WP2 / UNIFIED SOLIDS LIBRARY

Gabriele Cosmo, PH/SFT

Outline

- Motivations & outlook
- Achieved status
- Progress made in the last months
  - Consolidated implementation of existing shapes
  - Enhanced testing suite
- Synergy with VecGeom developments
- Future evolution

Motivations for a common solids library

- Optimize and guarantee better long-term maintenance of ROOT and Geant4 solids libraries
- Create a single high quality library to replace solid libraries in Geant4 and ROOT
  - Starting from what exists today in Geant4 and ROOT
  - Adopt a single type for each shape
  - Significantly optimize (Multi-Union, Tessellated Solid, Polyhedra, Polycone)
  - Reach complete conformance to GDML solids schema
- Create extensive testing suite

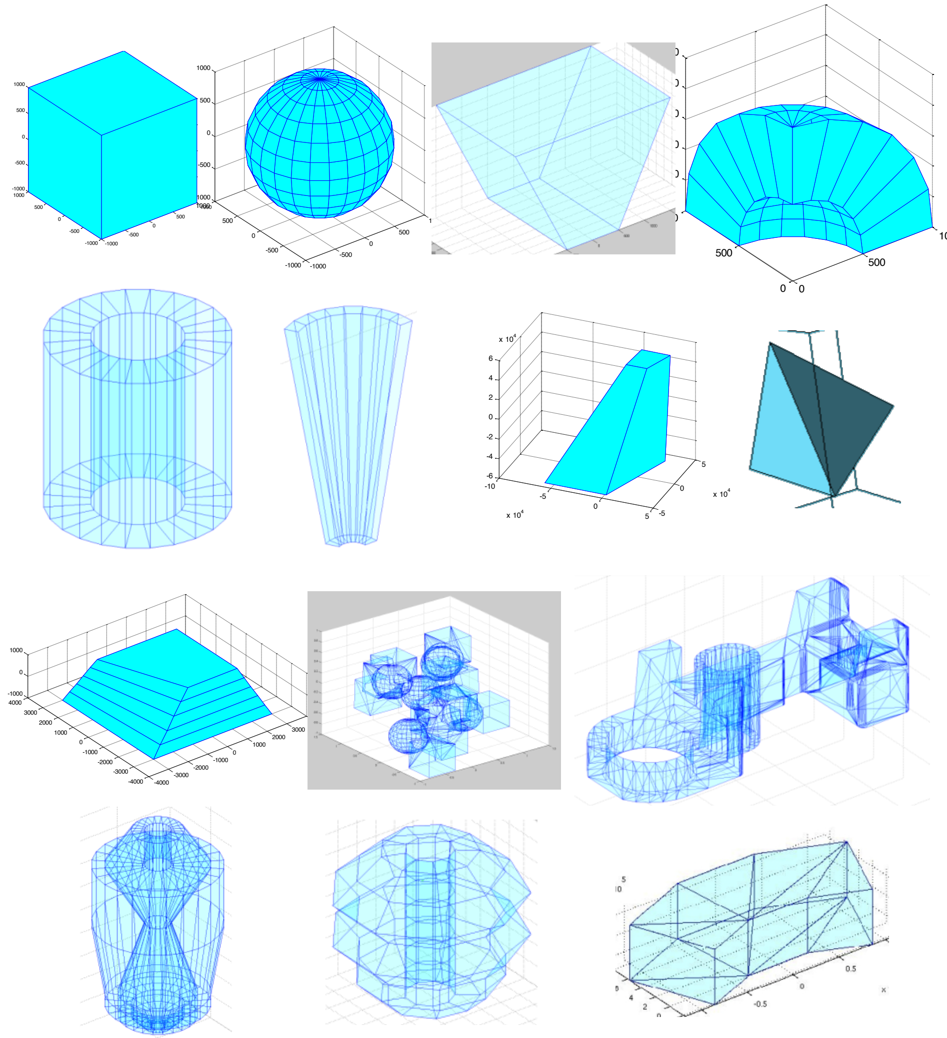
Resources involved

- Contributions from:
  - John Apostolakis (PH/SFT)
  - Gabriele Cosmo (PH/SFT)
  - Marek Gayer (AIDA Fellow PH/SFT from 1/7/2011 to 1/10/2013)
  - Andrei Gheata (ALICE)
  - Jean-Marie Guyader (CERN Summer Student until 31/8/2011)
  - Tatiana Nikitina (PH/SFT)

- 0.4 FTEs since Marek Gayer left

Solids implemented so far

- Box
- Orb
- Trapezoid
- Sphere (+ sphere section)
- Tube (+ cylindrical section)
- Cone (+ conical section)
- Generic trapezoid
- Tetrahedron
- Arbitrary Trapezoid
- **Multi-Union**
- Tessellated Solid
- **Polycone**
- Generic Polycone
- Polyhedra
- Extruded solid



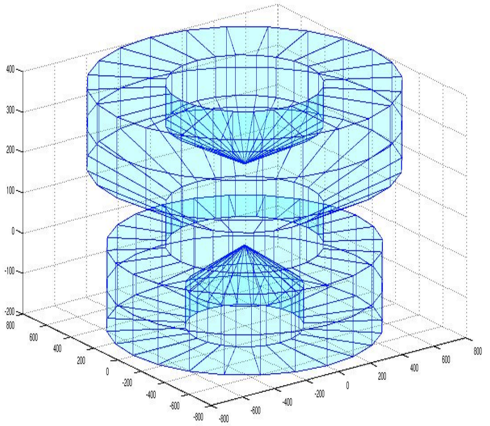
Status of USolids library

- Library in the current form distributed as optional module in the latest Geant4 release 10.1
  - Firstly introduced in Geant4 release 10.0 last year in reduced form
  - Now possible to use it also as external independent library
  - Expecting feedback !
  - Validation of shapes on realistic detector geometries ongoing
- Testing suite further extended for performance/accuracy measurements
- Code available in the AIDA SVN repository
  - Using standard AIDA CMake setup for build/installation
- Documentation available from web
  - http://aidasoft.web.cern.ch/USolids

Recent Progress

Revised Ordinary Polycone

- UPolycone: composite shape constructed from sections of Tubs and Cons
  - With spatial optimisation over Z
  - Excellent scalability over the number of Z sections
  - Significant performance improvement
- Added missing methods:
  - GetPointOnSurface, Capacity, SurfaceArea
  - Visualization in wrappers
- Reviewed implementation of main methods:
  - Corrected treatment of Convexity and Safety

