GEANT4 BEGINNERS COURSE

GSSI, L'Aquila (Italy)
27-30 June 2016

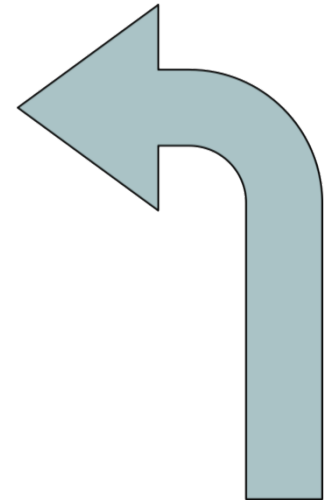# Detector description: materials and geometry

Geant 4 tutorial course

Introduction

Mandatory user classes in a Geant4:

- **G4VUserPrimaryGeneratorAction**

- **G4VUserDetectorConstruction**

- **G4VUserPhysicsList**

Materials, Volumes, SD and Fields to be used in the simulation must be defined in the G4VUserDetectorConstruction class

Part I: Materials

Materials:

- System of units & constants
- Definition of elements
- Materials and mixtures
- Some examples …
- NIST Data Base

Units

- Geant4 has no default unit.
- To introduce input data, unit **must** be "multiplied" to the number.
  - for example :

    ```
    G4double width = 12.5*m;

    G4double density = 2.7*g/cm3;
    ```
  - Almost all commonly used units are available.
  - The user can define new units.
  - Refer to CLHEP: `SystemOfUnits.h`
- To output the data you can **divide** a variable by a unit you want to get.

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

System of Units

- System of units are defined in CLHEP, based on:
  - millimetre (`mm`), nanosecond (`ns`), Mega eV (`MeV`), positron charge (`eplus`) degree Kelvin (`kelvin`), the amount of substance (`mole`), luminous intensity (`candela`), radian (`radian`), steradian (`steradian`)
- All other units are computed from the basic ones
- Alternative way to output data: Geant4 can choose the most appropriate unit to use. Just 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

Defining new Units

- New units can be defined directly as constants,
  - `static const G4double inch = 2.54*cm;`
- or via `G4UnitDefinition` (suggested way)
  - `new 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`

- To print the list of units:
  - From the code
    `G4UnitDefinition::PrintUnitsTable();`
  - At run-time, as UI command:
    `Idle> /units/list`

Definition of materials

- Different kinds of materials can be defined:
  - isotopes `<>` `G4Isotope`
  - elements `<>` `G4Element`
  - molecules `<>` `G4Material`
  - compounds and mixtures `<>` `G4Material`

- Attributes associated:
  - temperature, pressure, state, <u>density</u>

- `G4Isotope` and `G4Element` describe *atomic* properties:
  - 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, refractive index, etc… → linking a `G4MaterialPropertyTable`

- `G4Material` is the only class used and visible to the toolkit: it is used by tracking, geometry and physics

Elements and isotopes

- Isotopes can be assembled into elements

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

- … building elements as follows:

```
G4Element (const G4String& name,
            const G4String& symbol, // element symbol
                G4int     nIso ); // n. of isotopes
G4Element::AddIsotope(G4Isotope* iso,   // isotope
                G4double relAbund);   // fraction of
                                      // atoms
                                      // per volume
```

Materials of one element and molecules

- Single element material:

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

- A molecule material (composition by chemical formula):

```
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);
```

Compound and mixture

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

Example: gas

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

```
G4double density = 0.3*mg/cm3;
G4double temperature = 500.*kelvin;
G4double pressure = 2.*atmosphere;
G4Material* steam = new G4Material("Water steam", density,
 ncomponents=1, kStateGas, temperature, pressure);
steam->AddMaterial(H2O, fractionmass = 1.);
```

- <u>Absolute vacuum does not exist</u>: gas at very low density !
  - Materials must have 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);
```

NIST Material Data-Base in Geant4

- NIST database for materials is imported inside Geant4
  http://physics.nist.gov/PhysRefData
- Additional interfaces defined
- 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

Z		A	m	error	(%)	A_{eff}
=============================							
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
 4  G4_Air                     0.00120479      85.7
      6    0.000124
      7    0.755268
      8    0.231781
     18    0.012827
 2  G4_CsI                     4.51            553.1
     53    0.47692
     55    0.52308
```

- NIST Elementary materials:
  - **H -> Cf (Z = 1 -> 98)**
- NIST compounds:
  - e.g. "**G4_ADIPOSE_TISSUE_ICRP**"
- HEP and Nuclear materials:
  - e.g. **Liquid Ar, PbWO**
- It is possible to build mixtures of NIST and user-defined materials

13

How to use the NIST DB

- No need to predefine elements and materials

- Retrieve materials from NIST manager:

```
G4NistManager* manager = G4NistManager::Instance();

G4Material* H2O = manager->FindOrBuildMaterial("G4_WATER");
```

- Some UI commands …

```
/material/nist/printElement     ← print defined elements
/material/nist/listMaterials    ← print defined materials
```

Part II: Geometry

Geometry:

- Detector description: the basic
  - Define Detector geometry components
  - Combine components

- Describing a detector
  - Solids
  - Logical volumes
  - Physical volumes

- Tools for geometry check

Describe your detector

- A detector geometry is made of a number of volumes

- The largest volume is called **World** volume
  - It must contain all other volumes

- Derive your own concrete class from `G4VUserDetectorConstruction` abstract base class.

- Implementing the method `Construct()`:
  - Modularize it according to each detector component:
    - Define shapes/solids
    - Construct all materials
    - Construct and place volumes
    - ➢ Define sensitive detectors and identify detector volumes which to associate them
    - ➢ Associate magnetic field to detector regions
    - ➢ Define visualization attributes for the detector elements

Detector geometry components

- Three conceptual layers

  - G4VSolid -- *shape, size*

  - G4LogicalVolume -- *daughter physical volumes,*

    material, sensitivity, magnetic field, user limits, etc.

