CS 112 - Illumination and Shading

Illumination/Lighting

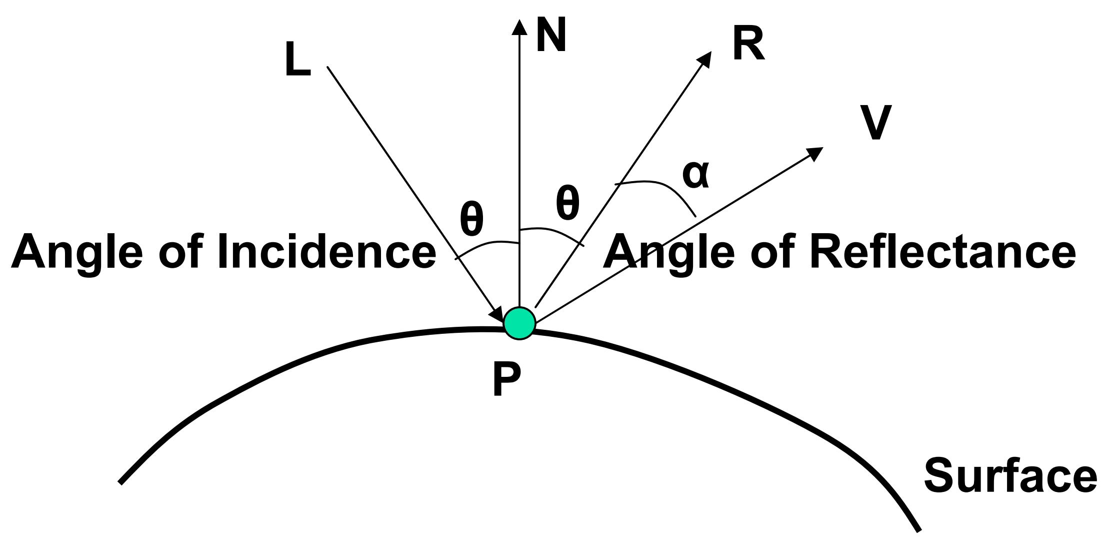
- Interaction between light and surfaces
  - Physics of optics and thermal radiation
  - Very complex: Light bounces off several surface before reaching the eye
- Approximations are required
  - Simple and at the same time believable
  - Graphics pipeline uses such illumination models

Illumination/Shading

- How does light interaction at any *point* of a surface to generate the color at that *point?*
  - Illumination
  - Evaluated only at triangle vertices
- From illumination generated at a *set of sample points,* how do we shade the whole *surface*
  - Shading
  - Shade planar triangles from the vertices

Lighting at a point on surface

- Do NOT think of triangulated surfaces
- All vectors are unit vectors
- Monochromatic light
- Object does not have color



Ambient Lighting

- Diffuse non-directional source of light
  - Sends equal amount of light in all direction
- Ambient light
  - Result of multiple reflections from multiple surfaces
  - Impinges equal light from all direction equally on all surfaces

Ambient Lighting

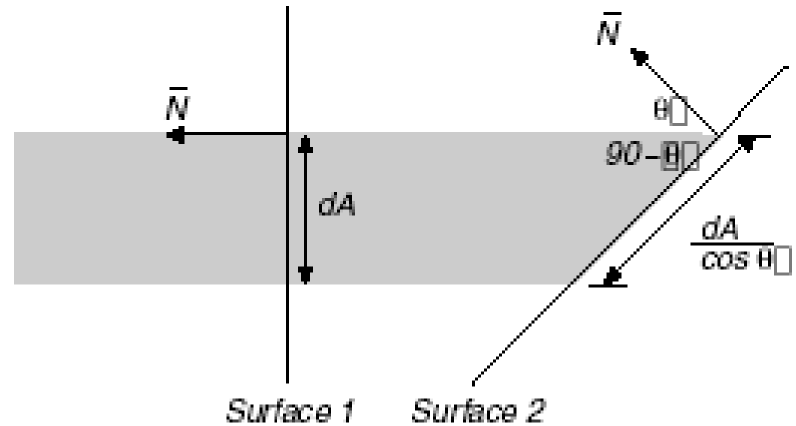
- $I = I_a k_a$
  - I_a = intensity of ambient light
  - k_a = percentage of the light reflected by the object
    - Coefficient of ambient reflection

