Classical Ray Tracing

Turner Whitted's 1980 paper popularized ray tracing.

Ray tracing is a simulation using classical ray optics to make images.
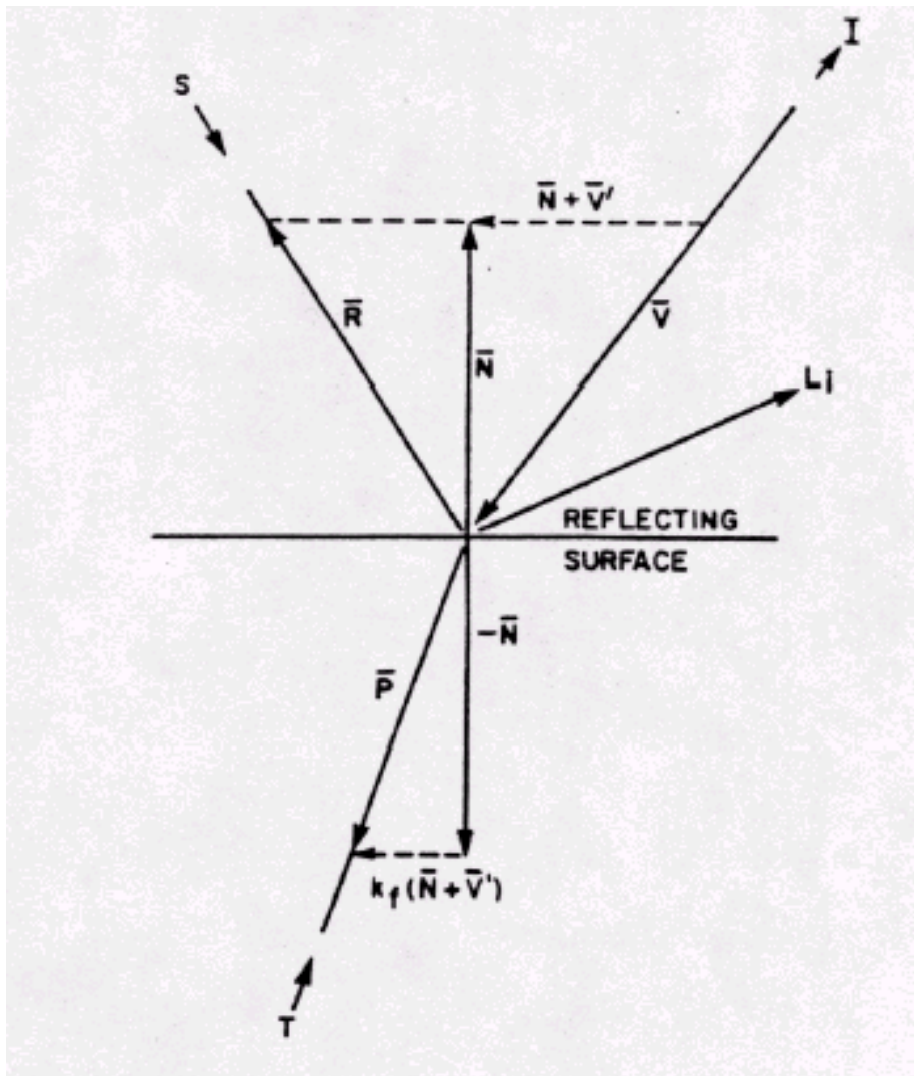
Forward ray tracing
> **Illustrate**
> Why isn't this good? (Actually it has been used – point fwd to Monte Carlo)

Backward ray tracing, from eye.
> Careful when you read papers. Light comes IN reflected ray.

Forward/Backward confusing and used both ways. Maybe use unambiguous terms like light and eye.

Surface model.
    From Turner's paper (draw diagram on board)

$$I = I_a + k_d \sum_{lights} (\overline{N} \bullet \overline{L}_j) + k_s S + k_t T$$

Eye ray

What's a shadow ray?
    Trying to capture whether light affects surface pt.
    Turner attenuated color linearly (hack).
    Don't forget the light color.

