Geometry - 3

I. Hrivnacova, IPN Orsay

Credits: T. Nikitina, J.Apostolakis, G.Cosmo, A. Lechner (CERN), M. Asai (SLAC) and others

Geant4 School at IFIN-HH, Bucharest,
14 - 18 November 2016

Outline

- Repeated placements
- Special techniques of placements

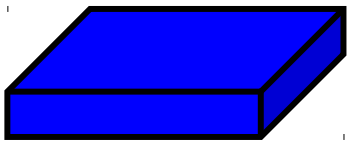
Repeated Placements

Physical Volumes

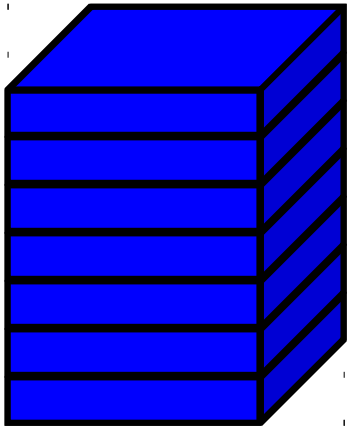
- Physical volume represents a placement of a daughter volume in its mother volume
  - It holds the information about the position of the daughter in the mother reference frame
- Physical volume types:
  - Simple placement: “placement”
  - Repeated placement: “replica”, “division”, “parameterised volume”
- A mother volume can contain either
  - More simple volume placements OR
  - One repeated volume

Replicated Volumes

- The mother volume is sliced into replicas, all of the same size and dimensions.
- Depending on the mother shape, 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



a daughter logical volume to be replicated



mother volume

G4PVReplica

- G4PVReplica constructor:

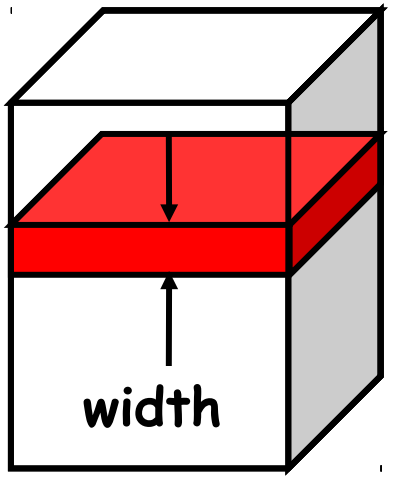
```
G4PVReplica(
   const G4String& name,          // physical volume name
   G4LogicalVolume* currentLV, // volume being replicated
   G4LogicalVolume* motherLV,  // mother logical volume
   const EAxis axis,              // axis of replication
   const G4int nofReplicas,    // number of replicas
   const G4double width,          // replication width
   const G4double offset = 0); // offset (optional)
```

- 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
  - An offset may be used only for tube/cone segment

Replica – axis, width, offset

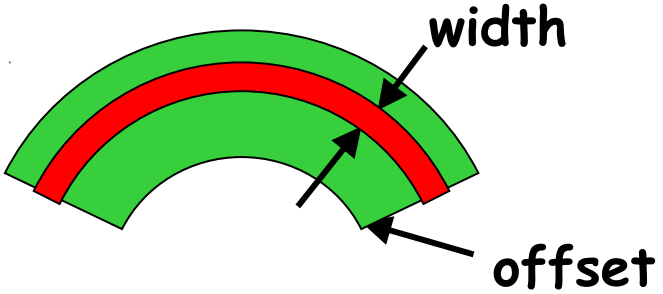
- Cartesian axes - kXAxis, kYAxis, kZAxis
  - Offset shall not be used
  - Center of n-th daughter is given as

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



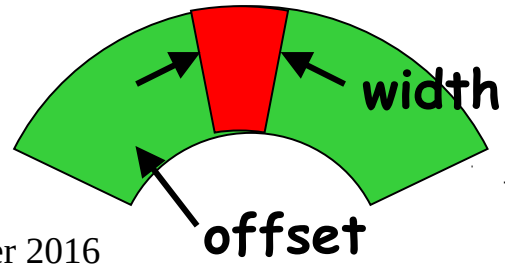
- Radial axis - kRho
  - Center of n-th daughter is given as

