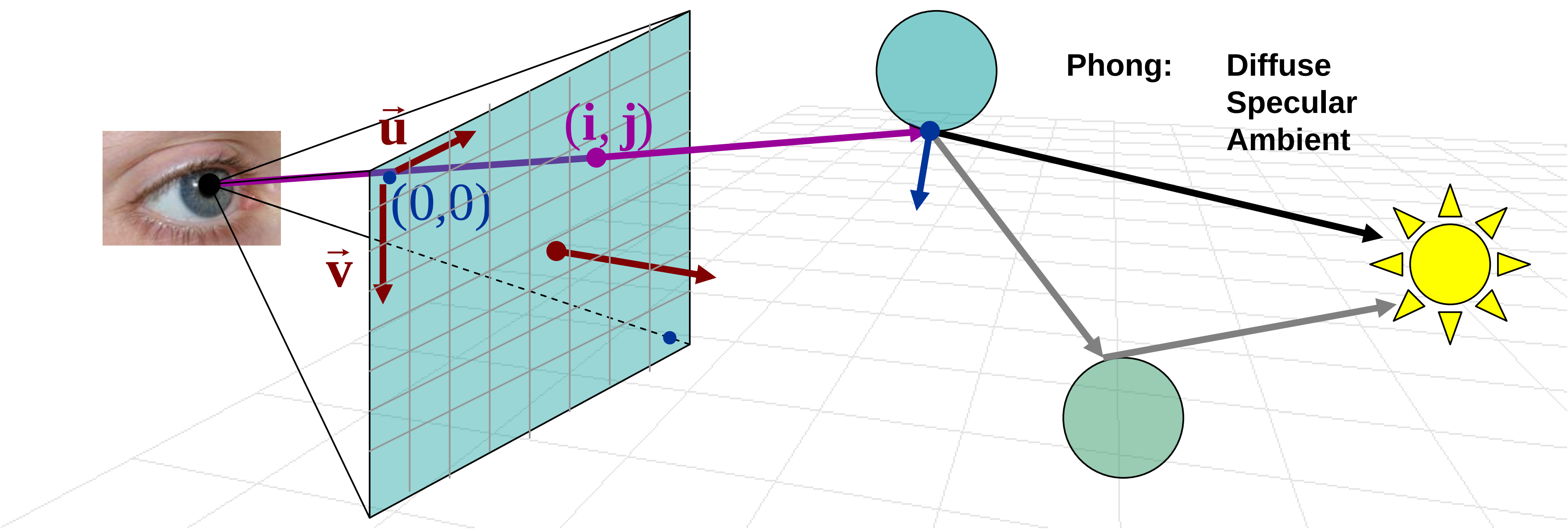
Announcements

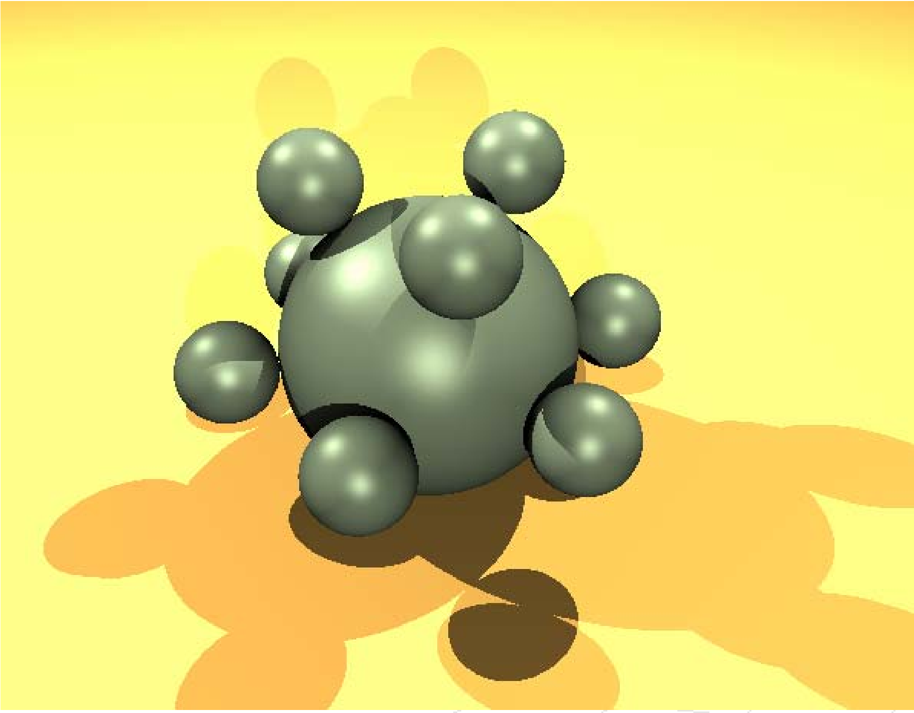
- **Assignment 2**
  - Programming **due Friday**
- **Assignment 3**
  - **Programming** will be given out **first**
  - **Theory** will be given out **later**
  - Due dates will be shifted accordingly
- **Office Hours for Alex**
  - After class **today from 11-11:45**

Ray Tracing Review

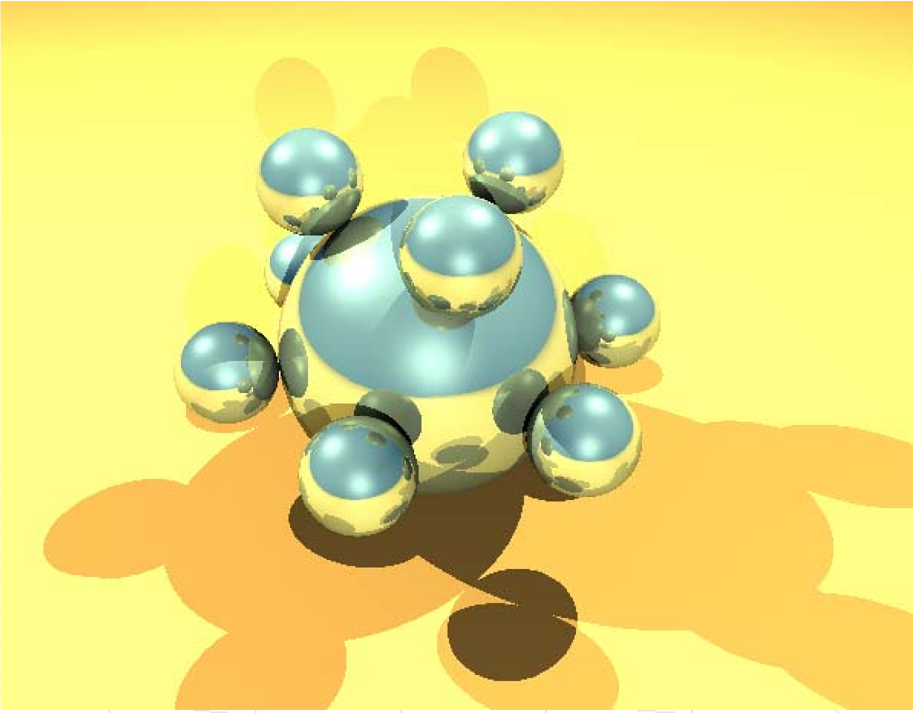
- For each pixel
  - Form a ray (a.k.a. ray casting)
  - Find intersection of this ray with objects in the scene
  - Find closest object intersection (there could be multiple object intersections for any given ray)
  - Find normal at the closest intersection point (a.k.a hit point)
  - Evaluate reflectance model at the hit point (**global** + local)



Effects of Ray Tracer Recursion



No recursive rays (local lighting)

