IV INTERNATIONAL GEANT4 SCHOOL

Belgrade
23 – 28 October 2016

Detector description: materials and geometry

G. Milluzzo, J. Pipek,
F. Romano, ...

INFN Geant4

Geant 4 tutorial

Introduction

Mandatory user classes in a Geant4:

- **G4VUserDetectorConstruction**
- **G4VUserPhysicsList**
- **G4VUserPrimaryGeneratorAction**

Materials, ***Volumes***, ***Sensitive detectors*** *and* ***Fields*** *to be used in the simulation must be defined in a daughter class of* ***G4VUserDetectorConstruction****.*

Part I: Units & materials

- System of units & constants
- Definition of elements
- Materials and mixtures
- NIST database

Units in Geant4

- **Geant4 has no default unit!**
  - **When specifying dimensions, always* multiply by an appropriate unit:**

```
G4double width = 12.5 * m;
G4double density = 2.7 * g/cm3;
```

  - **Most common units are defined in CLHEP:**

► G4SystemOfUnits.hh

► CLHEP/Units/SystemOfUnits.h

  - **You can define new units (later)**

- **Output data in terms of a specific unit:**
  - **divide a value by the unit:**

```
G4cout << dE / MeV << " (MeV)" << G4endl;
```

System of Units

Basic units defined in CLHEP:

- nanosecond (ns)
- megaelectronvolt (MeV)
- millimetre (mm)
- kelvin
- unit charge (eplus)
- radian
- steradian
- candela

- All other units derived from the basic ones.

- **Useful feature:** Geant4 can select the most appropriate unit to use
  - specify the *category* for the data (**Length**, **Time**, **Energy**, etc…):

```
G4cout << G4BestUnit(StepSize, "Length");
```

StepSize will be printed in **km**, **m**, **mm** or … **fermi**, depending on its actual value

How to define a new unit

- New units can be defined directly as constants, or (suggested way) as a **G4UnitDefinition**:

```
G4UnitDefinition(name, symbol, category, value);
```

- Example (speed):

```
new G4UnitDefinition("km/hour", "km/h", "Speed", km/(3600*s));
```

  - The new category "**Speed**" will be registered in the kernel in **G4UnitsTable** (if not present)

► G4UnitsTable.hh/.cc

- **Define it in each thread!** => e.g. in run action
- To print the list of units:
  - From the code:

```
G4UnitDefinition::PrintUnitsTable();
```

  - At run-time, as UI command:

```
/units/list
```

Materials

- Different kinds of materials can be defined:
  - Isotopes → `G4Isotope`
  - Elements → `G4Element`
  - Molecules → `G4Material`
  - compounds and mixtures → `G4Material`
- `G4Isotope` and `G4Element` describe properties of the *atoms*:
  - Atomic number, number of nucleons, mass of a mole, shell energies, cross-sections per atoms, etc...
- `G4Material` describes the *macroscopic* properties of the matter:
  - temperature, pressure, state, density
  - Radiation length, absorption length, etc...
- `G4Material` is used by tracking, geometry and physics

Elements and isotopes

- Isotopes…

```
G4Isotope(const G4String& name,
                 G4int     z,    // atomic number
                 G4int     n,    // number of nucleons
                 G4double  a );  // mass of mole
```

- …can be assembled into elements as follows:

```
G4Element (const G4String& name,
           const G4String& symbol,   // element symbol
                 G4int     nIso );   // n. of isotopes
```

```
G4Element::AddIsotope(G4Isotope* iso,   // isotope
             G4double relAbund);        // Relative abundance
```

Materials of one element and molecules

- Single element material definition:

```
G4double z, a, density;
density = 1.390*g/cm3;
a = 39.95*g/mole;
G4Material* lAr = new G4Material("liquidAr", z=18, a,
density);
```

- A molecule material (composition by number of atoms):

```
a = 1.01*g/mole;
G4Element* elH = new G4Element("Hydrogen", symbol="H", z=1.,
a);

a = 16.00*g/mole;
G4Element* elO = new G4Element("Oxygen", symbol="O", z=8., a);

density = 1.000*g/cm3;
G4Material* H2O = new G4Material("Water", density,
ncomponents=2);
H2O->AddElement(elH, natoms=2);
H2O->AddElement(elO, natoms=1);
```

Compounds and mixtures

- Composition by fraction of mass:

```
a = 14.01*g/mole;
G4Element* elN  = new G4Element(name="Nitrogen",symbol="N", z= 7., a);
a = 16.00*g/mole;
G4Element* elO  =  new G4Element(name="Oxygen",symbol="O", z= 8., a);
density = 1.290*mg/cm3;
G4Material* Air = new G4Material(name="Air", density, ncomponents=2);
Air->AddElement(elN, 70.0*perCent);
Air->AddElement(elO, 30.0*perCent);
```

- Composition of compound materials:

```
G4Element* elC  = …;   // define “carbon” element
G4Material* SiO2 = …;  // define “quartz” material
G4Material* H2O = …;   // define “water” material
density = 0.200*g/cm3;

G4Material* aerogel = new G4Material("Aerogel",
                                       density, ncomponents=3);
aerogel->AddMaterial(SiO2,fractionmass=62.5*perCent);
aerogel->AddMaterial(H2O, fractionmass=37.4*perCent);
aerogel->AddElement (elC, fractionmass= 0.1*perCent);
```

Example: gas

- It may be necessary to specify temperature and pressure
  - (**dE/dx** computation affected)

```
G4double density = 27. * mg/cm3;
G4double temperature = 325. * kelvin;
G4double pressure = 50. * atmosphere;

G4Material* CO2 = new G4Material("CO2Gas", density,
    ncomponents=2, kStateGas, temperature, pressure);
CO2->AddElement(C, natoms = 1);
CO2->AddElement(O, natoms = 2);
```

- Absolute vacuum does not exist: gas at very low density !
  - Cannot define materials composed of multiple elements through **Z** or **A**, or with ρ=**0**

```
G4double atomicNumber = 1.;
G4double massOfMole = 1.008*g/mole;
G4double density = 1.e-25*g/cm3;
G4double temperature = 2.73*kelvin;
G4double pressure = 3.e-18*pascal;
G4Material* Vacuum = new G4Material("interGalactic",
    atomicNumber, massOfMole, density,
    kStateGas,temperature, pressure);
```

Use the NIST material database

- No need to predefine elements and materials
- Retrieve materials from NIST manager:

```
G4NistManager* manager = G4NistManager::Instance();
G4Material* air = manager->FindOrBuildMaterial("G4_AIR");
G4Material* H2O = manager->FindOrBuildMaterial("G4_WATER");
G4Material* vacuum=manager->FindOrBuildMaterial("G4_Galactic");
```

