Introduction to Computer Graphics

11. Advanced Rendering (A)

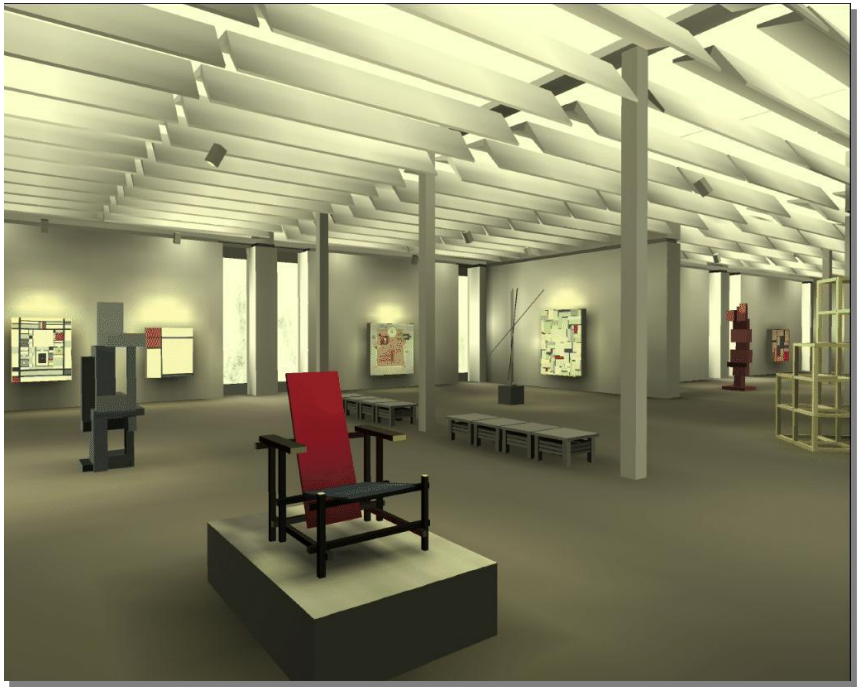
National Chiao Tung Univ, Taiwan
By: I-Chen Lin, Assistant Professor

Textbook: E.Angel, Interactive Computer Graphics, 5th Ed., Addison Wesley
Ref:Hearn and Baker, Computer Graphics, 3rd Ed., Prentice Hall

Outline

- Going beyond pipeline rendering.
- Ray tracing.
- Rendering equation.
- Radiosity

Can We Render Images Like These?

