Geometry

L. Garnier, IRISA Rennes

Credits:
I. Hrivnacova, T. Nikitina, J.Apostolakis, G.Cosmo, A. Lechner, (CERN), S.Incerti (CENBG), Tatsumi Koi (SLAC), M. Verderi (LLR) and others

Outline

- Volumes hierarchy
- Solids

Volumes Hierarchy

Describe Your Detector

- To describe your detector you have to derive your own concrete class from G4VUserDetectorConstruction abstract base class.
- Implement the virtual method Construct(), where you
  - Instantiate all necessary materials
  - Instantiate volumes of your detector geometry
- Optionally, implement the virtual method ConstructSDandField(), where you
  - Instantiate your sensitive detector classes and set them to the corresponding logical volumes
  - Instantiate magnetic (or other) field
- Optionally you can define
  - Regions for any part of your detector
  - Visualization attributes (color, visibility, etc.) of your detector elements

Before 10.x

- Instantiate your sensitive detector and magnetic (or other) field in Construct()

Creating a Volume

1) Start with its shape & size
   - Box 3 x 5 x 7 cm, sphere r = 8m

   ⇒ Solid

2) Add properties:
   - Material
   - Magnetic/electric
   - Make it sensitive

   ⇒ Logical volume

3) Place it in another volume
   - Just once
   - Repeatedly using a function

   ⇒ Physical volume

Creating a Volume

Example

```
#include "G4Box.hh"
#include "G4LogicalVolume.hh"
#include "G4PVPlacement.hh"
{
  ...
  // Define its shape & size (solid)
  G4double hxy = 10*m;
  G4double hz  = 20*m;
  G4VSolid* boxS
    = new G4Box("MyBox", hxy, hxy, hz);

  // Define its properties (logical volume)
  G4LogicalVolume* boxLV
    = new G4LogicalVolume(boxS, boxMaterial, "MyBox");

  // Define its placement (physical volume)
  G4RotationMatrix* rotation = 0;
  G4ThreeVector position;
  G4VPhysicalVolume* boxPV
    = new G4PVPlacement(
          rotation, position, boxLV, "MyBox", motherLV,
          false, 0);

  ...
}
```

Solid

Logical volume

Physical volume

G4LogicalVolume

- Contains all information of volume except position:
  - Shape and dimension (G4VSolid)
  - Material, sensitivity, visualization attributes
  - Position of daughter volumes
  - Magnetic field, User limits
  - Shower parameterisation
  - Region
- Can be shared by more physical volumes of a same type

Reminder
- *Solid: Shape & size*
- *Logical volume: + properties*
- *Physical volume: + inside a volume*

G4LogicalVolume (2)

- G4LogicalVolume constructor:

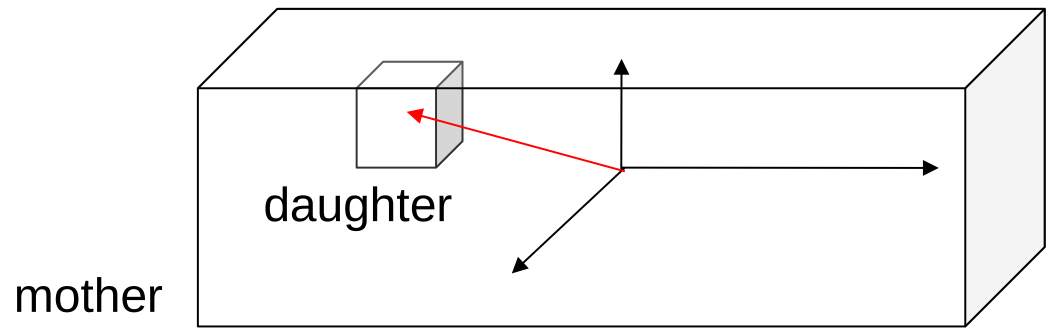
```
G4LogicalVolume(
  G4VSolid* solid,
  G4Material* material,
  const G4String& name,
  G4FieldManager* fieldManager = 0,
  G4VSensitiveDetector* sd = 0,
  G4UserLimits* userLimits = 0,
  G4bool optimise = true)
```

optional arguments

- The pointers to solid and material must NOT be null
- Once created it is automatically stored in Geant4 kernel (G4LogicalVolumeStore)
- It is not meant to act as a base class

