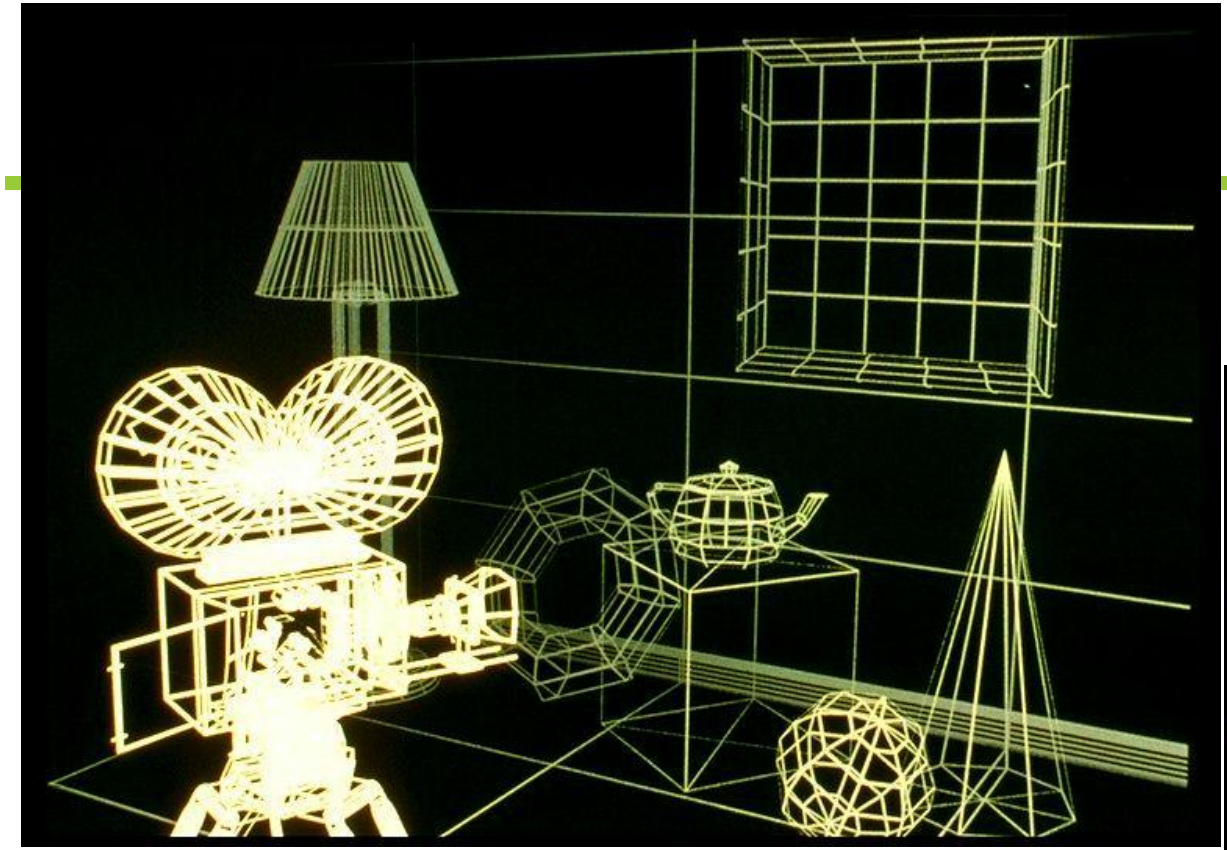
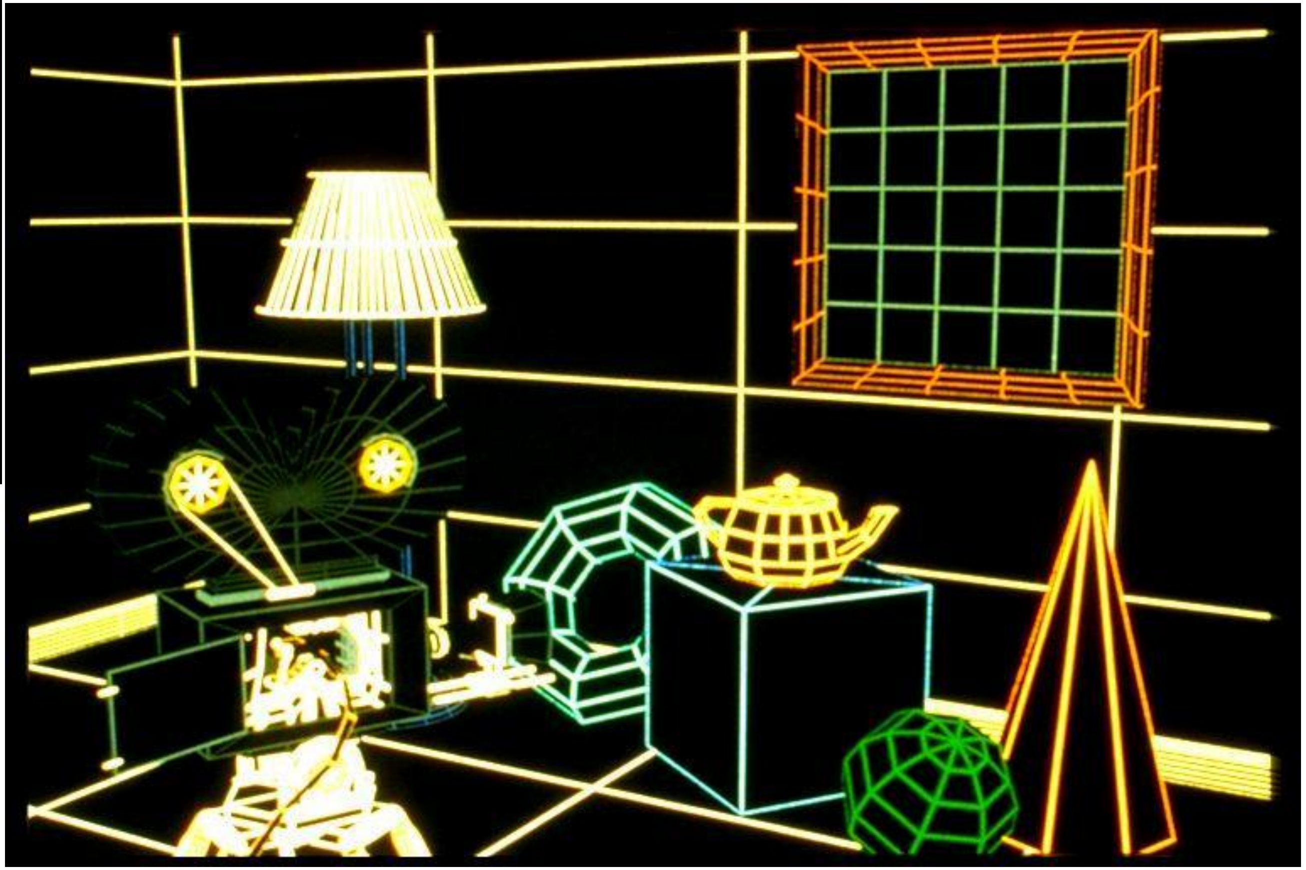
Shading

SURFACE RENDERING METHODS

Contents

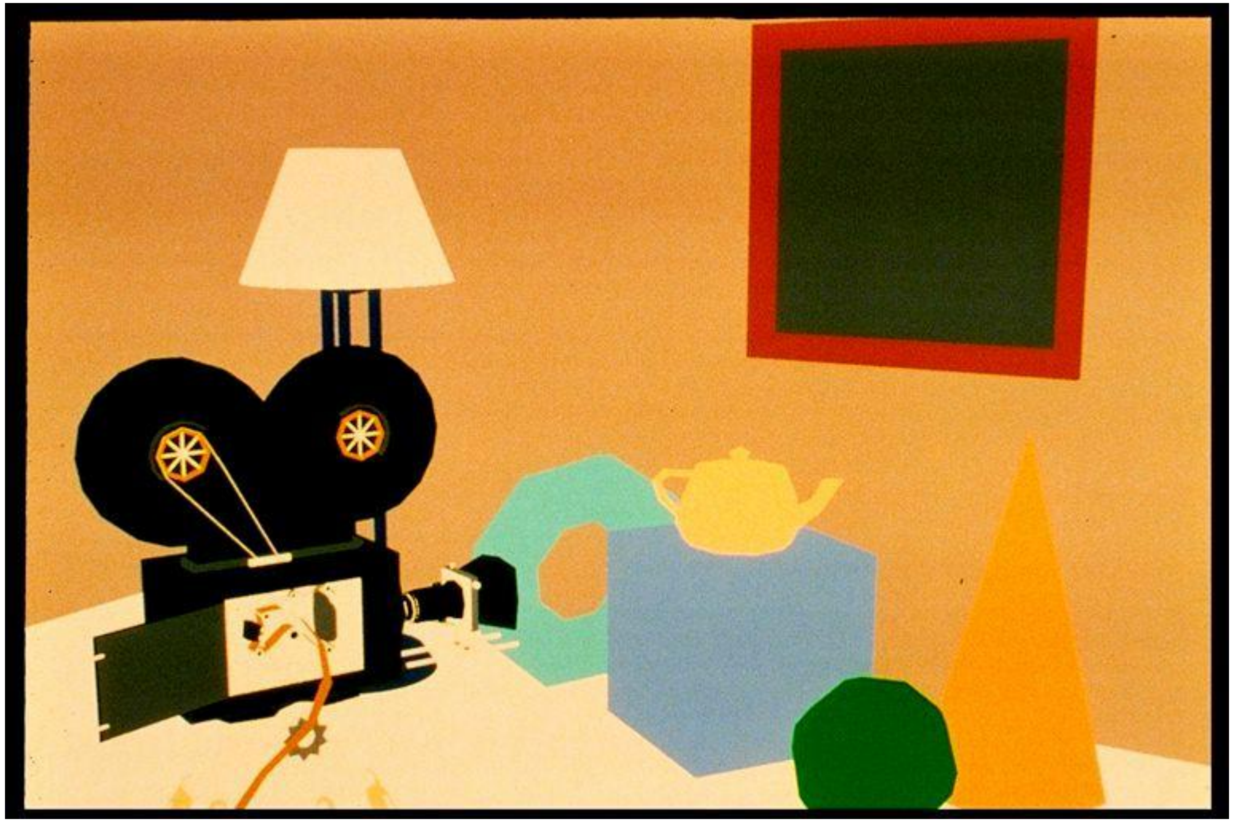
- Today we will start to look at rendering methods used in computer graphics
  - Flat surface rendering
  - Gouraud surface rendering
  - Phong surface rendering



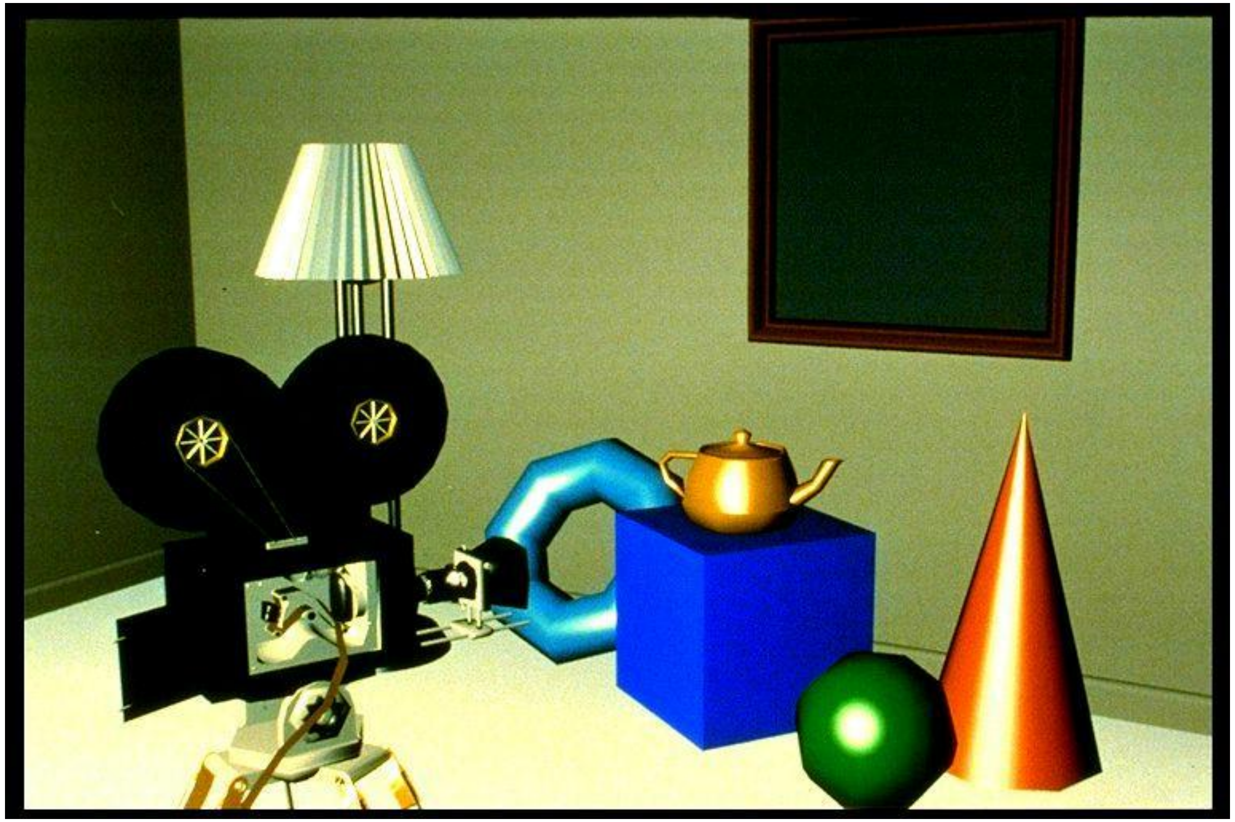


Wireframe model of a scene

No Surface Rendering Vs Surface Rendering



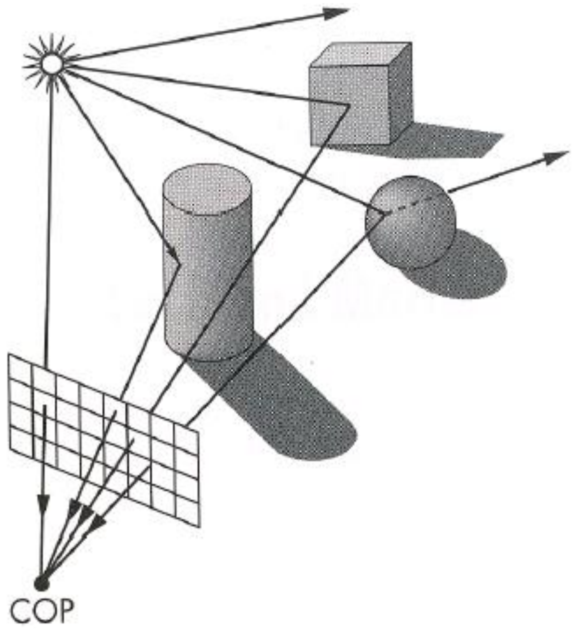
Object Rendering



With Surface Rendering

Surface Rendering: Shading

- Determine a Color for Each Filled Pixel
- How to Choose a Color for Each Filled Pixel
  - Each illumination calculation for a ray from the eyepoint through the view plane provides a radiance sample



