Stochastic Path Tracing


Pacific Rim. Arnold renderer

Georgios Papaioannou - 2015

MONTE CARLO INTEGRATION BASICS

AUEB COMPUTER GRAPHICS GROUP

Monte Carlo Integration (1)

- Is an important tool for evaluating the global illumination equation
- Evaluates integrals based on a selection of random samples in the integration domain

- Given an integral of an arbitrary function $f(x)$ in the interval $[a,b]$ and a uniform selection of N random samples in $[a,b]$:

$$I = \int_a^b f(x)dx \Rightarrow < I >= \frac{(b-a)}{N} \sum_{i=1}^{N} f(x_i)$$

Monte Carlo Integration (2)

Proof:

$$E[\langle I \rangle] = E[\frac{(b-a)}{N}\sum_{i=1}^{N} f(x_i)] = \frac{(b-a)}{N}\sum_{i=1}^{N} E[f(x_i)] =$$

$$\frac{(b-a)}{N}\sum_{i=1}^{N}\int_a^b f(x)\,p(x)dx = \frac{(b-a)}{N}\cdot N\cdot\int_a^b \frac{f(x)}{(b-a)}dx =$$

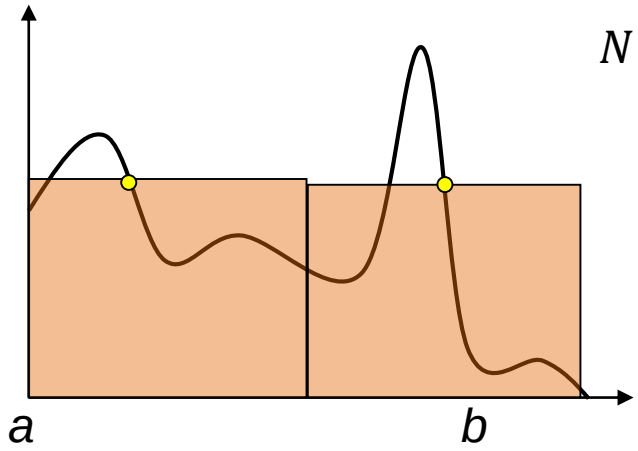
$$\left[p(x) = \frac{1}{b-a}\right]$$

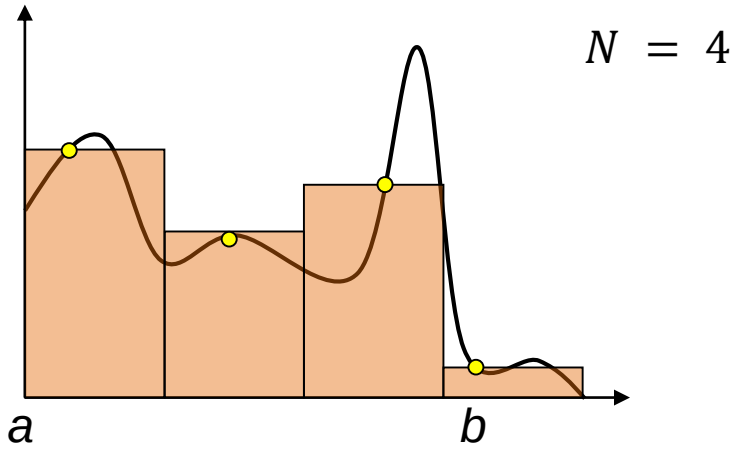
$$= \int_a^b f(x)dx = I$$

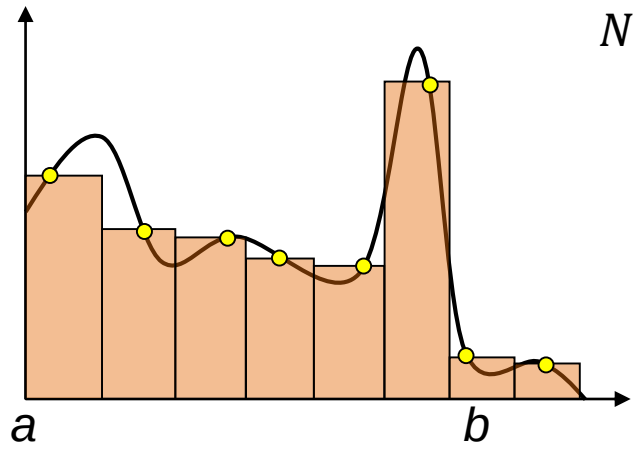
- Every different computation of $\langle I \rangle$ will yield a different result
- But on average, we will get the same answer

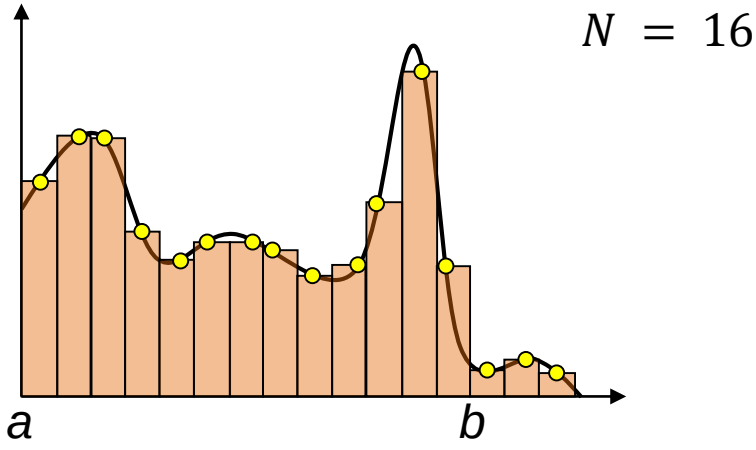
Monte Carlo Integration (3)

Examples of uniform sampling:









Monte Carlo Integration - Advantages

- Works no matter how complex the function to be integrated is (even discontinuous)
- Does not even require the a priori knowledge of the integrand!

Monte Carlo Integration - Drawbacks

- Slow converge rate: when drawing N samples, we expect a converge rate of $1/\sqrt{N}$
  - E.g., to decrease the error by a factor of 2 → requires 4*N* samples
- The converge speed:
  - Is lower than that of many other integration techniques, but
  - Is independent of the number of dimensions in the integral

Importance Sampling (1)

- Drawing the samples with a non-uniform distribution with pdf $p(x)$ can bias the sample selection towards significant directions:

$$\langle I \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{p(x_i)}, \qquad \sigma^2[\langle I \rangle] = \frac{1}{N} \int_a^b \left(\frac{f(x)}{p(x)} - I\right)^2 dx$$

- Again $E(\langle I \rangle) = I$

- Requirements:
  - $p(x) > 0$ for every x in $[a,b]$
  - $\int_a^b p(x)dx = 1$

Importance Sampling (2)

Proof:

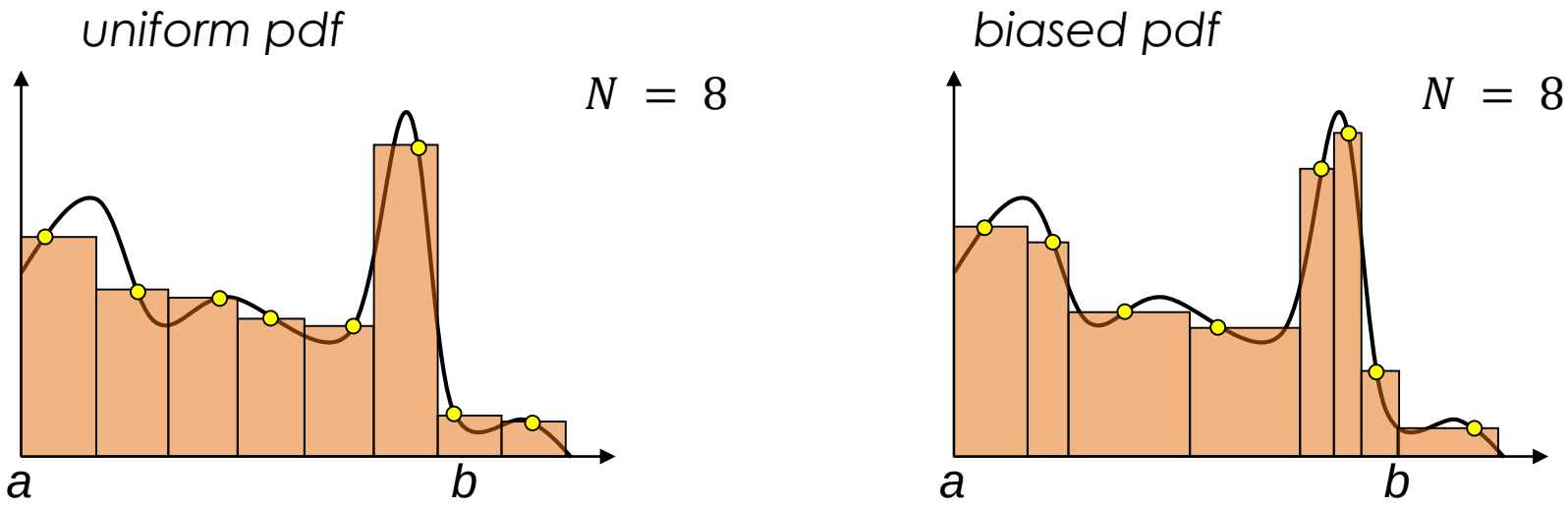
$$E[\langle I \rangle] = E[\frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}] = \frac{1}{N}\sum_{i=1}^{N}E[\frac{f(x)}{p(x)}] =$$

$$\frac{1}{N}\sum_{i=1}^{N}\int_a^b \frac{f(x)}{p(x)}\,p(x)dx = \frac{1}{N}\cdot N\cdot\int_a^b f(x)dx =$$

$$= \int_a^b f(x)dx = I$$

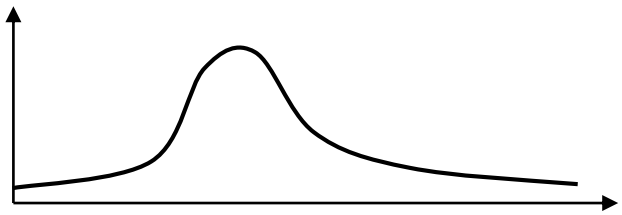
Importance Sampling (3)

