Optical Photon Simulation on GPUs

Simon C. Blyth

May 10, 2016

Contents

1 Overview

During the past year I have focussed on solving the problem of optical photon simulation in neutrino detectors through the development of a package I have named Opticks. Opticks works together with the Geant4

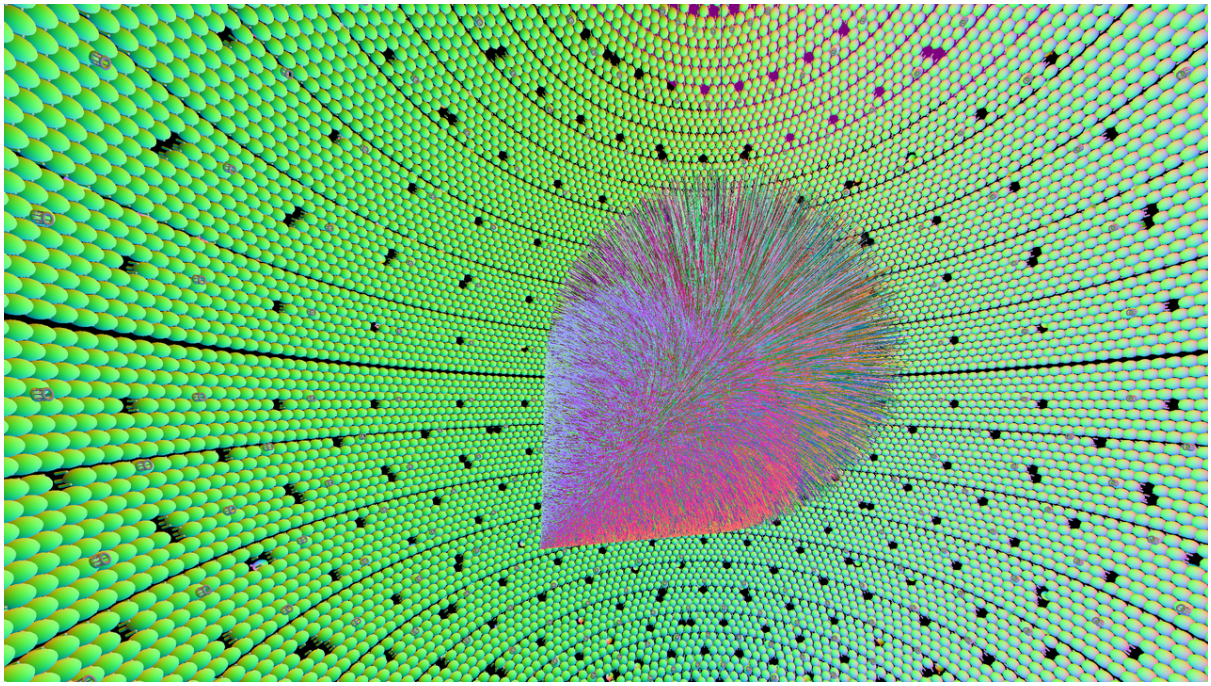

Figure 1: Simulated Cerenkov and scintillations photons from a 100 GeV muon travelling across the JUNO antineutrino detector viewed from inside the spherical scintillator. Primary particles are simulated by Geant4, generation "steps" of the primaries are transferred to the GPU and photons are generated, propagated and visualized all on the GPU. Photons that hit PMTs are returned back to Geant4 to be included into standard hit collections. Photon colors indicate the polarization direction.

toolkit, it replaces the Geant4 optical photon simulation with an equivalent optical simulation implemented on the GPU and accelerated by the NVIDIA OptiX ray tracing engine. Validations by comparison of distributions from Geant4+Opticks with Geant4 alone are ongoing. Simple geometries such as a single photomultiplier tube in mineral oil have been validated and demonstrate performance speedup factors of 200x relative to Geant4 with a mobile GPU with only 348 cores. Performance factors exceeding 1000x are expected with workstation GPUs by extrapolation of CUDA core counts. Also, CPU memory contraints for the optical photons are eliminated.

