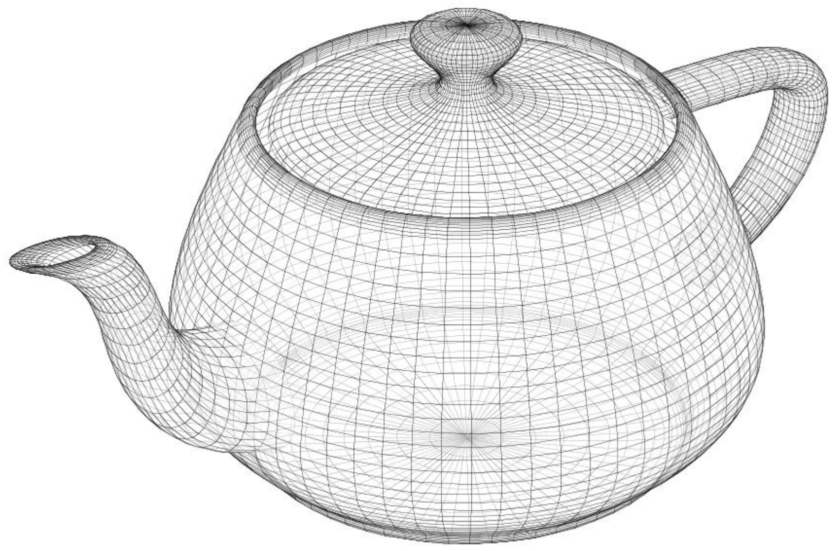


Path Tracing

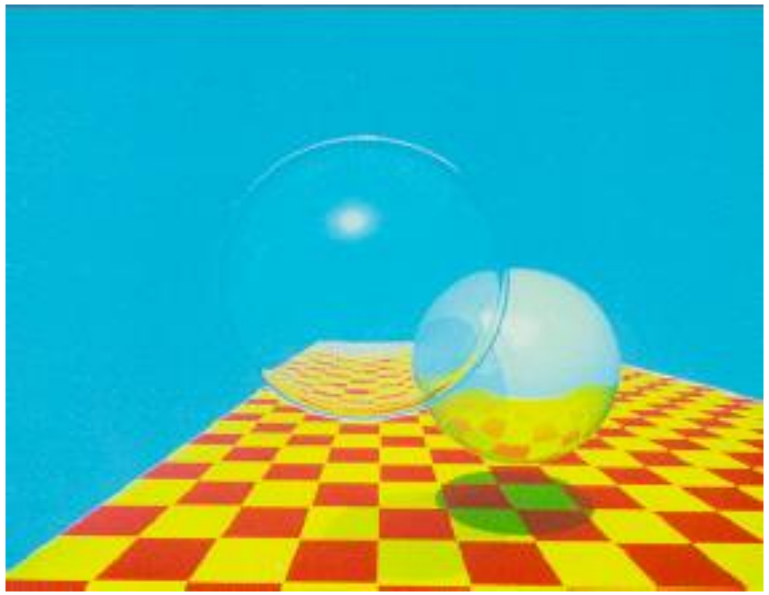
Production Computer Graphics

Eric Shaffer

A Brief History of Path Tracing

- Path tracing ≈ recursive ray-tracing
- Turner Whitted [1979]: perfect specular reflection and refraction
- Cook et al [1984]: stochastic ray-tracing
  - glossy effects
  - motion blur
  - depth of field
- Kajiya [1986]: rendering equation, full global illumination
- Lafortune [1993]: Bidirectional path-tracing
- Jensen [1995]: Photon mapping

