

VIII Seminario sul Software per la Fisica Nucleare, Subnucleare e Applicata

Hotel Porto Conte, Alghero
6-10 June 2011

Detector description: materials and geometry

Geant4 tutorial course



Part I: Materials

Materials:

- The System of units & constants
- Definition of elements
- Materials and mixtures
- Some examples ...
- The NIST Data Base

Units

- Geant4 has no default unit. To give a number, unit must be “multiplied” to the number.
 - for example :

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

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...):

```
G4cout << G4BestUnit(StepSize, "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` ("grampercm2", "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`

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
- **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 the class used for geometry definition

Materials of one element and molecules

- 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):

```
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);
```

Compound and mixture

- Compound: 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 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
 - (dE/dx computation affected)

```
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);
```

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

```
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); 9
```

NIST Material Data-Base in Geant4

- 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

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*

NIST materials in Geant4

```

=====
### 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_Csl 4.51 553.1
53 0.47692
55 0.52308

```

- 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:

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

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

- Some UI commands ...

```
/material/nist/printElement ← print defined elements
```

```
/material/nist/listMaterials ← print defined materials
```

Part II: Geometry

Geometry:

- Detector description: the basic
 - Detector geometry components
 - Define detector geometry
- Describing a detector
 - Solids
 - Logical volumes
 - Physical volumes
- Tools for geometry check

Describe your detector

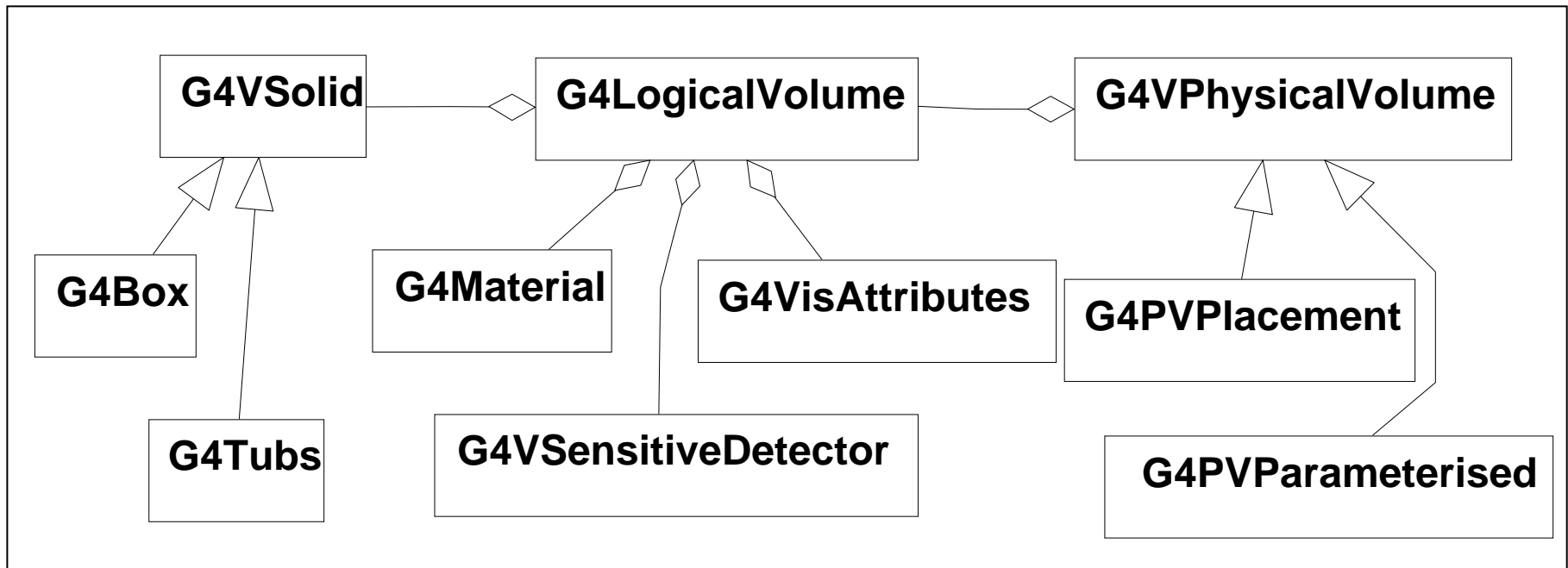
- Derive your own concrete class from `G4VUserDetectorConstruction` abstract base class.
- Implementing the method `construct()`:
 - Modularize it according to each detector component or sub-detector:
 - Construct all necessary materials
 - Define shapes/solids required to describe the geometry
 - Construct and place volumes of your detector geometry
 - Define sensitive detectors and identify detector volumes which to associate them
 - Associate magnetic field to detector regions
 - Define visualization attributes for the detector elements

Detector geometry components

- 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*

Detector geometry components

- Three conceptual layers
 - **G4VSolid** -- *shape, size*
 - **G4LogicalVolume** -- daughter physical volumes,
material, sensitivity, user limits, etc.
 - **G4VPhysicalVolume** -- *position, rotation*

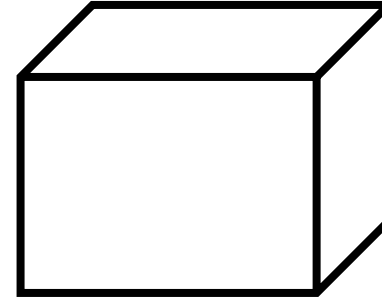


Define detector geometry

- Basic strategy

Solid : shape and size

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



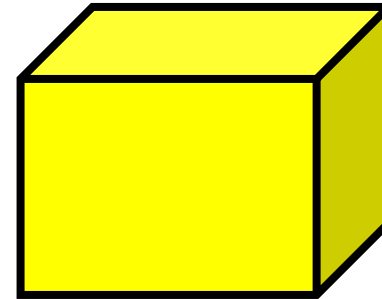
Step 1
Create the
geom. object :
box

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

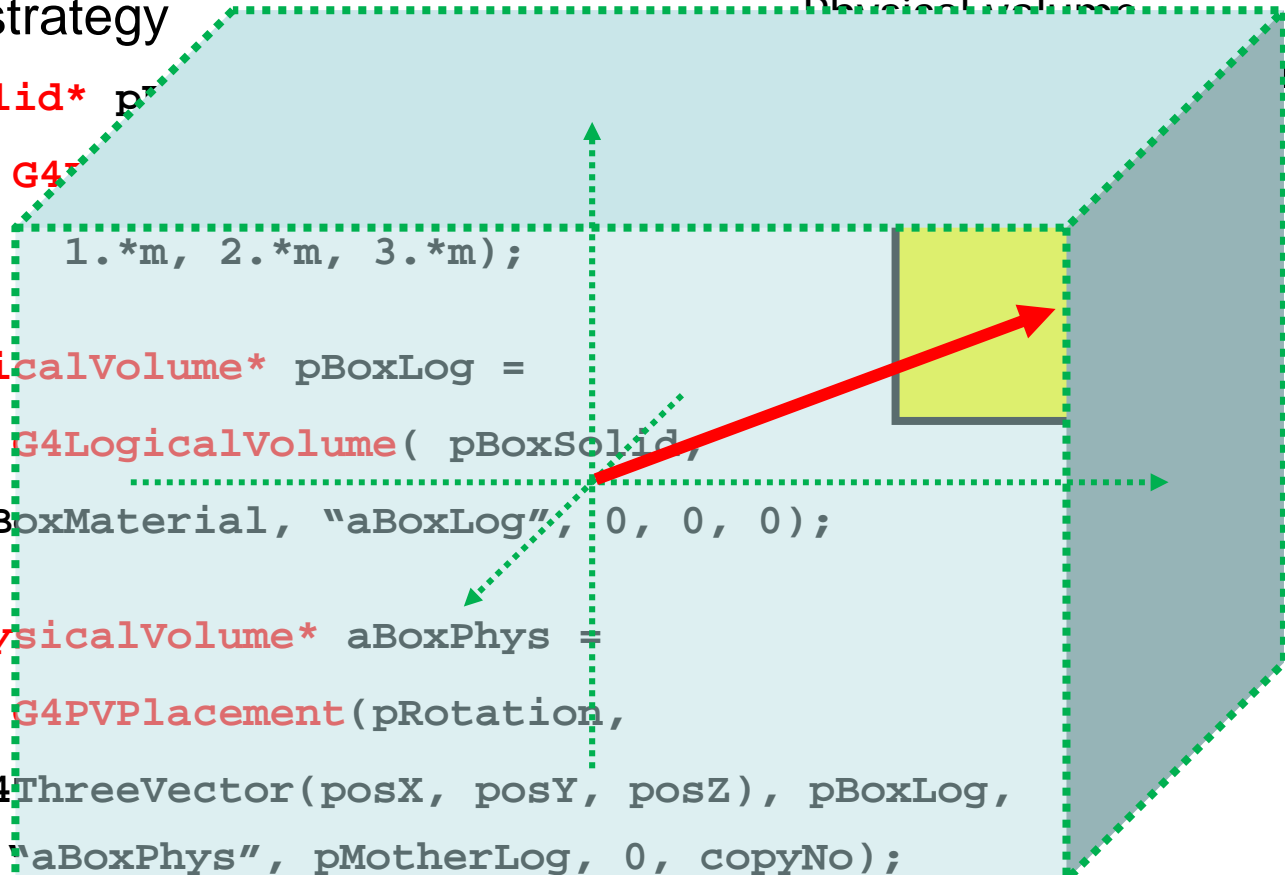


Step 2
Assign properties
to object :
material

Define detector geometry

- Basic strategy

