Visualisation, Rendering and Animation

2 VO / 1 KU (2001-2004)

Heinz Mayer, Franz Leberl & Andrej Ferko

ferko@icg.tu-graz.ac.at

Short podcast version 2020

5.
Light-Material Interaction

Illumination models

„Computer Graphics...

- … can be formulated as a radiometrically „weighted“ counterpart of computational geometry…
- … rendering is done through the application of a simulation process to quantitative models of light and materials to predict/synthesize appearance“
-

  - D. Dobkin & S. Teller, 1999

Computer Graphics...

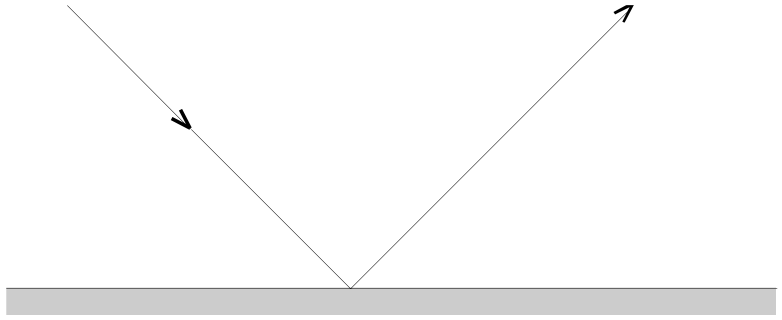
- *... must account geometry*
- *material properties: reflectance/color, refractive index, opacity, and (for light sources) emmisivity*
- *radiometry*
- *output for viewing: explicitly or implicitly psychophysics*

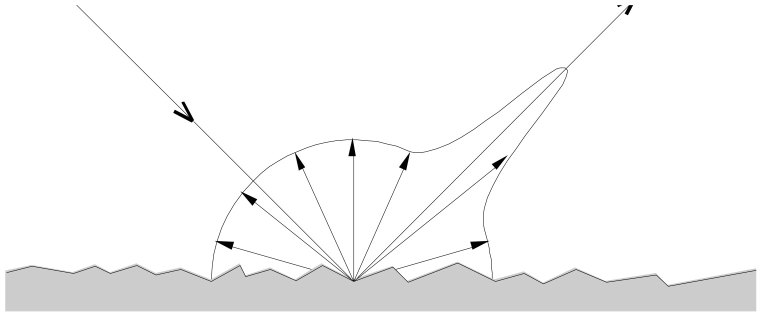
- *by D. Dobkin & S. Teller*

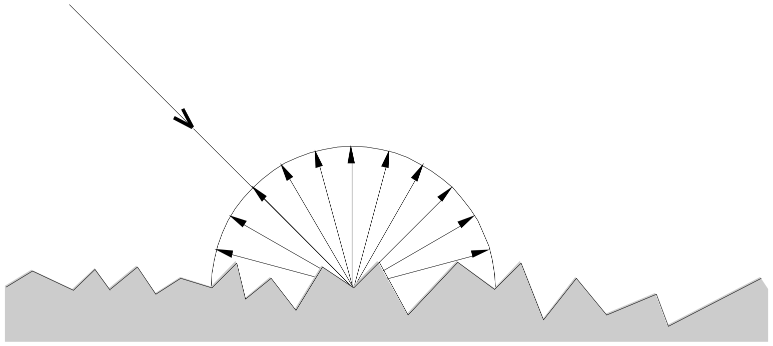
Illumination Models

- Local Illumination Models (first order)
  - Empiric Models (feasible)
  - Physical Models (possible, but expensive)
- Globale Illumination Models (second order)
  - Ray-Tracing (photons)
  - Radiosity (waves, „key is the light“)

