Computer Graphics

Ray Tracing

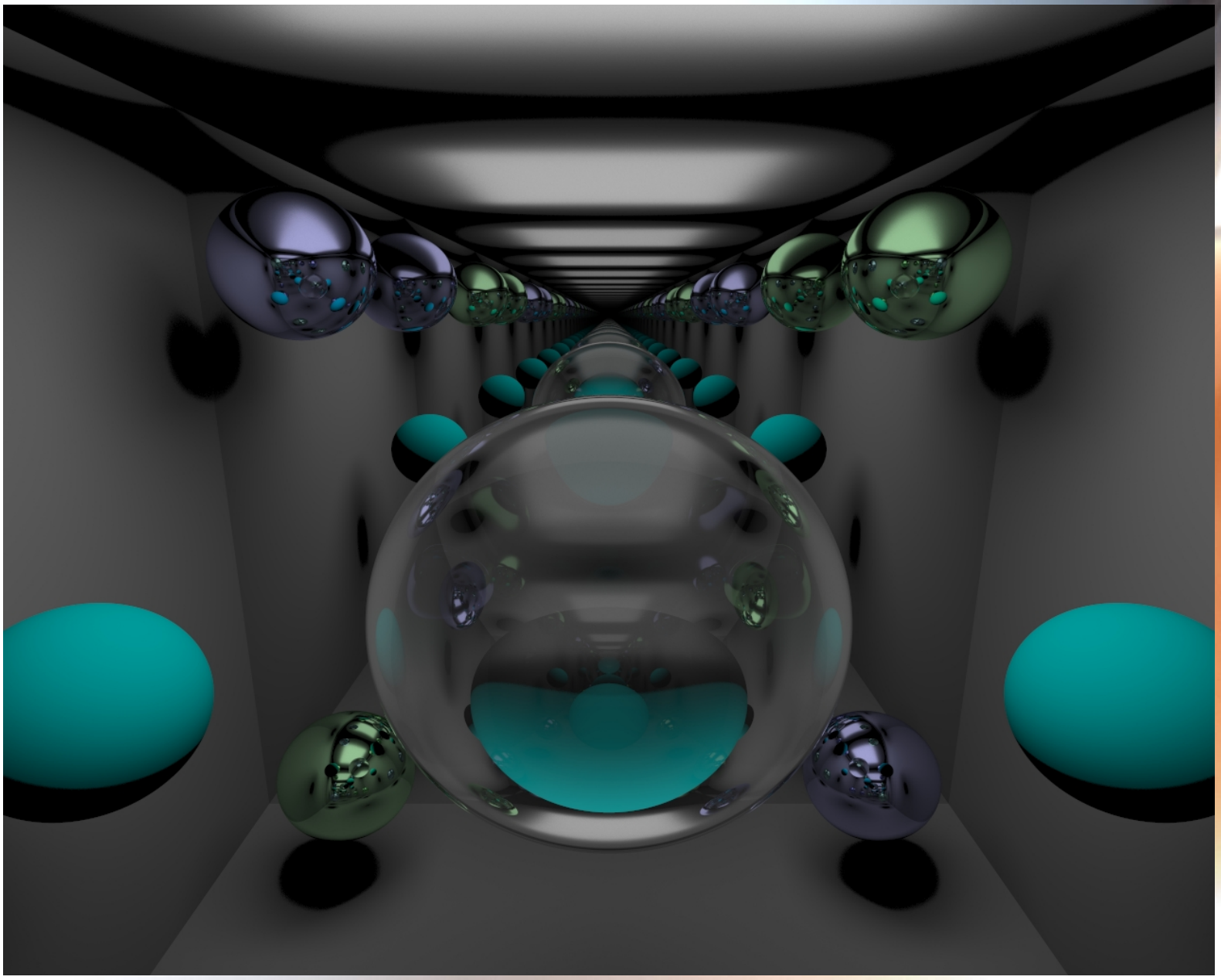
Based on slides by Dianna Xu, Bryn Mawr College

Ray Tracing Example

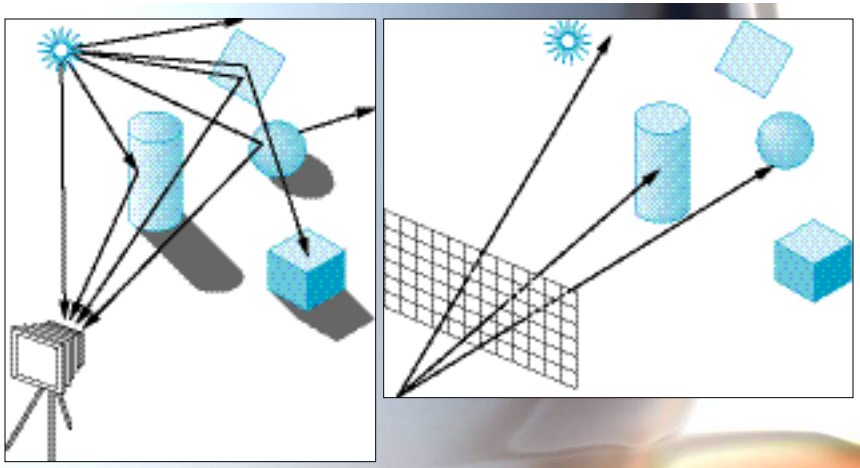
Created by Anto Matkovic

Ray Tracing Example

Ray Tracing Example



Ray Tracing



- Most light rays do not reach the eye
- Start from eye point
- Cast one ray through each pixel
- Each ray either intersects or not
- If no intersection – assign background color
- If intersection – trace reflected, refracted, and shadow rays.

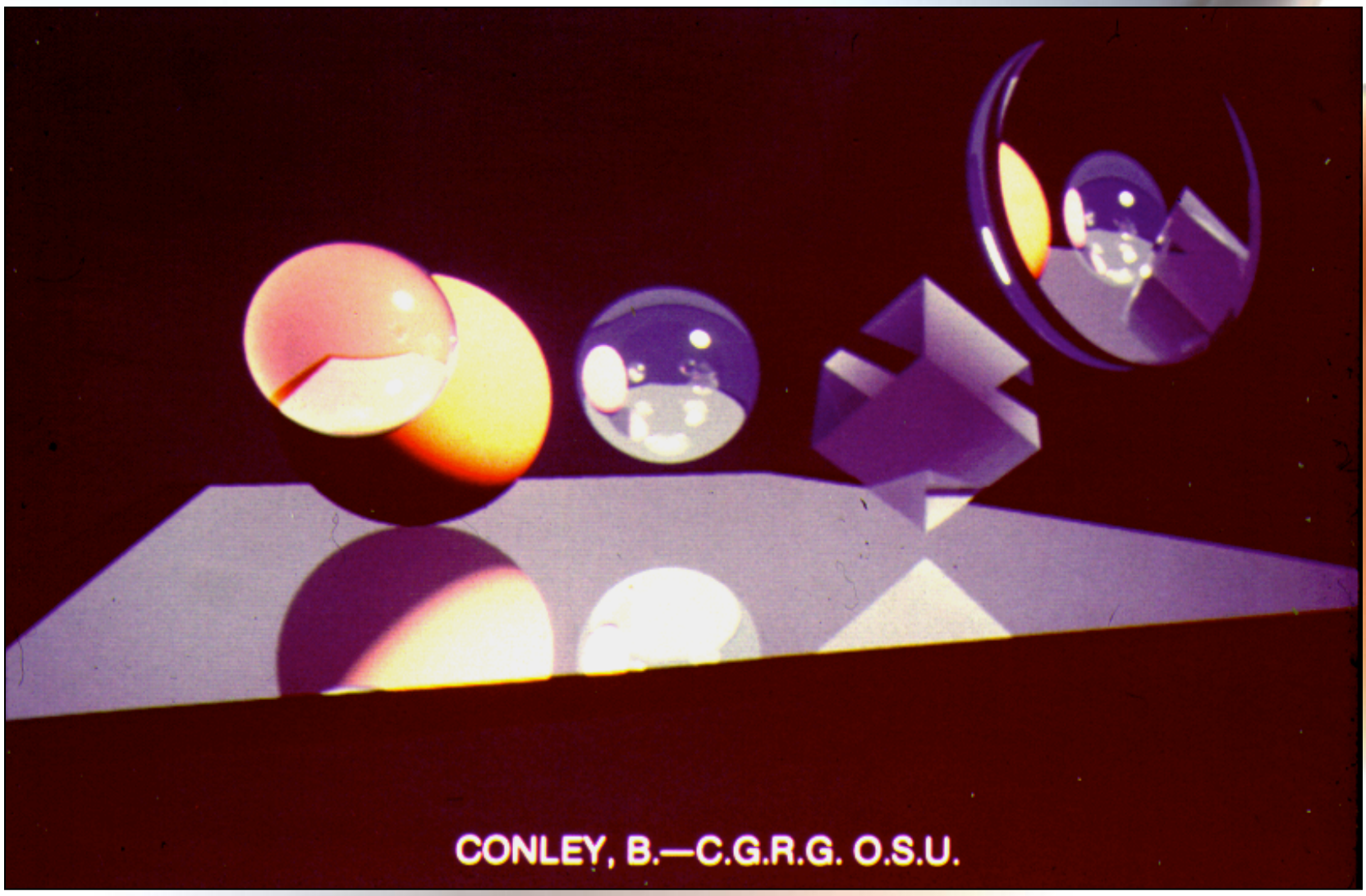
Simple Ray Tracing Algorithm

```
Color trace(Point p, Vector d, int step) {
  Color local, reflected, transmitted;
  Point q; Vector normal;
  if (step > max) return (background_color);

  q = intersect(p, d, status);
  if (status == no_intersection) return
  (background_color);
  normal = do_normal(q);
  Vector reflectdir = reflect(q, normal);
  Vector transdir = transmit(q, normal);
```

