Geometry strategy

Paul Gessinger
CERN
2024-03-11

Introduction

- Naming:
  - `TrackingGeometry` with layers: **Gen1**
  - `Detector` without layers: **Gen2**
- Goal: **we need to converge on the geometry models**
- We need to ensure we can support the geometry and geometry code going forward
- Geometry building is the very first thing *clients* interact with, so it needs to be easy to get into
- Recently thought about a path forward (with some input from Andi St. and Noemi 🙏)

Current landscape of Gen2 geometry I

- Eliminates layer concept ✅
- Navigation delegate for surfaces and portals (abstracts internal structure, acceleration structures) ✅
- Blueprint tree for geometry construction ✅
- Gluing still inspects portals from created volumes instead of taking instructions from blueprint node ❌
- Eliminating redundant container volumes ✅
- Builder/factory/provider chain arguably more complex than Gen1 helpers/creators/builders ❌
- VolumeBounds construction from external constraints uses bound value vector ❌
- Navigation delegates manipulate the state directly ❌

Current landscape of Gen2 geometry II

- Navigation state explicitly stores current portal and surface separately, navigator has to if/else on them (wanted to unify) ❌
- Widespread use of weakly typed binning enum + complex ”casting” code ❌
- No volume hierarchy (need root volume finder instead) ❌
  - Could have delegate purely for acceleration structure for internal lookup
  - Do hierarchy search to find ”root volume” by consulting ”find child volume” delegate
  - Avoid having to duplicate some lookup data structure for ”root volume” finding
- Ownership model:
  - Internally stores mutable, access based on outer reference ✅
  - Many `shared_ptr` instead of clearly defined ownership ❌
- Portal volume update delegate instead of explicit binning: good idea but
  - Requires holding volumes in portals anyway ❌
  - Prevents “parallel fusing” (could be used to avoid redundant container volumes instead) ❌

Proposal

Take everything we've learned in Gen2, refactor Gen1 to become Gen2+

(= Gen3?)

Proposal I

1. Delete `TrackingVolume` ”confined volumes”: volumes are owned by parent, accessible by portals (enter subvolume, exit current volume to neighbor), transparent to `Navigator`
2. Refactor `BoundarySurface` to resemble `Portal`
   - Drop volume updater
   - Reintroduce explicit binning to `Portal`
   - Implement ”parallel” fusing to allow joining adjacent volumes at the Portal level
3. Refactor `TrackingVolume` ownership and access
   - Storage: `shared_ptr` to mutable `Surface` / `Volume`
   - Access:
     - If mutable: as `shared_ptr`, as unpacked mutable range (on the fly unpacking of `shared_ptr`)
     - If const: as unpacked const range (on the fly unpacking of `shared_ptr`)
4. Keep most of layer structure builder and associated auto-configuration

Proposal II

5. Add navigation delegate mechanism to TrackingVolume modeled after DetectorVolume
   - Maybe: return range (vector?) of intersections of surfaces (sensitives, passives + portals) instead of writing into navigation state to make the data flow more obvious
6. Implement blueprint tree style construction of TrackingGeometry
   - Make InternalStructureBuilder / ExternalStructureBuilder internal to Node (interface + derived class)
   - Encode attachment direction in parent-child relationship
     - Ordering explicit or implicit?
     - Can volume->volume child relationships be used to register ”confined” volumes, whose portals are not fused at all? E.g. portal fusing only happens at container nodes, but non-container nodes can also have children?
   - Use portal fusing to avoid back-and-forth during construction and just have children construct volume bounds on their own, then fuse portals

Proposal III

- Requires being able to merge material grids (i.e. consistent equidistant binning or variable binning), or could be implemented by material grids becoming a tree (with a cache?)
- Material designation can be part of the node tree itself, post merge, so would only need to merge protobinning

- Reuse fill-gaps functionality to simplify wrap-insert-attach
- Container builders become derived classes of Node and operate on their subtree only
- Enable client-usable API to construct blueprint tree
  - Enable programmatic building with or without pre-grouped surfaces
  - Enable mixing different geometry source (think DD4hep tracker + programmatic calo + TGeo muon system)
  - All ”auto-detect” / ”auto-construct” helpers (KdTree) return blueprint subtrees that can be attached to an existing blueprint tree

7. Write navigator that uses navigation delegate in Gen3 geometry