

GPU-Accelerated Optical Simulation Workflow with Symphony

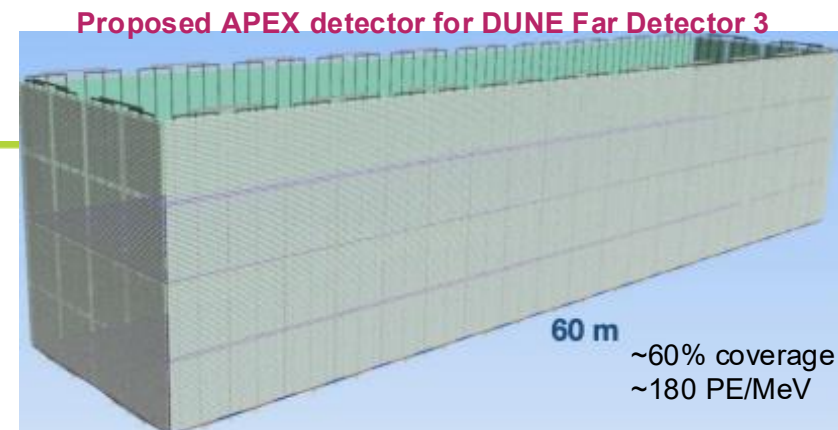
Xuyang Ning (BNL), Gabor Galgoczi (BNL), Clark McGrew (SBU), Brett Viren (BNL) and Chao Zhang (BNL)

CPAD 2026

Motivation

Optical simulation is a bottleneck for future large detector study

- ❑ Next generation detector: kilo tons scintillator, tens of meters size, large photon detection coverage.
- ❑ In liquid argon, a 2.5 GeV electron shower produce about 61 million optical photons
- ❑ optical details matters:
 - geometry boundary conditions
 - accurate physics model
 - wavelength shifting
 - timing



Geant4: validated but slow

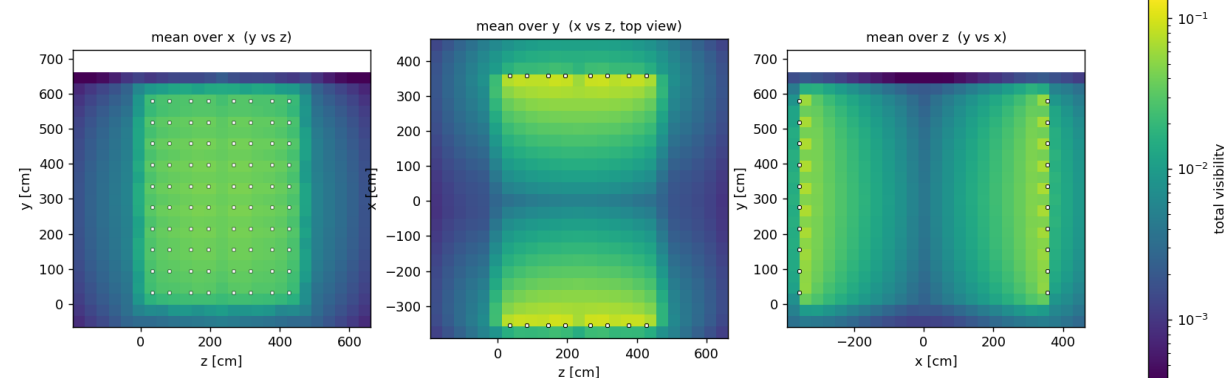
- Widely used and extensively validated
- But Optical photon transport is computationally expensive
- About 800 s per 2.5GeV electron in an event for optical transport in the simplest geometry. Real detector geometries take longer.

Faster approximation ways

- Look-up tables, semi-analytic methods, and machine learning
- Faster but lose details: too approximate for detailed study detector optimization

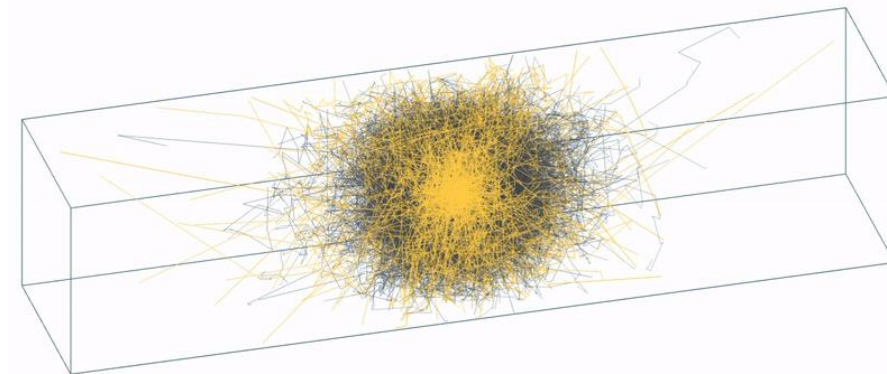
Requirement for this workflow:

- ❑ Practical runtime with full photon simulation
- ❑ Detailed, configurable physics
- ❑ Modular components that can be replaced or extended



An example of the lookup table in ProtoDUNE-HD

General workflow with Symphony



GeGeDe · geometry construction

create shared GDML input for edep-sim and Symphony

.gdml

edep-sim · simulates particle interactions on the CPU

Geant4 physics & geometry

geometry from .gdml

include **Doke-Birks** recombination + LAr quenching

Charge simulation

ionisation dE/dx → TG4HitSegments

Light simulation on CPU

(excitation+recombined energy) → TG4PhotonHit

edep-sim ROOT output (TG4Event)

charge + light hits (CPU or GPU) + truth

edep-simphony-plugin

Plugin loaded at run time. Replaces CPU optical transport.

photon hits → TG4Event

Symphony · transports optical photons on the GPU

Geant4 EM physics on CPU

Symphony's own front-end (also available)

GPU light simulation

OptiX traces photons through scattering, absorption, wavelength shifting, and detection

Symphony output

standalone path, own file (also available)

geometry + gensteps

photon hits

Geometry build: GeGeDe

GeGeDe (General Geometry Description) builds constructive solid geometry for Geant4 and ROOT applications and exports it as GDML. It is implemented in Python. `gegede` · github.com/brettviren/gegede

