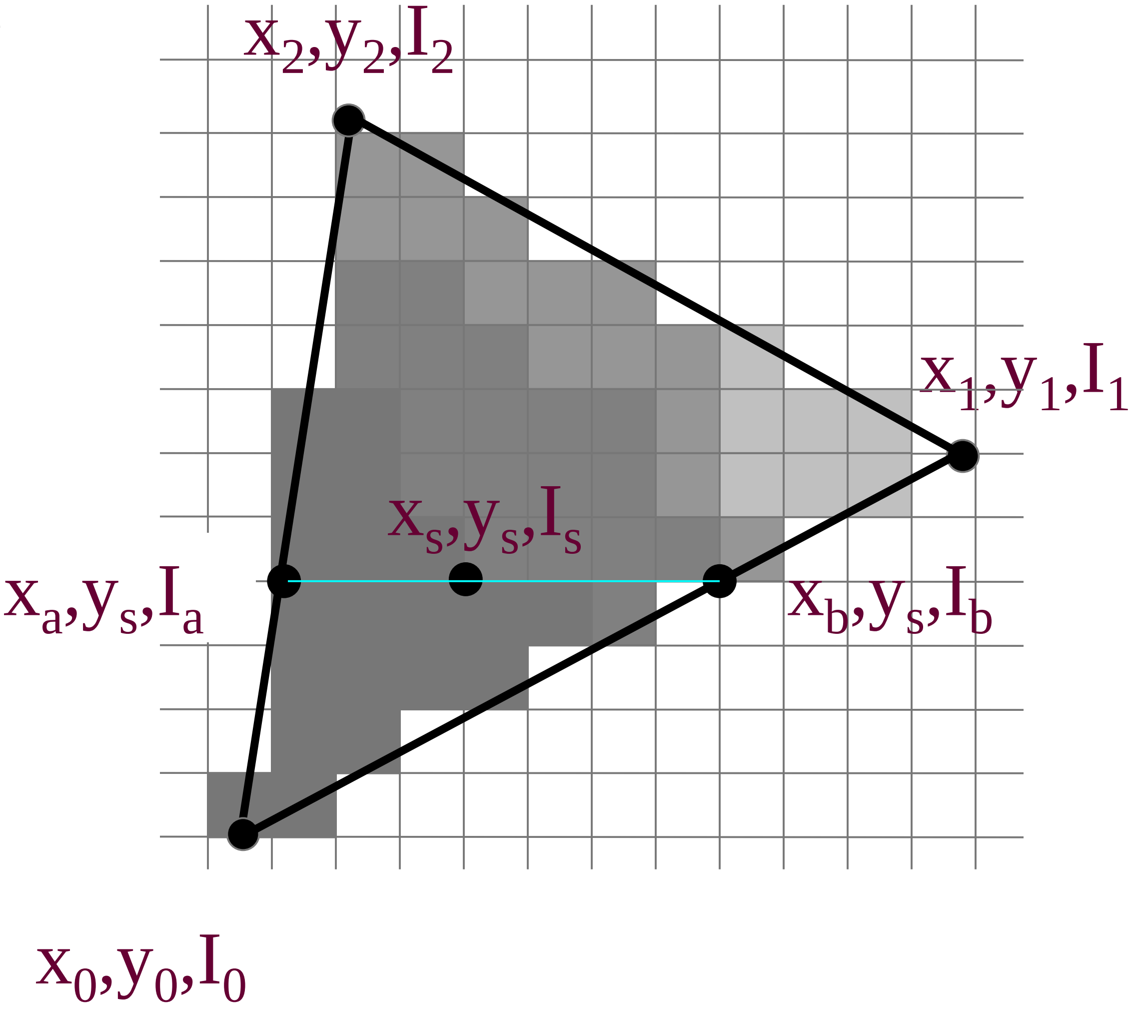


- Alternatively, we can use incremental forms, e.g. for I_s

$$I_{s,n} = I_{s,n-1} + \Delta I_s \qquad \Delta I_s = \frac{\Delta x}{x_b - x_a}(I_b - I_a)$$

