# Geant 4

*Geometry, Material, Particle Source*

A. Schälicke (DESY)

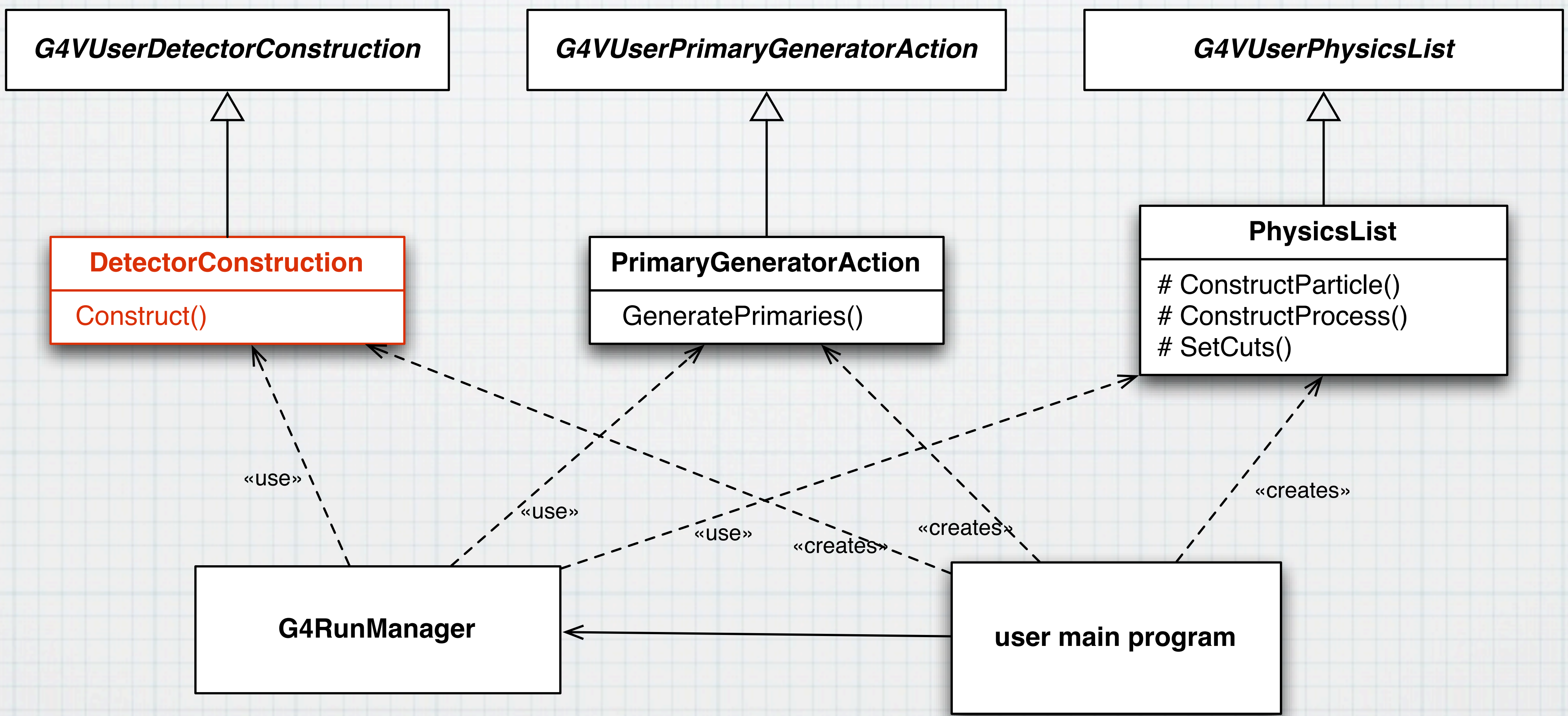
MCPAD/Helmholtz Training event
28.-30.01.2010
DESY, Hamburg

# 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

```
#include "G4VUserDetectorConstruction.hh"

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

- Example: DetectorConstruction.cc

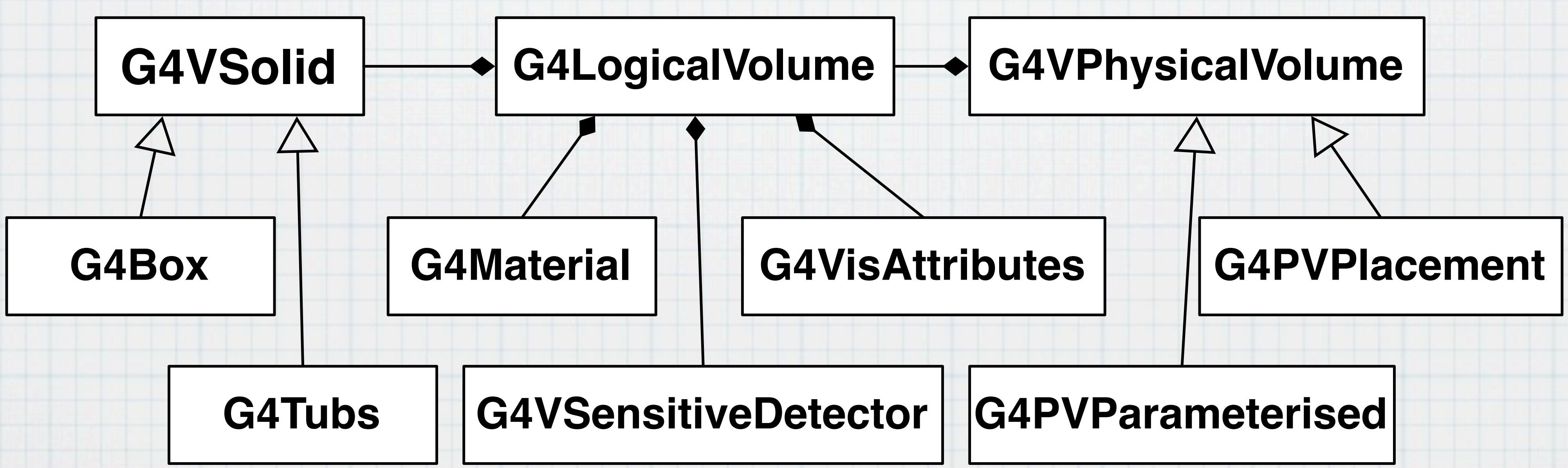
```
#include "G4DetectorConstruction.hh"

