WPI

CS 543:
Computer Graphics

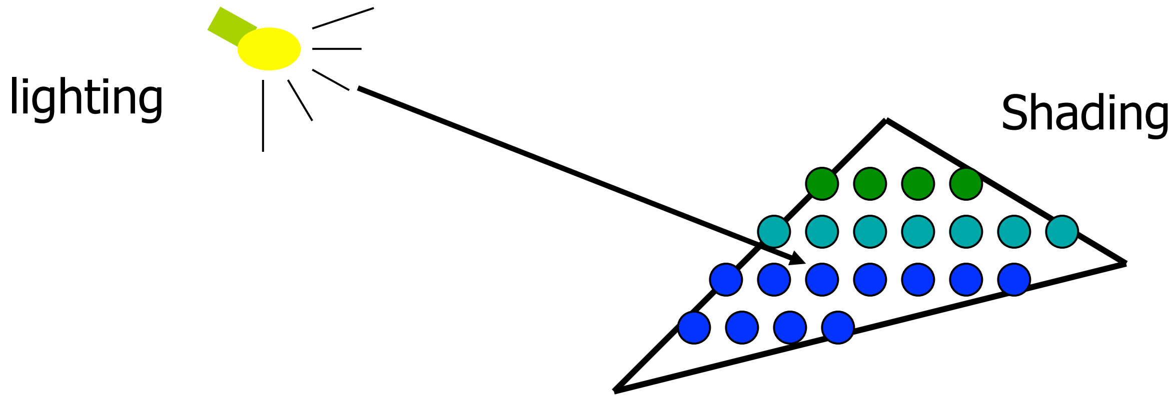
Illumination & Shading I

Robert W. Lindeman
Associate Professor
Interactive Media & Game Development
Department of Computer Science
Worcester Polytechnic Institute
gogo@wpi.edu

(with lots of help from Prof. Emmanuel Agu :-)

Illumination and Shading

- Problem: Model light/surface point interactions to determine final color and brightness
- Apply the lighting model at a set of points across the entire surface

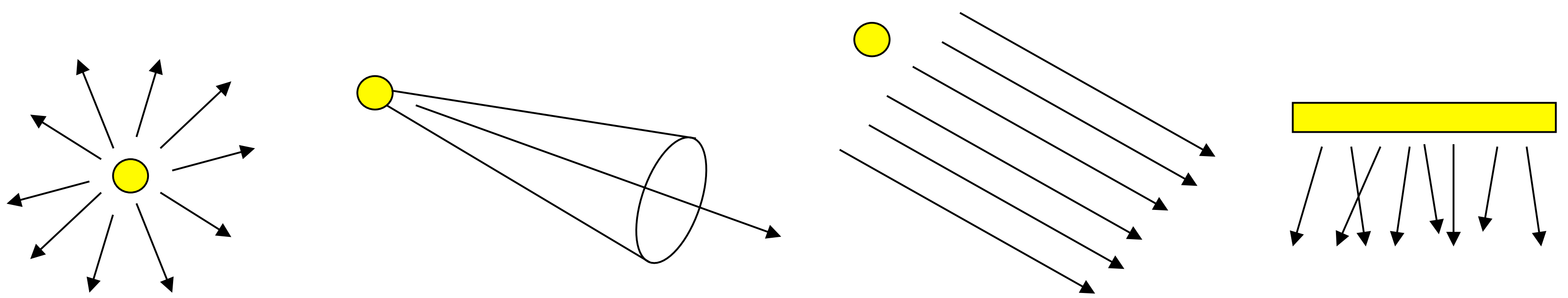


Illumination Model

- The governing principles for computing the illumination
- An illumination model usually considers
  - Light attributes (intensity, color, position, direction, shape)
  - Object surface attributes (color, reflectivity, transparency, *etc.*)
  - Interaction among lights and objects

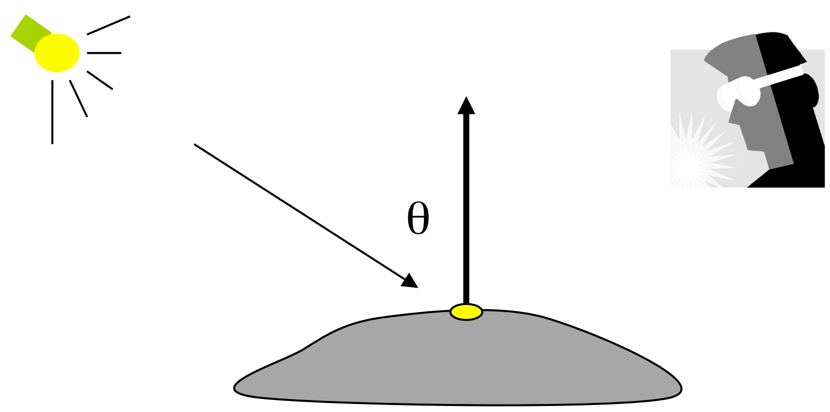
Basic Light Sources

- Light intensity can be independent or dependent of the distance between object and the light source



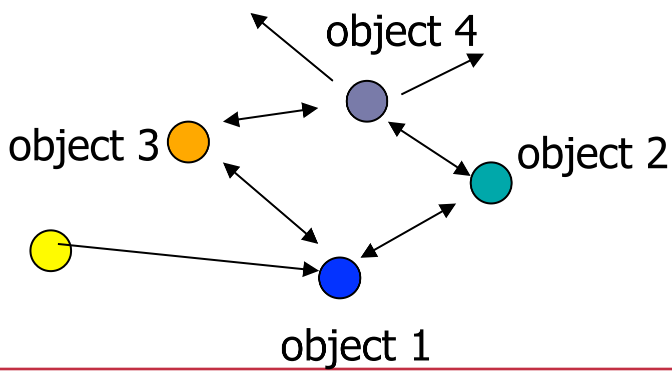
Local Illumination

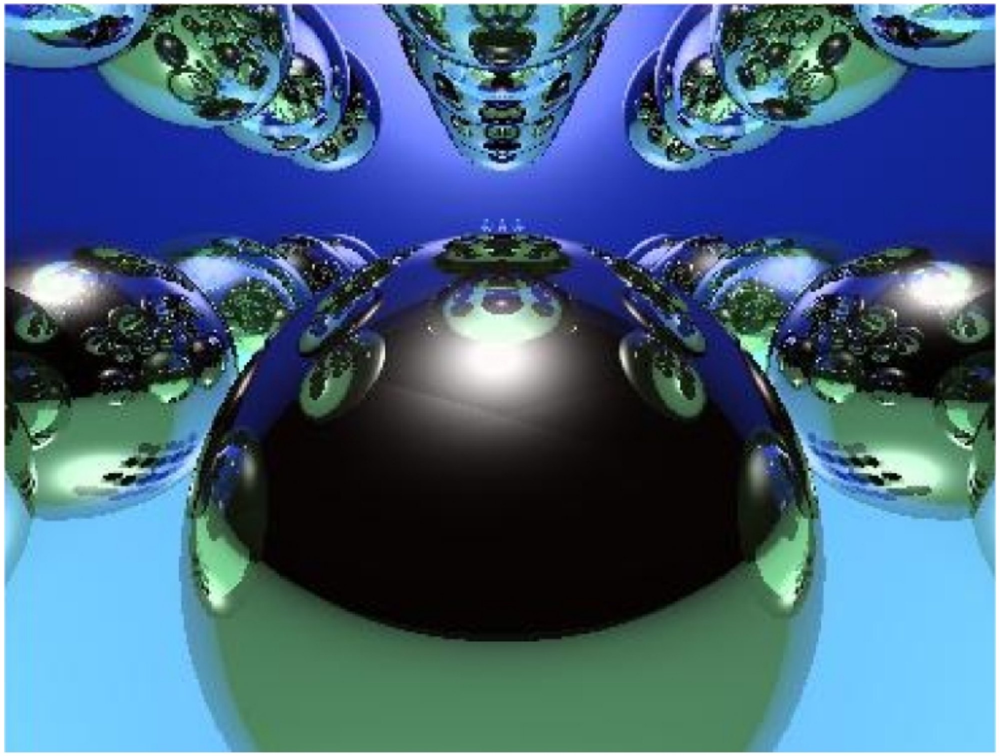
- Only consider the light, the observer position, and the object material properties
- Almost all renderers do at least this



