Geant4/CaTS/Opticks: optical photon propagation on a GPU

Hans Wenzel

Soon Yung Jun
Krzysztof Genser

Outline

- Motivation:
    - The computational challenge.
    - Simulation of optical photons: the ideal application to be ported to GPU's.
- Opticks/G4(CX)Opticks.
    - GenSteps.
- CaTS a stand alone Geant4 application.
    - CaTS workflow.
    - Performance.
- Recent developments
- artg4tk/larg4: What is what?
- Roadmap to integrate Opticks with LArSoft/lArG4.
- Opticks/CaTS resources, further reading.

CaTS: Calorimetry and Tracking Simulation

The computational challenge: Simulating a single 2GeV electron shower results in about 70 million VUV photons

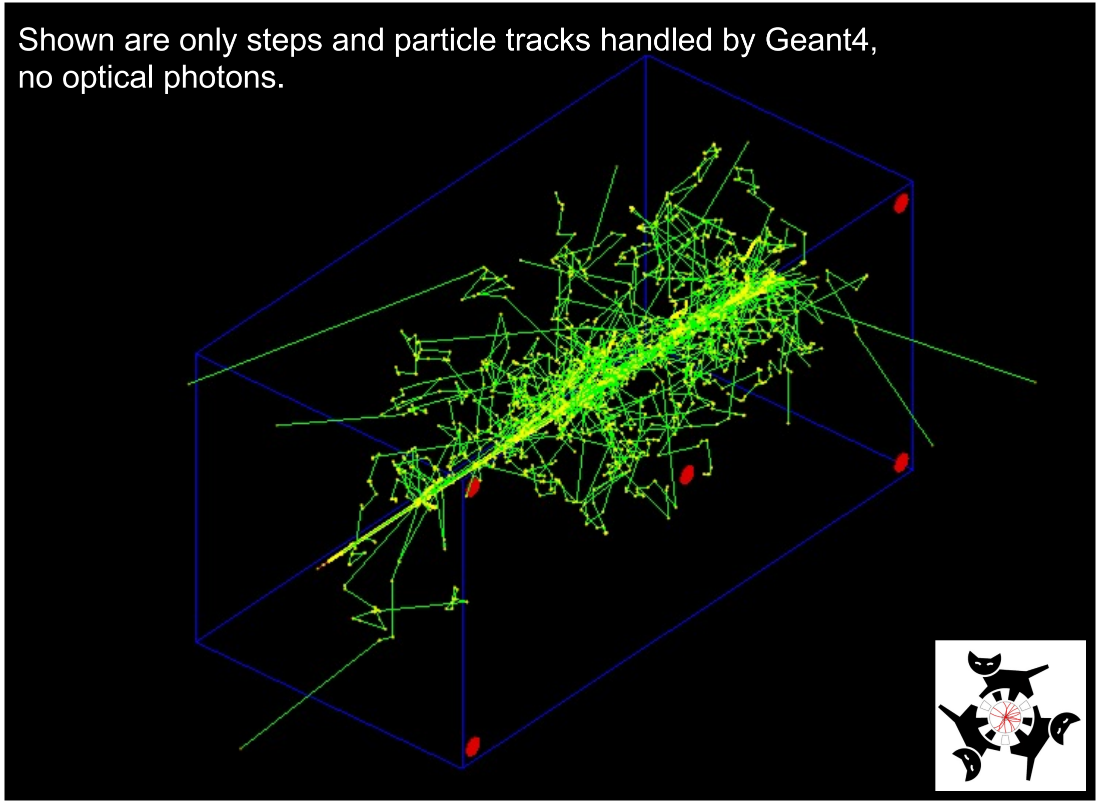
Simple Geometry:
Liquid Argon: x y z: 1 x 1 x 2 m (blue)
5 photo detectors (red)
photon yield (no E-field): 50000 γ/MeV single 2GeV electron (shower not fully contained)
(low Z=18, low $\rho = 1.78$ g/cm^3).

- 70 000 000 VUV scintillation photons are produced.
- Using Geant4 (11.1.p01) to simulate photon generation and propagation on the CPU takes:

~ 10 minutes/event

(Compared to 0.034 sec/event when no optical photon simulation)

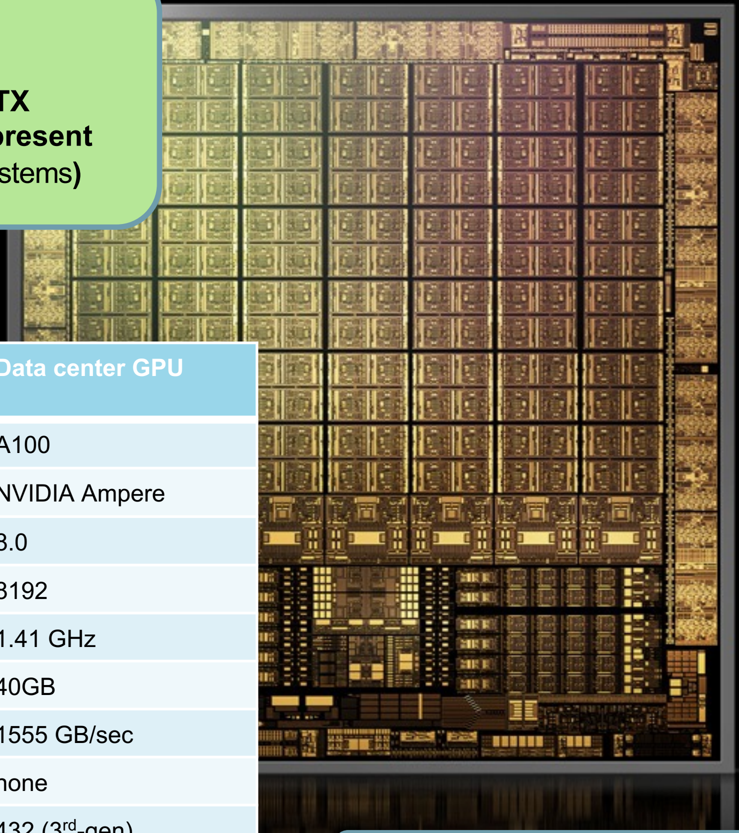
Shown are only steps and particle tracks handled by Geant4, no optical photons.

