Fundamentals of
Computer Graphics and Image Processing

Shading and Lighting (04)

doc. RNDr. Martin Madaras, PhD.
martin.madaras@fmph.uniba.sk

Overview

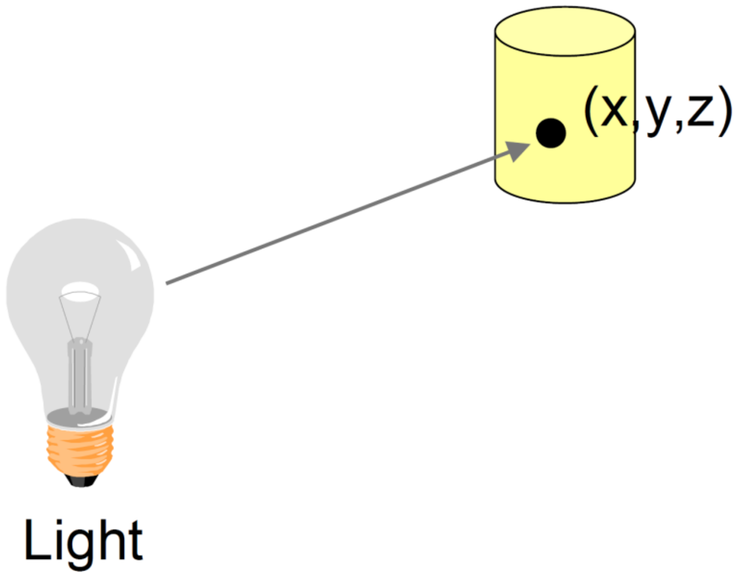
- Direct Illumination
  - **Emission at light sources**
  - Scattering at surfaces
  - Gouraud shading
- Global Illumination
  - Shadows
  - Refractions
  - Inter-object reflections

How the lectures should look like #1

- Ask questions, please!!!
- Be communicative
- More active you are, the better for you!

Overview

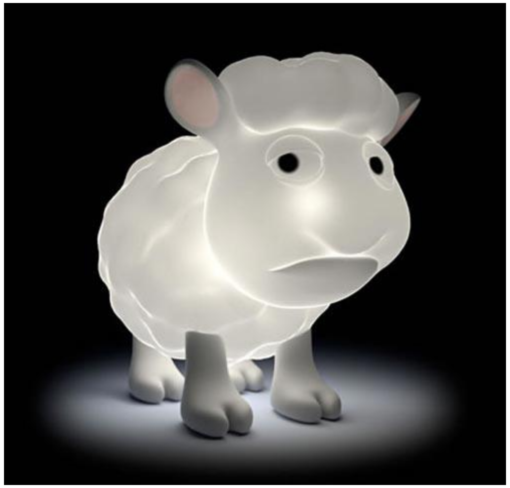
- $I_L(x, y, z, \theta, \varphi, \lambda)$
  - describes intensity of energy,
  - leaving a light source, ...
  - arriving at location (x,y,z), ...
  - from direction (θ,ϕ), ...
  - with wavelength λ



Light Source types

- Omnidirectional
- Spotlight
- Area
- Directional
- Object

- what are the differences?

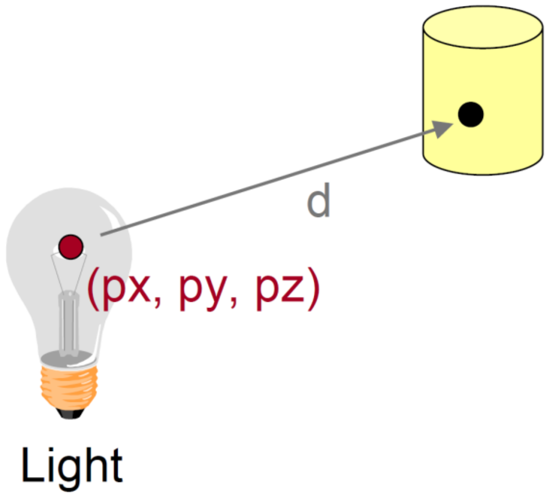


Light Source Models

- Simple mathematical models
- OpenGL support
  - Point light
  - Directional light
  - Spot light

Point Light Source

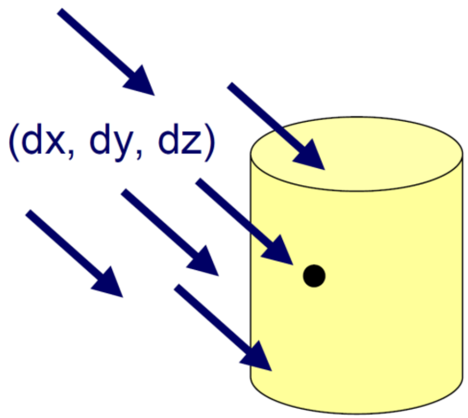
- Models omni-directional source
  - intensity (I_0),
  - position (x, y, z),
  - factors (k_c, k_l, k_q) for attenuation with distance (d)



$$I_L = \frac{I_0}{k_c + k_l d + k_q d^2}$$

Directional Light Source

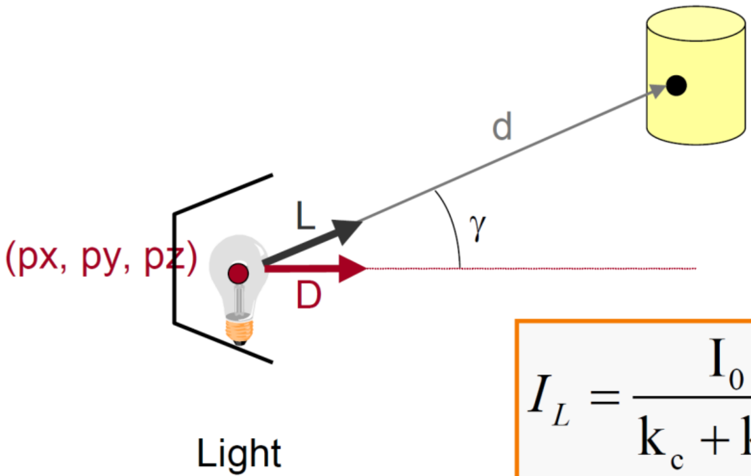
- Models point light source at infinity
  - intensity (I_0),
  - direction (dx, dy, dz),
  - no attenuation with distance



$$\boxed{I_L = I_0}$$

Spot Light Source

- Models point light source with direction
  - intensity (I_0),
  - position (x, y, z),
  - direction (d),
  - attenuation $\left(k_c, k_l, k_q\right)$.



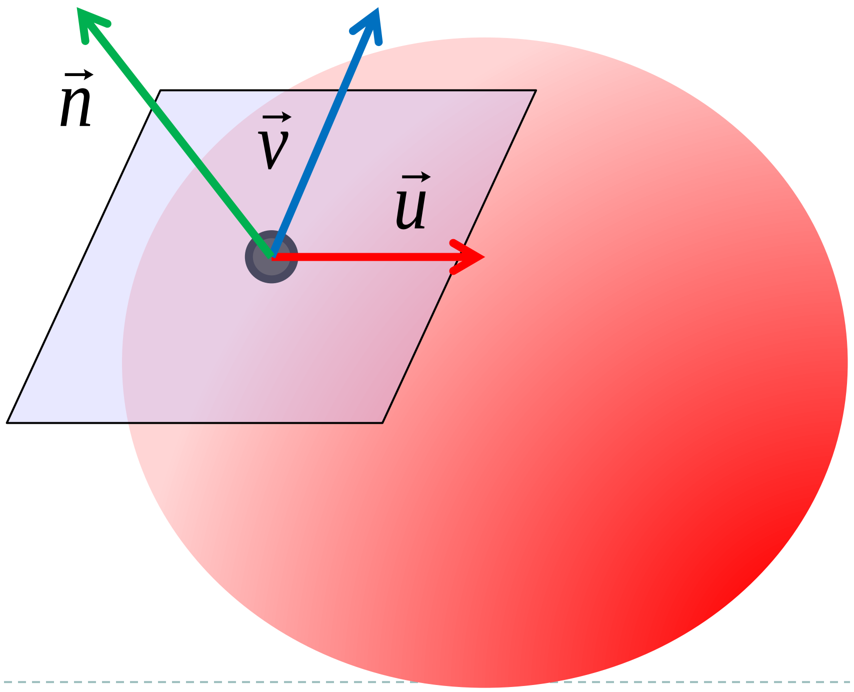
$$I_L = \frac{\mathrm{I}_0 (D \bullet L)}{\mathrm{k_c} + \mathrm{k_l} d + \mathrm{k_q} d^2}$$

Overview

- Direct Illumination
  - Emission at light sources
  - **Scattering at surfaces**
  - Gouraud shading
- Global Illumination
  - Shadows
  - Refractions
  - Inter-object reflections

Surface normal vector

- Perpendicular to the surface at the point
- Computation:
  - Usually from tangent vectors
  - Vector cross product $\vec{n} = \vec{u} \times \vec{v}$
  - Depends on the object representation



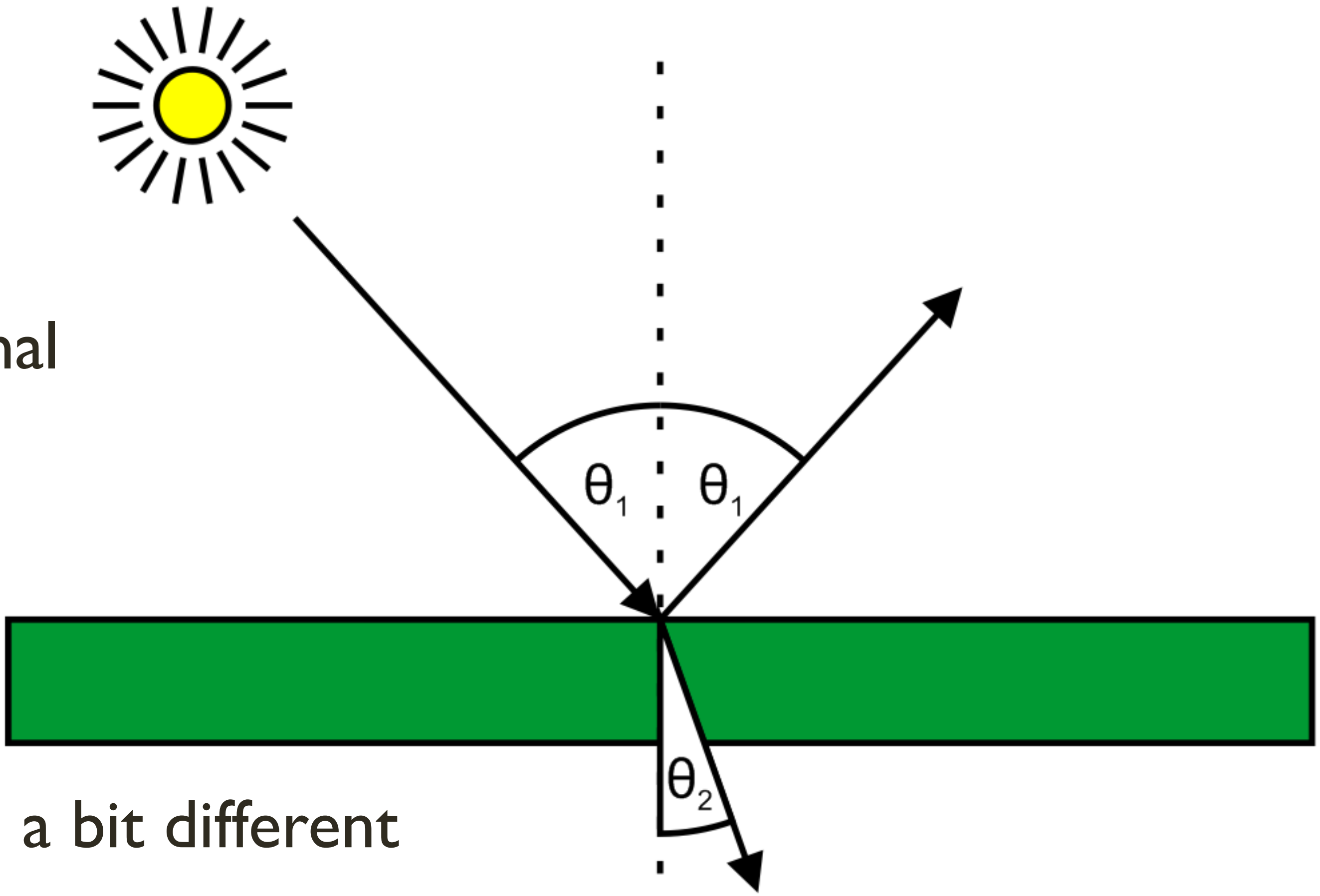
- Vector normalization $\hat{n} = \dfrac{\vec{n}}{|\vec{n}|}$

Tangent vectors

- Parametric representation
  - $X = x(u,v)$
  - $Y = y(u,v)$
  - $Z = z(u,v)$
  - Partial derivation by $u,v \rightarrow$ vectors t_u, t_v
- Polygonal representation
  - Tangent vectors are edge vectors
  - Mind the orientation!

Elementary theory

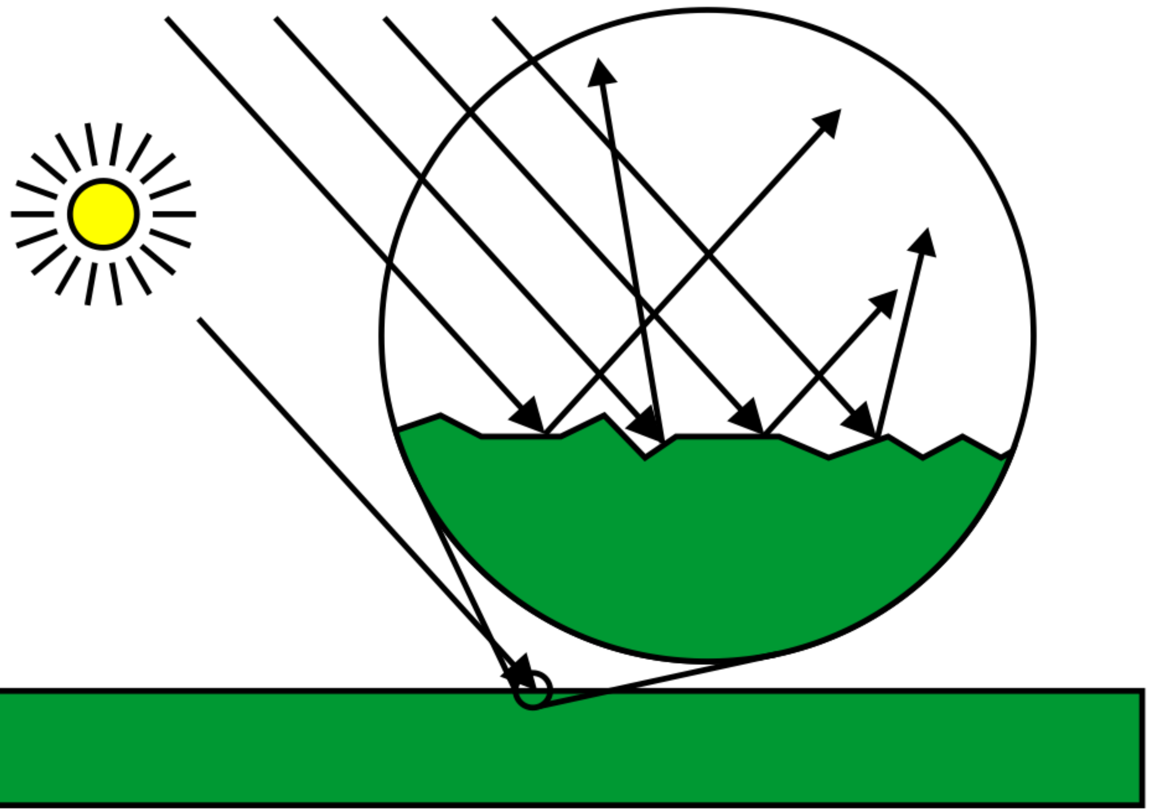
- Light-surface interaction
- Reflection
- Refraction
  - Snell's law
- Surface normal vector



