



Version 10.7

Geometry

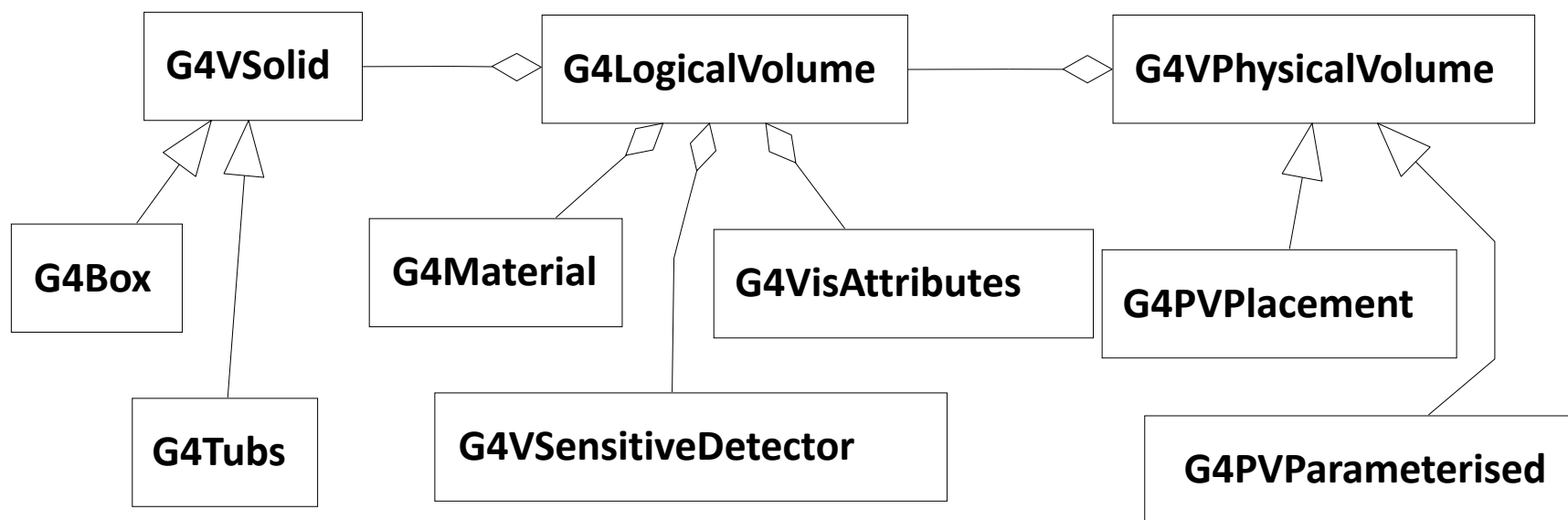
Overview & basic concepts

Gabriele Cosmo (CERN)
Geant4 Beginners Course

- Introduction
- User Detector Construction class
- Solids/shapes & constructs
- Logical volumes
- Physical volumes
- Geometry checking tools

Introduction

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

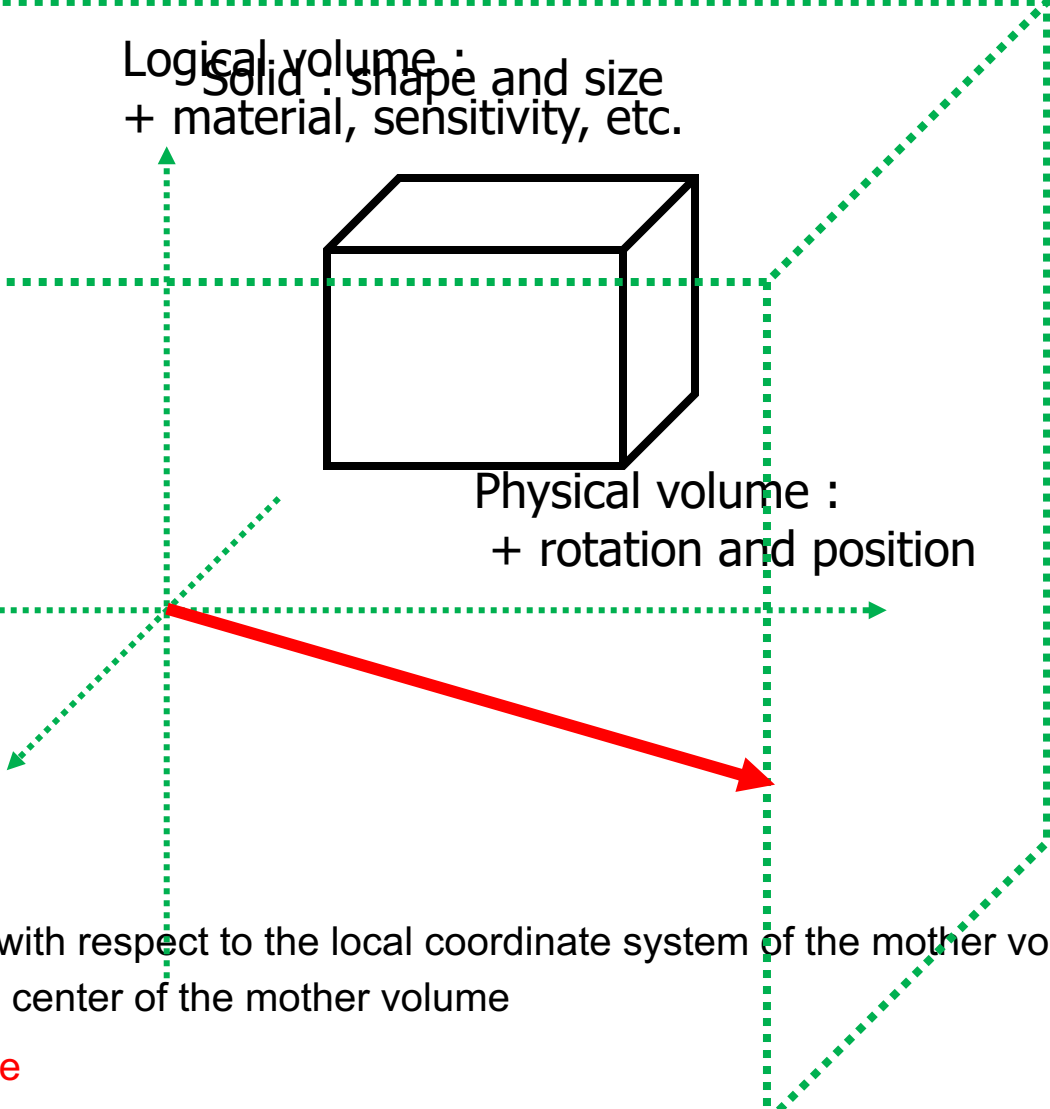


Define detector geometry

- Basic strategy

Always specify the units !