    `width*(n+0.5)+offset`



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

    `width*(n+0.5)+offset`



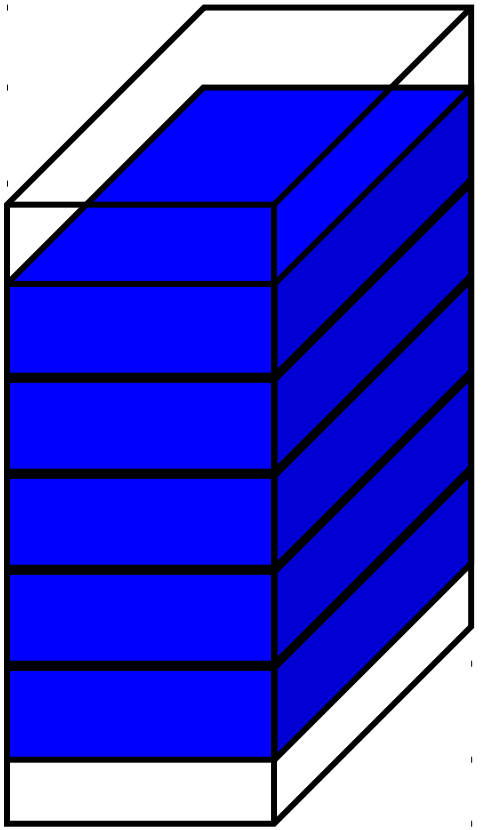
Example

- Tube replicated in phi axis

```
// mother tube volume
G4double dphi = 360.*deg;
G4VSolid* tubeS
  = new G4Tubs("tube", 20*cm, 50*cm, 30*cm, 0., dphi);
G4LogicalVolume* tubeLV
  = new G4LogicalVolume(tubeS, Ar, "tube");
new G4PVPlacement(0, G4ThreeVector(),
                  tubeLV, "tube", worldLV, false, 0);

// division in 6 phi segments
G4double divDphi = dphi/6.;
G4VSolid* divTubeS
  = new G4Tubs("divTube", 20*cm, 50*cm, 30*cm,
               -divDphi/2., divDphi);
G4LogicalVolume* divTubeLV
  = new G4LogicalVolume(divTubeS, Ar, "divTube");
new G4PVReplica("divTube", divTubeLV, tubeLV, kPhi, 6, divDphi);
```

Divisions

- Implemented as “special” kind of parameterised volumes:
  - G4PVDivision class derived from G4PVParameterised
  - But simpler to define as the parameterisation is calculated automatically using the values provided in input
- Similar to G4PVReplica
  - But it allows gaps in between mother and daughter volumes or between daughters (offset)
- Applies to CSG-like and some specific solids only:
  - Box, tubs, cons, para, trd, polycone, polyhedra



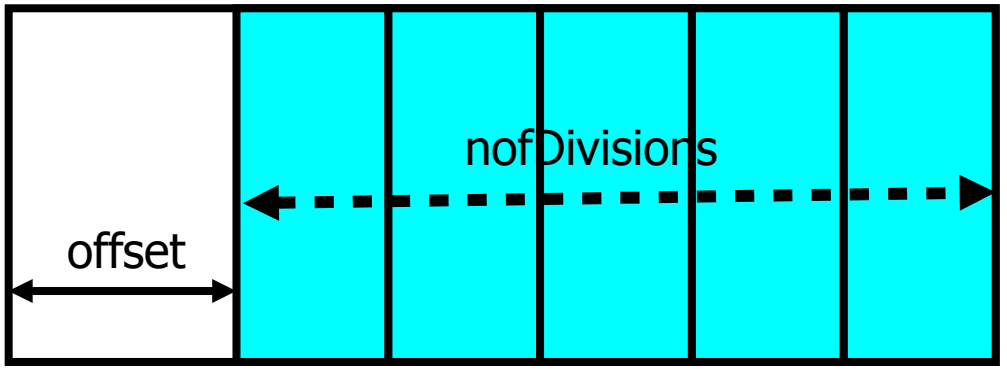
G4PVDivision (1)

- Constructor 1:

```
G4PVDivision(
   const G4String& name,          // physical volume name
   G4LogicalVolume* currentLV,    // volume being replicated
   G4LogicalVolume* motherLV,     // mother logical volume
   const EAxis axis,              // axis of replication
   const G4int nofDivisions,      // number of divisions
   const G4double offset);        // division offset
```

- The size (width) of the daughter volume is calculated as

  `( (size of mother) - offset ) / nofDivisions`



G4PVDivision (2)

