Global Illumination

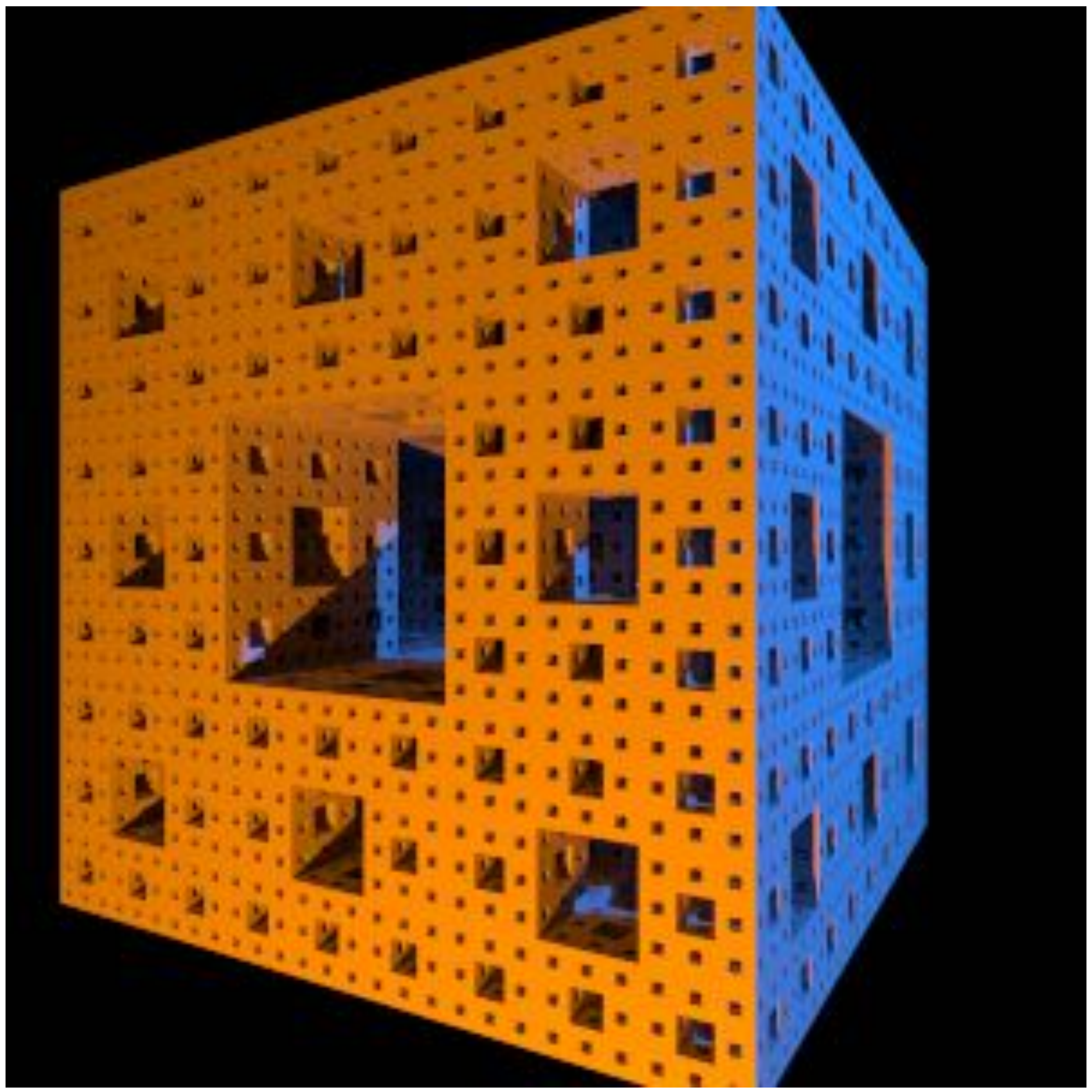
Today

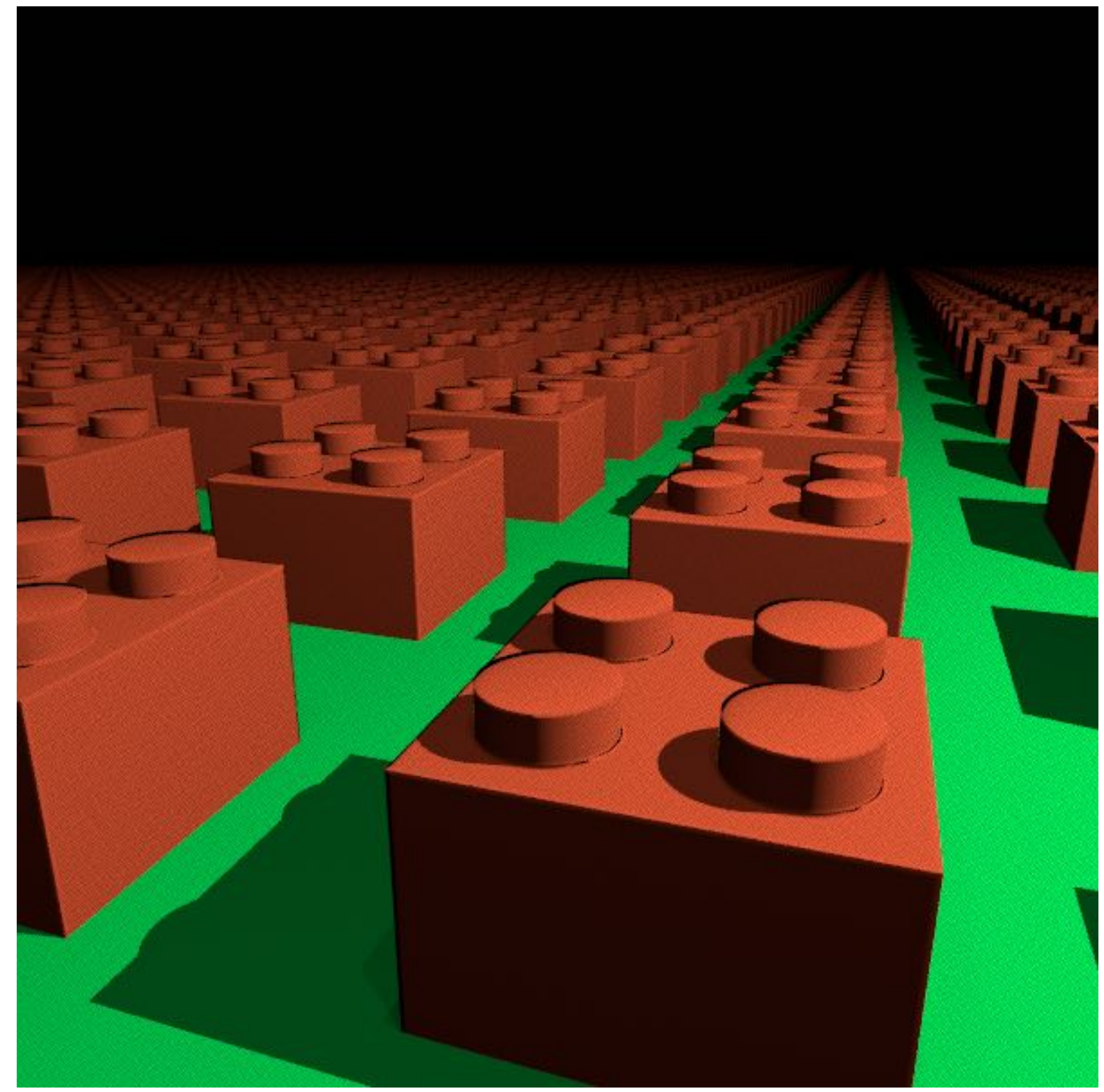
- Direct vs. indirect illumination
- Energy balance and the rendering equation
- Path tracing
- Russian roulette
- Path guiding

Next

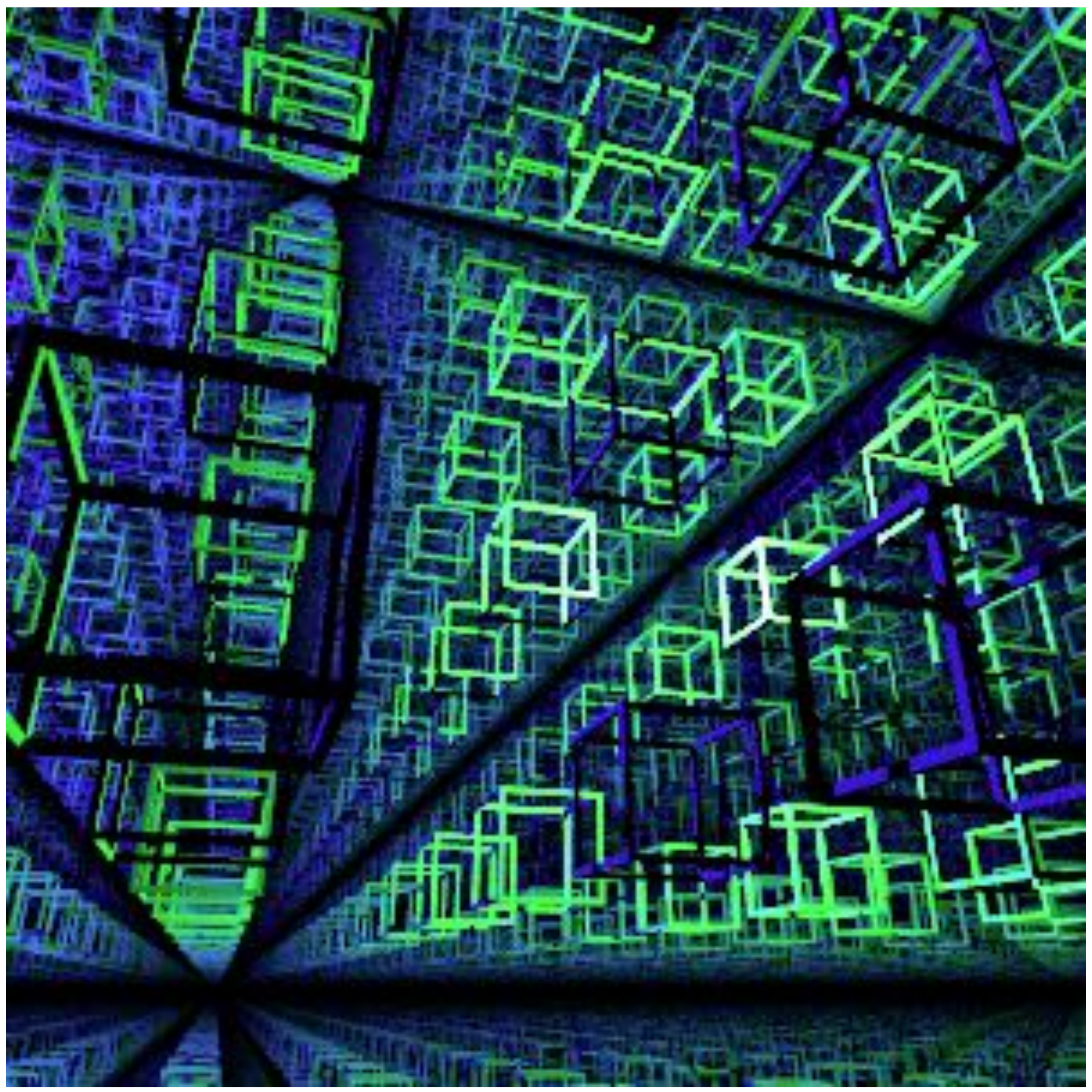
- Bidirectional light transport
- Volume scattering