  - G4VPhysicalVolume -- *position, rotation*
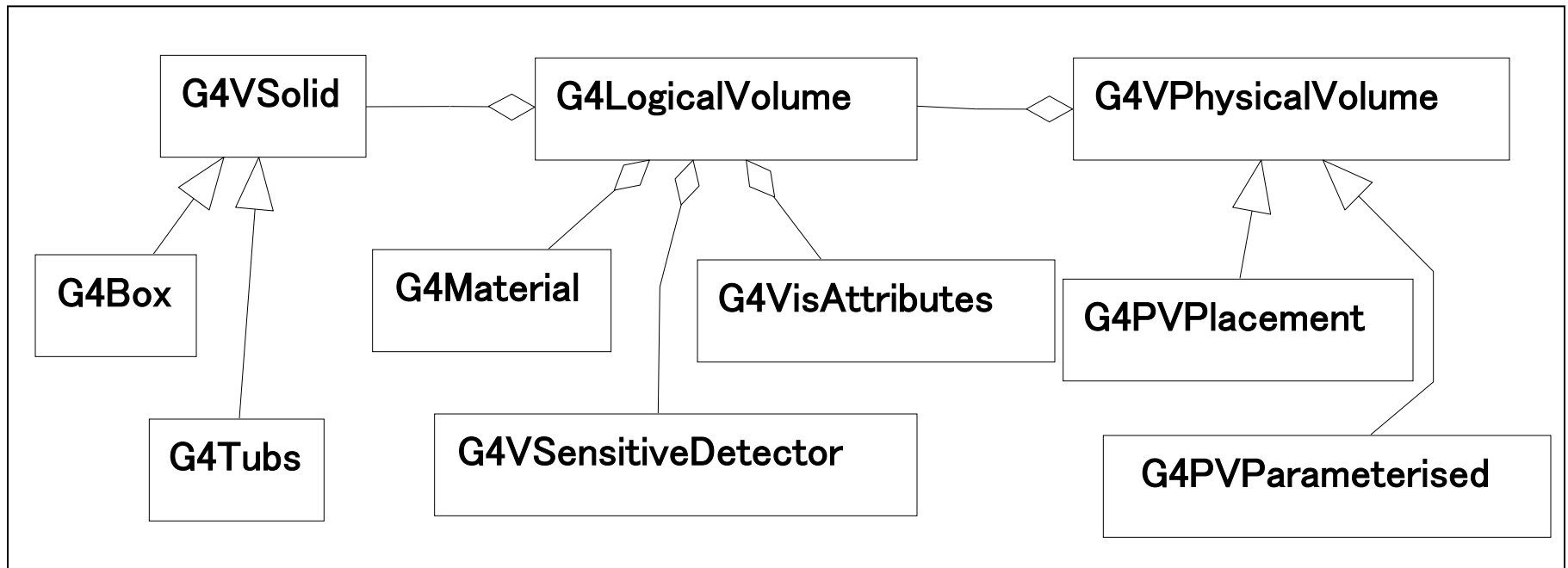
Define detector geometry

- Basic strategy

```
G4VSolid* pBoxSolid =

    new G4Box("aBoxSolid",

            1.*m, 2.*m, 3.*m);
```

Solid : shape and size
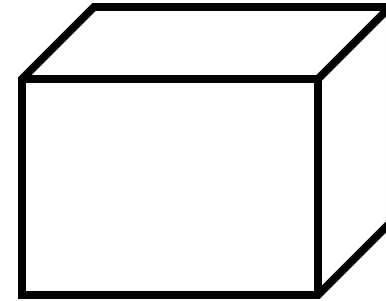
Step 1
Create the geom. object : box

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.
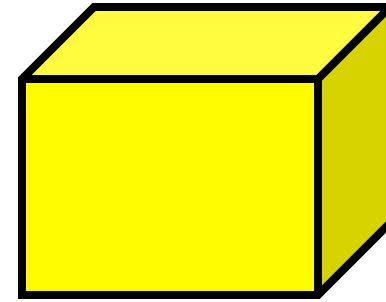
Step 1
Create the geom. object : box
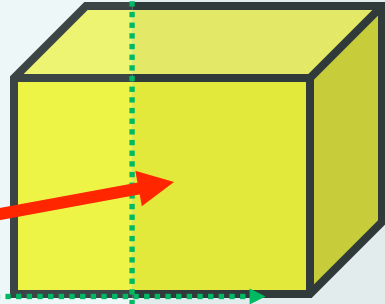
Step 2
Assign properties to object : material

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 contain all other components