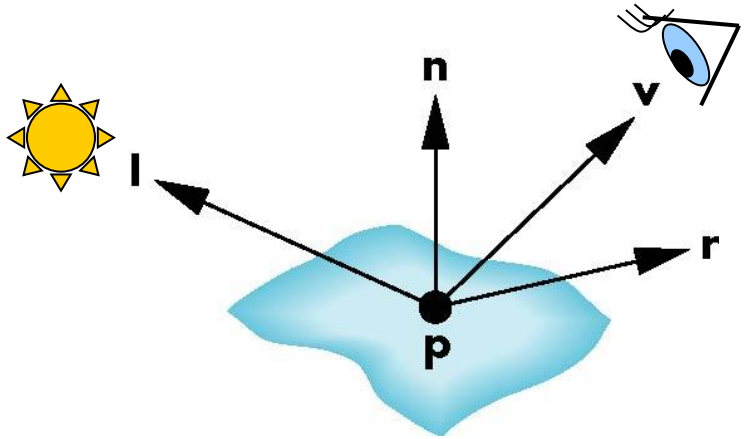


Picture from http://www.graphics.cornell.edu/online/realistic/

Local Illumination

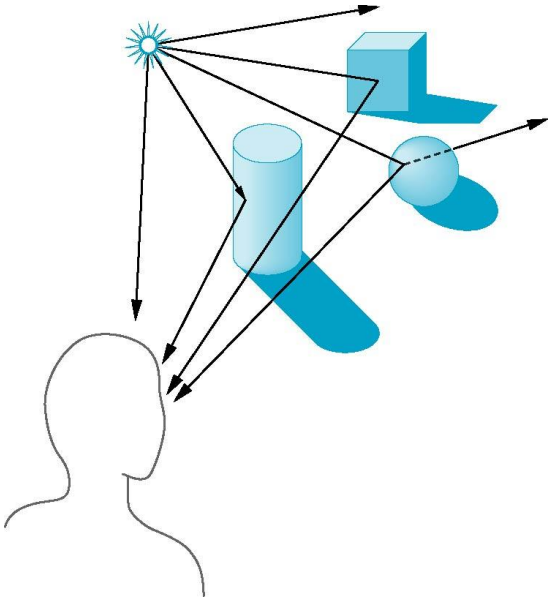
- The Phong model is a local illumination model
  - Shaded color depends only on
    - Surface normal, viewing direction, light direction
    - Ambient, diffuse, and specular reflectances

  - $I = I_{ambient} + I_{diffuse} + I_{specular}$
    $= k_a I_a + k_d I_d(\mathbf{l} \cdot \mathbf{n}) + k_s I_s (\mathbf{v} \cdot \mathbf{r})^a$



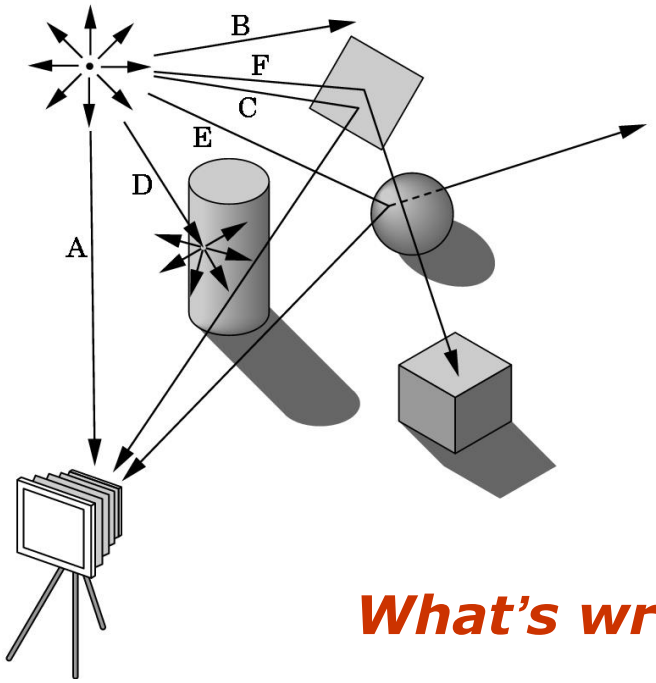
Local Illumination (cont.)

- Don’t take other surfaces into account !
  - Other surfaces cannot block light (no shadows)
  - Omitting light from reflection or refraction of other objects.
- These interactions happen in reality!



Forward Ray Tracing

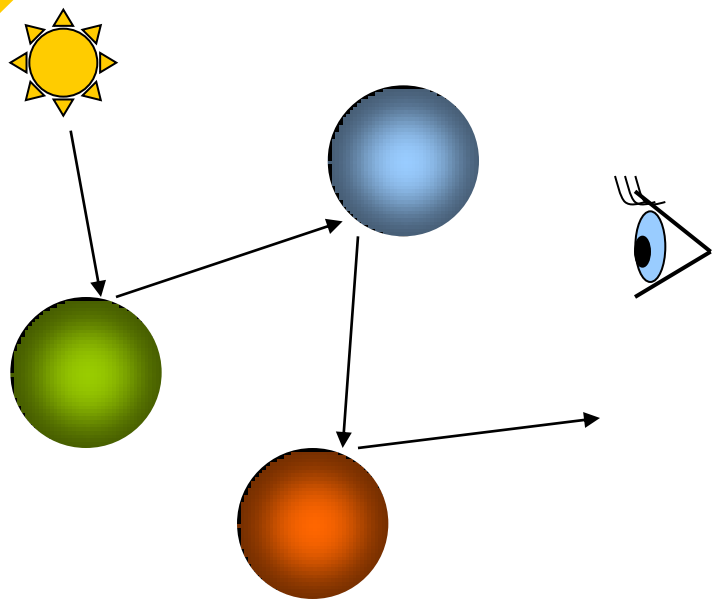
- Rays emanate from light sources and bounce around in the scene.
- Rays that pass through the projection plane and contribute to the final image.