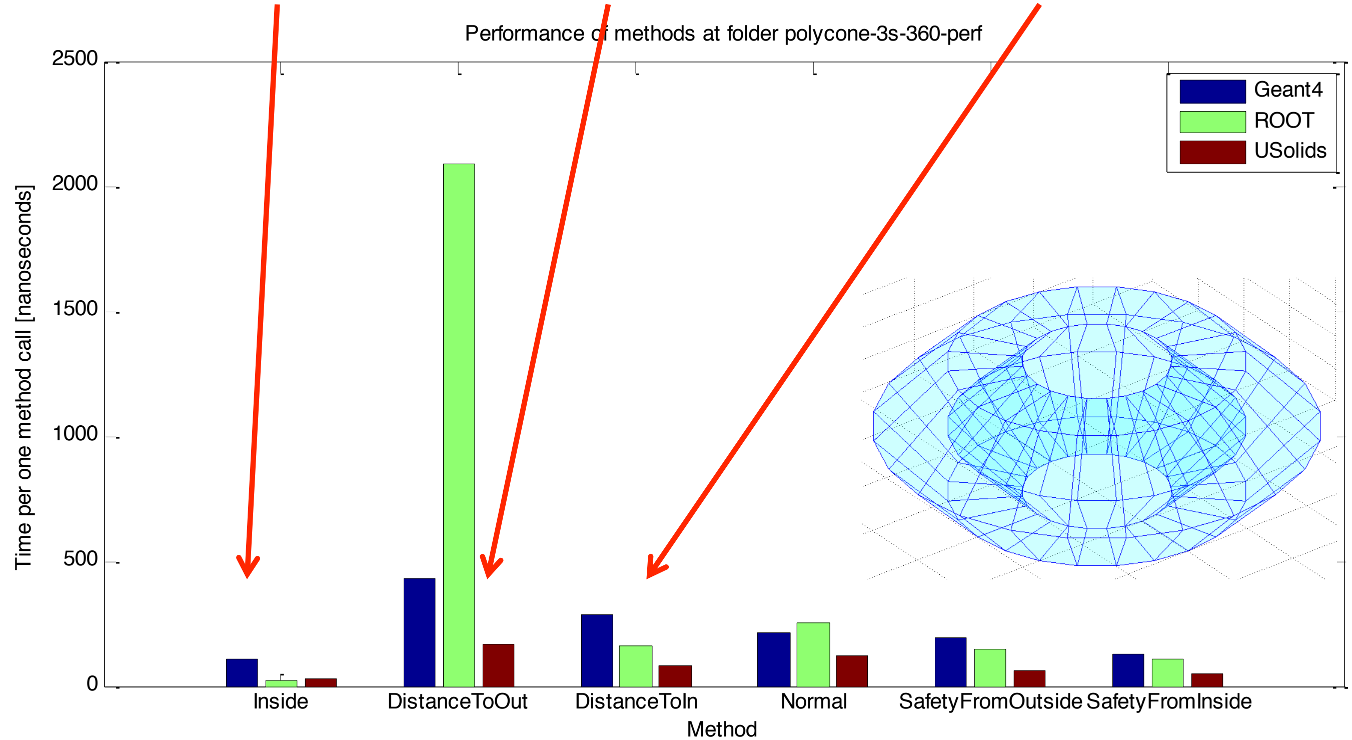


Revised UPolycone performance

example: 3 Z-sections

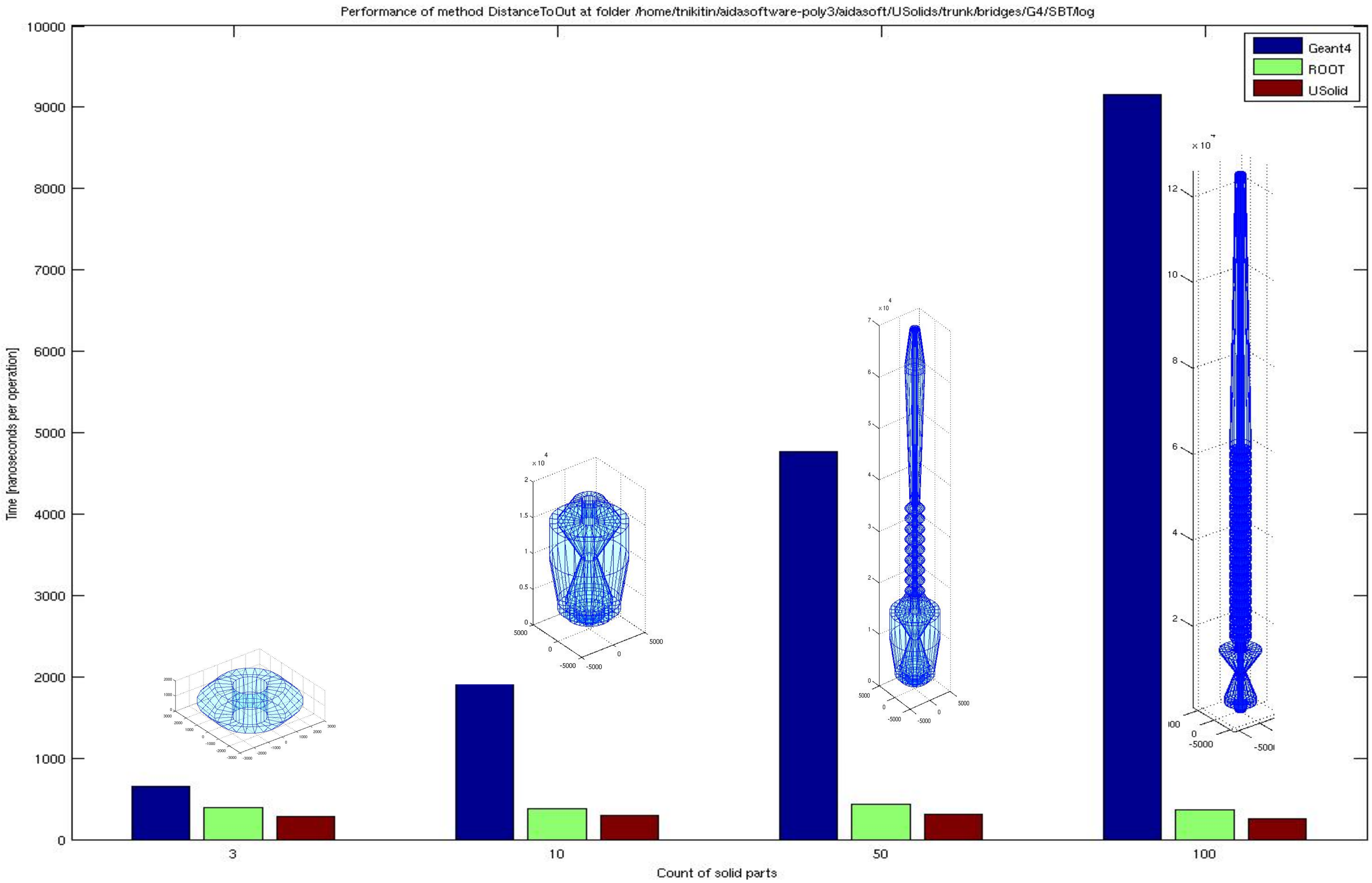
- Speedup factor **3.3x** vs. Geant4, **7.6x** vs. Root
  - for most performance critical methods, i.e.:

Inside *DistanceToOut* *DistanceToIn*



Revised UPolycone performance

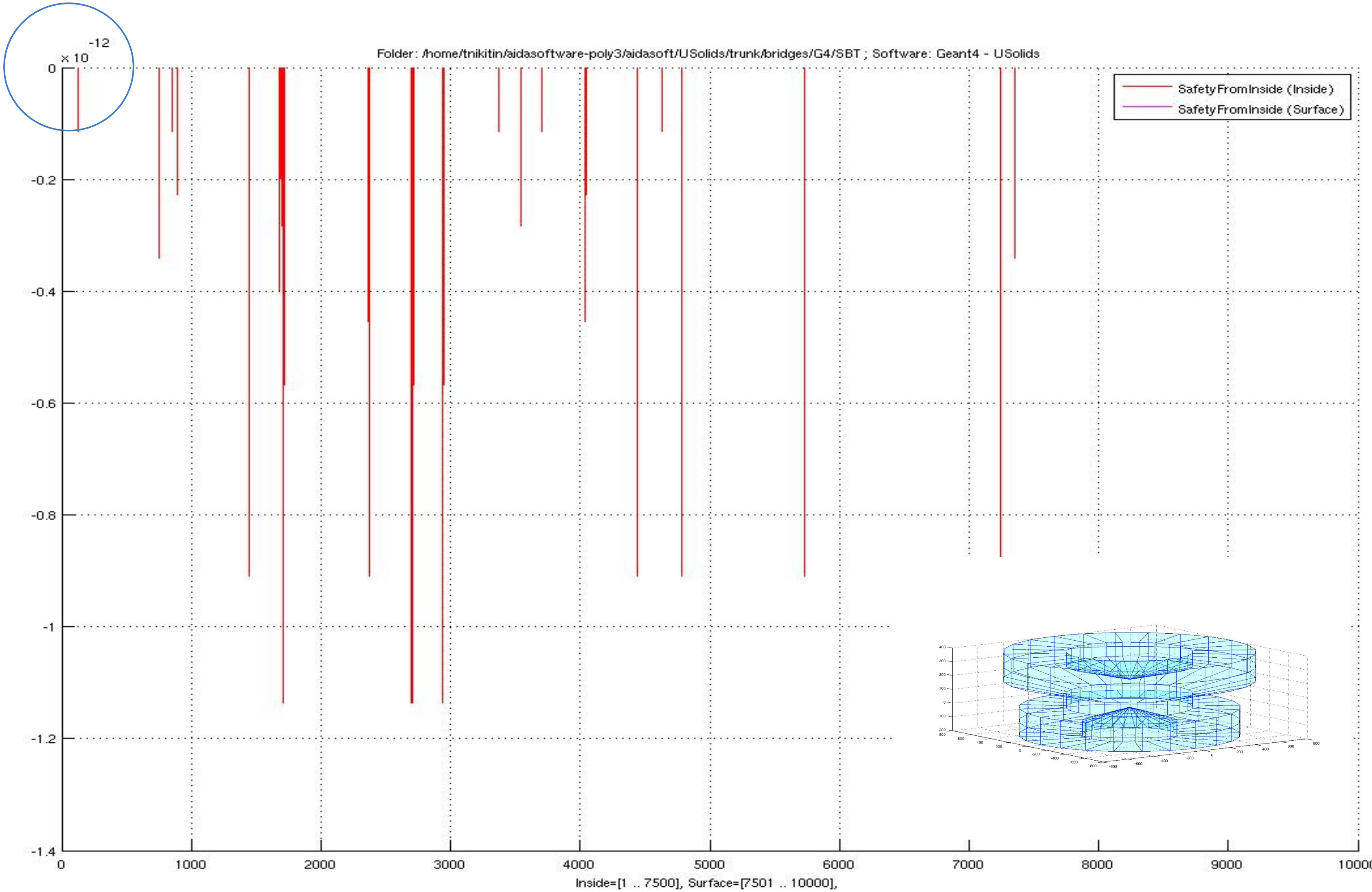
Scalability for DistanceToOut()



NOTE: *Geant4 poor performance scalability (high number of Z sections) due to absence of spatial optimization*

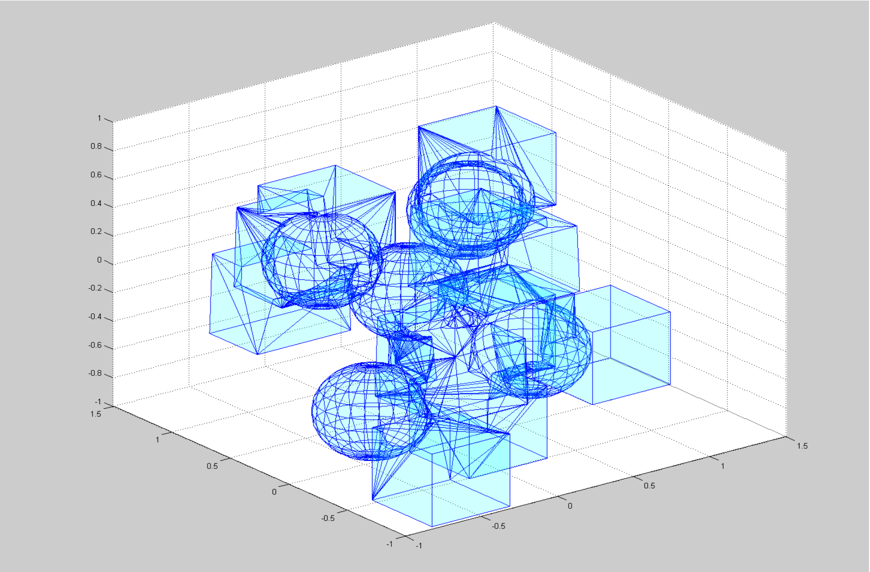
Revised UPolycone:

Differences in SafetyFromInside() USolids-Geant4



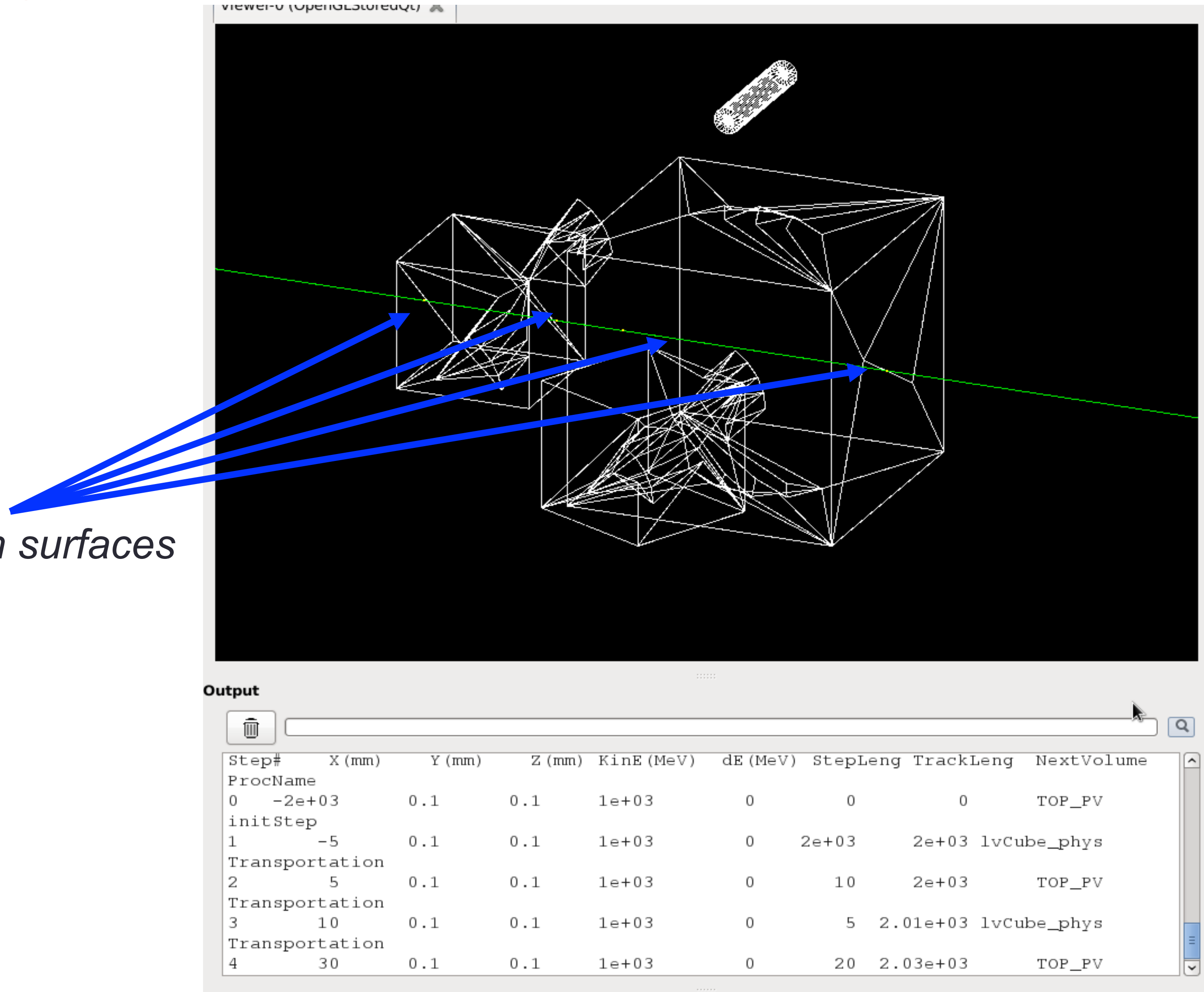
Revised MultiUnion structure

- UMultiUnion structure: representing a union of many [displaced] solids
  - adopting voxelisation technique for optimisation on location of components
- Added missing methods:
  - GetPointOnSurface, Capacity, SurfaceArea
  - Visualization in wrappers
- Corrected treatment of transformations
- Integration in GDML
  - Ability to import/export as GDML (3.1.1) in Geant4



