

A Stochastic Ray Tracing Solution for Perceptually Stable Real-Time Global Illumination

Paul Iordache

iordache.o.paul@student.utcluj.ro
Technical University of Cluj-Napoca
Cluj-Napoca, Cluj, Romania

Constantin Nandra✉

Technical University of Cluj-Napoca
Cluj-Napoca, Cluj, Romania
constantin.nandra@cs.utcluj.ro

ABSTRACT

Ray Tracing (RT) is a widely used technique in cinema, architecture, and engineering due to its elegant nature. It simulates the behavior of light in a simple and precise way. Path tracing had a breakthrough in the media industry with the release of *Monsters University*. Real-time ray tracing has not been viable before the introduction of the Turing architecture in 2018. Until that point, wholly photorealistic images were only possible using offline approaches. In this day and age, developing a high-performing, consistent ray tracing system with a denoiser that does not take a great toll on performance is crucial in order to deliver a qualitative user experience. The system described within this paper provides an overview of a low sample per pixel ray tracing integration, a practice that facilitates performance and stability, but introduces variance, which translates to visual noise, mitigated by SVGF. The usability of the application, expressed in terms of low-stuttering sequence frames and below or equal to 16 milliseconds processing image times, will be discussed, as well.

AUTHOR KEYWORDS

Ray tracing; performance; physically based rendering; stability; real-time rendering; Vulkan.

DOI: 10.37789/icusi.2026.36

Introduction

Real-time rendering applications are held to a strict standard of producing a new image, on average, every 16.6 milliseconds to sustain a 60 frames per second experience considered acceptable for interactive use. Until recently, ray tracing was mainly used in offline rendering, for a single frame consumed hours of processing across thousands of CPU cores [8]. Interactive applications were restricted to using rasterization, because it was not feasible to render realistic images due to the lack of computation power and dedicated ray tracing hardware components. The first documented iteration of ray tracing is Appel's [3] introduction of point-by-point Lambertian shading. Nevertheless, to achieve realistic light in different mediums, a recursive approach is required [35]. The difficulty of implementing accurate illumination algorithms stems from the computational hurdles a system endures during execution. Infinite recursion is an insurmountable idea in development, and, despite this limitation, Kajiya's early path tracer [16] still serves as the reference formulation of future proceedings. Path Tracing, by industry's standards, implies several light bounces need to be optimized and deliberately selected as a way to avoid spending time on unproductive beams.

Monte Carlo integration [31] is synonymous with this rendering technique, since stochastically generating thousands of

rays, synchronously weighing and evaluating their contribution, takes a toll on the system's ability to converge to a final, stable product [27]. Multiple importance sampling (MIS) was developed by Veach and Guibas [33], thereafter Veach published his PhD [32] which is, notwithstanding, reference to modern ray tracers. By relying on Monte Carlo methods to simulate in real-time light's relation with materials and surfaces, visual noise will be undoubtedly present. Even though MIS equations attempt to alleviate the impact of variance on image quality, they are powerless in predicting ray tracing's incoherent nature [27].

With the introduction of the RTX 20 series in 2018, programming specialized shaders and API [23] integration has become widespread among developers. Dedicated ray tracing cores paved the way for realistic reflections and effects that were not attainable before when it comes to interactive media [24]. Nowadays, artificial intelligence aided denoisers and image reconstruction algorithms swayed the focus towards hardware integration of detailed AI components in GPUs [25]. While hardware acceleration and AI denoising have made real-time ray tracing computationally feasible, this milestone alone does not guarantee a usable interactive experience. Frame rate is frequently treated as the sole indicator of real-time viability, yet recent studies show that consistency of frame delivery affects user experience independently of average throughput [20].

This paper presents an efficient and scalable ray tracing system which is optimized for rendering complex scenes and does not present high f/s variance. Its performant, hybrid essence emanates from dividing the workflow into separate stages - geometry, ray tracing, image reconstruction, composition - that communicate and are intrinsically entire pipelines [12]. The ray tracing stage focuses on modeling the bidirectional reflectance distribution function (BRDF), omitting transmittance and non-opaque objects addressed by the bidirectional scattering distribution function (BSDF) or the bidirectional scattering surface reflectance distribution function (BSSRDF). As 1 sample per pixel (spp) ray tracing is inherently noisy, this system adopts Schied's et al. [29] spatiotemporal variance guided filter and modifies it to support separate diffuse and specular lighting filtering.

This architecture is evaluated not just on theoretical visual fidelity, but also on strict performance scaling and end-user usability. By rigorously profiling hardware execution times across discrete pipeline stages, identifying geometric bottlenecks, and measuring temporal variance to account for micro-stuttering, it is demonstrated a resilient rendering application.

Bibliographic Research

Physically Based Rendering

James T. Kajiya introduced the inaugural rendering equation [16] as highly inspired by radiative heat transfer, the formula

has its roots in pure physics and graphics engineers have spent decades in order to approximate it. Kajiya's method cannot be solved analytically, it dictates that it accounts for every possible point; since it integrates over S , which represents the union of all surfaces, even state of the art GPUs do not have the computational capabilities to render the entire scene in real-time, with decent framerates. As discussed in Pharr et al.'s book [27], a more suitable equation to achieve both performance and image quality is The Light Transport Equation. It states that the exitant radiance L_o must be equal to emitted radiance L_e plus the fraction of incident radiance that is scattered.

