Geometry framework status

- **Definition**
- **Detector geometry service**
- **Detector Element Approach**
  - –Outline
  - –New software
  - –Implications for users
  - –An example
  - –Requirements from sub-detector SW
- **Summary**
- **Work in progress**

Juan P. Palacios
CERN

Working definition

- **Provide users of detector geometry description with geometrical information derived from two places:**
  - Nominal alignment of all the sub-components of LHCb
    - Placements of logical volumes within hierarchy (XMLDDDB)
  - Deviations from the nominal alignment
    - Thought of as conditions: could change
- **Provide mechanism for users to modify the deviations from nominal alignment for a given sub-component**
  - ensures modifications are propagated coherently to the geometry description
- **Provides a mechanism to write a new set of deviations to the conditions database**

The very basic functionality to allow for alignment algorithms to work

Detector geometry service

- **Accessible through DetectorElement::geometry() methods**
- **Return IGeomertyInfo* which has local to global and global to local transformations and a host of other services**
  - See **http://lhcb-release-area.web.cern.ch/LHCb-release-area/LHCB/doc/html/class_i_geometry_info.html**
- **Incorporate mialignments into IGeometryInfo**
  - Each detector element will now have ideal and misaligned geometry information
  - User code should remain the same

Detector element approach

- **Basic idea:**
  - Make transformations accessible to the Detector Element and let it do the work
- **Two limits:**
  1. **ALIGNMENT**: One detector element per alignable object. Needs basically one transformation.
  2. **OTHER CONDITIONS (?):** A detector element that knows enough about its PVolume daughters (and their daughters, and their daughters' daughters, and their daughters' daughters' daughters, and …) to assign the correct transformations to each

The basic idea

- **One detector element per alignable object**
  - Can be simply detelemfer in XML and default detector element
  - VELO example: whole VELO, detector halves, r-ϕ pairs, individual sensors
- **One "delta" transformation matrix for each detector element: *conditions catalogue + AlignmentCondition***
  - Catalogue with one to one mapping to alignable objects
  - Paths point to plain XML file or address on conditions data store interface
  - Delta held in new specialised ***AlignmentCondition*** class
- **New *IGeometryInfo* implementation:**
  - Makes own AlignmentCondition object from catalogue path

New IGeometryInfo

- **Handles both ideal and delta transformations:**
  - DetectorElement public interface stays the same.
  - IGeometryInfo toGlobal and toLocal methods now deal with combined ideal + delta transformations
  - Methods to get ideal geometry have been added
  - Possible for users to update delta matrix
  - Easy to “refresh” state of new GeometryInfo. All caching controlled from one method, aptly named ***cache()***
  - GeometryInfo accesses CondDB (or test XML file), generates all necessary matrices
- **Implementation ready and tested with XML conditions catalogue**

What does GeometryInfo do?

- **Constructor uses path to get AlignmentCondition**
- **Scans down tree picking up parent transformations**
  - Iterative procedure finding GeometryInfos of support detector elements
  - Stores ideal and delta transformations for each level in vectors
    - Allow to re-calculate after updates
  - Combines ideal matrices to get ideal case local to global
  - Combines all matrices to get local to global with misalignments
- **“Local” misalignment matrix can be updated by user**
  - Re-calculates “global” matrix automatically
  - At the moment this is de-coupled from automatic update mechanism

Conditions catalogue

- **Added "condition" attribute to geometryinfo in structure.dtd**
  - Only necessary to add an

    ```
    <geometryinfo condition="somePath" />
    ```

    in each <detelem …. /> definition
  - Then store the condition somewhere:

```
<condition classID="6" name="somePath">
     <paramVector name="dPosXYZ" type="double"> 0. 0. 0.</paramVector>
     <paramVector name="dRotXYZ" type="double"> 0. 0. 0.</paramVector>
</condition>
```

- **At the moment, test XML files**
- **Can store as XML strings in CondDB: keep format the same!**

Creation of an AlignmentCondition

- **At the moment, via XML and dedicated "converter"**
- **classID = 6 maps to XmlAlignmentConditionCnv**
  - This is a little template which simply instantiates the right kind of condition
- **AlignmentCondition has access to the parameters and constructs the matrices**

```
<condition classID="6" name="/dd/Conditions/LHCb/myDetector/Module67">
     <paramVector name="dPosXYZ" type="double"> 0. 0. 0.</paramVector>
     <paramVector name="dRotXYZ" type="double"> 0. 0. 0.</paramVector>
</condition>
```

