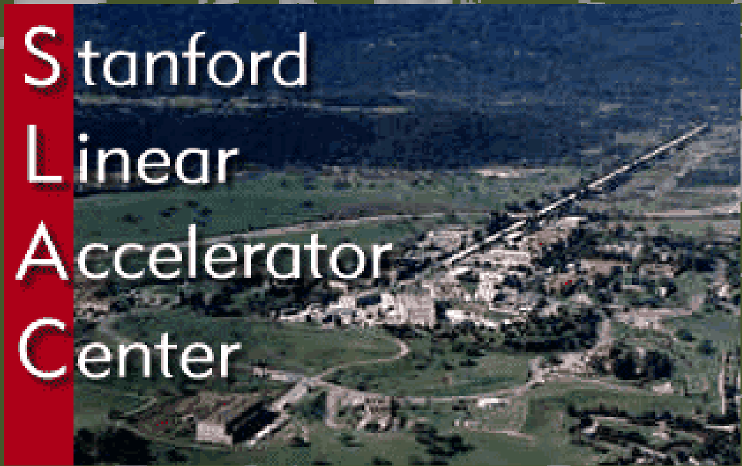


Geometry I

Makoto Asai (SLAC)

Geant4 Tutorial Course

Geant 4

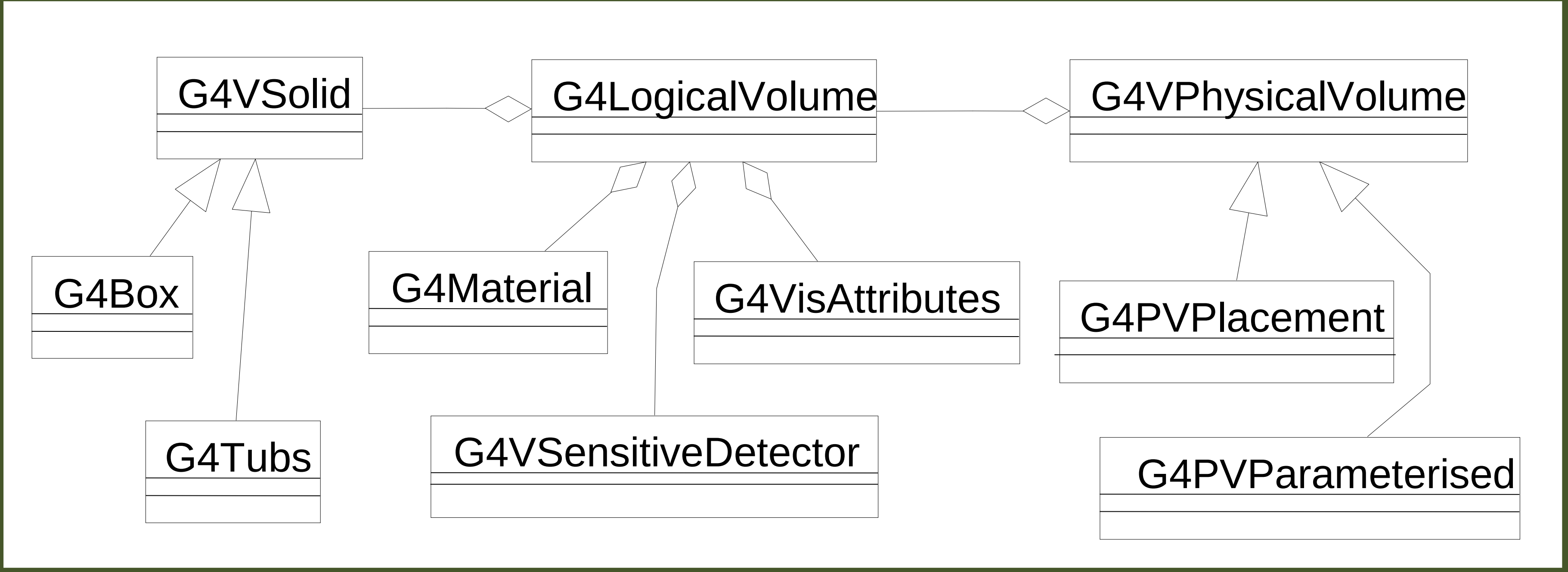
Contents

- Introduction
- G4VUserDetectorConstruction class
- Solid and shape
- Logical volume

Introduction

Detector geometry

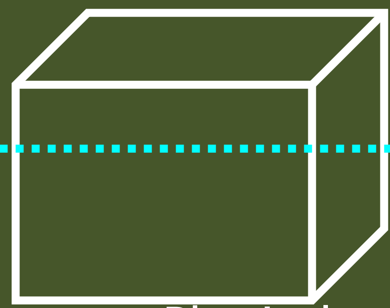
- Three conceptual layers
  - G4VSolid -- *shape, size*
  - G4LogicalVolume -- *daughter physical volumes,*
    material, sensitivity, user limits, etc.
  - G4VPhysicalVolume -- *position, rotation*



