FERMILAB-SLIDES-20-119-SCD

Fermilab U.S. DEPARTMENT OF ENERGY Office of Science

Using Geant4 and Opticks[a] to simulate liquid Argon TPC's

Hans Wenzel, Krzysztof Genser ,Soon Yung Jun, Alexei Strelchenko

Fermilab

HSF/WLCG workshop

Nov. 19-24th 2020

This manuscript has been authored by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics.

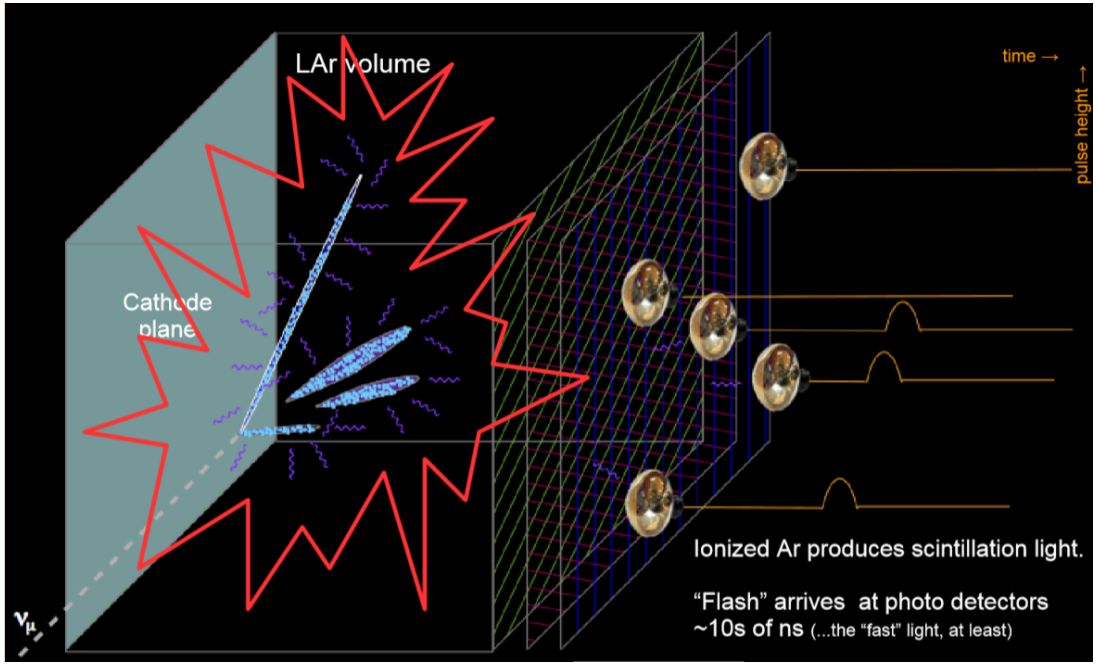
GEANT4 A SIMULATION TOOLKIT

[a] Author: Simon Blyth

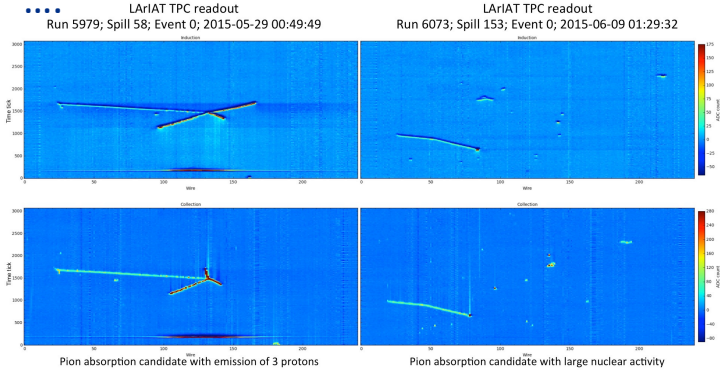
Outline

- Motivation: physics of liquid Argon TPC's.
- Opticks/G4Opticks
- Prototype example: G4OpticksTest
- Performance
- Summary and Plans

Single-phase liquid Argon TPC (lArTPC)



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



Pion absorption candidate with emission of 3 protons

Pion absorption candidate with large nuclear activity

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

Fermilab

Scintillation/Ionization properties in liquid Argon[1]

Light yield ~ few 10,000's of photons per MeV (dependences on E field, particle type and purity)

where minimum ionization is 2.105 MeV/cm

Wavelength of emission is 128nm

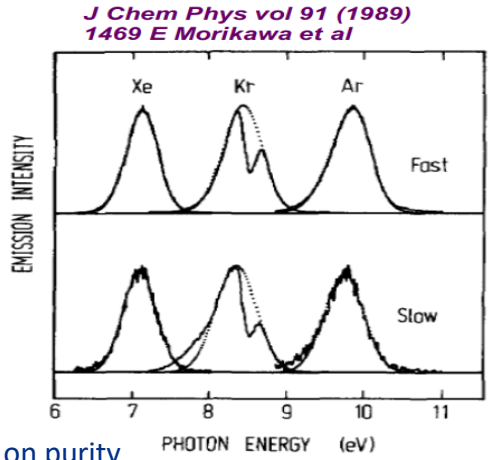
VUV → wavelength shifting required for detection

Light with two characteristic time constants:

- *fast component, 6 ns*
- *slow component, 1500 ns*

Argon is highly transparent to its own scintillation light.