Volumes Hierarchy

- 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 the origin of its solid coordinate system (eg. the at the center of the box)
  - Daughter volumes cannot protrude from the mother volume
  - Daughter volumes cannot overlap
- One or more volumes can be placed to mother volume



Reminder
- *Solid: Shape & size*
- *Logical volume: + properties*
- *Physical volume: + inside a volume*

Volumes Hierarchy (2)

- The logical volume of mother knows the physical volumes it contains
  - It is uniquely defined to be their mother volume
  - If the logical volume of the mother is placed more than once, all daughters appear by definition in all these physical instances of the mother
- World volume = the root volume of the hierarchy
  - The world volume must be a unique physical volume which fully contains all other volumes
  - The world defines the global coordinate system
  - The origin of the global coordinate system is at the center of the world volume
  - Should not share any surface with contained geometry

Physical Volumes

- Physical volume represents a placement of a daughter volume in its mother volume
  - It holds the information about the position of the daughter in the mother reference frame
- Physical volume types:
  - Simple placement: “placement”
  - Repeated placement: “replica”, “division”, “parameterised volume”
- A mother volume can contain either
  - More simple volume placements OR
  - One repeated volume

Reminder
- *Solid: Shape & size*
- *Logical volume: + properties*
- *Physical volume: + inside a volume*

G4PVPlacement

- Contains the information about the volume position
  - Rotation and translation (= transformation) of the volume in the mother reference frame
  - Name
  - Logical volume
  - Mother logical volume
  - Copy number (defined by the user)
- It is derived from G4VPhysicalVolume base class which serves also as a base for repeated placements

Reminder
- *Solid: Shape & size*
- *Logical volume: + properties*
- *Physical volume: + inside a volume*

G4PVPlacement (2)

- G4PVPlacement constructor:

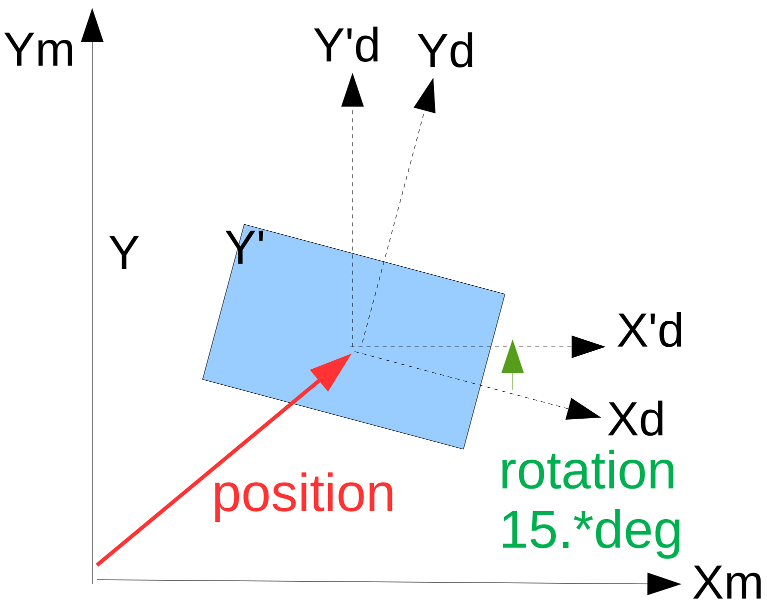
```
G4PVPlacement(
  G4RotationMatrix* rotation,        // rotation
  const G4ThreeVector& translation,  // translation
  G4LogicalVolume* currentLV,        // volume being placed
  const G4String& name,              // physical volume name
  G4LogicalVolume* motherLV,         // mother logical volume
  G4bool many,                       // not used
  G4int copyNumber,                  // position (copy) number
  G4bool surfaceCheckk = false);     // option to activate
                                     // overlap checking
```

- Three additional constructors are available
  - Besides a simple variation (using the mother physical volume instead of its logical volume) it is also possible to use G4Transform3D to represent the direct (object) rotation and translation of the solid instead of the frame