Define detector geometry

- Basic strategy

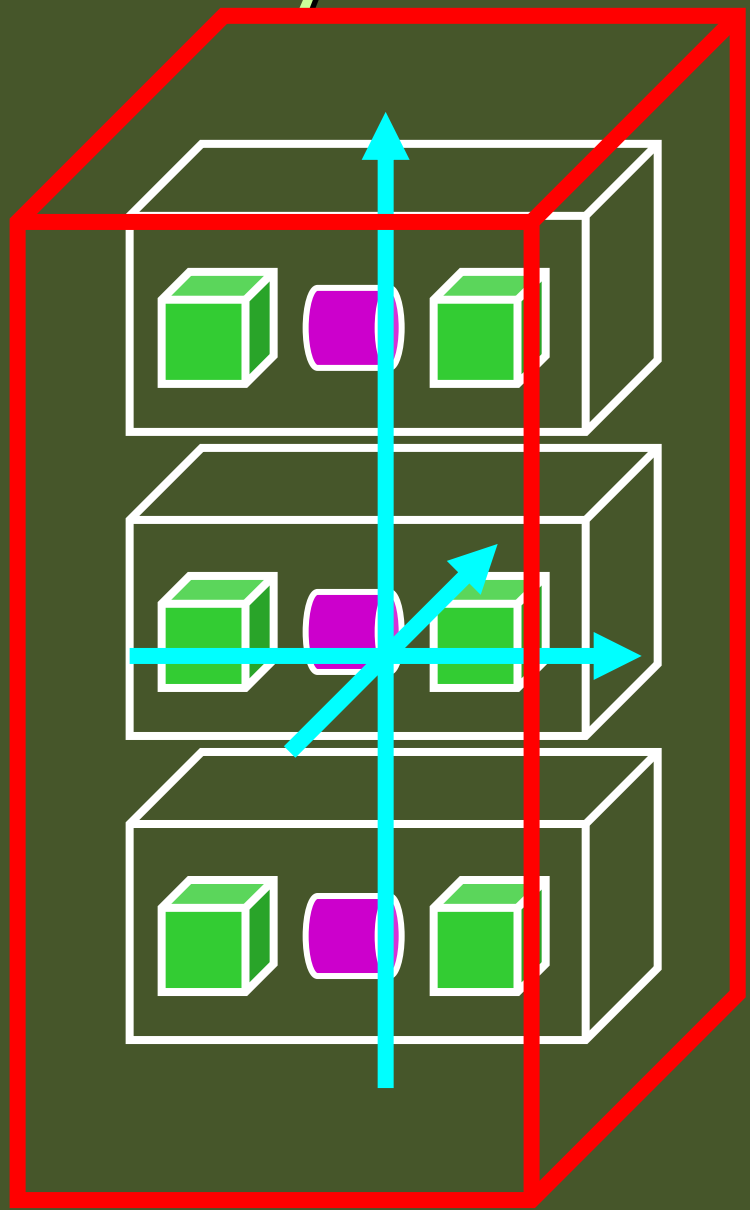
```
G4VSolid* pBoxSolid =
  new G4Box("aBoxSolid",
      1.*m, 2.*m, 3.*m);
G4LogicalVolume* pBoxLog =
  new G4LogicalVolume( pBoxSolid,
    pBoxMaterial, "aBoxLog", 0, 0, 0);
G4VPhysicalVolume* aBoxPhys =
  new G4PVPlacement( pRotation,
    G4ThreeVector(posX, posY, posZ),
    pBoxLog, "aBoxPhys", pMotherLog,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



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

Geometrical hierarchy

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



G4VUserDetectorConstruction

User classes

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

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

G4VUserDetectorConstruction

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

#ifndef G4VUserDetectorConstruction_h
#define G4VUserDetectorConstruction_h 1

class G4VPhysicalVolume;

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

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

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

#endif
```