- Real world is a bit different

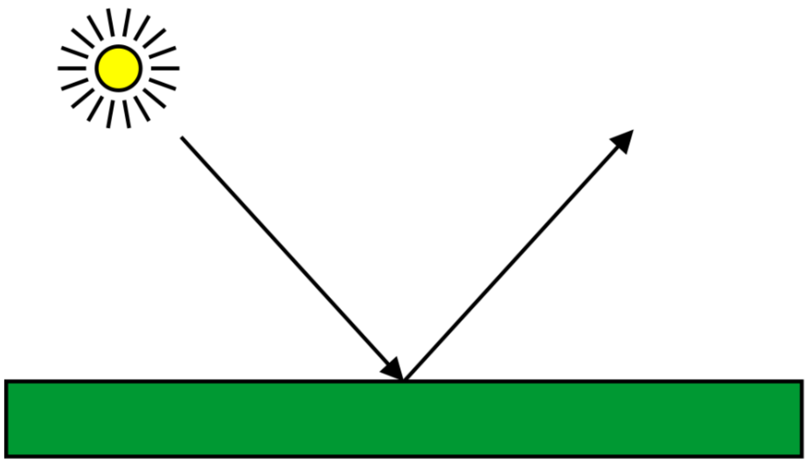
Surface types

- Reflective
- Diffuse – Lambertian
- Both

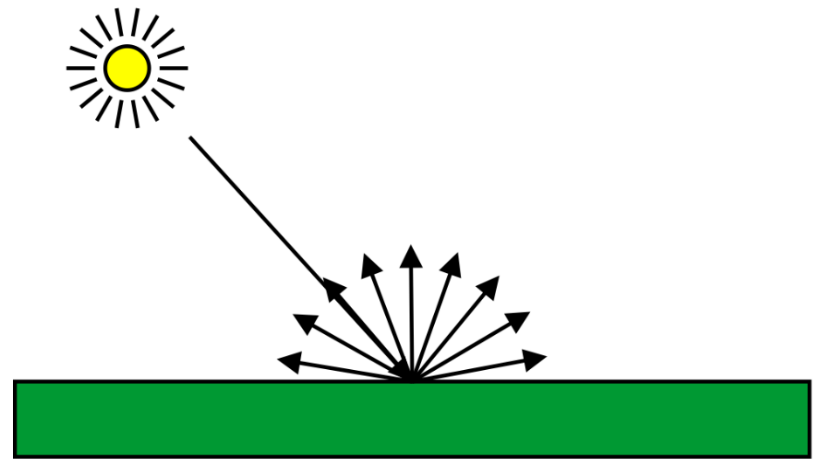


Surface types

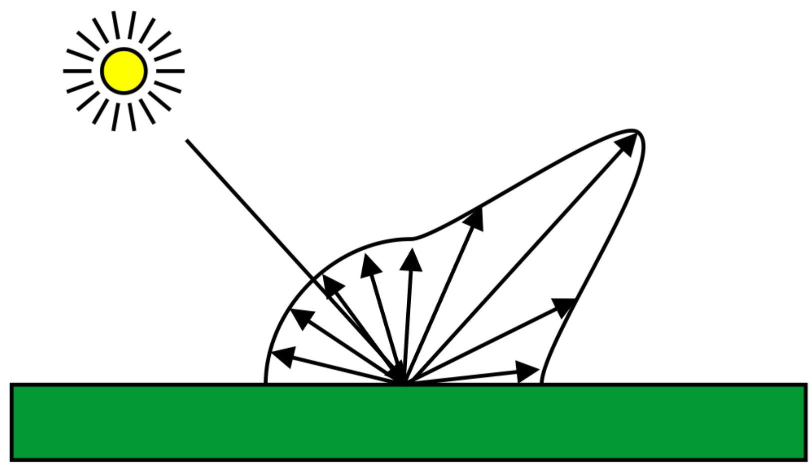
Mirror



Matte



directional component



indirectional component

Light models

- Empiric – e.g. Phong lighting model
  - cheap computation
  - physically incorrect
  - visually plausible

- Physically-based
  - energy transfer, light propagation
  - closer to real-world physics
  - expensive

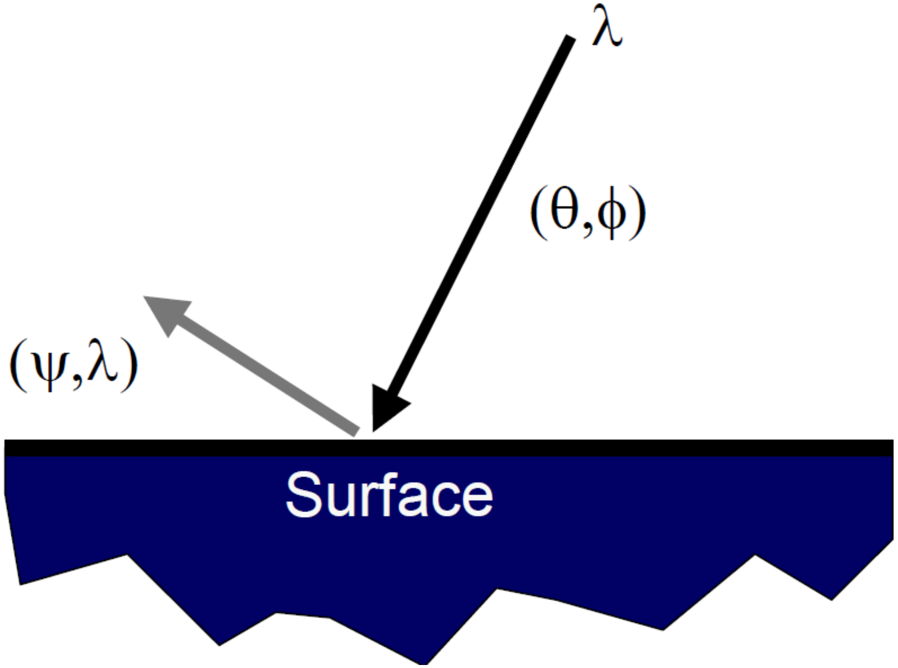
Local illumination models

- Fast but inaccurate
- Empirical (no physical background)
- Many physical effects are impossible to achieve

- Computer games, real-time rendering

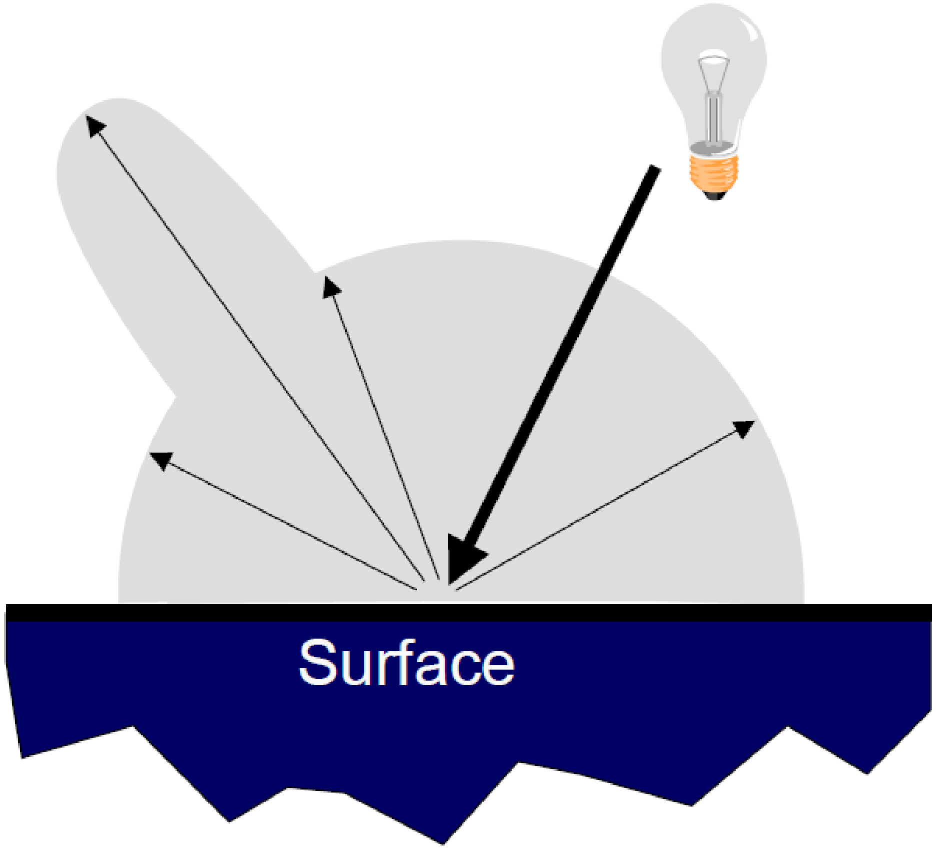
Modeling Surface Reflectance

- Rs(θ,ϕ,γ,ψ,λ) …
  - describes the amount of incident energy, …
  - arriving from direction (θ,ϕ), …
  - leaving in direction (γ,ψ), …
  - with wavelength λ



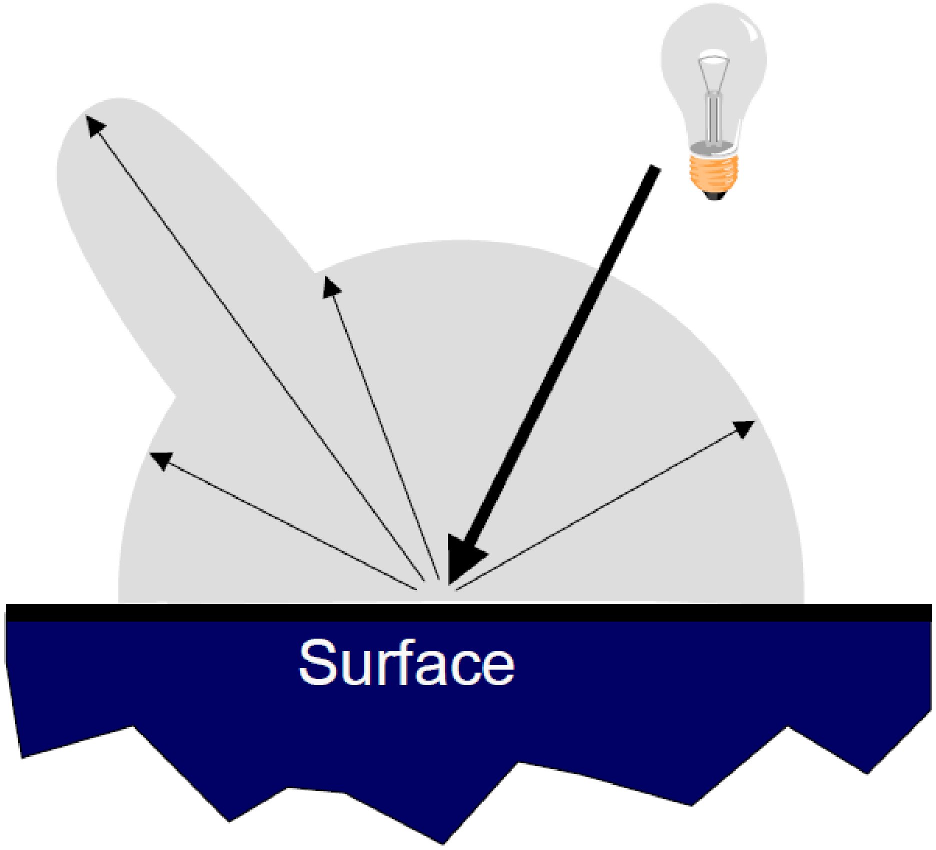
OpenGL Reflectance Model

- Simple empirical model:
  - diffuse reflection +
  - specular reflection +
  - emission +
  - “ambient”
  - Bui Tuong Phong, 1973



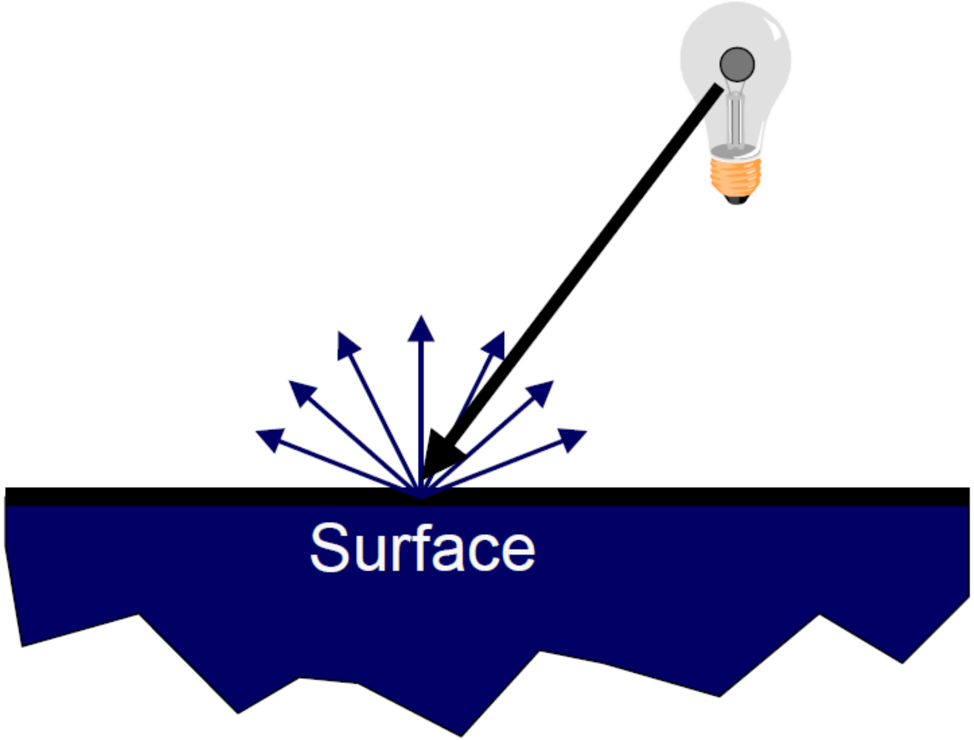
OpenGL Reflectance Model

- Simple empirical model:
  - diffuse reflection +
  - specular reflection +
  - emission +
  - “ambient”



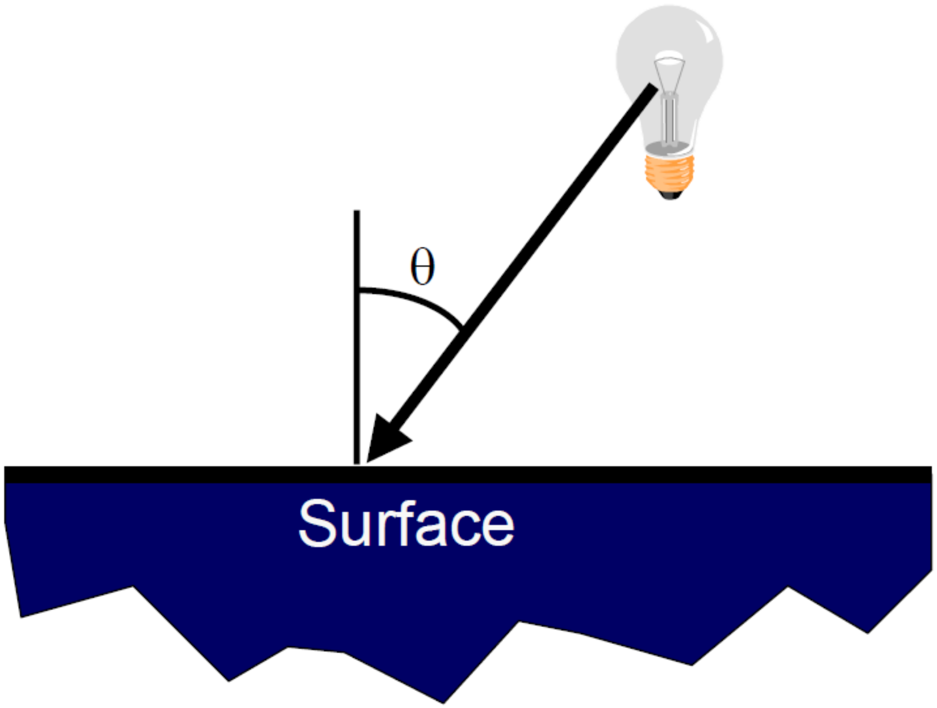
Diffuse Reflection

- Assume surface reflects equally in all directions
  - Examples: chalk, clay



Diffuse Reflection

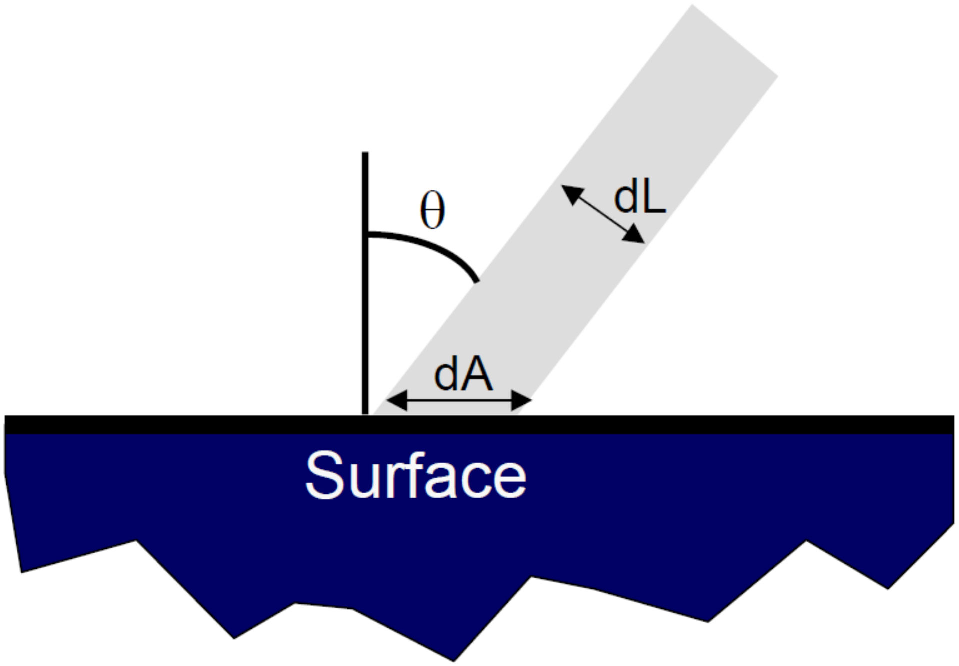
- How is light reflected ?
  - Depends on an angle of incident light



