GEOMETRY INTRODUCTION

Training course at International User Conference on Medicine and Biology applications

Bordeaux, 8-11 October 2013

V. Ivanchenko

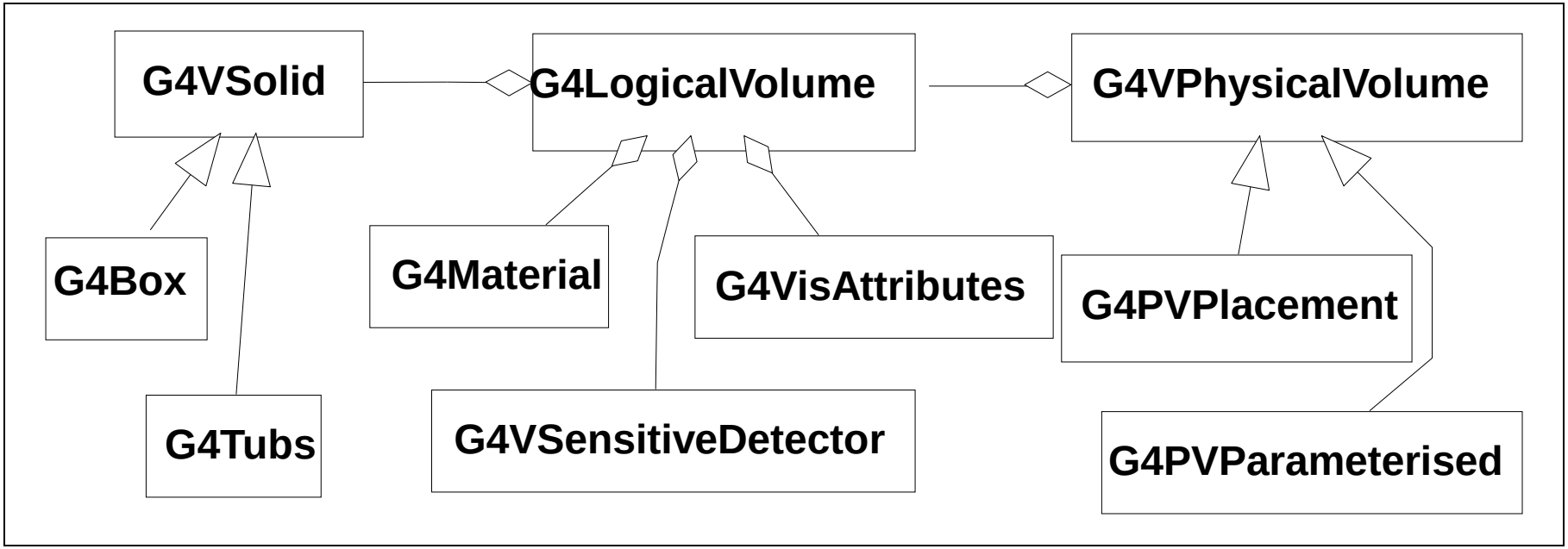
adaptation of the original lectures of

Makoto Asai (SLAC)

Contents

Detector geometry

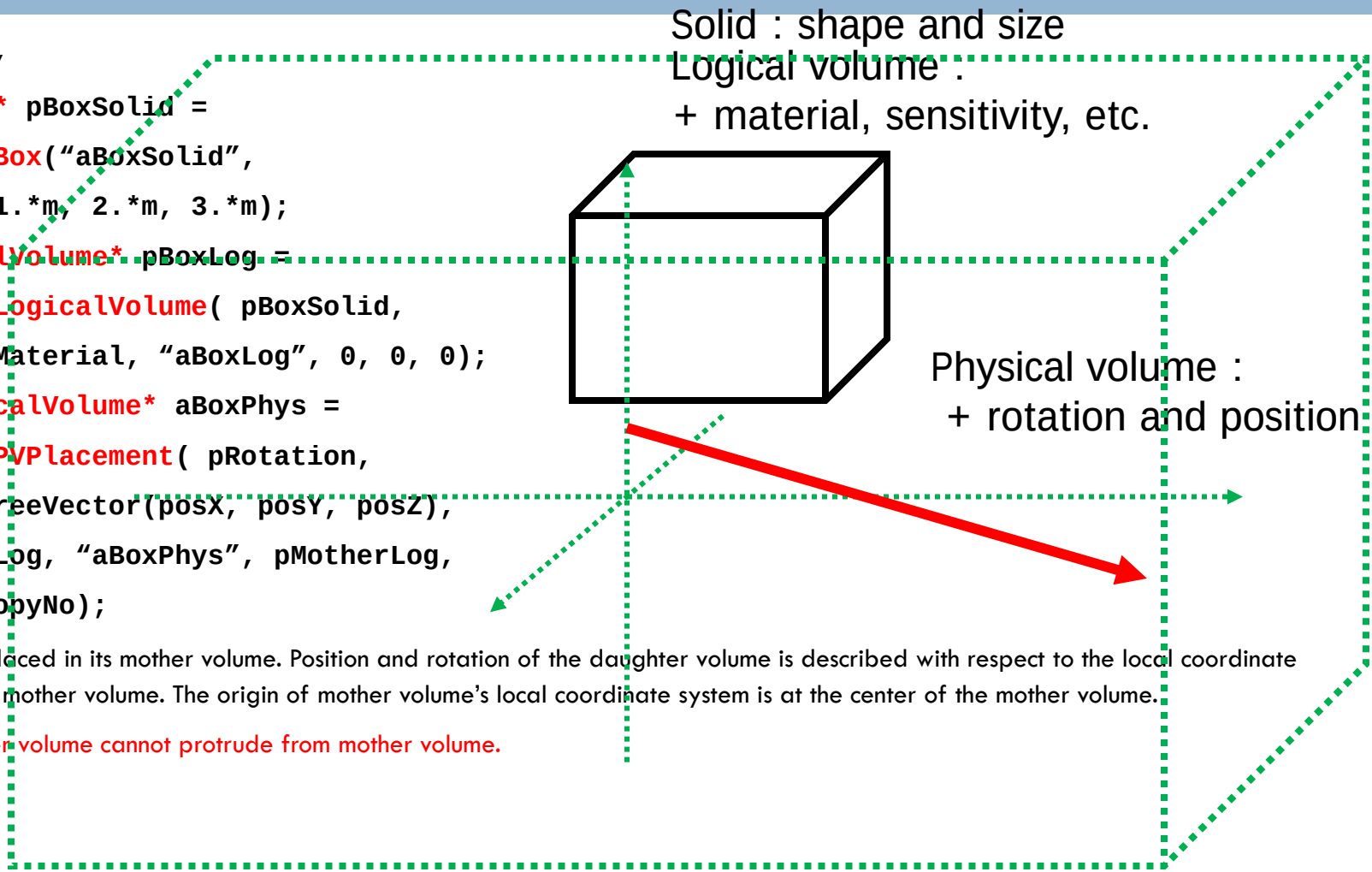
- Three conceptual layers
  - G4VSolid -- *shape, size*
  - G4LogicalVolume -- *daughter physical volumes, material, sensitivity, user limits, etc.*
  - G4VPhysicalVolume -- *position, rotation*



Define detector geometry

- Basic strategy

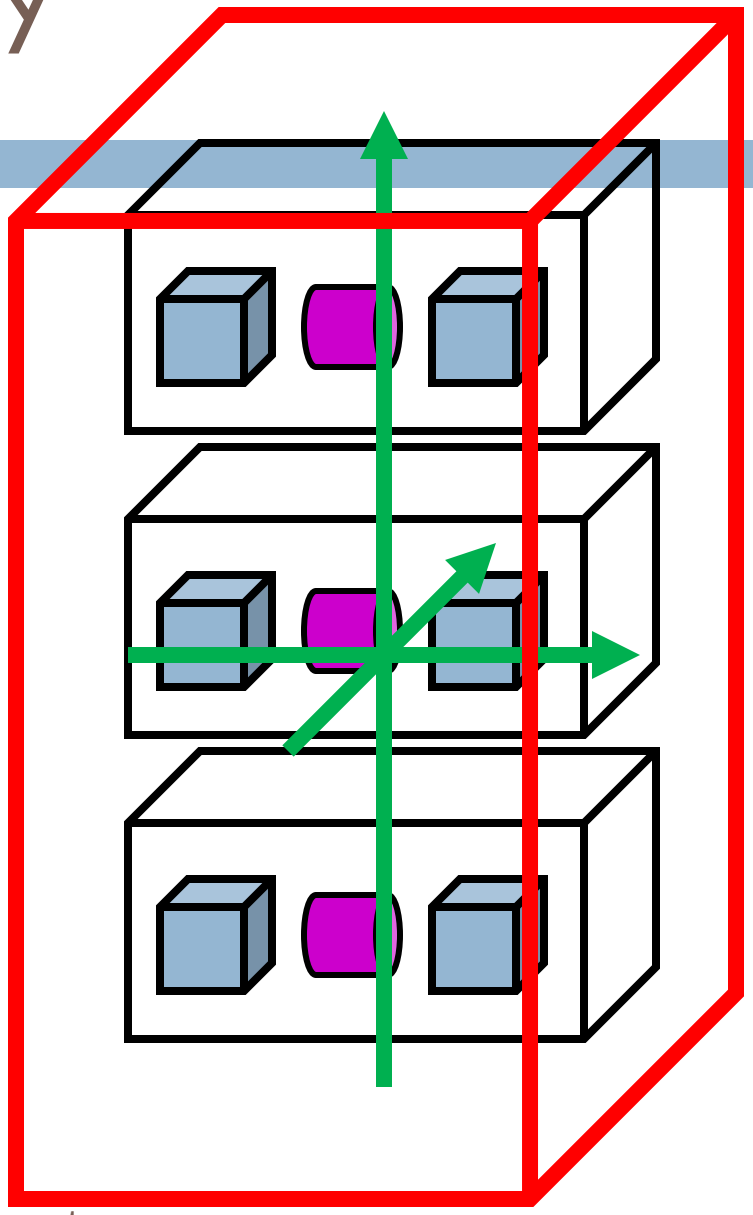
```
G4VSolid* pBoxSolid =
  new G4Box("aBoxSolid",
        1.*m, 2.*m, 3.*m);
G4LogicalVolume* pBoxLog =
  new G4LogicalVolume( pBoxSolid,
    pBoxMaterial, "aBoxLog", 0, 0, 0);
G4VPhysicalVolume* aBoxPhys =
  new G4PVPlacement( pRotation,
    G4ThreeVector(posX, posY, posZ),
    pBoxLog, "aBoxPhys", pMotherLog,
    0, copyNo);
```



