Raytracing Implementation

- Raytracing breaks down into two tasks:
  - Constructing the rays to cast
  - Intersecting rays with geometry
- The former problem is simple vector arithmetic
- The intersection problem arises in many areas of computer graphics
  - Collision detection
  - Other rendering algorithms
- Intersection is essentially root finding (as we will see)
  - Any root finding technique can be applied

Constructing Rays

- Define rays by an initial point and a direction: $\mathbf{x}(t)=\mathbf{x}_0+t\mathbf{d}$
- Eye rays: Rays from the eye through a pixel
  - Construct using the eye location and the pixel's location on the image plane. $\mathbf{x}_0=\mathbf{eye}$
- Shadow rays: Rays from a point on a surface to the light.
  - $\mathbf{x}_0$=point on surface
- Reflection rays: Rays from a point on a surface in the reflection direction
  - Construct using laws of reflection. $\mathbf{x}_0$=surface point
- Transmitted rays: Rays from a point on a transparent surface through the surface
  - Construct using laws of refraction. $\mathbf{x}_0$=surface point

Ray-Object Intersections

- Aim: Find the parameter value, t_i, at which the ray first meets object i
- Transform the ray into the object's local coordinate system
  - Makes ray-object intersections generic: ray-sphere, ray-plane, …
- Write the surface of the object implicitly: $f(\mathbf{x})=0$
  - Unit sphere at the origin is $\mathbf{x}^2-1=0$
  - Plane with normal $\mathbf{n}$ passing through origin is: $\mathbf{n}\bullet\mathbf{x}=0$
- Put the ray equation in for $\mathbf{x}$
  - Result is an equation of the form $f(t)=0$ where we want t
  - Now it's just root finding