In addition to development I have started the process of marketing Opticks via numerous collaboration, group and conference presentations as well as demonstration videos on video sharing websites and I have accepted an invitation to present a course on Opticks at the 2016 Weihai Summer School organized by Shandong University. Presentation slides and videos including a video of my 25 minute NVIDIA GPU Technology Conference talk are accessible from http://simoncblyth.bitbucket.org

2 Introduction

2.1 Optical Photon Simulation Problem

A detailed understanding of the generation and propagation of optical photons is vital to the design, operation and analysis of photomultiplier based neutrino detectors such as the Daya Bay and JUNO experiments. Detailed detector simulations are essential to develop such an understanding and provides crucial inputs to data analysis. The Geant4 based Daya Bay simulation was found to expend more than 95% of CPU time propagating optical photons during simulation of cosmic muon background events. The problem with Geant4 optical photon simulation can be summarised as poor ray tracing performance caused by a geometry model which inhibits the use of modern acceleration techniques and hardware. More detailed descriptions of neutrino detection via optical photons and the optical photon simulation problem are included in my 2015 report.

2.2 Solving the Optical Photon Simulation Problem

As optical photons in neutrino detectors can be considered to be produced by only the scintillation and Cerenkov processes and yield only hits on photomultiplier tubes it is straightforward to integrate an external optical photon simulation with a Geant4 simulation of all other particles. However in order to create an external GPU simulation consistent with the Geant4 simulation it is necessary to duplicate the Geant4 context on the GPU. In order to do this I developed the G4DAE geometry exporter and the G4DAEOpticks runtime bridge. Details on these packages are provided in my 2015 report.

3 Opticks Development

The bulk of the development of Opticks was done in the past 12 months. Opticks is a C++/CUDA package using geometry intersection acceleration from the NVIDIA OptiX ray tracing engine. NVIDIA OptiX provides state-of-the-art GPU acceleration of ray geometry intersection with efficient GPU/multi-GPU utilization and regular updates tuned to fully exploit new GPU architectures. Further details on NVIDIA OptiX are provided in my 2015 report. Opticks is organized according to dependencies into 15 packages which can be grouped into four categories with responsibilities listed below.

- *Infrastructure* : configuration, array handling/persistency, network transport
- *Geometry* : parsing G4DAE exports and conversion into form suitable for upload to GPU
- *GPU Operations* : Optical photon generation and propagation in CUDA programs, Thrust photon indexing, OptiX ray tracing, OpenGL visualization.
- *Steering* : High level control

The below sections highlight some of the most important and most challenging aspects of the development.

3.1 Handling Very Large Detector Geometries

The JUNO geometry with 19,000 20 inch PMTs and 34,000 3 inch PMTs in tesselated form corresponds to 90 million triangles. Such a large geometry does not fit into the 2 GB memory of the development GPU forcing use of more memory efficient representations. I developed a technique that automatically finds repeated portions of the geometry tree based on ancestor transform digests. This yields geometry subtrees together with all transforms of that subtree allowing OptiX and OpenGL instancing to be used where repeated information is avoided by using a single definition of the repeated geometry together with 4x4 homogenous matrix transforms representing the position and orientation of each instance of that geometry. Such techniques allowed a memory reduction factor of approximately 900 for the JUNO geometry which is sufficient to fit into the available GPU memory.

Parsing the G4DAE XML of the JUNO geometry takes several minutes, to avoid this expense at every startup geometry caching was developed based on the NumPy serialization described in my 2015 report. This allows startup time to be reduced to several seconds only during which geometry buffers are read and directly uploaded to the GPU. The approach developed to convert Geant4 detector geometries exported with G4DAE into a GPU representation using my fork of the Assimp project is described in my 2015 report.