- A volume is placed in its mother volume. Position and rotation of the daughter volume is described with respect to the local coordinate system of the mother volume. The origin of mother volume's local coordinate system is at the center of the mother volume.
  - Daughter volume cannot protrude from mother volume.

Geometrical hierarchy

- One logical volume can be placed more than once. One or more volumes can be placed to a mother volume.
- Note that the mother-daughter relationship is an information of G4LogicalVolume.
  - If the mother volume is placed more than once, all daughters are by definition appear in all of mother physical volumes.
- The world volume must be a unique physical volume which fully contains all the other volumes.
  - The world volume defines the global coordinate system. The origin of the global coordinate system is at the center of the world volume.
  - Position of a track is given with respect to the global coordinate system.



USER CLASSES

User classes

- main()
  - Geant4 does not provide *main()*.

  Note : classes written in red are mandatory.
- Initialization classes
  - Use G4RunManager::SetUserInitialization() to define.
  - Invoked at the initialization
    - G4VUserDetectorConstruction
    - G4VUserPhysicsList
- Action classes
  - Use G4RunManager::SetUserAction() to define.
  - Invoked during an event loop
    - G4VUserPrimaryGeneratorAction
    - G4UserRunAction
    - G4UserEventAction
    - G4UserStackingAction
    - G4UserTrackingAction
    - G4UserSteppingAction

G4VUserDetectorConstruction


```
//
// $Id: G4VUserDetectorConstruction.hh,v 1.4 2001/07/11 10:08:33 gunter Exp $
// GEANT4 tag $Name: geant4-08-00-patch-01 $
//

#ifndef G4VUserDetectorConstruction_h
#define G4VUserDetectorConstruction_h 1

class G4VPhysicalVolume;

// class description:
//
//  This is the abstract base class of the user's mandatory initialization class
// for detector setup. It has only one pure virtual method Construct() which is
// invoked by G4RunManager when it's Initialize() method is invoked.
//  The Construct() method must return the G4VPhysicalVolume pointer which represents
// the world volume.
//

class G4VUserDetectorConstruction
{
  public:
    G4VUserDetectorConstruction();
    virtual ~G4VUserDetectorConstruction();

  public:
    virtual G4VPhysicalVolume* Construct() = 0;
};

#endif
```

Construct() should return the pointer of the world physical volume. The world physical volume represents all of your geometry setup.

Your detector construction


```
#ifndef MyDetctorConstruction_h
#define MyDetctorConstruction_h 1
#include “G4VUserDetectorConstruction.hh”
class MyDetctorConstruction
      : public G4VUserDetectorConstruction
{
 public:
  G4VUserDetectorConstruction();
  virtual ~G4VUserDetectorConstruction();
  virtual G4VPhysicalVolume* Construct();
 public:
  // set/get methods if needed
 private:
  // granular private methods if needed
  // data members if needed
};
#endif
```

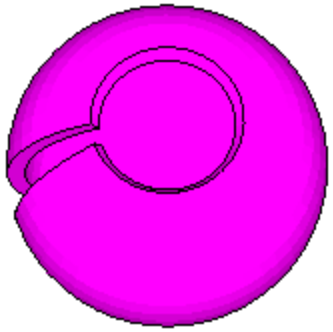
Describe your detector

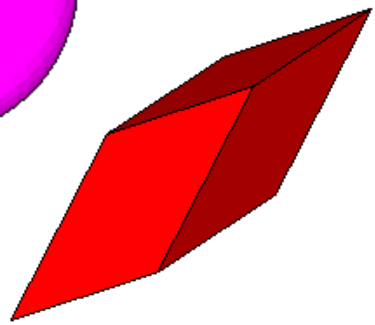
- Derive your own concrete class from G4VUserDetectorConstruction abstract base class.
- Implement the method Construct()
  1) Construct all necessary materials
  2) Define shapes/solids
  3) Define logical volumes
  4) Place volumes of your detector geometry
  5) Associate (magnetic) field to geometry *(optional)*
  6) Instantiate sensitive detectors / scorers and set them to corresponding logical volumes *(optional)*
  7) Define visualization attributes for the detector elements *(optional)*
  8) Define regions *(optional)*
- Set your construction class to G4RunManager inside your main()
- It is suggested to modularize Construct() method w.r.t. each component or sub-detector for easier maintenance of your code.

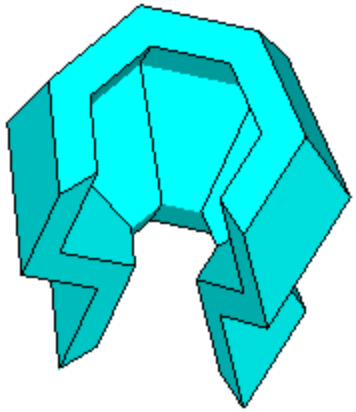
SOLID AND SHAPE

Solids

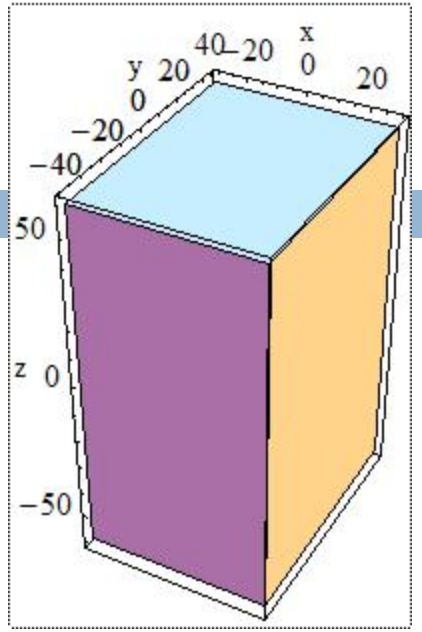
- Solids defined in Geant4:
  - CSG (Constructed Solid Geometry) solids
    - G4Box, G4Tubs, G4Cons, G4Trd, ...
    - Analogous to simple GEANT3 CSG solids
  - Specific SCG solids
    - G4Polycone, G4Polyhedra, G4Hype, ...
  - Boolean solids
    - G4UnionSolid, G4SubtractionSolid, ...
  - Tessalated solid
    - Surface is represented by many triangles



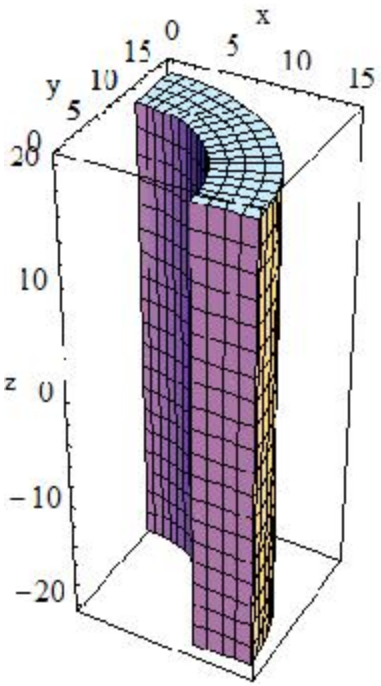




CSG: G4Box, G4Tubs

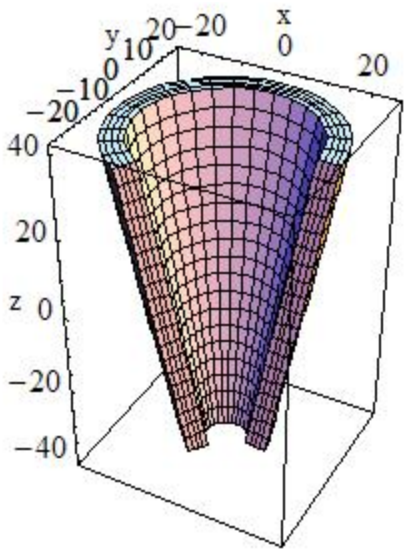



```
G4Box(const G4String &pname,    // name
            G4double half_x,    // X half size
            G4double half_y,    // Y half size
            G4double half_z);   // Z half size
```