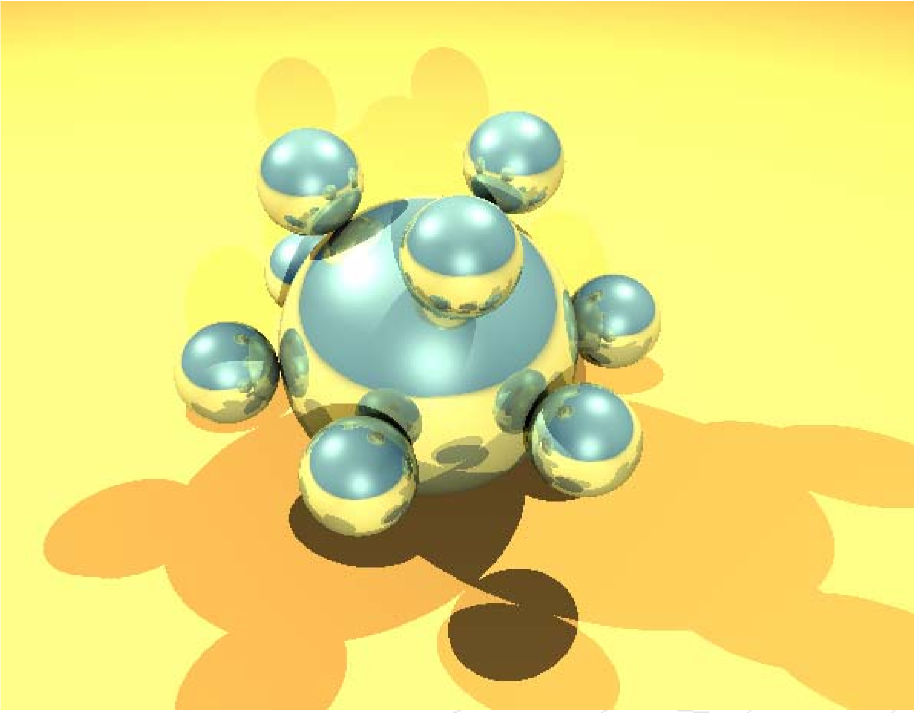


1 Level of recursive reflection

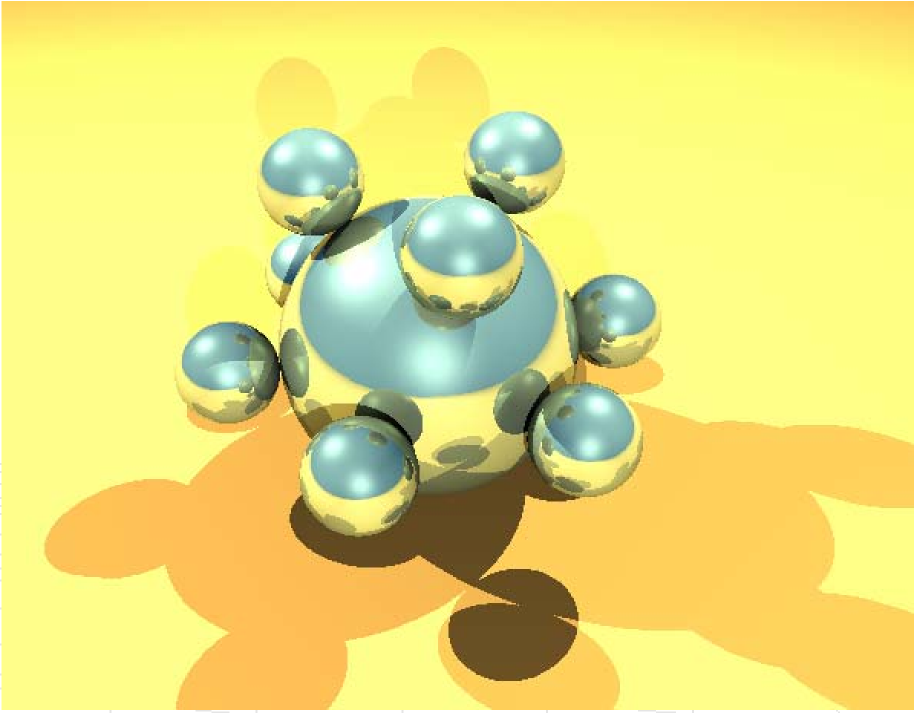
[images taken from https://agora.cs.uiuc.edu/download/attachments/10454060/RayTracing_suppl.ppt?version=1]

Effects of Ray Tracer Recursion

Recursion level of 1 or 2 is usually sufficient, unless we have mirrors That reflect in one another



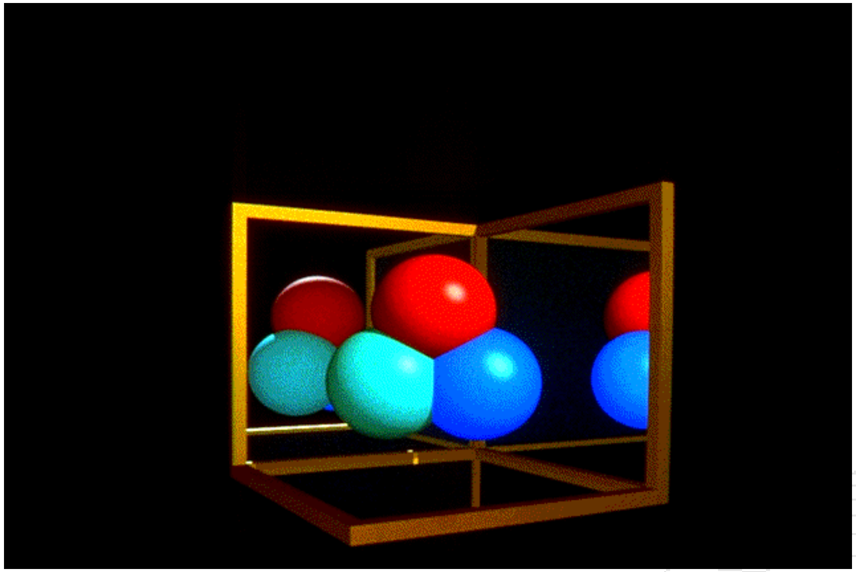
2 Levels of recursive reflection



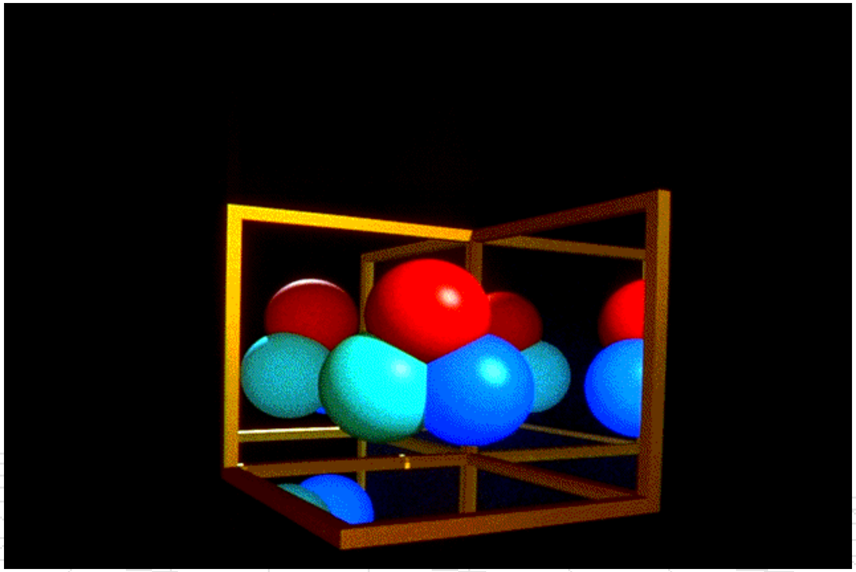
1 Level of recursive reflection

[images taken from https://agora.cs.uiuc.edu/download/attachments/10454060/RayTracing_suppl.ppt?version=1]

Effects of Ray Tracer Recursion



1 Level of recursive reflection



2 Levels of recursive reflection

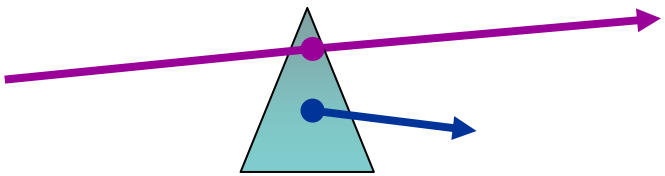
[images taken from lecture notes of Karan Singh]

Texture (last time this went by quickly)

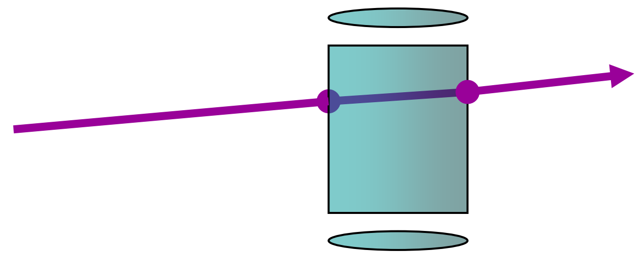
- Texture can be used to modulate diffuse and ambient reflection coefficients, as with Gouraud or Phong shading
- All we need, is a way of mapping a point on the surface (hit point) to a point in the texture space
  - e.g. given a hit point of parametric surface, we can convert the 3D point coordinates to surface parameters, and use them to get texture coordinates (as with standard texture mapping)
- Unlike with Gouraud or Phong shading models we don't need to interpolate texture coordinates over polygons
- Anti-aliasing and super-sampling we will cover later (next week)

