DD4hep

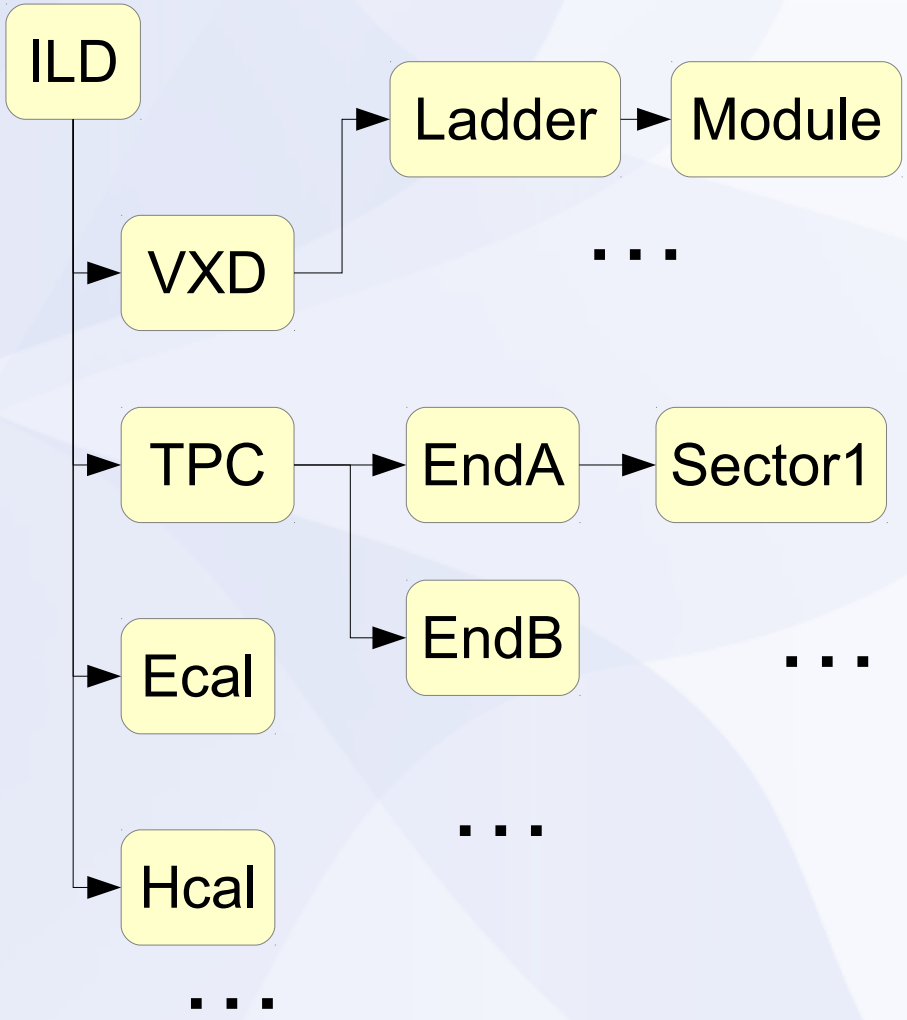
Tutorial Session

Assumption: You followed the introduction Tuesday morning

- **Refresher of the Design**
- Discussion of the different components
  - A few words to the C++ API
  - XML compact description and DTD structure
  - Detector constructors
  - Visualization
  - Detector Views

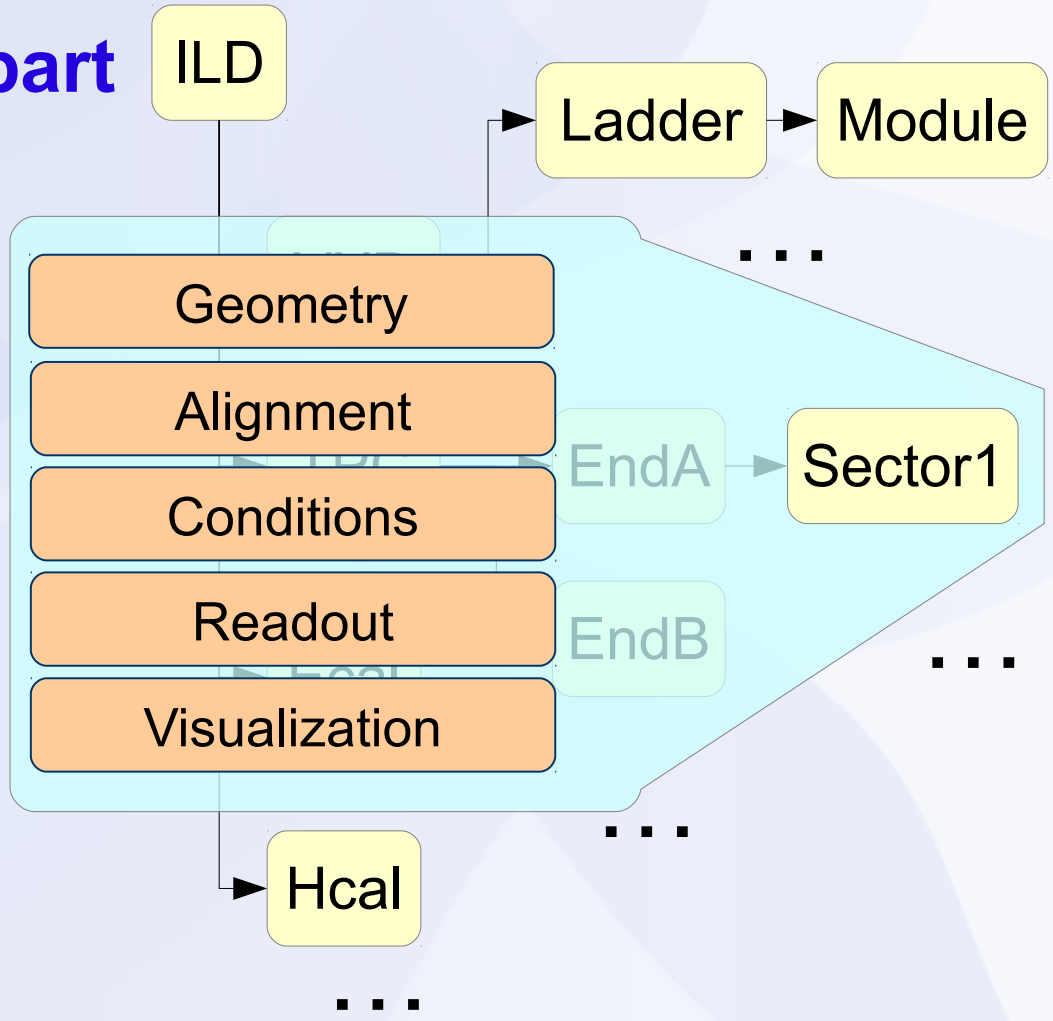
What is Detector Description ?

- **Description of a tree-like hierarchy of "detector elements"**
  - **Subdetectors or parts of subdetectors**
  - **Example:**
    - Experiment
    - TPC
    - Endcap A/B
    - Sector
    ...



What is a Detector Element ?