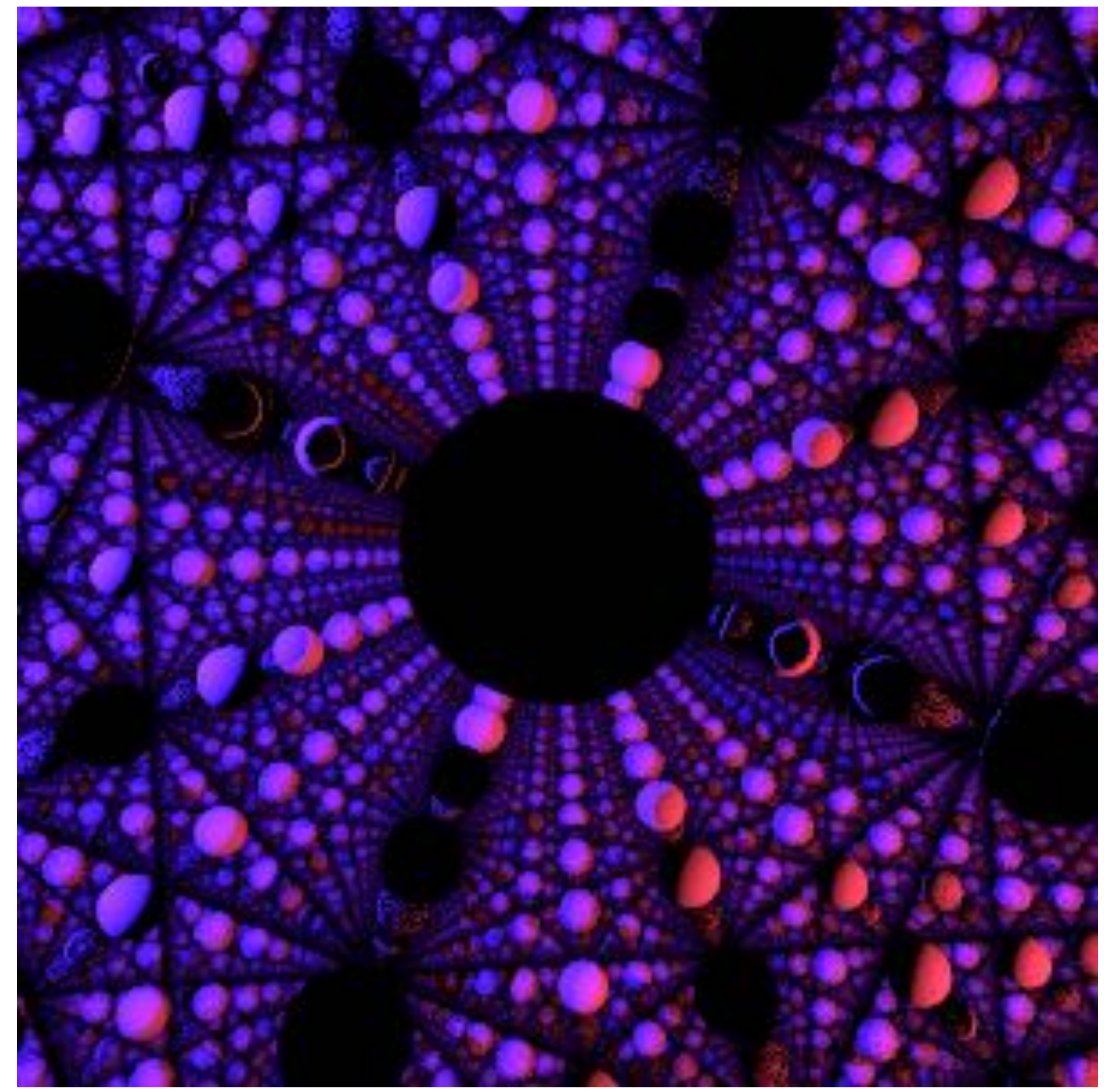
But first... cool pics from HW2



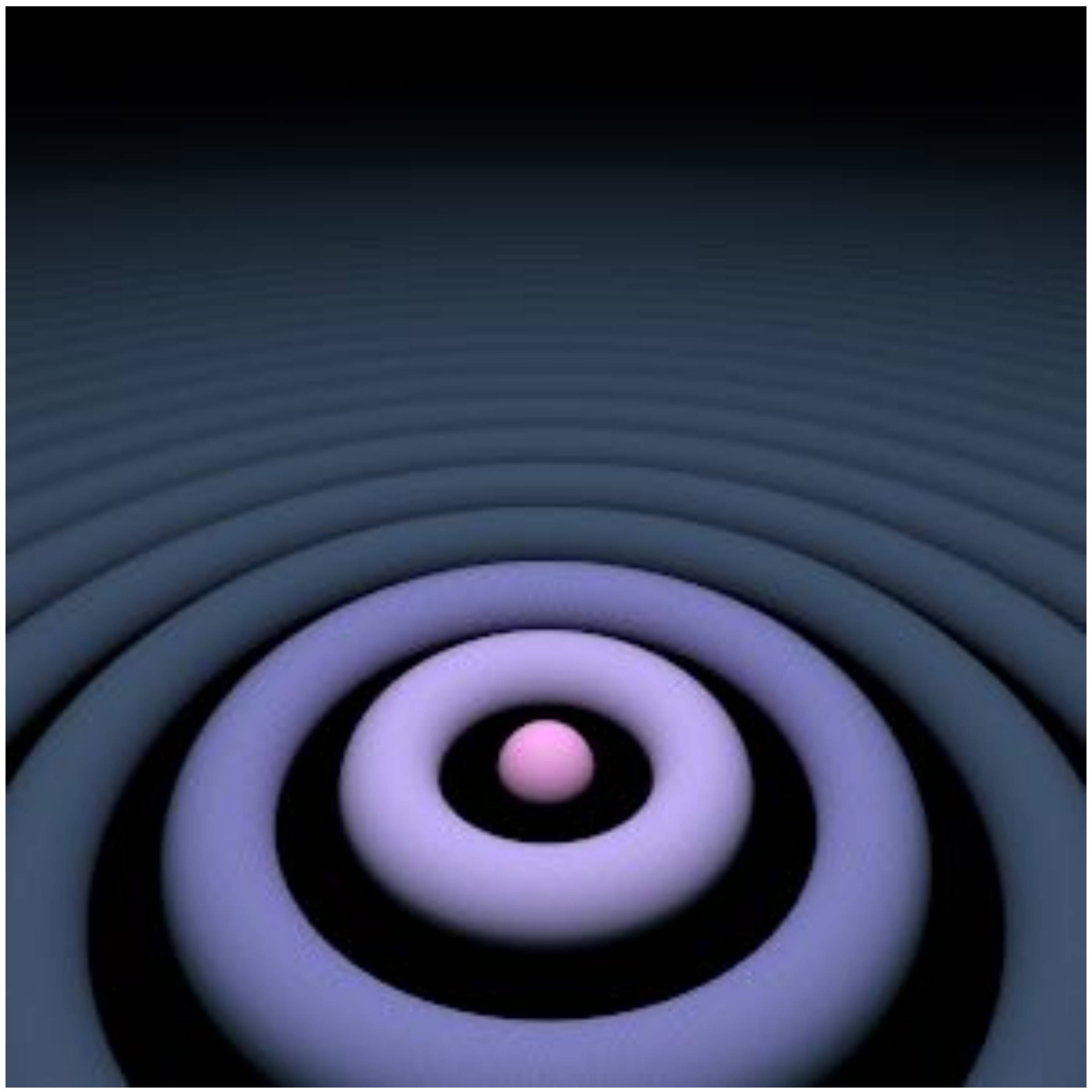


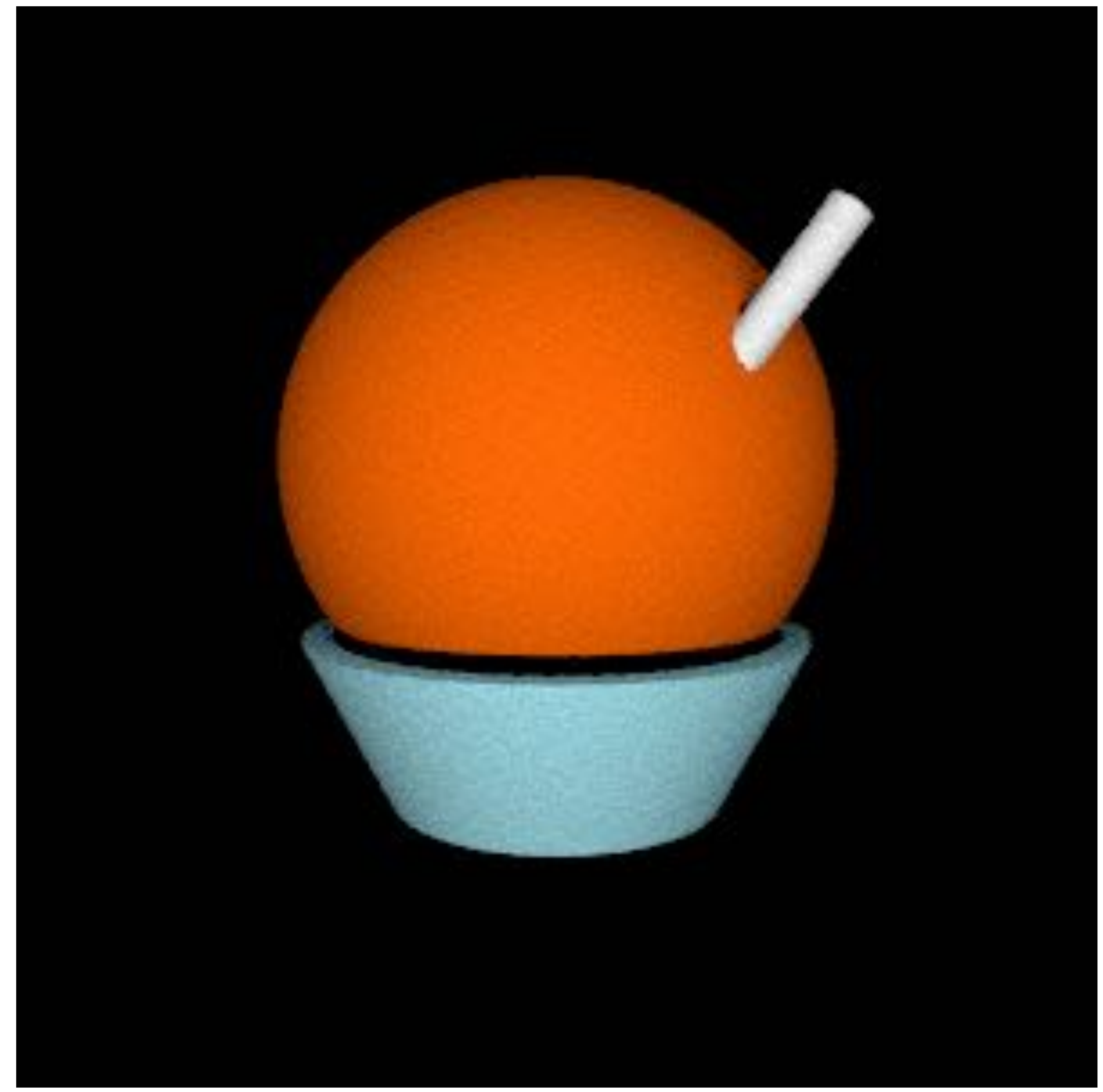
But first... cool pics from HW2



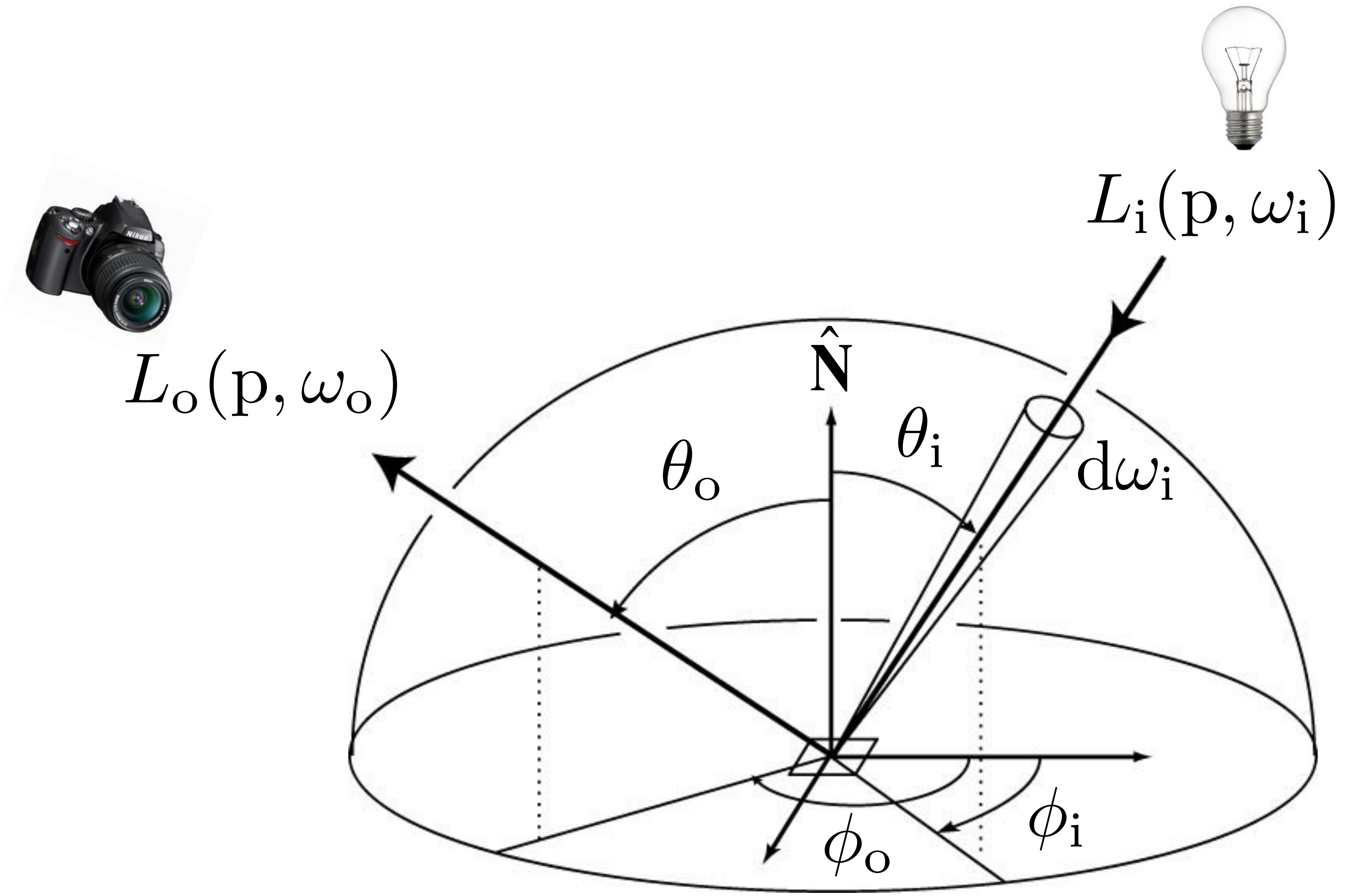


But first... cool pics from HW2





Last Time: the Reflection Equation



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

Monte Carlo Estimate for Reflection Eqn.

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

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

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

1 Bounce (Direct Illumination)

Direct illumination + reflection + transparency

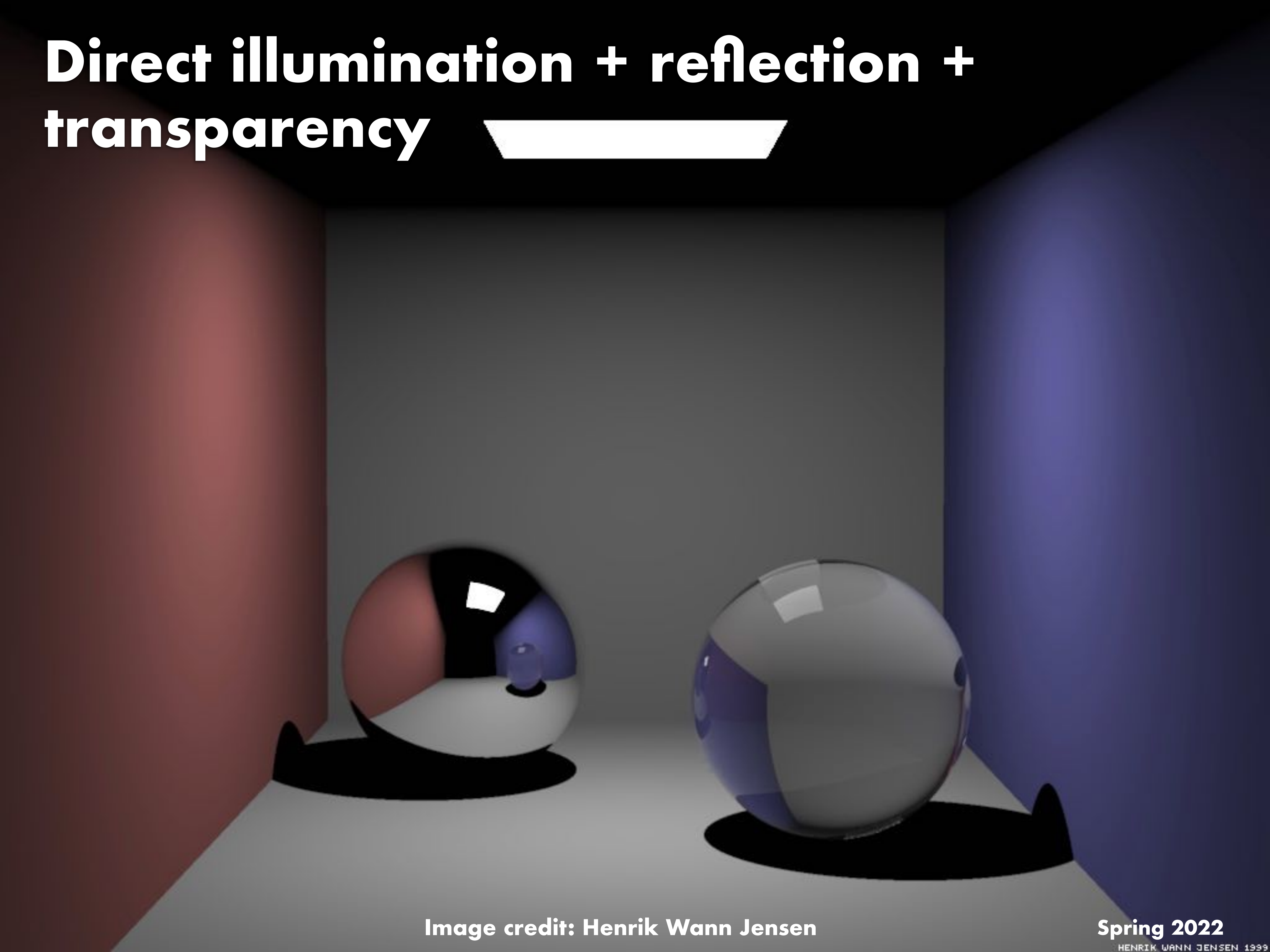


