Shading

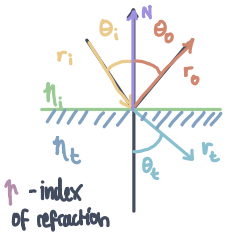
↳ after visible surface for pixel is known (ray hit), compute pixel value (color) by evaluating a Shading Model:

Basic Idea of Shading Model: Capture process of Light reflection

→ surfaces
- are illuminated by light sources
- reflect part of light to camera

Geometry of Reflection & Refraction

Snell's Law



η - index of refraction

r_i : incident ray → ray of light that hits a surface

θ_i : angle of incidence → the angle between r_i and the surface normal N (perpendicular line to surface)

r_o : reflected ray → ray of light that is reflected by the surface of a r_i

θ_o : angle of reflection → angle between r_m and N

r_t : refracted/transmitted ray → ray of light which is transmitted through the surface of a r_i

θ_t : angle of refraction → angle between r_t and N

η : refraction index of material/medium : quantity describing how fast light travels through the material

Law of Reflection
for a specular surface θ_o is always equal to θ_i

Law of Conservation of Energy
power of r_i must equal the sum of power in r_o, r_t and any power absorbed in the surface

$$\eta_i \cdot \sin\theta_i = \eta_t \cdot \sin\theta_t \Leftrightarrow \frac{\sin\theta_i}{\sin\theta_t} = \frac{\eta_t}{\eta_i}$$

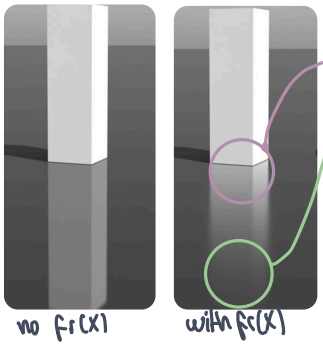
So we have: one refracted ray r_t
one reflected ray r_o which is the mirror of the incident ray r_i around the surface normal N

And Snell's Law governs the geometry of r_i and r_t (how they look like) as well as allowing us to calc. the geometry of r_o

Amount of Light transmitted / reflected

Fresnel Effect: describes reflection and transmission of light when r_i hits surface

→ depending on θ_i the intensity of r_o and r_t will change (non-linearly)

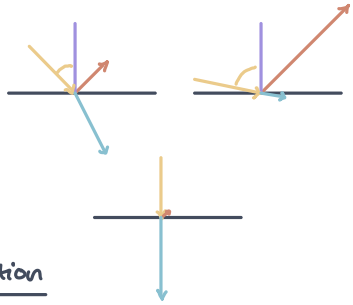


no fr(x) with fr(x)

- θ_i increasing → reflectance ↑ refraction ↓
- θ_i decreasing → reflectance ↓ refraction ↑

⇒ reflectance increases with grazing angle
⇒ refraction decreases with grazing angle

Skizze



Schlicks approximation based on normal reflection

if $\theta_i = 0 \Rightarrow R_0 = \left(\frac{n_1 - n_2}{n_1 - n_2}\right)^2$

$$R \approx R_0 + (1 - R_0) \cdot (1 - \cos\theta_i)^5$$

Reflection Types → Diffuse, Glossy, Mirror, Anisoptric, Translucent, Transparent

- **Diffuse**: equal light distribution in direction & brightness
  ↳ view-independent

- **Glossy**: light scattered in "cone" of directions

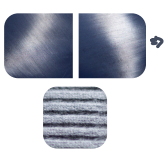
glossy basically a "rougher" version of mirror

view angle affects brightness of reflected ray ⇒ view-dependent

- **Mirror**: light reflected in one direction

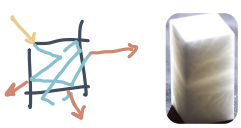
energy of light carried in one direction ⇒ view independent

- **Anisoptric**: material structure affects visible light ("texture") reflected



⇒ view-dependent

- **Translucent**: Light enters material and interacts with the "particles" inside of it



gets absorbed or reflected

then light eventually leaves material ⇒ direction is arbitary due to the unpredictable bounces inside

- **Transparent**: Light reflected in one direction and also enters material and leaving somewhere else

changes direction on entry
↓
changes direction when leaving

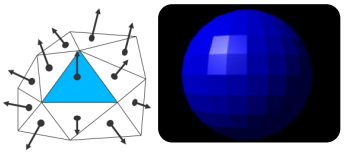
⇒ view of distorted image behind object

Shading - Evaluation of Reflections

→ use BRDF for representing/characterizing reflectance
basically
↳ ratio of reflected radiance to incident irradience

