Geometry

Shigeyuki Tajima

Duke/NCCU and TUNL

Slides courtesy of SLAC GEANT4 Team

Special thanks to Makoto Asai

Outline

- Introduction
- G4VUserDetectorConstruction class
- Solid and Shape
- Logical Volume
- Physical Volume
- EM Field

Keywords

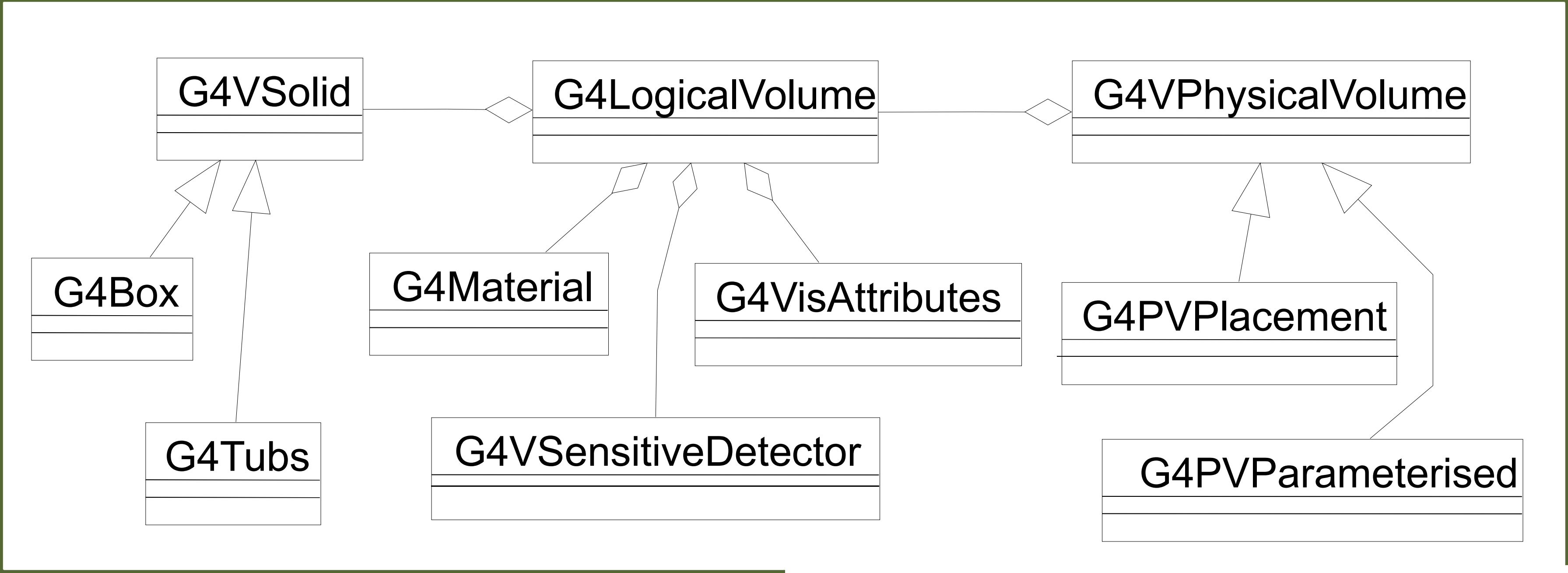
- World Volume (contains ALL the volumes), such as LAB
- Logical Volume (contains all information of volume except for position and rotation)
  - ➡ shape, dimension, material, magnetic field, ...
- Daughter Volume is contained in a Mother Volume
  - ➡ Mother/Daughter volume refers to a Logical volume
- Physical Volume (It's one positioned volume placed inside a mother volume)
- Replicated Volume, Parameterised volume ...
- solid, boolean solid, CSG (constructed solid geometry) solids

