CaTS: Integration of Geant4 and Opticks

Hans Wenzel

Soon Yung Jun
Krzysztof Genser

Outline

- Motivation:
  - The computational challenge for TPCs based on liquid Argon (LArTPCs).
  - Simulation of optical photons: an ideal application to be ported to GPU’s.
- Opticks.
- CaTS is an advanced example Geant4 application.
  - CaTS workflow.
  - Performance.
- Plans.

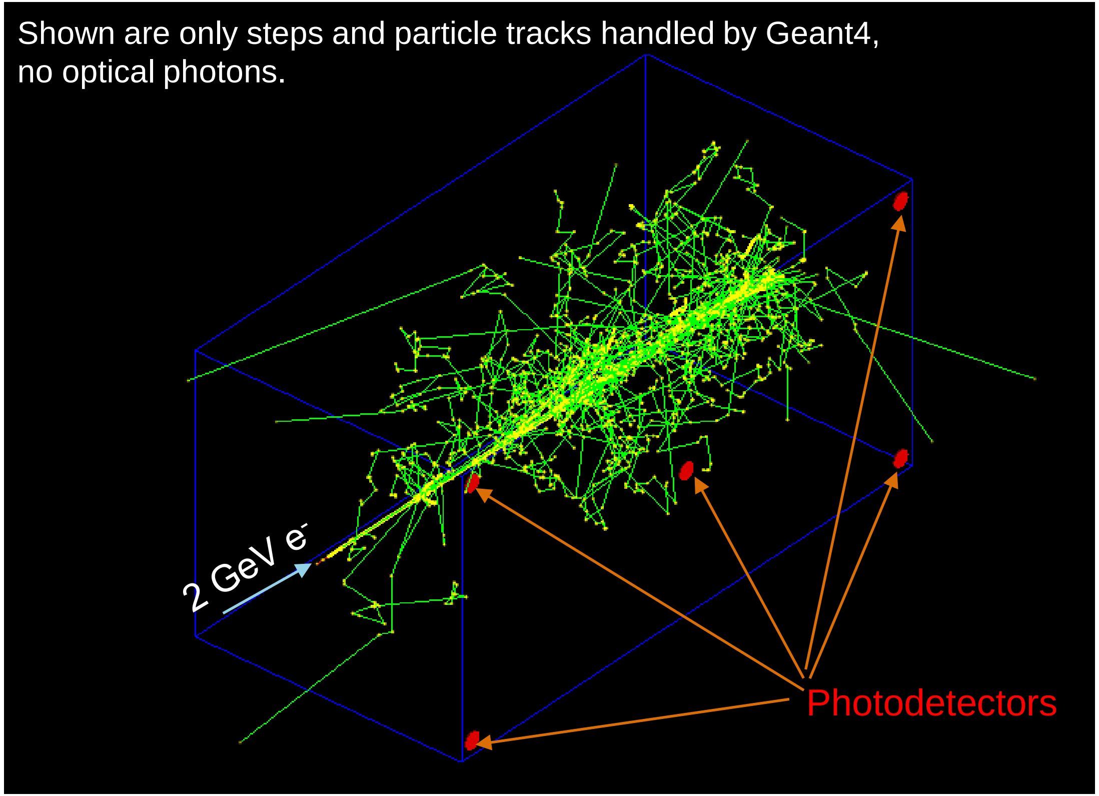
The computational challenge for TPCs based on liquid Argon (LArTPCs):

Test Detector Geometry:
Liquid Argon: x y z: 1 x 1 x 2 m (blue)
5 photo detectors (red)
photon yield (no E-field): 50000 γ/MeV
single 2 GeV electron (shower not fully contained)
(low Z=18, low $\rho = 1.78$ g/cm^3).

