Geant 4

Navigation – status & ideas for future

J. Apostolakis

for the G4 Geometry & Transportation WG

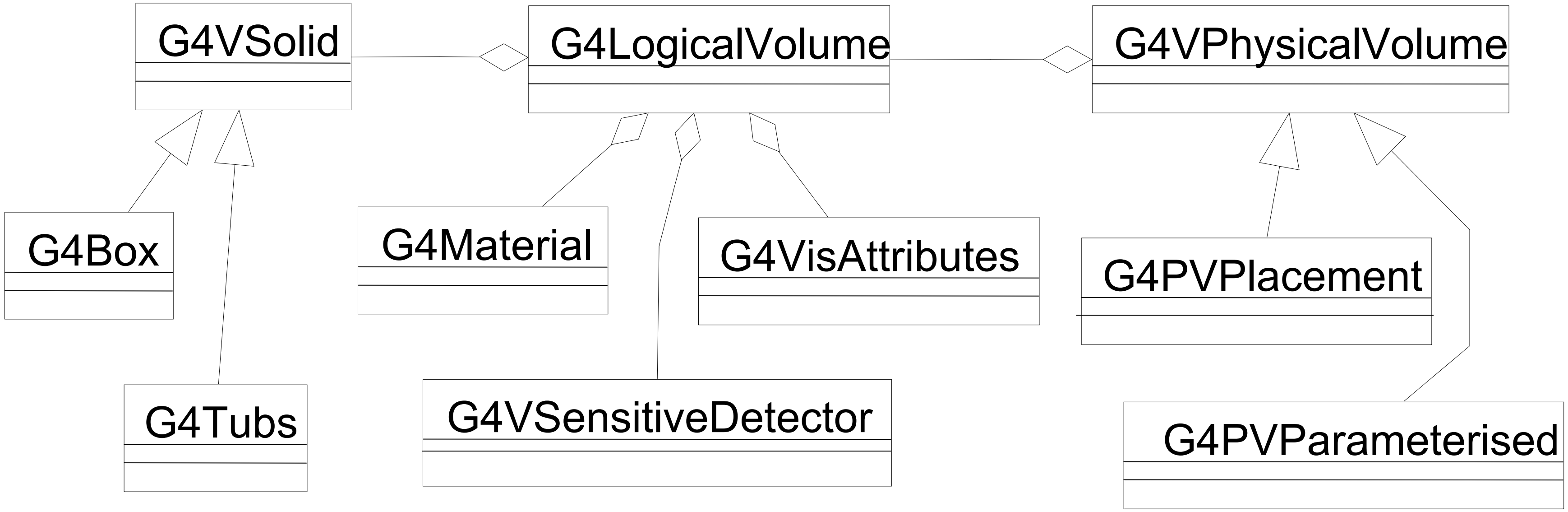
cern.ch/geant4

Outline

- Volumes and their relation to Navigation
  - The G4Navigator
  - Touchables
- Classes used by tracking
  - Touchables,
- Advanced functionality
  - Multiple Geometries & Navigators
- Future extensions (under study)
  - Double Navigator