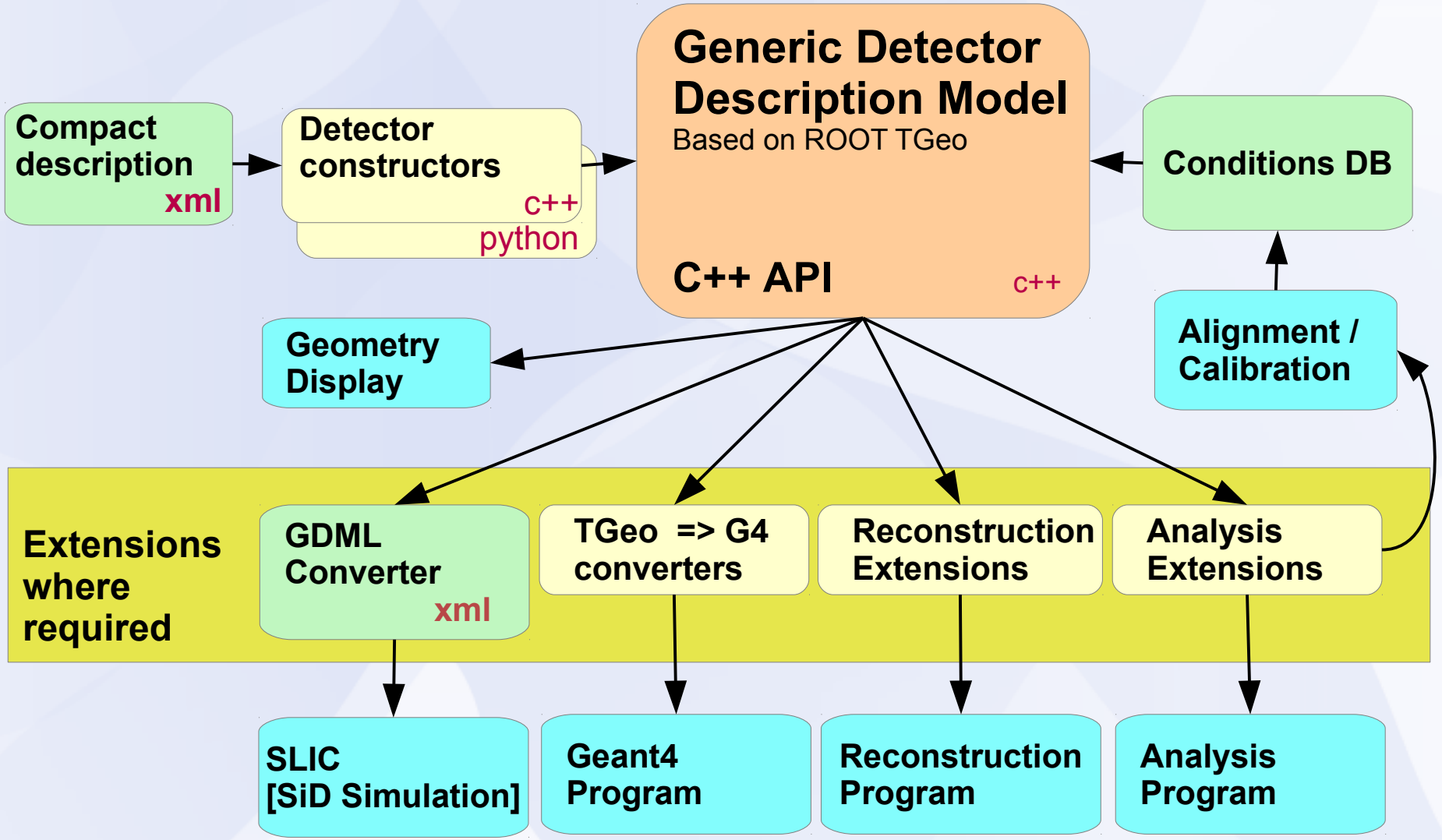
- **Subdetector or the part of a subdetector including the description of its state**
  - **Geometry**
  - **Environmental conditons**
  - **Properties required to process event data**



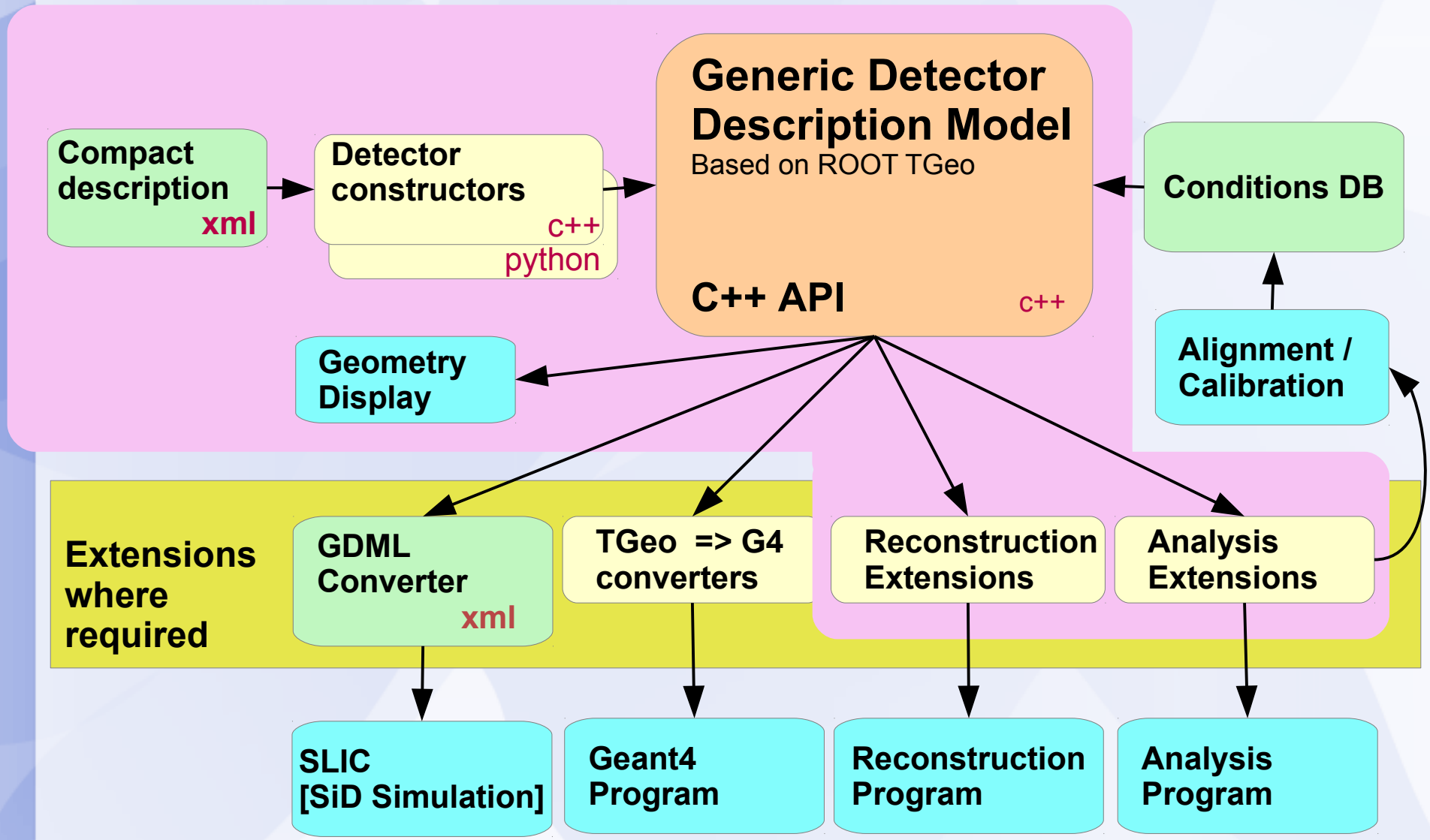
What is a Detector Element ?

- Detector Elements describe parts of a sub-detector of 'reasonable size and complexity'
  - Apply common sense
  - An explosion of complexity is counter productive
  - There cannot be a strict rule
- Rule of thumb: 'Something worth having a name'
  - Good: TPC – SideA – Sector 8
  - Bad: TPC – SideA – Sector 8 – Sense wire 2856
  - If a thing is so complex, that you need an enterprise to describe it ... maybe you did not think enough

DD4Hep - The Big Picture



The Scope of the Tutorial



Aim of this Session

- Tutorial like style
  - If questions, ask right away
  - I was told there is plenty of time
  - Hence if not … punish the organizers
- Introduction of the main detector description components
  - Compact description – structure of XML file
  - Detector constructors
  - How to visualize
  - User extension object

To upgrade the hype factor, follow the exercises (lxplus ONLY!!)