- **~ 7x10^7** VUV scintillation photons are produced/event.
- Using Geant4 (11.1.p01) to simulate photon generation and propagation o using a single core on an Intel® Core i9-10900k@ 3.7Ghz takes :

~ minutes/event

(Compared to **0.034 seconds/event** without optical photon simulation) → LArTPC-Experiments use look up tables and parameterizations instead of full simulation for photon response.



Shown are only steps and particle tracks handled by Geant4, no optical photons.

Simulation of optical photons: an ideal application to be ported to GPU's.

- Only one particle type is involved (optical photon), but many of them (~10^7/event)→ allows for massive parallelism (low latency, no big fluctuations in computing time).
- No new particles types besides optical photons are produced.
- Only a few physics processes need to be implemented on the GPU. The processes are:
  - G4Cerenkov (generate photons),
  - G4Scintillation (Reemission) (generate photons),
  - G4OpAbsorption,
  - G4OpRayleigh,
  - G4OpBoundaryProcess,
  - G4OpWLS (not yet implemented, need it for LArTPCs).
- These processes don't need a lot of input data (collected in so called GenSteps for the Cerenkov and Scintillation processes)→ little data transfer from host to device.
- Only a small fraction of photons reach the Photodetectors and produce a PhotonHit → so very little data to transfer from device to host.
- Optical ray tracing is a well-established field → benefit from available efficient algorithms (OptiX®).
- Use NVIDIA® hardware and software (NVIDIA® CUDA, NVIDIA® OptiX ®).

Opticks

Opticks is an open-source project developed by Simon Blyth: https://bitbucket.org/simoncblyth/Opticks/ See also talk at CHEP2023. There are 2 major versions: legacy Opticks based on OptiX ® 6 using the G4Opticks API and reengineered Opticks based on OptiX ® 7 using the G4CXOpticks API.

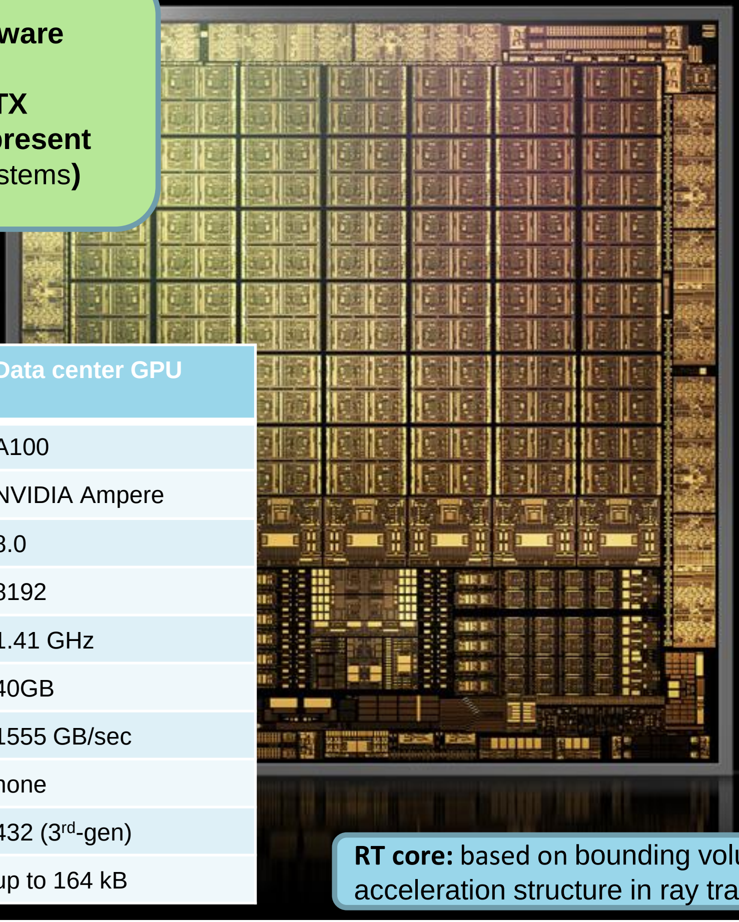
Opticks accelerates optical photon simulation by:

- Translating the Geant4 geometry to OptiX ® without approximation (limited number od shapes).
- Implementing the Geant4 optical processes on the GPU.
- Integrating NVIDIA GPU ray tracing (accessed via NVIDIA OptiX ®).

