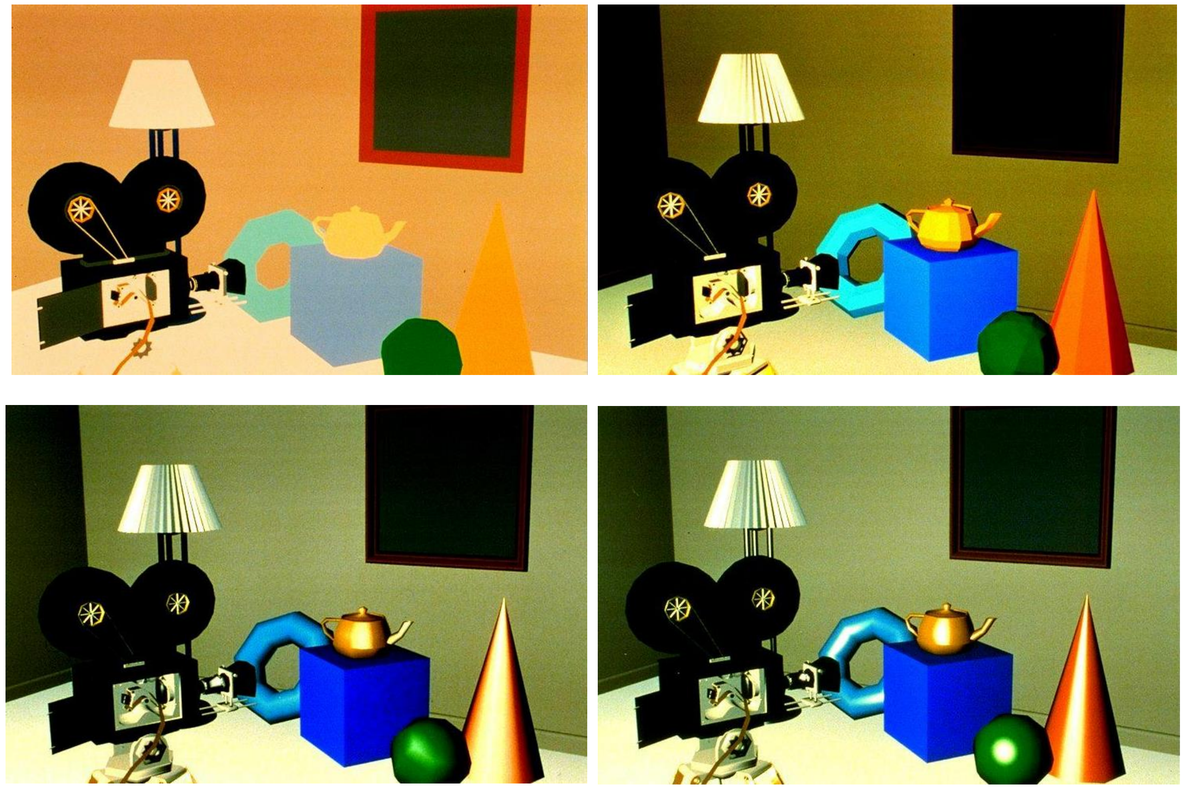
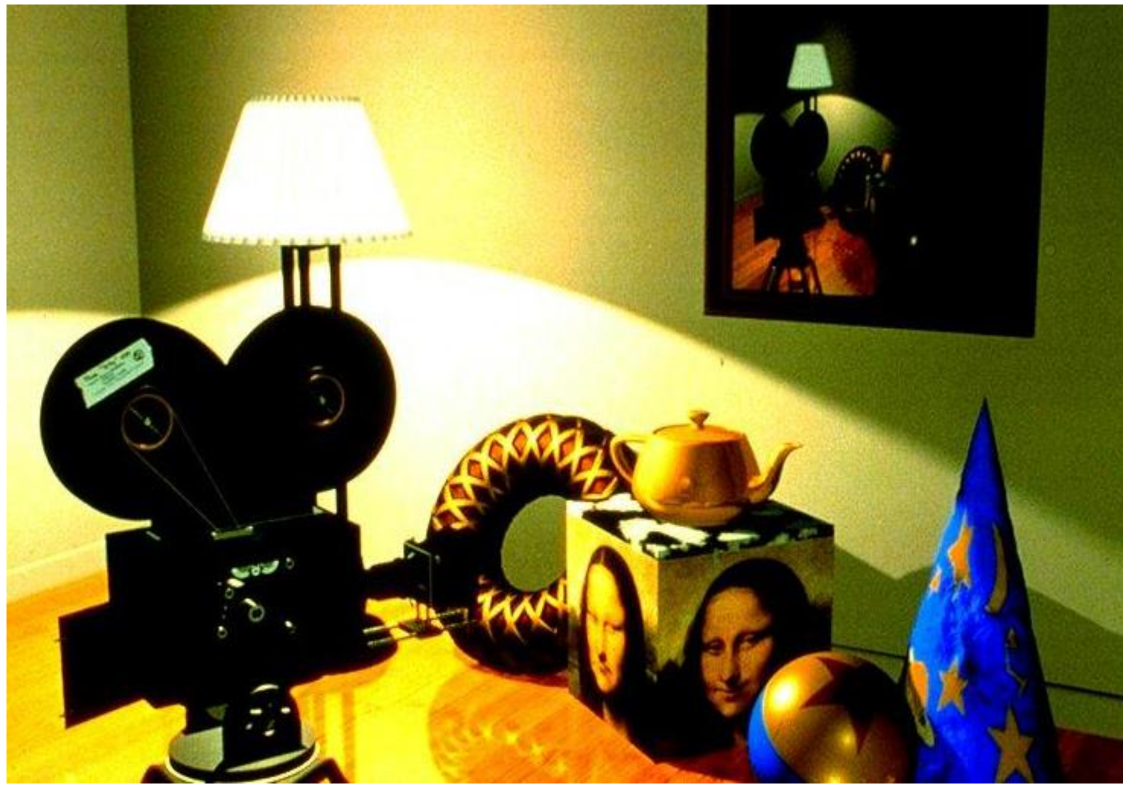



Computer Graphics

Polygon Rendering Methods

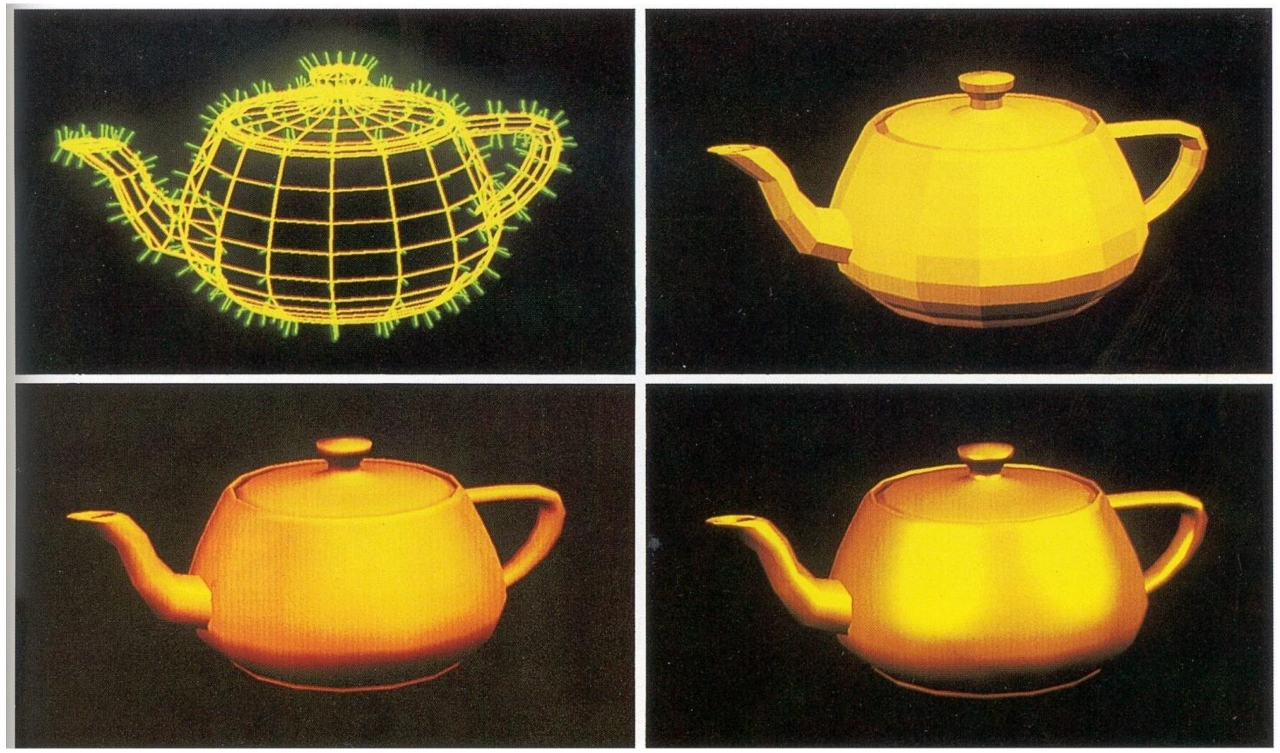
Andreas Aristidou
andarist@ucy.ac.cy
http://www.andreasaristidou.com

Polygonal Rasterization Rendering Techniques

Περιεχόμενα

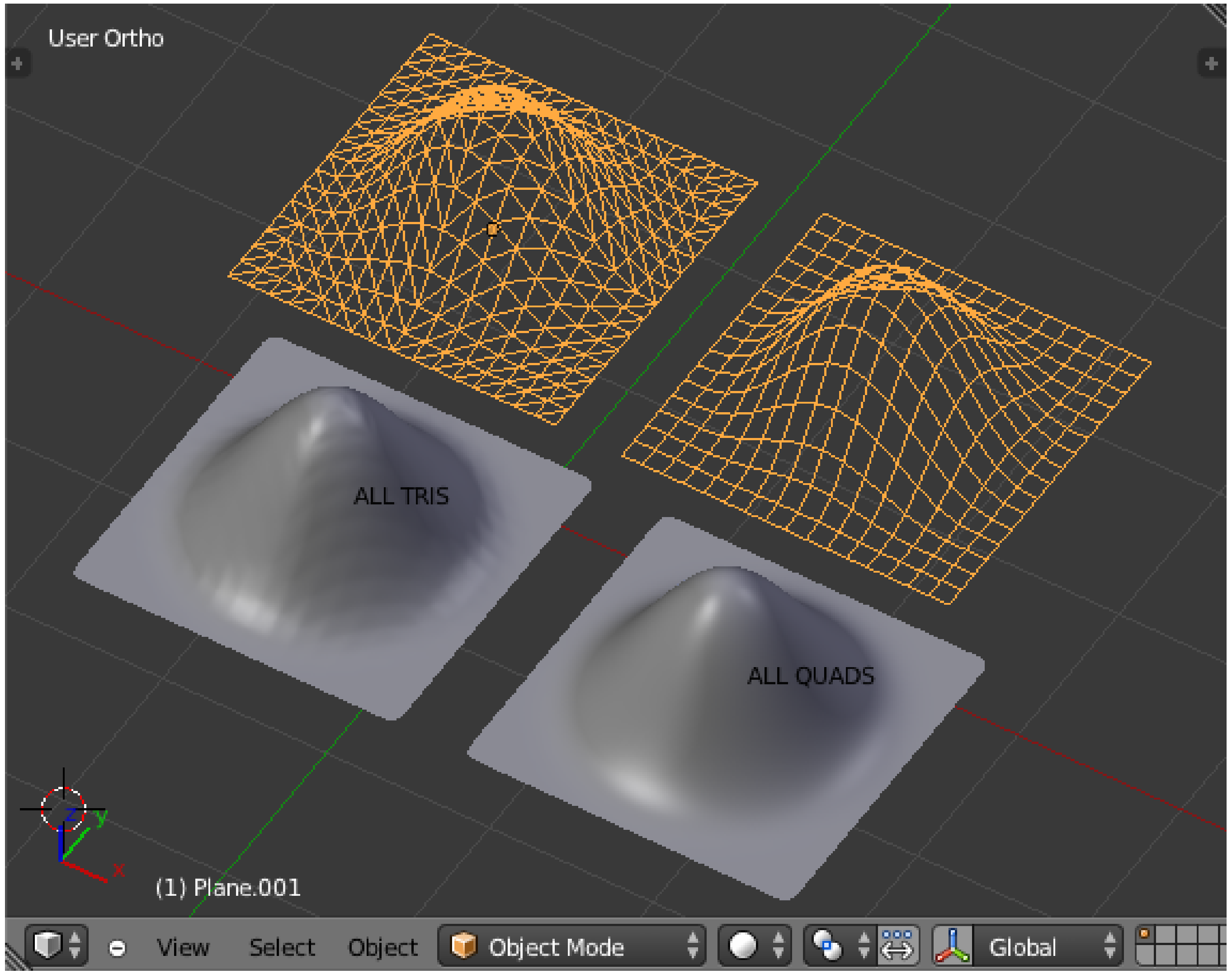
- Today we will start to look at rendering methods used in computer graphics
  - Flat surface rendering
  - Gouraud surface rendering
  - Phong surface rendering

Polygon Rendering Methods



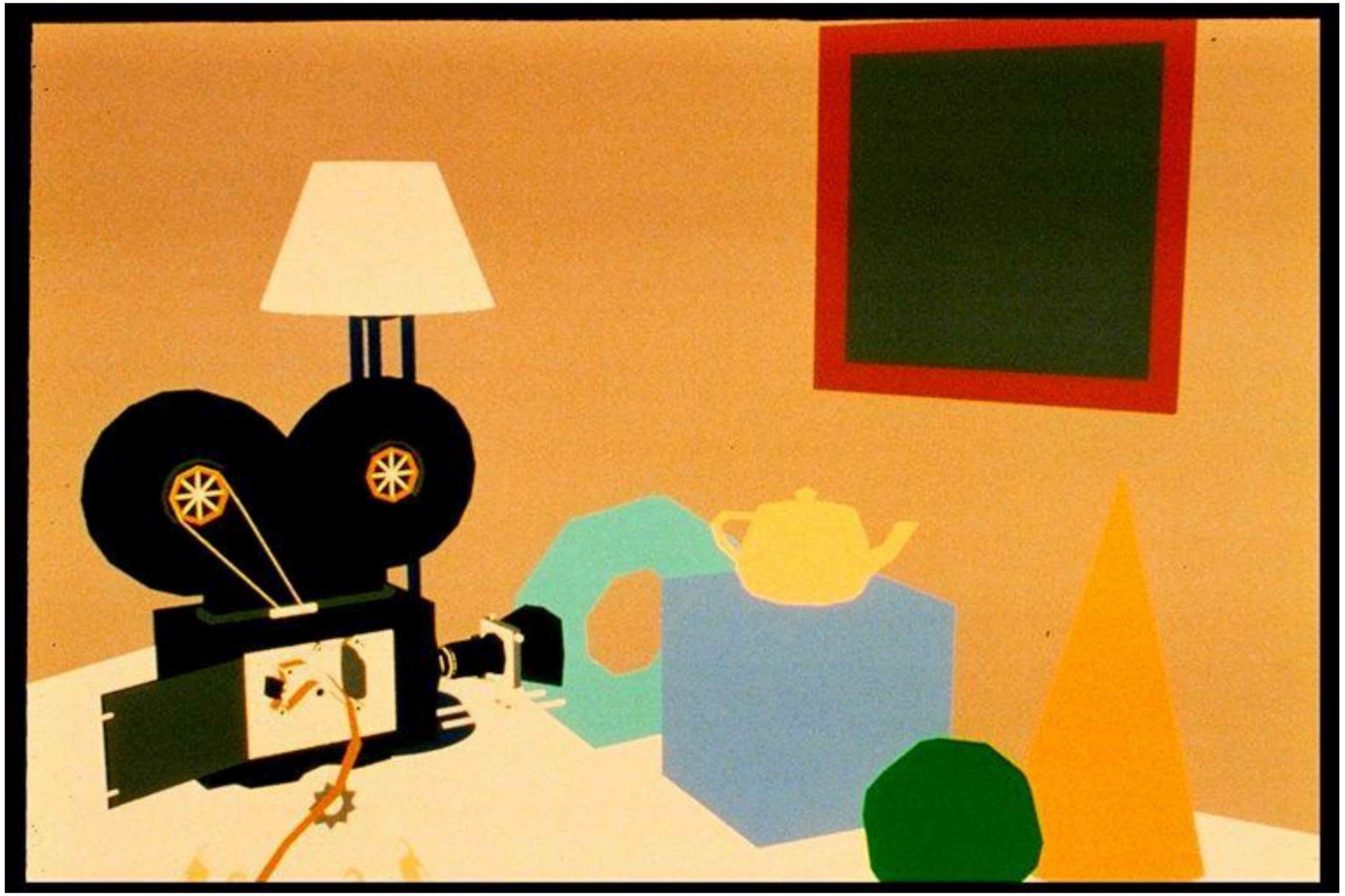
Polygon Rendering Methods

- We consider the application of a lighting model as the performance in polygons with local surfaces.
- There are two ways of rendering on surfaces:
  - Each polygon can be rendered at a constant intensity
  - The intensity at each point of the polygon is calculated by interpolation.



