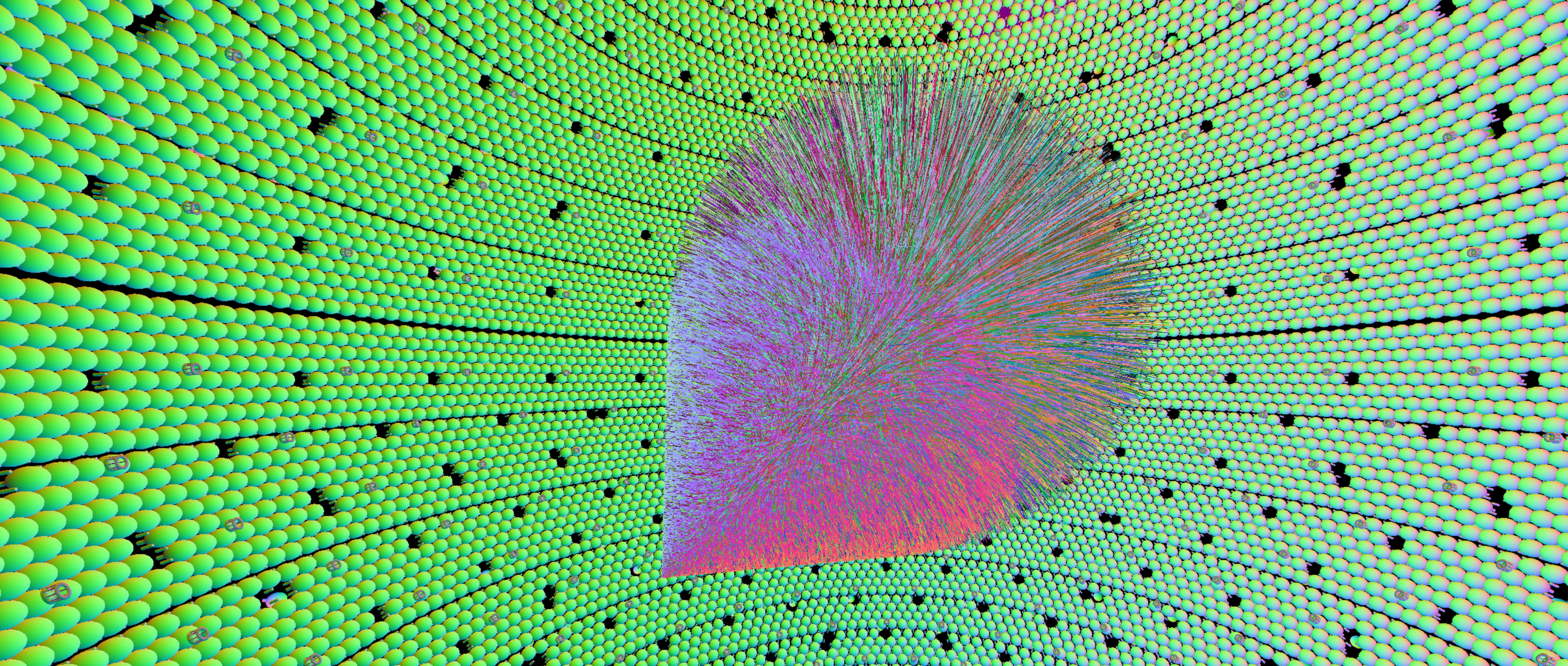


Opticks : GPU Optical Photon Simulation for Particle Physics with NVIDIA® OptiX™



Simon C Blyth, National Taiwan University – <https://bitbucket.org/simoncblyth/opticks> – Oct 2016, CHEP

Opticks > 1000x Geant4 (*)

GPU massive parallelism **eliminates bottleneck**.

- optical photon simulation time --> zero
- optical photon CPU memory --> zero

[zero: effectively, compared to rest of simulation]

More Photons -> More Benefit

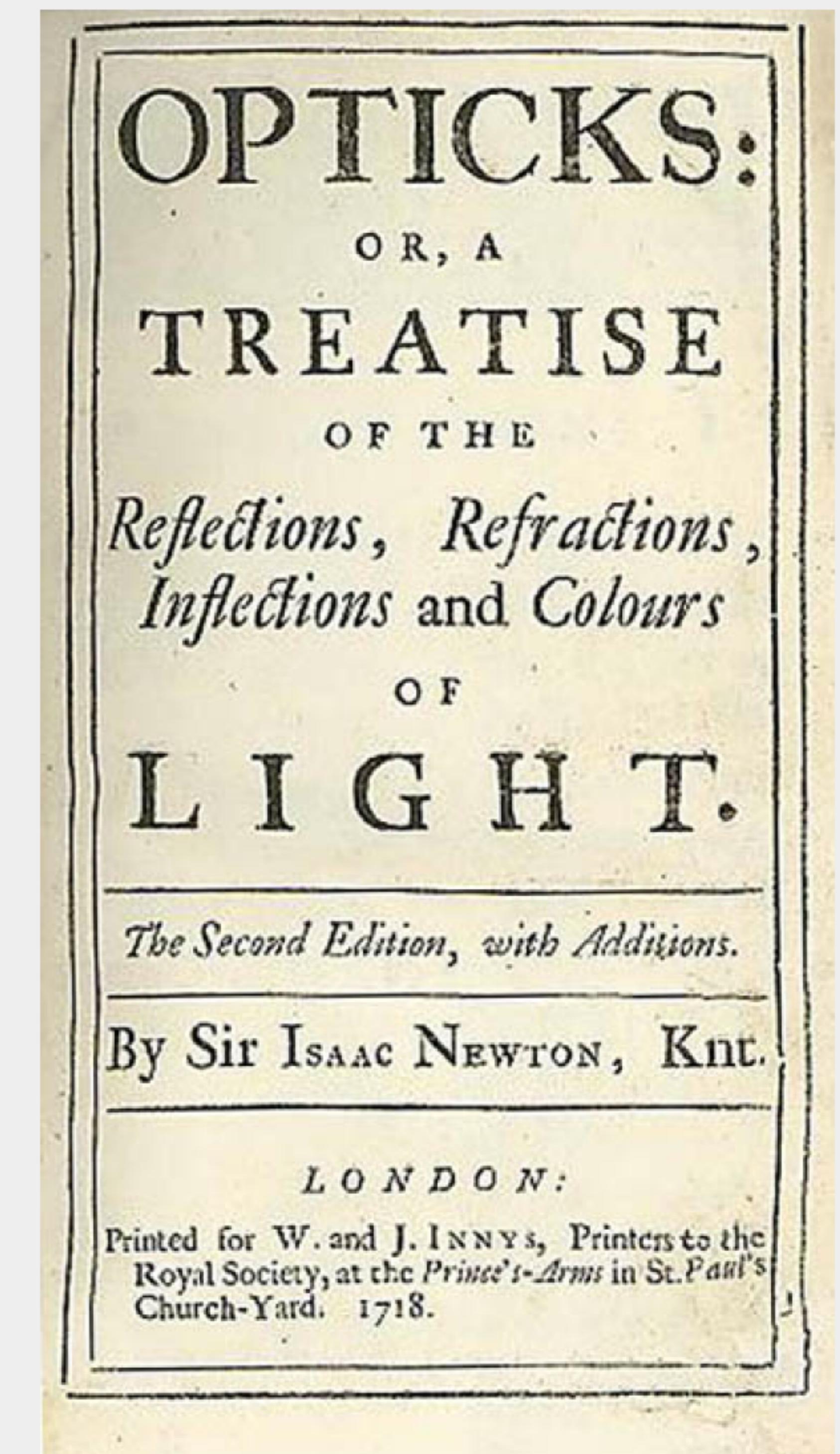
- Neutrino detectors can benefit the most, eg JUNO and Dayabay