```
G4VSolid* pBoxSolid =  
new G4Box(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);
```



Step 1
Create the
geom. object :
box

Step 2
Assign properties
to object :
material

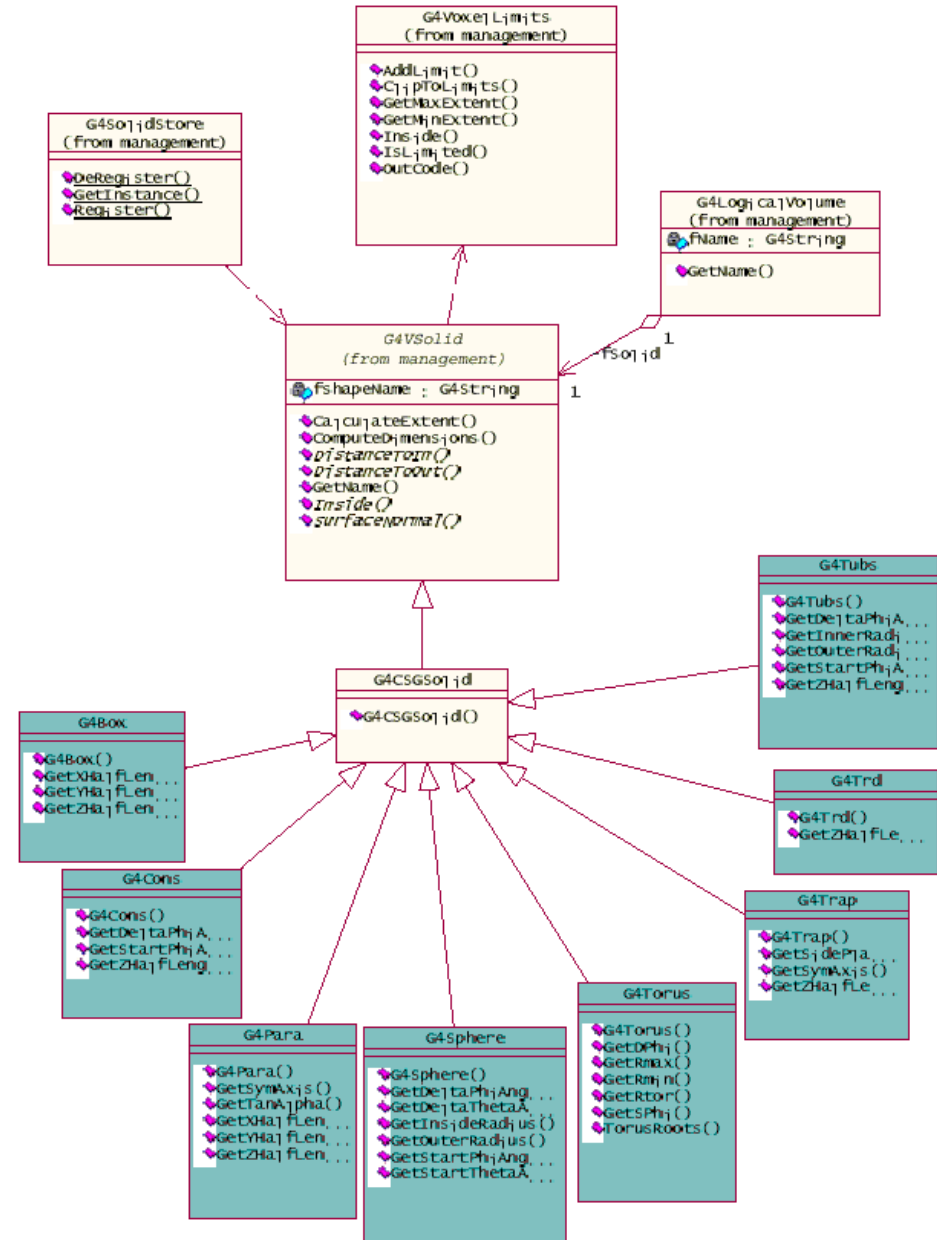
Step 3
Place it in the
coordinate
system of mother
volume

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

➤ The world volume

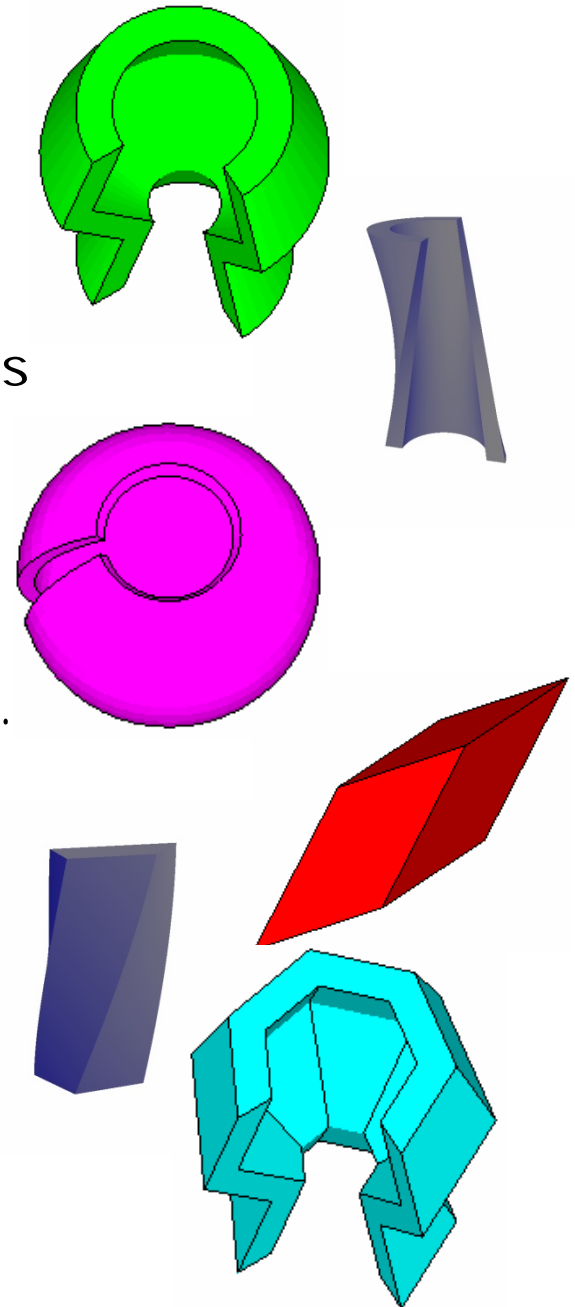
G4VSolid

- Abstract class. All solids in Geant4 derive from it
 - Defines but does not implement all functions required to:
 - compute distances to/from the shape
 - check whether a point is inside the shape
 - compute the extent of the shape
 - compute the surface normal to the shape at a given point
- Once constructed, each solid is automatically registered in a specific solid store



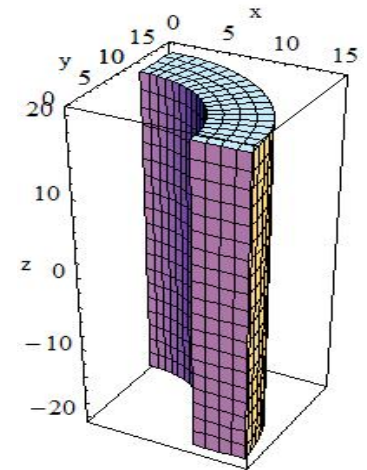
Solids

- 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, ...