3.2 Handling Very Large Photon Step Records

During development and validation it is essential to record every step of the optical photons, to allow step by step distribution comparisons. Simulation of Daya Bay cosmic muon events can typically require several millions of optical photons and of order ten steps for each are of interest. Storing position, direction, time, wavelength and history flags would typically take 512 bits per step, with 30M steps this corresponds to almost 2 GB almost filling the GPU memory with just photon step records. Instead aggressive compression was adopted using the known domains of position, time, wavelength and polarization reducing the storage per step down to 128 bits, corresponding to a total of almost 500MB. The compression is applied only to the recording of the photon steps, the final photon parameters are retained at full precision.

3.3 Optical Photon Generation and Propagation

Optical photon generation from scintillation and Cerenkov processes using buffers of generation step parameters collected from Geant4 are implemented in OptiX CUDA programs, as are the propagation of photons. The OptiX CUDA implementations are all based on the corresponding Geant4 implementations adapted for a different geometry model and environment. The implementations make use of efficient computation techniques, for example for Fresnel reflection/transmission the Maxwell's equation boundary condition matching technique from Geant4 is adopted but with transcendental functions replaced using approaches from the ray tracing literature.

Scintillation and Cerenkov generation step parameters define a line segment and the number of photons to create along it together with other parameters used to generate the appropriate angular and wavelength distributions. Seeding the photon buffer with generation step buffer indices with the appropriate number of repeats for the number of photons for each generation step is straightforwardly done on the CPU. However this simple approach introduces large CPU memory requirements and necessitates the transfer of large photon buffers to the GPU. Instead an entirely GPU side seeding approach was developed using the CUDA Thrust library that allows the photons to become GPU resident. With this approach the millions of optical photons can be generated with zero CPU memory requirements, all allocation for the photons being done on the GPU and only photons that hit photomultiplier tubes need to have CPU side allocations. Development of interoperation machinery allowing the GPU libraries: OptiX, CUDA, Thrust and OpenGL to share GPU photon buffers was necessary for the photons to become entirely GPU resident; this work was the most technically challenging aspect of Opticks so far.

3.4 Tesselated Geometry Mesh Fixing

A boundary based geometry model is used that associates every primitive with inner and outer material indices and optionally inner and outer surface indices. The simulation intersects a ray with the geometry to determine the material index and uses this together with the photon wavelength to lookup interpolated wavelength dependent material properties from a float4 texture providing in a single highly optimized GPU texture lookup the refractive index, absorption length, scattering length and reemission probability.

A bug in the Geant4 G4Polyhedron tesselation algorithm was found due to the resulting incorrect material assignments. In some cases union solids are tesselated as multiple cleaved meshes resulting in extraneous close parallel faces. A workaround was developed that identifies extraneous faces, deletes them from the mesh and stitches the split mesh together. This mesh fixing was developed based on the open source OpenMesh project.

3.5 Analytic Photomultipler Tube Geometry

NVIDIA OptiX provides only the GPU acceleration of geometrical intersection, not the intersection itself. I implemented ray geometry intersection initially only for triangle primitives using a well known optimized algorithm. During validation of single photomultiplier tubes the facetted nature of the tesselated geometry was found to be too unrealistic.

To improve realism and efficiency I developed an analytic description of the Daya Bay photomultiplier tube geometry. Geant4 geometry can be considered to be comprised of a tree of boolean operations such as unions and intersections acting upon primitives such as spheres and cylinders. In order to avoid the implementation of full constructive solid geometry (CSG) tree boolean processing on the GPU I developed a partitioning approach that exploits the rotational symmetry of the photomultiplier tube allowing the tree representation to be simplified into a list of partial primitives by splitting at the geometrical intersections of the basis volumes. The five solids representing the Daya Bay photomultiplier tube are partitioned into a total of twelve single primitive parts which are copied to the GPU instead of the close to 3000 triangles of the tesselated geometry. These parts together with ray geometry intersection CUDA OptiX programs for cylinder and partial sphere primitives provide the analytic geometry representation.

Fortunately it is straightforward within OptiX to combine tesselated geometries together with larger primitive analytic geometry allowing only the parts of the geometry along the critical optical path of the neutrino detector to be represented with the more realistic analytic primitives whilst leaving most of the geometry in tesselated form.

