SDE885 - Photorealistic Rendering Algorithms Final Project

Matthew Georgy [1] Euan Hughes [2]

[1]Université de Montréal [2]École de technologie supérieure

December 03, 2025

Table of Contents

Table of Contents

Artistic Vision I

Theme: L’évolution du temps

Artistic Vision I

Theme: L'évolution du temps

- The passing of time often leads to aging and decay. What was once strong becomes weak.

Artistic Vision I

Theme: L'évolution du temps

- The passing of time often leads to aging and decay. What was once strong becomes weak.
- Simultaneously, time can also lead to development and advancement.

Artistic Vision II

Goal: Two eras collide

Artistic Vision II

Goal: Two eras collide

- An old desert temple - the remains of an ancient civilization.

Artistic Vision II

Goal: Two eras collide

- An old desert temple - the remains of an ancient civilization.
- Weathered materials, plant overgrowth, and dusty atmosphere illustrate its age.

Artistic Vision II

Goal: Two eras collide

- An old desert temple - the remains of an ancient civilization.
- Weathered materials, plant overgrowth, and dusty atmosphere illustrate its age.
- Blue-light side chamber with modern technology - a new civilization has come to study and learn from the old.

Artistic Vision III

Figure 1: Images generated with DeepAI [DeepAI 2025] and ChatGPT-5 [OpenAI 2025].

Requirements to Support Our Scene

Requirements to Support Our Scene

Capturing surfaces realistically

- Diverse set of materials
- Interesting appearance (texture, decay)

Requirements to Support Our Scene

Capturing surfaces realistically

- Diverse set of materials
- Interesting appearance (texture, decay)

Complex lighting

- Indirect illumination
- Caustics

Requirements to Support Our Scene

Capturing surfaces realistically

- Diverse set of materials
- Interesting appearance (texture, decay)

Complex lighting

- Indirect illumination
- Caustics

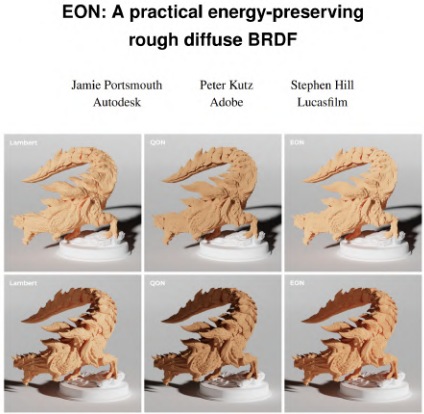
Participating media

- Dusty room
- Lighting that tells a story

Table of Contents

New BRDFs

MERL BRDF Database [Matusik et al. 2003]



Energy-preserving Oren-Nayar [Portsmouth et al. 2024]

MERL I

- MERL = Mitsubishi Electric Research Laboratories
- Collection of 100 measured BRDFs
- Not all were used; just some metallic paints and woods

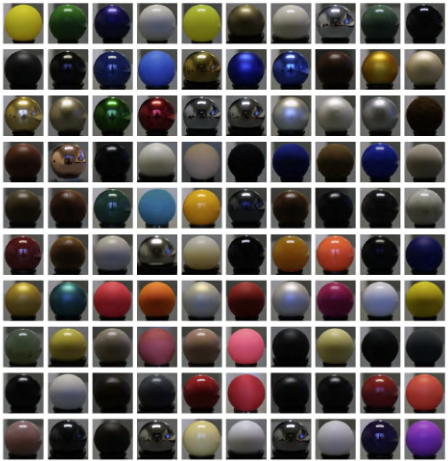
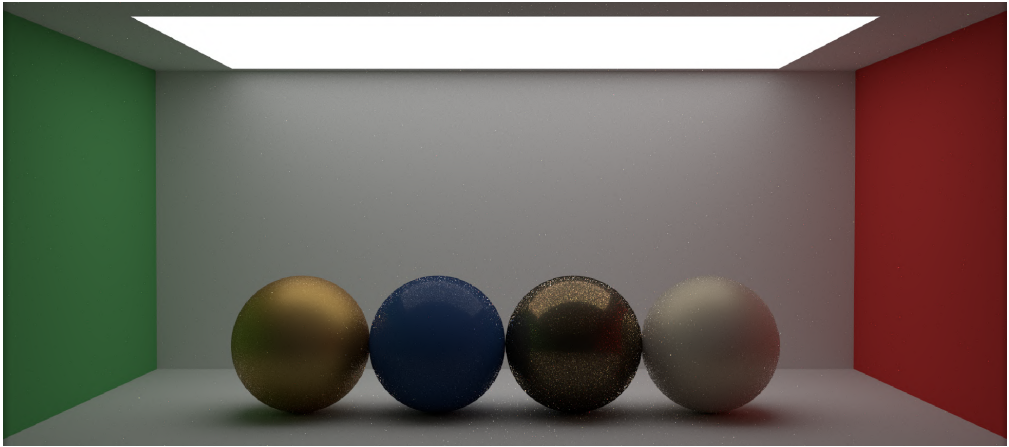


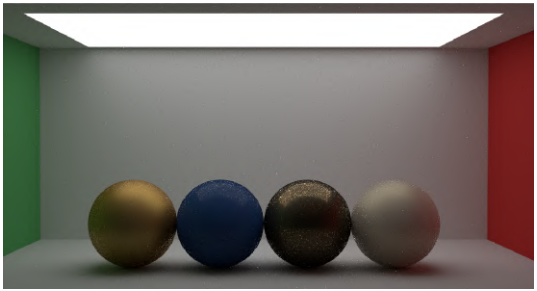
Figure 2: Figure 5 from [Matusik et al. 2003]



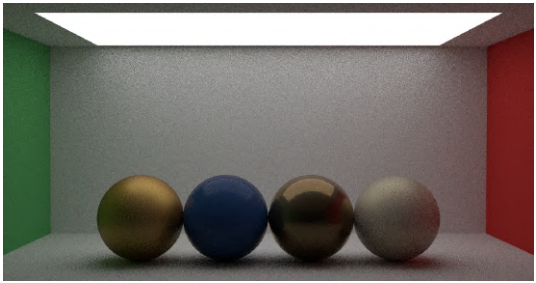
Path-MIS: 512 SPP (72 sec)

Figure 3: Sample of 4 materials: gold-metallic-paint, blue-acrylic, brass, pearl-paint.