 - BREP (Boundary REPresented) solids
 - G4BREPSolidPolycone, G4BSplineSurface, ...
 - Any order surface
 - Boolean solids
 - G4UnionSolid, G4SubtractionSolid, ...

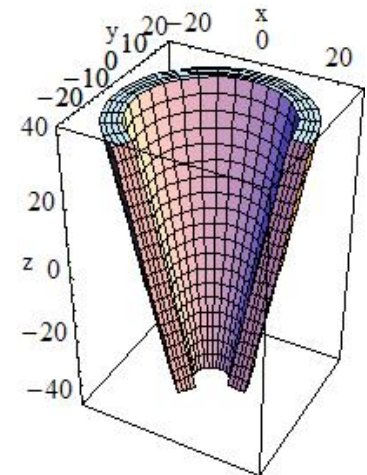


CSG: G4Tubs, G4Cons

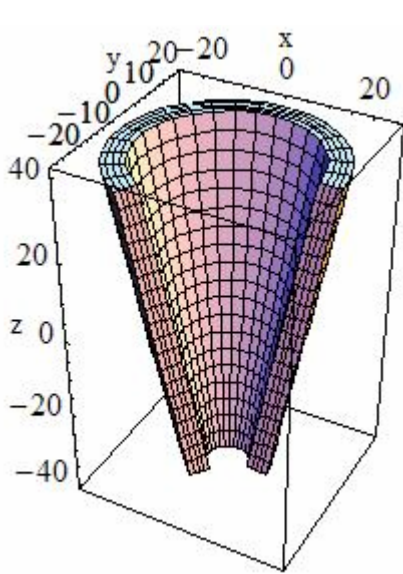
```
G4Tubs(const G4String& pname, // name
        G4double pRmin, // inner radius
        G4double pRmax, // outer radius
        G4double pDz, // Z half length
        G4double pSphi, // starting Phi
        G4double pDphi); // segment angle
```



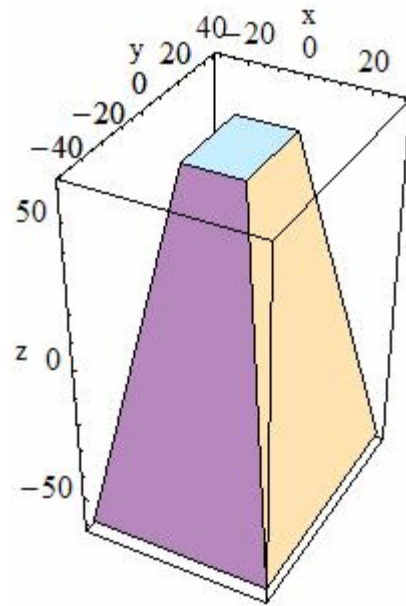
```
G4Cons(const G4String& pname, // name
        G4double pRmin1, // inner radius -pDz
        G4double pRmax1, // outer radius -pDz
        G4double pRmin2, // inner radius +pDz
        G4double pRmax2, // outer radius +pDz
        G4double pDz, // Z half length
        G4double pSphi, // starting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