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

# Define your Detector volumes

**Important!**

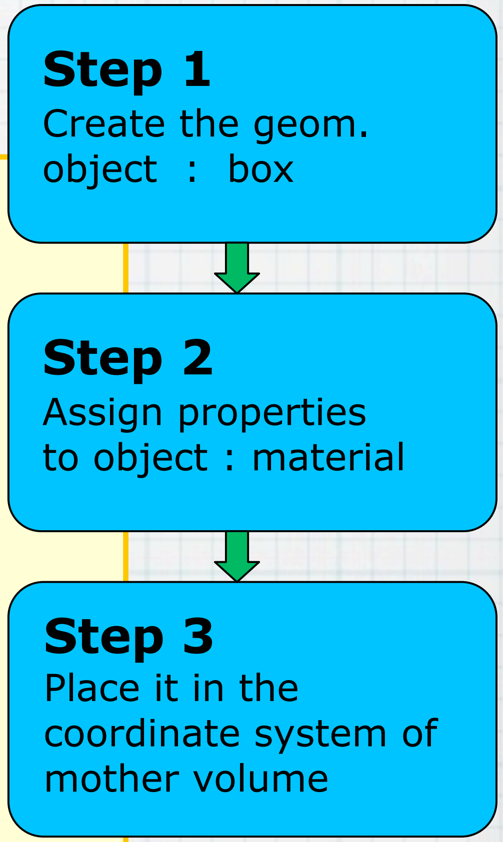
- Three conceptual layers
- Start with its Shape & Size `G4VSolid`
  - Box 2x4x8 $cm^3$, sphere R=7 m
- Add properties `G4LogicalVolume`
  - material, B/E field,
  - make it sensitive
- Place it in another volume `G4VPhysicalVolume`
  - in one place
  - repeatedly using a function