Shading

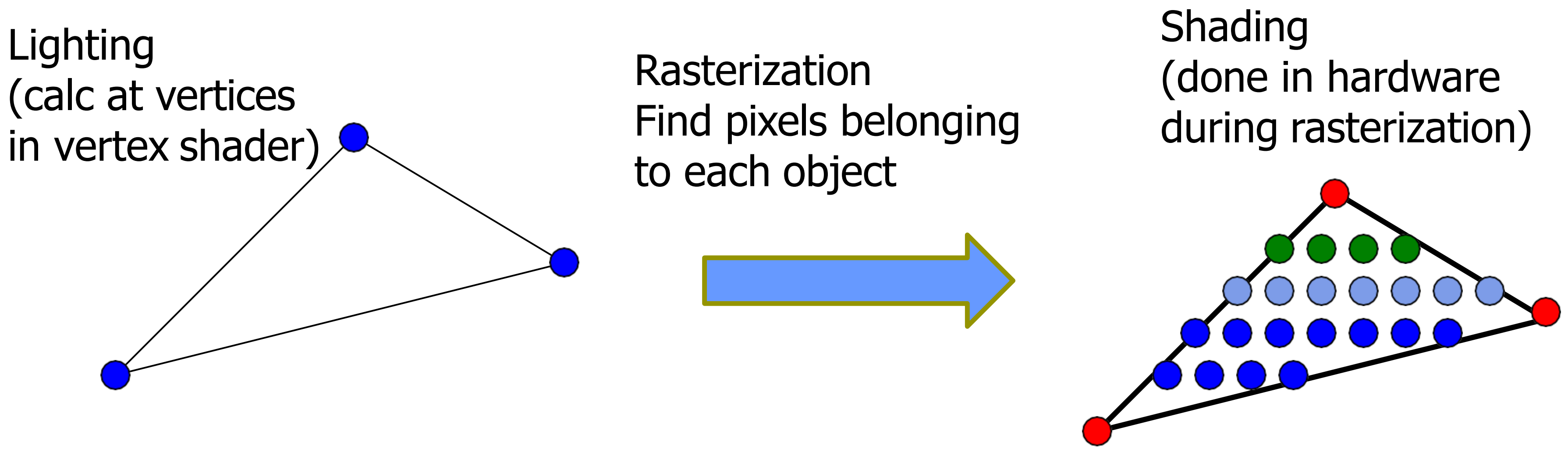
- Surface rendering means ***a procedure for applying a lighting model to obtain*** pixel intensities for all the projected surface positions in a scene.

- A surface-rendering algorithm uses the intensity calculations from an illumination model to determine the light intensity for all projected pixel positions for the various surfaces in a scene.

- Surface rendering can be performed by applying the illumination model to every visible surface point

Shading?

- After triangle is rasterized (converted to pixels)
  - Per-vertex lighting calculation means color at vertices is accurate, known (red dots)
- Shading: Graphics hardware figures out color of interior pixels (blue dots)
- How? Assume linear change => interpolate

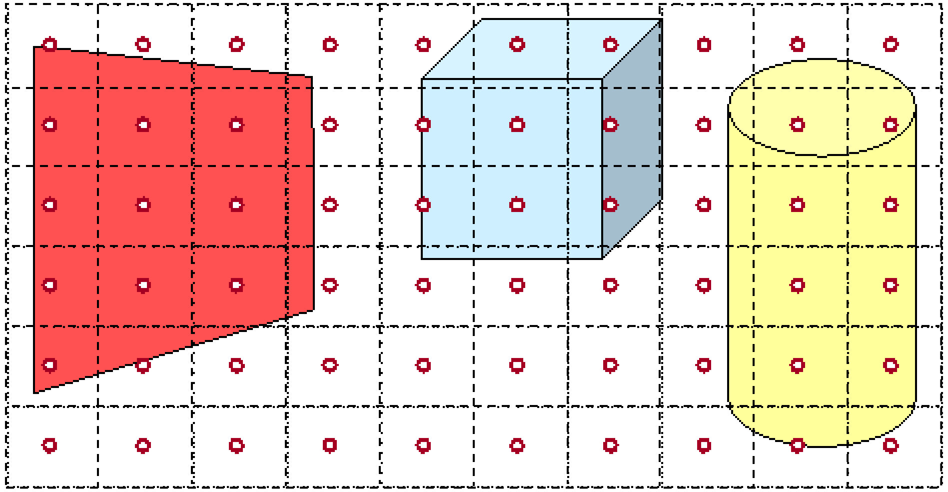


Shading Methods

- Ray Casting
  - Polygon Shading
- Ray Tracing
- Radiosity

Ray Casting

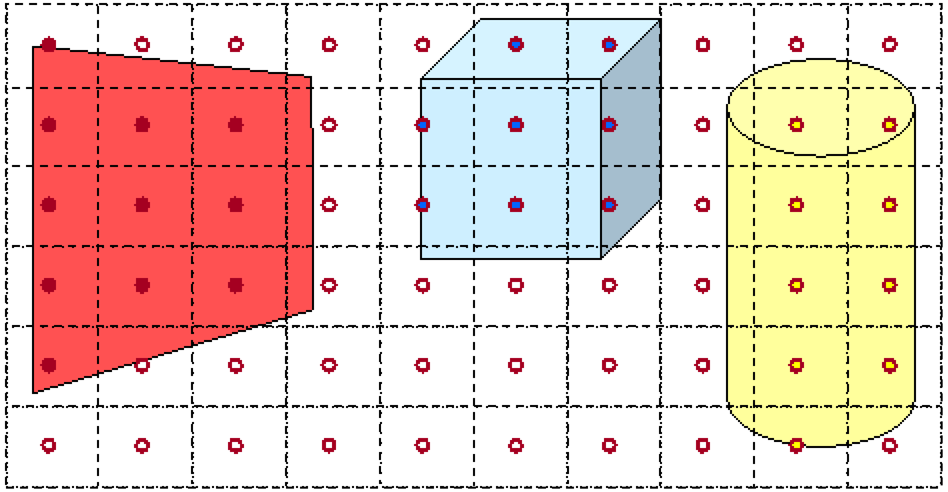
- Simplest Shading Approach
  - Perform independent lighting calculation for every pixel



$$I = I_E + K_A I_{AL} + \sum_i (K_D (N \cdot L_i) I_i + K_S (V \cdot R_i)^n I_i)$$

Polygon Shading

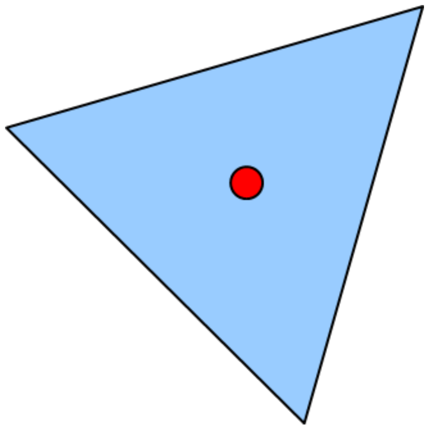
- Can Take Advantage of Spatial Coherence
  - Illumination calculations for pixels covered by same primitive are related to each other

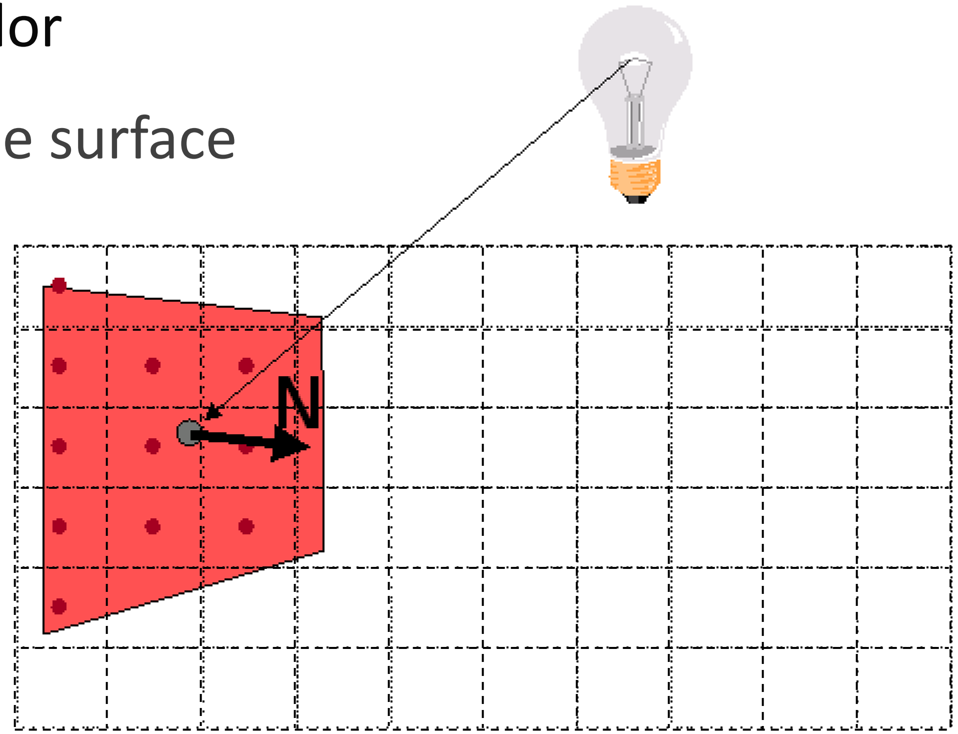


$$I = I_E + K_A I_{AL} + \sum_i (K_D (N \cdot L_i) I_i + K_S (\mathrm{V} \cdot \mathrm{R}_i)^n I_i)$$

Flat Shading

- Simplest method, same color is assigned to all surface positions
- One Illumination Calculation per Polygon
  - Assign all pixels inside each polygon the same color
- Illumination at a single point (usually center) on the surface is calculated and used for the entire surface

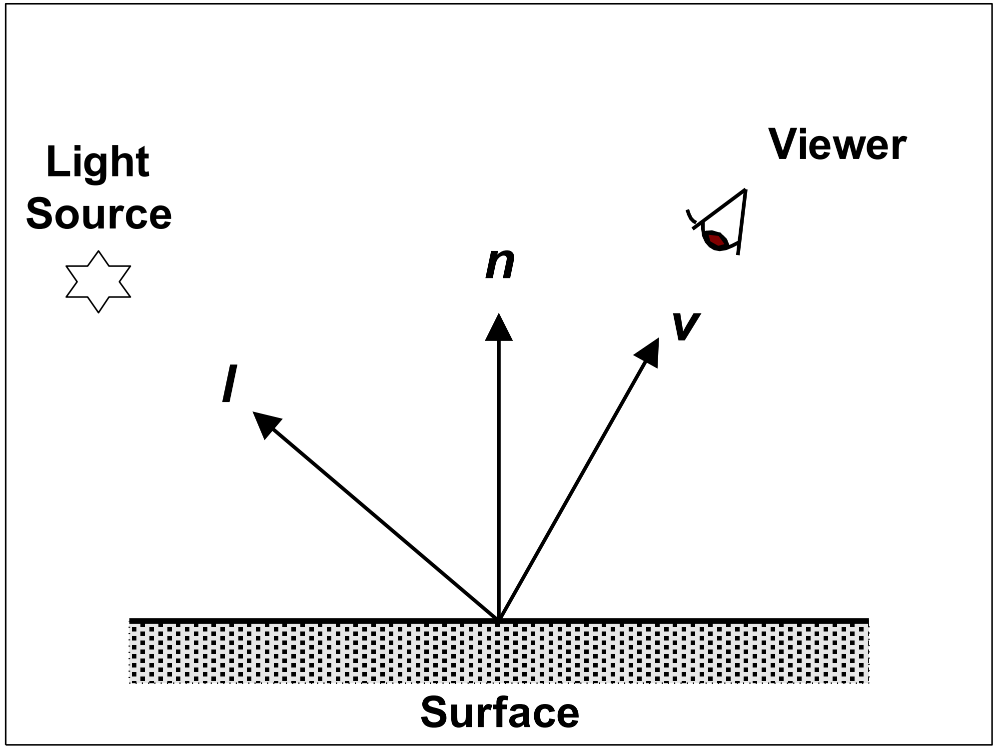




Flat Surface Rendering

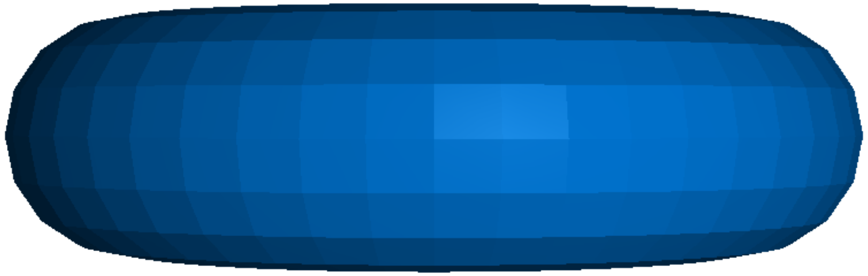
- Assumptions - For each surface
  - Light source at infinity $\vec{n} \cdot \vec{l}$ is constant
  - Viewer at infinity $\vec{n} \cdot \vec{v}$ is constant
  - The polygon represents the actual surface being modeled

