Geant 4

Detector Description: Basics

http://cern.ch/geant4

PART III

Detector Description: the Basics

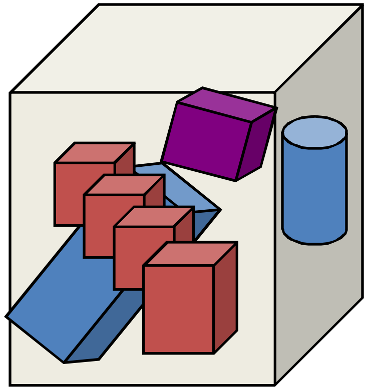
- *Parameterised Volumes and Replicas*
- *Detector description persistency: GDML*

Describing a detector - IV

- *Parameterised Volumes and Replicas*

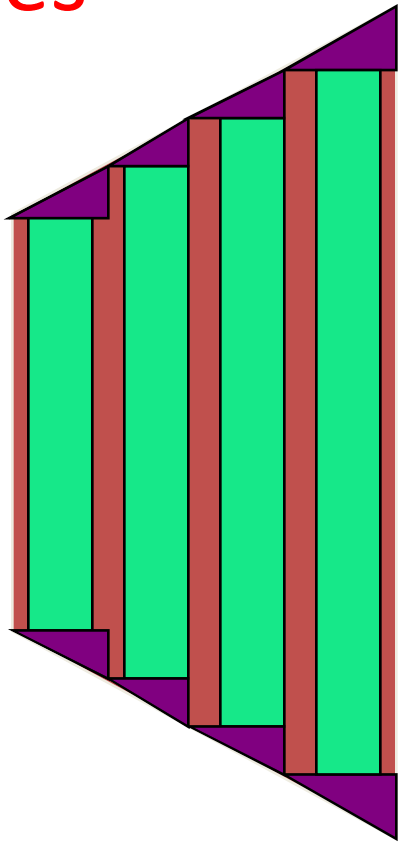
Parameterised Physical Volumes

- User written functions define:
  - the size of the solid (dimensions)
    - Function **`ComputeDimensions(…)`**
  - where it is positioned (transformation)
    - Function **`ComputeTransformations(…)`**
- Optional:
  - the type of the solid
    - Function **`ComputeSolid(…)`**
  - the material
    - Function **`ComputeMaterial(…)`**
- Limitations:
  - Applies to a limited set of solids
  - Daughter volumes allowed only for special cases
- Very powerful
  - Consider parameterised volumes as "leaf" volumes



Uses of Parameterised Volumes

- Complex detectors
  - with large repetition of volumes
    - regular or irregular
- Medical applications
  - the material in animal tissue is measured
    - cubes with varying material



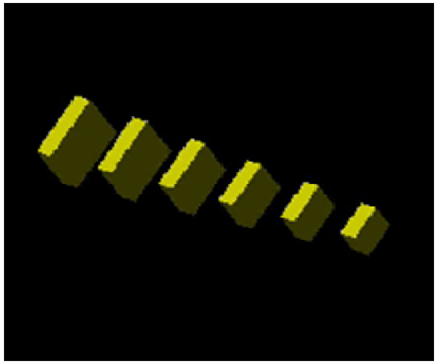
G4PVParameterised

```
G4PVParameterised(const G4String& pName,
                  G4LogicalVolume* pCurrentLogical,
                  G4LogicalVolume* pMotherLogical,
                  const EAxis pAxis,
                  const G4int nReplicas,
                  G4VPVParameterisation* pParam,
                  G4bool pSurfChk=false);
```

- Replicates the volume **`nReplicas`** times using the parameterisation **`pParam`**, within the mother volume
- The positioning of the replicas is dominant along the specified Cartesian axis
  - If **`kUndefined`** is specified as axis, 3D voxelisation for optimisation of the geometry is adopted
- Represents many touchable detector elements differing in their positioning and dimensions. Both are calculated by means of a **`G4VPVParameterisation`** object
- Alternative constructor using pointer to physical volume for the mother

Parameterisation

example - 1



```
G4VSolid* solidChamber = new G4Box("chamber", 100*cm, 100*cm, 10*cm);
G4LogicalVolume* logicChamber =
   new G4LogicalVolume(solidChamber, ChamberMater, "Chamber", 0, 0, 0);
G4double firstPosition = -trackerSize + 0.5*ChamberWidth;
G4double firstLength = fTrackerLength/10;
G4double lastLength  = fTrackerLength;
G4VPVParameterisation* chamberParam =
   new ChamberParameterisation( NbOfChambers, firstPosition,
                                ChamberSpacing, ChamberWidth,
                                firstLength, lastLength);
G4VPhysicalVolume* physChamber =
   new G4PVParameterised( "Chamber", logicChamber, logicTracker,
                          kZAxis, NbOfChambers, chamberParam);
```

Use **kUndefined** for activating 3D voxelisation for optimisation

Parameterisation
example - 2

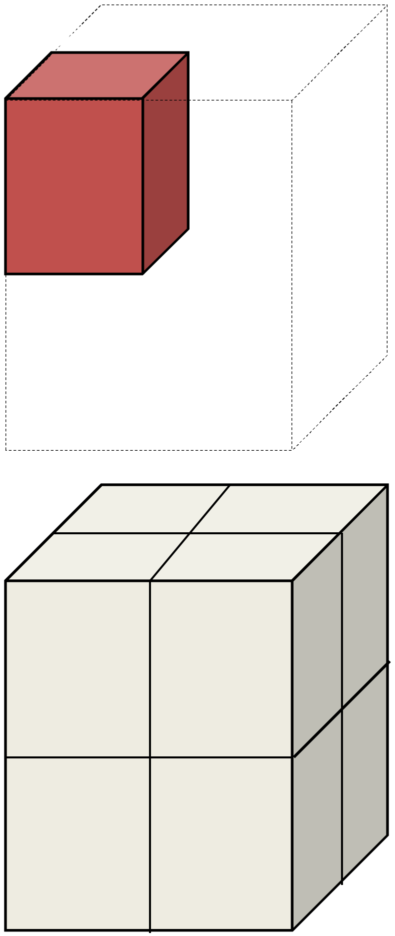
```
class ChamberParameterisation : public G4VPVParameterisation
{
  public:
    ChamberParameterisation( G4int NoChambers, G4double startZ,
                             G4double spacing, G4double widthChamber,
                             G4double lenInitial, G4double lenFinal );
   ~ChamberParameterisation();
   void ComputeTransformation (const G4int copyNo,
                               G4VPhysicalVolume* physVol) const;
   void ComputeDimensions (G4Box& trackerLayer, const G4int copyNo,
                           const G4VPhysicalVolume* physVol) const;
}
```