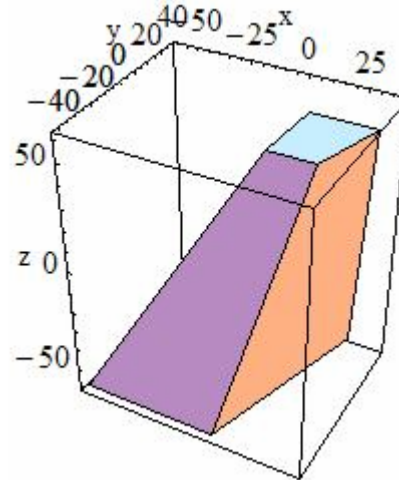
Other CSG solids



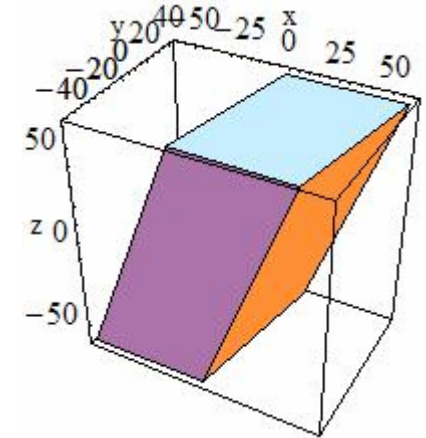
G4Cons



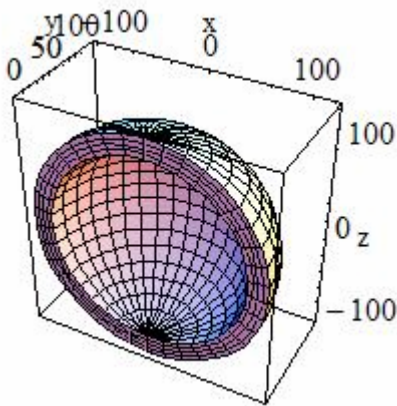
G4Trd



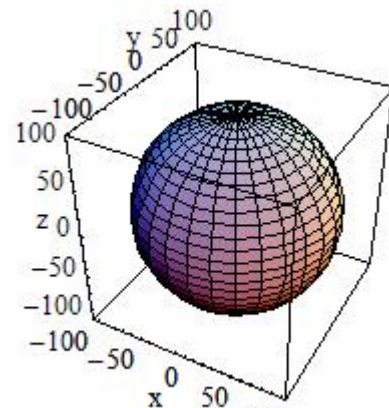
G4Trap



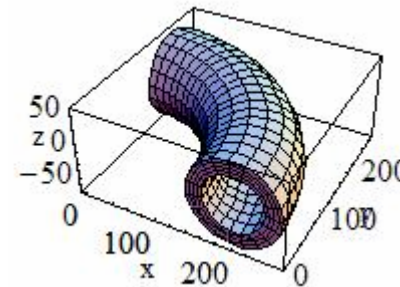
G4Para
(parallelepiped)



G4Sphere



G4Orb
(full solid sphere)



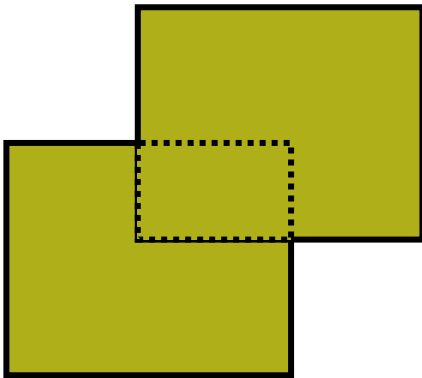
G4Torus

Consult to [Section 4.1.2 of Geant4 Application Developers Guide](#) for all available shapes.

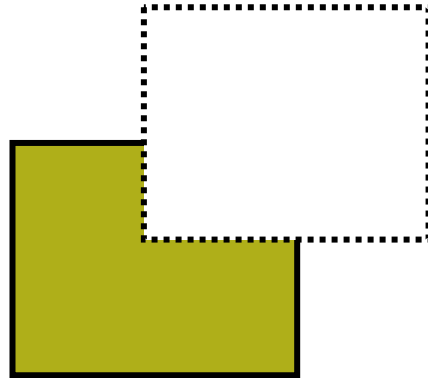
Boolean Solids

- ▶ 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 the third solid can be combined to 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