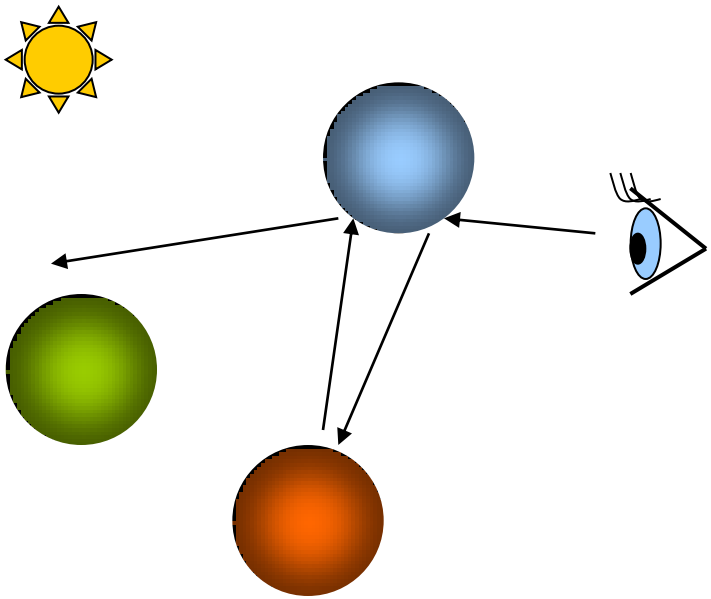
What's wrong with this method?

Forward vs. Backward

Starting at the light



Starting at the eye



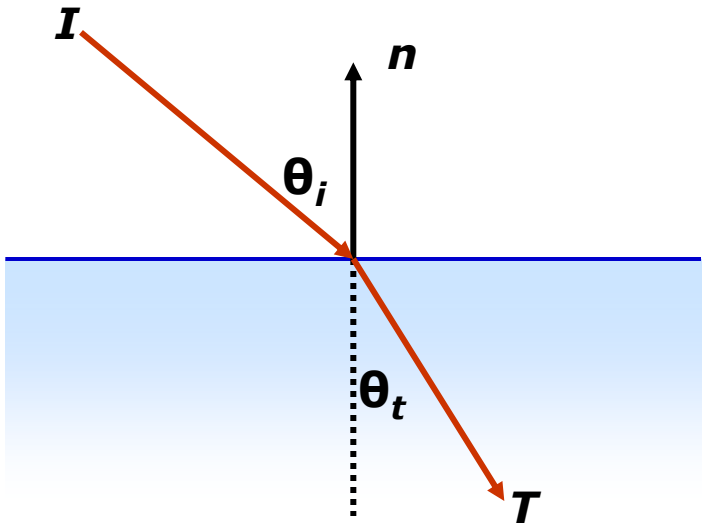
Refraction of Light

- Rays transitioning between materials are bent around normal
  - every material has an index of refraction
- Angles with surface normal obey Snell's Law

$$\frac{\sin\theta_i}{\sin\theta_t} = \eta_{ti} = \frac{\eta_t}{\eta_i}$$

Where η is the 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

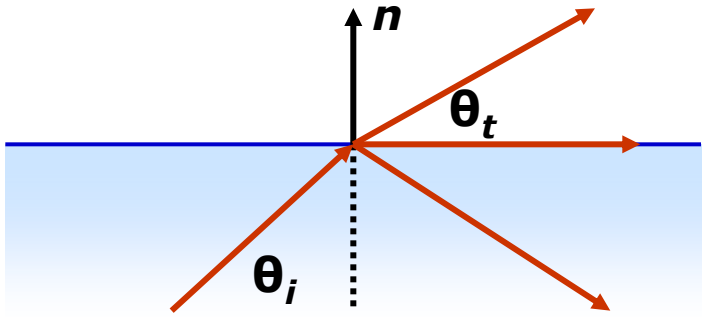


Refraction of Light (cont.)