The volume classes

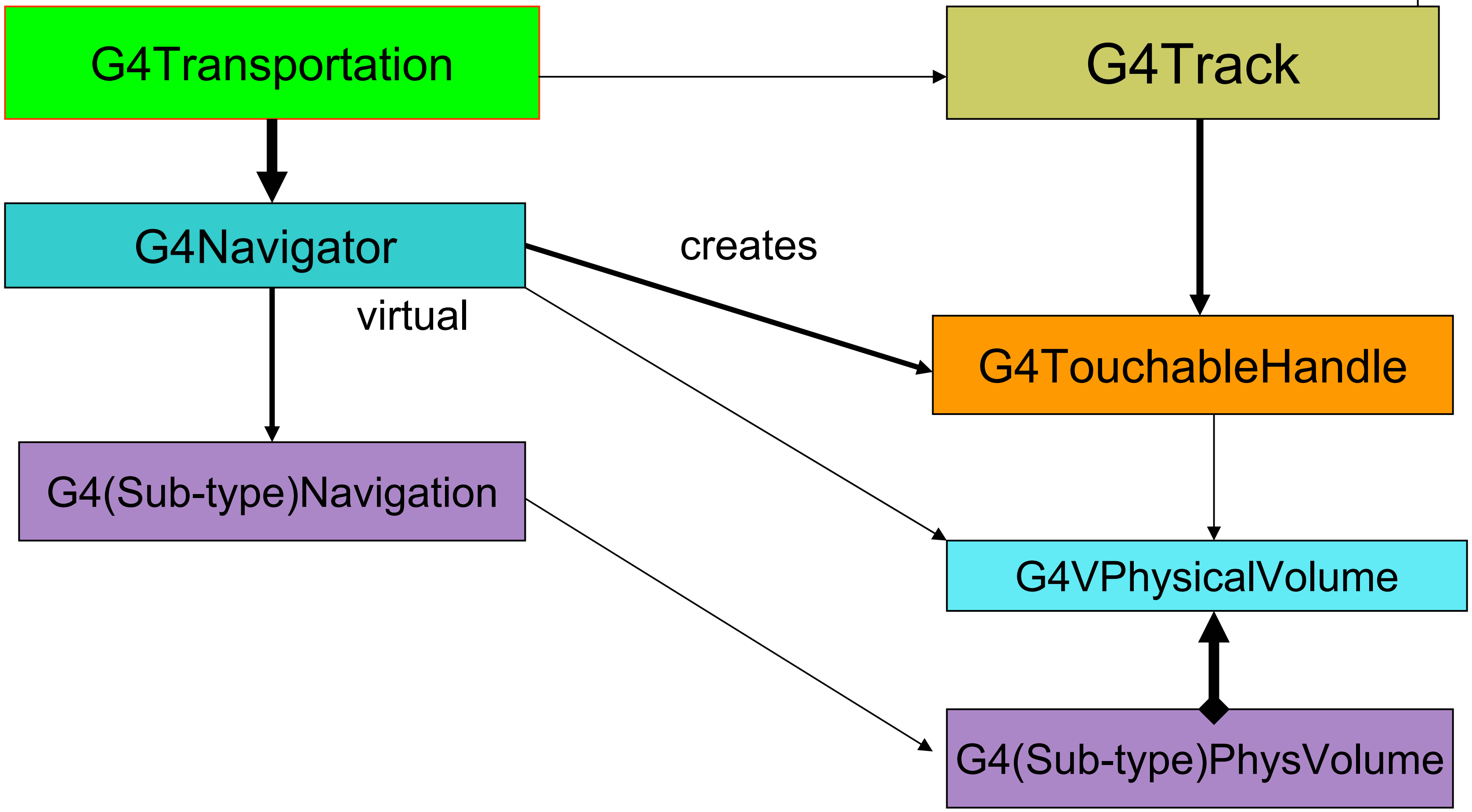
- Three conceptual layers
  - **G4VSolid** -- *shape, size*
  - **G4LogicalVolume** -- *daughter physical volumes, material, sensitivity, user limits, etc.*
  - **G4VPhysicalVolume** -- *position, rotation*



