Recent Developments in USolids/VecGeom
Status + Plans

Sandro Wenzel / CERN-PH-SFT

Geant4 collaboration meeting, Fermilab, 30.09.2015

Outline

- Status of USolids - AIDA phase 1
- Motivation + Status of VecGeom - AIDA phase 2
  - What is VecGeom ??
  - Status of shape implementations in VecGeom
  - New features; improvements and some ideas
- Plans

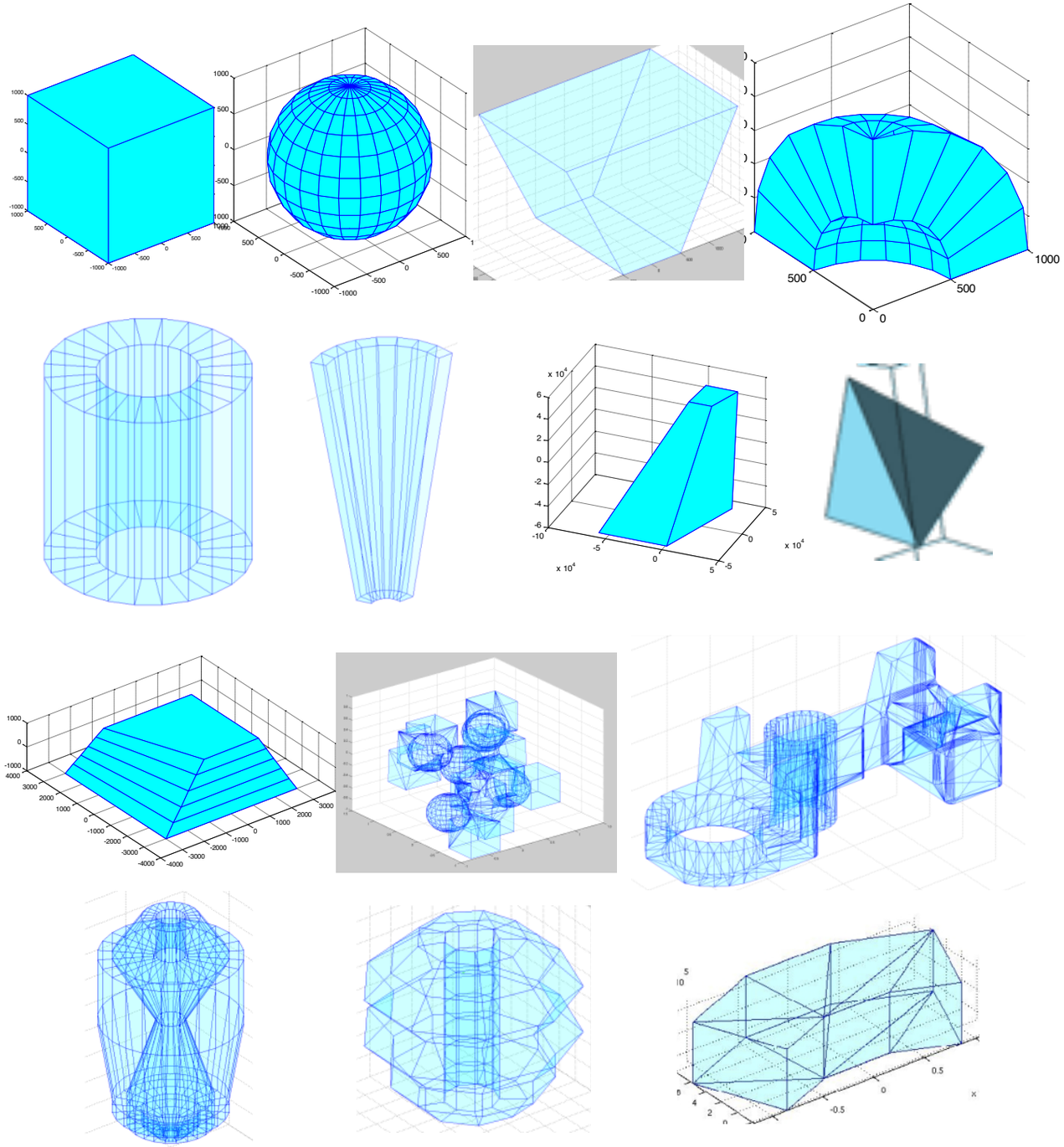
Motivation for original AIDA USolids

- Optimize and guarantee better long-term maintance of Geant4 and ROOT **solid** 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 complex shapes such as Polycone, Polyedra, Multi-Union, Tesselated solid
  - Reach complete conformance to GDML solids
- Create extensive testing suite