Global Illumination

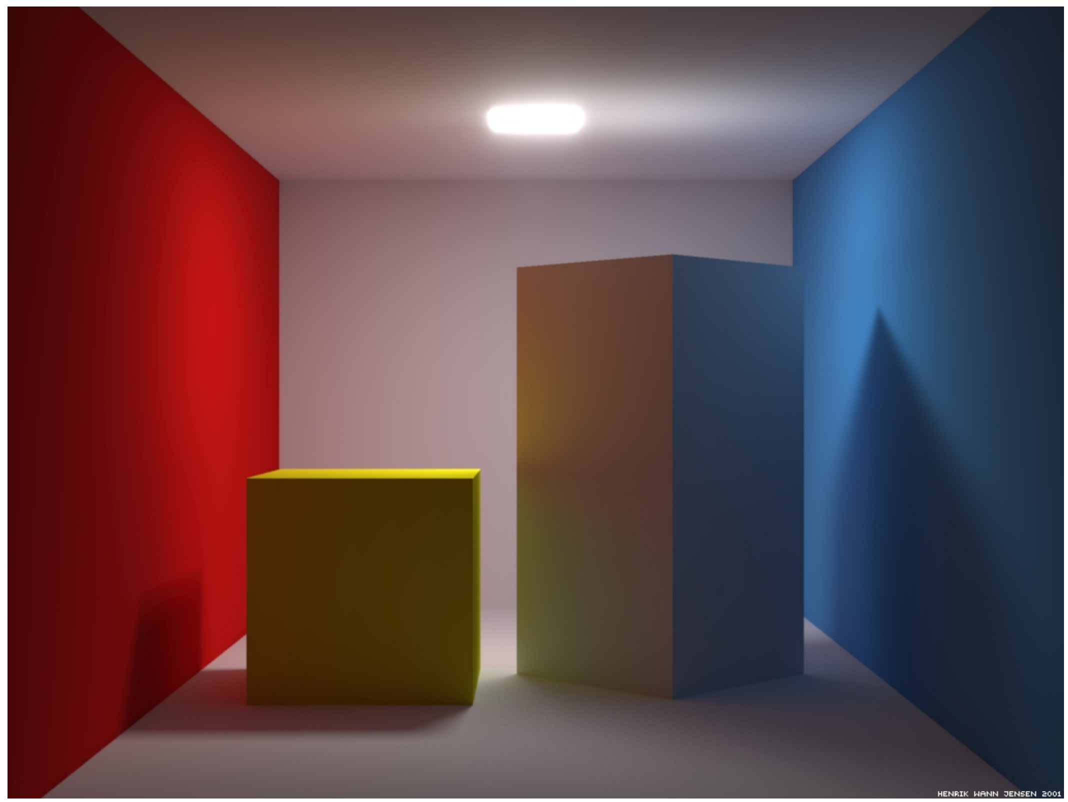
- Take into account the interaction of light from all the surfaces in the scene
- Example:
  - Ray Tracing
  - Model light rays bouncing around





Global Illumination (cont.)

- Example:
  - Radiosity
    - Model *energy* moving from emitters (e.g., lights) into the scene
    - View independent

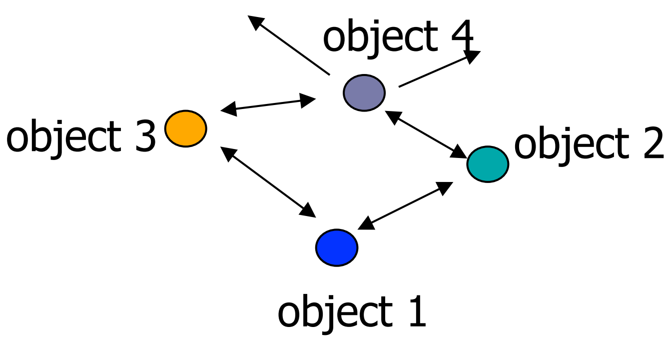


Simple Local Illumination

- The model used by OpenGL
- Reduce the complex workings of light to three components
  - Ambient
  - Diffuse
  - Specular
- Final illumination at a point (vertex) = ambient + diffuse + specular
- Materials reflect each component differently
  - Use different material reflection coefficients
    - K_a, K_d, K_s

Ambient Light Contribution

- Ambient light = background light
- Light that is scattered by the environment
  - It's just there
- **Frequently assumed to be constant**
- Very simple approximation of global illumination
- No direction: independent of light position, object orientation, observer's position/orientation

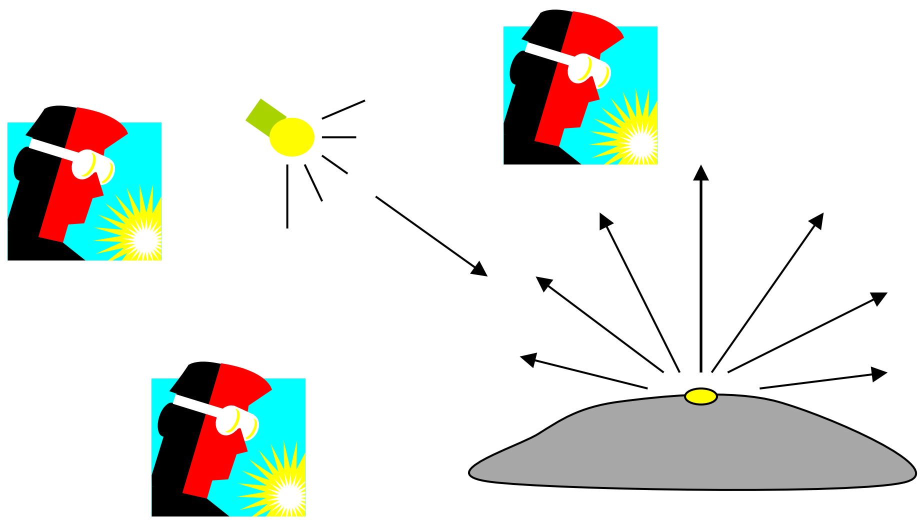


constant

$$\text{Ambient} = I \times K_a$$

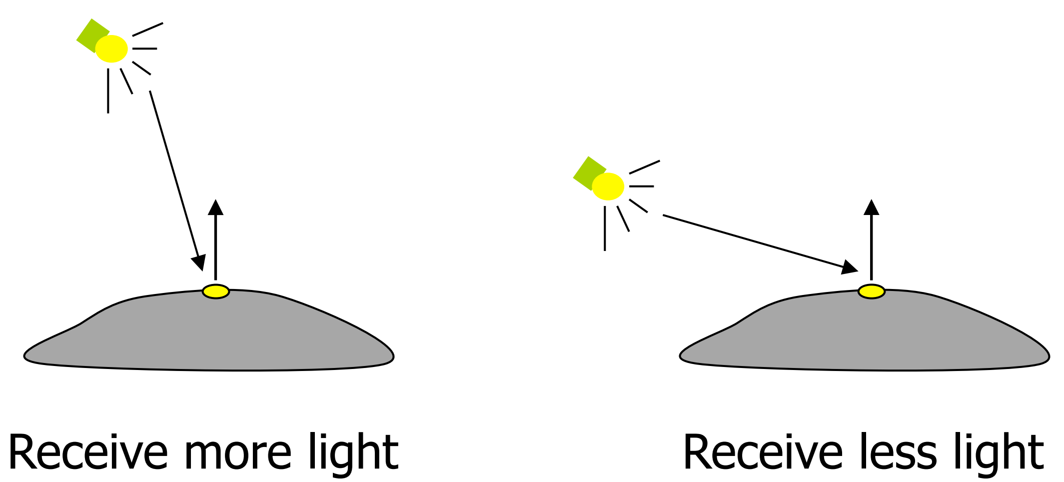
Diffuse Light Contribution

- Diffuse light: The illumination that a surface receives from a light source that reflects equally in all directions
  - Eye point does not matter



Diffuse Light Calculation

- Need to decide how much light the object point receives from the light source
  - Based on Lambert's Law



Diffuse Light Calculation (cont.)