<http://bitbucket.org/simoncblyth/opticks> □

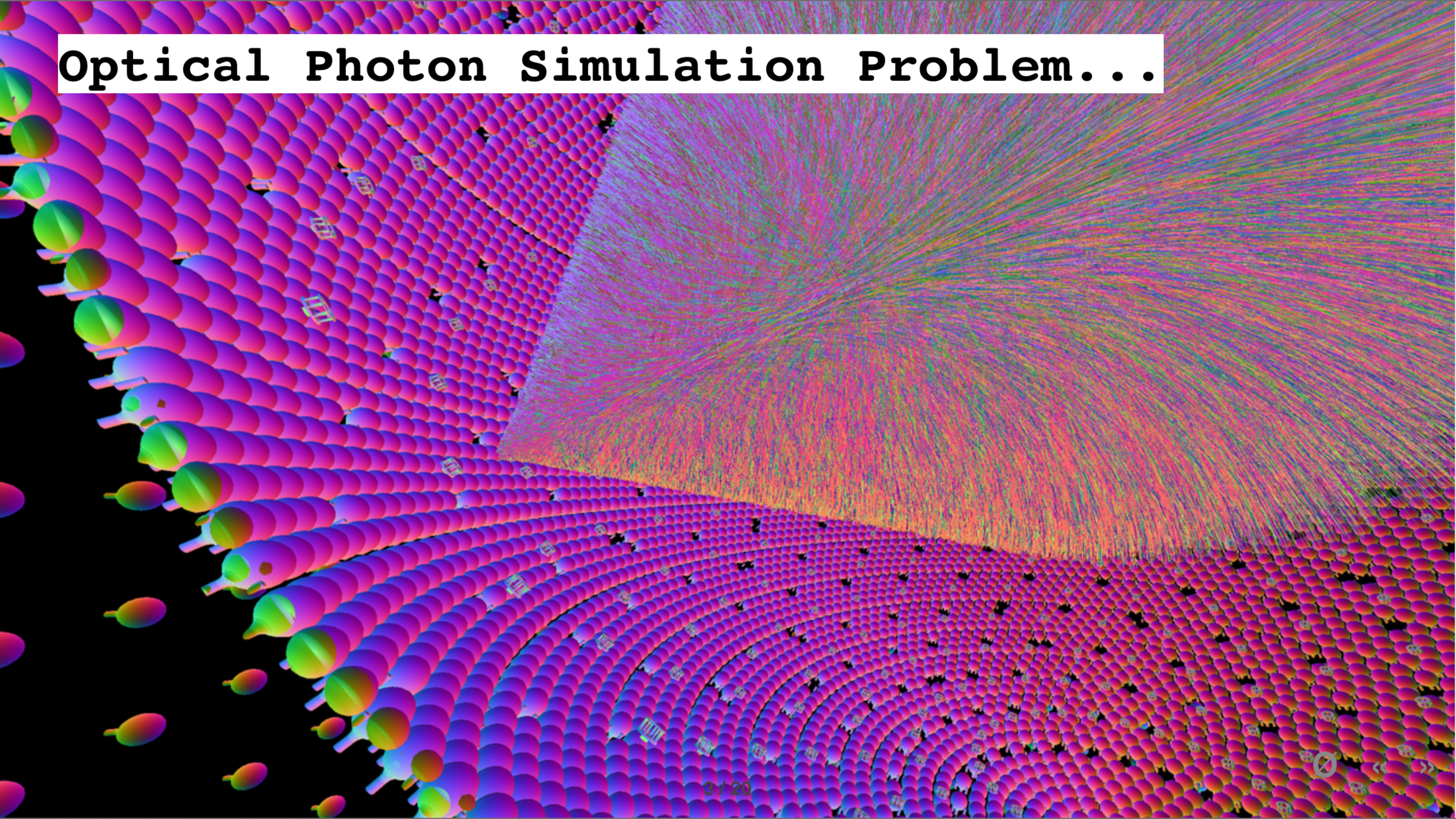
(*) core extrapolated from mobile GPU speed

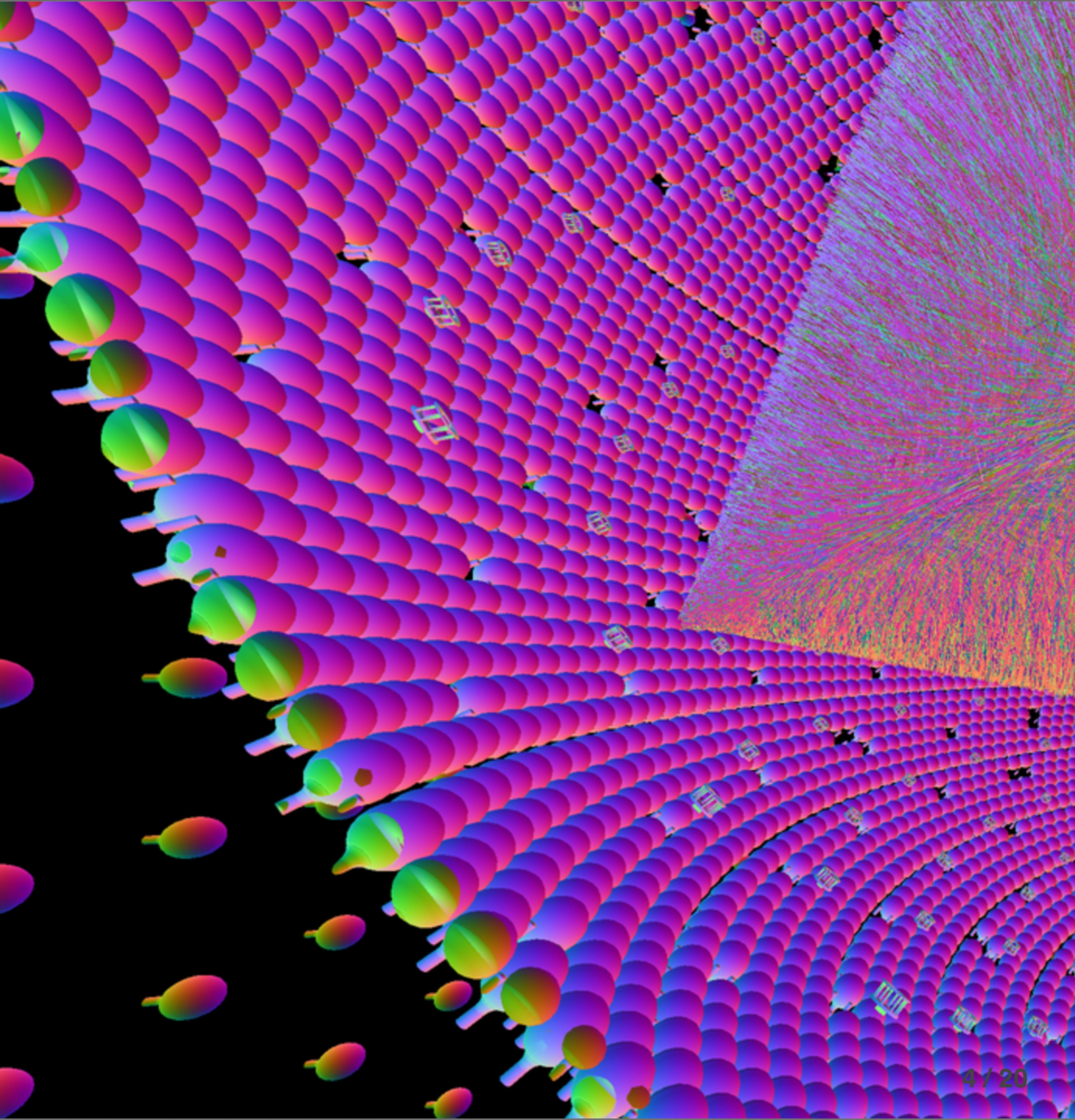
Outline

- Short video
- Problem and approach to solving
 - Optical Photon Simulation problem
 - Hybrid solution : Geant4 + Opticks
 - NVIDIA OptiX Ray Tracing Engine
- *Opticks* : optical photon simulation with NVIDIA OptiX
 - geometry workflow : geocache, GPU textures
 - large geometry techniques, geometry modelling
 - Geant4/Opticks event workflow
- Validation, performance comparisons
- Summary



Optical Photon Simulation Problem...





Optical Photon Problem

Cosmic muon backgrounds

many millions of optical photons in Daya Bay
(more in JUNO)

- optical propagation dominates *Geant4* CPU time, ~95% for Daya Bay, more for JUNO
- severe CPU memory constraint

Optical photons:

- produced by Cerenkov+Scintillation processes
- yield only Photomultiplier PMT hits

Isolated nature -> **easily separated propagation**

Hybrid Solution Possible : Geant4 + Opticks

Ray Traced Image Synthesis ≈ Optical Photon Simulation

Geometry, light sources, optical physics ->

- pixel values at image plane
- photon parameters at detectors (eg PMTs)

Ray tracing has many applications :

- advertising, design, entertainment, games,...
- BUT : most ray tracers just render images

NVIDIA OptiX had foresight to be less specific:

- general geometry intersection API
- **OptiX is to ray tracing what OpenGL is to rasterization**