Parameterisation
example - 3

```
void ChamberParameterisation::ComputeTransformation
(const G4int copyNo, G4VPhysicalVolume* physVol) const
{
  G4double Zposition= fStartZ + (copyNo+1) * fSpacing;
  G4ThreeVector origin(0, 0, Zposition);
  physVol->SetTranslation(origin);
  physVol->SetRotation(0);
}
void ChamberParameterisation::ComputeDimensions
(G4Box& trackerChamber, const G4int copyNo,
 const G4VPhysicalVolume* physVol) const
{
  G4double  halfLength= fHalfLengthFirst + copyNo * fHalfLengthIncr;
  trackerChamber.SetXHalfLength(halfLength);
  trackerChamber.SetYHalfLength(halfLength);
  trackerChamber.SetZHalfLength(fHalfWidth);
}
```

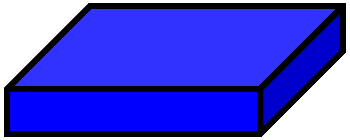
Replicated Physical Volumes

- The mother volume is sliced into replicas, all of the same size and dimensions.
- Represents many touchable detector elements differing only in their positioning.
- Replication may occur along:
  - Cartesian axes (X, Y, Z) – slices are considered perpendicular to the axis of replication
    - Coordinate system at the center of each replica
  - Radial axis (Rho) – cons/tubs sections centered on the origin and un-rotated
    - Coordinate system same as the mother
  - Phi axis (Phi) – phi sections or wedges, of cons/tubs form
    - Coordinate system rotated such as that the X axis bisects the angle made by each wedge

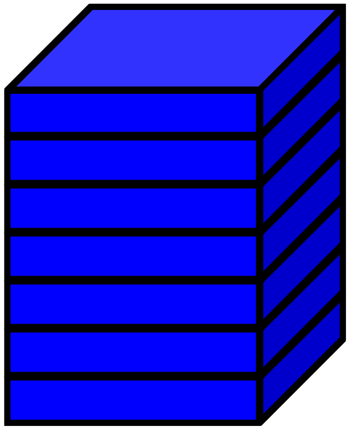


repeated

G4PVReplica




```
G4PVReplica(const G4String& pName,
            G4LogicalVolume* pCurrentLogical,
            G4LogicalVolume* pMotherLogical,
            const EAxis pAxis,
            const G4int nReplicas,
            const G4double width,
            const G4double offset=0);
```

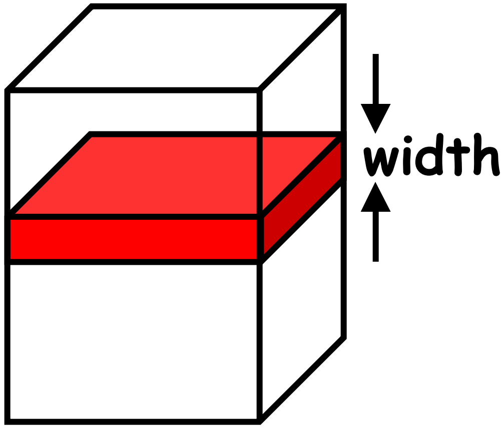


- Alternative constructor:
  - Using pointer to physical volume for the mother
- An **`offset`** can be associated
  - Only to a mother offset along the axis of replication
- Features and restrictions:
  - Replicas can be placed inside other replicas
  - Normal placement volumes can be placed inside replicas, assuming no intersection or overlaps with the mother volume or with other replicas
  - No volume can be placed inside a *radial* replication
  - Parameterised volumes cannot be placed inside a replica

Replica – axis, width, offset

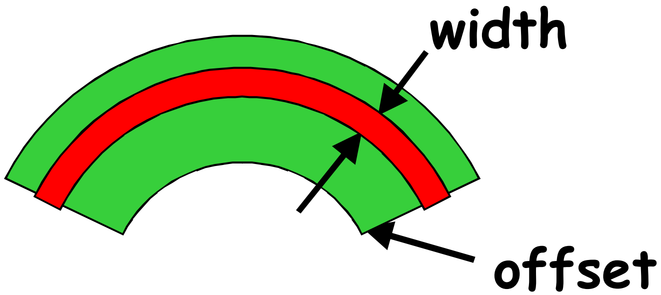
- Cartesian axes - **kXaxis, kYaxis, kZaxis**
  - offset shall not be used
  - Center of n-th daughter is given as

    -width*(nReplicas-1)*0.5+n*width



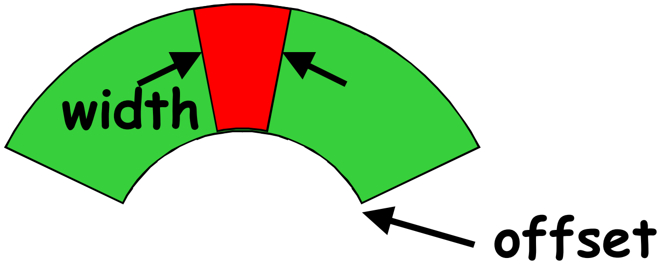
- Radial axis - **kRaxis**
  - Center of n-th daughter is given as

    width*(n+0.5)+offset



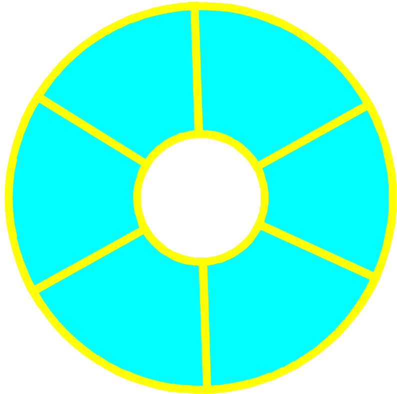
- Phi axis - **kPhi**
  - Center of n-th daughter is given as

