Computer graphics III – Path tracing

Jaroslav Křivánek, MFF UK
Jaroslav.Krivanek@mff.cuni.cz

Tracing paths from the camera

```
renderImage()
{
  for all pixels
  {
    Color pixelColor = (0,0,0);
    for k = 1 to N
    {
      ωk := random direction through the pixel
      pixelColor += getLi(camPos, ωk)
    }
    pixelColor /= N;
    writePixel(pixelColor);
  }
}
```

Path tracing, v. zero (recursive form)

```
getLi (x, ω):
    y = traceRay(x, ω)
    return
        Le(y, –ω) +                      // emitted radiance
        Lr (y, –ω)                       // reflected radiance

Lr(x, ω):
    ω′ = genUniformHemisphereRandomDir( n(x) )
    return 2π * brdf(x, ω, ω′) * dot(n(x), ω′) * getLi(x, ω′)
```

Path Tracing – Loop version

```
getLi(x, w)
{
   Color thrput = (1,1,1)
   Color accum  = (0,0,0)
   while(1)
   {
        hit = NearestIntersect(x, w)
        if no intersection
                return accum + thrput * bgRadiance(x, w)
        if isOnLightSource(hit)
                accum += thrput * Le(hit.pos, -w)
        ρ = reflectance(hit.pos, -w)
        if rand() < ρ // russian roulette – survive (reflect)
                wi  := SampleDir(hit)
                thrput *= fr(hit.pos, wi, -w) * dot(hit.n, wi) / (ρ * pdf(wi))
                x   := hit.pos
                w   := wi
        else // absorb
            break;
   }
   return accum;
}
```

Terminating paths – Russian roulette

```
getLi(x, w)
{
   Color thrput = (1,1,1)
   Color accum  = (0,0,0)
   while(1)
   {
        hit = NearestIntersect(x, w)
        if no intersection
                return accum + thrput * bgRadiance(x, w)
        if isOnLightSource(hit)
                accum += thrput * Le(hit.pos, -w)
        ρ = reflectance(hit.pos, -w)
        if rand() < ρ // russian roulette – survive (reflect)
                wi  := SampleDir(hit)
                thrput *= fr(hit.pos, wi, -w) * dot(hit.n, wi) / (ρ * pdf(wi))
                x   := hit.pos
                w   := wi
        else // absorb
            break;
   }
   return accum;
}
```

Terminating paths – Russian roulette

- Continue the path with probability q
- Multiply weight (throughput) of surviving paths by 1 / q

$$Z = \begin{cases} Y / q & \text{if } \xi < q \\ 0 & \text{otherwise} \end{cases}$$

- RR is unbiased!

$$E[Z] = \frac{E[Y]}{q} \cdot q + 0 \cdot \frac{1}{q-1} = E[Y]$$

Survival probability

- It makes sense to use the surface reflectivity ρ as the survival probability
  - If the surface reflects only 30% of energy, we continue with the probability of 30%. That's in line with what happens in reality.
- What if we cannot calculate ρ? Then there's a convenient alternative:
  1. First sample a random direction ω_i according to $p(\omega_i)$
  2. Use the sampled ω_i it to calculate the survival probability as

$$q_{\text{survival}} = \min\left\{ 1,\ \frac{f_r(\omega_i \to \omega_o)\cos\theta_i}{p(\omega_i)} \right\}$$

Direction sampling

```
getLi(x, w)
{
   Color thrput = (1,1,1)
   Color accum  = (0,0,0)
   while(1)
   {
        hit = NearestIntersect(x, w)
        if no intersection
                return accum + thrput * bgRadiance(x, w)
        if isOnLightSource(hit)
                accum += thrput * Le(hit.pos, -w)
        ρ = reflectance(hit.pos, -w)
        if rand() < ρ // russian roulette – survive (reflect)
                wi := SampleDir(hit)
                thrput *= fr(hit.pos, wi, -w) * dot(hit.n, wi) / (ρ * pdf(wi))
                x  := hit.pos
                w  := wi
        else // absorb
            break;
   }
   return accum;
}
```

Direction sampling

- We usually sample the direction ω_i from a pdf similar to

$$f_r(\omega_i, \omega_o) \cos \theta_i$$

  - Ideally, we would want to sample proportionally to the integrand itself

$$L_i(\omega_i)\, f_r(\omega_i, \omega_o) \cos \theta_i,$$

    but this is difficult, because we do not know L_i upfront. With some precomputation, it is possible to use a rough estimate of L_i for sampling [Jensen 95, Vorba et al. 2014], cf. "guiding".

