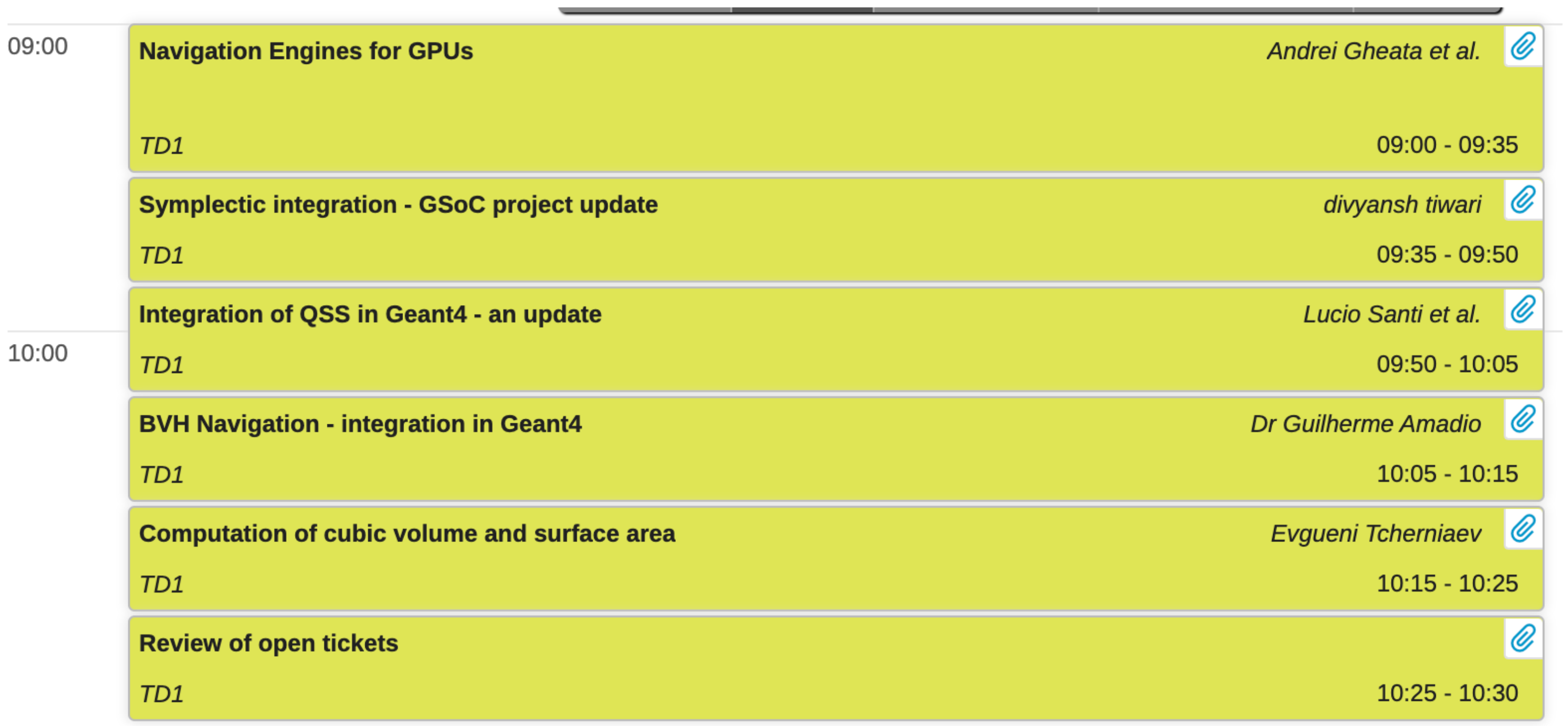
Summary of Geometry Parallel Session 4A



Time	Title	Speaker	Room	Slot
09:00	**Navigation Engines for GPUs**	*Andrei Gheata et al.*	*TD1*	09:00 - 09:35
	Symplectic integration - GSoC project update	*divyansh tiwari*	*TD1*	09:35 - 09:50
10:00	**Integration of QSS in Geant4 - an update**	*Lucio Santi et al.*	*TD1*	09:50 - 10:05
	BVH Navigation - integration in Geant4	*Dr Guilherme Amadio*	*TD1*	10:05 - 10:15
	Computation of cubic volume and surface area	*Evgueni Tcherniaev*	*TD1*	10:15 - 10:25
	Review of open tickets		*TD1*	10:25 - 10:30

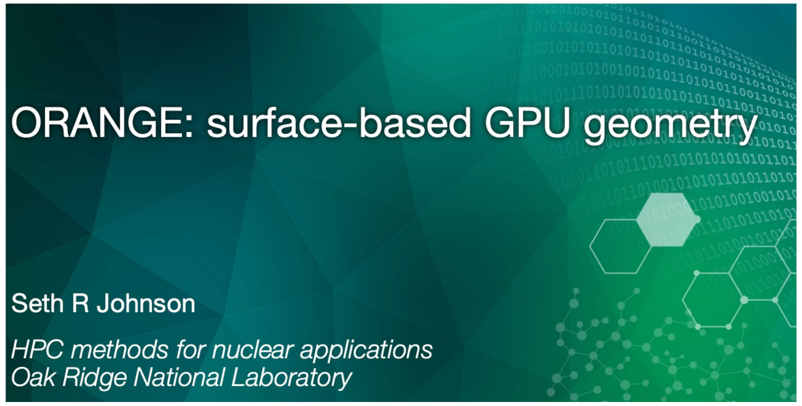
John Apostolakis - 29 Sept 2022

R&D on Surface-based Modellers for Performant Navigation on GPU

Orange
- intersections using unbounded surfaces, aided by Bounding Boxes

VecGeom extention
- intersections using bounded surfaces (surfaces with outlines – masks.)