3.6 Indexing Histories of Millions of Photons

Each step of an optical photon has a flag recording whether reflection, transmission, scattering, reemission or absorption occurred. In order to check the simulated results it is essential to categorize the sequences of these step flags.

My initial CPU attempt at indexing the histories of 3 million photons with up to 10 steps per photon was found to take almost a minute. This is unreasonable given that the GPU simulation can take fractions of a second. Thus I developed a GPU indexing algorithm based on CUDA Thrust sparse histogram sorting that allowed photon history indexing to be achieved in fractions of a second. In addition to this speedup it is also advantageous that copying the large step buffer from the GPU to the CPU is avoided.

Using OpenGL/OptiX/Thrust/CUDA interoperation techniques I developed a graphical interface to the photon index allowing the interactive selection and visualization according to the the flag or material histories of the photons. Visualization of photon categories has proved invaluable during debugging of the optical photon simulation as the categories precisely correspond to different paths through the code, causing problems to typically focus within particular categories.

3.7 Visualizing Millions of Photons

Opticks uses modern OpenGL shader GPU based rendering techniques to provide animated photon visualizations including interactive time scrubbing that treats the event time as an input to the rendering. In addition camera viewpoint animation is supported allowing for example the camera position to be panned along a line parallel to that of the originating muon at appropriate event times to yield an illustrative view of the propagation.

These unprecedented within the field visualizations are enabled by GPU performance and the efficient sharing of GPU buffers due to the interoperation capabilities of CUDA, Thrust, OpenGL and OptiX. Screen capture videos of the visualizations have proved useful to illustrate the principals of neutrino detection to diverse audiences and are available on video sharing web sites linked from http://simoncblyth.bitbucket.org.

3.8 Tests with Dynamic Geometry and Photon Sources

Debugging the optical simulation was found be benefit greatly from tests with very simple geometries such as boxes and prisms together with simple photon sources with various shapes, angular distributions and wavelength distributions. To provide a fast development cycle while doing such tests requires the ability to dynamically construct light sources and geometry with boundaries including various materials and surfaces without having to recreate geometry caches for each configuration.

In order to support this dynamic functionality a substantial refactoring was done to the geometry representation. Boundaries on the CPU side are now represented simply as inner and outer material indices and optionally inner and outer surface indices which are only converted into the GPU boundary texture just prior to simulation. Such a simple boundary representation is easily extended allowing simple geometries and light sources specified on the command line to be simulated and visualized within a few seconds. This test system was used for example to compare the simulated deviation angle of a prism against analytic expectatations of Snell's law at two boundararies related by the prism geometry. Good agreement was found.

3.9 Validation of Test Geometries

The Opticks CfG4 package was implemented to provide comparisons of Opticks simulation results with Geant4 using simple test geometries and light sources specified on the command line. Geant4 particle generators and geometries including materials and surfaces are constructed from the GPU geometry cache configured by the same command line arguments as Opticks. A Geant4 photon step recorder was implemented that collects the Geant4 photon step informatin within Opticks format buffers, allowing Opticks visualization and analysis tools to be used on the Geant4 simulated events in exactly the same manner as the Opticks simulated events.

Initial validations are done by comparing photon counts within each history sequence categories and material sequence categories. Subsequently chi-squared comparisons of all photon distributions such as position, time, direction, polarization and wavelength at every step of the propagation within each of the sequence categories

