More on Geometry

I. Hrivnacova, IJCLab Orsay

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

Geant4 IN2P3 and ED PHENIICS Tutorial,
16 – 20 May 2022

Outline

- More on detector description
  - Repeated placements
  - Special techniques of placements
  - Parallel geometries
- Geometry checking tools
- Optimization techniques

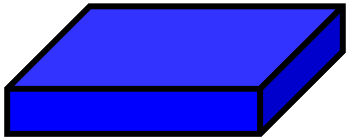
Repeated Placements

Physical Volumes

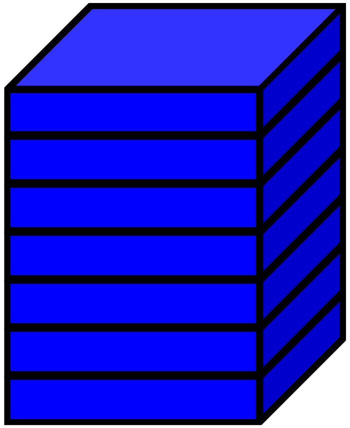
- Physical volume represents a placement of a daughter volume in its mother volume
  - It holds the information about the position of the daughter in the mother reference frame
- Physical volume types:
  - Simple placement: “placement”
  - Repeated placement: “replica”, “division”, “parameterised volume”
- 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:

