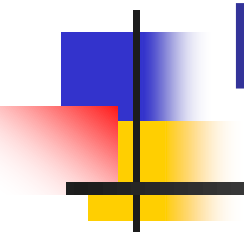
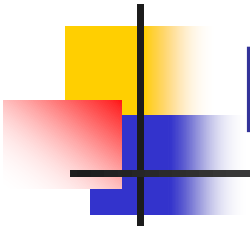
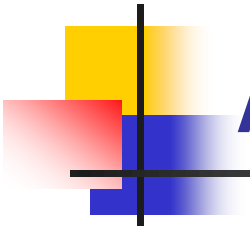


Polygon Shading



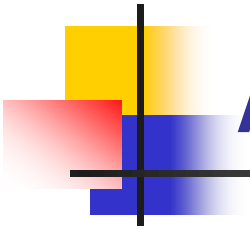
Lighting Review

- Lighting Models
  - Ambient
    - Normals don’t matter
  - Lambert/Diffuse
    - Angle between surface normal and light
  - Phong/Specular
    - Surface normal, light, and viewpoint



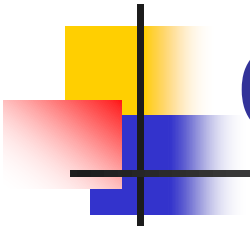
Applying Illumination

- We now have an illumination model for a point on a surface
- Assuming that our surface is defined as a mesh of polygonal facets, *which points should we use?*
- Keep in mind:
  - It's a fairly expensive calculation
  - Several possible answers, each with different implications for the visual quality of the result



Applying Illumination

- With polygonal/triangular models:
  - Each facet has a constant surface normal
  - If the light is directional, the diffuse reflectance is constant across the facet. Why?

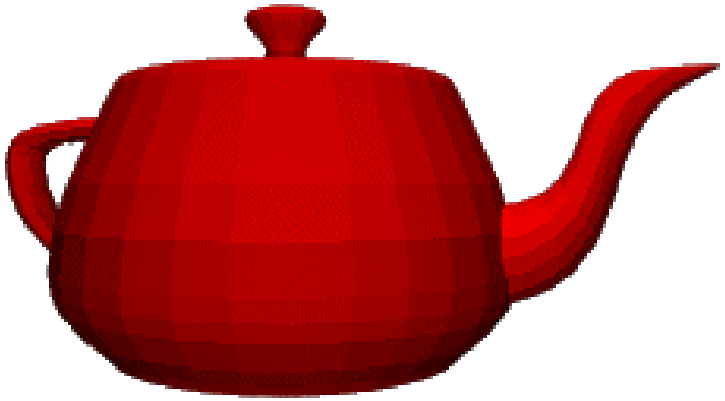


Constant-Intensity Shading

- Single intensity for each polygon
- Also known as Flat shading
- Useful for quickly displaying the general appearance of objects

Flat Shading

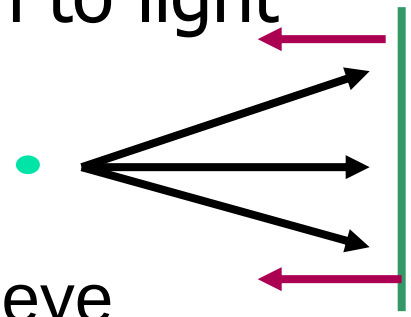
- The simplest approach, *flat shading*, calculates illumination at a single point for each polygon:



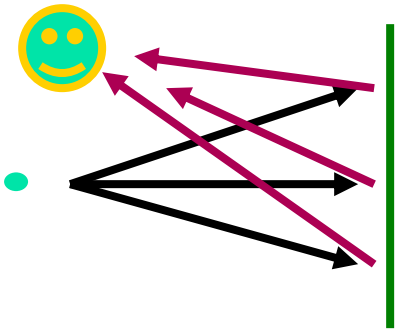
- If an object really is faceted, is this accurate?

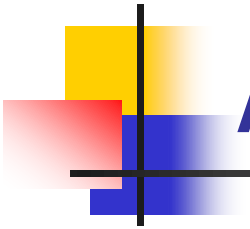
Is flat shading realistic for faceted object?

- No:
  - For point sources, the direction to light varies across the facet



– For specular reflectance, direction to eye varies across the facet



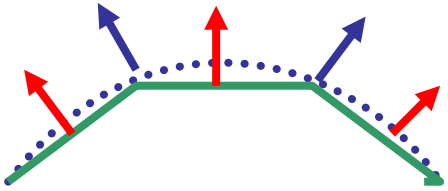


Assumption on Flat Shading

- The object is polygon-based, without a curved surface
- Light source is sufficiently far from the surface so that $N \cdot L$ is constant over the surface
- Viewing position is sufficiently far from the surface so that $V \cdot R$ is constant