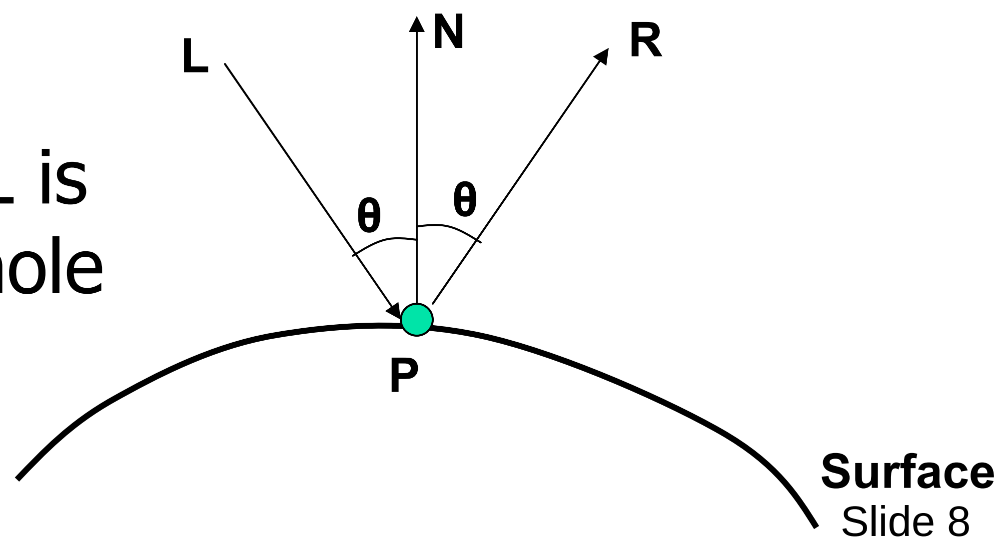
Diffused Lighting

- Point light sources: Illumination varies with
  - *Distance* of the light from the surface
  - *Orientation* of the light with respect to the surface
- Equal amount of light reflected in all direction
  - Independent of viewer location

Lighting at a point on surface

- $R \propto \cos\theta$
- $I = I_p k_d \cos\theta$
  - I_p = intensity of light
  - k_d = coefficient of diffuse reflection
- $I = I_p k_d (N.L)$
- If light is at infinity, L is constant over the whole surface





Ambient and Diffused Lighting

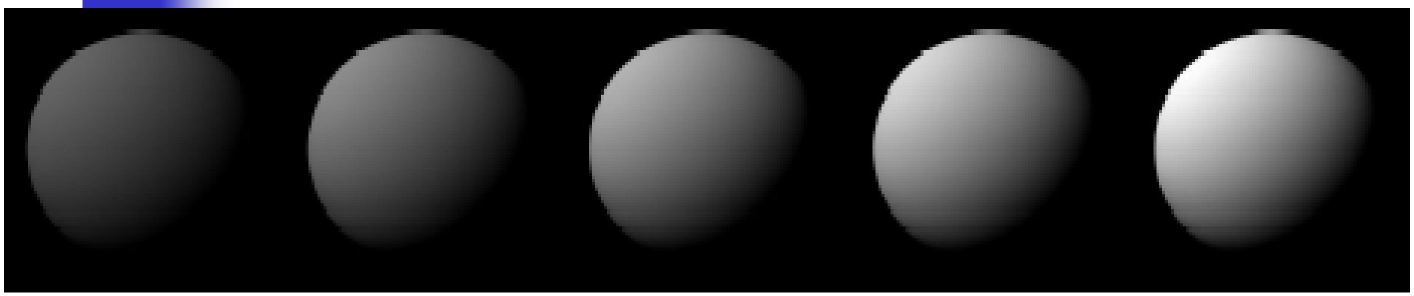


FIGURE 10. Diffuse reflection for k_d = 0.4, 0.55, 0.7, 0.85, 1.0.
(© [AW94] Figure 14.03)

