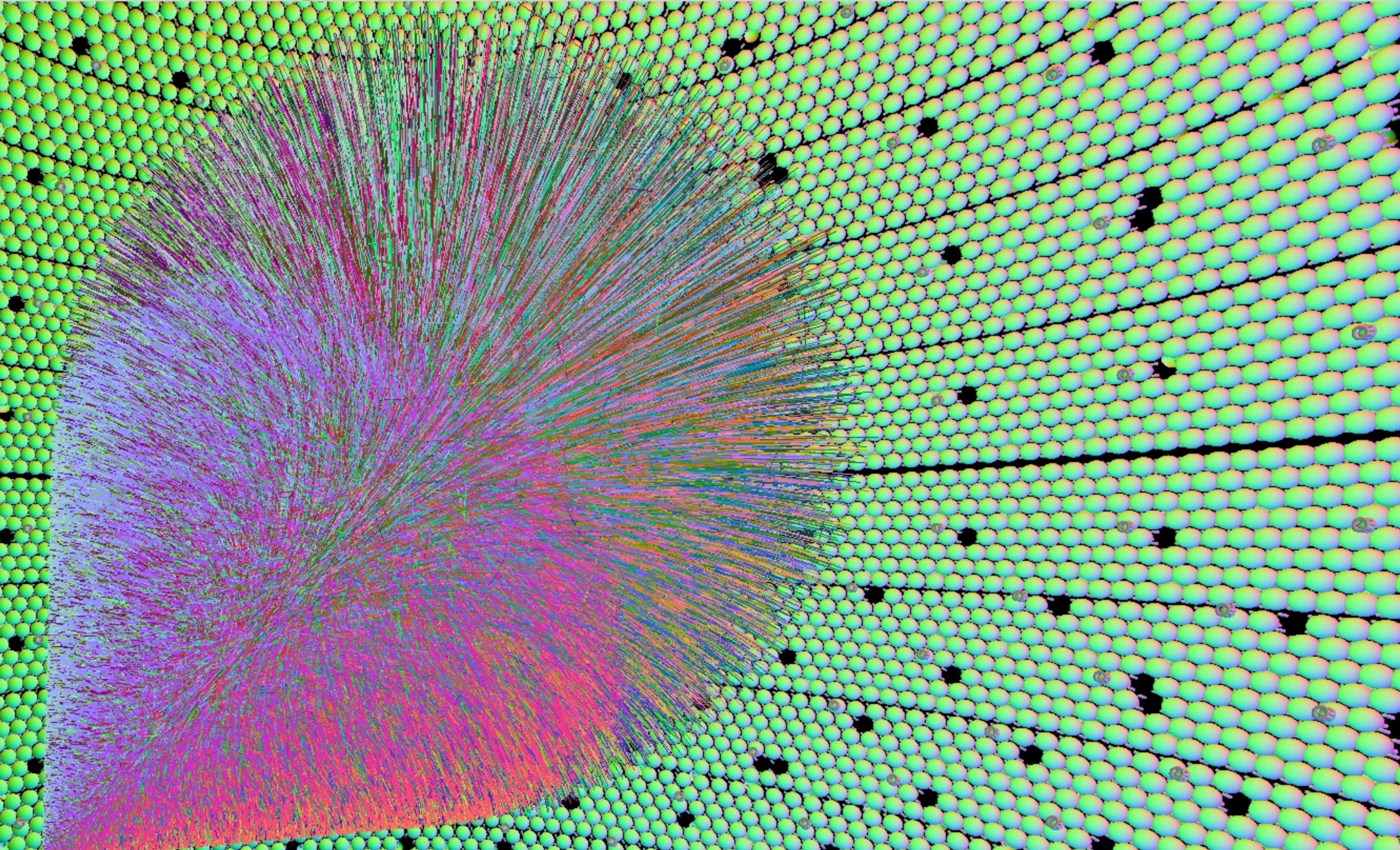


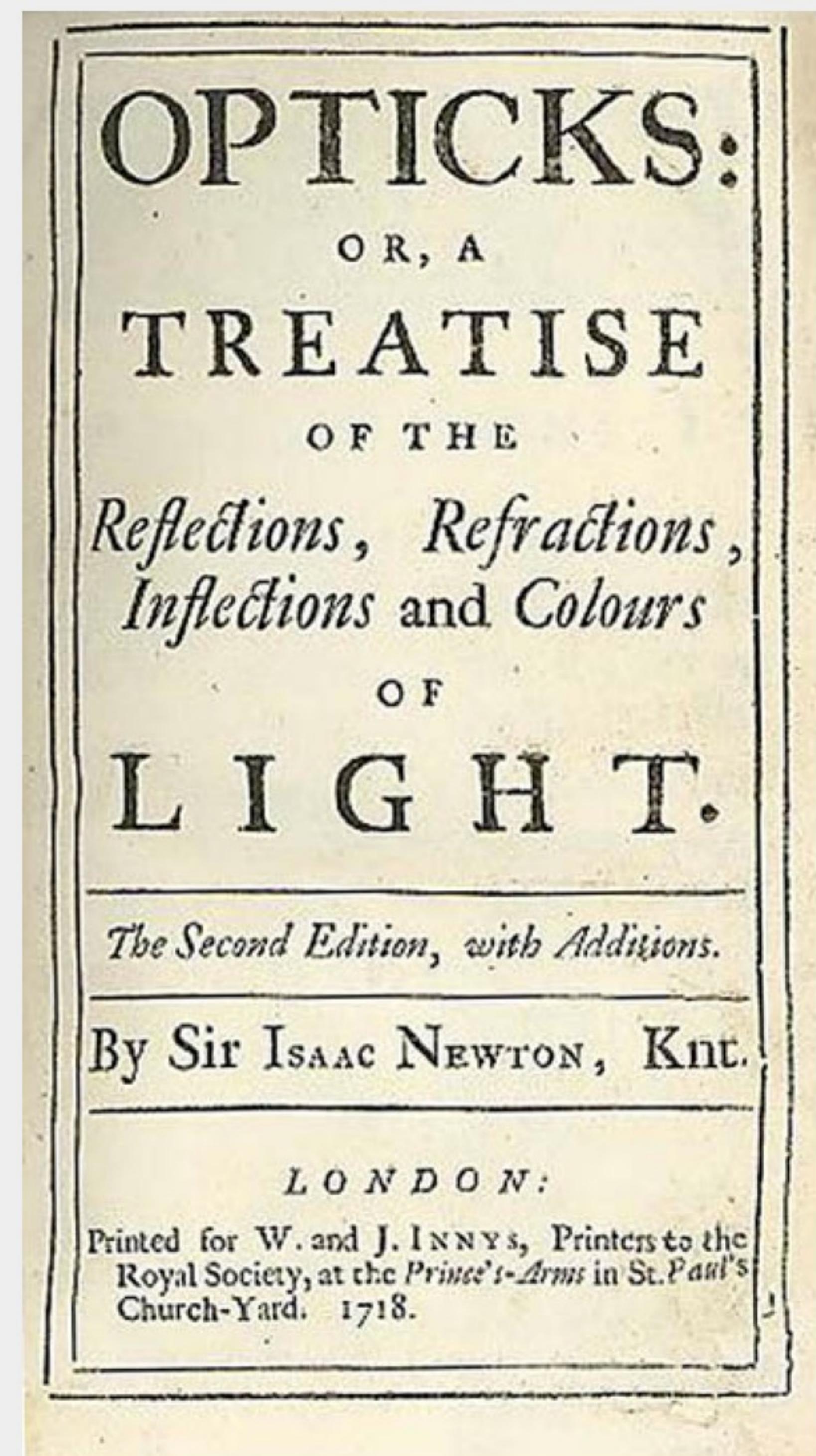
Meeting the Challenge of JUNO Simulation with Opticks : GPU Optical Photon Acceleration via NVIDIA® OptiX™

Open source, <https://bitbucket.org/simoncblyth/opticks>



Outline

- Context and Problem
 - Jiangmen Underground Neutrino Observatory (JUNO)
 - Optical Photon Simulation Problem...
- Tools to create Solution
 - Optical Photon Simulation \approx Ray Traced Image Rendering
 - Rasterization and Ray tracing
 - Turing Built for RTX
 - BVH : Bounding Volume Hierarchy
 - NVIDIA OptiX Ray Tracing Engine
- Opticks : The Solution
 - Geant4 + Opticks Hybrid Workflow : External Optical Photon Simulation
 - Opticks : Translates G4 Optical Physics to CUDA/OptiX
 - Opticks : Translates G4 Geometry to GPU, Without Approximation
 - CUDA/OptiX Intersection Functions for \sim 10 Primitives
 - CUDA/OptiX Intersection Functions for Arbitrarily Complex CSG Shapes
- Validation and Performance
 - Random Aligned Bi-Simulation -> Direct Array Comparison
 - Performance Scanning from 1M to 400M Photons
- Overview + Links



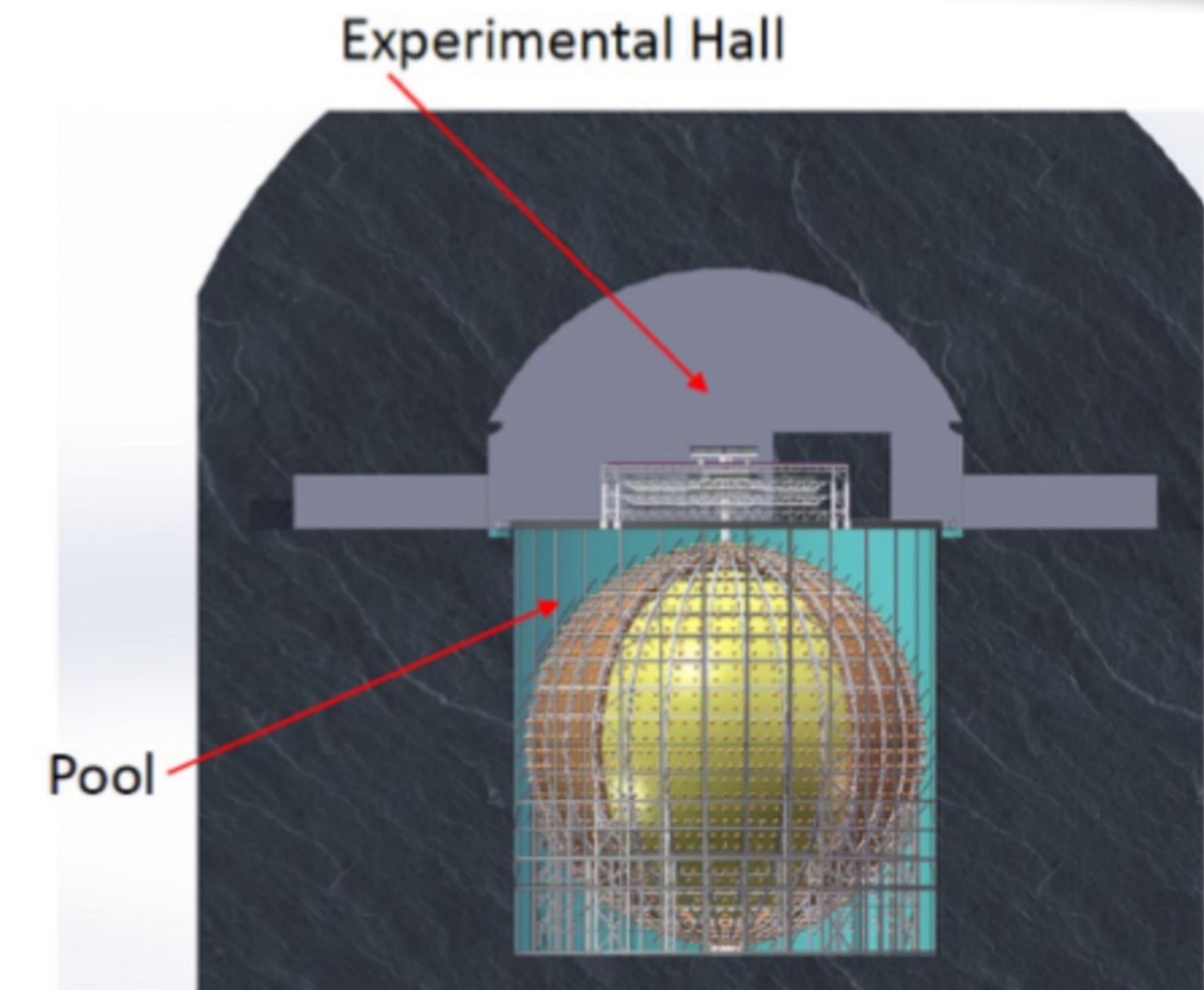
Jiangmen Underground Neutrino Observatory (JUNO)

Complete the construction by 2021

Designed to determine Neutrino Mass Ordering from reactor electron anti-neutrinos



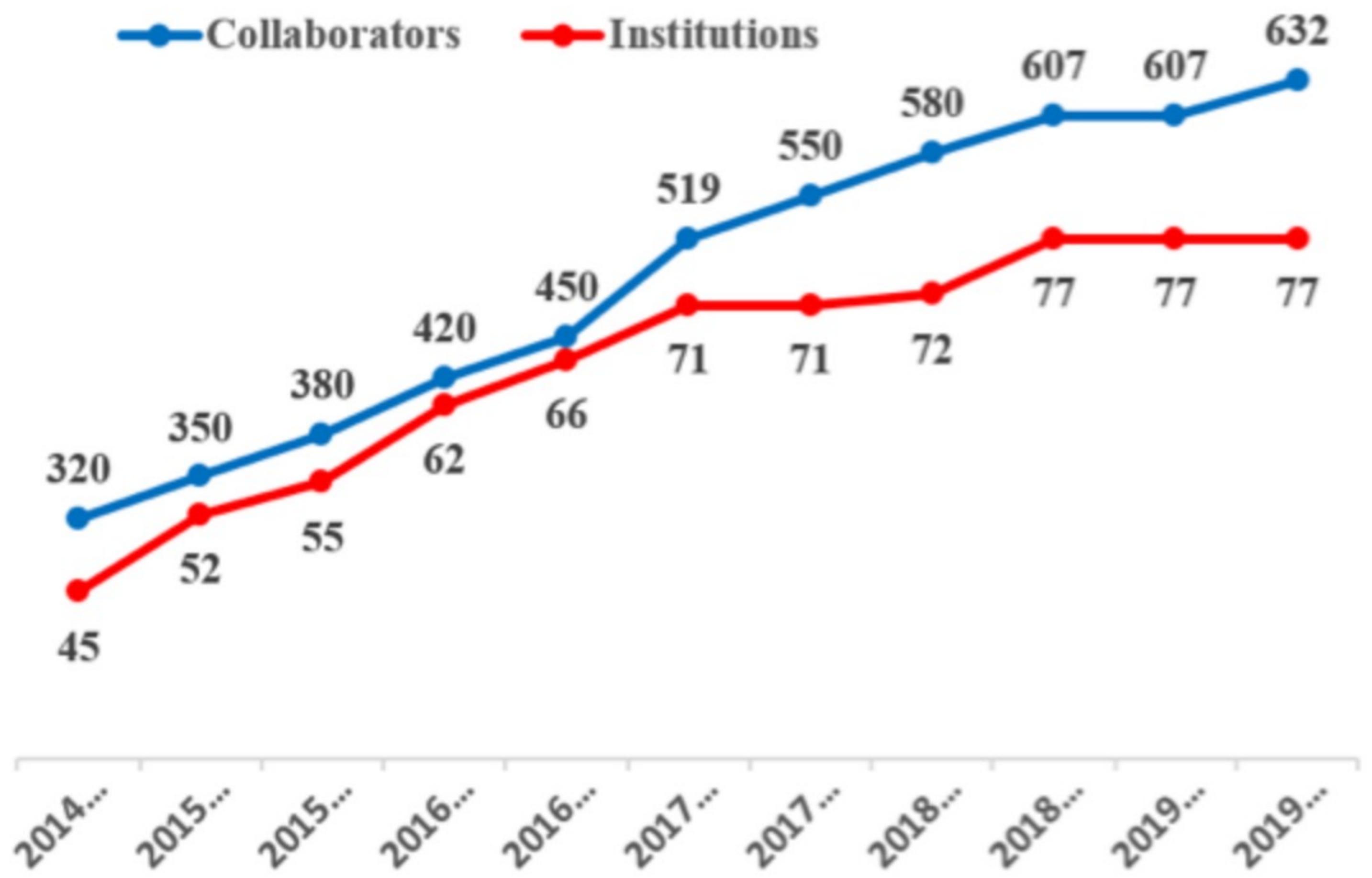
Neutrinos from two Nuclear Power Plants
26.6 GW_{th} power by 2020 (35.8 GW_{th} final)



