GPU-Accelerated Optical Simulation Workflow with Simphony

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

CPAD 2026

Brookhaven National Laboratory

Motivation

Optical simulation is a bottleneck for large detectors

- In liquid argon, a 2.5 GeV electron shower produces about 61 million optical photons.
- Geant4 takes about 800 s per event for optical transport in the simplest geometry. Real detector geometries take longer.

optical detail matters

- accurate physics models
- wavelength shifting
- timing
- correct geometry boundary conditions

Geant4: validated but slow

Widely used and extensively validated
But Optical photon transport is computationally expensive

Faster approximations

Look-up tables, semi-analytic methods, and machine learning

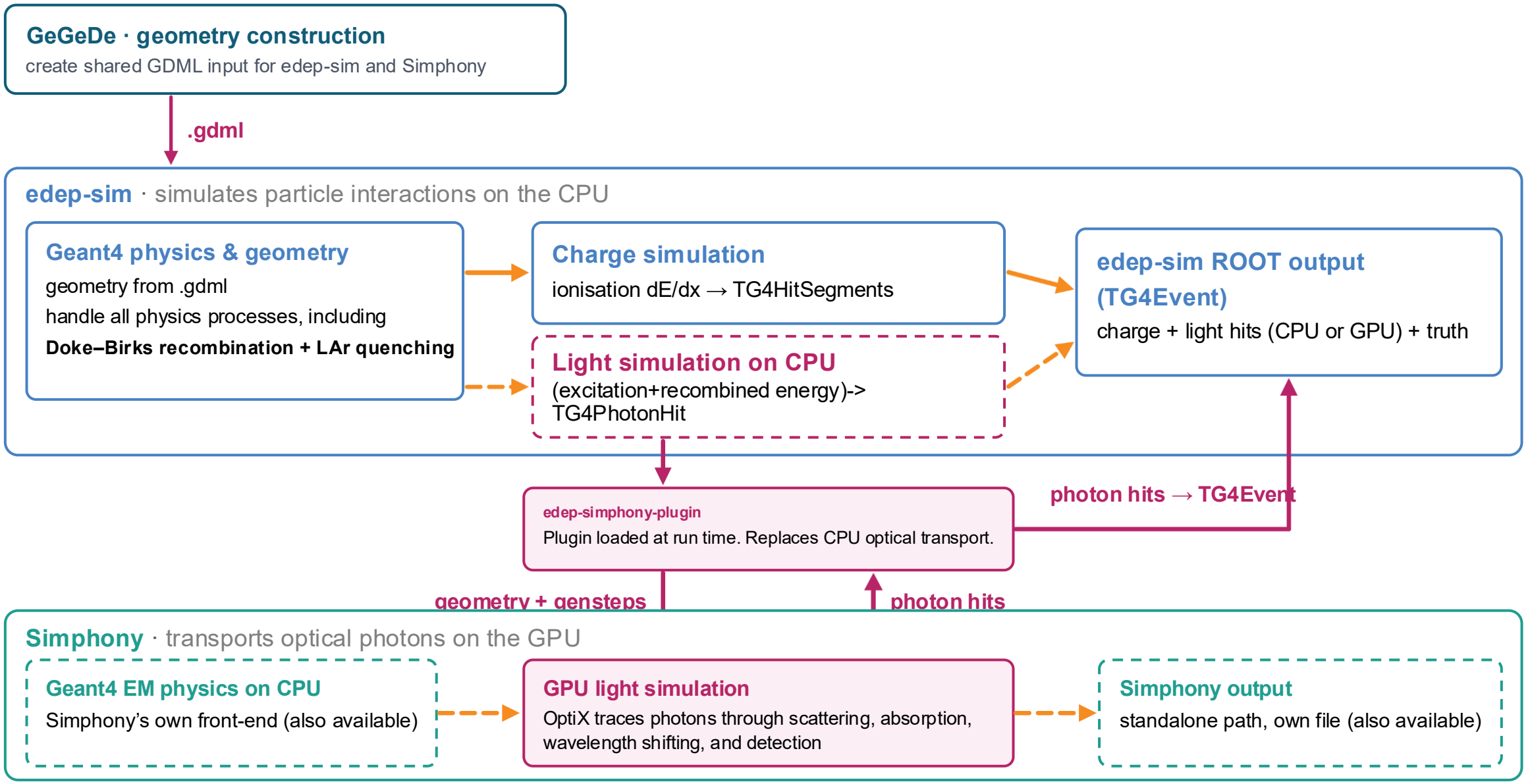
Faster but lose details: often too approximate for detailed study detector optimization

