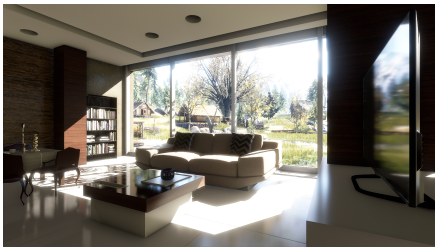
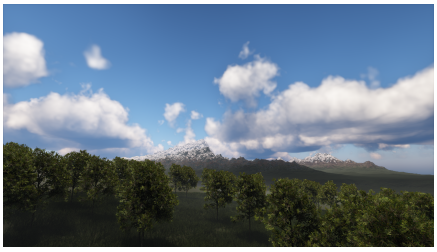
Practical Dynamic Lighting for Large-Scale Game Environments

Kyungjoon Cho
Pearl Abyss

Kwanghyeon Go
Pearl Abyss

Daeil Kim
Pearl Abyss

ABSTRACT

Dynamic lighting techniques are vital in giving artists rapid feedback and reducing iteration time. In this technical postmortem, we will talk about dynamic lighting techniques for large-scale game environments that reflect changes in the time of day and include many dynamic lights. Moreover, a unified atmospheric scattering technique with clouds will also be discussed.

CCS CONCEPTS

• **Computing methodologies** → Computer graphics; Rendering.

KEYWORDS

Additional Key Words and Phrases: Real-time rendering, global illumination, atmospheric scattering

ACM Reference Format:
Kyungjoon Cho, Kwanghyeon Go, and Daeil Kim. 2019. Practical Dynamic Lighting for Large-Scale Game Environments. In *Proceedings of SIGGRAPH '19 Talks*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3306307.3328189

SIGGRAPH '19 Talks, July 28 - August 01, 2019, Los Angeles, CA, USA

ACM ISBN 978-1-4503-6317-4/19/07.
https://doi.org/10.1145/3306307.3328189

1 DIRECT LIGHTING

One of the key elements to achieving plausible visual quality in real-time interactive games is the number of lights. For performance reasons, the number of lights was tightly restricted in our game engine. As an alternative, clustered shading[Persson 2013] was chosen as the direct lighting solution so that artists could use more dynamic light sources. This is suitable for handling long range scenery with many lights. It was also used to render decals and volume fog, which will be discussed later. Nevertheless, with many dynamic lights, it is difficult to handle shadowing while also maintaining good performance. To avoid shadowing all the light sources and make the lighting shapes much more realistic, IES profiles and light clipping geometries were applied to each of the lights.

2 INDIRECT LIGHTING

Indirect illumination is the most challenging part of real-time rendering. For fully dynamic environments and fast iteration, all techniques based on precomputed information were excluded from our consideration. The most common approaches to represent indirect illumination for real time rendering are probes and voxels. Probes are efficient, but require precomputation for visibility and suffer from light leaking; lack of specular reflections is another weakness. Voxels, on the other hand, require a substantial amount of sampling and generation time, but are capable of both diffuse and specular illuminations and produce more plausible results.

An earlier attempt for voxel-based indirect illumination was SVOGI[Crassin et al. 2011]. However, since its structure is not efficient for modern GPUs, it was not widely adopted as a GI solution. Another attempt was NVIDIA's VXGI[Panteleev 2014]. VXGI introduced 3D clipmaps, a GPU friendly structure as opposed to Sparse Voxel Octree, but its implementation was a bit too heavy for real-time applications.

2.1 Voxelization

Our approach adopts 3D clipmaps of VXGI, while reducing bandwidth and memory usage significantly by presenting voxels omnidirectionally. Since the omnidirectional voxels have some limitations such as light leaks and self-occlusions, we added directionality to the voxels based on the 2nd band of spherical harmonics, which resolved the issues accordingly. For voxelization, we used the screen

space approach[Gaitatzes and Papaioannou 2013], which is far more efficient than geometry-based rasterization. Progressive Voxelization performs great but its running time increases rapidly with the size of voxels, which does not fit the requirements of large-scale environments. Instead, we build a list of injections using only screen space G-Buffers and lighting results, where both opacity and emittance are injected at the same time. By using parallel reduction on the compute pipeline, the problem of overdraw on the same voxel reduced greatly. In the cleanup phase, we simply mask dynamic objects on injection and build a list of dynamic injections. The actual cleanup process is a simple subtraction of voxels in the list.

2.2 Diffuse Indirect Illumination

Diffuse indirect illumination uses 7 cones to cover a hemispherical surface. Each cone traces about 8 to 12 sphere samples depending on the configuration. Diffuse cones are traced only once for every 4 pixels, and reconstructed to full resolution temporally.

