Detector Simulation
Geometry

Witek Pokorski
Alberto Ribon
CERN

13-14.02.2017
Chalenge

- how to implement (efficiently) this in your computer program?
  - you need ‘bricks’
    - ‘solids’, ‘shapes’
    - you need to position them
    - you want to ‘reuse’ as much as possible the same ‘templates’
Building blocks

- set of **solids** (shapes) classes
  - box, sphere, tube, etc, etc...
  - boolean operations on solids
- **logical volumes**
  - unpositioned volumes with associated materials and possibly with ‘daughter’ volumes
    - unpositioned hierarchies of volumes
- **physical volumes**
  - concrete ‘placements’ of logical volumes
  - can reuse the same logical volume several times
Volumes

- Logical volume
  - Sensitive detector
  - Solid
  - Material
    - Daughter physical volumes

- Physical volume
  - Mother volume
    - Translation
    - Rotation
  - Logical volume
Hierarchy

world (PV)

experiment cavern (LV)

tracker 1 (PV)  tracker 2 (PV)  calorimeter (PV)

calorimeter (LV)

cell1 (PV)  cell2 (PV)  cell LV

module LV

module 1(PV)  module 1(PV)  module 1(PV)  module 1(PV)
Units

• In Geant4 quantities should be explicitly multiplied by units
  – for example:
    G4double width = 12.5*m;
    G4double density = 2.7*g/cm3;
  – If no unit is specified, the *internal* G4 unit will be used, but **this is discouraged**!
  – Almost all commonly used units are available.
  – The user can define new units.
  – Refer to CLHEP: SystemOfUnits.h

• Divide a variable by a unit you want to get.
  G4cout << dE / MeV << “ (MeV)” << G4endl;
Describe your detector

• Derive your own concrete class from G4VUserDetectorConstruction abstract base class.

• Implementing the method Construct():
  – Modularize it according to each detector component or sub-detector:
    1. Construct all necessary materials
    2. Define shapes/solids required to describe the geometry
    3. Construct and place volumes of your detector geometry

  ➢ Define sensitive detectors and identify detector volumes which to associate them
  ➢ Associate magnetic field to detector regions
  ➢ Define visualization attributes for the detector elements
Definition of Materials

• Different kinds of materials can be defined:
  – isotopes <> G4Isotope
  – elements <> G4Element
  – molecules <> G4Material
  – compounds and mixtures <> G4Material

• Attributes associated:
  – temperature, pressure, state, density
Isotopes, Elements and Materials

- **G4Isotope** and **G4Element** describe the properties of the atoms:
  - Atomic number, number of nucleons, mass of a mole, shell energies
  - Cross-sections per atoms, etc...

- **G4Material** describes the *macroscopic* properties of the matter:
  - temperature, pressure, state, density
  - Radiation length, absorption length, etc...

- **G4Material** is the class used for geometry definition
Elements & Isotopes

• Isotopes can be assembled into elements

\[
\text{G4Isotope (const G4String& name,} \\
\text{\hspace{1cm} G4int \hspace{0.5cm} z, \hspace{0.5cm} // atomic number} \\
\text{\hspace{1cm} G4int \hspace{0.5cm} n, \hspace{0.5cm} // number of nucleons} \\
\text{\hspace{1cm} G4double \hspace{0.5cm} a \}); \hspace{0.5cm} // mass of mole}
\]

• ... building elements as follows:

\[
\text{G4Element (const G4String& name,} \\
\text{\hspace{1cm} const G4String& symbol, \hspace{0.5cm} // element symbol} \\
\text{\hspace{1cm} G4int \hspace{0.5cm} nIso \}); \hspace{0.5cm} // \# \ of \ isotopes}
\]
\[
\text{G4Element::AddIsotope(G4Isotope* iso, \hspace{0.5cm} // isotope} \\
\text{\hspace{1cm} G4double relAbund);} \hspace{0.5cm} // \ fraction \ of \ atoms} \\
\text{\hspace{1cm} // per \ volume}
\]
Material of one element