Requirement for this workflow:

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

Numbers: Galgoczi et al., arXiv:2606.05385 (Geant4 11.3.2, Intel Xeon w7-3445)

General workflow



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

layered design; lets users work at any level

parameters
high-level, human-readable configuration

▶

builders
procedural code that assembles the geometry

▶

objects
in-memory representation of the final geometry

▶

exports
exports the geometry to GDML and other formats

Already in use: DUNE and this work

- DUNE GGD: geometries for the DUNE far detectors
- DUNE ND GGD: geometries for (some) DUNE near detectors
- This work: all our test geometries (LAr box, 14.7 kt benchmark)

Update to support optical simulation

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

Why GeGeDe

- Geometry written in Python, not hand-written XML
- Reusable builders, selectable through configuration
- One command writes the GDML
- Independent: pure Python, no Geant4 or ROOT needed
- GDML checked against its schema;
- JSON, ROOT, USD can also be exported

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:** NEST-based 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

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
  - ➢ externel GPU engine (Simphony) can takes over the photon transport
- GPU photon hits stored like CPU hits, in the same event as charge and truth

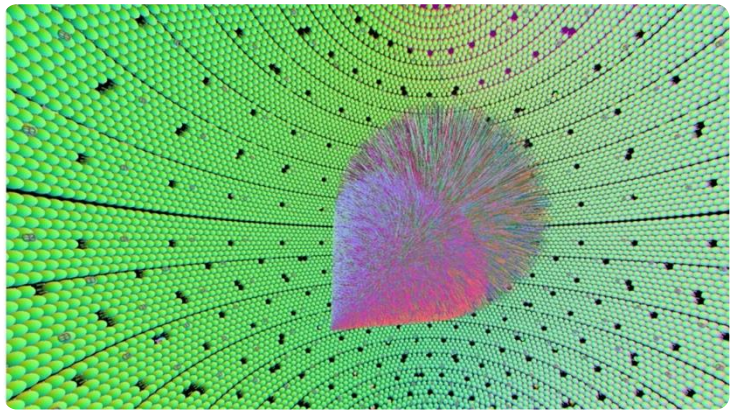
Released in edep-sim **4.4.0** (Sep 2026)

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: ~10^7 photons per second
- Geometry translated automatically from Geant4 into GPU primitives (box, sphere, torus, …)

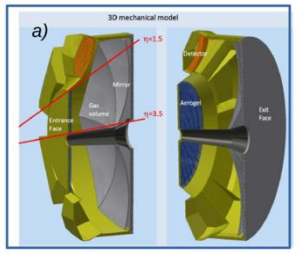


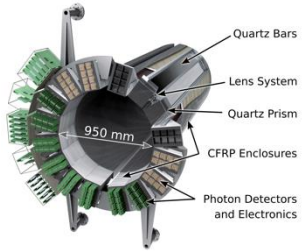
JUNO: ~10^7 photons from a 200 GeV muon, simulated with Opticks

Used at the EIC (ePIC), validated against Geant4

Cherenkov detectors: dRICH, hpDIRC, pfRICH

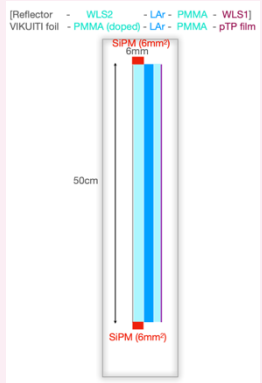
Pictured: dRICH (left), hpDIRC (right)





Simphony additions:

- Wavelength shifting on the GPU
- Monte Carlo truth propagation
- Asynchronous CPU/GPU launches
- New volume types: G4Trd, G4Trap, cylindrical and spherical wedges
- Speed-ups



X-Arapuca: a WLS design

Simphony 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

Simphony 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.

