CPSC 591/691: Rendering, Fall 2023

UNIVERSITY OF CALGARY

A Real-Time Path Tracing Rendering Engine Using the ReSTIR Algorithm

Travis Dow

1. Introduction

- Monte Carlo path tracing has become widely used for offline rendering and is starting to be used for real-time applications.
- Real-time rendering presents many challenges, with limited computation times, dynamic scenes, and user inputs
    - 30 FPS = 33.33ms / frame; 60 FPS = 16.67ms /frame.
- These constraints have spurred research into hardware accelerated ray tracing, improved sampling techniques, and real-time denoising algorithms to improve image fidelity.

2. Background

- The goal of any photo-realistic renderer is to solve the rendering equation:

Geometry term for attenuation and cosine between points x and y.

Integrated over all light-emitting surfaces A composed of points x.

$$L(y,\omega) = \int_A \rho(y, \overrightarrow{yx} \leftrightarrow \vec{\omega})\, L_e(x \rightarrow y)\, G(x \leftrightarrow y)\, V(x \leftrightarrow y)\, \mathrm{d}A_x, \qquad (1)$$

Outgoing radiance at point y in direction ω.

BSDF ρ at point y for light coming from x and direction ω.

Emitted radiance from point x to point y.

Mutual visibility term between point x and y.

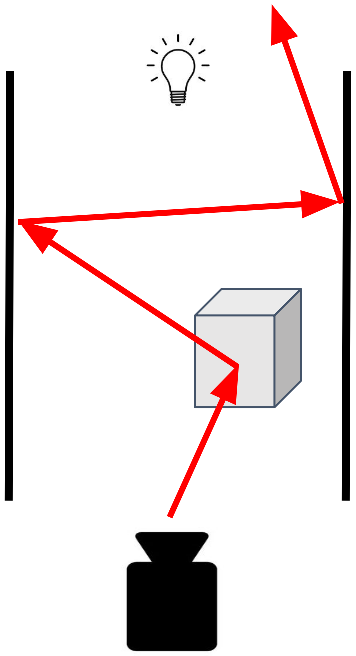
- Truncated: $$\boxed{L = \int_A f(x)\,\mathrm{d}x,} \quad \text{where} \quad f(x) \equiv \rho(x)\, L_e(x)\, G(x)\, V(x). \qquad (2)$$

2. Background

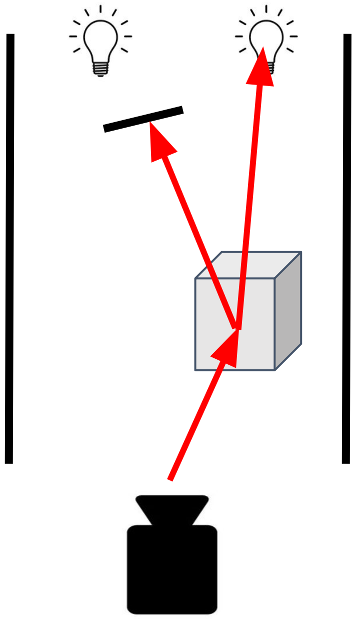
$$L(y,\omega) = \int_A \rho(y, \overrightarrow{yx} \leftrightarrow \vec{\omega})\, L_e(x \rightarrow y)\, G(x \leftrightarrow y)\, \boxed{V(x \leftrightarrow y)}\, \mathrm{d}A_x, \qquad (1)$$

- The Mutual Visibility term V dictates if the current point being rendered, y, can be “seen” by the point on a light source x.
- Tracing rays is a computationally expensive process, one of the downsides of path tracing is that some paths may not contribute to the radiance at all if it never hits an emitting triangle.
- If a light source cannot “see” a point, then we know that point is in shadow relative to that particular light source and that it’s contribution to the total radiance of that point is therefore limited.
- We can leverage this property to improve the efficiency of our path tracing algorithm using what is called Next Event Estimation.