Diffuse Reflection

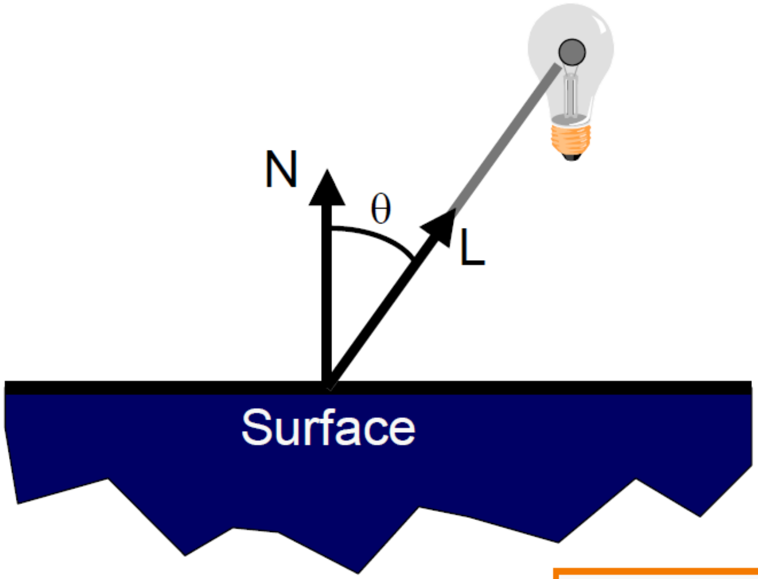
- How is light reflected ?
  - Depends on an angle of incident light

$$dL = dA \cos \Theta$$



Diffuse Reflection

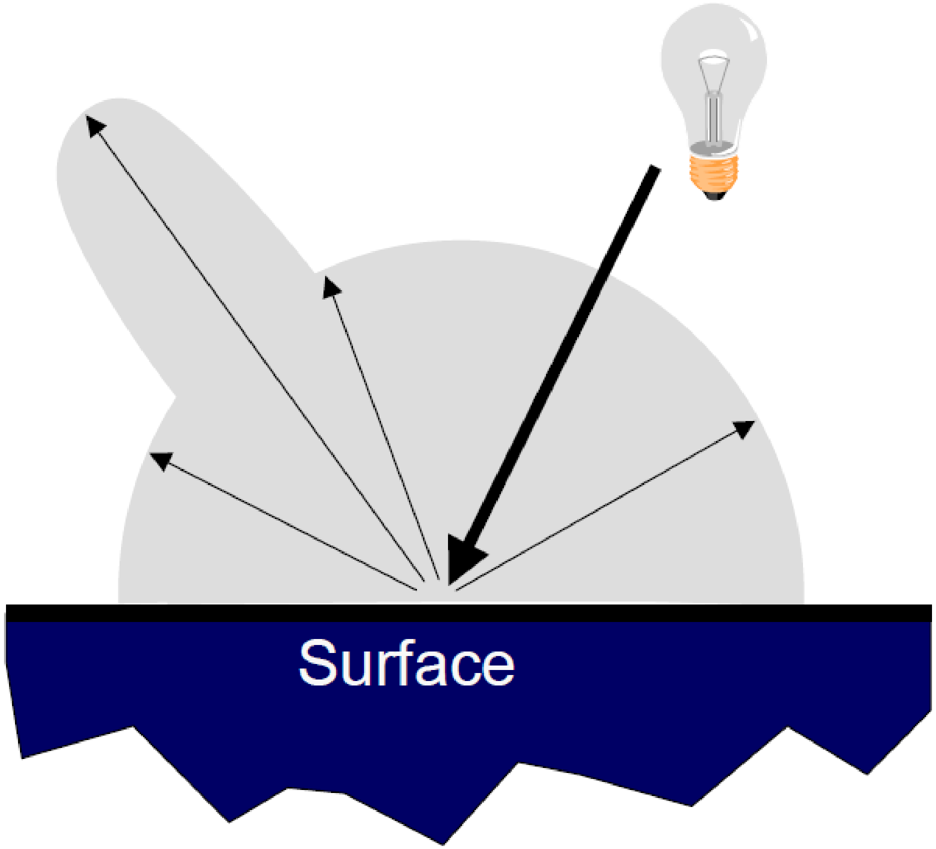
- Lambertian model
  - cosine law (dot product)



$$\boxed{I_D = K_D (N \bullet L) I_L}$$

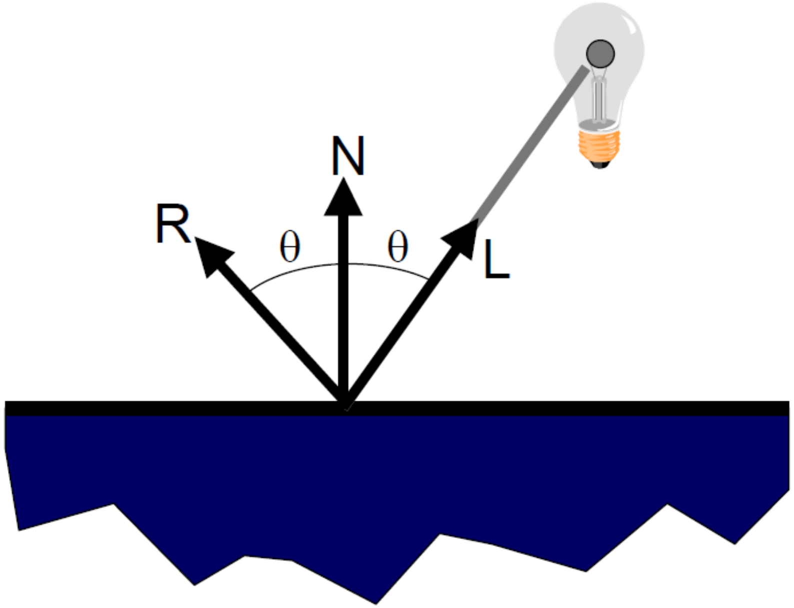
OpenGL Reflectance Model

- Simple empirical model:
  - diffuse reflection +
  - specular reflection +
  - emission +
  - "ambient"



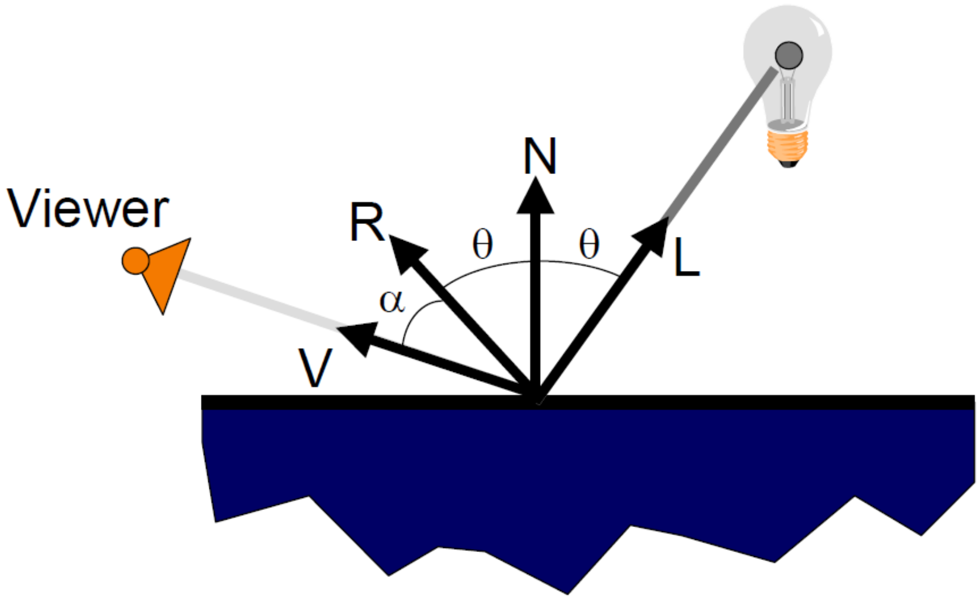
Specular Reflection

- Reflection is strongest near mirror angle
  - Examples: Mirrors, Metals



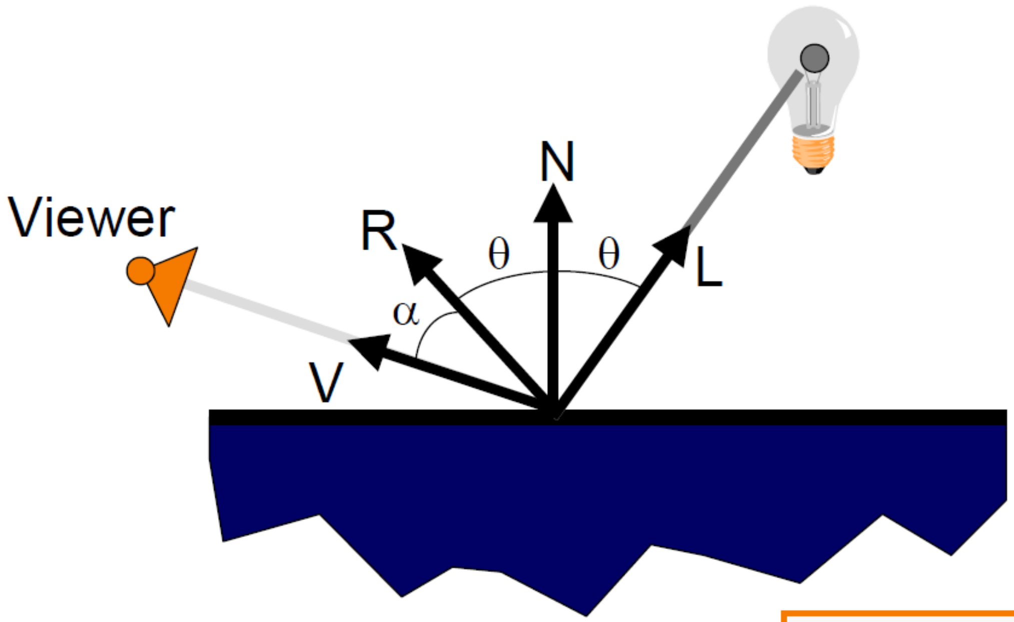
Specular Reflection

- How much light is seen ?
- Depends on:
  - angle of incident light
  - angle of viewer



Specular Reflection

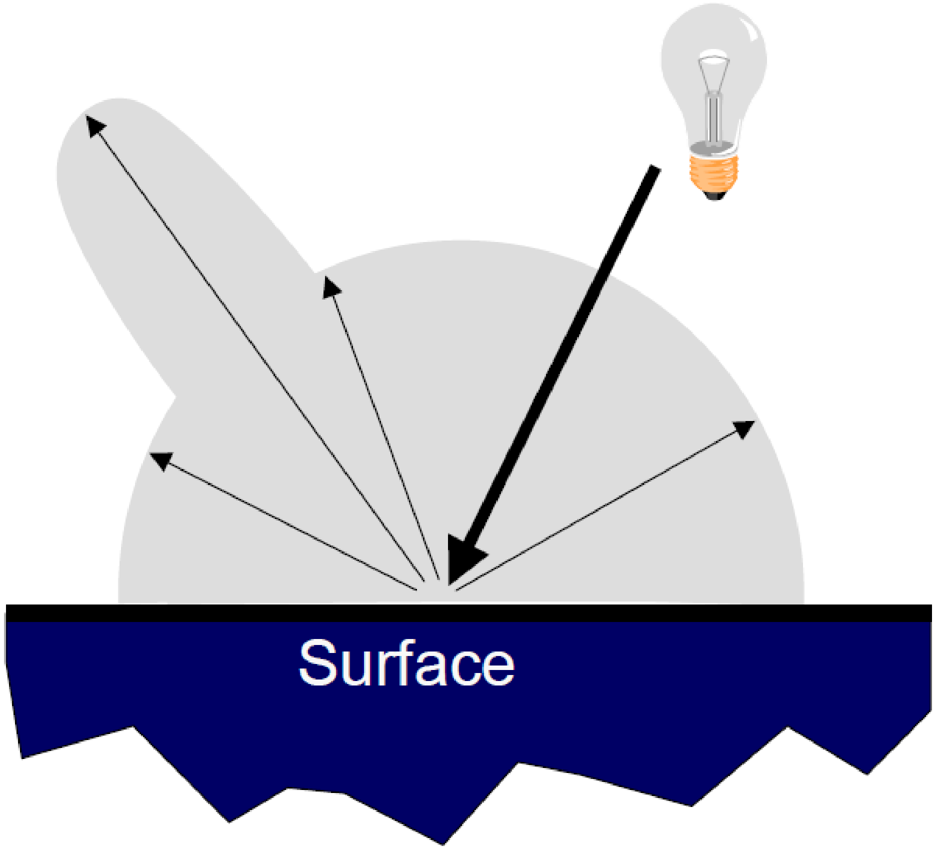
- Phong Model
  - cos(α)n *This is a physically motivated hack!*



$$\boxed{I_S = K_S (V \bullet R)^n I_L}$$

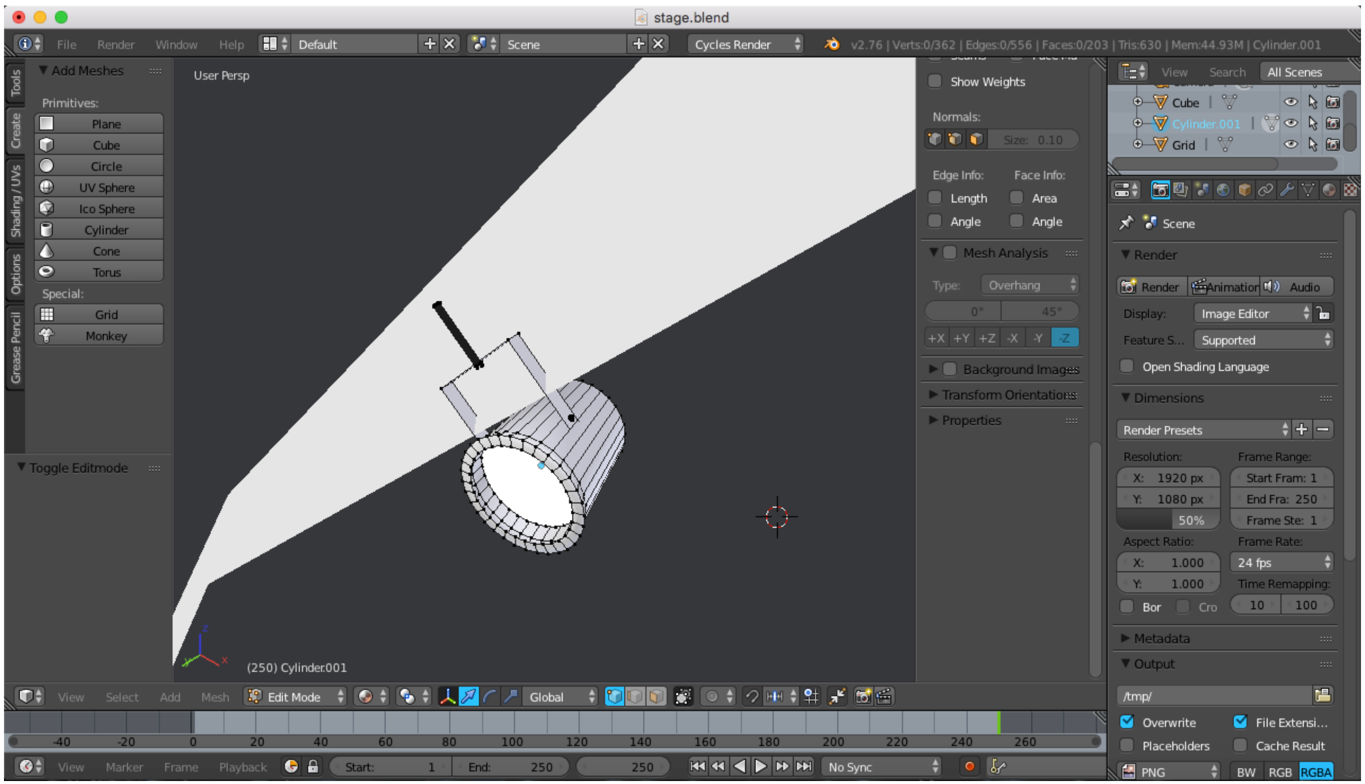
OpenGL Reflectance Model

- Simple empirical model:
  - diffuse reflection +
  - specular reflection +
  - emission +
  - “ambient”