Importance sampling... great quality but too slow



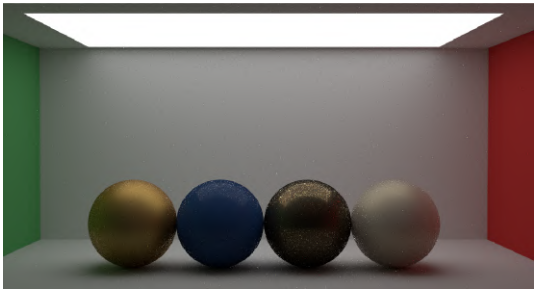
Path-MIS, naive: 128 SPP (20 sec)



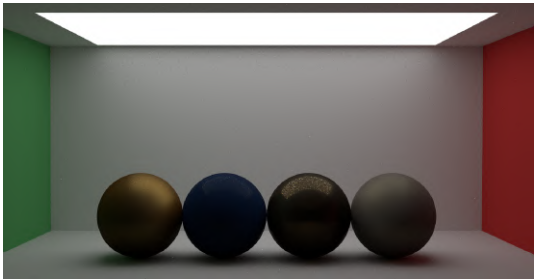
Path-MIS, method 1: 8 SPP (151 sec)

MERL III

Importance sampling... not quite correct but fast



Path-MIS, naive: 128 SPP (20 sec)



Path-MIS, method 2: 128 SPP (21 sec)

EON I

- Microfacet BRDF model
- Traditional Oren-Nayar models are not energy-preserving
- Paper introduces EON, which *is* energy-preserving

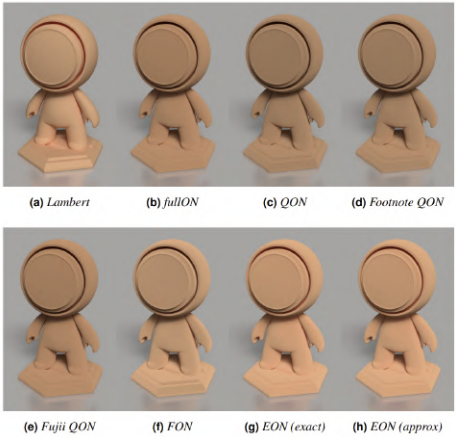
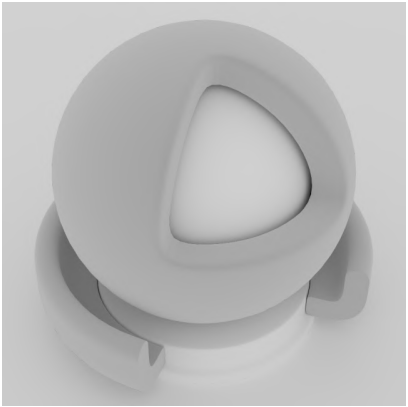


Figure 4: Lambertian vs various ON models; Figure 4 from [Matusik et al. 2003]

EON II

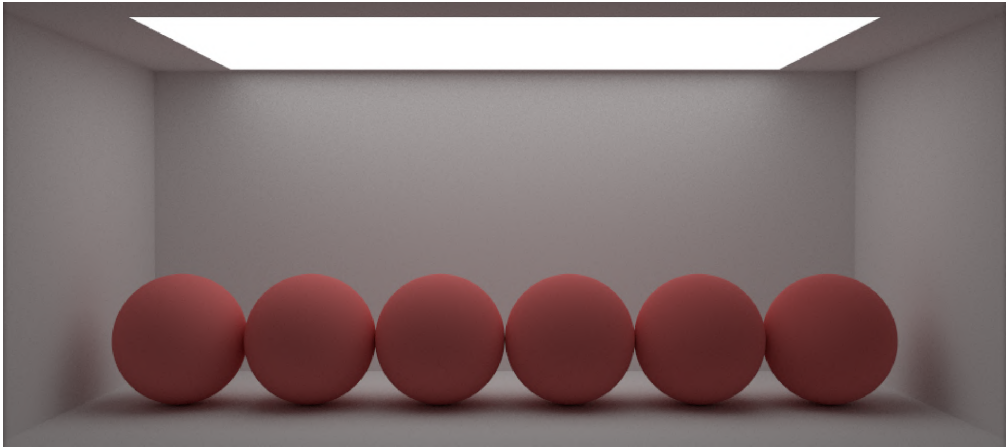


QON with $\sigma = \pi/2$

EON with $r = 1$

Figure 5: White furnace tests of QON vs EON; Figures 5, 9 [Portsmouth et al. 2024]

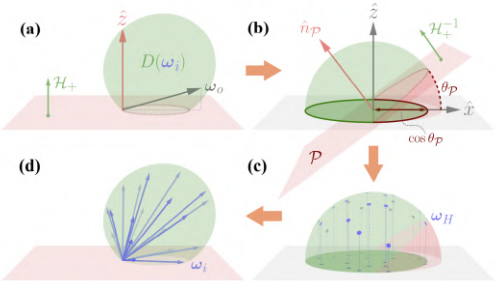
EON III



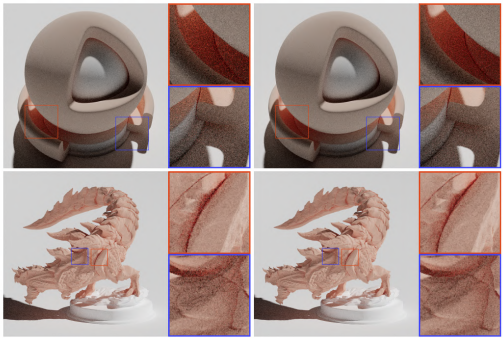
Path-MIS: 512 SPP (19.8 sec)

Figure 6: EON with varying roughness (from left-to-right): $r = 0, 0.2, 0.4, 0.6.0.8, 1$

EON IV



Geometry of CLTC importance sampling



Cosine sampling (left); CLTC sampling (right)

Figure 7: Importance sampling EON with CLTC; Figures 12, 14 [Portsmouth et al. 2024]

Normal Mapping

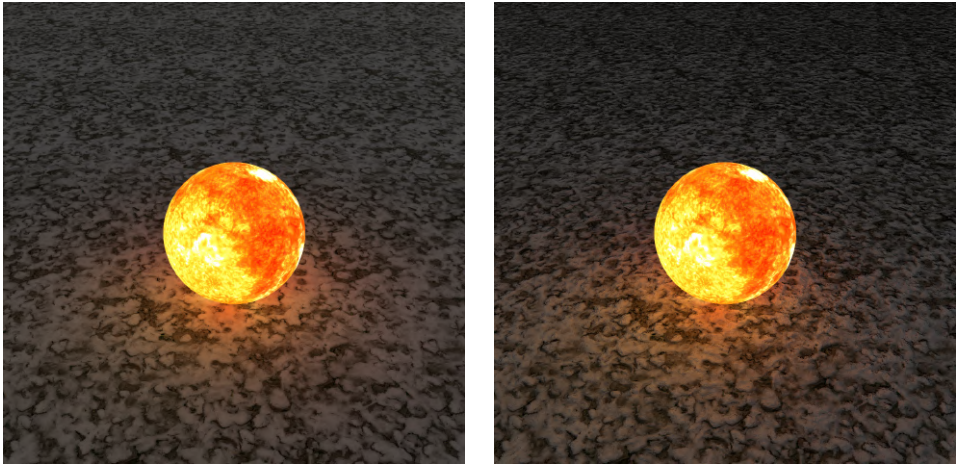


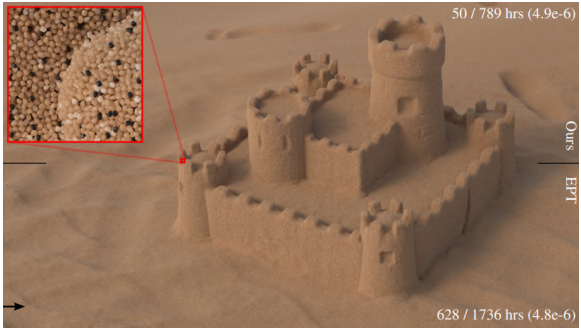
Figure 8: Diffuse surface without **(left)** and with **(right)** normal mapping. Textures from [Poly Haven Team 2021]