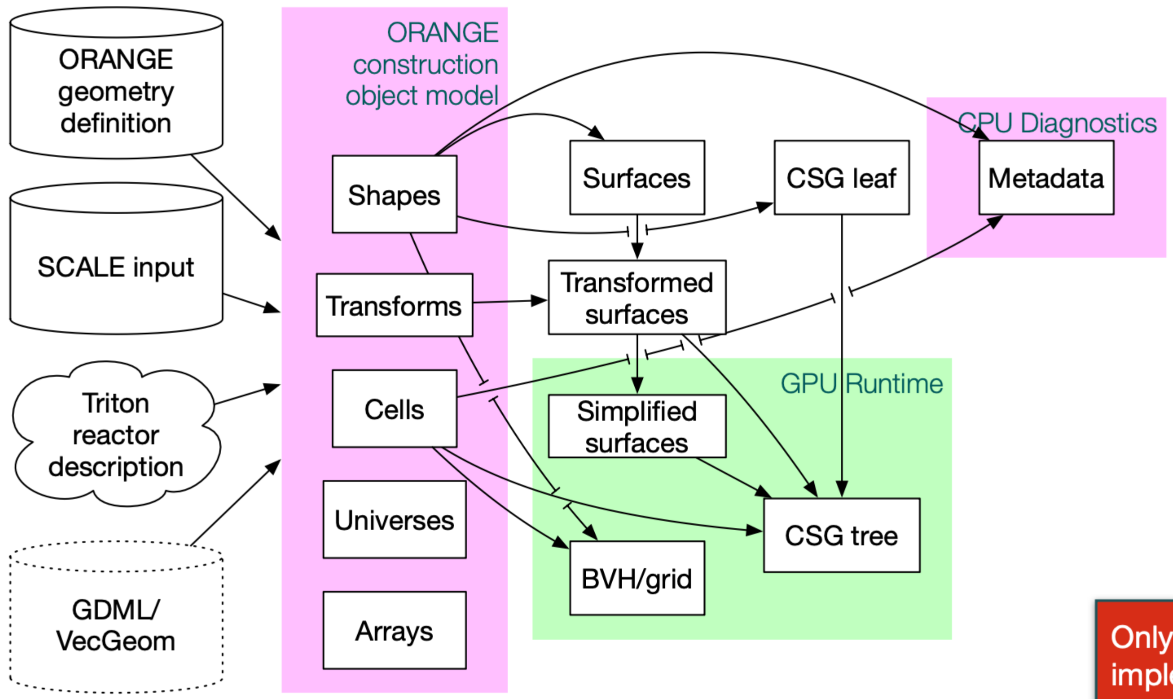
Orange – unbounded volume modeller


ORANGE: surface-based GPU geometry

Seth R Johnson

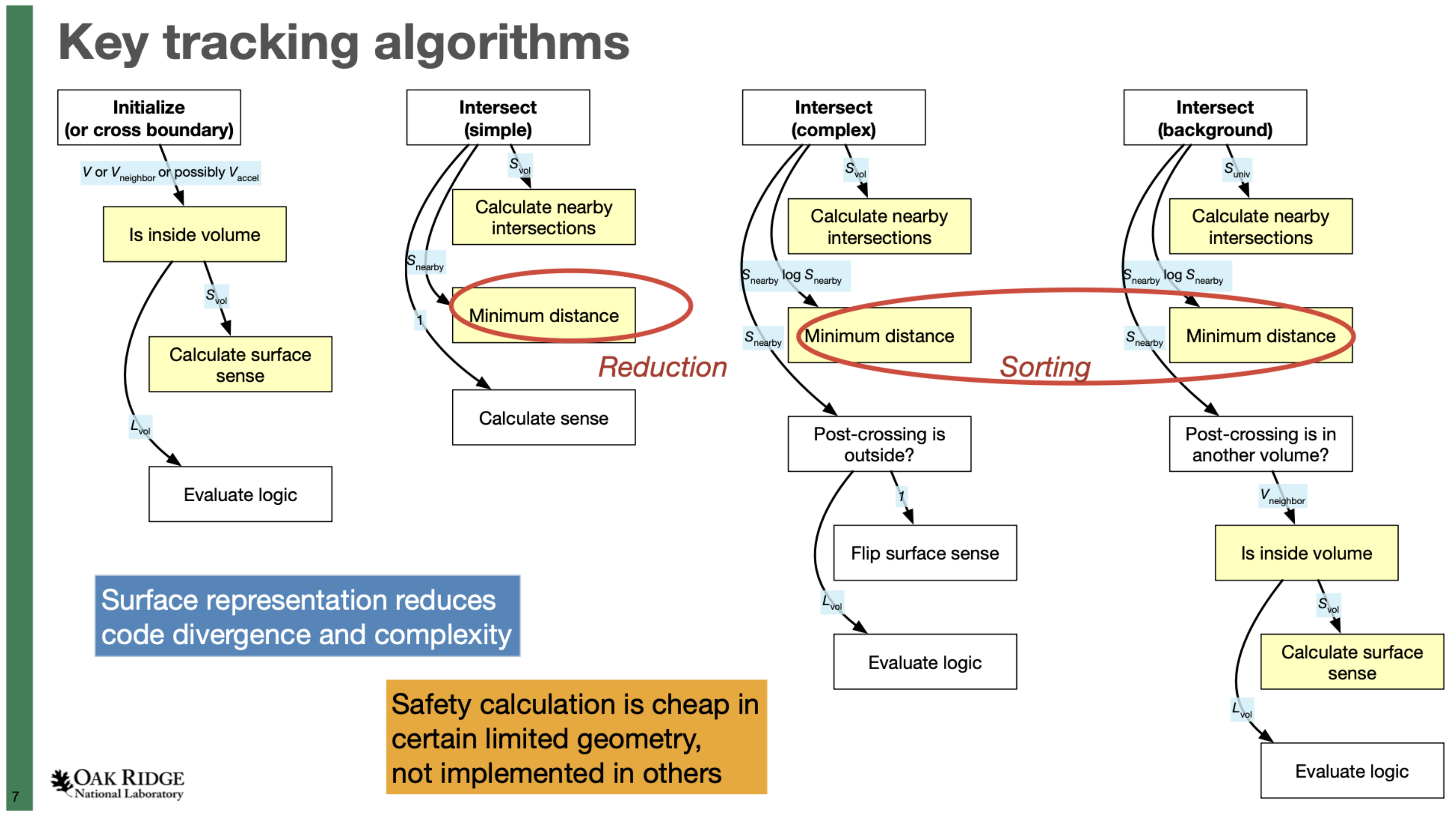
HPC methods for nuclear applications
Oak Ridge National Laboratory

