Geometrical Modeller – Alignment issues

Andrei Gheata

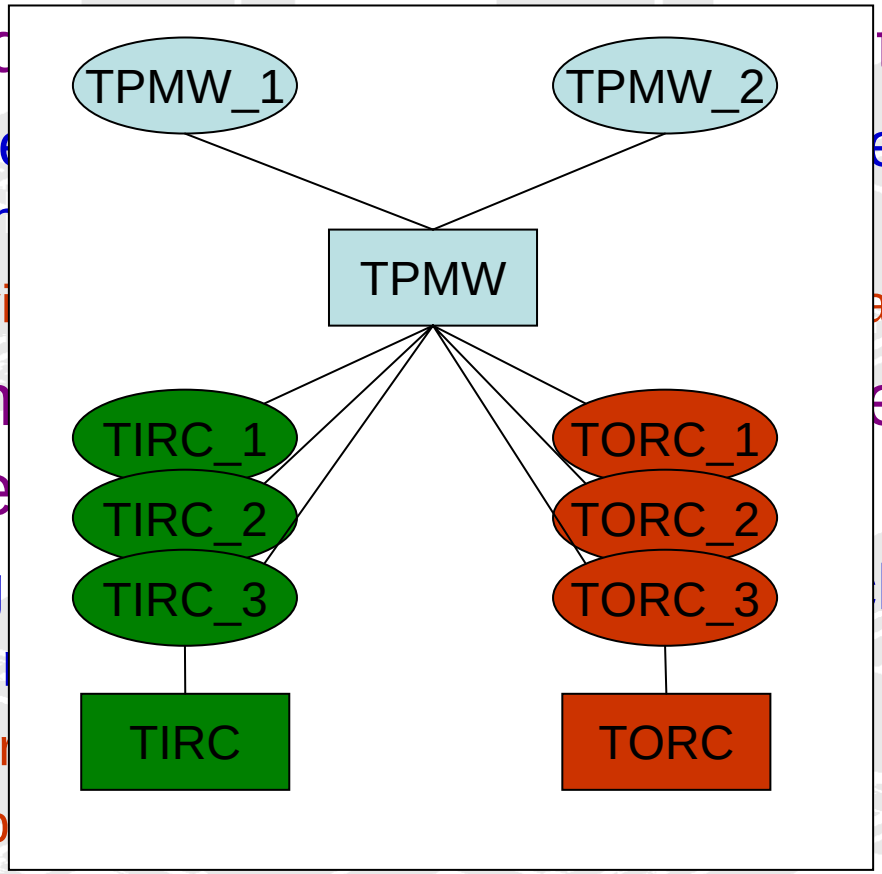
ALICE offline week, 3-7 Oct. 2005

Outline

- (Mis)Alignment in TGeo
- How to use it
- Assemblies of volumes
- Deformations - scaled shapes
- Conclusions