Intersections Algorithms

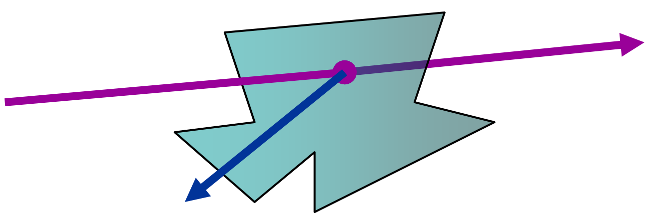
Triangles



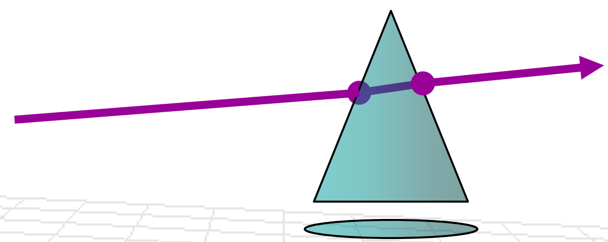
Cylinders



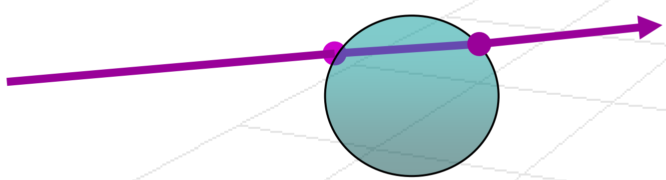
Polygonal Patches



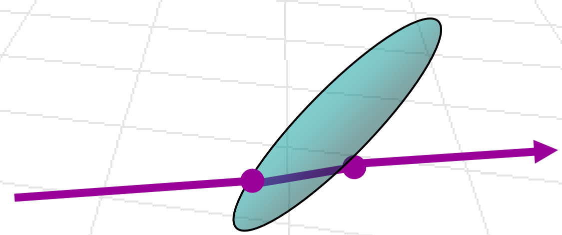
Conics



Spheres



Affinely Deformed Surfaces



Constructive Solid Geometry

Computer Graphics, CSCD18

Fall 2008

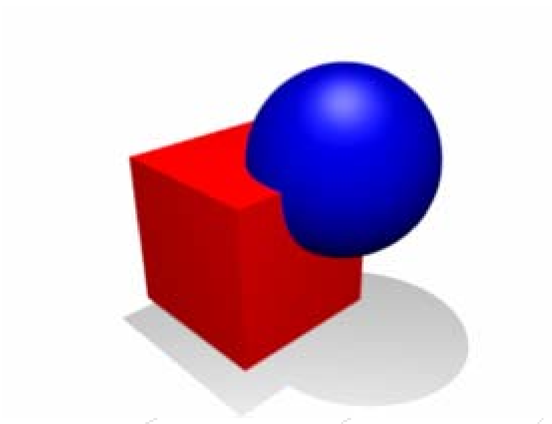
Instructor: Leonid Sigal

Constructive Solid Geometry

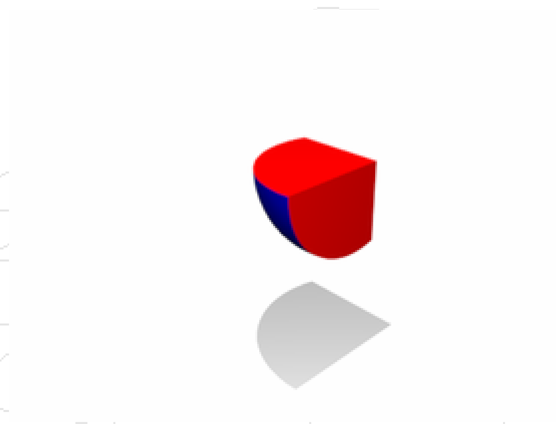
- **Idea:** construct a more expressive class of geometrical models by combining the basic geometric primitives we already studied

- To do this we define boolean operators on overlapping geometric objects

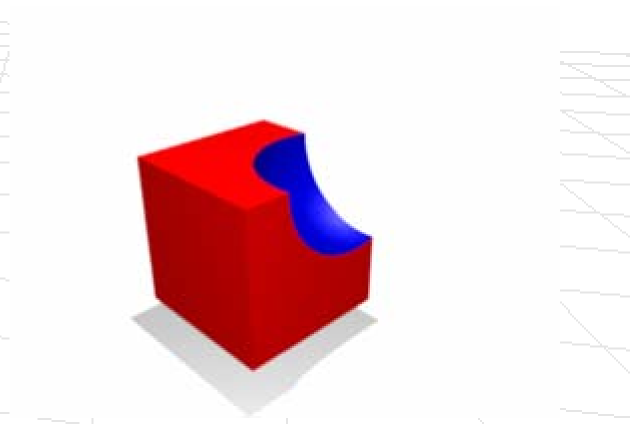
Union



Intersection

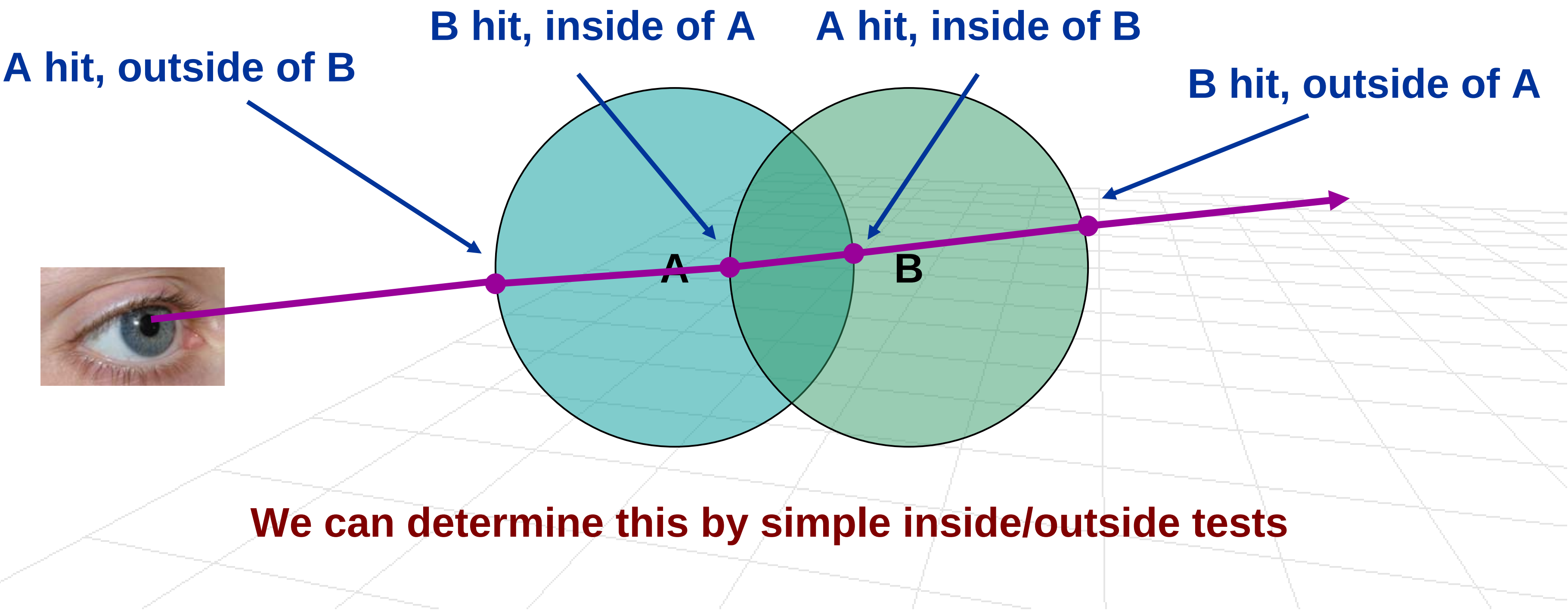


Subtraction



How do we intersect a CFG geometry?

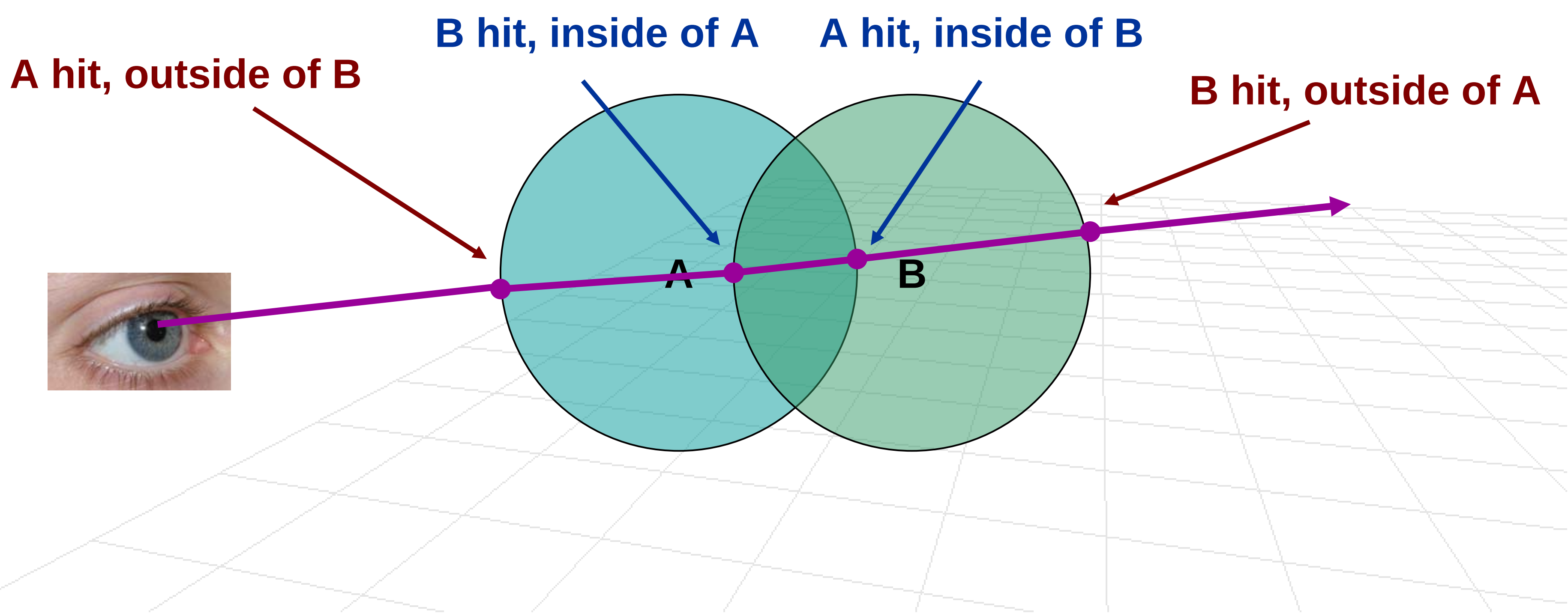
- We can use the original intersection tests + a bit of logic



How do we intersect a CFG geometry?