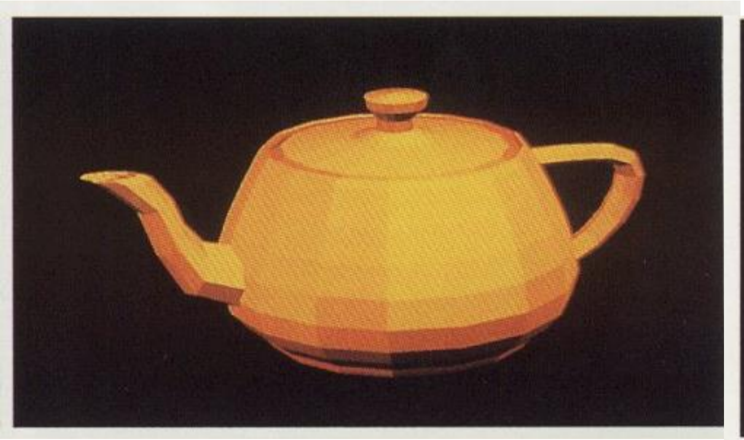
4. Gouraud interpolative shading

- Bilinearly interpolate the intensity value
- Fast integer-based interpolations are used

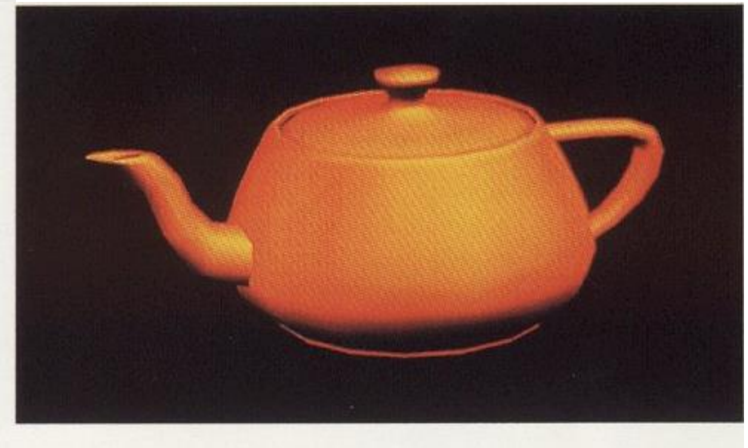


4. Gouraud interpolative shading

- This process generally fools the eye into not seeing the boundary edges between polygons



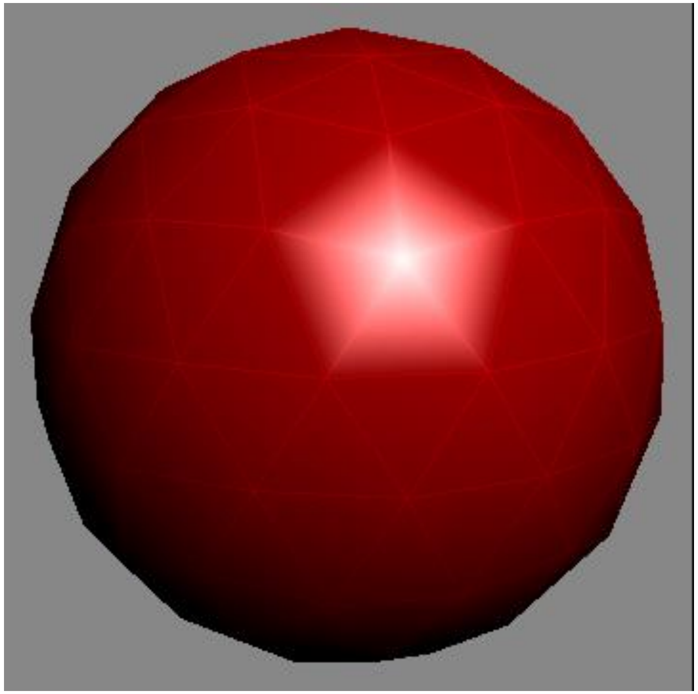
Flat shading – single colour for whole polygon calculated using polygon normal