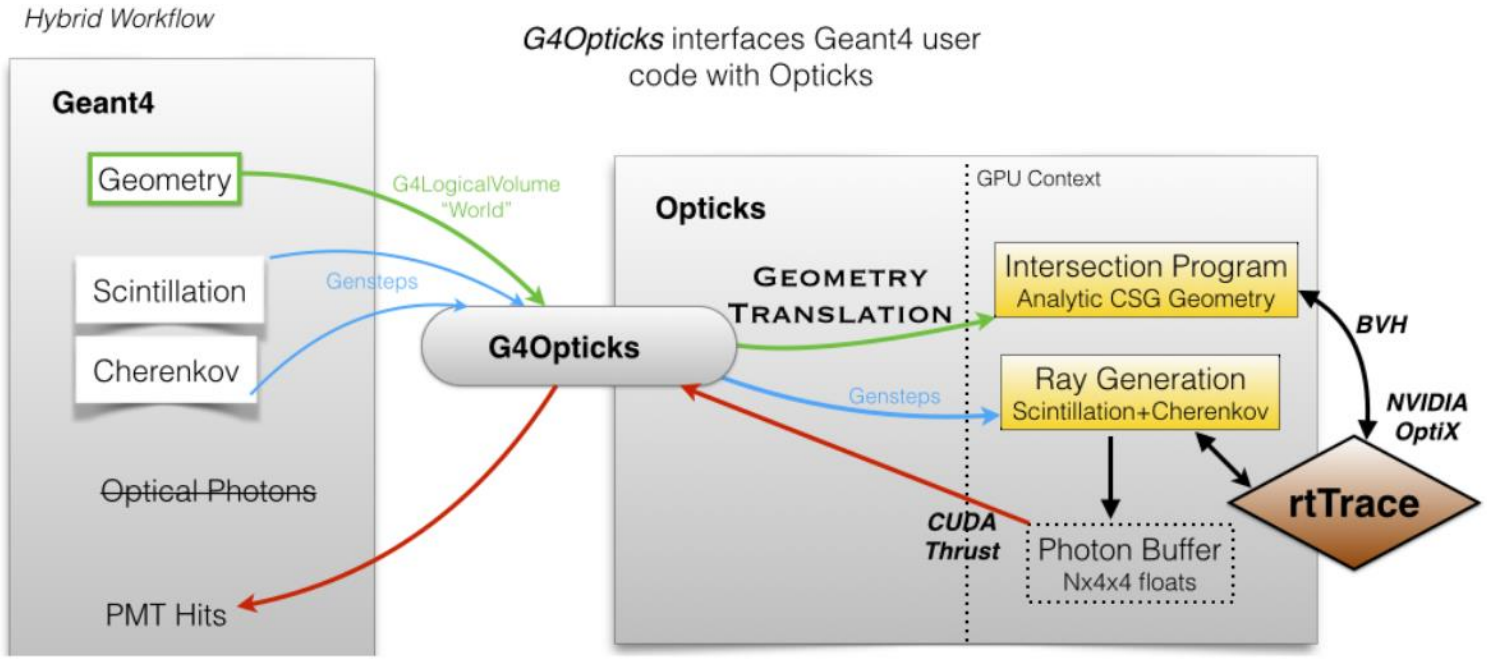
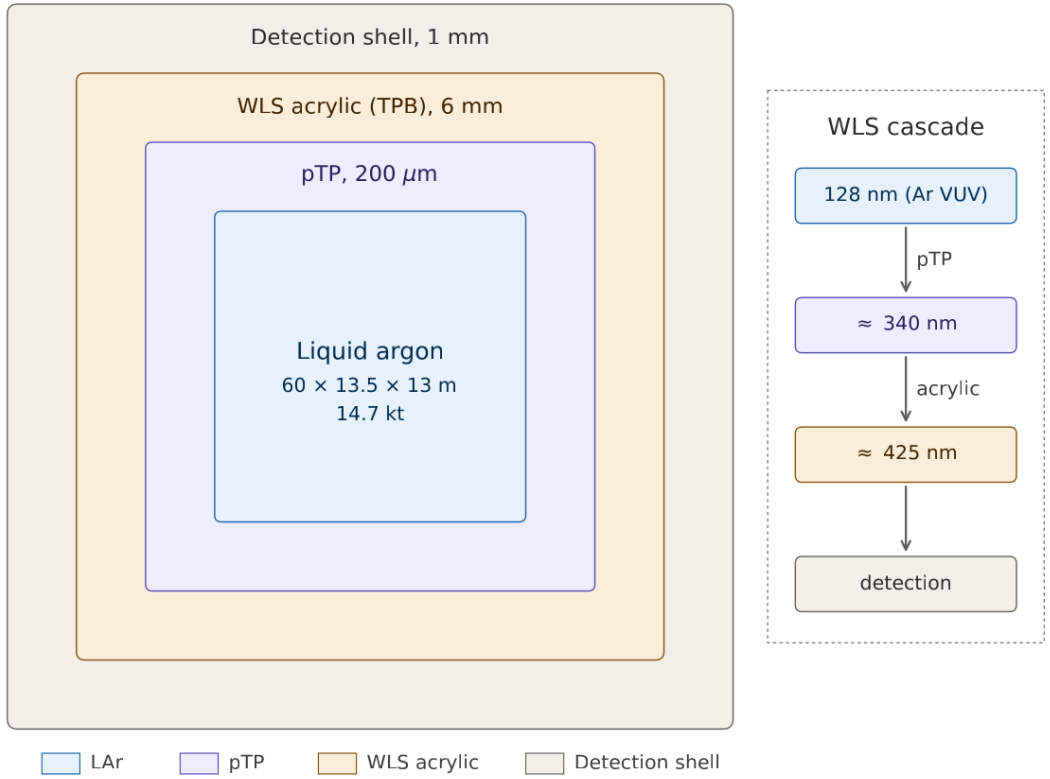