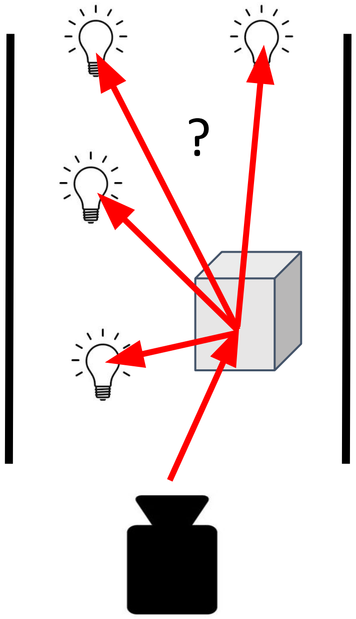
2. Background



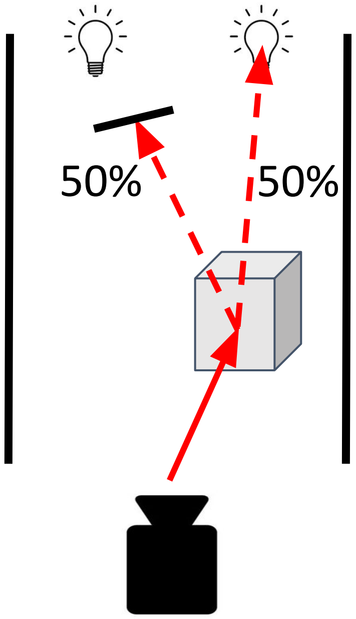
Normal path tracing



Next Event Estimation



Scales poorly



Darkens scene

2. Background

- To improve light sampling we can utilize importance sampling (IS):

$$\langle L \rangle_{\mathrm{is}}^{N} = \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{p(x_i)} \approx L. \quad (3)$$

- Where, p(xi) is a probability density function (PDF) used to describe the “importance” (i.e. probability) of selecting specific light-emitting points x.
- We adjust the sampling rate of lights based on how much they contribute to the final radiance of the fragment we are rendering.

2. Background

$$\boxed{L = \int_A f(x)\,\mathrm{d}x,} \quad \text{where} \quad f(x) \equiv \rho(x)\,L_e(x)\,G(x)\,V(x). \qquad (2)$$

- In a perfect world, p(xi) should be exactly proportional to f(xi).
- However, this is infeasible, because we would need to calculate f(xi) for all light-emitting points x (i.e. the solution to Eq. 2).
- Therefore, we must choose a sub-optimal importance sampling strategy (i.e. **intensity of light being emitted Le**) to generate our PDF.
- This could be improved by utilizing multiple importance sampling (MIS) strategies, where M is the number of such strategies and we simply just combine their effects together linearly:

$$\langle L \rangle_{\text{mis}}^{M,N} = \sum_{s=1}^{M} \frac{1}{N_s} \sum_{i=1}^{N_s} w_s(x_i) \frac{f(x_i)}{p_s(x_i)}. \qquad (4)$$

Here is where it gets weird!