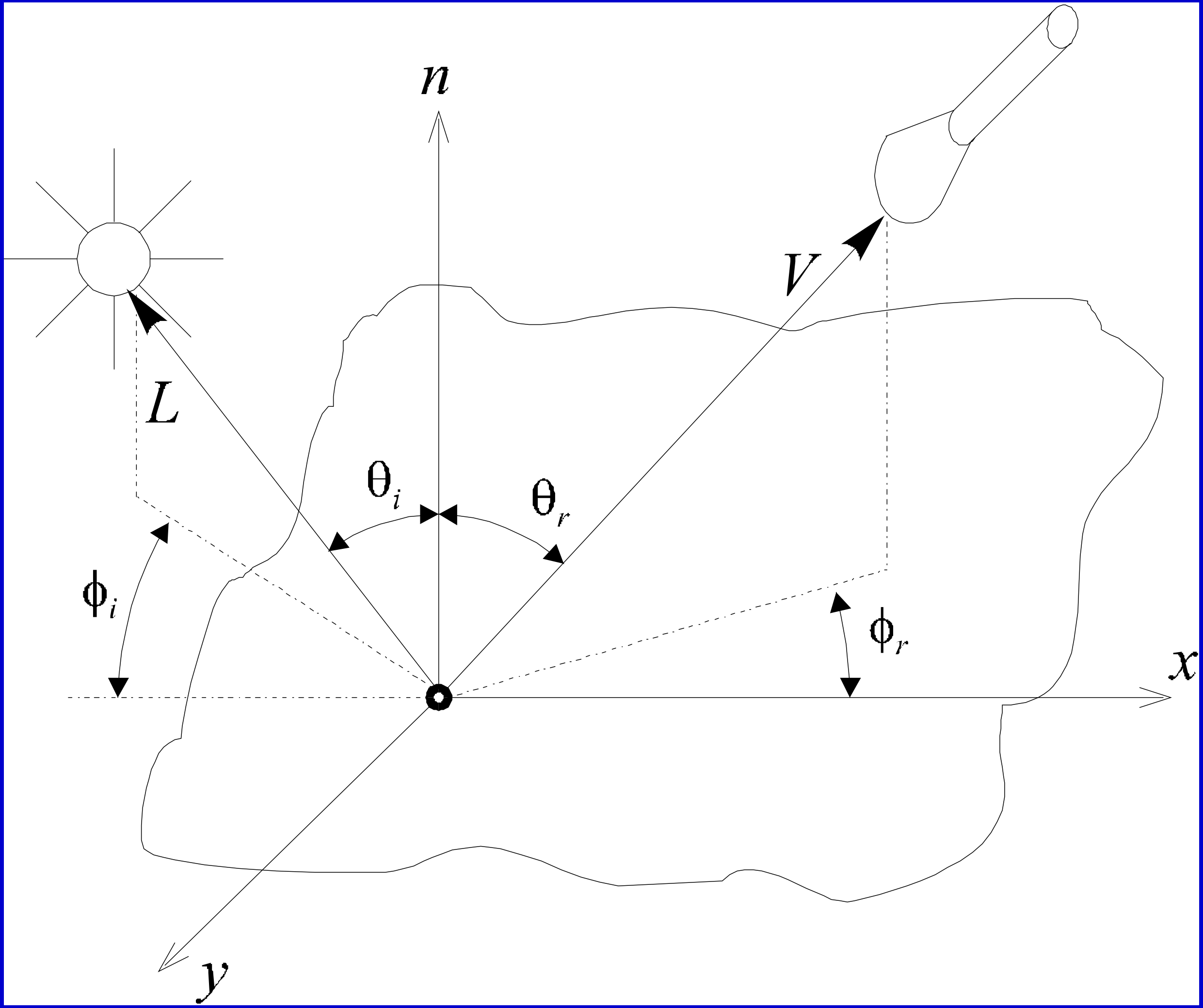
Reflexion Properties


Perfect Specular


Imperfect Specular


Diffuse

BRDF



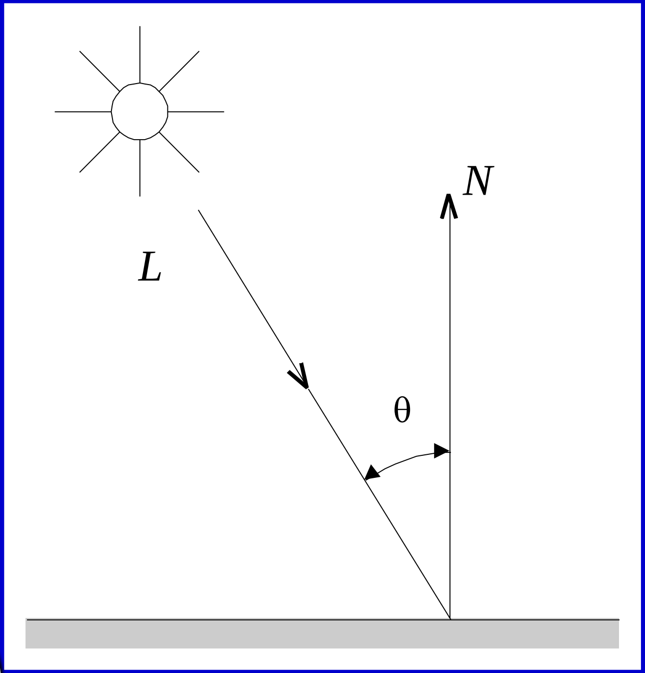
Ambient Light

- Daylight (diffuse, undirected) lightsource
- Intensity in the given scene constant
- Multiple reflections on surfaces in the scene
- Trivial Illumination Model: $I = I_a\, k_a$

