Ray Tracing
shallow depth of field, area light sources, diffuse interreflection
up to 16 reflections per ray
Basic Algorithm

for each pixel

1. **cast view ray**: compute view ray from camera through pixel into scene
2. **intersect**: find intersection of ray with closest object
3. **shade**: compute the color of the intersection point
Ray Tracing Program

for each pixel do
    compute viewing ray
    if ( ray hits an object with t in [0, inf] ) then
        compute n
        evaluate shading model and set pixel to that color
    else
        set pixel color to the background color
Recursive ray tracing

ray = ray(e,d,t0,t_max)

function ray_color(ray)
    if (Intersection(ray)) then
        point = ray.Point(ray.t_max)
        color c = color_ambient
        if (! Intersection(ray(point,l,eps,inf))
            h = halfway_vector
            c = c + color_diffuse + color_specular
            c = c + k_m * ray_color(ray(point,r,eps,inf))
        else
            color c = background color
Object-oriented design

class Surface
{
    public:
    void Intersection(RAY& ray)=0;
    Box Bounding_Box()=0;
}

Other objects: Ray, Light, Material, Camera, Film, World