**Analysis of Photonic Crystal Wave Containment**

S. David Lively

Southern Methodist University

*Periodic dielectric structures may be used to direct and contain electromagnetic waves. The ability of a crystal to affect wave behavior depends on the several factors, including the shape, scale and distribution of dielectrics as well as the relative wavelength of an incident wave. Simulation techniques such as FDTD may be used to model periodic dielectric structures and their response to different wavelengths in order test suitability of a given structure for a particular application. We present an analysis of three structures, their resonant frequencies, and software modifications to enable efficient simulation of such structures.*

The Finite Difference Time Domain method is a popular, robust method to simulate and analyze the propagation of electromagnetic waves within different media and geometries. While typically applied to non-periodic structures such as dielectric slabs and complex circuits, the algorithm may also be used to simulate periodic structures.

For this document, the existing GoLightly simulator was modified to facilitate periodic structures. The goals of such modifications include:

1. Unity3D FDTD implementation
2. Unit cell-based simulation definitions.
3. Tiling of image-based crystal unit cells.
4. Mathematically-defined periodic structures
5. Multi-resolution simulation
6. Source wavelength sweep
7. Interactive monitor charting and analysis
8. Summary data export

These modifications are detailed in the following sections.

Unity3D Implementation

The most-recent GoLightly implementation, created in 2015, utilized a combination of C++, NVIDIA CUDA and OpenGL. While effective at the time it was created, several factors indicated that a reimplementation was required for this project. The available hardware – specifically, a 2019 16” MacBook pro with AMD GPU – presented several challenges.

CUDA (Compute Unified Device Architecture), the GPGPU development ecosystem developed by NVIDIA, requires access to an NVIDIA GPU. In addition, the CUDA version used by GoLightly is nearly a decade old. Updating the application to use a more modern CUDA version would require substantial work which would not benefit the task at hand. In addition, CUDA kernels are based on the Cg shader language, which is very similar to HLSL (DirectX High Level Shader Language).

Given that available hardware consisted of a MacBook Pro with an AMD GPU, it seemed logical to port the CUDA kernels to HLSL Compute Shader language. This is platform-agnostic system that may be cross-compiled for NVIDIA, AMD and Intel GPUs, and, through Unity3D, is available for all modern operating systems.

GoLightly’s user interface requires a complex CUDA/OpenGL interoperation mechanism. MacOS support for OpenGL has been deprecated by Apple. While it is still possible to use OpenGL on MacOS, implementation of the FDTD kernels in HLSL precludes use of OpenGL. While the FDTD kernels could have been re-written in GLSL (OpenGL Shader Language), the similarities between CUDA and HLSL made HLSL a more suitable platform.

Modern game development platforms such as Unity3D are well-suited for the creation of cross-platform GPU-centric applications with complex user interfaces. Several factors informed the decision to reimplement GoLightly in Unity3D.

* Cross-compilation support of HLSL to Metal, the preferred graphics API on MacOS
* Rich library of user interface software components
* Rapid development iteration cycle.
* Support for MacOS, Linux and Windows

**References**

**1**Pollock, C. R., and Michal Lipson. *Integrated Photonics* . Boston: Kluwer Academic , 2003. Print.