XIV Seminar on Software for Nuclear, Subnuclear and Applied Physics

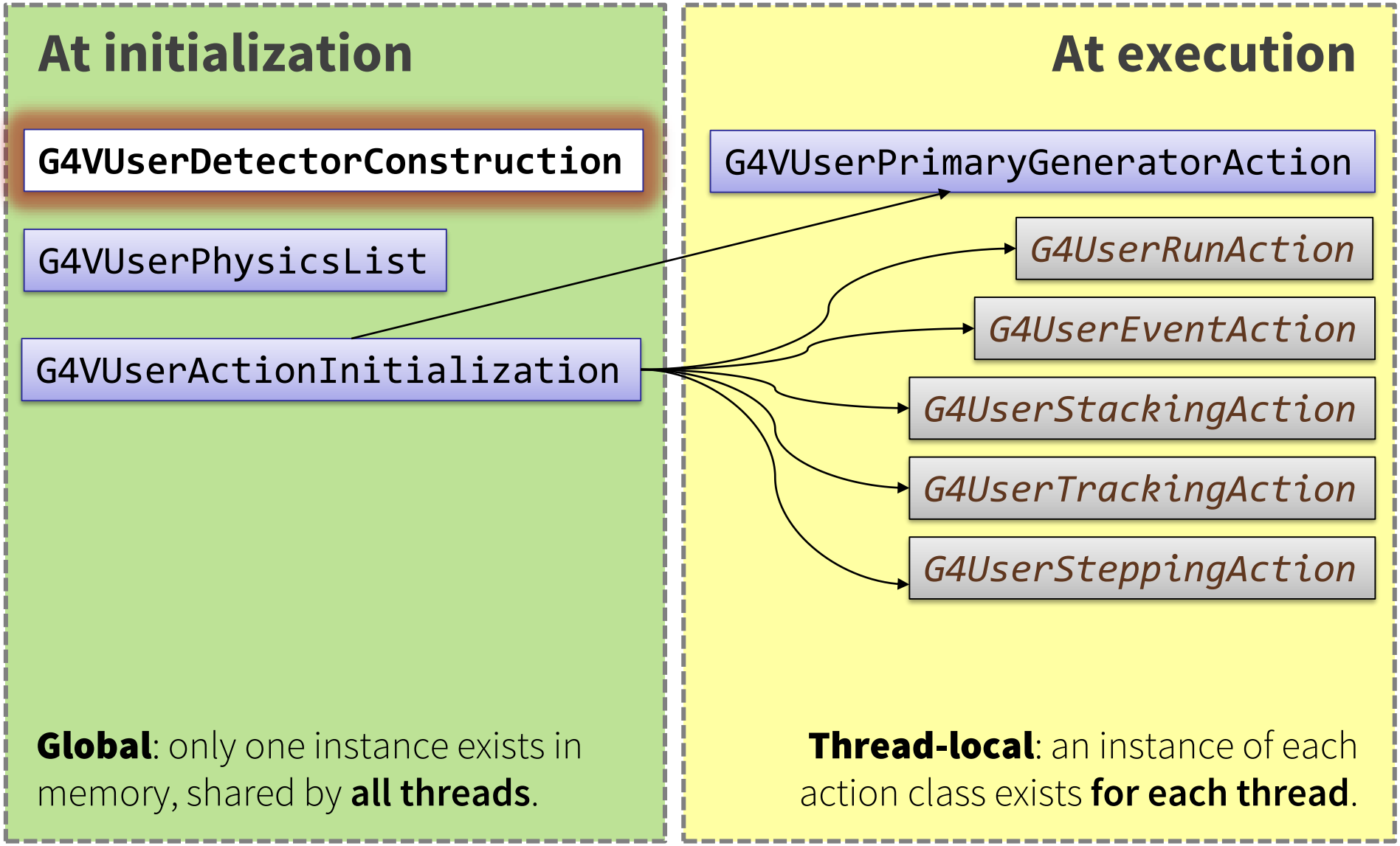
Alghero (ITALY)

04-09 June 2017

Detector description: materials and geometry

Geant4 tutorial

User classes (starting...)



Contents

- Units and materials
- Geometry construction

- Electromagnetic fields
- Repeated volumes
- CAD import and DICOM images

Note: **Geant4 basic types**

- Aliases for the **primitive data types** to provide cross-platform compatibility:
  - `G4double`, `G4float`, `G4int`, `G4bool`, `G4long`
- Enhanced version of string called **G4String**
  - inherits from std::string ⇒ all methods and operators
  - several additional methods
- **G4ThreeVector** is a three-component class corresponding to a real physics vector (example later)

```
G4ThreeVector dimensions {1.0, 2.0, 3.0 };
```

Please, use these types for best compatibility (e.g. G4int instead of int, etc., G4ThreeVector when it makes sense etc.)

Part I:
Units & materials

- System of units & constants
- Definition of elements
- Materials and mixtures
- NIST database

Units in Geant4

- **Don't use default units!**
  - When specifying dimensions, always* multiply by an appropriate unit:

```
G4double width = 12.5 * m;
G4double density = 2.7 * g/cm3;
```

  - Most common units are defined in **CLHEP** library (included in Geant4):

► G4SystemOfUnits.hh

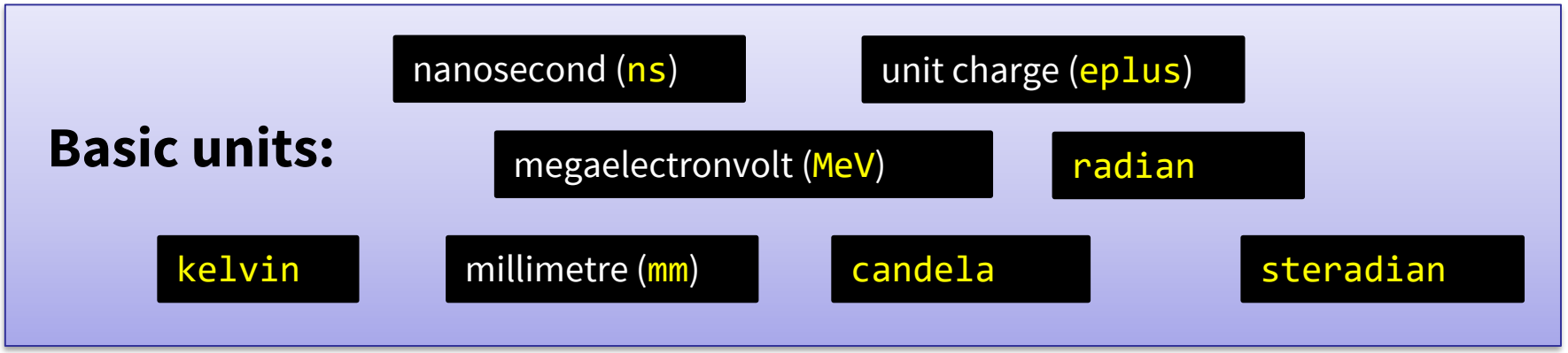
► CLHEP/SystemOfUnits.hh

  - You can define new units (see backup slides)

- Output data in terms of a specific unit:
  - **divide** a value by the unit:

```
G4cout << dE / MeV << " (MeV)" << G4endl;
```

System of Units



- All other units derived from the basic ones.