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

Your detector construction

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

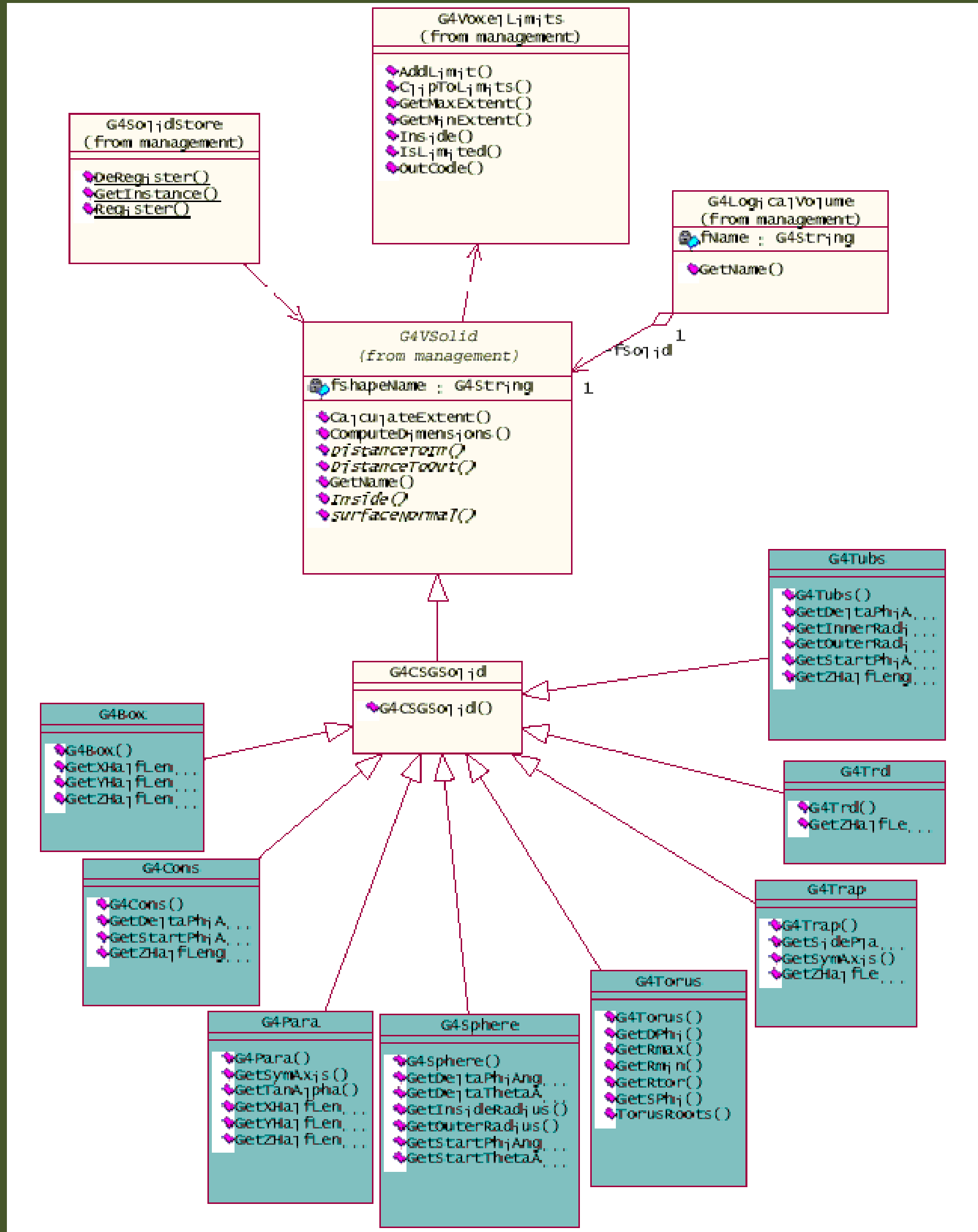
Describe your detector

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

Solid and shape

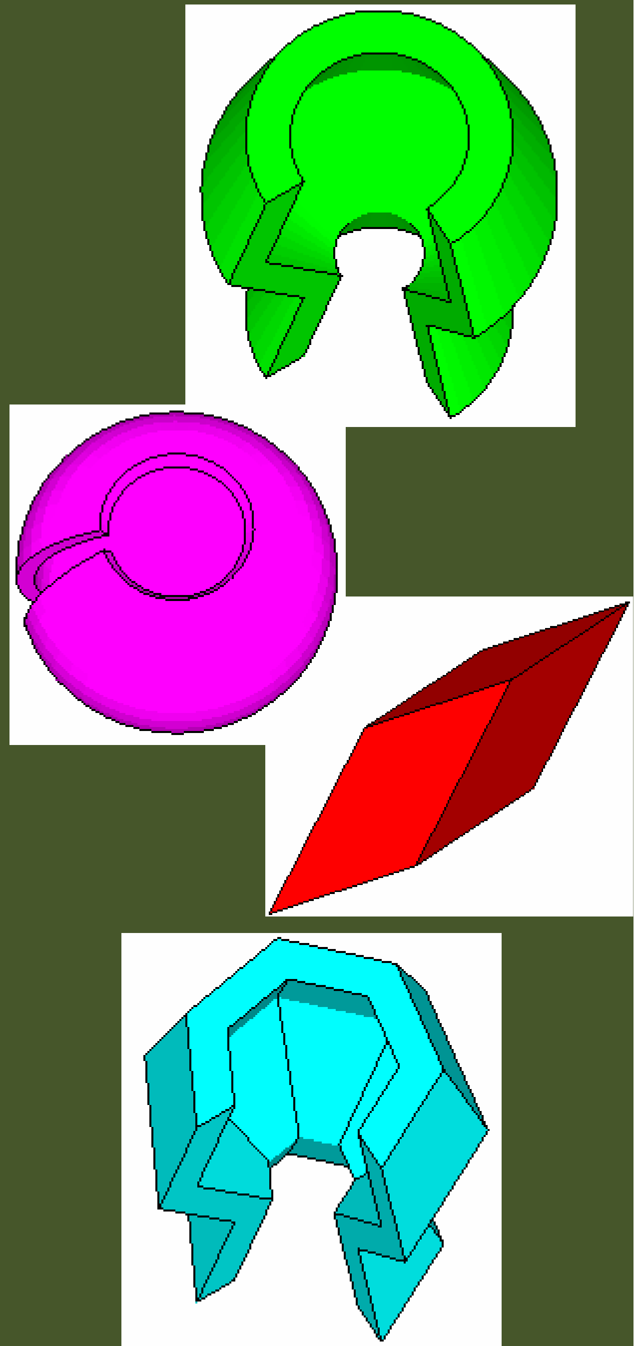
G4VSolid

- Abstract class. All solids in Geant4 are derived from it
- It defines but does 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



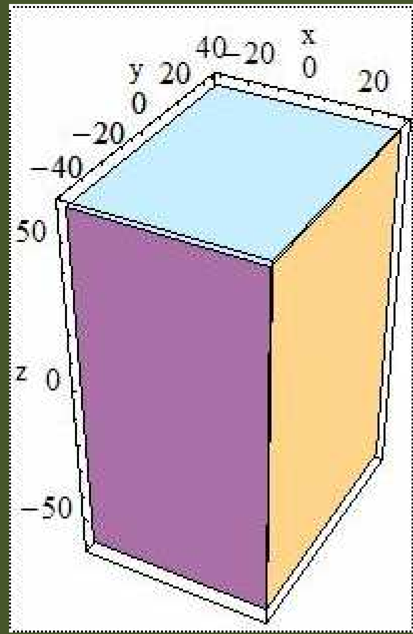
Solids

- 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, …
  - BREP (Boundary REPresented) solids
    - G4BREPSolidPolycone, G4BSplineSurface, …
    - Any order surface
  - Boolean solids
    - G4UnionSolid, G4SubtractionSolid, …
  - Tessellated solid and Extruded solid

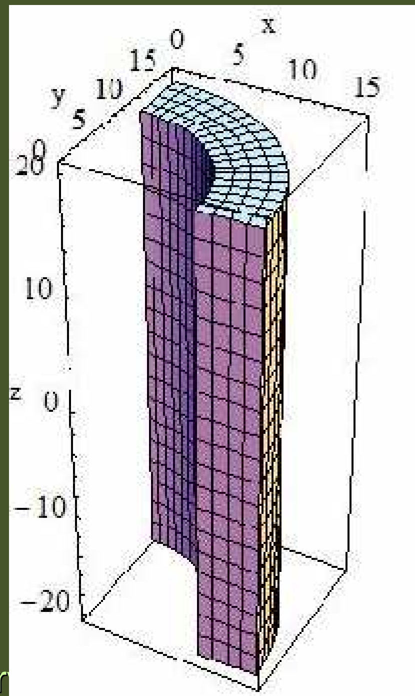


CSG: G4Box, G4Tubs

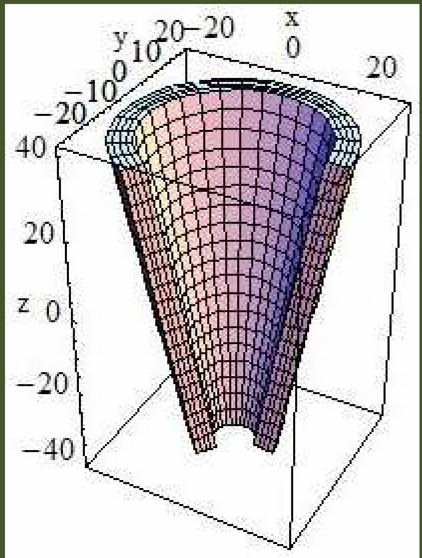
```
G4Box(const G4String &pname,   // name
            G4double half_x,   // X half size
            G4double half_y,   // Y half size
            G4double half_z);  // Z half size
```



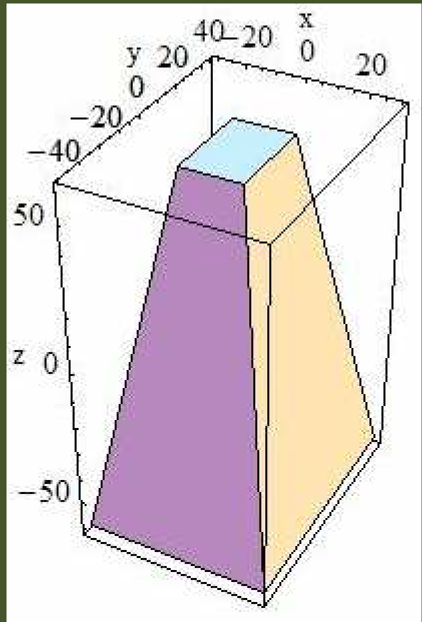

```
G4Tubs(const G4String &pname,  // name
             G4double  pRmin,  // inner radius
             G4double  pRmax,  // outer radius
             G4double  pDz,    // Z half length
             G4double  pSphi,  // starting Phi
             G4double  pDphi); // segment angle
```