- Why use non-uniform pdf?
  - Can significantly reduce variance (for the same N)
- Example:

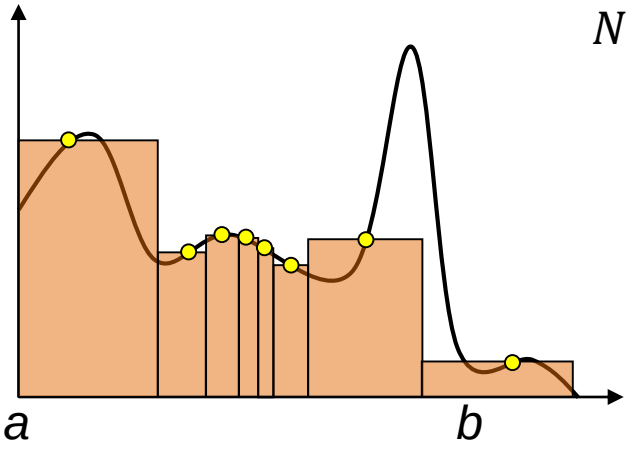


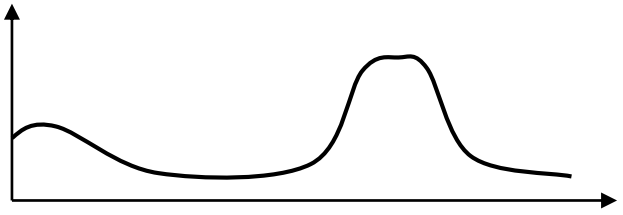
Importance Sampling (4)

- The selection of a “good” pdf is crucial to the effectiveness of importance sampling
- Example:

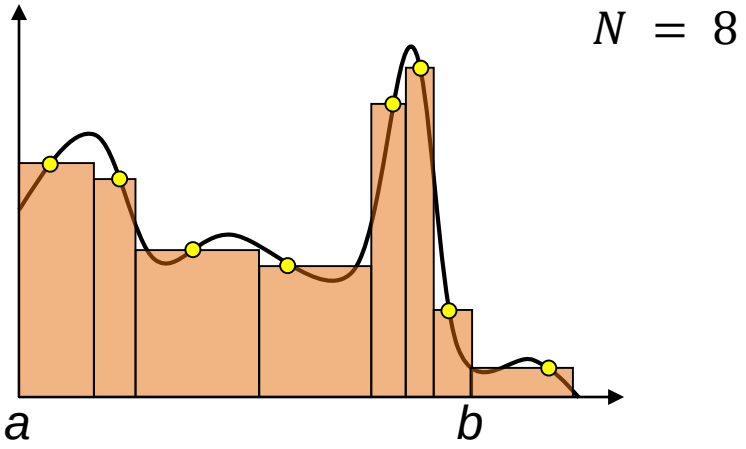


bad pdf





good pdf



Importance Sampling (5)

- What makes a pdf “good”?
  - Minimize variance →
  - Use less samples to achieve the same result in terms of quality

- What is an optimal pdf?
  - It has been shown that optimal samples are chosen from a distribution where $p(x) \propto f(x)$

Importance Sampling (6)

Ok… well… but, how can I know $f(x)$?

- We usually don't, but:
- Can have hints about good sampling directions
- Can estimate $f(x)$ with a few samples using a known distribution (e.g. uniform) and then construct a distribution proportional to $f(x)$ to continue drawing better samples

Importance Sampling (7)

- For multi-dimensional integrals of *M*-D functions:

$$I = \int_{s1}^{t1} \int_{s2}^{t2} \cdots \int_{sM}^{tM} f(x_1, x_2, ..., x_M) dx_M ... dx_2 dx_1,$$

$$\langle I \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_1, x_2, ..., x_M)}{p(x_1, x_2, ..., x_M)}$$

Sample Generation (1)

- To draw samples distributed according to $p(x)$:
  - Compute the cumulative distribution function $P(x)$:

$$P(x) = \int_a^x p(y)dy$$

$P(x)$ is a monotonic increasing function over $[a,b]$

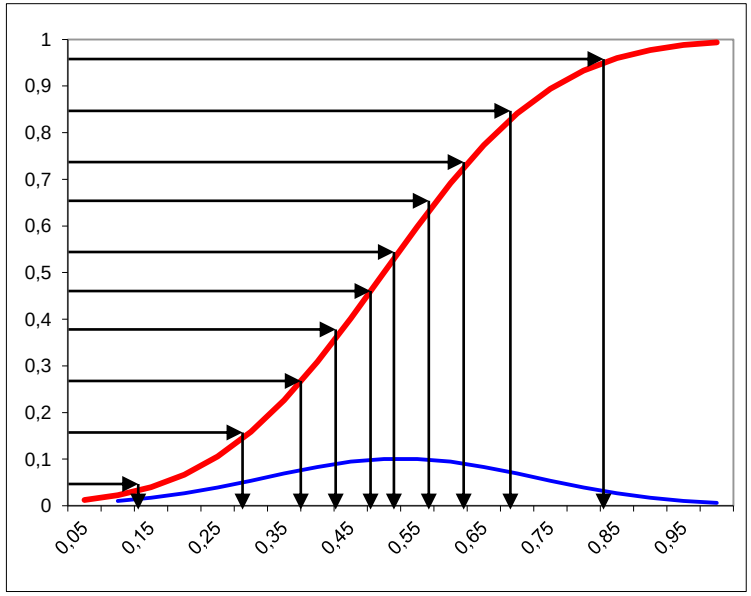
$P(a) = 0$ and $P(b) = 1$

  - If a random number t is generated uniformly over $[0,1]$, then

$x = P^{-1}(t)$ is distributed according to $p(x)$

Sample Generation (2)

- In practice, given a uniform sample generator, to compute the inverse value quickly:
  - Compute samples of $g(x) \propto f(x)$ and normalize over sampling domain
  - Compute CDF $P(x)$ and store a mapping $P(x) \rightarrow x$
  - Generate uniform samples (in the CDF domain)
  - Find closest values of stored $P(x)$ and interpolate the respective x to obtain a sample in the integration domain
- Or, of course, find the above analytically (inversion method)
  - Particularly useful when drawing samples from known distributions using uniformly distributed random numbers



STOCHASTIC PATH TRACING

Path Tracing Principles

- This is a direct approximation of the rendering equation by Monte Carlo integration
  - The integrand is unknown. It depends on the radiance coming from a direction ω_i (equivalently, a visible point $\mathbf{y}$):
  - Recursive evaluation!

- We need to efficiently sample the domain, spending more samples where they count!
- We need a termination mechanism for the recursion

Rendering Equation and MC Integration

- The rendering equation

$$L_o(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + \int_{\Omega} L(\mathbf{x}, \omega_i) f_s\left(\mathbf{x}, \varphi_o, \theta_{o,} \varphi_i, \theta_i\right) d\sigma_{\perp}(\omega_i)$$

- Is approximated by:

$$L_o(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + \frac{1}{N} \sum_{i=1}^{N} \frac{L(\mathbf{x}, \omega_i) f_s\left(\mathbf{x}, \omega_i, \omega_o\right) |\cos\theta_i|}{p(\omega_i)}$$

- Where $L(\mathbf{x}, \omega_i)$ is recovered by tracing a new ray towards ω_i

Path Tracing - Observations

- Using the MC rendering equation integral above we can trace truly random paths and
- Given enough samples, we can estimate the illumination in the scene
- But… how many paths actually reach light emitters?
  - Unfortunately, usually way too few!
  - Most of the rays spawned randomly contribute nothing!
  - We need to fix this…

Sampling the Lights (1)

- We know that light emitters (light sources) are the only source of direct illumination
- We should always attempt to sample them, at each gathering operation (rendering equation evaluation)
- Let's try this out:

$$L_o(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + \int_{\Omega} L(\mathbf{x}, \omega_i) f_s(\mathbf{x}, \omega_i, \omega_o) d\sigma_{\perp}(\omega_i)$$

