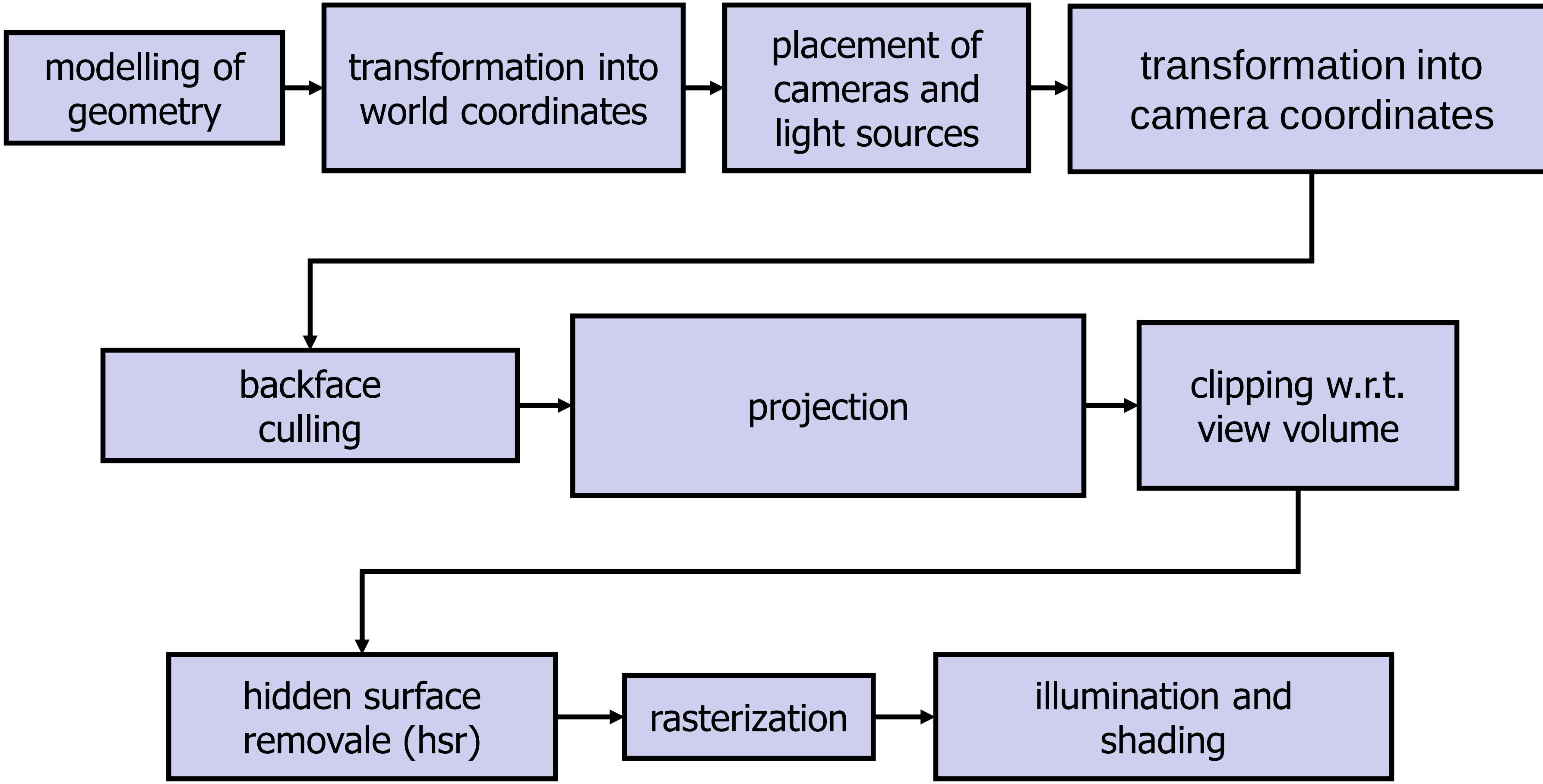
Computer Graphics

Illumination and Shading

Rendering Pipeline



Overview

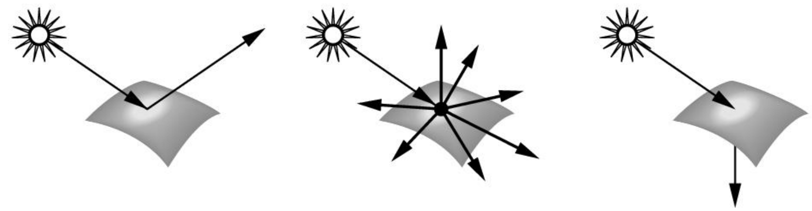
- illumination models
  - light sources
  - light interaction with surfaces
  - Phong illumination model
- shading
  - application of illumination models to rendering polygons and pixel-by-pixel
  - efficiency vs. quality
  - flat, Gouraud, Phong shading

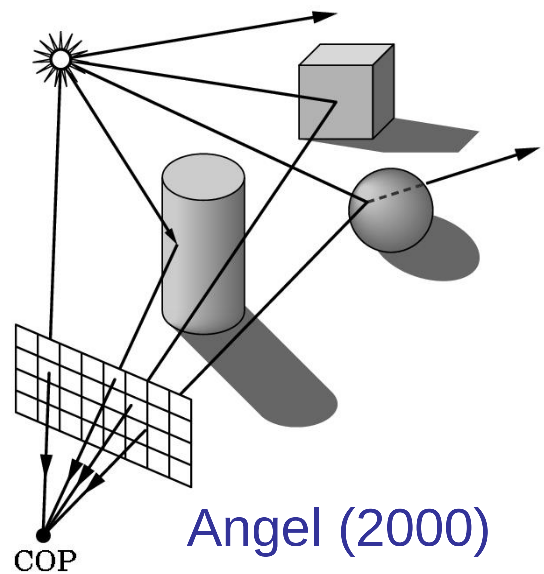
Illumination Models

General Introduction
Phong Illumination Model

Illumination Models

- real world:
  - surfaces emit, absorb, reflect, and scatter light
  - light intensity and color dependent on surface position and orientation w.r.t. the light source
  - light is usually reflected or refracted several times
  - usually several sources of light
  - final intensity/color at a point is sum of several light paths ending at that point





Angel (2000)

Illumination Models

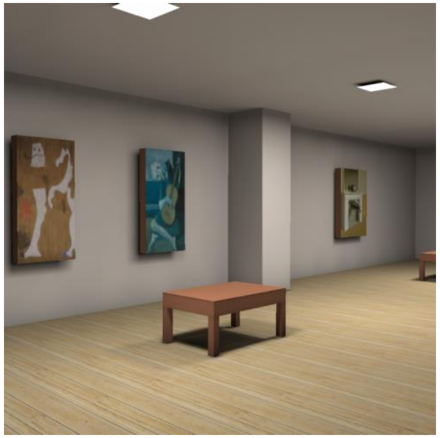
- computer graphics world:
  - mathematical description necessary
  - leads to equation using integrals: the *rendering equation*

$$L_o(x,\vec{\omega}') = L_e(x,\vec{\omega}') + \int_\Omega f_r(x,\vec{\omega},\vec{\omega}')\, L_i(x,\vec{\omega})(\vec{\omega}\cdot\vec{n})\, d\vec{\omega}$$

  - it's usually not solvable
  - we need approximation to speed-up things!
    - simplifying light sources
    - simplifying materials
    - simplifying computation

Illumination Models

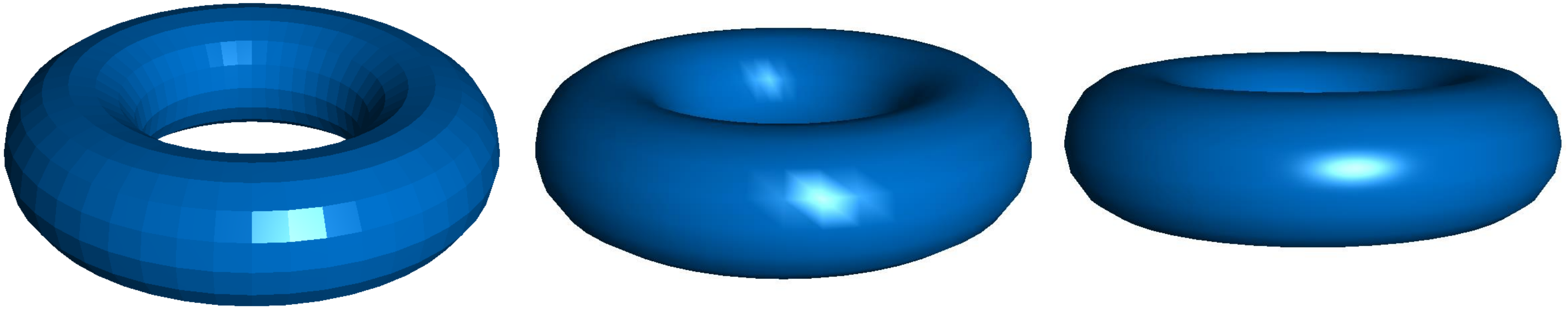
- description of the factors that influence the color and light intensity at a point
- **global illumination models**:
  - considering all objects of a scene to compute light at a specific point
  - e.g., radiosity and raytracing