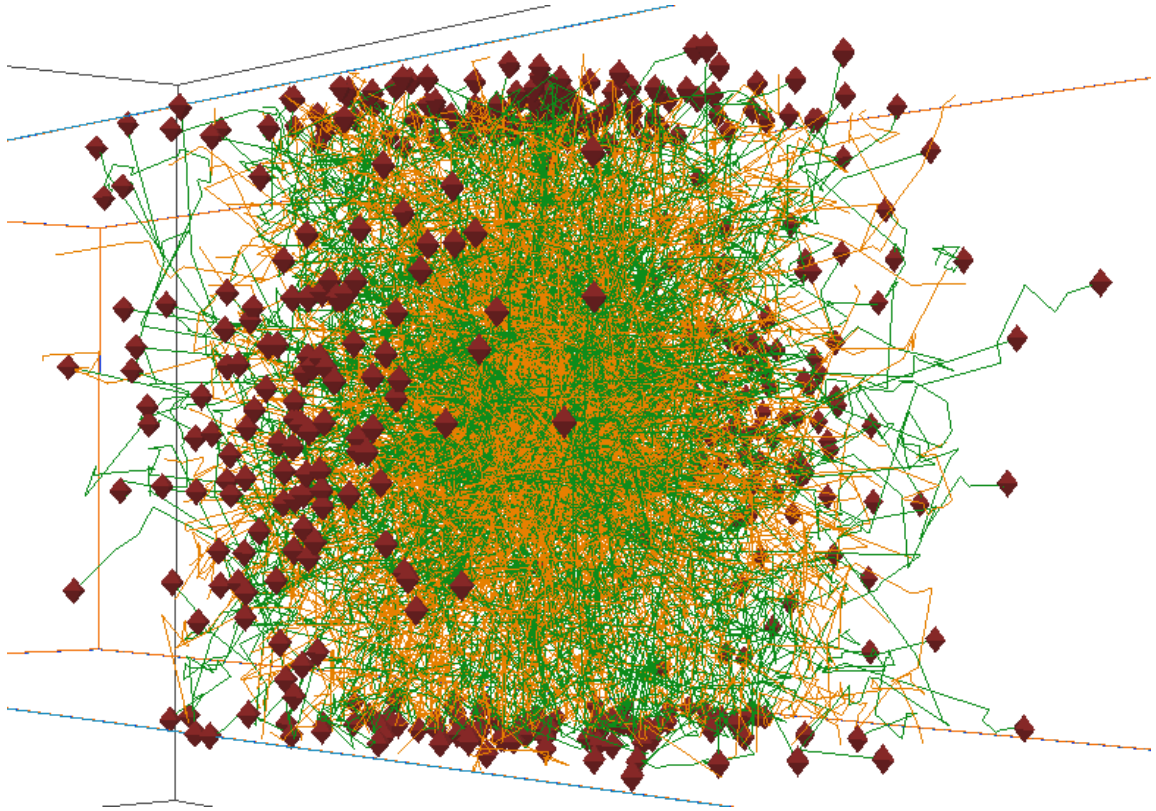