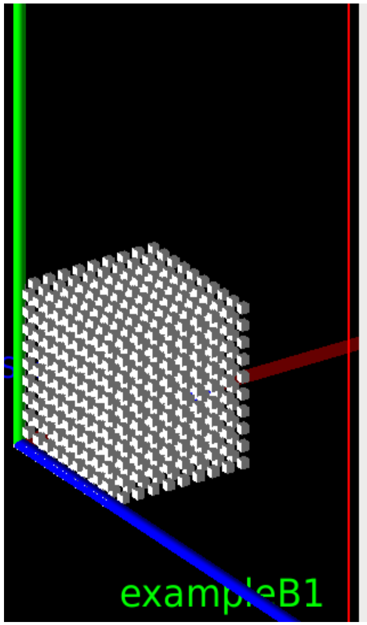
UMultiUnion structure:

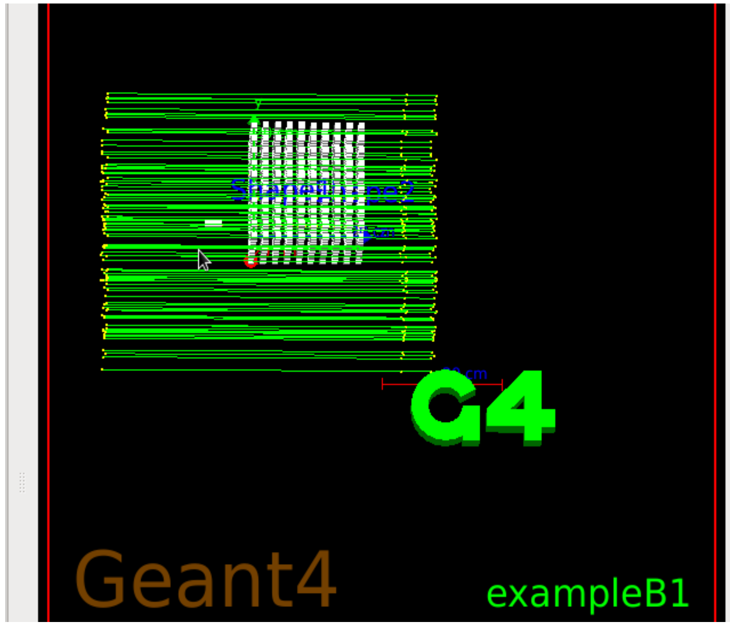
Visualization in Geant4



Intersection of ray with surfaces

UMultiUnion structure:
Voxelisation studies

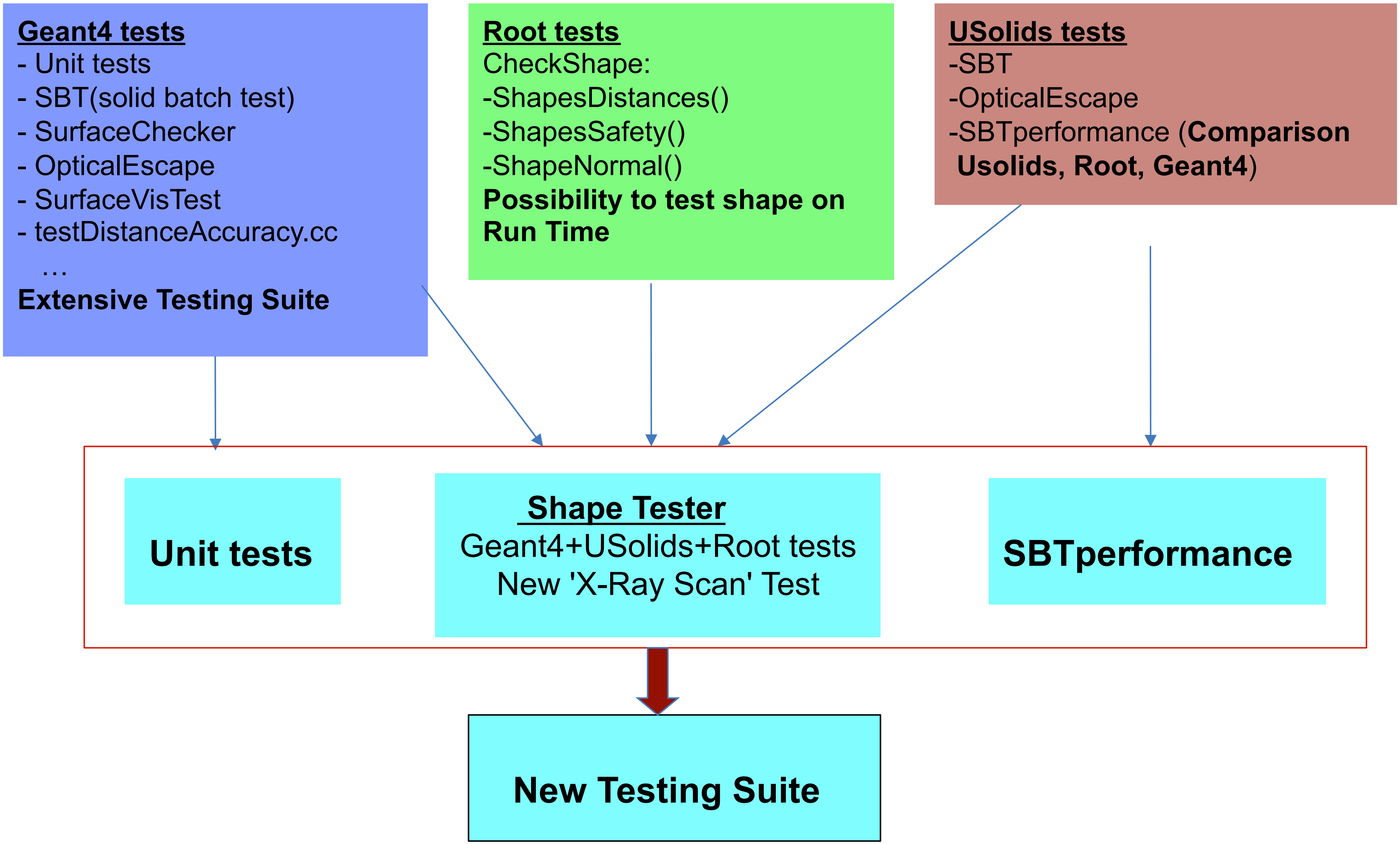




- Test case: a regular structure of boxes as UMultiUnion or as loop of simple placements
- Real transportation of 'geantinos'

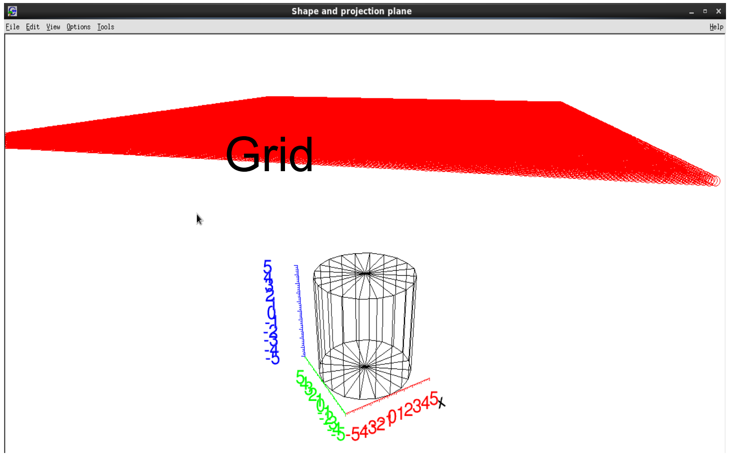
	Time for 125 solids(sec)	Time for 1000 solids(sec)	Ratio
G4UMultiUnion	10.9	13.25	1.22
Loop of Placements	12.43	17.58	1.41
Ratio	1.14	1.33	