No Surface Rendering

Images come from: www.siggraph.org/education/materials/HyperGraph/scanline/shade_models/shading.htm

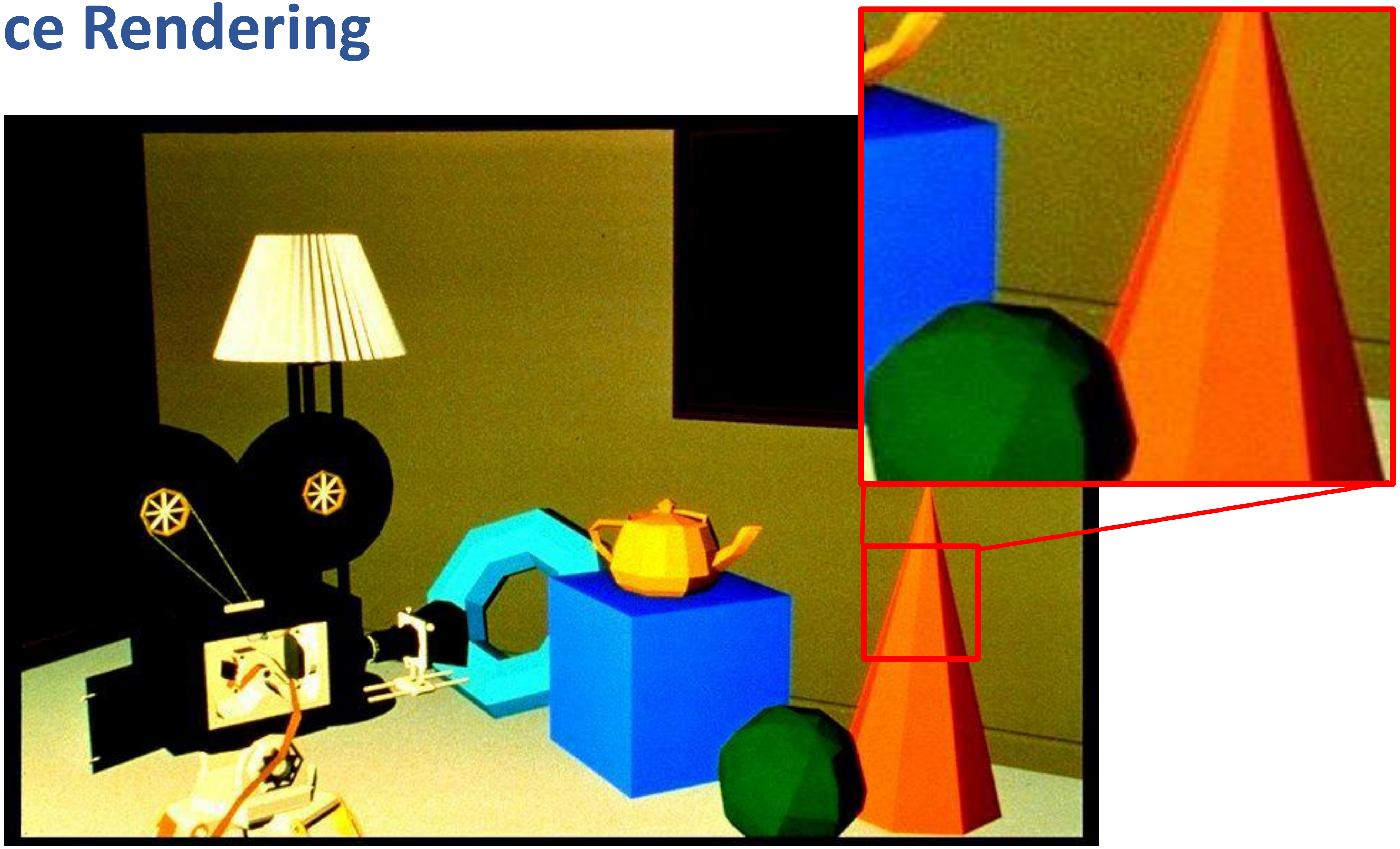


No interpolation, pick a single representative intensity and propagate it over entire object

Loses almost all depth cues

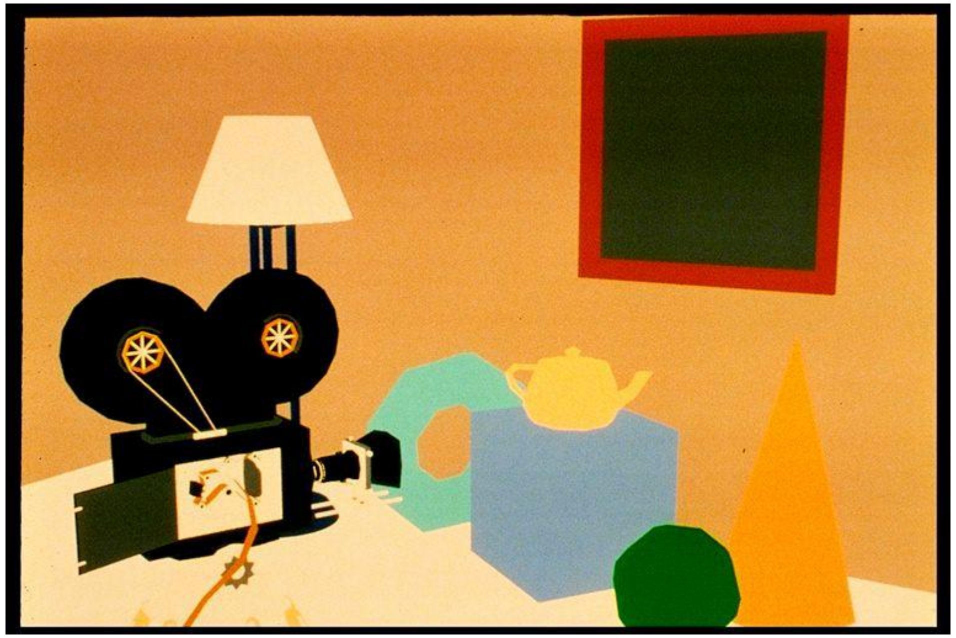
Flat Surface Rendering

Images come from: www.siggraph.org/education/materials/HyperGraph/scanline/shade_models/shading.htm

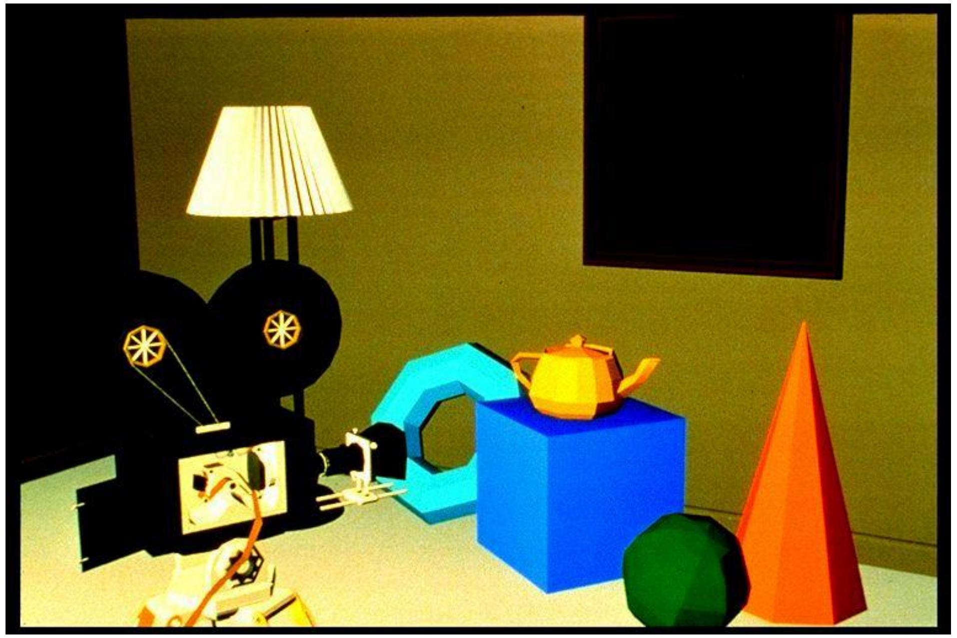


No Surface Rendering Vs Flat Surface Rendering

Images come from: www.siggraph.org/education/materials/HyperGraph/scanline/shade_models/shading.htm



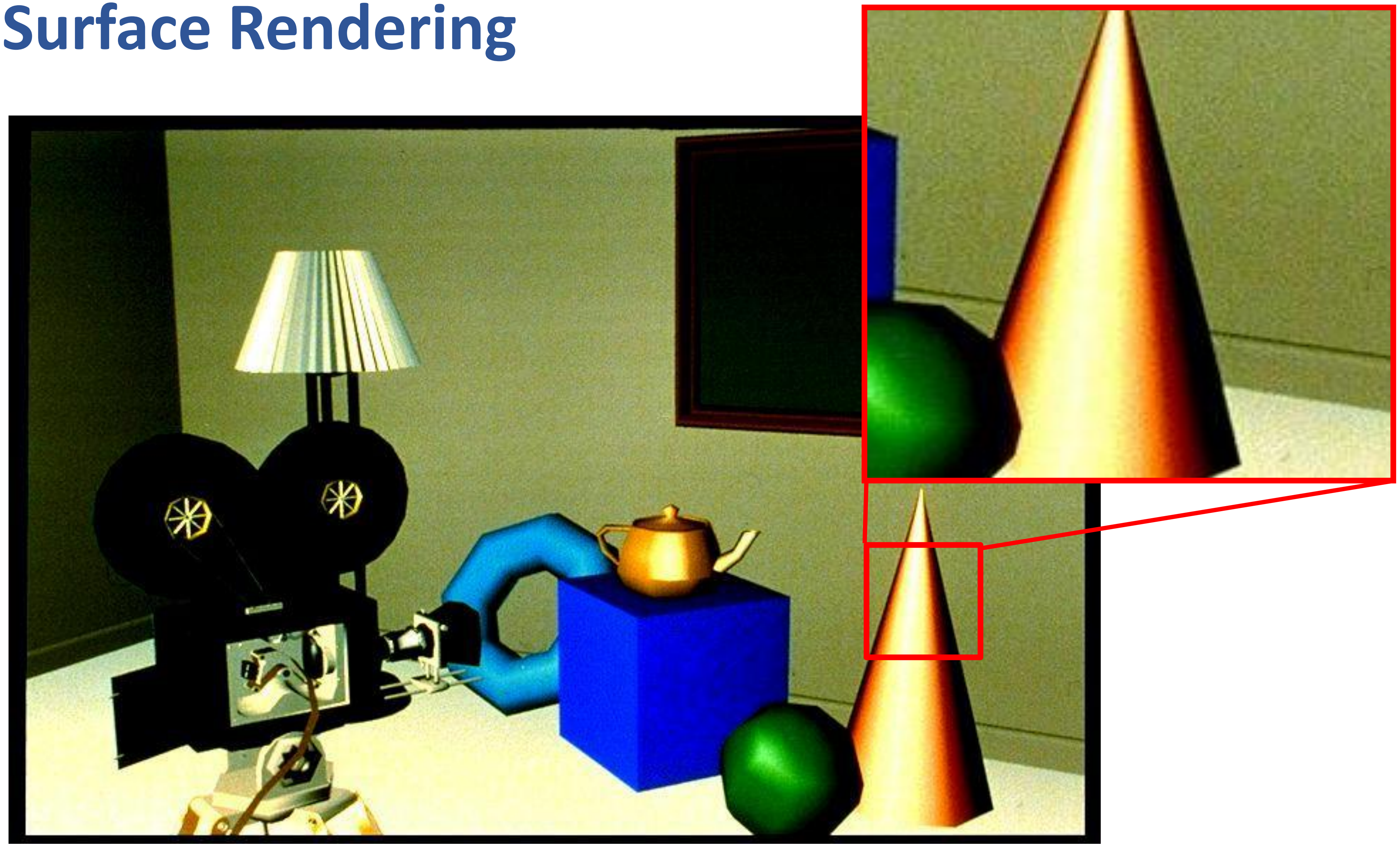
No Surface Rendering



Flat Surface Rendering

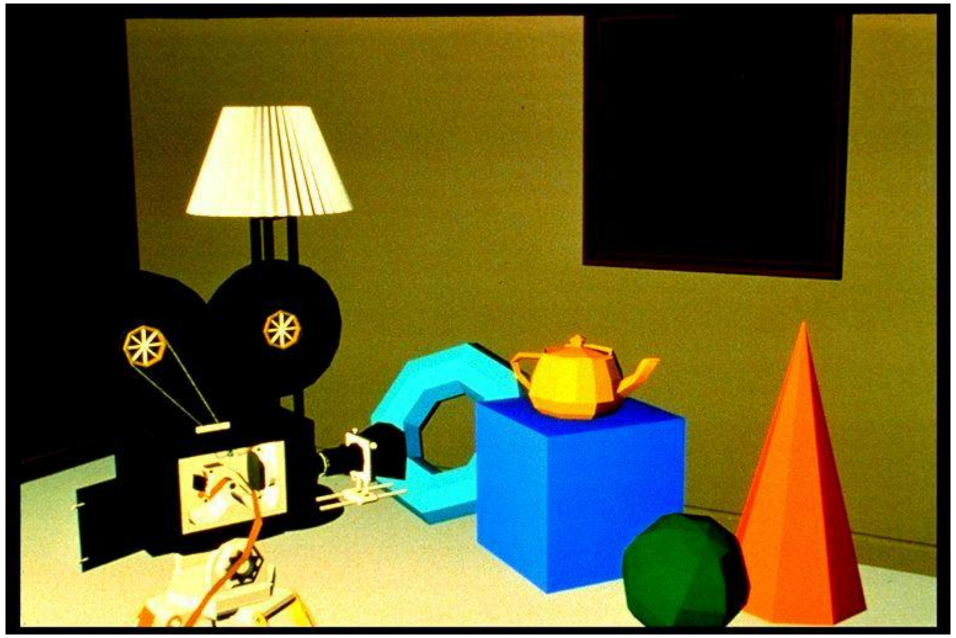
Gouraud Surface Rendering

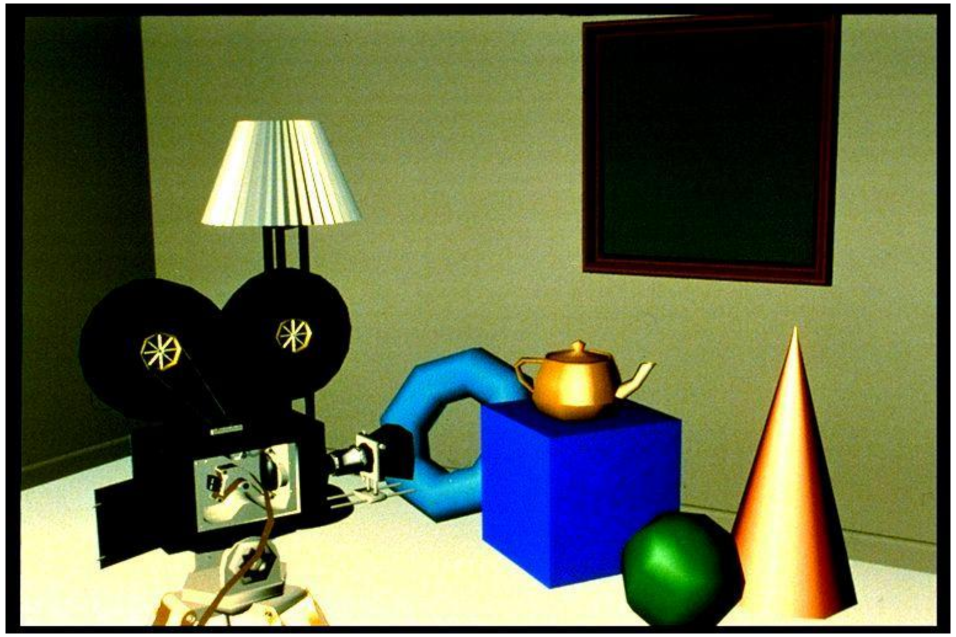
Images come from: www.siggraph.org/education/materials/HyperGraph/scanline/shade_models/shading.htm



