11. Ray Tracing

Reading

Required:

- Watt, Chapter 8.

Recommended:

- A. Glassner. An Introduction to Ray Tracing. Academic Press, 1989.
- T. Whitted. An improved illumination model for shaded display. *Communications of the ACM* 23(6), 343-349, 1980.

What is light?

- Descartes (ca. 1630)
  - Light is a pressure phenomenon in the "plenum"
- Hooke (1665)
  - Light is a rapid vibration – first wave theory
- Newton (1666)
  - Refraction experiment revealed rectilinear propagation
  - Light is a particle (corpuscular theory)
- Young (1801)
  - Two slit experiment
  - Light is a wave
- Maxwell (ca. 1860)
  - Light is an electromagnetic disturbance
- Einstein (1905)
  - Light comes in quanta – photons

Modern theory: wave-particle duality.

Geometric optics

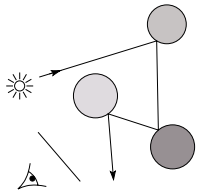
We will take the view of **geometric optics.**

- 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).

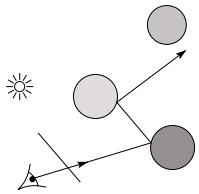
Eye vs. light ray tracing

Where does light begin?

At the light: light ray tracing (a.k.a., forward ray tracing or photon tracing)

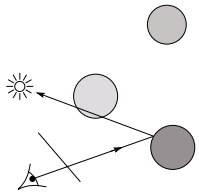


At the eye: eye ray tracing (a.k.a., backward ray tracing)

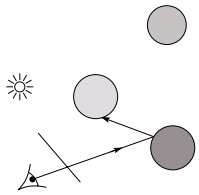


Hybrid methods

- Local illumination
  - Cast one eye ray, then shade according to light

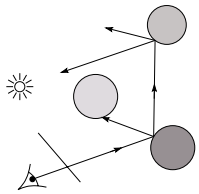


- Appel (1968)
  - Cast one eye ray + one ray to light

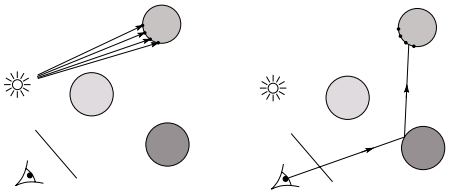


Hybrid methods, cont'd

- Whitted (1980)
  - Eye ray tracing + rays to light
  - Recursive ray tracing

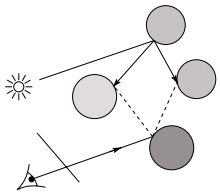


- Heckbert (1990)
  - Eye ray tracing + light ray tracing + light storage on surface

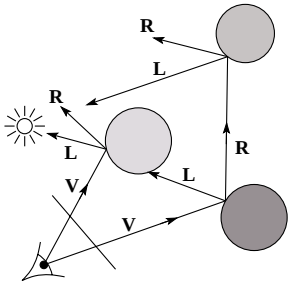


Hybrid methods, cont'd

- Veach (1995)
  - Eye ray tracing + light ray tracing + path connection



Whitted ray-tracing algorithm

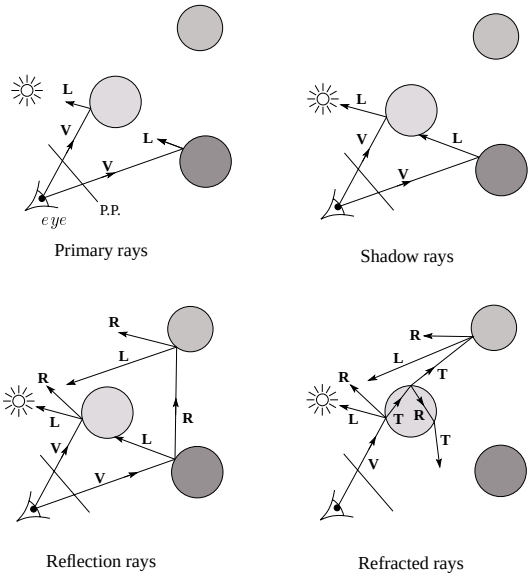


Algorithm:

1. For each pixel, trace a **primary ray** to the first visible surface.
2. For each intersection, trace **secondary rays**:
   - **Shadow rays** in directions $\mathbf{L}_i$ to light sources.
   - **Reflected ray** in direction $\mathbf{R}$.
   - **Refracted ray** or **transmitted ray** in direction $\mathbf{T}$.

