

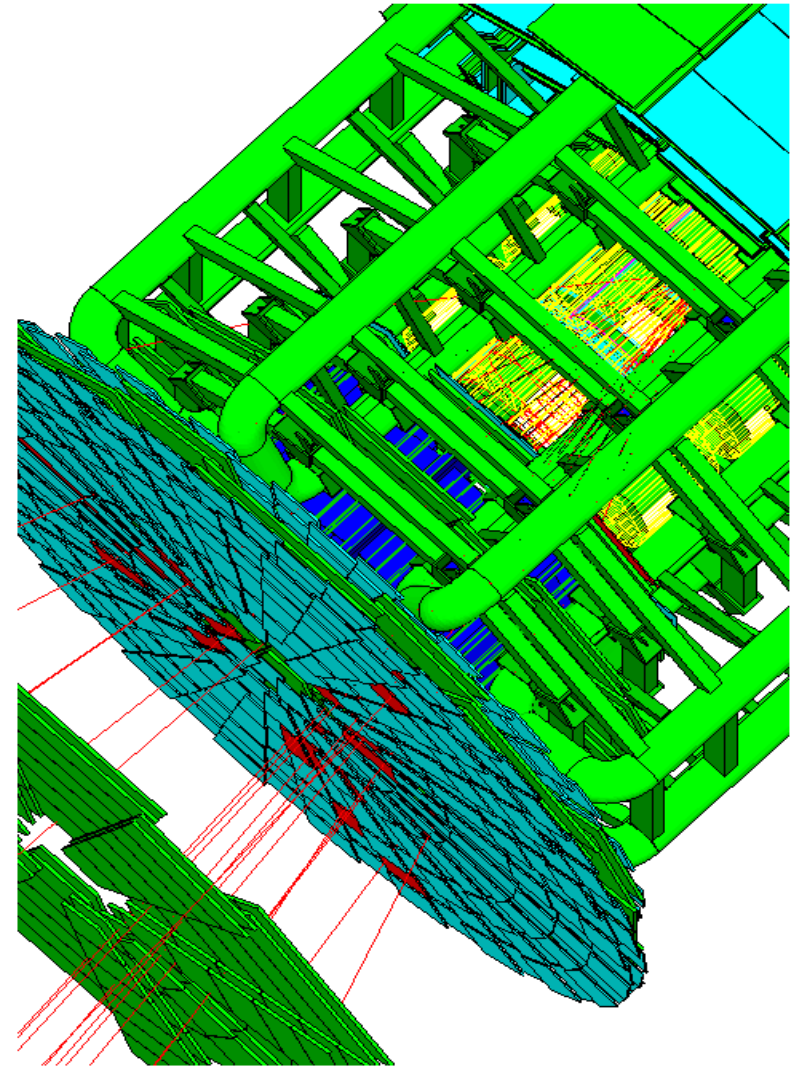
GEANT4
A SIMULATION TOOLKIT

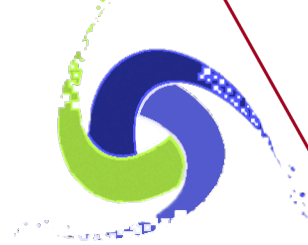
Version 10.5

Geometry II

Makoto Asai (SLAC)
Geant4 Tutorial Course

- Various ways of placing volumes
 - Simple placement volume
 - Parameterized volume
 - Replicated volume
 - Divided volume
 - Nested parameterization
 - Reflected volume
 - Assembly volume
- Touchable