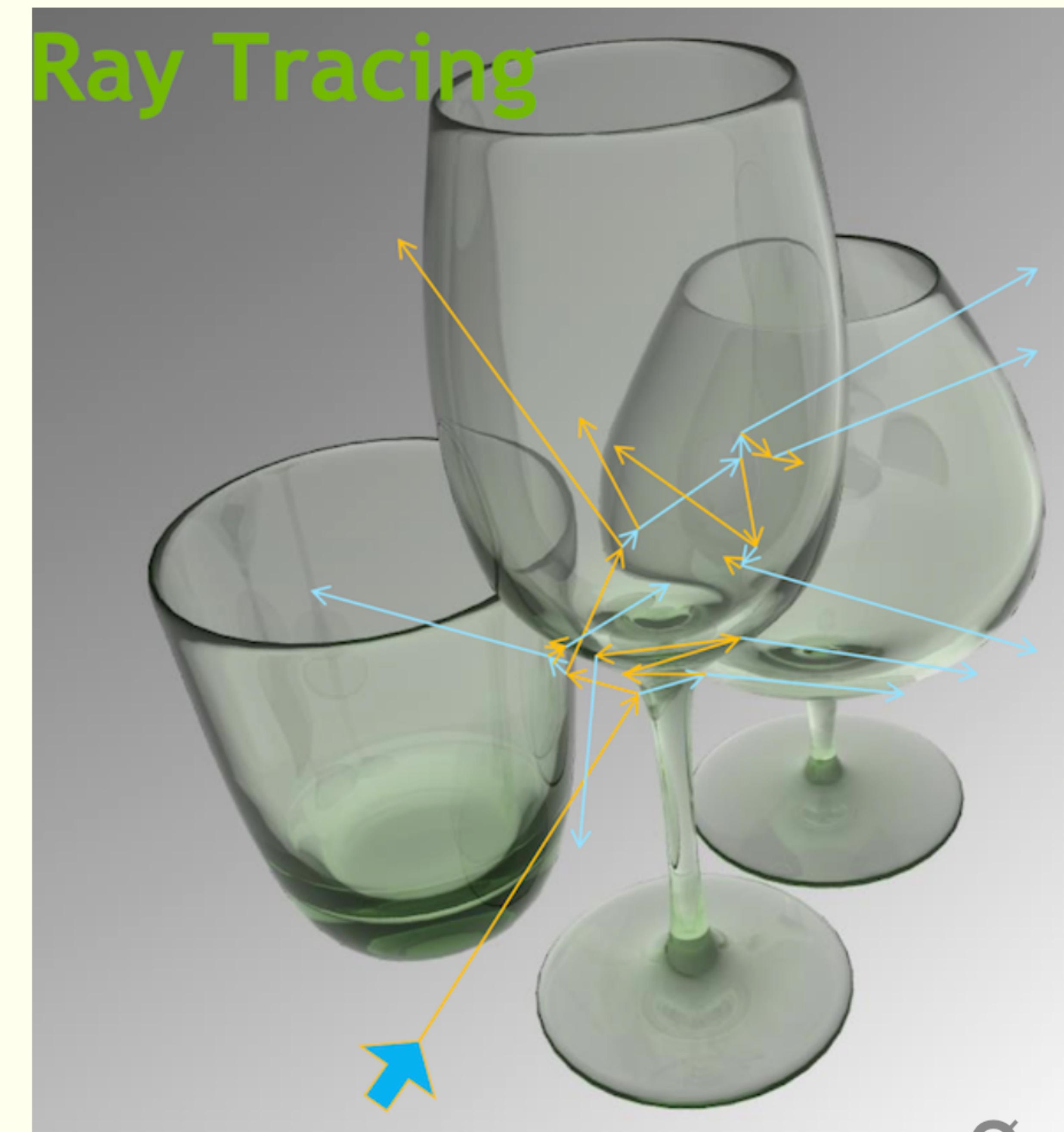
rasterization

project 3D primitives onto 2D image plane, combine fragments into pixel values

ray tracing

cast rays thru image pixels into scene, recursively reflect/refract with geometry intersected, combine returns into pixel values

OptiX Pixel Calculation



NVIDIA® OptiX™ Ray Tracing Engine

A software development kit for achieving high performance ray tracing on the GPU.



NVIDIA® OptiX™ Ray Tracing Engine

A software development kit for achieving high performance ray tracing.



<https://research.nvidia.com/publication/optix-general-purpose-ray-tracing-engine>

NVIDIA® OptiX™

<http://developer.nvidia.com/optix> □

- no rendering assumptions
- **just accelerates ray-geometry intersections**
- **compiler optimized for GPU ray tracing**
- regular improvements, new GPU tuning
- NVIDIA expertise on GPU/multi-GPU usage
 - ~linear scaling with cores across GPUs
- free to acquire, use, distribute non-commercially
- **simple : single-ray programming model**



Opticks Geometry : Recreate Geant4 "Context" on GPU

Export Tesselated Geometry

- Geant4 -> G4DAE XML
<https://bitbucket.org/simoncblyth/g4dae> □

Load G4DAE XML

- label primitives with **boundary indices**
- find repeated geometry "instances"
- write NumPy serialization `.npy` geocache

Geocache -> **few seconds startup, instead of few minutes**

Load/upload geocache

- load geocache from file
- upload OptiX textures to GPU
- upload geometry buffers to GPU

GPU geometry buffers read by:

- OptiX bounding box and intersection programs
- OpenGL GLSL shaders for visualization

Volumes -> Boundaries

CSG -> BREP

boundaries simpler than volumes on GPU

Material/surface boundary representation (4 indices)

- outer material (parent)
- outer surface (inward photons, parent -> self)
- inner surface (outward photons, self -> parent)
- inner material (self)

Primitives labelled with unique boundary index

- ray primitive intersection -> boundary index
- texture lookup -> material/surface properties

GPU textures also used for:

- scintillator reemission wavelength generation
- standard illuminant Plankian wavelength gen

