Topic 12:

Basic Ray Tracing

- Introduction to ray tracing
- Computing rays
- Computing intersections
  - ray-triangle
  - ray-polygon
  - ray-quadric
  - the scene signature
- Computing normals
- Evaluating shading model
- Spawning rays
- Incorporating transmission
  - refraction
  - ray-spawning & refraction

Computing Ray-Object Intersections

Basic loop:

for each pixel $\bar{q}$

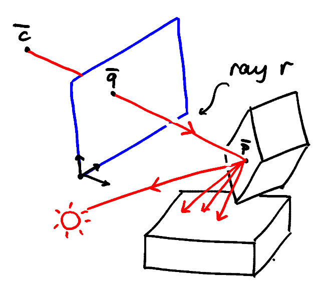
① cast ray r through $\bar{q}$

② find 1st intersection of $\bar{q}$ with scene (i.e. point $\bar{p}$)

③ estimate amount of light reaching $\bar{p}$

④ estimate amount of light travelling from $\bar{p}$ to $\bar{q}$ along ray r

computing ray-scene intersections

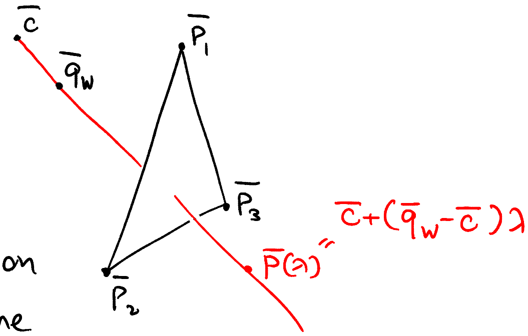


Computing Ray-Triangle Intersections

Algorithm #1

(a.) compute normal

$\vec{n} = (\bar{P}_2 - \bar{P}_1) \times (\bar{P}_3 - \bar{P}_1)$



(b.) compute intersection of ray and plane of triangle

i.e. find λ^* that satisfies

$$\left[\bar{P}_1 - \bar{P}(\lambda^*)\right] \cdot \vec{n} = 0$$

(c.) verify that $p(\lambda^*)$ falls within triangle

e.g. using half-space constraints (assignment 1)

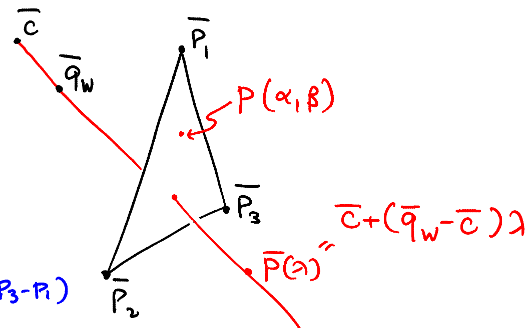
Computing Ray-Triangle Intersections

Algorithm #2

(a.) Parameterize the triangle plane

$p(\alpha, \beta) =$

$P_1 + \alpha(P_2 - P_1) + \beta(P_3 - P_1)$



(b.) Find α, β, λ^* that satisfy $p(\alpha, \beta) = p(\lambda^*)$

vectors expressed in euclidean 3D coords

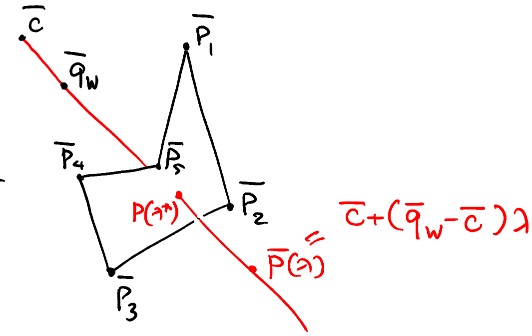
unknowns

$$\Rightarrow \text{solve} \quad \left[\underbrace{-(\bar{P}_2 - \bar{P}_1) \;\; -(\bar{P}_3 - \bar{P}_1) \;\; (\bar{q}_w - \bar{c})}_{3\times 3 \text{ matrix}}\right] \begin{bmatrix} \alpha \\ \beta \\ \lambda^* \end{bmatrix} = \bar{P}_1 - \bar{c}$$

(c.) $p(\lambda^*)$ inside triangle $\iff$ $\alpha \geq 0,\ \beta \geq 0,\ \alpha + \beta \leq 1$

Computing Ray-Polygon Intersections

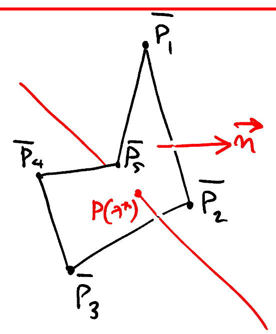
(a.) Compute $\bar{p}(\lambda^*)$ using Algorithm 1 or 2 (by picking 3 adjacent non-collinear vertices)

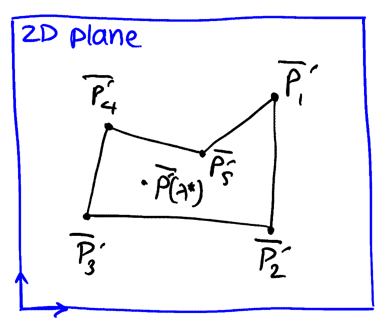


(b.) Verify that $p(\lambda^*)$ lies inside the polygon

- Convert the 3D polygon & $p(\lambda^*)$ to 2D
- Do the verification in 2D (recall assignment #1)

Computing Ray-Poly Intersections: Step a





- Suppose $\vec{n}$ is not along z-axis
- Project all vertices and $p^*(\lambda)$ onto xy-plane:

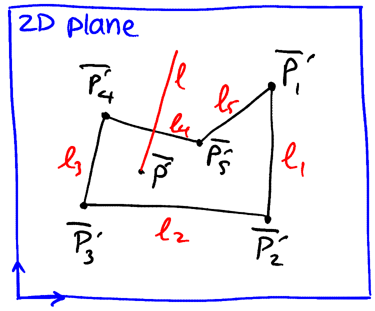
$$\bar{p}_i = \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \bar{p}_i'$$

if $\vec{n}$ is along z axis, project onto xz-plane

Computing Ray-Poly Intersections: Step b

Key theorem:

If $\bar{p}'$ inside, every 2D half-line starting at $\bar{p}'$ must intersect the polygon's boundary an odd # of times



Verification algorithm:

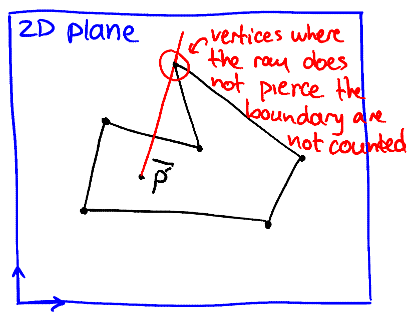
e.g. $\bar{q}' = \frac{1}{2}(\bar{P}_1' + \bar{P}_2')$

1. pick any non-vertex point on boundary
2. define lines l through $\bar{p}', \bar{q}'$ and l_i through $\bar{P}_i', \bar{P}_{i+1}'$
3. intersect l with each l_i
4. count intersections that are (a) on same side of $\bar{p}'$, and (b) on polygon boundary

Computing Ray-Poly Intersections: Step b

Key theorem:

If $\bar{p}'$ inside, every 2D half-line starting at $\bar{p}'$ must intersect the polygon's boundary an odd # of times



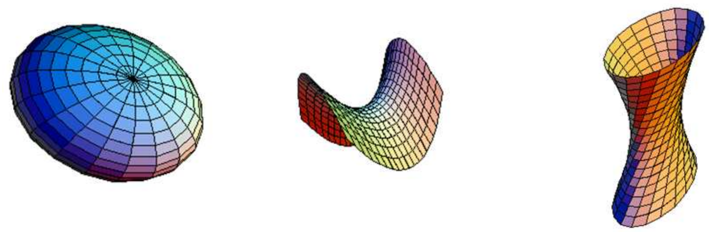
Verification algorithm:

e.g. $\bar{q}' = \frac{1}{2}(\bar{P}_1' + \bar{P}_2')$

1. pick any non-vertex point on boundary
2. define lines l through $\bar{p}', \bar{q}'$ and l_i through $\bar{P}_i', \bar{P}_{i+1}'$
3. intersect l with each l_i

4. count intersections ← counting is a bit more involved if line l intersects a polygon vertex

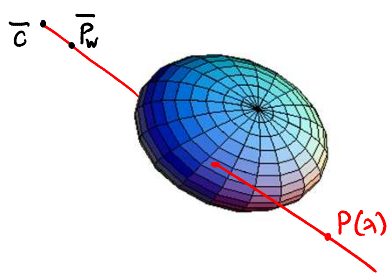
Computing Ray-Quadric Intersections



General implicit equation

$$[x\ y\ z\ 1]\underbrace{\begin{bmatrix} A & D & E & G \\ D & B & F & H \\ E & F & C & I \\ G & H & I & J \end{bmatrix}}_{\text{defined up to a scale factor}}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = 0$$