JUNO Central Detector
20 kton Liquid Scintillator Target²

JUNO Central Detector : 35 m diameter sphere of liquid scintillator : world's largest LS detector

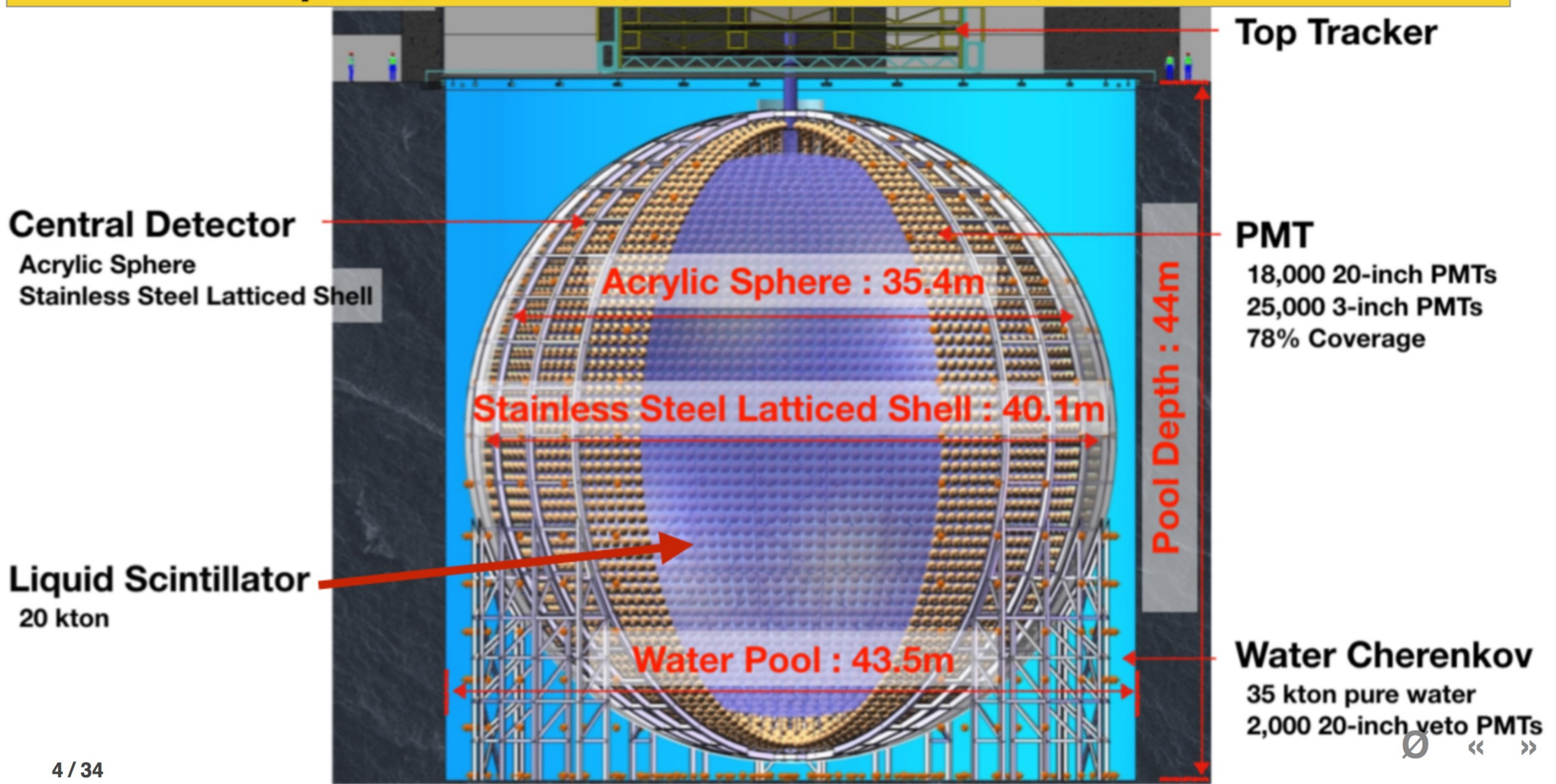
JUNO Collaboration : > 600 Collaborators, 77 Institutions, 17 Countries/Regions



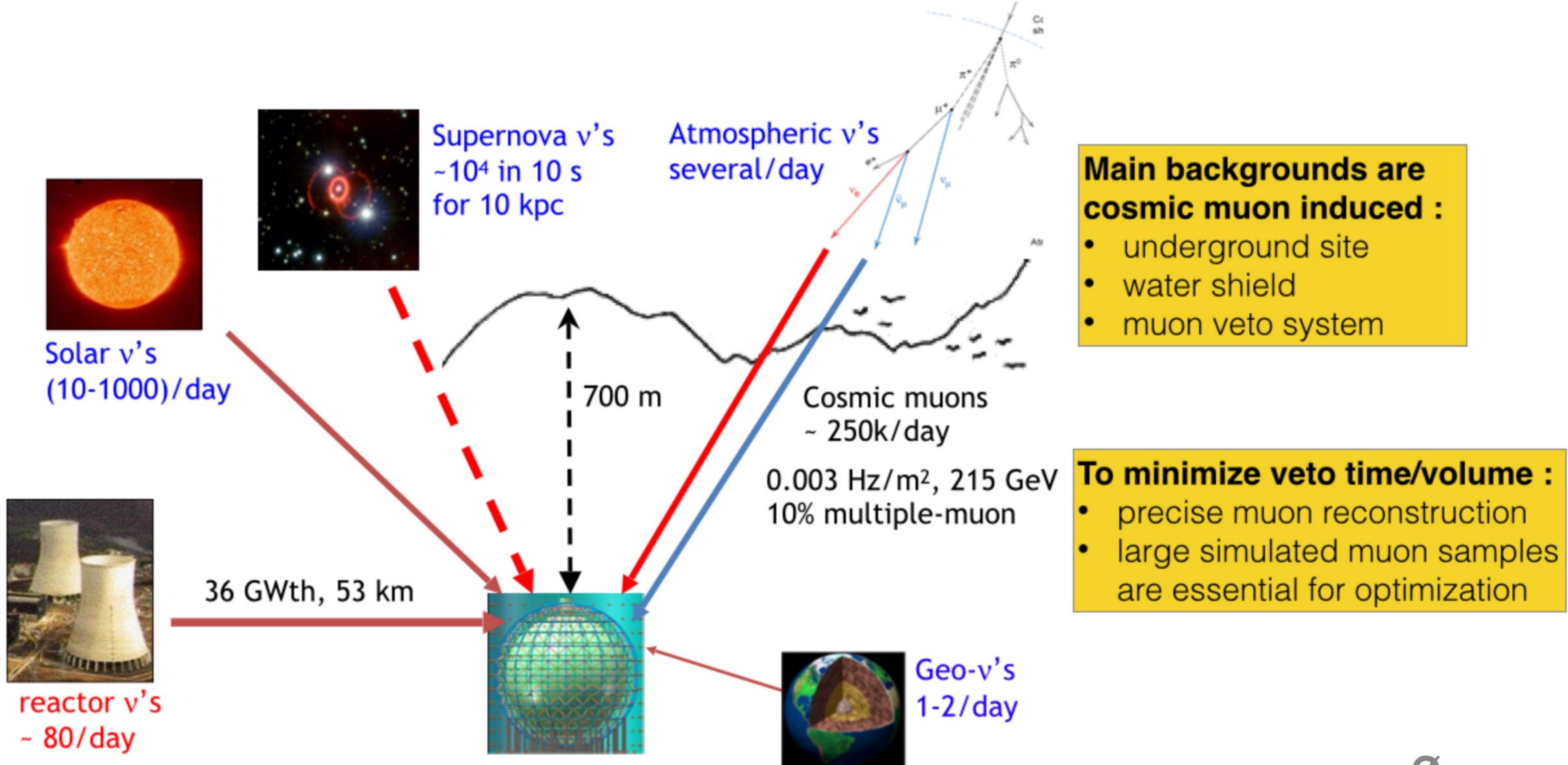
Country	Institute	Country	Institute	Country	Institute
Armenia	Yerevan Physics Institute	China	IMP-CAS	Germany	U. Mainz
Belgium	Universite libre de Bruxelles	China	SYSU	Germany	U. Tuebingen
Brazil	PUC	China	Tsinghua U.	Italy	INFN Catania
Brazil	UEL	China	UCAS	Italy	INFN di Frascati
Chile	PCUC	China	USTC	Italy	INFN-Ferrara
Chile	UTFSM	China	U. of South China	Italy	INFN-Milano
China	BISEE	China	Wu Yi U.	Italy	INFN-Milano Bicocca
China	Beijing Normal U.	China	Wuhan U.	Italy	INFN-Padova
China	CAGS	China	Xi'an JT U.	Italy	INFN-Perugia
China	ChongQing University	China	Xiamen University	Italy	INFN-Roma 3
China	CLAE	China	Zhengzhou U.	Latvia	IECS
China	DGUT	China	NUDT	Pakistan	PINSTECH (PAEC)
China	ECUST	China	CUG-Beijing	Russia	INR Moscow
China	Guangxi U.	China	ECUT-Nanchang City	Russia	JINR
China	Harbin Institute of Technology	Czech R.	Charles University	Russia	MSU
China	IHEP	Finland	University of Jyvaskyla	Slovakia	FMPICU
China	Jilin U.	France	LAL Orsay	Taiwan-China	National Chiao-Tung U.
China	Jinan U.	France	CENBG Bordeaux	Taiwan-China	National Taiwan U.
China	Nanjing U.	France	CPPM Marseille	Taiwan-China	National United U.
China	Nankai U.	France	IPHC Strasbourg	Thailand	NARIT
China	NCEPU	France	Subatech Nantes	Thailand	PPRLCU
China	Pekin U.	Germany	FZJ-ZEA	Thailand	SUT
China	Shandong U.	Germany	RWTH Aachen U.	USA	UMD1
China	Shanghai JT U.	Germany	TUM	USA	UMD2
China	IGG-Beijing	Germany	U. Hamburg	USA	UC Irvine
China	IGG-Wuhan	Germany	FZJ-IKP		



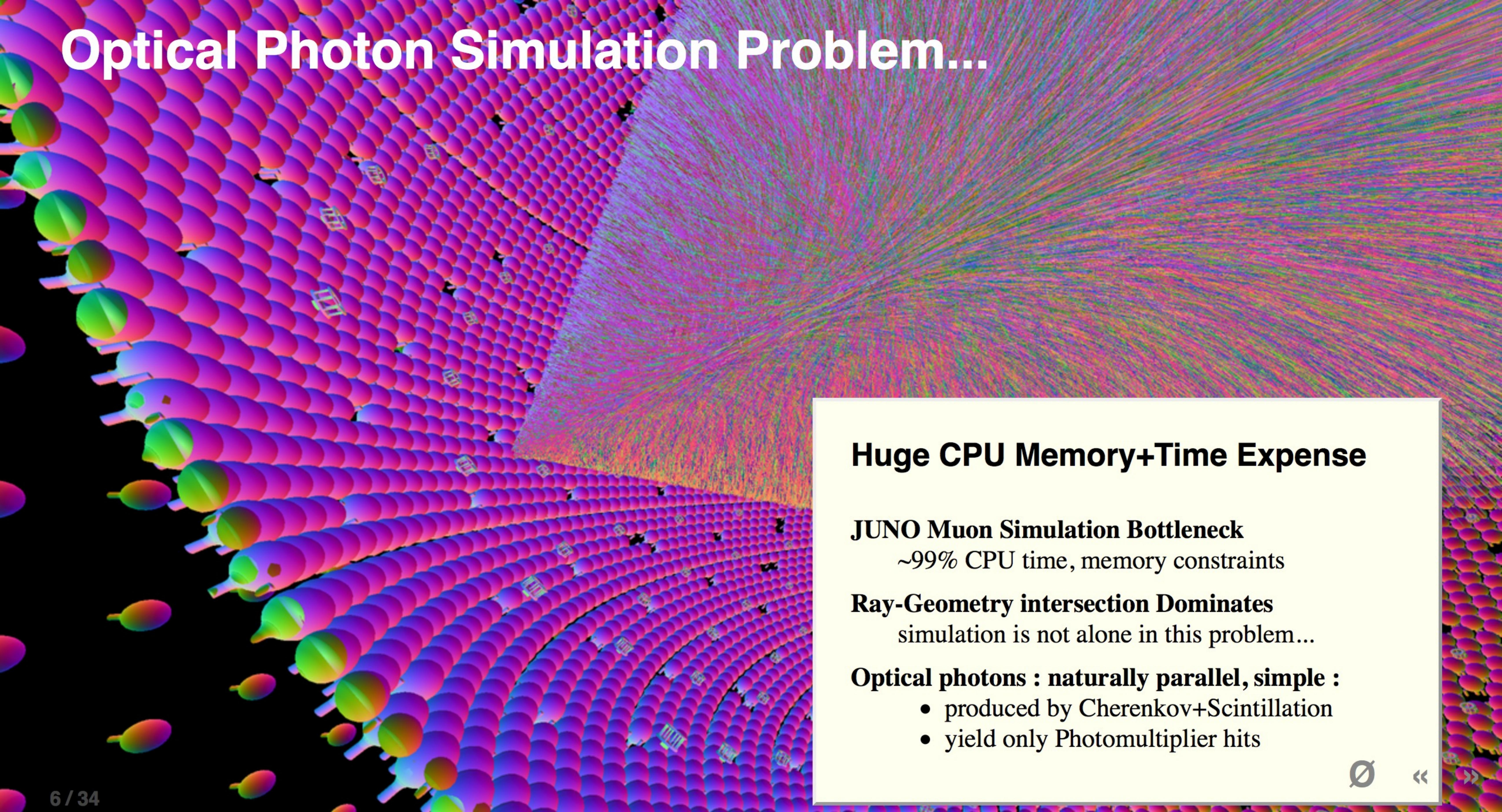
JUNO : Liquid Scintillator, 18k 20-inch PMTs, 25k 3-inch PMTs



JUNO : A Multipurpose Neutrino Observatory



Optical Photon Simulation Problem...