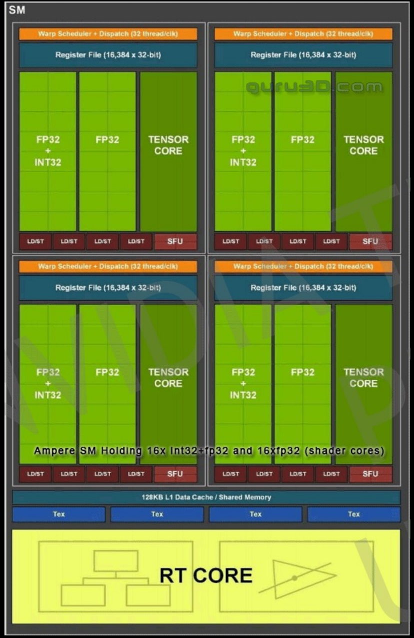


Simulation of optical photons: an ideal application to be ported to GPU’s.

- Only one particle type is involved (optical photon), but many of them (tens of millions)→ allows for massive parallelism (low latency, no big fluctuations in computing time).
- No new particles are produced when tracing the photons.
- Only a few simple physics processes need to be implemented on the GPU. Implementation on GPU is straightforward. The processes are:
  - G4Cerenkov (generate photons),
  - G4Scintillation (Reemission) (generate photons),
  - G4OpAbsorption,
  - G4OpRayleigh,
  - G4OpBoundaryProcess (only a few surface types),
  - G4OpWLS (not yet implemented, need it).
- 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 (RTX hardware acceleration now in the 4th generation but usually not present at HPC systems) and software (NVIDIA® CUDA, NVIDIA® OptiX ®).

Opticks will only run on:
NVIDIA® hardware and 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**)**




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

	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

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

Opticks

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/

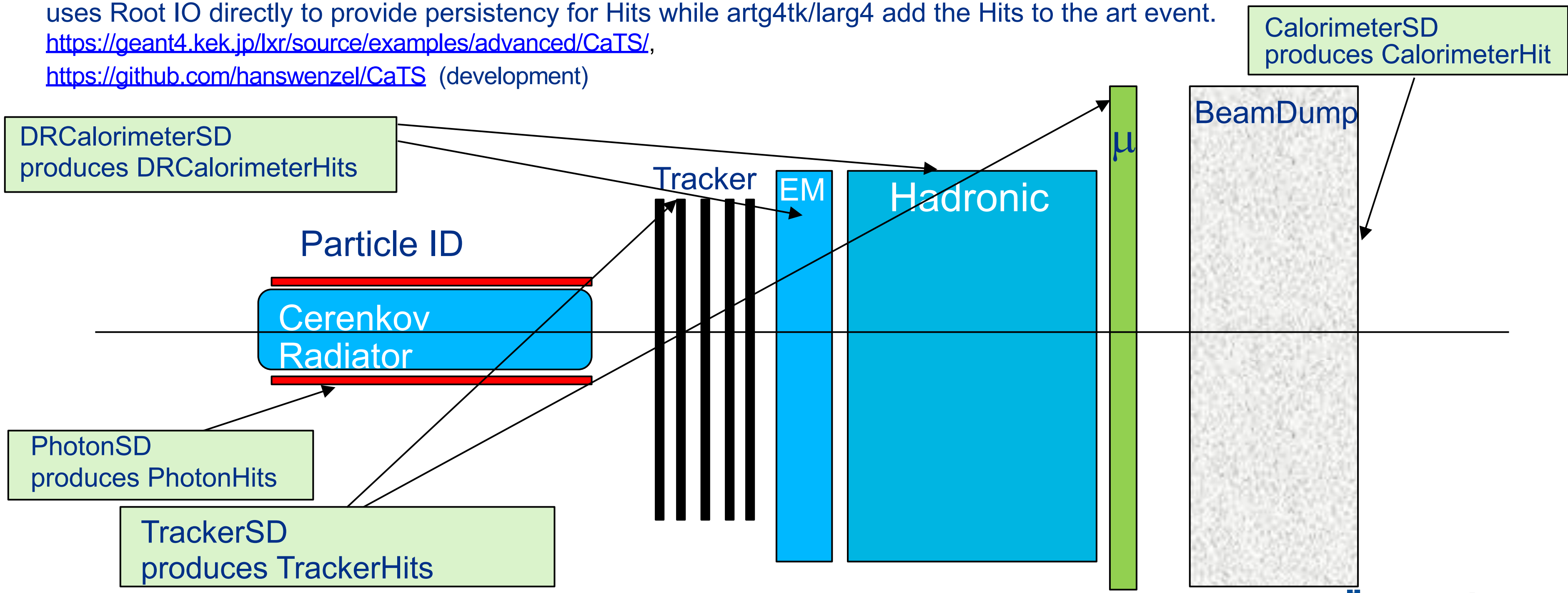
Two major versions: legacy Opticks based on OptiX ® 6 using the G4Opticks API and reengineered Opticks based on OptiX ® 7 using the G4CXOpticks API (work in progress).

In addition to ray tracing Opticks:

- implements the Geant4 optical processes on the GPU.
- Translates the Geant4 geometry to OptiX ® without approximation.
- G4(CX)Opticks provides an API. CaTS uses this API to implement a hybrid workflow where generation and tracing of optical photons is offloaded to Opticks (GPU/device) at stepping level whenever a certain amount photons is reached. Geant4 on the CPU/host handles all other particle types. The Geant4 Cerenkov and Scintillation processes are 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 (different for Cerenkov and Scintillation).

