Geant 4

Detector Description: Basics

http://cern.ch/geant4

PART II

Detector Description: the Basics

- *Components of a detector geometry and hierarchical relationship*
- *Volumes & solids*

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

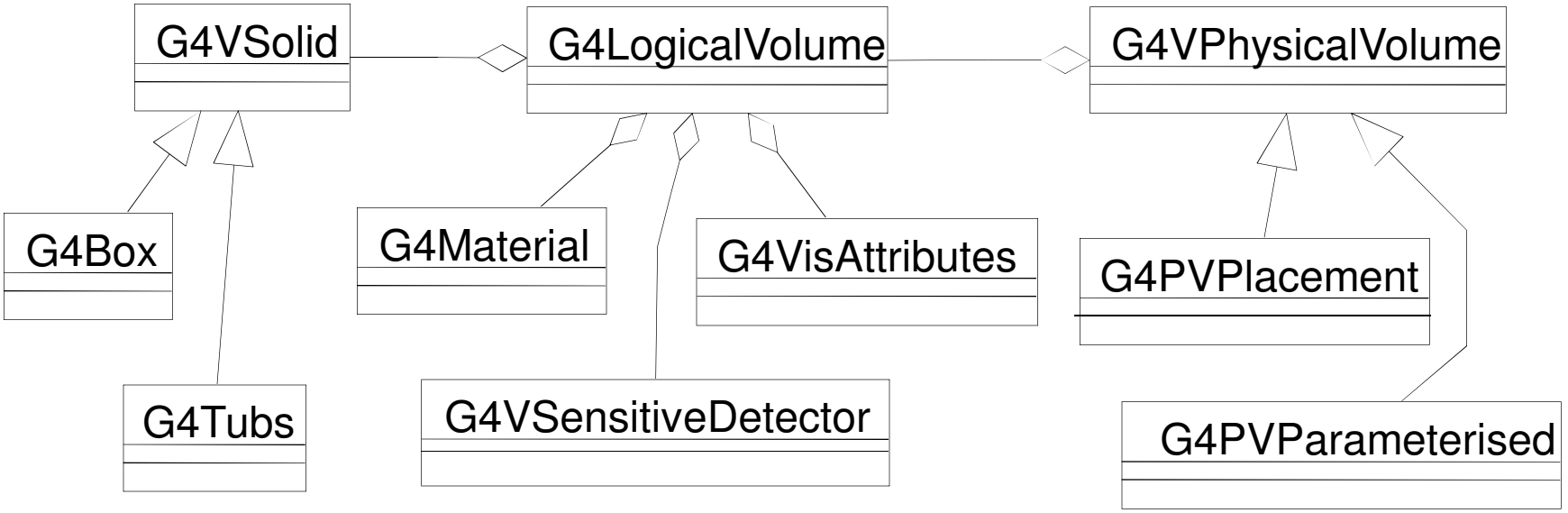
Creating a Detector Volume

- 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

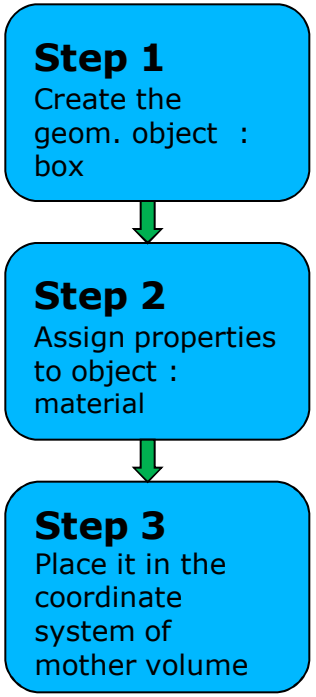
- Three conceptual layers
  - **G4VSolid** -- *shape, size*
  - **G4LogicalVolume** -- *daughter physical volumes, material, sensitivity, user limits, etc.*
  - **G4VPhysicalVolume** -- *position, rotation*