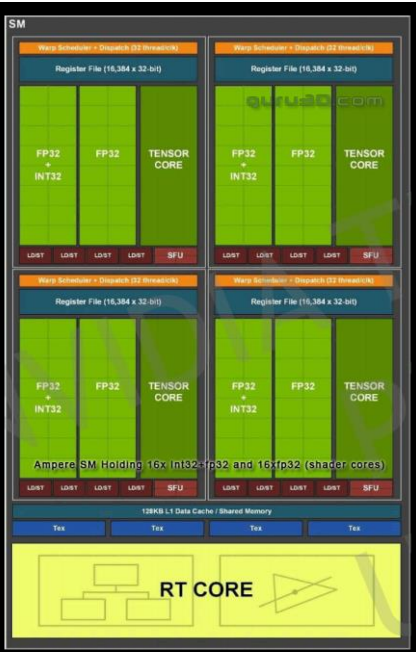
G4(CX)Opticks provides an API to interface Geant4 and Opticks. The Geant4 advanced example CaTS (**C**alorimetry **a**nd **T**racking **S**imulation) uses this API to implement a hybrid workflow:

- Geant4 on the CPU/host handles all particle types but the optical photons.
- The Geant4 Cerenkov and Scintillation processes are still used to calculate the number of optical photons to be generated at a given step and to provide all necessary quantities to generate the photons on the GPU.
- The information collected is the so called GenStep which are different for Cerenkov (needs e.g β^{-1}) and Scintillation (needs e.g. edep, scintillation time constants).
- Copying GenSteps to the GPU→ more efficient than e. g. copying optical photons.
- Generation and tracing of optical photons is offloaded to Opticks (GPU/device) at stepping level whenever a certain number of photons is reached.

Opticks will only run on:
NVIDIA® hardware and NVIDIA® software
Software: NVIDIA® CUDA, OptiX
OptiX 6: allows to select/deselect RTX
OptiX 7: RTX cores are used when present
(RTX is not usually available on HPC systems**)**



	Graphics card	Data center GPU
	GeForce RTX 3090	A100
architecture	NVIDIA Ampere	NVIDIA Ampere
Compute capability	8.6	8.0
CUDA cores	10,496	8192
Boost Clock	1,7 GHz	1.41 GHz
Memory	24 GB	40GB
Memory bandwidth	936 GB/sec	1555 GB/sec
RT cores	82 (2nd-gen)	none
Tensor cores	382 (3rd-gen)	432 (3rd-gen)
Shared Memory size	64kB	up to 164 kB


SM
Warp Scheduler + Dispatch (32 thread/clk)
Register File (16,384 x 32-bit)
FP32 + INT32
FP32
TENSOR CORE
LD/ST
SFU
Ampere SM Holding 16x Int32+fp32 and 16xfp32 (shader cores)
128KB L1 Data Cache / Shared Memory
Tex
RT CORE
guru3D.com

RT core: based on bounding volume hierarchy (BVH), a commonly used acceleration structure in ray tracing, ray-triangle intersection.

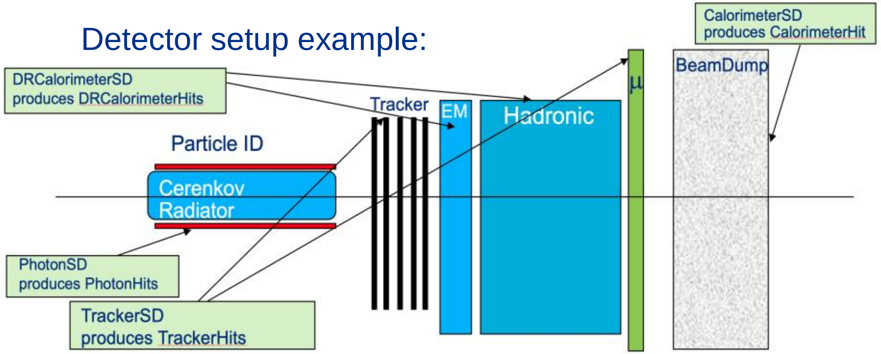
CaTS: Calorimeter and Tracker Simulation