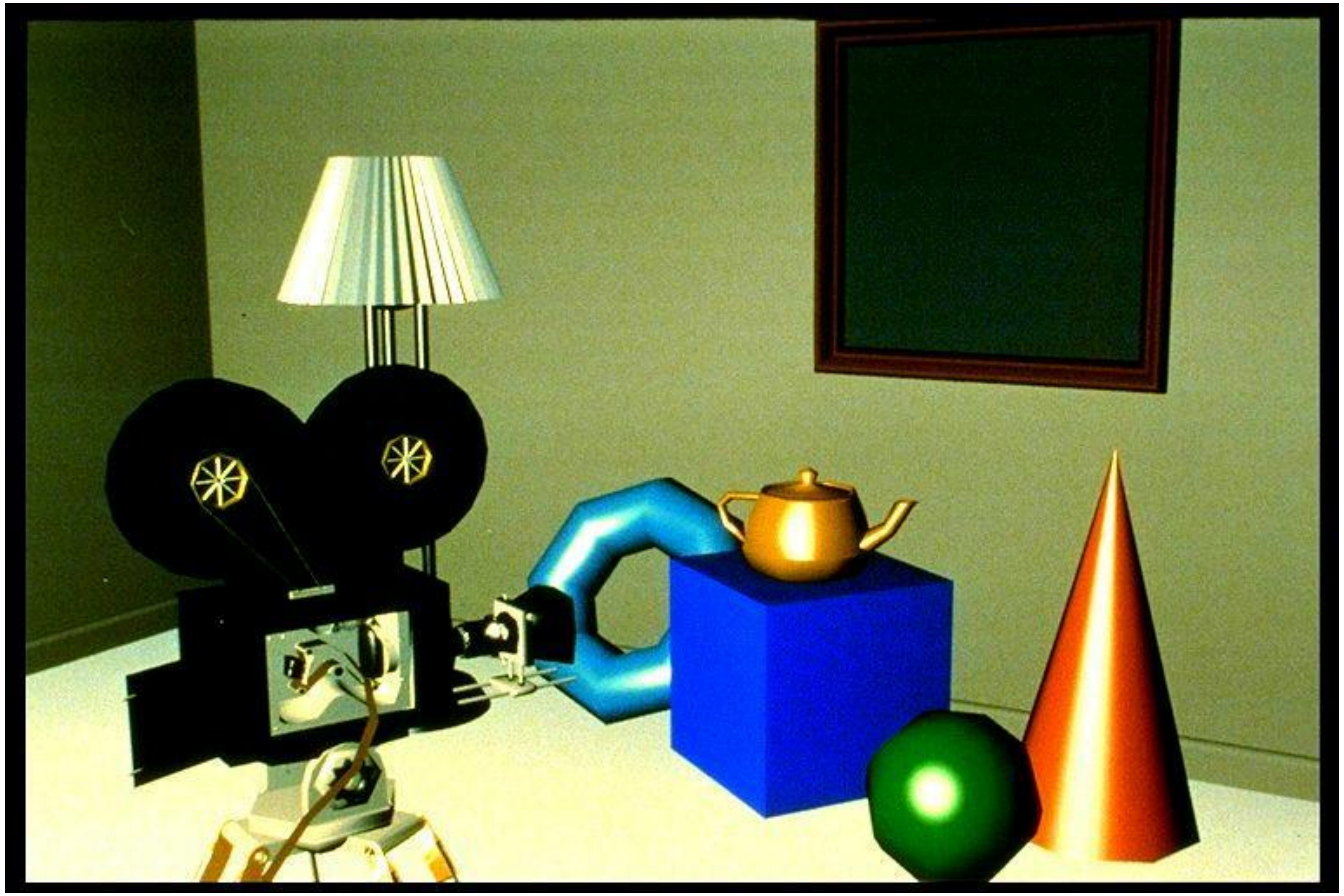
No Surface Rendering Vs Gouraud Surface Rendering

Images come from: www.siggraph.org/education/materials/HyperGraph/scanline/shade_models/shading.htm


Flat Surface Rendering


Gouraud Surface Rendering

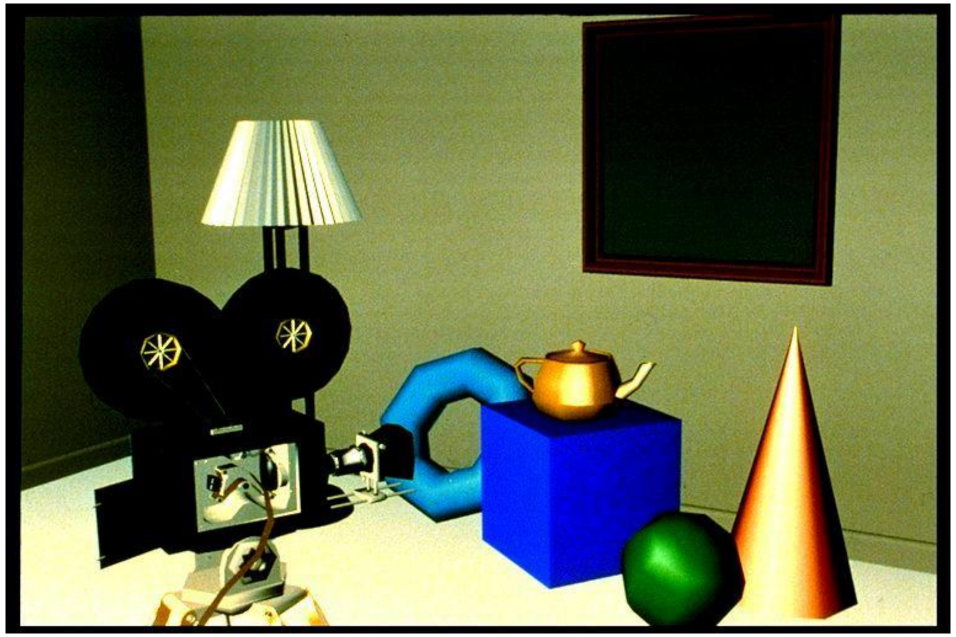
Phong Surface Rendering

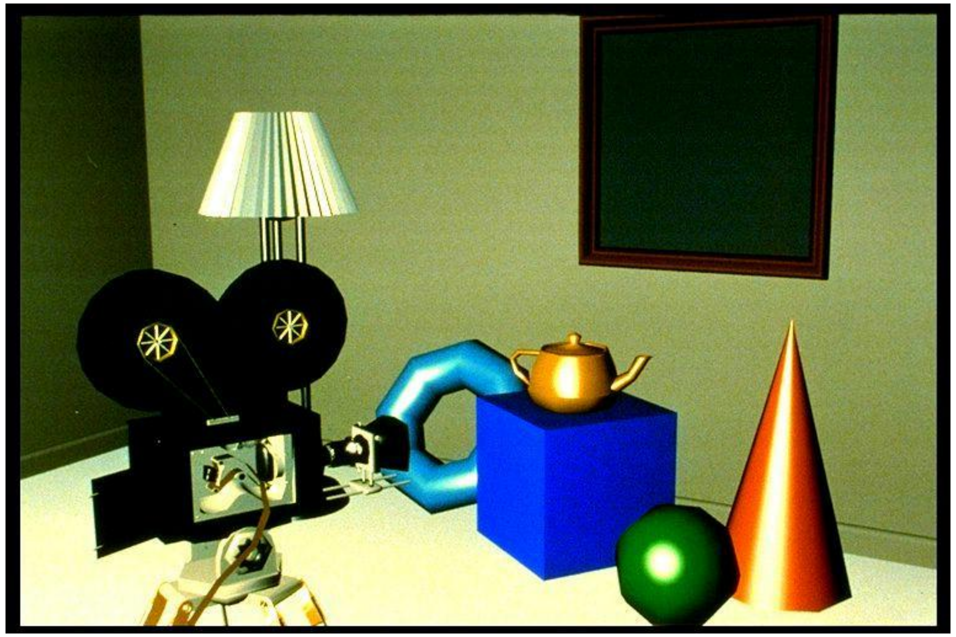


Images come from: www.siggraph.org/education/materials/HyperGraph/scanline/shade_models/shading.htm

No Surface Rendering Vs Flat Surface Rendering

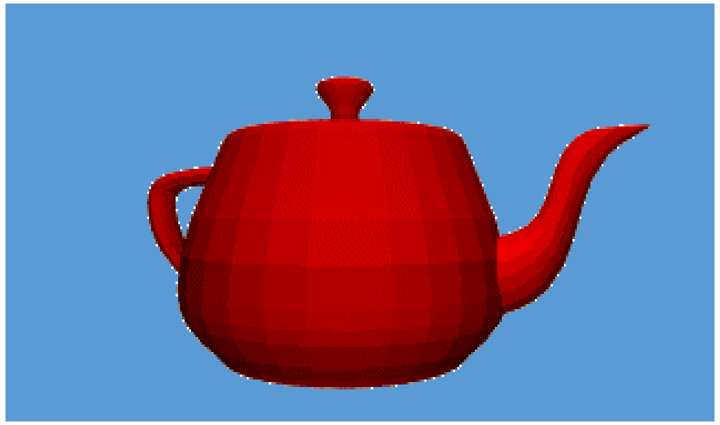
Images come from: www.siggraph.org/education/materials/HyperGraph/scanline/shade_models/shading.htm

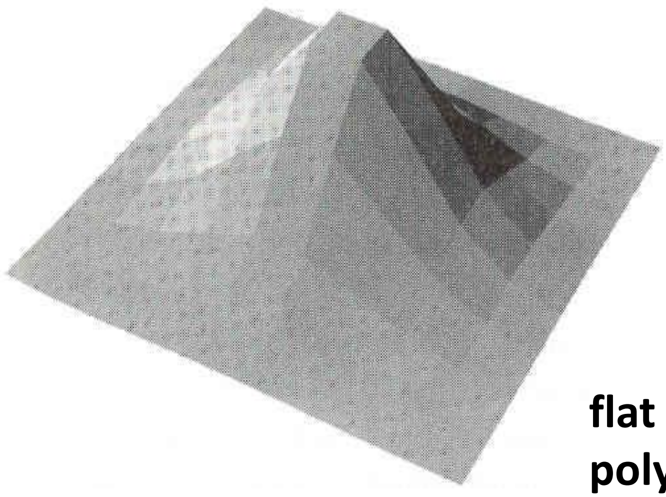

Gouraud Surface Rendering


Phong Surface Rendering

Flat Shading

- The simplest method for rendering a polygon surface. The same colour is assigned to all surface positions. The illumination at a single point on the surface is calculated and used for the entire surface.





flat shading of polygonal mesh

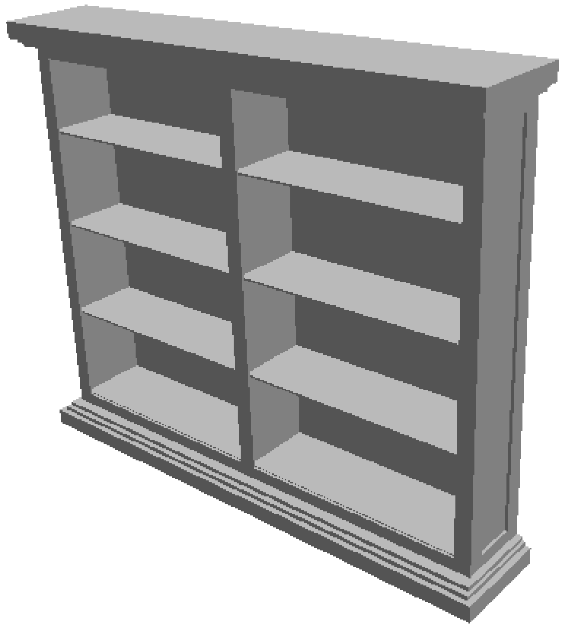
How accurate is this approach?
What are the disadvantages?

Flat surface rendering is extremely fast, but can be unrealistic

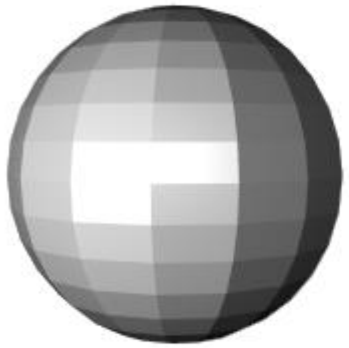
Flat Shading

- What happens when a faceted object is illuminated only by directional light sources and is either diffused or we see it from an unlimited long distance?

- Then each point on a polygon has the same lighting

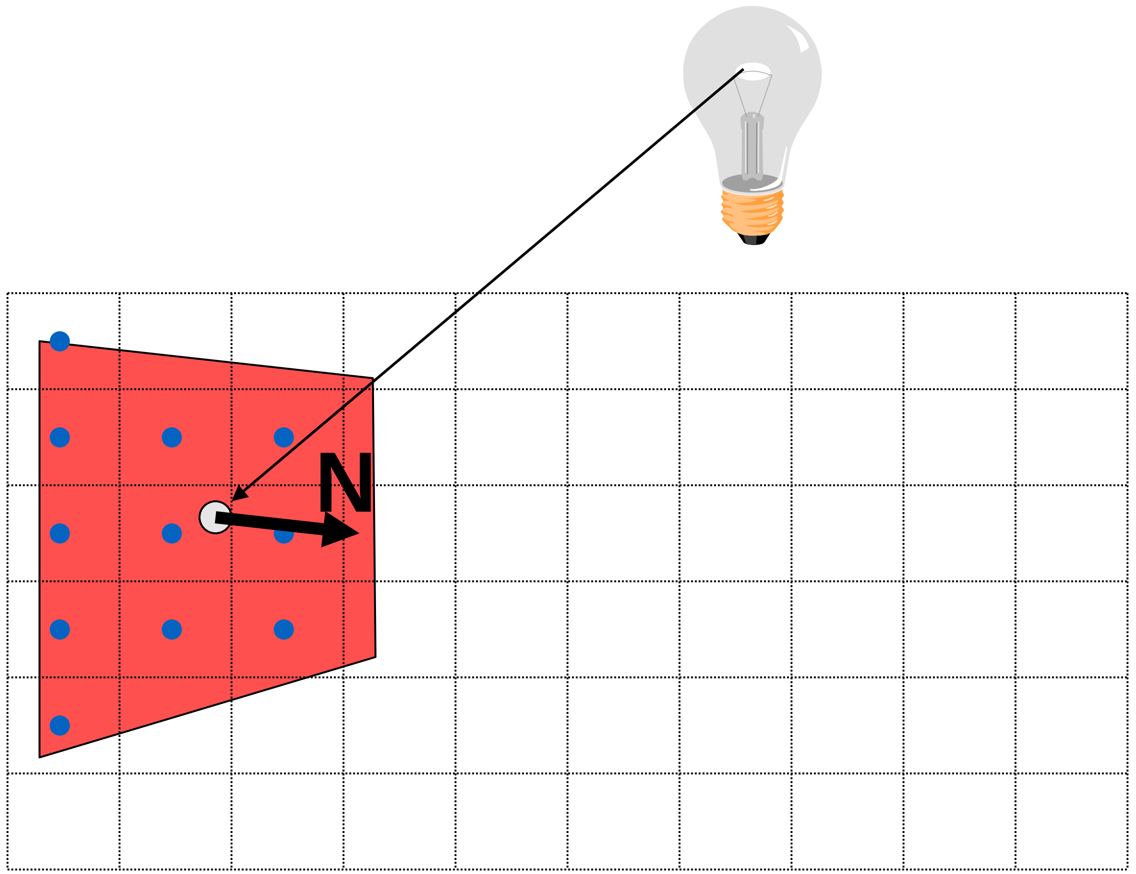


$$\boxed{I = k_a I_a + \sum_{j=1}^{p} I_j \left(k_d (n.l_j) + k_s (h_j.n)^m\right)}$$



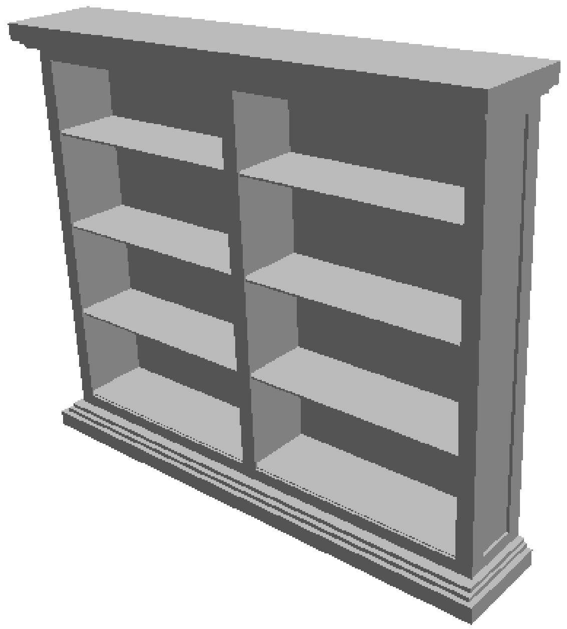
Flat Shading

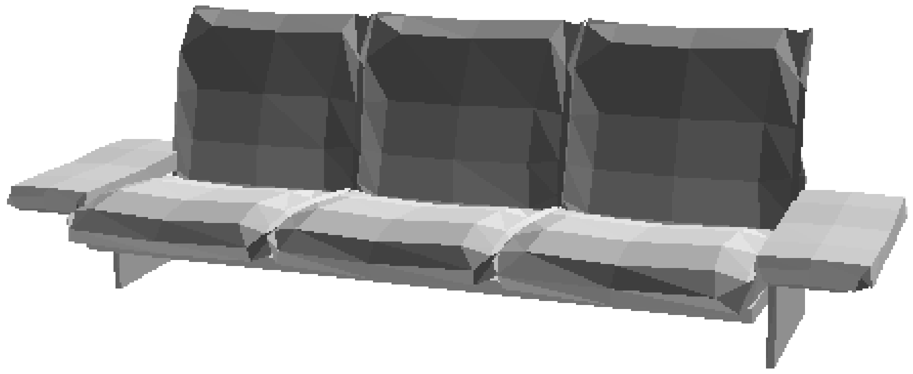
- One calculation for the entire polygon
  - Assign the same color to all pixels within the same polygon