Image credit: Henrik Wann Jensen

With indirect illumination

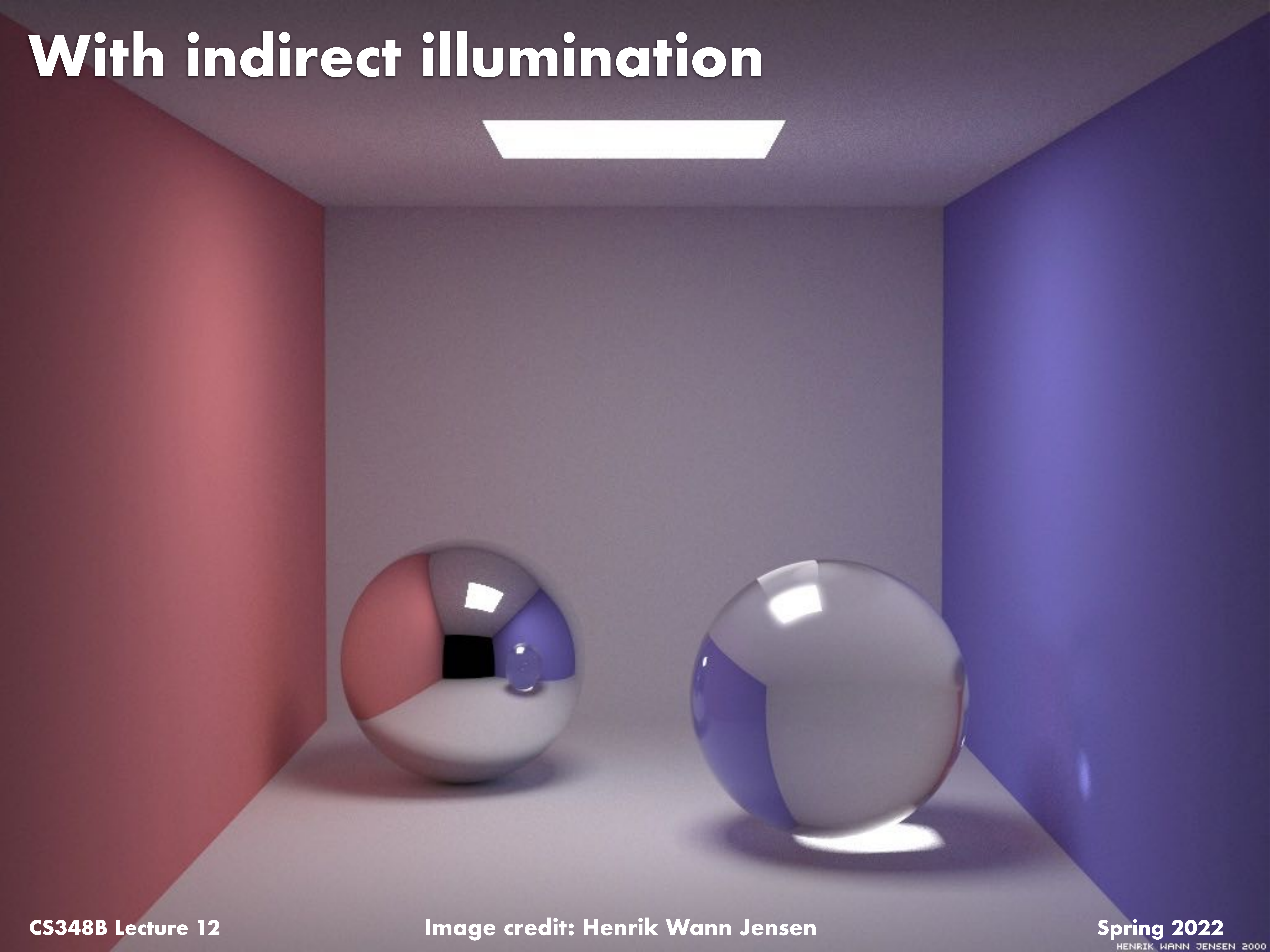


Image credit: Henrik Wann Jensen

Direct illumination only

Sixteen-bounce global illumination

Importance of indirect illumination

Importance of indirect illumination

Scene breakdown: by bounce

0 Bounces (visible lights)

1 Bounce (Direct Illumination)

2 Bounces

3 Bounces

4 Bounces

8 Bounces

Direct Illumination only

Indirect Illumination only

Direct + Indirect

Direct vs. Global Illumination

(a) Scene

(b) Direct Component

(c) Global Component

Fast Separation of Direct and Global Components of a Scene Using High Frequency Illumination, Nayar et al. 2006