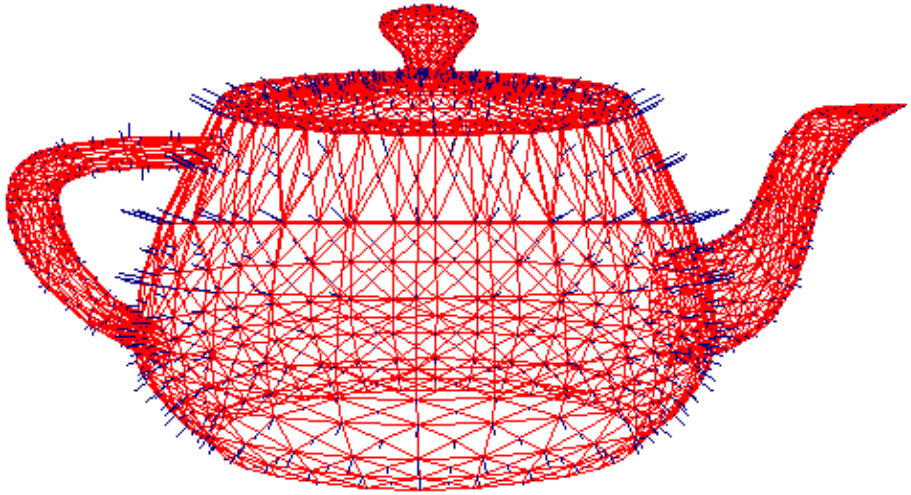
How to improve?

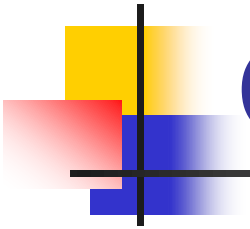
- We can refine it a bit by evaluating the Phong lighting model at each pixel of each polygon, but the result is still clearly faceted:
- To get smoother-looking surfaces we introduce *vertex normals* at each vertex
  - Usually different from facet normal
  - Used *only* for shading
  - Think of as a better approximation of the *real* surface that the polygons approximate



Vertex Normals

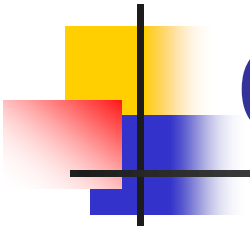
- Vertex normals may be
  - Provided with the model
  - Computed from first principles
  - Approximated by averaging the normals of the facets that share the vertex





Gouraud Shading

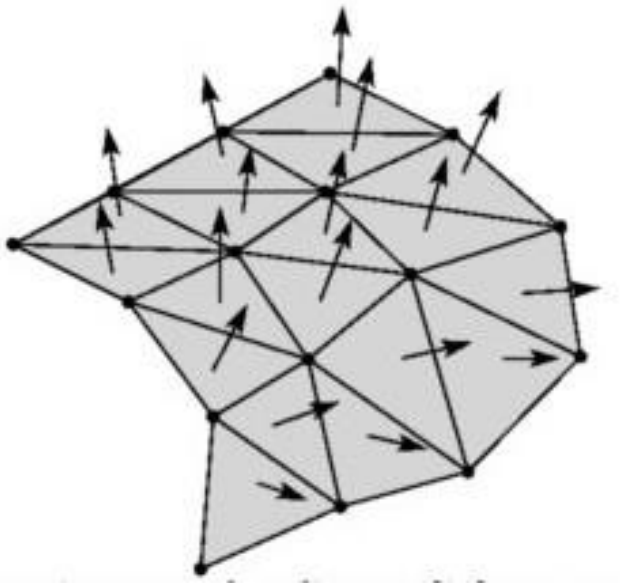
- Interpolation scheme
  - Finds out intensities for all internal pixels of a polygon surface by *linearly interpolating the intensity values* across the surface.
- Intensity values for each polygon are matched with the values of adjacent polygons



Overall Algorithm

- Determine the average unit normal vector at each polygon vertex
- Apply an illumination model to each vertex to calculate the vertex intensity
- Linearly interpolate the *vertex intensities* over the surface of the polygon

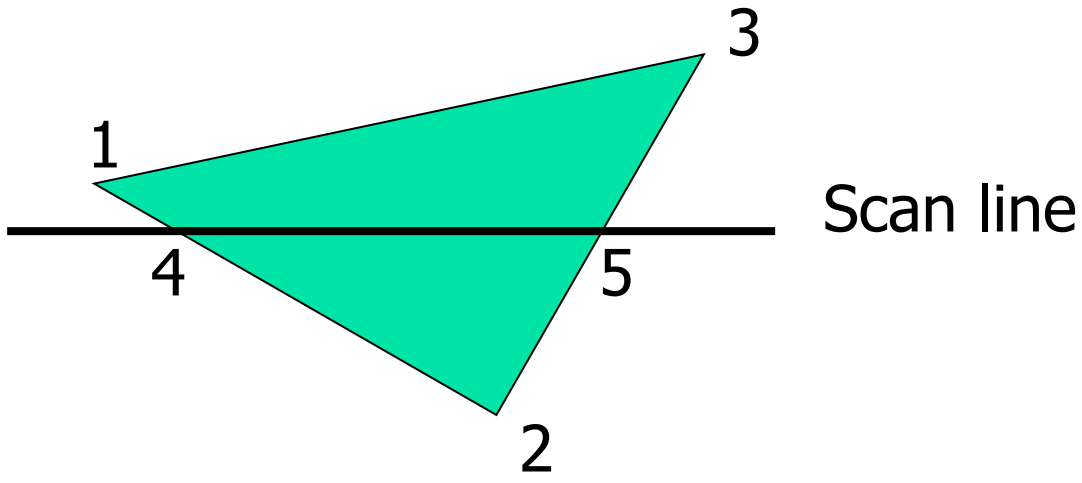
Normal Vectors



Normal vector at vertex is calculated as the average of the surface normals for each polygon sharing the vertex

