Geant 4

Geometry, Material, Particle Source

A. Schälicke (DESY)

MCPAD/Helmholtz Training event 2010
28.-30.01. and 4.-6.03.
DESY, Hamburg, 2010
Describing a detector
Part I

Geometry
Mandatory Classes

- Every Geant4 application must implement:
  - G4VUserDetectorConstruction
  - G4VUserPrimaryGeneratorAction
  - G4VUserPhysicsList
DetectorConstruction

What:
- Construct all necessary materials
- Define shapes/solids required to describe the geometry
- Construct and place volumes of your detector geometry
- Define sensitive detectors and identify detector volumes which to associate them (optional)
- Associate magnetic field to detector regions (optional)
- Define visualization attributes for the detector elements (optional)

How:
- Derive your own concrete class from `G4VUserDetectorConstruction` abstract base class.
- Implementing the method `Construct()`:
- Modularize it according to each detector component or sub-detector
DetectorConstruction

● Example: DetectorConstruction.hh

```cpp
#include "G4VUserDetectorConstruction.hh"

class DetectorConstruction : public G4VUserDetectorConstruction {
    public:
        G4VPhysicalVolume* Construct();
        // must return the pointer to the world physical volume
};
```

● Example: DetectorConstruction.cc

```cpp
#include "G4DetectorConstruction.hh"

G4VPhysicalVolume* DetectorConstruction::Construct()
{
    // define you detector here
    // ... 
}
```
Define your Detector volumes

- Three conceptual layers
- Start with its Shape & Size \texttt{G4VSolid}
  - Box \(2 \times 4 \times 8\) cm\(^3\), sphere \(R=7\) m
- Add properties \texttt{G4LogicalVolume}
  - material, B/E field,
  - make it sensitive
- Place it in another volume \texttt{G4VPhysicalVolume}
  - in one place
  - repeatedly using a function
Define your Detector volumes

### Basic strategy

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

- **The world volume**

```cpp
G4VSolid* aBoxSolid =
    new G4Box("aBoxSolid", 1.*cm, 2.*cm, 8.*cm);
G4LogicalVolume* aBoxLog =
    new G4LogicalVolume(aBoxSolid, pBoxMaterial, "aBoxLog");
G4VPhysicalVolume* aBoxPhys =
    new G4PVPlacement(pRotation,
        G4ThreeVector(posX, posY, posZ),
        pBoxLog, "aBoxPhys", pMotherLog,
        0, copyNo);
```

---

**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
Step 1: Solids

- All Solids derived from abstract \texttt{G4VSolid}
- Defines all functions required to compute all necessary information need for the navigation

Solids defined in Geant4:
- CSG (Constructed Solid Geometry) solids
  - \texttt{G4Box}, \texttt{G4Tubs}, \texttt{G4Cons}, \texttt{G4Trd}, ...
- Specific solids (CSG like)
  - \texttt{G4Polycone}, \texttt{G4Polyhedra}, \texttt{G4Hype}, ...
  - \texttt{G4TwistedTubs}, \texttt{G4TwistedTrap}, ...
- BREP (Boundary REPresented) solids
  - \texttt{G4BREPSolidPolycone}, \texttt{G4BSplineSurface}, ...
  - Any order surface
- Boolean solids
  - \texttt{G4UnionSolid}, \texttt{G4SubtractionSolid}, ...
Step 2: Logical Volumes

- 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
- 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

```cpp
G4LogicalVolume(G4VSolid* pSolid, G4Material* pMaterial,
                const G4String& name, G4FieldManager* pFieldMgr=0,
                G4VSensitiveDetector* pSDetector=0,
                G4UserLimits* pULimits=0,
                G4bool optimise=true);
```
How to place a volume?

- 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 must not protrude from the mother volume
    - Daughter volumes must not overlap

- One or more volumes can be placed in a mother volume
Step 3: Physical Volumes

- **G4PVPlacement**
  1 Placement = One Volume
  - Places a volume once inside a mother volume
  - this is the simplest type of physical volume
  - you can create many placements using the same logical volume
G4PVPlacement

- 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
Example - Rotation

- Single volume positioned relatively to the mother volume
  1. translate the frame origin
  2. rotate the frame
  3. place the object at the origin of the resulting frame

```c++
G4RotationMatrix * rm = new G4RotationMatrix();
rm->rotateY(dutTheta); // rotation angle
physiSecondSensor =
    new G4PVPlacement(rm, // rotation matrix
    G4ThreeVector(0., 15.*mm, -25.*mm), // translation
    logicSensorPlane,
    "DeviceUnderTest",
    logicWorld,
    false,
    1);
```

Mittwoch, 3. März 2010
Task 1.1 a

- Tutorial Material online:
  - http://www.ifh.de/geant4/g4course2010

- Exercise 1
  - place a sensor plane using G4PVPlacement

- Exercise 2
  - rotate the central sensor plane using G4RotationMatrix
One logical volume can be placed more than once.