- We can use the original intersection tests + a bit of logic

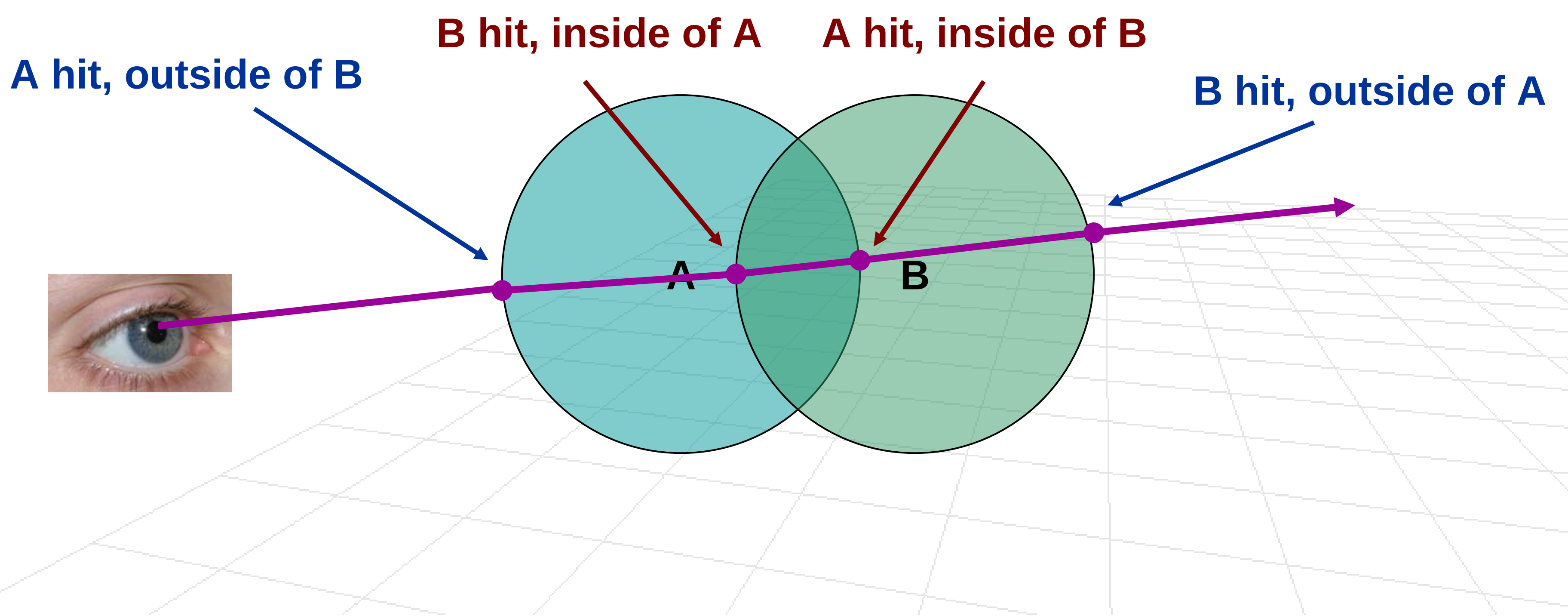
For example, for union operator we must consider



How do we intersect a CFG geometry?

- We can use the original intersection tests + a bit of logic

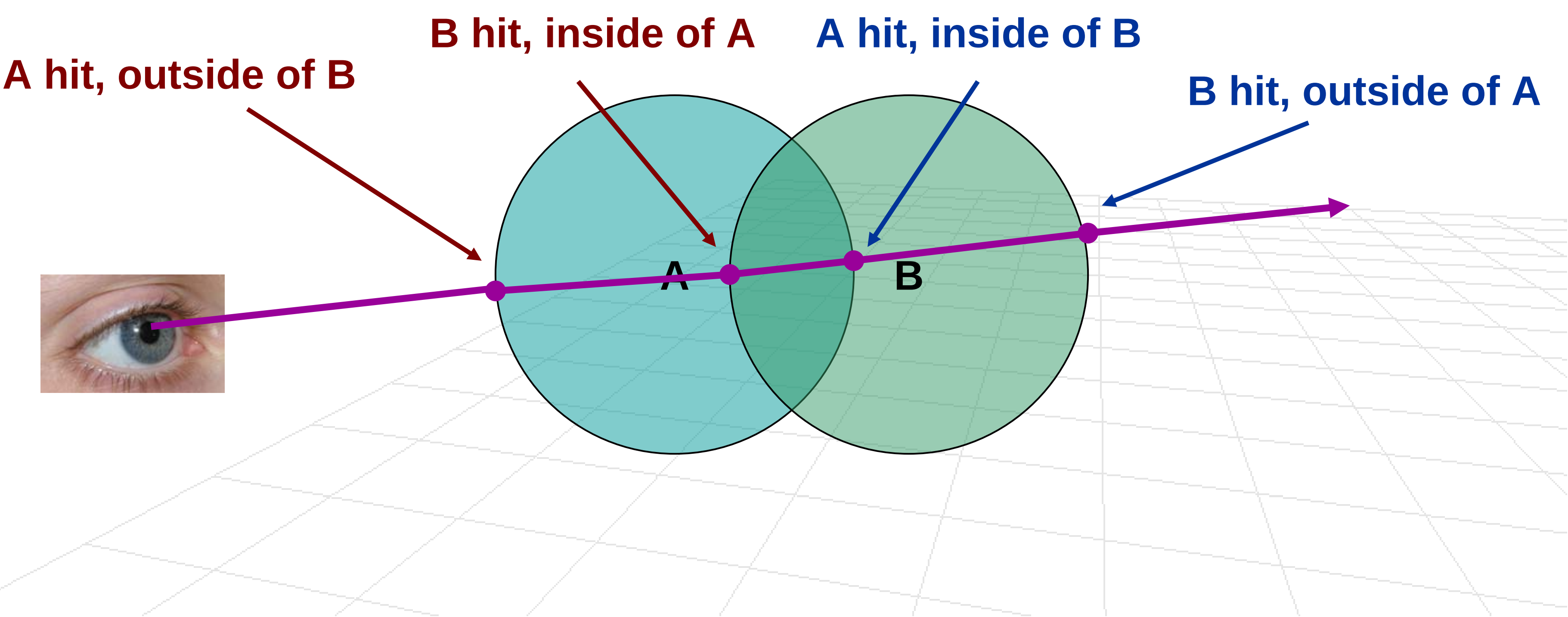
For example, for intersection operator we must consider



How do we intersect a CFG geometry?

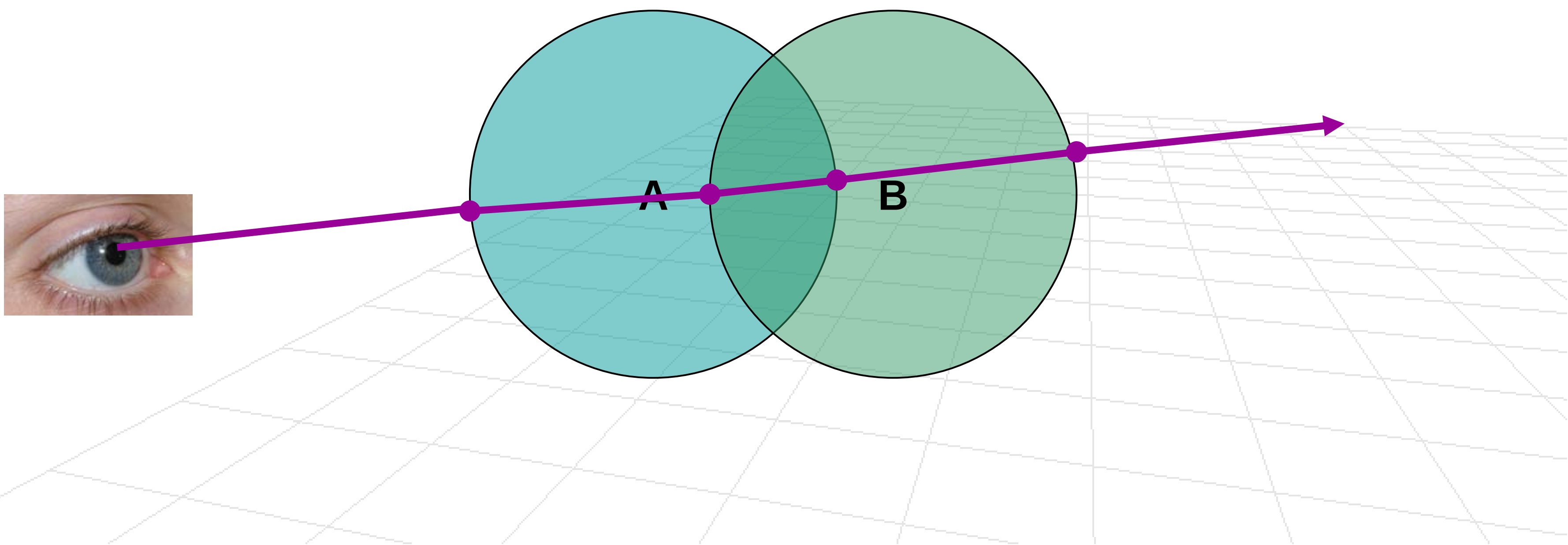
- We can use the original intersection tests + a bit of logic