Why GeGeDe

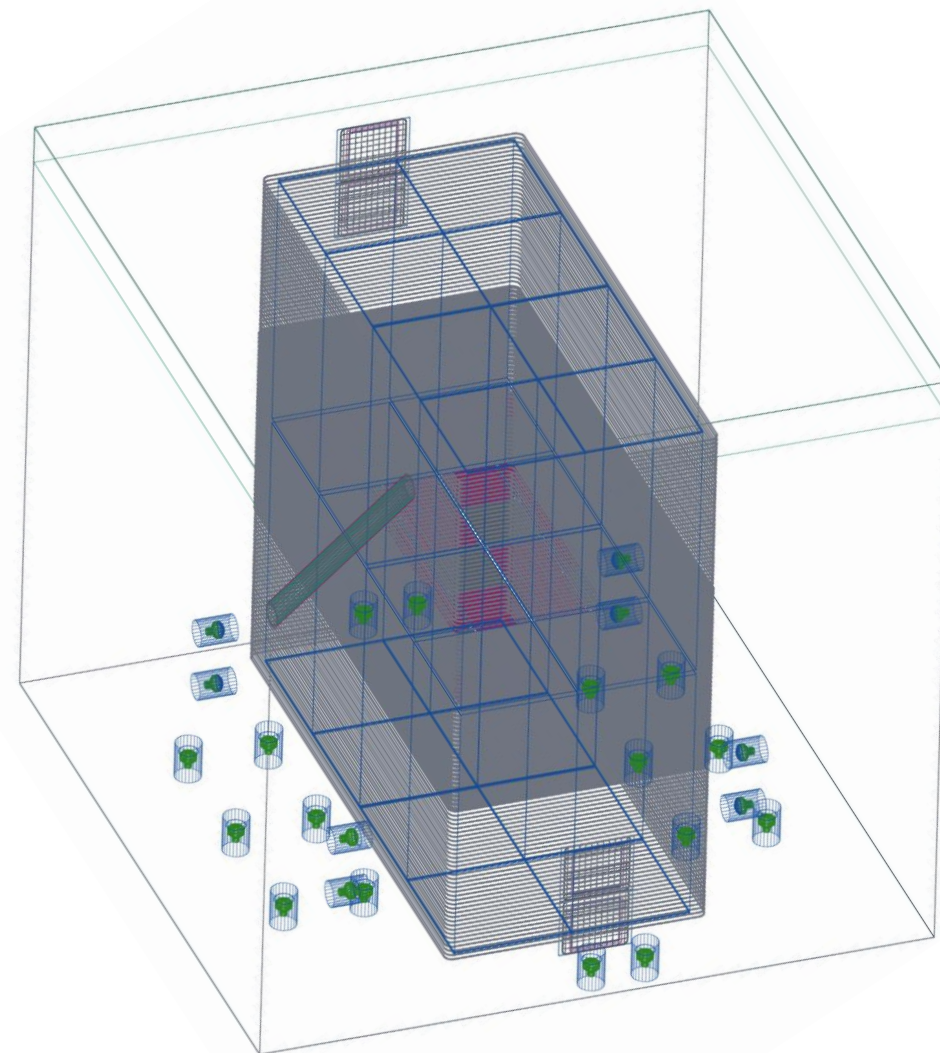
- Geometry written in Python, not hand-written XML
- User friendly layered design: Parameters · builders · object · export
- Independent: pure Python, no Geant4 or ROOT needed
- GDML checked against its schema.

- **Already in use: DUNE and this work**

- DUNE GGD: geometries for the DUNE far detectors
- DUNE ND GGD: DUNE near detectors
- ProtoDUNE-VD: DUNE prototype
- **This work: all our test geometries (LAr box, 14.7 kt benchmark)**
 - liquid-argon scintillation properties: absorption and scattering lengths, the refractive index
 - detector surface properties : reflectivity, absorption, detection efficiency.

- **Update to support optical surface**

- Optical surfaces is added to the schema and the GDML exporter:
 - OpticalSurface, BorderSurface and SkinSurface
- Schema: github.com/brettviren/gegede/.../schema.org



ProtoDUNE-VD written in GeGeDe

edep-sim: Geant4 physics and event output

A ready-to-run Geant4 wrapper for energy-deposition simulation · github.com/ClarkMcGrew/edep-sim

GDML + macro +
generator

edep-sim (full
Geant4)

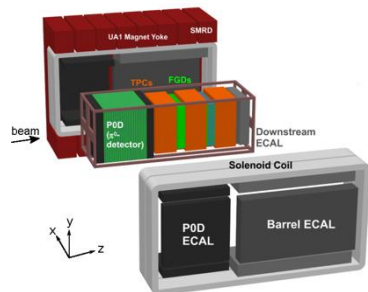
TG4Event (ROOT):
truth, charge, light

Why edep-sim:

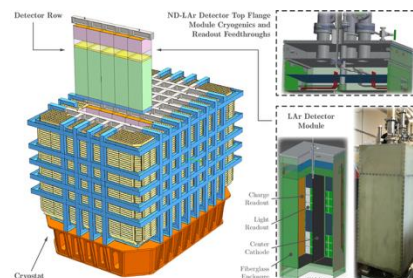
- **Ready to run:** No custom Geant4 main program is required. Geometry and sensitive volumes come from GDML.
- **Stays close to Geant4:** Uses standard Geant4 macros, physics lists, and user actions.
- **Detailed LAr physics:** Doke–Birks recombination model used to make deposited energy into ionization charge and scintillation light.