Introduction

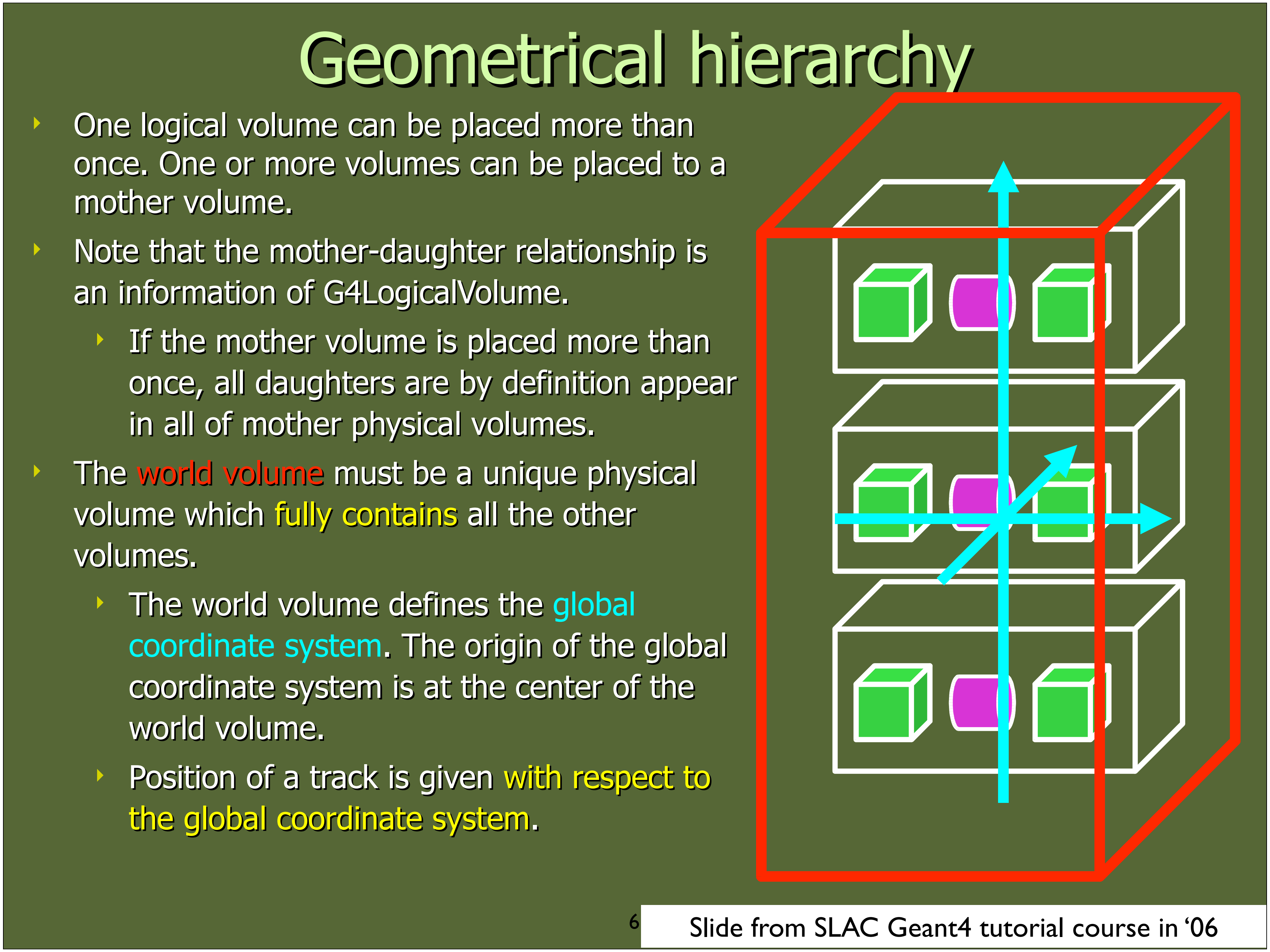
Slide from SLAC Geant4 tutorial course in ‘06