$$I = I_p k_d (N.L)$$

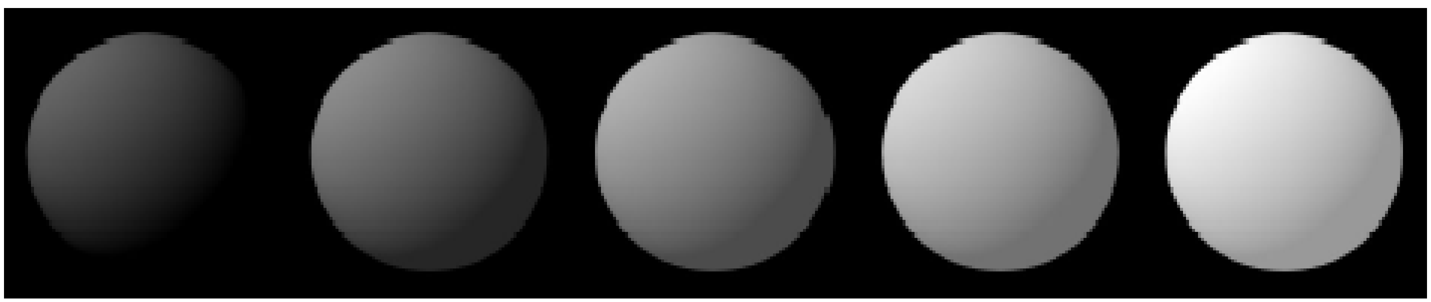


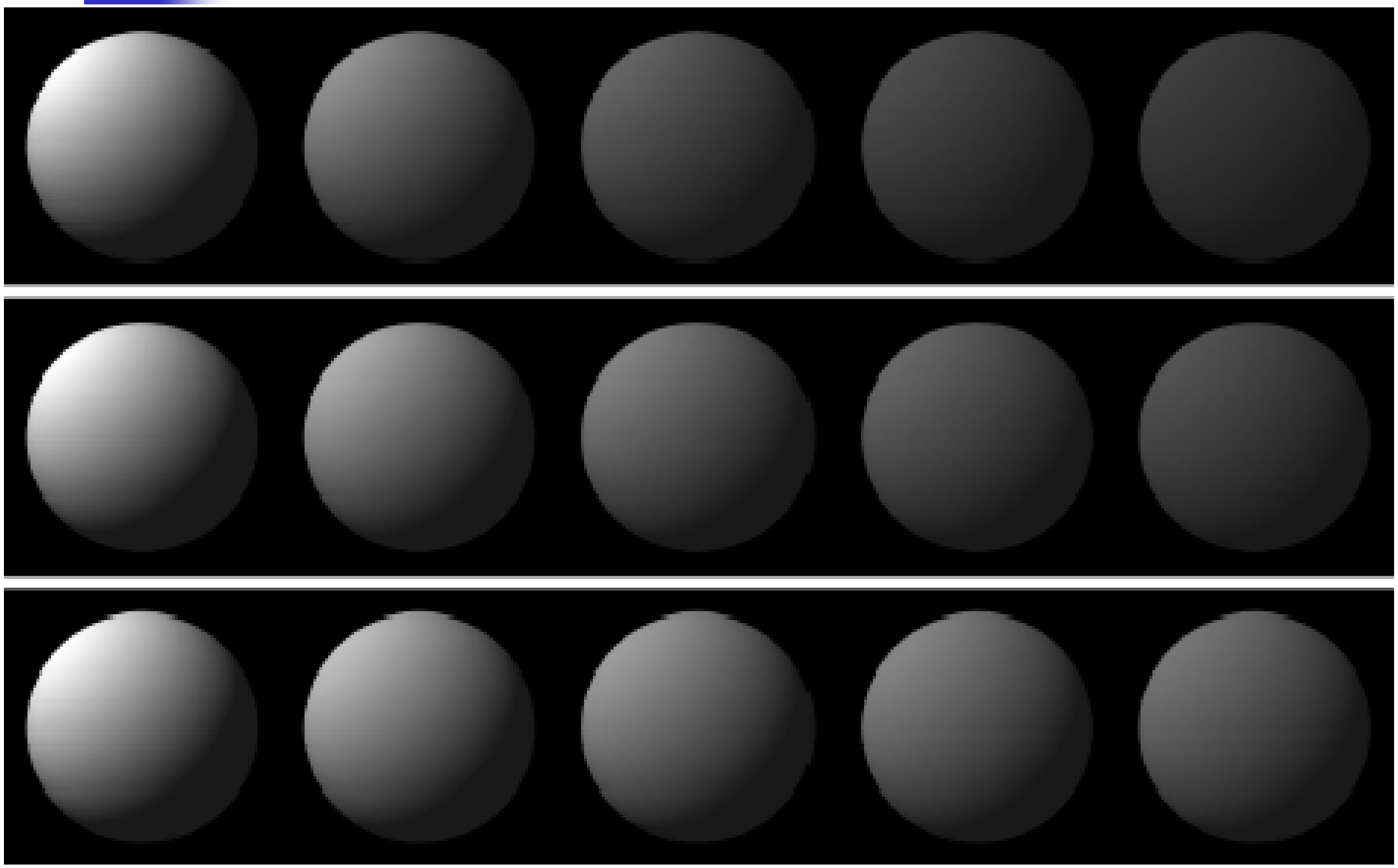
FIGURE 11. Ambient and diffuse reflection with k_d = 0.4 and k_a = 0.0, 0.15, 0.3, 0.45, 0.6.
(© [AW94] Figure 14.04)

