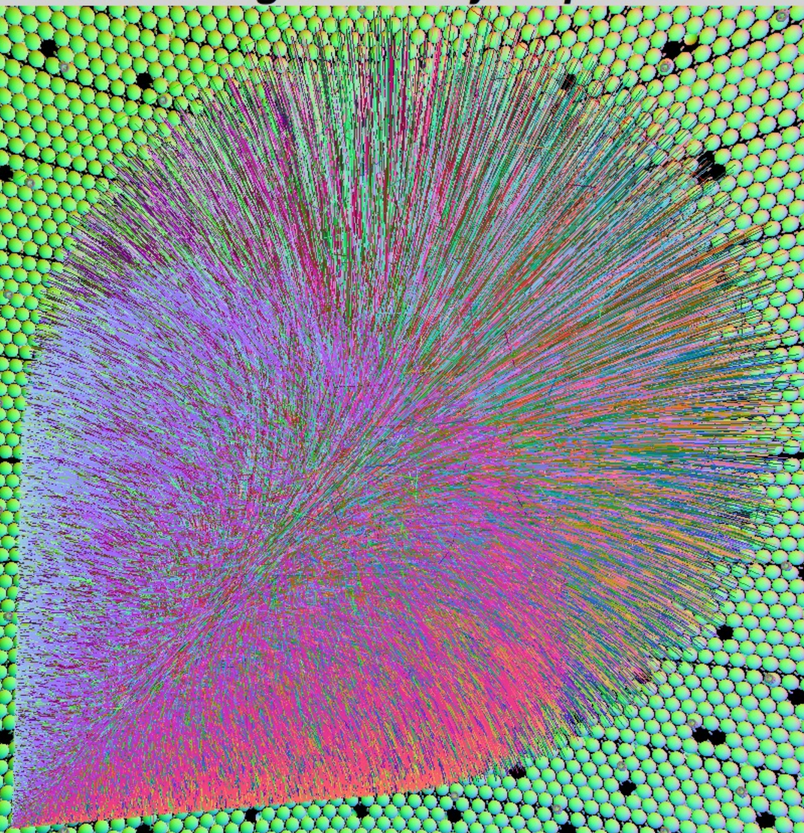
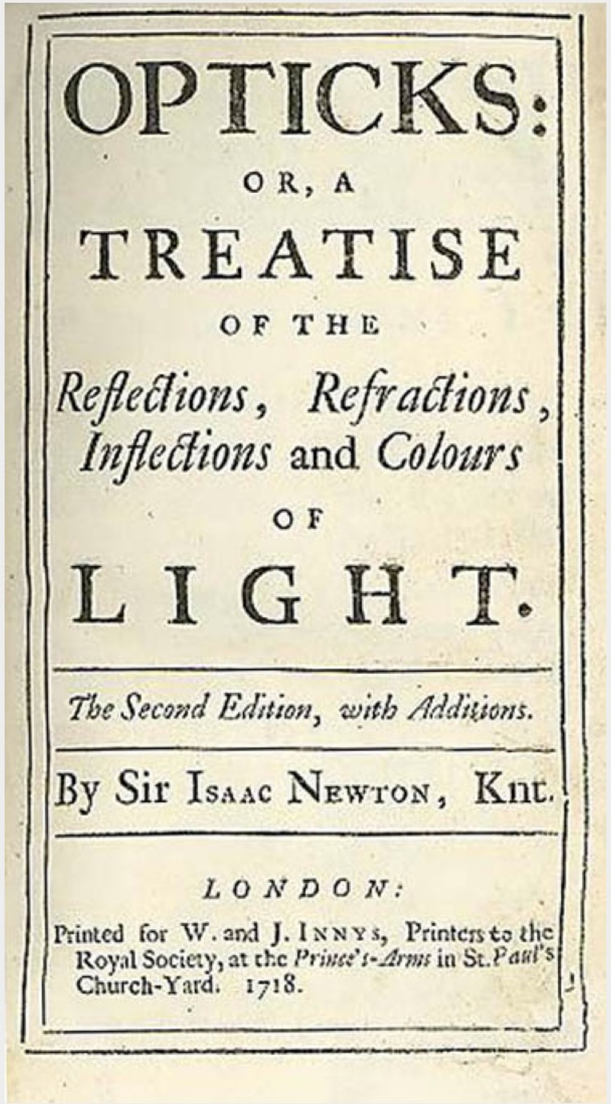
Integration of JUNO simulation framework with *Opticks* : GPU accelerated optical propagation via NVIDIA® OptiX™

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



Simon C Blyth, IHEP, CAS — On behalf of the JUNO Collaboration — vCHEP, 19 May 2021

Outline



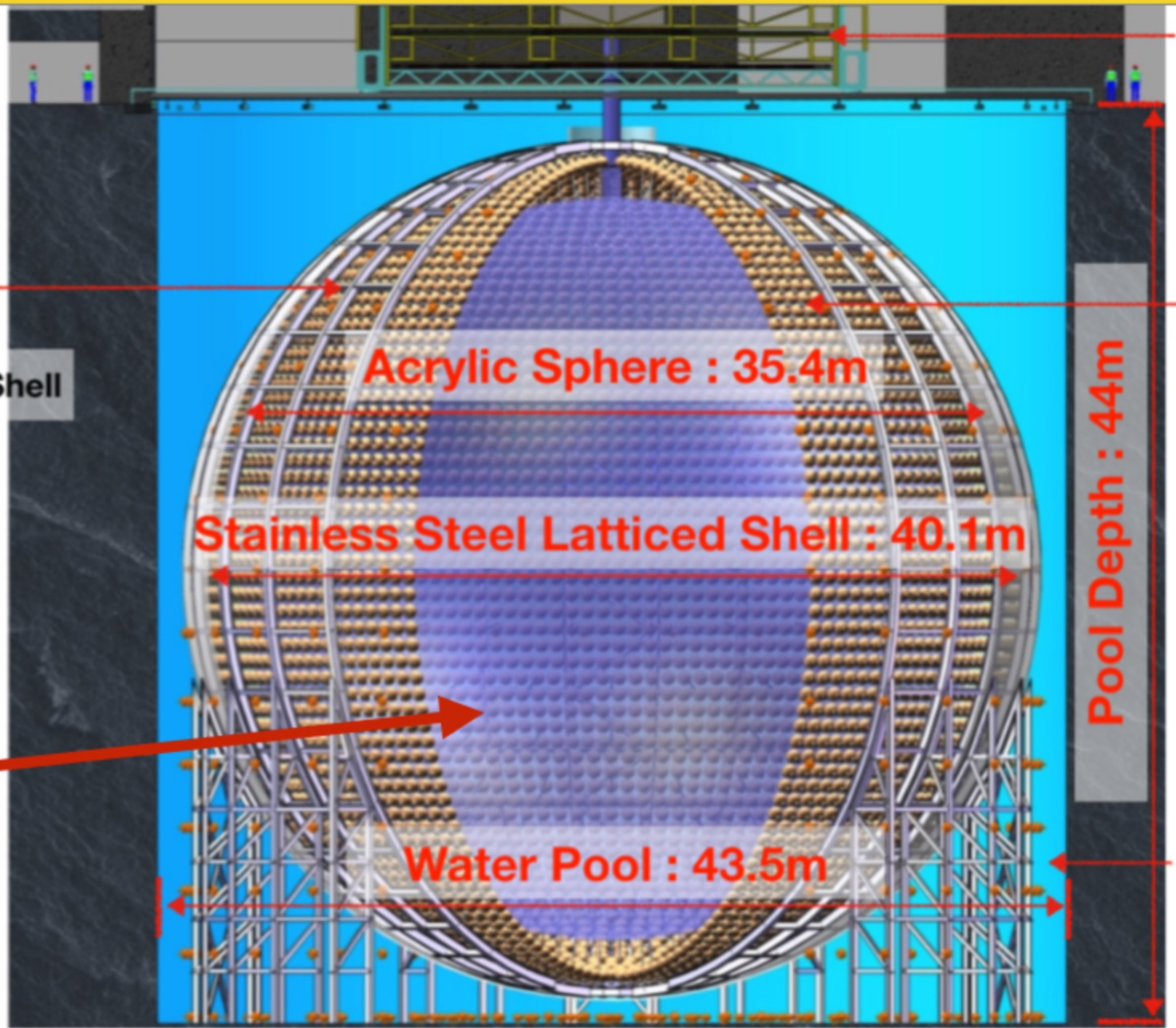
OPTICKS:
OR, A
TREATISE
OF THE
Reflections, Refractions, Inflections and *Colours*
OF
LIGHT.