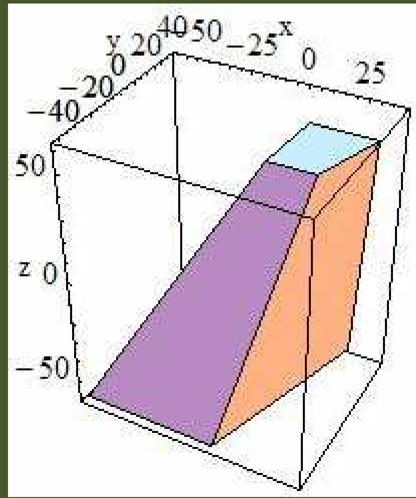
Other CSG solids



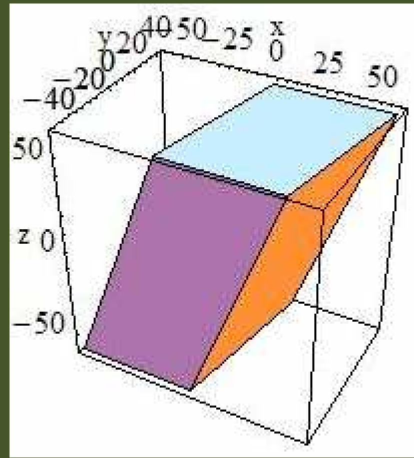
G4Cons



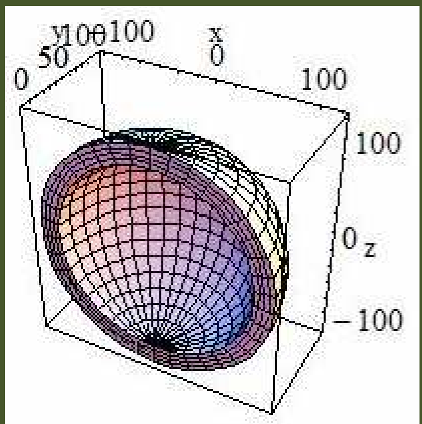
G4Trd



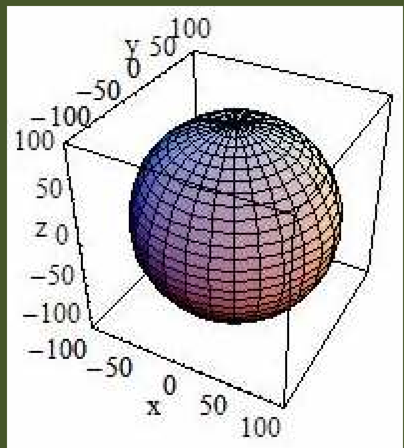
G4Trap



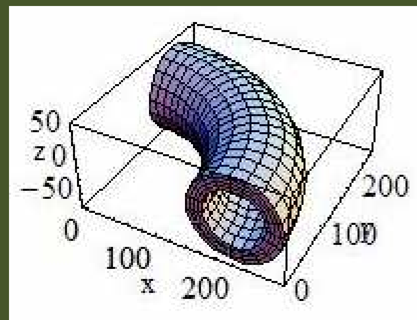
G4Para
(parallelepiped)



G4Sphere



G4Orb
(full solid sphere)

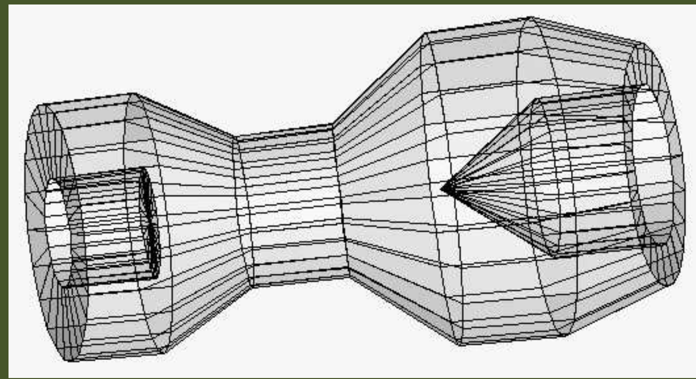


G4Torus

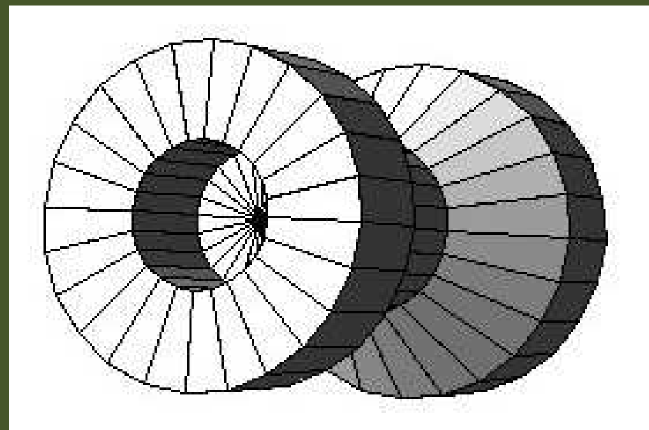
Consult to [Section 4.1.2 of Geant4 Application Developers Guide] for all available shapes.

Specific CSG Solids: G4Polycone

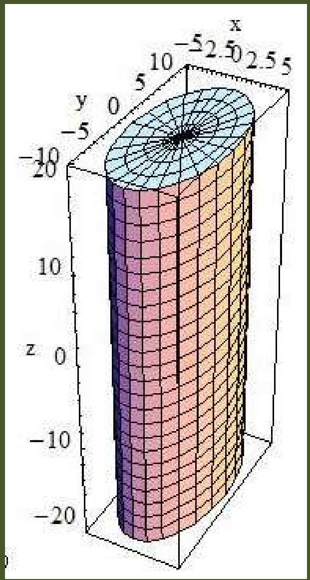
```
G4Polycone(const G4String& pName,
           G4double phiStart,
           G4double phiTotal,
           G4int numRZ,
           const G4double r[],
           const G4double z[]);
```



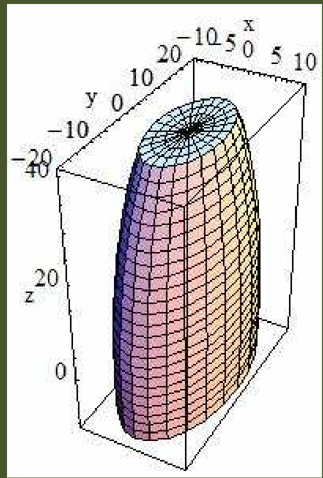
- **`numRZ`** - numbers of corners in the `r,z` space
- **`r, z`** - coordinates of corners