- **Useful feature:** Geant4 can select the most appropriate unit to use
  - specify the *category* for the data (**`Length`**, **`Time`**, **`Energy`**, etc…):

```
G4cout << G4BestUnit(StepSize, "Length");
```

`StepSize` will be printed in **`km`**, **`m`**, **`mm`** or … **`fermi`**, depending on its actual value

Materials

- Different levels of material description:
  - isotopes → `G4Isotope`
  - elements → `G4Element`
  - molecules → `G4Material`
  - compounds and mixtures → `G4Material`
- Attributes associated:
  - temperature, pressure, state, density
- `G4Isotope` and `G4Element` describe 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 used by tracking, geometry and physics

Elements and isotopes

Isotopes…

```
G4Isotope(const G4String& name,
                G4int     z,    // atomic number
                G4int     n,    // number of nucleons
                G4double  a );  // mass of mole
```

…can be assembled into elements as follows:

```
G4Element (const G4String& name,
           const G4String& symbol,   // element symbol
                 G4int     nIso );   // n. of isotopes
```

```
G4Element::AddIsotope(G4Isotope* iso,  // isotope
              G4double relAbund);      // fraction of atoms
```

Elements & compounds

Single-element material:

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

Molecule material (composition by number of atoms):

```
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, ncomponents=2);
H2O->AddElement(elH, natoms=2);
H2O->AddElement(elO, natoms=1);
```

Mixtures

Composition by fraction of mass:

```
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);
```

Composition of mixtures:

```
G4Element* elC  = …;   // define “carbon” element
G4Material* SiO2 = …;  // define “quartz” material
G4Material* H2O = …;   // define “water” material
density = 0.200*g/cm3;

G4Material* aerogel = new G4Material("Aerogel",
                                     density, ncomponents=3);
aerogel->AddMaterial(SiO2,fractionmass=62.5*perCent);
aerogel->AddMaterial(H2O, fractionmass=37.4*perCent);
aerogel->AddElement (elC, fractionmass= 0.1*perCent);
```

Example: **Gas**

- It may be necessary to specify temperature and pressure
  - (**dE/dx** computation affected)

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

G4Material* CO2 = new G4Material("CO2Gas", density,
    ncomponents=2, kStateGas, temperature, pressure);
CO2->AddElement(C, natoms = 1);
CO2->AddElement(O, natoms = 2);
```

- Absolute vacuum does not exist: gas at very low density!
  - Cannot define materials composed of multiple elements through **Z** or **A**, or with **ρ=0**

```
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 material database

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

```
G4NistManager* manager = G4NistManager::Instance();
G4Material* H2O =    manager->FindOrBuildMaterial("G4_WATER");
G4Material* air =    manager->FindOrBuildMaterial("G4_AIR");
G4Material* vacuum = manager->FindOrBuildMaterial("G4_Galactic");
```

- UI commands:

/material/nist/printElement ← print defined elements

/material/nist/listMaterials ← print defined materials

► G4NistManager.hh

NIST material database

- NIST database for materials is imported inside Geant4

  http://physics.nist.gov/PhysRefData
- 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

Z		A	m	error	(%)	A_{eff}
14	Si	22	22.03453		(22)	28.0855(3)
		23	23.02552		(21)	
		24	24.011546		(21)	
		25	25.004107		(11)	
		26	25.992330		(3)	
		27	26.98670476		(17)	
		28	27.9769265327		(20)	92.2297 (7)
		29	28.97649472		(3)	4.6832 (5)
		30	29.97377022		(5)	3.0872 (5)
		31	30.97536327		(7)	
		32	31.9741481		(23)	
		33	32.978001		(17)	
		34	33.978576		(15)	
		35	34.984580		(40)	
		36	35.98669		(11)	
		37	36.99300		(13)	
		38	37.99598		(29)	
		39	39.00230		(43)	
		40	40.00580		(54)	
		41	41.01270		(64)	
		42	42.01610		(75)	

- *Natural isotope compositions*
- *More than 3000 isotope masses*

http://geant4.cern.ch/UserDocumentation/UsersGuides/ForApplicationDeveloper/html/apas08.html

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 Elements:
  - **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

Part II: Geometry

- Basic concepts
- Implementation
  - Solids
  - Logical volumes
  - Physical volumes
  - Regions
- Tools for geometry checking

Geometry – basic implementations

- Implement a class inheriting from the abstract base class **G4VUserDetectorConstruction**:

```
class MyDetector : public G4VUserDetectorConstruction {
public:
    virtual G4VPhysicalVolume* Construct();       // required

    virtual void ConstructSDAndField();           // optional
    // ...
};
```

- Create an instance in the main program:

```
MyDetector* detector = new MyDetector();
runManager->SetUserInitialization(detector);
```

Note: Split the implementation into more classes & methods! (good programming practice)

  - especially for complex geometries!

Note: you should not delete the MyDetector instance! Run manager does that automatically.

G4VUserDetectorConstruction

- Method **Construct()**
  - Define materials
  - Define solids and volumes of the geometry
  - Build the tree hierarchy of volumes
  - Define visualization attributes

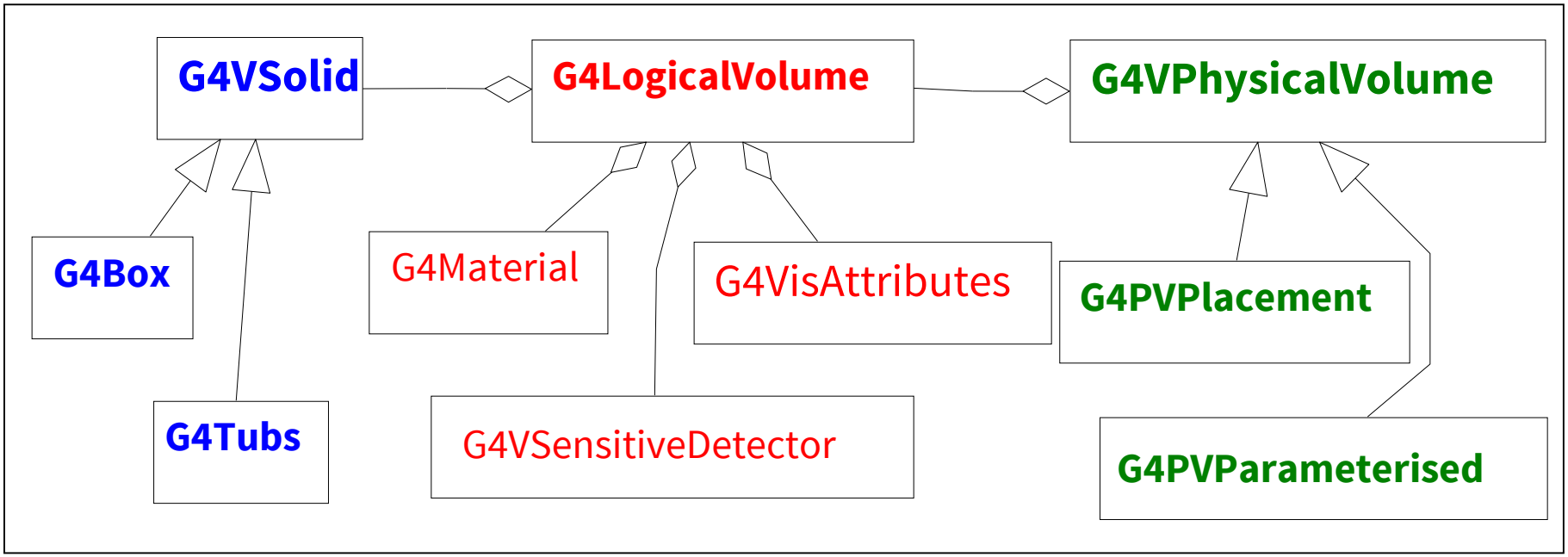
  - Return the world physical volume!