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



Optical Photon Simulation ≈ Ray Traced Image Rendering

Much in common : geometry, light sources, optical physics

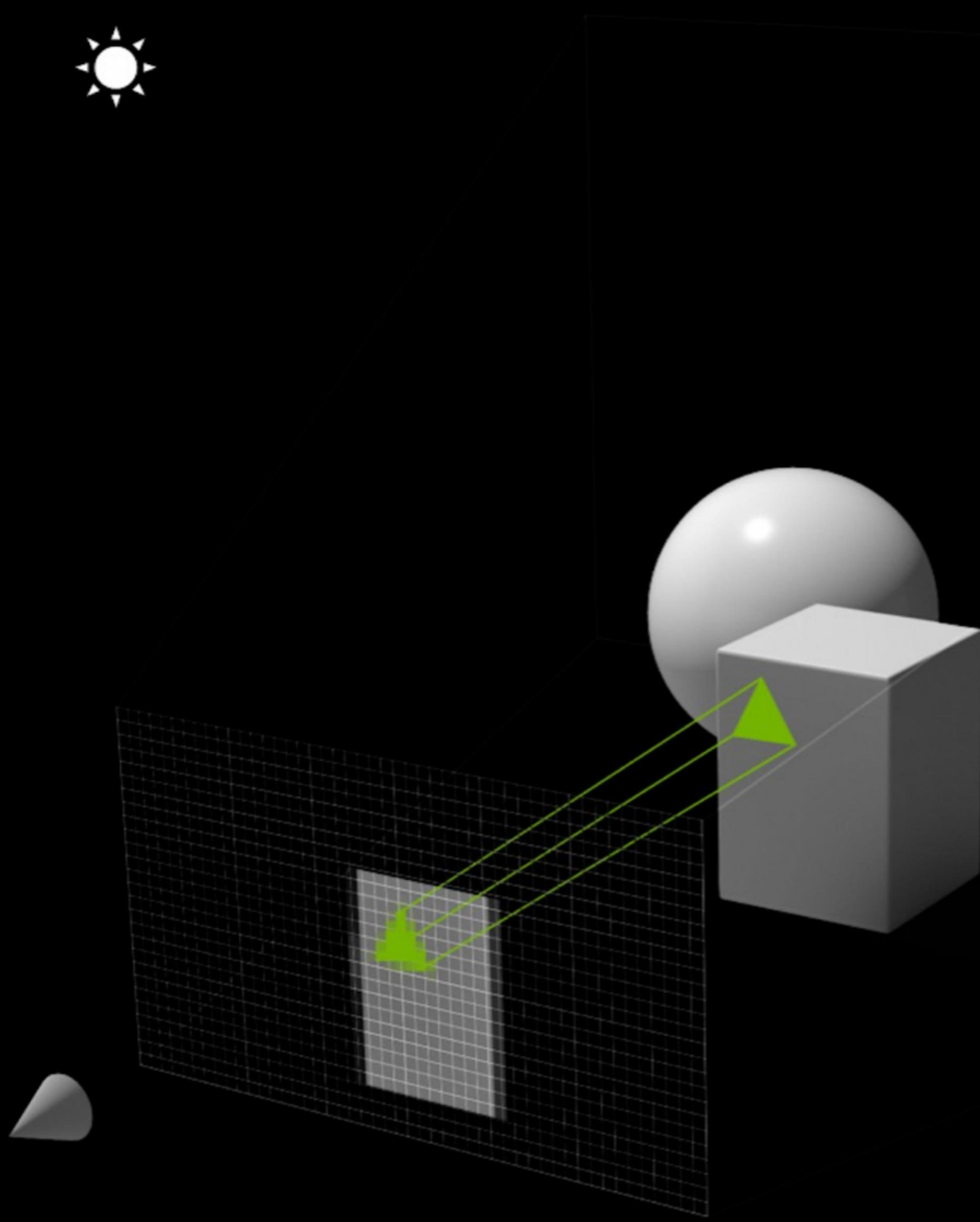
- simulation : photon parameters at PMT detectors
- rendering : pixel values at image plane
- both limited by ray geometry intersection, aka ray tracing

Many Applications of ray tracing :

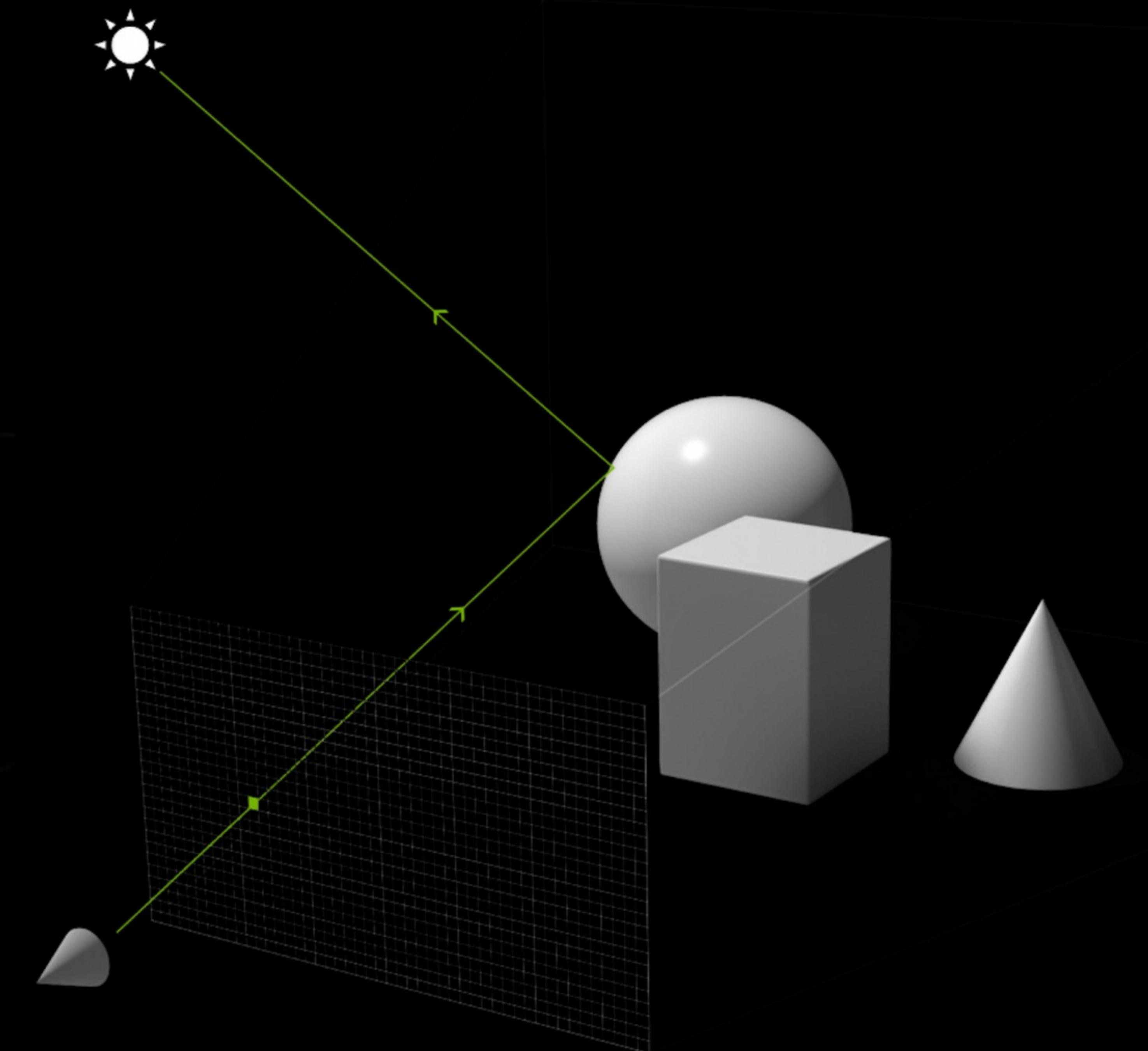
- advertising, design, architecture, films, games,...
- -> huge efforts to improve hw+sw over 30 yrs

Not a Photo, a Calculation





RASTERIZATION



RAY TRACING

SIGGRAPH 2018

ANNOUNCING QUAD

WORLD'S FIRST RAY TRACING GPU

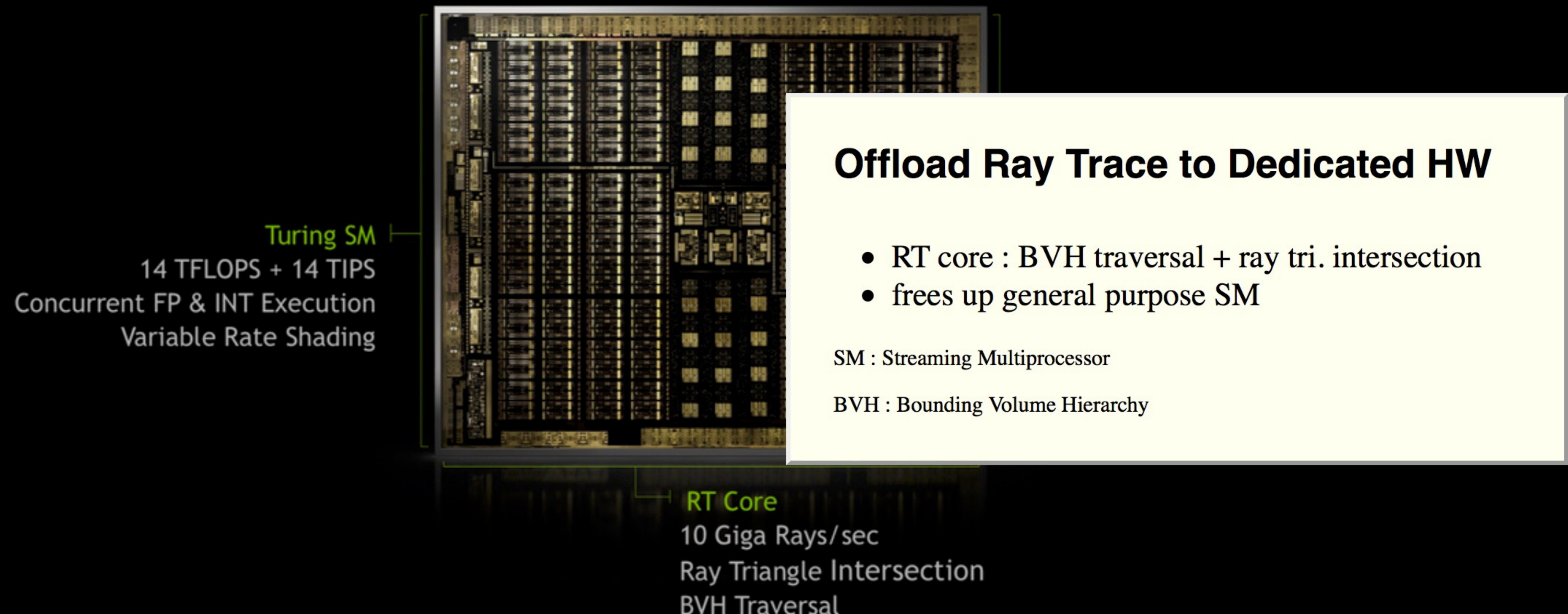


10 Giga Rays/s

Up to 100 TIPS
Up to 100 Giga Ops/sec
Up to 50% faster in NVLink

TURING BUILT FOR RTX

GREATEST LEAP SINCE 2006 CUDA GPU





RTX Platform : Hybrid Rendering

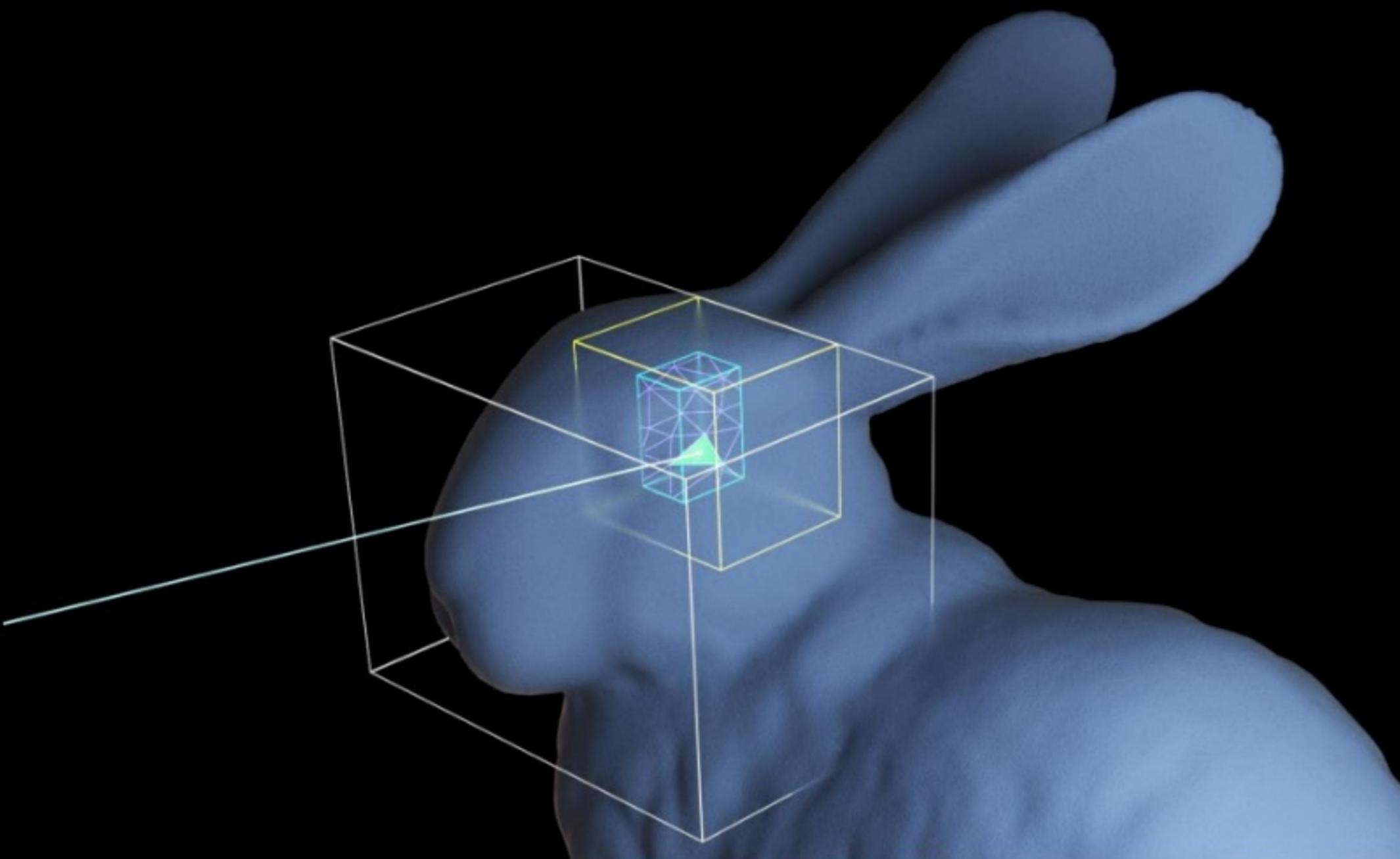
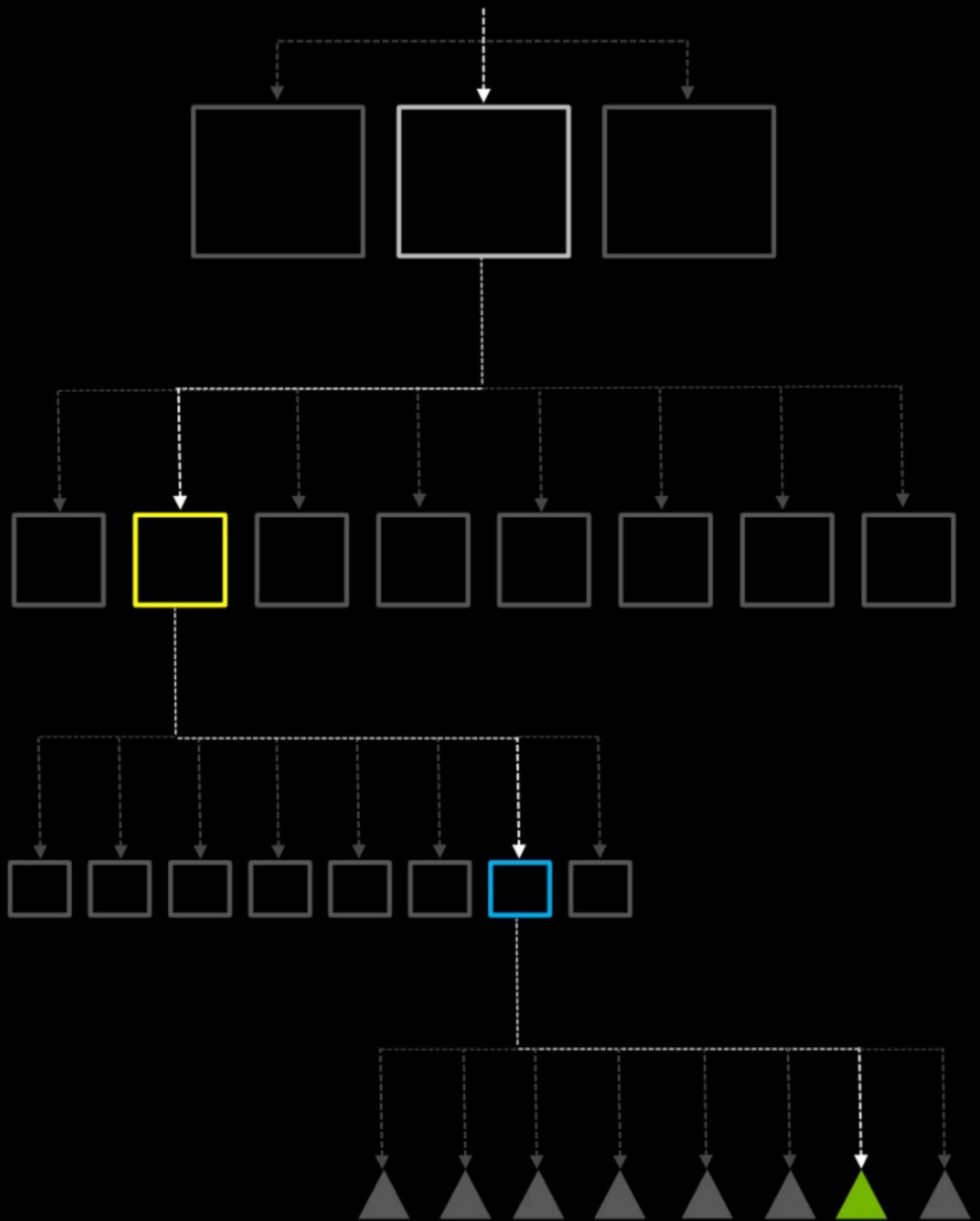
- Ray trace (RT cores)
- AI inference (Tensor cores) -> Denoising
- Rasterization (pipeline)
- Compute (SM, CUDA cores)