Well known, and used by multiple experiments :

- T2K ND280: the code started as its detector simulation (2013)
- CAPTAIN and miniCAPTAIN LArTPCs (2014)
- DUNE near detector: ND simulation production, 2x2 prototype



T2K_ND280



DUNE ND LAr

Updates for this work

Complete CPU optical chain

- Optical photon generation & physics:
 - Enable G4OpticalPhysics()
 - Doke–Birks scintillation model for Liquid Argon
- Optical photon transport & detection
 - *HitSurface* is created when photon reaches *SurfaceSD*

Plugin hooks for the GPU

- User actions loaded at run time: no recompiling of edep-sim
 - external GPU engine (Simphony) can take over the photon transport
- GPU photon hits stored like CPU hits, in the same event as charge and truth

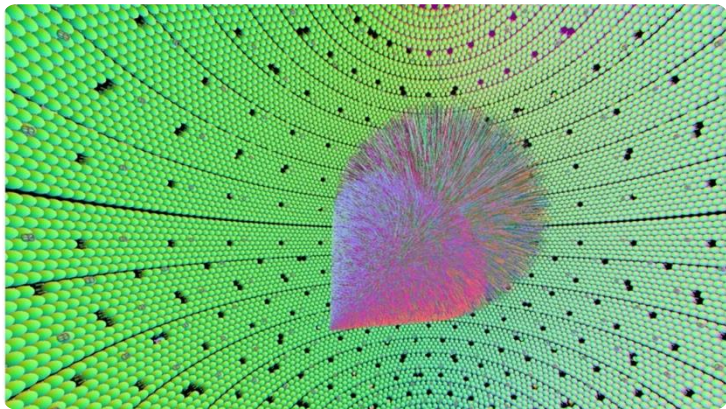
Released in edep-sim **4.4.0** (Sep 2026)

From Opticks to Simphony

Simphony is BNL's fork of Opticks, previously named EIC-Opticks · github.com/BNLNPPS/simphony

Opticks: the foundation

- Developed by Simon Blyth for JUNO
- Uses NVIDIA OptiX ray tracing: $\sim 10^7$ photons per second
- Geometry translated automatically from Geant4 into GPU primitives (box, sphere, torus, ...)

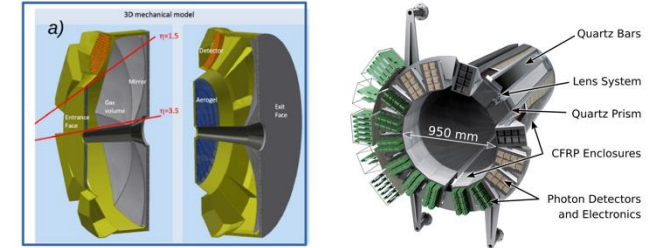


JUNO: $\sim 10^7$ photons from a 200 GeV muon, simulated with Opticks

Used at the EIC (ePIC), validated against Geant4

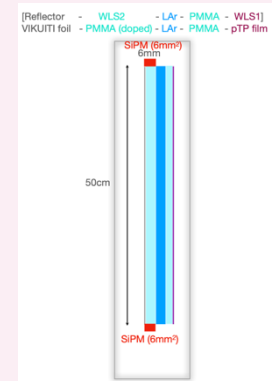
Cherenkov detectors: dRICH, hpDIRC, pfRICH

Pictured: dRICH (left), hpDIRC (right)



Simphony additions:

- Wavelength shifting on the GPU
- Monte Carlo truth propagation
- Asynchronous CPU/GPU launches
- Shader Execution Reordering: speeding up Opticks $\sim 10x$ for very divergent workloads
- New volume types: G4Trd, G4Trap, cylindrical and spherical wedges



X-Arapuca: a WLS design

Symphony architecture

1 · Geant4 on CPU

Simulates electromagnetic and hadronic interactions. Cherenkov and scintillation processes record **gensteps** instead of creating photons.

2 · Gensteps

Compact records describe photon production, including position, direction, polarization, photon count, and event ID.