GEANT4
A SIMULATION TOOLKIT

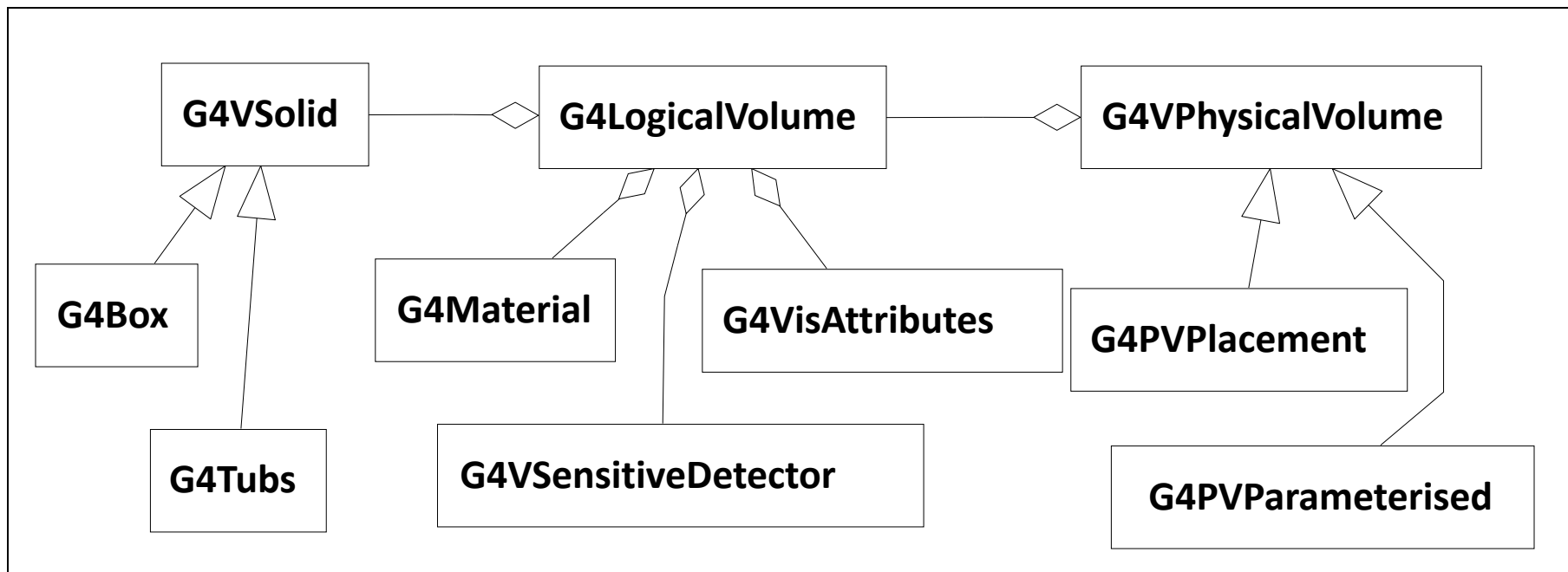
Version 10.5

Physical volume



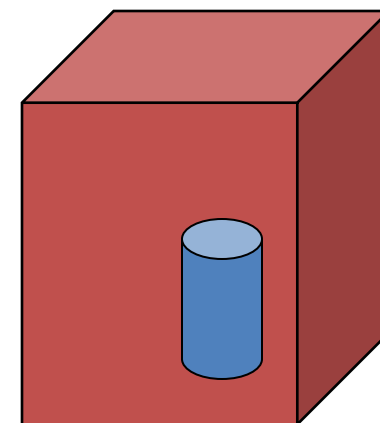
Detector geometry

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

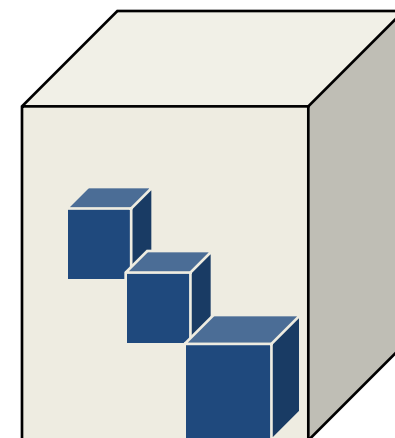


Physical Volumes

- Placement volume : it is one positioned volume
 - One physical volume object represents one “real” volume.
- Repeated volume : a volume placed many times
 - One physical volume object represents any number of “real” volumes.
 - reduces use of memory.
 - Parameterised
 - repetition w.r.t. copy number
 - Replica and Division
 - simple repetition along one axis
- A mother volume can contain **either**
 - many placement volumes
 - **or**, one repeated volume



placement



repeated

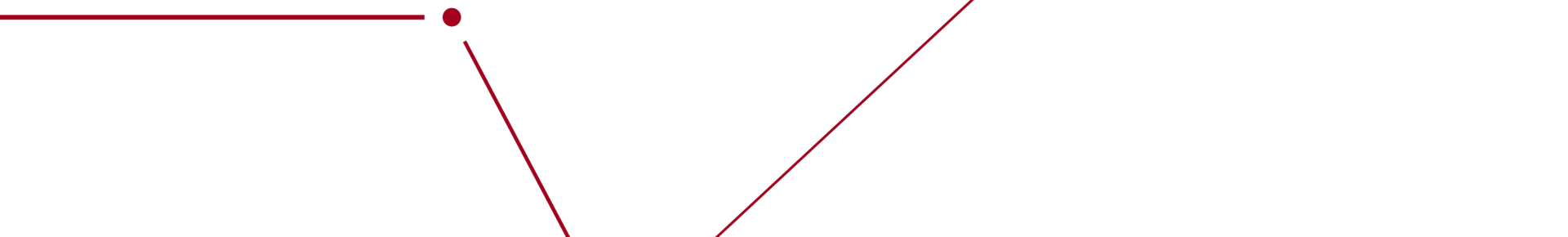
- **G4PVPlacement** 1 Placement = One **Placement Volume**
 - A volume instance positioned once in its mother volume
- **G4PVParameterised** 1 Parameterized = Many **Repeated Volumes**
 - Parameterized by the copy number
 - Shape, size, material, sensitivity, vis attributes, position and rotation can be parameterized by the **copy number**.
 - You have to implement a concrete class of **G4VPVParameterisation**.
 - Reduction of memory consumption
 - Currently: parameterization can be used only for volumes that either
 - a) have no further daughters, or
 - b) are identical in size & shape (so that grand-daughters are safely fit inside).
 - By implementing **G4PVNestedParameterisation** instead of **G4VPVParameterisation**, material, sensitivity and vis attributes can be parameterized by the copy numbers of ancestors.

- **G4PVReplica** 1 Replica = Many **Repeated Volumes**
 - Daughters of same shape are aligned along one axis
 - Daughters fill the mother completely without gap in between.
- **G4PVDivision** 1 Division = Many **Repeated Volumes**
 - Daughters of same shape are aligned along one axis and fill the mother.
 - There can be gaps between mother wall and outmost daughters.
 - No gap in between daughters.
- **G4ReflectionFactory** 1 Placement = a **pair** of **Placement volumes**
 - generating placements of a volume and its reflected volume
 - Useful typically for end-cap calorimeter
- **G4AssemblyVolume** 1 Placement = a set of **Placement volumes**
 - Position a group of volumes



Version 10.5

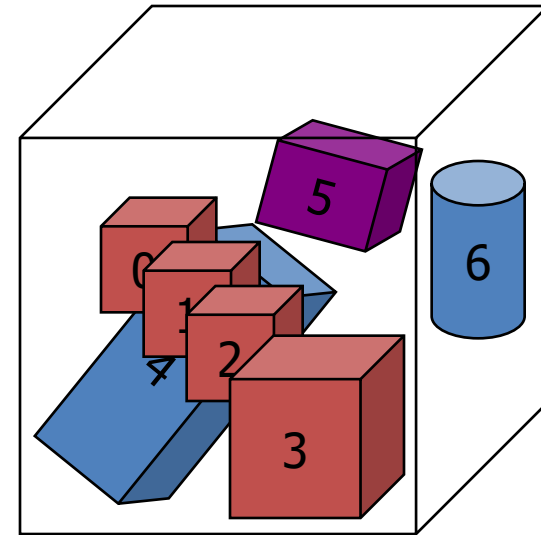
Parameterized volume




```
G4PVPParameterised(const G4String& pName,  
                    G4LogicalVolume* pLogical,  
                    G4LogicalVolume* pMother,  
                    const EAxis pAxis,  
                    const G4int nReplicas,  
                    G4VPVParameterisation* pParam  
                    G4bool pSurfChk=false);
```

- Replicates the volume **nReplicas** times using the parameterization **pParam**, within the mother volume **pMother**
- **pAxis** is a “suggestion” to the navigator along which Cartesian axis replication of parameterized volumes dominates.
 - **kXAxis**, **kYAxis**, **kZAxis** : one-dimensional optimization
 - **kUndefined** : three-dimensional optimization