- When entering material of lower index
  - Ray bends outward from normal
  - What if the angle is more than 90°?
    - Ray is actually reflected off the boundary
    - this is called total internal reflection (like fiber optics)
- Total internal reflection occurs when

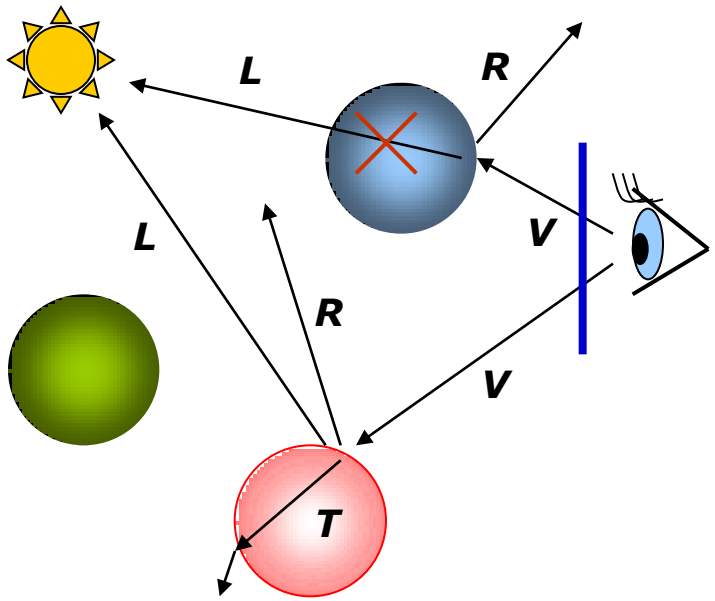
  $\theta_i > \theta_{critical}$, where $\theta_{critical} = \sin^{-1} \frac{\eta_t}{\eta_i}$

  - just need to check for this critical angle
  - if above it, use specular reflection for "transmission"



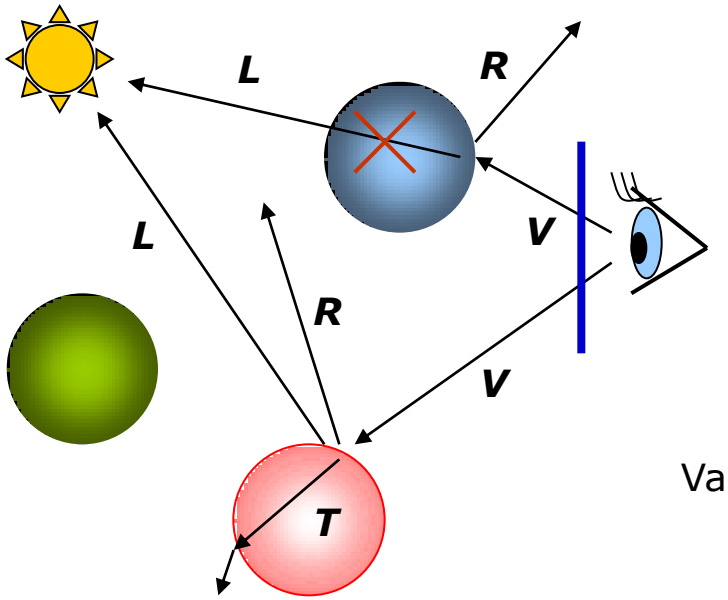
Whitted Ray-tracing

1. For each pixel, trace a ***primary ray*** (eye ray) to the first visible surface.
2. For each intersection trace secondary rays:
   - ***Shadow rays***: in directions L to light sources
   - ***Reflected ray***: in direction R
   - ***Refracted ray*** (transmitted ray): in direction T



Whitted Ray-tracing (cont.)

