Geant4 Materials

Geant4-11.1 reference - Based on previous Geant4 courses

Lorenzo Pezzotti
CERN EP-SFT

GitHub Code for this tutorial on GitHub

10th International Geant4 Course in Korea 2023
@ Jeju National University

Covered topics

- **Materials in `Geant4`**
- **Definition of materials**
- **The `G4NistManager`**

… how to associate materials to volumes is covered in the Geometry lessons

Materials in Geant4 (1/2)

Materials (`G4Material`) in Geant4 (and in the real world) are made of isotopes (`G4Isotope`), elements (`G4Element`), molecules and mixtures

Isotopes → Elements → Molecules → Mixtures

- Materials can be solid, liquid or gaseous and
- exist under various temperatures and pressure states

Materials in `Geant4` (2/2)

- In `Geant4` it is mandatory to set the density of each material
- On the other hand, the following materials conditions are optional
  - State (`kStateSolid, kStateLiquid, kStateGas`), by default is solid or liquid depending on the density
  - Temperature, by default temperature is $T = 273.15$ K
  - Pressure, by default pressure is 100 kPascal $\simeq 1$ atm

NOTE: if not needed otherwise, it is suggested to use information from the predefined NIST material database (explained later in this lesson)

Single-element material (1/3)

If a material is made of a *single element* (Fe, Pb, …) its definition is straightforward

DetectorConstruction.cc - Create pure iron material

```
#include "G4Material.hh"
#include "G4SystemOfUnits.hh"

G4VPhysicalVolume* DetectorConstruction::Construct(){

  G4int atomic_number = 26;                    // iron atomic number (z)
  G4double density = 7.87*g/cm3;               // iron density g/cm3
  G4double atomic_weight = 55.845*g/mole;  // iron atomic weight 55.845
  G4Material* pureIron = new G4Material("pureIron", atomic_number, atomic_weight, density);

}
```

Single-element material

Here are some useful *getters* from `G4Material`

DetectorConstruction.cc - Create pure iron material and get some information

```
#include "G4Material.hh"
#include "G4SystemOfUnits.hh"

G4VPhysicalVolume* DetectorConstruction::Construct(){

  G4int atomic_number = 26;                    // iron atomic number (z)
  G4double density = 7.87*g/cm3;               // iron density g/cm3
  G4double atomic_weight = 55.845*g/mole;      // iron atomic weight 55.845
  G4Material* pureIron = new G4Material("pureIron", atomic_number, atomic_weight, density);
  G4cout<<pureIron->GetName()<<"->"<<'\n'<<
        "density: "<<pureIron->GetDensity()/(g/cm3)<<" g/cm3"<<'\n'<<
        "number of elements: "<<pureIron->GetNumberOfElements()<<'\n'<<
        "radiation length: "<<pureIron->GetRadlen()<<" mm"<<'\n'<<
        "nucl. int. length: "<<pureIron->GetNuclearInterLength()<<" mm"<<'\n'<<
        "e- per volume: "<<pureIron->GetTotNbOfElectPerVolume()<<" e-/mm3"<<G4endl;

}
```

Single-element material (3/3)

Here are some useful *getters* from `G4Material`

Bash - execution

```
pureIron->
density: 7.87 g/cm3
number of elements: 1
radiation length: 17.5839 mm
nucl. int. length: 169.989 mm
e- per volume: 2.20655e+21 e-/mm3
```

NOTE: good example of code *modularity*, `Geant4` is a toolkit not a software

Molecules

A molecule is defined by (> 1) elements and its composition is specified with the `AddElement` method

DetectorConstruction.cc - Create water material as H_2O molecule

```
#include "G4Material.hh"
#include "G4SystemOfUnits.hh"
#include "G4Element.hh"

G4VPhysicalVolume* DetectorConstruction::Construct(){

  G4int ncomp = 2; // water components
  G4double water_density = 1.0*g/cm3; // water density
  G4Material* H2O = new G4Material(“Water",water_density,ncomp); // water material
  G4double a = 1.01*g/mole; // Hydrogen(element): A
  G4int z = 1; // Hydrogen(element): Z
  G4Element* elH = new G4Element(“Hydrogen”,"H",z,a); // Hydrogen(element)
  a = 16.00*g/mole;
  z = 8;
  G4Element* elO = new G4Element("Oxygen","O",z,a);
  G4int nAtoms;
  H2O->AddElement(elH, nAtoms=2); // add element by number of atoms
  H2O->AddElement(elO, nAtoms=1); // add element by number of atoms

}
```