I_a	intensity of ambient light
k_a	ambient reflection coeficient

Lambertian Illumination Model

- *Directional lightsource(s) added*
- *Diffuse reflection: independent from the camera position*
- *Illumination Model:*
  $I = I_p\, k_d \cos\theta = I_p\, k_d\, (N \bullet L)$



I_p *Intensity of directional lightsource, point*
k_d *diffuse reflection coeficient*

Intensity attenuation

- **Intensity contribution:**
  d_L lightsource distance

$$f_{att} = \frac{1}{d_L^2}$$

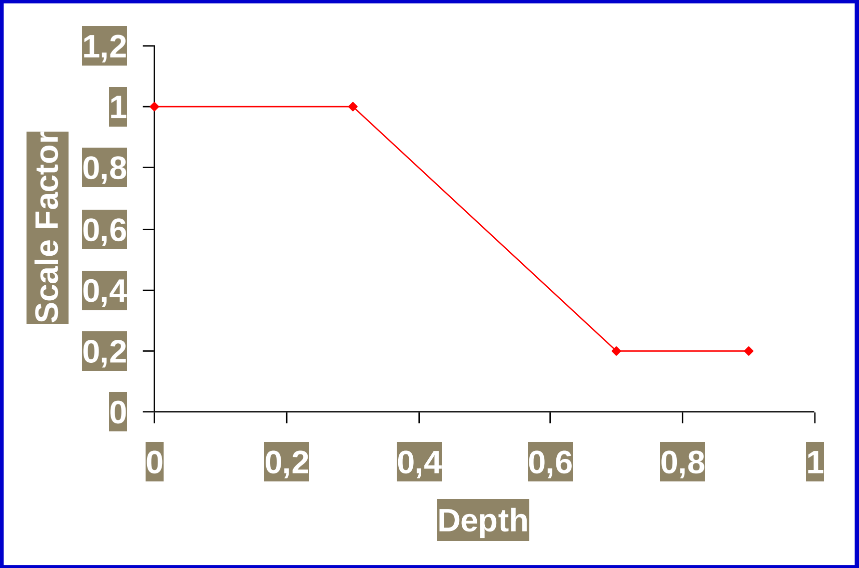
- **Alternative representation:**

$$f_{att} = \min\left(\frac{1}{c_1 + c_2 d_L + c_3 d_L^2}, 1\right)$$

- **Lighting model: $I = I_a k_a + f_{att} I_p k_d (N \bullet L)$**

Depth-cueing

- Distant objects appear darker (optionally „color-shift“, too)
- „Athmospheric perspective“



- Linear interpolation: $I' = s_0 I_f + (1- s_0)I_b$
- Scaling between „front/backplane“

