IEEE Nuclear Science Symposium and Medical Imaging Conference

Short Course

Simulation Techniques Using Geant4

Maria Grazia Pia (*INFN Genova, Italy)*
MariaGrazia.Pia@ge.infn.it

Dresden, 18 October 2008

http://www.ge.infn.it/geant4/events/nss2008/geant4course.html

This course exploits training material developed by several Geant4 Collaboration members: thanks to all of them!

Geometry

Basic concepts of Geant4 geometry

Special thanks to Gabriele Cosmo for creating the original material for Detector Description to be used in Geant4 training

Creating a Detector Volume

Recipe

- Start with its **shape** and size
  - Box 3x5x7 cm3, sphere R=8 m
- Add **properties**
  - material, magnetic/electric field, etc.
  - make it sensitive
- **Place** it in another volume
  - in one place
  - repeatedly using a function

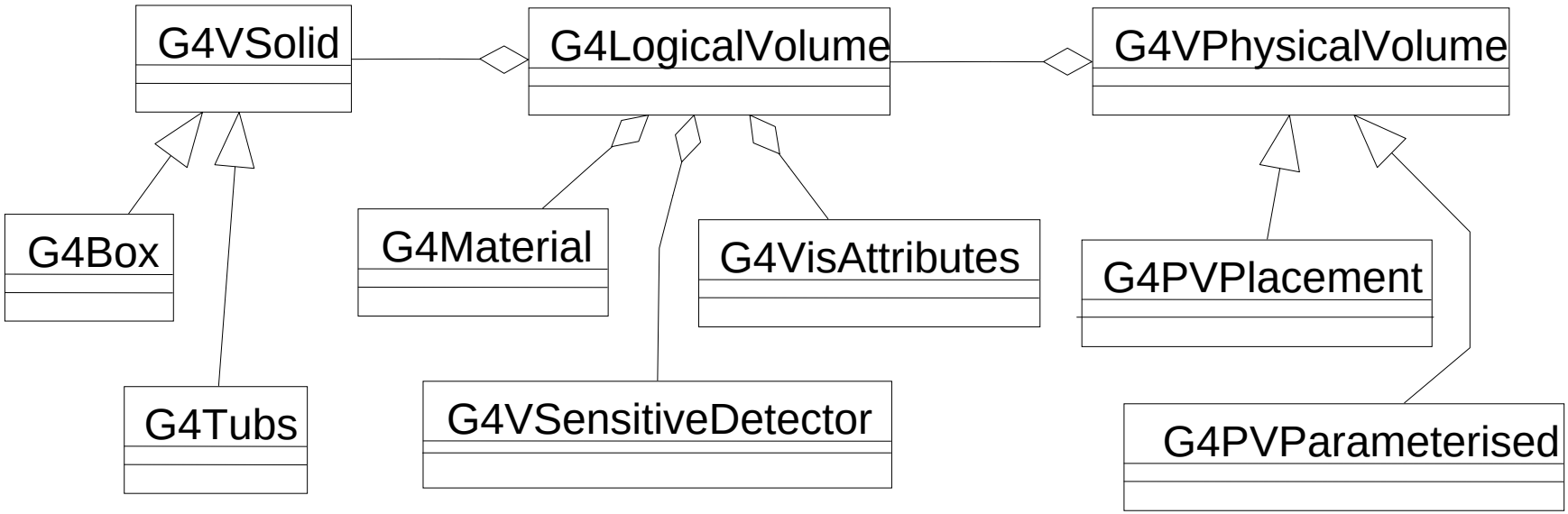
Solid

Logical Volume

Physical Volume

Define 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

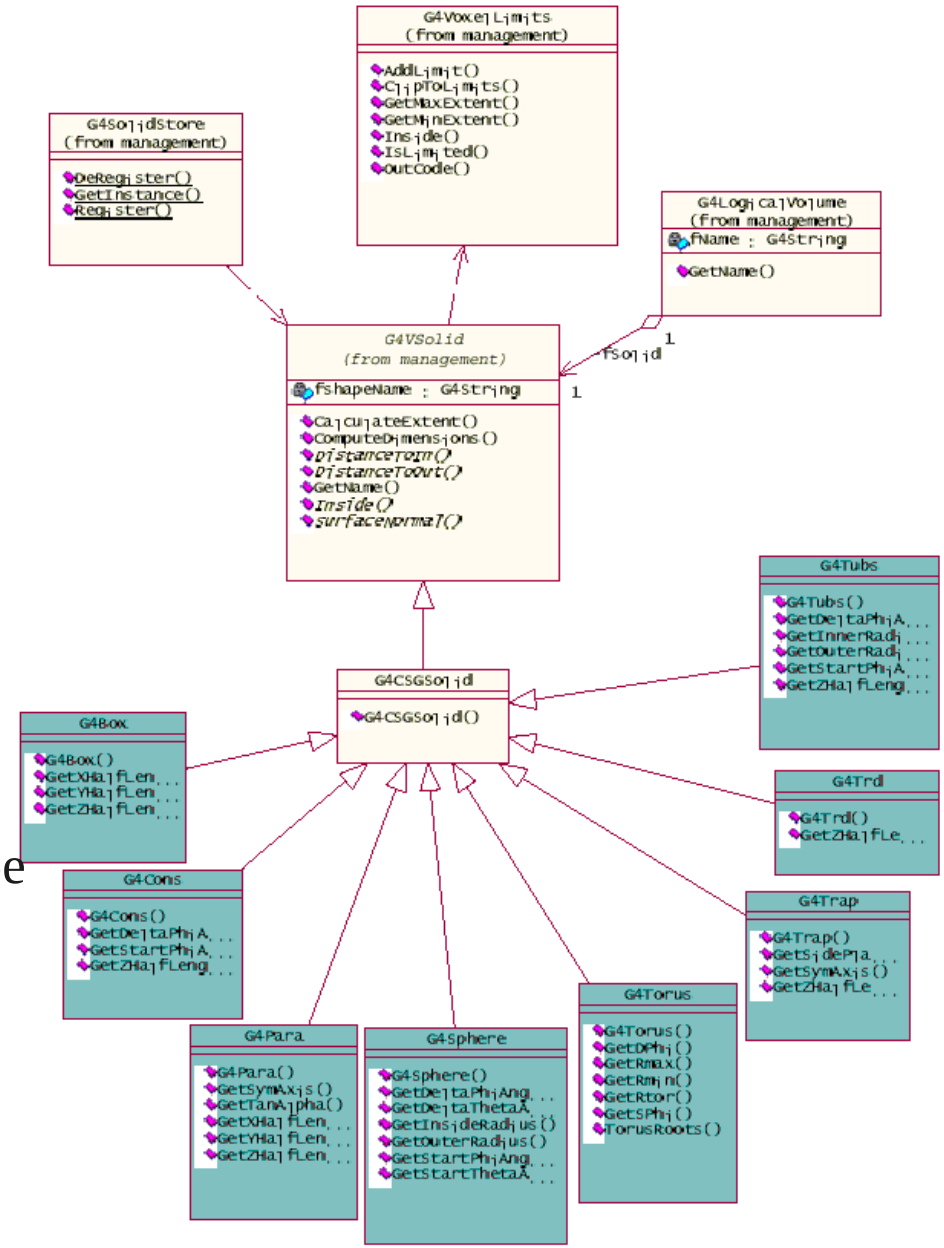
