CS5630 Physically Based Realistic Rendering

Steve Marschner

“Spring” 2026

11 Path Tracing

Global illumination

We have the tools to compute reflected light given incident light

- BRDF and reflection equation state the problem
- Monte Carlo integration sets up a framework for solving it
- MIS with luminaire and BRDF sampling provides a workable sampling strategy

So far, incident light has been from light sources

- area lights (surfaces that glow)
- environment maps (radiance coming from the scene background)
- in both cases we can just evaluate the incoming radiance in a direction

In the real world, reflected light is also incident light for other surfaces

- leads to a global balance problem

Surface reflection equations

Surface reflection equation

$$L_r(\mathbf{x}, \omega) = \int_{H^2} f_r(\mathbf{x}, \omega, \omega')\, L_i(\omega')\, d\mu(\omega')$$ — where $d\mu(\omega)$ stands for $|\omega \cdot \mathbf{n}|\, d\omega$

- this is simply an integral of known quantities

Outgoing light from a surface that might also emit

$$\begin{aligned} L_o(\mathbf{x}, \omega) &= L_e(\mathbf{x}, \omega) + L_r(\mathbf{x}, \omega) \\ &= L_e(\mathbf{x}, \omega) + \int_{H^2} f_r(\mathbf{x}, \omega, \omega')\, L_i(\omega')\, d\mu(\omega') \end{aligned}$$

The Rendering Equation

Incident light is outgoing light from another surface

- $L_i(\mathbf{x}, \omega) = L_o(\phi(\mathbf{x}, \omega), -\omega)$ — where ϕ is the “ray casting function” returning first intersection

Go ahead and substitute in the surface reflection equation:

$$L_o(\mathbf{x}, \omega) = L_e(\mathbf{x}, \omega) + \int_{H^2} f_r(\mathbf{x}, \omega, \omega')\, L_o(\phi(\mathbf{x}, \omega'), -\omega')\, d\mu(\omega')$$

- this form assumes the scene is closed so that rays always hit something
- now this is an *integral equation*: an equation relating a function to an integral of itself
- general form: $f(x) = g(x) + \int k(x, y)\, f(y)\, dy$, a Fredholm integral equation of the second kind