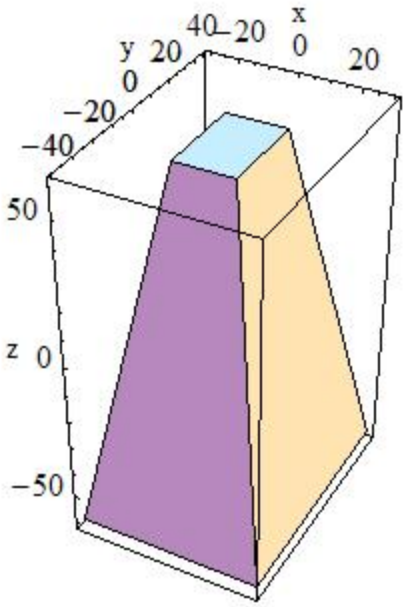



```
G4Tubs(const G4String &pname,   // name
             G4double  pRmin,   // inner radius
             G4double  pRmax,   // outer radius
             G4double  pDz,     // Z half length
             G4double  pSphi,   // starting Phi
             G4double  pDphi); // segment angle
```

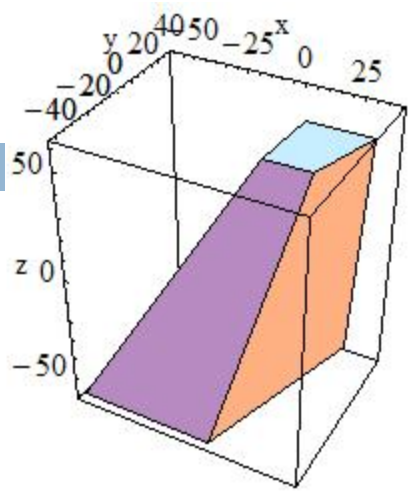
Other CSG solids



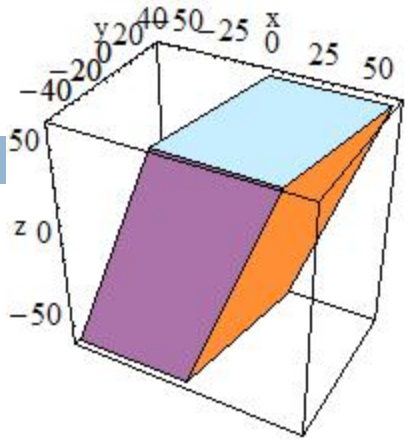
G4Cons



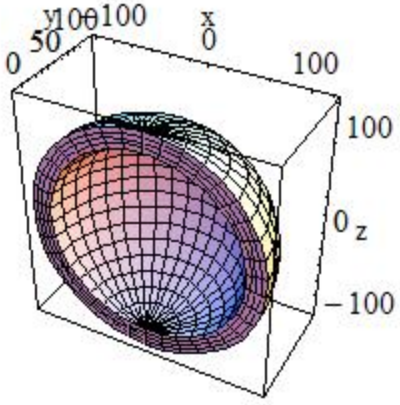
G4Trd



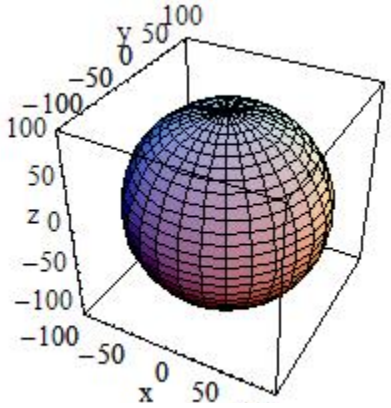
G4Trap



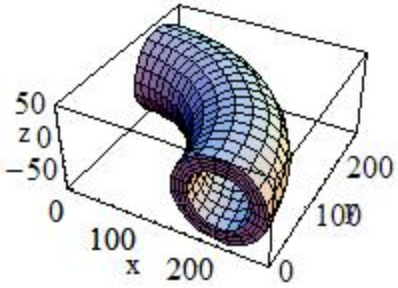
G4Para
(parallelepiped)



G4Sphere



G4Orb

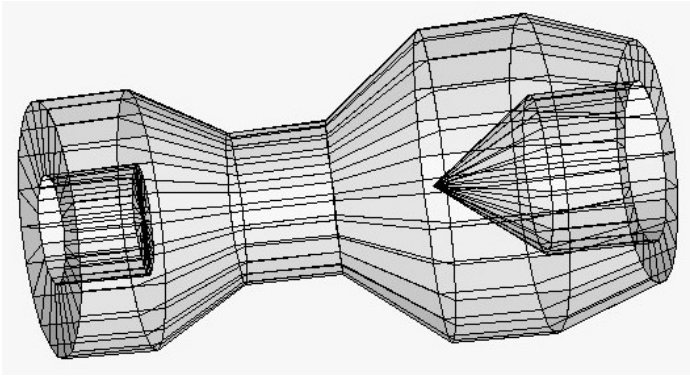


G4Torus

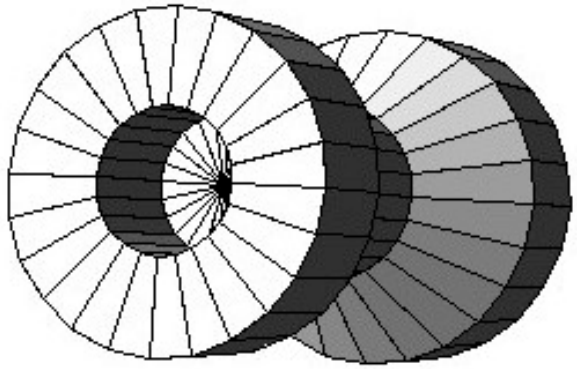
Consult to Section 4.1.2 of Geant4 Application Developers Guide for all available shapes.

Specific CSG Solids: G4Polycone

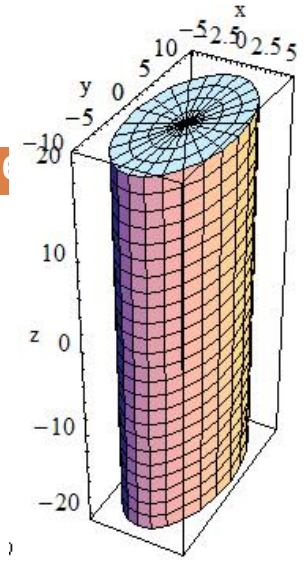

```
G4Polycone(const G4String& pName,
           G4double phiStart,
           G4double phiTotal,
           G4int numRZ,
           const G4double r[],
           const G4double z[]);
```



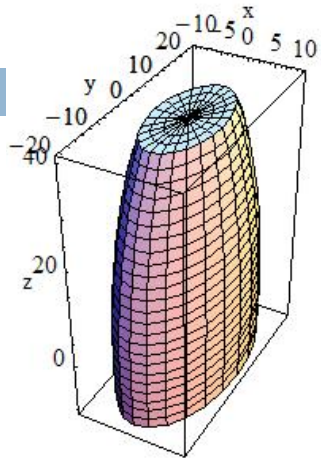
- **numRZ** - numbers of corners in the **r,z** space
- **r, z** - coordinates of corners



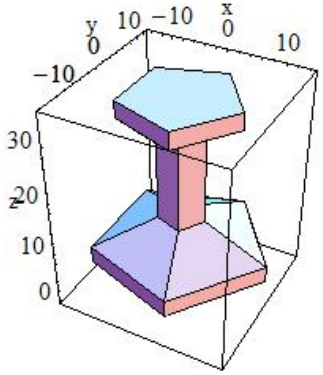
Other Specific CSG solids



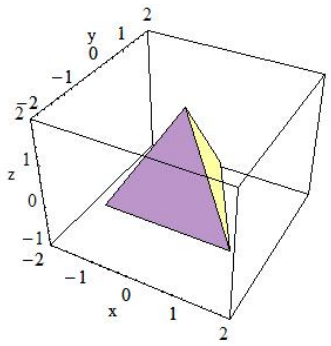
G4EllipticalTube



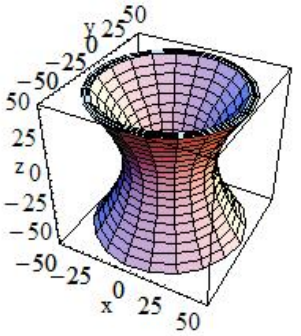
G4Ellipsoid



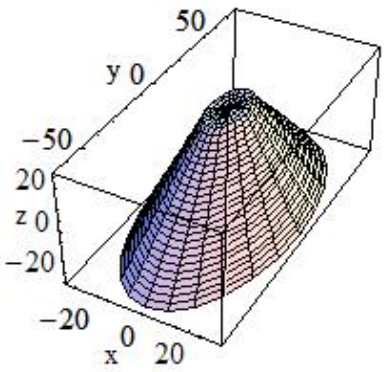
G4Polyhedra



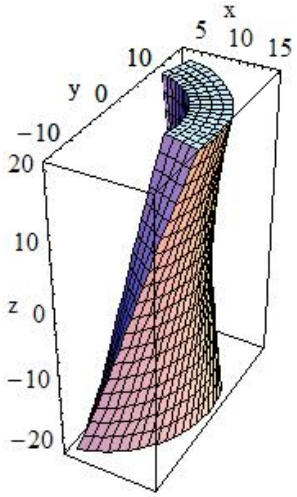
G4Tet
(tetrahedra)