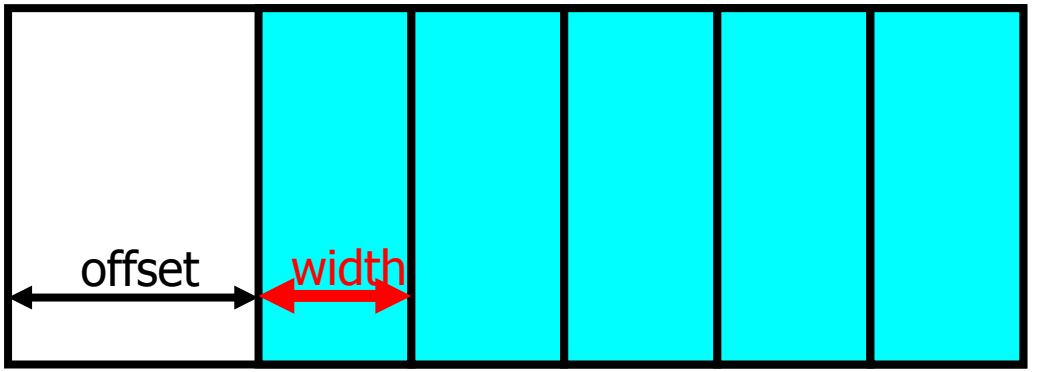
- Constructor 2:

```
G4PVDivision(
   const G4String& name,          // physical volume name
   G4LogicalVolume* currentLV,    // volume being replicated
   G4LogicalVolume* motherLV,     // mother logical volume
   const EAxis axis,              // axis of replication
   const G4double width,          // division width
   const G4double offset);        // division offset
```

- The number of daughters volumes is calculated as

  `( (size of mother) - offset ) / nofDivisions`

As many divisions as `width` and `offset` allow

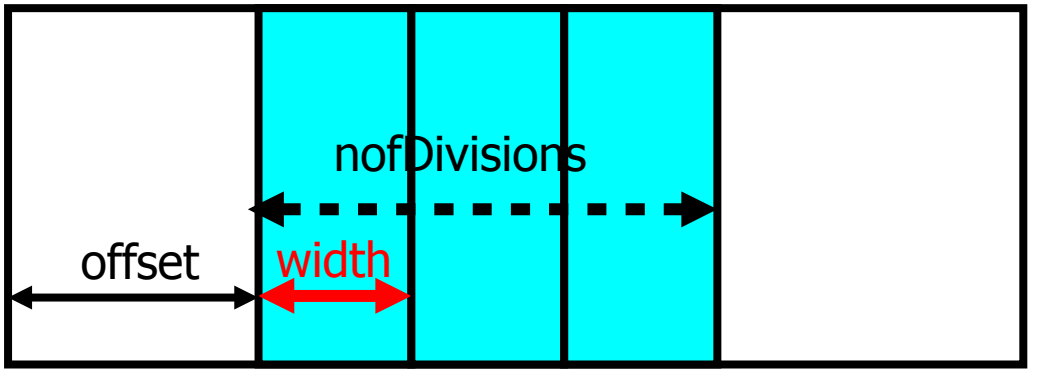


G4PVDivision (3)

- Constructor 3:

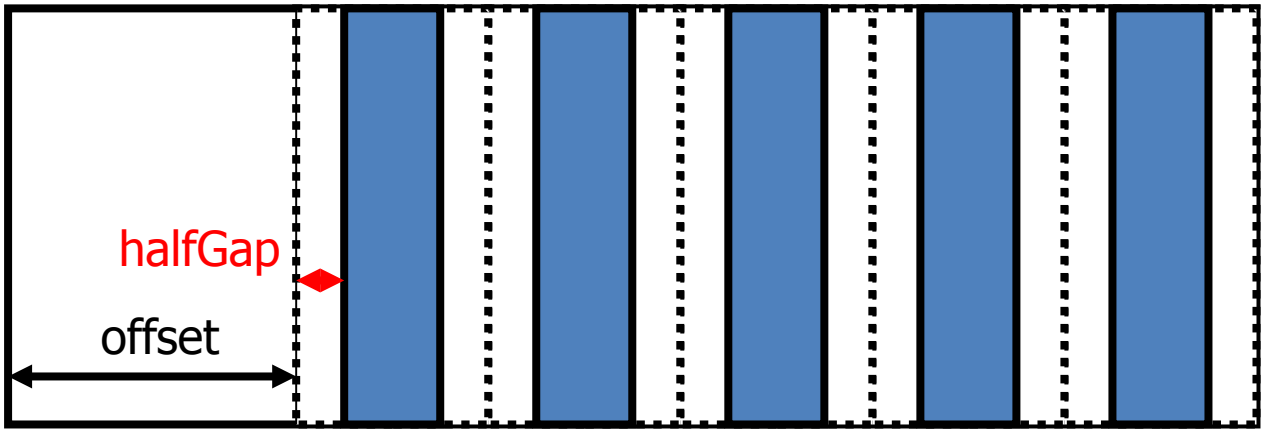
```
G4PVDivision(
   const G4String& name,          // physical volume name
   G4LogicalVolume* currentLV,    // volume being replicated
   G4LogicalVolume* motherLV,     // mother logical volume
   const EAxis axis,              // axis of replication
   const G4int nofDivisions,      // number of divisions
   const G4double width,          // division width
   const G4double offset);        // division offset
```