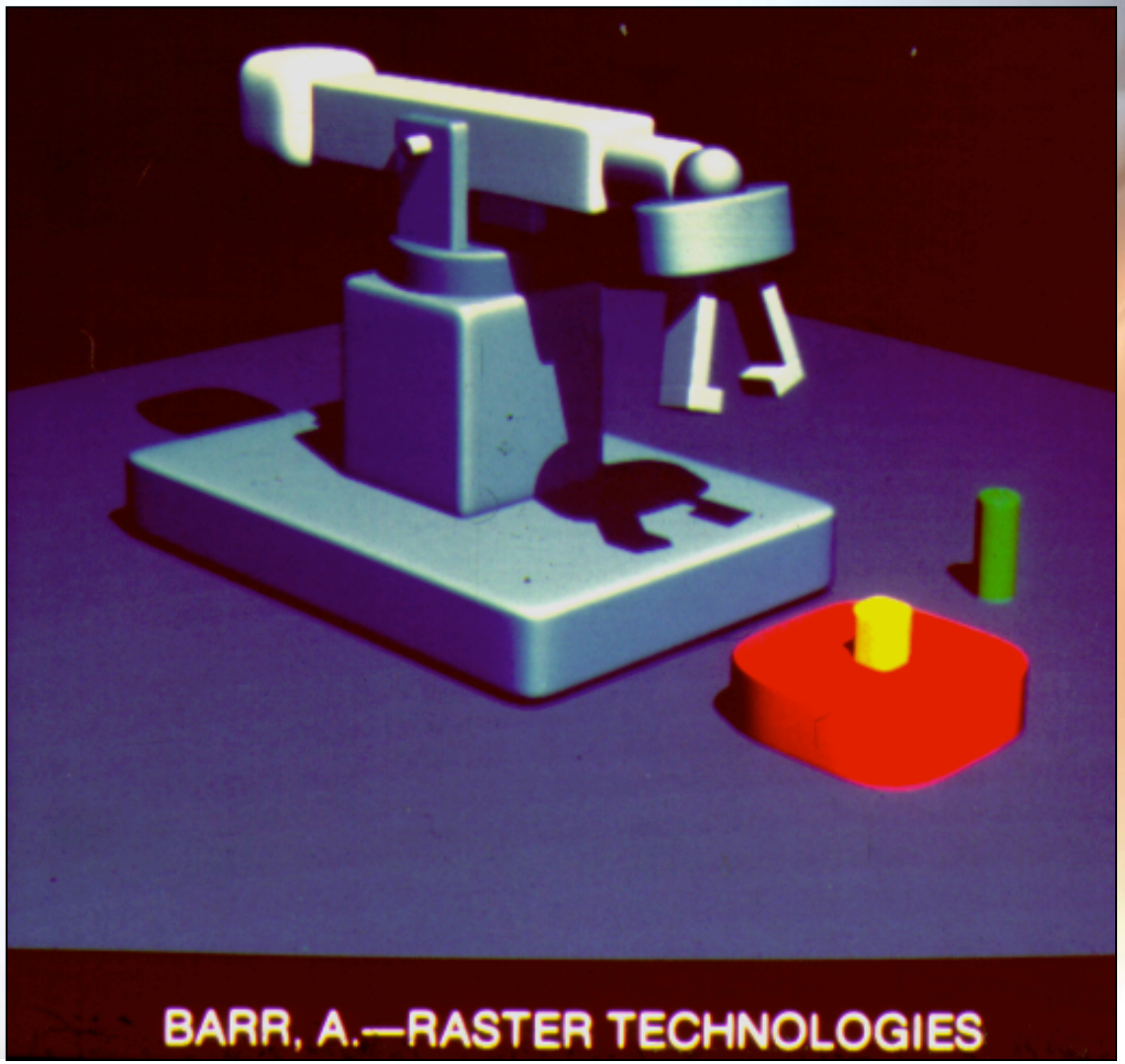
Simple Ray Tracing Algorithm

```
  // must check shadow ray
  local = phong(q, normal, reflectdir);
  reflected = trace(q, reflectdir, step+1);
  transmitted = trace(q, transdir, step+1);

  return(local+reflected+trasmitted);
}
```

Ray Tracing with Several Ray Effects



Ray Tracing Necessary to Render Superellipsoids



Ray-Sphere Intersection

- A sphere is described by its center c and radius r.
- A point p on sphere: $\left|\vec{c}\vec{p}\right| = \left|p - c\right| = r$
- A ray is described by its origin o, and a direction vector d.
- A point on a ray has the form: $o + t\vec{d}$

Ray-Sphere

- **Substituting into sphere equation:**

$$\left| o + t\vec{d} - c \right| = r$$

- **Squaring both sides we get:**

$$\left|\vec{c}\vec{o}\right|^2 + 2(\vec{c}\vec{o} \cdot \vec{d})t + \left|\vec{d}\right|^2 t^2 = r^{2`}$$

Quadratic Equation

- **Rewrite as quadratic quation:**

$$At^2 + Bt + C = 0$$

- **where** $A = \left|\vec{d}\right|^2, B = 2(\vec{co} \cdot \vec{d}), C = \left|\vec{co}\right|^2 - r^2$

- **Number of solutions depends on discriminant:**

$$\Delta = B^2 - 4AC$$

The Discriminant

1. $\Delta < 0$ ==> no solution (ray misses)

2. $\Delta = 0$ ==> one solution (ray tangent)

$$-B/2A$$

3. $\Delta > 0$ ==> two solutions (ray punctures)

$$(-B \pm \sqrt{\Delta})/2A$$

- Ray extends only in one direction (positive t) – want closest intersection – solution with least non-negative t.

The Result

- In case 1, report no insersection
- In case 2, report no intersection if t<0 ==> B>0, otherwise return solution (-B/2A) for t
- In case 3, return the least positive of the two solutions for t. If both negative, report no intersection
- If there is intersection, then the intersection point is o+td.

Ray-Triangle Intersection

- Compute intersection point of ray and the plane of the given triangle.
- If there is intersection, decide if the intersection point lies inside or outside of the triangle.

