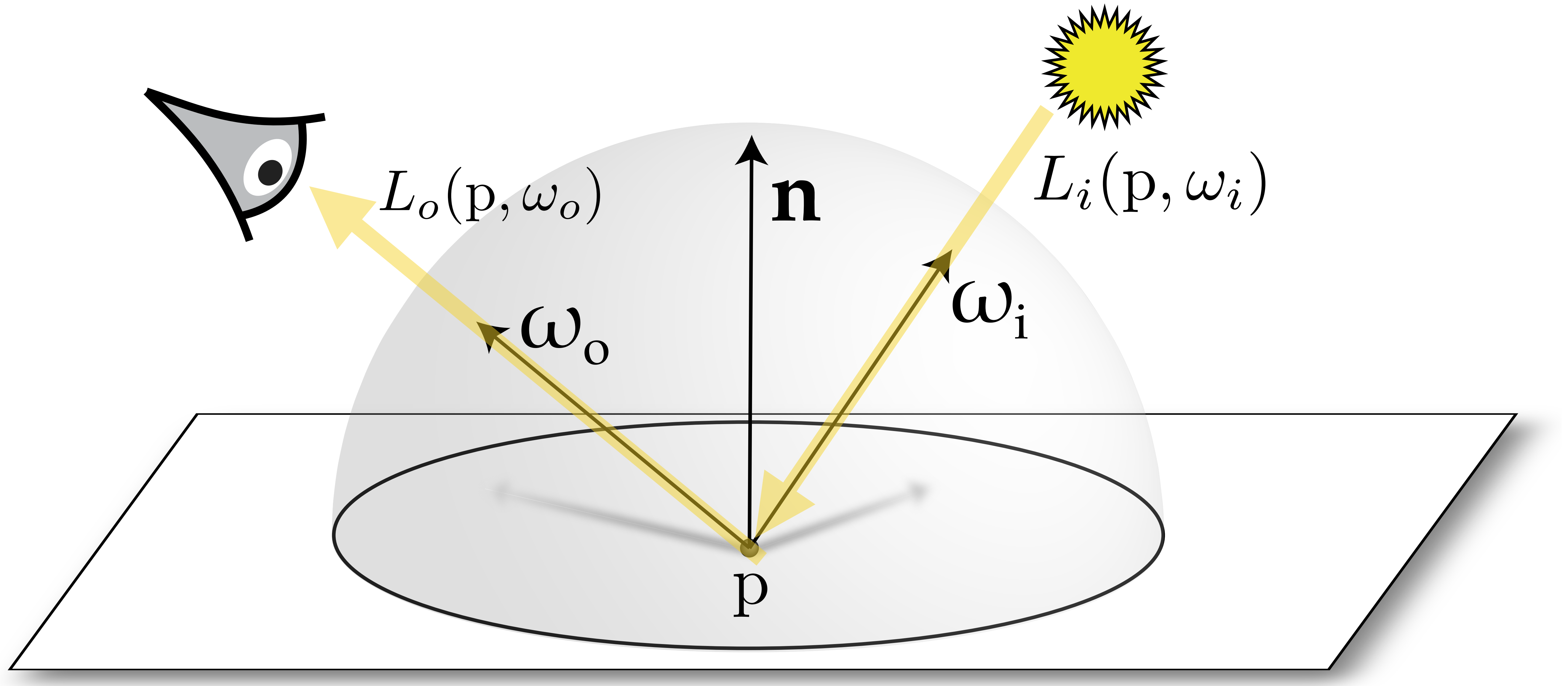
Lecture 13:

Global Illumination using Path Tracing

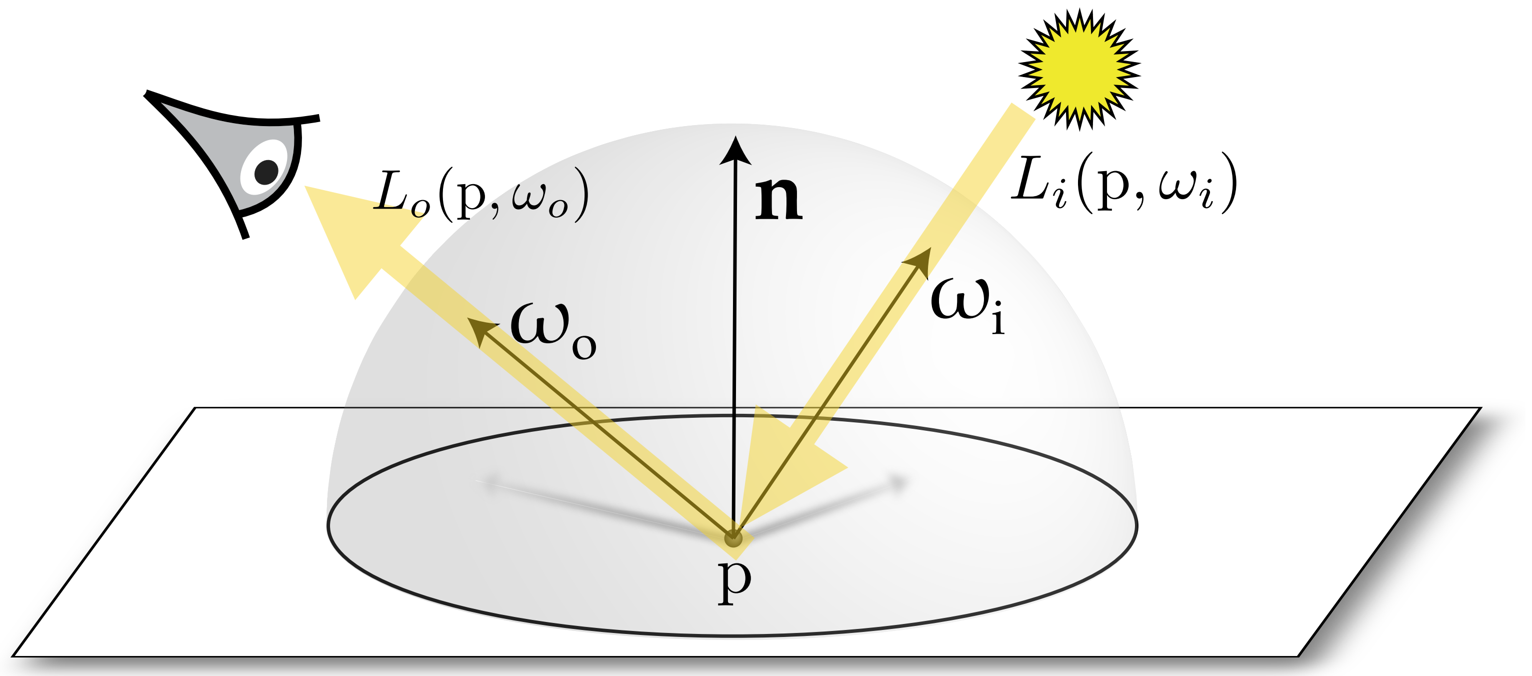
Computer Graphics: Rendering, Geometry, and Image Manipulation
Stanford CS248A, Winter 2024

Review (one more time): the reflection equation



$$L_{\mathrm{o}}(\mathrm{p}, \omega_{\mathrm{o}}) = \int_{\Omega^2} \underbrace{f_{\mathrm{r}}(\mathrm{p}, \omega_{\mathrm{i}} \rightarrow \omega_{\mathrm{o}})}_{\textbf{BRDF}} \underbrace{L_{\mathrm{i}}(\mathrm{p}, \omega_{\mathrm{i}}) \, \cos\theta_{\mathrm{i}}}_{\textbf{Illumination}} \, \mathrm{d}\omega_{\mathrm{i}}$$

Last time: Monte Carlo estimate for the reflection equation



Reflection equation in terms of integration over solid angles:

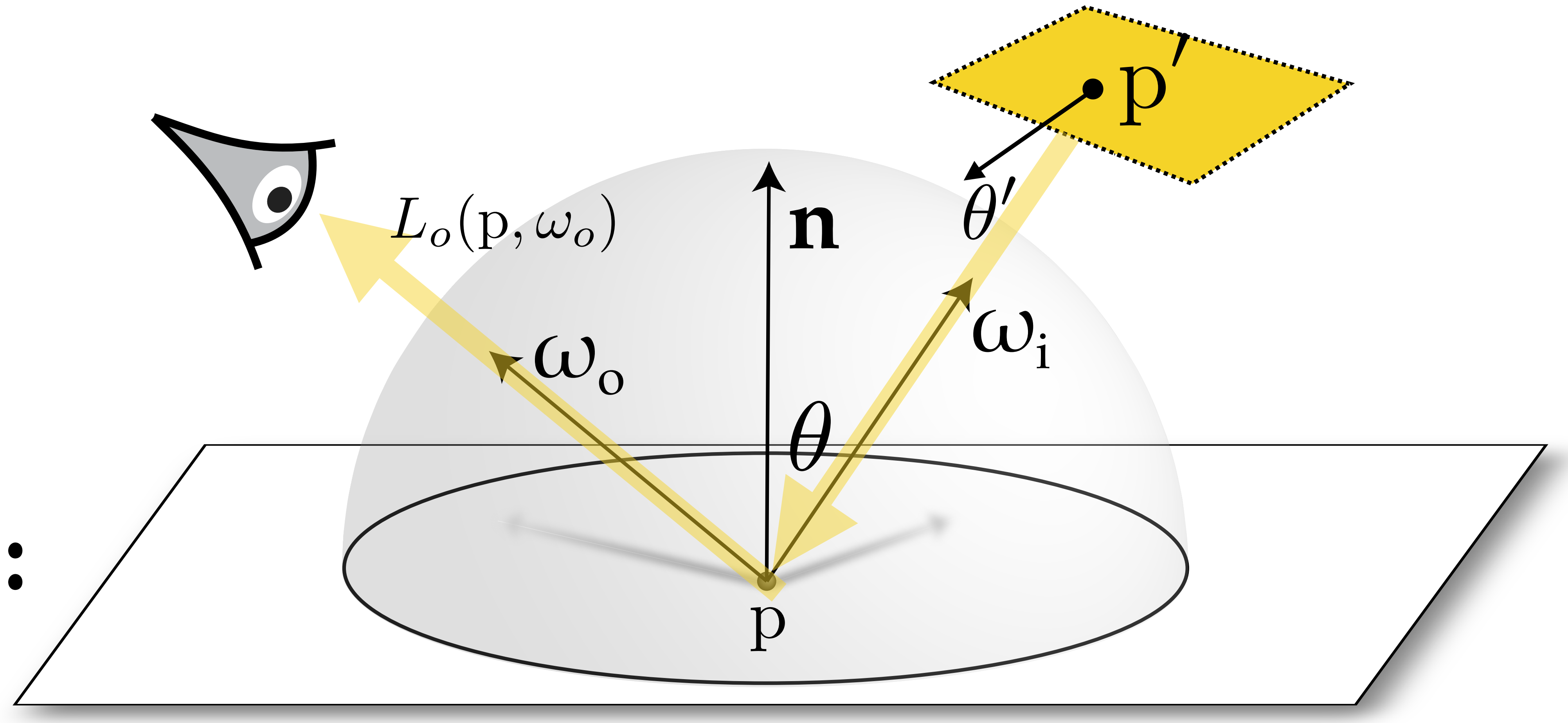
$$L_\mathrm{o}(\mathrm{p}, \omega_\mathrm{o}) = \int_{H^2} f_r(\mathrm{p}, \omega_\mathrm{i} \to \omega_\mathrm{o})\, L_\mathrm{i}(\mathrm{p}, \omega_\mathrm{i})\, \cos\theta_\mathrm{i}\, \mathrm{d}\omega_\mathrm{i}$$

Monte Carlo estimate via sampling solid angle according to the PDF: $p(\omega)$

Sample: $\omega_j \sim p(\omega)$

Estimate: $$\frac{1}{N}\sum_{j=1}^{N} \frac{f_r(\mathrm{p}, \omega_j \to \omega_\mathrm{o})\, L_\mathrm{i}(\mathrm{p}, \omega_j)\, \cos\theta_j}{p(\omega_j)}$$

Last time: Monte Carlo estimate for the reflection equation



Assume area light source emitting radiance L in all directions.

Reflection equation in terms of integration over one light source area:

$$L_o(\mathrm{p},\omega_o) = \int_{A'} f_r(\mathrm{p},(\mathrm{p}'-\mathrm{p})\to\omega_o)\, L\, \frac{\cos\theta\cos\theta'}{|\mathrm{p}-\mathrm{p}'|^2}\,\mathrm{dA}$$

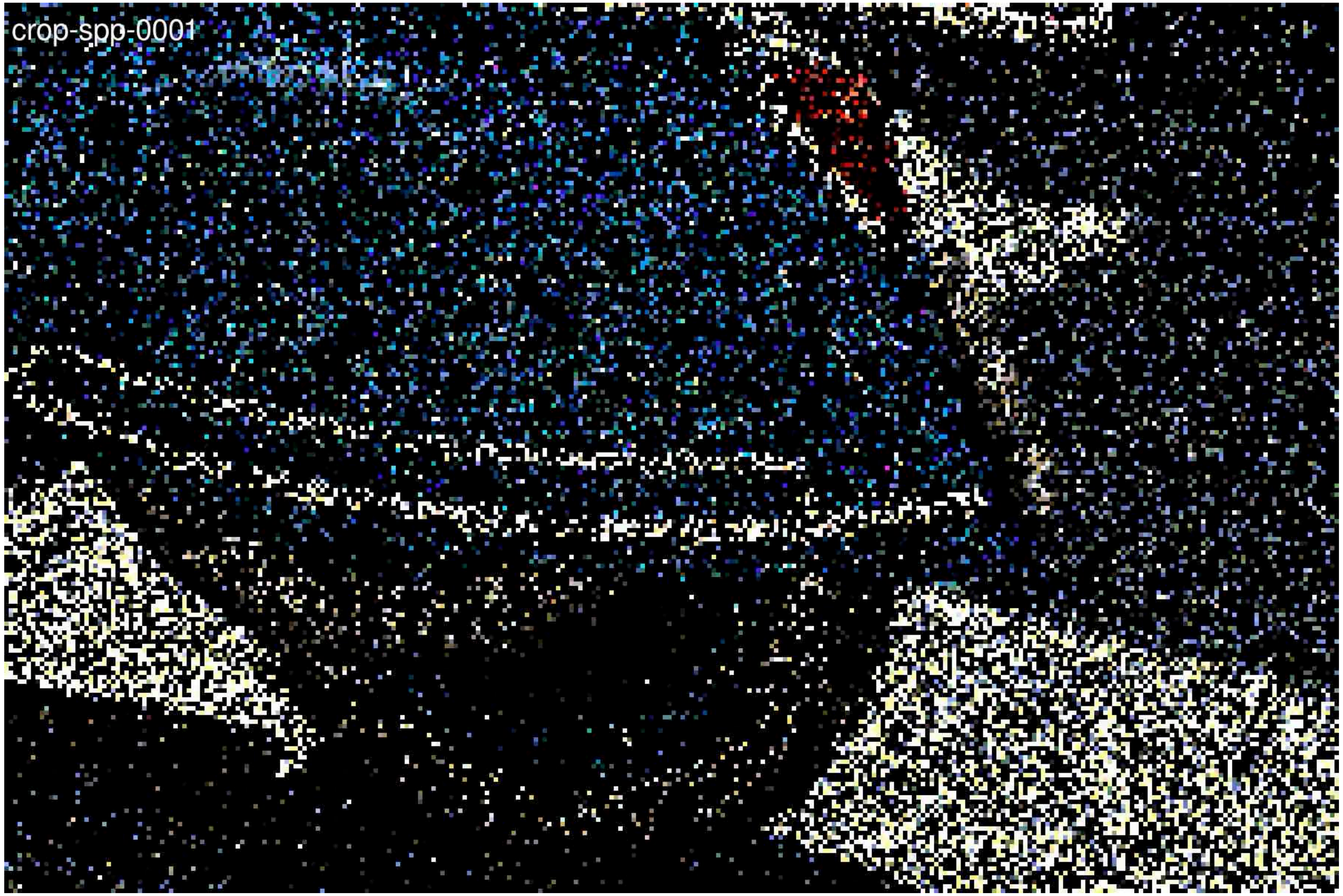
Monte Carlo estimate via sampling points on lights:

Sample:

$$\mathrm{p}_j \sim p(\mathrm{p})$$

Estimate: $$\frac{1}{N}\sum_{j=1}^{N}\frac{f_r(\mathrm{p},(\mathrm{p}'-\mathrm{p})\to\omega_o)\, L\, \cos\theta\cos\theta'}{p(\mathrm{p}_j)\,|\mathrm{p}-\mathrm{p}'|^2}$$

Review: talk to me about why this picture looks like this



For video see: http://pharr.org/matt/assets/bistro-spp.mp4

Major themes from last time

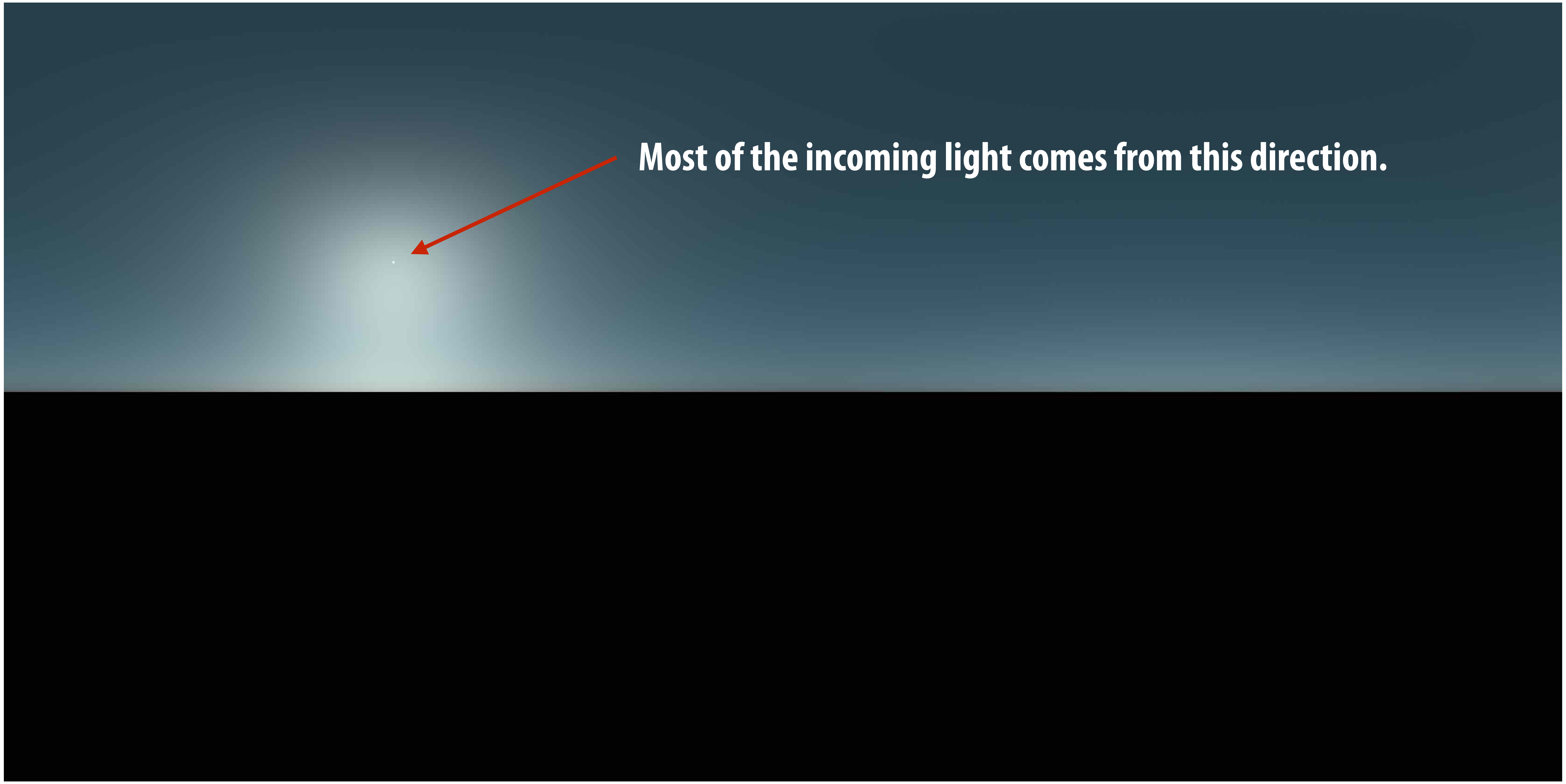
- Monte Carlo: draw samples from integration domain according to p(x)
- Evaluate estimator at all samples
- In estimation, average of estimators is the integral we wish to compute
- Importance sampling: biasing sampling towards parts of the domain where integrand is large reduces variance in Monte Carlo estimate
  - Consider: $\int_{\Omega^2} f_\mathrm{r}(\mathrm{p}, \omega_\mathrm{i} \rightarrow \omega_\mathrm{o})\, L_\mathrm{i}(\mathrm{p}, \omega_\mathrm{i})\, \cos\theta_\mathrm{i}\, \mathrm{d}\omega_\mathrm{i}$
  - E.g., shape $p(\omega)$ to be proportional to: $\cos\theta_\mathrm{i}$ If no priors of lighting/BRDF known, or if lighting environment and/or BDRF are largely constant

$L_\mathrm{i}(\mathrm{p}, \omega_\mathrm{i})$ If priors on the lighting are known: eg., environment light sampling, bias samples toward powerful lights in many-light setting

$f_\mathrm{r}(\mathrm{p}, \omega_\mathrm{i} \rightarrow \omega_\mathrm{o})$ If BRDF is highly peaked, like a very glossy surface