→ Absorption length ~ tenth of meters depending on purity



Rayleigh scattering length ~ 50-60cm.

Complex Scintillation process through Ionization

→ Argon can't be excited by its own scintillation light

Presence of electric field → allows for separation of ions and electrons created in Ionization process:

- Electrons drift to read out plane and produce signal.
- Slow moving Ions might contribute to space charge effects.
- Ionization and scintillation light are 'anti'- correlated in a complicated way.
- Photons are generated and traced by Opticks. Detection requires wavelength shifting from VUV to visible light. Photons hitting a Photodetector are returned as Photon Hits.
- Electrons are collected in Hit collection and then traced by separate module.

[1] Adapted from Ben Jones: https://microboone-exp.fnal.gov/public/talks/LArTPCWorkshopScintLight_bjpjone_2014.pdf

Motivation

- Using Geant4 to simulate photon propagation on the CPU takes ~ hours to simulate 1 event for a typical liquid Argon TPC.
- Currently lArTPC experiments use Look Up Tables (LUT) or parameterizations for photon response. But:
  - only approximation, information not complete,
  - LUT grow with detector size to a point that jobs can't run on a typical grid node,
  - still need full simulation to create the tables to run on traditional grids.
- Simon Blyth showed that Opticks speeds up the photon simulation to the level where it is comparable to the rest of the simulation:
  - Allows to run full optical simulation event by event.
  - Allows to investigate various ideas for improvements like:
    - Improve calorimetric energy resolution,
    - Using ratios like fast/slow scintillation light or light/ionization for Particle ID,
    - ….
- Not only Argon TPCs but also many experiments can benefit e.g. dark matter searches, dual readout calorimeters (e.g. Crystals: effect of Cerenkov light directionality, different TPC configurations …), various groups are investigating Opticks → we aim to develop a flexible framework where detector geometry/configuration, physics list, output etc. are determined at runtime and don't require recompilation.

  (like artg4tk/larsoft see: https://cdcvs.fnal.gov/redmine/projects/artg4tk/wiki/Artg4tk).

Opticks

JUNO https://simoncblyth.bitbucket.io/env/presentation/opticks_may2020_hsf.html[1]

EPJ Web of Conferences **214**, 02027 (2019)
https://doi.org/10.1051/epjconf/201921402027

Opticks : GPU Optical Photon Simulation for Particle Physics using NVIDIA® OptiX™

Simon Blyth

Huge CPU Memory+Time Expense

JUNO Muon Simulation Bottleneck
~99% CPU time, memory constraints

Ray-Geometry intersection Dominates
simulation is not alone in this problem...

Optical photons : naturally parallel, simple :
- produced by Cherenkov+Scintillation
- yield only Photomultiplier hits