The Second Edition, with Additions.

By Sir Isaac Newton, Knt.

LONDON:
Printed for W. and J. Innys, Printers to the Royal Society, at the *Prince's-Arms* in St. *Paul's* Church-Yard. 1718.

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



Top Tracker

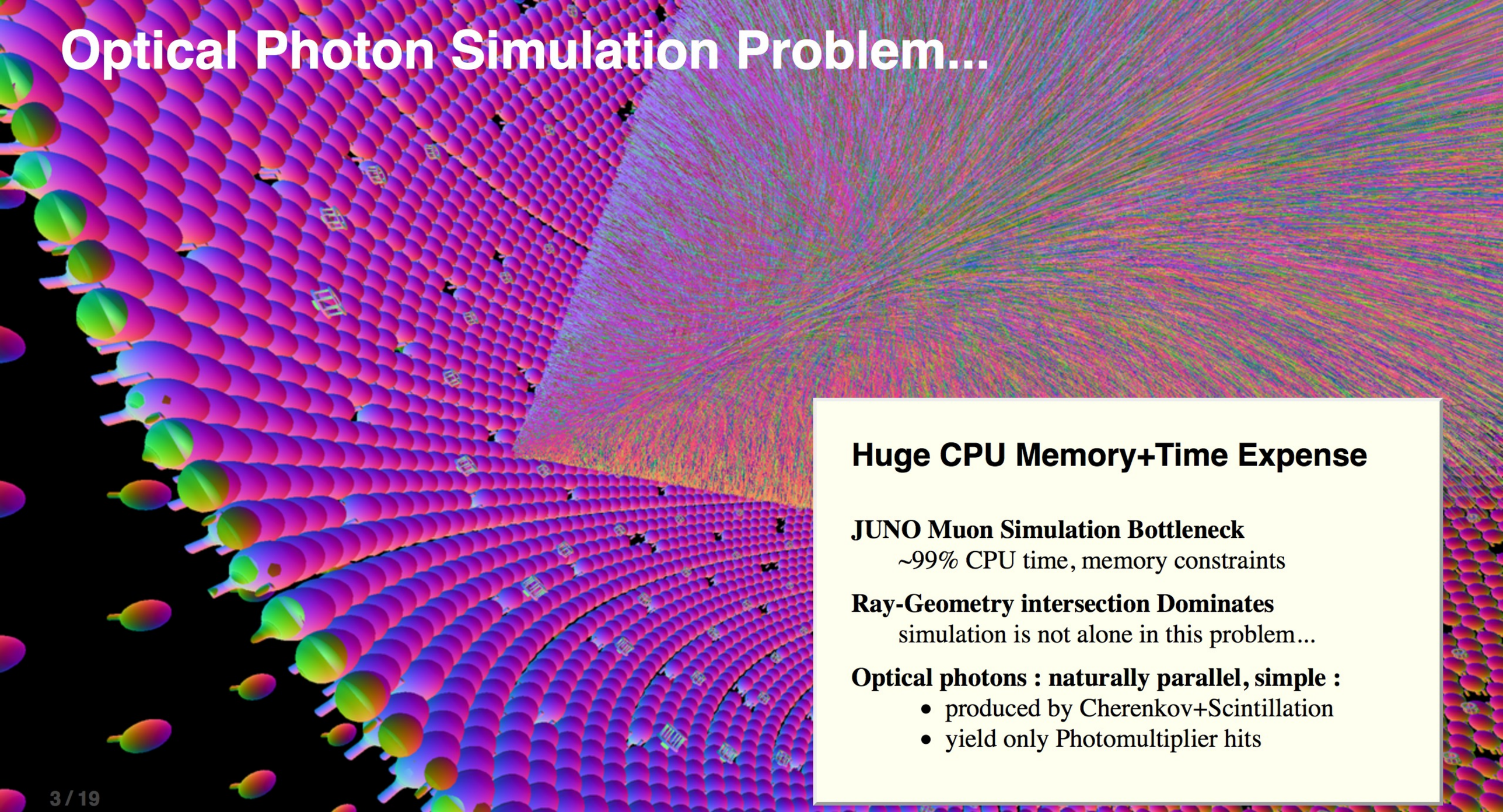
Central Detector
Acrylic Sphere
Stainless Steel Latticed Shell

PMT
18,000 20-inch PMTs
25,000 3-inch PMTs
78% Coverage

Liquid Scintillator
20 kton

Water Cherenkov
35 kton pure water
2,000 20-inch veto PMTs

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

simulation
photon parameters at sensors (PMTs)

rendering
pixel values at image plane

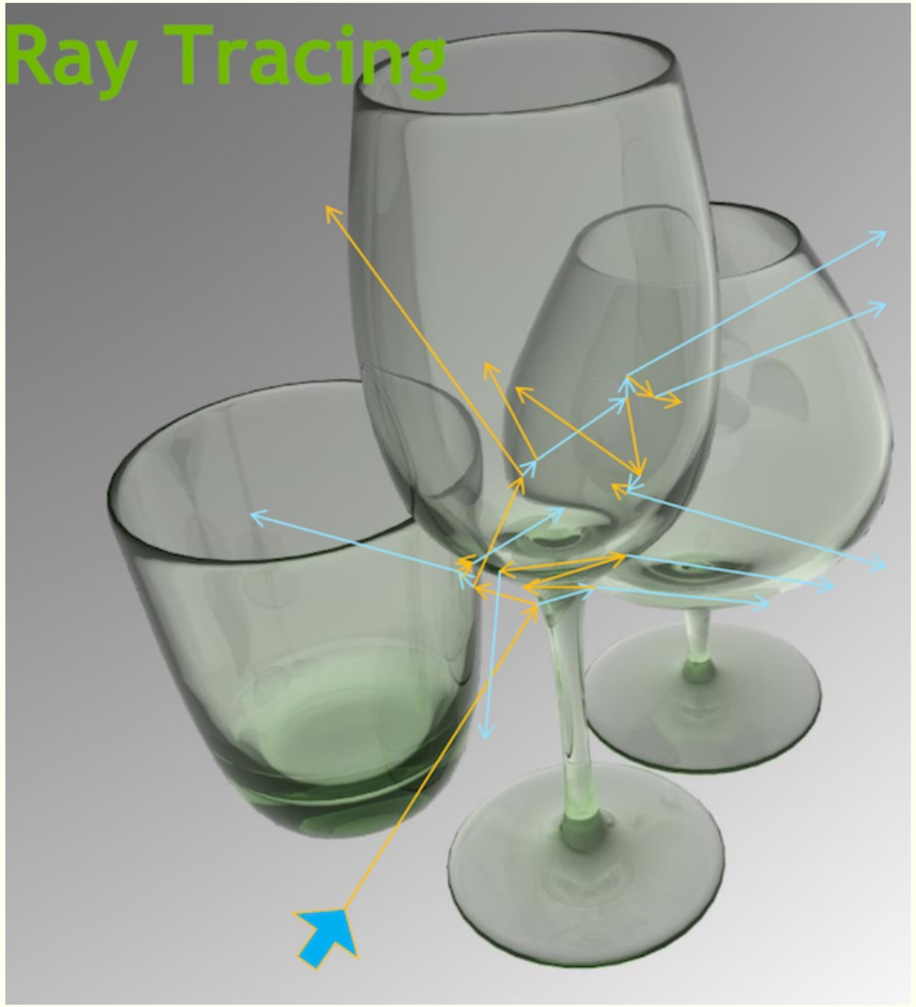
Much in common : geometry, light sources, optical physics

