GEANT4 BEGINNERS COURSE

LNS, Catania (Italy)
17 – 21 Nov 2014

Advanced Geometry

Geant 4 tutorial course

Advanced Geometry

Part I:

- Repeated Volumes
  - Replicas
  - Parametrised Volumes
  - Divided Volumes

Part II:

- Importing detector geometry from CAD

Replicas

A single Physical Volume represents multiple copies of a volume within their mother volume

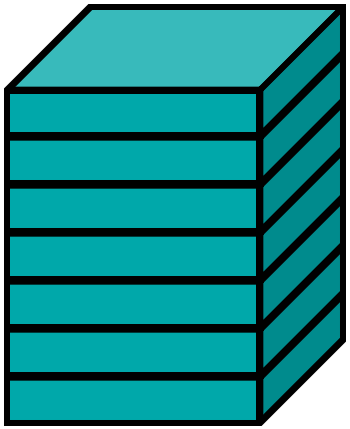
→ Save memory and machine time

a daughter logical volume to be replicated

- The mother volume is completely filled with replicas, all of which must be of the same type of the mother

Replication may occur along:

- *Cartesian axes (kXAxis, kYAxis, kZAxis)*
  - Coordinate system at the center of each replica
- *Radial axis (cylindrical polar) (kRho)* - *onion rings*
  - Coordinate system same as the mother
- *Phi axis (cylindrical polar) (Phi)* – *cheese wedges*
  - Coordinate system rotated so that the X axis bisects the angle made by each wedge



mother volume

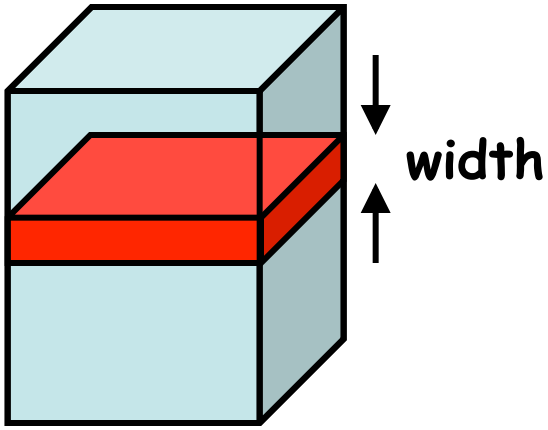
G4PVReplica

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

Features and restrictions:

- CSG solids only
- **`G4PVReplica`** MUST be the only daughter
- Replicas may be placed inside other replicas
- Normal placement volumes may be placed inside replicas
- Parameterised volumes cannot be placed inside a replica
- No volume can be placed inside a radial replication

Replicas - axis, width, offset

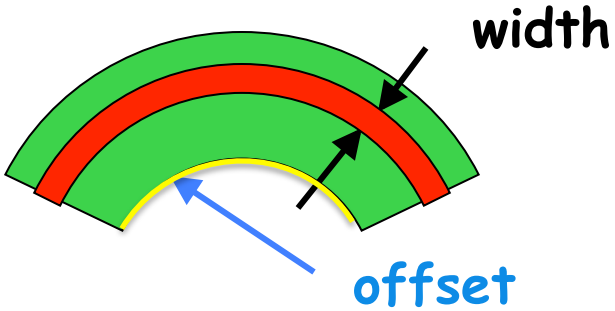


Center of nth daughter is given as

- Cartesian axes - kXaxis, kYaxis, kZaxis

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

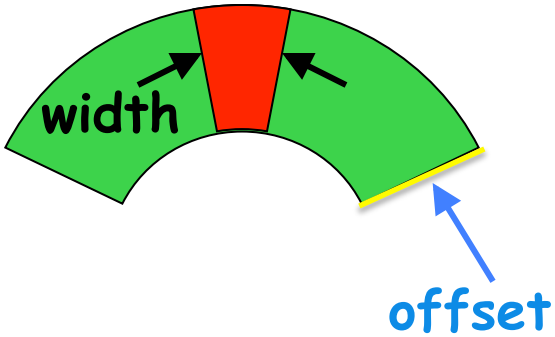
  Offset shall not be used



- Radial axis – kRho

  `width*(n+0.5)+offset`

  Offset must be the inner radius of the mother



- Phi axis – kPhi

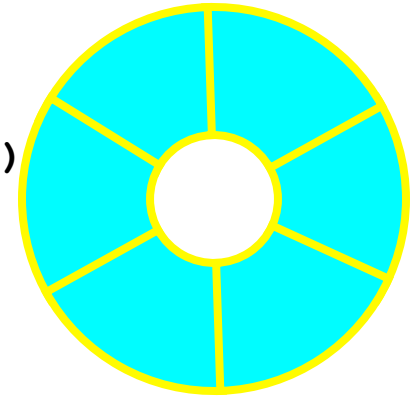
  `width*(n+0.5)+offset`

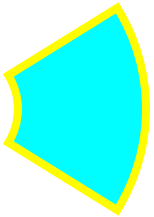
  Offset must be the starting angle of the mother