[1)]Figure from Simon's presentation

Opticks and NVIDIA®

Only on:
NVIDIA® hardware and software
NVIDIA® CUDA
NVIDIA® OptiX™

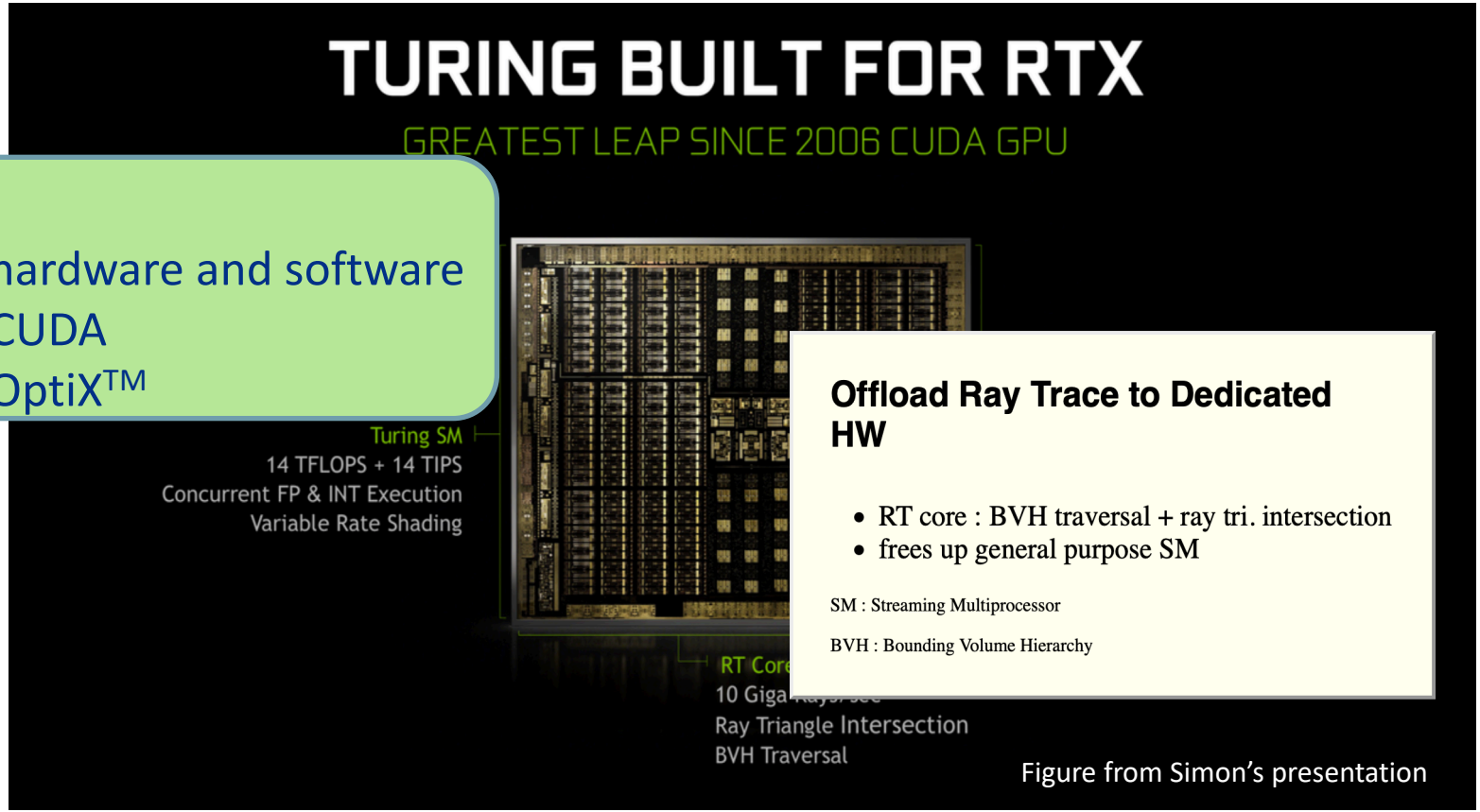


Figure from Simon's presentation

Opticks Open source: https://bitbucket.org/simoncblyth/opticks/

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**

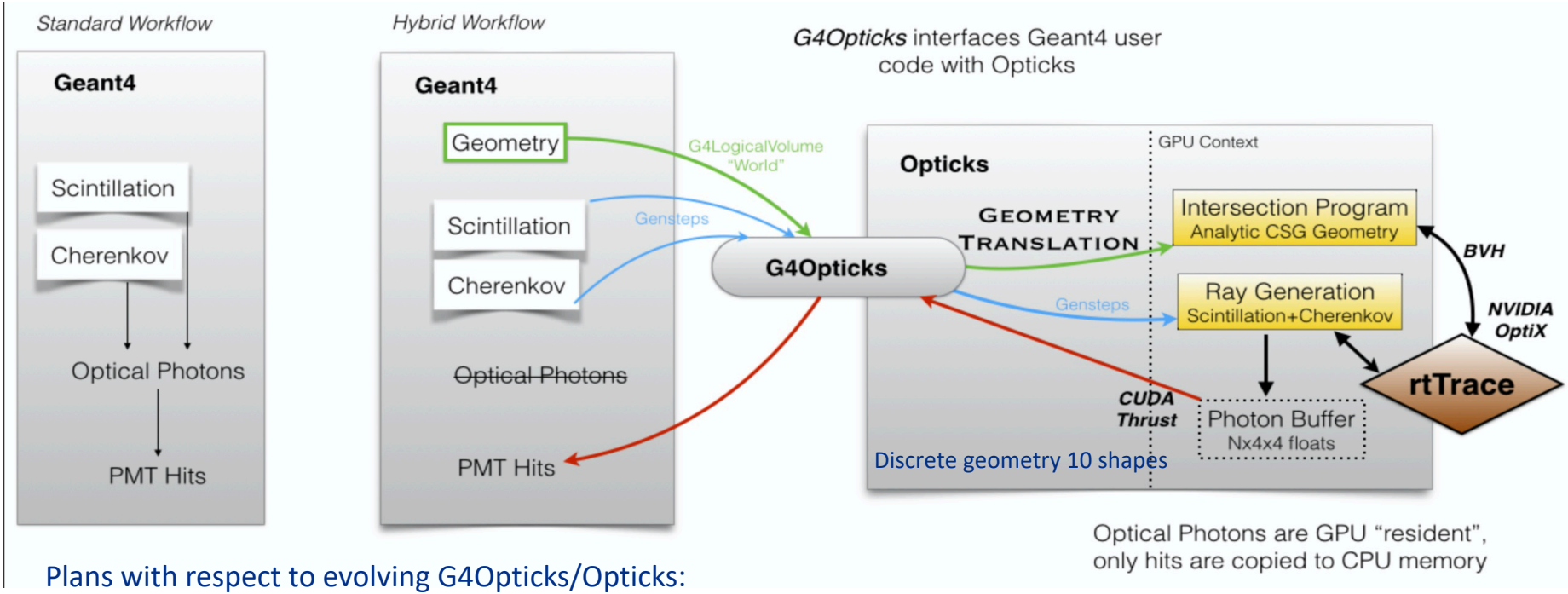
- https://developer.nvidia.com/Thrust

Figure from Simon's presentation

Need: wave-length shifting(WLS) process on GPU (CUDA) similar to reemission!