  ➢ The world volume

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, …
    - G4TwistedTubs, G4TwistedTrap, …
  - Boolean solids
    - G4UnionSolid, G4SubtractionSolid, G4IntersectionSolid
 - Tessellated Solids (made of facets)
    - G4TessellatedSolid
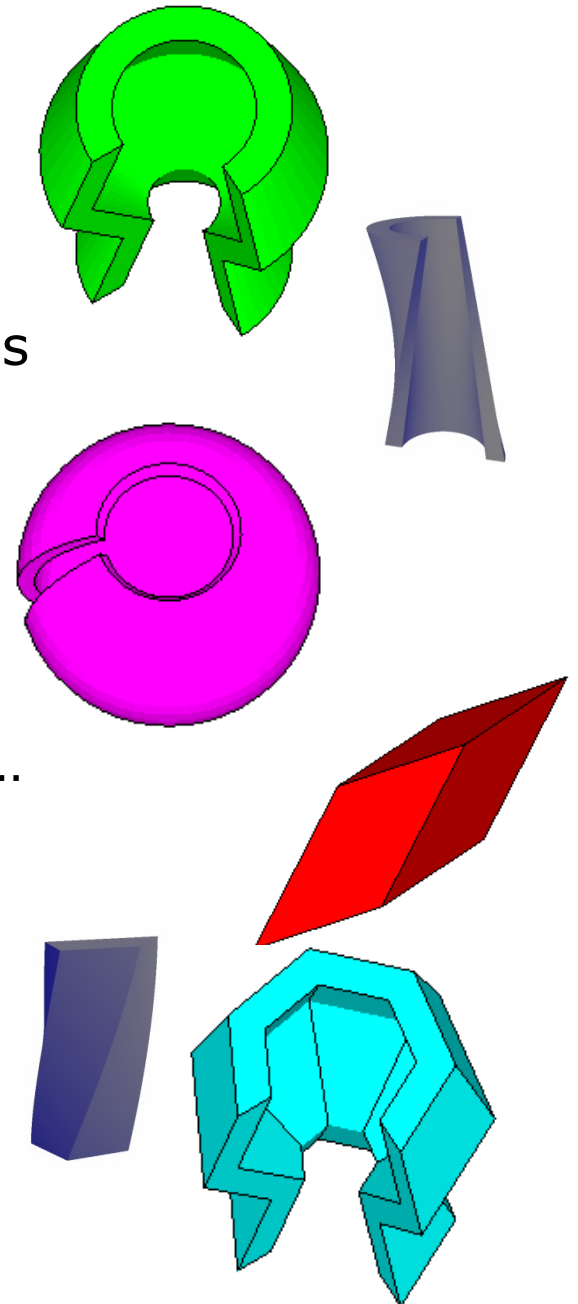
CSG: G4Tubs, G4Cons

```
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
```
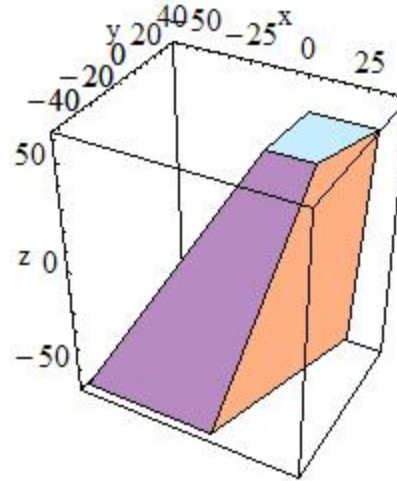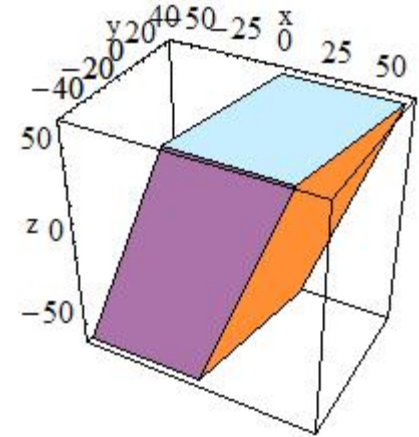
```
G4Cons(const G4String& pname,  // name
              G4double  pRmin1, // inner radius -pDz
              G4double  pRmax1, // outer radius -pDz
              G4double  pRmin2, // inner radius +pDz
              G4double  pRmax2, // outer radius +pDz
              G4double  pDz,    // Z half length
              G4double  pSphi,  // starting Phi
              G4double  pDphi); // segment angle
```
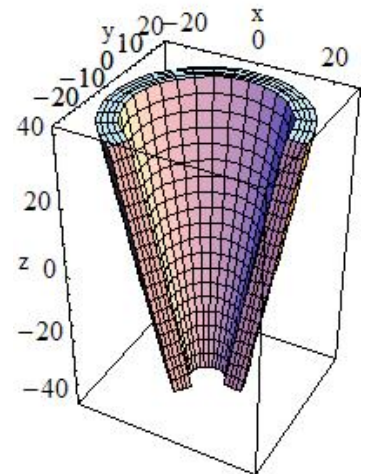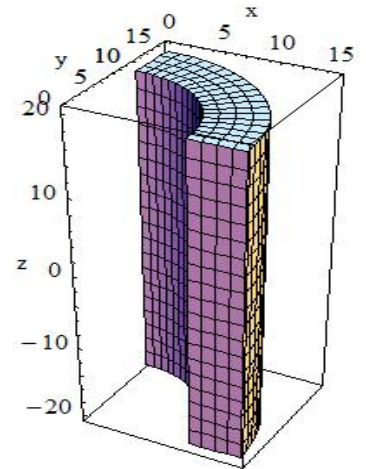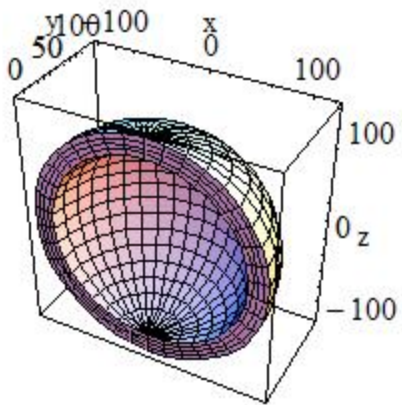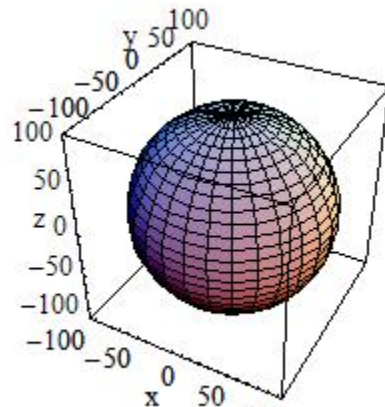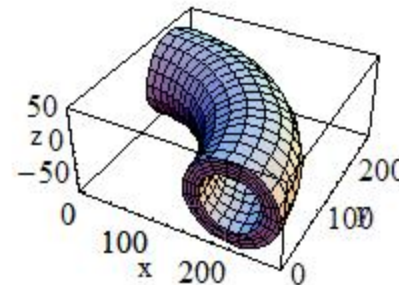
Other CSG solids
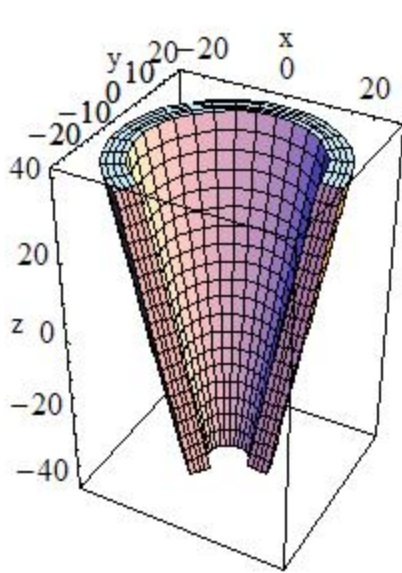
G4Cons
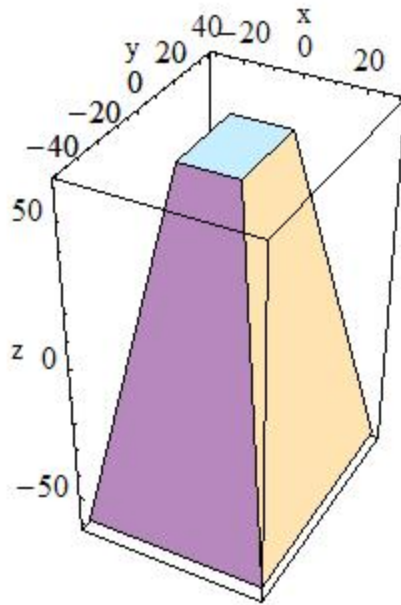
G4Trd