G4PVReplica : example

replication along the kPhi axis

```
G4double tube_dPhi = 2.* M_PI * rad;
G4VSolid* tube_sol =
   new G4Tubs("tubeS",20*cm,50*cm,30*cm,0.,tube_dPhi);
G4LogicalVolume * tube_log =
   new G4LogicalVolume(tube_sol, Air, "tubeL", 0, 0, 0)
G4VPhysicalVolume* tube_phys =
   new G4PVPlacement(0,G4ThreeVector(-200.*cm,0.,0.),
            "tubeP", tube_log, world_phys, false, 0);
G4double wedge_dPhi = tube_dPhi/6.;
G4VSolid* wedge =
   new G4Tubs(”wedge", 20*cm, 50*cm, 30*cm,
        -wedge_dPhi/2., wedge_dPhi);
G4LogicalVolume* wedge_log =
   new G4LogicalVolume(wedge,Pb,”wedgeL",0,0,0);
G4VPhysicalVolume* wedge_rep =
   new G4PVReplica(”wedge_rep", wedge_log,
        tube_log, kPhi, 6, wedge_dPhi);
```



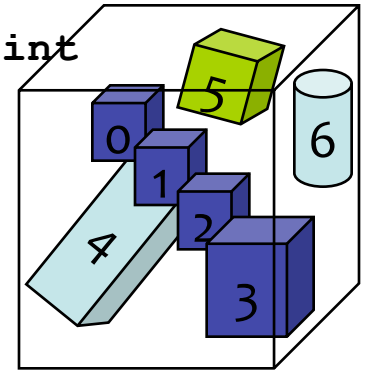


Parameterised Volumes

A single Physical Volume represents multiple volumes within their mother volume

- Repeated volumes can differ by size, solid type, material and transformation matrix, that can all be parameterised by the user as a function of the copy number
- User is asked to derive her/his own parameterisation class from **G4VPVParameterisation** that must implement the methods:

```
void ComputeTransformation(const G4int copyNo,
    G4VPhysicalVolume *physVol) const;
void ComputeDimensions(G4Tubs& trackerLayer, const G4int
    copyNo, const G4VPhysicalVolume *physVol)  const;
```



- Optional methods:

```
ComputeMaterial(…)   //see examples/extended/medical/DICOM
ComputeSolid(…)
```

- Daughters supported only for fixed type and size Parameterised Volumes

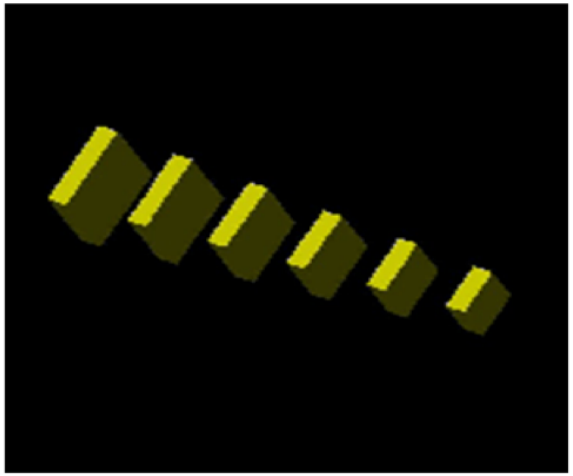
G4PVParameterised

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

- Replicates the volume **`nReplicas`** times using the parameterization **`pParam`**, within the mother volume **`pMother`**
- **`pAxis`** specifies the tracking optimisation algorithm to apply:
  - **`kXAxis, kYAxis, kzAxis`** → 1D voxelisation algorithm
  - **`kUndefined`** → 3-D voxelisation algorithm
- Each replicated volume is a touchable detector element
- CSG solids only

Parameterisation Example - 1

Imagine you have a Logical Volume (logicTracker) already defined and that you want to place inside the tracker a multi chamber volume, with chambers of different dimension



```
G4VSolid* solidChamber =
        new G4Box("chamber", 100*cm, 100*cm, 10*cm);
G4LogicalVolume* logicChamber =
        new G4LogicalVolume(solidChamber, ChamberMaterial,
                "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);
```

Parameterisation Example - 2

Example of user-defined parameterisation class: the header (class definition)

```
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

Example of user-defined parameterisation class: methods implementation

