Optical Photon Simulation on GPUs

Simon C. Blyth

May 1, 2017

Contents

1 Overview

Over the past year my focus has been to transform Opticks from a pilot project into a package that can be used in production by the JUNO experiment and other projects. Opticks is a GPU optical photon simulation package of unprecedented performance that I developed. The primary advances over the past year have been:

- Implementation of GPU ray tracing of complex constructive solid geometry (CSG) shapes.
- Adoption of implicit function geometry modelling and integration of polygonizations.
- Validation of GPU optical physics by achieving a match with Geant4.

Of these the highlight advance is the development of GPU ray tracing of CSG solids of arbitrary complexity, due to the wide ranging benefits that it enables, including:

- Geometry equivalence within numerical precision between Geant4 and Opticks on the GPU.
- Automated translation of Geant4 detector geometries into a GPU suitable form.
- Drastically simplified Opticks adoption, due to automated geometry translation.
- Allows Opticks to operate from Geant4 standard GDML files rather than tessellated G4DAE.

2 Introduction and Background

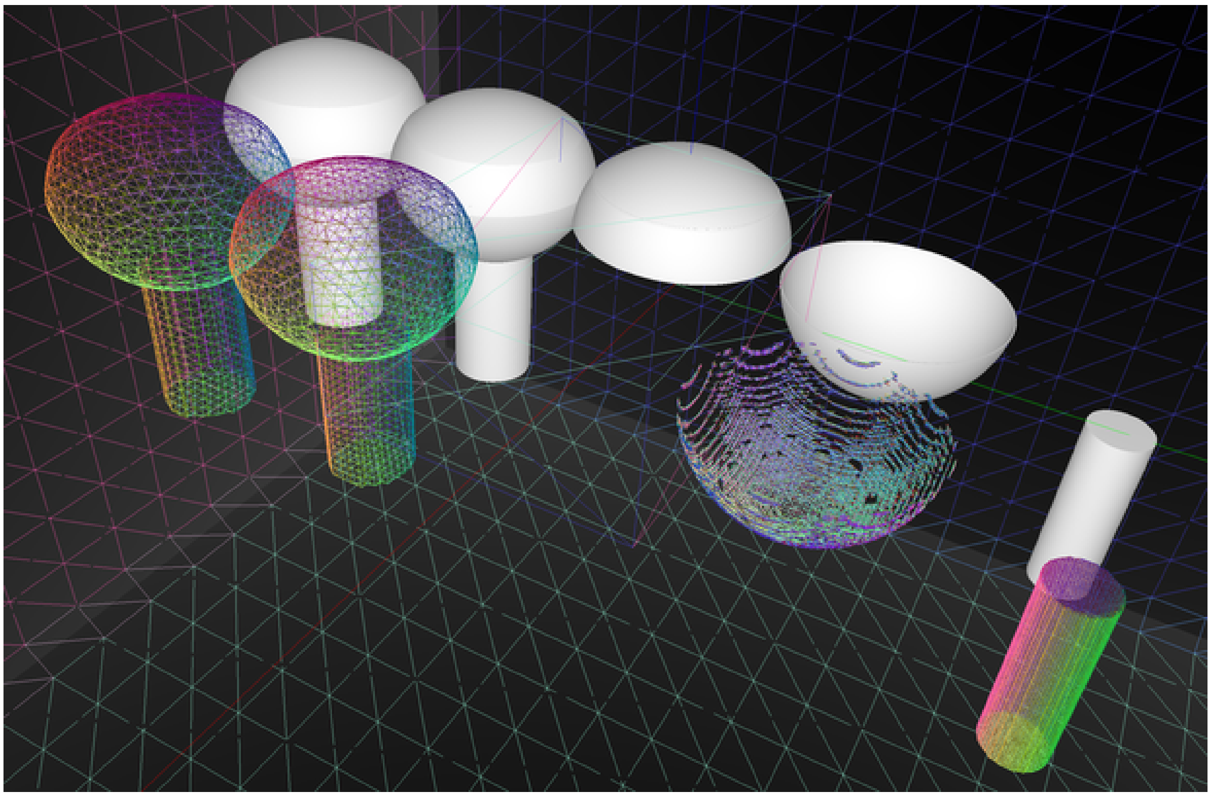

Figure 1: Five component solids of the Daya Bay photomultiplier tube, separately modelled as constructive solid geometry (CSG) node trees. Geant4 original geometry was auto-serialized and copied to the GPU where it was directly OptiX ray traced to yield the Lambertian shaded representations. Polygonizations of the node tree were performed on the CPU using implicit function modelling producing meshes which were uploaded to the GPU and rasterized with OpenGL. The rasterized and ray traced images are composited together to produce the screenshot. Polygonizations of the very thin cathode and base solids require further development.