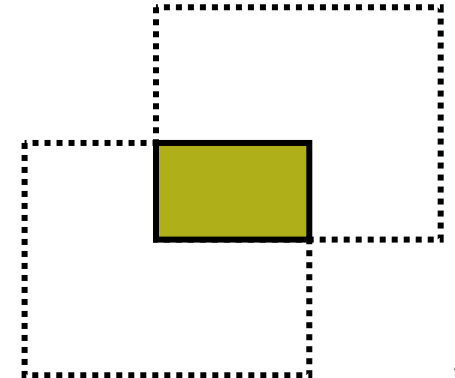
G4UnionSolid



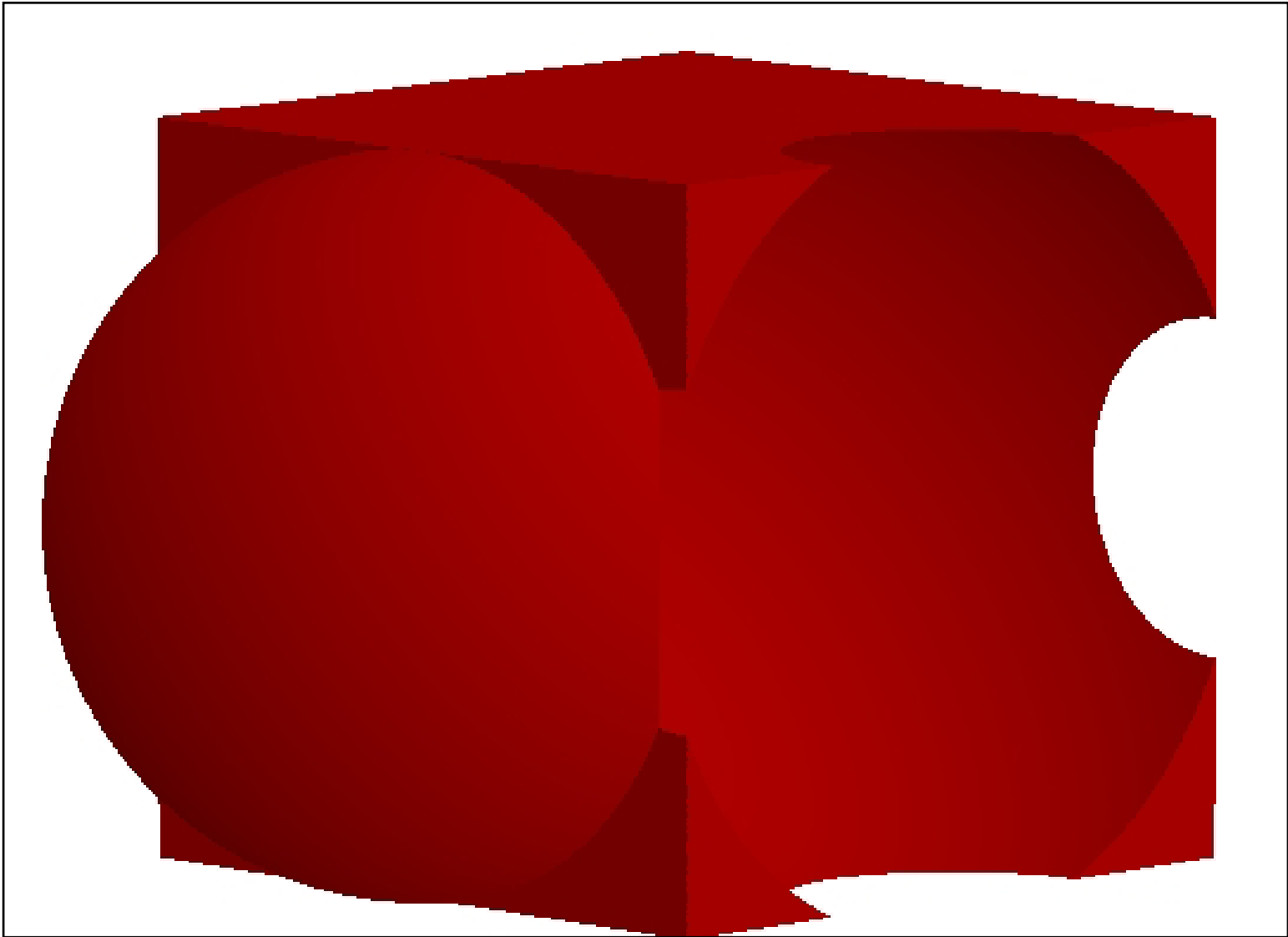
G4SubtractionSolid



G4IntersectionSolid



Boolean Solids



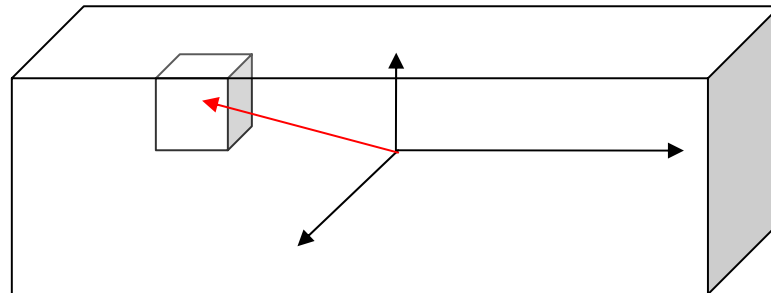
G4LogicalVolume

```
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

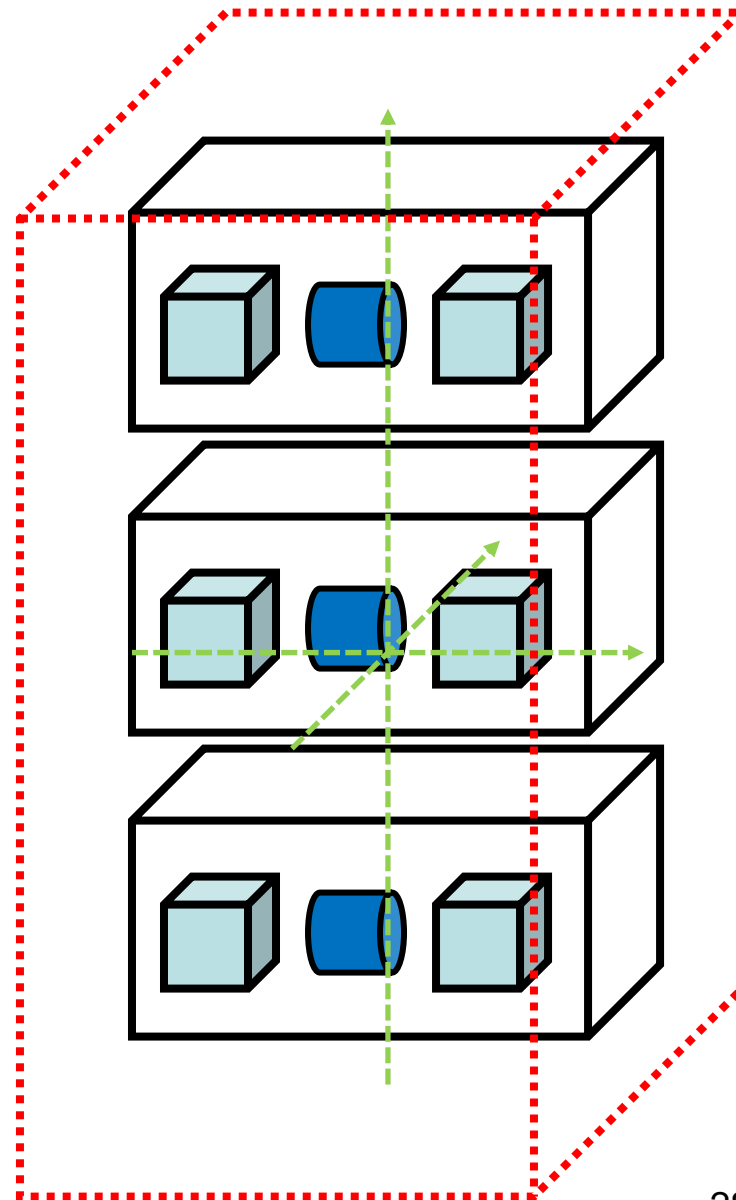
Geometrical hierarchy

- 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
 - The logical volume of mother knows the daughter volumes it contains
 - It is uniquely defined to be their mother volume



Geometrical hierarchy

- One logical volume can be placed more than once. One or more volumes can be placed in a mother volume
- 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 (root volume of the hierarchy)
 - 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

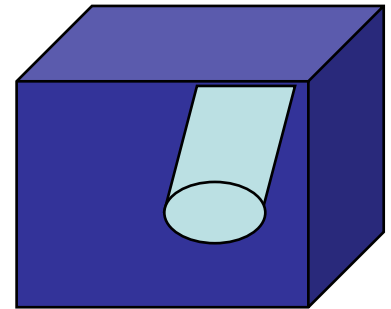


Region

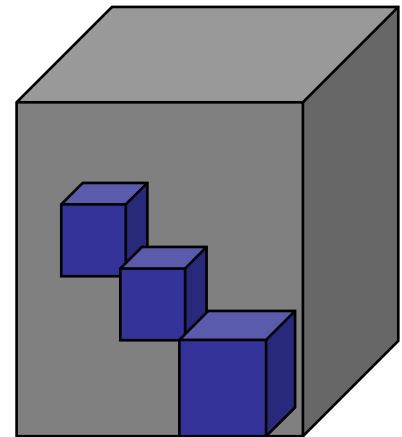
- A **region** may have its unique
 - Production thresholds (cuts)
 - If a region in the mass geometry does not have its own production thresholds, those of the default region are used (i.e., may not be those of the parent region).
 - User limits
 - Artificial limits affecting to the tracking, e.g. max step length, max number of steps, min kinetic energy left, etc.
 - You can set user limits directly to logical volume as well. If both logical volume and associated region have user limits, those of logical volume wins.
 - Fast simulation manager
 - Field manager
 - ...
- Please note :
 - World logical volume is recognized as **the default region**. User is **not** allowed to define a region to the world logical volume.

Physical Volumes

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



placement



repeated

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
```