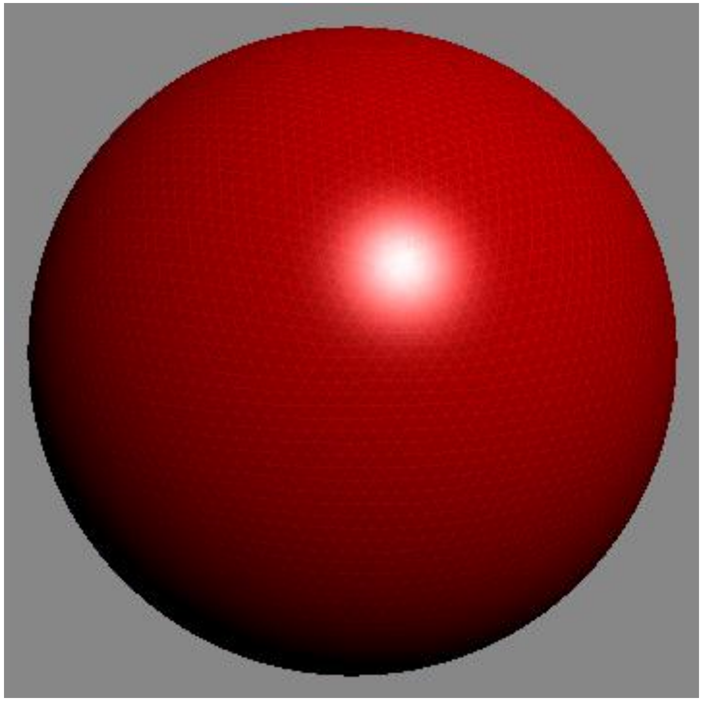


Gouraud interpolative shading without the specular term

4. Gouraud interpolative shading

- Since the intensity is only calculated at the vertices, the method cannot adequately deal with specular highlights
  - (unless the geometric resolution is such that polygons project to an area of 2x2 pixels)