Review: example of lighting environment with peaked distribution

$\theta \in [0, \pi]$

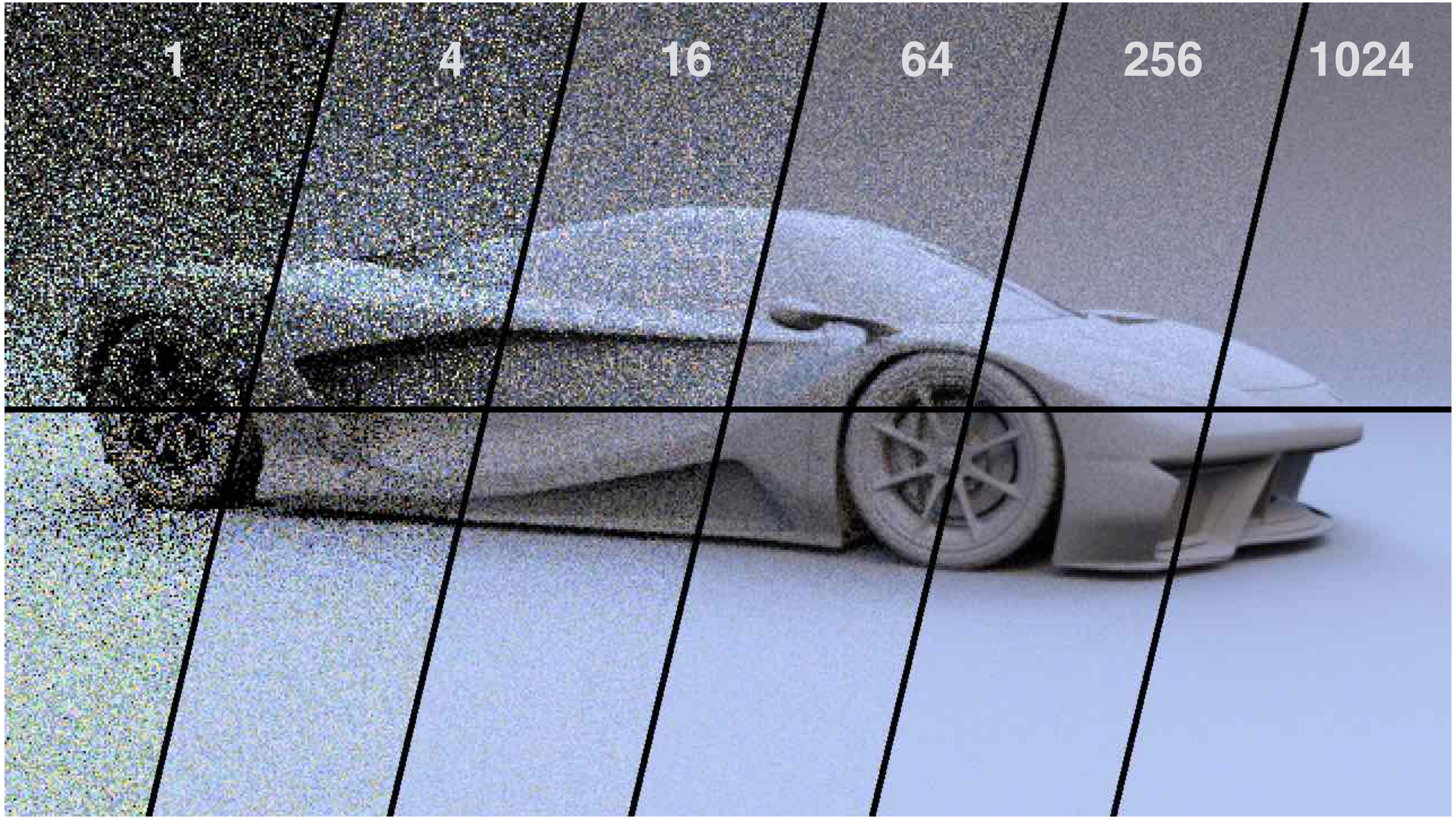


$\phi \in [0, 2\pi]$

Importance sampling reduces variance in reflection estimate

Uniform sampling

Importance sampling



What about an image like this?
Direct illumination + reflection + transparency

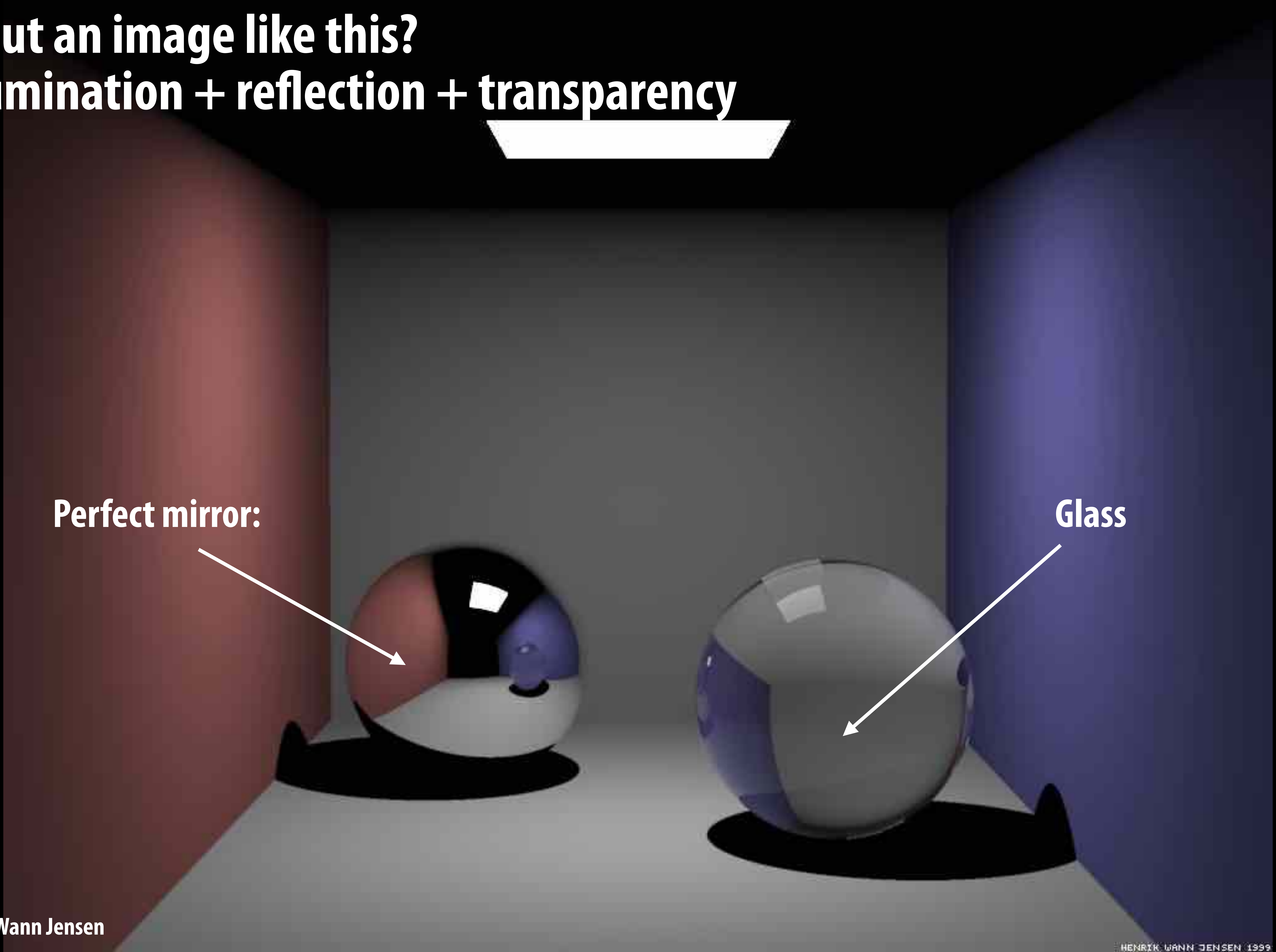
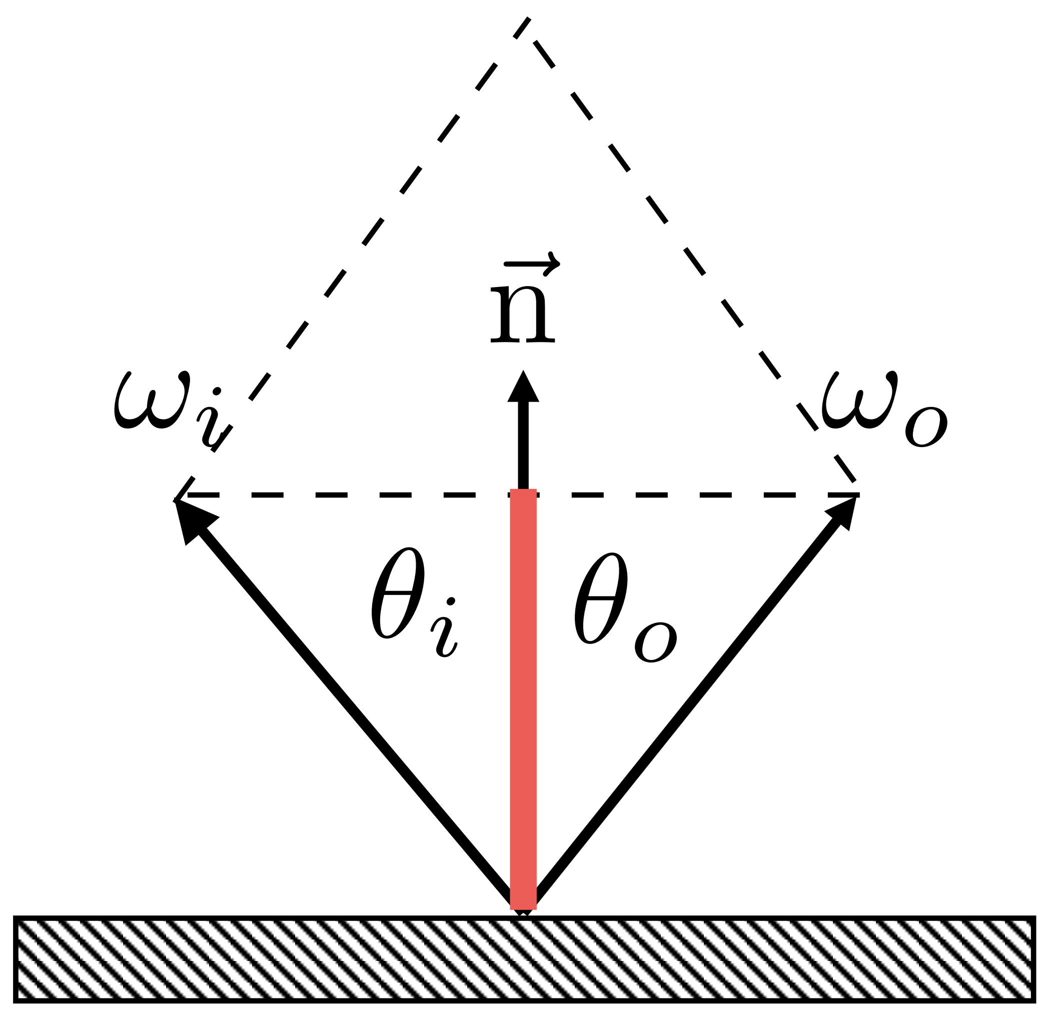


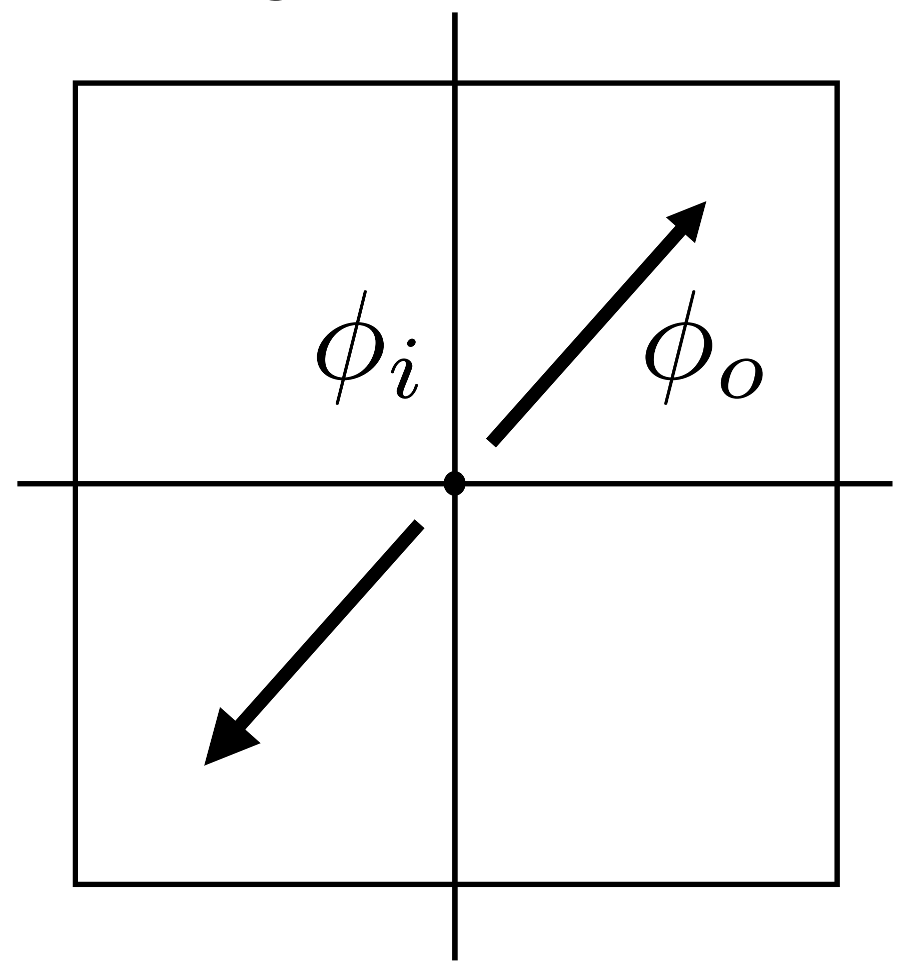
Image credit: Henrik Wann Jensen

Computing mirror reflection and transmission directions

Reflected ray:



Top-down view (looking down on surface)

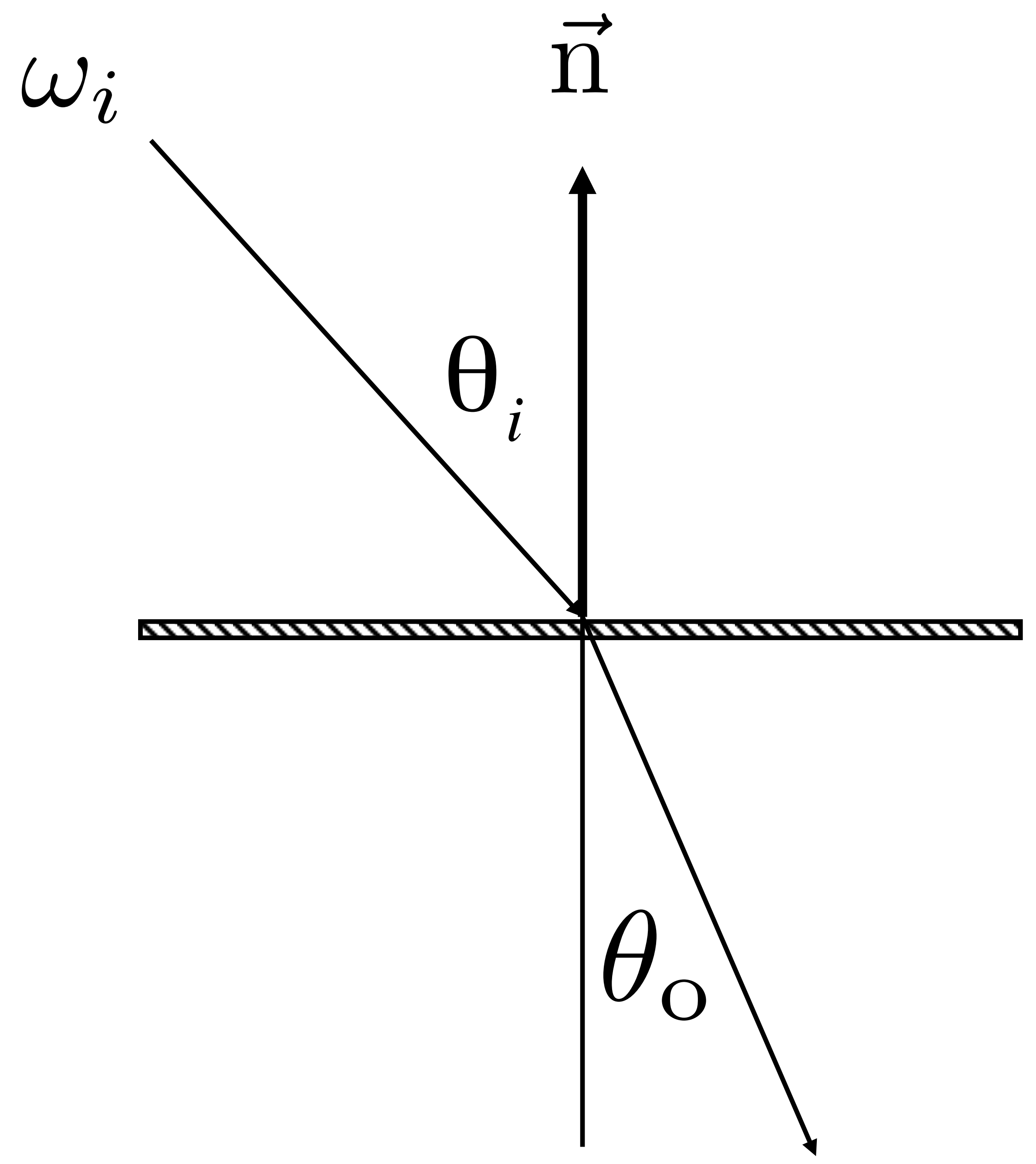


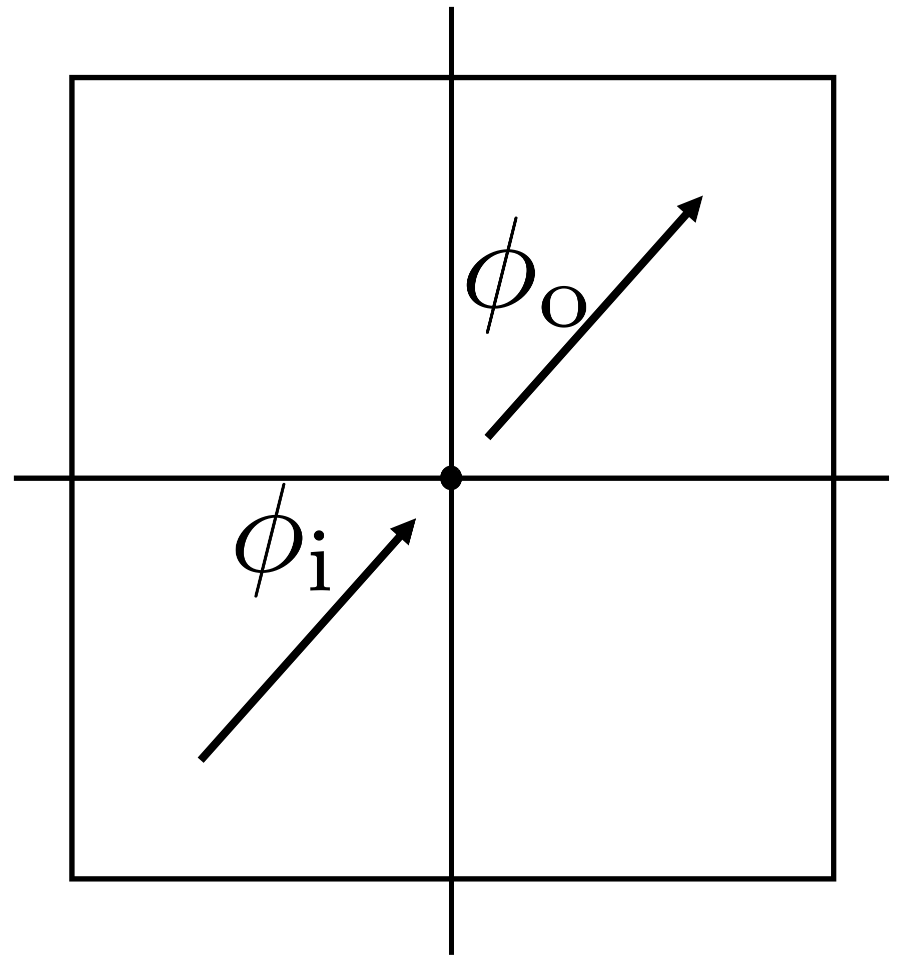
$$\theta = \theta_o = \theta_i$$

$$\phi_o = (\phi_i + \pi) \bmod 2\pi$$

$$\omega_o = -\omega_i + 2(\omega_i \cdot \vec{\mathrm{n}})\vec{\mathrm{n}}$$

Refracted ray: (transmission)





$$\eta_{\mathrm{i}} \sin\theta_{\mathrm{i}} = \eta_{\mathrm{o}} \sin\theta_{\mathrm{o}}$$

$$\phi_o = (\phi_i + \pi) \bmod 2\pi$$

Basic recursive ray tracing for a perfect mirror

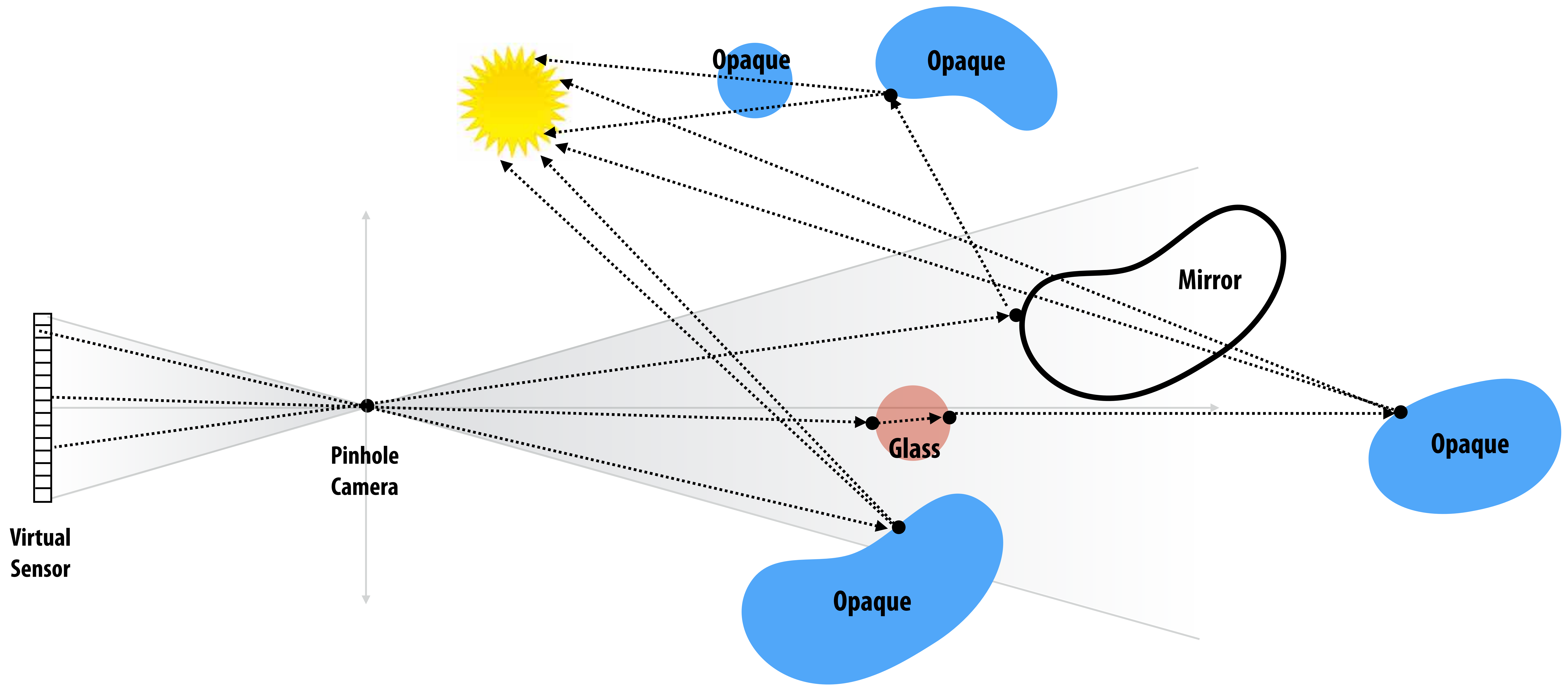
```
Spectrum TraceRay(Ray ray) {
  Intersection isect = scene->Intersect(ray);
  BSDF bsdf = isect.GetBSDF();
  Vector3f wo = -ray.d;

  if (bsdf->isMirror()) { // handle case of perfect mirror surface
    Vector3d wi = computeMirrorReflDir(isect.N, wo);
    if (ray.depth == MAX_DEPTH)
       return Spectrum(0.0);
    else
      return TraceRay(Ray(isect.P, wi, ray.depth+1));

  } else {                      // non-mirror case, compute reflectance due to unshadowed
                                // area source of radiance L
    Spectrum Lo;
    for (int i=0 to NUM_LIGHT_SAMPLES) {
       // sample point on light source to get: p_j, pdf(p_j)

       // compute MC estimator...

       Lo += estimator / NUM_LIGHT_SAMPLES;
    }
  }
}
```

Specular mirror reflectance

$$\frac{f_r(\mathrm{p}, (\mathrm{p}' - \mathrm{p}) \to \omega_o)\, L\, \cos\theta \cos\theta'}{p(\mathrm{p}_j)\, |\mathrm{p} - \mathrm{p}'|^2}$$

Reflectance due to direct lighting

Basic recursive ray tracing



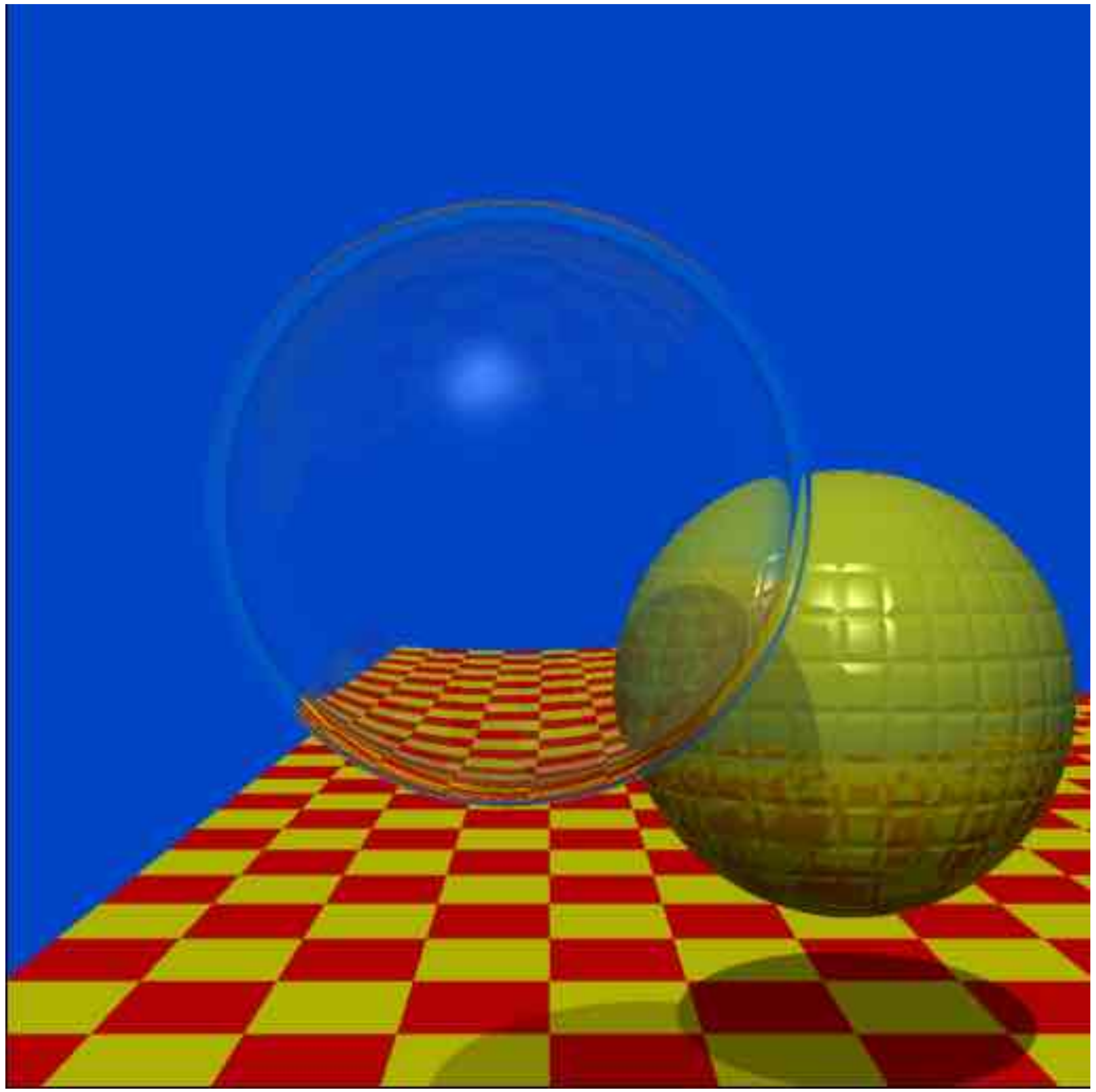
Whitted's recursive ray tracer

"An improved Illumination model for shaded display"
T. Whitted, CACM 1980

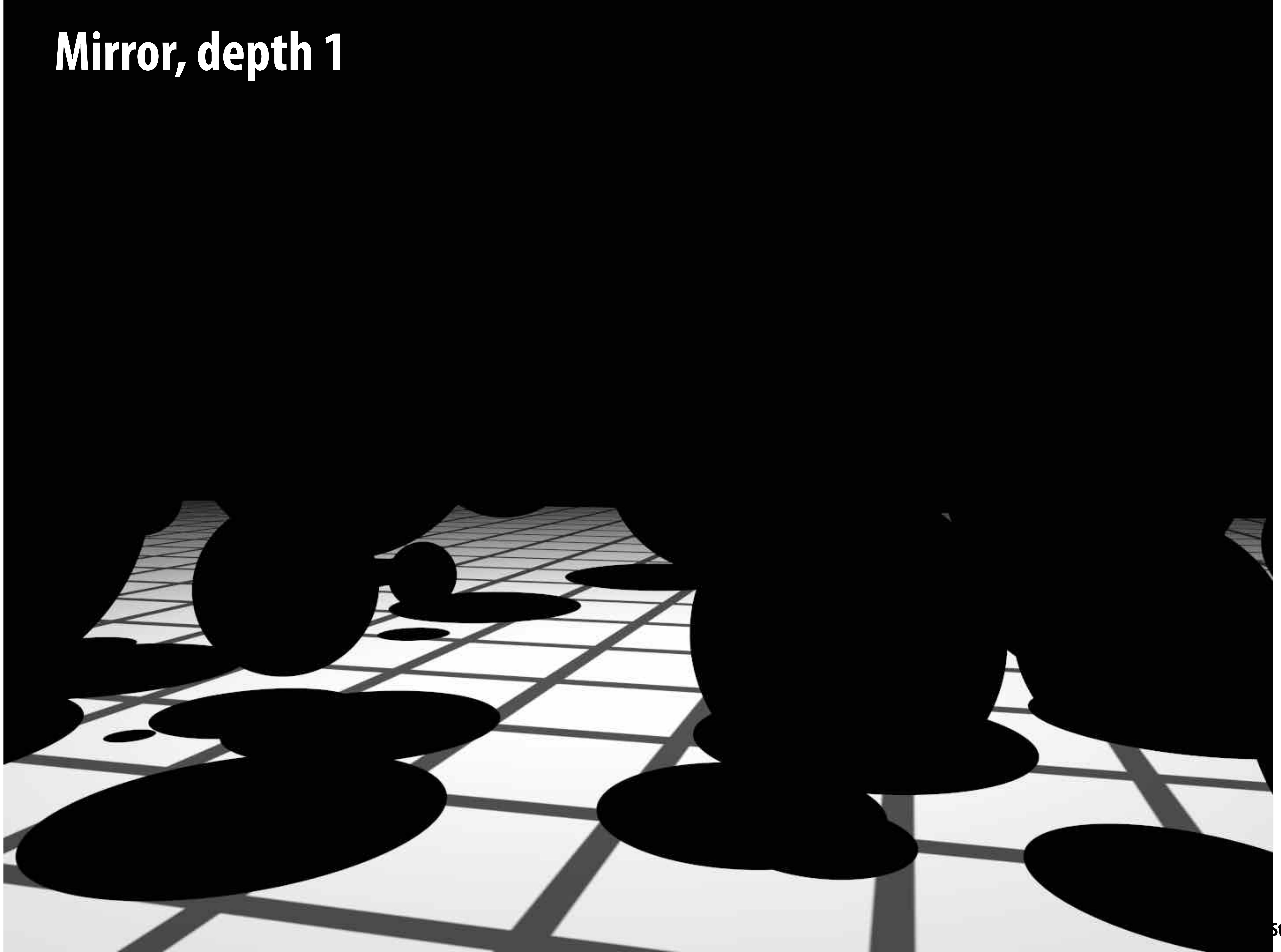
1. Always send ray
to the light source (unless glass or mirror)