- Method **ConstructSDAndField()** MT
  - Assign magnetic field to volumes / regions
  - Define sensitive detectors and assing them to volumes

► `G4VUserDetectorConstruction.hh`

Detector geometry components

- Three conceptual layers
  - **G4VSolid**: *shape, size*
  - **G4LogicalVolume:** *material, sensitivity, magnetic field, etc.*
  - **G4VPhysicalVolume:** *position, rotation*

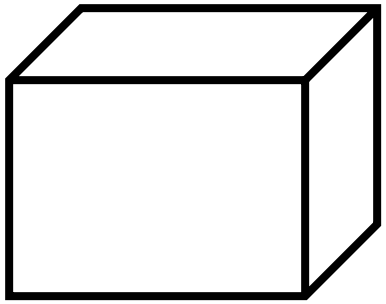


Define detector geometry

- Basic strategy

```
G4VSolid* pBoxSolid =
  new G4Box("aBoxSolid",
    1.*m, 2.*m, 3.*m);
```

Solid：shape and size



Step 1
Create the geom. object：box

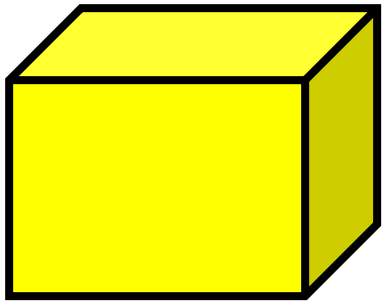
Define detector geometry

- Basic strategy

```
G4VSolid* pBoxSolid =
  new G4Box("aBoxSolid",
    1.*m, 2.*m, 3.*m);
```

```
G4LogicalVolume* pBoxLog =
  new G4LogicalVolume( pBoxSolid,
    pBoxMaterial, "aBoxLog", 0, 0, 0);
```

Logical volume :
+ material, sensitivity, etc.



Step 1
Create the geom. object : box

Step 2
Assign properties to object : material

Define detector geometry

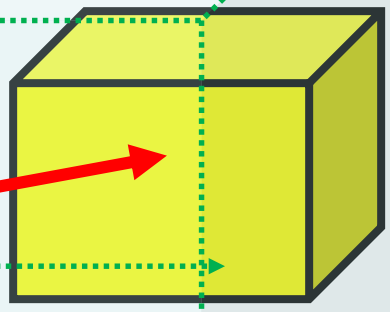
- Basic strategy

```
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);
```

Physical volume :
+ rotation and position



Step 1
Create the geom. object : box

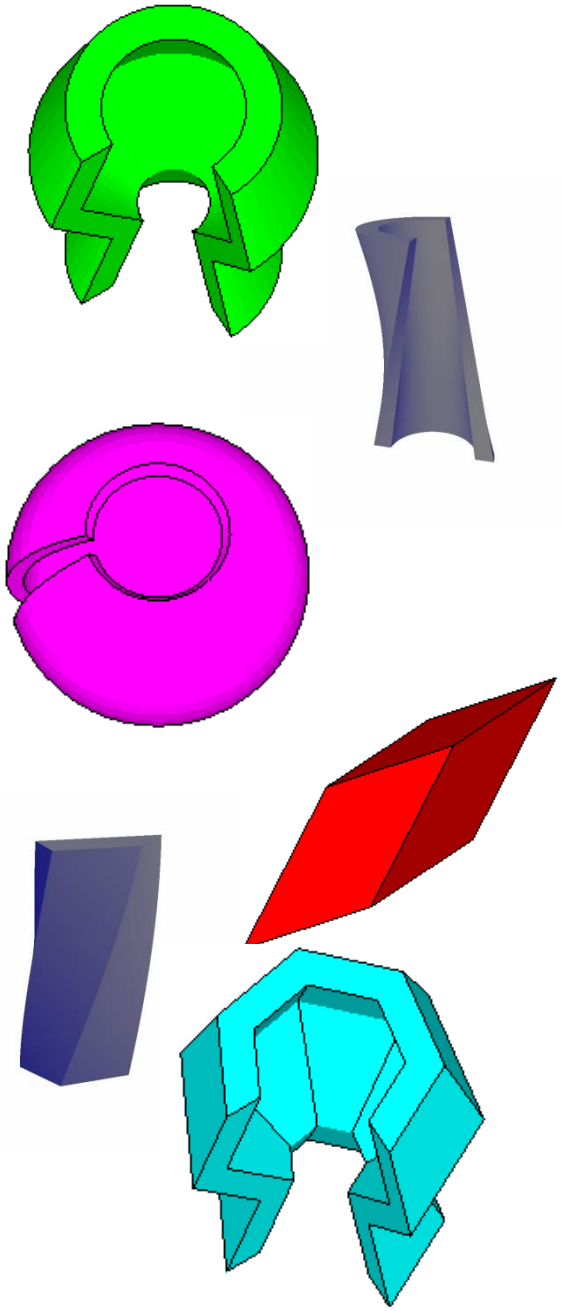
Step 2
Assign properties to object : material

Step 3
Place it in the coordinate system of mother volume

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

Solids

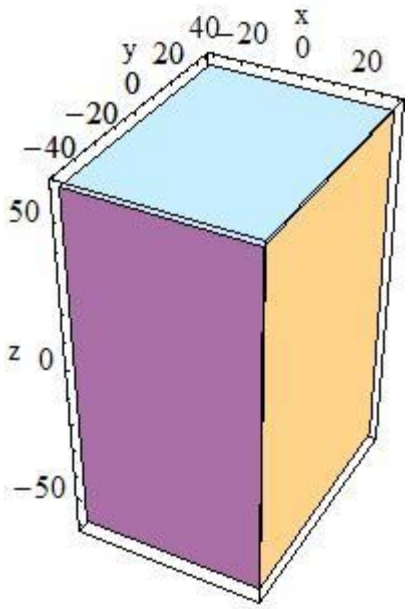
- Solids defined in Geant4:
  - CSG (Constructed Solid Geometry) solids
    - G4Box, G4Tubs, G4Cons, G4Trd, …
  - Specific solids (CSG like)
    - G4Polycone, G4Polyhedra, G4Hype, …
    - G4TwistedTubs, G4TwistedTrap, …
  - BREP (Boundary REPresented) solids
    - G4BREPSolidPolycone, G4BSplineSurface
    - Any order surface
  - Boolean solids
    - G4UnionSolid, G4SubtractionSolid, …



CSG: G4Box

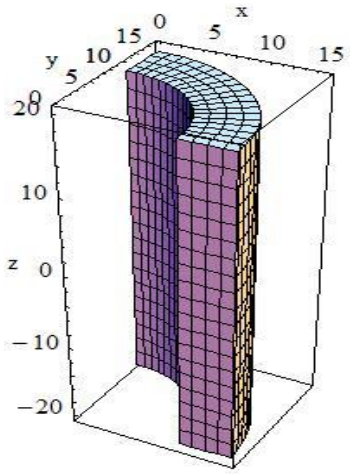
```
G4Box(const G4String& pname,  // name
                G4double  pX, // half-length in X
                G4double  pY, // half-length in Y
                G4double  pZ, // half-length in Z);
```

Note the half-length!

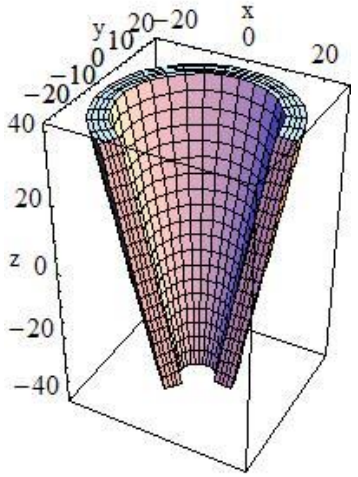


CSG: G4Tubs, G4Cons

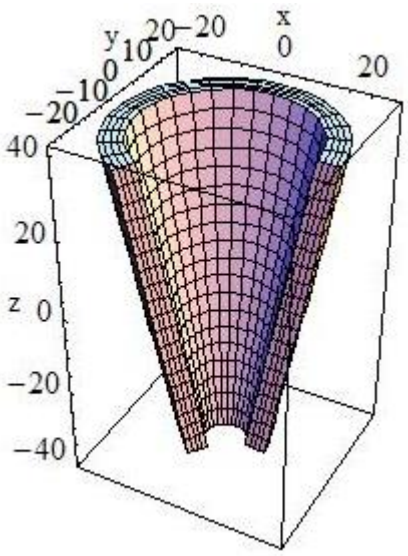
```
G4Tubs(const G4String& pname,   // name
             G4double  pRmin,   // inner radius (0)
             G4double  pRmax,   // outer radius
             G4double  pDz,     // Z half! length
             G4double  pSphi,   // starting Phi (0)
             G4double  pDphi); // segment angle (twopi)
```