2. Background

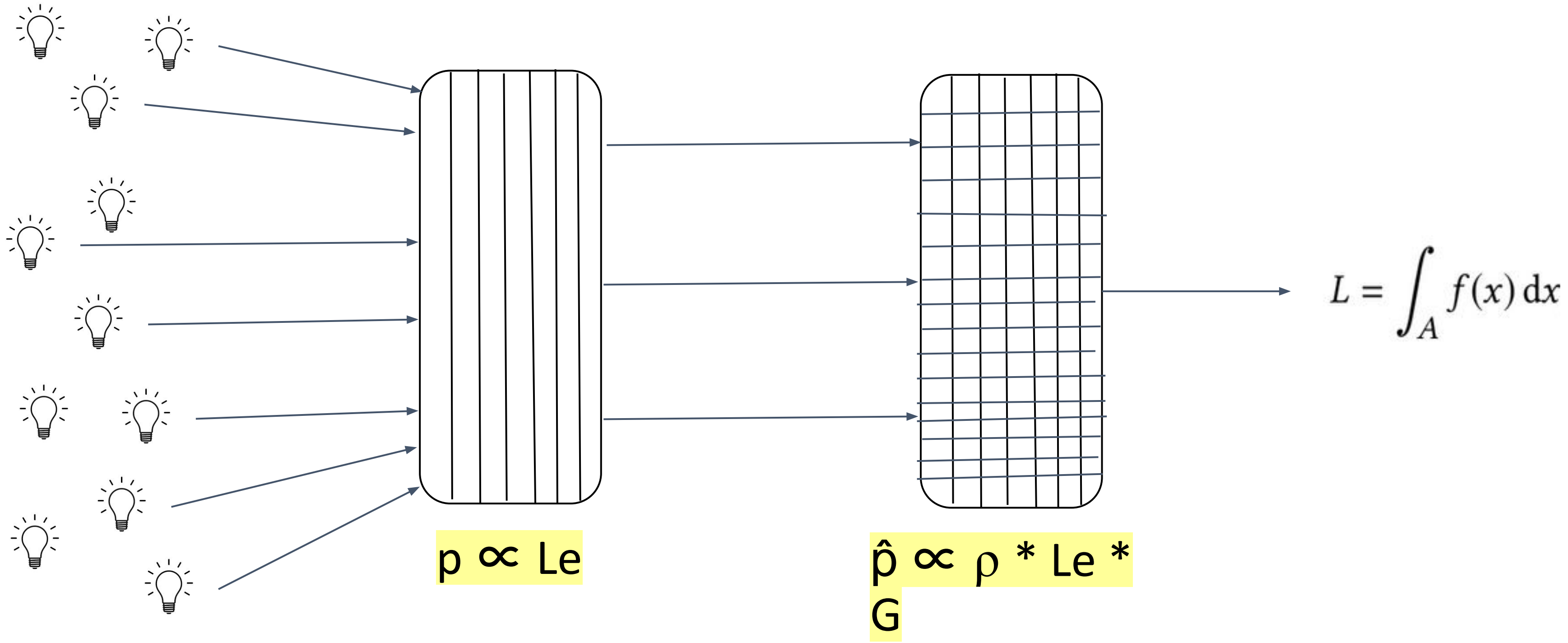
- Alternatively to sampling from a linear combination of terms using MIS, we can instead sample approximately proportional to the product of some of those terms.
- Resampled Importance Sampling (RIS) randomly generates candidate samples from a (sub-optimal) source distribution p and then randomly resamples **1** sample from those candidates with p(z|x):

$$p(z \mid \mathbf{x}) = \frac{\mathrm{w}(x_z)}{\sum_{i=1}^{M} \mathrm{w}(x_i)} \quad \text{with} \quad \mathrm{w}(x) = \frac{\hat{p}(x)}{p(x)}, \qquad (5)$$

- Where, i.e., $p \propto Le$ and $\hat{p} \propto \rho * Le * G$

$$L = \int_A f(x)\,\mathrm{d}x, \quad \text{where} \quad f(x) \equiv \boxed{\rho(x)\,L_e(x)\,G(x)}\,V(x). \qquad (2)$$

2. Background



$$L = \int_A f(x)\,dx$$

Resampled Importance Sampling

2. Background

$$\frac{\hat{p} \propto \rho * Le * G}{p \propto Le} \propto \rho * G$$

- Importantly, because the weight w(x) is the ratio of p̂(x) / p(x), it is self-correcting. That is, when p̂(x) is different than p(x), it will drive w(x) in that direction
- Infact, because we are considering the proportion, p̂(x) need not be an actual PDF, it can strictly be the product of the terms used
  - i.e. $\hat{p}(x) = \rho * Le * G$.
- However, because p̂(x) is an approximation, when we finally calculate the outgoing radiance Lo we must add a correction factor.

$$\langle L \rangle_{\text{ris}}^{1,M} = \frac{f(y)}{\hat{p}(y)} \cdot \left(\frac{1}{M} \sum_{j=1}^{M} \mathrm{w}(x_j) \right). \qquad (6)$$