Navigator Clients

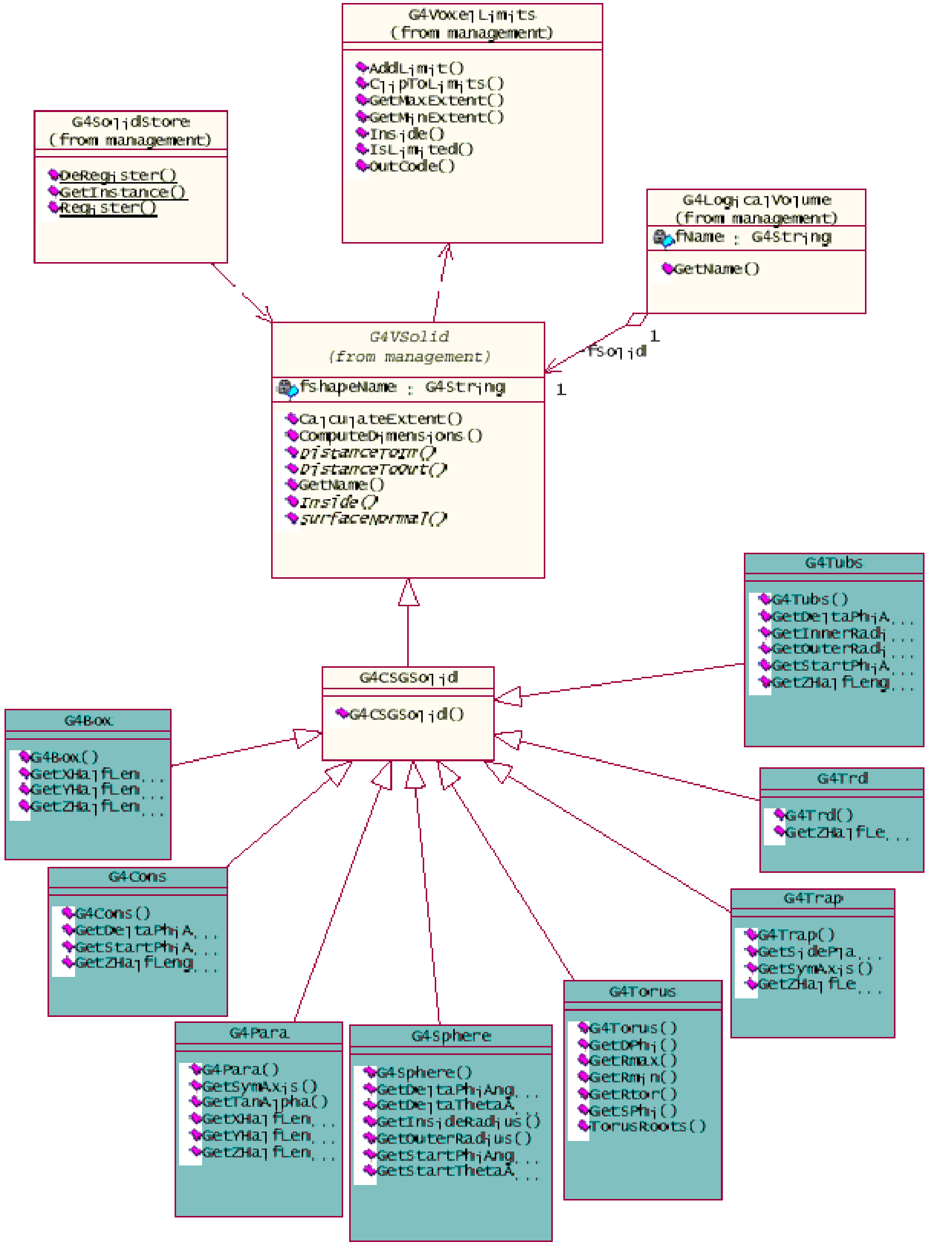
- All processes that move a particle
  - Transportation Process
  - Optical Processes
  - Geometrical biasing
    - Russian rouletter/splitting via importance
  - Fast simulation / Parameterization Process
  - Other processes that can move a particle
    - Multiple Scattering (eg turn before a boundary)
- Tracking
  - Uses Touchable, Physical / Logical Volume
- Propagator in Field
  - To intersect classes

Navigation and Volumes



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



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

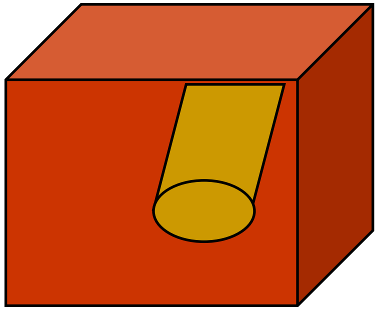
G4VPhysicalVolume

- G4PVPlacement 1 Placement = One Volume
  - One instance positioned in a mother volume
- Repeated volumes 1 Repeated = Many Volumes
  - G4PVParameterised
    - Parameterised by the copy number
      - Shape, size, material, position and rotation can be parameterised, by implementing a concrete class of `G4VPVParameterisation`.
  - G4PVReplica 1 Replica = Many Volumes
    - Slicing a volume into smaller pieces (if it has a symmetry)

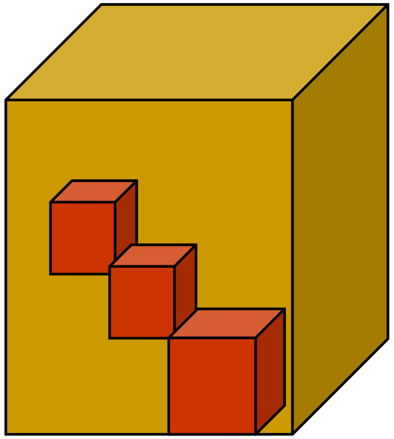
Note: Currently all repeated volumes can be used only for volumes that either a) have no further daughters or b) are identical in size & shape.