Shaders, shading models

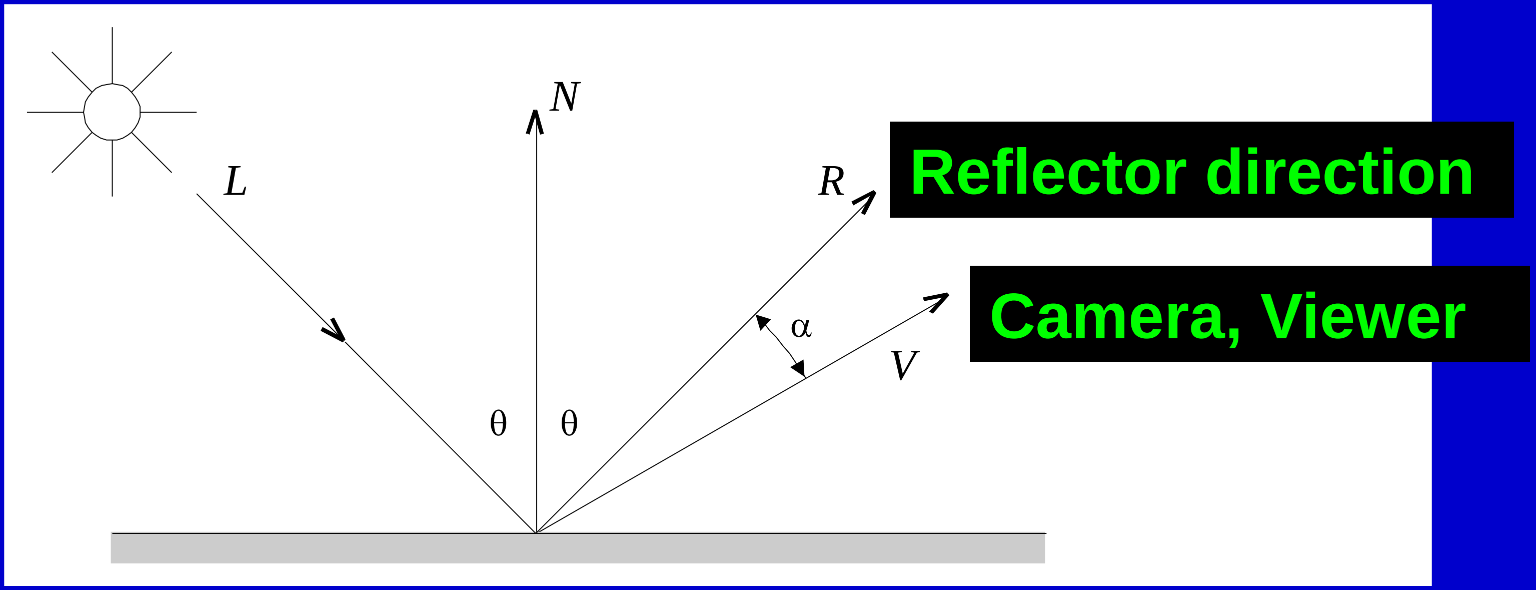
- Fill polygons after transformations and rasterization by color values
- Flat-Shading:
  - – Lamberts illumination model
  - – single color value for each polygon/triangle
  - – advantage: very fast
  - – drawbacks: Mach-bands, causing nonrealistic appearance
- Better ones: Gouraud-, Phong-Shading