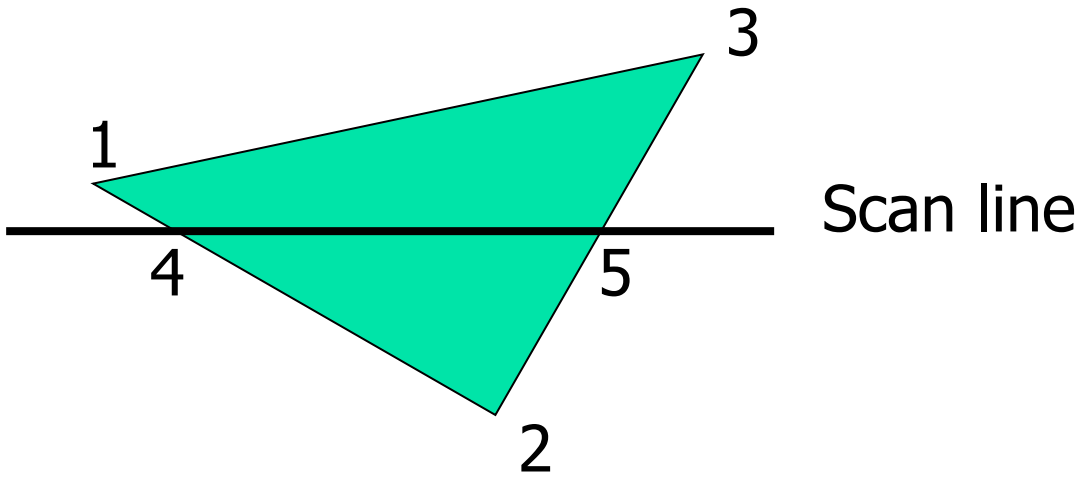
$$N_v = \frac{\sum_{k=1}^{n} N_k}{\left| \sum_{k=1}^{n} N_k \right|}$$

Interpolation



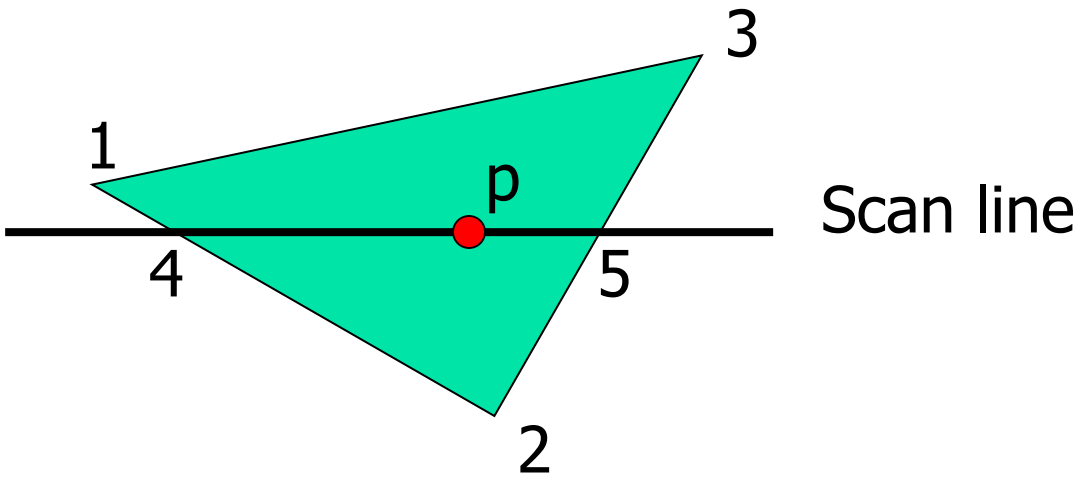
- Determine the intensity at the vertices
- Interpolating along the polygon edges

Interpolation



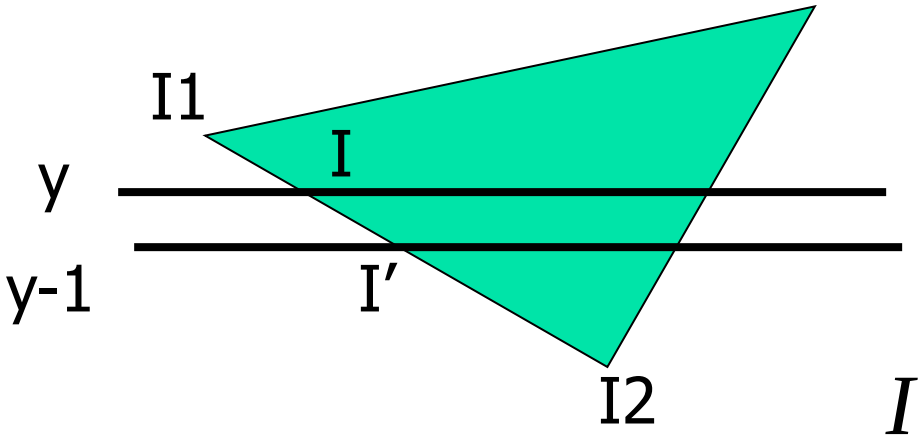
$$I_4 = \frac{y_4 - y_2}{y_1 - y_2} I_1 + \frac{y_1 - y_4}{y_1 - y_2} I_2$$