ORANGE surface/volume construction



Only **partially** implemented in Celeritas ORANGE

Orange – bounded volume modeller

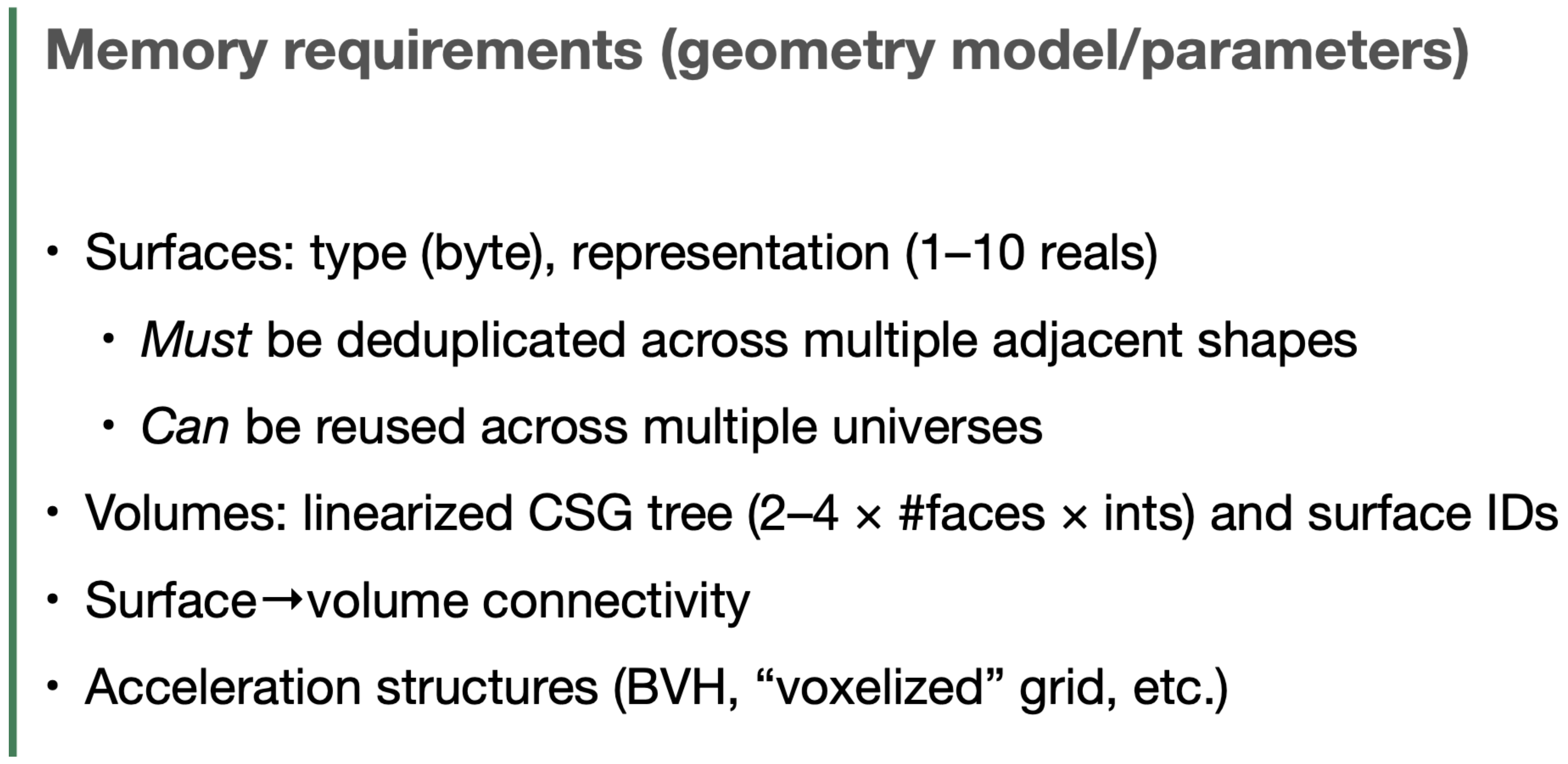


Key tracking algorithms

Surface representation reduces code divergence and complexity

Safety calculation is cheap in certain limited geometry, not implemented in others

Orange – unbounded volume modeller



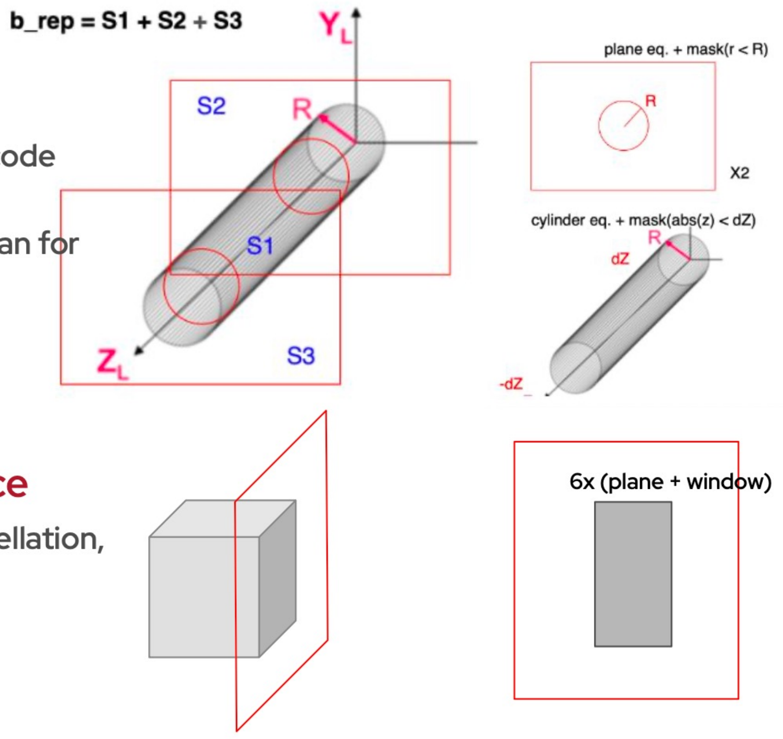
Memory requirements (geometry model/parameters)

- Surfaces: type (byte), representation (1–10 reals)
  - *Must* be deduplicated across multiple adjacent shapes
  - *Can* be reused across multiple universes
- Volumes: linearized CSG tree (2–4 × #faces × ints) and surface IDs
- Surface→volume connectivity
- Acceleration structures (BVH, “voxelized” grid, etc.)