Detector geometry components

- 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);
```



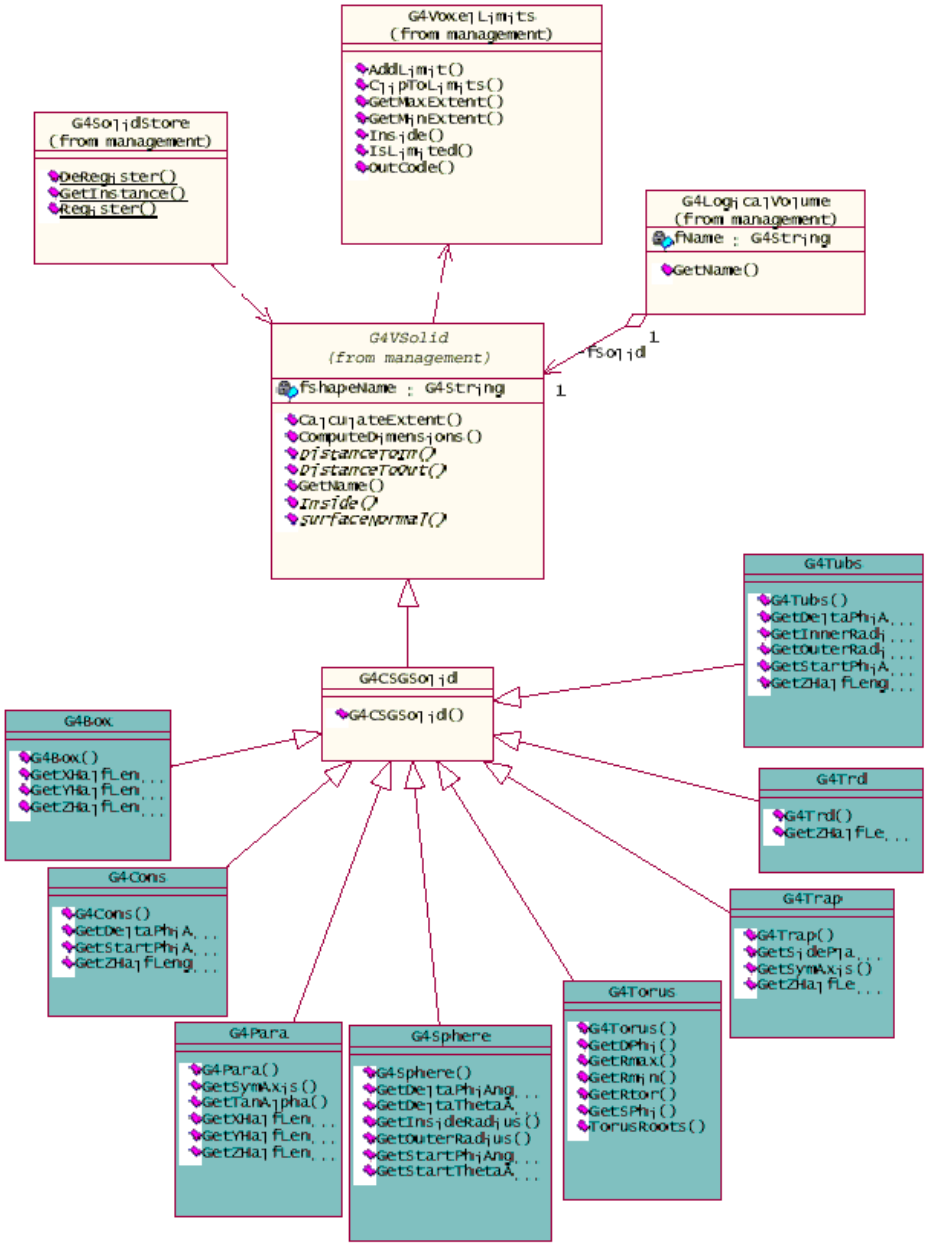
- A unique physical volume which represents the experimental area must exist and fully contains all other components
  - The world volume

Describing a detector - II

- *Solids*

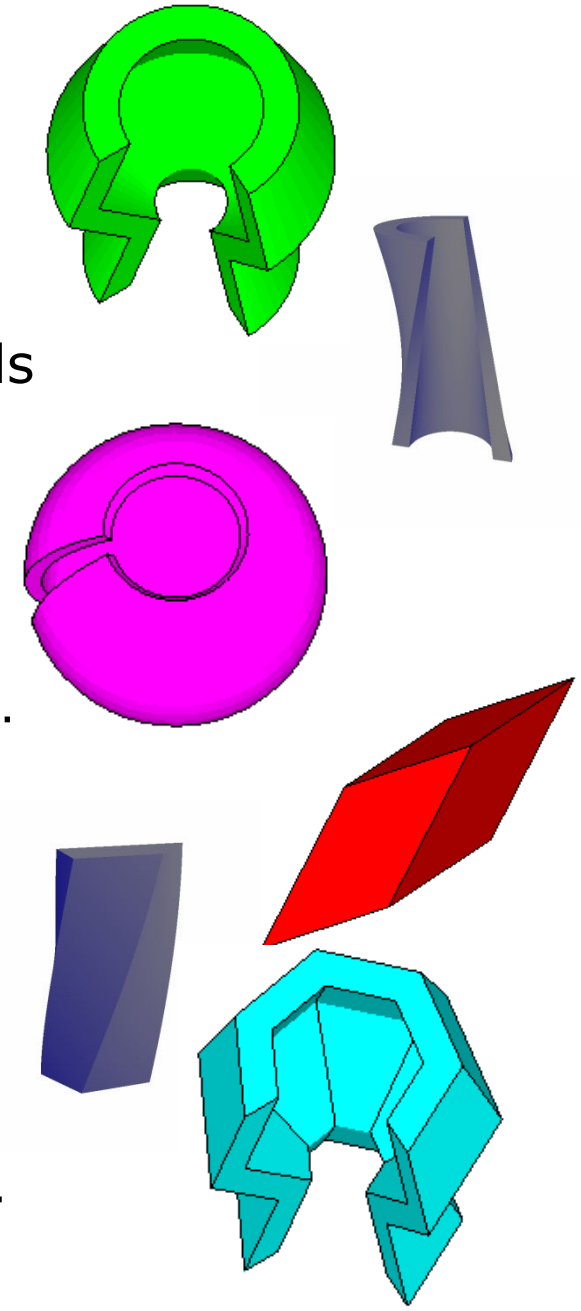
G4VSolid

- 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



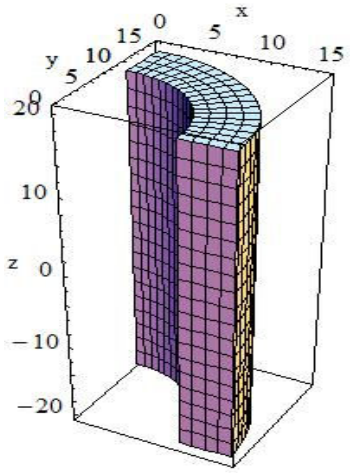
Solids

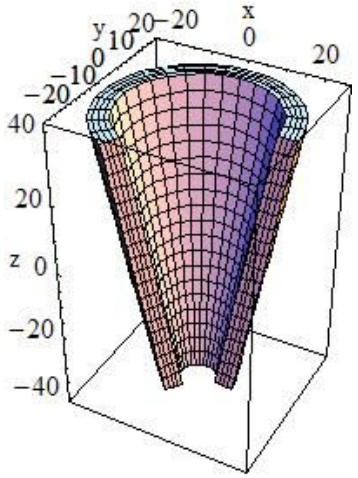
- Solids defined in Geant4:
  - CSG (Constructed Solid Geometry) solids
    - G4Box, G4Tubs, G4Cons, G4Trd, …