Interpolation for the intensity of internal pixels



$$I_p = \frac{x_5 - x_p}{x_5 - x_4} I_4 + \frac{x_p - x_4}{x_5 - x_4} I_5$$

Incremental Calculation

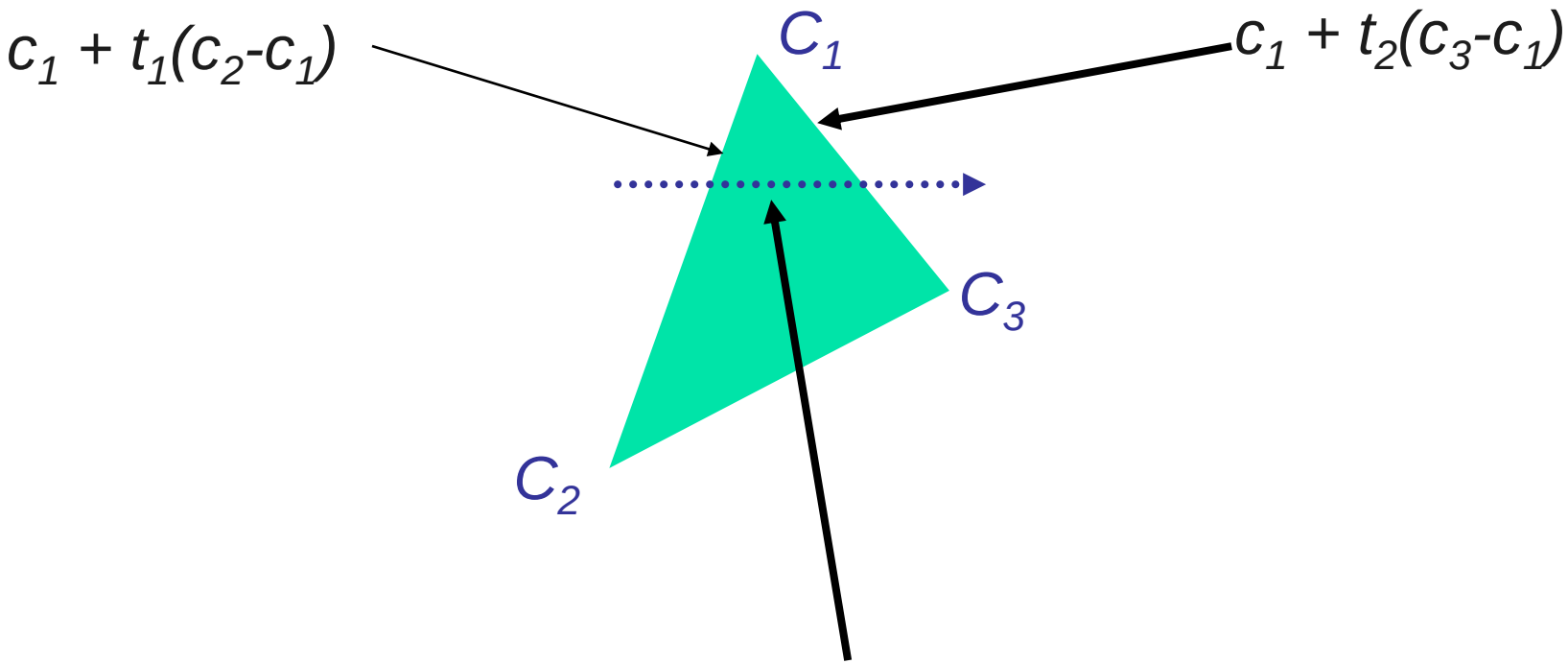


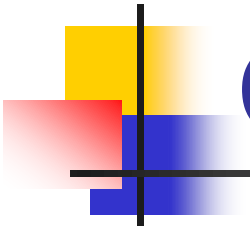
$$I = \frac{y - y_2}{y_1 - y_2} I_1 + \frac{y_1 - y}{y_1 - y_2} I_2$$