2. Background

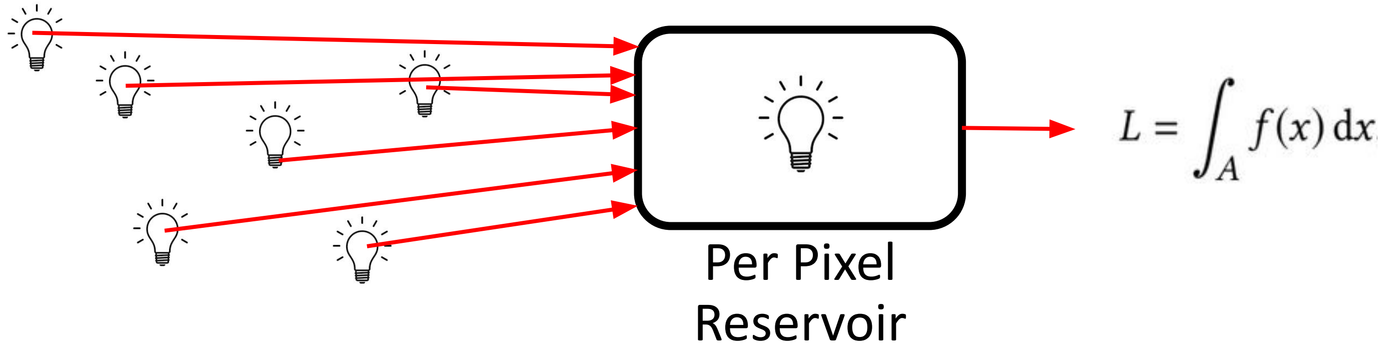
- Weighted Reservoir Sampling (WRS) is an algorithm for sampling N random elements from a “stream” of M elements in a single pass.

$$P_i = \frac{\mathrm{w}(x_i)}{\sum_{j=1}^{M} \mathrm{w}(x_j)}. \qquad (7)$$

- Where, w(xi) is the weight of each element in the stream.
- Importantly, reservoir sampling only needs to process each element once and does not need to know the number of elements, M, in advance.
- Reservoir sampling processes each element in order and keeps a reservoir of samples seen (only **1 reservoir per pixel**).
- Also importantly, reservoirs can be easily combined together.

2. Background

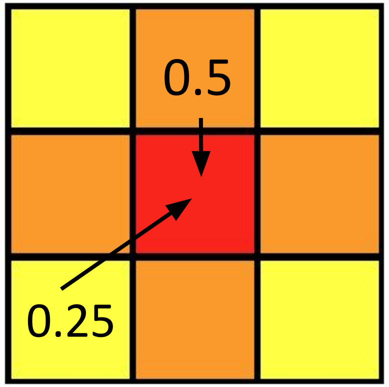
1. Add the weight of the light sample to the reservoir's total sum of weights seen.
2. Increment the number of samples seen
3. Choose a random number n between [0, 1]: if n < (new sample's weight) / (the reservoir's total sum of weights seen), then update the reservoir with this new sample.



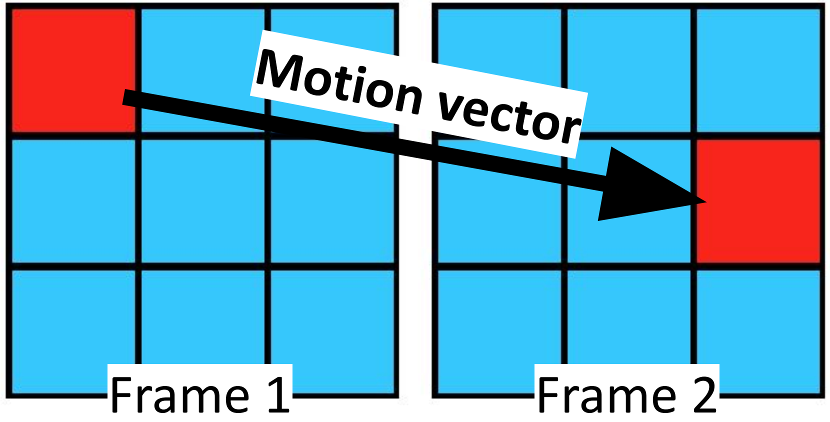
$$L = \int_A f(x)\,dx$$

2. Background

- Spatial denoising - increase per pixel information by sampling k neighbour pixels and adding a proportion of their contribution to the current fragment being rendered.