Can be expressed as a light transport operator:

$$L_o = L_e + T_{\Omega} L_i$$

Sampling the Lights (2)

$$L_o = L_e + T_\Omega L_i = L_e + T_{\Omega_\mathrm{L}} L_i + T_{\Omega - \Omega_\mathrm{L}} L_i$$

where Ω_L is the solid angle subtended by contributing light sources and $\Omega - \Omega_\mathrm{L}$ the rest of the domain

Now, applying this recursively:

$$L_o = L_e + T_{\Omega_\mathrm{L}} L_i + T_{\Omega - \Omega_\mathrm{L}} L_i =$$

$$L_e + T_{\Omega_\mathrm{L}} (L_{e_L} + T'_\Omega L_i{}') + T_{\Omega - \Omega_\mathrm{L}} (0 + T'_\Omega L_i{}') =$$

$$L_e + T_{\Omega_\mathrm{L}} L_{e_L} + T_\Omega \; T'_\Omega L_i{}'$$

Sampling the Lights (3)

$$L_o = L_e + T_{\Omega_L} L_{e_L} + T_\Omega \; T'_\Omega L_i{}'$$

And considering that for primary rays, light sources are directly sampled by the measurement equation:

$$L_o = \underbrace{T_{\Omega_L} L_{e_L}}_{\text{Direct}} + \underbrace{T_\Omega \; T'_\Omega L_i{}'}_{\text{Indirect}}$$

This means that we sample the light sources separately and indirect light gathering does not account for emittance (*) to avoid overshooting

* With the exception of truly specular events (S: $f_s \to \infty$), where both contributions are obtained by a deterministic ray

Sampling the Lights (4)

Moving back to the integral form, $L_o = T_{\Omega_L} L_{e_L} + T_\Omega \; T'_\Omega L_i{}'$ becomes:

$$L_o(\mathbf{x}, \omega_o) = L_{Direct} + L_{Indirect} =$$

$$\int_{\Omega_L} L_e(\mathbf{x}, \omega_i) f_s(\mathbf{x}, \omega_i, \omega_o) d\sigma_\perp(\omega_i) +$$

$$\int_{\Omega} L_i(\mathbf{x}, \omega_i) f_s(\mathbf{x}, \omega_i, \omega_o) d\sigma_\perp(\omega_i)$$

Sampling the Lights (4)

Since, for light sources we directly sample their (known) surfaces, we switch to the area integral form :

$$L_o(\mathbf{x}, \omega_o) = \int_{S_L} L_e(\mathbf{y}, \omega_i) f_s(\mathbf{x}, \omega_i, \omega_o) G(\mathbf{x}, \mathbf{y}) dA(\mathbf{y}) +$$

$$\int_{\Omega} L_i(\mathbf{x}, \omega_i) f_s(\mathbf{x}, \omega_i, \omega_o) d\sigma_{\perp}(\omega_i)$$

So we typically combine two integral forms

Direct Illumination Estimation (1)

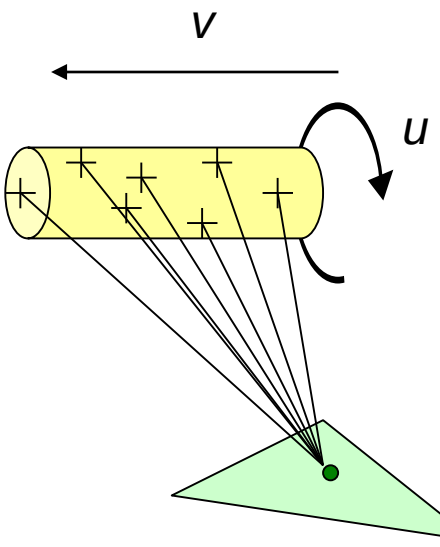
- For L_{Direct}, we use MC sampling on the surface of light sources:

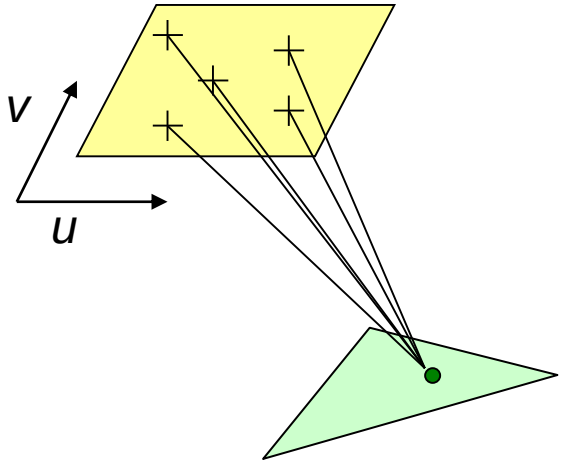
$$\langle L_{Direct}(\mathbf{x}, \omega_o) \rangle = \frac{1}{N_L} \sum_{i=1}^{N_L} \frac{L_e(\mathbf{y}_i, \omega_i) f_s(\mathbf{x}, \omega_i, \omega_o) V(\mathbf{x}, \mathbf{y}_i) G(\mathbf{x}, \mathbf{y}_i)}{p(\mathbf{y}_i)}$$

- For a single light:
  - We must determine N_L samples over its surface with a probability $p(\mathbf{y}_i)$
  - Then test for visibility $V(\mathbf{x}, \mathbf{y}_i)$ (0 if obscured)
  - Finally, gather its contribution with the above formula

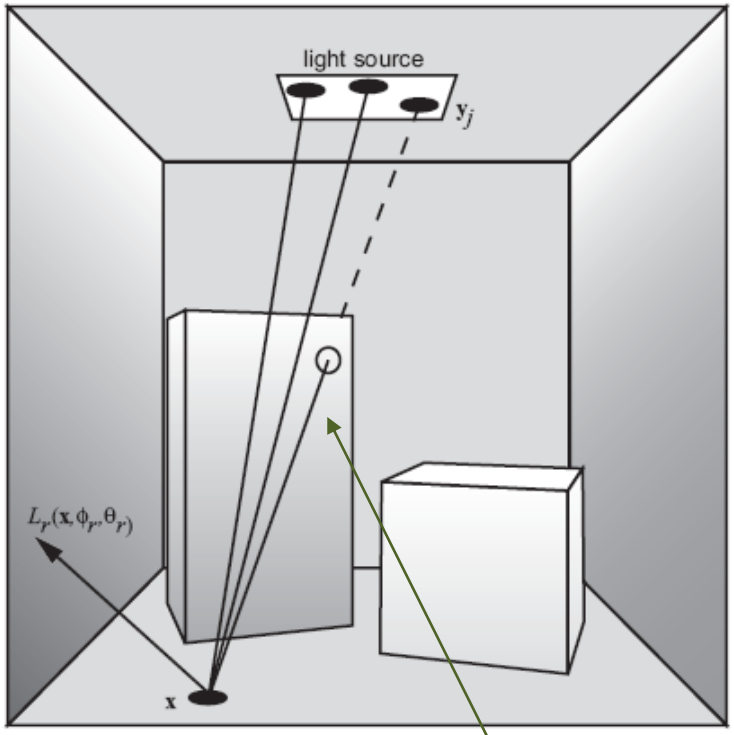
Direct Illumination Estimation (2)

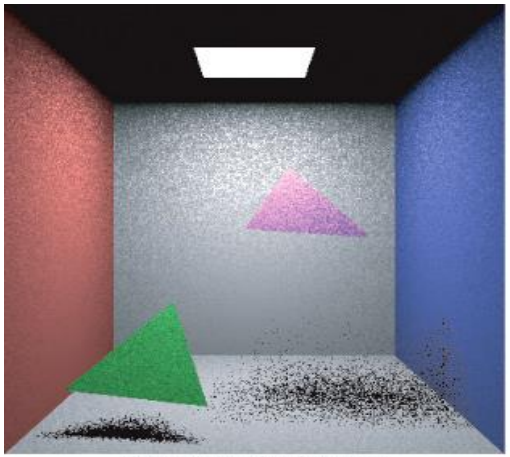
- Pdf $p(\mathbf{y})$:
  - generates surface points $\mathbf{y}_i$ over the total light source area
  - Is a 2D pdf (2 coordinates u and v)
  - (u,v) pair is transformed to a 3D point on the light surface with a proper mapping