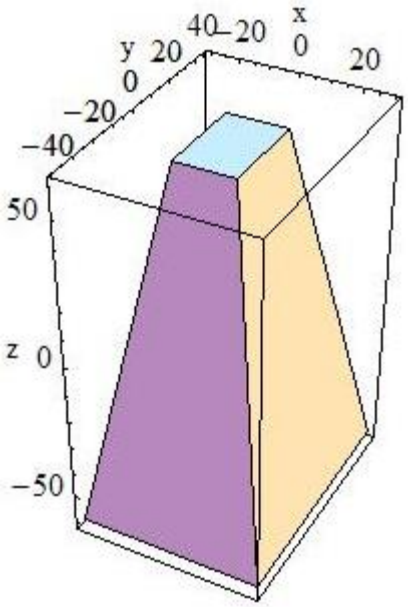

```
G4Cons(const G4String& pname,   // name
             G4double  pRmin1, // inner radius -pDz
             G4double  pRmax1, // outer radius -pDz
             G4double  pRmin2, // inner radius +pDz
             G4double  pRmax2, // outer radius +pDz
             G4double  pDz,     // Z half length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



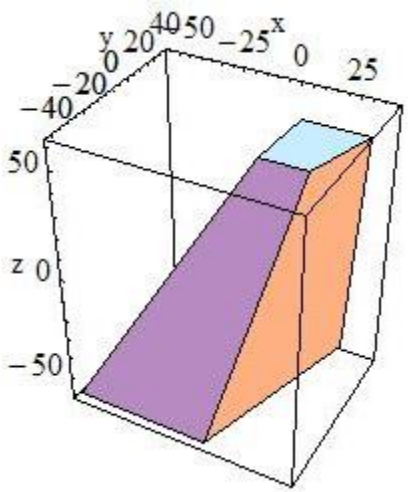
Other CSG solids



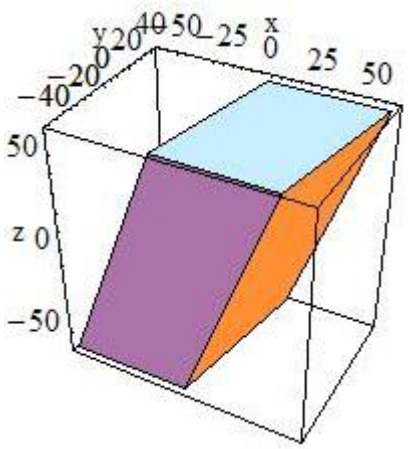
G4Cons



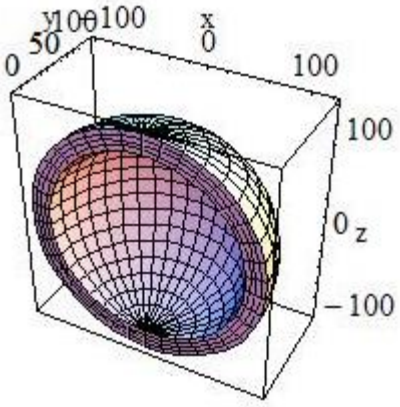
G4Trd



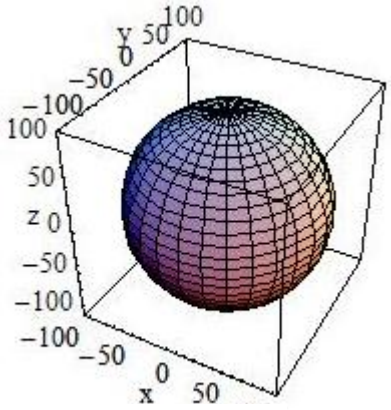
G4Trap



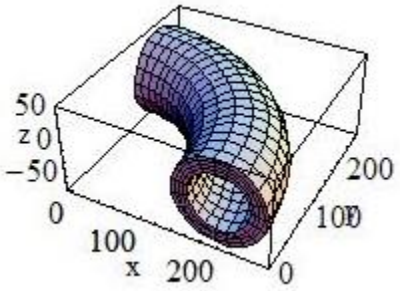
G4Para
(parallelepiped)



G4Sphere



G4Orb
(full solid sphere)



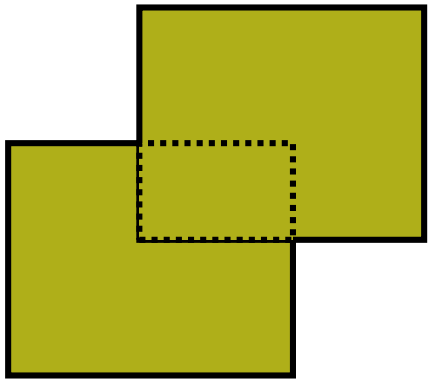
G4Torus

Consult Section 4.1.2 of Geant4 Application Developers Guide for all available shapes.

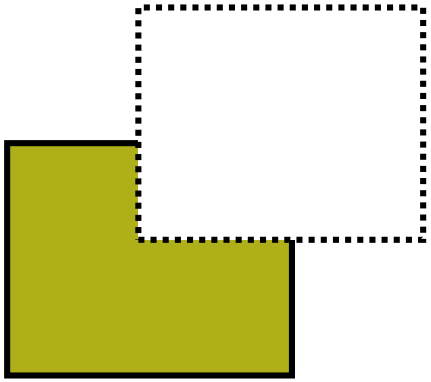
Boolean Solids

- Solids can be combined using boolean operations:
  - `G4UnionSolid, G4SubtractionSolid, G4IntersectionSolid`
  - Requires: 2 solids, 1 boolean operation, and an (optional) transformation for the 2nd solid
  - 2nd solid is positioned relative to the coordinate system of the 1st solid
  - Result of boolean operation becomes a solid. Thus a third solid can be combined with the resulting solid of first operation.
- Solids to be combined 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 CSG solids

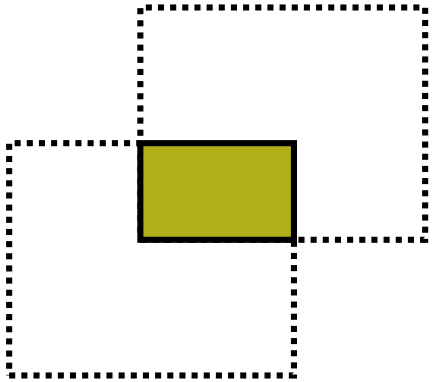
`G4UnionSolid`



`G4SubtractionSolid`



`G4IntersectionSolid`



Boolean Solids – example

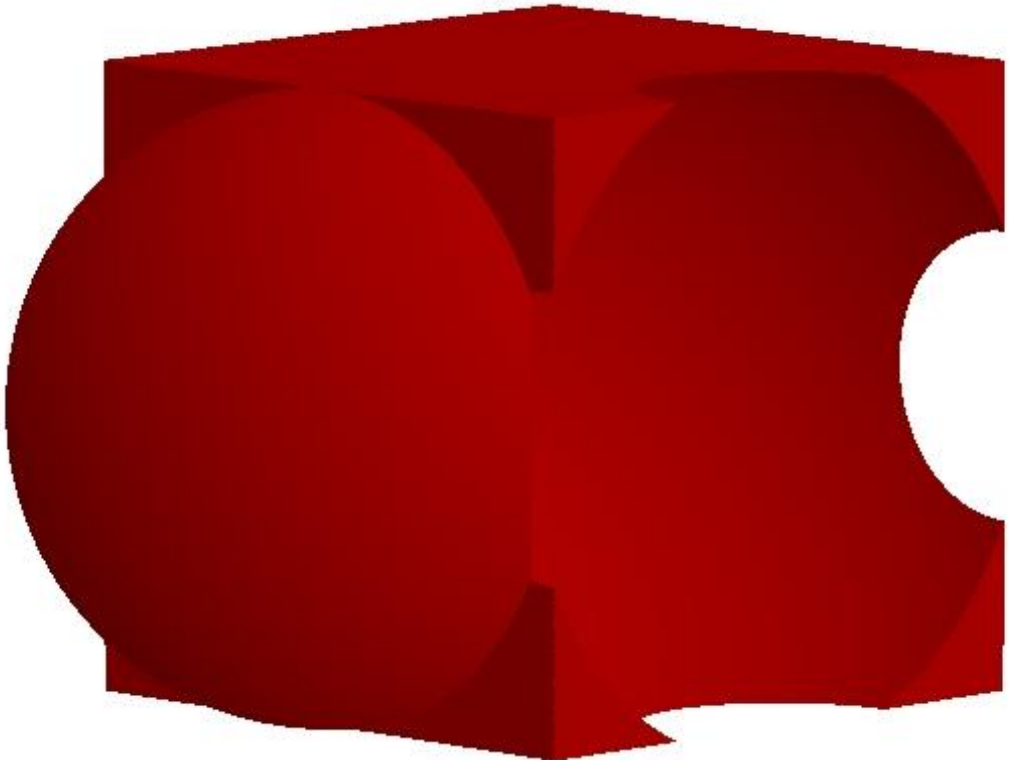
```
G4VSolid* box = new G4Box("Box",50*cm,60*cm,40*cm);
G4VSolid* cylinder =
    new G4Tubs("Cylinder",0.,50.*cm,50.*cm,0.,2*M_PI*rad);