- **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



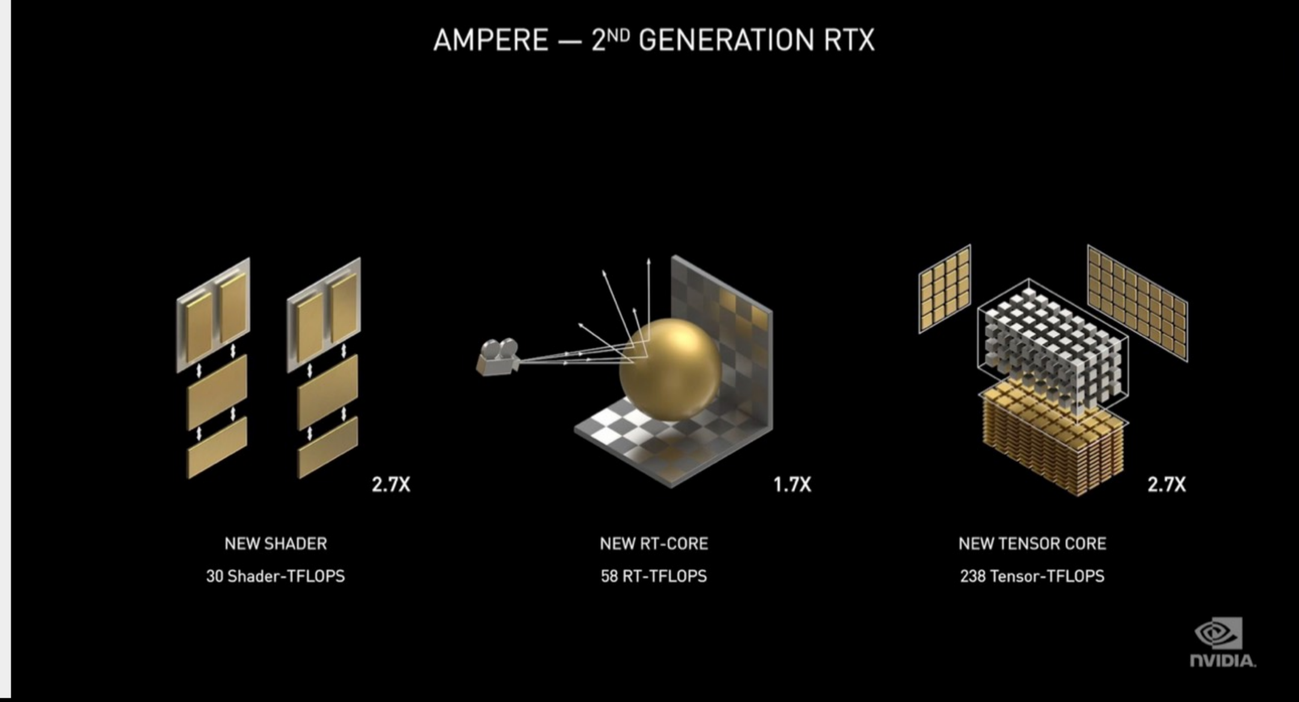
http://on-demand.gputechconf.com/siggraph/2013/presentation/SG3106-Building-Ray-Tracing-Applications-OptiX.pdf

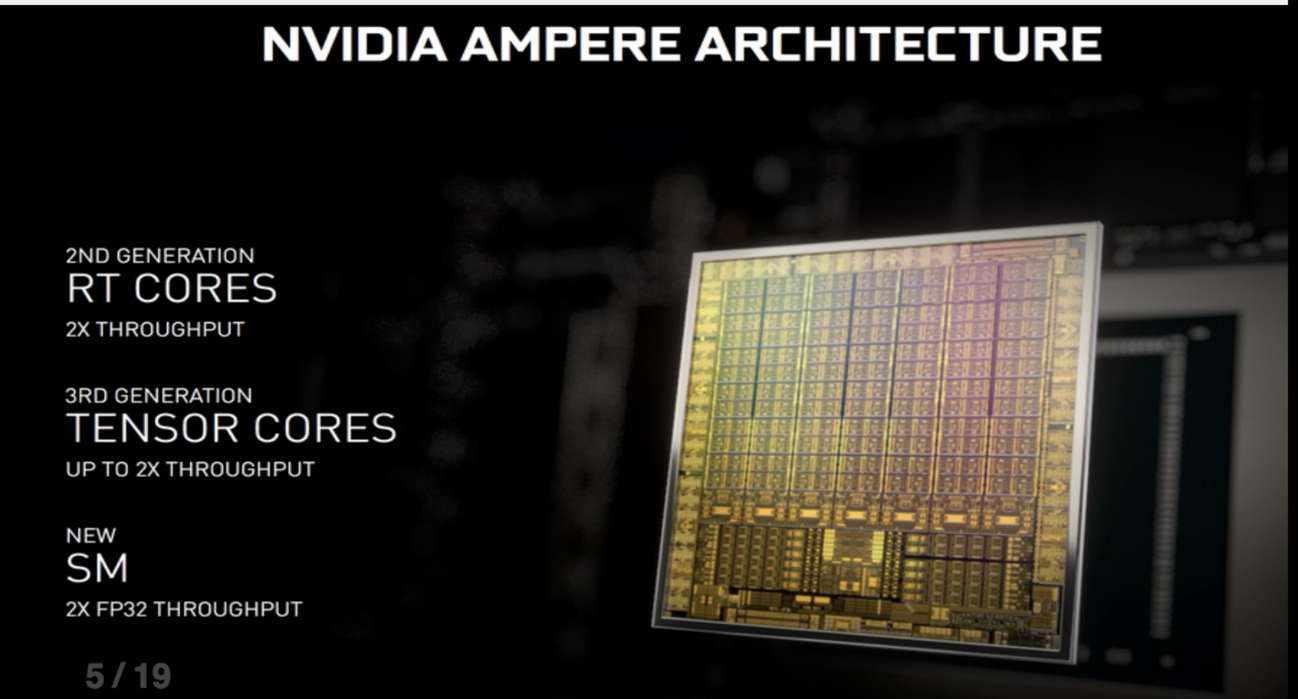
Ampere : 2nd Generation RTX

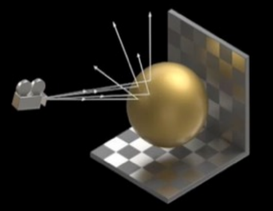
NVIDIA Ampere (2020):

"...triple double over Turing (2018, 10 GigaRays/s)..."