Flat Shading

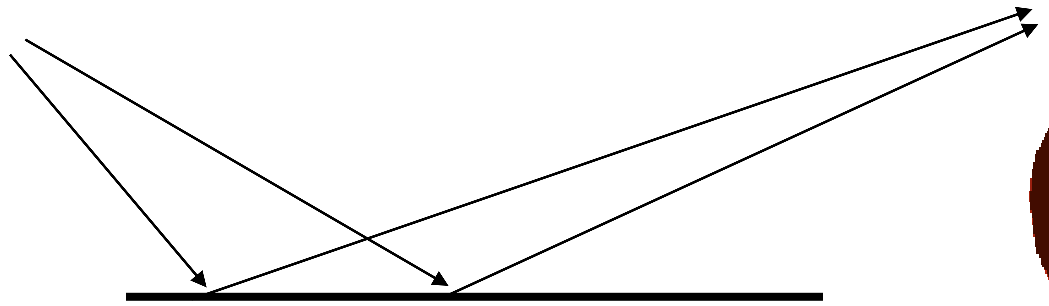
- Objects look as if they are made up of polygons
  - Ok for polyhedral objects
  - Not so good for smooth objects

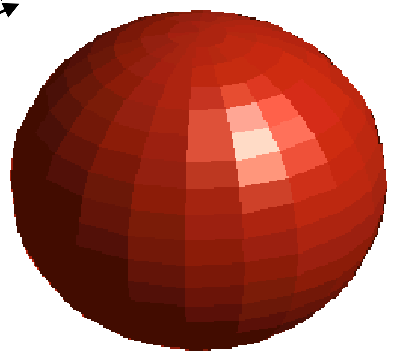




Flat Shading

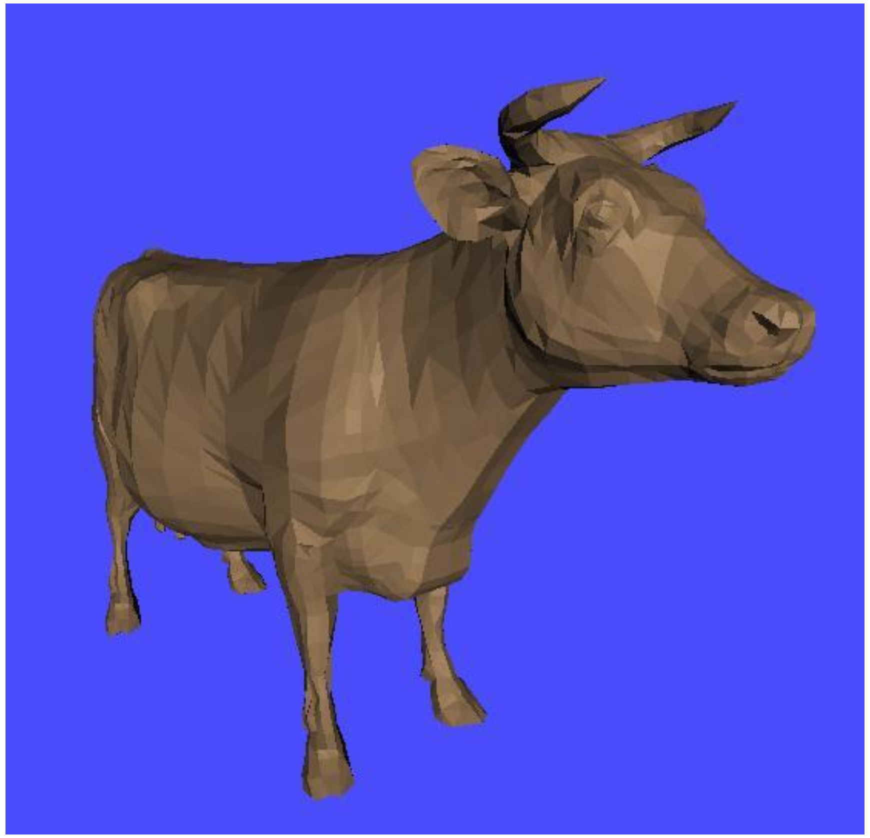
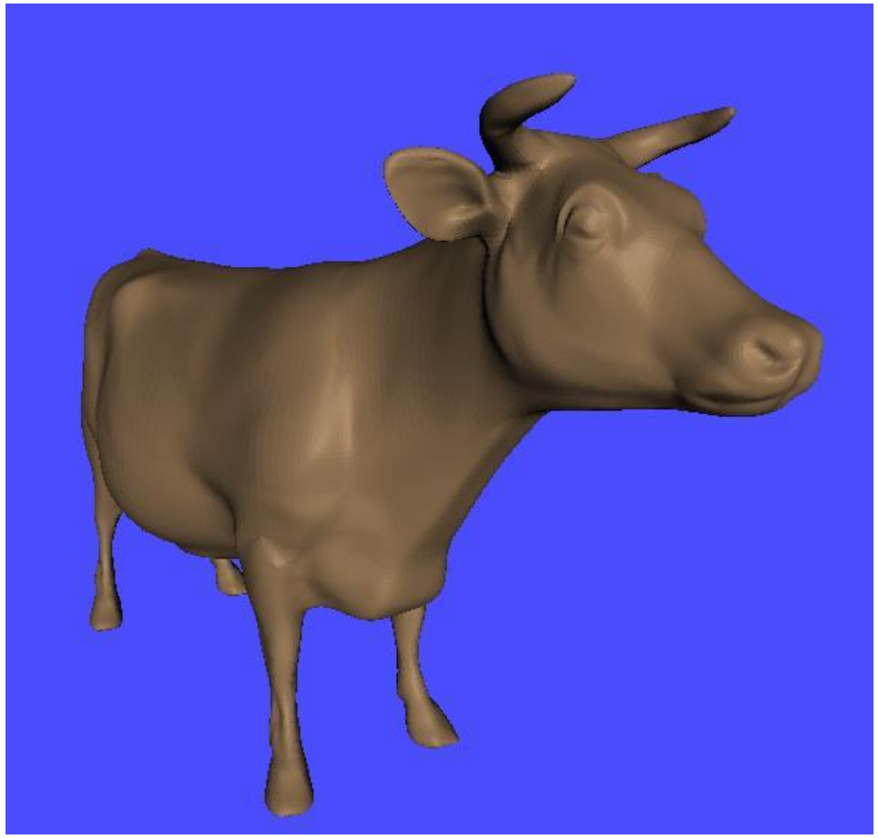
- Fast and simple method
- It gives reasonable results only if all the following assumptions apply:
  - The object is actually a polyhedron
  - The light source is very far from the surface so that N•L is stable for each polygon
  - Viewing position is very far from the surface so that the V•R is stable for each polygon





Flat Shading

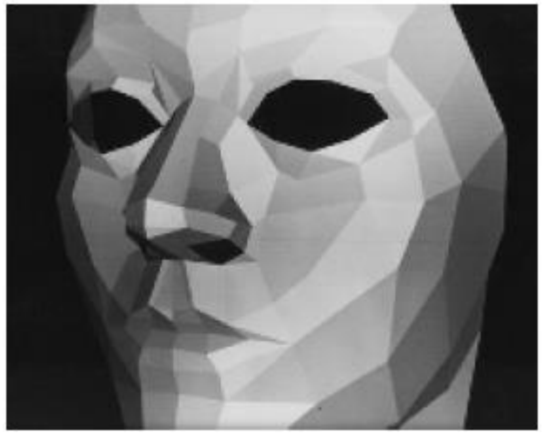
- We can Overcome Flat Shading Limitations by just adding lots and lots of polygons
  - Large files – Computational expensive

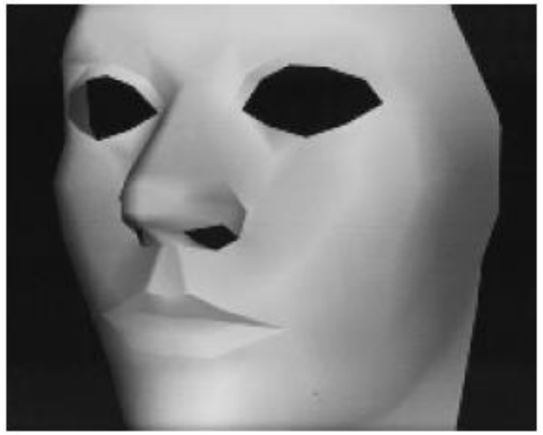



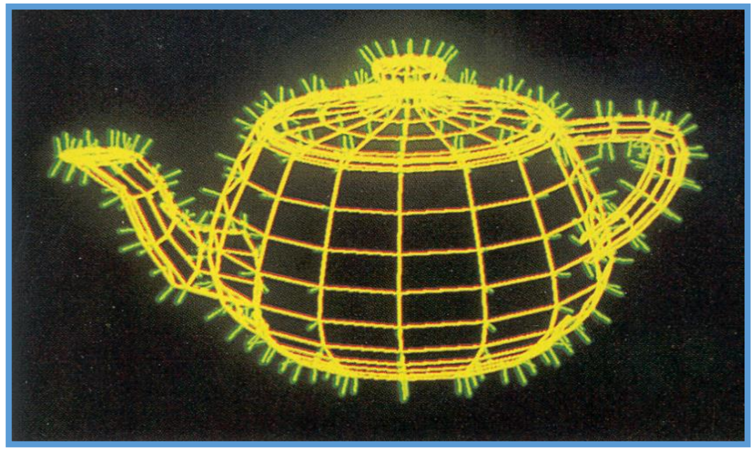
Images come from: http://www-static.cc.gatech.edu/classes/AY2004/cs4451a_spring/shading_models/linint.html

Gouraud Surface Rendering

- Gouraud surface shading was 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



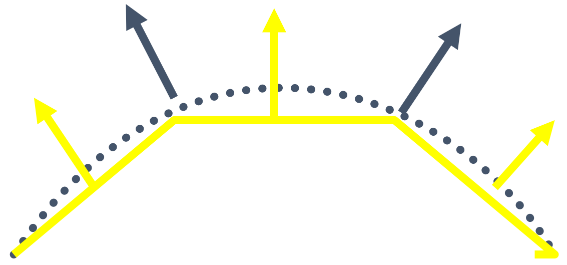




Watt Plate 7

Gouraud Surface Rendering

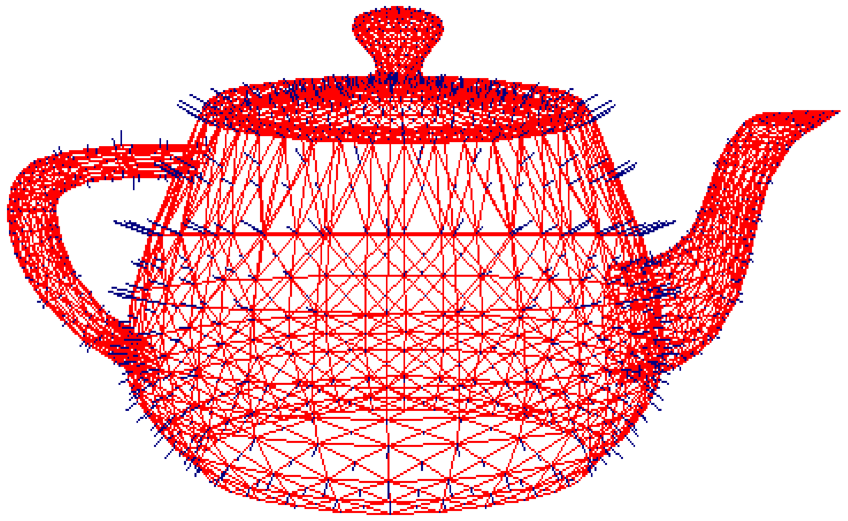
- This method calculates the intensity on a polygon surface by linearly interpolating the intensity values over the entire surface.
- To have smoother surfaces, calculate the **vertex normals** on each vertex
- They are usually different from facet normal
  - Used only for shading
  - Consider it to be a better approximation of the actual surface that polygons approach



Grauraud Shading

Vertex Normals:

- Vertex normals can
  - Given by default by the model
  - Calculated from the model
  - Be approached with the average of the normal of the faces that share the same top

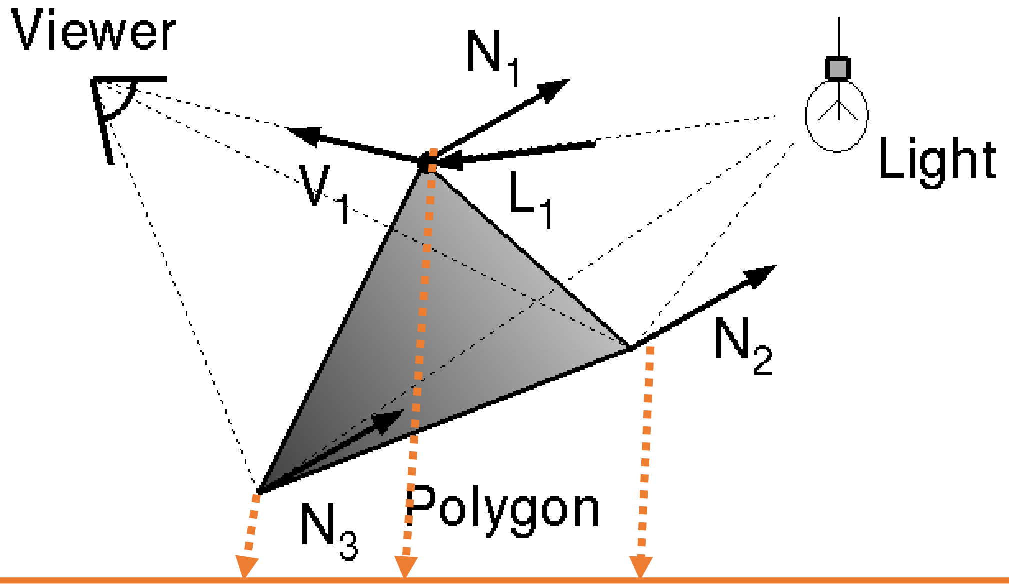


Gouraud Surface Rendering

- 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

- One lighting calculation per edge
  - Assign the pixels within the polygon by interpolation of the colors calculated on the vertices