Large Geometry Techniques : Instancing Mandatory

Geometry analysed to find instances

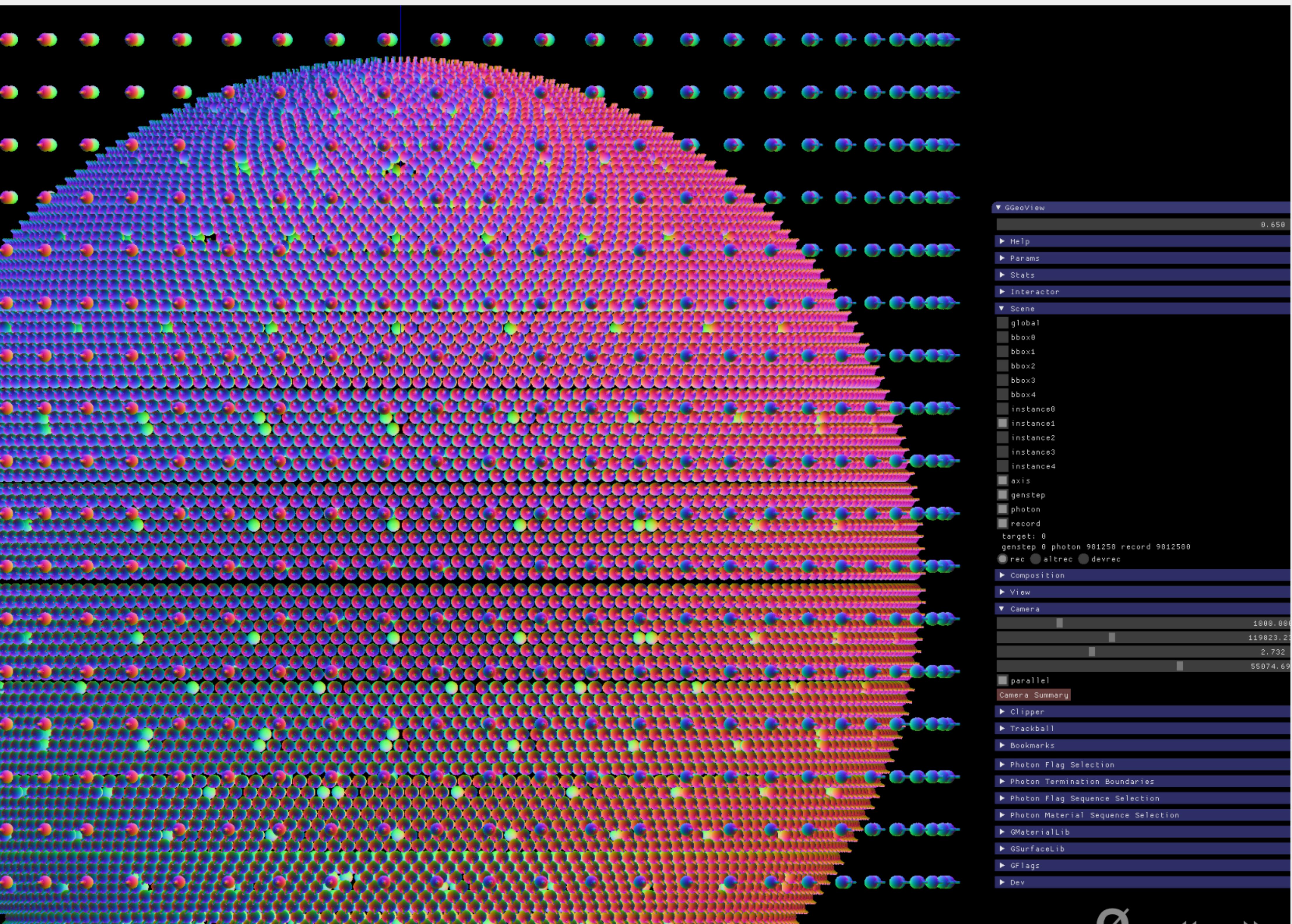
JUNO: ~90M --> 0.1M triangles

- 19,000 20" PMTs
- 34,000 3" PMTs

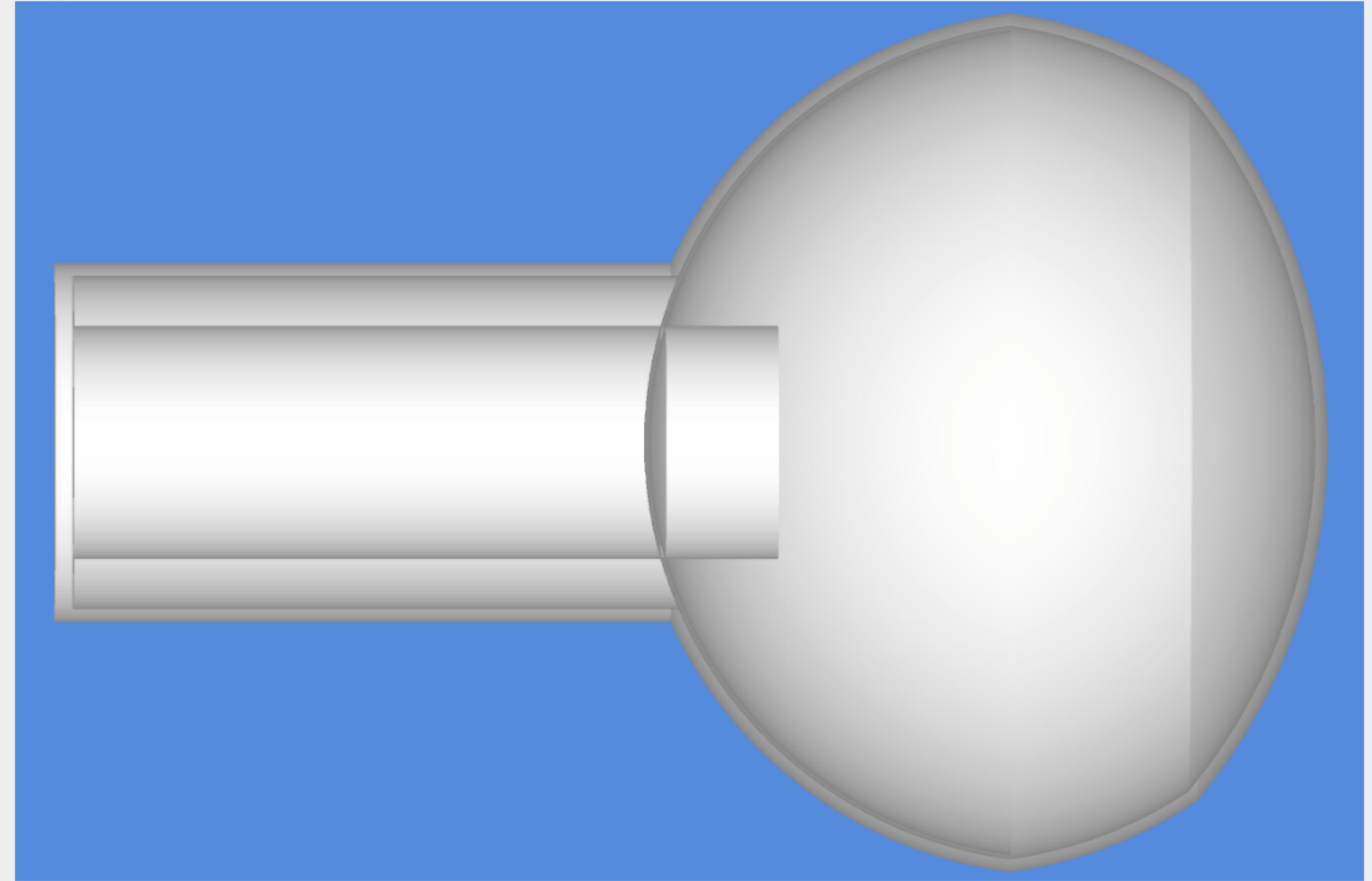
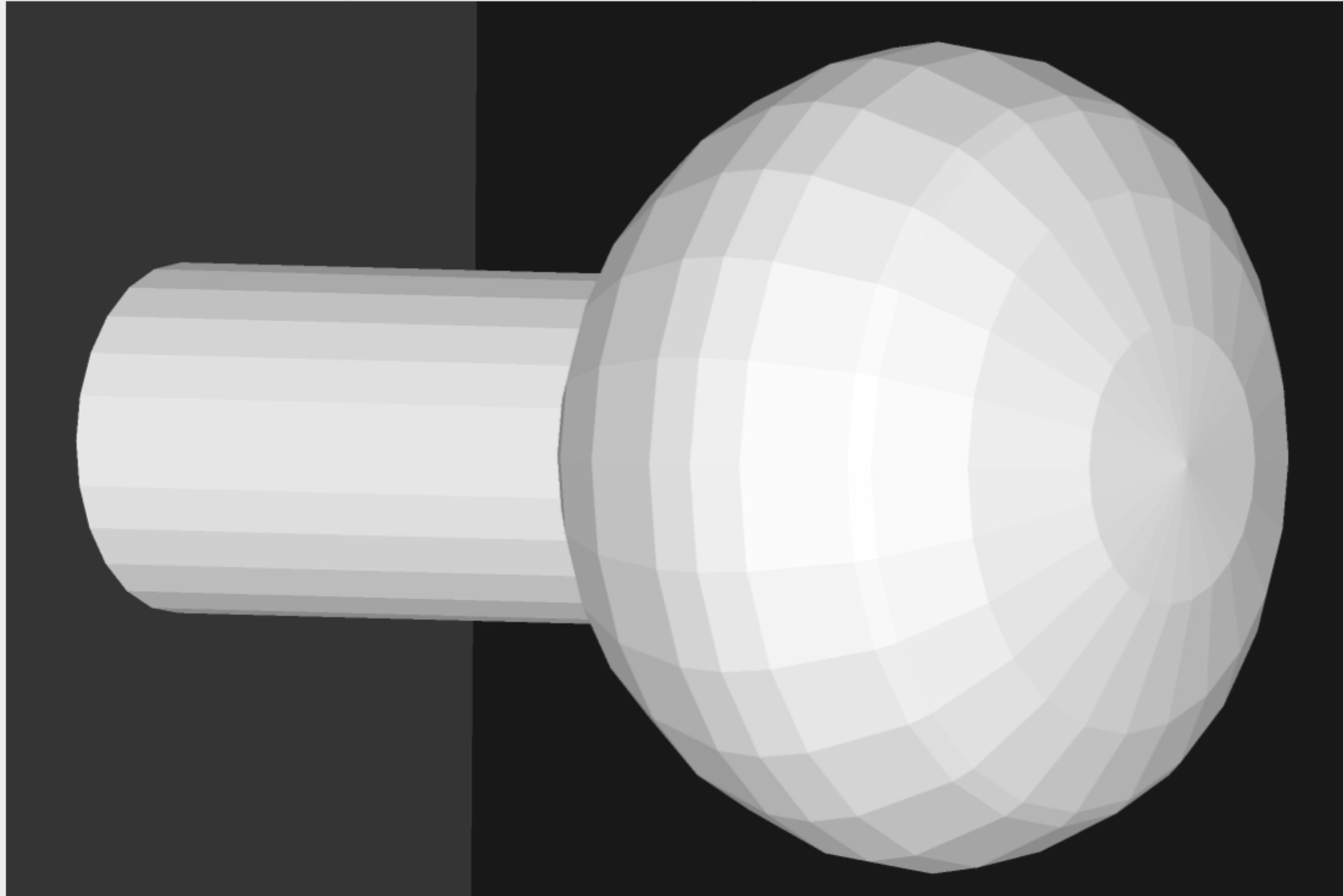
OpenGL/OptiX instancing

Multiple Renderers

- global, full instances, bbox instances
- rapid switching controls



Geometry Modelling : Tesselated vs Analytic Photomultiplier Tubes



Analytic : more realistic, faster, less memory, **much more effort**

For Dayabay PMT:

- partition CSG solids into 12 **single primitive** parts (instead of 2928 triangles)
- splitting at geometrical intersections avoids implementing general CSG boolean handling
- geometry provided to OptiX in form of ray intersection and bounding box code

Aim : **analytic description of geometry on critical optical path, remainder tesselated**

Hybrid Geant4/Opticks Event Workflow

Geant4/Detector Simulation

- modified Scintillation, Cerenkov processes
 - collect *genstep*
 - skip optical photon generation

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)

Geant4/Detector Simulation

- populate standard hit collections
- subsequent electronics simulation proceeds unaltered

Multi-event handling

- reuse/resize OptiX buffers for each event

GPU Resident Photons

Seeded on GPU

associate photons -> gensteps (via seed buffer)

Generated on GPU, using gensteps:

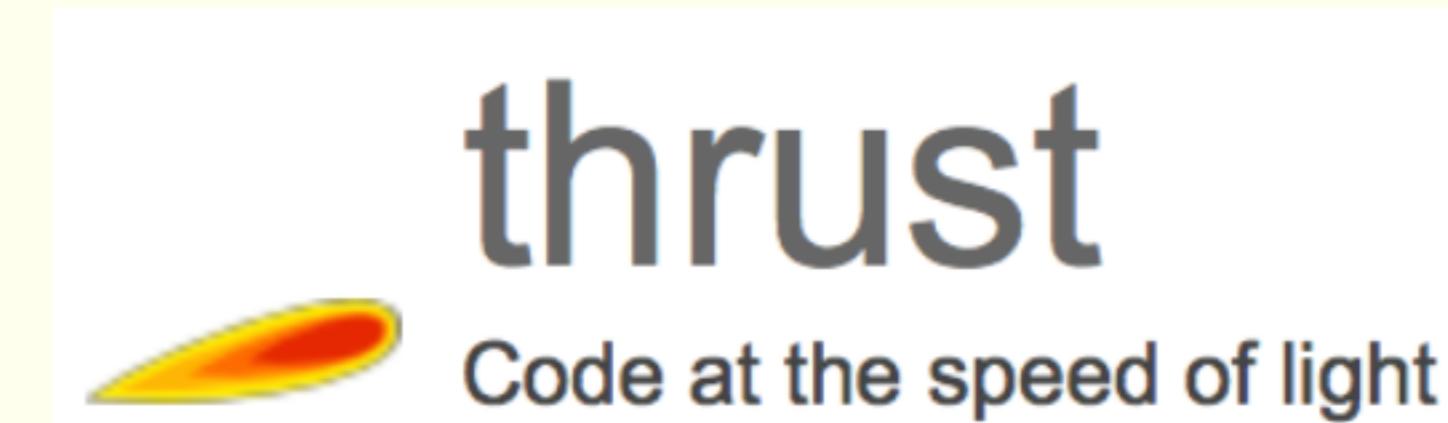
~Stacks collected before photon generation:

- number of photons to generate
- start/end position of step
- other quantities needed for GPU generation

Propagated on GPU

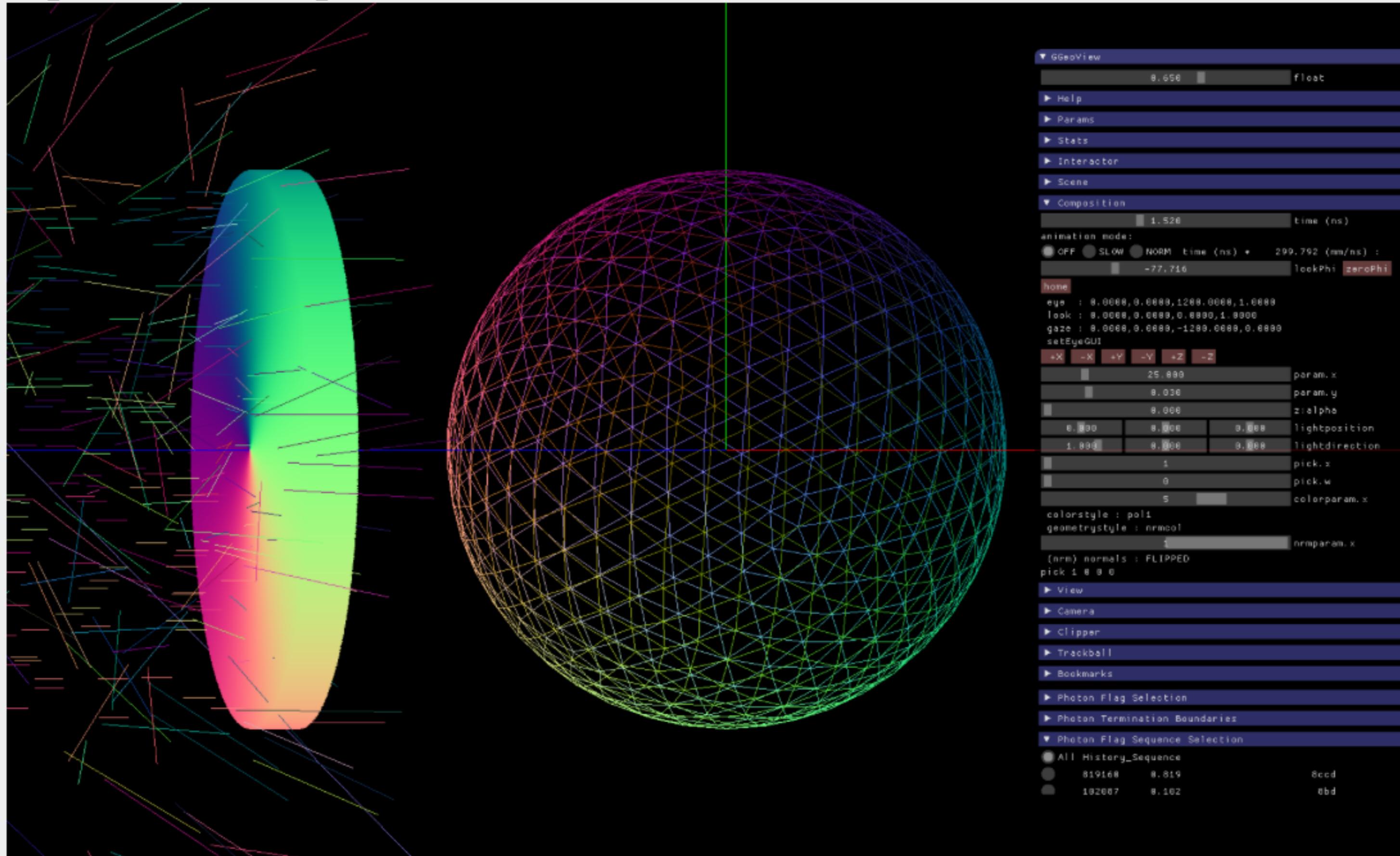
Only photons hitting PMTs copied to CPU

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

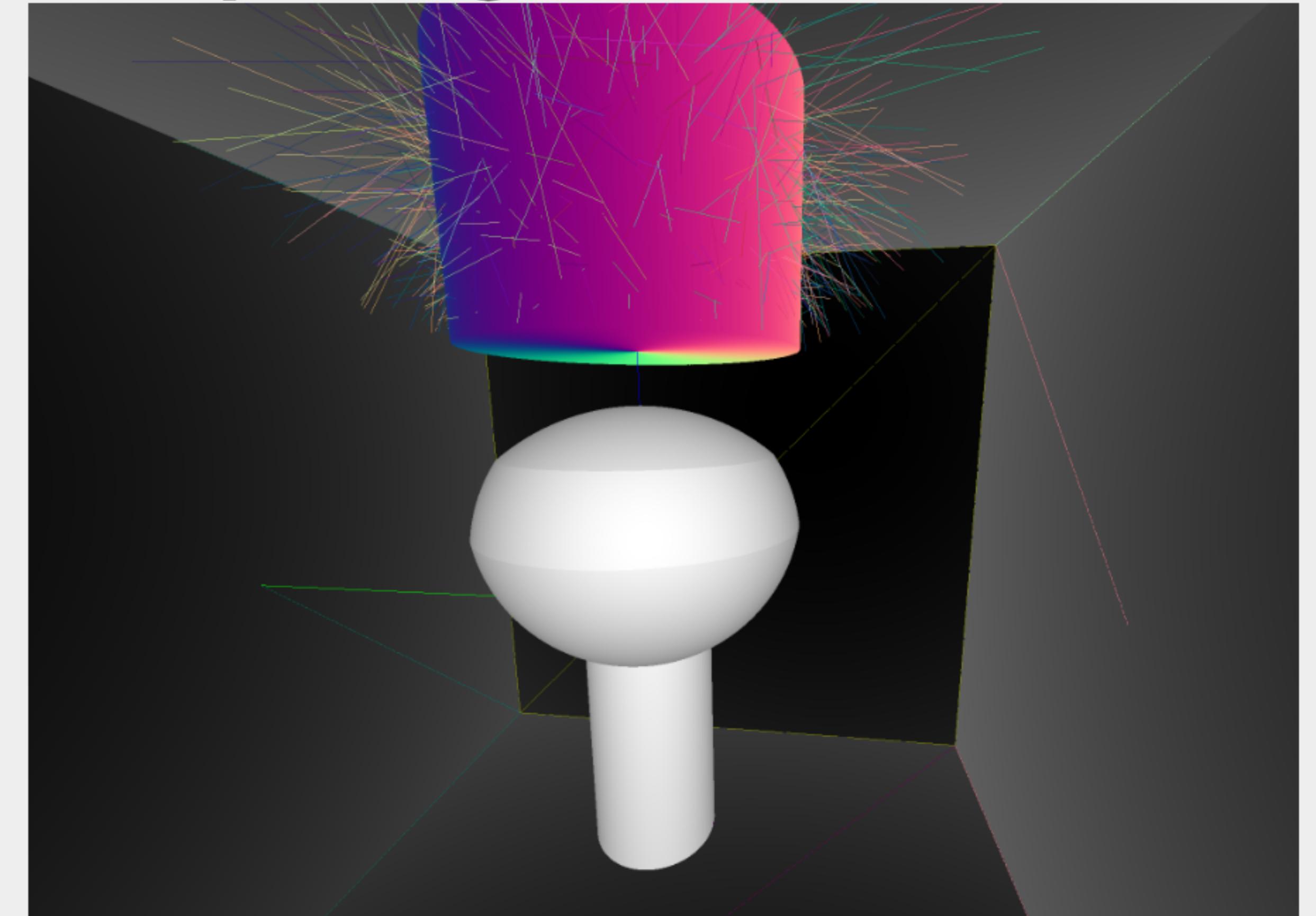


- <https://developer.nvidia.com/Thrust> □

Compare Opticks/Geant4 Simulations with Simple Lights/Geometries



1M Photons -> Water Sphere (S-Polarized)