VecGeom – bounded surfaces

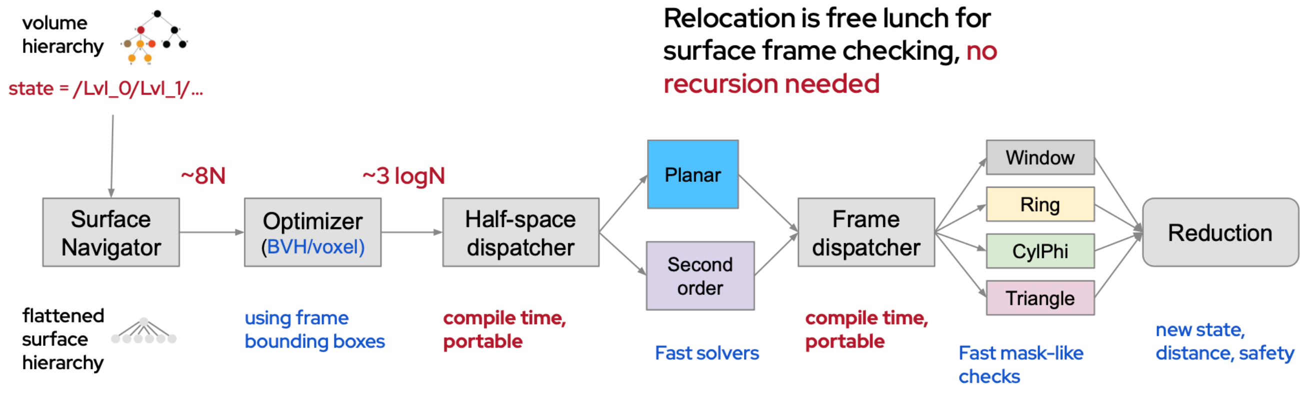
Addressing the problem: surface models

- Rationale: factoring the navigation problem at lower level
  - More simple and uniform code, even if code path is sometimes longer
  - Less branching for primitive surfaces than for primitive solids
  - Allow reducing the number and size of divergent critical sections
- Each face of a solid described as half-space + frame = FramedSurface
  - Same functionality as triangles in a tessellation, but providing accurate modelling
  - Box: 6 x (plane + window frame)



VecGeom bounded surface model

Navigation for a bounded surface model



🤞 better balanced input per particle due to flattening and mixing surfaces from different volumes

🤞 faster divergent sections with fewer branches (2 half-space types and 6-8 masks) (?)

Next steps

- Ready for implementing GPU awareness
  - Header-only implementation, POD types with indices to be transferred
  - No additional code should be needed except function annotations & copy to GPU
  - The data store is percolated through interfaces
- Comparative test of performance in AdePT for increasing geometry complexity
  - Plug-in into geometry-agnostic examples comparing with the volume approach
- If successful, expand the model and implement the missing features
  - Evolving the model to support more solids now much easier
  - Challenges foreseen for the Boolean solid implementation, which may need to map into a volume-based approach

Draft summary of discusion

- Agree(d) the set of benchmark geometries (with increasing complexity)
- Obtain initial benchmarks of the different approaches
- Evaluate the potential of each approach
- Clarify what remains to be implemented & estimate effort
- Estimated Timescale for first comparisons: 2-3 months

New methods for integration of tracks in magnetic fields

1. Symplectic integration for energy & phase space volume conservation
2. QSS method integration upate

Symplectic Integration

Maintaining energy, phase space volume during integration for 'many turns' in accelerator applications, muon (g-2)

Google Summer of Code
Symplectic Integrators

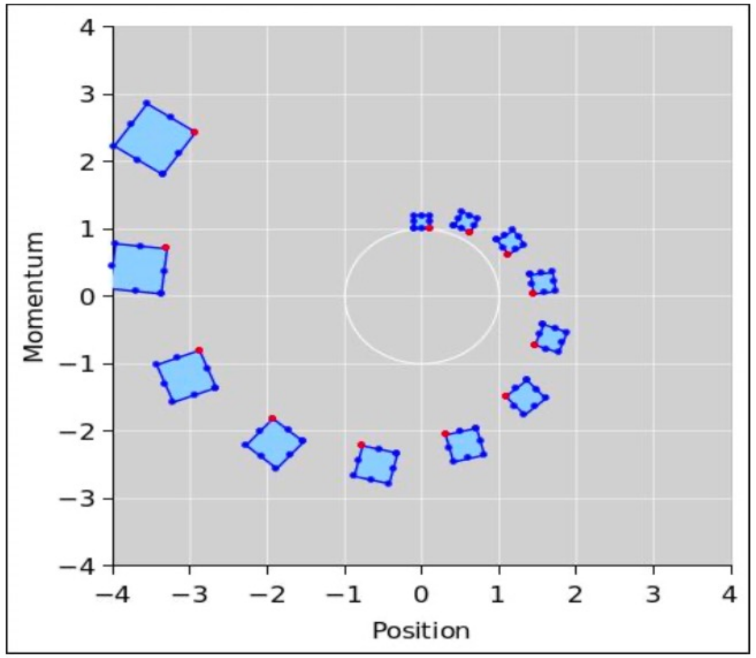
Divyansh Tiwari

Mentors:
Soon Yung Jun
John Apostolakis
Renee Fatemi

$$\frac{\mathbf{x}_{k+1} - \mathbf{x}_k}{\Delta t} = \mathbf{v}_{k+1/2},$$
$$\frac{\mathbf{v}_{k+1/2} - \mathbf{v}_{k-1/2}}{\Delta t} = \frac{q}{m}\left(\mathbf{E}_k + \frac{\mathbf{v}_{k+1/2} + \mathbf{v}_{k-1/2}}{2} \times \mathbf{B}_k\right)$$

The Boris Method

- We decided that a good starting point will be the boris algorithm.
- Due to explicitness, it is fast and conserves energy to 2nd order.