quoted from Gabriele Cosmo „AIDA final meeting slides“

USolids implementation status

- 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



taken from Gabriele Cosmo „AIDA final meeting slides“

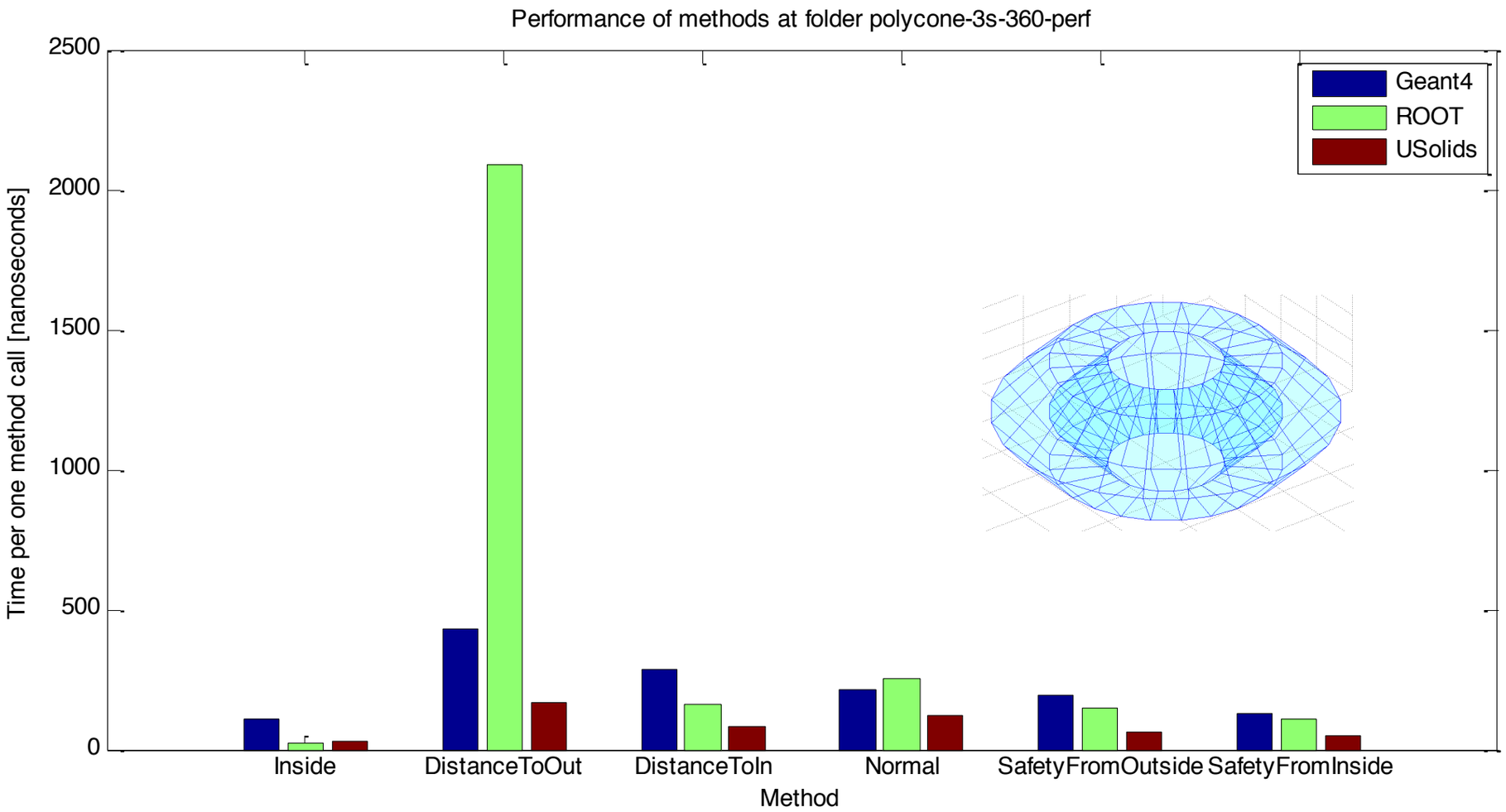
Reminder of some highlights of USolids

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*

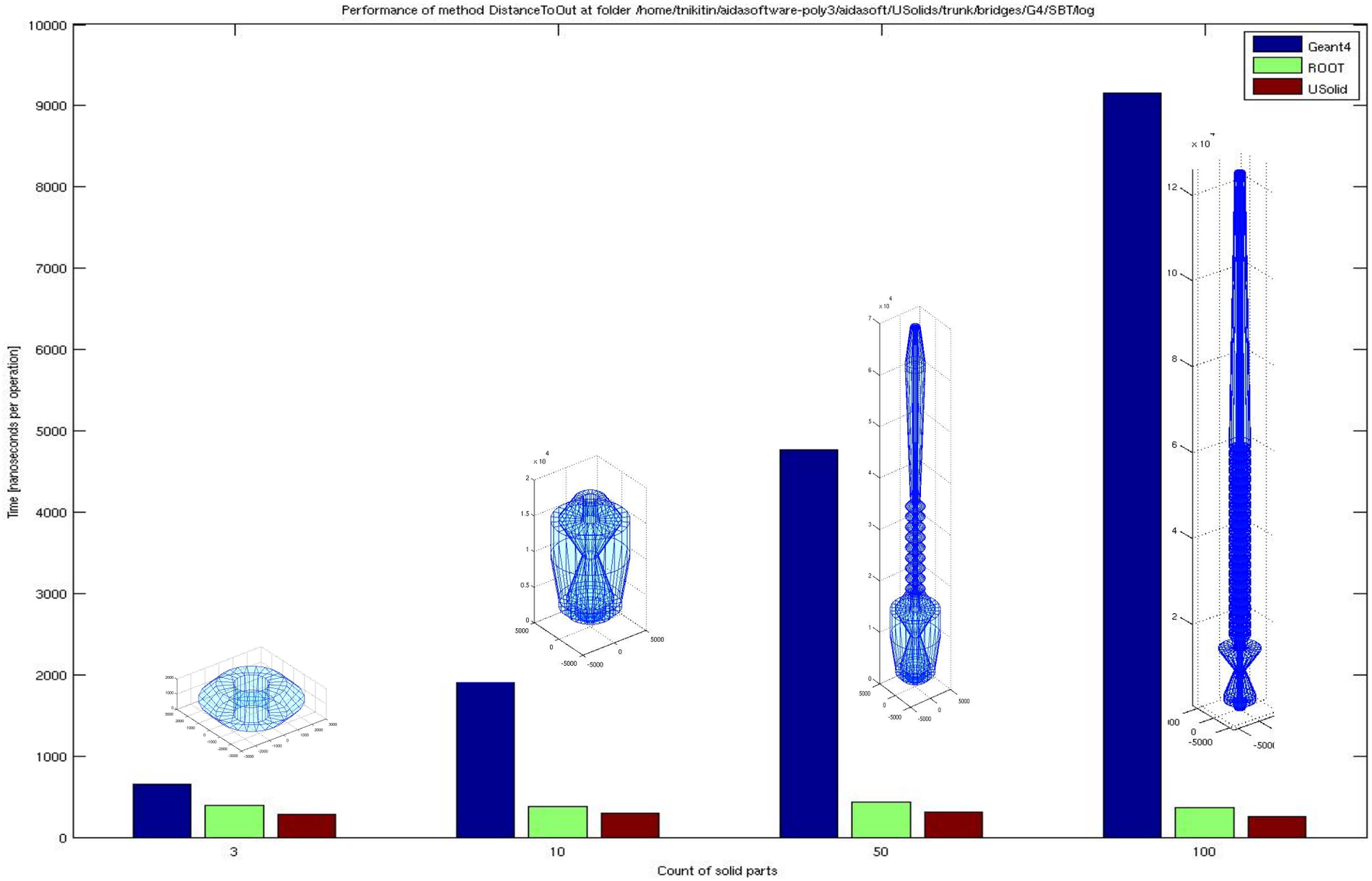


taken from Gabriele Cosmo „AIDA final meeting slides“

Improved scalability of USolids polycone

Revised UPolycone performance

Scalability for DistanceToOut()



taken from Gabriele Cosmo „AIDA final meeting slides“

USolids/Geant4 integration

- It is today possible to run Geant4 simulations with USolids shapes replacing Geant4 shapes (seamless to user)
  - Geant4 10.1. ships USolids internally
  - optionally one may also compile against external USolids installation
- Geant4 release 10.2. will remove internal module in favour of compiling/linking against external USolids/VecGeom library
  - less code duplication
- USolids source code repository: gitlab.cern.ch/VecGeom/VecGeom

see also talk by Guilherme Lima on USolids/VecGeom integration into G4

From USolids to VecGeom

New requirements came up ... which were not addressed by USolids during the AIDA I phase:

- not designed to target use of external/internal SIMD vectorization to further speed up the algorithms (becoming an absolute necessity nowadays)
- no interface to process many particles at once (see Geant-V initiative)
- no library support for GPUs
- design based on traditional C++90ish and no use of modern HPC features („templates“) which could further improve performance

From USolids to VecGeom

- VecGeom **is USolids** augmented with more functionality and usable on more platforms:

VecGeom = Evolved USolids

+ Many-Particle API

+ Geometry Model / Navigation

Vec / Geom = SIMD/GPU support / complete geometry modeler

- The VecGeom + USolids development teams are identical
- obtained AIDA2 funding as continuation of AIDA (targeting vectorization of USolids)
- official repository at gitlab.cern.ch/VecGeom/VecGeom

Main components of VecGeom

AIDA1 USolids

„Shapes“

Box, Tube,...

scalar API

double DistanceToOut(Vector3D const &p, Vector3D const &d)

Geometry Modeller

LogicalVolume

PlacedVolume

Transformations

Navigation

NavigationState

Navigator

scalar API

double ComputeStep(Vector3D, Vector3D)

Main components of VecGeom

AIDA1 USolids

„Shapes“

Box, Tube,...

scalar API

vector API

double DistanceToOut(Vector3D const &p, Vector3D const &d)

void DistanceToOut(„multitrack-interface“)

Geometry Modeller

LogicalVolume

PlacedVolume

Transformations

Navigation

NavigationState

Navigator

scalar API

vector API

double ComputeStep(Vector3D, Vector3D)

void ComputeStep(...“multi-track“ interface...)

This talk: focus on aspects relevant for Geant4 (single track functionality)

Tomorrow: more details on vector-API relevant for Geant-V

Shape development status

Shape	USolids	VecGeom
Box	yes	yes
Trap + Trd	yes	yes
Tube[s]	yes	yes
Cone[s]	yes	yes
GenericTrap/Arb8	yes	(yes)
Tet	yes	
Polycone	yes	yes
Polyhedron	yes	yes
Torus		yes
Parallelepiped		yes
Extruded solid	yes	
MultiUnion	yes	
Tesselated Solid	yes	
Composites		yes
Templat. Composites		(yes)
Hype,Ellipsoid, Parab		yes
Orb/Sphere	yes	yes
... the rest ...		

the rest is „Eltu, Twisted[*], ScaledShape, ...“

Shape development status

Shape	USolids	VecGeom	SIMD acceleration: Internal SIMD	SIMD acceleration: Multi-Track SIMD
Box	yes	yes		yes
Trap + Trd	yes	yes		yes
Tube[s]	yes	yes		yes
Cone[s]	yes	yes		(incomplete)
GenericTrap/Arb8	yes	(yes)	(yes)	(yes)
Tet	yes		(targeted)	
Polycone	yes	yes	(targeted)	
Polyhedron	yes	yes	yes	
Torus		yes		yes
Parallelepiped		yes		yes
Extruded solid	yes		(targeted)	
MultiUnion	yes		(targeted)	
Tesselated Solid	yes		(targeted)	
Composites		yes		
Templat. Composites		(yes)		(yes)
Hype,Ellipsoid, Parab		yes		yes
Orb/Sphere	yes	yes		yes
... the rest ...				

the rest is „Eltu, Twisted[*], ScaledShape, ...“

Shape development status

Shape	USolids	VecGeom	SIMD acceleration: Internal SIMD	SIMD acceleration: Multi-Track SIMD
Box	yes	yes		yes
Trap + Trd	yes	yes		yes
Tube[s]	yes	yes		yes
Cone[s]	yes	yes		(incomplete)
GenericTrap/Arb8	yes	(yes)		(yes)
Tet	yes			
Polycone	yes	yes		
Polyhedron	yes			
Torus				yes
Parallelepiped				yes
Extruded solid			(targeted)	
MultiUnion			(targeted)	
Tesselated Solid			(targeted)	
Composites		yes		
Templat. Composit		(yes)		(yes)
Hype,Ellipsoid, Parab		yes		yes
Orb/Sphere	yes	yes		yes
... the rest ...				

Disclaimer: Validation of VecGeom shapes not finished ...