-> real-time photoreal cinematic 3D rendering

Spatial Index Acceleration Structure

BVH ALGORITHM

Massive Improvement in Search Efficiency



Tree of Bounding Boxes (bbox)

- aims to minimize bbox+primitive intersects
- accelerates ray-geometry intersection

NVIDIA® OptiX™ Ray Tracing Engine -- <http://developer.nvidia.com/optix>

OptiX makes GPU ray tracing accessible

- accelerates ray-geometry intersections
- simple : single-ray programming model
- "...free to use within any application..."
- access RT Cores[1] with OptiX 6.0.0+ via RTX™ mode

NVIDIA expertise:

- ~linear scaling up to 4 GPUs
- acceleration structure creation + traversal (Blue)
- instanced sharing of geometry + acceleration structures
- compiler optimized for GPU ray tracing

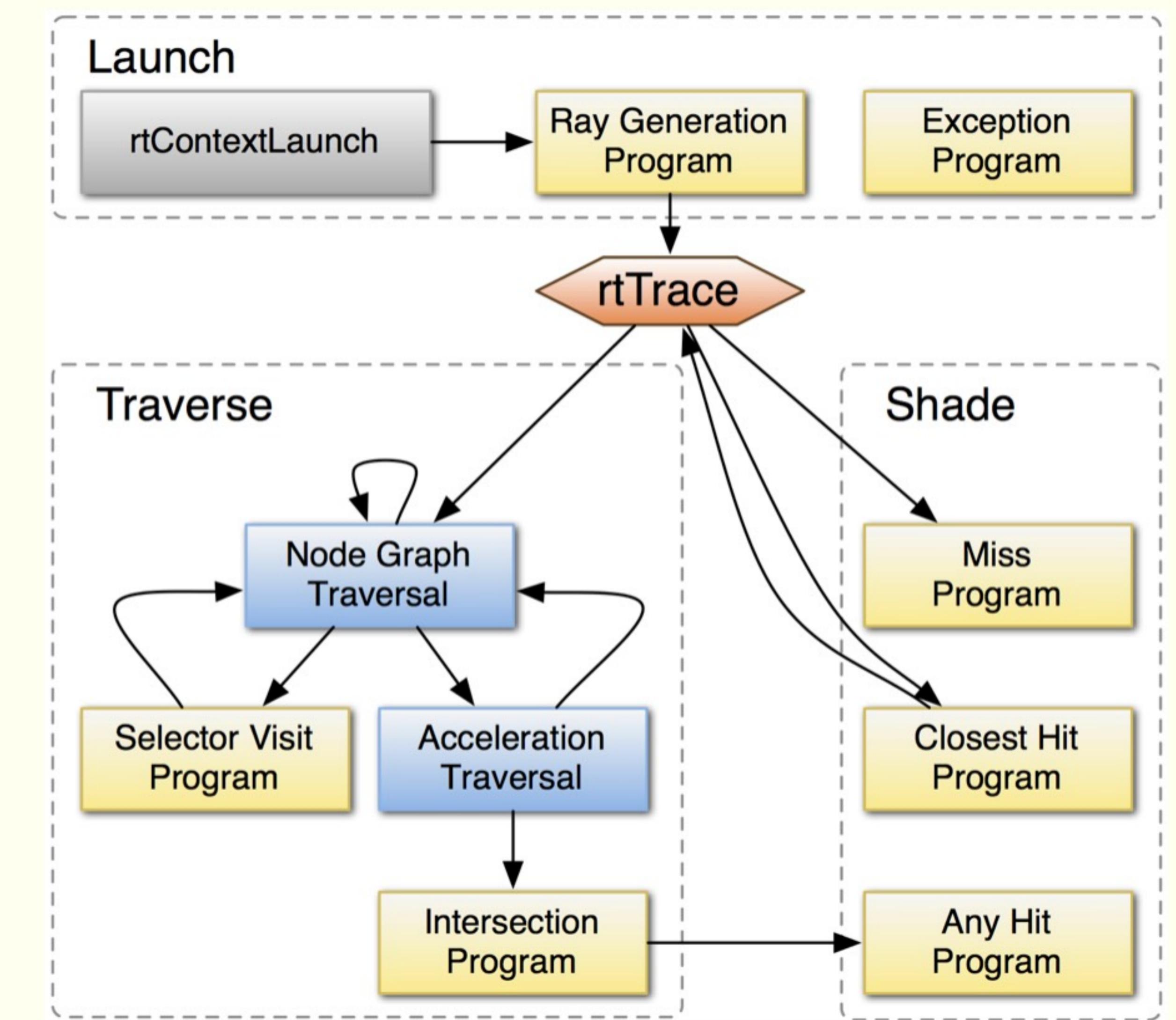
Opticks provides (Yellow):

- ray generation program
- ray geometry intersection+bbox programs

[1] Turing RTX GPUs

OptiX Raytracing Pipeline

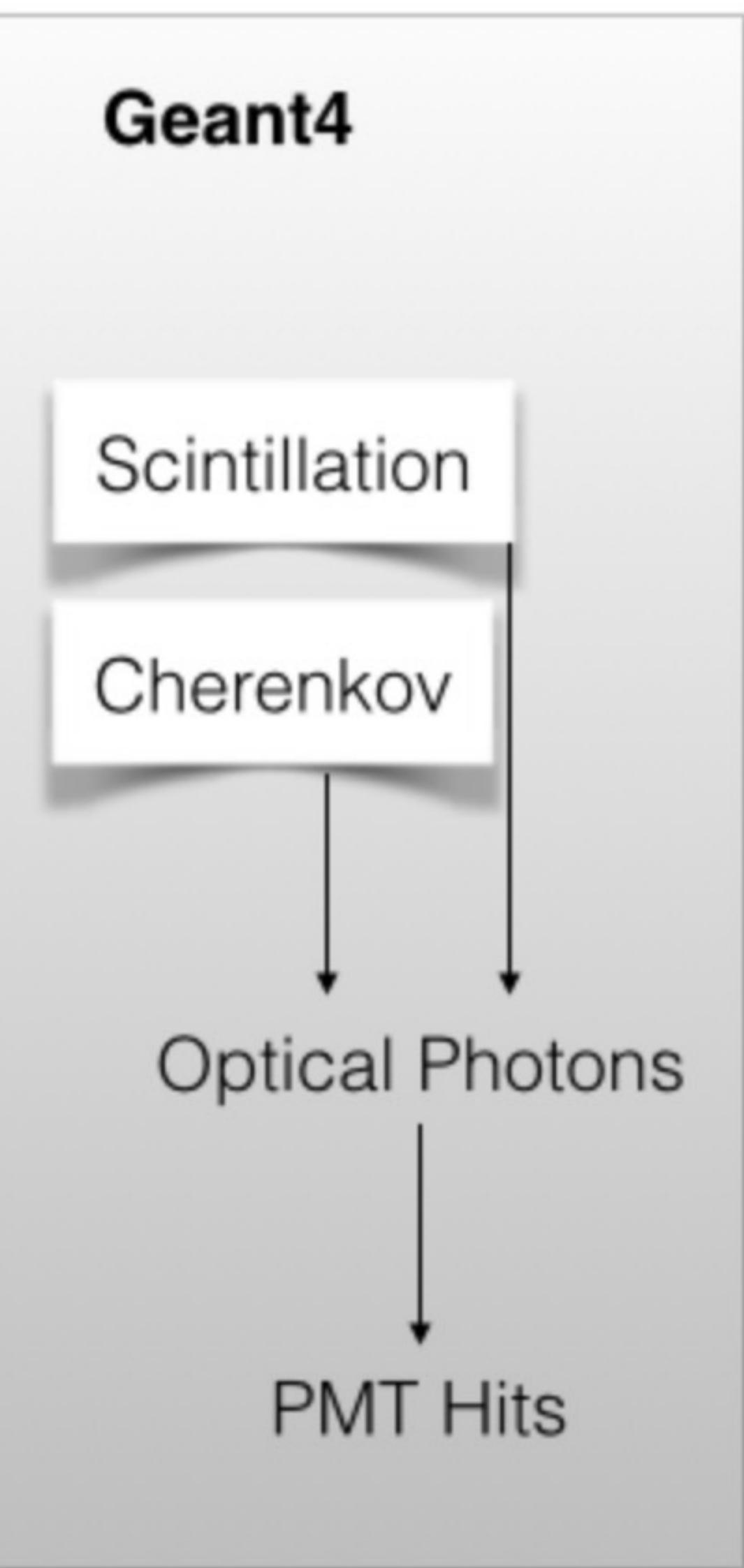
Analogous to OpenGL rasterization pipeline:



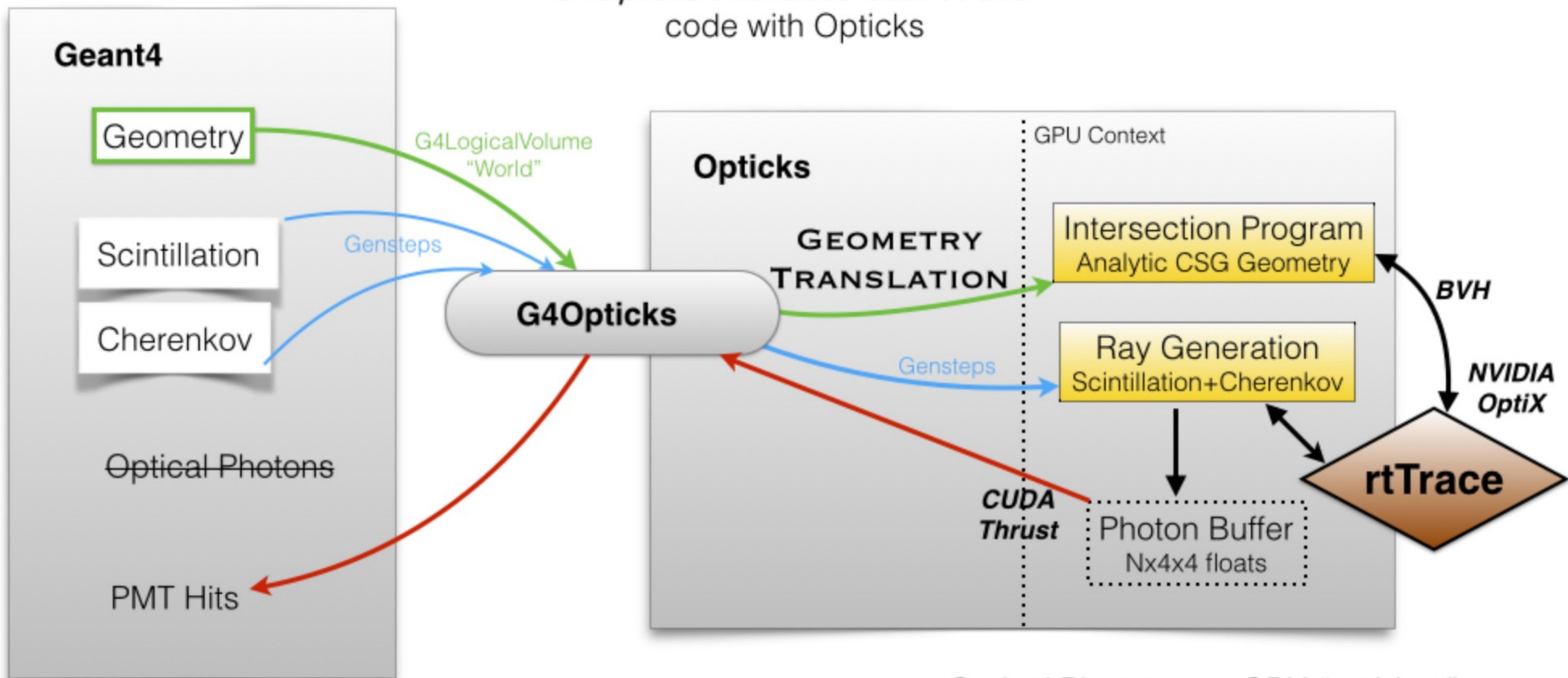
Geant4 + Opticks Hybrid Workflow : External Optical Photon Simulation

