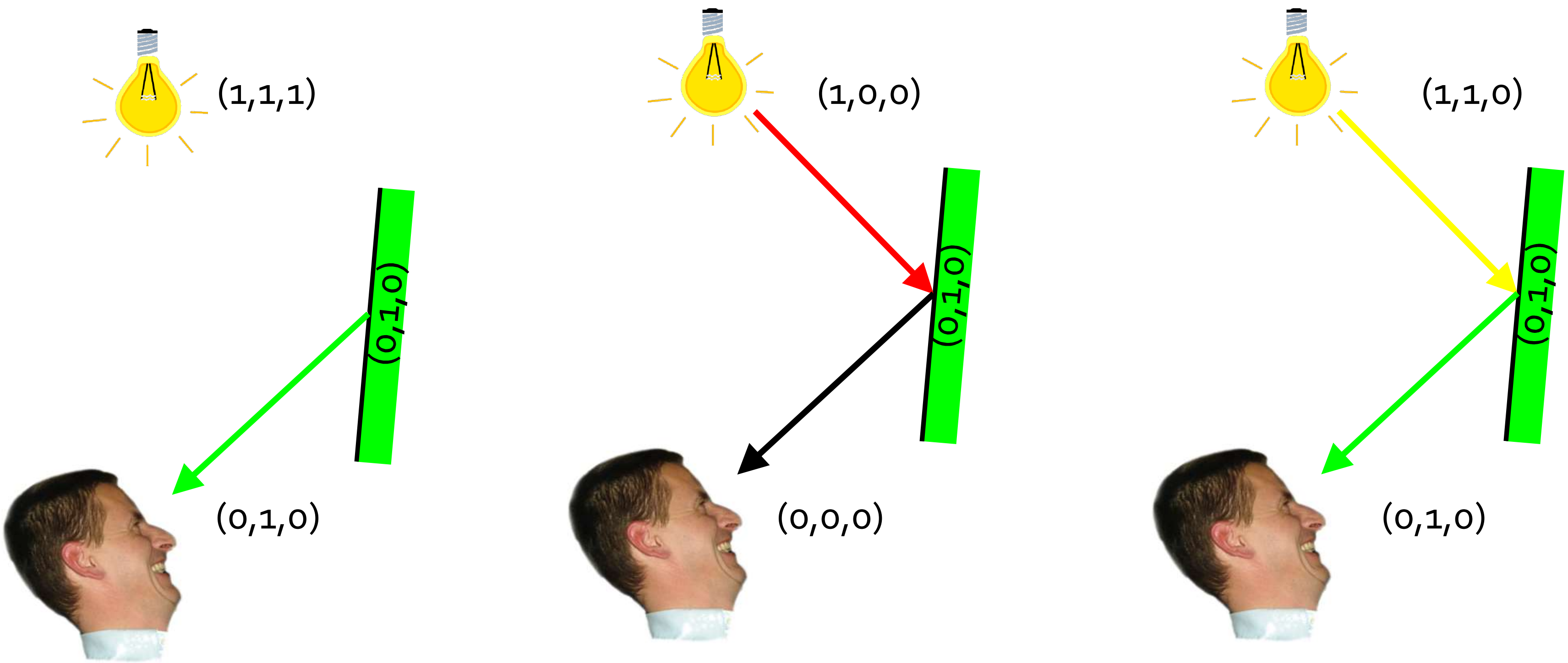
Lighting

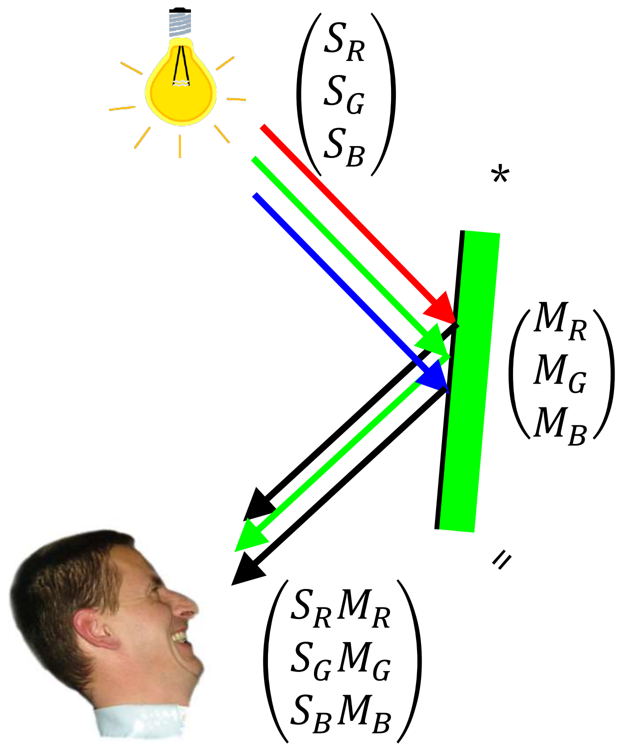
Light/Material Interaction



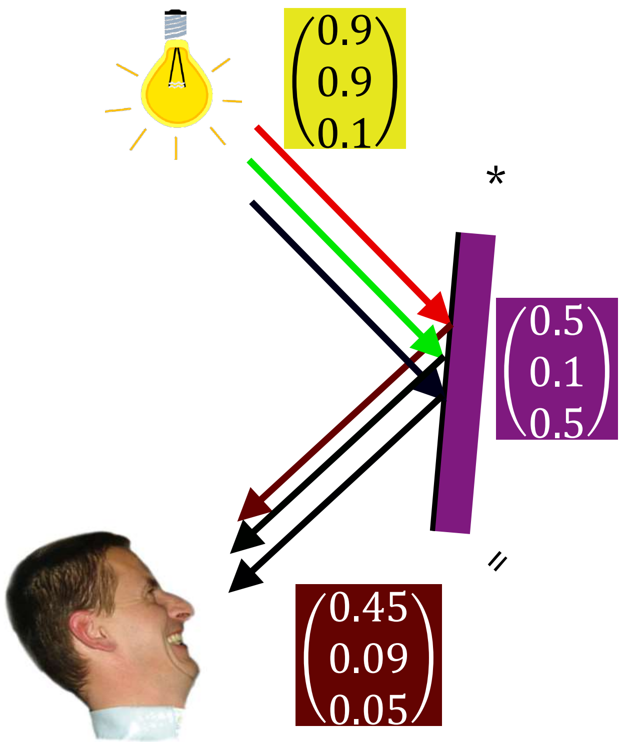
Color Multiplication – "*" Operator

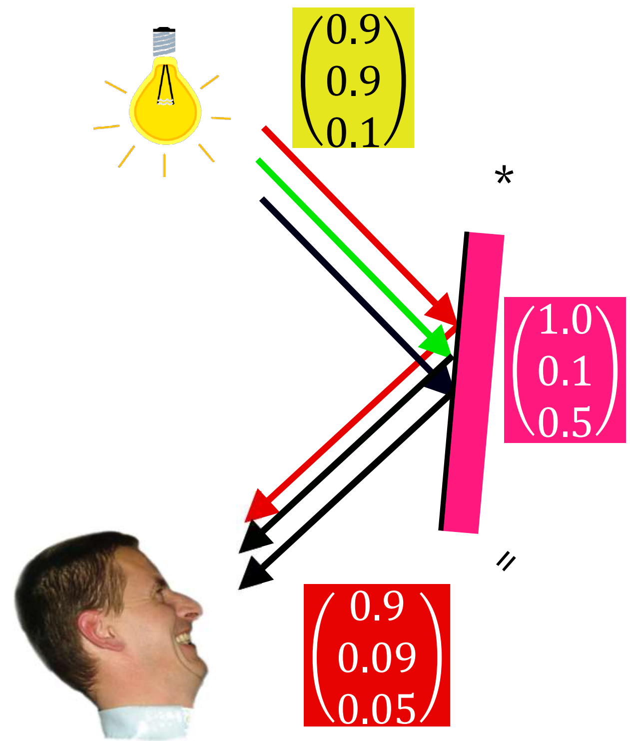
- Color channels are independent
- S…light color
- M…material color

$$\begin{pmatrix} S_R \\ S_G \\ S_B \end{pmatrix} * \begin{pmatrix} M_R \\ M_G \\ M_B \end{pmatrix} := \begin{pmatrix} S_R M_R \\ S_G M_G \\ S_B M_B \end{pmatrix}$$

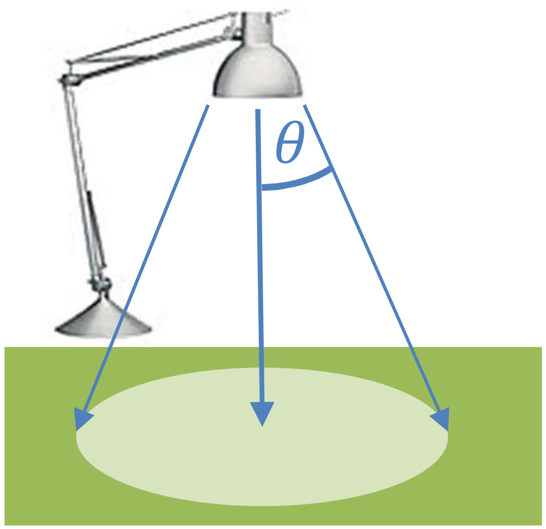


Color Multiplication – "*" Operator

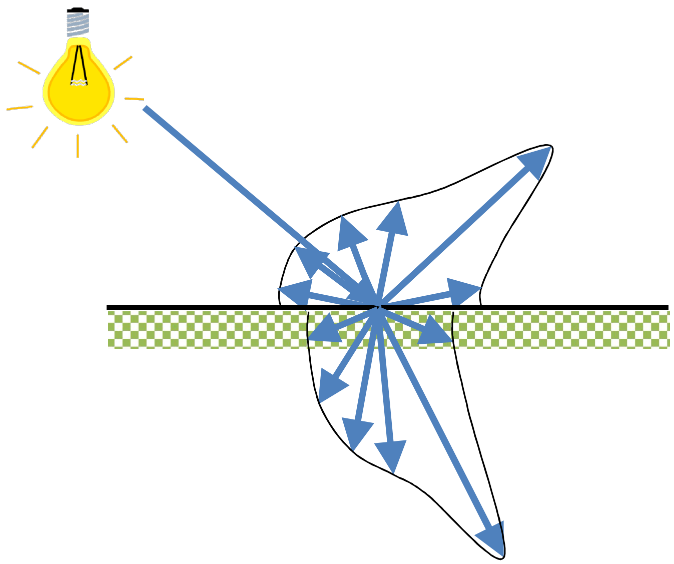




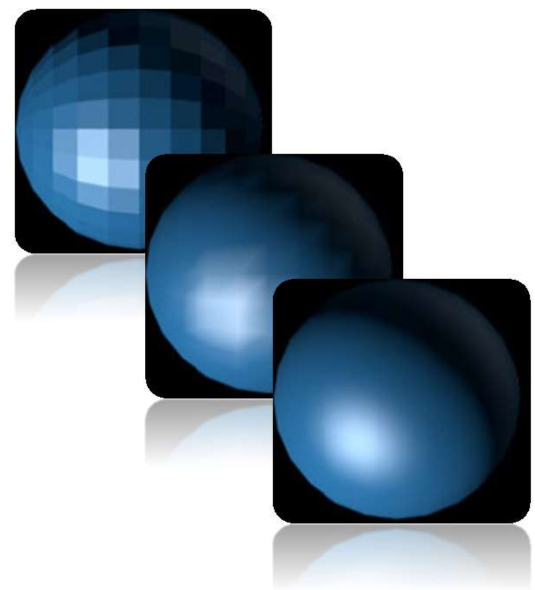
Light Sources



Reflection Models

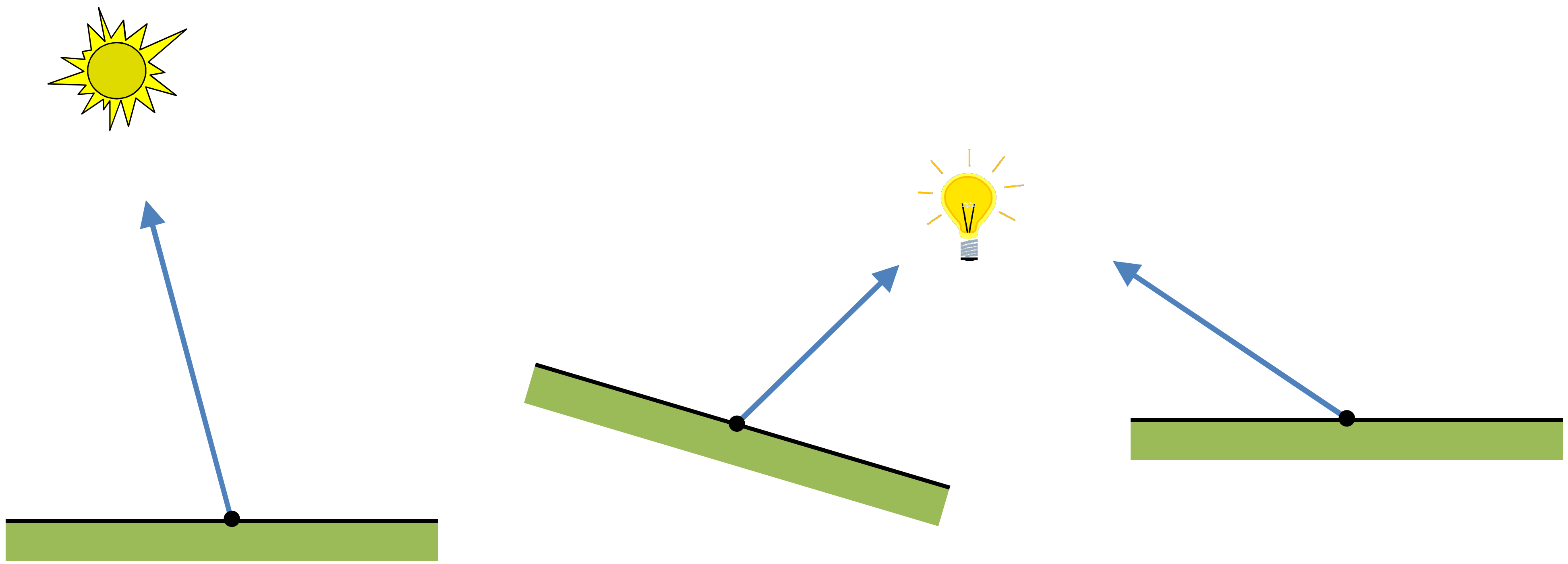


Shading Models



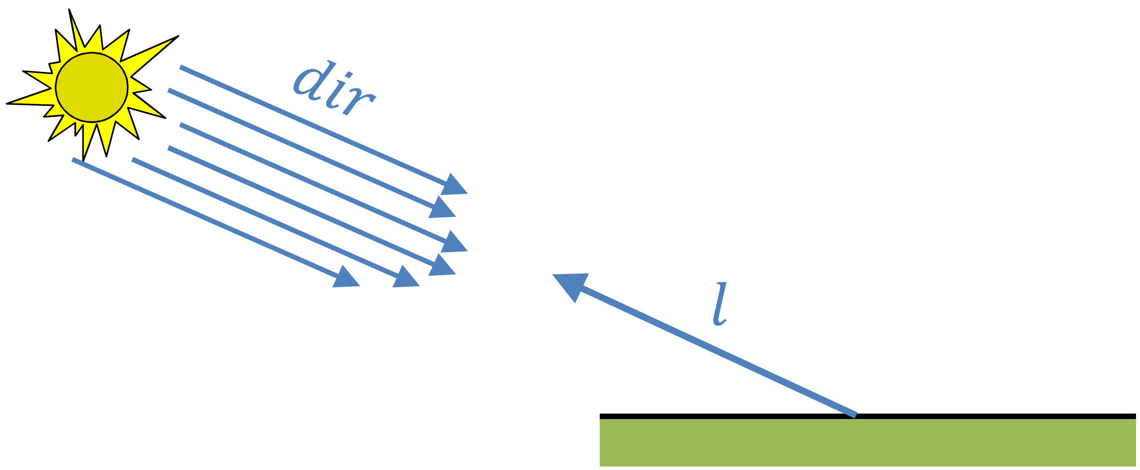
Light Sources

- Where is the light coming from?