Coombe et al., 2004

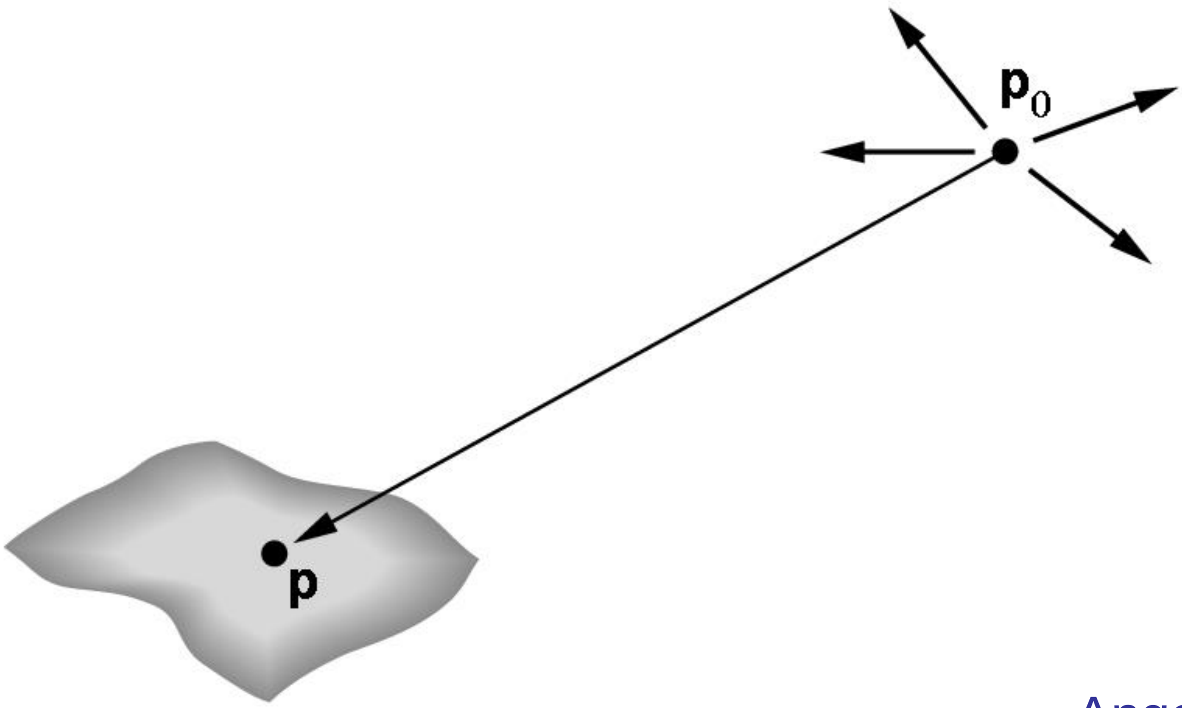
Illumination Models

- **local illumination models**:
  - i.e., traditional **"rendering"**
  - only object-light interaction, pipeline approach
  - heuristic approximation, simpler than global
  - Phong illumination model & polygonal shading



Light Sources in Computer Graphics

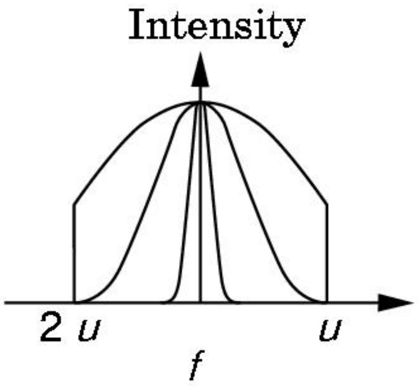
- **point light source**: has only position and no size, sends light equally in all directions $\rightarrow$ point in 3D & intensity

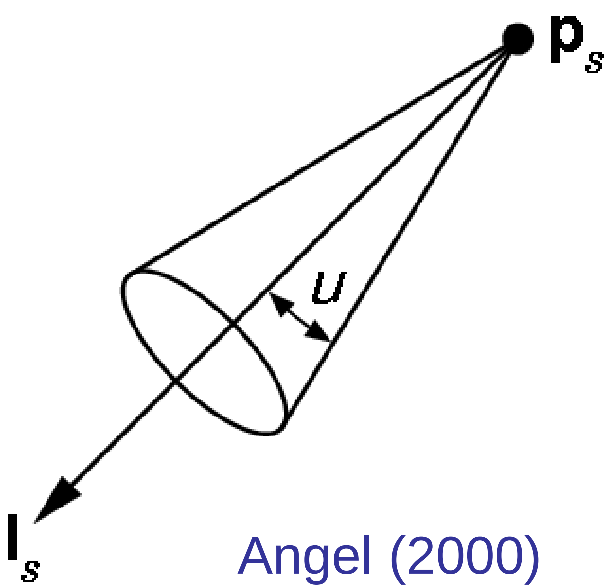


Angel (2000)

Light Sources in Computer Graphics

- **spot light source**: point light source sending out a cone of light with light intensity decreasing towards cone border → point + vector in 3D, angle, attenuation
- intensity defined by cosine function depending on angle from center ray, exponent

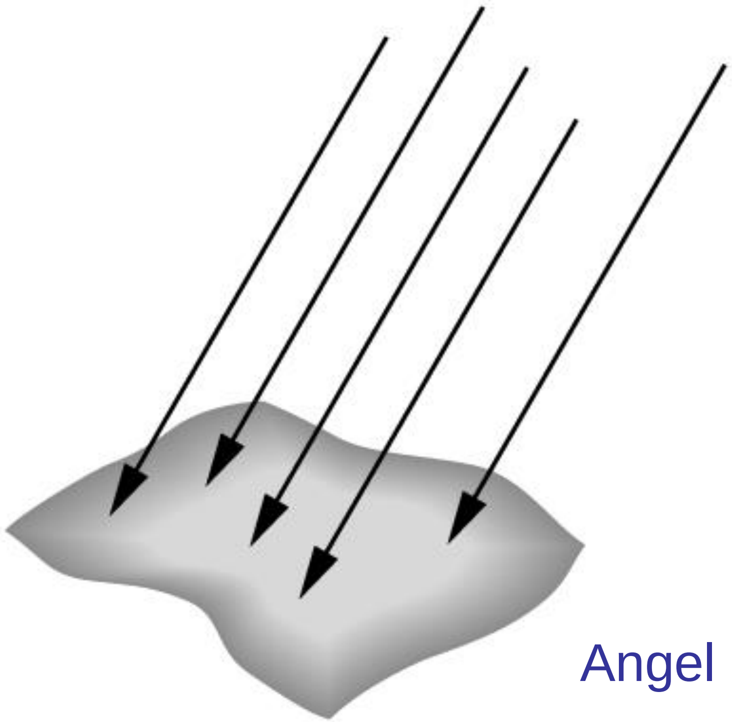




Angel (2000)

Light Sources in Computer Graphics

- **directional light source**: sends directed, parallel light rays, has no position
  $\rightarrow$ vector in 3D & intensity

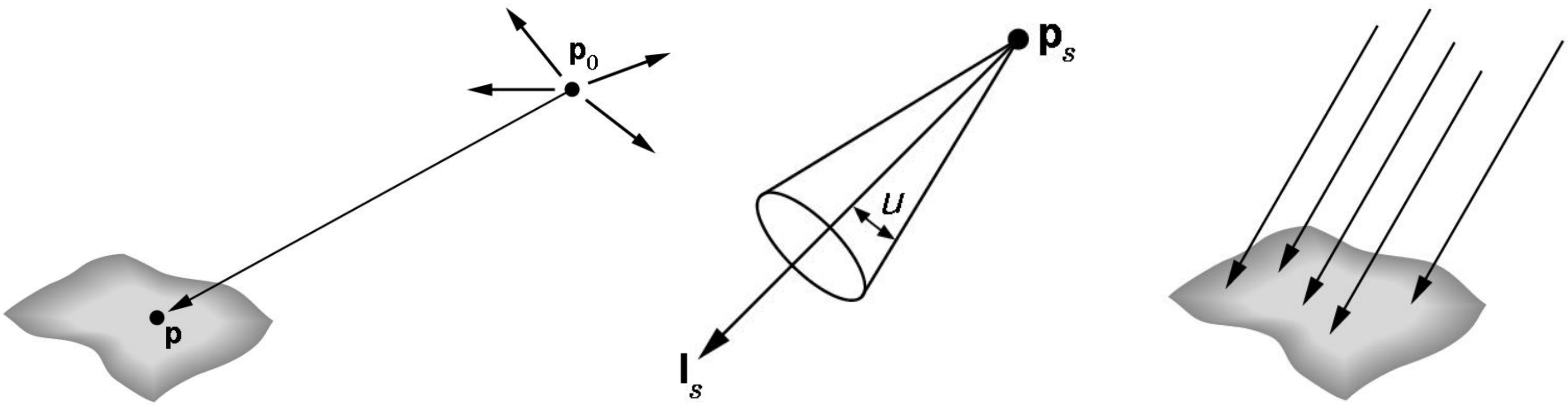


Angel (2000)

Light Sources in Computer Graphics

- results from simplification to point and simple directional light sources?