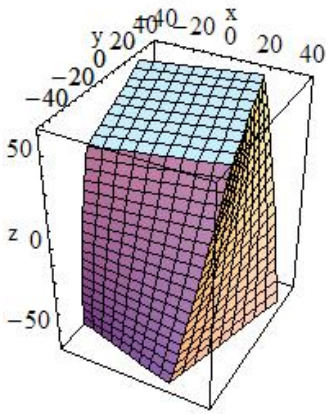
G4Hype



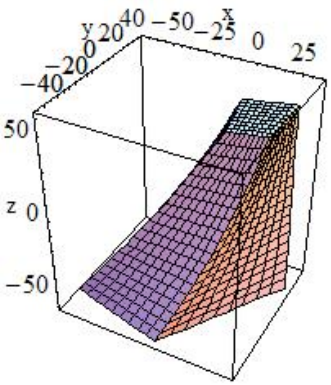
G4EllipticalCone



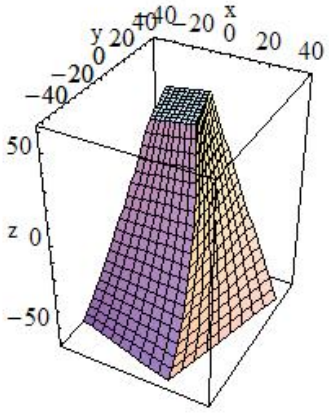
G4TwistedTubs



G4TwistedBox



G4TwistedTrap

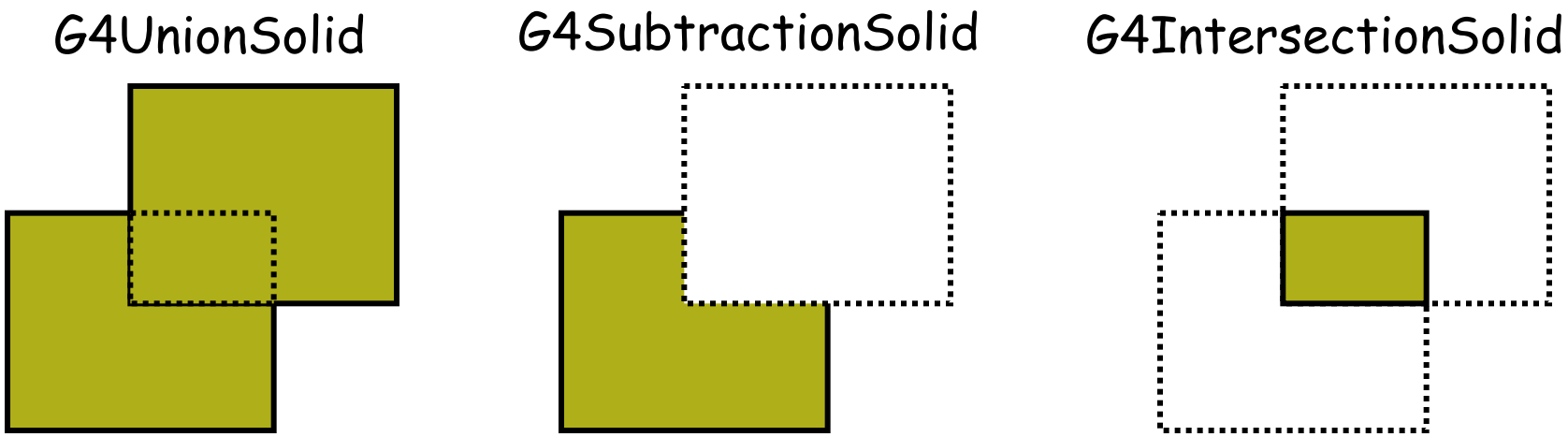


G4TwistedTrd

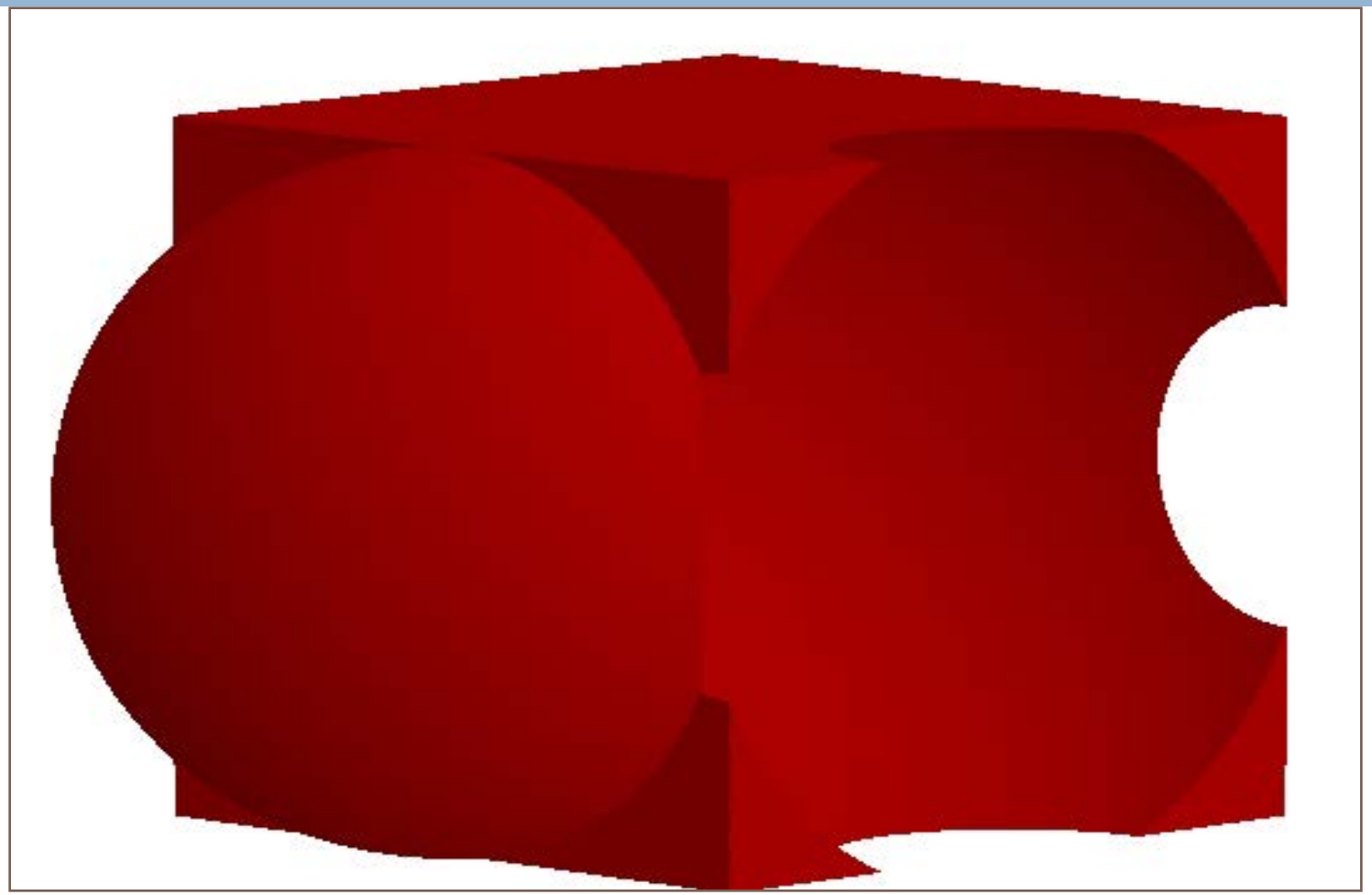
Consult to Section 4.1.2 of Geant4 Application Developers Guide for all available shapes.