    - Analogous to simple GEANT3 CSG solids
  - Specific solids (CSG like)
    - G4Polycone, G4Polyhedra, G4Hype, ..
    - G4TwistedTubs, G4TwistedTrap, …
  - BREP (Boundary REPresented) solids
    - G4BREPSolidPolycone, G4BSplineSurface, …
    - Any order surface
  - Boolean solids
    - G4UnionSolid, G4SubtractionSolid, …



CSG: G4Tubs, G4Cons

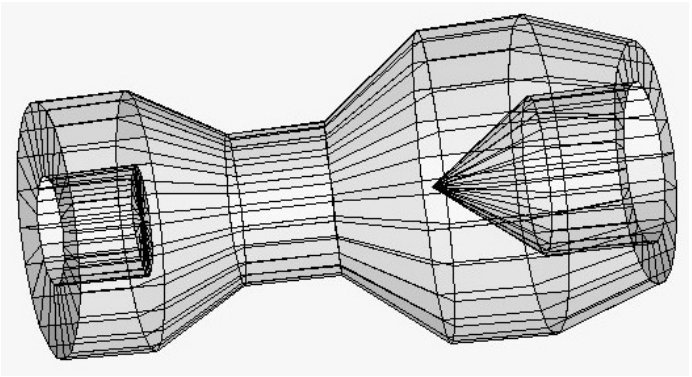
```
G4Tubs(const G4String& pname,   // name
       G4double  pRmin,   // inner radius
       G4double  pRmax,   // outer radius
       G4double  pDz,     // Z half length
       G4double  pSphi,   // starting Phi
       G4double  pDphi); // segment angle