For the proposed implementation, Burley's diffuse BRDF [5] model was omitted, as adopting the original Lambertian model avoids expensive computations while it introduces minor deviations regarding the visual quality. This decision is motivated by Karis' course notes [17], the justification touches on the difficulty implementing Image-Based Lighting (IBL) using Disney's model, as well. Karis also found out that modifying Fresnel by replacing the power with Spherical Gaussian approximation proves to be slightly more efficient and the difference is imperceptible.

Burley [6] has also recognized the inexactitude of Smith's Geometry term [34], and endorsed Eric Heitz's [14] adjustment regarding the formerly uncorrelated masking and shadowing probabilities. Further research [18] suggested that for offline renderers, a pre-calculation of energy loss for every view angle and surface roughness should be baked into a look-up table and reintroduced into the material when it is hit. Fdez-Agüera [10] developed an analytic solution to migrate multiple surface scattering at microscopic level.

Monte Carlo Integration

Careful sampling and practices are the core ideas of reducing variance and it is favorable to make more of a limited number of samples than to increase the amount [27]. Importance sampling is a powerful tool to enable an integral to converge quicker and is widely adopted and regarded as the predominant practice for variance reduction, compared to stratified sampling which performs inadequately given a large numbers of dimensions to sample from [27].

In his PhD thesis, Eric Veach [32] expanded on the idea of Multiple Importance Sampling and came up with various heuristics, concerning optimally drawing samples and weighting them for closely matching the shape of the integrand. The balance and power heuristics are most notable, with the latter reducing the variance further for an exponent $\beta = 2$.

Although the most ubiquitous for sampling specular lobes was the GGX model for its long-tailed shape blur effect of specular highlights, it was replaced by Heitz's VNDF in 2014 [14] for its inclusion of the viewing angle in the probability density function, while Walter et al.'s [34] microfacet distribution blindly evaluated half vectors and was burdening the computations with wasteful rays. However, VNDF was not pristine either with respect to performance, because the algebra was cumbersome and was later revisited and optimized [15] to bypass the computations by stretching the GGX distribution into a uniform hemisphere.

Another issue when sampling is concerned is rendering direct lighting from millions of dynamic light sources because they are directly proportional with tracing rays, leading to evaluation of several materials. Benedikt et al. [4] designed

a method to sample one-bounce direct lighting from many lights, called ReSTIR. It extends resampled importance sampling by introducing a "reservoir", a data structure that stores only accepted samples and by reusing statistic from temporal and spatial neighbors.

Noise

Importance sampling helps reducing noise by converging quicker, when ray budgets are prohibitive in real time ray tracing (RTRT), thus low-discrepancy sequences are vital for making the noise less apparent and visually appealing [21]. In 1960, J. H. Halton [13] established an advancement in sampling distributions, reducing Monte Carlo error bounds. Its high-dimensional correlation flaws are apparent, but it laid the groundwork for subsequent Quasi-Monte Carlo rendering techniques. I. M. Sobol in 1967 [30] revolutionized the field of sampling by introducing an algorithm that was reliant on base 2 numbers, maintaining strict spatial uniformity even in high dimensions and exceptional computation times.

Blue noise can be considered as a low-discrepancy distribution, for it aims to spread the points uniformly in the domain, as well. Mitchell's best-candidate algorithm [22] generates samples using the toroidal distance and assumes that candidates are able to be extracted with the same probability density. Although its rate of convergence is complementary to white noise, its starting error is lower [21], and compared to Halton [13] and Sobol [30] it struggles to reach small absolute error results after several data points extracted for a Gaussian function, but performs nearly as efficient as Halton's sequence given a diagonal step function. Blue noise mitigates visible error by distributing samples in a pattern that closely mimics the spatial arrangement of human retinal photoreceptors, as noted by Lanaro et al [19].

Ahmed and Wonka [2] developed an algorithm which circumvents traditional per-pixel optimization by approaching sample generation as a global, screen-space operation. It achieves a blue noise pattern distribution of samples by leveraging Sobol sequences, applying Morton ordering and wrapping these steps in a stochastic scrambling. Later, Ahmed et al. [1] introduced Gaussian Blue Noise, shifting the focus from screen-space distribution to the fundamental generation of optimal blue noise point sets, minimizing the variance of a sum of Gaussian kernels.

Performance

Geris et al. [11] outline the importance of frames per second (f/s), framing this concept as one of the determining factors in efficacy. Although the study focuses on the Virtual Reality (VR) experience, they discuss the experience disruption of the user when they interact with a low f/s application, which is relevant for this paper's scope. Dossena and Trentini [9] look at latency as another factor in a system's usability. High latency, as mentioned in the paper, renders the application as barely usable, and will be a certain consideration for the future development of the presented implementation. Liu et al. [20] emphasizes on the importance and effects of frame rate variance. They noted that, in an interactive environment, such as video games, lower frame rates have a greater impact on the user's experience compared to passively watching a video, over and above frame time variance proved to be a powerful indicator that may dictate the enjoyment of engagement with the product on user end.