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
  - Result of boolean operation becomes a solid. Thus the third solid can be combined to the resulting solid of first operation.
- Solids to be combined 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 CSG solids



Boolean solid



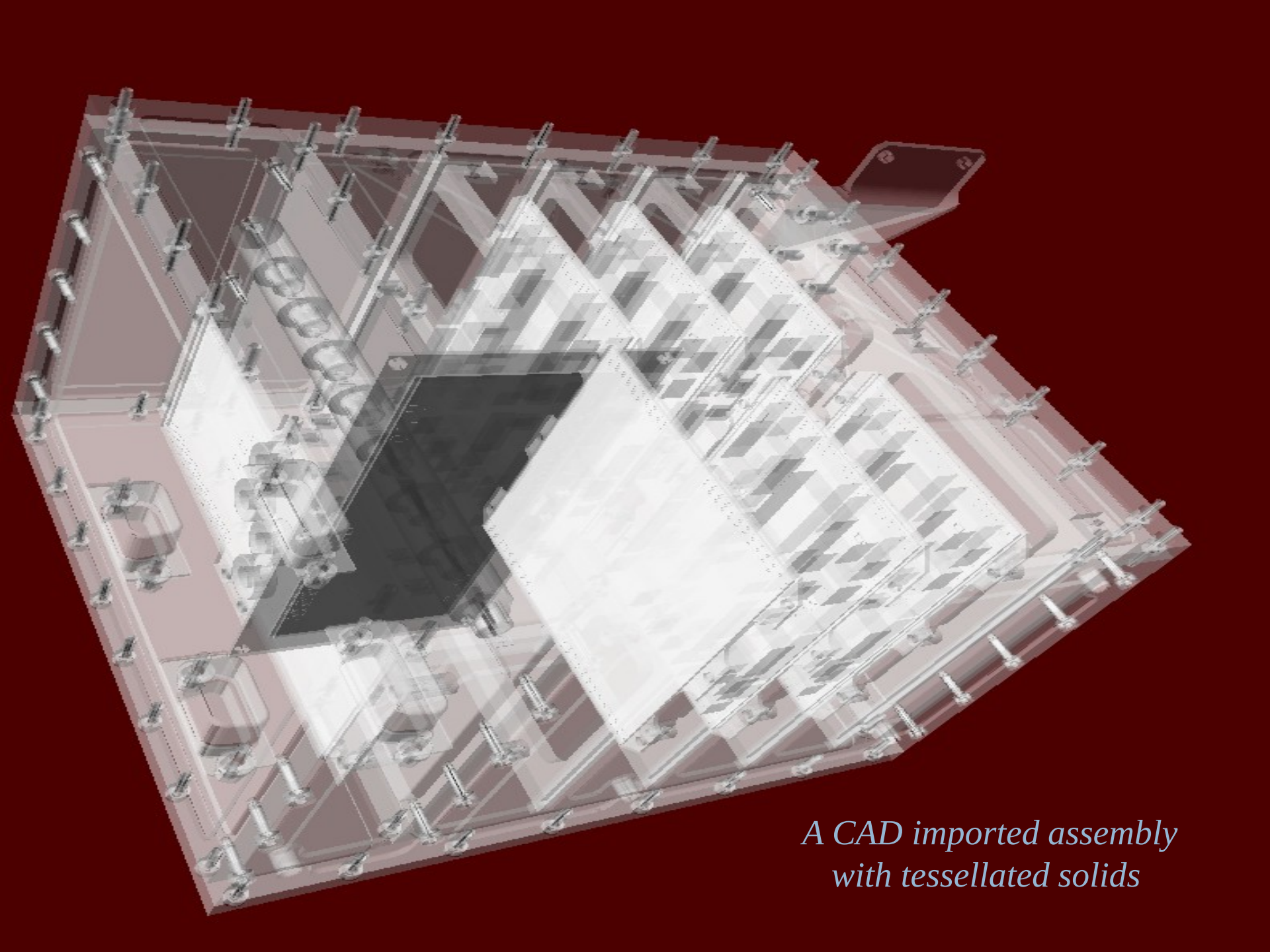
Boolean Solids - example


```
G4VSolid* box = new G4Box(“Box",50*cm,60*cm,40*cm);
G4VSolid* cylinder
 = new G4Tubs(“Cylinder”,0.,50.*cm,50.*cm,0.,2*M_PI*rad);
G4VSolid* union
 = new G4UnionSolid("Box+Cylinder", box, cylinder);
G4VSolid* subtract
 = new G4SubtractionSolid("Box-Cylinder", box, cylinder,
       0, G4ThreeVector(30.*cm,0.,0.));
G4RotationMatrix* rm = new G4RotationMatrix();
rm->RotateX(30.*deg);
G4VSolid* intersect
   = new G4IntersectionSolid("Box&&Cylinder",
       box, cylinder, rm, G4ThreeVector(0.,0.,0.));
```

- The origin and the coordinates of the combined solid are the same as those of the first solid.

Tessellated solids

- **`G4TessellatedSolid`**
  - Generic solid defined by a number of facets (`G4VFacet`)
    - Facets can be triangular (**`G4TriangularFacet`**) or quadrangular (**`G4QuadrangularFacet`**)
  - Constructs especially important for conversion of complex geometrical shapes imported from CAD systems
  - Can also be explicitly defined:
    - By providing the vertices of the facets in *anti-clock wise* order, in *absolute* or *relative* reference frame
  - GDML binding



A CAD imported assembly with tessellated solids

LOGICAL VOLUME

G4LogicalVolume


```
G4LogicalVolume(G4VSolid* pSolid,
                G4Material* pMaterial,
                const G4String &name,
                G4FieldManager* pFieldMgr=0,
                G4VSensitiveDetector* pSDetector=0,
                G4UserLimits* pULimits=0);
```

- Contains all information of volume except position and rotation
  - Shape and dimension (G4VSolid)
  - Material, sensitivity, visualization attributes
  - Position of daughter volumes
  - Magnetic field, User limits, Region
- Physical volumes of same type can share the common logical volume object.
- The pointers to solid must NOT be null.
- The pointers to material must NOT be null for tracking geometry.
- It is not meant to act as a base class.

Computing volumes and weights

- Geometrical volume of a generic solid or boolean composition can be computed from the solid:

```
G4double GetCubicVolume();
```

  - Exact volume is determinatively calculated for most of CSG solids, while estimation based on Monte Carlo integration is given for other solids.

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

```
G4double GetMass(G4bool forced=false,
  G4bool propagate=true, G4Material* pMaterial=0);
```

  - The computation may require a considerable amount of time, depending on the complexity of the geometry.
  - The return value is cached and reused until ***forced***=true.
  - Daughter volumes will be neglected if ***propagate***=false.

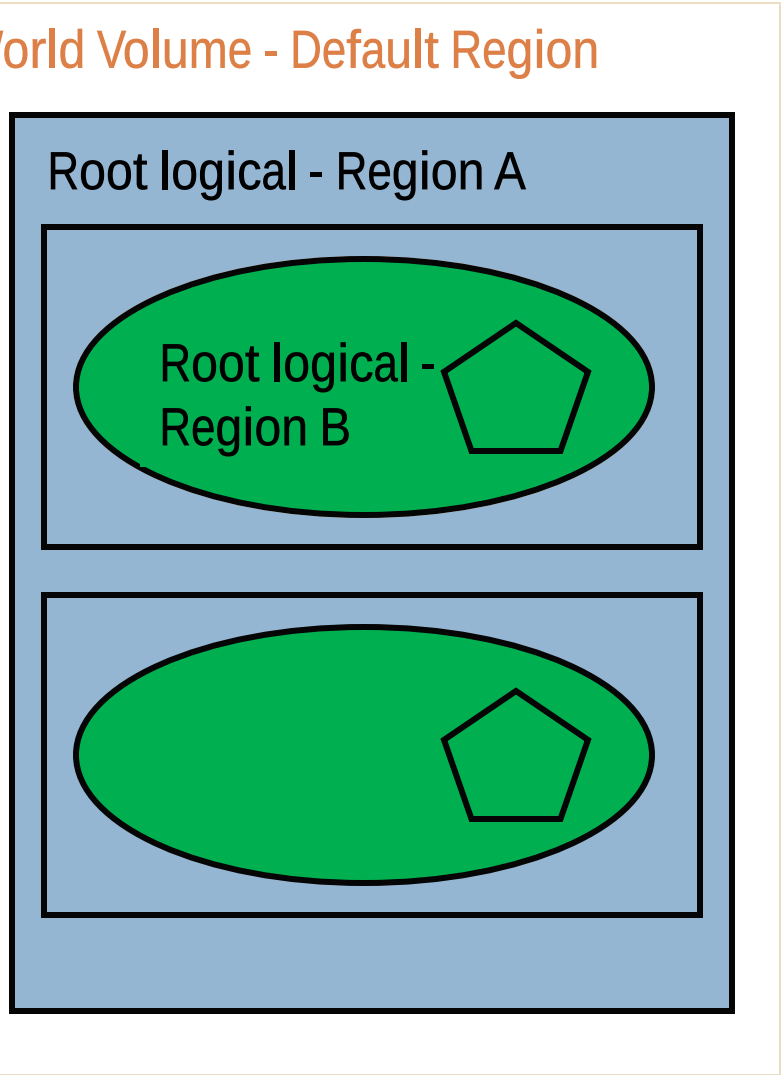
REGION

Region

- A region may have its unique
  - Production thresholds (cuts)
    - If a region in the mass geometry does not have its own production thresholds, those of the default region are used (i.e., may not be those of the parent region).
  - User limits
    - Artificial limits affecting to the tracking, e.g. max step length, max number of steps, min kinetic energy left, etc.
    - You can set user limits directly to logical volume as well. If both logical volume and associated region have user limits, those of logical volume wins.
  - User region information
    - E.g. to implement a fast Boolean method to identify the nature of the region.
  - Fast simulation manager
  - Regional user stepping action
  - Field manager
- Please note :
  - World logical volume is recognized as the default region. User is not allowed to define a region to the world logical volume.