Ray-Plane

- **Given vertices a1, a2, a3.**
- **Normal:** $\vec{n} = (a_2 - a_1) \times (a_3 - a_1)$
- **A point p on the plane of triangle:**

$$(p - a_1) \cdot \vec{n} = 0$$

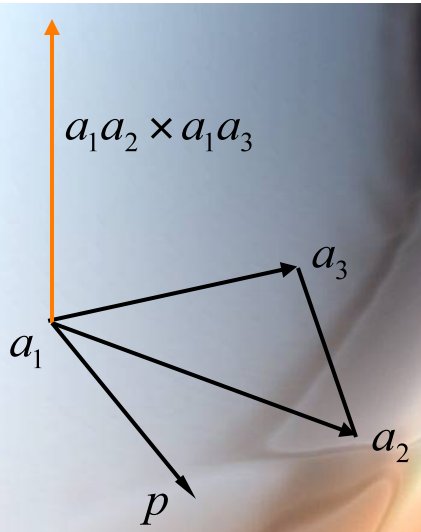
- **Substituting** $p = o + t\vec{d}$
- **Solves to**

$$t = -\frac{(o - a_1) \cdot \vec{n}}{\vec{d} \cdot \vec{n}}$$

Point inside Triangle

- **Check the cross product:**

$$a_1a_2 \times a_1a_3$$



- **If this cross product has the same direction as the normal $a_1a_2 \times a_1p$ then point p is on the same side of a1a2 as a3**
- **Repeat for the other two edges.**

Point inside Triangle

```
boolean SameSide(Point p,Point a1,Point
  a2,Point a3) {
   Vector cp1 = cp(a2-a1, p-a1);
   Vector cp2 = cp(a2-a1, a3-a1);
   if (dp(cp1, cp2)) >= 0
     return true;
   else return false; }
boolean PointInTriangle(Point p,Point
  a,Point b,Point c) {
 if (SameSide(p,a,b,c)&& SameSide(p,b,a,c)
  && SameSide(p,c,a,b))
    return true;
 else return false; }
```

Viewing Geometry for a Simple Ray Tracer

- Screen described by a point l – its lower left corner.
- Two vectors (h and v) used to iterate over the raster positions on the target NDC space.
- Eye is given by e.
- Compute ray direction from eye to pixel (i,j)

Ray Direction

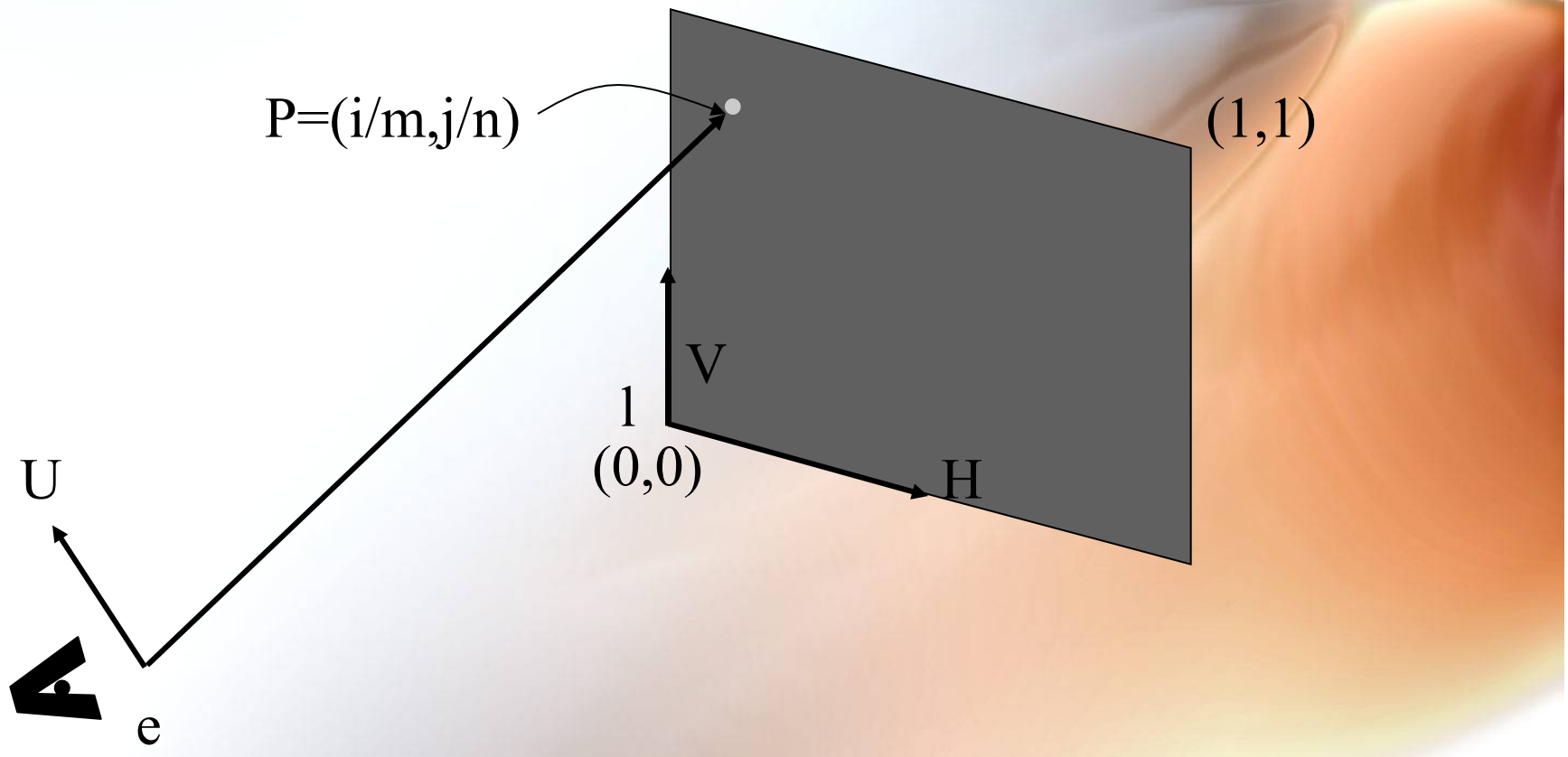
- To reach the lower left corner of desired pixel (i,j) from l : $l + (i * \vec{h} / m + j * \vec{v} / n)$
- The move to the center of the pixel:

$$l + (\frac{i + 0.5}{m} \vec{h} + \frac{j + 0.5}{n} \vec{v})$$

- Direction of ray is a vector from eye to center of desired pixel:

$$(l + (\frac{i + 0.5}{m} \vec{h} + \frac{j + 0.5}{n} \vec{v})) - e$$

Simple Viewing Geometry



The Rays and the Outer Loops

- Iterate this for each pixel (integer) coordinate pair.

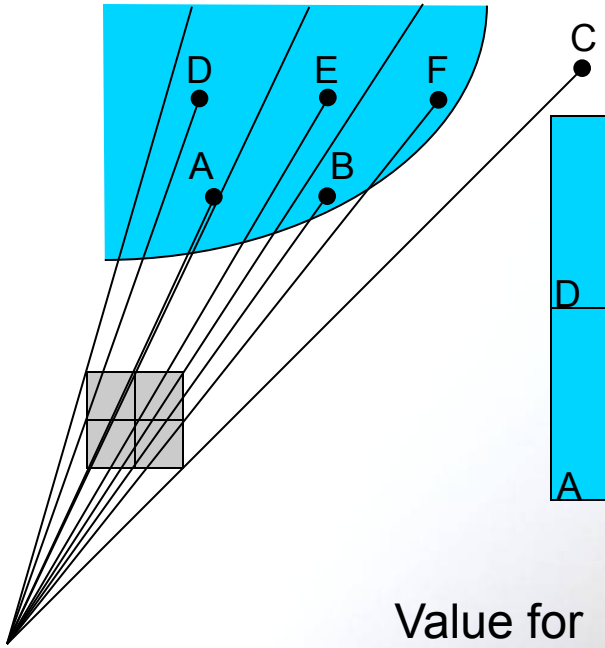
```
for pixel(x,y) x = 0 to width-1
   for pixel(x,y) y = 0 to height-1 {
     R=(l+(i+0.5)/m*h+(j+0.5)/n*v)-e;
      color = raytrace(e, R, 0)
      pixel(x,y) = color;
   }
```

Antialiasing

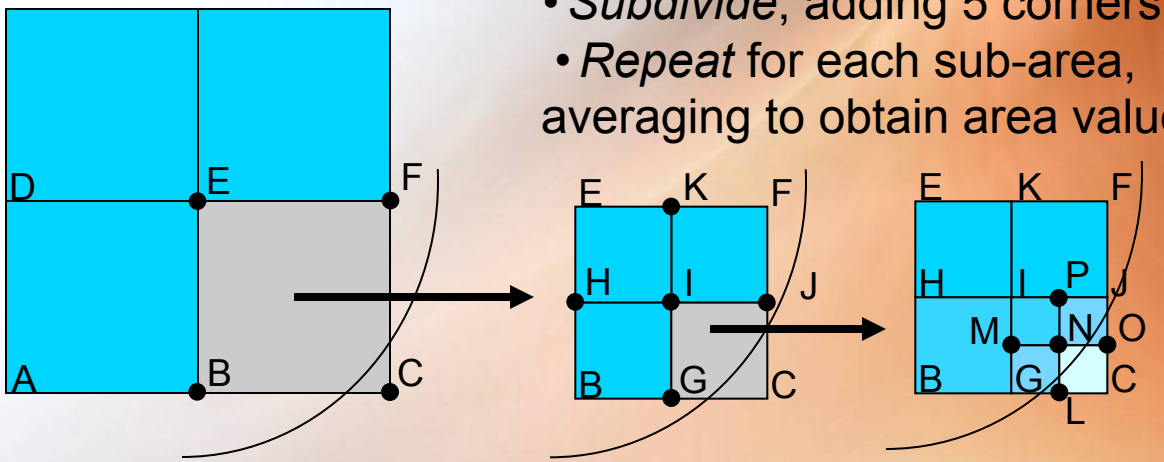
- Ray tracing is a sampling method, and is thus subject to aliasing errors.
- Over sample and then average
- Instead of one ray through the center of the pixel, shoot multiple rays just slightly off the original center (left, right, up, down), and average the results as the final output.