Computing Ray-Quadric Intersections



$$P(\lambda) = \bar{c} + \lambda(\bar{P}_w - \bar{c})$$

Must solve the equation

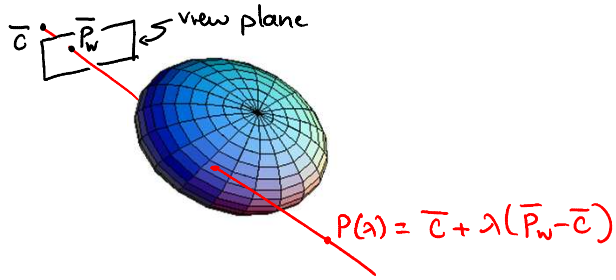
$$P(\lambda)^T \begin{bmatrix} A & D & E & G \\ D & B & F & H \\ E & F & C & I \\ G & H & I & J \end{bmatrix} P(\lambda) = 0$$

the only unknown is λ

expressed in homogeneous 3D coords

for a given quadric, this matrix is known

Computing Ray-Quadric Intersections: 3 Cases



$\Delta > 0$
$\Rightarrow$ 2 "hits"

$\Delta < 0$
$\Rightarrow$ 0 "hits"

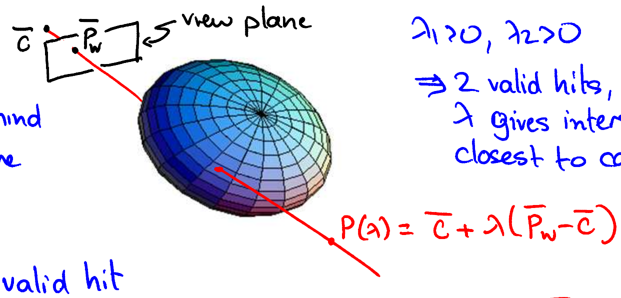
$\Delta = 0$
$\Rightarrow$ 1 "hit"

after expanding, we have a quadratic equation in terms of λ:

$$\alpha \lambda^2 + \beta \lambda + \gamma = 0$$

solution is $\lambda = \frac{-\beta \pm \sqrt{\Delta}}{2\alpha}$, $\Delta = \beta^2 - 4\alpha\gamma$

Ray-Quadric Intersections: Sub-cases for Δ>0



$\lambda_1, \lambda_2 < 0$
$\Rightarrow$ hits are behind the view plane

$\lambda_1 > 0, \lambda_2 > 0$
$\Rightarrow$ 2 valid hits, smallest λ gives intersection closest to camera

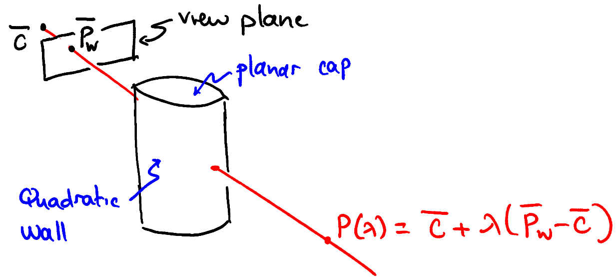
$\lambda_1 > 0, \lambda_2 < 0$
$\Rightarrow P(\lambda_1)$ is a valid hit

after expanding, we have a quadratic equation in terms of λ:

$$\alpha \lambda^2 + \beta \lambda + \gamma = 0$$

solution is $\lambda = \frac{-\beta \pm \sqrt{\Delta}}{2\alpha}$, $\Delta = \beta^2 - 4\alpha\gamma$

Intersecting Rays & Composite Objects



- When an object is bounded by multiple parametric surfaces we must test for intersection with each of the components
- Example: Cylinder = "Quadratic wall" + 2 planar "caps"

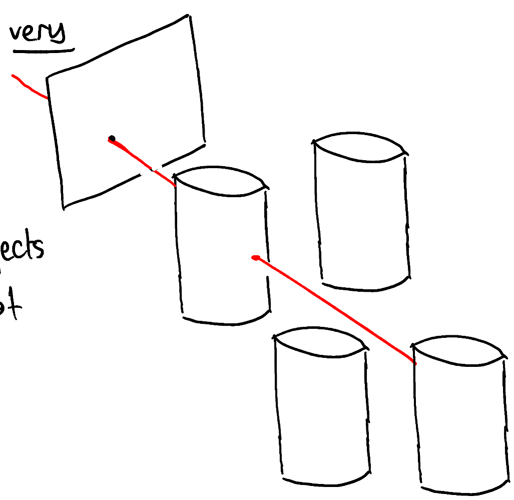
  Cone = "Quadratic wall" + 1 planar base

$\Rightarrow$ See Leonid Sigal's slides for more details

Ray Intersection: Efficiency Considerations

Intersection tests can be very expensive!

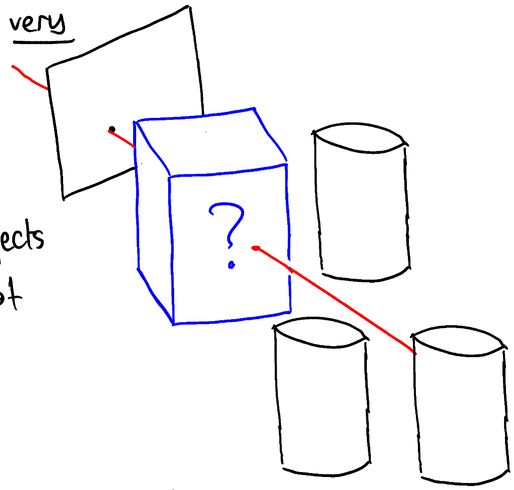
$\Rightarrow$ Use data structures to avoid testing intersection with objects that clearly do not intersect



Ray Intersection: Efficiency Considerations

Intersection tests can be <u>very</u> expensive!

⇒ Use data structures to avoid testing intersection with objects that clearly do not intersect



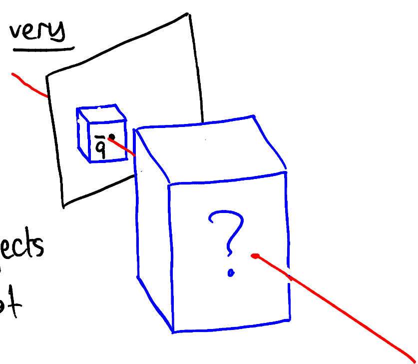
Examples:

- test intersection with object's bounding volume first, test ray-object intersection only if ray intersects volume
- apply this idea hierarchically, for part-based objects

Ray Intersection: Efficiency Considerations

Intersection tests can be <u>very</u> expensive!

⇒ Use data structures to avoid testing intersection with objects that clearly do not intersect



Examples:

- Image-space intersections: instead of intersecting ray & bounding volume, project volume & check whether pixel $\bar{q}$ falls inside that projection

Topic 12:

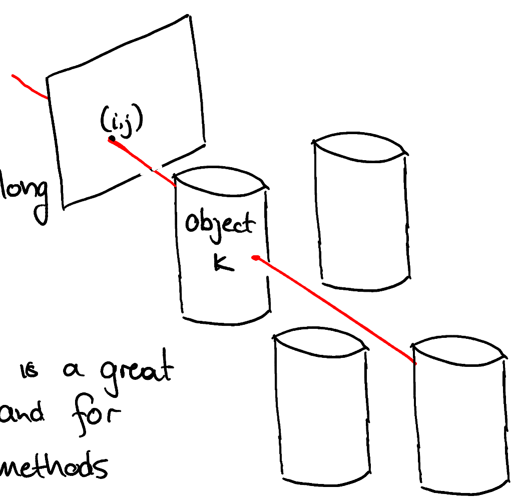
Basic Ray Tracing

- Introduction to ray tracing
- Computing rays
- **Computing intersections**
  - ray-triangle
  - ray-polygon
  - ray-quadric
  - **the scene signature**
- Computing normals
- Evaluating shading model
- Spawning rays
- Incorporating transmission
  - refraction
  - ray-spawning & refraction

The Scene Signature

Definition:

An image S where $S(i,j) = k$ if object k is the first object along ray through (i,j)

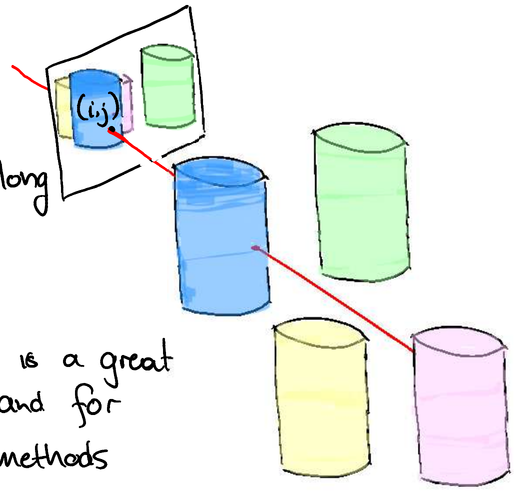


* The scene signature is a great tool for debugging and for testing intersection methods

The Scene Signature

Definition:

An image S where
$S(i,j) = k$ if object k
is the first object along
ray through (i,j)