- Advanced Geant4 example application (introduced with Geant4 11.0).
- No changes to Geant4 required to integrate Opticks! Only make use of provided interfaces: UserActions, Sensitive Detectors...
- Modular and extendible, allows to build detector setups from predefined components.
- Use GDML with extensions for flexible Detector construction. GDML extensions are used to:
  - Assign sensitive detectors to logical Volumes. A library of various sensitive detectors and associated Hit-classes is provided.
  - Assign step-limits and energy cuts to logical Volumes.
  - Assign visualization attributes.
- Creation of Hit collections and ROOT IO based IO thereof is automated.
- Currently supports legacy/new Opticks interface.
- Uses G4PhysListFactoryAlt to define and configure physics at runtime via command line option

```
../CaTS -g simpleLArTPC.gdml -pl 'FTFP_BERT+OPTICAL+STEPLI
```

- G4(CX)Opticks/Geant4 is a runtime/build time option.
- Collection of Scintillation and Cerenkov Gensteps by Geant4.

Detector setup example:



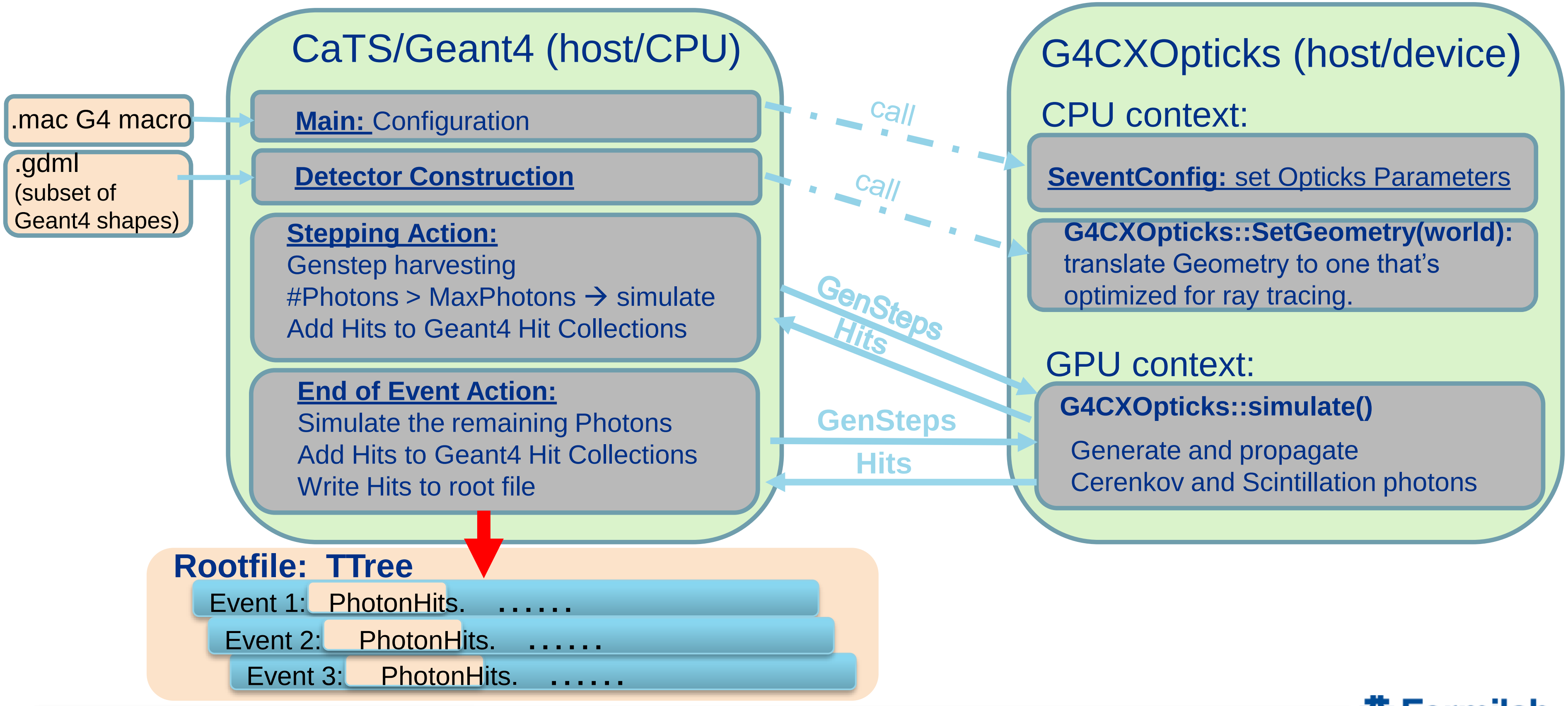
https://geant4.kek.jp/lxr/source/examples/advanced/CaTS/,
https://github.com/hanswenzel/CaTS (development)