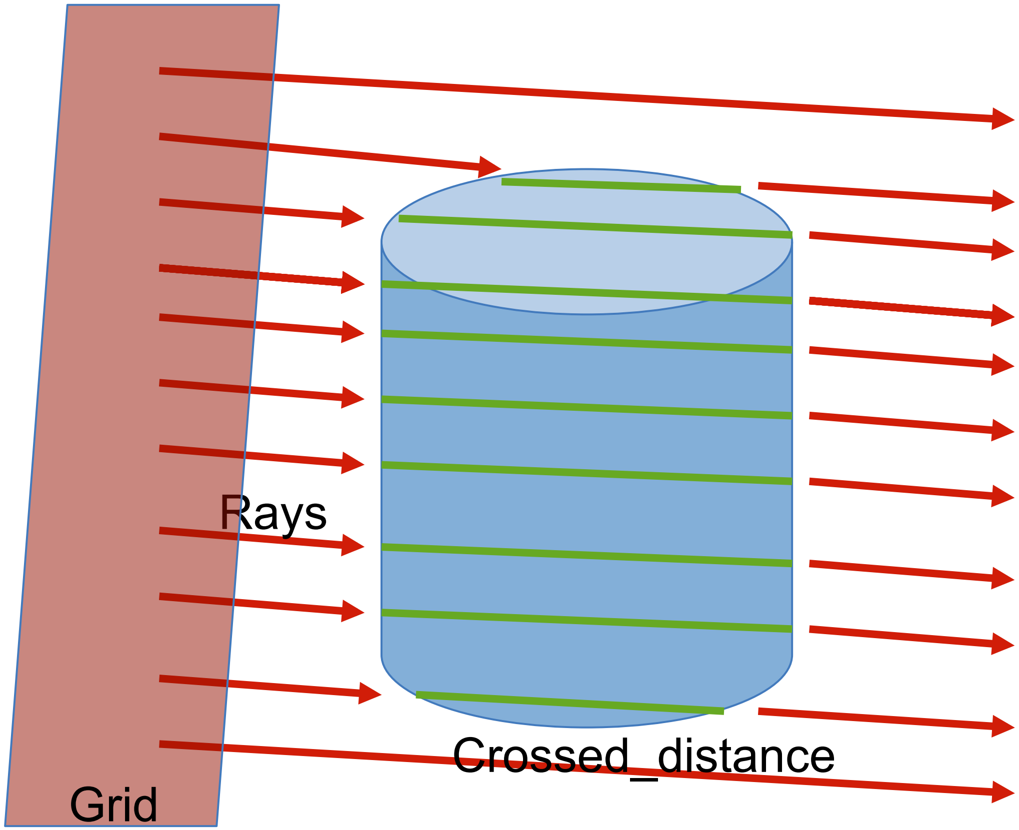
Enhanced testing suite



Extended testing suite

X-Ray Scan



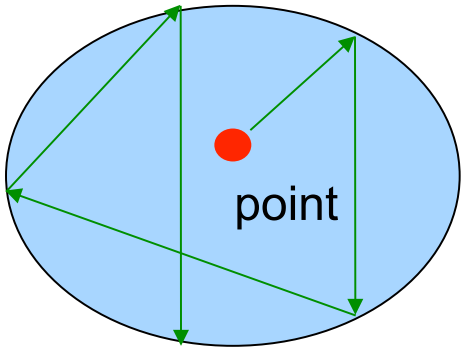


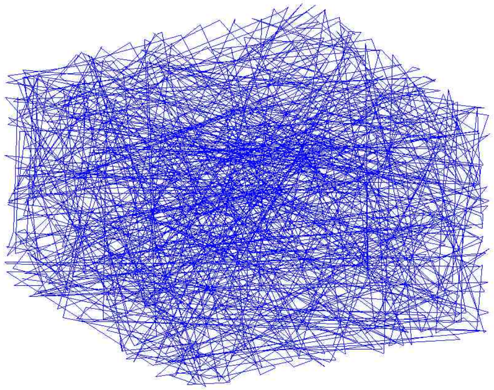
- Estimated Volume = ∑(distance × cell-area)
- Error = Analytic Volume − Estimated Volume

Scan can be done for different angles in Theta and Phi

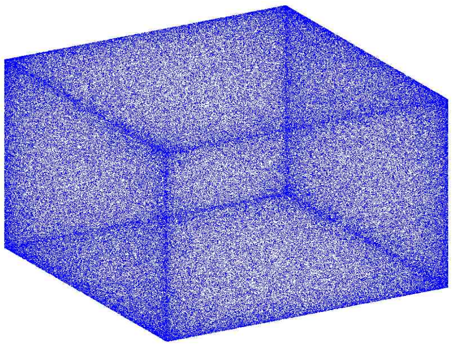
Extended testing suite

Optical Escape

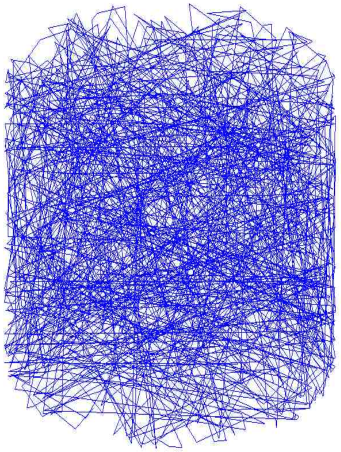




UBox(Rays)



UBox(Created points)

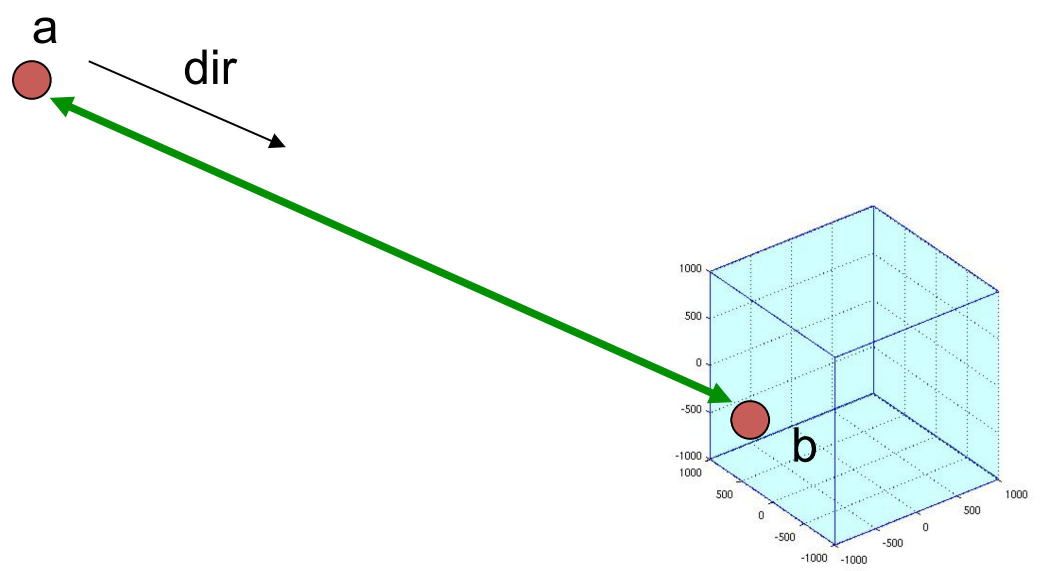


UTubs(Rays)