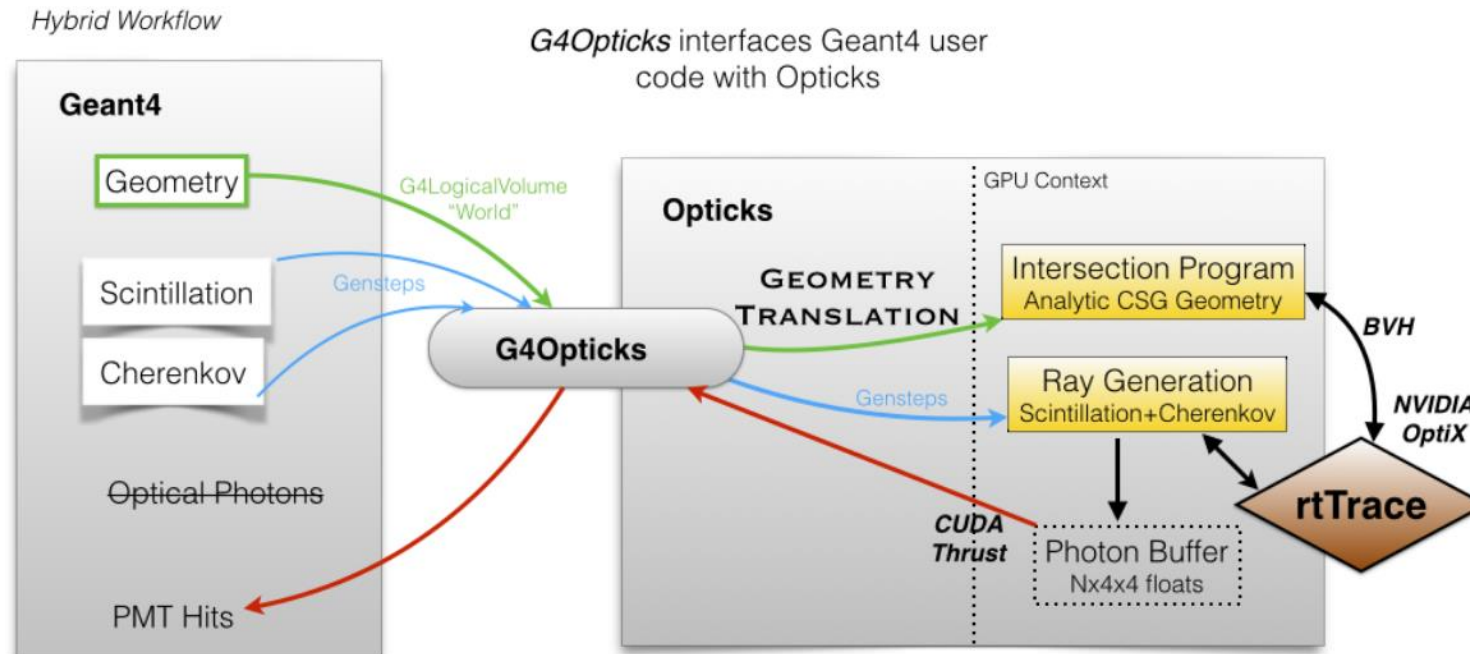
3 · GPU transport

Symphony generates and propagates the photons with NVIDIA OptiX

4 · Return detected Hits

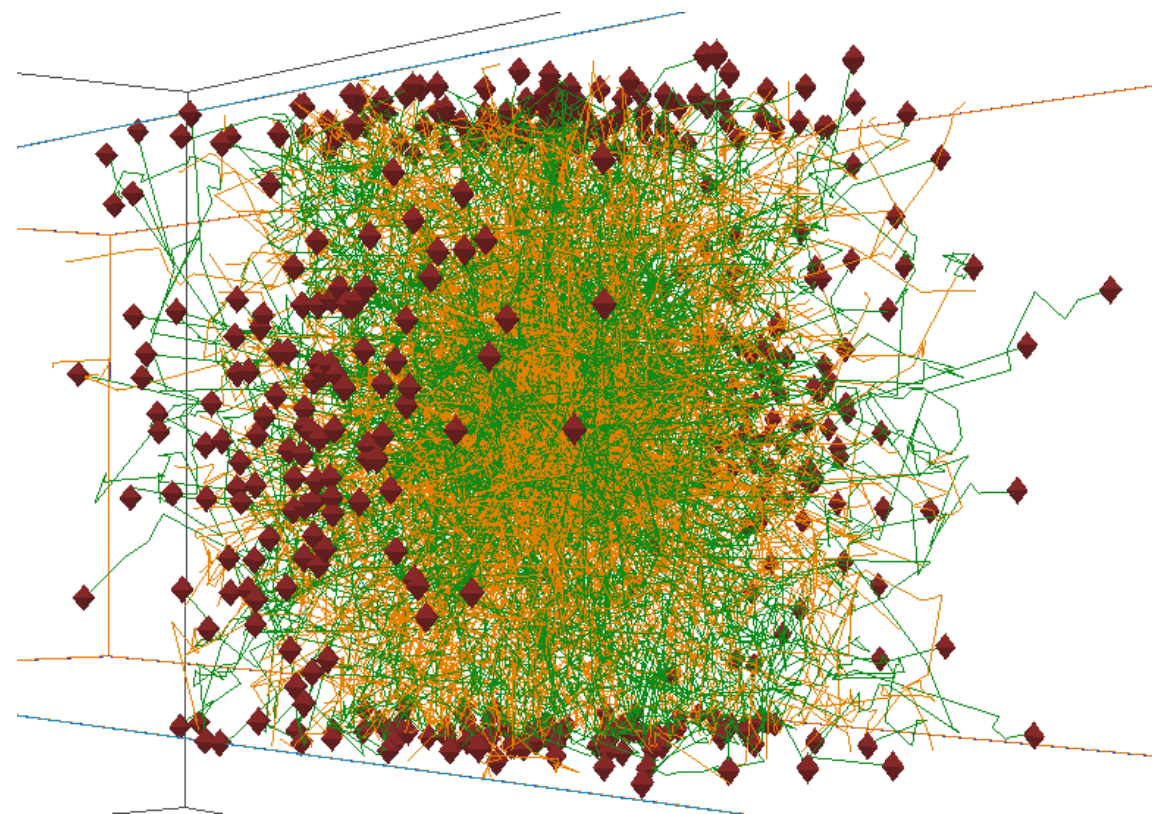
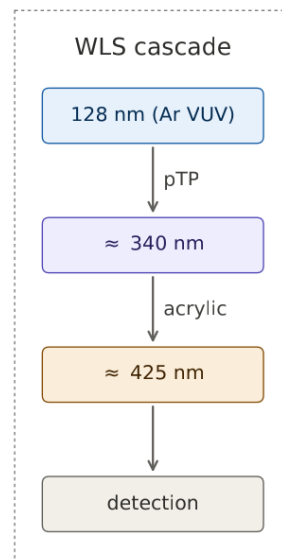
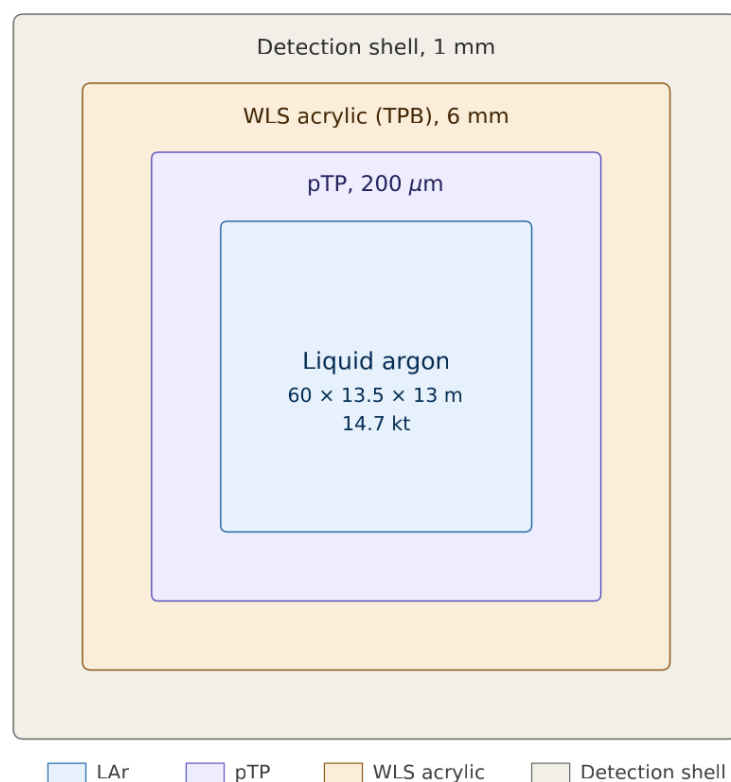
Only detected photons return to Geant4 through G4Opticks.

Geometry and optical properties are translated from Geant4 to the GPU; same physics, preserving the detector description, while accelerating optical transport.