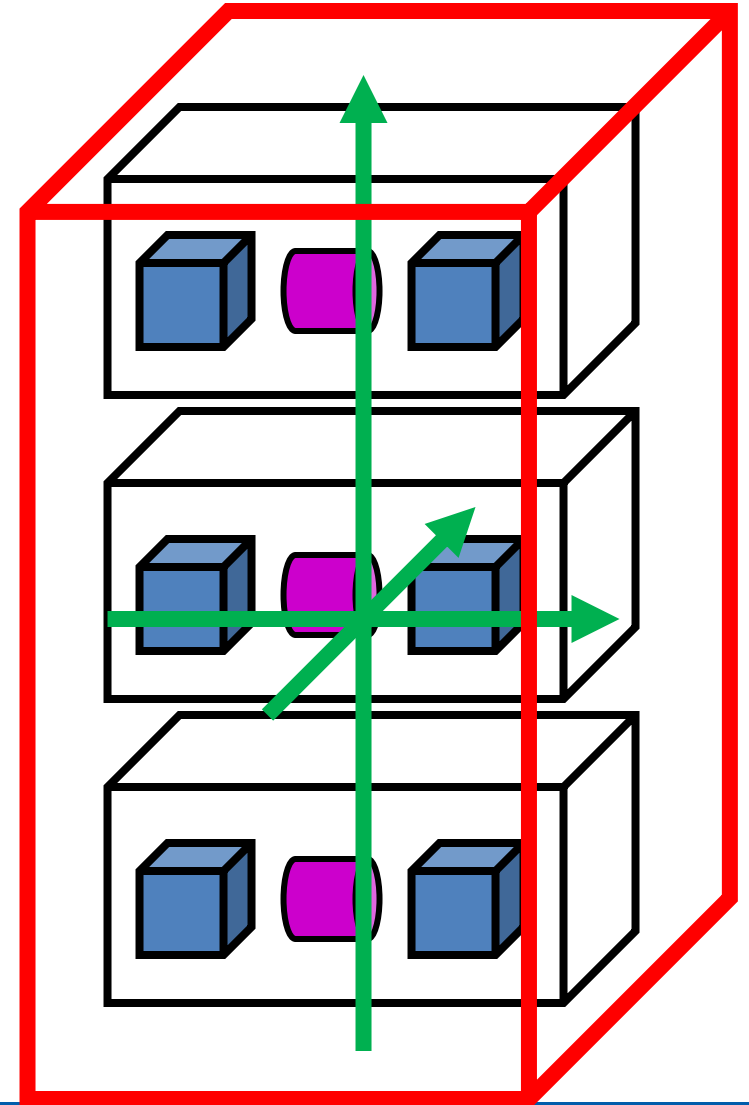
```
G4VSolid* pBoxSolid =  
    new G4Box("aBoxSolid",  
        1.*m, 2.*m, 3.*m);  
  
G4LogicalVolume* pBoxLog =  
    new G4LogicalVolume( pBoxSolid,  
        pBoxMaterial, "aBoxLog", 0, 0, 0);  
  
G4VPhysicalVolume* aBoxPhys =  
    new G4PVPlacement( pRotation,  
        G4ThreeVector(posX, posY,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



Logical volume :
Solid : shape and size
+ material, sensitivity, etc.

Physical volume :
+ rotation and position
- 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

- One logical volume can be placed more than once. One or more volumes can be placed in a mother volume
- Note that the mother-daughter relationship is an information of G4LogicalVolume
 - If the mother volume is placed more than once, all its daughters are by definition appearing in all placed physical volumes
- The **world volume** must be a unique physical volume which **fully contains with some margin** all 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 Detector Construction

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

Note: classes written in **red** are mandatory
- Initialization classes
 - Use `G4RunManager::SetUserInitialization()` to enable the user classes
 - Invoked at the initialization
 - **G4VUserDetectorConstruction**
 - **G4VUserPhysicsList**
 - **G4VUserActionInitialization**
- Action classes
 - Instantiated in `G4VUserActionInitialization`.
 - Invoked during an event loop
 - **G4VUserPrimaryGeneratorAction**
 - `G4UserRunAction`
 - `G4UserEventAction`
 - `G4UserStackingAction`
 - `G4UserTrackingAction`
 - `G4UserSteppingAction`




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

    virtual G4VPhysicalVolume* Construct() = 0;
    virtual void ConstructSDandField();
}
```

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

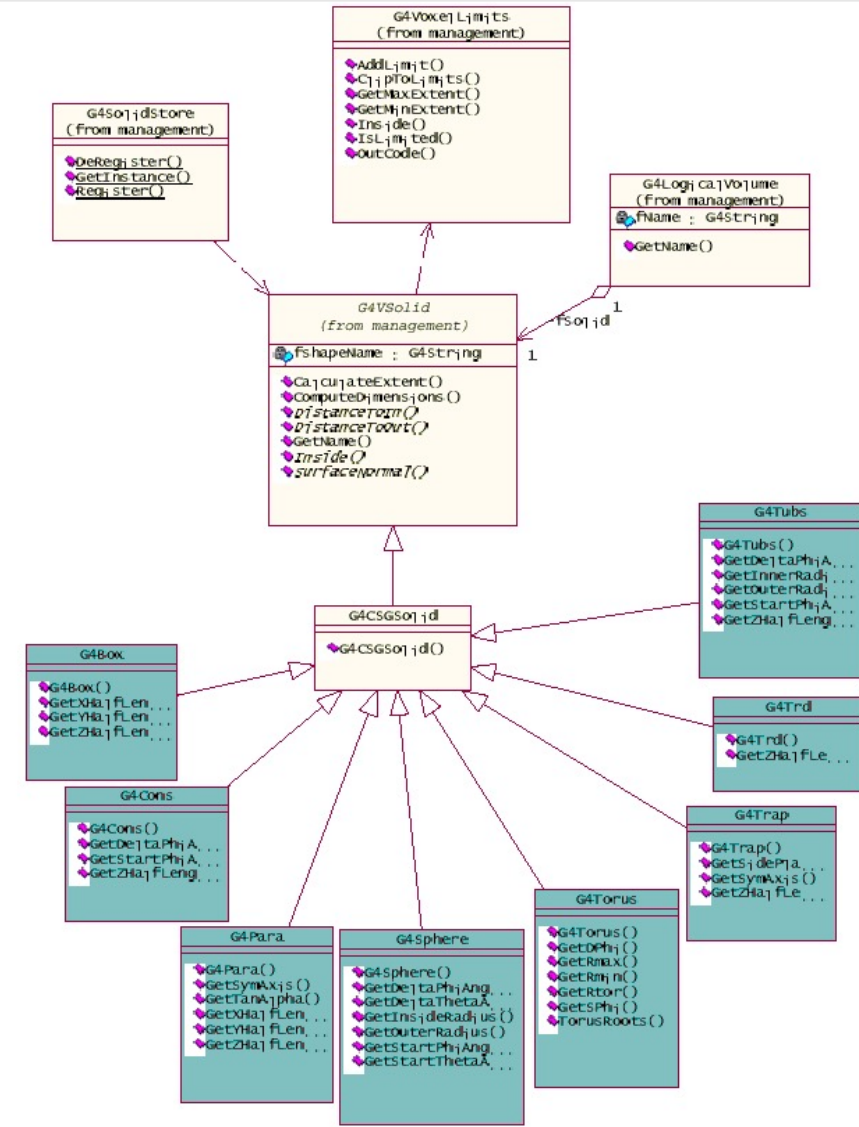
Sensitive detector and field should be instantiated and set to logical volumes in the *ConstructSDandField()* method