- **Execute:**
  $> . /afs/cern.ch/user/f/frankb/public/DD4hep_setup.sh
  [in svn: trunk/doc/build.sh]

- **This will check-out and build the software**
  - **The files used in the tutorial are typically mentioned**
  - **Look at the full file, since for the slides I had to shorten them**

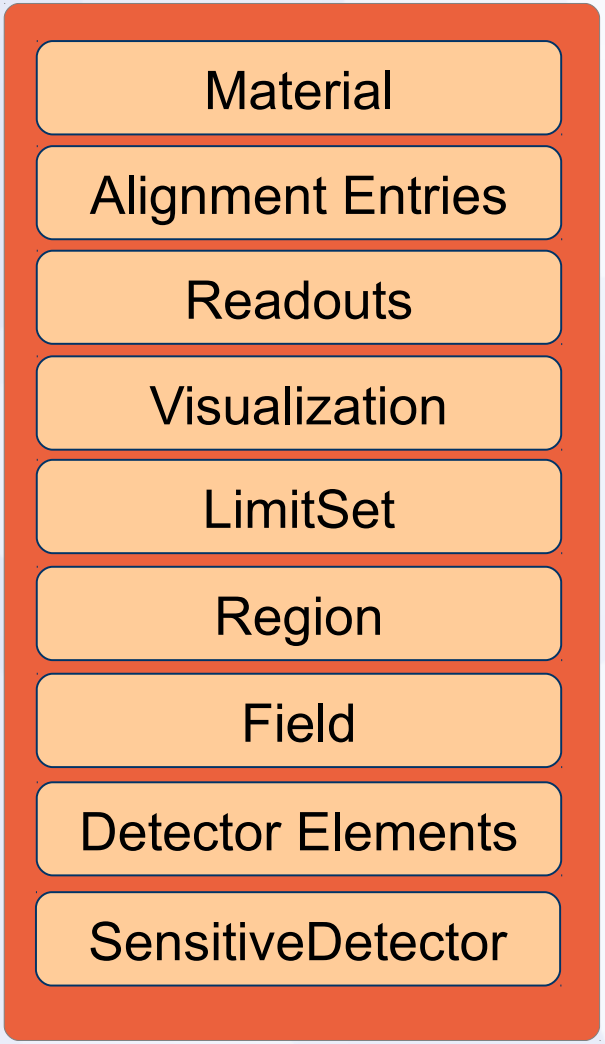
Useful URIs

- **Svn repository: https://svnsrv.desy.de/public/aidasoft/DD4hep/trunk/DDExamples**
- **ROOT documentation: http://root.cern.ch/root/html534/ClassIndex.html**
- **DD4hep page: http://aidasoft.web.cern.ch/DD4hep**

- Refresher of the Design
- **Discussion of the different components**
  - **A few words to the C++ API**
  - XML compact description and DTD structure
  - Detector constructors
  - Visualization
  - Detector Views

Top Level C++ Interface

- **The interface is a C++ ABC,**
  - **which allow access to entities describing the detector 'by name'**
  - **Mostly filled by the compact detector description**
- **No real intelligence, see it 'as set of shelves' with items necessary to 'construct the experiment'**



https://svnsrv.desy.de/public/aidasoft/DD4hep/trunk/DDCore/include/DD4hep → LCDD.h

- Refresher of the Design
- Discussion of the different components
  - A few words to the C++ API
  - **XML compact description and DTD structure**
  - Detector constructors
  - Visualization
  - Detector Views

XML – Compact Description[1]

- Human readable ASCII format
- Extensible: Easy definition of new structures
- Interpreter support: units and formulae
- Parsed by DD4hep core
- As close as possible to lccdd notation (DTD)

Intellectual property: J.McCormick / SLAC

XML Compact Description – Structure

- **lccdd** *Linear collider compact detector description*
  - **includes** *XML include files for material DB*
  - **info** *Info about the detector model, author etc.*
  - **define** *Constant definitions*
  - **materials** *Extensions to material DB*
  - **display** *Visualization settings*
  - **detectors** *Subdetector definitions*
  - **readouts** *Readout information for simulation*
  - **limits** *Limitsets for simulation*
  - **fields** *Electric/magnetic field definitions*

Also in DD C++ API

XML Compact Description – Structure

- **includes** *XML include files for material DB or location of python drivers*

```
<includes>
    <gdmlFile  ref="elements.xml"/>
    <gdmlFile  ref="materials.xml"/>
    <pyBuilder ref="../drivers"/>
</includes>
```

- **Material DB structure (same as 'materials' tag):**

```
<materials>
 <element Z="89" formula="Ac" name="Ac" >
  <atom type="A" unit="g/mol" value="227.028" />
 </element>
</materials>
```

XML Compact Description – Structure

- **define** *Section with constant definitions*

```
<define>
    <constant name="world_side"          value="7500*mm"/>
    <constant name="world_x"             value="world_side/2"/>
    <constant name="world_y"             value="world_side/2"/>
    <constant name="world_z"             value="world_side"/>
    <constant name="TPC_outer_radius"    value="1808*mm"/>
    <constant name="TPC_inner_radius"    value="329*mm"/>

    <constant name="Ecal_Tpc_gap"        value="35*mm"/>
    <constant name="Ecal_Lcal_ring_gap" value="54.8*mm"/>
    ...
</define>
```

  - *Parameters shared by subdetectors*
  - *Parameters are interpreted, 'reasonable' calculations are possible*
  - *Could possibly be generated from Mokka database*

XML Compact Description – Structure

- **display** *Visualization settings*

```
<display>
  <vis name="VXDVis" alpha="1.0" r="0.5" g="0.5" b="0.5"
                     drawingStyle="solid"
                     showDaughters="true"
                     visible="true"/>
  ...
<display>
```

  - Defines color and behavior of children
  - Visualization settings are properties of *Volumes*
    - Volumes of one detector element may have different visualization settings

XML Compact Description – Structure

- **readouts** *Readout information for simulation*

```
<display>
  <vis name="VXDVis" alpha="1.0" r="0.5" g="0.5" b="0.5"
                     drawingStyle="solid"
                     showDaughters="true"
                     visible="true"/>
  ...
<display>
```

  - Recipe to assign energy deposits created in Geant4 to sensitive detector volumes using volume identifiers
  - Identifiers need to be assigned to Volumes in the detector constructors
  - Used in simulation and reconstruction
  - So far not used (DD4hep not yet used for simulation)

XML Compact Description – Structure

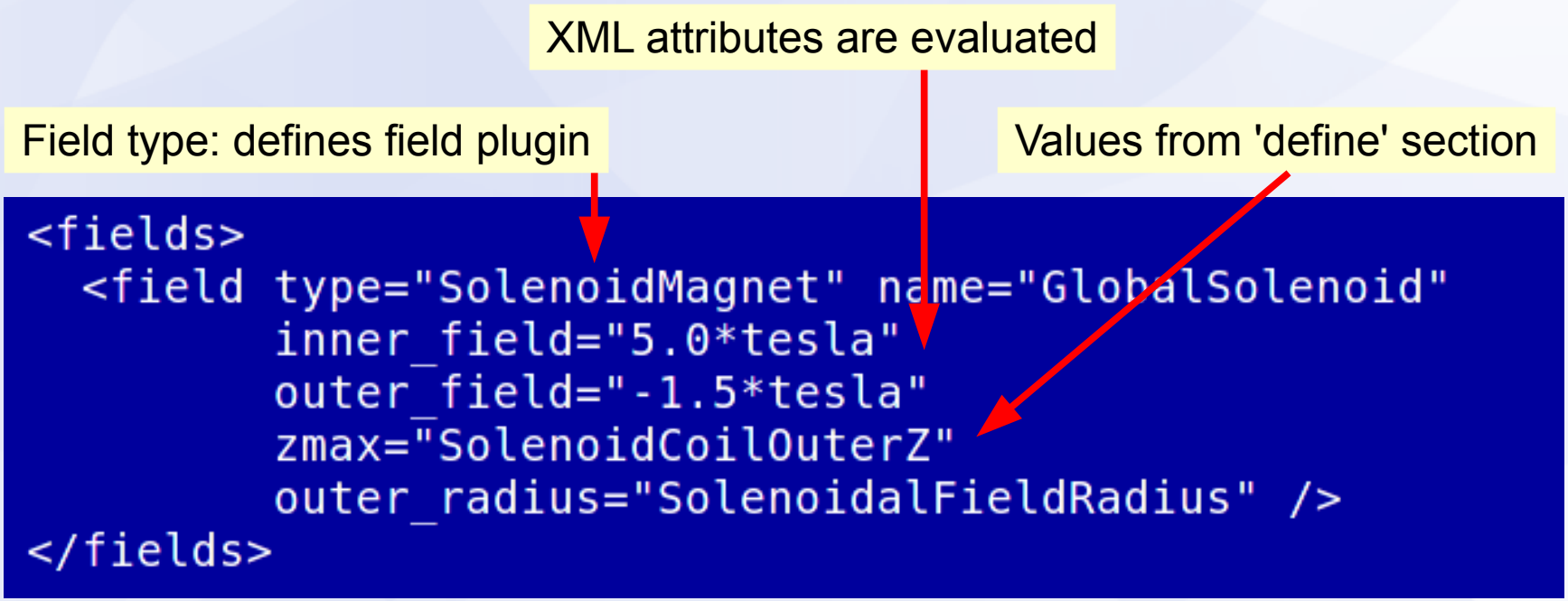
- **limits** *Limitsets for simulation*

```
<limits>
  <limitset name="calorimeter_limits">
    <!-- User Limits for G4UserLimits
     double maxStep  = 1.0;      // max step size in this volume
     double maxTrack = DBL_MAX; // max total track length
     double maxTime  = DBL_MAX; // max time
     double minEkine = 0;        // min kinetic energy  (Charged part)
     double minRange = 0;        // min remaining range (charged part) -->
    <limit name="step_length_max" particles="*" value="1" unit="mm" />
  </limitset>
<limits>
```