- **RT Core** : ray trace dedicated GPU hardware
- **NVIDIA GeForce RTX 3090**
  - 10,496 CUDA Cores, 28GB VRAM, USD 1499
- **ray trace performance continues rapid improvement**







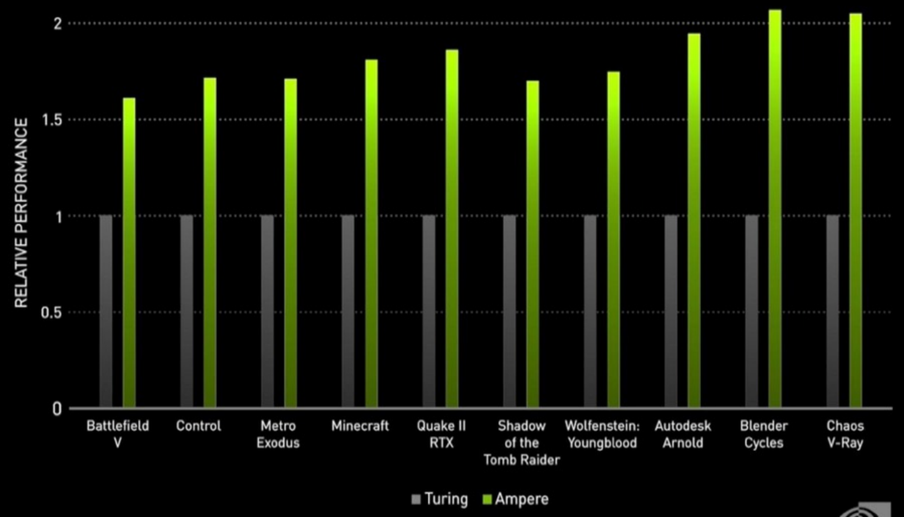
2ND GENERATION
RT CORE

Dedicated Hardware

2X Ray/Triangle Intersection

Concurrent RT + Graphics

Concurrent RT + Compute



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+ via RTX™ mode

NVIDIA expertise:

- OptiX pre-7 : **~linear scaling up to 4 GPUs**
- acceleration structure creation + traversal (Blue)
- instanced sharing of geometry + acceleration structures
- compiler optimized for GPU ray tracing

`https://developer.nvidia.com/rtx`

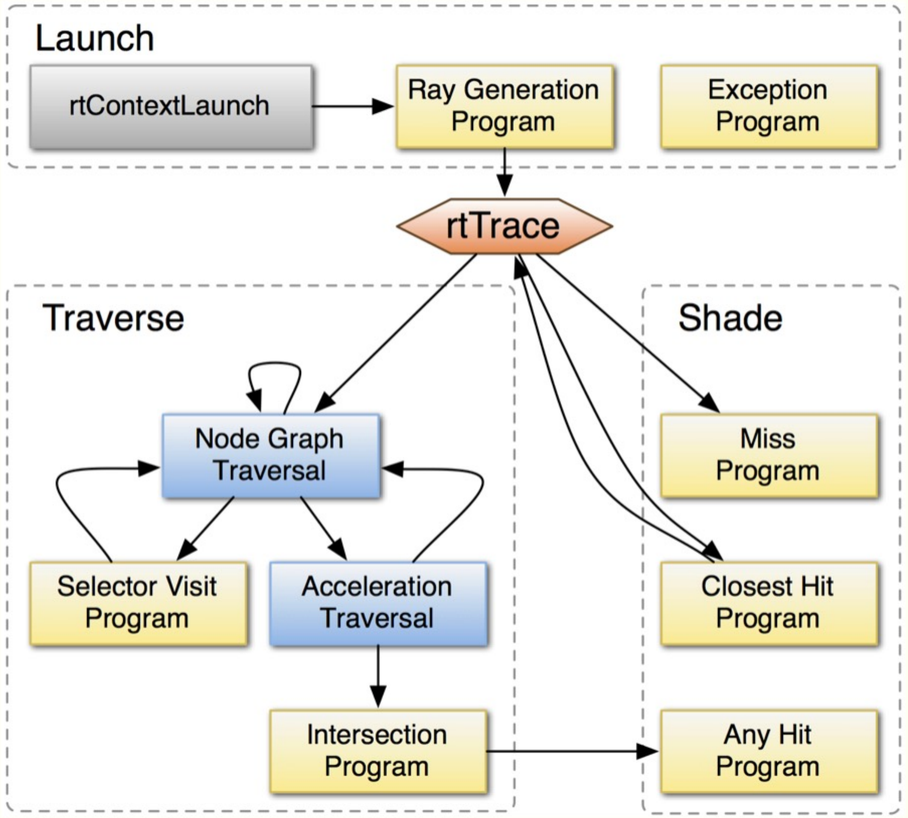
User provides (Yellow):

- ray generation
- geometry bounding box, intersects

[1] Turing+ GPUs eg NVIDIA TITAN RTX

OptiX Raytracing Pipeline

Analogous to OpenGL rasterization pipeline:



NVIDIA OptiX 7 : Entirely new thin API (Introduced Aug 2019)

NVIDIA OptiX 6->7 : **drastically slimmed down**

- headers only (no library, just Driver)
- low-level CUDA-centric thin API (Vulkan-ized)
- Minimal host state, All host functions are thread-safe
- GPU launches : explicit, asynchronous (CUDA streams)
- ~~near perfect scaling to 4 GPUs, for free~~
- ~~Shared CPU/GPU geometry context~~
- ~~GPU memory management~~
- ~~Multi-GPU support~~

Advantages

More control/flexibility over everything.

- Fully benefit from future GPUs
- **Keep pace with state-of-the-art GPU ray tracing**

Disadvantages

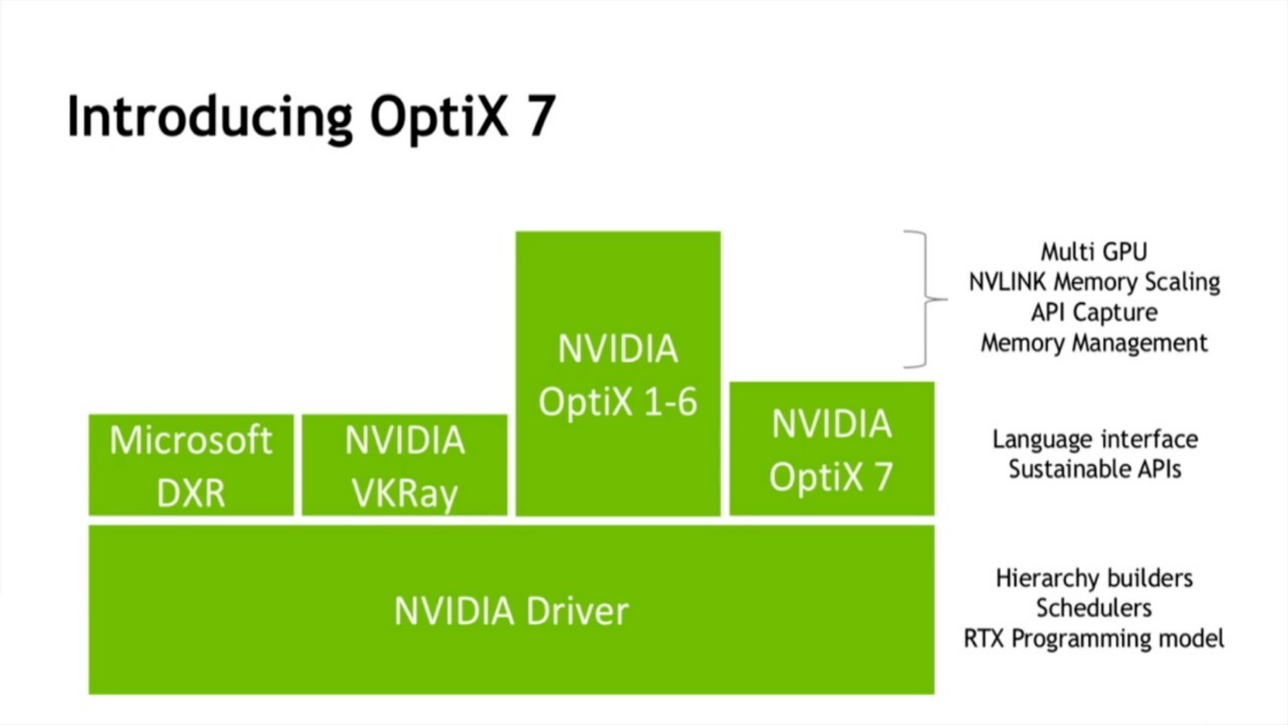
Demands much more developer effort than OptiX 6

- **Major re-implementation of Opticks required**

NEWS: Major Progress on 6->7 (p11,..)