Quantum State Systems

Progress in adapting the QSS Stepper to the current version of Geant4
Testing and benchmark results

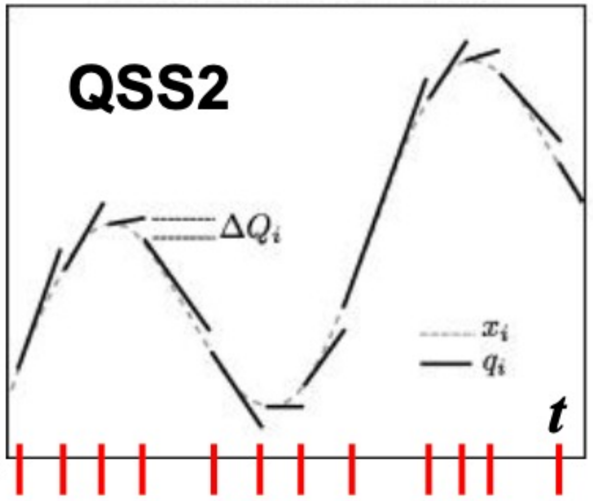
Rodrigo Castro and Lucio Santi

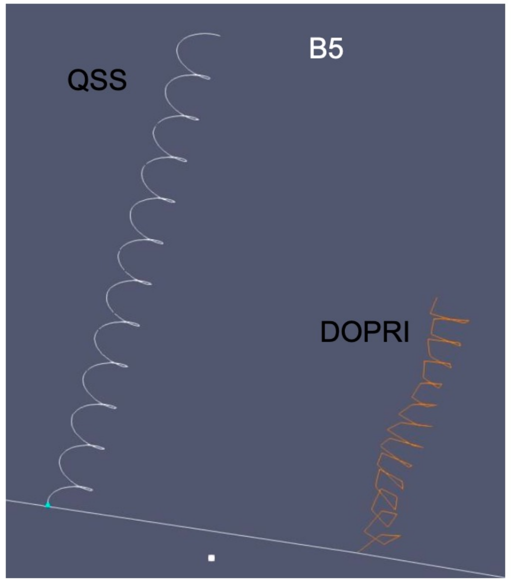
University of Buenos Aires and
ICC-CONICET, Argentina.
rcastro@dc.uba.ar

Results highlights

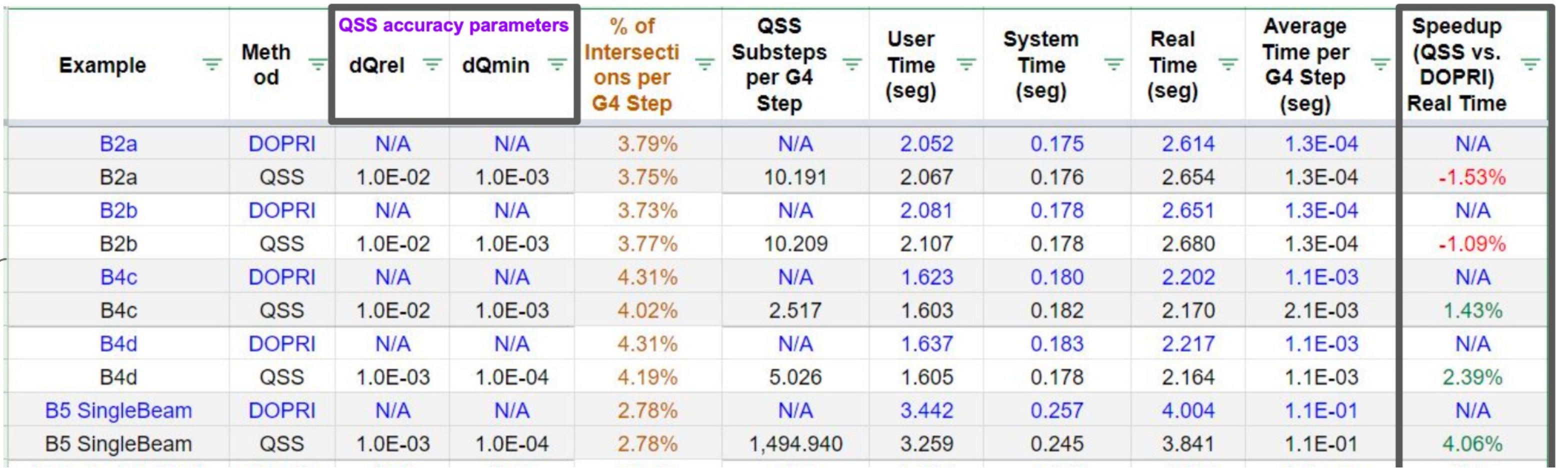
- 9 examples tested and verified successfully: **Basic** (B2a, B2b, B4c, B4d, B5) and **Extended** (with magnetic field: 01, 02, 03, 06)
- Benchmarks made against G4 (ver. 11.0.0-ref-02) default stepper (DOPRI with Interpolation Driver)
- In 5 cases *there exist **QSS accuracy parameters*** that can outperform DOPRI
  - However, the ratio of geometry intersections per G4 step remains below 19% in all tested examples (typically around 5%) => these are **not** "QSS-friendly" scenarios (not too many intersections per step)