• Single element material

G4double density = 1.390*g/cm3;
G4double a = 39.95*g/mole;
G4Material* lAr =
    new G4Material("liquidArgon",z=18.,a,density);
• A Molecule is made of several elements (composition by number of atoms):

```plaintext
a = 1.01*g/mole;
G4Element* elH  =
    new G4Element("Hydrogen",symbol="H",z=1.,a);
a = 16.00*g/mole;
G4Element* elO  =
    new G4Element("Oxygen",symbol="O",z=8.,a);
density = 1.000*g/cm3;
G4Material* H2O =
    new G4Material("Water",density,ncomp=2);
H2O->AddElement(elH, natoms=2);
H2O->AddElement(elO, natoms=1);
```
Material: compound

- Compound: composition by fraction of mass

\[
a = 14.01\text{g/mole};
\]

\[
\text{G4Element}\ast\ \text{elN} = \\
\quad\text{new G4Element(name=\"Nitrogen\",symbol=\"N\",z= 7.,a)};
\]

\[
a = 16.00\text{g/mole};
\]

\[
\text{G4Element}\ast\ \text{elO} = \\
\quad\text{new G4Element(name=\"Oxygen\",symbol=\"O\",z= 8.,a)};
\]

\[
density = 1.290\text{mg/cm3};
\]

\[
\text{G4Material}\ast\ \text{Air} = \\
\quad\text{new G4Material(name=\"Air\",density,ncomponents=2)};
\]

\[
\text{Air}\rightarrow\text{AddElement(elN, 70.0*perCent)};
\]

\[
\text{Air}\rightarrow\text{AddElement(elO, 30.0*perCent)};
\]
Material: mixture

• Composition of compound materials

G4Element* elC = ...;  // define “carbon” element
G4Material* SiO2 = ...;  // define “quartz” material
G4Material* H2O = ...;  // define “water” material

density = 0.200*g/cm3;
G4Material* Aerog =
    new G4Material("Aerogel",density,ncomponents=3);
Aerog->AddMaterial(SiO2,fractionmass=62.5*perCent);
Aerog->AddMaterial(H2O,fractionmass=37.4*perCent);
Aerog->AddElement(elC,fractionmass=0.1*perCent);
Example: gas

• It may be necessary to specify temperature and pressure – \(\frac{dE}{dx}\) computation affected

```cpp
G4double density = 27.*mg/cm3;
G4double temperature = 325.*kelvin;
G4double pressure = 50.*atmosphere;

G4Material* CO2 =
    new G4Material("CarbonicGas", density, ncomponents=2
                    kStateGas, temperature, pressure);
CO2->AddElement(C,natoms = 1);
CO2->AddElement(O,natoms = 2);
```
Example: vacuum

• Absolute vacuum does not exist. It is a gas at very low density!
  – Cannot define materials composed of multiple elements through $Z$ or $A$, or with $\rho=0$

```cpp
G4double atomicNumber = 1.;
G4double massOfMole = 1.008*g/mole;
G4double density = 1.e-25*g/cm3;
G4double temperature = 2.73*kelvin;
G4double pressure = 3.e-18*pascal;
G4Material* Vacuum =
    new G4Material("interGalactic", atomicNumber, massOfMole, density, kStateGas, temperature, pressure);
```
NIST Manager & Messenger

• NIST database for materials is imported inside Geant4
  http://physics.nist.gov/PhysRefData

• Additional interfaces defined

• UI commands specific for handling materials

• The best accuracy for the most relevant parameters guaranteed:
  • Density
  • Mean excitation potential
  • Chemical bounds
  • Element composition
  • Isotope composition
  • Various corrections
# NIST Elements & Isotopes

<table>
<thead>
<tr>
<th>$Z$</th>
<th>$A$</th>
<th>$m$</th>
<th>error</th>
<th>$A_{\text{eff}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>(%)</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Si</td>
<td>22</td>
<td>22.03453 (22)</td>
<td>28.0855(3)</td>
</tr>
<tr>
<td>23</td>
<td></td>
<td>23.02552 (21)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td>24.011546 (21)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td>25.004107 (11)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td>25.992330 (3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td>26.98760476 (17)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
<td>27.9769265327 (20)</td>
<td>92.2297 (7)</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
<td>28.97649472 (3)</td>
<td>4.6832 (5)</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td>29.97377022 (5)</td>
<td>3.0872 (5)</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td></td>
<td>30.97536327 (7)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td></td>
<td>31.9741481 (23)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td></td>
<td>32.978001 (17)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>34</td>
<td></td>
<td>33.978576 (15)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td></td>
<td>34.984580 (40)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td></td>
<td>35.986669 (11)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>37</td>
<td></td>
<td>36.99300 (13)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>38</td>
<td></td>
<td>37.99598 (29)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>39</td>
<td></td>
<td>39.00230 (43)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td></td>
<td>40.00580 (54)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>41</td>
<td></td>
<td>41.01270 (64)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>42</td>
<td></td>
<td>42.01610 (75)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Natural isotope compositions
- More than 3000 isotope masses
- Used for elements definition
### NIST Elementary Materials from the NIST Data Base