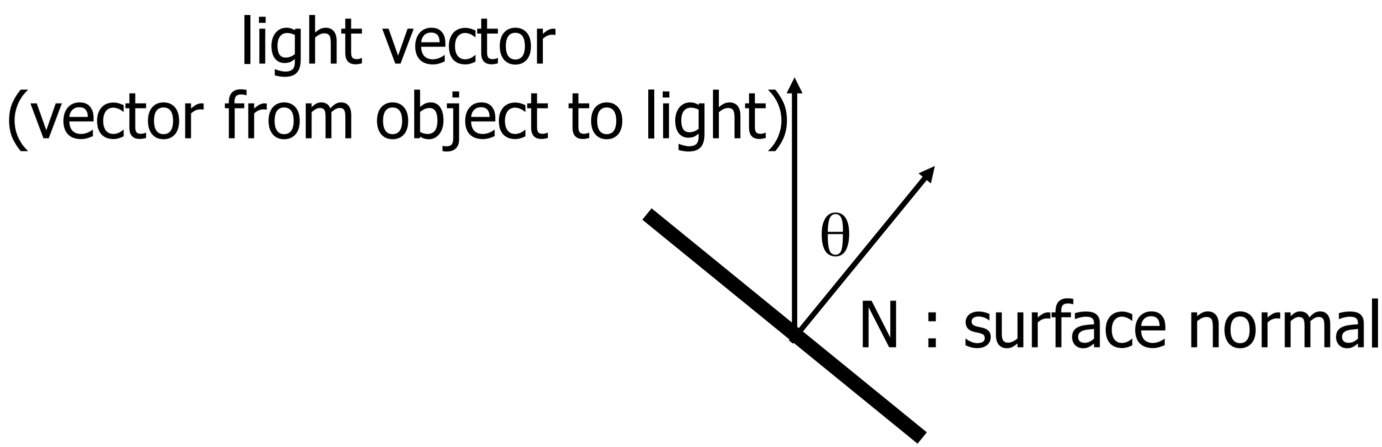
- Lambert's law: the radiant energy D that a small surface patch receives from a light source is:

$$Diffuse = K_d \times I \times \cos(\theta)$$

K_d: diffuse reflection coefficient

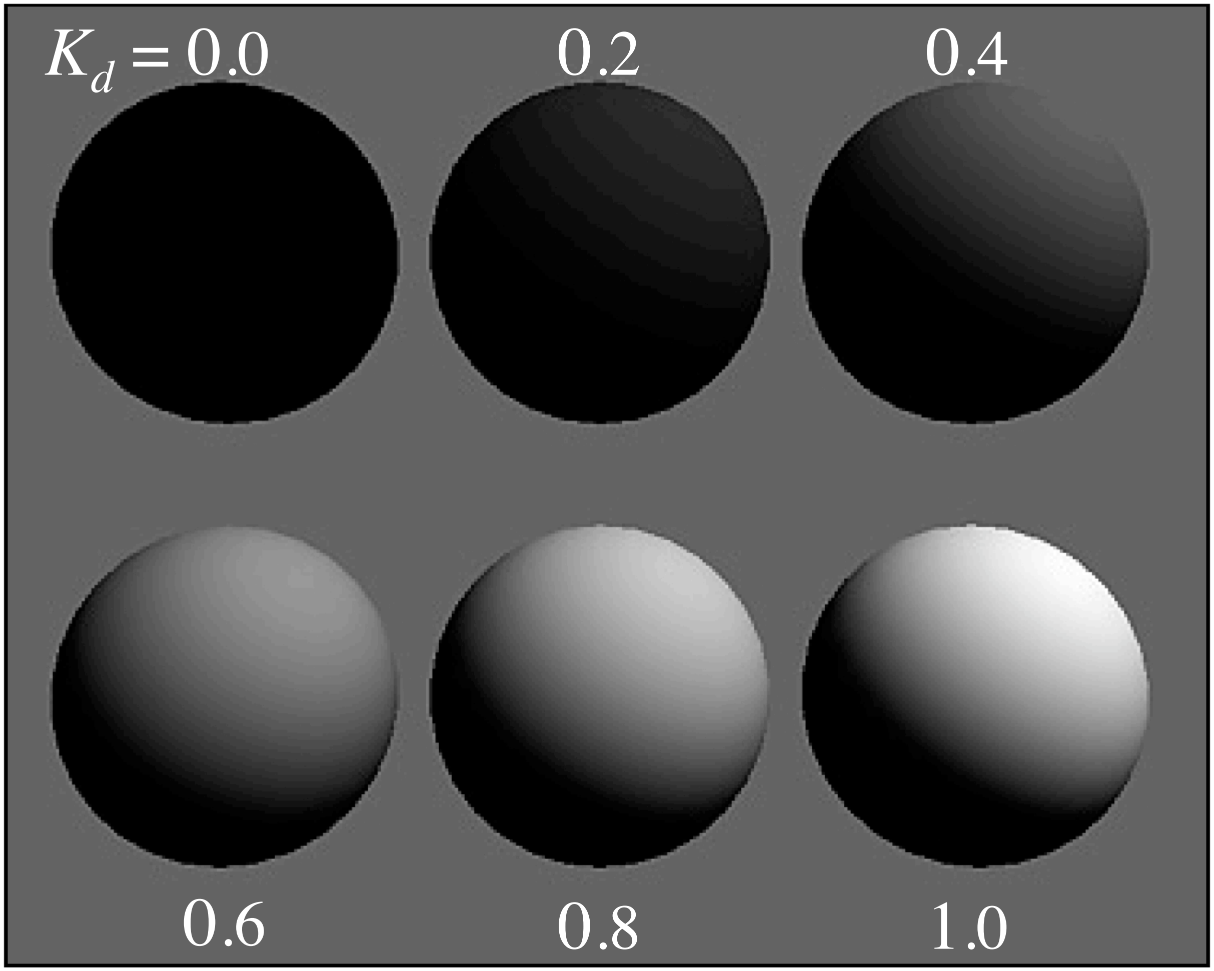
I: light intensity

θ: angle between the light vector and the surface normal



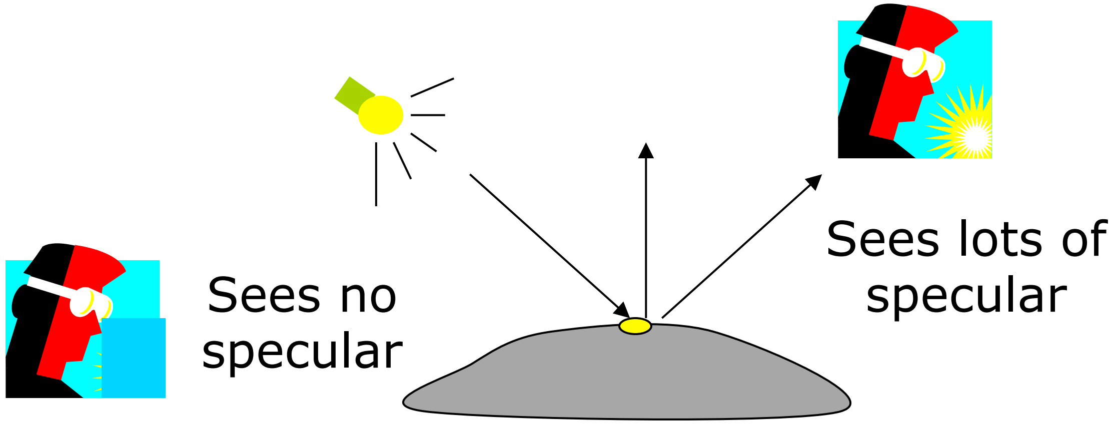
Diffuse Light Examples

I = 1.0



Specular Light Contribution

- The bright spot on the object
- The result of total reflection of the incident light in a concentrate region