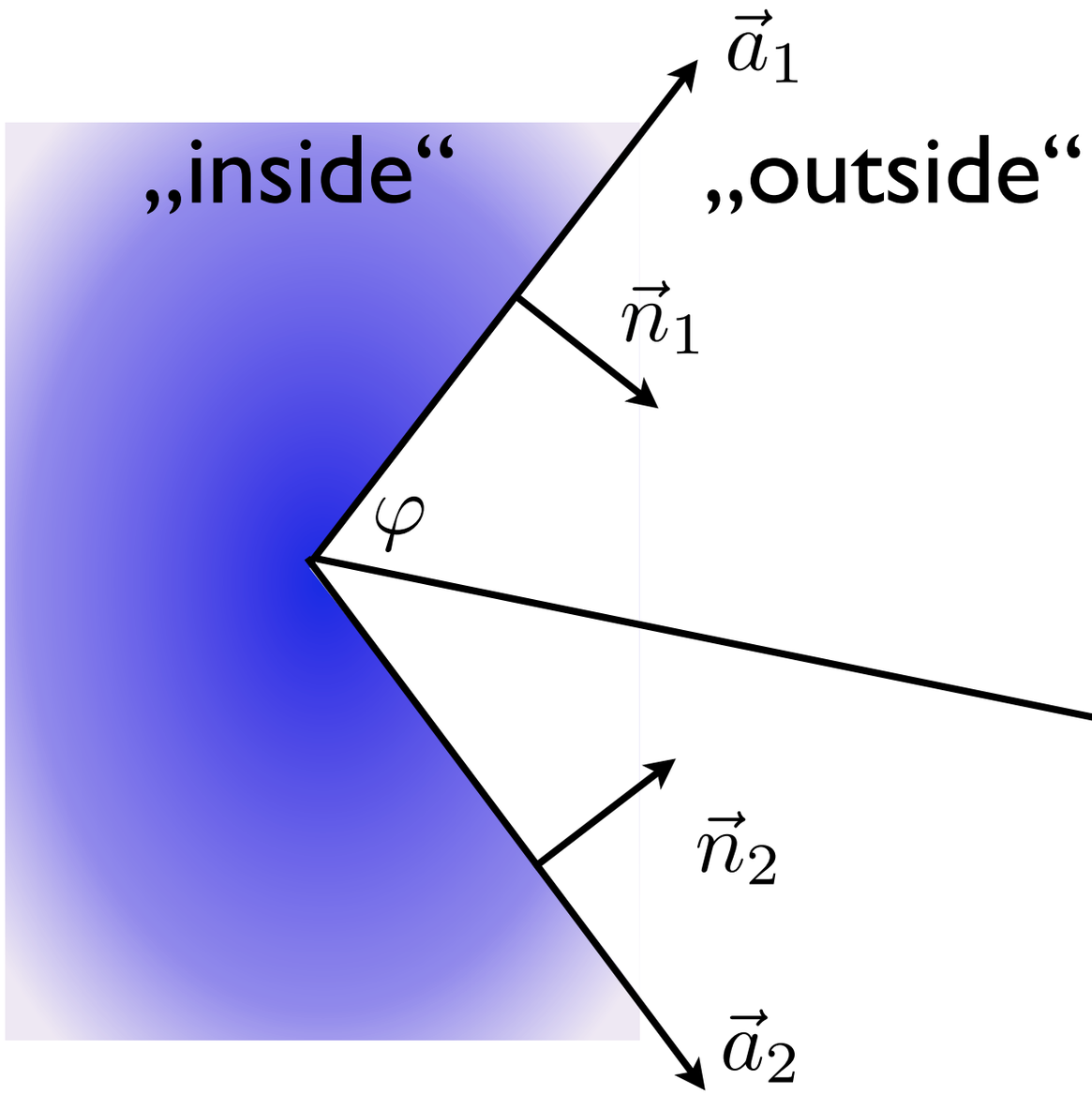
the rest is „Eltu, Twisted[*], ScaledShape, ...“

New Features of VecGeom shapes

- More interfaces (example: offer now both „Contains“ and „Inside“ to satisfy both G4 and ROOT/TGeo requirements better)
- Algorithmic improvements
- Pushing logical decomposition started in USolids further
- Explicitly targeting inner SIMD acceleration of algorithms
- Template shape specializations
- Placement shape specialization
- ....

Improved decomposition + algorithms (example)

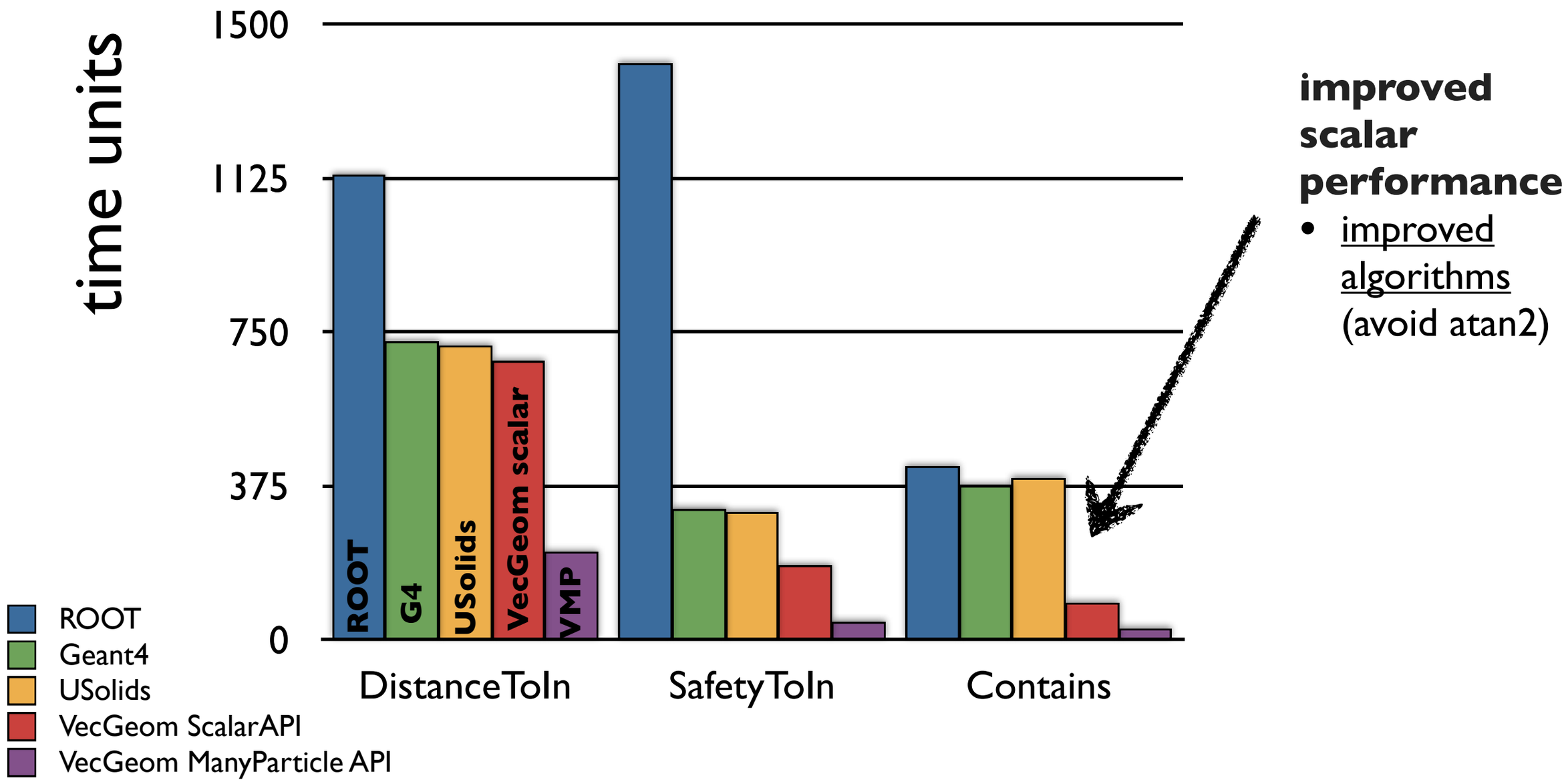
- Introduced Wedge class (half-space given by phi angle)
- Logical part of many shapes: tube-segments, cone-segments, pcon-segments
- Very simple but effective improvement over existing code in USolids and G4