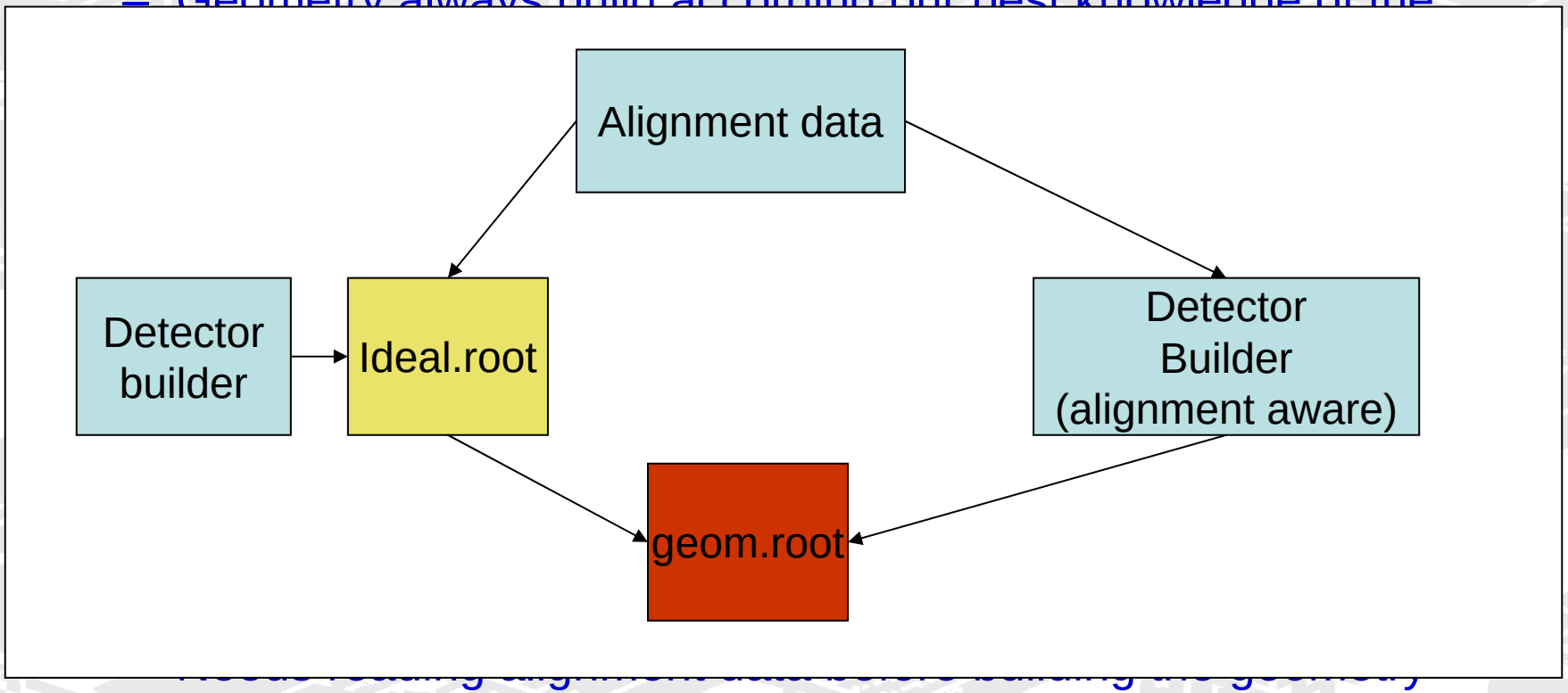
(Mis)Alignment for TGeo

- The mod ture
  - An obje ed in a contain
    - Movi ll its content
- Compon erally replicate
  - Chang hange the positio
    - Not r
    - An b be decoupled



Two approaches

- "Ideal" + perturbation approach
  - Geometry always build according our best knowledge of the
  - Needs reading alignment data before building the geometry
  - Needs re-thinking/re-writing the geometry code



Physical nodes

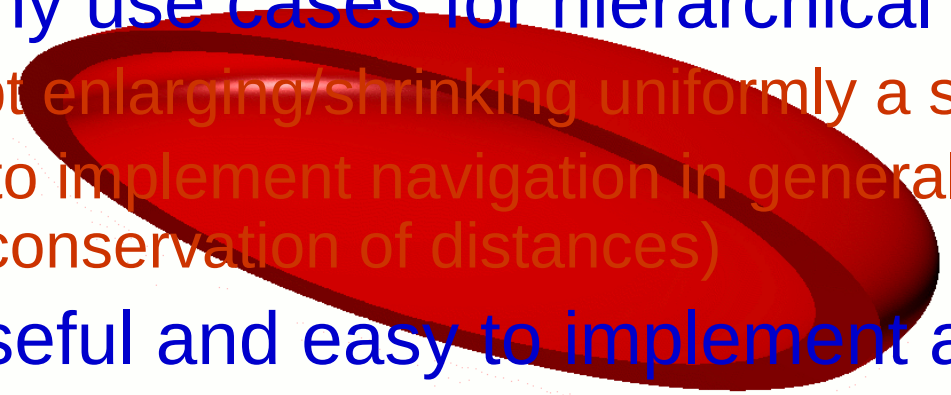
- Describing a single object in the geometry
  - Fully qualified by a path
    - E.g. /ALIC_1/TPC_1/TPMW_1/TORC_15
  - May point to a container object
    - E.g. /ALIC_1/TPC_1
    - In this case moving it will move the whole content
- Class TGeoPhysicalNode
  - TGeoPhysicalNode(const char *path)
  - TGeoPhysicalNode::Align(TGeoMatrix *newmat, TGeoShape *newshape, Bool_t check)
    - Can effectively change the position/shape of an element in the existing closed geometry
      - Why shape ? A possible use case are deformations -> scaled shape

TGeoPhysicalNode

- Represents a geometry object that during the experiment will have a different position than the ideal one
- Definition: new TGeoPhysicalNode(path)
  - Object that is completely decoupled from the active geometry
  - Keep pointers to all TGeoNode objects in the corresponding branch. The object subject to alignment is the last one in the branch
  - Stores the global transformation matrix for fast master⇔local conversions
- Allows redefinition of the relative positioning matrix of the last node in the branch
  - TGeoPhysicalNode::Align(pNewMatrix)
    - Duplicates the branch using the new matrix and connects it in the active geometry tree
    - Re-voxelizes the mother volume of the aligned node
    - Optionally checks if the new position produces overlaps