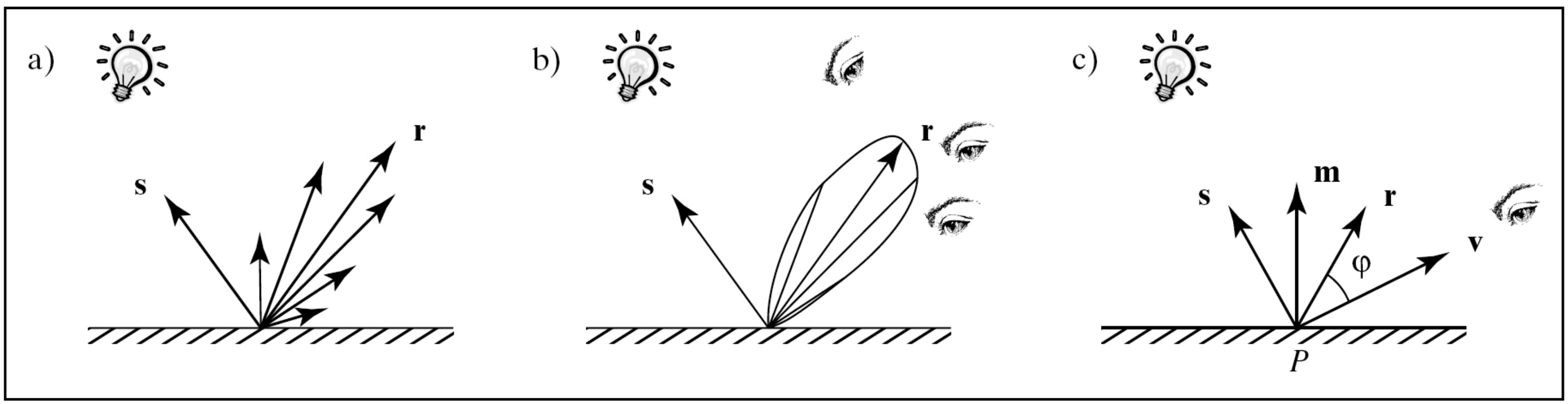


Specular Light Calculation

- How much reflection you can see depends on where you are
  - But for non-perfect surface you will still see specular highlight when you move a little bit away from the ideal reflection direction

  Φ is deviation of view angle from mirror direction

  - When ϕ is small, you see more specular highlight



Specular Light Calculation (cont.)

- Phong lighting model
  - Not Phong *shading* model

$$\text{Specular} = K_s \times I \times \cos^{f}(\phi)$$

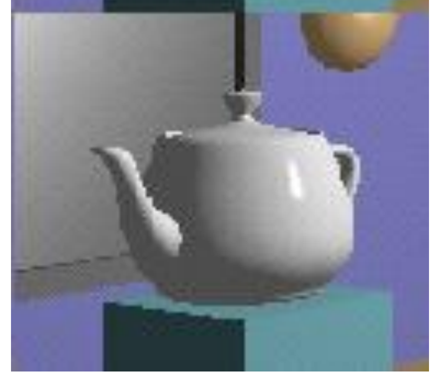
- The effect of 'f' in the Phong model

f = 10

f = 30

f = 90

f = 270

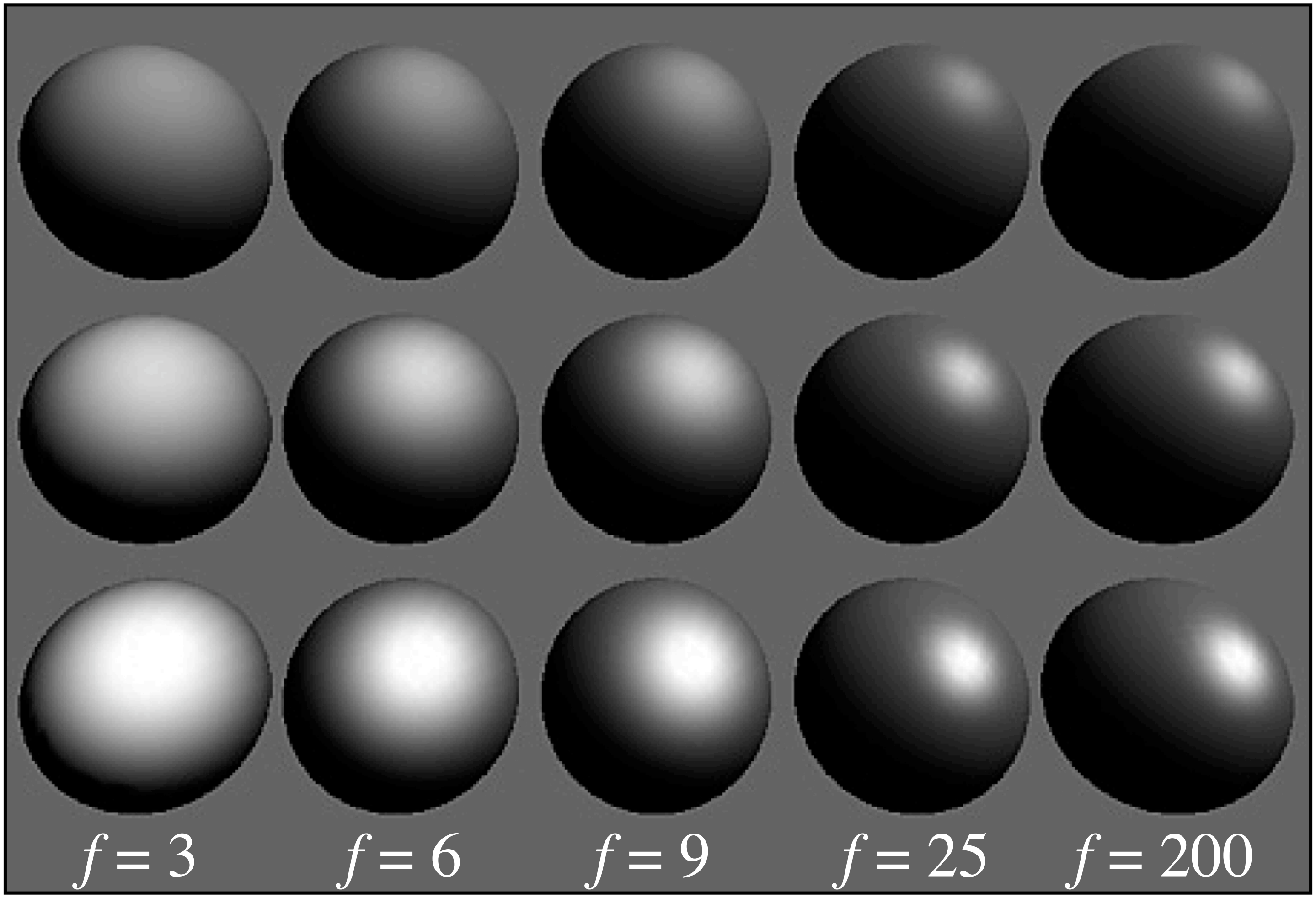


Specular Light Examples

$K_s = 0.25$

$K_s = 0.5$

$K_s = 0.75$



Putting It All Together

- Illumination from a light

  Illum = ambient + diffuse + specular

  $$= K_a \times I + K_d \times I \times \cos(\theta) + K_s \times I \times \cos^f(\phi)$$

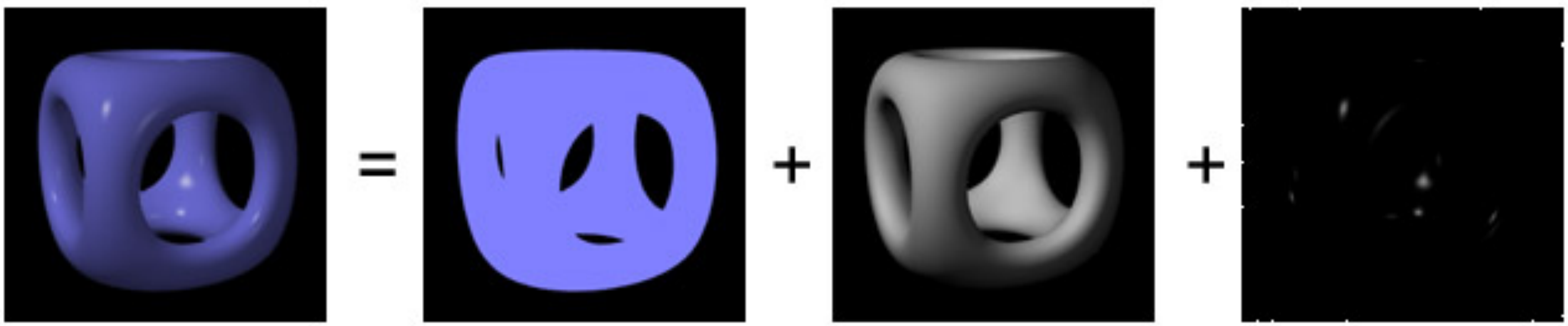
- If there are N lights

  Total illumination for a point P = Σ (Illum)

- Some more terms to be added
  - Self emission
  - Global ambient
  - Light distance attenuation and spot light effect

Putting It All Together (cont.)