0.5M Photons -> Dayabay PMT

Photon step records

128 bit per step : highly compressed position, time, wavelength, polarization vector, material/history codes

Photon flag sequence

16x 4-bit step flags recorded in uint64 sequence, indexed using Thrust GPU sort (1M indexed ~0.040s)

Sequence index -> interactive OpenGL selection of photons by flag sequence

Opticks/Geant4 Rainbow Step Sequence Comparison

Flags:

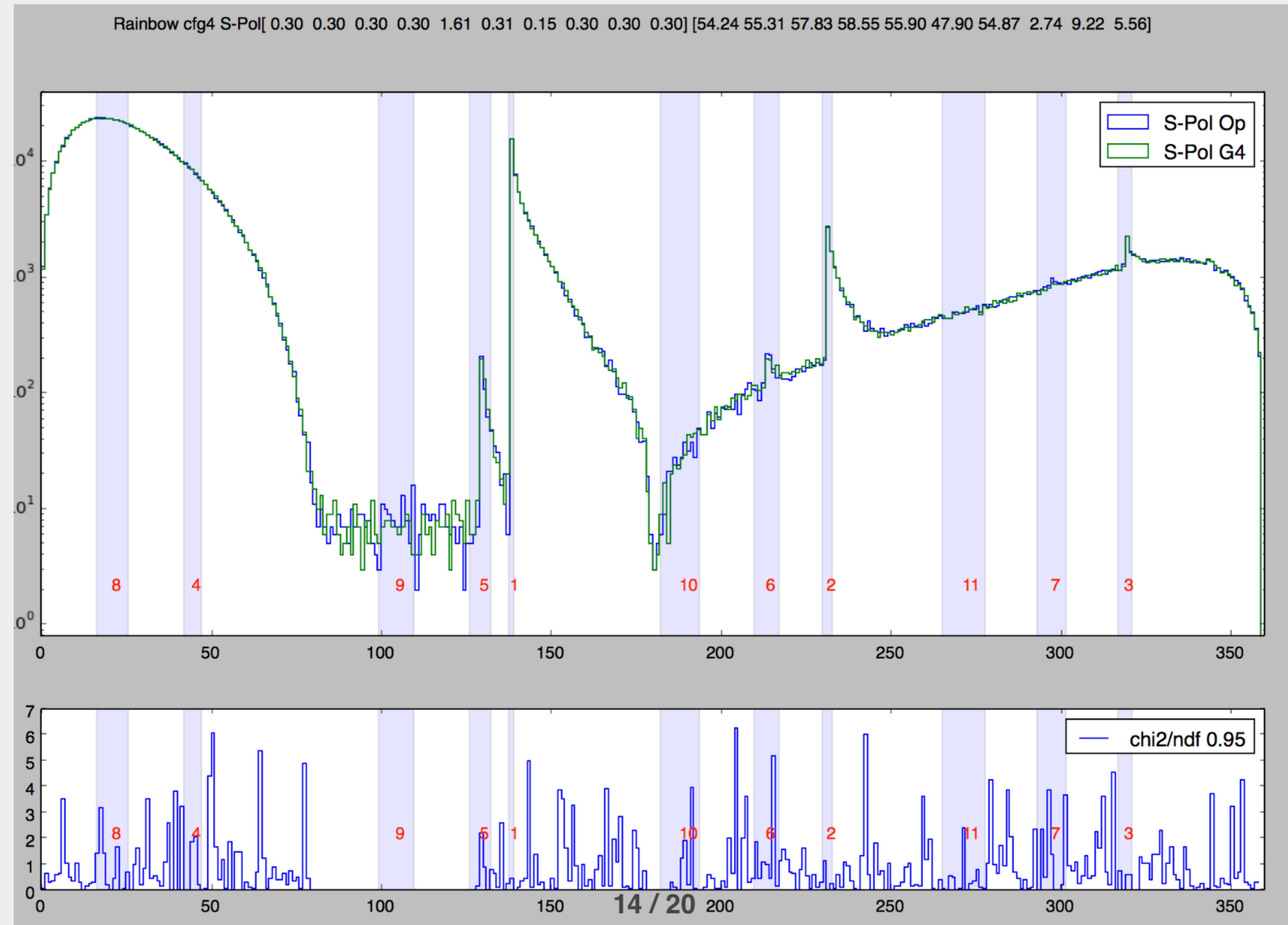
- BT/BR: boundary transmit/reflect
- TO/SC/SA: torch/scatter/surface absorb

Statistically consistent photon histories in the two simulations : Multiple orders of rainbow apparent

64-bit uint	Opticks	Geant4	chi2	(tag:5,-5)
8ccd	819160	819654	0.15 [4] TO BT BT SA	(cross droplet)
8bd	102087	101615	1.09 [3] TO BR SA	(external reflect)
8cbc	61869	61890	0.00 [5] TO BT BR BT SA	(bow 1)
8cbcb	9618	9577	0.09 [6] TO BT BR BR BT SA	(bow 2)
8cbbb	2604	2687	1.30 [7] TO BT BR BR BR BT SA	(bow 3)
8cbbbb	1056	1030	0.32 [8] TO BT BR BR BR BR BT SA	(bow 4)
86cc	1014	1000	0.10 [5] TO BT BT SC SA	
8cbbbbb	472	516	1.96 [9] TO BT BR BR BR BR BR BT SA	(bow 5)
86d	498	473	0.64 [3] TO SC SA	
bbbbbbb	304	294	0.17 [10] TO BT BR BR BR BR BR BR BR	(bow 8+ truncated)
8cbbbbbb	272	247	1.20 [10] TO BT BR BR BR BR BR BT SA	(bow 6)
cbbbbbb	183	161	1.41 [10] TO BT BR BR BR BR BR BR BT	(bow 7 truncated)

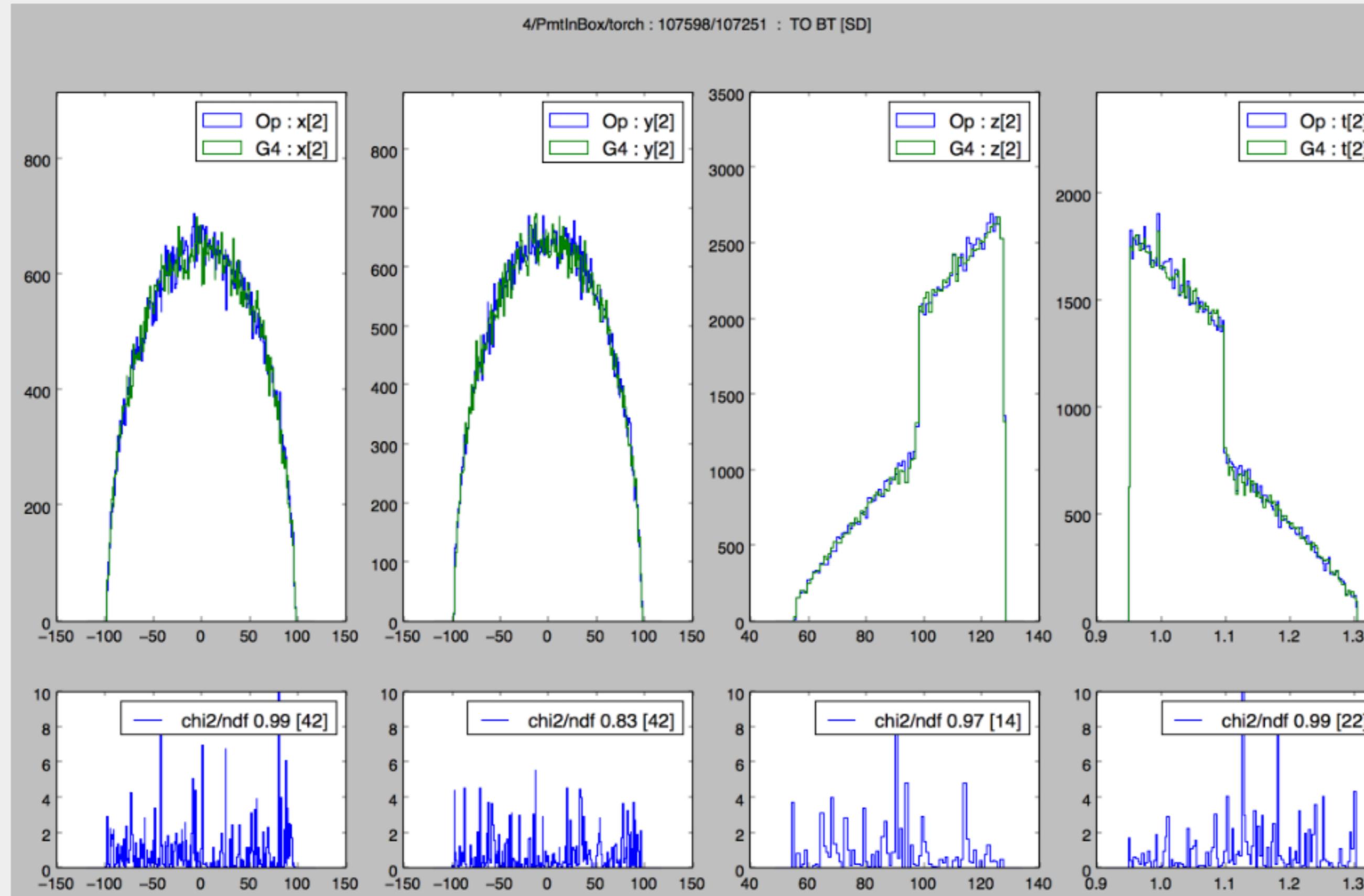
1M Rainbow S-Polarized, Comparison Opticks/Geant4

Deviation angle(degrees) of 1M parallel monochromatic photons in disc shaped beam incident on water sphere. Numbered bands are visible range expectations of first 11 rainbows. S-Polarized intersection (E field perpendicular to plane of incidence) arranged by directing polarization radially.