For example, for subtraction operator (assuming A-B) we must consider

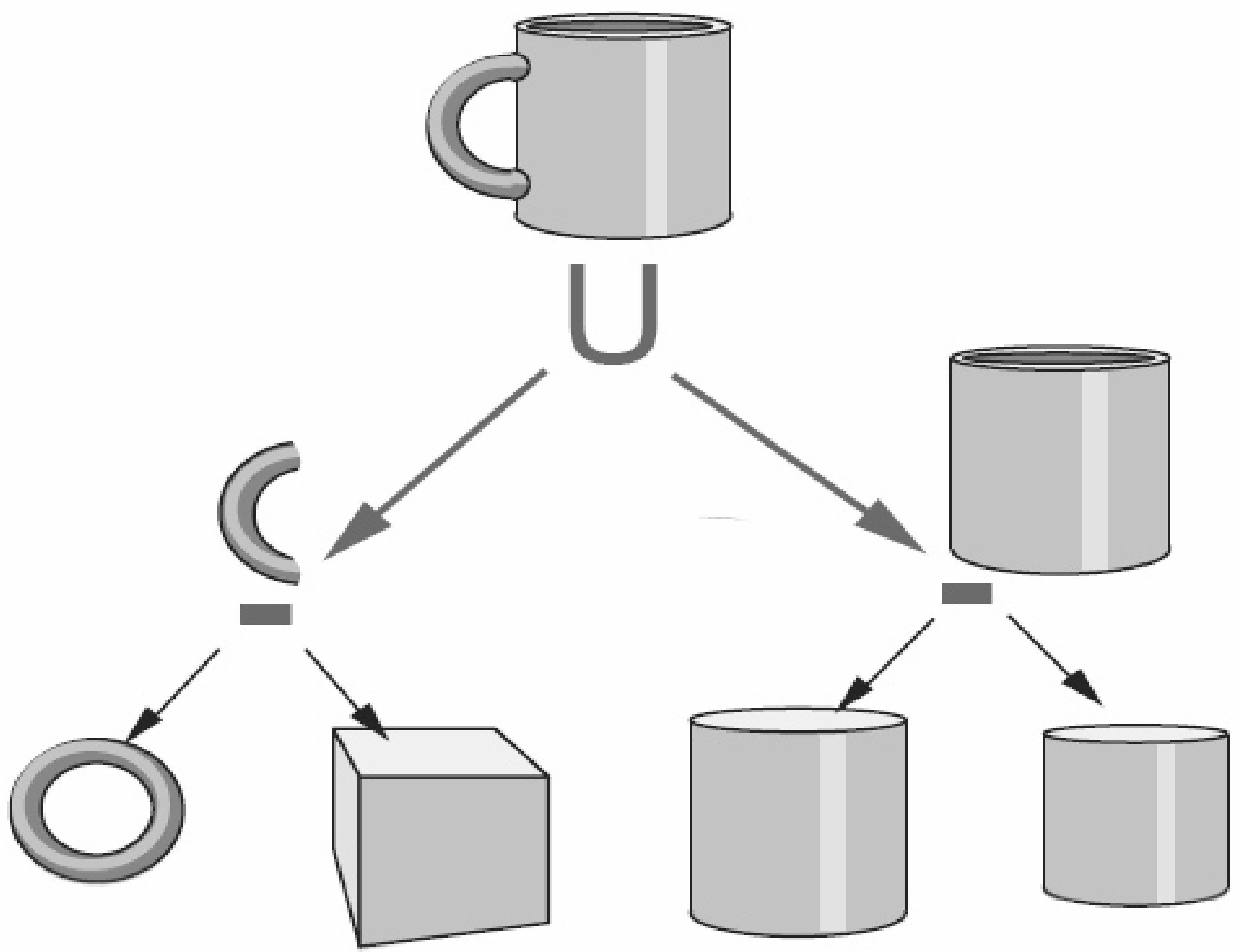


What about normals at “hit points”?

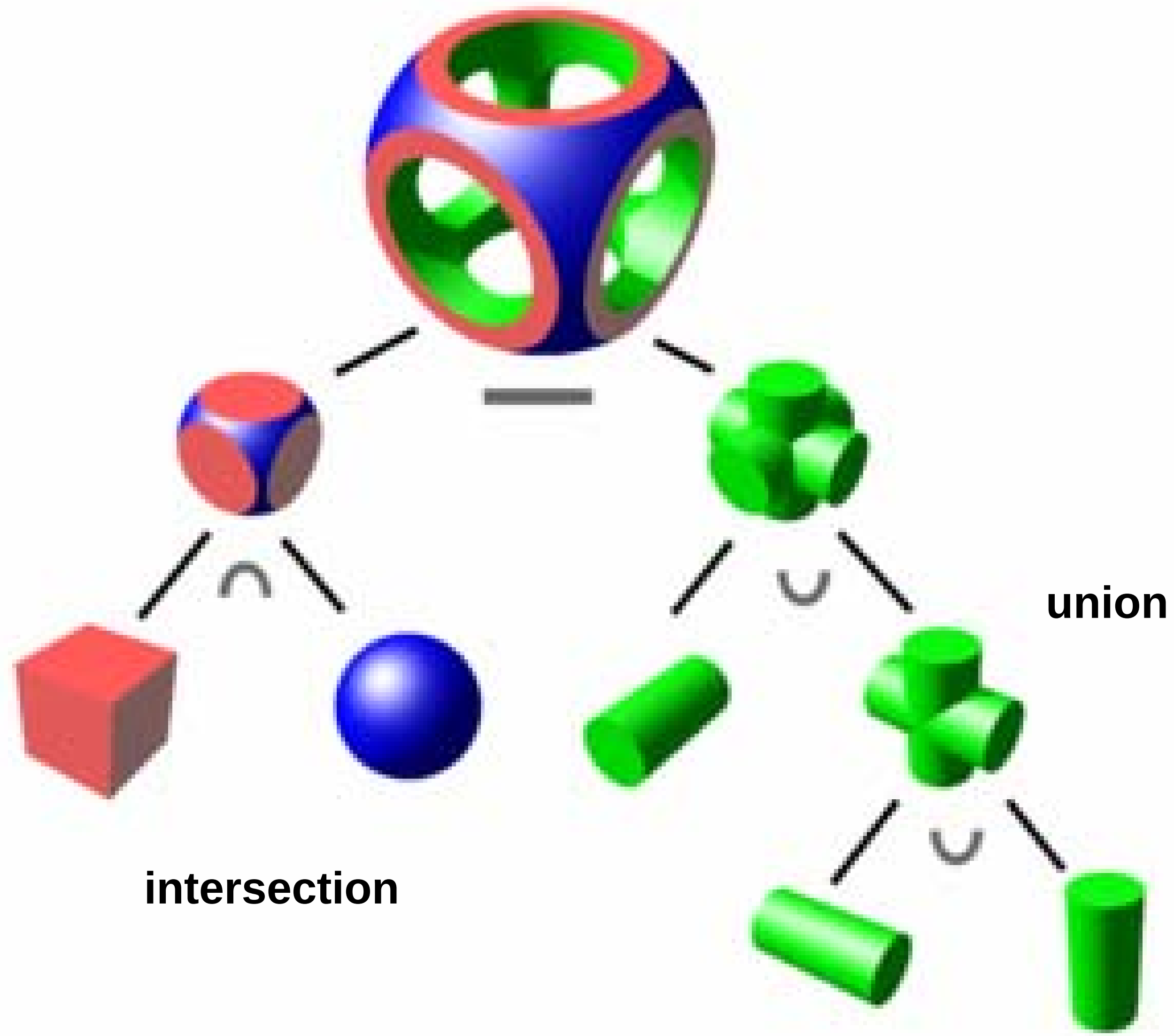
- Simple rules:
  - If **object is positive**, then the computed normal at the “hit point” is **outward facing**
  - If **object is negative**, then the computed normal at the “hit point” is **inward facing** (and needs to be flipped)



What can you build out of what we know?



Complex CFG



Benefits of CFG

- Builds **complex geometry** based on **simple primitives**
- Requires **no additional intersection code**
- Relatively inexpensive and easy to use in a Ray Tracer
- Naturally partitions objects and the scene in terms of hierarchical representation
  - Allows for efficient rendering

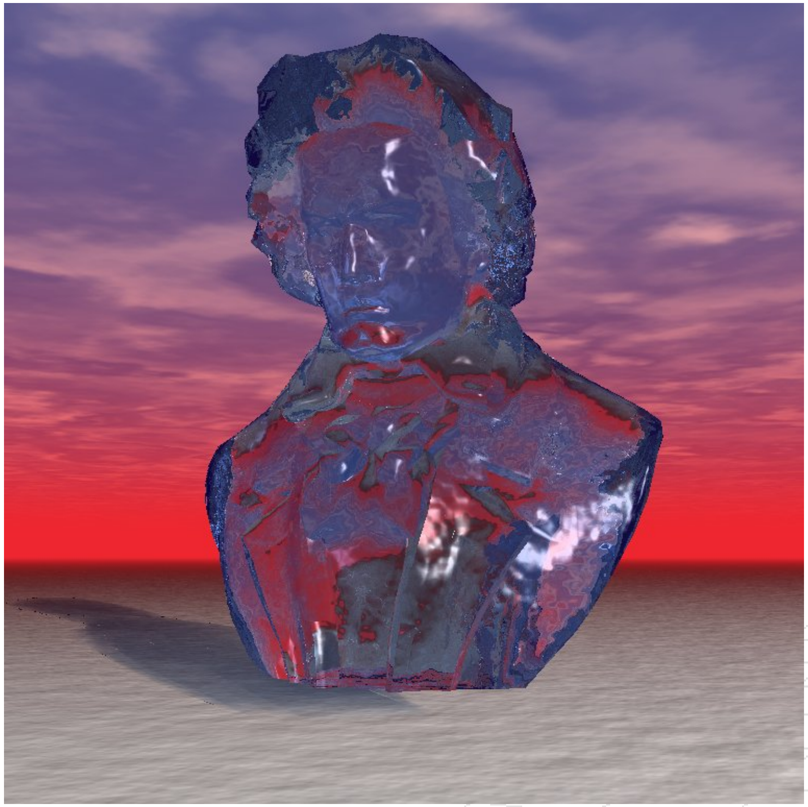
Ray Tracing
Part 3: Refraction and Shadows