Detector geometry

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

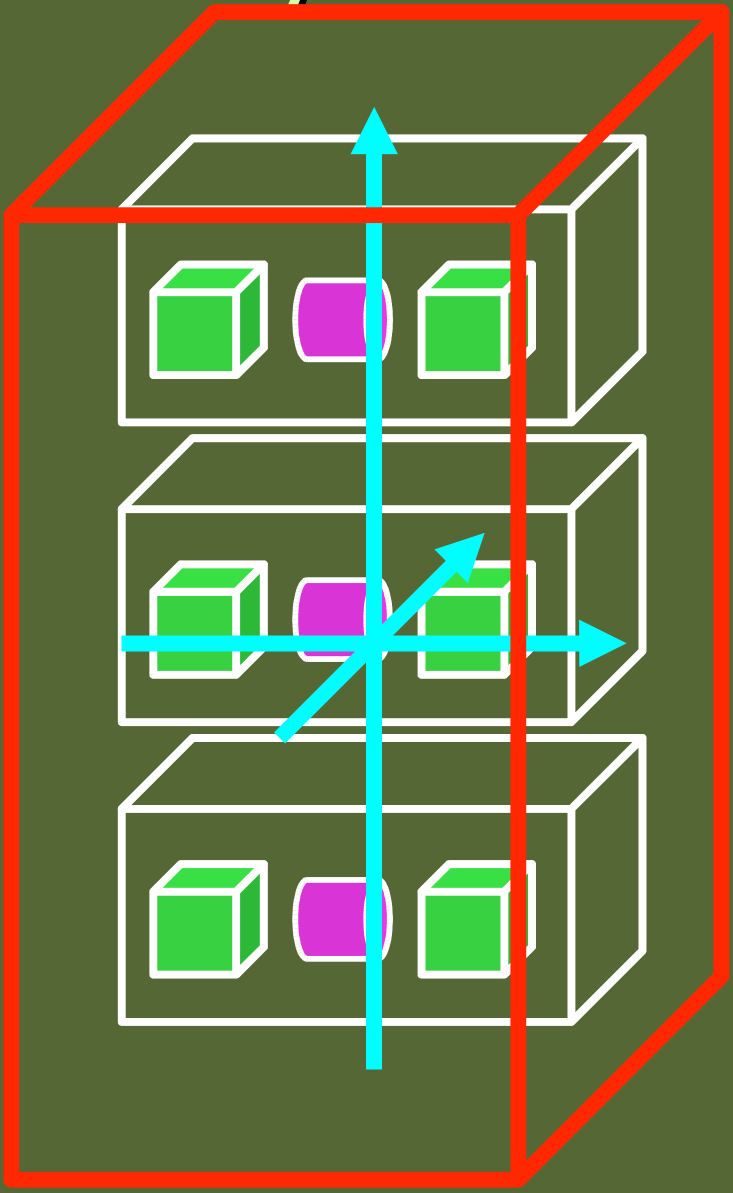


Slide from SLAC Geant4 tutorial course in ‘06



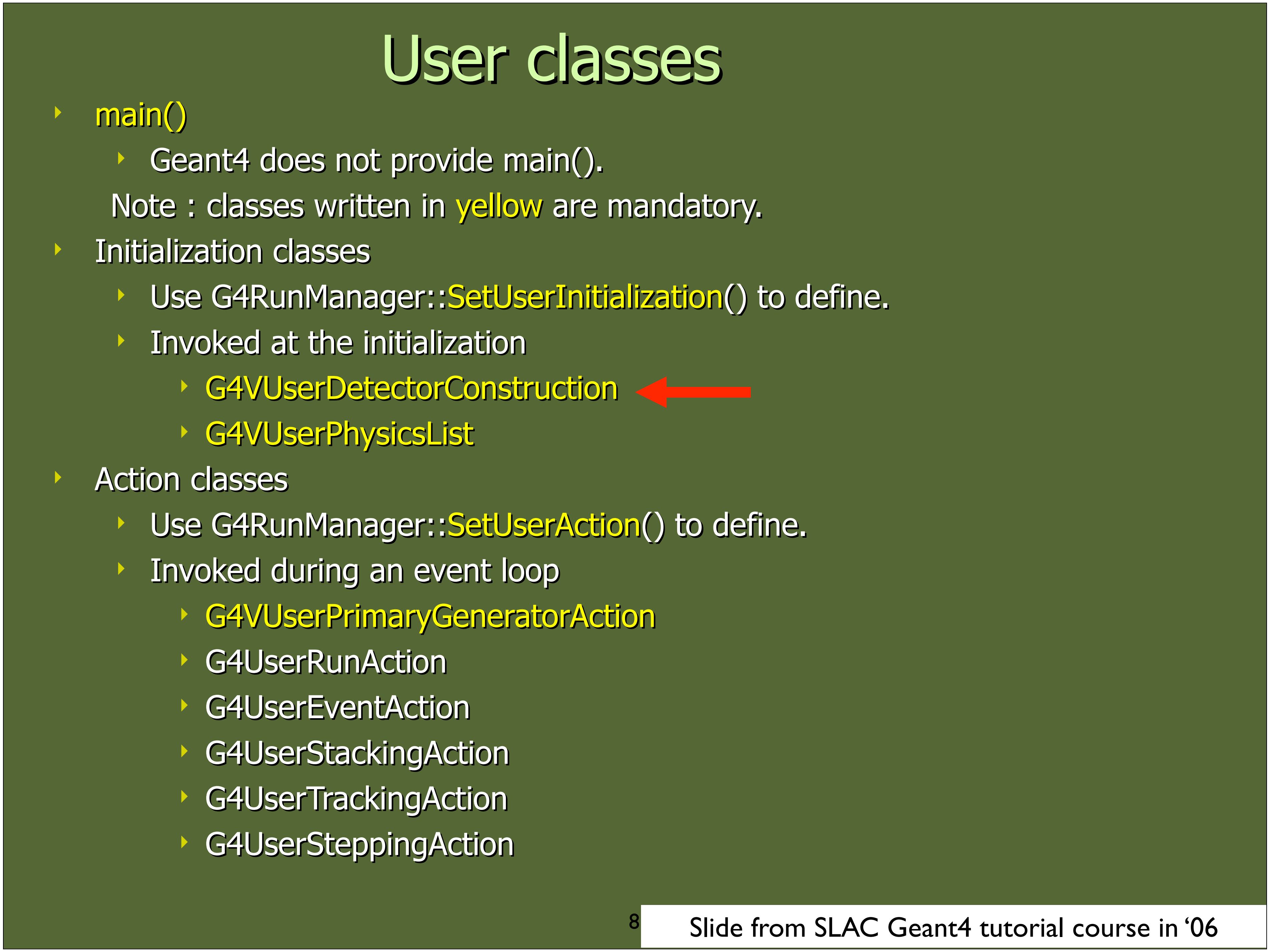
Geometrical hierarchy

- 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.