Perlin Noise

Perlin noise texture, based on *Raytracing: The Next Week* [Shirley 2020]

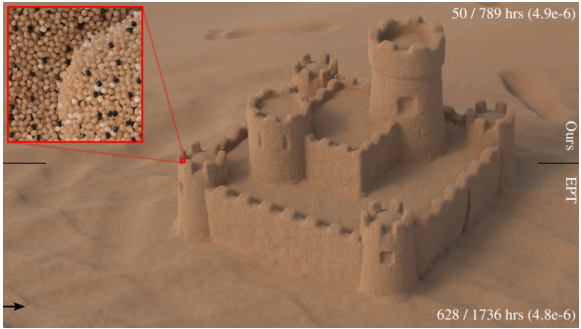
Figure 9: Different types of Perlin noise texture applied to various combinations of materials. Environment map from [Poly Haven Team 2021]

Sand I



Taken from Figure 1 of [Meng et al. 2015]

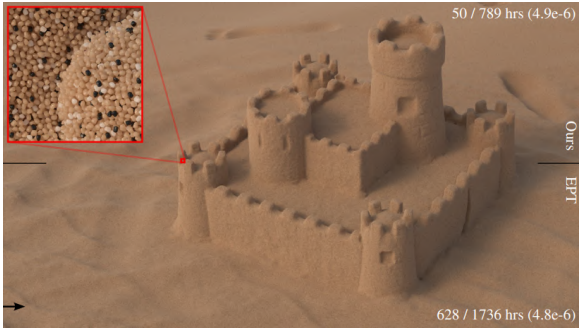
Sand I



Taken from Figure 1 of [Meng et al. 2015]

- Small geometry? Too slow, too much unneeded detail.

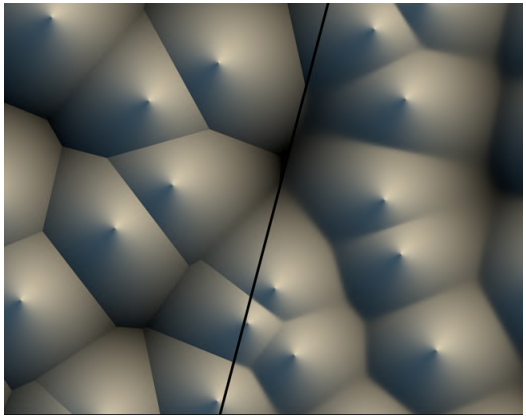
Sand I



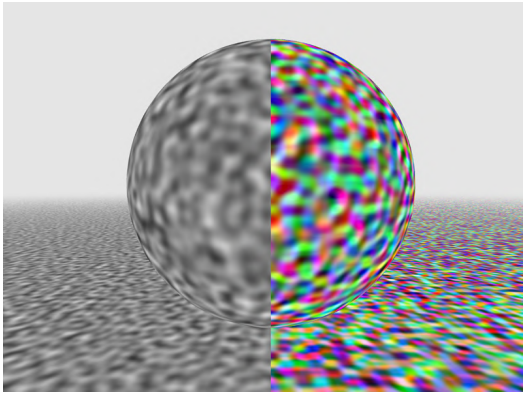
Taken from Figure 1 of [Meng et al. 2015]

- Small geometry? Too slow, too much unneeded detail.
- Procedural! Use noise to perturb the surface normal.

Sand II



Voronoi noise



Gradient noise

Figure 10: Example images of procedural noise from [Quilez 2012] and [Quilez 2017]

Sand III

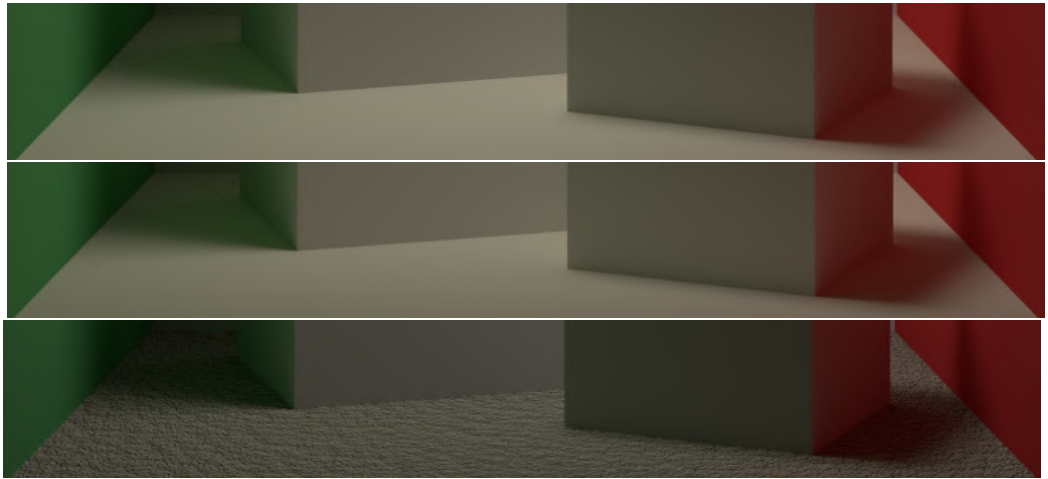


Figure 11: Cornell box floor: diffuse (top), EON (middle), EON + noise (bottom)

Table of Contents

Implementing Bidirectional Pathtracing I

```
1  # trace a path from the camera
2  def gen_eye_path(sampler, depth)
3
4  # trace a path from a random
       light
5  def gen_light_path(sampler, depth)
6
7  # connect eye vertex t and light
       vertex s, return contribution
8  def connect(e_path, l_path, t, s)
```

- Build an abstract interface for path space vertices (camera, light, surface)
- Perform random walk from the eye and light until max depth
- Create vertices at each interaction, tracking throughput
- Attempt a connection between all pairs of vertices, returning the path's contribution
- For a single camera vertex, must "splat" the world-space point onto the image plane