Root logical volume

- A logical volume can be a region. More than one logical volumes may belong to a region.
- A region is a part of the geometrical hierarchy, i.e. a set of geometry volumes, typically of a sub-system.
- A logical volume becomes a root logical volume once a region is assigned to it.
  - All daughter volumes belonging to the root logical volume share the same region, unless a daughter volume itself becomes to another root.
- Important restriction :
  - No logical volume can be shared by more than one regions, regardless of root volume or not.



G4Region

- A region is instantiated and defined by

```
G4Region* aRegion = new G4Region("region_name");
aRegion->AddRootLogicalVolume(aLogicalVolume);
```

  - Region propagates down to all geometrical hierarchy until the bottom or another root logical volume.

- Production thresholds (cuts) can be assigned to a region by

```
G4Region* aRegion
 = G4RegionStore::GetInstance()->GetRegion("region_name");
G4ProductionCuts* cuts = new G4ProductionCuts;
cuts->SetProductionCut(cutValue);
aRegion->SetProductionCuts(cuts);
```

PHYSICAL VOLUME

Geant 4

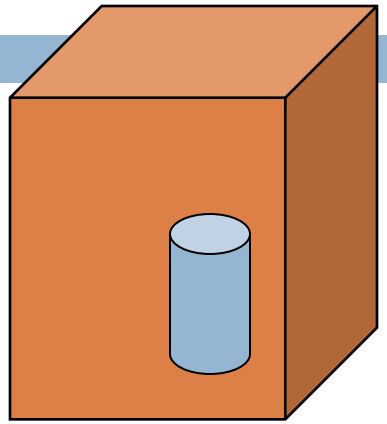
Define detector geometry

- Basic strategy

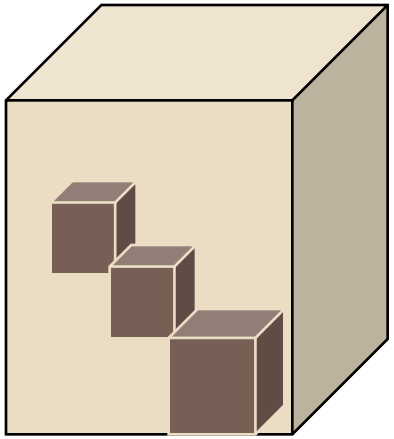
```
G4VSolid* pBoxSolid =
  new G4Box("aBoxSolid", 1.*m, 2.*m, 3.*m);
G4LogicalVolume* pBoxLog =
  new G4LogicalVolume( pBoxSolid, pBoxMaterial,
                       "aBoxLog", 0, 0, 0);
G4VPhysicalVolume* aBoxPhys =
  new G4PVPlacement( pRotation,
    G4ThreeVector(posX, posY, posZ), pBoxLog,
    "aBoxPhys", pMotherLog, 0, copyNo);
```

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
  - By implementing G4PVNestedParameterisation instead of G4VPVParameterisation, material, sensitivity and vis attributes can be parameterized by the copy numbers of ancestors.
- Limitation: a mother volume can contain either
  - many placement volumes
  - one repeated (parameterized) volume



placement



repeated

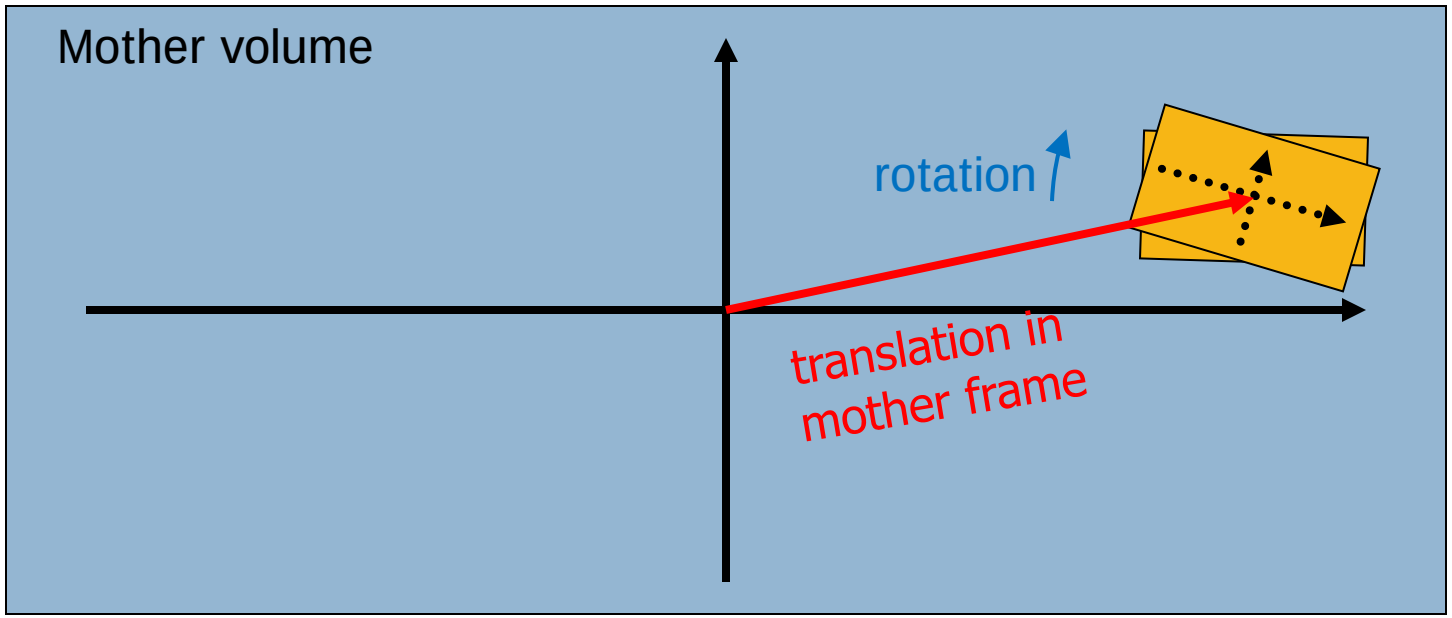
Physical volume

- 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

G4PVPlacement


```
G4PVPlacement(
  G4Transform3D(G4RotationMatrix &pRot, // rotation of daughter volume
                const G4ThreeVector &tlate, // position in mother frame
  G4LogicalVolume *pDaughterLogical,
  const G4String &pName,
  G4LogicalVolume *pMotherLogical,
  G4bool pMany, // ‘true’ is not supported yet…
  G4int pCopyNo, // unique arbitrary integer
  G4bool pSurfChk=false); // optional boundary check
```

- Single volume positioned relatively to the mother volume.

