Computer Graphics

Shading

Based on slides by Dianna Xu, Bryn Mawr College

Image Synthesis and Shading

SHADING

SHADOWS

REFLECTIONS

HIGHLIGHTS

TRANSLUCENCY

TEXTURE

BUMPS

REFRACTIONS

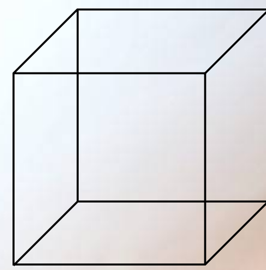
Perception of 3D Objects

- Displays almost always 2 dimensional.
- Depth cues needed to restore the third dimension.
- Need to portray planar, curved, textured, translucent, etc. surfaces.
- Model light and shadow.

Depth Cues

Eliminate hidden parts (lines or surfaces)

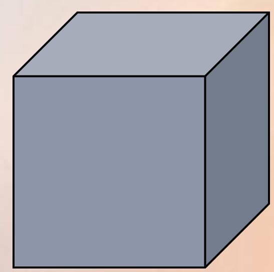
“Wire-frame”



Front?

Back?

“Opaque Object”



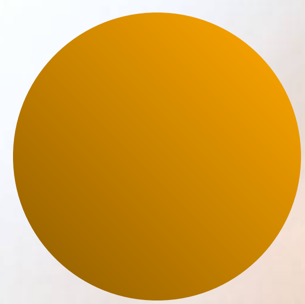
Convex?

Concave?

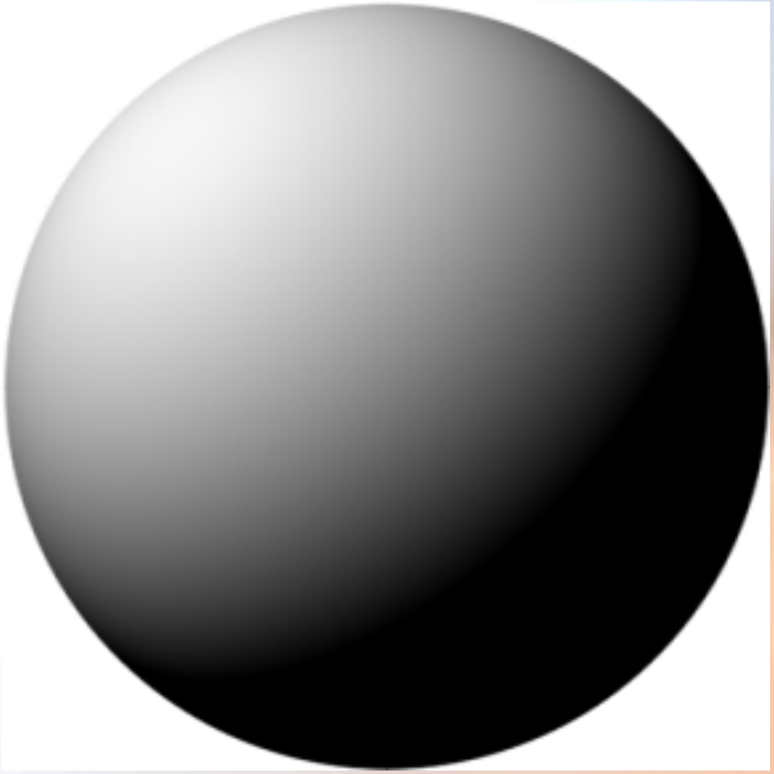
Why we need shading

- Suppose we build a model of a sphere using many polygons and color it with `glColor`. We get something like

- But we want



Shading implies Curvature

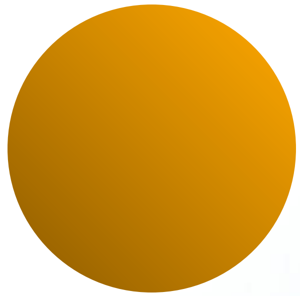


Shading Motivation

- Originated in trying to give polygonal models the *appearance* of smooth curvature.
- Numerous shading models
  - Quick and dirty
  - Physics-based
  - Specific techniques for particular effects
  - Non-photorealistic techniques (pen and ink, brushes, etching)

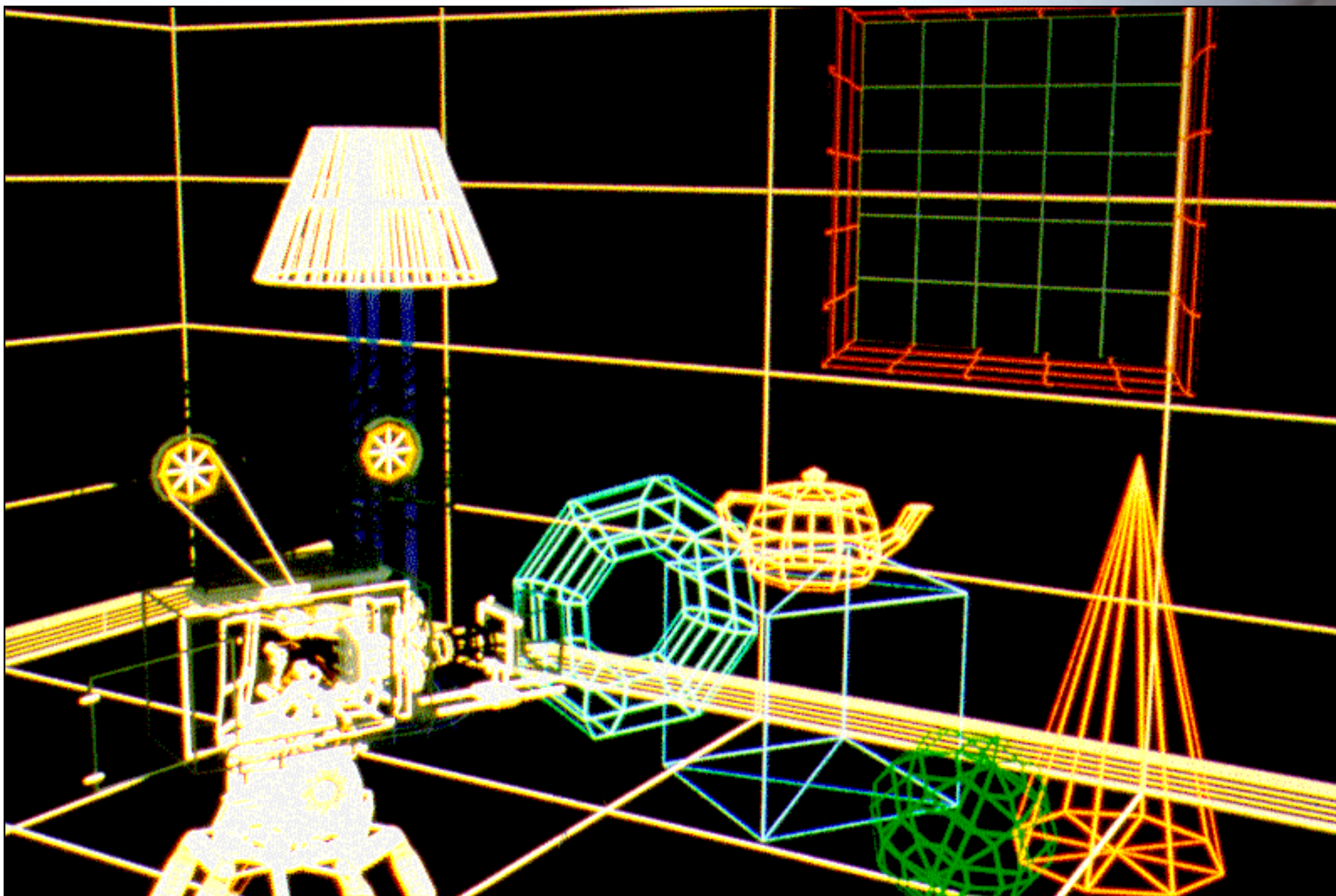
Shading

- Why does the image of a real sphere look like

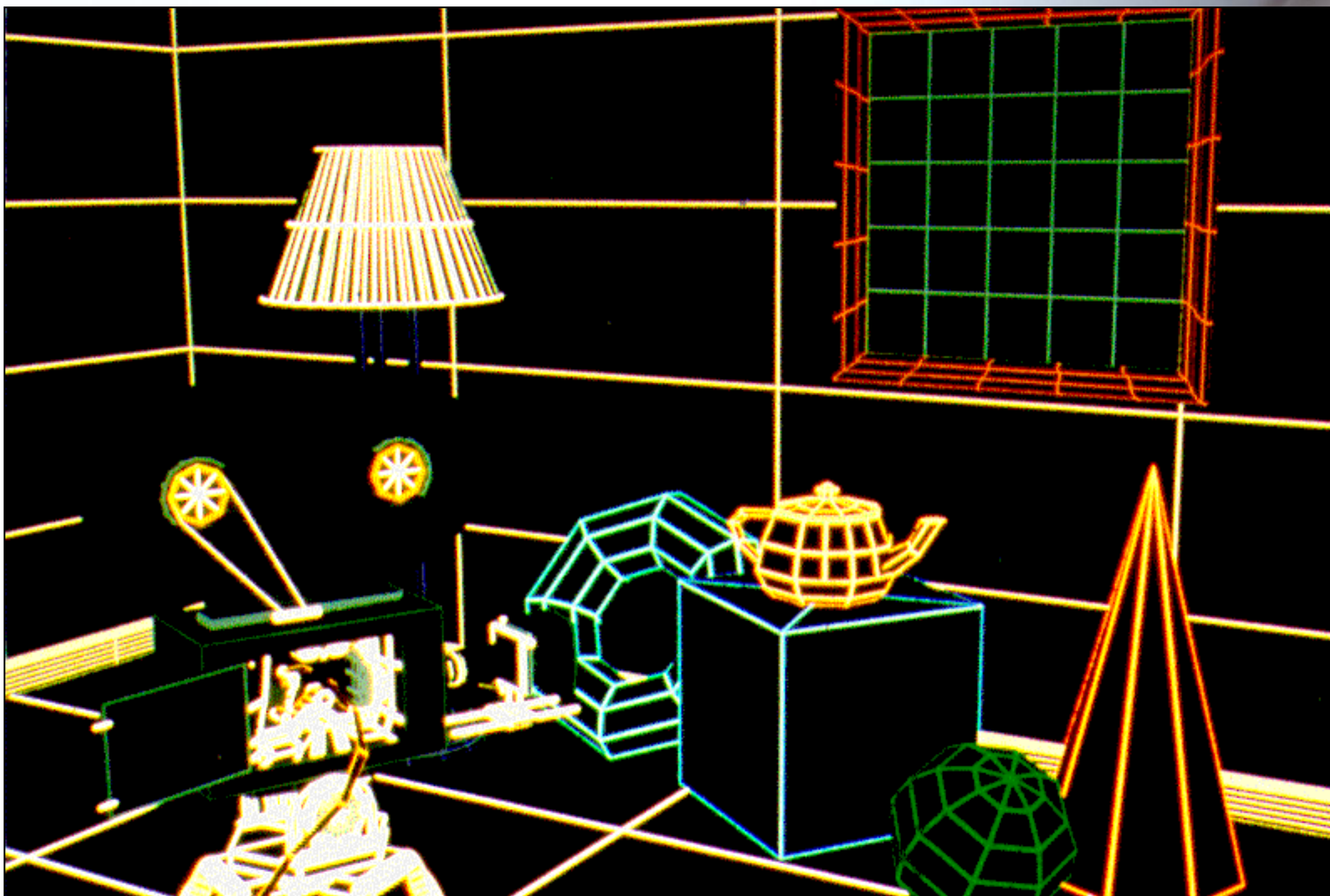


- Light-material interactions cause each point to have a different color or shade
- Need to consider
  - Light sources
  - Material properties
  - Location of viewer
  - Surface orientation