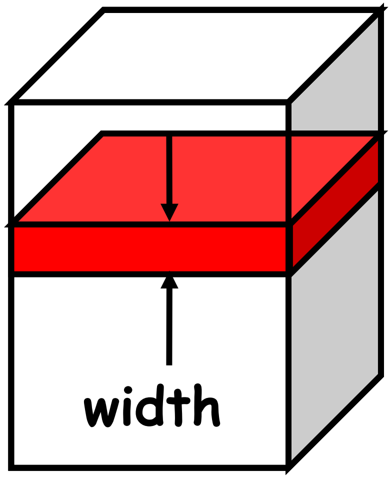
```
G4PVReplica(
   const G4String& name,         // physical volume name
   G4LogicalVolume* currentLV, // volume being replicated
   G4LogicalVolume* motherLV,  // mother logical volume
   const EAxis axis,             // axis of replication
   const G4int nofReplicas,      // number of replicas
   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

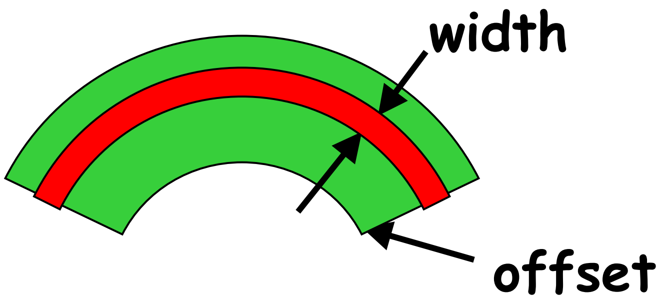
- Cartesian axes - kXAxis, kYAxis, kZAxis
  - Offset shall not be used
  - Center of n-th daughter is given as

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



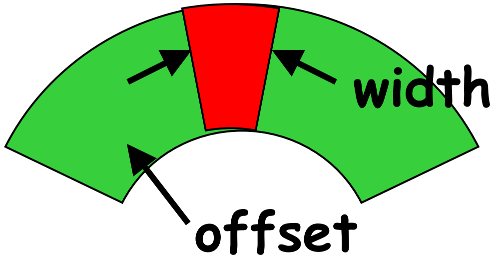
- Radial axis - kRho
  - Center of n-th daughter is given as

    `width*(n+0.5)+offset`



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

    `width*(n+0.5)+offset`



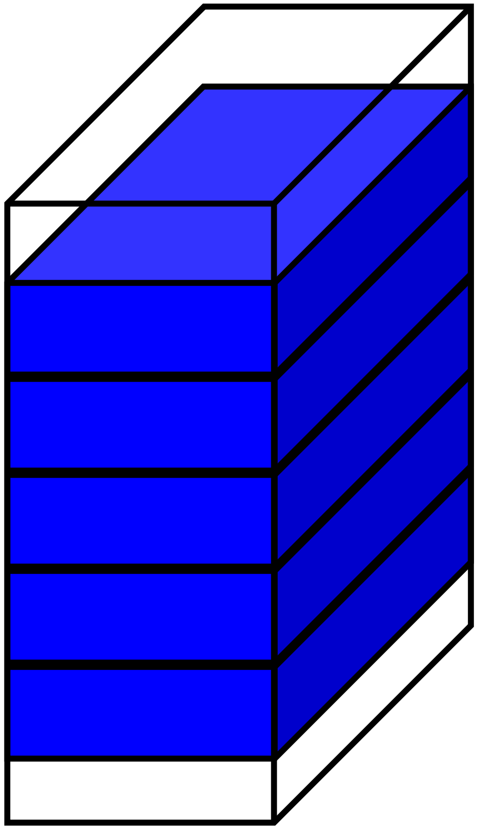
Example

- 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

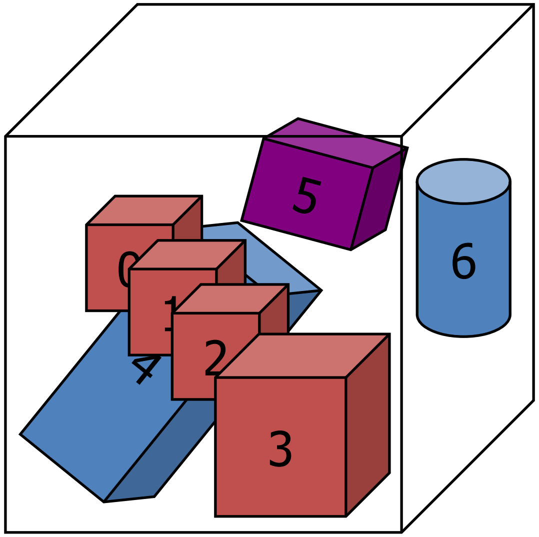
- Implemented as “special” kind of parameterised volumes:
  - G4PVDivision class derived from G4PVParameterised
  - But simpler to define as the parameterisation is calculated automatically using the values provided in input