  - Geant 4 user limit settings
  - Used in simulation only
    - Limit names depend on interpretation of the simulation converter / simulation application

XML Compact Description – Structure

- **fields** *Electric/magnetic field definitions*



  - Attributes other than 'name' and 'type' depend on the field implementation, e.g. file name for field map

XML Compact Description – Structure

- **detectors** *Subdetector definitions*
  - *This is the core section*
- *All top level detector elements are defined here*
  - *Uniquely identified by name*

```
<detectors>
  <detector id="1" name="Beampipe" type="Tesla_tubeX01"
            vis="TubeVis">
        ...
  </detector>
  <detector id="2" name="VXD" type="Tesla_VXD03"
            vis="VXDVis" readout="VXDHits">
    <support .... />
        ...
  </detector>
</detectors>
```

Definition of a Top Level Element

- **Identified with the XML element <detector/> within the <detectors/> section**
  - **Mandatory XML attributes**
    - 'name' : Name of the top element
Unique identifier used to access the detector element
    - 'type': Constructor type
Trigger code execution
  - **Optional attributes**
    - 'limits': Name of the limitset
    - 'vis': Top level visualization attributes
    - 'readout': Name of the hits collection
Link to name of the readout description

Definition of a Top Level Element

- Children of the <detector/> element
  - No restrictions, whatever is required to construct the hierarchy of the subdetector
  - But: this freedom is also the door to chaos
    Positive example: SiD lccdd description
    - Modular and understandable and uniform

```
<detector name="..." type="...">
  <module name=...>
    ... additional elements ...
    <module_component attrs... />
  </module>
  <layer id="...">
    ... additional elements ...
  </layer>
</detector>
```

- Refresher of the Design
- Discussion of the different components
  - A few words to the C++ API
  - XML compact description and DTD structure
  - **Detector constructors**
  - Visualization
  - Detector Views

From SiD example: PolyconeSupport

$> gedit DDExamples/CLICSiD/compact/compact_polycones.xml

```
<detector name="LumiReadout_Forward" type="PolyconeSupport" vis="LumiCalVis">
  <comment>Readout for Luminosity Calorimeter</comment>
  <material name="G10"/>
  <zplane rmin="LumiCal_rmax" rmax="LumiCalElectronics_rmax"
          z="LumiCal_zmin"/>
  <zplane rmin="LumiCal_rmax" rmax="LumiCalElectronics_rmax"
          z="LumiCal_zmin+LumiCal_thickness"/>
</detector>

<detector name="LumiReadout_Backward" type="PolyconeSupport" vis="LumiCalVis">
  <comment>Readout for Luminosity Calorimeter</comment>
  <material name="G10"/>
  <zplane rmin="LumiCal_rmax" rmax="LumiCalElectronics_rmax"
          z="-LumiCal_zmin"/>
  <zplane rmin="LumiCal_rmax" rmax="LumiCalElectronics_rmax"
          z="-(LumiCal_zmin+LumiCal_thickness)"/>
</detector>
```

$> svn co https://svnsrv.desy.de/public/aidasoft/DD4hep/trunk/DDExamples

```
static Ref_t create_detector(LCDD& lcdd, const xml_h& e, SensitiveDetector&)  {
  xml_det_t  x_det    = e;
  string     name     = x_det.nameStr();
  DetElement sdet    (name,x_det.id());
  Material   mat     (lcdd.material(x_det.materialStr()));
  vector<double> rmin,rmax,z;
  int num = 0;

  for(xml_coll_t c(e,_X(zplane)); c; ++c, ++num)  {
    xml_comp_t dim(c);
    rmin.push_back(dim.rmin());
    rmax.push_back(dim.rmax());
    z.push_back(dim.z()/2);
  }
  if ( num < 2 )  {
    throw runtime_error("PolyCone["+name+"]> Not enough Z planes. minimum is 2!");
  }
  Polycone   cone   (0.,2.*M_PI,rmin,rmax,z);
  Volume     volume (name, cone, mat);
  volume.setVisAttributes(lcdd, x_det.visStr());
  sdet.setPlacement(lcdd.pickMotherVolume(sdet).placeVolume(volume));
  return sdet;
}

DECLARE_DETELEMENT(PolyconeSupport,create_detector);
```

1) Create Detector Element

3) Create volume: *Shape of given Material*

4) Place volume in mother

6) Publish constructor

DD4hep/DDExamples/CLICSiD/src/PolyconeSupport_geo.cpp

Same procedure: TubeSegment

```
static Ref_t create_element(LCDD& lcdd, const xml_h& e, SensitiveDetector&)  {
  xml_det_t  x_det  (e);
  xml_comp_t x_tube (x_det.child(_X(tubs)));
  xml_dim_t  pos    (x_det.child(_X(position)));
  xml_dim_t  rot    (x_det.child(_X(rotation)));
  string     name   = x_det.nameStr();
  Tube       tub    (x_tube.rmin(),x_tube.rmax(),x_tube.zhalf());
  Volume     vol    (name,tub,lcdd.material(x_det.materialStr()));

  vol.setVisAttributes(lcdd, x_det.visStr());

  DetElement   sdet(name,x_det.id());
  Volume       mother = lcdd.pickMotherVolume(sdet);
  PlacedVolume phv =
    mother.placeVolume(vol,Position(pos.x(),pos.y(),pos.z()),
                       Rotation(rot.x(),rot.y(),rot.z()));
  phv.addPhysVolID(_A(id),x_det.id());
  sdet.setPlacement(phv);
  return sdet;
}

DECLARE_DETELEMENT(TubeSegment,create_element);
```

DD4hep/DDExamples/CLICSiD/src/TubeSegment_geo.cpp

====== >>>> Innocent line with an important macro

Declares constructor function with a name to the ROOT plugin manager.
If your constructor is not found: macro missing or library not loaded.

A word about Placements (1)

- TGeo offers two possibilities
- Choice 1:
  First translate, then rotate the object around the three angles in the mother coordinate system
  - Arguments: first translation position
    second rotation angles

```
PlacedVolume pv =
  mother_vol.placeVolume(daughter_vol,
                         Position(x,y,z),
                         Rotation(rotx,roty,rotz))
```

A word about Placements (2)

- **Choice 2:**
  Rotate the mothers coordinate system around the three angles, then translate along these axis
  - **Arguments: first rotation angles**
    second translation position

```
PlacedVolume pv =
  mother_vol.placeVolume(daughter_vol,
                         Rotation(rotx,roty,rotz),
                         Position(x,y,z))
```

A word about Placements (3)

- Unclear if placements according to free and generalized TGeo transformations are necessary
- If yes: the call would look like this:

```
PlacedVolume pv =
  mother_vol.placeVolume(daughter_vol,
                         <TGeoMatrix*>)
```

- The TGeoMatrix instance must then be manipulated in the user code

A word about Placements (4)

- Simulation/Reconstruction hints
  - Also know as “copy-number” or equivalent
  - In agreement to the SiD geometry called 'volume id'
  - Map of named identifiers per volume

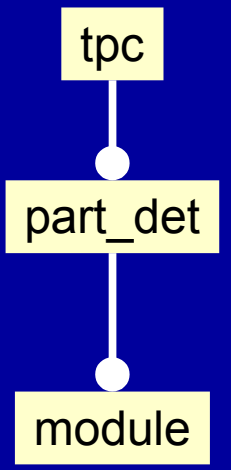
```
PlacedVolume pv = ...;
pv.addPhysVolID('module',identifier)
```