2.1 Optical Photon Simulation Problem

A detailed understanding of the generation and propagation of optical photons is vitally important to the design, operation and analysis of photomultiplier based neutrino detectors such as Daya Bay, JUNO, IceCube, Baikal GVD and MiniBooNE. Fast optical photon simulation is also crucial for some dark matter and rare decay search experiments such as LZ and EXO and also has industrial uses in the design of new medical imaging devices and analysis of existing ones. Physicists associated with all the above mentioned experiments and one working for a medical imaging device company have expressed an interest in using Opticks, and some have taken the first steps in evaluating it. Interest in Opticks has mostly been generated by the high ranking of `http://simoncblyth.bitbucket.io` in web searches for related terms.

Detailed detector simulations are an essential tool to develop an understanding of complex devices such as neutrino detectors or medical imaging scanners and provides crucial input to the analysis of data from these devices. However, Geant4 based JUNO simulations of crucial cosmic muon background events are found to expend more than 99% of CPU time propagating optical photons.

GATE, the Geant4 Application for Tomographic Emission, is used to simulate medical imaging devices and assist in the design of new devices. The below quote from the GATE documention illustrates the optical photon simulation problem within medical physics:

Most scintillators used in PET generate in the order of 10,000 optical photons at 511 keV, which means that approximately 10,000 more particles have to be tracked for each annihilation photon that is detected. Although the tracking of optical photons is relatively fast, a simulation with optical photon tracking can easily be a factor thousand slower than one without.

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.

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 not only migrate the optical physics simulation to the GPU but also to duplicate the Geant4 context to the GPU, including all geometry, material and surface properties.

Opticks replaces the Geant4 optical photon simulation with an equivalent optical simulation implemented on the GPU and accelerated by the NVIDIA OptiX ray tracing engine. All the optical physics processes implemented have been validated using simple purely analytic geometries. Performance speedup factors of 200x relative to Geant4 with a mobile GPU with only 348 cores have been achieved. 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.

2.3 Constructive Solid Geometry (CSG)

CSG is an intuitive geometry modelling technique whereby complex solid shapes are composed hierarchically from combinations of simpler solids using the boolean set operations of union, intersection and difference. CSG expressions for a geometry can be stored in binary trees with primitives such as spheres or cylinders represented in the leaf nodes and boolean operations represented in the internal nodes of the tree.

2.4 GDML and G4DAE

Geometry Description Markup Language (GDML) is the Geant4 native XML based file format that allows the in memory Geant4 geometry model to be written to file and read back from file.

Geant4 Digital Asset Exchange (G4DAE) is a file format containing triangulated Geant4 geometries written by the G4DAE geometry exporter that I developed. G4DAE is based upon the standard COLLADA/DAE 3D file format with extensions to include Geant4 material and optical surface properties.

2.5 NVIDIA OptiX GPU ray tracing engine

Opticks is based upon the NVIDIA OptiX GPU ray tracing engine. OptiX ray tracing pipelines are constructed from a small set of user provided CUDA programs analogously to how OpenGL rasterization pipelines are constructed from GLSL shaders.

OptiX provides only the acceleration of geometrical intersection, not the intersection itself. Geometry information is provided to OptiX in the form of CUDA programs that return bounding boxes and ray primitive intersection positions and surface normals.

3 GPU ray tracing of CSG solids

3.1 Development of CSG ray tracing algorithm

Most algorithms for calculating intersections of rays with constructive solid geometry require finding and storing all intersections of the ray with all primitives and then computing intersections by combination of the intervals appropriate to the boolean operations. This approach is poorly suited to the GPU environment where low memory usage for each thread is crucial to allow many thousands of threads to effectively run in parallel. In addition this approach would need to be supported within the intersect implementations of every primitive.