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

Standard Workflow



Hybrid Workflow



G4Opticks interfaces Geant4 user code with Opticks

Optical Photons are GPU “resident”,
only hits are copied to CPU memory

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

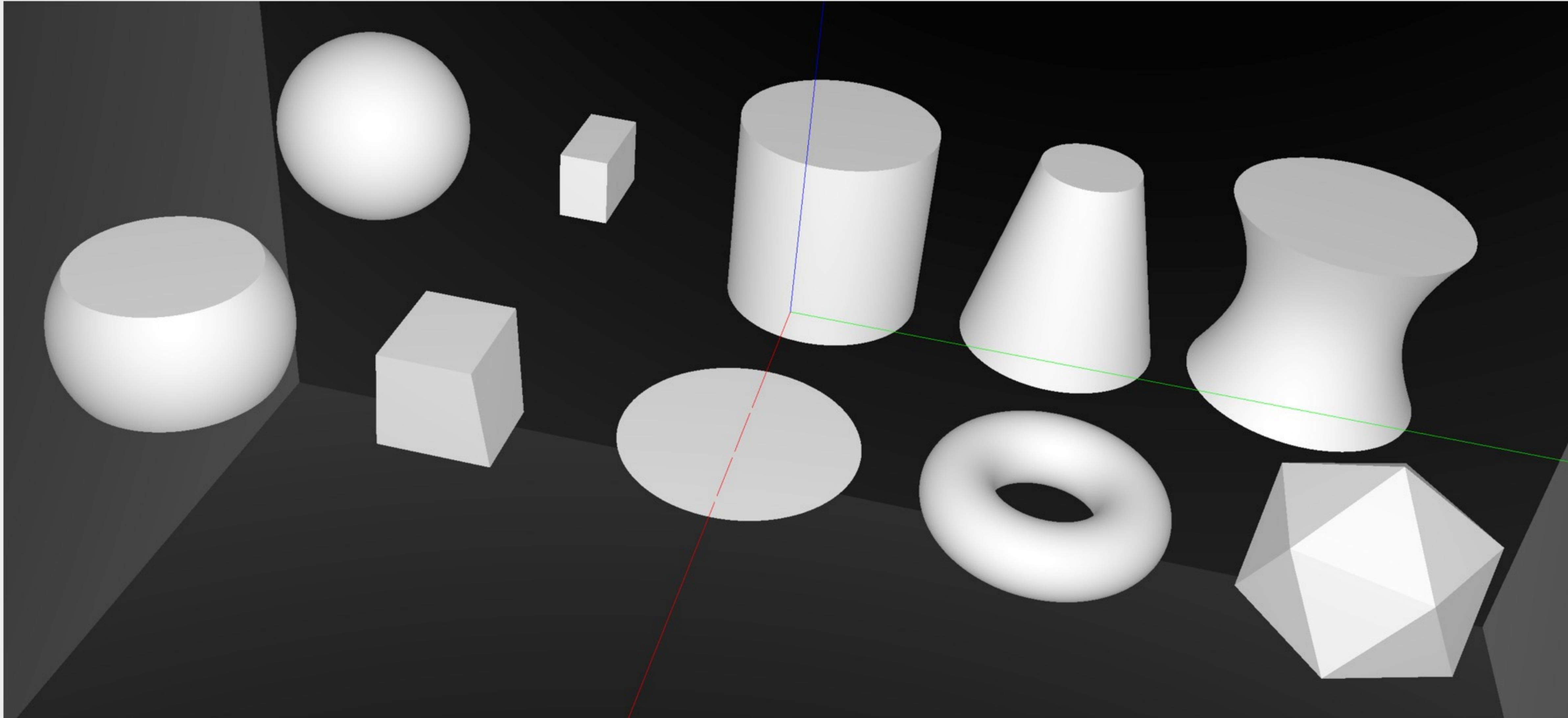
Thrust: high level C++ access to CUDA



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

G4Solid -> CUDA Intersect Functions for ~10 Primitives

- 3D parametric ray : $\text{ray}(x,y,z;t) = \text{rayOrigin} + t * \text{rayDirection}$
- 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**, ...*

G4Boolean -> CUDA/OptiX Intersection Program Implementing CSG

Complete Binary Tree, pick between pairs of nearest intersects:

<i>UNION tA < tB</i>	Enter B	Exit B	Miss B
Enter A	ReturnA	LoopA	ReturnA
Exit A	ReturnA	ReturnB	ReturnA
Miss A	ReturnB	ReturnB	ReturnMiss

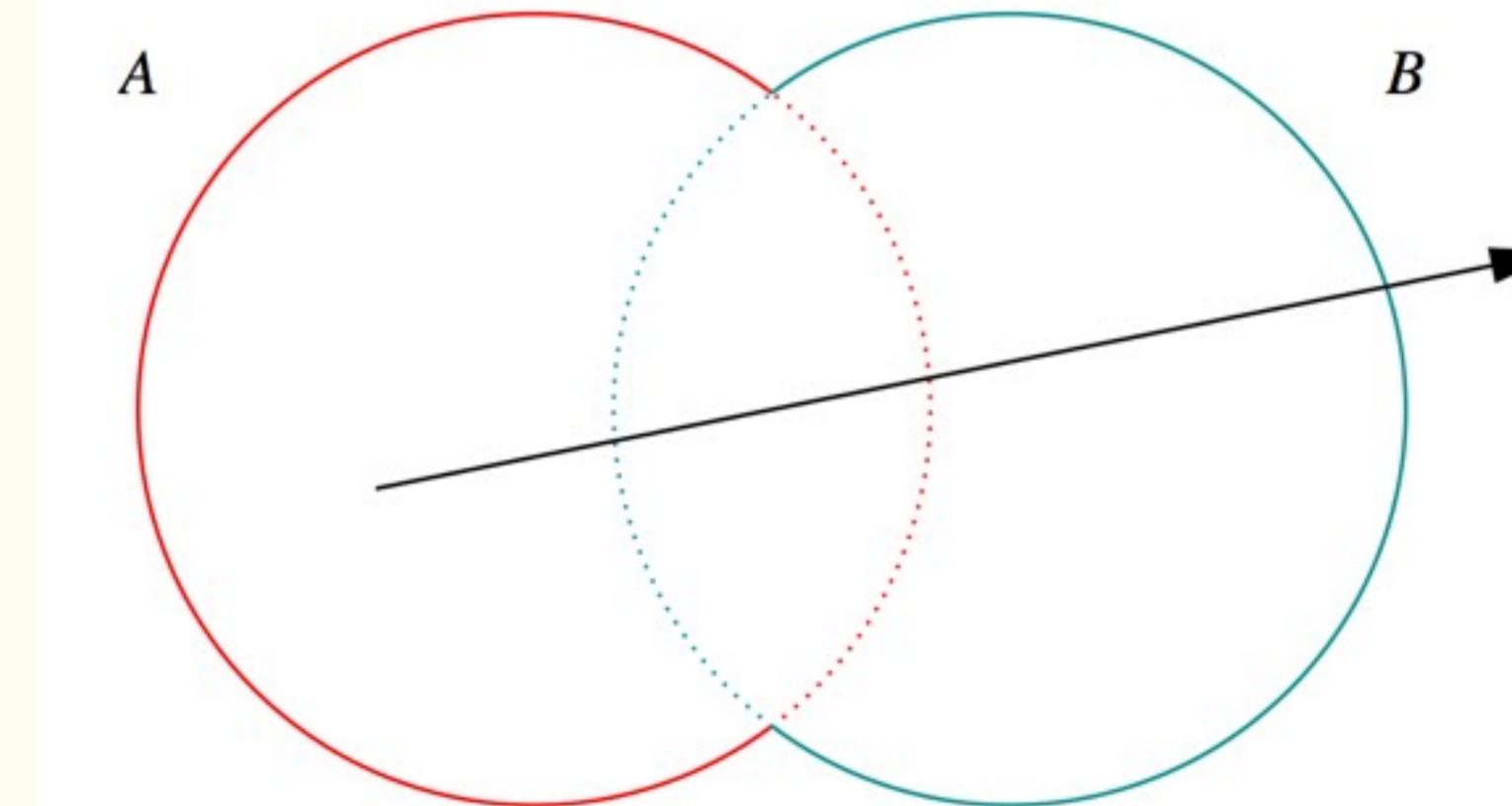
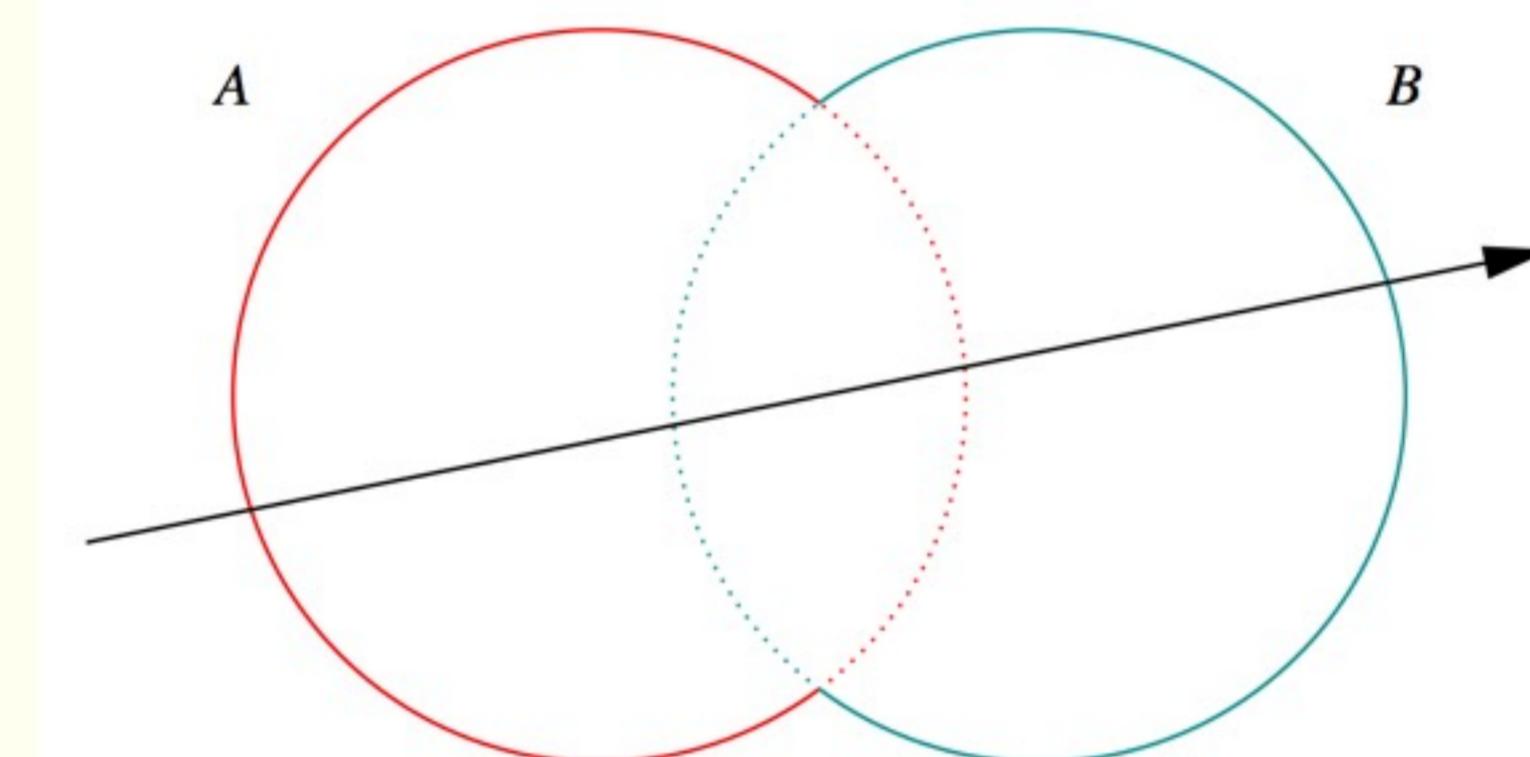
- *Nearest hit intersect algorithm* [1] avoids state
 - sometimes Loop : advance t_{min} , re-intersect both
 - classification shows if inside/outside
- *Evaluative* [2] implementation emulates recursion:
 - recursion not allowed in OptiX intersect programs
 - bit twiddle traversal of complete binary tree
 - stacks of postorder slices and intersects
- **Identical geometry to Geant4**
 - solving the same polynomials
 - near perfect intersection match

[1] Ray Tracing CSG Objects Using Single Hit Intersections, Andrew Kensler (2006)
with corrections by author of XRT Raytracer <http://xrt.wikidot.com/doc:csg> □

[2] https://bitbucket.org/simoncblyth/opticks/src/tip/optixrap/cu/csg_intersect_boolean.h □
Similar to binary expression tree evaluation using postorder traverse.

Outside/Inside Unions

dot(normal,rayDir) -> Enter/Exit



- **A + B** boundary not inside other
- **A * B** boundary inside other

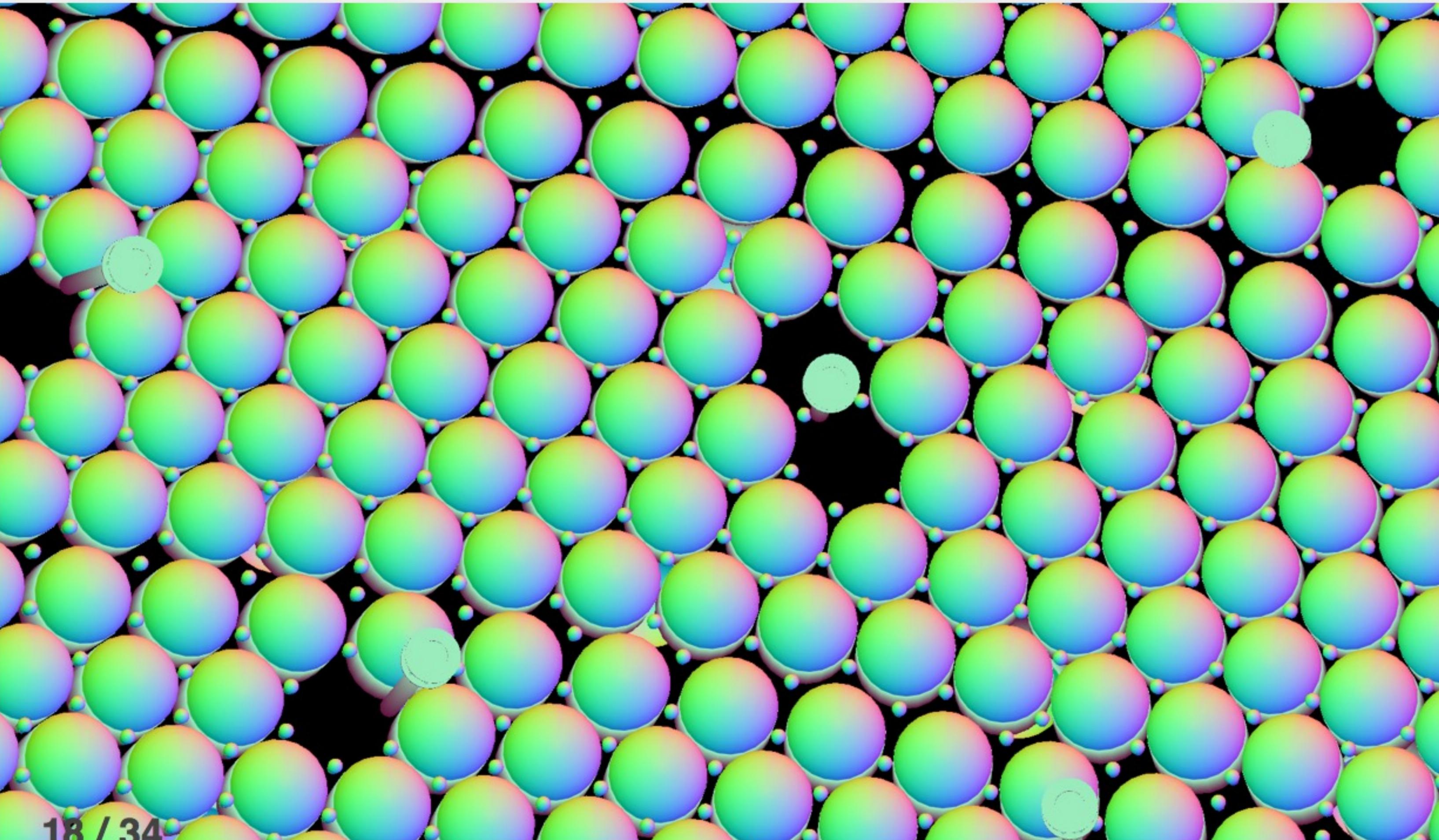
Opticks : Translates G4 Geometry to GPU, Without Approximation

G4 Structure Tree -> Instance+Global Arrays -> OptiX

Group structure into repeated instances + global remainder:

- auto-identify repeated geometry with "progeny digests"
 - JUNO : 5 distinct instances + 1 global
- instance transforms used in OptiX/OpenGL geometry

instancing -> huge memory savings for JUNO PMTs



Materials/Surfaces -> GPU Texture

Material/Surface/Scintillator properties

- interpolated to standard wavelength domain
- interleaved into "boundary" texture
- "reemission" texture for wavelength generation