$$I = (I_a k_a + I_p k_d (N.L))$$

Diffuse Lighting

- Did not take distance of the source from surface into account
- $I = I_p f_{att} k_d (N.L)$
  - $f_{att} = 1/(a+bd+cd^2)$
    - d = distance of light from the surface
    - a, b and c are user defined constants

Attenuation of Light



a=0, b=0, c=1

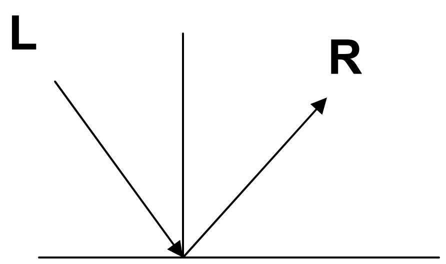
a=0.25, b=0.25, c=0.5

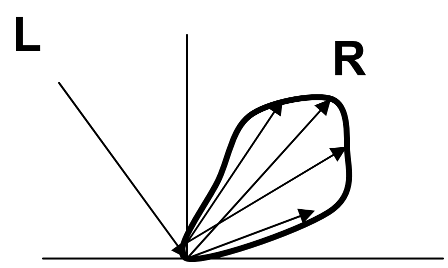
a=0, b=1, c=0

→ Increasing distance from the light source

Specular Lighting

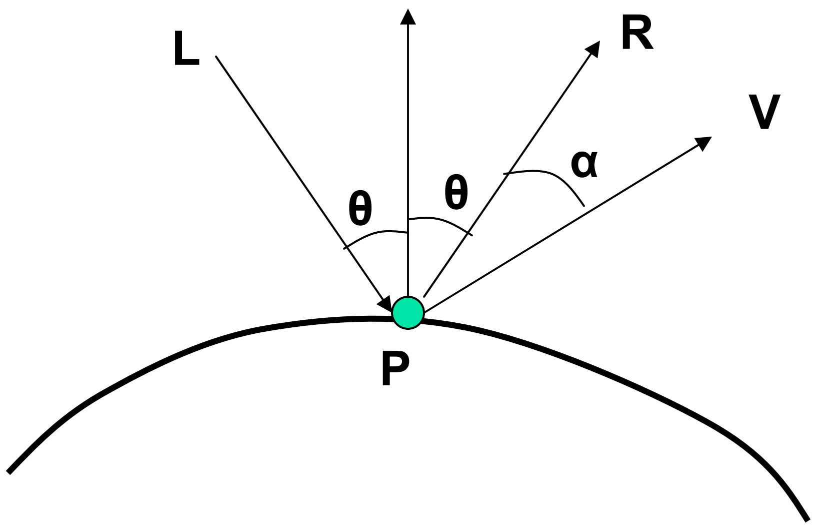
- Observed on shiny surfaces
- Amount of reflection changes with viewpoint
  - Think of a mirror, perfectly specular
- Phong Illumination Model