No incoming radiance information [Vorba et al. 2014]

"Guiding" by incoming radiance [Vorba et al. 2014]

BRDF importance sampling

- Let's see what happens when the pdf is **exactly proportional** to $f_r(\omega_i, \omega_o) \cos \theta_i$?

$$p(\omega_i) \propto f_r(\omega_i \to \omega_o) \cdot \cos\theta_i$$

- Normalization (recall that a pdf must integrate to 1)

$$p(\omega_i) = \frac{f_r(\omega_i \to \omega_o) \cdot \cos\theta_i}{\boxed{\int_{H(\mathbf{x})} f_r(\omega_i \to \omega_o) \cdot \cos\theta_i \, \mathrm{d}\omega_i}}$$

The normalization factor is nothing but the reflectance ρ

BRDF IS in a path tracer

- Throughput update for a general pdf

```
...
thrput *= fr(.) * dot(.) / ( ρ * p(wi) )
```

- A pdf that is exactly proportional to BRDF * cos keeps the throughput constant because the different terms cancel out!

$$p(\omega_\mathrm{i}) = f_r(\omega_\mathrm{i} \to \omega_\mathrm{o}) \cdot \cos\theta_i \; / \; \rho$$

```
...
thrput *= 1
```

  - Physicists and nuclear engineers call this the “**analog**” simulation, because this is how real particles behave.

Direct illumination calculation in a path tracer

Direct illumination: Two strategies

- At each path vertex **x**, we are calculating **direct illumination**
  - i.e. radiance reflected from a point **x** on a surface exclusively due to the light coming directly from the sources

- Two sampling strategies
  1. **BRDF-proportional sampling**
  2. **Light source area sampling**