Angel (2000)



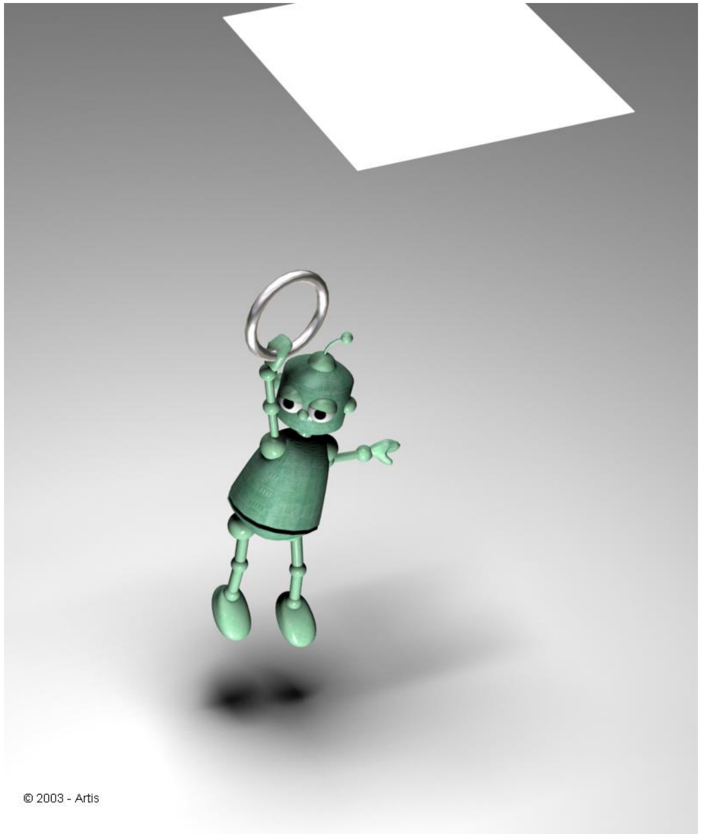
point light spot light directional light

Light Sources in Computer Graphics

- results from simplification to point and simple directional light sources?
  - sharp contrast between light and shade, no gradual change (penumbra)
  - aerial light sources have to be approximated by several point lights to have a penumbra

Light Sources in Computer Graphics

- results from simplification to point and simple directional light sources?
  - sharp contrast between light and shade, no gradual change (penumbra)
  - aerial light sources have to be approximated by several point lights to have a penumbra

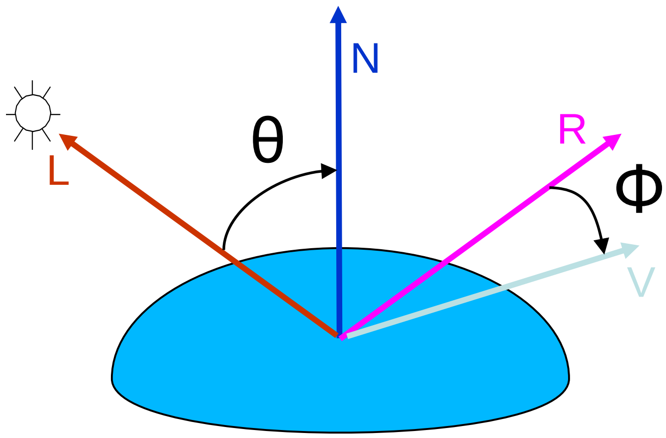


Light Sources in Computer Graphics

- light attenuation
  - light intensity reduces with growing distance
  - theoretically: $I \sim 1/d^2$
  - reality does not follow exactly this – why?
  - CG: $I \sim 1/(a+bd+cd^2)$, often only linear: $a,c = 0$
- light color
  - heuristic: color modeled using RGB values
  - approximation because light behavior depends on wave length – examples?

Light Interaction with Surfaces

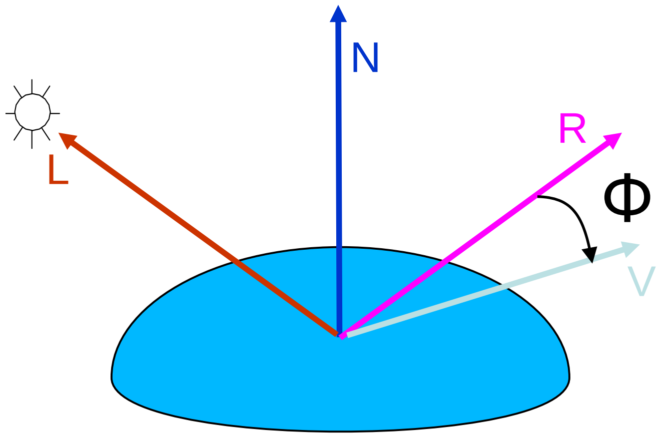
- angle θ between L & N determines diffuse reflection
- reflection angle equals θ
- angle Φ between R & V determines perceived brightness
- maximal reflection if R = V (Φ = 0)



L – vector to light source
N – surface normal vector
R – reflected light ray
V – vector to viewer/observer

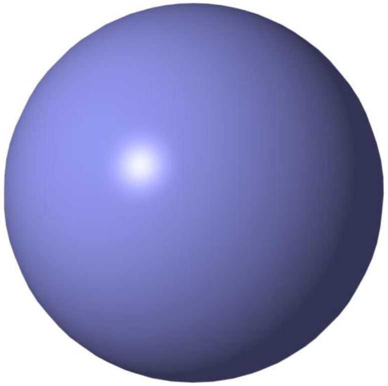
Light Interaction with Surfaces

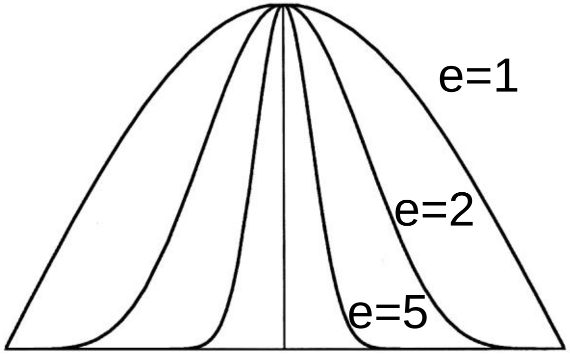
- **directed reflection**: reflection only for small Φ: smooth surfaces – examples?
- physical reality
  - – non-symmetric reflection around R
  - – materials with anisotropic reflection (reflection depends on angle Φ AND direction of L w.r.t. surface)

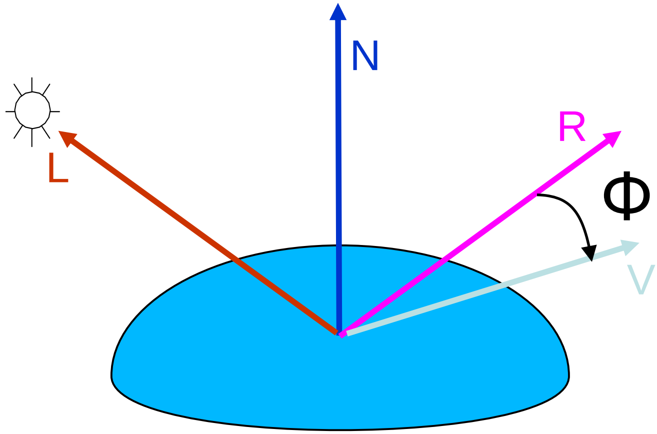


Light Interaction with Surfaces

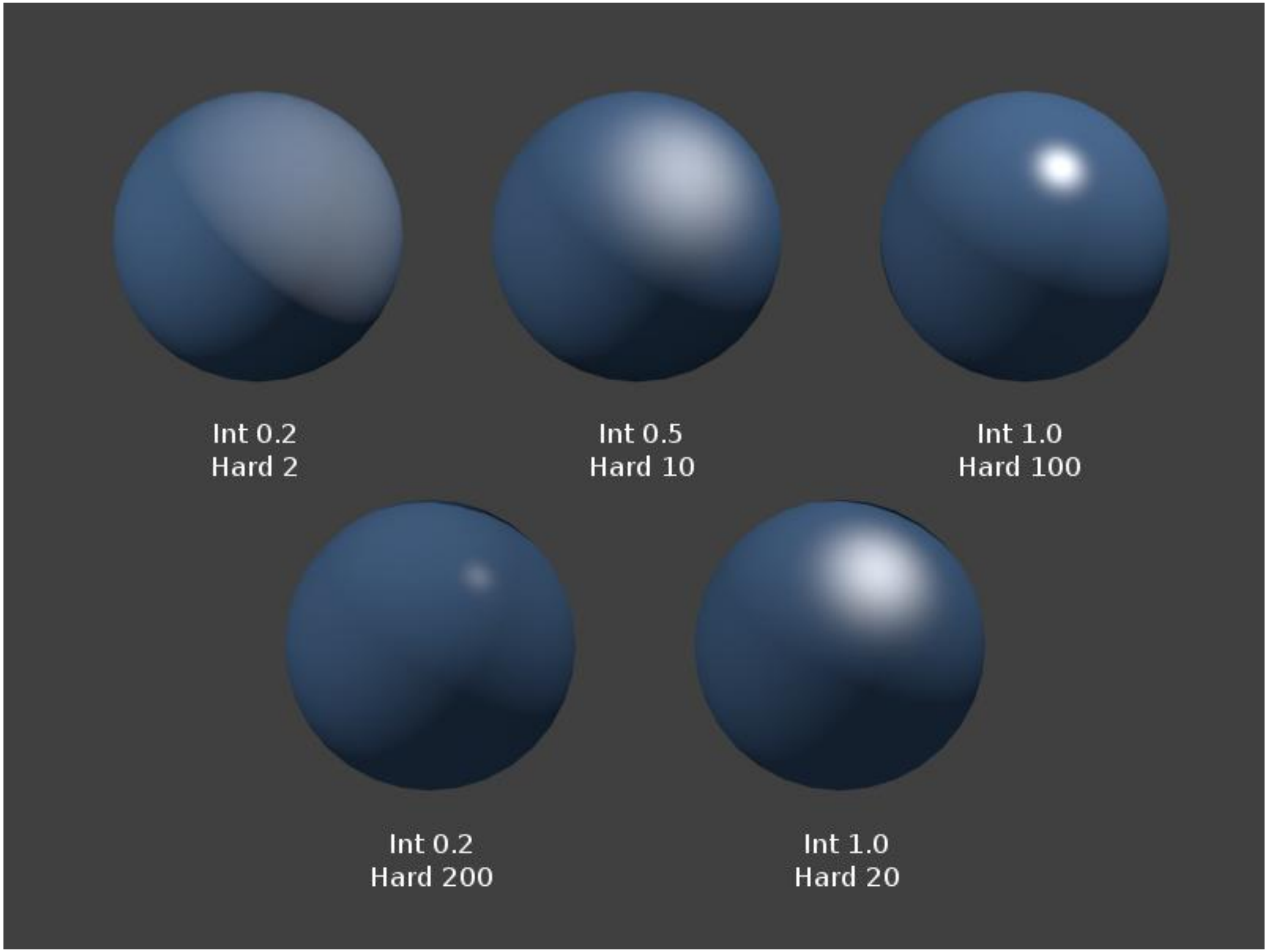
- light attenuation depending on angle between R and V: shininess
- modeled using cosine function and exponent: $I \sim \cos(\Phi)^e$
- metals: $e \approx 100$