G4Opticks hybrid workflow

Figure from Simon's presentation



Plans with respect to evolving G4Opticks/Opticks:

- Use the same implementation of the scintillation process on CPU and GPU, use the same optical properties.
- Implement WLS.
- Currently G4Opticks is invoked at Geant4 End of Event. Achieve true concurrency by using G4Tasking by J. Madsen. (available in next Geant4 release: 10.7)

G4OpticksTest:

- Geant4 Application demonstrating the use of the G4Opticks hybrid workflow,
  - we plan to make it a Geant4 advanced example
  - code available at https://github.com/hanswenzel/G4OpticksTest

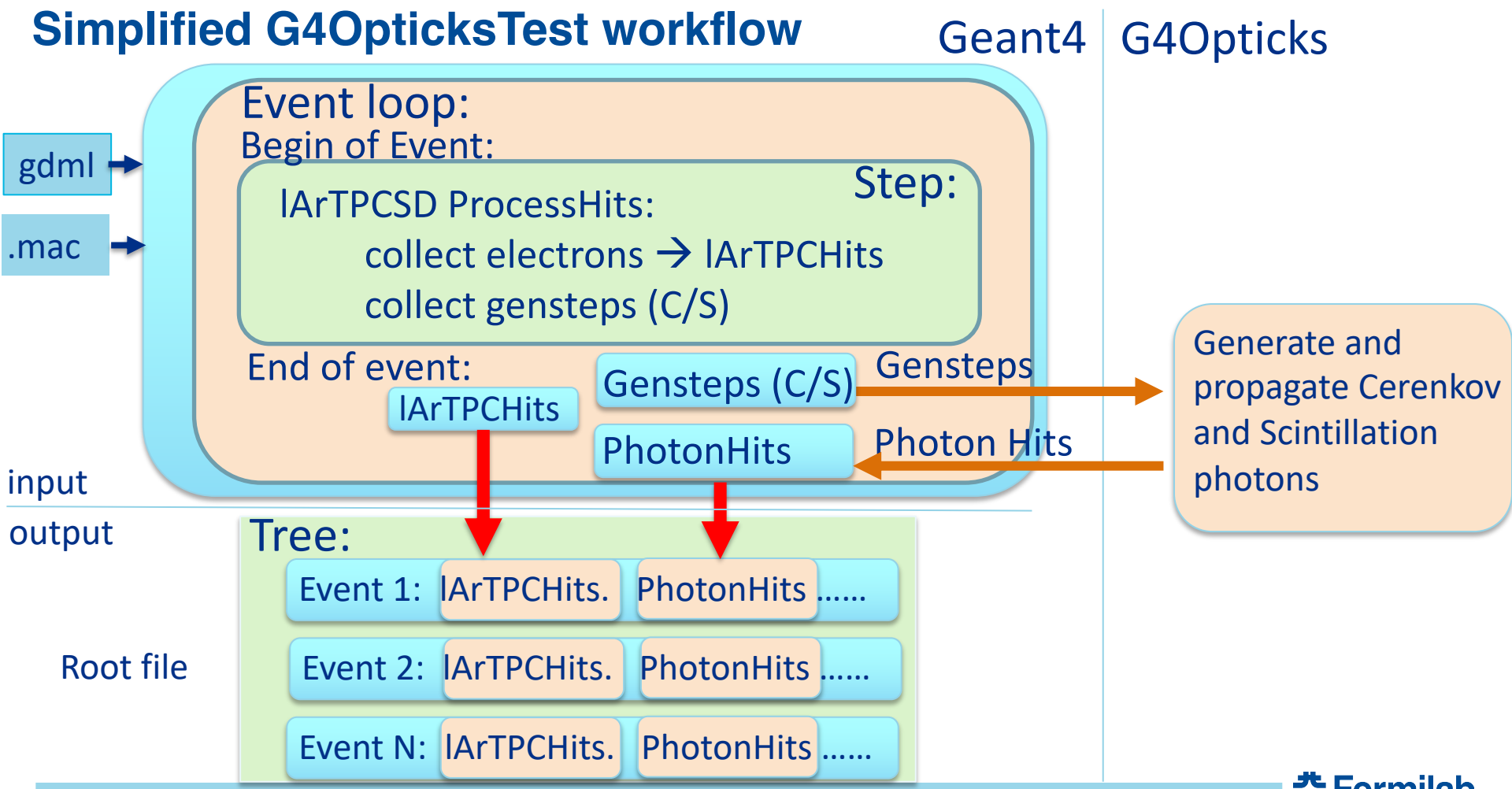
Features are:

- Uses Geant4 to harvest Scintillation and Cerenkov Gensteps[1]. The harvesting is moved to sensitive Detectors/UserSteppingAction.
- Uses Opticks to generate and propagate optical photons, returns Photon-detector Hits.
- Uses gdml with extensions for flexible Detector construction and to provide optical properties.
  The gdml extensions include:
  - Assigning sensitive detectors to logical Volumes. A library of various sensitive detector types is provided (specifically lArTPCSD, but also detectors for TOF, Muon detectors etc.).
  - Assigning step-limits to logical Volume to match Geant4 steps and TPC readout pitch.
  - Assigning visualization properties.
  - Assigning homogenous electric field.