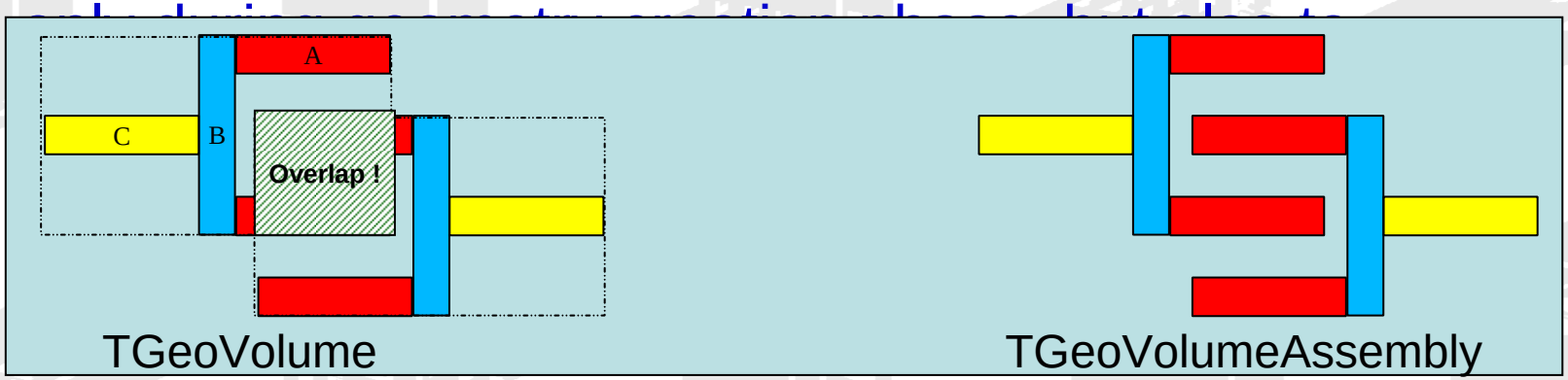
Scaled shapes

- Possibility to scale geometry requested long time ago by users
  - Not many use cases for hierarchical scaling
    - Except enlarging/shrinking uniformly a structure
    - Hard to implement navigation in general case (non-conservation of distances)
  - More useful and easy to implement at shape level
    - New shapes, deformations, …
- Scaled shapes available in CVS version



Volume assemblies

- Union of several different volumes positioned with respect to a common local frame
  - An assembly is just a volume having no shape container
    - No medium/material needed: a point INSIDE the assembly is always in one of the components
  - The assembly provides a logical grouping used not

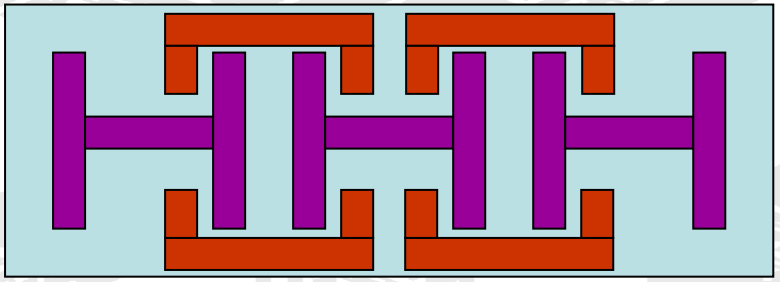


Usage of assemblies

- Easy creation using TGeoVolume interface
  - *TGeoVolume *vol = new TGeoVolumeAssembly(name);*
  - *vol->AddNode(pVol1, id, pMatrix);*
  - Same constrains as for volumes (no overlaps of components) except extrusion
- Very useful for defining complex structures where is hard to define a container made of a basic shape
  - Nesting allowed
  - No navigation performance penalty
- Example of usage in: $ROOTSYS/tutorials/assembly.C
  - Description in release notes

Converting existing geometry to assembly structures

- We have:
  - All daughters in same container (for “MANY” reasons)
    - CONTAINER/ A,B,C,…
- We want to create assemblies:
  - For all structures that should stick togeather
    - CONTAINER/ASM1/A,B,C
    - CONTAINER/ASM2/D,E,F



- We do:
  - TGeoVolumeAssembly *ASM1 = new TGeoVolumeAssembly(name);
    - Instead of CONTAINER->AddNode(A,…);
    - We do:
      - ASM1->AddNode(A,…);
      - CONTAINER->AddNode(ASM1);
    - E.g. we add an additional level between CONTAINER and A,B,C,…
- No interface at VMC level
  - One can add TGeo specific code protected by IsRootGeometrySupported

Overview

- TGeo provides support for:
  - Plugging alignment data in existing geometry (TGeoPhysicalNode)
  - Scaling shapes (TGeoScaledShape)
  - Describing assemblies as trackable structures to avoid flat geometries and potential overlaps of containers
- To adapt and optimize trackers in real conditions, detector code should be aware of:
  - What can be moved in their REAL geometry
  - At which scale
- Examples of usage of these features already exist:
  - $ROOTSYS/tutorials/assembly.C
  - $ROOTSYS/tutorials/geodemo.C
- Short demo following…