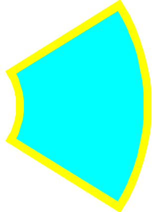
    width*(n+0.5)+offset



Replication
example

```
G4double tube_dPhi = 2.* M_PI * rad;
G4VSolid* tube =
   new G4Tubs("tube",20*cm,50*cm,30*cm,0.,tube_dPhi);
G4LogicalVolume * tube_log =
   new G4LogicalVolume(tube, Air, "tubeL", 0, 0, 0);
G4VPhysicalVolume* tube_phys =
   new G4PVPlacement(0,G4ThreeVector(-200.*cm,0.,0.),
            "tubeP", tube_log, world_phys, false, 0);
G4double divided_tube_dPhi = tube_dPhi/6.;
G4VSolid* div_tube =
   new G4Tubs("div_tube", 20*cm, 50*cm, 30*cm,
        -divided_tube_dPhi/2., divided_tube_dPhi);
G4LogicalVolume* div_tube_log =
   new G4LogicalVolume(div_tube,Pb,"div_tubeL",0,0,0);
G4VPhysicalVolume* div_tube_phys =
   new G4PVReplica("div_tube_phys", div_tube_log,
   tube_log, kPhi, 6, divided_tube_dPhi);
```



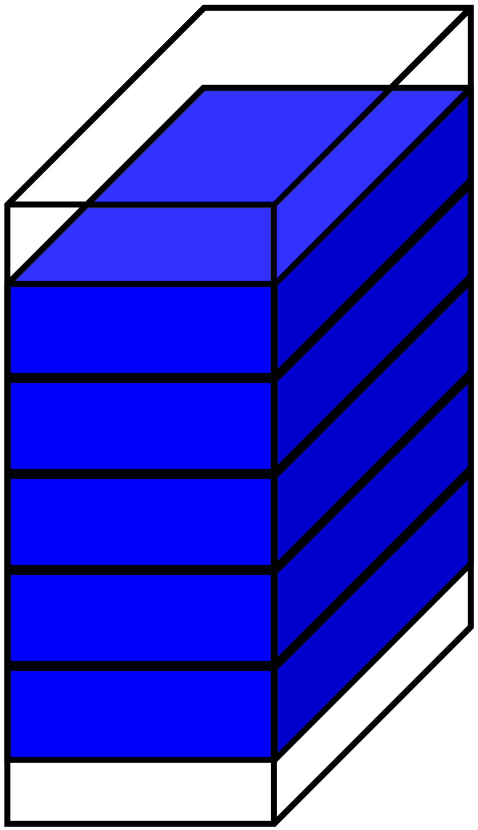


Divided Physical Volumes

- Implemented as “special” kind of parameterised volumes
  - Applies to CSG-like solids only (box, tubs, cons, para, trd, polycone, polyhedra)
  - Divides a volume in identical copies along one of its axis (copies are not strictly identical)
    - e.g. - a tube divided along its radial axis
    - Offsets can be specified
- The possible axes of division vary according to the supported solid type
- Represents many touchable detector elements differing only in their positioning
- **`G4PVDivision`** is the class defining the division
  - The parameterisation is calculated automatically using the values provided in input

Divided Volumes - 2

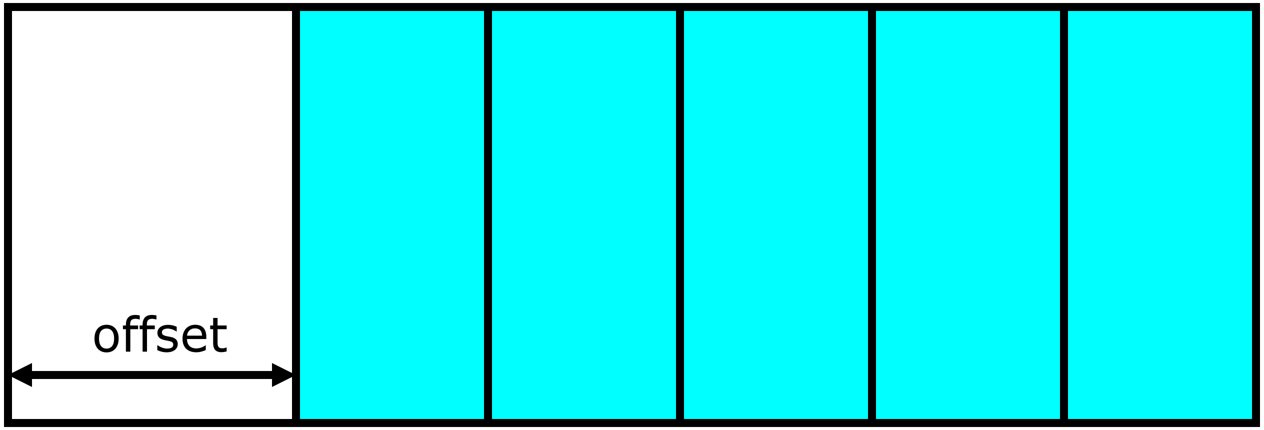
- `G4PVDivision` is a special kind of parameterised volume
  - The parameterisation is automatically generated according to the parameters given in `G4PVDivision`.
- Divided volumes are similar to replicas but …
  - Allowing for gaps in between mother and daughter volumes
    - *Planning to allow also gaps between daughters and gaps on side walls*
- Shape of all daughter volumes must be same shape as the mother volume
  - Solid (to be assigned to the daughter logical volume) must be the same type, but different object.
- Replication must be aligned along one axis
- If no gaps in the geometry, `G4PVReplica` is recommended
  - For identical geometry, navigation in pure replicas is faster