Implementing Bidirectional Pathtracing II

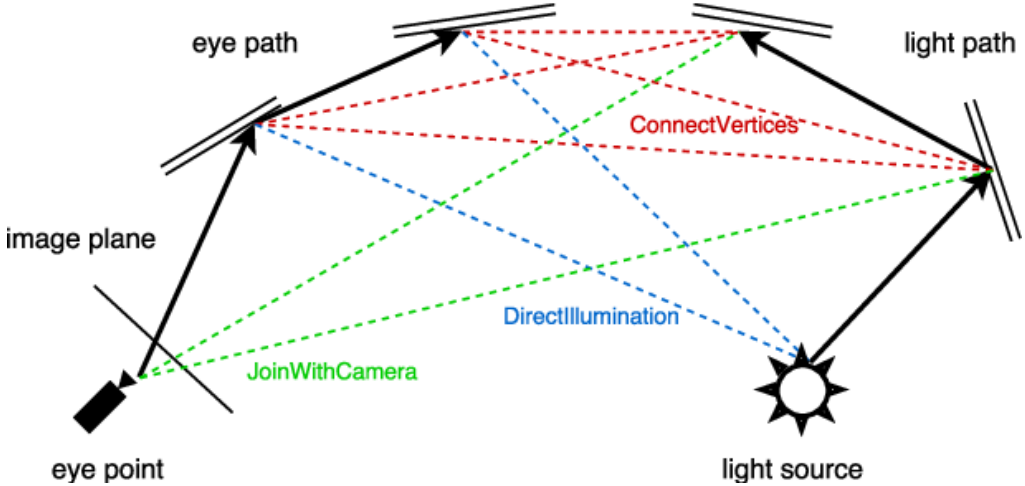


Figure 12: Overview of subpath connection strategies in BDPT. Image from [Vlnas 2022]

Figure 13: The individual bidirectional pathtracing strategies. Image from [Pharr et al. 2023]

Figure 14: MIS weighted contributions. Image from [Pharr et al. 2023]

Implementing Bidirectional Pathtracing III

The probability density of strategy s given a path of length $n = s + t$:

$$p_s(\overline{x}) = \text{pdfFwd}(x_0) \cdots \text{pdfFwd}(x_{s-1}) \cdot \text{pdfRev}(x_s) \cdots \text{pdfRev}(x_{n-1})$$

Implementing Bidirectional Pathtracing III

The probability density of strategy s given a path of length $n = s + t$:

$$p_s(\overline{x}) = \text{pdfFwd}(x_0) \cdots \text{pdfFwd}(x_{s-1}) \cdot \text{pdfRev}(x_s) \cdots \text{pdfRev}(x_{n-1})$$

We can cache pdfFwd and pdfRev in each vertex *while generating subpaths***

Implementing Bidirectional Pathtracing III

The probability density of strategy s given a path of length $n = s + t$:

$$p_s(\overline{x}) = \text{pdfFwd}(x_0) \cdots \text{pdfFwd}(x_{s-1}) \cdot \text{pdfRev}(x_s) \cdots \text{pdfRev}(x_{n-1})$$

We can cache pdfFwd and pdfRev in each vertex *while generating subpaths***

Recall: Balance Heuristic

Weight of strategy s:

$$w_s(\overline{x}) = \frac{p_s(\overline{x})}{\sum_i p_i(\overline{x})}$$

Implementing Bidirectional Pathtracing III

The probability density of strategy s given a path of length $n = s + t$:

$$p_s(\overline{x}) = \text{pdfFwd}(x_0) \cdots \text{pdfFwd}(x_{s-1}) \cdot \text{pdfRev}(x_s) \cdots \text{pdfRev}(x_{n-1})$$

We can cache pdfFwd and pdfRev in each vertex *while generating subpaths***

Recall: Balance Heuristic

Weight of strategy s:

$$w_s(\overline{x}) = \frac{p_s(\overline{x})}{\sum_i p_i(\overline{x})}$$

Problems with naive computation:

- under/overflow, even in double precision
- $O(n^4)$ time complexity!!

Implementing Bidirectional Pathtracing IV

Solution:

Ratio-based reformulation:

$$w_s(\overline{x}) = \frac{1}{1 + r_i(\overline{x})}, \qquad r_i(\overline{x}) = \begin{cases} 1, & \text{if } i = s \\ \dfrac{\text{pdfRev}(x_i)}{\text{pdfFwd}(x_i)}\, r_{i+1}(\overline{x}), & \text{if } i < s \\ \dfrac{\text{pdfFwd}(x_{i-1})}{\text{pdfRev}(x_{i-1})}\, r_{i-1}(\overline{x}), & \text{if } i > s \end{cases}$$

Full proof in PBRTv3 section 16.3.4 [Pharr et al. 2023]

Implementing Bidirectional Pathtracing IV

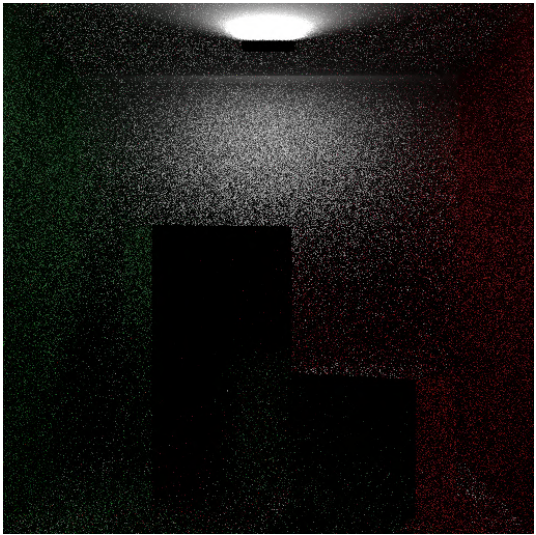
Solution:

Ratio-based reformulation:

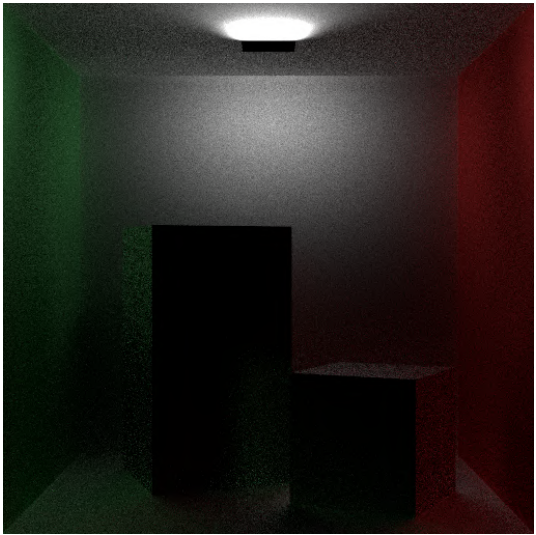
$$w_s(\overline{x}) = \frac{1}{1 + r_i(\overline{x})}, \qquad r_i(\overline{x}) = \begin{cases} 1, & \text{if } i = s \\ \dfrac{\text{pdfRev}(x_i)}{\text{pdfFwd}(x_i)}\, r_{i+1}(\overline{x}), & \text{if } i < s \\ \dfrac{\text{pdfFwd}(x_{i-1})}{\text{pdfRev}(x_{i-1})}\, r_{i-1}(\overline{x}), & \text{if } i > s \end{cases}$$

Full proof in PBRTv3 section 16.3.4 [Pharr et al. 2023]

Result: Better numerical stability, $O(n^2)$ time complexity.