Symphony for noble-liquid detectors

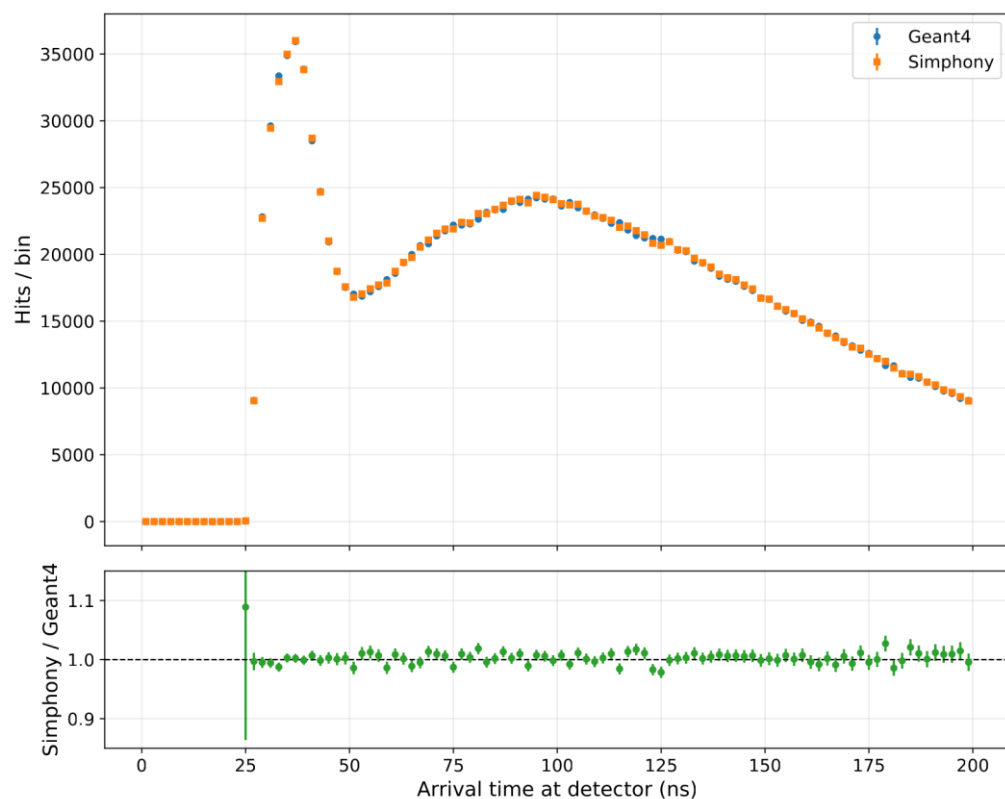
- G. Galgoczi et al., “GPU optical photon Monte Carlo for noble liquid detectors: validation against Geant4 in a large liquid argon TPC benchmark”, [arXiv:2606.05385](https://arxiv.org/abs/2606.05385)
- **Implemented the wavelength-shifting process on the GPU**
- We implemented wavelength shifting on the GPU and defined a 14.7 kt liquid-argon benchmark to validate the physics and measure performance.



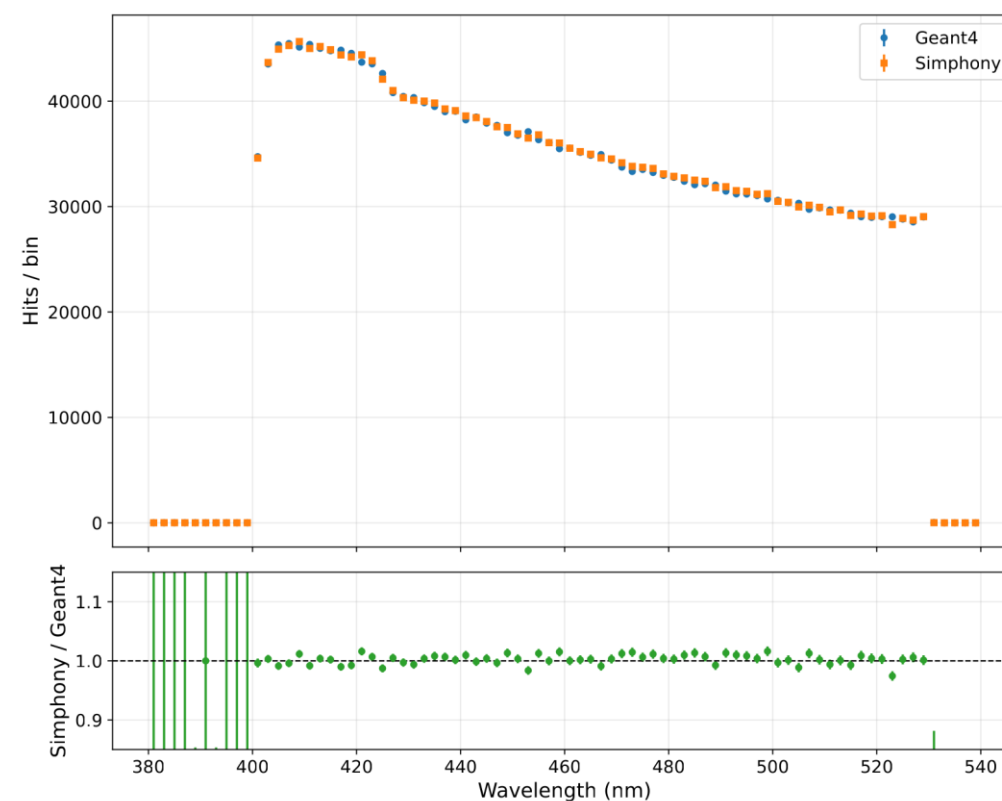
Photon tracks from a Simphony GPU simulation of a 0.5 MeV electron in the benchmark geometry
green: detected directly;
yellow: enter the wavelength shifter, escape back into the liquid argon, re-enter the shifting layer, and are finally detected

Validation against Geant4

- CPU (Geant4 11.3.2) and GPU runs use identical gensteps, isolating optical transport. The tests run inside Simphony without edep-sim.
- Detected-photon ratio, Simphony/Geant4:
 - 2.5 GeV e^- (61 M photons/event) 1.0022 ± 0.0002 · 1 GeV μ^- 1.0017 ± 0.0008 · 400 MeV p 1.0005 ± 0.0014
 - All three **samples agree within 0.25%**
- For 2.5 GeV e^- : arrival-time spectrum $\chi^2/\text{ndf} = 0.98$; wavelength spectrum $\chi^2/\text{ndf} = 1.08$