then

$$I' = I + \frac{I_2 - I_1}{y_1 - y_2}$$

Parametric Notation



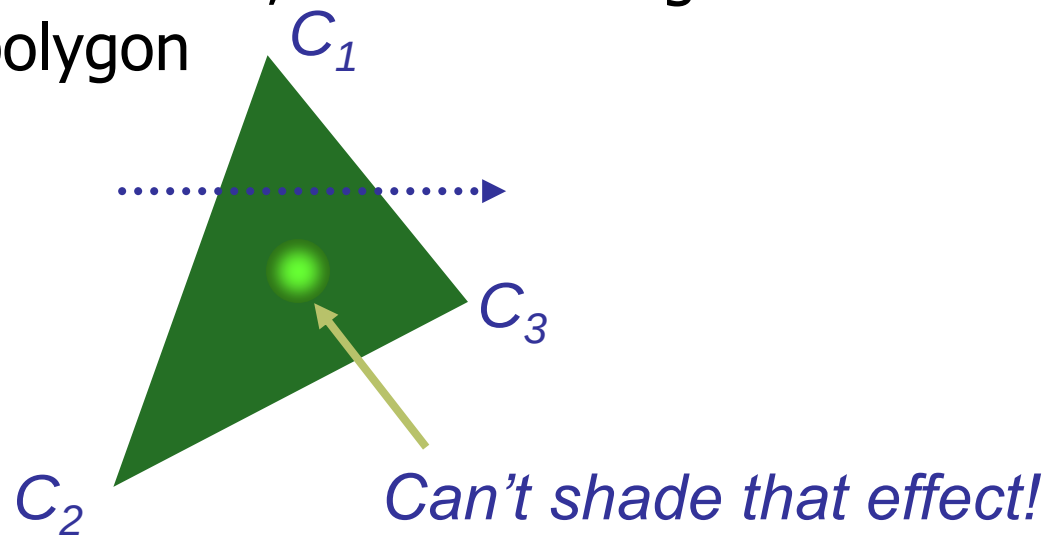


Gouraud Shading

- This is the most common approach
  - Perform Phong lighting at the vertices
  - Linearly interpolate the resulting colors over faces
    - Along edges
    - Along scanlines
- This is what OpenGL does for smooth shading
- *Does this eliminate the facets?*

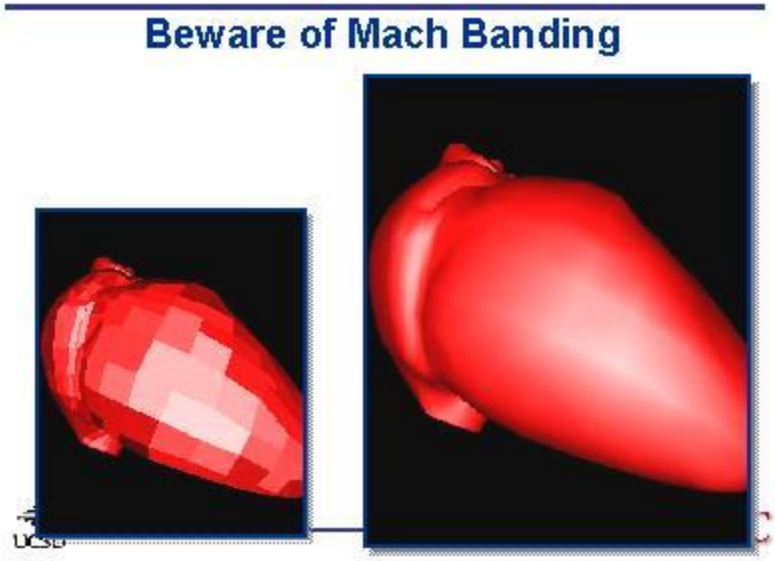
Gouraud Shading

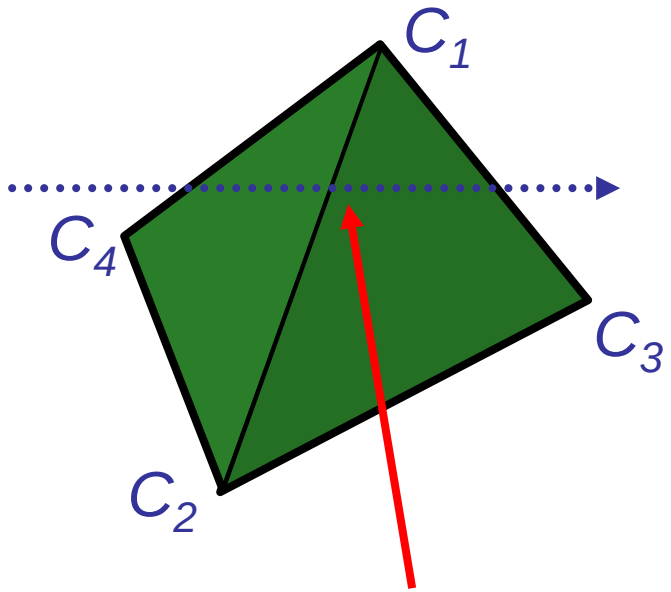
- Artifacts
  - Often appears dull, chalky
  - Lacks accurate specular component
    - If included, will be averaged over entire polygon



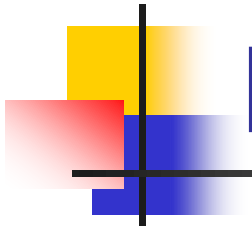
Gouraud Shading Artifacts

- Mach Banding
  - Artifact at discontinuities in intensity or intensity slope



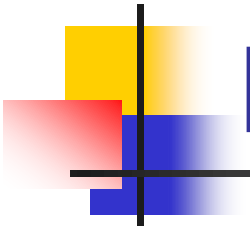


Discontinuity in rate of color change occurs here



Phong Shading

- *Phong shading* is not the same as Phong lighting, though they are sometimes mixed up
  - Phong lighting: the empirical model we've been discussing to calculate illumination at a point on a surface
  - Phong shading: linearly interpolating the surface normal across the facet, applying the Phong lighting model at every pixel
    - Same input as Gouraud shading
    - Usually very smooth-looking results:
    - But, considerably more expensive