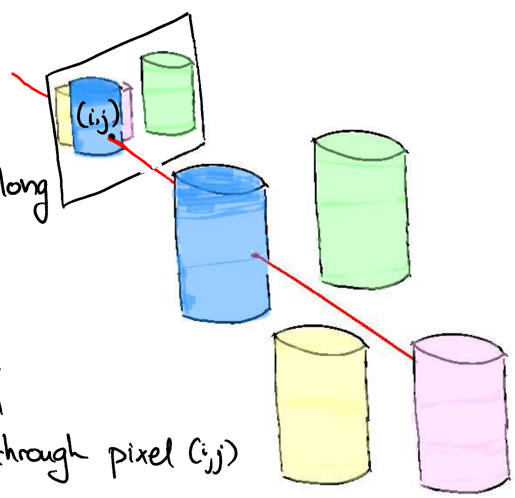


* The scene signature is a great tool for debugging and for testing intersection methods

Computing the Scene Signature

Definition:

An image S where
$S(i,j) = k$ if object k
is the first object along
ray through (i,j)



Algorithm pseudocode:

```
for i = 0 to Nrows-1
    for j = 0 to Ncols-1
        construct ray through pixel (i,j)
        λi,j = ∞
        for k = 0 to Nobjects
            λ* = closest intersection of ray with object k
            if λ* > 0 and λ* < λi,j , set λi,j = λ*, S(i,j) = k
```

Computational Issues in Basic Ray Tracing

Basic loop:

for each pixel $\bar{q}$

① cast ray r through $\bar{q}$

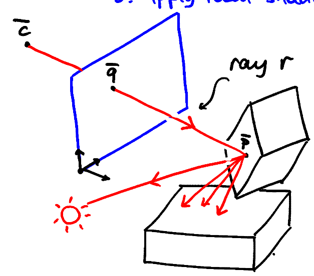
② find 1st intersection of $\bar{q}$ with scene (i.e. point $\bar{p}$)

③ estimate amount of light reaching $\bar{p}$

④ estimate amount of light travelling from $\bar{p}$ to $\bar{q}$ along ray r

a. Compute surface normal at $\bar{p}$

b. Apply local shading model



Topic 12:

Basic Ray Tracing

- Introduction to ray tracing
- Computing rays
- Computing intersections
  - ray-triangle
  - ray-polygon
  - ray-quadric
  - the scene signature
- Computing normals
- Evaluating shading model
- Spawning rays
- Incorporating transmission
  - refraction
  - ray-spawning & refraction

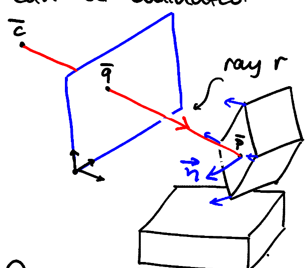
Computing the Normal at a Hit Point

Option #1:

- Smoothly interpolate normal from vertices or adjacent faces (eg. using the linear interpolation technique covered with Phong + scan conversion)

Option #2:

- For parametric shapes, normal can be evaluated directly at $\bar{p}$



Implicit form

$$\vec{n}(\bar{p}) = \frac{\nabla f(\bar{p})}{\|\nabla f(\bar{p})\|}$$

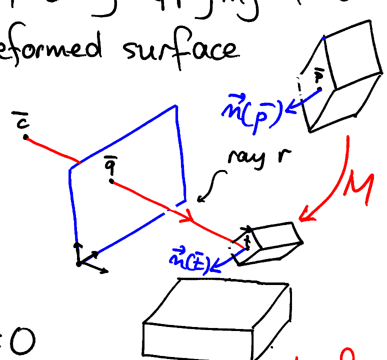
explicit form

$$\vec{n}(p) = \text{unit vector along } \frac{\partial}{\partial\alpha} S(\alpha,\beta) \times \frac{\partial}{\partial\beta} S(\alpha,\beta)$$

Computing the Normal at a Hit Point

Option #3 (affinely-deformed shapes)

- Let $f(\bar{p}) = 0$ be an implicit surface
- Let M be a 4x4 affine transformation matrix
- Suppose we deform the surface by applying M to it
- Point $\bar{t}$ will be on the deformed surface if there is a $\bar{p}$ such that $\bar{t} = M\bar{p}$ ← expressed in homogeneous coords



$\Leftrightarrow$ $\bar{p} = M^{-1}\bar{t}$

$\Leftrightarrow$ implicit eq is

$$F(\bar{t}) = f(M^{-1}\bar{p}) = 0$$

$\Leftrightarrow$ $\vec{n}(\bar{t}) = (M^{-1})^T \vec{n}(\bar{p}) \,/\, \|(M^{-1})^T \vec{n}(\bar{p})\|$ ← see notes for complete proof

Topic 12:

Basic Ray Tracing

- Introduction to ray tracing
- Computing rays
- Computing intersections
    - ray-triangle
    - ray-polygon
    - ray-quadric
    - the scene signature
- Computing normals
- **Evaluating shading model**
- Spawning rays
- Incorporating transmission
    - refraction
    - ray-spawning & refraction

Evaluating the Shading Model

Use a two-component model

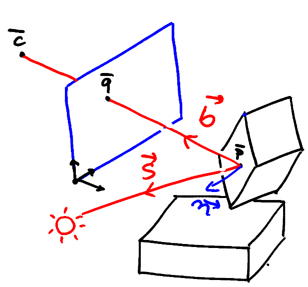
$$I(\bar{q}) = L(\vec{b},\vec{n},\vec{s}) + G(\bar{p})\cdot r_s$$

specular reflection coeff

intensity at pixel $\bar{q}$

local shading model at $\bar{p}$

global shading component at $\bar{p}$



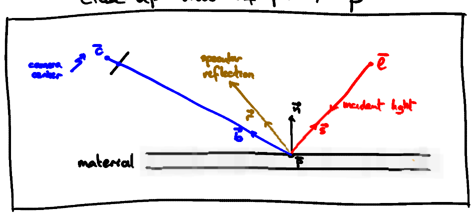
Evaluating the Shading Model

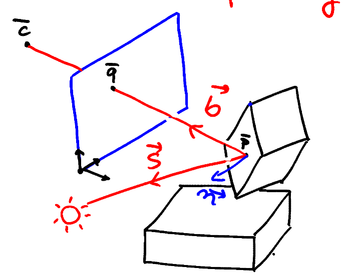
Use a two-component model

$$I(\bar{q}) = \underbrace{L(\vec{b}, \vec{n}, \vec{s})}_{\text{Phong model}} + \underbrace{G(\bar{p})}_{\text{computed after ray spawning}} \cdot r_s$$

$$\boxed{\underbrace{r_a I_\alpha}_{\text{ambient}} + \underbrace{r_d I_d \max(0, \vec{n} \cdot \vec{s})}_{\text{diffuse}} + \underbrace{r_s I_s \max(0, \vec{r} \cdot \vec{b})^\alpha}_{\text{specular}}}$$

close-up "view" of point $\bar{p}$





Evaluating the Shading Model: Using Textures

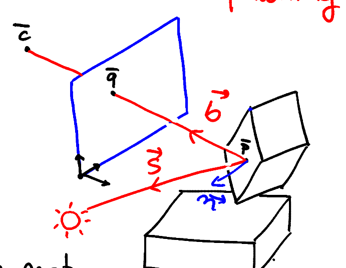
Use a two-component model

$$I(\bar{q}) = \underbrace{L(\vec{b}, \vec{n}, \vec{s})}_{\text{Phong model}} + \underbrace{G(\bar{p})}_{\text{computed after ray spawning}} \cdot r_s$$

$$\boxed{\underbrace{r_a I_\alpha}_{\text{ambient}} + \underbrace{r_d I_d \max(0, \vec{n} \cdot \vec{s})}_{\text{diffuse}} + \underbrace{r_s I_s \max(0, \vec{r} \cdot \vec{b})^\alpha}_{\text{specular}}}$$

Texture can be used to modulate r_a and r_d:

- need to compute $\bar{p}$'s texture coordinates
- unlike scan-conversion, we compute $\bar{p}$'s texture coordinates by linear interpolation on the polygon plane, not in image space ($\Leftarrow$ no distortion artifacts)



Computational Issues in Basic Ray Tracing

Basic loop:

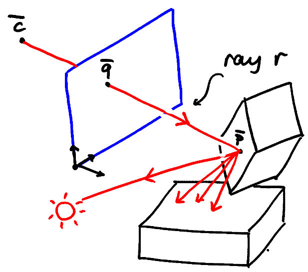
for each pixel $\bar{q}$

① cast ray r through $\bar{q}$

② find 1st intersection of $\bar{q}$ with scene (i.e. point $\bar{p}$)

③ estimate amount of light reaching $\bar{p}$

④ estimate amount of light travelling from $\bar{p}$ to $\bar{q}$ along ray r



Computational Issues in Basic Ray Tracing

Basic loop:

for each pixel $\bar{q}$

① cast ray r through $\bar{q}$

② find 1st intersection of $\bar{q}$ with scene (i.e. point $\bar{p}$)