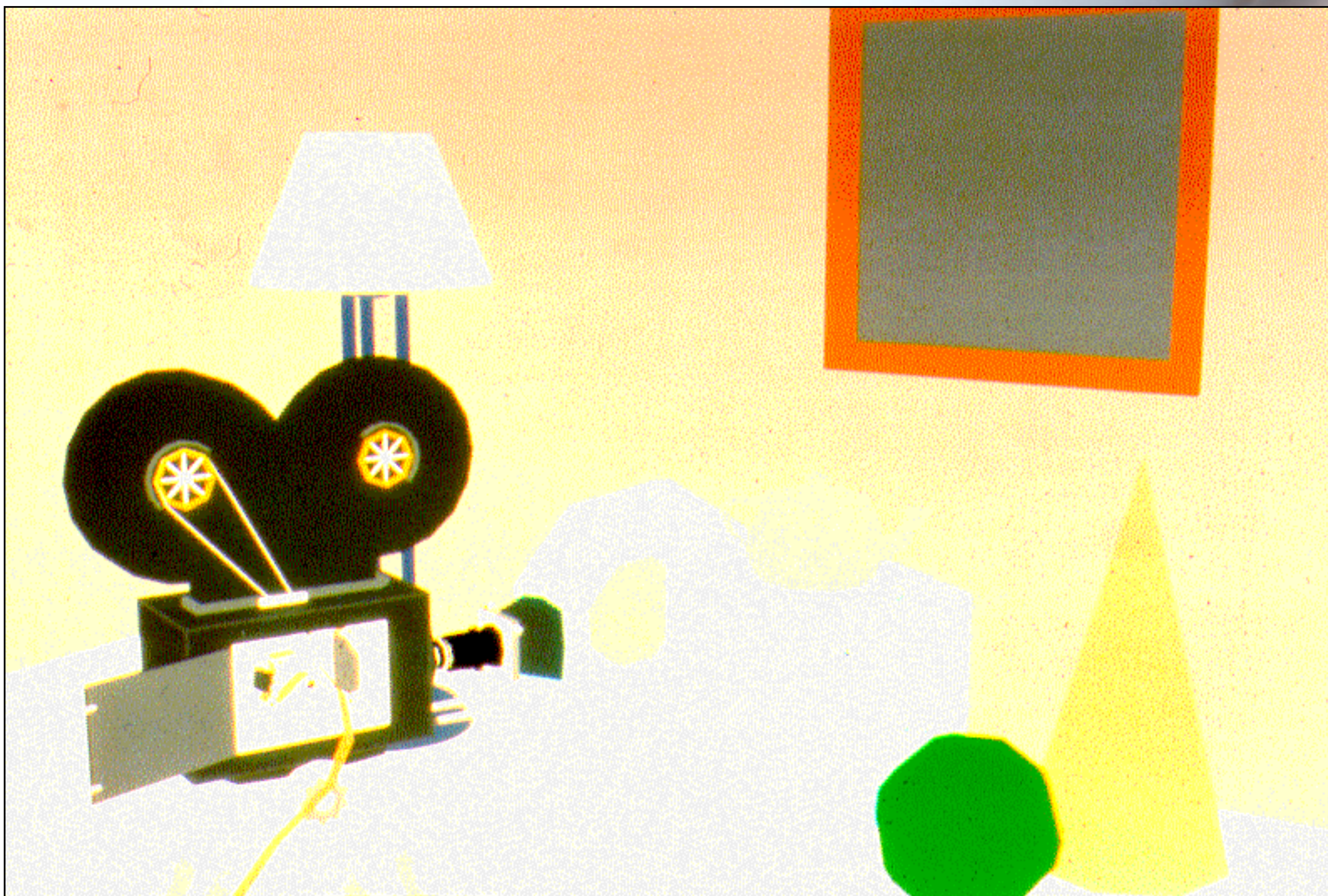
Wireframe: Color, no Substance



Substance but no Surfaces

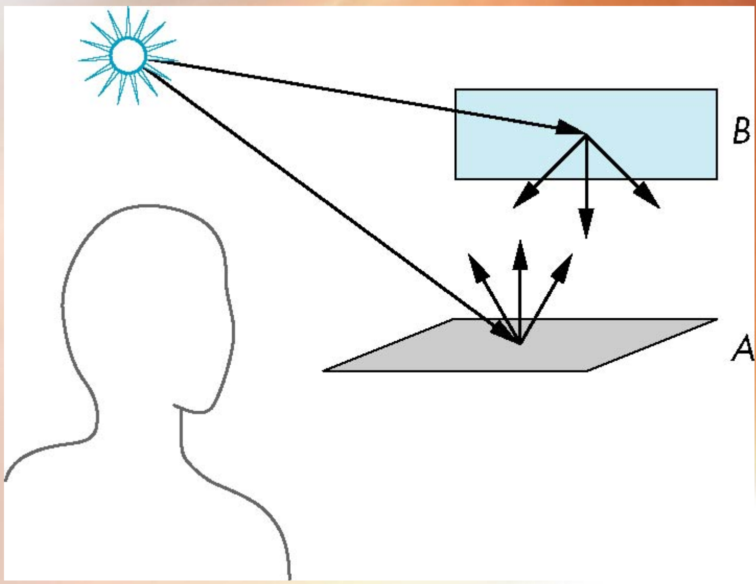


Why the Surface Normal is Important



Scattering

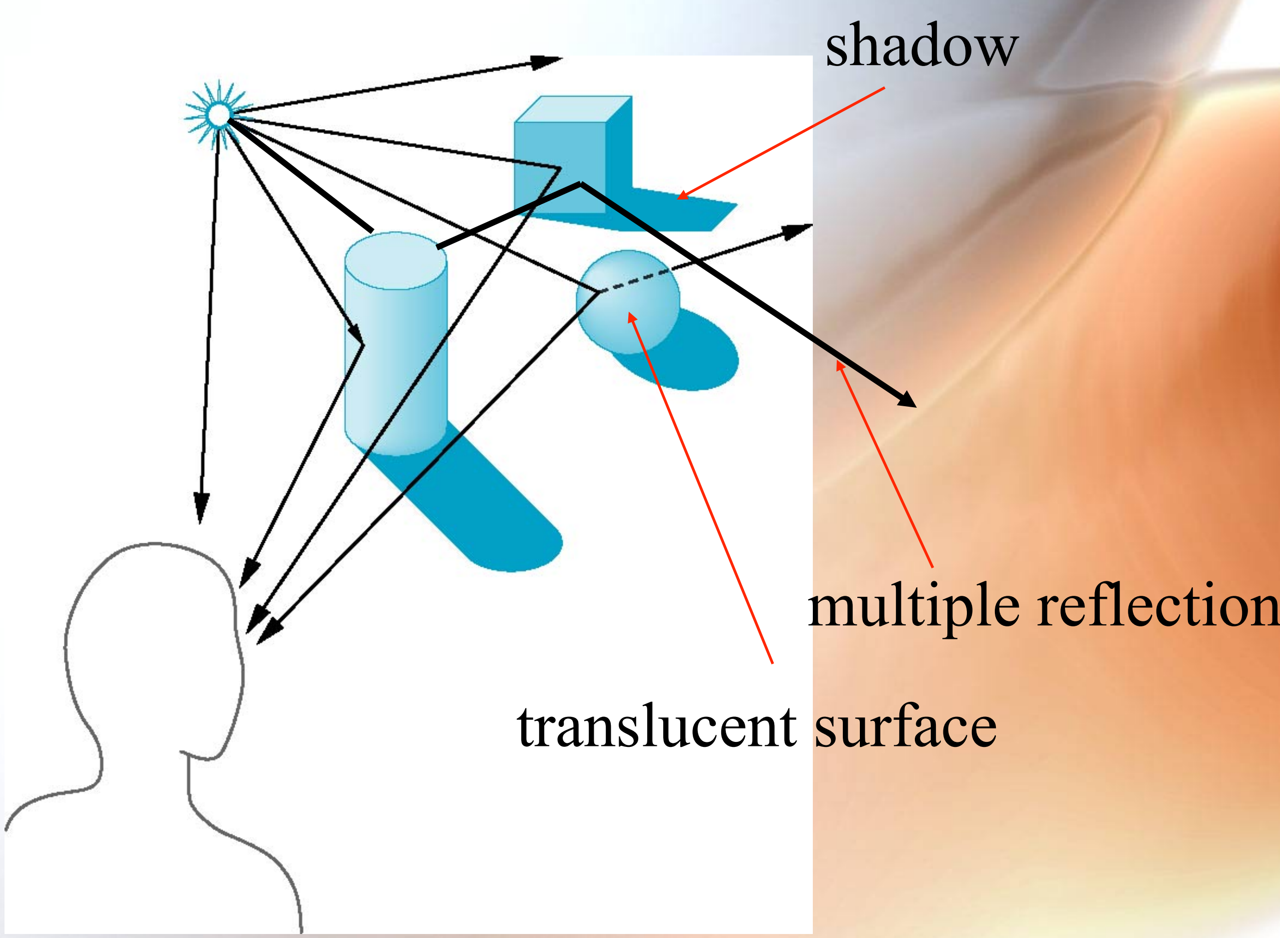
- Light strikes A
  - Some scattered
  - Some absorbed
- Some of scattered light strikes B
  - Some scattered
  - Some absorbed
- Some of this scattered light strikes A and so on



Rendering Equation

- The infinite scattering and absorption of light can be described by the *rendering equation*
  - Cannot be solved in general
  - Ray tracing is a special case for perfectly reflecting surfaces
- Rendering equation is global and includes
  - Shadows
  - Multiple scattering from object to object

Global Effects



Local vs Global Rendering

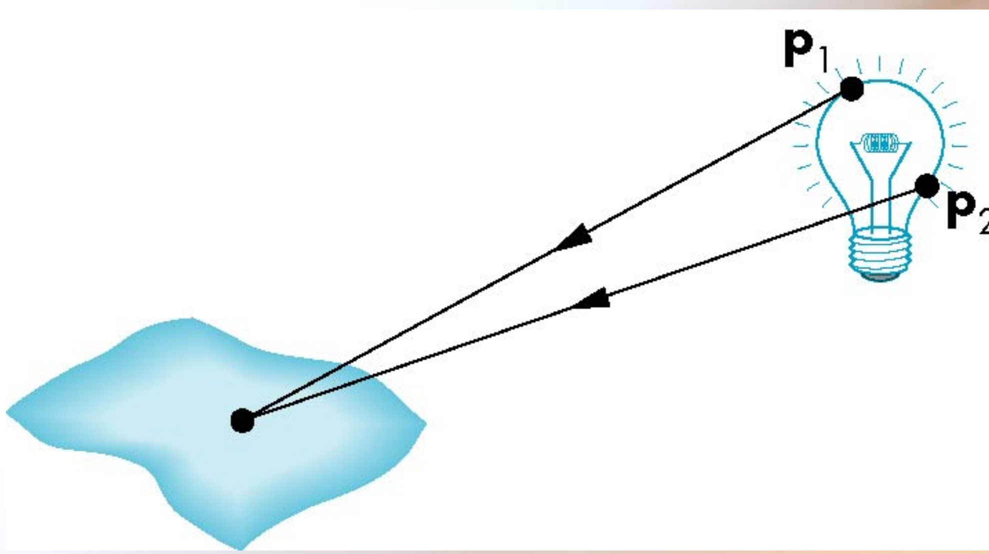
- Correct shading requires a global calculation involving all objects and light sources
  - Incompatible with pipeline model which shades each polygon independently (local rendering)
- However, in computer graphics, especially real time graphics, we are happy if things “look right”
  - Exist many techniques for approximating global effects

Light-Material Interaction

- Light that strikes an object is partially absorbed and partially scattered (reflected)
- The amount reflected determines the color and brightness of the object
  - A surface appears red under white light because the red component of the light is reflected and the rest is absorbed
- The reflected light is scattered in a manner that depends on the smoothness and orientation of the surface

Light Sources

General light sources are difficult to work with because we must integrate light coming from all points on the source

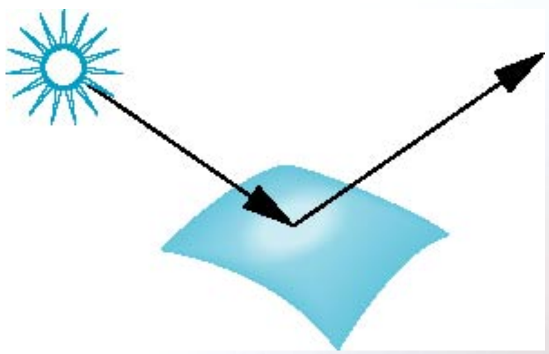


Simple Light Sources

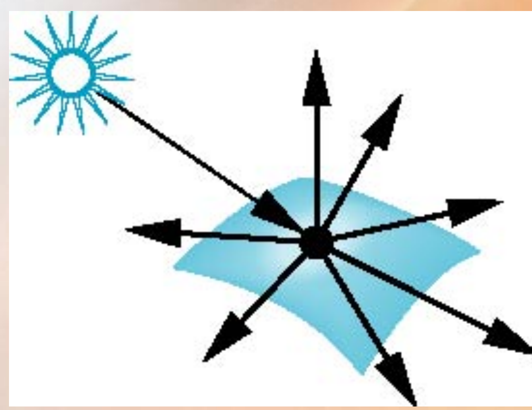
- Point source
  - Model with position and color
  - Distant source = infinite distance away (parallel)
- Spotlight
  - Restrict light from ideal point source
- Ambient light
  - Same amount of light everywhere in scene
  - Can model contribution of many sources and reflecting surfaces

Surface Types

- The smoother a surface, the more reflected light is concentrated in the direction a perfect mirror would reflected the light
- A very rough surface scatters light in all directions



smooth surface



rough surface

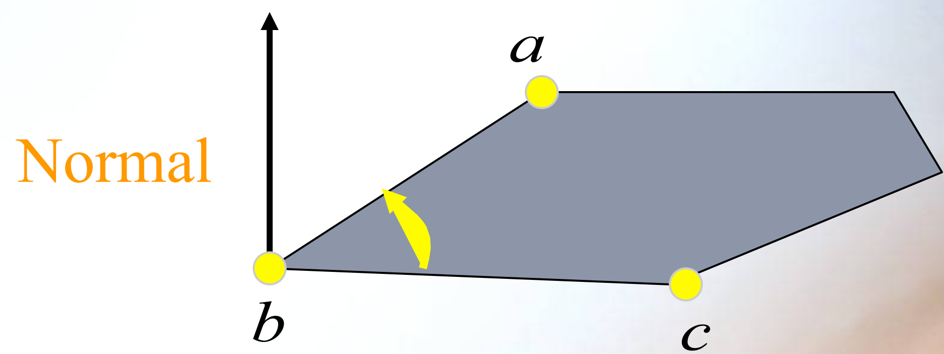
Shading Computation

WANT: Color at some object point P.

