# Report VecGeom

- status information/workplan/ideas -

Sandro Wenzel; for the VecGeom team

Fermilab, 20.10.2014

# reminder what we talked about last visit

## High performance geometry

-- ideas for future direction

( or reasons to start from scratch )

--

**Sandro Wenzel / CERN-PH-SFT**

meeting at Fermilab, 21.1.2013

argued for geometry code rewrite:

- generic ( scalar + vector)
- platform indep (CPU + GPU)
- increased modularity
- increased performance

**challenges continued ... / implications**

- targeting different backends ( vector (Vc, CilkPlus ), GPU, scalar ) sounds like a lot of code repetition if we continue to code the way it was done in the past
  - will be a nightmare for maintenance and testing
- We should hence ( these points are related )
  - write code which is **generic**
    - kernels which work with scalar or vector arguments
  - **reuse code as much as possible without performance loss**
    - example: many kernels for tube / cone / polycone are shared and should be written only once ( without function calls )
    - **write code which is composeable of smaller kernels**

# Today:  Overview (plus points)

Largely put into practice all our primary goals in „VecGeom“:

- developed a general abstraction layer as a foundation to code generic geometry algorithms for CPU-scalar/CPU-vector/CUDA use cases
  - based on traits, templates, function overloading, abstraction layer for ifs, etc...
- CPU-vector is independent of concrete SIMD wrapper class (in theory)
- provided generic algorithms for a handful of geometry primitives
- class structure to represent detectors
- provide ways to copy geometries from CPU to GPU
- provide simple navigation for CPU-scalar/CPU-vector/CUDA
- excellent performance ( scalar, vector, CUDA? )
- USolids compatible and shared USolids/VecGeom repo
- started with systematic testing effort/suite

# Today: Overview (minus points)

- points where we are not doing so well until now:
  - documentation
  - coding conventions
  - some type and function namings which are confusing
  - support for OpenCL
  - testing, testing, testing ( standalone unit tests, shape stress tests, continous integration )
  - benchmarks too limited
  - no continous performance monitoring
  - issue tracking (bugs should be reported ... )

# Status overview

Status overview given for the follow points:

- **Implementation Status:** generic/portable implementations of essential navigation method: Contains/Inside, SafetyTo[In|Out], DistanceTo[In|Out] 
- **GPU tested:** whether code currently compiled on GPU  
  - by construction, our shapes will be usable on GPU; a cross here usually just means „not yet tried“ or „small compilation problems to fix“
- **USolid compatible:** whether the vecgeom shape supports all VUSolid functions ( Normal, GeneratePointOnSurface, Capacity, SurfaceArea, .... )  
  - usually no big effort to achieve this
- **Stress tested:** whether the shape is succesfully stress tested with the new stress-testing framework (Tatiana)  
  - cross here: potentially some hard work to do; not necessarily a blocker though

# Status of shape implementations

| Shape | algthm. Impl. | GPU tested | unit tests | stress test | Usolids compati |
|---|---|---|---|---|---|
| Box | ● | ✓ | ✓ | ✓ | ✓ |
| Paraboloid | ● | ✓ | ✓ | ✓ | ✓ |
| Orb | ● | ✓ | ✓ | ✓ | ✓ |
| Trapezoid | ● | ✓ | ✗ | ✗ | ! |
| Tube[s] | ● | ✓ | ✗ | ✗ | ✗ |
| Trd[1\|2] | ● | ✓ | ✗ | ✗ | ✗ |
| Parallelepepid | ● | ✓ | ✗ | ✗ | ✗ |
| Hyperboloid | ◕ | ✗ | ✗ | ✗ | ? |
| Torus | ◕ | ✗ | ✗ | ✗ | ✗ |
| Polyhedra | ◕ | ✗ | ✗ | ✗ | ✗ |

# Status of shape implementations (2)

| Shape | algthm. Impl. | GPU tested | unit tests | stress test | Usolids compati |
|---|---|---|---|---|---|
| Sphere | ◑ | ✓ | ✗ | ✗ | ✗ |
| Cone[s] | ◑ | ✗ | ✗ | ✗ | ✗ |
| Arb8 + Tet | ◑ | ✗ | ✗ | ✗ | ✗ |
| Ellipsoid | ◔ | ✗ | ? | ✗ | ✗ |
| Polycone | ◔ | ✗ | ✗ | ✗ | ✗ |
| Composites | ◔ | ✗ | ✗ | ✗ | ✗ |
| CutTube | ○ | ✗ | ✗ | ✗ | ✗ |
| Twisted[*] | ○ | ✗ | ✗ | ✗ | ✗ |
| Extruded | ○ | ✗ | ✗ | ✗ | ✗ |
| ScaledShape | ○ | ✗ | ✗ | ✗ | ✗ |
| TesselatedS | ○ | ✗ | ✗ | ✗ | ✗ |