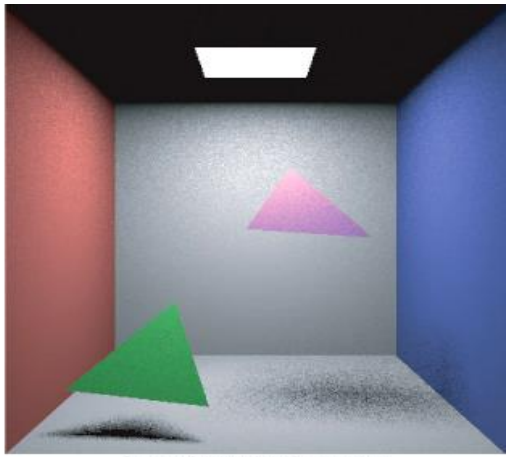


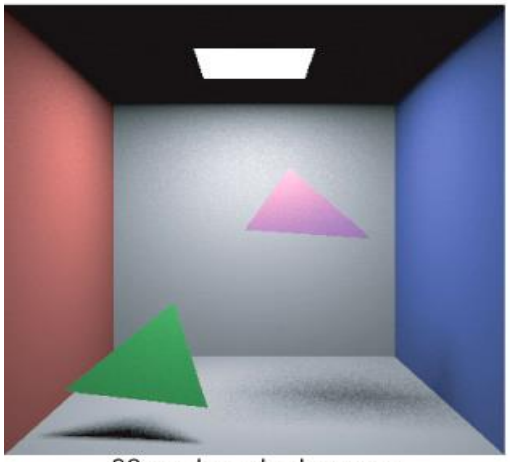


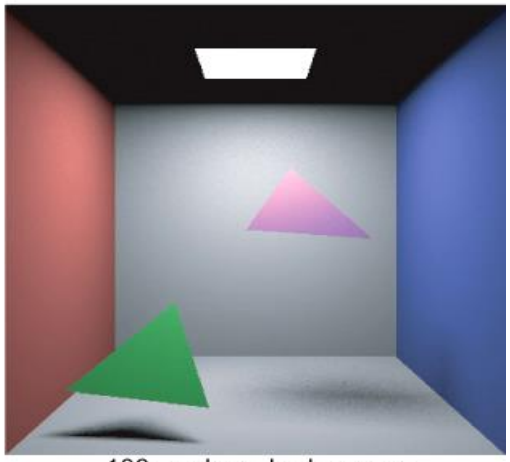
Direct Illumination Estimation (3)




1 random shadow ray


9 random shadow rays


36 random shadow rays


100 random shadow rays

visibility test (failed here)

Light Sampling – Noise (1)

- The visibility function $V(\mathbf{x}, \mathbf{y}_i)$:
  - $V(\mathbf{x}, \mathbf{y}_i) = 1$ $\mathbf{y}_i \rightarrow$ light source fully visible to $\mathbf{x}$
  - $V(\mathbf{x}, \mathbf{y}_i) = 0$ $\mathbf{y}_i \rightarrow$ light source fully occluded

- Large light sources require larger N_L to adequately sample the penumbrae for a smooth soft shadow

Light Sampling – Noise (2)

- The geometric coupling term $G(\mathbf{x}, \mathbf{y}_i)$:
  - For points $\mathbf{x}$ close to large emitters:

    $1/\ r_{\mathbf{x}\mathbf{y}j}$ takes on arbitrary large values for light source samples very close to the receiver

    (instability) → Results to very bright pixels

Multiple light sources

Two strategies:

- Separate light source sampling:
  - Can use a different number of samples per light, e.g. according to power or size
  - At least one sample per light!
- Combined light sources
  - Combined Monte Carlo integration domain
  - Can even shoot a single ray for all lights combined!
  - Samples can be statistically biased toward certain sources

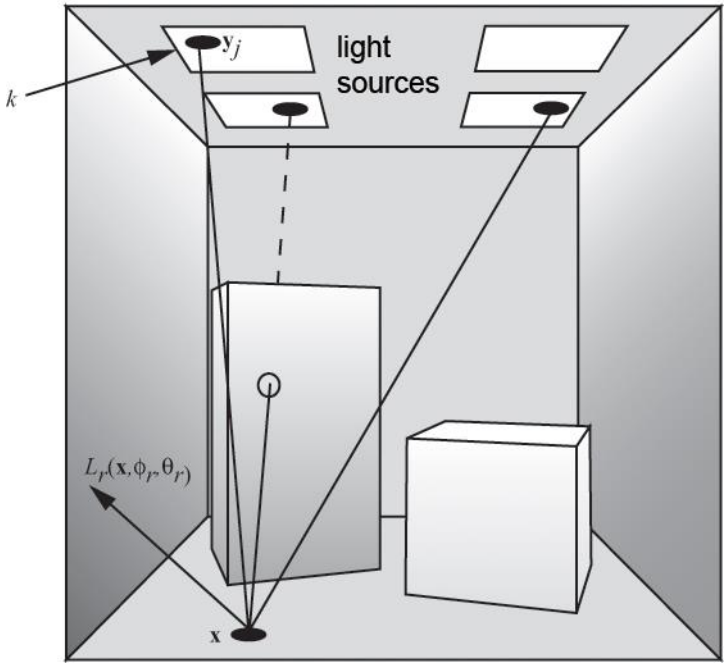
Multiple light sources – Combined (1)

- Assign each light a probability value for it being chosen
- It is a two-stage approach:
  - A discrete pdf $p_L(k)$ generates a randomly selected light source k_i among N_L lights
  - A surface point $\mathbf{y}_i$,on the selected light source k is selected using a conditional pdf $p(\mathbf{y}|k_i)$
- Combined estimator:

$$\langle L_{Direct}(\mathbf{x}, \omega_o) \rangle = \frac{1}{N_L} \sum_{i=1}^{N_L} \frac{L_e(\mathbf{y}_i, \omega_i) f_s(\mathbf{x}, \omega_i, \omega_o) V(\mathbf{x}, \mathbf{y}_i) G(\mathbf{x}, \mathbf{y}_i)}{p(k_i) p(\mathbf{y}_i|k_i)}$$

Multiple light sources – Combined (2)

- Any valid $p_L(k)$ and $p(\mathbf{y}|k)$ pdfs → unbiased results
- pdfs affect the variance of the estimators
- Most common pdfs:
  - Uniform source selection + uniform area sampling
  - Power-proportional source selection + uniform sampling



Multiple light sources – Combined (3)

Drawbacks:

- We must be careful not to exclude any lights that contribute to $\langle L_{Direct}(\mathbf{x}, \omega_o) \rangle$
- 3 random numbers are needed to generate a shadow ray:
  - One to select the light source k
  - 2 to select a specific surface point

Truly Random Paths – Complexity

- At each hit point, the rendering equation is evaluated using a number of direct and indirect samples
- For indirect samples $N > 1$, the rendering time grows exponentially w.r.t the recursion depth $O(N^{depth})$
- With always 1 indirect sample:
  - $O(depth)$
  - We trace a single path for each pixel sample
  - Slower convergence but faster feedback
  - Typically hundreds to thousands of paths are traced

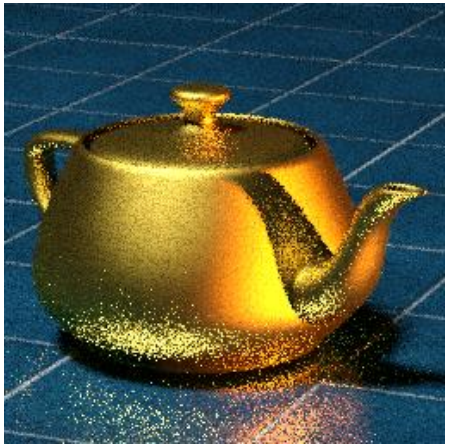
AUEB COMPUTER GRAPHICS GROUP

Convergence

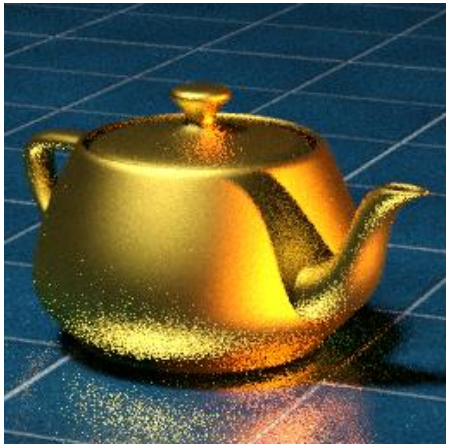
1 sample

4 samples

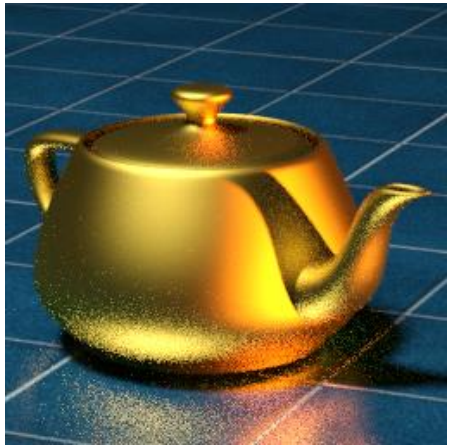
16 samples



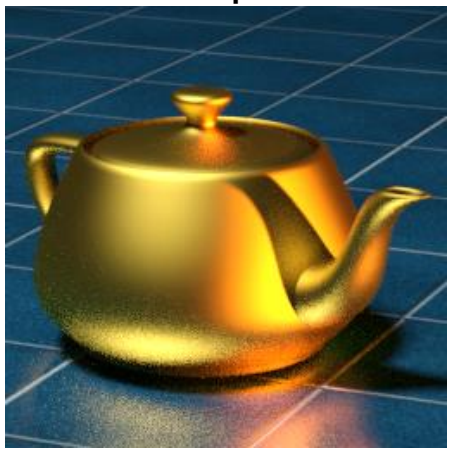
64 samples



256 samples

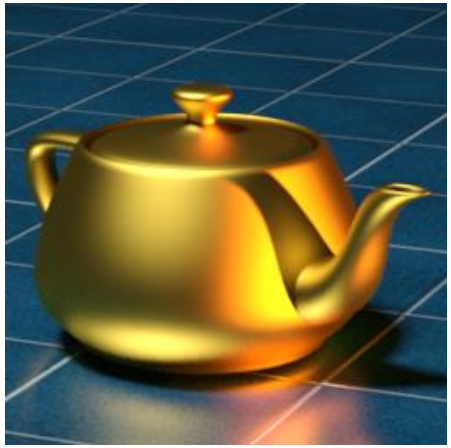


1024 samples



4096 samples

16384 samples



Truly Random Paths – Termination (1)