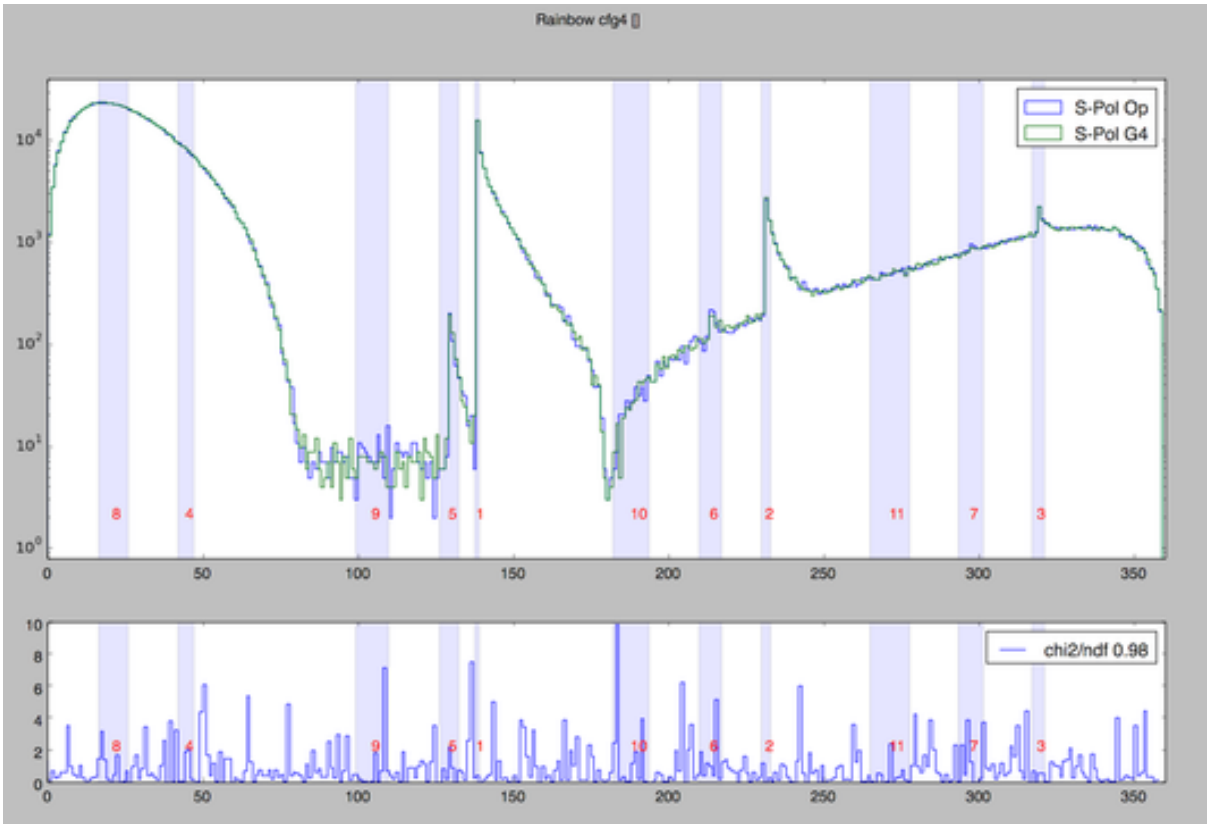

Figure 2: Opticks/Geant4 comparison of the distribution of deviation angles(degrees) of 1M parallel monochromatic photons incident on a spherical droplet of water. The numbered bands show expectations of the first 11 rainbows angles across the visible range. As indicated by the chi-squared lower plot, good agreement is achieved. The Opticks simulation took 0.25s, the Geant4 simulation 50s on Macbook Pro laptop with mobile GPU. Performance on workstation machines with multiple GPUs has been shown to be a further 20x faster than with a mobile GPU leading to expected Opticks performance greater than 1000x Geant4.

are made. Such segmented comparisons were found to rapidly identify problems as the categories correspond to distinct paths through the simulation implementation.

A single spherical droplet of water illuminated by a circular beam of parallel photons yields caustic enhancements at particular deviation angles that correspond to the multiple orders of rainbows that arise according to the number of reflections inside the droplet. Excellent agreement between Opticks and Geant4 were achieved and the deviation angles matched analytic expectations for the rainbow angles.

Using the analytic photomultiplier geometry description described above has enabled excellent agreement to be achieved between Opticks and Geant4 for the test geometry of a single photomultiplier tube within a box of mineral oil illuminated by a circular beam of parallel photons.