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
- The most simple shape to describe the world is a box
Physical Volumes - 2

- **G4PVPlacement**
  - 1 Placement = One Volume
  - One volume instance positioned in the mother volume

- **G4PVReplica**
  - 1 Replica = Many Volumes
  - Slices a volume into smaller pieces
    (if it has a symmetry)
G4PVReplica

- The mother volume is sliced into pieces = replicas
  - together all pieces must fill up the mother volume
  - typically all pieces are of same size and dimension
- The replica represents many (touchable) detector elements
  - they differ in their position
- 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

Features and restrictions:
- Replicas can be placed inside other replicas
- Normal placement volumes can be placed inside replicas, assuming no intersection/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

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

Features and restrictions:
- Replicas can be placed inside other replicas
- Normal placement volumes can be placed inside replicas, assuming no intersection/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

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

- **Cartesian axes** - \( kXaxis, kYaxis, kZaxis \)
  - Offset shall not be used
  - Center of \( n \)-th daughter is given as
    \[-width\times(nReplicas-1)\times0.5+n\times width\]

- **Radial axis** - \( kRaxis \)
  - Center of \( n \)-th daughter is given as
    \[width\times(n+0.5)+offset\]

- **Phi axis** - \( kPhi \)
  - Center of \( n \)-th daughter is given as
    \[width\times(n+0.5)+offset\]
Physical Volumes - 3

- **G4PVPlacement**
  - 1 Placement = One Volume
  - A volume instance positioned once in a mother volume

- **G4PVReplica**
  - 1 Replica = Many Volumes
  - Slicing a volume into smaller pieces (if it has a symmetry)
  - Replicas can be placed inside other replicas
  - Shape of all daughter volumes must be same shape as the 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
    - Currently: parameterisation can be used only for volumes that either a) have no further daughters or b) are identical in size & shape.
Task 1.1 b

- Tutorial Material online:
  - http://www.ifh.de/geant4/g4course2010

- Exercise 3
  - subdivide all sensor planes using G4PVReplica
Describing a detector
Part II

Material
Definition of Materials

- Each Logical Volume has a pointer to its Material
- Different kinds of materials can be defined:
  - isotopes <-> G4Isotope
  - elements <-> G4Element
  - molecules <-> G4Material
  - compounds and mixtures <-> G4Material
- Attributes associated:
  - state, density
  - possibly temperature, pressure
    - for a gas
    - may effect dE/dx
Material of one element

- most simple case:
  - single element material

```c
G4double density = 1.390*g/cm3;
G4double a = 39.95*g/mole;
G4Material* lAr =
  new G4Material("liquidArgon",z=18.,a,density);
```

- Prefer low-density material to vacuum
Material: molecule

- A Molecule is made of several elements (composition by integer number of atoms):

```cpp
G4double z, a, density;
G4int natoms, ncomp;
G4String symbol;

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);
H2O->GetIonisation()->SetMeanExcitationEnergy(78.*eV);
```
Material: compound

- Compound: composition by fraction of mass

```cpp
G4double z, a, density;
G4int natoms, ncomponents;
G4String symbol, name;

a = 14.01*g/mole;
G4Element* elN =
    new G4Element(name="Nitrogen",symbol="N",z= 7.,a);
a = 16.00*g/mole;
G4Element* elO =
    new G4Element(name="Oxygen",symbol="O",z= 8.,a);
density = 1.290*mg/cm3;
G4Material* Air =
    new G4Material(name="Air",density,ncomponents=2);
Air->AddElement(elN, 70.0*perCent);
Air->AddElement(elO, 30.0*perCent);
```

- Note: meaning of AddElement differs if called with integer or float!
NIST Manager

- No need to predefine elements and materials (since G4 7.1)
- Retrieve materials from NIST manager:

```cpp
G4NistManager* manager = G4NistManager::Instance();
G4Element* elm = manager->FindOrBuildElement("Ga", iso=true);
G4Element* elm = manager->FindOrBuildElement(Z=33, iso=true);
G4Material* mat = manager->FindOrBuildMaterial("G4_Air", iso=true);
G4Material* mat = manager->ConstructNewMaterial("name",
    const std::vector<G4String>& sym,
    const std::vector<G4double>& weight,
    G4double density, G4bool iso);
G4double isotopeMass = manager->GetMass(G4int Z, G4int N);
```

- Useful UI commands ...

```bash
# print defined elements and material
/material/nist/printElement  <elementname>
/material/nist/listMaterials
```
### Elementary Materials from the NIST Data Base

<table>
<thead>
<tr>
<th>Z</th>
<th>Name</th>
<th>ChFormula</th>
<th>density (g/cm^3)</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></td>
<td>0.0011652</td>
<td>82</td>
</tr>
<tr>
<td>8</td>
<td>G4_O</td>
<td></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>

### Compound Materials from the NIST Data Base

<table>
<thead>
<tr>
<th>N</th>
<th>Name</th>
<th>ChFormula</th>
<th>density (g/cm^3)</th>
<th>I(eV)</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>G4_Adipose_Tissue</td>
<td></td>
<td>0.92</td>
<td>63.2</td>
</tr>
</tbody>
</table>

- NIST Elementary Materials
- NIST Compounds
- HEP Materials ...
- It is possible to build mixtures of NIST and user-defined materials
Summary of Materials