Slide from SLAC Geant4 tutorial course in ‘06

G4VUserDetectorConstruction



User classes

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

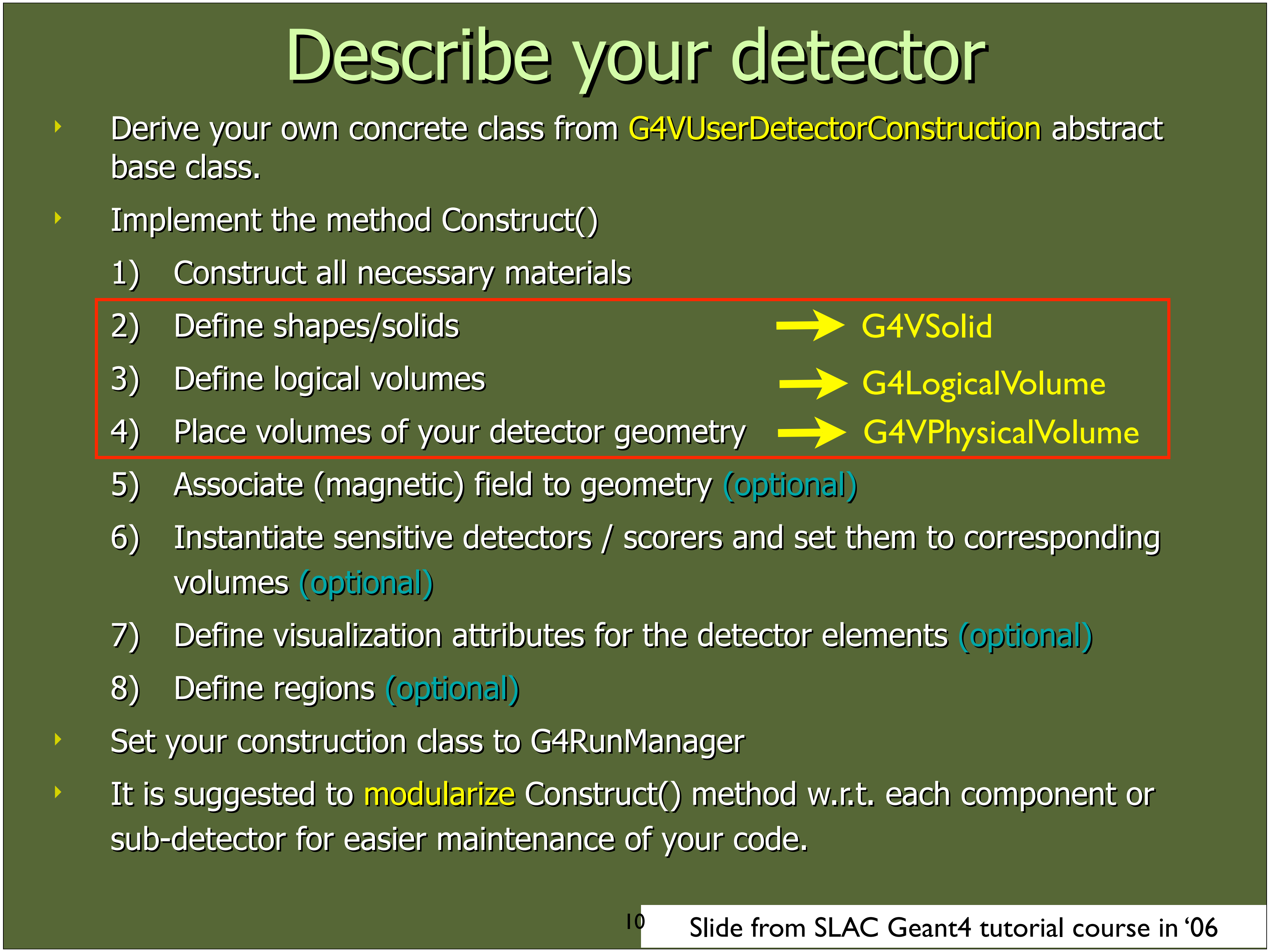
  Note : classes written in yellow are mandatory.
- Initialization classes
  - Use G4RunManager::SetUserInitialization() to define.
  - Invoked at the initialization
    - G4VUserDetectorConstruction
    - G4VUserPhysicsList
- Action classes
  - Use G4RunManager::SetUserAction() to define.
  - Invoked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