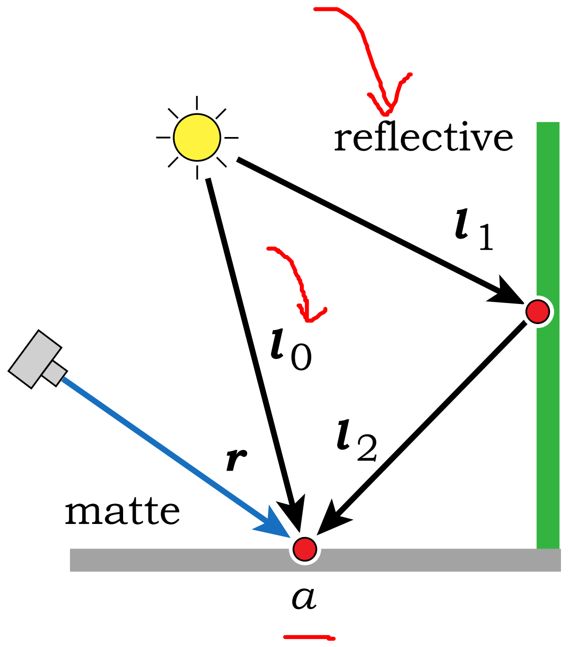
Whitted Ray-Tracing



- 1979, Turner Whitted
- As an aside:
  - 74 minutes at 640x480
  - By Moore's Law we should now have 1024x1024 at 16 f.p.s.

Global Illumination

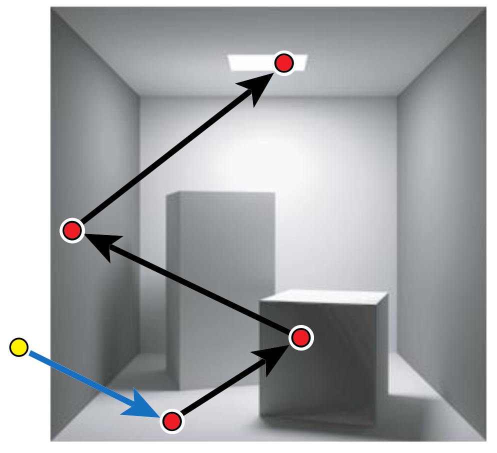
- Whitted ray-tracing can’t discover certain light paths
- The l1 to l2 path below wouldn’t be discovered

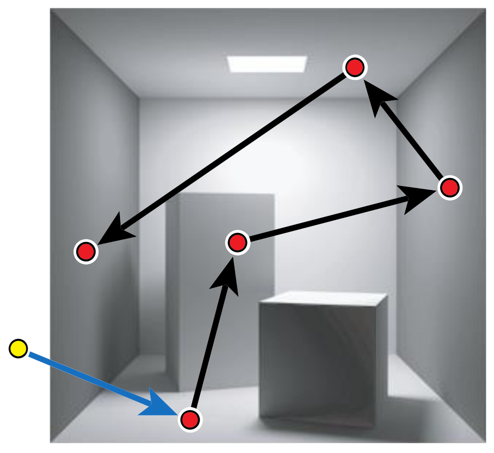


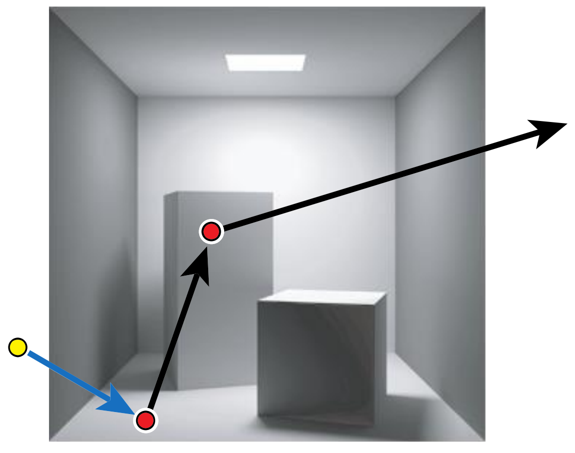
Path Tracing

Path Tracing: more completely approximates the rendering equation

- Each ray is followed in a scene until:
  - It hits a light (a)
  - Maximum recursion depth is reached (b)
  - The ray leaves the scene (c)