2.3 Ambient Occlusion

For more fine-grained ambient occlusions, we adopted the popular HBAO[Bavoil et al. 2008]. For large scale occlusions, we first take shadow maps from uniformly distributed directions, and then use the shadow maps to determine the visibility of the sky when shading pixels.

2.4 Specular Indirect Illumination

For specular reflections, a larger sphere radius yields partial occlusion on thin walls, resulting in hollow artifacts. Furthermore, cone traced results do not fit the specular BRDF well, especially at grazing angles. To overcome this, we chose to implement raymarching instead of sphere cone tracing. The rays are distributed randomly, weighted by the distribution of matching BRDF(GGX), but only 1 ray per pixel is taken on a single frame for the real-time constraint. This gave us an extremely noisy output for rough surfaces, so some sort of noise reduction filter was required. NVIDIA demonstrated a good example in their RTX demo[Schied et al. 2017], but the GPU time demand was too heavy for real time applications. Instead, we adopted the simpler À-Trous wavelet transform[Dammertz et al. 2010] with our heuristic weight function, which takes into account surface roughness to prevent overly blurry images. Some initial step offset is required to avoid quantization errors caused by voxelization, which results in band-shaped seams at concave edges. To overcome this and to achieve sharper reflections, we combined the Screen Space Reflections[Stachowiak and Uludag 2015] with voxel raymarching. Since both techniques share the same importance samples and denoising filter, specular reflection seams are hardly noticeable on most surfaces, except for highly glossy surfaces. When both results are unavailable for a surface, we use the environment cubemap as a fallback, which is generated in real-time.

3 ATMOSPHERE

We considered several approaches to support a realistic aerial representation. Most related works treat atmosphere, light shaft, clouds, clouds shadow and fog separately, however this can cause visual inconsistency among the atmospheric elements. Additionally, if raymarching were to be used for the light shaft, there would be duplicate raymarching, along with the one for clouds. That is why we suggest a unified solution. During the process of raymarching for atmospheric scattering, cloud and fog densities are applied to scattering equations, then its extinction affects the next raymarching steps. Accumulated cloud density toward the sun is computed prior to the raymarching steps. Shadowing is calculated using cascaded shadow maps and the accumulated clouds density during the process so that it naturally produces light shaft and cloud shadow from both shadow maps and cloud extinction. A similar approach to the one used in Horizon Zero Dawn[Schneider 2016] was applied to represent cloud shapes. For tackling relatively short-range volume fog with dynamic lights, the screen space aligned voxel solution[Wronski 2014] was applied. At the light injection stage, the clustered shading data set which was mentioned earlier, is used. Sun and cloud shadows are also applied to the volume fog in the same way as the atmospheric scattering raymarching process.

REFERENCES

Louis Bavoil, Miguel Sainz, and Rouslan Dimitrov. 2008. Image-space horizon-based ambient occlusion. In *ACM SIGGRAPH 2008 talks*. ACM, 22.

Cyril Crassin, Fabrice Neyret, Miguel Sainz, Simon Green, and Elmar Eisemann. 2011. Interactive indirect illumination using voxel cone tracing. In *Computer Graphics Forum*, Vol. 30. Wiley Online Library, 1921–1930.

Holger Dammertz, Daniel Sewtz, Johannes Hanika, and Hendrik Lensch. 2010. Edge-avoiding À-Trous wavelet transform for fast global illumination filtering. In *Proceedings of the Conference on High Performance Graphics*. Eurographics Association, 67–75.

Athanasios Gaitatzes and Georgios Papaioannou. 2013. Progressive Screen-Space Multichannel Surface Voxelization. *GPU Pro 4: Advanced Rendering Techniques* 4 (2013), 137.

Alexey Panteleev. 2014. Practical real-time voxel-based global illumination for current GPUs. In *ACM SIGGRAPH 2014 presentations*.

Emil Persson. 2013. Practical clustered shading. *SIGGRAPH Course: Advances in Real-Time Rendering in Games* (2013).

Christoph Schied, Anton Kaplanyan, Chris Wyman, Anjul Patney, Chakravarty 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*. ACM, 2.

Andrew Schneider. 2016. Real-time volumetric cloudscapes. *GPU Pro 7: Advanced Rendering Techniques* 97 (2016).

Tomasz Stachowiak and Yasin Uludag. 2015. Stochastic screen-space reflections. *ACM SIGGRAPH Courses 2015: Advances in Real-Time Renderingin Games* (2015).

Bartlomiej Wronski. 2014. Volumetric Fog: Unified Compute Shader-Based Solution to Atmospheric Scattering. In *ACM SIGGRAPH*.