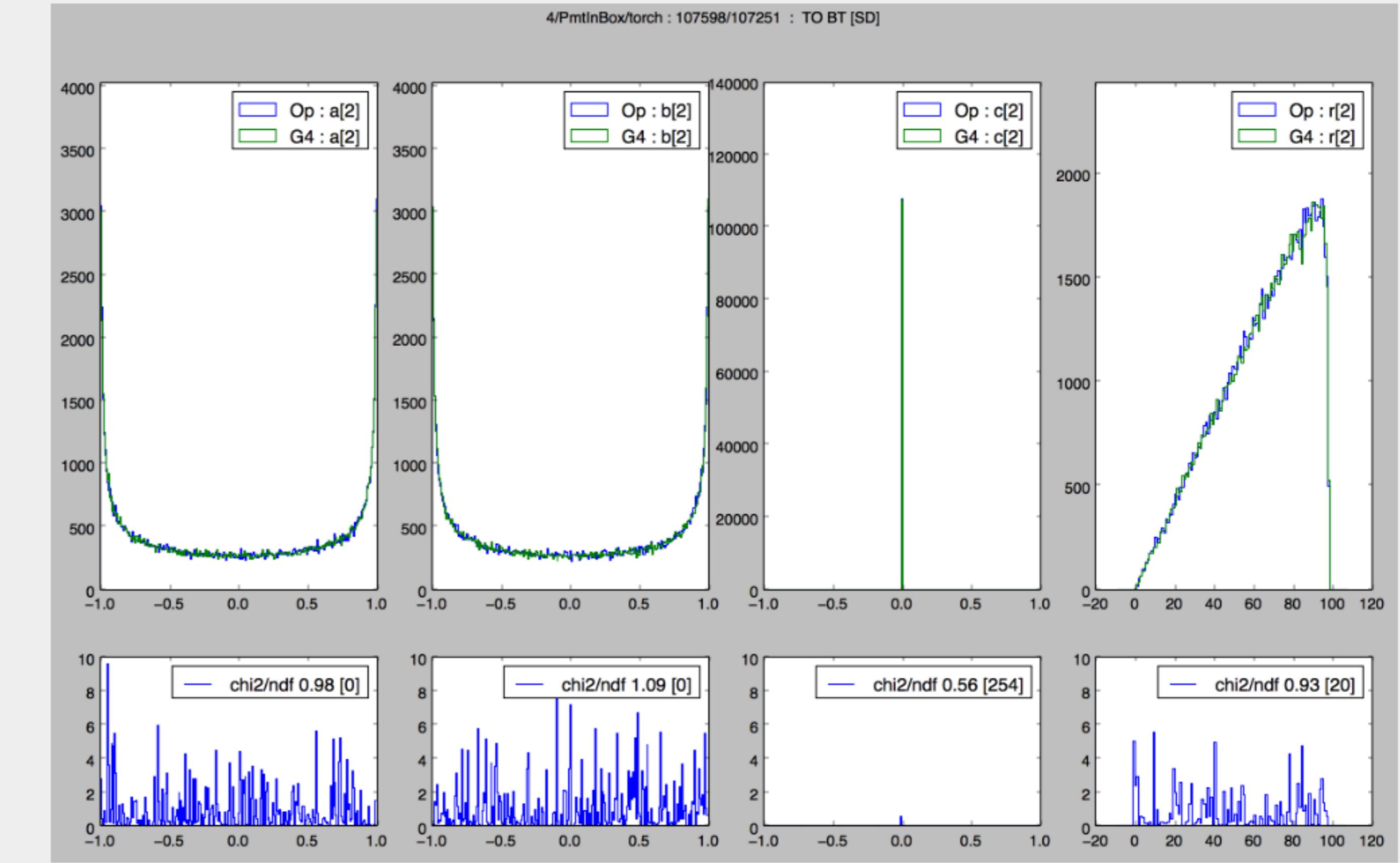


PMT Opticks/Geant4 step distribution comparison TO BT [SD]

Good agreement reached, after several fixes: geometry, total internal reflection, group velocity



position(xyz), time(t)



polarization(abc), radius(r)

PMT Opticks/Geant4 step distribution comparison : chi2/ndf

4/PMT In Box/torch :	X	Y	Z	T	A	B	C	R
340271/340273 : [TO] BT SA	1.15	1.00	0.00	0.00	1.06	1.03	0.00	1.21
340271/340273 : TO [BT] SA	1.15	1.00	1.06	0.91	1.06	1.03	0.00	1.21
340271/340273 : TO BT [SA]	0.97	1.02	1.05	0.99	1.06	1.03	0.00	1.29
107598/107251 : [TO] BT SD	0.91	0.73	0.56	0.56	0.98	1.09	0.56	0.94
107598/107251 : TO [BT] SD	0.91	0.73	0.81	0.93	0.98	1.09	0.56	0.94
107598/107251 : TO BT [SD]	0.99	0.83	0.97	0.99	0.98	1.09	0.56	0.93
23217/23260 : [TO] BT BT SA	0.94	0.82	0.04	0.04	0.97	0.89	0.04	0.57
23217/23260 : TO [BT] BT SA	0.94	0.82	0.70	0.50	0.97	0.89	0.04	0.57
23217/23260 : TO BT [BT] SA	0.91	0.94	0.43	0.60	0.97	0.89	0.04	0.05
23217/23260 : TO BT BT [SA]	0.94	0.88	0.04	0.35	0.97	0.89	0.04	0.72
18866/19048 : [TO] AB	0.99	1.10	0.87	0.87	0.85	0.84	0.87	1.00
18866/19048 : TO [AB]	0.99	1.10	0.93	0.92	0.85	0.84	0.87	1.00
3179/3133 : [TO] SC SA	1.07	0.83	0.34	0.34	0.86	0.96	0.34	0.73
3179/3133 : TO [SC] SA	1.07	0.83	0.98	1.05	0.98	1.06	0.98	0.73
3179/3133 : TO SC [SA]	0.96	1.04	0.93	0.97	0.98	1.06	0.98	1.10
2204/2249 : [TO] BT AB	0.85	1.04	0.45	0.45	0.99	0.92	0.45	1.06
2204/2249 : TO [BT] AB	0.85	1.04	0.95	0.88	0.99	0.92	0.45	1.06
2204/2249 : TO BT [AB]	0.98	0.94	1.01	1.00	0.99	0.92	0.45	0.90
1696/1732 : [TO] BT BT AB	1.05	0.85	0.38	0.38	0.86	1.09	0.38	0.26
1696/1732 : TO [BT] BT AB	1.05	0.85	1.48	1.28	0.86	1.09	0.38	0.26
1696/1732 : TO BT [BT] AB	0.99	0.86	1.17	1.40	0.86	1.09	0.38	0.86
1696/1732 : TO BT BT [AB]	1.15	0.88	1.08	1.06	0.86	1.09	0.38	0.79
1446/1455 : [TO] BR SA	1.21	0.94	0.03	0.03	0.90	0.87	0.03	1.09
1446/1455 : TO [BR] SA	1.21	0.94	1.02	1.01	0.90	0.87	0.03	1.09
1446/1455 : TO BR [SA]	1.00	0.93	0.97	0.99	0.90	0.87	0.03	1.04