G4Trap

G4Para
(parallelepiped)

G4Sphere

G4Orb
(full solid sphere)

G4Torus

Consult 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 is a solid which coordinate system is the same as the 1st solid

▸ Solids to be combined can be either CSG or other Boolean solids.

▸ <u>Note:</u> tracking cost for the navigation in a complex Boolean solid is proportional to the number of constituent CSG solids
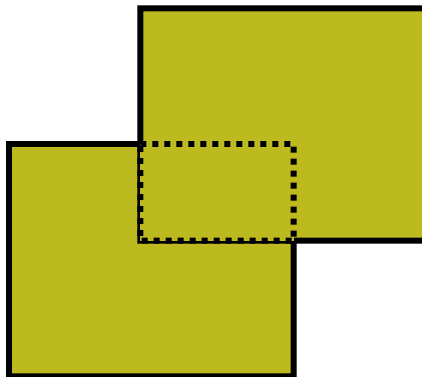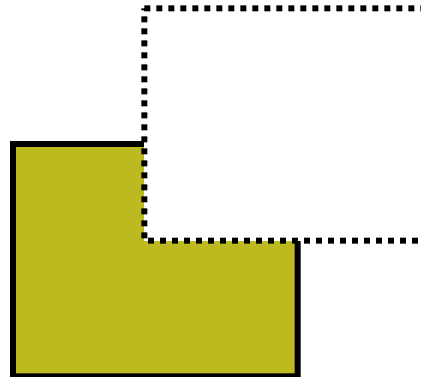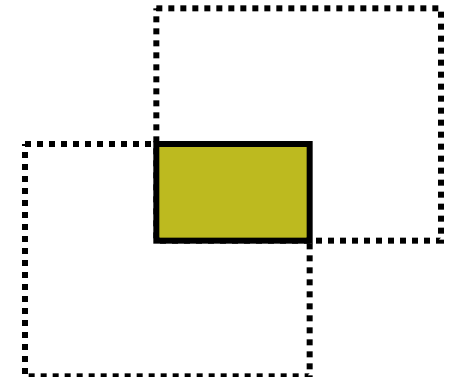
G4UnionSolid   G4SubtractionSolid   G4IntersectionSolid

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.));