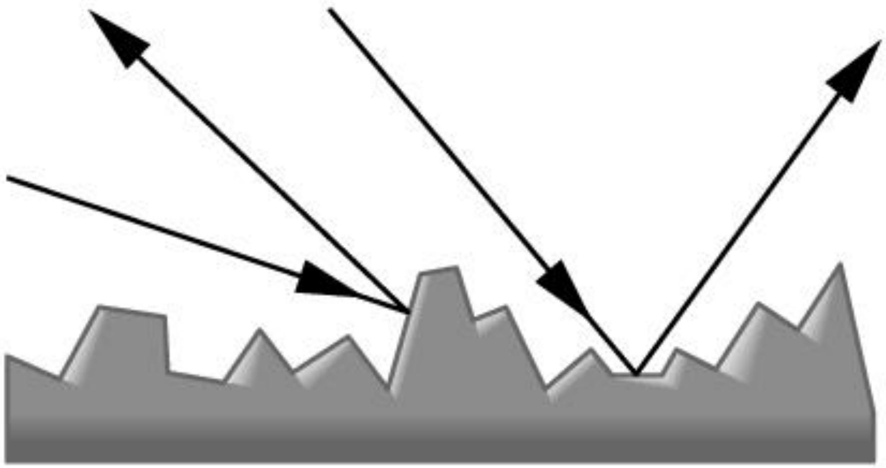


Example of Different Highlights



Light Interaction with Surfaces

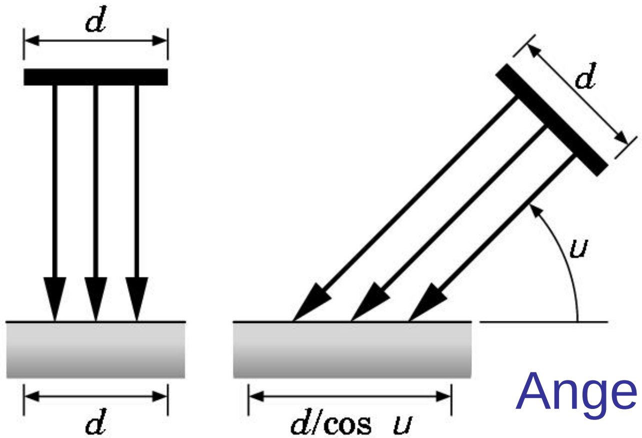
- **diffuse reflection**: equal reflection in all directions on rough surfaces, depends only on θ and not observer – examples?
- due to light scattering on rough surfaces on randomly oriented microscopic facets



Angel (2000)

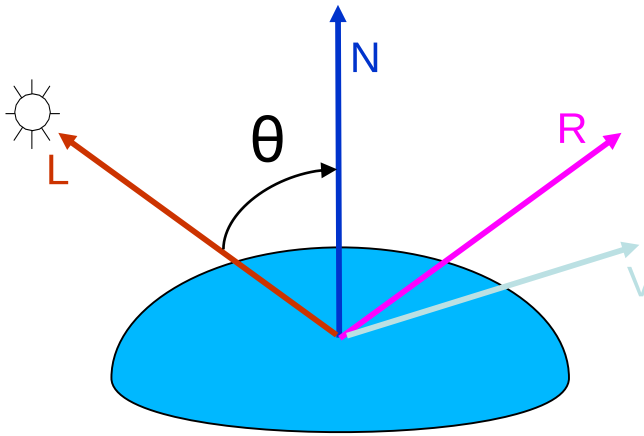
Light Interaction with Surfaces

- diffuse reflection modeled according to Lambert's law by cosine function: $I \sim \cos\theta = L \bullet N$ for normalized L, N



Angel (2000)

because light distributes over larger area

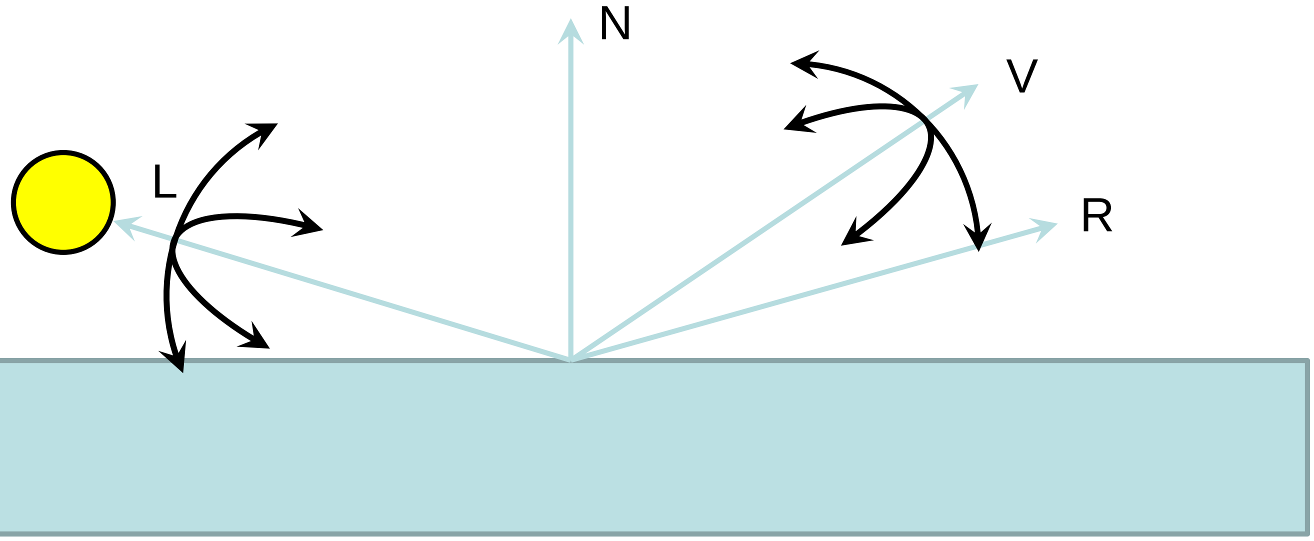


$\theta = 0^o \rightarrow$ max. intensity

$\theta = 90^o \rightarrow 0$ intensity

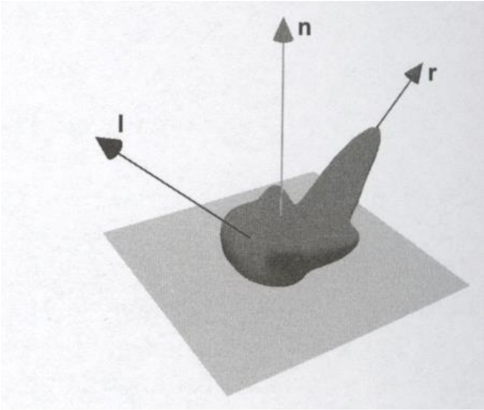
Light Interaction with Surfaces

- combined effects of diffuse and directed reflection make up material properties
- in reality very complex function, can be captured with BRDFs (Bidirectional Reflectance Distribution Function)



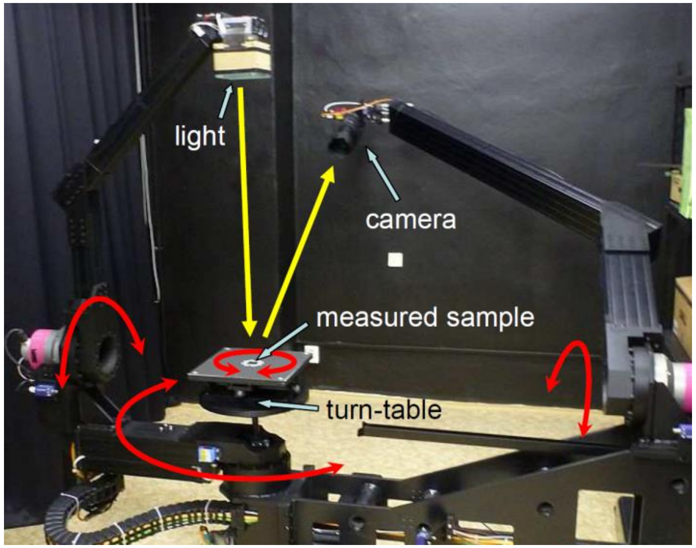
Light Interaction with Surfaces

- BRDF visualized for one incoming light direction:



Bender/Brill (2003)