Antialiased Ray Tracing

- **Adaptive Supersampling (Weighted sub-pixel averaging)**



- Cast rays through pixel corners (not centers)
- Take *average* of corner values
- If one corner differs significantly from average,
- *Subdivide*, adding 5 corners
- *Repeat* for each sub-area, averaging to obtain area value



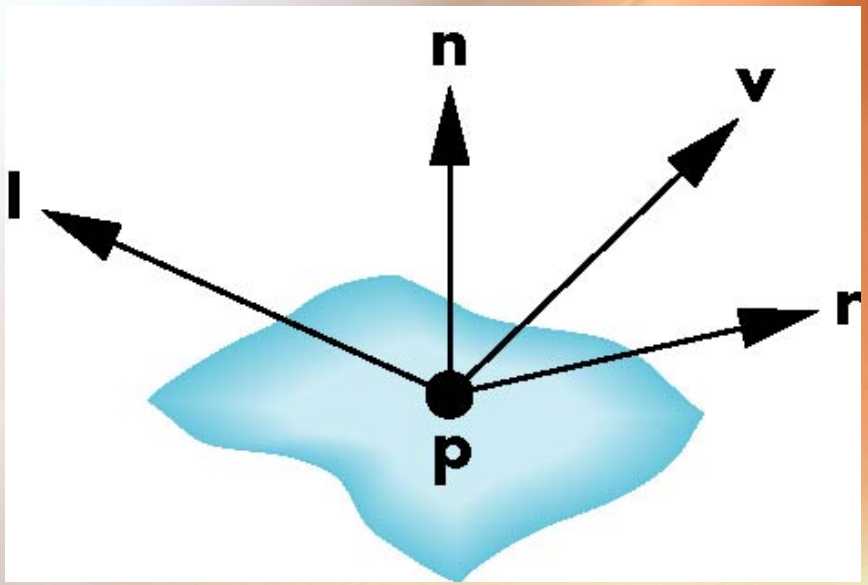
Value for pixel BCEF = $\frac{1}{4}\left[\frac{1}{4}\left[\frac{G+L+M+N}{4}+\frac{L+C+N+O}{4}+\frac{M+N+I+P}{4}+\frac{N+O+P+J}{4}\right]+\frac{B+G+H+I}{4}+\frac{H+I+E+K}{4}+\frac{I+J+K+F}{4}\right]$

Local Reflection Model

- A simple model that can be computed rapidly
- The intensity of the reflected light as a function of:
  - the orientation of the surface
  - with respect to the position of a point light source
  - and surface properties.
- Physical phenomena simulated
  - Perfect specular reflection
  - Imperfect specular reflection
  - Perfect diffuse reflection

The Phong Model

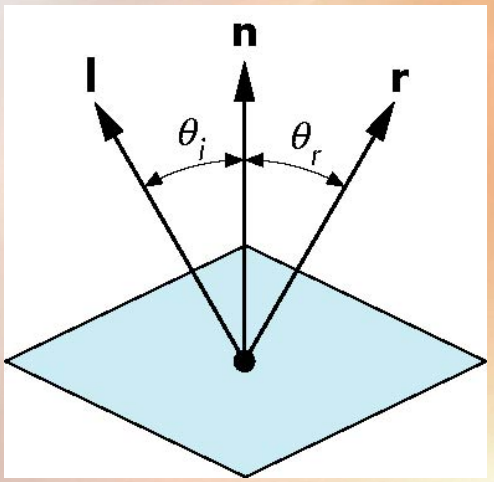
- Has three components
  - Diffuse
  - Specular
  - Ambient
- Uses four vectors
  - To source (l)
  - To viewer (v)
  - Normal (n)
  - Perfect reflector (r)



Ideal Reflector

- Normal is determined by local orientation
- Angle of incidence = angle of reflection
- The three vectors must be coplanar

$$R = 2(l \cdot n)n - l$$



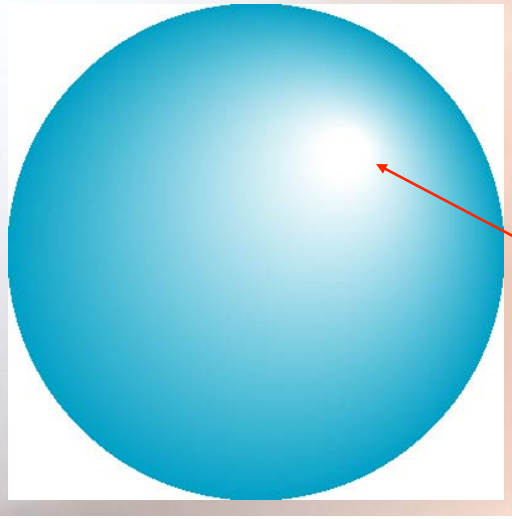
Perfectly Diffuse Reflector

- **Matte/Lambertian surface**
- **Light scattered equally in all directions**
- **Amount of light reflected is proportional to the vertical component of incoming light**
  - **reflected light depends on** $\cos\theta$
  - $\cos\theta =$ **if vectors normalized** $l \cdot n$

$$I_r = k_d \cos\theta I_d = k_d (l \cdot n) I_d$$

Specular Surfaces

- **Most surfaces are neither ideal diffusers nor perfectly specular (ideal reflectors)**
- **Smooth surfaces show specular highlights due to incoming light being reflected in directions concentrated close to the direction of a perfect reflection**



Modeling Specular Reflections

- **Phong proposed using a term that dropped off as the angle ϕ between the viewer and the ideal reflection increased**