Diagram: S. Blyth, EPJ Web of Conferences 245, 11003 (2020)

Simphony 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
- **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.

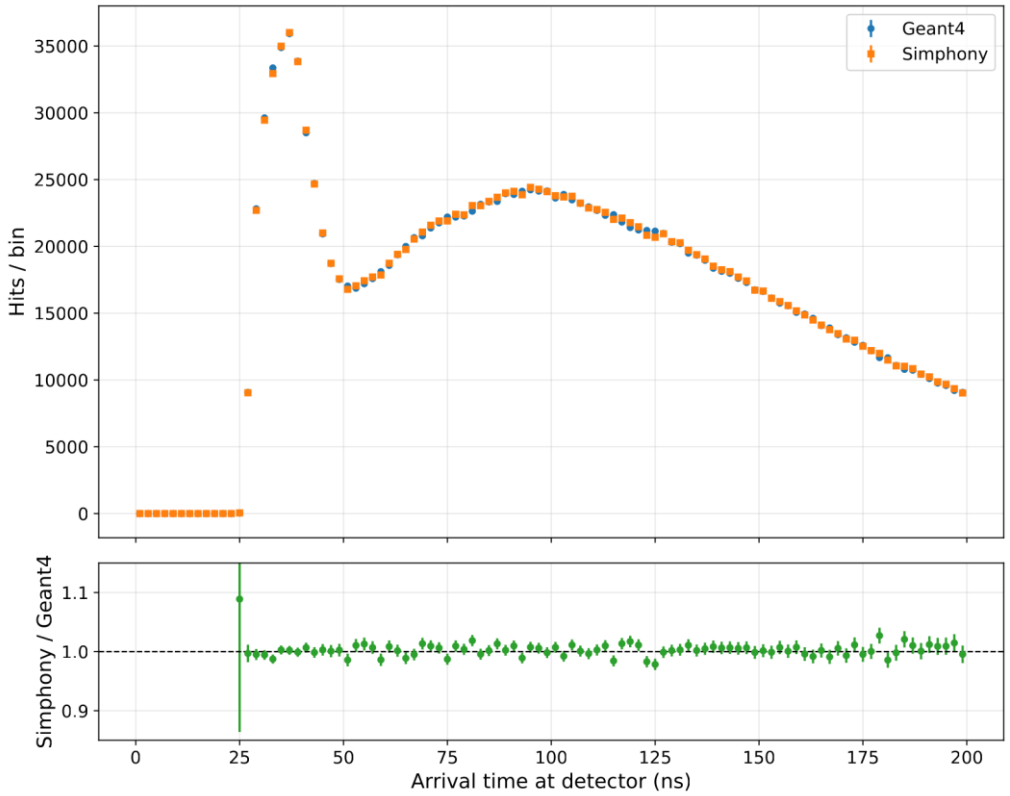




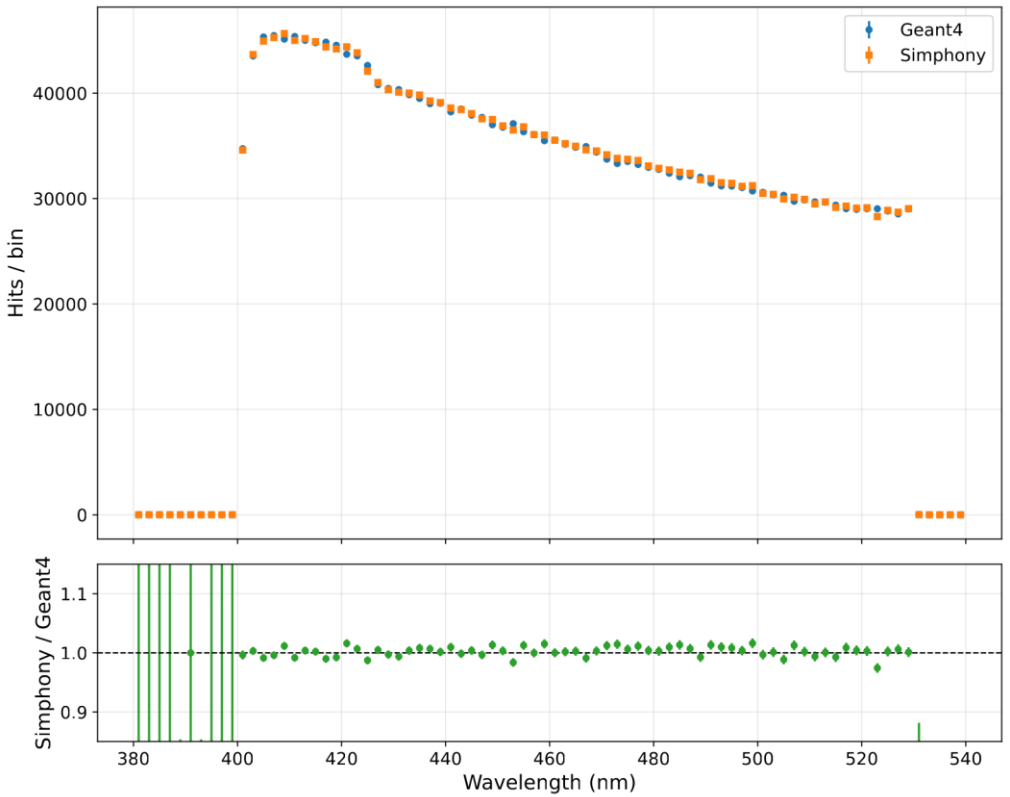
Photon tracks from a Simphony GPU simulation of a 0.5 MeV electron in the benchmark geometry