2. Recursively generate reflected rays (mirror) and
transmitted rays (glass)

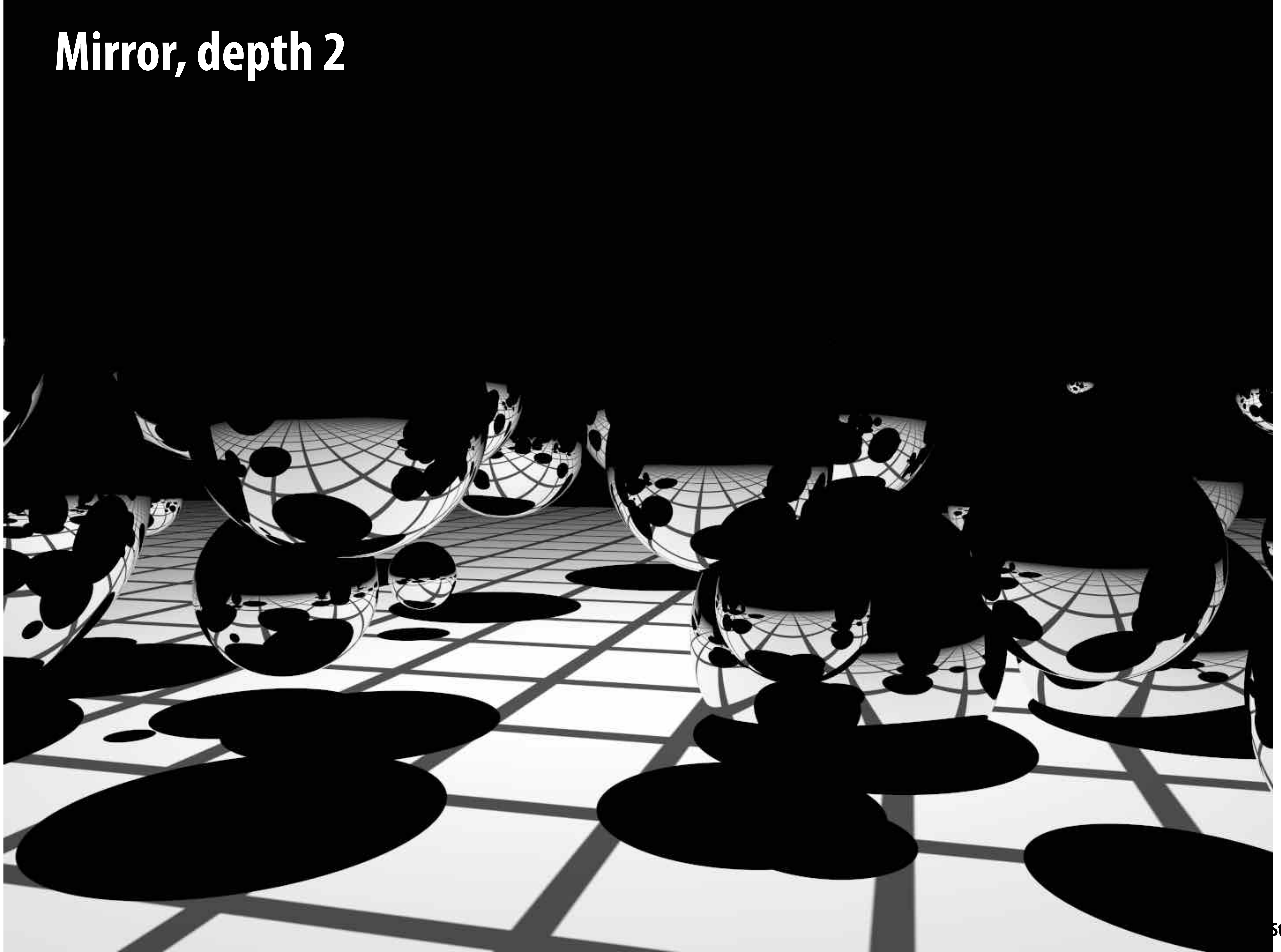
Time:
- **VAX 11/780 (1979) 74m**
- **PC (2006) 6s**
- **GPU (2012) 1/30s**


