High-Performance Ray Tracing for Nonimaging Applications

Paul Manstetten[1], Luiz Felipe Aguinsky[1], Siegfried Selberherr[2], and Josef Weinbub[1]
[1]Christian Doppler Laboratory for High Performance TCAD at the
[2]Institute for Microelectronics, TU Wien, Austria
manstetten@iue.tuwien.ac.at

Today, ray tracing is most commonly associated with computer graphics applications where the objective is to render an image from a virtual scene. Such a scene is typically a collection of objects represented mathematically. The image is generated by tracing rays originating from a viewport into the scene. The intersections of these original rays with the objects in the scene commonly generate subsequent rays to approximate the rendering equation, generating an equilibrium of the radiance in the scene. The radiance towards the viewport finally defines the values of the pixels in the rendered image [1].

Nonimaging applications which use ray tracing as a computational method can potentially profit from the algorithmic advances and optimized computational performance of ray tracing engines developed for rendering, like Embree [2], Optix Prime [3], and OpenVDB [4]. Ray tracing can, for example, be applied in the simulation of radiative heat transport [5], nonimaging optics [6], [7], and particle transport [8]. In fact, for advanced semiconductor process simulations, the computational performance of a feature-scale etching or deposition process is largely determined by the ray tracing performance [9].

We assess the performance of the above-mentioned ray tracing engines using a generic test case with a spatially distributed incoherent ray workload, reproducing a common situation for nonimaging applications. Furthermore, we discuss the implications arising when a ray tracing engine is integrated into an existing simulation framework by the example of a topography simulator.

Acknowledgments. The financial support by the Austrian Federal Ministry of Science, Research and Economy, and the National Foundation for Research, Technology and Development is gratefully acknowledged.

[1] M. Pharr *et al.*, *Physically Based Rendering* (Morgan Kaufmann, 2016).
[2] Embree. https://embree.github.io
[3] Optix Prime. https://developer.nvidia.com/optix
[4] OpenVDB. https://github.com/AcademySoftwareFoundation/openvdb
[5] J. R. Mahan., *Radiation Heat Transfer: A Statistical Approach* (Wiley, 2002).
[6] OpticsStudio. https://www.zemax.com/products/opticstudio
[7] S. Scharring *et al.*, Opt. Eng. **56**, 011007 (2017).
[8] ViennaTS. https://viennats.github.io
[9] P. Manstetten *et al.*, Procedia Comp. Sci. **108**, 245 (2017).