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 χ^2/ndf = 0.98; wavelength spectrum χ^2/ndf = 1.08

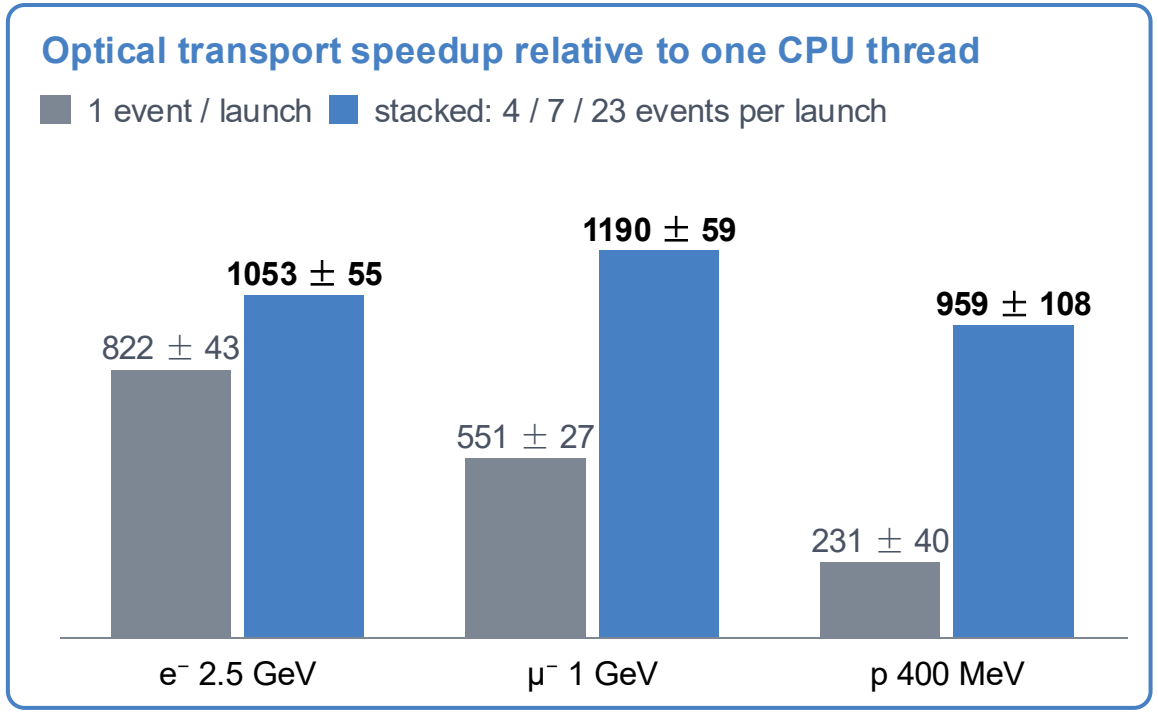


Arrival time at the detector



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

Optical transport speedup



What makes Simphony 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 ~10× faster propagation kernel, without changing the simulated hits
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 ± 5)×. The CPU portion now accounts for about 90% of wall time.**

* These results were obtained fully within Simphony (standalone Geant4 + Simphony); 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.

Geant4 step
correspounding to a track ID

Genstep for the GPU
plugin keeps a genstep → track ID map

On the GPU (Simphony)
1 Generate: Each photon receives an index tied to its genstep
2 Propagate: Detected photons receive the SURFACE_DETECT flag
3 Compact: Only detected photons return, retaining position, time, wavelength, and photon index.