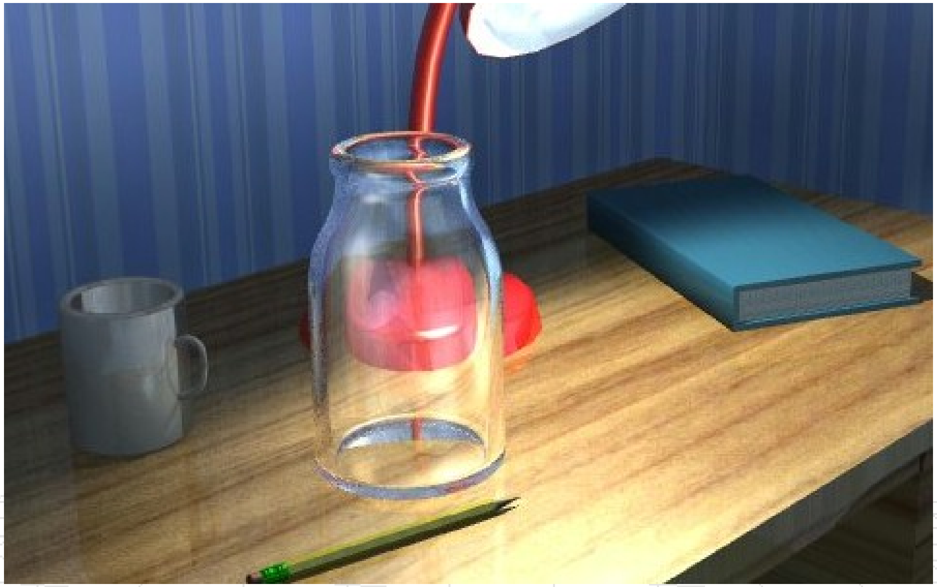
Computer Graphics, CSCD18

Fall 2008

Instructor: Leonid Sigal

What do we want to model?





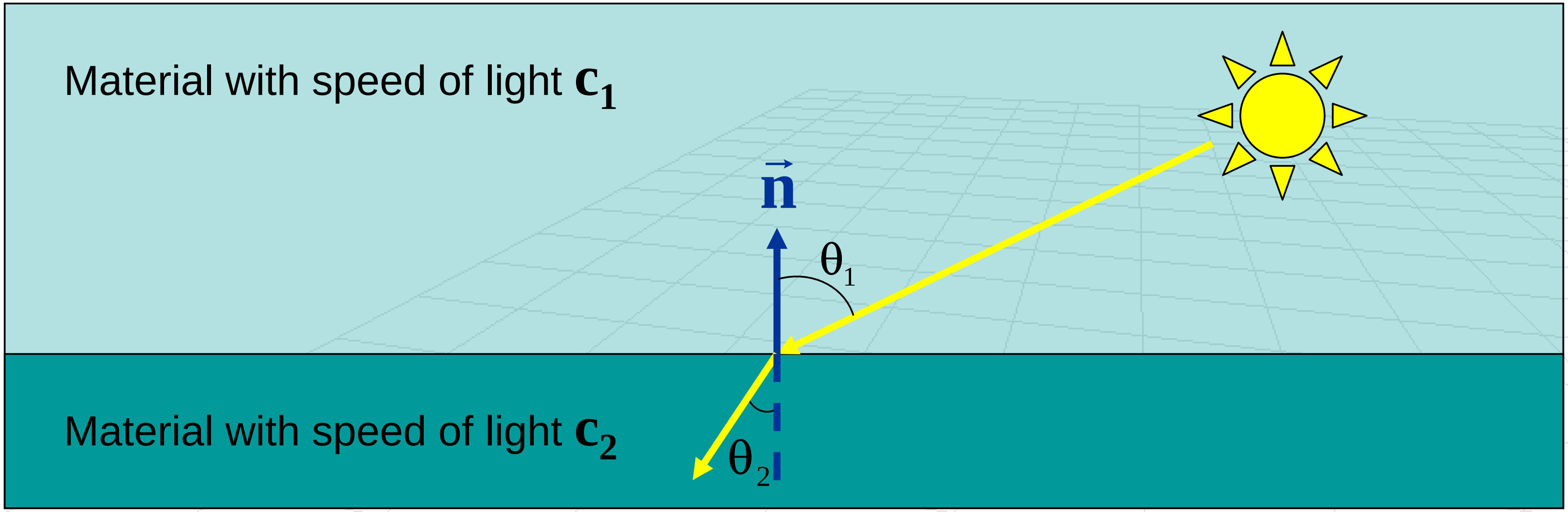
Transmission or refraction effects in semi-transparent surfaces

Transmission or Refraction

- **Physics:** light that penetrates a (partially or fully) transparent surface or material is refracted (bent) to account for change in the speed of light transmission in different media
- **Snell's law governs refractions** $$\frac{\sin\theta_1}{\sin\theta_2} = \frac{\mathbf{c}_1}{\mathbf{c}_2}$$

Transmission or Refraction

- **Physics:** light that penetrates a (partially or fully) transparent surface or material is refracted (bent) to account for change in the speed of light transmission in different media
- **Snell's law governs refractions** $$\frac{\sin\theta_1}{\sin\theta_2} = \frac{\mathbf{c}_1}{\mathbf{c}_2}$$



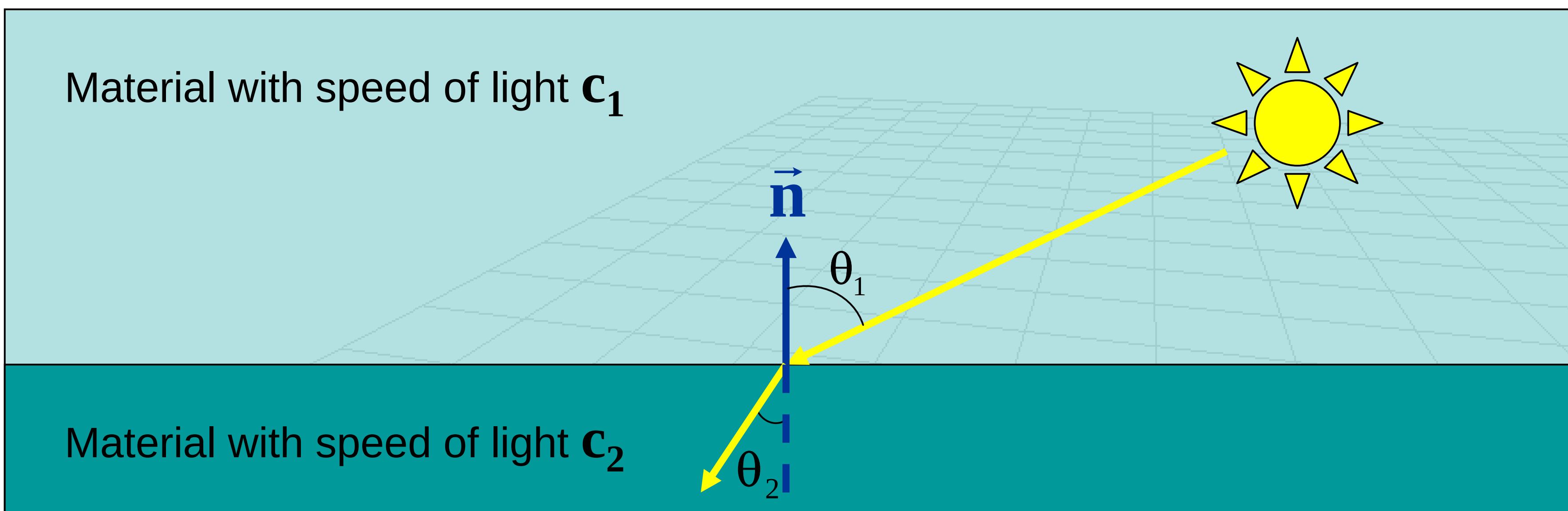
Transmission or Refraction

- **Physics:** light that penetrates a (partially or fully) transparent surface or material is refracted (bent) to account for change in the speed of light transmission in different media