```
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);
}
```

Divided Volumes

A single Physical Volume represents multiple copies of a volume within its mother volume

"special" kind of parameterised volumes

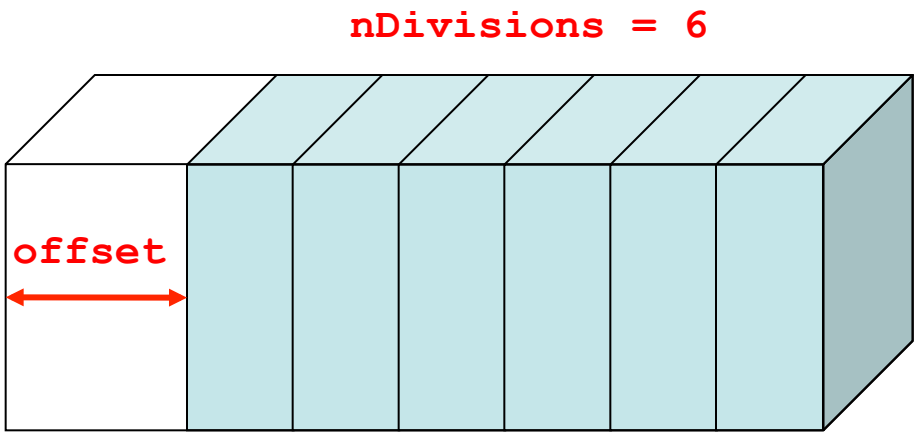
- The mother volume is divided in volumes of the same type of the mother, like for replicas, but:
  - Offset can be set
  - Completely filling of the mother is not mandatory
- CSG solids only (box, tubs, cons, para, trd, polycone, polyhedra)
- Axes of division vary according to solid type
- Each division is a touchable detector element
- **`G4PVDivision`** class automatically defines the volume division using values provided in input. It returns a G4VPhysicalVolume. **Three different constructors**

G4PVDivision - 1

```
G4PVDivision(const G4String &pName,
             G4LogicalVolume* pLogical,
             G4LogicalVolume* pMother,
             const EAxis pAxis,
             const G4int nDivisions,
             const G4double offset);
```

Offset and number of divisions are given; the size (width) of the daughter volume is calculated as

```
( (size of mother) - offset ) / nDivisions
```

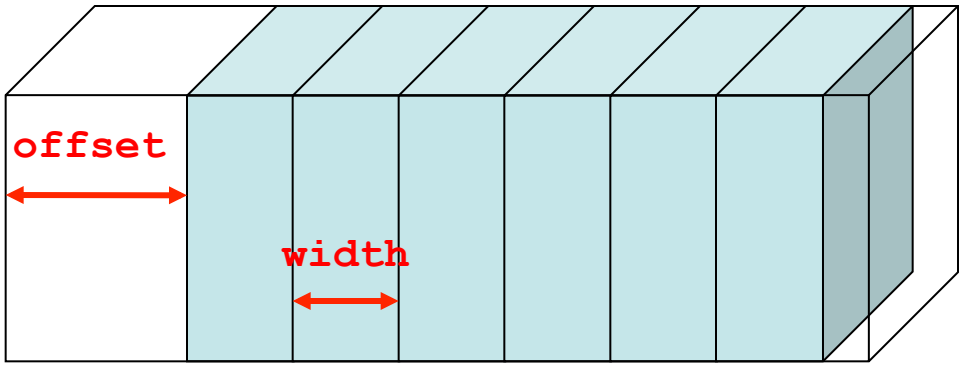


G4PVDivision - 2

```
G4PVDivision(const G4String &pName,
             G4LogicalVolume* pLogical,
             G4LogicalVolume* pMother,
             const EAxis pAxis,
             const G4double width,
             const G4double offset);
```

Width and Offset are given; the number of daughter volumes is calculated as `int( ( (size of mother) - offset ) / width )`

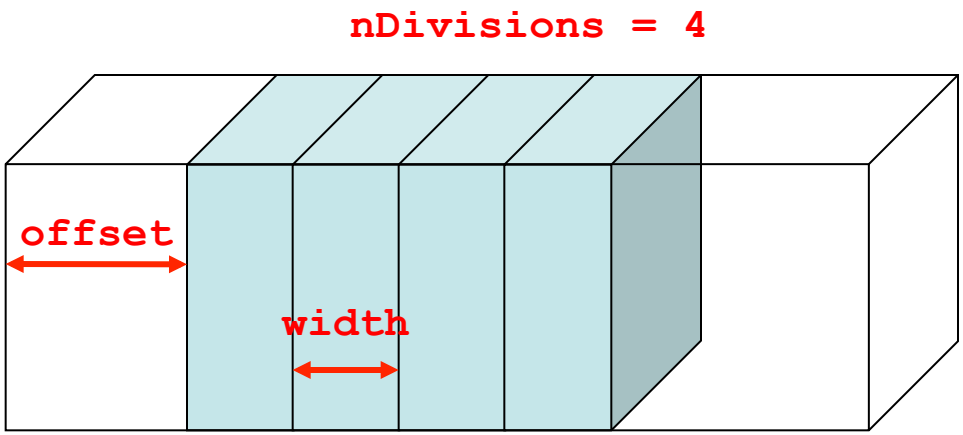
Allowed divisions = 6



G4PVDivision - 3