- Similar to G4PVReplica
  - But it allows gaps in between mother and daughter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



mother volume

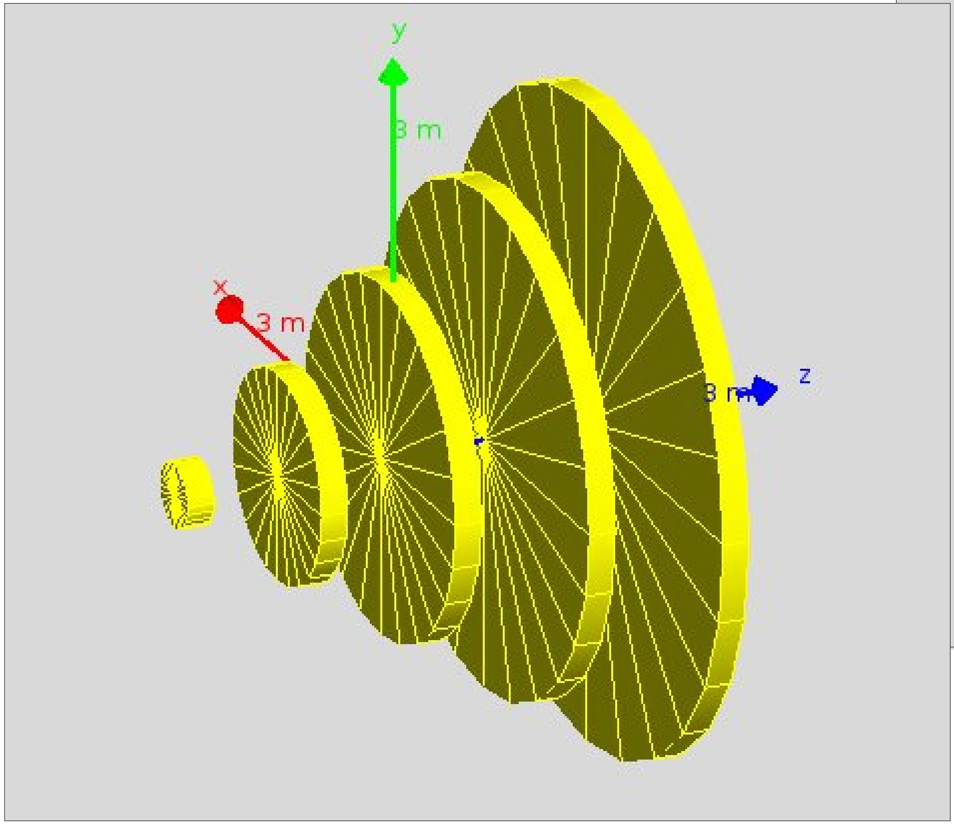
Parameterised Volumes

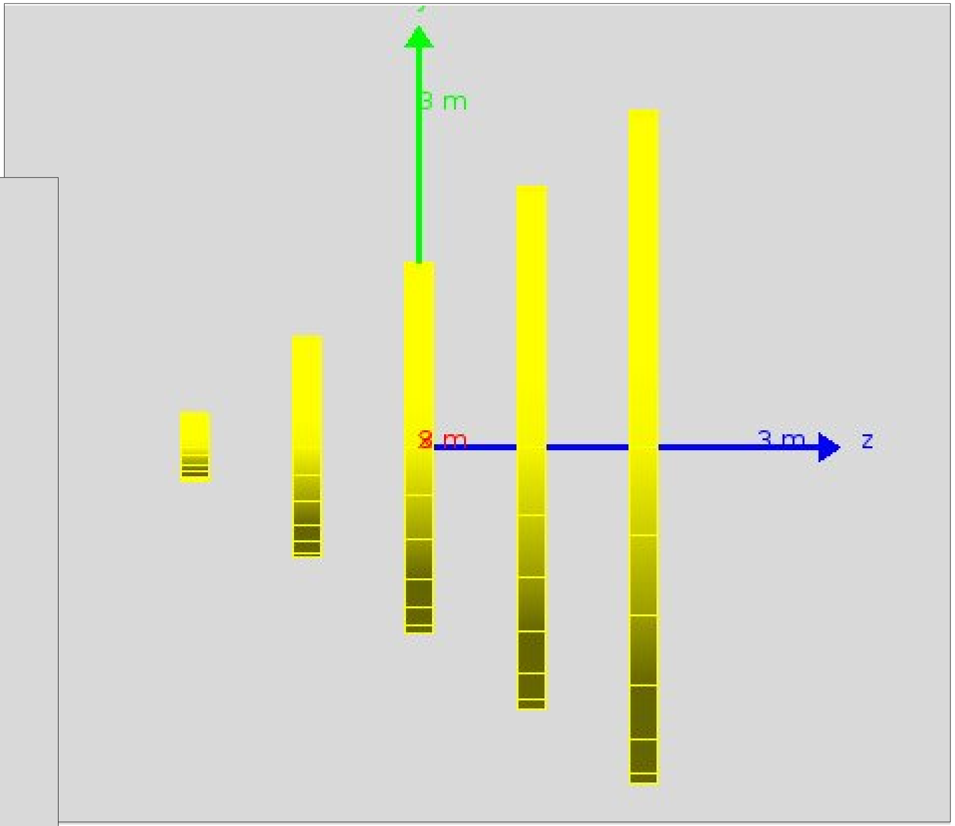
- Defined by G4PVParameterised
  - More general than G4PVReplica and G4PVDivisions
  - The properties of the “replicas” in their mother volume are defined in a user parameterisation class derived from G4VPVParameterisation
- The properties which must be always provided:
  - Where it is positioned (transformation, 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



Example

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



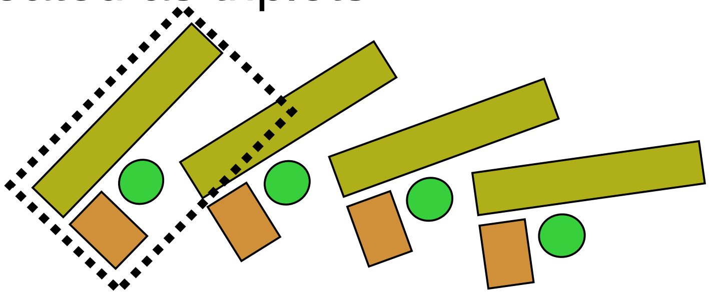


See the implementation details in backup slides

Special Techniques of Placements
Assemblies, Reflections

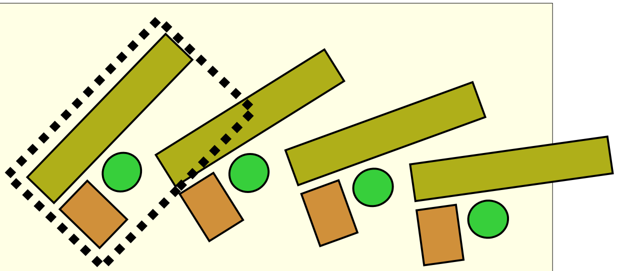
Grouping Volumes

- 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
  - G4AssemblyVolume class
  - 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

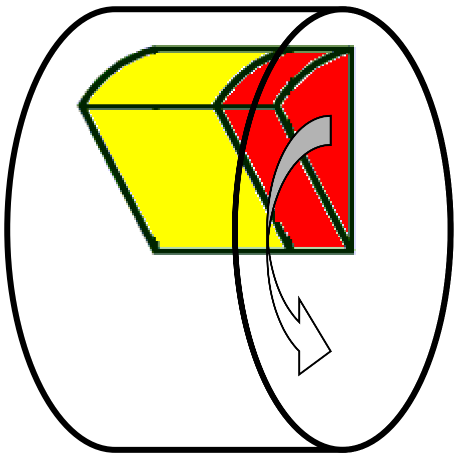


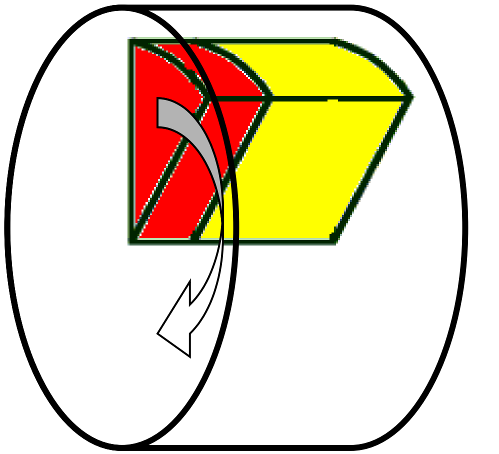
G4AssemblyVolume: Example