Slide from SLAC Geant4 tutorial course in ‘06

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
```

Slide from SLAC Geant4 tutorial course in ‘06



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 → G4VSolid
  3) Define logical volumes → G4LogicalVolume
  4) Place volumes of your detector geometry → G4VPhysicalVolume
  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.

Slide from SLAC Geant4 tutorial course in '06

Define detector geometry

- Basic strategy

1. `G4VSolid* pBoxSolid =`
```
     new G4Box(“aBoxSolid”, 1.*m, 2.*m, 3.*m);
```
2. `G4LogicalVolume* pBoxLog =`
```
     new G4LogicalVolume( pBoxSolid, pBoxMaterial,
                          “aBoxLog”, 0, 0, 0);
```
3. `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.

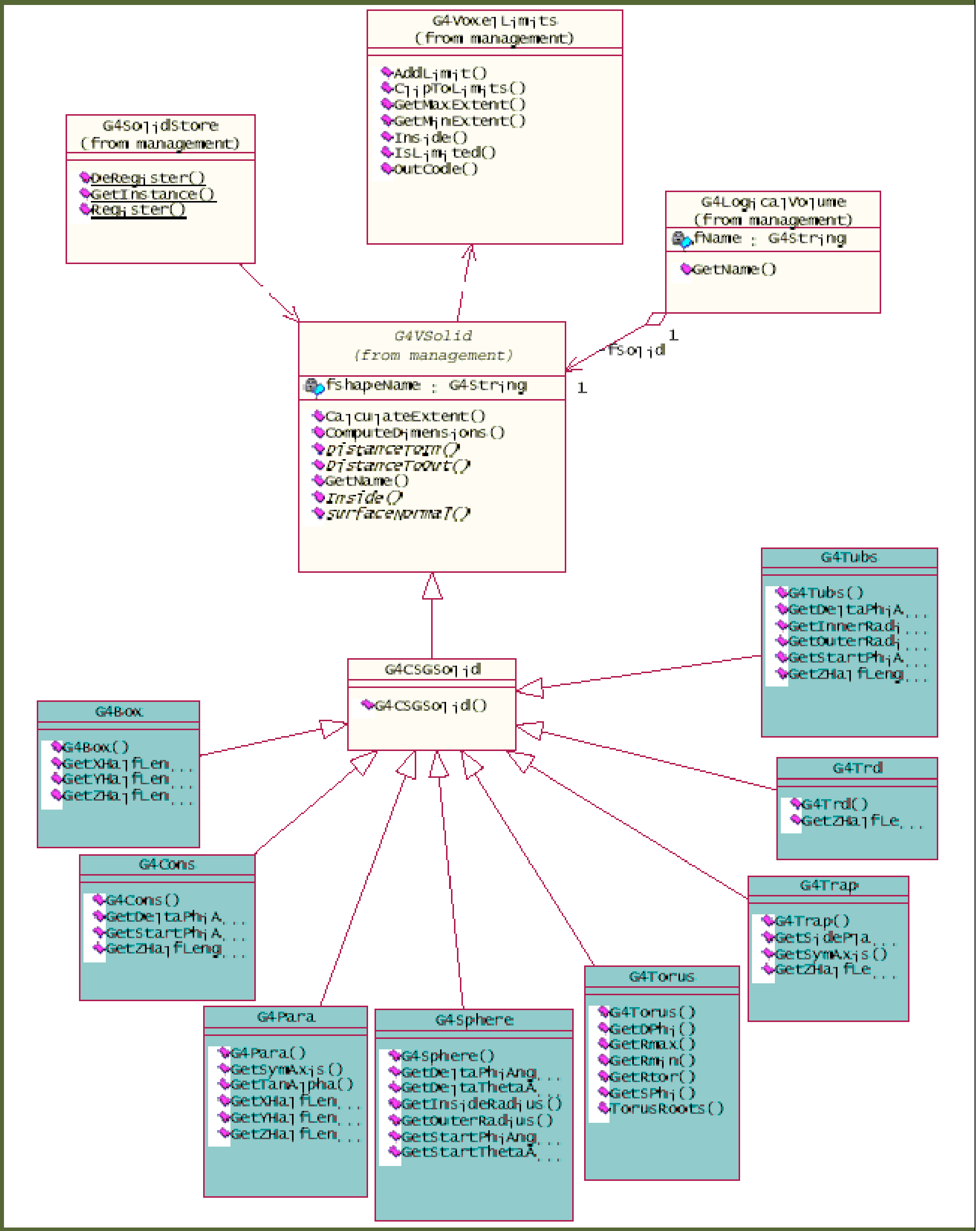
Daughter volume cannot protrude from mother volume.

Slide from SLAC Geant4 tutorial course in ‘06

Solid and shape

G4VSolid

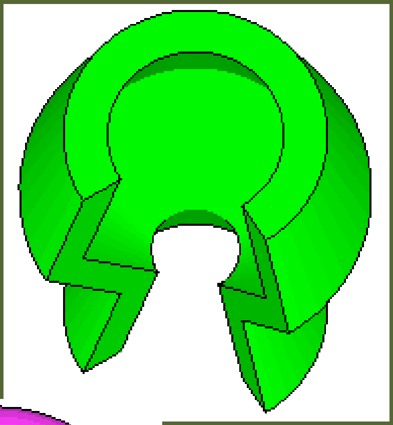
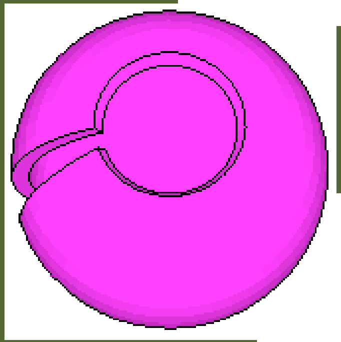
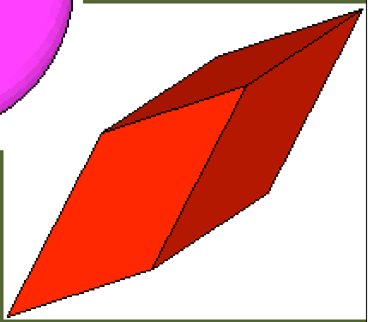