G4VSolid* union =
    new G4UnionSolid("Box+Cylinder", box, cylinder);

G4VSolid* subtract =
    new G4SubtractionSolid("Box-Cylinder", box, cylinder,
        0, G4ThreeVector(30.*cm,0.,0.));

G4RotationMatrix* rm = new G4RotationMatrix();
rm->RotateX(30.*deg);
G4VSolid* intersect =
    new G4IntersectionSolid("Box&&Cylinder",
        box, cylinder, rm, G4ThreeVector(0.,0.,0.));
```

Boolean Solids



Alternative: Unified solids (Geant4 10.0)

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, *see above*)
  - Material, sensitivity, visualization attributes
  - Position of daughter volumes
  - Magnetic field, User limits

- Physical volumes of same type can share a logical volume.
- The pointers to solid and material must not be null!

Visualization attributes

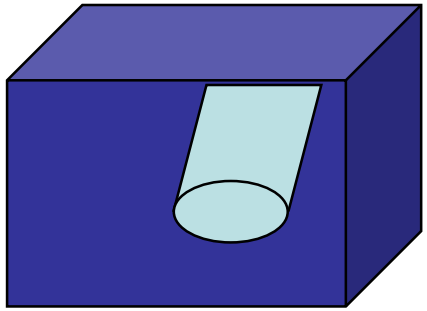
- Instance of **G4VisAttributes** class assigned to a logical volume
  - visible / invisible
  - colour as G4Colour instance
  - solid / wireframe

```
// ..
G4LogicalVolume* myLog = new G4LogicalVolume(...);
G4VisAttributes* red = new G4VisAttributes(G4Colour::Red());
red->SetVisibility(true);
red->SetForceSolid(true);
myLog->SetVisAttributes(red);
//...
G4VisAttributes* blue = new G4VisAttributes(G4Colour::Blue());
blue->SetVisibility(true);
blue->SetForceWireframe(true);
myLog->SetVisAttributes(blue);
```

Physical Volumes

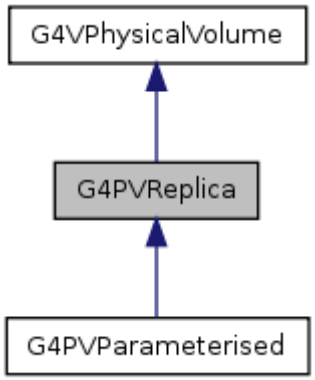
A physical volume is a positioned instance of a logical volume inside a another logical volume (the mother volume).

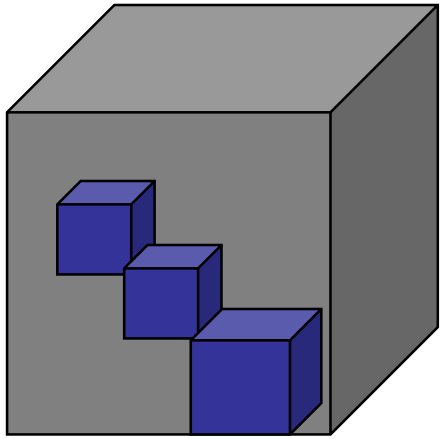
- G4PVPlacement: a single positioned volume
  Note that a Placement Volume can still represent multiple detector elements, if several copies exist of the mother logical volume



G4PVPlacement

- Repeated Volumes: a volume placed many times within its mother volume
  - Reduce use of memory
  - G4PVReplica (= simple repetition)
  - G4PVParameterised (= more complex)





G4PVParameterised

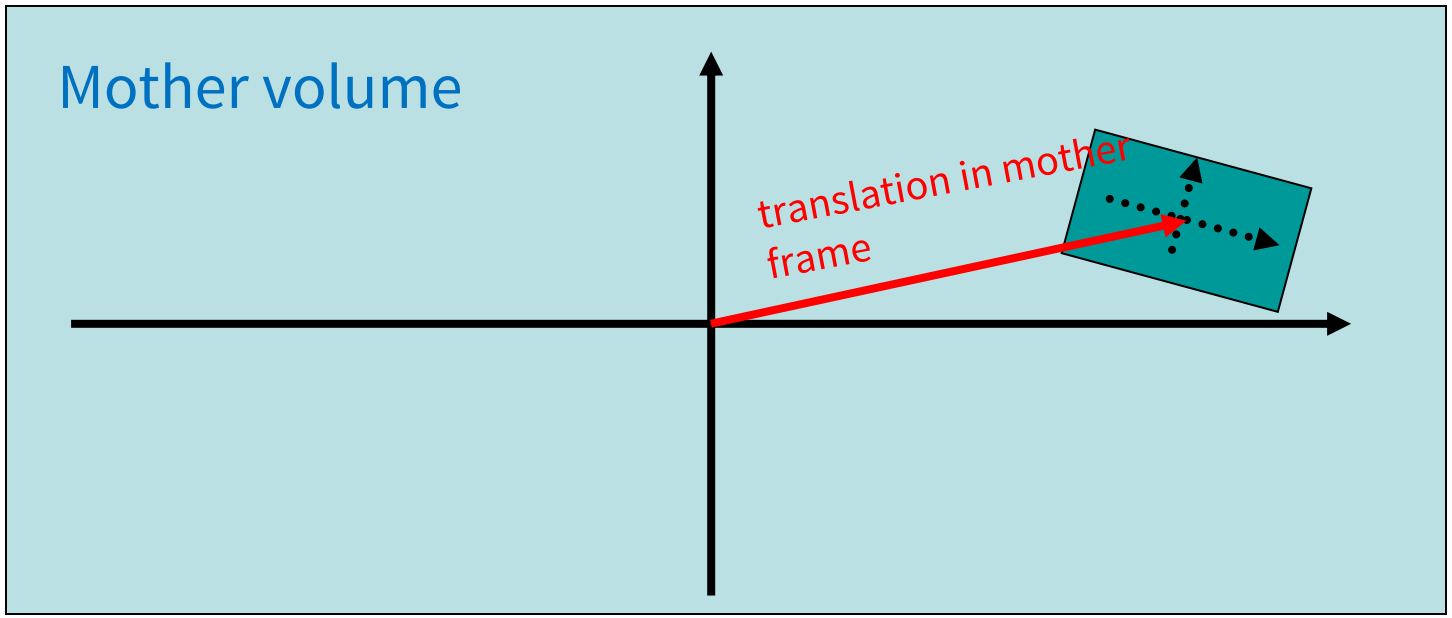
G4PVPlacement

- Single volume positioned relatively to the mother volume
  - In a frame rotated and translated relative to the coordinate system of the mother volume
- Four constructors:
  - **logical** OR **physical** volume as mother
  - **active** OR **passive** transformation of the coordinate system

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
```