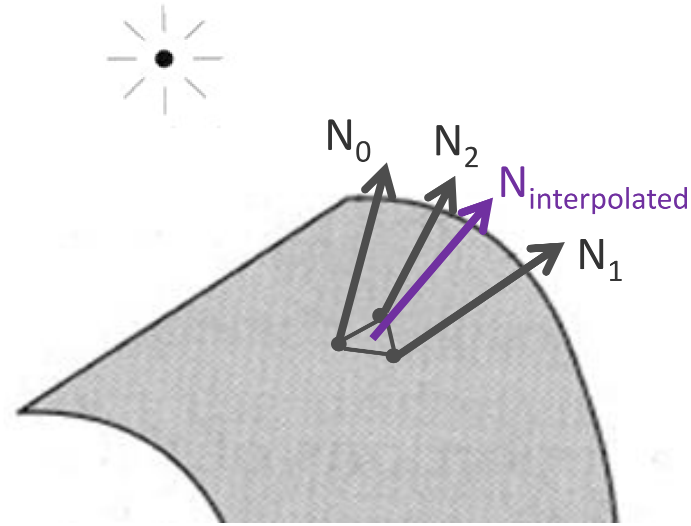


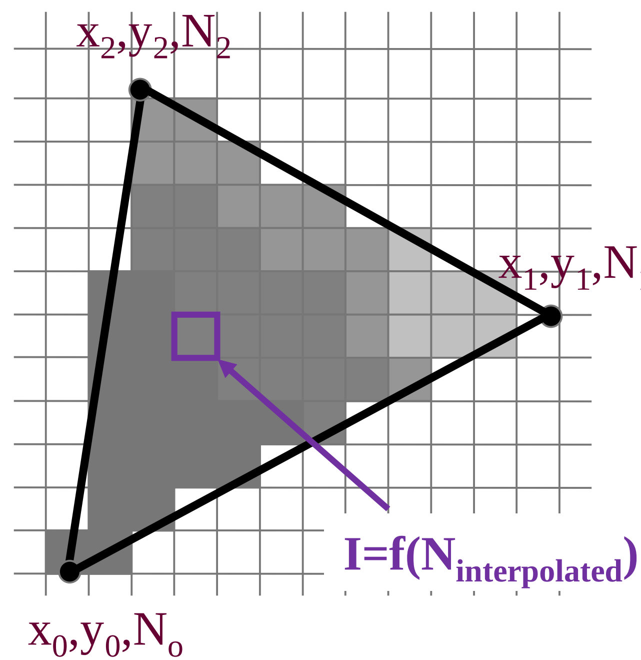
Polygon edges are drawn to emphasize the effect

Images from http://en.wikipedia.org/wiki/Gouraud_shading

5. Phong interpolative shading

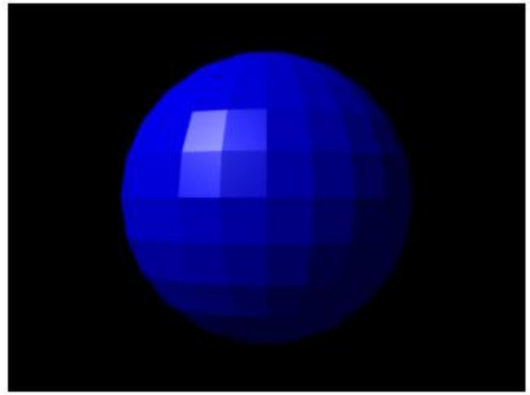
- Each vertex has a normal
- Each vertex is projected to screen space
- World space vertex normals are interpolated using screen space calculations
- Interpolated normal is used to calculate intensity
- Lighting calculations done in world space using the interpolated normal
    - 1975: Phong: ambient, diffuse *and* specular



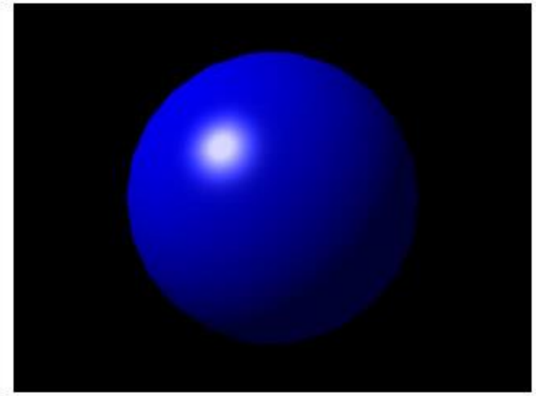


5. Phong interpolative shading

- Highlights are now correctly rendered



FLAT SHADING



PHONG SHADING

http://en.wikipedia.org/wiki/File:Phong-shading-sample.jpg

Fig. 1. A cone represented by means of planar approximation.

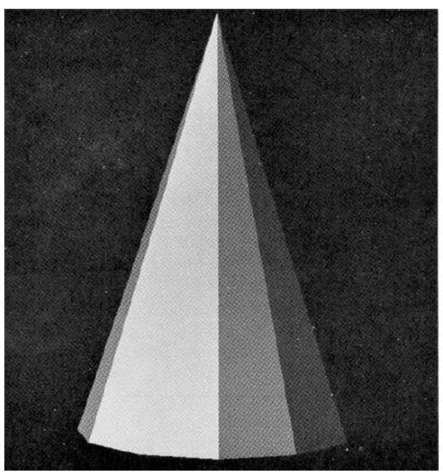


Fig. 4. Gouraud shading, applied to approximated cone of Fig. 1.