HitSurface in edep-sim
values from the GPU hit; The photon index identifies its genstep, which maps to the parent track ID.

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

Each GPU hit is an EDepSim::HitSurface

- **Parent track**, remapped by edep-sim to the nearest saved trajectory, as for CPU hits
- **Position and time** at detection
- **Energy** from the detected wavelength
- **Creator process recorded** using Geant4 conventions: shifted photons report OpWLS

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

Saved: TG4Event (edep-sim output)

Primaries + Trajectories (truth)
SegmentDetectors (charge)
PhotonDetectors (light)
[GDML detector name] → CPU hits
["SimphonyPhotonDetector"] → GPU hits

charge, light and truth in one event

Alternative downstream path
directly to a downstream process, e.g. a detector-response simulation, with edep-sim used as a library

geo-vis: interactive 3D event viewer

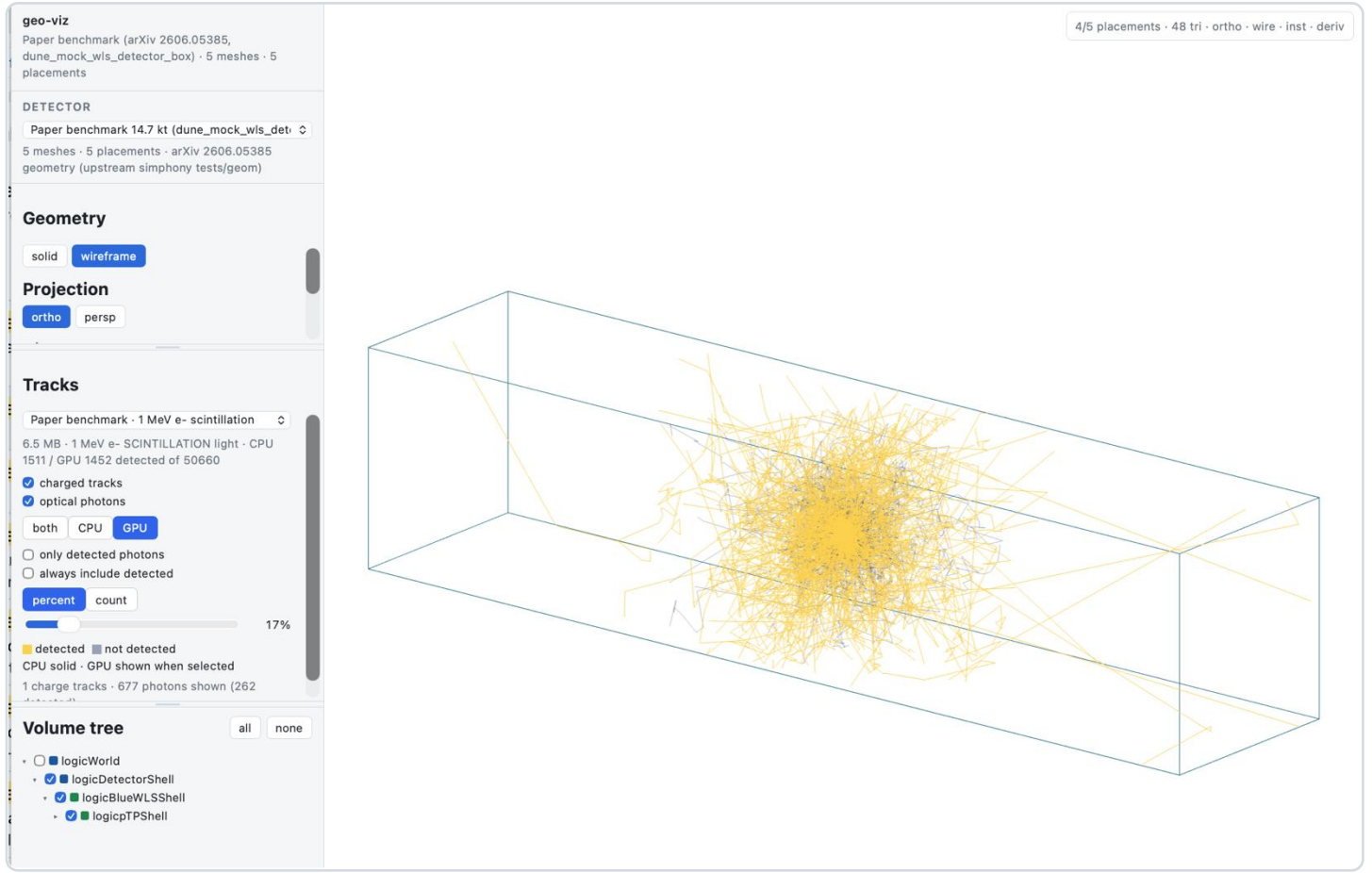
View detector geometry, energy deposits, and CPU/GPU optical tracks in one browser-based display· github.com/Ningclover/geo-vis