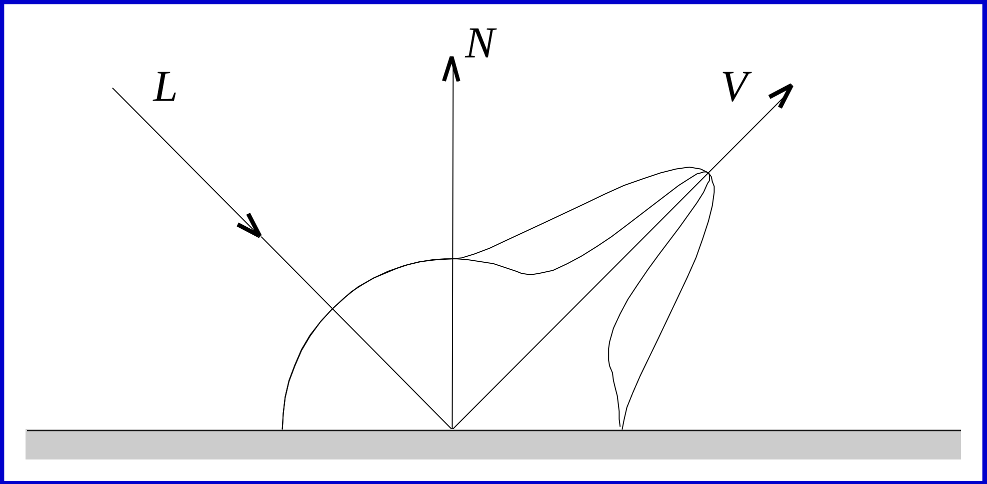
Phong Illumination Model

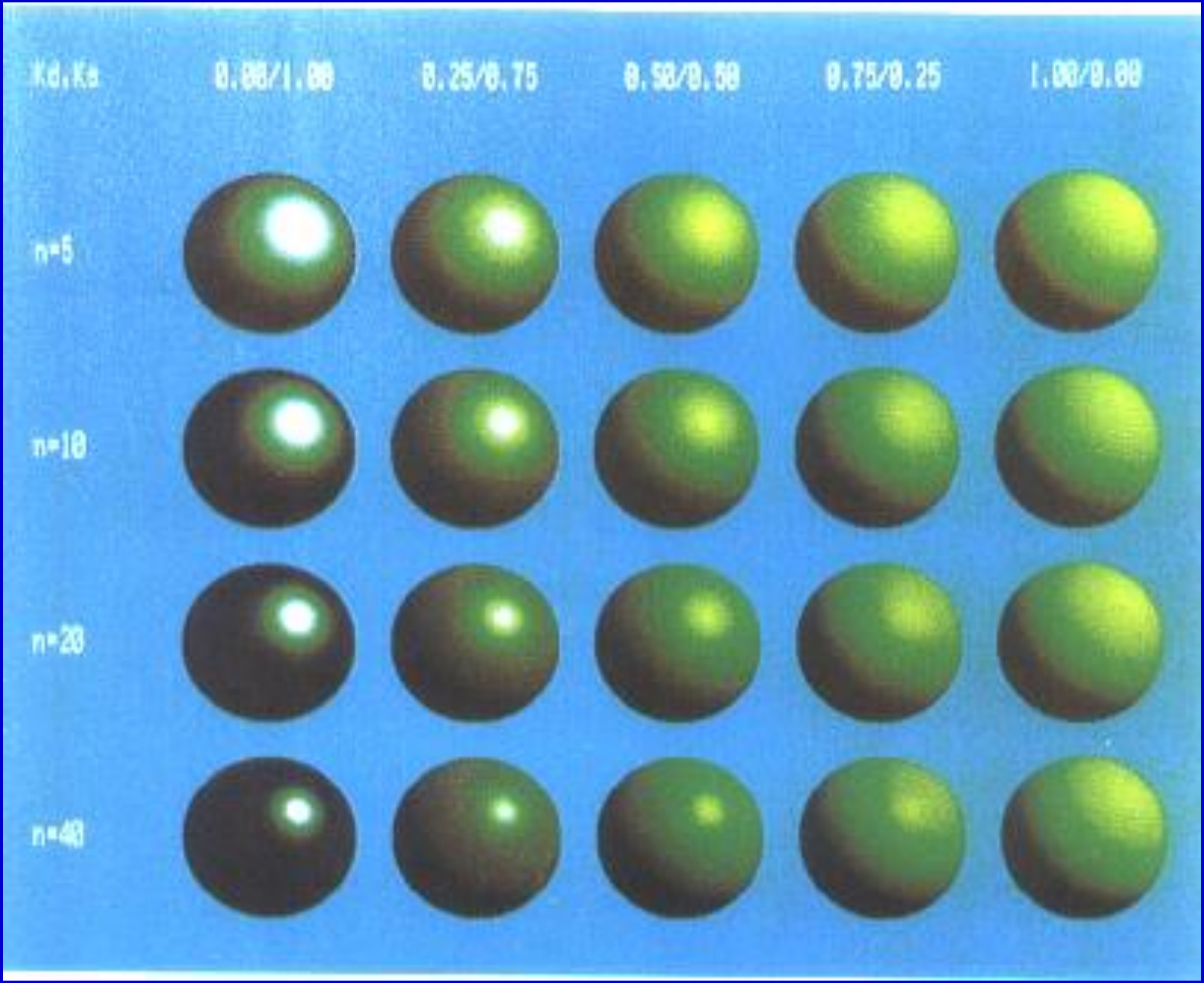
- Adding specular reflection (depends on camera position)
- New Illumination Model:

$$I = I_a\, k_a + f_{att}\, I_p\, (k_d \cos\theta + k_s \cos^n \alpha) =$$
$$I_a\, k_a + f_{att}\, I_p\, [k_d\, (N \cdot L) + k_s\, (R \cdot V)^n]$$

k_d	diffuse reflection coeficient
n	(Spiegelneigung), „shininess“ parameter
R	Reflected photon direction vector
V	Viewer/Camera direction vector