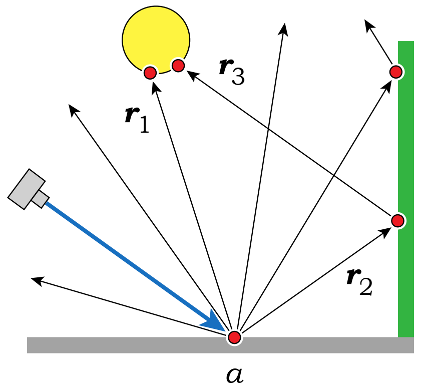

(a)


(b)


(c)

Path Tracing

- Shoot multiple rays per pixel
- A primary ray and its reflected rays are a *ray path*
- At each hit point, a PDF is determines reflected ray direction
- In pure Path Tracing radiance only comes from area lights
    - no point or directional lighting



The blue ray represents multiple primary rays!

Each primary ray generates a single path.

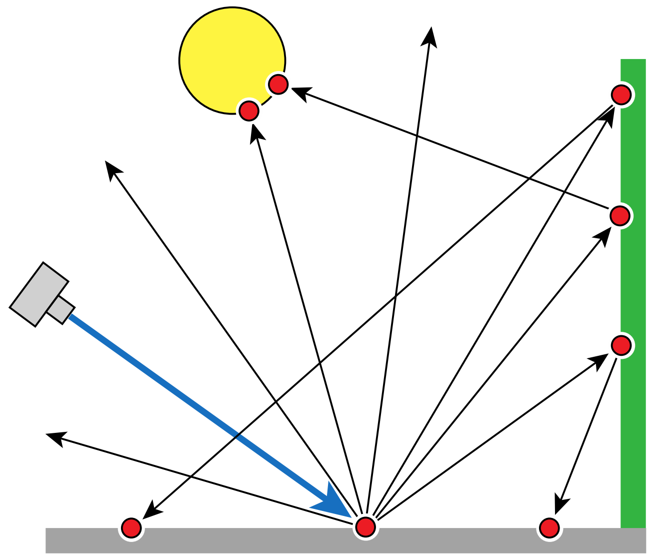
Path Tracing

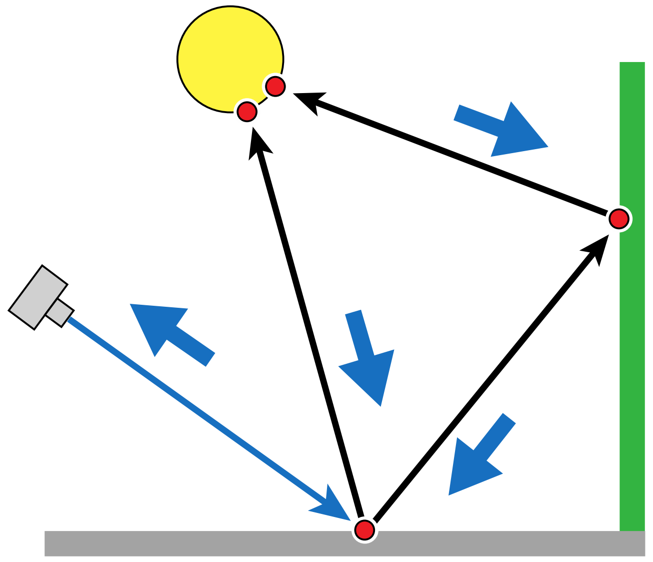
Material type determines ray orientation