```
G4VSolid* boxSolid =
  new G4Box(“aBoxSolid”, 1.*m, 2.*m, 3.*m);
G4LogicalVolume* boxLogical =
  new G4LogicalVolume( boxSolid, boxMaterial,
                       “aBoxLog”, 0, 0, 0);
G4VPhysicalVolume* boxPhysical =
  new G4PVPlacement( rotation,
                     G4ThreeVector(posX, posY, posZ),
                     boxLogical, “aBoxPhys”, motherLogical,
                     0, copyNo);
```

- A unique physical volume which represents the experimental area must exist and fully contain all other components
  - The world volume

Solids

G4VSolid

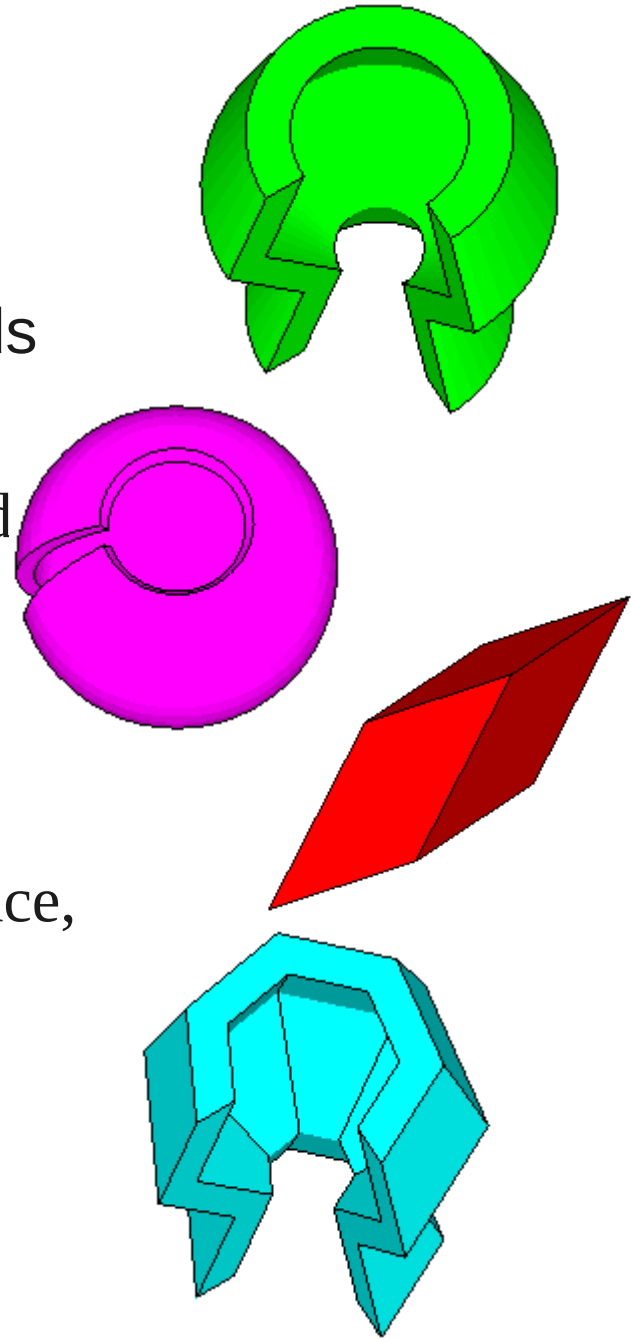
- Abstract class
  - All solids in Geant4 derive from it
  - All solids can be treated transparently in Geant4 kernel through their base class
- Defines but does not implement all functions required to:
  - compute distances to/from the shape
  - check whether a point is inside the shape
  - compute the extent of the shape
  - compute the surface normal to the shape at a given point