CaTS: Calorimeter and Tracker Simulation

CaTS: stand alone Geant4 application. It is a modular and extendible system that allows to build detector setups from predefined components. The creation of Hit collections and I/O thereof is automated once a sensitive detector is assigned to a logical volume. CaTS was included in Geant4 11.0. CaTS serves as a prototype/testbed for artg4tk and larg4 which share many functionalities e.g. same DetectorConstruction (gdml files are interchangeable), same mechanism to build physics. But CaTS uses Root IO directly to provide persistency for Hits while artg4tk/larg4 add the Hits to the art event.
https://geant4.kek.jp/lxr/source/examples/advanced/CaTS/,
https://github.com/hanswenzel/CaTS (development)



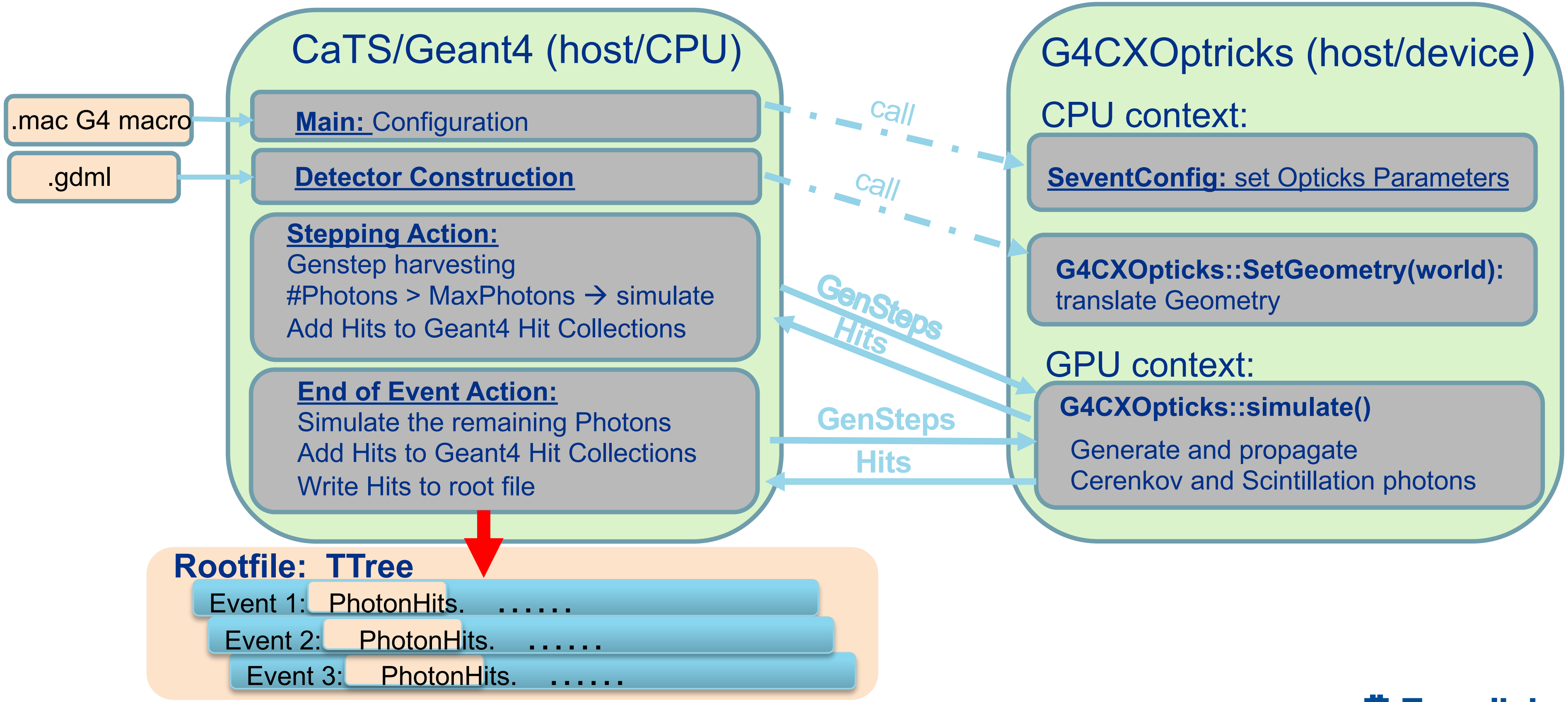
CaTS: Calorimeter and Tracker Simulation (cont.)

- No changes to Geant4 required! Only make use of provided interfaces: UserActions, Sensitive Detectors...
- Currently supports legacy/new Opticks interface.
- 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. Available:
        - lArTPCSD, TrackerSD, PhotondetectorSD, MscSD, CalorimeterSD, DRCalorimeterSD,….
    - Assigning step-limits, energy cuts to logical Volume.
    - Assigning visualization attributes.
- Allows to visualize the detector using Geant4 visualization tools.
- Uses G4PhysListFactoryAlt to define and configure physics at runtime via command line option.

```
./CaTS -g simpleLArTPC.gdml -pl 'FTFP_BERT+OPTICAL+STEPLIMIT' -m time.mac
```

- G4(CX)Opticks/Geant4 is a runtime/build time option.
- Uses Geant4 to collect Scintillation and Cerenkov Gensteps. The number of photons to be generated is always calculated by Geant4 even when using Opticks!