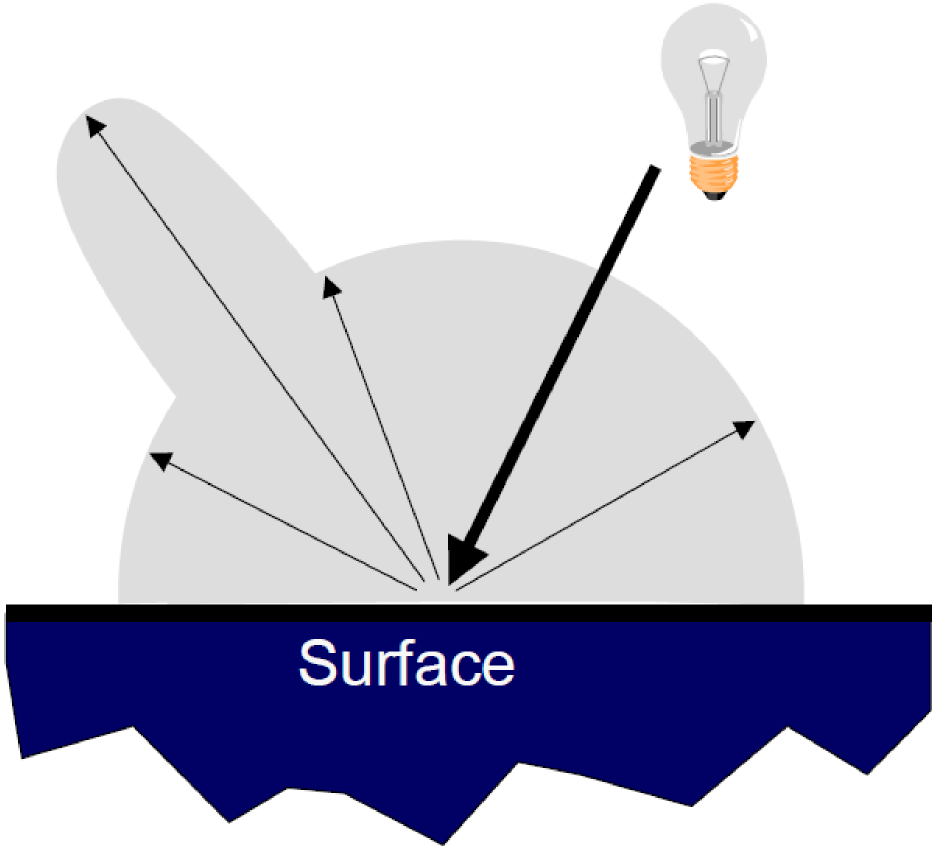
Emission

- Represents light emitted directly from surface



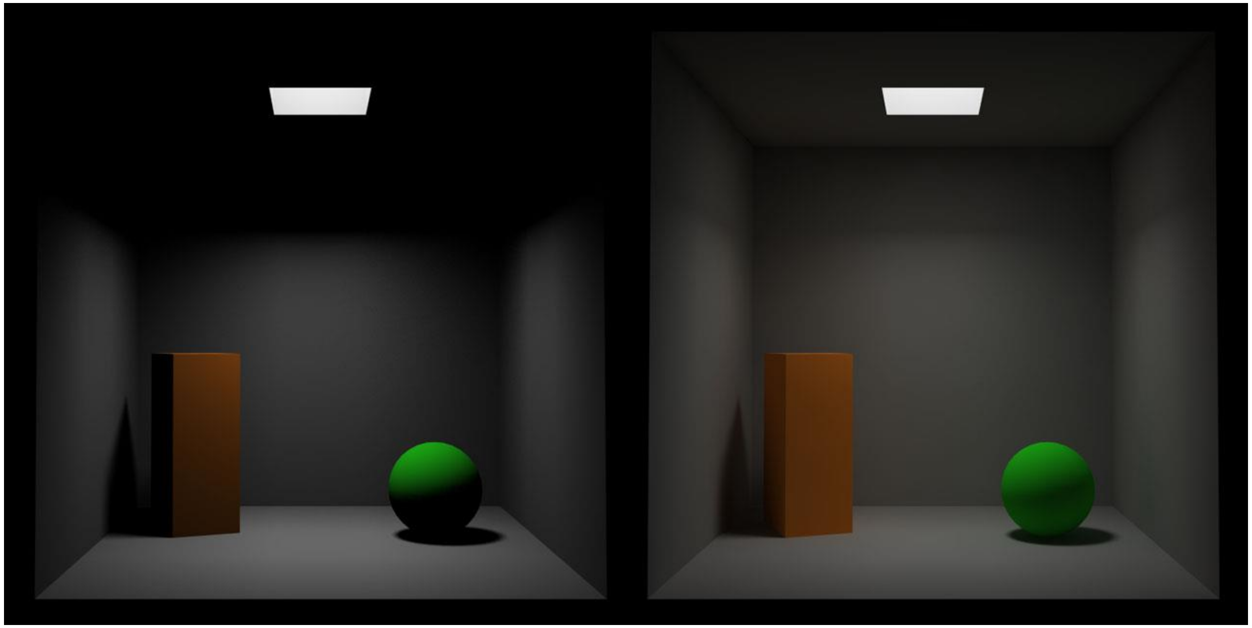
OpenGL Reflectance Model

- Simple empirical model:
  - diffuse reflection +
  - specular reflection +
  - emission +
  - "ambient"



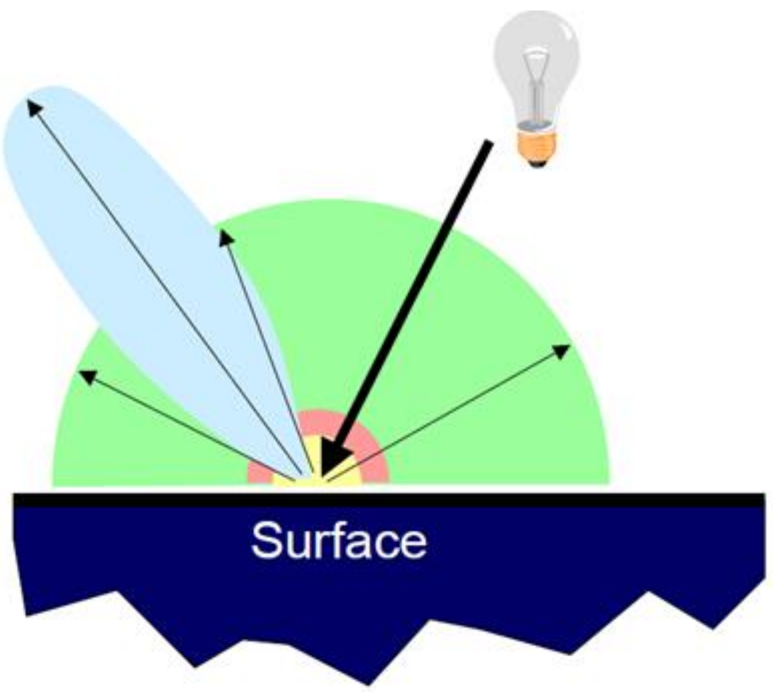
Ambient term

- Represents reflection of all indirect illumination
- This is a total hack
  - (avoids complexity of global illumination)!



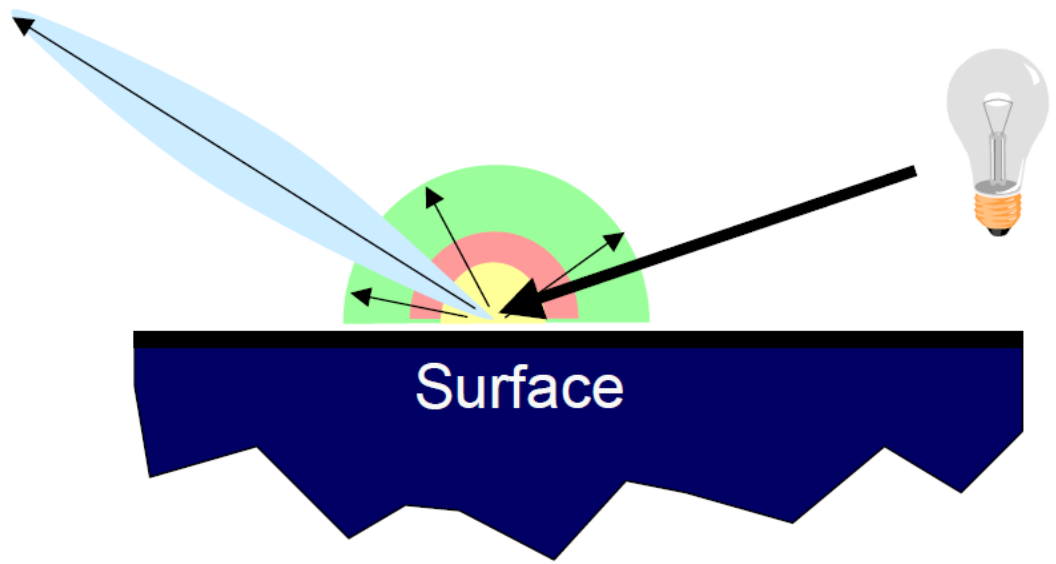
OpenGL Reflectance Model

- Simple empirical model:
  - diffuse reflection +
  - specular reflection +
  - emission +
  - "ambient"



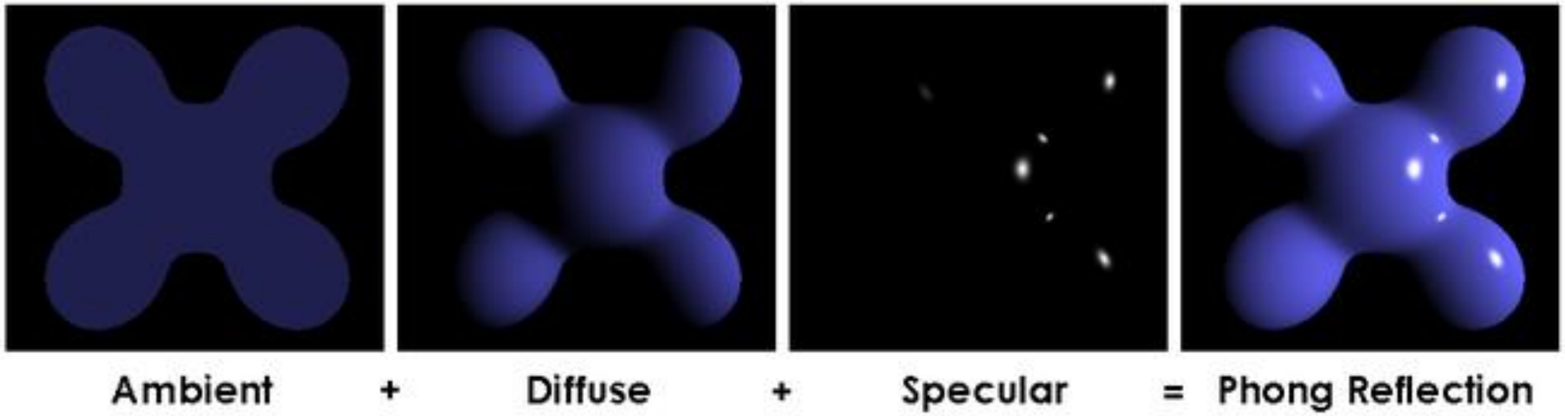
OpenGL Reflectance Model

- Simple empirical model:
  - diffuse reflection +
  - specular reflection +
  - emission +
  - “ambient”



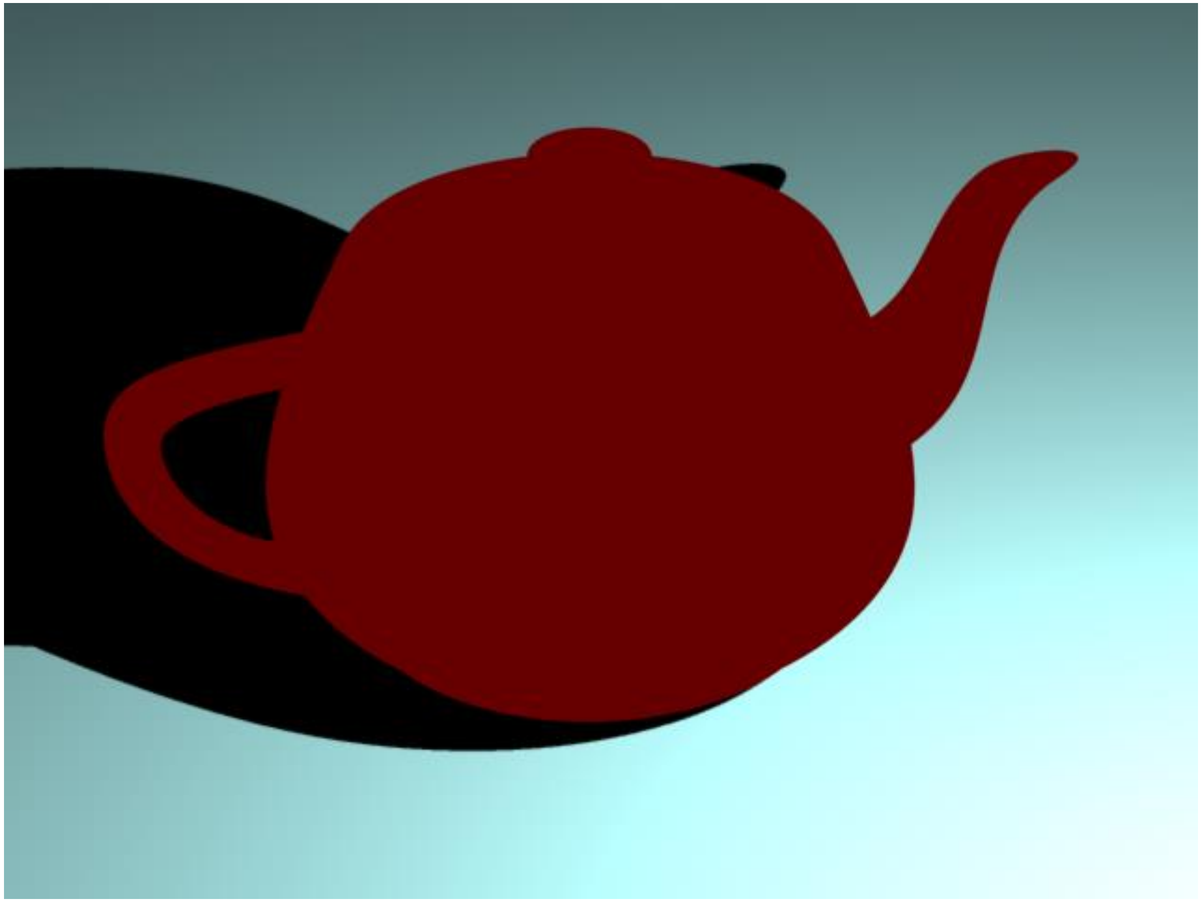
OpenGL Reflectance Model

- Simple empirical model:
  - diffuse reflection +
  - specular reflection +
  - emission +
  - “ambient”