Solids

Solids defined in Geant4:

- **CSG** (Constructed Solid Geometry) solids
  - G4Box, G4Tubs, G4Cons, G4Trd, ...
  - Analogous to simple GEANT3 CSG solid
- **Specific solids** (CSG like)
  - G4Polycone, G4Polyhedra, G4Hype, ...
- **BREP** (Boundary REPresented) solids
  - G4BREPSolidPolycone, G4BSplineSurface, ...
  - Any order surface
- **Boolean solids**
  - G4UnionSolid, G4SubtractionSolid, ...



CSG: G4Tubs, G4Cons

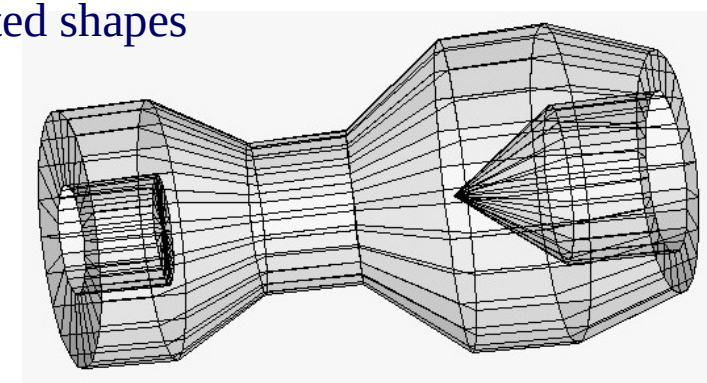
These are simple shapes you would probably like to exercise in your first application

```
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

G4Cons(const G4String& pname,  // name
             G4double  pRmin1, // inner radius -pDz
             G4double  pRmax1, // outer radius -pDz
             G4double  pRmin2, // inner radius +pDz
             G4double  pRmax2, // outer radius +pDz
             G4double  pDz,    // Z half length
             G4double  pSphi,  // starting Phi
             G4double  pDphi); // segment angle
```

Specific CSG Solids: G4Polycone

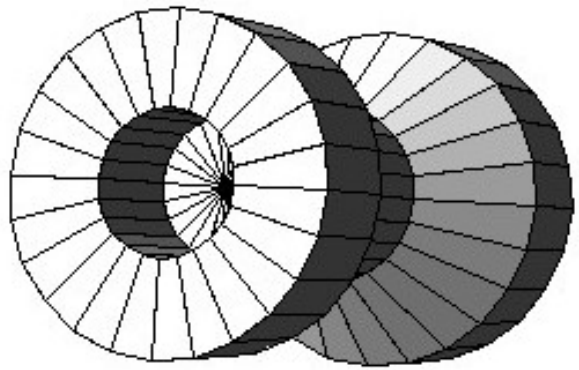
…then you may want to play with more complicated shapes



```
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

- Additional constructor using planes



Logical Volume

Logical Volume

- 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

G4LogicalVolume

To create a logical volume for a given material and solid, the user must instantiate a G4LogicalVolume:

```
G4LogicalVolume(G4VSolid* solid,
                G4Material* material,
                const G4String& name,
                G4FieldManager* fieldManager=0,
                G4VSensitiveDetector* sensitiveDet=0,
                G4UserLimits* userLimits=0,
                G4bool optimise=true);
```

- The pointers to solid and material should NOT be null

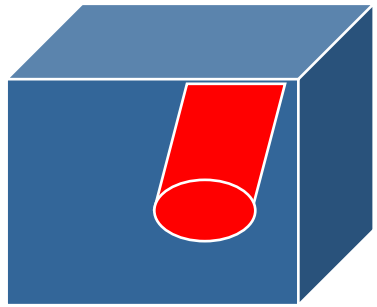
Placing volumes: Physical Volumes

Physical Volumes

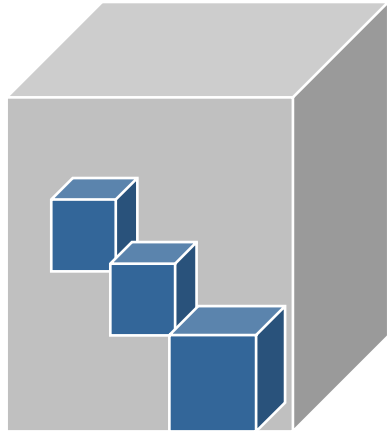
- Physical volumes are **placed** instances of volumes
- Several techniques can be used:
  - Single placement
  - Repeated placement (replicas, parameterization, ...)
- Volumes are part of a **geometrical hierarchy**:
  - Volumes always have a mother volume (except the world volume)
  - Volumes may have several daughter volumes
- **G4VPhysicalVolume** is the base class of physical volumes
  - Use the **derived classes** to place your logical volumes

Placing Physical Volumes

- **Placement:** one positioned volume



- **Repeated:** a volume placed many times
  - reduces use of memory
  - **Replica**
    - simple repetition, similar to GEANT 3 divisions
  - **Parameterised**



- A **mother** volume can contain **either**
  - **many placement** volumes
  - **OR**
  - **one repeated** volume

G4VPhysicalVolume

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

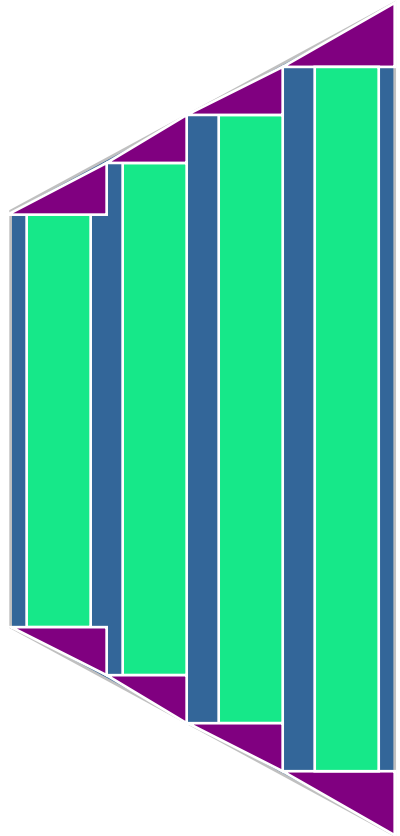
Placement

```
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:
  - Variants for greater flexibility

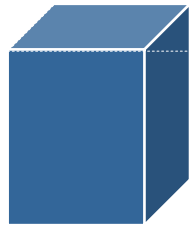
Parameterised Physical Volumes

- One can place a volume multiple times, where position and dimension are parameterised w.r.t. the copy number
- Require a **user-supplied function** to specify the parameterization
- User functions define:
  - the **size** of the solid (dimensions)
  - where it is **positioned** (transformation)
- Optional:
  - the **type** of the solid
  - the **material**
- **Limitations**
  - Applies to simple CSG solids only
  - Daughter volumes allowed only for special cases