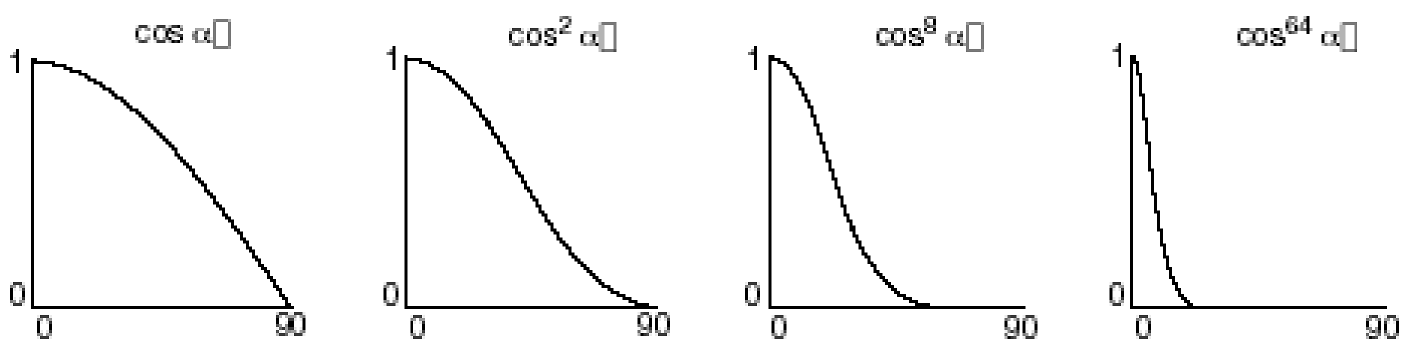




Phong Illumination Model

- $I_p k_s Cos^n(\alpha)$
- $Cos(\alpha)$: fall off as V moves away from R
- n gives the sharpness





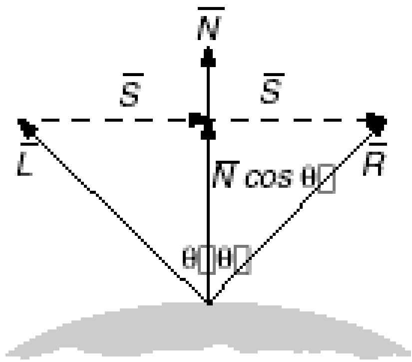
Phong Illumination Model

- $S = N\cos\theta - L$
- $R = N\cos\theta + S$

  $= 2N\cos\theta - L$

  $= 2N(N.L) - L$
- $\cos(\alpha) = R.V$

  $= (2N(N.L) - L).V$



Chromatic Light

- Ambient Light : (I_{aR}, I_{aG}, I_{aB})
- Point (I_{pR}, I_{pG}, I_{pB})
  - May have diffused and specular components
  - (I_{dR}, I_{dG}, I_{dB}) and (I_{sR}, I_{sG}, I_{sB})
- Object's color by a RGB value: (O_R, O_G, O_B)
  - Can have ambient, diffuse and specular components
  - (O_{aR}, O_{aG}, O_{aB}), (O_{dR}, O_{dG}, O_{dB}), (O_{sR}, O_{sG}, O_{sB})

Chromatic Light

- Each channel treated independent
  - Ambient : $I_{aC}k_aO_C$
  - Diffuse: $f_{att}I_{pC}k_dO_C(N.L)$
  - Specular: $I_{pC}k_s(R.V)O_C$
- Total for each channel
  - $O_C(I_{aC}k_a + f_{att}I_{pC}k_d(N.L) + I_{pC}k_s(R.V))$
- Different components
  - $O_{aC}I_{aC} + f_{att}O_{dC}I_{dC}(N.L) + O_{sC}I_{sC}(R.V)$

Multiple Light sources

- Only one ambient light source
- Multiple point light sources
  - Addition of light from different light sources

Ambient

Ambient + Diffuse

Ambient + Diffuse + Specular

What is Shading?

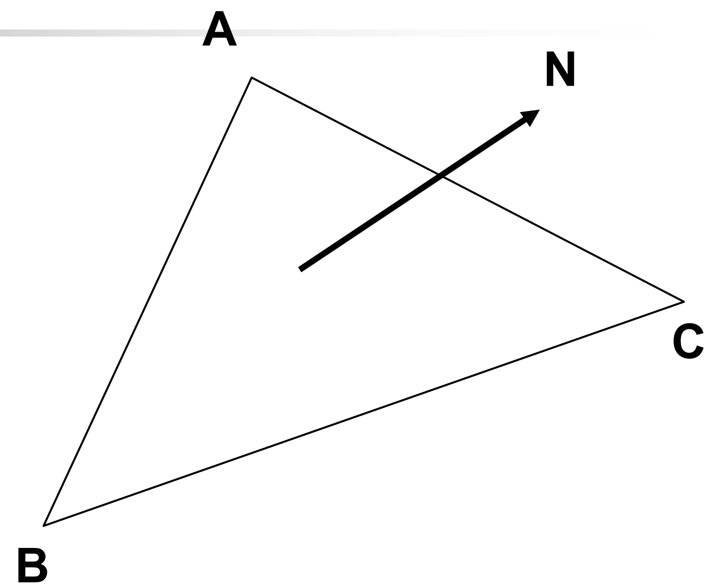
- Illumination model
- How do we use these models to ***shade*** the triangles in the graphics pipeline?
- How did we generate the picture on the right?

