Mobile-DDGI: Lightweight Probe-Based Global Illumination via Adaptive Budget Allocation

Taekgeun You
Dept. of Comp. Sci. & Eng.
Sogang University
Seoul, Korea

Woong Seo
Donghee Han
Samsung Electronics
Gyeonggi-do, Korea

Insung Ihm
Dept. of Comp. Sci. & Eng.
Sogang University
Seoul, Korea

Abstract

While Dynamic Diffuse Global Illumination (DDGI) and its extensions achieve efficient real-time global illumination on desktop hardware, their computational overhead remains a prohibitive bottleneck for resource-constrained mobile platforms. To address this limitation, we introduce an adaptive, probe-based framework that aggressively prioritizes updates within the camera frustum. Central to our approach is a novel probe importance metric that optimally distributes a strictly constrained mobile ray budget to the most perceptually significant probes. Evaluated across four distinct mobile ray tracing pipelines, our method significantly reduces the overall ray budget while robustly adapting to rapid radiometric and geometric changes without compromising visual fidelity.

ACM Reference Format:
Taekgeun You, Woong Seo, Donghee Han, and Insung Ihm. 2026. Mobile-DDGI: Lightweight Probe-Based Global Illumination via Adaptive Budget Allocation. In *Proceedings of 2026 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D '26).* ACM, New York, NY, USA, 2 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Problem specification

In an attempt to integrate ray-traced diffuse indirect illumination into real-time rendering pipelines, the method of Dynamic Diffuse Global Illumination (DDGI)[1] combined traditional irradiance probes with hardware-accelerated ray tracing and visibility-aware interpolation to compute high-quality, leak-free global illumination in fully dynamic 3D scenes. To satisfy the rigorous performance requirements of commercial game engines, subsequent research has focused on optimizing these probe-based architectures. For example, Majercik et al.[2] and the NVIDIA RTXGI-DDGI SDK[5] enhanced update efficiency and scalability by implementing cascaded tracking windows—a mechanism similar to infinite scrolling that updates only logical indices as the camera moves—alongside a probe-state classification system that minimizes the computational overhead of inactive probes. For large-scale environments, Ubisoft's approach[4] utilizes multiple camera-attached volumes with varying densities, employing a sequential, round-robin update scheme for a subset of probes to adhere to a strict per-frame computational budget. Furthermore, while Datta et al.[3] proposed an adaptive probe sampling method using Markov Chain Monte Carlo to dynamically allocate resources to changing regions, the significant initial overhead required for environmental scanning via pilot rays limits its applicability to constrained mobile devices.

These techniques have shown their efficacy on high-performance desktop GPUs. However, as current mobile GPUs still struggle to efficiently execute even classical Whitted-style ray tracing in complex 3D scenes, achieving real-time, fully dynamic global illumination on mobile platforms persists as an open challenge.

2 Our contribution

In this paper, we develop a lightweight ray-traced rendering model tailored for mobile platforms, adapting the aforementioned probe-based global illumination techniques. To efficiently represent and dynamically update indirect diffuse lighting, we utilize camera-attached multiple probe volumes with a dimension of 16×8×16 per cascade, inspired by recent optimization strategies [4]. Each volume level doubles the grid spacing of the previous one, and the scene is covered by three to four cascaded volumes depending on its scale. Furthermore, we adopt the tracking window technique [2, 5] which also updates only the logical indices of probes as the camera moves.

Unlike previous approaches, we introduce a novel strategy to distribute our strictly limited per-frame probe update budget (i.e., the total allowable ray count per frame). We allocate the dynamic update budget using the following prioritization: (1) probes newly entering the active region due to camera movement are updated first; (2) two-thirds of the remaining budget is allocated to visually critical probes, dynamically selected using a custom importance metric (described below); and (3) the residual budget is used to update the remaining probes sequentially in a round-robin fashion.