- Know direction of eye/viewer **v** and (point) light source **l**.
- Know surface geometry, or can otherwise compute the surface normal **n** at P.
- Know desired surface reflectance properties at P. (We'll look at some of these soon, too.)
- Assume all vectors are normalized to unit length

Computing the Polygon Face Normal Vector

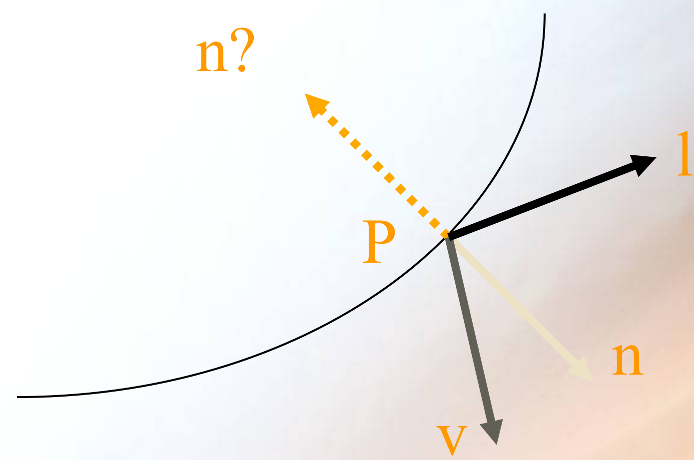
$$\mathbf{n} = (\mathbf{c} - \mathbf{b}) \times (\mathbf{a} - \mathbf{b})$$



Normal = cross product of 3 or more non-collinear vertices

$$Normal(a,b,c) = \begin{cases} (c_y - b_y)(a_z - b_z) - (c_z - b_z)(a_y - b_y) \\ (c_z - b_z)(a_x - b_x) - (c_x - b_x)(a_z - b_z) \\ (c_x - b_x)(a_y - b_y) - (c_y - b_y)(a_x - b_x) \end{cases}$$

Finding the Surface Normal at Point P



Want outward-facing normal -- the one with a component towards v.
If $(n \cdot v) < 0$ then $n \leftarrow -n$

For curved surfaces the normal can be computed explicitly.

Shading Computation

Given **l, n, v**

Compute reflected color as a function of l, n, v and other attributes of the surface and lighting.

- **A local reflection model enables the calculation of the the reflected light intensity from a point on the surface of an object.**

Light Reflection from a Surface at a Point

Ignoring color for now: the necessary computations are the same for each source and for each primary color

$$I = \underbrace{k_a}_{I_a} L_a + \underbrace{k_d}_{I_d} L_d + \underbrace{k_s}_{I_s} L_s \quad for \quad k_a + k_d + k_s = 1$$

to account for all reflected light.

This means the surface shade is a function of

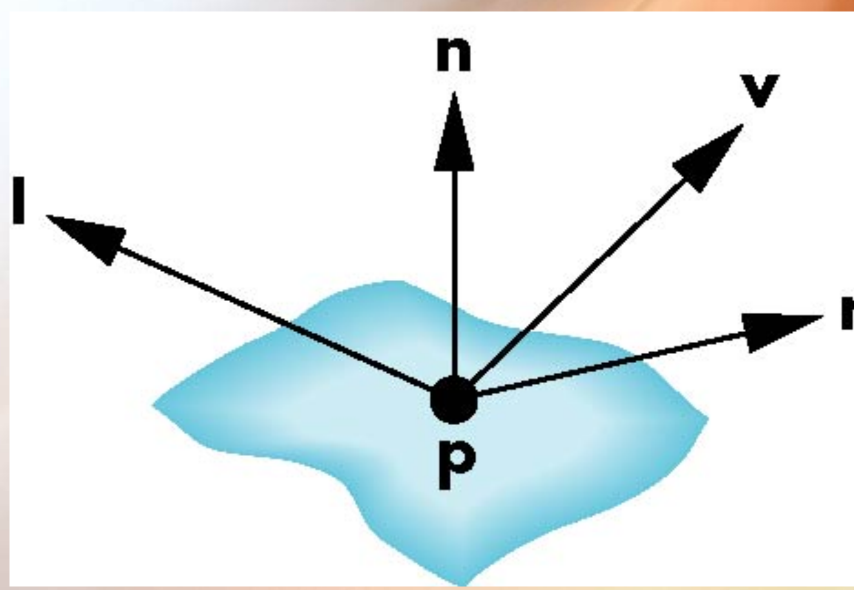
L_a ambient light

L_d diffuse light

L_s specular light

Phong Model

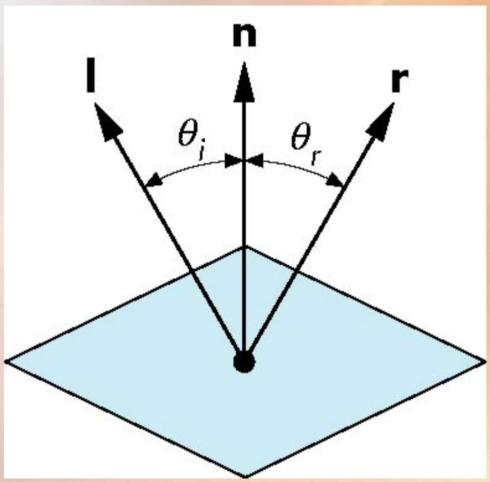
- A simple model that can be computed rapidly
- Has three components
  - Diffuse
  - Specular
  - Ambient
- Uses four vectors
  - To source **l**
  - To viewer **v**
  - Normal **n**
  - Perfect reflector **r**



Ideal Reflector

- Normal is determined by local orientation
- Angle of incidence = angle of reflection
- The three vectors must be coplanar

$$\mathbf{r} = 2\,(\mathbf{l} \cdot \mathbf{n})\,\mathbf{n} - \mathbf{l}$$



Ambient Light

- Ambient light is the result of multiple interactions between (large) light sources and the objects in the environment
- Amount and color depend on both the color of the light(s) and the material properties of the object
- Add $k_a \, I_a$ to model ambient term

reflection coef intensity of ambient light

Lambertian Surface

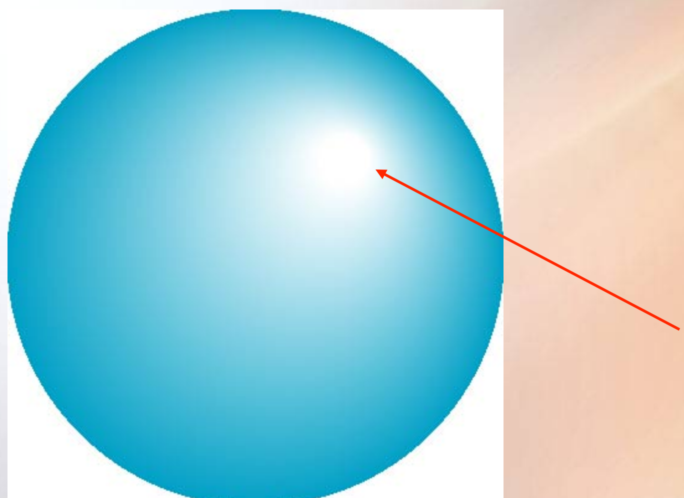
- Perfectly diffuse reflector
- Light scattered equally in all directions
- Amount of light reflected is proportional to the vertical component of incoming light
  - reflected light can be modeled as $\cos \theta_i$
  - $\cos \theta_i = l \cdot n$ if vectors normalized

The Reflectance Terms

- **Diffuse:** $I_d = k_d L_d (l \bullet n)$
- **Ambient:** $I_a = k_a L_a$
- **Specular: ?**

Specular Surfaces

- **Most surfaces are neither ideal diffusers nor perfectly specular (ideal refectors)**
- **Smooth surfaces show specular highlights due to incoming light being reflected in directions concentrated close to the direction of a perfect reflection**



Modeling Specular Relections

- Phong proposed using a term that dropped off as the angle between the viewer and the ideal reflection increased

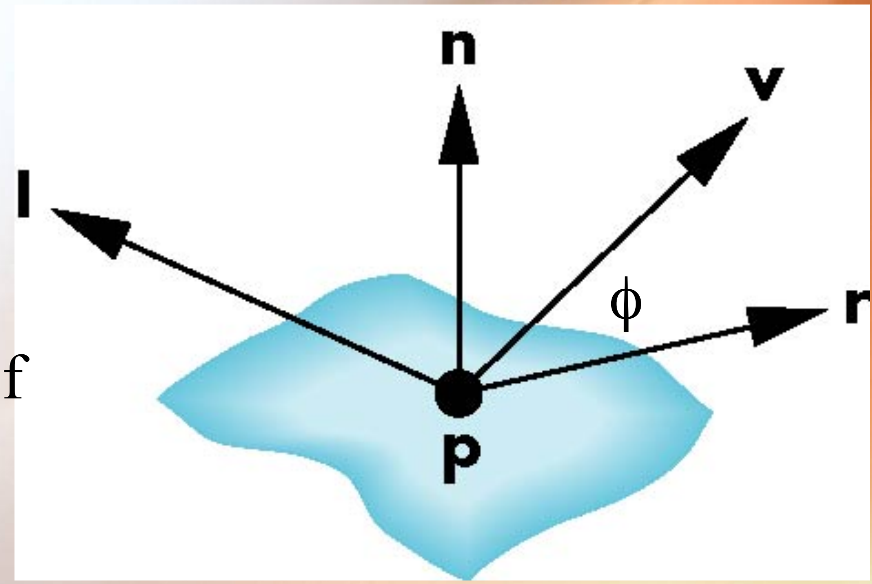
$$I_r \sim k_s \, \mathbf{I} \cos^{\alpha}\phi$$

reflected intensity

absorption coef

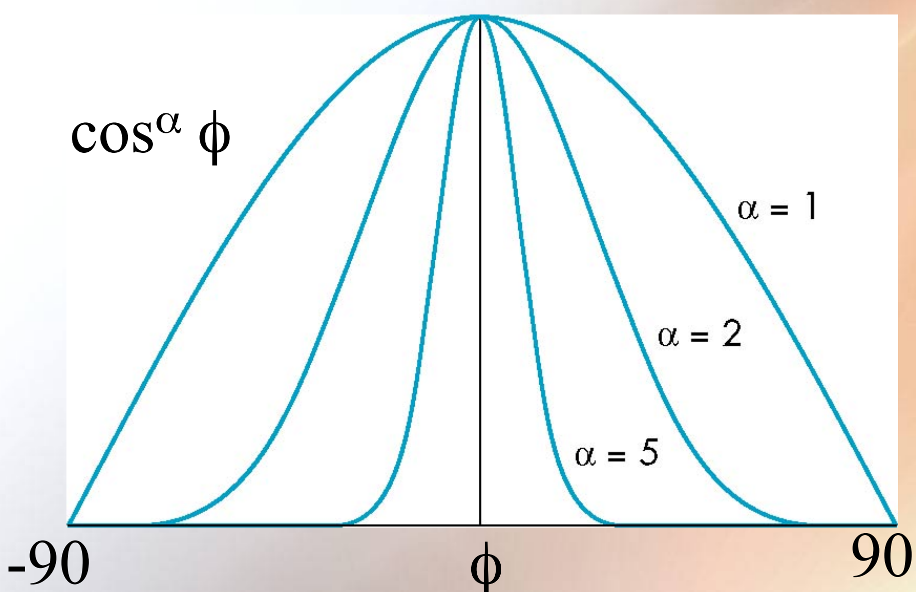
incoming intensity

shininess coef



The Shininess Coefficient

- Values of α between 100 and 200 correspond to metals
- Values between 5 and 10 give surface that look like plastic



The Reflectance Terms

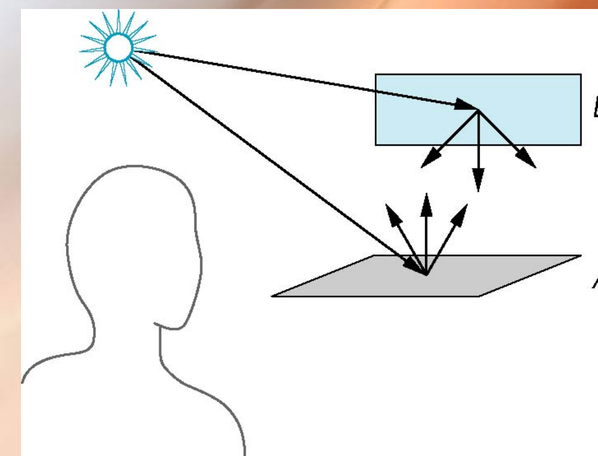
- **Diffuse:** $I_d = k_d L_d (l \bullet n)$
- **Ambient:** $I_a = k_a L_a$
- **Specular:** $I_s = k_s L_s (v \bullet r)^\alpha$

r is the direction of reflection
a approximates the amount of mirror reflection

Distance Terms

- The light from a point source that reaches a surface is inversely proportional to the square of the distance between them

- We can add a factor of the form $1/(ad + bd + cd^2)$ to the diffuse and specular terms



- The constant and linear terms soften the effect of the point source

Light Sources

- In the Phong Model, we add the results from each light source
- Each light source has separate diffuse, specular, and ambient terms to allow for maximum flexibility even though this form does not have a physical justification
- Separate red, green and blue components
- Hence, 9 coefficients for each point source
  - I_{dr}, I_{dg}, I_{db}, I_{sr}, I_{sg}, I_{sb}, I_{ar}, I_{ag}, I_{ab}

Material Properties

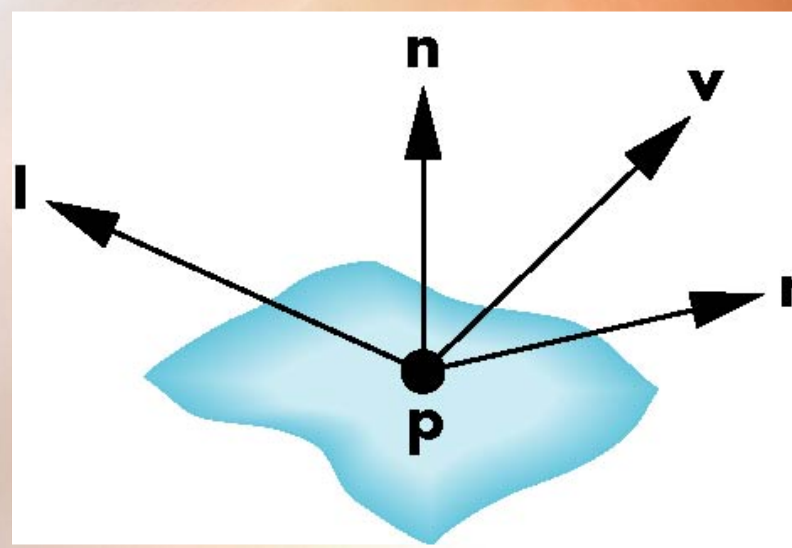
- Material properties match light source properties
  - Nine absorbtion coefficients
    - k_{dr}, k_{dg}, k_{db}, k_{sr}, k_{sg}, k_{sb}, k_{ar}, k_{ag}, k_{ab}
  - Shininess coefficient α