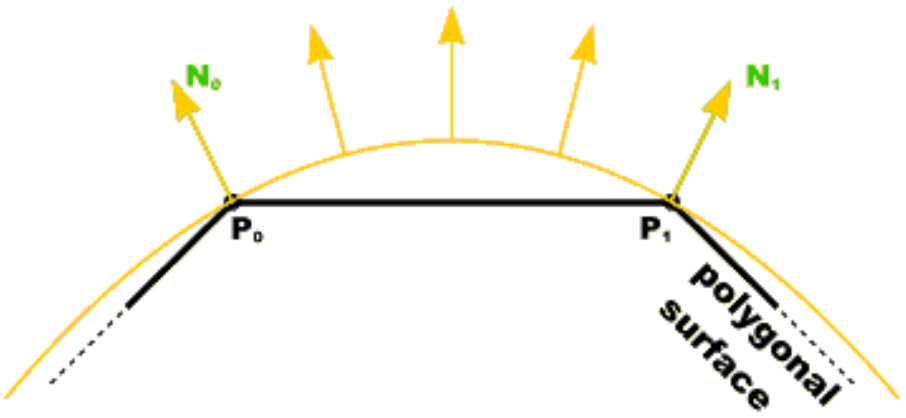


Phong Shading

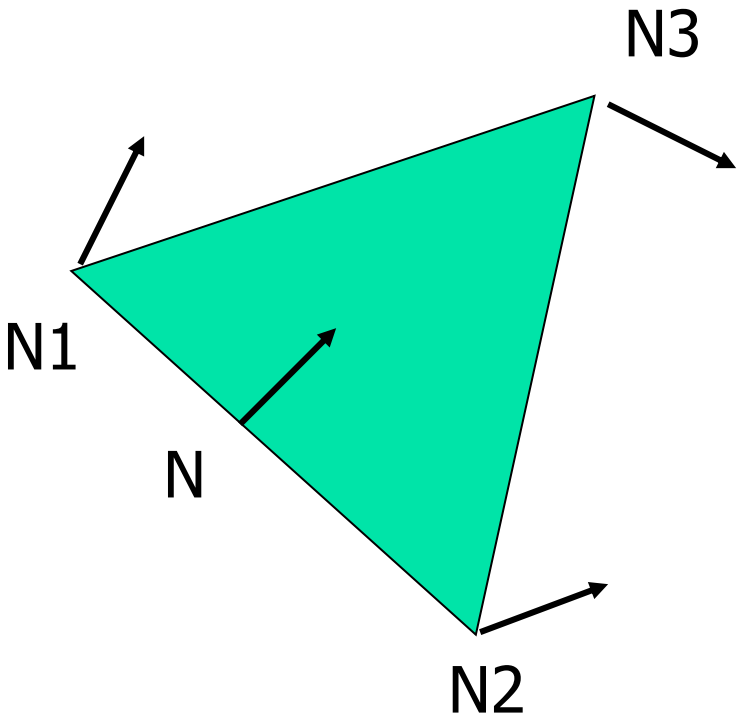
- Normal vector interpolation shading
- Apply the illumination model to each surface point
- Displays more realistic highlights on a surface and greatly reduces the Mach-band effect

Overall Algorithm

- Determine the average unit normal vector at each surface point
- Linearly interpolate the *vertex normals* over the surface of the polygon
- Apply illumination model along each scan line to calculate projected pixel intensities for the surface points



Interpolation of surface normal

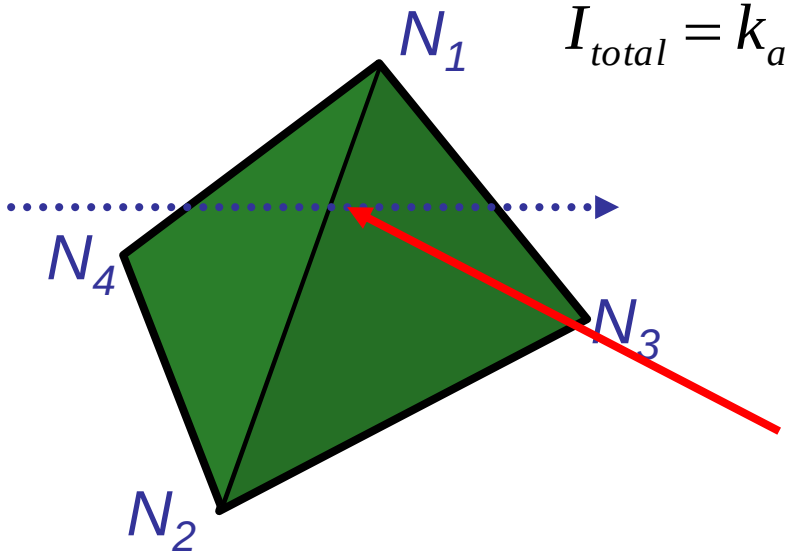


$$N = \frac{y - y_2}{y_1 - y_2} N_1 + \frac{y_1 - y}{y_1 - y_2} N_2$$

Phong Shading

- Linearly interpolate the vertex normals
  - Compute lighting equations at each pixel
  - Can use specular component

$$I_{total} = k_a I_{ambient} + \sum_{i=1}^{\#lights} I_i \left(k_d \left(\vec{N} \cdot \vec{L}_i \right) + k_s \left(\vec{V} \cdot \vec{R}_i \right)^{n_{shiny}} \right)$$