- User should implement a class derived from **G4VPVParameterisation** abstract base class and define following **as a function of copy number**
 - where it is positioned (transformation, rotation)
- Optional:
 - the size of the solid (dimensions)
 - the type of the solid, material, sensitivity, vis attributes
- All daughters must be fully contained in the mother.
- Daughters should not overlap to each other.
- Limitations:
 - Applies to simple CSG solids only
 - Granddaughter volumes allowed only for special cases
 - Consider parameterised volumes as “leaf” volumes
- Typical use-cases
 - Complex detectors
 - with large repetition of volumes, regular or irregular
 - Medical applications
 - the material in animal tissue is measured as cubes with varying material



```
G4VSolid* solidChamber =  
    new G4Box("chamber", 100*cm, 100*cm, 10*cm);  
  
G4LogicalVolume* logicChamber =  
    new G4LogicalVolume  
        (solidChamber, ChamberMater, "Chamber", 0, 0, 0);  
  
G4VPVParameterisation* chamberParam =  
    new ChamberParameterisation();  
  
G4VPhysicalVolume* physChamber =  
    new G4PVParameterised("Chamber", logicChamber,  
        logicMother, kZAxis, NbOfChambers, chamberParam);
```

G4VPVParameterisation : example

```
class ChamberParameterisation : public G4VPVParameterisation
{
public:
    ChamberParameterisation();
    virtual ~ChamberParameterisation();
    virtual void ComputeTransformation // position, rotation
        (const G4int copyNo, G4VPhysicalVolume* physVol) const;
    virtual void ComputeDimensions // size
        (G4Box& trackerLayer, const G4int copyNo,
         const G4VPhysicalVolume* physVol) const;
    virtual G4VSolid* ComputeSolid // shape
        (const G4int copyNo, G4VPhysicalVolume* physVol);
    virtual G4Material* ComputeMaterial // material, sensitivity, visAtt
        (const G4int copyNo, G4VPhysicalVolume* physVol,
         const G4VTouchable *parentTouch=0);
        // G4VTouchable should not be used for ordinary parameterization
};
```

G4VPVParameterisation : example

```
void ChamberParameterisation::ComputeTransformation
(const G4int copyNo, G4VPhysicalVolume* physVol) const
{
    G4double Xposition = ... // w.r.t. copyNo
    G4ThreeVector origin(Xposition, Yposition, Zposition);
    physVol->SetTranslation(origin);
    physVol->SetRotation(0);
}
```

```
void ChamberParameterisation::ComputedDimensions
(G4Box& trackerChamber, const G4int copyNo,
const G4VPhysicalVolume* physVol) const
{
    G4double XhalfLength = ... // w.r.t. copyNo
    trackerChamber.SetXHalfLength(XhalfLength);
    trackerChamber.SetYHalfLength(YhalfLength);
    trackerChamber.SetZHalfLength(ZhalfLength);
}
```

G4VPVParameterisation : example

```
G4VSolid* ChamberParameterisation::ComputeSolid
    (const G4int copyNo, G4VPhysicalVolume* physVol)
{
    G4VSolid* solid;
    if(copyNo == ...) solid = myBox;
    else if(copyNo == ...) solid = myTubs;
    ...
    return solid;
}

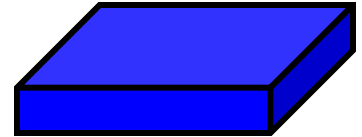
G4Material* ComputeMaterial // material, sensitivity, visAtt
    (const G4int copyNo, G4VPhysicalVolume* physVol,
     const G4VTouchable *parentTouch=0);
{
    G4Material* mat;
    if(copyNo == ...)
    {
        mat = material1;
        physVol->GetLogicalVolume()->SetVisAttributes( att1 );
    }
    ...
    return mat;
}
```



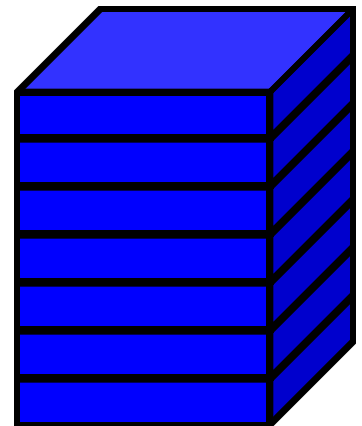
Version 10.5

Replicated volume

- The mother volume is **completely filled** with replicas, all of which are the **same size (width)** and **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

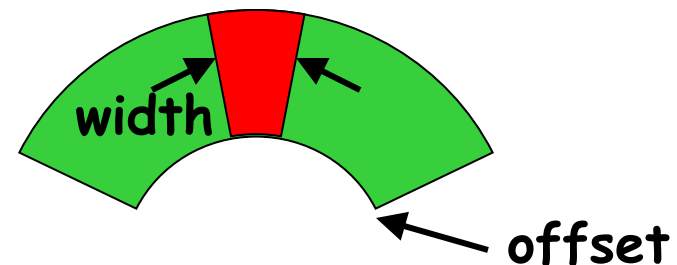
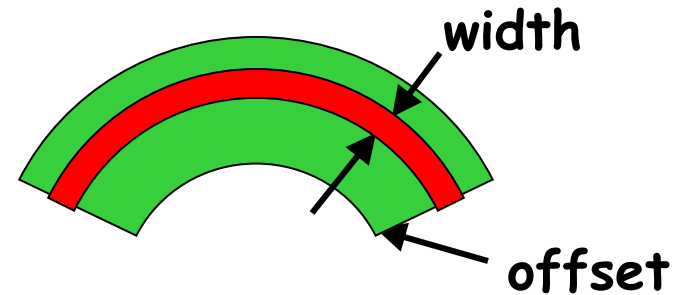
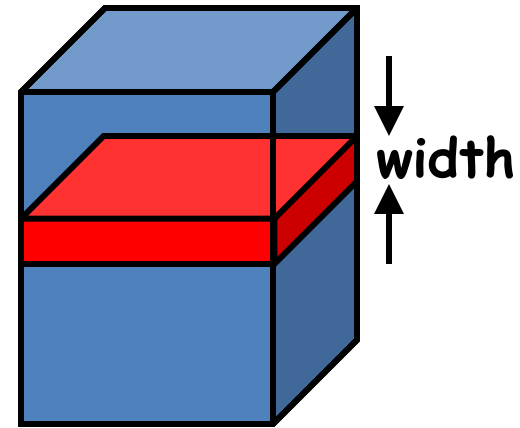


mother volume


```
G4PVReplica(const G4String &pName,  
            G4LogicalVolume *pLogical,  
            G4LogicalVolume *pMother,  
            const EAxis pAxis,  
            const G4int nReplicas,  
            const G4double width,  
            const G4double offset=0.);
```