Recent developments

Re-implementing Opticks for OptiX ® 7 required huge changes due to the new and very different OptiX ® 7 API → This gave an opportunity to redesign the Opticks code. Goals of re-implementation: flexible, modular GPU simulation, easily testable, less code. For details see e.g Opticks presentation at CHEP.

- CaTS has been modified to use the new Opticks API. The CaTS workflow has been adjusted accordingly. User Actions were utilized → no changes to Geant4 itself required.
- **But:** Opticks API just recently changed ☹. Need to adjust all the Opticks call outs to the new API → in progress! Demonstrates the need for Opticks releases.
- CaTS/Opticks were modified to work with Geant4 API changes introduced in 11.1.
- With legacy versions of Opticks (based on Optix 6) we observed speed ups in the order of $2\text{x}10^2$. Evaluation, profiling and optimizing the performance with recent updates is in progress.

CaTS workflow using the new version of Opticks based on OptiX®7:



.mac G4 macro

.gdml (subset of Geant4 shapes)

CaTS/Geant4 (host/CPU)

Main: Configuration

Detector Construction

Stepping Action:
Genstep harvesting
#Photons > MaxPhotons → simulate
Add Hits to Geant4 Hit Collections

End of Event Action:
Simulate the remaining Photons
Add Hits to Geant4 Hit Collections
Write Hits to root file

G4CXOpticks (host/device)

CPU context:

SeventConfig: set Opticks Parameters

G4CXOpticks::SetGeometry(world):
translate Geometry to one that's optimized for ray tracing.

GPU context:

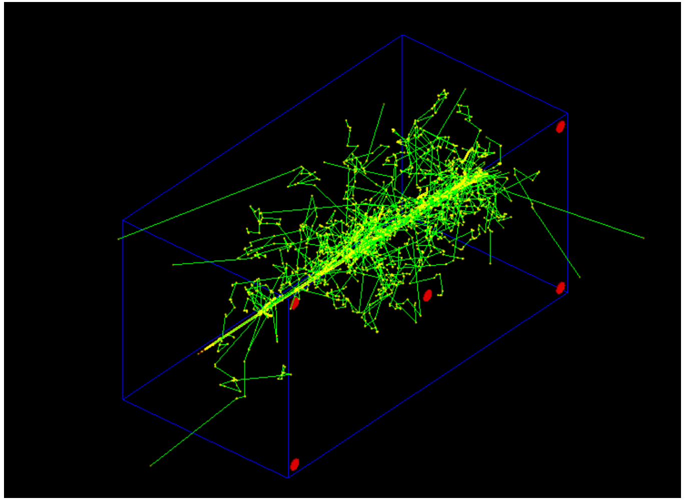
G4CXOpticks::simulate()
Generate and propagate Cerenkov and Scintillation photons