1-st order quantizer
2-nd order method: QSS2





Summary of results: QSS vs. DOPRI



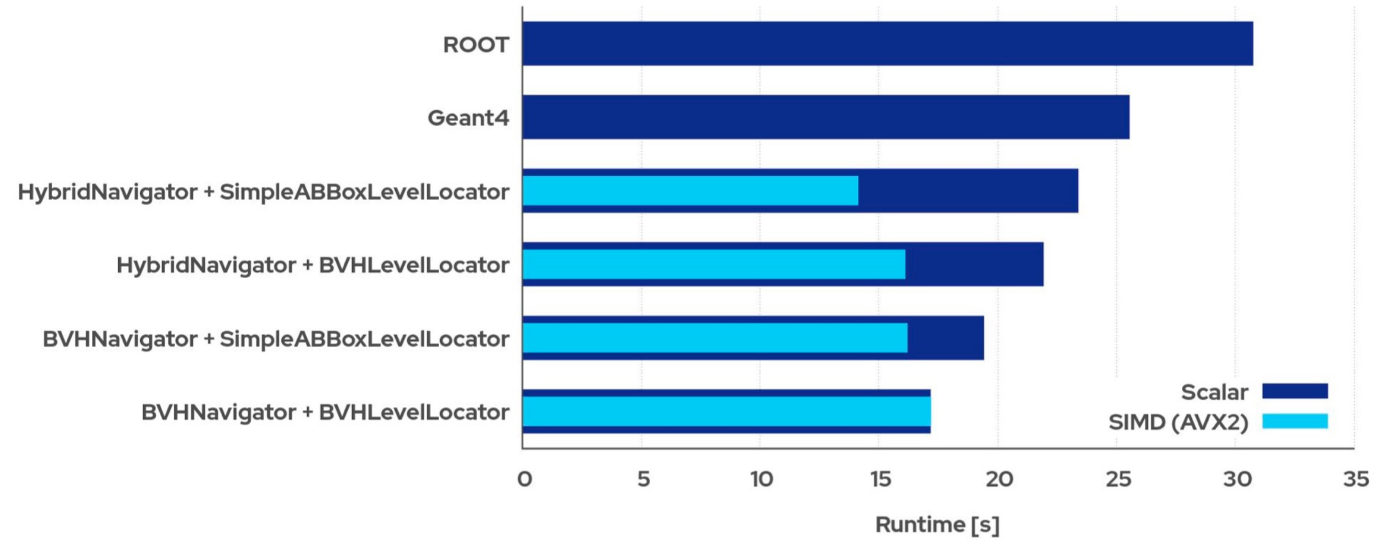
Example	Method	QSS accuracy parameters dQrel	dQmin	% of Intersections per G4 Step	QSS Substeps per G4 Step	User Time (seg)	System Time (seg)	Real Time (seg)	Average Time per G4 Step (seg)	Speedup (QSS vs. DOPRI) Real Time
B2a	DOPRI	N/A	N/A	3.79%	N/A	2.052	0.175	2.614	1.3E-04	N/A
B2a	QSS	1.0E-02	1.0E-03	3.75%	10.191	2.067	0.176	2.654	1.3E-04	-1.53%
B2b	DOPRI	N/A	N/A	3.73%	N/A	2.081	0.178	2.651	1.3E-04	N/A
B2b	QSS	1.0E-02	1.0E-03	3.77%	10.209	2.107	0.178	2.680	1.3E-04	-1.09%
B4c	DOPRI	N/A	N/A	4.31%	N/A	1.623	0.180	2.202	1.1E-03	N/A
B4c	QSS	1.0E-02	1.0E-03	4.02%	2.517	1.603	0.182	2.170	2.1E-03	1.43%
B4d	DOPRI	N/A	N/A	4.31%	N/A	1.637	0.183	2.217	1.1E-03	N/A
B4d	QSS	1.0E-03	1.0E-04	4.19%	5.026	1.605	0.178	2.164	1.1E-03	2.39%
B5 SingleBeam	DOPRI	N/A	N/A	2.78%	N/A	3.442	0.257	4.004	1.1E-01	N/A
B5 SingleBeam	QSS	1.0E-03	1.0E-04	2.78%	1,494.940	3.259	0.245	3.841	1.1E-01	4.06%

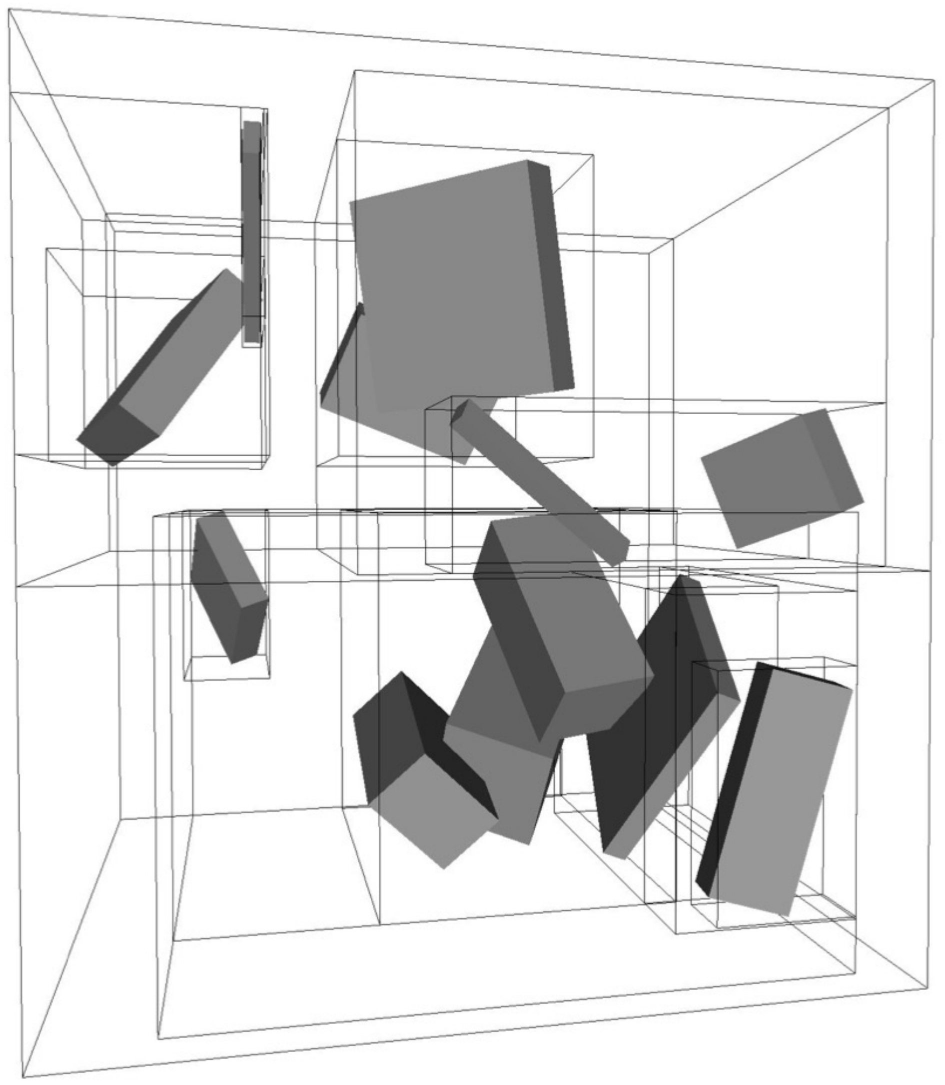
(*)

