- Ray Tracing
- Ray casting
  - From: Appel '68; Gold & Nagel '68
  - Problem
    - Polygons undersample the displayed image
    - Images don't have shadows, reflections, refractions
  - Basic idea
    - So sample once for every image pixel
  - General approach
    - For each screen pixel
      - Find the ray from the eye through the pixel
      - For each object in the scene
        - If the ray intersects the object, and is closest yet
          - Record intersection and object
      - Find color for closest intersection
    - This simple algorithm is a visible surface alg, called ray casting
  - Complexity
    - Worst case: object x pixel
      - A ray/object intersection for every object and pixel combo
  - Example intersections
    - Nice for mathematical surfaces, e.g. spheres
      - Take the parametric eq of line, plug into sphere equation
      - $(x - a)^2 + (y - b)^2 + (z - c)^2 = r^2$
      - Simplify: $x^2 - 2ax + a^2$ ...
      - Plug in: $(x_0 + t\,dx)^2 + \ldots$
      - Result is quadratic in t, find roots
        - If no roots, no intersection
        - If one root, ray grazes sphere
        - If two, ray pierces sphere
          - smallest t closest
      - Find location by plugging in intersection value of t
      - Find normal by drawing line to center sphere from intersect
    - Harder case is polys
      - Use similar approach to solve for point on poly plane
      - Result is a constant ratio
        - If denominator is zero, ray/plane are parallel
      - Can use barycentric coordinates to test for containment in triangle
        - These coords are based on a set of points, rather than vectors
        - For triangles, we have:
          - Three triangle vertices A, B, C
          - Three weights a_1, a_2, a_3
            - Constrained to sum to one
          - Basic equation
            - $P = a_1 A + a_2 B + a_3 C$
          - With this system
            - Can address any pt on triangle's plane
            - If any a is > 1 or < 0, pt outside triangle
            - If one a is 0, on triangle edge
            - If two as are 0, on vertex
      - Convert projected point to barycentric cords
        - Leverage constraint:
          - $a_3 = 1 - a_1 - a_2$
          - $P = a_1 A + a_2 B + C(1 - a_1 - a_2)$
          - $P = (A-C)a_1 + (B-C)a_2 + C$
        - Need two equations, for two unknowns
          - Choose two coords, e.g.
          - If 2 0's in poly normal, must choose those coords
          - $P_x = (A_x-C_x)a_1 + (B_x-C_x)a_2 + C_x$
          - $P_y = (A_y-C_y)a_1 + (B_y-C_y)a_2 + C_y$
      - Intersection if
        - a_1, a_2 and a_3 are > 0 and < 1
- Optimizations

  - Speedups
    - Can use bounding volumes
      - E.g. intersect with a box, if hit the box, intersect with the object
    - Can use object hierarchies (in model space) to increase speed
      - E.g. Octrees -- each box contains eight smaller ones
        - Check smaller ones only if hit larger one
    - Can use image space hierarchies (in screen) to increase speed
      - E.g. Beam tracing, make pixels bigger
      - E.g. Quadtrees
  - Quality improvements (Whitted '80)
    - Adaptive supersampling
    - Sample at corner of pixels
    - If differences of four corners at a pixel are not too large
      - Use the average
    - Otherwise, subsample with new smaller pixels and corners
- Illumination extensions
  - Shadows
    - For each light source
      - Fire an extra ray from intersection to light source
      - If intersection with object, in shadow, and ignore spec/diff terms
  - Reflections
    - Fire a reflected ray along R
    - Add a new term: ... + Ks Ir (once for r, g, b)
      - Where Ir is the intensity of the light found at reflected intersection
    - Reflect until new light is minimal, or just X times
  - Refractions
    - Fire an additional ray through the object, according to Snell's law
    - Need info about the indices of refraction for the material
    - Add a new term: ... + Kt It (for each r, g, b)
      - Where It is the intensity of the light found at the transmission intersection
    - Snell's law:
      - Image
        - Normal N
        - Ray I strikes surface
        - Ray T is refracted vector
        - ThetaI, angle of incidence, between I and N
        - ThetaT, angle of refraction, between N and T
      - sin (thetaI) / sin (thetaT) = muT / muI
        - Mu's are indices of refraction
          - Depends on material, wavelength, temperature
        - Let muR equal *inverse* of either side of term (mu's faster)
      - The vector of transmission T:
        - $\underline{T}$ = sin (thetaT) $\underline{M}$ - cos (thetaT) $\underline{N}$
          - where M is tangent to surface, projection of T on surface
      - With Snell's law plugged in, after various additional algebra/trig:
        - $\underline{T}$ = -muR $\underline{I}$ + $\underline{N}$ (muR($\underline{N}$*$\underline{I}$) - sqrt(1-muR^2(1-($\underline{N}$*$\underline{I}$)^2)))
    - Total internal reflection
      - When angle of refraction is larger than 90 degrees, light is in fact reflected
      - E.g. you press your hand against glass, view from other side
        - If viewing angle is right, see only part of hand on glass
      - This happens when sqrt above is imaginary
- Frameless ray tracing
  - Frames
    - Most moving imagery is framed
      - E.g. movies, TV
    - A series of complete images, or frames
      - Each represents a complete spatial sample at one instant
      - Every pixel represents the same instant
  - Double buffering
    - Problem:
      - In interactive rendering, when render to frame
      - Screen shows part of old, part of new image

- Solution
  - Use two frame buffers
  - One for currently displayed image
  - One for image being rendered
  - Swap them when new frame is ready
- Frameless
  - Frameless ray tracing discards this notion
  - Renders pixels as soon as calculated
  - Stochastically chooses next pixel to render
  - End effect is a fuzzy look -- objects moving fast are blurry
  - Critique:
    - +: More continuous sampling in time
    - +: Great reduction in delay
      - Double buffering makes you wait for:
        - Rendering time of displayed frame
        - Rendering time of next frame
    - +: intuitive motion blur
    - -: things are blurry spatially
    - -: things are blurry temporally – lots of times on one screen

Critique

- +: shadows, spec reflection, refraction
- +: higher quality image
- -: slower: intersections, depth
- -: no diffuse interreflection
- -: area, anisotropic light sources are hard
- -: no soft shadows
- -: no caustics, diffraction