# Define you Detector volumes

- **Basic strategy**

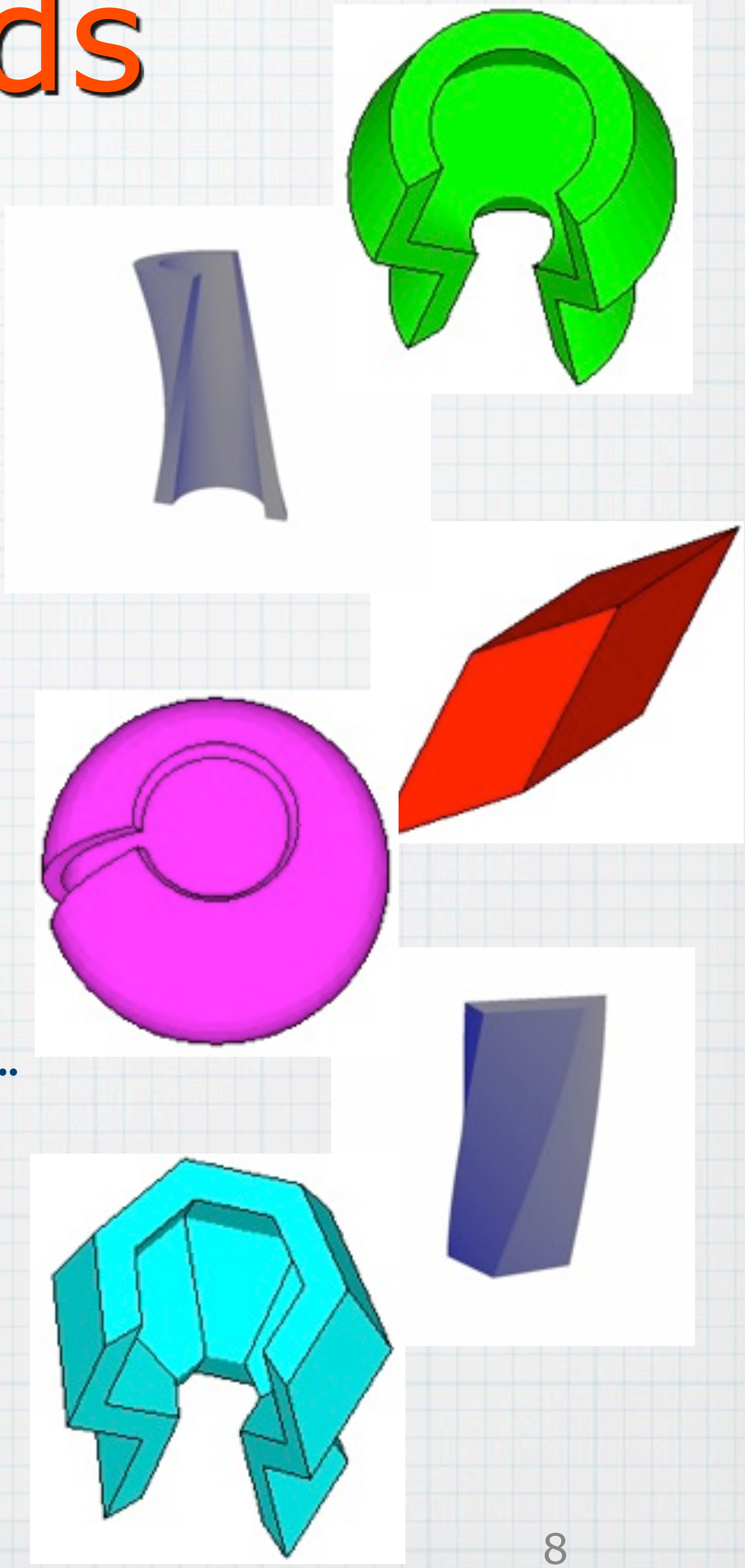
```
G4VSolid* aBoxSolid =
   new G4Box(“aBoxSolid”, 1.*cm, 2.*cm, 8.*cm);
G4LogicalVolume* aBoxLog =
   new G4LogicalVolume( aBoxSolid, pBoxMaterial,
                         “aBoxLog”);
G4VPhysicalVolume* aBoxPhys =
   new G4PVPlacement( pRotation,
                      G4ThreeVector(posX, posY, posZ),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




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

# Step 1 : Solids

- All Solids derived from abstract `G4VSolid`
  - Defines all functions required to compute all necessary information need for the navigation
- Solids defined in Geant4:
  - CSG (Constructed Solid Geometry) solids
    - `G4Box, G4Tubs, G4Cons, G4Trd, …`
  - Specific solids (CSG like)
    - `G4Polycone, G4Polyhedra, G4Hype, …`
    - `G4TwistedTubs, G4TwistedTrap, …`
  - BREP (Boundary REPresented) solids
    - `G4BREPSolidPolycone, G4BSplineSurface, …`
    - `Any order surface`
  - Boolean solids
    - `G4UnionSolid, G4SubtractionSolid, …`



# Step 2: Logical Volumes

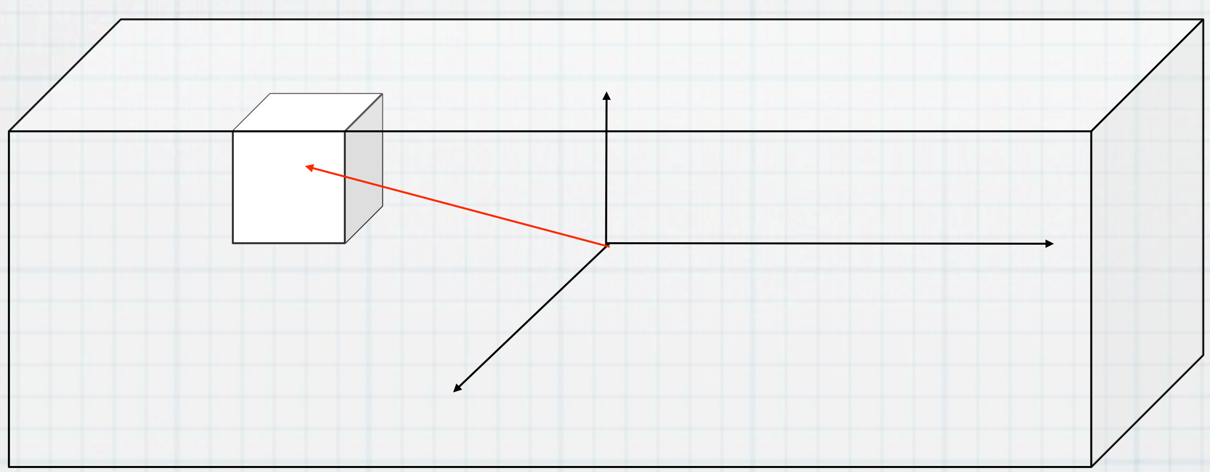
for Reference

```
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
- 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