A quite different approach is described in a note “Ray Tracing CSG Objects Using Single Hit Intersections” by Andrew Kensler, with corrections from the XRT renderer author http://xrt.wikidot.com/doc:csg. Ray intersections with each sub-object are classified as miss, enter or exit based on the angle of the ray to the surface normal direction. Combinations of the classifications yield an action such as returning a hit or a miss or advancing the ray and intersecting again. These actions are encoded into state tables for each of the boolean operations.

The note provided pseudocode for a recursive algorithm for the combination of two sub-objects. I extended the algorithm to handle general CSG node trees of any depth within a recursive python prototype.

3.2 Implementation of CSG ray tracing within Opticks

Although NVIDIA OptiX supports recursive ray tracing, it does not support recursion within geometry intersection. Recursion is typically the simplest way to handle tree data structures such as CSG node trees but it is inherently stack memory intensive making it inappropriate for use in the resource constrained GPU environment.

My CSG implementation was inspired by the realization that CSG node tree intersection directly parallels binary expression tree evaluation and that techniques to simplify expression tree evaluation such as using postorder traversals could be applied. Binary expression trees are used to represent and evaluate mathematical expressions. A postorder traversal of a node tree visits every node in sequence such that child nodes are visited before their parents. Factoring out the postorder sequence allowed an iterative solution to be developed for a recursive problem.

The CSG node tree is constructed on the host using either a simple python language description or directly in C++. The node tree is then serialized as a complete binary tree into a buffer and uploaded to the GPU. Use of the complete binary tree serialization greatly simplifies handling on the GPU as the regularity of the tree structure allows child and parent indices to be computed rather than stored and even allows the postorder sequence to be derived by simple bit manipulations alone. This means that the buffer containing the serialized tree can be directly ray traced without deserialization. The implementation supports concatenation of multiple serialized trees within a single buffer.

4 Implicit Geometry Modelling

4.1 Opticks CSG Primitives

Opticks currently supports a small number of solid primitives: sphere, box, cylinder and z-sliced sphere. The z-sliced sphere and cylinder have endcap implementations, as CSG constituents are required to be closed solids. Each primitive requires:

- CUDA function providing the bounding box of the solid
- CUDA function providing position and surface normal at ray intersections
- CPU signed distance function

General homogeneous transformation matrices representing translations, rotations or non-uniform scaling can be associated to any node of the CSG tree, providing hierarchical geometry transformations. Support for non-uniform scaling allows ellipsoidal shapes to be represented by scaling of the sphere primitive, avoiding the need to implement an ellipsoid primitive.

4.2 Signed Distance Functions (SDF) and R-function composition

Adoption of signed distance functions for all primitives is a major advance in Opticks geometry modelling. A signed distance function (SDF) provides the distance of any point to the surface of a solid, with negative values conventionally corresponding to positions within the solid, zero values on the surface and positive values outside the solid. The great advantage of using this implicit geometry description compared to explicit parametric descriptions is that CSG combinations of solids can be constructed using Rvachev functions (R-functions), such as the min and max functions. The sign of an R-function is uniquely determined by the signs of its arguments, making R-functions parallel CSG operations, for example in the combination of two solids:

- SDF(Union(A,B)) ~ min(SDF(A),SDB(B))
- SDF(Intersection(A,B)) ~ max(SDF(A),SDF(B))
- SDF(Difference(A,B)) ~ max(SDF(A),-SDF(B))

Recursive application of these combinations within CSG node trees yield implicit function representations. Descartes 1637 treatise "Geometry" established the analytic representation of geometry. Subsequently analytic forms for a few simple shapes were discovered but the general inverse problem of finding an analytic representation of an arbitrary shape, was only solved in the 1960s by Rvachev.

4.3 Polygonization of CSG node trees

Conversion of the implicit CSG geometry description into an explicit surface description in the form of triangle meshes using polygonization is not strictly required within Opticks, as the package uses ray tracing to both visualize CSG geometries and to provide intersection positions for the optical photon propagation simulation.

However OpenGL rasterized visualizations, which require meshes, are subtantially faster than ray traced visualizations making the generation of meshes an important cross check and debugging tool due to the fast visualizations that it enables. This geometry cross check is made particularly useful within Opticks due to the use of compositing techniques that allow the OptiX ray trace and the OpenGL rasterization to be simultaneously visualized in an interactive 3D space.