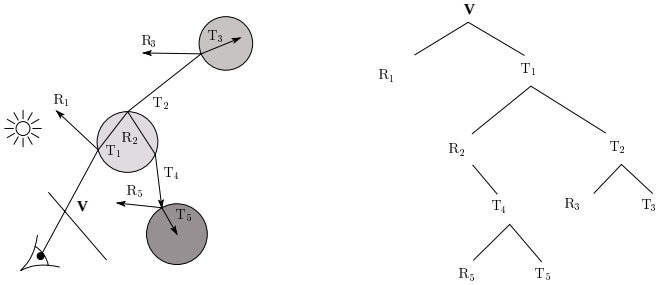
Whitted ray-tracing algorithm, cont'd

Let's look at this in stages:

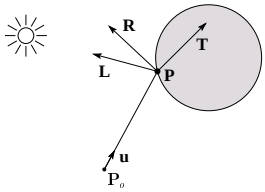


The ray tree

As the recursively traced rays ricochet through the scene, they are added to a "ray tree":



Shading



Let $I(P_o, \mathbf{u})$ be the intensity seen from point P_o along direction $\mathbf{u}$:

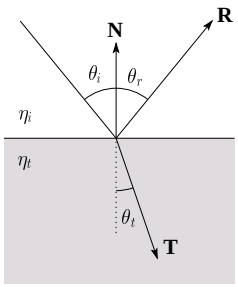
$$I(P_o, \mathbf{u}) \;=\; I_{\text{direct}} \;+\; I_{\text{reflected}} \;+\; I_{\text{transmitted}}$$

where

- I_{direct} is computed from the Phong model (next lecture)
- $I_{\text{reflected}} \;=\; k_{\text{reflected}}\, I(P, \mathbf{R})$
- $I_{\text{transmitted}} \;=\; k_{\text{transmitted}}\, I(P, \mathbf{T})$

A common choice sets $k_{\text{reflected}} = k_s$.

Reflection and transmission



Law of reflection:

$$\theta_i \;=\; \theta_r$$

Snell's law of refraction:

$$\eta_i \sin\theta_i \;=\; \eta_t \sin\theta_t$$

where η_i, η_t are **indices of refraction**.

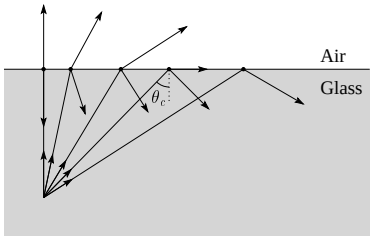
Total Internal Reflection

The equation for the angle of refraction can be computed from Snell's law:

What happens when $\eta_i > \eta_t$?

When θ_t is exactly 90^o, we say that θ_i has achieved the "critical angle," θ_c.

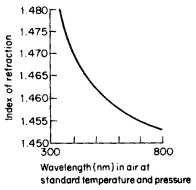
For $\theta_i > \theta_c$, no rays are transmitted, and only reflection occurs, a phenonenon known as "total internal reflection" or TIR.



Refraction and wavelength dependence

The index of refraction varies according to the properties of the material.

Medium	Index of refraction
Vaccum	1
Air	1.0003
Water	1.33
Fused quartz	1.46
Glass, crown	1.52
Glass, dense flint	1.66
Diamond	2.42



Index of refraction variation for fused quartz

The index of refraction varies with wavelength – **dispersion**.

Usually, this effect is ignored in computer graphics.

Fresnel coefficient

The amount of light that is reflected is determined by the Fresnel coefficient. It varies with:

- Angle of incidence
- Wavelength

Dielectrics: light reflected and transmitted must sum to incident light.

Example: Glass is transparent viewed head on, but acts like a mirror at grazing angles.

Metals: no light is transmitted (absorbed instead), can have strong wavelength dependence.

Example: brass gives a yellowed reflection.