- UI commands:

```
/material/nist/printElement
```

← print defined elements

```
/material/nist/listMaterials
```

← print defined materials

► G4NistManager.hh

NIST Material Data-Base in Geant4

- NIST database for materials is imported inside Geant4

  http://physics.nist.gov/PhysRefData
- UI commands specific for handling materials
- The best accuracy for the most relevant parameters guaranteed:
  - Density
  - Mean excitation potential
  - Chemical bounds
  - Element composition
  - Isotope composition
  - Various corrections
- Database imported in Geant4 can be found at :

http://geant4.web.cern.ch/geant4/UserDocumentation/UsersGuides/ForApplicationDeveloper/html/apas06.html

```
Z     A      m      error     (%)       Aeff
====================================
14    Si  22   22.03453        (22)
      28.0855(3)
            23   23.02552        (21)
            24   24.011546      (21)
            25   25.004107      (11)
            26   25.992330      (3)
            27   26.98670476   (17)
            28   27.9769265327  (20)
      92.2297 (7)
            29   28.97649472    (3)
      4.6832 (5)
            30   29.97377022    (5)
      3.0872 (5)
            31   30.97536327    (7)
            32   31.9741481    (23)
            33   32.978001      (17)
            34   33.978576      (15)
            35   34.984580      (40)
            36   35.98669        (11)
            37   36.99300        (13)
            38   37.99598        (29)
            39   39.00230        (43)
            40   40.00580        (54)
            41   41.01270        (64)
            42   42.01610        (75)
```

- *Natural isotope compositions*
- *More than 3000 isotope masses*

NIST materials in Geant4

```
===================================
### Elementary Materials from the NIST Data Base
===================================
 Z Name       ChFormula          density(g/cm^3)      I(eV)
===================================
1    G4_H    H_2               8.3748e-05      19.2
2    G4_He                      0.000166322     41.8
3    G4_Li                      0.534                 40
4    G4_Be                      1.848                 63.7
5    G4_B                       2.37                  76
6    G4_C                       2                      81
7    G4_N    N_2               0.0011652        82
8    G4_O    O_2               0.00133151      95
9    G4_F                       0.00158029       115
10   G4_Ne                      0.000838505     137
11   G4_Na                      0.971                149
12   G4_Mg                      1.74                 156
13   G4_Al                      2.6989              166
14   G4_Si                      2.33                 173
```

```
===================================
### Compound Materials from the NIST Data Base
===================================
 N Name    ChFormula     density(g/cm^3)  I(eV)
===================================
13  G4_Adipose_Tissue        0.92        63.2
         1    0.119477
         6    0.63724
         7    0.00797
         8    0.232333
         11   0.0005
         12   2e-05
         15   0.00016
         16   0.00073
         17   0.00119
         19   0.00032
         20   2e-05
         26   2e-05
         30   2e-05
```

- It is also possible to create new material starting from the existing database one, for instance changing the density:

```
density = 1.03*mg/cm3;
G4NistManager* man = G4NistManager::Instance();
G4Material* water2 =
man->BuildMaterialWithNewDensity("Water_1.03","G4_WATER",density);
```

- It is possible to build mixtures of NIST and user-defined materials

Part II: Geometry

- Construction
  - Basic concepts
  - Basic implementation
- Detailed description
  - Solids
  - Logical volumes
  - Physical volumes
  - Regions
- Tools for geometry check
- (Magnetic) fields

Geometry – implementation basics

- Implement a class inheriting from the abstract base class **G4VUserDetectorConstruction**:

```
class MyDetector : public G4VUserDetectorConstruction {
public:
    virtual G4VPhysicalVolume* Construct();        // required

    virtual void ConstructSDAndField();            // optional
    // ...
};
```

- Create an instance in the main program:

```
MyDetector* detector = new MyDetector();
runManager->SetUserInitialization(detector);
```

- **Note:** Split the implementation into more classes & methods! (good programming practice)
  - especially for complex geometries!
- **Note:** you should not delete the **MyDetector** instance! Run manager does that automatically.

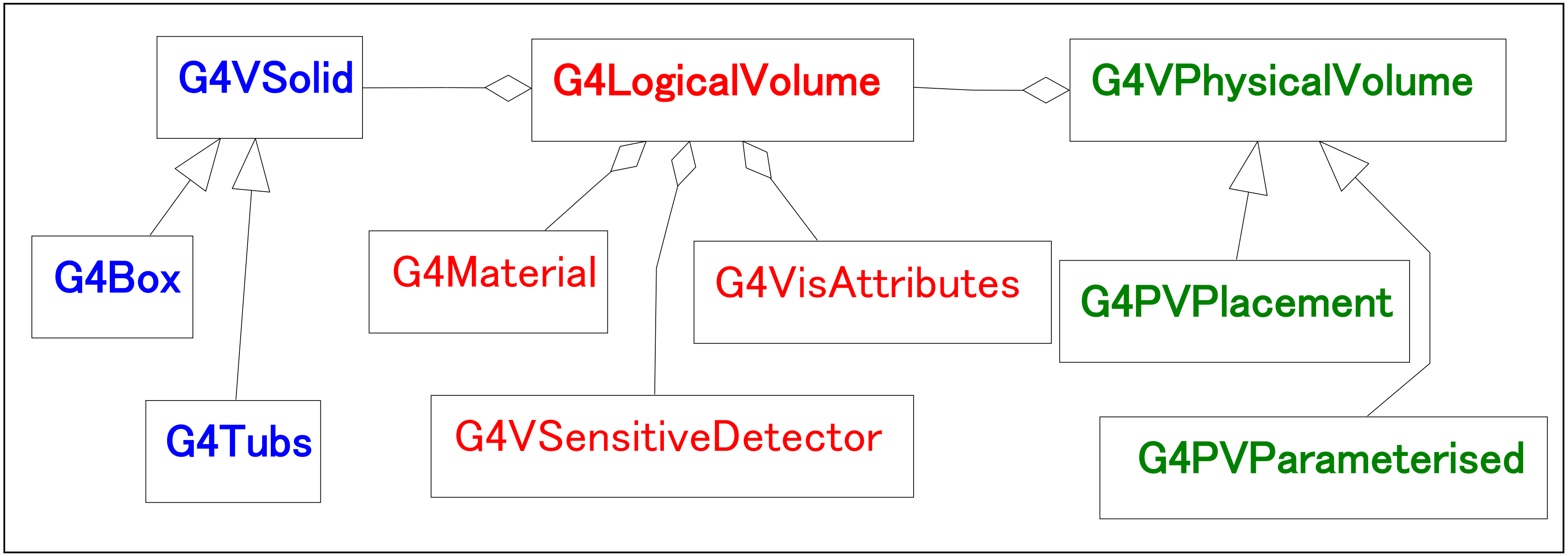
G4VUserDetectorConstruction

- Method **Construct()**
  - Define materials
  - Define solids and volumes of the geometry
  - Build the tree hierarchy of volumes
  - Define visualization attributes
  - Return the world physical volume!

