Integration of Opticks¹ and Geant4 (an advanced example: CaTS)

Hans Wenzel
Soon Yung Jun
Krzysztof Genser
Alexei Strelchenko

¹developed by Simon Blyth
(Institute of High Energy Physics, Chinese Academy of Sciences)
Outline

- Opticks/G4Opticks/CaTS overview.
  - Opticks resources.
- CaTS:
  - Latest developments.
  - CaTS resources.
- Status of re-implementing Opticks for Optix7.
- Plans.
Opticks is an open-source project that accelerates optical photon simulation by integrating NVIDIA GPU ray tracing, accessed via NVIDIA OptiX.

Developed by Simon Blyth:
https://bitbucket.org/simoncblyth/opticks/

CaTS: interfaces Geant4 user code with Opticks using the G4Opticks interface provided by Opticks. It defines a hybrid workflow where generation and tracing of optical photons is offloaded to Opticks (GPU), while Geant4 (CPU) handles all other particles. CaTS was included in Geant4 11.0 as an advanced example:
https://geant4.kek.jp/lxr/source/examples/advanced/CaTS/
Opticks resources: https://simoncblyth.bitbucket.io/env/presentation/opticks_may2020_hsf.html

EPJ Web of Conferences 214, 02027 (2019), https://doi.org/10.1051/epjconf/20192140202
Simon Blyth: "Opticks : GPU Optical Photon Simulation for Particle Physics using NVIDIA® OptiX™ “

Detector geometry in Opticks: https://indico.cern.ch/event/975008/
Documentation: https://simoncblyth.bitbucket.io/opticks/index.html

Code repositories: https://bitbucket.org/simoncblyth/opticks/: main development repository
The most recent tag is https://github.com/simoncblyth/opticks/releases/tag/v0.1.7
→ Starting point from ‘our’ github fork.
CaTS: advanced Geant4 example

- Uses Geant4 to collect Scintillation and Cerenkov Gensteps. A Genstep collects all the data necessary to generate Cerenkov/Scintillation photons on the GPU. The harvesting is done in Sensitive Detectors (SD) (RadiatorSD/IArTPCSD). The number of photons to be generated is calculated by Geant4 and constrained to be identical whether one uses the Geant4 optical physics or G4Opticks.
- Use of G4Opticks is both a build and run-time option.
- The PhotonHits collected by the PhotonSD sensitive detector have the same content whether Geant4 or G4Opticks is used.
- Uses GDML with extensions for flexible Detector construction and to provide optical properties at runtime. The gdml extensions include:
  - Assigning Sensitive Detectors to logical Volumes.
  - **RadiatorSD**, IArTPCSD, PhotonSD.
  - TrackerSD, CalorimeterSD, DRCalorimeterSD, ...
  - Assigning step-limits to logical Volumes.
  - Assigning production Cuts by regions.
  - Assigning visualization attributes.
  - Note there are Opticks specific keywords!
- Uses G4PhysListFactoryAlt to define and configure physics.
- Uses Root IO to provide persistency for Hits.

Achieved speed up in the order of a few times $10^2$, depends strongly on detector geometry, hardware and settings.
Latest developments:

- For the Geant4 advanced example:
  - Lots of code and cmake cleanup, make use of C++17 features, follow Geant4 code conventions added visual code configuration.
  - More examples: (e.g. scintillation crystals, WLS examples, ...).
  - Help users getting started with CaTS.
  - Ensure that gdml examples are compatible with Geant4 10/11 and Opticks.
- Various optimizations.
- Changes to RootIO.
- Allow for Geant4 event-level multi-threading for offloading Opticks–autolock for G4Opticks call (generate and propagate Photons). But poor scaling due to mutex (as expected as kernels calls are sequential).
Hardware: Nvidia RTX3090, Core i9-10900k@ 3.7Ghz
10 CPU cores, results are very preliminary.