Isosurface extraction techniques are used to construct polygonizations of the implicitly defined CSG solids using only the R-function recursively combined signed distance functions. Several open source polygonization implementations have been integrated with Opticks either as optional externals or internally.

- PyMCubes : fixed resolution marching cubes implementation
- Dual Contouring : adaptive multi-resolution Octree approach
- Implicit Mesher : fast surface following approach

Unfortunately all these polygonization techniques are found to perform badly with very thin solids such as cathode of the Daya Bay photomultiplier tube, which is modelled using the CSG difference of two z-sliced sphere primitives of almost equal radius. It is expected that a hybrid parametric and implicit polygonization approach can be devised to solve the thin solid problem.

5 Opticks Validation

5.1 Optical Physics Processes ported to the GPU

All of the Geant4 optical photon propagation processes relevant to Daya Bay and JUNO are implemented on the GPU within OptiX CUDA programs. The processes include absorption, Rayleigh scattering, Fresnel reflection and refraction, diffuse reflection and scintillator reemission. Similarly optical photon generation from scintillation and Cerenkov processes using buffers of generation step parameters collected from Geant4 are implemented on the GPU within OptiX CUDA programs.

5.2 Optical Physics : Opticks/Geant4 chi-squared minimization

Validation comparisons use a single executable that performs both the Geant4 and Opticks simulations and writes two events using an Opticks event format that includes highly compressed information for the first 16 photon propagation points. These events are compared by forming chi-squared distances for:

- photon history counts : within the 100 most frequent categories
- photon step-by-step distributions : 8 quantities, position, time, polarization and wavelength