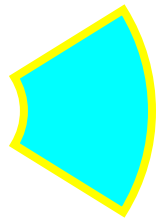
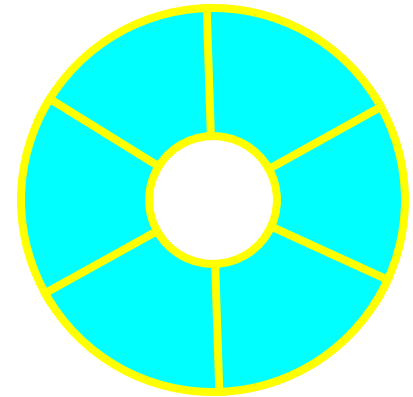
- `offset` may be used only for tube/cone segment
- Features and restrictions:
 - Replicas can be placed inside other replicas
 - Normal placement volumes can be placed inside replicas, assuming no intersection/overlaps with the mother volume or with other replicas
 - No volume can be placed inside a **radial** replication
 - Parameterised volumes **cannot** be placed inside a replica

- Cartesian axes - **kXaxis**, **kYaxis**, **kZaxis**
 - Center of n-th daughter is given as
 $-\text{width} * (\text{nReplicas} - 1) * 0.5 + \text{n} * \text{width}$
 - Offset shall not be used
- Radial axis - **kRaxis**
 - Center of n-th daughter is given as
 $\text{width} * (\text{n} + 0.5) + \text{offset}$
 - Offset must be the inner radius of the mother
- Phi axis - **kPhi**
 - Center of n-th daughter is given as
 $\text{width} * (\text{n} + 0.5) + \text{offset}$
 - Offset must be the starting angle of the mother



G4PVReplica : example

```
G4double tube_dPhi = 2.* M_PI * rad;
G4VSolid* tube =
    new G4Tubs("tube",20*cm,50*cm,30*cm,0.,tube_dPhi);
G4LogicalVolume * tube_log =
    new G4LogicalVolume(tube, Air, "tubeL", 0, 0, 0);
G4VPhysicalVolume* tube_phys =
    new G4PVPlacement(0,G4ThreeVector(-200.*cm,0.,0.),
        "tubeP", tube_log, world_phys, false, 0);
G4double divided_tube_dPhi = tube_dPhi/6.;
G4VSolid* div_tube =
    new G4Tubs("div_tube", 20*cm, 50*cm, 30*cm,
        -divided_tube_dPhi/2., divided_tube_dPhi);
G4LogicalVolume* div_tube_log =
    new G4LogicalVolume(div_tube,Pb,"div_tubeL",0,0,0);
G4VPhysicalVolume* div_tube_phys =
    new G4PVReplica("div_tube_phys", div_tube_log,
        tube_log, kPhi, 6, divided_tube_dPhi);
```

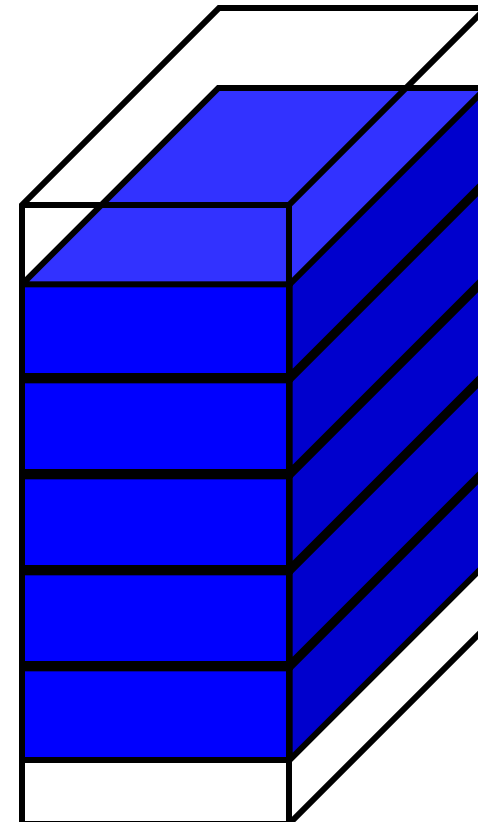




Version 10.5

Divided volume

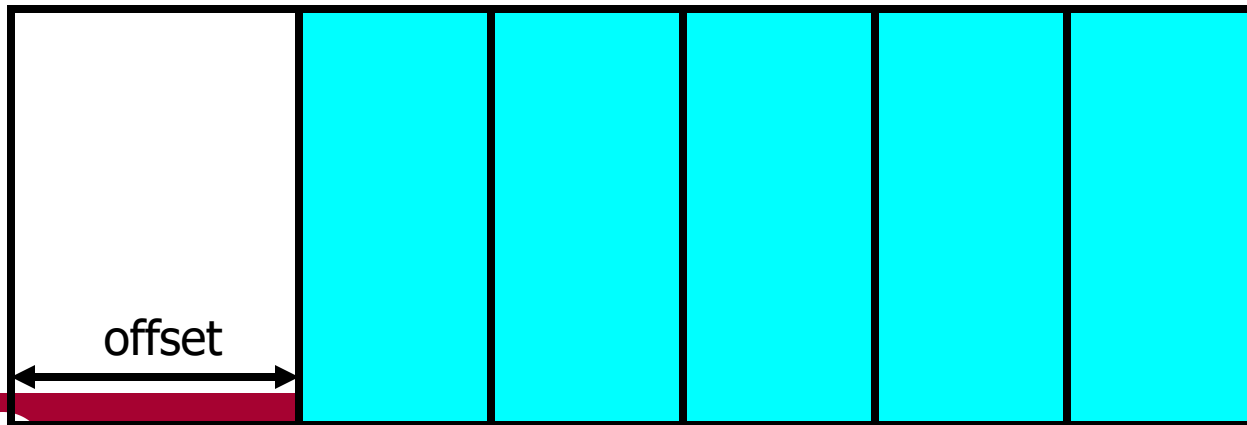
- G4PVDivision is a special kind of G4PVParameterised.
 - G4VPVParameterisation is **automatically generated** according to the parameters given in G4PVDivision.
- G4PVDivision is similar to G4PVReplica but
 - It currently **allows gaps in between** mother and daughter volumes
 - We are extending G4PVDivision to allow gaps between daughters, and also gaps on side walls. We plan to release this extension in near future.
- **Shape of all daughter volumes must be same shape as the mother volume.**
 - G4VSolid (to be assigned to the daughter logical volume) must be the same type, but different object.
- **Replication must be aligned along one axis.**
- If your geometry does not have gaps, use **G4Replica**.
 - For identical geometry, navigation of G4Replica is faster.