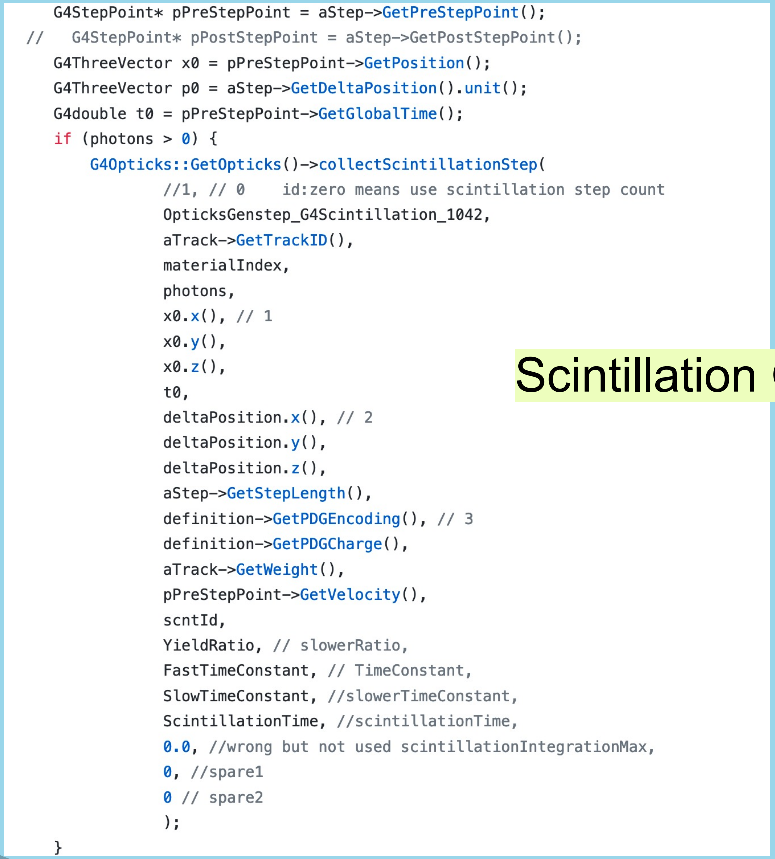
CaTS workflow using the new version of Opticks based on OptiX®7:



GenSteps

A GenStep collects all information necessary to generate Scintillation and Cerenkov photons on the GPU → more efficient than copying optical photons

Scintillation GenStep:

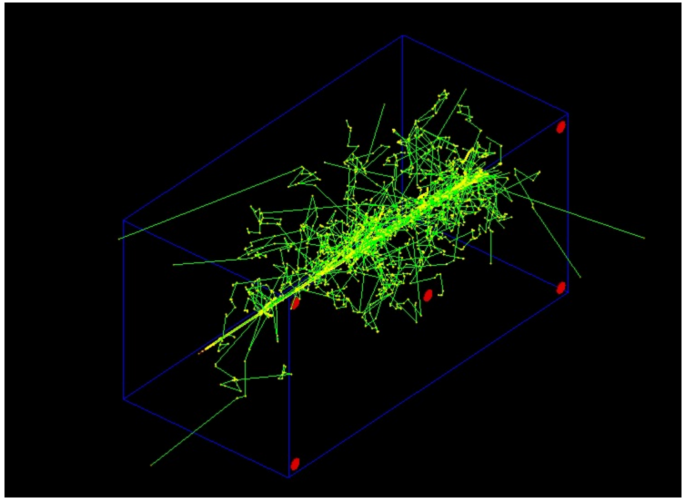


```
    G4StepPoint* pPreStepPoint = aStep->GetPreStepPoint();
//   G4StepPoint* pPostStepPoint = aStep->GetPostStepPoint();
    G4ThreeVector x0 = pPreStepPoint->GetPosition();
    G4ThreeVector p0 = aStep->GetDeltaPosition().unit();
    G4double t0 = pPreStepPoint->GetGlobalTime();
    if (photons > 0) {
        G4Opticks::GetOpticks()->collectScintillationStep(
                //1, // 0    id:zero means use scintillation step count
                OpticksGenstep_G4Scintillation_1042,
                aTrack->GetTrackID(),
                materialIndex,
                photons,
                x0.x(), // 1
                x0.y(),
                x0.z(),
                t0,
                deltaPosition.x(), // 2
                deltaPosition.y(),
                deltaPosition.z(),
                aStep->GetStepLength(),
                definition->GetPDGEncoding(), // 3
                definition->GetPDGCharge(),
                aTrack->GetWeight(),
                pPreStepPoint->GetVelocity(),
                scntId,
                YieldRatio, // slowerRatio,
                FastTimeConstant, // TimeConstant,
                SlowTimeConstant, //slowerTimeConstant,
                ScintillationTime, //scintillationTime,
                0.0, //wrong but not used scintillationIntegrationMax,
                0, //spare1
                0 // spare2
                );
    }
```

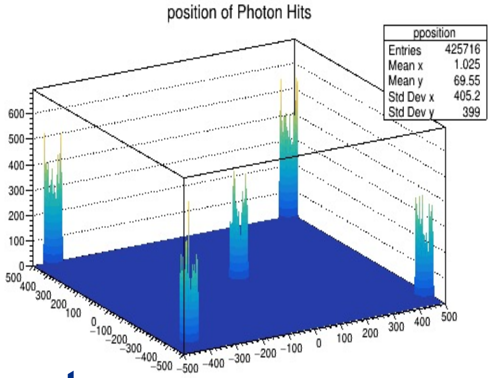
Performance:

Depending on Geometry

Hardware:	
CPU	Intel(R) Core Core i9-10900k@ 3.7Ghz 10 CPU cores
GPU	GeForce RTX 3090
Geant4: 11.0, Opticks based on OptiX® 6	



Number of CPU threads	**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

Recent developments

Re-implementing Opticks for OptiX ® 7[1] required huge changes due to the new and very different OptiX ® 7 API → So good time to rethink the simulation code. Goals of re-implementation: flexible, modular GPU simulation, easily testable, less code.

