Ray Tracing

Wikimedia Commons
shallow depth of field, area light sources, diffuse interreflection
up to 16 reflections per ray

Greg L., Wikimedia Commons
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

\[
\text{ray} = \text{ray}(e, d, t_0, t_{\text{max}})
\]

\[
\textbf{function} \ \text{ray\_color}(\text{ray})
\]
\[
\text{if } (\text{Intersection}(\text{ray})) \ \textbf{then}
\]
\[
\text{point} = \text{ray}.\text{Point}(\text{ray}.t_{\text{max}})
\]
\[
\text{color } c = \text{color\_ambient}
\]
\[
\textbf{if } (! \ \text{Intersection}(\text{ray}(\text{point}, l, \text{eps}, \text{inf})))
\]
\[
\text{h} = \text{halfway\_vector}
\]
\[
\text{c} = \text{c} + \text{color\_diffuse} + \text{color\_specular}
\]
\[
\text{c} = \text{c} + k_m \ \text{ray\_color}(\text{ray}(\text{point}, r, \text{eps}, \text{inf}))
\]
\[
\textbf{else}
\]
\[
\text{color } c = \text{background\ color}
\]
Object-oriented design

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

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