Ray-tracing pseudocode

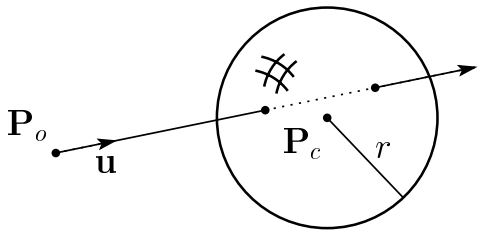
function $RayTrace(P_o, \mathbf{u})$:
  $(P, Obj) \leftarrow RayCast(P_o, \mathbf{u}))$
  $I \leftarrow 0$
  for each light source ℓ **do**:
    $(P', LightObj) \leftarrow RayCast(P, Dir(P, \ell))$
    if $LightObj = \ell$ **then**:
      $I \leftarrow I + \langle \text{diffuse term} \rangle + \langle \text{spec term} \rangle$
    end if
  end for
  $I \leftarrow I + Obj.k_s * RayTrace(P, \mathbf{R})$
  $I \leftarrow I + Obj.k_t * RayTrace(P, \mathbf{T})$
  return I
end function

Terminating recursion

Q: How do you bottom out of recursive ray tracing?

Possibilities:

Intersecting rays with spheres



Given:

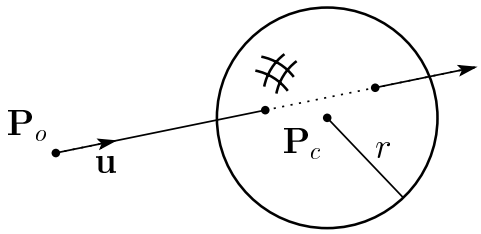
- A ray $\mathbf{P}(s)$ through initial position $\mathbf{P}_0$ in direction $\mathbf{u}$ (a unit vector):

$$\mathbf{P}(s) \;=\; \mathbf{P}_0 \;+\; s\,\mathbf{u}$$

- A sphere S centered at $\mathbf{P}_c$ with radius r

Find: The intersection of $\mathbf{P}(s)$ with S.

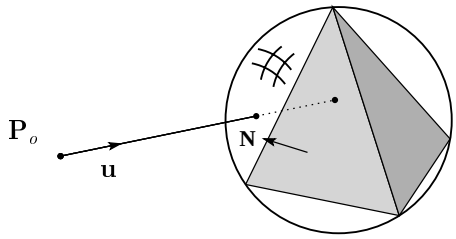
Intersecting rays with spheres



Solution:

Q: What is the normal to the sphere?

Intersecting rays with polyhedra



To intersect a ray with a polyhedron:

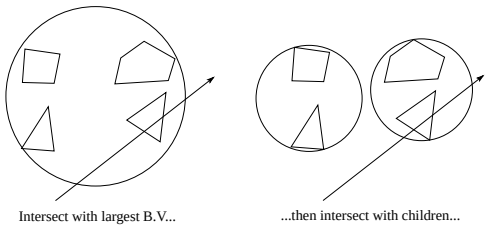
1. Test intersection of ray with bounding sphere.
2. Locate the "front-facing" faces of the polyhedron with
$$\mathbf{u} \cdot \mathbf{N} < 0$$
3. Intersect the ray with each front face's supporting plane.
4. Use a point-in-polygon test to see if the ray is inside the face.
5. Sort intersections according to smallest s.

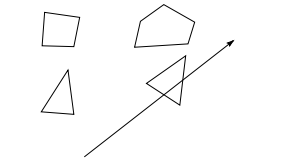
Acceleration: hierarchical bounding volumes

Vanilla ray tracing is really slow!

In practice, some acceleration technique is almost always used.

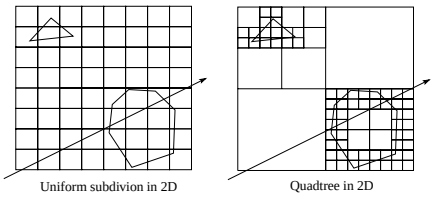
One approach is to use **hierarchical bounding volumes**.

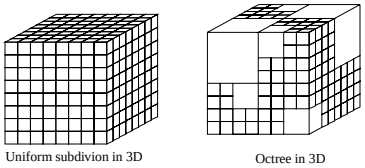




Acceleration: spatial subdivision

Another approach is **spatial subdivision**.





Idea:

- Partition objects spatially.
- Trace ray through voxel array.

Partition can be uniform or adaptive (e.g., octrees).

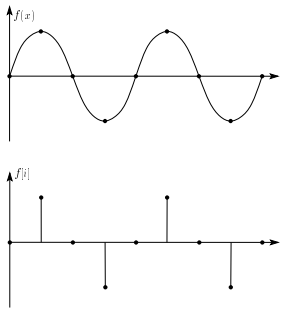
Aliasing

Ray tracing is a form of sampling and can suffer from annoying visual artifacts...