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
```





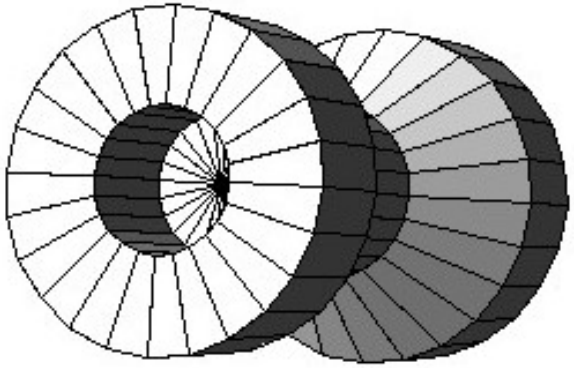
Specific CSG Solids: G4Polycone

```
G4Polycone(const G4String& pName,
                 G4double phiStart,
                 G4double phiTotal,
                 G4int numRZ,
           const G4double r[],
           const G4double z[]);
```

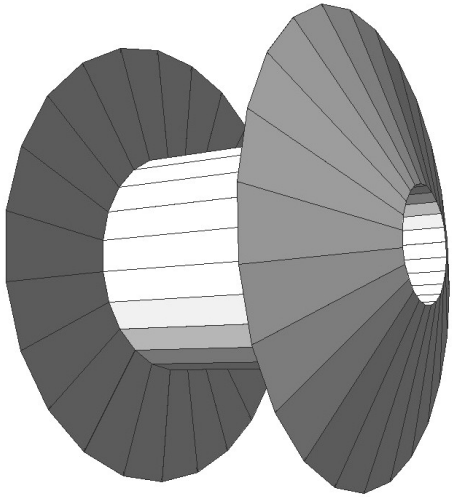


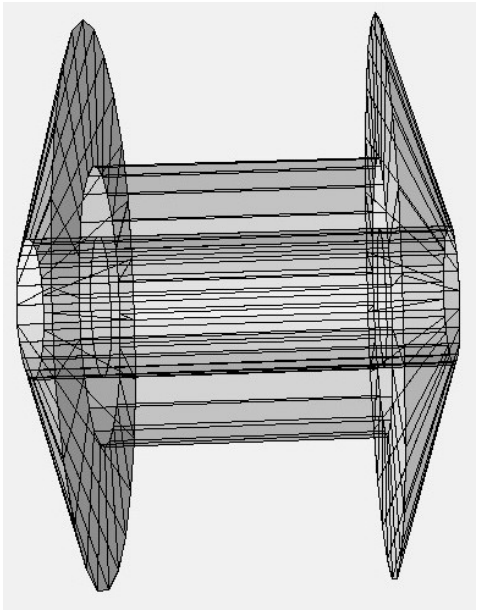
- **numRZ** - numbers of corners in the **r, z** space
- **r, z** - coordinates of corners

- Also available additional constructor using planes



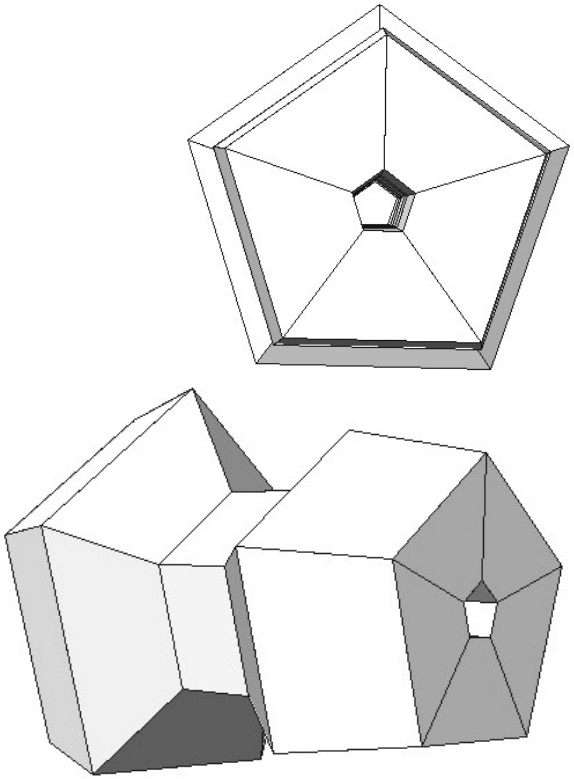
BREP Solids





- *BREP = Boundary REPresented Solid*
- Listing all its surfaces specifies a solid
  - e.g. 6 squares for a cube
- Surfaces can be
  - planar, 2nd or higher order
    - elementary BREPS
  - Splines, B-Splines,
    NURBS (Non-Uniform B-Splines)
    - advanced BREPS
- Few elementary BREPS pre-defined
  - box, cons, tubs, sphere, torus, polycone, polyhedra
- Advanced BREPS built through CAD systems

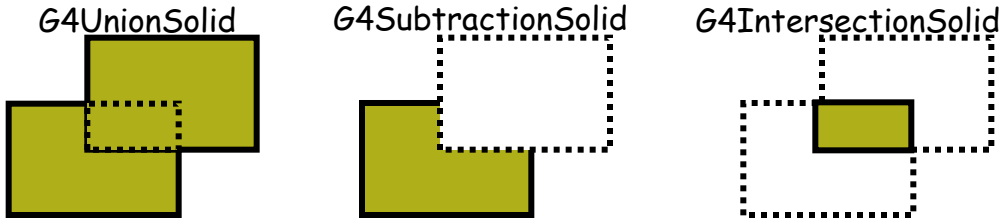
BREPS example:
G4BREPSolidPolyhedra



```
G4BREPSolidPolyhedra(const G4String& pName,
                     G4double phiStart,
                     G4double phiTotal,
                     G4int sides,
                     G4int nZplanes,
                     G4double zStart,
               const G4double zval[],
               const G4double rmin[],
               const G4double rmax[]);
```

- **sides** - numbers of sides of each polygon in the **x–y** plane
- **nZplanes** - numbers of planes perpendicular to the **z** axis
- **zval[]** - **z** coordinates of each plane
- **rmin[], rmax[]** - Radii of inner and outer polygon at each plane

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
    - Component solids must not be disjoint and must well intersect

```
G4Box box("Box", 20, 30, 40);
G4Tubs cylinder("Cylinder", 0, 50, 50, 0, 2*M_PI);  // r:     0 -> 50
                                                    // z:   -50 -> 50
                                                    // phi:   0 ->  2 pi
G4UnionSolid union("Box+Cylinder", &box, &cylinder);
G4IntersectionSolid intersect("Box Intersect Cylinder", &box, &cylinder);
G4SubtractionSolid subtract("Box-Cylinder", &box, &cylinder);
```

- Solids 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 solids

Areas, volumes and masses

- Surface area and geometrical volume of a generic solid or Boolean composition can be computed from the **solid**:

```
G4double GetSurfaceArea();
G4double GetCubicVolume();
```

- Overall mass of a geometry setup (sub-detector) can be computed from the **logical volume**:

```
G4double GetMass(G4Bool forced=false,
                 G4Material* parameterisedMaterial=0);
```

Describing a detector - III

- *Logical and Physical Volumes*

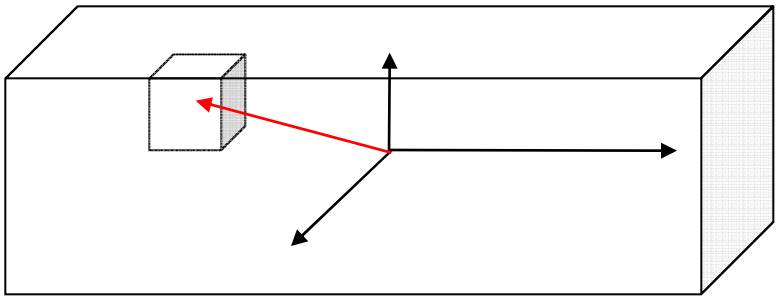
G4LogicalVolume

```
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
  - One or more volumes can be placed in a mother volume



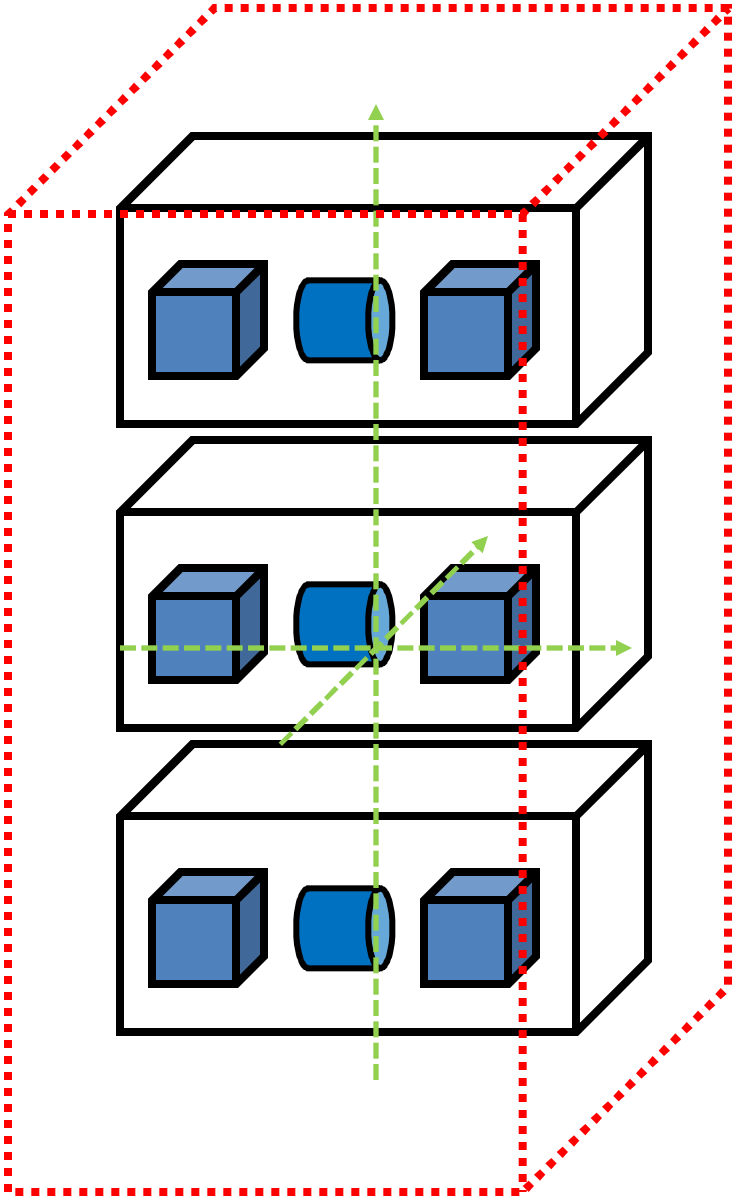
Geometrical hierarchy

- Mother and daughter volumes (cont.)
  - The logical volume of mother knows the daughter volumes it contains
    - It is uniquely defined to be their mother volume
    - If the logical volume of the mother is placed more than once, all daughters appear by definition in all these physical instances of the mother
- World volume is the root volume of the hierarchy
  - The world volume must be a unique physical volume which **fully contains with some margin** all other volumes
    - The world defines the global coordinate system
    - The origin of the global coordinate system is at the center of the world volume
    - Should not share any surface with contained geometry