Physical Volumes

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



placement



repeated

What can a touchable do ?

- All generic touchables can reply to these queries:
  - positioning information (rotation, position)
    - `GetTranslation()`, `GetRotation()`
- Specific types of touchable also know:
  - (solids) - their associated shape: `GetSolid()`
  - (volumes) - their physical volume: `GetVolume()`
  - (volumes) - their replication number: `GetReplicaNumber()`
  - (volumes hierarchy or touchable history):
    - info about its hierarchy of placements: `GetHistoryDepth()`
      - At the top of the history tree is the world volume
    - modify/update touchable: `MoveUpHistory()`, `UpdateYourself()`
      - take additional arguments

Touchables & their handles

- A touchable used for tracking must
  - Implement all relevant methods
    - Including ‘history’ methods

    like G4TouchableHistory
  - be lightweight (as many can be created)
    - Any process or user code can obtain a handle
      - Sharing a (constant) history ‘stack’

Relation of Navigator / Phys Volumes

- Navigator and Physical Volumes co-work
  - Sub-Navigator for each Physical Volume type
- 'Live' tree of geometry must be kept consistent
  - Up to Geant4 5.2, backpointers in G4VPhysicalVolume to mother PhysVol
    - Now information from Touchable(s) only
  - Repeated volumes 'set' to next volume
    - Parameterised volume's attributes set for right one

To be noted

- Solids must have a surface

- Navigator's methods are called in order
  - Locate first & before each step
  - ComputeStep
  - ComputeSafety

- Touchable handles have an independent lifetime
  - A process can keep one
  - A secondary particle carries it

Changes in Geant4 6.0

- G4Navigator revised and made virtual
  - Interface cleaned-up
  - Several methods made virtual
    - LocateGlobalPointAndSetup
    - ComputeStep
    - ComputeSafety
- Physical Volume back-pointer eliminated

Some Advanced features

Already possible

- Change a geometry in a single Geant4 job
- Use several geometries at the same time
  - A mass geometry in which physics happens
  - A fast simulation / parameterization ‘ghost’ geometry for shower parameterization
  - A biasing geometry with importance values

To note (today):

- A parallel geometry will not by itself limit the step
  - A process must do this:
    - FastSimulationManagerProcess (FSMP) for parameterization;

Ideas for future changes

New functionality (under study)

- Enable transparent use of two geometries: “Dual Navigator”
  - Enable G4 geometry to limit the step also at the boundary of a parallel geometry
  - Allows simple use of biasing for charged particles in a parallel geometry
  - Simplifies complex processes which need to know about two geometries, eg FSMP
- Enable run-time choice of surface thickness, ie ‘tolerance’ parameter
  - Currently kCarTolerance is compile time constant
    - And its default value is 1.0 e-9 * mm

Backup slides

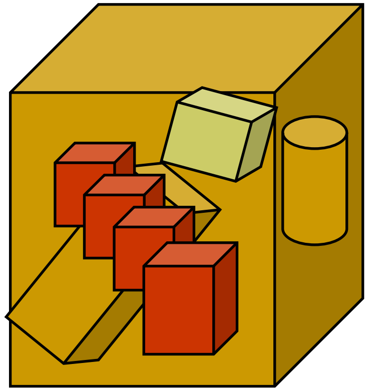
G4PVPlacement

```
G4PVPlacement(G4RotationMatrix* pRot,
               const G4ThreeVector& tlate,
              G4LogicalVolume* pCurrentLogical,
               const G4String& pName,
               G4LogicalVolume* pMotherLogical,
               G4bool pMany,
               G4int pCopyNo);
```

- 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
  - The combination of the two variants above

Parameterised Physical Volumes

- User written functions define:
  - the size of the solid (dimensions)
    - Function `ComputeDimensions(…)`
  - where it is positioned (transformation)
    - Function `ComputeTransformations(…)`
- Optional:
  - the type of the solid
    - Function `ComputeSolid(…)`
  - the material
    - Function `ComputeMaterial(…)`
- Limitations:
  - Applies to simple CSG solids only
  - Daughter volumes allowed only for special cases
- Very powerful
  - Consider parameterised volumes as “leaf” volumes



G4PVParameterised

```
G4PVParameterised(const G4String& pName,
                  G4LogicalVolume* pCurrentLogical,
                  G4LogicalVolume* pMotherLogical,
                  const EAxis pAxis,
                  const G4int nReplicas,
                  G4VPVParameterisation* pParam);
```