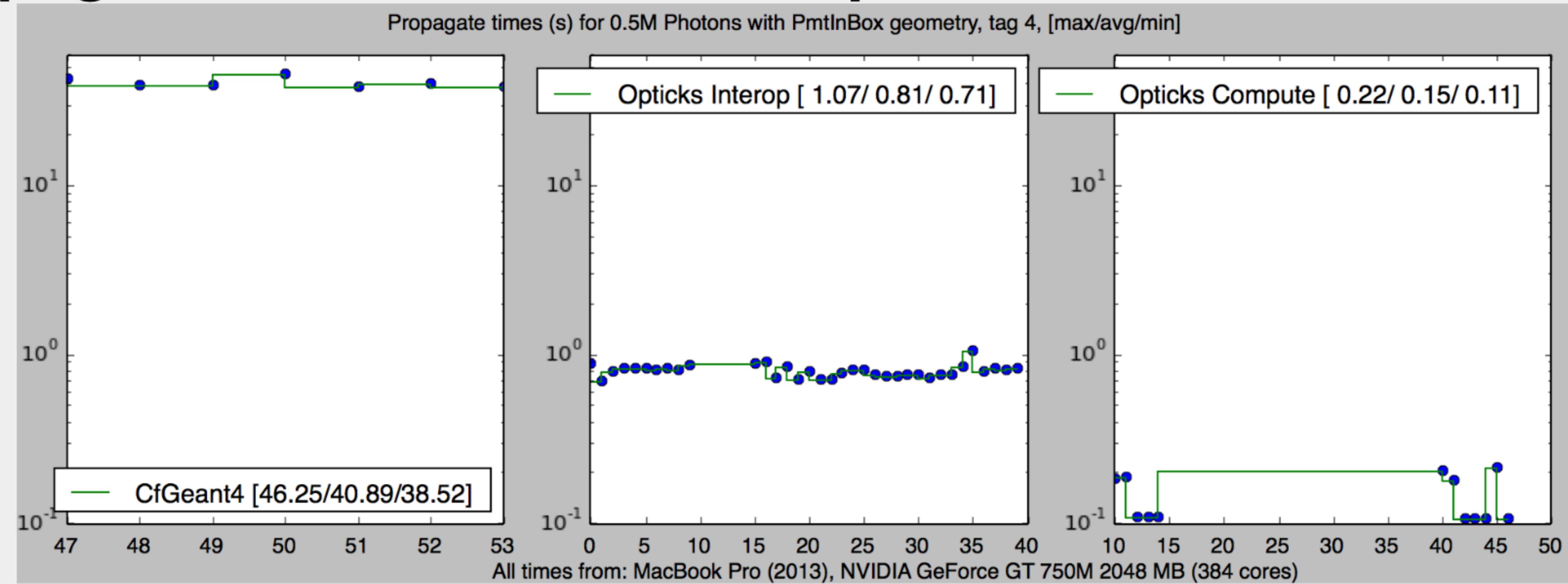
Consistent : chi2/ndf ~ 1

Very good Opticks/Geant4 agreement

- identical geometries
- identical optical physics

XYTZ: position, time ABCR: polarization, radius

Photon Propagation Times Geant4 cf Opticks



Test	Geant4 10.2	Opticks Interop	Opticks Compute
Rainbow 1M(S)	56 s	1.62 s	0.28 s
Rainbow 1M(P)	58 s	1.71 s	0.25 s
PmtInBox 0.5M	41 s	0.81 s	0.15 s

- **Opticks > 200X Geant4** with only 384 core mobile GPU[1] (multi-GPU workstation up to 20x more cores)
- **Interop** uses OpenGL buffers allowing visualization, **Compute** uses OptiX buffers
- **Interop/Compute** : perfectly identical results, monitored by digest

[1] MacBook Pro (2013), NVIDIA GeForce GT 750M, 2048 MB, 384 cores

Opticks : GPU Accelerated Optical Photon Simulation using CUDA

Version: 0.0.1

Date: August 16, 2016

The Opticks repository <http://bitbucket.org/simoncblyth/opticks> contains the sources and building externals and Opticks itself. Instructions for using these scripts can be found below.

Contents:

- [Opticks Install Instructions](#)
 - [Using a Shared Opticks Installation](#)
 - [Opticks Installation Requirements](#)
 - [OpenGL Version Requirements](#)
 - [Get Opticks](#)
 - [Build Tools](#)
 - [CMake](#)
 - [Full Building Example](#)
 - [Externals](#)
 - [Configuring and Building Opticks](#)
 - [Configuration Machinery](#)
 - [Building Opticks](#)
 - [Testing Installation](#)
 - [Issues With Tests](#)
 - [Running Opticks Scripts and Executables](#)
- [Opticks Overview](#)
 - [Project Dependencies](#)
 - [Roles of the Opticks projects](#)
- [Geometry Cache](#)
 - [Setting IDPATH](#)
 - [Structure of the geocache](#)
- [Tools : bash, python, numpy, ipython, matplotlib](#)
 - [IPython Install on a mac with pip](#)
 - [Working with bash functions](#)

Open Source Opticks

- <http://simoncblyth.bitbucket.org/opticks/>
- <http://bitbucket.org/simoncblyth/opticks/>

Documentation, install instructions. Repository.

- Mac, Linux, Windows (*)
- 16 C++ projects, ordered by dependency
- ~200 "Unit" Tests (CMake/CTest)
- 12 integration tests: tpmt, trainbow, tprism, treflect, tlens, tnewton, tg4gun, ...
- NumPy/Python analysis/debugging scripts

Geometry/event data use NumPy serialization:

```
import numpy as np
a = np.load("photons.npy")
```

(*) Windows VS2015, non-CUDA only so far

Summary

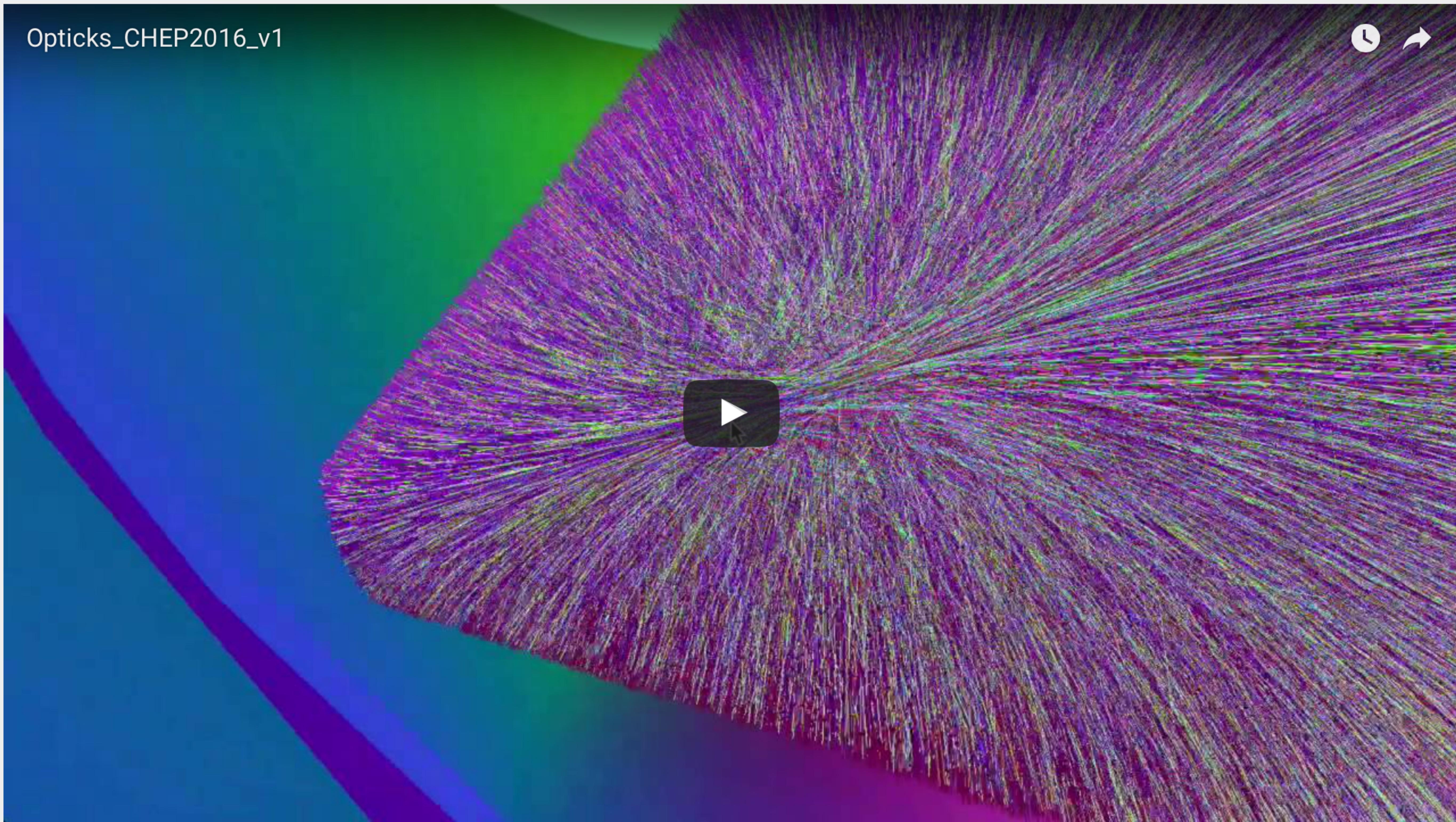
Opticks enables *Geant4* based simulations to benefit from optical photon simulation **taking effectively zero time and zero CPU memory**, thanks to massive parallelism made accessible by NVIDIA OptiX.

Overview

- **Opticks 200x Geant4** with mobile GPU
- Expect: **Opticks > 1000x Geant4** (with workstation GPUs)
- **photon propagation time --> zero**
- **analytic geometry --> precise Geant4 match**

- The more photons the bigger the overall speedup (99% -> 100x)
- Drastic speedup -> better detector understanding -> greater precision
- Large PMT based neutrino experiments, such as JUNO, can benefit the most

<https://bitbucket.org/simoncblyth/opticks> 



<https://www.youtube.com/watch?v=CBpOha4RzIs> □