```
G4PVDivision(const G4String &pName,
             G4LogicalVolume* pLogical,
             G4LogicalVolume* pMother,
             const EAxis pAxis,
             const G4int nDivisions,
             const G4double width,
             const G4double offset);
```

`nDivisions` daughters of **`width`** thickness



Importing detector geometry from CAD

- Importing CAD design of your detector into Geant4 is an attractive solution:
  - Detailed geometry description already available
  - Geometry check "for free"
    - CAD systems provide the most powerful clash detection algorithms, treating the intersection between the volumes in their topological form
- Unfortunately, importing procedure is not immediate:
  - No common standard for geometry description (between G4 and CAD)
  - CAD systems exchange geometrical data via **STEP** ISO standard
  - Geant4 solid modeler is STEP compliant, but Geant4 can't read this standard for the moment
  - Geant4 can read in **GDML** → can this be useful to our purposes?

GDML

Geometry Description Markup Language

application-indepedent geometry description format based on XML

The language is based on a syntactic mechanism (TAGS) that allows to define and control the meaning of the elements contained in the GDML document (text file).

Profitably used for geometry exchange between:

- Fluka and Geant4 → physics validation
- Geant4 and ROOT → geometry visualization
- CAD and Geant4 → geometry import
  - avoids hard-coding the geometry
  - allows running the same application with different geometries

GDML document example

positions, rotations

materials

solids

geometry tree

'world' volume

```
<?xml version="1.0" encoding="UTF-8"?>
<gdml xsi:noNamespaceSchemaLocation="GDMLSchema/gdml.xsd">
  <define>
        …
        <position name="TrackerinWorldpos" unit="mm" x="0" y="0" z="100"/>
  </define>
  <materials>
        …
        <material formula=" " name="Air" >
                  <D value="1.290" unit="mg/cm3"/>
                  <fraction n="0.7" ref="Nitrogen" />
                  <fraction n="0.3" ref="Oxygen" />
        </material>
  </materials>
  <solids>
        …
        <box lunit="mm" name="Tracker" x="50" y="50" z="50"/>
  </solids>
  <structure>
        …
        <volume name="World" >
                  <materialref ref="Air" />
                  <solidref ref="world" />
                  <physvol>
                            <volumeref ref="Tracker" />
                            <positionref ref="TrackerinWorldpos"/>
                            <rotationref ref="TrackerinWorldrot"/>
                  </physvol>
        </volume>
  </structure>
  <setup name="Default" version="1.0" >
        <world ref="World" />
  </setup>
</gdml>
```

- Similar to HTML, explicit tags for elements, …
- ASCII file: easy to create, read, debug, modify,…

GDML components

GDML Schema

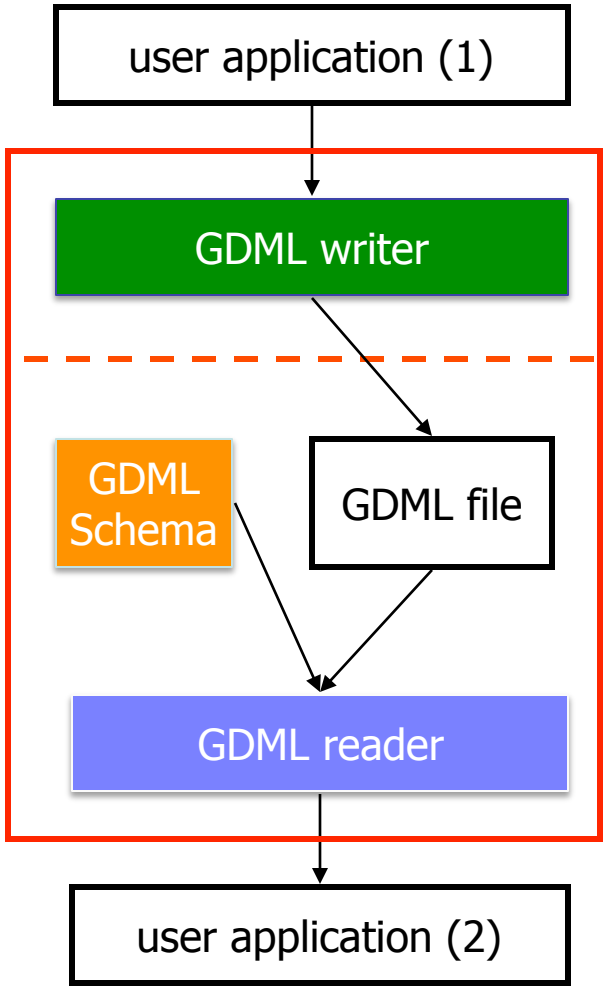
- self-consistent definition of GDML syntax
- defines document structure and the list of legal elements

GDML Reader

- Interprets the GDML geometry description using the GDML Schema
- Creates 'in-memory' representation of it

GDML Writer

- Allows exporting geometry (in GDML) on a file



Reader and Writer are integrated in packages like Geant4 and ROOT providing GDML compliant interfaces

Reading / Writing GDML files in Geant4

In your DetectorConstructor::Construct()