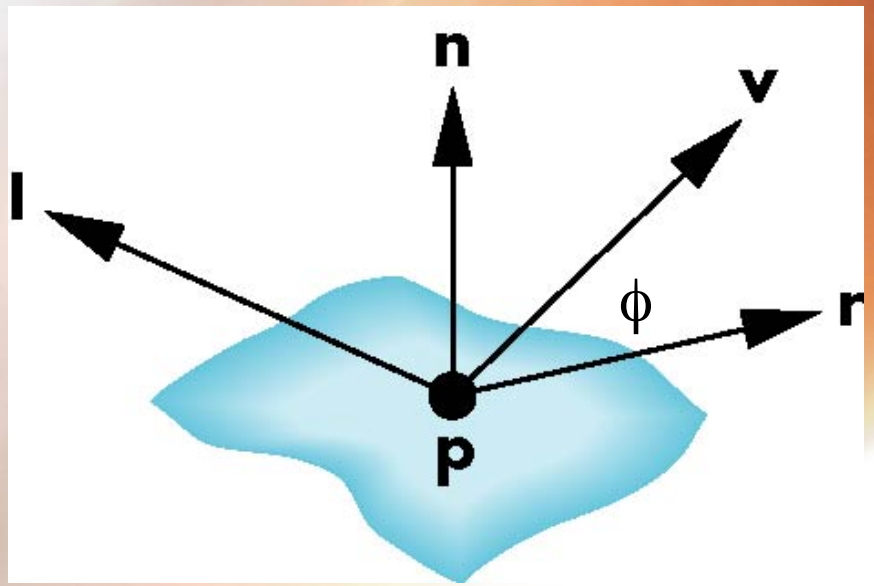
$$I_r \sim k_s I_s \cos^{\alpha}\phi$$

reflected intensity

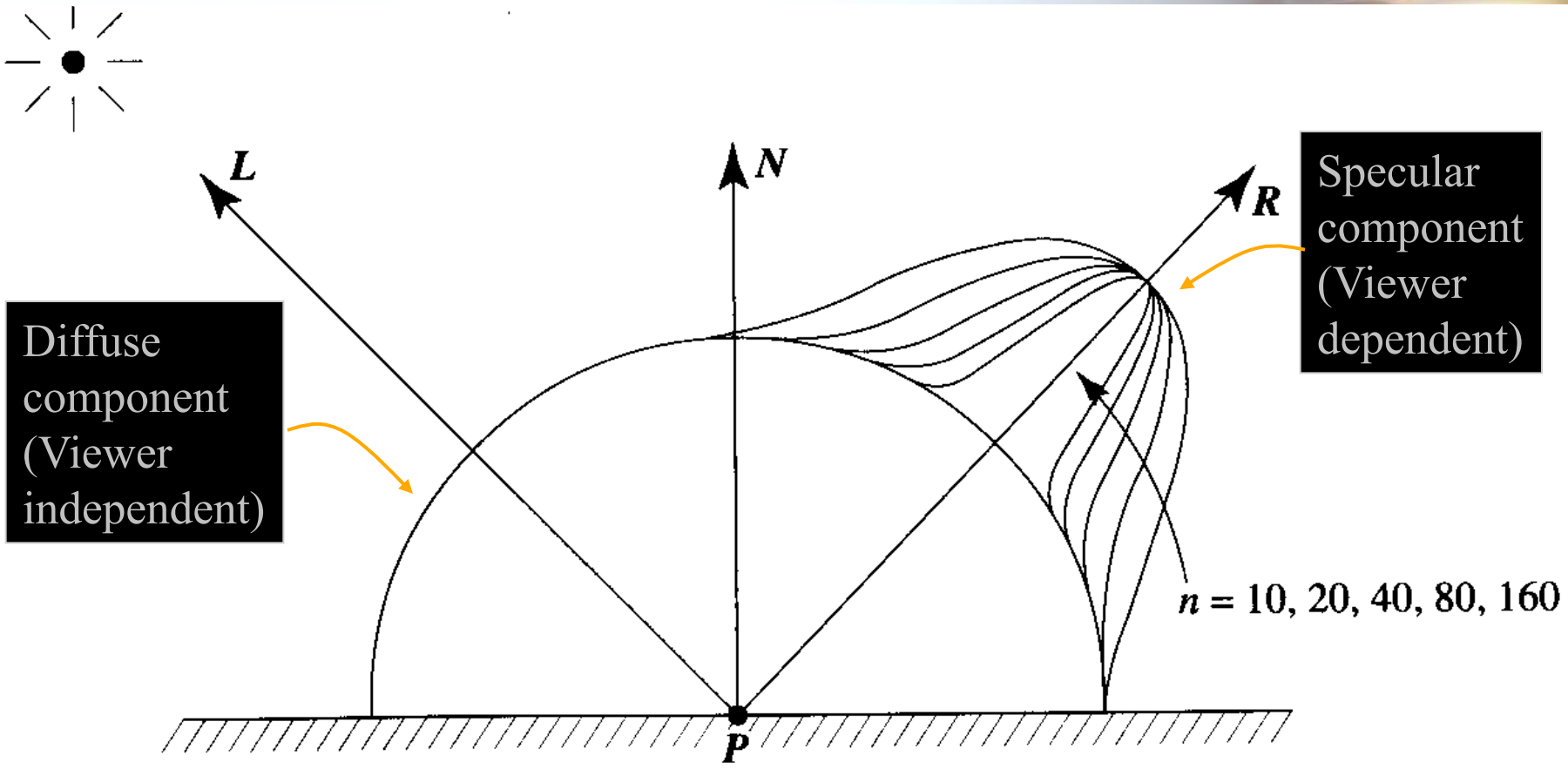
reflection coef

incoming intensity

shininess coef

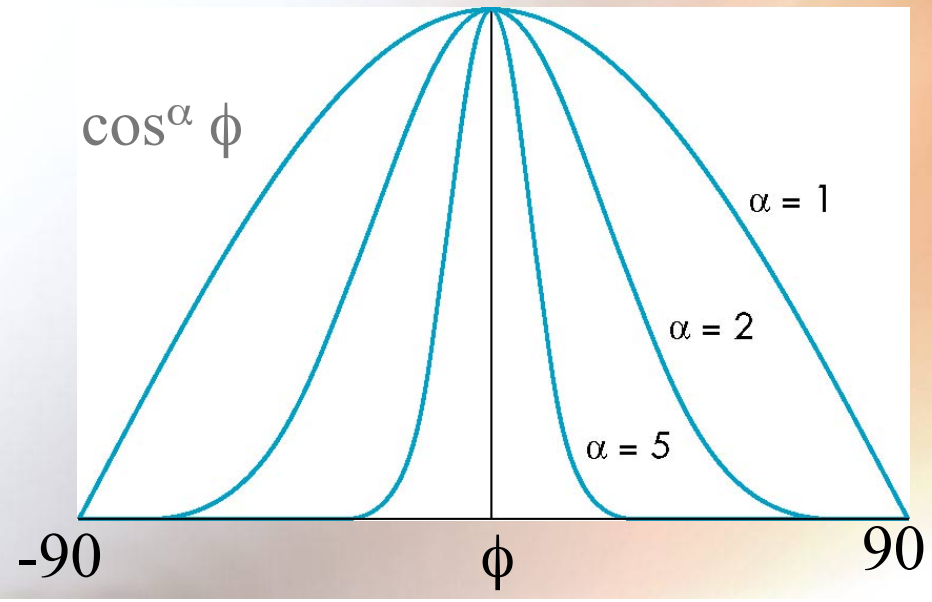


Specular "Bump" Adds to Diffuse Component



The Shininess Coefficient

- As α increases, the reflected light is concentrated over a narrower region.
- Values of α between 100 and 200 correspond to metals , smaller values give surface that look like plastic



Ambient Light

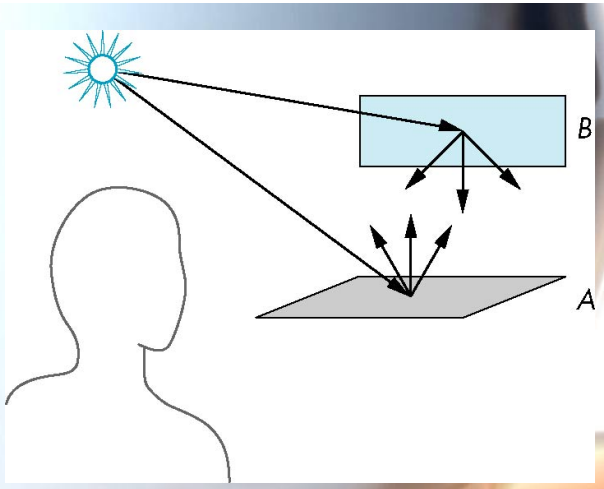
- Ambient light is the result of multiple interactions between (large) light sources and the objects in the environment
- Appears to be uniform all over.
- Add $k_a I_a$ to diffuse and specular terms

$$I_a = k_a L_a$$

reflection coef

amount of ambient light

Distance Terms



- The amount of light from a point source that reaches a surface is inversely proportional to the square of the distance between them
- We can add a factor $\frac{1}{a + bd + cd^2}$ to the diffuse and specular terms
- The constant and linear terms soften the effect of the point source

Light Sources

- In the Phong Model, we add the results from each light source
- Each light source has separate diffuse, specular, and ambient terms
- Separate red, green and blue components
- Hence, 9 coefficients for each point light source
  - $L_{dr}, L_{dg}, L_{db}, L_{ar}, L_{ag}, L_{ab}, L_{sr}, L_{sg}, L_{sb}$