- Improved original "Optical Escape" test to use random reflection on surface
  - Better distribution of points

Extended testing suite

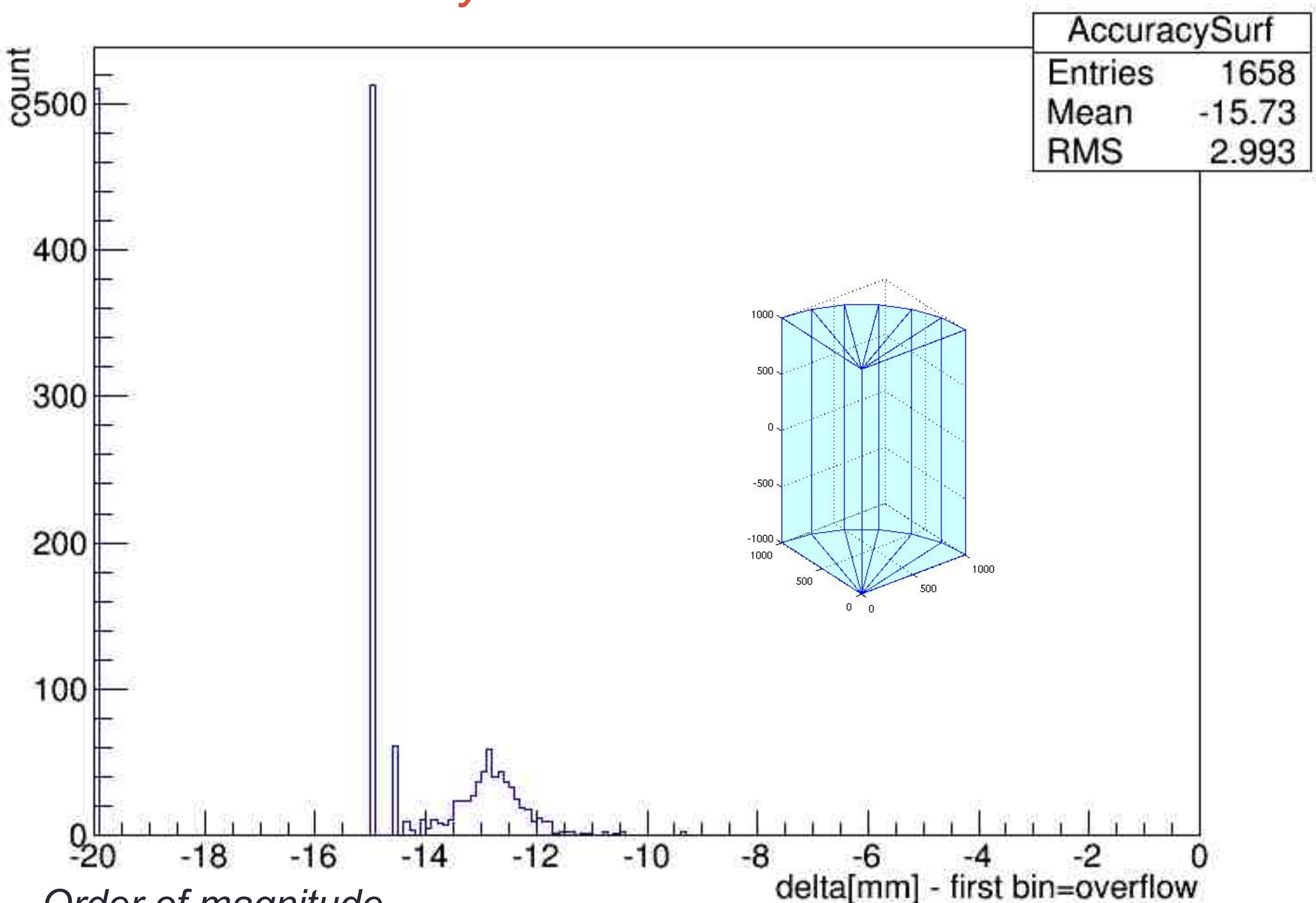
DistanceToIn accuracy



- Point ‘b’ located on surface
- Accuracy = |DistanceToIn(a,dir) − |a - b||

Extended testing suite

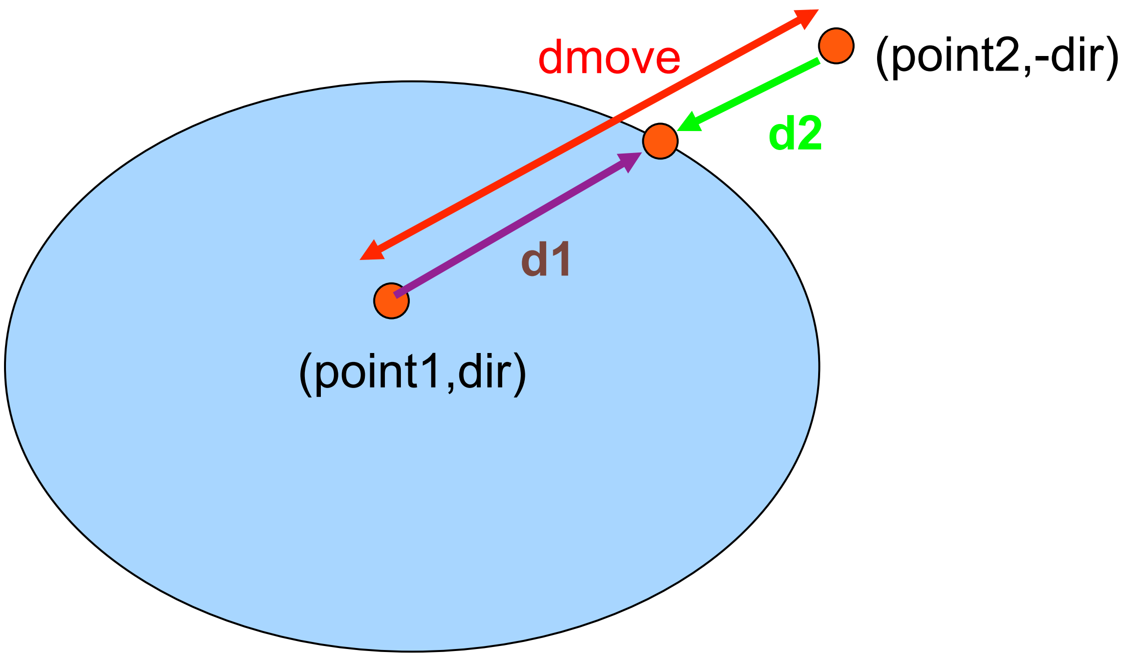
DistanceToIn accuracy test: UTubs



Order of magnitude

Extended testing suite

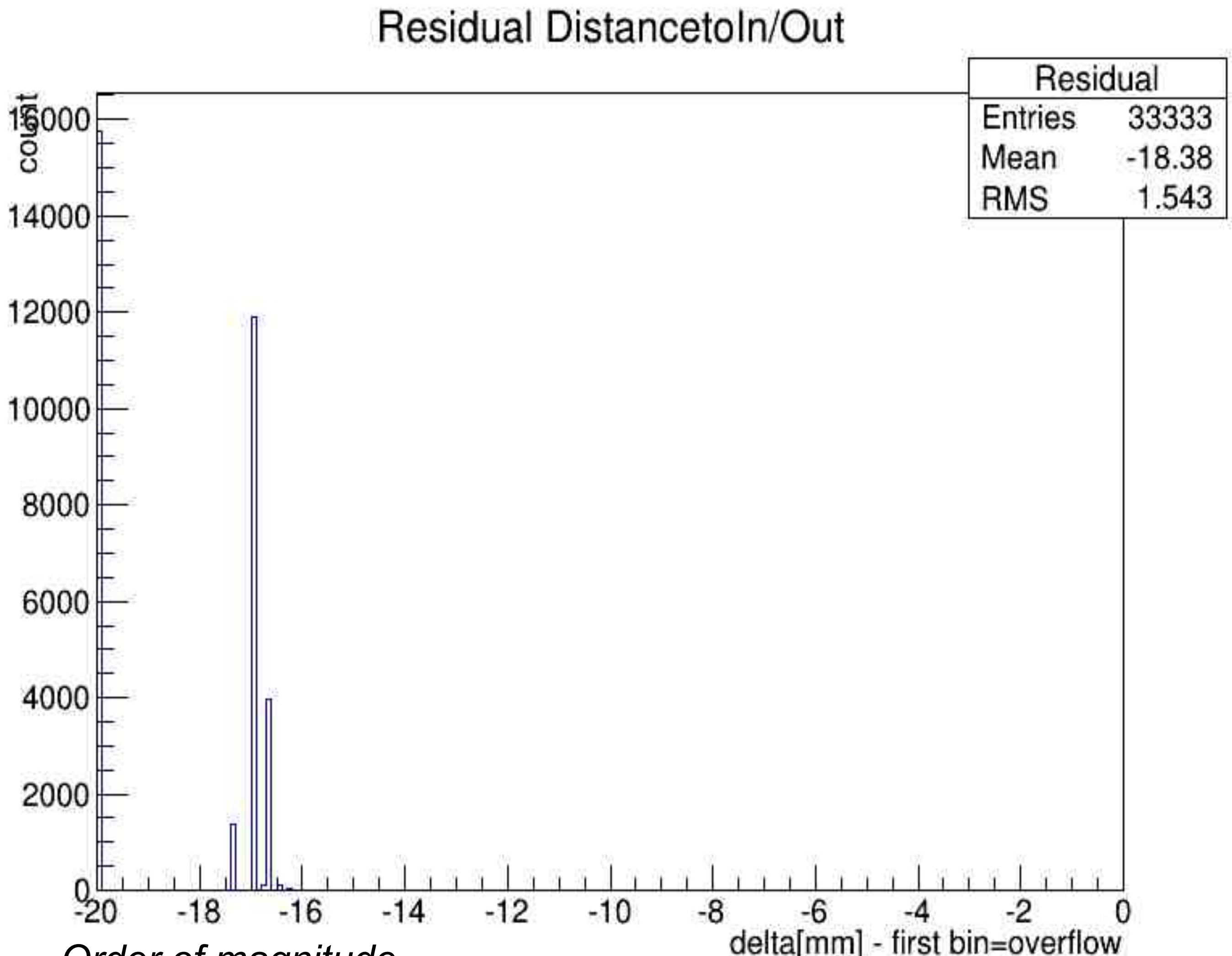
DistanceToIn()/DistanceToOut() accuracy



Difference = max (| dmove – d1 – d2 |)

Extended testing suite

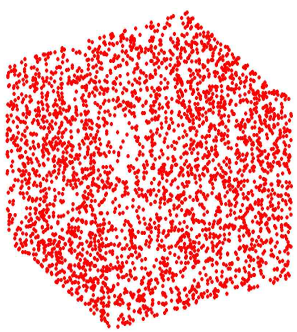
DistanceToIn()/DistanceToOut() accuracy test: UBox



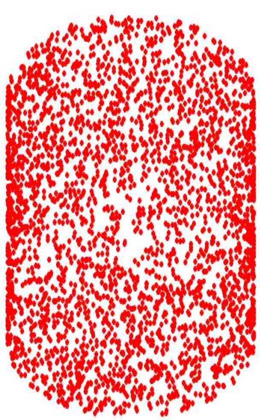
Order of magnitude

Extended testing suite

Consistency checks. Example: Test SurfacePoints



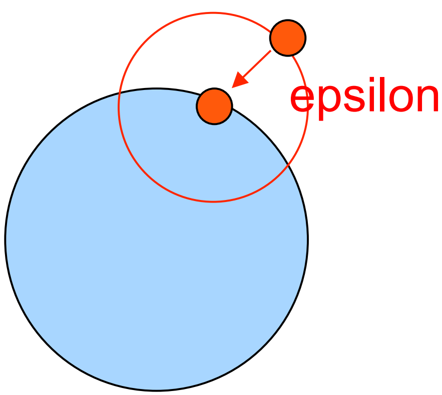
GetPointOnSurface()
UBox



GetPointOnSurface()
UTubs

- Test consistency between
  - Inside() and GetPointOnSurface()
    - *Inside(GetPointOnSurface()) == kSurface*
  - DistanceToIn() and DistanceToOut()