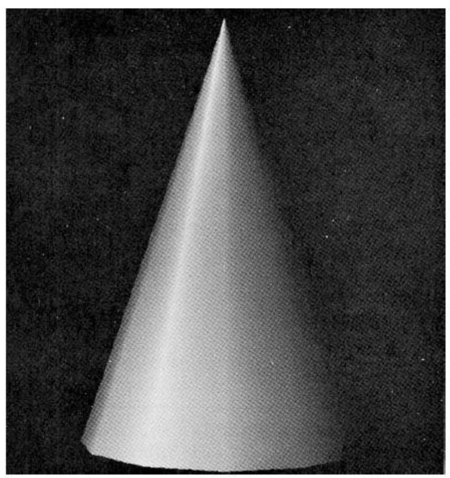
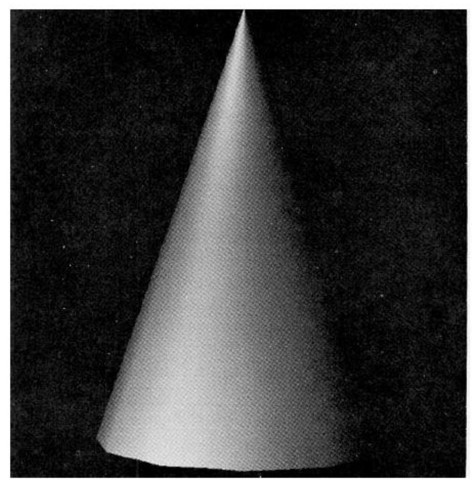
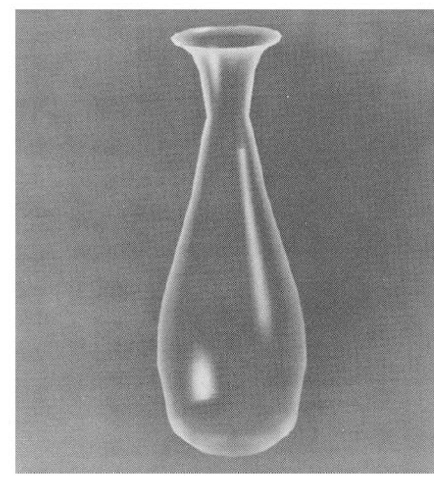


Fig. 8. Improved shading, applied to approximated cone of Fig. 1.