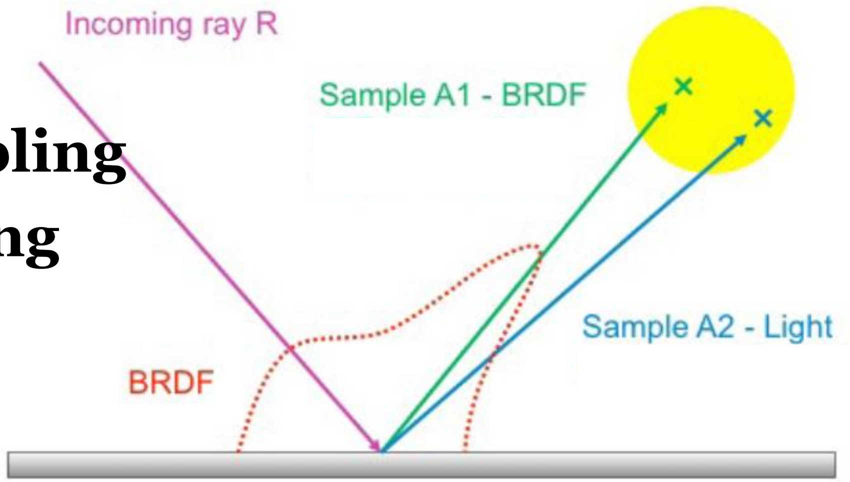
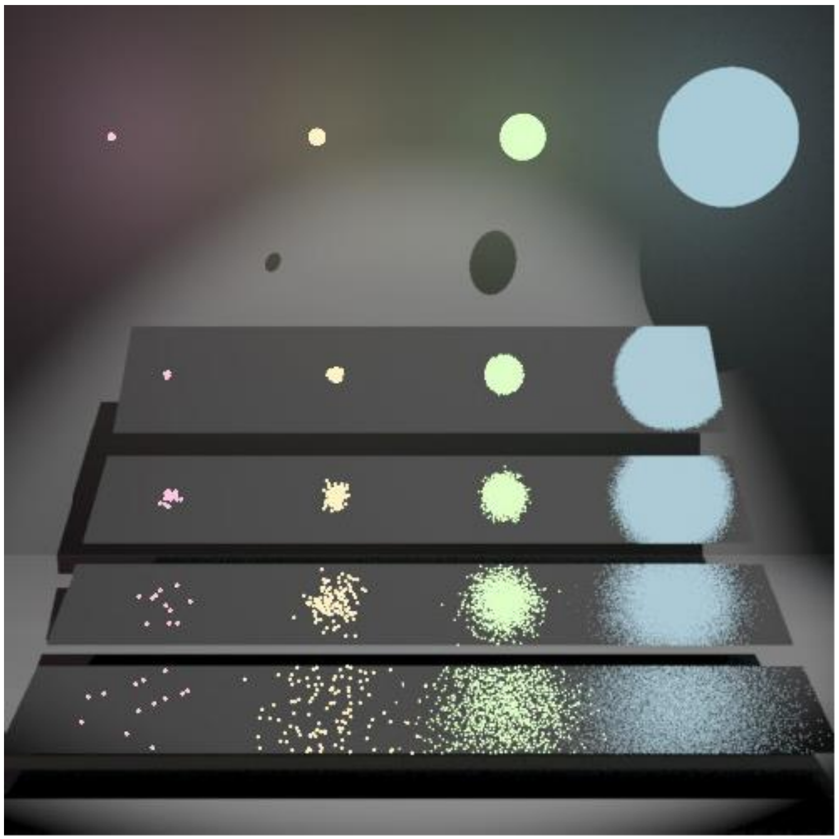
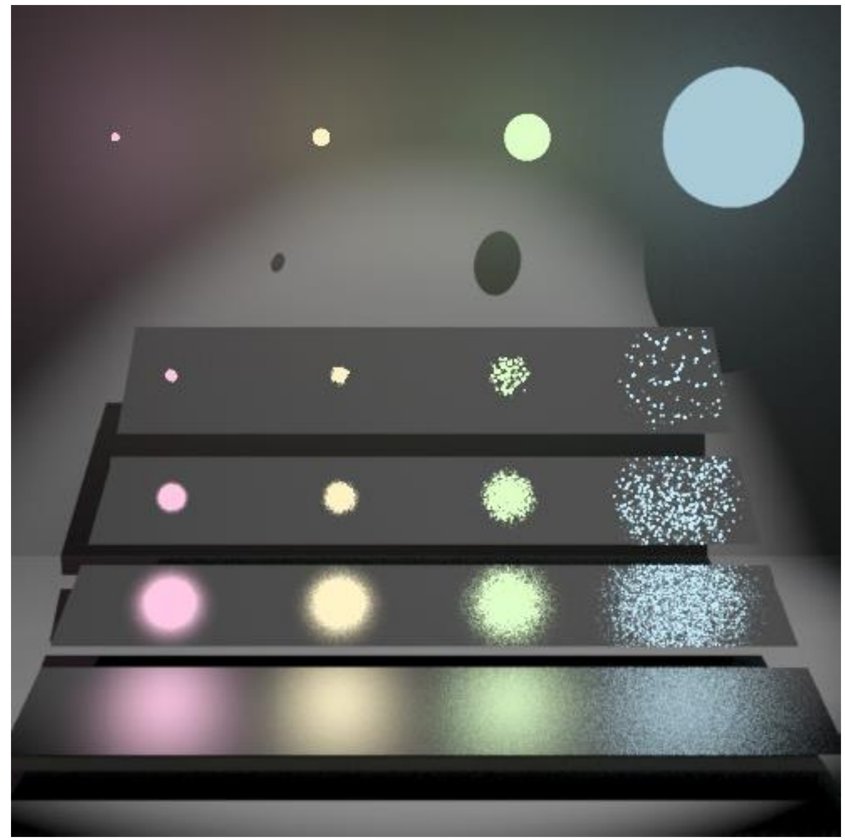