Solving the Rendering Equation

Go ahead and substitute the equation into itself…

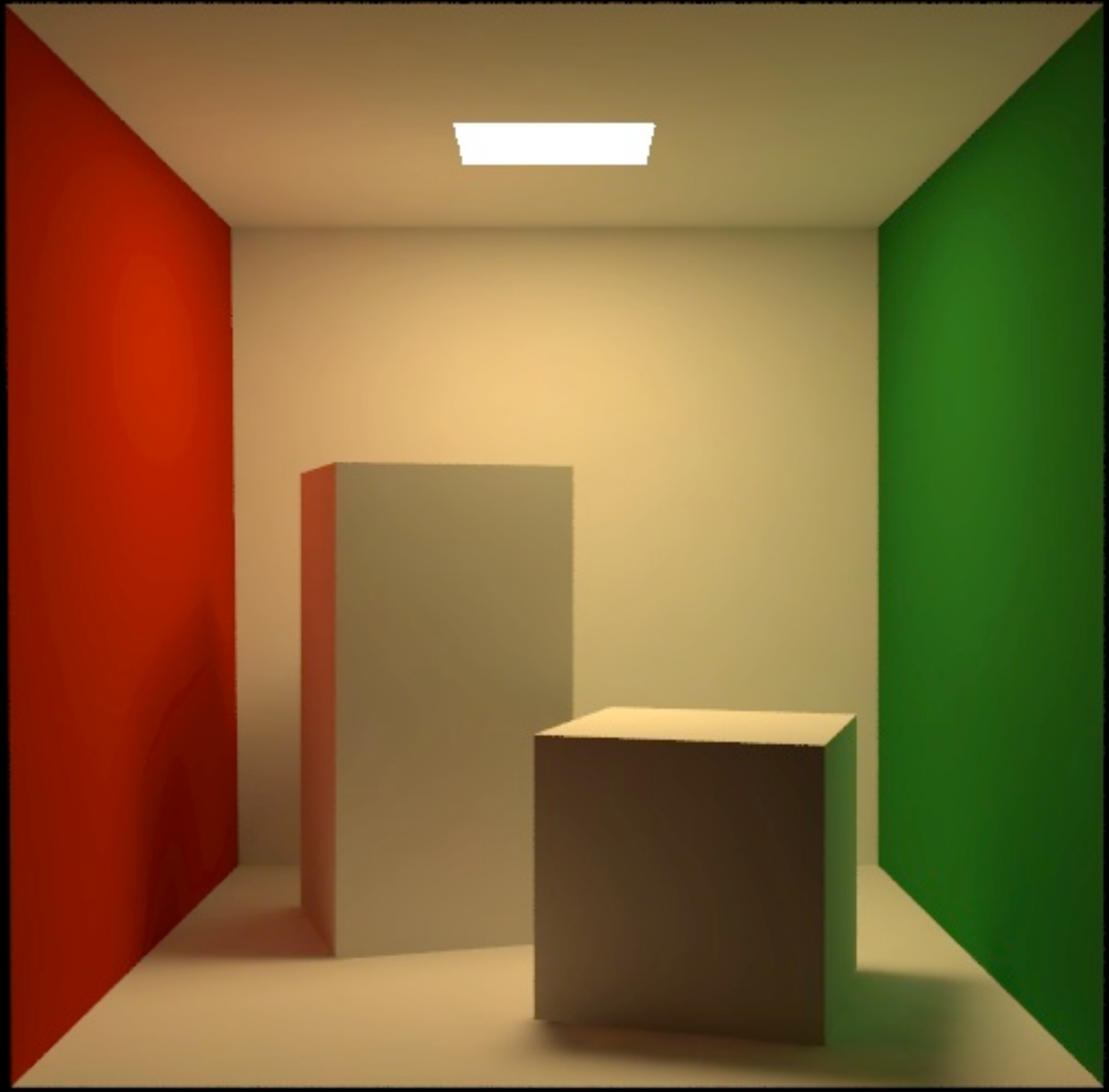
$$
\begin{aligned}
L_o(\mathbf{x}, \omega) &= L_e(\mathbf{x}, \omega) + \int_{H^2} f_r(\mathbf{x}, \omega, \omega') \left[L_e(\mathbf{x}', -\omega') + \int_{H^2} f_r(\mathbf{x}', -\omega', \omega'')\, L_o(\phi(\mathbf{x}', \omega''), -\omega'')\, d\mu(\omega'') \right] d\mu(\omega') \\
&= L_e(\mathbf{x}, \omega) \\
&\quad + \int_{H^2} f_r(\mathbf{x}, \omega, \omega')\, L_e(\mathbf{x}, -\omega') \\
&\quad + \int_{H^2} \int_{H^2} f_r(\mathbf{x}, \omega, \omega') f_r(\mathbf{x}, -\omega', \omega'')\, L_e(\mathbf{x}, -\omega'') \\
&\quad + \ldots
\end{aligned}
$$