- `nofDivisions` daughters of `width` thickness



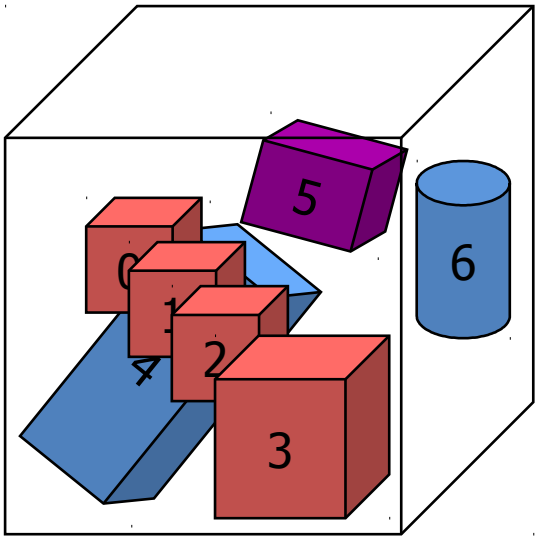
G4PVDivision (4)

- It is also possible to add `const G4double halfGap` paramater in addition to those in previous constructors



Parameterised Volumes

- The properties of the “replicas” in their mother volume are defined in a user parameterisation class derived from G4VPVParameterisation
- The properties which must be always provided:
  - Where it is positioned (transformation, rotation)
- Optional:
  - The size of the solid (dimensions)
  - The type of the solid, material, sensitivity, vis attributes
- The properties of the “replicas” are defined via their copyNumber



G4PVParameterised

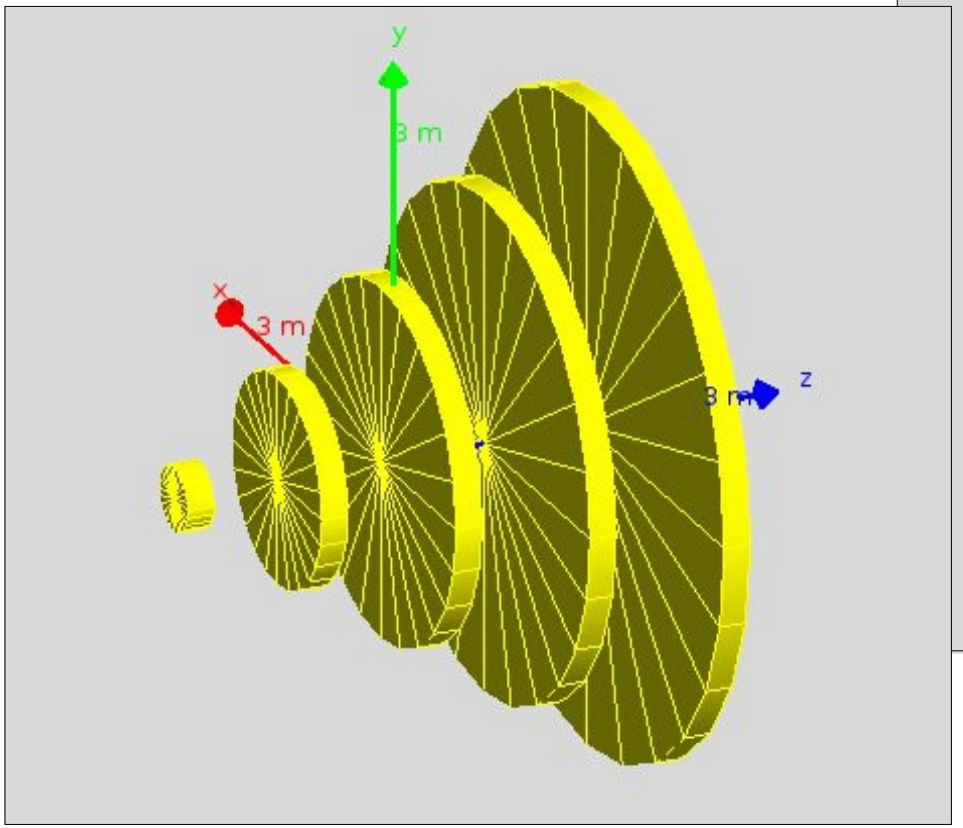
- G4PVParemeterised constructor:

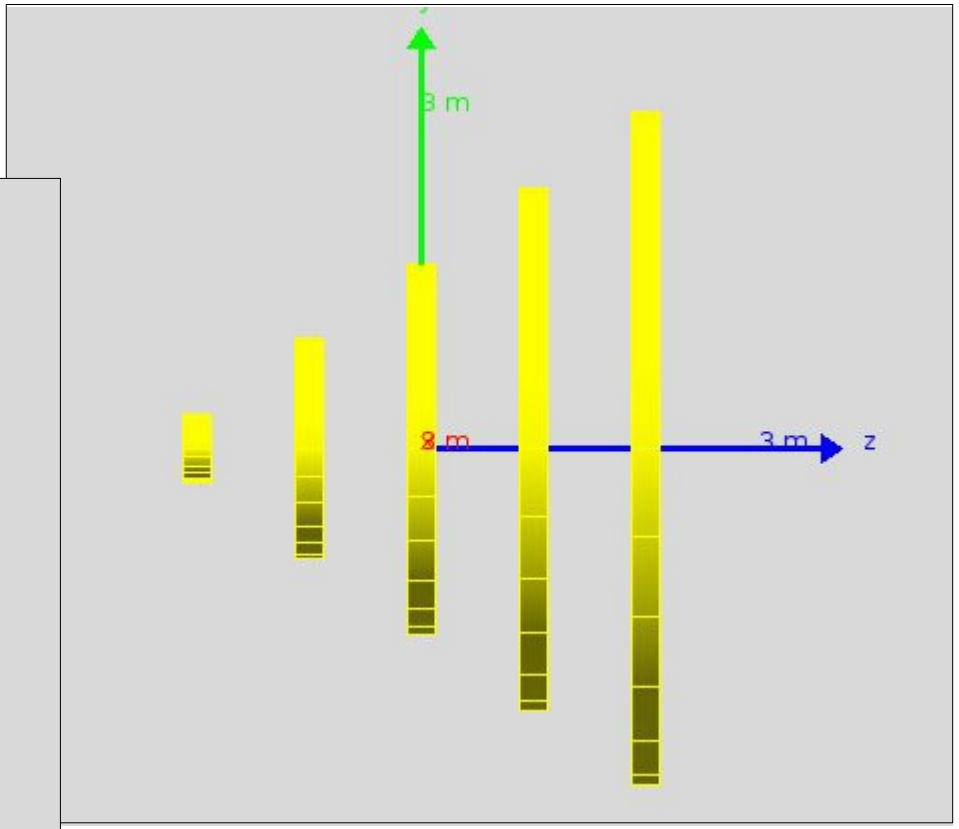
```
G4PVParameterised(
  const G4String& name,          // physical volume name
  G4LogicalVolume* currentLV, // volume being replicated
  G4LogicalVolume* motherLV,  // mother logical volume
  const EAxis axis,              // axis of replication
  const G4int nofReplicas,    // number of replicas
  G4VPVParameterisation* myParam); // parameterisation
```

- Features and restrictions:
  - Replicates the volume `nofReplicas` times using the parameterisation `myParam`, within the mother volume
  - The positioning of the replicas is dominant along the specified Cartesian axis
    - If kUndefined is specified as axis, 3D voxelisation for optimisation of the geometry is adopted