Rootfile: TTree

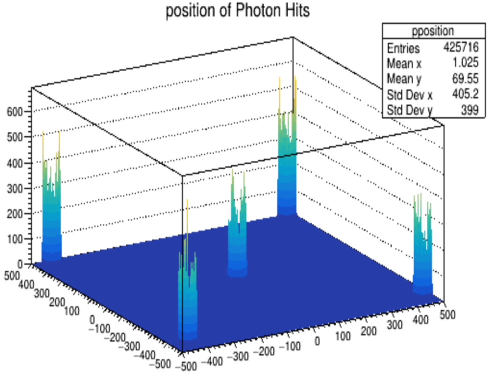
Event 1: PhotonHits. ......
Event 2: PhotonHits. ......
Event 3: PhotonHits. ......

Performance: (Legacy Opticks)

Hardware:	
CPU	Intel® Core i9-10900k@ 3.7 GHz, 10 CPU cores
GPU	NVIDIA GeForce RTX 3090 @ 1.7 GHz, 10496 cores
Software:	
Geant4: 11.0, Opticks based on OptiX® 6	



Number of CPU threads	**Single threaded. Geant4 [sec/evt]**	**Opticks [sec/evt]**	**Gain/speed up**
1	330	1.8	189x



→It becomes feasible to run full optical simulation event by event! But comparison is to single threaded Geant4→ somehow unfair! Single geant4 threat can saturate the GPU and doesn't allow the use of multiple CPU cores.

Plans

- Adjust to the new Opticks API → in progress. At the moment we keep compatibility with the legacy Opticks → propably drop this at some point.
- Once it’s working:
  - Benchmark the performance compared to legacy Opticks.
  - Compare Opticks with multithreaded Geant4 on multicore machines for fair comparison.
  - Profiling using nvprof.
  - Physics validation.
- Update the Geant4 advanced example CaTS:
  - Make part of the next Geant4 release.
  - make part of Geant4 continuous integration.
- In the process of integrating Opticks with artg4tk/larg4 to make it available to the LArTPC based experiments.
- Ensure physics is the same in Opticks and geant4.
- Add missing processes (wavelength shifting, various scintillation options to Opticks.)

Plans (cont.)

- Use the Root TBufferMerger for RootIO running Geant4 in multithreaded mode. Currently each thread opens its own file and the files are merged in the end.
- Integration of Geant4 with GPU
- Extend API to return Gensteps
- Modify processes to be more suited for GPU (header files) etc.
- Separate optical properties
- Optical materials database (crystals, scintillators, liquid noble gases)
- Allow to specify optical properties as a function (e.g. Sellmeier equation, emission spectra, Rayleigh scattering….)

- Extra slides

Plans (cont.)

- Use TBufferMerger for CaTS RootIO.
- Begin discussion with Geant4/Opticks developers on future collaboration. E.g.
    - Is there a way to avoid reimplementing the entire geant4 optical processes on the GPU? Could it be part of geant4? Provide the algorithm in a way suitable for GPU’s.
    - Allow for different ray tracers than Opticks to avoid dependency on nvidia hardware and code. → general API.

Collaboration meeting:

- Similar presentation to the previous one. But by then:
- Finished adjusting to the new Opticks API.
- Add benchmarking results using the new Opticks (profiling).
- Fair comparison of Opticks with multithreaded Geant4 with regard of CPU and memory use. (e.g. opticks.fnal.gov has 10 cores so with hyperthreading allows for 20 processes). → Summer student.
- Finished porting to gitlab to be ready for new Geaant4 release.
- Updated documentation (in progress)
- Talk to Simon