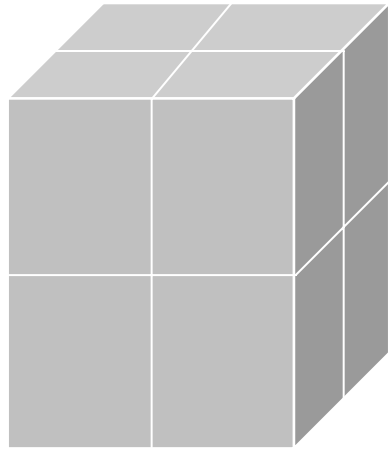


Replicated Physical Volumes

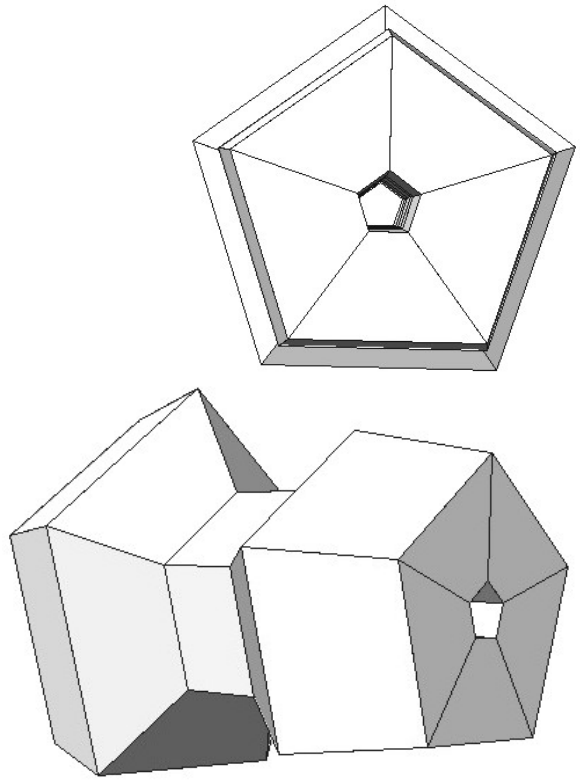
- The mother volume is sliced into replicas, all of the same shape and size



- Represents many detector elements differing only in their positioning
- Replication may occur along:
  - Cartesian axes (X, Y, Z)
  - Radial axis (Rho)
  - Phi axis (Phi)



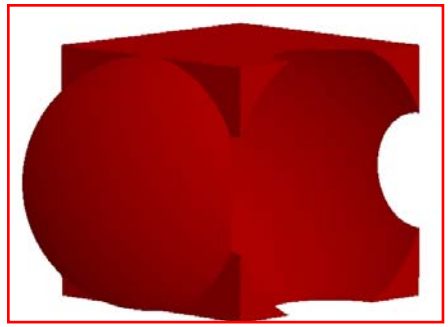
BREPS:
G4BREPSolidPolyhedra



```
G4BREPSolidPolyhedra(const G4String& pName,
                     G4double phiStart,
                     G4double phiTotal,
                     G4int sides,
                     G4int nZplanes,
                     G4double zStart,
               const G4double zval[],
               const G4double rmin[],
               const G4double rmax[]);
```

- `sides` - numbers of sides of each polygon in the `x-y` plane
- `nZplanes` - numbers of planes perpendicular to the `z` axis
- `zval[]` - `z` coordinates of each plane
- `rmin[], rmax[]` - Radii of inner and outer polygon at each plane

Boolean Solids



- Solids can be **combined** **using** **boolean operations**:
  - **`G4UnionSolid, G4SubtractionSolid, G4IntersectionSolid`**
  - Requires: 2 solids, a boolean operation, an (optional) *transformation* for the 2nd solid
  - The 2nd solid is positioned relative to the coordinate system of the 1st solid
  - The result of boolean operation becomes a solid
  - Solids to be combined can be either CSG or other Boolean solids
- Note: tracking cost for navigation in a complex Boolean solid is proportional to the number of constituent CSG solids