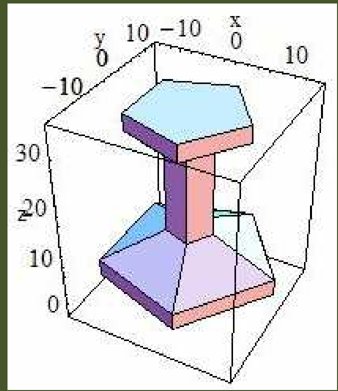
Other Specific CSG solids



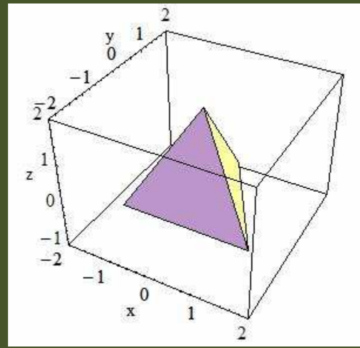
G4EllipticalTube



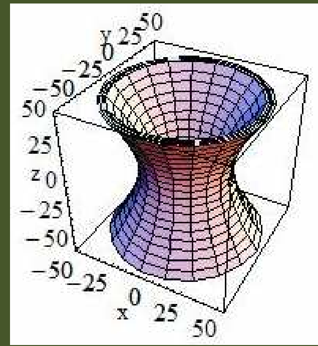
G4Ellipsoid



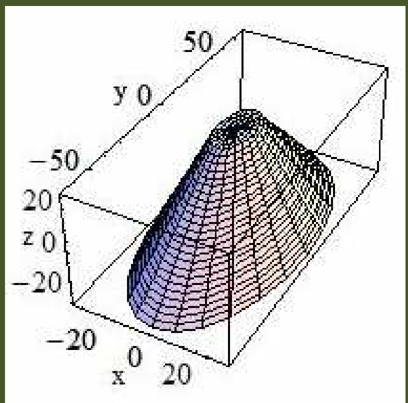
G4Polyhedra



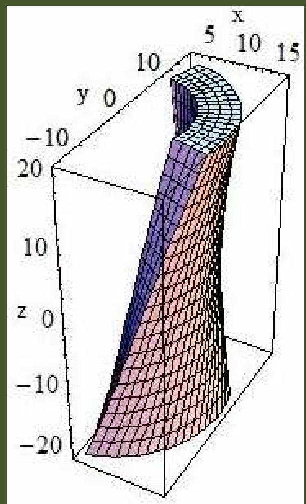
G4Tet
(tetrahedra)