```
// gdml parser
#include "G4GDMLParser.hh"
G4GDMLParser fParser;

// importing geometry
    fParser.Read("detectorgeometry.gdml");     // reads and
                                               // store in
                                               // memory
    G4VPhysicalVolume* fWorldPhysVol;
    fWorldPhysVol = fParser.GetWorldVolume(); // get world

// exporting geometry
fParser.Write("geometrydump.gdml", fWorldPhysVol);
```

This is the simplest example. Options are available to require geometry check during the import, to choose the hierarchy depth during the export or to specify the actual path where your GDML schema stays.

GDML Parser can be built and installed in G4 as an optional choice

Tessellated solids

The geometry imported from GDML will be made by tessellated solids

`G4TessellatedSolid`

Generic solid defined by a number of facets (**`G4VFacet`**)

- Facets can be triangular or quadrangular

Geant4 CAD interface: two applications

ESABASE2 (eta_max)

FASTRAD (TRAD)

Free for academic non commercial use (Unix and Windows)

Free demo (30 days, 20 volumes max, Windows only)

- 3D Modelling & Visualisation tools
- Radiation calculation & analysis

Geometry

- Basic shapes
- Boolean solids
- Tesselated shapes

Materials

- Large database
- Management tools

Import

- STEP, IGES formats from the main CAD software (AutoCAD, CATIA, …)

Export

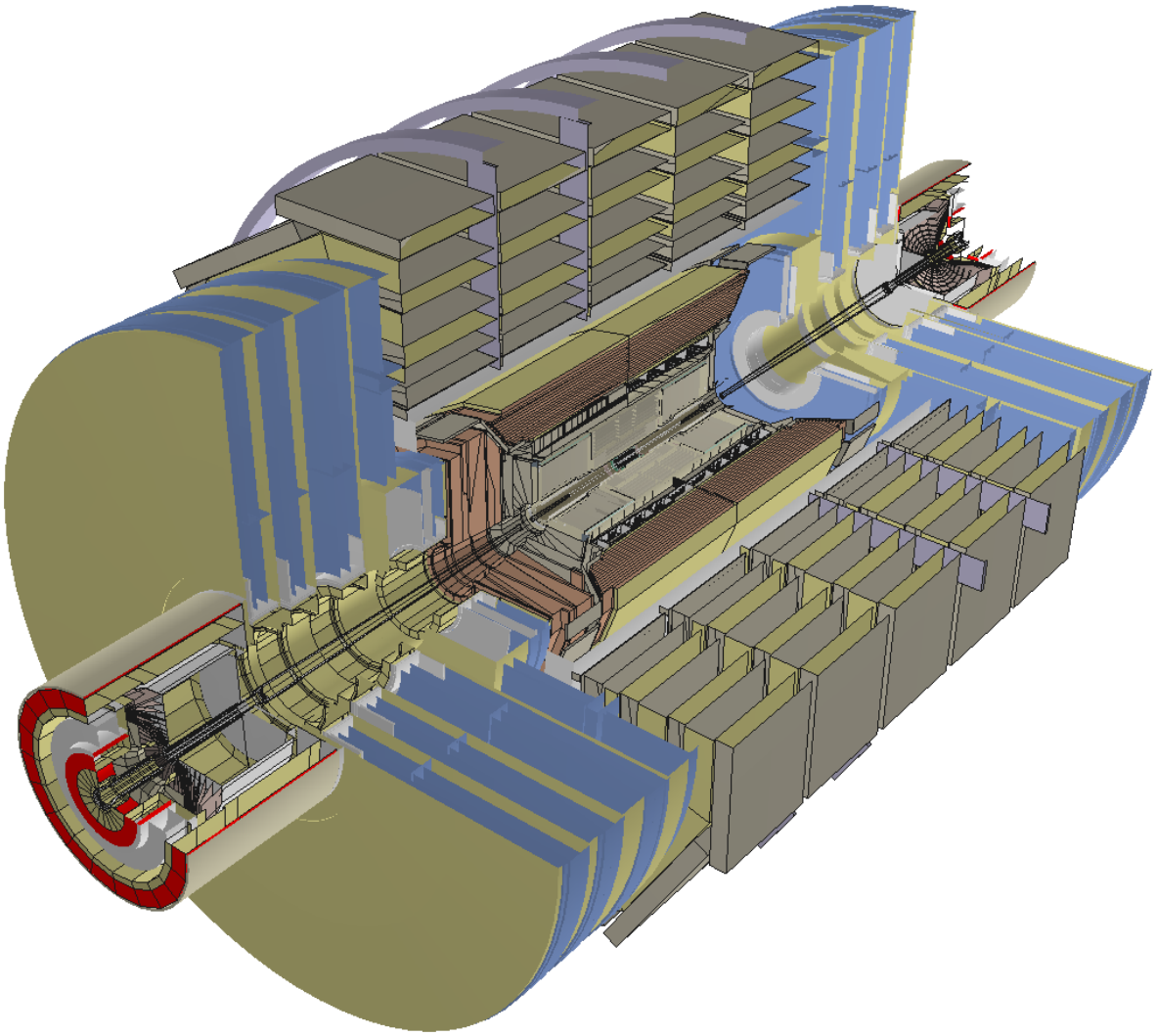
- GEANT4 (*C++ Project* or GDML)
- STEP, IGES formats
- VRML

Other software + STViewer

Other tools which can be used to generate meshes to be then imported in Geant4 as tessellated solids are:

- STViewer – import STEP, export (.geom + .tree) proprietary format, also readable by Geant4, but much less portable