<table>
<thead>
<tr>
<th>Number of threads</th>
<th>Geant4 [sec/evt]</th>
<th>Opticks [sec/evt]</th>
<th>Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>330</td>
<td>1.8</td>
<td>189</td>
</tr>
<tr>
<td>2</td>
<td>175</td>
<td>1.8</td>
<td>101</td>
</tr>
<tr>
<td>4</td>
<td>105</td>
<td>1.8</td>
<td>59</td>
</tr>
<tr>
<td>8</td>
<td>45</td>
<td>1.9</td>
<td>24</td>
</tr>
<tr>
<td>16</td>
<td>36.5</td>
<td>2.0</td>
<td>18</td>
</tr>
</tbody>
</table>
CaTS/Opticks resources

- **https://github.com/hanswenzel/opticks**, our fork to work on Opticks with Optix 6.5.

  Changes include:
  - Changes to make it compatible with the Geant4 11 API.
  - Bulk reemission process during photon propagation disabled. If reemission is necessary, it can be expressed as WLS process where Scintillation and WLS share the same PDF.
  - Extract properties relevant to WLS.

- Main git repository: **https://github.com/hanswenzel/CaTS/**: used for development.

- Instructions how to build and run CaTS:  
  [https://github.com/hanswenzel/CaTS/blob/master/README.md](https://github.com/hanswenzel/CaTS/blob/master/README.md)

- Instructions how to build Opticks and how to install all necessary software:  
  [https://github.com/hanswenzel/CaTS/blob/master/Instructions.md](https://github.com/hanswenzel/CaTS/blob/master/Instructions.md)

- Instruction how to run various CaTS examples (examples consist of: gdml file, Geant4 macro and an application that makes histograms from the hits collections):  
  [https://github.com/hanswenzel/CaTS/blob/master/Examples.md](https://github.com/hanswenzel/CaTS/blob/master/Examples.md)

- [https://gitlab.cern.ch/geant4/geant4](https://gitlab.cern.ch/geant4/geant4): will be used for snapshots and tagged releases
Status of re-implementing Opticks for Optix 7\textsuperscript{1}.

Huge change unavoidable from new OptiX API $\rightarrow$ So profit from rethink of simulation code $\rightarrow$ 2nd implementation advantage. Goals of re-implementation: flexible, modular GPU simulation, easily testable, less code:

- COMPLETED: Full Simulation re-implementation for OptiX 7 API, but new workflow not ready for testing yet.
- Many packages were removed or are planned to be removed.
- Move code that doesn’t require Optix or Cuda out of GPU context (SYSRap, not QUDARap).
- Rather monolithic .cu was replaced by many small GPU+CPU headers.

\textsuperscript{1} Extracted from status report by Simon Blyth for more details see: https://simoncblyth.bitbucket.io/env/presentation/opticks_20220718_towards_production_use_juno_collab_meeting.html
Plans

G4Opticks/Opticks:

- Add opticks to the framework used by the liquid Argon TPC community. That requires Cuda, Optix, Opticks … being packaged in the way required by these frameworks (ups/upd, spack).
- Try out the new Opticks as soon as Simon gives the go-ahead.
- Use the same implementation of the scintillation process on CPU and GPU, use the same optical properties/keywords.
- Implement Wavelength shifting process (WLS).
- Have a look at simplifying cmake, e.g., get rid of obsolete modules.

CaTS:

- Achieve true concurrency by using G4Tasking. Allow to fully utilize GPU resources like vram, cpu cores, multiple GPU’s.
- Change to use Root TBufferMerger for RootIO instead of using separates file and merging them when in multithreaded mode.
- Provide benchmarking and physics validation results with realistic (including WLS) liquid Argon TPC geometry.

Geant4:

- Move the harvesting of Gensteps to the Geant4 optical producer processes (G4Cerenkov, G4Scintillation)→ general interface to external ray tracing programs like Opticks. Trigger the G4Opticks from UserSteppingAction whenever sufficient photons/gensteps are collected for efficient processing on GPU.