- Any ray tracing algorithm needs a stopping condition
- Otherwise:
  - the generated paths could be of infinite length
  - the algorithm would not come to a halt
- But finite length → introduces bias to the final image (disregards possibly important segments)

- Have to find a way to:
  - limit the length of the paths
  - still be able to obtain a correct solution

Truly Random Paths – Termination (2)

- Classic ray-tracing uses 2 techniques:

(a) Fixed depth:

- cuts off the recursion after a fixed number of evaluations
- upper bound on the amount of rays that need to be traced
- May ignore important light transport → biased image
- Manual setting of depth according to type of materials

Truly Random Paths – Termination (3)

(b) Adaptive recursion depth:

- The cumulative ray "strength" of the path so far is maintained. Terminates when strength too low
- more efficient technique
- Can still omit highly contributing sub-paths (intense lights) → bias

Russian Roulette Ray Generation

- Addresses the problem of keeping the lengths and number of the paths manageable
- Leaves room for exploring all possible paths of any length
- Produces an unbiased image by:
  - Applying a hit/miss strategy to replace the probabilistic bias
  - Not indiscriminately eliminating unimportant samples but doing so with a probability ξ.

Russian Roulette Principle (1)

- Russian Roulette sampling is a very important tool in rendering
- The idea is that an estimator F can be replaced by:

$$F' = \begin{cases} \frac{1}{p}F, with\ probability\ p \\ 0, \qquad\qquad otherwise \end{cases}$$

- In simple terms, if for N trials, (1-p)N times we choose to discard the solution, the pN trials must be boosted to counter the loss

Russian Roulette Principle (2)

Example:

- Let at a particular light bounce the probability of rejecting a ray be 3/4
- If we do nothing, this bounce will contribute 3/4 less light than normal, because we are deliberately zeroing its contribution 3 out 4 times
- To counter this, the radiance returned by the 1 ray actually cast out of 4 attempts, must be quadrupled

Russian Roulette Principle (3)

Why Russian Roulette works?

- $E[<I_{RR}>]$ evaluates to I
- Missed samples in the integral evaluation are compensated by scaling the incoming radiance by $1/p$
- If 1-p (absorption) is small:
  - the recursion will continue many times
  - the final estimator will be more accurate
- If 1-p is large:
  - the recursion will stop sooner
  - the estimator will have a higher variance

Practical Russian Roulette Implementation

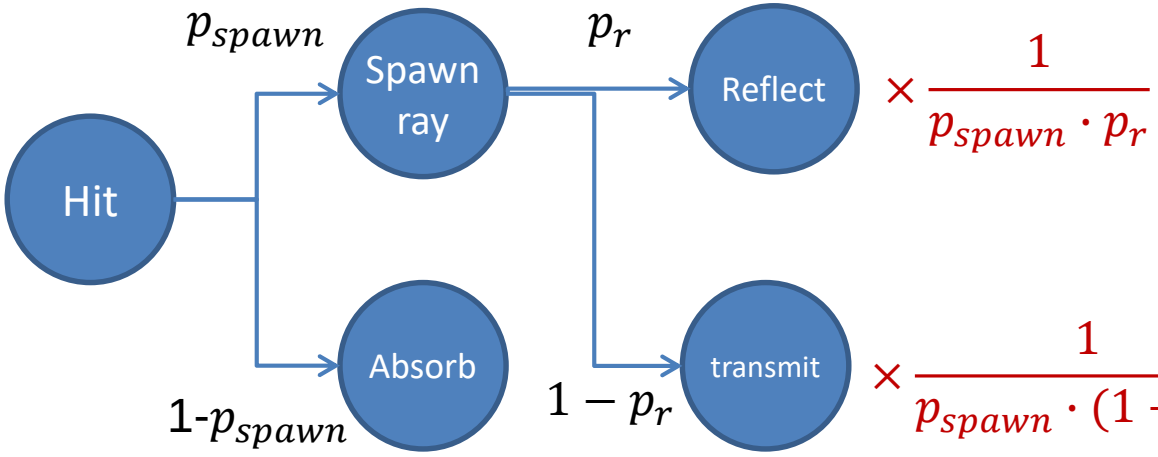
- Choose a reasonable probability p to cast a ray, often related to:
  - Max {reflectivity , transmission coef}
  - Total path contribution up to this event
  - Recursion depth (i.e. bounce)
  - …
- Generate a uniform random number ξ
- If ξ < p, cast the ray and weight the result by 1/p
- Otherwise, don’t spawn a ray

Sampling (1)

- At each ray hit, the following questions must be answered:
  - How many rays to spawn? (when forming a single path, 1 or 0: RR termination)
  - Which direction should the ray take?
  - What distribution to use?

Sampling (2)

- Typically, paths are distributed differently under reflection and transmission
- In a single path implementation, we must choose either transmission or reflection
- We can use RR to determine the above!



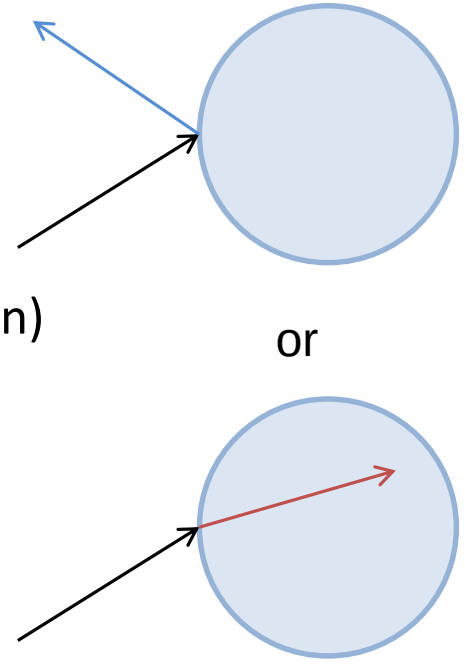
$p_{spawn} = transmitted\ importance$

$p_r = F\ (Fresnel\ term)$

Sampling the BSDF (1)

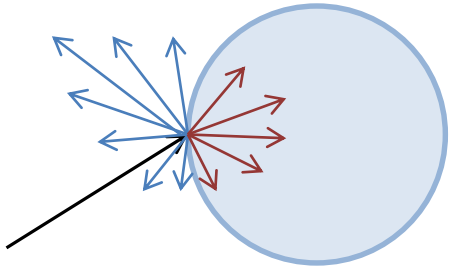
- Depending on the material attributes (roughness, reflectance, shader, etc) and the event chosen (reflection, transmission), we can determine the next path direction using the appropriate directional sampling distribution:

- Ideally specular - BTDF or BRDF $\rightarrow \infty$:
  - Determine ray direction analytically
  - Do not sample the lights separately
    (0 contribution outside ideal scattering direction)
  - Measure both L_e and GI at hit location
  - Use probability $p(\omega_i) = 1$

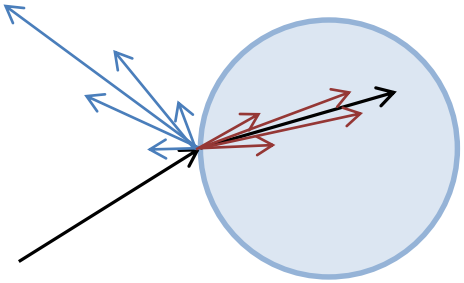


Sampling the BSDF (2)

- Low gloss / high out-scatter – large BRDF / BTDF spread:
  - Use hemisphere cosine sampler
  - Use $p(\omega_i) = {\cos\theta_i}/{\pi}$



- High gloss / tight focus:
  - Use microfacet pdf of local shading model to generate incident direction



Multiple Importance Sampling (1)

- We have seen how to sample directions towards the lights and use them either as a separate direct lighting component or as the direction of the next path segment
- We have used BRDF-based sampling to give preference to the scattering direction
- However, the pdf we chose only conforms to one of the terms in the rendering equation integrant, therefore the distribution does not necessarily match the integrant!