Opticks: Completed

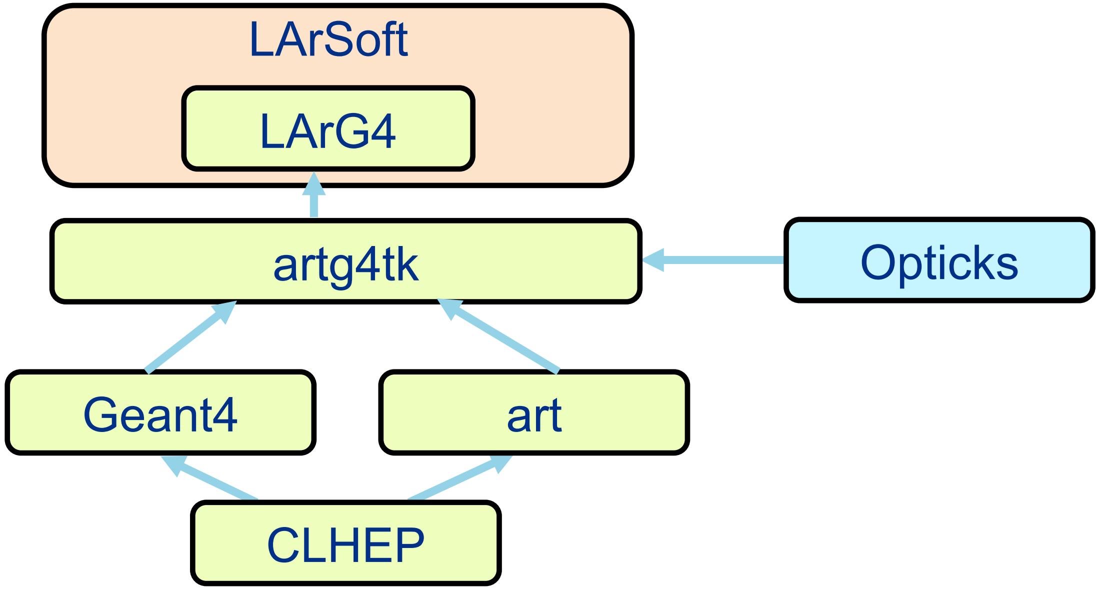
- Full Simulation re-implementation for OptiX ® 7 API.
- Many packages were removed or are planned to be removed.
- Move code that doesn't require OptiX or Cuda out of GPU context.
- Rather monolithic .cu was replaced by small GPU+CPU headers.
- Changed Opticks to work with Geant4 >11. (new API for optical processes and material properties).

- CaTS has been modified to use the new Opticks based on the new OptiX API. The CaTS workflow has been adjusted accordingly. User actions were utilized → no changes to Geant4 itself required.
- With legacy versions of Opticks (based on Optix 6) we observed speed ups in the order of 2x10^2 . Evaluation and optimizing the performance with recent updates is in progress.

[1] https://simoncblyth.bitbucket.io/env/presentation/Opticks_20220718_towards_production_use_juno_collab_meeting.html

artg4tk/larg4: What is what?

artg4tk: general Geant4 module for art, it depends only on art and Geant4. The Geant4 data objects (e.g. Hits) are added to the art event record (started with artg4 as developed for the g-2 exp.).
https://github.com/art-framework-suite/art-g4tk



Larg4 is based on artg4tk. It is a module for lArSoft. All LArSoft dependencies (e.g. lardataobjects) are encapsulated here: https://github.com/LArSoft/larg4

Roadmap to integrate Opticks with LArSoft/lArG4

- Build CaTS (with Opticks) on Wilson Cluster (WC/IC) using the most recent UPS/UPD Geant4 product (done)→ Make sure it runs on the WC GPU nodes.
- Create release of artg4tk based on the new versions of Geant4/art.
- Add necessary G4CXOpticks calls to artg4tk.
- Products like cuda, Optix, Opticks will not be available as ups/upd products (moving to spack) so will have to be integrated as external products.
- Implement Wavelength shifting process (WLS) in Opticks.
- Evaluate if more efficient random number management on the GPU can be implemented.

further reading

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/201921402027
Simon Blyth:
"Opticks : GPU Optical Photon Simulation for Particle Physics using NVIDIA® OptiX^{TM} "

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 (new Opticks)
https://github.com/simoncblyth/opticks : used for snapshots and tagged releases.
The most recent tag is https://github.com/simoncblyth/opticks/releases/tag/v0.1.7
→ Starting point from 'our' github fork.