- Pure Lambertian surfaces reflect rays in a random direction

Relying only on hitting lights leads to long convergence times

- Few paths will hit lights in the recursion depth limit



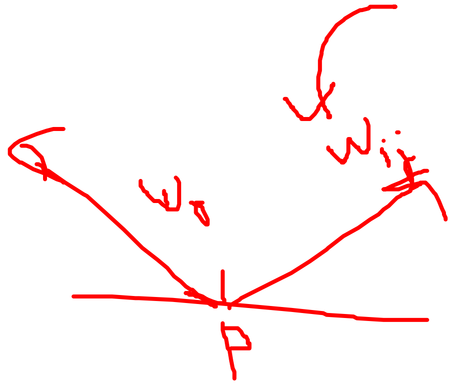


A Monte Carlo Estimator for Path Tracing

Note: subscript *i* refers to incoming irradiance…not an index

- To evaluate the radiance at ***each pixel:***

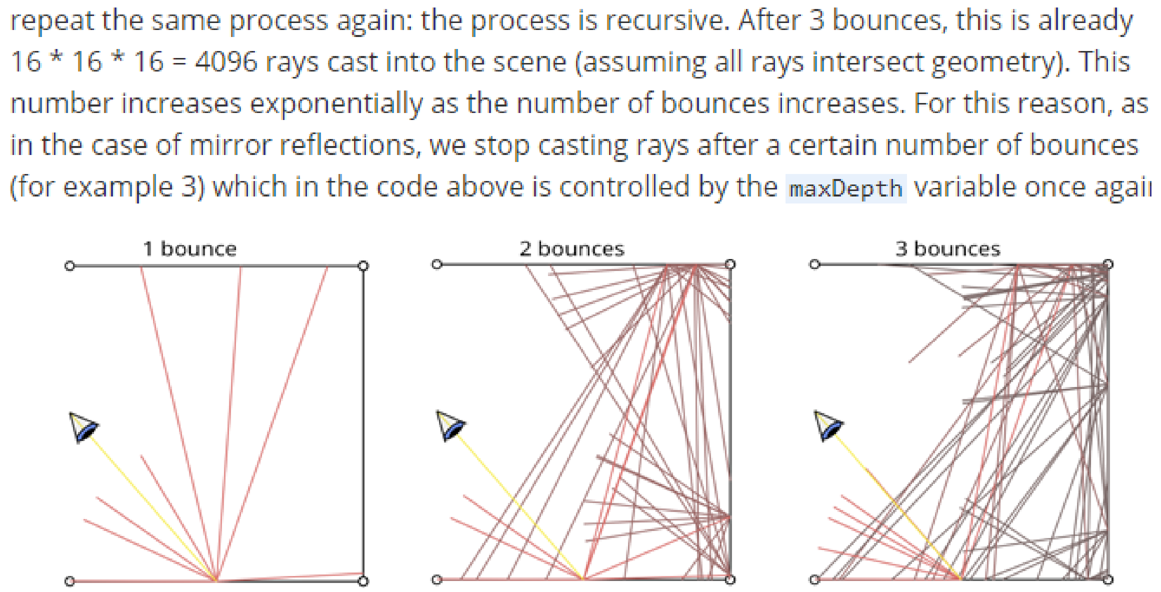
$$\langle Lr(p,\omega_o)\rangle = \frac{1}{n_s}\sum_{j=1}^{n_s}\frac{f_r(p,\omega_{i,j},\omega_o)L_i(p,\omega_{i,j})\cos\theta_{i,j}}{p(\omega_{i,j})} + \varepsilon$$



- We are generating n$_s$ rays per pixel
  - The expression inside the summation computes radiance from first hit
  - But, of course, it's recursive