Example

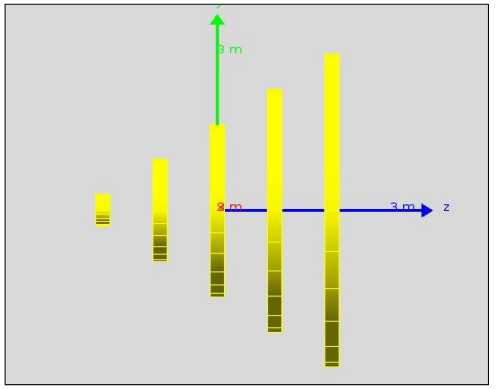
- Basic example B2: the same geometry with simple placements (B2a) and parameterised volume (B2b)





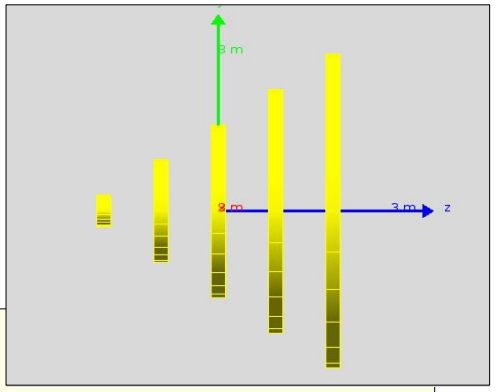
six tracking chambers of increasing transverse size

MyParameterisation.hh



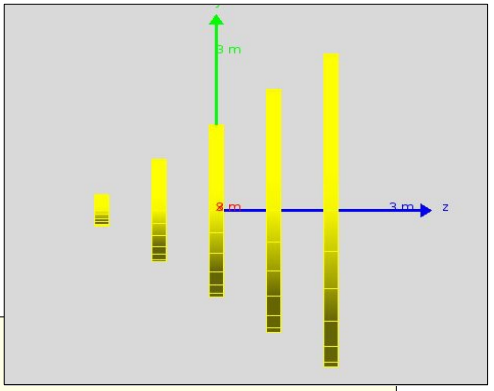
```
class B2bChamberParameterisation : public G4VPVParameterisation
{
  public:
    B2bChamberParameterisation(..);
    virtual ~B2bChamberParameterisation();

    virtual void ComputeTransformation(
                    const G4int copyNo,
                    G4VPhysicalVolume* physVol) const;
    Virtual void ComputeDimensions (
                    G4Tubs & trackerLayer, const G4int copyNo,
                    const G4VPhysicalVolume* physVol) const;
   private:
    // Dummy declarations to get rid of warnings ...
    // Data members of the class (with self-descriptive names)
};
```

MyParameterisation.cc



```
void B2bChamberParameterisation::ComputeTransformation(
  const G4int copyNo, G4VPhysicalVolume* physVol) const
{
  // Note: copyNo will start with zero!
  G4double zPosition = fStartZ + copyNo * fSpacing;
  G4ThreeVector origin(0, 0, zPosition);
  physVol->SetTranslation(origin);
  physVol->SetRotation(0);
}

void B2bChamberParameterisation::ComputeDimensions(
  G4Tubs& trackerChamber, const G4int copyNo,
  const G4VPhysicalVolume*) const
{
  // Note: copyNo will start with zero!
  G4double rmax = fRmaxFirst + copyNo * fRmaxIncr;
  trackerChamber.SetInnerRadius(0);
  trackerChamber.SetOuterRadius(rmax);
  trackerChamber.SetZHalfLength(fHalfWidth);
  trackerChamber.SetStartPhiAngle(0.*deg);
  trackerChamber.SetDeltaPhiAngle(360.*deg);
}
```

MyDetectorConstruction.cc

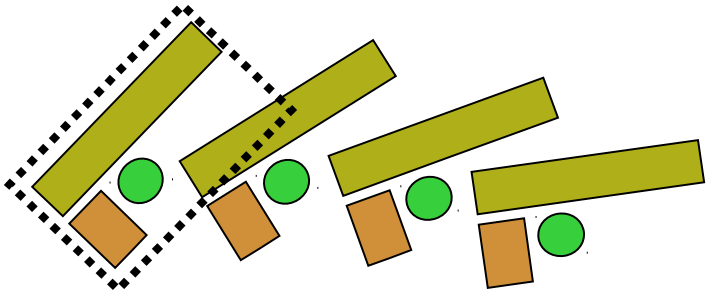



```
void B2bDetectorConstruction:Construct {
  ...
  G4Tubs* chamberS
    = new G4Tubs("tracker",0, 100*cm, 100*cm, 0.*deg, 360.*deg);
  fLogicChamber
    = new G4LogicalVolume(chamberS,fChamberMaterial,"Chamber",0,0,0);

  G4VPVParameterisation* chamberParam
    = new B2bChamberParameterisation(..);

  new G4PVParameterised(
          "Chamber",       // their name
          fLogicChamber,   // their logical volume
          trackerLV,       // Mother logical volume
          kZAxis,          // Are placed along this axis
          NbOfChambers,    // Number of chambers
          chamberParam,    // The parametrisation
          fCheckOverlaps); // checking overlaps
}
```