- **Illum = ambient + diffuse + specular**



color and ambient

diffuse

specularity

Ambient Lighting Example

Diffuse Lighting Example

Specular Lighting Example

Adding Color

- Sometimes light or surfaces are colored
- Treat R, G and B components separately
  - *i.e.*, can specify different RGB values for either light or material
- Illumination equation goes from

Illum = ambient + diffuse + specular

$$= K_a \times I + K_d \times I \times \cos(\theta) + K_s \times I \times \cos^f(\phi)$$

To:

$$Illum_r = K_{ar} \times I_r + K_{dr} \times I_r \times \cos(\theta) + K_{sr} \times I_r \times \cos^f(\phi)$$

$$Illum_g = K_{ag} \times I_g + K_{dg} \times I_g \times \cos(\theta) + K_{sg} \times I_g \times \cos^f(\phi)$$

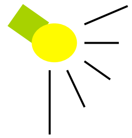
$$Illum_b = K_{ab} \times I_b + K_{db} \times I_b \times \cos(\theta) + K_{sb} \times I_b \times \cos^f(\phi)$$

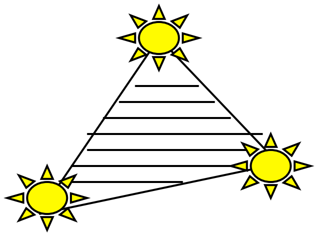
Adding Color (cont.)

Material	Ambient K_{ar}, K_{ag}, K_{ab}	Diffuse K_{dr}, K_{dg}, K_{db}	Specular K_{sr}, K_{sg}, K_{sb}	Exponent *f*
Black plastic	0.0 0.0 0.0	0.01 0.01 0.01	0.5 0.5 0.5	32
Brass	0.329412 0.223529 0.027451	0.780392 0.568627 0.113725	0.992157 0.941176 0.807843	27.8974
Polished Silver	0.23125 0.23125 0.23125	0.2775 0.2775 0.2775	0.773911 0.773911 0.773911	89.6

Lighting Steps

- Adopt Phong lighting model
  - Ambient + Specular + Diffuse lights
  - Lighting is computed at vertices
    - Interpolate across surface (Gouraud/smooth shading)
- Setting up lighting
  - Light properties
  - Surface material properties
  - Provide correct surface normals
  - Set light model properties

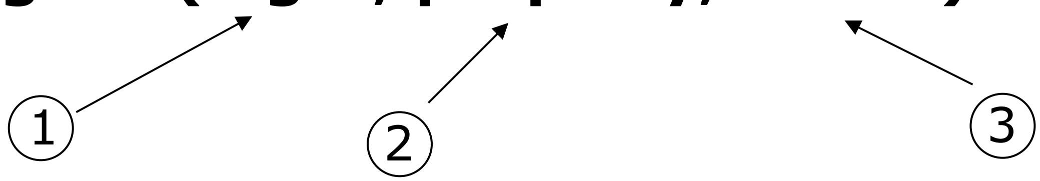




Light Properties

- Properties
  - Colors / Position and type / attenuation

glLightfv(light, property, value)



(1) Constant: specify which *light* you want to set the property for
e.g., `GL_LIGHT0, GL_LIGHT1, GL_LIGHT2 ...`
you can create multiple lights (OpenGL allows at least 8 lights)

(2) Constant: specify which light *property* you want to set
e.g., `GL_AMBIENT, GL_DIFFUSE, GL_SPECULAR, GL_POSITION`
(check the red book, or the Web, for more)

(3) The value you want to set to the property

Property Example

- Define colors and position a light

```
Var light_ambient[]  = { 0.0, 0.0, 0.0, 1.0 };
Var light_diffuse[]  = { 1.0, 1.0, 1.0, 1.0 };
Var light_specular[] = { 1.0, 1.0, 1.0, 1.0 };
Var light_position[] = { 0.0, 0.0, 1.0, 1.0 };


Pass along data to shaders…see book code
```

Colors

Position

What if I set Position to (0, 0, 1, 0)?

Types of Lights

- WebGL doesn‵t support lights, you provide your own (see book code):
  - Local light (point light)
  - Infinite light (directional light)
- WebGL determines the light type by:
  - w = 0: Infinite light source
  - w != 0:Point light
    - Position = (x/w, y/w, z/w)

```
vec3 light = lightPosition.xyz; //shader: already in world space
```

Controlling Light Position

- Modelview matrix affects a light's position
- Two options
  - Option a:
    - Treat light like *vertex*
    - Perform a **`translate`**, **`rotate on light position`**
    - Call **`LookAt`**
    - Light moves independently of camera
  - Option b:
    - Load identity matrix in modelview matrix
    - Call **`LookAt`**
    - Light appears at the eye (like a miner's lamp)

Material Example

- Define ambient/diffuse/specular reflection and shininess

```
var mat_amb_diff[] = { 1.0, 0.5, 0.8, 1.0 };
var mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
var shininess[] = { 5.0 };
```

Refl. coeff. (mat_amb_diff, mat_specular)

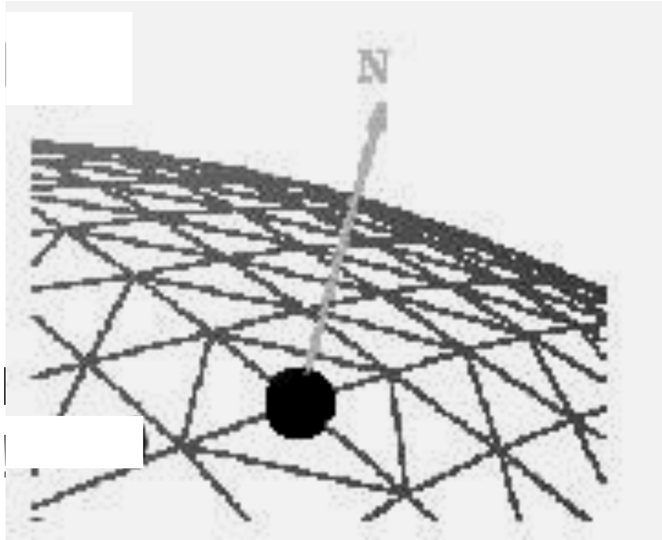
shininess — Range: dull 0 – very shiny 128

Pass to the shader… (see book code for details)

Surface Normals

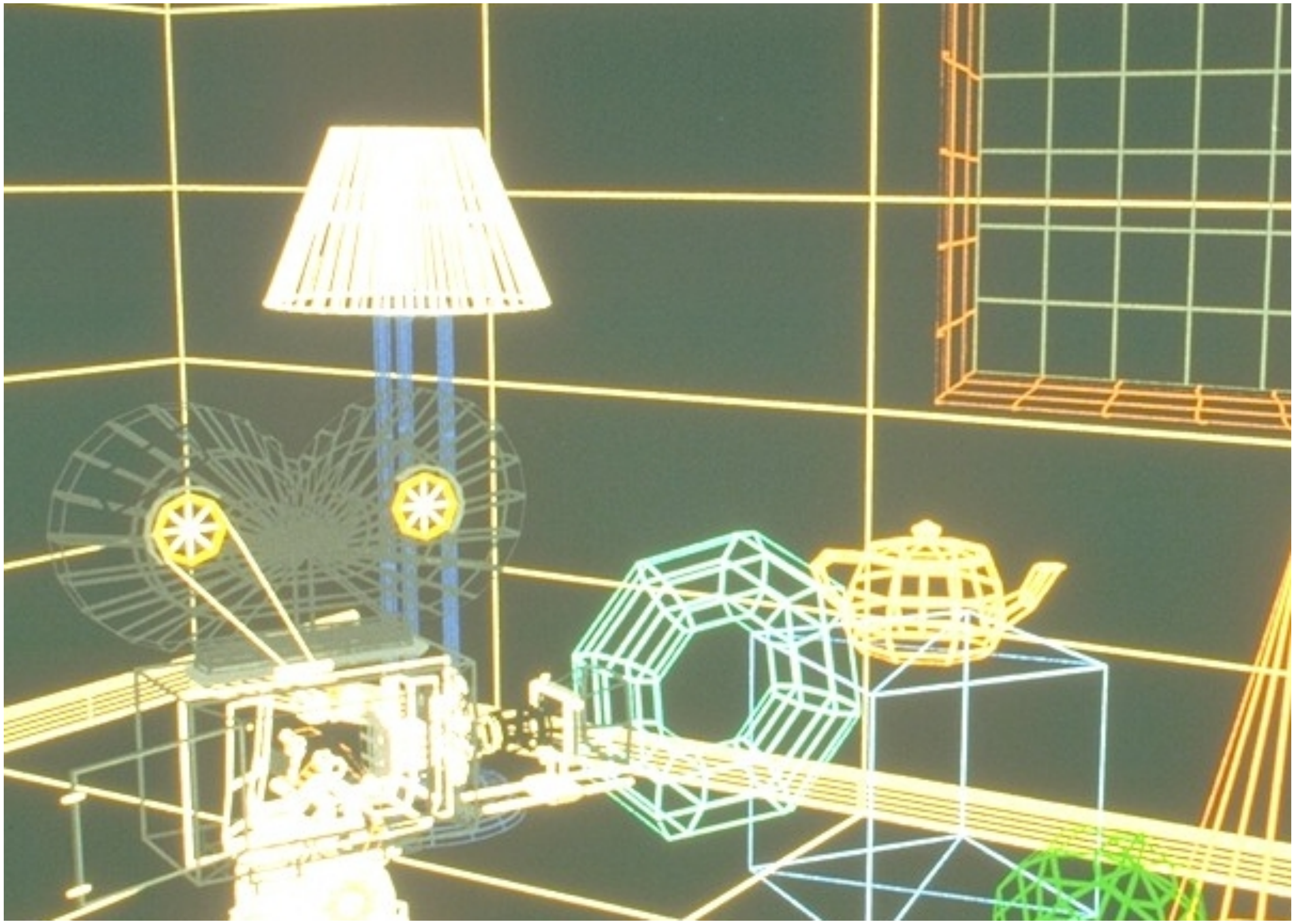
- Correct normals are essential for correct lighting
- Associate a normal with each vertex

```
normalsArray.push( u, v, n );
pointsArray.push( x, y, z );
...
```