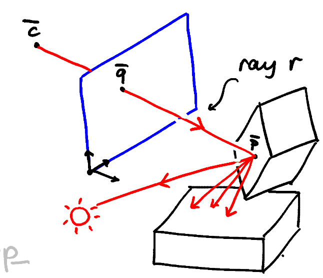
③ estimate amount of light reaching $\bar{p}$

a. "spawn" rays $r_1, r_2, \ldots, r_k$ from $\bar{p}$ in various directions

b. if ray r_i hits a light source, estimate light travelling along r_i and stop

c. else apply loop recursively to ray r_i

④ estimate amount of light travelling from $\bar{p}$ to $\bar{q}$ along ray r



Topic 12:

Basic Ray Tracing

- Introduction to ray tracing
- Computing rays
- Computing intersections
  - ray-triangle
  - ray-polygon
  - ray-quadric
  - the scene signature
- Computing normals
- Evaluating shading model
- **Spawning rays**
- Incorporating transmission
  - refraction
  - ray-spawning & refraction

Whitted Ray Tracing

Basic idea:

- Spawn only <u>one</u> ray
- Ray is along ideal specular direction

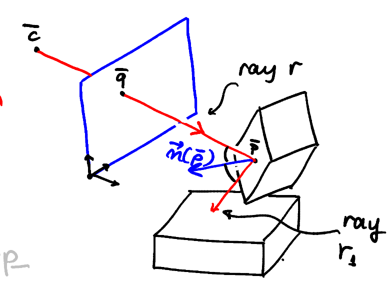
---

③ estimate amount of light reaching $\bar{p}$

a. "spawn" rays r_1, ~~$r_2, \ldots, r_k$~~ from $\bar{p}$ in ~~various~~ specular direction ~~directions~~ only

b. if ray r_i hits a light source, estimate light travelling along r_i and <u>stop</u>

c. else apply loop recursively to ray r_i



Whitted Ray Tracing

Motivation:

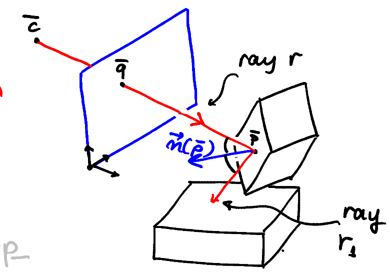
- Computationally efficient (1 spawned ray / bounce)
- Models the most important light path (in terms of "light energy" transfered from r_1 to r)

③ estimate amount of light reaching $\bar{p}$

a. "spawn" rays ~~$r_1, r_2, \ldots, r_k$~~ r_1 from $\bar{p}$ in ~~various~~ specular direction only ~~directions~~

b. if ray r_i hits a light source, estimate light travelling along r_i and stop

c. else apply loop recursively to ray r_i



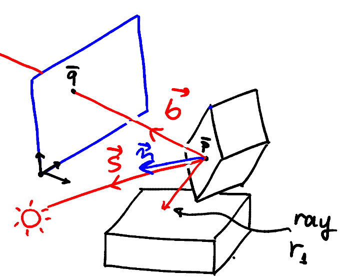
Whitted Ray Tracing: An Example

Use a two-component model

$$I(\bar{q}) = \underbrace{L(\vec{b}, \vec{n}, \vec{s})}_{\text{Phong model}} + \underbrace{G(\bar{p})}_{\text{Global specular term}} \cdot r_s$$

Phong model: $\underbrace{r_a I_a}_{\text{ambient}} + \underbrace{r_d I_d \max(0, \vec{n} \cdot \vec{s})}_{\text{diffuse}} + \underbrace{r_s I_s \max(0, \vec{r} \cdot \vec{b})^{\alpha}}_{\text{specular}}$

Global specular term



Whitted Ray Tracing: An Example

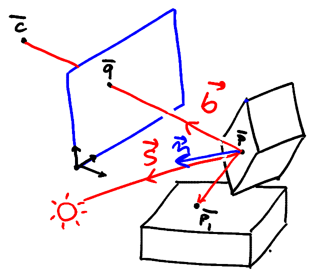
Use a two-component model

$$I(\bar{q}) = L(\vec{b},\vec{m},\vec{s}) + G(\bar{p}) \cdot r_s$$

Phong model

$$\boxed{\underbrace{r_a I_a}_{\text{ambient}} + \underbrace{r_d I_d \max(0,\vec{n}\cdot\vec{s})}_{\text{diffuse}} + \underbrace{r_s I_s \max(0,\vec{r}\cdot\vec{b})^{\alpha}}_{\text{specular}}}$$

Global specular term



Whitted Ray Tracing: An Example

Use a two-component model

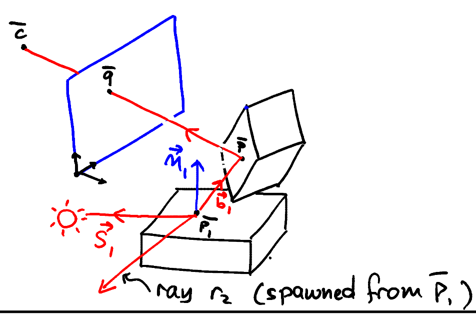
$$I(\bar{q}) = L(\vec{b},\vec{m},\vec{s}) + G(\bar{p}) \cdot r_s$$

Phong model

$$\boxed{\underbrace{r_a I_a}_{\text{ambient}} + \underbrace{r_d I_d \max(0,\vec{n}\cdot\vec{s})}_{\text{diffuse}} + \underbrace{r_s I_s \max(0,\vec{r}\cdot\vec{b})^{\alpha}}_{\text{specular}}}$$

Global specular term

$$\boxed{L(\vec{b}_1,\vec{m}_1,\vec{s}_1) + G(\bar{p}_1) r_s}$$



Whitted Ray Tracing: An Example

Use a two-component model

$$I(\bar q) = \underbrace{L(\vec b, \vec m, \vec s)}_{\text{Phong model}} + \underbrace{G(\bar p)}_{\text{Global specular term}} \cdot r_s$$

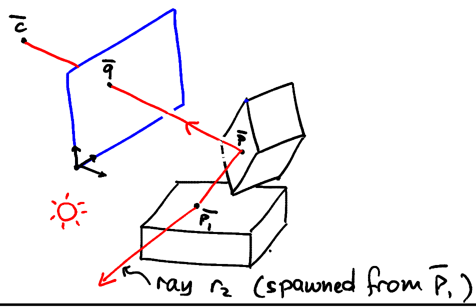
Phong model

$$\underbrace{r_a I_a}_{\text{ambient}} + \underbrace{r_d I_d \max(0, \vec n \cdot \vec s)}_{\text{diffuse}} + \underbrace{r_s I_s \max(0, \vec r \cdot \vec b)^\alpha}_{\text{specular}}$$

Global specular term

$$L(\vec b_1, \vec m_1, \vec s_1) + \underbrace{G(\bar p_1)}_{\text{computed by recursively ray tracing along } r_2} r_s$$

computed by recursively ray tracing along r_2



Simulating Shadows

Use a two-component model

$$I(\bar q) = \underbrace{L(\vec b, \vec m, \vec s)}_{\text{Phong model}} + \underbrace{G(\bar p)}_{\text{Global specular term}} \cdot r_s$$

Phong model

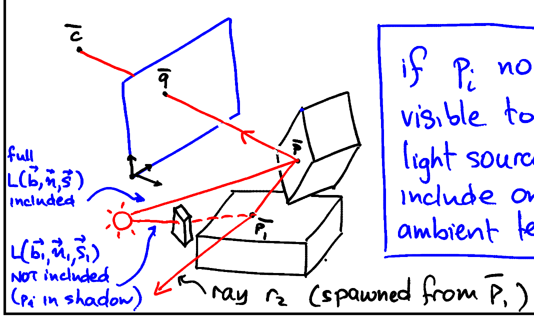
$$\underbrace{r_a I_a}_{\text{ambient}} + \underbrace{r_d I_d \max(0, \vec n \cdot \vec s)}_{\text{diffuse}} + \underbrace{r_s I_s \max(0, \vec r \cdot \vec b)^\alpha}_{\text{specular}}$$

Global specular term

if p_i not visible to light source, include only ambient term

$$\cancel{L(\vec b_1, \vec m_1, \vec s_1)} \quad \underbrace{G(\bar p_1)}_{\text{computed by recursively ray tracing along } r_2} r_s$$

computed by recursively ray tracing along r_2



Simulating Shadows

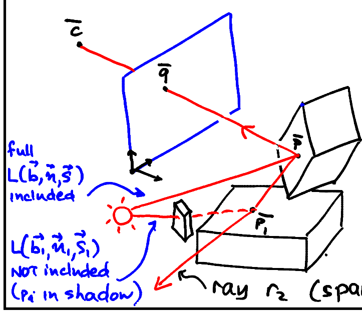
Use a two-component model

$$I(\bar{q}) = L(\vec{b},\vec{m},\vec{s}) + G(\bar{p})\cdot r_s$$

Phong model

$$\boxed{\underbrace{r_a I_a}_{\text{ambient}} + \underbrace{r_d I_d \max(0,\vec{n}\cdot\vec{s})}_{\text{diffuse}} + \underbrace{r_s I_s \max(0,\vec{r}\cdot\vec{b})^\alpha}_{\text{specular}}}$$