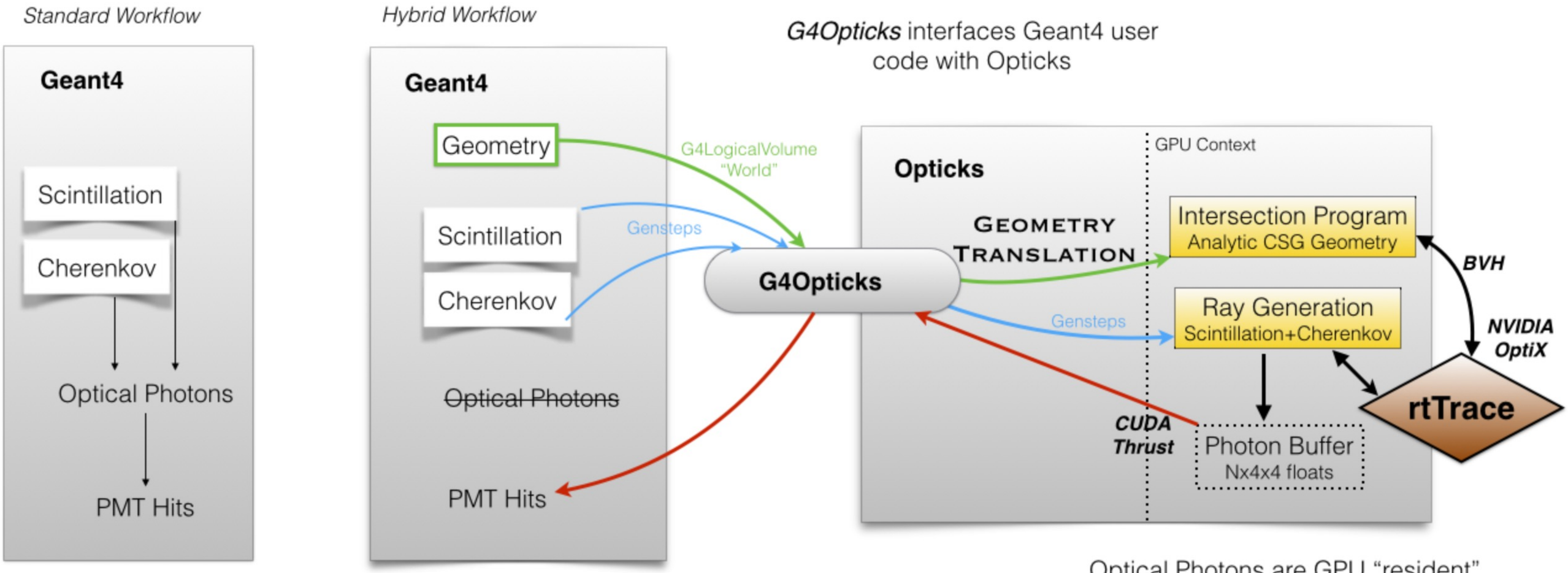
GPU Ray Tracing APIs Converged

- 3 APIs (DXR,VKRay,OptiX7) over RTX
- Driver updates **independent of application**
- Support new GPUs, performance improvements



Geant4 + Opticks Hybrid Workflow : External Optical Photon Simulation

https://bitbucket.org/simoncblyth/opticks



G4Opticks interfaces Geant4 user code with Opticks

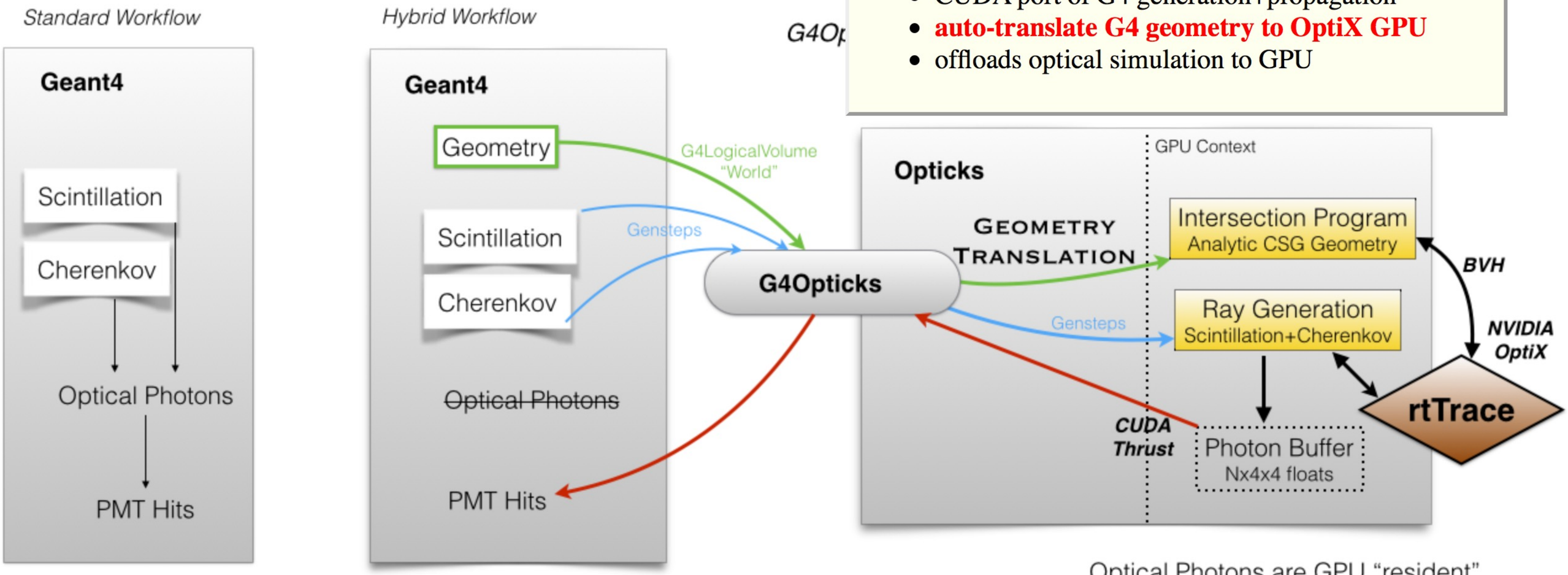
Optical Photons are GPU "resident", only hits are copied to CPU memory

Geant4 + Opticks Hybrid Workflow : Exter

https://bit

Opticks : GPU Optical Photons

- CUDA port of G4 generation+propagation
- **auto-translate G4 geometry to OptiX GPU**
- offloads optical simulation to GPU