$$I = k_a I_a + \sum_{j=1}^{p} I_j \left(k_d (n.l_j) + k_s (h_j.n)^m\right)$$

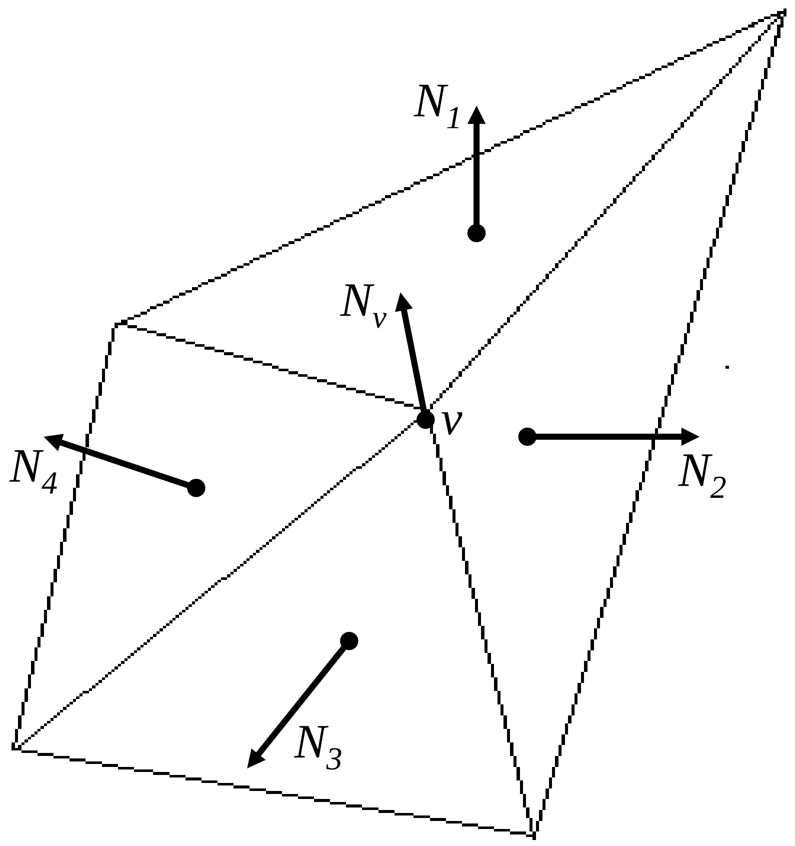
Gouraud Surface Rendering

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

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

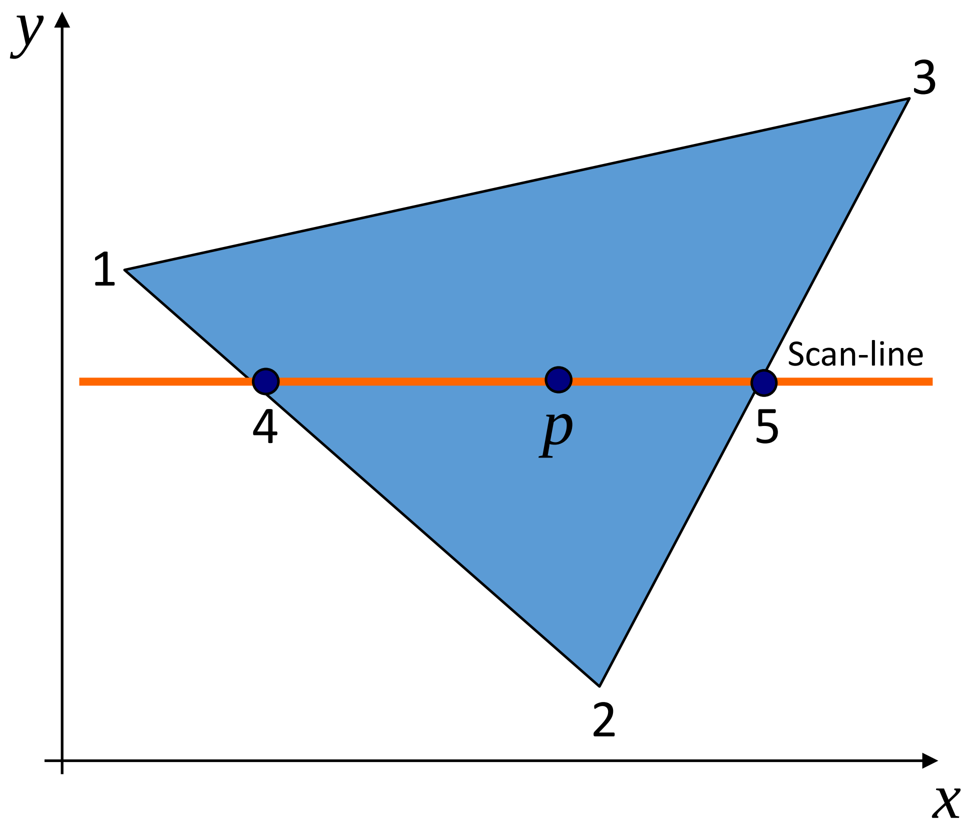
- or more generally :

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



Gouraud Surface Rendering

- Illumination values are linearly interpolated across each scan-line

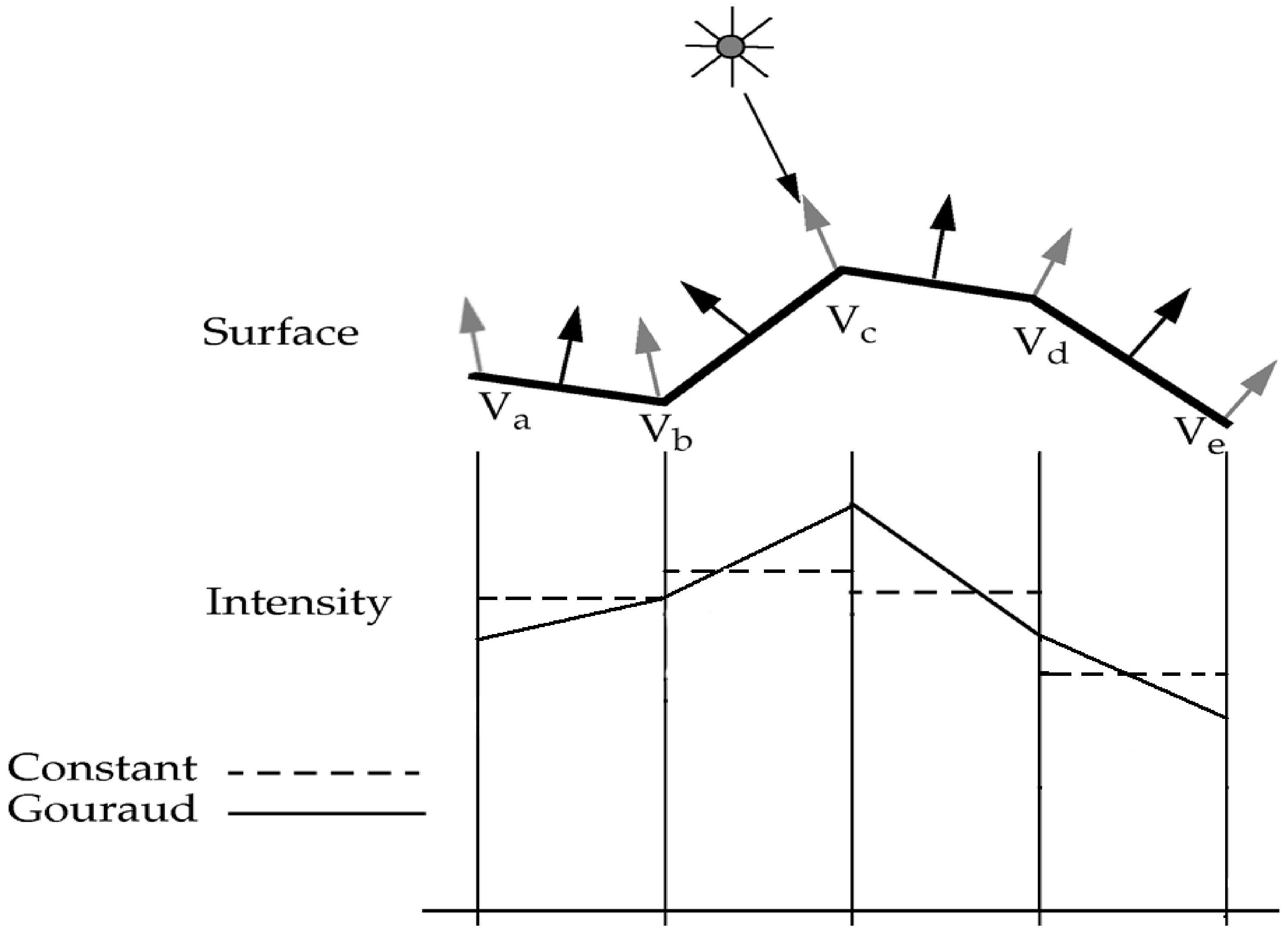


$$I_4 = \frac{y_4 - y_2}{y_1 - y_2} I_1 + \frac{y_1 - y_4}{y_1 - y_2} I_2$$

$$I_5 = \frac{y_5 - y_2}{y_3 - y_2} I_3 + \frac{y_3 - y_5}{y_3 - y_2} I_2$$

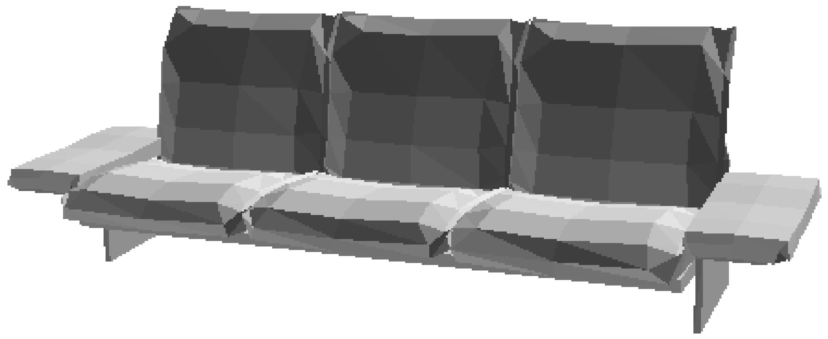
$$I_p = \frac{x_5 - x_p}{x_5 - x_4} I_4 + \frac{x_p - x_4}{x_5 - x_4} I_5$$

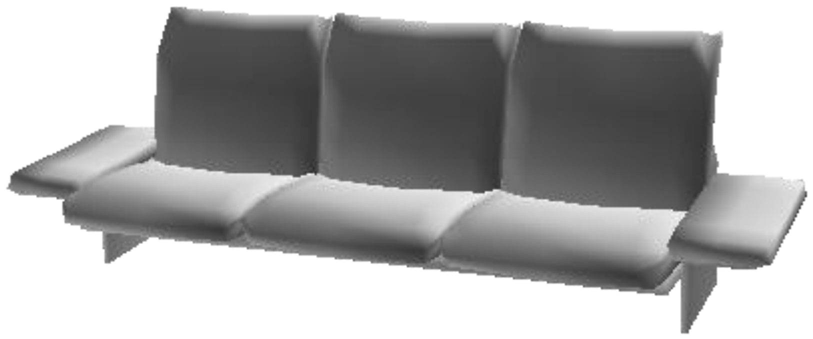
Advantages of Gouraud Surface Rendering



Gouraud Shading

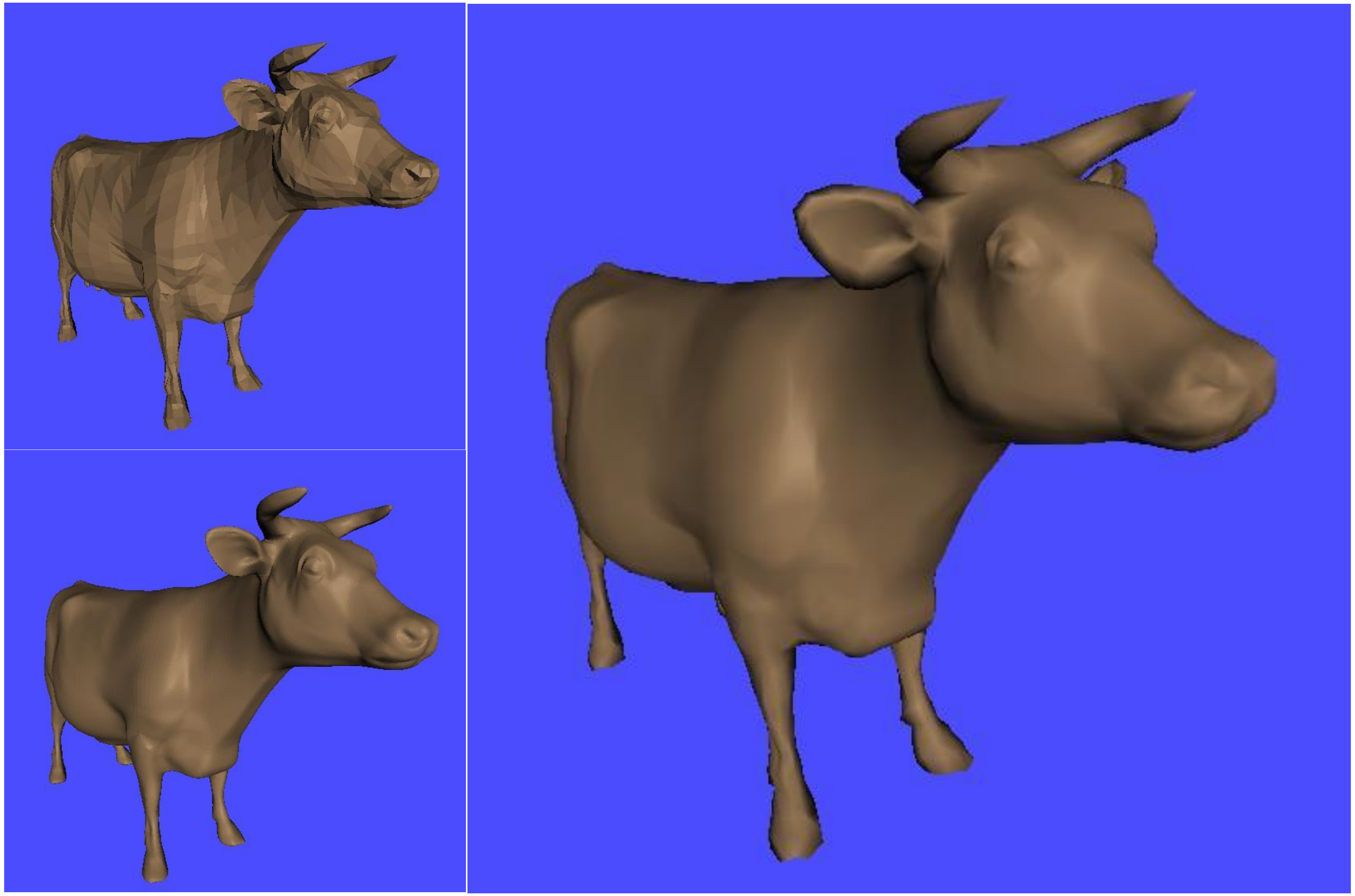
- Produces smooth color-graded (shaded) polygonal meshes
- Piecewise linear approach
  - Need fine mesh to capture subtle lighting effects


Flat Shading


Gouraud Shading

Gouraud Surface Rendering: *Example*

Images come from: http://www-static.cc.gatech.edu/classes/AY2004/cs4451a_spring/shading_models/linint.html

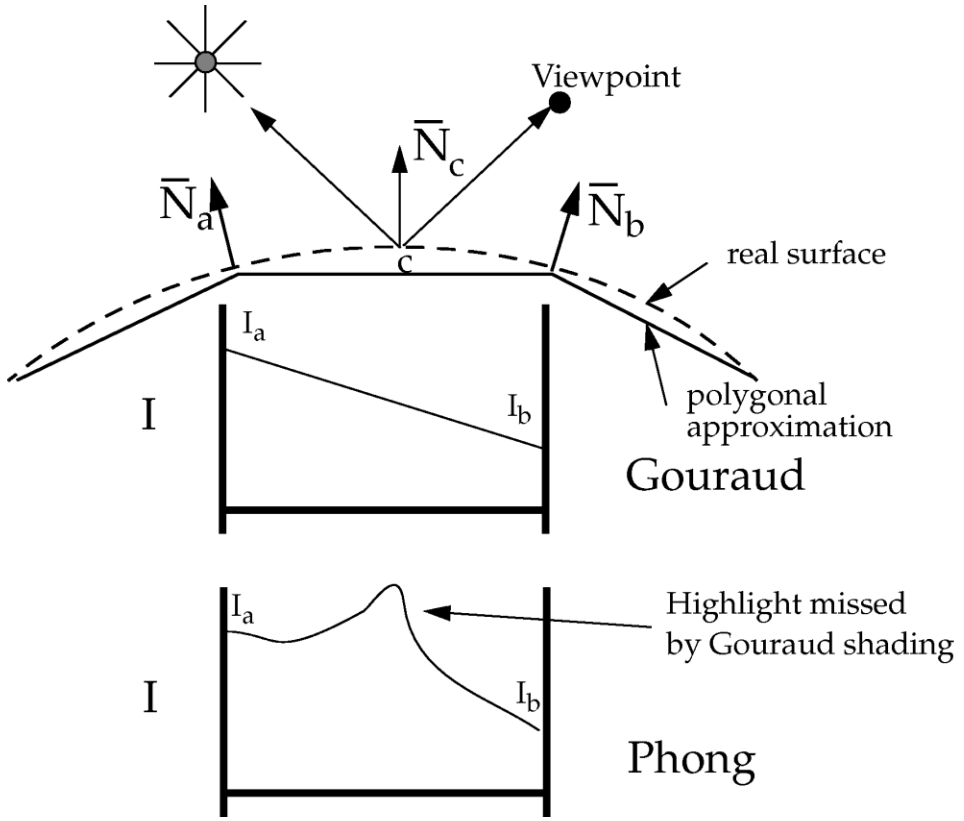


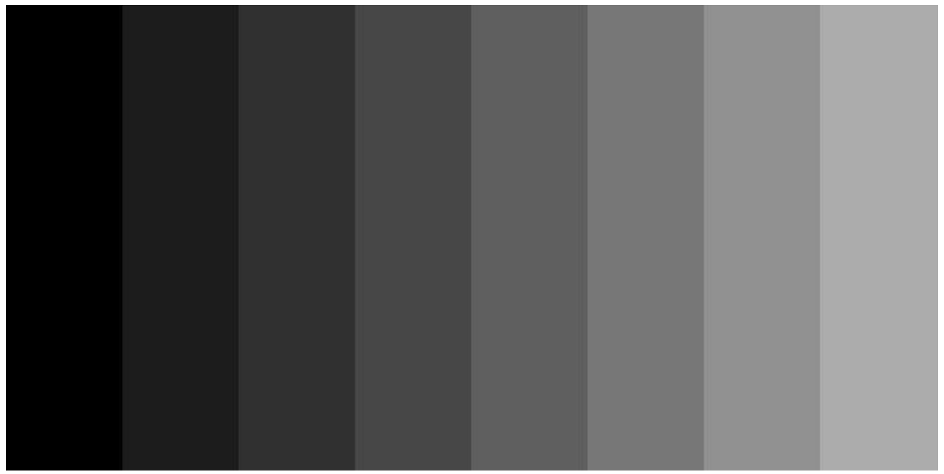
Gouraud Surface Rendering Implementation