  Make use of formulae, variables, loops, predefined NIST materials →compact, human readable gdml.
- Uses G4PhysListFactoryAlt (R. Hatcher) to define and configure physics: e.g.
  G4PhysListRegistry::GetModularPhysicsList <FTFP_BERT+OPTICAL+STEPLIMIT+NEUTRONLIMIT>
- Uses Root IO to provide persistency.

[1)] See backup slides

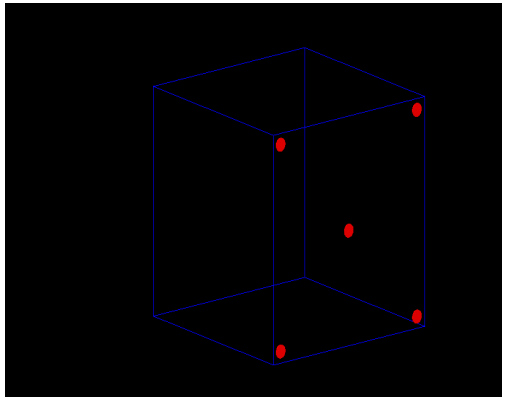
Simplified G4OpticksTest workflow



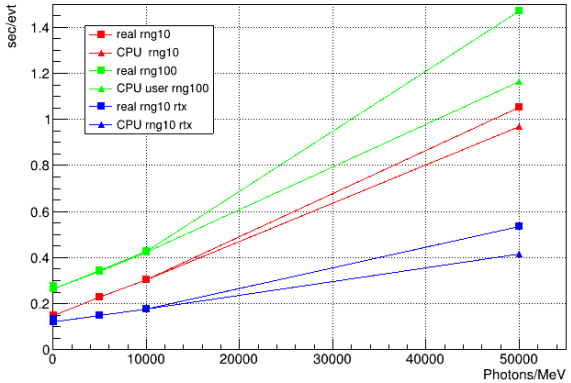
Performance: very preliminary!

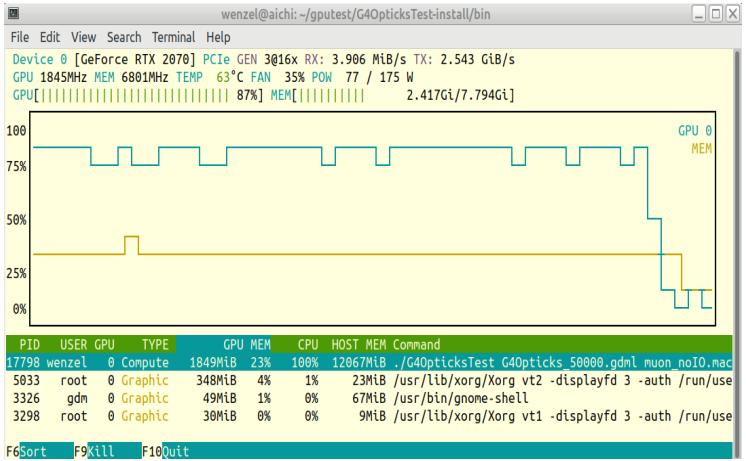
CPU	Intel(R) Core i7-9700K 3.6GHz
GPU	GeForce RTX 2070" CUDA Driver Version /11.0 CUDA Capability: 7.5 VRAM: 7981 Mbytes Cores: 2304

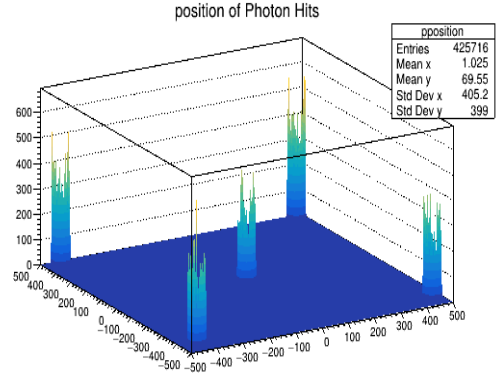
Simple Geometry:
Liquid Argon: 1 m^3
5 photo detectors (red)
photon yield varied from 100-50000 Photons/MeV
single 1GeV muon



G4OpticksTest Benchmarking results







Compare to ~ hour/evt using Geant4 on CPU, RTX -> 2x

Summary and Plans

Summary:

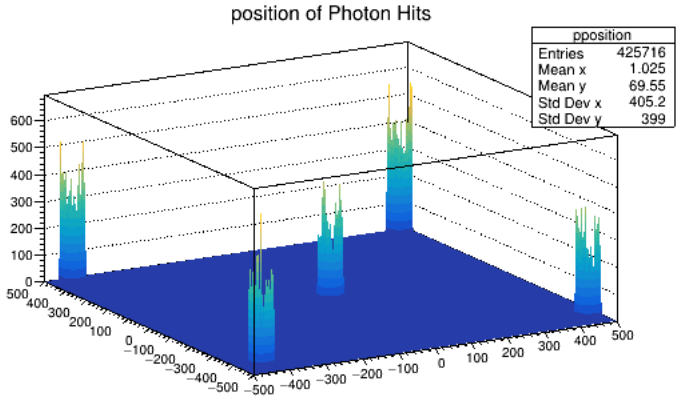
- We made Simon Blyth's Opticks work with Geant4 10.6.p03.
- A prototype application: G4OpticksTest is available and we plan to make it an advanced Geant4 example. Various geometries are available as gdml.
- Preliminary benchmarking looks very promising.

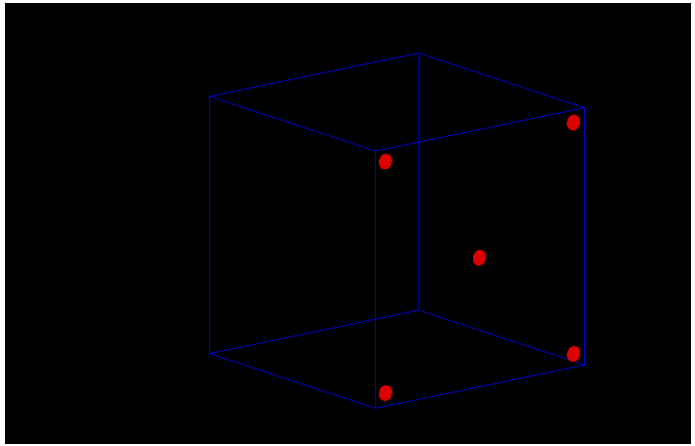
Plans:

- Implement current Geant4 Scintillation process on GPU.
- Implement wavelength shifting process on GPU.
- Develop Geant4 Task application with Opticks where:
  - Gensteps chunks are collected in-situ during the tracking/stepping loop, once a predetermined chunk size is reached the optical photon propagation is offloaded to the GPU (device), while the rest of simulation and gensteps collection continues on the CPU (host).

special thanks to Simon Blyth!!

Backup Slides


position of Photon Hits
pposition
Entries 425716
Mean x 1.025
Mean y 69.55
Std Dev x 405.2
Std Dev y 399



Gensteps (Scintillation and Cerenkov)

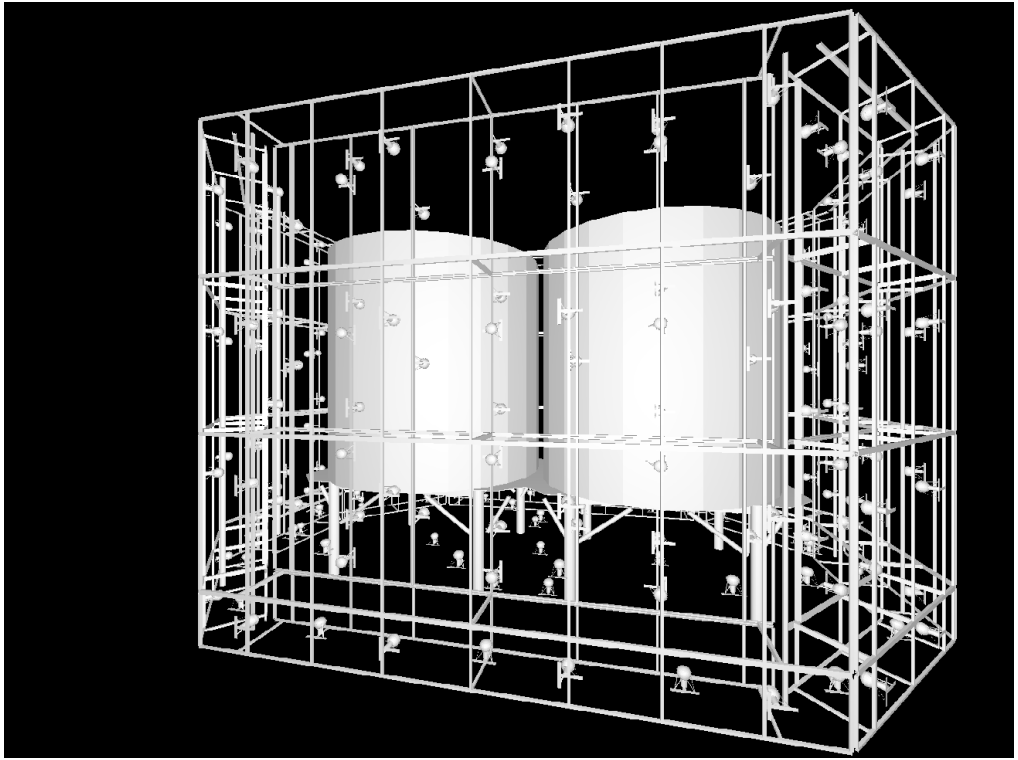
A Genstep collects all information necessary to generate Scintillation and Cerenkov photons on the GPU.