19

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
  - 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

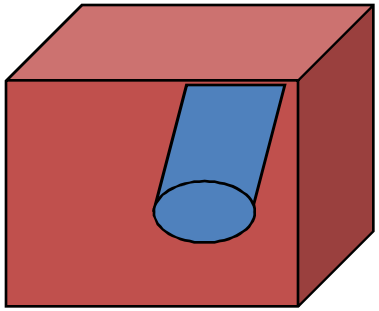


Kinds of G4VPhysicalVolume

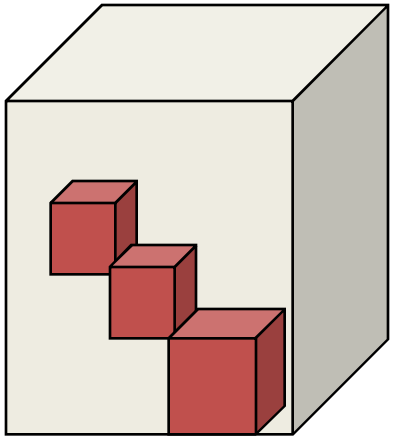
- G4PVPlacement 1 Placement = One Volume
  - A volume instance positioned once in a mother volume
- G4PVParameterised 1 Parameterised = Many Volumes
  - Parameterised by the copy number
    - Shape, size, material, position and rotation can be parameterised, by implementing a concrete class of **`G4VPVParameterisation`**.
  - Reduction of memory consumption
    - Parameterisation can be used only for volumes that either a) have no further daughters or b) are identical in size & shape.
- G4PVReplica 1 Replica = Many Volumes
  - Slicing a volume into smaller pieces (if it has a symmetry)