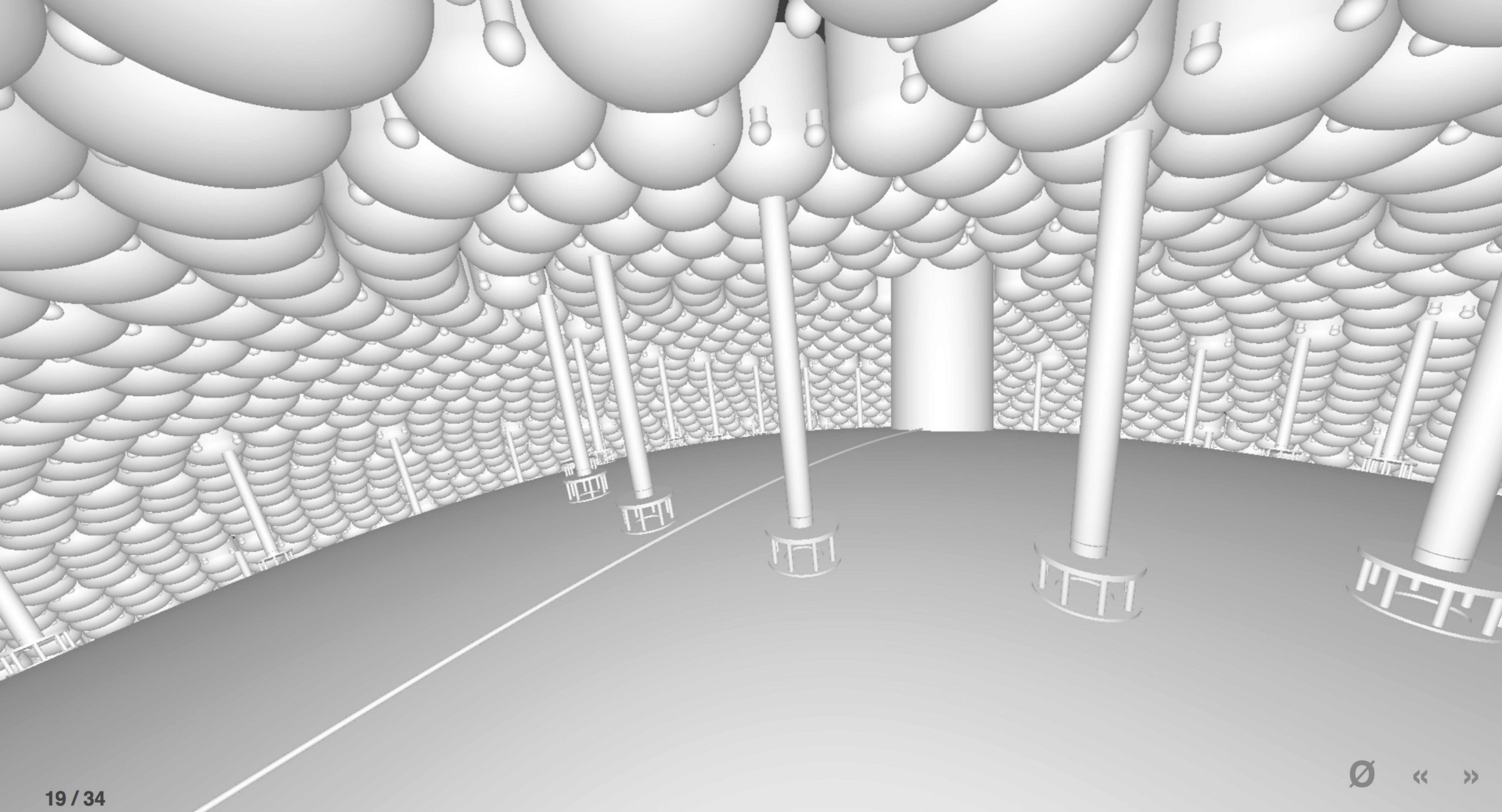
Material/surface boundary : 4 indices

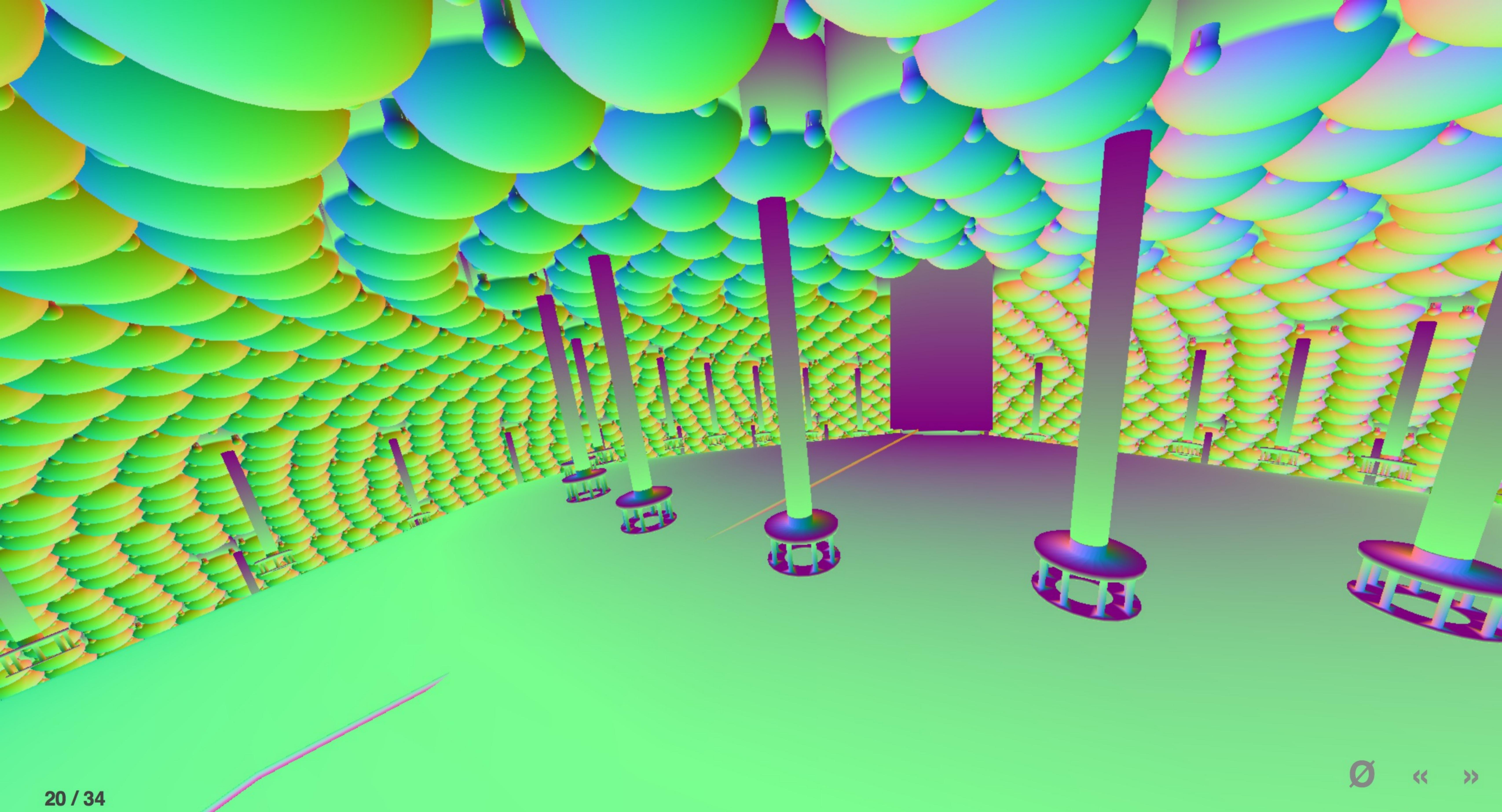
- outer material (parent)
- outer surface (inward photons, parent -> self)
- inner surface (outward photons, self -> parent)
- inner material (self)

Primitives labelled with unique boundary index

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

simple/fast properties + reemission wavelength





Validation of Opticks Simulation by Comparison with Geant4

Bi-simulations of all JUNO solids, with millions of photons

mis-aligned histories

mostly < 0.25%, < 0.50% for largest solids

deviant photons within matched history

< 0.05% (500/1M)

Primary sources of problems

- grazing incidence, edge skimmers
- incidence at constituent solid boundaries

Primary cause : float vs double

Geant4 uses *double* everywhere, *Opticks* only sparingly (observed *double* costing 10x slowdown with RTX)

Conclude

- neatly oriented photons more prone to issues than realistic ones
- perfect "technical" matching not feasible
- instead shift validation to more realistic full detector "calibration" situation

Random Aligned Bi-Simulation

Same inputs to *Opticks* and *Geant4*:

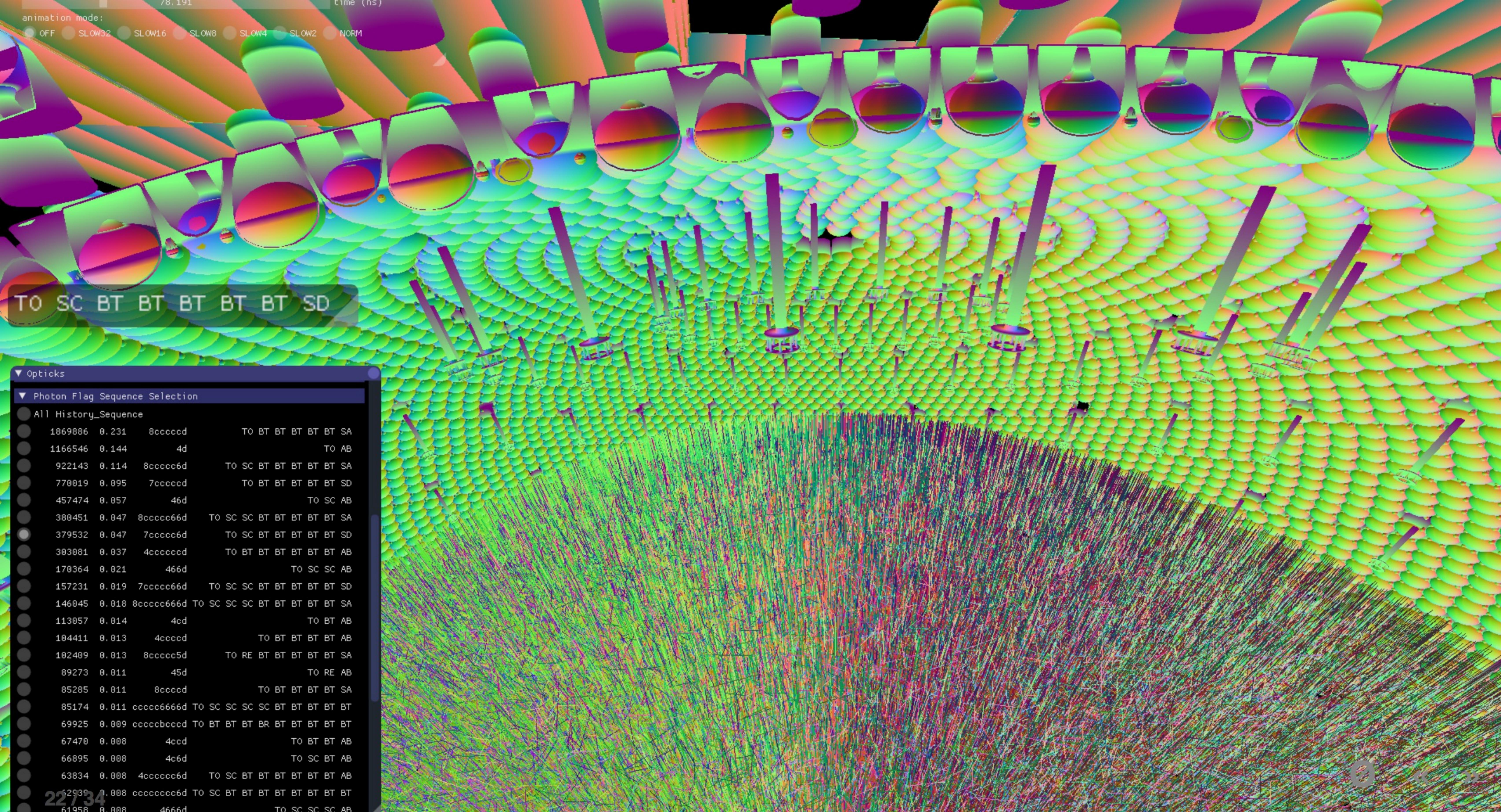
- CPU generated photons
- GPU generated randoms, fed to *Geant4*

Common recording into *OpticksEvents*:

- compressed photon step record, up to 16 steps
- persisted as *NumPy* arrays for python analysis

Aligned random consumption, direct comparison:

- ~every **scatter, absorb, reflect, transmit** at matched positions, times, polarization, waven



78.191
time (ns)

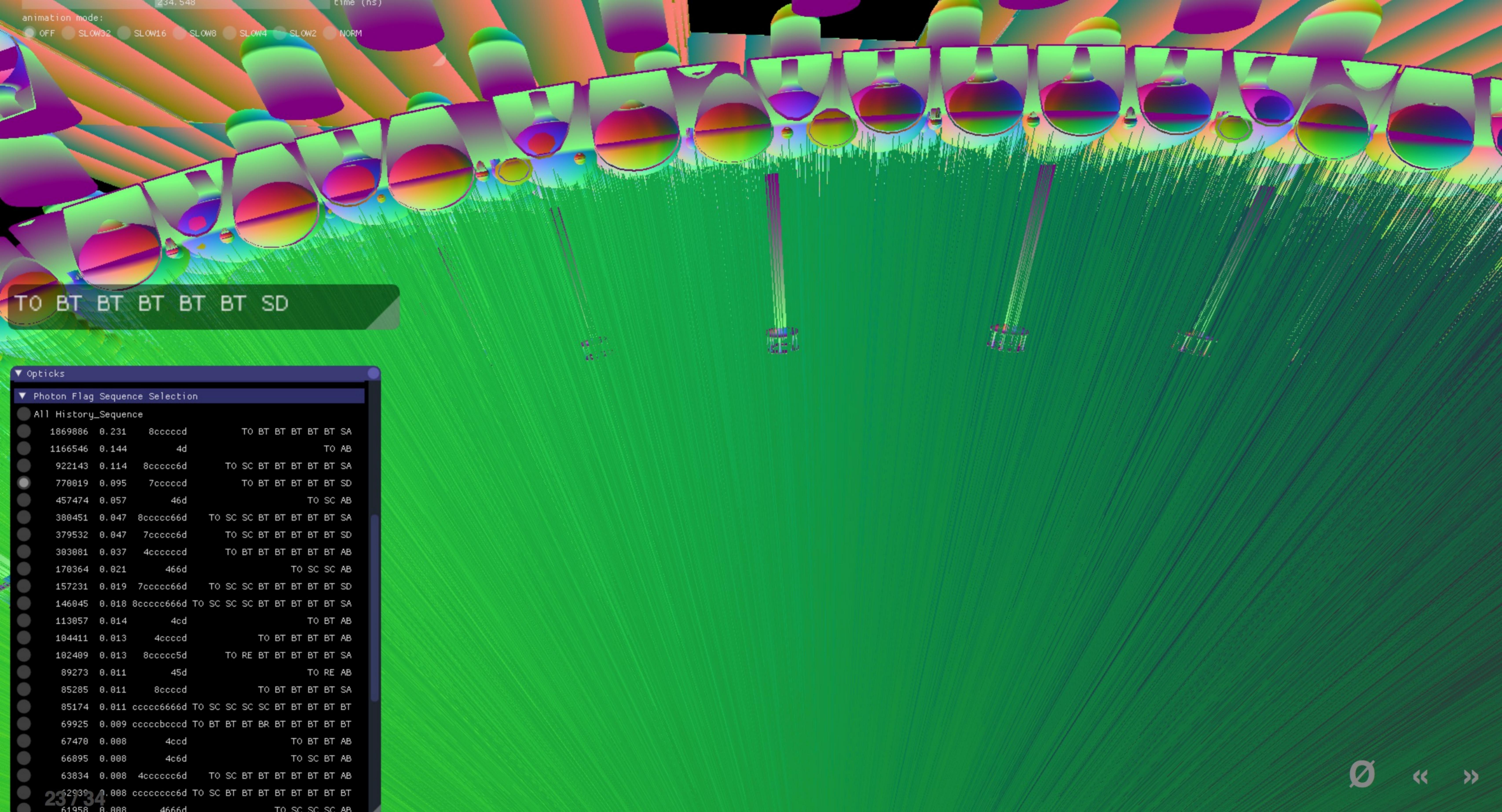
animation mode:
OFF SLOW32 SLOW16 SLOW8 SLOW4 SLOW2 NORM