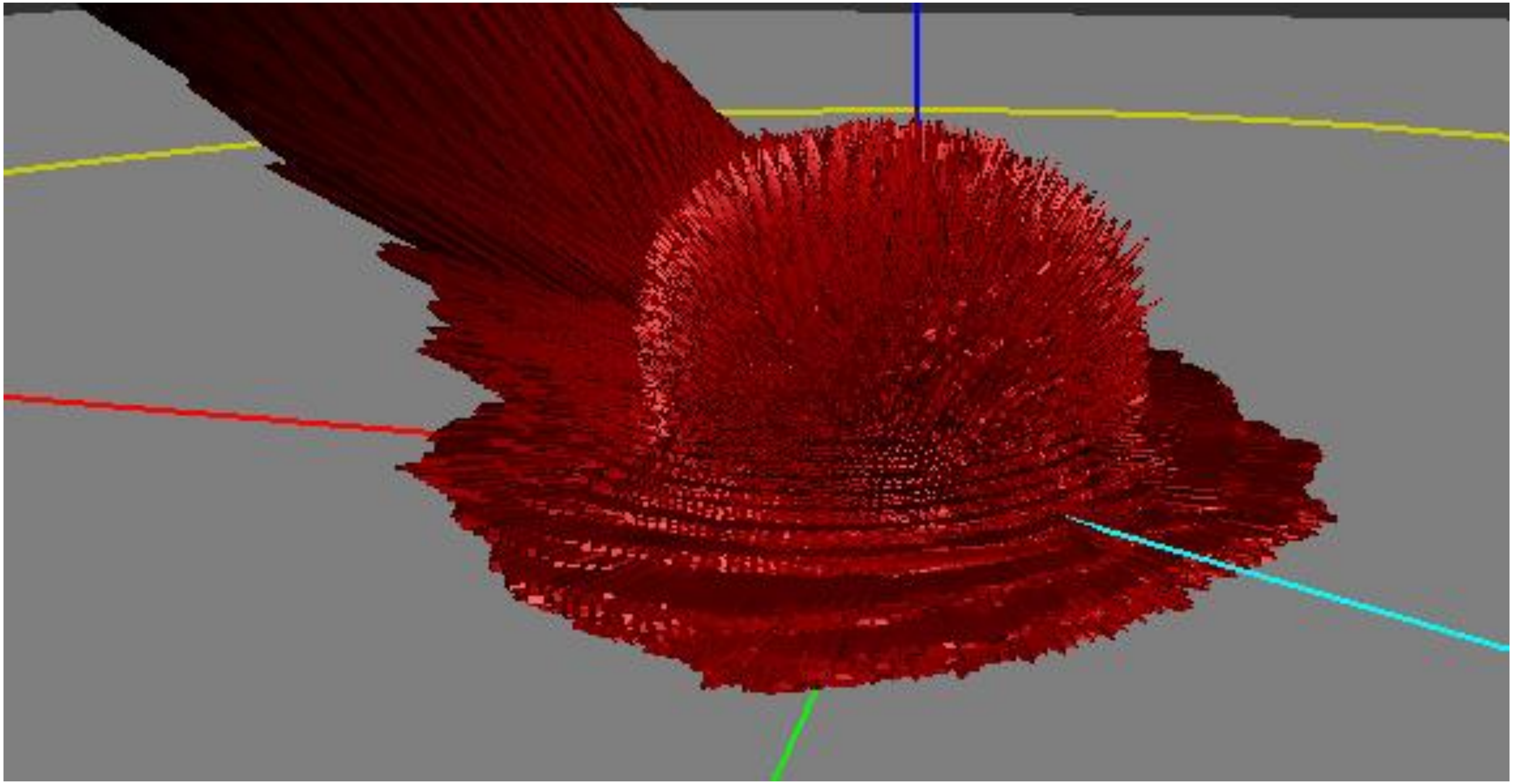
- BRDF can be physically measured and used as look-up table for realistic illumination:



Filip et al. (2013)

Light Interaction with Surfaces

- measured BRDF example: nylon

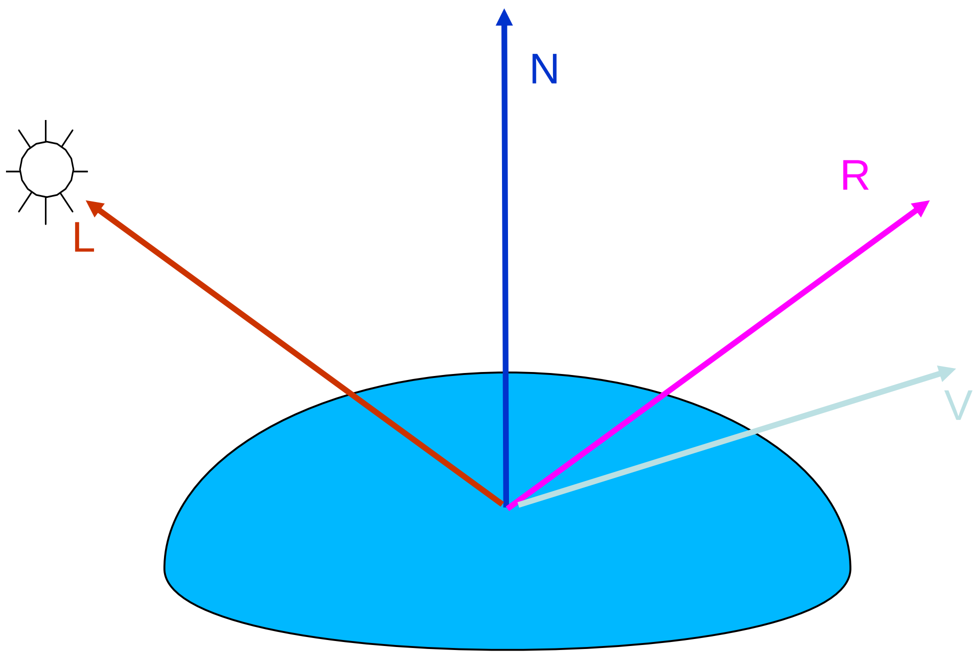


Phong Illumination Model (1973)

- most common CG model for illumination (by Bùi Tường Phong): $I_{Phong} = I_a + I_d + I_s$
- ambient light: base illumination of scene
  - simulates light scattering on objects
  - necessary because repeated diffuse reflection is not considered in local illumination model
  - depends on color of all objects in scene
  - should always be kept very small
- diffuse light: light from diffuse reflection
- specular light: light from directed reflection

Phong Illumination Model

$$I_{Phong} = I\,k_a + \frac{1}{a + bd + cd^2}\left(I\;k_d (L \cdot N) + I\;k_s (R \cdot V)^e \right)$$



L – vector to light source
N – surface normal vector
R – reflected light ray
V – vector to viewer/observer

has to be evaluated for all light sources and for each of the base colors

Phong Illumination Model

$$I_{Phong} = I\,k_a + \frac{1}{a + bd + cd^2}\left(I\,k_d(L \cdot N) + I\,k_s(R \cdot V)^e\right)$$

- light and k-parameter factors
  - k values defined per color channel
  - light intensity I is computed once per color channel: red, green, blue
  - we thus get I_{red}, I_{green}, I_{blue}
  - these RGB values are used to set the color of the computed pixel

Phong Illumination Model

$$I_{Phong} = I\,k_a + \frac{1}{a + bd + cd^2}\left(\underline{I}\;k_d(L \cdot N) + \underline{I}\;k_s(R \cdot V)^e\right)$$

- parameters example:

light color/intensity (r, g, b) → I

```
Light      [pos x-z]    .8 .8 .4
Material   1.0 .8 .0    .2 .8 .3   1
```

material color (r, g, b): basis for deriving k_a & k_d

factors to compute actual k_a, k_d, & k_s

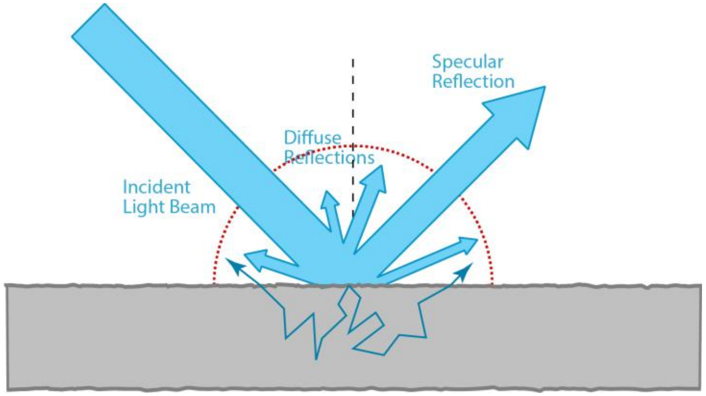
e

- $k_a^{red} = 0.2 \cdot 1.0$; $k_a^{green} = 0.2 \cdot 0.8$; $k_a^{blue} = 0.2 \cdot 0.0$
- $k_d^{red} = 0.8 \cdot 1.0$; $k_d^{green} = 0.8 \cdot 0.8$; $k_d^{blue} = 0.8 \cdot 0.0$
- $k_s^{red} = k_s^{green} = k_s^{blue} = 0.3$ (unaffected by mat. color)
- other ways possible

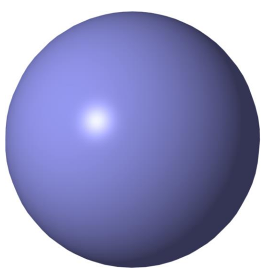
Why specular reflection unaffected?