Reflection ray
    Well known. Angle of reflection is angle of incidence, and reflection ray is on
    plane with incident and surface normal.
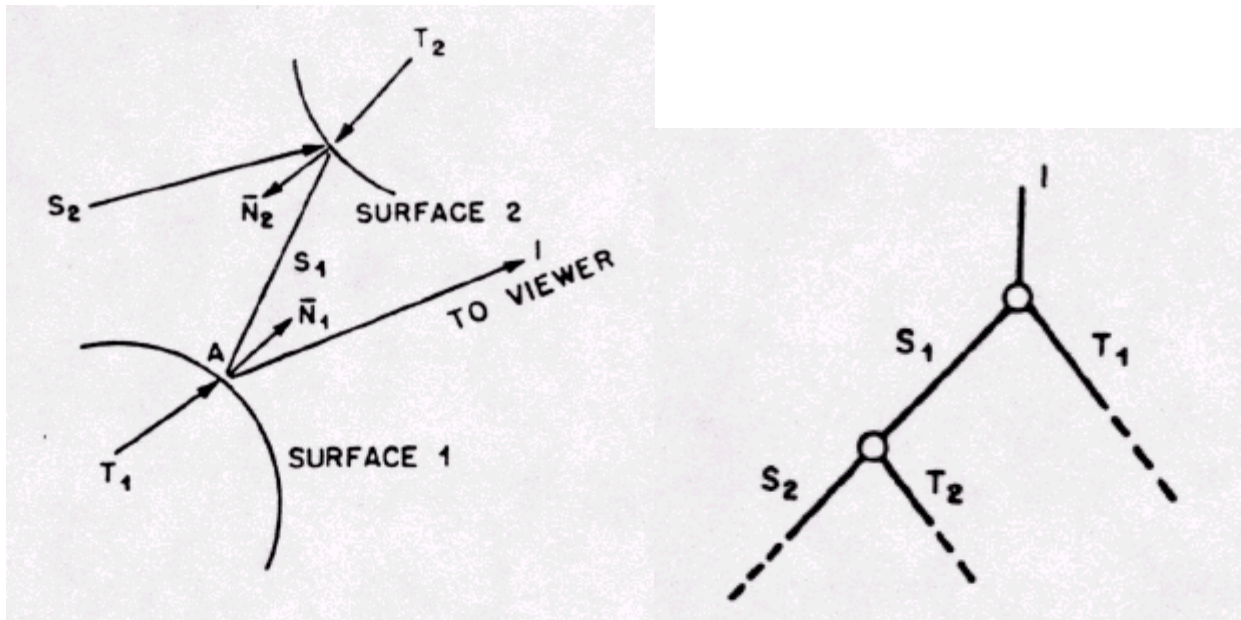
Transparency ray
Refraction
    Snell's law
    $\eta_i \sin\theta_i = \eta_r \sin\theta_r$ where i is the material of incidence and r is the material
    of refraction. **Draw geometry on board.**

    Indices, air 1.00003, water 1.33, glass about 1.5. Make a prism-like material by
    changing with wavelength!

Ray tree example
    Draw a ray tree and illustrate what happens.

How to stop.
>   Ray leaving scene takes on color of background (blue, black).
>   Place model inside a sphere, or maybe a texture mapped room (see Porter).
>   Maybe add yourself as part of the environment.
>   Set a threshold on the amount of contribution. If value is too low, cut off ray.

Ray intersection algorithms.

Haines course notes (placed in reading room)
Slides
Sphere.
Sphere is an easy bounding volume for all primitives.
Haines discusses many optimizations.
1. Check inside of sqrt(). If negative, non-real, so no intersection.
2. Check the t-sqrt() first, because we want the closest intersection. If it's negative (intersection behind ray), check the other possibility.
3. Precompute as much as possible, like radius squared, inverse of radius (for normals calculation).
4. Watch your normals! Wrong if we're inside sphere.
5. A whole other section on optimization, using conditionals to short-cut tests that will fail. I'd program straightforward method first (and save as comment).
6. **Discussion of precision problems.** For example, what if first hit is on sphere but, due to numerical error, t seems to be greater than 0. Sphere could self-shadow. Haines has several remedies, including recognizing that it's the same primitive (used by Whitted), and using a tolerance (if t < some small number, count it as 0).
7. Computing parametric coordinates from the point of intersection. Good for texture cords (images, bump mapping, etc.). The inverse mapping is easy for a sphere because we just have to compute dot products with a sphere up vector and a vector to the "prime meridian". Illustrate on board.
8.

Polygon a little more trouble. Could:
1. See if it crosses plane, compute intersection, see if in polygon.
2. Compute parametric solution (especially if you need the texture coordinates anyway), and see if u & v are from [0, 1).

Don't backface cull, usually. Watch the normals!

Optimization is key. It will take surprisingly long to run these programs.

Start your ray tracer. Simple procedural model, spheres.

Warning – ray tracing is addictive.