Arrival time at the detector

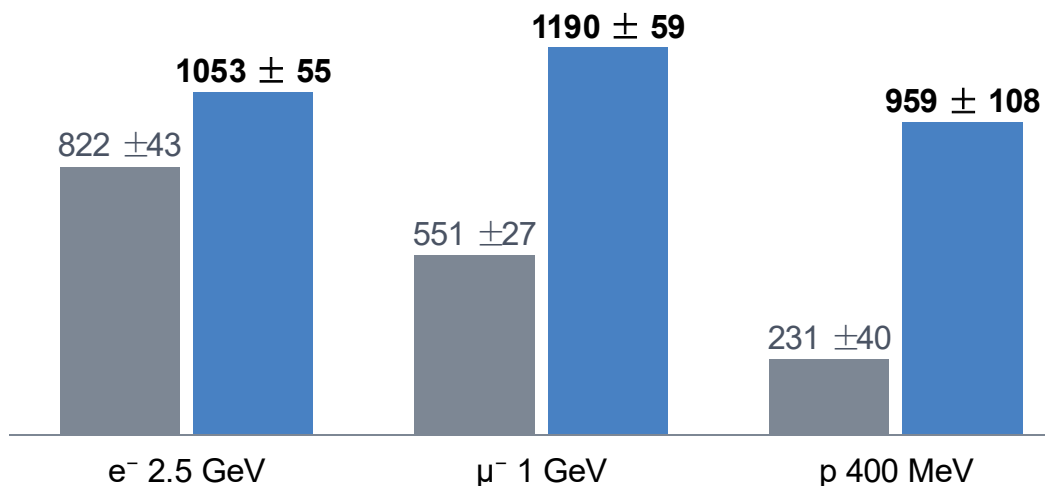


Wavelength at the detector: **wavelength shifting agrees**

Optical transport speedup

Optical transport speedup relative to one CPU thread

■ 1 event / launch ■ stacked: 4 / 7 / 23 events per launch



What makes Symphony fast

1

Hardware ray tracing

OptiX BVH traversal on RTX ray-tracing cores; Geant4 geometry → GPU CSG primitives

2

Gensteps instead of photons

~67 000 compact records per 2.5 GeV e^- ; the 61 M photons are created on the GPU (upload < 2 ms)

3

Event stacking

several events in one GPU launch share the launch overhead (grey → blue bars)

4

Shader Execution Reordering (SER)

Reorders GPU work so long photon paths do not delay shorter ones : up to $\sim 10\times$ faster propagation kernel, without changing the simulated hits
[arxiv: 2608.21396](https://arxiv.org/abs/2608.21396)

5

Asynchronous CPU/GPU launches

Geant4 continues on the CPU while the GPU propagates the previous batch of gensteps

- Hardware: NVIDIA RTX 4090 vs one Intel Xeon w7-3445 thread
- When simulating 4, 2.5GeV electron in one GPU call:
 - Throughput: 243 M photons in 3.03 s → **80.2 M photons/s**
- **End-to-end speedup, including the Geant4 electromagnetic shower and optical transport: $(89 \pm 5)\times$. The CPU portion now accounts for about 90% of wall time.**

* These results were obtained fully within Symphony (standalone Geant4 + Symphony); the edep-sim + plugin route is not yet included.

edep-simphony-plugin

A thin shared library sends gensteps from edep-sim to Simphony and returns detected photon hits. · github.com/Ningclover/edep-simphony-plugin

1 · CPU · edep-sim

Collect gensteps

- Scintillation and Cerenkov steps record gensteps instead of photons
- A genstep → trackID maps each genstep to its parent Geant4 track



2 · GPU · Simphony

Transport photons on GPU

- At the end of each event, OptiX traces all photons in parallel
- Scattering, absorption and wavelength shifting until detected or lost