Physical Volumes

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



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
```

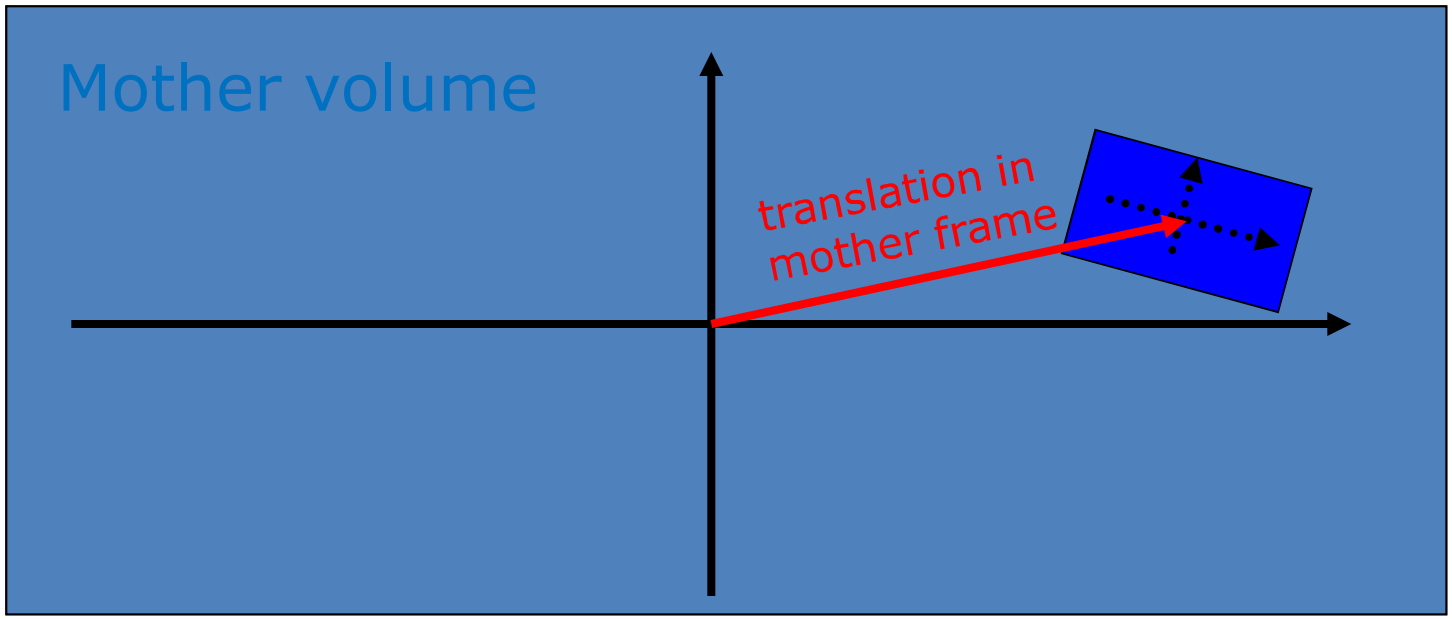
- Single volume positioned relatively to the mother volume
  - In a frame rotated and translated relative to the coordinate system of the mother volume
- Three additional constructors:
  - A simple variation: specifying the mother volume as a pointer to its physical volume instead of its logical volume.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

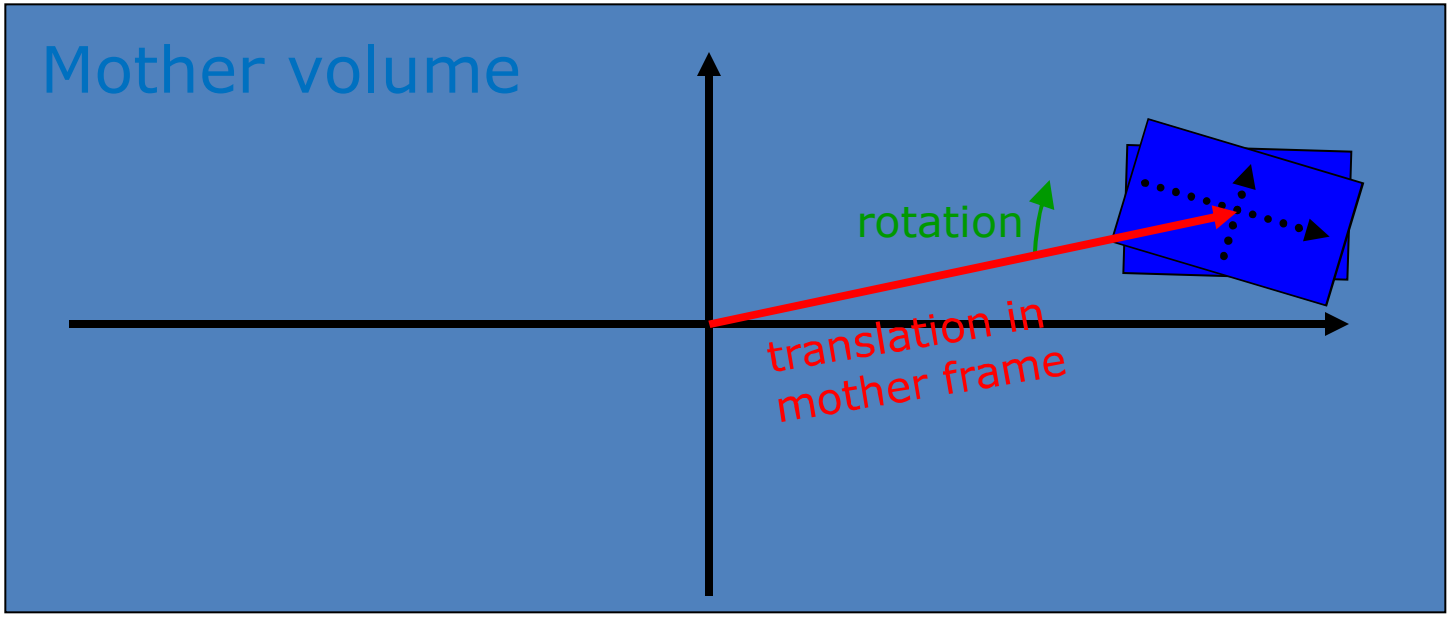
- Single volume positioned relatively to the mother volume



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
```



Exercise 1b

- *Placements*