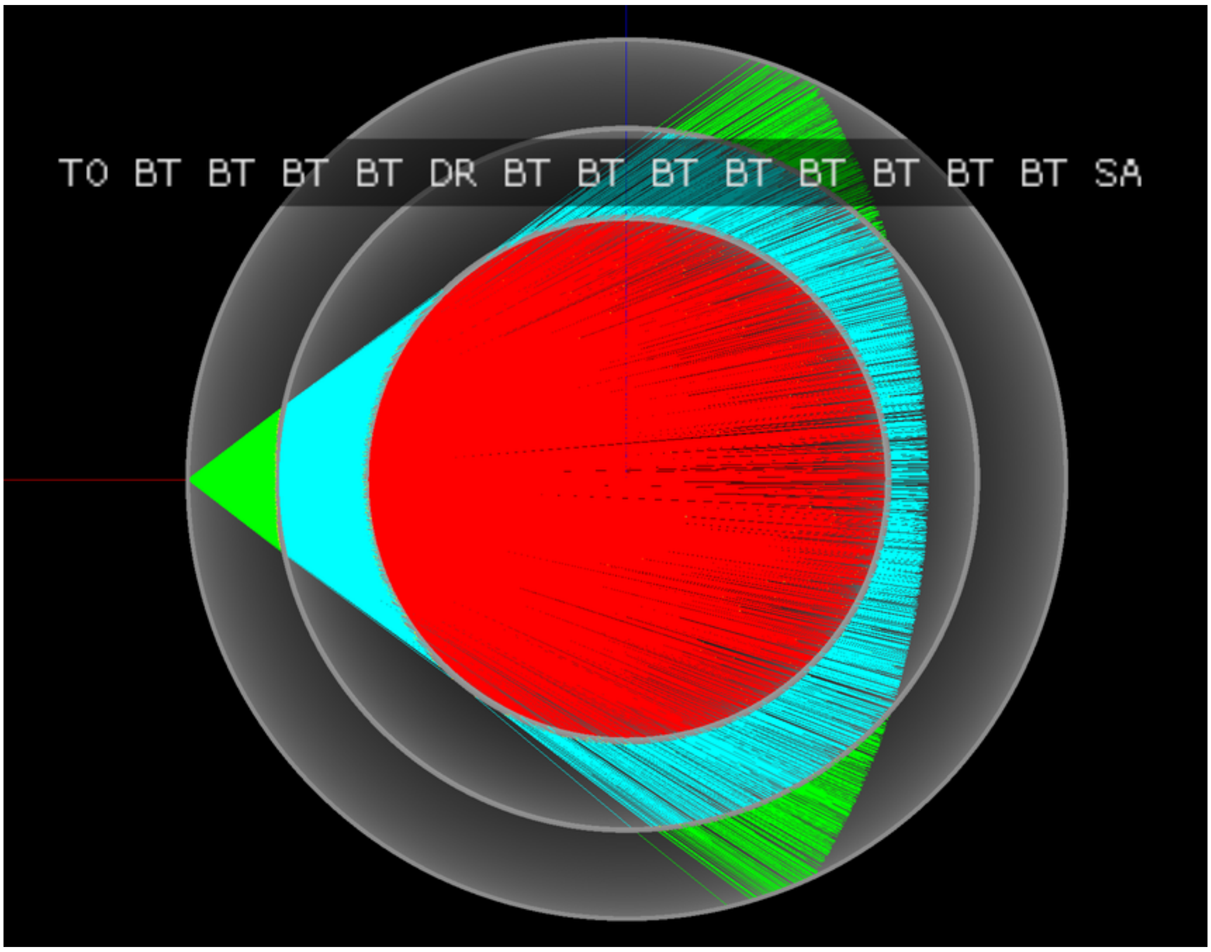

Figure 2: Simplified three liquid detector geometry arranged in concentric spheres separated by acrylic. Photons from history category “TO BT BT BT BT DR BT BT BT BT BT BT BT BT SA” are shown, where the abbreviations are, TO:Torch, BT:Boundary Transmit, DR:Diffuse Reflect, SA:Surface Absorb and the initial photons all travel along the X axis indicated by the red line. The line colors represent the material the photon is traversing, red:Gadolinium doped liquid scintillator, cyan: liquid scintillator and green: mineral oil. The simplicity of this test geometry was adopted in order to debug an issue of Geant4 using the group velocity from the wrong material after refraction.

```
.       1000000   1000000       373.13/356 =  1.05  (pval 0.256 prob 0.744)
        Opticks    Geant4
0000     669843    670001          0.02    TO BT BT BT BT SA
0001      83950     84149          0.24    TO AB
0002      45490     44770          5.74    TO SC BT BT BT BT SA
0003      28955     28718          0.97    TO BT BT BT BT AB
0004      23187     23170          0.01    TO BT BT AB
0005      20238     20140          0.24    TO RE BT BT BT BT SA
0006      10214     10357          0.99    TO BT BT SC BT BT SA
0007      10176     10318          0.98    TO BT BT BT BT SC SA
0008       7540      7710          1.90    TO BT BT BT BT DR SA
0009       5976      5934          0.15    TO RE RE BT BT BT BT SA
0010       5779      5766          0.01    TO RE AB
0011       5339      5269          0.46    TO BT BT BT BT DR BT BT BT BT BT BT BT BT SA
0012       5111      4940          2.91    TO BT BT RE BT BT SA
0013       4797      4886          0.82    TO SC AB
0014       4494      4469          0.07    TO BT BT BT BT DR BT BT BT BT SA
0015       3317      3302          0.03    TO BT BT SC BT BT BT BT BT BT SA
0016       2670      2675          0.00    TO SC SC BT BT BT BT SA
0017       2432      2383          0.50    TO BT BT BT BT DR AB
0018       2043      1991          0.67    TO SC BT BT BT BT AB
0019       1755      1826          1.41    TO SC BT BT AB
```

The above text table shows the photon counts from Opticks and Geant4 for the top 20 photon history categories obtained from a simulation of 1 million photons within the tconcentric test geometry together with the chi-square distance of each category and the overall chi-squared distance. The abbreviations are TO:torch, BT:boundary transmit, SA:surface absorb, AB:bulk absorb, RE:reemission, SC:scatter, DR:diffuse reflect. The overall chi-squared per degree of freedom and a similar one obtained for the distributions at each propagation point is used to check the consistency of the simulations.

The top 100 photon history categories correspond to ~900 photon propagation points with 8 quantities per point this corresponds to 7200 histogram pairs that are compared. After numerous bug fixes directed by the next largest chi-square contributor statistically consistent GPU and CPU simulations for the photon counts and distributions have been achieved.