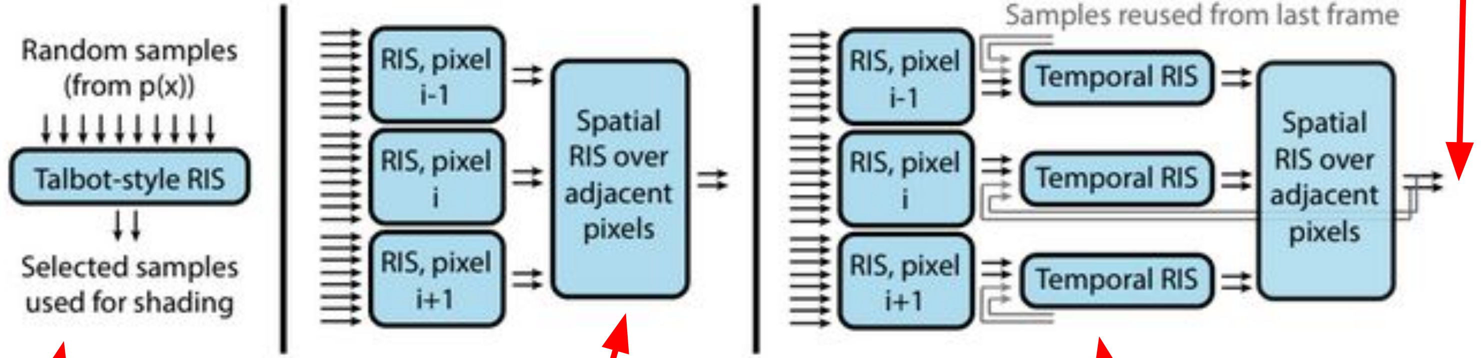


- Temporal denoising - increase per pixel information by sampling the same pixel in j frames of animation and adding their contribution to the current fragment being rendered.



2. Background

Still only 1 reservoir per pixel


Random samples (from p(x))
Talbot-style RIS
Selected samples used for shading
RIS, pixel i-1
RIS, pixel i
RIS, pixel i+1
Spatial RIS over adjacent pixels
Samples reused from last frame
Temporal RIS
Temporal RIS
Temporal RIS
Spatial RIS over adjacent pixels

New samples every frame

New neighbours every frame

New neighbour memories every frame

2. Background

- Bias: Because each pixel is independently calculating its own $\hat{p}$ term, when we combine samples in a reservoir, we must also consider that they all have independent and different PDFs.
- This creates an inherent bias in our solution to the rendering equation when we attempt to combine them, which will cause it to converge to an incorrect (biased) solution.
- The authors propose a method for correcting these errors and producing an unbiased version of the algorithm, but we will leave this out of this presentation for brevity (see paper for more details).

3. **Research Objectives/Hypotheses**

Research Question: How can real-time ray tracing algorithms be optimized for real-time applications on contemporary hardware to achieve near photorealistic rendering?

- For my project, I implemented the **Reservoir-based Spatio-Temporal Importance Resampling (ReSTIR)** algorithm in a custom-made C++ rendering engine with hardware accelerated ray tracing using the Vulkan graphics API.

4. **Methodology**

- C++ rendering engine using Vulkan API.
- GeForce RTX 3060 GPU.
- Amazon Lumberyard Bistro Exterior: https://developer.nvidia.com/orca/amazon-lumberyard-bistro
- 2,826,672 triangles in scene (slightly modified from original).
- 20,658 emissive triangles.
- Albedo and normal maps (no metalness/roughness maps due to VRAM limitations on laptop GPU 😢).
- 1 sample per pixel (SPP); path depth = “1” (+ 1 visibility ray).