Adding up the Components

For each light source and each color component, the Phong model can be written (without the distance terms) as

$$I = \underbrace{k_a L_a}_{I_a} + \underbrace{k_d L_d (l \bullet n)}_{I_d} + \underbrace{k_s L_s (v \bullet r)^{\alpha}}_{I_s} \qquad k_a + k_d + k_s = 1$$

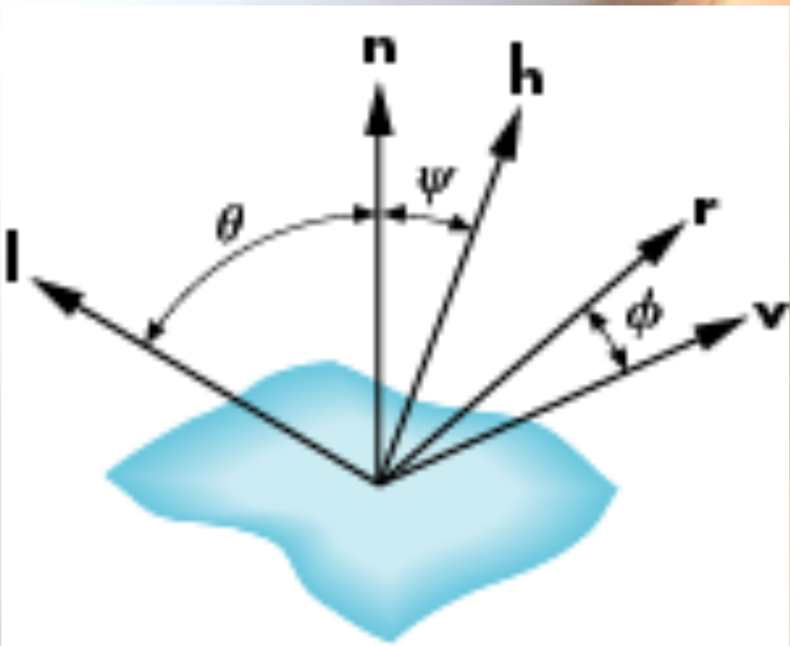
For each color component we add contributions from all sources



The Halfway Vector

- **Consider the unit vector halfway between** l **and** v

h **=** **(**l+v**)**/**|**l**+**v**|**



If we use n·h **rather than** r·v, **we avoid calculation of** r

Modified Phong Model

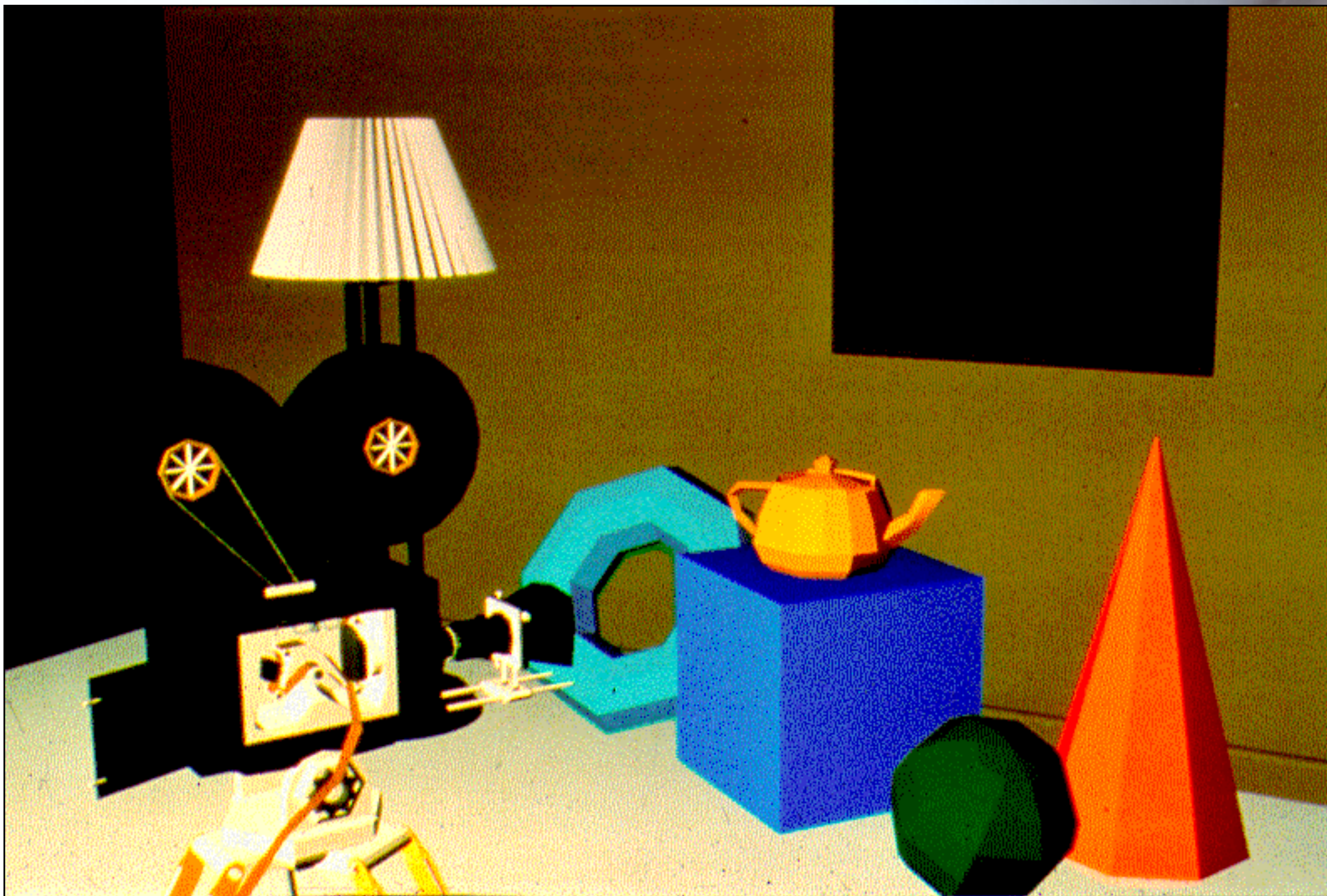
- When v, l, n and r lie in the same plane, $2\psi = \phi$
  - So, the same exponent α yields a smaller specular highlight
  - Absorb the change in angle into α

- The use of the halfway vector was first suggested by Blinn
- The resulting lighting model is known as the Blinn-Phong or modified Phong model
- OpenGL default is the modified Phong model

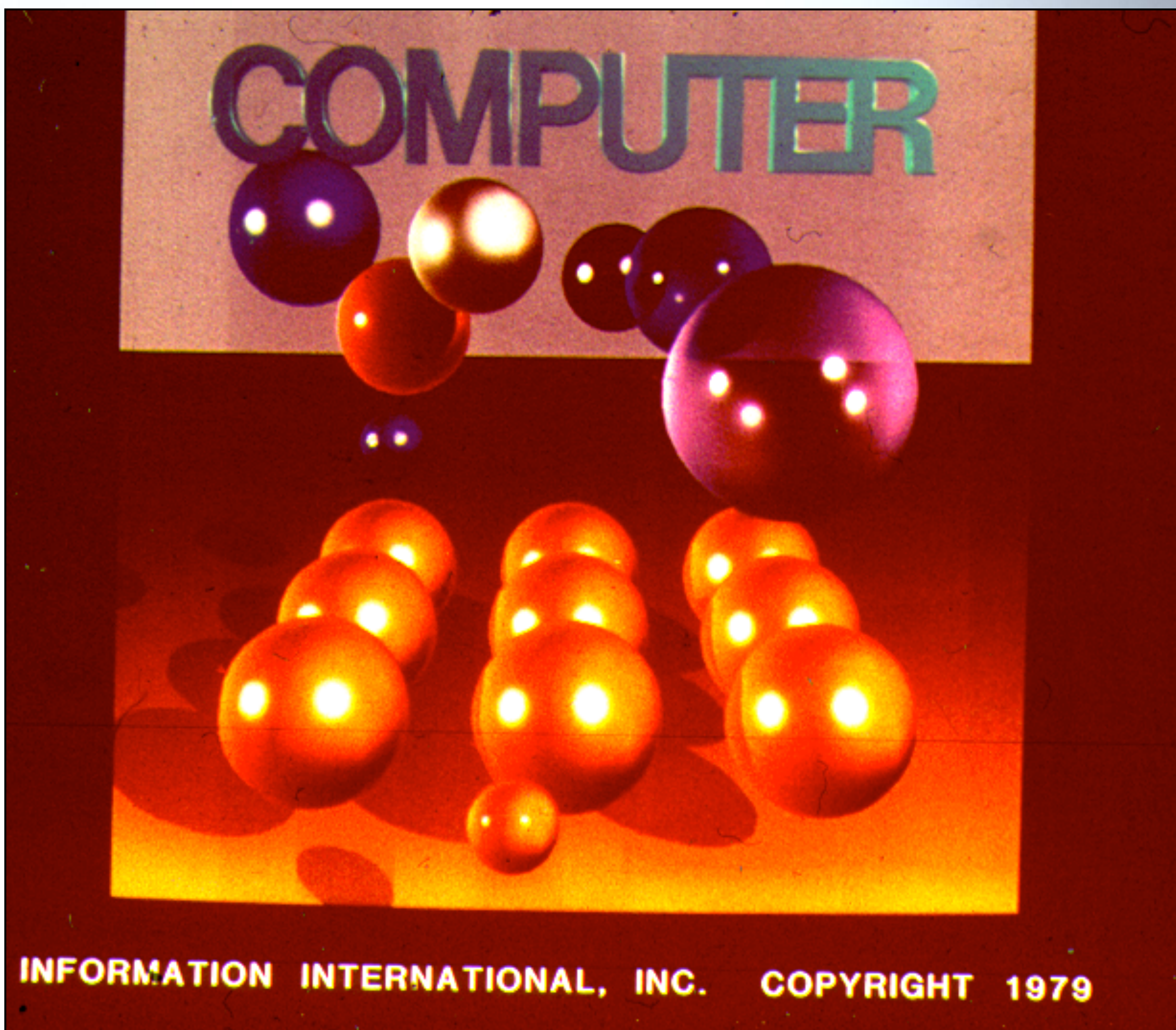
Shading Polygons

- In our model, shading depends on the three vectors l, n, and v
- If viewer is distant, then v is constant
- If light is distant, then l is constant
- Distant is interpreted relative to the size of the polygon
- In OpenGL, v and l are constant by default