Consider a continuous function $f(x)$. Now sample it at intervals Δ to give $f[i] = f(i\Delta)$.

Q: How well does $f[i]$ approximate $f(x)$?

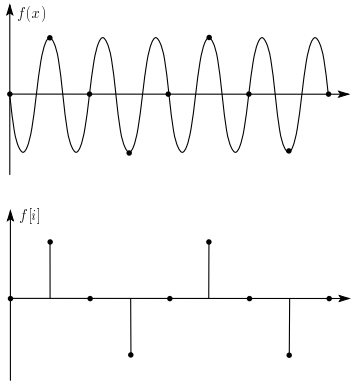
Consider sampling a sinusoid:



In this case, the sinusoid is reasonably well approximated by the samples.

Aliasing, cont'd

Now consider sampling a higher frequency sinusoid

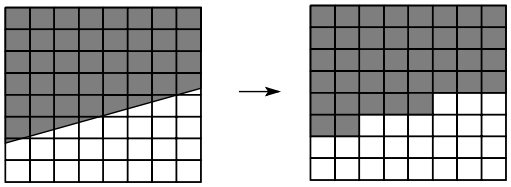


We get the exact same samples, so we seem to be approximating the first lower frequency sinusoid again.

We say that, after sampling, the higher frequency sinusoid has taken on a new "alias", i.e., changed its identity to be a lower frequency sinusoid.

Practical examples of aliasing

Drawing a polygon into the frame buffer:



Temporal aliasing:

Anti-aliasing

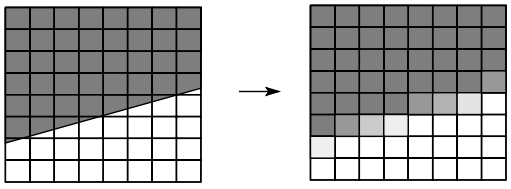
Q: How do we avoid aliasing artifacts?

1. Sampling:

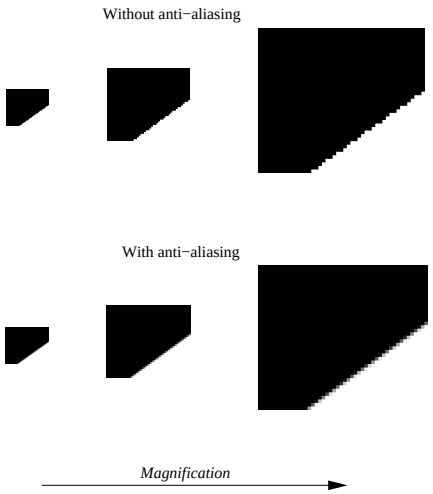
2. Filtering:

3. Combination:

Example - polygon revisited:



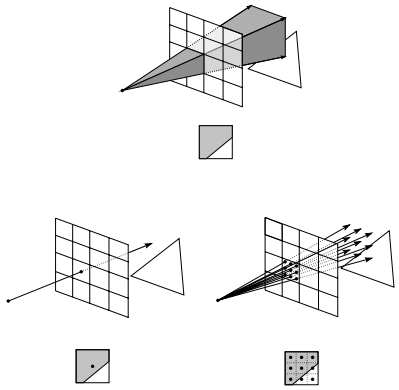
Polygon anti-aliasing



Q:What about temporal aliasing?

Antialiasing in a ray tracer

We would like to compute the average intensity in the neighborhood of each pixel.



When casting one ray per pixel, we are likely to have aliasing artifacts.

To improve matters, we can cast more than one ray per pixel and average the result.

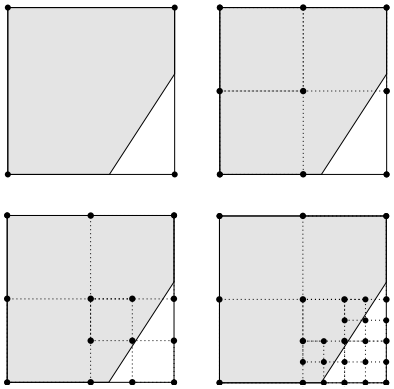
A.k.a., **super-sampling and averaging down**.

Antialiasing by adaptive sampling

Casting many rays per pixel can be unnecessarily costly.

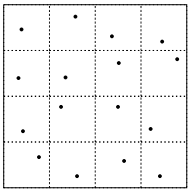
For example, if their are no rapid changes in intensity at the pixel, maybe only a few samples are needed.