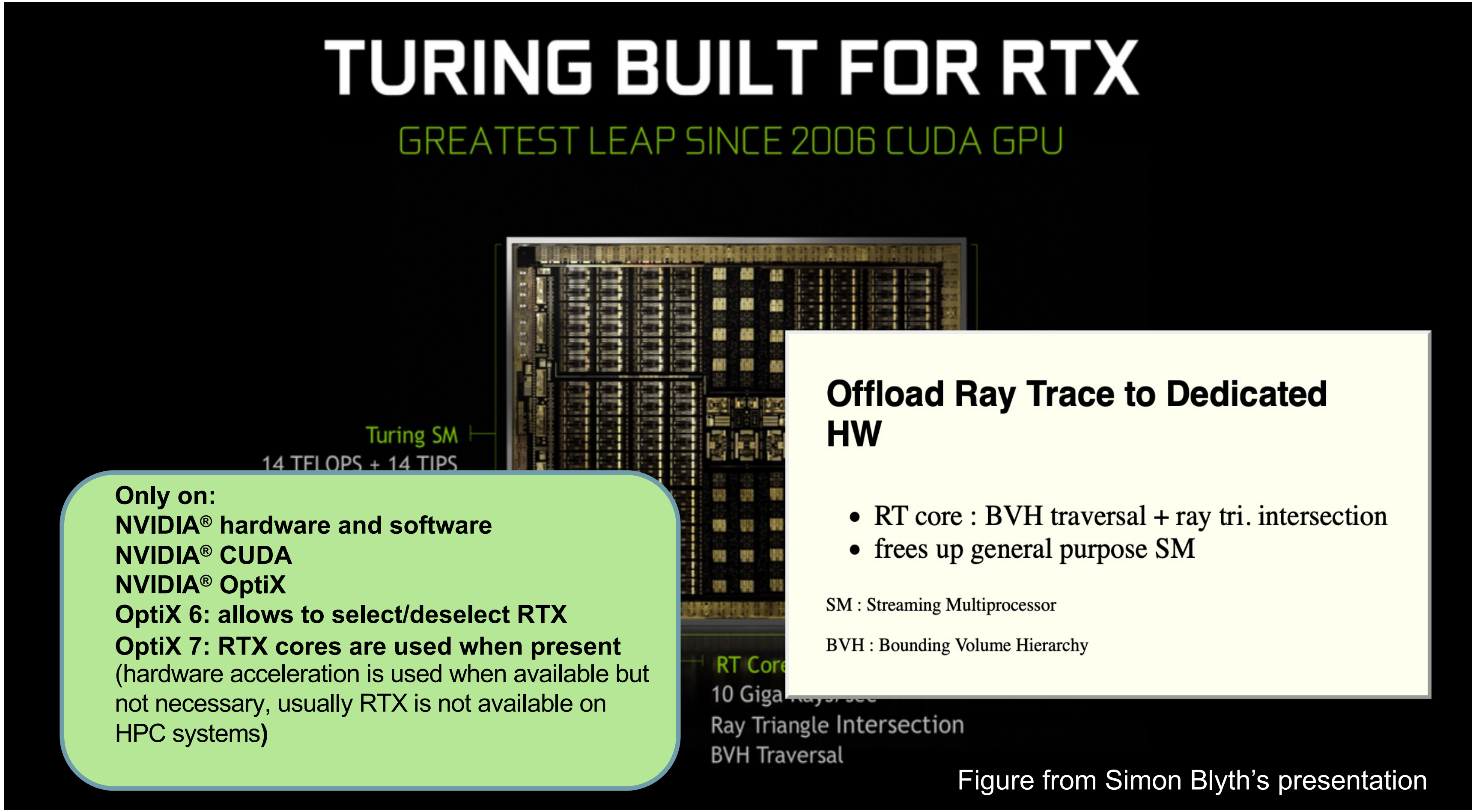
CaTS/Opticks resources

- 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
- Instructions how to build Opticks and how to install all necessary software:

  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://gitlab.cern.ch/geant4/geant4 : will be used for snapshots and tagged releases

Fermilab

Backup slides



Only on:
NVIDIA® hardware and software
NVIDIA® CUDA
NVIDIA® OptiX
OptiX 6: allows to select/deselect RTX
OptiX 7: RTX cores are used when present (hardware acceleration is used when available but not necessary, usually RTX is not available on HPC systems)

Offload Ray Trace to Dedicated HW

- RT core : BVH traversal + ray tri. intersection
- frees up general purpose SM

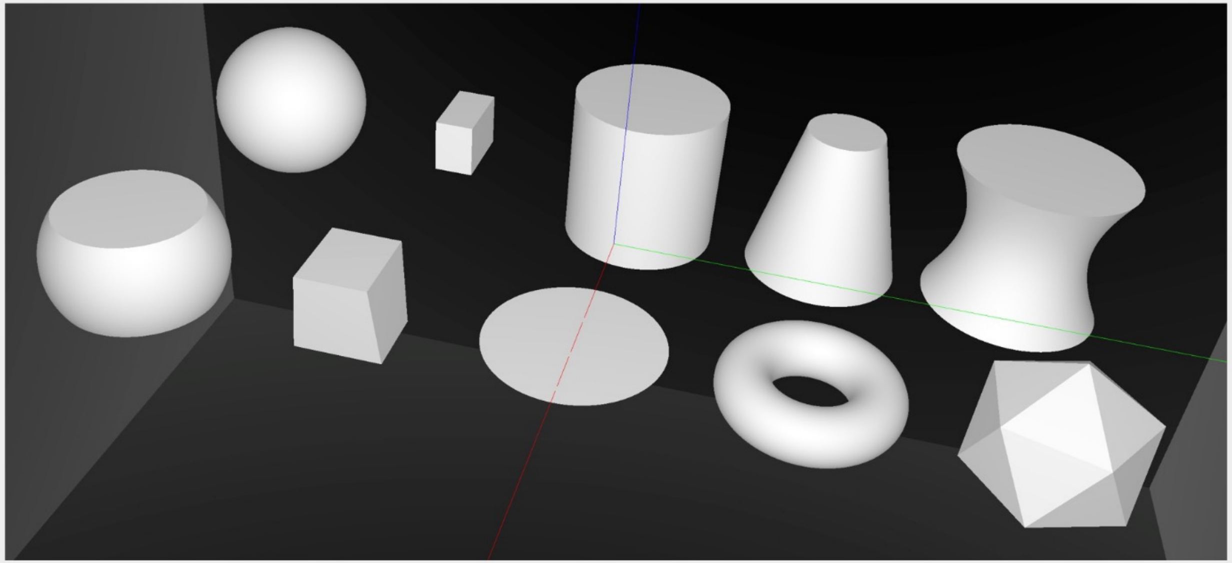
SM : Streaming Multiprocessor

BVH : Bounding Volume Hierarchy

Figure from Simon Blyth's presentation

G4VSolid -> CUDA Intersect Functions for ~10 Primitives

- 3D parametric ray : **ray(x,y,z;t) = rayOrigin + t * rayDirection**
- implicit equation of primitive : **f(x,y,z) = 0**
- -> polynomial in **t** , roots: **t > t_min** -> intersection positions + surface normals



Sphere, Cylinder, Disc, Cone, Convex Polyhedron, Hyperboloid, Torus, ...

Plans concerning CaTS/Opticks

CaTS:

- Make the latest developments and documentation available in the Geant4/CaTS advanced example.
- Achieve true concurrency by using G4Tasking. Allow to configure jobs to fully utilize CPU and GPU resources.
- Change to use in-memory Root file merging (TBufferMerger) when using multi-threading.