Energy Balance

Accountability

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

Macro level:

- The total light energy put into the system must equal the energy leaving (usually, via heat)

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

Energy Balance

Accountability

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

Micro level:

- The energy flowing into a small region of phase 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]

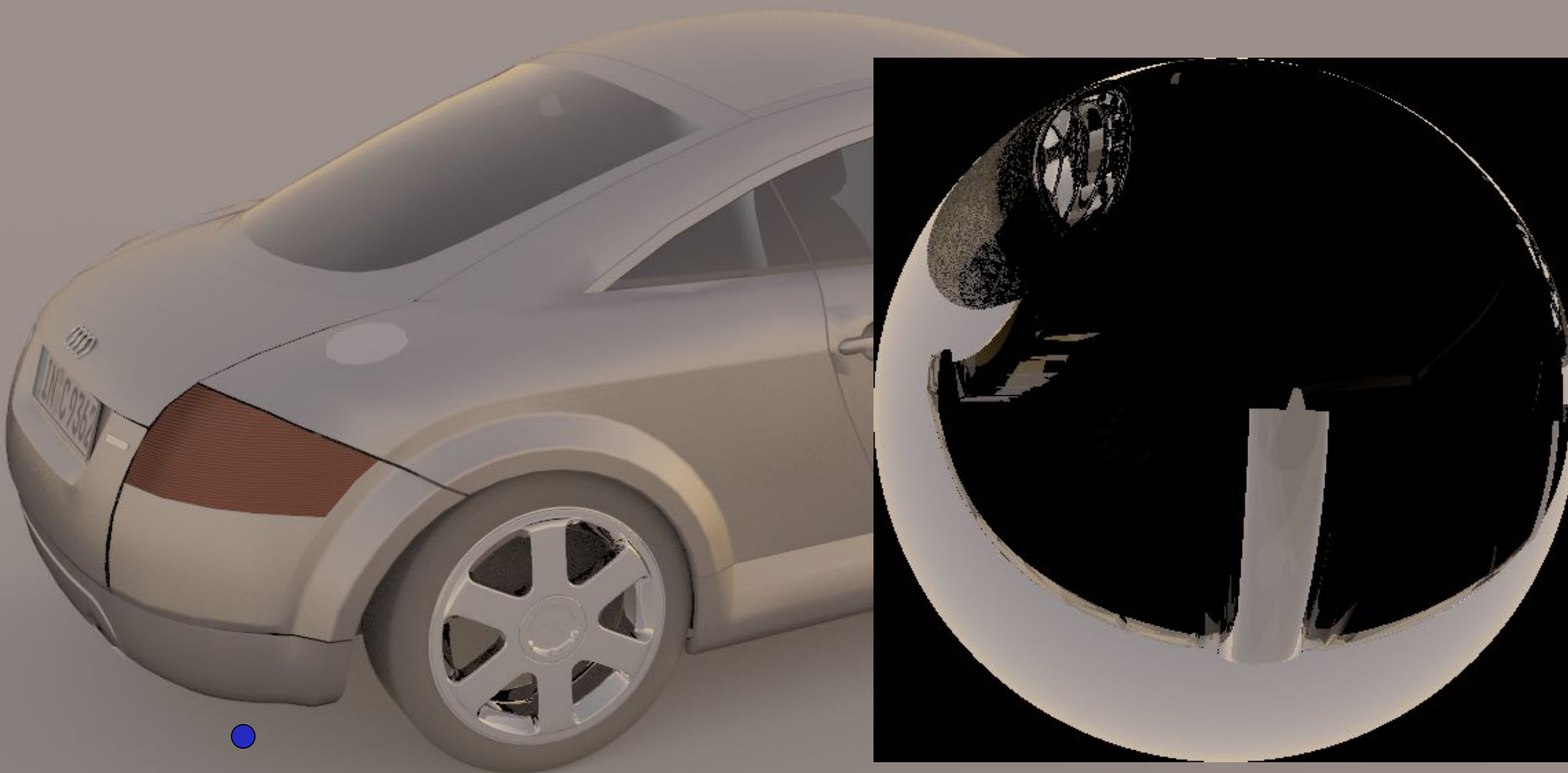
[reflected] = [incoming] - [absorbed]

[outgoing] = [emitted] + [reflected]

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

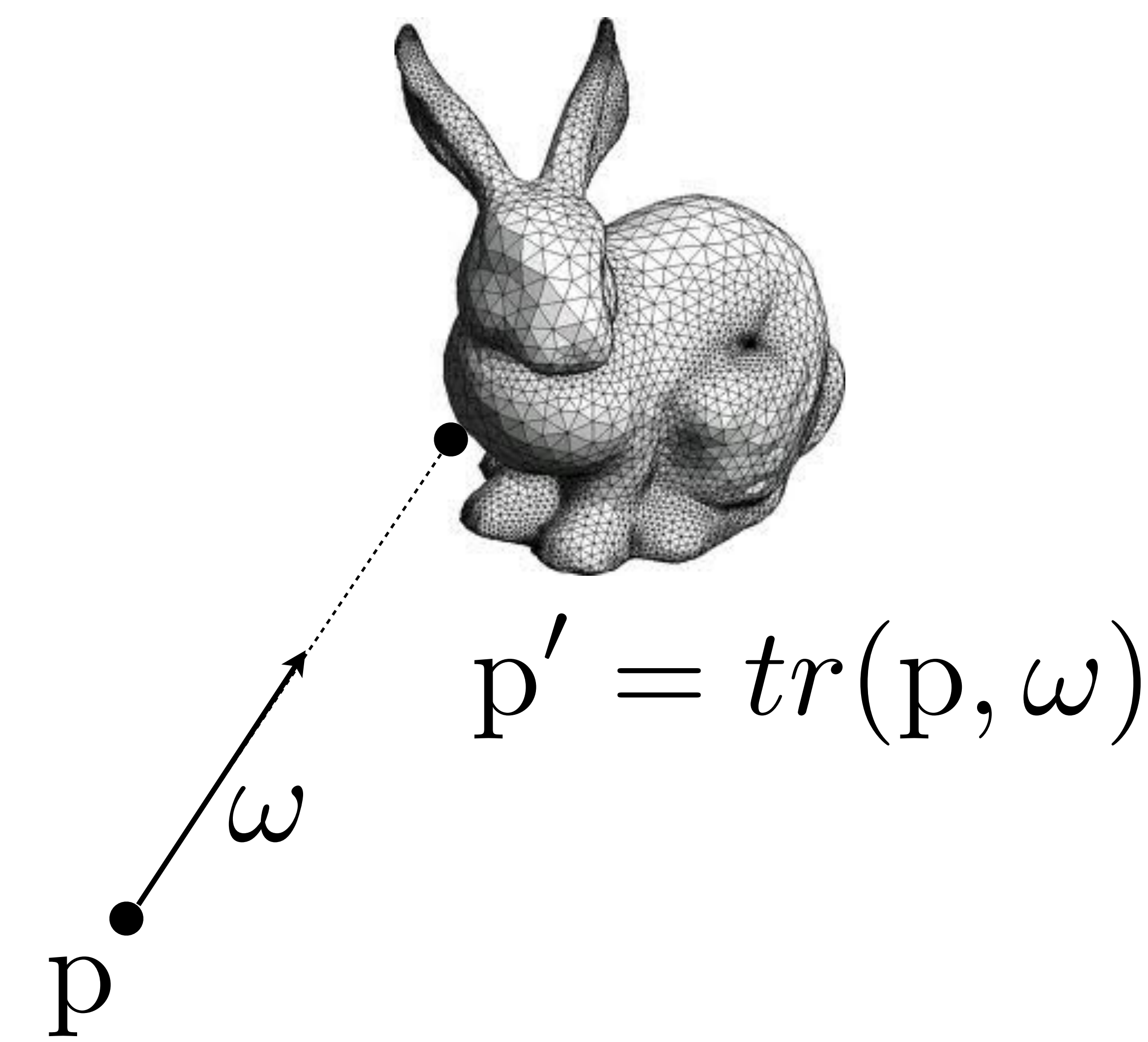
Need to know incident radiance.

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:

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

Light scattering

Light transport

The Rendering Equation

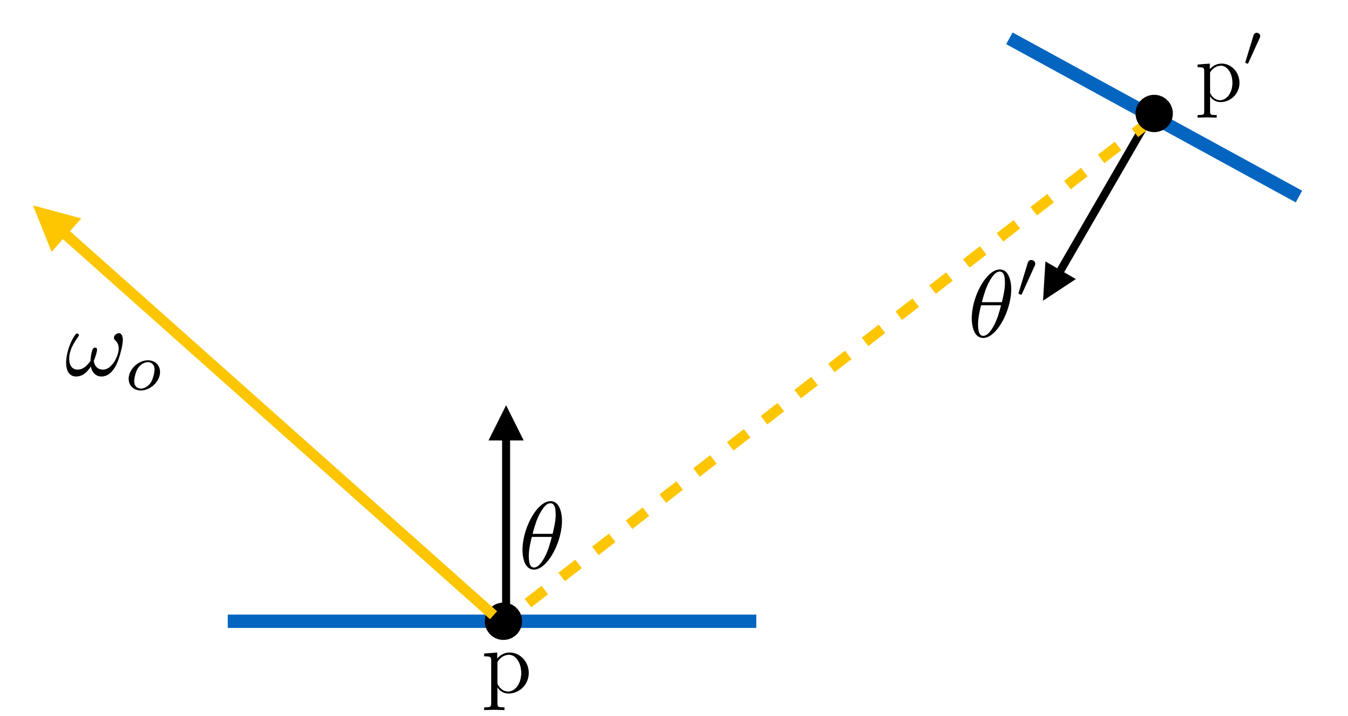
$$L(\mathrm{p}, \omega_o) = L_e(\mathrm{p}, \omega_o) + \int_{H^2} f_r(\mathrm{p}, \omega_i \to \omega_o)\, L(tr(\mathrm{p}, \omega_i), -\omega_i)\, \cos\theta_i \, \mathrm{d}\omega_i$$