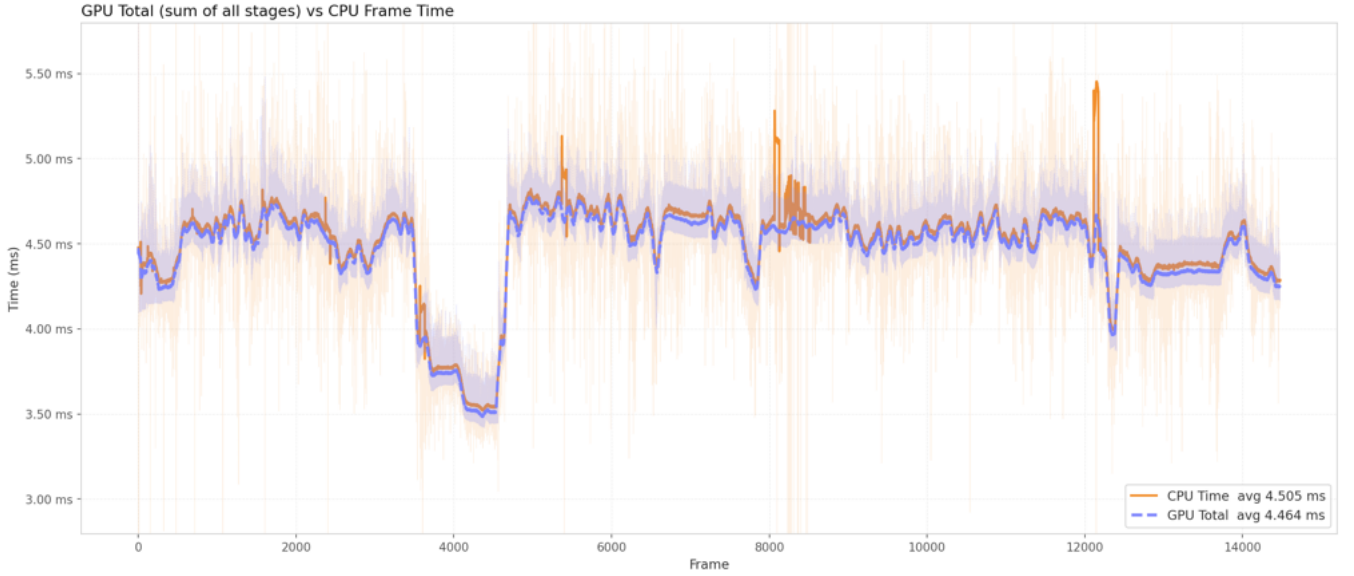


Figure 1: GPU vs. CPU

Hybrid Pipeline

Below is provided a high-level overview of the underlying hybrid pipeline which blends rasterization with ray tracing with the objective of leveraging light computation for the latter. This section presents the relationship between the deferred stages, as they become the core of the rendering system, given that they scale independently based on the scene’s complexity.

G-Buffer Stage

In an effort to effectively generate the scene’s geometry and forward crucial textures to subsequent stages, adopting a G-buffer methodology is imperative. This is the foundation of a hybrid system, for ray tracing could trace primary rays from the camera to the screen pixel, though it tremendously slow down the application in the process. Rasterization is specialized in identifying drawable objects, although it falls short in portraying true to life visual effects, such as reflections and indirect lighting.

Ray Tracing Stage

In this stage, akin to Appel’s [3] introductory method, the system casts one ray for light evaluation, not restraining from considering other diffuse models, and another ray for visibility which offers the possibility of producing soft shadows. Exploiting Vulkan’s hardware ray tracing integration, flexibility in developing dedicated shaders is facilitated.

Diffuse Reflectance Model. Burley [5] observed that several materials show a drop in grazing retroreflection, while some show a peak. Furthermore, traditional diffuse models assume a smooth surface or ignore the Fresnel effect, which predicts a grazing shadow. They put forward a model that integrates Fresnel to closely match the MERL database. Karis [17], on the other hand, saw little differences between Burley’s model and the original Lambertian model, thus justifying its omission by bringing into discussion the additional computation cost and complexity in efficient use in the case of image-based or spherical harmonic lighting. Another worth mentioning model is Oren-Nayar’s generalization [26], which

treats rough surfaces as a collection of long, symmetric V-cavities, described by a spherical Gaussian distribution. Although, it fails the white-furnace test, an issue addressed by Portsmouth et al. [28] in its implementation of the Energy-Preserving Oren-Nayar (EON) model. Both of the mentioned Oren-Nayar iterations present visual bugs when implemented, issues discussed and showcased in Section .

Specular Reflectance Model. The Torrance-Sparrow model coupled with its corresponding probability density function (PDF) is one of the most recent and suitable approaches in regards to the specular model [27] (see Equation 1). It is a combination of statistics, in the form of microfacet theory, and physics. The Fresnel term is noted as being performance heavy, with the Schlick’s approximation offering 32 times faster performance with less than 1.5% average error [21]. The Lazányi-Schlick is an alternative approximation which gives a reduced average error of 0.22% with a maximum of 0.65% [21]. The system ultimately implements Schlick’s approximation with the power replaced with the Spherical Gaussian approximation [17], for it does not track an object’s index of refraction.

$$f_{\text{Torrance-Sparrow}} = \frac{D(\omega_m)F(\omega_o \cdot \omega_m)G(\omega_i, \omega_o)}{4 \cos \theta_i \cos \theta_o} \quad (1)$$

Sampling a point on the light source is beneficial if there should be a far illuminant object. Conversely, sampling the BRDF does not guarantee that the chosen direction will hit the source. Thus, multiple importance sampling offers the solution to this issue of deciding which method to use [32]. The formerly mentioned technique integrates Wolfe’s method [21] of using a blue noise texture to reproduce soft shadows.