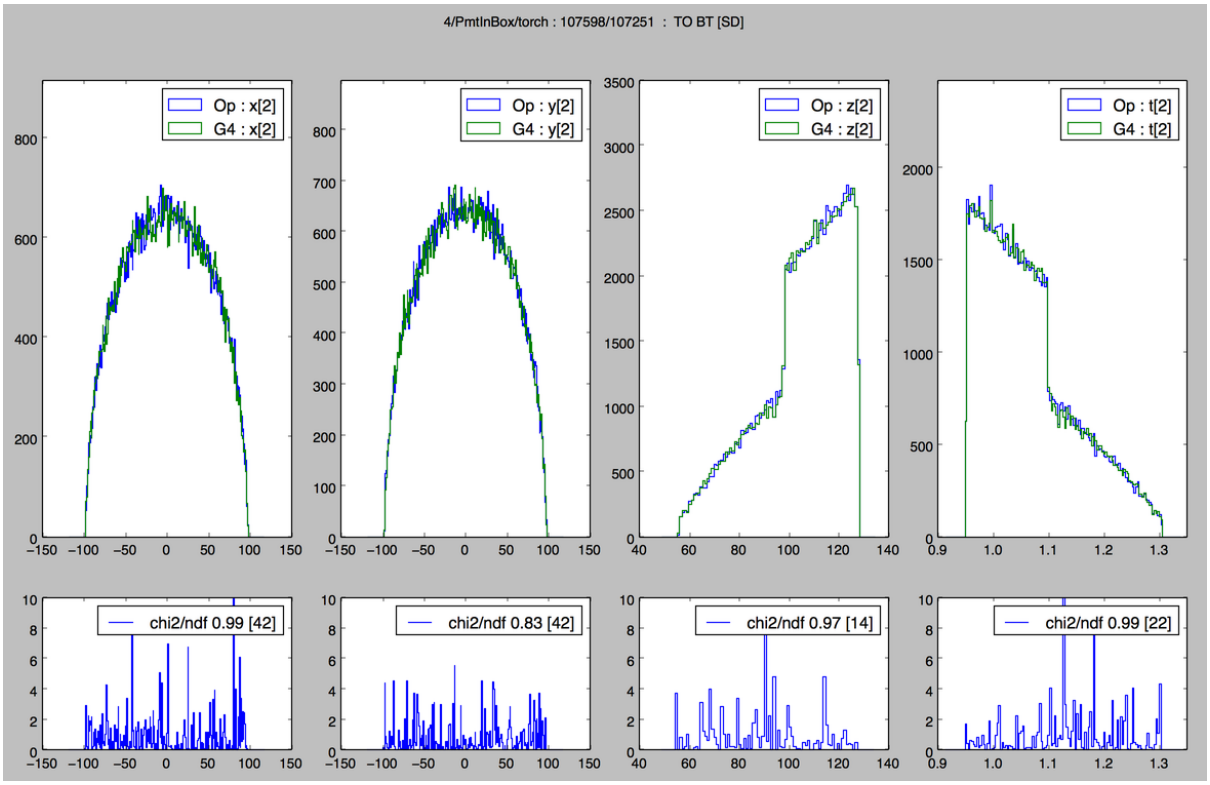

Figure 3: Opticks/Geant4 comparison of photon position and time distributions for single Daya Bay photomultiplier tube within a box of mineral oil geometry illuminated by circular beam of photons. Only the category of an single boundary transmit prior to detection is shown. As indicated by the chi-squared lower plot, good agreement is achieved. Chi-squared comparisons of all photon distributions of the 25 most frequently occuring categories all give good agreement.

4 Opticks Integration with Geant4

4.1 Integration of G4DAE Geometry Exporter with Geant4

At the 19th Geant4 Collaboration Meeting in September 2014, the Geant4 Collaboration accepted my proposal to contribute the G4DAE exporter to the 2015 Geant4 release. However during 2015 my ongoing Opticks validation effort revealed several bugs in the exported Daya Bay geometry, including the cleaved meshes issue described above and some missing optical surfaces that requires further investigation.