- 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

for Reference

```
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
```

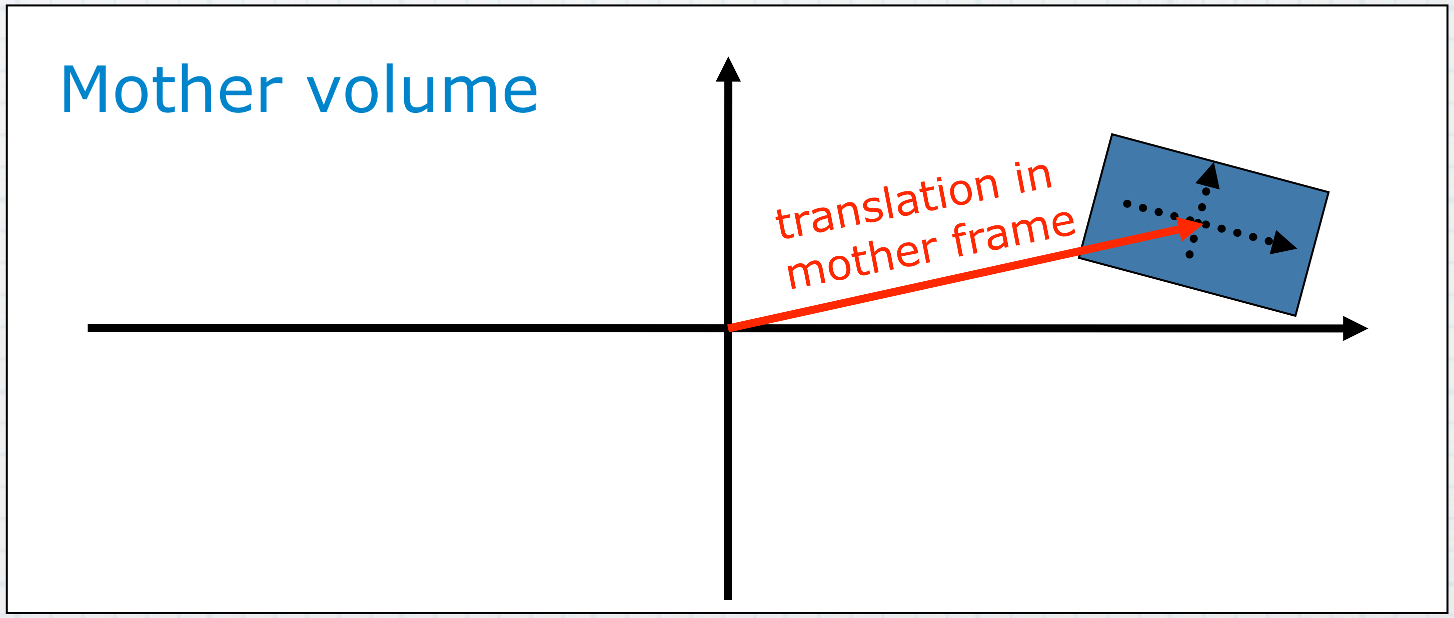
- Single volume positioned relatively to the mother volume
  - In a frame rotated and translated relative to the coordinate system of the mother volume
- Three additional constructors:
  - A simple variation: specifying the mother volume as a pointer to its physical volume instead of its logical volume.
  - 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