Spheres and Checkerboard
T. Whitted, 1979

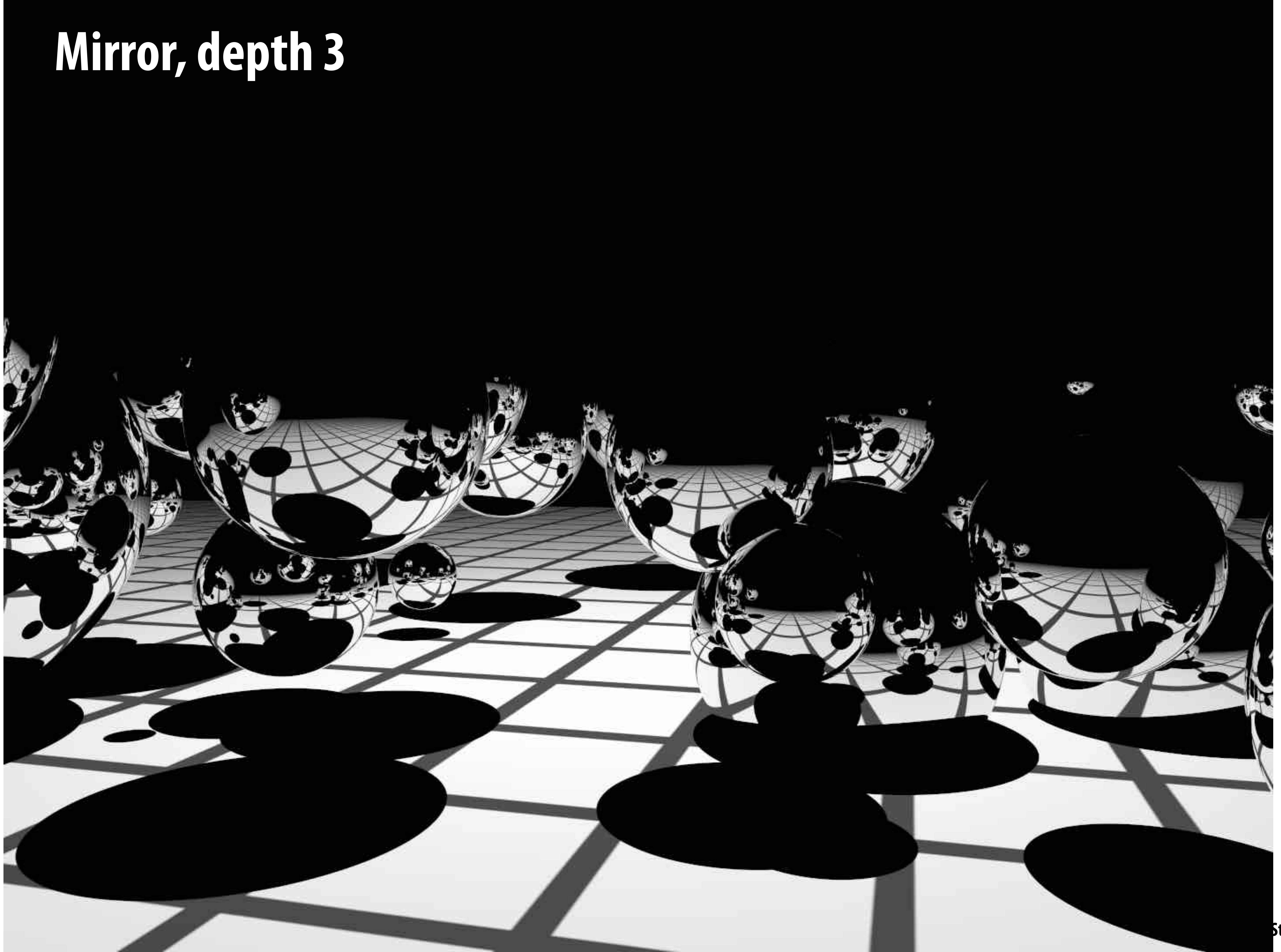
Mirror, depth 1



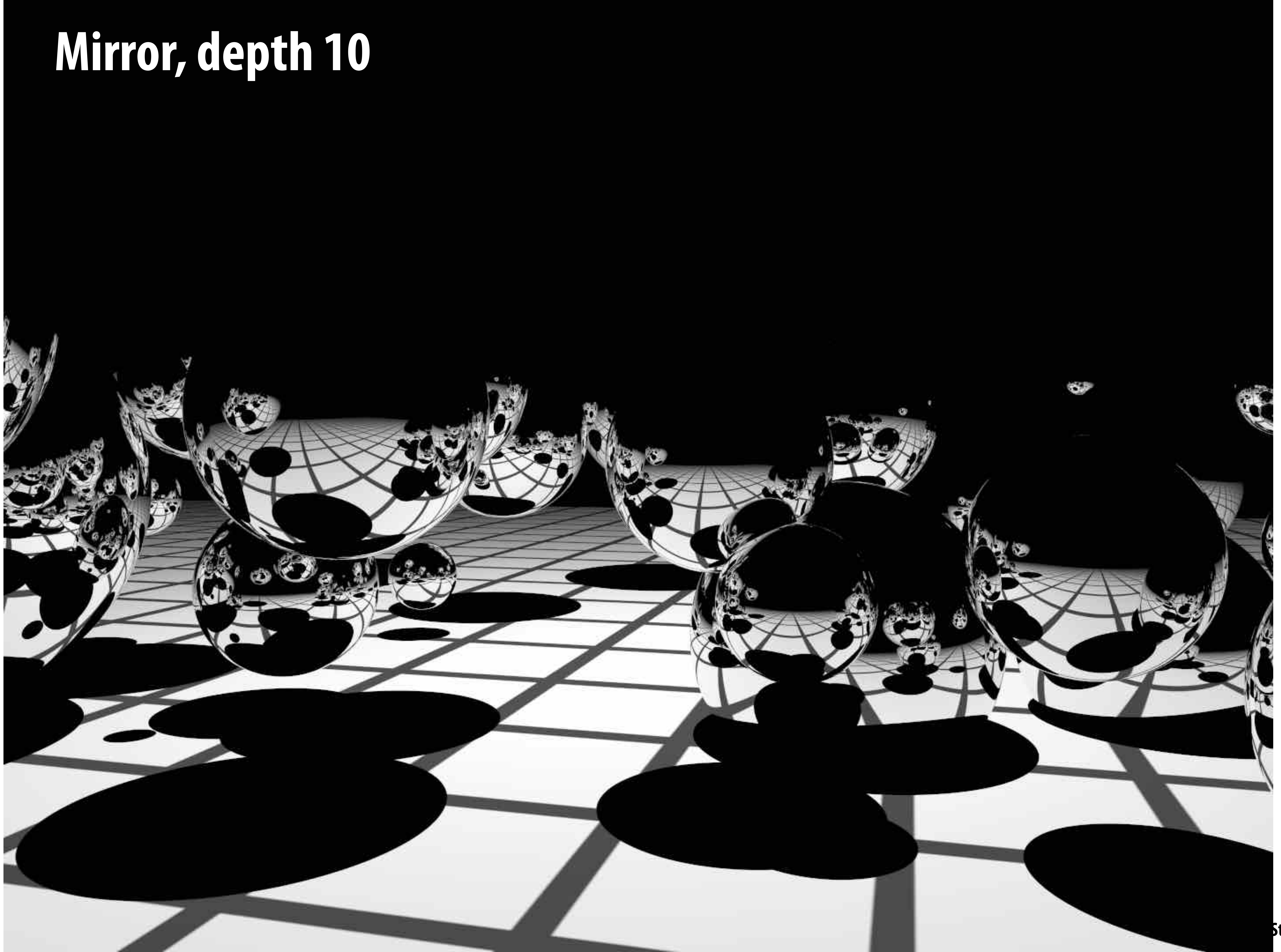
Mirror, depth 2



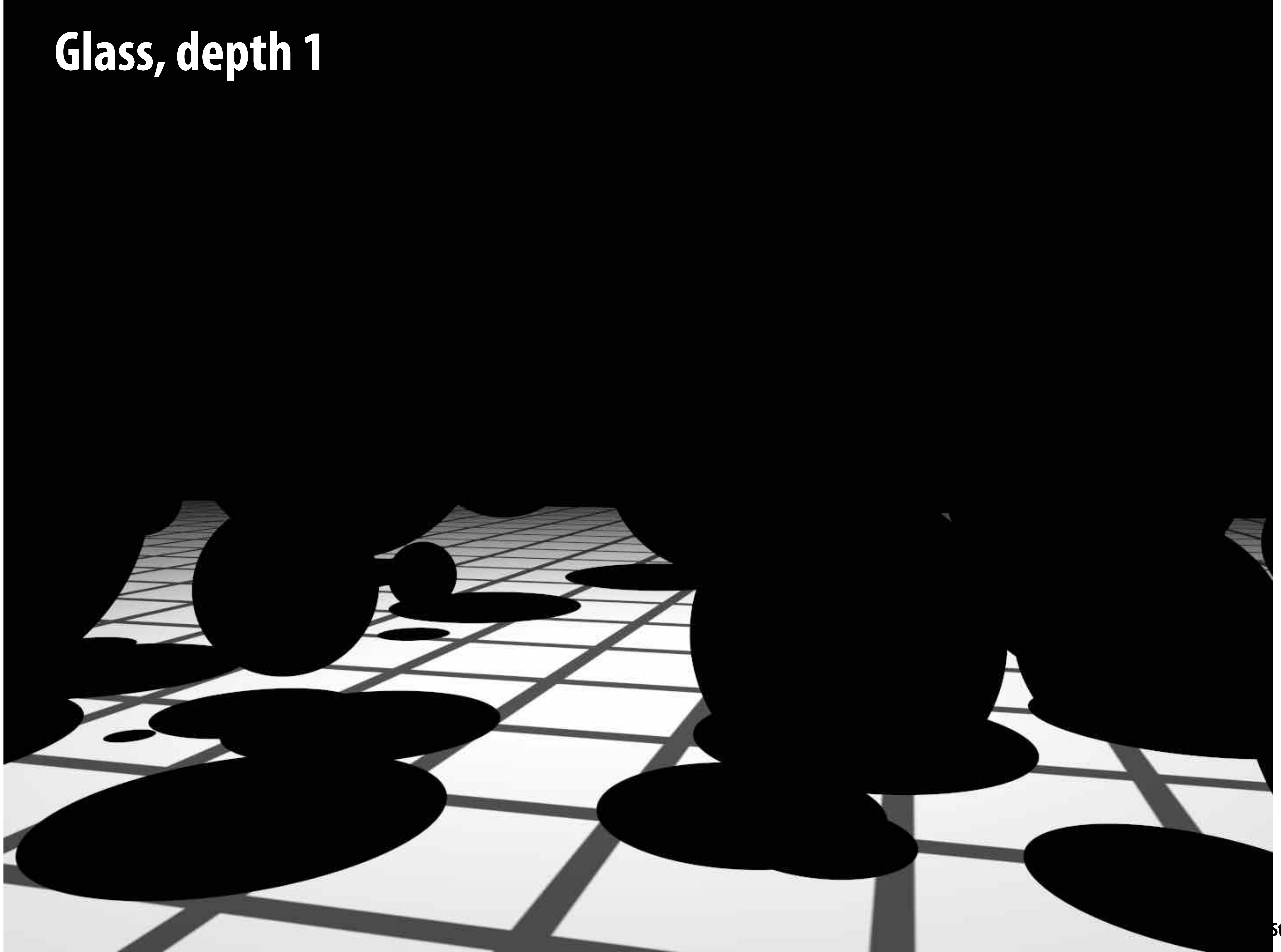
Mirror, depth 3



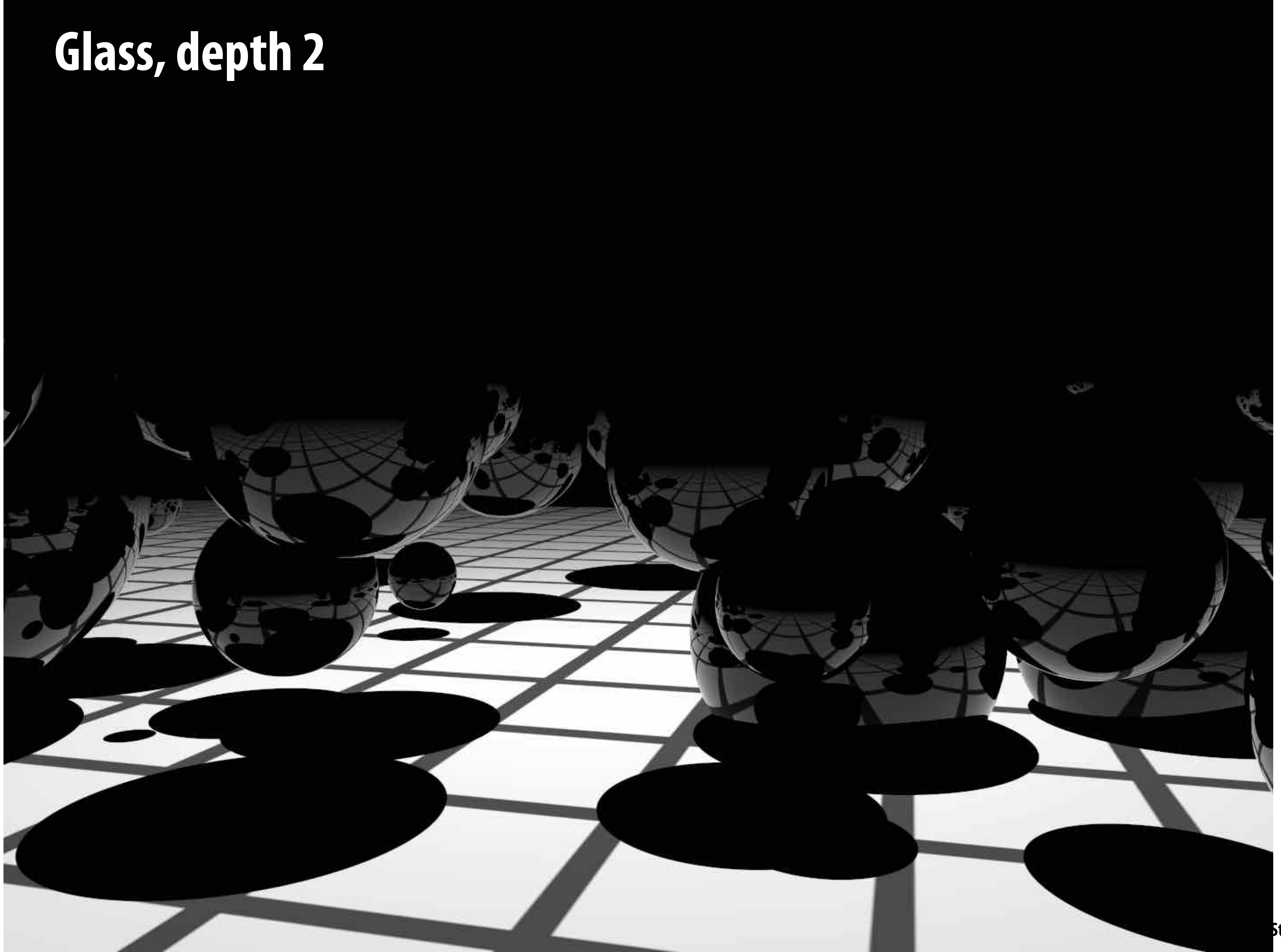
Mirror, depth 10



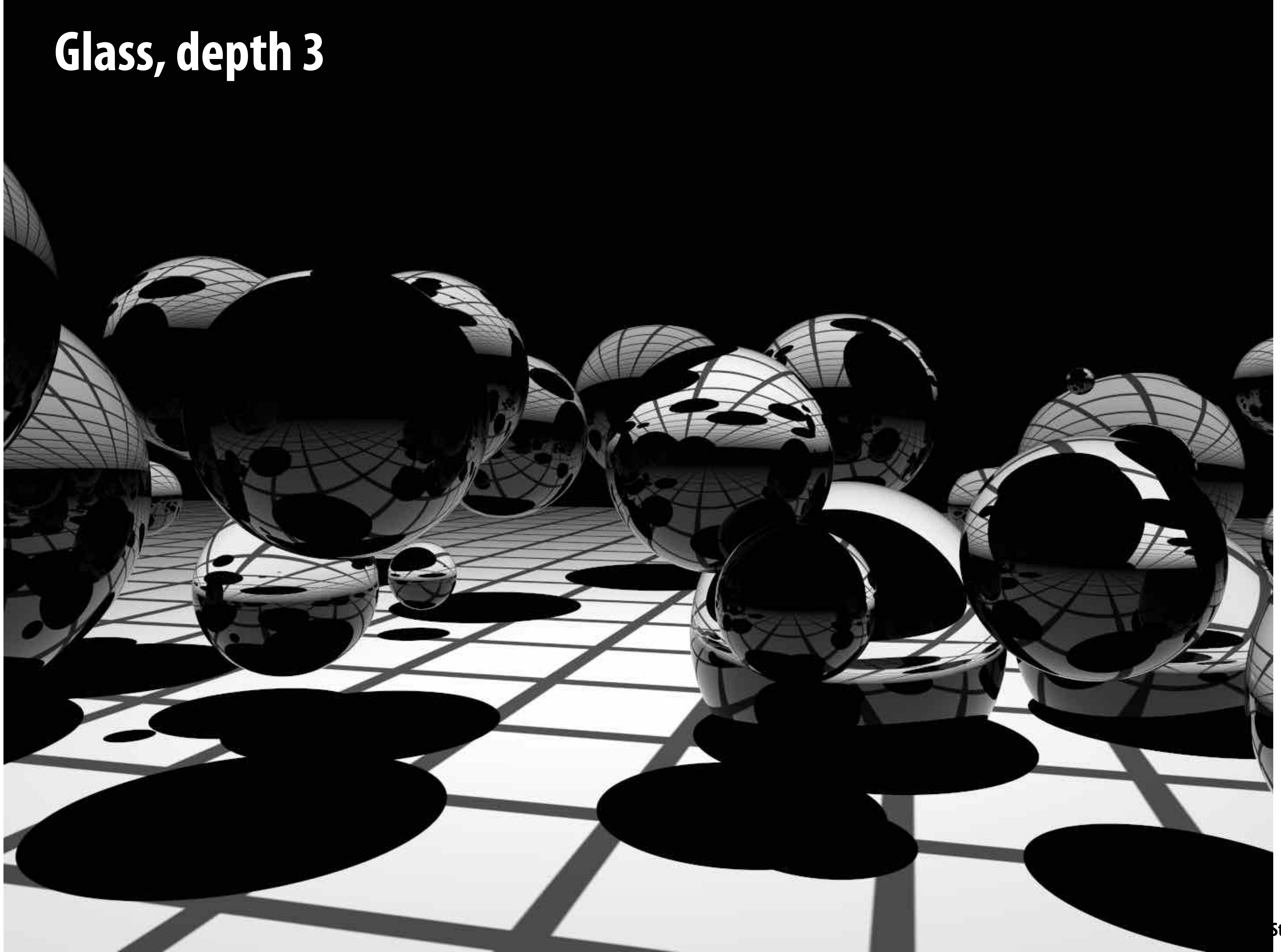
Glass, depth 1



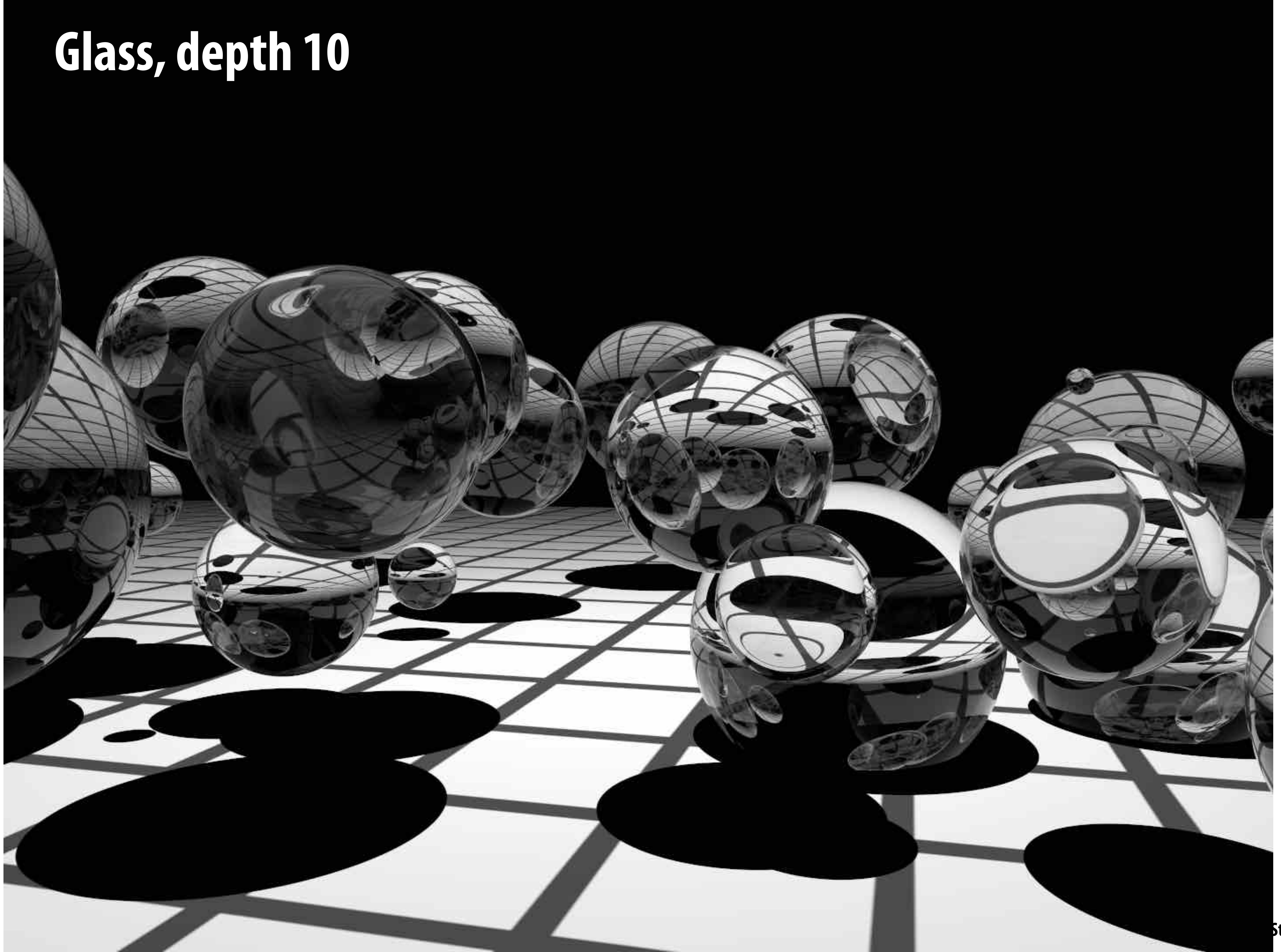
Glass, depth 2



Glass, depth 3



Glass, depth 10



Direct illumination + reflection + transparency

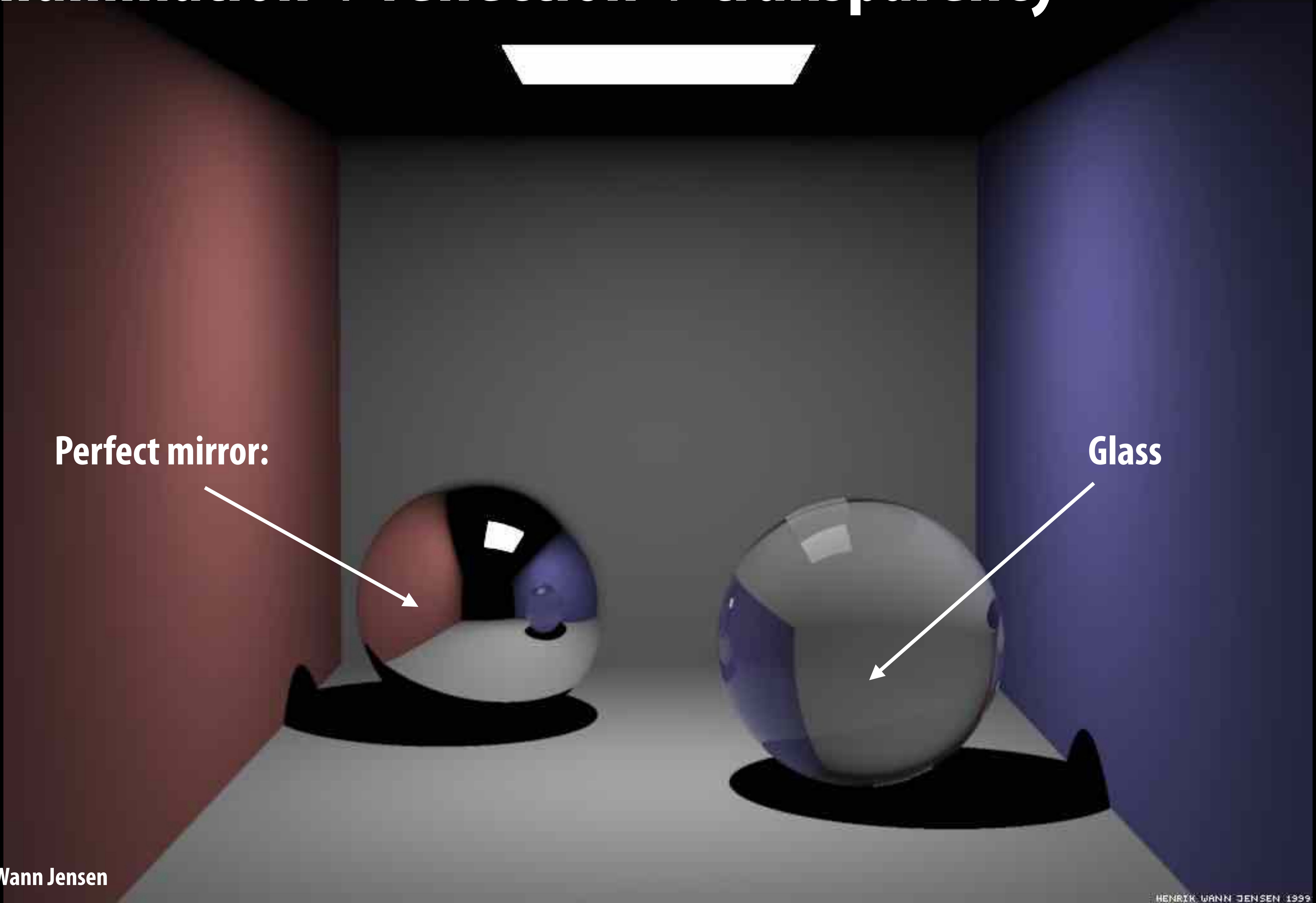


Image credit: Henrik Wann Jensen

With indirect illumination

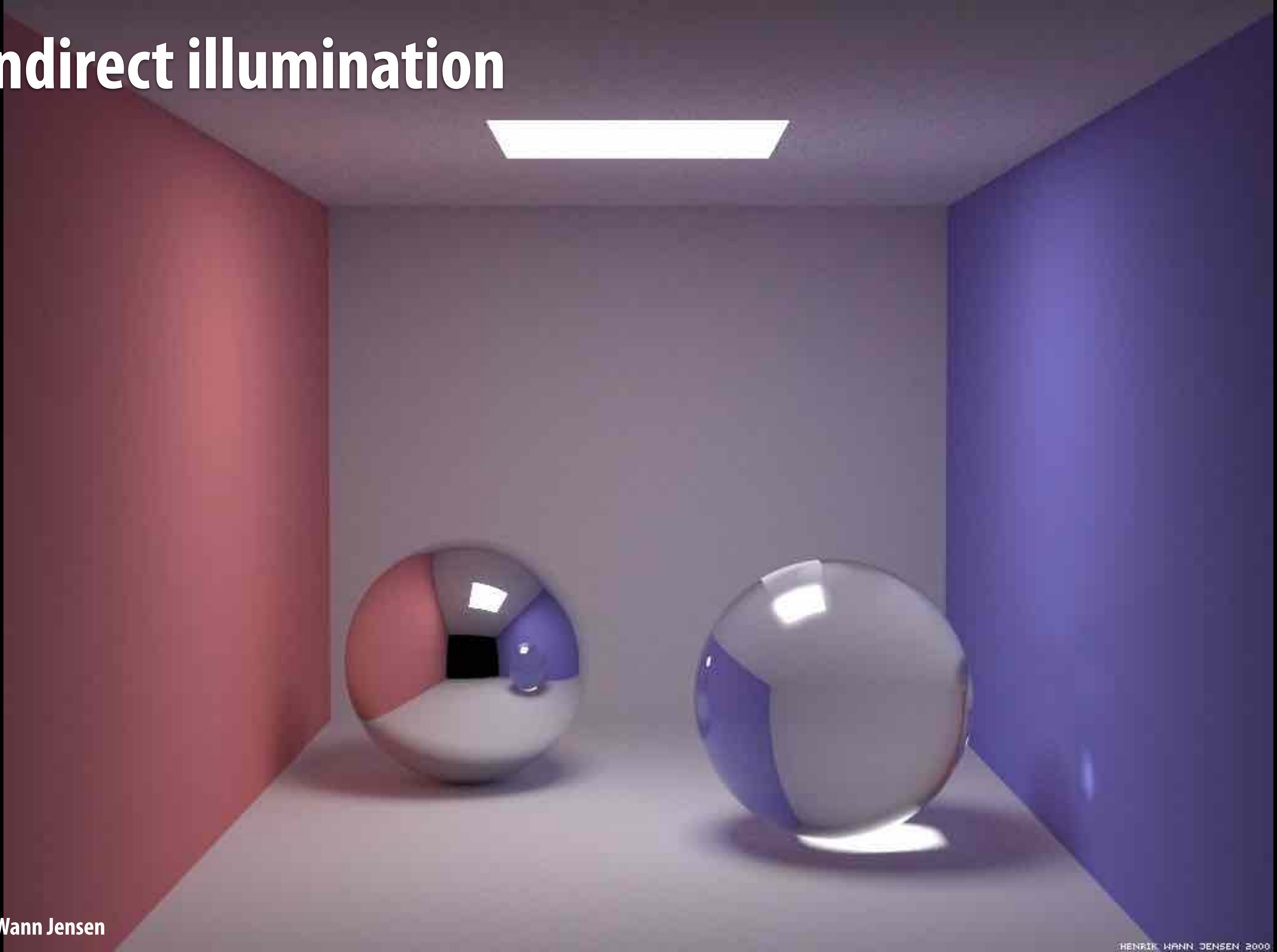
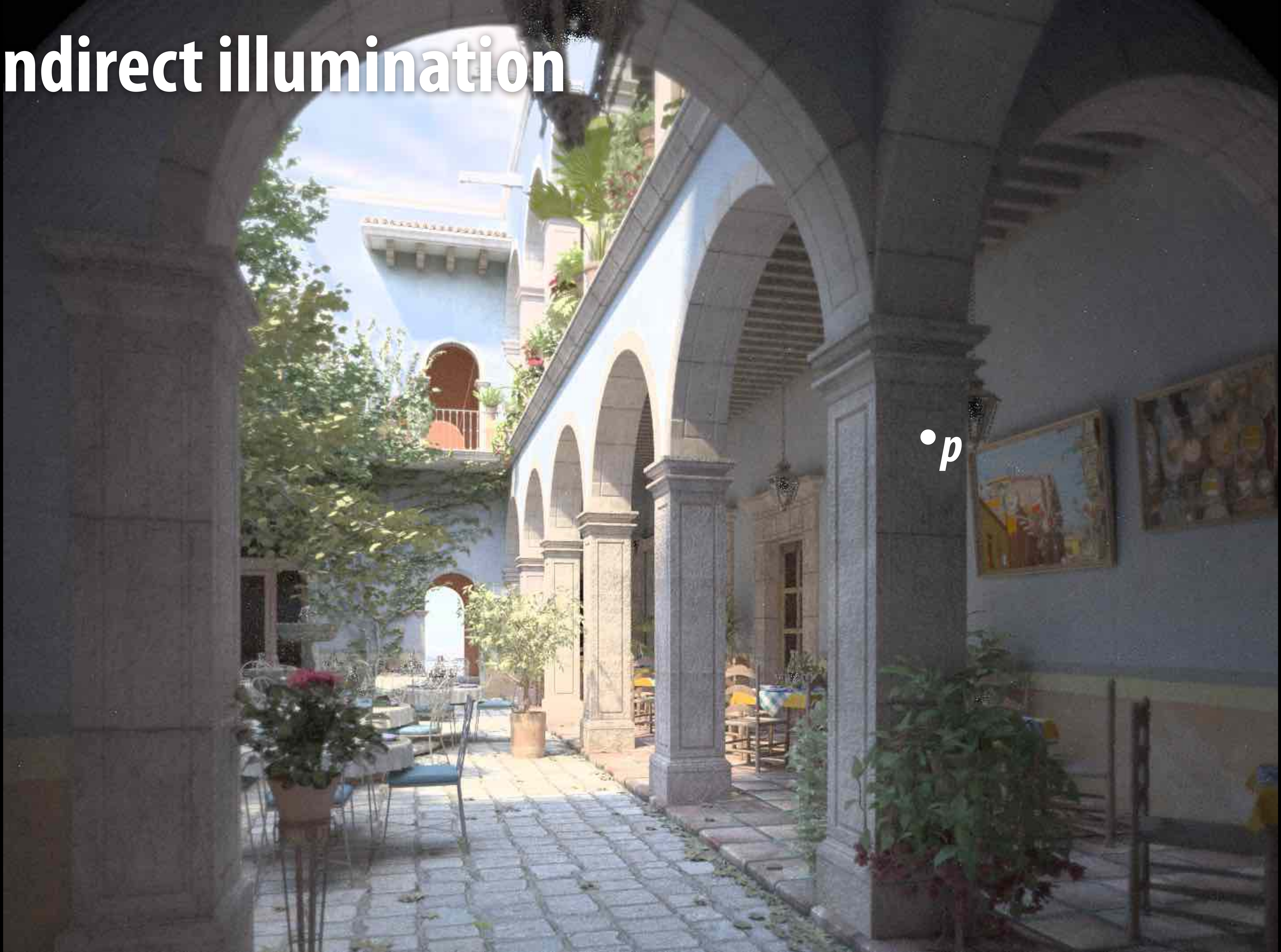


Image credit: Henrik Wann Jensen

Direct illumination only

With indirect illumination



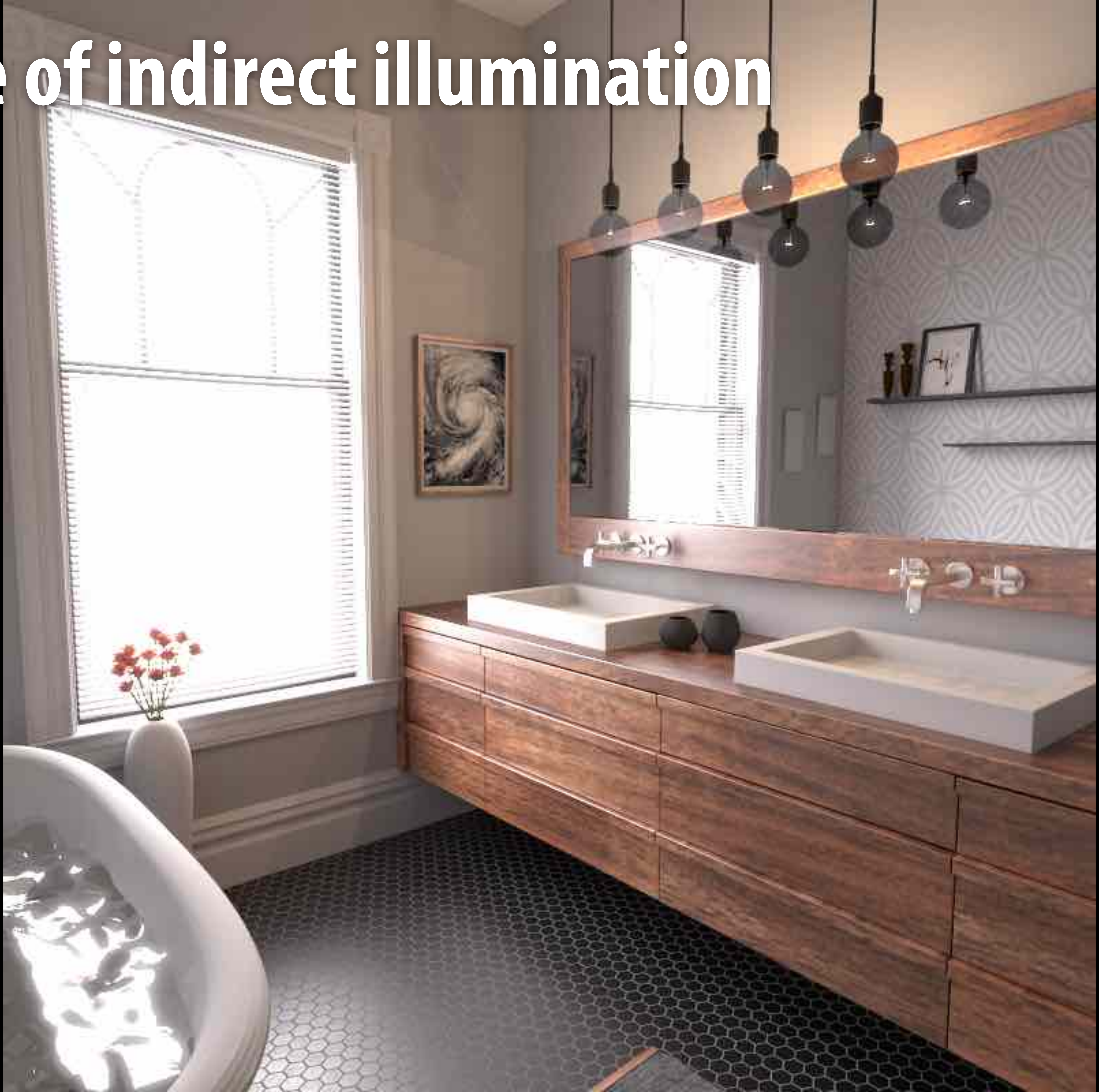
Importance of indirect illumination

Image credit: Brennan Shacklett

Importance of indirect illumination

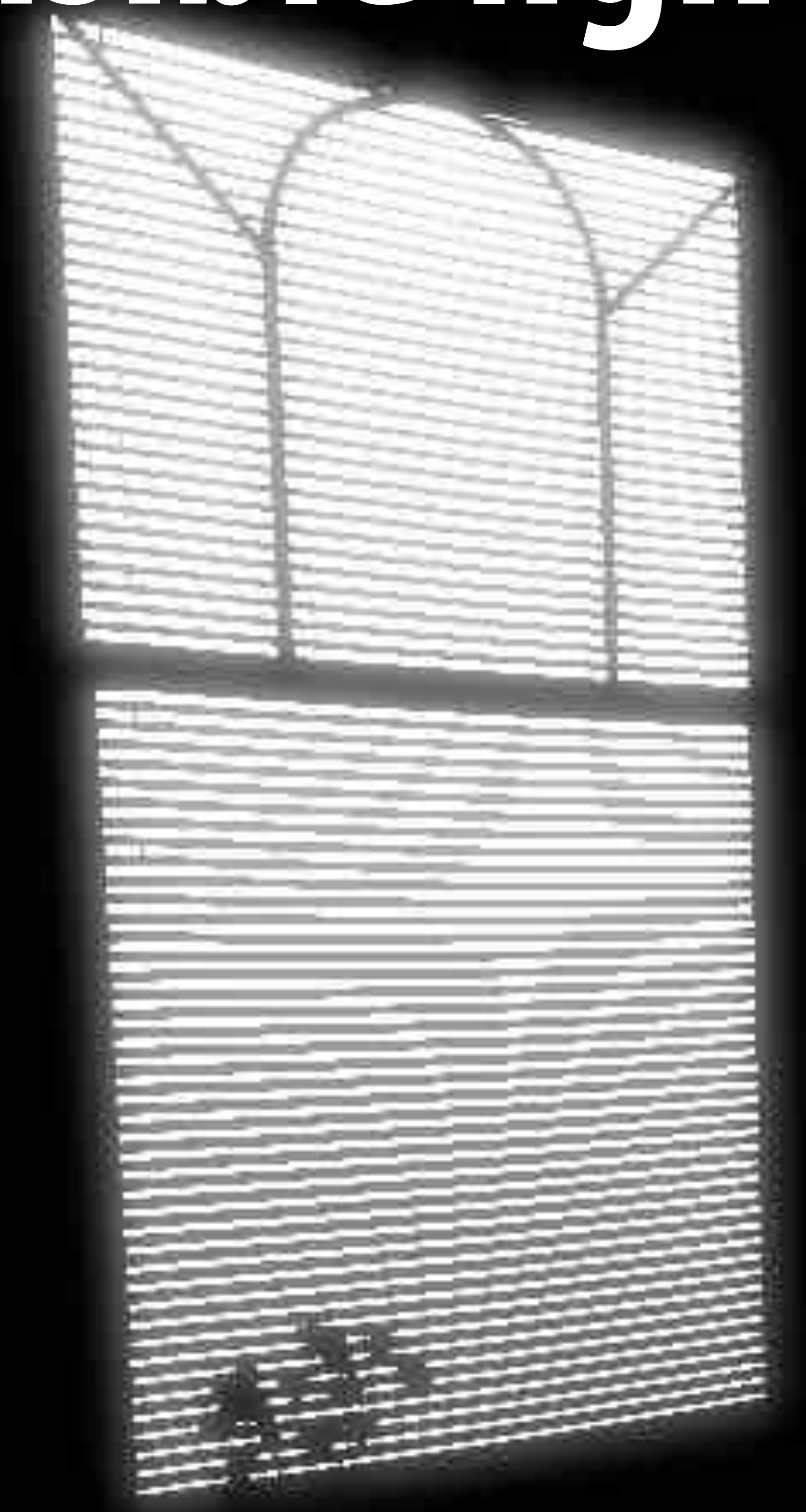
Image credit: Brennan Shacklett

Importance of indirect illumination

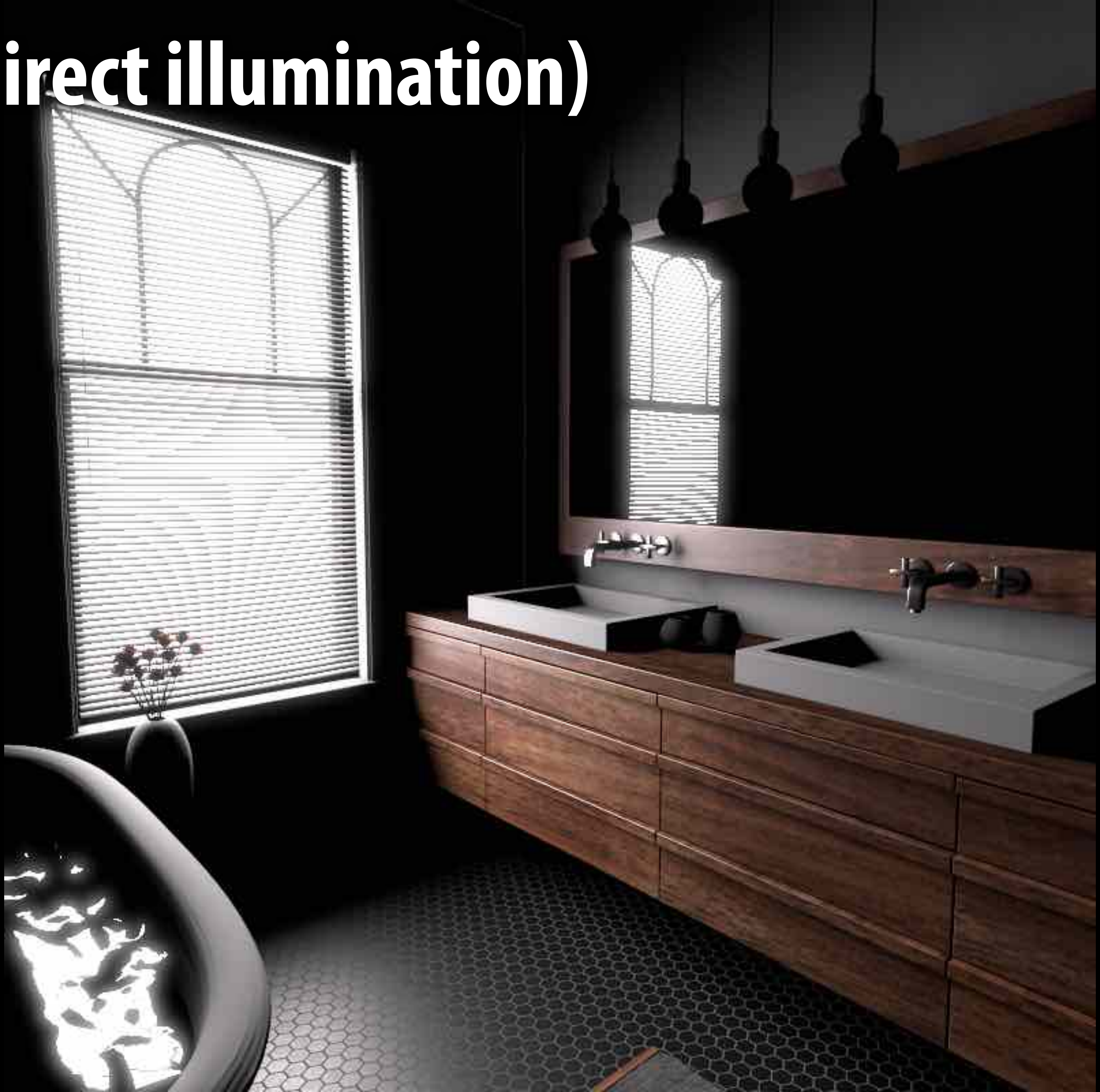


Scene breakdown: by bounce

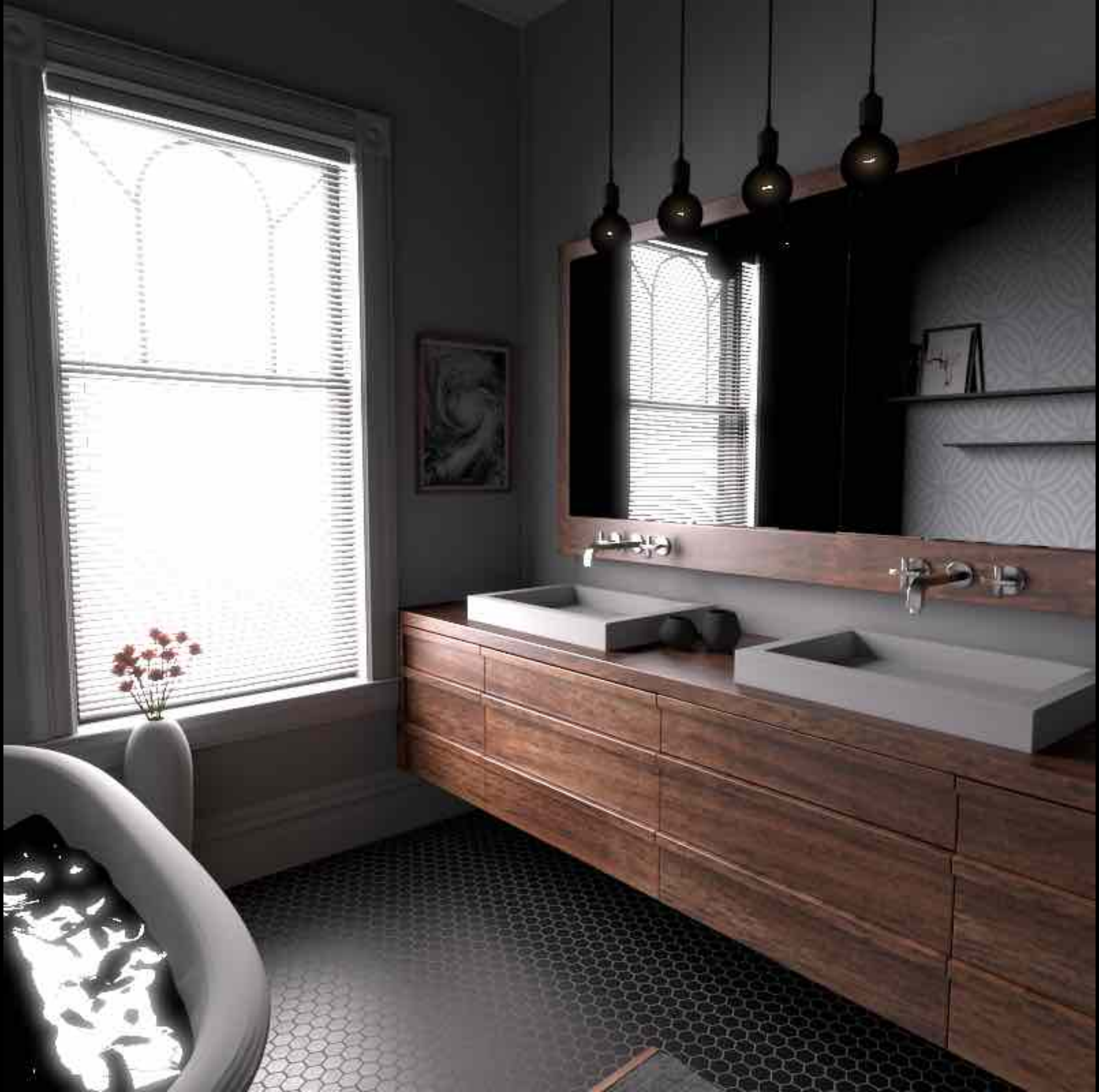
0 bounces (visible light sources)