In retrospect it is clear that the only way to validate an exported geometry is to be able to validate simulations within that geometry by comparison with originals. Thus I have deferred the integration of the G4DAE geometry exporter with Geant4 until after Opticks simulation validation has reached a mature state.

4.2 Interfacing Opticks with Geant4

Following my presentation on Opticks at NVIDIA's GPU Technology Conference some of the Geant4 developers from the Stanford Linear Accelerator Center (SLAC) expressed an interest in how Opticks might be integrated with Geant4. Discussion of approaches to take are ongoing.

The current G4DAEOpticks interface was developed prior to the development of Opticks and uses a message queue to communicate between a python process and the Geant4 process. Although this approach can still be used with Opticks the C++ implementation of Opticks allows a simpler single process approach to be investigated.

5 Opticks Distribution and Marketing

Opticks provides unprecedented optical photon simulation performance making the time to simulate optical photons effectively zero compared to other processing with the additional benefit of zero CPU memory requirements for the optical photons. Opticks can greatly benefit any physics experiment with a simulation limited by optical photons.

The primary marketing objective is to bring the above message to the simulation groups of the majority of the worlds large photomultipler based experiments in a way that causes them to test Opticks with their geometries. A targetted approach to contact simulation groups beyond Daya Bay and JUNO will be appropriate once validations are mature and the installation and usage path for initial users has been prepared and tested. Physicists associated with several experiments have already expressed an interest in Opticks, namely EXO (Enriched Xenon Observatory), Lake Baikal GVD (Gigaton Volume Detector) and MiniBooNE (Mini Booster Neutrino Experiment).

A secondary objective is to raise awareness of the usefulness of GPU techniques within the field, and to contribute to education and outreach regarding neutrino detectors to a wide audience. Opticks visualizations are especially helpful for young researchers in neutrino experiments. To this end I have dedicated significant time to the preparation of collaboration talks, conference talks and creation of outreach materials such as the videos described above.

In order to prepare the installation and usage path for Opticks users I have developed a standard CMake based cross platform build infrastructure that is intended to make getting, building and installing Opticks straightforward. Also I have started to prepare migration of the Opticks software to a dedicated bitbucket repository website to manage the code, documentation and issue tracking.

Full Opticks simulation functionality requires a machine with a recent NVIDIA GPU due to the dependency on CUDA, Thrust and OptiX. However the OpenGL visualization part of Opticks can be used on a much wider class of machines. To widen the applicability of Opticks a layered dependency refactoring has been done to allow installation of a visualization only variant of Opticks that is able to load photon propagations that have been created and saved on CUDA capable machines. This is particularly relevant within an educational and outreach context where CUDA capable machines are likely to remain rare.

5.1 Marketing Activities

Slides and some videos of the presentations below are accessible from http://simoncblyth.bitbucket.org.

Planned

- 19-21 July 2016, Particle Physics Summer School, Weihai, Organized by Shandong University.
  Opticks : Optical Photon Simulation for Particle Physics with NVIDIA OptiX
  Invited course on Opticks, including 90 minute lecture and two 90 minute tutorial sessions
- 16th May, 2016, Leung Center for Cosmology and Particle Astrophysics (LeCosPA) Seminar, National Taiwan University, Taipei.
  Opticks : Optical Photon Simulation for Particle Physics with NVIDIA OptiX
  Invited seminar.

During the past 12 months

- April 2016, NVIDIA's GTC (GPU Technology Conference), San Jose, California.
  Opticks : Optical Photon Simulation for Particle Physics with NVIDIA OptiX
  Invited conference talk on Opticks to a diverse audience including several SLAC Geant4 developers and medical physics Geant4 developers.
- April 2016, Shared Opticks simulation animation to YouTube and YouKu video sharing sites.
  Opticks GTC 001
  A total of 239 views have been logged over 5 weeks.
- March 2016, Daya Bay Collaboration Meeting, IHEP, Beijing.
  Opticks : GPU Optical Photon Simulation
  Update to Daya Bay Collaboration, including single PMT validation
- Jan 2016, PSROC Meeting, SYSU, Kaoshiung, Taiwan.
  Opticks : GPU Optical Photon Simulation
  Same ground as Xiamen talk, but aiming at more diverse audience
- Jan 2016, JUNO Meeting, Xiamen University.
  Opticks : GPU Optical Photon Simulation
  Progress report on handling JUNO geometry and beginning validations.
- July 2015, JUNO Collaboration Meeting, IHEP, Beijing.
  Optical Photon Simulation with NVIDIA OptiX.
  Introduced JUNO Collaboration to Opticks and NVIDIA OptiX.