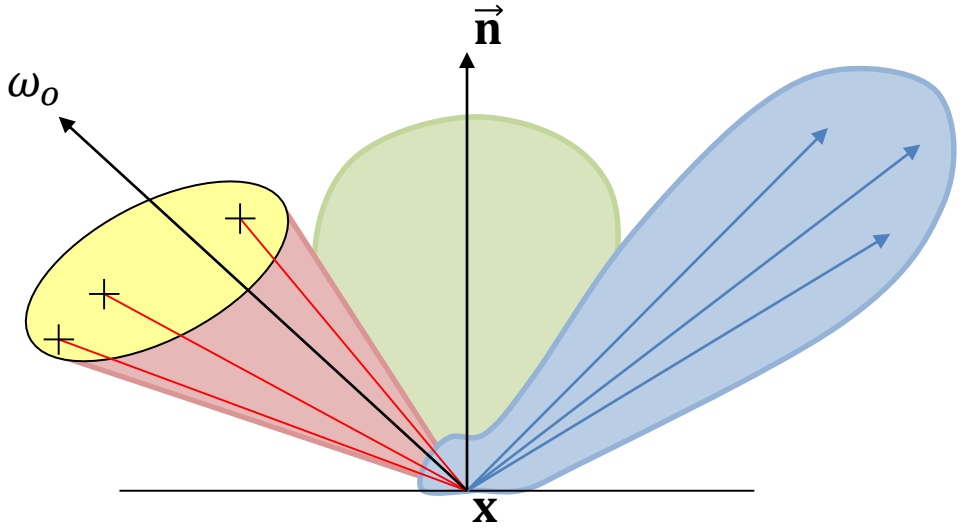
  → Can introduce serious variance

Multiple Importance Sampling (2)

However, the pdf we chose only conforms to one of the terms in the rendering equation integrant, therefore the distribution does not necessarily match the integrant!

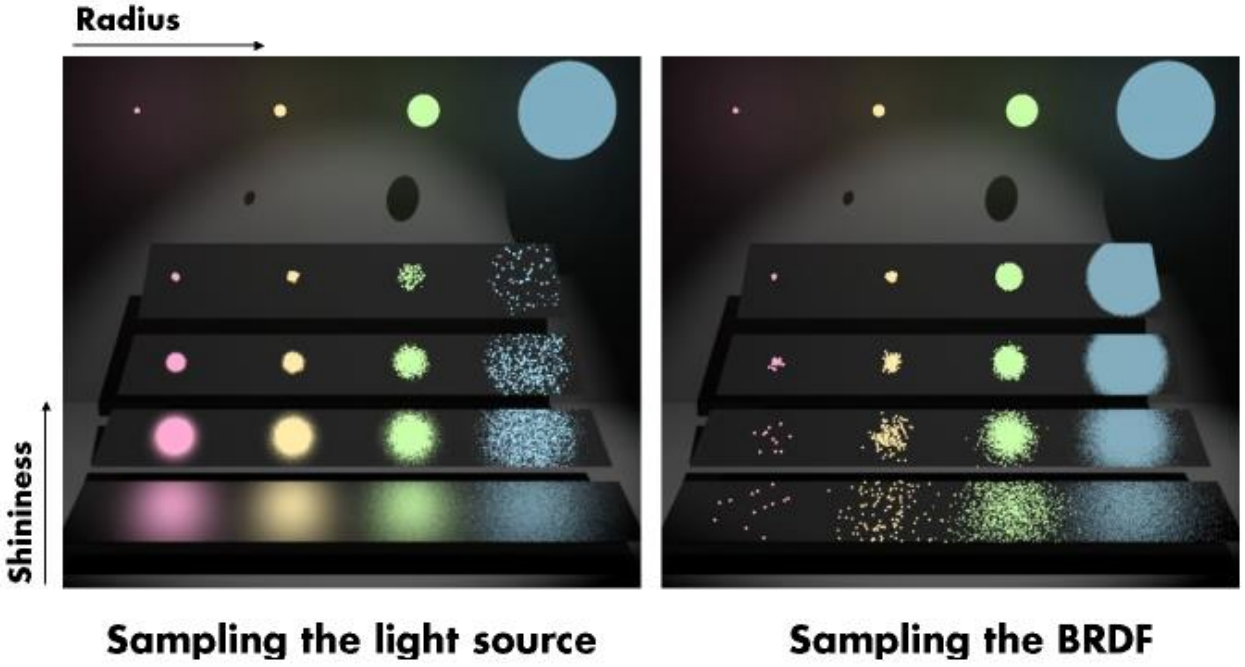
→ Can introduce serious variance

$$L_o(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + \frac{1}{N}\sum_{i=1}^{N} \frac{L(\mathbf{x}, \omega_i) f_s\,(\mathbf{x}, \omega_i, \omega_o\,)|\cos\theta_i|}{p(\omega_i)}$$



Multiple Importance Sampling (3)

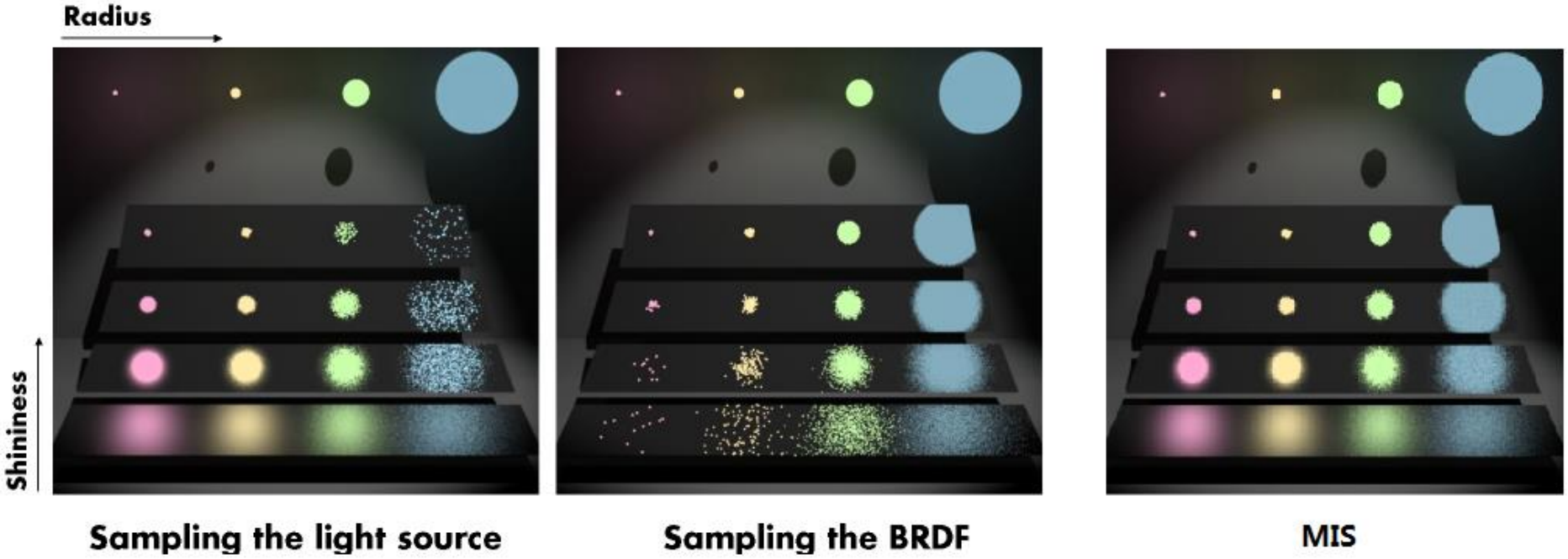
- Sampling the light sources for highly glossy BRDFs leads to zero-valued evaluations of the latter and only erratic contribution of light
- Sampling rough BRDF for lights subtended through very small solid angles has the same effect



Multiple Importance Sampling (3)

Solution:

- Sample directions using both strategies simultaneously! → MIS



Multiple Importance Sampling (4)

- Draw samples from all distributions
  - Can use different number of samples per distribution
  - Can combine any sampling strategy. Here: cosine-weighted, BRDF sampling and light domain sampling
- Weight their relative contribution based on the quality of the samples drawn (pdf value):

Number of samples drawn from each distribution

Weight per sample per distribution

Number of distributions

MC sampling using i-th distribution

$$F = \sum_{i=1}^{N} \frac{1}{N_i} \sum_{j=1}^{N_i} w_i(X_{i,j}) \frac{f(X_{i,j})}{p(X_{i,j})}$$

Multiple Importance Sampling (5)

- Arithmetic average is maybe the worst weighting function: Additively blends two or more possibly bad distributions → Always increases variance
- The Balance heuristic: $w_i(x) = \frac{N_i p_i(x)}{\sum_k N_k p_k(x)}$
  - It is almost optimal
  - $\sum_k w_k(x)$=1
- Each sample generator contributes to the result proportionally to its pdf

Multiple Importance Sampling (6)

Practical example:

- Draw one sample ω_L towards the light source using the area sampling approach with pdf $p_L(\omega_L)$
- Draw one sample ω_f using BRDF sampling with pdf $p_f(\omega_f)$
- Evaluate the MC estimator for each one:

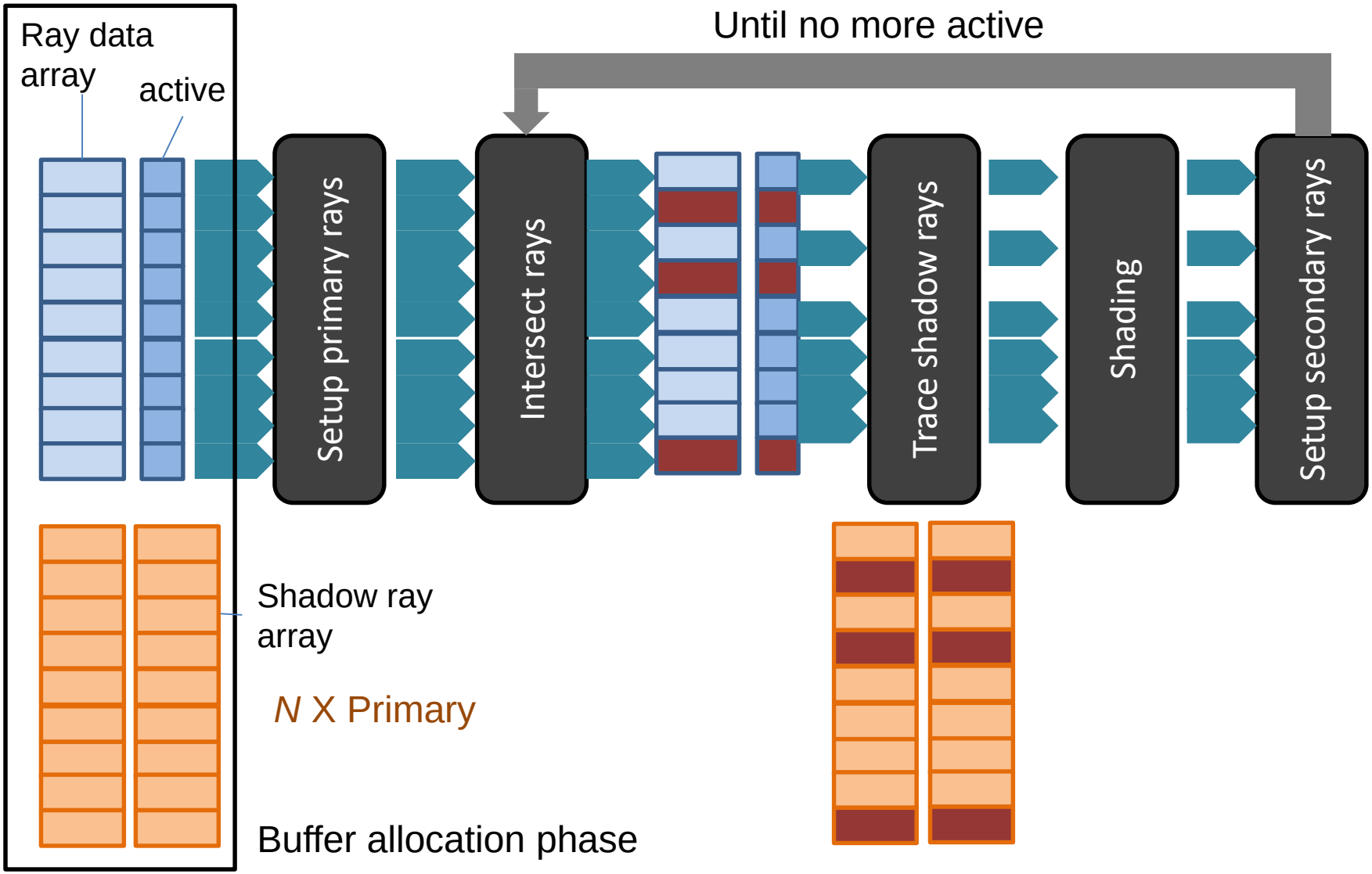
$$I_L = \frac{f_r(\omega_L,\omega_o)L(\omega_L)|cos\theta_L|}{p(\omega_L)}, I_f = \frac{f_r(\omega_f,\omega_o)L(\omega_f)|cos\theta_f|}{p(\omega_f)}$$

- Estimate the weights:
- $w_L(\omega_L) = \frac{p_L(\omega_L)}{p_L(\omega_L)+p_f(\omega_L)}, w_f(\omega_f) = \frac{p_f(\omega_f)}{p_L(\omega_f)+p_f(\omega_f)}$
- Evaluate the rendering equation:
- $\langle I \rangle = w_L(\omega_L)I_L + w_f(\omega_f)I_f = \frac{f_r(\omega_L,\omega_o)L(\omega_L)|cos\theta_L|}{p_L(\omega_L)+p_f(\omega_L)} + \frac{f_r(\omega_f,\omega_o)L(\omega_f)|cos\theta_f|}{p_L(\omega_f)+p_f(\omega_f)}$

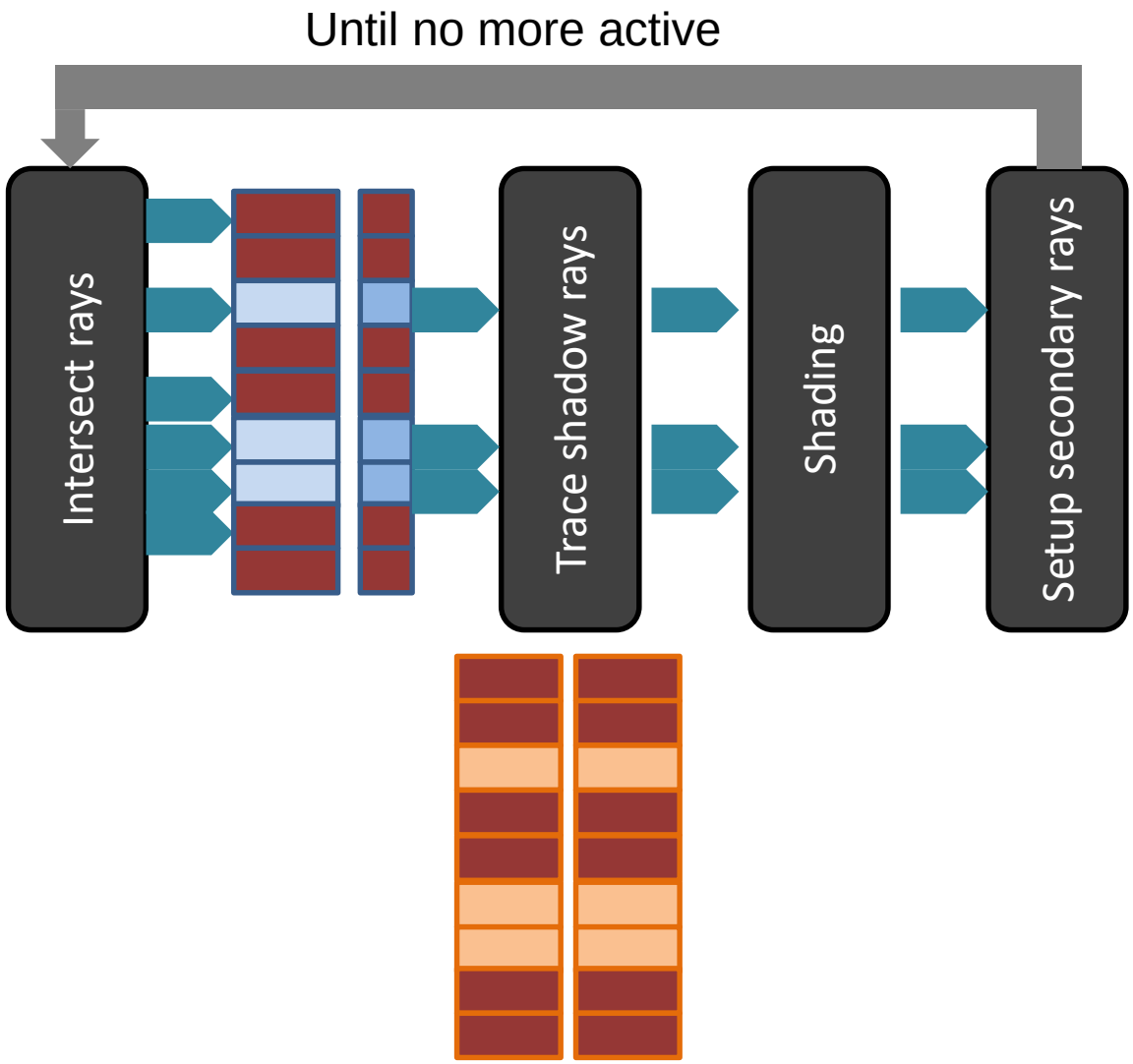
Efficient Implementation (1)

- Recursive implementation is not optimally implemented:
  - Excessive call stack thrashing
  - Incoherent rays
  - Bad resource pooling & cache coherence
- Use a ray-parallel iterative implementation
  - Allocate once and reuse ray storage
  - No recursive function call
  - Easy vectorization / threaded execution (also in GPUs)