Divided Volumes - 3

```
G4PVDivision(const G4String& pName,
             G4LogicalVolume* pDaughterLogical,
             G4LogicalVolume* pMotherLogical,
             const EAxis pAxis,
             const G4int nDivisions,    // number of division is given
             const G4double offset);
```

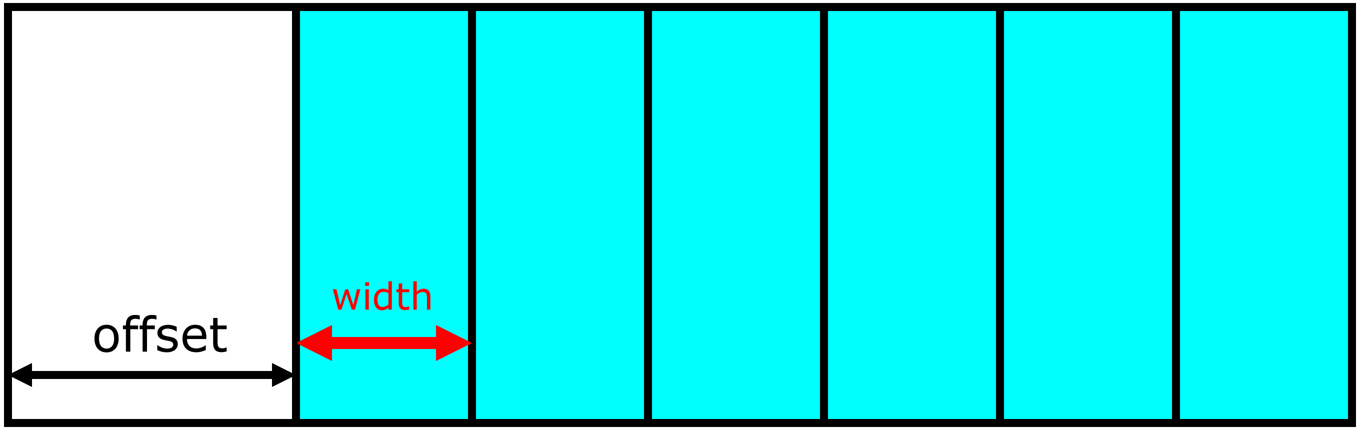
- The size (width) of the daughter volume is calculated as
  `( (size of mother) - offset ) / nDivisions`



Divided Volumes - 4

```
G4PVDivision(const G4String& pName,
             G4LogicalVolume* pDaughterLogical,
             G4LogicalVolume* pMotherLogical,
             const EAxis pAxis,
             const G4double width,    // width of daughter volume is given
             const G4double offset);
```

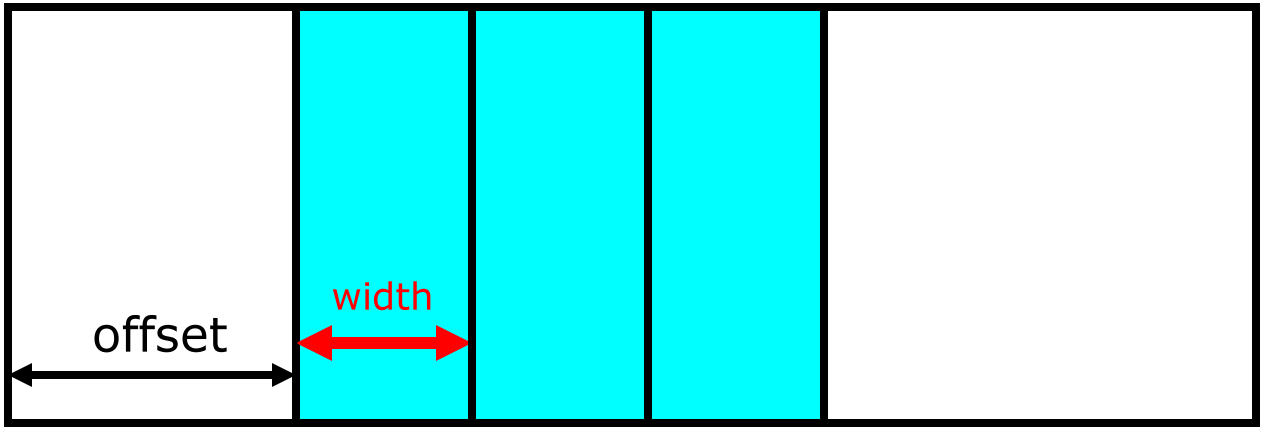
- The number of daughter volumes is calculated as
  `int( ( (size of mother) – offset ) / width )`
- As many daughters as width and offset allow



Divided Volumes - 5

```
G4PVDivision(const G4String& pName,
             G4LogicalVolume* pDaughterLogical,
             G4LogicalVolume* pMotherLogical,
             const EAxis pAxis,
             const G4int nDivisions,  // both number of divisions
             const G4double width,    // and width are given
             const G4double offset);
```

- *nDivisions* daughters of *width* thickness



Divided Volumes - 6

- Divisions are allowed for the following shapes / axes:
  - **`G4Box : kXAxis, kYAxis, kZAxis`**
  - **`G4Tubs : kRho, kPhi, kZAxis`**
  - **`G4Cons : kRho, kPhi, kZAxis`**
  - **`G4Trd : kXAxis, kYAxis, kZAxis`**
  - **`G4Para : kXAxis, kYAxis, kZAxis`**
  - **`G4Polycone : kRho, kPhi, kZAxis`**
  - **`G4Polyhedra : kRho, kPhi, kZAxis`**
    - **`kPhi`** - the number of divisions has to be the same as solid sides, (i.e. **`numSides`**), the width will not be taken into account
- In the case of division along **`kRho`** of **`G4Cons, G4Polycone, G4Polyhedra`**, if width is provided, it is taken as the width at the **`-Z`** radius; the width at other radii will be scaled to this one