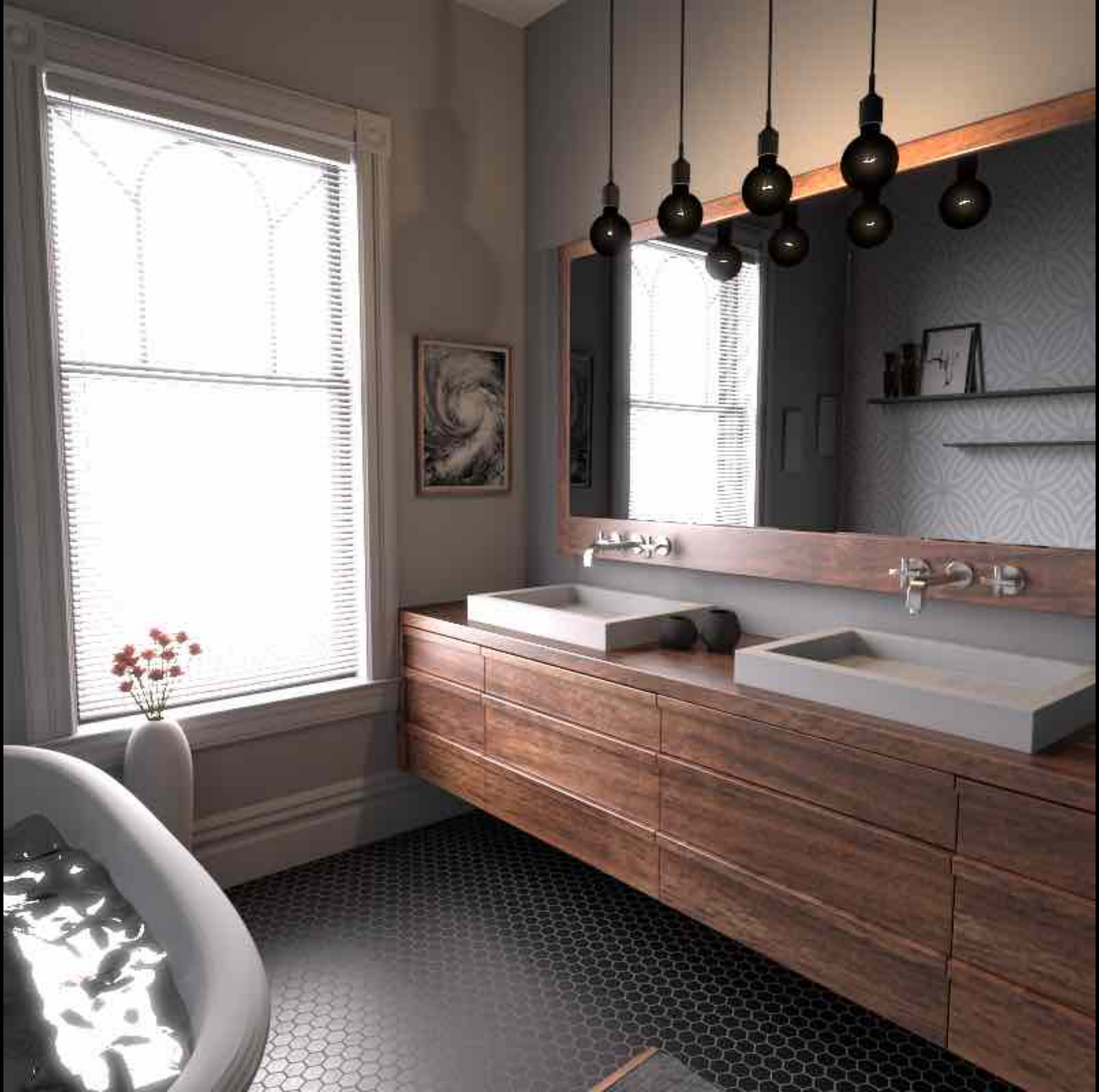
1 bounce (direct illumination)



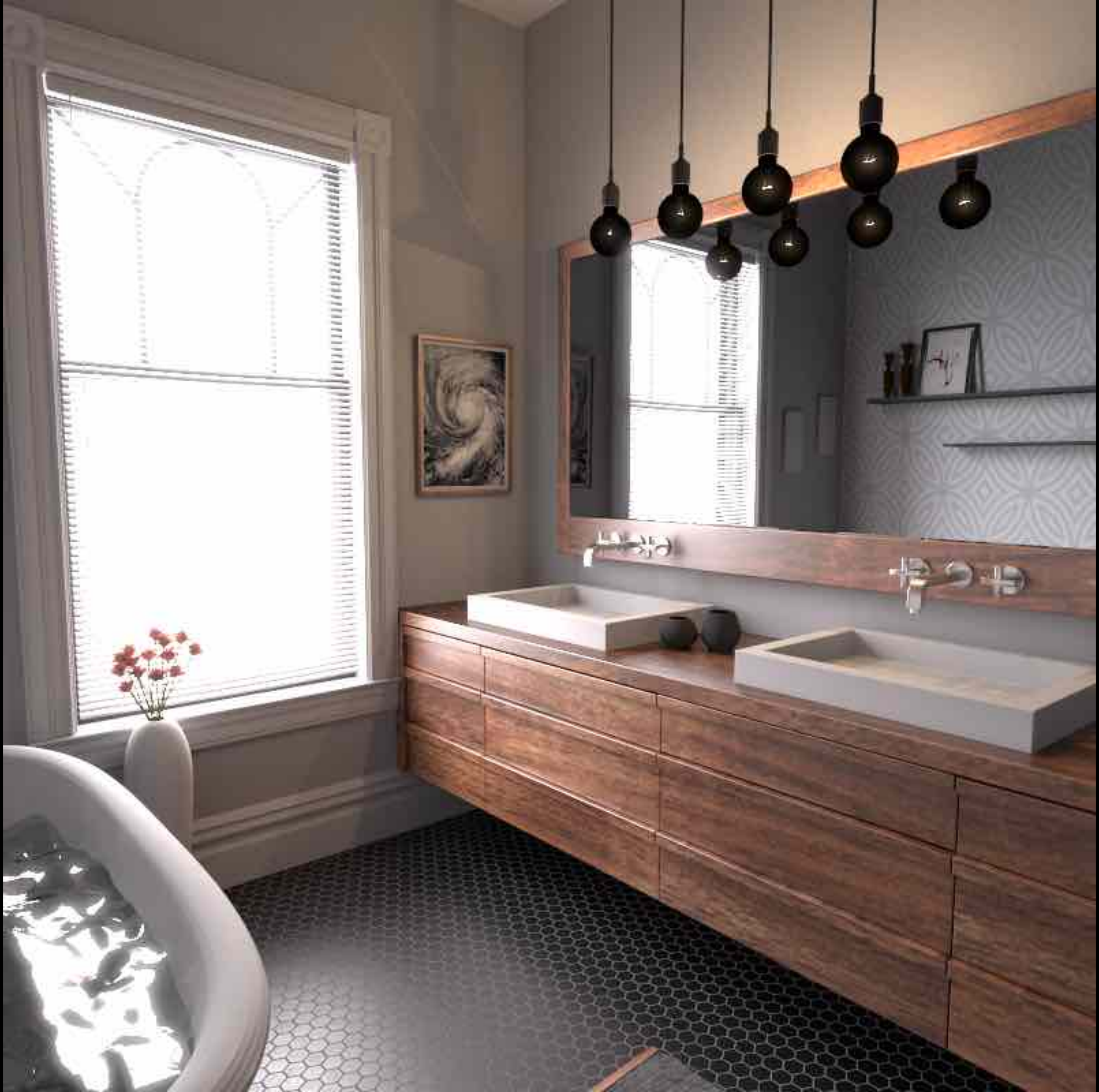
2 bounces



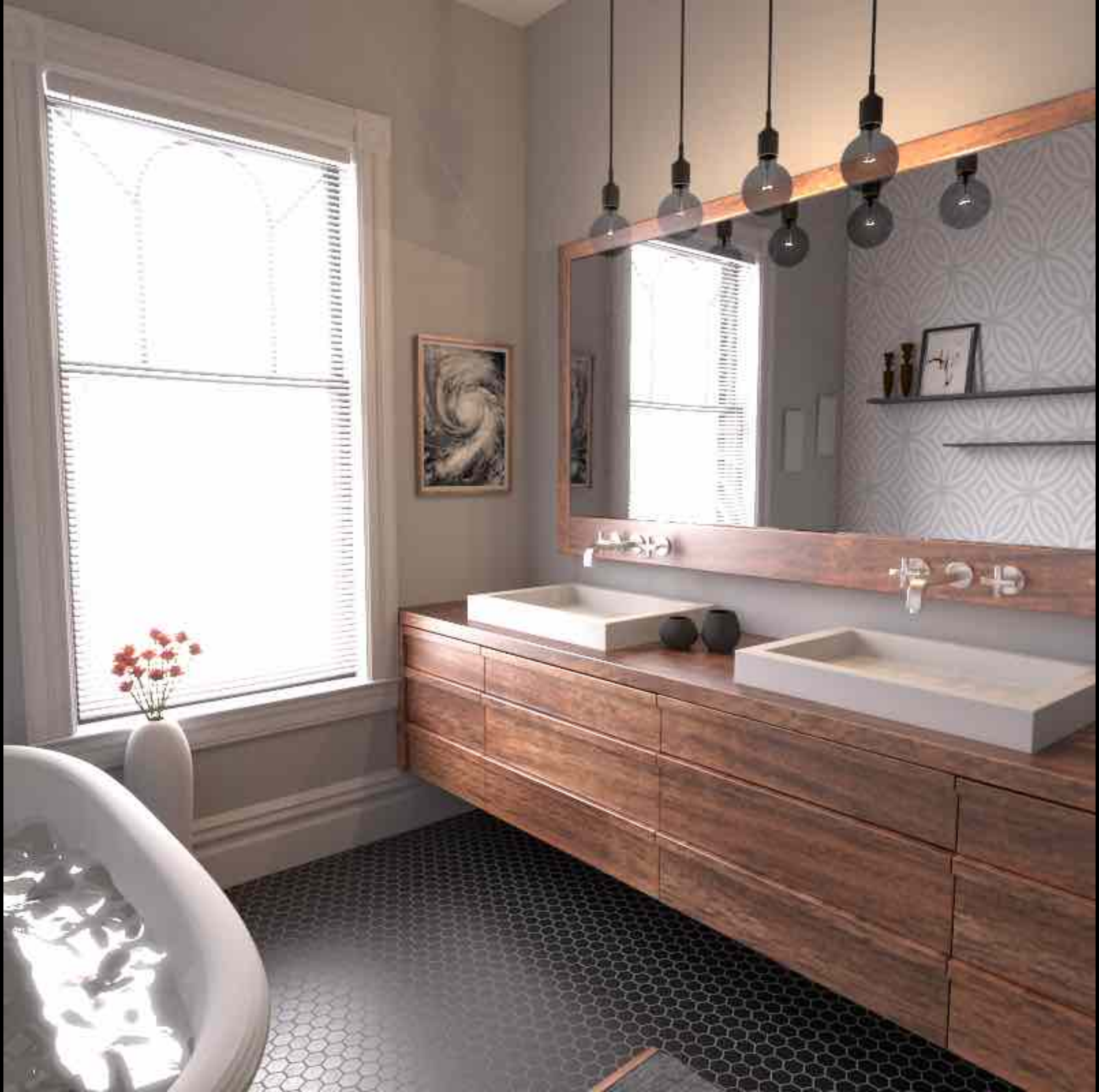
3 bounces



4 bounces



8 bounces



Energy balance

- Accountability
  - [outgoing] - [incoming] = [emitted] - [absorbed]

- Macro level:
  - The total light energy put into the system must equal the energy leaving

$$\Phi_o - \Phi_i = \Phi_e - \Phi_a$$

Energy balance

- **Accountability**
  - **[outgoing] - [incoming] = [emitted] - [absorbed]**

- **Micro level:**
  - **The energy flowing into a small region of space must equal the energy flowing out:**

$$E_o(\mathrm{p}) - E_i(\mathrm{p}) = E_e(\mathrm{p}) - E_a(\mathrm{p})$$

Surface balance equation

[outgoing] = [emitted] + [incoming] - [absorbed]

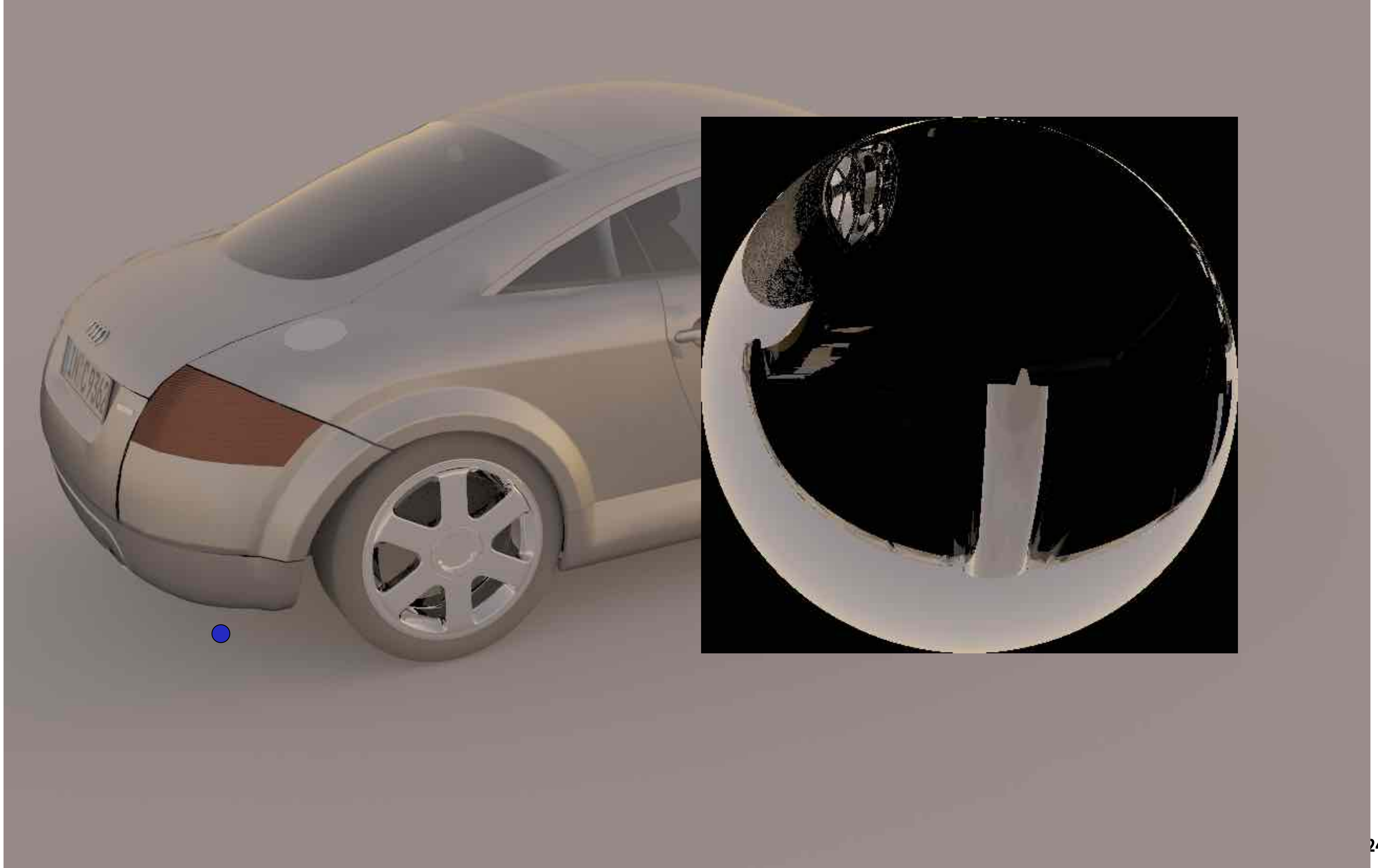
[reflected] = [incoming] - [absorbed]

[outgoing] = [emitted] + [reflected]

$$
\begin{aligned}
L_o(\mathrm{p}, \omega_o) &= L_e(\mathrm{p}, \omega_o) + L_r(\mathrm{p}, \omega_o) \\
&= L_e(\mathrm{p}, \omega_o) + \int_{H^2} f_r(\mathrm{p}, \omega_i \rightarrow \omega_o)\, \underset{\uparrow}{L_i}(\mathrm{p}, \omega_i)\, \cos\theta_i \, \mathrm{d}\omega_i
\end{aligned}
$$

The problem: need to know incident radiance!

Incident radiance function

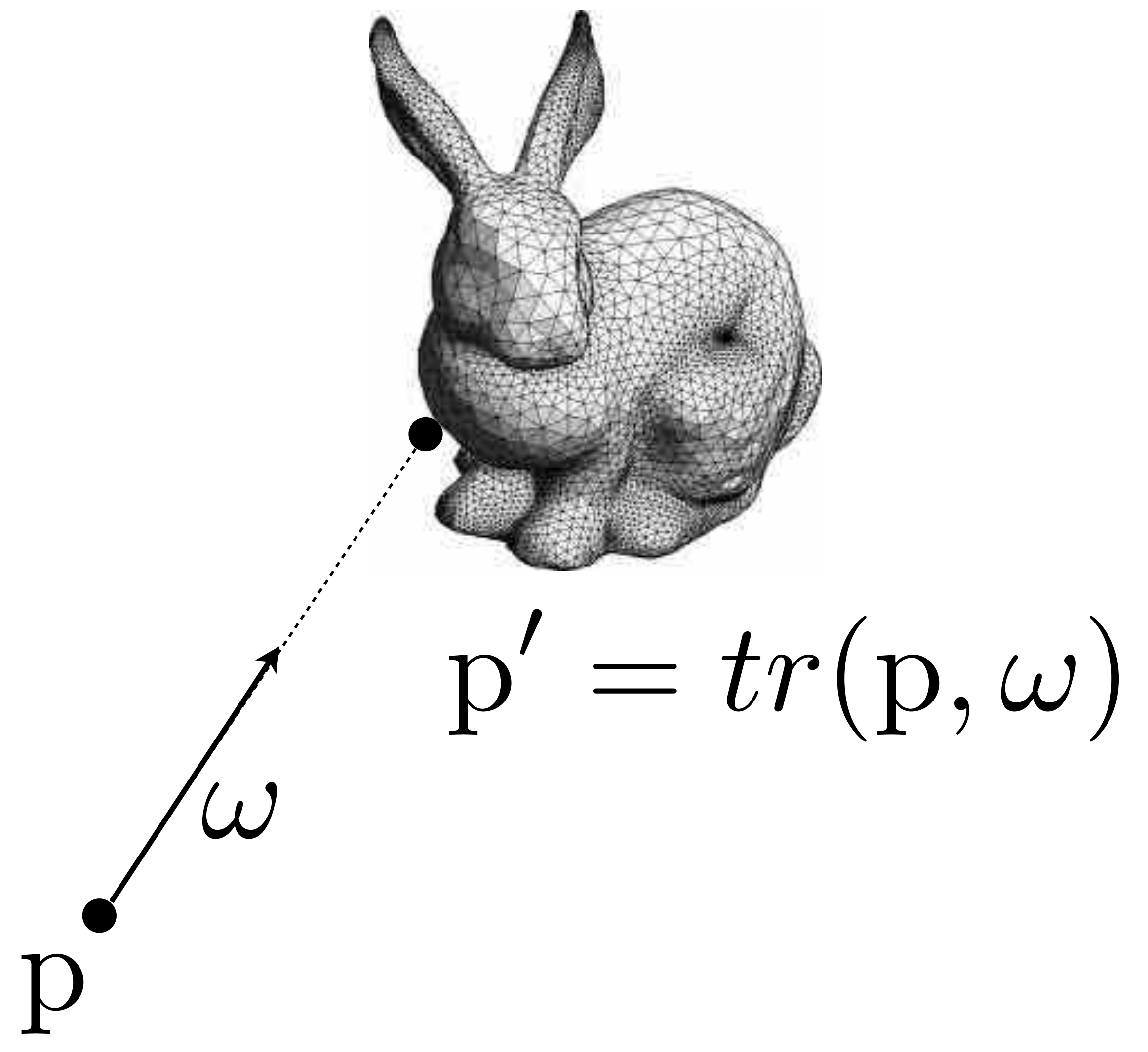


The rendering equation

Radiance invariance along rays:

$$L_i(\mathrm{p}, \omega_i) = L_o(tr(\mathrm{p}, \omega_i), -\omega_i)$$

"Radiance arriving at p from direction ω_i is the same as radiance leaving p' from direction $-\omega_i$."