Simple Placement
Example

```
G4double posX = 100.*cm;
G4double posY =  80.*cm;
G4double posZ =   0.*cm;
G4ThreeVector position(posX, posY, posZ);

G4RotationMatrix* rotation
  = new G4RotationMatrix;
//Rotate around Z-axis
rotation->rotateZ(15.*deg);

new G4PVPlacement(
      rotation,
      position,
      boxLV,
      “MyBox”,
      motherLV,
      false,
      1);    // copyNumber
```

MyBox volume is positioned **in a frame** which is rotated by *rotation* and translated by *position* relative to the coordinate system of the mother volume



Solids

Solids Types

Solid types available in Geant4:

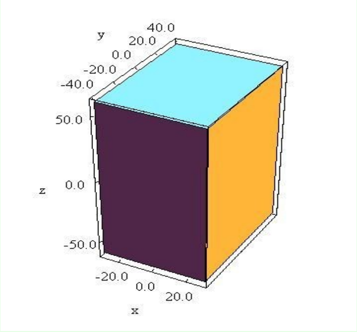
- CSG (Constructed Solid Geometry) solids
  - Box, tube (segment), cone (segment), trapezoid, ...
  - Analogous to simple GEANT3 CSG solids
- Specific solids (CSG like)
  - Polycone, polyhedra, tube with a hyperbolic profile, tessellated solid, tetrahedra, twisted tube, ...
- Boolean solids
  - Union, subtraction and intersection solid, ...
- Unified solids
  - New, alternative implementation, provided for experimental use
  - The code is part of the AIDA Unified Solids Library and is provided with Geant4 since 10.00

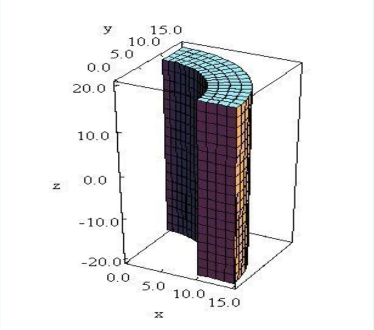
G4VSolid

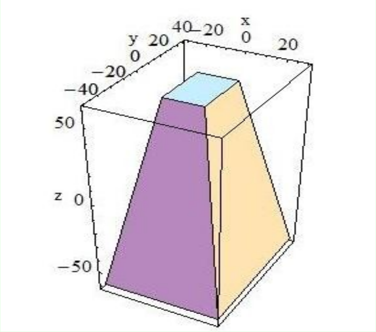
- All solids in Geant4 derive from the abstract base class G4VSolid
- It defines (but does not implement) all functions required for geometry navigations
- Once constructed, each solid is automatically registered in Geant4 kernel (G4SolidStore)

CSG: G4Box, G4Tubs, G4Trd

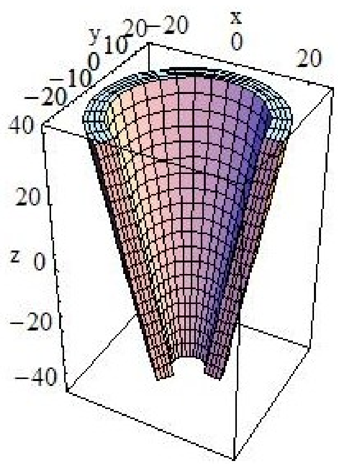
```
G4Box(const G4String& name, // name
      G4double hx,    // x half size
      G4double hy,    // y half size
      G4double hz);   // z half size

G4Tubs(const G4String& name, // name
      G4double rmin,  // inner radius
      G4double rmax,  // outer radius
      G4double hz,    // z-half length
      G4double sphi,  // starting Phi
      G4double dphi); // segment angle

G4Trd(const G4String& name,  // name
      G4double dx1,   // x half size at -dz
      G4double dx2,   // x half size at +dz
      G4double dy1,   // y half size at -dz
      G4double dy2,   // y half size at +dz
      G4double hz);.  // z half size
```