Testing and Validation

Evaluation of this paper’s real time ray tracer is done on a number of metrics. Tests regarding performance in milliseconds per frame, time spent per graphical stage, overall GPU along with CPU execution times, and stability have been carefully evaluated and critically interpreted in order

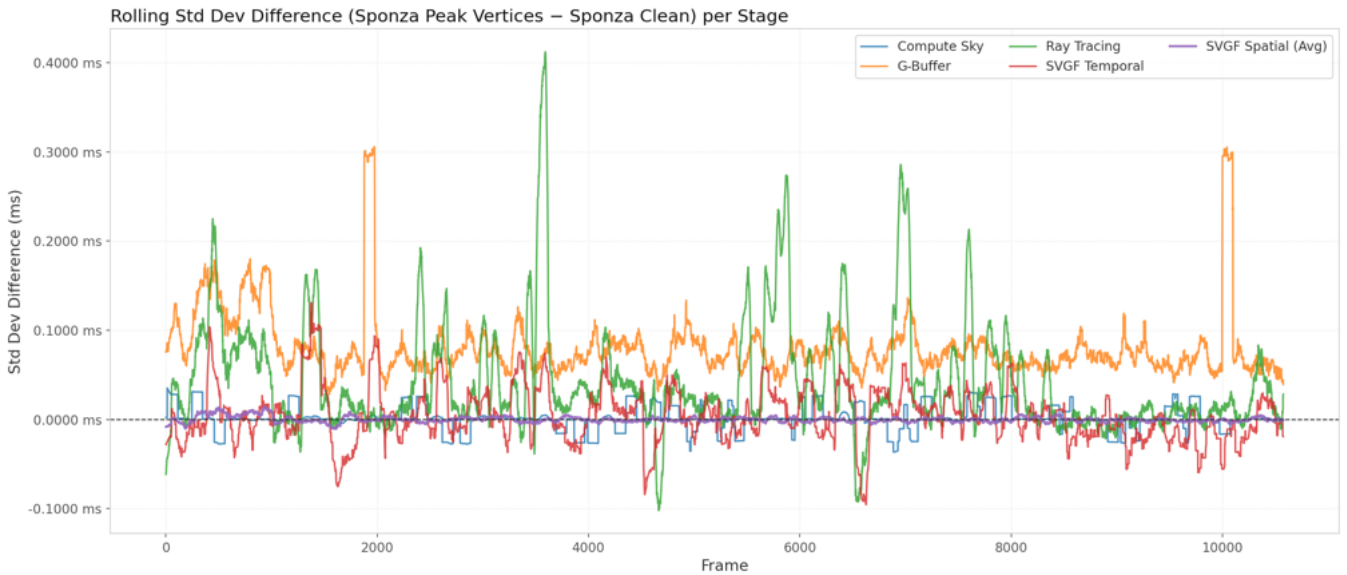


Figure 2: Rolling Standard Deviation for All Stages

to highlight the implemented system's ability of fast rendering under various conditions. Increasing the number of vertices and light sources serve as the definitive compass in determining whether the application handles such conditions. The implementation uses Vulkan for the API communication with the GPU. All reported performance are recorded on an NVIDIA RTX 5070 (Blackwell architecture), with a resolution of 1080p, 60 hz.

GPU and CPU Time

Evaluating the overall performance of the rendering engine requires a comparative analysis of the CPU and GPU workloads, so as to determine a possible logical bottleneck between the two components. Figure 1 illustrates the aggregate GPU render time contrasted against the CPU frame preparation time. The data demonstrates that, given hardware compatible technology, there is no GPU, nor CPU bounding. The application maintains a consistent GPU execution time of approximately 4.46 ms (217 f/s) for the Sponza model, and 7.42 ms (134 f/s) for the peak complexity scene that is comprised of multiple meshes that sum up to 23 million vertices. Concurrently, the CPU time nearly overlaps with the GPU's. Tracking GPU rendering time alone is not a definitive method in proving the system's efficiency and performance, for it is



Figure 3: Sponza Scene

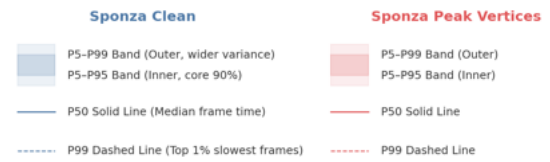


Figure 4: Percentile Bands Explained

composed of several other metrics which uncover information not explained by computing the total mean of all stages.

Per Stage Times

Each graphical stage behaves differently given certain parameters. From our analysis, tracing rays in a Vulkan acceleration structure environment takes up, on average, 53.3% of the total GPU measurement (2.37 ms individual, 2.38 truncated). Once the number of vertices is drastically escalated, there is an increased difference of 21.2028%, jumping to 2.88 ms. Though the percentage increase is close to a quarterly transformation, built-in acceleration structures and Vulkan specific ray tracing optimizations augment intersection tests capabilities.

However, the G-buffer stage is incredibly affected when rendering the peak vertex scene, averaging an increase of 605.214% (0.4104 truncated ms to 2.8942 ms) in time and a raise by 97.7303% of instability. This is caused by the massive bandwidth sizes of the multiple render targets imposed by the G-buffer methodology.