Describe your detector

- Derive your own concrete class from **G4VUserDetectorConstruction** abstract base class
- Implement **Construct()** and **ConstructSDandField()** methods
 - 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 the Run Manager

Solids/shapes & constructs

- Abstract class. All solids in Geant4 are derived from it
- It defines but not implement all functions required to:
 - compute distances between the shape and a given point
 - 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
- User can create his/her own solid class



CSG (Constructed Solid Geometry) solids

- ▶ G4Box, G4Tubs, G4Cons, G4Trd, G4Para, G4Trap, G4Torus, G4CutTubs, G4Orb, G4Sphere

Specific solids (CSG like)

- ▶ G4Polycone, G4Polyhedra, G4Hype, G4Ellipsoid, G4EllipticalTube, G4Tet, G4EllipticalCone, G4Hype, G4GenericPolycone, G4GenericTrap, G4Paraboloid

Tessellated solids

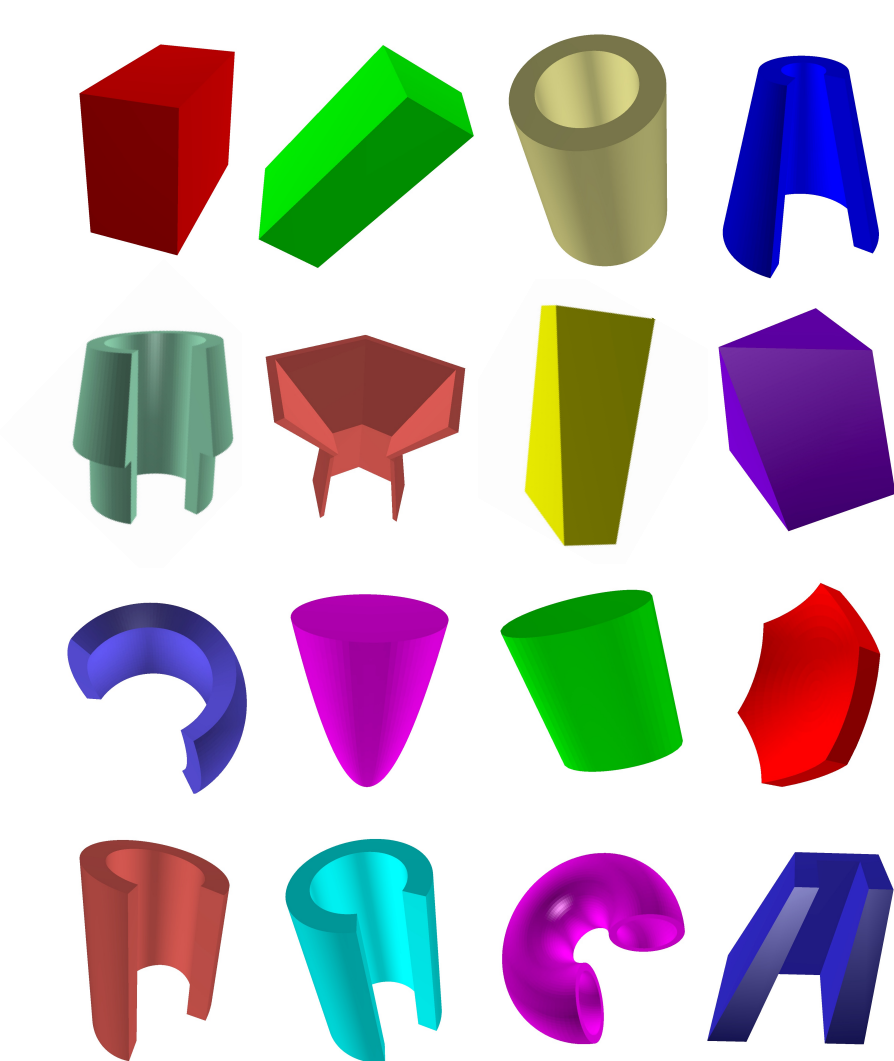
- ▶ G4TessellatedSolid, G4ExtrudedSolid

Boolean & scaled solids

- ▶ G4UnionSolid, G4SubtractionSolid, G4IntersectionSolid, G4MultiUnion, G4ScaledSolid

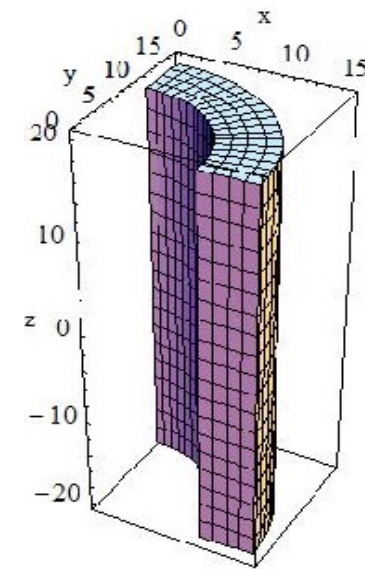
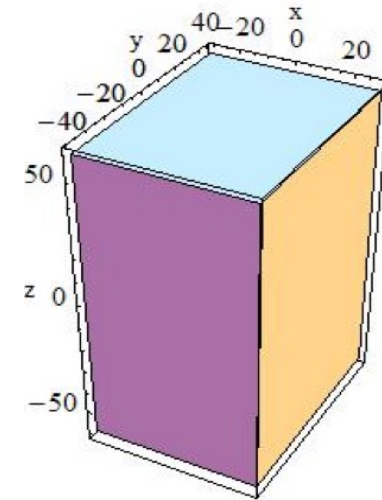
Twisted shapes