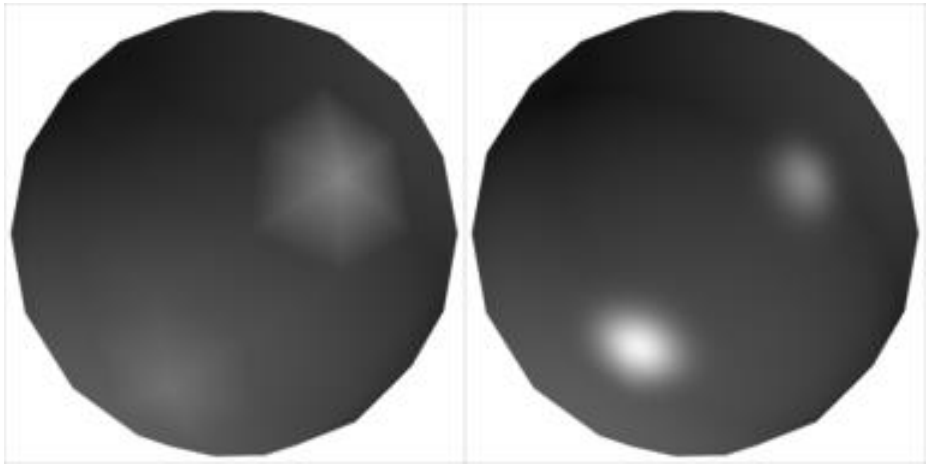


Remember: Normals used in diffuse and specular terms

Discontinuity in normal's rate of change is harder to detect

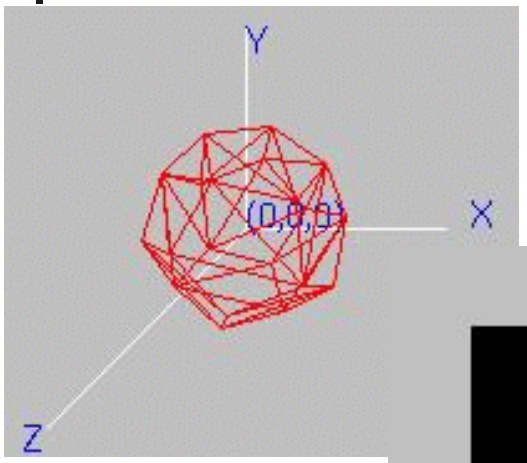
Shortcomings of Surface Shading