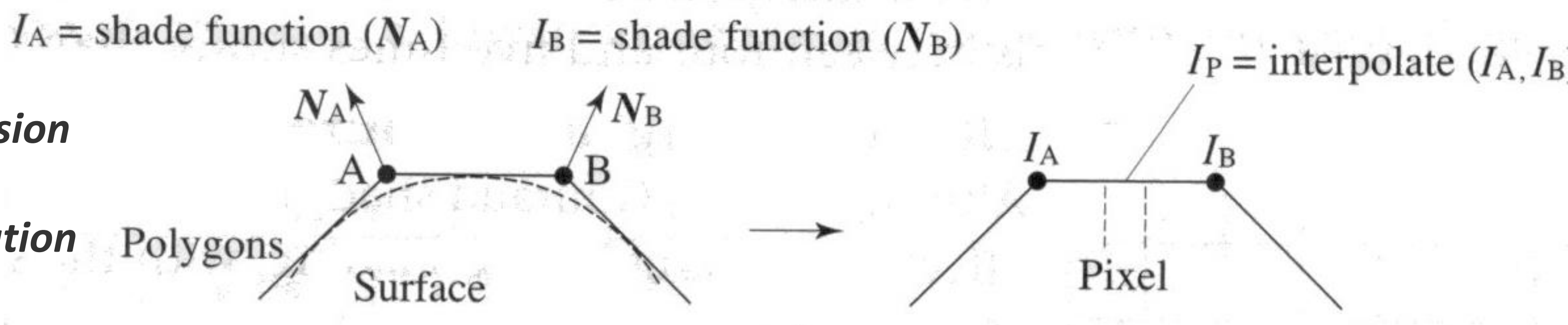


B. T. Phong, Illumination for computer generated pictures, Communications of ACM 18 (1975), no. 6, 311–317.

6. Gouraud versus Phong interpolative shading

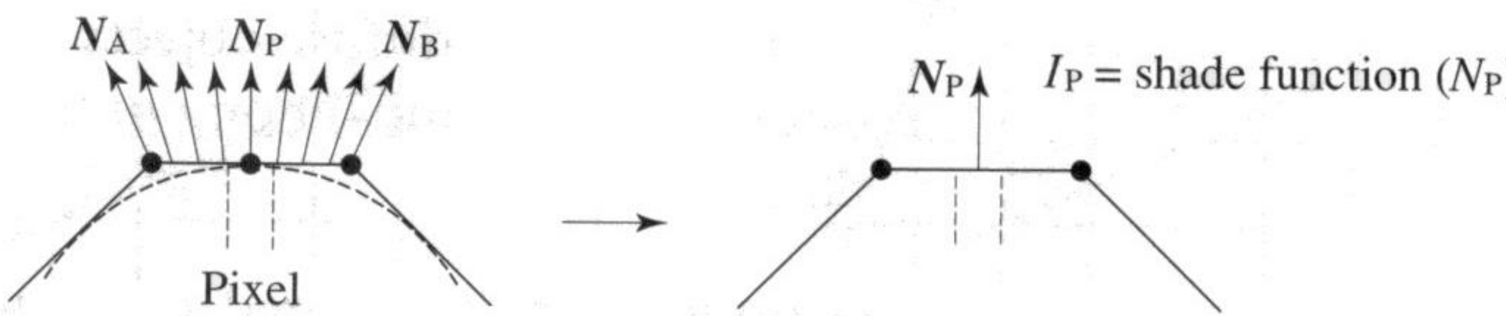
- Gouraud: Intensity calculated at each vertex using normal; Intensities interpolated across polygon

2D version for illustration



- Phong: Vertex normals are interpolated across the polygons; Intensity calculation is done for each pixel using interpolated normal