specular: reflection only on surface

diffuse: also sub-surface scattering

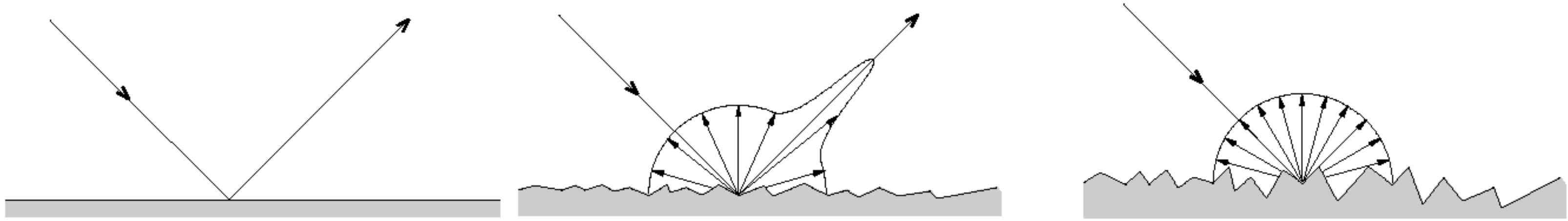


Phong Illumination Model: Materials

	k_a	k_d	k_s	e
 	0.1	1.0	0.0	1
	0.1	0.0	1.0	500
	0.1	0.5	0.3	10

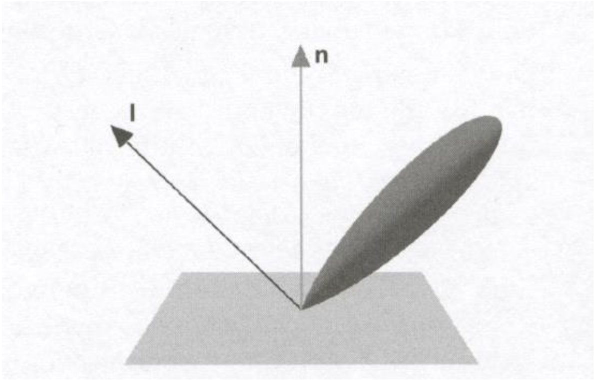
Phong Illumination Model

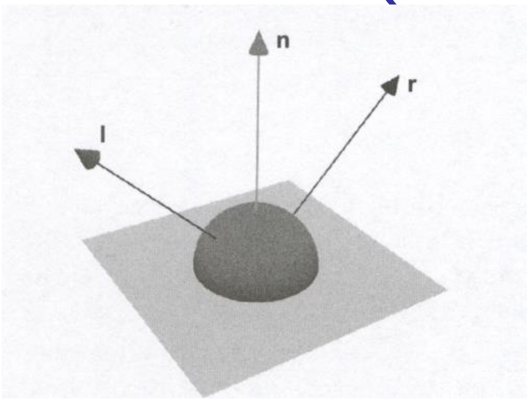
- perfectly specular, mixed, perfectly diffuse:



Angel (2000)

- imperfect specular and ideally diffuse reflection as 3D functions (BRDFs):





Bender/Brill (2003)

Shading Techniques

Flat Shading

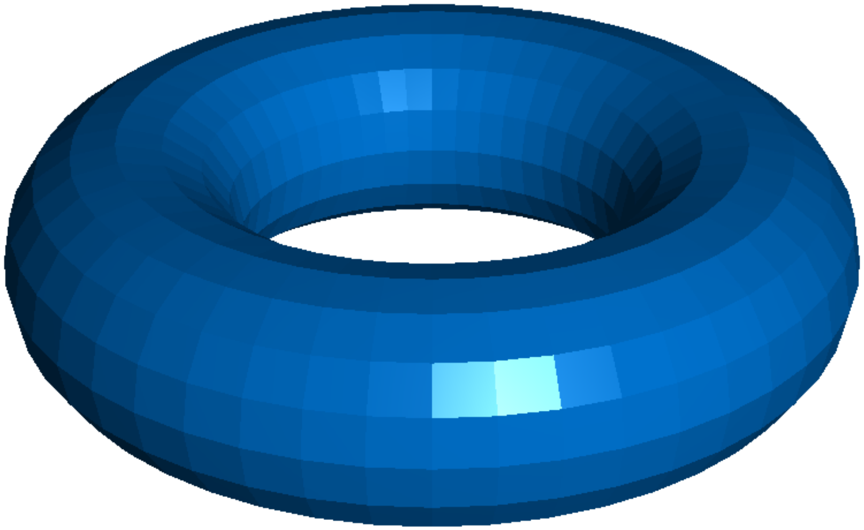
Gouraud Shading

Phong Shading

Shading of Polygonal Models

- *status*: we can approximate color at a point
- *goal*: we want to render the whole model
- *constraint*: efficiency and quality
- *approach*: **shade** all pixels of a triangle based on color computation at a few points
- *three techniques*:
  - flat shading
  - Gouraud shading
  - Phong shading (≠ Phong illumination)

Flat Shading

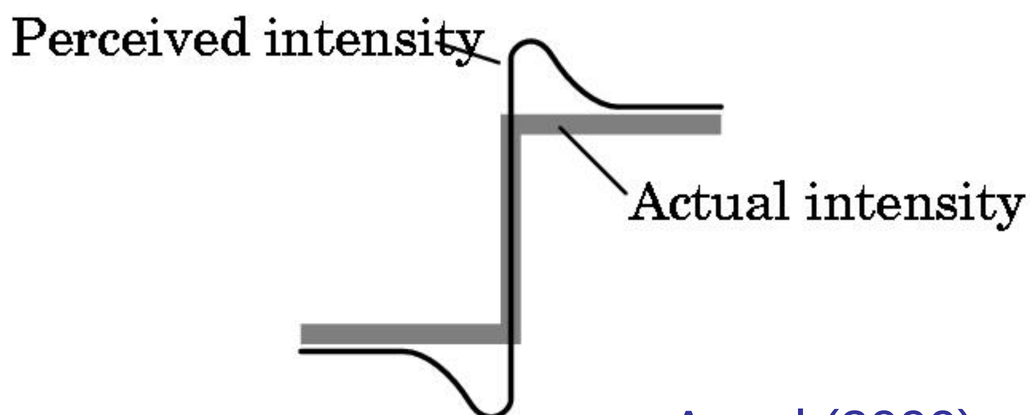


- no interpolation
- all pixels same color
- two methods:
  - one point per triangle/quad
  - average of triangle's/quad's vertices
- low quality: single primitives easily visible
- fast computation & easy implementation

Flat Shading: Edge Perception

- discontinuities easily visible and distracting
  - reason in human perception
  - contrast are enhanced by visual system
  - perceived brightness differences are bigger than the physical reality

Mach band effect

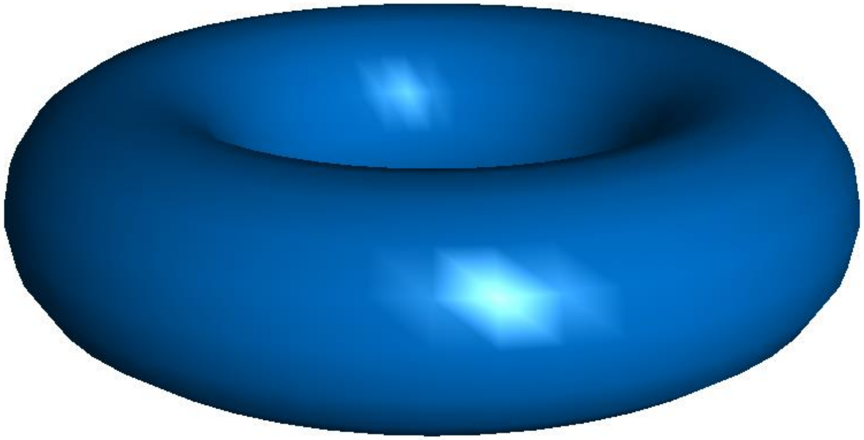


Angel (2000)

Gouraud Shading (1971)

- computation of colors at all vertices
- linear interpolation of colors over primitive
- more computation but better quality than flat shading
- usually implemented in graphics hardware
- highlights problematic: highlight shapes and highlights in the middle of triangles

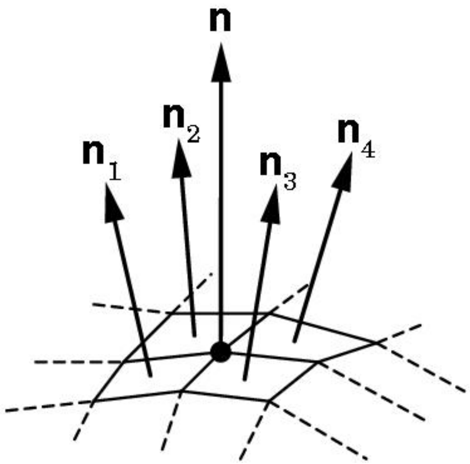
Gouraud Shading (1971)



- computation of colors at all vertices
- linear interpolation of colors over primitive
- more computation but better quality than flat shading
- usually implemented in graphics hardware
- highlights problematic: highlight shapes and highlights in the middle of triangles

Gouraud Shading: Vertex Normals

- prerequisite: normal vectors at vertices
- need to be interpolated from face normals

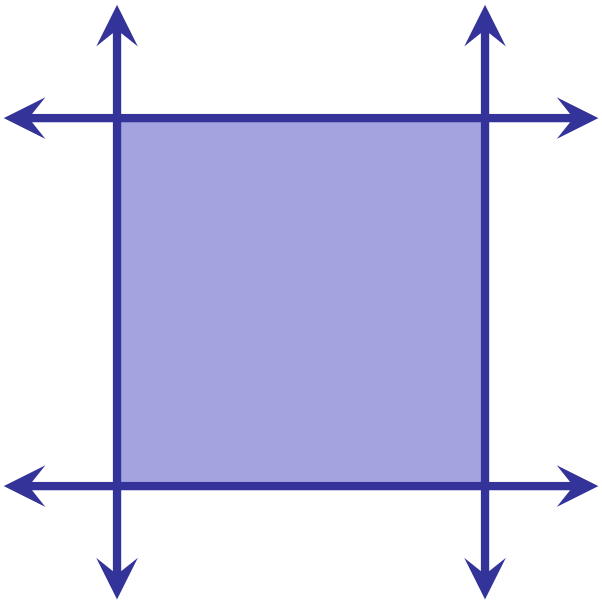


$$n = \frac{n_1 + n_2 + n_3 + n_4}{\left| n_1 + n_2 + n_3 + n_4 \right|}$$

- requires data structure supporting cross-referencing of vertices, normals, faces etc.