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

Opticks + JUNO Progress : Efficiency Hit Culling on GPU

On GPU Efficiency Culling

- upload sensor angular efficiencies
- -> `COLLECT` OR `CULL` hit flags

CUDA Thrust **stream compaction**

- `thrust::count_if`
- `thrust::copy_if`
- Download hits with : `hitmask = COLLECT`

- **Reduces hit CPU memory by factor of the efficiency**
- reduce size of GPU->CPU downloads by factor of efficiency
- **all** downloaded hits -> electronics simulation

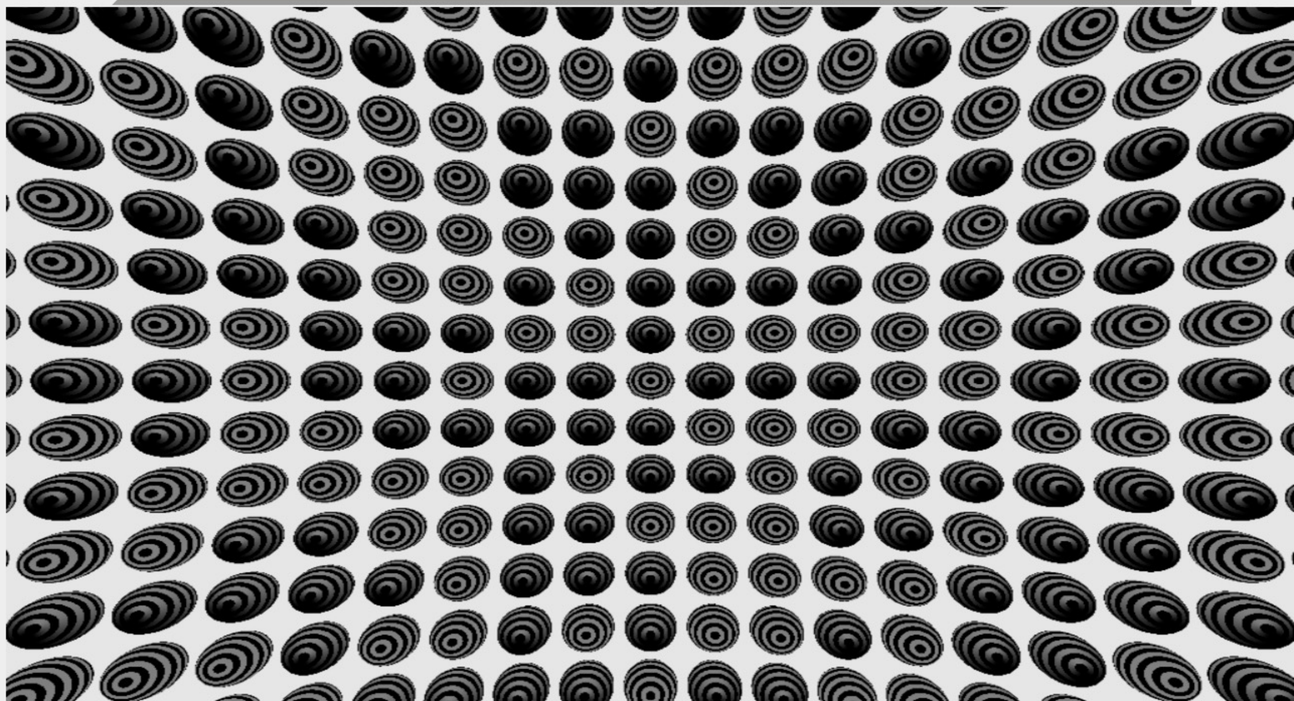
Mock angular efficiency test of GPU texture machinery :

- texturing sensor surfaces with the efficiency
- mockup striped theta efficiency, two random categories
- cosine phi efficiency variation for one category

ε, ε [θ, ϕ] -> GPU Tex

Separate Textures for each sensor type

```
G4Opticks::
   setSensorData
   setSensorAngularEfficiency
```



New "Foundry" Model : Shared CPU/GPU Geometry Context

- **replaces geometry context dropped in OptiX 6->7**
- array-based -> simple, inherent serialization + persisting
- entire geometry in 4 GPU allocations

Simple intersect headers, common CPU/GPU types

- use with : pre-7, 7 + testing on CPU

https://github.com/simoncblyth/CSG "Foundry" model
csg_intersect_tree.h/csg_intersect_node.h/...
simple headers common to pre-7/7/CPU-testing

https://github.com/simoncblyth/CSG_GGeo
Convert *Opticks/GGeo* -> *CSGFoundry*

https://github.com/simoncblyth/CSGOptiX
OptiX 7 + pre-7 rendering

GAS : Geometry Acceleration Structure

IAS : Instance Acceleration Structure

CSG : Constructive Solid Geometry

IAS < Inst < Solid < Prim < Node

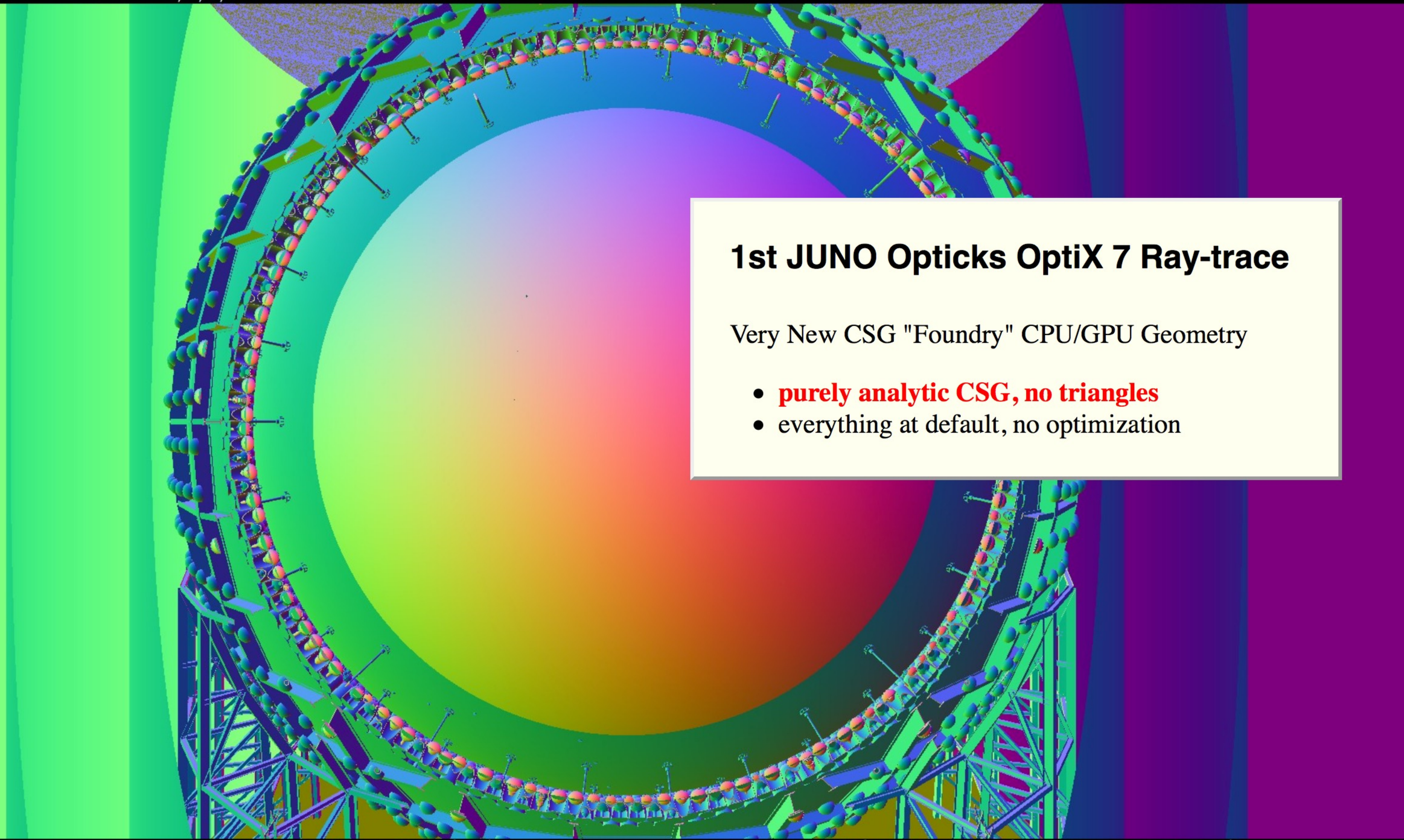
- **Inst** : 4x4 tran. + **Solid** ref. (**Inst** -> 1 **IAS**)
- **Solid** : 1 or more **Prim** : (**Solid** -> **GAS**)
- **Prim** : 1,3,7,15,31,... **Node** : (**Prim** ~ *G4VSolid*)

```
struct CSGFoundry
{
   void upload(); // to GPU
...
   std::vector<CSGSolid>  solid ; // compounds (eg PMT)
   std::vector<CSGPrim>   prim ;
   std::vector<CSGNode>   node ; // shapes, operators

   std::vector<float4> plan ; // planes
   std::vector<qat4>   tran ; // CSG transforms
   std::vector<qat4>   itra ; // inverse CSG transforms
   std::vector<qat4>   inst ; // instance transforms

   // entire geometry in four GPU allocations
   CSGPrim*    d_prim ;
   CSGNode*    d_node ;
   float4*     d_plan ;
   qat4*       d_itra ;
 };
```