```
// 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);
}
```

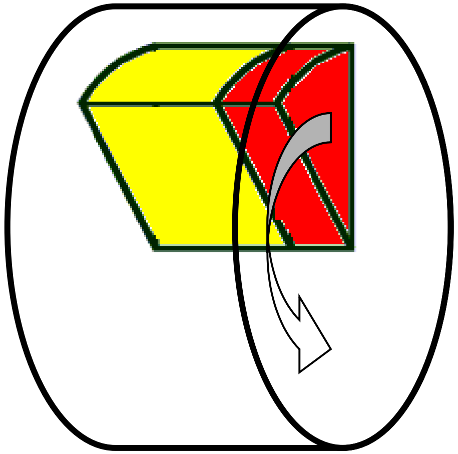


Reflecting volumes

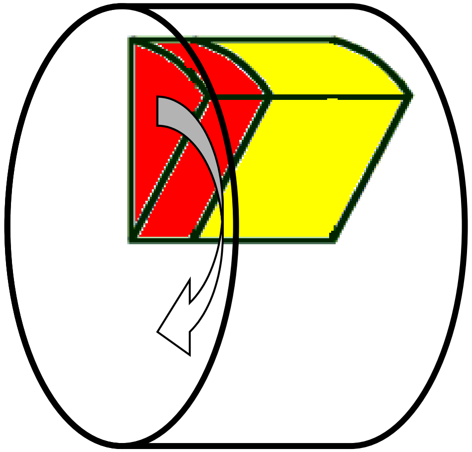




- Let's take an example of a pair of mirror symmetric volumes.
- Such geometry cannot be made by parallel transformation or 180 degree rotation.



Reflecting volumes



- Hierarchies of volumes based on CSG or specific solids can be reflected by means of G4ReflectionFactory
  - Singleton object using G4ReflectedSolid for generating placements of reflected volumes
  - The functions Place(..) or Replicate(..) should be used instead of G4PVPlacement or G4PVReplica when placing or replicating a volume in a volume hierarchy containing reflections
    - See more details in backup slides
- 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
  - Reflections can be applied to CSG and specific solids

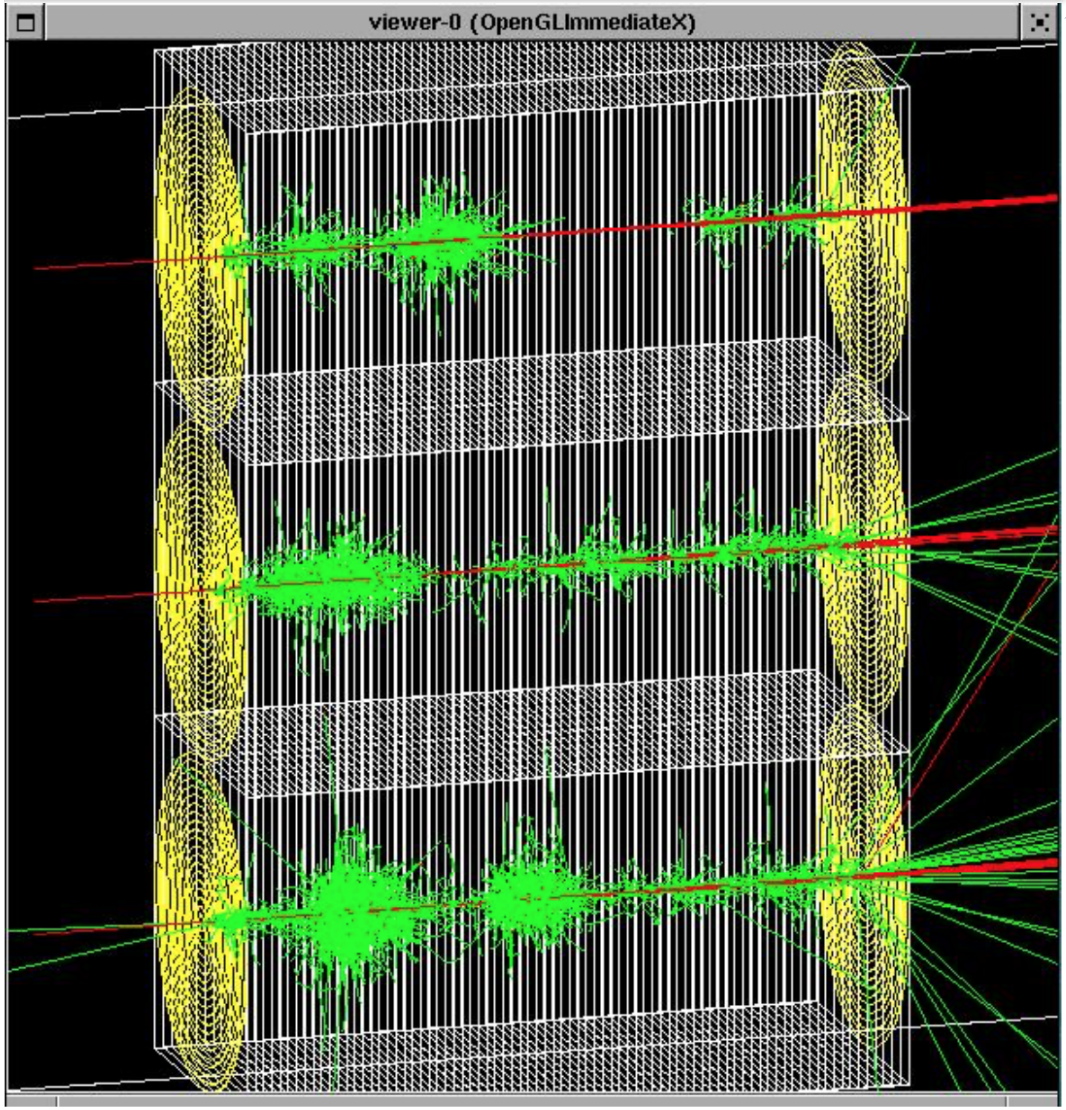
Parallel Geometries

Parallel Navigation

- Occasionally, it may not be straightforward to define attributes, like sensitivity, importance or envelope to be assigned to volumes in the mass geometry
  - Typically applicable to geometries imported from CAD, GDML, DICOM, etc.
- Parallel navigation functionality allows to define more than one overlapping geometry setups (worlds) simultaneously
  - The G4Transportation process can act on all worlds simultaneously
  - A step is limited not only by the boundary of the original mass geometry but also by the boundaries of each parallel geometry
  - Materials, production thresholds and EM field are used only from the mass geometry
  - In a parallel world, the user can define volumes in arbitrary manner with sensitivity, regions with shower parameterization, and/or importance field for biasing
  - Volumes in different worlds may overlap

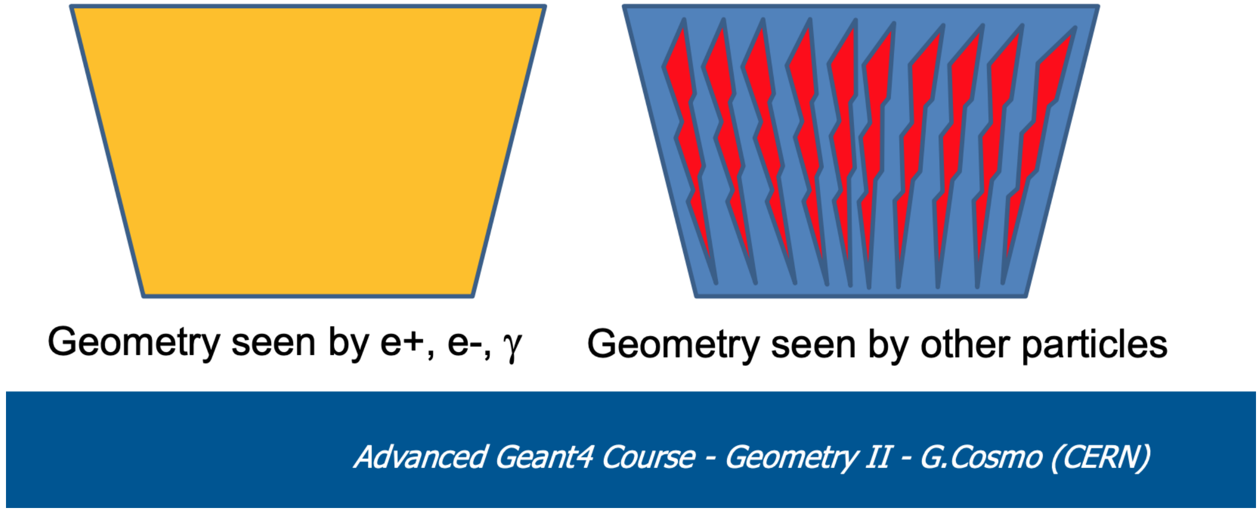
Example RE06

- extended/runAndEvent/RE06
- Mass geometry
  - Sandwich of rectangular absorbers and scintillators
- Parallel scoring geometry
  - Cylindrical layers



HEP use case

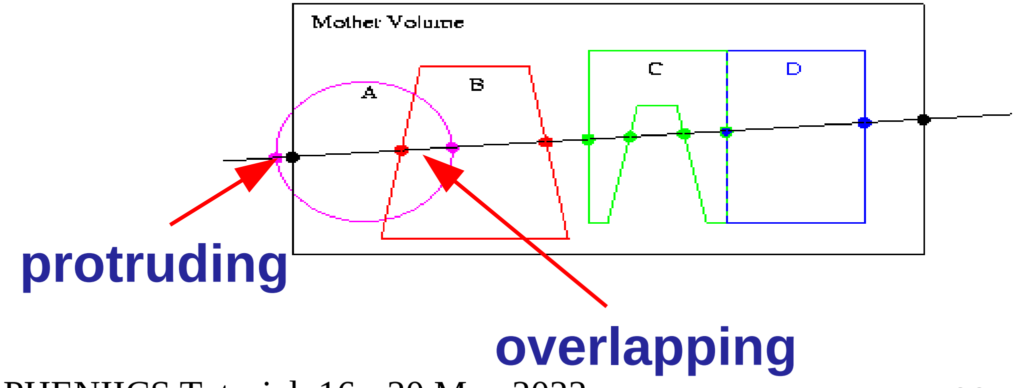
- A parallel world may be associated only to some limited types of particles
- May define geometries of different levels of detail for different particle types
- Example for **sampling calorimeter**: the mass world defines only the crude geometry with averaged material, while a parallel world with all the detailed geometry. Real materials in detailed parallel world geometry are associated with all particle types except e+, e- and gamma
  - e+, e- and gamma do not see volume boundaries defined in the parallel world, i.e. their steps won't be limited
- Shower parameterisations may have their own geometry



Geometry Checking Tools

Debugging Geometries

- A **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 (vis commands)



Optional Checks at Construction

- The option to check overlaps at geometry construction can be activated in the G4PVPlacement constructor :