- outside test for point P was so far exclusively done using **atan2**
- now **very fast test** using only 2 dot products of 2D vectors
- enormously **speeding up** „Contains“, Safety, ... for **many shapes**

Performance example Wedge

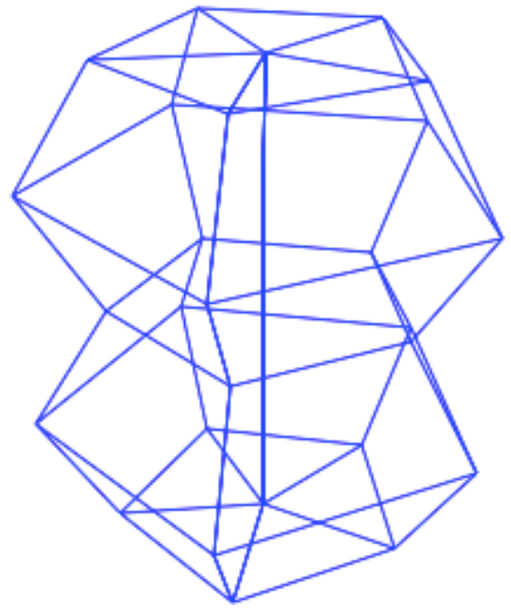
- Effect of „wedge“ on TubeSegment shape (SafetyToIn and Contains)



gcc 4.7; -O3 -funroll-loops -mavx; no FMA; Geant4 10.1 (Release); Root 5.34.18 (Release); benchmark with 1000 particles

VecGeom Polyhedron: Internal Vectorization

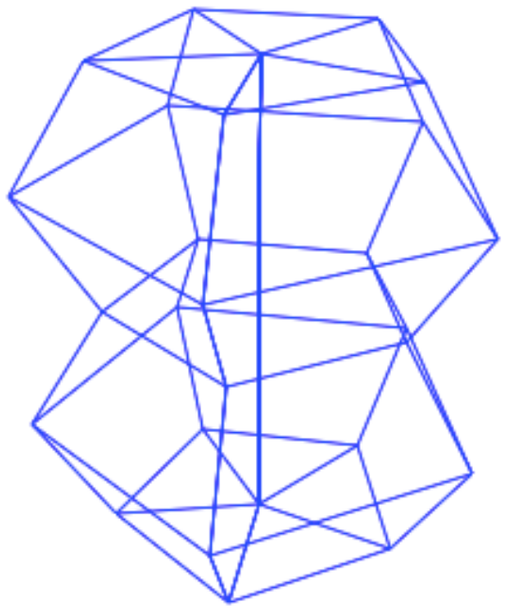
- Regular polyhedra very often used in detectors
- USolids offered an improved variant compared to Geant4; scales very well for large polyhedrons
- Composed of many quadrangular facets in regular arrangement; algorithmically this (implies) the presence of inner loops



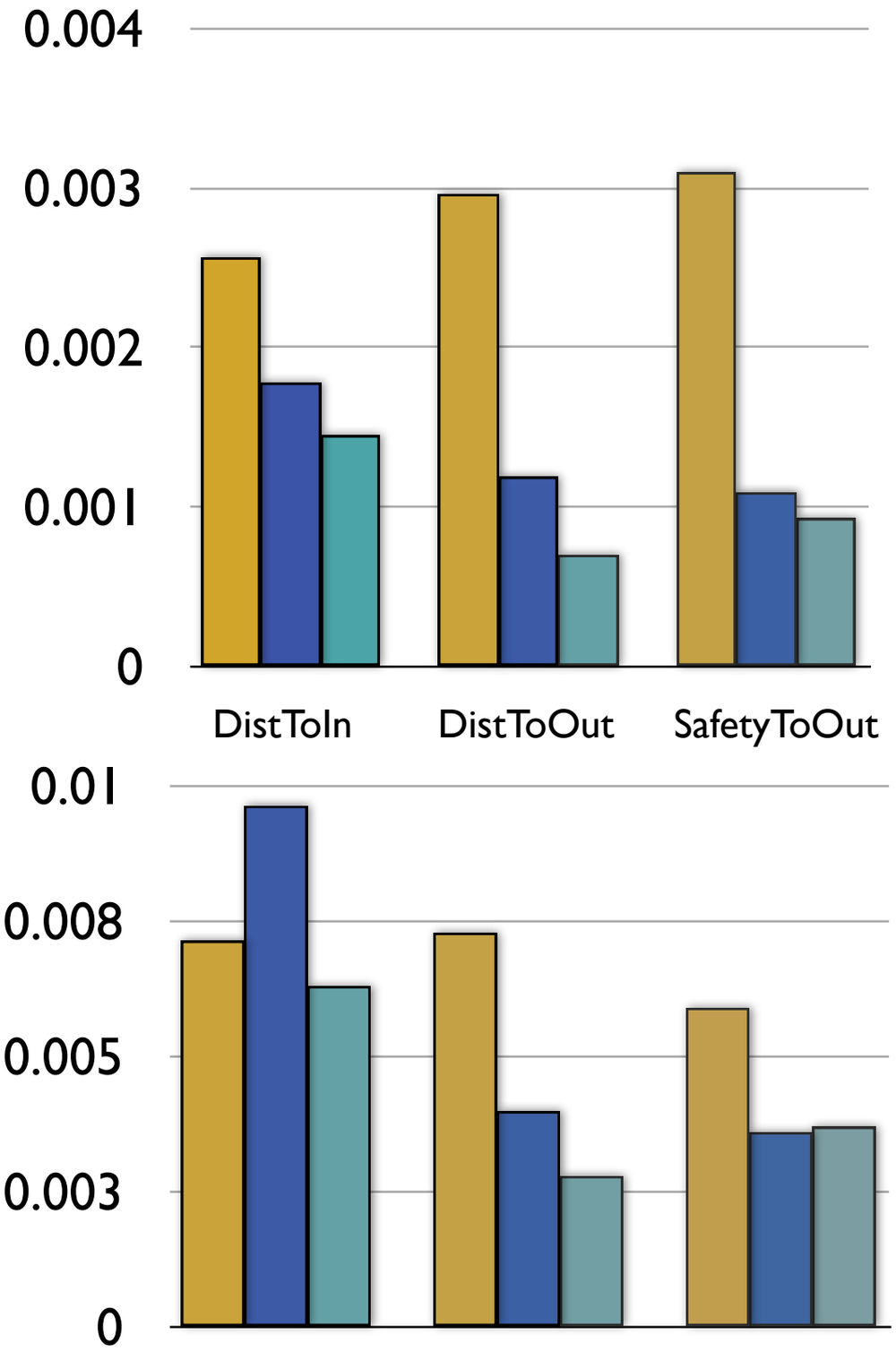
```
for( quadrangle : allquadrangles ){
    quadrangle->Distance();
}
```