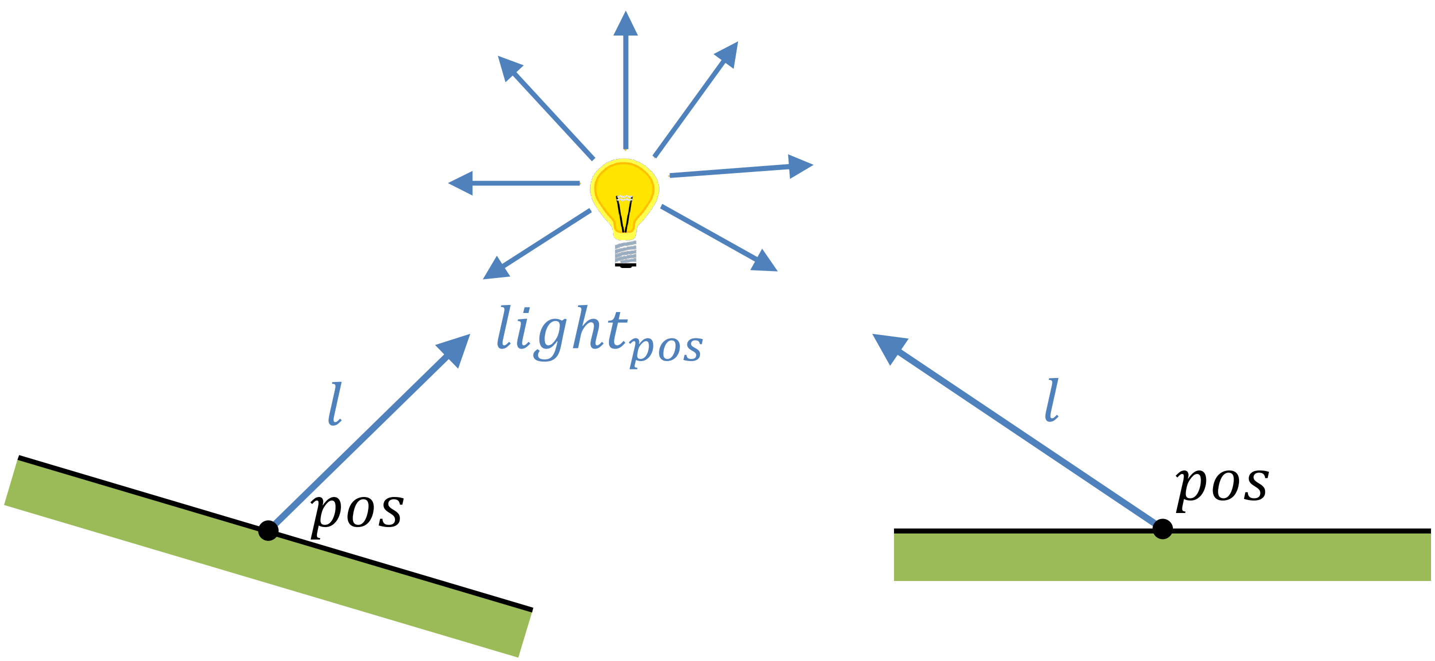
Directional Light

- Light source is infinitely far away
- Light rays are parallel, like sun
- l … direction to the light $= -dir$



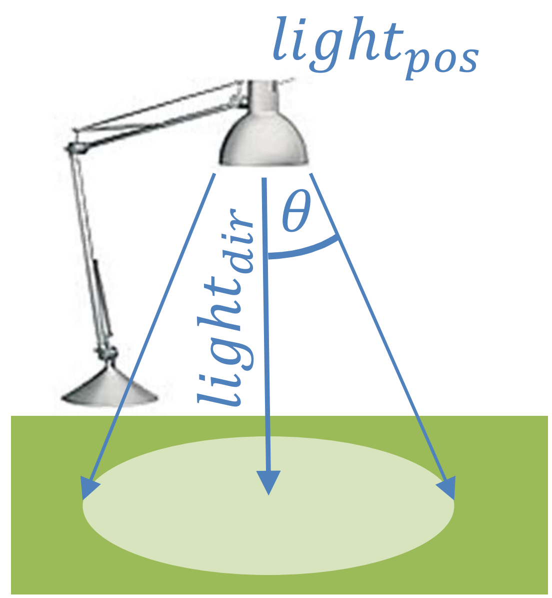
Point Light

- Has a certain position in space $l = \text{normalize}(light_{pos} - pos)$



Spot Light

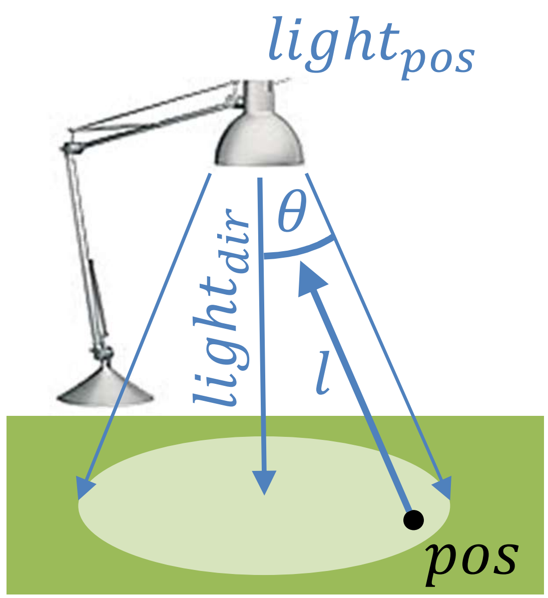
- Has a certain position and cone in space
- Cone can be specified by opening angle and central direction



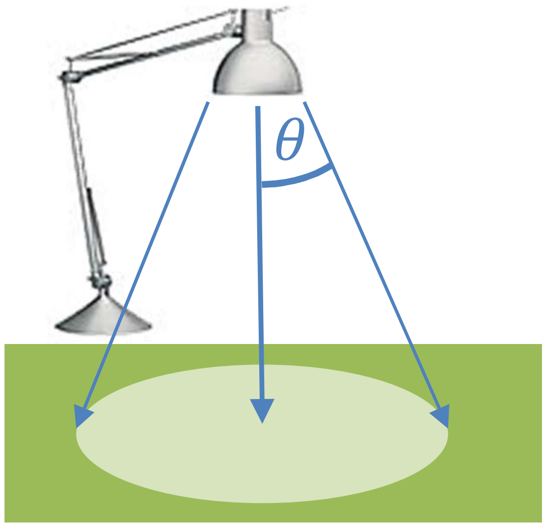
Spot Light

- Point is in cone iff

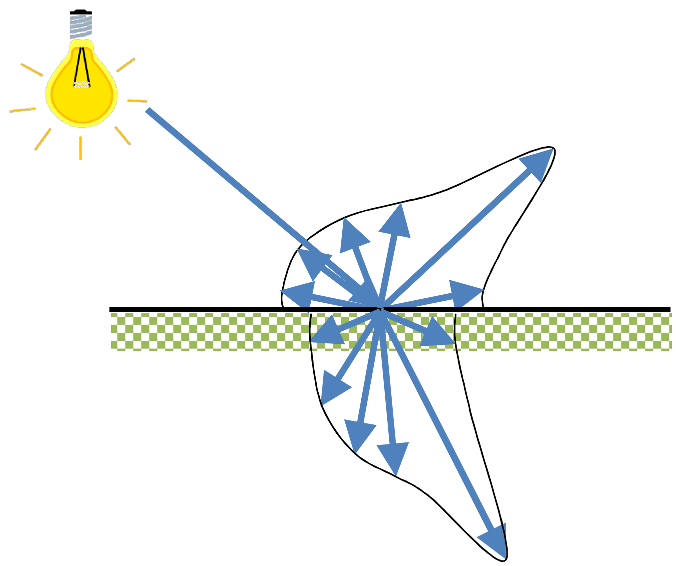
$$\cos^{-1} dot(l, -light_{dir}) < \theta$$



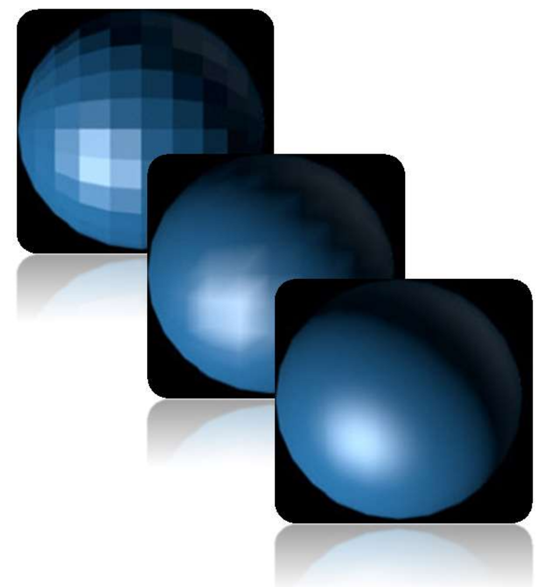
Light Sources



Reflection Models



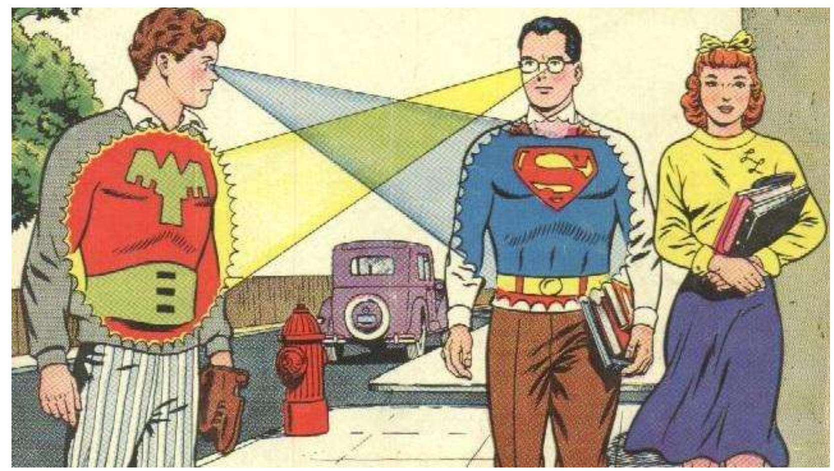
Shading Models



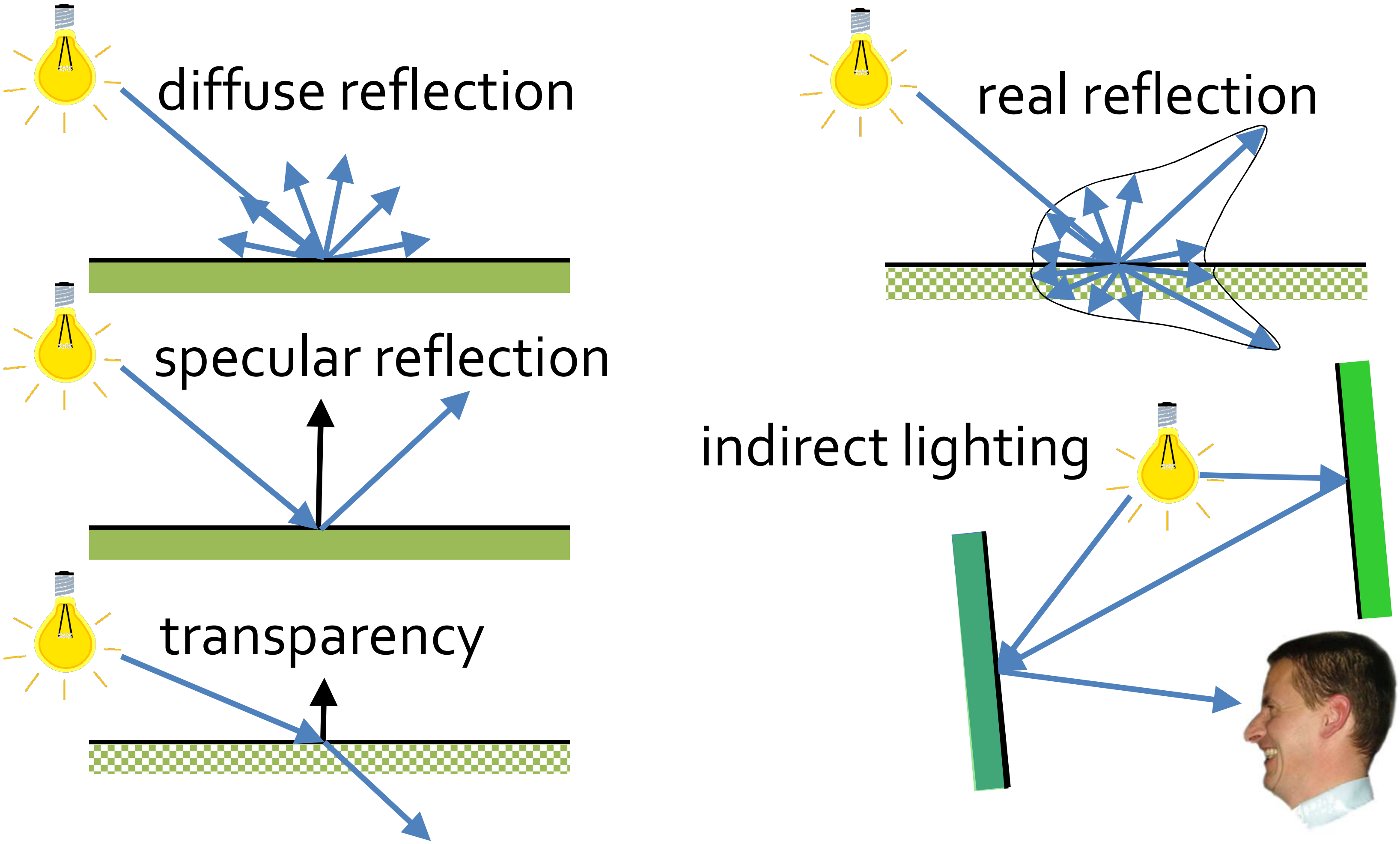
Reflection/Illumination/Lighting model

- How is light reflected by a surface?
- What is the resulting color?
- What properties can we simulate with a given model?
- What lighting effects can we create?