- **GeometryInfo accesses condition via data service:**

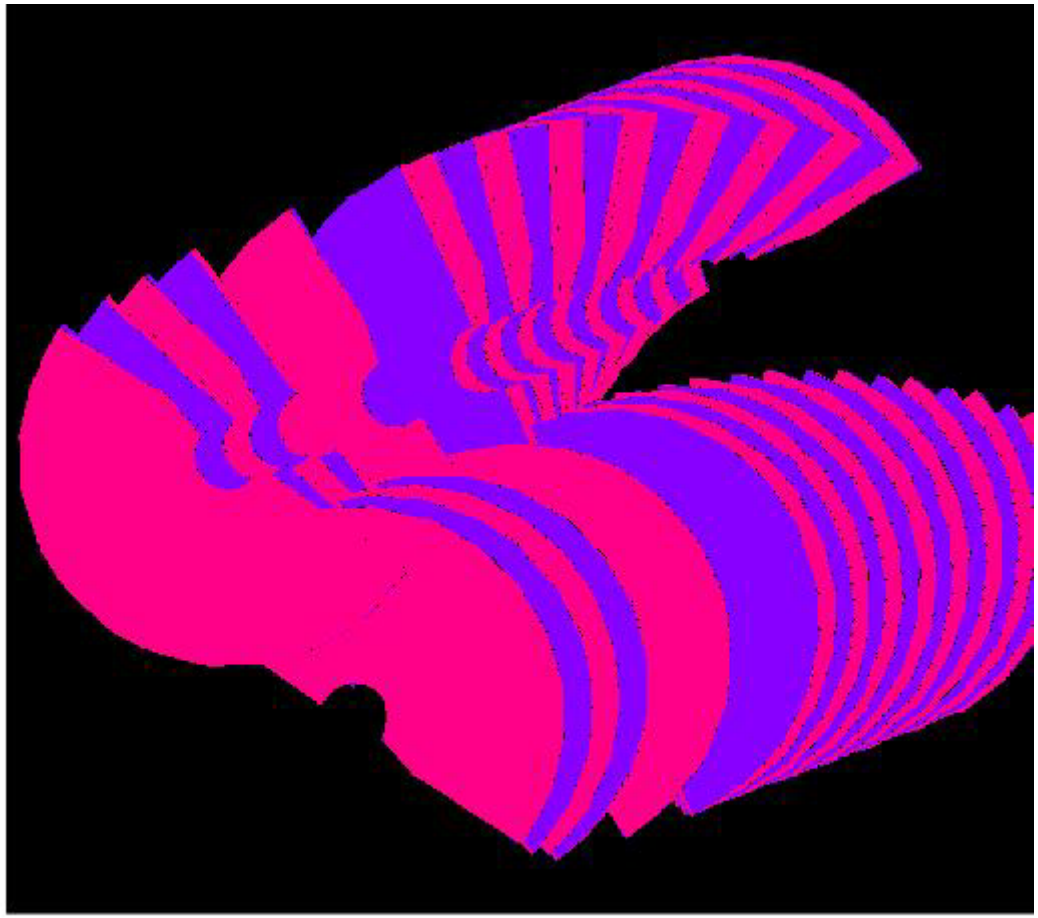
```
SmartDataPtr<AlignmentCondition> cond(datasvc(),
                    "/dd/Conditions/LHCb/myDetector/Module67");
```

How to use misalignment information

- **Keep currently used DetectorElement interface**
  - DetectorElement::geometry() still points to IGeometryInfo
  - IGeometryInfo handles ideal and misaligned geometry
  - Standard transformation methods deal with misaligned geometry
    - User code will automatically get misaligned geometry transformations if unchanged
  - New methods allow to perform ideal transformations
  - Delta matrix also available – and can be modified