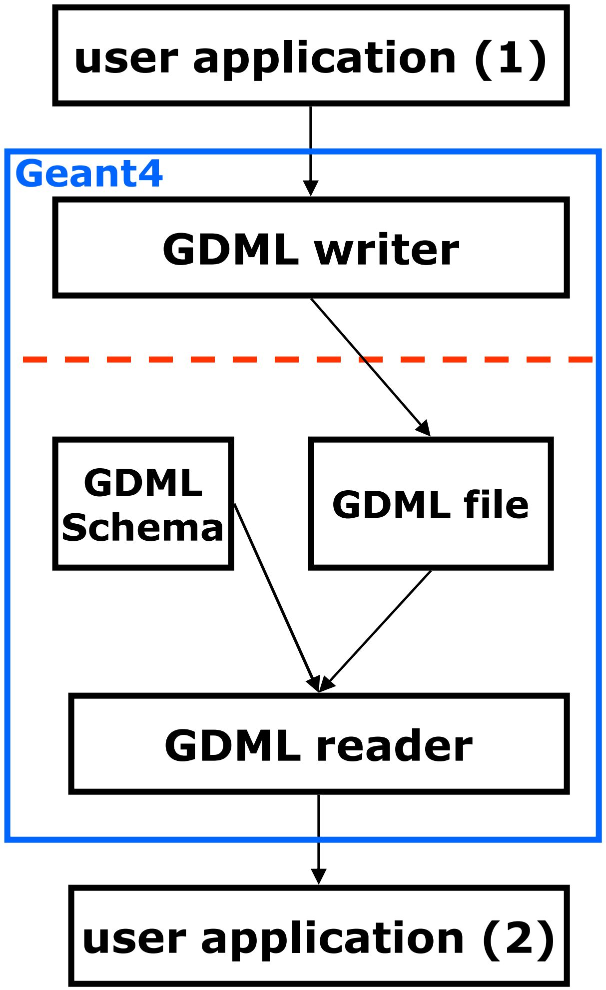
GDML

- *Importing and exporting detector descriptions*

GDML components

- GDML (Geometry Description Markup Language) is defined through XML Schema (XSD)
  - XSD = XML based alternative to Document Type Definition (DTD)
  - defines document structure and the list of legal elements
  - XSD are in XML -> they are extensible

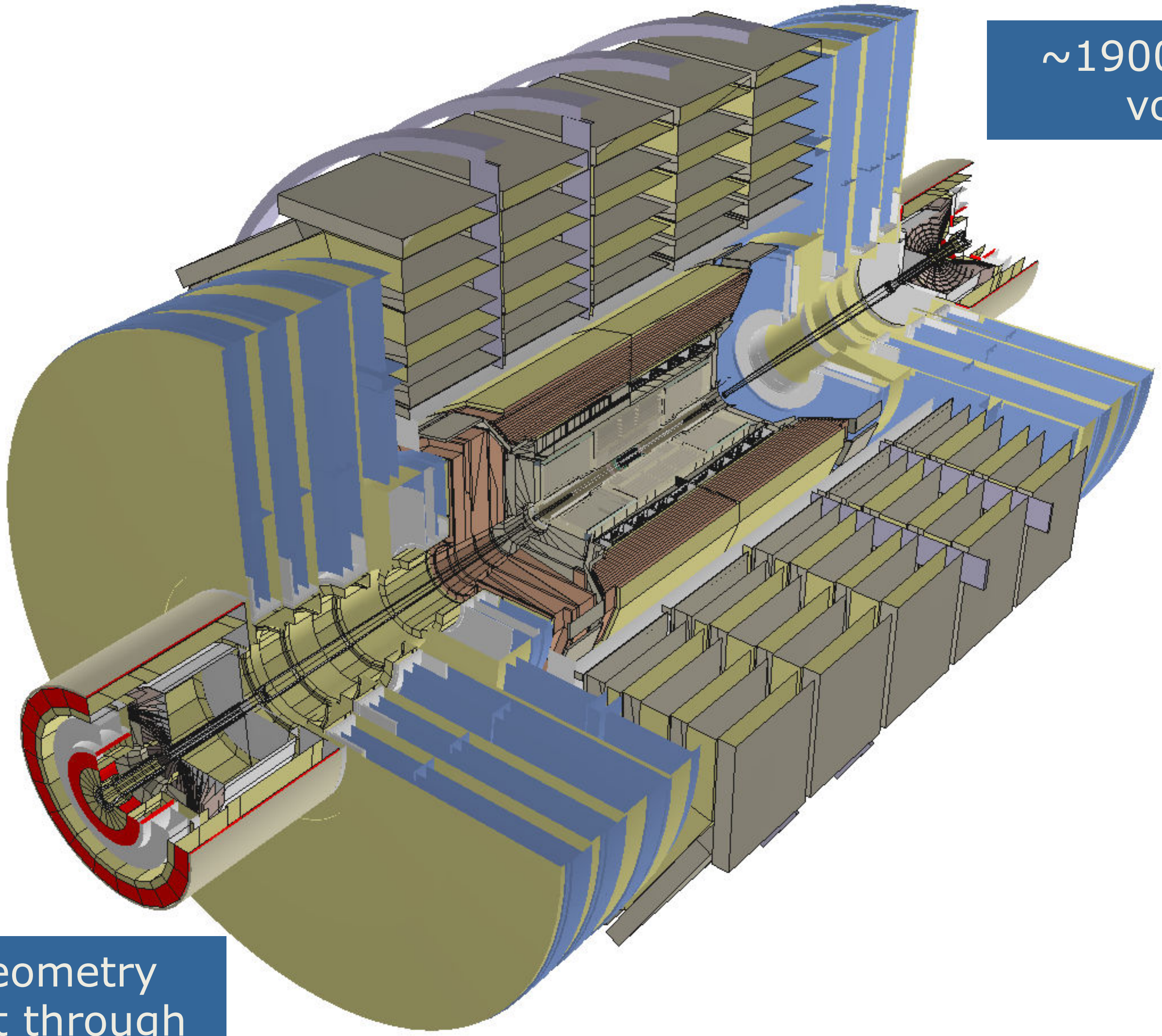
- GDML can be written by hand or generated automatically in Geant4
  - 'GDML writer' allows exporting a GDML file
- GDML needs a "reader", integrated in Geant4
  - 'GDML reader' imports and creates 'in-memory' the representation of the geometry description