Method

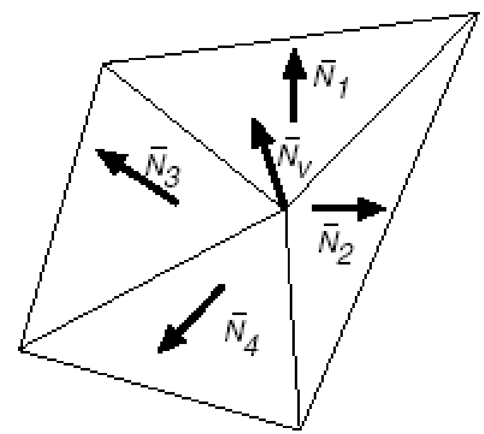
- Evaluate illumination model at the vertices of the triangles
  - After model-view transformation
- Use interpolation to color the interior of the triangles during rasterization
  - Different shading methods use different interpolation
- Assume that the polygonal models approximate smooth surfaces

Normal Computation

- Normal of a triangle
  - N = (B-A) x (C-A)
    - Vertices are in anticlockwise direction with respect to normal



- Normal of a vertex
  - Average of all the triangle incident on the vertex
  - $N_v = (N_1+N_2+N_3+N_4)/4$

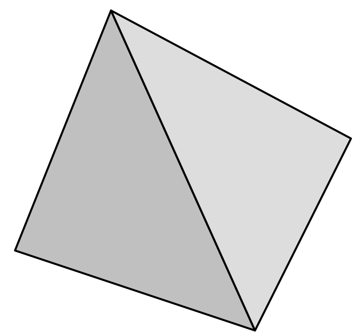


Constant/Flat/Faceted Shading

- Illumination model applied once per triangle
- Using normal of the triangle
- Shade the whole triangle uniformly
  - Color associated with triangles and not vertices

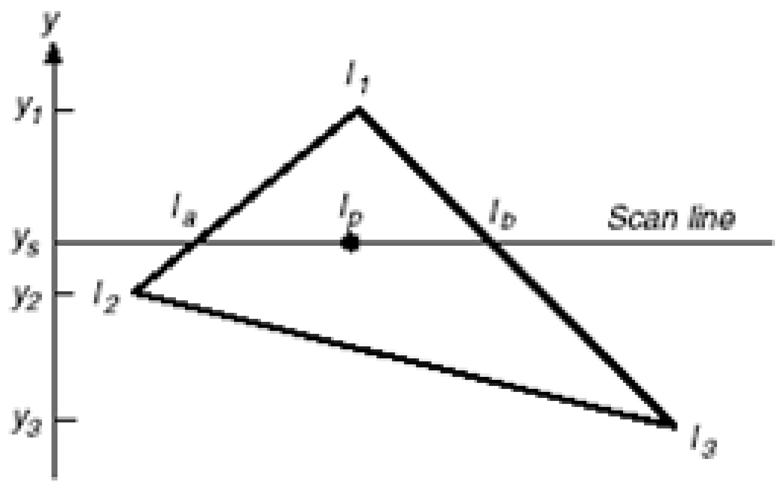
Validity

- Light is at infinity
  - N.L is constant over the plane of the triangle
- Viewer is at infinity
  - R.V is constant over the plane of the triangle
- Polygonal surface represents the actual surface being modeled
  - Not true
  - Shading is not continuous at edges



Gouraud Shading

- Interpolating illumination between vertices
  - Calculate the illumination using vertex normals at vertices
  - Bilinear interpolation across the triangle



$$I_a = I_1 - (I_1 - I_2)\frac{y_1 - y_s}{y_1 - y_2}$$

$$I_b = I_1 - (I_1 - I_3)\frac{y_1 - y_s}{y_1 - y_3}$$

$$I_p = I_b - (I_b - I_a)\frac{x_b - x_p}{x_b - x_a}$$

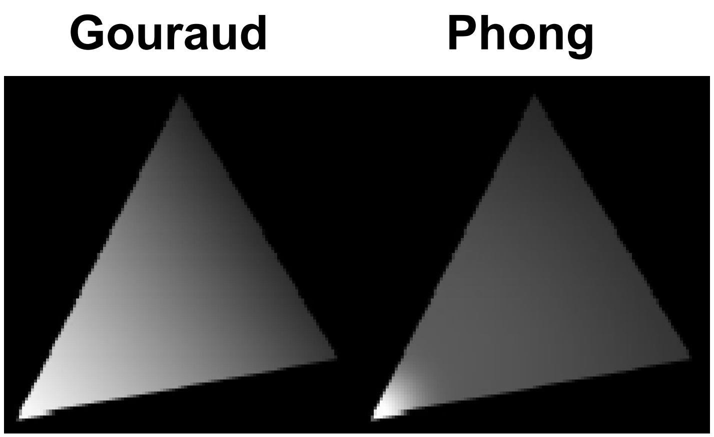
Gouraud Shading

- Edges get same color, irrespective of which triangle they are rendered from
  - Shading is continuous at edges
- Tends to spread sharp illumination spots over the triangle