Detector Element Hierarchy

```
static Ref_t create_element(LCDD& lcdd, const xml_h& e, SensitiveDetector& sens)  {
  xml_det_t   x_det = e;
  string      name  = x_det.nameStr();
  DetElement tpc(name,x_det.typeStr(),x_det.id());

  DetElement  part_det(part_nam,px_det.typeStr(),px_det.id());
  for(xml_coll_t m(px_det, _X(modules)); m; ++m)  {
    string      m_nam  = modules.nameStr();
    for(xml_coll_t r(modules,_X(row)); r; ++r)  {
      xml_comp_t  row(r);
      int nmodules = row.nModules();
      int rowID=row.RowID();
      string      m_nam=m_name+_toString(rowID,"_Row%d")+_toString(md,"_M%d");
      DetElement  module(part_det,m_nam,mdcount);

      module.addExtension<PadLayout>(new RectangularPadRowLayout(module));
    }//modules
  }//rows
 }//module groups
 //endplate
 tpc.add(part_det);
...
}
```



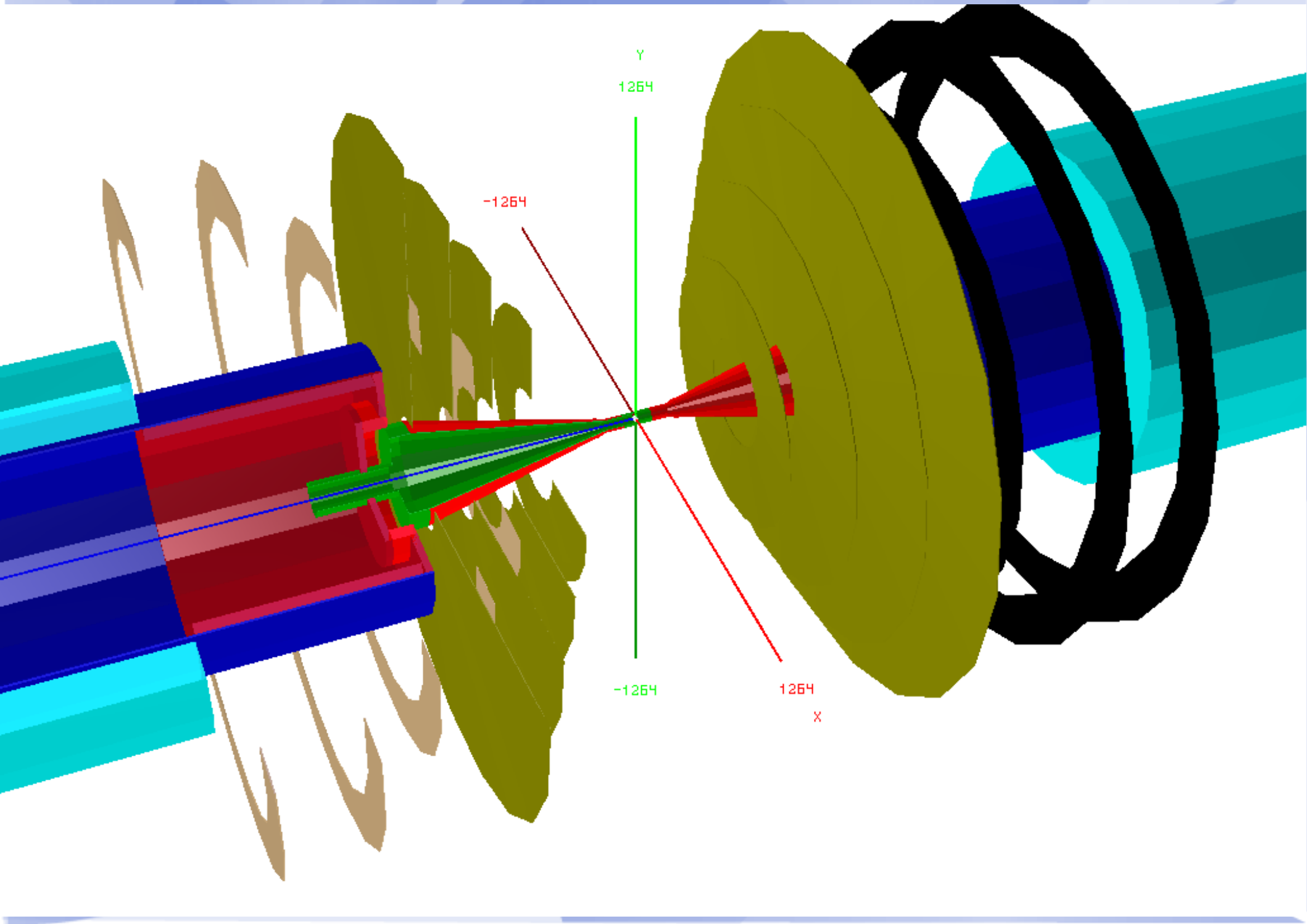
Parent-child relationship through constructor

Parent-child relation through explicit call

DD4hep/DDExamples/ILDExDet/src/compact/TPCPrototype_geo.cpp

Visualize it

- Currently there several programs to visualize geometries
  - This is bad and an artefact of
  - my ignorance of customizing cmake,
  - because I cannot use the full capabilities of the ROOT plugin manager
- This will be corrected
- ./DDExamples/CLICSiDDisplay/CLICSiDtest \
  file:../DD4hep/DDExamples/CLICSiD/compact/compact_polycones.xml


Y
1264
-1264
-1264
1264
X

Client Extensions

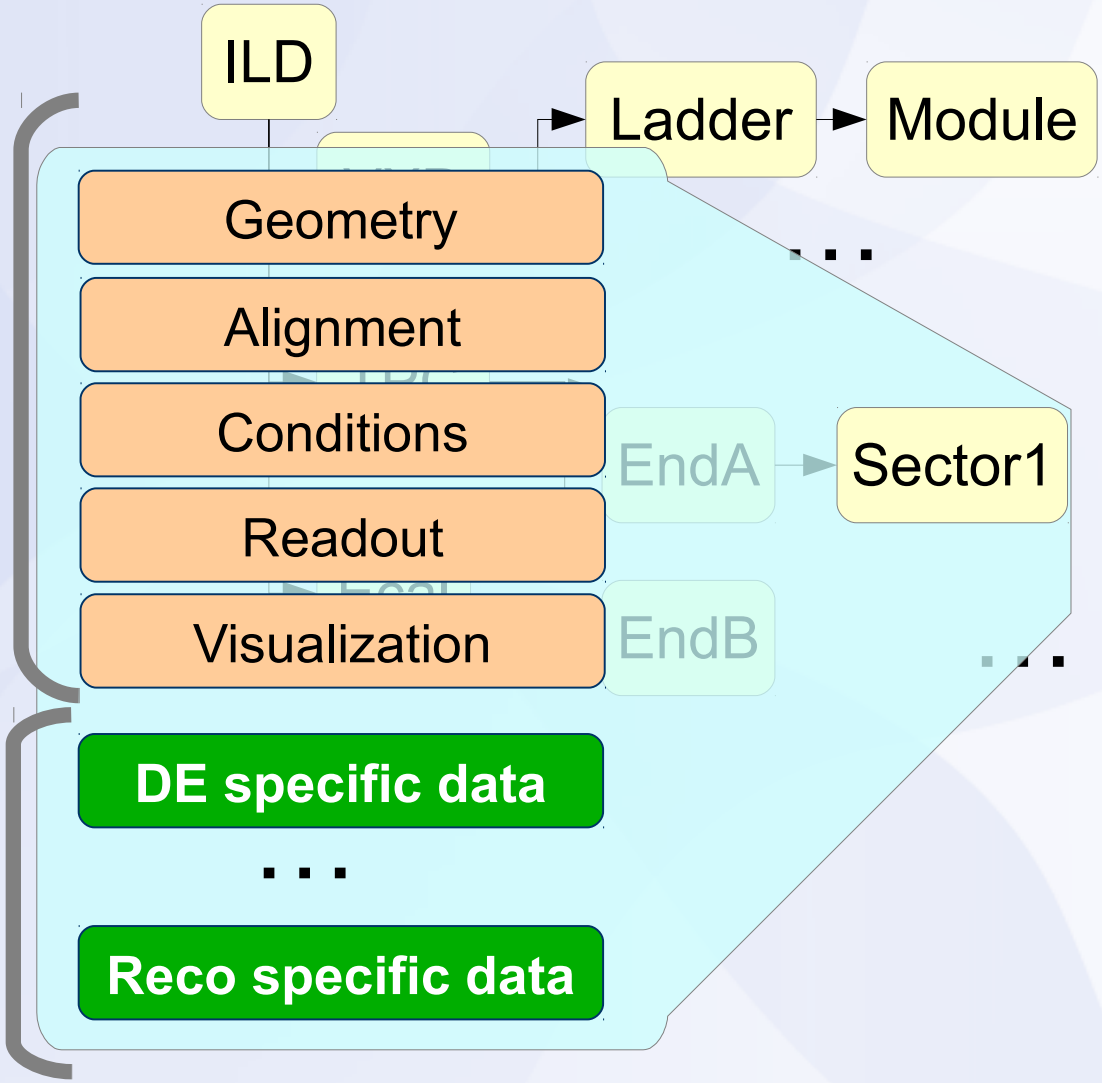
- **Different use cases require different functionality**
  - **Example: Optimization of coordinate transformations local TPC hit to experiment coordinates => specialized data required (cache of precomputed results)**
  - **Need to extend the detector element's data**