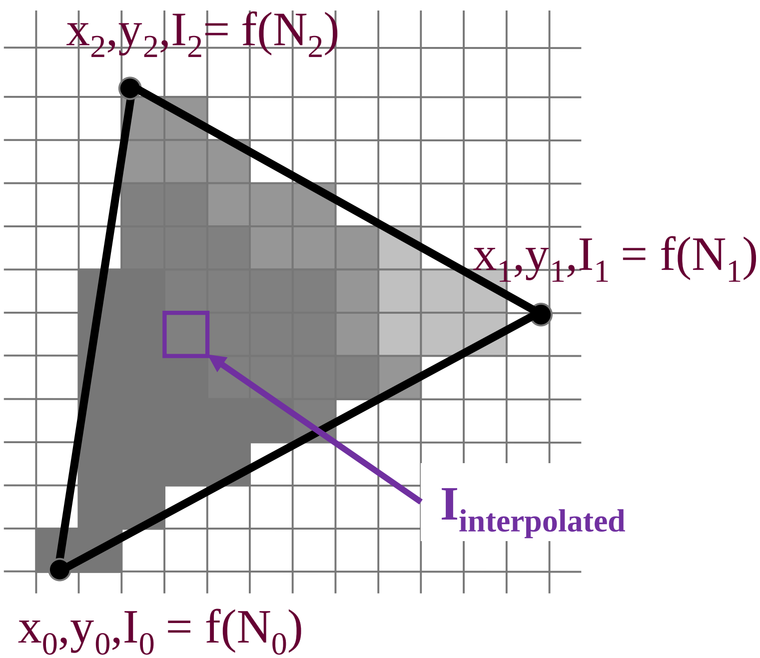
2D version for illustration

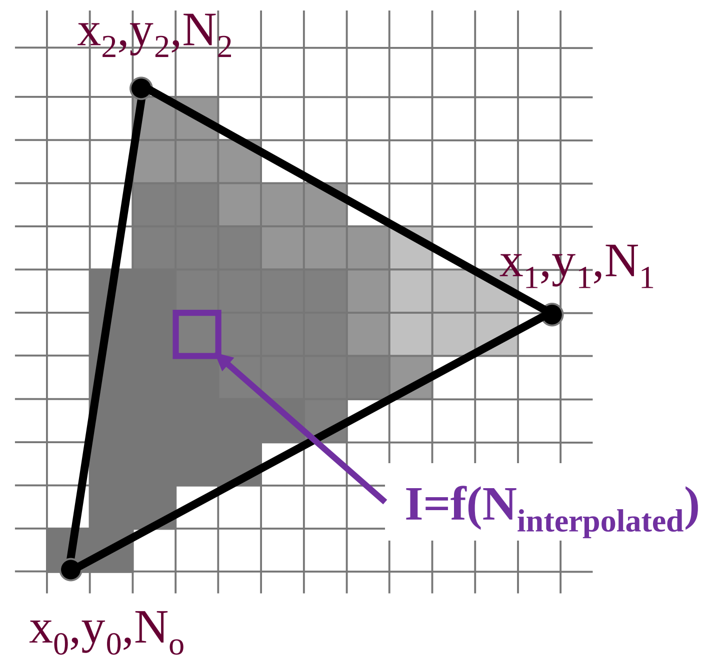


6. Gouraud versus Phong interpolative shading

Extra costs of Phong:

- Vector interpolation is three times as expensive as intensity interpolation
- At each step, the normal has to be normalized
- At each step (i.e. each pixel), lighting calculations are done