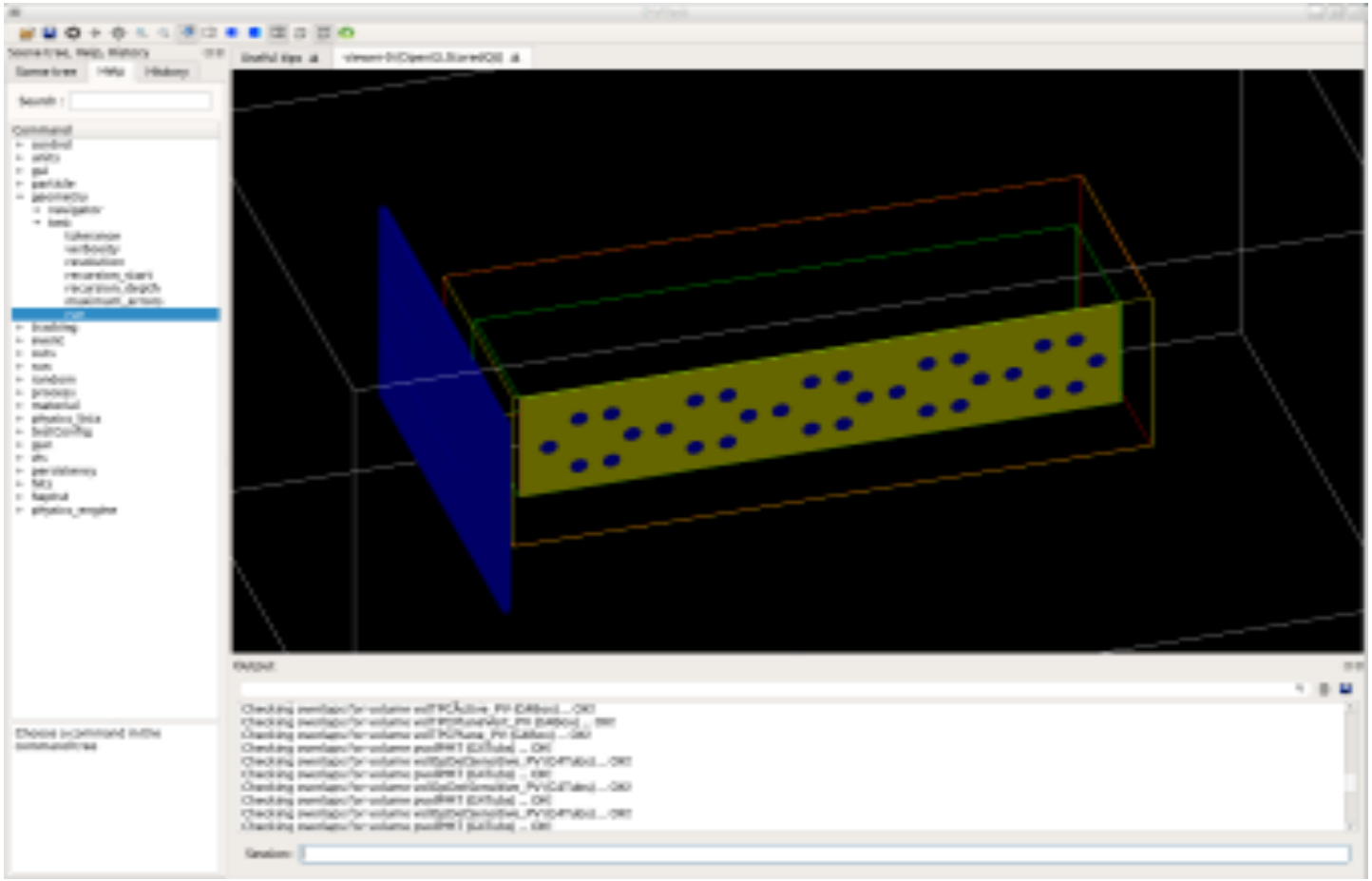
In order to simplify G4Opticks code and to decrease the need for future maintenance, we modified Geant4 10.7 to include small changes to the G4Cerenkov API to retrieve all needed information to generate and propagate Cerenkov photons with Opticks. No changes were necessary for Scintillation photons.