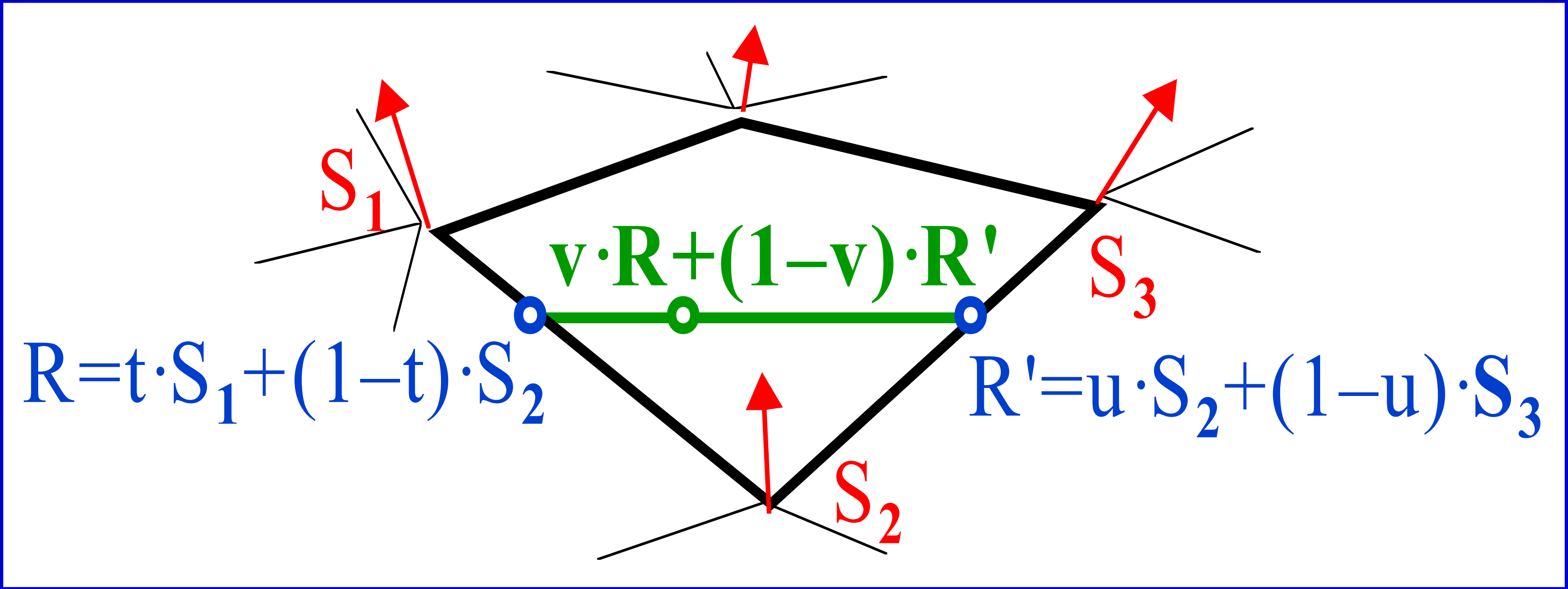

N
L
R
α
V
θ
θ
Reflector direction
Camera, Viewer


N
L
V


Kd,Ka
0.00/1.00
0.25/0.75
0.50/0.50
0.75/0.25
1.00/0.00
n=5
n=10
n=20
n=40

Gouraud Shader

- Lambert / Phong Illumination Model
- Color values in the vertices
- Normal vectors given by:
  - Face normals
  - Face normals averaging
- Linear Interpolation of Color:
  - Along the edges
  - Along the „scan-lines“
- Drawback: quality
- Advantage: speed



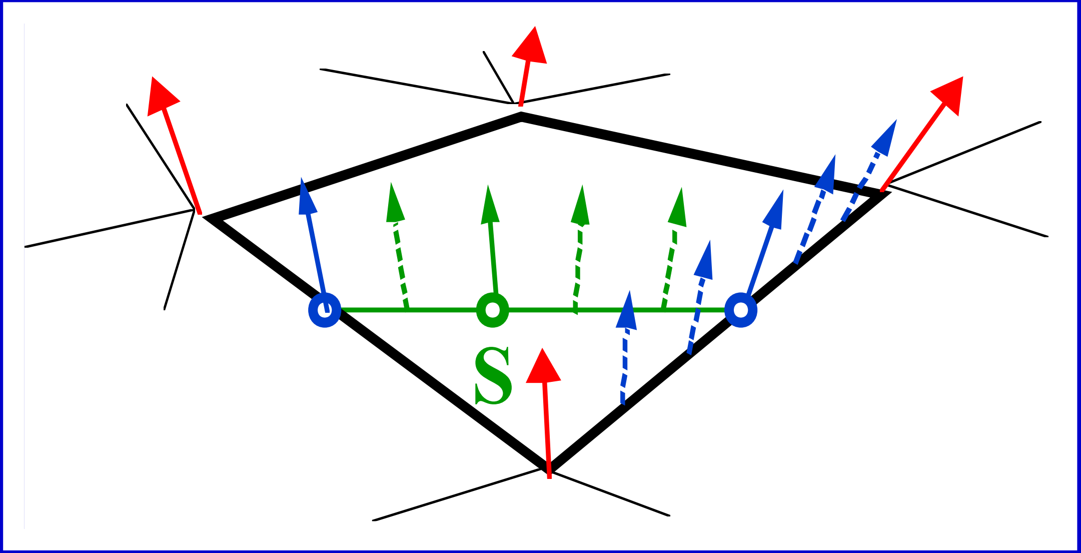
S_1
$v \cdot R+(1-v) \cdot R'$
S_3
$R=t \cdot S_1+(1-t) \cdot S_2$
$R'=u \cdot S_2+(1-u) \cdot S_3$
S_2