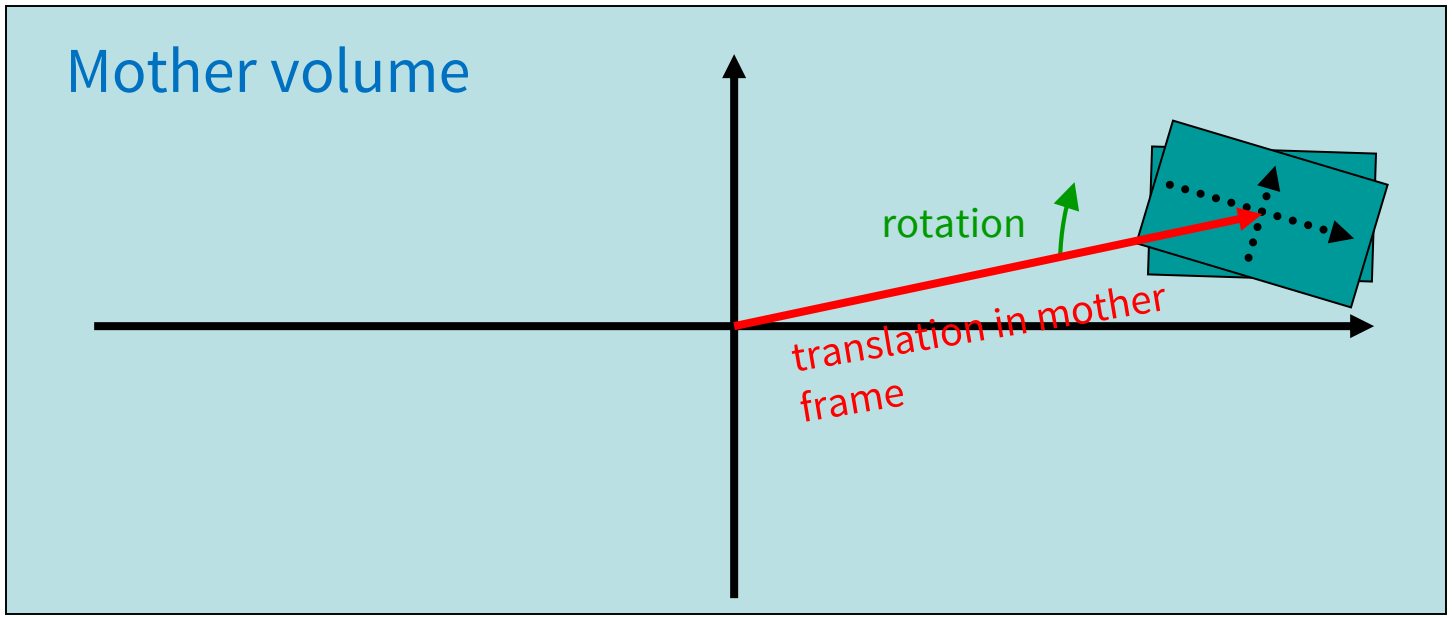
- Single volume positioned relatively to the mother volume (passive transformation)



G4PVPlacement
Rotation in mother frame …

```
G4PVPlacement(G4Transform3D(
                    G4RotationMatrix &pRot,         // rotation in 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
```

- Single volume positioned relatively to the mother volume (active transformation)

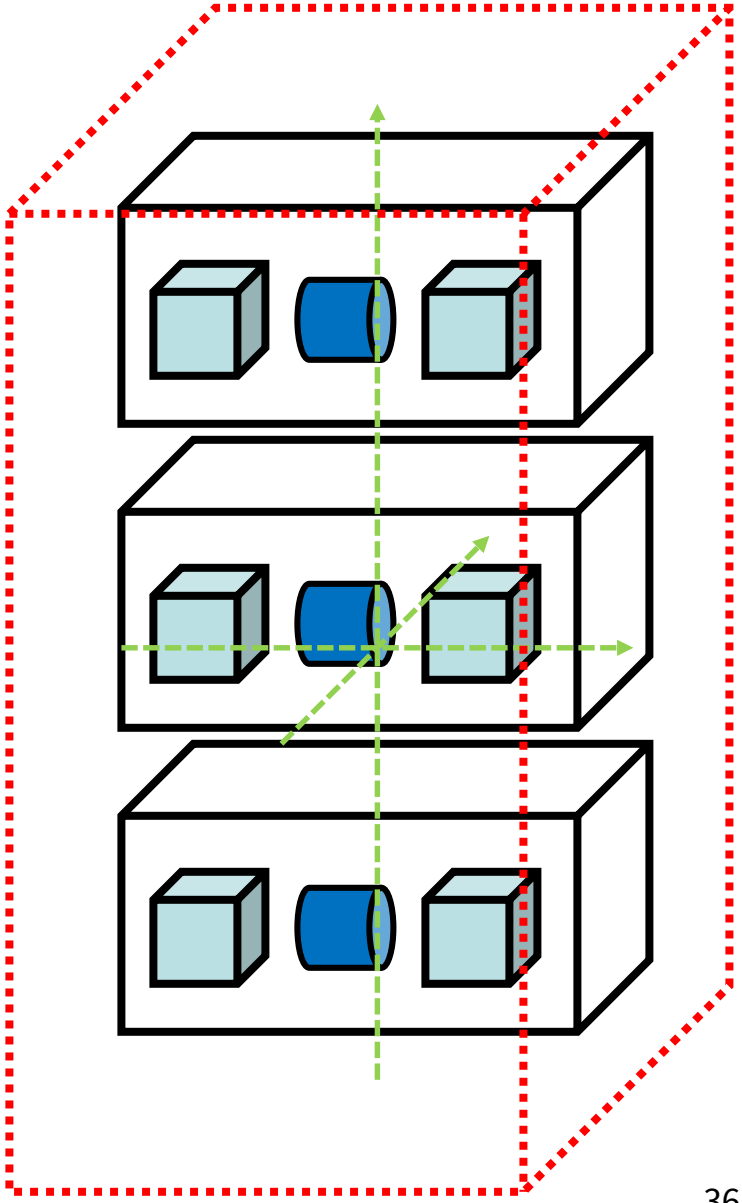


Geometrical hierarchy

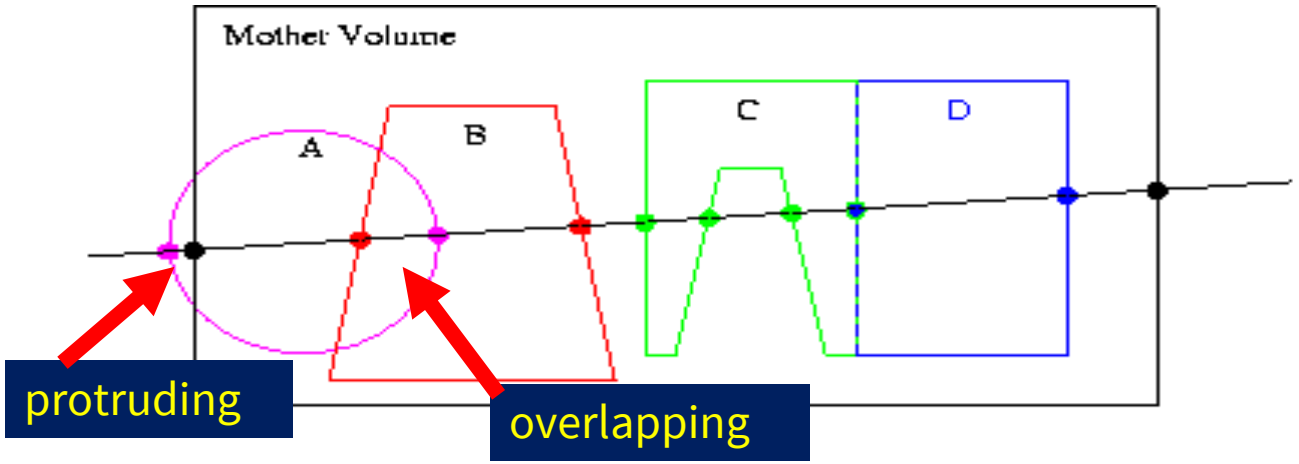
- It forms a **tree.**
- There is just one physical **world volume** forming a root of the tree:
  - The world volume defines the **global coordinate system**. The origin of the global coordinate system is at the **center of the world volume**
  - Position of a track is given with respect to this coordinate system
  - It **contains all other volumes** (with no protruding!)