Solution: **adaptive sampling**.



Q: When do we decide to cast more rays in a particular area?

Distribution ray tracing

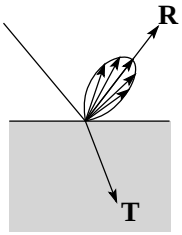


Idea:

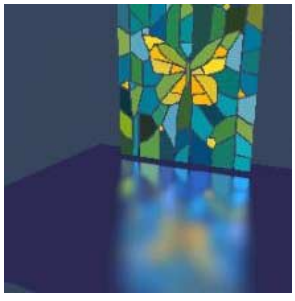
- Use non-uniform (jittered) samples.
- Replaces aliasing artifacts with noise.
- Provides additional effects if we distribute rays in other dimensions:
  - Reflection and refractions
  - Light source area
  - Camera lens area
  - Time

Originally called "distributed ray tracing," but we will call it **distribution ray tracing** so as not to confuse with parallel computing.

Distribution ray tracing, cont.



Distributing rays over reflection and/or refraction directions gives:

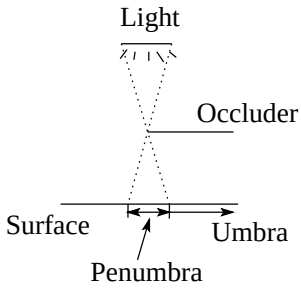


Distribution ray tracing, cont.

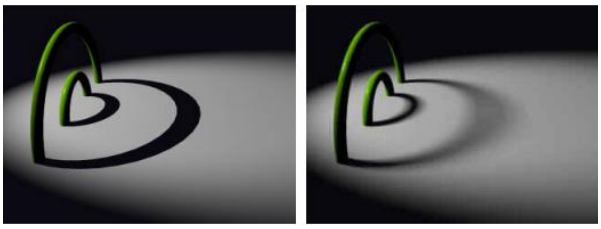
Operationally:

1. Partition the reflection directions into 16 angular regions. Assign each region a unique number between 1 and 16.
2. Partition each pixel into 16 regions. Assign each region a unique number between 1 and 16.
3. Select sub-pixel $m = 1$.
4. Cast a ray through sub-pixel m, jittered within its region.
5. After finding the first intersection, reflect into direction region m, jittered within that region. Repeat for future reflections.
6. Add result to current pixel total.
7. Increment m and, if $m \leq 16$, go to 4.
8. Divide by 16, store the result, choose the next pixel and go to 3.

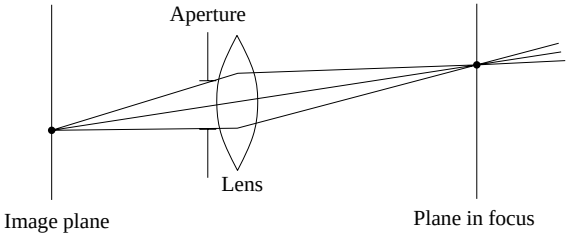
Distribution ray tracing, cont.



Distributing rays over light source area gives:

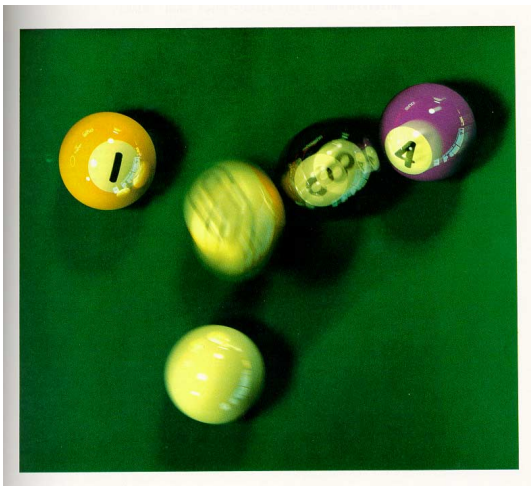


Distribution ray tracing, cont.



Distributing rays over a finite aperture gives:

Distribution ray tracing, cont.



Distributing rays over time gives:

Summary

What to take home from this lecture:

1. The meanings of all the boldfaced terms.
2. Enough to implement basic recursive ray tracing.
3. How reflection and transmission directions are computed.
4. How ray–object intersection tests are performed.
5. Basic acceleration strategies.
6. An intuition for what aliasing is.
7. How to reduce aliasing artifacts in a ray tracer
8. Concept of distribution ray tracing.