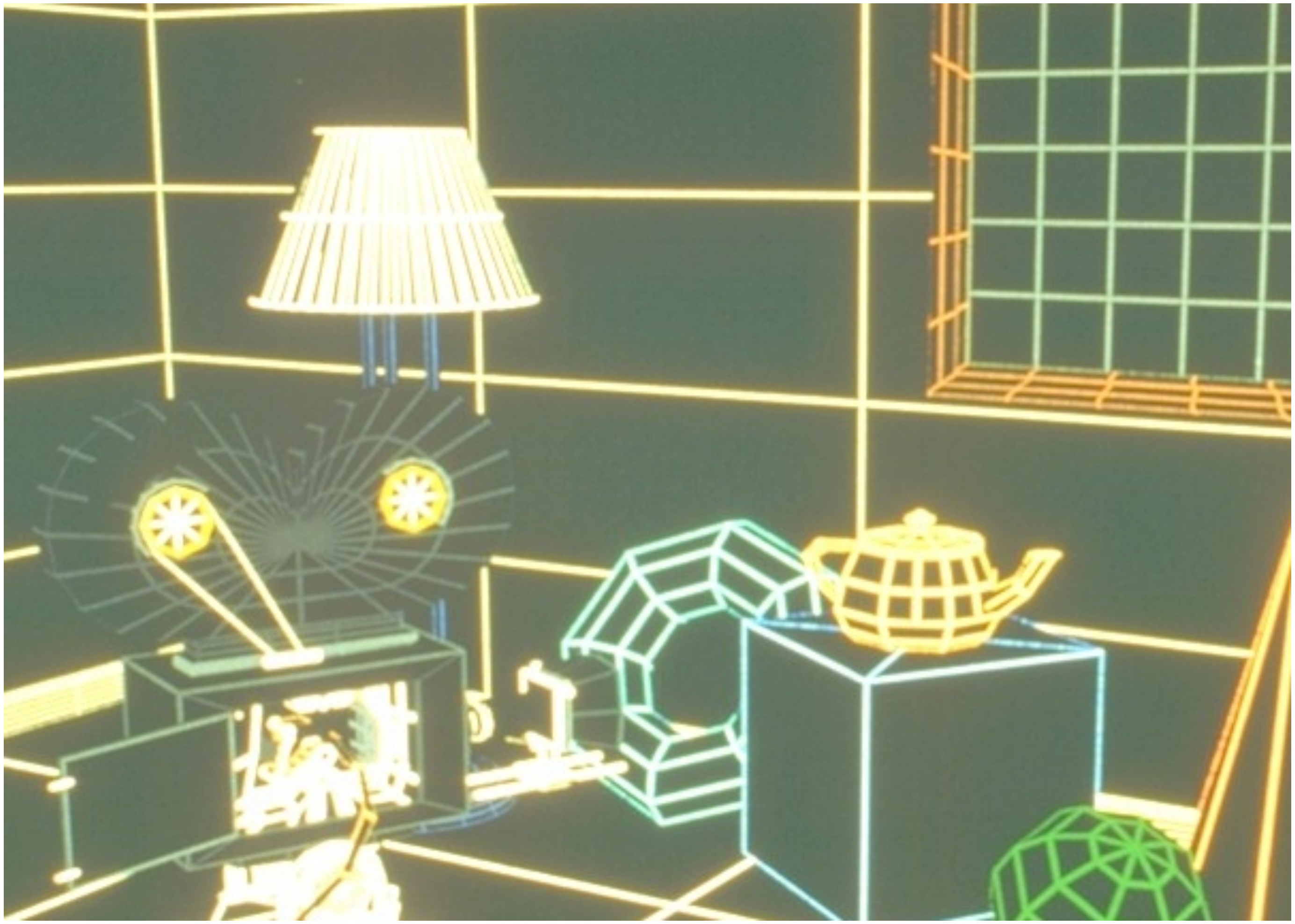


- All normals must be specified i
  - More on why in the next slide set .