Material Properties

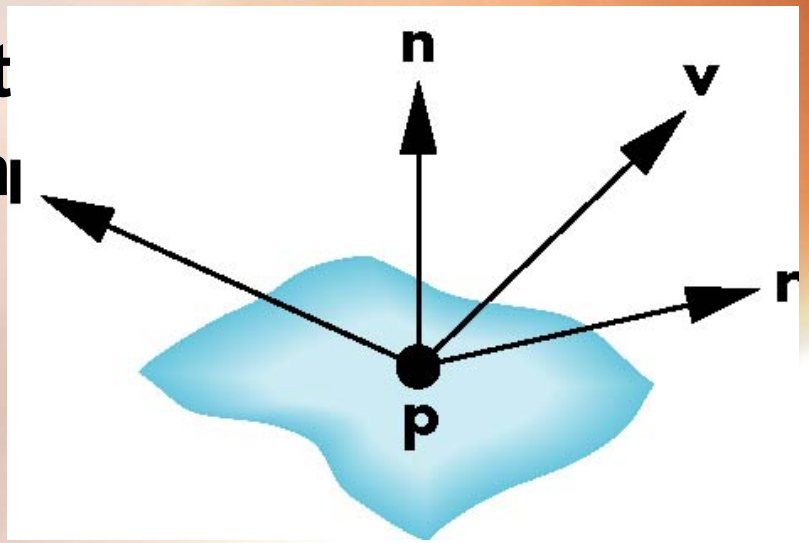
- **Material properties match light source properties**
  - **9 reflection coefficients**
  - $k_{dr}, k_{dg}, k_{db}, k_{ar}, k_{ag}, k_{ab}, k_{sr}, k_{sg}, k_{sb}$
  - **Shininess coefficient** α

Adding up the Components

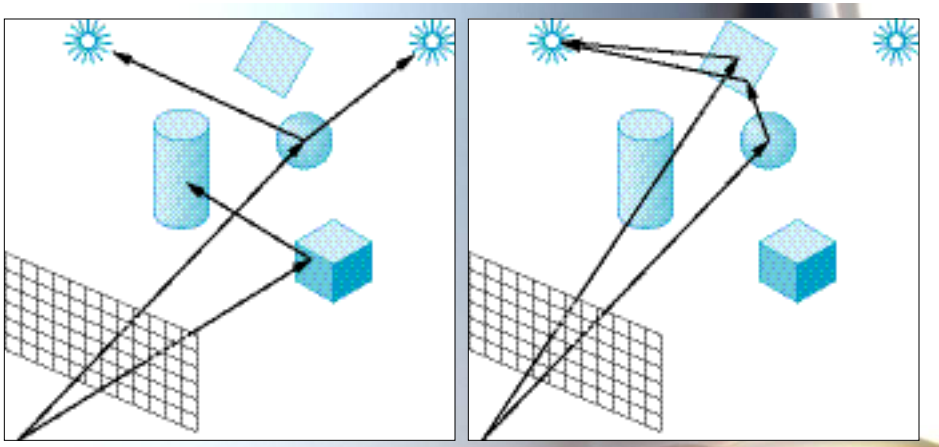
- For each light source and each color component, the Phong model can be written (without the distance terms) as

$$I = k_d L_d (l \cdot n) + k_s L_s (v \cdot r)^{\alpha} + k_a L_a$$

- For each color component we add contributions from all sources



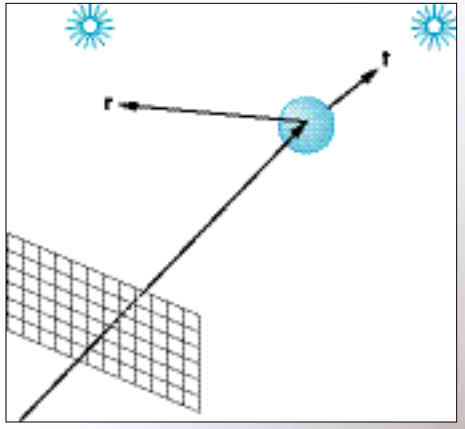
Shadow Ray

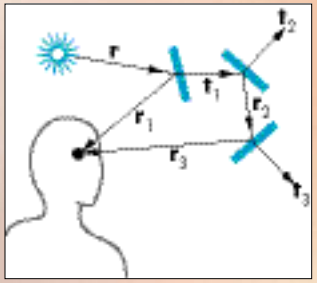


- Trace ray from point of intersection back towards light source.
  - If blocked by another surface before reaching light, then original point is in shadow – only lit with ambient term
- If some surfaces are highly reflective (mirrors)
  - Shadow ray will bounce off and continue
  - This is usually done recursively

Reflection and Transmission

- If a ray hits a surface not in shadow
  - Do reflection calculation according to Phong model
  - Cast ray in direction of perfect reflection
  - Cast ray in direction of transmitted ray





Transmission Vector

$$\frac{\sin(A_i)}{\sin(A_t)} = \frac{\mu_t}{\mu_i} \qquad \left[\sin^2(A_i)\right]\left(\frac{\mu_i}{\mu_t}\right)^2 = \sin^2(A_t)$$

$$\left[1-\cos^2(A_i)\right]\left(\frac{\mu_i}{\mu_t}\right)^2 = 1-\cos^2(A_t)$$

$$\left[1-\cos^2(A_i)\right]\left(\frac{\mu_i}{\mu_t}\right)^2 - 1 = -\cos^2(A_t)$$

$$= -\left[-N \bullet T\right]^2$$

$$= -\left[-N \bullet (\alpha I + \beta N)\right]^2$$

$$= -\left[\alpha(-N \bullet I) + \beta(-N \bullet N)\right]^2$$

$$= -\left[\alpha(-N \bullet I) - \beta\right]^2$$

Transmission Vector (cont.)

$$
\begin{aligned}
1 &= T \bullet T \\
&= (\alpha I + \beta N) \bullet (\alpha I + \beta N) \\
&= \alpha^2 (I \bullet I) + 2\alpha\beta (I \bullet N) + \beta^2 (N \bullet N) \\
&= \alpha^2 + 2\alpha\beta (I \bullet N) + \beta^2
\end{aligned}
$$

We now have two equations and two unknowns!

Some algebra and a look at the geometry gives us:

$$
\alpha = \frac{\mu_i}{\mu_t} \qquad \beta = \left(\frac{\mu_i}{\mu_t}\right)\cos(A_i) - \sqrt{1 + \left(\frac{\mu_i}{\mu_t}\right)^2 \left(\cos^2(A_i) - 1\right)}
$$