- Gouraud surfacing rendering can be implemented relatively efficiently using an iterative approach
- Typically Grouaud shading is implemented as part of a visible surface detection technique

Problems with Gouraud Shading

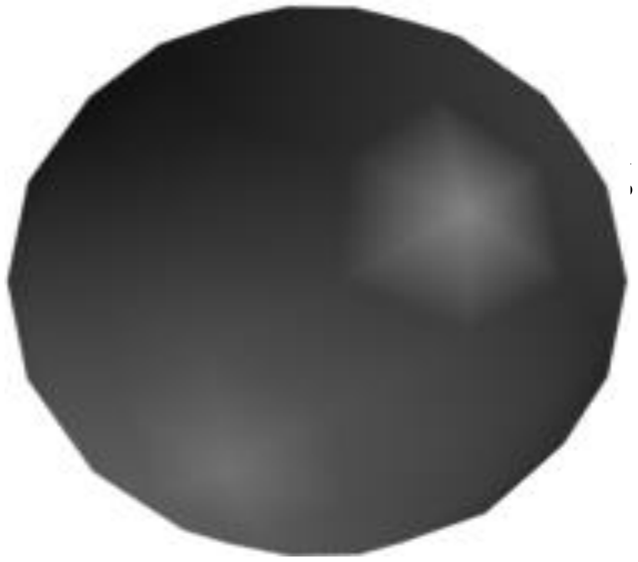
- Gouraud shading tends to miss certain highlighting
  - In particular Gouraud shading has a problem with specular reflections
- Also, Gouraud shading can introduce anomalies known as **Mach bands**

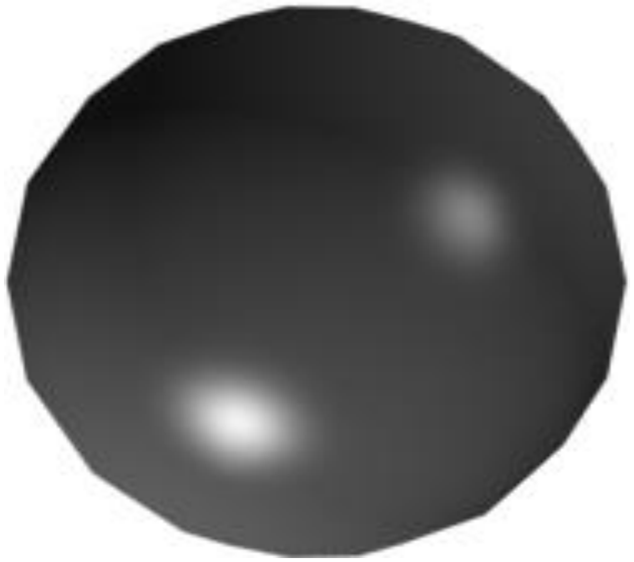



Mach bands

Problems with Gouraud Shading

- The major problem with Gouraud shading is in handling specular reflections


Gouraud


Phong

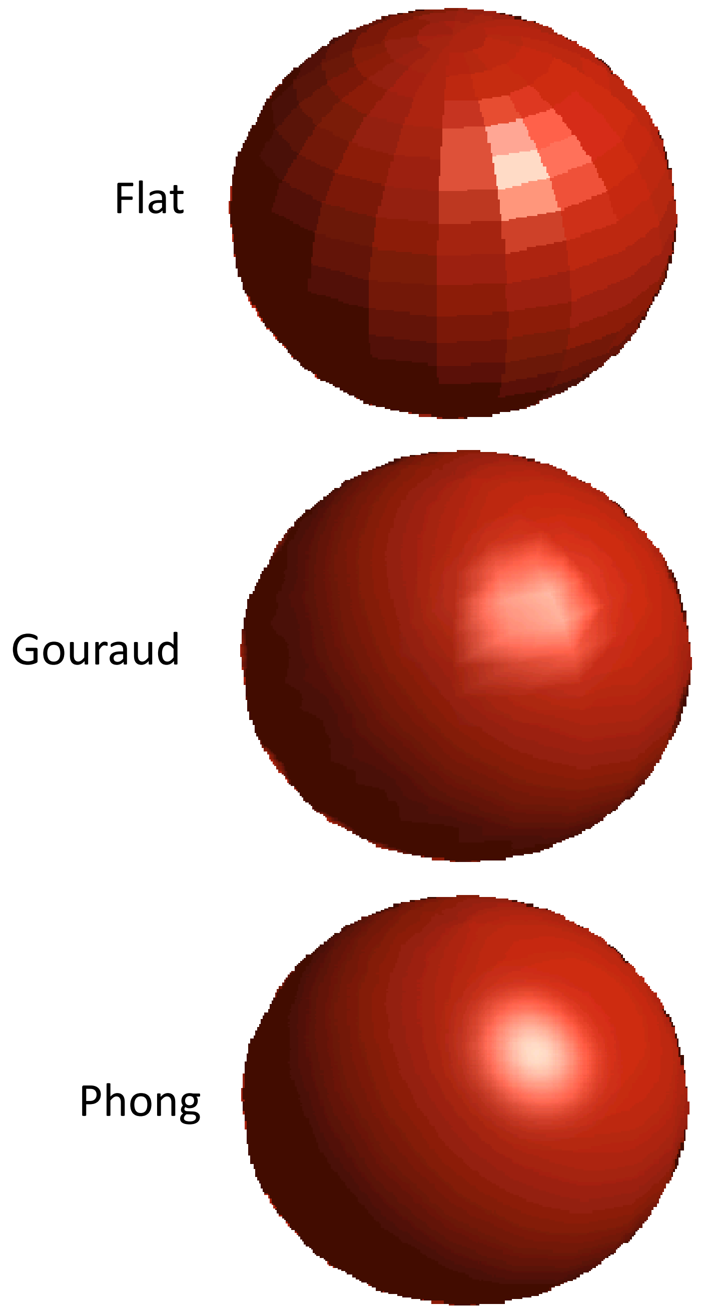
Gouraud Shading*: Summary*

- Determination of the mean perpendicular unit vector at each polygon vertex
- Application of the lighting model at each vertex to calculate the intensity.

- **Linear interpolation** of the intensities of the vertices on the surface of the polygon.

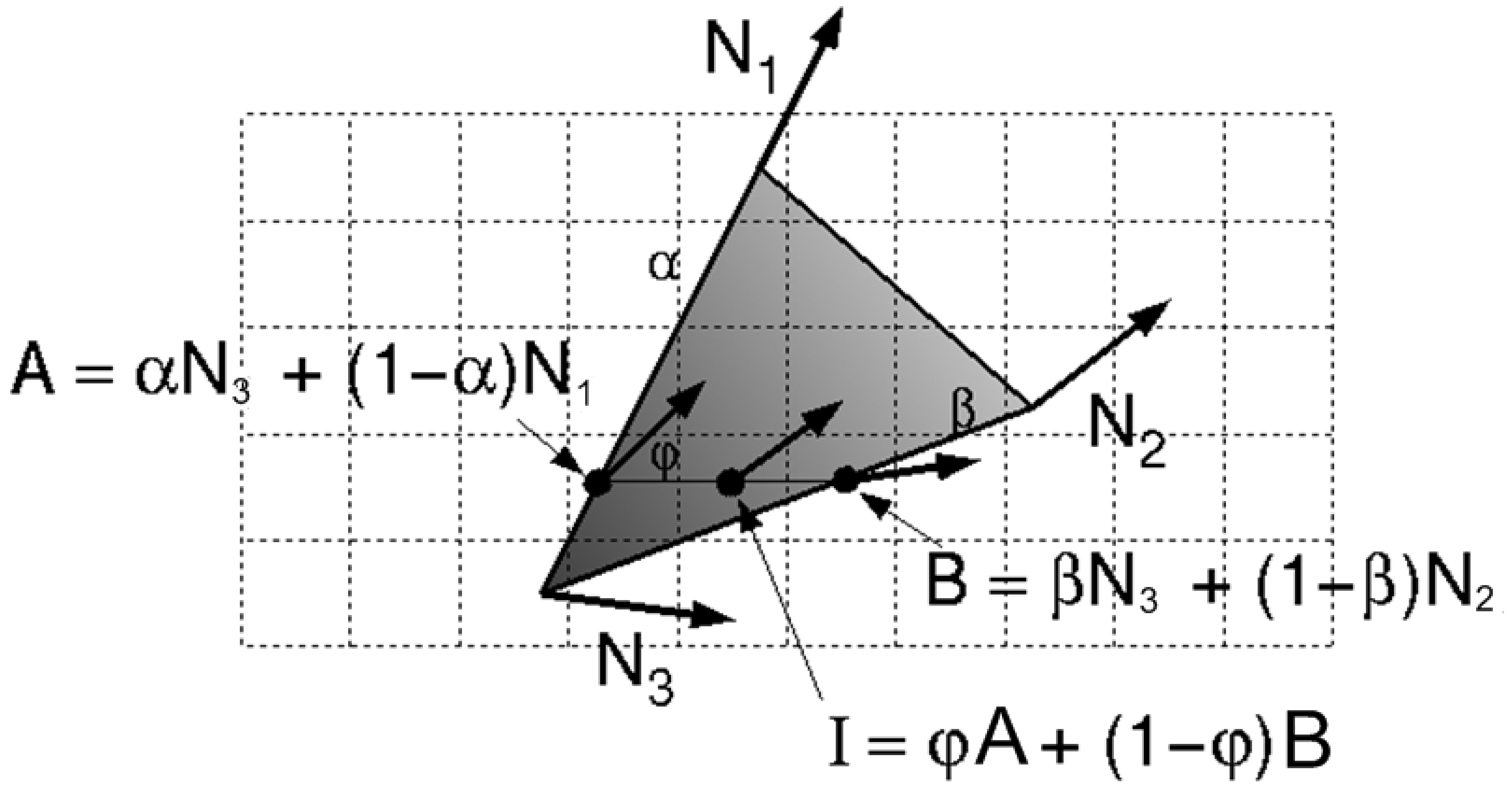
Phong Surface Rendering

- 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
  - The representation of the surface of the polygons is done by interpolating the perpendiculars (normals), and applying the illumination model at every point of the surface.
- However, much more expensive model



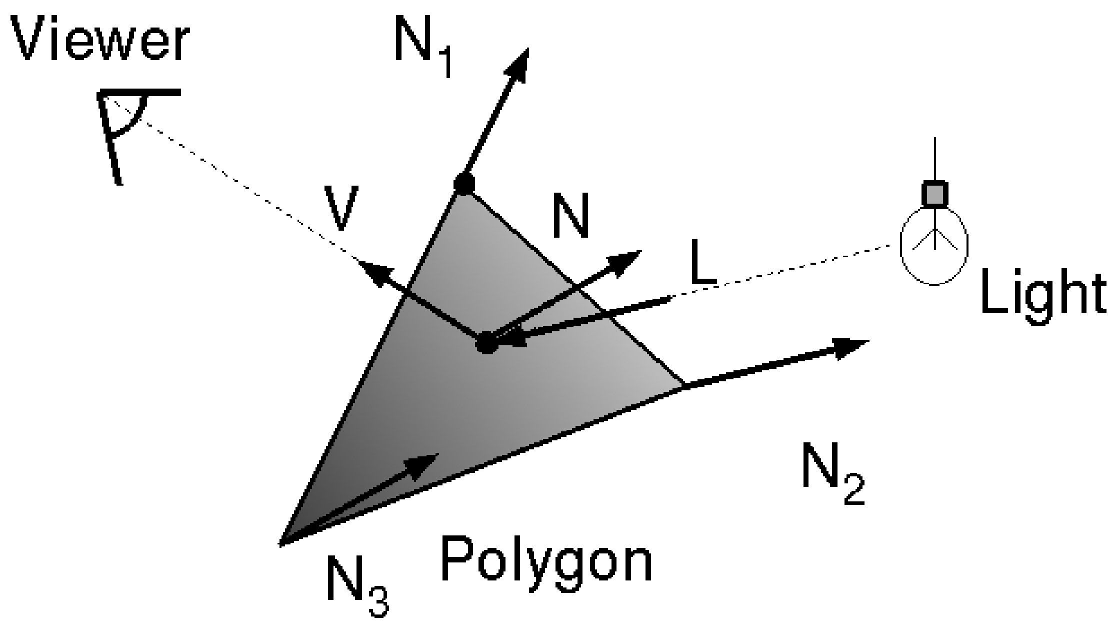
Phong Shading

- We calculate the perpendicular at each point with bilinear interpolation of the perpendiculars on the surface vertexes, along the scanning lines


N_1
α
$A = \alpha N_3 + (1-\alpha)N_1$
φ
β
N_2
$B = \beta N_3 + (1-\beta)N_2$
N_3
$I = \varphi A + (1-\varphi)B$

Phong Shading

- Calculate the illumination for each pixel

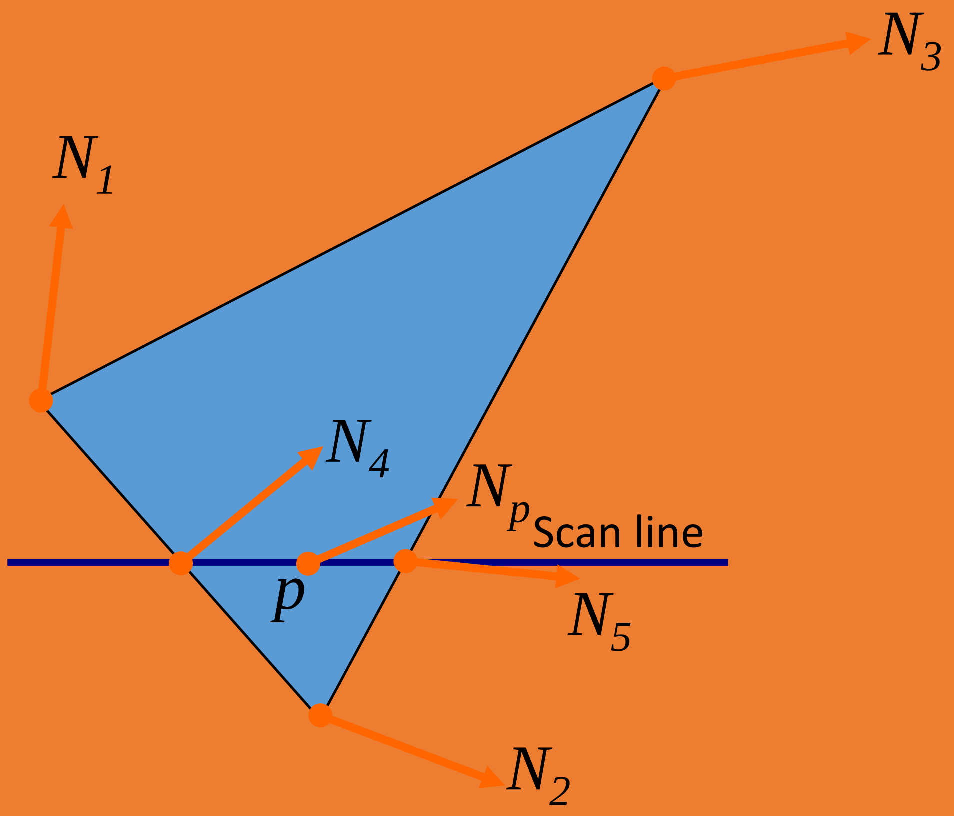


$$I = k_a I_a + \sum_{j=1}^{p} I_j \left(k_d (n.l_j) + k_s (h_j.n)^m \right)$$

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
    3. Apply an illumination model at positions along scan lines to calculate pixel intensities using the interpolated normal vectors