- **Shading Types**

- **Flat Shading**
  - per surface normal
  - ⇒ flat shading
  - ⇒ not good for spheres

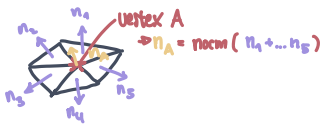


- **Gocaud Shading**
  - per vertex normal

→ just one method

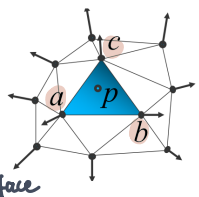
1. get vertex normals by averaging all surface normals of polygons containing this vertex

$$\vec{n}_v = \text{norm}(n_1 + \ldots + n_k)$$



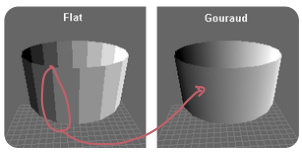
2. evaluate reflectance model at each vertex to get vertex intensity L_v

$$L_v \approx f_r(\vec{\omega}_o, \vec{n}_v, \vec{\omega}_i) \cdot L_i$$



3. Interpolate vertex intensities for every point on surface

$$L_p = \lambda_1 \cdot L_a + \lambda_2 \cdot L_b + \lambda_3 \cdot L_c \quad \rightarrow \text{with barycentric-coordinates}$$



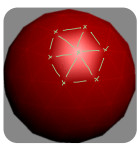
- blurring of edges ⇒ smooth transitions
- gradient

- not quite right highlighting in some cases (most apparent in sphere)

Problems: 1) Highlight dependent on normal vector on hit point (like on an apple) disappears if vertices are not on that spot

⇒ no highlight returned

2) Highlight overlapping vertices leads to correct render at vertex but will spread "unnaturally"



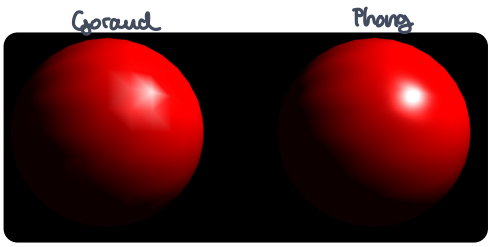
- **Phong Shading** (also called Phong interpolation)
  - similar to Gorcaud:

1. get vertex normals $\vec{n}_v$ of polygon containing hit point
2. interpolate vertex normals to get surface normal of point p in polygon

$$\vec{n}_p = \text{norm}(\lambda_1 \vec{n}_a + \lambda_2 \vec{n}_b + \lambda_3 \vec{n}_c) \Rightarrow \text{barycentric}$$

3. evaluate reflectance model at every point in polygon

$$L_p \approx f_r(\vec{\omega}_o, \vec{n}_p, \vec{\omega}_i) \cdot L_i$$



  - Differences:
    - → Phong interpolates the vertices normals to get the normal for a point in the polygon then using this normal to evaluate the intensity for that point
    - → whereas Gorcaud evaluates the intensities at the vertices normals and then interpolates between these intensities for each point in the polygon

⇒ Phong has to determine for each point the intensity value (evaluate individually)
Gorcaud evaluates the vertices "once", then uses interpolation to shade whole polygon

→ Phong computationally expensive

Reflection Models

↳ describes how the evaluation process → implementations of BRDF

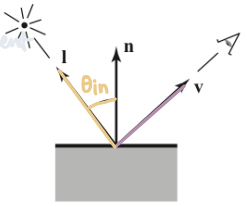
For one Light Source (usually Point Lights):

- **Lambertian - Shading**
  - ↳ returns **diffuse reflection**

$$L_D = L_i \cdot k_d \cdot \left[\cos(\theta_{in}) \text{ or } i \cdot n \right]$$

L_i: intensity coming in; k_d: surface color

no regard to view ⇒ view-independent



- depends on angle θ_{in} (angle of surface to light)
  - → n facing directly towards light ($n \parallel i$) = maximum light on surface
  - → n tangent or facing away to light ($n \perp i$) = no light
  - → in between: light on surface is proportional to $\cos(\theta_{in})$