Bounding Volume Hierarchy Acceleration in Geant4

G. Amadio 29 Sep 2022

Performance of BVH Navigator in VecGeom



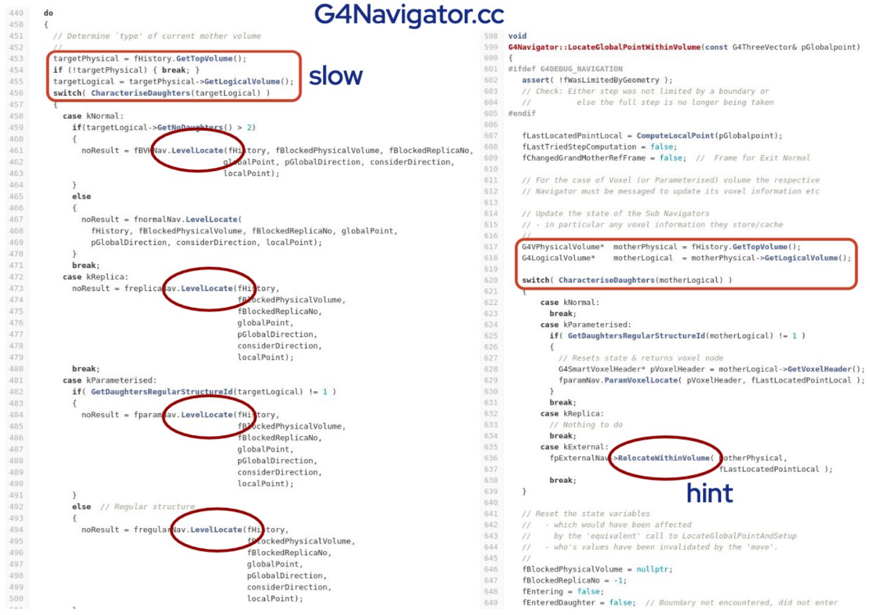


Bounding Volume Hierarchy Acceleration in Geant4

G. Amadio 29 Sep 2022

G4Navigation needs Code Refactoring

This is screaming for an abstract base class, but due to this sort of switch statement appearing in several places in Geant4 geometry with slight variations, so it's not easy to refactor the code without invasive changes.



```
do
{
  // Determine `type' of current mother volume
  //
  targetPhysical = fHistory.GetTopVolume();
  if (!targetPhysical) { break; }
  targetLogical = targetPhysical->GetLogicalVolume();
  switch( CharacteriseDaughters(targetLogical) )
  {
    case kNormal:
      if(targetLogical->GetNoDaughters() > 2)
      {
        noResult = fBVHNav.LevelLocate(fHistory, fBlockedPhysicalVolume, fBlockedReplicaNo,
                                       globalPoint, pGlobalDirection, considerDirection,
                                       localPoint);
      }
      else
      {
        noResult = fnormalNav.LevelLocate(
          fHistory, fBlockedPhysicalVolume, fBlockedReplicaNo, globalPoint,
          pGlobalDirection, considerDirection, localPoint);
      }
      break;
    case kReplica:
      noResult = freplicaNav.LevelLocate(fHistory,
                                         fBlockedPhysicalVolume,
                                         fBlockedReplicaNo,
                                         globalPoint,
                                         pGlobalDirection,
                                         considerDirection,
                                         localPoint);
      break;
    case kParameterised:
      if( GetDaughtersRegularStructureId(targetLogical) != 1 )
      {
        noResult = fparamNav.LevelLocate(fHistory,
                                         fBlockedPhysicalVolume,
                                         fBlockedReplicaNo,
                                         globalPoint,
                                         pGlobalDirection,
                                         considerDirection,
                                         localPoint);
      }
      else  // Regular structure
      {
        noResult = fregularNav.LevelLocate(fHistory,
                                           fBlockedPhysicalVolume,
                                           fBlockedReplicaNo,
                                           globalPoint,
                                           pGlobalDirection,
                                           considerDirection,
                                           localPoint);
```