Special Techniques of Placements
Assemblies, Reflections

Grouping Volumes

- To represent a regular pattern of positioned volumes, composing a more or less complex structure
  - structures which are hard to describe with simple replicas or parameterised volumes
  - structures which may consist of different shapes
  - too densely positioned to utilize a mother volume
- Assembly volume
  - acts as an envelope for its daughter volumes
  - its role is over once its logical volume has been placed
  - daughter physical volumes become independent copies in the final structure
- Participating daughter logical volumes are treated as triplets
  - logical volume
  - translation w.r.t. envelop
  - rotation w.r.t. envelop



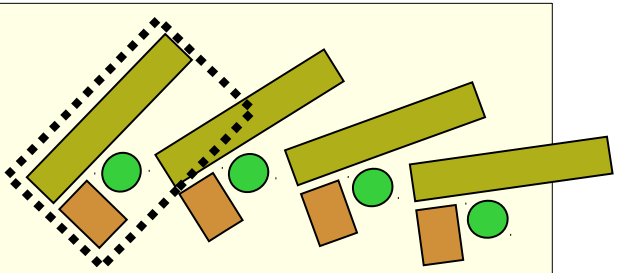
G4AssemblyVolume

```
G4AssemblyVolume::AddPlacedVolume(
                        G4LogicalVolume* volume,
                        G4ThreeVector& translation,
                        G4RotationMatrix* rotation );
```

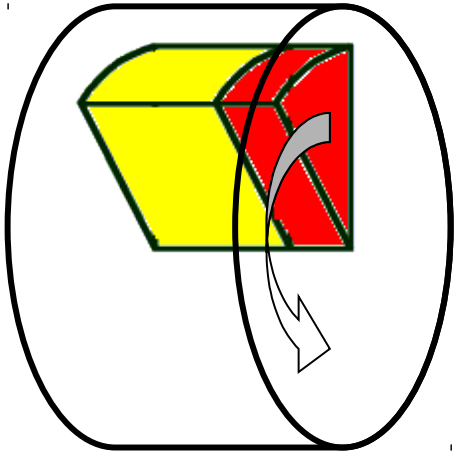
- Helper class to combine daughter logical volumes in arbitrary way
  - Imprints of the assembly volume are made inside a mother logical volume through G4AssemblyVolume::MakeImprint(...)
- Each physical volume name is generated automatically
  - Format: av_WWW_impr_XXX_YYY_ZZZ
    - WWW – assembly volume instance number
    - XXX – assembly volume imprint number
    - YYY – name of the placed logical volume in the assembly
    - ZZZ – index of the associated logical volume
- Generated physical volumes (and related transformations) are automatically managed (creation and destruction)

G4AssemblyVolume: Example

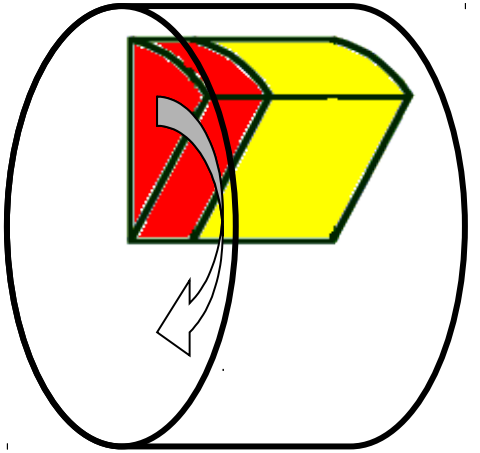
```
// Define the green box
G4VSolid* longBoxS = new G4Box("longBox", ...);
G4LogicalVolume* longBoxLV
  = new G4LogicalVolume(longBoxS, Pb, "longBox");
// Define the smallbox -> smallBoxLV
// Define the tube -> tubeLV

// Make assembly
G4AssemblyVolume* myAssembly = new G4AssemblyVolume();
// Define transformations of volume inside the assembly:
// lbPosition, lbRotation, ...
myAssembly->AddPlacedVolume(longBoxLV,lbPosition, lbRotation);
myAssembly->AddPlacedVolume(smallBoxLV,sbPosition, sbRotation);
myAssembly->AddPlacedVolume(tubeLV,tubePosition, tubeRotation);

// Now place the assemblies
for (G4int int i = 0; i < 4; i++ ) {
  // Define the position and rotation of each assembly
  // ithPosition, ithRotation
  myAssembly->MakeImprint(worldLV, ithPosition, ithRotation);
}
```

