**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

The decision to reimplement GoLightly in Unity3D yielded many benefits including reduced software complexity. The C++ GoLightly implementation contained a substantial amount of code purely to manage the user interface. Unity3D provides much of this functionality by default, leading to the exclusion of many thousands of lines of source code.

Once the Unity3D implementation reached feature parity with the C++ version, some modifications were required to facilitate support for periodic structures. The prior version relied on the scale-invariant nature of Maxwell’s equations that describe the physical behavior of electromagnetic waves.

For instance, the FDTD algorithm defines the size of a Yee cell as 1/10th of the wavelength of the highest frequency source in the simulation. All physical constants such as **u0**, **eps0**, and **c** are defined as 1. While these assumptions simplify some calculations in FDTD, they present issues when dealing with structures defined in real physical units. Since analysis of the dielectric lattice requires a frequency sweep to identify resonant wavelengths, it is convenient to define the Yee grid in terms of feature size rather than wavelength[[1]](#footnote-1).

To that end, we define the normal wavelength of the simulation in terms of the smallest feature size in the simulation. For the first structure to be analyzed[[2]](#footnote-2), the smallest feature is a rod with diameter 480nm. We define the normal wavelength[[3]](#footnote-3) of the simulation to be 1/10 of that value, or 48nm. The frequency scanning portion of the simulation may then use wavelengths as low as 10\*48um= 480nm. (For each of the lattice simulations to be performed, the minimum wavelength of interest is on the order of 600nm.)

Having determined a rational way to convert physical units to the unitless parameter space used by GoLightly, a few other modifications required.

GoLightly uses an image-based model definition format to eliminate any dependence on complex CAD applications or scripting. While well-suited to represent complex, arbitrary dielectric geometries, investigation of the lattices of interest required a certain flexibility not easily adapted to the image-based model definition. Namely, it was required to repeatedly adjust the Yee grid resolution when exploring potential solutions. This required the creation of many versions of the lattice unit cell tiles.

Rather than performing this process manually, we wrote a Unity3D script to generate the unit cells which were then exported to the PNG model definition format used by GoLightly. At that point, we determined that generating a tile, exporting it to a PNG image, reloading and decoding that image and using it to populate the dielectric material array used by the FDTD kernels was, to be frank, redundant, inefficient and suboptimal. Since the software to generate the models had already been created, we chose to use it to directly populate the FDTD dielectric material array.

**References**

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

1. Choosing a minimum wavelength that is proportional the smallest feature of interest would also have been valid. [↑](#footnote-ref-1)
2. A square lattice of rods with spacing of 1.2um, diameter 480um. [↑](#footnote-ref-2)
3. FDTD assumes that the size of a Yee cell is at most 1/10th of the shortest wavelength in the simulation. [↑](#footnote-ref-3)