Variance Analysis

In real-time rendering, purely measuring average values constitutes a significant flaw. While the mathematical mean μ provides a broad indication of overall computational throughput, it entirely obfuscates the temporal distribution of frame delivery.

Temporal Instability via Differential Rolling Variance. Variance in the context of perception serves as an intrinsic metric for temporal consistency, correlating to the phenomenon of micro-stuttering. Frequent fluctuation between 8 ms and 15 ms yields, compared to a highly consistent frame delivery of 33.33 ms (30 f/s), an unacceptable experience for the end-user. Rolling variance refers to the concept of using a sliding window instead of measuring the standard deviation for the whole frame indices. The rolling formulation preserves time as an axis, as a result.

$$\sigma_t = \sqrt{\frac{1}{w-1} \sum_{i=t-w/2}^{t+w/2} (x_i - \bar{x}_t)^2} \quad (2)$$

Computing $\Delta\sigma_t = \sigma_t^{peak} - \sigma_t^{clean}$ serves a specific analytical purpose, controlling for the baseline volatility that is common for both scenes. A positive $\Delta\sigma_t$ indicates that the heavier scene introduces additional variance at frame index t . Figure 2 extracts the system's such behavior. Analyzing the graph, it is clear that, overall, Sponza Peak draws significantly more differences in every aspect, with a single distinction regarding spatial SVGF, for it revolves around $y = 0.00$ ms, signaling low variance between the scenes.

Percentile Bands. In Figure 6 it is observed a tight ceiling for the 90% of frames, though spiking mildly, especially when referring to Sponza Clean. Other than the aforementioned and well documented performance hurdles, the G-buffer stage processes geometry in an ordered fashion. In contrast, ray tracing is incoherent and unpredictable, hence the recurrent P99 spike and wide bands as observed in Figure 5.

Kernel Density Estimation. Frame distribution offers us insight regarding system consistency, and, for such task, KDEs provide the required graphical representation. Looking at the G-buffer stage (see Figure 7), we observe that the computation times slightly vary in the worst case (red). Ray tracing is, as mentioned before, notoriously incoherent, as Figure 8 depicts in the best case (blue), though it computationally peaks once and the data is narrowly scattered. In the worst case, f/s are unstable, as it portrays two outspread peaks. The SVGF stage shows a similar behavior as G-buffer.

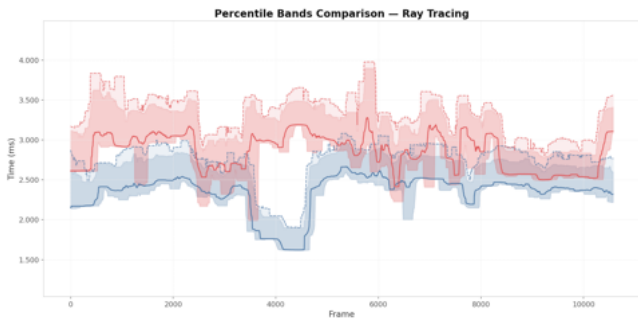


Figure 5: Percentile Bands for Ray Tracing Stage

Table 1: QoE scores (1 to 5)

Case	Std Score	95% Score
Sponza Clean	4.58	4.08
Sponza Peak	4.57	3.25

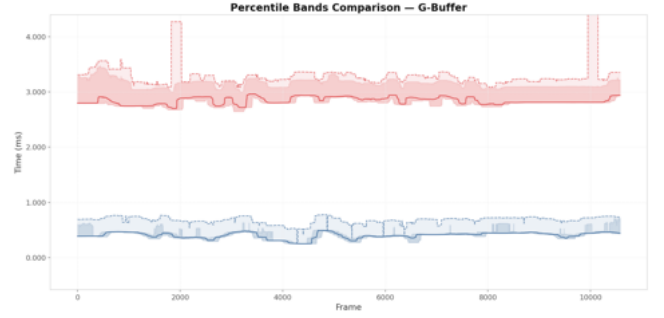


Figure 6: Percentile Bands for G-buffer Stage

Table 2: Performance scaling

Vertices	Avg GPU time	Slowdown vs min
184,583	4.530 ms	×1.00
233,797	4.770 ms	×1.05
252,539	4.680 ms	×1.03
261,451	4.640 ms	×1.02
355,049	4.760 ms	×1.05
775,280	4.990 ms	×1.10
1,323,051	5.300 ms	×1.17
2,124,745	5.280 ms	×1.17
22,963,288	7.450 ms	×1.64
23,784,937	7.400 ms	×1.63
42,846,589	9.600 ms	×2.12

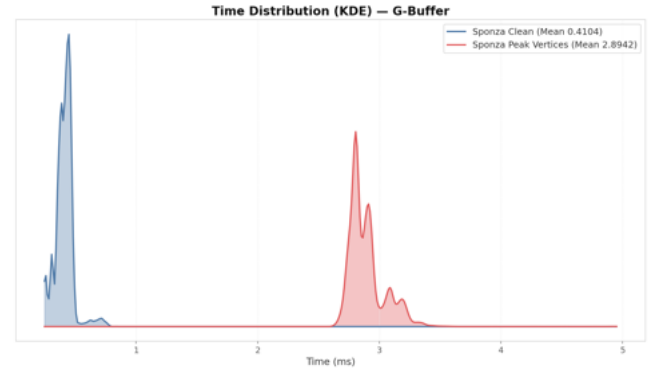


Figure 7: KDE G-buffer Stage

Frame Rate Floor. As Liu et al. [20] observed, testing for frame time standard deviation per game (see Equation 3) yields completely different R^2 values, indicating that there are other factors involved when evaluating quality of experience (QoE) against f/s variance.

$$Q = -0.16 \cdot s + 4.63 \quad (3)$$

The 95% frame rate floor is an effective predictor of QoE, as given by Equation 4.

$$Q = 0.025 \cdot s + 1.95 \quad (4)$$

where s is the normalized 95% frame rate floor percentage and Q is the QoE final score.