Implications

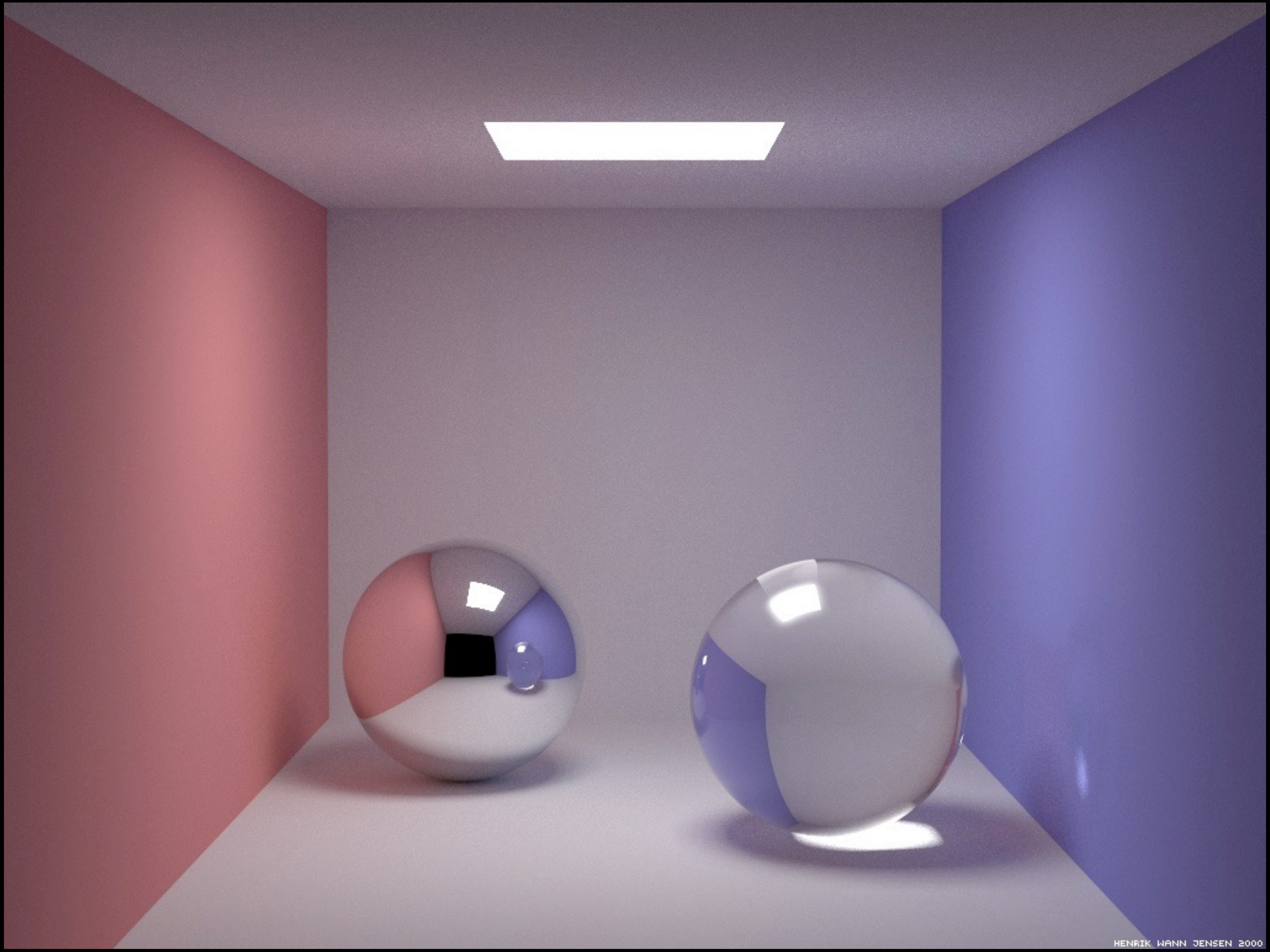
Surface radiance is a sum:

- emitted radiance
- radiance reflected once (direct illumination)
- radiance reflected twice, three times, … (indirect illumination)