Phong Surface Rendering

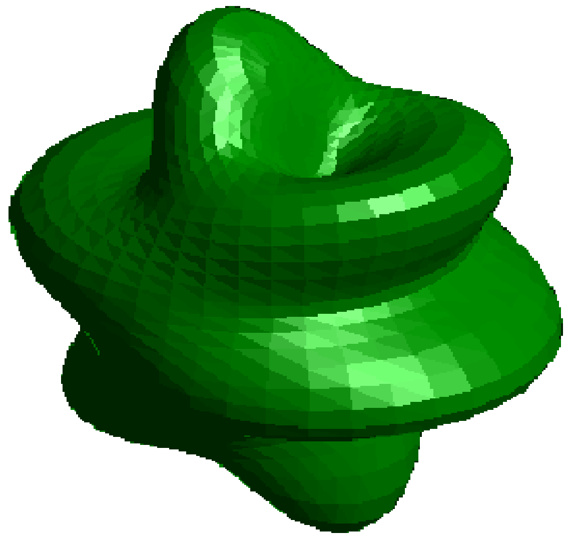


$$N_4 = \frac{y_4 - y_2}{y_1 - y_2} N_1 + \frac{y_1 - y_4}{y_1 - y_2} N_2$$

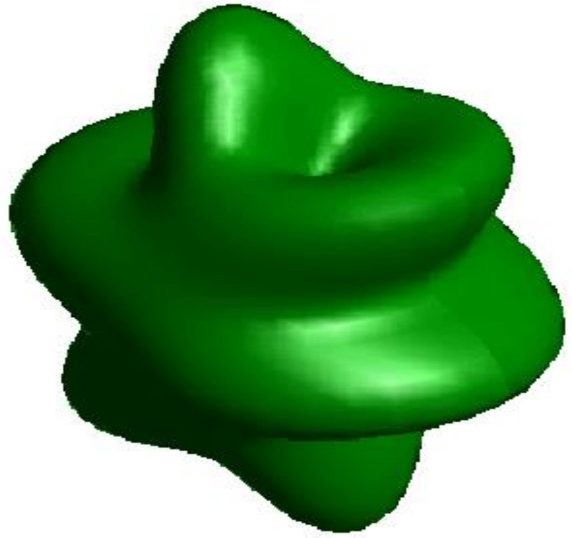
$$N_5 = \frac{y_5 - y_2}{y_3 - y_2} N_3 + \frac{y_3 - y_5}{y_3 - y_2} N_2$$

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

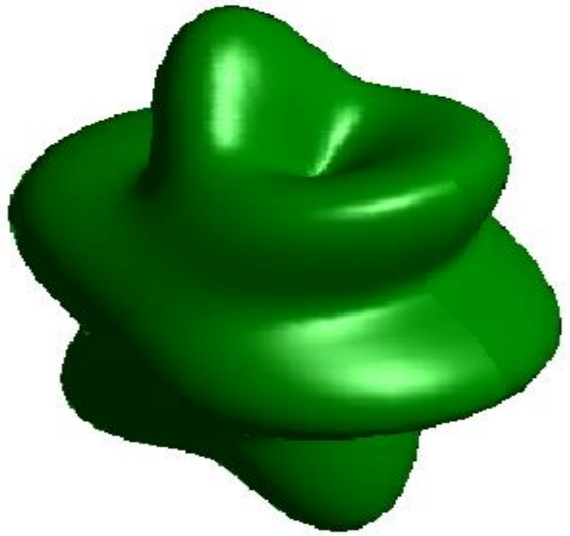
Flat Shading



Gouraud Shading



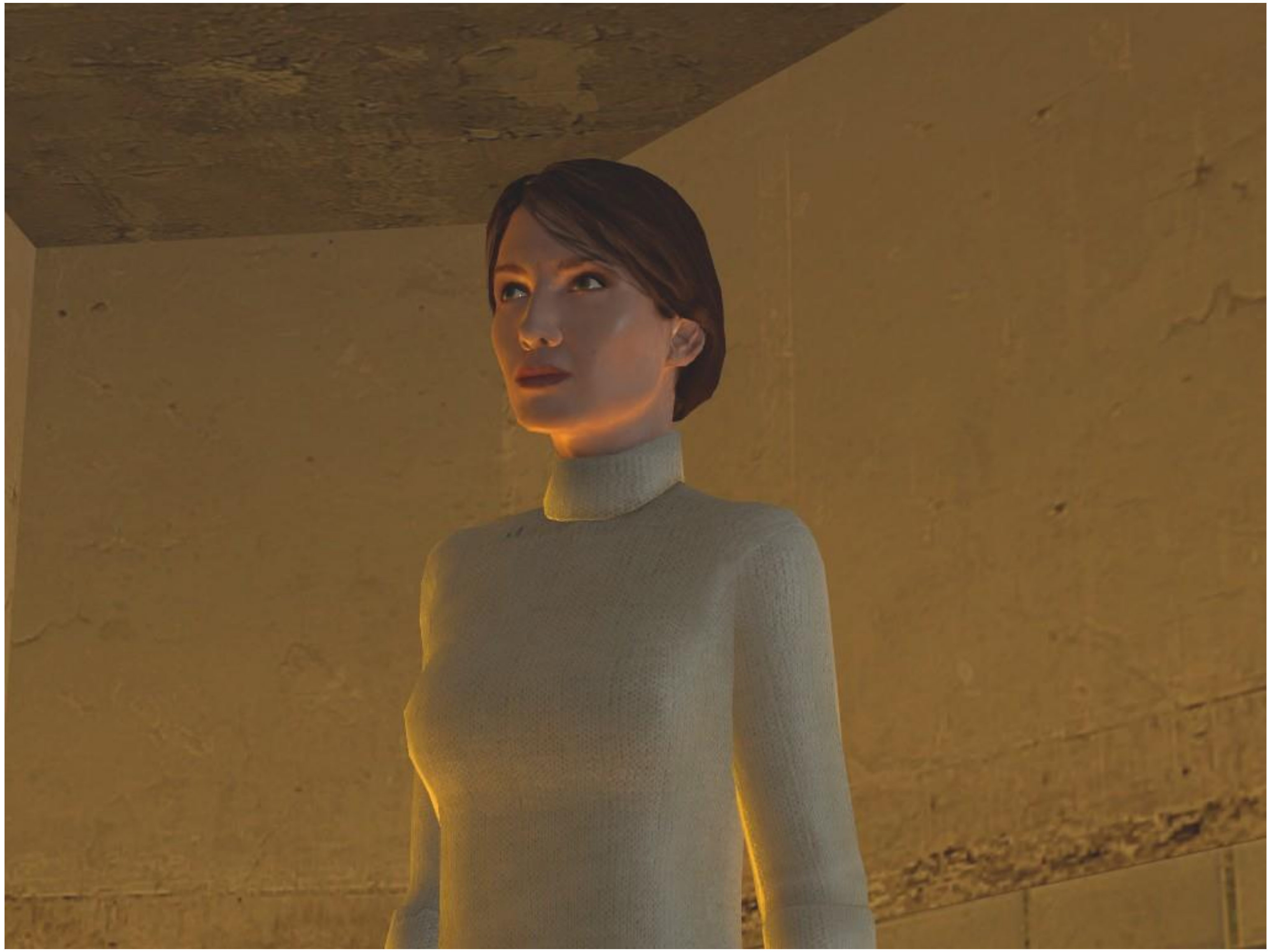
Phong Shading



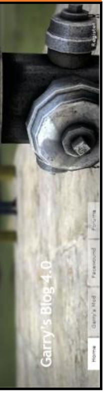
Phong Surface Rendering Implementation

- Phong shading is much slower than Gouraud shading as the lighting model is revaluated so many times
- However, there are fast Phong surface rendering approaches that can be implemented iteratively
- Typically Phong shading is implemented as part of a visible surface detection technique

Phong Shading Examples



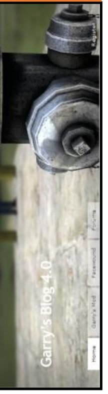
Images come from Garry's Mod: http://www.garry.tv/



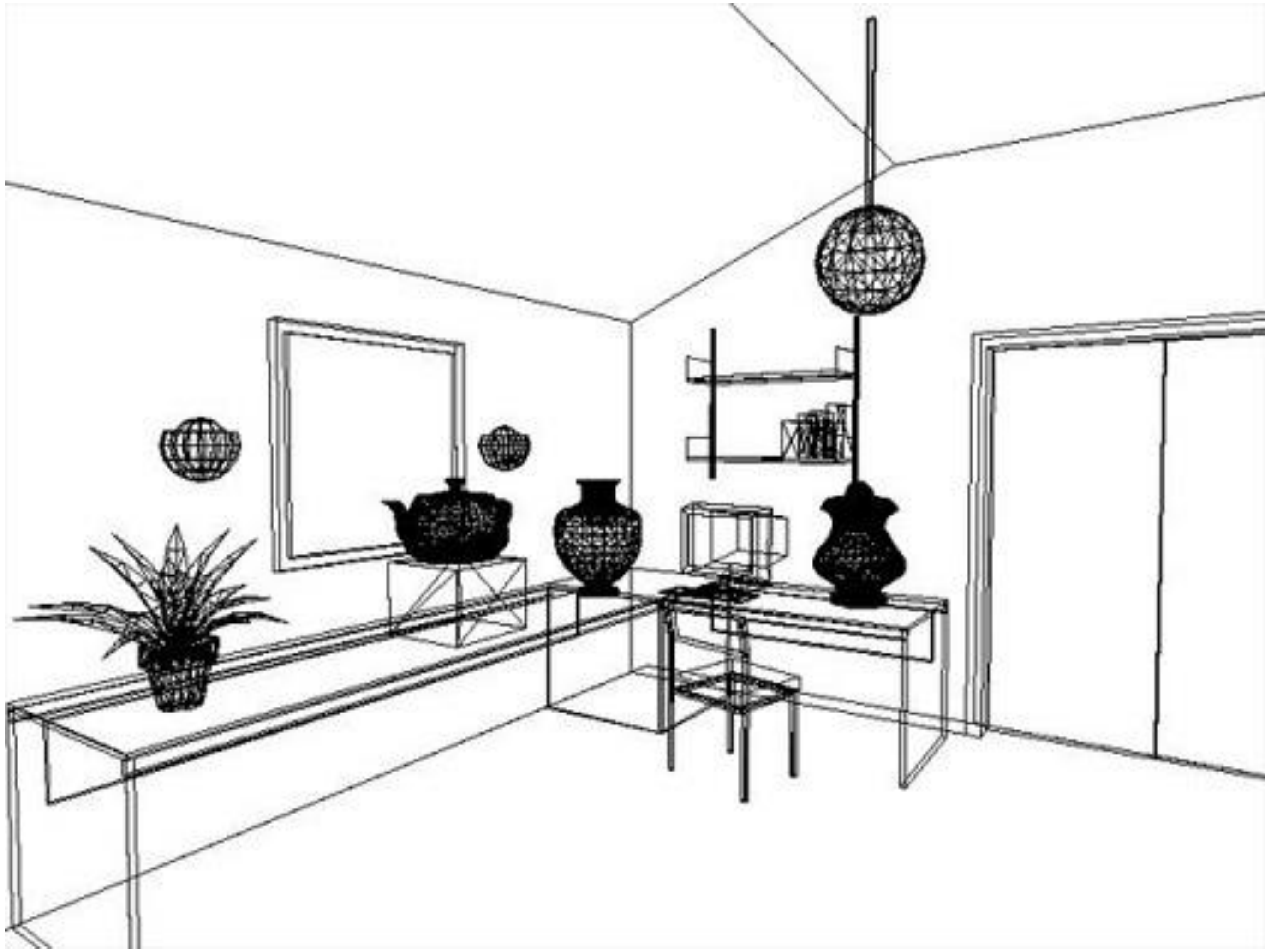
Phong Shading Examples

HEALTH 20
AMMO 3

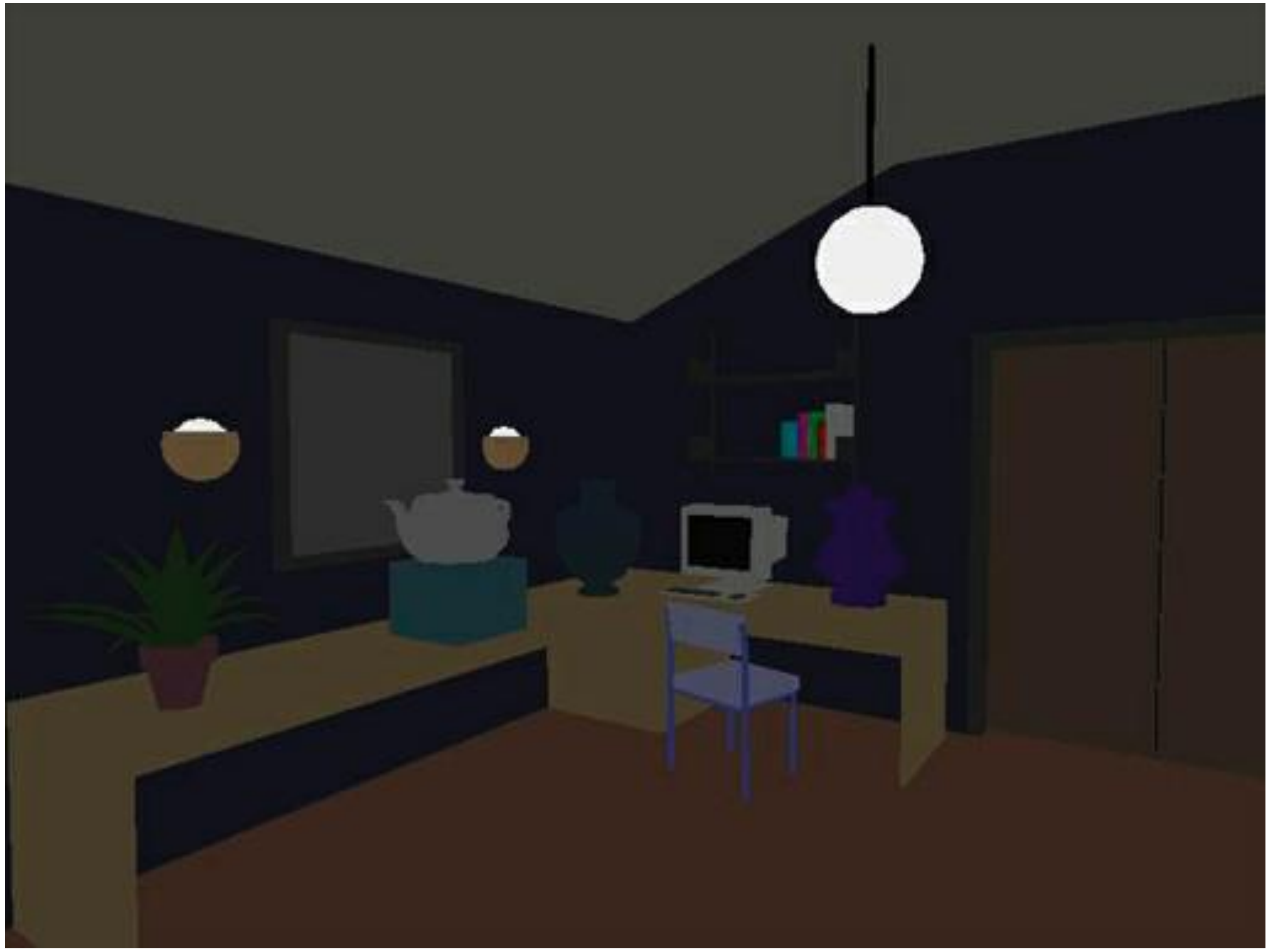
Images come from Garry's Mod: http://www.garry.tv/