Specifically, the importance metric ρ of each probe, which dictates its dynamic update priority, is represented as the product of three weights: $\rho = w_{ref} \cdot w_{dist} \cdot w_{center}$. The first term, w_{ref} acts as a usage indicator; it is set to 1.0 if the probe is currently referenced for indirect diffuse shading, and a small positive constant w_{ref}^{min} (set to 0.1 in our experiments) otherwise, ensuring the final priority never strictly zeros out. Second, to reflect proximity to the camera, we adopt the distance heuristic proposed by Datta et al.[3]: $w_{dist} = 1.0$ if $||p - c|| < k$ and $e^{-(||p-c||-k)}$ otherwise, where p and c denote the world-space coordinates of the probe and camera, respectively. The distance threshold k is configured to be strictly larger than the grid spacing of the densest cascade. Finally, w_{center} assigns higher priority to probes located near the user's focal center. We calculate this using a standard smoothstep function: $w_{center} = \max(1.0 - \text{smoothstep}(0.7, 1.0, n_d), w_{center}^{min})$, with $w_{center}^{min} = 0.1$ in our experiments. Here, the parameter $n_d = \text{clamp}(||\mathbf{d}||, 0.0, 1.0)$

I3D '26, San Francisco, CA

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

measures the radial distance from the view center in normalized device coordinates (NDC), defined as $\mathbf{d} = (\frac{X_{clip}}{W_{clip}}, \frac{Y_{clip}}{W_{clip}})$. (Although our view-centric heuristic currently relies on the center of the view frustum, it can be readily extended to support foveated rendering by tracking dynamic gaze points via an eye tracker in VR environments.) Once evaluated, the probes are ranked by their importance ρ through a parallel radix sort. The highest-ranking probes are then aggressively selected to exhaust the allocated update budget prior to executing the final sequential round-robin updates (Figure 2(a)).

3 Results

To rigorously evaluate the proposed method, we implemented a baseline Whitted-style ray tracer using the Vulkan API on a Samsung Exynos 2600 mobile application processor equipped with an Xclipse 960 GPU. To assess its viability across diverse rendering environments, we developed two foundational architectures: a full ray tracing (FRT) pipeline and a hybrid ray tracing (HRT) pipeline that leverages deferred rendering to resolve primary visibility. Furthermore, on the target mobile GPU, incorporating hardware ray tracing units based on the AMD RDNA architecture, we implemented two distinct ray traversal strategies: The first utilizes the dedicated hardware ray tracing pipeline (via Vulkan's VK_KHR_ray_tracing_pipeline extension, denoted as RT), while the second employs inline ray queries within the standard graphics pipeline (via the VK_KHR_ray_query extension, denoted as RQ). Building upon these four combinations of baseline renderers, we integrated our proposed DDGI framework.

In our experiments, we implemented three configurations of lightweight probe-based GI on a mobile platform: The first method, denoted as Static Volume, deploys a fixed number of static probe volumes covering the entire scene (one for SPONZA, and three for the others). In each frame, probes are classified into *Active* and *Inactive* states following the approach of [2, 5]. We then update the active probes by tracing 128 rays per probe, while inactive probes are updated with 32 rays. The second method, denoted as Moving Volume, employs camera-attached, multi-resolution cascaded volumes of dimension $16 \times 8 \times 16$ (three for SPONZA, and four for the others). It applies the same binary state classification as the first method, thereby drastically reducing the total number of updated probes. The third configuration represents our proposed approach (Ours), which allocates a strict update budget of 200 probes per $16 \times 8 \times 16$ cascade. Consequently, in a setup with four cascaded volumes comprising 8,192 probes in total, for example, only 800 probes are adaptively updated per frame as described above.