- **One logical volume** can be placed **more than once**.
- **One or more volumes** can be placed **in one mother volume**.
- There must be **no overlapping** or protrusion.
- `G4LogicalVolume` carries the information about daughter volumes

  => if it is placed more than once, all daughters by definition appear in each physical instance



Problems in geometry



- Geant4 does not allow malformed geometries, neither protruding nor overlapping.
  - The behavior of navigation is unpredictable for such cases!
- Overlaps become an important issue in complex geometries.
- There are tools for detecting wrong positioning:
  - Optional checks at construction
  - Kernel run-time commands
  - Graphical tools (DAVID, OLAP)

Tools for geometry checking

- Constructors of G4PVPlacement and G4PVParameterised have an optional argument "pSurfChk".

```
G4PVPlacement(G4RotationMatrix* pRot,
    const G4ThreeVector &tlate,
    G4LogicalVolume *pDaughterLogical,
    const G4String &pName,
    G4LogicalVolume *pMotherLogical,
    G4bool pMany, G4int pCopyNo,
    G4bool pSurfChk=false);
```

If this flag is true, overlap check is done at the construction:

- some number of points are randomly sampled on the surface of creating volume.

This check requires lots of CPU time, but it is worth to try at least once.

- Built-in run-time commands to activate verification tests for the user geometry:

/geometry/test/run or **/geometry/test/grid_test**

to start verification of geometry for overlapping regions based on a standard grid setup, limited to the first depth level

/geometry/test/recursive_test

applies the grid test to all depth levels (may require lot of CPU time!)

/geometry/test/line_test

to shoot a line along a specified direction and position

Tools for geometry checking

Source

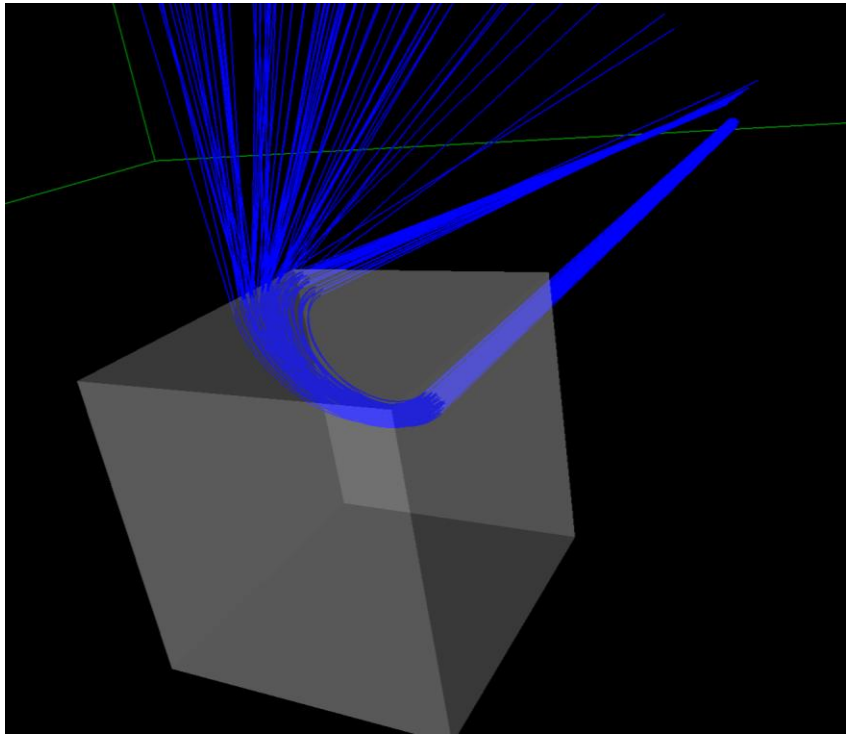
```
void MyDetectorConstruction::CheckOverlaps()
{
  G4PhysicalVolumeStore* thePVStore = G4PhysicalVolumeStore::GetInstance();
  G4cout << thePVStore->size() << " physical volumes are defined" << G4endl;
  G4bool overlapFlag = false;
  G4int res = 1000;
  G4double tol = 0.;        // tolerance
  for (size_t i=0; i<thePVStore->size(); i++)
  {
      overlapFlag = (*thePVStore)[i]->CheckOverlaps(res,tol,fCheckOverlapsVerbosity)
          | overlapFlag;
  }
  if (overlapFlag)
      G4cout << "Check: there are overlapping volumes" << G4endl;
}
```

Output

```
Checking overlaps for volume BeamLineSupport ... OK!
Checking overlaps for volume BeamLineCover ... OK!
Checking overlaps for volume BeamLineCover2 ... OK!
Checking overlaps for volume VacuumZone ... OK!
Checking overlaps for volume FirstScatteringFoil ... OK!
……
-------- WWWW ------- G4Exception-START -------- WWWW -
*** G4Exception : GeomVol1002    issued by : G4PVPlacement::CheckOverlaps()
Overlap with volume already placed !
          Overlap is detected for volume BrassTube2
          with HoleNozzleSupport volume's
          local point (12.6381,12.8171,-25.1867), overlapping by at least: 3.5 mm
*** This is just a warning message. ***
-------- WWWW -------
…
```

This method can be called at any point after run->Initialize();

EM fields

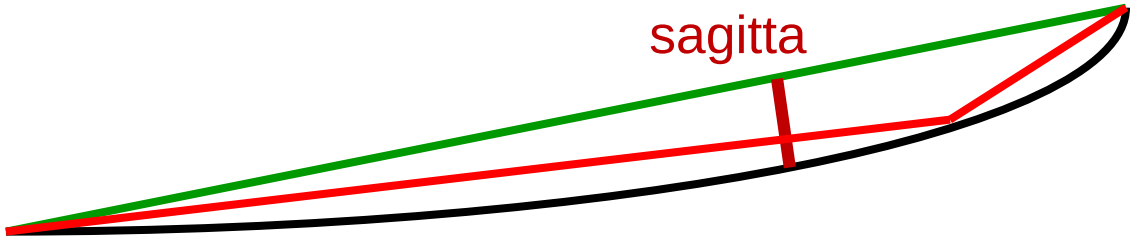


EM fields – Tracking in fields

- Divide the trajectory of the particle in **"steps"**
  - Straight free-flight tracks between consecutive physics interactions
- In presence of EM fields, the free-flight part between interactions is **not straight**
  - Change of *direction* (B-field) or *energy* (E-field)
  - Effect of fields must be incorporated into the tracking algorithm → **CPU-demanding**
- Notice: most codes handle only **weak fields**
  - An e^- at rest will not accelerate, no syncrothron radiation, no avalanche

EM fields – Tracking in fields

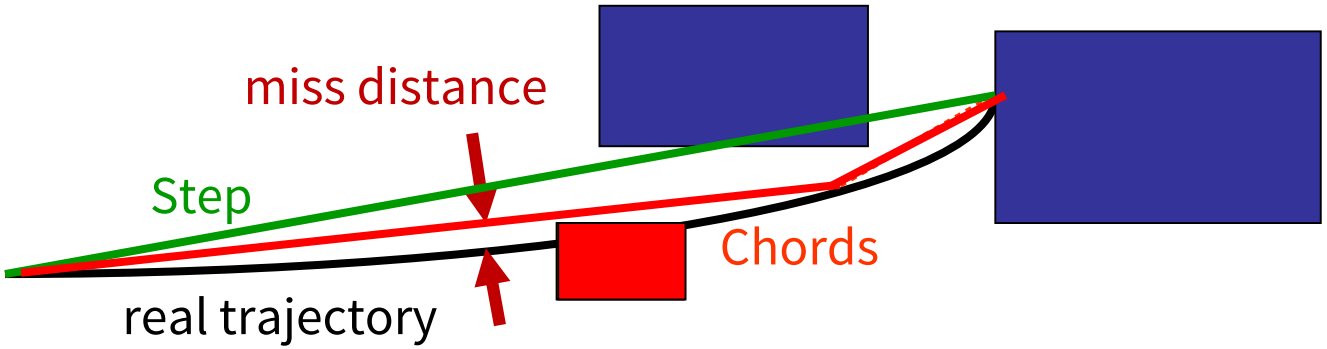
- In order to propagate a particle inside a field the equation of motion of the particle in the field is integrated numerically
- In general this is best done using a Runge-Kutta (RK) method for the integration of ordinary differential equations
  - Other methods are also available
- Once the curved path is calculated, Geant4 breaks it up into **linear chord segments**