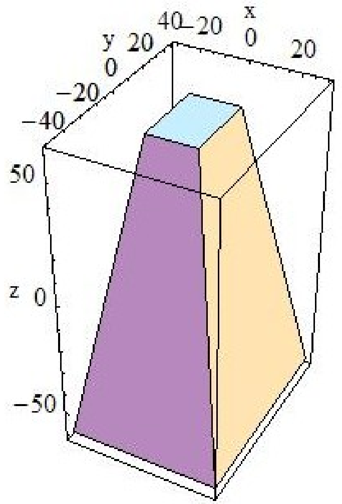




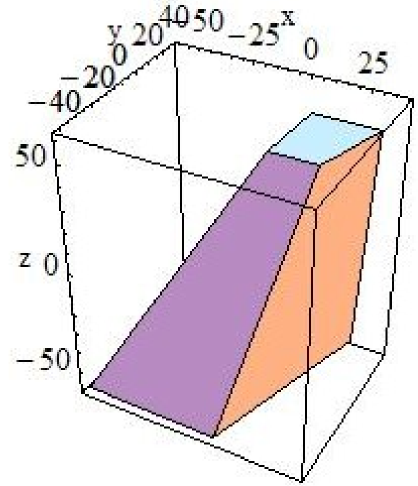
Other CSG Solids



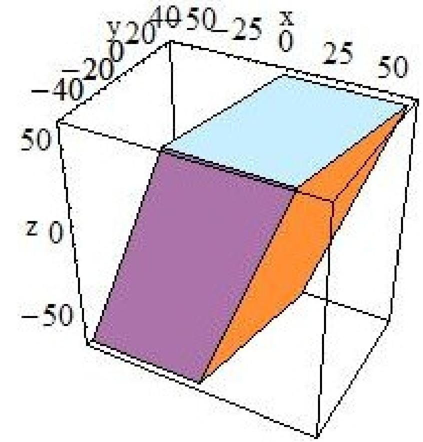
G4Cons



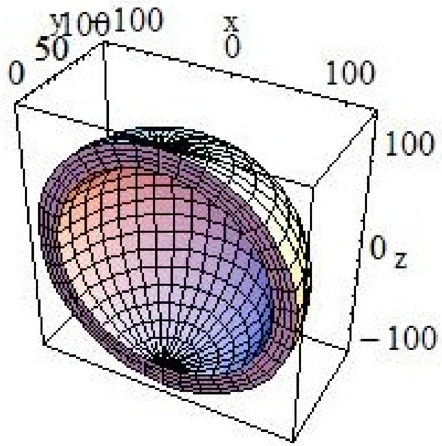
G4Trd



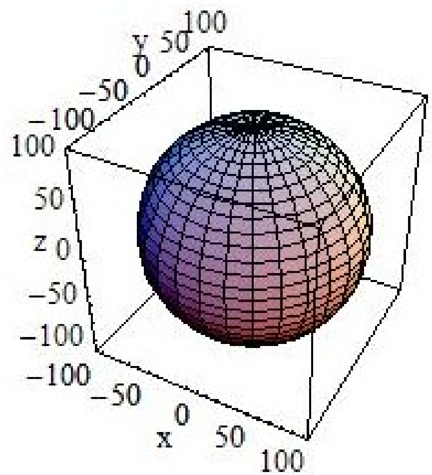
G4Trap



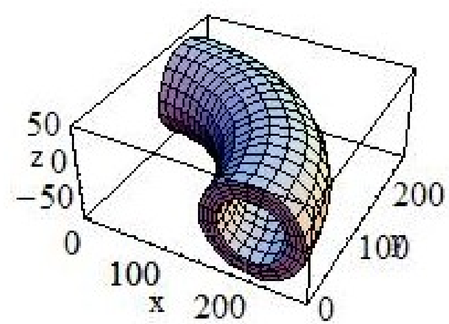
G4Para
(parallelepiped)



G4Sphere



G4Orb (full solid sphere)