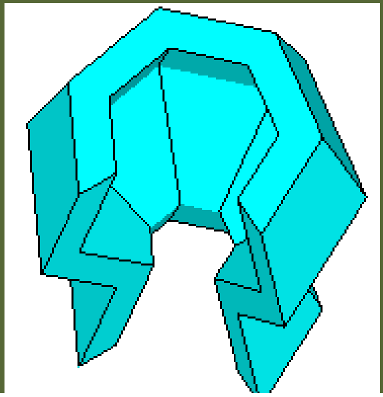
- Abstract class. All solids in Geant4 are derived from it
- It defines but does not implement all functions required to:
  - compute distances between the shape and a given point
  - check whether a point is inside the shape
  - compute the extent of the shape
  - compute the surface normal to the shape at a given point
- User can create his/her own solid class



Slide from SLAC Geant4 tutorial course in ‘06

Solids

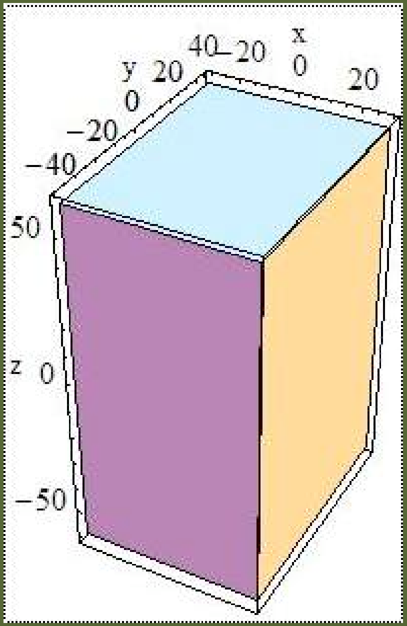
- Solids defined in Geant4:
  - CSG (Constructed Solid Geometry) solids
    - G4Box, G4Tubs, G4Cons, G4Trd, …
    - Analogous to simple GEANT3 CSG solids
  - Specific solids (CSG like)
    - G4Polycone, G4Polyhedra, G4Hype, …
  - BREP (Boundary REPresented) solids
    - G4BREPSolidPolycone, G4BSplineSurface, …
    - Any order surface
  - Boolean solids
    - G4UnionSolid, G4SubtractionSolid, …

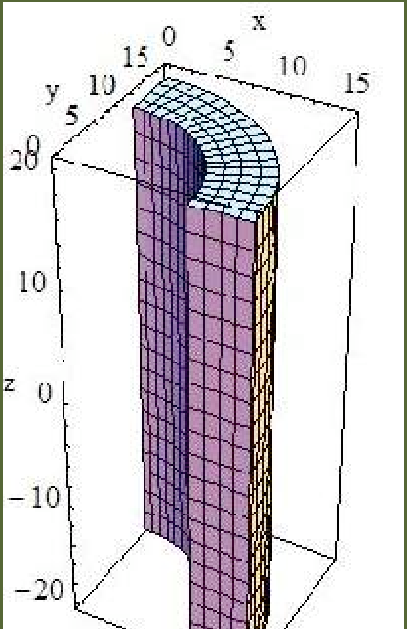
Slide from SLAC Geant4 tutorial course in ‘06

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
```