# Example - Rotation

```
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);
```

- 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

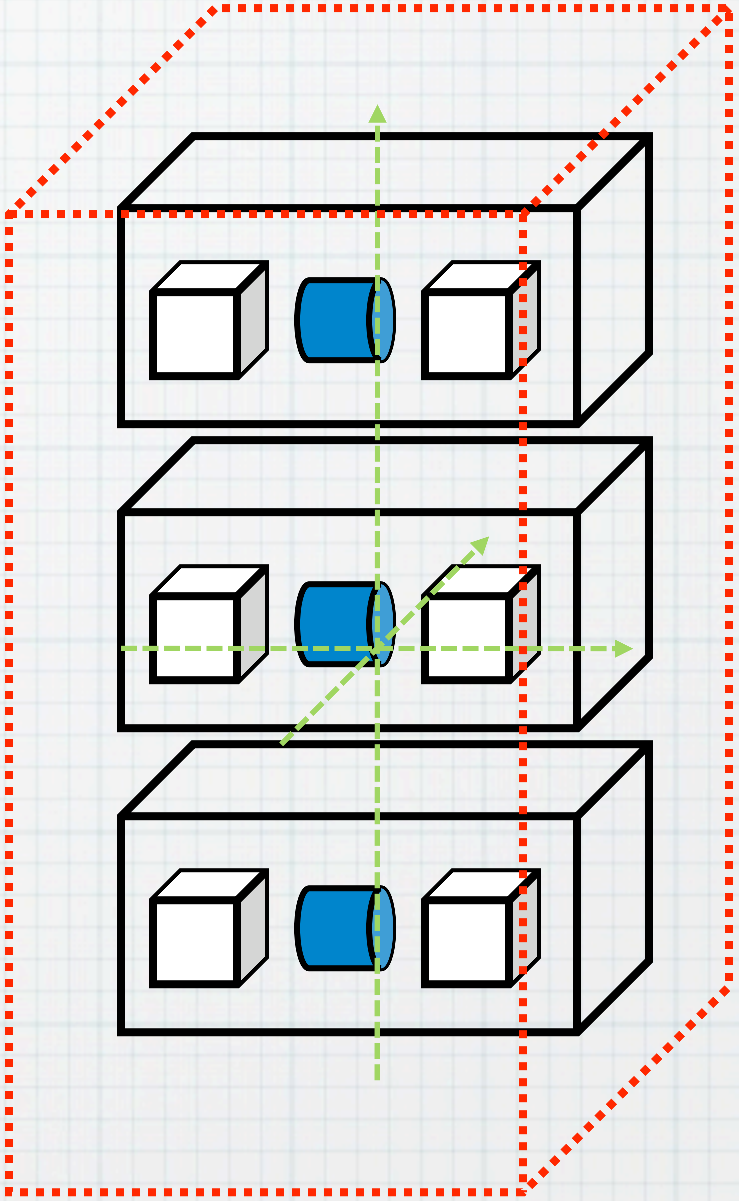


# Task 1.1 a

- 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`

# Geometrical Hierarchy - 2

- 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

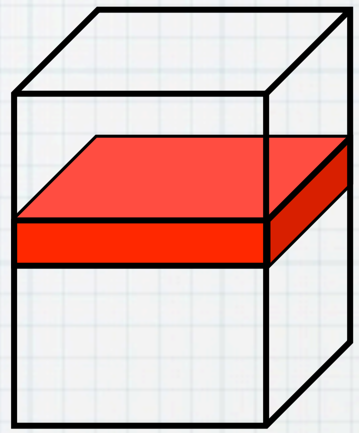


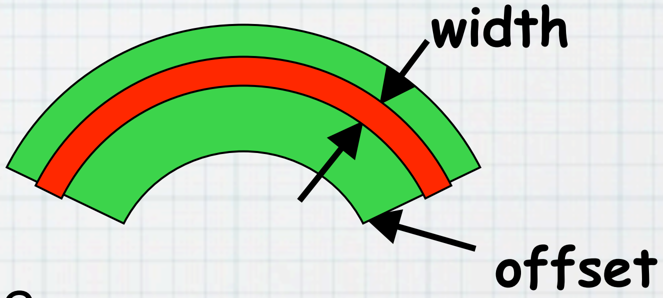
# Physical Volumes - 2

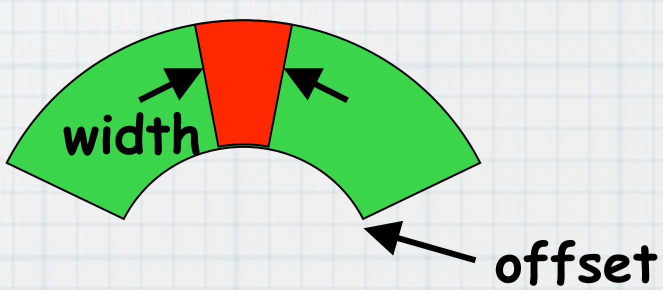
- G4PVPlacement 1 Placement = One Volume
  - One volume instance positioned in the mother volume
- G4PVReplica 1 Replica = Many Volumes
  - Slices a volume into smaller pieces
    (if it has a symmetry)

# 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

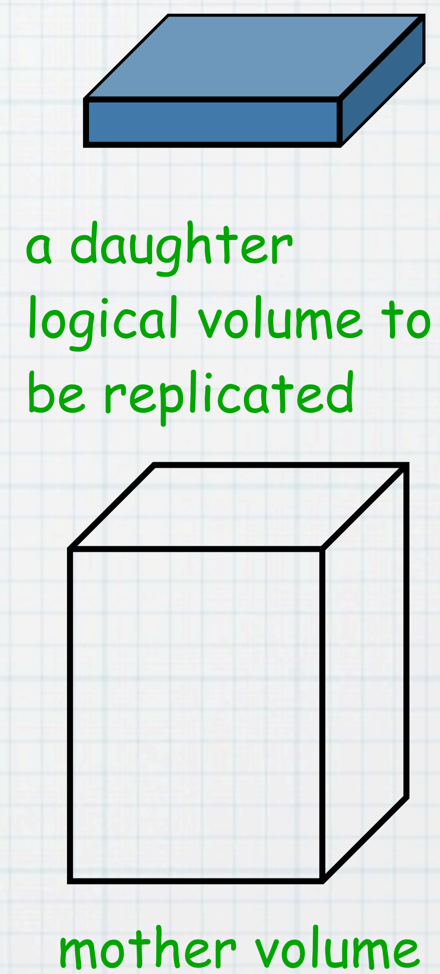




# G4PVReplica

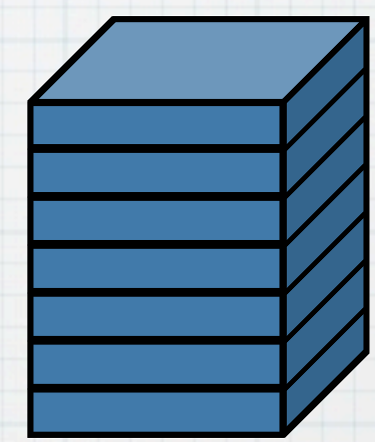
```
G4PVReplica(const G4String& pName,
            G4LogicalVolume* pCurrentLogical,
            G4LogicalVolume* pMotherLogical,
            const EAxis pAxis,
            const G4int nReplicas,
            const G4double width,
            const G4double offset=0);
```