Implementation: Client Extensions

- Functionality achieved by 'views'
  - Corollary of the design choice to separate 'data' from 'behavior'
  - Possibility of many views based on the same data
    - All views share the same data __OR__
    - Same 'data' can be associated to different 'behaviors'
    - All views are consistent
  - Public data describing a detector
    - User objects may be attached to data
  - Views are 'handles' to the data
    - Creating views is efficient and fast
    - Typically only a pointer needs to be copied

Client Extensions

- **Default DetElement data**
- **Added subdetector specific data**



Example: TPC (A.Muennich)

- View of a TPC module as seen by the user code
  - PadLayout is user defined (different implementations)

```
struct TPCModule : public Geometry::DetElement {
  PadLayout* padLayout;

  std::string getPadType() const;
  int getNPads() const;
  int getNRows() const;
  int  getNPadsInRow(int row) const;
  double getRowHeight (int row) const;
  ....
};


int TPCModule::getNPads()  const {
  return padLayout->getNPads();
}
```

DD4hep/DDExamples/ILDExDet/include/TPCModule.h

Example: TPC (A.Muennich)

- In the shared extensions data may be cached
- And/or complex calculations may be done

```
std::vector<double>  RectangularPadRowLayout::getPadCenter (int pad) const {
  if(pad>getNPads())
    throw OutsideGeometryException("getPadCenter: Requested pad not on module querried!");
  int row=getRowNumber(pad);
  //shift coordinates from pad system where 0,0 is on lower left corner of module into module
  //system where 0,0 is in the middle of the module box
  double pad_y=(row+0.5)*getRowHeight(0)-box->GetDY();
  double pad_x = (getPadNumber(pad)+0.5)*getPadPitch(pad)-box->GetDX();
  //trafo to global coordinates
  Position global_w, local(pad_x,pad_y,0);
  module.localToWorld(local,global_w);

  vector<double> center;
  center.push_back(global_w.x);
  center.push_back(global_w.y);
  return center;
}
```

DD4hep/DDExamples/ILDExDet/src/RectangularPadRowLayout.cpp

Example: TPC (A.Muennich)

- How does the pointer to the PadLayout appear in the detector element ?
- Where is this done ?

TPC – Detector Constructor

```
static Ref_t create_element(LCDD& lcdd, const xml_h& e, SensitiveDetector& sens)  {
  xml_det_t   x_det = e;
  string      name  = x_det.nameStr();
  DetElement tpc(name,x_det.typeStr(),x_det.id());

  for(xml_coll_t c(e,_X(detector)); c; ++c)  {
    DetElement  part_det(part_nam,px_det.typeStr(),px_det.id());
    for(xml_coll_t m(px_det,_X(modules)); m; ++m)  {
      xml_comp_t  modules  (m);
      string      m_name  = modules.nameStr();
      for(xml_coll_t r(modules,_X(row)); r; ++r)  {
        xml_comp_t  row(r);
        int nmodules = row.nModules();
        int rowID=row.RowID();
        for(int md=0;md<nmodules;md++)  {          //placing modules
          string      m_nam=m_name+_toString(rowID,"_Row%d")+_toString(md,"_M%d");
          DetElement  module(part_det,m_nam,mdcount);
          ...
          module.addExtension<PadLayout>(new RectangularPadRowLayout(module));
        }//modules
      }//rows
    }//module groups
  }//endplate
  tpc.add(part_det);
...
}
```

DD4hep/DDExamples/ILDExDet/src/compact/TPCPrototype_geo.cpp

TPC – Detector Constructor

```
DetElement  module(part_det,m_nam,mdcount);
PadLayout*  pl = new RectangularPadRowLayout(module);
module.addExtension<PadLayout>(pl);
```

Detector element to extend

Public type of the extension object
(May be ABC or interface like here)

Extension object

- **Any number of extensions**
  - **Must differ by public type**
- **Adding an extension is possible anywhere**
  - **Only happens to be here in the detector constructor**
  - **Could also be somewhere in the reconstruction code**

TPC Module View

```
TPCModule(const Geometry::DetElement& e)
: Geometry::DetElement(e), padLayout(0)
{
  getExtension();
}

void TPCModule::getExtension() {
  padLayout = isValid() ? extension<PadLayout>() : 0;
}
```

DD4hep/DDExamples/ILDExDet/src/TPCModule.cpp

- **The PadLayout is retrieved from the detector element if present**
  - **Lookup relatively cheap, but not for free Hence: extension pointer is cached**
  - **Map lookup by type_info**

Simple Views

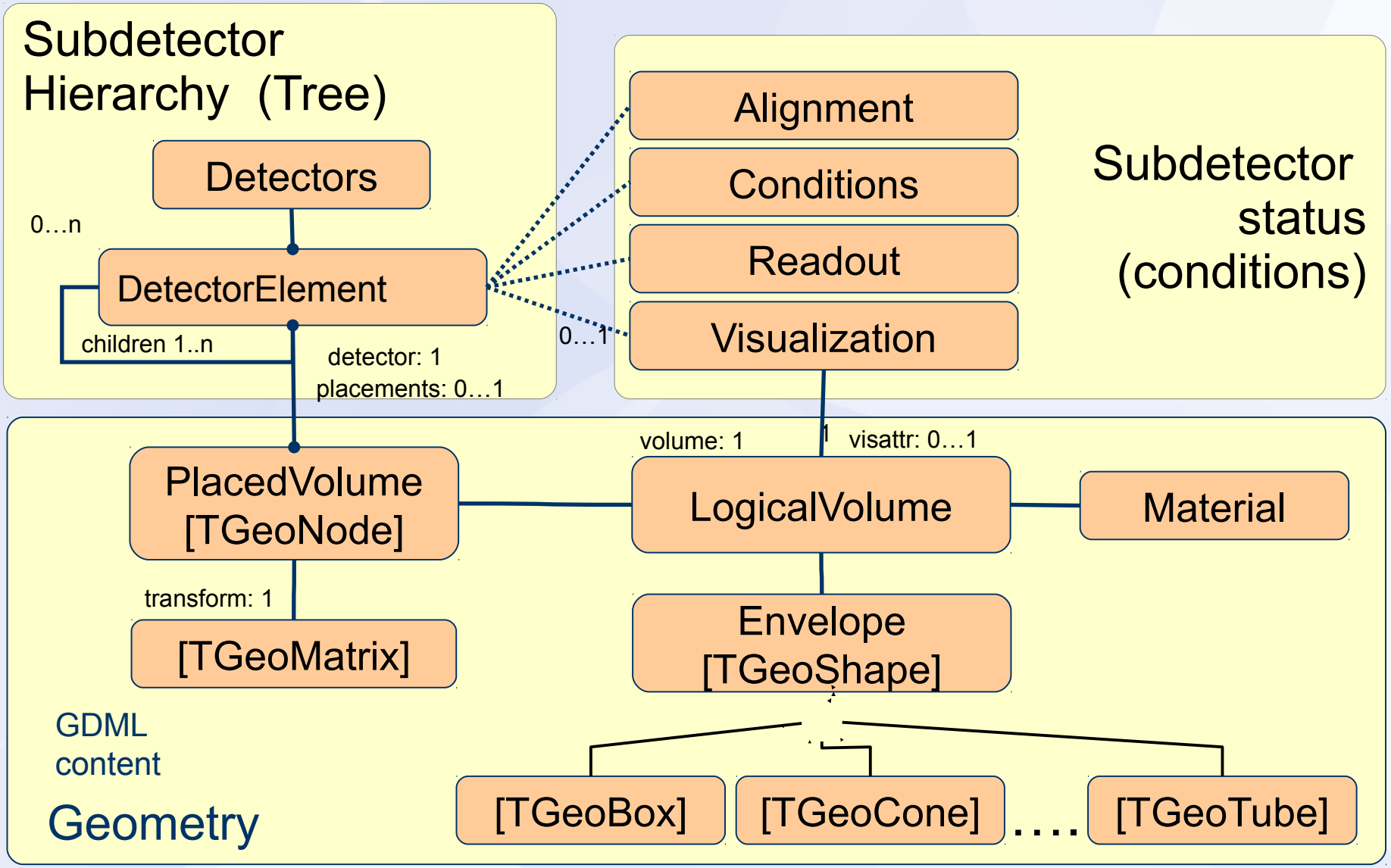
- Some views may not require any additional data
  - if only a few operations and/or navigations should be combined
- Then no extension object needs to be defined
- Directly implement the view using the DetElement data