<table>
<thead>
<tr>
<th>Z</th>
<th>Name</th>
<th>Chemical Formula</th>
<th>Density (g/cm³)</th>
<th>I (eV)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>G4_H</td>
<td>H_2</td>
<td>8.3748e-05</td>
<td>19.2</td>
</tr>
<tr>
<td>2</td>
<td>G4_He</td>
<td></td>
<td>0.000166322</td>
<td>41.8</td>
</tr>
<tr>
<td>3</td>
<td>G4_Li</td>
<td></td>
<td>0.534</td>
<td>40</td>
</tr>
<tr>
<td>4</td>
<td>G4_Be</td>
<td></td>
<td>1.848</td>
<td>63.7</td>
</tr>
<tr>
<td>5</td>
<td>G4_B</td>
<td></td>
<td>2.37</td>
<td>76</td>
</tr>
<tr>
<td>6</td>
<td>G4_C</td>
<td></td>
<td>2</td>
<td>81</td>
</tr>
<tr>
<td>7</td>
<td>G4_N</td>
<td>N_2</td>
<td>0.0011652</td>
<td>82</td>
</tr>
<tr>
<td>8</td>
<td>G4_O</td>
<td>O_2</td>
<td>0.00133151</td>
<td>95</td>
</tr>
<tr>
<td>9</td>
<td>G4_F</td>
<td></td>
<td>0.00158029</td>
<td>115</td>
</tr>
<tr>
<td>10</td>
<td>G4_Ne</td>
<td></td>
<td>0.000838505</td>
<td>137</td>
</tr>
<tr>
<td>11</td>
<td>G4_Na</td>
<td></td>
<td>0.971</td>
<td>149</td>
</tr>
<tr>
<td>12</td>
<td>G4_Mg</td>
<td></td>
<td>1.74</td>
<td>156</td>
</tr>
<tr>
<td>13</td>
<td>G4_Al</td>
<td></td>
<td>2.6989</td>
<td>166</td>
</tr>
<tr>
<td>14</td>
<td>G4_Si</td>
<td></td>
<td>2.33</td>
<td>173</td>
</tr>
</tbody>
</table>

### NIST Compound Materials from the NIST Data Base

<table>
<thead>
<tr>
<th>N</th>
<th>Name</th>
<th>Density (g/cm³)</th>
<th>I (eV)</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>G4_Adipose_Tissue</td>
<td>0.92</td>
<td>63.2</td>
</tr>
<tr>
<td>4</td>
<td>G4_Air</td>
<td>0.00120479</td>
<td>85.7</td>
</tr>
<tr>
<td>2</td>
<td>G4_CsI</td>
<td>4.51</td>
<td>553.1</td>
</tr>
</tbody>
</table>

- **NIST Elementary materials:**
  - H -> Cf (Z = 1 -> 98)
- **NIST compounds:**
  - e.g. “G4_ADIPOSE_TISSUE_IRCP”
- **HEP and Nuclear materials:**
  - e.g. Liquid Ar, PbWO
- It is possible to build mixtures of NIST and user-defined materials
How to use the NIST DB

• No need to predefine elements and materials
• Retrieve materials from NIST manager:

```cpp
G4NistManager* manager = G4NistManager::Instance();

G4Material* H2O = manager->FindOrBuildMaterial("G4_WATER");

G4Material* mat = manager->ConstructNewMaterial("name", const std::vector<G4String>& elements, const std::vector<G4double>& weights, G4double density, G4bool isotopes);
G4double isotopeMass = manager->GetMass(G4int Z, G4int N);
```

• Some UI commands ...

```
/material/nist/printElement ➔ print defined elements
/material/nist/listMaterials ➔ print defined materials
```
Creating a Detector Volume