Glossy → Matte



Surface lighting effects



Lighting Models

photorealistic
reflection models

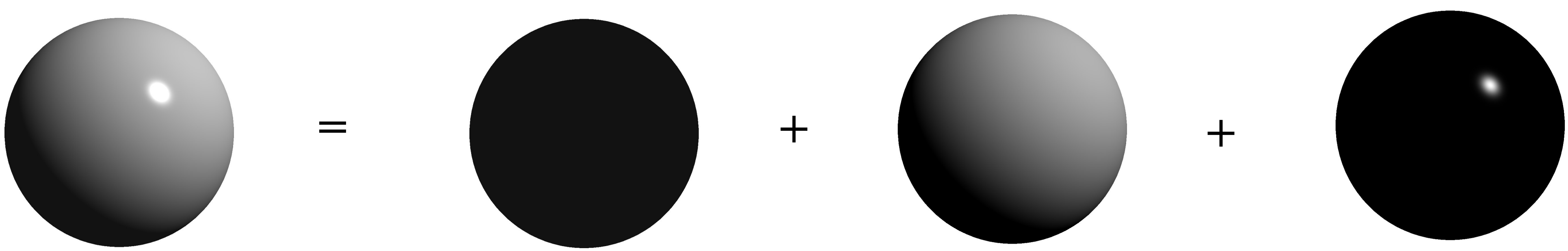
non-photorealistic

local

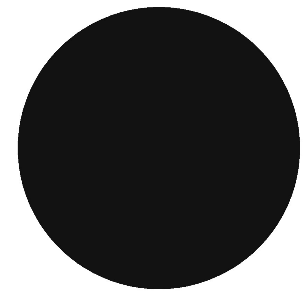
global

Phong Illumination Model

- $\text{Color} = ambient + diffuse + specular$
- $component$ … lighting model component



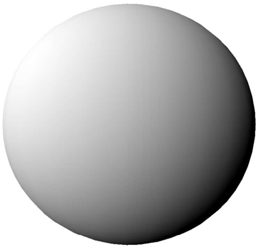
Ambient Light Reflection



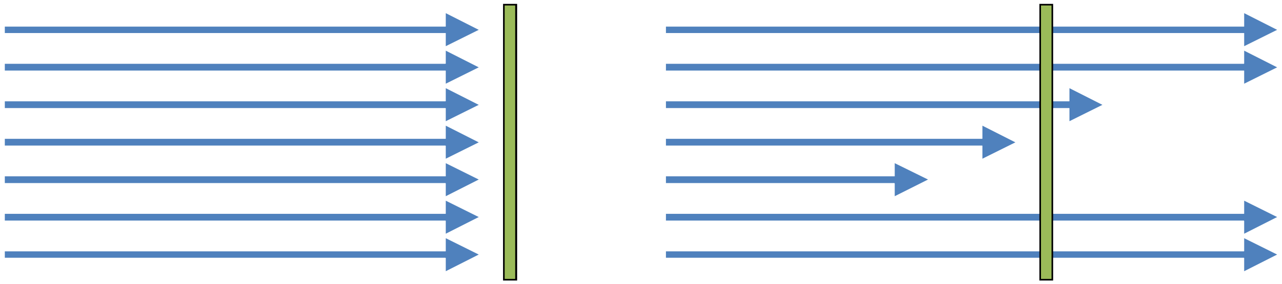
- Background light
  - No direction because scattered so often
  - “Color inside shadow”
  - Approximation of global diffuse lighting effects
- $S_{ambient}$…background light color
- M…material color

$$ambient = M * S_{ambient}$$

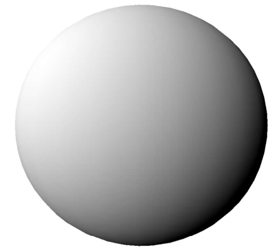
Diffuse Light Reflection



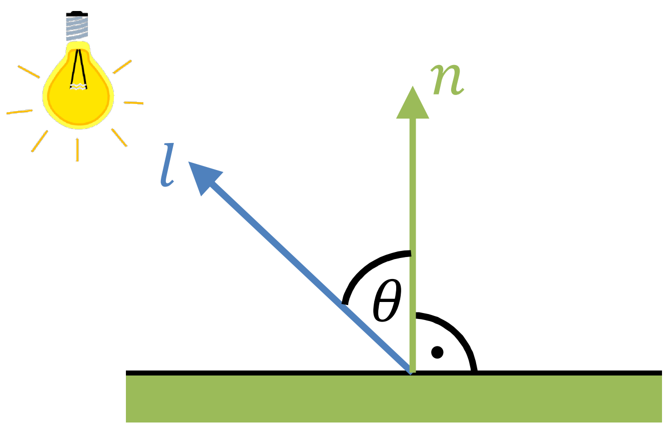
- The flatter light falls on a surface,
  the darker it will appear
- Ideal diffuse reflectors (Lambertian reflectors)
- Brightness depends on orientation of surface



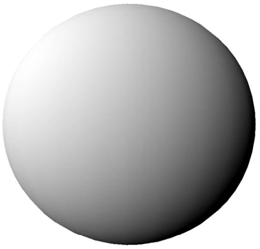
Diffuse Light Reflection



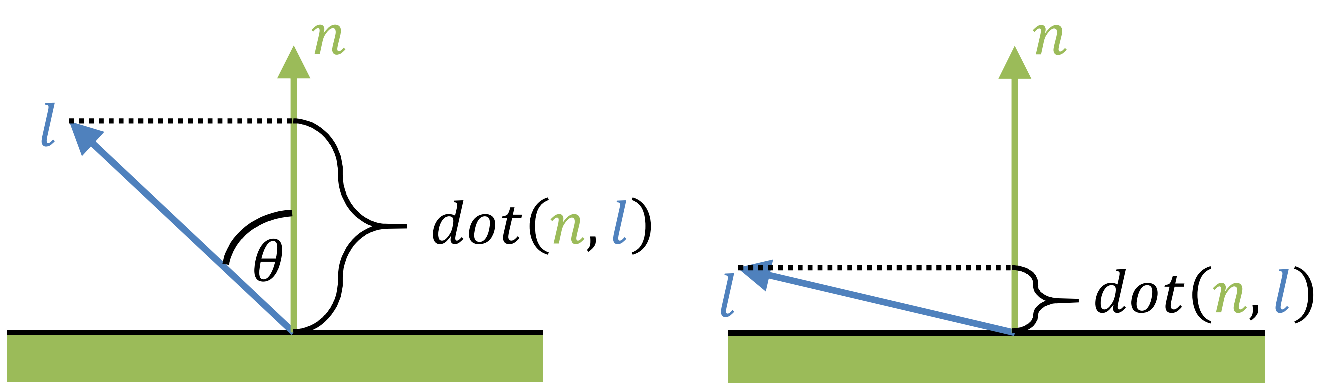
- Diffuse brightness is dependent on angle between
  n ... surface normal and
  l ... direction to the light



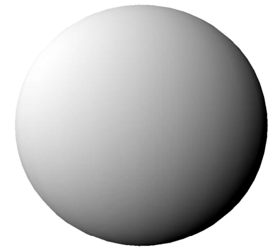
Diffuse Light Reflection



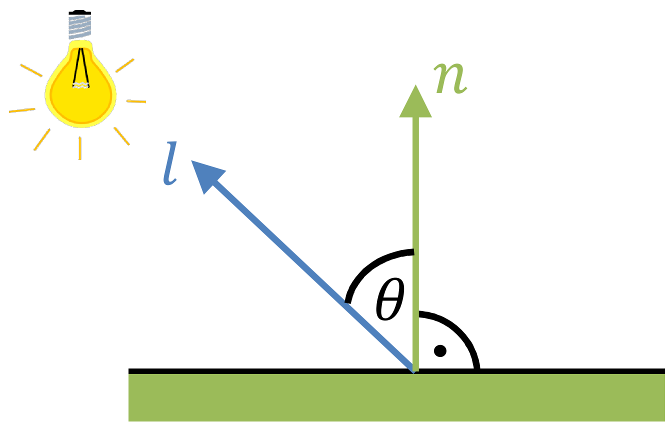
- Diffuse brightness is dependent on angle between
  n … surface normal and
  l … direction to the light
- $\cos\theta = dot(n, l)$



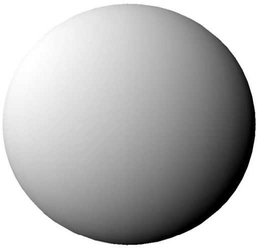
Diffuse Light Reflection



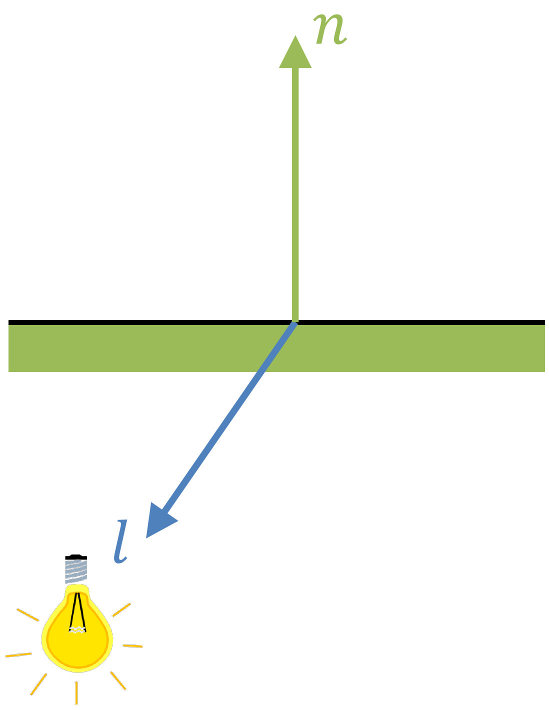
- S…light color
- M…material color
- $diffuse = M * S \cdot \mathrm{dot}(n, l)$



