Ray Tracing

Reading

Foley *et al.*, 16.12

Optional :

- Glassner, An introduction to Ray Tracing, Academic Press, Chapter 1.
- T. Whitted. “An improved illumination model for shaded display”. *Communications of the ACM*} 23(6), 343-349, 1980.

2

Geometric optics

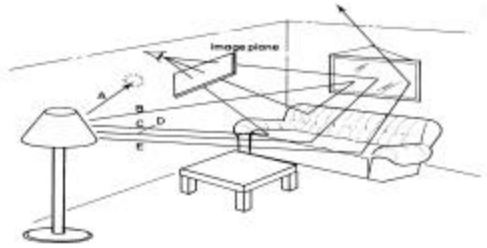
We will take the view of **geometric optics**

- Light is a flow of photons with wavelengths. We'll call these flows ``light rays."
- Light rays travel in straight lines in free space.
- Light rays do not interfere with each other as they cross.
- Light rays obey the laws of reflection and refraction.
- Light rays travel form the light sources to the eye, but the physics is invariant under path reversal (reciprocity).

3

Forward Ray Tracing

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

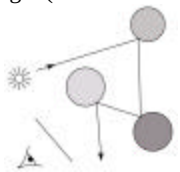


- What’s wrong with this method?

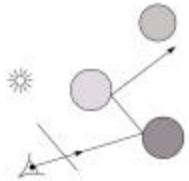
4

Eye vs. Light

- Starting at the light (a.k.a. forward ray tracing, photon tracing)



- Starting at the eye (a.k.a. backward ray tracing)

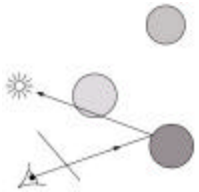


5

Precursors to ray tracing

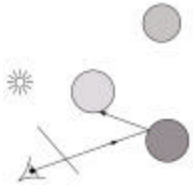
Local illulmination

- Cast one ray, shade according to light



Appel (1968)

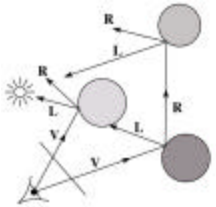
- Cast one eye ray & one ray to light



6

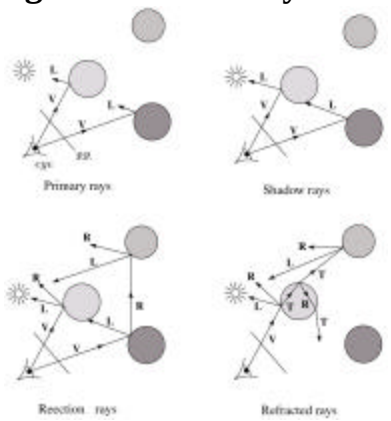
Whitted ray-tracing algorithm

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



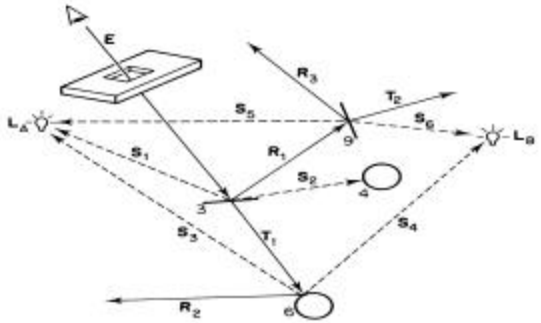
7

Stages of Whitted ray-tracing

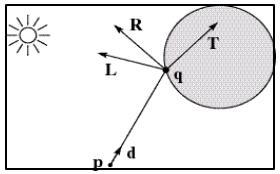


8

Example of Ray Tracing



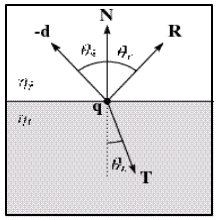
Shading



- A ray is defined by an origin **p** and a unit direction **d** and is parameterized by *t*:

  $\mathbf{p} + t\mathbf{d}$
- Let $I(\mathbf{p}, \mathbf{d})$ be the intensity seen along that ray. Then:
- $I(\mathbf{p}, \mathbf{d}) = I_{\text{direct}} + I_{\text{reflected}} + I_{\text{transmitted}}$
- where
  - I_{direct} is computed from the Phong model
  - $I_{\text{reflected}} = k_r I(\mathbf{q}, \mathbf{R})$
  - $I_{\text{transmitted}} = k_t I(\mathbf{q}, \mathbf{T})$
- Typically, we set $k_r = k_s$ and $k_t = 1 - k_s$.