The End

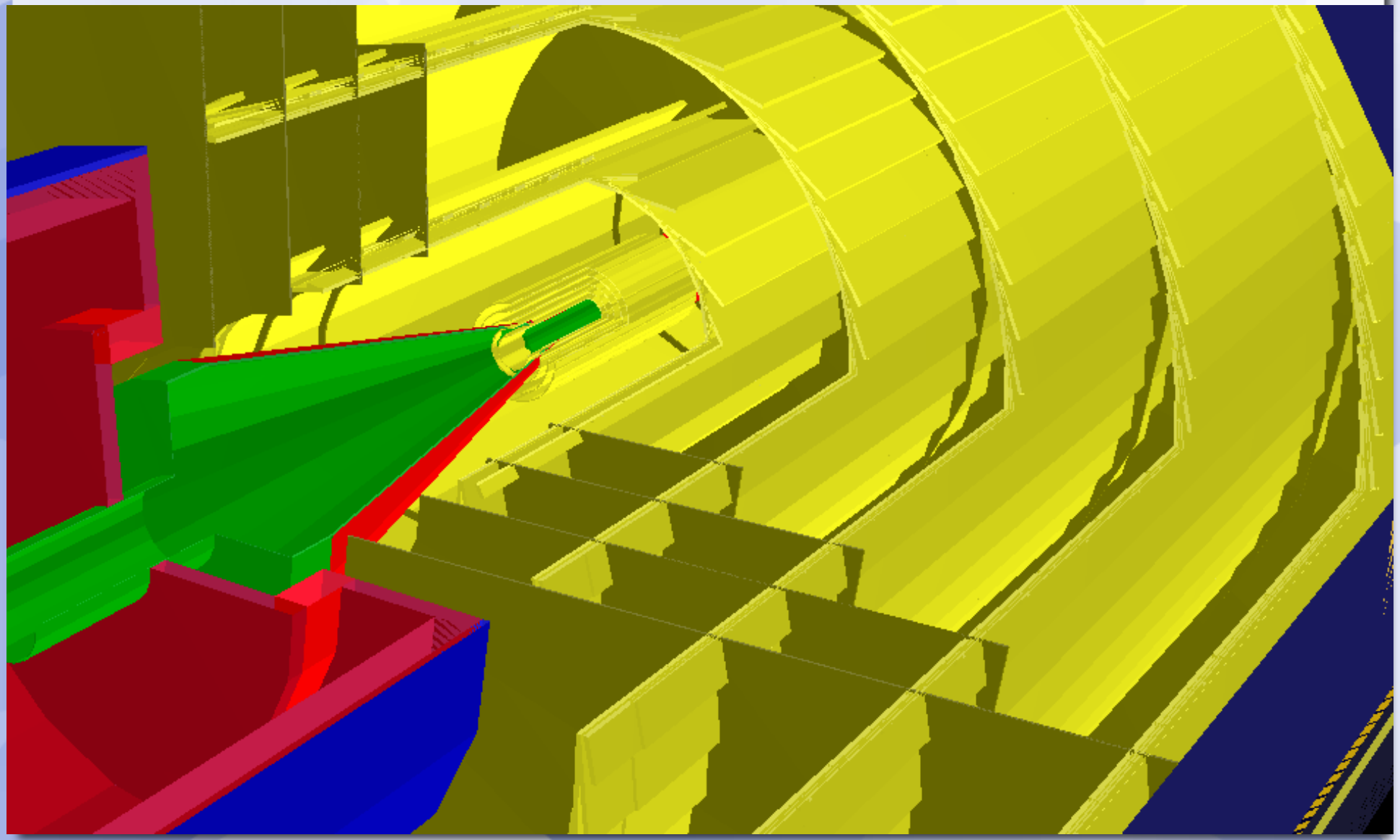
Any Questions ?