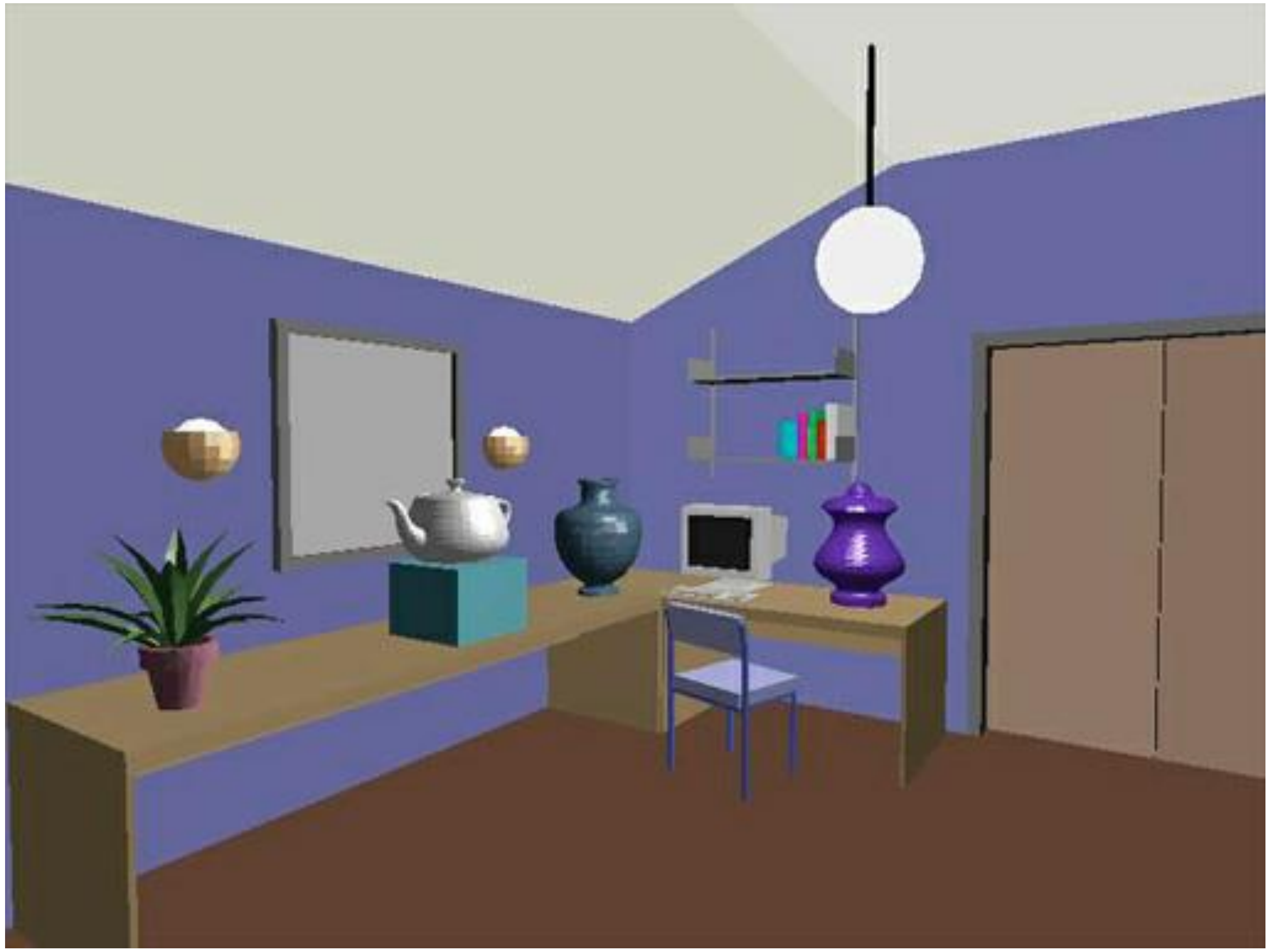
Wire-frame



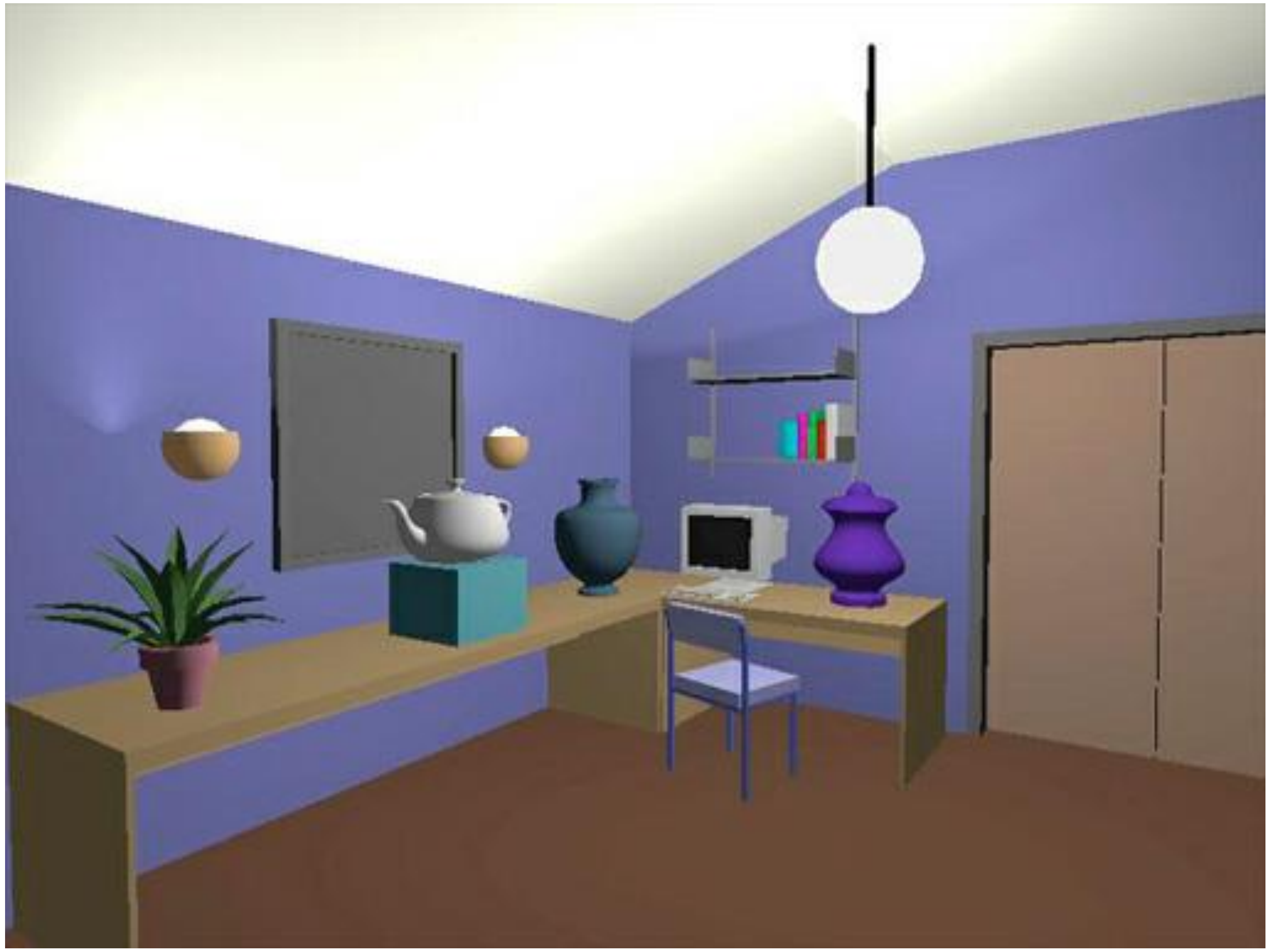
Ambient Illumination Only



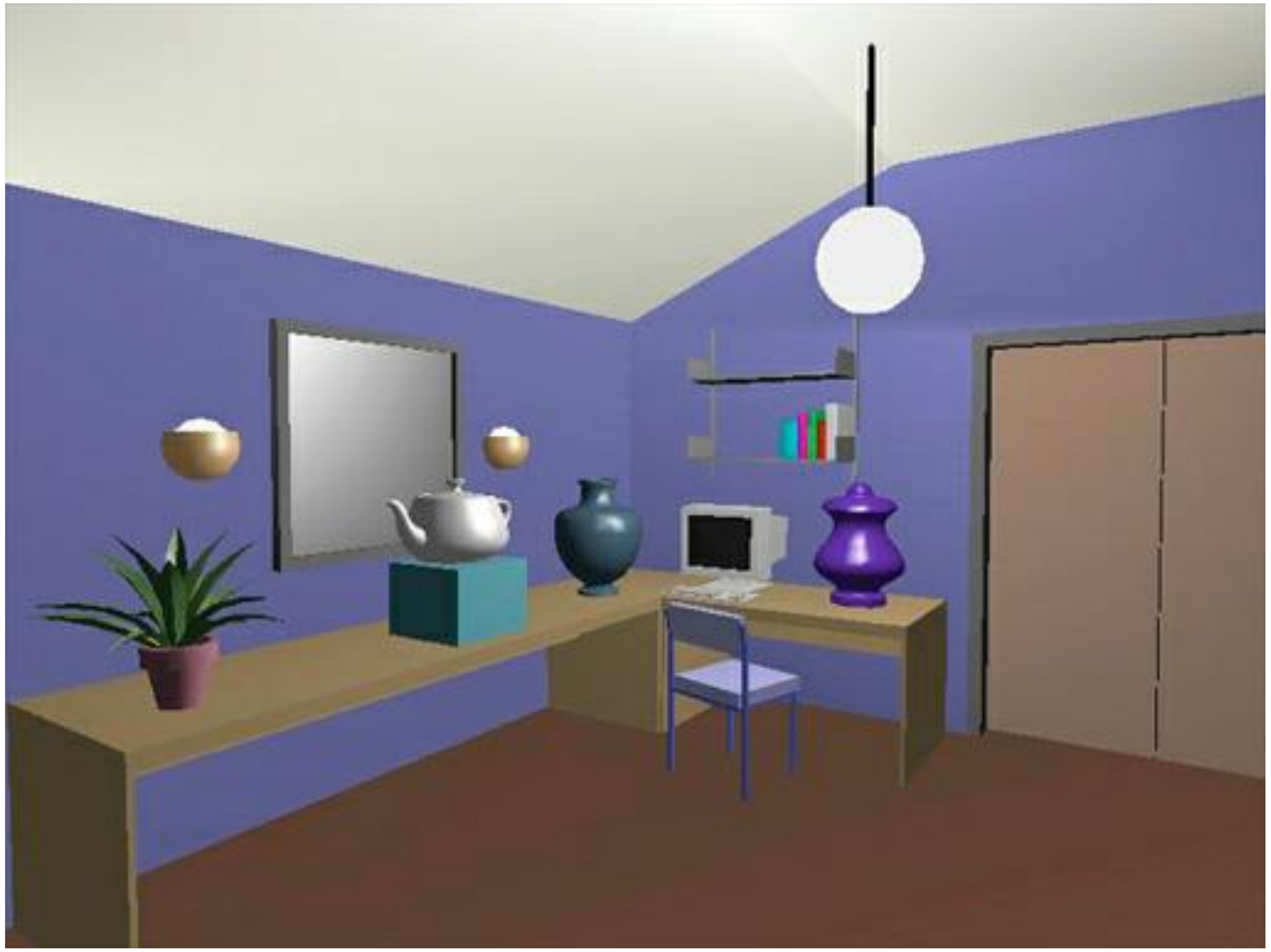
Flat-Shading



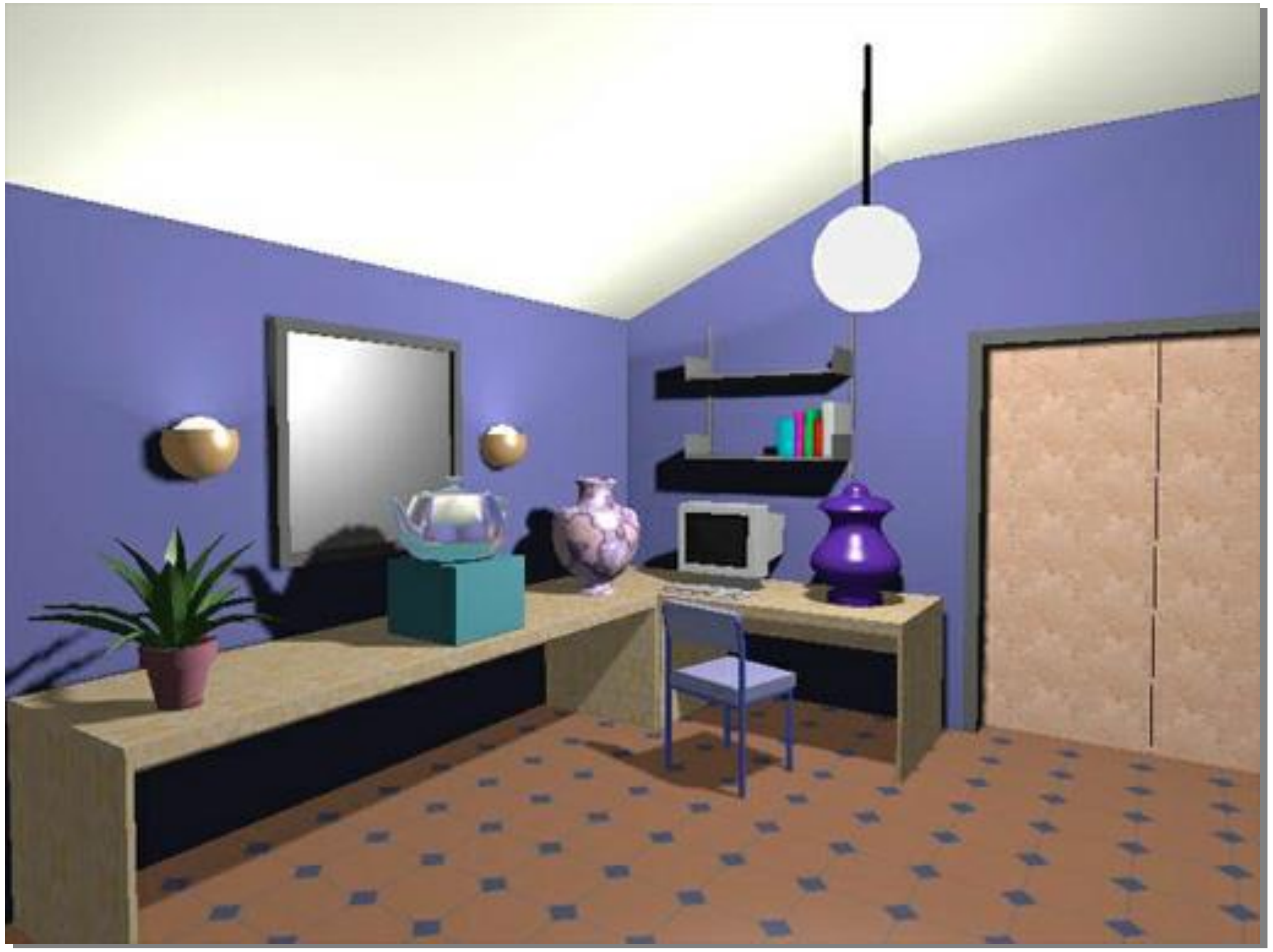
Gouraud Shading



Phong Shading

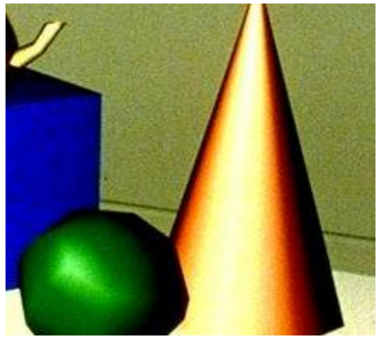


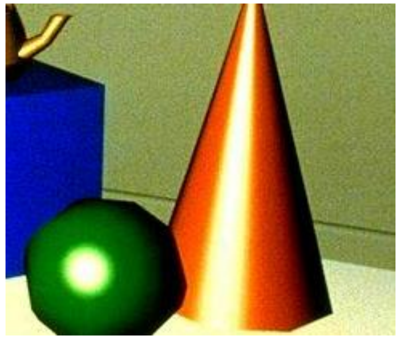
Shadows, Texture & Enviromental Mappings

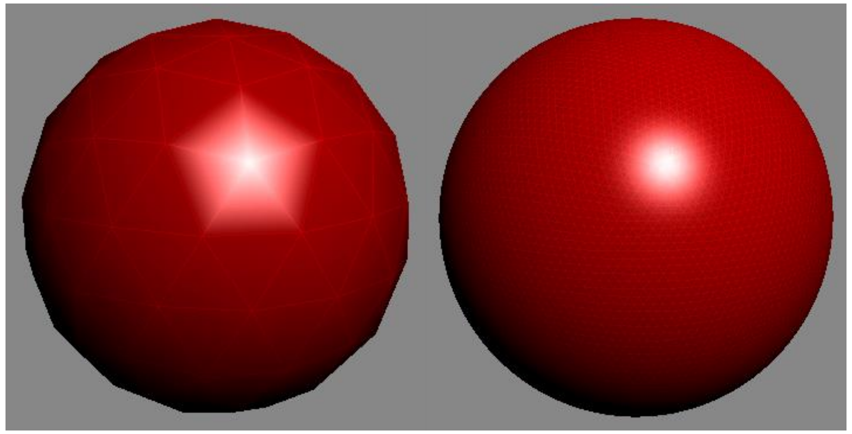


Περίληψη

- 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 better still, but can be slow


Gouraud


Phong


Gouraud Phong

http://www.cgchannel.com/2010/11/cg-science-for-artists-part-2-the-real-time-rendering-pipeline/

More Sophisticated Techniques/Hacks

- What we have doesn't cover most scenes – use upcoming hacks to approximate desired features
- Hacks for geometric detail
  - Complicated color, geometry – texture mapping, normal and bump mapping, etc.
- Hacks for global illumination effects
  - Reflections, shadows – environment mapping, shadow stencil volumes and shadow mapping
- Many of these hacks (esp. geometry hacks) originally designed for polygonal rendering are used in combination with ray/path-tracing rendering methods to get greater photo-realism