mother volume

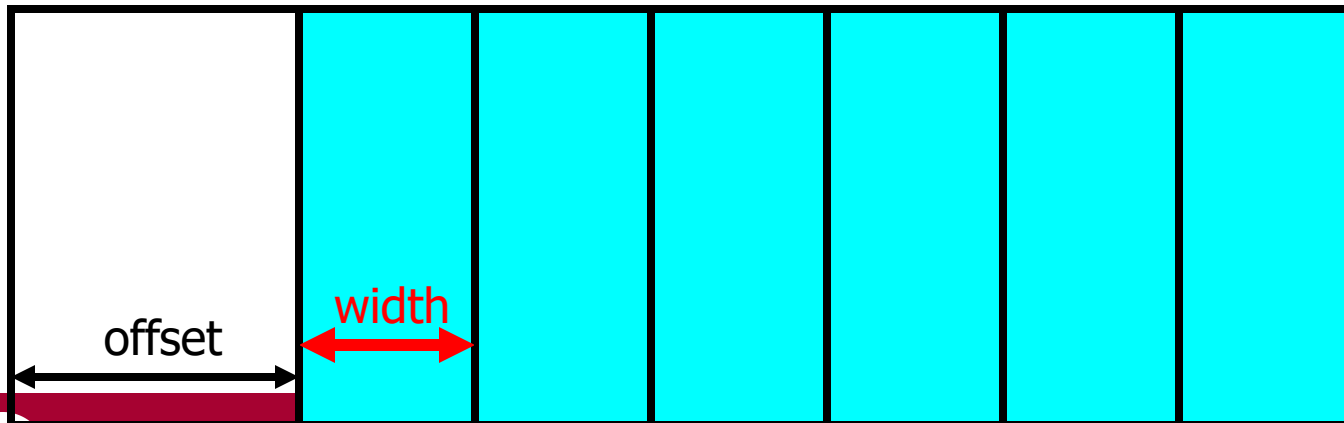
```
G4PVDivision(const G4String& pName,  
             G4LogicalVolume* pDaughterLogical,  
             G4LogicalVolume* pMotherLogical,  
             const EAxis pAxis,  
             const G4int nDivisions, // number of division is given  
             const G4double offset);
```

- The size (width) of the daughter volume is calculated as
 $(\text{size of mother} - \text{offset}) / \text{nDivisions}$



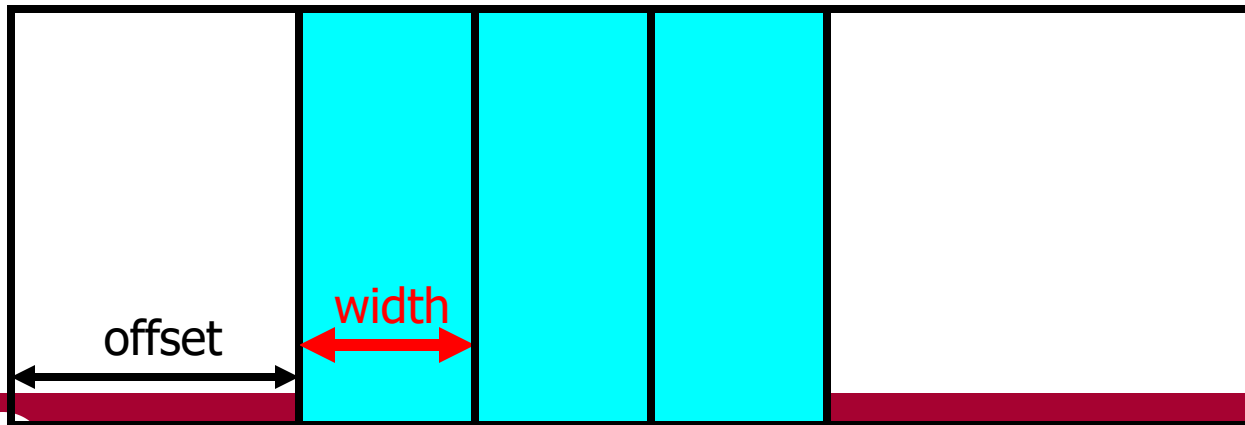
```
G4PVDivision(const G4String& pName,  
             G4LogicalVolume* pDaughterLogical,  
             G4LogicalVolume* pMotherLogical,  
             const EAxis pAxis,  
             const G4double width, // width of daughter volume is given  
             const G4double offset);
```

- The number of daughter volumes is calculated as
`int(((size of mother) - offset) / width)`
– As many daughters as width and offset allow



```
G4PVDivision(const G4String& pName,  
             G4LogicalVolume* pDaughterLogical,  
             G4LogicalVolume* pMotherLogical,  
             const EAxis pAxis,  
             const G4int nDivisions,  
             const G4double width, // both number of division and width are given  
             const G4double offset);
```

- *nDivisions* daughters of *width* thickness



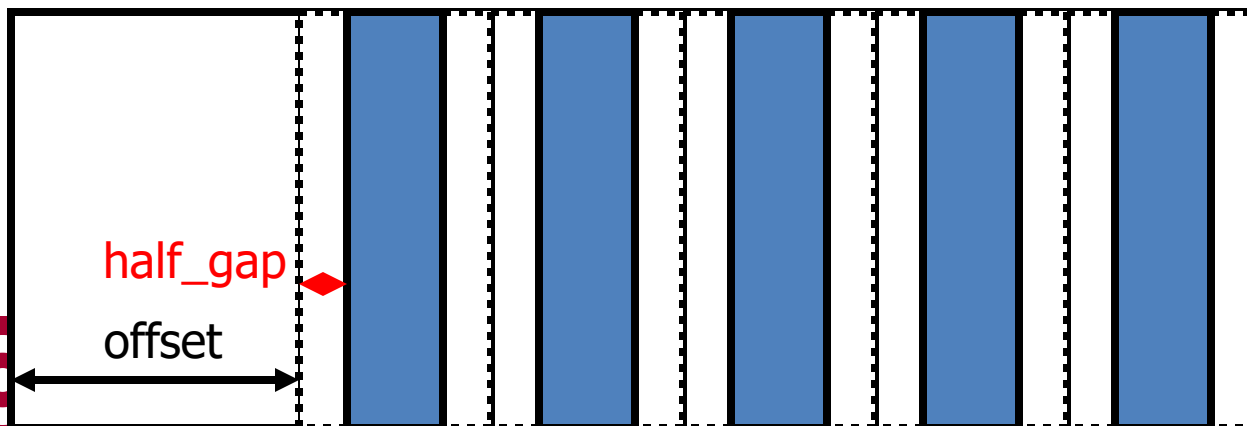
- G4PVDivision currently supports following shapes / axes.
 - G4Box : kXAxis, kYAxis, kZAxis
 - G4Tubs : kRho, kPhi, kZAxis
 - G4Cons : kRho, kPhi, kZAxis
 - G4Trd : kXAxis, kYAxis, kZAxis
 - G4Para : kXAxis, kYAxis, kZAxis
 - G4Polycone : kRho, kPhi, kZAxis
 - kZAxis - the number of divisions has to be the same as solid sections, (i.e. numZPlanes-1), the width will **not** be taken into account.
 - G4Polyhedra : kRho, kPhi, kZAxis
 - kPhi - the number of divisions has to be the same as solid sides, (i.e. numSides), the width will **not** be taken into account.
 - kZAxis - the number of divisions has to be the same as solid sections, (i.e. numZPlanes-1), the width will **not** be taken into account.
- In the case of division along kRho of G4Cons, G4Polycone, G4Polyhedra, if width is provided, it is taken as the width at the -Z radius; the width at other radii will be scaled to this one.