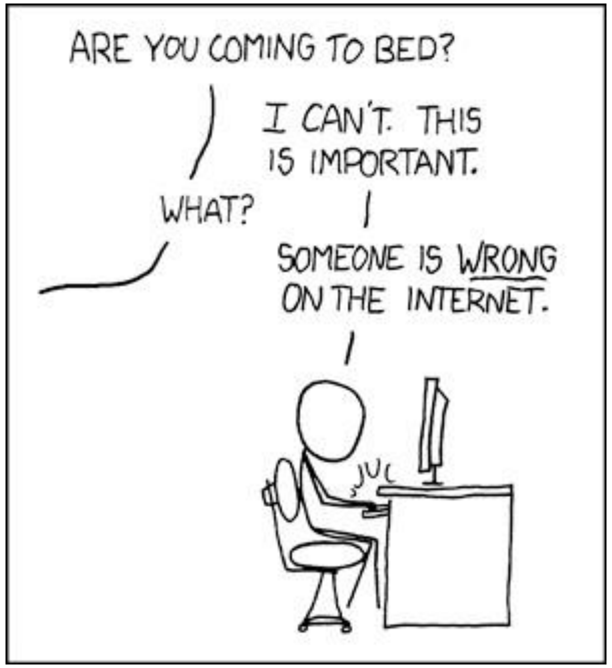
Things that are wrong on the internet

Path tracing does not branch…and yet



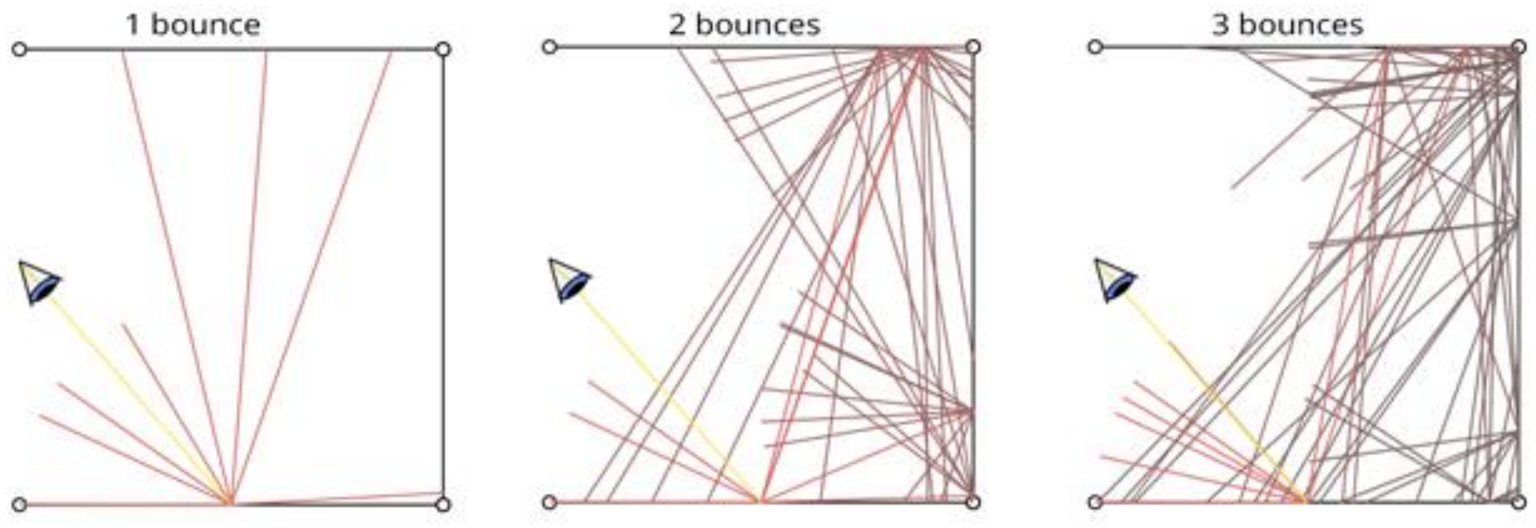
repeat the same process again: the process is recursive. After 3 bounces, this is already 16 * 16 * 16 = 4096 rays cast into the scene (assuming all rays intersect geometry). This number increases exponentially as the number of bounces increases. For this reason, as in the case of mirror reflections, we stop casting rays after a certain number of bounces (for example 3) which in the code above is controlled by the `maxDepth` variable once agai

1 bounce
2 bounces
3 bounces

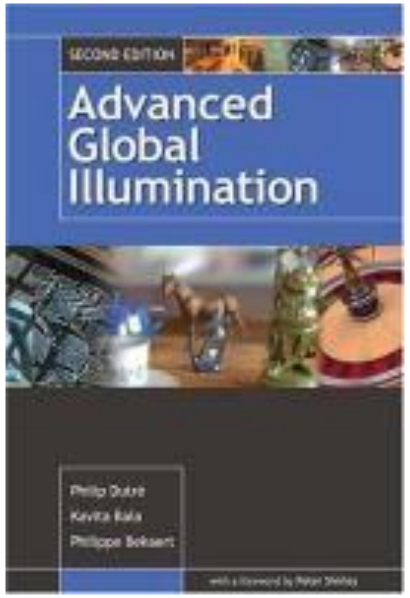

ARE YOU COMING TO BED?
I CAN'T. THIS IS IMPORTANT.
WHAT?
SOMEONE IS WRONG ON THE INTERNET.