G4RotationMatrix* rm = new G4RotationMatrix();
rm->RotateX(30.*deg);
G4VSolid* intersect = new G4IntersectionSolid("Box&&Cylinder",
        box, cylinder, rm, G4ThreeVector(0.,0.,0.));
```

Boolean Solids
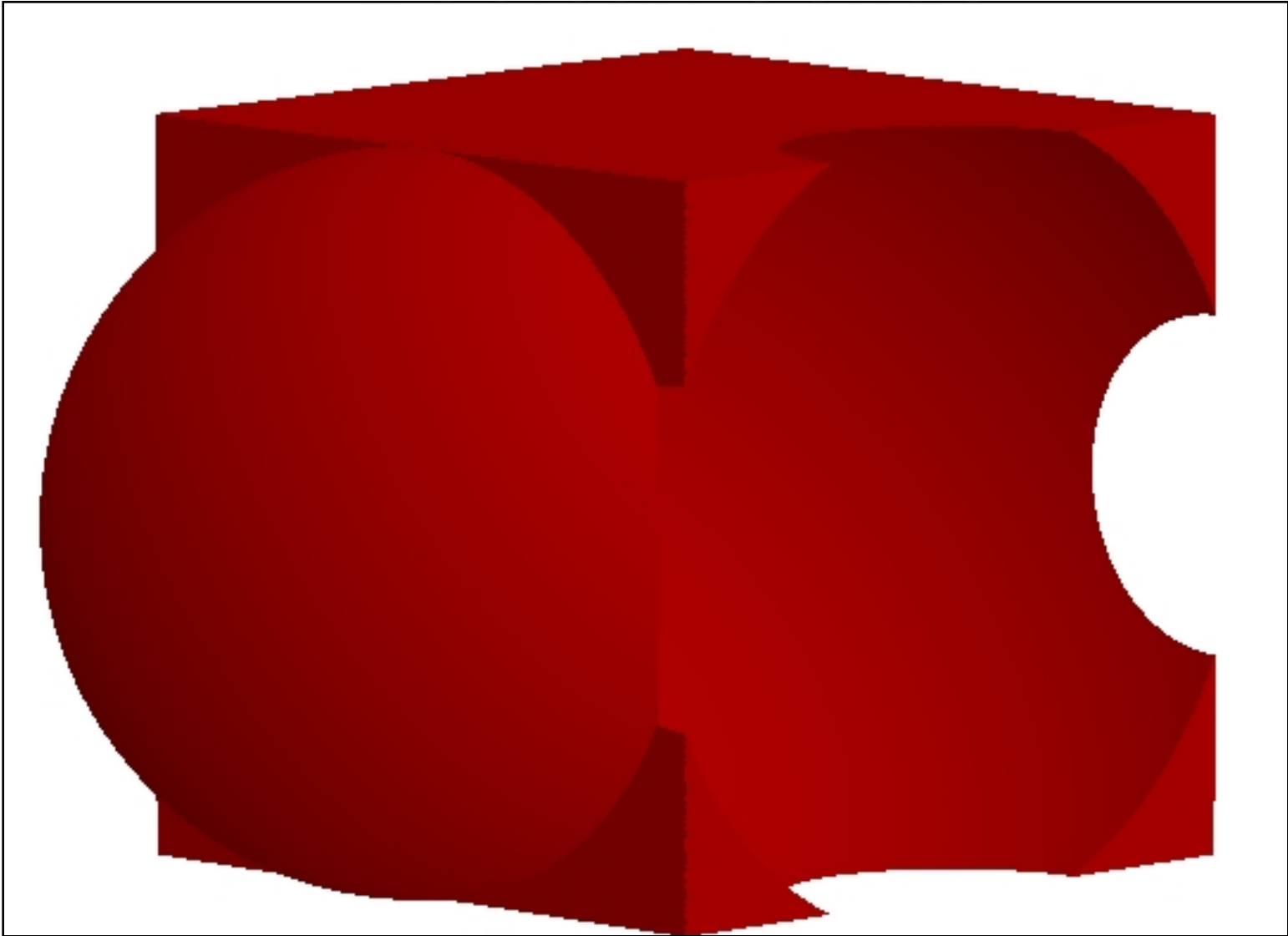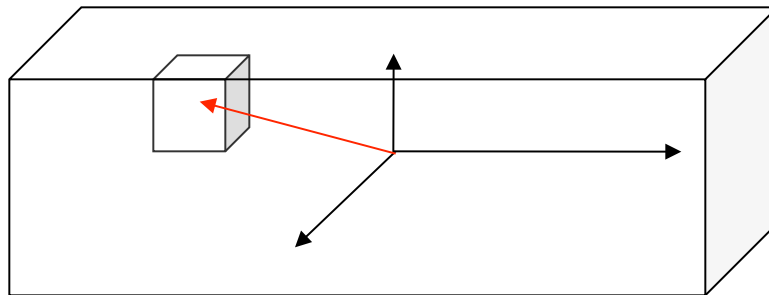
G4LogicalVolume

```
G4LogicalVolume(G4VSolid* pSolid,
                G4Material* pMaterial,
                const G4String& name,
                G4FieldManager* pFieldMgr=0,
                G4VSensitiveDetector* pSDetector=0,
                G4UserLimits* pULimits=0,
                G4bool tracking_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 be NOT null

Geometrical hierarchy

- Mother and daughter volumes
  - 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 the mother's local coordinate system is at the center of the mother volume
    - Daughter volumes <u>cannot protrude</u> from the mother volume
    - Daughter volumes <u>cannot overlap</u> with each other
  - The logical volume of mother knows the daughter volumes it contains
    - It is uniquely defined to be their mother volume

Geometrical hierarchy

- One or more volumes can be placed in a mother volume. One logical volume can be placed more than once.

- Note: mother-daughter relationship is an information of G4LogicalVolume
  - If the mother volume is placed more than once, all daughters appear in each placed physical volume

- World volume must be a unique physical volume which fully contains all 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
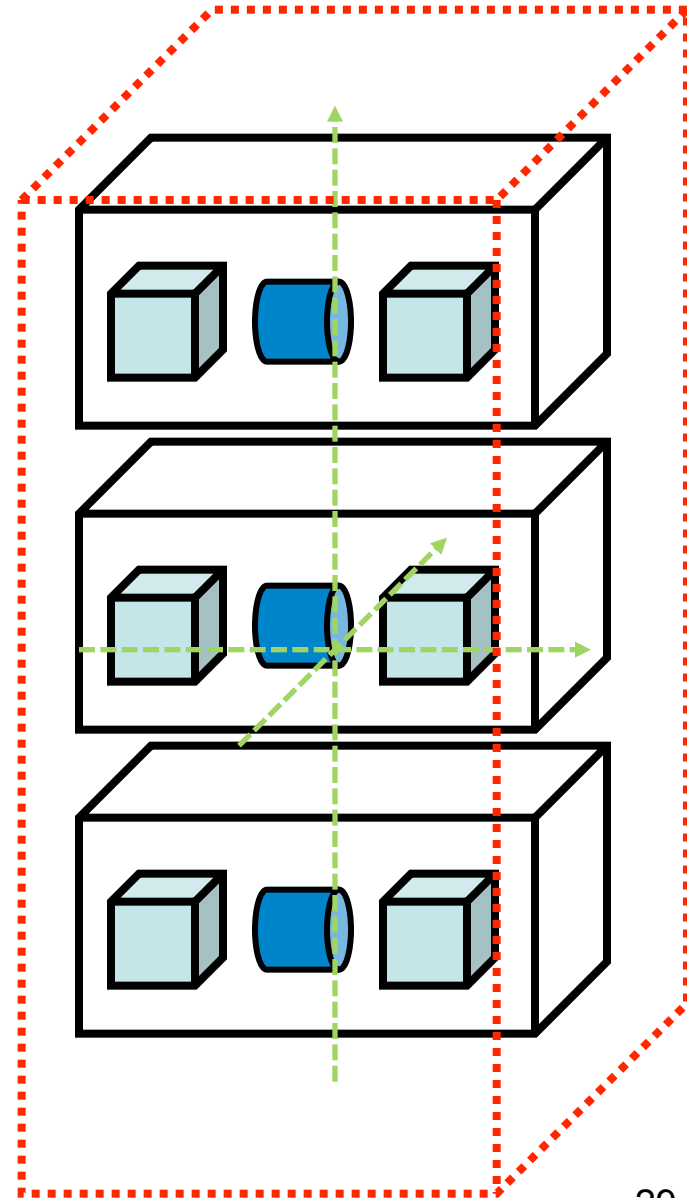
Region