# Relevant for CMS (2014/15 gdml file)

| Shape | algthm. Impl. | GPU tested | unit tests | stress test | Usolids compati |
|---|---|---|---|---|---|
| Box | ● | ✓ | ✓ | ✓ | ✓ |
| Tube[s] | ● | ✓ | ✗ | ✗ | ✗ |
| Cone[s] | ◑ | ✗ | ✗ | ✗ | ✗ |
| Trapezoid | ● | ✓ | ✗ | ✗ | ! |
| Torus | ◕ | ✗ | ✗ | ✗ | ✗ |
| Polyhedra | ◕ | ✗ | ✗ | ✗ | ✗ |
| Polycone | ◔ | ✗ | ✗ | ✗ | ✗ |
| Composites | ◔ | ✗ | ✗ | ✗ | ✗ |

# Shapes: immediate future work

- finish the CMS shapes (including testing)
- test shapes on GPU
- however, even if we don‘t manage everything can always dispatch to ROOT shapes underneath ( on the CPU )

# Shapes: longer term goals

- finish all shapes
- provide Exact + approx Safeties for all shapes ( or maybe ExactSafetySquared )
- DistanceToOut in both versions ( with and without normal calculation ) by using same generic templated code
- step by step integration of vecgeom shapes into USolids ( started already with Paraboloid )
- [your suggestions ...]

# Navigation components: status

- based on new „NavigationState“ objects
- simple (brute-force) navigation algorithm implemented
- navigator is stateless; state is totally encapsulated in „NavigationStates“
- scalar + vector version
- successfully tested in Geant-V; compared against TGeo
- should run on GPU but not yet tested (**good item for this week?**)

# Navigation components: future work

- synchronization (copying) of NavigationState objects between CPU + GPU ( needed to start simulation on CPU and continuing on GPU )

- voxelization for „locate“ functionality; should be easy

- voxelization for „distance“ functionality; might be hard to combine with vectorization; one approach could be to use extreme „voxelization“ as suggested by Rene for the 2 or 3 most important logical volumes.

- [your input...]

# Performance aspects: currently done

- our code performance is very good

- currently we benchmark mostly individual shapes; (for scalar/vector/CUDA) and compare them with Geant4/ ROOT/USolids performance

- benchmark cases and parameters often some standard values  (e.g., hit-biases) which might not be representative for experiments

# Performance aspects: wishes for future

- go beyond shape benchmarks: benchmark navigation on logical volume level
- take many different benchmark cases; ideal scenario: take geometries from experiements + real track data („profile guided benchmarking and optimization )
  - could then choose best navigation algo/parameters + on a logical volume basis
  - started this process ( Heegon + Federico )
- continous performance monitoring ( Jenkins ) with graphics output
- compare performance to industry solutions ( game engines, ray tracing engines, etc. ) -- nice topics to students -- some contacts to industry exist

# Coding conventions; documentations; code structure

- we are not doing well with documentation ( in code and documents explaining the algorithm )
- not doing well following coding conventions
- some namings/interfaces which are still weird
- due to very dynamic team evolution and a very goal oriented procedure
- Propositions??

# Tests

- we are not doing well with testing

- nearly all base classes are missing important unit tests; tests are not run automatically when we commit (or even in Jenkins); usually it is Philippe/Guilherme who point out that something is broken

- immediate actions: we complete the ctests ( easy ):
  - make sure that all tests have proper return codes
  - before each commit we run „make test“

# OpenCL

- initial study done by Gabor Biro
- current conclusion is that OpenCl not able to compile our generic code ( even with AMD C++ extensions )
  - problems are: (virtual functions); system include files, :: operator, ...
- probably wait for next generation compilers, contacted „codeplay“ for beta version of SYCL compiler; should get it soon

# Other (longer term) topics

- **IO** ( gdml, ROOT, other formats: triangles )
- **Visualization**; Rendering of detector elements;
  - started to look into „three.js“

# Other points you‘d like to discuss

- ....