- 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

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

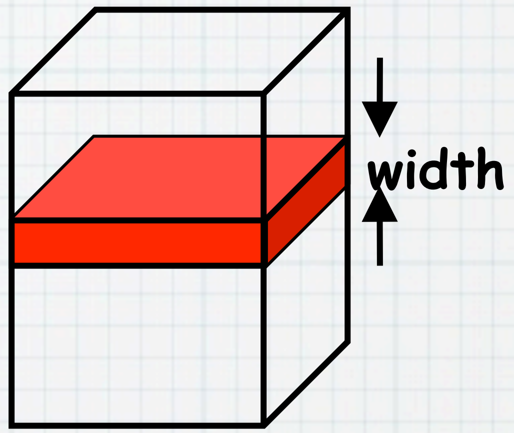




- Features and restrictions:
  - Replicas can be placed inside other replicas
  - Normal placement volumes can be placed inside replicas, assuming no intersection/overlaps with the mother volume or with other replicas
  - No volume can be placed inside a *radial* replication
  - Parameterised volumes cannot be placed inside a replica

# Replica – axis, width, offset

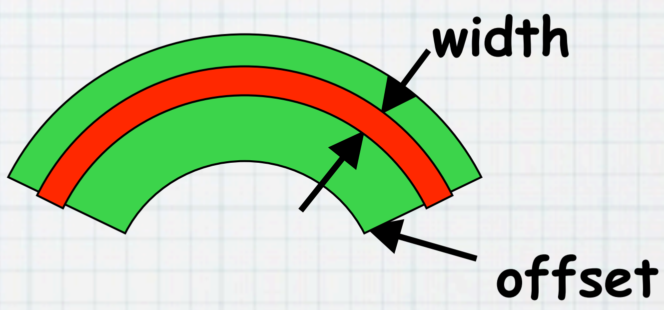
- Cartesian axes - **`kXaxis, kYaxis, kZaxis`**
  - offset shall not be used
  - Center of n-th daughter is given as

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



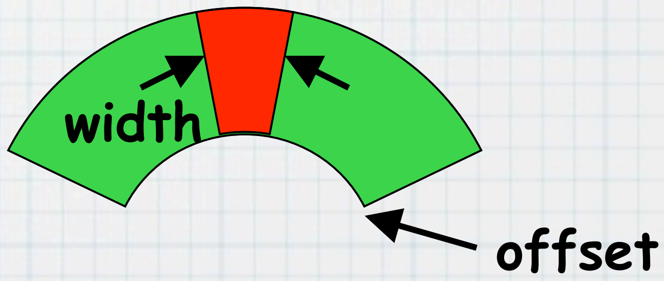

- Radial axis - **`kRaxis`**
  - Center of n-th daughter is given as

    **`width*(n+0.5)+offset`**




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

    **`width*(n+0.5)+offset`**



# 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

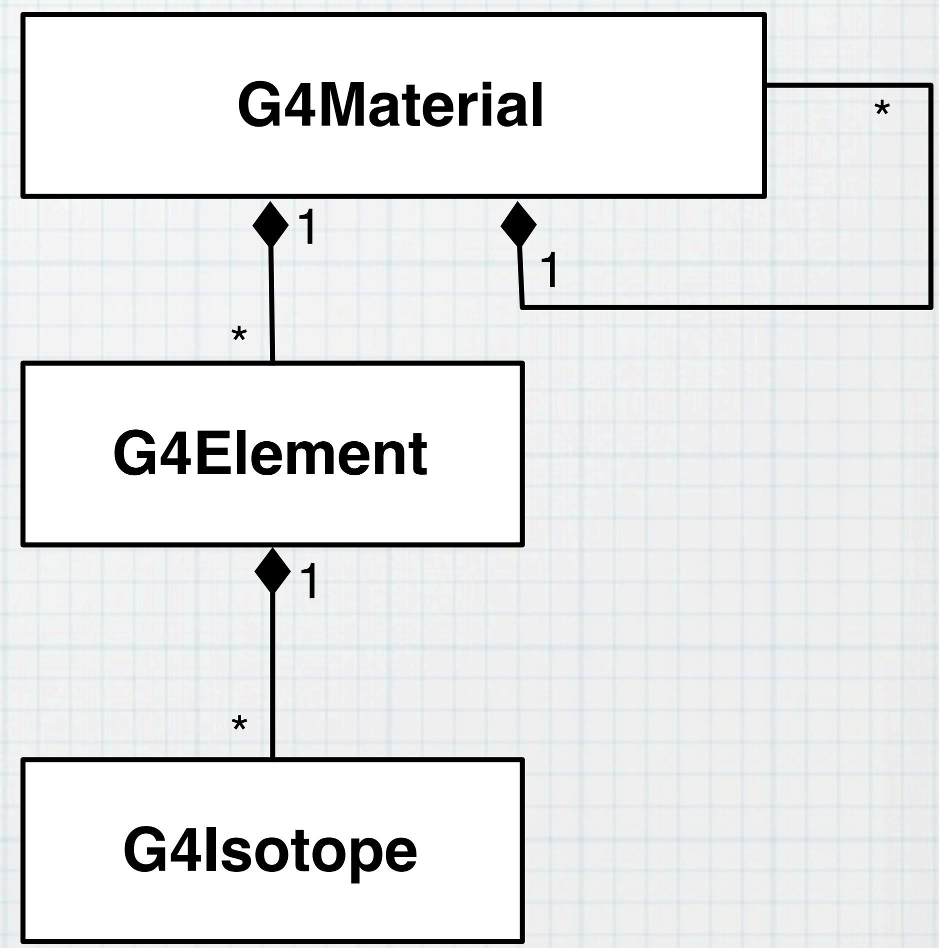
- Exercise 3
    - subdivide all sensor planes using `G4PVReplica`