GDML – Geant4 binding

- XML schema available from http://cern.ch/gdml
  - Also available within Geant4 distribution
    - See in **`geant4/source/persistency/gdml/schema/`**
  - Latest schema release GDML_3_0_0 (as from 9.2 release)
- Requires XercesC++ XML parser
  - Available from: http://xerces.apache.org/xerces-c
  - Tested with versions 2.8.0 and 3.0.1
- Optional package to be linked against during build
  - **`G4LIB_BUILD_GDML`** and **`XERCESCROOT`** variables
  - Examples available: **`geant4/examples/extended/persistency/gdml`**

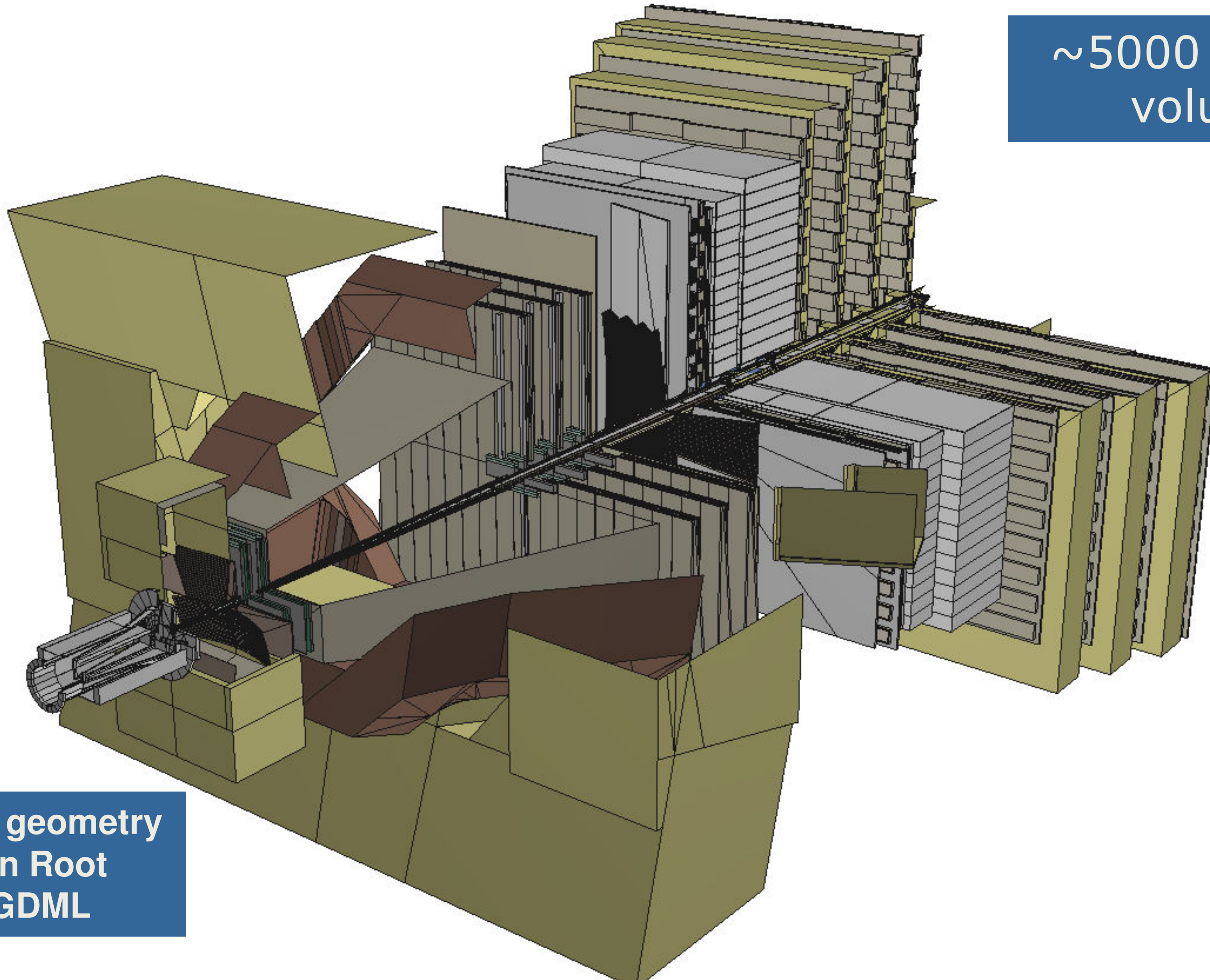
CMS detector through GDML



~19000 physical volumes

Geant4 CMS geometry imported in Root through GDML

LHCb detector through GDML



~5000 physical volumes

Geant4 LHCb geometry imported in Root through GDML

Using GDML in Geant4

to write:

```
#include "G4GDMLParser.hh"
G4GDMLParser parser;
parser.Write("g4test.gdml", pWorld, true, "path_to_schema/gdml.xsd");
```