- ▶ G4TwistedBox, G4TwistedTrap, G4TwistedTrd, G4TwistedTubs

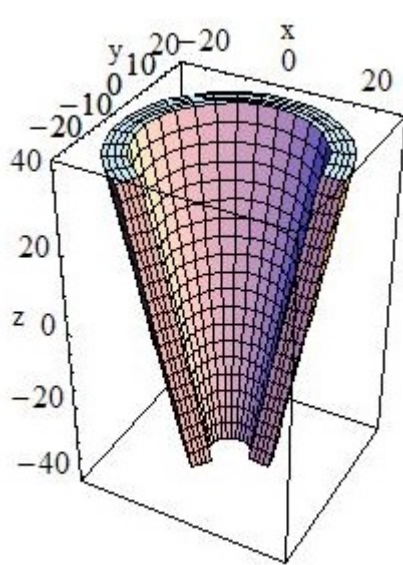


```
G4Box(const G4String &pname,    // name
      G4double half_x,         // X half size
      G4double half_y,         // Y half size
      G4double half_z);        // Z half size

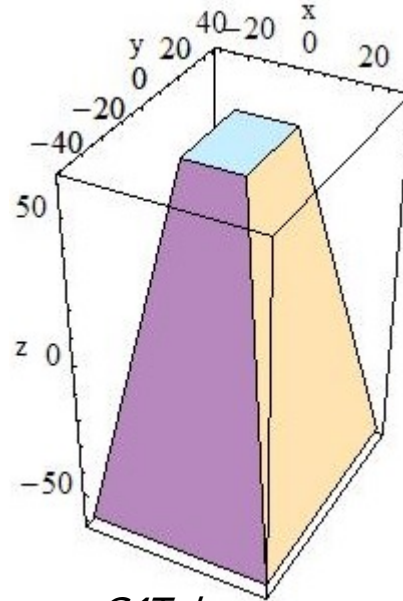
G4Tubs(const G4String &pname,    // name
      G4double pRmin,          // inner radius
      G4double pRmax,          // outer radius
      G4double pDz,            // Z half length
      G4double pSphi,          // starting Phi
      G4double pDphi);         // segment angle
```



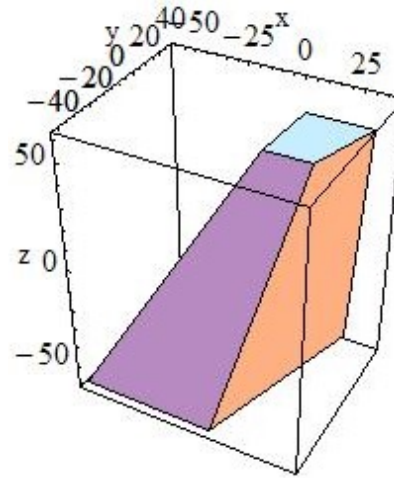
Other CSG Solids



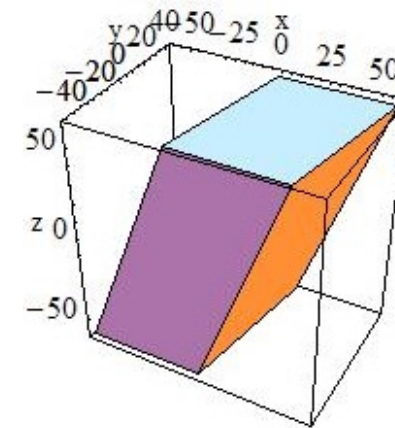
G4Cons



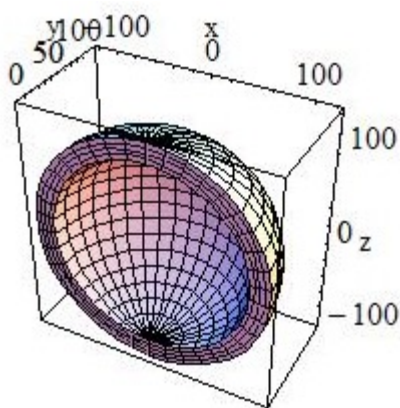
G4Trd



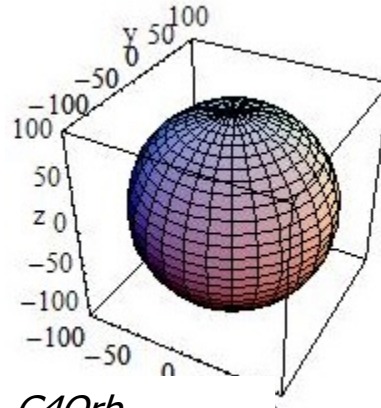
G4Trap



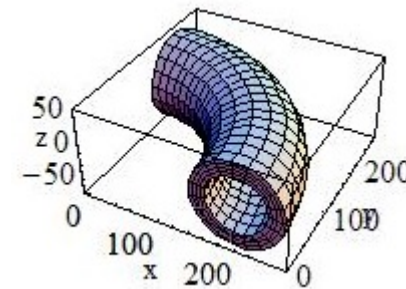
G4Para
(parallelepiped)



G4Sphere



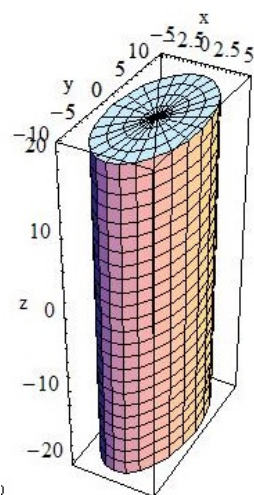
G4Orb
(full solid sphere)



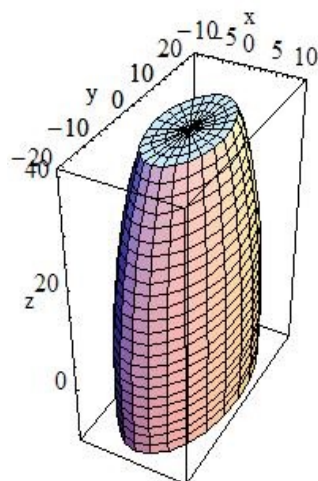
G4Torus

Consult the [Geant4 Application Developers Guide](#) for all available shapes

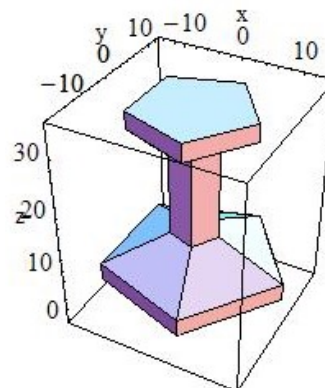
Other specific CSG solids



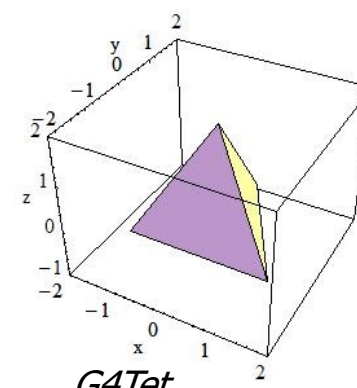
G4EllipticalTube



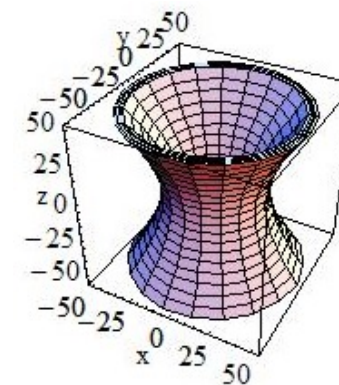
G4Ellipsoid



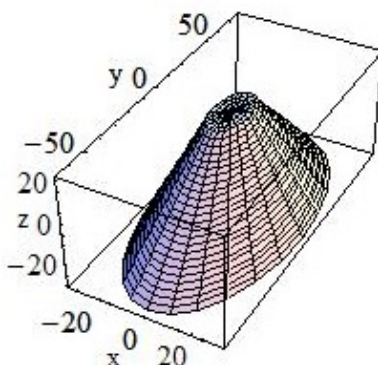
G4Polyhedra



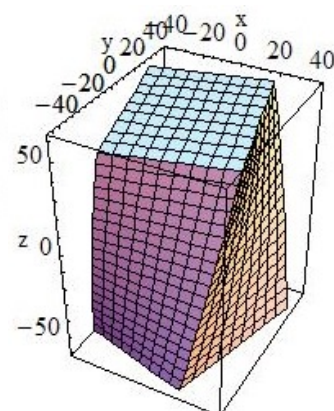
G4Tet
(tetrahedra)