• Start with its Shape & Size
  – Box 3x5x7 cm, sphere R=8m

• Add properties:
  – material, B/E field,
  – make it sensitive

• Place it in another volume
  – in one place
  – repeatedly using a function

➢ Solid
➢ Logical-Volume
➢ Physical-Volume
Solids

- Solids defined in Geant4:
  - CSG (Constructed Solid Geometry) solids
    - G4Box, G4Tubs, G4Cons, G4Trd, ...
    - Analogous to simple GEANT3 CSG solids
  - Specific solids (CSG like)
    - G4Polycone, G4Polyhedra, G4Hype, ...
    - G4TwistedTubs, G4TwistedTrap, ...
  - Boolean solids
    - G4UnionSolid, G4SubtractionSolid, ...

22
CSG: G4Tubs, G4Cons

```cpp
G4Tubs(const G4String& pname, // name
        G4double pRmin, // inner radius
        G4double pRmax, // outer radius
        G4double pDz,   // Z half length
        G4double pSphi, // starting Phi
        G4double pDphi); // segment angle

G4Cons(const G4String& pname, // name
        G4double pRmin1, // inner radius -pDz
        G4double pRmax1, // outer radius -pDz
        G4double pRmin2, // inner radius +pDz
        G4double pRmax2, // outer radius +pDz
        G4double pDz,    // Z half length
        G4double pSphi,  // starting Phi
        G4double pDphi); // segment angle
```
Specific CSG Solids: G4Polycone

G4Polycone(const G4String& pName,
            G4double phiStart,
            G4double phiTotal,
            G4int numRZ,
            const G4double r[],
            const G4double z[]);

• numRZ - numbers of corners in the r, z space
• r, z - coordinates of corners

• Also available additional constructor using planes
Boolean Solids

- Solids can be combined using boolean operations:
  - `G4UnionSolid`, `G4SubtractionSolid`, `G4IntersectionSolid`

- Requires: 2 solids, 1 boolean operation, and an (optional) transformation for the 2\textsuperscript{nd} solid
  - 2\textsuperscript{nd} solid is positioned relative to the coordinate system of the 1\textsuperscript{st} solid
  - Component solids must not be disjoint and must well intersect

```c
G4Box box("Box", 20, 30, 40);
G4Tubs cylinder("Cylinder", 0, 50, 50, 0, 2*M_PI);  // r:     0 -> 50
                                                        // z:   -50 -> 50
                                                        // phi:   0 ->  2 pi

G4UnionSolid union("Box+Cylinder", &box, &cylinder);
G4IntersectionSolid intersect("Box Intersect Cylinder", &box, &cylinder);
G4SubtractionSolid subtract("Box-Cylinder", &box, &cylinder);
```

- Solids can be either CSG or other Boolean solids

- **Note:** tracking cost for the navigation in a complex Boolean solid is proportional to the number of constituent solids
G4LogicalVolume

G4LogicalVolume(G4VSolid* pSolid, G4Material* pMaterial, const G4String& name, G4FieldManager* pFieldMgr=0, G4VSensitiveDetector* pSDetector=0, G4UserLimits* pULimits=0, G4bool optimise=true);

– Contains all information of volume except position:
  • Shape and dimension (G4VSolid)
  • Material, sensitivity, visualization attributes
  • Position of daughter volumes
  • Magnetic field, User limits
  • Shower parameterisation

– Physical volumes of same type can share a logical volume.
– The pointers to solid and material must be NOT null
– Once created it is automatically entered in the LV store
– It is not meant to act as a base class
Geometrical hierarchy

- Mother and daughter volumes
  - A volume is placed in its mother volume
    - Position and rotation of the daughter volume is described with respect to the local coordinate system of the mother volume
    - The origin of the mother's local coordinate system is at the center of the mother volume
    - Daughter volumes cannot protrude from the mother volume
    - Daughter volumes cannot overlap
  - One or more volumes can be placed in a mother volume
Geometrical hierarchy

• Note that the mother-daughter relationship is an information of **G4LogicalVolume**
  – If the mother volume is placed more than once, all daughters by definition appear in each placed physical volume

• The **world volume** must be a unique physical volume which fully contains with some margin all the other volumes
  – The world volume defines the **global coordinate system**. The origin of the global coordinate system is at the center of the world volume
  – Position of a track is given with respect to the global coordinate system
G4PVPlacement