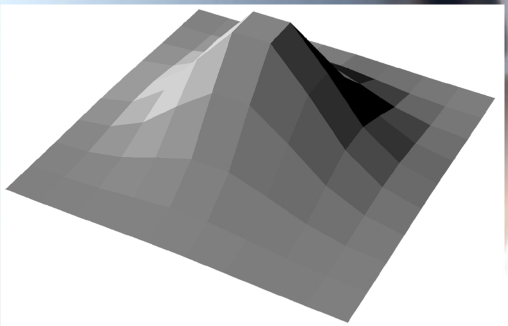
Diffuse Shading on Polygon Surfaces



Phong Specular Spheres

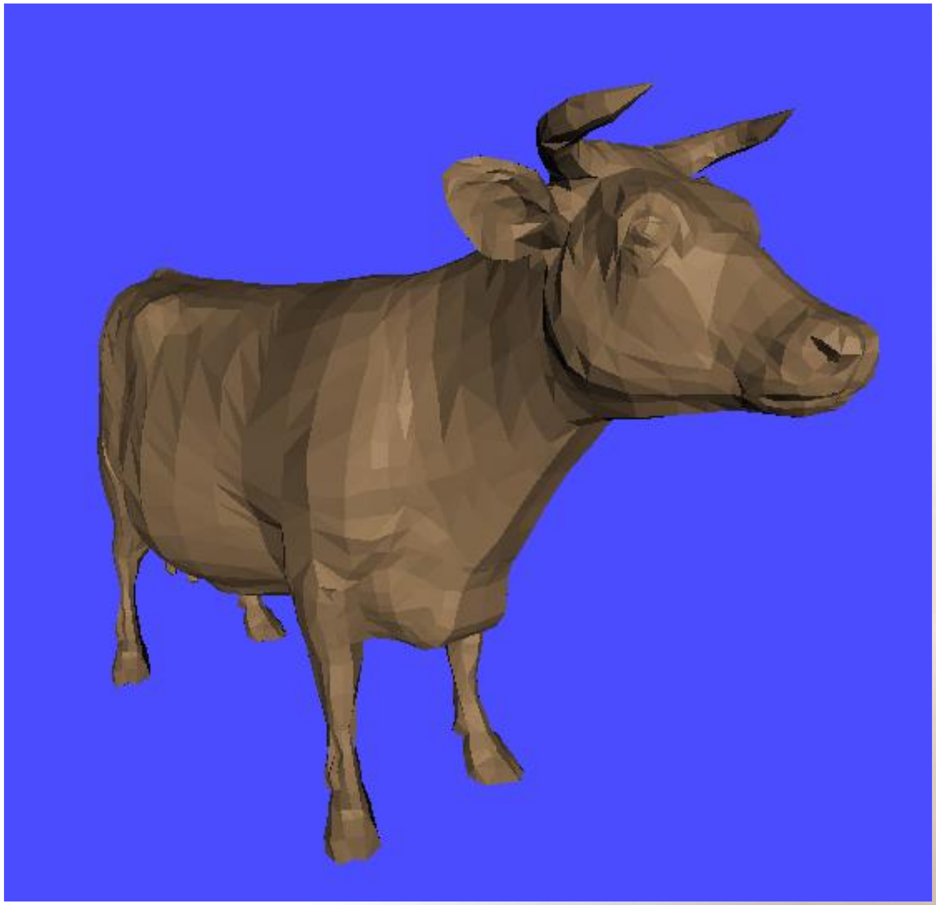


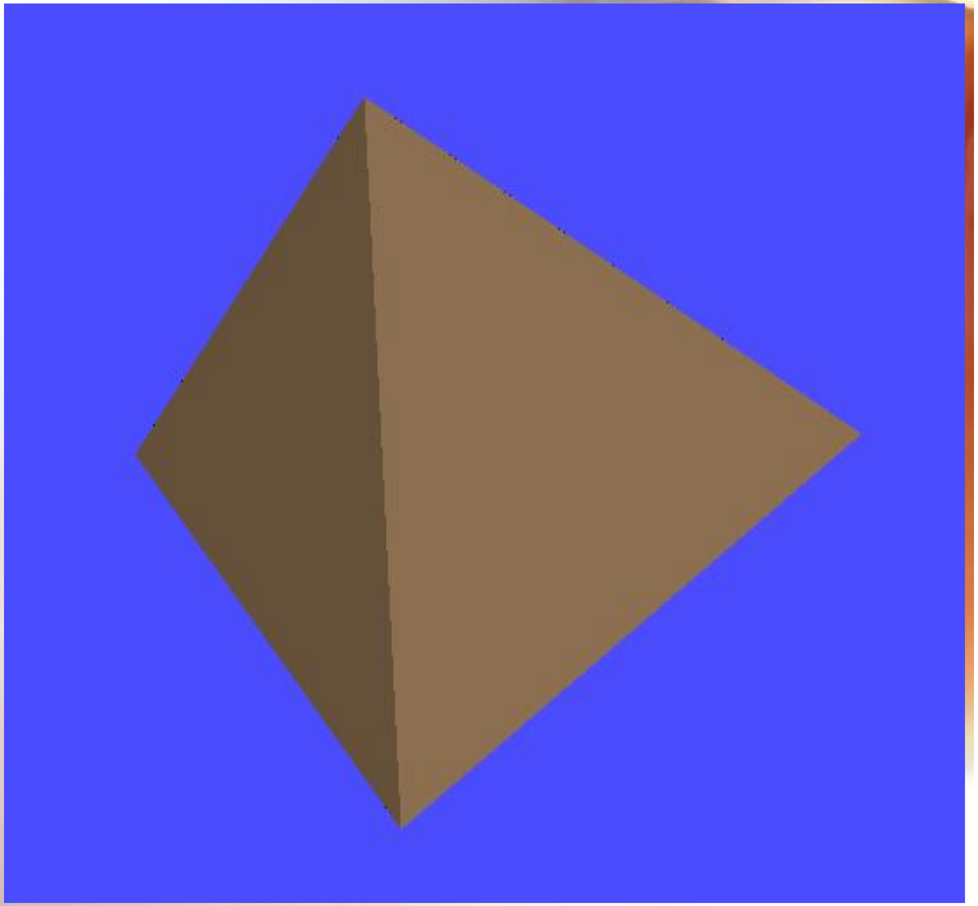
Flat Shading



- `glShadeModel(GL_FLAT);`
- When n is also constant over the entire polygon.
- Shading is only calculated once for each polygon.
- Efficient, but shows too much (too abrupt) shading difference between adjacent polygons.

Flat Cow





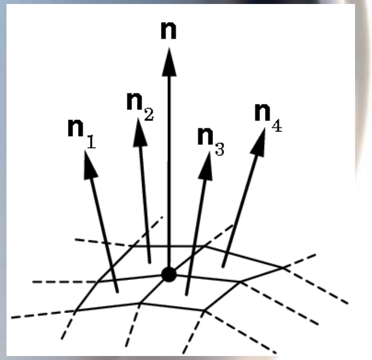
Shading to Fake Surface Curvature on Polygon Models

- Fake curvature due to neighboring polygons
  - Often make edge type dependent on dihedral angle between polygon faces:
  - If around 180° plus/minus a threshold (say 15°), shade smoothly.
  - Otherwise treat as sharp (defined) edge.

"hard" edges

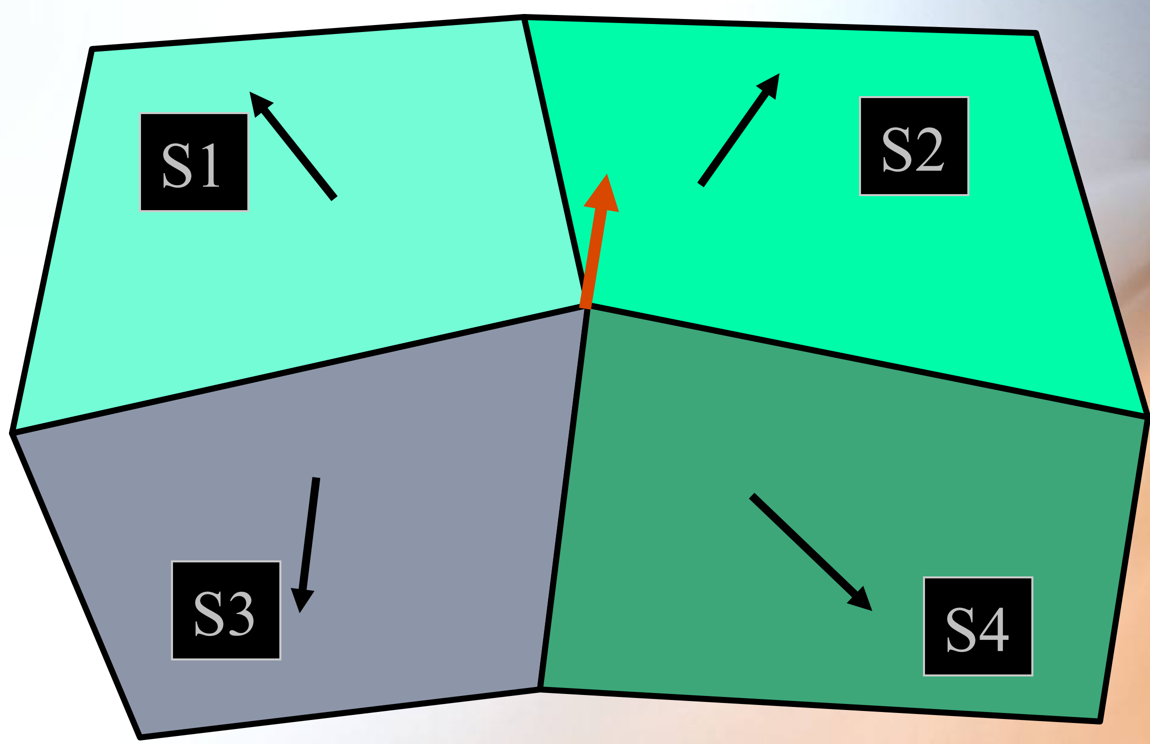
"soft" edges

Gouraud Shading



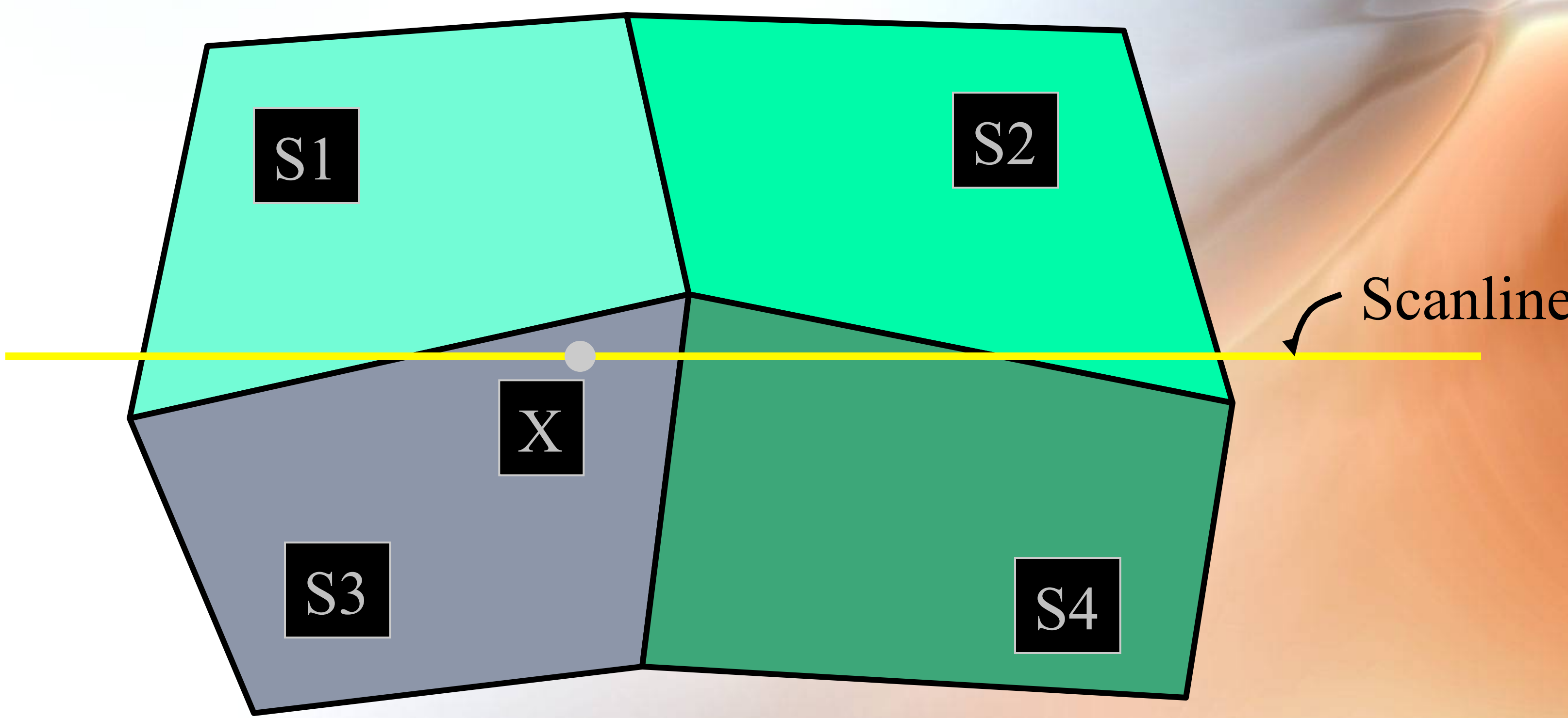
- **`glShadeModel(GL_SMOOTH);`**
- Also known as interpolative shading.
- Only effective if true vertex normals are given, i.e. the vertices of the polygon all have different normals.
- Gouraud shading defines a vertex normal to be the (normalized) average of normals of all polygons sharing this vertex.

Interpolating Normals

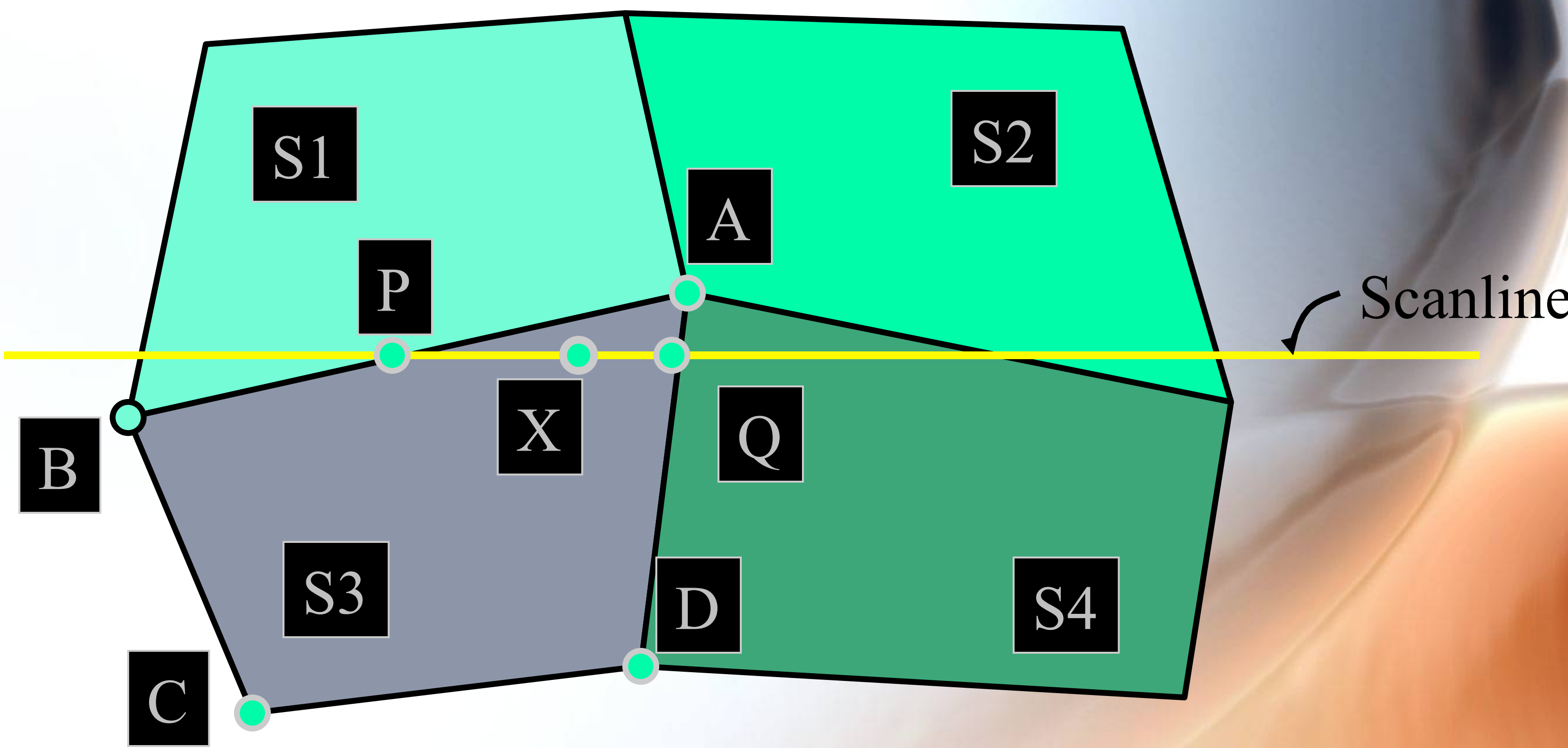


= ***Vertex Normal*** = average of neighboring normals.
Note that normals do not formally exist at vertices or along edge.