- STL2GDML - free STL to GDML converter
- SALOME - Open-source software. Imports STEP/BREP/IGES/ACIS, exports to STL
- CADMesh – Based on the VCG Library. Reads STL files and import in Geant4
- Cogenda - Commercial TCAD software for generation of 3D meshes and export to GDML

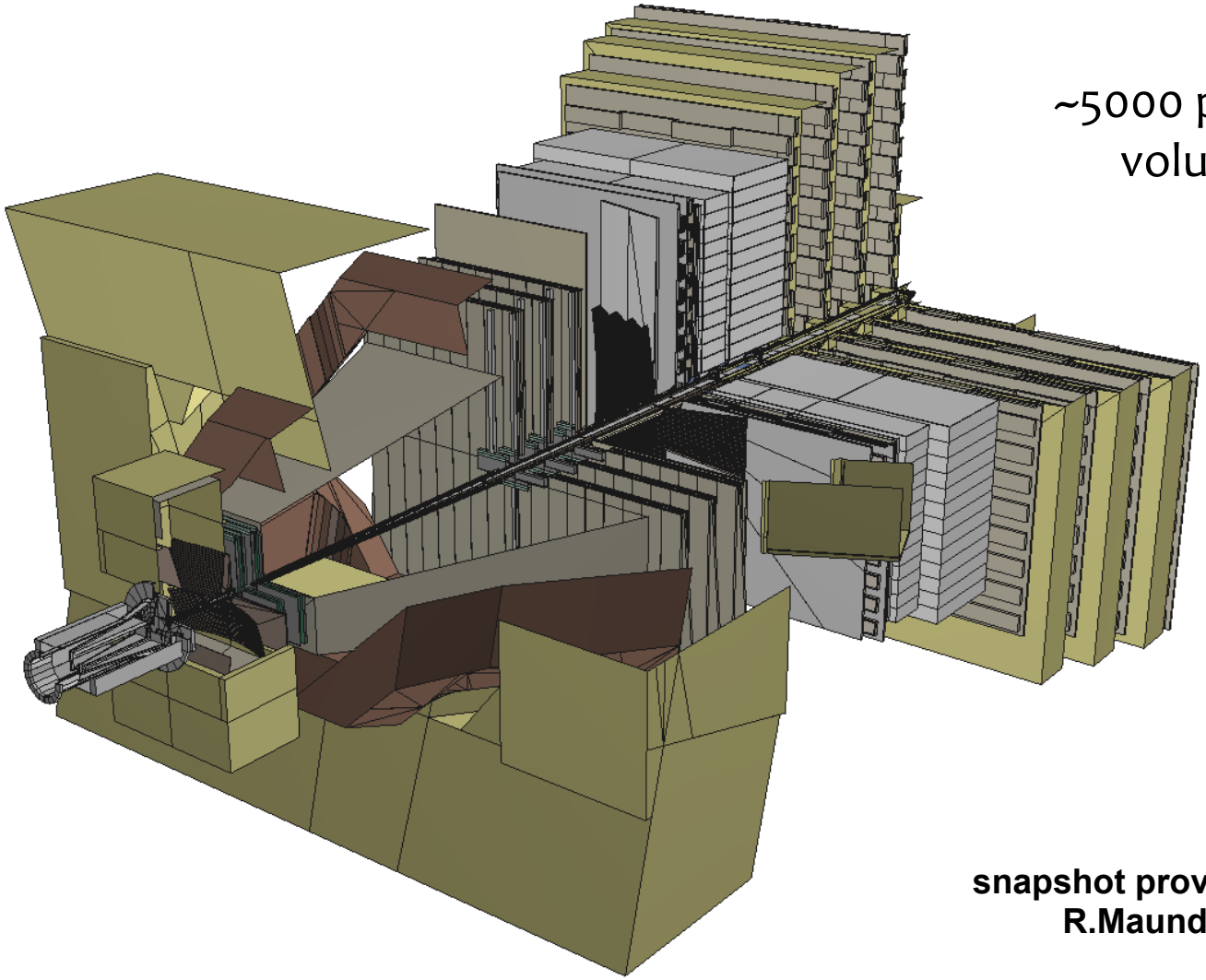
CMS detector: G4→GDML→ROOT



~19000 physical volumes

snapshot provided by R.Maunder

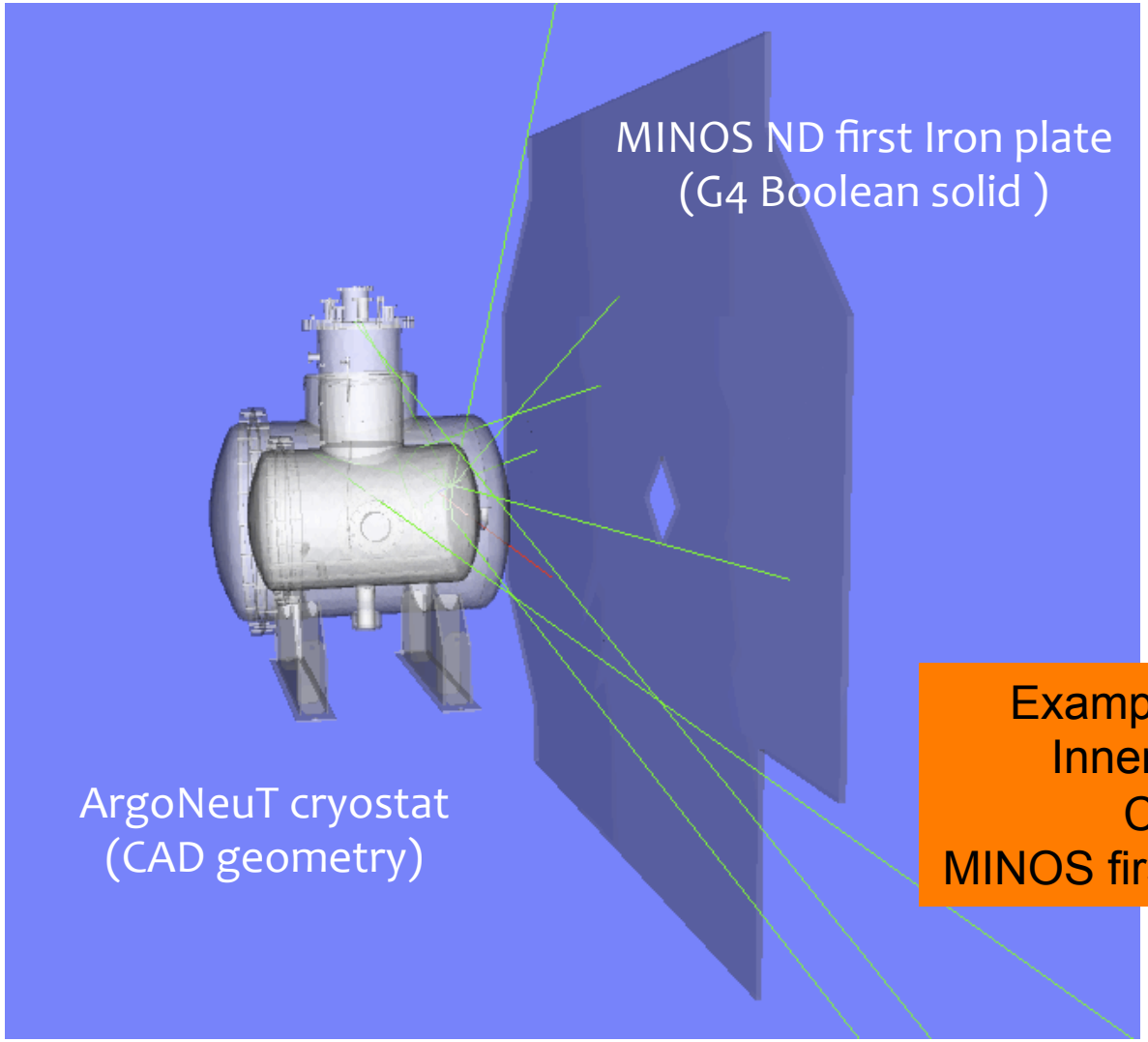
LHCb Detector: G4→GDML→ROOT



~5000 physical volumes

snapshot provided by R.Maunder

ArgoNeuT detector
CAD project → STEP → FASTRAD → GDML→ G4



Example of integrated geometry:
Inner detector → CSG solids
Cryostat → from CAD
MINOS first plate → Boolean G4 Solids

Thanks for your attention

Volume Stores

- All volumes must be allocated using `new` in the user's application