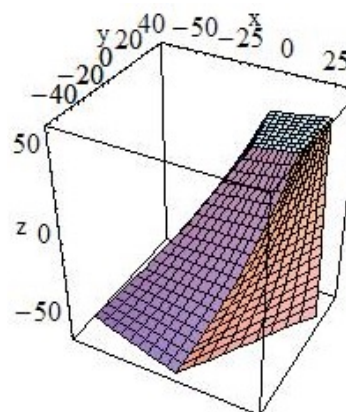
G4Hype



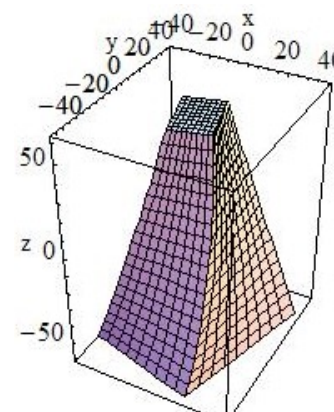
G4EllipticalCone



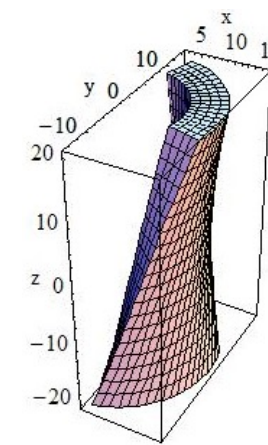
G4TwistedBox



G4TwistedTrap



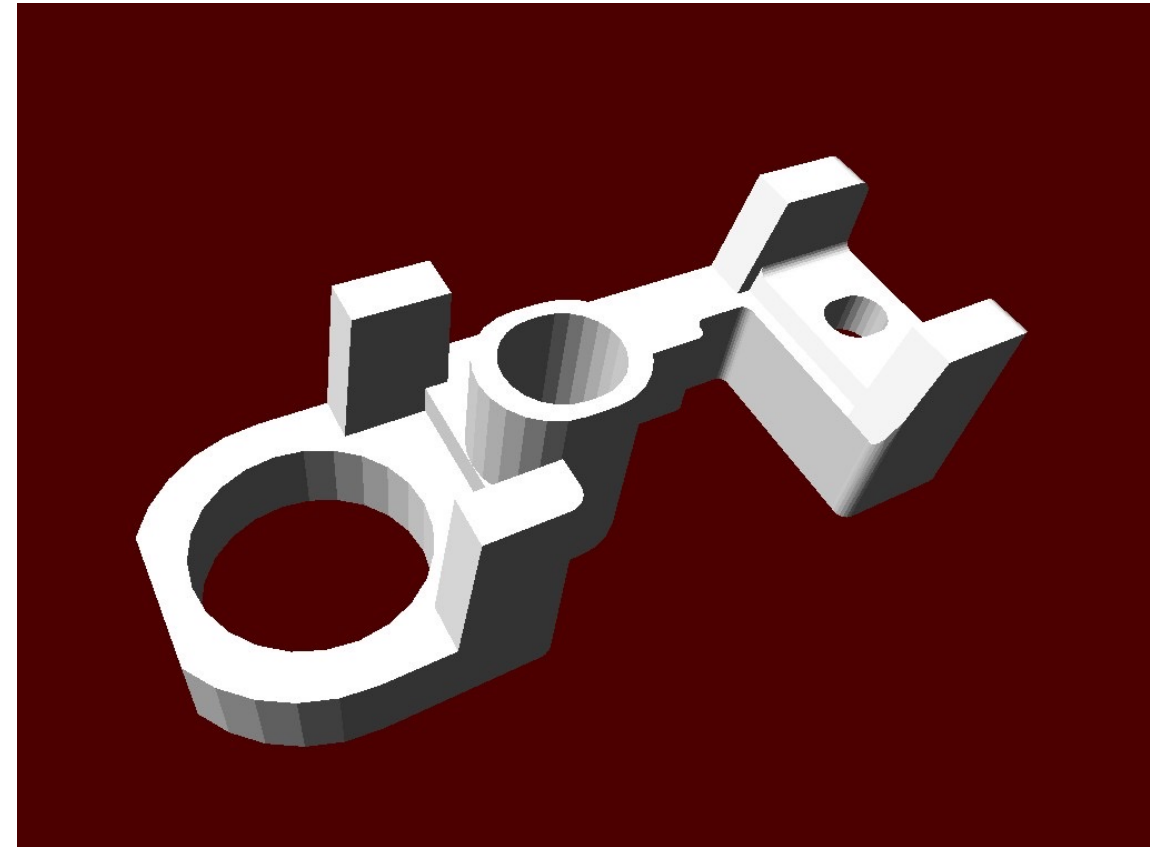
G4TwistedTrd

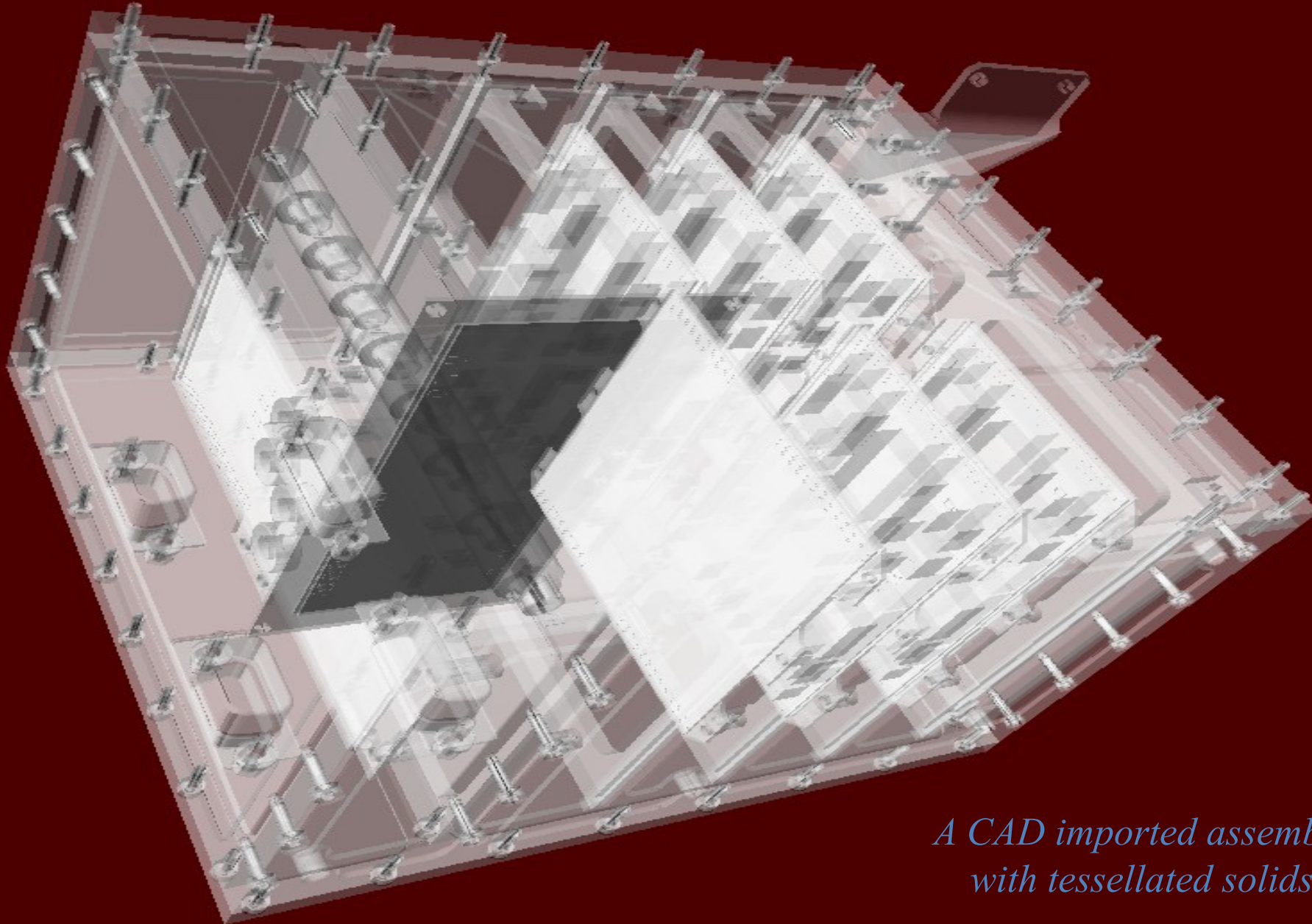


G4TwistedTubs

Consult the [Geant4 Application Developers Guide](#) for all available shapes