5. Results

DEMO TIME

6. Algorithm Details

DEMO TIME

7. Discussion

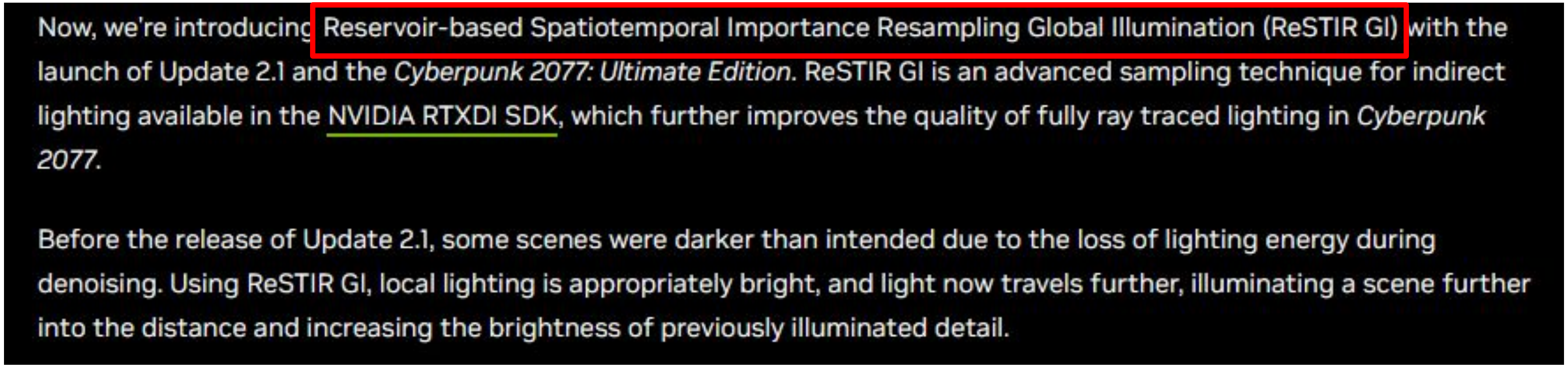
- ReSTIR Direct Illumination is working with a few visual artifacts.
- Will work on implementing the unbiased version of the algorithm and ReSTIR GI before project completion.
- By far, the most time consuming portion of this project thus far has been learning to use the Vulkan graphics API.
- Numerous C++/Vulkan improvements could be made (i.e. a better material loading system), but these types of seemingly simple changes are taking a long time to implement due to Vulkan’s verbosity.

8. **Conclusion**

- For this project, I chose to implement the Reservoir-based Spatial-Temporal Importance Resampling (ReSTIR) algorithm.
- I was able to successfully implement the biased version of the algorithm and will continue to complete the unbiased and Global Illumination versions.
- Further, I learned how to use the Vulkan graphics API, including RTX support which greatly improves the performance of ray tracing on compatible hardware (NVIDIA RTX GPUs).
- The final version of my project will be available in a publicly hosted Github repo to help others learn this method.
- This project was challenging and I was able to learn a lot, specifically about path tracing, statistics, and graphics in general.

9. **Postscript**

https://www.nvidia.com/en-us/geforce/news/rtx500-celebration-dlss-ray-tracing-new-games-win-prizes/#cyberpunk-2077



Now, we're introducing Reservoir-based Spatiotemporal Importance Resampling Global Illumination (ReSTIR GI) with the launch of Update 2.1 and the *Cyberpunk 2077: Ultimate Edition*. ReSTIR GI is an advanced sampling technique for indirect lighting available in the NVIDIA RTXDI SDK, which further improves the quality of fully ray traced lighting in *Cyberpunk 2077*.

Before the release of Update 2.1, some scenes were darker than intended due to the loss of lighting energy during denoising. Using ReSTIR GI, local lighting is appropriately bright, and light now travels further, illuminating a scene further into the distance and increasing the brightness of previously illuminated detail.

December 4, 2023

THANK YOU!