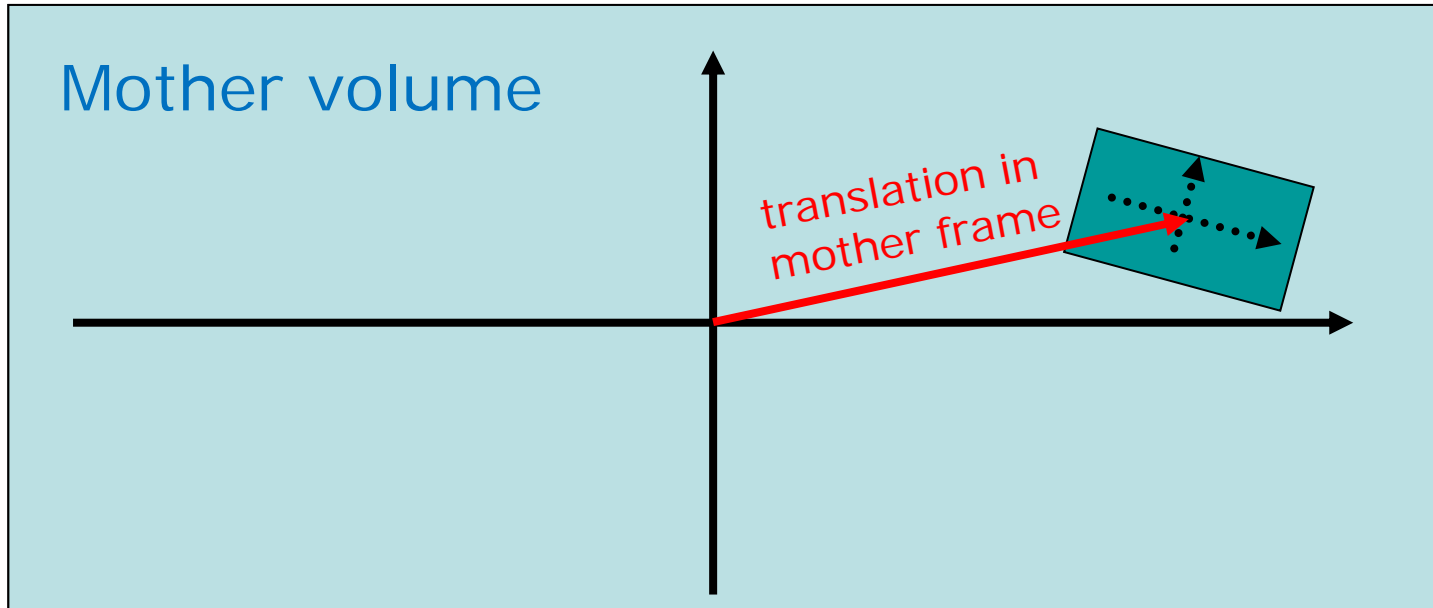
- Single volume positioned relatively to the mother volume
 - In a frame rotated and translated relative to the coordinate system of the mother volume
- Three additional constructors:
 - A simple variation: specifying the mother volume as a pointer to its physical volume instead of its logical volume.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