G4UnionSolid

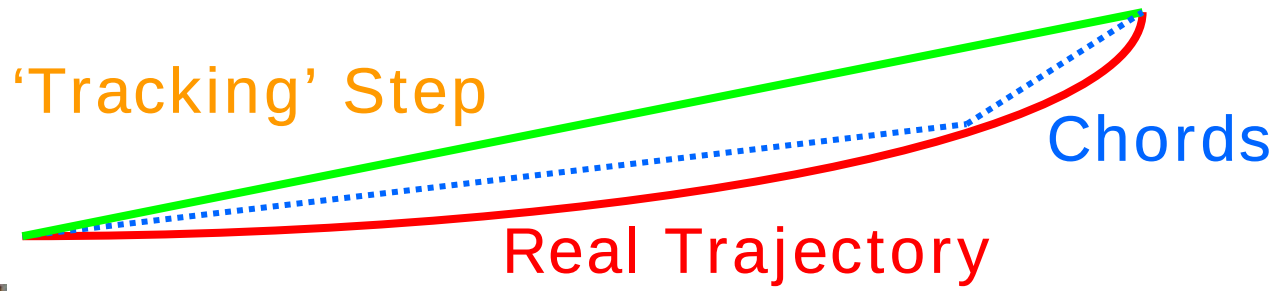
G4SubtractionSolid

G4IntersectionSolid

Electric and Magnetic Field

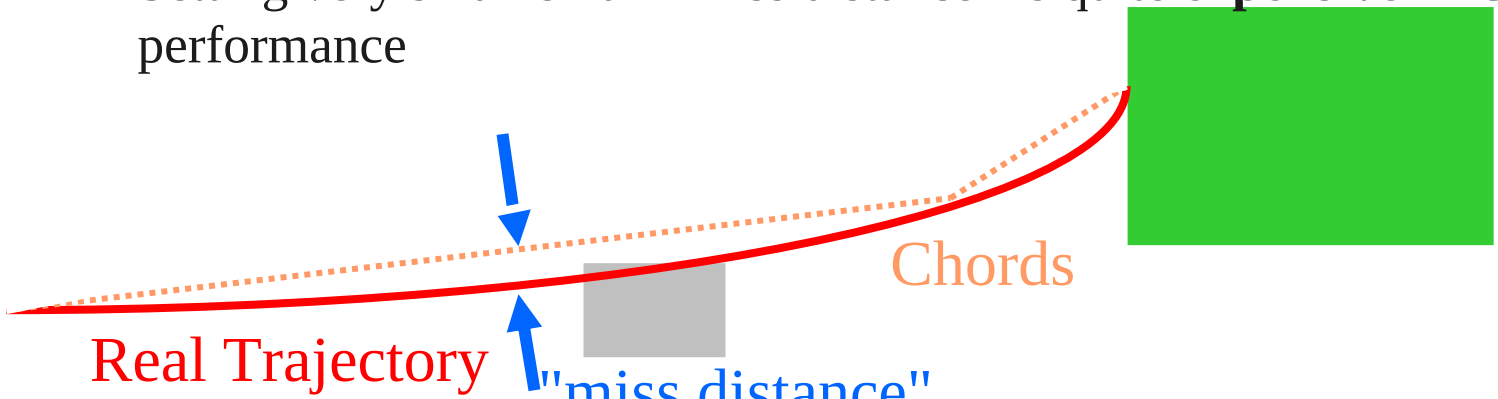
Field integration

- To propagate a particle inside a field, solve the equation of motion of the particle in the field
  - (e.g. using the Runge-Kutta method for the integration of the equations of motion)
- In specific cases other solvers can also be used:
  - In a uniform field, using the analytical solution
  - In a smooth but varying field, with RK+helix
- Using the method to calculate the track's motion in a field, Geant4 breaks this curved path into linear chord segments
  - The chord segments are determined so that they closely approximate the curved path



Tracking in field

- Use the chords to interrogate `G4Navigator`, to see whether the track has crossed a volume boundary
- One tracking step can create several chords
  - In some cases, one step consists of several helix turns
- The user can set the accuracy of the volume intersection
  - By setting a parameter called the **"miss distance"**
    - It is a measure of the error in whether the approximate track intersects a volume
    - Setting very small small "miss distance" is quite **expensive in CPU** performance



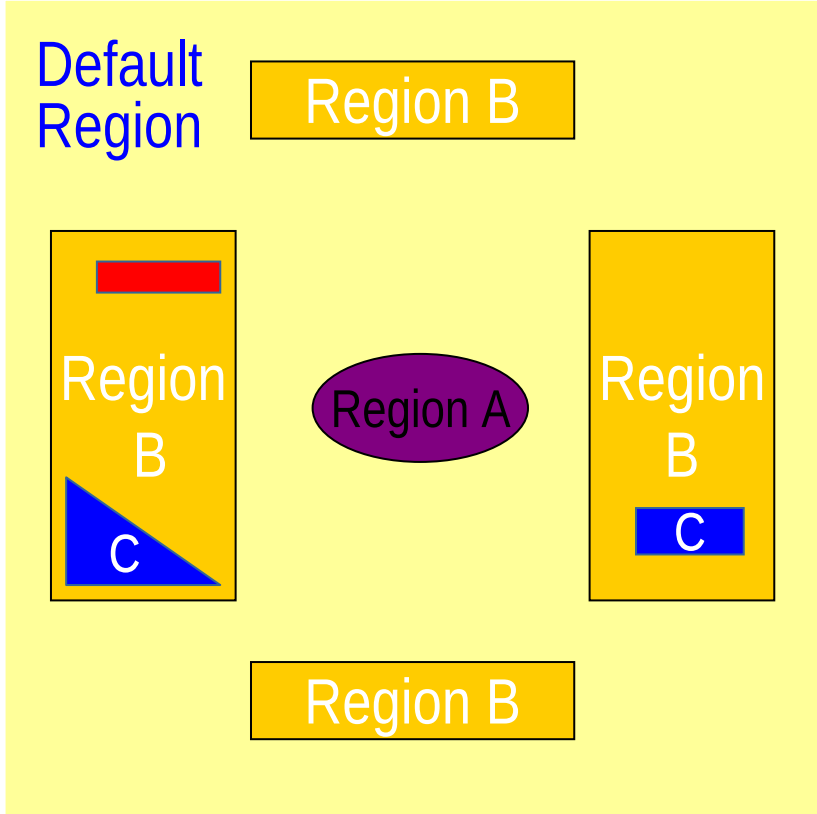
Regions

Cuts per Region

- Geant4 originally had a **unique production threshold** ('cut') expressed in length (i.e. minimum range of secondary)
  - For *all volumes,* but possibly different for each particle (e^+, e^-, γ-rays)
  - It can still be used in this mode
- **Appropriate length scales** can vary greatly between different areas of a large detector
  - E.g. the spatial resolution of a vertex detector (order of μm) and a muon detector (order of cm)
  - Having a unique (small) cut can create a **performance penalty**
- Geant4 allows for several cuts
  - Globally or per particle
  - Enabling the **tuning of production thresholds** at the level of a **region**
  - Cuts are applied only for γ, electron and positron and only for processes which have infrared divergence (e.g. soft δ-rays or soft Bremsstrahlung photons)