The rendering equation: area form

Can equivalently write rendering equation as an integral over surface area of objects in the scene

- **Apply change of variables** $\mathrm{d}\omega = \dfrac{\cos\theta}{r^2}\mathrm{d}A$
- **Introduce binary visibility function:** $V(\mathrm{p} \leftrightarrow \mathrm{p}')$

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



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

The rendering equation: sum over paths

$$L_o(\mathrm{p}_1 \to \mathrm{p}) = L_e(\mathrm{p}_1 \to \mathrm{p})$$

Path of length 1

$$+ \int_A L_e(\mathrm{p}_2 \to \mathrm{p}_1) f(\mathrm{p}_2 \to \mathrm{p}_1 \to \mathrm{p}) G(\mathrm{p}_1 \leftrightarrow \mathrm{p}_2) \mathrm{d}A(\mathrm{p}_2)$$

Paths of length 2

$$+ \int_A \int_A L_e(\mathrm{p}_3 \to \mathrm{p}_2) f(\mathrm{p}_3 \to \mathrm{p}_2 \to \mathrm{p}_1) G(\mathrm{p}_2 \leftrightarrow \mathrm{p}_3) f(\mathrm{p}_2 \to \mathrm{p}_1 \to \mathrm{p}) G(\mathrm{p}_1 \leftrightarrow \mathrm{p}_2) \mathrm{d}A(\mathrm{p}_3) \mathrm{d}A(\mathrm{p}_2)$$

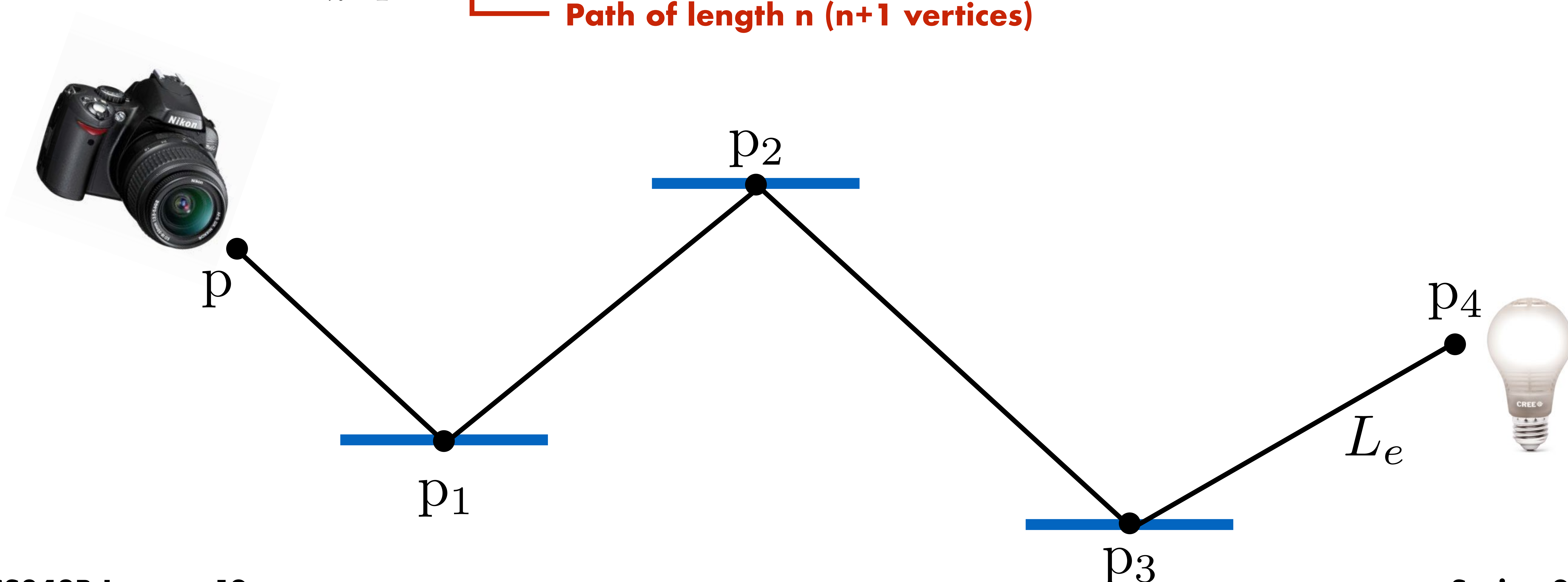
$$+ \cdots$$

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)



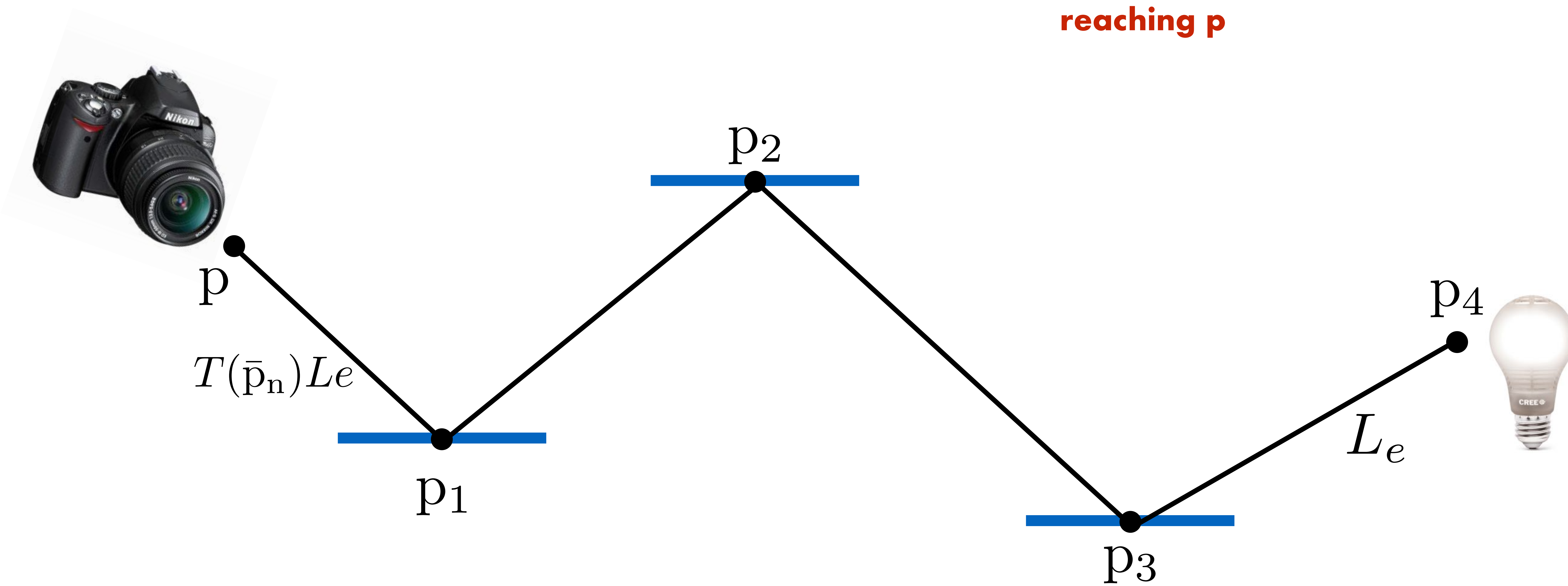
The rendering equation: sum over paths

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

$$\times \boxed{\left(\prod_{i=1}^{n-1} f(\mathrm{p}_\mathrm{i+1} \rightarrow \mathrm{p}_\mathrm{i} \rightarrow \mathrm{p}_\mathrm{i-1}) G(\mathrm{p}_\mathrm{i+1} \leftrightarrow \mathrm{p}_\mathrm{i}) \right)} \mathrm{d}A(\mathrm{p}_2) \cdots \mathrm{d}A(\mathrm{p}_\mathrm{n})$$

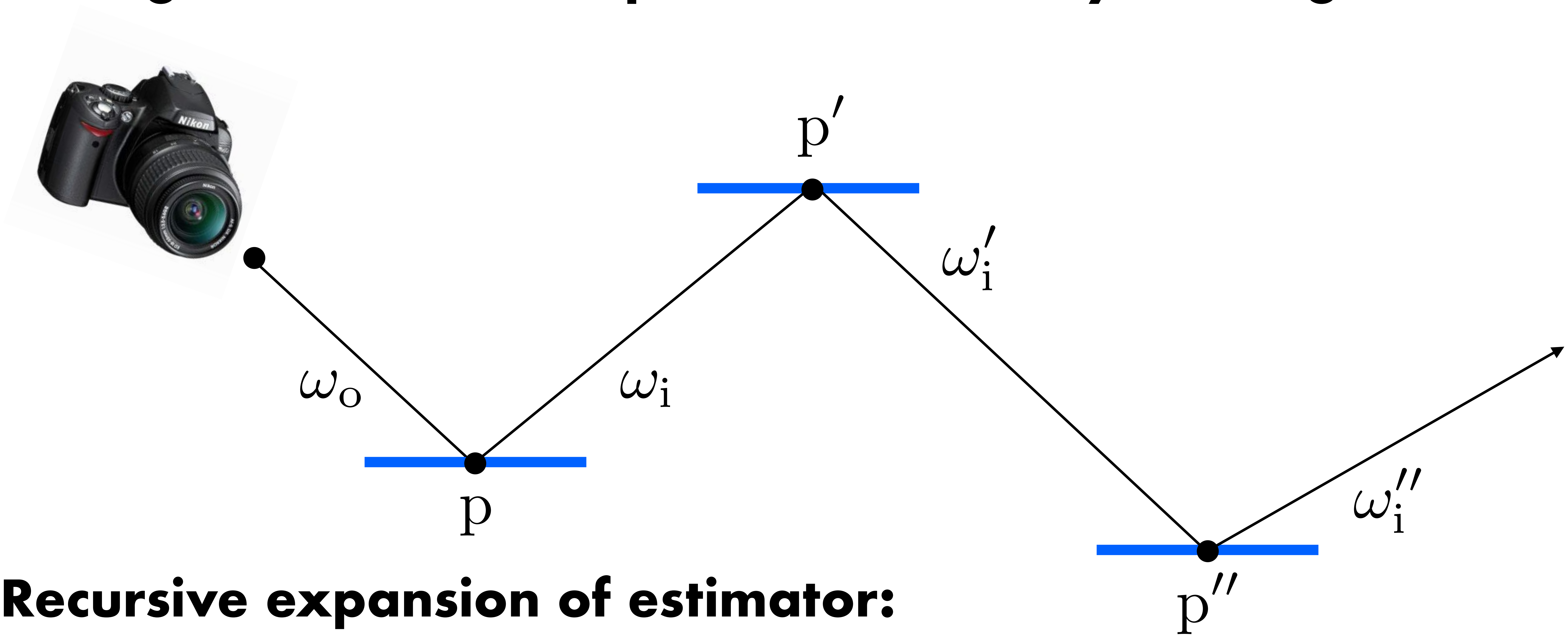
$$= \int_A \int_A \int_A \cdots \int_A L_e(\mathrm{p}_\mathrm{n} \rightarrow \mathrm{p}_\mathrm{n-1}) \boxed{T(\bar{\mathrm{p}}_\mathrm{n})} \mathrm{d}A(\mathrm{p}_2) \cdots \mathrm{d}A(\mathrm{p}_\mathrm{n})$$

Path "throughput" = fraction of light from light source at point 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: indirect illumination modulated by product of probabilities of individual path segments.