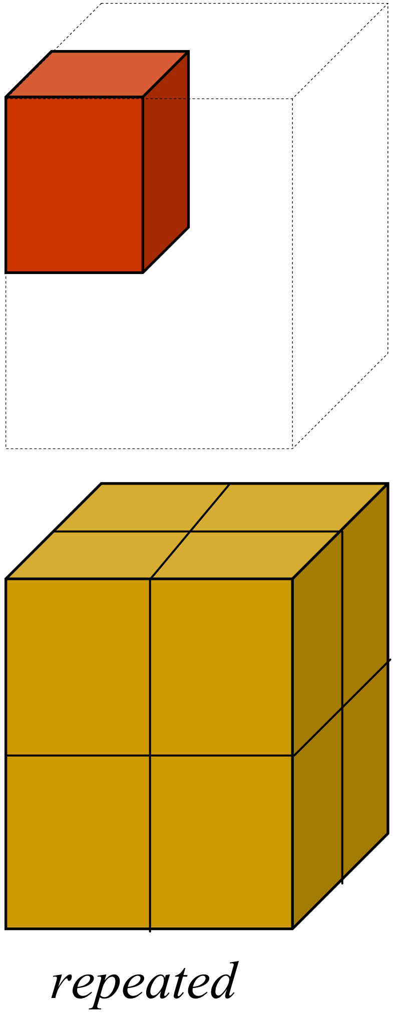
- Replicates the volume `nReplicas` times using the parameterisation `pParam`, within the mother volume
- The positioning of the replicas is dominant along the specified Cartesian axis
  - If `kUndefined` is specified as axis, 3D voxelisation for optimisation of the geometry is adopted
- Represents many touchable detector elements differing in their positioning and dimensions. Both are calculated by means of a `G4VPVParameterisation` object
- Alternative constructor using pointer to physical volume for the mother

Parameterisation
example

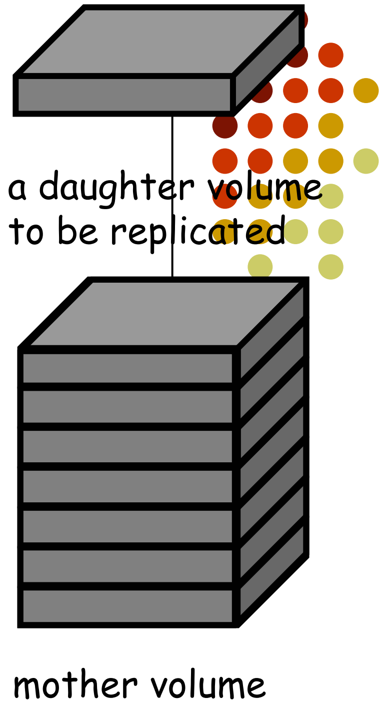
```
class ChamberParameterisation : public G4VPVParameterisation
{
  public:
    ChamberParameterisation( G4int NoChambers, G4double startZ,
                             G4double spacing, G4double widthChamber,
                             G4double lenInitial, G4double lenFinal );
   ~ChamberParameterisation();
   void ComputeTransformation (const G4int copyNo,
                               G4VPhysicalVolume* physVol) const;
   void ComputeDimensions (G4Box& trackerLayer, const G4int copyNo,
                           const G4VPhysicalVolume* physVol) const;
}
```

Replicated Physical Volumes

- The mother volume is sliced into replicas, all of the same size and dimensions.
- Represents many touchable detector elements differing only in their positioning.
- 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