3 · CPU · edep-sim

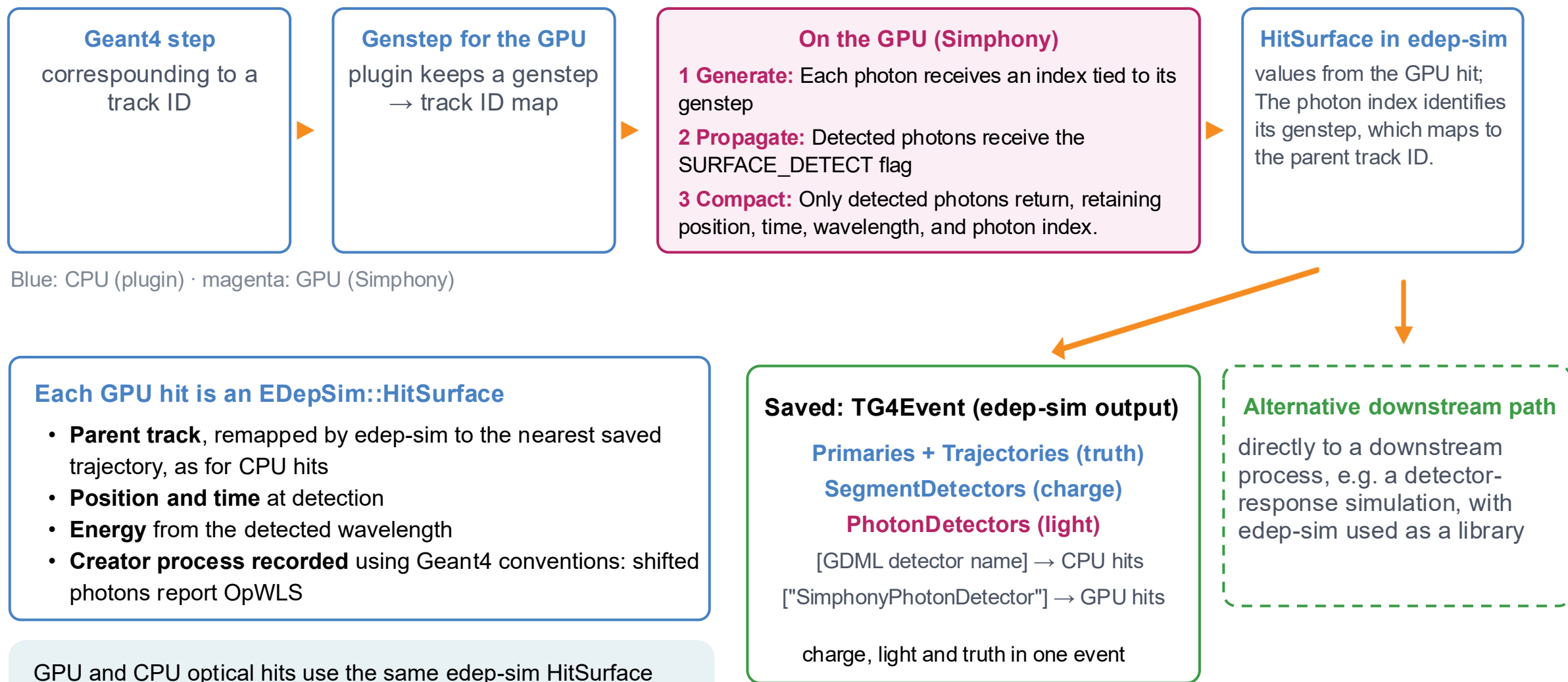
Store hits and restore truth links

- Each hit is linked to the track that produced it
- Saved in edep-sim's own event, same format as CPU hits: charge + light in one file

- **Runtime plugin:** loaded by edep-sim at run time, no recompiling of edep-sim or Simphony
- **One geometry:** the same GDML geometry and optical properties on CPU and GPU
- **Operating modes:** GPU, CPU, or dual mode for event-by-event validation

Truth flow: from GPU hit back to the particle

Every detected GPU photon becomes an edep-sim **HitSurface** linked to the Geant4 track that generated it.



Blue: CPU (plugin) · magenta: GPU (Simphony)

GPU and CPU optical hits use the same edep-sim HitSurface class.: downstream code reads GPU light like any edep-sim hit. Merged into edep-sim, released in 4.4.0

geo-vis: interactive 3D event viewer

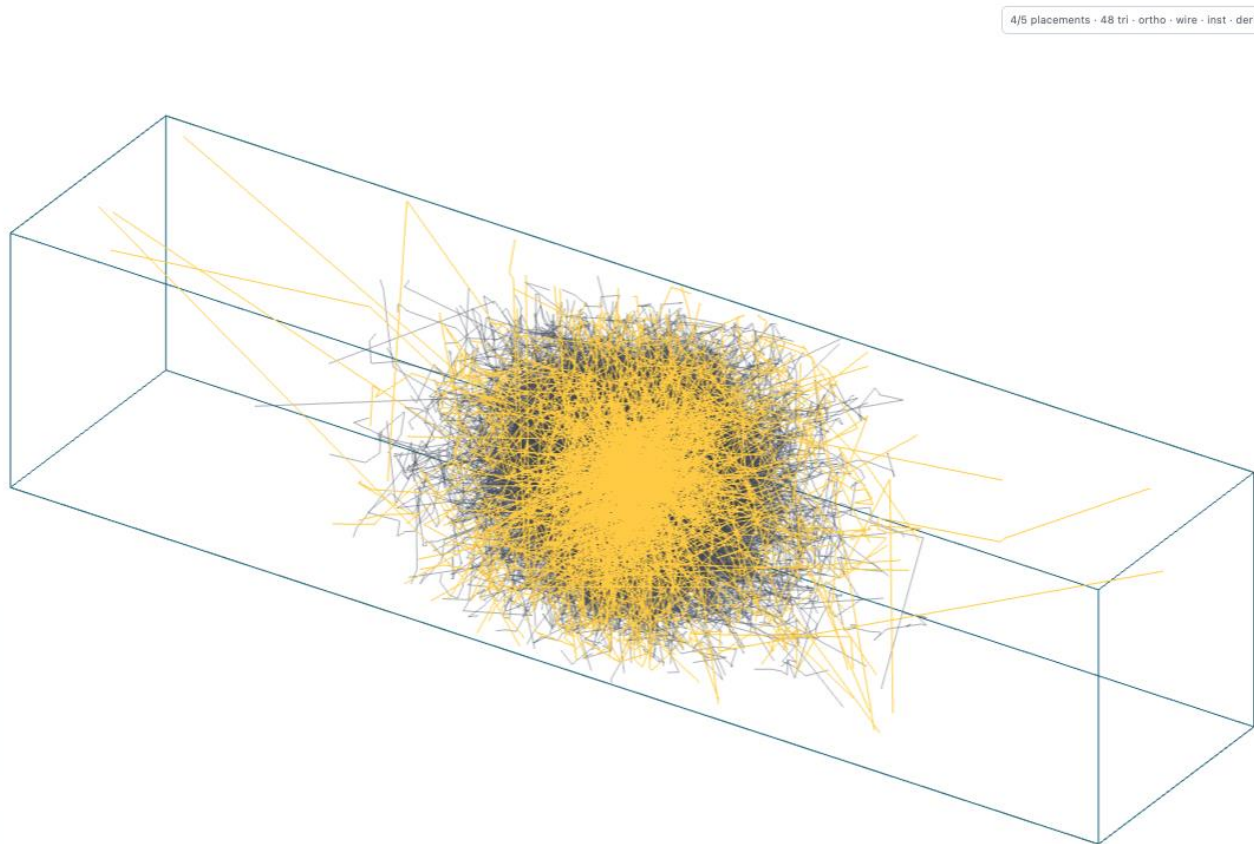
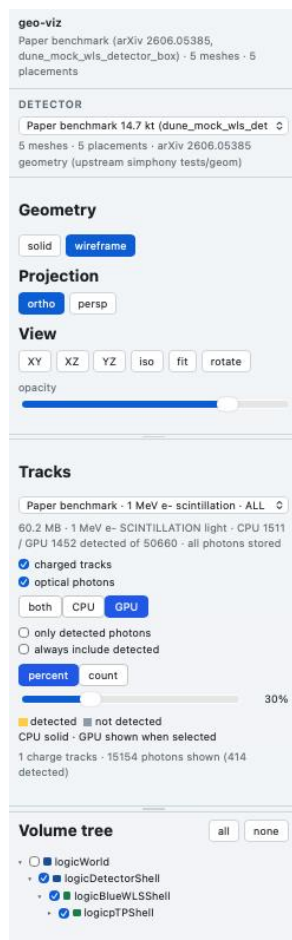
View detector geometry, energy deposits, and CPU/GPU optical tracks in one browser-based display · github.com/Ningclover/geo-vis