- Every surface intersection spawns
  - 1 reflected ray
  - 1 transmitted ray
  - 1 shadow ray per light
  - Shaded color of V_i = Valid(L_i) x *PhongModel* + *ReflectedRay* + *TransmittedRay*



Valid(*Li*) = 1, if visible to the light source

0, otherwise

A Simple Ray Tracer

```
void raytrace()
    for all pixels (x,y)
        image(x,y) = trace(compute_eye_ray(x,y))

rgbColor trace(ray r)
    for all surfaces s
        t = compute_intersection(r, s)
        closest_t = MIN(closest_t, t)

        if( hit_an_object )
            return shade(s, r, t)
        else
            return background_color
```

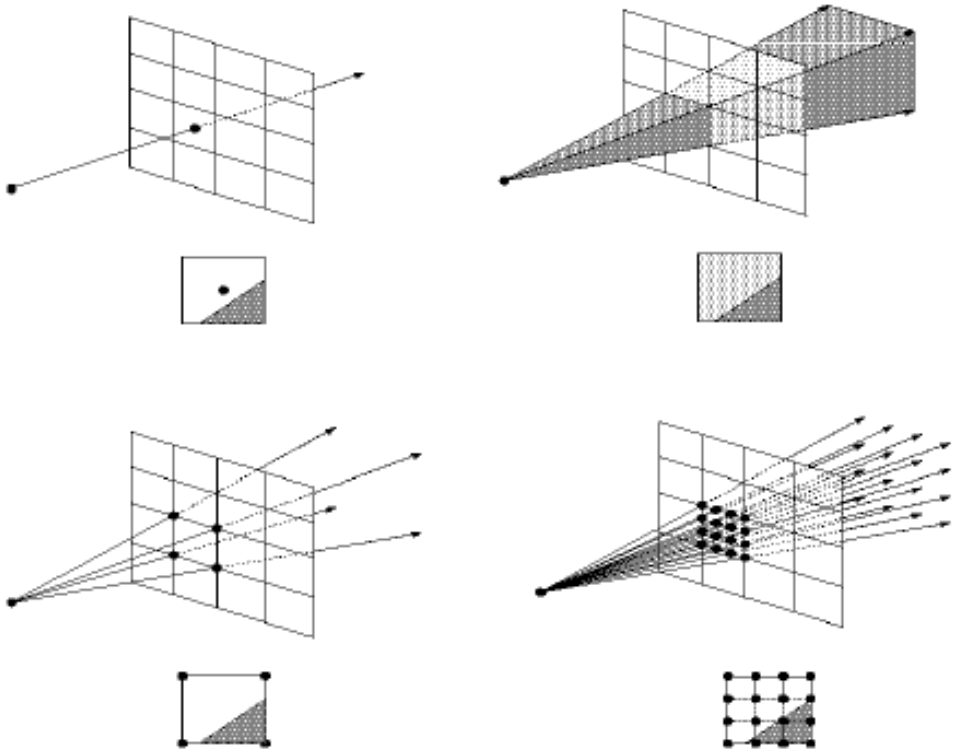
A Simple Ray Tracer (cont.)

```
rgbColor shade(surface s, ray r, double t)
    point x = r(t)
    rgbColor color = black

    for each light source L
        if( closest_hit(shadow_ray(x, L)) >= distance(L) ) {
            color += shade_phong(s, x)
            color += k_specular * trace(reflected_ray(s,r,x))
            color += k_transmit * trace(transmitted_ray(s,r,x))
        }
    return color
```

Supersampling

- Aliasing problems.
- We can approximate the average color of a pixel's area by firing multiple rays and averaging the result.



Efficiency of Ray Tracing

- Consider this example
  - image resolution of 1024x768 = 786,432 pixels
  - 3x3 supersampling = 7 million eye rays
  - recursion depth 5 = 63*7 = 441 million rays
  - each tested against 10,000 polygons
  - 4.4 trillion intersection tests (ignoring shadow rays)

Most of the time is spent in the calculation of intersection !

Efficiency of Ray Tracing (cont.)

- How to efficiently calculate intersections?
  - Efficient representation of an object.
  - Bounding boxes
  - Space partitioning
    - Octree, BSP tree, etc.
  - Distributed ray tracing (non-uniform ray distribution)
  - ……

Efficiency of Ray Tracing (cont.)

- How to utilize more than 1 computer?

Figure from http://www.linuxjournal.com
ILM RackSaver Linux render farm has 1,500 processors in 2003.

- The efficiency of realistic synthetic image rendering (in movie quality)