G4PVPlacement(G4RotationMatrix* pRot, // rotation of mother frame
          const G4ThreeVector& tlate, // position in rotated frame
          G4LogicalVolume* pCurrentLogical,
          const G4String& pName,
          G4LogicalVolume* pMotherLogical,
          G4bool pMany, // not used. Set it to false...
          G4int pCopyNo, // unique arbitrary index
          G4bool pSurfChk=false); // optional overlap check

• Single volume positioned relatively to the mother volume
  – In a frame rotated and translated relative to the coordinate system of the mother volume

• Three additional constructors:
  – A simple variation: specifying the mother volume as a pointer to its physical volume instead of its logical volume.
  – Using G4Transform3D to represent the direct rotation and translation of the solid instead of the frame (alternative constructor)
  – The combination of the two variants above
G4PVPlacement

Rotation of mother frame ...

G4PVPlacement(G4RotationMatrix* pRot,       // rotation of mother frame
            const G4ThreeVector& tlate, // position in mother frame
            G4LogicalVolume* pCurrentLogical,
            const G4String& pName,
            G4LogicalVolume* pMotherLogical,
            G4bool pMany,       // not used. Set it to false...
            G4int pCopyNo,      // unique arbitrary index
            G4bool pSurfChk=false ); // optional overlap check

• Single volume positioned relatively to the mother volume
G4PVPlacement
Rotation in mother frame...

G4PVPlacement( G4Transform3D( G4RotationMatrix &pRot,       // rotation of daughter frame
               const G4ThreeVector &tlate), // position in mother frame
               G4LogicalVolume *pDaughterLogical,
               const G4String &pName,
               G4LogicalVolume *pMotherLogical,
               G4bool pMany,             // not used, set it to false...
               G4int pCopyNo,            // unique arbitrary integer
               G4bool pSurfChk=false ); // optional overlap check

Mother volume
rotation
translation in mother frame
Detector geometry components

• Basic strategy

```cpp
G4Material* pBoxMaterial = new G4Material(...);
G4VSolid* pBoxSolid =
    new G4Box(“aBoxSolid”, 1.*m, 2.*m, 3.*m);
G4LogicalVolume* pBoxLog =
    new G4LogicalVolume( pBoxSolid, pBoxMaterial, “aBoxLog”, 0, 0, 0);
G4VPhysicalVolume* aBoxPhys =
    new G4PVPlacement( pRotation, G4ThreeVector(posX, posY, posZ), pBoxLog, “aBoxPhys”, pMotherLog, 0, copyNo);
```

• A unique physical volume which represents the experimental area must exist and fully contains all other components

➢ The world volume

---

Step 1
Create the geom. object : box

Step 2
Assign properties to object : material

Step 3
Place it in the coordinate system of mother volume
Conclusion

• we know how to describe our detector geometry
  • we create materials
  • instantiate solids
• build the volumes hierarchy
Backup
System of Units

- System of units are defined in CLHEP, based on:
  - millimetre (mm), nanosecond (ns), Mega eV (MeV), positron charge (eplus) degree Kelvin (kelvin), the amount of substance (mole), luminous intensity (candela), radian (radian), steradian (steradian)
- All other units are computed from the basic ones
- In output, Geant4 can choose the most appropriate unit to use. Just specify the category for the data (Length, Time, Energy, etc...):

  \[
  \text{G4cout} << \text{G4BestUnit}(\text{StepSize}, \text{"Length"});
  \]

  StepSize will be printed in km, m, mm or ... fermi, depending on its value
Defining new units

• New units can be defined directly as constants, or (suggested way) via G4UnitDefinition
  - G4UnitDefinition (name, symbol, category, value)

• Example (mass thickness):
  - G4UnitDefinition ("grammpercm2", "g/cm2", "MassThickness", g/cm2);
  - The new category "MassThickness" will be registered in the kernel in G4UnitsTable

• To print the list of units:
  - From the code
    G4UnitDefinition::PrintUnitsTable();
  - At run-time, as UI command:
    Idle> /units/list
Kinds of G4VPhysicalVolume