TO SC BT BT BT BT BT SD

▼ Opticks

▼ Photon Flag Sequence Selection

All History_Sequence			
1869886	0.231	8cccccd	TO BT BT BT BT BT SA
1166546	0.144	4d	TO AB
922143	0.114	8cccccc6d	TO SC BT BT BT BT BT SA
770019	0.095	7cccccd	TO BT BT BT BT BT SD
457474	0.057	46d	TO SC AB
380451	0.047	8cccccc66d	TO SC SC BT BT BT BT BT SA
379532	0.047	7cccccc6d	TO SC BT BT BT BT BT SD
303081	0.037	4cccccccd	TO BT BT BT BT BT AB
170364	0.021	466d	TO SC SC AB
157231	0.019	7cccccc66d	TO SC SC BT BT BT BT BT SD
146045	0.018	8cccccc666d	TO SC SC SC BT BT BT BT BT SA
113057	0.014	4cd	TO BT AB
104411	0.013	4ccccd	TO BT BT BT BT AB
102409	0.013	8cccccc5d	TO RE BT BT BT BT SA
89273	0.011	45d	TO RE AB
85285	0.011	8ccccd	TO BT BT BT BT SA
85174	0.011	cccccc6666d	TO SC SC SC SC BT BT BT BT BT
69925	0.009	cccccbccc	TO BT BT BT BR BT BT BT BT BT
67470	0.008	4ccd	TO BT BT AB
66895	0.008	4c6d	TO SC BT AB
63834	0.008	4cccccc6d	TO SC BT BT BT BT BT AB
62930	0.008	cccccccc6d	TO SC BT BT BT BT BT BT BT BT
61958	0.008	4666d	TO SC SC SC AB



Performance : Scanning from 1M to 400M Photons

Full JUNO Analytic Geometry j1808v5

- "calibration source" genstep at center of scintillator

Production Mode : does the minimum

- only saves hits
- skips : genstep, photon, source, record, sequence, index, ..
- no *Geant4* propagation (other than at 1M for extrapolation)

Multi-Event Running, Measure:

interval

avg time between successive launches, including overheads:
(upload gensteps + **launch** + download hits)

launch

avg of 10 OptiX launches

- overheads < 10% beyond 20M photons

Test Hardware + Software

Workstation

- DELL Precision 7920T Workstation
- Intel Xeon Gold 5118, 2.3GHz, 48 cores, 62G
- NVIDIA Quadro RTX 8000 (48G)

Software

- Opticks 0.0.0 Alpha
- Geant4 10.4p2
- NVIDIA OptiX 6.5.0
- NVIDIA Driver 435.21
- CUDA 10.1

IHEP GPU Cluster

- 10 nodes of 8x NVIDIA Tesla GV100 (32G)

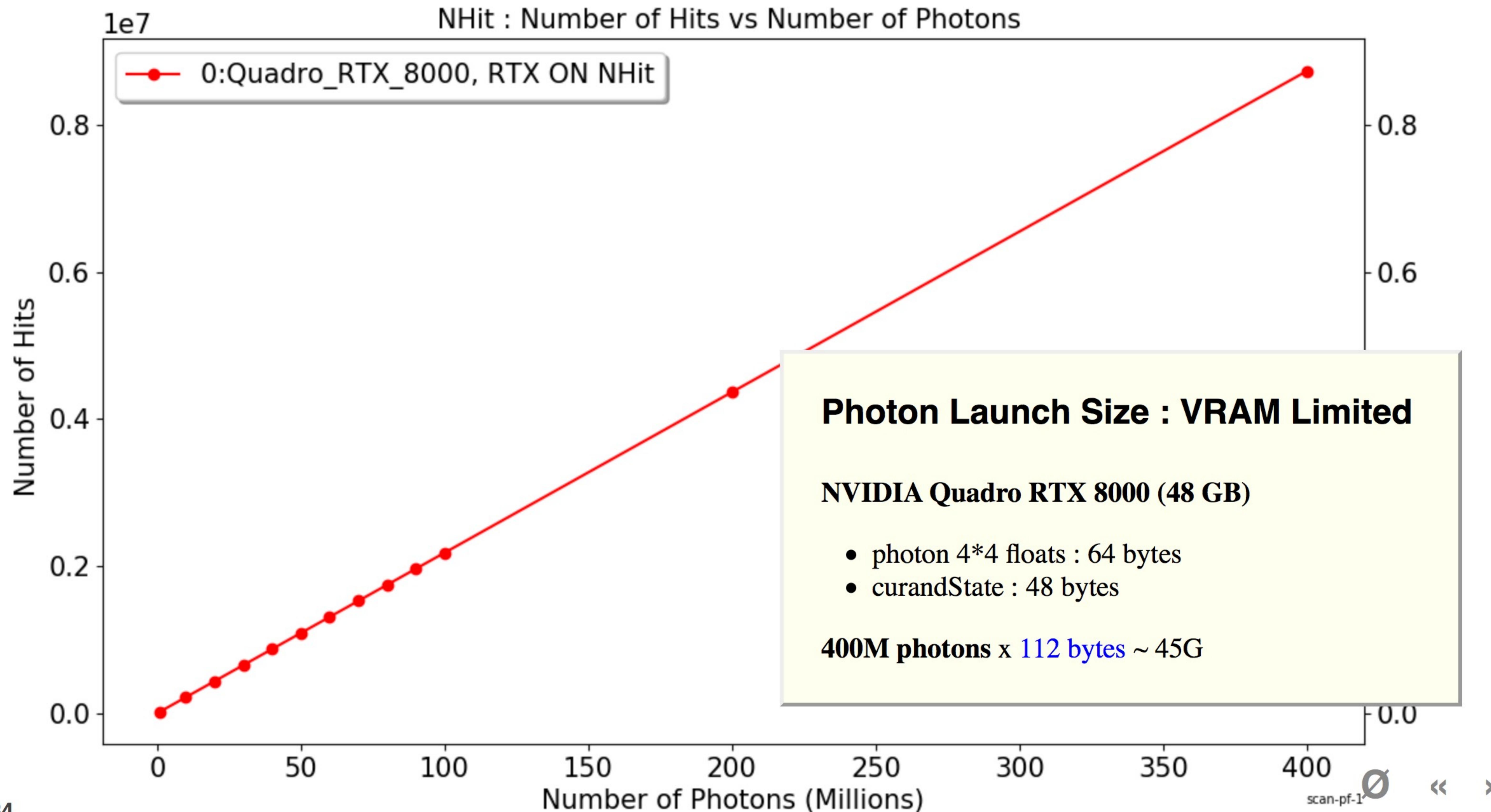
NVIDIA Quadro RTX 8000 (48G)