1. ***find normal vectors at corners and calculate shading (intensities) there***
2. ***interpolate intensities along the edges linearly***
3. ***interpolate intensities along scanlines linearly***

Phong Shader, phong

- Normal vectors like Gouraud
- Linear interpolation of normal vectors instead of color value
- Color computation per pixel
- Pro: specular highlights in the given polygon rendered correctly
- Con: computationally expensive

phong



1. normal vectors at vertices
2. interpolate normal vectors along the edges
3. interpolate normal vectors along scanlines and calculate shading (intensities) for every pixel

Interpolation Problems

- Polygon-Silhouette
- Interpolations artefacts from the given „scanline“ - direction
  - Orientation dependent
  - Perspective dependent
- Not representative vertex no
- <u>Hint:</u> refine the triangulation (more complex model)

Rendering Polygonal Scene

- 1. Extract polygons from the database
- 2. Transform to WC and VRC
- 3. Backface culling and visibility
- 4. Clip against the visible volume
- 5. Projection of clipped polygons
- 6. Shading by incremental shader:
  - 1. Rasterize,
  - 2. Depth and visibility, (z-buffer)
  - 3. Shading (constant, Gouraud, Phong…)

Local Illumination Summary

- Empirical shading models
  - constant, Gouraud, Phong...
- Ambient, diffuse and specular reflection
- Light rays only
- Polygonal scenes
- Rendering summary (polygonal case)
- More: transparency, bumpy surfaces, textures, global illumination, animation...

Local Illumination Online

- ***Applet by Patrick Min at***

  http://www.cs.princeton.edu/~min/cs426/classes/light.html

- *http://www.siggraph.org/education/materials/HyperGraph/illumin/illum0.htm*
- *http://www.siggraph.org/education/materials/HyperGraph/illumin/vrml/pellucid.html*

Definition of Light Sources

- Point light source
- Multiple point sources… <u>area</u>
- 4 abstract lightsources -
  ambient, directional, point, flood
- intensity/fog = I/(a*d*d…*d + b)
- flood: powers of cosine (Phong)

Lighting Optimization

(polygonal model, light sources)

In: Real-Time Rendering [Moeller-Haines, 1999, pp. 248n]

www.realtimerendering.com, 2nd edition in the year 2002

- 1. Is lighting needed?
  - Sometimes, higher constant for ambient light helps
  - Alternatives: texturing,
  - texturing with colors at the vertices,
  - colors at the vertices

  - Example: background polygon

Lighting Optimization 2

- *Light speed constant, unlike lighting speed :-)*

- *Directional light sources, point lights, spot lights*
- *Directional light source gives normalised vector*
- *Spot light: in-cone test for vertices, exponential fall-off from the center of the light cone*
- *Gray-scale light source is faster than a colored light source*
- *The number of light sources*

  (some graphics accelerators are optimized for lighting calculations with only one source)

Lighting Optimization 3

- Non-local viewer trick (camera at infinity)

- Simplification of specular highlight computation and environment mapping calculations
- Normally, the vector from the eye to the vertex being illuminated is computed and normalised
- This is replaced by (0,0,-1) or (0,0,1) - if the viewer is looking along the positive z-axis
- With directional lights this means that the halfway vector h is computed only once for each scene
- The error (slight shift in the locations of the specular highlights) of this simplification is not usually not detectable