Diffuse light

Ambient light

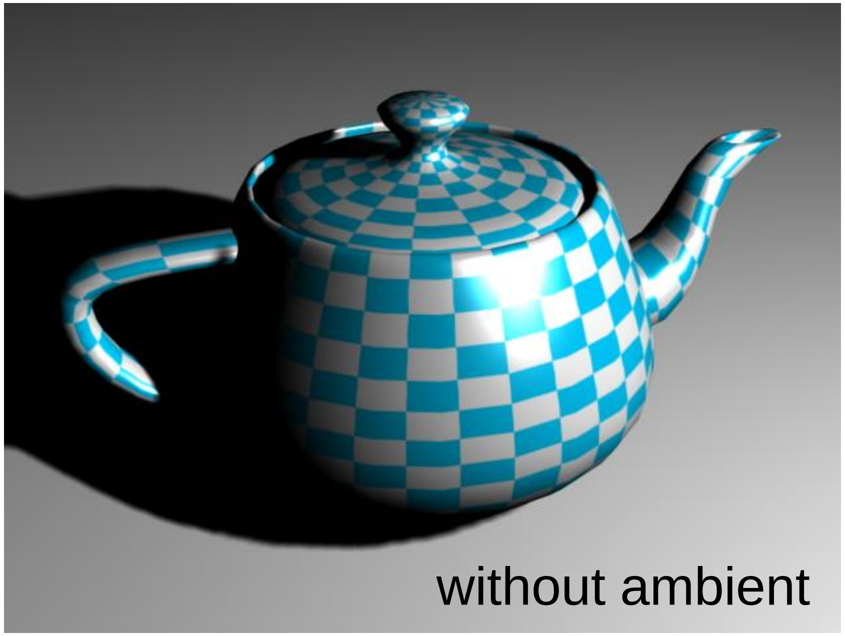


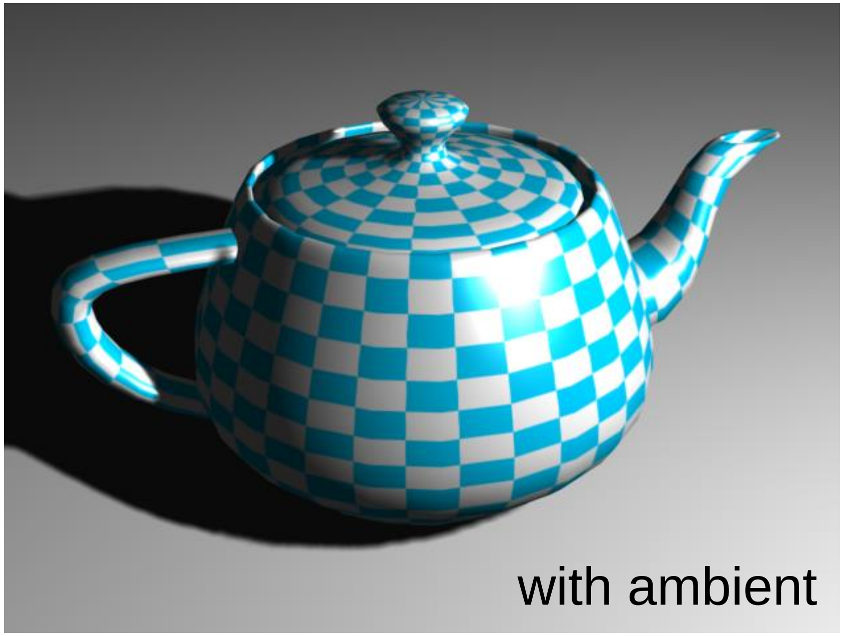
Diffuse + Ambient light

Specular + Diffuse + Ambient light

Phong lighting model

- **Ambient** + Diffuse + Specular components





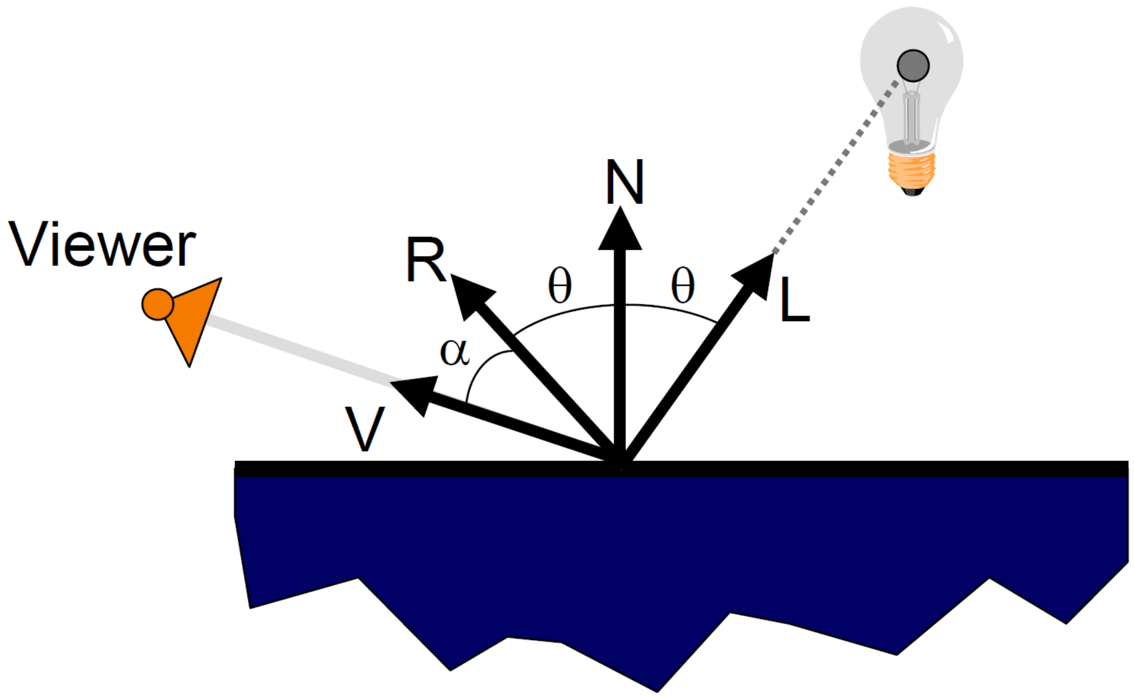
- Simulates global light scattered in the scene and reflected from other objects

Other lighting models

- Blinn-Phong
  - generalization of Phong's model
- Cook-Torrance
  - microfacets
- Oren-Nayar
  - rough surfaces
- Anisotropic microfacet distribution

Direct Illumination

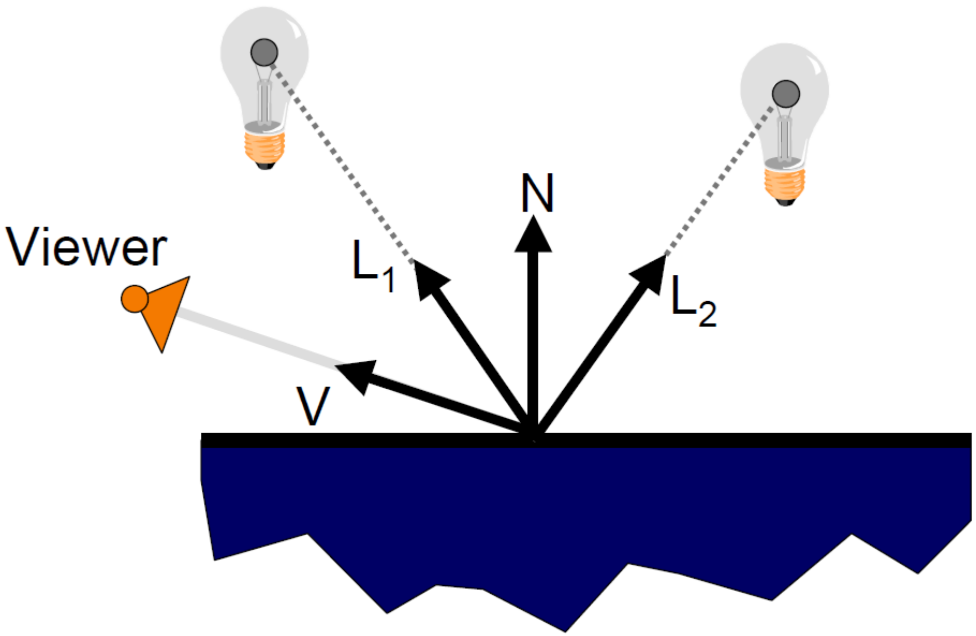
- Single light source example



$$I = I_E + K_A I_{AL} + K_D (N \bullet L) I_L + K_S (V \bullet R)^n I_L$$

Direct Illumination

- Multiple light source example



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

3D rendering pipeline

3D polygons

Modeling Transformation

⇩

Lighting

⇩

Viewing Transformation

⇩

Projection Transformation

⇩

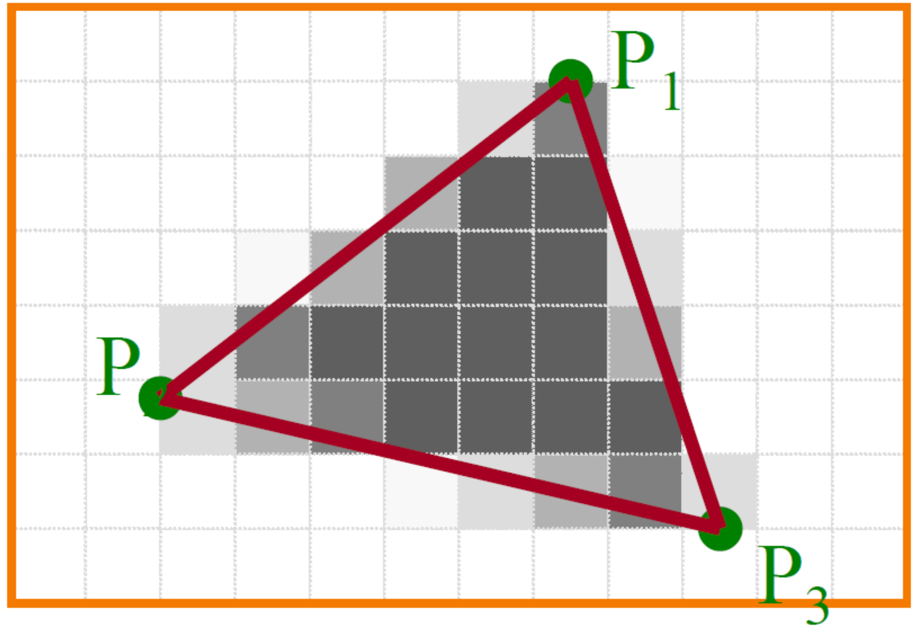
Clipping

⇩

Scan Conversion

2D Image

Draw pixels

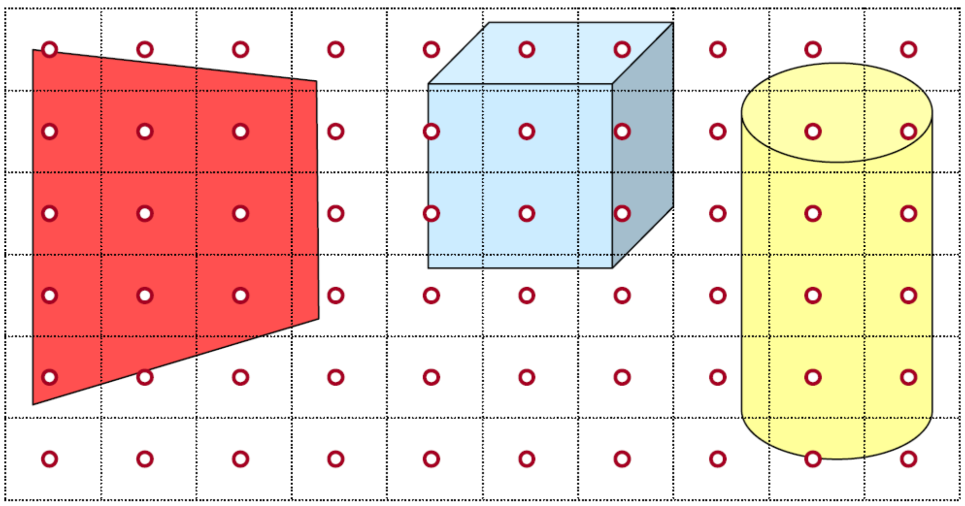


Pixel Shading

How to effectively get the value
of the illumination model
for each pixel ?

Ray Casting

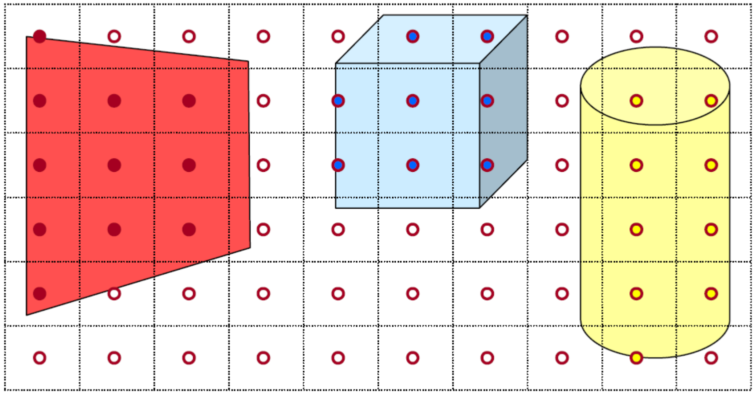
- Independent lighting calculation for each pixel
  - Computationally expensive

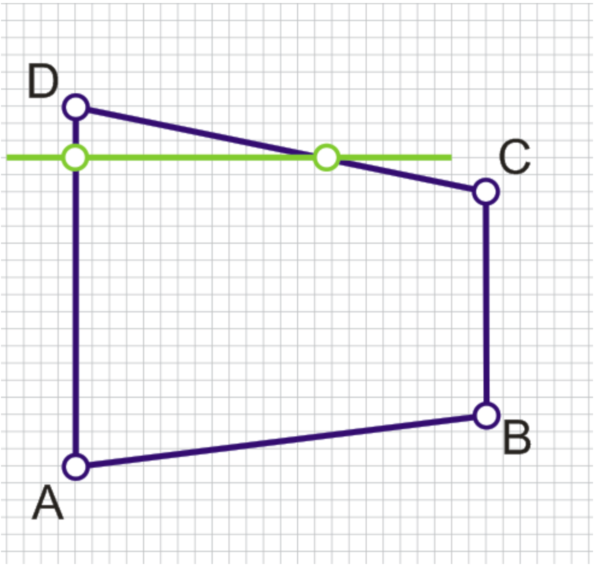


$$I = I_E + K_A I_{AL} + \sum_i (K_D (N \bullet L_i) I_i + K_S (V \bullet R_i)^n I_i)$$

Polygon Shading

- Can take advantage of spatial coherence
  - Illumination calculations of pixels of a triangle are related
  - Scanline rasterization





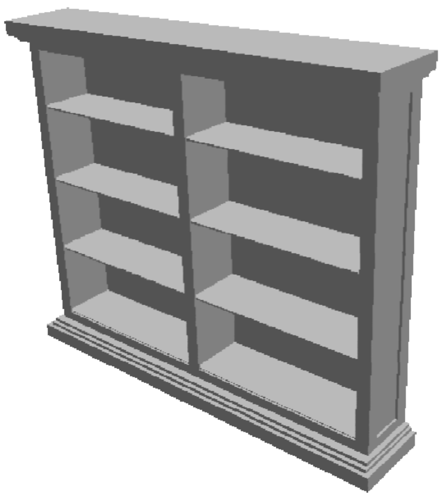
$$I = I_E + K_A I_{AL} + \sum_i (K_D (N \bullet L_i) I_i + K_S (V \bullet R_i)^n I_i)$$

Overview

- Global Illumination
  - Shadows
  - Refractions
  - Inter-object reflections
- **Shading**
  - Flat
  - Gouraud
  - Phong