Global specular term



ray r_2 (spawned from $\bar{p}_1$)

if p_i not visible to light source, include only ambient term $r_a \cdot I_a$

$G(\bar{p}_1) r_s$ computed by recursively ray tracing along r_2

Simulating Shadows

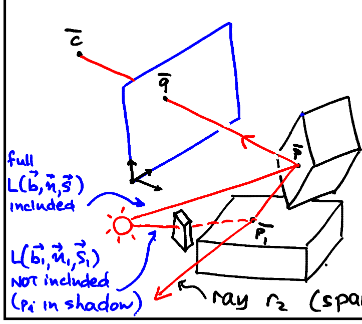
Use a two-component model

$$I(\bar{q}) = L(\vec{b},\vec{m},\vec{s}) + G(\bar{p})\cdot r_s$$

Phong model

$$\boxed{\underbrace{r_a I_a}_{\text{ambient}} + \underbrace{r_d I_d \max(0,\vec{n}\cdot\vec{s})}_{\text{diffuse}} + \underbrace{r_s I_s \max(0,\vec{r}\cdot\vec{b})^\alpha}_{\text{specular}}}$$

Global specular term



ray r_2 (spawned from $\bar{p}_1$)

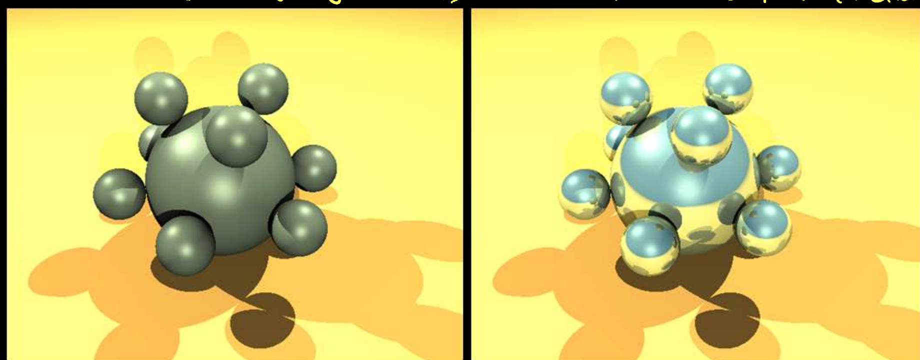
if p_i not visible to light source, include only ambient term

$r_a \cdot I_a + G(\bar{p}_1) r_s$

determined by casting a ray from p_i to the (point) light source and testing for object hits between p_i & that source

Non-recursive vs. Recursive Ray Tracing

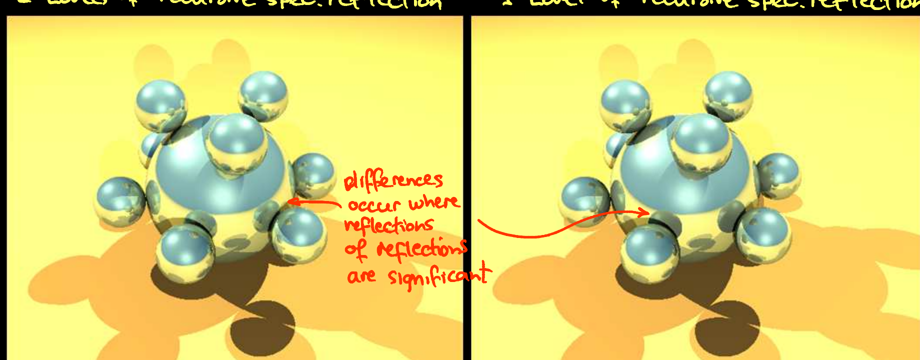
No recursion (local lighting + shadows) 1 Level of recursive spec. reflection



https://agora.cs.uiuc.edu/download/attachments/10454060/RayTracing_suppl.ppt?version=1

Ray tracing in the movies

2 Levels of recursive spec. reflection 1 Level of recursive spec. reflection

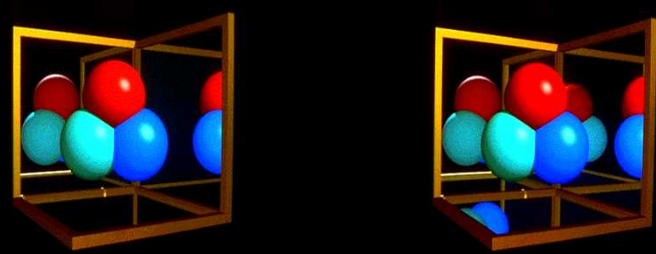


https://agora.cs.uiuc.edu/download/attachments/10454060/RayTracing_suppl.ppt?version=1

Ray tracing in the movies

1 recursive level

2 recursive levels

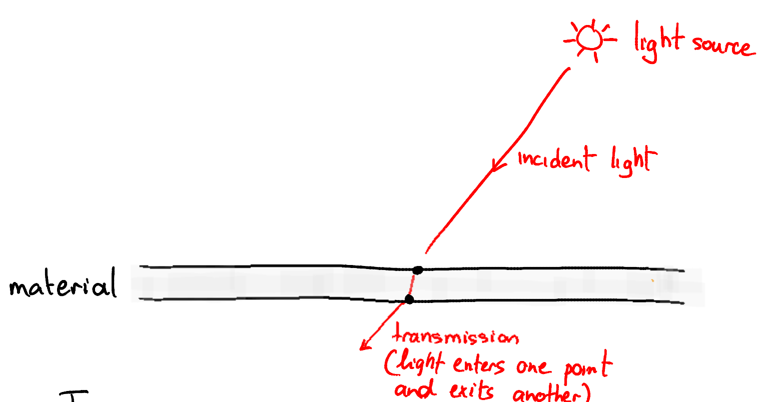


Topic 12:

Basic Ray Tracing

- Introduction to ray tracing
- Computing rays
- Computing intersections
  - ray-triangle
  - ray-polygon
  - ray-quadric
  - the scene signature
- Computing normals
- Evaluating shading model
- Spawning rays
- Incorporating transmission
  - refraction
  - ray-spawning & refraction

Modeling Reflection: Transmission

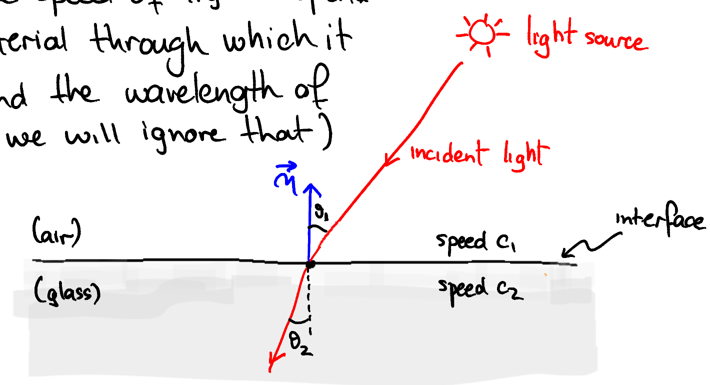


Transmission:

- Caused by materials that are not perfectly opaque
- Examples include glass, water and translucent materials such as skin

Physics of Refraction

Physics: the speed of light depends on the material through which it travels (and the wavelength of light, but we will ignore that)



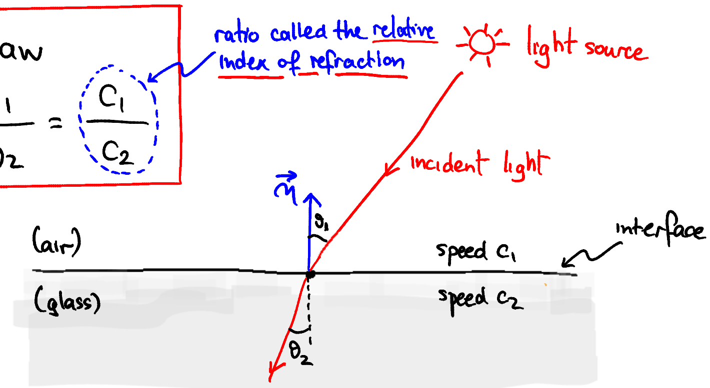
Refraction (bending of rays) occurs when light crosses an interface between two media with different speeds of light

Physics of Refraction

Snell's law

$$\frac{\sin\theta_1}{\sin\theta_2} = \frac{c_1}{c_2}$$

ratio called the relative index of refraction

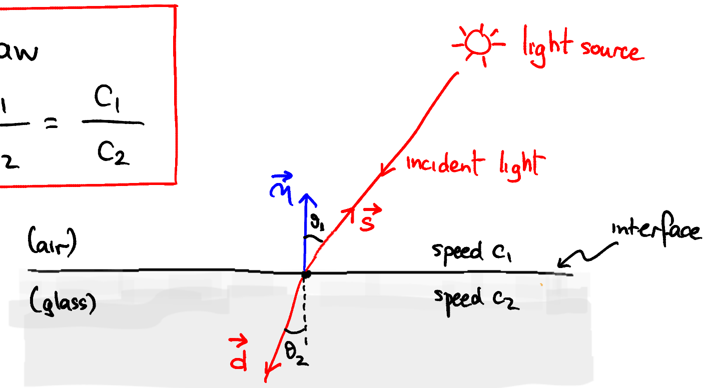


Refraction (bending of rays) occurs when light crosses an interface between two media with different speeds of light