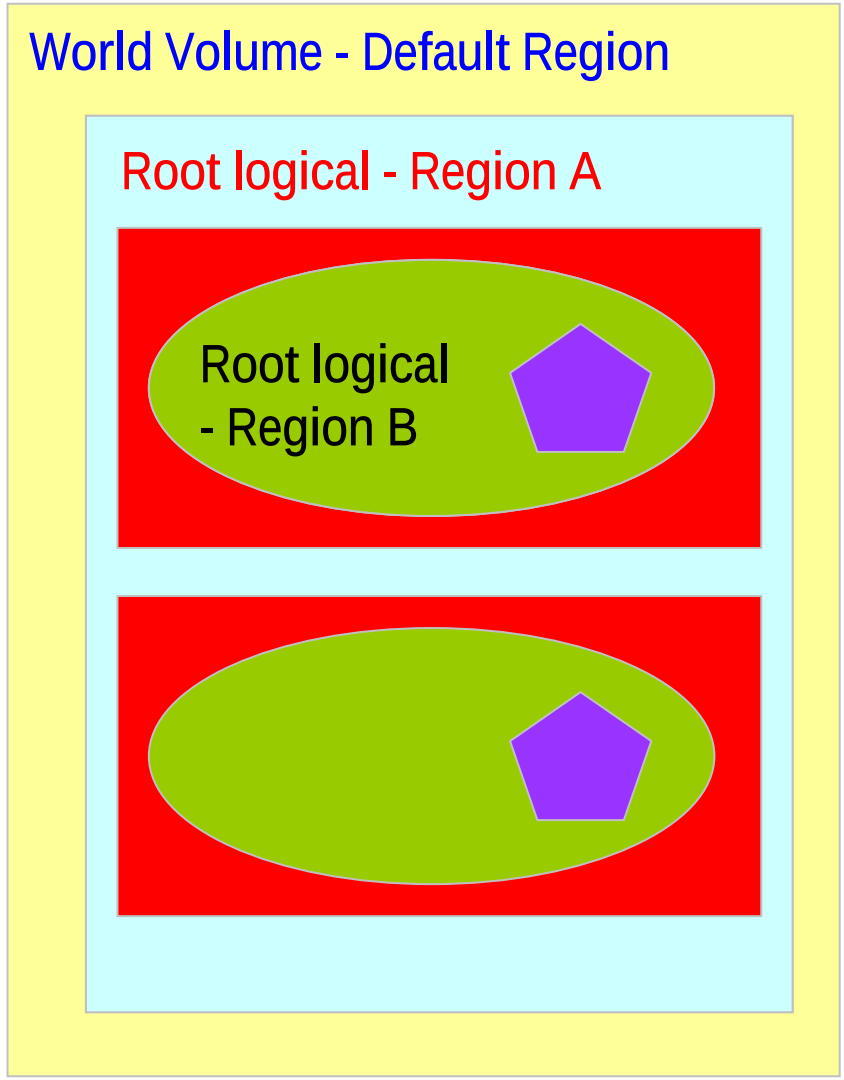
Detector Region

- Concept of region:
  - Set of geometry volumes, typically of a sub-system
    - barrel + end-caps of a calorimeter
    - areas of support structures
    - Or any group of volumes
- A **set of cuts** in range is **associated** to a **region**
  - a *different range cut* for each particle among γ, e-, e+ is allowed in a region



Region and cut

- Each region has its **unique set of cuts**
- The World volume is recognized as the default region
  - The default cuts defined in PhysicsList are used for it
- A logical volume becomes a root logical volume once it is assigned to a region
  - **All daughter** volumes belonging to the root logical volume share the same region (and cut), unless a daughter volume itself becomes another root
- Important restriction :
  - No logical volume can be shared by more than one region, regardless of root volume