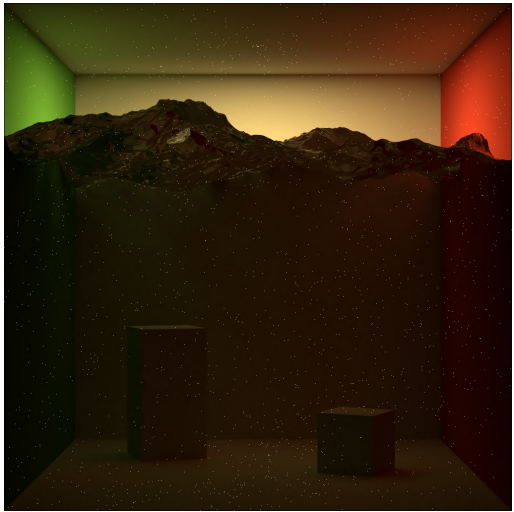


Path-MIS: 165 SPP (6.5 sec)

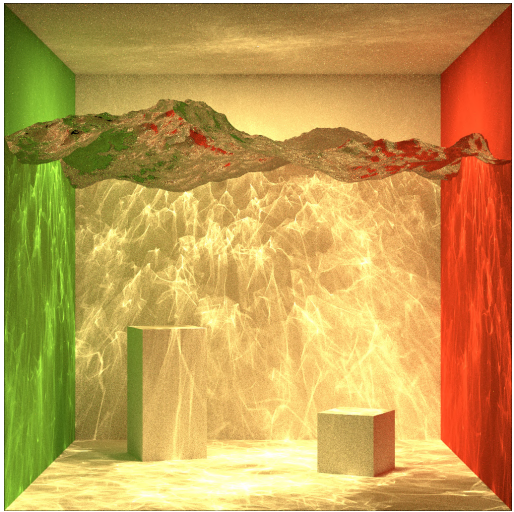


BDPT: 50 SPP (6.1 sec)

Figure 15: A glossy Cornell box with indirect illumination



Path-MIS: 1400 SPP (1124.6 sec)



BDPT: 45 SPP (98.2 sec)

$> \times 11.4$!!!

Figure 16: The *Water Caustic* scene, recreated using assets from [Bitterli 2016]

Primary Sample Space Markov Chain

Primary Sample Space Markov Chain

Key Idea:

treat the *random numbers* that generate a path as a chain **u** (an array of *primary samples*).

Primary Sample Space Markov Chain

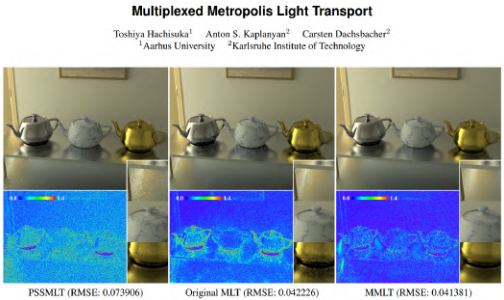
Key Idea:

treat the *random numbers* that generate a path as a chain **u** (an array of *primary samples*).

```
class MLTSampler:
    # propose new samples
    def mutate()

    # proposed ==> new state
    def accept()

    # revert to previous state
    def reject()
```

- The sampler maintains a *chain* of these samples (all samples = a path).
- On each MCMC iteration we propose a *small* or a *large* step:
  - **Small step:** Small Gaussian perturbation to the entries of **u** (local exploration).
  - **Large step:** fully resample **u** (global exploration).
- **Important:** store pre-mutation values. If the proposed chain is *rejected*, restore the values.

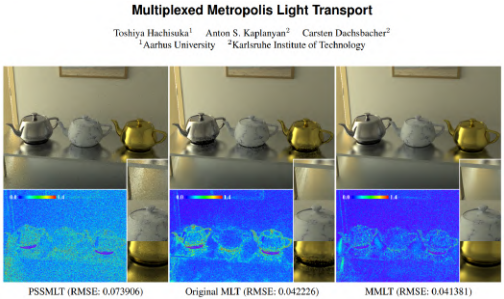
Multiplexed Metropolis Light Transport



[Hachisuka et al. 2014]

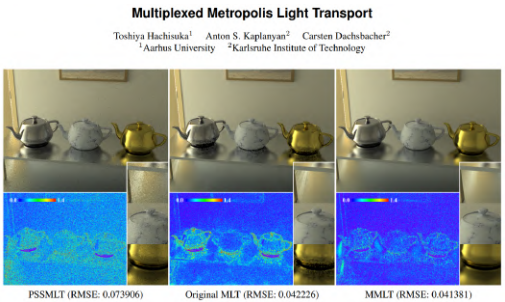
Multiplexed Metropolis Light Transport

Standard PSSMLT [Kelemen et al. 2002] uses BDPT MIS weights **only** in evaluation.



[Hachisuka et al. 2014]

Multiplexed Metropolis Light Transport

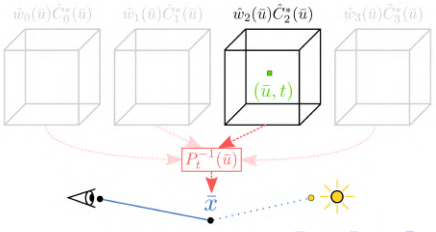


[Hachisuka et al. 2014]