    - *Both cannot be zero*

- Check accuracy of DistanceToIn() and DistanceToOut()



- *The new testing suite forms the basis for applying full systematics testing on all shapes and their combinations*
  - *Ongoing work*

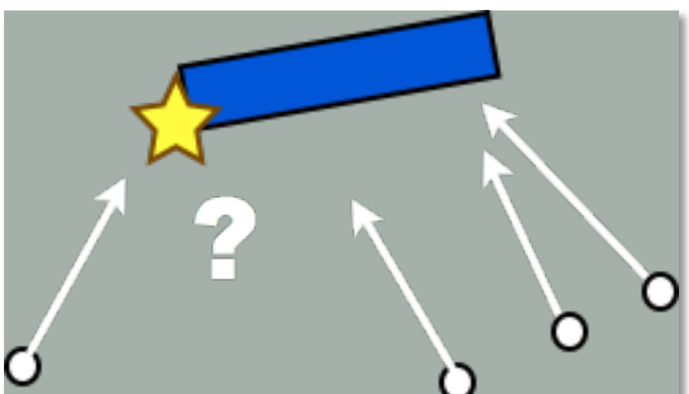
VecGeom

Ongoing Developments on Primitives

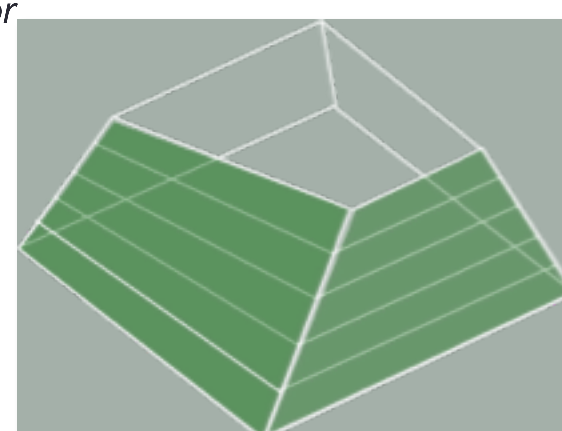
VecGeom

- Started as feasibility study of vectorization for geometry
  - Part of the development going on for the *Geant Vector Prototype*
  - Extending signatures of classes to enable use of vectorisation
  - Review algorithms on all developed shapes to efficiently apply vectorisation and strong code specialization
  - Also submitted as AIDA2 proposal
- Geometry primitives code development to be considered as long-term evolution of USolids
  - To replace/complement the current implementation in USolids
  - Developed back-to-back with USolids as independent library
  - Sharing same interfaces

Internal algorithm vectorization: loop over lateral planes for distance calculation