referencing by offset, count



1st JUNO Opticks OptiX 7 Ray-trace

Very New CSG "Foundry" CPU/GPU Geometry

- **purely analytic CSG, no triangles**
- everything at default, no optimization

Current JUNO Geometry : Auto-Factorized by "progeny digest"

ridx	plc	prim	component	note
0	1	3084	3084:sWorld	non-repeated remainder
1	25600	5	5:PMT_3inch_pmt_solid	4 types of PMT
2	12612	5	5:NNVTMCPPMTsMask	
3	5000	5	5:HamamatsuR12860sMask	
4	2400	5	5:mask_PMT_20inch_vetosMask	
5	590	1	1:sStrutBallhead	4 parts of same assembly, BUT not grouped as siblings (not parent-child)
6	590	1	1:uni1	
7	590	1	1:base_steel	
8	590	1	1:uni_acrylic3	
9	504	130	130:sPanel	repeated parts of TT

Factorize ~300,000 vol -> 10 comp

- **ridx**: repeat index
- **plc**: number of placements of the instance
- **prim**: number of Prim/volumes in the instance
- **component**: numPrim:outerPrimName

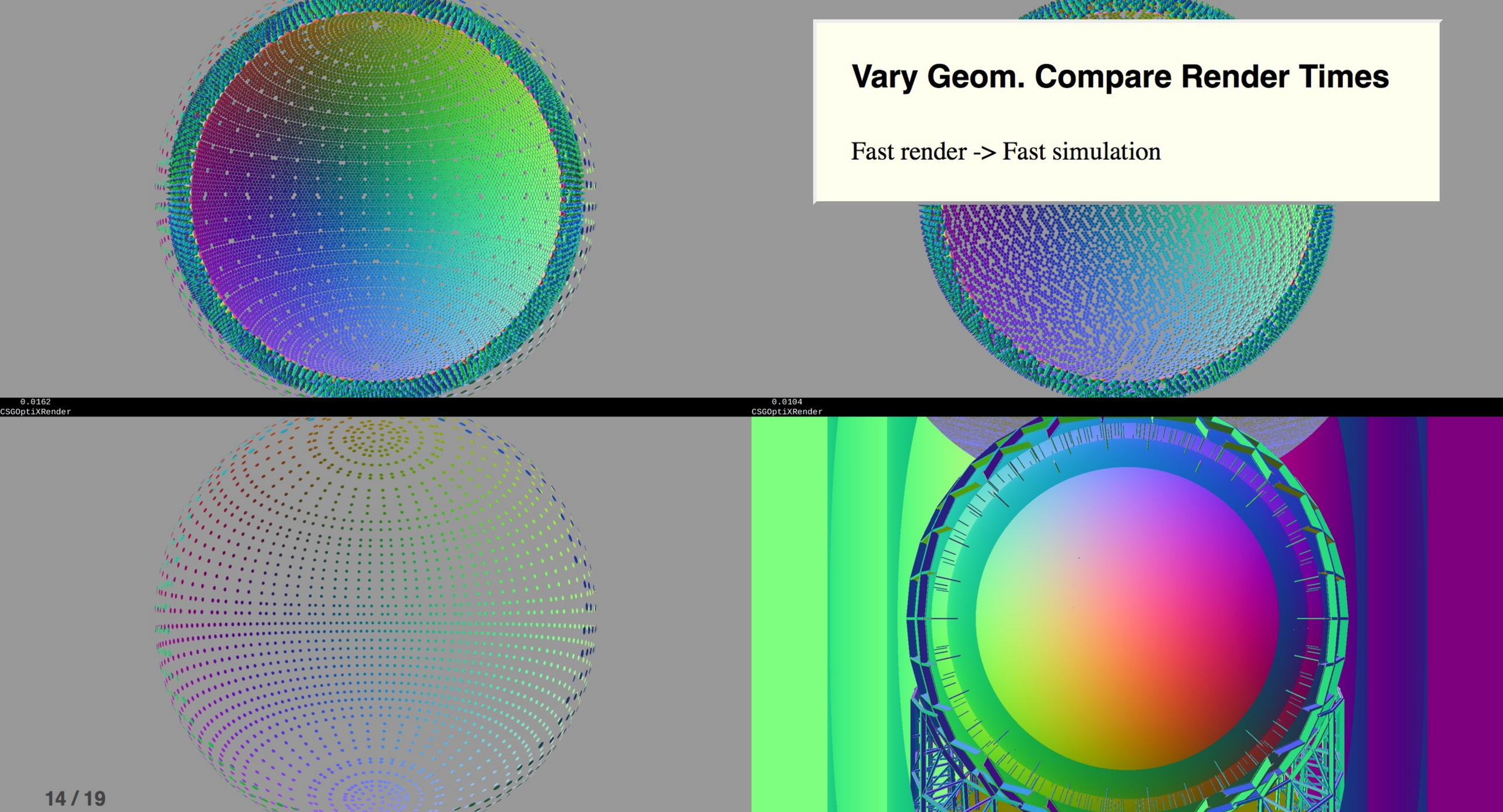
"progeny digest" characterizes subtree of every volume-node

- **ridx:0** "remainder" Prim
  - Prim that did not pass instancing criteria, on number of repeats + complexity
  - TODO: tune criteria to instance more, reducing remainder Prim (Expect: 3084->~ 84)
- **ridx:1,2,3,4**
  - four types of PMT, all with 5 Prim
- **ridx:5,6,7,8**
  - same 590x assembly **but not grouped together** : as siblings (not parent-child like PMTs)
  - TODO: implement instancing of siblings, combining 4 -> 1