Mixtures (1/2)

Similarly to combining elements in molecules, it is possible to *combine materials and elements* in mixtures

DetectorConstruction.cc - Construct simplified air mixture

```
#include "G4Material.hh"
#include "G4SystemOfUnits.hh"
#include "G4Element.hh"

G4VPhysicalVolume* DetectorConstruction::Construct(){

    // Simplified Air, mass fraction: 70% Nitrogen, 30% Oxigen
    //
    G4int z;
    G4int ncomponents;
    G4double a = 14.01*g/mole;
    G4Element* elN = new G4Element(name="Nitrogen",symbol="N",z= 7.,a);
    G4double 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); //add element by frac mass
    Air->AddElement(elO, 30.0*perCent); //add element by frac mass

}
```

Mixtures (2/2)

Similarly to combining elements in molecules, it is possible to *combine materials and elements* in mixtures

DetectorConstruction.cc - Construct aerogel mixture

```
#include "G4Material.hh"
#include "G4SystemOfUnits.hh"
#include "G4Element.hh"

G4VPhysicalVolume* DetectorConstruction::Construct(){

    // Simplified Aereogel material, frac mass: 62.5% SiO2, 37.4% H20
    //                                           0.1% C
    G4Element* elC = ... ;   // define carbon element
    G4Material* H2O = ... ;  // define water molecule (previously done)
    G4Material* SiO2 = ... ; // define SiO2 molecule (left as exercise)

    G4double density = 0.20*g/cm3;
    G4int ncomp = 3;
    G4double fracMass;
    G4Material* Aerog = new G4Material(“Aerogel”, density, ncomp); // Aereogel material
    Aerog->AddMaterial(SiO2, fracMass = 62.5*perCent); // Note that we are combining elements with materials
    Aerog->AddMaterial(H2O, fracMass = 37.4*perCent);  // Note that we are combining elements with materials
    Aerog->AddElement(elC, fracMass = 0.1*perCent);    // Note that we are combining elements with materials

}
```

Isotopes (1/2)

By default any `G4Element` is treated according to its *natural isotopic abundance* regardless of the atomic weight specified

- This choice is needed because hadronic processes only work with specific isotopes (not elements)

DetectorConstruction.cc - Retrieve number of iron isotopes

```
G4VPhysicalVolume* DetectorConstruction::Construct(){

    G4int atomic_number = 26;
    G4double density = 7.87*g/cm3;
    G4double atomic_weight = 55.845*g/mole;
    G4Material* pureIron = new G4Material("pureIron", atomic_number, atomic_weight, density);
    auto element = pureIron->GetElement(0);
    G4cout<<pureIron->GetName()<<" # isotopes: "<<element->GetNumberOfIsotopes()<<G4endl;
}
```

Bash - execution

```
pureIron # isotopes: 4
```

→ Iron has 4 naturally abundant isotopes: $^{54}Fe, ^{56}Fe, ^{57}Fe, ^{58}Fe$

Isotopes (2/2)

It is possible to create an element with *non natural isotopic abundance* assigning to it a list of `G4Isotopes`