Kajiya, 1986

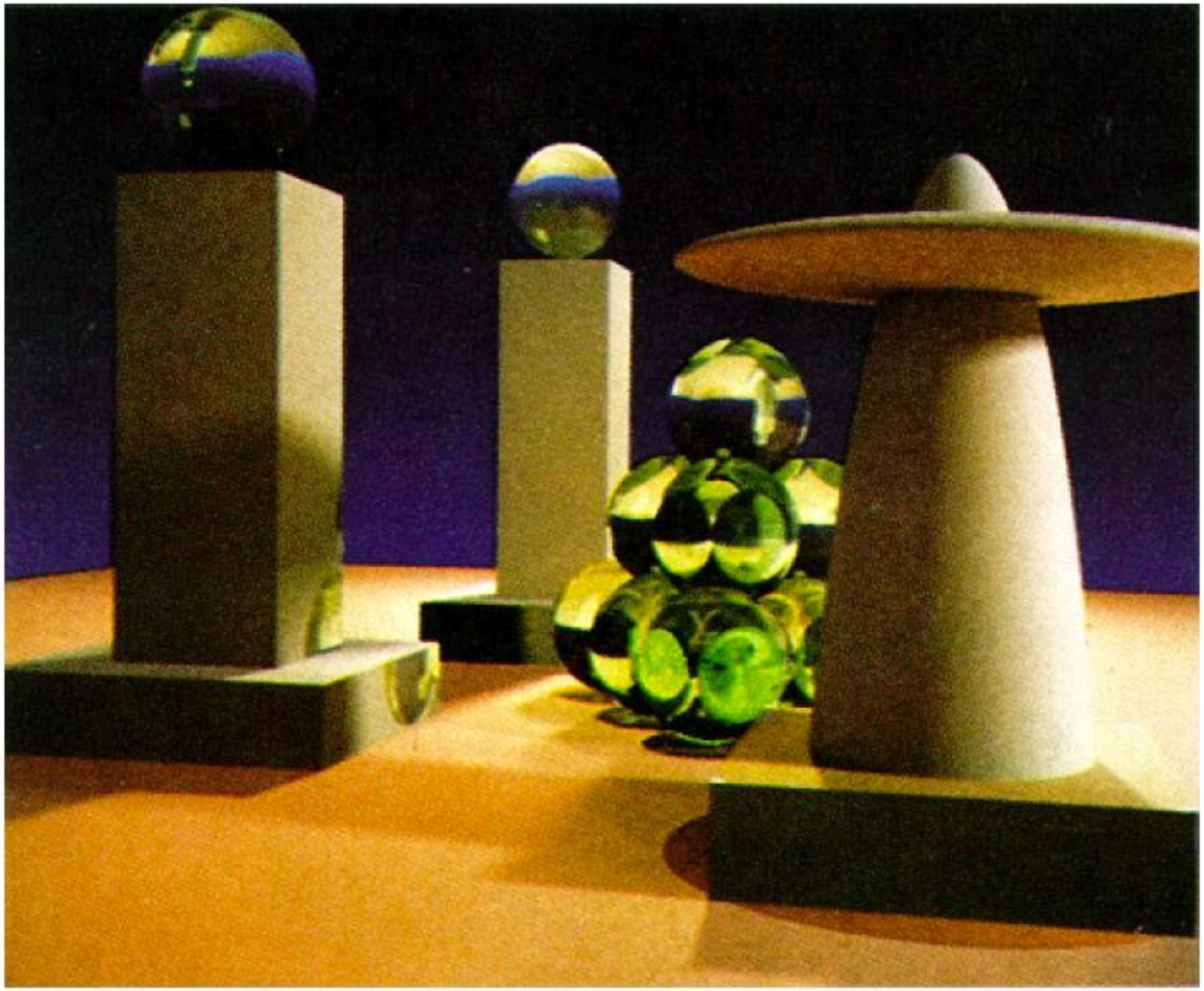
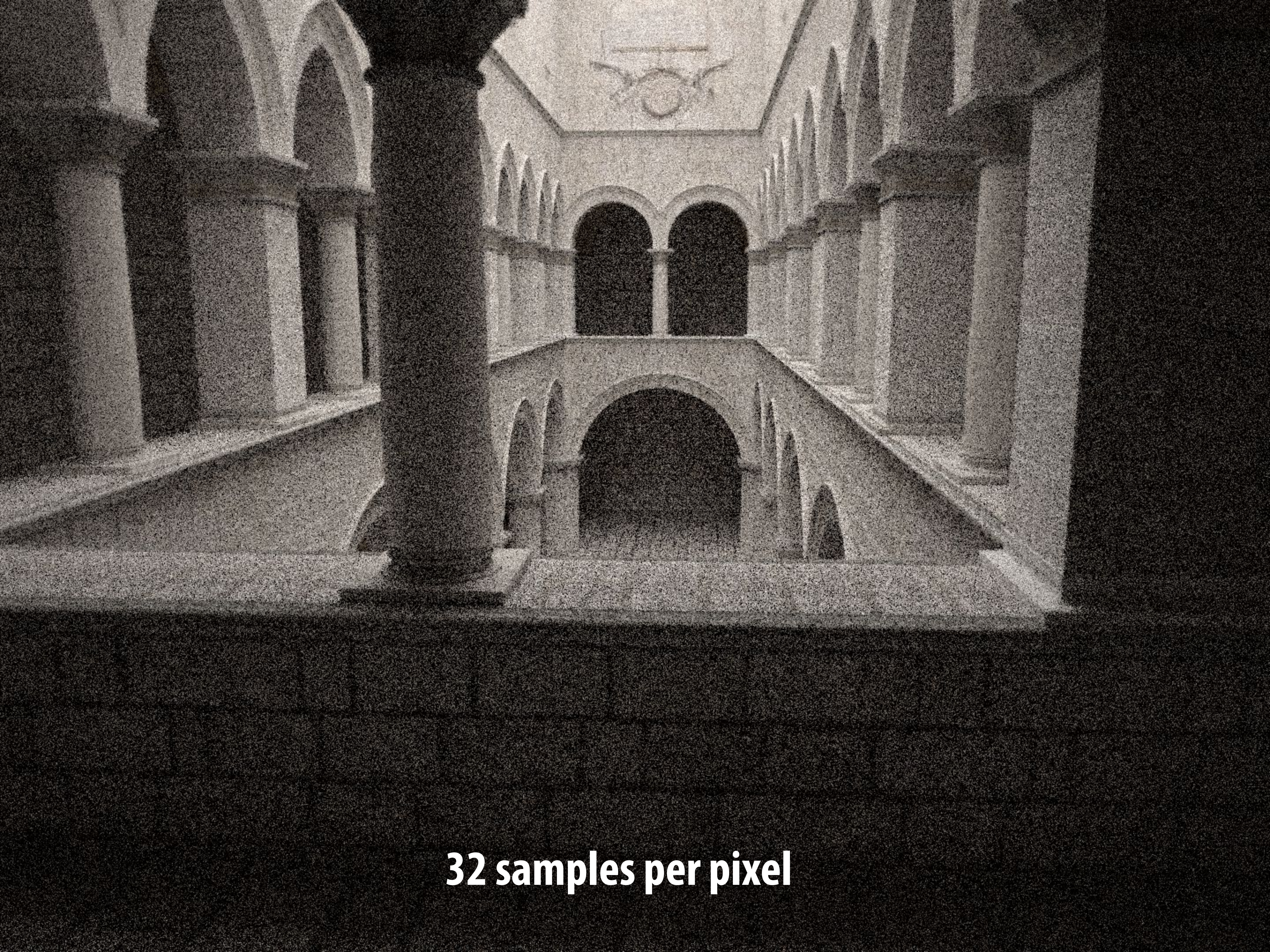


Figure 6. A sample image. All objects are neutral grey. Color on the objects is due to caustics from the green glass balls and color bleeding from the base polygon.

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: $L_{i,d}(\mathrm{p}, \omega_i)$

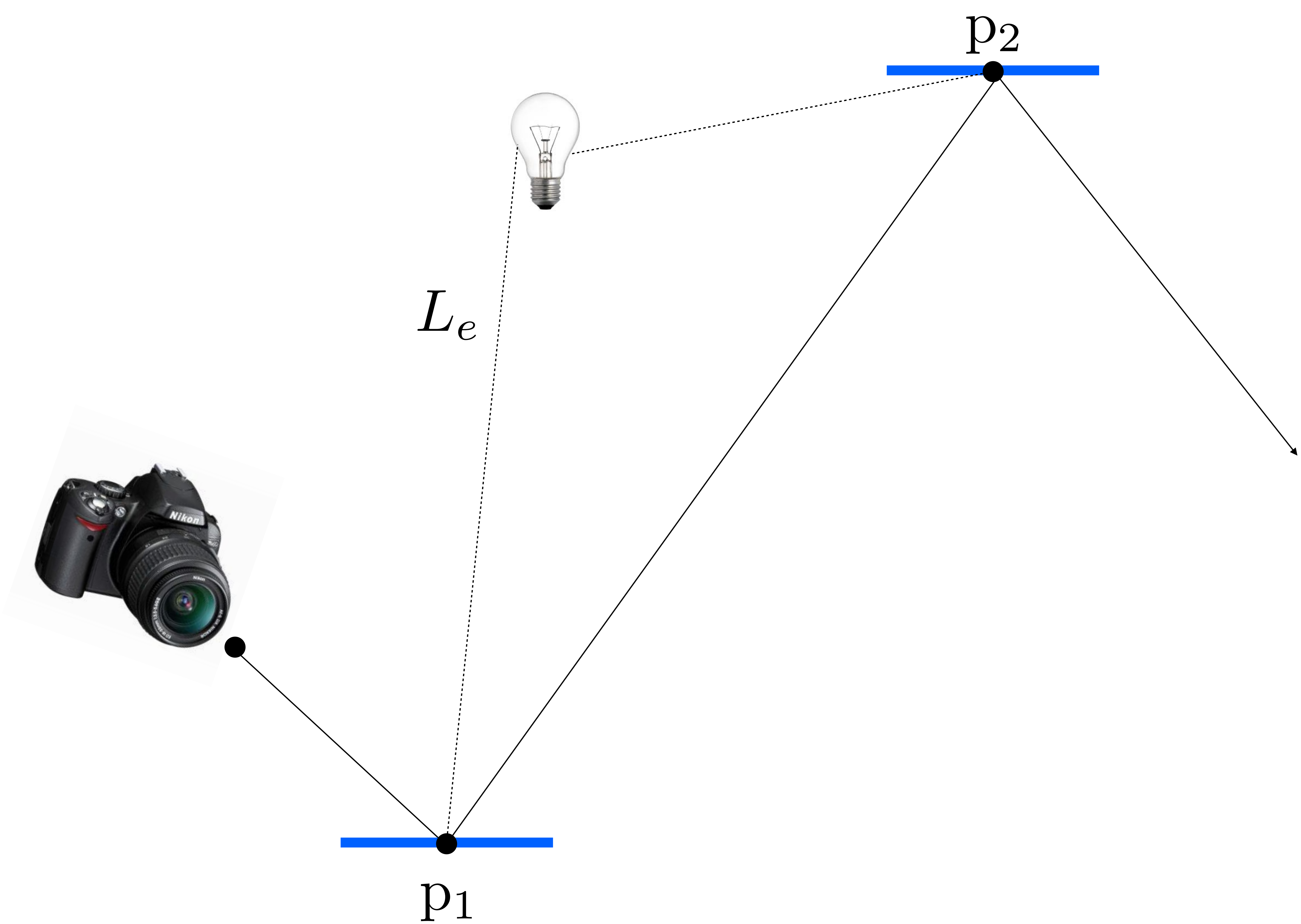
- **Sample lights+BRDFs, use MIS**

Incident indirect illumination: $L_{i,i}(\mathrm{p}, \omega_i)$

- **Recursive evaluation of rendering eqn.**

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

Path Tracing



Path Tracing: 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: $\dfrac{f_r(\omega_i \to \omega_o)\, L_{i,i}(\mathrm{p}, \omega_i)\, \cos\theta_i}{p(\omega_i)}$