- Implemented a VecGeom polyhedron which targets acceleration of such loops via SIMD vectorization; Works very well for not too complex polyhedra
- Research not finished ... complex polyhedra may be sped up with other techniques from ray-tracing (vectorized BVH; see plans for tesselated solid)
- Algorithm is orthogonal to USolids polyhedron; both may have advantages and may complement each other

Improvements in Polyhedron: Some numbers



small test

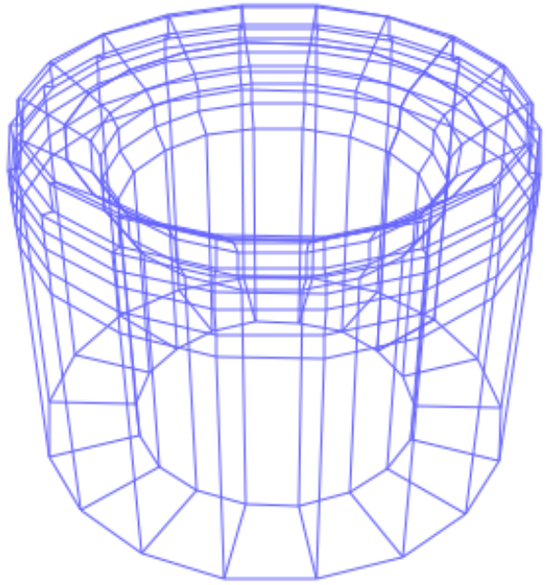


USolids

VecGeom noSIMD

VecGeom SIMD

- for some polyhedra considerable overall improvement compared to USolids implementation



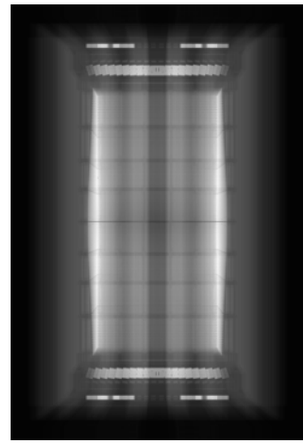
HBHalf@CMS

- For very complex shapes: USolid implementation might be better choice
- demonstrated gain from internal vectorization (typically factor 1.4 ish)
- test done on AVX with 1000 particles

Validating shape implementation

- Spent effort to improve testing/validation of shape implementations
- One new feature is option to compile runtime-checks against Geant4 or TGeo implementations into the VecGeom library
- Development of higher verification tools
  (XRayBenchmarker --> pixel by pixel comparison of navigation)
- Can leverage more Geant4 testing via the VecGeom to Geant4 integration
- Setup of a database for shape tests
- Development of a ShapeStressTester

  see dedicated talk in this session (G. Cosmo + T. Nikitina)



Geometry Model and Navigation

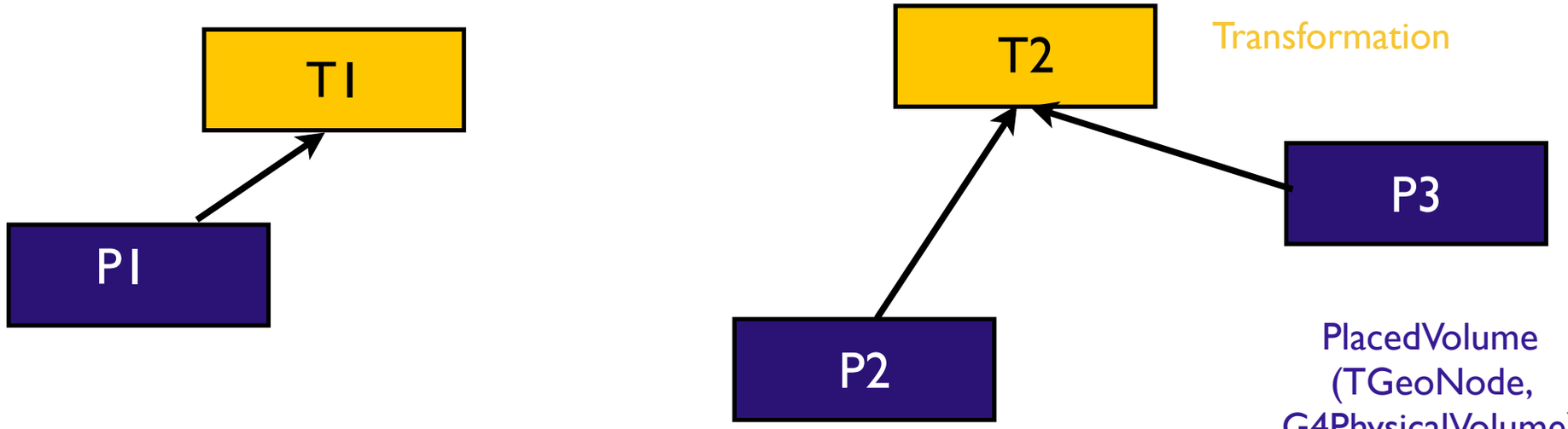
some quick facts

- VecGeom has a hierarchical simple geometry model based on the usual „LogicalVolume - PhysicalVolume“ paradigm
  - do not yet have higher order structures such as parametrizations, divisions, replications
- VecGeom provides navigation functionality in „Navigator“ classes
- Navigator is stateless; state is carried in NavigationState classes; each particle in flight has a NavigationState associated (currently the case in Geant-V)
  - VecGeom is thread safe; can deal many particles at same time
- VecGeom is the navigation system used by Geant-V
- can currently handle the CMS detector...