- Method **ConstructSDAndField()** MT **->Thread-local**
  - Assign magnetic field to volumes / regions
  - Define sensitive detectors and assing them to volumes

Detector geometry components

- Three conceptual layers
  - **G4VSolid**: *shape, size*
  - **G4LogicalVolume**: *material, sensitivity, magnetic field, etc.*
  - **G4VPhysicalVolume**: *position, rotation*

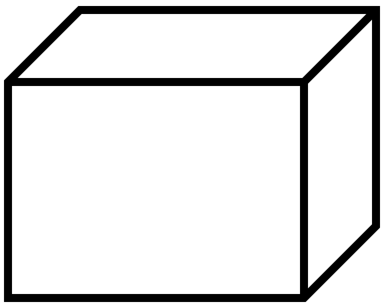


Define detector geometry

- Basic strategy

```
G4VSolid* pBoxSolid =
  new G4Box(“aBoxSolid”,
        1.*m, 2.*m, 3.*m);
```

Solid : shape and size



Step 1
Create the geom. object : box

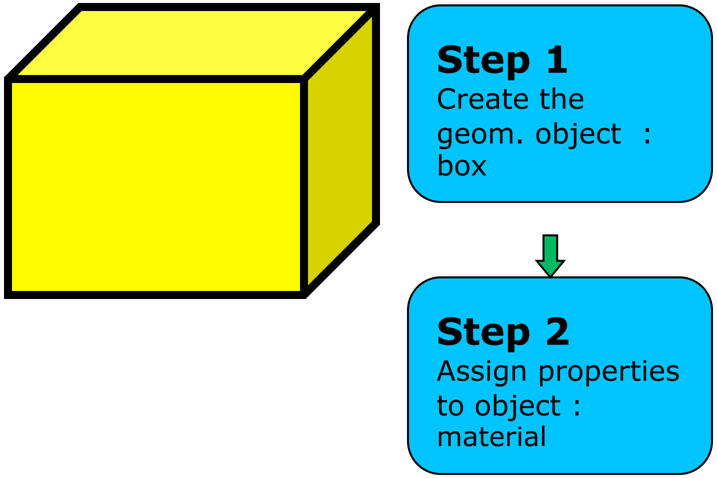
Define detector geometry

- Basic strategy

```
G4VSolid* pBoxSolid =
  new G4Box("aBoxSolid",
        1.*m, 2.*m, 3.*m);

G4LogicalVolume* pBoxLog =
  new G4LogicalVolume( pBoxSolid,
    pBoxMaterial, "aBoxLog", 0, 0, 0);
```

Logical volume :
+ material, sensitivity, etc.



Define detector geometry

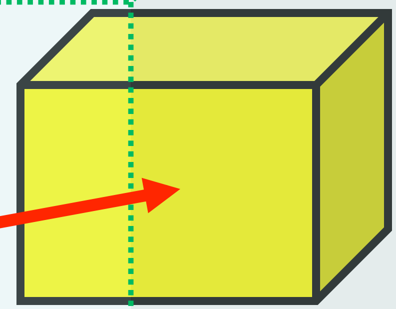
- Basic strategy

Physical volume :
+ rotation and position

```
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);
```



Step 1
Create the geom. object : box

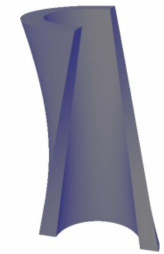
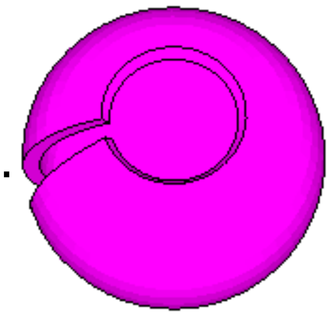
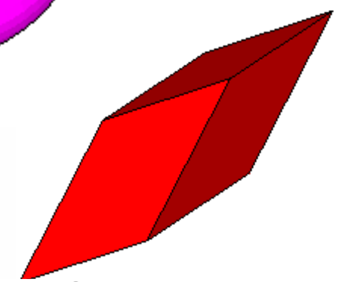
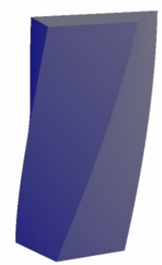
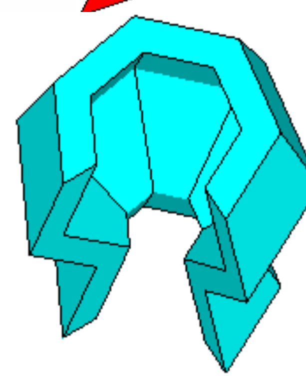
Step 2
Assign properties to object : material

Step 3
Place it in the coordinate system of mother volume

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

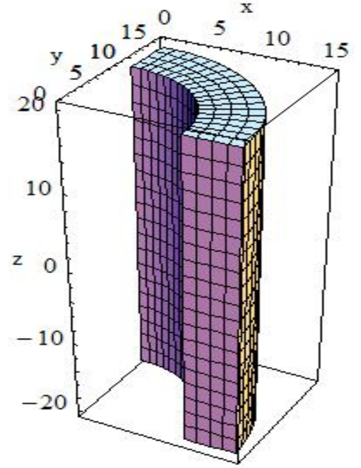
Solid Volumes

- Solids defined in Geant4:
  - CSG (Constructed Solid Geometry) solids
    - G4Box, G4Tubs, G4Cons, G4Trd, …
  - Specific solids (CSG like)
    - G4Polycone, G4Polyhedra, G4Hype, ..
    - G4TwistedTubs, G4TwistedTrap, …
  - BREP (Boundary REPresented) solids
    - G4BREPSolidPolycone, G4BSplineSurface, …
    - Any order surface
  - Boolean solids
    - G4UnionSolid, G4SubtractionSolid,

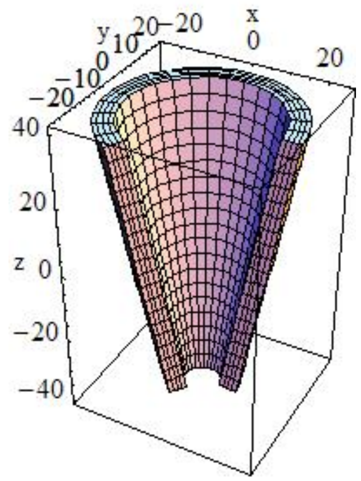







CSG: G4Tubs, G4Cons

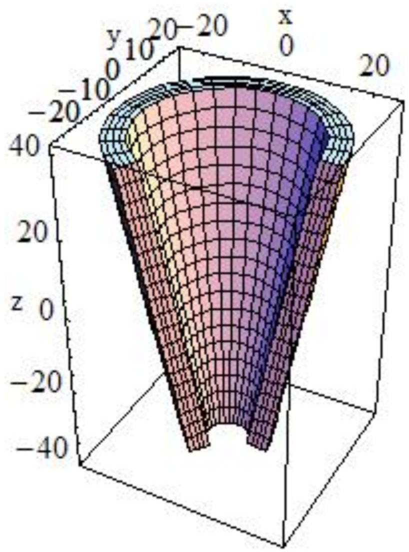
```
G4Tubs(const G4String& pname,  // name
             G4double  pRmin,  // inner radius
             G4double  pRmax,  // outer radius
             G4double  pDz,    // Z half length
             G4double  pSphi,  // starting Phi
             G4double  pDphi); // segment angle
```