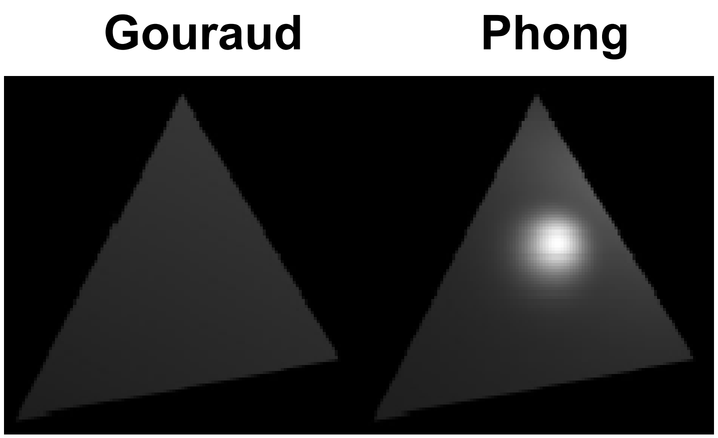
Phong Shading

- Interpolate the normal across the triangle
- Calculate the illumination at every pixel during rasterization
  - Using the interpolated normal
- Slower than Gouraud
- Does not miss specular highlights
  - Good for shiny specular objects

Gouraud vs. Phong Shading



Spreads highlights across the triangle



Misses a highlight completely

Flat Shading

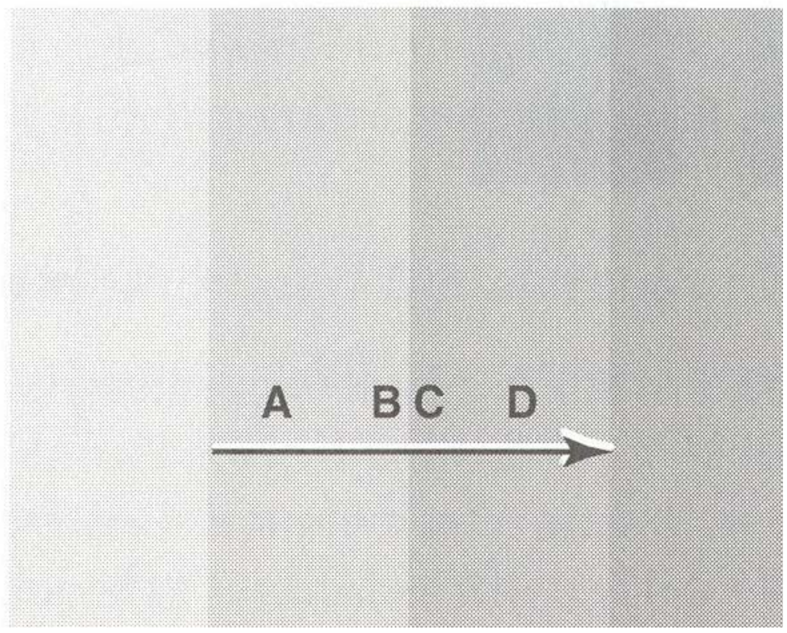
Gouraud Shading

Phong Shading

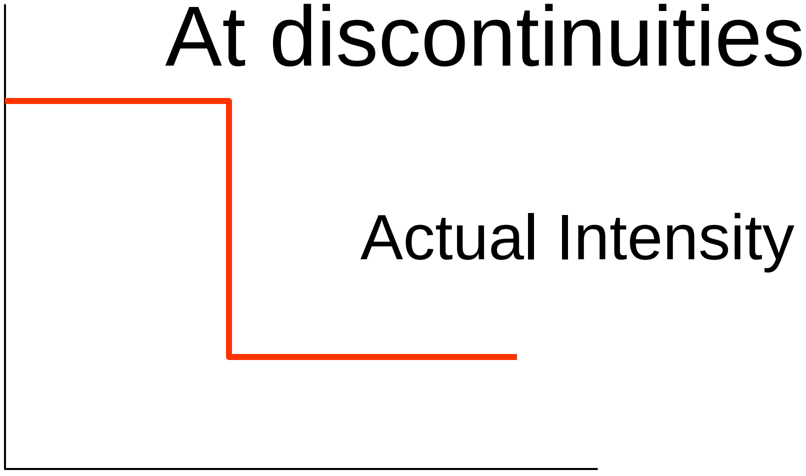
Shading

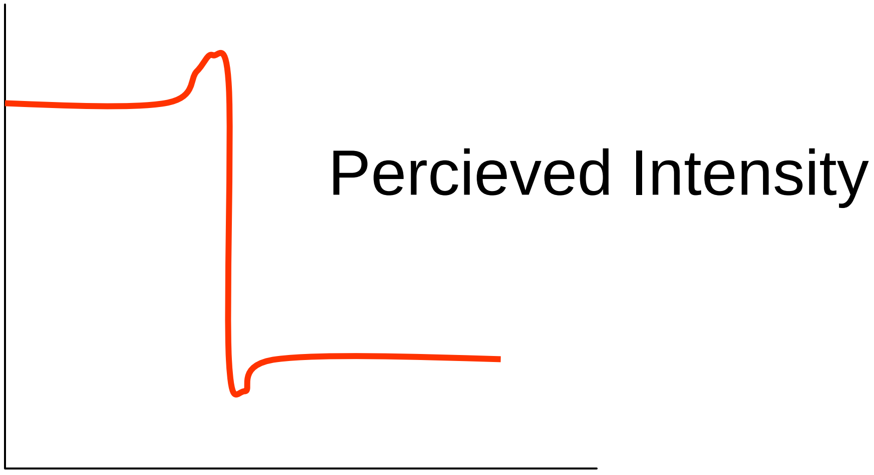
- Independent of the Illumination model used
- Phong Shading and Phong Illumination
- Artifacts
  - Piecewise planar approximation
  - Screen Space Interpolation
- Simple and hence widely used

Artifacts: Mach Bands