Light from Behind

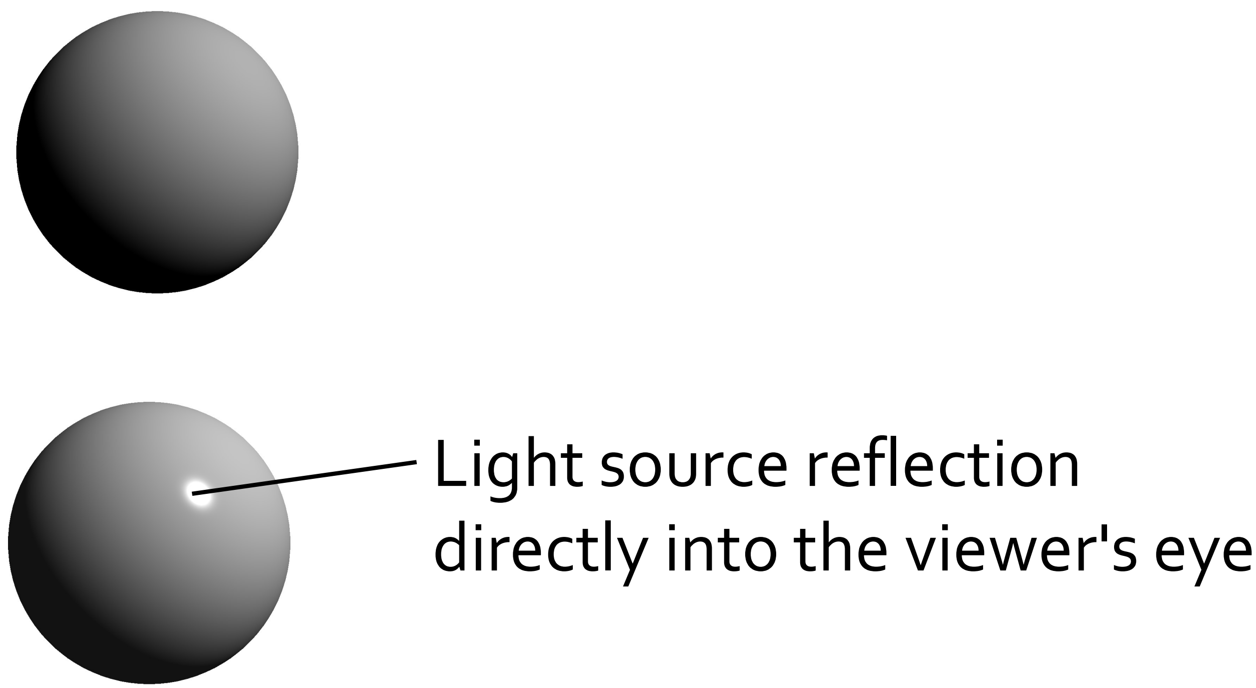


- Should be ignored
- $diffuse = M * S \cdot \max(0, \mathrm{d}ot(n, l))$



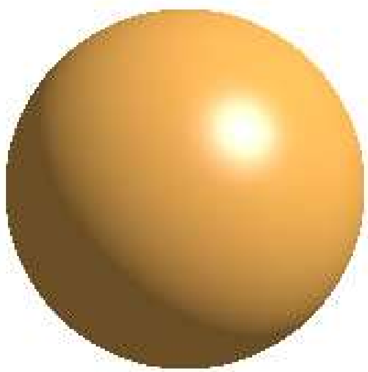
Lambertian (Diffuse) Reflection

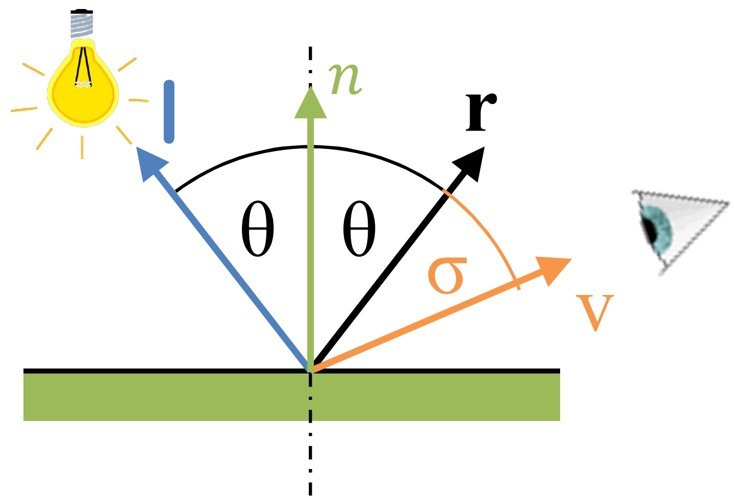
Specular Highlights



Specular Reflection Model

- Reflection of incident light around specular-reflection angle

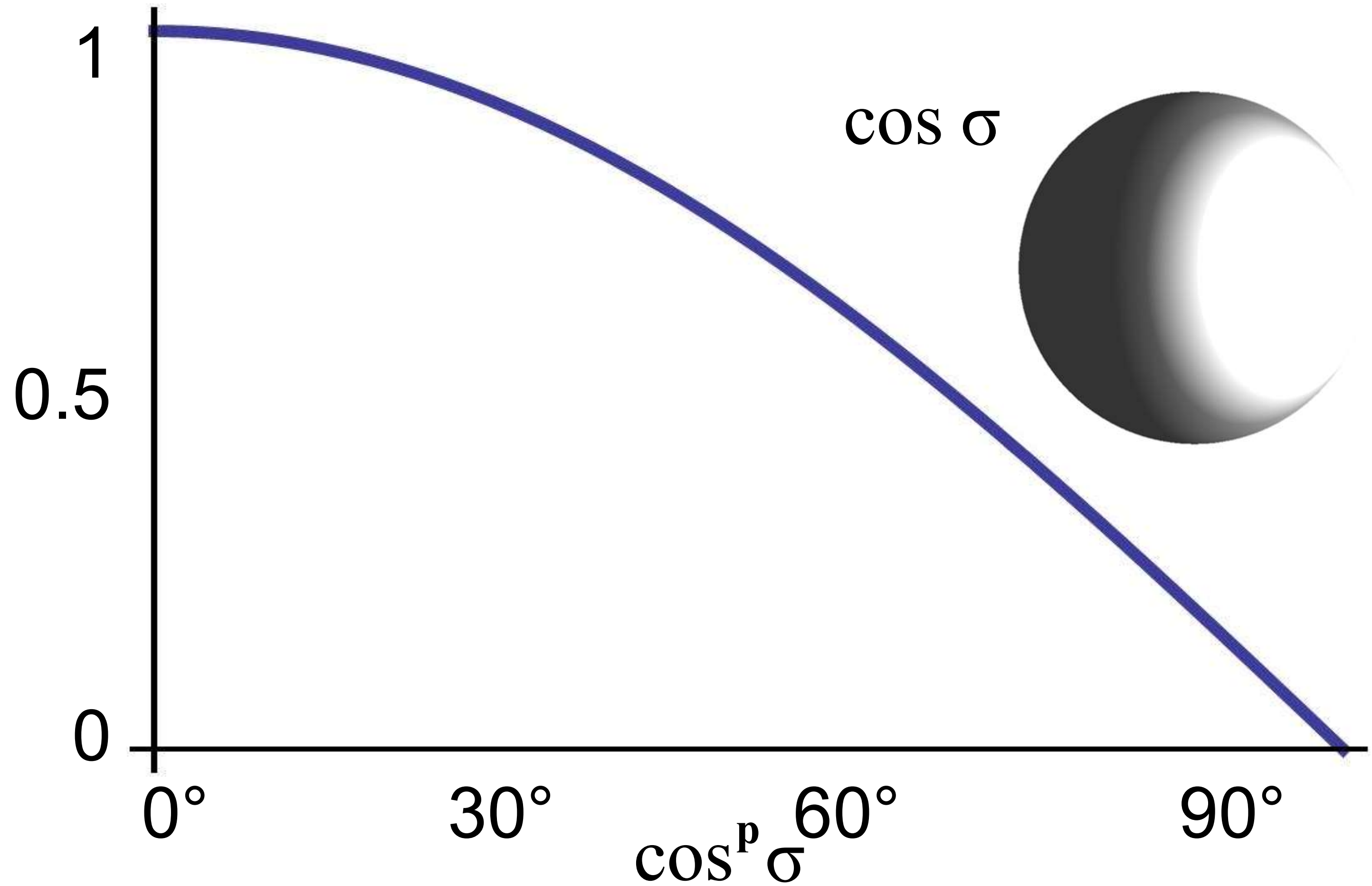




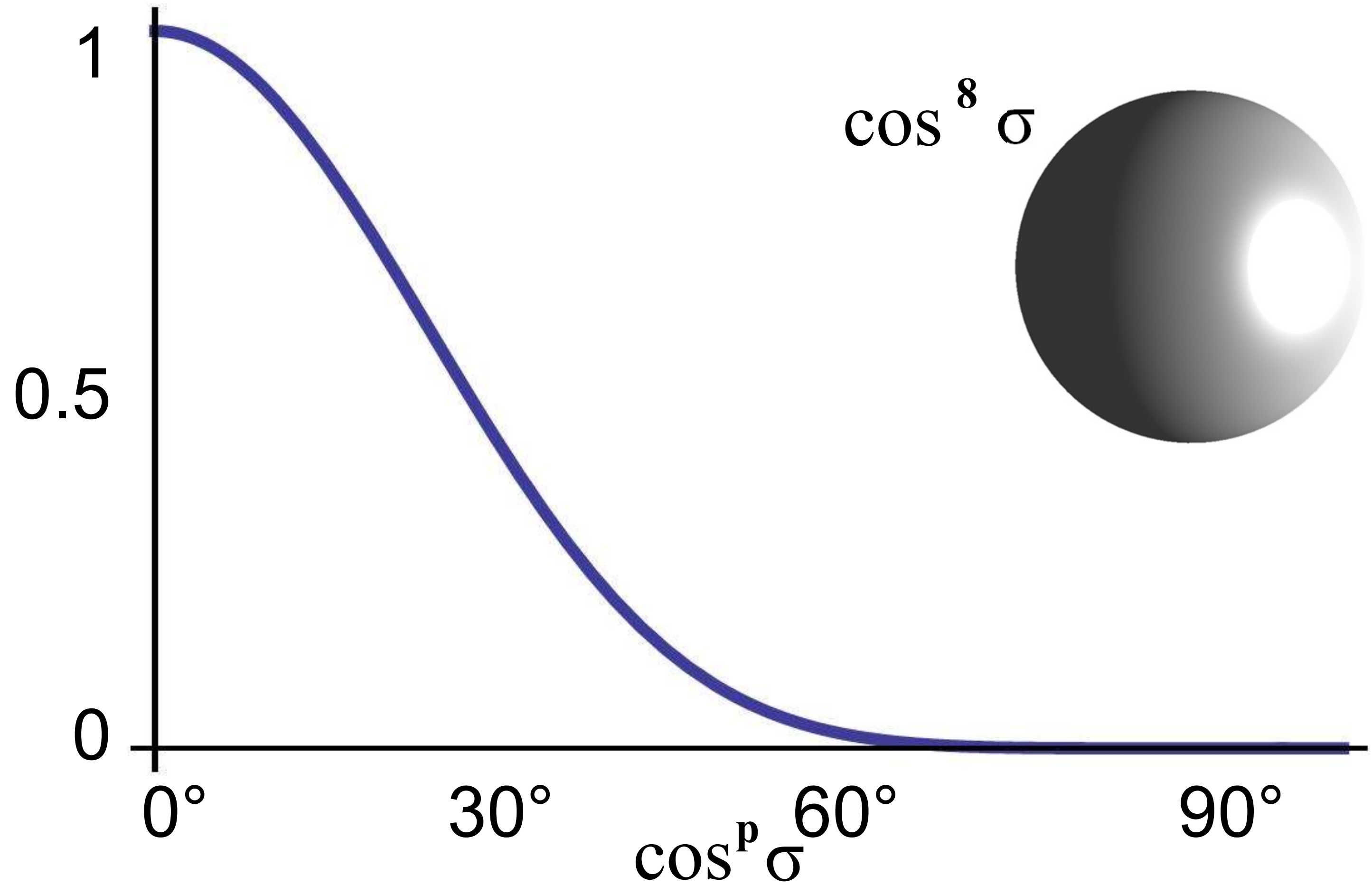
- Empirical Phong model

$$L_{spec} = M*S \bullet \cos^p \sigma$$

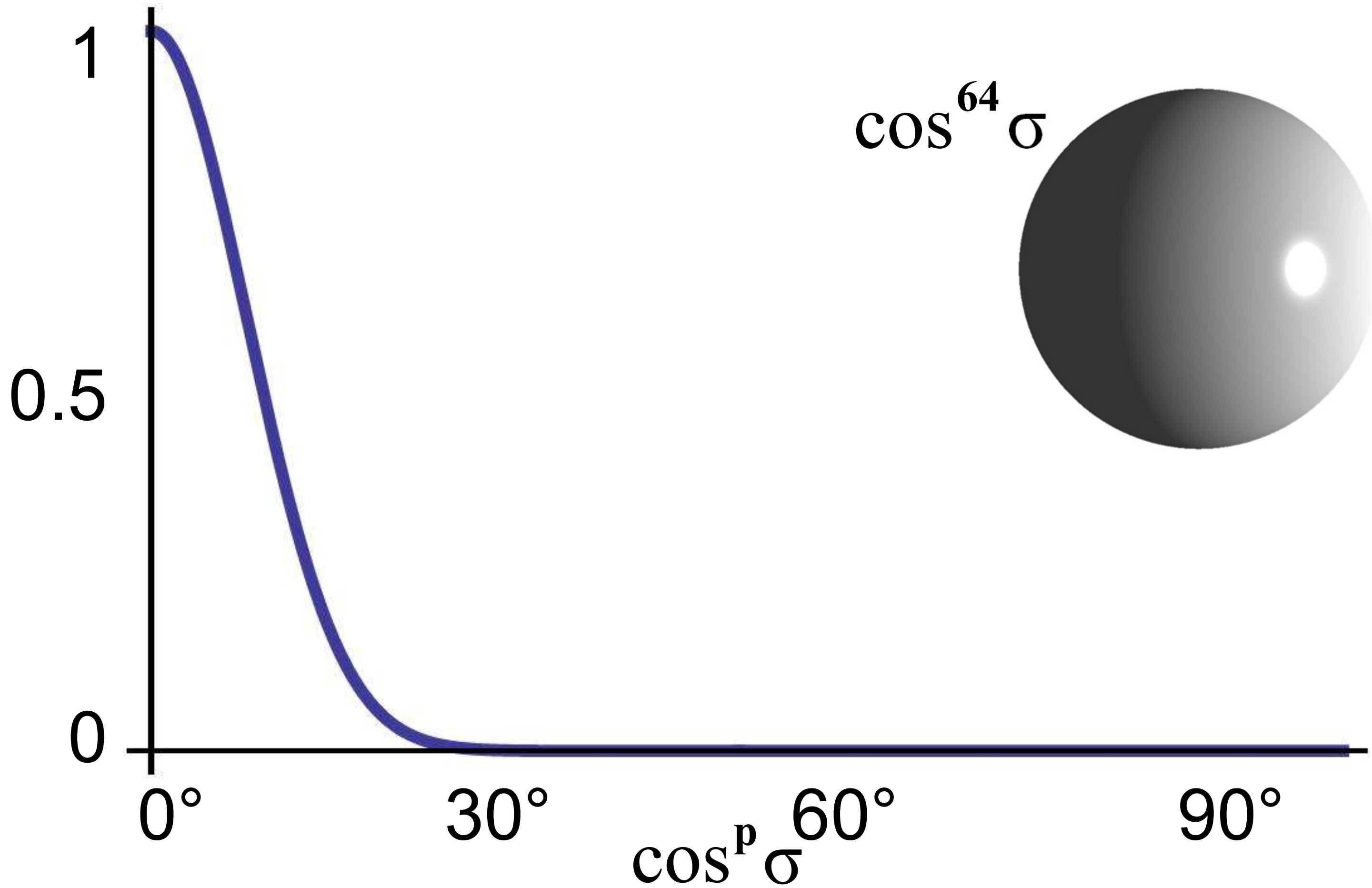
Specular Reflection Coefficient p