Image: Alexander Wilkie

Direct illumination: Two strategies



BRDF proportional sampling



Light source area sampling

Images: Eric Veach

Direct illumination calculation using MIS

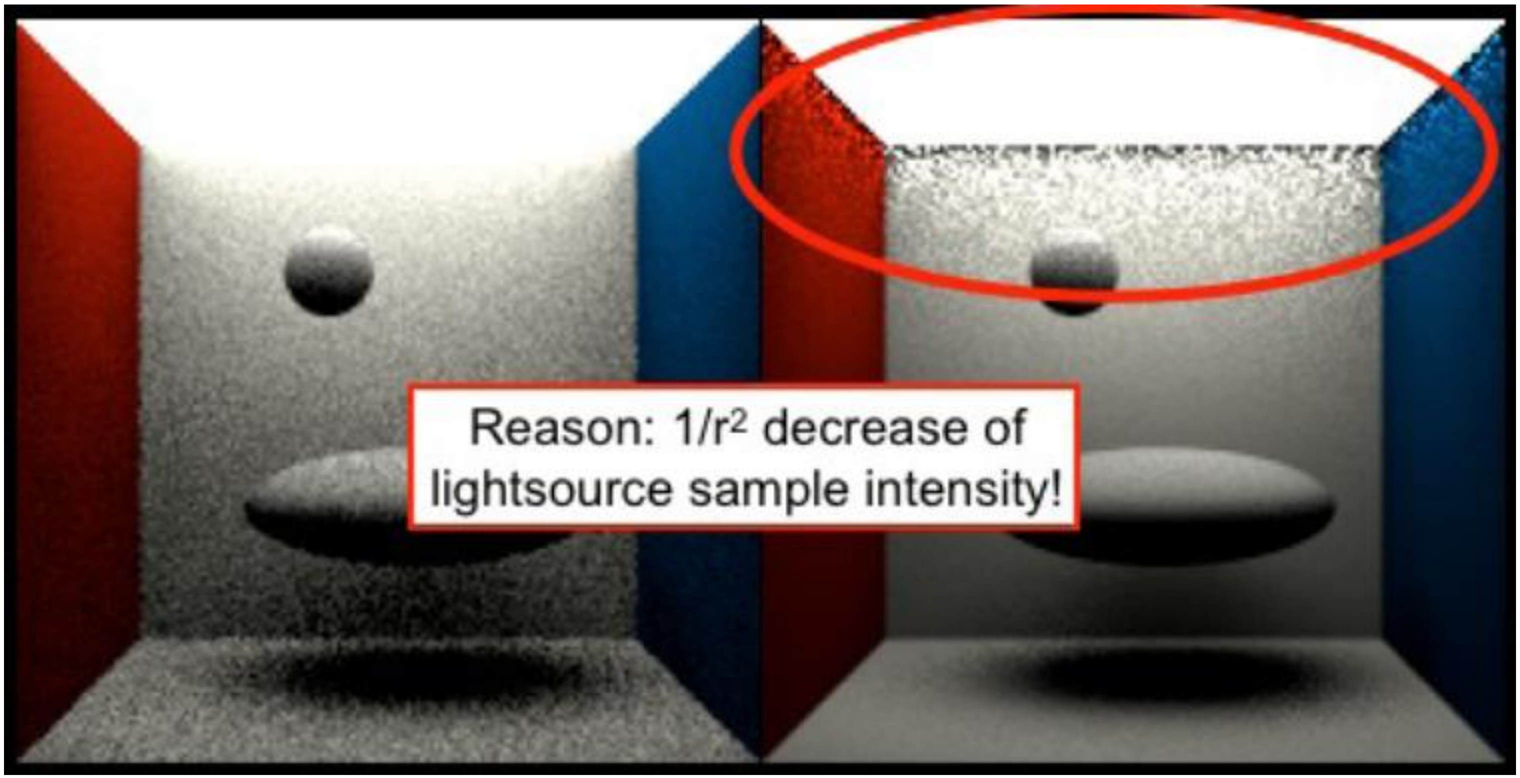
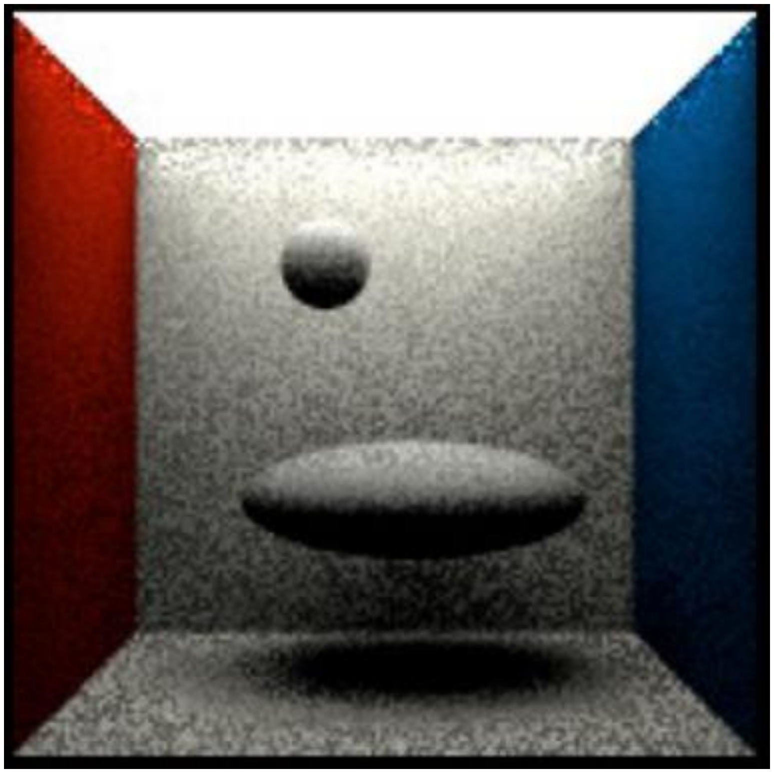