Flat Shading

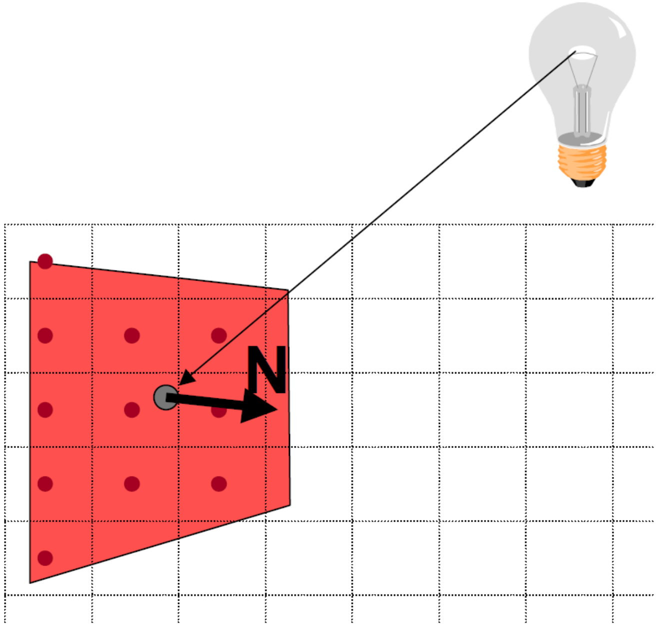
- Fill triangles using single calculated color



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

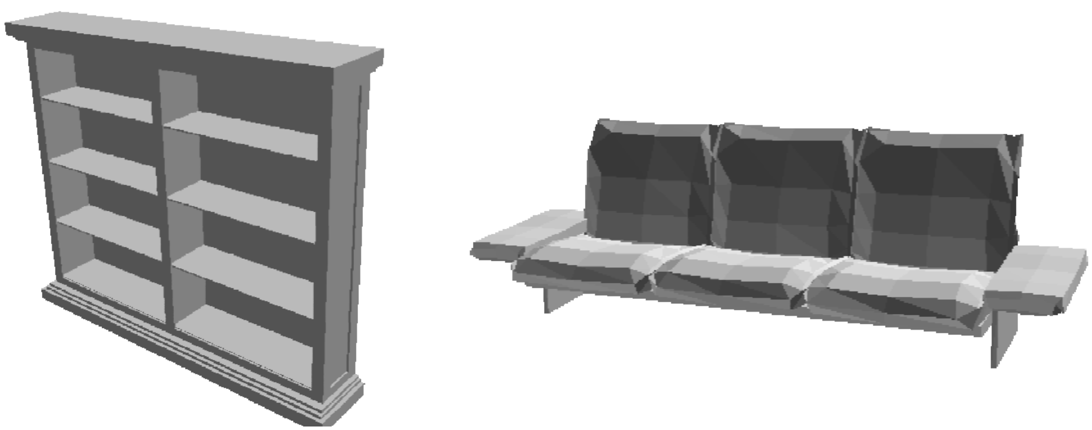
Flat Shading

- One Illumination calculation per polygon
- Assign all pixels of each polygon the same color



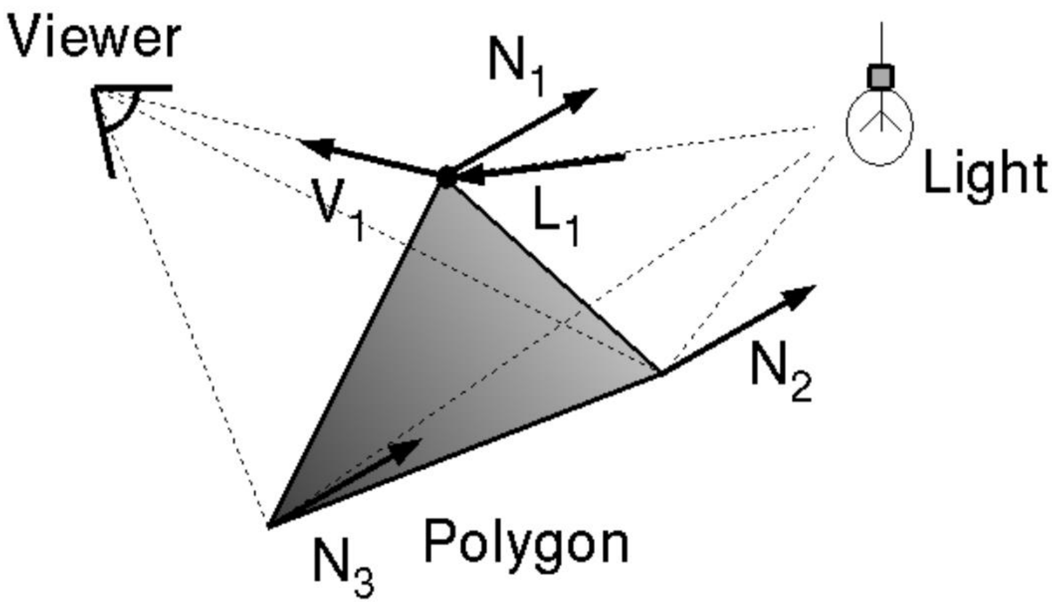
Flat Shading

- Objects look like they are composed of polygons
- Ok for polyhedral objects
- Not so good for smooth surfaces



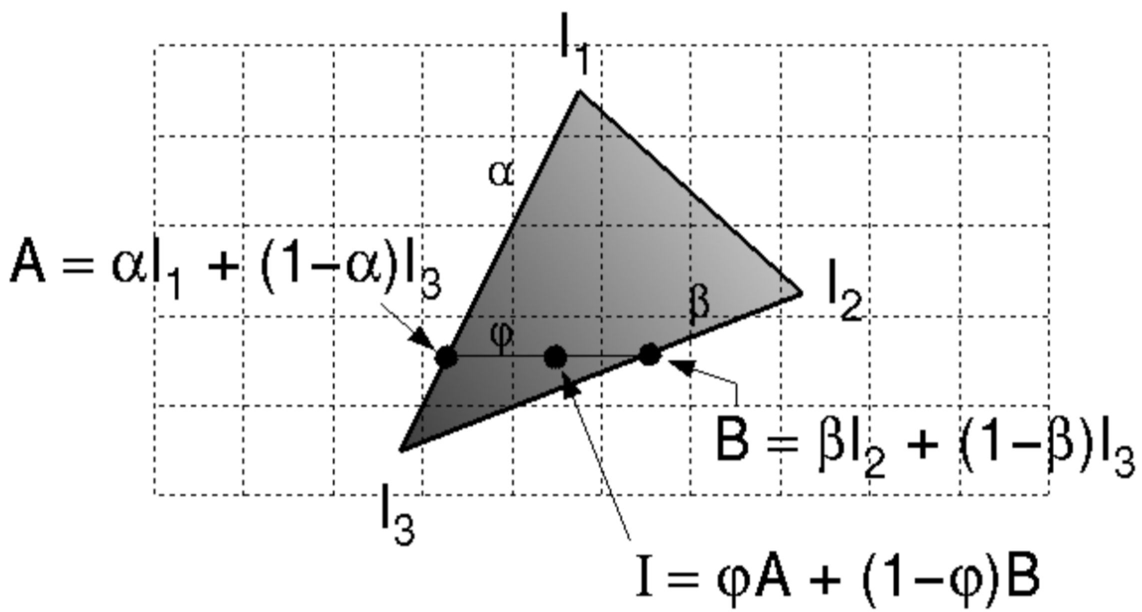
Gouraud Shading

- One lighting calculation per vertex
- Assign pixels inside polygon by interpolating colors computed at vertices



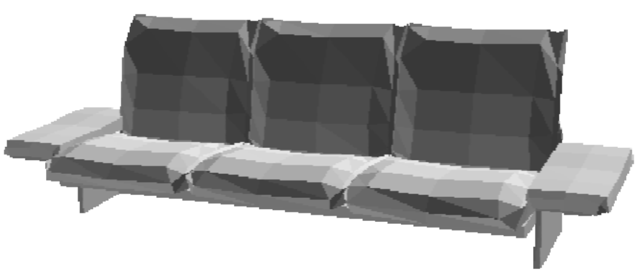
Gouraud Shading

- Bilinearly interpolate colors of triangle across scan line

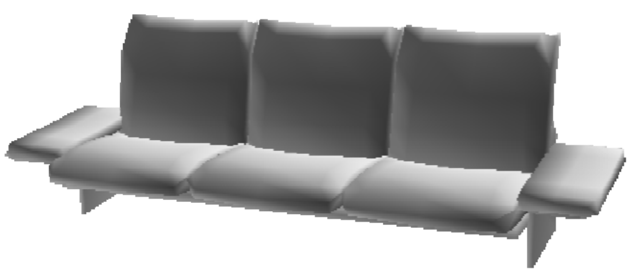


Gouraud Shading

- Smooth shading over adjacent polygons
  - Curved surfaces
  - Illumination highlights
- Produces smoothly shaded polygonal mesh
  - Fast linear approximation
  - Needs fine mesh to capture subtle lighting effects

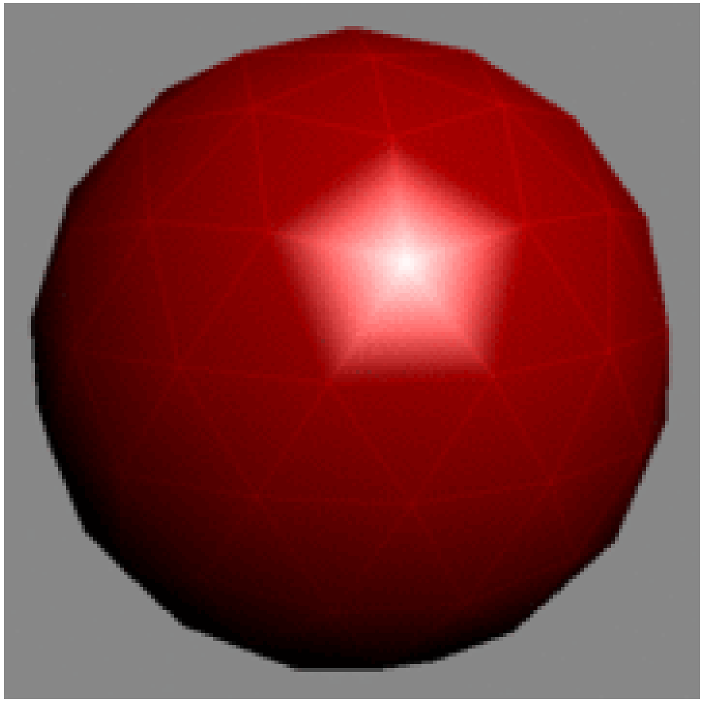


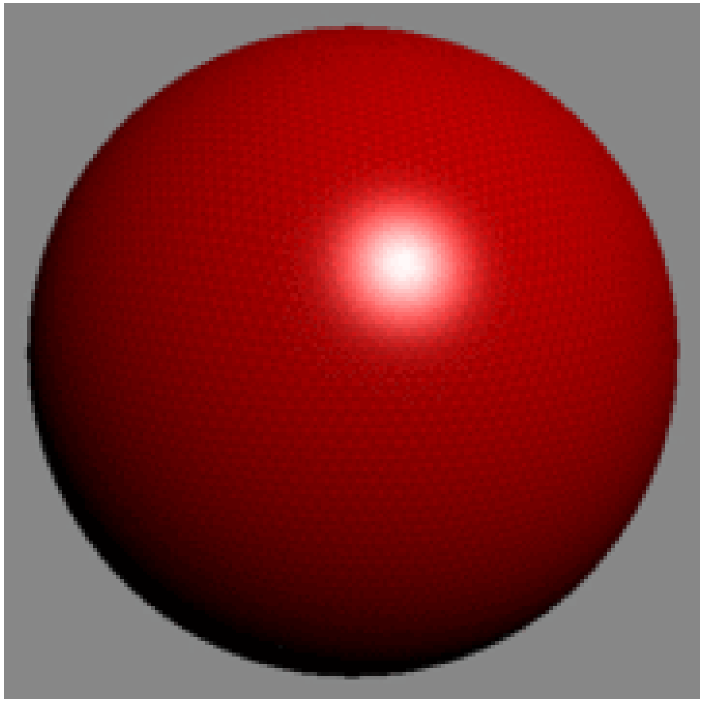
Flat Shading



Gouraud Shading

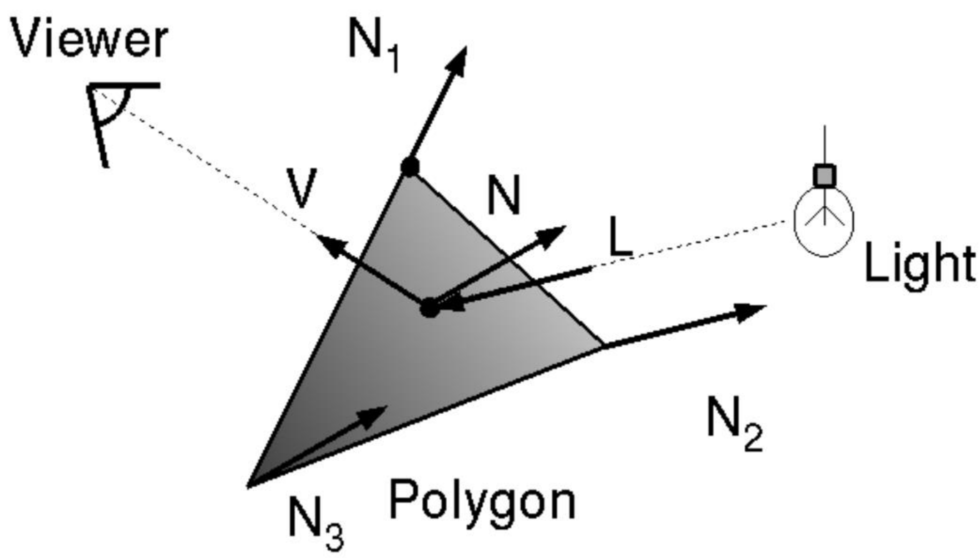
Gouraud Shading





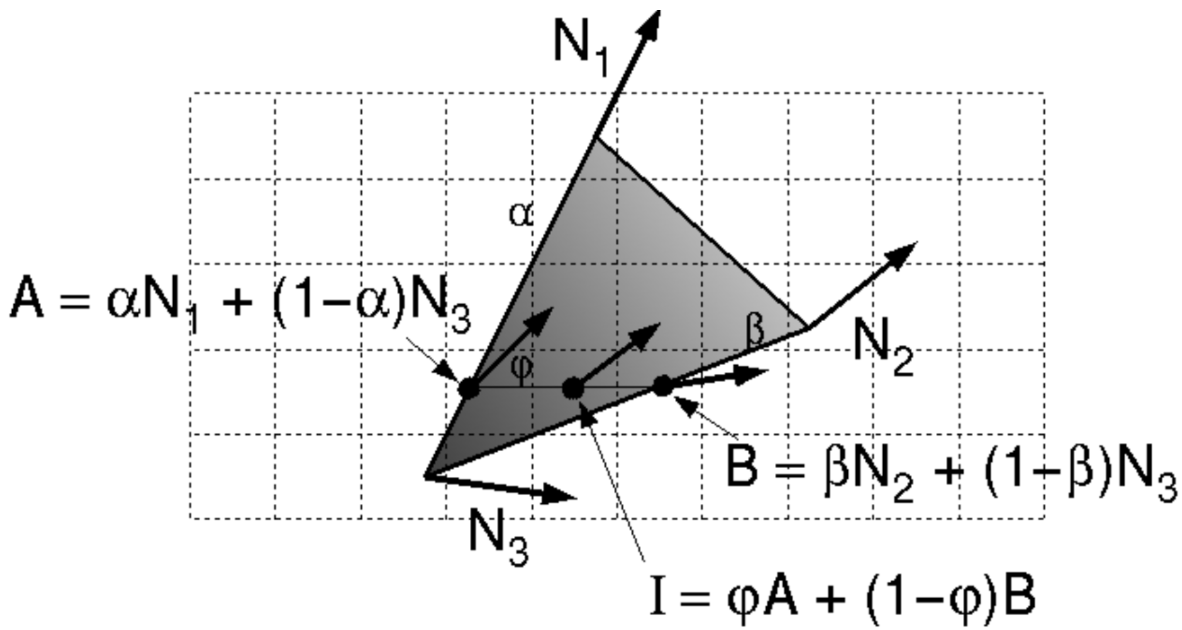
Phong Shading

- NOT Phong lighting model
- One lighting calculation per pixel
  - Approximate surface normals inside polygon using bilinear interpolation of normals from vertices