Region and Cuts – How to use

```
G4Region* emCalorimeter = new G4Region("EM-Calorimeter");
emCalorimeter->AddRootLogicalVolume(emCalorimeterLV);
[...]
G4String regionName = “EM-calorimeter";

G4Region* region=G4RegionStore::GetInstance()->GetRegion(regionName);

cuts = new G4ProductionCuts;

cuts->SetProductionCut(0.01*mm,G4ProductionCuts::GetIndex("gamma"));
cuts->SetProductionCut(0.1*mm,G4ProductionCuts::GetIndex("e-"));
cuts->SetProductionCut(0.1*mm,G4ProductionCuts::GetIndex("e+"));

region-> SetProductionCuts cuts;
```

create a region

attach a logical volume to the region

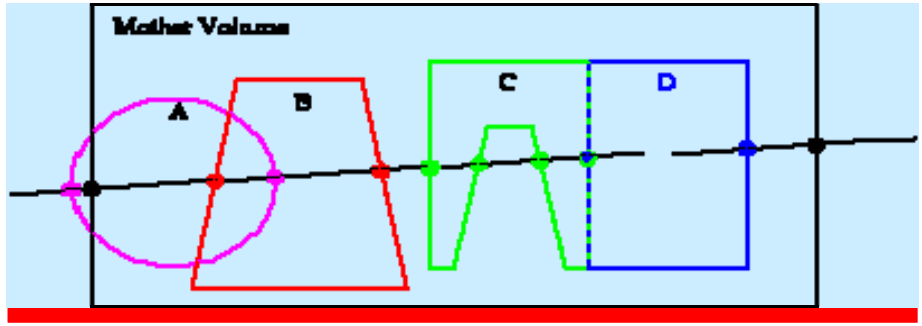
retrieve a region

create production cuts

attach cuts to the region

Debugging a geometrical description in a user application

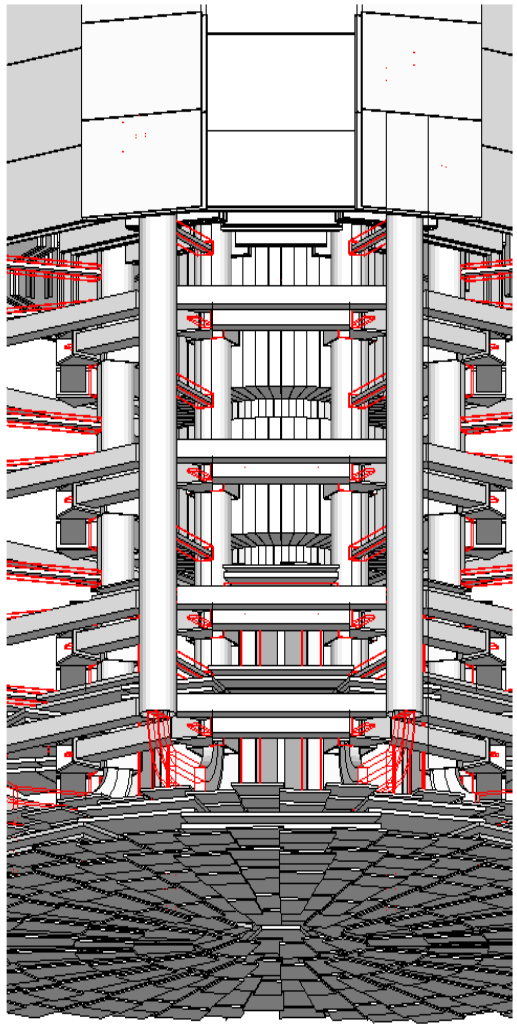
Debugging geometries



- An *overlapping* volume is a contained volume which actually **protrudes** from its mother volume
- Geant4 does not allow for malformed geometries:
  - Otherwise a point in space cannot be univocally assigned to a physical volume
- Utilities are provided for detecting wrong positioning
  - **Graphical tools**
  - **Kernel commands and built-in utilities**

Debugging tools: DAVID

- DAVID is a graphical debugging tool for detecting potential intersections of volumes
- The accuracy of the graphical representation can be tuned to the exact geometrical description
  - physical-volume surfaces are automatically decomposed into 3D polygons
  - intersections of the generated polygons are parsed
  - If a polygon intersects with another one, the physical volumes associated to these polygons are highlighted in color (**red** is the default).
- DAVID can be downloaded from the Web as external tool for Geant4
  - http://geant4.kek.jp/GEANT4/vis/DAWN/About_DAVID.html



Debugging run-time commands

- Built-in run-time commands to activate verification tests for the user geometry
- Tests can be applied recursively to all depth levels (would require CPU time!): `[recursion_flag]`

```
Idle> geometry/test/run [recursion_flag] or

Idle> geometry/test/grid_test [recursion_flag]
```

to start verification of geometry for overlapping regions based on a standard grid setup

```
Idle> geometry/test/line_test [recursion_flag]
```

to shoot a line along a specified direction and position

```
Idle>geometry/test/position and geometry/test/direction
```

to specify position & direction for the `line_test`

- Resolution/dimensions of grid/cylinders can be tuned

Debugging commands

- An other tool to check for overlaps is the **`CheckOverlap()`** method of `G4VPhysicalVolume`

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

```
myPhysicalVolume->CheckOverlap();
```

- It is convenient to run this tool when the geometry is complete (all volumes defined)
  - retrieving all the Physical Volumes from `G4PhysicalVolumeStore()`