- **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

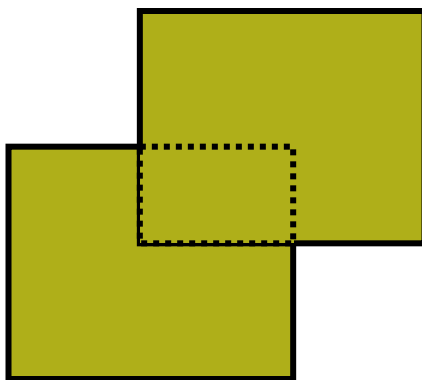




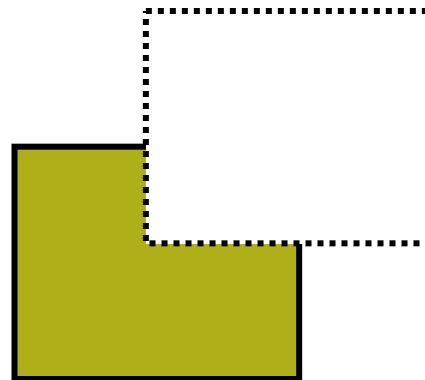
*A CAD imported assembly
with tessellated solids*

- ▶ 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
 - ▶ A third solid can be combined to the resulting solid of first operation
 - ▶ So: solids can be combined with other Boolean solids