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

Path Tracing: Recursive

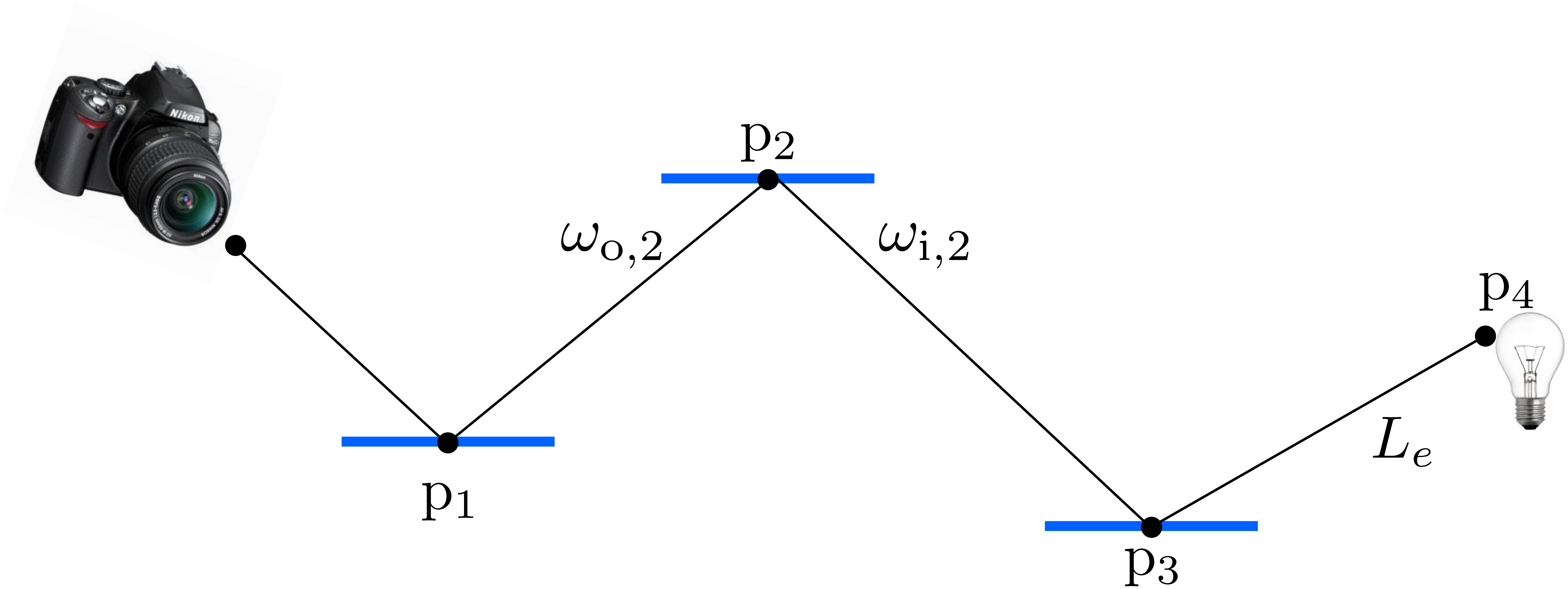
$$L_o(\mathrm{p}, \omega_o) = L_e(\mathrm{p}, \omega_o) + \int_{H^2} f_r(\omega_i \rightarrow \omega_o)\, L_{i,d}(\mathrm{p}, \omega_i)\, \cos\theta_i \, \mathrm{d}\omega_i + \int_{H^2} f_r(\omega_i \rightarrow \omega_o)\, L_o(tr(\mathrm{p}, \omega_i), -\omega_i)\, \cos\theta_i \, \mathrm{d}\omega_i$$

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

  Spectrum Ld = DirectLighting(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;
}
```

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

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

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

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

Problem?

Russian Roulette

Define path termination probability q

Randomly terminate path based on q**;**

for surviving paths, scale contribution by $\dfrac{1}{1-q}$

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;
  while (true) {
    Intersection isect = scene->Intersect(ray);
    Vector3f wo = -ray.d;
    if (depth == 0) Lo += isect.Le(wo);

    BSDF bsdf = isect.GetBSDF();
    Lo += beta * DirectLighting(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);
  }
  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 w_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**

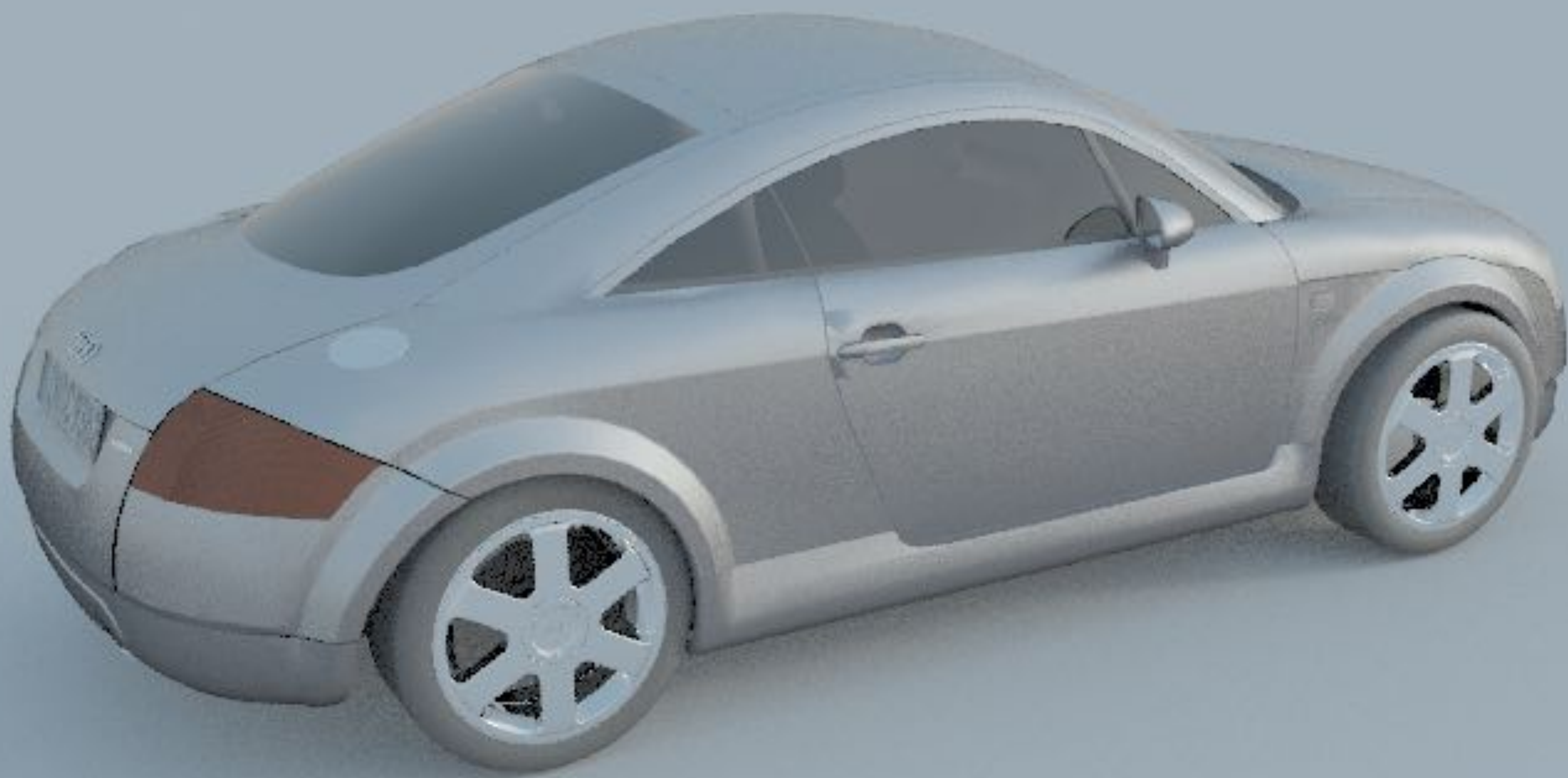
Russian Roulette: Better

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

    BSDF bsdf = isect.GetBSDF();
    Lo += beta * DirectLighting(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;
  }
  return Lo;
}
```

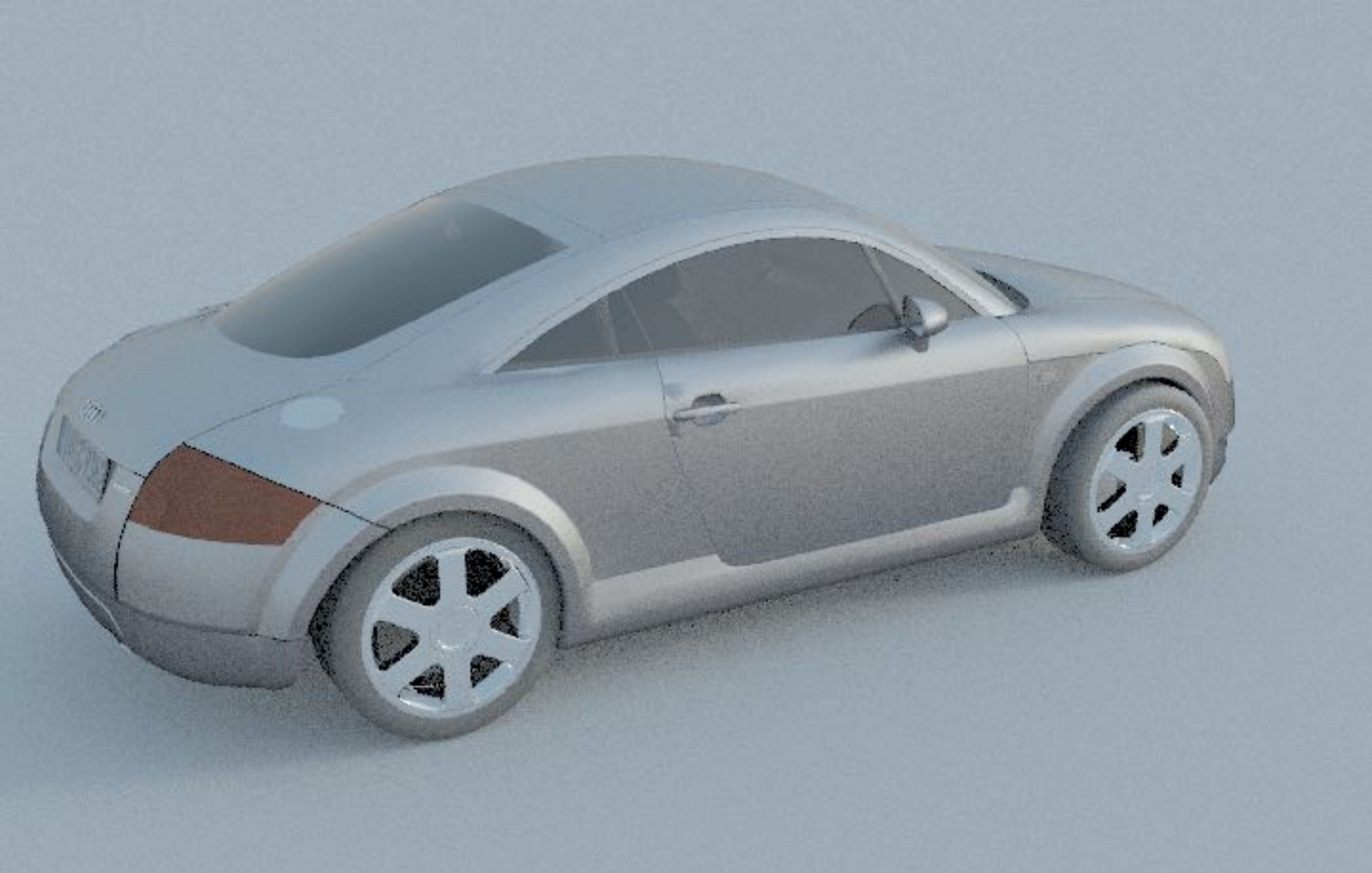


No Russian Roulette: 3.9 seconds
MSE 0.00379, MC Efficiency 67.4

Terminate 50% with luminance < 0.25: 3.3 seconds
MSE 0.003808, MC Efficiency 78.59