What is a good argument against branching?

Consider the relative contribution of rays cast at different depths

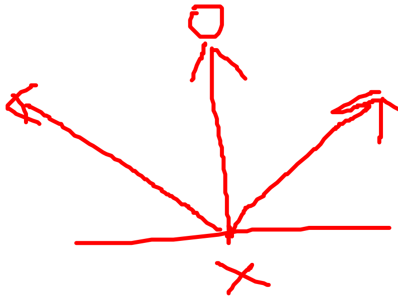


Path Tracer Example


SECOND EDITION
Advanced Global Illumination
Philip Dutré
Kavita Bala
Philippe Bekaert
with a foreword by Peter Shirley

How does this code deviate from pure path tracing?

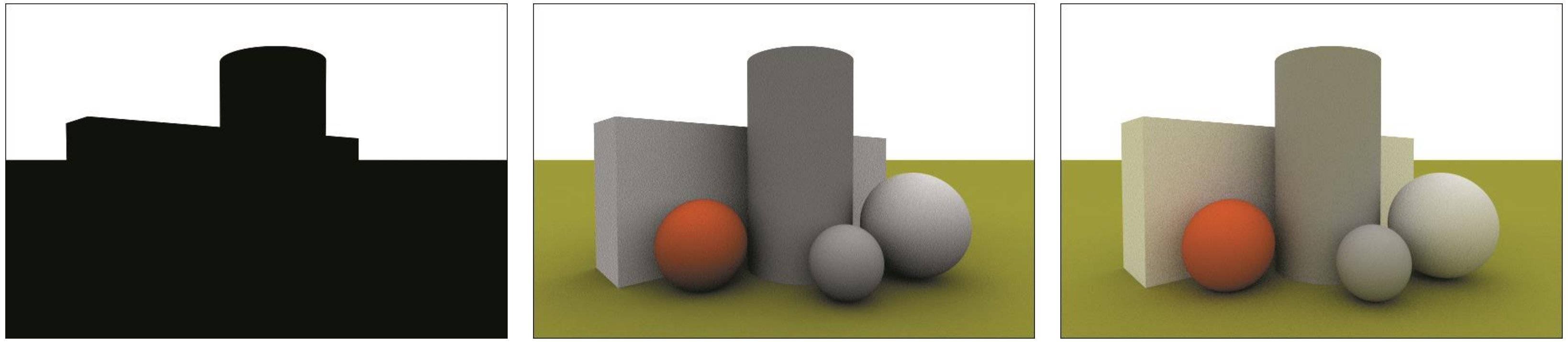
Why does it do that?



```
// global illumination algorithm
// stochastic ray tracing
computeImage(eye)
    for each pixel
        radiance = 0;
        H = integral(h(p));
        for each sample          // Np viewing rays
            pick sample point p within support of h;
            construct ray at eye, direction p-eye;
            radiance = radiance + rad(ray)*h(p);
        radiance = radiance/(#samples*H);

rad(ray)
    find closest intersection point x of ray with scene;
    return(Le(x,eye-x) + computeRadiance(x, eye-x));

computeRadiance(x, dir)
    estimatedRadiance += directIllumination(x, dir);
    estimatedRadiance += indirectIllumination(x, dir);
    return(estimatedRadiance);

directIllumination (x, theta)
    estimatedRadiance = 0;
    for all shadow rays          // Nd shadow rays
        select light source k;
        sample point y on light source k;
        estimated radiance +=
            Le * BRDF * radianceTransfer(x,y)/(pdf(k)pdf(y|k));
    estimatedRadiance = estimatedRadiance / #paths;
    return(estimatedRadiance);

indirectIllumination (x, theta)
    estimatedRadiance = 0;
    if (no absorption)           // Russian roulette
        for all indirect paths   // Ni indirect rays
            sample direction psi on hemisphere;
            y = trace(x, psi);
            estimatedRadiance +=
                compute_radiance(y, -psi) * BRDF *
                cos(Nx, psi)/pdf(psi);
        estimatedRadiance = estimatedRadiance / #paths;
    return(estimatedRadiance/(1-absorption));

radianceTransfer(x,y)
    transfer = G(x,y)*V(x,y);
    return(transfer);
```