# 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

```
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):

```
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.);
```

# Material: compound

- Compound: composition by fraction of mass

```
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:

```
G4NistManager* manager = G4NistManager::GetPointer();

G4Element* elm = manager->FindOrBuildElement("symb", G4bool iso);

G4Element* elm = manager->FindOrBuildElement(G4int Z, G4bool iso);

G4Material* mat = manager->FindOrBuildMaterial("name", G4bool iso);

G4Material* mat = manager->ConstructNewMaterial("name",
                             const std::vector<G4int>& Z,
                             const std::vector<G4double>& weight,
                             G4double density, G4bool iso);

G4double isotopeMass = manager->GetMass(G4int Z, G4int N);
```

- Useful UI commands ...

```
# print defined elements and material
/material/nist/printElement
/material/nist/listMaterials
```

# NIST Materials

=====================================

### Elementary Materials from the NIST Data Base

=====================================

| Z | Name | ChFormula | density(g/cm^3) | I(eV) |
|---|---|---|---|---|
| 1 | G4_H | H_2 | 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 | N_2 | 0.0011652 | 82 |
| 8 | G4_O | O_2 | 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 | | 2.6989 | 166 |
| 14 | G4_Si | | 2.33 | 173 |

=====================================

### Compound Materials from the NIST Data Base

=====================================

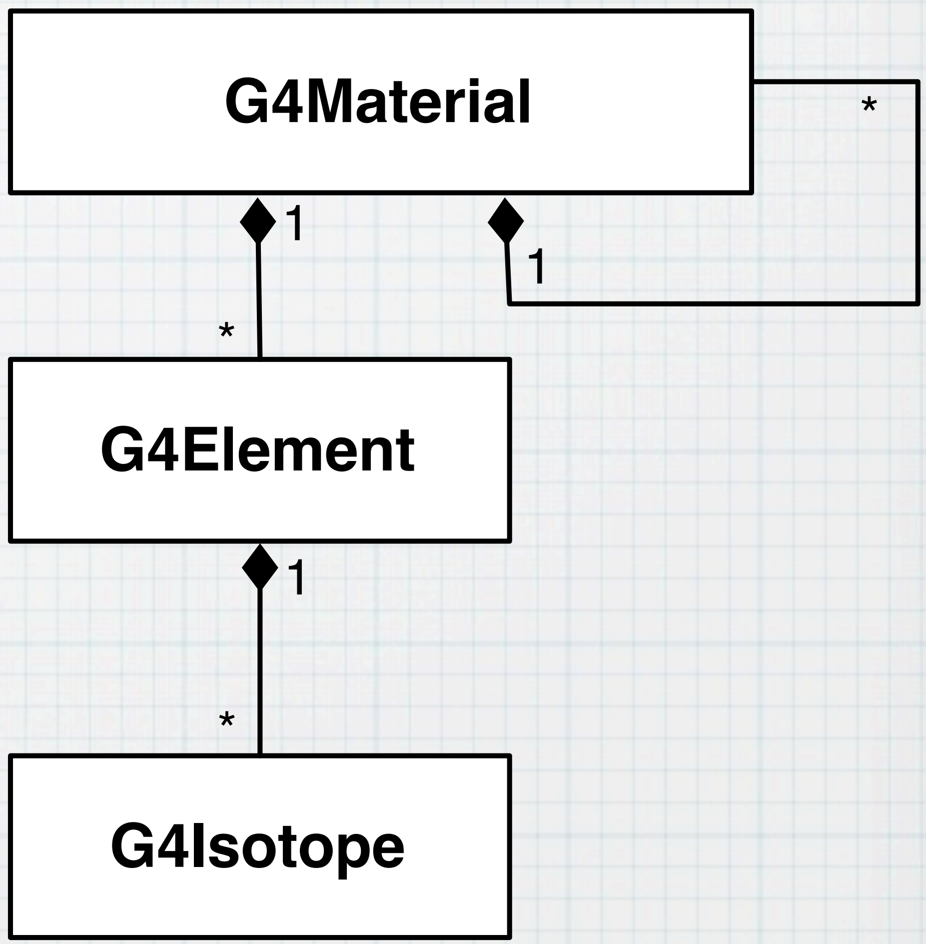
| N | Name | ChFormula | density(g/cm^3) | I(eV) |
|---|---|---|---|---|
| 13 | G4_Adipose_Tissue | | 0.92 | 63.2 |
| | 1 | 0.119477 | | |
| | 6 | 0.63724 | | |
| | 7 | 0.00797 | | |
| | 8 | 0.232333 | | |
| | 11 | 0.0005 | | |
| | 12 | 2e-05 | | |
| | 15 | 0.00016 | | |
| | 16 | 0.00073 | | |
| | 17 | 0.00119 | | |
| | 19 | 0.00032 | | |
| | 20 | 2e-05 | | |
| | 26 | 2e-05 | | |
| | 30 | 2e-05 | | |
| 4 | G4_Air | | 0.00120479 | 85.7 |
| | 6 | 0.000124 | | |
| | 7 | 0.755268 | | |
| | 8 | 0.231781 | | |
| | 18 | 0.012827 | | |
| 2 | G4_CsI | | 4.51 | 553.1 |
| | 53 | 0.47692 | | |
| | 55 | 0.52308 | | |