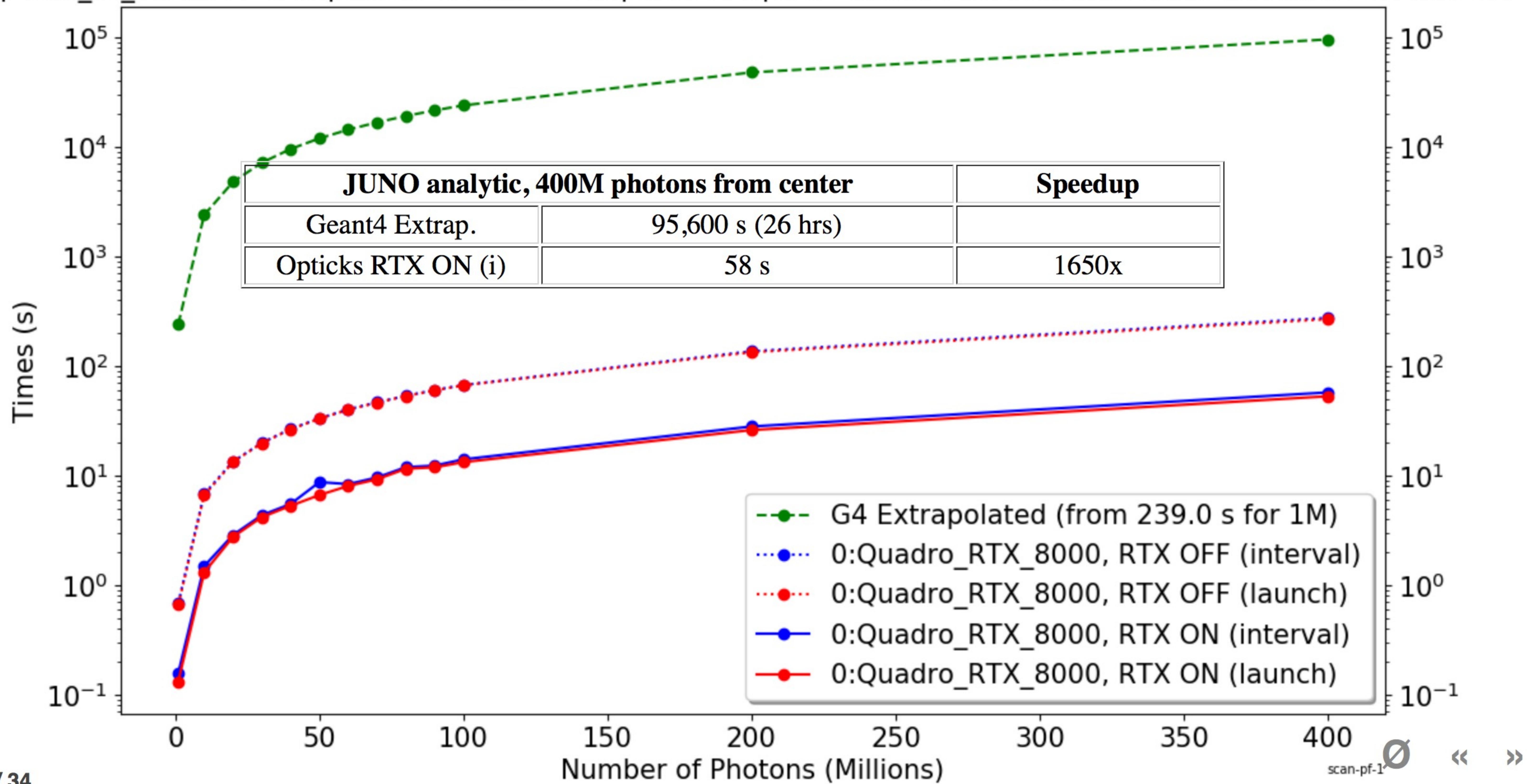
RTX

谢谢 NVIDIA China
for loaning the card

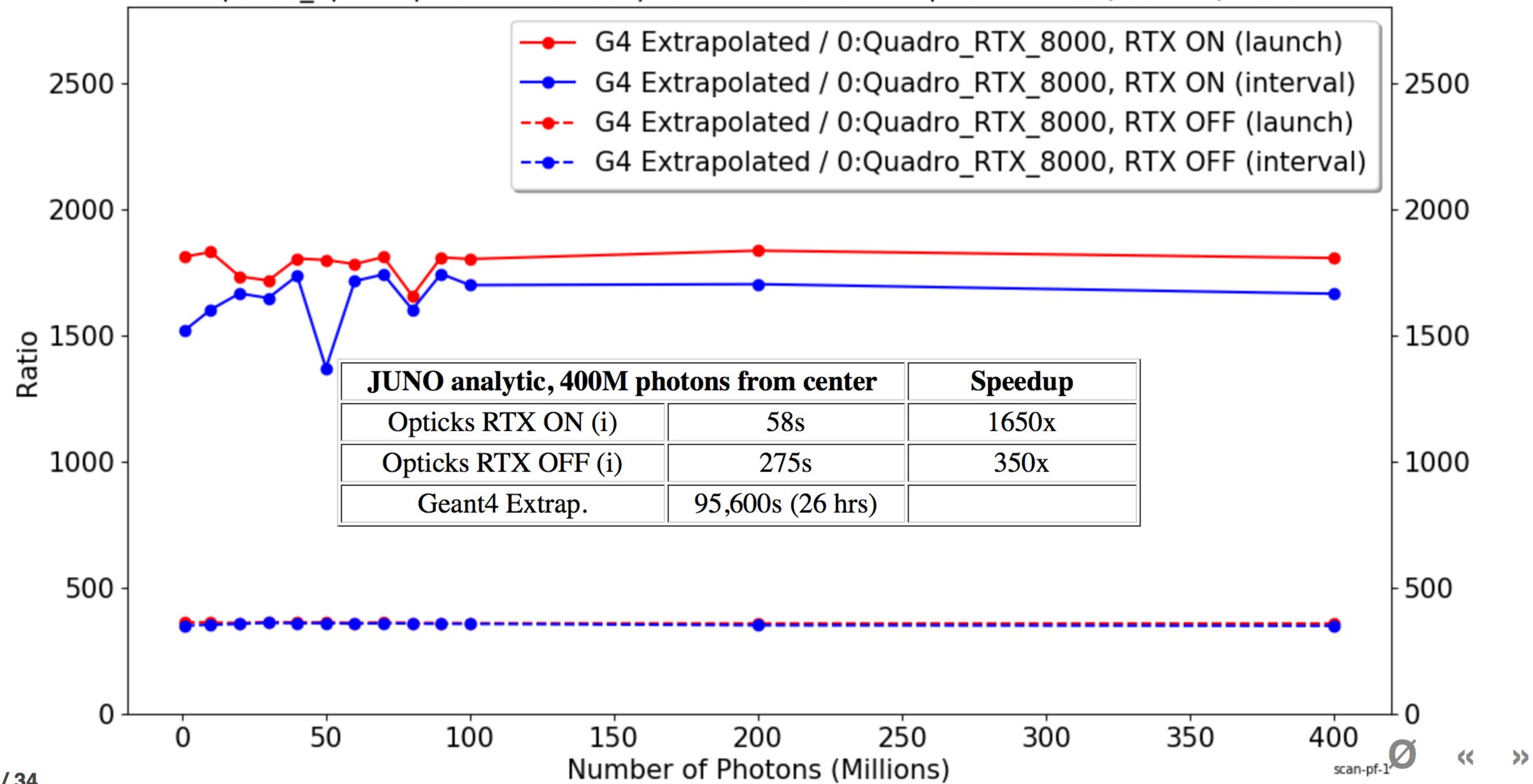


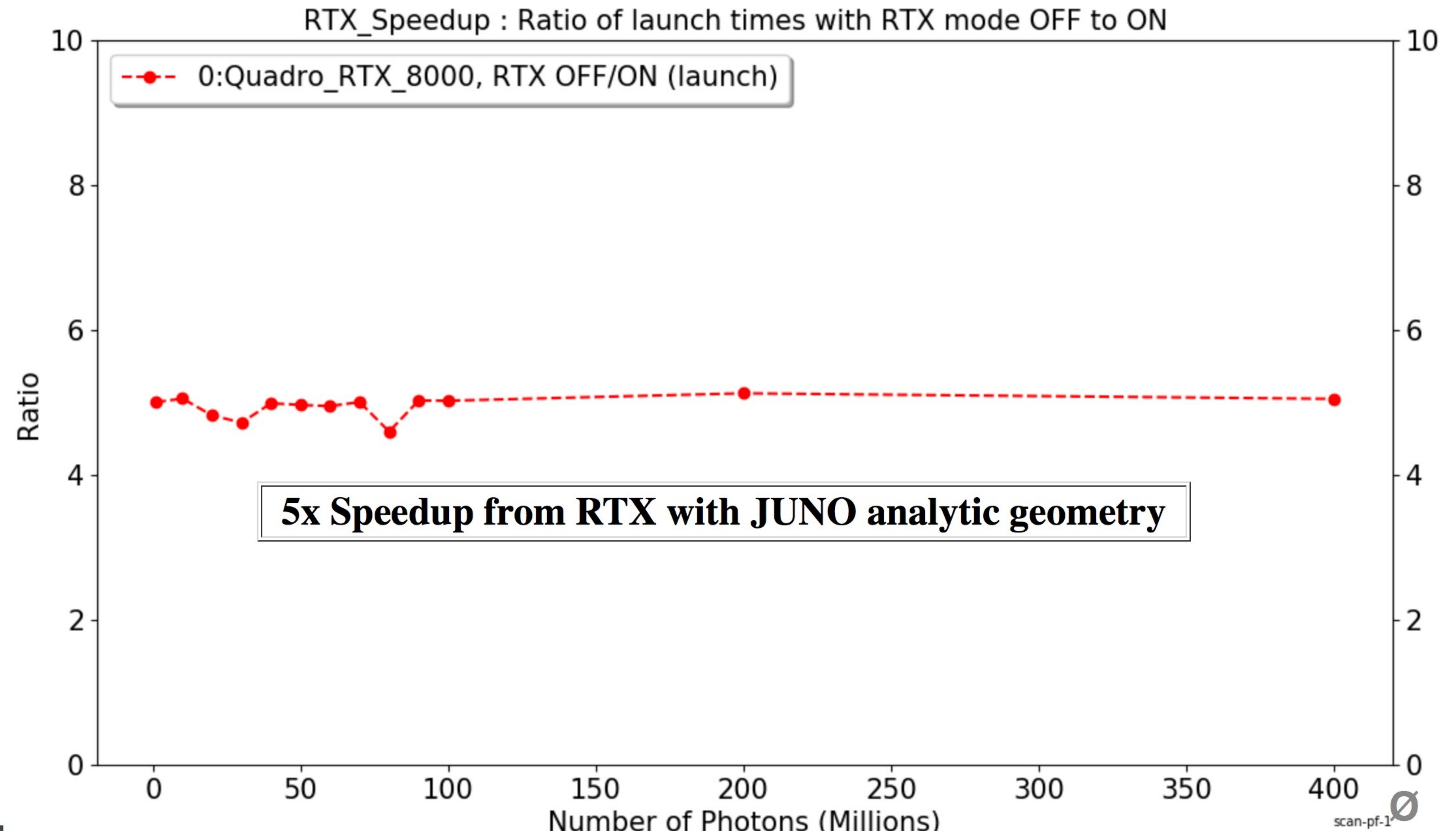


Opticks_vs_Geant4 : Extrapolated G4 times compared to Opticks launch+interval times with RTX mode ON and OFF



Opticks_Speedup : Ratio of extrapolated G4 times to Opticks launch(interval) times

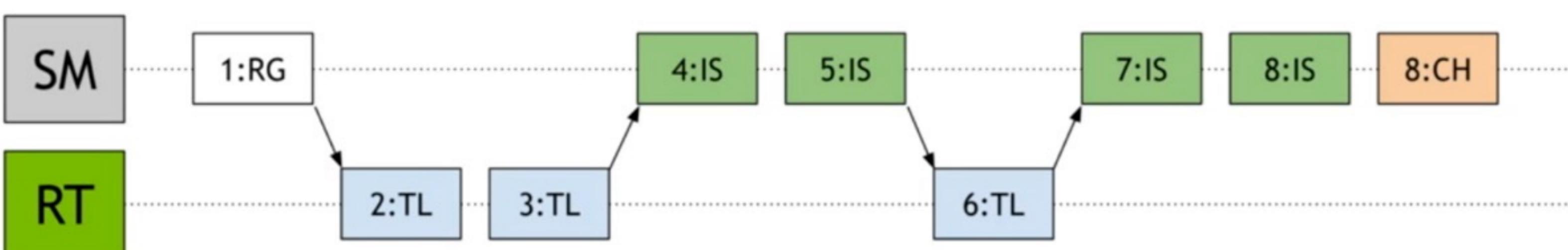
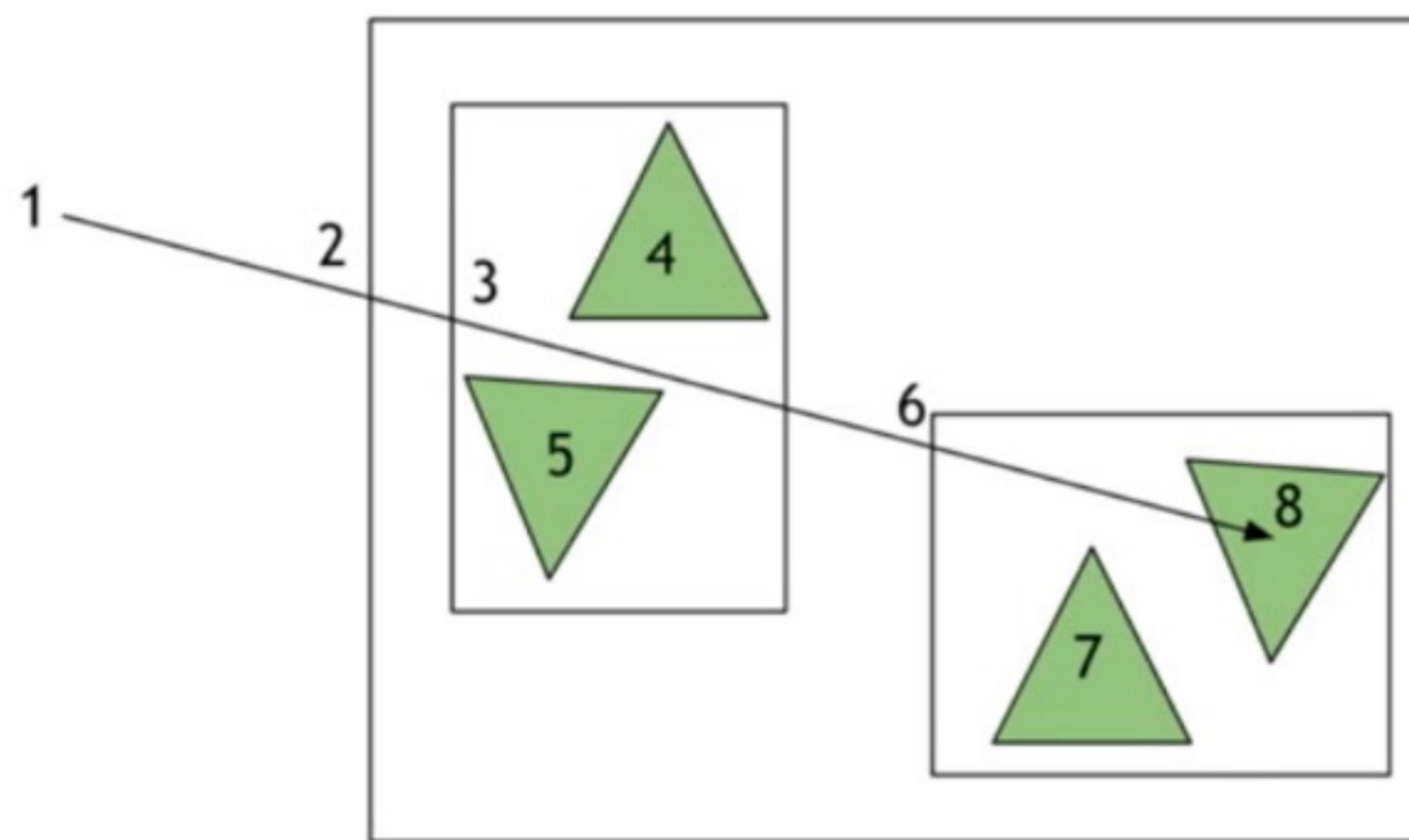




Useful Speedup > 1000x : But Why Not Giga Rays/s ? (1 Photon ~10 Rays)

- NVIDIA claim : 10 Giga Rays/s with RT Core
- -> 1 Billion photons per second
- RT cores : built-in triangle intersect + 1-level of instancing
- flatten scene model to avoid SM<->RT roundtrips ?

RTX traversal: custom primitives



*NB: conceptual model of execution, not timing.

OptiX Performance Tools and Tricks, David Hart, NVIDIA
<https://developer.nvidia.com/siggraph/2019/video/sig915-vid> □

100M photon RTX times, avg of 10

Launch times for various geometries				
Geometry	Launch (s)	Giga Rays/s	Relative to ana	
JUNO ana	13.2	0.07		
JUNO tri.sw	6.9	0.14	1.9x	
JUNO tri.hw	2.2	0.45	6.0x	
Boxtest ana	0.59	1.7		
Boxtest tri.sw	0.62	1.6		
Boxtest tri.hw	0.30	3.3	1.9x	

- ana : Opticks analytic CSG (SM)
- tri.sw : software triangle intersect (SM)
- tri.hw : hardware triangle intersect (RT)

JUNO 15k triangles, 132M without instancing

Simple Boxtest geometry gets into ballpark



Where Next for Opticks ?

JUNO+Opticks into Production

- optimize geometry modelling for RTX
- full JUNO geometry validation iteration
- JUNO offline integration
- optimize GPU cluster throughput:
 - split/join events to fit VRAM
 - job/node/multi-GPU strategy
- support OptiX 7, find multi-GPU load balancing approach

Geant4+Opticks Integration : Work with Geant4 Collaboration

- finalize *Geant4+Opticks* extended example
 - aiming for *Geant4* distrib
- prototype *Genstep* interface inside *Geant4*
 - avoid customizing *G4Cerenkov G4Scintillation*

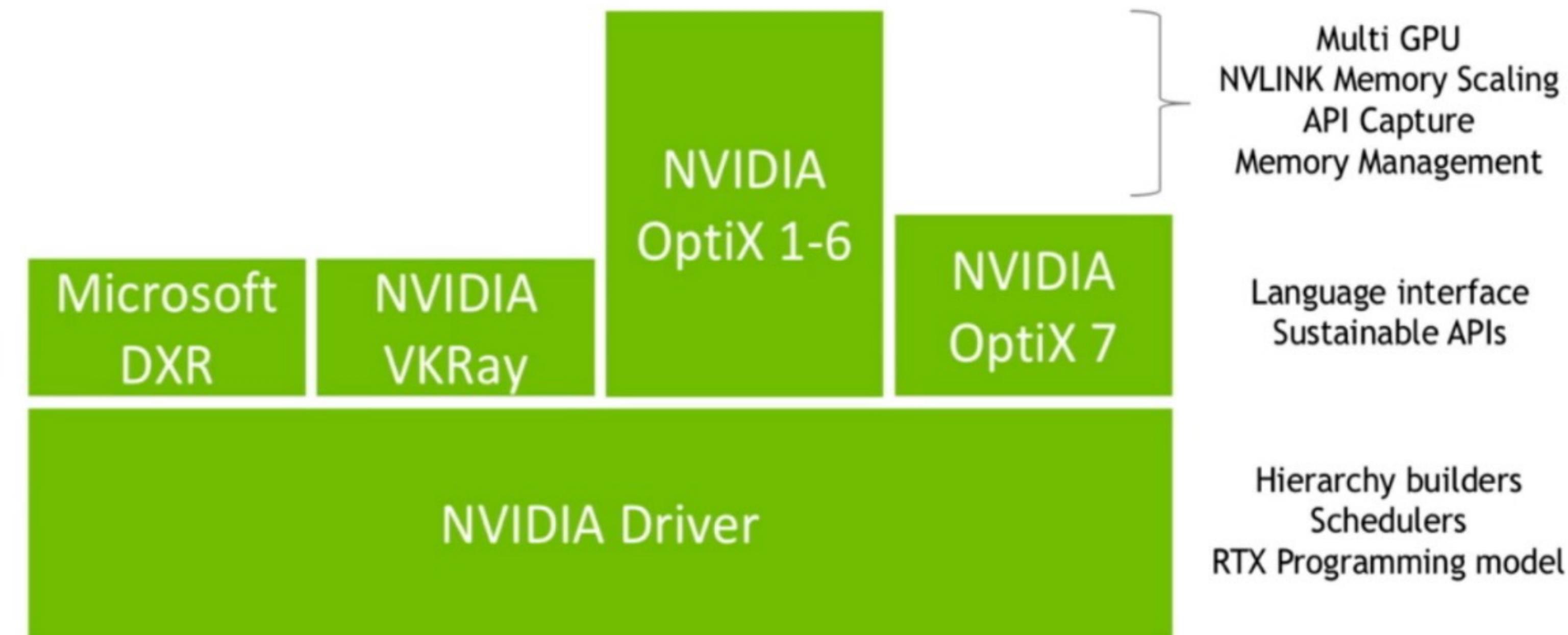
Alpha Development ----->-----> Robust Tool

- many more users+developers required (current ~10+1)
- if you have an optical photon simulation problem ...
 - start by joining : <https://groups.io/g/opticks> □

NVIDIA OptiX 7 : Entirely new API

- introduced August 2019
- low-level CUDA-centric thin API
- ~~near perfect scaling to 4 GPUs, for free~~

Introducing OptiX 7



Dramatically Improved Optical Photon Simulation Performance...

Three revolutions reinforcing each other:

- games -> graphics revolution -> GPU -> cheap TFLOPS
- internet scale big datasets -> ML revolution
- computer vision revolution for autonomous vehicles

Deep rivers of development, ripe for re-purposing

- analogous problems -> solutions
- **experience across fields essential to find+act on analogies**

Example : DL denoising for faster ray trace convergence

- analogous to hit aggregation
- skip the hits, jump straight to DL smoothed probabilities
 - **blurs the line between simulation and reconstruction**

Re-evaluate long held practices in light of new realities:

- large ROOT format (C++ object) MC samples repeatedly converted+uploaded to GPU for DL training ... OR:
- small Genstep NumPy arrays uploaded, dynamically simulated into GPU hit arrays in fractions of a second

How is >1000x possible ?

Progress over 30 yrs, Billions of Dollars

- industry funded : game, film, design, ...
- re-purposed by translating geometry to GPU
 - tree of C++ objects -> arrays -> BVH

Photon Simulation ideally suited to GPU

- millions of photons -> abundantly parallel
- simple phys. -> small stack -> many in flight
- decoupled -> no synchronization

Dynamically generated simulation feasible ?

- current reconstruction -> custom simulation
- no more : limited MC stats in edge cases

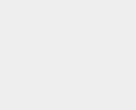
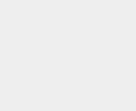
Meeting the Challenge of JUNO Simulation with Opticks

Opticks : state-of-the-art GPU ray tracing applied to optical photon simulation and integrated with *Geant4*, giving a leap in performance that eliminates memory and time bottlenecks.

Highlights 2019

- Benefit from hardware accelerated ray tracing
- **Opticks > 1000x Geant4** (one Turing GPU)

- Drastic speedup -> better detector understanding -> greater precision
 - **any simulation limited by optical photons can benefit**
 - more photon limited -> more overall speedup (99% -> 100x)

https://bitbucket.org/simoncblyth/opticks 	code repository
https://simoncblyth.bitbucket.io 	presentations and videos
https://groups.io/g/opticks 	forum/mailing list archive
email: opticks+subscribe@groups.io	subscribe to mailing list

JUNO Presentations at CHEP 2019

JUNO Simulation Software (talk)

Ziyan Deng, IHEP

Design and Performance of the JUNO Conditions Database (talk)

Xingtao Huang, SDU

DIRAC-based solutions for JUNO production system (talk)

Xiaomei Zhang, IHEP

China-EU scientific cooperations on JUNO distributed Computing (talk)

Giuseppe Andronico, INFN Catania

JUNO Calibration Simulation (poster)

Qingmin Zhang, IHEP

GPU applications in JUNO reconstruction (talk)

Wuming Luo, IHEP

A first step of event reconstruction in JUNO (talk)

Miao He, IHEP

Machine Learning Approaches for Event Reconstruction in JUNO (talk)

Yury Malyshkin, INFN Roma 3

Jupyter-based service for JUNO analysis (poster)

Weidong Li, IHEP