Important Earlier Activities

- September 2014, 19th Geant4 Collaboration Meeting, Okinawa.
  G4DAE : Export Geant4 Geometry to COLLADA/DAE XML files
  Invited guest talk to the Geant4 Collaboration introducing geometry exporter.

6 Future Plans

6.1 Validation of Full Geometries

As the simulation is expanded to include more of the Daya Bay and JUNO geometries additional details of the Geant4 optical physics simulation and the detector geometry will need to be worked on in order to achieve a match between Opticks and Geant4.

These details include handling dielectric-metal boundaries and non-specular reflection and also dealing with detector customizations to the optical photon simulation such as the Daya Bay handling of it's ESR (Enhanced Specular Reflector) mirrors. Also the currently tesselated Daya Bay Inner Acrylic Vessel (IAV) and Outer Acrylic Vessel (OAV) need to be replaced by an analytic geometry.

6.2 Attracting Users and Developers

Opticks was conceived and has so far been developed almost entirely by myself. Although there have been several expressions of interest to help with Opticks by JUNO colleagues the steep learning requirements have so far prevented any substantial contributions.

In order to ensure the survival of Opticks it is vital to attract users and developers. Beyond the technical aspects of installation scripts and a dedicated Opticks website, documentation and tutorial materials need to be prepared that cover user and developer tasks. For example users need to be able to work with an existing detector setup whereas developers need to be able to adapt Opticks to work with a new geometry.

Preparations for the planned Opticks course at the Weihai Summer school will entail development of user level documentation and tutorials.

6.3 Moving Into Production

The IHEP group plan to setup a cluster of GPU workstations that hopefully can be used to develop and test production running of Opticks + Geant4. Creating large Monte Carlo samples for validation and use by analysis groups will require working closely with the Monte Carlo Production groups of the Daya Bay and JUNO Collaborations.

7 Conclusion

Opticks provides unprecedented optical photon simulation performance making the time to simulate optical photons effectively zero compared to other processing with the additional benefit of zero CPU memory requirements for the optical photons.

The strategy of adopting NVIDIA OptiX acceleration provides the performance thanks to the massive parallelism that OptiX makes accessible. Due to the active support and ongoing improvements to NVIDIA OptiX the performance of Opticks with future GPU architectures looks to be assured. Adopting OptiX however has entailed substantial and ongoing costs of porting the Geant4 optical physics and especially with bringing the geometry to the GPU.

Beyond validation of full geometries the major focus of the year ahead is bringing this work into production usage first within the Daya Bay and JUNO Collaborations and then expanding marketing efforts to encourage adoption of Opticks by other photomultipler based particle physics experiments. Once validations of full geometries have been established preparation of a technical paper describing Opticks would be appropriate.

8 Other Work

8.1 Daya Bay Infrastructure

Most of the software infrastructure systems I have developed or integrated over the past years remain in continuous operation with multiple processes running around the clock at the main Daya Bay institutions, monitoring every database update and every change to the Daya Bay software, performing software builds and tests and notifying experts of failure conditions.

My responsibilities include:

- software infrastructure (Trac and SVN servers)
- offline database (management, operating procedures, software interfaces)
- online slow control database (monitoring and "scraping" from online to offline)

Occasional urgent requests for help with development of custom scripts, or with the usage of tools I developed are the most time consuming aspects of these responsibilities. Unfortunately a HKU colleague that had taken on support responsibilities regarding the Software Infrastructure and testing system has left his position, so some additional support requirements are expected in the year ahead.