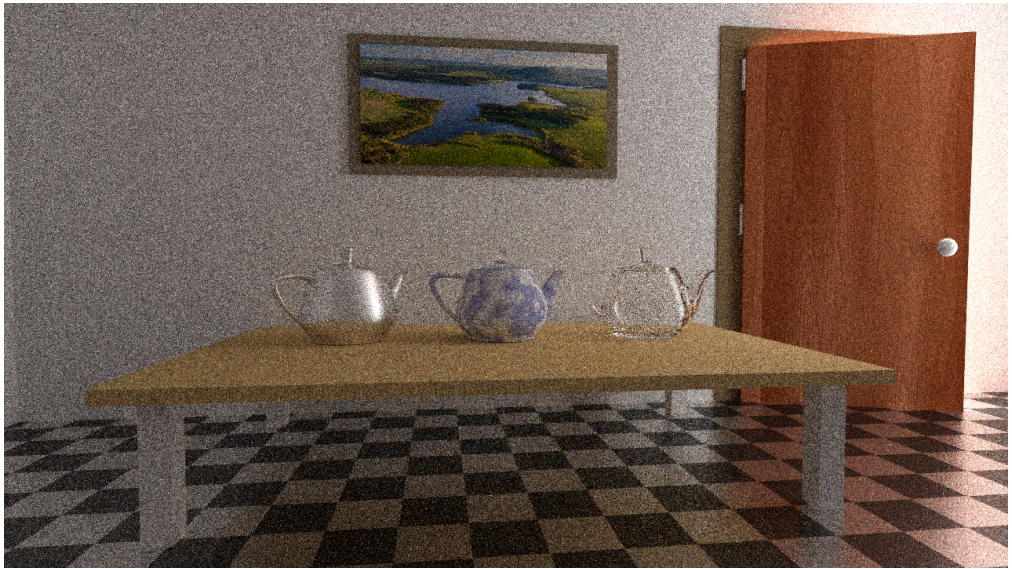
Standard PSSMLT [Kelemen et al. 2002] uses BDPT MIS weights **only** in evaluation.

MMLT

Incorporates MIS into the *sampling distribution itself* by allowing the Markov Chain to switch between all BDPT techniques according to their MIS importance.

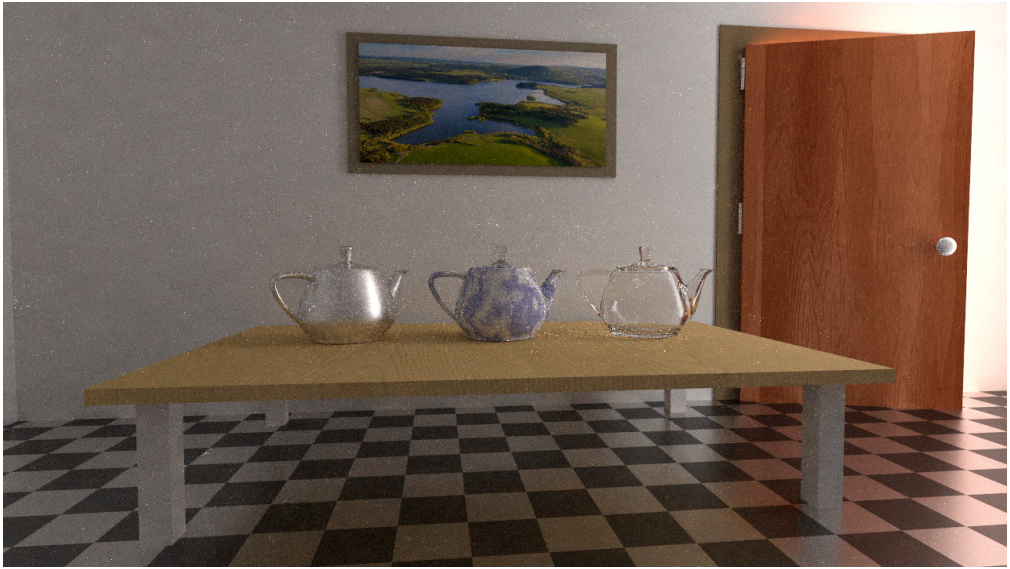


```
def chain_mutation_step(MLTSampler, k, Cur):
    MLTSampler.mutate()

    t = floor((k+2) * MLTSampler.t)
    s = (k+1) - t

    eye = gen_eye_path(MLTSampler, t)
    light = gen_light_path(MLTSampler, s)

    if eye.size() == t and light.size() == s:
        Proposed = connect(eye, light, t, s) * (k+2)

        a = Proposed.L / Cur.L
        if rand() < a:
            Cur = Proposed
            MLTSampler.accept()
        else
            MLTSampler.reject()

    accumulate(Cur)
```

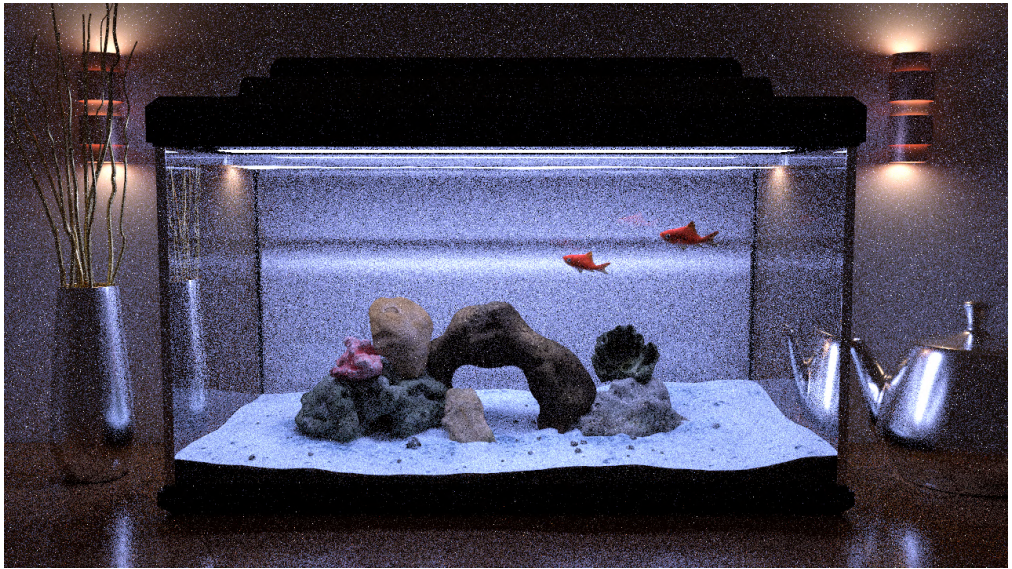


BDPT: 88 SPP (240.0 sec)

Figure 17: The *Veach Ajar* scene, recreated using assets from [Bitterli 2016]



MMLT: 700 MPP (235.7 sec)



BDPT: 470 SPP (30 min, 31 sec)

Figure 18: The *Aquarium* scene, recreated using assets from [Rioux-Lavoie et al. 2020]

MMLT: 2300 MPP (29 min, 54 sec)