- A **region** is an Envelope of Logical Volumes which may have common characteristics

- A region may have its unique
  - Production thresholds (cuts)
    - If a region 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 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
  - Fast simulation manager
  - Field manager

- Notice :
  - World logical volume is recognized as the default region. User is not allowed to define a region and assign it to the world logical volume

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

- In general this is best done using a **Runge-Kutta** (RK) method for the integration of ordinary differential equations
  - Several RK methods are available

- When analytical solution is available, specific solvers can also be used

- When approximate analytical solution is available, it is iteratively applied to converge to the solution at the required precision

- **Once the curved path is calculated**, Geant4 breaks it up into linear chord segments, used for the navigation
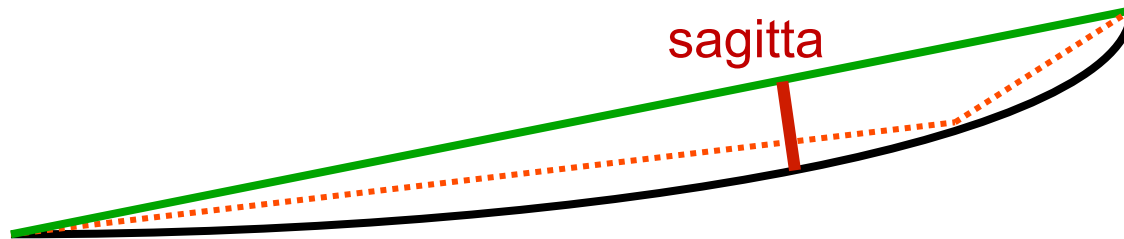
sagitta

- The chord segments are determined to closely approximate the curved path; approximation level is set through an upper bound to the sagitta (maximum distance between the curved path and the straight line)

Fields

- Chords are used to interrogate the Navigator
  - to check whether the track has crossed a volume boundary

miss distance

Step

Chords

real trajectory
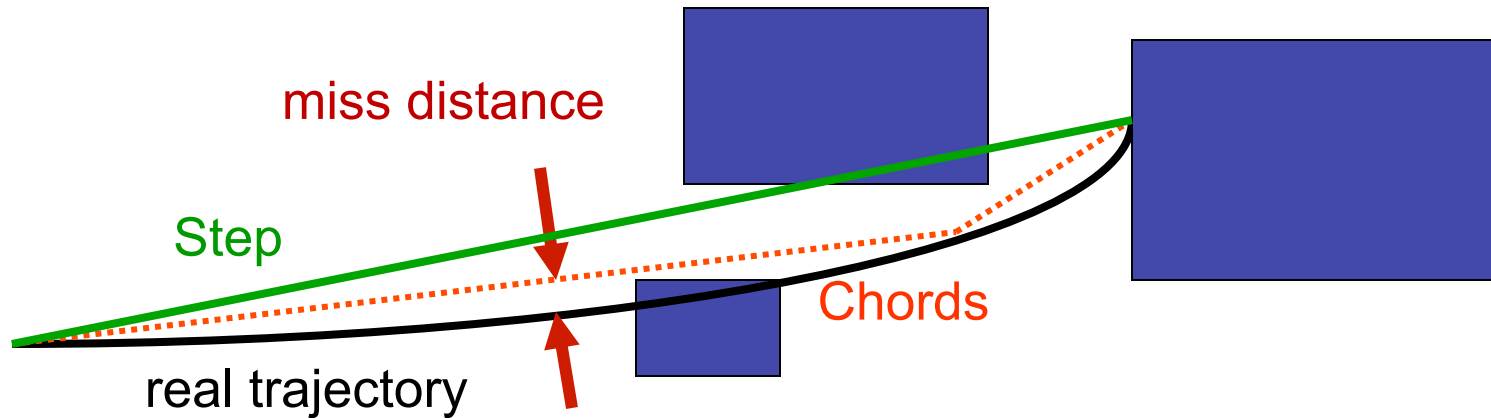
- The accuracy of the volume intersection can be tuned
  - by setting a parameter called the "*miss distance*":
    - Upper bound for the value of the sagitta
    - Setting small *miss distance* may be highly CPU consuming

- One step can consist of more than one chord
  - In some cases, one step consists of several turns

Example: Create a Magnetic Field

✦ Uniform field :
- Create the field
  ```
  G4MagneticField* magField =
  new G4UniformMagField(G4ThreeVector(1.*Tesla,0.,0.));
  ```

- Set it as default field
  ```
  G4FieldManager* globalFieldMgr =
  G4TransportationManager::GetTransportationManager()
          ->GetFieldManager();
  globalFieldMgr->SetDetectorField(magField);
  ```

- Create the chord finder
  ```
  globalFieldMgr->CreateChordFinder(magField);
  ```

✦ Non-uniform field :
  ✦ Create a class, derived from `G4MagneticField`, with a method
  ```
  void MyField::GetFieldValue(const double Point[4],
          double *field) const
  ```

Associate a Field to a logical volume

✧ It is possible to describe a field inside a logical volume and all its daughters.