- New extension of G4Division introduced with version 9.4.
- It allows gaps in between divided volumes.

```
G4PVDivision(const G4String& pName, G4LogicalVolume* pDaughterLogical,  
             G4LogicalVolume* pMotherLogical, const EAxis pAxis,  
             const G4int nDivisions, const G4double half_gap, const G4double offset);
```

```
G4PVDivision(const G4String& pName, G4LogicalVolume* pDaughterLogical,  
             G4LogicalVolume* pMotherLogical, const EAxis pAxis,  
             const G4double width, const G4double half_gap, const G4double offset);
```

```
G4PVDivision(const G4String& pName, G4LogicalVolume* pDaughterLogical,  
             G4LogicalVolume* pMotherLogical, const EAxis pAxis,  
             const G4int nDivisions, const G4double width,  
             const G4double half_gap, const G4double offset);
```



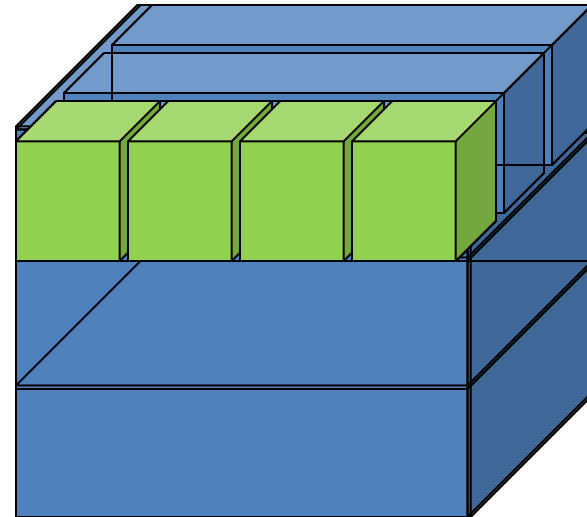
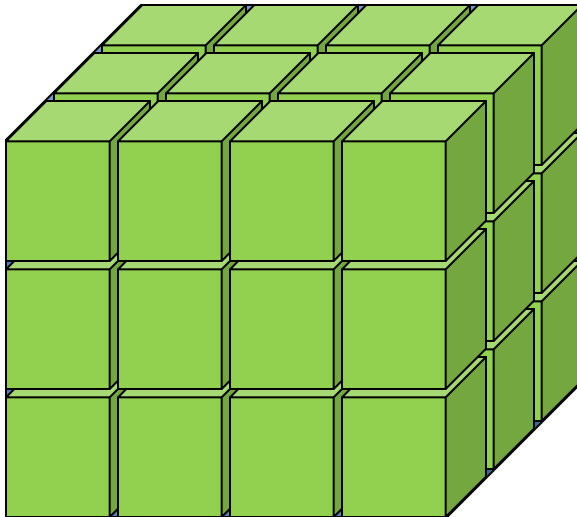


Version 10.5

Nested parameterization

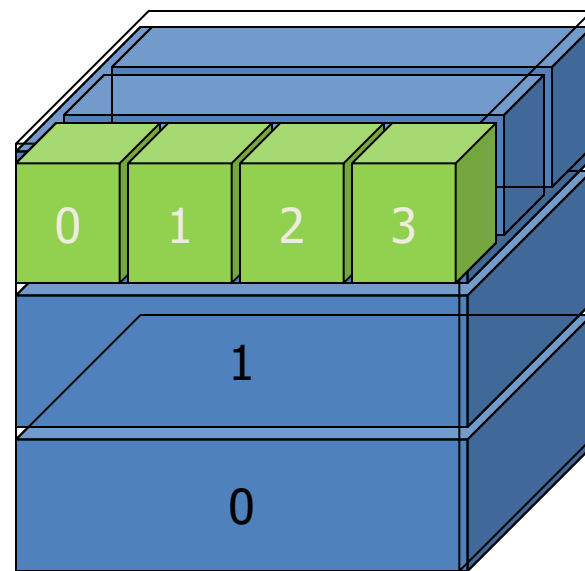
Nested parameterization

- ▶ Suppose your geometry has three-dimensional regular repetition of same shape and size of volumes without gap between volumes. And material of such volumes are changing according to the position.
 - ▶ E.g. voxels made by CT Scan data (DICOM)
- ▶ Instead of direct three-dimensional parameterized volume, use replicas for the first and second axes sequentially, and then use one-dimensional parameterization along the third axis.



- ▶ It requires much less memory for geometry optimization and gives much faster navigation for ultra-large number of voxels.