At discontinuities

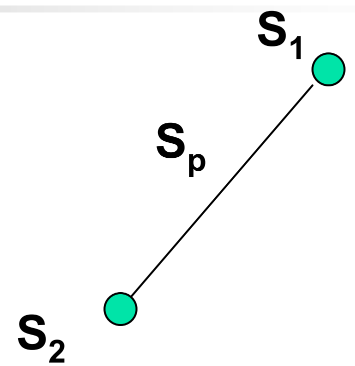




Artifacts: Mach Bands

- Common in flat shading since shading is discontinuous at edges
- Also present in Gouraud shading
  - Gradient of the shading may change suddenly
- Phong shading reduces it significantly
  - But cannot be eliminated
  - At sharp changes in surface gradient

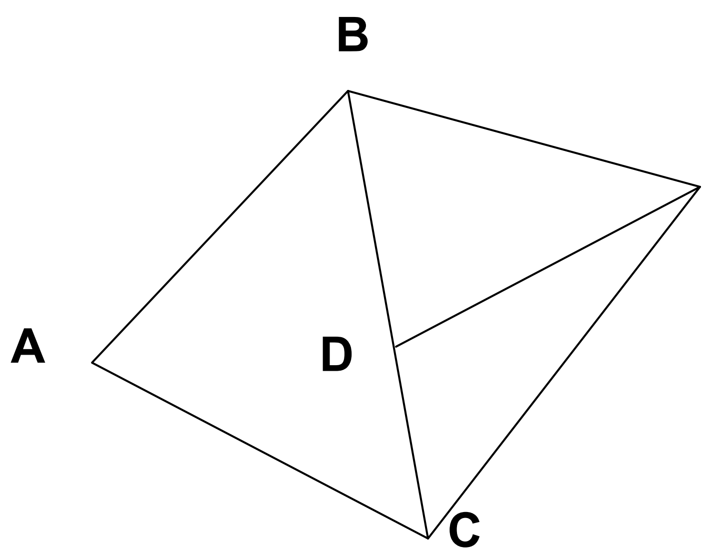
Artifacts: Screen Space Interpolation



- Shading is interpolated while rasterization
- $S_p = (S_1+S_2)/2$
  - $z_s \neq (z_1+z_2)/2$

Artifacts: T-junctions

- The shading at the T-junction are different when calculated from different triangles
- Shading discontinuity



Artifacts:Vertex Normals

- Vertex normal does not reflect the curvature of the surface adequately
  - Appear more flat than it actually is