DetectorConstruction.cc - Create nuclear fuel (UF6 - uranium hexafluoride)

```
G4VPhysicalVolume* DetectorConstruction::Construct(){

    G4int z_iso, a_iso; // Create uranium 235 and 238
    G4Isotope* u235 = new G4Isotope("U235",z_iso=92,a_iso=235.,235.044*g/mole);
    G4Isotope* u238 = new G4Isotope("U238",z_iso=92,a_iso=238.,238.051*g/mole);

    G4int u_comp;        // Create enriched uranium
    G4double u_abundance;
    G4Element* enrichedU = new G4Element("enrichedU","eU",u_comp=2);
    enrichedU->AddIsotope(u235,u_abundance=5.0*perCent);   // add isotope to enrichedU
    enrichedU->AddIsotope(u238,u_abundance=95.0*perCent);  // add isotope to enrichedU

    G4Element* elF = new G4Element("Fluorine","F",9,18.998*g/mole);

    // Create nuclear fuel (UF6)
    //
    G4Material* fuel = new G4Material("NuclearFuel",5.09*g/cm3,ncomp=2,  // mandatory arguments
                                  kStateGas,640*kelvin,1.5e7*pascal);// optional arguments
                                                                     // set to STP if not specified
    fuel->AddElement(elF,6);
    fuel->AddElement(enrichedU,1);
}
```

The NIST material database

- The NIST material database includes
  - most of the materials useful in simulations (biomedical, space-science, …)
  - all elements with natural isotopic abundance
  - > 3000 isotopes
- Using materials from the NIST database guarantees the best parameters for
  - density
  - isotopic composition of elements
  - element composition of materials
  - mean ionization potential
  - chemical bonds

Example: Print list of NIST materials

```
/material/nist/listMaterials    # UI command
```

Bash - execution

```
=======================================================
###   Simple Materials from the NIST Data Base      ###
=======================================================
 Z   Name   density(g/cm^3)  I(eV)
=======================================================
 1    G4_H    8.3748e-05       19.2
 2   G4_He   0.000166322       41.8
 3   G4_Li         0.534         40
 4   G4_Be         1.848       63.7
 5    G4_B          2.37         76
 6    G4_C             2         81
 7    G4_N     0.0011652         82
 8    G4_O    0.00133151         95
 9    G4_F    0.00158029        115
10   G4_Ne   0.000838505        137
11   G4_Na         0.971        149
12   G4_Mg          1.74        156
13   G4_Al         2.699        166
```

Using the NIST material database (1/2)

`Geant4` provides a (singleton) class, the `G4NistManager`, to handle information from the NIST database

DetectorConstruction.cc - Get iron element and material from NIST database

```
G4VPhysicalVolume* DetectorConstruction::Construct(){

    //Get nist manager (singleton)
    //
    G4NistManager* nist = G4NistManager::Instance();

    G4int Z_nist;
    G4Element* elFe = nist->FindOrBuildElement(Z_nist=26); //Get iron element from NIST

    G4Material* fe = nist->FindOrBuildMaterial("G4_Fe");   //Get iron material from NIST

}
```

Using the NIST material database (2/2)

`Geant4` provides a (singleton) class, the `G4NistManager`, to handle information from the NIST database

- It is possible to get information on elements/materials even without creating them

DetectorConstruction.cc - Get iron element and material information directly from NIST database

```
G4VPhysicalVolume* DetectorConstruction::Construct(){

    G4NistManager* nist = G4NistManager::Instance();

    G4cout<<"Natural uranium mass: "<<nist->GetAtomicMassAmu(92)<<" amu"<<'\n'<<
            "Hydrogen Nb of isotopes: "<<nist->GetNumberOfNistIsotopes(1)<<'\n'<<
            "Iron material density: "<<nist->GetNominalDensity(26)/(g/cm3)<<" g/cm3"<<G4endl;

    // Or print information per material/elements using
    nist->PrintElement("Al");    // equivalent to UI command /material/nist/printElement Al
}
```

Bash - execution

```
Natural uranium mass: 238.029 amu
Hydrogen Nb of isotopes: 6
Iron material density: 7.874 g/cm3
```

Recap of Geant4 materials

- In `Geant4`, materials (`G4Material`) are defined in terms of elements (`G4Element`) and isotopes (`G4Isotopes`)
  - Materials can be made of single-elements or molecules (multi-elements), or,
  - they can be mixtures of elements and materials (by mass fraction)
- Every material exists under a certain density condition (mandatory) as well as state, temperature and pressure condition (optional)
- Whenever possible, it is recommended to use the `G4NistManager` to retrieve both elements and materials from the NIST Material Database