6 Opticks Marketing Activities

The highlight marketing activity from the past year has been my presentation of Opticks at the 22nd International Conference on Computing in High Energy and Nuclear Physics (CHEP) to an audience which included many of the core Geant4 developers. My talk and introductory video were well received, prompting much interest and many questions. Subsequently, I have heard that my work has inspired some Geant4 developers to initiate investigations of GPU ray tracing. My conference proceedings paper has recently been accepted for publication in the IOP Conference Series.

Slides and videos of my presentations are accessible from `http://simoncblyth.bitbucket.io`

During the past 12 months

- April 2017, CHEP Proceedings accepted for publication in IOP Conference Series.
  Opticks : GPU Optical Photon Simulation for Particle Physics using NVIDIA OptiX

- December 2016, JUNO Workshop, LLR, Ecole Polytechnique, Paris.
  Opticks : Optical Photon Simulation for Particle Physics with NVIDIA OptiX
  Invited workshop talk.

- October 2016, 22nd International Conference on Computing in High Energy and Nuclear Physics (CHEP). Hosted by SLAC and LBNL, San Francisco.
  Opticks : Optical Photon Simulation for Particle Physics with NVIDIA OptiX
  Contributed conference talk.

- 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
- May 2016, LeCosPA Seminar, National Taiwan University, Taipei.
  Opticks : Optical Photon Simulation for Particle Physics with NVIDIA OptiX
  Invited seminar.

Important Earlier Activities

- 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,
  `http://on-demand.gputechconf.com/gtc/2016/video/s6320-simon-blyth-opticks-nvidia-opt`
- 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.

7 Future Plans

7.1 Exact Full Detector Geometry Translation

General CSG ray tracing on the GPU now makes it possible to develop a fully automated conversion of Geant4 detector geometries into an exact equivalent form that is appropriate for massively parallel usage on the GPU.

Currently the translation is operational with small geometry subtrees such as the handful of solids that describe a single photomultiplier tube. In order to still benefit from OptiX acceleration structures and OptiX and OpenGL instancing the translation is intended to be done solid by solid. Some experimentation is required to determine the granularity at which to split up the full geometry trees into repeated subtree instances.

Full translation of Daya Bay geometry requires implementations of a few more primitives: trapezoid, cone and compound cone with varying radius at multiple z planes.

7.2 Full Detector Validation

The exact geometry equivalence between Geant4 on the CPU and Opticks on the GPU is expected to allow simple step-by-step chi-squared distance comparisons of photon parameters such as wavelength, polarization and position.

7.3 Opticks Integration with JUNO Offline framework

I plan to implement as much as possible of the integration beneath the level of the experiment, at the Geant4 level, allowing this work to be carried over to other experiments straightforwardly. To what extent this is possible will depend on how tightly JUNO Offline couples with Geant4. Fortunately JUNO Offline is in active development so it is possible that changes can be made to ease the integration.

Opticks requires generation steps to be collected from Geant4 scintillation and Cerenkov processes and photon detector hits are subsequently returned en masse to the Geant4 hit collections allowing subsequent electronics simulation to proceed unmodified.

Cosmic muon events with maximum path length through the JUNO central detector sphere of 35m diameter of liquid scintillator are expected to yield of order 60 millions of optical photons. Automated GPU launch splitting depending on available GPU memory will probably be required to generate and propagate such numbers of photons. Such splitting can be straightforwardly implemented as every scintillation or Cerenkov generation step that is transferred to the GPU carries the number of photons to be generated for the step.

8 Conclusion

Opticks provides unprecedented optical photon simulation performance thanks to the massive GPU parallelism that the NVIDIA OptiX ray tracing engine makes accessible. Simulation time for optical photons becomes effectively zero compared to other processing and CPU memory allocation is only required for photons that hit detectors.

The very recent implementation of general CSG ray tracing on the GPU within Opticks promises to finally enable full detector validations of the Opticks optical physics implementations due to the exact geometry equivalence between Geant4 on the CPU and Opticks on the GPU. In addition the fully automated conversion of Geant4 geometries into a form suitable for GPU accelerated ray tracing drastically simplifies adoption of Opticks by new projects.

Beyond full Opticks validation the major focus of the year ahead is bringing Opticks into production usage first within the JUNO Collaboration and then working with early adopters from other projects to simplify Opticks usage within their frameworks.