Figure 1 compares the rendering performance on the tested mobile platform across the four example scenes. To prevent thermal throttling and ensure consistent measurements, the GPU clock frequency was strictly capped at 836 MHz (~85%). For each scene, rendering times were averaged across four carefully selected camera views, rendered at an FHD+ resolution (2340×1080). Overall, our approach (Ours) achieved significant speedups over the baseline implementations across all four ray tracing configurations. Leveraging the dedicated hardware ray tracing units, the pipelines employing the RT extension (FRT-RT and HRT-RT) outperformed those using inline ray queries (FRT-RQ and HRT-RQ) by 8.7% to 21.4%. Consequently, our lightweight DDGI framework achieved rendering frame rates ranging from 26.8 fps / 30.7 fps (FRT-RT / HRT-RT for BISTRO) to 51.5 fps / 64.4 fps (FRT-RT / HRT-RT for SPONZA). Despite this enhanced performance, the visual degradation was strictly bounded to acceptable levels, yielding rendering results highly comparable to the Static Volume baseline (Figure 2(b)). Please refer to the supplementary video for further visual evaluation.

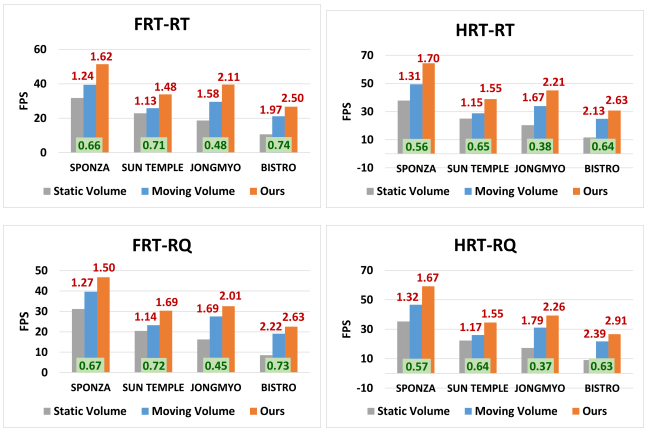
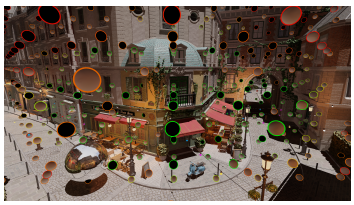


Figure 1: Performance evaluation of three probe-based GI methods on SPONZA, SUN TEMPLE, JONGMYO, **and** BISTRO **(comprising 304K, 608K, 1,570K, and 2,831K triangles, respectively). The values highlighted in red represent the speedups achieved by the two dynamic volume methods over the** Static Volume **baseline, while the values in green indicate the frame rate degradation over the Whitted-style ray tracer.**



(a) Dynamic update priority (BISTRO)

Scene	MV	Ours
SPONZA	33.5	33.3
SUN TEMP.	28.5	28.5
JONGMYO	29.5	28.5
BISTRO	29.6	29.3

(b) HRT-RQ

Figure 2: (a) The estimated probe importance metric (color-coded from 0.0 (red) to 1.0 (green)). (b) Quantitative comparison of visual quality for the two optimization methods, with the Static Volume **baseline serving as the ground truth (PSNR).**

Acknowledgments

This work was supported by Samsung Electronics Co., Ltd (IO251222-14921-01).

References

[1] Majercik et al. 2019. Dynamic Diffuse Global Illumination with Ray-Traced Irradiance Fields. *Journal of Computer Graphics Techniques* 8, 2 (2019), 1–30.
[2] Majercik et al. 2021. Scaling Probe-Based Real-Time Dynamic Global Illumination for Production. *Journal of Computer Graphics Techniques* 10, 2 (2021), 1–29.
[3] S. Datta et al. 2023. Adaptive Dynamic Global Illumination. doi:10.48550/ARXIV.2301.05125
[4] L. Leblanc and M. Conte. 2025. Ray Tracing the World of Assassin's Creed Shadows. Presentation at SIGGRAPH 2025 Advances in Real-Time Rendering in Games course, Vancouver, Canada. https://doi.org/10.1145/3721241.3744989
[5] A. Marrs. 2023. *RTX Global Illumination (v1.3.7).* Retrieved March 1, 2026 from https://github.com/NVIDIAGameWorks/RTXGI-DDGI