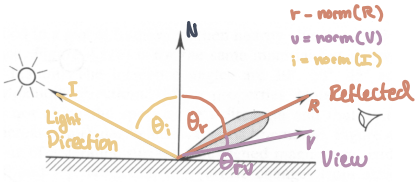
surface normal

- **Phong Reflection Model**
  - ↳ returns **specular reflection** of (cosine power) lobe distribution

$$L_S = L_i \cdot k_s \cdot \left[\cos^{k_e}(\theta_{rv}) \text{ or } (\vec{r} \cdot \vec{v})^{k_e} \right]$$

L_i: intensity coming in; k_s: maybe to simulate "fall off" / absorption of material

basic reflectance distribution



Consists of:

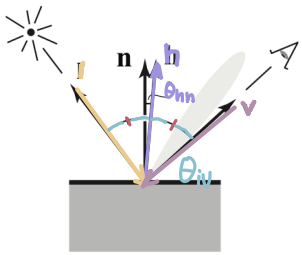
- cosine lobe: Reflection falls off when angle $\theta_{rv} = \vec{r} \cdot \vec{v}$ (dot product) gets larger
- Phong Coefficient k_s: steer strength of Reflection
- Phong Exponent k_e: "shape" or size of highlight → k_e ↑: lobe ↓ ⇒ controls "shininess" of surface
  - k_e ↓: lobe ↑

Problems:

- not "realistic" → no conservation of energy
  - → non-reciprocal (not bidirectional)
- plastic-like appearance

- **Blinn-Phong Model** (better version of Phong)

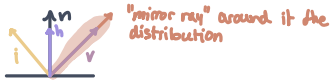
↳ returns **specular reflection** of (cosine power) lobe distribution



$$L_s = L_i \cdot k_s \cdot \left[\cos^{k_e}(\theta_{hn}) \text{ or } (\vec{n}\cdot\vec{h})^{k_e}\right]$$

L_i: intensity coming in; k_s: maybe to simulate "fall off" / absorption of material; basic reflectance distribution

- uses half-way vector $h = \frac{i+v}{\|i+v\|}$ (which is the vector bisecting angle θ_{iv}) and Surface Normal $\vec{n}$ to create lobe distribution
- when i and v are symmetrically positioned across $\vec{n}$ ⇒ $\vec{n}$ and h aligned ⇔ r and v aligned ⇒ mirror reflection



Reflection decreases smoothly when deviating from the mirror reflection alignment

- Difference to Phong:
  - uses $\cos(\theta_{hn})$ instead of $\cos(\theta_{rv})$
  - energy conserving
  - computation faster

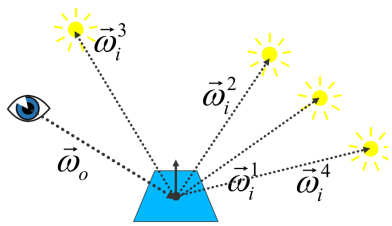
For multiple Light Sources:

- **Phong-Illumination Model**

$$L_r = k_a L_{i,a} + k_d \sum_l L_l \overbrace{(I_l \cdot N)}^{\cos(\theta)} + k_s \sum_l L_l \overbrace{(R(I_l)\cdot V)^{k_e}}^{\cos(\theta)\ \text{Phong}} \quad \text{or} \quad + k_s \sum_l L_l \overbrace{(H_l \cdot N)^{k_e}}^{\cos(\theta)\ \text{Blinn}}$$

Ambient (↳ contradicts physics) | Diffuse | Specular/Glossy | Specular/Glossy

Ambient: surfaces without any illumination, won't be completely dark (black)
- ⇒ add a constant "ambient" light to surfaces (when not ray-traced & viewed)
- ⇒ simulate Global Illumination



Diffuse: determines surface color

Glossy: highlights based on view

Occlusion: Point on surface in shadow from some object
- ➡ trace ray to Light source ⇒ test for occlusion (collision)

$$L_r = \sum_N f(\vec{\omega}_o, \vec{\omega}_i^k)\, v(p, \vec{\omega}_i^k)\, L_i^k \cos(\theta^k)$$

- is a heuristic Model
  - ⇒ contradicts physics (constant Ambient Light)
  - ⇒ purely local Illumination
    - ↳ direct light from sources
    - ↳ no further reflection on other surfaces
    - ⇒ only relies on source direction and viewing direction (no other properties like material, ...)