- Each Logical Volume has a pointer to its Material
- Different kinds of materials can be defined:
  - isotopes <-> G4Isotope
  - elements <-> G4Element
  - molecules <-> G4Material
  - compounds and mixtures <-> G4Material
- Attributes associated:
  - density, state, temperature, pressure,
    - most effect dE/dx
- Relations:
  - G4Element may contain many G4Isotopes
  - G4Materials may consist of many G4Elements
  - complex Materials may be composed of other Materials
Task 1.1 c

- Tutorial Material online:
  - http://www.ifh.de/geant4/g4course2010

- Exercise 1.1.4
  - change sensor material to `G4_GALLIUM_ARSENIDE`
  - create a customized material
Primary Generator Action

Particle Source
Mandatory User Classes

- User Action:
  - PrimaryGeneratorAction

```
G4VUserDetectorConstruction
   \arrow{use}
   DetectorConstruction
   \arrow{use}
   Construct()

G4VUserPrimaryGeneratorAction
   \arrow{use}
   PrimaryGeneratorAction
   \arrow{use}
   \arrow{create}
   \arrow{create}
   GeneratePrimaries()

G4VUserPhysicsList
   \arrow{use}
   \arrow{create}
   PhysicsList
   \arrow{create}
   \arrow{create}
   # ConstructParticle()
   # ConstructProcess()
   # SetCuts()

G4RunManager
   \arrow{create}
   user main program
```
G4VUserPrimaryGeneratorAction

- Controls the generation of primary particles (‘primaries’)
  - What kind of particle, what energy, (how many)
  - Where: position, direction, polarisation, etc

- Must invoke GeneratePrimaryVertex() of primary generator(s) to make each primary (G4VPrimaryGenerator)

- Geant4 provides some several implementations of G4VPrimaryGenerator:
  - G4ParticleGun - simplest
  - G4GeneralParticleSource - versatile
  - G4HEPEvtInterface, G4HEPMMCInterface - read in
G4ParticleGun

- Concrete implementations of G4VPrimaryGenerator
  - A good example for experiment-specific primary generator implementation

- It **shoots one primary particle** of a certain energy from a certain point at a certain time to a certain direction.
  - Various C++ set methods are available
  - UI commands are also available for setting initial values

- /gun/List
  - List available particles
- /gun/particle
  - Set particle to be generated
- /gun/direction
  - Set momentum direction
- /gun/energy
  - Set kinetic energy
- /gun/momentum
  - Set momentum
- /gun/position
  - Set starting position of the particle
- ...


Mittwoch, 3. März 2010
To implement your own, you must write

- **Constructor**
  - Instantiate primary generator(s)
  - Set default values to it (them)

- **GeneratePrimaries() method**
  - Randomize particle-by-particle value(s)
  - Set these values to primary generator(s)
  - Invoke GeneratePrimaryVertex() method of primary generator(s)

- **G4ParticleGun** can be employed in most cases
  - used in the series of examples, but
  - users still needs to code (C++) almost every change and
  - add related UI commands for interactive control
**G4GeneralParticleSource**

- **Requirements** for advanced primary particle modelling are often **common to many users** in different communities
  - E.g. uniform vertex distribution on a surface, isotropic generation, energy spectrum, ...

- **G4GeneralParticleSource** offers
  - an advanced concrete implementation of G4VPrimaryGenerator
  - pre-defined many common (and not so common) options
  - Position, angular and energy distributions
  - Multiple sources, with user defined relative intensity
  - Capability of event biasing (variance reduction).
  - All features can be used via C++ or via UI command line (or macro)
G4GeneralParticleSource

- can be extremely simple

```cpp
#include "G4GeneralParticleSource.hh"

PrimaryGeneratorAction::PrimaryGeneratorAction()
{
    particleGun = new G4GeneralParticleSource();
}

PrimaryGeneratorAction::~PrimaryGeneratorAction()
{
    delete particleGun;
}

void PrimaryGeneratorAction::GeneratePrimaries(G4Event* anEvent) {
    particleGun->GeneratePrimaryVertex(anEvent);
}
```

- All user instructions given via macro UI commands
G4GeneralParticleSource

- some UI commands:

```plaintext
# simple commands
/gps/energy 2. GeV
/gps/position 0. 0. 0. m
/gps/direction 0. 0. 1.
#
# Gauss distribution in position
/gps/pos/type Beam
/gps/pos/sigma_x 0.1 mm
/gps/pos/sigma_y 0.1 mm
#
# Gauss distribution in angle
/gps/ang/type beam2d
/gps/ang/sigma_x 0.1 mrad
/gps/ang/sigma_y 0.1 mrad
/gps/ang/rot1 -1. 0. 0.
```
Task 1.1 d

- Tutorial Material online:
  - http://www.ifh.de/geant4/g4course2010

- Exercise 1.1.5
  - implement G4GeneralParticleSource in PrimaryGeneratorAction.cc
Task 1.1 d

- Tutorial Material online:
  - http://www.ifh.de/geant4/g4course2010

- Exercise 1.1.6 (advanced)
  - implement the Device Under Test (DUT)