Rewrite incident radiance in terms of exitant radiance at 1st visible surface:

$$L_o(\mathrm{p}, \omega_o) = L_e(\mathrm{p}, \omega_o) + \int_{H^2} \underbrace{f_r(\mathrm{p}, \omega_i \to \omega_o)}_{\text{Light scattering}} \, \underbrace{L_o(tr(\mathrm{p}, \omega_i), -\omega_i)}_{\text{Light transport}} \, \cos\theta_i \, \mathrm{d}\omega_i$$

The rendering equation

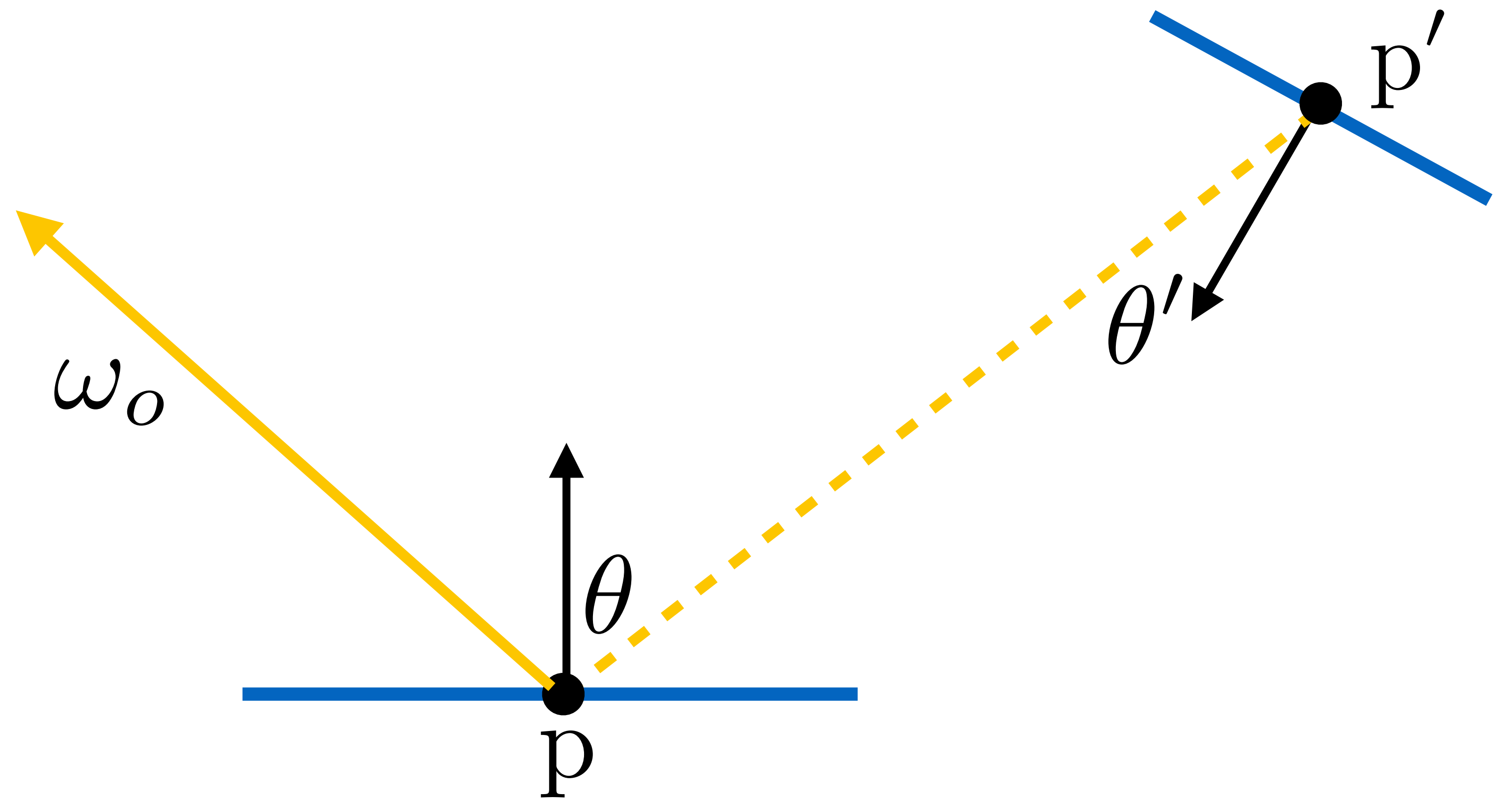
$$L(\mathrm{p}, \omega_o) = L_e(\mathrm{p}, \omega_o) + \int_{H^2} f_r(\mathrm{p}, \omega_i \to \omega_o)\, L(tr(\mathrm{p}, \omega_i), -\omega_i)\ \cos\theta_i \, \mathrm{d}\omega_i$$

The rendering equation: area form

- **Can rewrite rendering equation as an integral over surface area of objects in the scene**
  - **Apply change of variables** $\mathrm{d}\omega = \frac{\cos\theta}{r^2}\mathrm{d}A$
  - **Introduce binary visibility function:** $V(\mathrm{p} \leftrightarrow \mathrm{p}')$

$$L_o(\mathrm{p},\omega_o) = L_e(\mathrm{p},\omega_o) + \int_A f_r(\mathrm{p}, \underbrace{(\mathrm{p}'-\mathrm{p})}_{\omega_i} \to \omega_o)\, L_o(\mathrm{p}', \underbrace{(\mathrm{p}-\mathrm{p}')}_{-\omega_i})\, \cos\theta\, V(\mathrm{p} \leftrightarrow \mathrm{p}')\, \frac{\cos\theta'}{|\mathrm{p}-\mathrm{p}'|^2}\mathrm{d}\mathrm{p}'$$

$$= L_e(\mathrm{p},\omega_o) + \int_A f_r(\mathrm{p}, (\mathrm{p}'-\mathrm{p}) \to \omega_o)\, L_o(\mathrm{p}', (\mathrm{p}-\mathrm{p}'))\, \boxed{G(\mathrm{p} \leftrightarrow \mathrm{p}')}\, \mathrm{d}\mathrm{p}'$$



$$\boxed{G(\mathrm{p} \leftrightarrow \mathrm{p}') = V(\mathrm{p} \leftrightarrow \mathrm{p}')\frac{\cos\theta\cos\theta'}{|\mathrm{p}-\mathrm{p}'|^2}}$$

The rendering equation as a sum over paths

$$\begin{aligned} L_o(\mathrm{p}_1 \to \mathrm{p}) =& L_e(\mathrm{p}_1 \to \mathrm{p}) \\ &+ \int_A L_e(\mathrm{p}_2 \to \mathrm{p}_1) f(\mathrm{p}_2 \to \mathrm{p}_1 \to \mathrm{p}) G(\mathrm{p}_1 \leftrightarrow \mathrm{p}_2) \mathrm{d}A(\mathrm{p}_2) \\ &+ \int_A \int_A L_e(\mathrm{p}_3 \to \mathrm{p}_2) f(\mathrm{p}_3 \to \mathrm{p}_2 \to \mathrm{p}_1) G(\mathrm{p}_2 \leftrightarrow \mathrm{p}_3) f(\mathrm{p}_2 \to \mathrm{p}_1 \to \mathrm{p}) G(\mathrm{p}_1 \leftrightarrow \mathrm{p}_2) \mathrm{d}A(\mathrm{p}_3) \mathrm{d}A(\mathrm{p}_2) \\ &+ \cdots \end{aligned}$$

Path of length 1

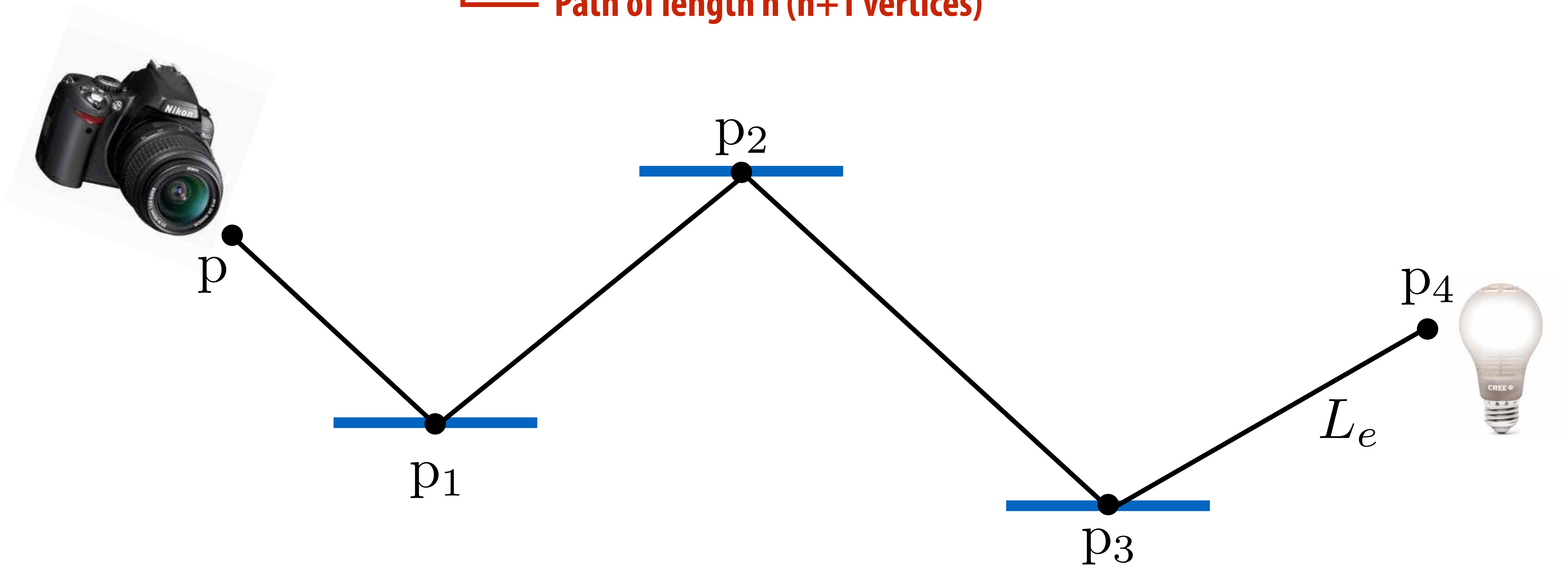
Paths of length 2

Paths of length 3

Energy reaching p from all paths of length n

$$L_o(\mathrm{p}_1 \to \mathrm{p}) = \sum_{n=1}^{\infty} P(\bar{\mathrm{p}}_\mathrm{n})$$

Path of length n (n+1 vertices)



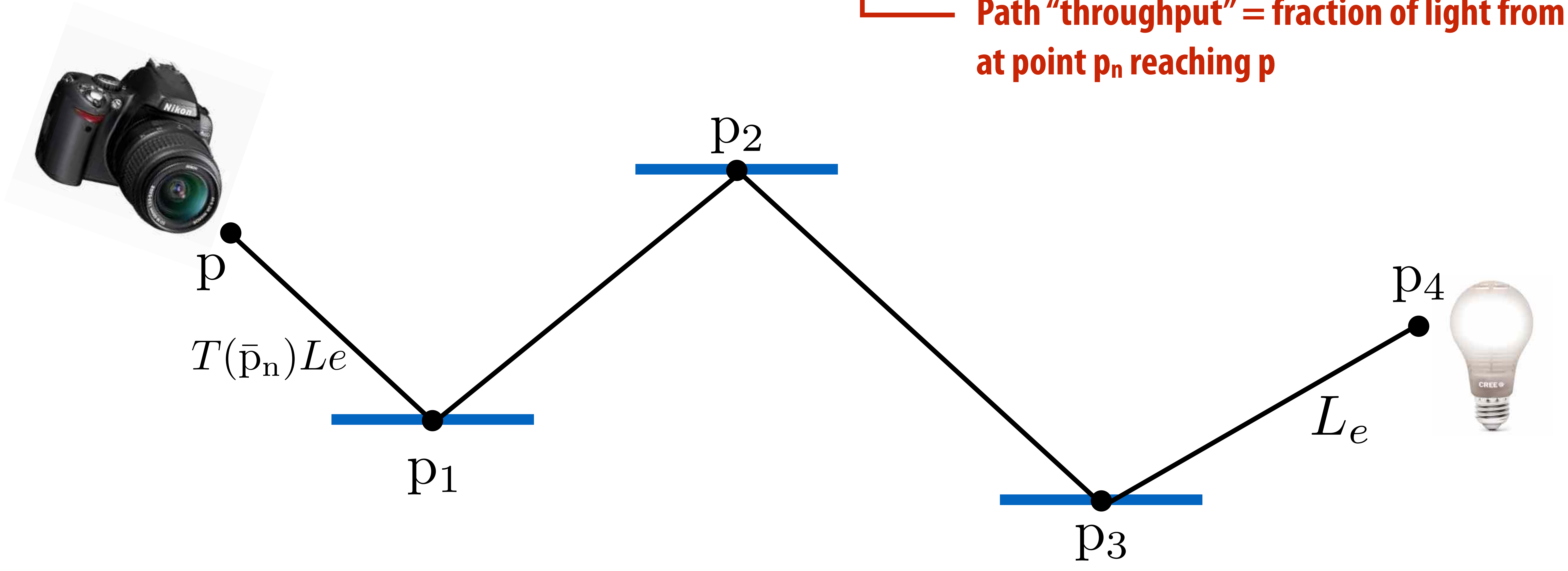
The rendering equation as a sum over paths

$$P(\bar{\mathrm{p}}_\mathrm{n}) = \int_A \int_A \int_A \cdots \int_A L_e(\mathrm{p_n} \to \mathrm{p_{n-1}})$$

$$\times \left(\prod_{i=1}^{n-1} f(\mathrm{p_{i+1}} \to \mathrm{p_i} \to \mathrm{p_{i-1}}) G(\mathrm{p_{i+1}} \leftrightarrow \mathrm{p_i}) \right) \mathrm{d}A(\mathrm{p_2}) \cdots \mathrm{d}A(\mathrm{p_n})$$

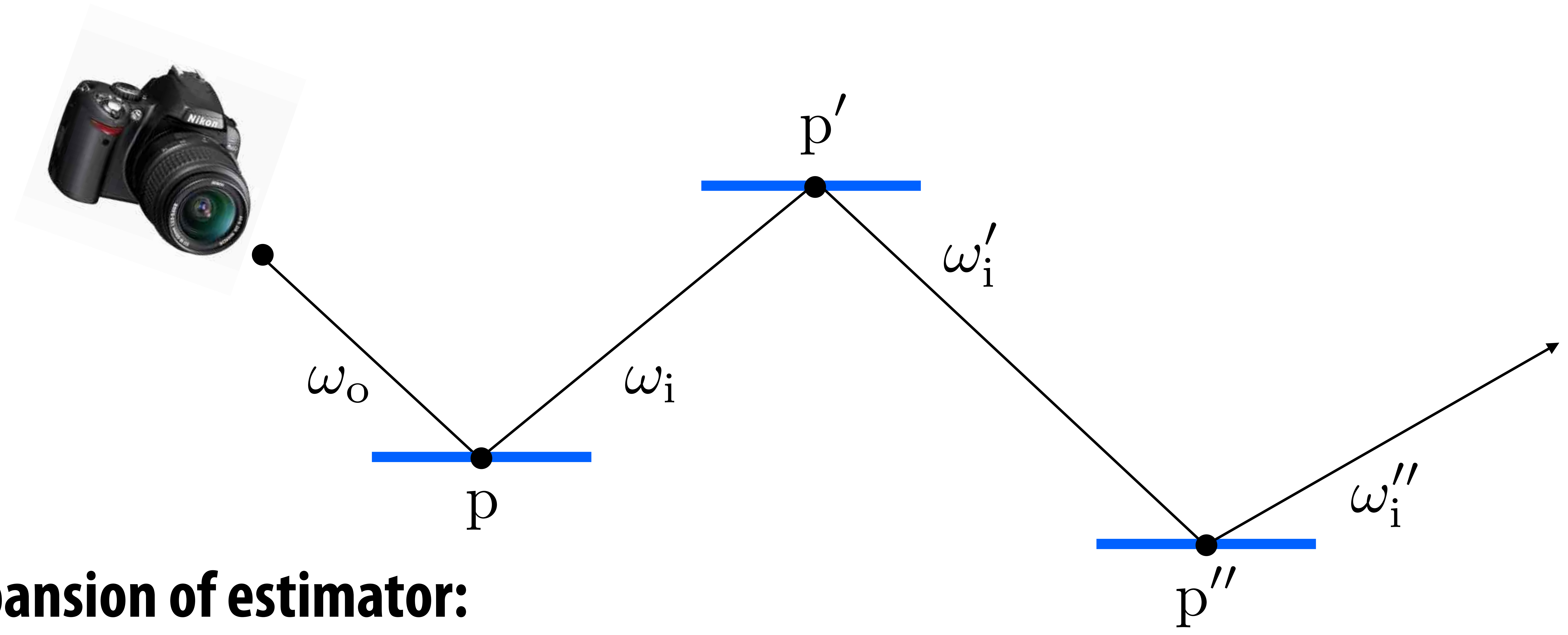
$$= \int_A \int_A \int_A \cdots \int_A L_e(\mathrm{p_n} \to \mathrm{p_{n-1}}) T(\bar{\mathrm{p}}_\mathrm{n}) \mathrm{d}A(\mathrm{p_2}) \cdots \mathrm{d}A(\mathrm{p_n})$$

Path "throughput" = fraction of light from light source at point $\mathrm{p_n}$ reaching p



How do we sample paths?

Idea: generate random path incrementally starting at camera



Recursive expansion of estimator:

$$L_o(\mathrm{p}, \omega_o) = L_e(\mathrm{p}, \omega_o) + \frac{f_r(\omega_o, \omega_i) \cos\theta_i}{p(\omega_i)} L_o(\mathrm{p}', -\omega_i)$$

$$= L_e + \frac{f_r \cos\theta_i}{p(\omega_i)} \left[L_e(\mathrm{p}', -\omega_i) + \frac{f_r(-\omega_i, \omega_i') \cos\theta_i'}{p(\omega_i')} L_o(\mathrm{p}'', \omega_i'') \right]$$

$$= \cdots$$

Basic recursive path tracing

```
Spectrum PathLo(Ray ray) {
  Intersection isect = scene->Intersect(ray);
  BSDF bsdf = isect.GetBSDF();
  Vector3f wo = -ray.d, wi;
  Float pdf;
  Spectrum fr = bsdf.Sample_f(wo, &wi, &pdf);

  return isect.Le(wo) +
    fr * PathLo(Ray(isect.P, wi)) * Dot(wi, isect.N) / pdf;
}
```

Note how over the entire path the indirect illumination is modulated by product of probabilities of individual path segments.

Kajiya, 1986

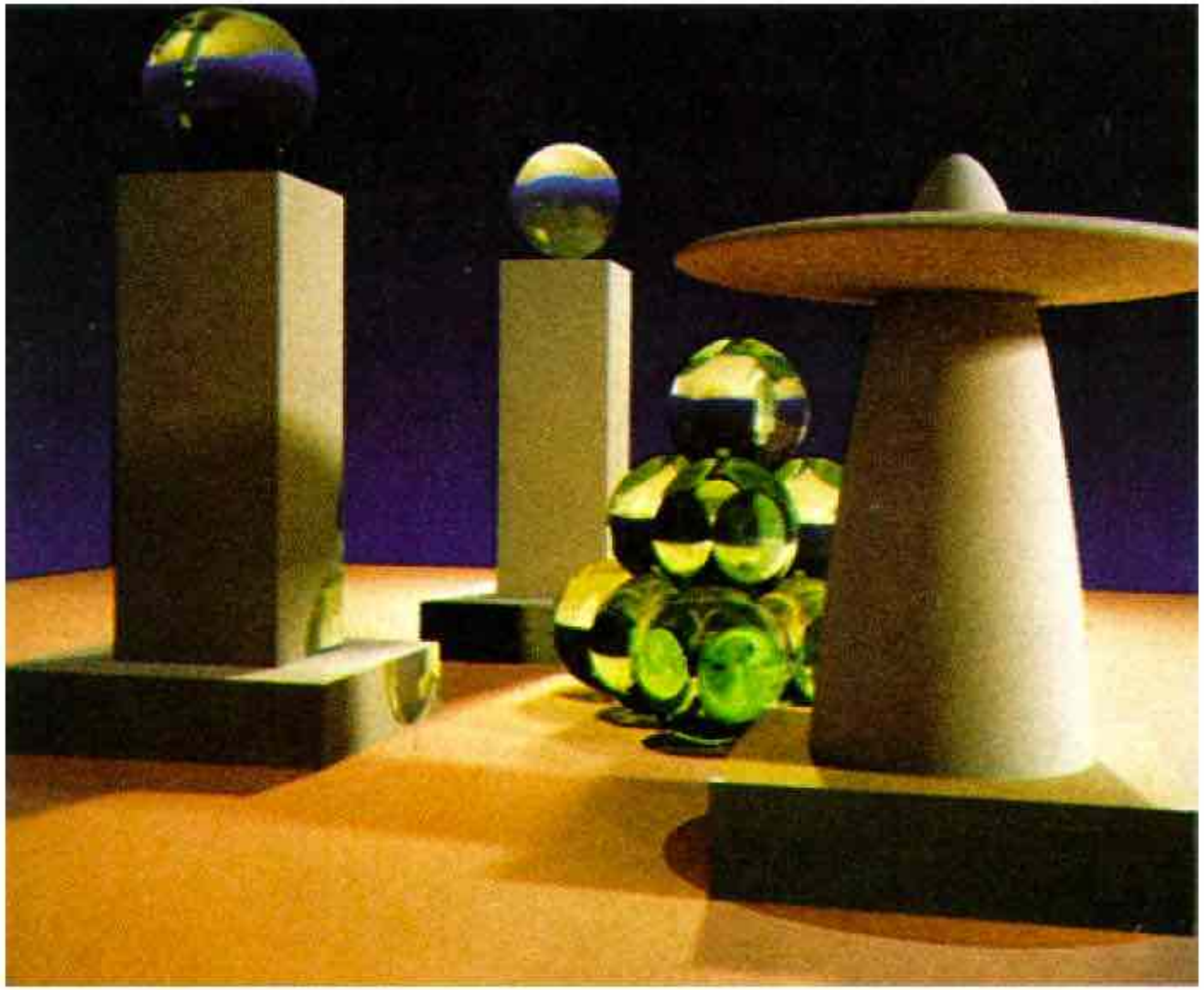
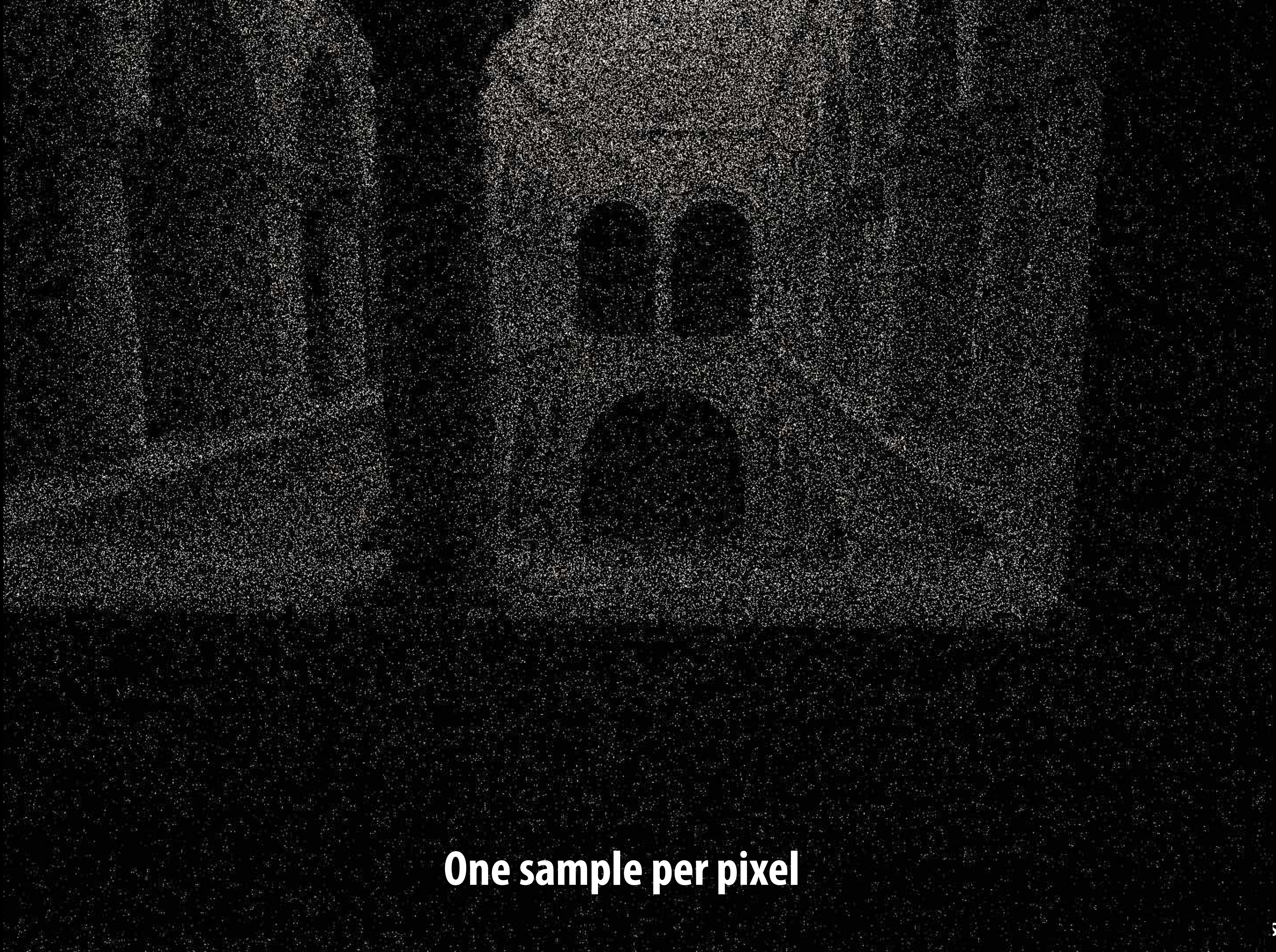
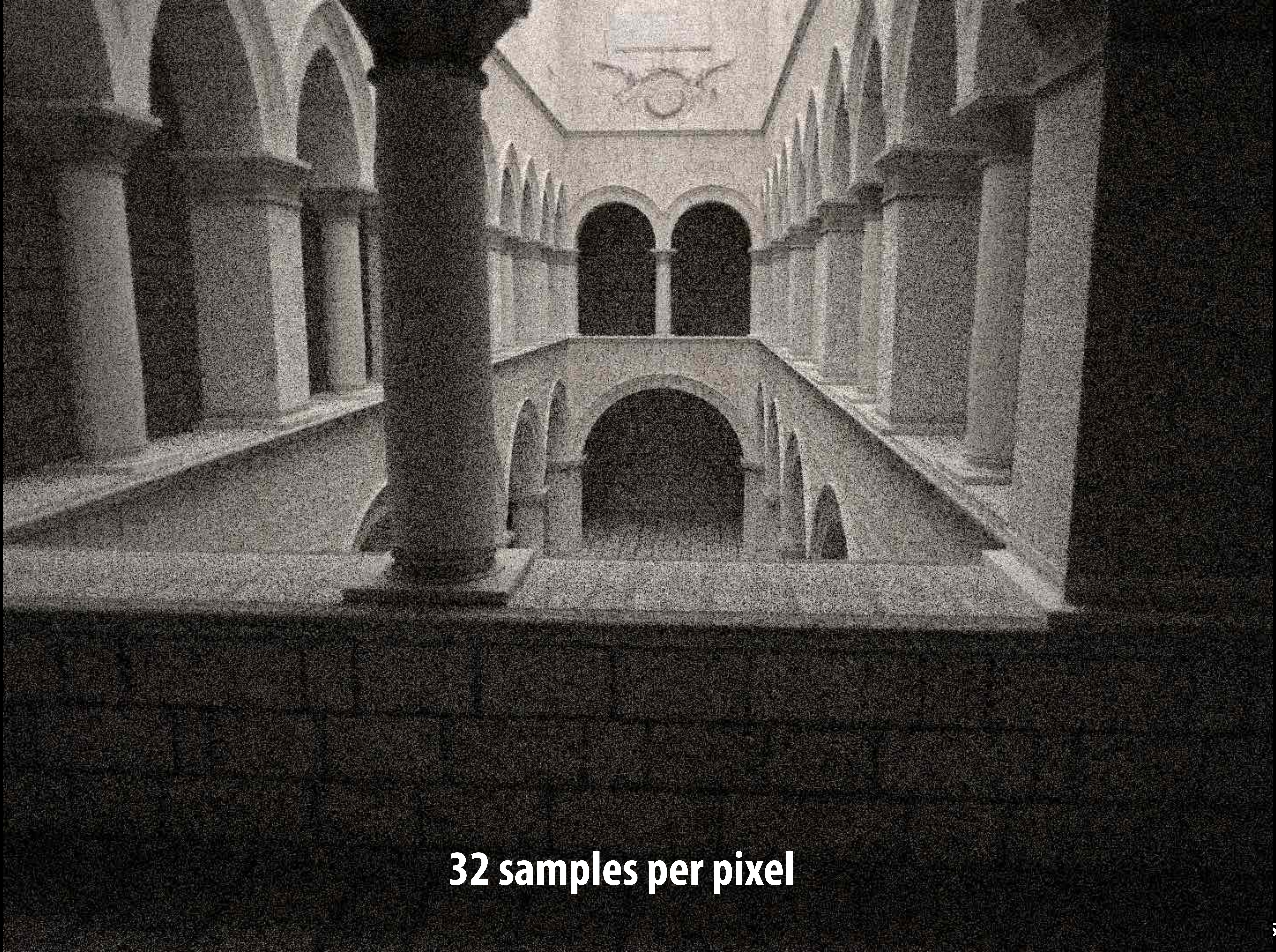
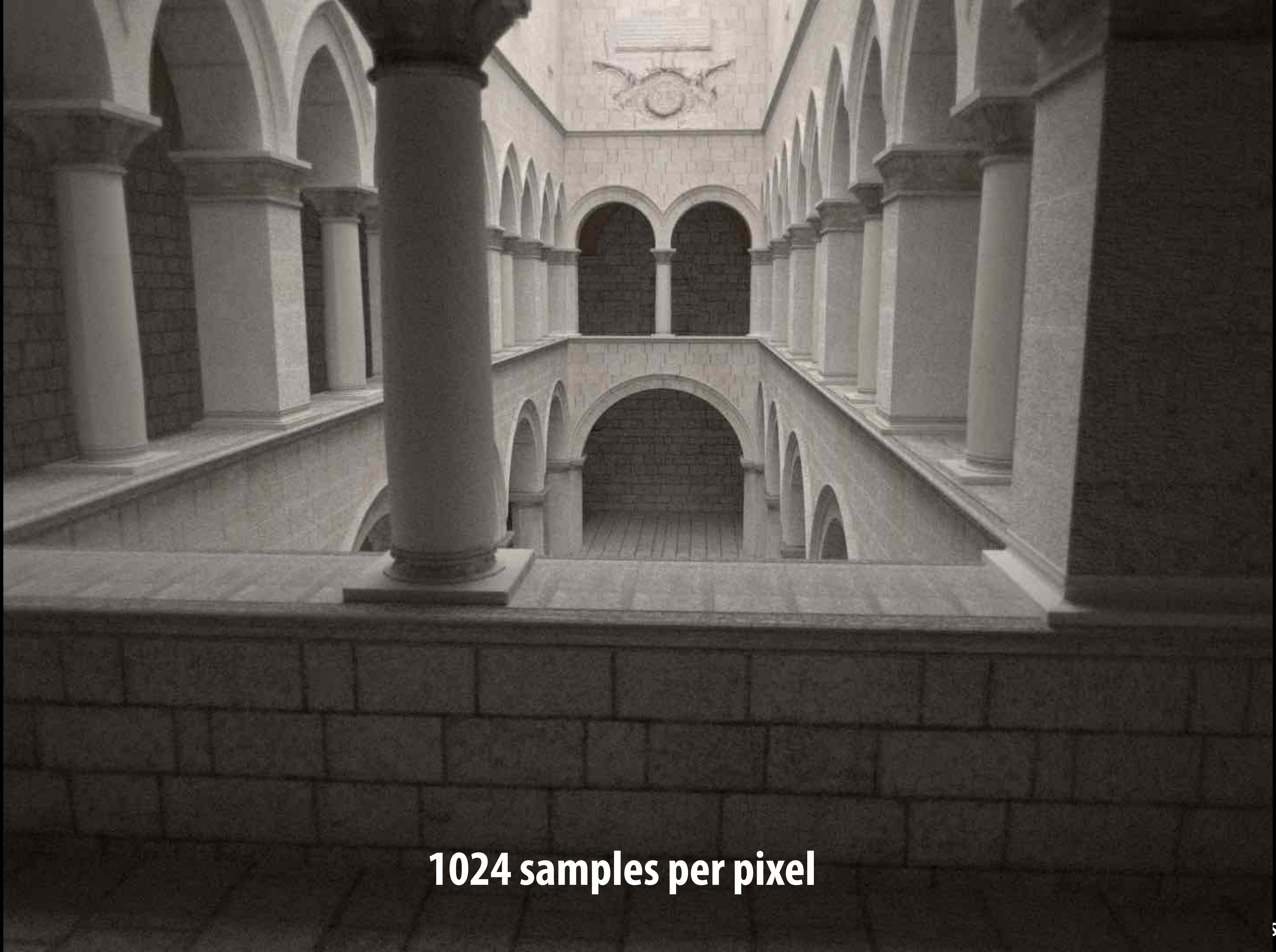


Figure 6. A sample image. All objects are neutral grey. Color on the objects is due to caustics from the green glass balls and color bleeding from the base polygon.