Backup Slides

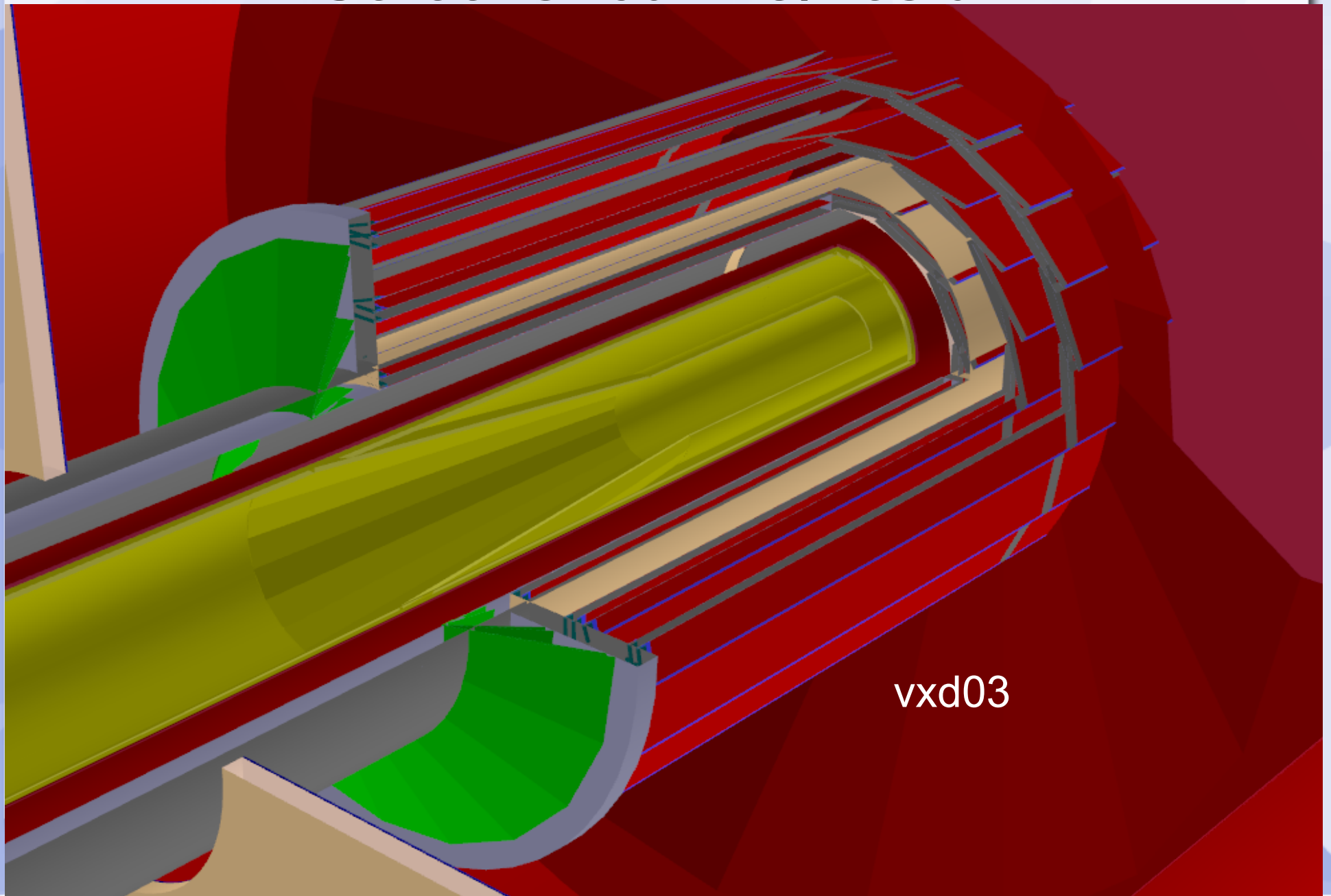
Implementation: Geometry



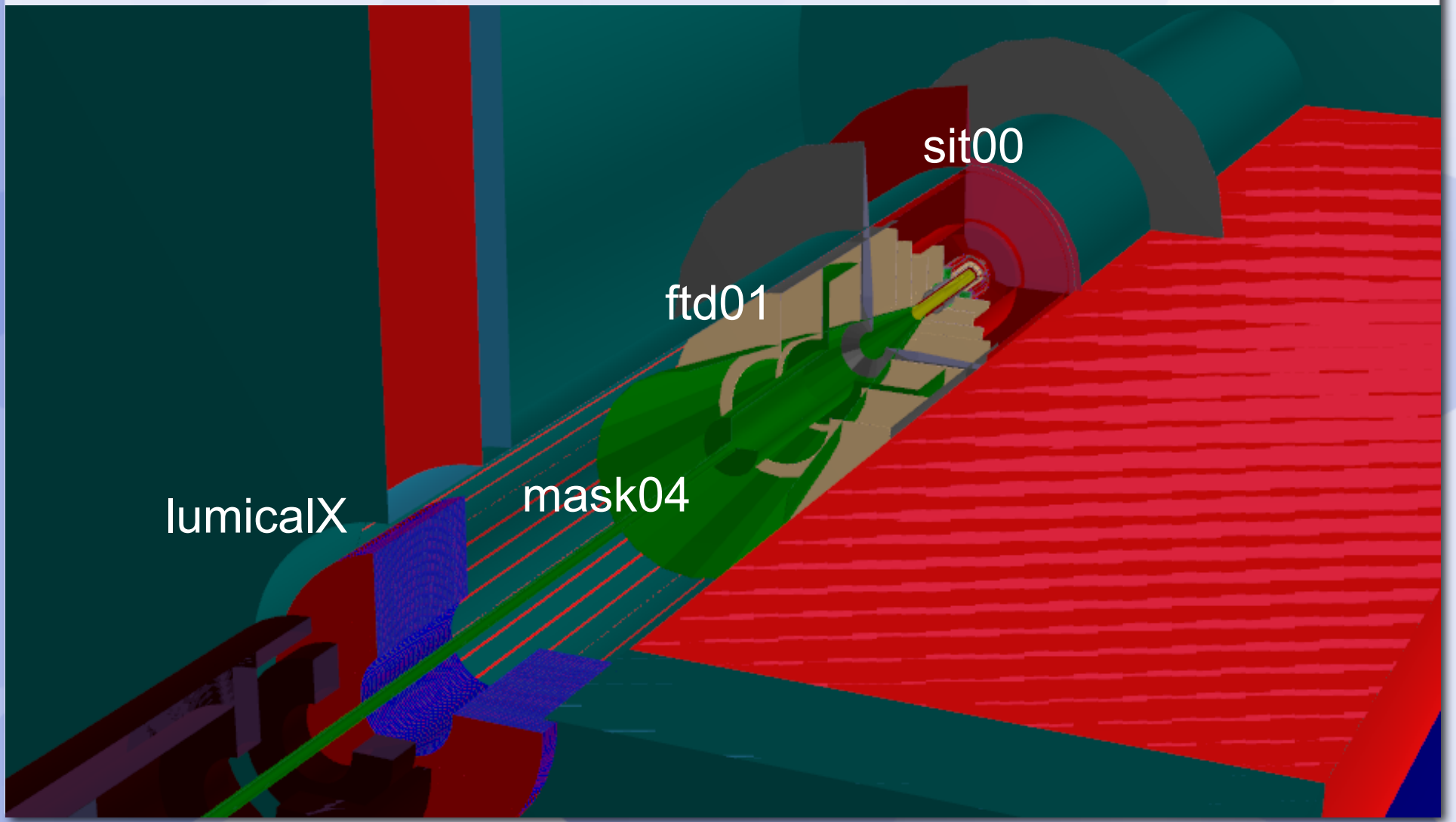
Screenshot: SiD



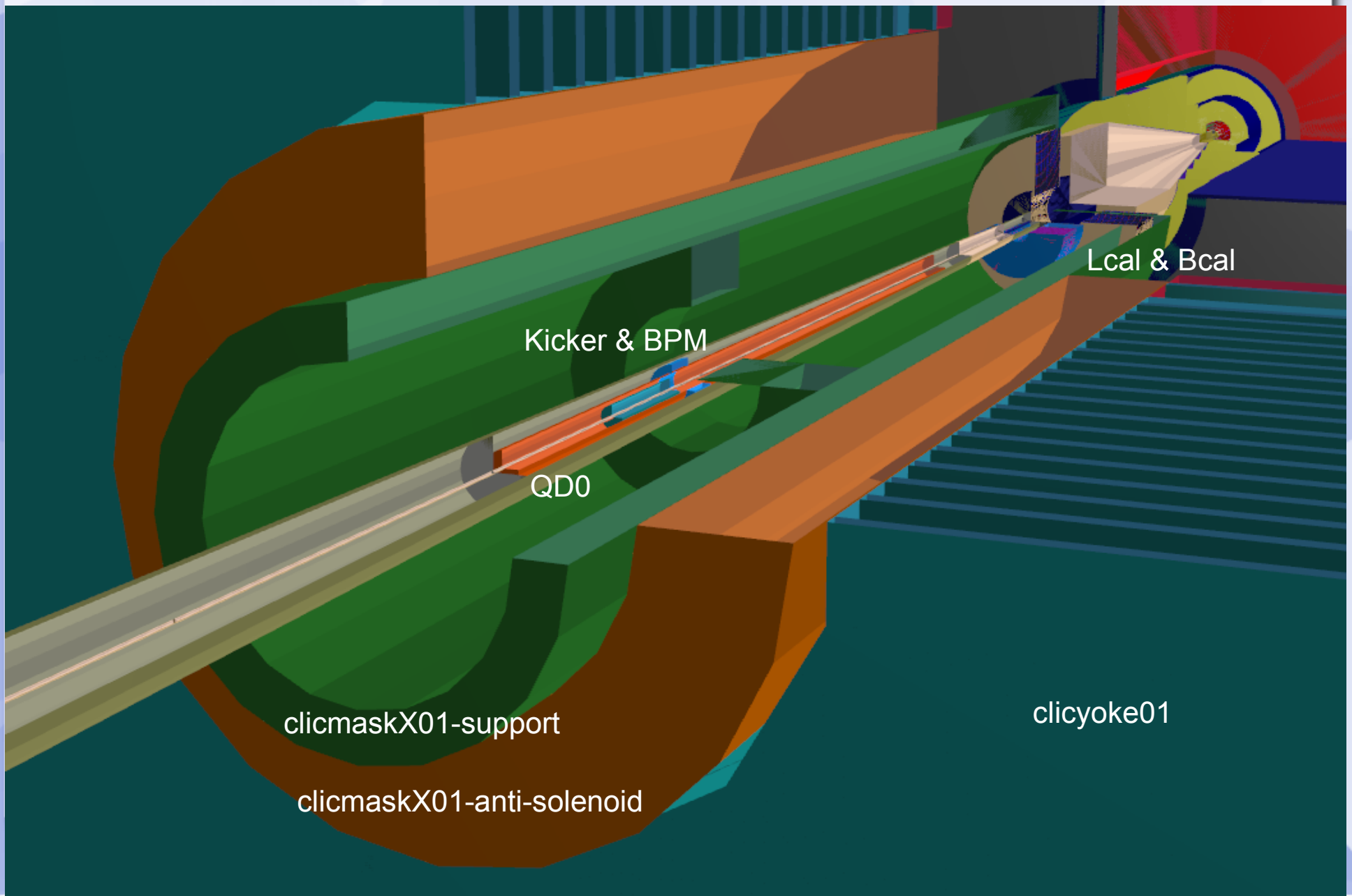
Screenshot: ILC/Tesla



Screenshot: ILC/Tesla



Screenshot: ILC/Tesla



Screenshot: ILC/Tesla