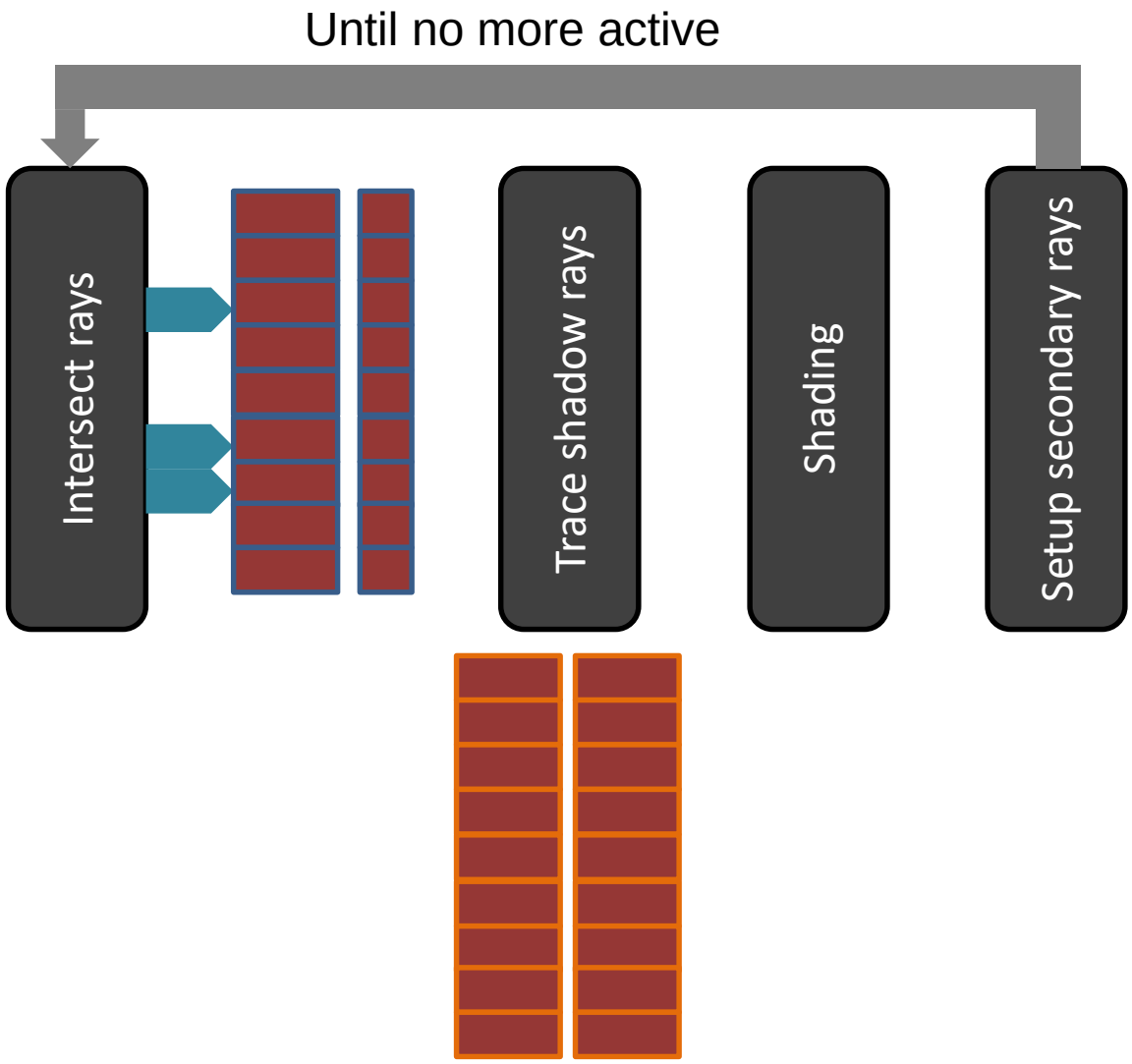
Efficient Implementation (2)



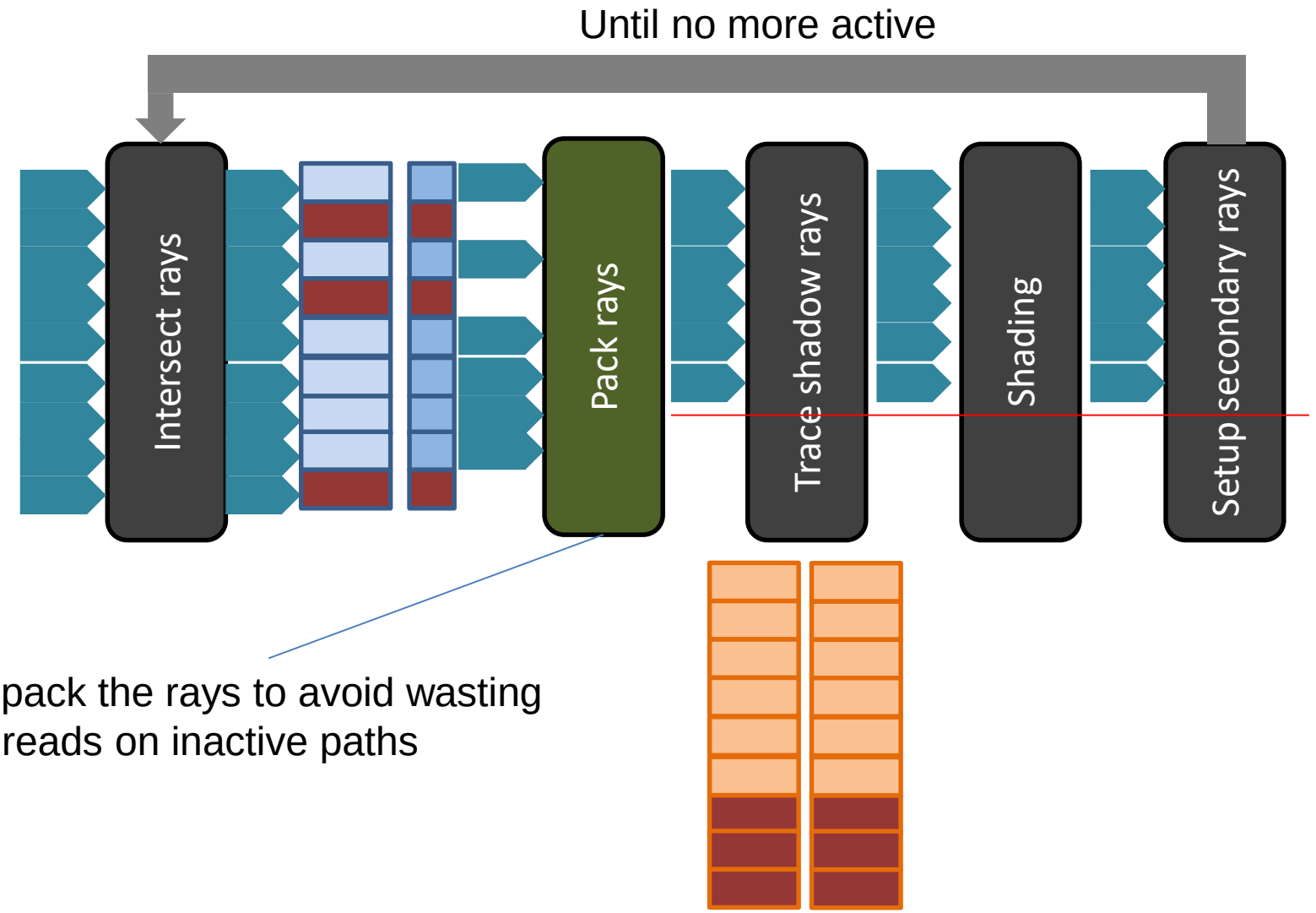
Efficient Implementation (3)



Efficient Implementation (4)



Efficient Implementation (5)



You can pack the rays to avoid wasting cycles/threads on inactive paths

Efficient Implementation (6)

- Notes:
  - You can combine MIS with RR to only evaluate one sampling direction at a time, even with multiple samplers
  - In the light vs BRDF sampler example:

$$\langle I \rangle = w_L(\omega_L) I_L + w_f(\omega_f) I_f = \frac{f_r(\omega_L,\omega_o) L(\omega_L) |cos\theta_L|}{p_L(\omega_L)+p_f(\omega_L)} + \frac{f_r(\omega_f,\omega_o) L(\omega_f) |cos\theta_f|}{p_L(\omega_f)+p_f(\omega_f)}$$

We can choose I_L or I_f using RR with $p_{SelectLight}$:

$$\langle I \rangle = \begin{cases} w_L(\omega_L) I_L / p_{SelectLight} \\ w_f(\omega_f) I_f / (1 - p_{SelectLight}) \end{cases}$$

Efficient Implementation (6)

- Pros:
  - Well-behaved threaded execution, with predictable loads and load balancing
  - Tiling efficiently boosts parallel unit utilization
  - No recursion
  - Simple, generic and well-structured code for next segment sampling
- Cons:
  - Increased variance: Every event spawns only one ray
  - Paths with multiple samples along the segments very tricky to handle (volumetric scattering): must handle samples as separate events

AUEB COMPUTER GRAPHICS GROUP

Contributors

- Georgios Papaioannou