GOURAUD Shading



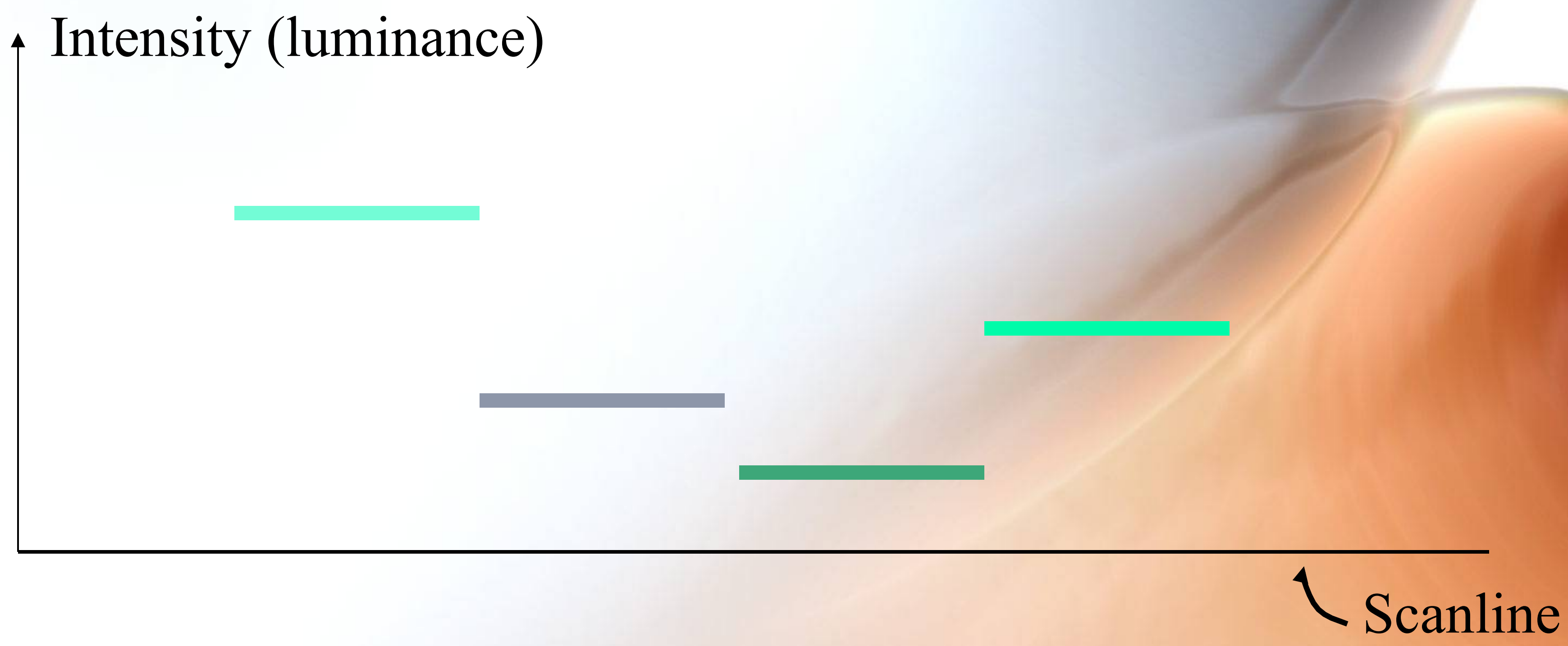
What is the Gouraud shade at X?



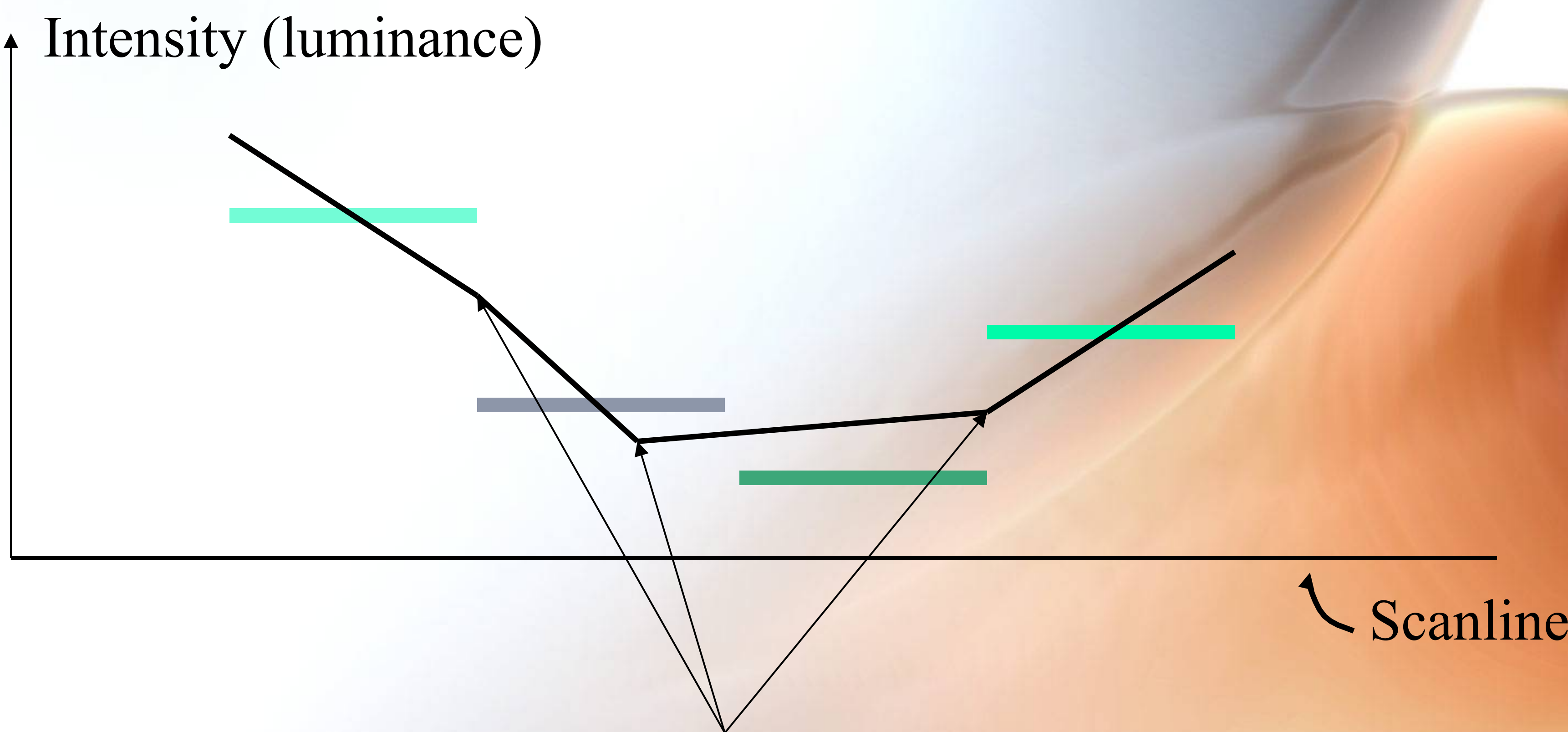
- Shade(A) = 1/4 ($n_{S1} + n_{S2} + n_{S3} + n_{S4}$)
- Similar computations for Shades at B, C, and D.
- Shade(P) = weighted average of Shade(A) and Shade(B)
- Shade(Q) = weighted average of Shade(A) and Shade(D)

- Shade(X) = weighted average of Shade(P) and Shade(Q)

The colors before interpolation along the scanline

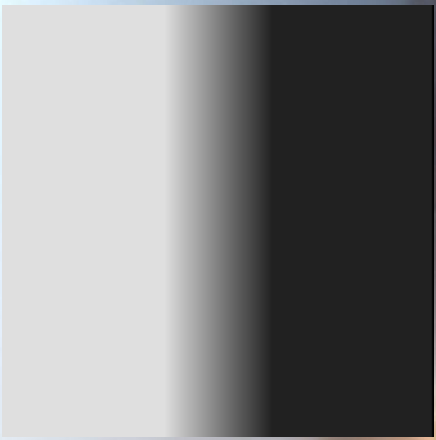


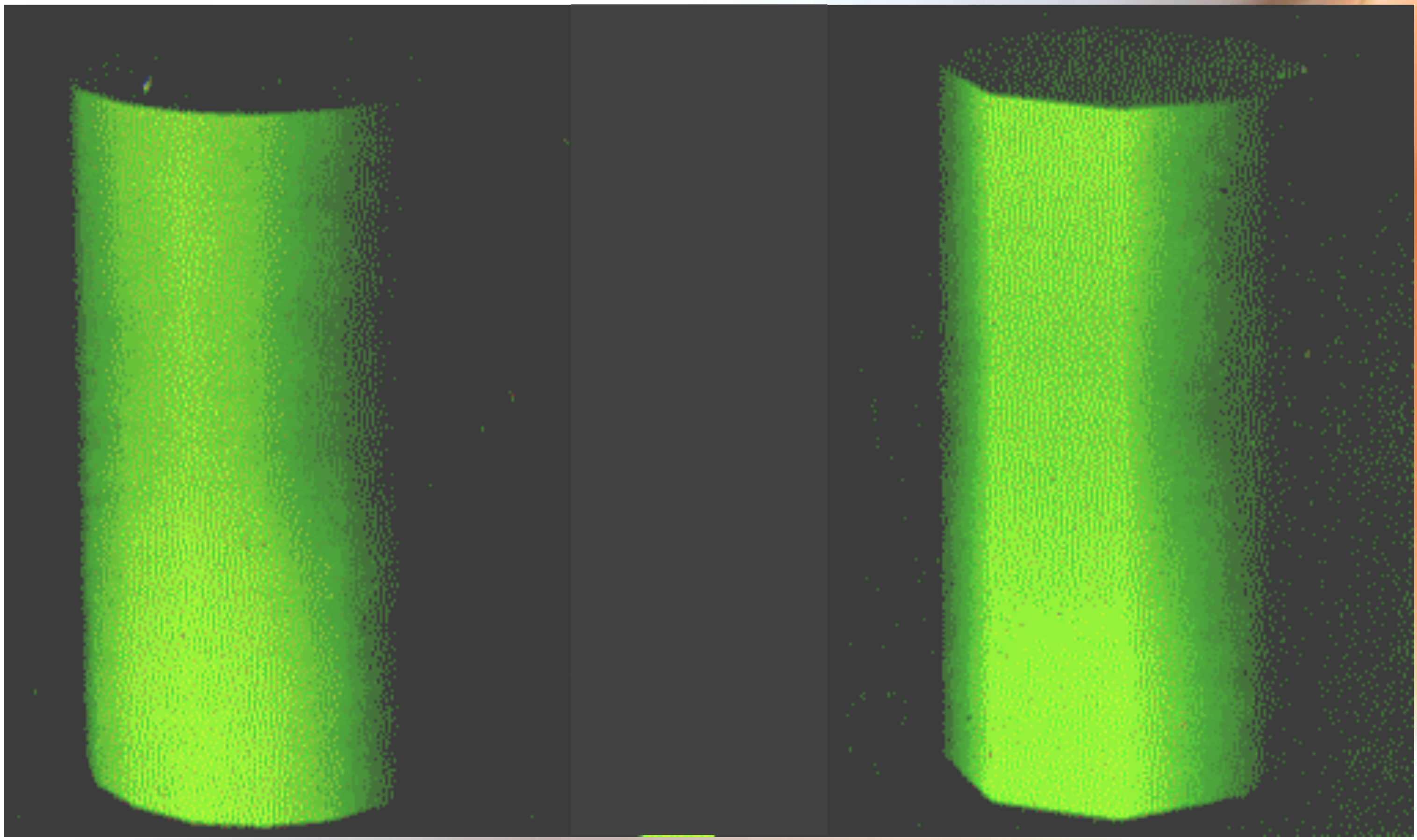
After linear interpolation along the scanline



Because we interpolate linearly, we get smooth ramps BUT *DISCONTINUOUS FIRST DERIVATIVES* at every edge: creating MACH Bands!