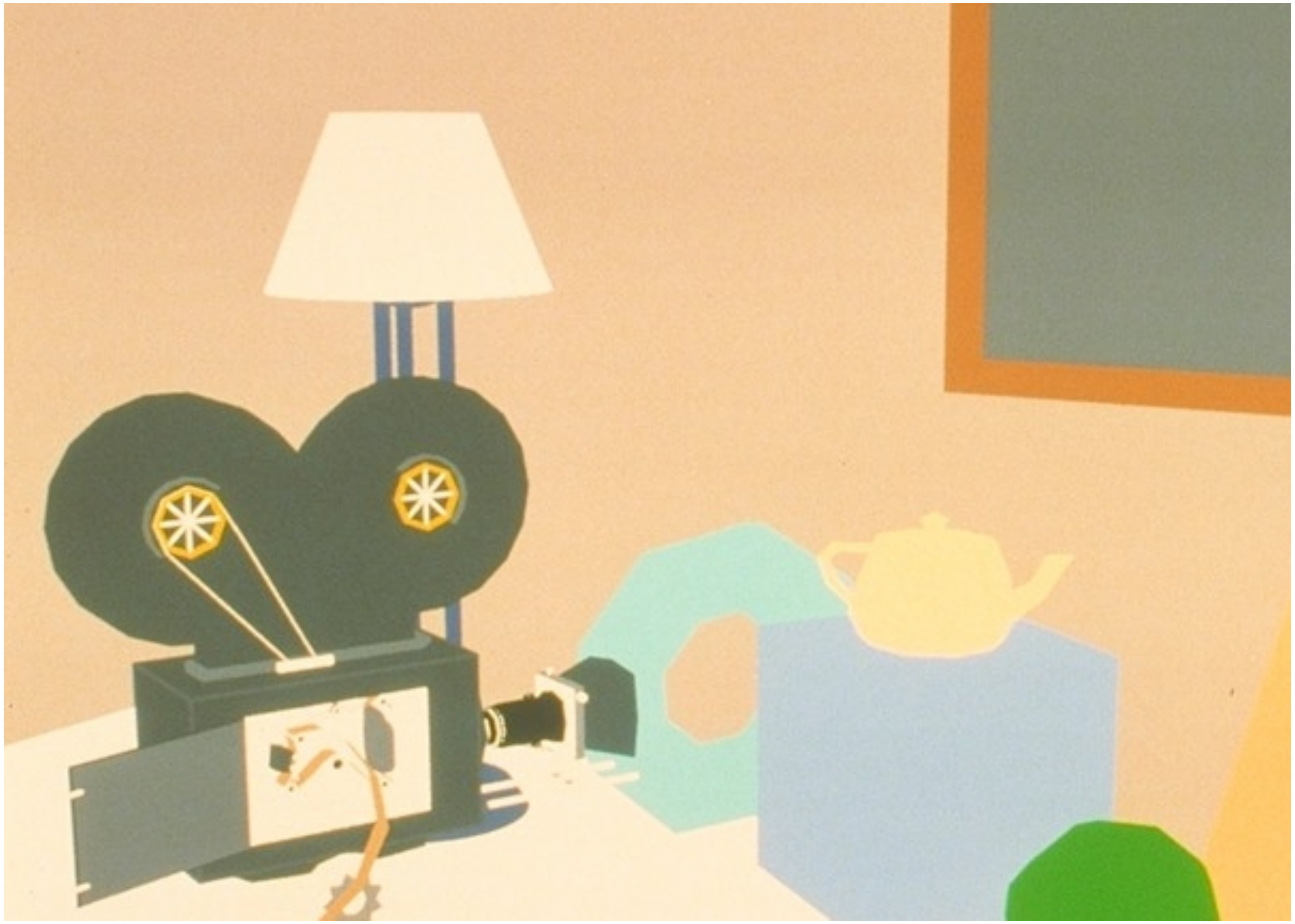
Colored Wireframe



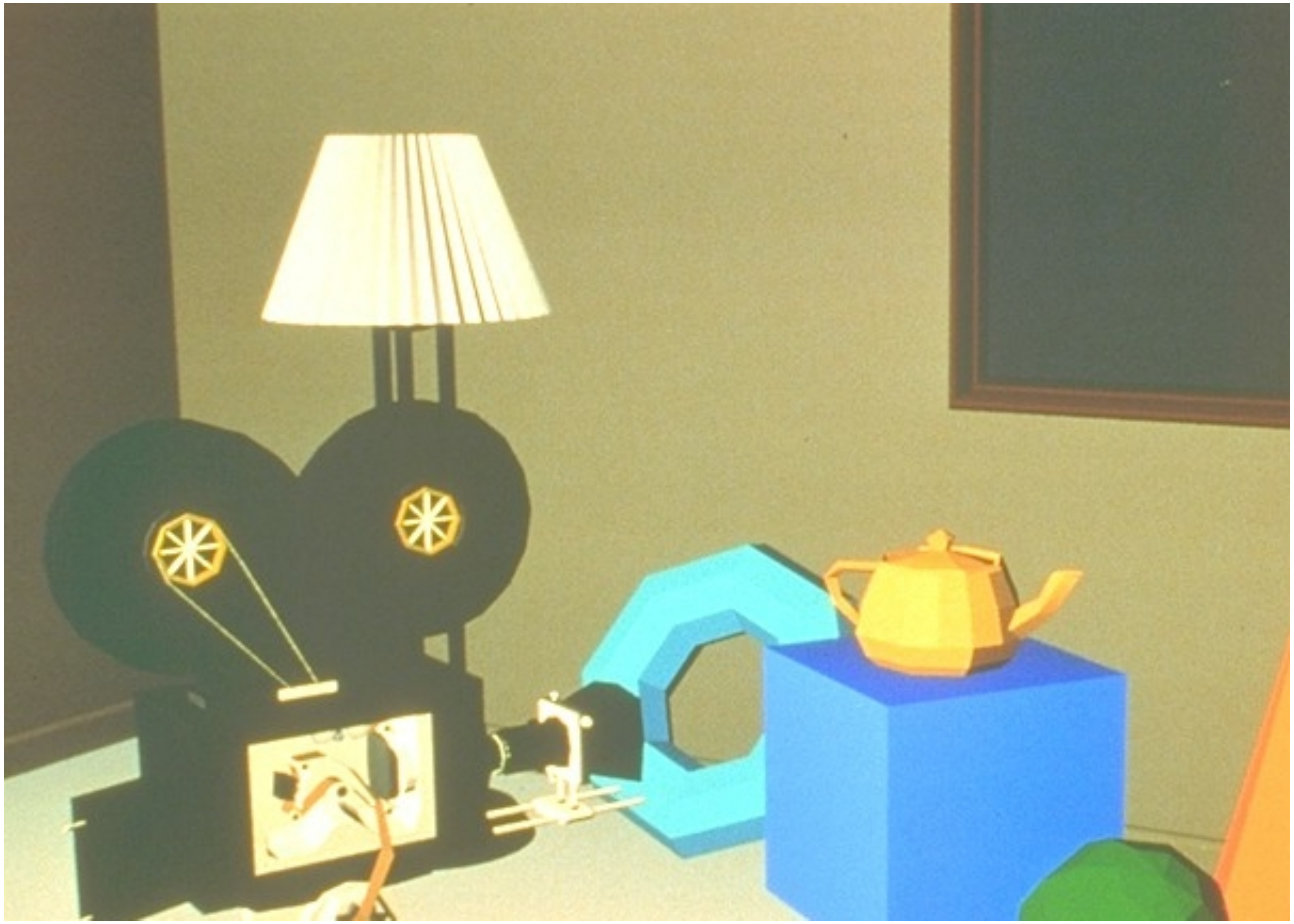
Colored Hidden-Line Removal



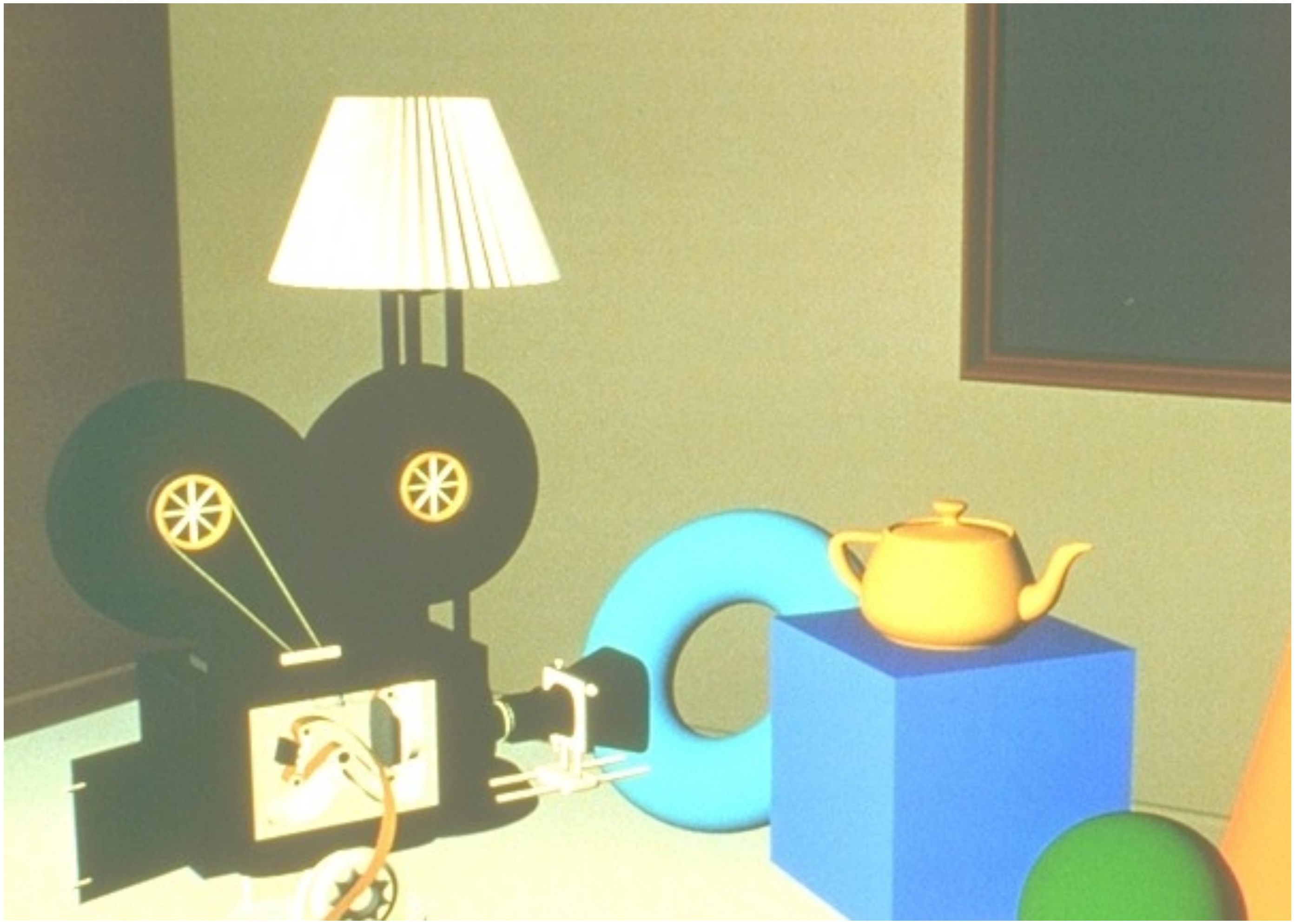
Ambient Term Only



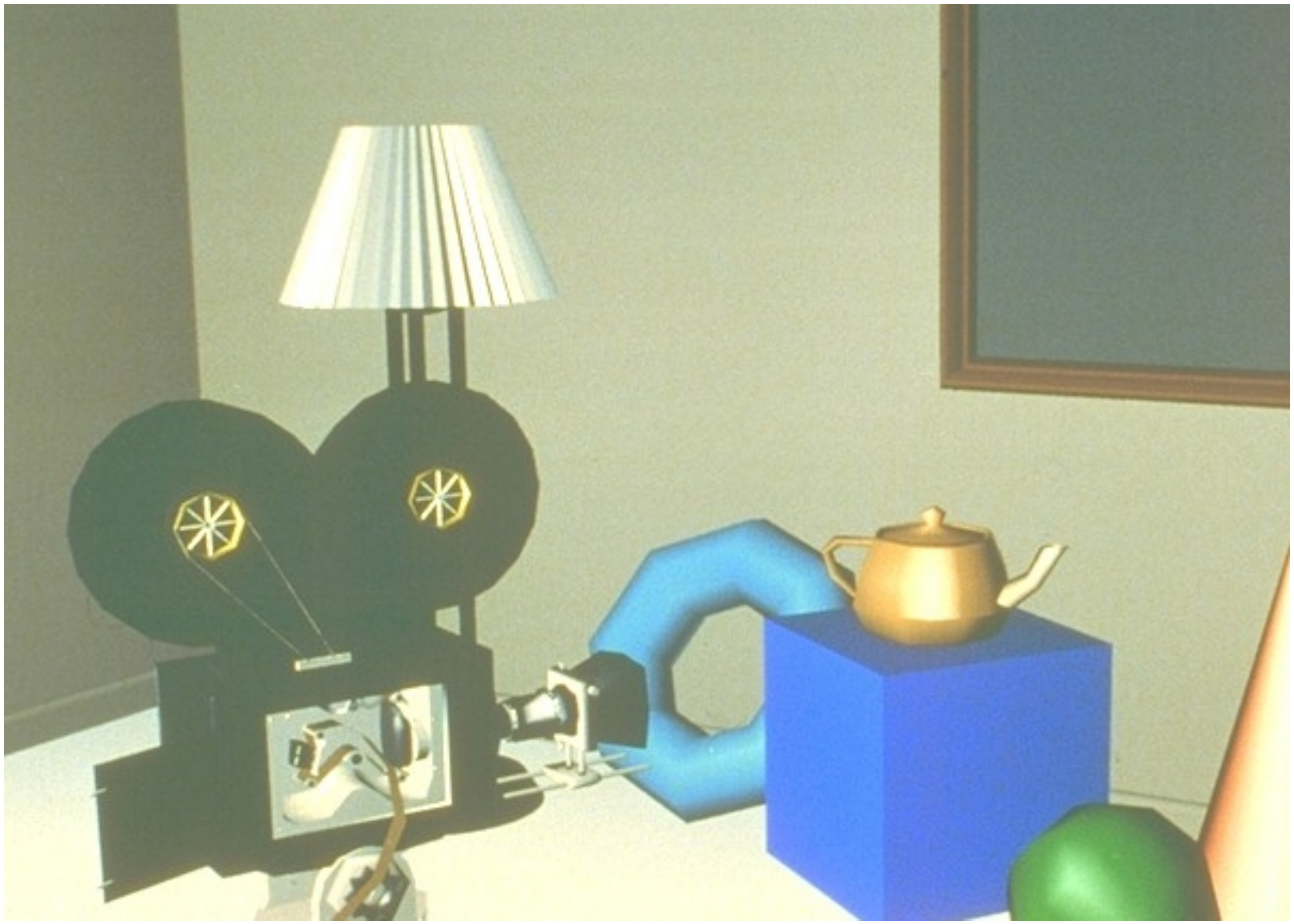
Flat Shading