- **Snell's law governs refractions**

index of refraction

$$\frac{\sin\theta_1}{\sin\theta_2} = \frac{\mathbf{c}_1}{\mathbf{c}_2}$$

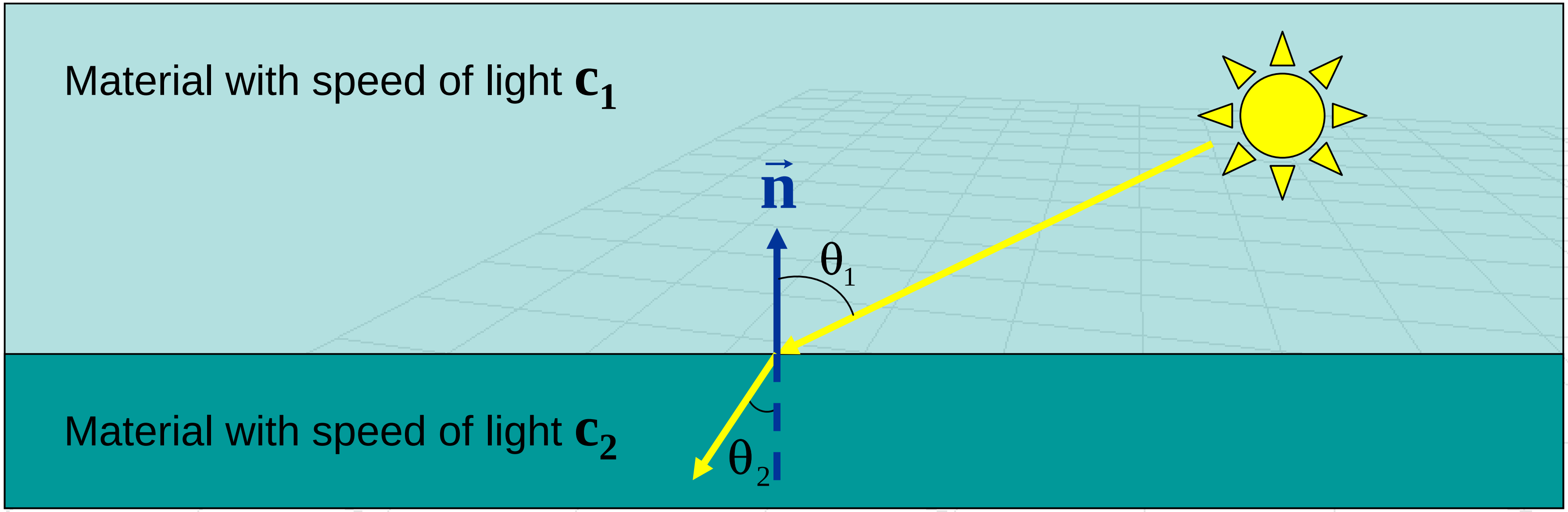


Transmission or Refraction

- For example, $\frac{\mathbf{c}_{\mathbf{air}}}{\mathbf{c}_{\mathbf{water}}} \approx 1.33$ $\frac{\mathbf{c}_{\mathbf{air}}}{\mathbf{c}_{\mathbf{glass}}} \approx 1.8$

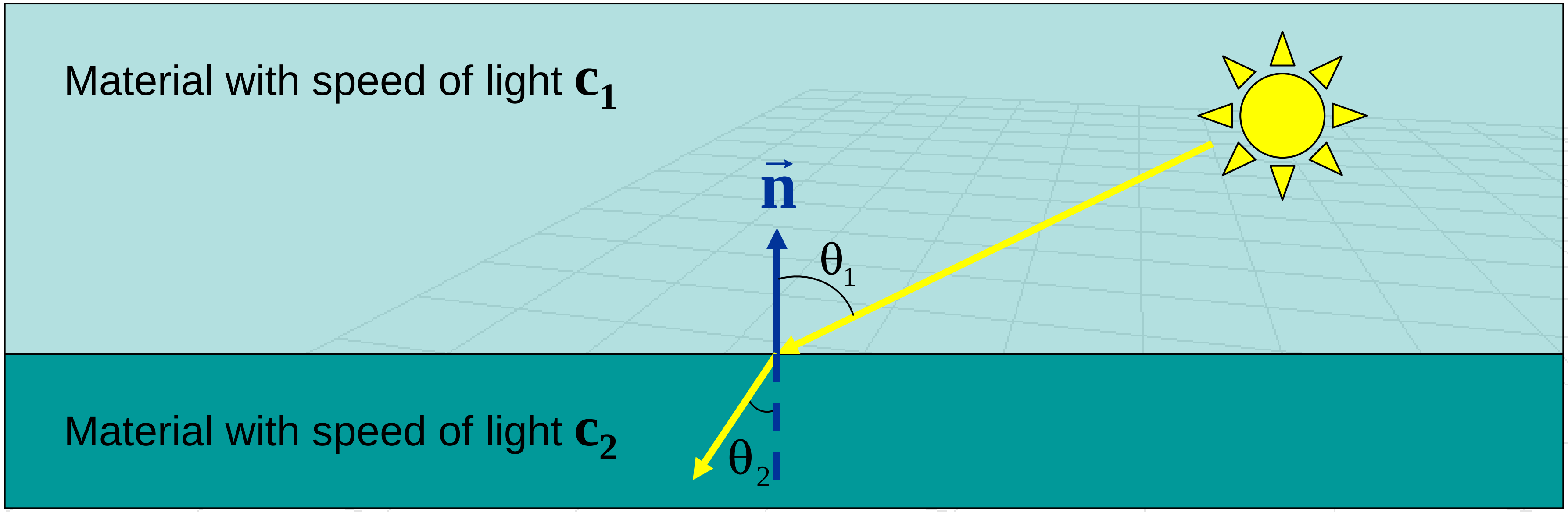
$\mathbf{c}_2 < \mathbf{c}_1$ light bends toward the normal (eg. air to water)

$\mathbf{c}_2 > \mathbf{c}_1$ light bends away from the normal (eg. water to air)



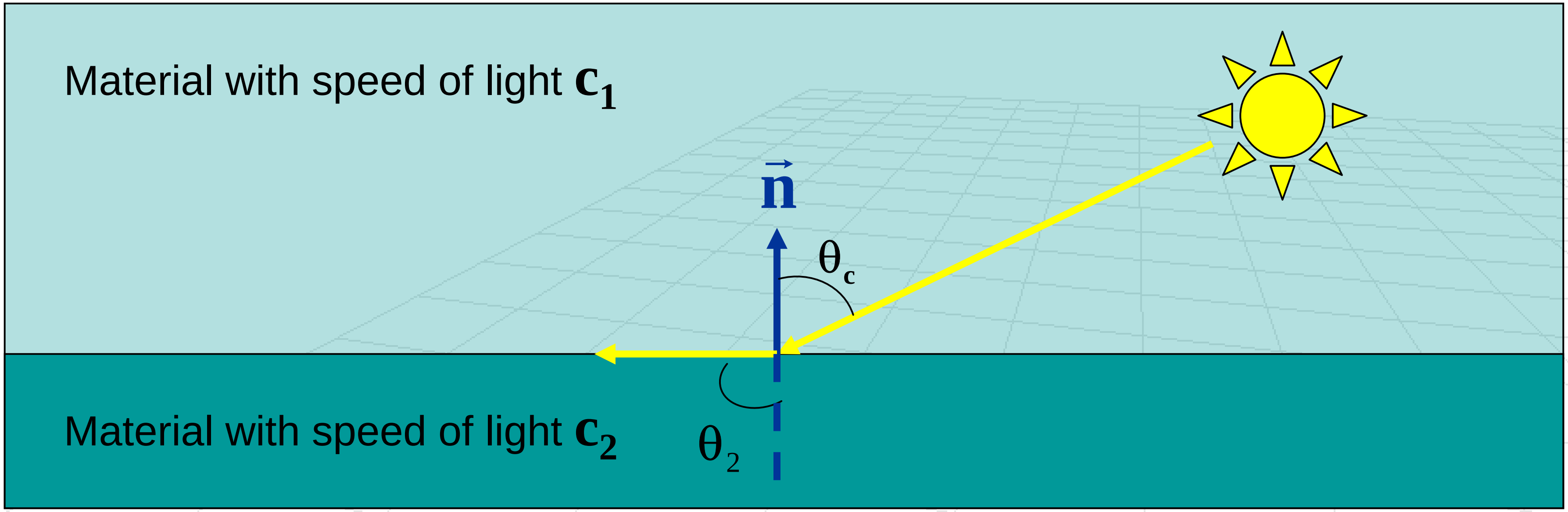
Common Indices of Refraction

Material	Index of Refraction
vacuum	1.0
ice	1.309
water	1.333
ethyl alcohol	1.36
glass	1.5–1.6
diamond	2.417



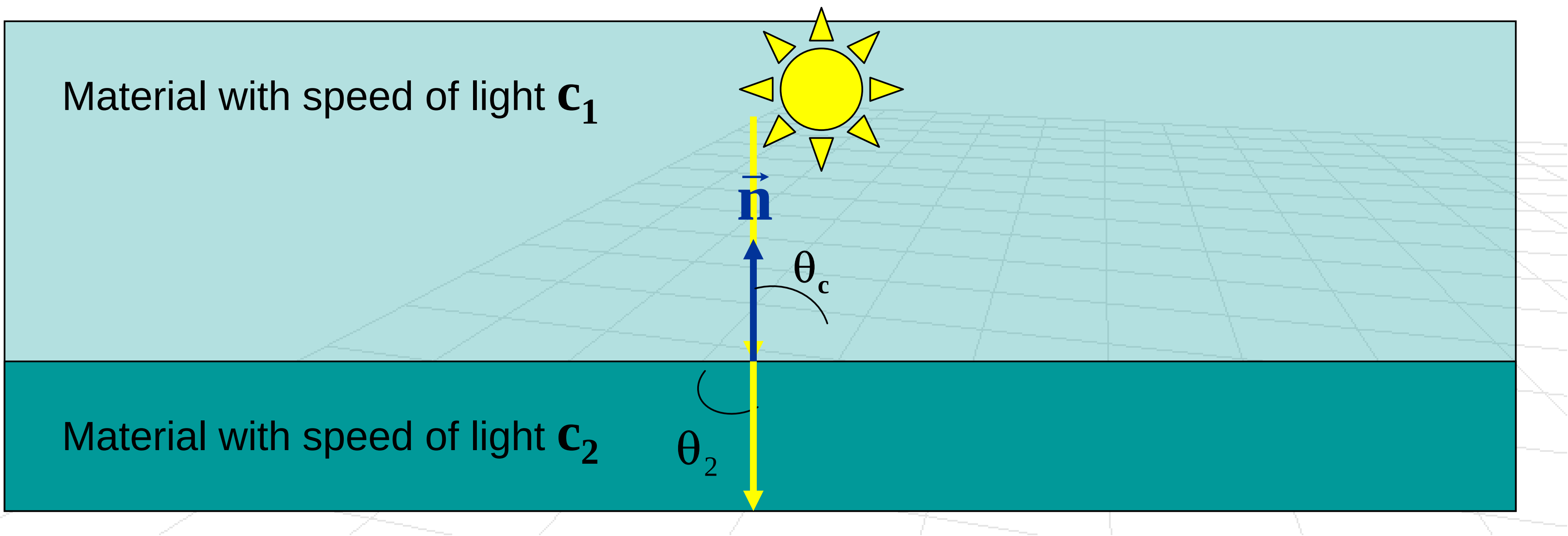
Transmission or Refraction

- Critical angle (for $\mathbf{c_2} > \mathbf{c_1}$)
  - As incoming angle approaches critical angle, the outgoing angle approaches 90 degrees
  - No light enters the material