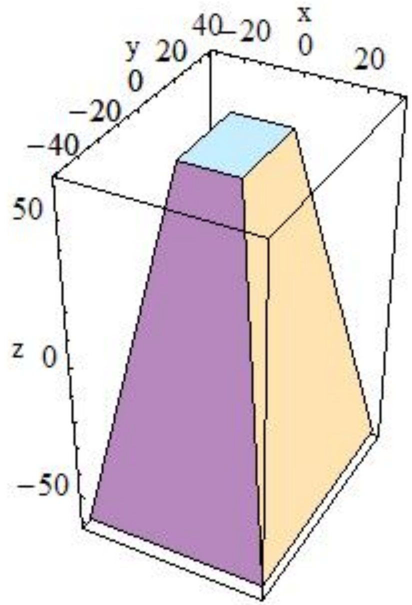

```
G4Cons(const G4String& pname,  // name
             G4double  pRmin1, // inner radius -pDz
             G4double  pRmax1, // outer radius -pDz
             G4double  pRmin2, // inner radius +pDz
             G4double  pRmax2, // outer radius +pDz
             G4double  pDz,    // Z half length
 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



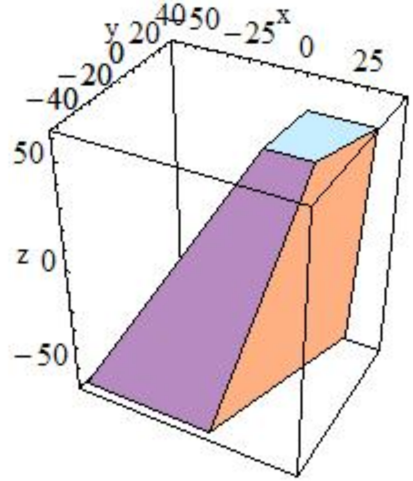
Other CSG solids



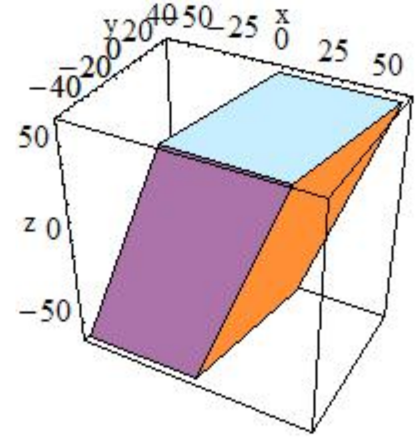
G4Cons



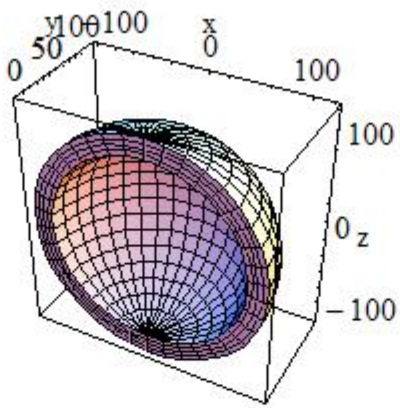
G4Trd



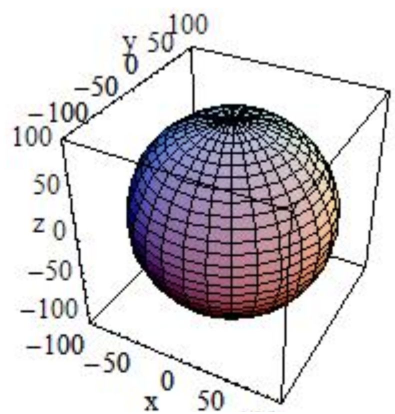
G4Trap



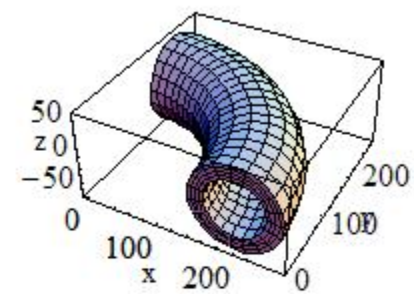
G4Para
(parallelepiped)



G4Sphere



G4Orb
(full solid sphere)



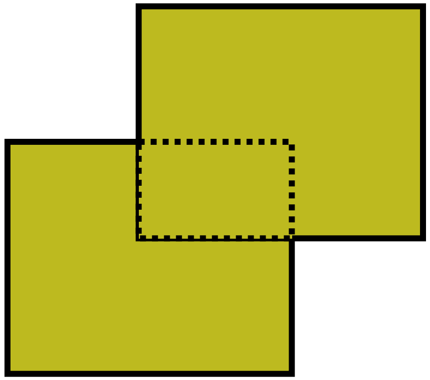
G4Torus

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

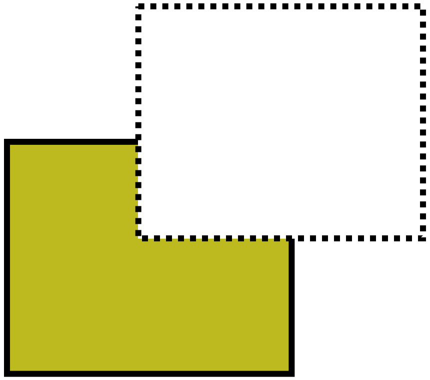
Boolean Solids

- 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 a third solid can be combined with 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

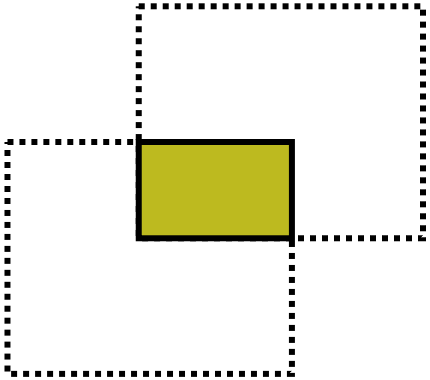
G4UnionSolid



G4SubtractionSolid



G4IntersectionSolid



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.)); ->position of the cylinder with  respect to
                                               the box center

G4RotationMatrix* rm = new G4RotationMatrix();
rm->RotateX(30.*deg);
G4VSolid* intersect =
    new G4IntersectionSolid("Box&&Cylinder",
        box, cylinder, rm, G4ThreeVector(0.,0.,0.)); ; ->position amd rotation of
                                    the cylinder with  respect to  the box center
```

Boolean Solids

Alternative: Unified solids (Geant4 10.0)

G4LogicalVolume

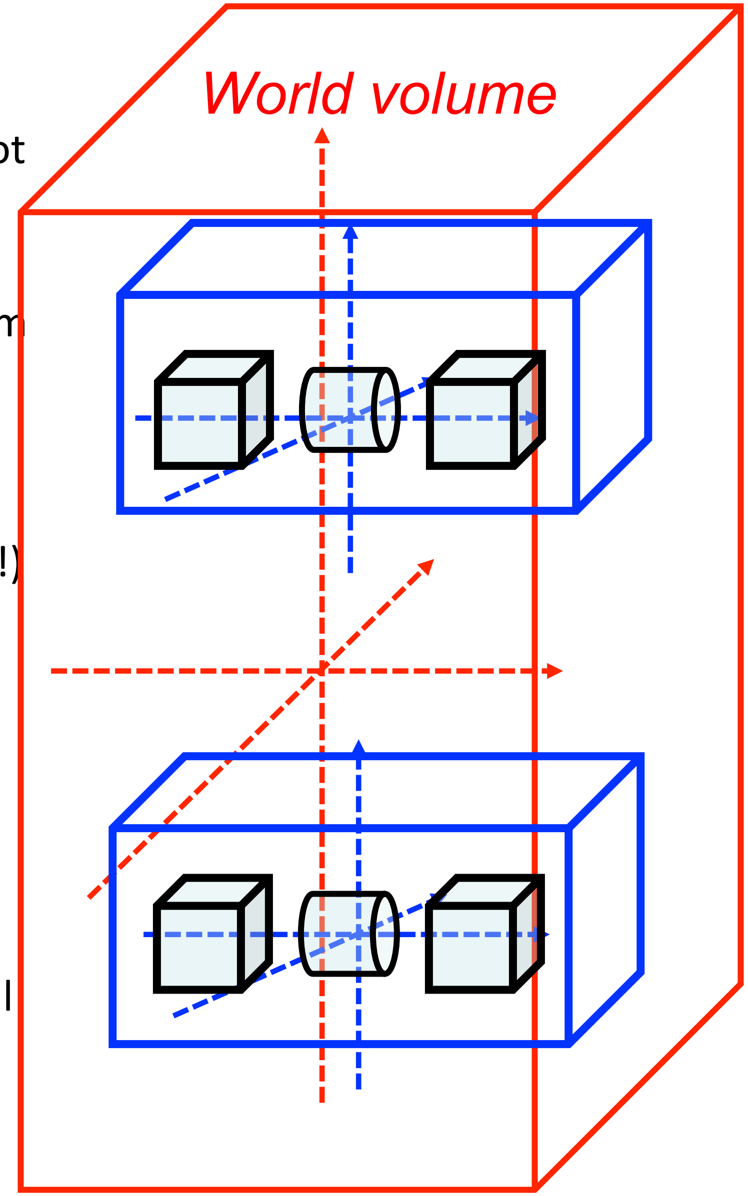
```
G4LogicalVolume(G4VSolid* pSolid,
                G4Material* pMaterial,
                const G4String& name,
                G4FieldManager* pFieldMgr=0,
                G4VSensitiveDetector* pSDetector=0,
                G4UserLimits* pULimits=0,
                G4bool optimise=true);
```