✧ This can be done creating a local **G4FieldManager** and attaching it to a logical volume:

```
G4FieldManager* localFieldMgr =
    new G4FieldManager(magField);

G4bool alldaug = true;

logicVolWithField->SetFieldManager(localFieldMgr,alldaug);
```

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

Physical Volumes

A physical volume represents the spatial positioning of the volumes describing your detector

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

placement

- ## Repeated Volumes: a single Physical Volume represents multiple copies of a volume within its mother volume

  - – reduces use of memory
  - – Replicas
    - • simple repetition, similar to G3 divisions
  - – Parameterised Volumes
  - – Divided Volumes
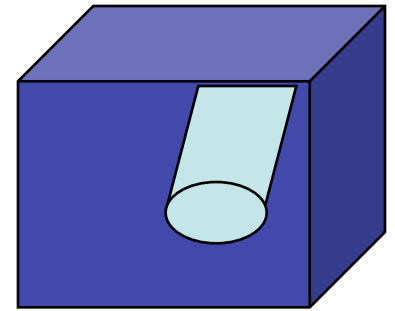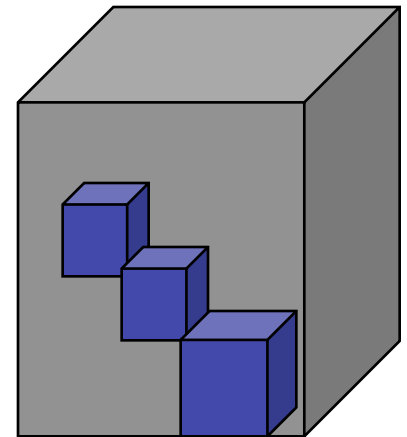
repeated

G4PVPlacement

```
G4PVPlacement(G4RotationMatrix* pRot,      // rotation of mother frame
              const G4ThreeVector& tlate, // position in rotated 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
  - 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 (*alternative constructor*)
  - The combination of the two variants above

G4PVPlacement
Rotation of mother frame …

```
G4PVPlacement(G4RotationMatrix* pRot,        // rotation of mother 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 index
              G4bool pSurfChk=false ); // optional overlap check
```

- Single volume positioned relatively to the mother volume

Mother volume

translation in mother frame
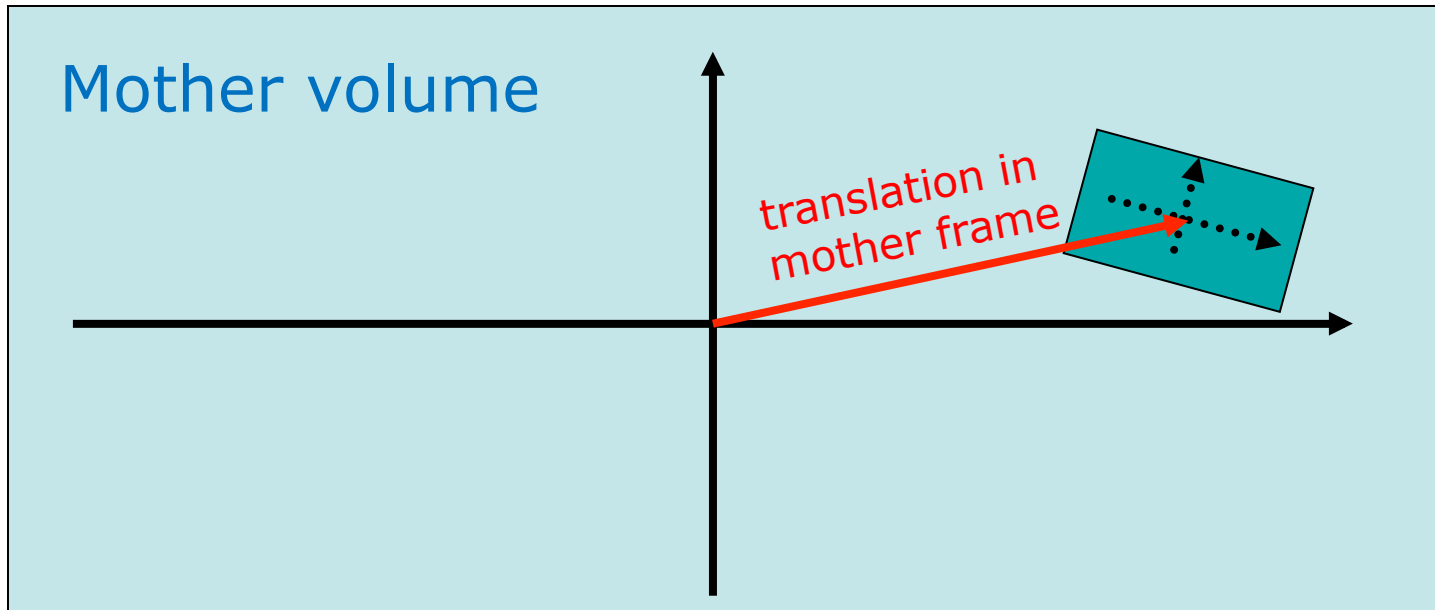
G4PVPlacement
Rotation in mother frame …