- At construction, they get registered to a proper Volume Store
- At the end of the job, they will also be automatically deallocated

- Stores:
  - G4SolidStore
  - G4LogicalVolumeStore
  - G4PhysicalVolumeStore
  - G4RegionStore

- Volumes can be retrieved at any convenience, for instance:
  - by index

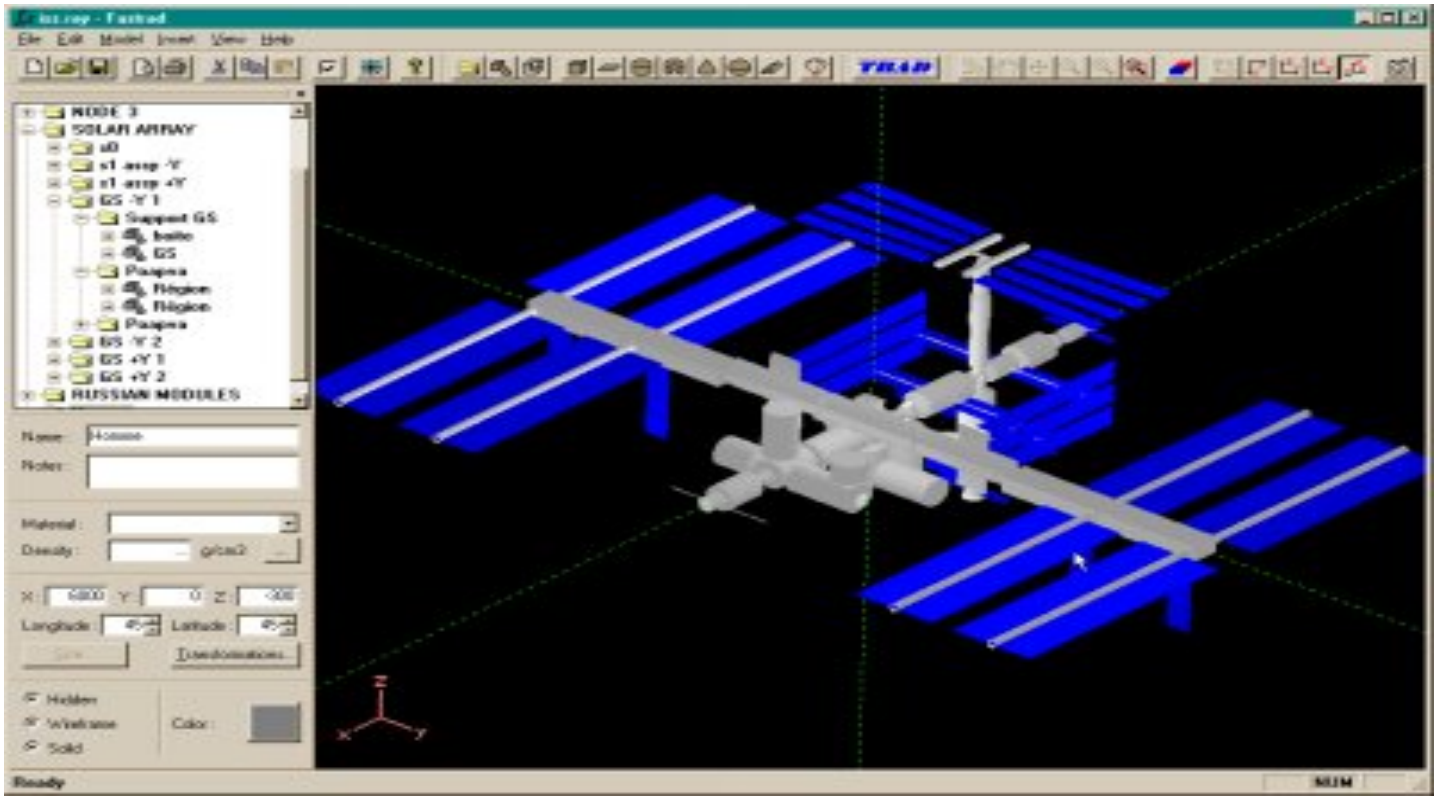
```
int NofV=G4LogicalVolumeStore::GetInstance()->size();
for(int i=0;i<NofV;i++){
        lVol=(*(G4LogicalVolumeStore::GetInstance()))[i];
}
```

  - by name

```
G4LogicalVolume* logicVol =
G4LogicalVolumeStore::GetInstance()->GetVolume("logicLAr");
```

FASTRAD3

- Commercial Software “The 3D CAD Tool For Radiation Shielding Analysis” (for Windows only)



- Free Demo on www.fastrad.net (30 days license, import 20 volumes at max)
- Very useful crossed-support (FASTRAD + Geant4) on the Geant4 user forum