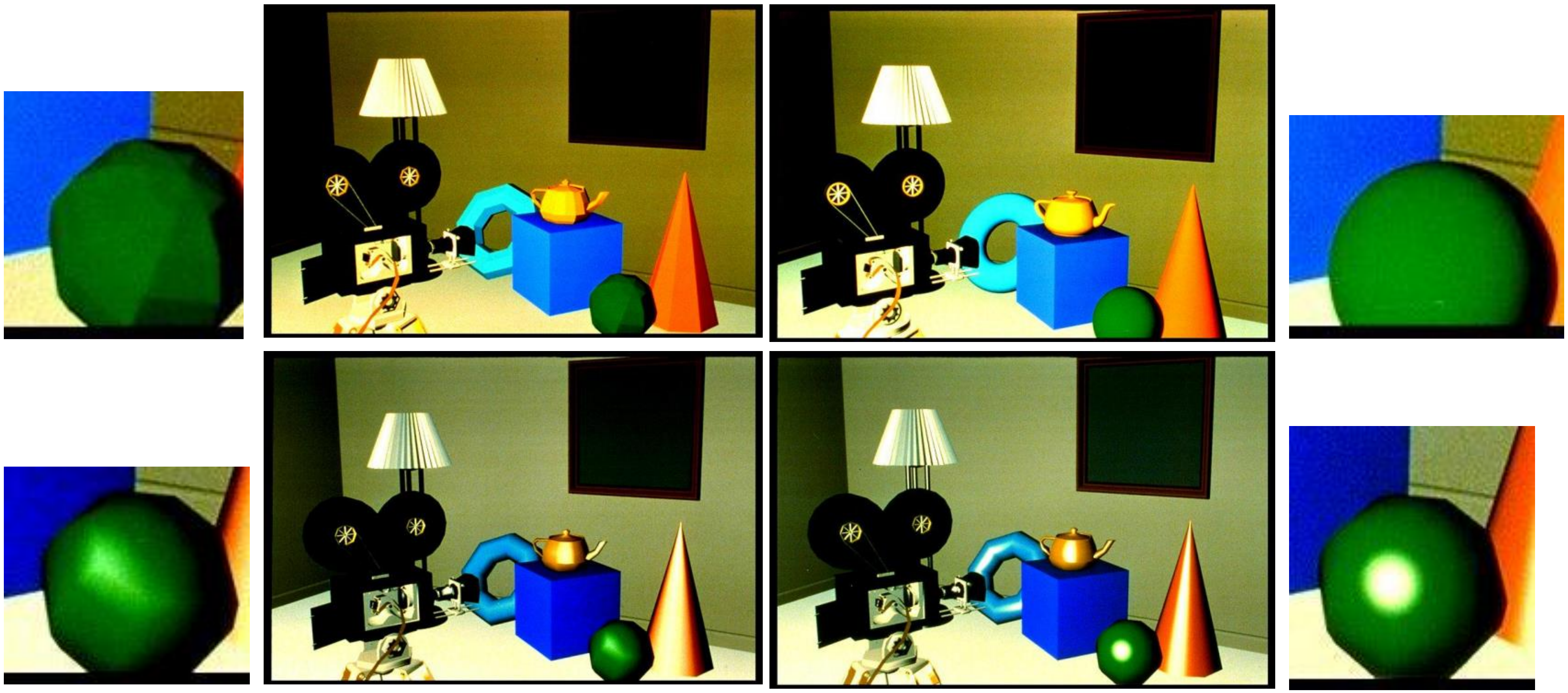




7. Flat, Gouraud, Phong

Flat or faceted shading

Gouraud interpolative shading. No specular term

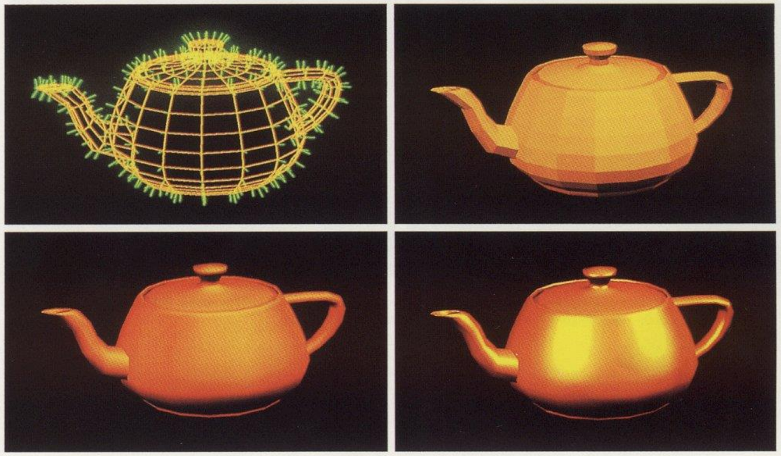


Gouraud interpolative shading with specular term

Phong interpolative shading

Pixar Shutterbug Image Series

8. Renderer shading options



2 The Utah teapot and shading options. Note the visibility of piecewise linearities along silhouette edges since this representation contains only 512 polygons. (*top left*) Wireframe plus vertex normals. (*top right*) Constant shaded polygons. (*bottom left*) Gouraud shading. (*bottom right*) Phong shading.

- Wireframe:
  - Quick – use for planning a scene or an animation
  - Edges may be drawn twice
- Flat shaded polygons:
  - Polygon normal is used, so that a single lighting calculation is used to shade the whole polygon
- Gouraud shading:
  - Only uses local reflection model at vertices and then interpolates intensities
- Phong shading:
  - Uses vector interpolation and then uses local reflection model at each pixel to calculate intensity
- Mixing Phong and Gouraud: Only use Phong for specular objects

9. Summary

- Local model – only approximates reality for single objects in free space
- Consider L and V at infinity, then lighting calculation is I = f(N)
  - Blinn-Phong model makes use of a H vector
- There are other local reflection models that incorporate other effects:
  - E.g. microfacet models (Cook & Torrance, 82), models for particular materials, subsurface scattering
- Interpolative shading:
  - Gouraud intensity interpolation, with a Phong reflection model at the polygon vertices, is the most common for a fixed function pipeline
  - Phong normal interpolation, with a Phong reflection model for each pixel, is the ‘standard’ quality shading method – this can be achieved using per-pixel shading on the GPU
- More info: http://en.wikipedia.org/wiki/Portal:Computer_graphics