One sample per pixel

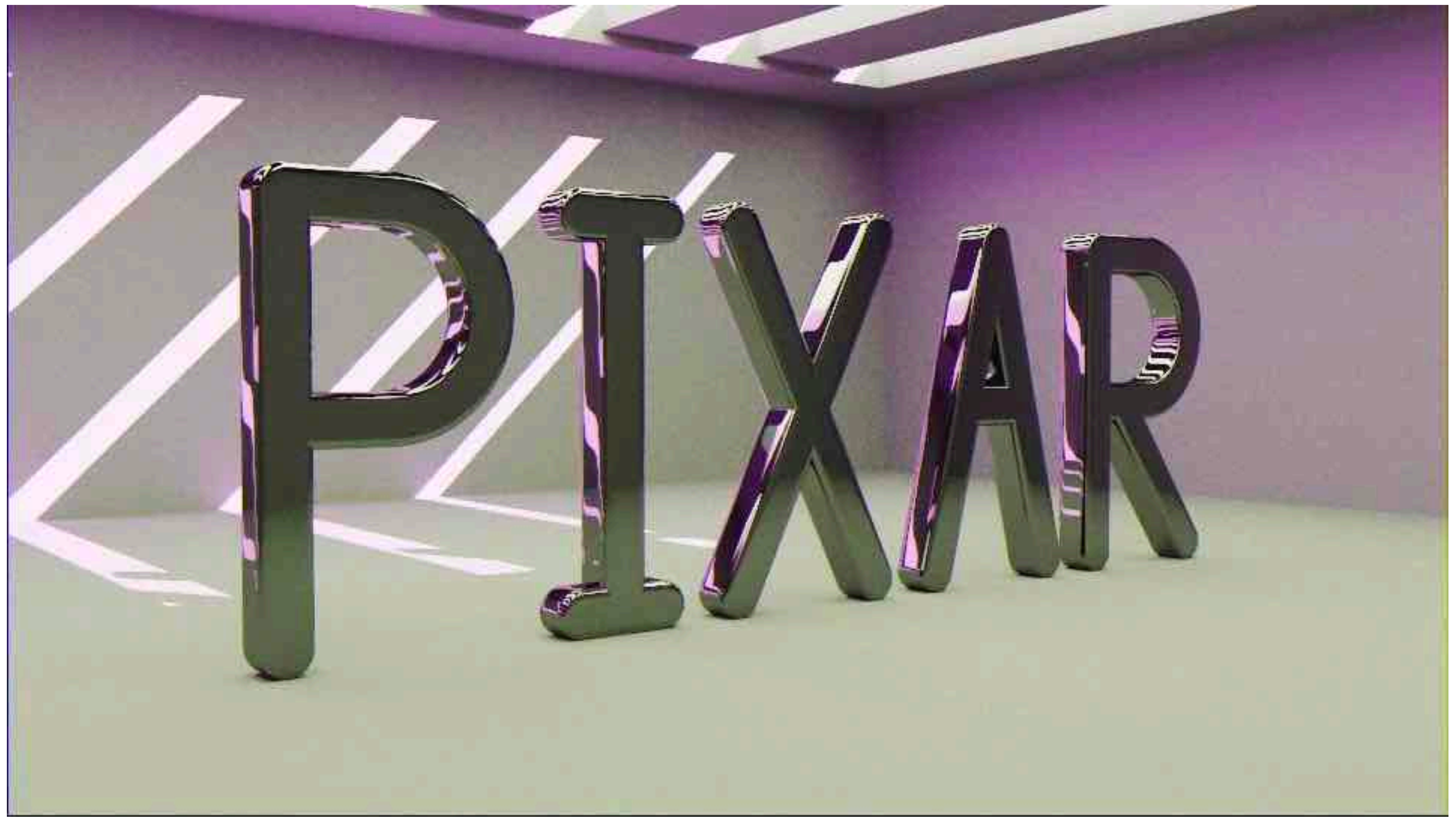

32 samples per pixel



1024 samples per pixel

The postcard-sized path tracer

```
#include <stdlib.h> // card > pixar.ppm
#include <stdio.h>
#include <math.h>
#define R return
#define O operator
typedef float F;typedef int I;struct V{F x,y,z;V(F v=0){x=y=z=v;}V(F a,F b,F
c=0){x=a;y=b;z=c;}V O+(V r){R V(x+r.x,y+r.y,z+r.z);}V O*(V r){R V(x*r.x,y*r.
y,z*r.z);}F O%(V r){R x*r.x+y*r.y+z*r.z;}V O!(){R*this*(1/sqrtf(*this%*this)
);}};F L(F l,F r){R l<r?l:r;}F U(){R(F)rand()/RAND_MAX;}F B(V p,V l,V h){l=p
+l*-1;h=h+p*-1;R-L(L(L(l.x,h.x),L(l.y,h.y)),L(l.z,h.z));}F S(V p,I&m){F d=1\
e9;V f=p;f.z=0;char l[]="5O5_5W9W5_9_COC_AOEOA_E_IOQ_I_QOUOY_Y_]OWW[WaOa_aW\
eWa_e_cWiO";for(I i=0;i<60;i+=4){V b=V(l[i]-79,l[i+1]-79)*.5,e=V(l[i+2]-79,l
[i+3]-79)*.5+b*-1,o=f+(b+e*L(-L((b+f*-1)%e/(e%e),0),1))*-1;d=L(d,o%o);}d=sq\
rtf(d);V a[]={V(-11,6),V(11,6)};for(I i=2;i--;){V o=f+a[i]*-1;d=L(d,o.x>0?f\
absf(sqrtf(o%o)-2):(o.y+=o.y>0?-2:2,sqrtf(o%o)));}d=powf(powf(d,8)+powf(p.z,
8),.125)-.5;m=1;F r=L(-L(B(p,V(-30,-.5,-30),V(30,18,30)),B(p,V(-25,17,-25),V
(25,20,25))),B(V(fmodf(fabsf(p.x),8),p.y,p.z),V(1.5,18.5,-25),V(6.5,20,25)))
;if(r<d)d=r,m=2;F s=19.9-p.y;if(s<d)d=s,m=3;R d;}I M(V o,V d,V&h,V&n){I m,s=
0;F t=0,c;for(;t<100;t+=c)if((c=S(h=o+d*t,m))<.01||++s>99)R n=!V(S(h+V(.01,0
),s)-c,S(h+V(0,.01),s)-c,S(h+V(0,0,.01),s)-c),m;R 0;}V T(V o,V d){V h,n,r,t=
1,l(!V(.6,.6,1));for(I b=3;b--;){I m=M(o,d,h,n);if(!m)break;if(m==1){d=d+n*(
n%d*-2);o=h+d*.1;t=t*.2;}if(m==2){F i=n%l,p=6.283185*U(),c=U(),s=sqrtf(1-c),
g=n.z<0?-1:1,u=-1/(g+n.z),v=n.x*n.y*u;d=V(v,g+n.y*n.y*u,-n.y)*(cosf(p)*s)+V(
1+g*n.x*n.x*u,g*v,-g*n.x)*(sinf(p)*s)+n*sqrtf(c);o=h+d*.1;t=t*.2;if(i>0&&M(h
+n*.1,l,h,n)==3)r=r+t*V(500,400,100)*i;}if(m==3){r=r+t*V(50,80,100);break;}}
R r;}I main(){I w=960,h=540,s=16;V e(-22,5,25),g=!(V(-3,4,0)+e*-1),l=!V(g.z,
0,-g.x)*(1./w),u(g.y*l.z-g.z*l.y,g.z*l.x-g.x*l.z,g.x*l.y-g.y*l.x);printf("P\
6 %d %d 255 ",w,h);for(I y=h;y--;)for(I x=w;x--;){V c;for(I p=s;p--;)c=c+T(e
,!(g+l*(x-w/2+U())+u*(y-h/2+U())));c=c*(1./s)+14./241;V o=c+1;c=V(c.x/o.x,c.
y/o.y,c.z/o.z)*255;printf("%c%c%c",(I)c.x,(I)c.y,(I)c.z);}}// Andrew Kensler
```



[Andrew Kensler]

[Decyphering the postcard sized path tracer, Fabien Sanglard]

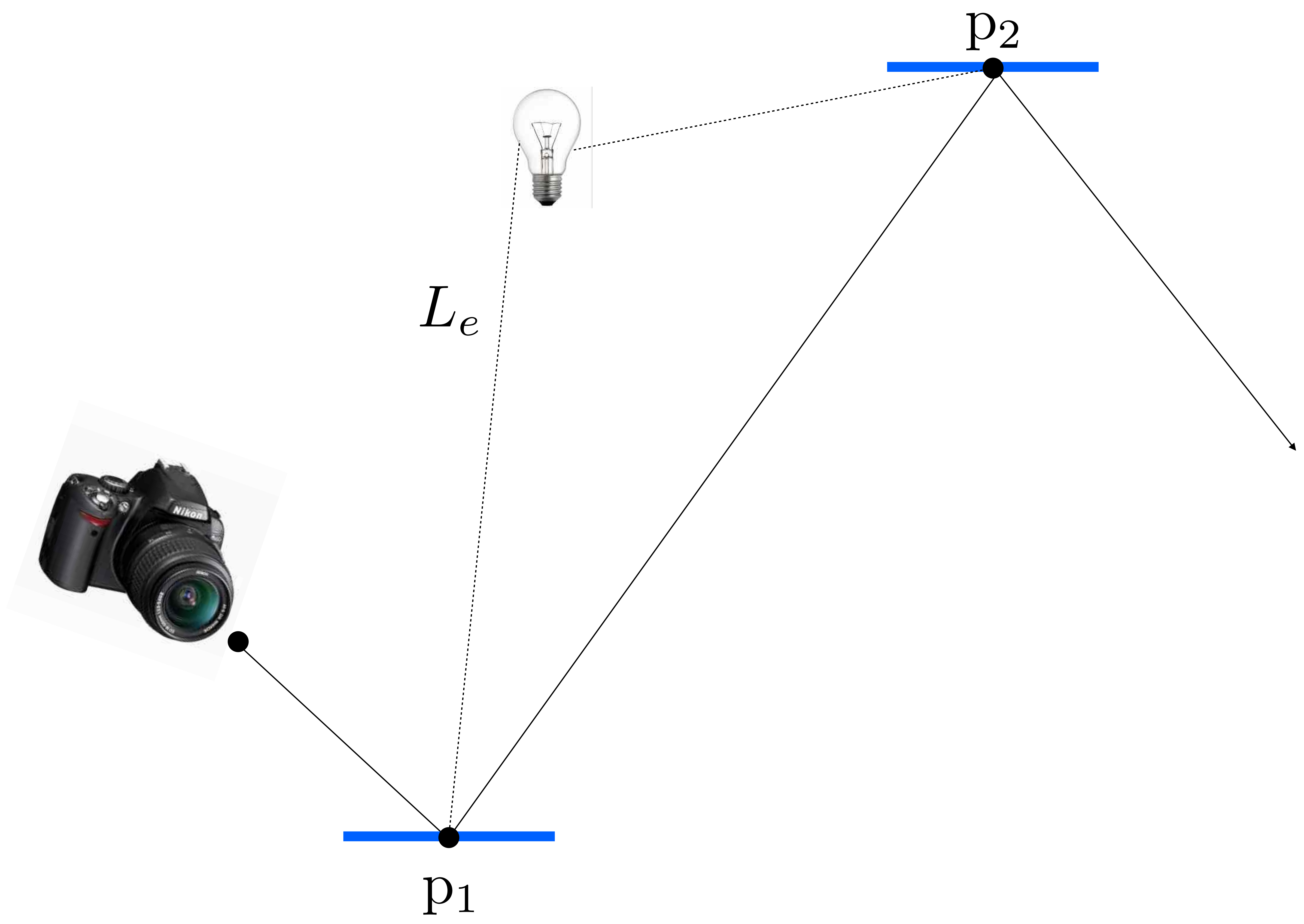
Partitioning the rendering equation

$$L_i(\mathrm{p}, \omega_i) = L_{i,d}(\mathrm{p}, \omega_i) + L_{i,i}(\mathrm{p}, \omega_i)$$

- **Incident direct illumination:** $L_{i,d}(\mathrm{p}, \omega_i)$
  - **Sample lights+BRDFs**
- **Incident indirect illumination:** $L_{i,i}(\mathrm{p}, \omega_i)$
  - **Recursive evaluation of rendering eqn.**

$$\begin{aligned} L_o(\mathrm{p}, \omega_o) =& L_e(\mathrm{p}, \omega_o) + \\ & \int_{H^2} f_r(\omega_i \rightarrow \omega_o)\, L_{i,d}(\mathrm{p}, \omega_i)\ \cos\theta_i \,\mathrm{d}\omega_i + \\ & \int_{H^2} f_r(\omega_i \rightarrow \omega_o)\, L_{i,i}(\mathrm{p}, \omega_i)\ \cos\theta_i \,\mathrm{d}\omega_i \end{aligned}$$

Path tracing



Path tracing: sampling indirect illumination

$$\int_{H^2} f_r(\omega_i \to \omega_o)\, L_{i,i}(\mathrm{p}, \omega_i)\, \cos\theta_i \, \mathrm{d}\omega_i$$

- **Sample incoming direction from some distribution (e.g. proportional to BRDF):** $\omega_i \sim p(\omega)$
- **Recursively call path tracing function to compute incident indirect radiance**
- **Estimator:**

$$\frac{f_r(\omega_i \to \omega_o)\, L_{i,i}(\mathrm{p}, \omega_i)\, \cos\theta_i}{p(\omega_i)}$$

$$\frac{f_r(\omega_i \to \omega_o)\, L_o(tr(\mathrm{p}, \omega_i), -\omega_i)\, \cos\theta_i}{p(\omega_i)}$$

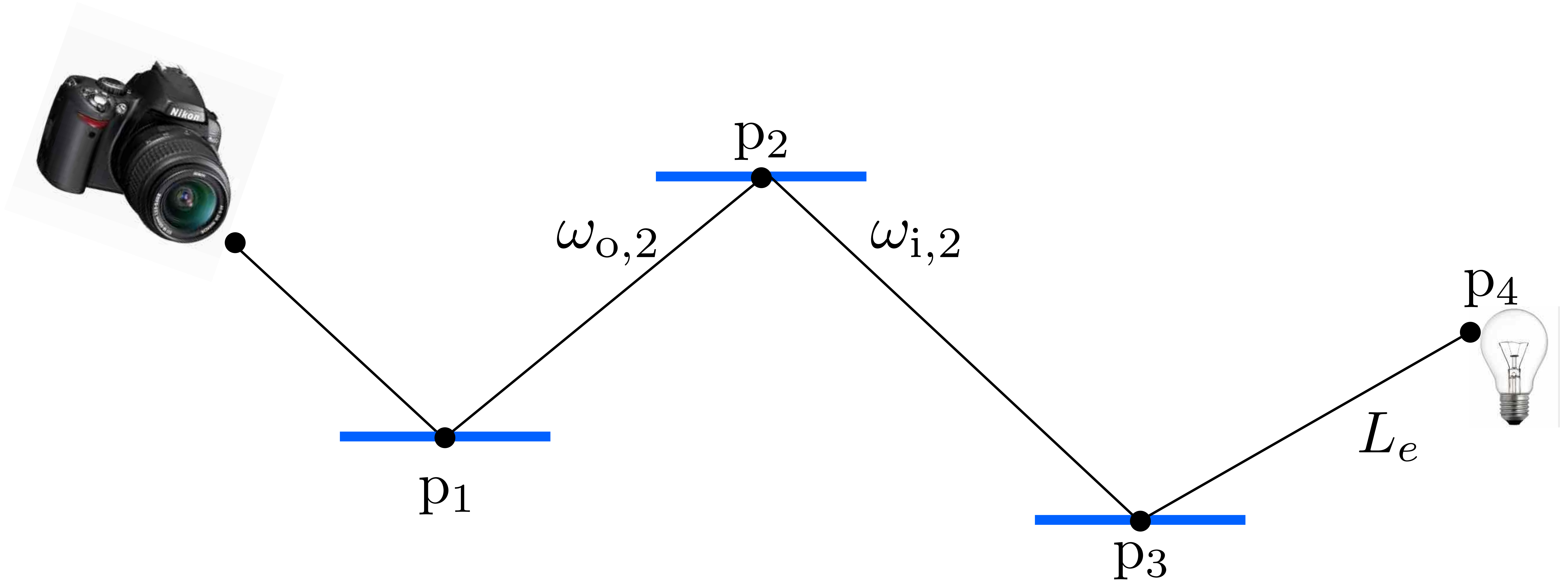
Path tracing: recursive

$$L_o(\mathrm{p},\omega_o) = L_e(\mathrm{p},\omega_o) + \int_{H^2} f_r(\omega_i \to \omega_o)\, L_{i,d}(\mathrm{p},\omega_i)\, \cos\theta_i \, \mathrm{d}\omega_i + \int_{H^2} f_r(\omega_i \to \omega_o)\, L_o(tr(\mathrm{p},\omega_i), -\omega_i)\, \cos\theta_i \, \mathrm{d}\omega_i$$

```
Spectrum PathLo(Ray ray) {
  Intersection isect = scene->Intersect(ray);
  BSDF bsdf = isect.GetBSDF();
  Vector3f wo = -ray.d;

  Spectrum Ld = ReflFromDirectLighting(bsdf, wo);

  Spectrum fr = bsdf.Sample_f(wo, &wi, &pdf);
  return (depth == 0 ? isect.Le(wo)) : 0.) + Ld +
    fr * PathLo(Ray(isect.P, wi)) * Dot(wi, isect.N) / pdf;
}
```

Reflectance due to incident direct lighting

Path contribution



$$\beta(\bar{\mathrm{p}}) = \prod_j \frac{f_r(\mathrm{p}_j, \omega_{\mathrm{o},j}, \omega_{\mathrm{i},j}) \cos\theta_{\mathrm{i},j}}{p(\omega_{\mathrm{i},j})}$$

$$L_i = \sum \beta(\bar{\mathrm{p}}) L_e$$

Path tracing: iterative version

```
Spectrum PathLo(Ray ray) {
  Spectrum Lo = 0, beta = 1;
  int depth = 0;
  while (true) {
    Intersection isect = scene->Intersect(ray);
    Vector3f wo = -ray.d;
    if (depth == 0) Lo += isect.Le(wo);
    BSDF brdf = isect.GetBSDF();

    Lo += beta * ReflFromDirectLighting(bsdf, wo);

    Spectrum fr = bsdf.Sample_f(wo, &wi, &pdf);
    beta *= fr * Dot(wi, isect.N) / pdf;
    depth++;
    ray = Ray(isect.P, wi);
  }
  return Lo;
}
```

What's the problem with this implementation?

Russian Roulette

- **Define path termination probability** q
- **Randomly terminate path based on** q**;**
  - **for surviving paths, scale contribution by** $\frac{1}{1-q}$
- **Russian roulette gives expectation:**

$$(1-q)E\left[\frac{X}{1-q}\right] + qE[0] = E[X]$$

Russian Roulette

```
Spectrum PathLo(Ray ray) {
  Spectrum Lo = 0, beta = 1;
  int depth = 0;
  while (true) {
    Intersection isect = scene->Intersect(ray);
    Vector3f wo = -ray.d;
    if (depth == 0) Lo += isect.Le(wo);

    BSDF bsdf = isect.GetBSDF();
    Lo += beta * ReflFromDirectLighting(bsdf, wo);

    Spectrum fr = bsdf.Sample_f(wo, &wi, &pdf);
    beta *= fr * Dot(wi, isect.N) / pdf;

    float q = 0.25;
    if (randomFloat() < q) break;
    else beta /= (1-q);
    depth++;
    ray = Ray(isect.P, wi);
  }
  return Lo;
}
```

Recall the estimator:

$$\frac{f_r(\omega_i \to \omega_o)\, L_o(tr(\mathrm{p}, \omega_i), -\omega_i)\, \cos\theta_i}{p(\omega_i)}$$

P(choosing ω_i | not_terminating) P(not terminating)

Improving Russian Roulette

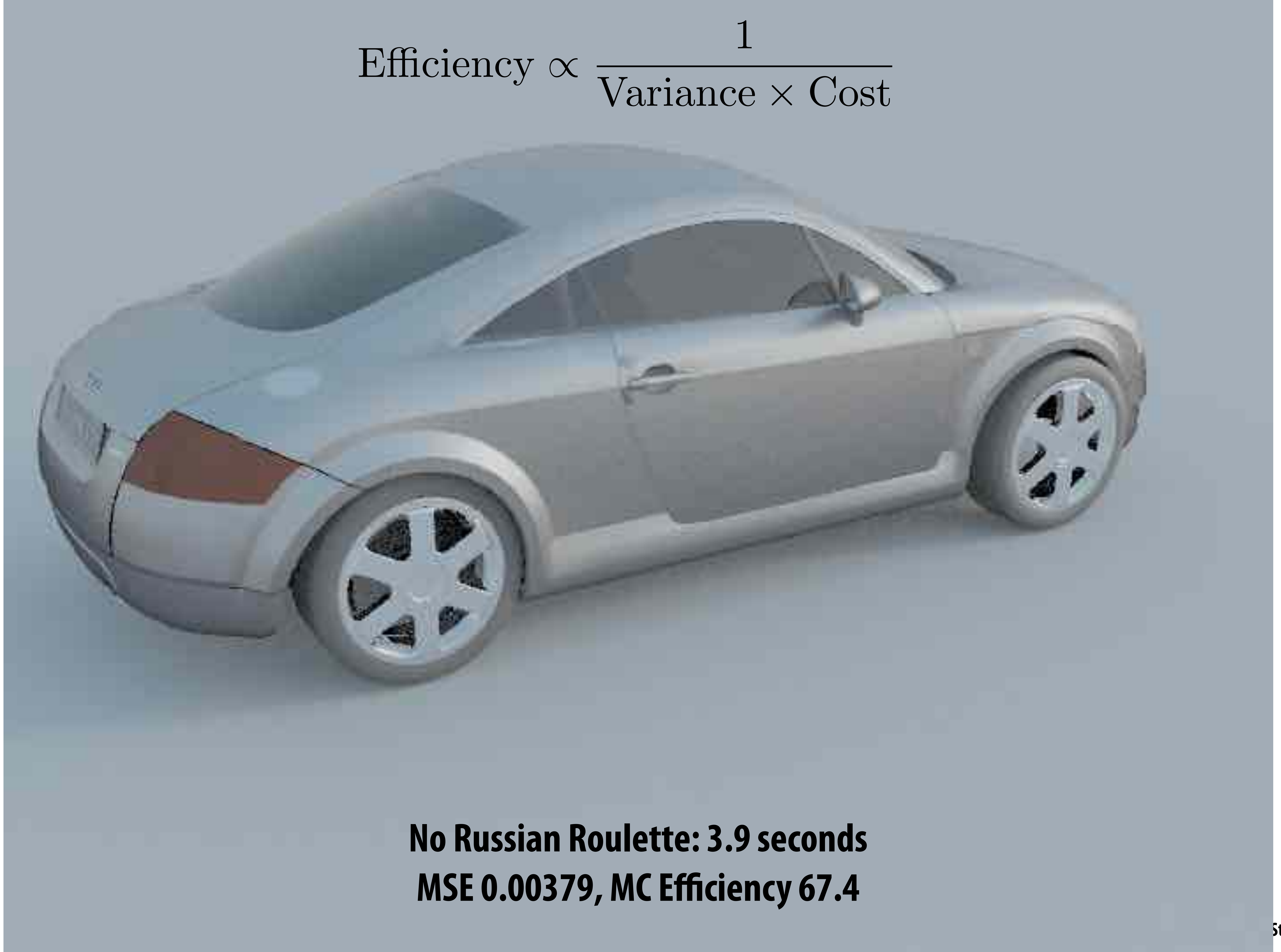
- Recurring principle: It’s best to avoid spending computation on samples that make a small contribution
- How do you know a sample will make a small contribution before you evaluate it?
- Recall: $L_i = \beta(\bar{\mathrm{p}}) L_e$
- $\beta(\bar{\mathrm{p}})$ is a reasonable proxy for expected contribution