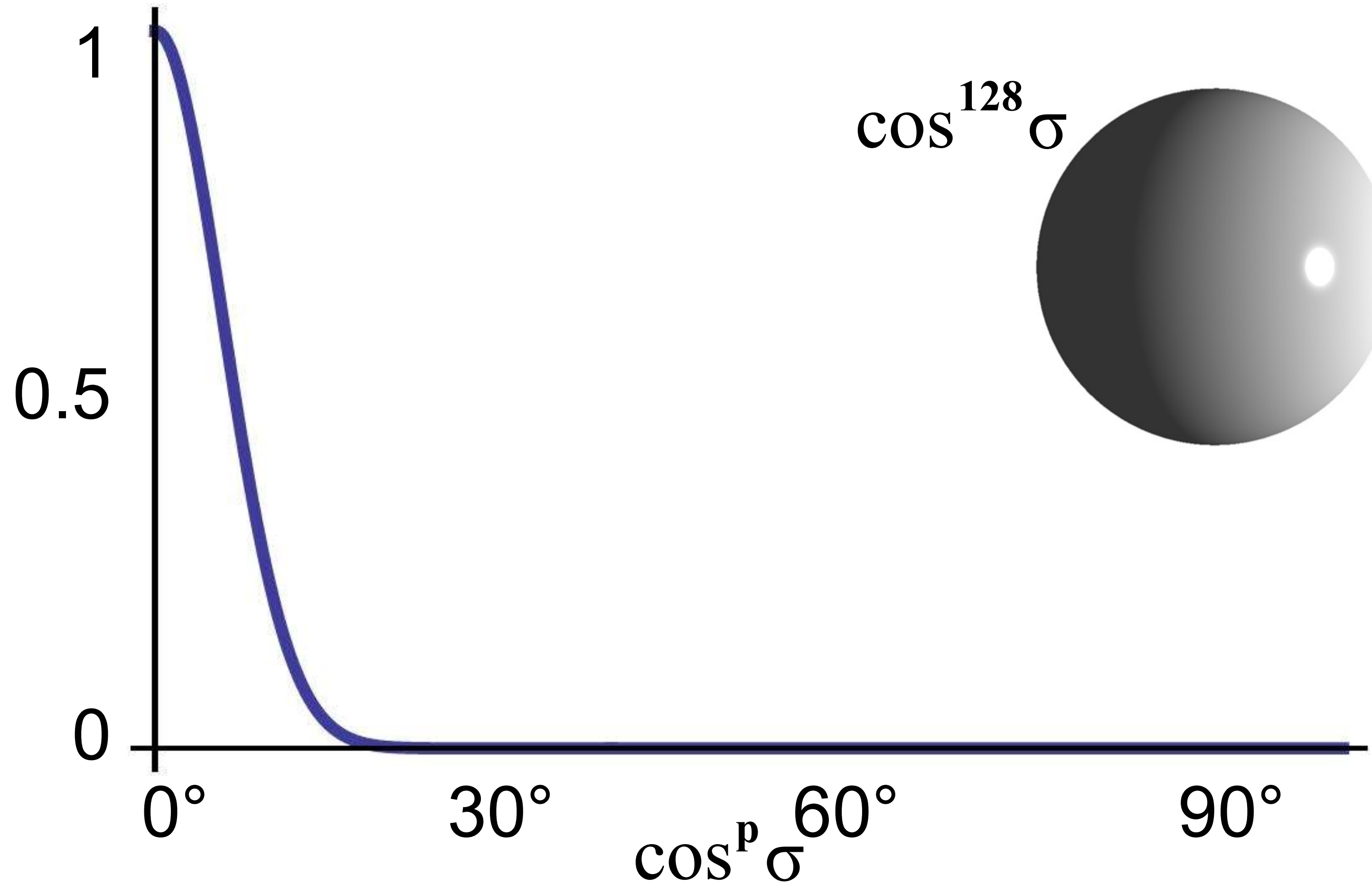
Specular Reflection Coefficient p



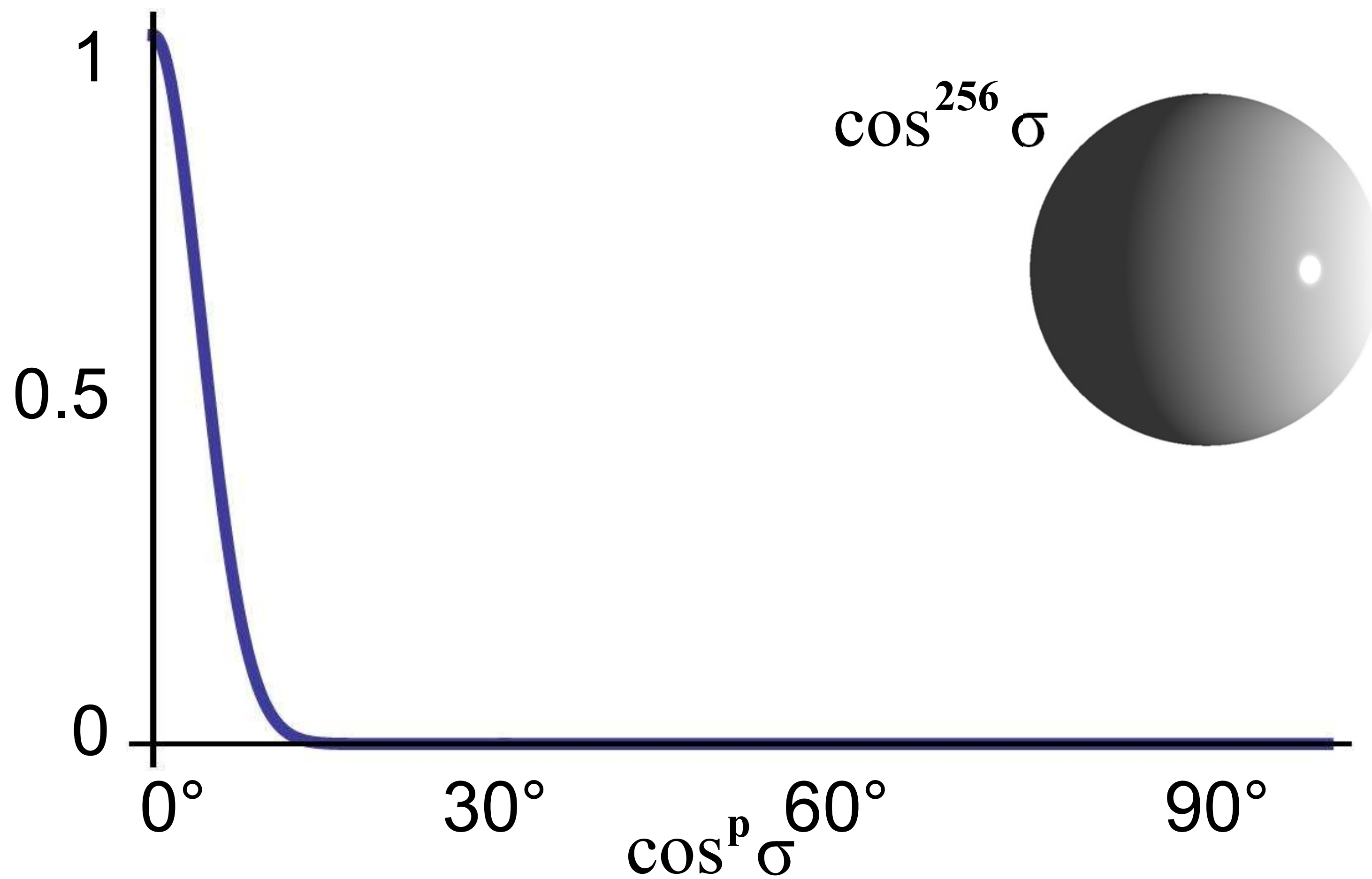
Specular Reflection Coefficient p



Specular Reflection Coefficient p



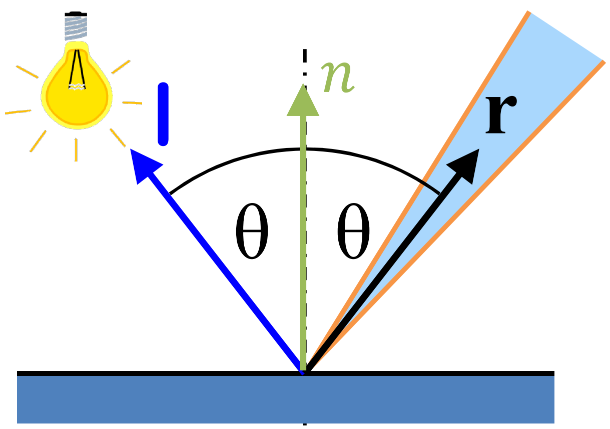
Specular Reflection Coefficient p



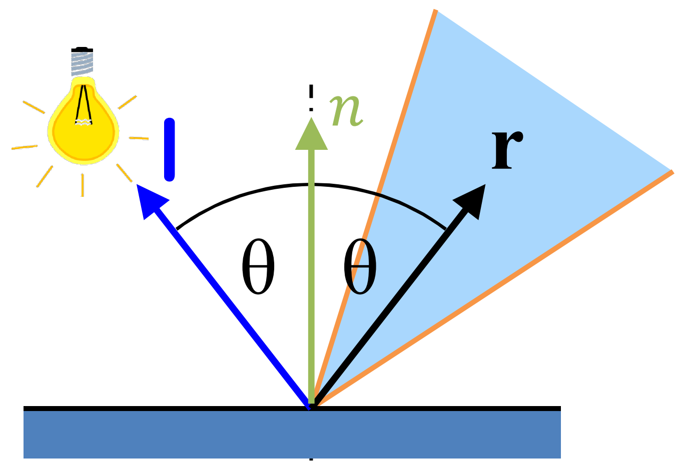
Specular Reflection Coefficient

- Empirical Phong model
  - p large $\Rightarrow$ shiny surface
  - p small $\Rightarrow$ dull surface

$$L_{spec} = M*S \cdot \cos^p\sigma$$

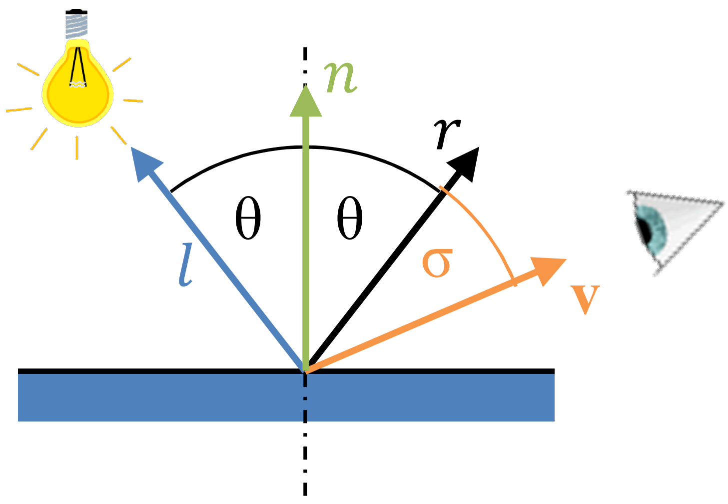


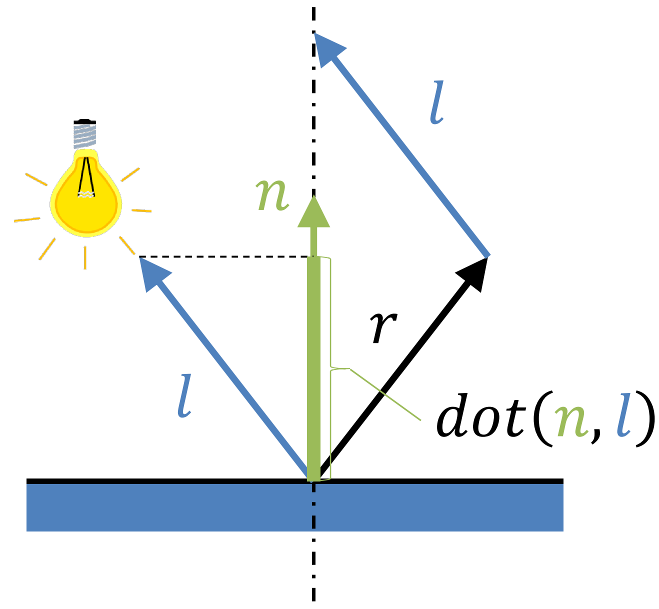
shiny surface
(large p)



dull surface
(small p)