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

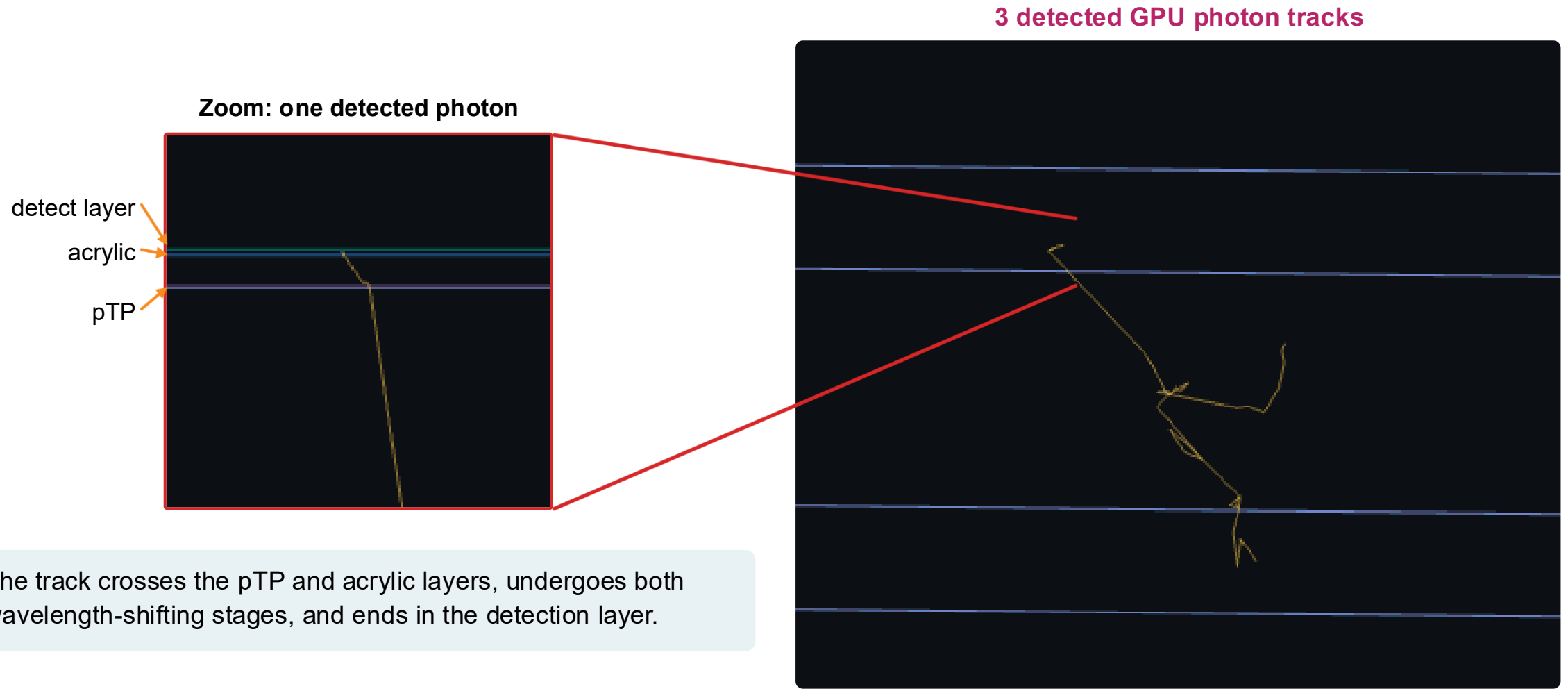
1 MeV e^- in the 14.7 kt benchmark: **CPU 1511 / GPU 1452** detected photons



GPU photon tracks shown (17 % 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-vi



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.
- **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 and the latest Simphony speed-ups in the plugin
- GPU electromagnetic showers (AdePT + Simphony) → the whole simulation on the GPU
- Extend the workflow to other scintillation detectors