The chord segments are determined to closely approximate the curved path

EM fields – Customization

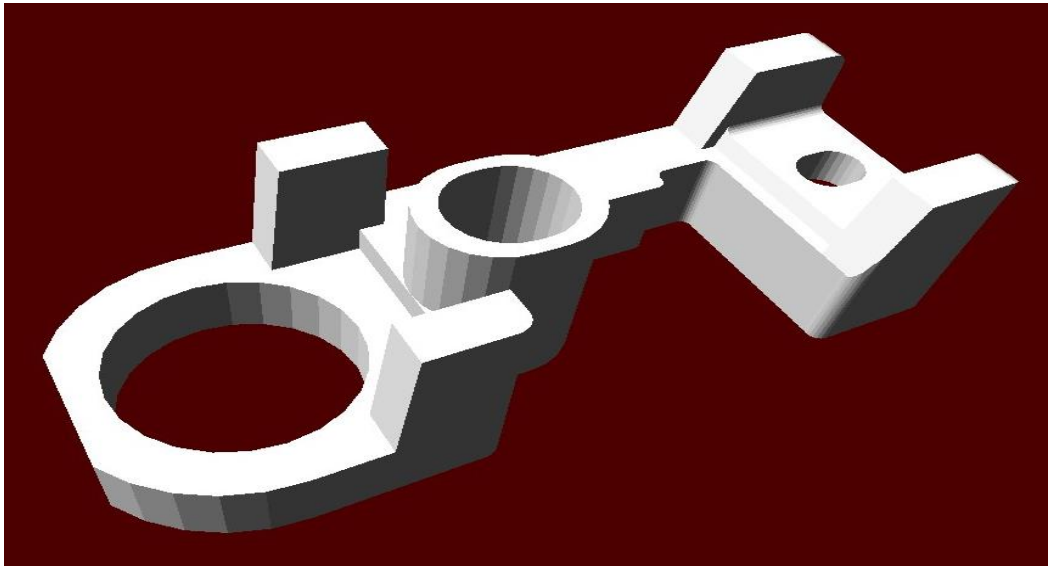
- A few parameters to customize the precision of the tracking in EM fields. Most critical: "miss distance"
  - Upper bound for the value of the sagitta (default: 3 mm)
  - May be highly CPU consuming



- Integration calculated by 4th-order Runge-Kutta (**`G4ClassicalRK4`**), robust and general purpose
  - If the field is not smooth (e.g. field map), lower-order (and faster) integrators can be appropriate
  - 3rd order **`G4SimpleHeum`**, 2nd order **`G4ImplicitEuler`**, 1st order **`G4ExplicitEuler`**

CAD import

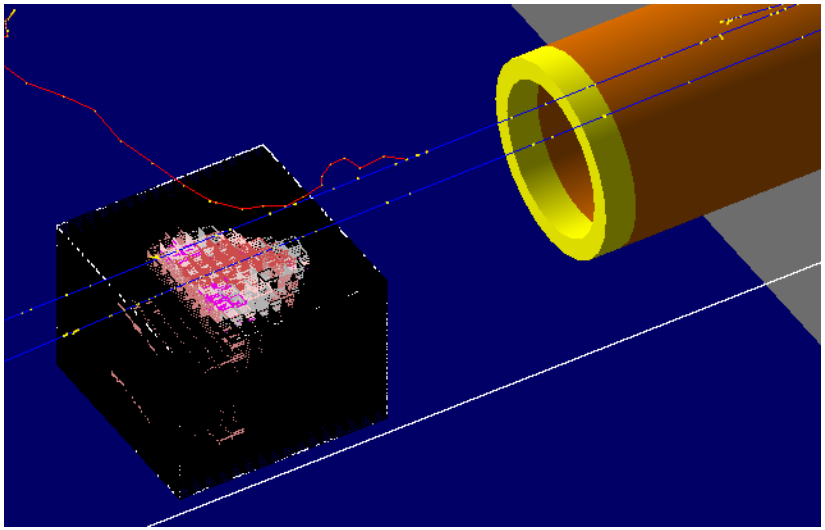
- Typical request: import CAD technical drawings as Geant4 geometries
- Difficulties:
  - Proprietary, undocumented or changing CAD formats
  - Usually no connection between geometry and materials
  - CAD is never as easy as you might think
- Possible solution (a lot of work!)
  - Convert **CAD into STEP** (no material information)
  - Convert **STEP into GDML** and restore manually material information
    - Needs special software (ST-viewer, FastRAD)

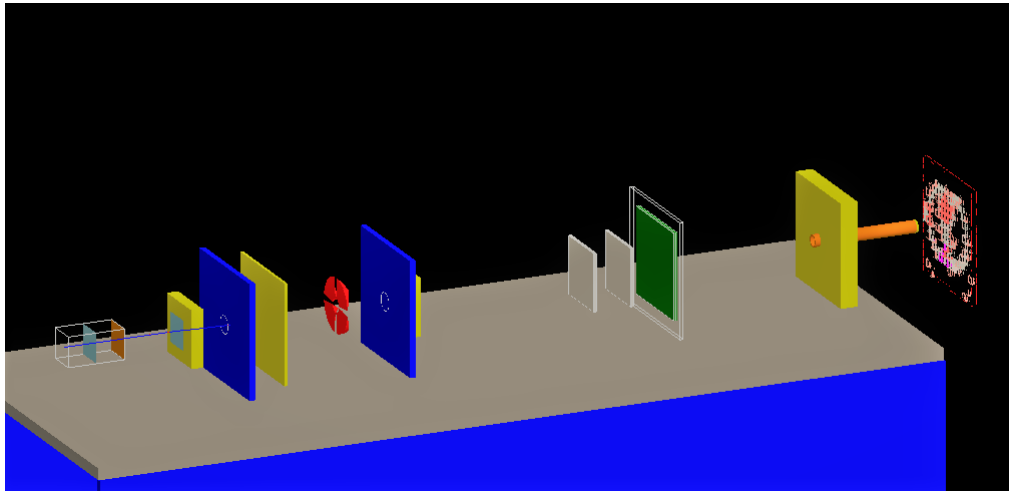


DICOM import

It is possible to import every kind of DICOM images.

- Convert **HU** into **physical density**;
- Associate **physical density** to **tissue or organs composition**.





```
#########################################################
#     Density Range                       Material      #
#-------------------------------------------------------#
#       mg/cm3                                  -       #
#-------------------------------------------------------#
#  [ 0.     , 0.207 )                  Air               #
#  [ 0.207  , 0.481 )                  Lungs (inhale)    #
#  [ 0.481  , 0.919 )                  Lungs (exhale)    #
#  [ 0.919  , 0.979 )                  Adipose           #
#  [ 0.979  , 1.004 )                  Breast            #
#  [ 1.004  , 1.043 )                  Phantom           #
#  [ 1.043  , 1.109 )                  Liver             #
#  [ 1.109  , 1.113 )                  Muscle            #
#  [ 1.113  , 1.496 )                  Trabecular Bone   #
#  [ 1.496  , 1.654 ]                  Dense Bone        #
#########################################################
```

Conclusion

Geant4 has very broad range of capabilities for geometry description:

- extensible set of **units**
- complex hierarchy of **volumes**
- many types of **solids** and their combinations
- **material** description down to isotope level
- **CAD** & **DICOM** importing geometry