- ▶ Given geometry is defined as two sequential replicas and then one-dimensional parameterization,
 - ▶ Material of a voxel must be parameterized not only by the copy number of the voxel, but also by the copy numbers of ancestors.
 - ▶ Material is indexed by three indices.
- ▶ **G4VNestedParameterisation** is a special parameterization class derived from G4VPVParameterisation base class.
 - ▶ ComputeMaterial() method of **G4VNestedParameterisation** has a touchable object of the **parent** physical volume, in addition to the copy number of the voxel.
 - ▶ Index of first axis = `theTouchable->GetCopyNumber(1);`
 - ▶ Index of second axis = `theTouchable->GetCopyNumber(0);`
 - ▶ Index of third axis = copy number



- G4VNestedParameterisation is derived from G4VPVParameterization.
- G4VNestedParameterisation class has three **pure virtual** methods you have to implement,
 - in addition to ComputeTransformation() method, which is mandatory for all G4VPVParameterization classes.

```
virtual G4Material* ComputeMaterial(G4VPhysicalVolume *currentVol,  
    const G4int repNo, const G4VTouchable *parentTouch=0)=0;
```

- Return a material pointer w.r.t. copy numbers of itself and ancestors.
- Must cope with parentTouch=0 for navigator's sake. Typically, return a default material if parentTouch=0.

```
virtual G4int GetNumberOfMaterials() const=0;
```

- Return total number of materials which may appear as the return value of ComputeMaterial() method.

```
virtual G4Material* GetMaterial(G4int idx) const=0;
```

- Return idx-th material.
- “idx” is not a copy number. idx = [0, nMaterial-1]

- G4VNestedParameterisation is a kind of G4VPVParameterization.
 - It can be used as an argument of G4PVPParameterised.
 - All other arguments of G4PVPParameterised are unaffected.
- Nested parameterization of placement volume is **not** supported.
 - All levels used as indices of material must be **repeated volume**.
There cannot be a level of placement volume in between.

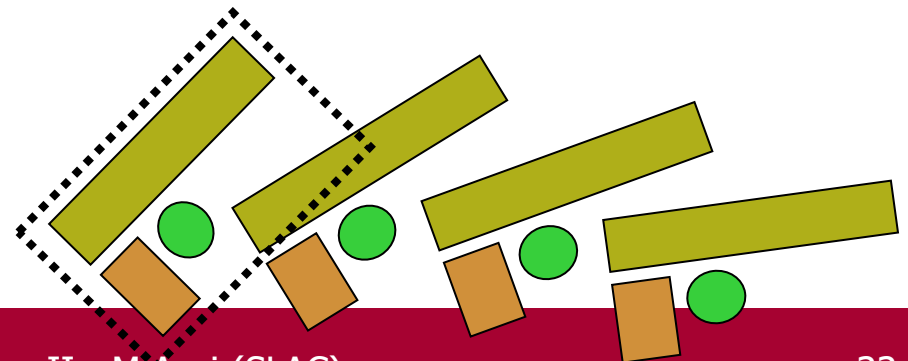


Version 10.5

Assembly volume

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::AddPlacedVolume`

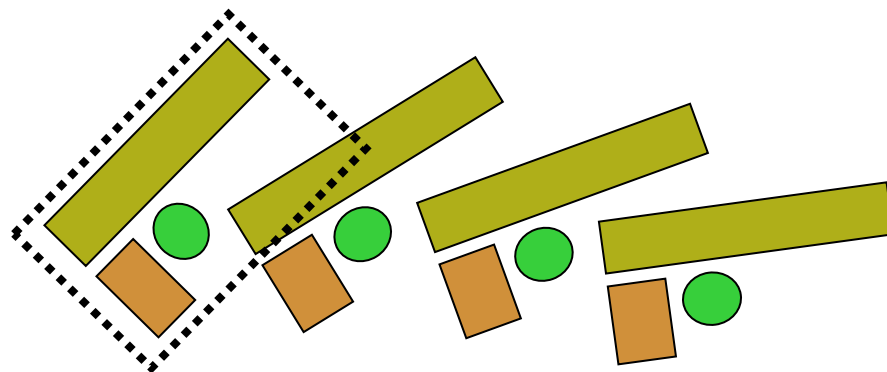
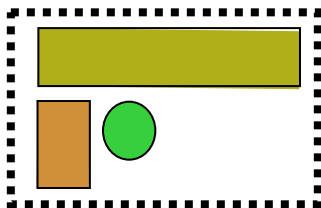
```
( G4LogicalVolume* volume,  
  G4ThreeVector& translation,  
  G4RotationMatrix* rotation );
```

- 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

```
G4AssemblyVolume* assembly = new G4AssemblyVolume();
G4RotationMatrix Ra;
G4ThreeVector Ta;
Ta.setX(...); Ta.setY(...); Ta.setZ(...);
assembly->AddPlacedVolume( plateLV, Ta, Ra );
... // repeat placement for each daughter

for( unsigned int i = 0; i < layers; i++ ) {
    G4RotationMatrix Rm(...);
    G4ThreeVector Tm(...);
    assembly->MakeImprint( worldLV, Tm, Rm );
}
```

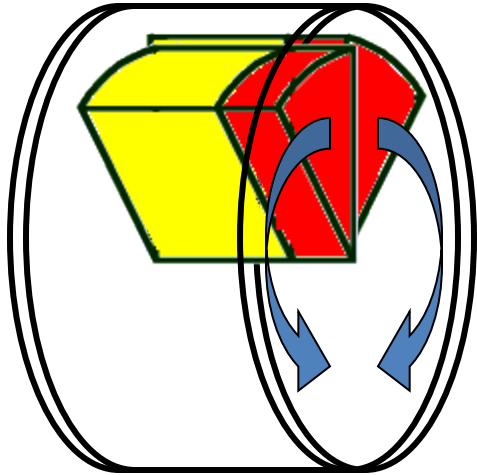




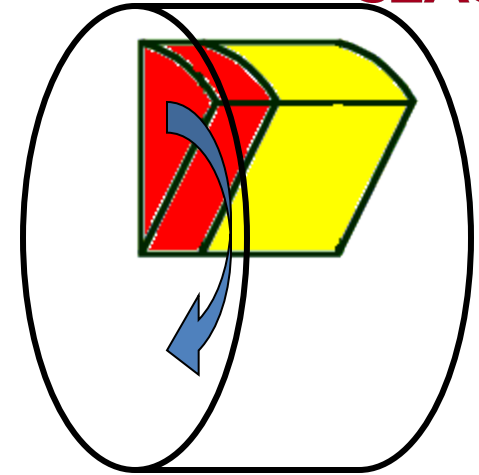
Version 10.5

Reflected volume





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



- **G4ReflectedSolid** (derived from G4VSolid)
 - Utility class representing a solid shifted from its original reference frame to a new **mirror symmetric** one
 - The reflection (G4Reflect[X/Y/Z]3D) is applied as a decomposition into rotation and translation
- **G4ReflectionFactory**
 - Singleton object using G4ReflectedSolid for generating placements of reflected volumes
- Reflections are currently limited to simple CSG solids.
 - will be extended soon to all solids