Specular reflection





$$L_{spec} = M*S\cdot(\mathbf{v}\cdot\mathbf{r})^p$$

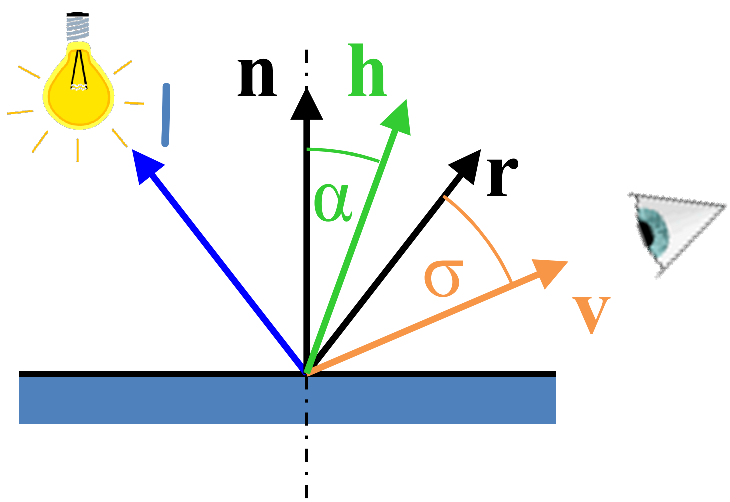
$$r + l = 2 \cdot dot(n, l) \cdot n$$

$$r = 2 \cdot dot(n, l) \cdot n - l$$

Blinn-Phong

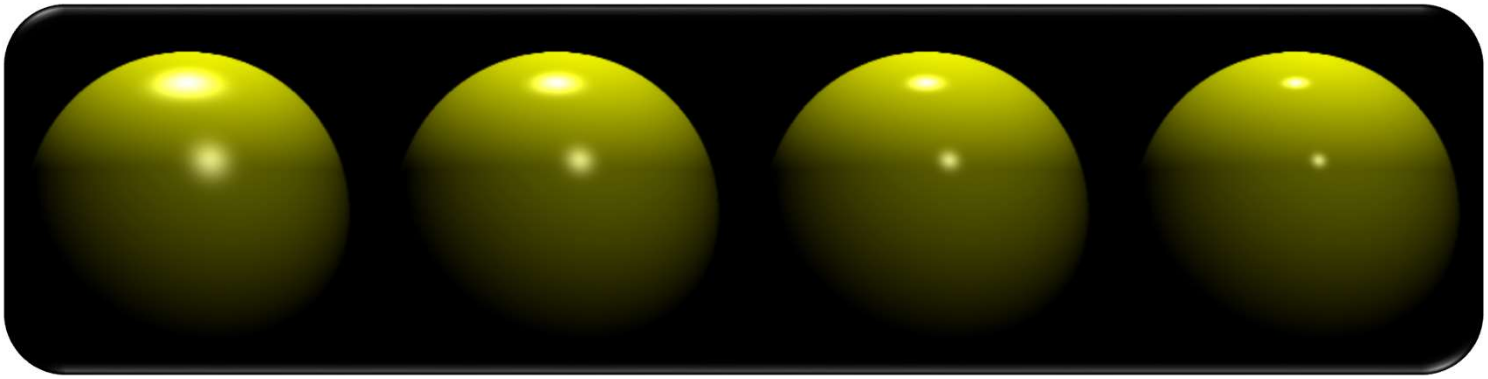
- Halfway vector $\mathbf{h}$

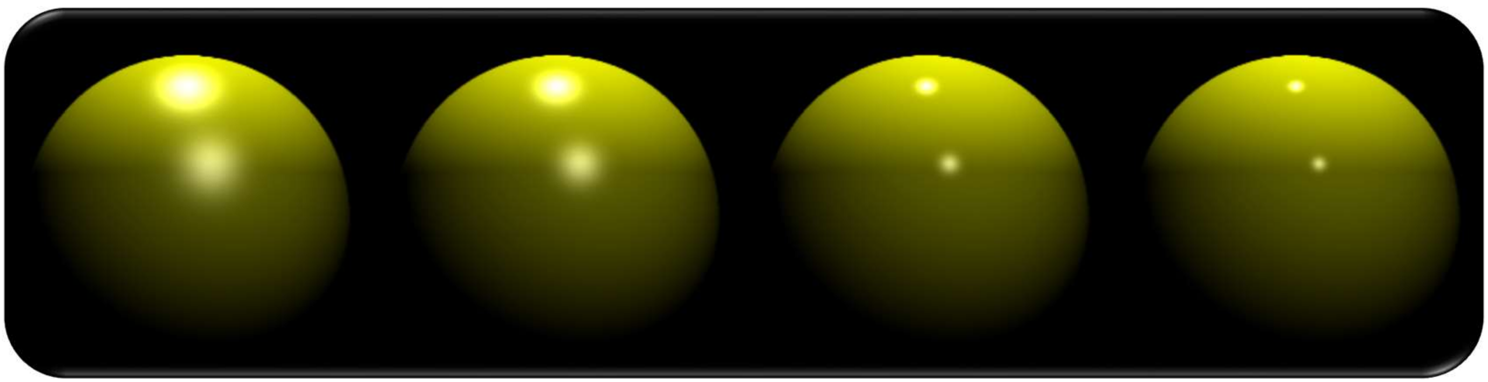
$$L_{spec} = M{*}S\cdot(\mathbf{v}\cdot\mathbf{r})^p \rightarrow \quad L_{spec} = M{*}S\cdot(\mathbf{n}\cdot\mathbf{h})^p$$



$$\mathbf{h} = \frac{\mathbf{l}+\mathbf{v}}{\|\mathbf{l}+\mathbf{v}\|}$$

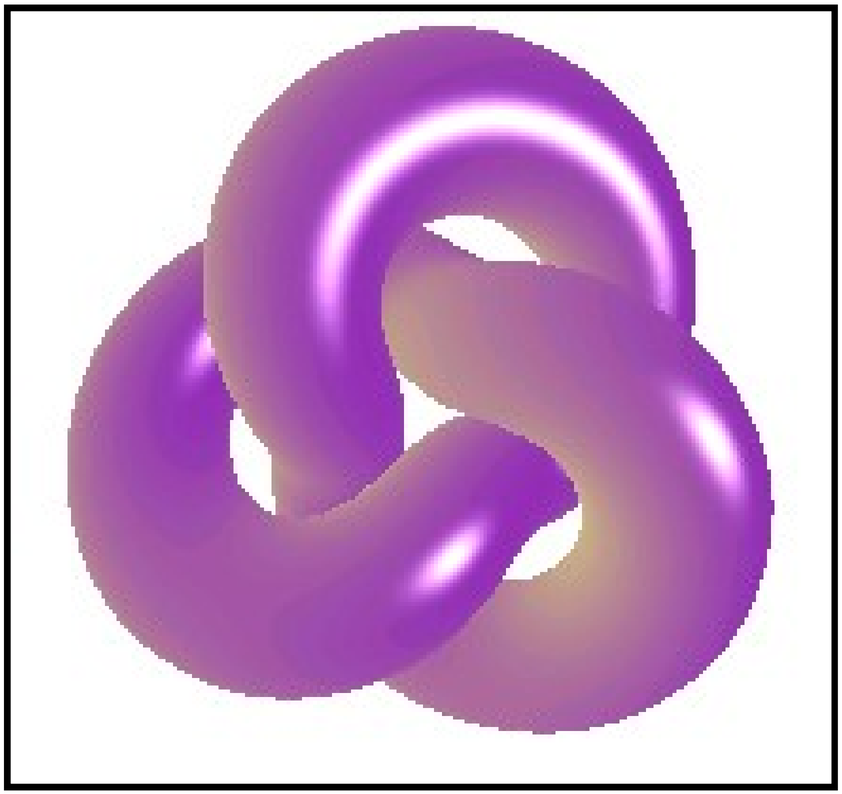
Phong vs Blinn-Phong





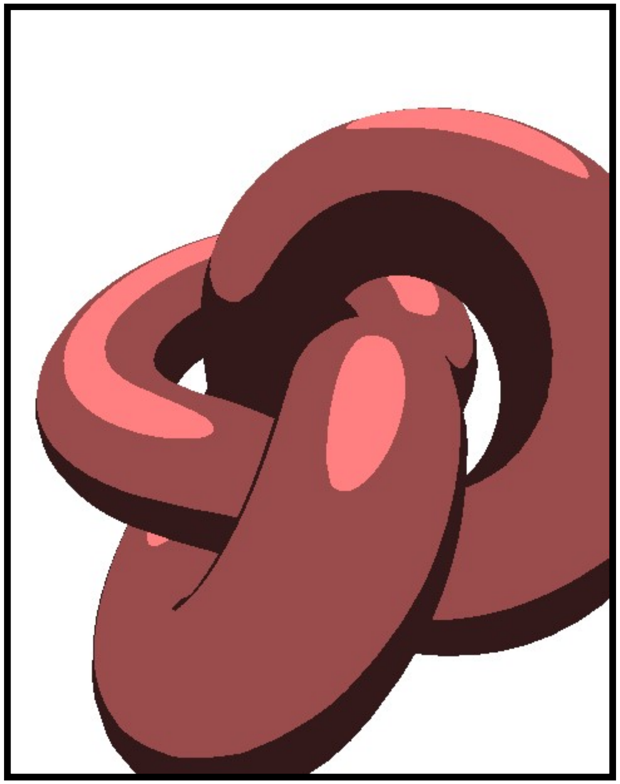
Gooch

- Blend between a cool and a warm color



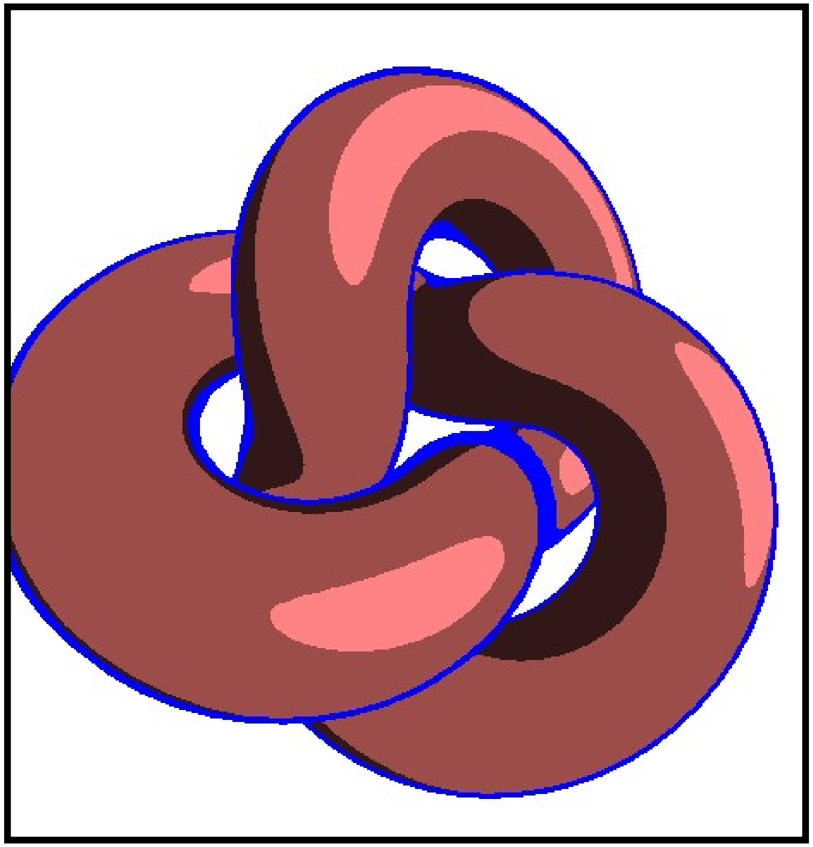
Toon shading

- Discrete color steps for diffuse



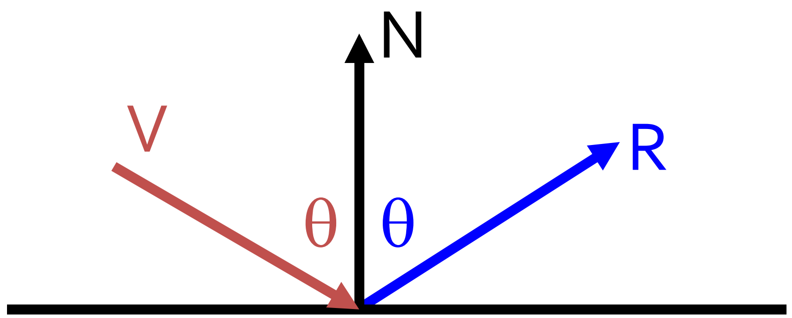
Cel shading

- Detect edges and color them



Reflective Environment Mapping

- Angle of incidence = angle of reflection



R = V - 2 (N dot V) N
= `reflect(V,N)`

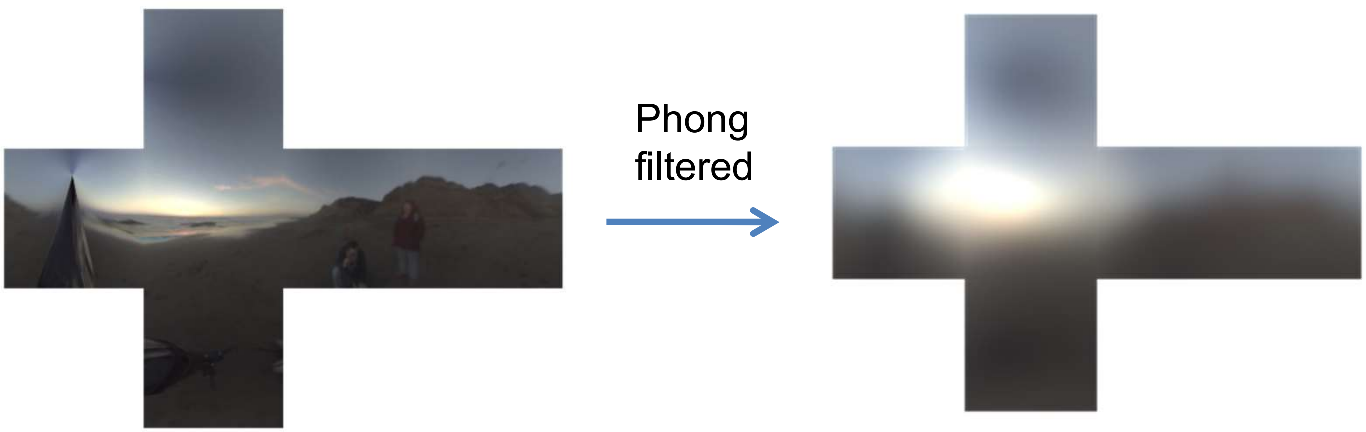
V and N normalized!

V is incident vector!