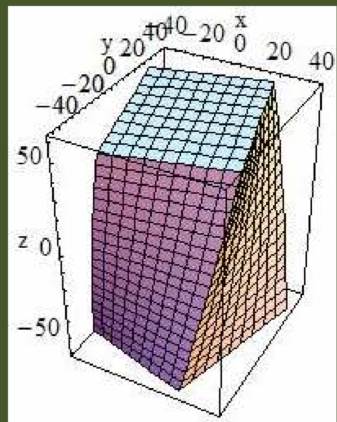
G4Hype



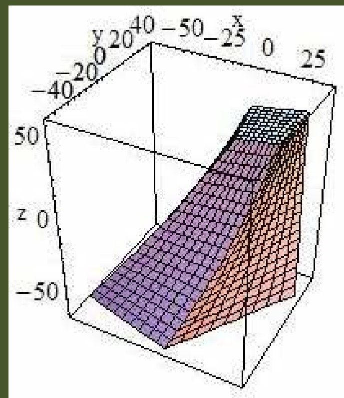
G4EllipticalCone



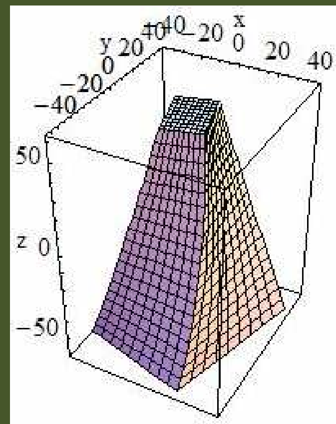
G4TwistedTubs



G4TwistedBox



G4TwistedTrap

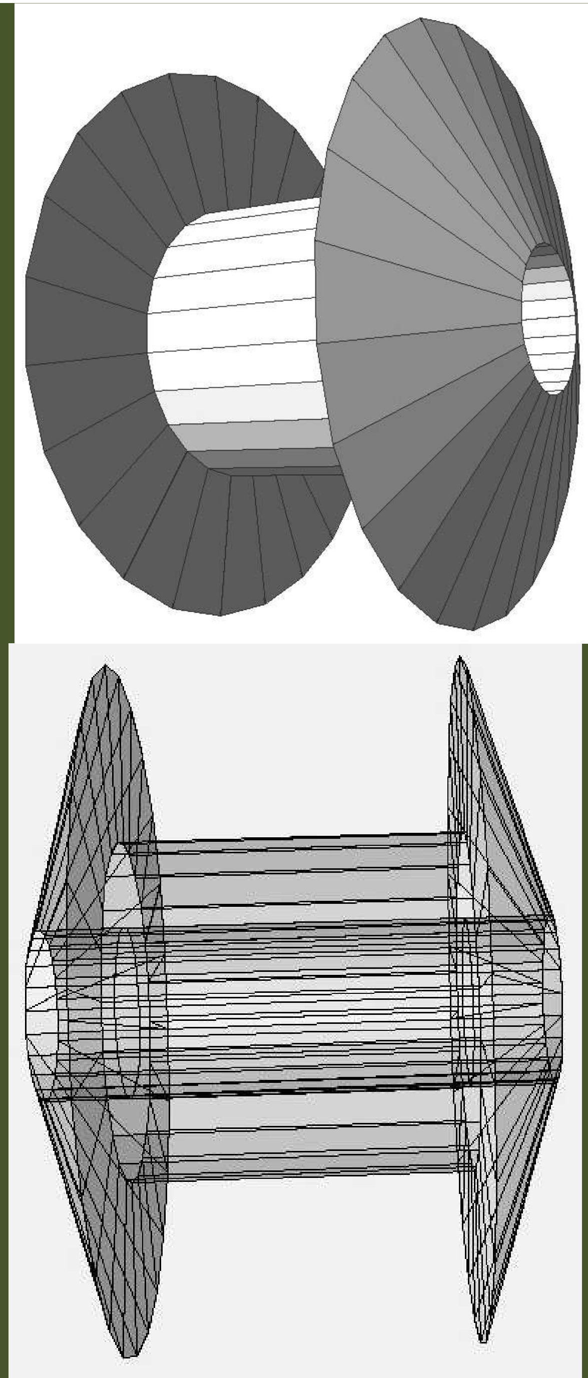


G4TwistedTrd

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

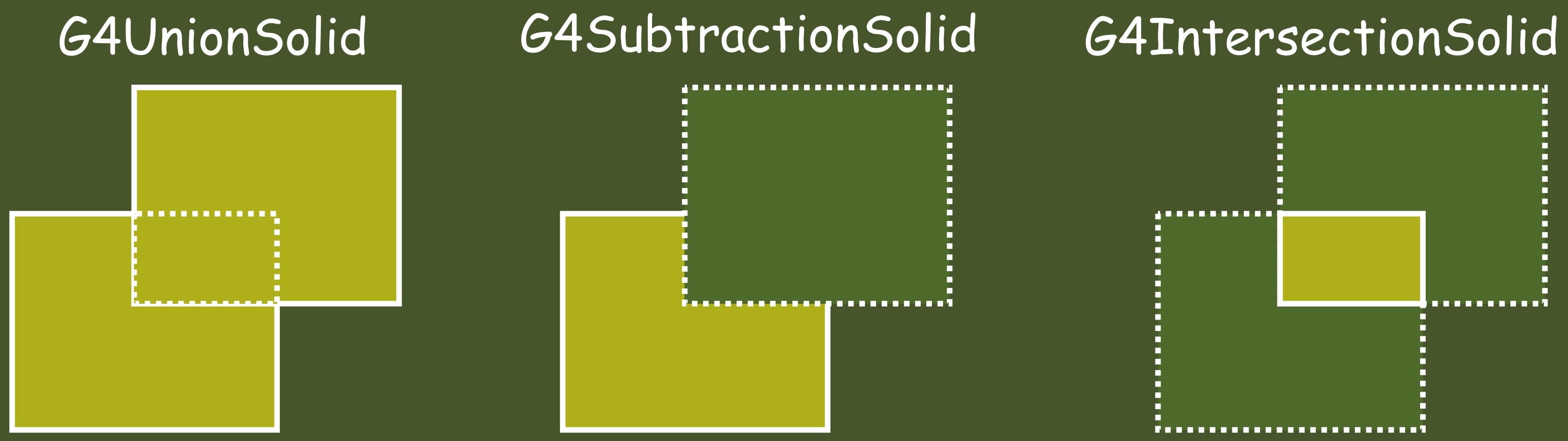
BREP Solids

- *BREP = Boundary REPresented Solid*
- Listing all its surfaces specifies a solid
  - e.g. 6 planes for a cube
- Surfaces can be
  - planar, 2^{nd} or higher order
    - elementary BREPS
  - Splines, B-Splines,
    NURBS (Non-Uniform B-Splines)
    - advanced BREPS
- Few elementary BREPS pre-defined
  - box, cons, tubs, sphere, torus, polycone, polyhedra
- Advanced BREPS built through CAD systems