- Contains all information of volume except position:
  - Shape and dimension (G4VSolid)
  - Material, sensitivity, visualization attributes
  - Position of daughter volumes
  - Magnetic field, User limits
- Physical volumes of same type can share a logical volume.
- The pointers to solid and material must not be null!

Geometrical hierarchy

- It forms a **tree.**
- There is just one physical **world volume** forming a root of the tree:
  - 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 this coordinate system
  - It contains all other volumes (with no protruding!)

- One logical volume can be placed more than once.
- One or more volumes can be placed in one mother volume.
- There must be no overlapping or protrusion.
- `G4LogicalVolume` carries the information about daughter volumes => if it is placed more than once, all daughters by definition appear in each physical instance

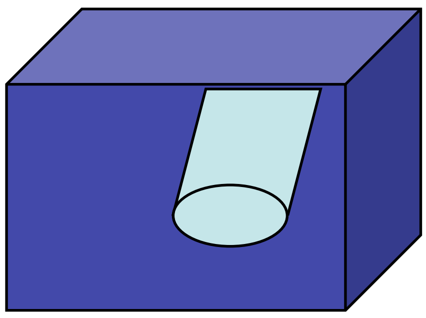


Physical Volumes

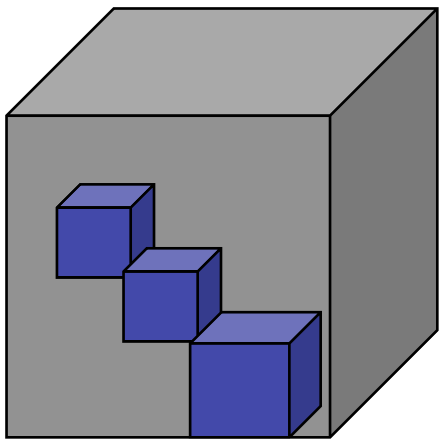
A physical volume is a positioned instance of a logical volume inside a another logical volume (the mother volume).

- G4PVPlacement: a single positioned volume
  Note that a Placement Volume can still represent multiple detector elements, if several copies exist of the mother logical volume

- Repeated Volumes: a single physical volume represents multiple copies of a volume within its mother volume
  - G4PVReplica (= simple repetition)
  - G4PVParameterised (= more complex)



G4PVPlacement



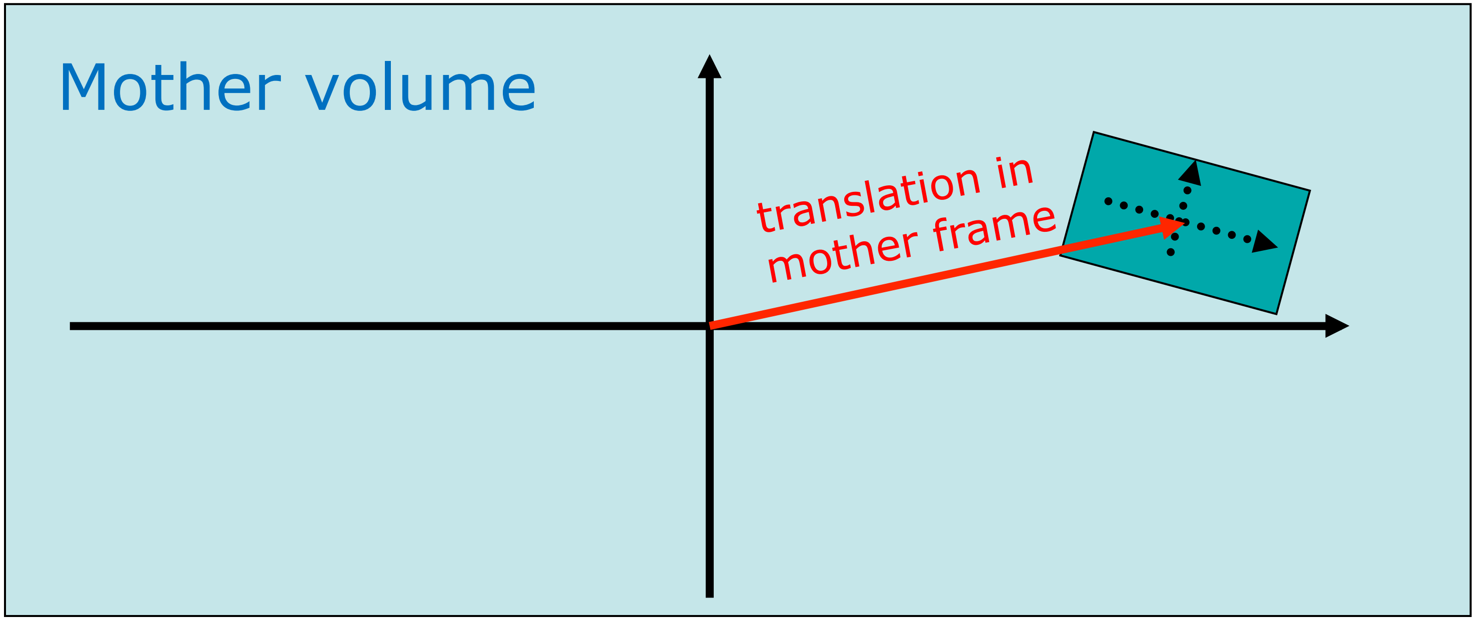
G4PVParameterised

G4PVPlacement

Rotation of mother frame …

```
G4PVPlacement(G4RotationMatrix* pRot,         // rotation of mother frame
              const G4ThreeVector& tlate,     // position in mother frame
              G4LogicalVolume* pCurrentLogical,
              const G4String& pName,
              G4LogicalVolume* pMotherLogical,
              G4bool pMany,                 // not used. Set it to false…
              G4int pCopyNo,                // unique arbitrary index
              G4bool pSurfChk=false );      // optional overlap check
```

- Single volume positioned relatively to the mother volume (passive transformation)