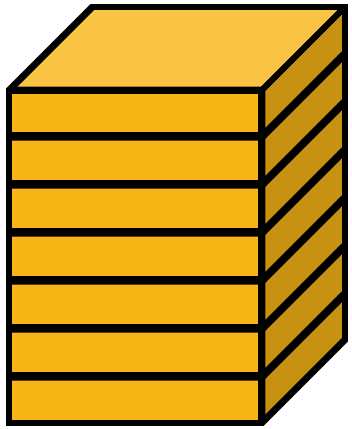


REPLICATED VOLUMES

Replicated Volumes

- 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



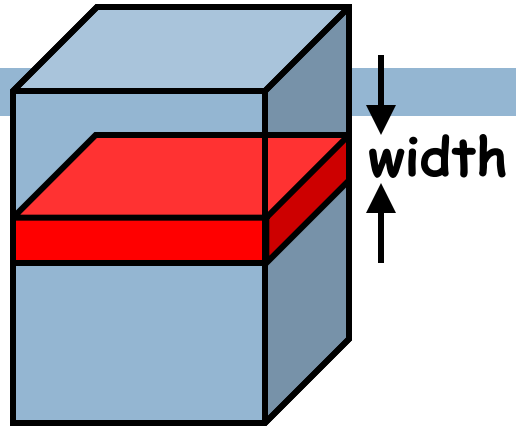
mother volume

G4PVReplica


```
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

Replica - axis, width, offset

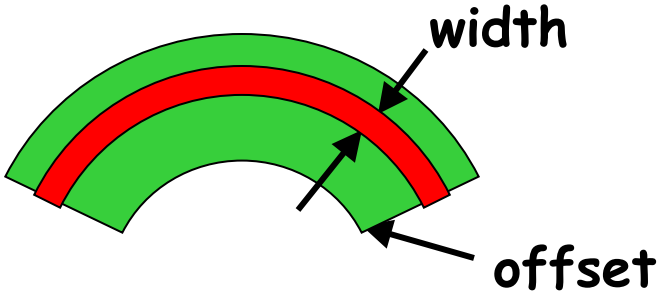


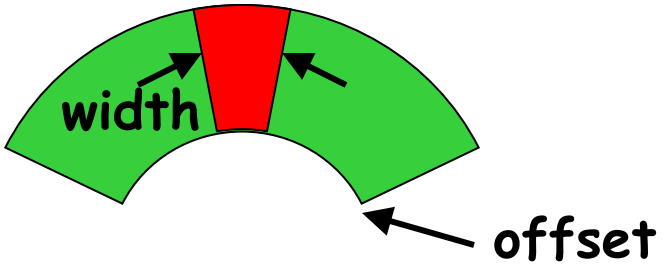
- Cartesian axes - **`kXaxis, kYaxis, kZaxis`**
  - Center of n-th daughter is given as

    `-width*(nReplicas-1)*0.5+n*width`
  - Offset shall not be used
- Radial axis - **`kRaxis`**
  - Center of n-th daughter is given as

    `width*(n+0.5)+offset`
  - Offset must be the inner radius of the mother
- Phi axis - **`kPhi`**
  - Center of n-th daughter is given as

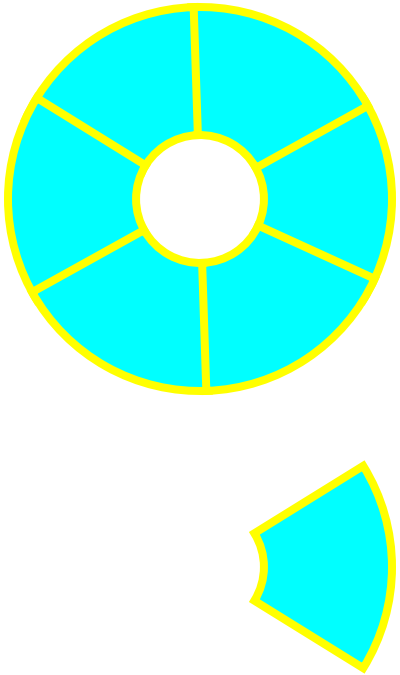
    `width*(n+0.5)+offset`
  - Offset must be the starting angle of the mother





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);
```

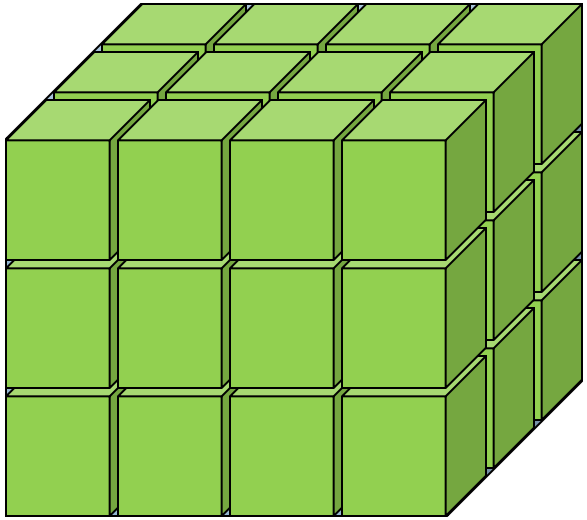


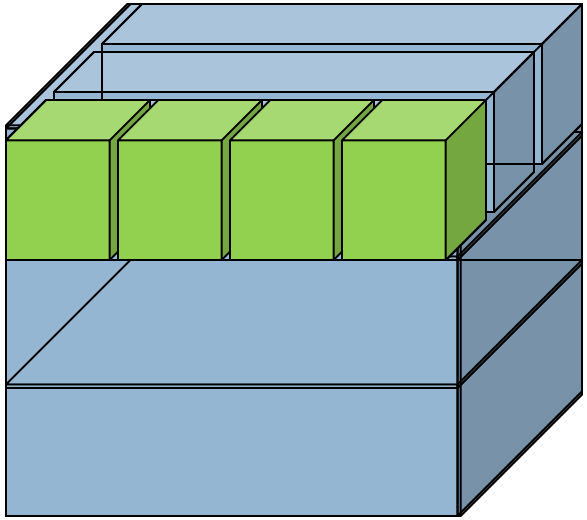
NESTED PARAMETERISATION

Nested parameterization

- Suppose your geometry has three-dimensional regular reputation 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,

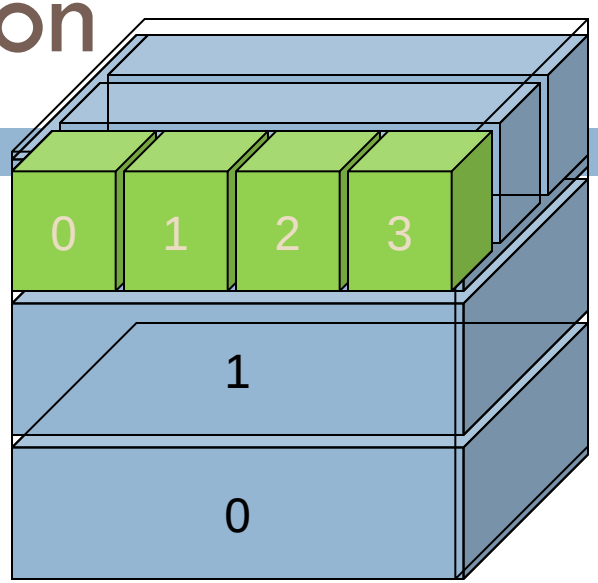
  use replicas for the first and second axes sequentially, and then use one-dimensional parameterization along the third axis.





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

Nested parameterization



- 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.
- 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

- G4VNestedParameterisation is a kind of G4VPVParameterization.
  - It can be used as an argument of G4PVParameterised.
  - All other arguments of G4PVParameterised 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.

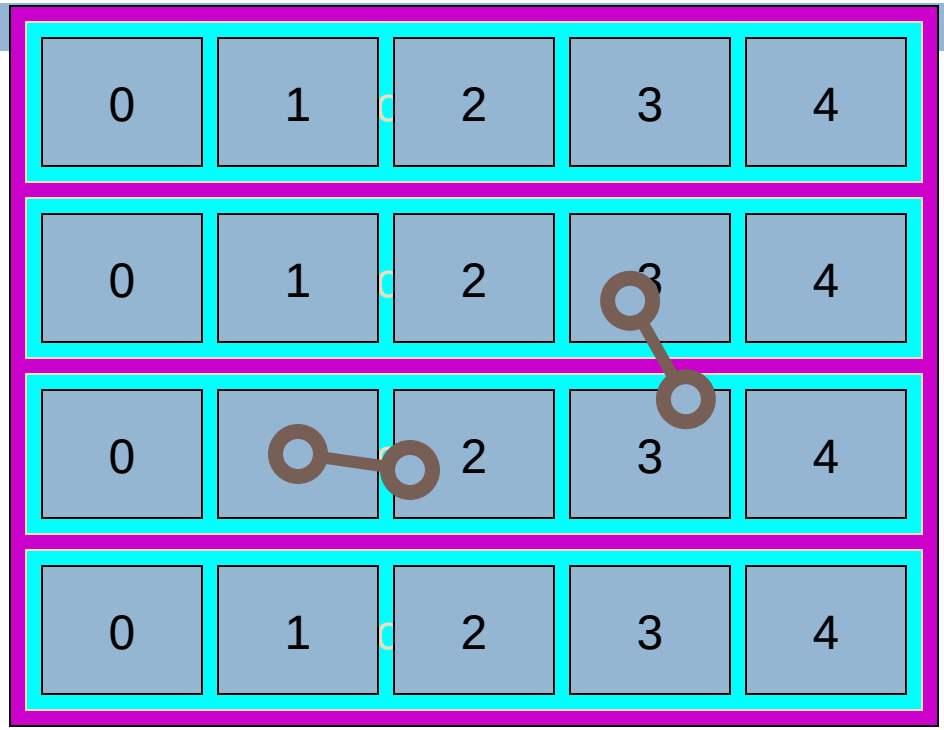
TOUCHABLE

Step point and touchable

- 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

Copy number

- Suppose a phantom 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.



- 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.

Touchable

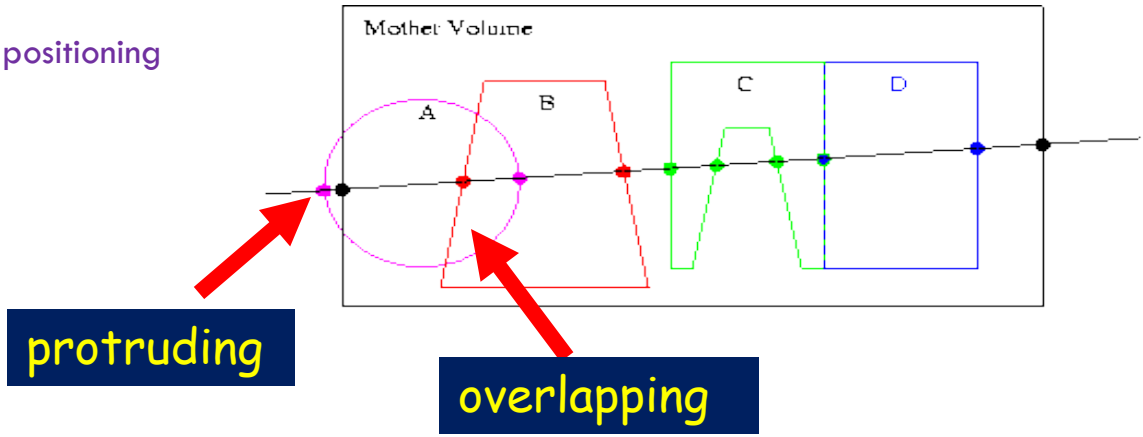
- G4TouchableHistory has information of geometrical hierarchy of the point.

```
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);
```

GEOMETRY CHECKING TOOLS

Debugging geometries

- An 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 (DAVID, OLAP)



Optional checks at construction

- Constructors of G4PVPlacement and G4PVParameterised have an optional argument "pSurfChk".