Let’s look at some images…

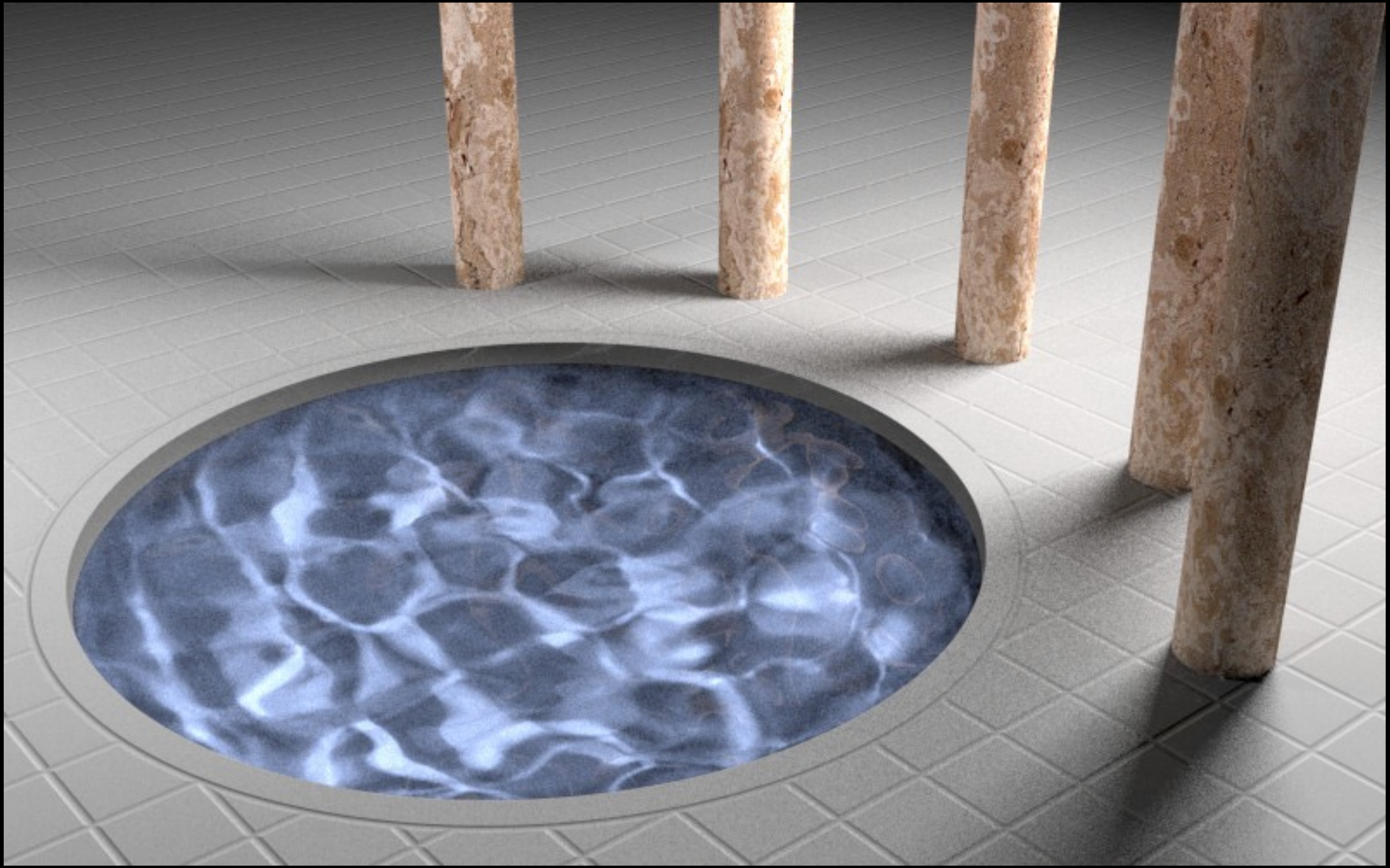


Cornell PCG



Henrik Wann Jensen

Veach & Guibas 1994



Veach & Guibas 1997

Kajiya-style path tracing, version 0:

```
rayRadianceEst(x, ω):
   y = traceRay(x, ω)
   return emittedRadiance(y, –ω) + reflectedRadianceEst(y, –ω)

reflectedRadianceEst(x, ωr):
   ωi = uniformRandomPSA(n(x))
   return π * brdf(x, ωi, ωr) * rayRadianceEst(x, ωi)
```