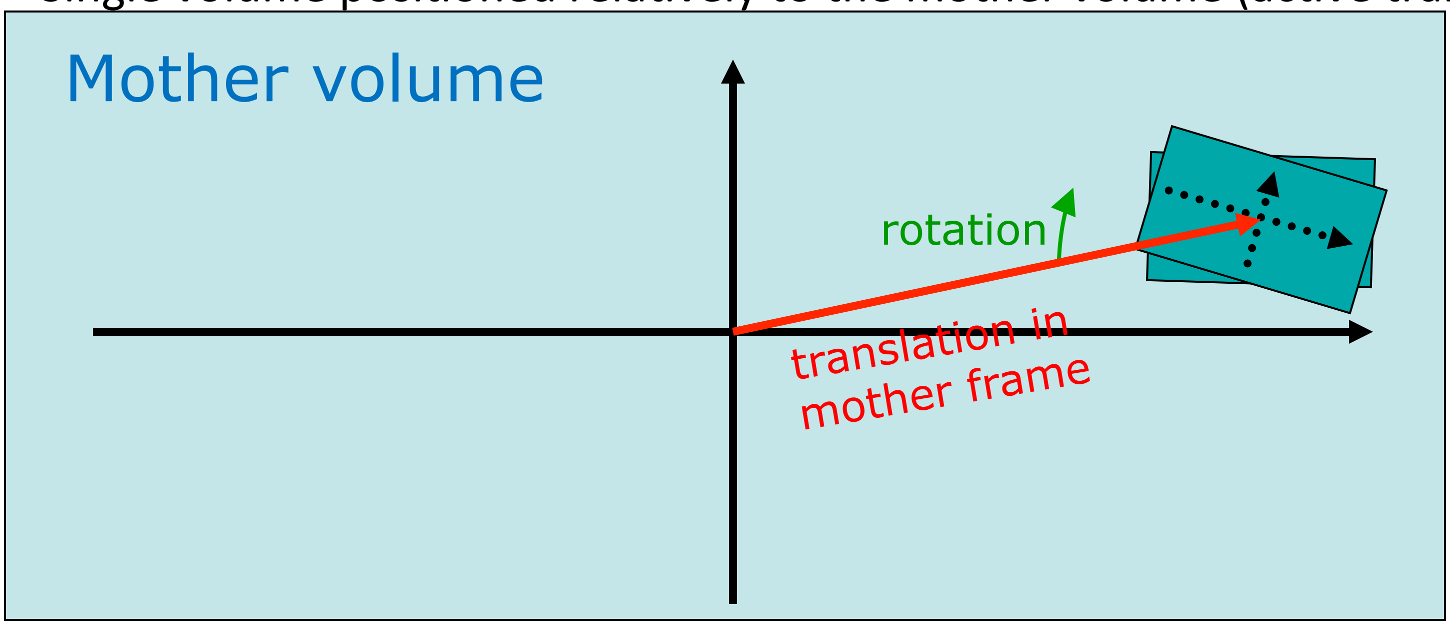


G4PVPlacement

Rotation in mother frame …

```
G4PVPlacement(G4Transform3D(
                  G4RotationMatrix &pRot,         // rotation of daughter frame
                  const G4ThreeVector &tlate),    // position in mother frame
              G4LogicalVolume *pDaughterLogical,
              const G4String &pName,
              G4LogicalVolume *pMotherLogical,
              G4bool pMany,                      // not used, set it to false…
              G4int pCopyNo,                     // unique arbitrary integer
              G4bool pSurfChk=false );           // optional overlap check
```

- Single volume positioned relatively to the mother volume (active transformation)



Regions

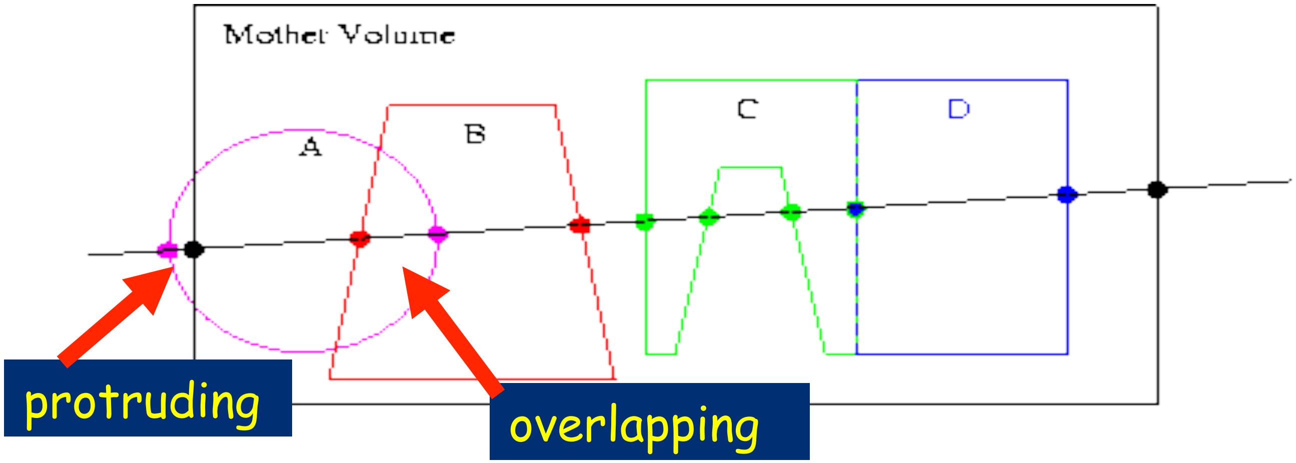
- A **region** is a collection of related logical volumes collected together independent of the geometry tree

- 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.
    - Limits assigned to logical volumes have higher priority
  - Fast simulation manager
  - Field manager

Please note :

- World logical volume is recognized as the default region. User is not allowed to define a region for the world logical volume.

Problems in geometry



- Geant4 does not allow malformed geometries, neither protruding nor overlapping.
  - The behavior of navigation is unpredictable for such cases!
- Overlaps become an important issue in complex geometries.
- There are tools for detecting wrong positioning:
  - Optional checks at construction
  - Kernel run-time commands
  - Graphical tools (DAVID, OLAP)

Tools for geometry checking

- 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.

This check requires lots of CPU time, but it is worth to try at least once.

- Built-in run-time commands to activate verification tests for the user geometry:

`/geometry/test/run` or `/geometry/test/grid_test`

to start verification of geometry for overlapping regions based on a standard grid setup, limited to the first depth level

`/geometry/test/recursive_test`

applies the grid test to all depth levels (may require lot of CPU time!)