Compact Memory Model in VecGeom

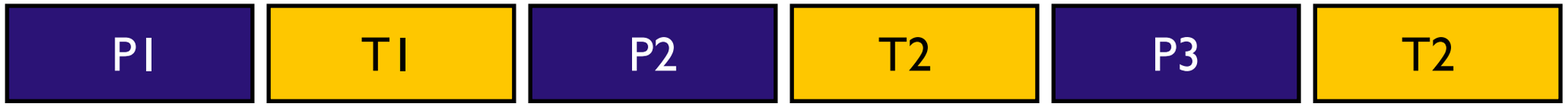
From:

Geometry objects spread in memory



To:

Contiguous array of placed volumes



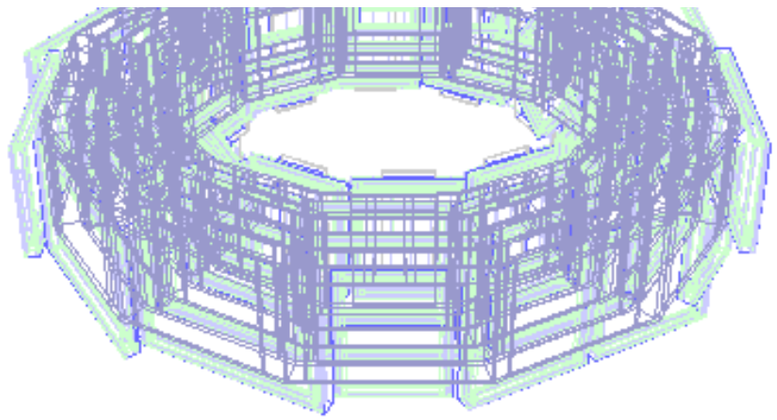
- In VecGeom currently done after loading geometry
- **~10% speed improvement** in complex geometry tracing from compactifying placed volumes alone
- User should not keep pointer to volumes before compactification!
- Extension to other geometry data??

Accelerating Navigation in VecGeom

- VecGeom only had very primitive navigation algorithms up until now (~O(n) scaling with number of daughter volumes)
- simple algorithms are not enough and cannot compete with Geant4/ROOT voxelization techniques (~O(log(n))
- recent R&D activity to improve navigation with a focus on algorithms that can benefit from SIMD vector units
  - inspired by similar progress in ray-tracing (see, e.g., Shallow bounding volume hierarchies for fast SIMD ray tracing of incoherent rays, 2008) 10.1111/j.1467-8659.2008.01261.x

- implemented various algorithms based on „clustering“ volumes into regular hierarchies of (aligned) bounding boxes
- example results: can navigate in MBWheel_1N **>2x faster** than G4 voxelized navigator
- speedup **not due** to shape performance

MBWheel_1N (~700 volumes); most complex element in CMS detector



preliminary, Yang Zhang (KIT) + Sandro Wenzel (CERN)

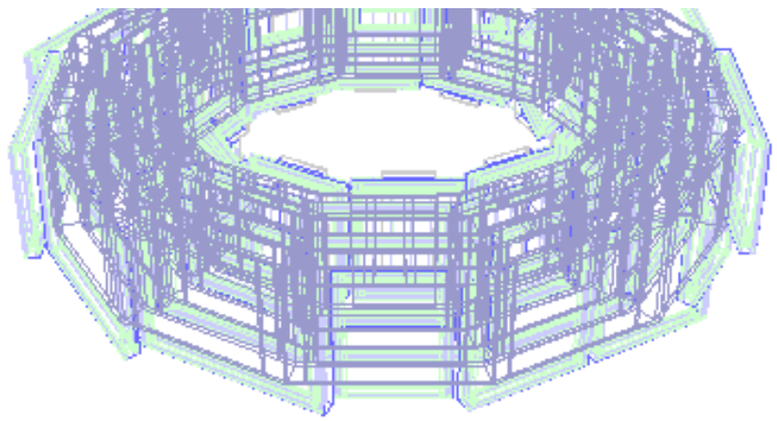
Accelerating Navigation in VecGeom

- VecGeom only had very primitive navigation algorithms up until now (~O(n) scaling with number of daughter volumes)
- simple algorithms are not enough and cannot compete wit[h] nt4/ROOT voxelization techniques (~O(log(n))
- recent R&D activity to improve navigation[ ] that can benefit from SIMD vector units
  - inspired by similar progress in[ ] volume hierarchies for fast SIMD[ ] 10.1111/j.1467-8659.2008.01261.x

VecGeom has competitive navigation (alpha version); ongoing effort to choose best navigator for given logical volumes

- implem[ ] on „clust[ ]hierarchies of (aligned[ ]
- example [ ]an navigate in MBWheel_1N **>2x faste[r]** than G4 voxelized navigator
- speedup **not due** to shape performance

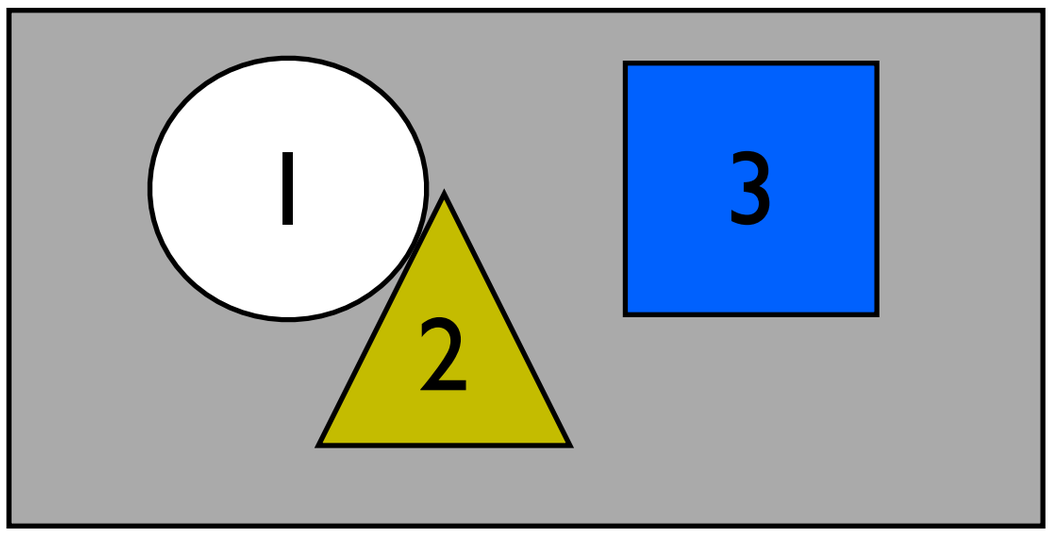
MBWheel_1N (~700 volumes); most complex element in CMS detector



preliminary, Yang Zhang (KIT) + Sandro Wenzel (CERN)