Increasing instancing : reduces memory for geometry -> improved performance

Vary Geom. Compare Render Times

Fast render -> Fast simulation



JUNO Geometry : OptiX 7 Ray Trace Times ~2M pixels : TITAN RTX

idx	-e	time(s)	relative	enabled geometry description
0	9,	0.0017	0.1702	ONLY: 130:sPanel
1	7,	0.0017	0.1714	ONLY: 1:base_steel
2	6,	0.0019	0.1923	ONLY: 1:uni1
3	5,	0.0027	0.2780	ONLY: 1:sStrutBallhead
4	4,	0.0032	0.3268	ONLY: 5:mask_PMT_20inch_vetosMask
5	1,	0.0032	0.3287	ONLY: 5:PMT_3inch_pmt_solid
6	2,	0.0055	0.5669	ONLY: 5:NNVTMCPPMTsMask
7	3,	0.0074	0.7582	ONLY: 5:HamamatsuR12860sMask
8	**1,2,3,4**	**0.0097**	**1.0000**	**ONLY PMT**
9	t8,0	0.0099	1.0179	EXCL: 1:uni_acrylic3 3084:sWorld
10	**0,**	**0.1171**	**12.0293**	**ONLY: 3084:sWorld**
11	**t8,**	**0.1186**	**12.1769**	**EXCL: 1:uni_acrylic3**
12	**t0,**	**0.5278**	**54.2066**	**EXCL: 3084:sWorld**
13	**8,**	**0.5310**	**54.5298**	**ONLY: 1:uni_acrylic3**
14	t3,	0.6017	61.7954	EXCL: 5:HamamatsuR12860sMask
15	t2,	0.6043	62.0620	EXCL: 5:NNVTMCPPMTsMask
16	t5,	0.6171	63.3787	EXCL: 1:sStrutBallhead
17	t6,	0.6196	63.6301	EXCL: 1:uni1
18	t7,	0.6226	63.9458	EXCL: 1:base_steel
19	t0	0.6240	64.0879	3084:sWorld
20	t4,	0.6243	64.1169	EXCL: 5:mask_PMT_20inch_vetosMask
21	t9,	0.6335	65.0636	EXCL: 130:sPanel
22	t1,	0.6391	65.6384	EXCL: 5:PMT_3inch_pmt_solid

Same viewpoint, vary GPU geometry

- **-e** : controls components : "t" means ~ (NOT)
- **time(s)** : GPU ray trace CUDA launch time
- **relative** : compares to "ONLY PMT" baseline

Very large range of times 1:600

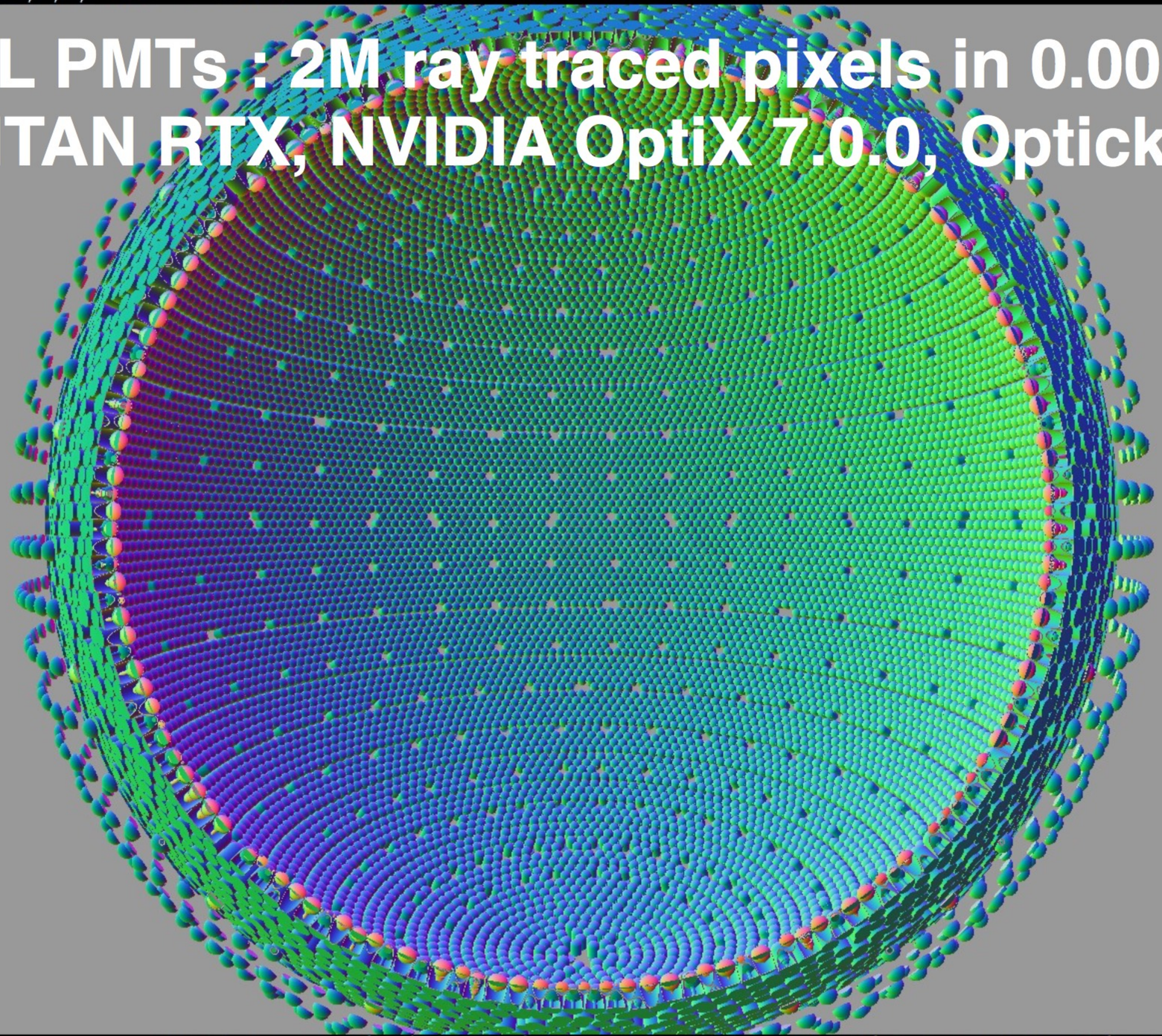
Table identifies slow geometry to fix :

- **3084:sWorld** (too many non-instanced Prim)
- **1:uni_acrylic** (CSG sub. 35m diam. sphere)

Good performance for **ONLY PMTs** :

- **45,612 PMT instances handled without issue**

JUNO ALL PMTs : 2M ray traced pixels in 0.0097 s : NVIDIA TITAN RTX, NVIDIA OptiX 7.0.0, Opticks



Summary and Links

Opticks : state-of-the-art GPU ray traced optical simulation integrated with *Geant4*. Geometry progress with OptiX 7 suggests integration within months is achievable.

Next Steps

- Foundry JUNO geometry validation
- migrate optical physics to NVIDIA OptiX 7
- JUNO Production validation

- Efficiency culling decision moved to GPU, reducing CPU hit memory
- "Foundry" geometry model implemented to support NVIDIA OptiX 7 API.
- **First JUNO OptiX 7 renders achieved.**

https://bitbucket.org/simoncblyth/opticks ❏	code repository
https://github.com/simoncblyth/opticks/releases ❏	.zip .tar.gz archives
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

Acknowledgement : Opticks "Hackathon" Series for NVIDIA OptiX 6->7

- Suggested and organized by LZ, LBNL, NERSC
- ~fortnightly meetings since Feb 2021
- **Valuable advice from NVIDIA engineers**
- see next talk by Oisin Creaner, **LZ with Opticks**

LZ with Opticks (Images from Sam Eriksen, University of Bristol)