Terminate 50% with luminance < 0.5: 3.2 seconds
MSE 0.003846, MC Efficiency 80.54

Terminate 90% with luminance < 1: 2.5 seconds
MSE 0.0124, MC Efficiency 32.90



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

p_{rr} = 0.5, 612s
MSE 1.201, MC Efficiency 0.00136

p_{rr} = max RGB, 3170s
MSE 0.0209, Efficiency 0.01512

Path Guiding

Recall the MC estimator:

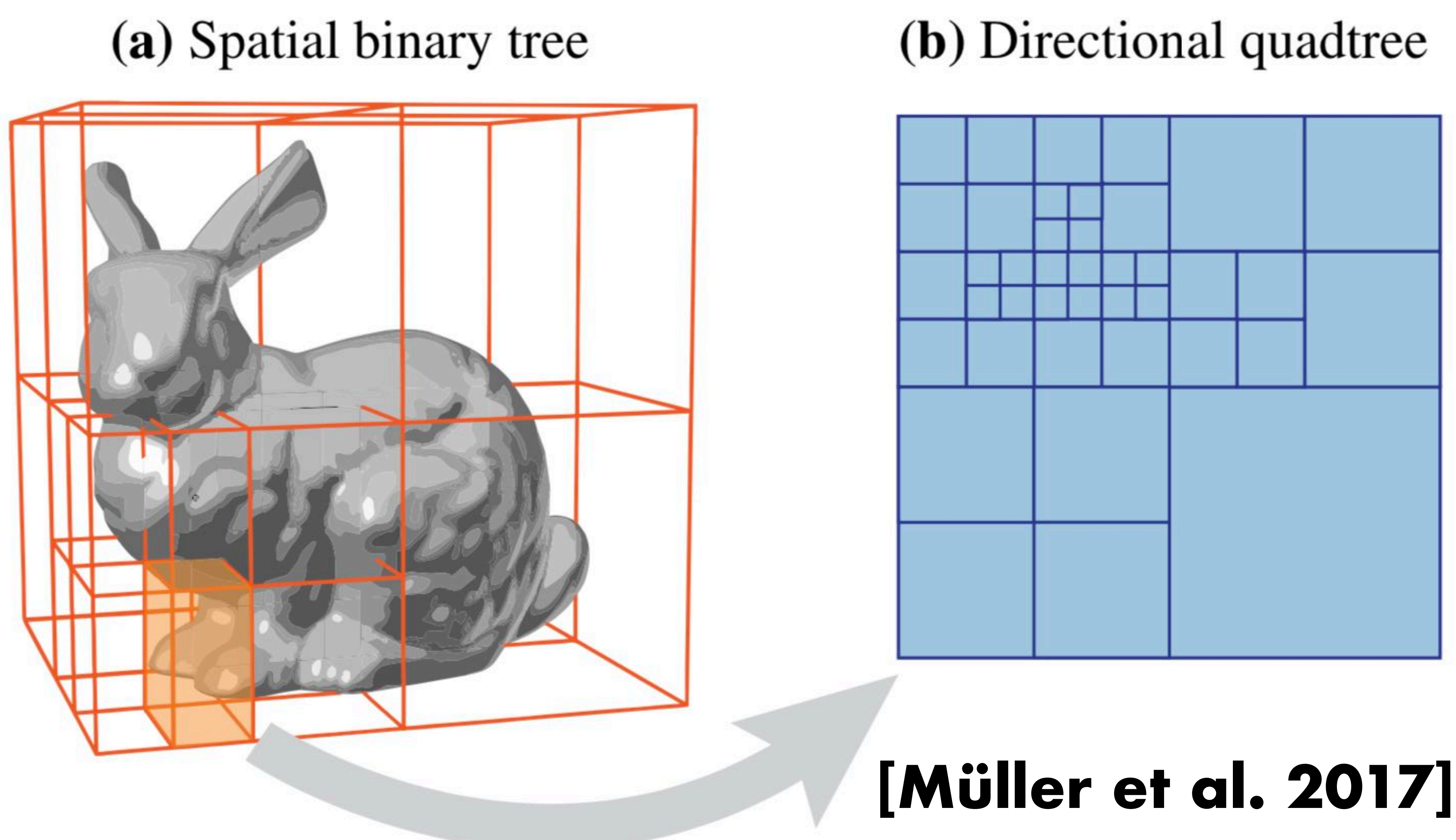
$$\frac{f_r(\omega_i \to \omega_o)\, L_o(tr(\mathrm{p}, \omega_i), -\omega_i)\, \cos\theta_i}{p(\omega_i)}$$

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

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

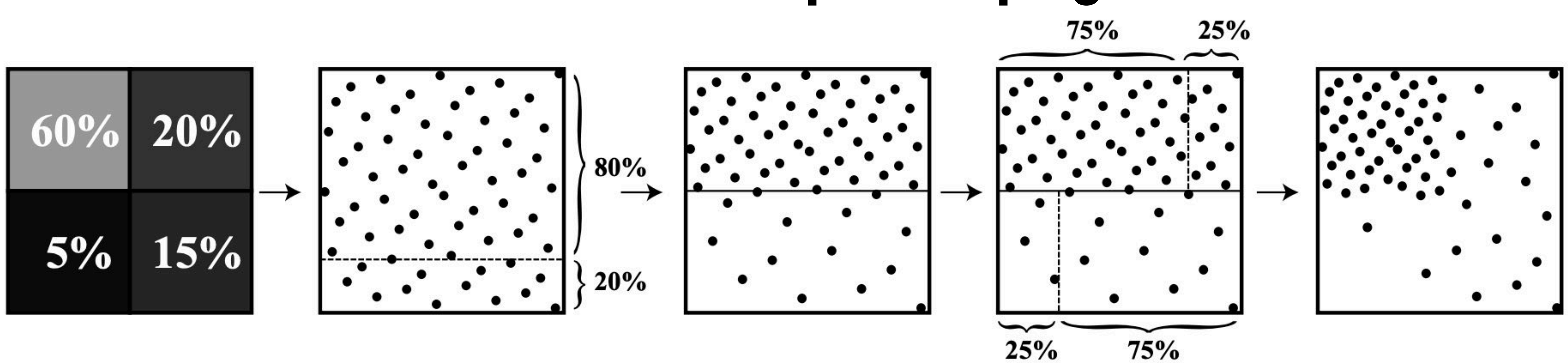
Idea: learn the distribution of light in the scene to guide sampling

Path Guiding



[Müller et al. 2017]

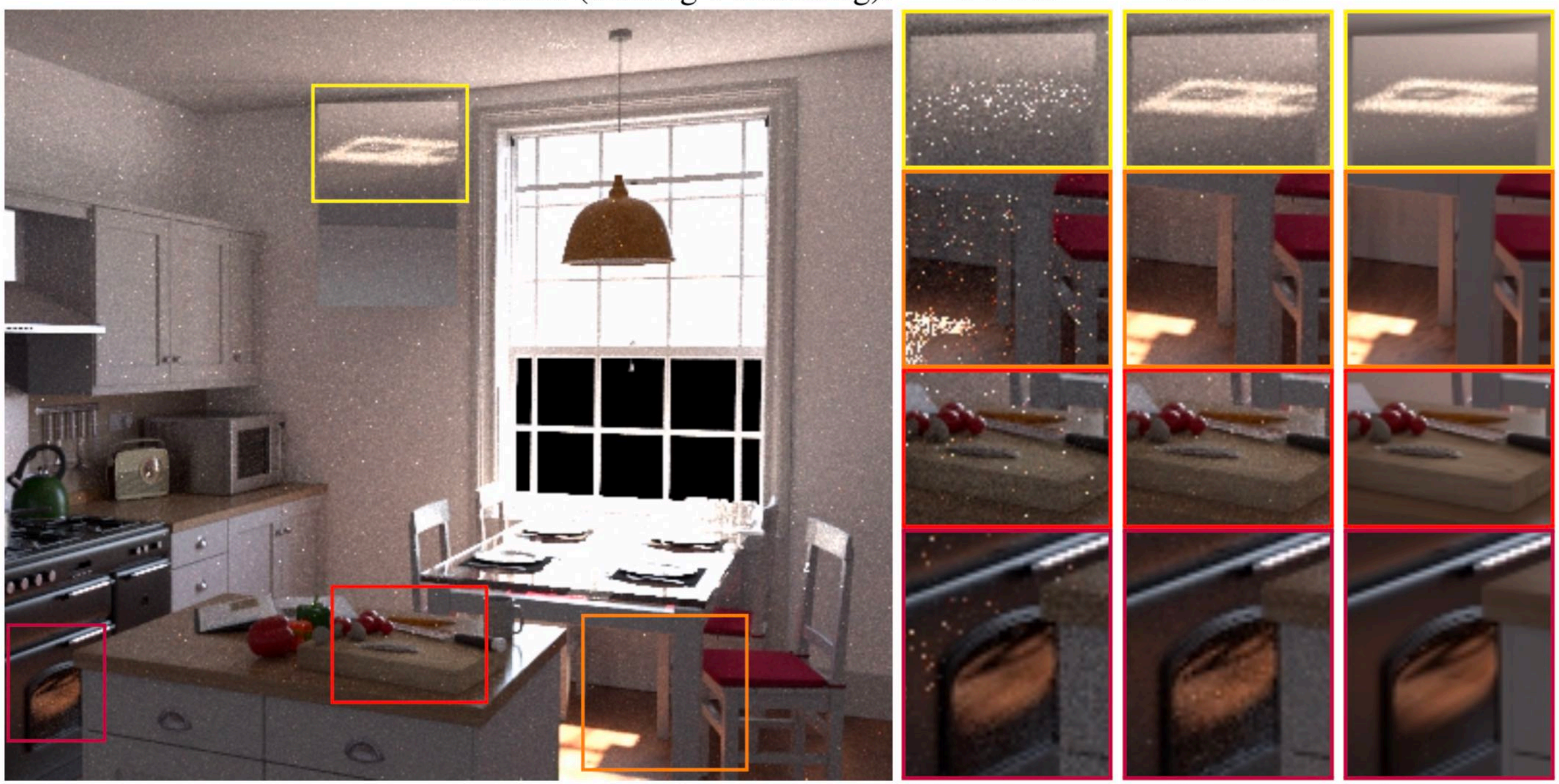
Hierarchical Sample Warping



[Clarberg et al. 2005]

Path Guiding

	PT w/ NEE	Ours	Reference
MSE:	7.949	0.694	—
Samples per pixel:	3100	1812	—
Minutes (training + rendering):	0 + 5.1	1.1 + 3.9	—



[Müller et al. 2017]

Neural Path Guiding

Baseline PPG Neural

[Müller et al. 2018]

Course Projects