Static geometry properties

- R&D activity to accelerate navigation by exploiting **„static knowledge“** about the detector
- trying out ideas not yet present in Geant4/ROOT:
  - shape convexity property
  - PlacedVolume connectivity / touching properties



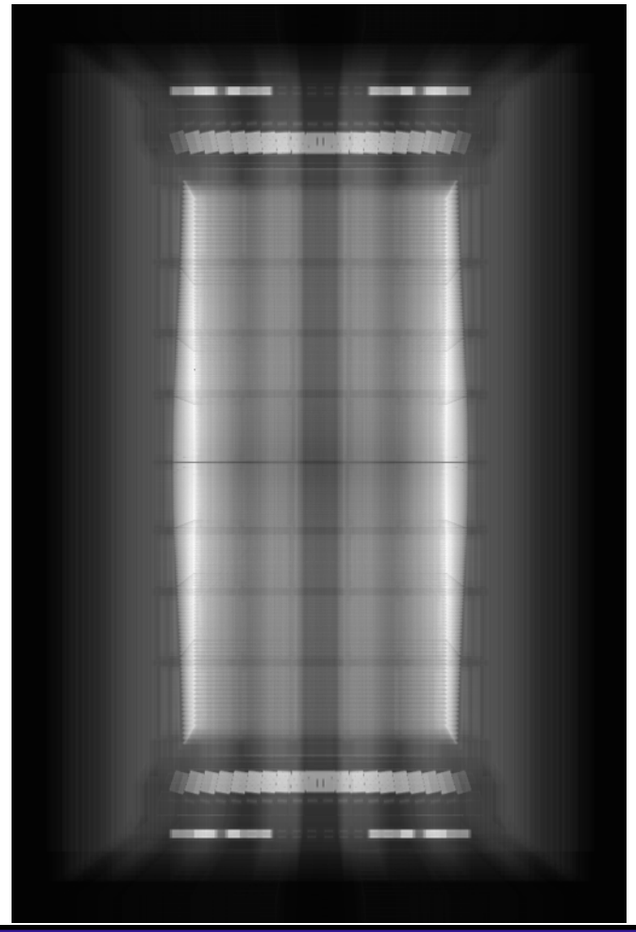
$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \end{array}$$

- status: can „compute/approximate“ connectivity matrix; todo: use in navigation (may speed up relocation)

preliminary, Yang Zhang (KIT) + Sandro Wenzel (CERN)

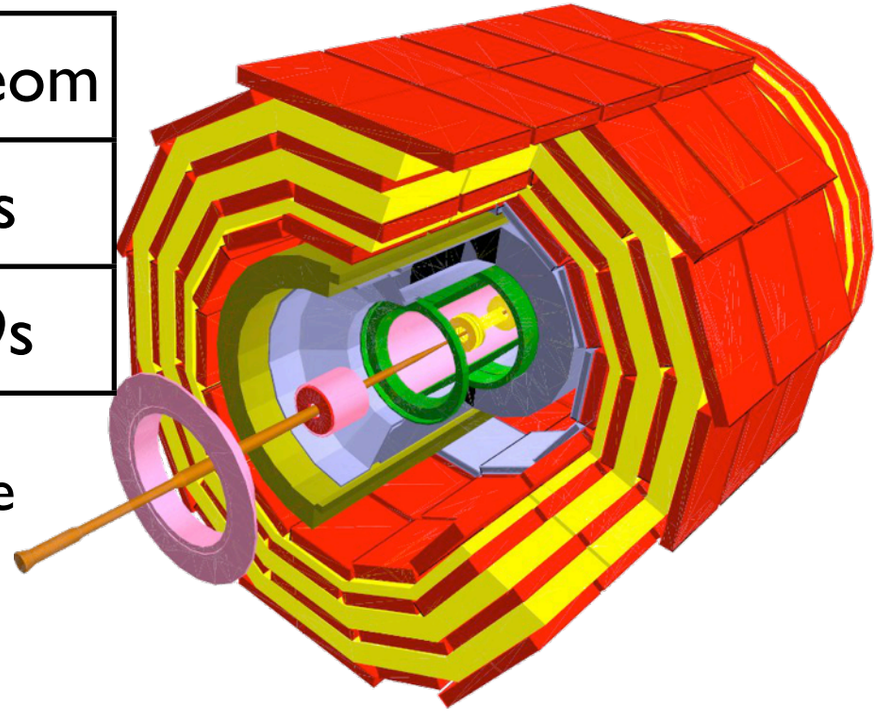
A global performance evaluation

- Trying to benchmark complete geometry modeller: shapes + navigation
- Developed X-Ray benchmark: propagate geantinos pixel-by-pixel
- not a realistic benchmark ... (G4 is not optimized for geantino tracing) ... but an indication that we are globally moving into the right direction
- todo: run G4 test with USolids (instead of native G4 shapes)



dir	G4	ROOT	VecGeom
y	21.5s	12.7s	5.9s
z	10.7s	6.58s	4.09s

time to obtain the X-Ray image for the CMS calorimeter
(VecGeom timing not yet using latest navigators)



see also talk on VecGeom performance tomorrow

(Short term) Plans

Shape level

- Work on remaining shapes completely missing in USolids/VecGeom
- Iterate on other shapes already existing in USolids
- Concrete ideas
  - Tessellated solid ---> implement using SIMD accelerated structures (possibly with Bounding Volume Hierarchies (BVH) or similar)
    - look also into industrial libraries (e.g., Intel Embree)
  - Multi-Union (same)
  - Extruded solid

Geometry level

- Assembly shape of TGeo
- Implementation of replicated structures / divisions / parametrized solids
- Consolidate navigation module

The VecGeom developers

active contributors

Guilherme Amadio (UNESP), John Apostolakis (CERN), Calebe de Paula Bianchini (UNESP), Abhijit Bhattacharyya (BARC), Philippe Canal (FNAL), Federico Carminati (CERN), Gabriele Cosmo (CERN), Andrei Gheata (CERN), Mihaela Gheata (CERN), Guilherme Lima (FNAL), Tatiana Nikitina (CERN), Raman Sehgal(BARC), Sandro Wenzel (PI, CERN)

previous contributors

Marilena Bandieramonte, Georgios Bitzes, Marek Gayer, Heegon Kim, Johannes de Fine Licht, Yang Zhang