Testing this paper's framework results using both the aforementioned formulas, with a target f/s of 240, we arrive at the scores noted in Table 1.

Vertex Bounded Tests

The graphs before compared two scenes that are different in terms of the vertex count. Though, it has not been studied

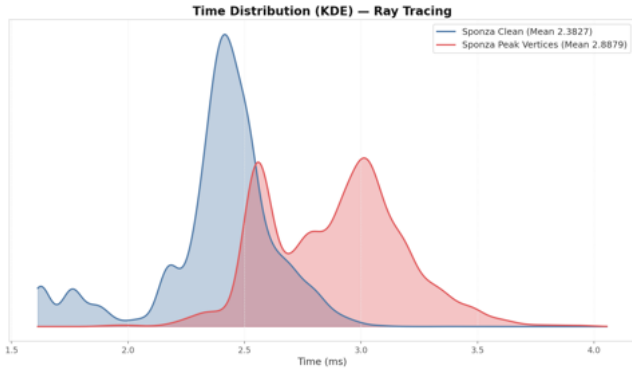


Figure 8: KDE Ray Tracing Stage

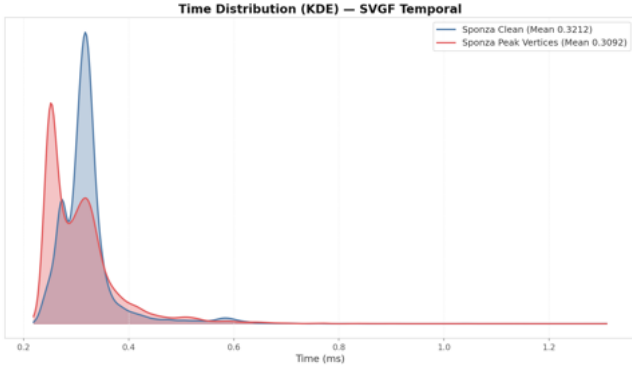


Figure 9: KDE Temporal SVGF

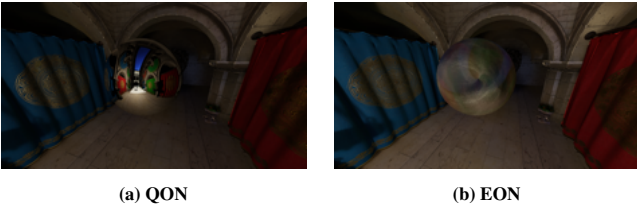


Figure 10: QON result compared to EON

how the system scales or reacts to incremental values. One takeaway is clear: the strain begins past 10 million vertices and hits that cluster around 23 million vertices, deviating from the trend logarithmic line slightly. Table 2 exemplifies the slowdown and average GPU time. At 42.8 million vertices the render time spikes to 9.6 ms (104 f/s), resulting in a 2.12x slowdown.

Diffuse Models Comparison

The results indicate that all four diffuse BRDF models perform very similarly, with average total GPU times tightly clustered between 4.82 ms and 4.90 ms. Contrary to standard expectations, the Qualitative Oren-Nayar (QON) model proved to be the most computationally efficient option in this

Table 3: Diffuse BRDF Results

Diffuse BRDF Comparison					
Model	RT	Min	Max	Avg	Total %
Disney	2.56	1.94	3.77	4.90	52.37
EON	2.58	1.63	4.99	4.87	53.04
QON	2.51	1.81	3.53	4.82	52.21
Lambert	2.52	1.92	3.51	4.86	51.78

test, recording both the lowest Frame Average (2.51 ms) and Average GPU Time (4.82 ms) (see Table 3).

Lambert performed marginally slower than QON (0.01 ms slower), while the Energy-Preserving Oren-Nayar (EON) and Disney models were the most demanding. EON, in particular, presented the highest average frame time (2.58 ms) and the largest frame duration spikes (maximum of 4.99 ms) and a reflection visual error (see Figure 10). Given that QON delivers the desired visual characteristics of the Oren-Nayar model while simultaneously being the most performant, it seems to be the optimal choice moving forward. However, it also exhibits a noticeable visual bug: the edges of some objects have disproportionately blown-out white details, which cannot be ignored and defeats the purpose of realistic simulation environments.

Conclusions

The main target of this thesis was to engineer a highly optimized, stable, and efficient ray tracing engine, which produces superior quality images under a strict 16.6 millisecond prohibition. Clear communication between several rendering passes, such as the high speed geometric processing of rasterization and physical accuracy of ray tracing, has been ingrained in the classic methods of graphics development for years. However, the difficulty always presents itself as the endless choices and trade offs required to balance realism with performance. Utilizing Liu's et al. [20] frame rate floor evaluation, clear data results regarding stability prove this framework's usability across different scenarios.