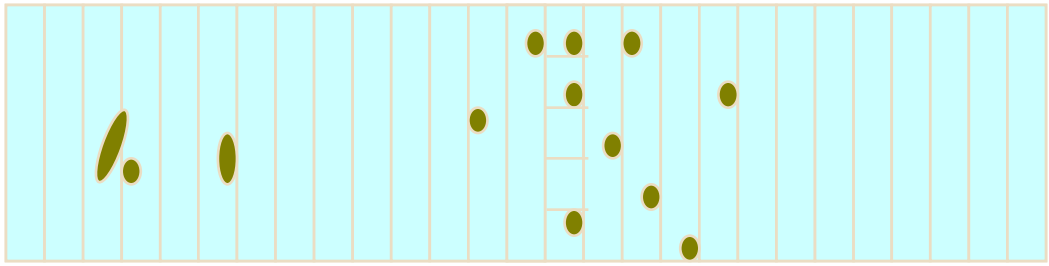
```
G4PVPlacement(G4RotationMatrix* pRot,
  const G4ThreeVector &tlate,
  G4LogicalVolume *pDaughterLogical,
  const G4String &pName,
  G4LogicalVolume *pMotherLogical,
  G4bool pMany, G4int pCopyNo,
  G4bool pSurfChk=false);
```

- If this flag is true, overlap check is done at the construction.
  - Some number of points are 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.

GEOMETRY OPTIMISATION ("VOXELIZATION")

Smart voxelisation

- In case of other Monte Carlo tools, the user had to carefully implement his/her geometry to maximize the performance of geometrical navigation.
- While in Geant4, user's geometry is automatically optimized to most suitable to the navigation. - "Voxelisation"
  - For each mother volume, one-dimensional virtual division is performed.
  - Subdivisions (slices) containing same volumes are gathered into one.
  - Additional division again using second and/or third Cartesian axes, if needed.
- *"Smart voxels"* are computed at initialisation time
  - When the detector geometry is *closed*
  - Does not require large memory or computing resources
  - At tracking time, searching is done in a hierarchy of virtual divisions



Detector description tuning

- Some geometry topologies may require 'special' tuning for ideal and efficient optimisation
  - for example: a dense nucleus of volumes included in very large mother volume
- Granularity of voxelisation can be explicitly set
  - Methods **`Set/GetSmartless()`** from **`G4LogicalVolume`**
- Critical regions for optimisation can be detected
  - Helper class **`G4SmartVoxelStat`** for monitoring time spent in detector geometry optimisation
    - Automatically activated if **`/run/verbose`** greater than 1

```
Percent     Memory     Heads    Nodes   Pointers    Total CPU    Volume
-------     ------     -----    -----   --------    ---------    -----------
  91.70         1k         1       50         50         0.00    Calorimeter
   8.30         0k         1        3          4         0.00    Layer
```

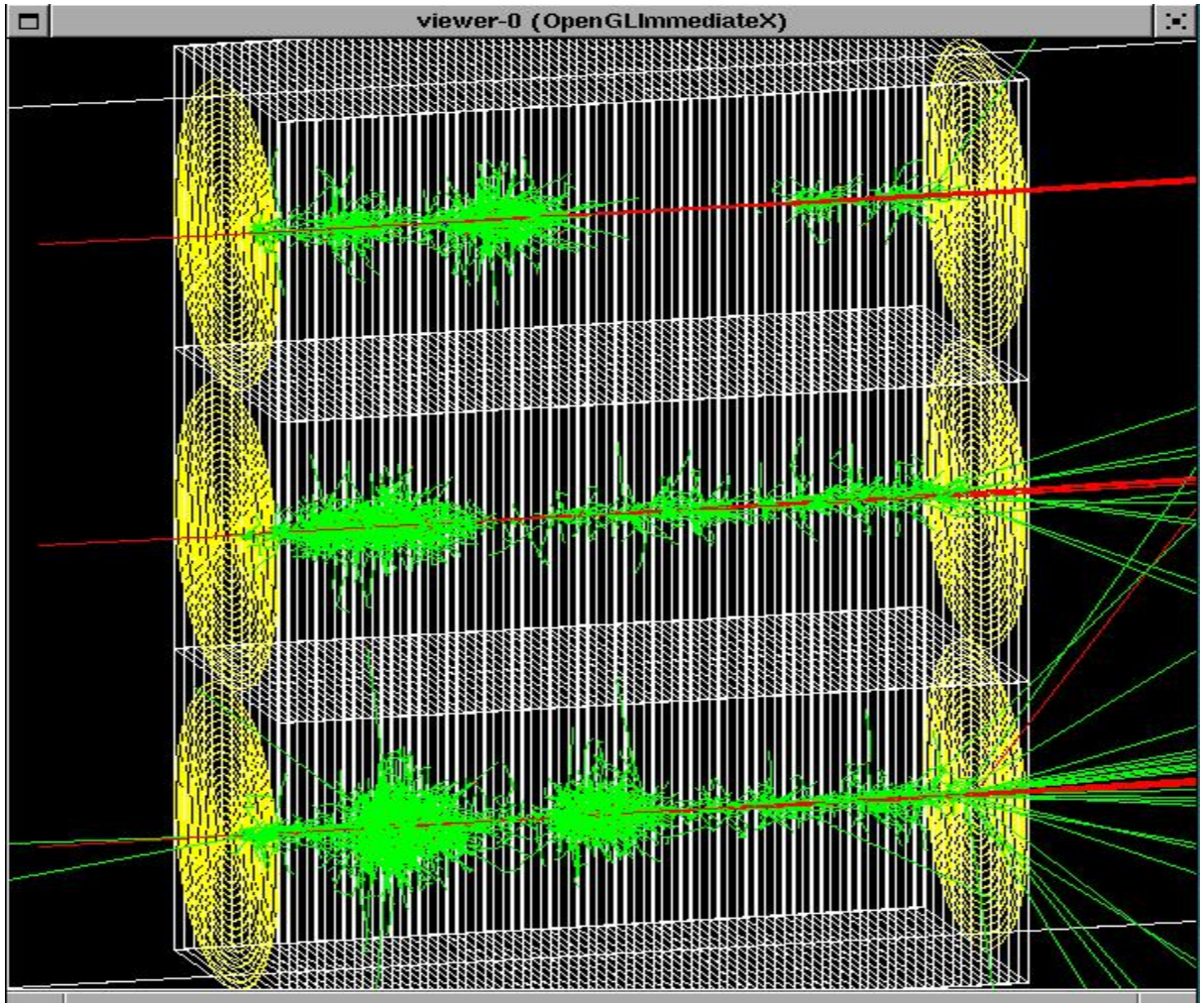
PARALLEL GEOMETRY

Parallel navigation

- Occasionally, it is not straightforward to define sensitivity, importance or envelope to be assigned to volumes in the mass geometry.
  - Typically a geometry built machinery by CAD, GDML, DICOM, etc. has this difficulty.
- New parallel navigation functionality allows the user to define more than one worlds simultaneously.
  - New G4Transportation process sees all worlds simultaneously.
  - A step is limited not only by the boundary of the mass geometry but also by the boundaries of parallel geometries.
  - 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.

exampleN07

- Mass geometry
  - sandwich of rectangular absorbers and scintilators
- Parallel scoring geometry
  - Cylindrical layers



MOVING OBJECTS

Moving objects

- In some applications, it is essential to simulate the movement of some volumes.
  - E.g. particle therapy simulation
- Geant4 can deal with moving volume
  - In case speed of the moving volume is slow enough compared to speed of elementary particles, so that you can assume the position of moving volume is still within one event.
- Two tips to simulate moving objects :
  1. Use parameterized volume to represent the moving volume.
  2. Do not optimize (voxelize) the mother volume of the moving volume(s).

Moving objects - tip 2

- Do not optimize (voxelize) the mother volume of the moving volume(s).
  - If moving volume gets out of the original optimized voxel, the navigator gets lost.

motherLogical -> SetSmartless(number_of_daughters);

  - With this method invocation, the one-and-only optimized voxel has all daughter volumes.
  - For the best performance, use hierarchal geometry so that each mother volume has least number of daughters.

4D RT Treatment Plan

Source: Lei Xing, Stanford University