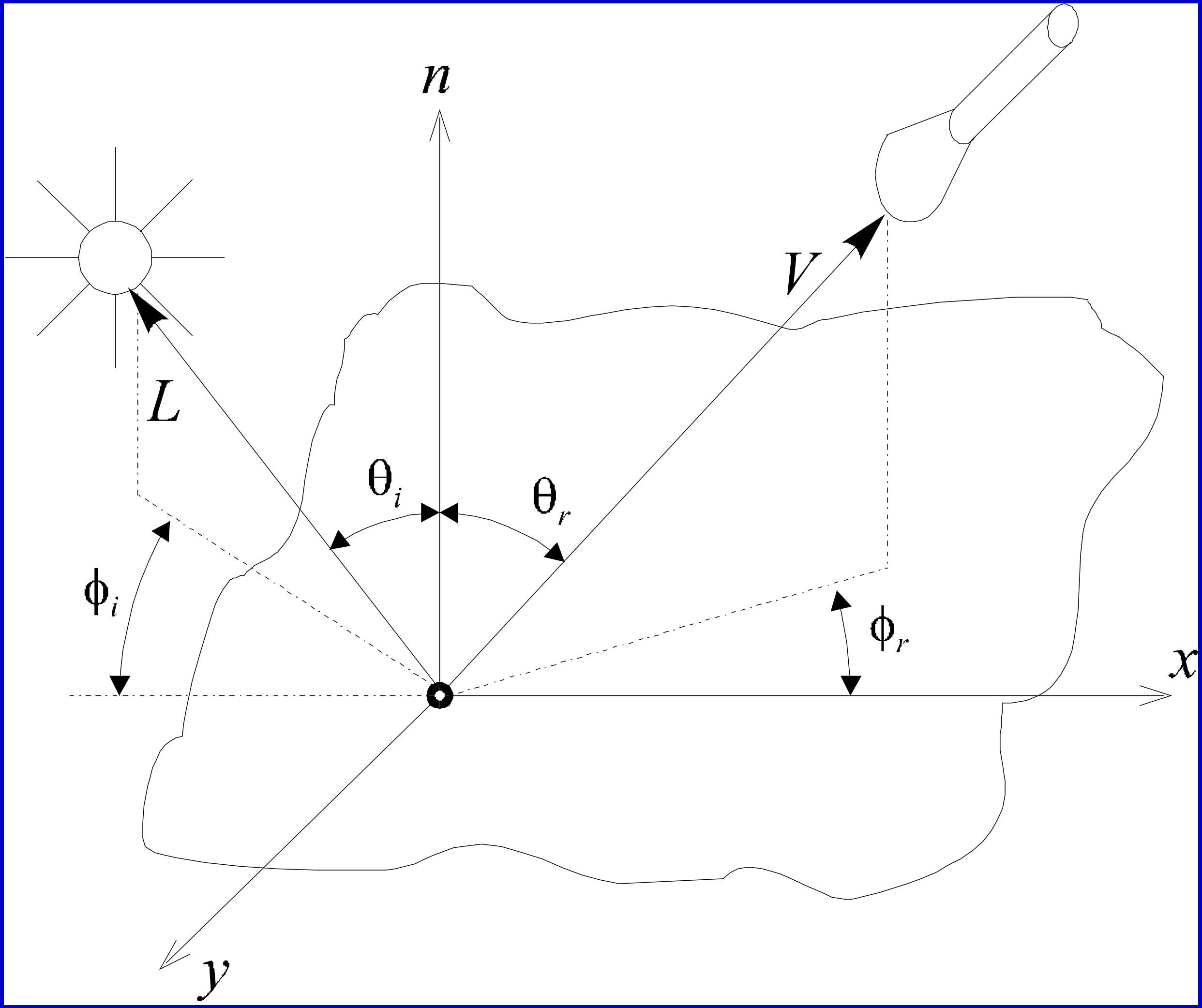
Lighting Optimization 4+

– *4. If the model will be not scaled during rendering all normals should be normalised as a preprocess*
– *5. Flat shading is faster than the Gouraud one*
– *6. If a non needed lighting is computed for both sides of a polygon then turn this feature off*
– *7. If possible, set the material specular component to (0,0,0) to supress the expensive higlight calculations*

Lighting Optimization 8

8. If the light sources are static with respect to geometry, and the polygonal data is attached to a material without specular parameters, then the diffuse and ambient lighting can be precomputed and stored as colors at the vertices. Simple shadows can be captured by direct visibility tests between lights and vertices. Even the radiosity can also be precomputed and stored as colors at the vertices or as light maps. Specular (view dependent) contributions can be added in (no highlights in areas which are in shadow)…

BRDF



Visualisation, Rendering and Animation

2 VO / 1 KU (2001-2004)

Heinz Mayer, Franz Leberl & Andrej Ferko

ferko@icg.tu-graz.ac.at

Short podcast version 2020