Reflecting hierarchies of volumes - 1

`G4PhysicalVolumesPair G4ReflectionFactory::Place`

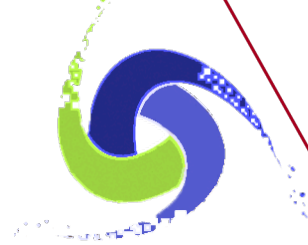
```
(const G4Transform3D& transform3D, // the transformation
 const G4String& name,           // the name
 G4LogicalVolume* LV,           // the logical volume
 G4LogicalVolume* motherLV,     // the mother volume
 G4bool noBool,                 // currently unused
 G4int copyNo)                  // optional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

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

G4PhysicalVolumesPair G4ReflectionFactory::Replicate

```
(const G4String&  name,          // the actual name
 G4LogicalVolume* LV,          // the logical volume
 G4LogicalVolume* motherLV,    // the mother volume
 Eaxis           axis          // axis of replication
 G4int           replicaNo     // number of replicas
 G4int           width,        // width of single replica
 G4int           offset=0)     // optional mother offset
```

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

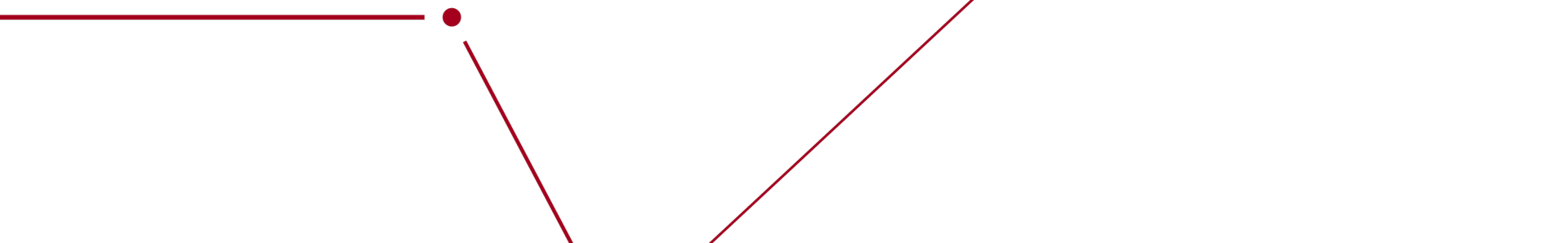


GEANT4

A SIMULATION TOOLKIT

Version 10.5

Touchable



NATIONAL
ACCELERATOR
LABORATORY

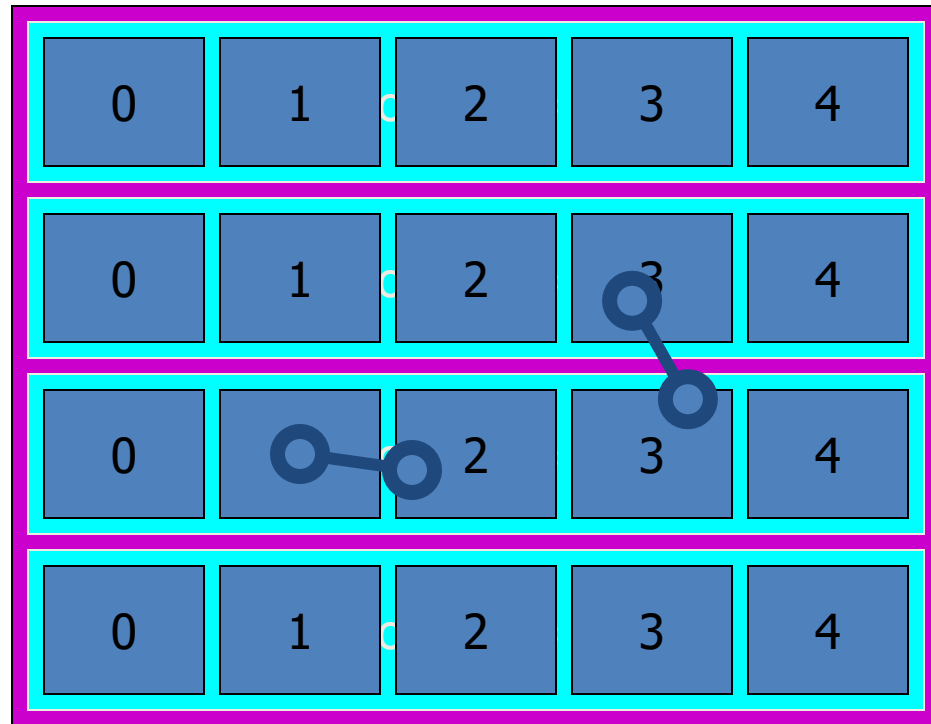


U.S. DEPARTMENT OF
ENERGY

Office of Science

- As mentioned already, G4Step has two G4StepPoint objects as its starting and ending points. All the geometrical information of the particular step should be taken from “PreStepPoint”.
 - Geometrical information associated with G4Track is identical to “PostStepPoint”.
- Each G4StepPoint object has
 - Position in world coordinate system
 - Global and local time
 - Material
 - G4TouchableHistory for geometrical information
- G4TouchableHistory object is a vector of information for each geometrical hierarchy.
 - copy number
 - transformation / rotation to its mother
- Since release 4.0, *handles* (or *smart-pointers*) to touchables are intrinsically used. Touchables are reference counted.

- Suppose a calorimeter is made of 4x5 cells.
 - and it is implemented **by two levels of replica**.
- In reality, there is **only one** physical volume **object** for each level. Its position is parameterized by its copy number.
- To get the copy number of each level, suppose what happens if a step belongs to two cells.



- ▶ Remember geometrical information in G4Track is identical to "PostStepPoint".
- ▶ You **cannot** get the correct copy number for "PreStepPoint" if you directly access to the physical volume.
- ▶ **Use touchable** to get the proper copy number, transform matrix, etc.

- G4TouchableHistory has information of geometrical hierarchy of the point.

```
G4Step* aStep;  
G4StepPoint* preStepPoint = aStep->GetPreStepPoint();  
G4TouchableHistory* theTouchable =  
    (G4TouchableHistory*) (preStepPoint->GetTouchable());  
G4int copyNo = theTouchable->GetVolume()->GetCopyNo();  
G4int motherCopyNo  
    = theTouchable->GetVolume(1)->GetCopyNo();  
G4int grandmotherCopyNo  
    = theTouchable->GetVolume(2)->GetCopyNo();  
G4ThreeVector worldPos = preStepPoint->GetPosition();  
G4ThreeVector localPos = theTouchable->GetHistory()  
    ->GetTopTransform().TransformPoint(worldPos);
```