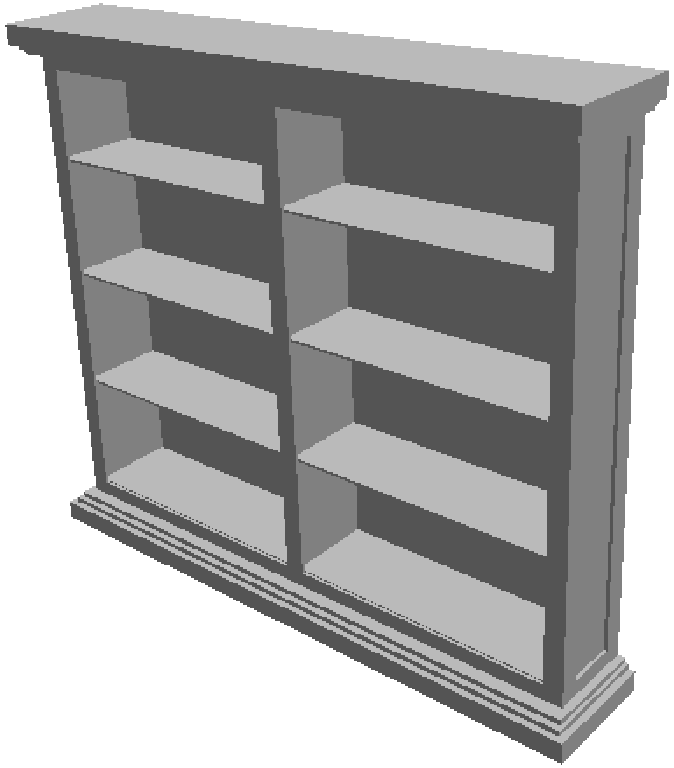
$$I = I_E + K_A I_{AL} + \sum_i \left(K_D (N \cdot L_i) I_i + K_S (\mathrm{V} \cdot \mathrm{R}_i)^n I_i\right)$$



Flat Surface Rendering

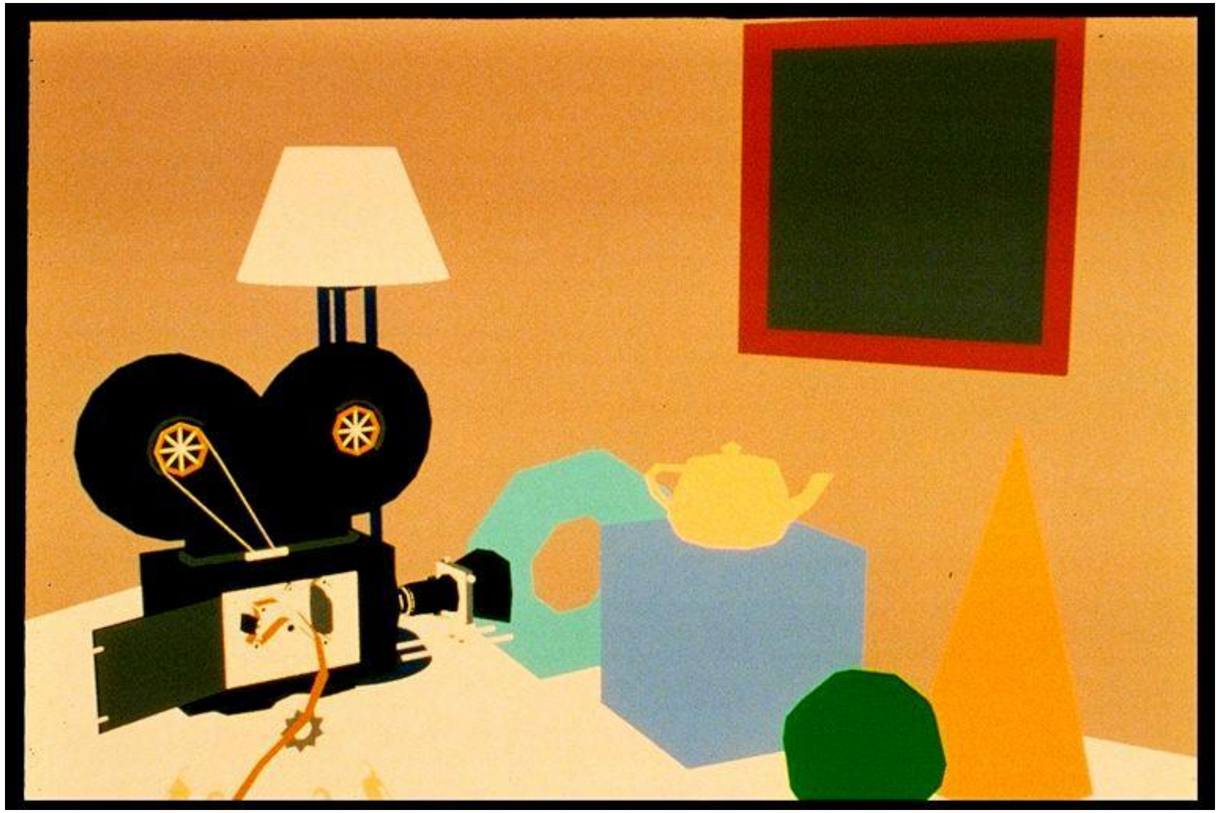
- Ok if:
  - Object consists of planar faces, and
  - Light sources are far away, and
  - Eye point is far away,
- or
  - Polygons are about a pixel in size.



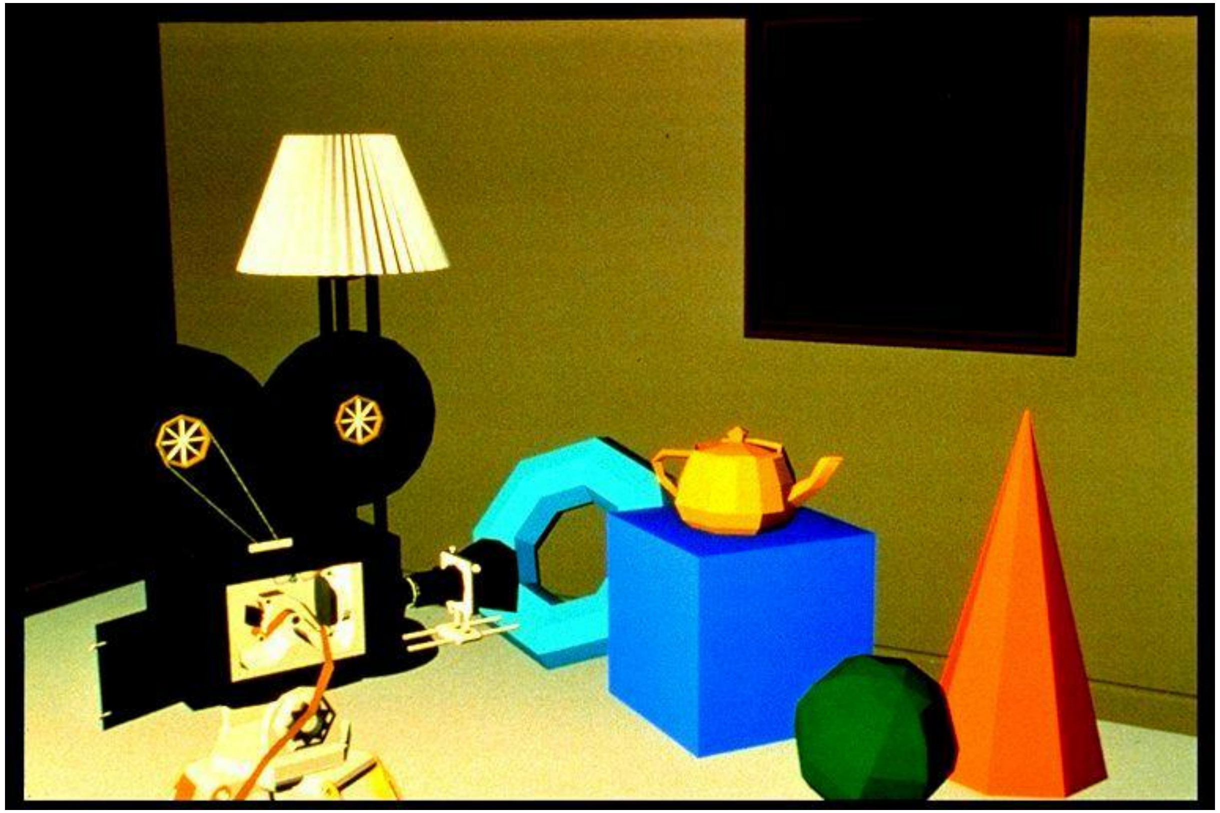


- Surface rendering is extremely fast, but can be unrealistic
  - Highlights not visible,
  - Facetted appearance, increased by Mach banding effect.

No Surface Rendering Vs Flat Surface Rendering



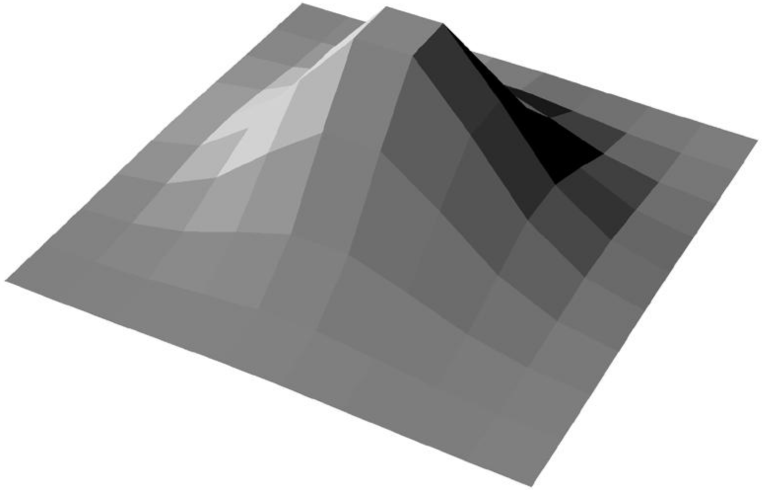
No Surface Rendering

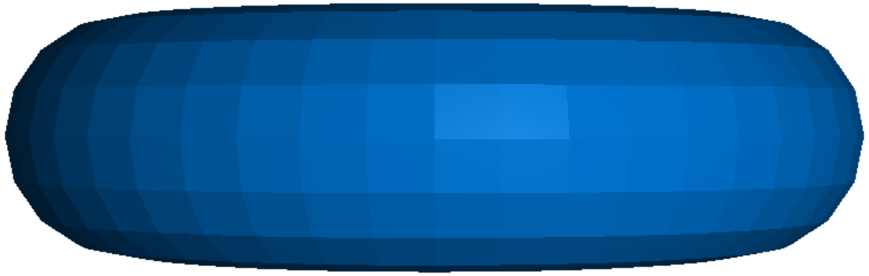


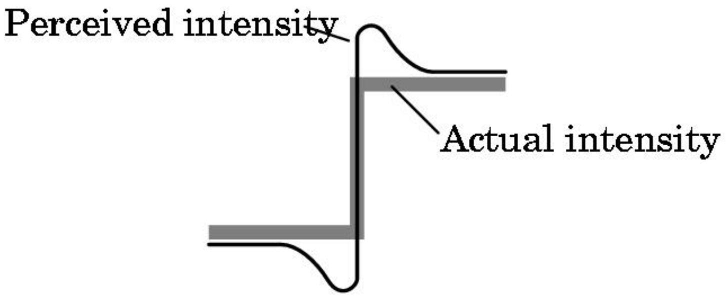
Flat Surface Rendering

Flat shading drawbacks

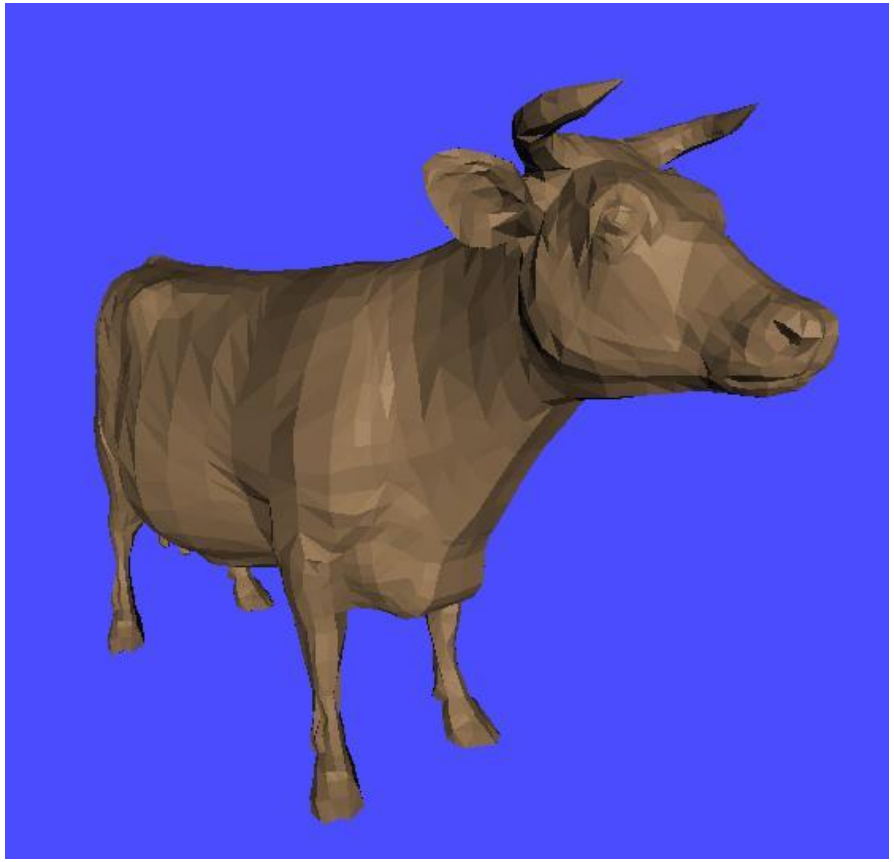
- The human visual system enhances edges
- We see stripes
  (known as Mach Bands) along edges
- Much like a sharpening convolution!
- How to avoid?