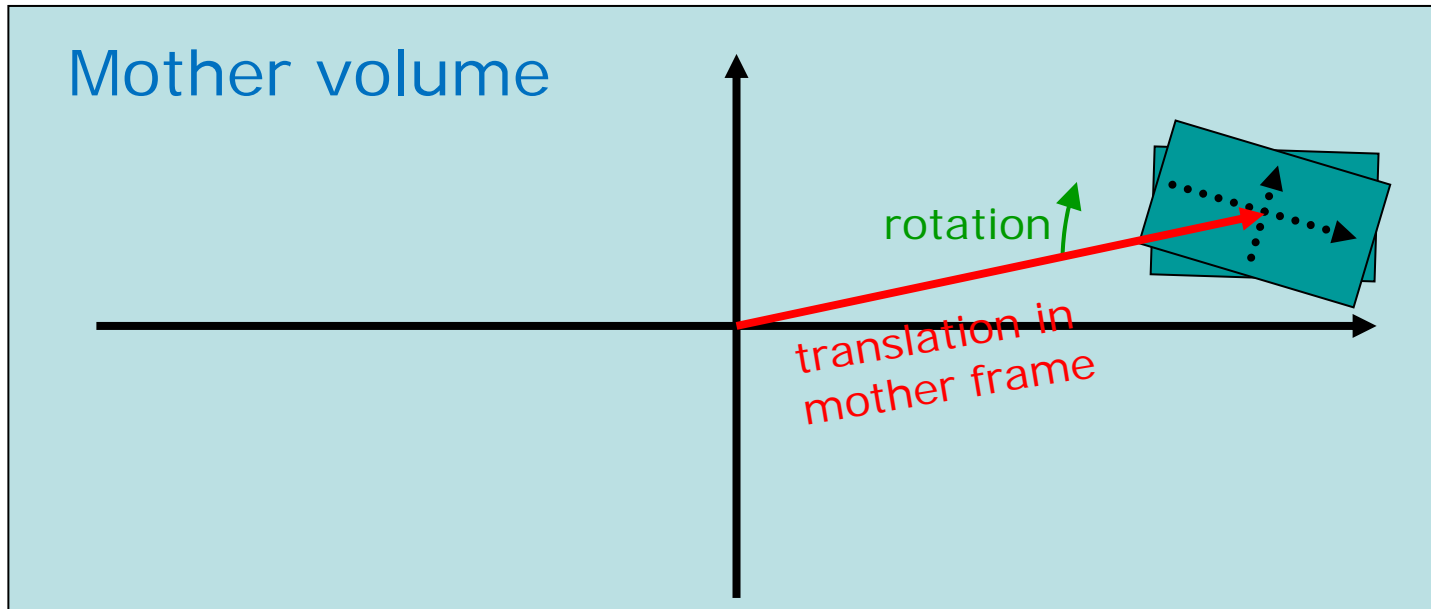
- Single volume positioned relatively to the mother volume



G4PVPlacement

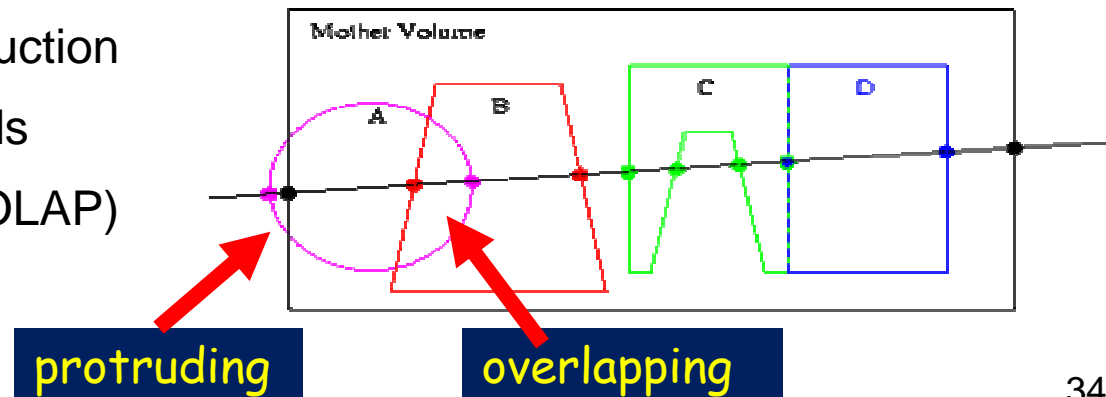
Rotation in mother frame ...

```
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
```



Tools for geometry check

- An **protruding** volume is a contained daughter volume which actually **protrudes** from its mother volume.
- Volumes are also often positioned in a same volume with the intent of not provoking intersections between themselves. When volumes in a common mother actually **intersect themselves** are defined as **overlapping**.
- Geant4 **does not allow** for malformed geometries, **neither protruding nor overlapping**.
 - The behavior of navigation is unpredictable for such cases.
- The problem of detecting overlaps between volumes is bounded by the complexity of the solid models description.
- Utilities are provided for detecting wrong positioning
 - Optional checks at construction
 - Kernel run-time commands
 - Graphical tools (DAVID, OLAP)



Tools for geometry check

- 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 lots of CPU time!)
 - **geometry/test/line_test**
to shoot a line along a specified direction and position
 - ...