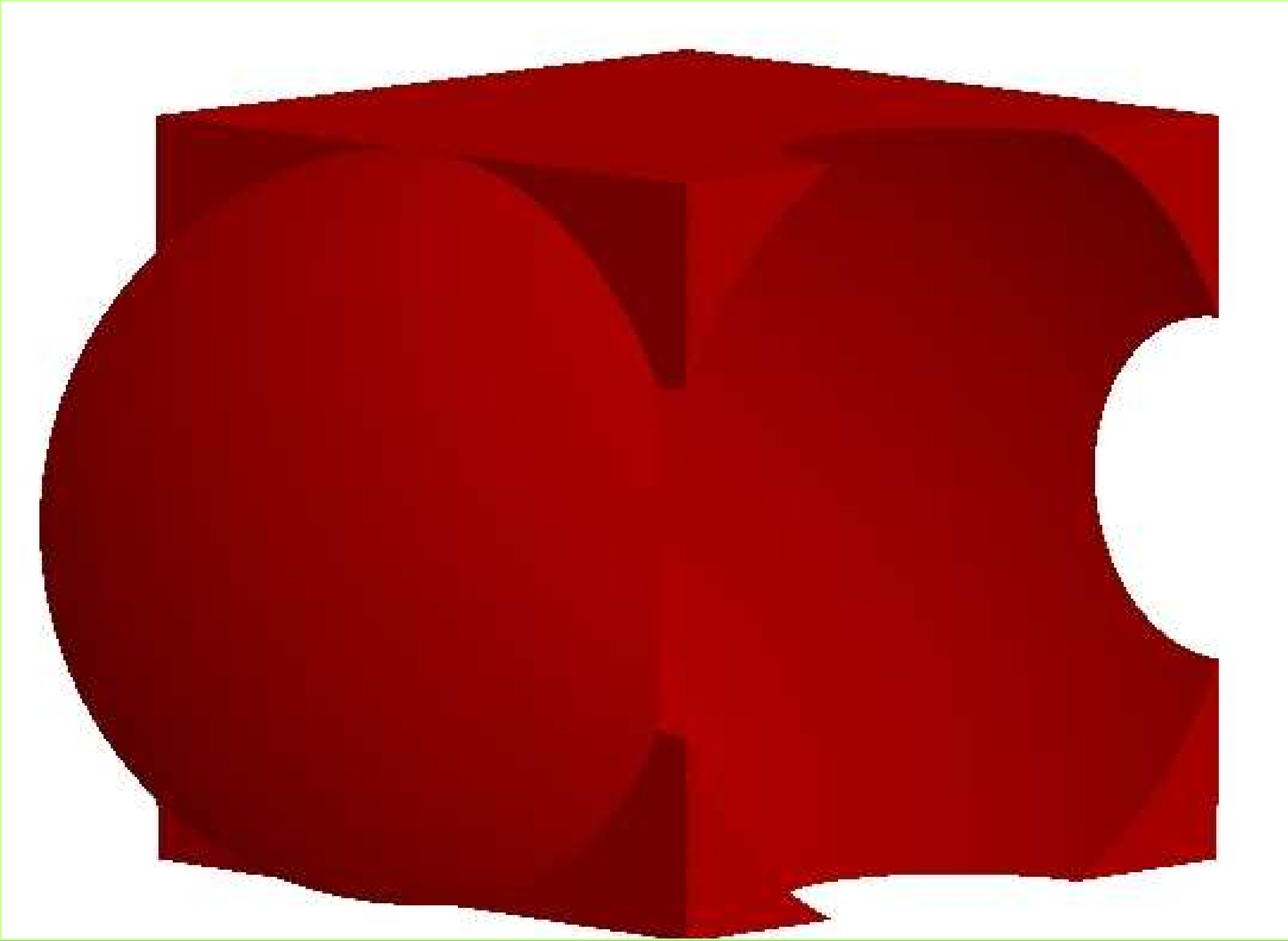


Boolean Solids

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



Boolean solid



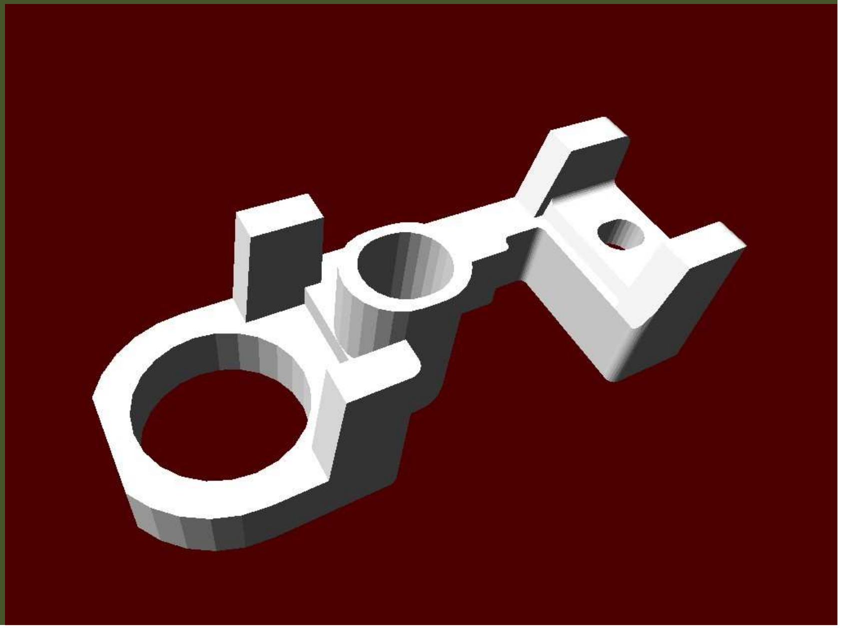
Boolean Solids - example

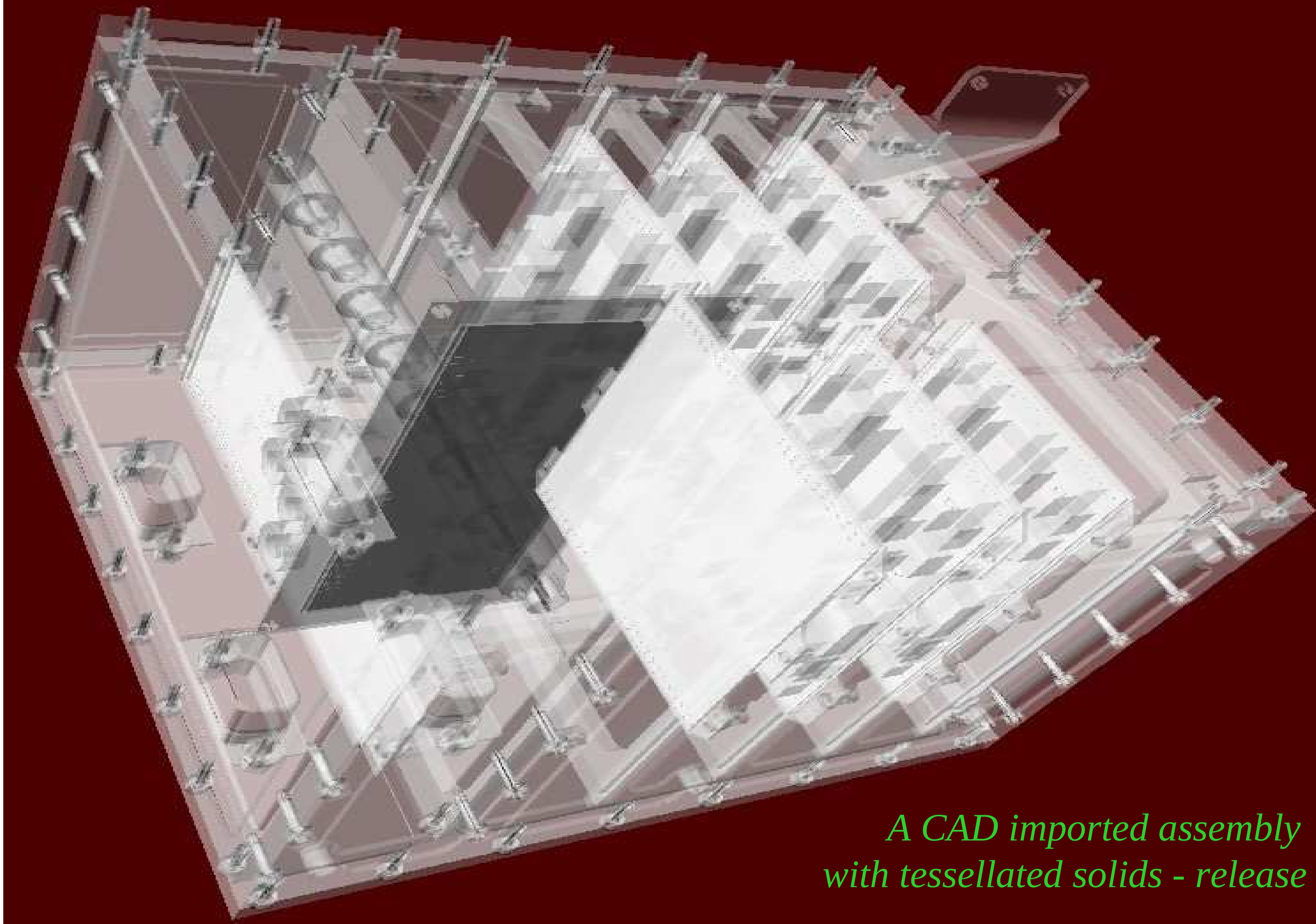
```
G4VSolid* box = new G4Box(“Box",50*cm,60*cm,40*cm);
G4VSolid* cylinder
 = new G4Tubs(“Cylinder”,0.,50.*cm,50.*cm,0.,2*M_PI*rad);
G4VSolid* union
 = new G4UnionSolid("Box+Cylinder", box, cylinder);
G4VSolid* subtract
 = new G4SubtractionSolid("Box-Cylinder", box, cylinder,
       0, G4ThreeVector(30.*cm,0.,0.));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

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

Tessellated solids

- **`G4TessellatedSolid`** (since 8.1)
  - 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
  - But 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

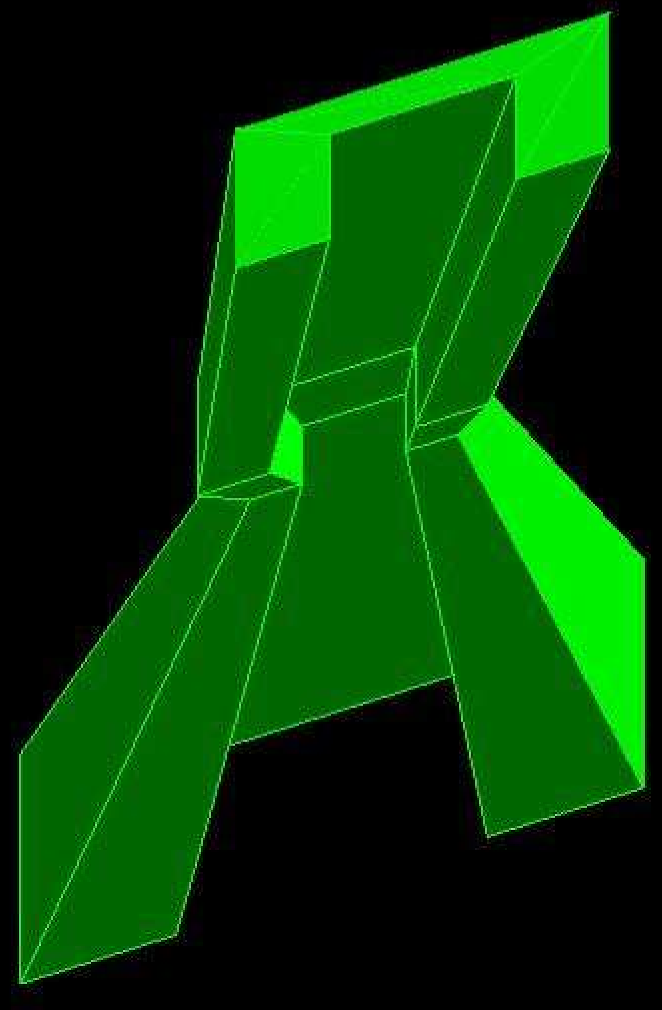




A CAD imported assembly
with tessellated solids - release 8.1

G4ExtrudedSolid

- G4ExtrudedSolid, which was introduced in v8.3, is a specific case of tessellated solid.
- G4ExtrudedSolid is a solid which represents the extrusion of an arbitrary polygon with fixed outline in the defined Z sections.
- The z-sides of the solid are the scaled versions of the same polygon.
- The solid is implemented as a specification of G4TessellatedSolid.



G4LogicalVolume

G4LogicalVolume

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

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

Computing volumes and weights

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