G4PVReplica

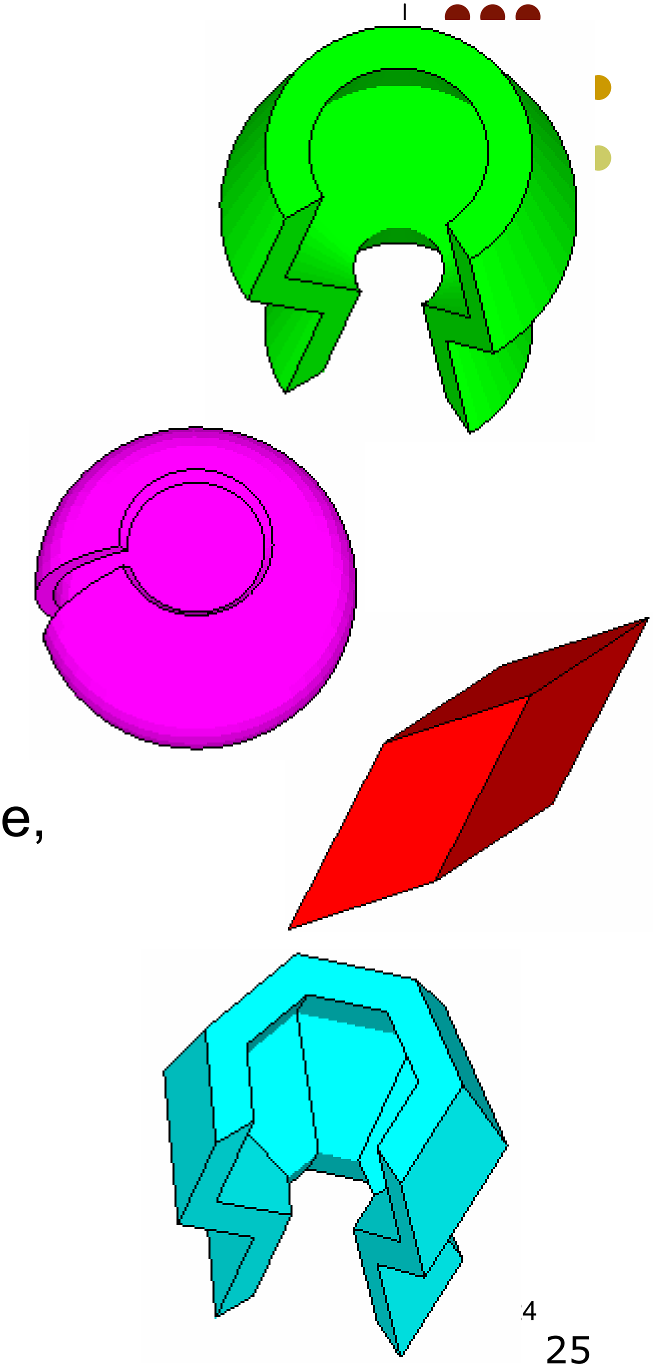


```
G4PVReplica(const G4String& pName,
            G4LogicalVolume* pCurrentLogical,
            G4LogicalVolume* pMotherLogical,
            const EAxis pAxis,
            const G4int nReplicas,
            const G4double width,
            const G4double offset=0);
```

- Alternative constructor: using pointer to physical volume for the mother
- An `offset` can only be associated to a mother offset along the axis of replication
- 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