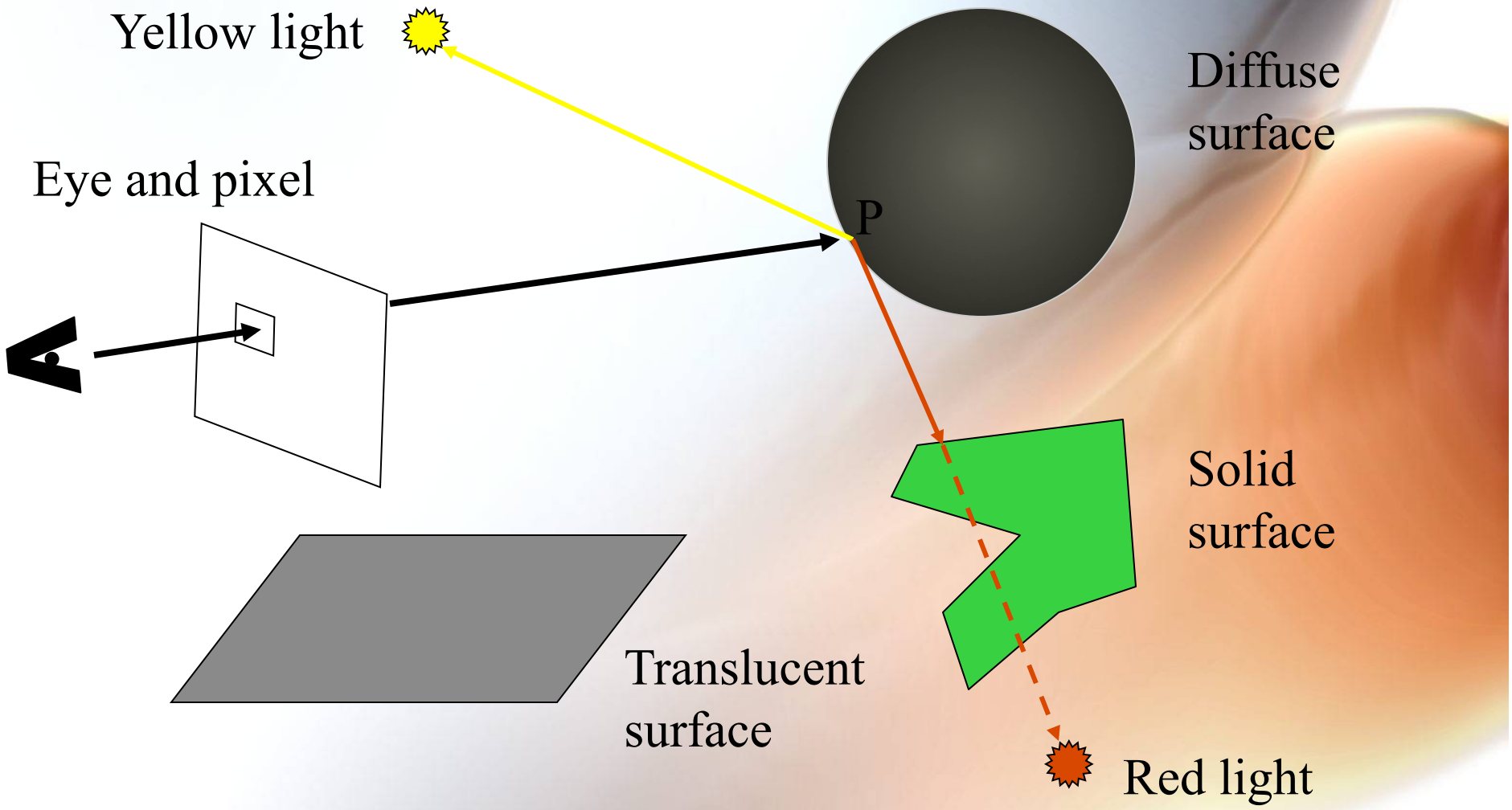
Ray Tracing with Recursion

- Add shadow rays to light sources.
- Stop at diffuse surface, or when maximum number of recursive steps exceeded.
- Color at visible point is established by recursively combining the shades computed at the end of each ray – which depends of course, on what, if anything, it hits.

Recursive Ray Tracing

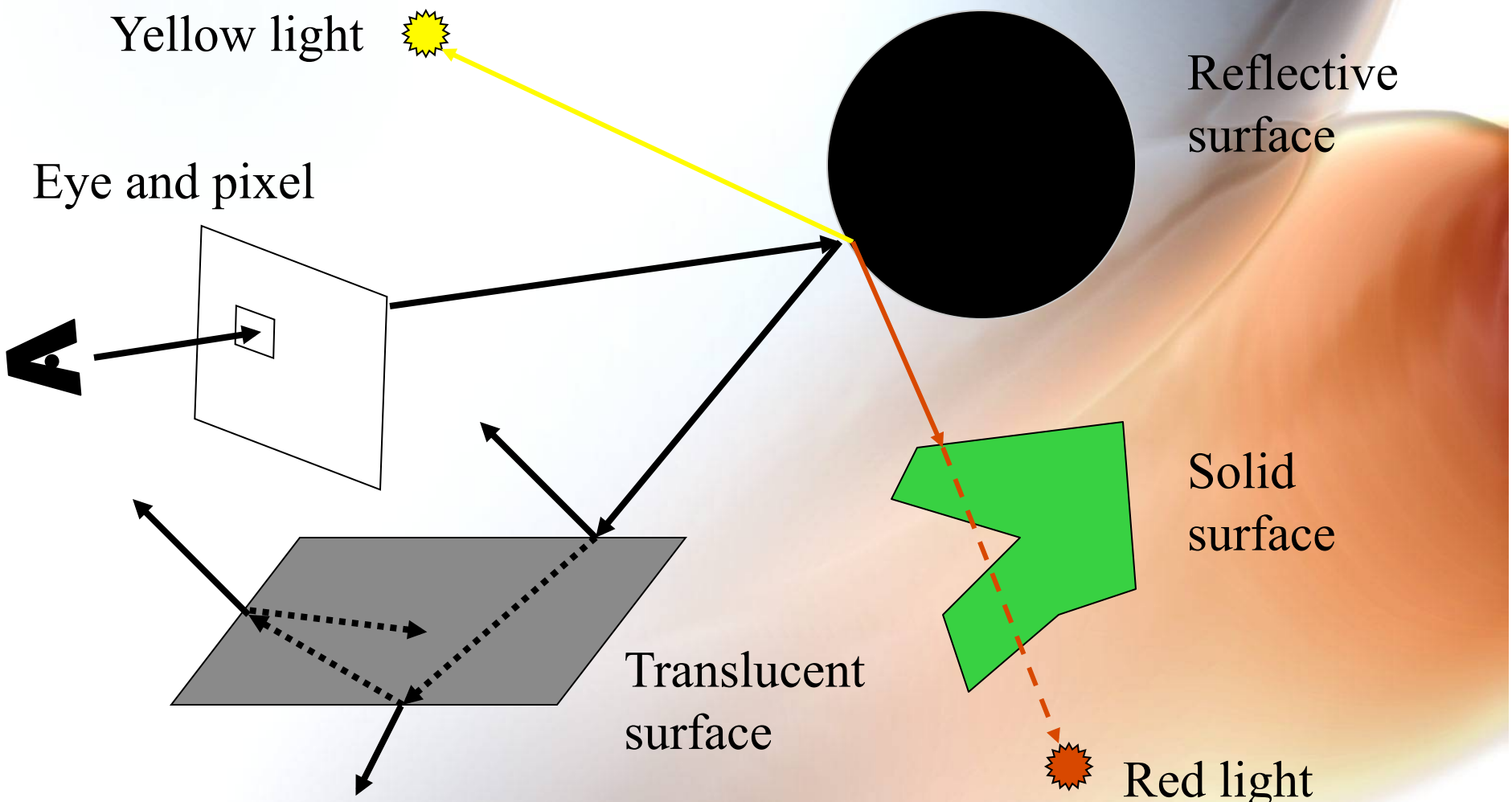
- If an eye ray intersects an object:
  - Cast shadow rays to each light source.
    - diffuse illumination
    - shadows
  - Cast reflection ray, search for object intersection.
    - ambient illumination
  - Cast transmission ray, search for object intersection.
    - Transparency
- For each new reflection and transmission ray, repeat.
  - Until no intersection
  - Until N intersections (recursion depth) ($3 \leq N \leq 7$)

Ray Tracing Diagram (Diffuse)



P is visible; illuminated by yellow light; but not by red light

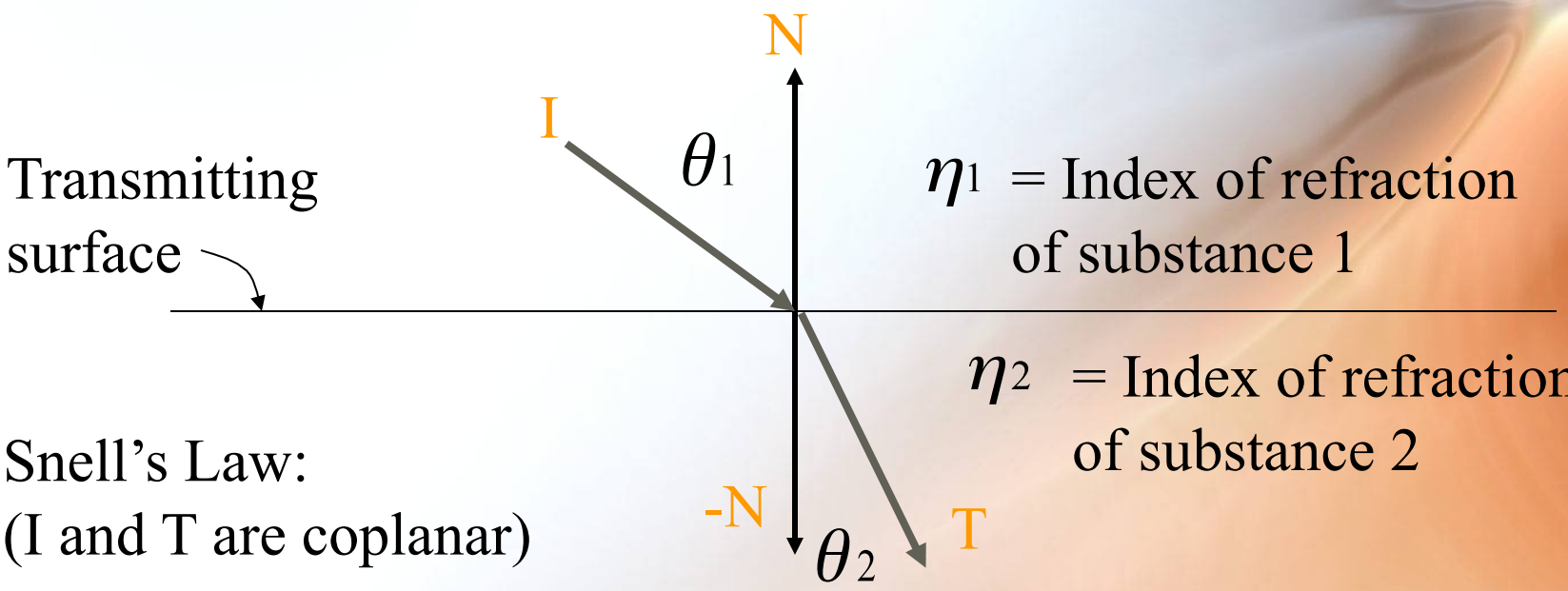
Ray Tracing Diagram (Reflective)