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
               );
   }
```

Visualization of Daya Bay Detector





G4Solid -> 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

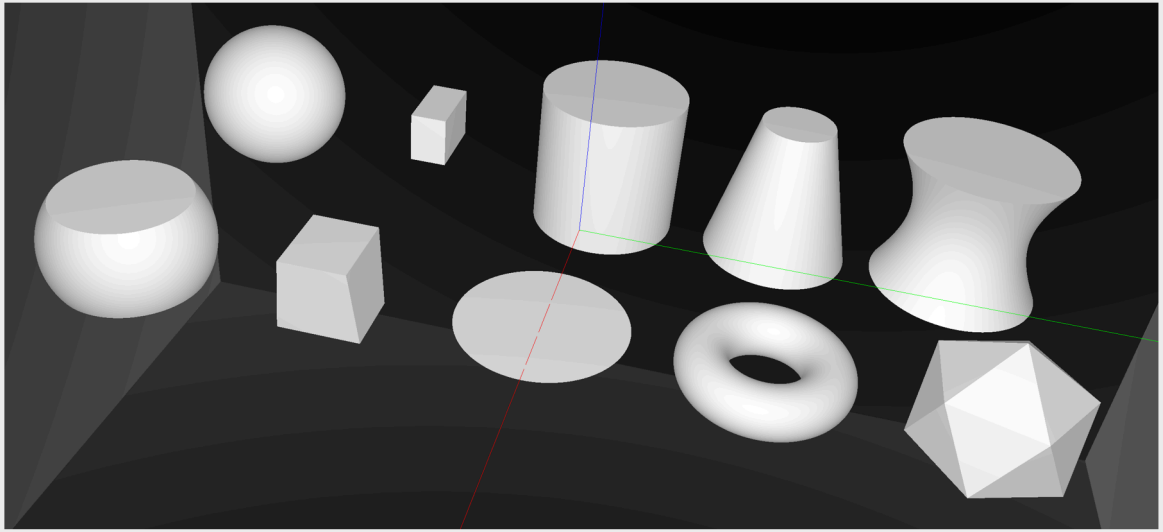
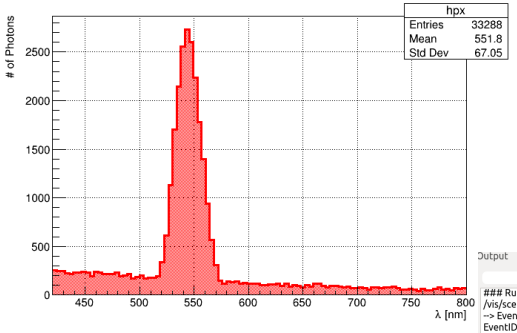
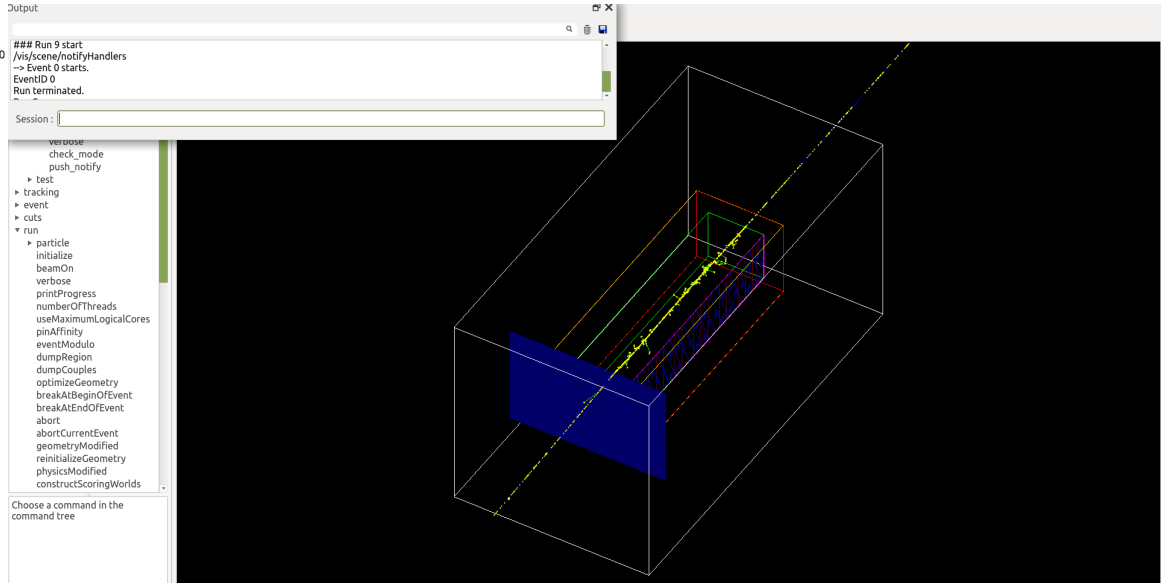


Figure from Simon's presentation

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


hpx
Entries 33288
Mean 551.8
Std Dev 67.05
# of Photons
λ [nm]


Output
Run 9 start
/vis/scene/notifyHandlers
--> Event 0 starts.
EventID 0
Run terminated.
Session :
verbose
check_mode
push_notify
test
tracking
event
cuts
run
particle
initialize
beamOn
verbose
printProgress
numberOfThreads
useMaximumLogicalCores
pinAffinity
eventModulo
dumpRegion
dumpCouples
optimizeGeometry
breakAtBeginOfEvent
breakAtEndOfEvent
abort
abortCurrentEvent
geometryModified
reinitializeGeometry
physicsModified
constructScoringWorlds
Choose a command in the command tree