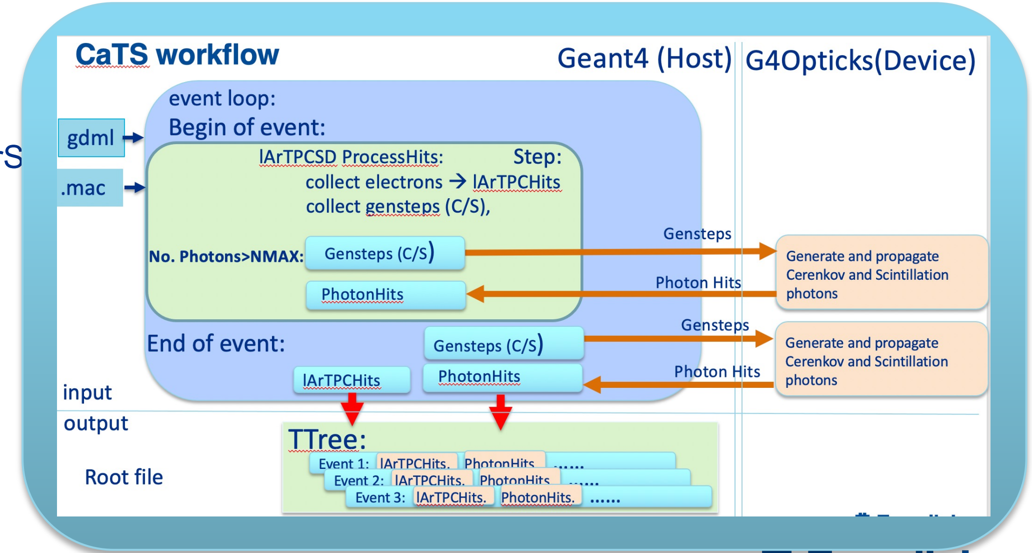
G4CXOpticks/Opticks:

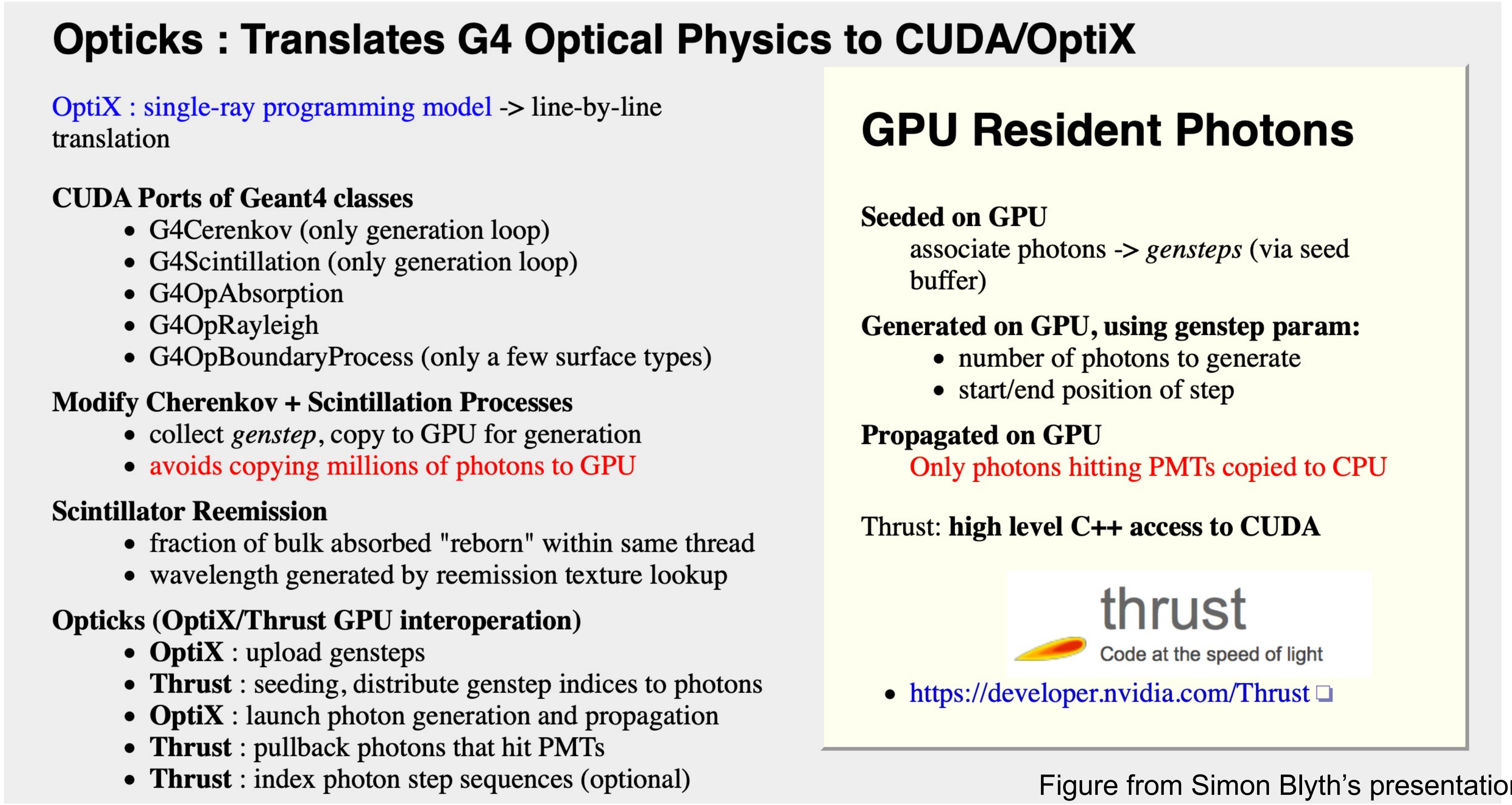
- Use the same implementation of the scintillation process on CPU (Geant4) and GPU (Opticks), use the same optical properties/keywords.
- Implement Wavelength shifting process (WLS).