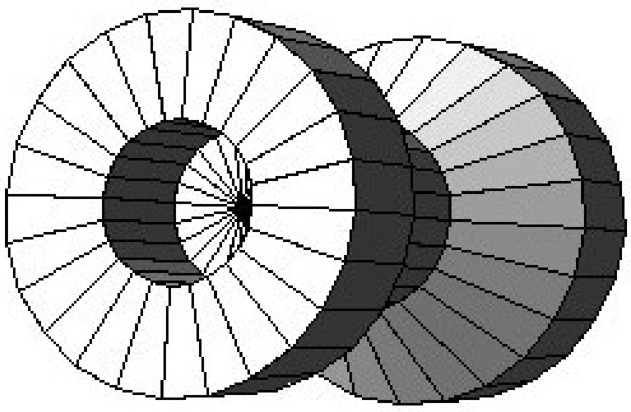
G4Torus

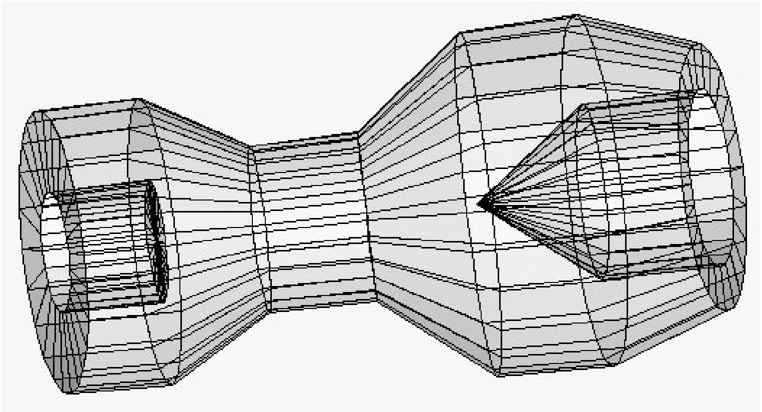
See Section 4.1.2 of
Geant4 Application
Developers Guide
for all available shapes.

Specific CSG Solids: G4Polycone

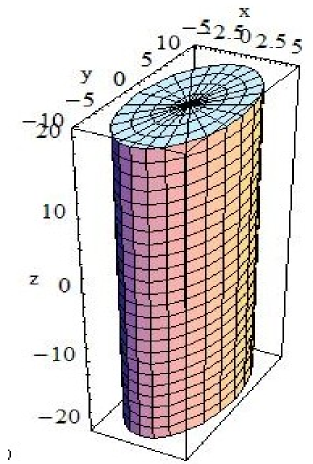
```
G4Polycone(const G4String& name, // name
      G4double sphi,             // x half size
      G4double dphi,             // y half size
      G4int numRZ,               // number of corners in RZ space
      const G4double r[],        // r coordinate of the corners
      const G4double z[]);       // z coordinate of the corners
```