Mach bands



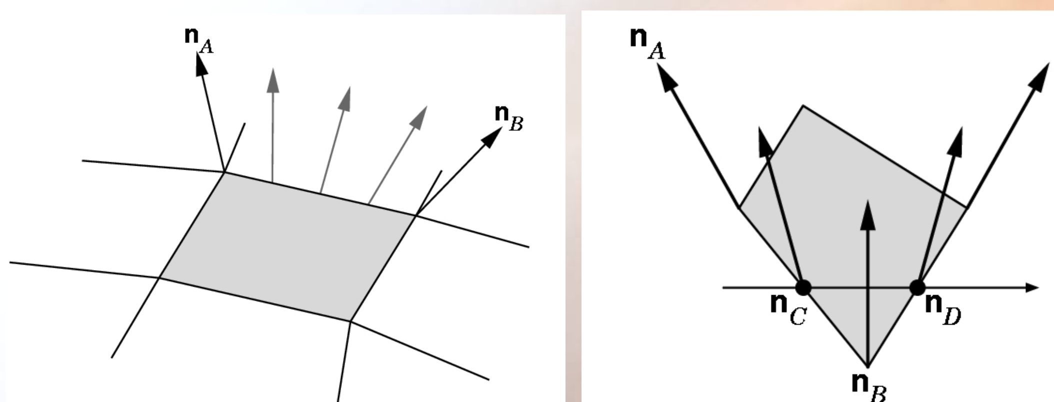


Gouraud and Phong Shading

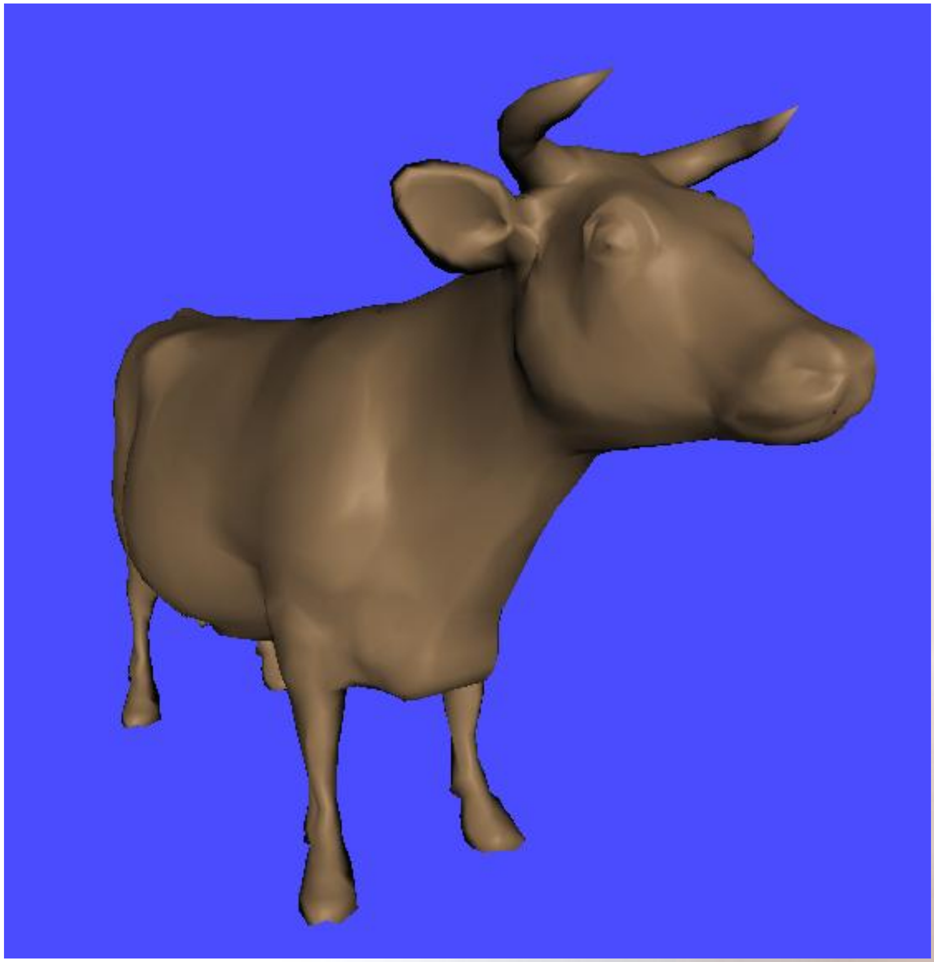
- Gouraud Shading
  - Find average normal at each vertex (vertex normals)
  - Apply Phong model at each vertex
  - Interpolate vertex shades across each polygon
- Phong shading
  - Find vertex normals
  - Interpolate vertex normals across edges
  - Find shades along edges
  - Interpolate edge shades across polygons

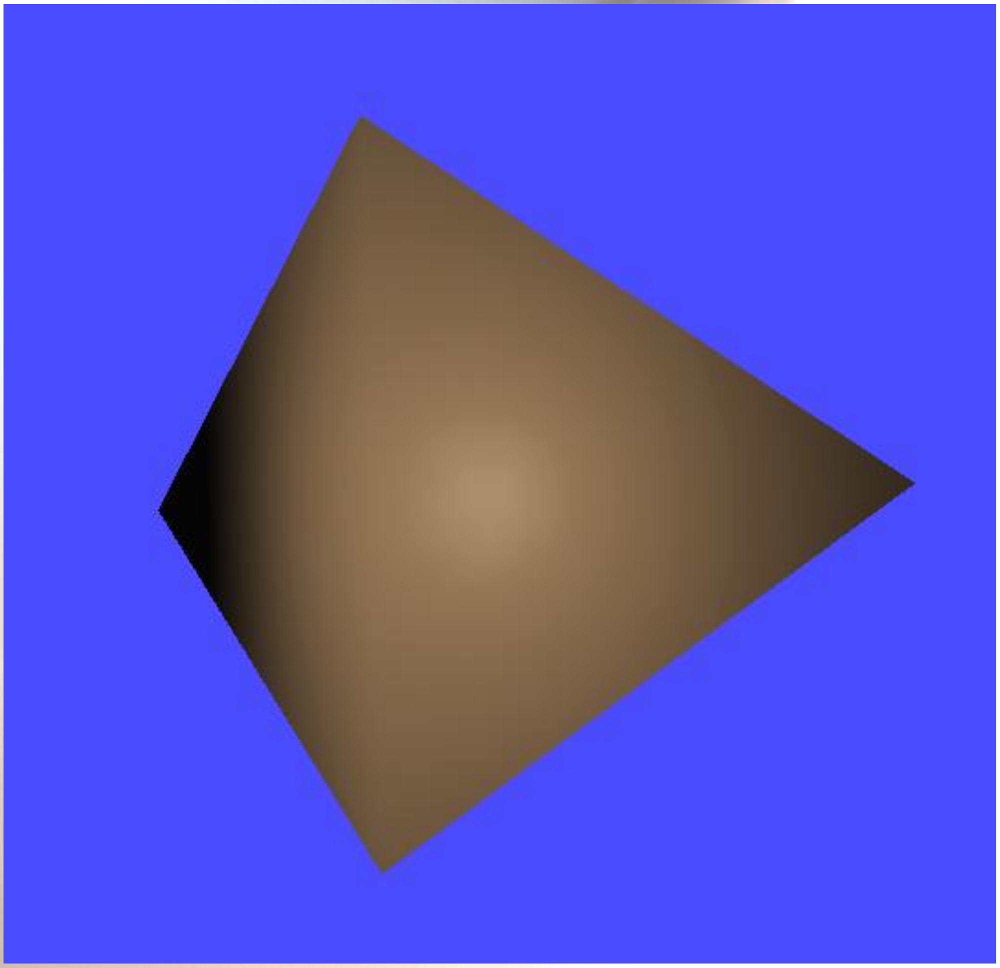
Phong Shading

- **Interpolate normals at vertices and then edges, then at every interior point**
- **Then make independent shading calculation based on each point's normal.**