```
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
```
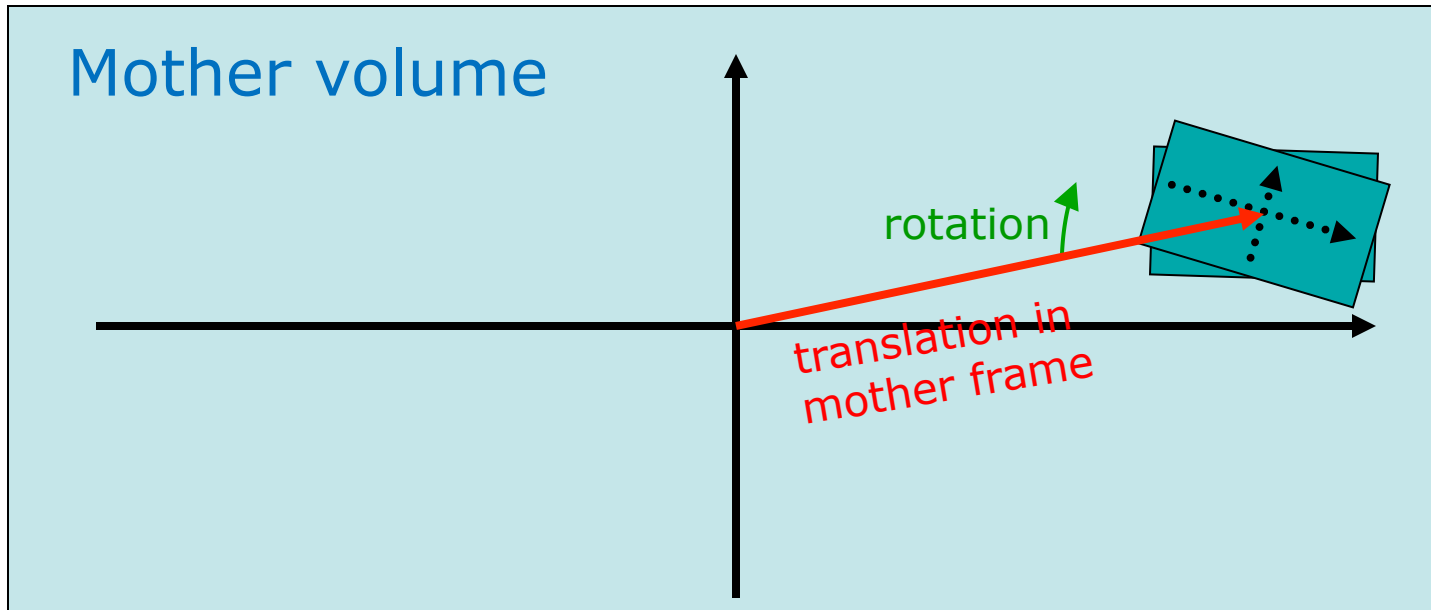

Mother volume

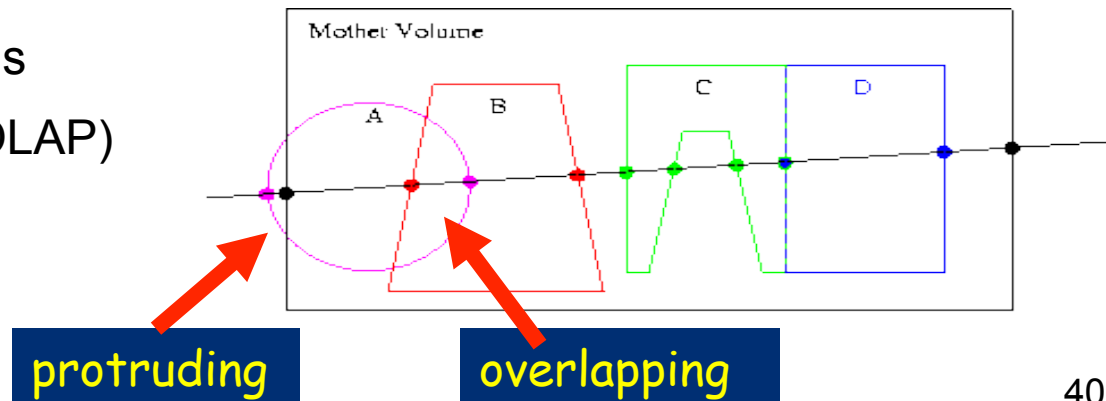rotation

translation in mother frame

Volume Stores

- All volumes must be allocated using `new` in the user's application
- At construction, they get registered to a proper Volume Store
- At the end of the job, they will also be automatically deallocated

- Stores:
  - G4SolidStore
  - G4LogicalVolumeStore
  - G4PhysicalVolumeStore
  - G4RegionStore

- Volumes can be retrieved at any convenience, for instance:
  - by index
    ```
    int NofV=G4LogicalVolumeStore::GetInstance()->size();
    for(int i=0;i<NofV;i++){
            lVol=(*(G4LogicalVolumeStore::GetInstance())))[i];
    }
    ```
  - by name
    ```
    G4LogicalVolume* logicVol =
    G4LogicalVolumeStore::GetInstance()->GetVolume("logicLAr");
    ```

Tools for geometry check

- A protruding volume is a contained daughter volume which actually protrudes from its mother volume.

- 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 in 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)

protruding   overlapping

Tools for geometry check

- Constructors of **G4PVPlacement**, **G4PVParameterised** and **G4PVDivision** have an optional argument "pSurfChk".

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

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 and configure verification tests for the user geometry:

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

  - **/geometry/test/recursion_depth**
    Set the depth in the geometry tree for recursion

  - **/geometry/test/tolerance**
    Define tolerance (in mm) by which overlaps reports

Thanks for your attention

Tools for geometry check

```
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 example:

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 -
…
…

This method can be called at any point after run->Initialize();