- Additional constructor using z planes

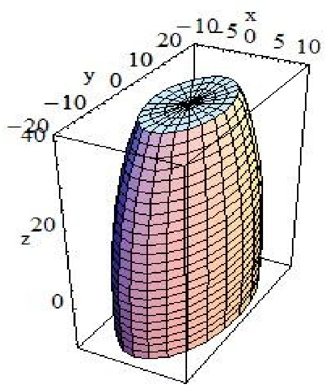




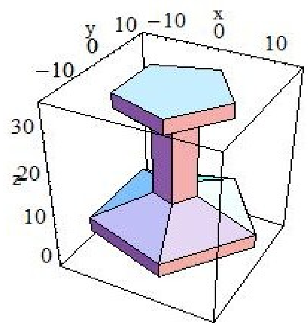
Other Specific CSG Solids



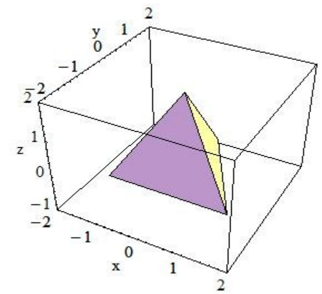
G4EllipticalTube



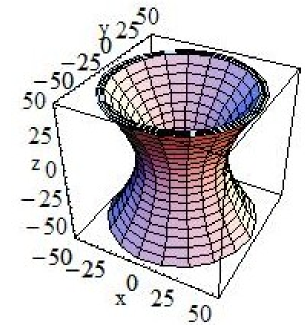
G4Ellipsoid



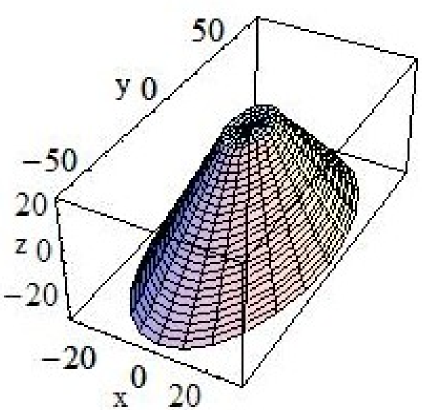
G4Polyhedra



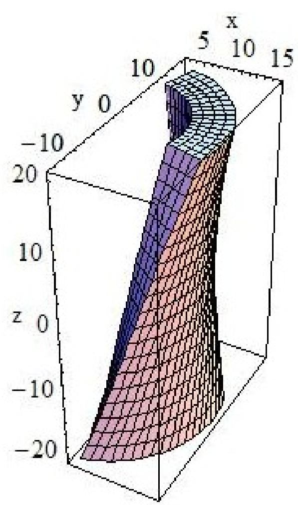
G4Tet
(tetrahedra)



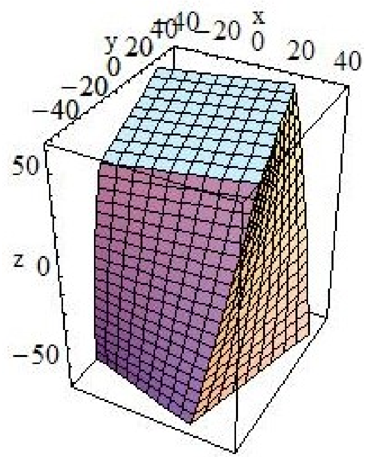
G4Hype



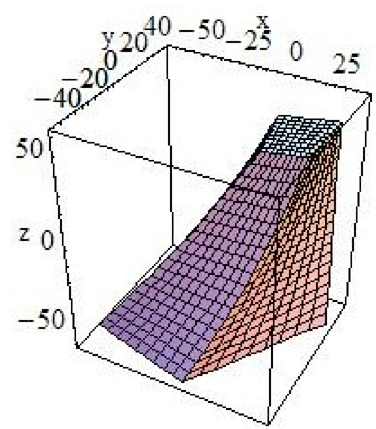
G4EllipticalCone



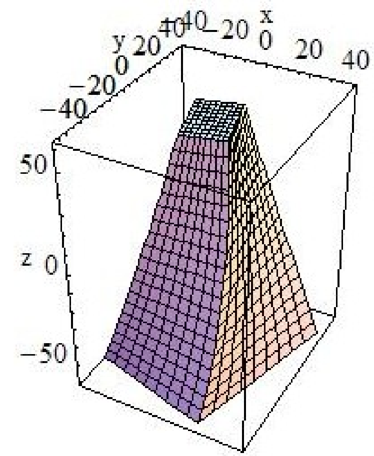
G4TwistedTubs



G4TwistedBox



G4TwistedTrap

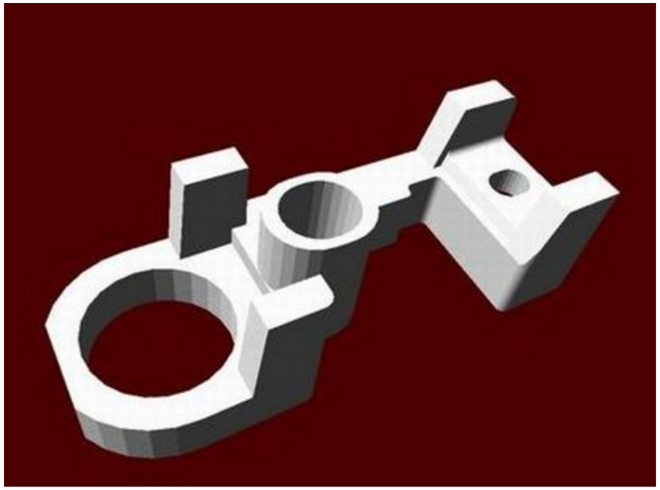


G4TwistedTrd

See Section 4.1.2 of Geant4 Application Developers Guidefor all available shapes.