Image: Alexander Wilkie

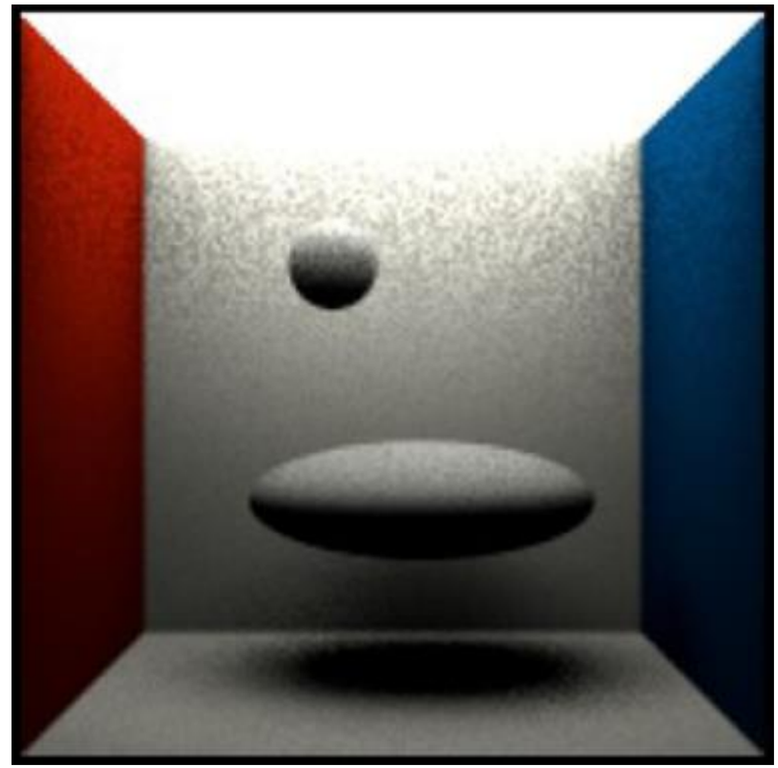
Sampling technique (pdf) p_1:
BRDF sampling

Sampling technique (pdf) p_2:
Light source area sampling

Combination



Arithmetic average
Preserves **bad** properties
of both techniques



Balance heuristic
Bingo!!!

Image: Alexander Wilkie

MIS weight calculation

Sample weight for BRDF sampling

$$w_1(\omega_j) = \frac{p_1(\omega_j)}{p_1(\omega_j) + p_2(\omega_j)}$$

PDF for BRDF sampling

PDF with which the direction ω_j would have been generated, if we used light source area sampling

PDFs

- **BRDF sampling: $p_1(\omega)$**
  - Depends on the BRDF, e.g. for a Lambertian BRDF:

$$p_1(\omega) = \frac{\cos\theta_{\mathbf{x}}}{\pi}$$

- **Light source area sampling: $p_2(\omega)$**

$$p_2(\omega) = \frac{1}{|A|} \underline{\frac{\| \mathbf{x} - \mathbf{y} \|^2}{\cos\theta_{\mathbf{y}}}}$$

Conversion of the uniform pdf 1/|A| from the area measure (dA) to the solid angle measure (dω)

Where is the conversion factor coming from?

- Pdfs (unlike ordinary function) change under a change of coordinates. In general, it must always hold:

$$p(\omega)d\omega = p(\mathbf{y})dA$$

- And so

$$p(\omega) = p(\mathbf{y})\frac{dA}{d\omega}$$

conversion factor

The use of MIS in a path tracer

- For each path vertex:
  - Generate an explicit shadow ray for the techniques p_2 (light source area sampling)
  - Secondary ray for technique p_1 (BRDF sampling)
    - One ray can be shared for the calculation of both **direct** and **indirect** illumination
    - But the MIS weight is – of curse – applied only on the direct term (indirect illumination is added unweighted because there is no second technique to calculate it)

Dealing with multiple light sources

- Option 1:
  - Loop over all sources and send a shadow ray to each one
- Option 2:
  - Choose one source at random (with prob proportional to power)
  - Sample illumination only on the chosen light, divide the result by the prob of picking that light
  - (Scales better with many sources but has higher variance per path)
- Beware: The probability of choosing a light influences the sampling pds and therefore also the MIS weights.