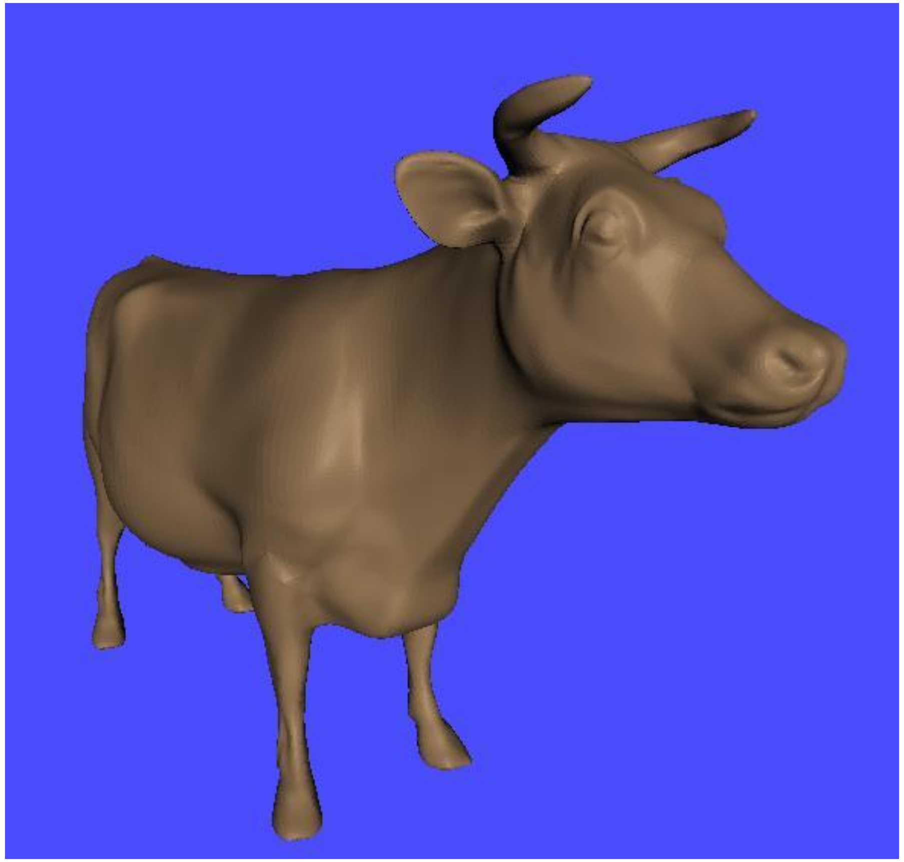






Overcoming Flat Shading Limitations





- Just add lots and lots of polygons – however, this is SLOW!

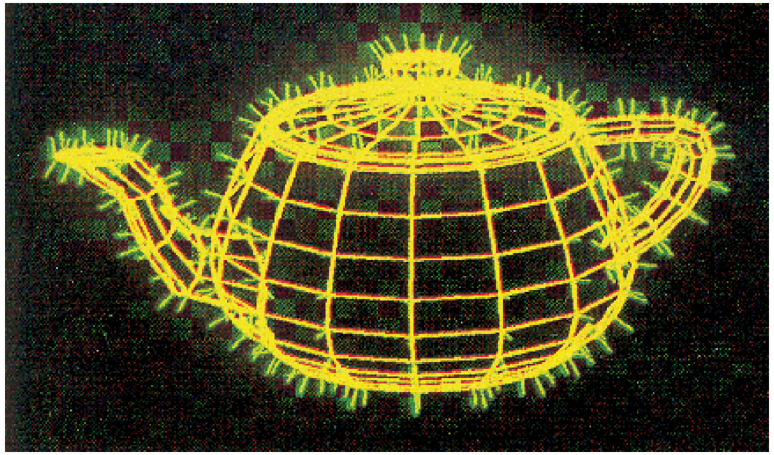
Gouraud Shading

- Developed in the 1970s by Henri Gouraud
- Worked at the University of Utah along with Ivan Sutherland and David Evans
- Often also called **intensity- interpolation surface rendering**

- Intensity levels are calculated at each vertex and interpolated across the surface

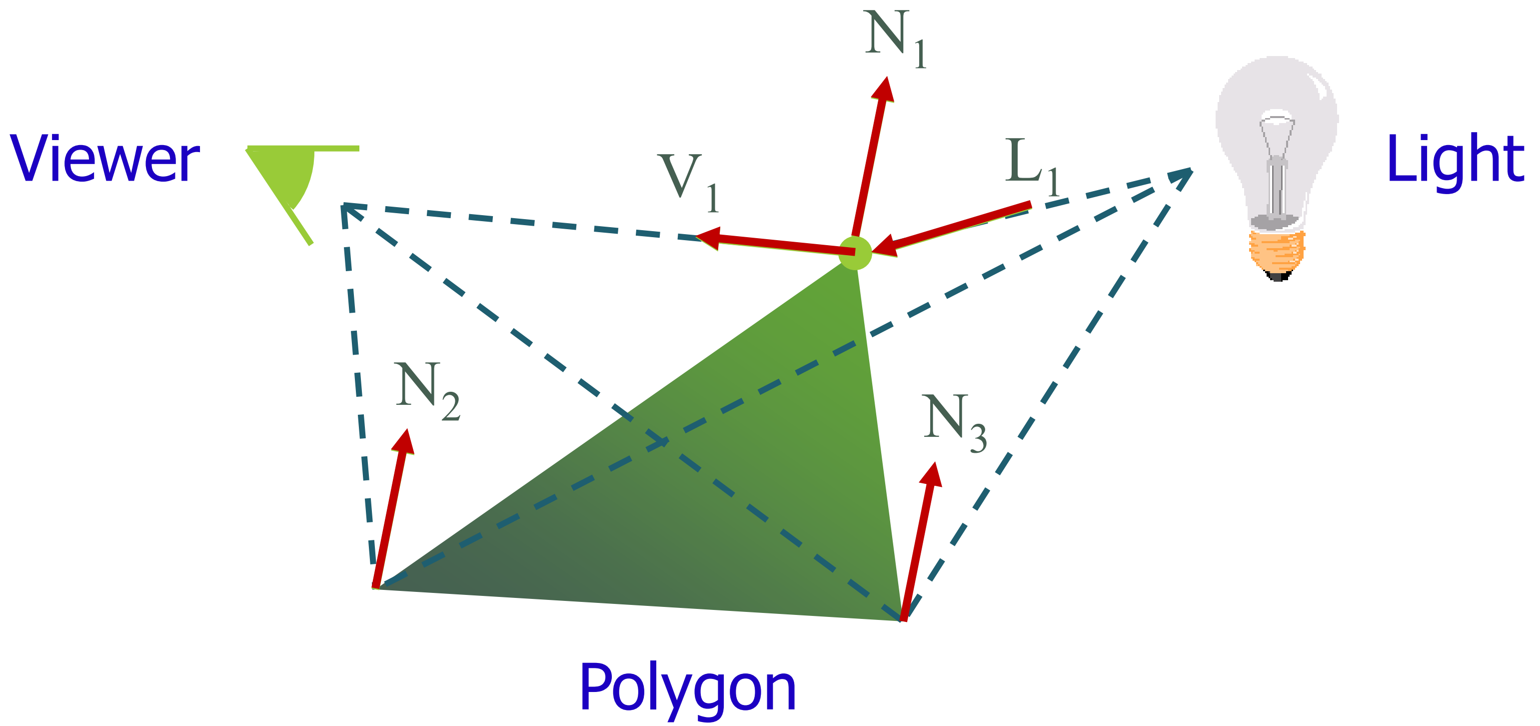
Gouraud Shading

- Smooth Surface are
  - Represented by polygonal mesh with a normal at each vertex



Gouraud Shading

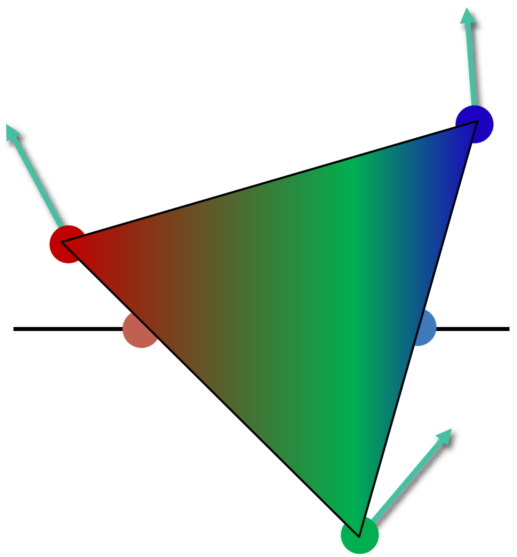
- One Lighting Calculation per Vertex
  - Assign pixels inside polygon by interpolating/ lerping colors computed at vertices



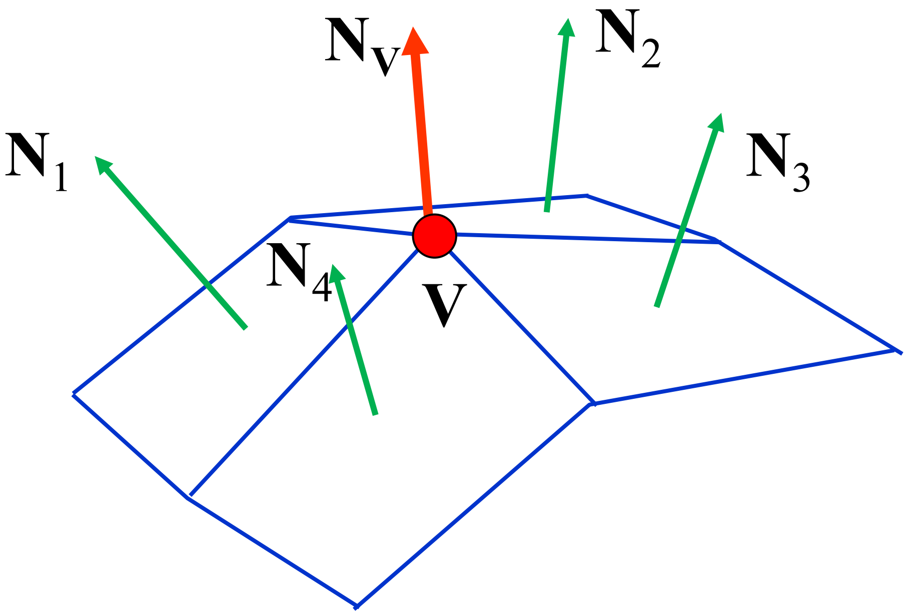
$$I = I_E + K_A I_{AL} + \sum_i \left(K_D (N \cdot L_i) I_i + K_S (V \cdot R_i)^n I_i \right)$$

Gouraud Shading

- To render a polygon, Gouraud surface rendering proceeds as follows:
  1. Determine the average unit **normal** vector **at each vertex** of the polygon
  2. **Apply an illumination model** at each polygon vertex to obtain the **light intensity** at that position
  3. **Linearly interpolate** the vertex intensities over the projected area of the polygon



Gouraud Shading



- The average unit normal vector at V is given as:

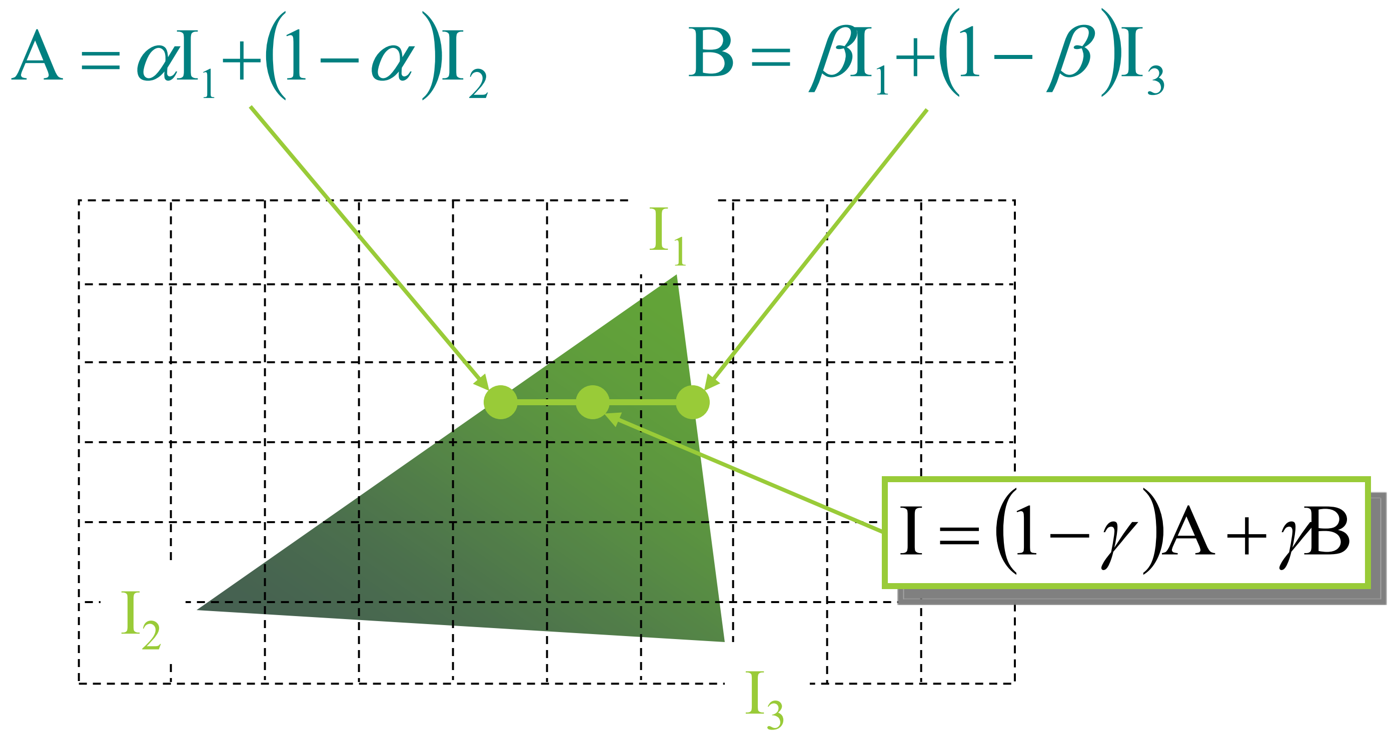
$$N_v = \frac{N_1 + N_2 + N_3 + N_4}{\left|N_1 + N_2 + N_3 + N_4\right|}$$