Gouraud Shading: Vertex Normals

- otherwise (no normal interpolation)?
- normal definition at model time
  - can be derived from modeling process
  - can capture different normals at same vertex
  - store and use these modeled vertex normals in addition to vertex positions
  - no need for vertex normal interpolation



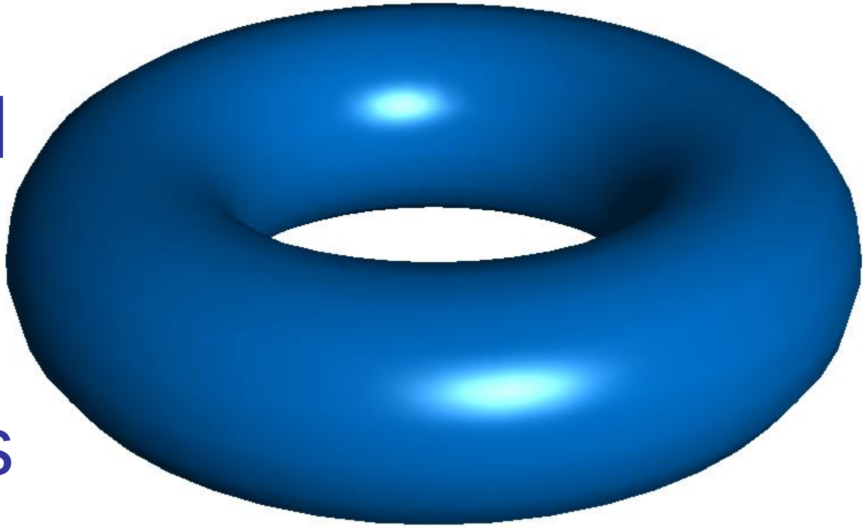
Phong Shading (1973)

- linear interpolation of normals for each pixel
- color computation for each pixel separately
- best quality, highlights are shown correctly

- but **computationally more expensive**
- problems:

Phong Shading (1973)

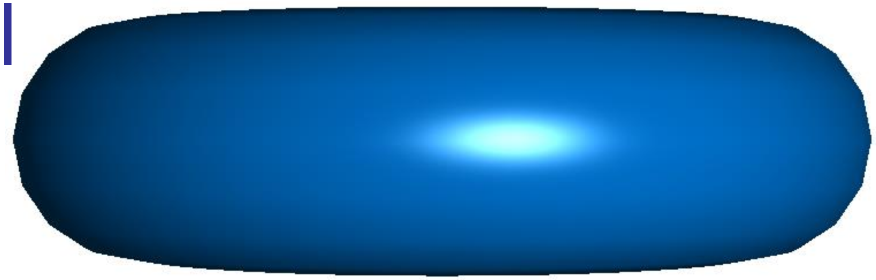
- linear interpolation of normals for each pixel
- color computation for each pixel separately
- best quality, highlights are shown correctly



- but **computationally more expensive**
- problems:

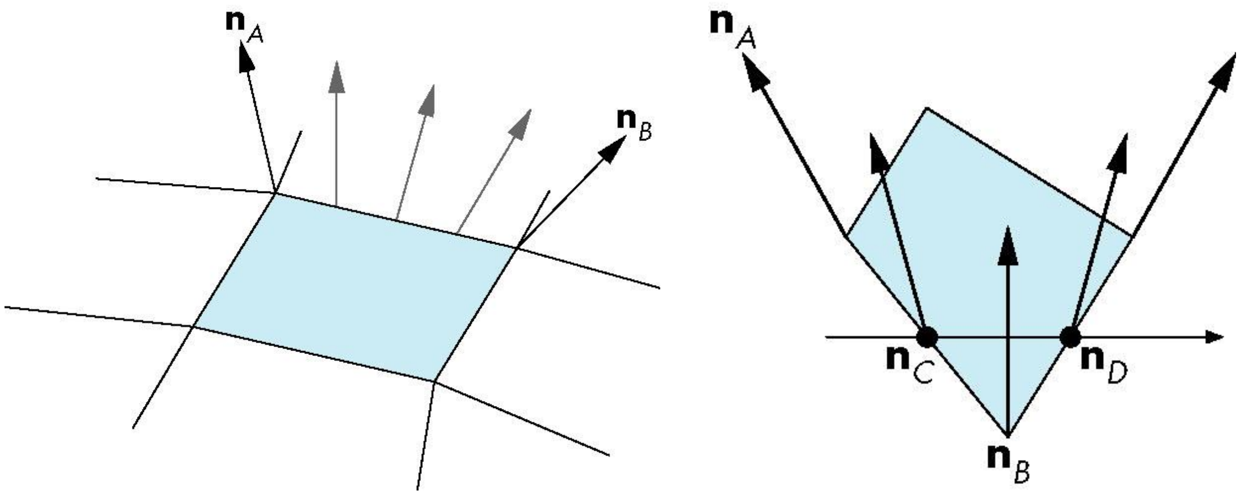
Phong Shading (1973)

- linear interpolation of normals for each pixel
- color computation for each pixel separately
- best quality, highlights are shown correctly
- but **computationally more expensive**
- problems:
  - polygons still visible at silhouettes
  - traditionally not implemented in hardware (nowadays not a problem with shaders)



Phong Shading: Normal Interpolation

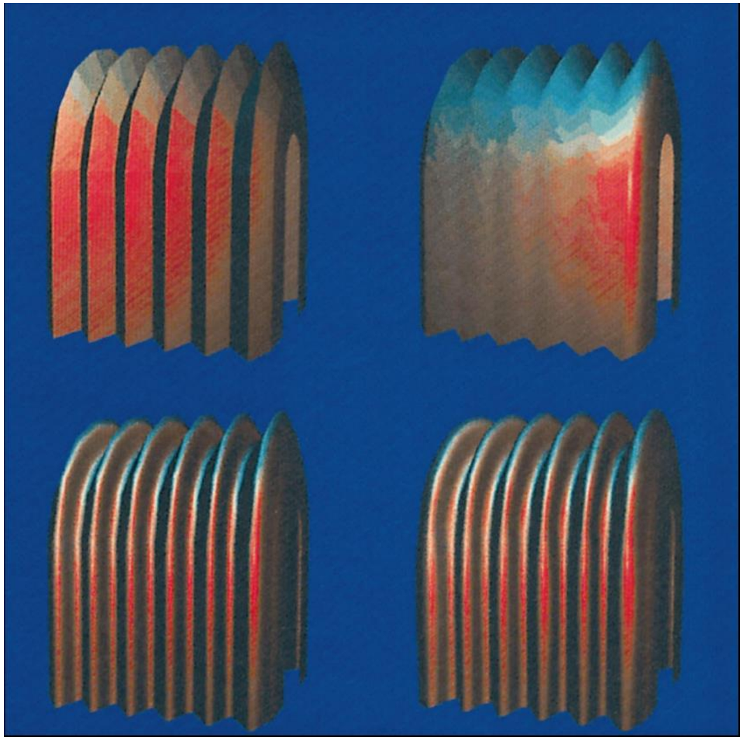
- first: normal vector interpolation along edges on per-scanline basis
- second: normal vector interpolation between edges along scanlines on per-pixel basis

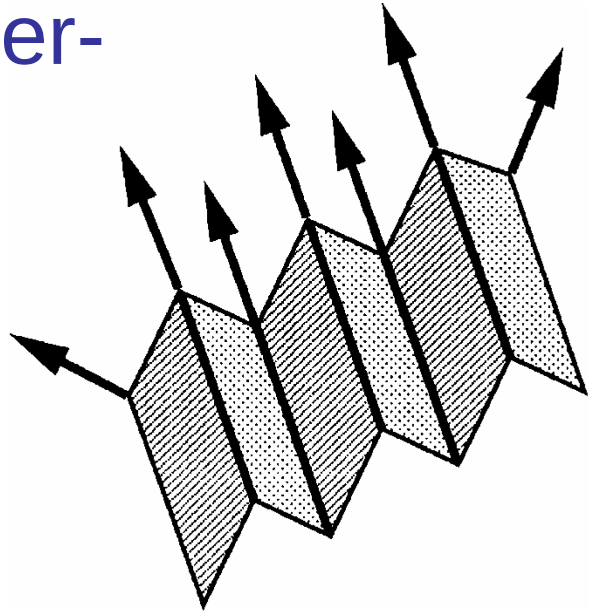


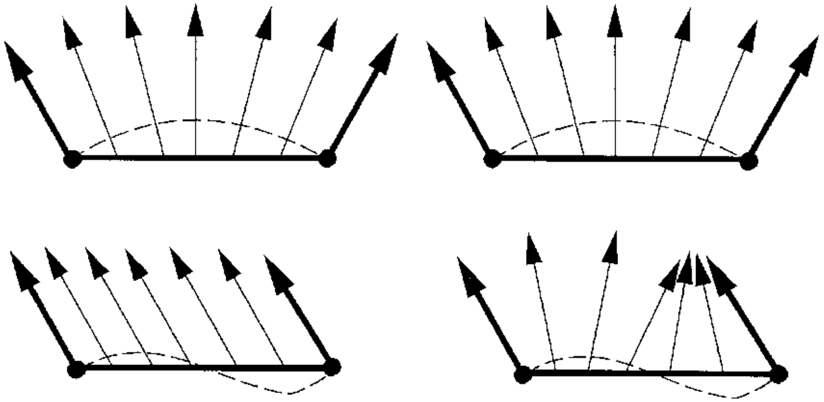
Angel (2003)

Phong Shading: Normal Interpolation

- alternative: quadratic interpolation (van Overveld and Wyvill, 1997)