- Cube map needs reflection vector in coordinates (where map was created)

Specular Environment Mapping

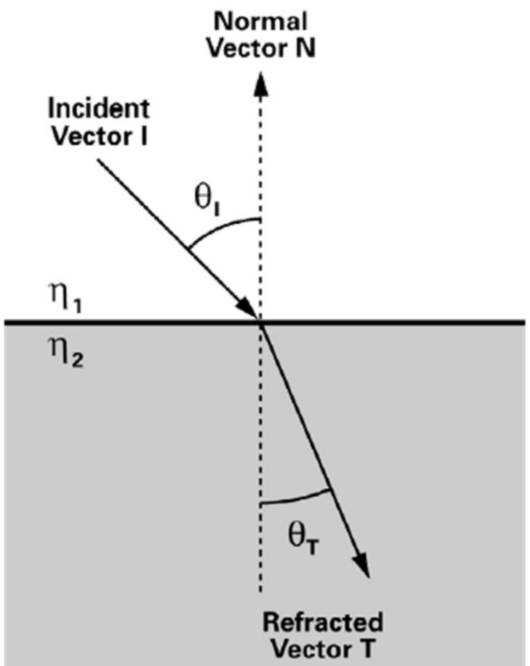
- We can pre-filter the environment map
  - Equals specular integration over the hemisphere
  - Phong lobe (cos^n) as filter kernel
  - **`textureLod;level`** according to glossiness
  - R as lookup



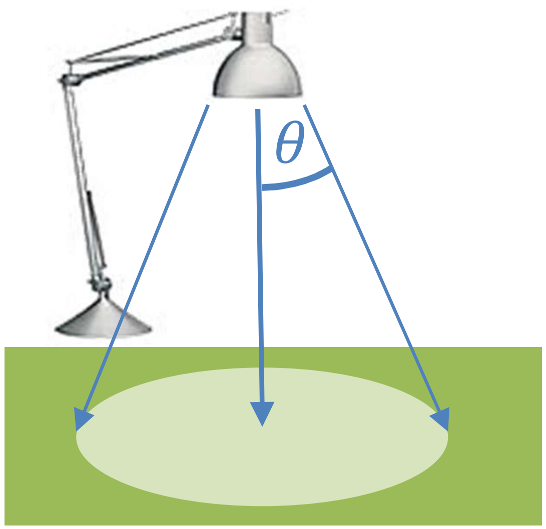
Refractive Environment Mapping

- Use refracted vector for lookup:
  - Snells law:

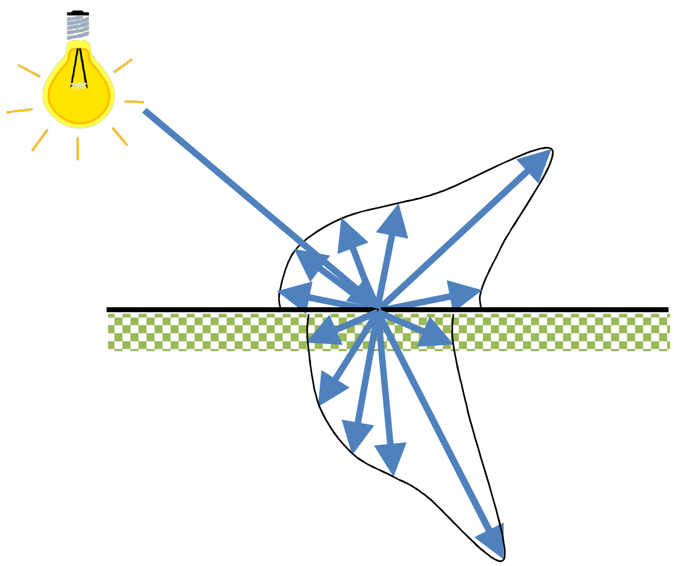
$$\eta_1 \sin\theta_I = \eta_2 \sin\theta_T$$



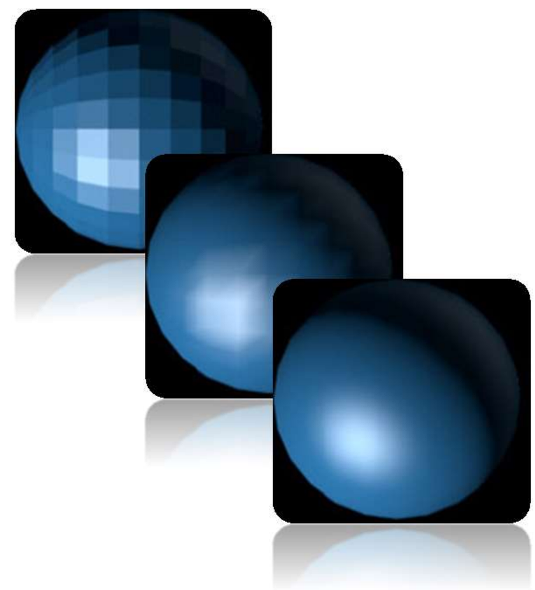
Light Sources



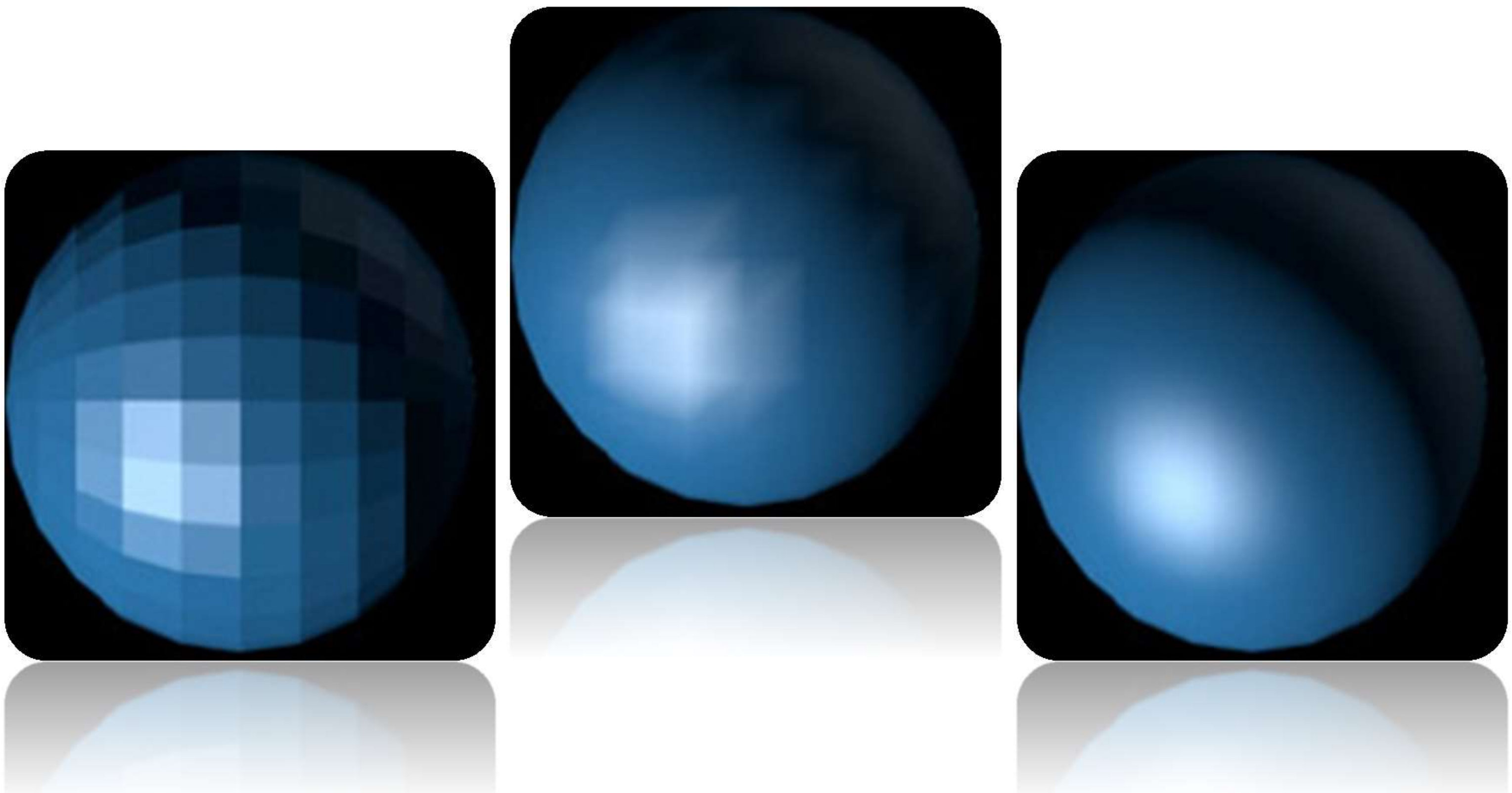
Reflection Models



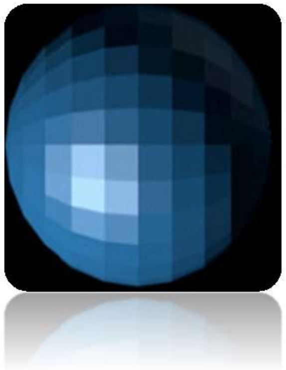
Shading Models



Shading Models

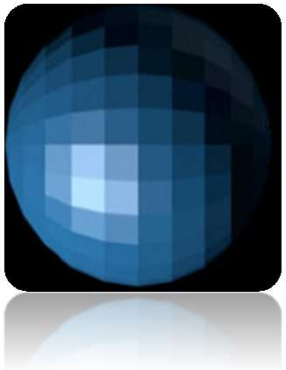


Shading model



- Shading ≠ shadows (shadowing)
- Coloring / shading the model
- When to evaluate lighting model

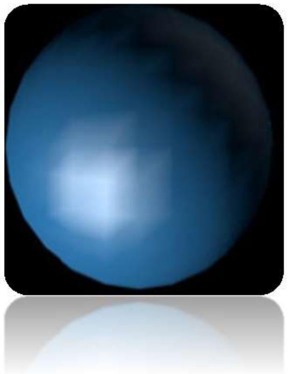
Flat-shading

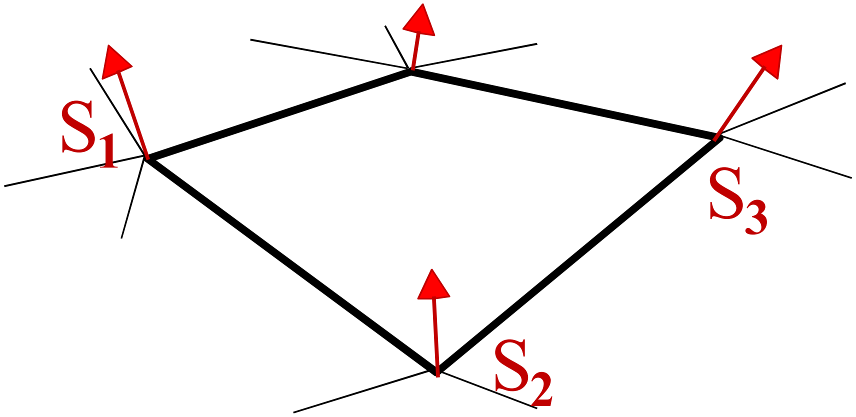


- 1 color for the mesh (polygon)
- Really fast
- Really ugly
- *If an object really <u>is</u> faceted, is this accurate?*
- No:
  - Point light sources
    - Direction to light varies across the facet
  - Specular reflectance
    - Direction to eye varies across the facet

Gouraud shading

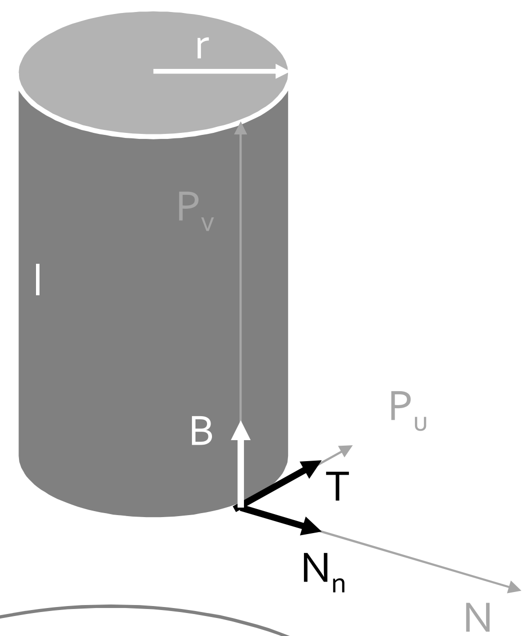
1. Calculate the normal vector for each vertex
2. Calculate the intensitity for each vertex

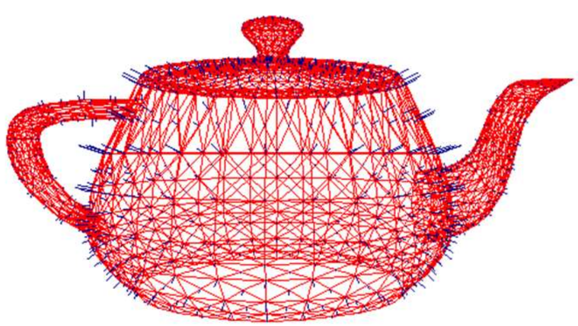


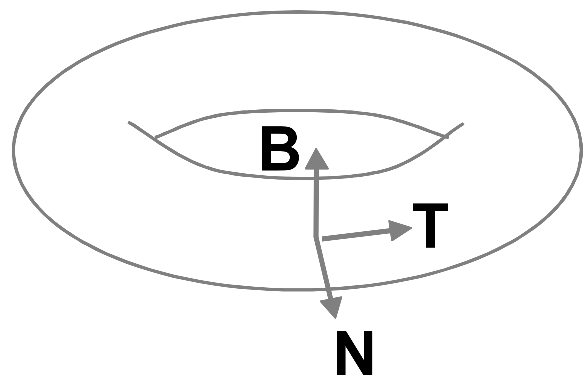


Vertex Normals

- Vertex normals may be
  - Provided with the model
    - Artist
    - 3d program
  - Computed from first principles
    - Mathematic description of model

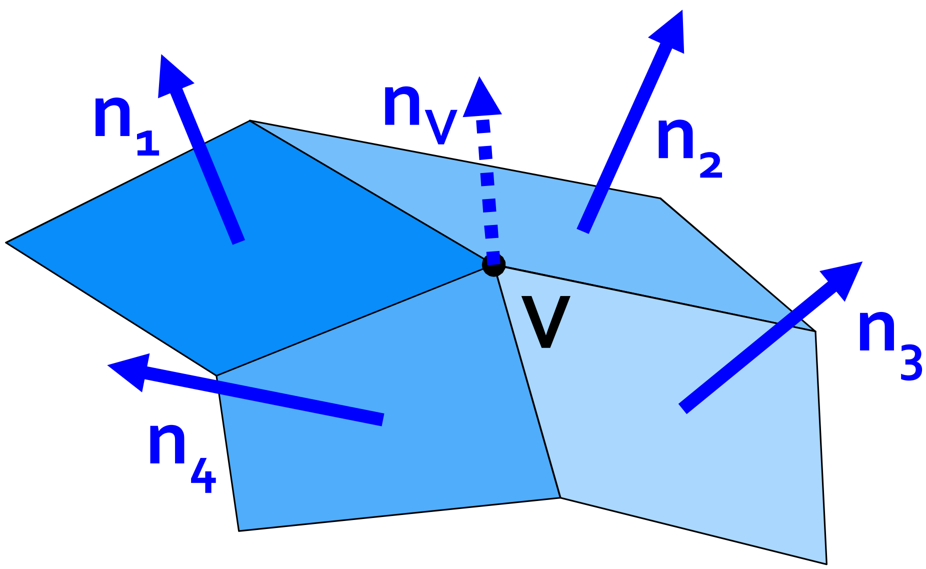






Vertex Normals

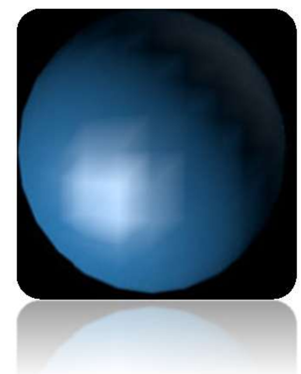
- Approximated by averaging the normals of the facets that share the vertex