Geometry of Refraction: Transmission Vector

Snell's law

$$\frac{\sin\vartheta_1}{\sin\vartheta_2} = \frac{c_1}{c_2}$$



① Incident ray, outgoing ray & normal always lie on the same plane ⇒

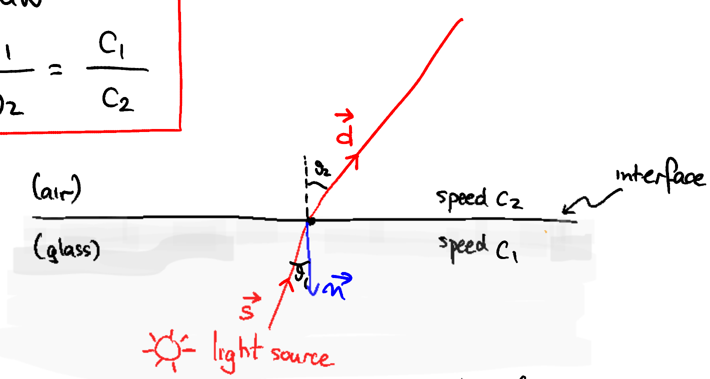
$$\vec{d} \text{ along } -\frac{c_2}{c_1}\vec{s} + \left[\frac{c_2}{c_1}\cos\vartheta_1 - \cos\vartheta_2\right]\vec{n}$$

exercise: prove this

Geometry of Refraction: Path Reversibility

Snell's law

$$\frac{\sin\vartheta_1}{\sin\vartheta_2} = \frac{c_1}{c_2}$$

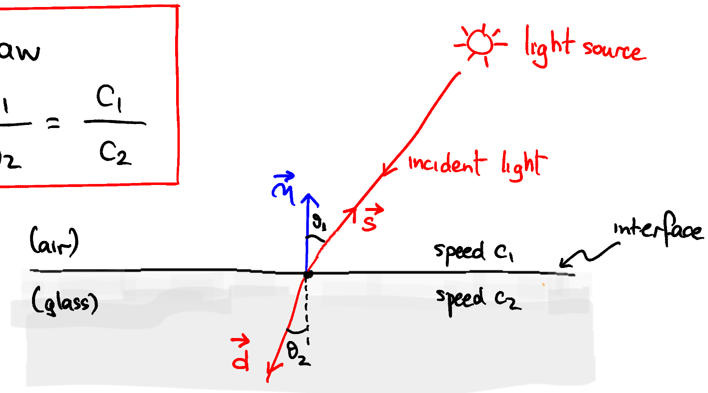


② Light paths are always reversible (i.e. light is transmitted exactly the same way if its direction of travel is reversed)

Geometry of Refraction

Snell's law

$$\frac{\sin\theta_1}{\sin\theta_2} = \frac{c_1}{c_2}$$

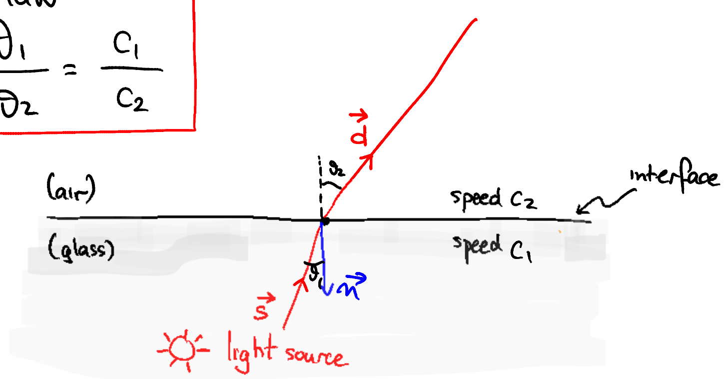


③ If $c_2 < c_1$ light bends toward the normal

Geometry of Refraction

Snell's law

$$\frac{\sin\theta_1}{\sin\theta_2} = \frac{c_1}{c_2}$$

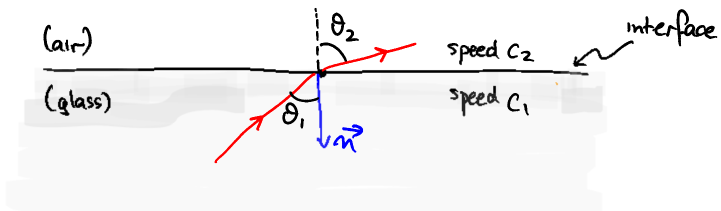


③ If $c_2 < c_1$ light bends toward the normal
If $c_2 > c_1$ light bends away from normal

Geometry of Refraction

Snell's law

$$\frac{\sin\theta_1}{\sin\theta_2} = \frac{c_1}{c_2}$$

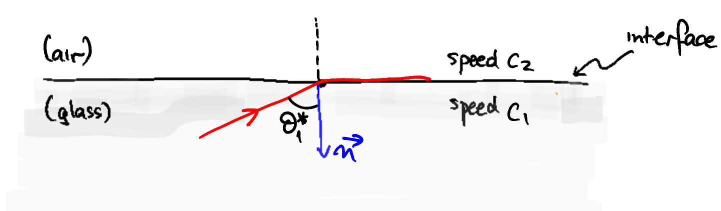


③ If $c_2 < c_1$ light bends toward the normal
If $c_2 > c_1$ light bends away from normal

Geometry of Refraction: The Critical Angle

Snell's law

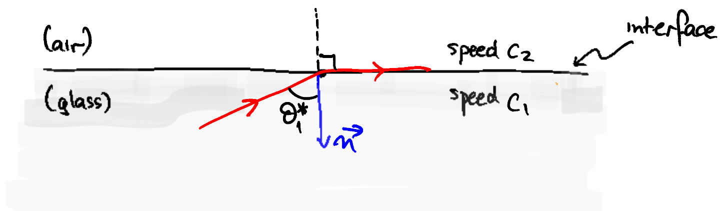
$$\frac{\sin\theta_1}{\sin\theta_2} = \frac{c_1}{c_2}$$



④ If $c_2 > c_1$ there is a critical angle above which no transmission occurs ($\Rightarrow$ have total internal reflection)

Geometry of Refraction: Total Internal Reflection

$\boxed{\text{Snell's law} \quad \dfrac{\sin\theta_1}{\sin\theta_2} = \dfrac{c_1}{c_2}}$



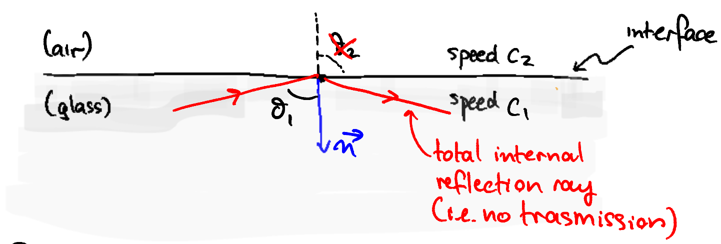
④ Deriving the critical angle: from Snell's law,

$$\cos\theta_2 = \sqrt{1-\left(\frac{c_2}{c_1}\right)^2 \sin^2\theta_1}$$

at critical angle, $\theta_2 = \frac{\pi}{2} \Rightarrow \sin\theta_1^* = \frac{c_1}{c_2}$

Geometry of Refraction: Total Internal Reflection

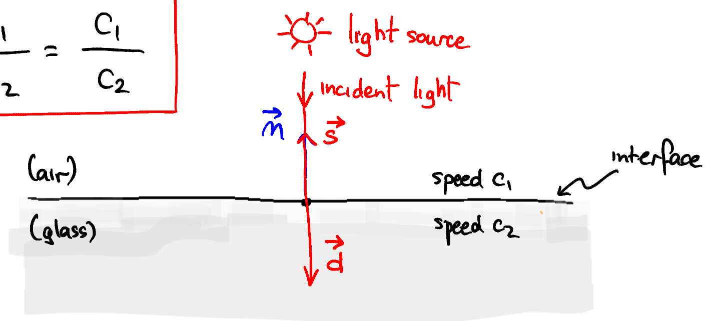
$\boxed{\text{Snell's law} \quad \dfrac{\sin\theta_1}{\sin\theta_2} = \dfrac{c_1}{c_2}}$



④ for $\theta_1 > \theta_1^*$, θ_2 is undefined

Geometry of Refraction: Normal Incidence

Snell's law
$$\frac{\sin\theta_1}{\sin\theta_2} = \frac{c_1}{c_2}$$



⑤ If $\theta_1 = 0$, no bending occurs

$\vec{d}$ along $-\frac{c_2}{c_1}\vec{s} + \left[\frac{c_2}{c_1}\cos\theta_1 - \cos\theta_2\right]\vec{n}$

Topic 12:

Basic Ray Tracing

- Introduction to ray tracing
- Computing rays
- Computing intersections
  - ray-triangle
  - ray-polygon
  - ray-quadric
  - the scene signature
- Computing normals
- Evaluating shading model
- Spawning rays
- **Incorporating transmission**
  - refraction
  - **ray-spawning & refraction**

Whitted Ray Tracing with Refraction

Basic idea:

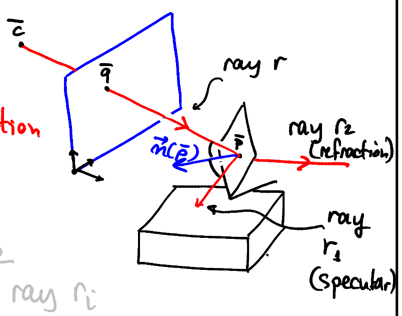
- Spawn two rays
- One ray is along ideal specular direction
- One is along refraction direction

③ estimate amount of light reaching $\bar{p}$

a. "spawn" rays r_1, r_2 ~~$r_1, r_2, \ldots, r_k$~~ from $\bar{p}$ in ~~various~~ specular direction and refraction direction ~~directions~~

b. if ray r_i hits a light source, estimate light travelling along r_i and stop

c. else apply loop recursively to ray r_i



Whitted Ray Tracing with Refraction

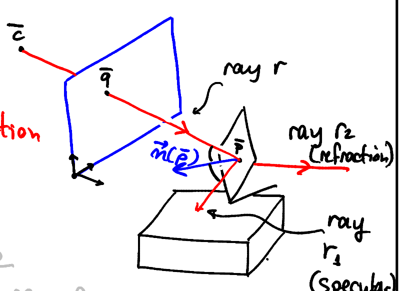
Much less efficient that specular-only ray tracing because 2^n rays are spawned after n bounces (instead of n)

③ estimate amount of light reaching $\bar{p}$

a. "spawn" rays r_1, r_2 ~~$r_1, r_2, \ldots, r_k$~~ from $\bar{p}$ in ~~various~~ specular direction and refraction direction ~~directions~~

b. if ray r_i hits a light source, estimate light travelling along r_i and stop

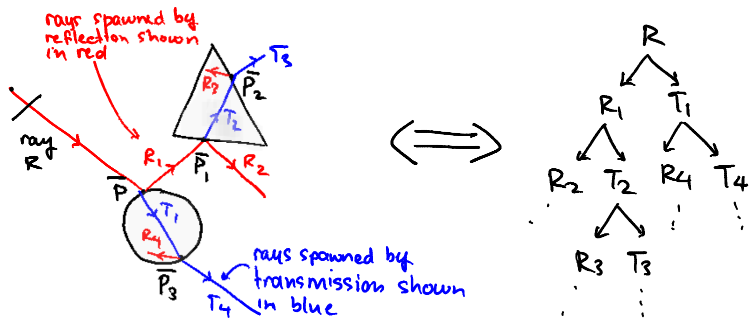
c. else apply loop recursively to ray r_i

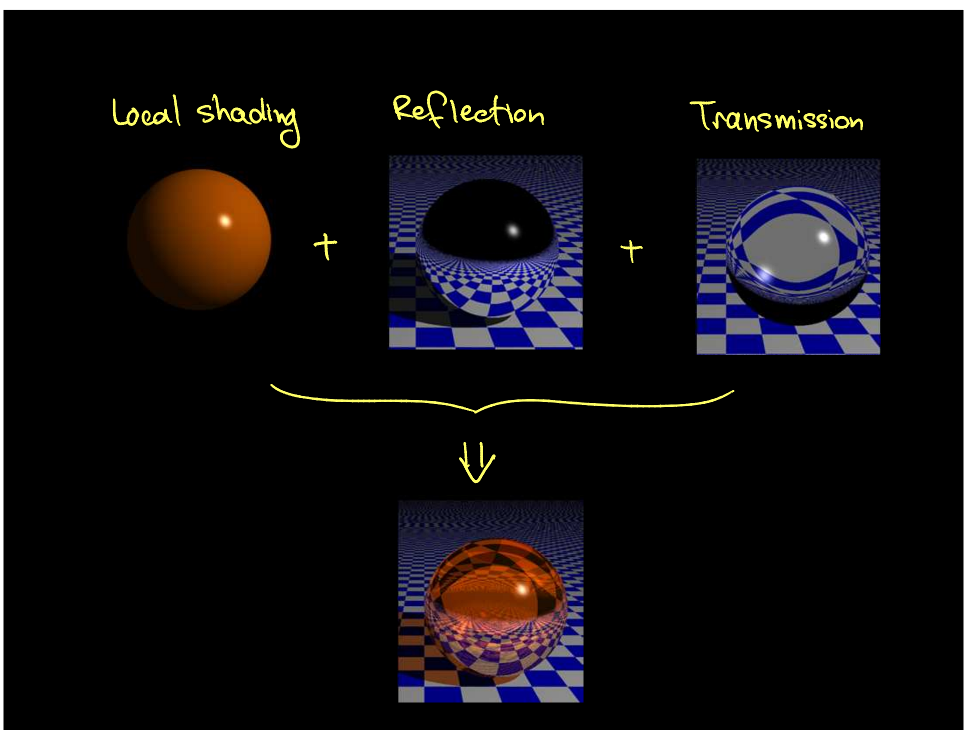


Visualizing the Spawned Rays

Much less efficient that specular-only ray tracing because 2^n rays are spawned after n bounces (instead of n)

Visualizing ray-spawning as a tree:



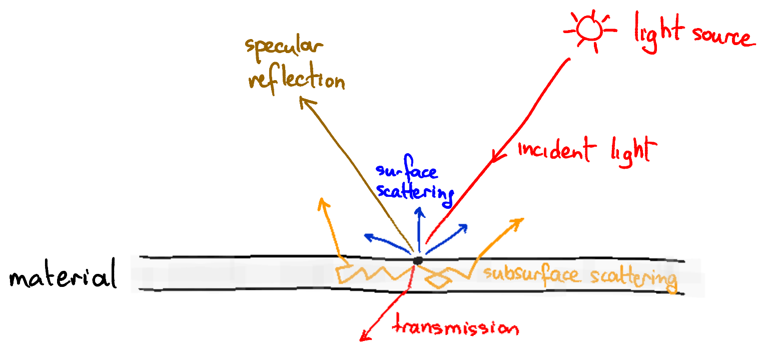


Topic 13:

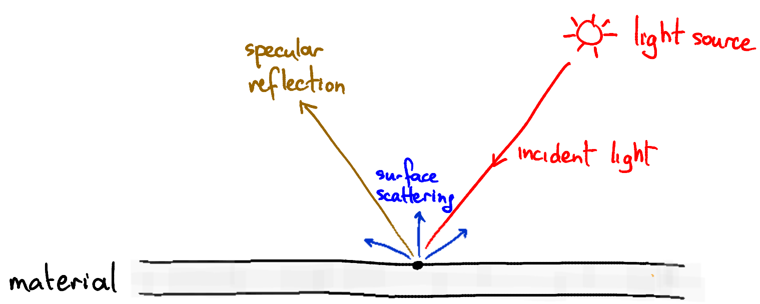
Radiometry

- The big picture
- Measuring light coming from a light source
- Measuring light falling onto a patch: Irradiance
- Measuring light leaving a patch: Radiance
- The Light Transport Cycle
- The Bidirectional Reflectance Distribution Function

The Common Modes of "Light Transport"



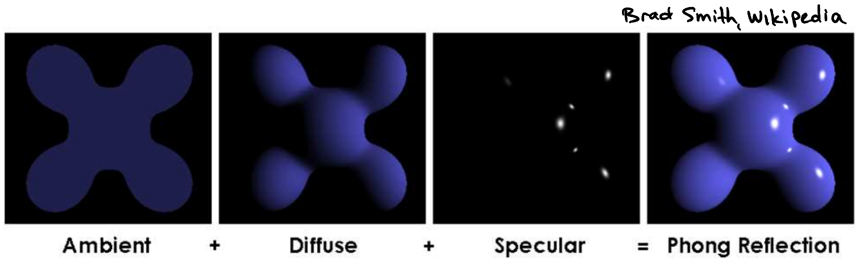
The Phong Reflectance Model



Phong model: A simple, computationally-efficient model that has 3 components:

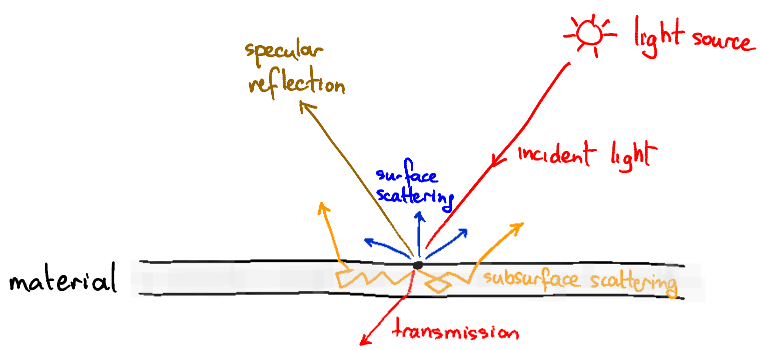
- Diffuse
- Ambient
- Specular

Phong Reflection: The General Equation

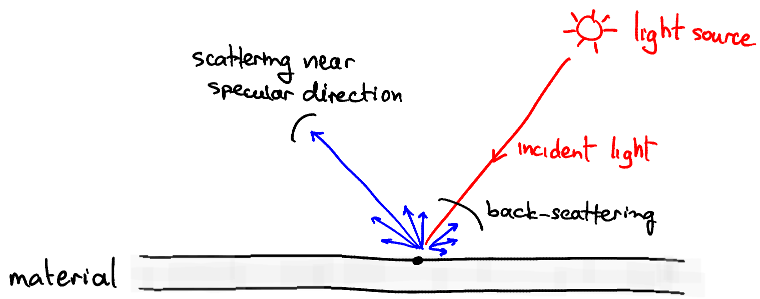


$$\underbrace{L(\vec{b},\vec{n},\vec{s})}_{\text{intensity at projection of point } \bar{p}} = \underbrace{r_a I_\alpha}_{\text{ambient}} + \underbrace{r_d I_d \max(0, \vec{n}\cdot\vec{s})}_{\text{diffuse}} + \underbrace{r_s I_s \max(0, \vec{r}\cdot\vec{b})^\alpha}_{\text{specular}}$$

The Common Modes of "Light Transport"

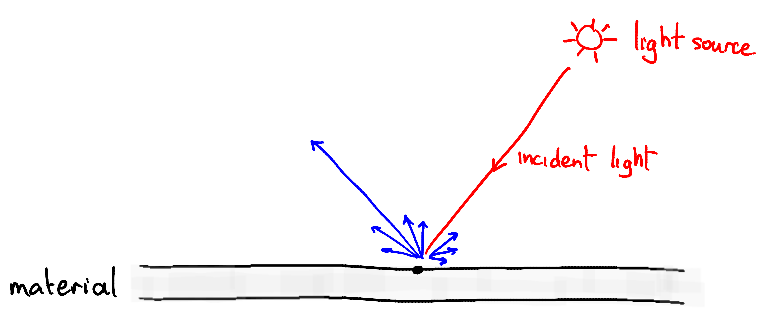


Generalizing the Phong Model



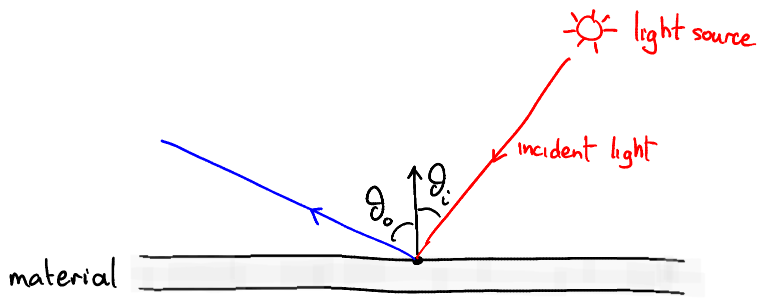
- All reflected light can be thought of as a form of scattering
- For most real materials, the Phong-based distinction into specular + diffuse reflection is a crude approximation

Generalizing the Phong Model: How?



- Seek to answer the following question: given a specific incident direction how much light is reflected along a specific outgoing direction?

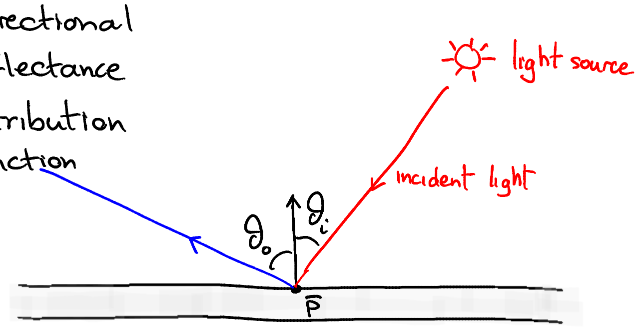
Generalizing the Phong Model: How?



- Seek to answer the following question:
  given a specific incident direction
  how much light is reflected along a
  specific outgoing direction?

The BRDF of a Surface Point (in 2D)

BRDF = Bidirectional
Reflectance
Distribution
Function



- It is a function $\rho_{\bar{p}}: [-\frac{\pi}{2}, \frac{\pi}{2}) \times [-\frac{\pi}{2}, \frac{\pi}{2}) \longrightarrow [0,1]$

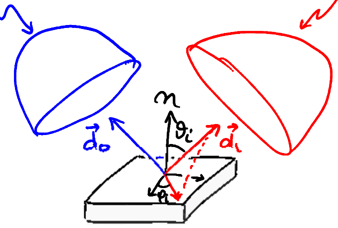
$$\rho_{\bar{p}}(\theta_i, \theta_o) = \frac{\text{emitted light in direction } \theta_o}{\text{incident light in direction } \theta_i}$$

incoming direction (θ_i) outgoing direction (θ_o)

The BRDF of a Surface Point (in 3D)

hemisphere of all possible outgoing directions

(parameterized by two angles ϑ_o, φ_o)



hemisphere of all possible incoming directions

(parameterized by 2 angles ϑ_i, φ_i)

swap the range of ϑ_o, φ_o

$$\rho : [-\pi,\pi) \times [0, \tfrac{\pi}{2}) \times [-\pi,\pi) \times [0, \tfrac{\pi}{2}) \to [0,1]$$

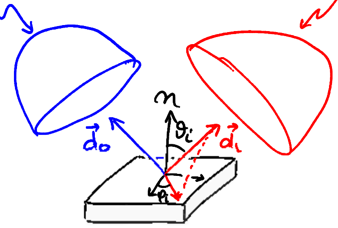
$$\rho(\vec{d}_i, \vec{d}_o) = \frac{\text{emitted light in direction } \vec{d}_o}{\text{incident light in direction } \vec{d}_i}$$

incoming direction outgoing direction

The BRDF of a Surface Point (in 3D)

hemisphere of all possible outgoing directions

(parameterized by two angles ϑ_o, φ_o)



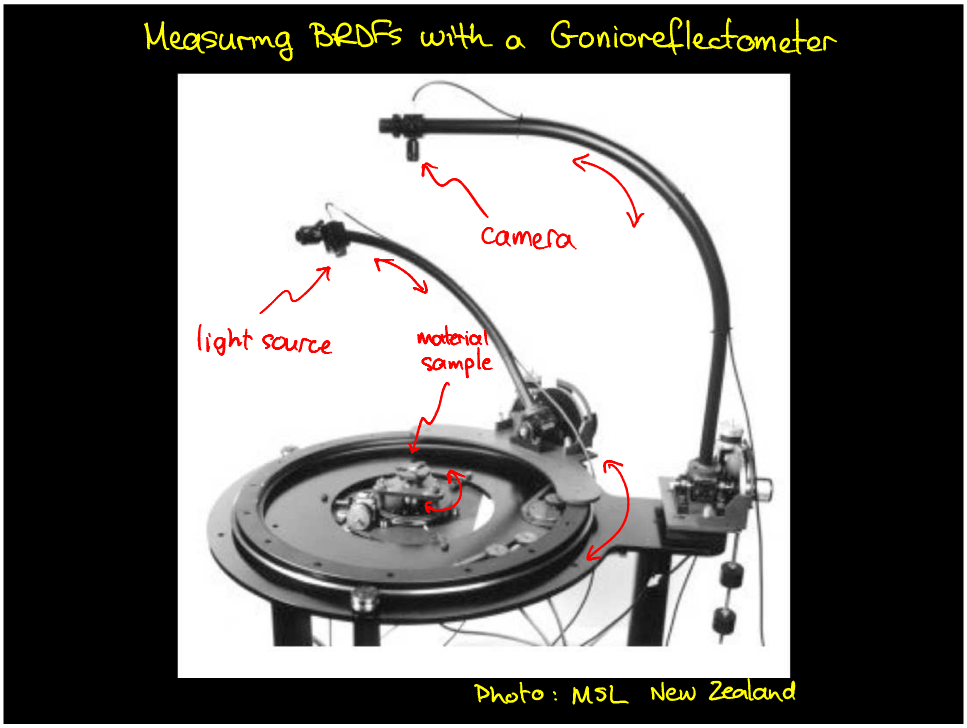
hemisphere of all possible incoming directions

(parameterized by 2 angles ϑ_i, φ_i)

$$\rho : [-\pi,\pi) \times [0, \tfrac{\pi}{2}) \times [-\pi,\pi) \times [0, \tfrac{\pi}{2}) \to [0,1]$$

$$\rho(\vartheta_i, \varphi_i, \vartheta_o, \varphi_o) = \frac{\text{emitted light in direction } \vartheta_o, \varphi_o}{\text{incident light in direction } \vartheta_i, \varphi_i}$$

incoming direction outgoing direction


Measuring BRDFs with a Gonioreflectometer
camera
light source
material
sample
Photo : MSL New Zealand


Visualizing BRDFs
The MERL BRDF database