- or more generally:

$$N_v = \frac{\sum_{i=1}^{n} N_i}{\left|\sum_{i=1}^{n} N_i\right|}$$

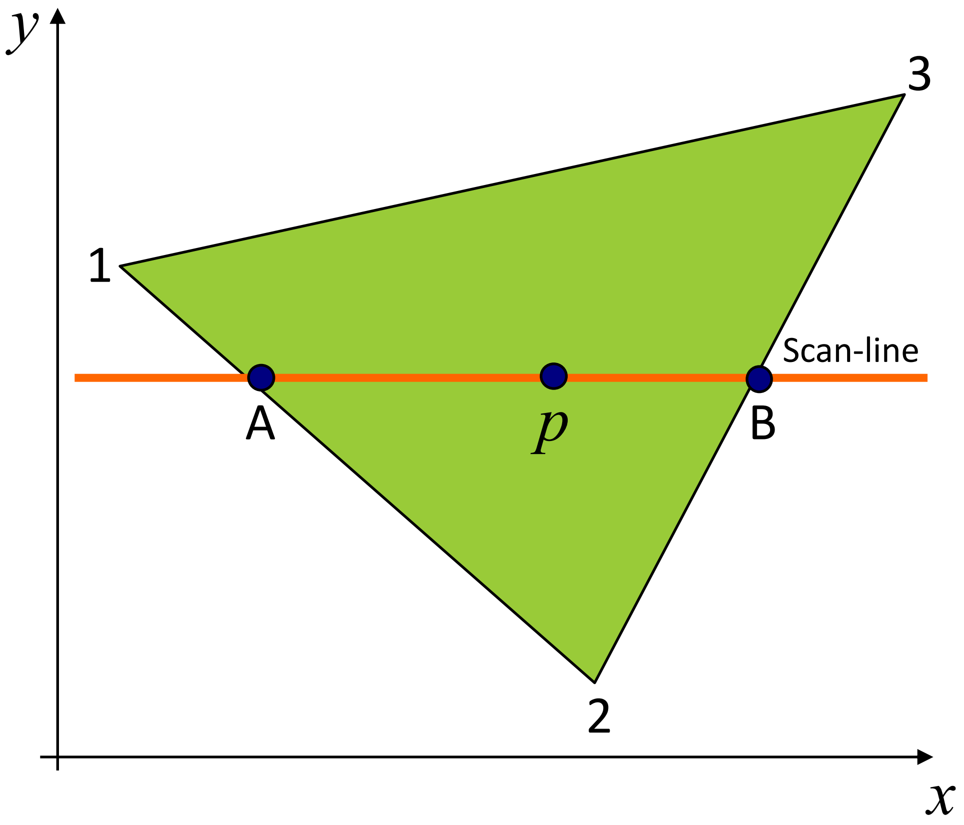
Gouraud Shading

- Bilinearly Interpolate Colors at Vertices Down and Across Scan Lines



Gouraud Shading

- Illumination values are bilinearly interpolated across each scan-line



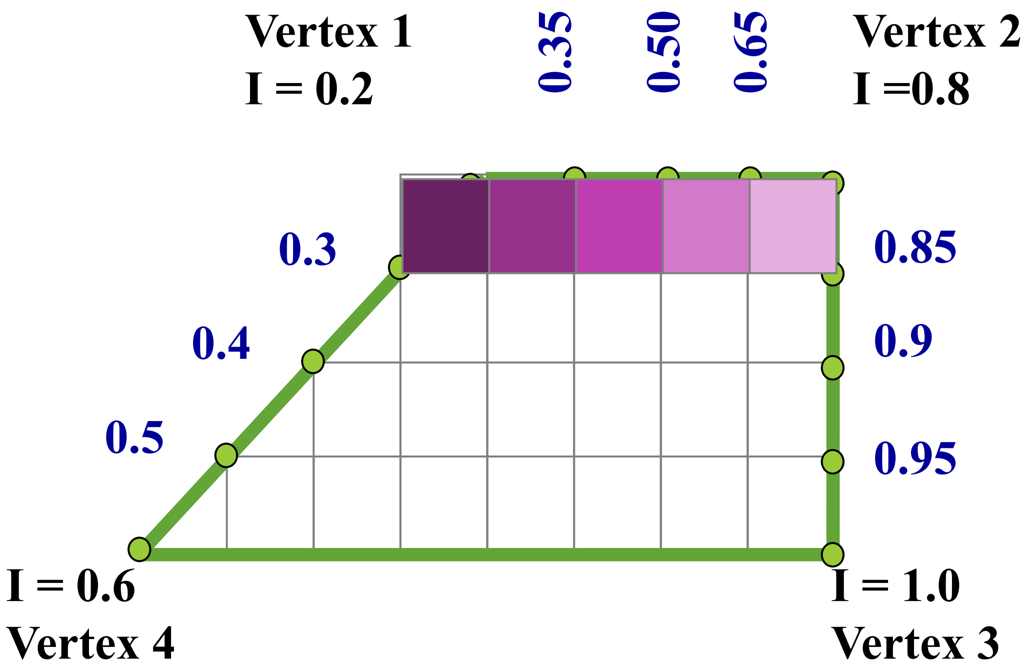
$$I_A = \frac{y_A - y_2}{y_1 - y_2} I_1 + \frac{y_1 - y_A}{y_1 - y_2} I_2$$

$$I_B = \frac{y_B - y_2}{y_3 - y_2} I_3 + \frac{y_3 - y_B}{y_3 - y_2} I_2$$

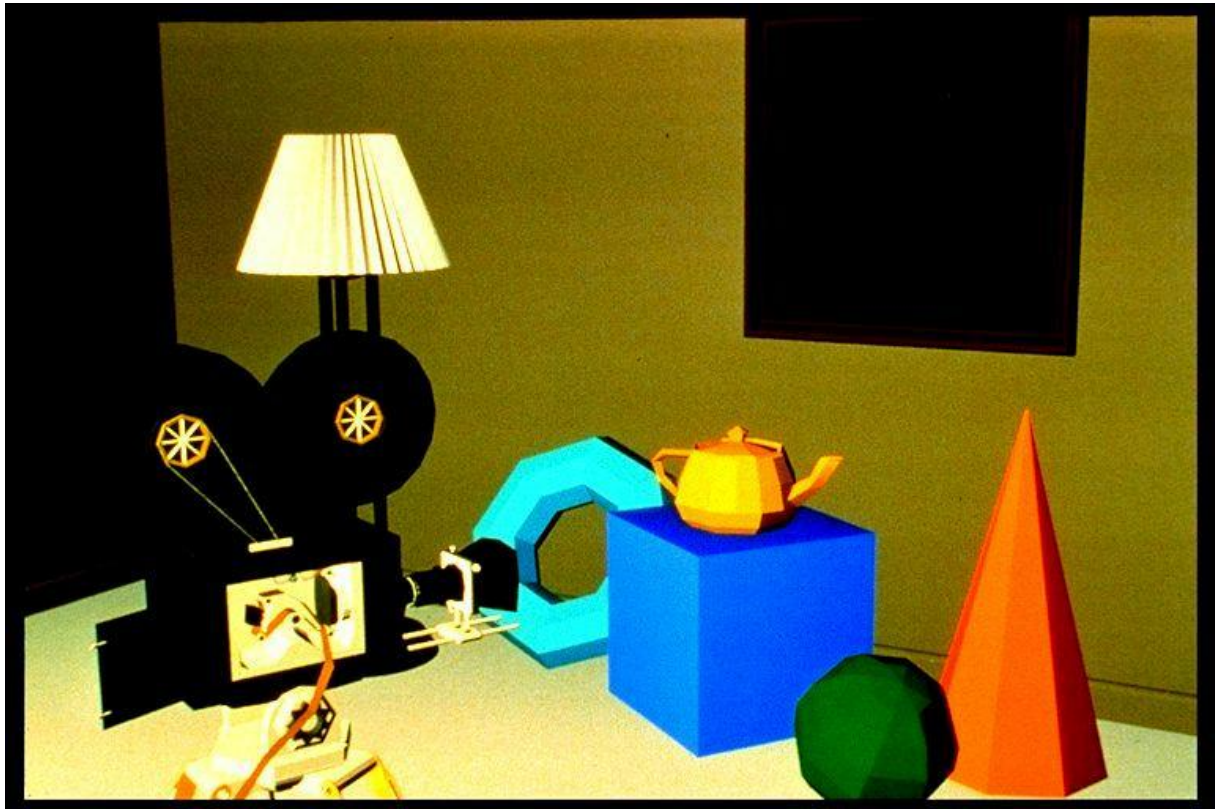
$$I_p = \frac{x_B - x_p}{x_B - x_A} I_A + \frac{x_p - x_A}{x_B - x_A} I_B$$