```
G4PVPlacement(
  G4RotationMatrix* rotation,        // rotation
  const G4ThreeVector& translation, // translation
  G4LogicalVolume* currentLV,        // volume being placed
  const G4String& name,              // physical volume name
  G4LogicalVolume* motherLV,         // mother logical volume
  G4bool many,                       // not used
  G4int copyNumber,                  // position (copy) number
  G4bool surfaceCheck = false);      // option to activate
                                     // overlap checking
```

- Some number of points are then randomly sampled on the surface of creating volume.
- Each of these points are examined
  - If it is outside of the mother volume, or
  - If it is inside of already existing other volumes in the same mother volume.
- This check requires lots of CPU time, but it is worth to try at least once when you implement your geometry of some complexity.

Optional Checks at Construction - 2

- Alternatively, one can use explicitly the overlaps check for a simple physical volume:

```
G4bool CheckOverlaps(G4int res=1000, G4double tol=0.,
                     G4bool verbose=true, G4int errMax=1)
```

  - Verifies if the placed volume is overlapping with existing daughters or with the mother volume.
  - Returns true if the volume is overlapping.

Debugging Run-time Commands

- Verification of geometry for overlapping regions recursively through the volumes tree can be a run with a built-in run-time command:
  - `geometry/test/run`
- Volumes are recursively asked to verify for overlaps for points generated on the surface against their respective mother and volumes at the same level, performing for daughters and daughters of daughters etc.
  - It may take a very long time in complex geomteries
  - Parameters which can be tuned:
    - `recursion_start` – starting depth level (default 0)
    - `recursion_depth` – the total depth level for checking overlaps (default -1, which mean all levels)
    - `tolerance` – tolerance by which overlaps should not be `reported.`
    - `resolution` - the number of points on surface to be generated and checked for each volume (default is '10000')
    - `maximum_errors` - the threshold for the number of errors to be reported for a single volume (default is 1)

Debugging Run-time Commands (2)

- Example of a test output

```
GeomTest: no daughter volume extending outside mother detected.
GeomTest Error: Overlapping daughter volumes
    The volumes Tracker[0] and Overlap[0],
    both daughters of volume World[0],
    appear to overlap at the following points in global coordinates: (list truncated)
  length (cm)    ----- start position (cm) -----  ----- end position (cm) -----
    240            -240      -145.5      -145.5     0        -145.5      -145.5
Which in the mother coordinate system are:
  length (cm)    ----- start position (cm) -----  ----- end position (cm) -----
    . . .
Which in the coordinate system of Tracker[0] are:
  length (cm)    ----- start position (cm) -----  ----- end position (cm) -----
    . . .
Which in the coordinate system of Overlap[0] are:
  length (cm)    ----- start position (cm) -----  ----- end position (cm) -----
```

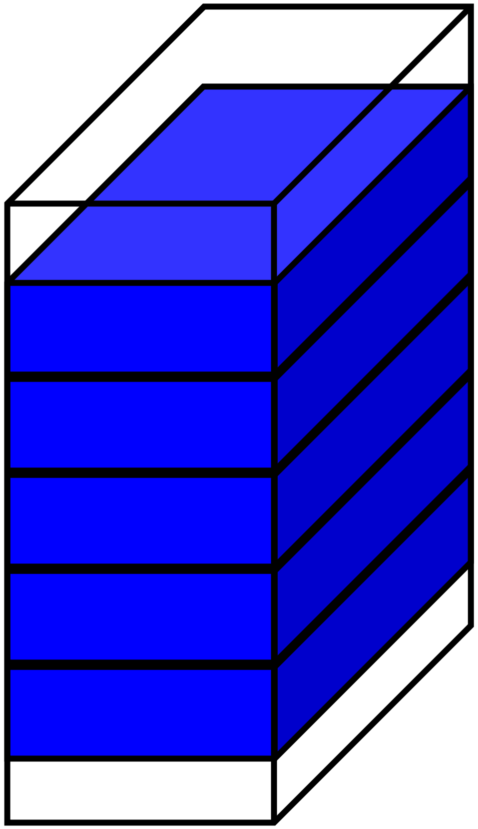
Summary

- Several classes can be used to define a repeated placement: G4PVReplica, G4PVDivisin and G4PVParameterisedVolume
- The volumes can be grouped together in a G4AssemblyVolume object and the whole group can be placed as a single “virtual” volume
- The volume hierarchies can be reflected using G4ReflectionFactory
- Geant4 does not allow for malformed geometries, neither protruding nor overlapping
  - Geometry checking tools are available to detect such cases

Backup

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



mother volume

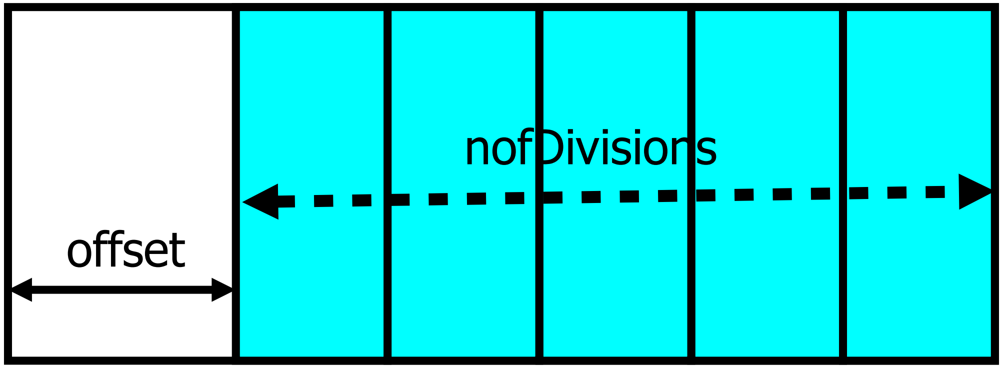
G4PVDivision (1)

- 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)

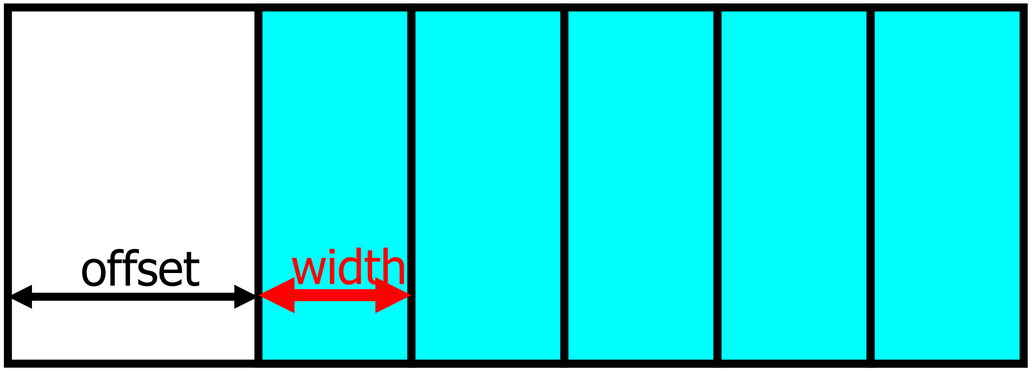
- Constructor 2:

```
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

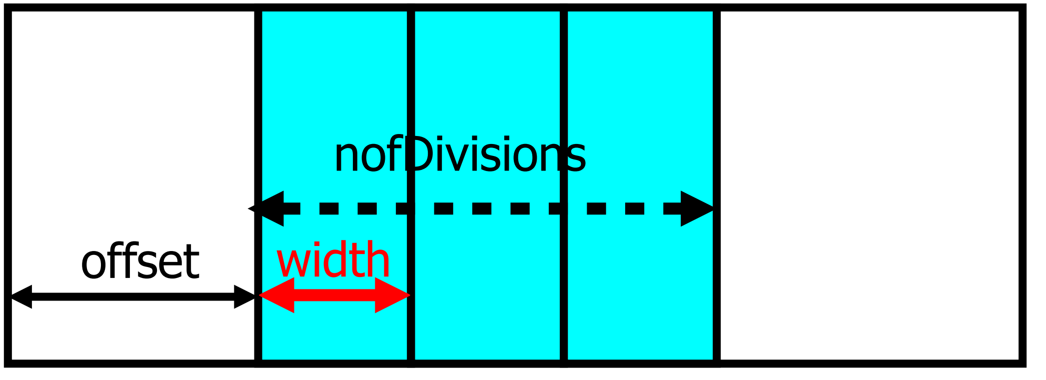


G4PVDivision (3)

- Constructor 3:

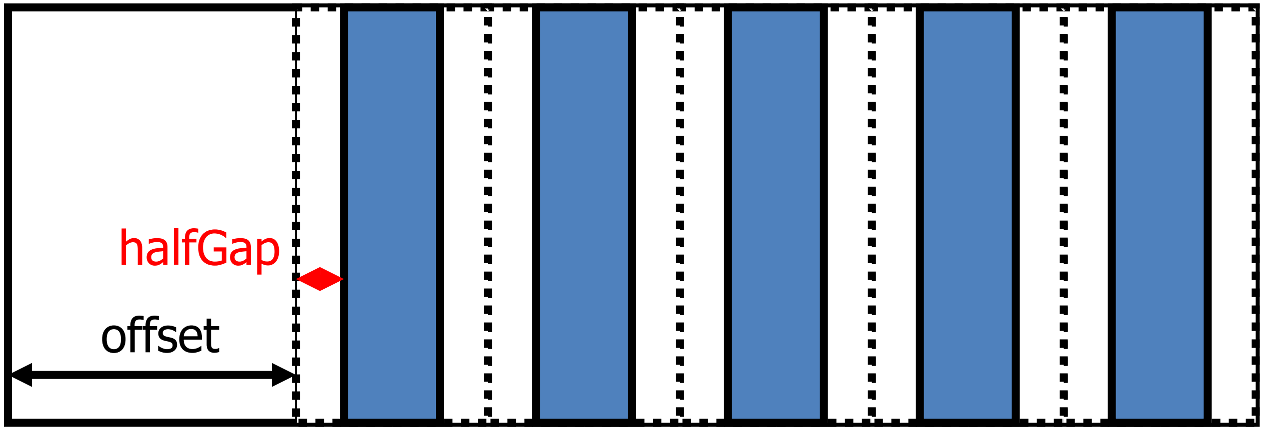
```
G4PVDivision(
   const G4String& name,         // physical volume name
   G4LogicalVolume* currentLV,  // volume being replicated
   G4LogicalVolume* motherLV,   // mother logical volume
   const EAxis axis,             // axis of replication
   const G4int nofDivisions,     // number of divisions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

- `nofDivisions` daughters of `width` thickness



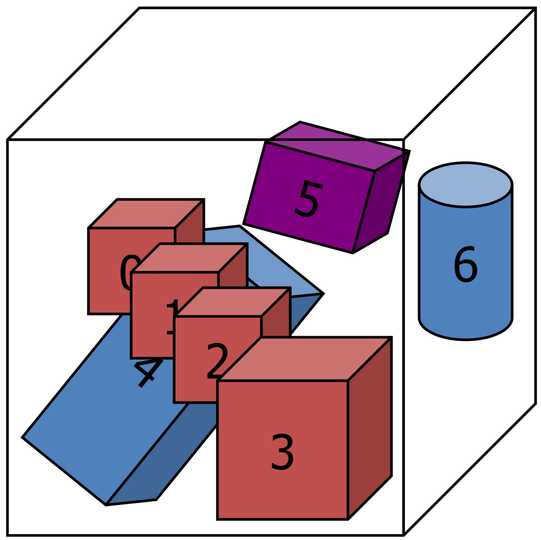
G4PVDivision (4)

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



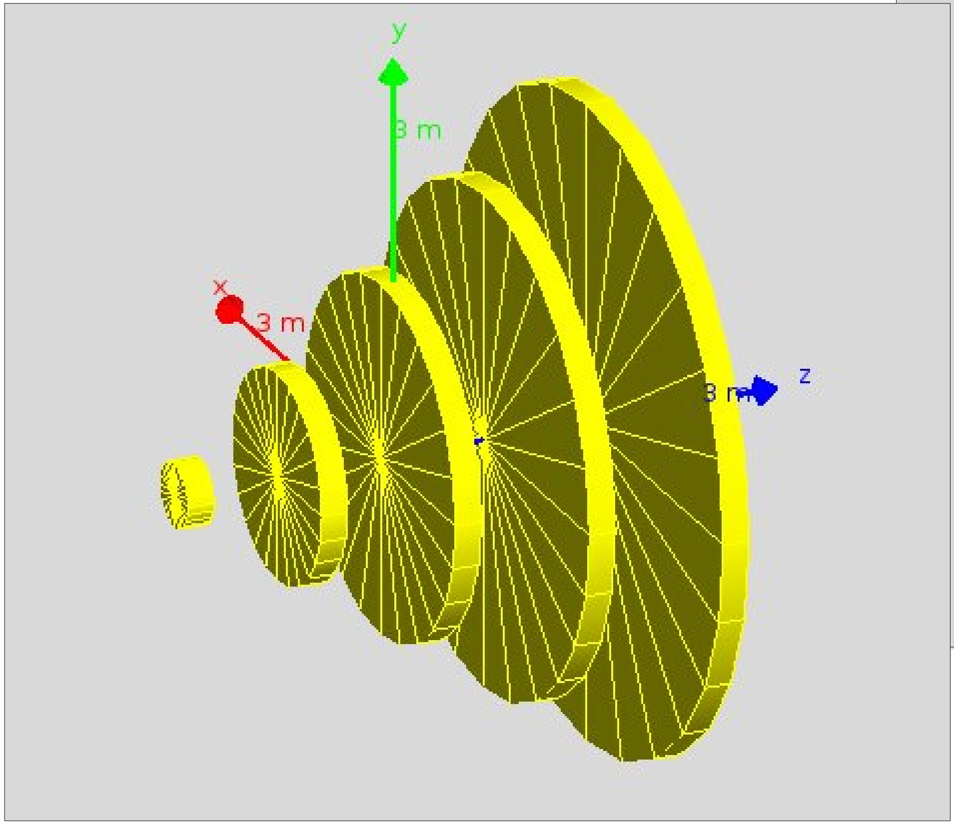
Parameterised Volumes

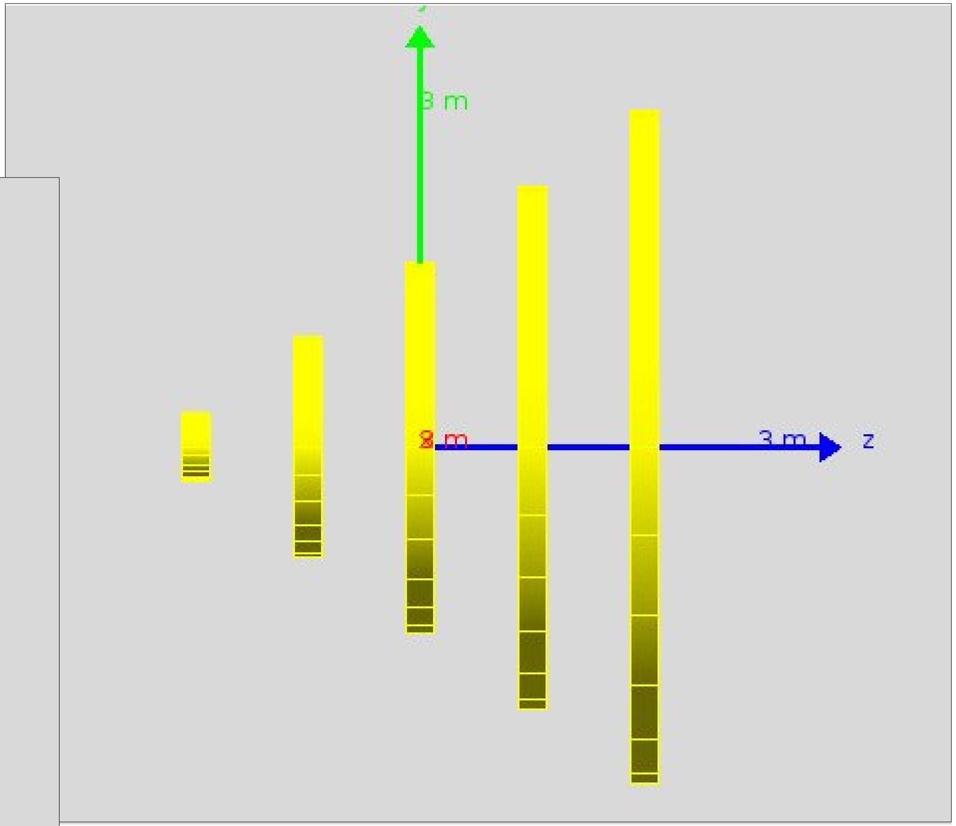
- Defined by G4PVParameterised
  - More general than G4PVReplica and G4PVDivisions
  - The properties of the “replicas” in their mother volume are defined in a user parameterisation class derived from G4VPVParameterisation
- The properties which must be always provided:
  - Where it is positioned (transformation, 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



Example

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





Six tracking chambers of increasing transverse size

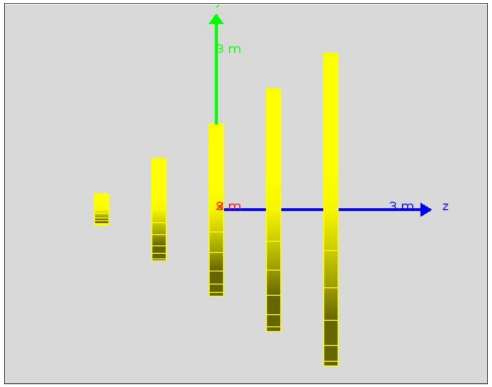
G4PVParameterised

- 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
```