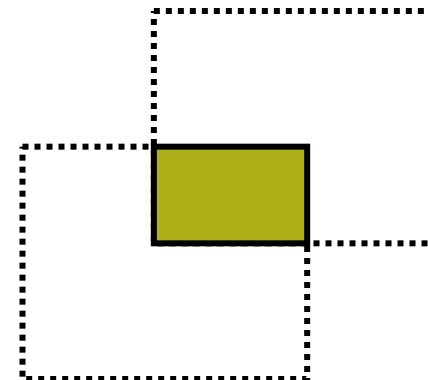
G4UnionSolid



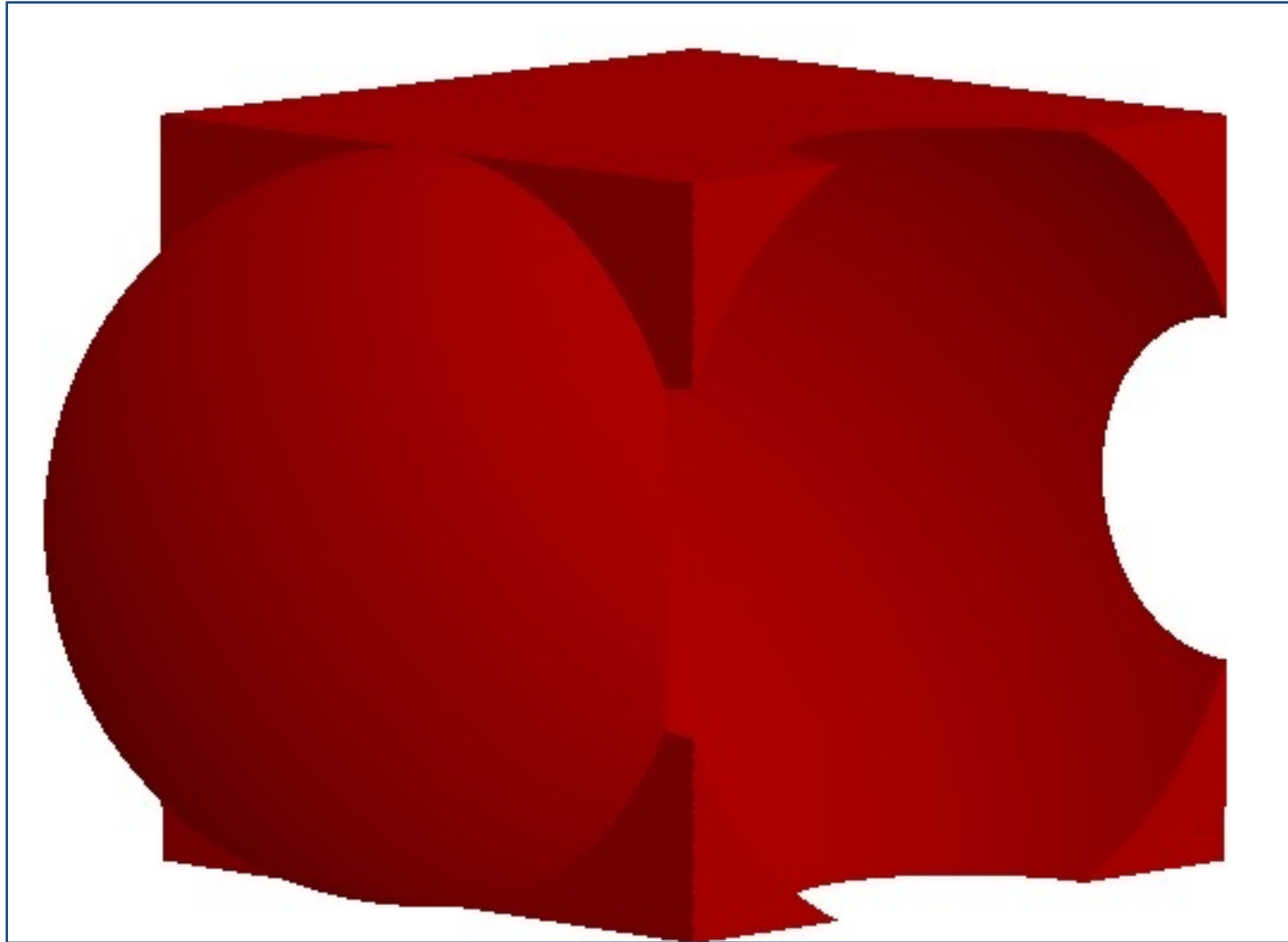
G4SubtractionSolid



G4IntersectionSolid



Boolean solid



Logical Volumes

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

- Contains all information of a 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 pointer to the solid must **NOT** be null
- The pointer to the material must **NOT** be null for the tracking geometry
- It is not meant to act as a base class

- The geometrical capacity of a generic solid or Boolean composition can be computed from the **solid**:

```
G4double GetCubicVolume() ;
```

- Exact capacity is mathematically calculated for most of the solids, while estimation based on Monte Carlo integration may be given for other solids

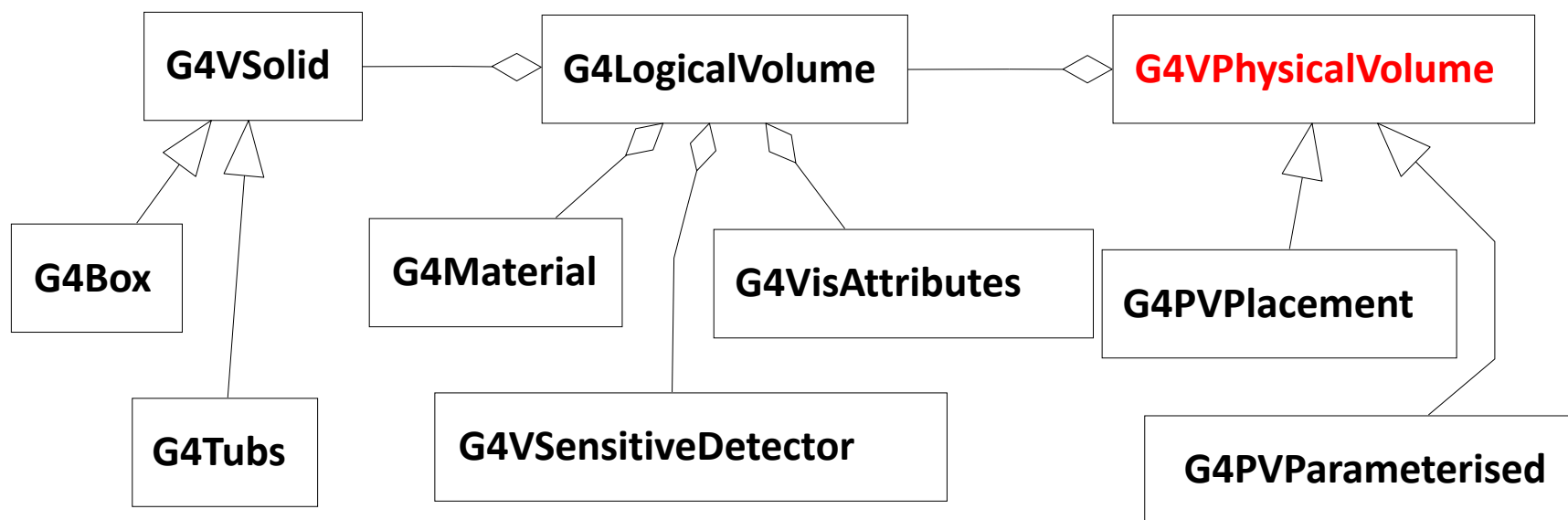
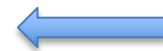
- Overall weight of a geometry setup can be computed from the **logical volume**:

```
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***

Physical Volumes

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



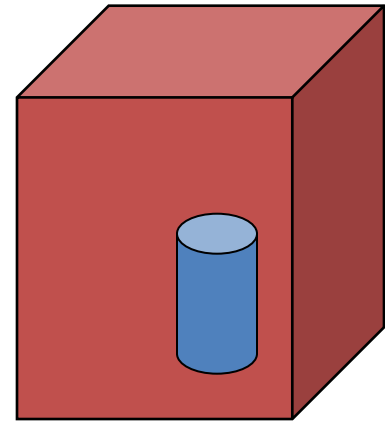
Define a detector geometry

Always specify the units !

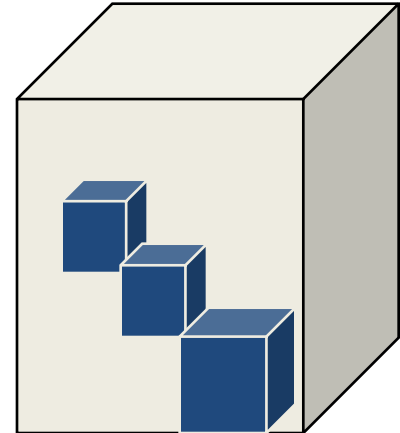
- Basic strategy