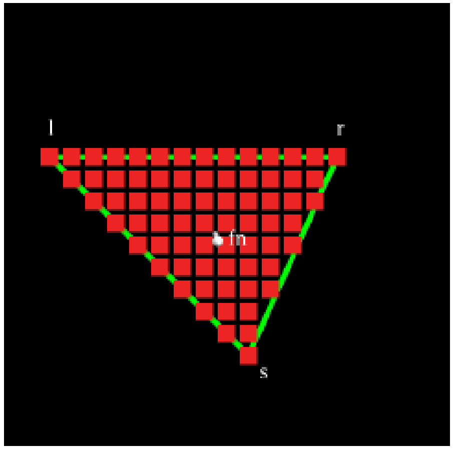


Polygonal Shading

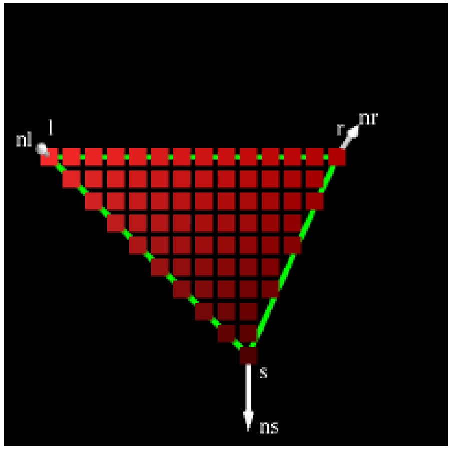
- when in the pipeline?
- typically last step together with rasterization and z-buffering
- problem:
  - normals would be perspectively projected
  - normals not perpendicular to surface anymore
  - light position w.r.t. surface may change
- determine illumination before projection!
- shading after projection

Polygonal Shading: Comparison

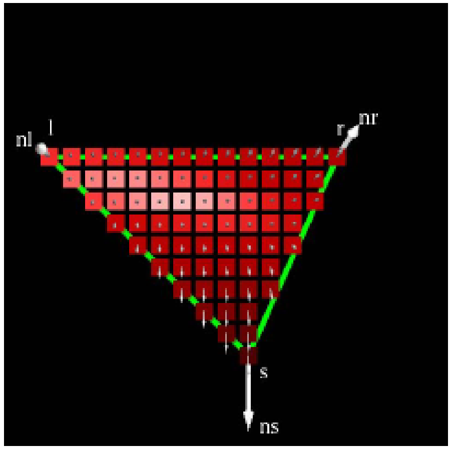
- comparison on a by-pixel basis:



flat shading

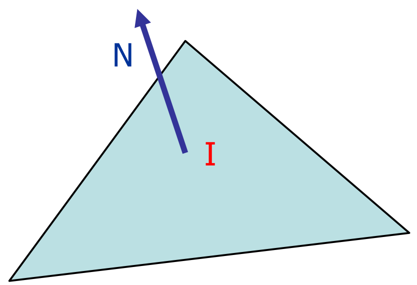
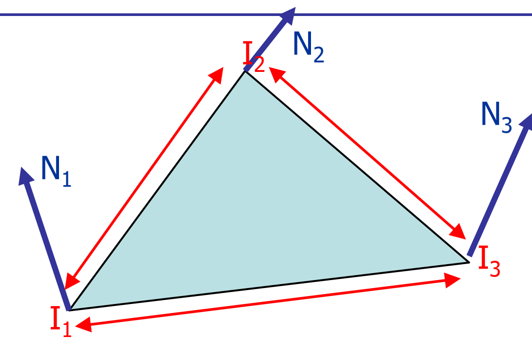
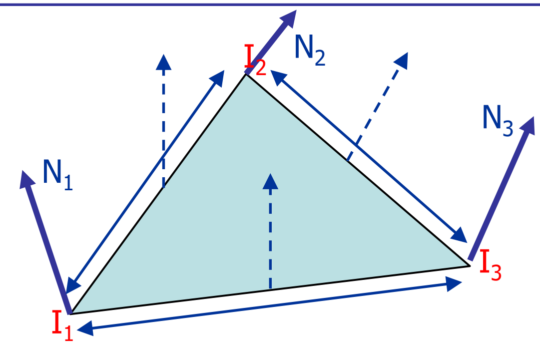
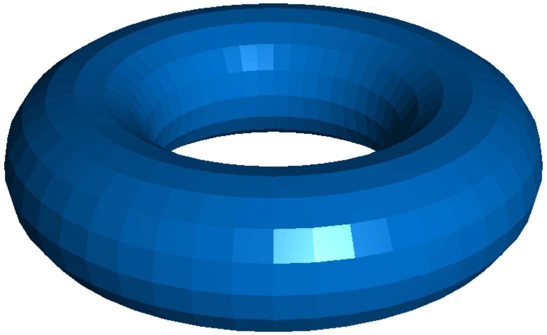
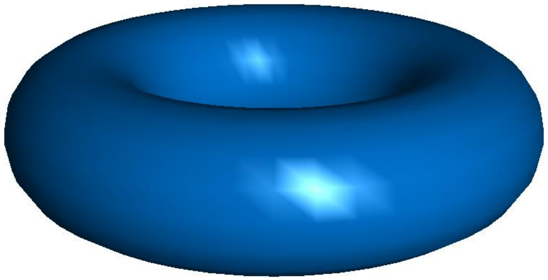
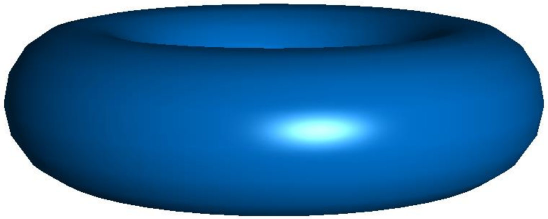


Gouraud shading

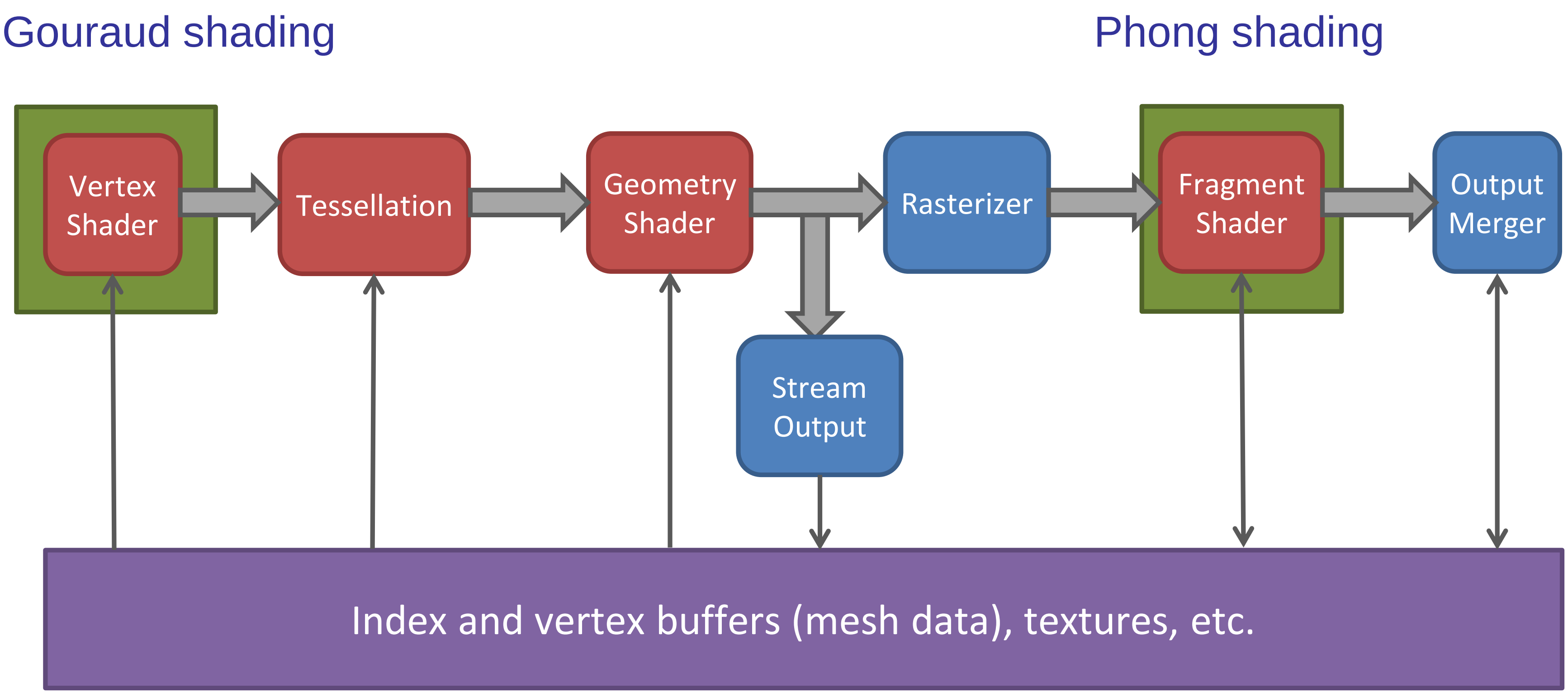


Phong shading

Polygonal Shading: Comparison

Flat	Gouraud	Phong
		
one color value for entire polygon	one color value per vertex & interpolation inside the triangle	vertex normals interpolated and one color value per pixel
		

Computation on today's GPUs



Summary

- Phong illumination model: heuristic to compute color at one surface point
  - ambient, diffuse, and specular components
  - only computes direct (local) illumination
- three shading methods to color polygons
  - flat, Gouraud, and Phong shading
  - not physically based
  - speed vs. quality tradeoff
  - illumination computation in camera coordinates (before projection)!