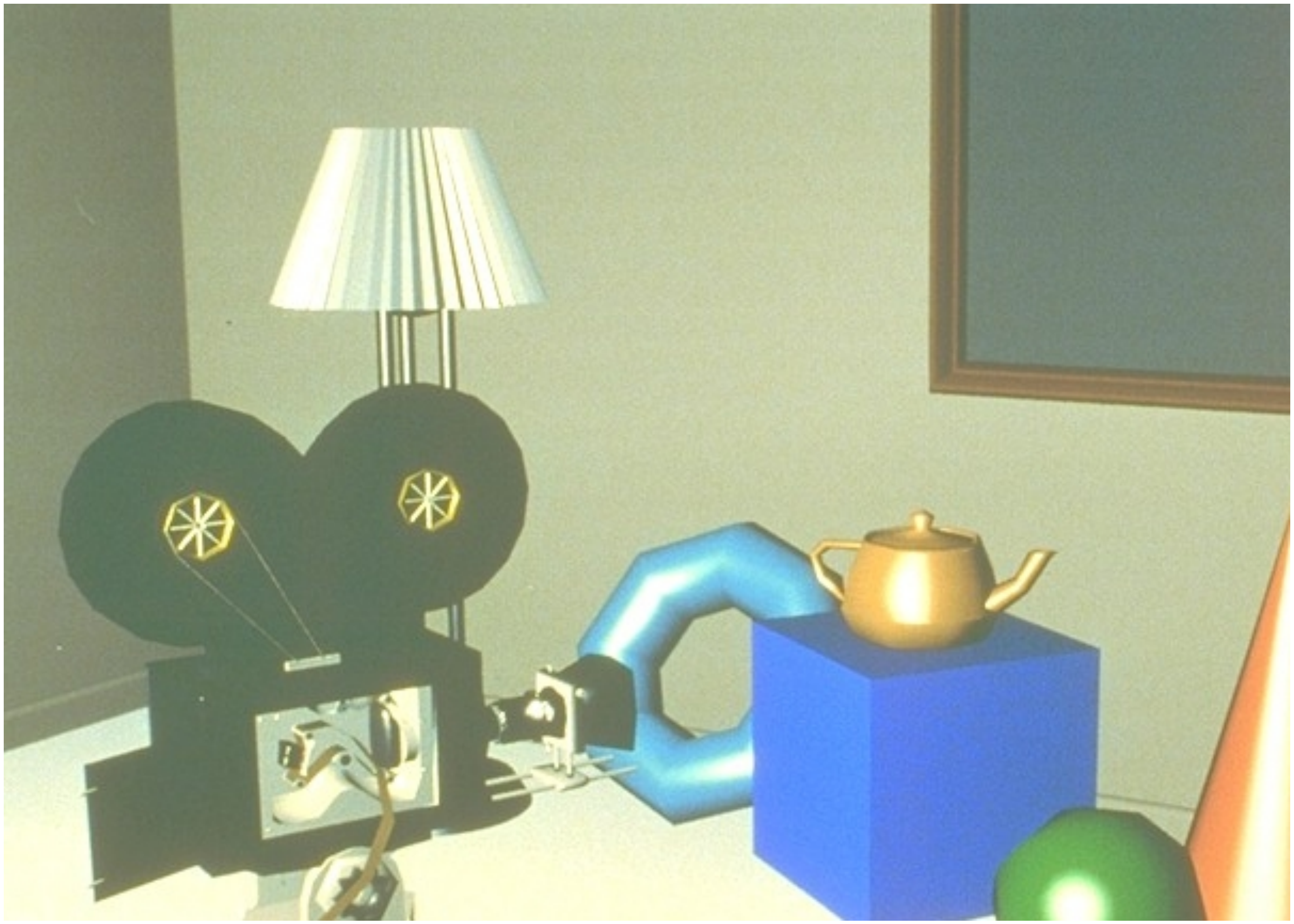
Diffuse Shading + Interp. Normals



Gouraud Shading



Ambient + Diffuse + Specular



Ambient + Diffuse + Specular + Interpolated Normals

Radiosity

Texture Mapping

Texture Mapping + Ray Tracing