```
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);
```

- 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*
 - Shape, size, material, sensitivity, vis attributes, position and rotation can be parameterized by the **copy number**
 - *Replica and Division*
 - simple repetition along one axis
- A mother volume can contain **either**
 - many placement volumes
 - **or**, one repeated volume



placement

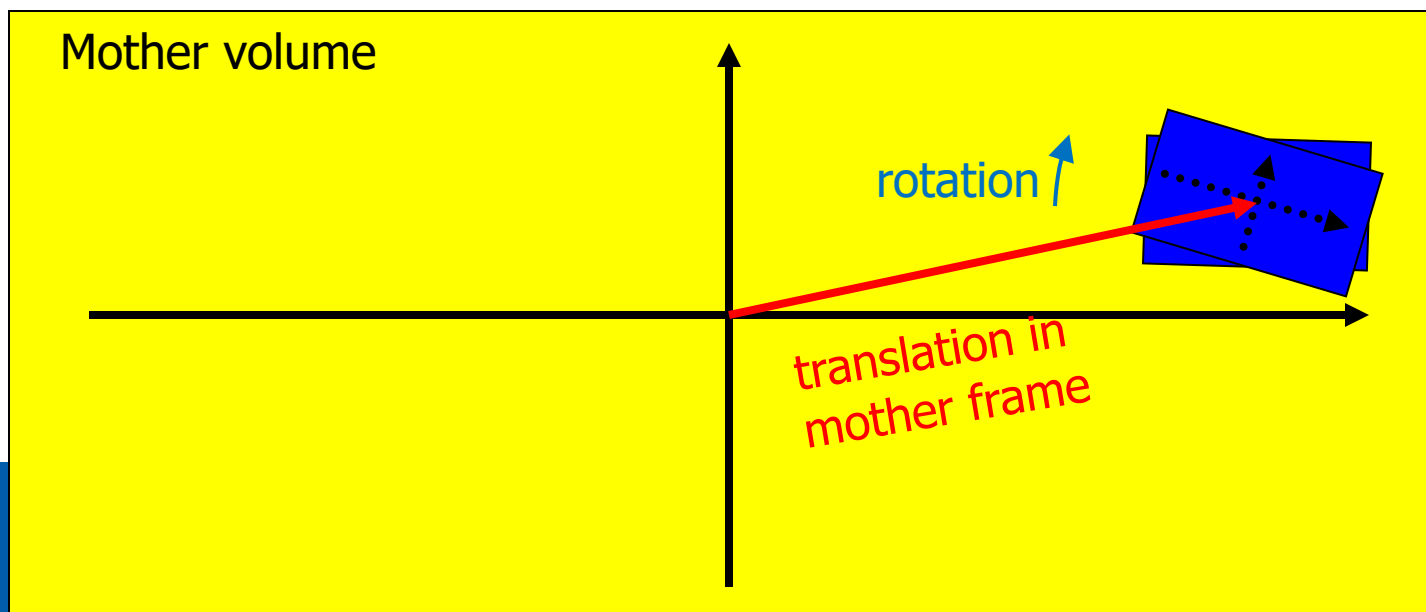


repeated

G4PVPlacement(

```
G4Transform3D(G4RotationMatrix &pRot, // rotation of daughter volume
               const G4ThreeVector &tlate), // position in mother frame
G4LogicalVolume *pDaughterLogical,
const G4String &pName,
G4LogicalVolume *pMotherLogical,
G4bool pMany, // not supported/for future use...
G4int pCopyNo, // unique arbitrary integer
G4bool pSurfChk=false); // optional boundary check
```

- Single volume positioned relatively to the mother volume



Alternative G4PVPlacement construct

```
G4PVPlacement(G4RotationMatrix* pRot,    // rotation of mother frame
              const G4ThreeVector &tlate, // position in mother frame
              G4LogicalVolume *pDaughterLogical,
              const G4String &pName,
              G4LogicalVolume *pMotherLogical,
              G4bool pMany, // not supported/for future use...
              G4int pCopyNo, // unique arbitrary integer
              G4bool pSurfChk=false); // optional boundary check
```

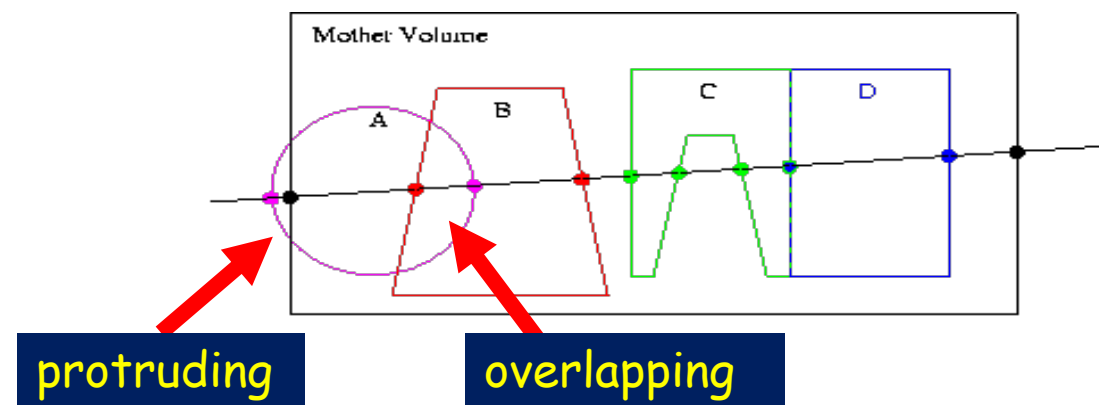
- Single volume positioned relatively to the mother volume frame

Note:

- This G4PVPlacement is identical to the previous one if there is no rotation
 - For power-users: previous one is much easier to understand
- The advantage of this second constructor is setting the pointer of the rotation matrix rather than providing the values of the matrix
 - You may change the matrix without accessing to the physical volume

Checking Geometry for overlaps

- A **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 can be unpredictable in such cases
- The problem of detecting overlaps between volumes is bounded by the complexity of the solids model description
- Utilities are provided for detecting wrong positioning:
 - Optional checks at construction
 - Kernel run-time commands
 - Graphical tools (vis commands)



- 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 (1000 by default) are randomly sampled on the surface when creating the 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 may require lots of CPU time depending on the geometry complexity, it is therefore suggested to apply it to portions of a setup progressively
- Alternatively, one can use explicitly the overlaps check for a simple volume:

```
G4bool CheckOverlaps(G4int res=1000, G4double tol=0., G4bool verbose=true)
```


- Built-in run-time commands to activate verification tests for the user geometry are defined
 - to start verification of geometry for overlapping regions recursively through the volumes tree:
`geometry/test/run`
 - To set the starting depth level in the volumes tree from where checking for overlaps. Default is level '0' (i.e. the world volume):
`geometry/test/recursion_start [int]`
 - To set the total depth in the volume tree for checking for overlaps. Default is '-1' (i.e. checking the whole tree). Recursion will stop after having reached the specified depth":
`geometry/test/recursion_depth [int]`
 - To define tolerance by which overlaps should not be reported. Default is '0':
`geometry/test/tolerance [double] [unit]`
 - To set verbosity mode. Default is 'true':
`geometry/test/verbosity [bool]`
 - To establish the number of points on surface to be generated and checked for each volume. Default is '10000':
`geometry/test/resolution [int]`
 - To fix the threshold for the number of errors to be reported for a single volume. By default, for each volume, reports stop after the first error reported:
`geometry/test/maximum_errors [int]`

Debugging run-time reports

Example layout:

GeomTest: no daughter volume extending outside mother detected.

GeomTest Error: Overlapping daughter volumes

The volumes Tracker[0] and Overlap[0],

both daughters of volume World[0],

appear to overlap at the following points in global coordinates: (list truncated)

length (cm)	-----	start position (cm)	-----	-----	end position (cm)	-----
240		-240	-145.5	-145.5	0	-145.5

Which in the mother coordinate system are:

length (cm)	-----	start position (cm)	-----	-----	end position (cm)	-----
. . .						

Which in the coordinate system of Tracker[0] are:

length (cm)	-----	start position (cm)	-----	-----	end position (cm)	-----
. . .						

Which in the coordinate system of Overlap[0] are:

length (cm)	-----	start position (cm)	-----	-----	end position (cm)	-----
. . .						