Russian Roulette: better

```
Spectrum PathLo(Ray ray) {
  Spectrum Lo = 0, beta = 1;
  int depth=0;
  while (true) {
    Intersection isect = scene->Intersect(ray);
    Vector3f wo = -ray.d;
    if (depth == 0) Lo += isect.Le(wo);

    BSDF bsdf = isect.GetBSDF();
    Lo += beta * ReflFromDirectLighting(bsdf, wo);

    Spectrum fr = bsdf.Sample_f(wo, &wi, &pdf);
    beta *= fr * Dot(wi, isect.N) / pdf;

    Float q = 1 - beta.MaxComponentValue();
    if (UniformFloat() < q) break;
    beta /= 1 - q;
    depth++;
    ray = Ray(isect.P, wi);
  }
  return Lo;
}
```

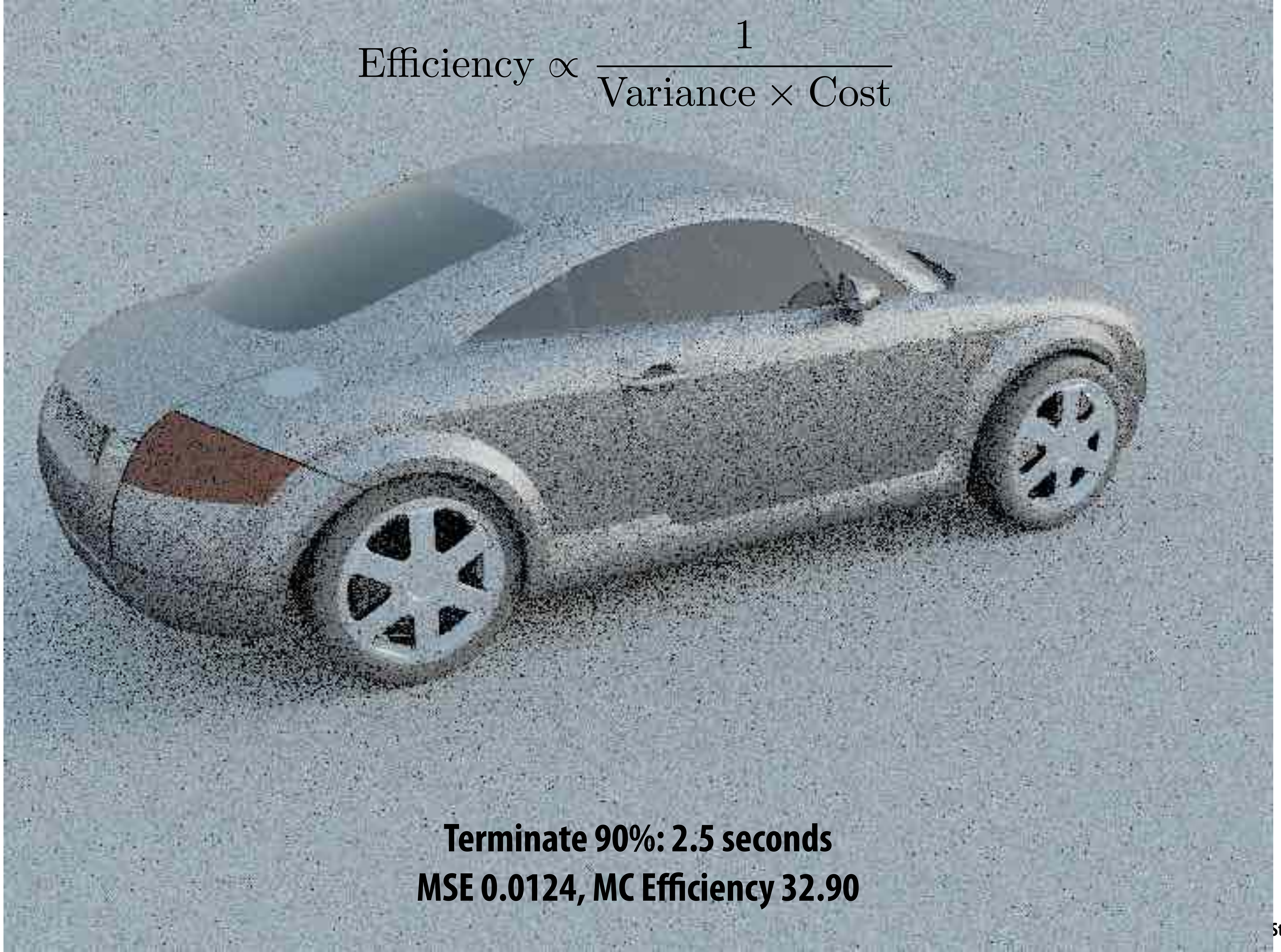
High path throughput yields low termination probability

$$\text{Efficiency} \propto \frac{1}{\text{Variance} \times \text{Cost}}$$



No Russian Roulette: 3.9 seconds
MSE 0.00379, MC Efficiency 67.4

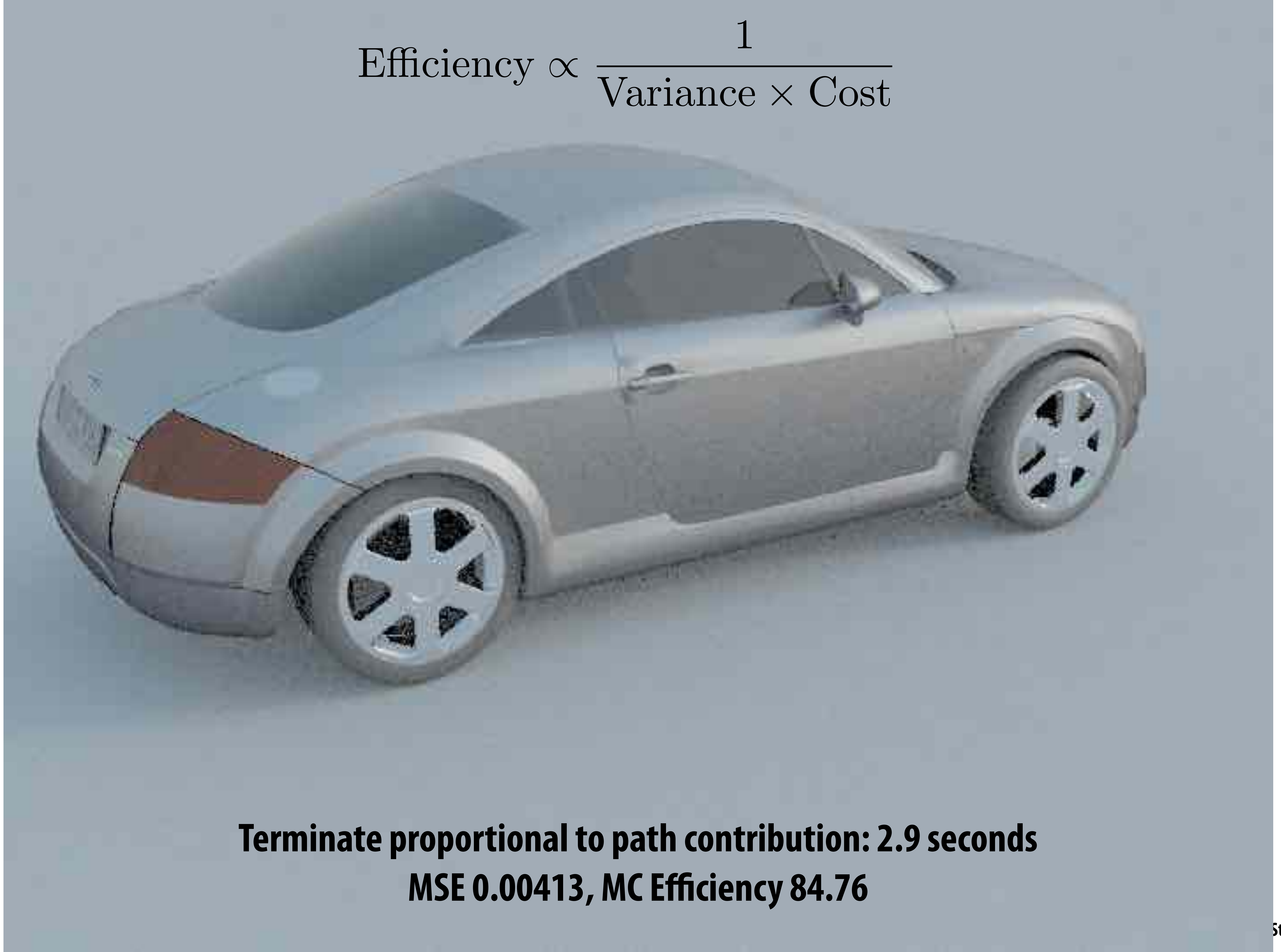
$$\text{Efficiency} \propto \frac{1}{\text{Variance} \times \text{Cost}}$$



Terminate 90%: 2.5 seconds

MSE 0.0124, MC Efficiency 32.90

$$\text{Efficiency} \propto \frac{1}{\text{Variance} \times \text{Cost}}$$



Terminate proportional to path contribution: 2.9 seconds
MSE 0.00413, MC Efficiency 84.76

Advanced topic: path guiding

- **Recall the MC estimator:**

$$\frac{f_r(\omega_i \to \omega_o)\, L_o(tr(\mathrm{p}, \omega_i), -\omega_i)\ \cos\theta_i}{p(\omega_i)}$$

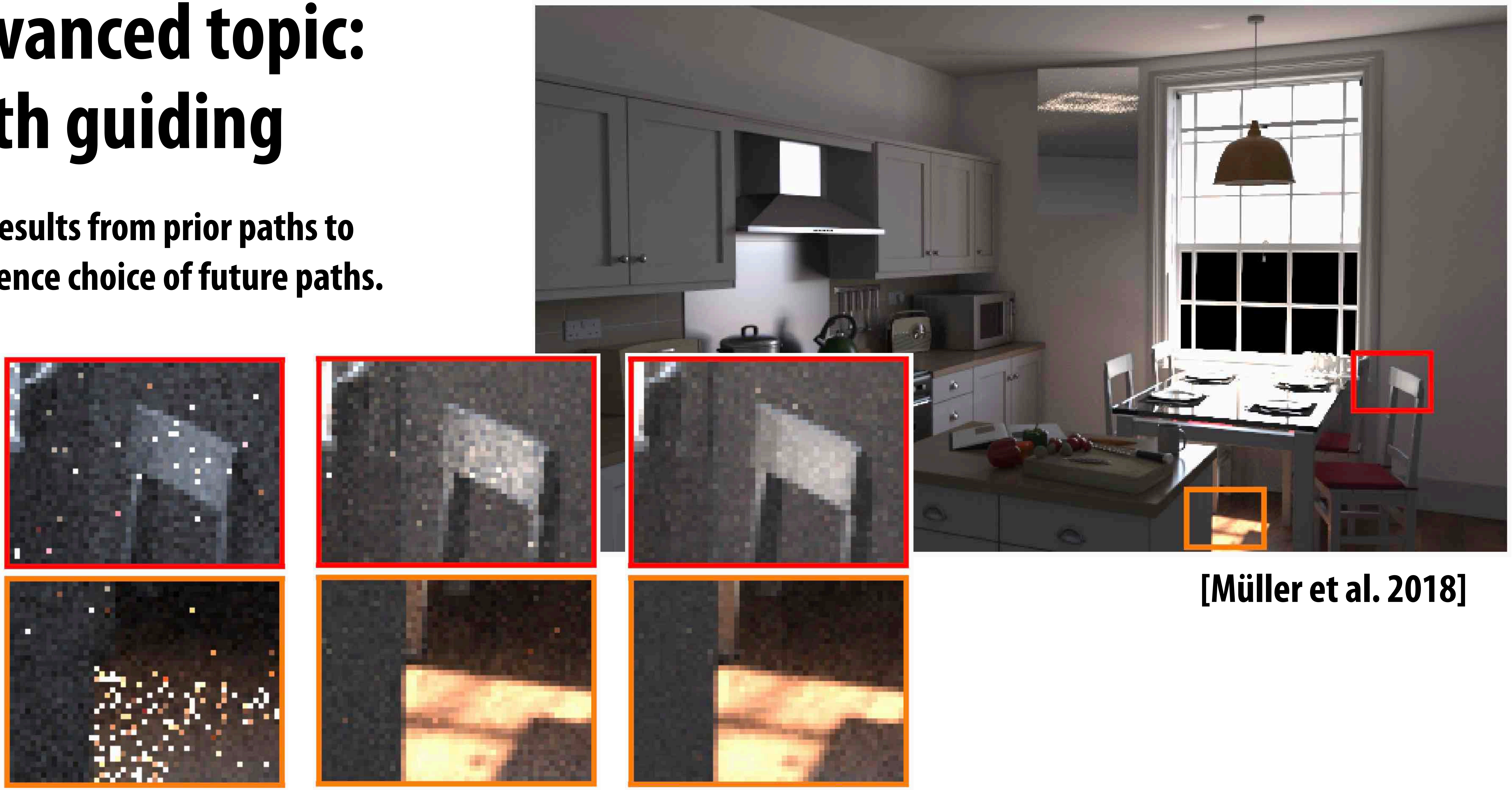
- **Regular path tracing: sample (or something similar to it)** $w_i \sim f_r(\omega_i \to \omega_o)$

- **But: really want to sample** $\propto f_r\, L_o\, \cos\theta$

- **Idea: learn the distribution of light in the scene to guide sampling**

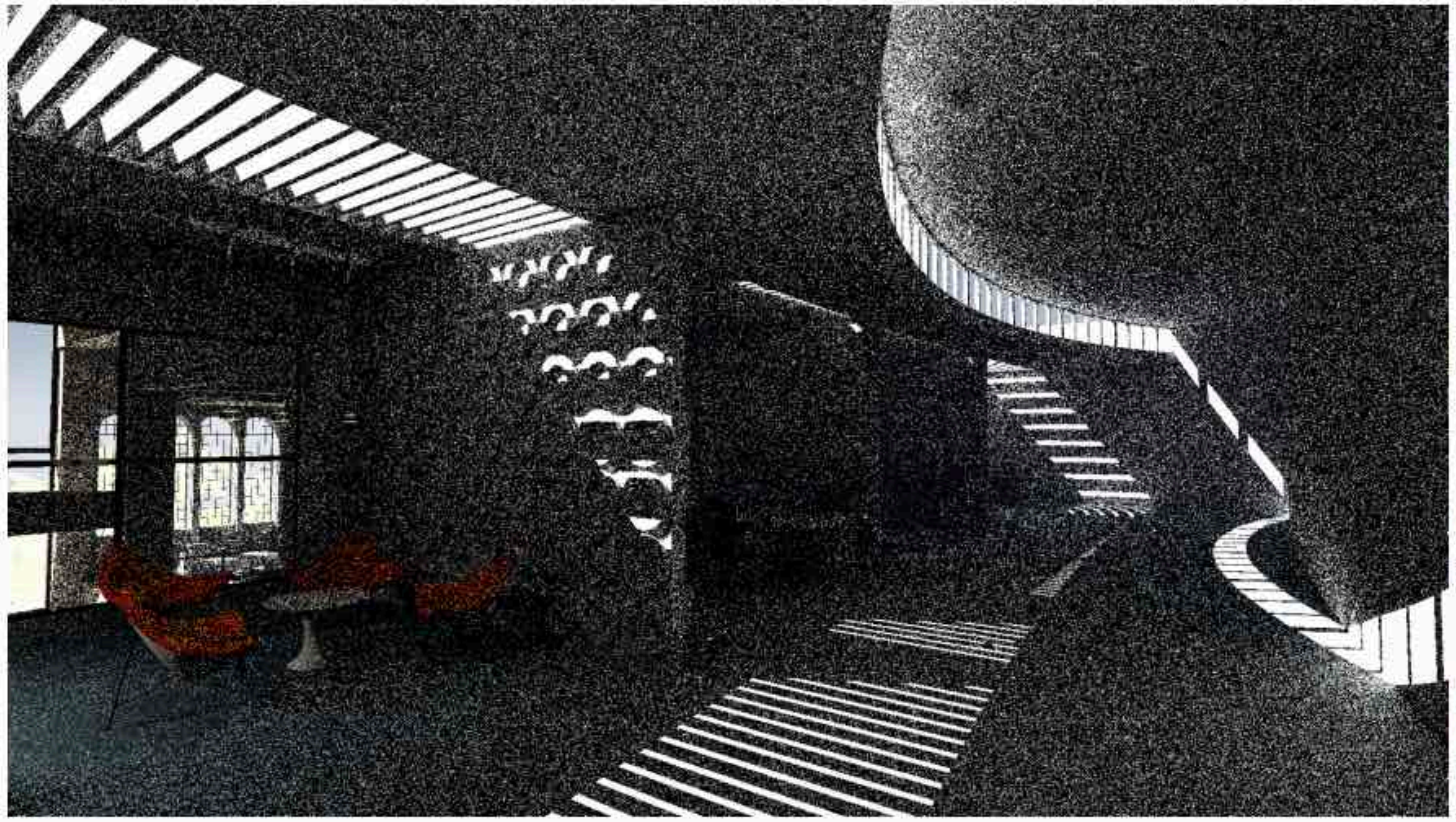
Advanced topic: path guiding

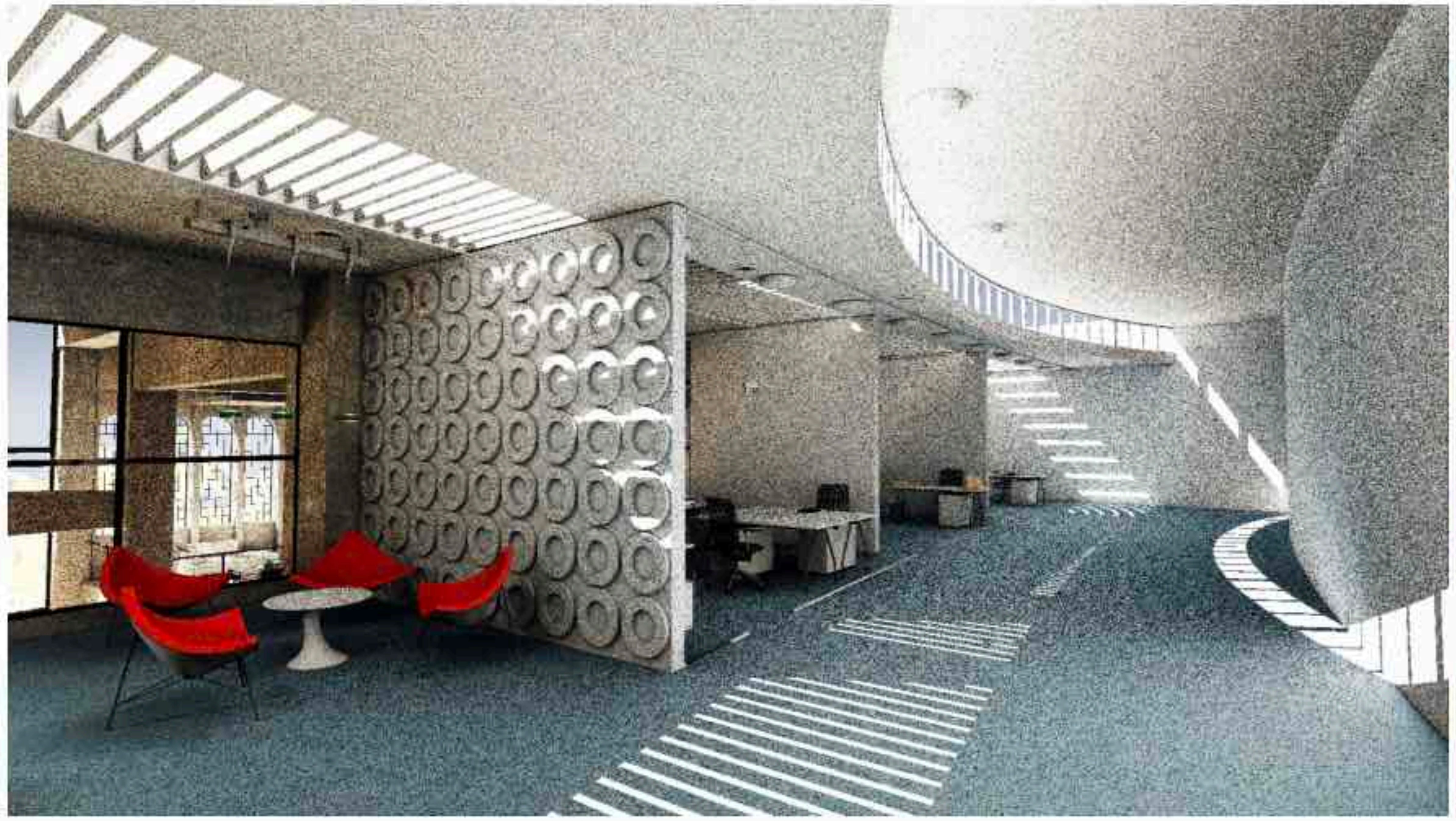
Use results from prior paths to influence choice of future paths.



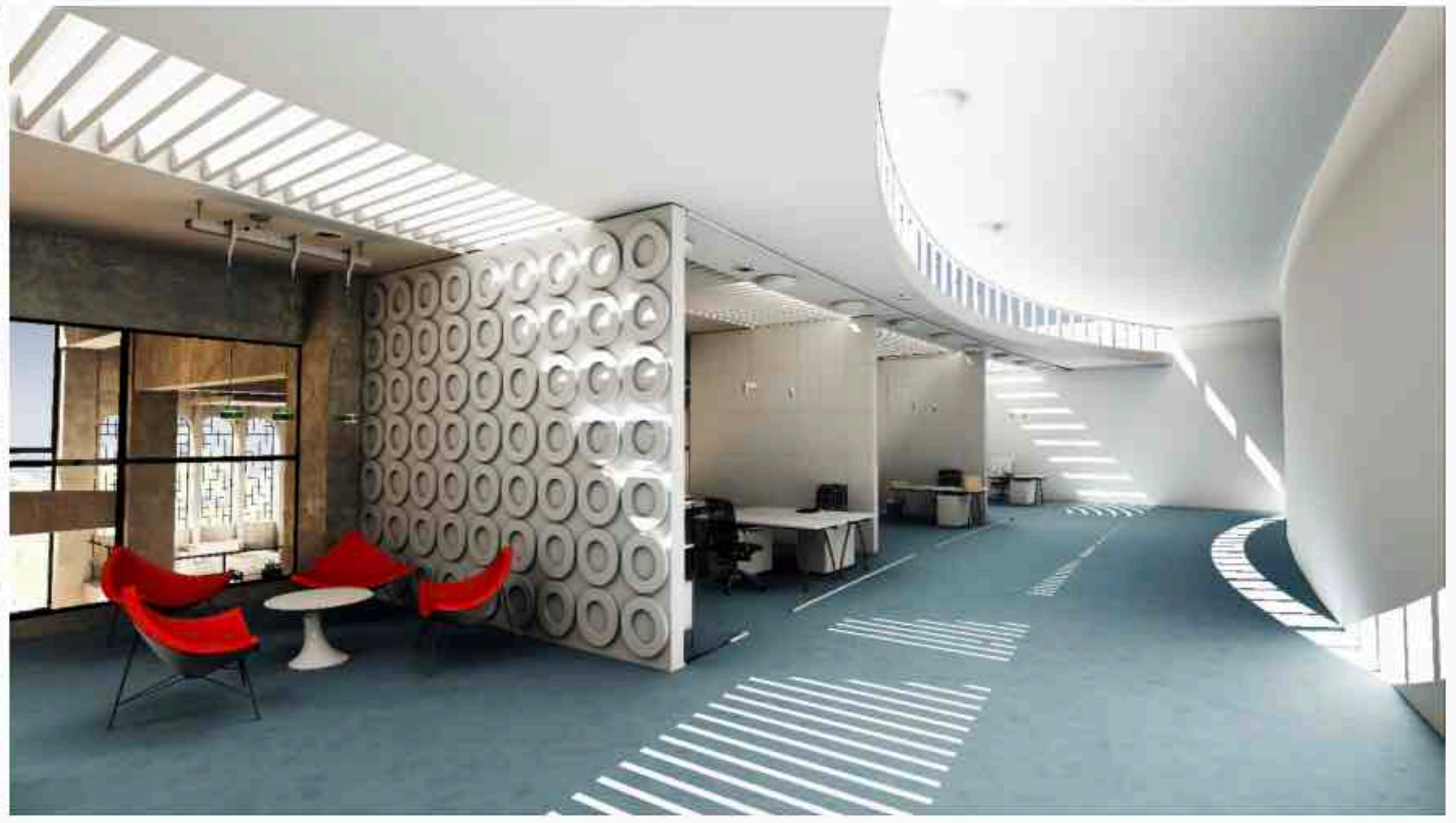
[Müller et al. 2018]

Recent work: caching/reusing good paths


Path traced: 1 path/pixel (8 ms/frame)


Path traced: 1 path/pixel using ReSTIR GI (8.9 ms/frame)

Key idea: cache good paths, reuse good paths found from from prior frames or for prior pixels in same frame


High sample count path traced "ground truth"

[Ouyang et al. 2021]

Path tracing summary

- Path tracing: sample paths through scene (from the domain of all possible paths)
  - Unbiased estimator of solution to the full rendering equation (provided paths are correctly weighted by their probabilities… (incorrect PDFs are a common source of bugs)

- Simple, elegant, brute force! ;-)

- Efficiency comes from intelligent biasing of paths toward the most "important" paths.
  - But that's a topic for CS348B