Kajiya-style path tracing, version 0.5:

```
rayRadianceEst(x, ω):
   y = traceRay(x, ω)
   return emittedRadiance(y, –ω) + reflectedRadianceEst(y, –ω)

reflectedRadianceEst(x, ωr):
   if random() < survivalProbability:
      ωi = uniformRandomPSA(n(x))
      return π * brdf(x, ωi, ωr) * rayRadianceEst(x, ωi) / survivalProbability
   else
      return 0
```

Kajiya-style path tracing, version 0.75:

```
rayRadianceEst(x, ω):
   y = traceRay(x, ω)
   return emittedRadiance(y, –ω) + reflectedRadianceEst(y, –ω)

reflectedRadianceEst(x, ωr):
   if random() < survivalProbability:
      ωi, pdf = brdfSample(x, n(x))
      return brdf(x, ωi, ωr) * rayRadianceEst(x, ωi) / (pdf * survivalProbability)
   else
      return 0
```

Kajiya-style path tracing, version 1.0:

```
rayRadianceEst(x, ω):
   y = traceRay(x, ω)
   return emittedRadiance(y, –ω)
      + reflectedRadianceEst(y, –ω)

directRadianceEst(x, ωr):
   ωi, pdf = luminaireSample(x, n(x))
   y = traceRay(x, ωi)
   return brdf(x, ωi, ωr)
      * emittedRadiance(y, –ωi) / pdf

reflectedRadianceEst(x, ωr):
   return directRadianceEst(x, ωr)
      + indirectRadianceEst(x, ωr)

indirectRadianceEst(x, ωr):
   if random() < survivalProbability:
      ωi, pdf = brdfSample(x, n(x))
      y = traceRay(x, ωi)
      return brdf(x, ωi, ωr)
         * reflectedRadianceEst(y, –ωi)
         / (pdf * survivalProbability)
   else:
      return 0
```

Kajiya-style path tracing, version 1.0m:

```
directRadianceEst(x, ωr):
   ωl, pll = luminaireSample(x, n(x))
   pbl = brdfPDF(ωl)
   ωb, pbb = brdfSample(x, n(x))
   plb = luminairePDF(ωb)
   yl = traceRay(x, ωl)
   yb = traceRay(x, ωb)
   fl = brdf(x, ωl, ωr)
      * emittedRadiance(yl, –ωi)
   fb = brdf(x, ωb, ωr)
      * emittedRadiance(yb, –ωi)
   return fl / (pll + pbl) + fb / (plb + pbb)
```

```
reflectedRadianceEst(x, ωr):
   return directRadianceEst(x, ωr)
      + indirectRadianceEst(x, ωr)

indirectRadianceEst(x, ωr):
   if random() < survivalProbability:
      ωi, pdf = brdfSample(x, n(x))
      y = traceRay(x, ωi)
      return brdf(x, ωi, ωr)
         * reflectedRadianceEst(y, –ωi)
         / (pdf * survivalProbability)
   else:
      return 0
```

Kajiya-style path tracing, version 1.1:

```
reflectedRadianceEst(x, ωr):
   ωl, pll = luminaireSample(x, n(x))
   pbl = brdfPDF(ωl)
   ωb, pbb = brdfSample(x, n(x))
   plb = luminairePDF(ωb)
   yl = traceRay(x, ωl)
   yb = traceRay(x, ωb)
   fl = brdf(x, ωl, ωr)
      * emittedRadiance(yl, –ωl)
   fb = brdf(x, ωb, ωr)
      * emittedRadiance(yb, –ωb)
   reflRad = fl / (pll + pbl) + fb / (plb + pbb)
   if random() < survivalProbability:
      reflRad += brdf(x, ωb, ωr) / pbb
         * reflectedRadianceEst(yb, –ωb)
         / survivalProbability
   return reflRad
```