instantiate GDML parser

Concatenate or not pointers to entity names

pass the 'top' volume to the writer

to read:

```
parser.Read( "g4test.gdml", true );
```

Activate or de-activatate schema validation

get pointer to 'top' world volume

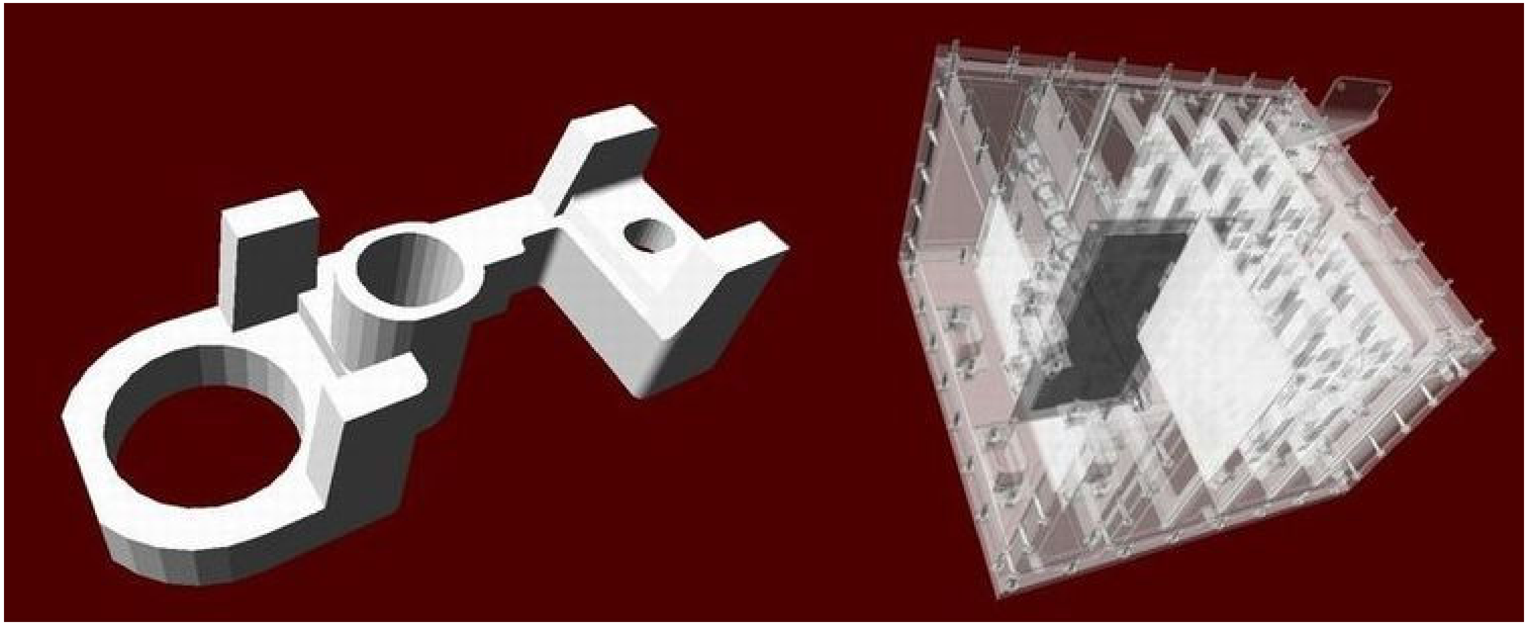
```
pWorld = GDMLProcessor::GetInstance()->GetWorldVolume();
```

Using GDML in Geant4 - 2

- Any geometry tree can be dumped to file
  - … just provide its physical volume pointer (**`pVol`**):
    ```
    parser.Write("g4test.gdml", pVol);
    ```
- A geometry setup can be split in modules
  - … starting from a geometry tree specified by a physical volume:
    ```
    parser.AddModule(pVol);
    ```
  - … indicating the depth from which starting to modularize:
    ```
    parser.AddModule(depth);
    ```
- Provides facility for importing CAD geometries generated through STEP-Tools
- Allows for easy extensions of the GDML schema and treatment of auxiliary information associated to volumes
- Full coverage of materials, solids, volumes and simple language constructs (variables, loops, etc…)

Importing CAD geometries with GDML

- CAD geometries generated through STEP-Tools (**`stFile.geom`**, **`stFile.tree`** files) can be imported through the GDML reader:
  - **`parser.ParseST("stFile", WorldMaterial, GeomMaterial);`**



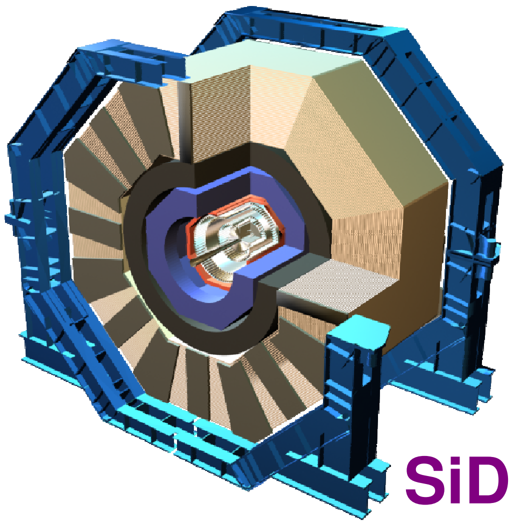
- Tools like FastRad allow for importing CAD STEP files and directly convert to GDML

GDML processing performance

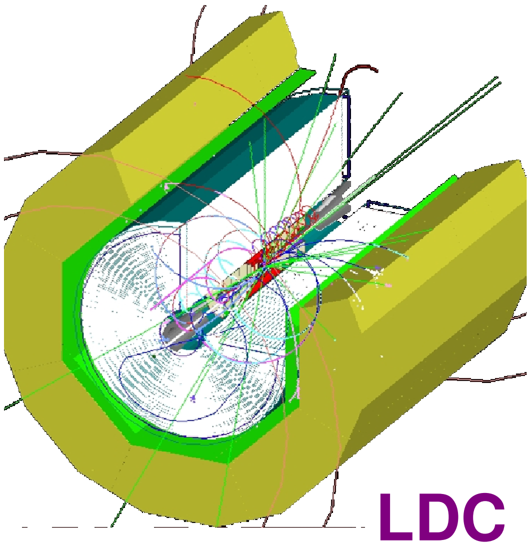
- GDML reader/writer tested on
  - complete LHCb and CMS geometries
  - parts of ATLAS geometry
    - full ATLAS geometry includes custom solids
- for LHCb geometry (~5000 logical volumes)
  - writing out ~10 seconds (on P4 2.4GHz)
  - reading in ~ 5 seconds
  - file size ~2.7 Mb (~40k lines)
- for CMS geometry (~19000 logical volumes)
  - writing out ~30 seconds
  - reading in ~15 seconds
  - file size ~7.9 Mb (~120k lines)