```
G4double GetCubicVolume();
```

  - Exact volume is determinatively calculated for most of CSG solids, while estimation based on Monte Carlo integration is given for other solids.
- Overall weight of a geometry setup (sub-geometry) can be computed from the logical volume:

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

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

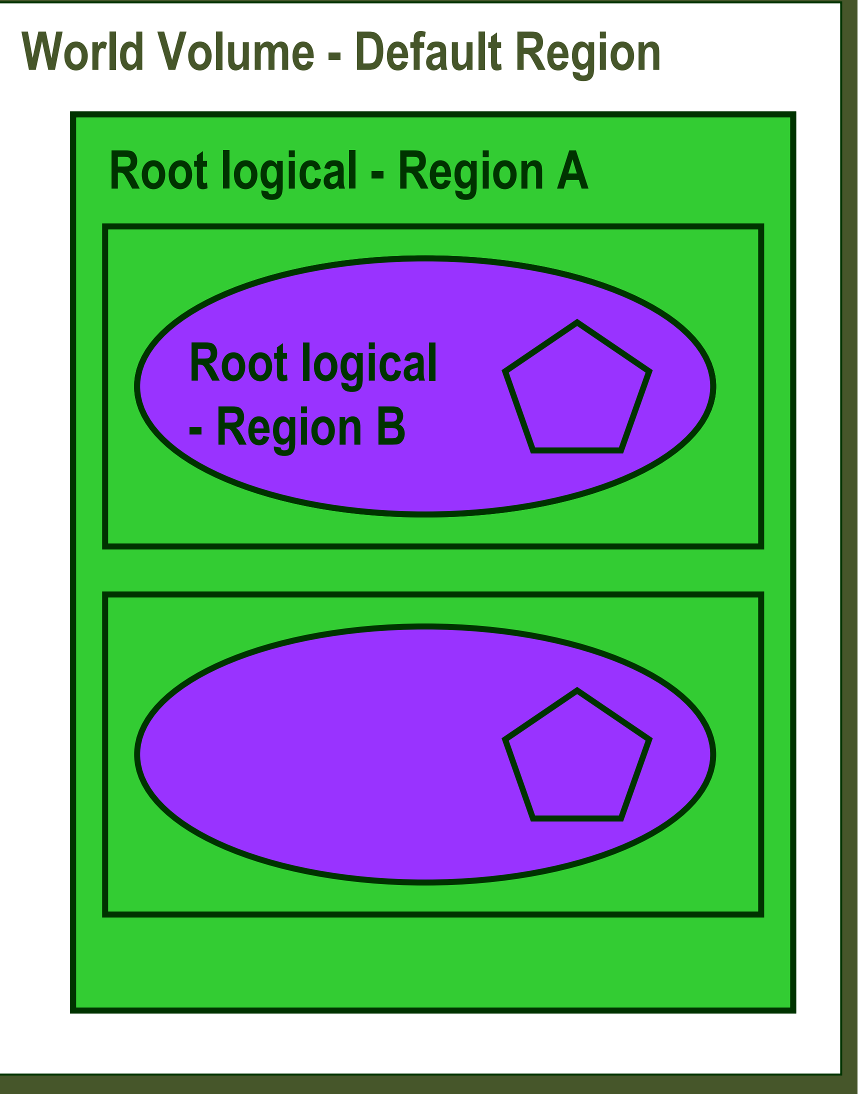
Backup
(G4Region)

Region

- A region may have its unique
  - Production thresholds (cuts)
    - If a region in the mass geometry does not have its own production thresholds, those of the default region are used (i.e., may not be those of the parent region).
  - User limits
    - You can set user limits directly to logical volume as well. If both logical volume and associated region have user limits, those of logical volume wins.
  - User region information
    - E.g. to implement a fast Boolean method to identify the nature of the region.
  - Fast simulation manager
    - Shower envelope
  - Regional user stepping action (new with version 9.0)
- Please note :
  - World logical volume is recognized as the default region. User is not allowed to define a region to the world logical volume.

Root logical volume

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



G4Region

- A region is instantiated and defined by

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

  - Region propagates down to all geometrical hierarchy until the bottom or another root logical volume.
- Production thresholds (cuts) can be assigned to a region by

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