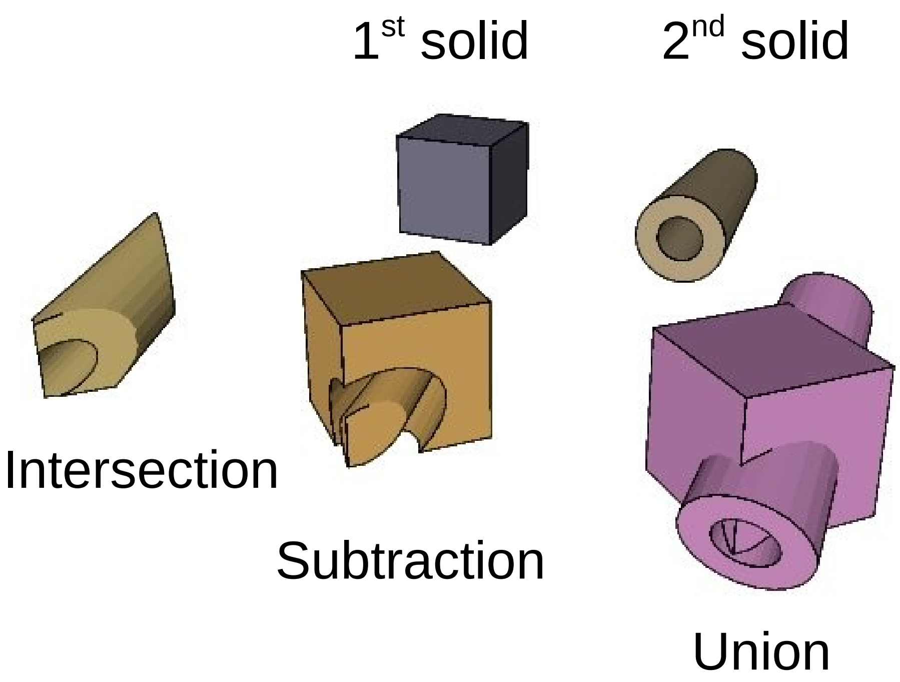
Tessellated Solids

- G4TessellatedSolid: generic solid defined by a number of facets
  - Facets can be triangular or quadrangular
- 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



Boolean Solids

- Solids can be combined using boolean operations:
  - G4UnionSolid, G4SubtractionSolid, G4IntersectionSolid
  - Requires: two solids, a Boolean operation, and a transformation (optional) for the 2nd solid (displacement)
  - 2nd solid is positioned relatively to the coordinate system of the 1st solid



- Solids 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 solids

Boolean Solids

Example

```
// Create solids
G4VSolid* solid1
  = new G4Box("boxS", 50.*cm, 50.*cm, 50.*cm);
G4VSolid* solid2
  = new G4Cons("consS", 10.*cm, 30.*cm, 20.*cm, 40.*cm, 100.*cm,
                0., 360.*deg) ;

        const G4double z[]);  // z coordinate of the corners
// solid2 displacement
G4RotationMatrix* rot2 = new G4RotationMatrix();
rot2->rotateY( 45.*deg);
rot2->rotateX(-30.*deg);
G4ThreeVector tr2(20.*cm, 0., 0.);

// Intersection
G4VSolid* intersectionS
  = new G4IntersectionSolid("intersectionS", solid1, solid2, rot2, tr2);

// Subtraction
G4VSolid* subtractionS
  = new G4SubtractionSolid("subtractionS", solid1, solid2, rot2, tr2);

// Union
G4VSolid* unionS
  = new G4UnionSolid("unionS", solid1, solid2, rot2, tr2);
```

Summary

- Basic “bricks” to define geometry
  - Solid, Logical volume, Physical volume
- Volumes Hierarchy
  - Mother and daughter volumes, simple placements
- Available solids in Geant4
  - CSG (box, tube, etc.), Specific (polygon, polyhedra, ..), Boolean