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

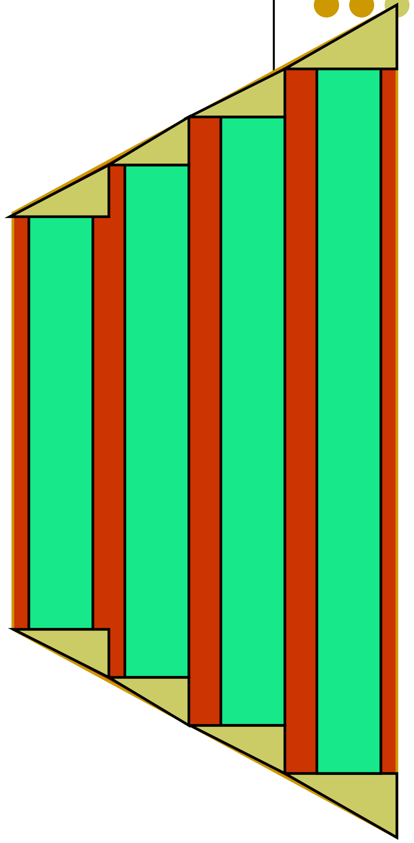


Divided Physical Volumes

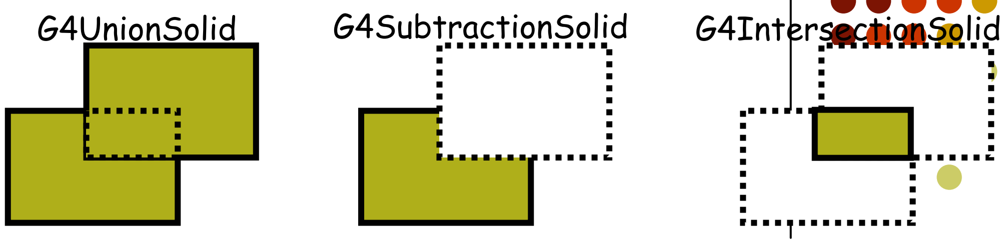
- Implemented as “special” kind of parameterised volumes
  - Applies to CSG-like solids only (box, tubs, cons, para, trd, polycone, polyhedra)
  - Divides a volume in identical copies along one of its axis (copies are not strictly identical)
    - e.g. - a tube divided along its radial axis
    - Offsets can be specified
- The possible axes of division vary according to the supported solid type
- Represents many touchable detector elements differing only in their positioning
- `G4PVDivision` is the class defining the division
  - The parameterisation is calculated automatically using the values provided in input

Uses of Parameterised Volumes

- Complex detectors
  - with large repetition of volumes
    - regular or irregular
- Medical applications
  - the material in animal tissue is measured
    - cubes with varying material