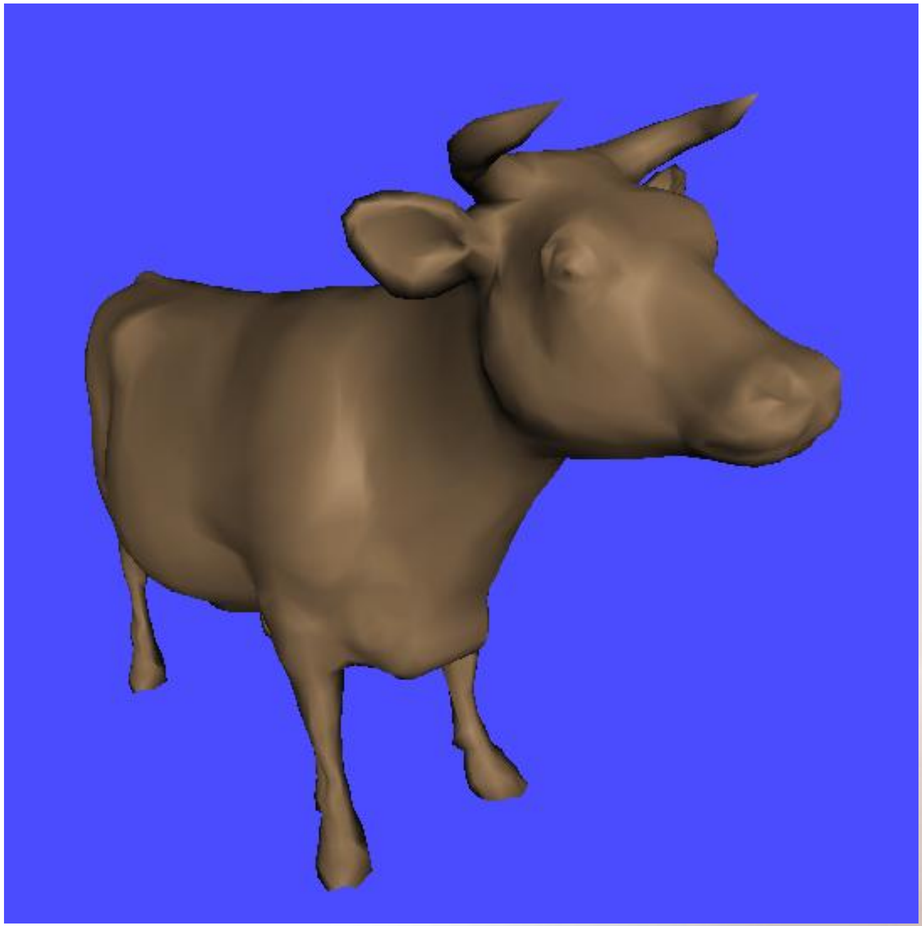


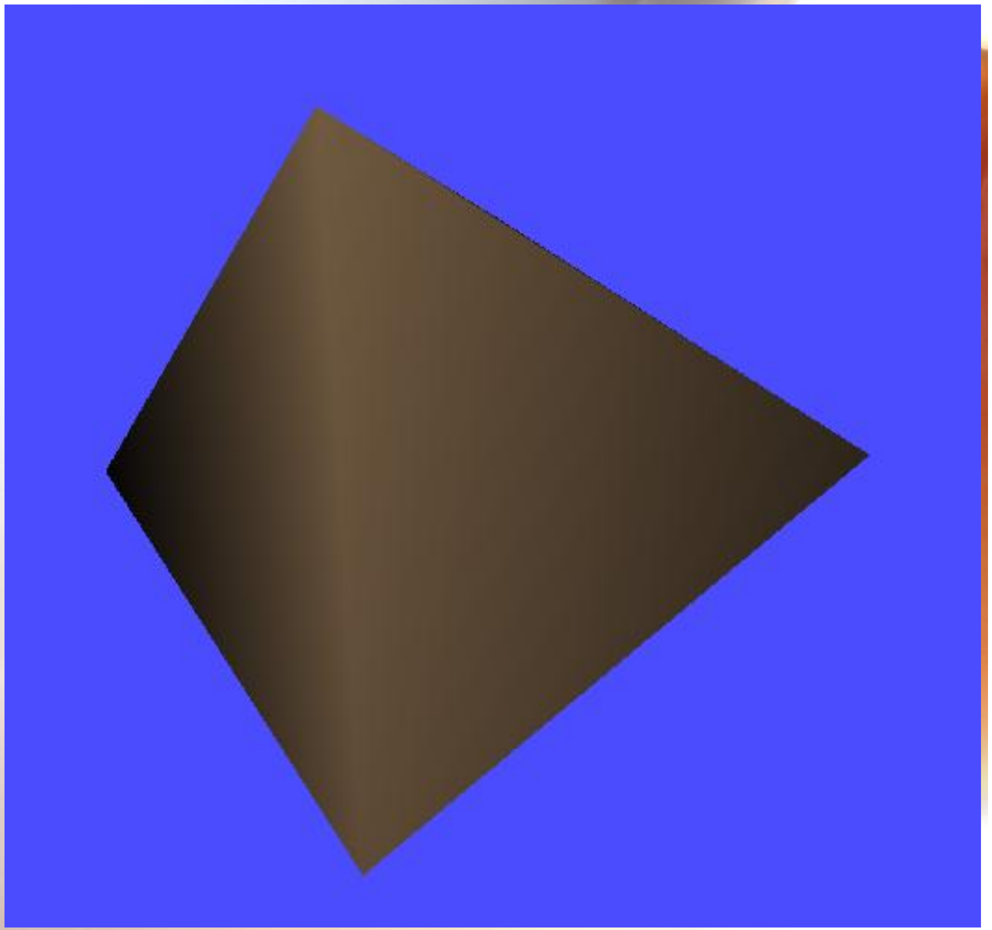
Phong Cow



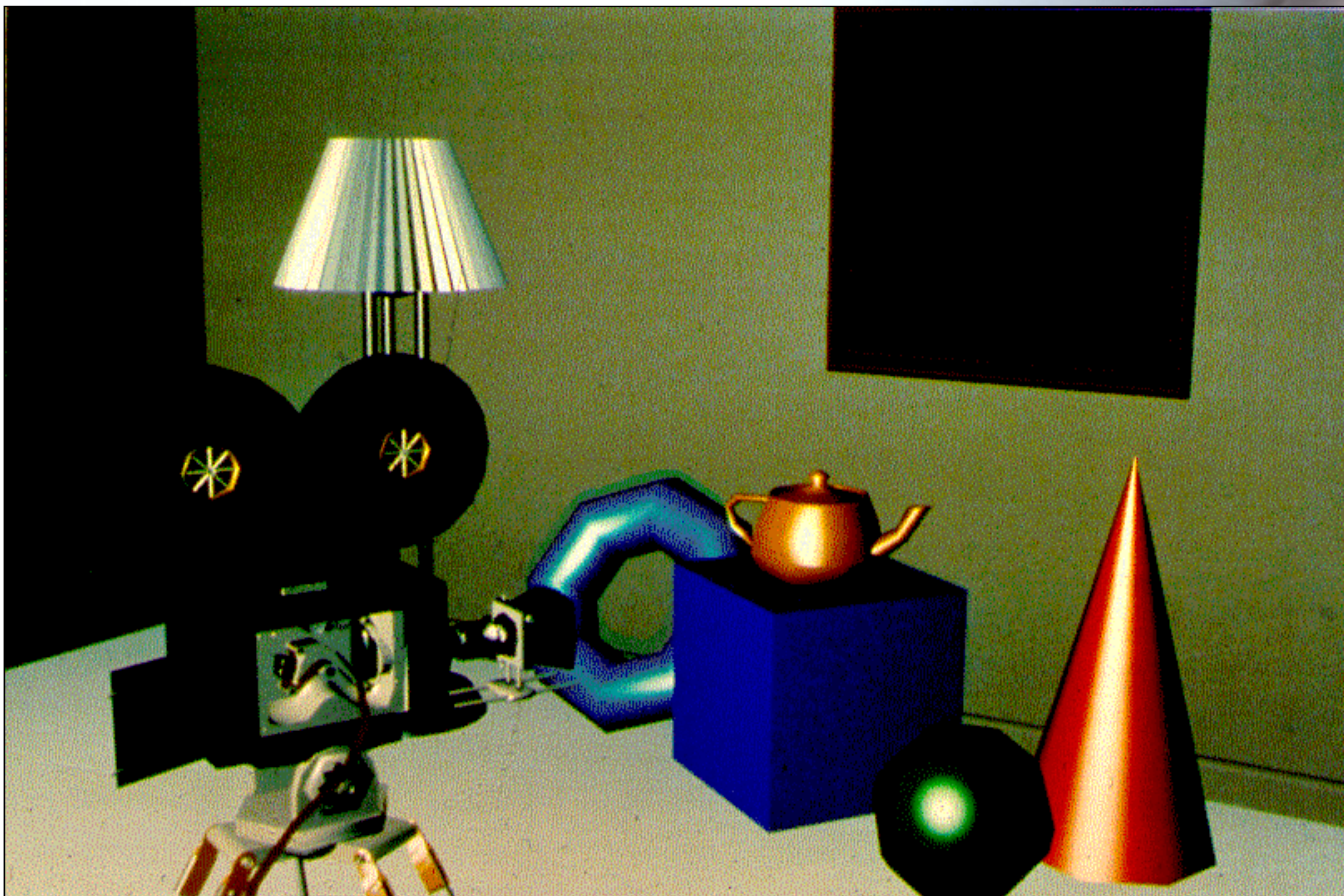


Gouraud Cow



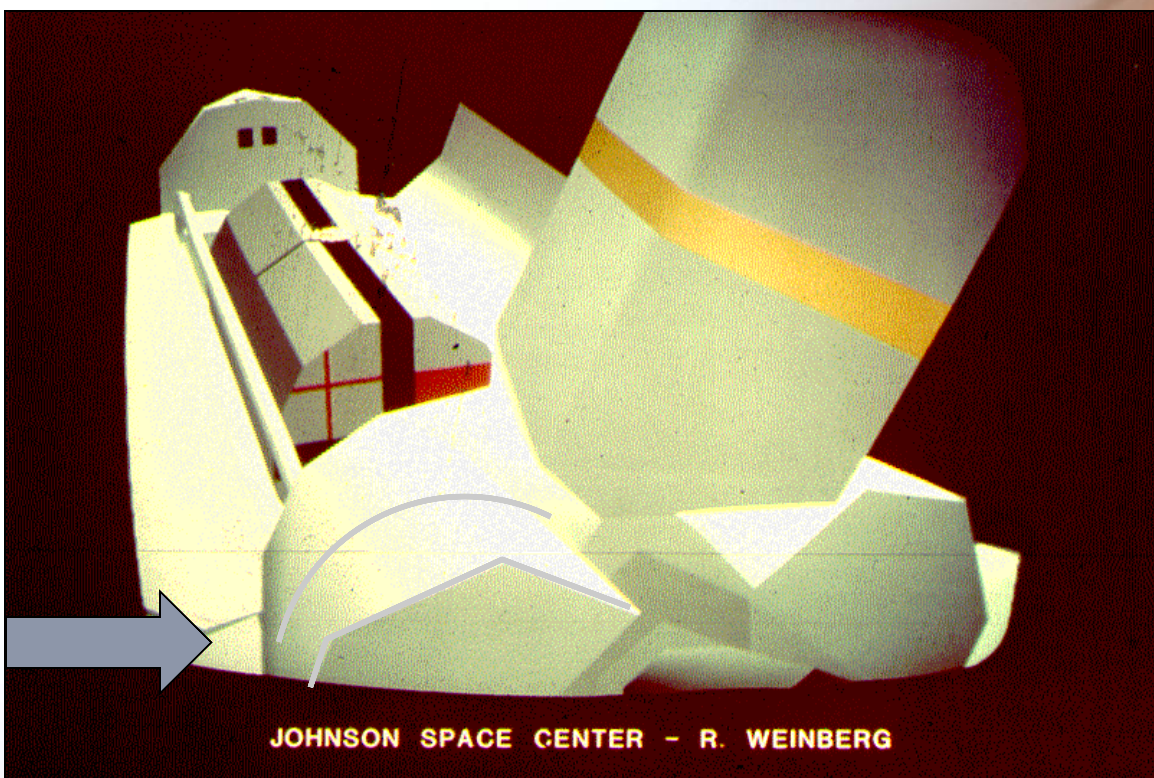


Phong Shaded Polygons

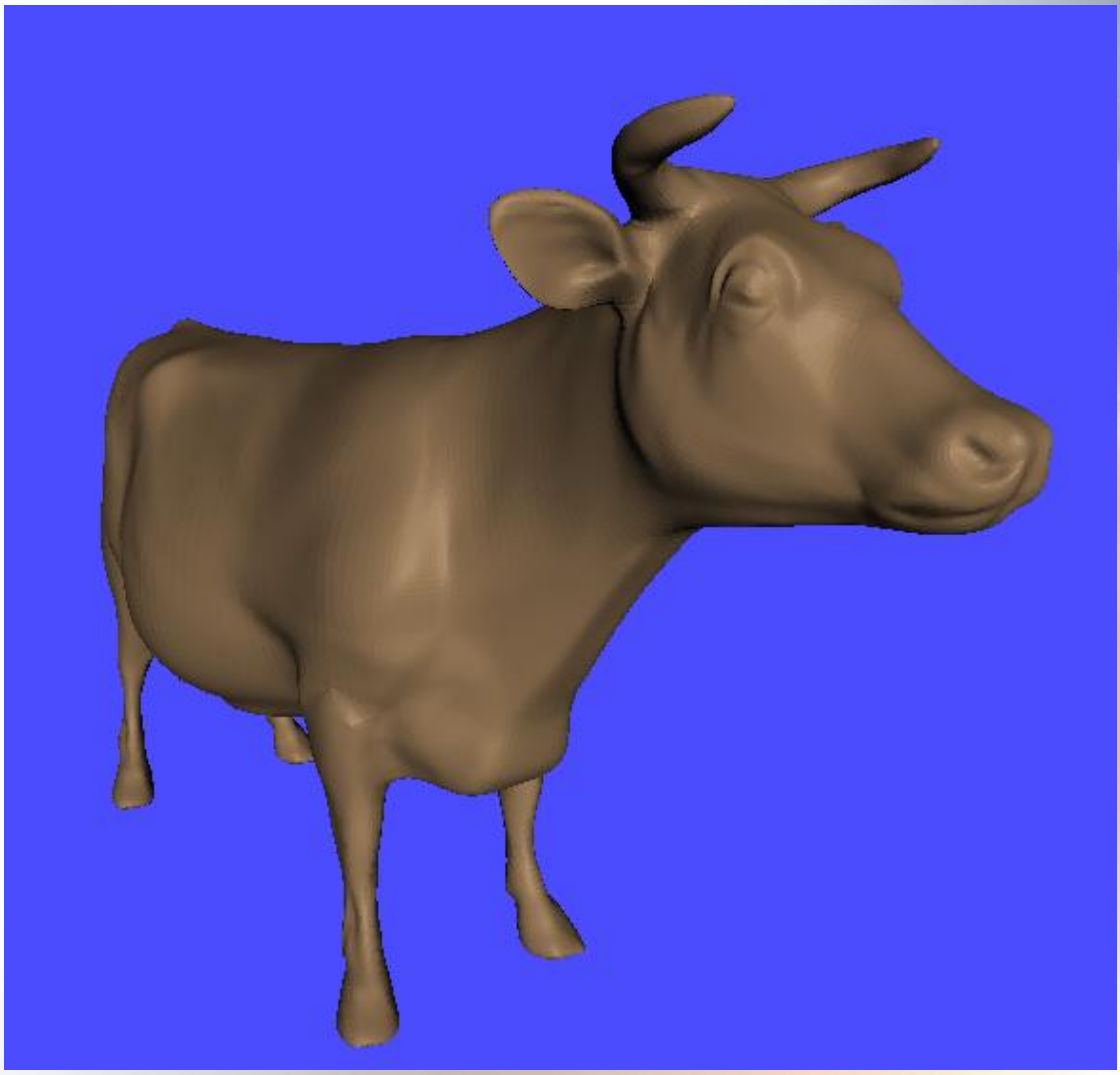


Contour Edge Problem

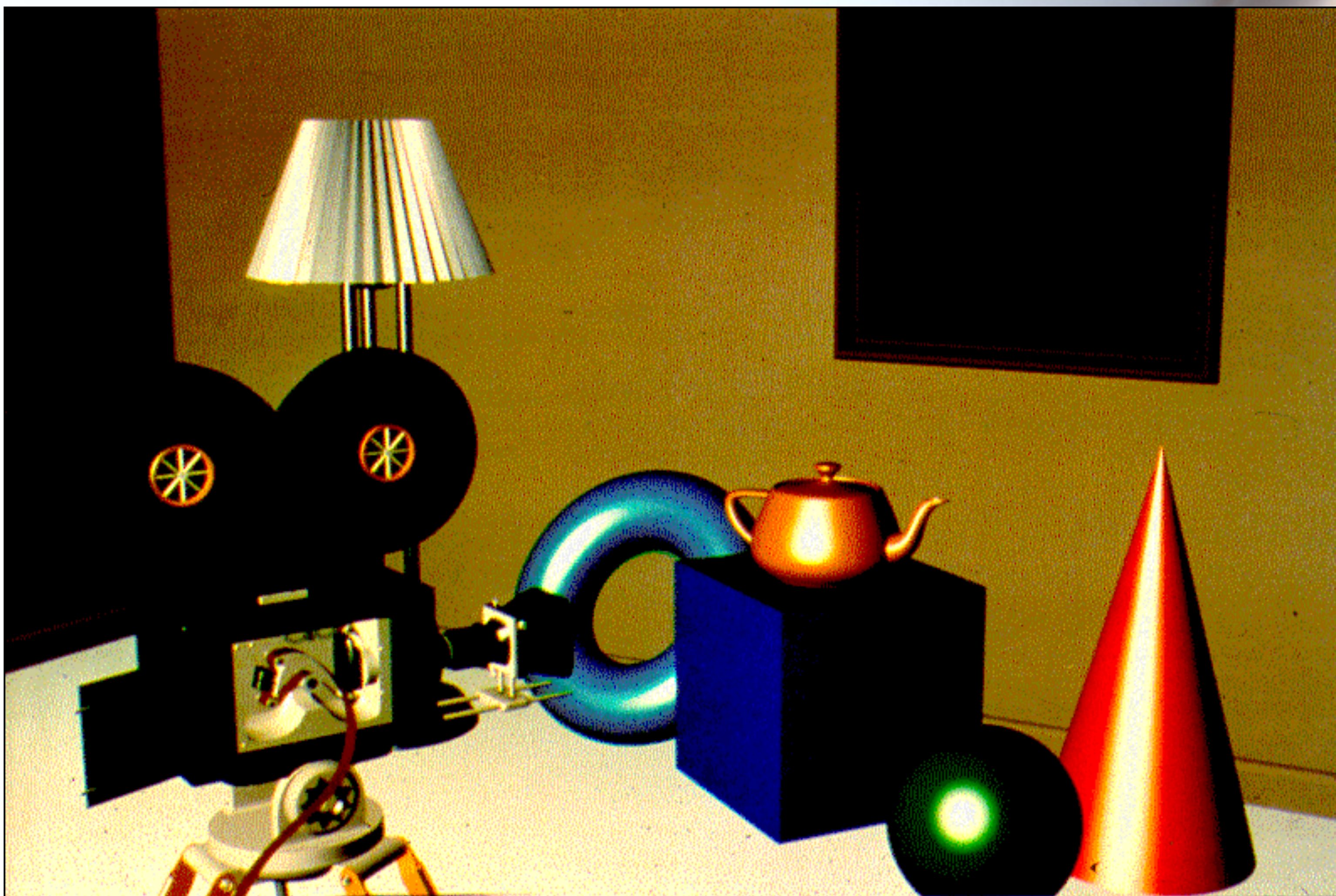
- No shading can possibly change the underlying polygon model.



Cow Times 16



Polygons Interpreted as Curved Surface Control Mesh



Multiple Light Sources, Curved Surfaces, and Phong Shading

Teapots!

Only differences in these teapots are the parameters in the Phong model