Reflection and transmission



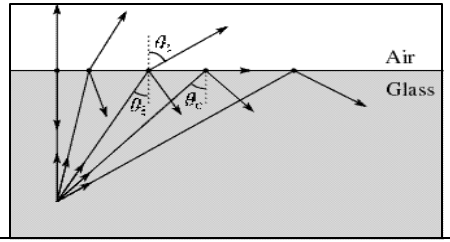
- Law of reflection:

  $q_i = q_r$
- Snell's law of refraction:

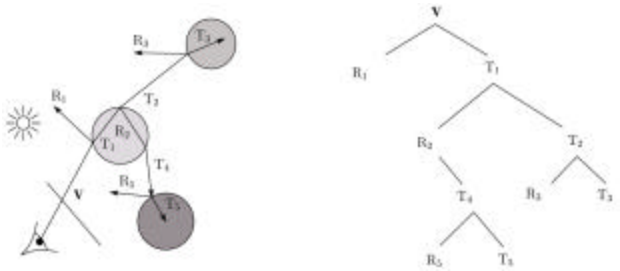
  $h_i \sin q_I = h_t \sin q_t$
- where h_i , h_t are **indices of refraction**.

Total Internal Reflection

- The equation for the angle of refraction can be computed from Snell's law: $h_i \sin q_I = h_t \sin q_t$
- What happens when $h_i > h_t$?
- When q_t is exactly 90°, we say that q_I has achieved the "critical angle" q_c .
- For $q_I > q_c$, *no rays are transmitted*, and only reflection occurs, a phenomenon known as "total internal reflection"



The Ray Tree



Parts of a Ray Tracer

- What major components make up the core of a ray tracer?

Ray-tracing pseudocode

We build a ray traced image by casting rays through each of the pixels.

```
function traceImage (scene):
   for each pixel (i,j) in image
      s = pixelToWorld(i,j)
      p = COP
      d = (s - p)/||s – p||
      I(i,j) = traceRay(scene, p, d)
   end for
end function
```


Ray Tracing Pseudocode

```
color trace( point P0, direction D )
{
        (P,Oi) = intersect( P0, D );
        I = 0
        for each light source l {
                (P', LightObj) = intersect(P, dir(P, l))
                if LightObj = l {
                        I = I + I(l)
                }
        }
        I = I + Obj.Ks * trace(P, R)
        I = I + Obj.Kt * trace(P, T)
        return I
}
```

Controlling Tree Depth

- Ideally, we'd spawn child rays at every object intersection forever, getting a "perfect" color for the primary ray.
- In practice, we need heuristics for bounding the depth of the tree (i.e., recursion depth)
- ?

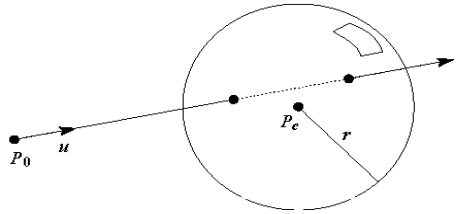
17

Ray-Object Intersection

- Must define different intersection routine for each primitive
- The bottleneck of the ray tracer, so make it fast!
- Most general formulation: find all roots of a function of one variable
- In practice, many optimized intersection tests exist (see Glassner)

18

Ray-Sphere Intersection



- Given a sphere centered at P_c =[0,0,0] with radius r and a ray $P(t) = P_0 + t\boldsymbol{u}$, find the intersection(s) of $P(t)$ with the sphere.
- **Q**: What is the normal to the sphere at a point (x,y,z) on the sphere?

19

Intersecting with transformed geometry

- What if the sphere were transformed by a matrix M (e.g., to make a rotated, translated, ellipsoid)?
- Apply M^{-1} to the ray first and intersect in object (local) coordinates!

20

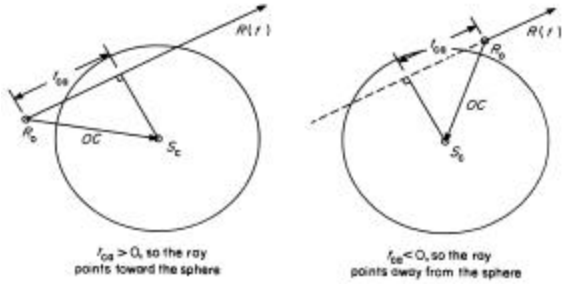
Intersecting with transformed geometry

- The intersected normal is in object (local) coordinates. How do we transform it to world coordinates?

21

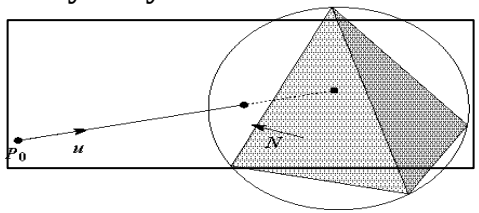
Fast Failure

- We can greatly speed up ray-object intersection by identifying cheap tests that guarantee failure
- Example: if origin of ray is outside sphere and ray points away from sphere, fail immediately.



22

Ray-Polymesh Intersection



1. Use bounding sphere for fast failure
2. Test only front-facing polygons
3. Intersect ray with each polygon's supporting plane
4. use a point-in-polygon test
5. Intersection point is smallest t

23

Numerical Error

- Floating-point roundoff can add up in a ray tracer, and create unwanted artifacts
  - Example: intersection point calculated to be ever-so-slightly *inside* the intersecting object. How does this affect child rays?
- Solutions:
  - Perturb child rays
  - Use global ray epsilon

24