Table of Contents

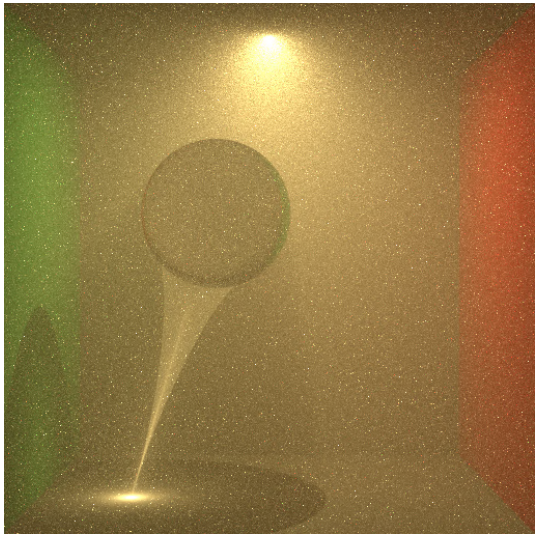
Adding Mediums to BDPT

```
class Medium:
    Color3 albedo, extinction
    PhaseFunction phase

    # sample a scatter direction
    def samplePhase(u2, wo)

    # pdf for incoming/outgoing
        pair
    def pdf(wo, wi)

    # sample free-flight distance
    def sampleDist(u)

    # attenuation between x and y
    def transmittance(x, y)
```

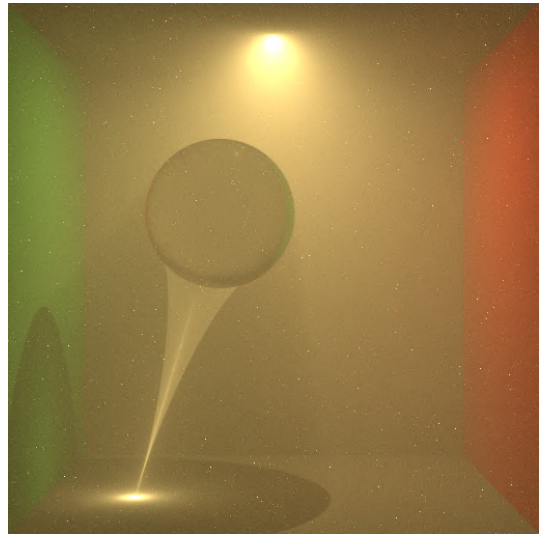
Adding Mediums to BDPT

```
class Medium:
    Color3 albedo, extinction
    PhaseFunction phase

    # sample a scatter direction
    def samplePhase(u2, wo)

    # pdf for incoming/outgoing pair
    def pdf(wo, wi)

    # sample free-flight distance
    def sampleDist(u)

    # attenuation between x and y
    def transmittance(x, y)
```

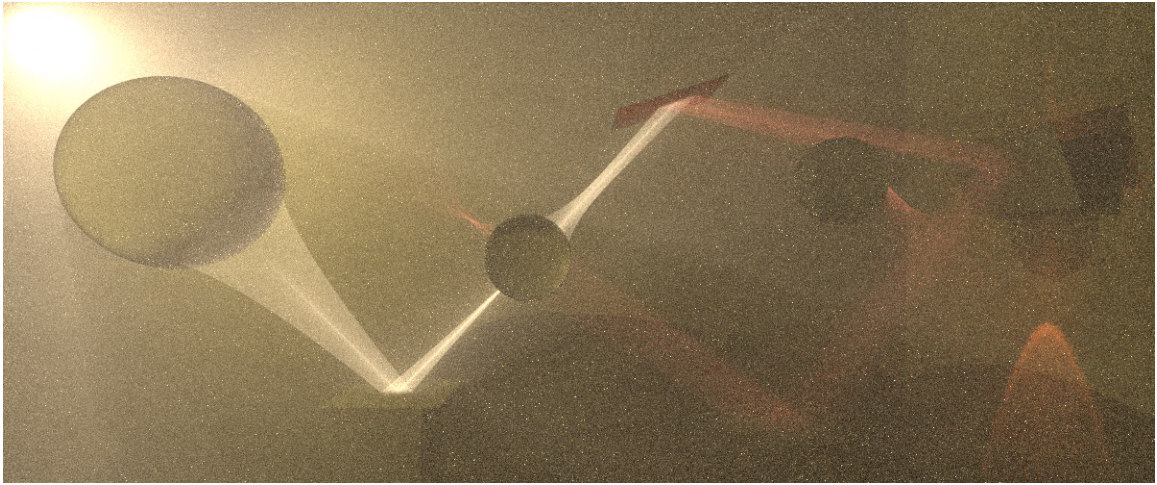
- Extend BDPT vertex abstraction with a **MediumVertex**
- During random walk: call `sampleDist(u)`; scatter if sampled distance $<$ distance-to-surface
- Call `samplePhase(u2, wo)`; create MediumVertex; continue walk
- When connecting, throughput scaled by transmittance between vertices
- Use an `in_medium` flag toggled when a ray transmits through a surface
- Allow camera/light to start either inside or outside the medium



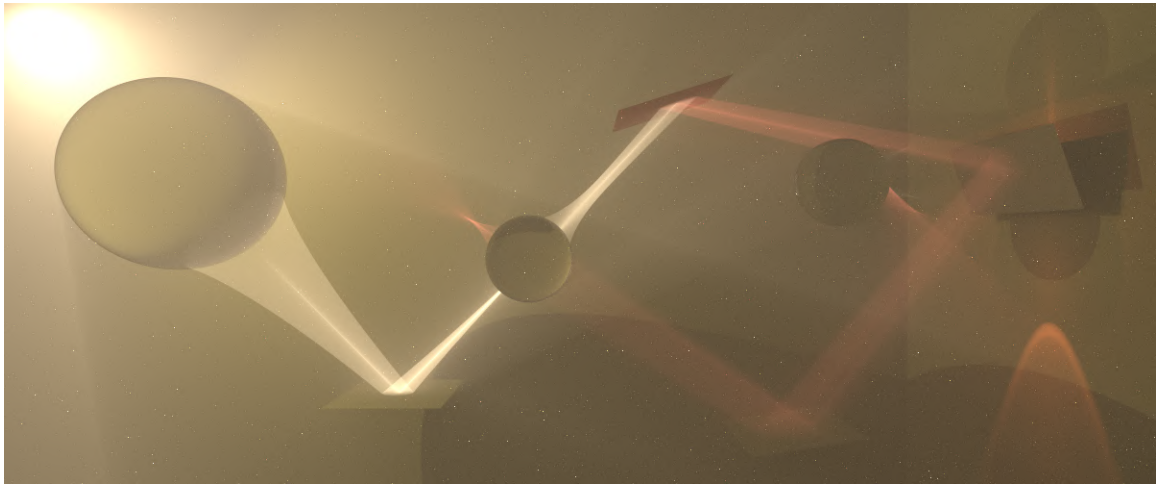
BDPT: 120 SPP (56.6 sec)



MMLT: 1500 MPP (55.9 sec)



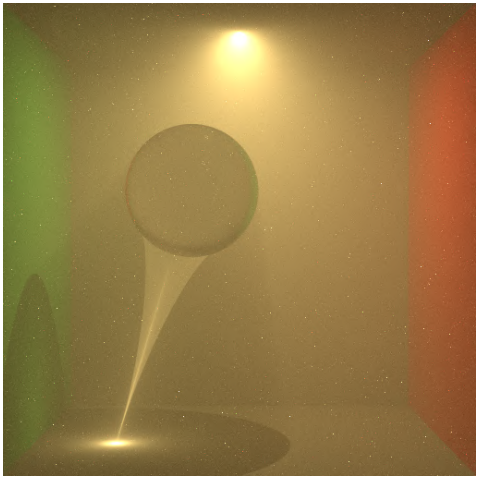
MMLT: 50 MPP (5.0 sec!!)