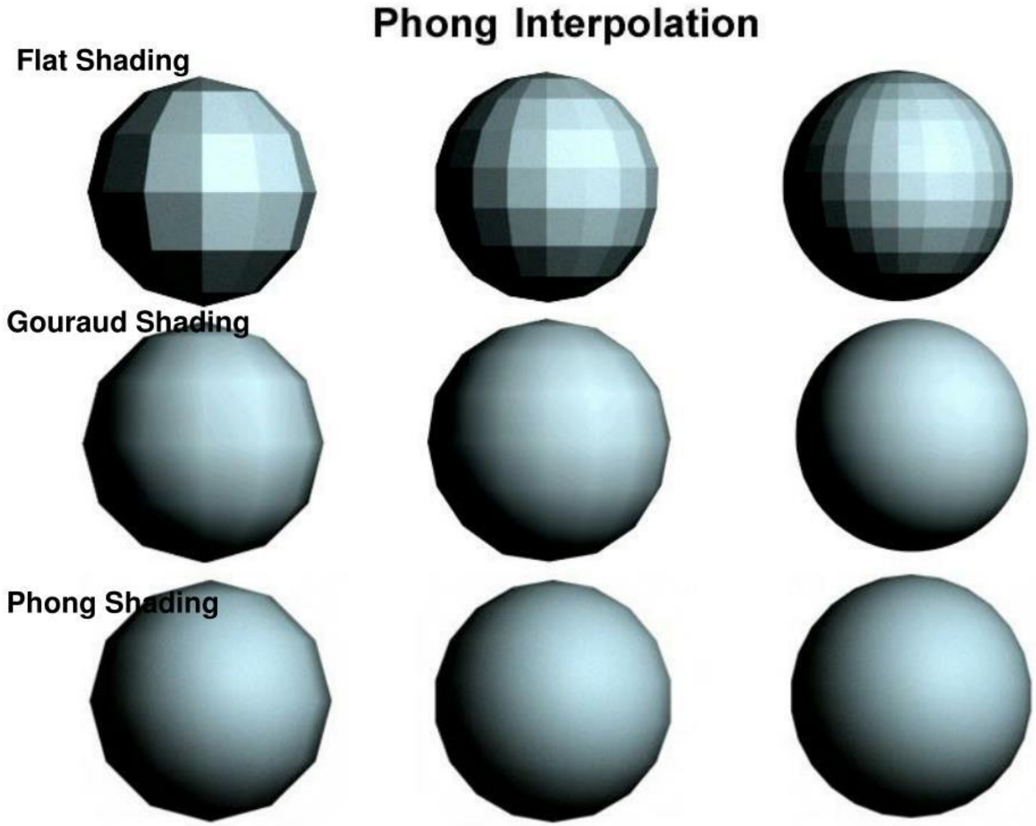


Phong Shading

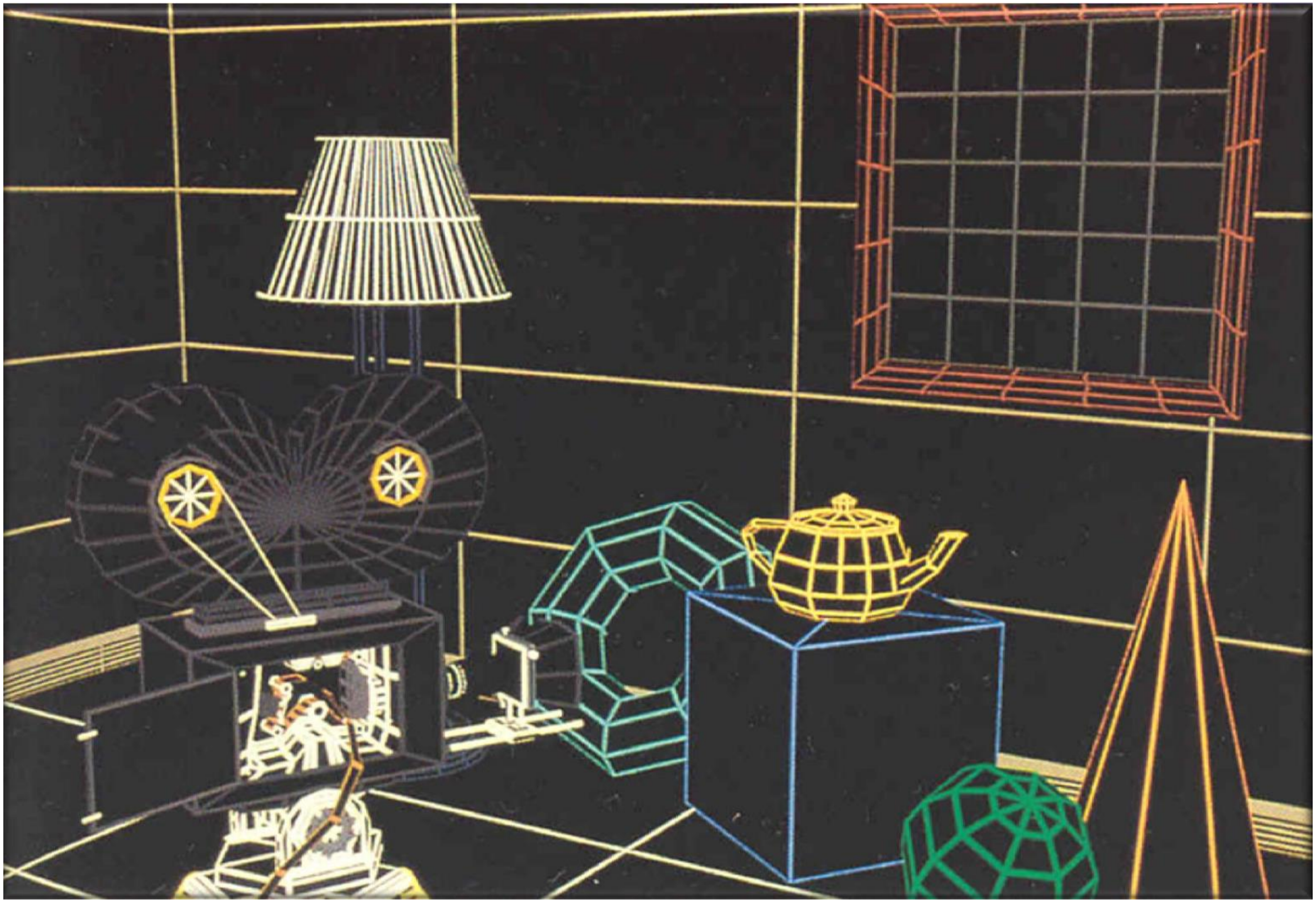
- Bilinearly interpolate surface normals at vertices down and across scan lines