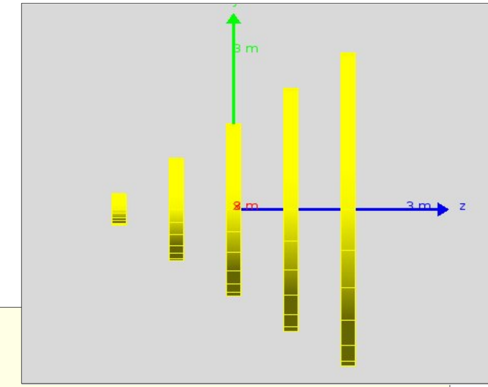
- Features and restrictions:
  - Replicates the volume `nofReplicas` times using the parameterisation `myParam`, within the mother volume
  - The positioning of the replicas is dominant along the specified Cartesian axis
    - If kUndefined is specified as axis, 3D voxelisation for optimisation of the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

MyParameterisation.hh



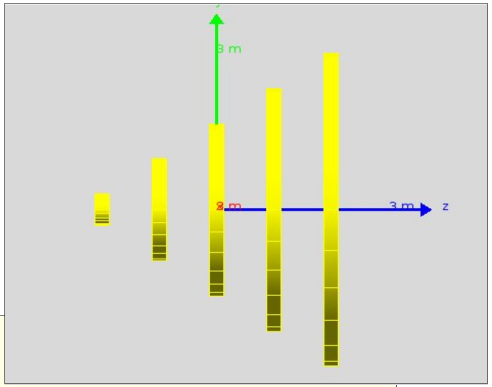
```
class B2bChamberParameterisation : public G4VPVParameterisation
{
  public:
    B2bChamberParameterisation(..);
    virtual ~B2bChamberParameterisation();

    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



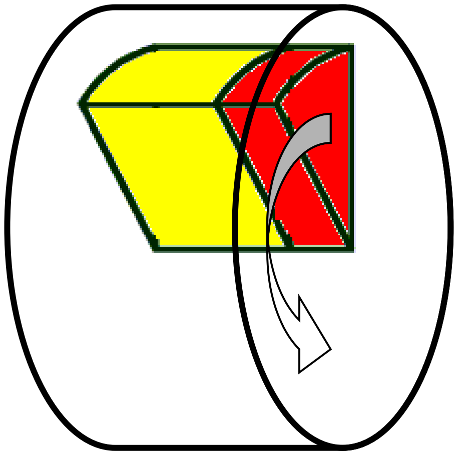
```
void B2bDetectorConstruction:Construct {
  ...
  G4Tubs* chamberS
    = new G4Tubs("tracker",0, 100*cm, 100*cm, 0.*deg, 360.*deg);
  fLogicChamber
    = new G4LogicalVolume(chamberS,fChamberMaterial,"Chamber",0,0,0);

  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
}
```

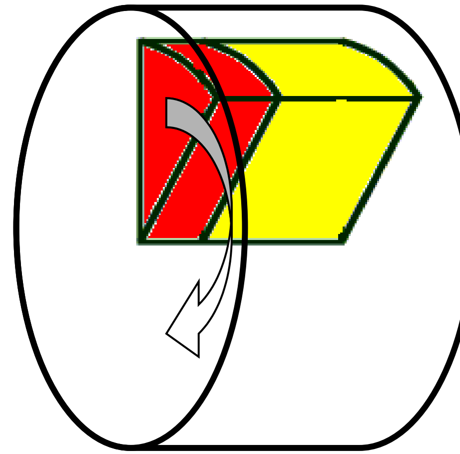
G4AssemblyVolume

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

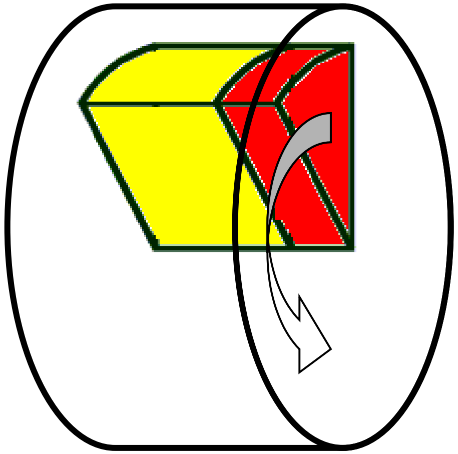
- Helper class to combine daughter logical volumes in arbitrary way
  - Imprints of the assembly volume are made inside a mother logical volume through G4AssemblyVolume::MakeImprint(...)
- Each physical volume name is generated automatically
  - Format: av_WWW_impr_XXX_YYY_ZZZ
    - WWW – assembly volume 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)

Reflecting volumes

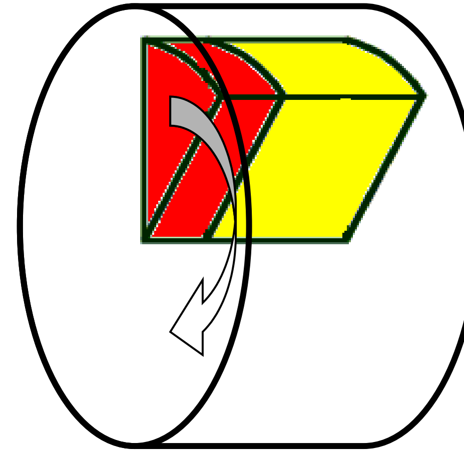




- Let's take an example of a pair of mirror symmetric volumes.
- Such geometry cannot be made by parallel transformation or 180 degree rotation.



Reflecting volumes



- Hierarchies of volumes based on CSG or specific solids can be reflected by means of G4ReflectionFactory
  - Singleton object using G4ReflectedSolid for generating placements of reflected volumes
  - The functions Place(..) or Replicate(..) should be used instead of G4PVPlacement or G4PVReplica when placing or replicating a volume in a volume hierarchy containing reflections
    - See more details in backup slides
- 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
  - 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