Achievements

Deferred Hybrid Architecture. This paper presented a summary of the engine's architecture and modular design, effectively isolating independent components. Such decision enhances the application's ability to scale and introduce several other improvements.

Advanced Image Reconstruction. Constrained to a 1 spp budget, the raw output inherently exhibits high-variance visual noise. To deliver a clean result, a modified version of the original SVGF [29] was implemented, separating specular from diffuse light [7] and supplementing material glossiness evaluation with roughness reliant formulas.

Real-Time Execution and Stability. The pipeline proved highly efficient under standard workloads, maintaining a 4.46 ms average (217 f/s) for the standard Sponza scene. When tested against a geometrically dense environment composed of 23 million vertices, the application maintained real time performance with a mean GPU time of 7.42 ms.

Further Development and Limitations

Light Scaling. The most critical architectural limitation exposed by the testing phase is the linear $O(N)$ scaling of the light evaluation loop inside the ray tracing stage. The natural evolution of the current system would be the adoption of Re-STIR [4]. Another choice for solving this issue would be a simpler algorithm, based on a uniform grid structure, namely ReGIR [21], necessitating an additional pass that builds the grid before evaluation.

SVGF Advancements. The original implementation that sits at the basis of this paper's signal reconstruction phase is

functional under standard conditions, though it struggles with ghosting and other visual discrepancies from reality. Future iterations provide clearer and more robust formulations of gradients that mitigate previously mentioned artifacts.

Acknowledgments

The research presented in this article has been carried out within the ChemiNova Project, which has received funding from the European Union's Horizon Europe Framework Programme under the grant agreement 101132441. Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or Horizon Europe Framework Programme. Neither the European Union nor the granting authority can be held responsible for them.