`/geometry/test/line_test`

to shoot a line along a specified direction and position

Tools for geometry checking

Source

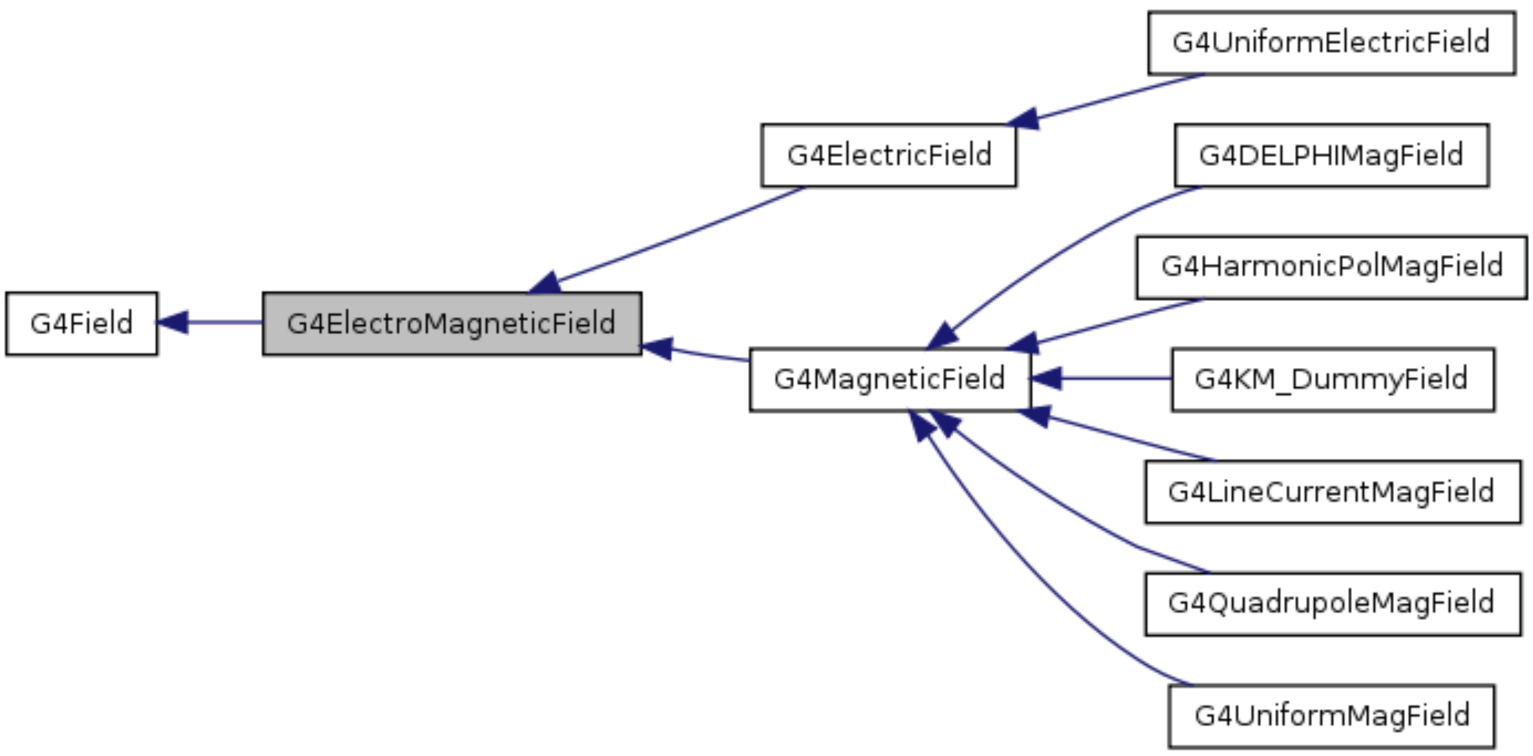
```
void MGManagerDetectorConstruction::CheckOverlaps()
{
  G4PhysicalVolumeStore* thePVStore = G4PhysicalVolumeStore::GetInstance();
  G4cout << thePVStore->size() << " physical volumes are defined" << G4endl;
  G4bool overlapFlag = false;
  G4int res = 1000;
  G4double tol = 0.;       // tolerance
  for (size_t i=0; i<thePVStore->size(); i++)
  {
      overlapFlag = (*thePVStore)[i]->CheckOverlaps(res,tol,fCheckOverlapsVerbosity)
           | overlapFlag;
  }
  if (overlapFlag)
      G4cout << "Check: there are overlapping volumes" << G4endl; }
```

Output

```
136 physical volumes are defined
Checking overlaps for volume BeamLineSupport ... OK!
Checking overlaps for volume BeamLineCover ... OK!
Checking overlaps for volume BeamLineCover2 ... OK!
Checking overlaps for volume VacuumZone ... OK!
Checking overlaps for volume FirstScatteringFoil ... OK!
……
-------- WWWW ------- G4Exception-START -------- WWWW -
*** G4Exception : GeomVol1002    issued by : G4PVPlacement::CheckOverlaps()
Overlap with volume already placed !
          Overlap is detected for volume BrassTube2
          with HoleNozzleSupport volume's
          local point (12.6381,12.8171,-25.1867), overlapping by at least: 3.5 mm
*** This is just a warning message. ***
-------- WWWW -------- G4Exception-END --------- WWWW -
…
```

(Electromagnetic) fields

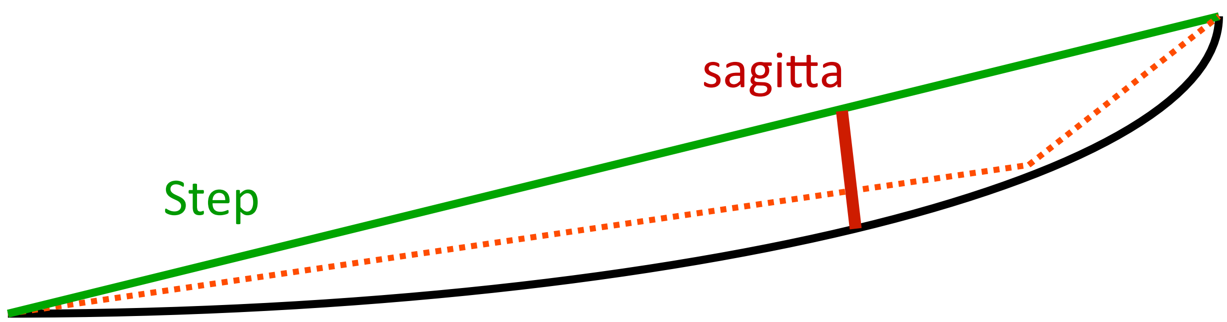
- Geant4 enables tracking in user-defined fields
- Defined in **ConstructSDAndField()** of the detector construction
- Electromagnetic fields (and variants) by default, others may be included



Navigation in fields

- In order to propagate a particle inside a field (e.g. magnetic, electric or both), we integrate the equation of motion of the particle in the field
  - Most common approach: **Runge-Kutta** (RK) method
  - RK methods of different orders are available (most common: 4th)
  - In specific cases, other solvers can also be used

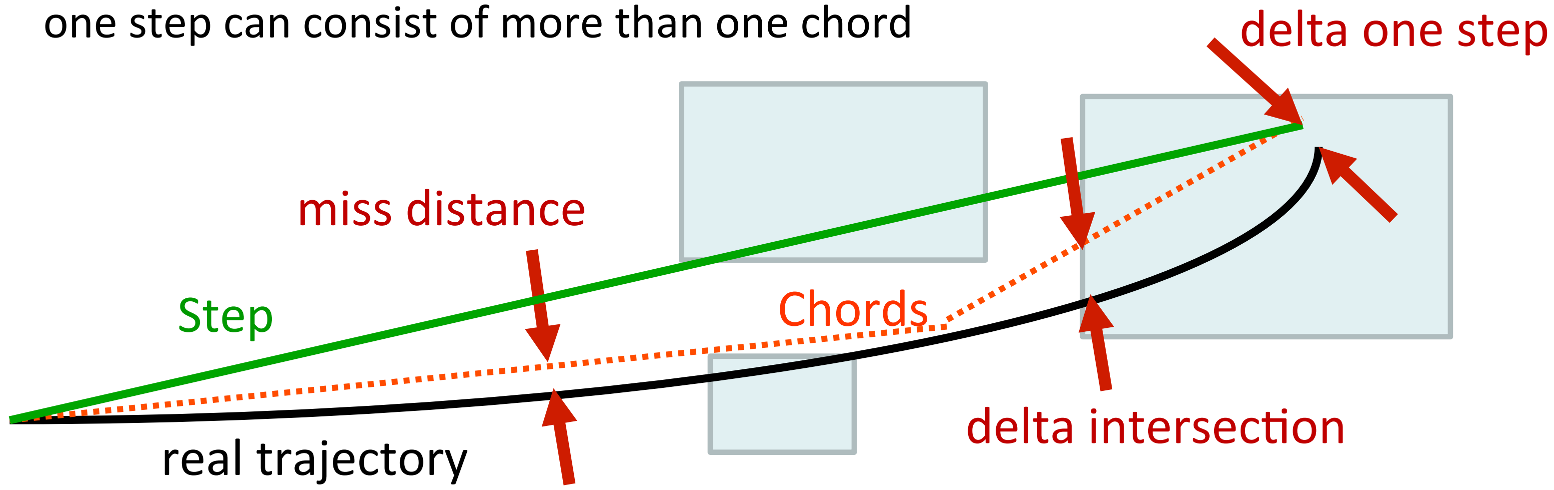
- Geant4 breaks up this curved trajectory into linear chord segments:



- Chord segments are determined so that they closely approximate the curved path; they're chosen so that their sagitta is small enough
  - The *sagitta* is the maximum distance between the curved path and the straight line of the step

Navigation in fields (2)

- Chords are used to interrogate the Navigator
  - to see whether the track has crossed a volume boundary
  - one step can consist of more than one chord



- Correctness of the navigation can be tuned by setting parameters:
  - **miss distance:** upper bound for sagitta
  - **delta intersection:** max error at intersection
  - **delta one step:** max error in position in general

- Setting the values too low has bad impact on CPU!

Example: Uniform magnetic field

In the MyDetector::ConstructSDAndField() method:

- Create the field itself:

```
G4MagneticField* magField =
new G4UniformMagField(G4ThreeVector(1.*Tesla, 0., 0.));
```

- Register the field to the global field manager:

```
G4FieldManager* globalFieldMgr =
    G4TransportationManager::GetTransportationManager()
        ->GetFieldManager();
globalFieldMgr->SetDetectorField(magField);
```

- Crete a chord finder for it:

```
globalFieldMgr->CreateChordFinder(magField);
```

Example: Non-uniform magnetic field

- Create a class, derived from `G4MagneticField`
  - override the GetFieldValue method:

```
void MyField::GetFieldValue(
    const G4double point[4], G4double *field) const
{
    // Artificial example
    G4double k = 1 * tesla / m;

    field[0] = k * point[1];
    field[1] = k * point[0];
    field[2] = 0;
}
```

- Then follow as in the uniform field case

- **Note:** point as well as field values are in global coordinates!

Assign a field to a logical volume

- It is possible to describe a field for a specific logical volume (and its daughters):
  - Create a local **G4FieldManager** and attach it to the logical volume:

```
G4FieldManager* localFieldMgr = new G4FieldManager(magField);
G4bool allLocal = true;
logicVolWithField->SetFieldManager(localFieldMgr, allLocal);
```

If true, field assigned to all daughters
If false, field assigned only to daughters w/o their own field manager

Thank you

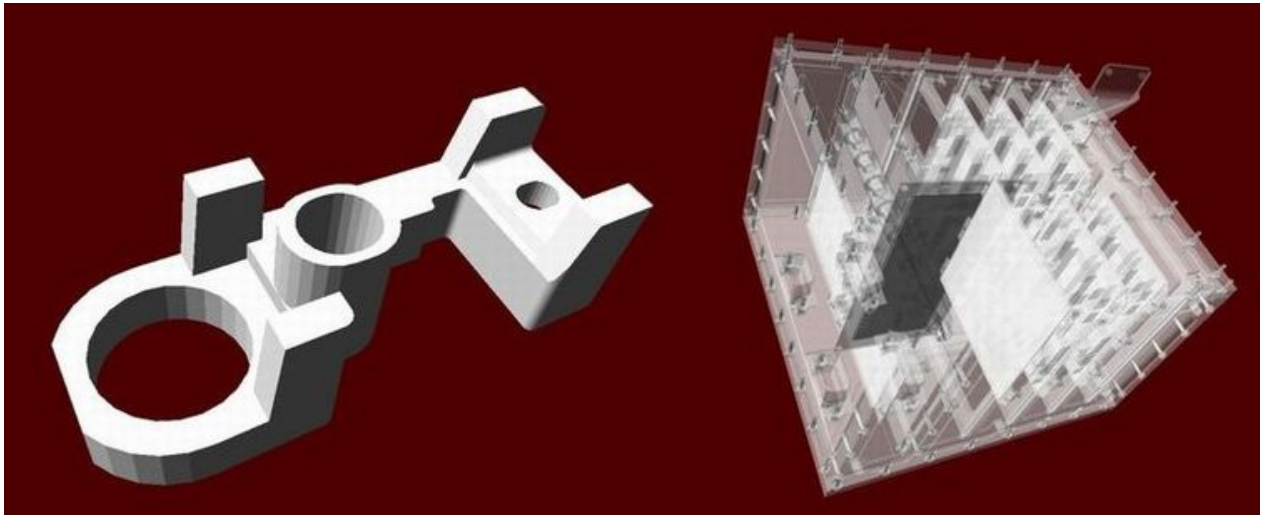
~~Optional~~ ~~Advanced~~ ~~Additional…~~
…Topics

GDML / CAD

- GDML is a XML-based format that enables to describe many aspects of geometry
  - you can load the geometry from external file
  - you can also save the geometry to an external file
  - includes materials, solids, logical & physical volumes

► examples/extended/persistency/gdml

- To import **CAD**, you can construct the geometry using the G4TessellatedSolid class, see documentation.

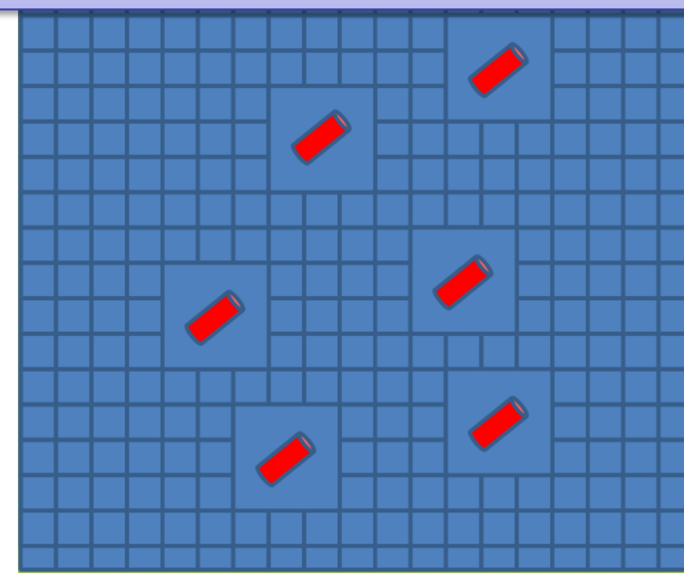


Parallel worlds

- Implementation is complex (not to be covered here)
- Advantages for construction & navigation
- Two examples:

Brick floating on water surface

Brachytherapy:
= metal seeds in voxel geometry



Taken from M. Asai with(out) permission