Gouraud Shading

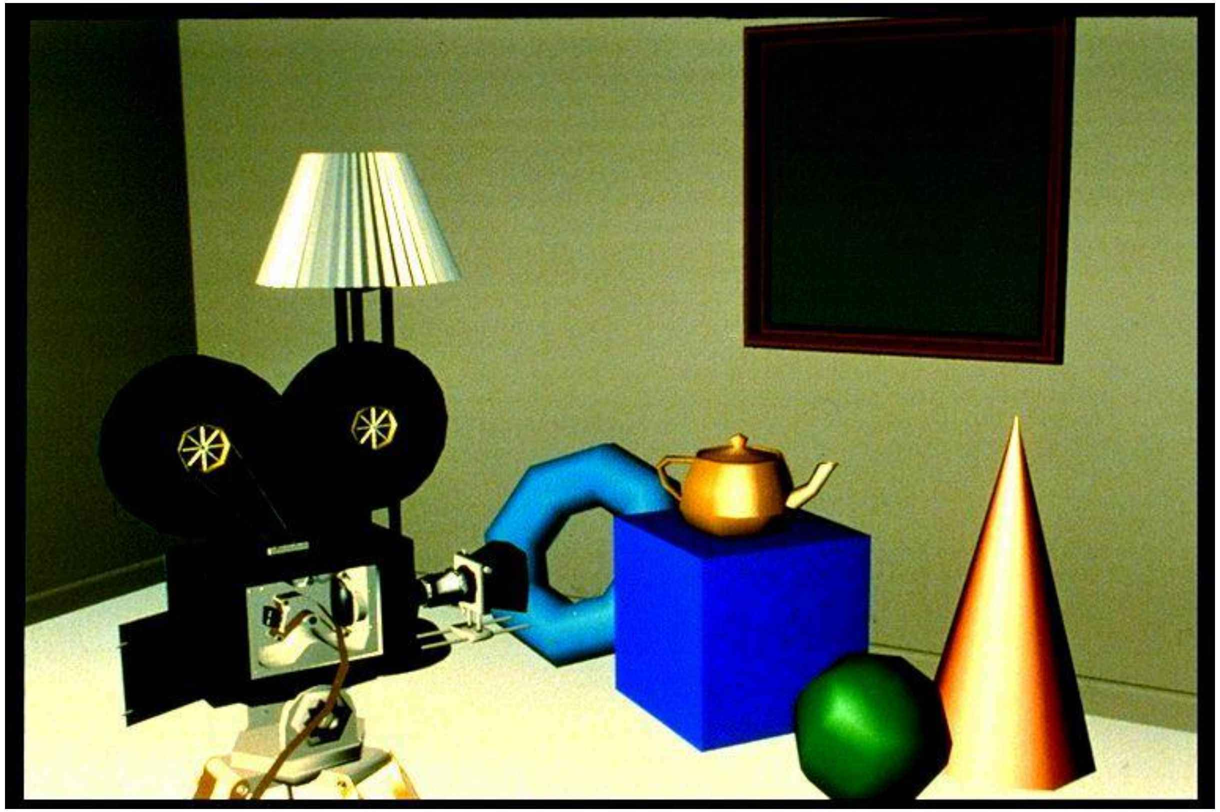
An Example



Flat Vs Gouraud Rendering



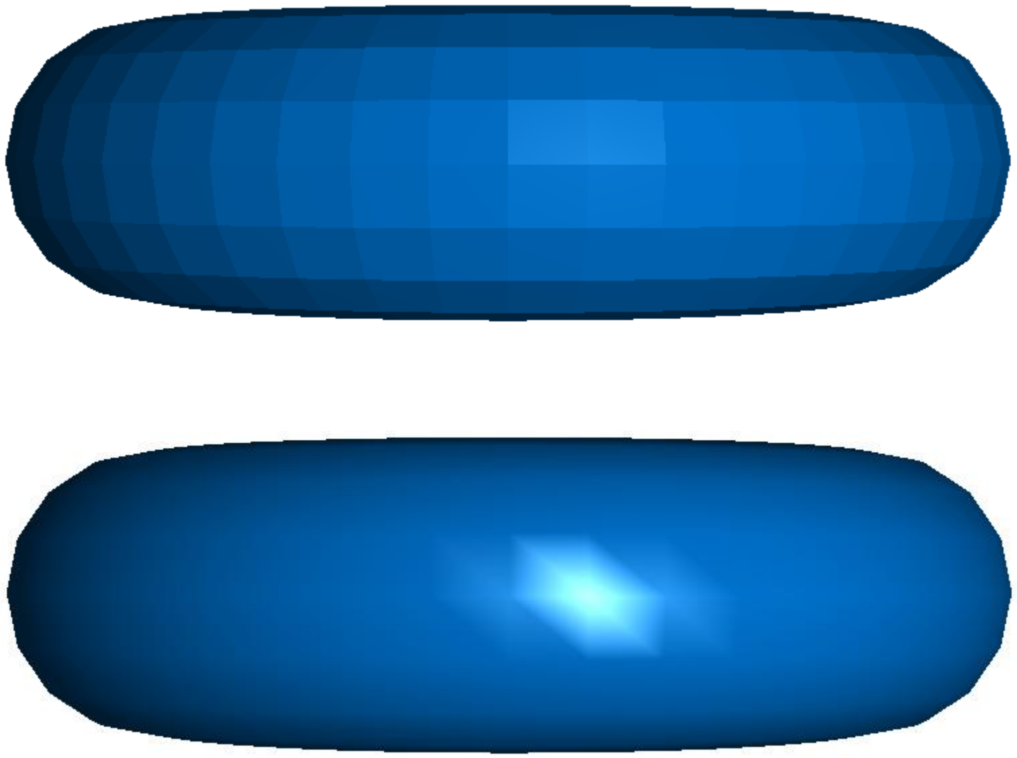
Flat Surface Rendering

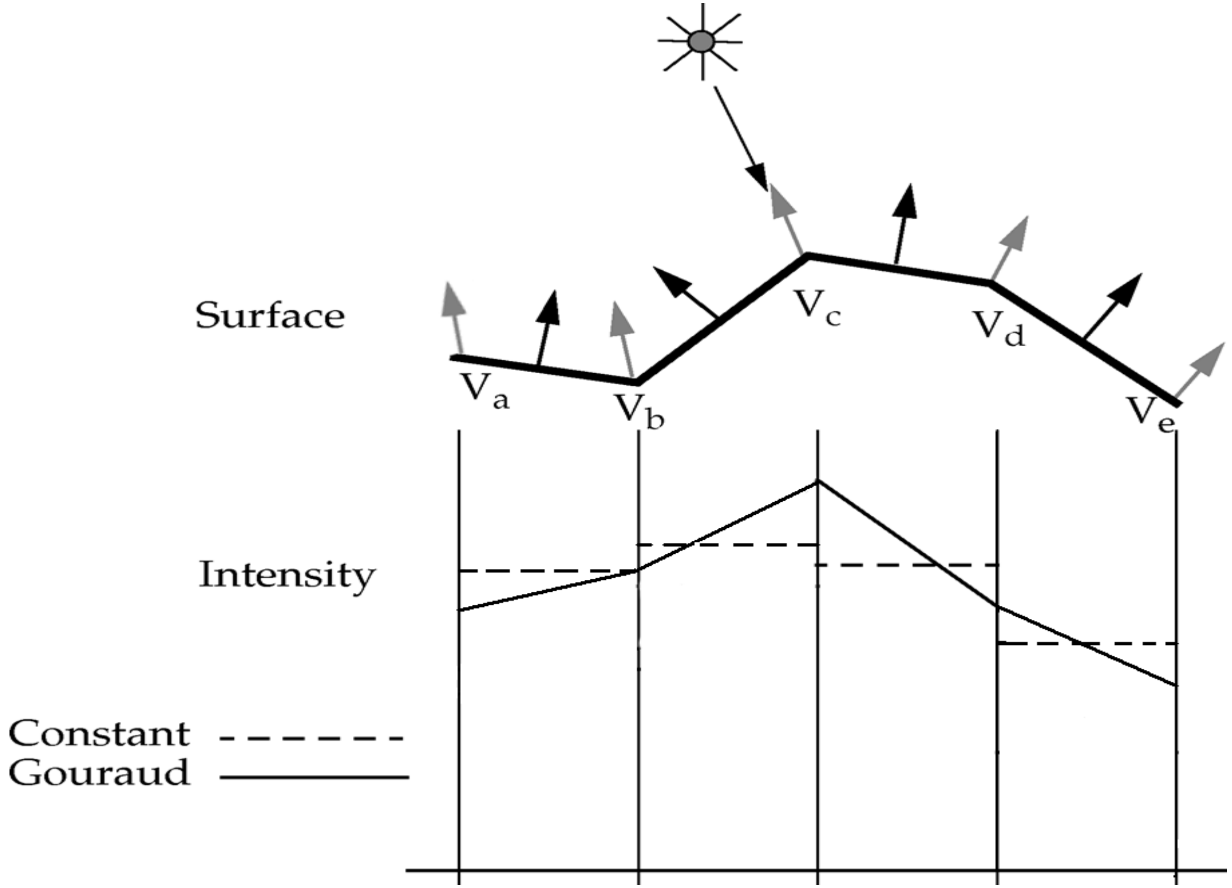


Gouraud Surface Rendering

Gouraud Shading

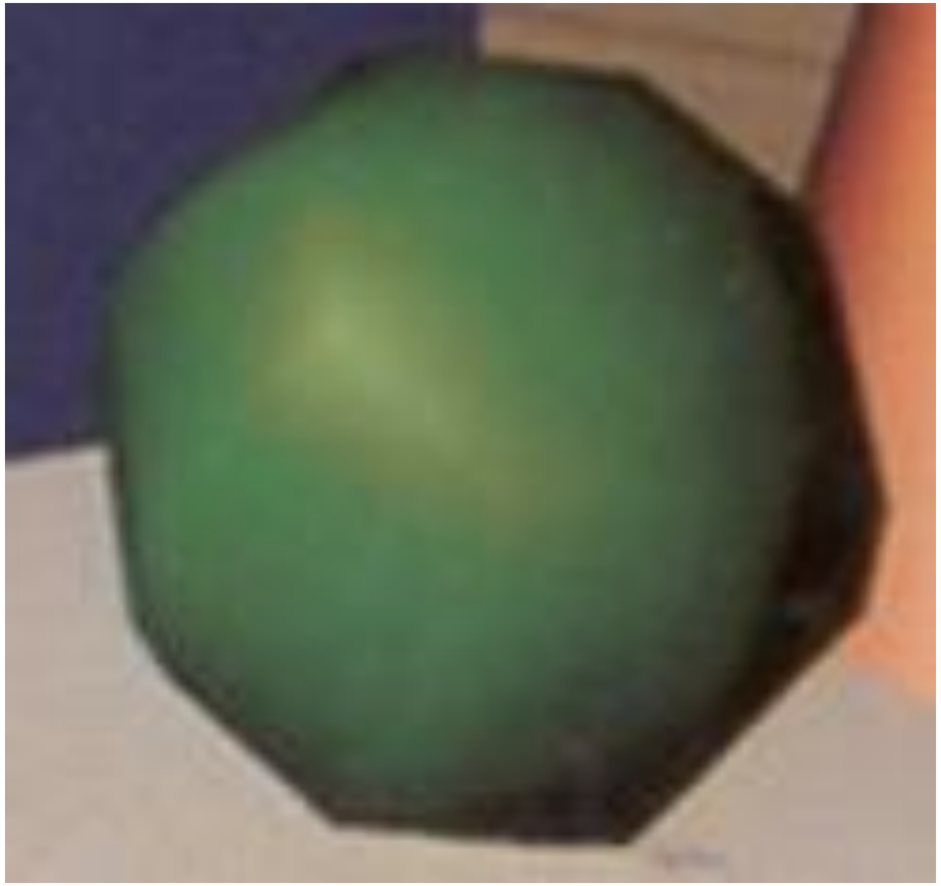
- Much better result for curved surfaces





Gouraud Shading - Drawbacks

- Polygon edges are still visible
- Brightness is modelled as a linear function, but that’s not really accurate
- Real highlights are small and bright, and drop off sharply
  - If polygons are too large, highlights get distorted and dimmed (notice the funny shape)

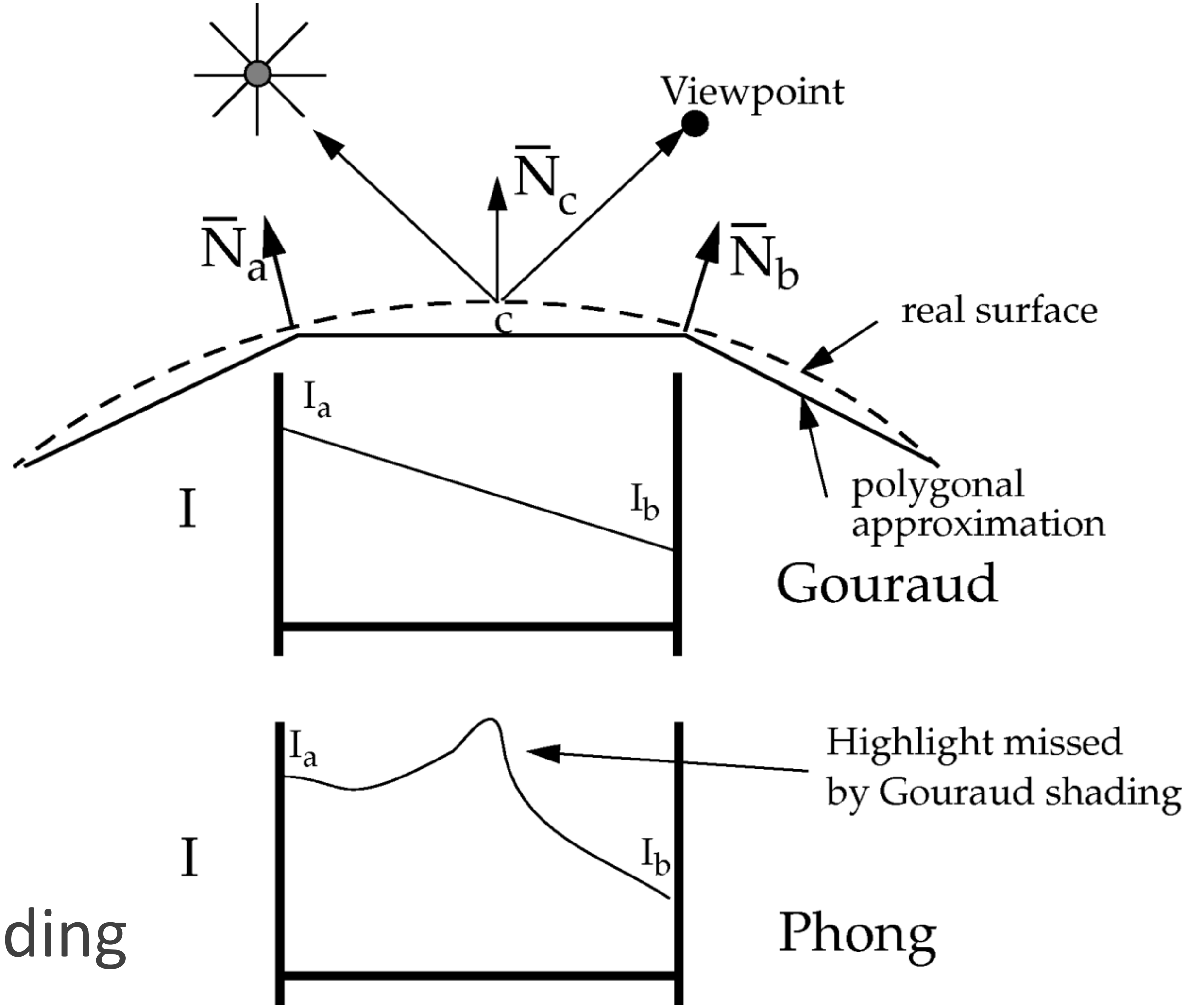


- How to avoid these artifacts?

Gouraud Shading - Drawbacks

- It has a problem with

  specular reflections
  - Completely miss
  - interpolate



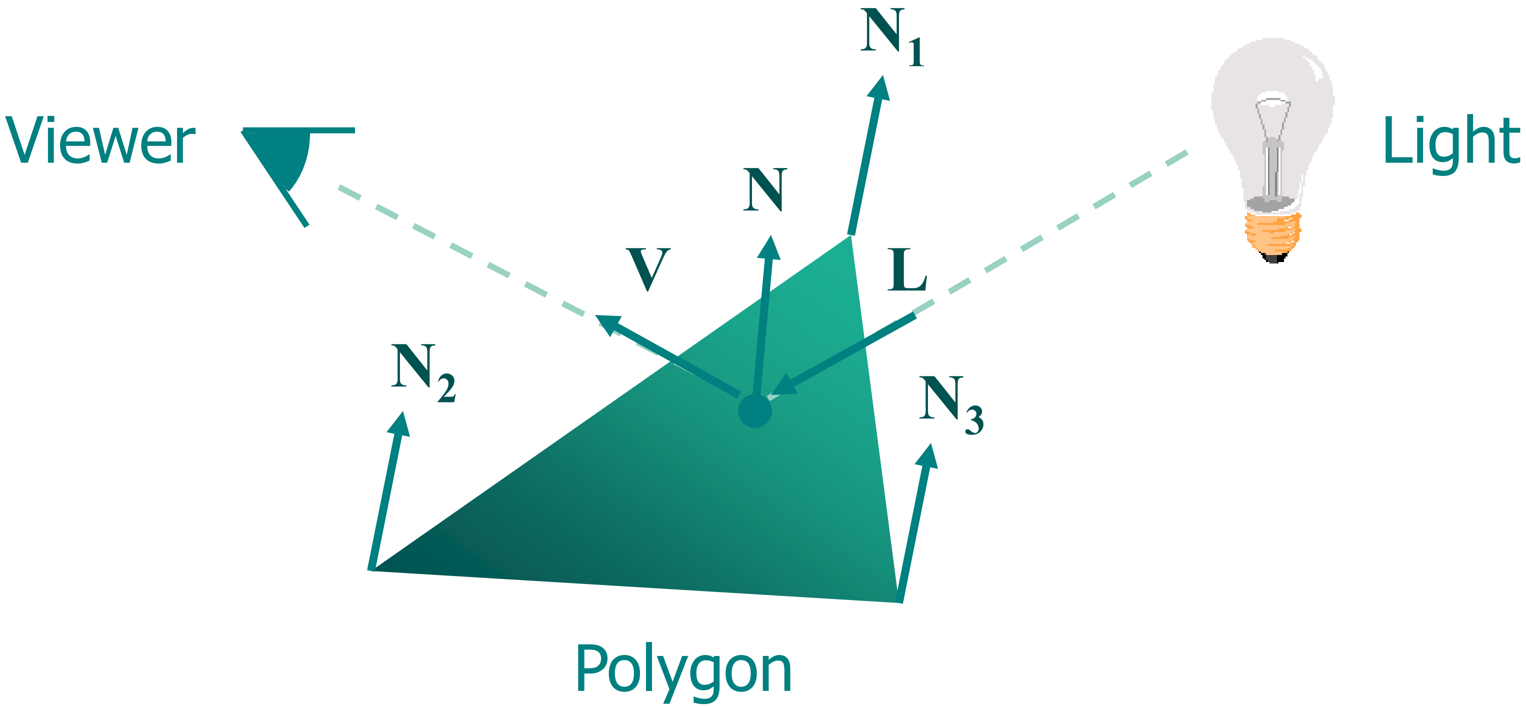
- Linear interpolation still gives Mach banding
  - Silhouettes are still not smooth

Phong Shading

- A more accurate interpolation based approach for rendering a polygon was developed by Phong Bui Tuong
- Basically the Phong surface rendering model (or **normal-vector interpolation rendering**) interpolates normal vectors instead of intensity values

Phong Shading

- One Lighting Calculation per Pixel
  - Approximate surface normals for points inside polygons by bilinear interpolation of normals from vertices