GDML as primary geometry source

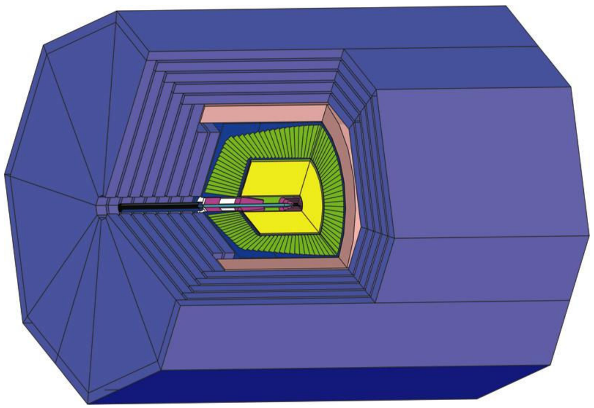
- Linear Collider
  - Linear Collider Detector Description (LCDD) extends GDML with Geant4-specific information (sensitive detectors, physics cuts, etc)
  - GDML/LCDD is generic and flexible
    - several different full detector design concepts, including SiD, GLD, and LDC, where simulated using the same application



SiD



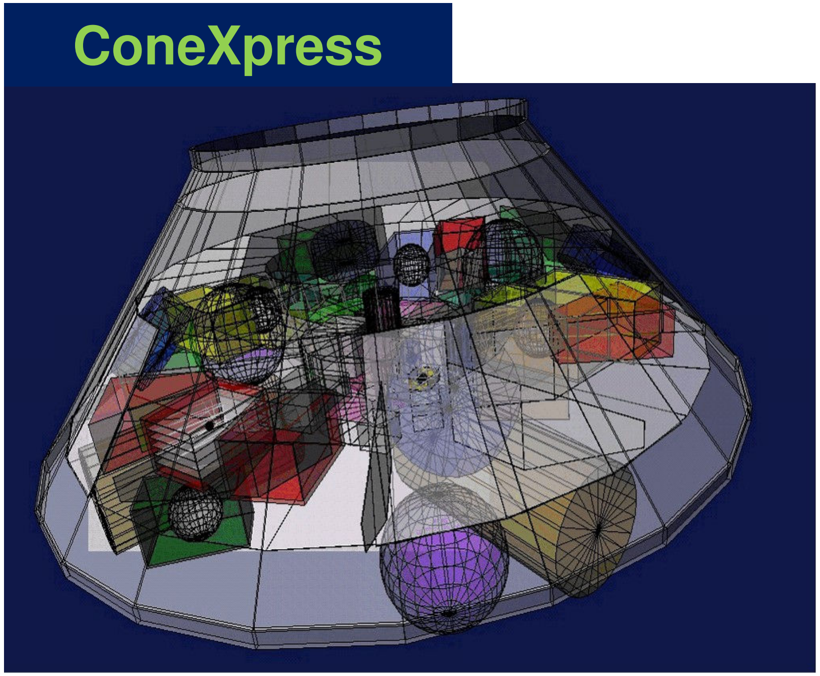
LDC



GLD

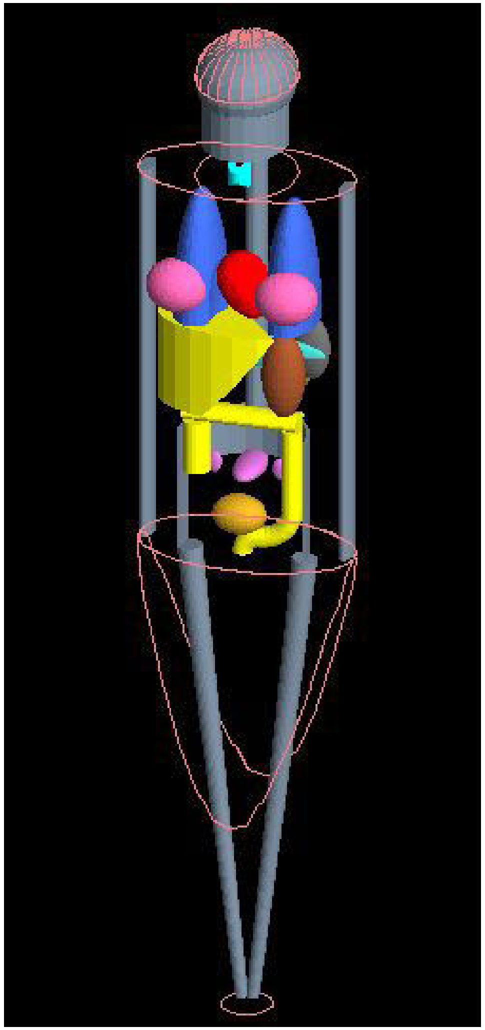
GDML as primary geometry source - 2

- Space Research @ ESA
  - Geant4 geometry models
    - component degradation studies (JWST, ConeXpress,...)
    - GRAS (Geant4 Radiation Analysis for Space)
  - enables flexible geometry configuration and changes
  - main candidate for CAD to Geant4 exchange format



GDML as primary geometry source - 3

- Anthropomorphic Phantom
  - Modeling of the human body and anatomy for radioprotection studies
  - no hard-coded geometry, flexible configuration



Exercise 1c

- *GDML*