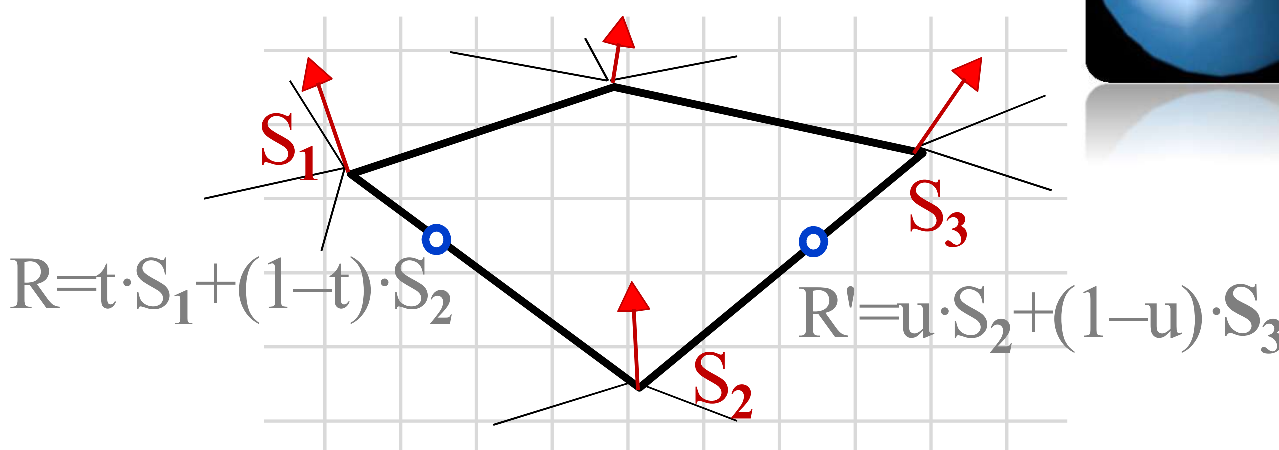


$$\mathbf{n}_V = \frac{\sum_{k=1}^{N} \mathbf{n}_k}{\left\| \sum_{k=1}^{N} \mathbf{n}_k \right\|}$$

Gouraud shading

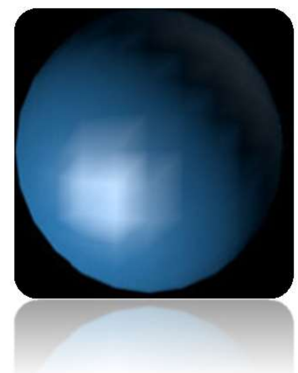
1. Calculate the normal vector for each vertex
2. Calculate the intensitity for each vertex
3. Color interpolation along edges

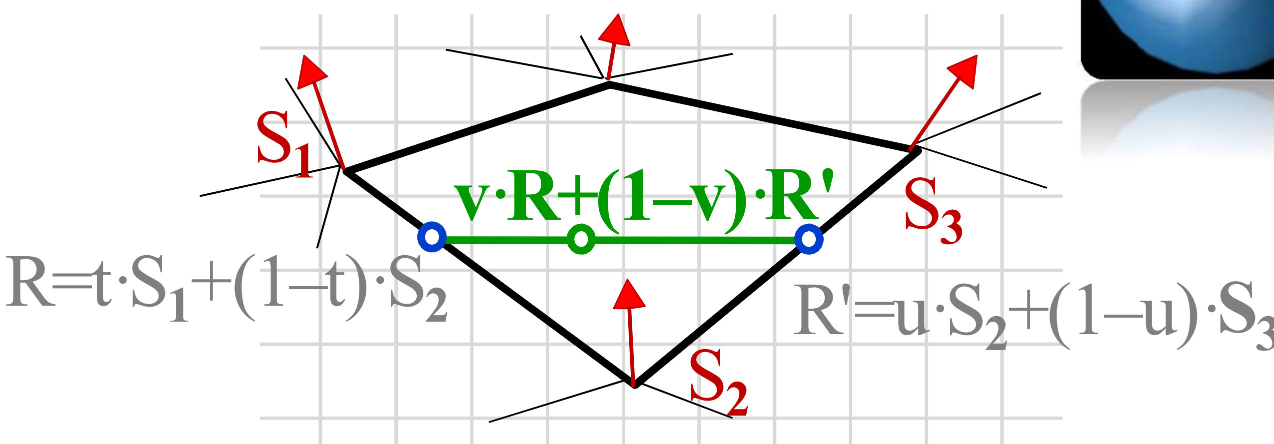




Gouraud shading

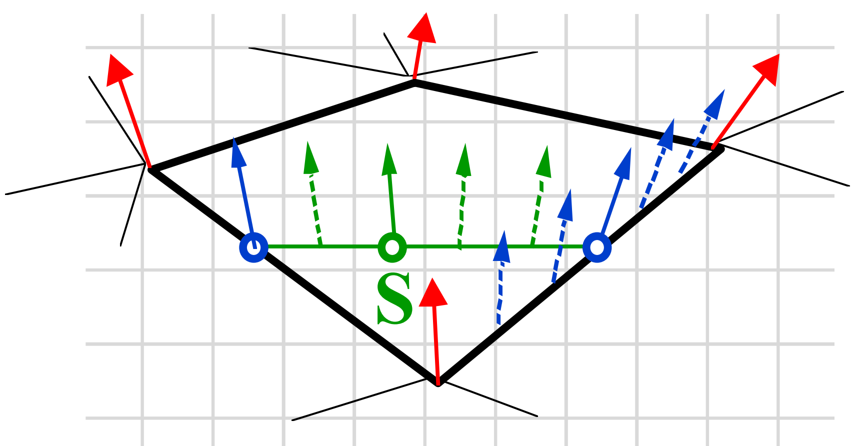
1. Calculate the normal vector for each vertex
2. Calculate the intensitity for each vertex
3. Color interpolation along edges
4. Color interpolation along scanline

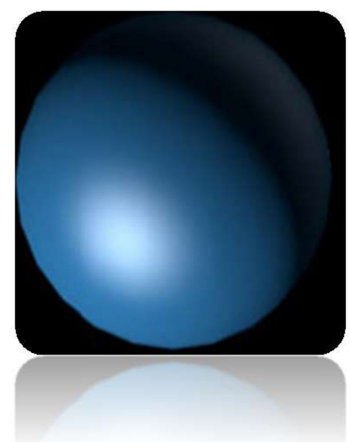




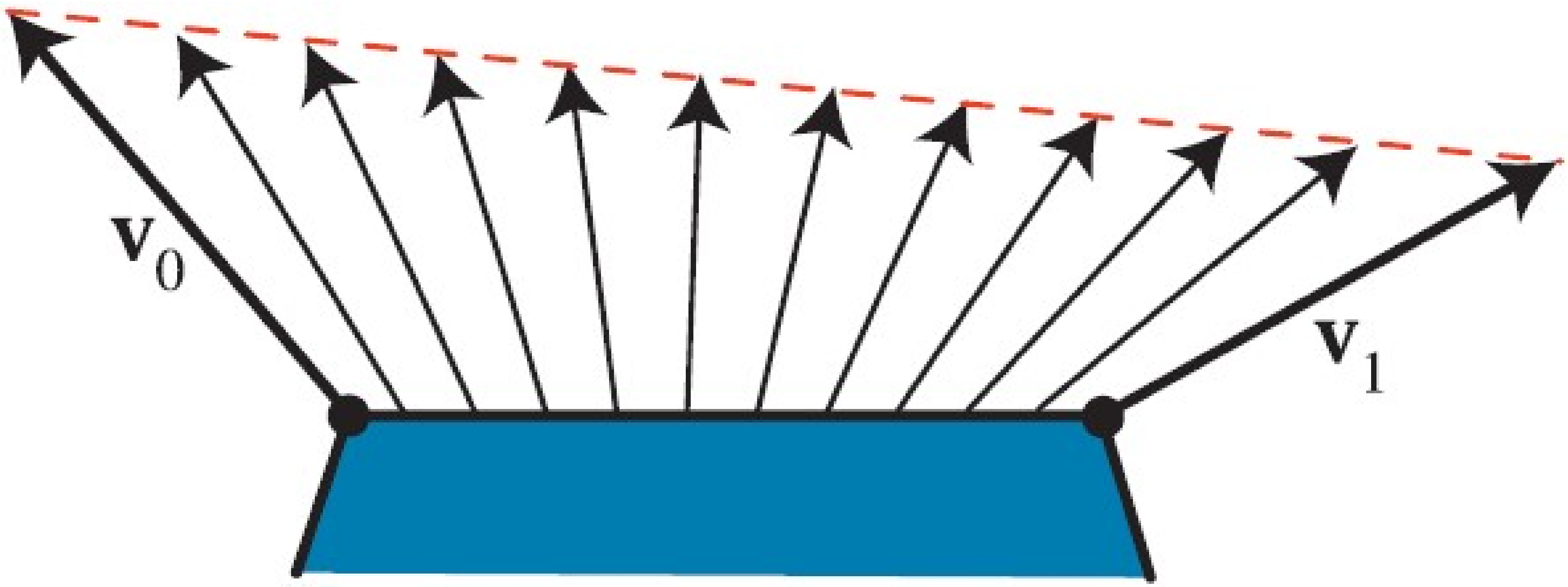
Phong Shading Model

1. Normal Vector for each vertex
2. Normal vectors are interpolated along the edge
3. Normal vectors are interpolated along the scanline
4. Calulate the intensity using the normal vectors

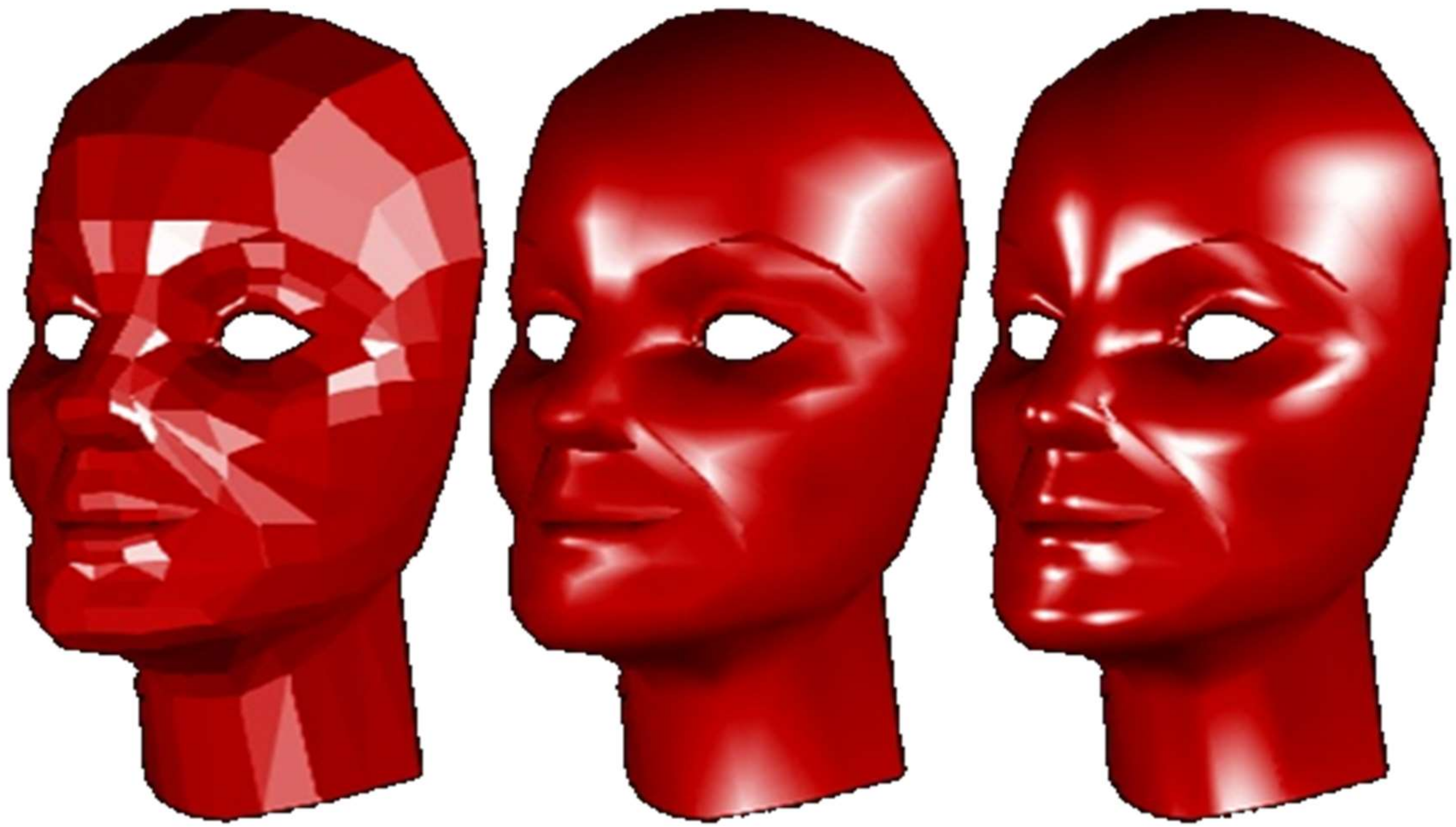




Normal Interpolation



Flat / Gouraud / Phong Comparison



Transforming Normals

- Differential scaling changes shape and normals
- If **M** transforms points,
  then $(\mathbf{M}^T)^{-1}$ transforms normals