References

- [1] Abdalla G. M. Ahmed, Jing Ren, and Peter Wonka. 2022. Gaussian Blue Noise. *ACM Trans. Graph.* 41, 6, Article 260 (Nov. 2022), 15 pages. doi:10.1145/3550454.3555519
- [2] Abdalla G. M. Ahmed and Peter Wonka. 2020. Screen-space blue-noise diffusion of monte carlo sampling error via hierarchical ordering of pixels. *ACM Trans. Graph.* 39, 6, Article 244 (Nov. 2020), 15 pages. doi:10.1145/3414685.3417881
- [3] Arthur Appel. 1968. Some techniques for shading machine renderings of solids. In *Proceedings of the April 30–May 2, 1968, Spring Joint Computer Conference* (Atlantic City, New Jersey) (*AFIPS '68 (Spring)*). Association for Computing Machinery, New York, NY, USA, 37–45. doi:10.1145/1468075.1468082
- [4] Benedikt Bitterli, Chris Wyman, Matt Pharr, Peter Shirley, Aaron Lefohn, and Wojciech Jarosz. 2020. Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting. *ACM Trans. Graph.* 39, 4, Article 148 (Aug. 2020), 17 pages. doi:10.1145/3386569.3392481
- [5] Brent Burley. 2012. Physically-Based Shading at Disney. <https://api.semanticscholar.org/CorpusID:7260137>
- [6] Brent Burley. 2015. Extending the Disney BRDF to a BSDF with Integrated Sub-surface Scattering. In *SIGGRAPH Course: Physically Based Shading in Theory and Practice*. ACM. https://blog.selfshadow.com/publications/s2015-shading-course/burleys2015_pbs_disney_bsdf_notes.pdf
- [7] Chakravarthy R. Alla Chaitanya, Anton S. Kaplanyan, Christoph Schied, Marco Salvi, Aaron Lefohn, Derek Nowrouzezahrai, and Timo Aila. 2017. Interactive reconstruction of Monte Carlo image sequences using a recurrent denoising autoencoder. *ACM Trans. Graph.* 36, 4, Article 98 (July 2017), 12 pages. doi:10.1145/3072959.3073601
- [8] Per Christensen, Julian Fong, David Laur, and Dana Batali. 2006. Ray Tracing for the Movie "Car". (01 2006). doi:10.1109/RT.2006.280207
- [9] Federico Dossena and Andrea Trentini. 2022. OpenLDAT—A system for the measurement of display latency metrics. *Journal of the Society for Information Display* 30, 8 (2022), 599–608. arXiv:<https://sid.onlinelibrary.wiley.com/doi/pdf/10.1002/jsid.1104> doi:10.1002/jsid.1104
- [10] Carmelo J. Fdez-Agüera. 2019. A Multiple-Scattering Microfacet Model for Real-Time Image Based Lighting. *Journal of Computer Graphics Techniques (JCGT)* 8, 1 (22 January 2019), 45–55. <http://jcgt.org/published/0008/01/03/>
- [11] Ali Geris, Banş Çukurbası, Murat Kılınc, and Orkun Teke. 2024. Balancing performance and comfort in virtual reality: A study of FPS, latency, and batch values. *Software: Practice and Experience* 54 (06 2024), 2336–2348. doi:10.1002/spe.3356
- [12] Eric Haines and Tomas Akenine-Möller (Eds.). 2019. *Ray Tracing Gems*. Apress. <http://raytracinggems.com>.
- [13] J. H. Halton. 1960. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numer. Math.* 2, 1 (Dec. 1960), 84–90. doi:10.1007/BF01386213
- [14] Eric Heitz. 2014. Understanding the Masking-Shadowing Function in Microfacet-Based BRDFs. *Journal of Computer Graphics Techniques (JCGT)* 3, 2 (30 June 2014), 48–107. <http://jcgt.org/published/0003/02/03/>
- [15] Eric Heitz. 2018. Sampling the GGX Distribution of Visible Normals. *Journal of Computer Graphics Techniques (JCGT)* 7, 4 (30 November 2018), 1–13. <http://jcgt.org/published/0007/04/01/>
- [16] James T. Kajiya. 1986. The rendering equation. *SIGGRAPH Comput. Graph.* 20, 4 (Aug. 1986), 143–150. doi:10.1145/15886.15902
- [17] Brian Karis. 2013. Real Shading in Unreal Engine 4. In *SIGGRAPH Course: Physically Based Shading in Theory and Practice*. ACM. <https://cdn2.unrealengine.com/Resources/files/2013SiggraphPresentationsNotes-26915738.pdf>
- [18] Christopher Kulla and Alejandro Conty. 2017. Revisiting Physically Based Shading at Imageworks. In *ACM SIGGRAPH 2017 Courses*. ACM, 1–18.
- [19] Matteo P. Lanaro, Hélène Perrier, David Coeurjolly, Victor Ostromoukhov, and Alessandro Rizzi. 2019. Blue-noise sampling for human retinal cone spatial distribution modeling. arXiv:1906.05075 [cs.GR] <https://arxiv.org/abs/1906.05075>
- [20] Shengmei Liu, Atsuo Kuwahara, James J. Scovell, and Mark Claypool. 2023. The Effects of Frame Rate Variation on Game Player Quality of Experience. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) (*CHI '23*). Association for Computing Machinery, New York, NY, USA, Article 573, 10 pages. doi:10.1145/3544548.3580665
- [21] Adam Marrs, Peter Shirley, and Ingo Wald (Eds.). 2021. *Ray Tracing Gems II*. Apress. <http://raytracinggems.com/rtg2>.
- [22] Don P. Mitchell. 1991. Spectrally optimal sampling for distribution ray tracing. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '91)*. Association for Computing Machinery, New York, NY, USA, 157–164. doi:10.1145/122718.122736
- [23] Jose Emilio Muñoz Lopez and Steven Winston. 2025. Hands-on Vulkan® Ray Tracing With Dynamic Rendering. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Labs (SIGGRAPH Labs '25)*. Association for Computing Machinery, New York, NY, USA, Article 26, 1 pages. doi:10.1145/3721251.3742861
- [24] NVIDIA Corporation. 2018. NVIDIA Turing GPU Architecture Whitepaper. [Online; accessed May 29, 2026]. <https://images.nvidia.com/aem-dam/en-zz/Solutions/design-visualization/technologies/turing-architecture/NVIDIA-Turing-Architecture-Whitepaper.pdf>
- [25] NVIDIA Corporation. 2025. NVIDIA RTX Blackwell GPU Architecture. [Online; accessed May 29, 2026]. <https://images.nvidia.com/aem-dam/Solutions/geforce/blackwell/nvidia-rtx-blackwell-gpu-architecture.pdf>
- [26] Michael Oren and Shree K. Nayar. 1994. Generalization of Lambert's reflectance model. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '94)*. Association for Computing Machinery, New York, NY, USA, 239–246. doi:10.1145/192161.192213
- [27] Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2023. *Physically Based Rendering: From Theory to Implementation* (4th ed.). The MIT Press.
- [28] Jamie Portsmouth, Peter Kutz, and Stephen Hill. 2025. EON: A practical energy-preserving rough diffuse BRDF. arXiv:2410.18026 [cs.GR] <https://arxiv.org/abs/2410.18026>
- [29] Christoph Schied, Anton Kaplanyan, Chris Wyman, Anjul Patney, Chakravarthy R. Alla Chaitanya, John Burgess, Shiqiu Liu, Carsten Dachsbacher, Aaron Lefohn, and Marco Salvi. 2017. Spatiotemporal variance-guided filtering: real-time reconstruction for path-traced global illumination. In *Proceedings of High Performance Graphics* (Los Angeles, California) (*HPG '17*). Association for Computing Machinery, New York, NY, USA, Article 2, 12 pages. doi:10.1145/3105762.3105770
- [30] I.M. Sobol'. 1967. On the distribution of points in a cube and the approximate evaluation of integrals. *U. S. S. R. Comput. Math. and Math. Phys.* 7, 4 (1967), 86–112. doi:10.1016/0041-5553(67)90144-9
- [31] Ilya M. Sobol. 1994. *A primer for the Monte Carlo method*. CRC press.
- [32] Eric Veach. 1997. Robust Monte Carlo methods for light transport simulation. <https://api.semanticscholar.org/CorpusID:260812352>
- [33] Eric Veach and Leonidas J. Guibas. 1995. Optimally combining sampling techniques for Monte Carlo rendering. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '95)*. Association for Computing Machinery, New York, NY, USA, 419–428. doi:10.1145/218380.218498
- [34] Bruce Walter, Stephen R. Marschner, Hongsong Li, and Kenneth E. Torrance. 2007. Microfacet models for refraction through rough surfaces. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques* (Grenoble, France) (*EGSR '07*). Eurographics Association, Goslar, DEU, 195–206.
- [35] Turner Whitted. 1980. An improved illumination model for shaded display. *Commun. ACM* 23, 6 (June 1980), 343–349. doi:10.1145/358876.358882