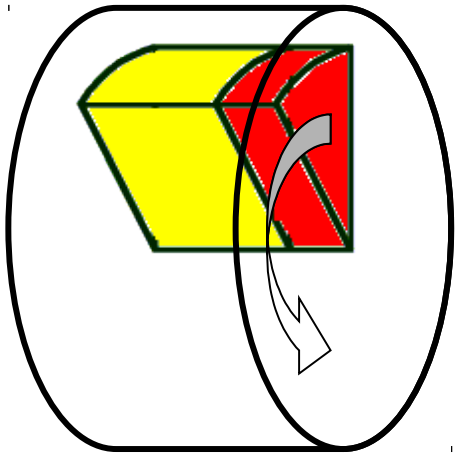


Reflecting volumes

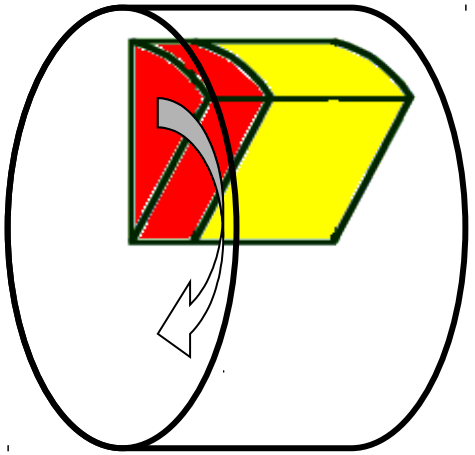


- Let's take an example of a pair of mirror symmetric volumes.
- Such geometry cannot be made by parallel transformation or 180 degree rotation.





Reflecting volumes



- G4ReflectedSolid
  - Utility class representing a solid shifted from its original reference frame to a new symmetric one
  - The reflection (G4Reflect[X/Y/Z]3D) is applied directly to a solid, and a reflected solid is placed with “pure” rotation and translation
- G4ReflectionFactory
  - Singleton object using G4ReflectedSolid for generating placements of reflected volumes
  - Provides tools to detect/return a reflected volume
  - Reflections can be applied to CSG and specific solids

Reflecting hierarchies of volumes - 1

```
G4PhysicalVolumesPair G4ReflectionFactory::Place(
  const G4Transform3D& transform3D, // the transformation
  const G4String& name,             // the name
  G4LogicalVolume* currentLV,       // the logical volume
  G4LogicalVolume* motherLV,        // the mother volume
  G4bool noBool,                    // currently unused
  G4int copyNo);                    // optional copy number
```

- Used for normal placements:
  1) Performs the transformation decomposition
  2) Generates a new reflected solid and logical volume.
     - Retrieves it from a map if the reflected object is already created.
  3) Transforms any daughter and places them in the given mother
  4) Returns a pair of physical volumes, the second being a placement in the reflected mother
- G4PhysicalVolumesPair is
  - `std::map<G4VPhysicalVolume*,G4VPhysicalVolume*>`

Reflecting hierarchies of volumes - 2

```
G4PhysicalVolumesPair G4ReflectionFactory::Replicate(
  const G4String& name,                 // the name
  G4LogicalVolume* currentLV,           // the logical volume
  G4LogicalVolume* motherLV,            // the mother volume
  Eaxis axis,                           // axis of replication
  G4int nofReplica,                     // number of replicas
  G4double width,                       // width of single replica
  G4int offset = 0);                    // offset (optional)
```

- Creates replicas in the given mother volume
- Returns a pair of physical volumes, the second being a replica in the reflected mother