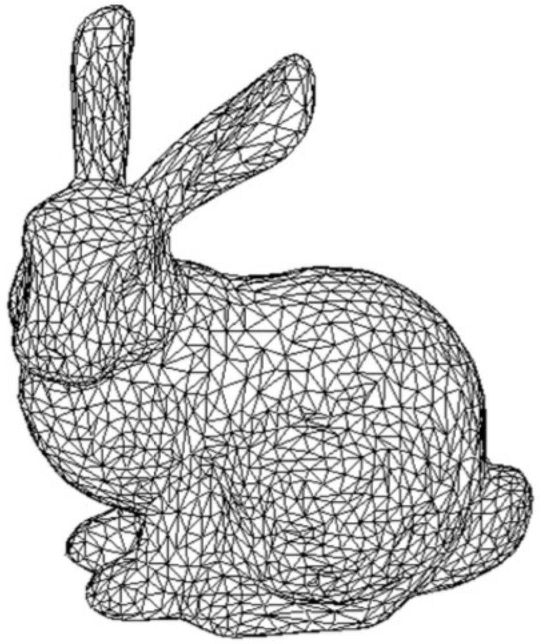
```
void
G4Navigator::LocateGlobalPointWithinVolume(const G4ThreeVector& pGlobalpoint)
{
#ifdef G4DEBUG_NAVIGATION
   assert( !fWasLimitedByGeometry );
   // Check: Either step was not limited by a boundary or
   //         else the full step is no longer being taken
#endif

   fLastLocatedPointLocal = ComputeLocalPoint(pGlobalpoint);
   fLastTriedStepComputation = false;
   fChangedGrandMotherRefFrame = false;  //  Frame for Exit Normal

   // For the case of Voxel (or Parameterised) volume the respective
   // Navigator must be messaged to update its voxel information etc

   // Update the state of the Sub Navigators
   // - in particular any voxel information they store/cache
   //
   G4VPhysicalVolume*  motherPhysical = fHistory.GetTopVolume();
   G4LogicalVolume*    motherLogical  = motherPhysical->GetLogicalVolume();

   switch( CharacteriseDaughters(motherLogical) )
   {
       case kNormal:
         break;
       case kParameterised:
         if( GetDaughtersRegularStructureId(motherLogical) != 1 )
         {
           // Resets state & returns voxel node
           G4SmartVoxelHeader* pVoxelHeader = motherLogical->GetVoxelHeader();
           fparamNav.ParamVoxelLocate( pVoxelHeader, fLastLocatedPointLocal );
         }
         break;
       case kReplica:
         // Nothing to do
         break;
       case kExternal:
         fpExternalNav->RelocateWithinVolume( motherPhysical,
                                              fLastLocatedPointLocal );
         break;
   }

   // Reset the state variables
   //   - which would have been affected
   //     by the 'equivalent' call to LocateGlobalPointAndSetup
   //   - who's values have been invalidated by the 'move'.
   //
   fBlockedPhysicalVolume = nullptr;
   fBlockedReplicaNo = -1;
   fEntering = false;
   fEnteredDaughter = false;  // Boundary not encountered, did not enter
```

Summary

- BVH implementation from VecGeom is working inside Geant4
  - Uses native Geant4 navigation functionality
  - G4BVHNavigator needs further work to support replicas and parameterized volumes
  - Can use BVH for normal volumes and voxel navigator for the rest with current implementation
  - Proper integration needs more code refactoring of core geometry classes
- Performance is comparable to G4VoxelNavigator for a full detector simulation
  - Not worth investing a lot of time for a small gain in performance
  - Performance benefit of BVH more visible with volumes with lots of children
  - Detector geometries have many logical volumes with only a few children, which limits benefit

Cubic volume and Surface area Computation

Evgueni Tcherniaev



$$V = \frac{1}{3}\left|\sum_{F}(P_F \cdot N_F)\,\mathrm{area}(F)\right|$$

where the sum is over faces F of the solid
P_F is a point on face F
N_F is a normal to F pointing outside of the solid
The dot $(\cdot)$ is the dot product

Geant4 Solids

- **Solids bounded by planes**
  G4Box, G4Para, G4Trd, G4Trap, G4Tet, G4Polyhedra, G4ExtrudedSolid, G4TessellatedSolid
  - no problem with analytical expressions
- **Solids with curved surfaces**
  a) G4Orb, G4Sphere, G4Tubs, G4Cons, G4Polycone, G4GenericPolycone, G4Torus
  - analytical expressions for these solids are well known or easy to derive
  b) G4CutTubs
  - special case
  c) G4EllipticalTube, G4EllipticalCone, G4Paraboloid, G4Hype, G4TwistedTube, G4TwistedBox, G4TwistedTrd
  - analytical expressions these solids are less known, or not so easy to derive on your own
  d) G4TwistedTrap, G4GenericTrap, G4Ellipsoid
  - no analytical expressions
- **Composite solids**
  Boolean solids: G4UnionSolid, G4SubtractionSolid, G4IntersectionSolid, G4MultiUnion; G4ScaledSolid, G4ReflectedSolid, G4DisplacedSolid
  - the MC method is used for the Boolean solids and for the computation of the surface are of G4ScaledSolid

A comprehensive walkthrough & review of the topic of ‘cubic volume’ and surface area.

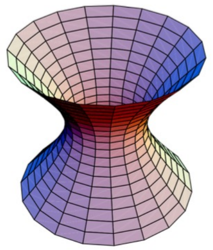
New analytical results, new methods.

Cubic volume and Surface area Computation

G4Hype(name, r_{in}, r_{out}, $stereo_{in}$, $stereo_{out}$, z)

The computation of the Cubic volume and Surface area is based on the expressions for One-Sheeted Hyperboloid:

https://mathworld.wolfram.com/One-SheetedHyperboloid.html



Cartesian equation: $\frac{x^2}{a^2} + \frac{y^2}{a^2} - \frac{z^2}{c^2} = 1$

Volume: $V = \frac{1}{3}\pi h(2a^2 + R^2)$

Surface area: $S = 2\pi a \left[\frac{h\sqrt{\cancel{(a^2+c^2)}[4c^4+(a^2+c^2)h^2]}}{4c^2} + \frac{c^2 \sinh^{-1}\left(\frac{h\sqrt{a^2+c^2}}{2c^2}\right)}{\sqrt{a^2+c^2}} \right]$

radius at $z = 0$, h – full height, R – radius at the top cross section

Note: There is a mistake in the expression for the Surface area (see crossed term), that is still present on the referred page despite being reported several months ago

Fast numerical integration

- Very often a 3D surface can be represented in a parametric form:
$$\vec{r} = \vec{r}(s,t)$$
- The surface area can be computed with the following double integral:
$$S = \int_c^d \int_a^b \left| \frac{\partial \vec{r}}{\partial s} \times \frac{\partial \vec{r}}{\partial t} \right| ds dt$$
- If the double integral cannot be found analytically, then we can evaluate it by numerical integration - the surface is divided into small quadrangles (e.g. 100x100) and the surface area is calculated as a sum of the areas of the quadrangles.
- We can try to also do something better – try to find an analytical expression for the inner integral, and then evaluate the outer integral numerically, the evaluation will be faster and more precise:
$$S = \int_c^d F(t) dt$$
- For comparison:
  - The MC method – 1 M random points, precision 10^{-2}
  - Ordinary numerical integration – 100x100 steps along two parameters, precision 10^{-4}
  - Fast numerical integration – 1000 steps along one parameter, precision 10^{-6}