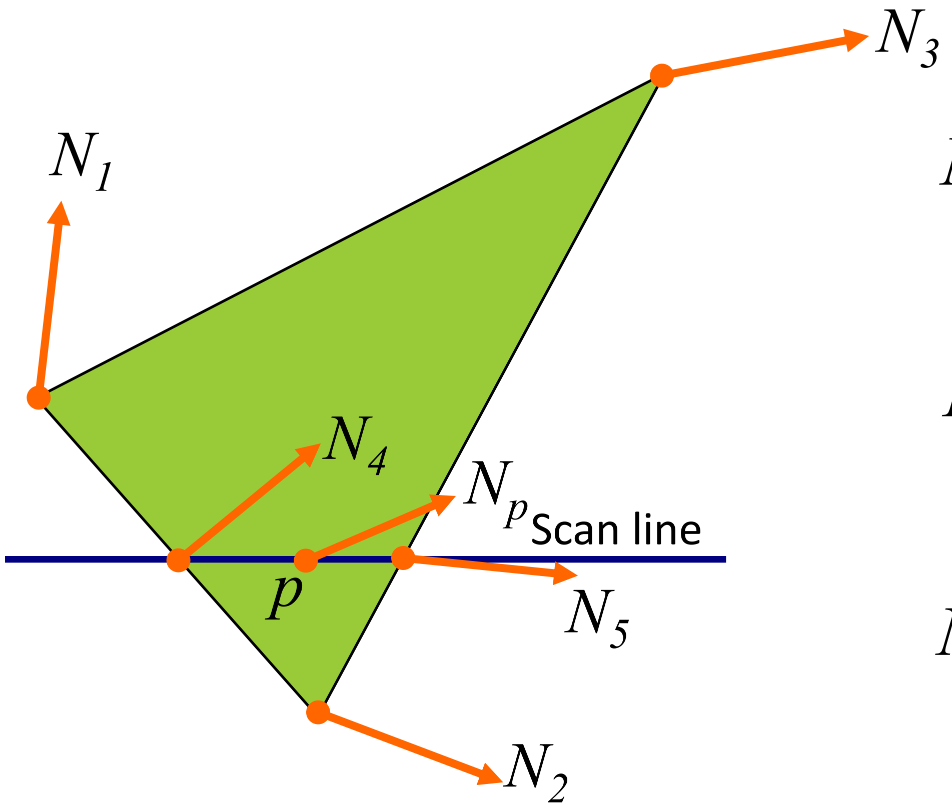


Phong Shading

- To render a polygon, Phong surface rendering proceeds as follows:
  1. Determine the average unit normal vector at each vertex of the polygon
  2. Linearly **interpolate** the vertex **normals** over the projected area of the polygon
     - Normalize it.
     - (Interpolation of unit vectors does not preserve length).
  3. Apply an illumination model at positions along scan lines to calculate pixel intensities using the interpolated normal vectors

Phong Shading

- Bilinearly Interpolate Normals at Vertices Down and Across Scan Lines

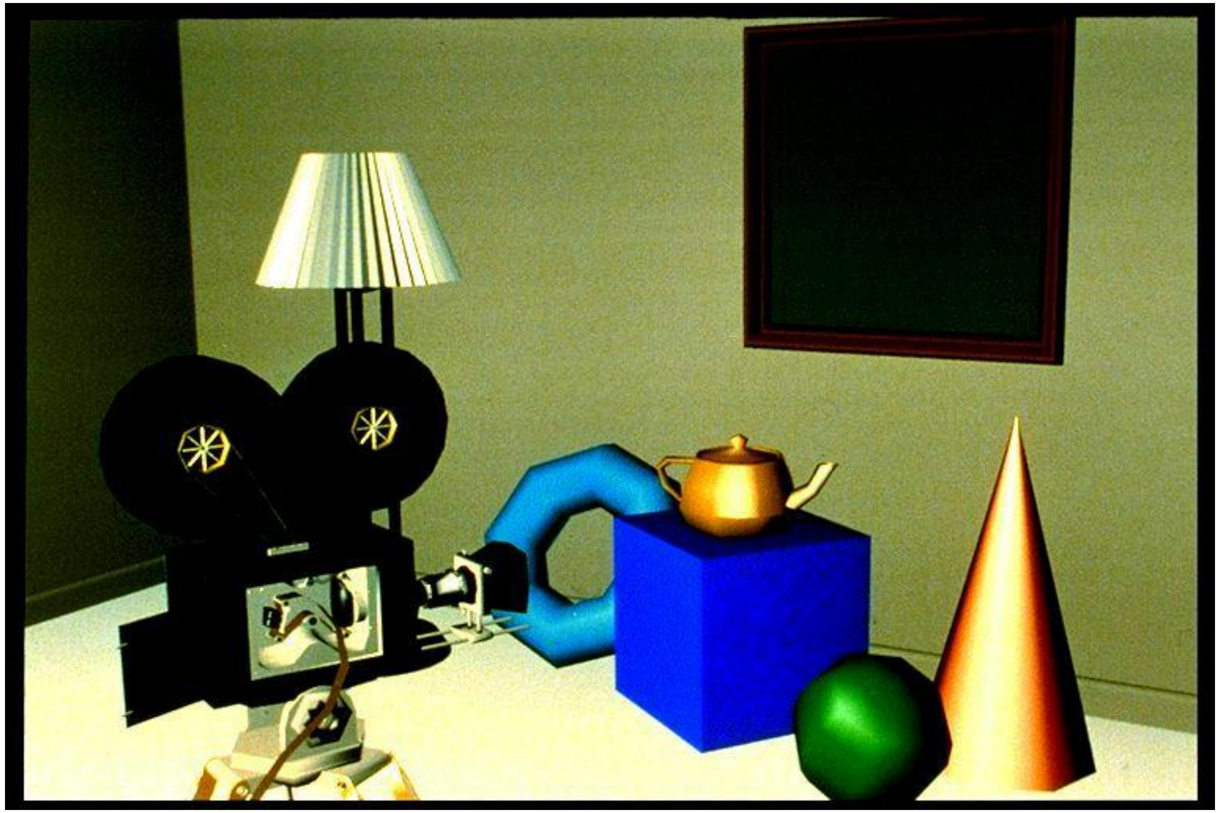


$$N_4 = \frac{y_4 - y_2}{y_1 - y_2} N_1 + \frac{y_1 - y_4}{y_1 - y_2} N_2$$

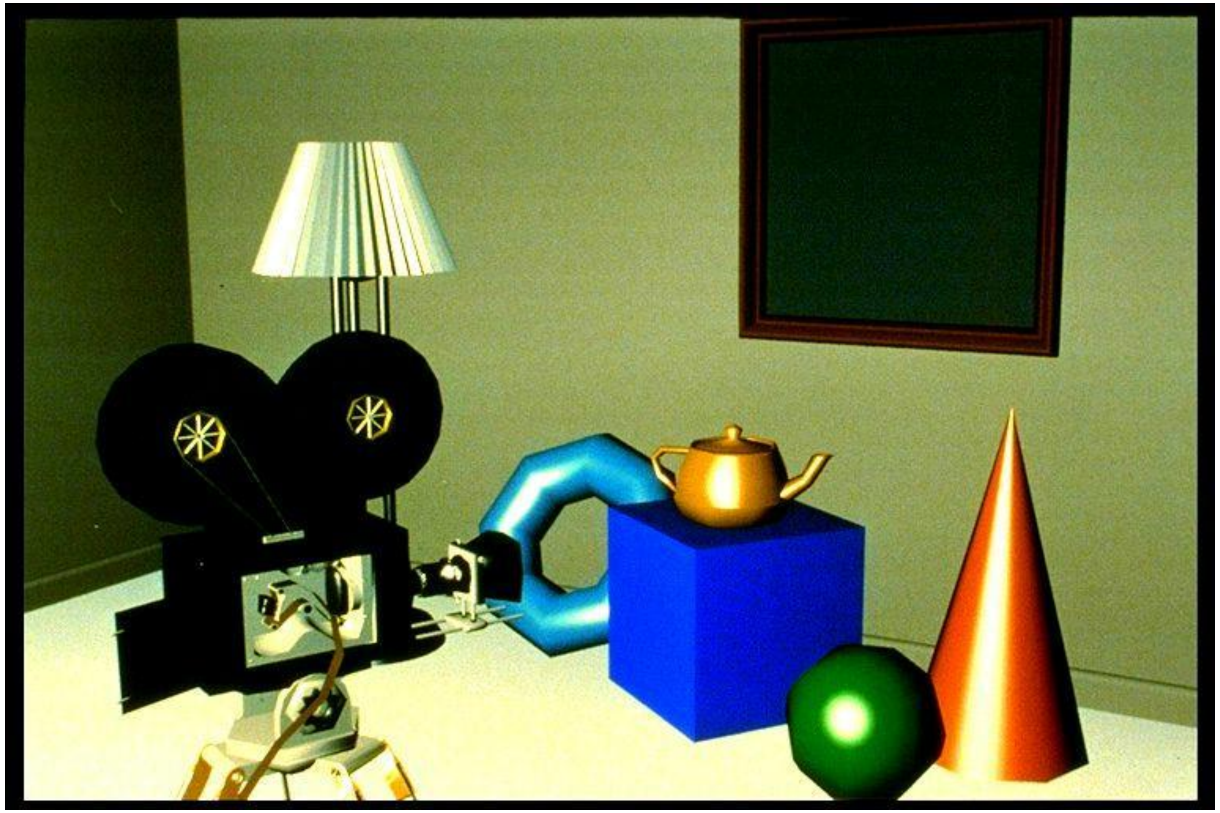
$$N_5 = \frac{y_5 - y_2}{y_3 - y_2} N_3 + \frac{y_3 - y_5}{y_3 - y_2} N_2$$

$$N_p = \frac{y_p - y_5}{y_4 - y_5} N_4 + \frac{y_4 - y_p}{y_4 - y_5} N_5$$

Gouraud Vs Phong Surface Rendering

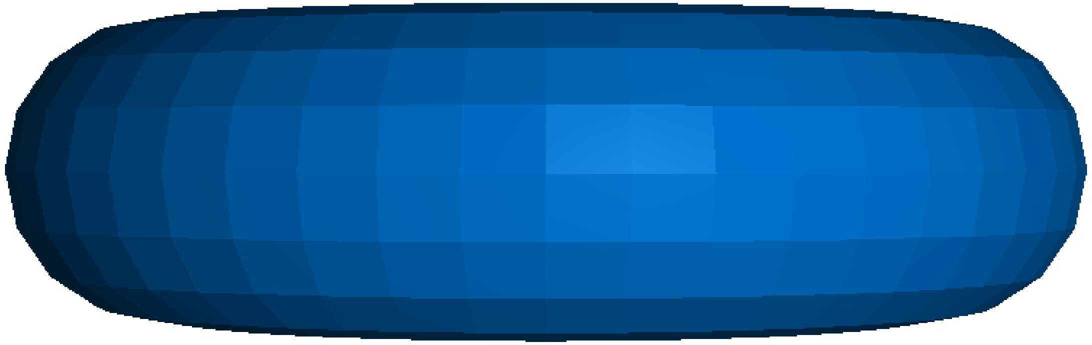


Gouraud Surface Rendering

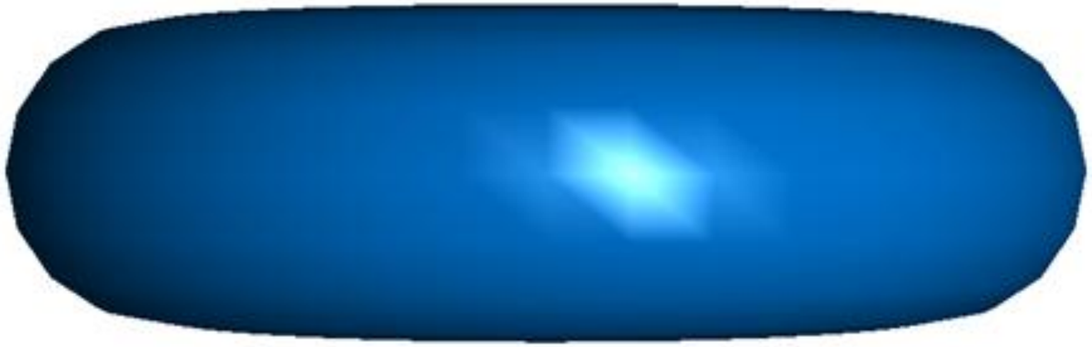


Phong Surface Rendering

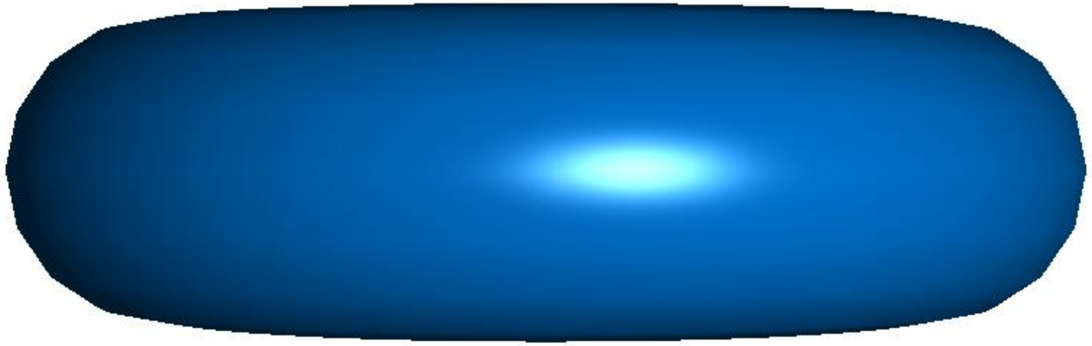
Phong Shading



Flat



Gouraud



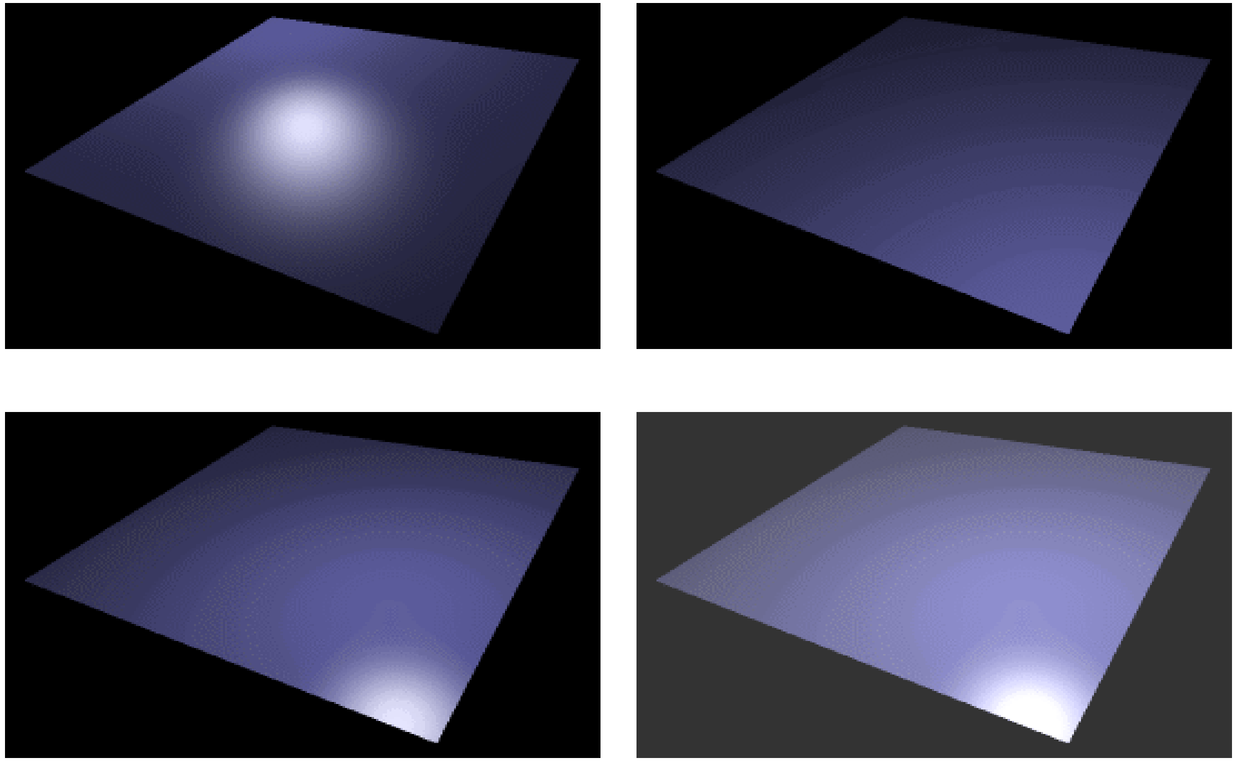
Phong

Phong Shading

- Even better result for curved surfaces
- No errors at high lights
- No Mach banding
- Phong shading is much slower than Gouraud shading as the lighting model is revaluated so many times
- There are **fast Phong** surface rendering approaches that can be implemented iteratively
- Typically, implemented as part of a visible surface detection technique
- **Not supported in OpenGL**