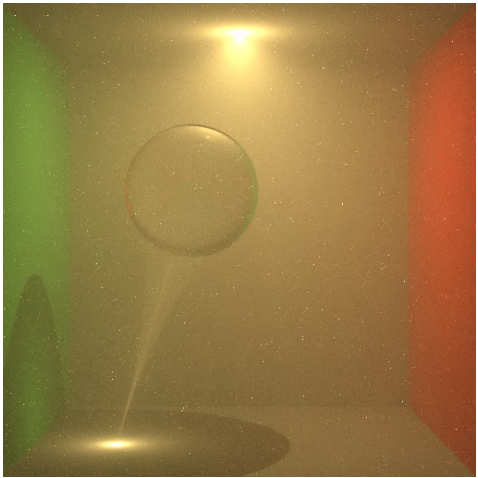
MMLT: 5000 MPP (508 sec)

Forward Scattering

Forward scattering using the *Henyey-Greenstein* phase function [Toublanc 1996]



Isotropic: 1500 MPP (55.9 sec)



HG ($g = 0.8$): 1500 MPP (62.7 sec)

Directional Lights

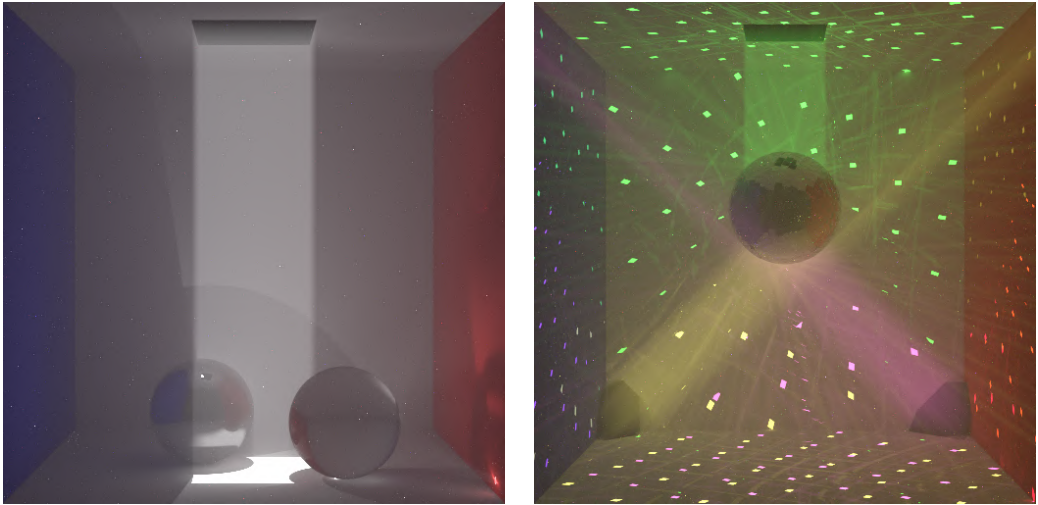


Figure 19: Discoball from [TurboSquid 2025]

Table of Contents

Acknowledgements

All assets in the final render were either created by us or sourced from BlenderKit [*BlenderKit Online Asset Library* 2024].

References I


Bitterli, Benedikt (2016). *Rendering resources*. https://benedikt-bitterli.me/resources/.


BlenderKit Online Asset Library (2024). Accessed: 2025-12-03. BlenderKit. URL: https://www.blenderkit.com.


DeepAI (2025). *AI Image Generator*. URL: https://deepai.org/machine-learning-model/text2img.


Hachisuka, Toshiya, Anton S Kaplanyan, and Carsten Dachsbacher (2014). "Multiplexed metropolis light transport". In: *ACM Transactions on Graphics (TOG)* 33.4, pp. 1–10.


Kelemen, Csaba, László Szirmay-Kalos, György Antal, and Ferenc Csonka (2002). "A simple and robust mutation strategy for the metropolis light transport algorithm". In: *Computer Graphics Forum*. Vol. 21. 3. Wiley Online Library, pp. 531–540.


Matusik, W., H. Pfister, M. Brand, and L. McMillan (July 2003). "A Data-Driven Reflectance Model". In: *ACM Transactions on Graphics (TOG)* 22.3, pp. 759–769. ISSN: 0730-0301. DOI: 10.1145/882262.882343. URL: https://www.merl.com/publications/TR2003-83.


Meng, Johannes, Marios Papas, Ralf Habel, Carsten Dachsbacher, Steve Marschner, Markus Gross, and Wojciech Jarosz (2015). "Multi-Scale Modeling and Rendering of Granular Materials". In: *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*.


OpenAI (2025). *ChatGPT (GPT-5)*. Version GPT-5. Large language model. URL: https://chat.openai.com.


Pharr, Matt, Wenzel Jakob, and Greg Humphreys (2023). *Physically based rendering: From theory to implementation*. MIT Press.


Poly Haven Team (2021). *Poly Haven 3D Asset Library*. https://polyhaven.com/. Accessed: 2025-12-02.


Portsmouth, Jamie, Peter Kutz, and Stephen Hill (2024). "EON: A practical energy-preserving rough diffuse BRDF". In: *arXiv preprint arXiv:2410.18026*.

References II


Quilez, Inigo (2012). “smoothvoronoi”. In: URL: https://iquilezles.org/articles/smoothvoronoi/.


— (2017). “gradientnoise”. In: URL: https://iquilezles.org/articles/gradientnoise/.


Rioux-Lavoie, Damien, Joey Litalien, Adrien Gruson, Toshiya Hachisuka, and Derek Nowrouzezahrai (2020). “Delayed rejection Metropolis light transport”. In: *ACM Transactions on Graphics (TOG)* 39.3, pp. 1–14.


Shirley, Peter (2020). “Ray tracing: The next week”. In: *URL: https://raytracing. github. io/books/RayTracingTheNextWeek. html*.


Toublanc, Dominique (1996). “Henyey–Greenstein and Mie phase functions in Monte Carlo radiative transfer computations”. In: *Applied optics* 35.18, pp. 3270–3274.


TurboSquid (2025). *TurboSquid – 3D Models for Every Budget*. URL: https://www.turbosquid.com/ (visited on 12/02/2025).


Vlnas, Michal (2022). “Bidirectional Path Tracing”. In: *22nd Central European Semi- 61 nar on Computer Graphics* 1, pp. 9–11.