Transmission or Refraction

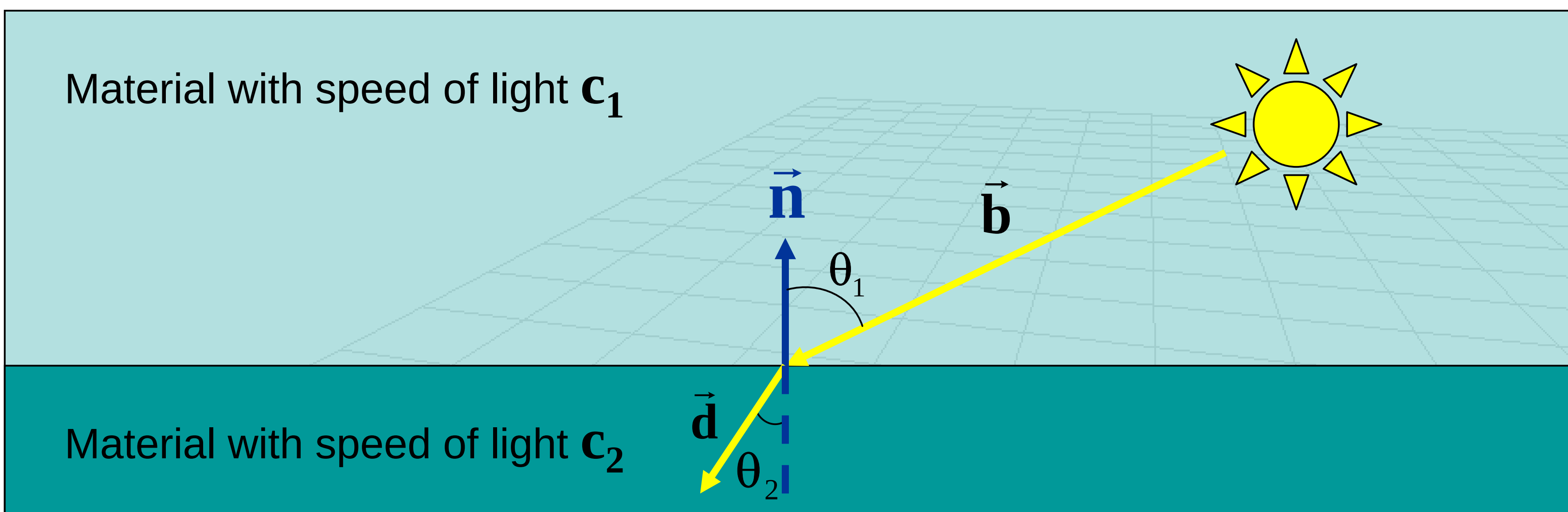
- Special case
  - If incoming angle is 0 the outgoing angle is also 0
  - No light bending



Refraction in Ray Tracing

- We can treat global refraction/transmission just like global specular reflection (i.e. cast one ray)
  - Need to keep track of the speed of light in the current medium
- Perfect refraction direction

$$\vec{\mathbf{d}} = \frac{\mathbf{c}_2}{\mathbf{c}_1}\vec{\mathbf{b}} + \left(\frac{\mathbf{c}_2}{\mathbf{c}_1}\left(\vec{\mathbf{n}}\cdot\vec{\mathbf{b}}\right) - \left(1 - \left[\frac{\mathbf{c}_2}{\mathbf{c}_1}\right]^2 \left(1 - \left(\vec{\mathbf{n}}\cdot\vec{\mathbf{b}}\right)^2\right)\right)^{1/2} \right)\vec{\mathbf{n}}$$



More complex scene with refraction

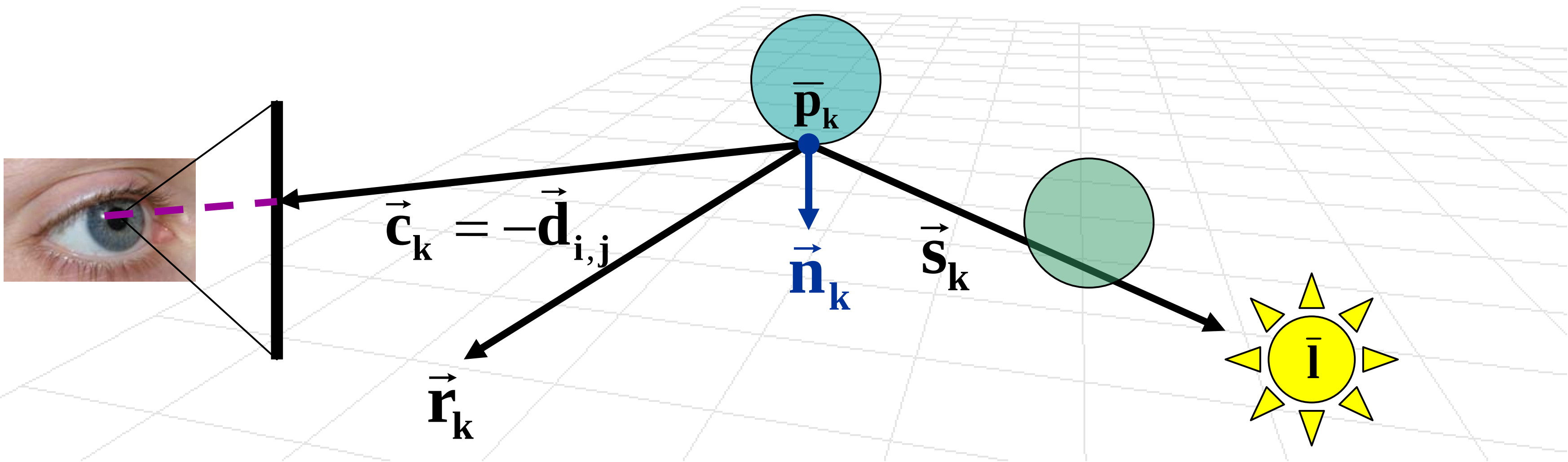
Shadows

- Easy to deal with in ray tracing
  - When point is in shadow, turn off local reflection
- To do so, cast a ray towards a light source

$$\bar{\mathbf{r}}(\lambda) = \bar{\mathbf{p}}_k + \lambda(\bar{\mathbf{l}} - \bar{\mathbf{p}}_k)$$

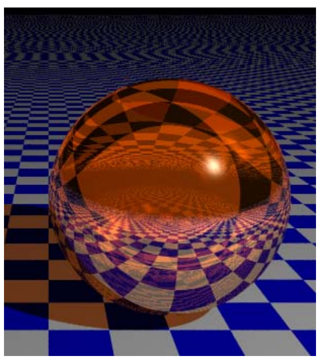
if there is a hit point $0 \le \lambda \le 1$, turn off local reflection (diffuse and specular components of Phong)

$$\mathbf{E}_k = \mathbf{r}_d \mathbf{I}_d \max(0, \vec{\mathbf{s}}_k \cdot \vec{\mathbf{n}}_k) + \mathbf{r}_a \mathbf{I}_a + \mathbf{r}_s \mathbf{I}_s \max(0, \vec{\mathbf{r}}_k \cdot \vec{\mathbf{c}}_k)^\alpha + \mathbf{r}_g \mathbf{I}_{spec}$$



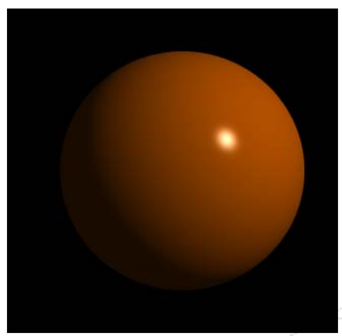
Review of Ray Tracer Shading

Final Image



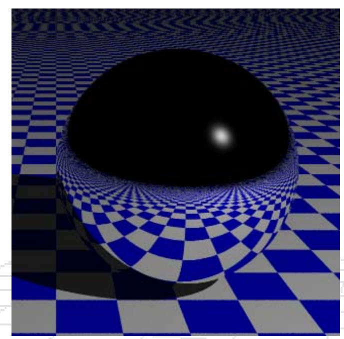
=

Local Lighting Phong



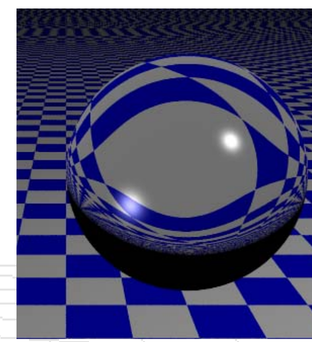
+

Global Reflections (and shadows)



+

Refractions



[images from lecture notes by John Dingliana]