P is visible; illuminated by yellow light; but not by red light

“Typical” Ray Traced Image

Refracted Ray Geometry

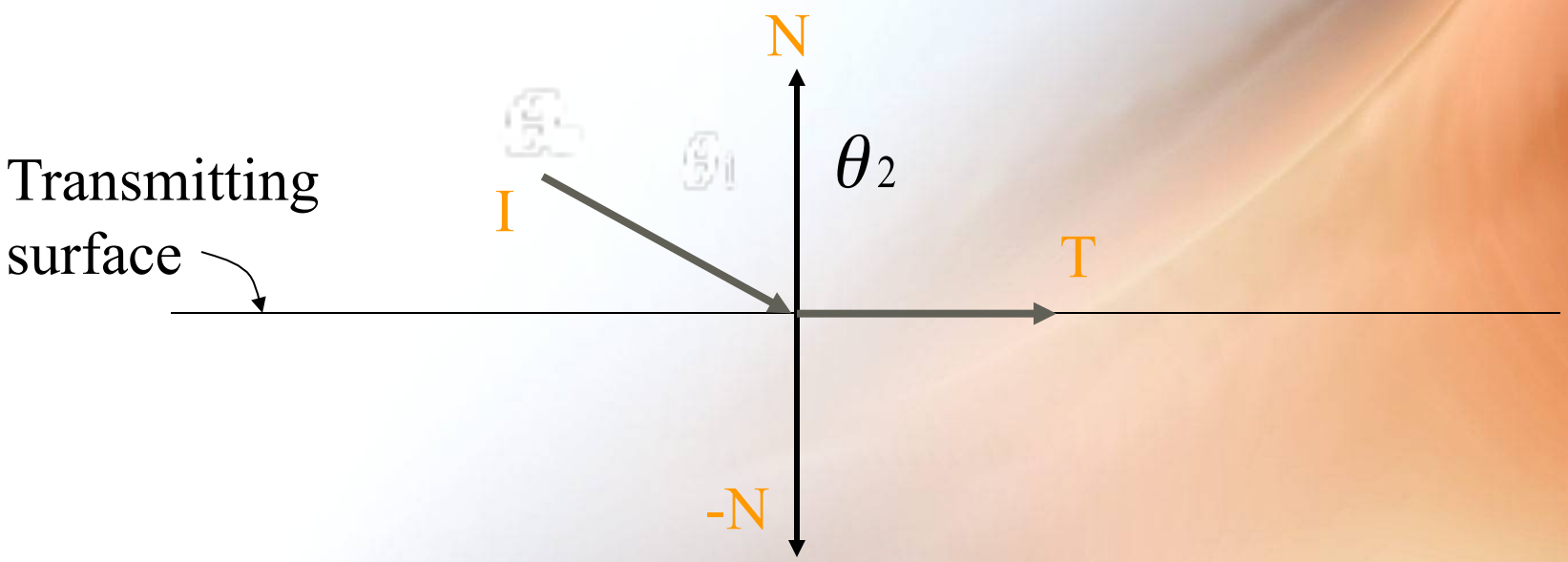


Snell's Law:
(I and T are coplanar)

$$\frac{\sin\theta_1}{\sin\theta_2} = \frac{\eta_1}{\eta_2}$$

Refraction and the Critical Angle

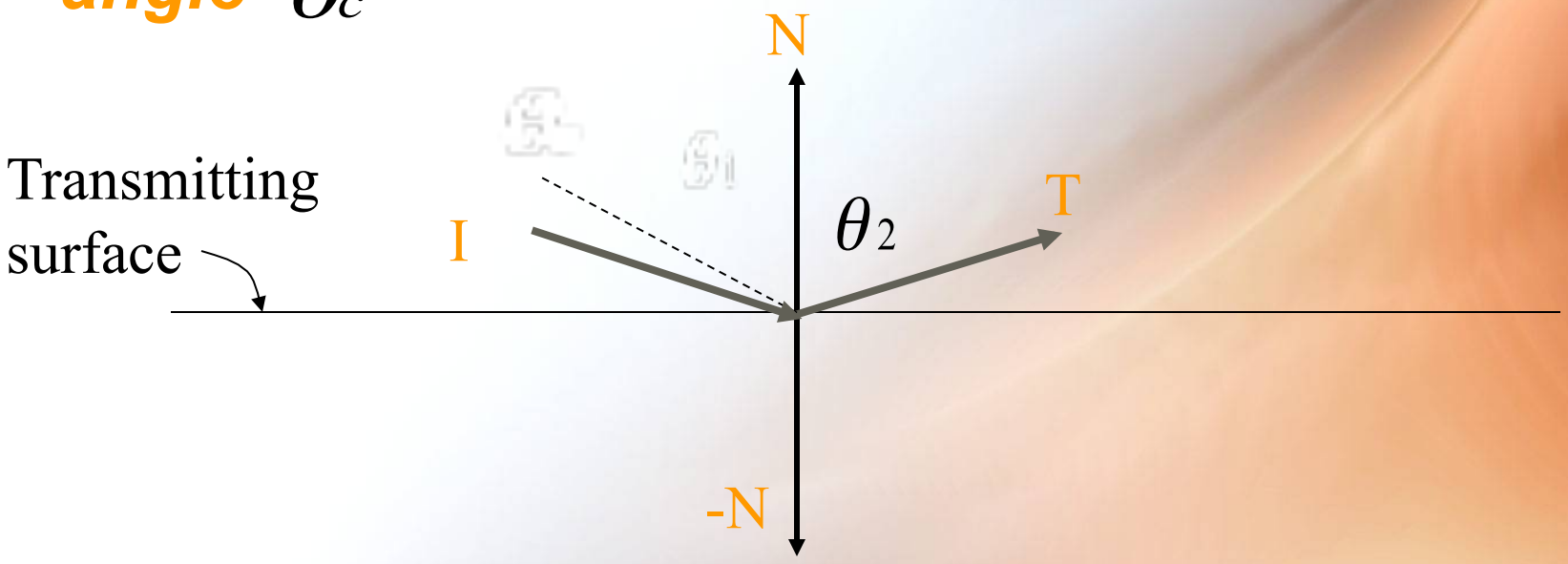
- **Different effect at the *critical angle*** θ_c



$$if \theta_1 = \theta_c \, then \, \theta_2 = \pi / 2$$

Refraction and the Critical Angle

- **Internal reflection effect beyond the *critical angle*** θ_c



$$if \theta_1 > \theta_c \, then \, \theta_2 = \theta_1$$

Index of Refraction

Medium	Index of Refraction
Water	1.33
Ethyl Alcohol	1.36
Carbon Bisulfide	1.63
Air	1.0003
Methylene Iodide	1.74
Fused Quartz	1.46
Glass, Crown	1.52
Glass, Dense Flint	1.66
Sodium Chloride	1.53

Note: These are approximations. The index of refraction actually differs with each wavelength of light.

Reflection and Refraction

Closer to Real Glass

Ray Tracing Cost Considerations

$COST = X\ Y\ 2^a\ (m + 1)\ (2^n - 1)\ P$

- X = number of pixels horizontally
- Y = number of pixels vertically
- a = anti-aliasing super-sampling factor
- m = number of (point) light sources
- n = tree (recursion) depth
- P = number of primitives tested for ray intersections

if X=Y=1000, a=0, m=1, n=10, P=1000 then

COST ~ 2,046,000,000,000 ray-primitive intersection tests!

Problems With Ray Tracing

- Ray/object intersection calculations become expensive with complex objects and more objects.
- Light from other objects (other than pure reflection) is not modeled.
  - Still use the constant ambient light component.
- Shadows often appear too sharp
- Surfaces appear too glossy or smooth
- Images take a long time to render, must recalculate for each new view
- Have to tweak illumination model constants to get desired effect