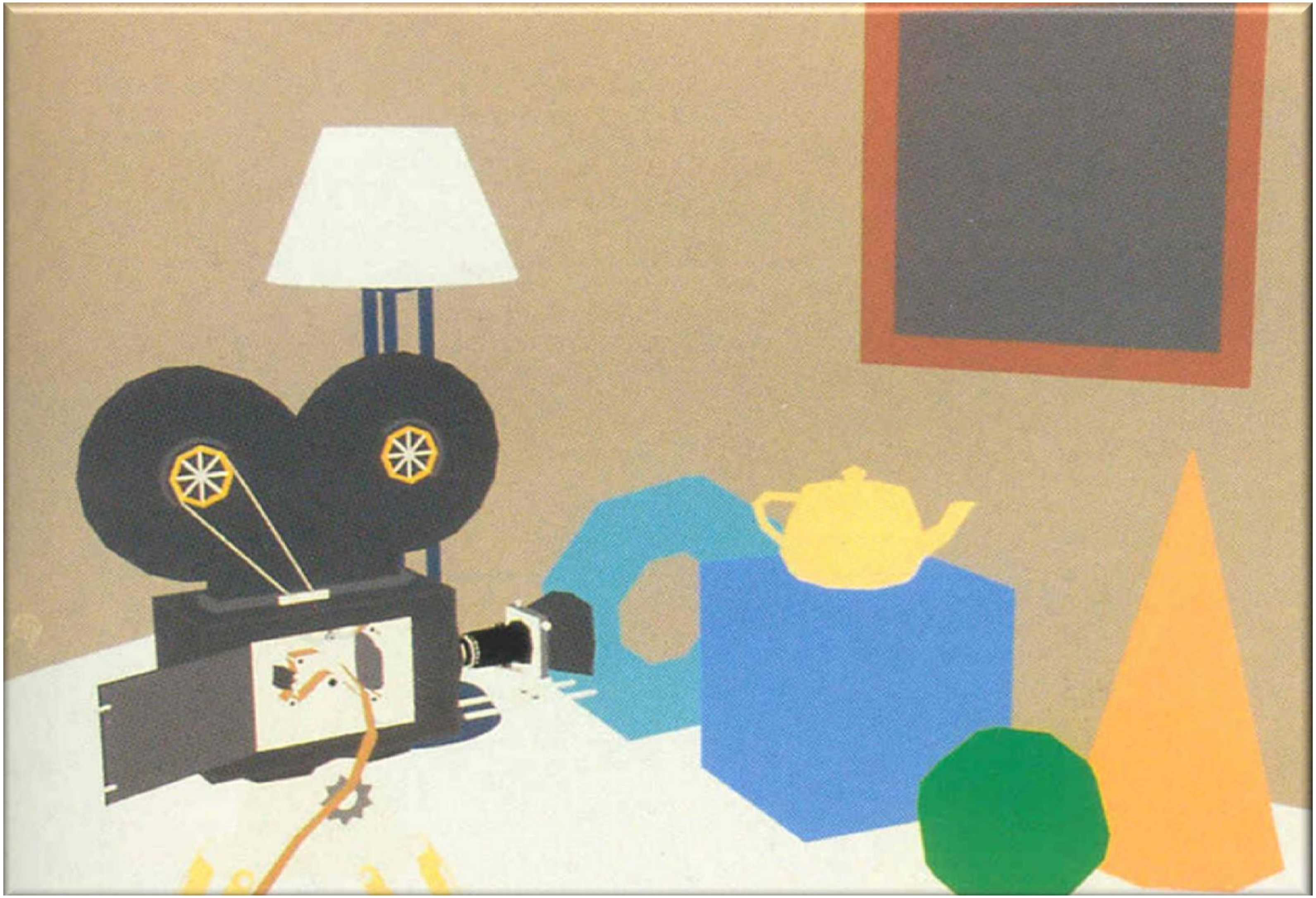
Polygon Shading Algorithms



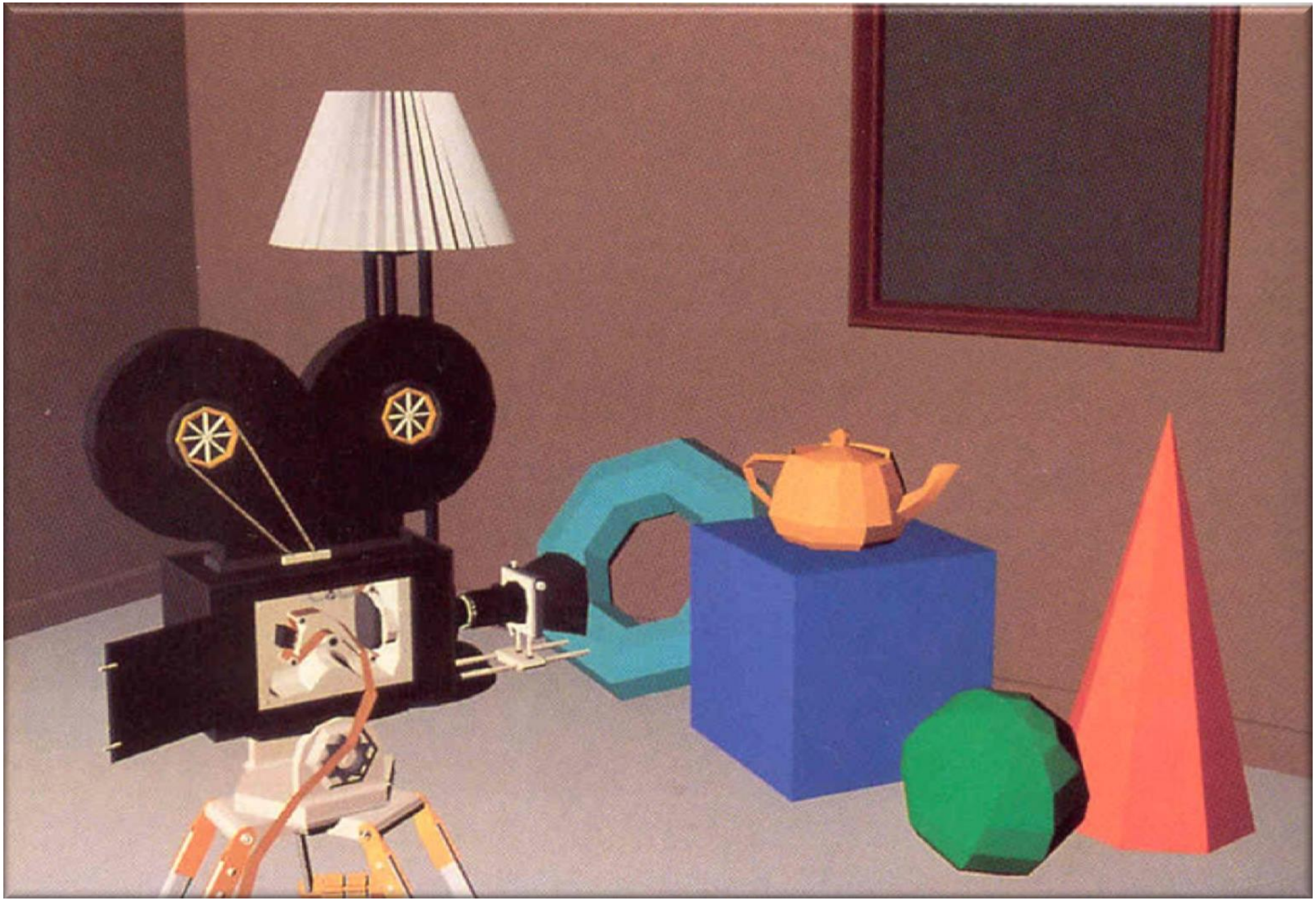
Example: Wireframe scene



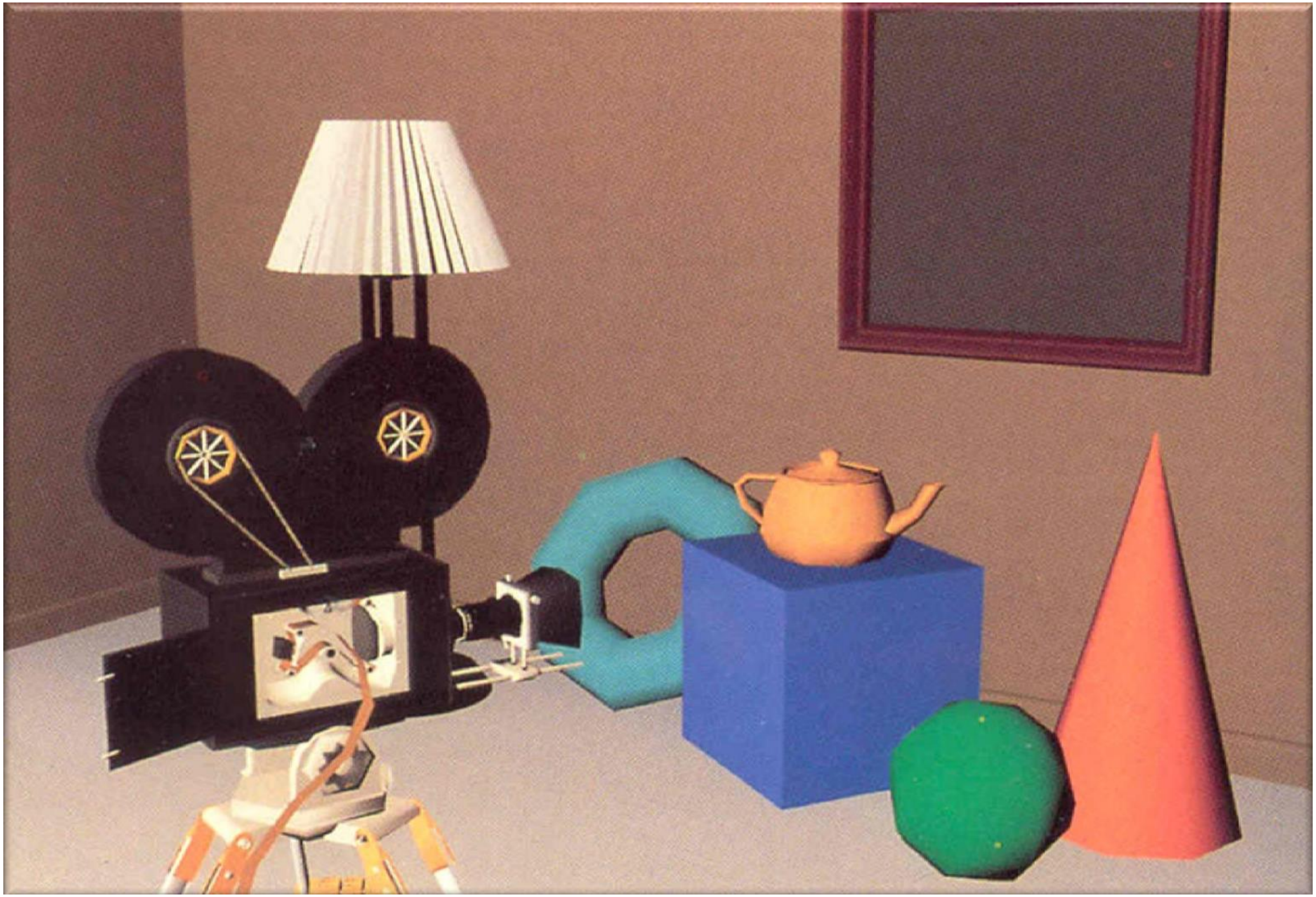
Example: Ambient only



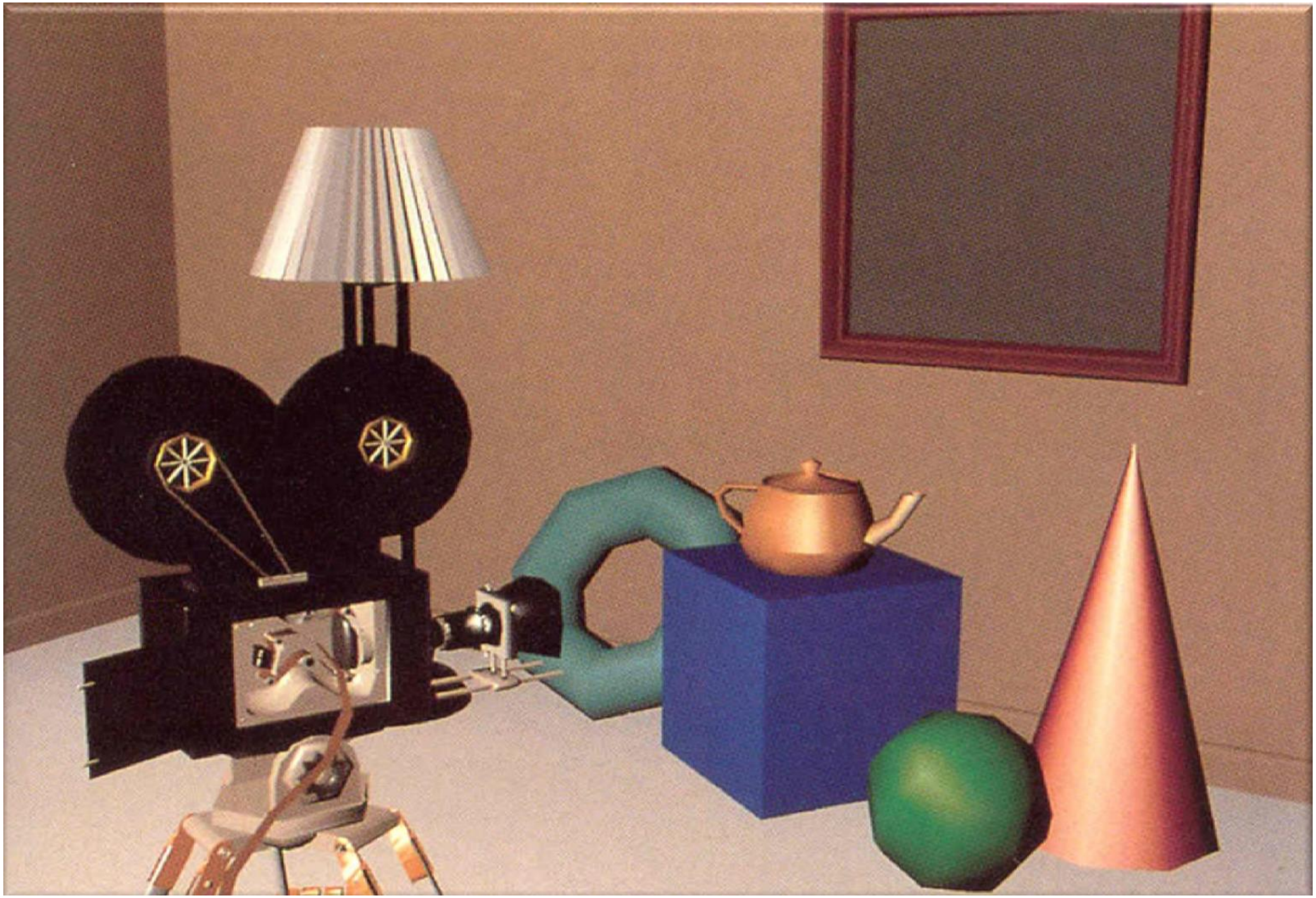
Example: Flat shading with Diffuse



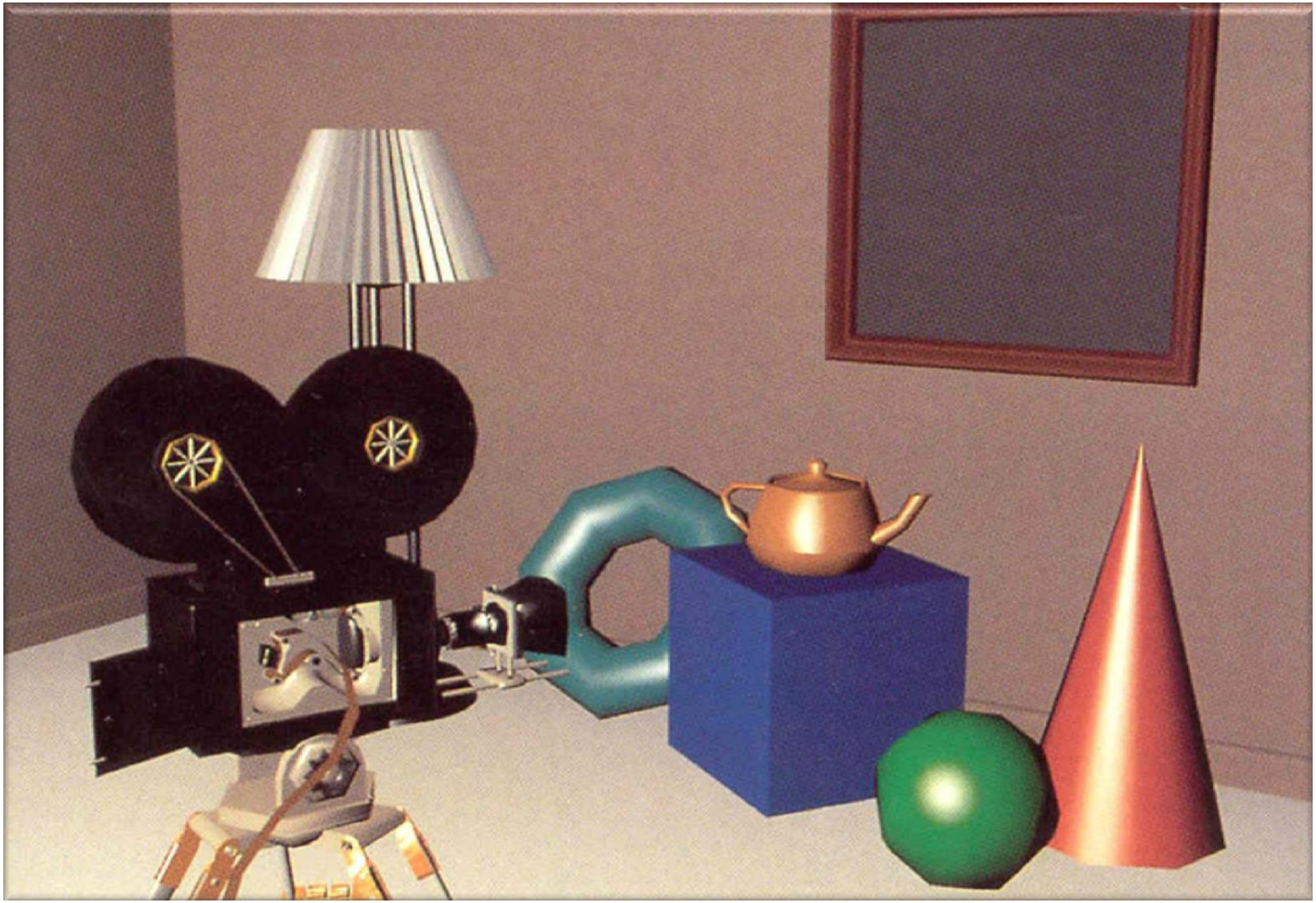
Example: Gouraud shading with Diffuse



Example: Gouraud shading with Specular



Example: Phong shading with Specular



Shading Issues

- Problems with interpolated shading
  - Problems computing shared vertex normals
  - Perspective distortion
  - Problems at T-vertices

Shading Benefits

- Good performance and quality of output
  - Excellent for hardware
  - Works well with subdivision surfaces

How the lectures should look like #2

- Ask questions, please!!!
- Be communicative
- More active you are, the better for you!

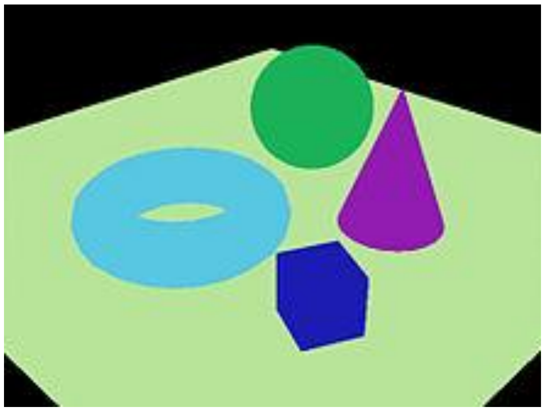
Overview

- Advanced Shading and Mapping
  - Deferred Shading

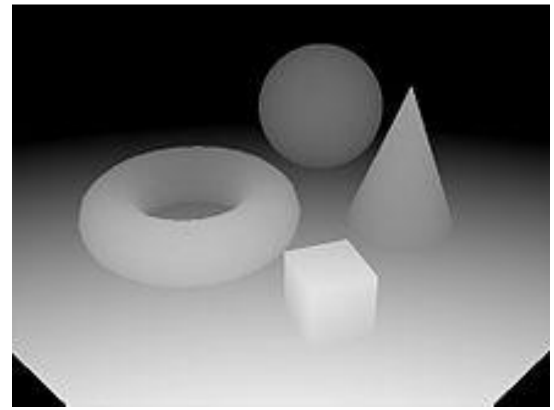
Deferred Shading

- Compute Lighting in Screen-Space
- Two pass approach
- Decoupling of geometry and lighting
- G-Buffer stores positions, normals, materials …
- Lighting is a per-pixel operation
- Problems with transparency and G-buffer size
- O(objects+lights)

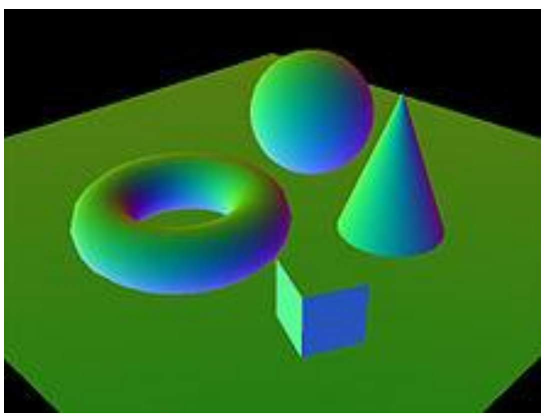
Deferred Shading



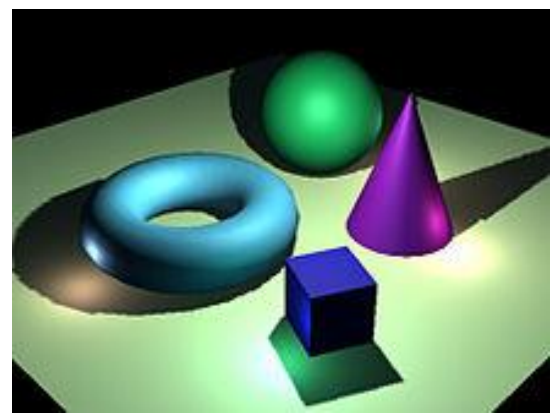
Diffuse Color



Z Buffer



Surface Normals

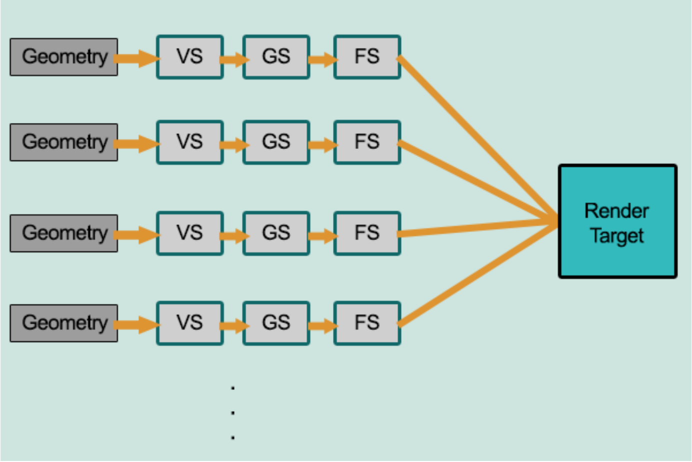


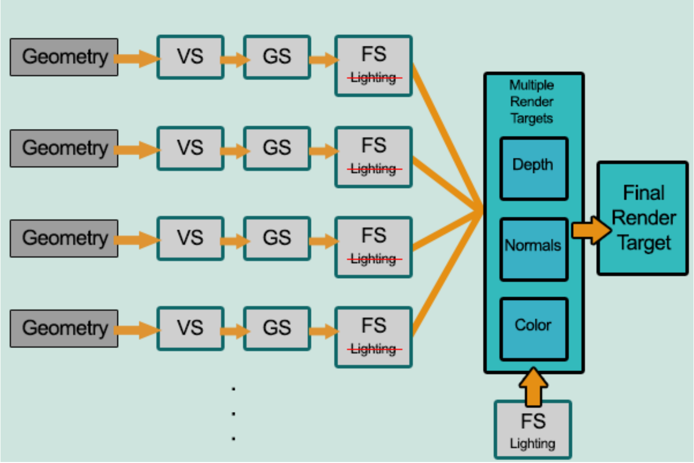
Final Composition

Deferred Shading

Forward rendering

O(num_geometry_fragments * num_lights)





https://gamedevelopment.tutsplus.com/articles/forward-rendering-vs-deferred-rendering--gamedev-12342

Deferred shading

O(screen_resolution * num_lights)

Deferred Shading Pros/Cons

- Pros
  - The primary advantage of deferred rendering is that the lighting is only computed for fragments that are visible
  - In forward rendering, each fragment requires a loop over all light sources and for each, an evaluation of the lighting model
  - Deferred rendering provides better scalability as the number of light sources increases.
- Cons
  - The primary disadvantage of deferred rendering is the additional storage for the g-buffer
  - This becomes a difficulty when different materials are required for different objects in the scene

Next Lecture

Visibility, Culling

Acknowledgements

- Thanks to all the people, whose work is shown here and whose slides were used as a material for creation of these slides:

Matej Novotný, GSVM lectures at FMFI UK

Peter Drahoš, PPGSO lectures at FIIT STU

Output of all the publications and great team work

Very best data from 3D cameras

Questions ?!

Skeletex
RESEARCH

www.skeletex.xyz

madaras@skeletex.xyz

martin.madaras@fmph.uniba.sk