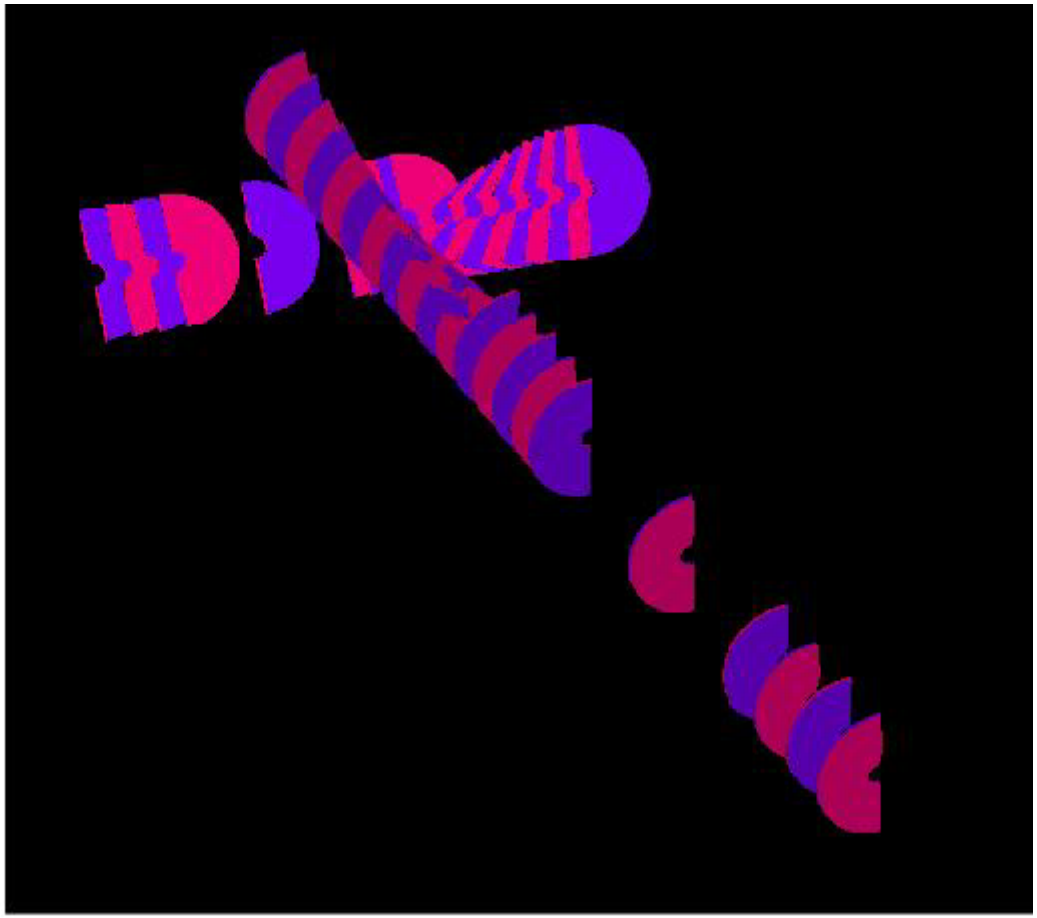
CODE USING DetectorElement::geometry() WILL NOW GET MISALIGNED GEOMETRY!

An example of a misaligned detector



Open VELO with Z-dependent ϕ misalignemt from test XML conditions file

Another example of a misaligned detector



Open VELO with Z-dependent ϕ misalignemt and rotation about X axis. From test XML conditions file

Requirements from detectors

- **A detector element per alignable object**
  - Must have path to an alignment condition
  - In current scheme this means a condition in the XML <geometryinfo /> definition of the detector element. This contains the correct path in the data store
- **A catalogue of conditions with one-to-one mapping to detector elements**
  - In absence of condition identity matrix is created so current XML descriptions still work for ideal
- **A geometry description where the alignable objects can be associated to an LVolume**
  - Actually, an ASSEMBLY might do, but can't say for sure yet!
- **A good knowledge of where and how geometry information is cached: more of this next week!**
  - But basically, should strive to do all caching from a small number of methods as they will have to be registered somewhere

Summary: what is there

- **XML conditions catalogue**
  - Changes to structure.dtd condition definition and XmlBaseConditionCnv to allow for conditionrefs inside conditions
  - VELO example tree of references
    - Working for now with dummy XML alignment parameter vectors
    - Populating CondDB with XML strings of the same format seems trivial
- **New GeometryInfo class**
  - New implementation
  - Instantiated in DetectorElement from XmlBaseDetElemCnv
  - Modifications to both the above to pass condition address
- **AlignmentCondition class**
  - Assigned **classID 6** and “Wrote” XmlAlignmentConditionCnv
  - Instantiated in GeometryInfo using path to get alignment parameters from (XML based for now) data store.

Work in progress

- **Use of CondDB data store interface**
  - Conditions retrieval
  - Updating conditions in data store and DB
- **Incorporate update manager**
  - Mechanism to update necessary cached info after a condition has changed.

Have necessary code from Marco. Incorporate this week.

- **Update of conditions on the fly:**
  - Implemented, needs testing
- ~~**Visualisation**~~
  - ~~Could we see move volumes in Panoramix? Good for debugging/interpreting~~

DONE

- **Related note: what about simulation? Can the misalignments be used there?**

In progress. Gauss crashing and don't know why yet