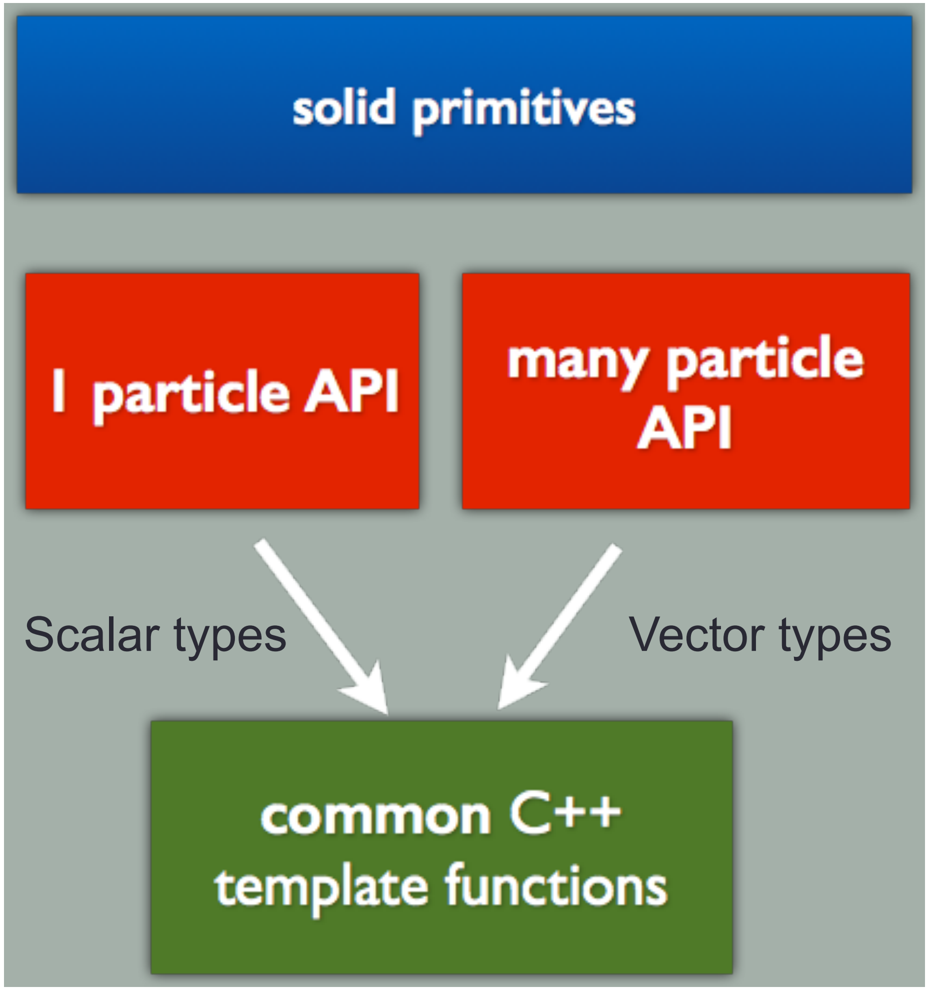


Vector signatures: "parallel" collision detection



VecGeom

The approach

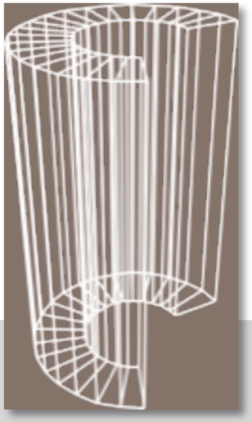


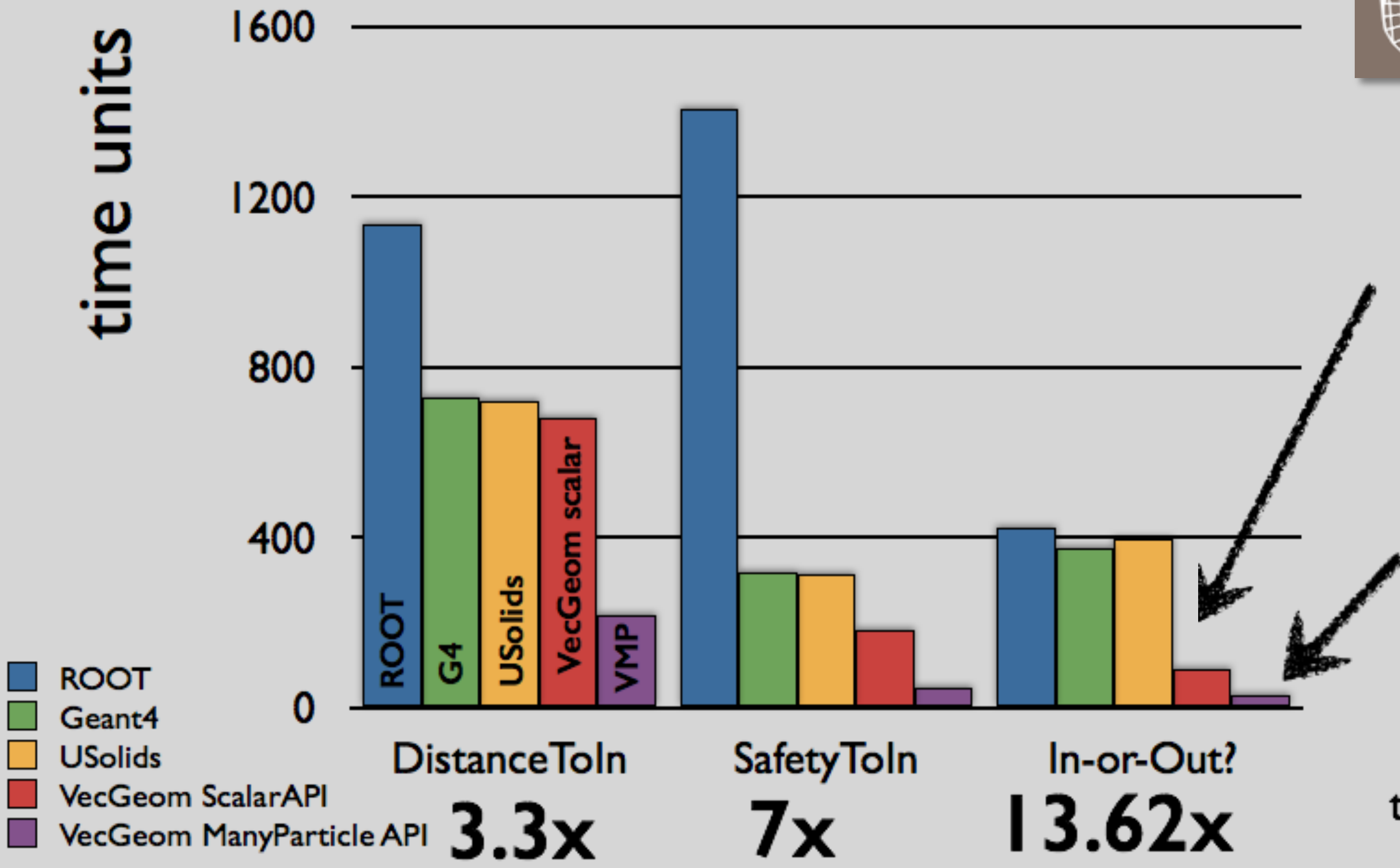
efficient SIMD vectorization achieved by using vector libraries (e.g. Vc) providing C++ approach to explicit vectorization

template C++ programming to reduce code multiplication due to proliferation of interfaces

Performance case study

Tube segment





improved scalar performance

- improved algorithms (avoid atan2)
- template shape specialization

excellent SIMD vector performance

total speedup cmp to USolids

Current Status & Plans

USolids / VecGeom plan of work

- Missing shapes from the standard set
  - Cut Tube, Torus, Ellipsoid, Elliptical Tube, Elliptical Cone, Hyperboloid, Paraboloid, Twisted shapes
  - Boolean compositions: Simple Union, Subtraction, Intersection
- Currently under development through VecGeom
  - Torus, Ellipsoid, Hyperboloid, Paraboloid
  - Boolean composite shapes with templated signature
- Integration of VecGeom shapes in USolids library
  - Adiabatic replacement of current USolids shapes
- Further enhancement to the testing suite
  - Creation of *Comparison Solid* for comparison of methods of two shapes during run-time
  - Testing and benchmarking on complex geometries

Thanks