Boolean Solids



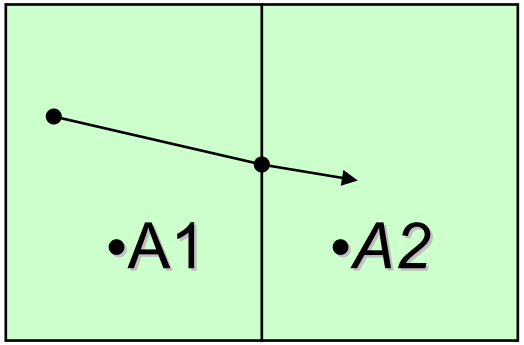
- Solids can be combined using boolean operations:
  - `G4UnionSolid, G4SubtractionSolid, G4IntersectionSolid`
  - Requires: 2 solids, 1 boolean operation, and an (optional) transformation for the 2nd solid
    - 2nd solid is positioned relative to the coordinate system of the 1st solid
- Example:

```
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

Benefits of Touchables in track

- Permanent information stored
  - to avoid implications with a “live” volume tree
- Full geometrical information available
  - to processes
  - to sensitive detectors
  - to hits

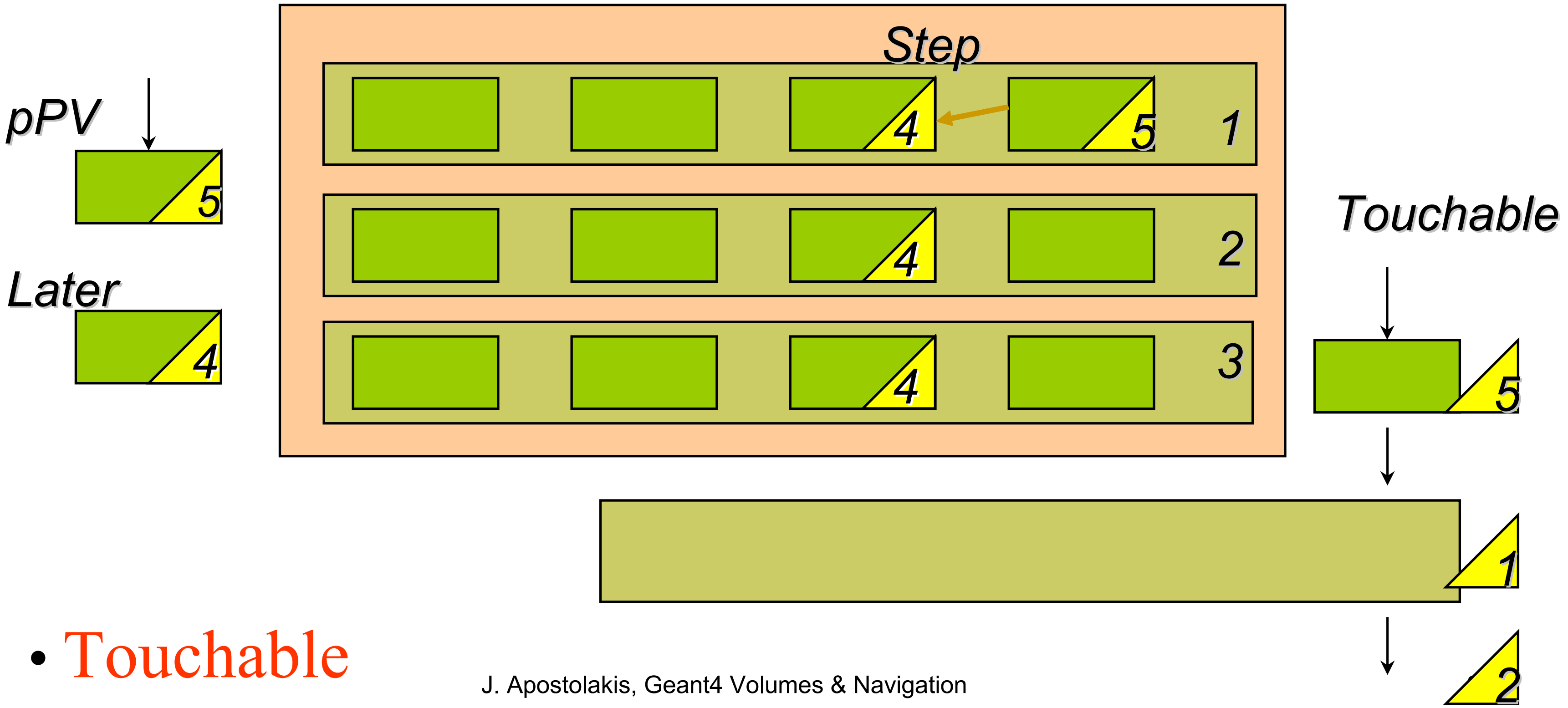


Touchable - 1

- G4Step has two G4StepPoint objects as its starting and ending points. All the geometrical information of the particular step should be got from "PreStepPoint"
  - Geometrical information associated with G4Track is basically same as "PostStepPoint"
- Each G4StepPoint object has:
  - position in world coordinate system
  - global and local time
  - material
  - G4TouchableHistory for geometrical information
    - Copy-number, transformations
- *Handles* (or *smart-pointers*) to touchables are intrinsically used. Touchables are reference counted

How to identify a volume uniquely?

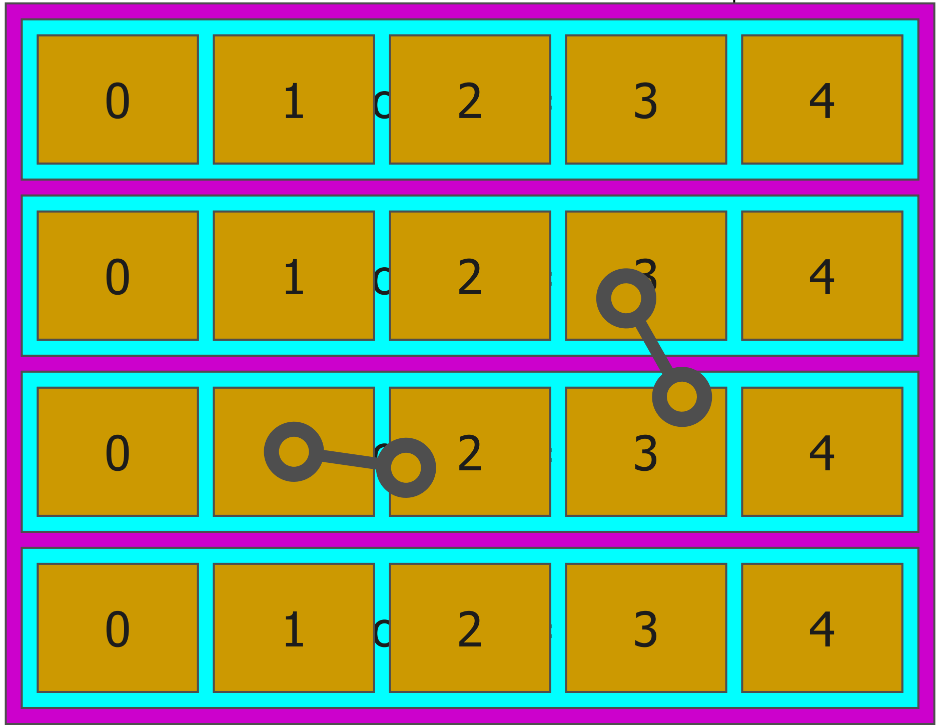
- Need to identify a volume uniquely
- Is a physical volume pointer enough? NO!



- Touchable

Copy numbers

- 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 collect copy number for "PreStepPoint" if you directly access to the physical volume
- Use touchable to get the proper copy number, transform matrix,…