Phong vs Gouraud Shading

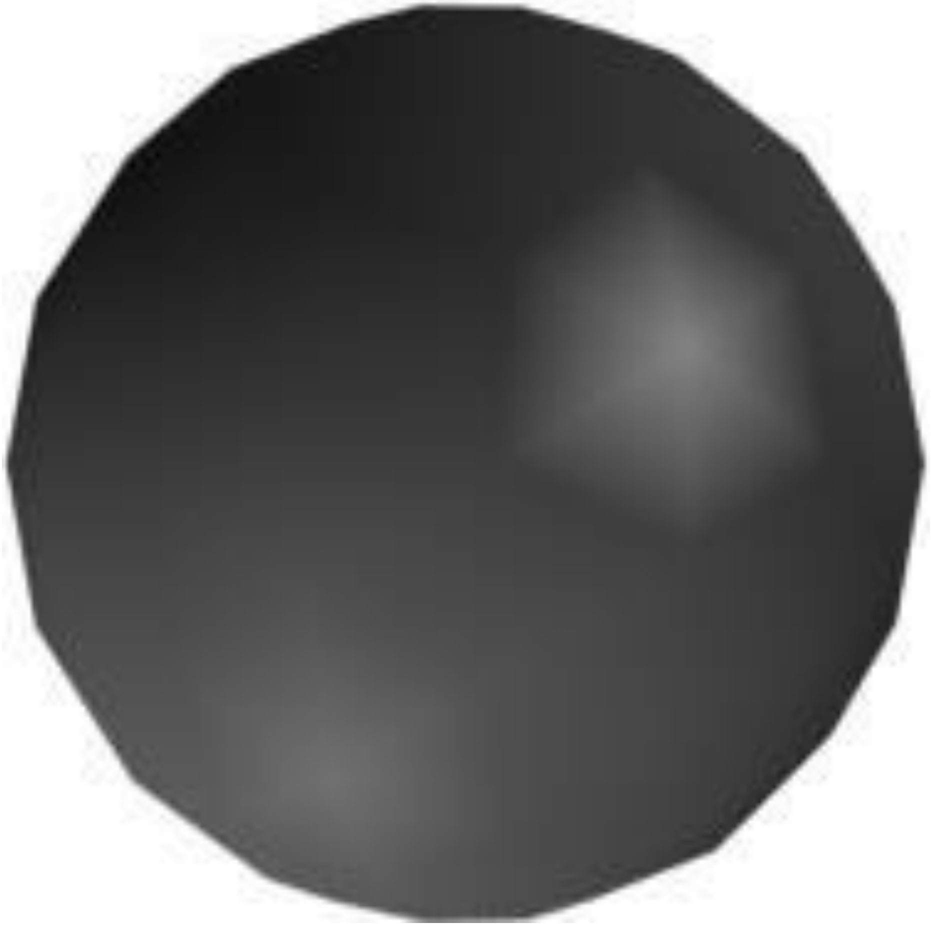
- If a highlight does not fall on a vertex Gouraud shading may miss it completely, but Phong shading does not.
- if highlight falls on vertex, Gourad shading will spread the highlight over the polygon



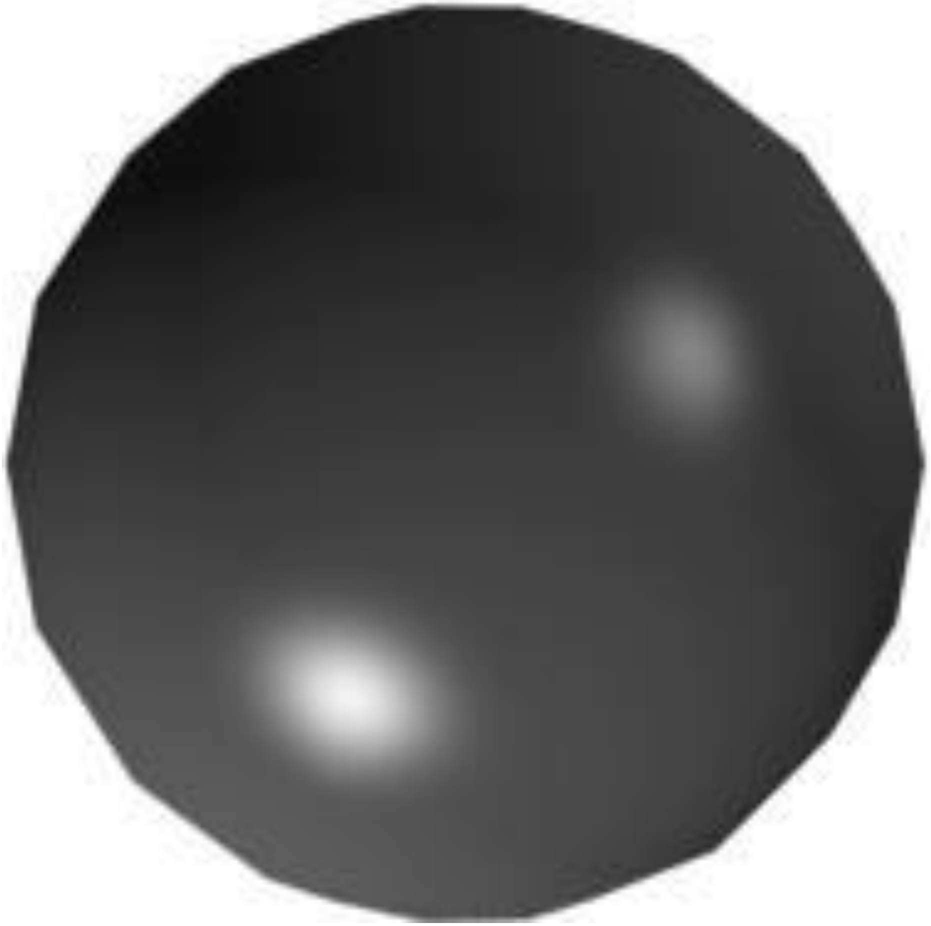
Gouraud Tea Pot Example

Phong Tea Pot Example

Phong vs Gouraud Shading



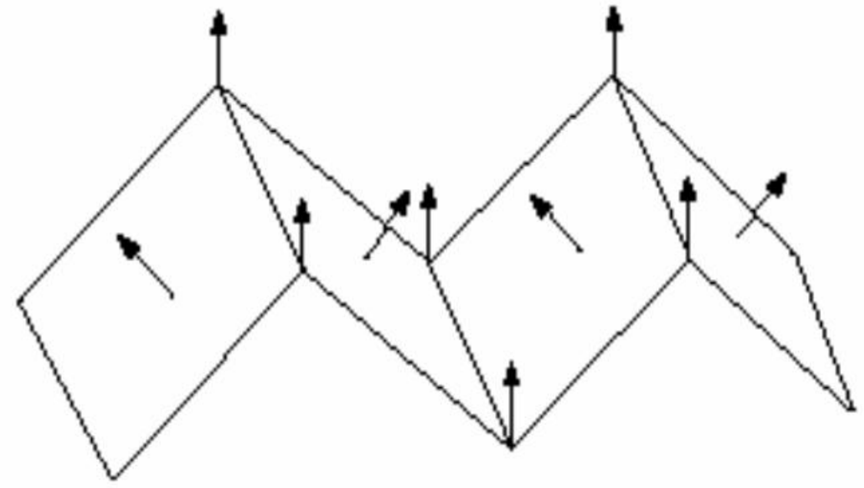
Gouraud



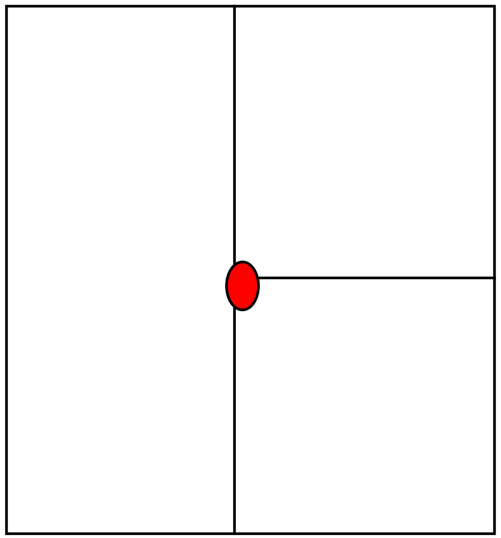
Phong

Interpolative Shading artifacts

- Vertex normal does not always reflect the curvature of the surface adequately
- Incorrect Vertex normals – no variation in shade
  - Appear less flat than actual



- The shading at the T-junction are different when calculated from different triangles
  - shared by right polygons and not by one on left
  - Shading discontinuity

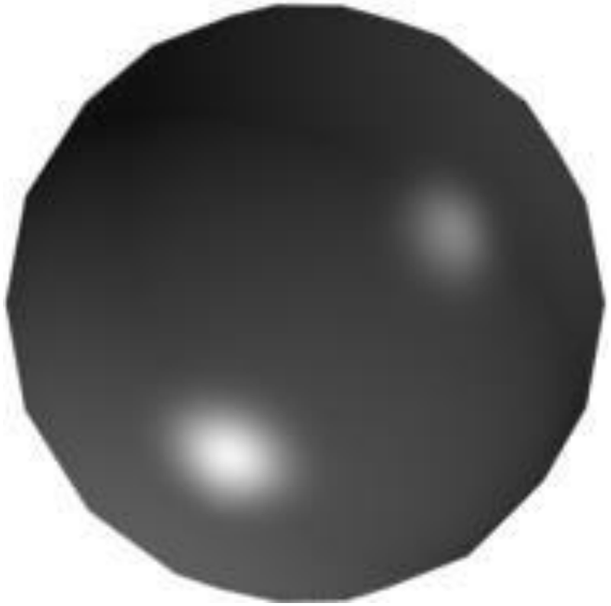


Interpolative Shading artifacts – Mach Bands

- 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

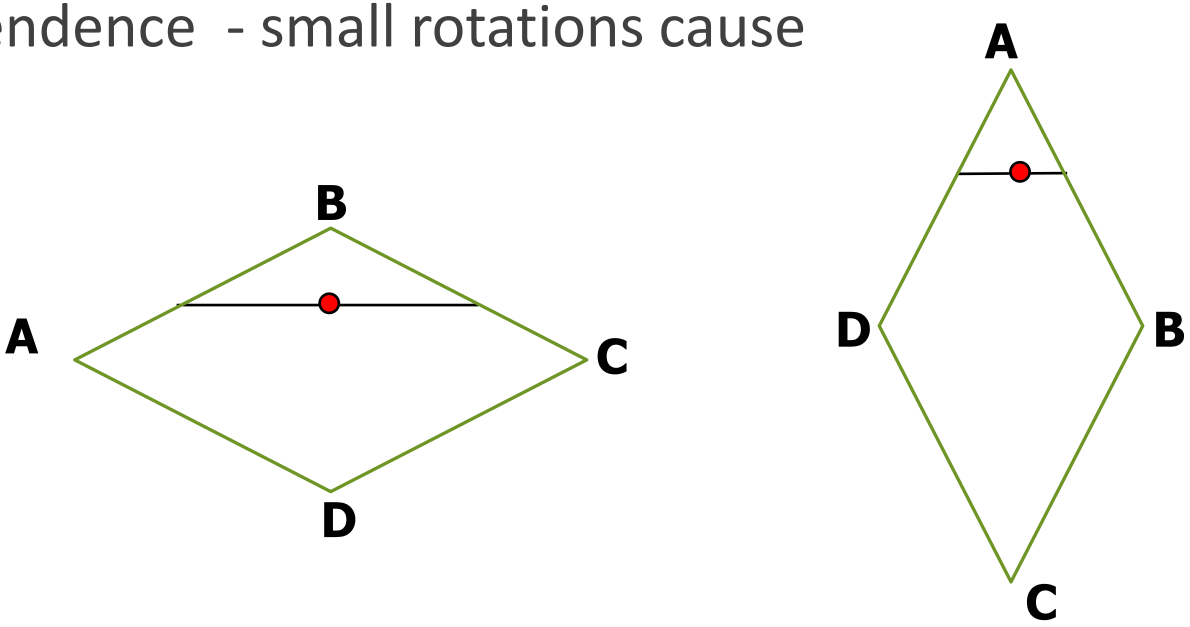
Interpolative Shading artifacts

- Polygonal silhouette – edge is always polygonal
- Perspective distortion – interpolation is in screen space and hence foreshortening takes place
- In both cases finer polygons can help !



Phong

- Orientation dependence - small rotations cause problems



Other Types of Per-pixel Shading

- Ray tracing.
  - Doesn't use Gouraud or Phong shading.
  - Each pixel uses own ray to determine color.
    - Can apply arbitrary lighting model.
    - Classical (Whitted) ray tracing uses Phong model.
  - Since ray tracing determines colors based on intersections, don't have to use polygonal geometry.
    - Thus, can potentially use exact normals, rather than interpolation.

Summary

- For realistic rendering of polygons we need interpolation methods to determine lighting positions
- Flat shading is fast, but unrealistic
- Gouraud shading is better, but does not handle specular reflections very well
- Phong shading is even better, but can be slow