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

# Summary of Materials

for Reference

- 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

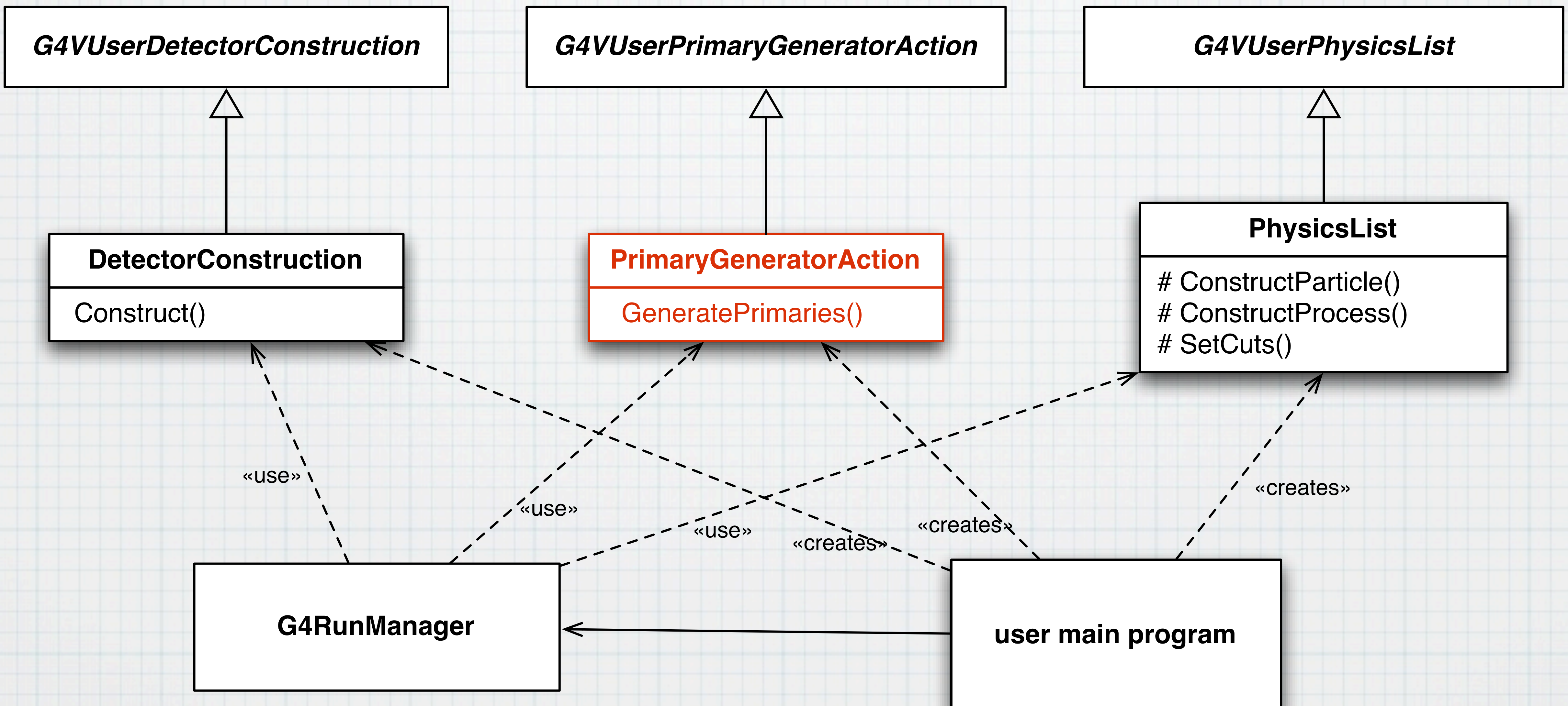
- Tutorial Material online:
  - http://www.ifh.de/geant4/g4course2010
- Exercise 1.1.4
  - change sensor material to `G4_GALLIUM_ARSENIDE`
  - create a customized material

# Primary Generator Action

*Particle Source*

# Mandatory User Classes

- User Action:
  - PrimaryGeneratorAction



# 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, G4HEPMCInterface - read in

# G4ParticleGun

for Reference

- 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 |
| – ... | |

# Custom PrimaryGeneratorAction
## for Reference

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

```
#include "G4GeneralParticleSource.hh"

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

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

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

- All user instructions given via macro UI commands

# G4GeneralParticleSource

- some UI commands:

```
# 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)