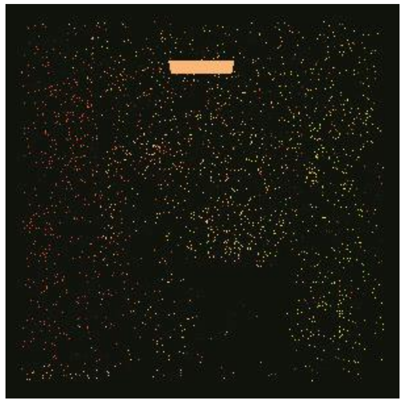
Path Tracing with an Environment Light

Zero bounces…one bounce….five bounces

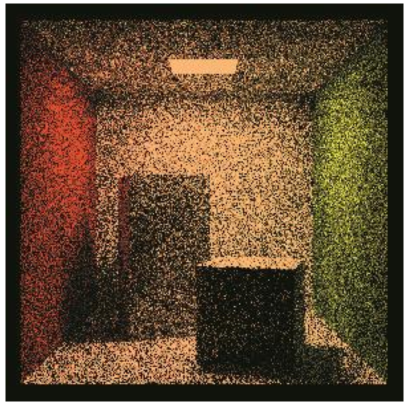


Path Tracing with an Area Light

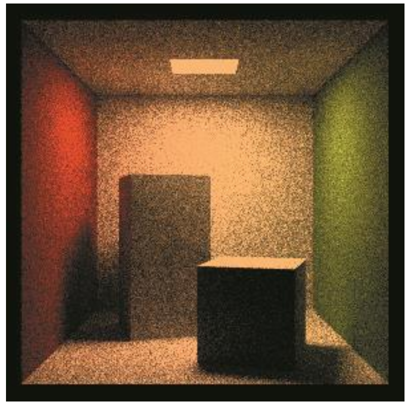
1 sample per pixel



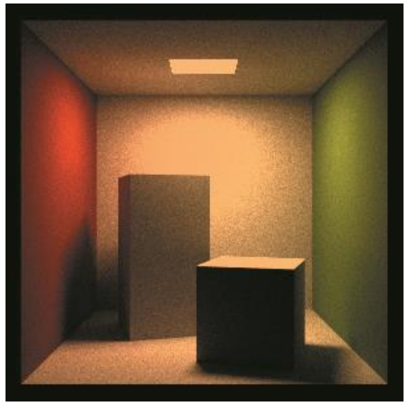
100



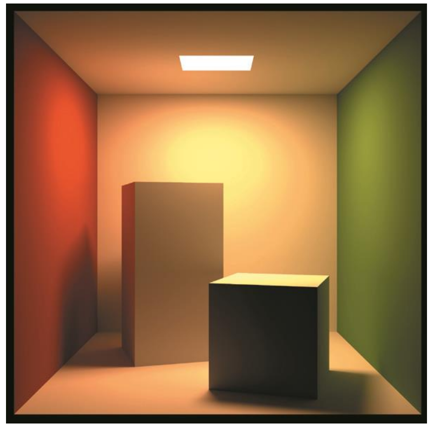
1024



10,000



100,000



Path Tracing : Speedup through Light Sampling

Can employ a hybrid approach

- Direct illumination computed by sampling lights
- Indirect illumination computed by path tracing

Must take care not to render direct illumination twice

- Sample direct illumination with a shadow ray
- Indirect illumination as before
  - Hit on a light returns radiance if ray depth ≠ 1