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 maintenance 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
  - Normal
  - SafetyFromOutside
  - SafetyFromInside

![Performance graph]

Time per one method call [nanoseconds]

Method

- Inside
- DistanceToOut
- DistanceToIn
- Normal
- SafetyFromOutside
- SafetyFromInside

Performance of methods at folder polycone-3s-360-perf

Revised UPolycone performance example: 3 Z-sections

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“
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 1 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:

\[
\text{VecGeom} = \text{Evolved USolids}
\]

+ Many-Particle API
+ Geometry Model / Navigation

\[
\begin{align*}
\text{Vec} & = \text{SIMD/GPU support} \\
\text{Geom} & = \text{complete geometry modeler}
\end{align*}
\]

[*] 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

double ComputeStep(Vector3D, Vector3D)
This talk: focus on aspects relevant for Geant4 (single track functionality)

Tomorrow: more details on vector-API relevant for Geant-V
### Shape development status

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

the rest is ,,Eltu, Twisted[*], ScaledShape, ...“
## Shape development status

### SIMD acceleration

<table>
<thead>
<tr>
<th>Shape</th>
<th>USolids</th>
<th>VecGeom</th>
<th>Internal SIMD</th>
<th>Multi-Track SIMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Box</td>
<td>yes</td>
<td>yes</td>
<td></td>
<td>yes</td>
</tr>
<tr>
<td>Trap + Trd</td>
<td>yes</td>
<td>yes</td>
<td></td>
<td>yes</td>
</tr>
<tr>
<td>Tube[s]</td>
<td>yes</td>
<td>yes</td>
<td></td>
<td>yes</td>
</tr>
<tr>
<td>Cone[s]</td>
<td>yes</td>
<td>yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GenericTrap/Arb8</td>
<td>yes</td>
<td>(yes)</td>
<td></td>
<td>(yes)</td>
</tr>
<tr>
<td>Tet</td>
<td>yes</td>
<td></td>
<td></td>
<td>(targeted)</td>
</tr>
<tr>
<td>Polycone</td>
<td>yes</td>
<td>yes</td>
<td></td>
<td>(targeted)</td>
</tr>
<tr>
<td>Polyhedron</td>
<td>yes</td>
<td>yes</td>
<td></td>
<td>(targeted)</td>
</tr>
<tr>
<td>Torus</td>
<td></td>
<td>yes</td>
<td></td>
<td>yes</td>
</tr>
<tr>
<td>Parallelepiped</td>
<td></td>
<td>yes</td>
<td></td>
<td>yes</td>
</tr>
<tr>
<td>Extruded solid</td>
<td>yes</td>
<td></td>
<td></td>
<td>(targeted)</td>
</tr>
<tr>
<td>MultiUnion</td>
<td></td>
<td>yes</td>
<td></td>
<td>(targeted)</td>
</tr>
<tr>
<td>Tesselated Solid</td>
<td></td>
<td>yes</td>
<td></td>
<td>(targeted)</td>
</tr>
<tr>
<td>Composites</td>
<td></td>
<td>yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Templat. Composites</td>
<td></td>
<td>(yes)</td>
<td></td>
<td>(yes)</td>
</tr>
<tr>
<td>Hype, Ellipsoid, Parab</td>
<td></td>
<td>yes</td>
<td></td>
<td>yes</td>
</tr>
<tr>
<td>Orb/Sphere</td>
<td>yes</td>
<td>yes</td>
<td></td>
<td>yes</td>
</tr>
</tbody>
</table>

... the rest ...

the rest is „Eltu, Twisted[*], ScaledShape, ...“
## Shape development status

### SIMD acceleration

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

<table>
<thead>
<tr>
<th>Internal SIMD</th>
<th>Multi-Track SIMD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(incomplete)</td>
</tr>
<tr>
<td></td>
<td>(targeted)</td>
</tr>
<tr>
<td></td>
<td>(targeted)</td>
</tr>
<tr>
<td></td>
<td>(targeted)</td>
</tr>
<tr>
<td></td>
<td>(yes)</td>
</tr>
<tr>
<td></td>
<td>(yes)</td>
</tr>
<tr>
<td></td>
<td>yes</td>
</tr>
</tbody>
</table>

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 $\text{atan2}$

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)

- Improved scalar performance
  - Improved algorithms (avoid atan2)

Graph showing performance comparison between different libraries:
- ROOT
- Geant4
- USolids
- VecGeom ScalarAPI
- VecGeom ManyParticle API

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
  
  ```cpp
  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

- USolids
- VecGeom noSIMD
- VecGeom SIMD

For some polyhedra considerable overall improvement compared to USolids implementation.

- For very complex shapes: USolid implementation might be better choice.

- Demonstrated gain from internal vectorization (typically factor 1.4ish).

- 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...
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)

- 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

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 with Geant4/ROOT voxelization techniques (~O(log(n)))

- Recent R&D activity to improve navigation with focus on algorithms that can benefit from SIMD vector units

  - Inspired by similar progress in ray-tracing, such as shallow bounding volume hierarchies for fast SIMD ray tracing of incoherent rays, 2008

- Implemented various algorithms based on clustering volumes into regular hierarchies of (aligned) bounding boxes

- Example: Can navigate in MBWheel_1N >2x faster than Geant4 voxelized navigator

- Speedup not due to shape performance

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

- 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)

<table>
<thead>
<tr>
<th>dir</th>
<th>G4</th>
<th>ROOT</th>
<th>VecGeom</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>21.5s</td>
<td>12.7s</td>
<td>5.9s</td>
</tr>
<tr>
<td>z</td>
<td>10.7s</td>
<td>6.58s</td>
<td>4.09s</td>
</tr>
</tbody>
</table>

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
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