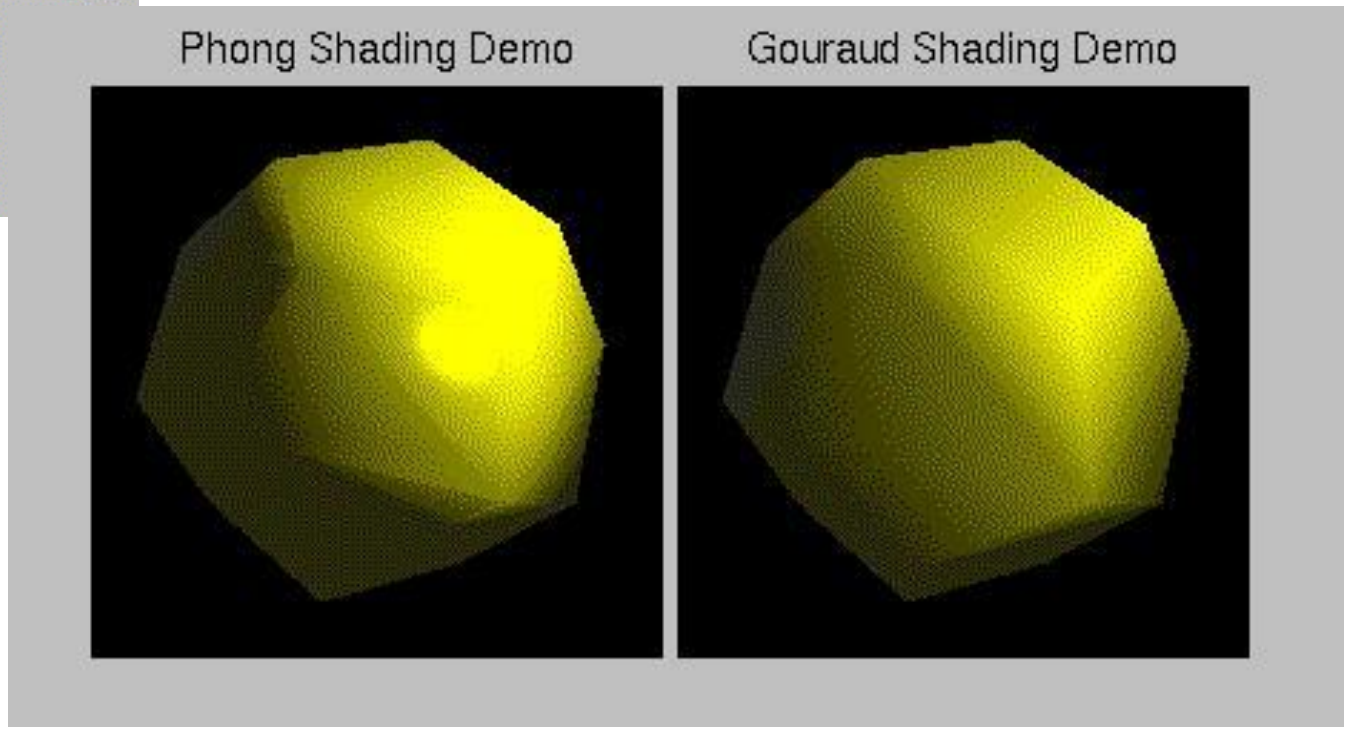
- Polygonal silhouettes remain



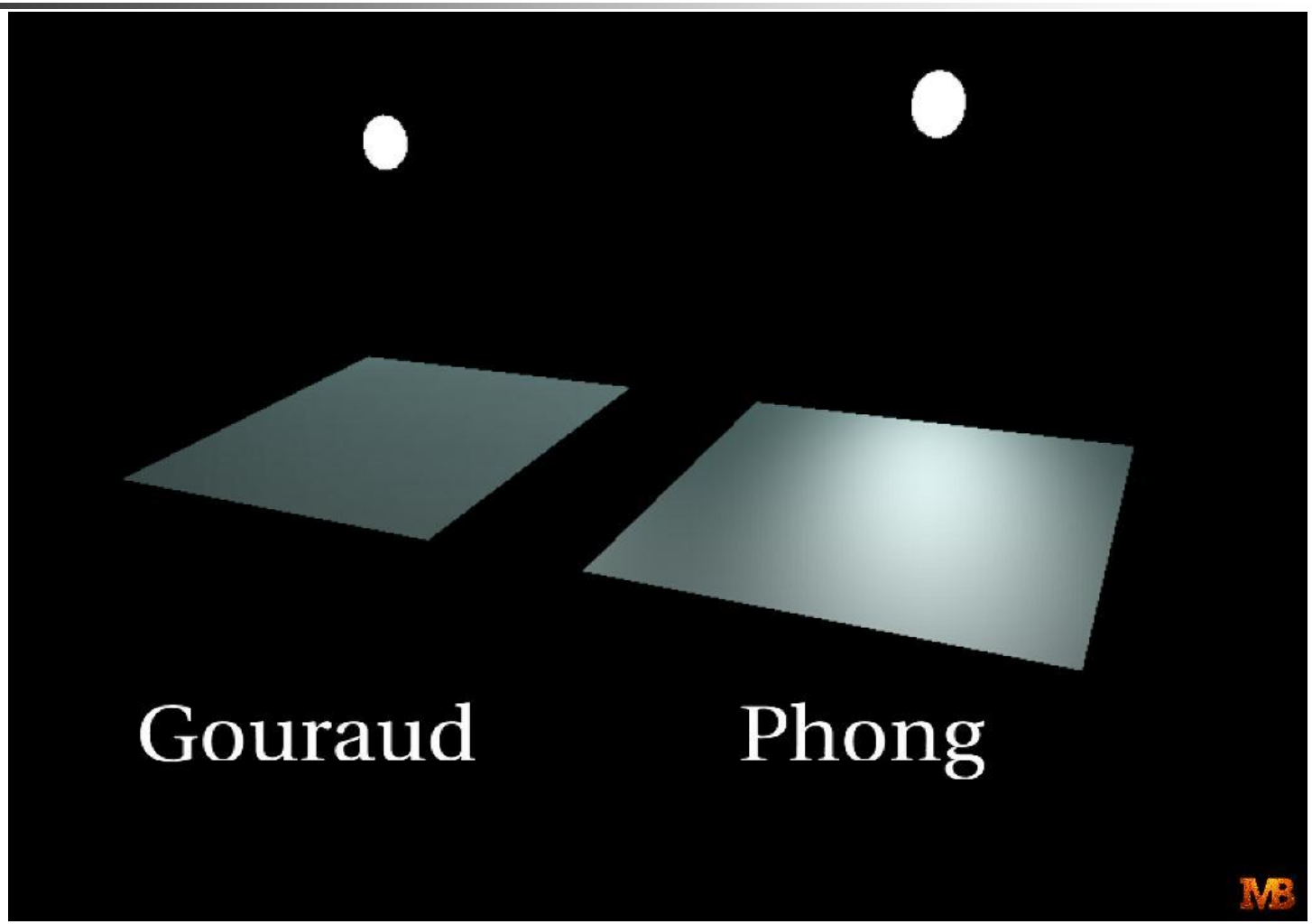
Gouraud *Phong*

Comparison





Comparison



per vertex lighting

per fragment lighting