What it does

- **Any GDML geometry in true 3D:** Renders GDML geometry using tessellated solids and rotations rather than bounding boxes
- **Volume tree:** Provides per-volume visibility controls
- Uses instancing to keep files compact
- Overlays charge deposits and CPU/GPU photon tracks
- Exports a self-contained HTML file that runs in any browser

Example on the right

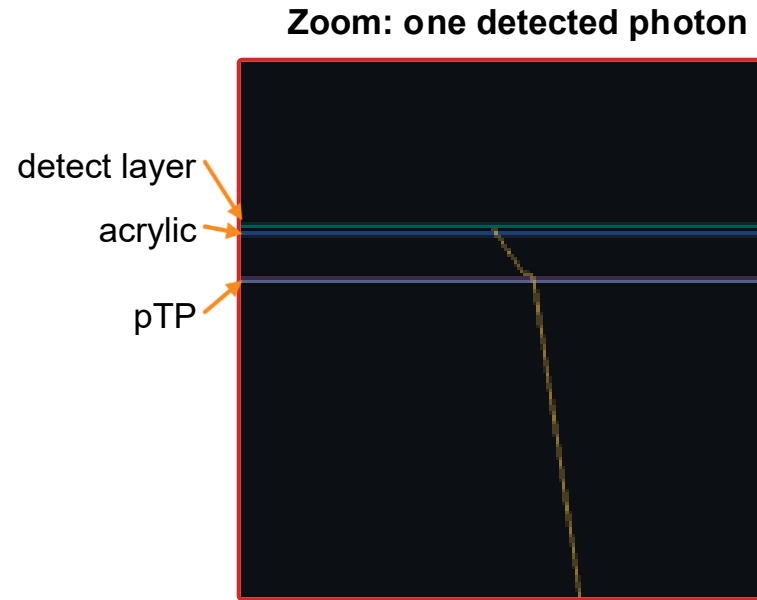
1 MeV e^- in the 14.7 kt benchmark: **CPU 1511** / **GPU 1452** detected photons



GPU photon tracks shown (29 % sample): yellow = detected, grey = not detected

Event display example

1 MeV electron (E-field 500 V/cm) in the 14.7 kt benchmark geometry, viewed in geo-vis



3 detected GPU photon tracks



The track crosses the pTP and acrylic layers, undergoes both wavelength-shifting stages, and ends in the detection layer.

Summary and outlook

We provide a complete workflow for GPU optical simulation

- **GeGeDe**: geometry construction;
 - optical surfaces are added to its schema and GDML exporter.
- **edep-sim**: Simulates particle interactions with Geant4;
 - added the full CPU optical chain and plugin hooks for GPU optics
- **Simphony**: (fork of Opticks) Transports photons using NVIDIA ray tracing;
 - added GPU wavelength shifting, MC truth, asynchronous launches and SER.
 - 14.7 kt benchmark detector:
 - Detected-photon yields agree with Geant4 within 0.25%.
 - Optical transport is about 1000× faster than one CPU thread in ideal case.
- **Plugin**: connects everything:
 - Stores GPU photon hits and their parent-track links in standard edep-sim output. ready for any downstream use
- **geo-viz**: interactive 3D viewer of the geometry, charge and CPU/GPU light tracks of an event, in any browser

Outlook

- working in a realistic detector geometries
- Asynchronous launches in the plugin.
- GPU electromagnetic showers (AdePT + Simphony) → the whole simulation on the GPU
- Extend the workflow to other scintillation detectors

Backup

Property	Value or range
Liquid argon (G4_1Ar)	
RINDEX	1.23 → 1.6 over 2.34–11.7 eV, rising
GROUPVEL	151–244 mm/ns, dispersion corrected
RAYLEIGH	≈ 0.99 m at 128 nm (Babicz <i>et al.</i> 2020 [27]); 21-point $\lambda^4/(n^2 - 1)^2$ dispersion across the LAr active band
ABSLNGTH	80 m for $E \leq 7.5$ eV, 10 m for $E \geq 8.0$ eV
SCINTILLATIONCOMPONENT	truncated Gaussian, peak 128 nm, FWHM 10 nm, window 113–155 nm (Heindl <i>et al.</i> 2010 [28]); used for COMPONENT1/COMPONENT2/FAST/SLOW
SCINTILLATIONYIELD	24 000 photons/MeV
Time constants, fast/slow	7 ns / 1400 ns, with yields 0.75 / 0.25
Bulk parameters	$T = 87$ K, $\rho = 1.396$ g/cm ³ , $I_{\text{exc}} = 188$ eV
Para-terphenyl (pTP, inner WLS)	
RINDEX	1.65 flat
GROUPVEL	181.7 mm/ns flat
WLSCOMPONENT	Emission peak at 340 nm, with vibronic shoulders at 400–425 nm
WLSABSLNGTH	0.5 μ m at 128 nm, rising to ≥ 10 m above 340 nm
ABSLNGTH	Shared ABSLENGTH_acrylic table: 1 μ m in the VUV → 10 m in the visible
WLSTIMECONSTANT	1.136 ns
TPB-doped acrylic (bluewlsacrylic, outer WLS)	
RINDEX	1.58 flat, EJ-286 PVT base
GROUPVEL	189.7 mm/ns flat
WLSCOMPONENT	Emission peak at 425 nm, with weak 530 nm tail
WLSABSLNGTH	0.1 μ m in the VUV; 0.8 mm at 340–400 nm peak; ≥ 2 m above 400 nm
ABSLNGTH	Shared ABSLENGTH_acrylic, see pTP
WLSTIMECONSTANT	1.26 ns
SiPM surface	
EFFICIENCY	1.0 flat, idealised photon counter; UNIFIED dielectric–metal, polished finish