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/lArTPCSD). 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 runt-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. Available:
    - **RadiatorSD**, **lArTPCSD**, **PhotonSD.**
    - TrackerSD, CalorimeterSD, DRCalorimeterS
  - 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.





Opticks : Translates G4 Optical Physics to CUDA/OptiX

OptiX : single-ray programming model -> line-by-line translation

CUDA Ports of Geant4 classes

- G4Cerenkov (only generation loop)
- G4Scintillation (only generation loop)
- G4OpAbsorption
- G4OpRayleigh
- G4OpBoundaryProcess (only a few surface types)

Modify Cherenkov + Scintillation Processes

- collect *genstep*, copy to GPU for generation
- avoids copying millions of photons to GPU

Scintillator Reemission

- fraction of bulk absorbed "reborn" within same thread
- wavelength generated by reemission texture lookup

Opticks (OptiX/Thrust GPU interoperation)

- **OptiX** : upload gensteps
- **Thrust** : seeding, distribute genstep indices to photons
- **OptiX** : launch photon generation and propagation
- **Thrust** : pullback photons that hit PMTs
- **Thrust** : index photon step sequences (optional)

GPU Resident Photons

Seeded on GPU
associate photons -> *gensteps* (via seed buffer)

Generated on GPU, using genstep param:

- number of photons to generate
- start/end position of step

Propagated on GPU
Only photons hitting PMTs copied to CPU

Thrust: **high level C++ access to CUDA**

thrust
Code at the speed of light

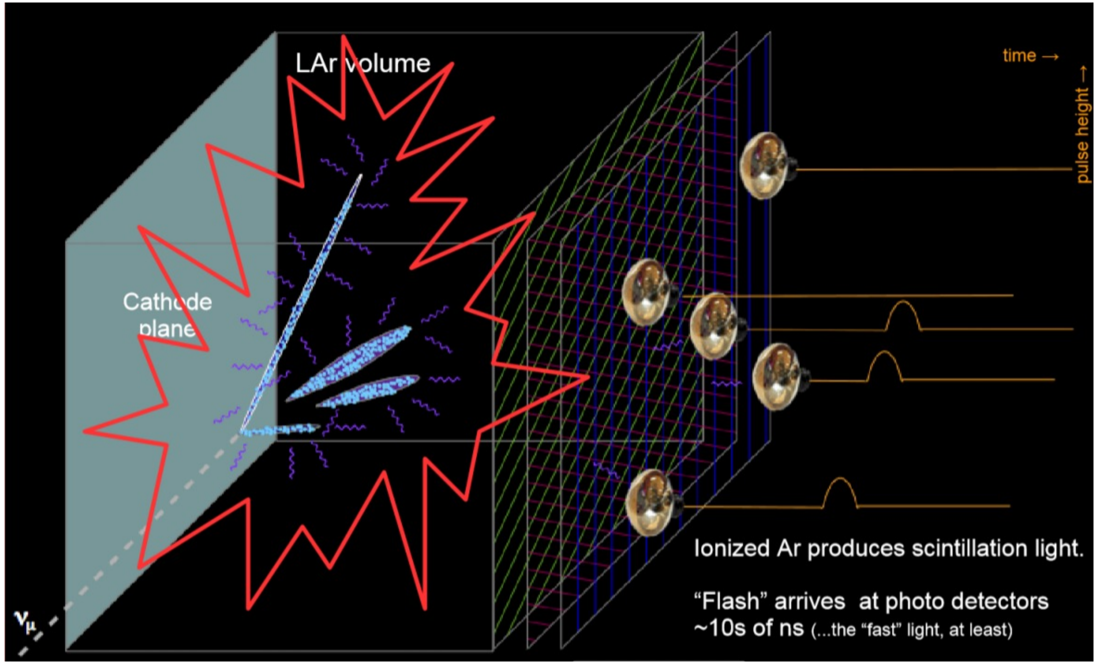
- https://developer.nvidia.com/Thrust ❏

Figure from Simon Blyth's presentation

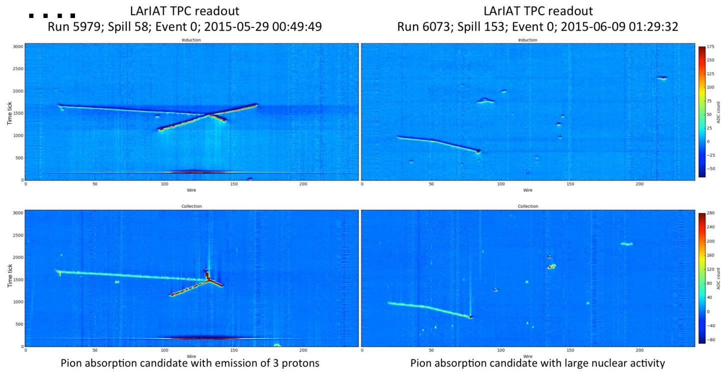
For liquid Argon TPC we need: wave-length shifting process on GPU

Motivation

Liquid Argon TPC's are the technology of choice for many neutrino experiments: DUNE, protoDUNE, μBoone, LArIAT, ICARUS, SBND… as well as dark matter searches: DarkSide, ARGO, …


LAr volume
Cathode plane
ν_μ
time →
pulse height →
Ionized Ar produces scintillation light.
"Flash" arrives at photo detectors ~10s of ns (...the "fast" light, at least)

The image below shows a display of the ionization signal: time + stereo wires or pixels allows for 3D reconstruction
dEdx for PID
track length for energy/momentum
….



See: Dorota Stefan(CERN/NCBJ (Warsaw PL)), https://indico.cern.ch/event/575069/contributions/2326563/attachments/1363382/2064171/LArPrinicipals.pdf