- G4PVPlacement 1 Placement = One Volume
  • A volume instance positioned once in a mother volume
- G4PVParameterised 1 Parameterised = Many Volumes
  • Parameterised by the copy number
    - Shape, size, material, position and rotation can be parameterised, by implementing a concrete class of G4VPVParameterisation.
  • Reduction of memory consumption
    - Parameterisation can be used only for volumes that either a) have no further daughters or b) are identical in size & shape.
- G4PVReplica 1 Replica = Many Volumes
  • Slicing a volume into smaller pieces (if it has a symmetry)
Physical Volumes

- **Placement**: it is one positioned volume
- **Repeated**: a volume placed many times
  - can represent any number of volumes
  - reduces use of memory.
- **Replica**
  - simple repetition, similar to G3 divisions
- **Parameterised**

- A **mother** volume can contain **either**
  - many placement volumes **OR**
  - one repeated volume
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 \(n\text{Replicas}\) times using the parameterisation \(p\text{Param}\), within the mother volume
• The positioning of the replicas is dominant along the specified Cartesian axis
  – If \text{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
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;
}
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
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

\[ \text{G4PVDIVISION}(\text{const G4String& pName,} \]
\[ \text{G4LogicalVolume* pDaughterLogical,} \]
\[ \text{G4LogicalVolume* pMotherLogical,} \]
\[ \text{const EAxis pAxis,} \]
\[ \text{const G4int nDivisions, // number of division is given} \]
\[ \text{const G4double offset);} \]

- The size (width) of the daughter volume is calculated as
  \[ \frac{(\text{size of mother}) - \text{offset}}{\text{nDivisions}} \]
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

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

- \text{\textit{nDivisions} daughters of \textit{width} thickness}
Divided Volumes - 6

- Divisions are allowed for the following shapes / axes:
  - \texttt{G4Box} : \texttt{kXAxis}, \texttt{kYAxis}, \texttt{kZAxis}
  - \texttt{G4Tubs} : \texttt{kRho}, \texttt{kPhi}, \texttt{kZAxis}
  - \texttt{G4Cons} : \texttt{kRho}, \texttt{kPhi}, \texttt{kZAxis}
  - \texttt{G4Trd} : \texttt{kXAxis}, \texttt{kYAxis}, \texttt{kZAxis}
  - \texttt{G4Para} : \texttt{kXAxis}, \texttt{kYAxis}, \texttt{kZAxis}
  - \texttt{G4Polycone} : \texttt{kRho}, \texttt{kPhi}, \texttt{kZAxis}
  - \texttt{G4Polyhedra} : \texttt{kRho}, \texttt{kPhi}, \texttt{kZAxis}
    - \texttt{kPhi} - the number of divisions has to be the same as solid sides, (i.e. \texttt{numSides}), the width will \textbf{not} be taken into account
- In the case of division along \texttt{kRho} of \texttt{G4Cons}, \texttt{G4Polycone}, \texttt{G4Polyhedra}, if width is provided, it is taken as the width at the \texttt{-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 Schema

• defines document structure and the list of legal elements
  • materials
    - material, isotope, element, mixture
  • solids
    - box, sphere, tube, cone, polycone, parallelepiped, trapezoid, torus, polyhedra, hyperbolic tube, elliptical tube, ellipsoid
    - boolean solids
  • volumes
    - assembly volumes and reflections
    - replicas and divisions
    - parameterised volumes (position, rotation and size)
      - first implementation
GDML document

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>
    <element name="Nitrogen" formula="N" Z="7.">
      <atom value="14.01"/>
    </element>
    <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 rel="world"/>
      <physvol>
        <volumeref ref="Tracker"/>
        <positionref ref="TrackerinWorldpos"/>
        <rotationref ref="TrackerinWorldrot"/>
      </physvol>
    </volume>
  </structure>
</gdml>
```
GDML – Geant4 binding

• XML schema available from [http://cern.ch/gdml](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](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:

```cpp
#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
- Activate or de-activate schema validation

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

get pointer to 'top' world volume

to read:

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

get pointer to 'top' world volume
• 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);`
  - Example provided in examples/extended/persistency/gdml/G02

- 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 physical volumes)
  - writing out ~10 seconds (on P4 2.4GHz)
  - reading in ~ 5 seconds
  - file size ~2.7 Mb (~40k lines)
- for CMS geometry (~19000 physical 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
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