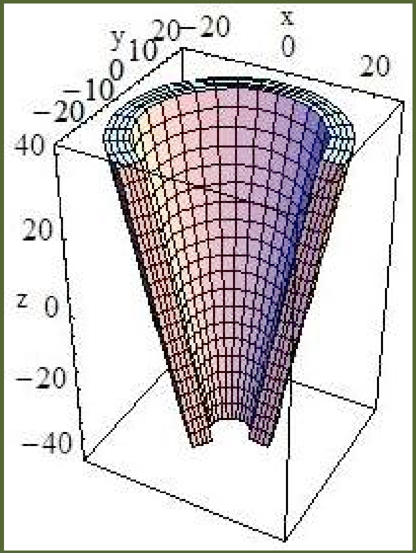
```
G4Tubs(const G4String &pname,  // name
             G4double  pRmin,  // inner radius
             G4double  pRmax,  // outer radius
             G4double  pDz,    // Z half length
             G4double  pSphi,  // starting Phi
             G4double  pDphi); // segment angle
```

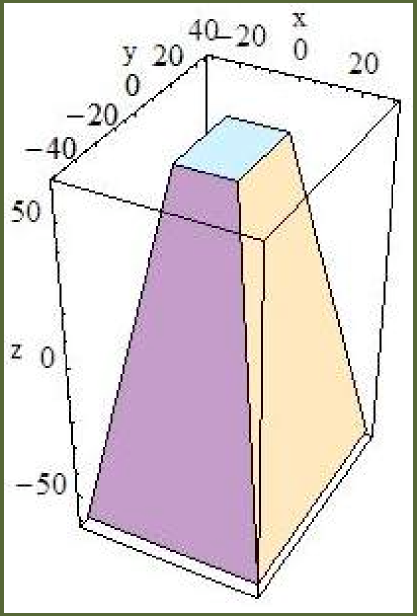


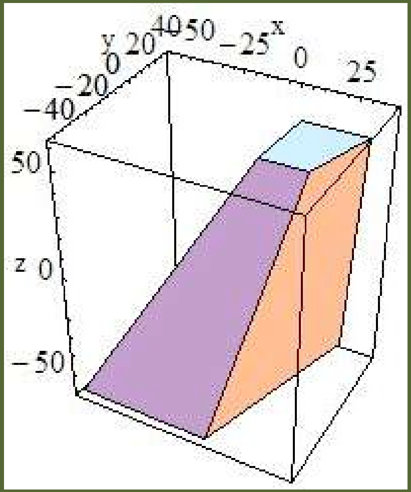
(To make a solid cylinder, set pRmin=0.*cm, pSphi=0.*deg, pDphi=360.*deg)

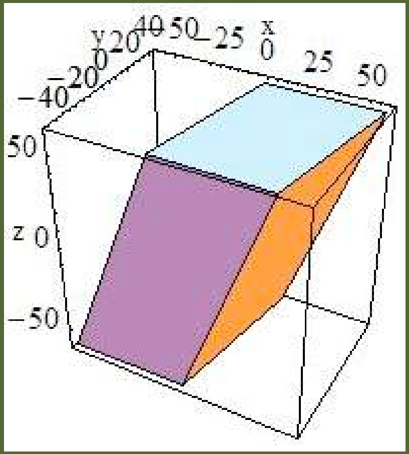
Slide from SLAC Geant4 tutorial course in ‘06

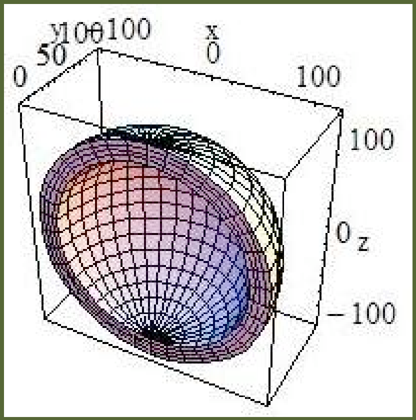
Other CSG solids

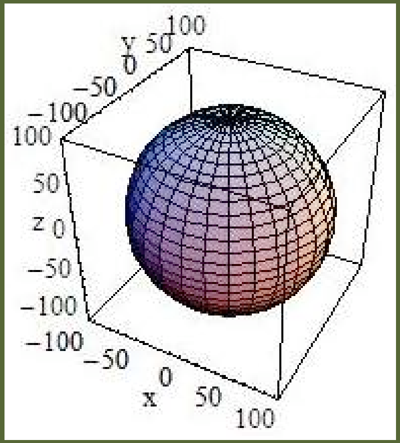

G4Cons

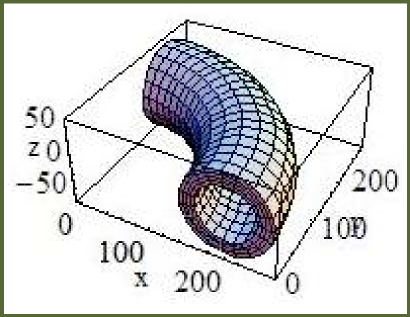

G4Trd


G4Trap


G4Para (parallelepiped)


G4Sphere


G4Orb (full solid sphere)

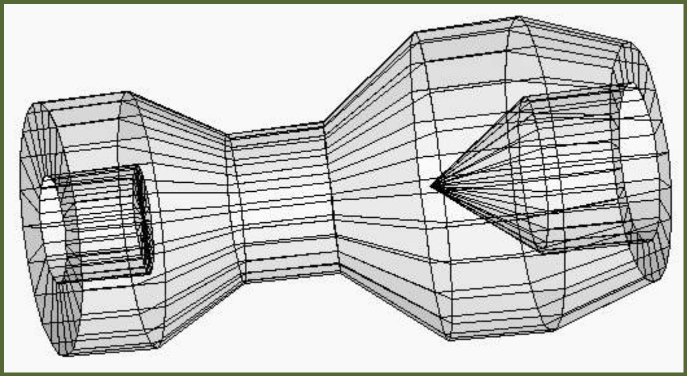

G4Torus

Consult to Section 4.1.2 of Geant4 Application for all available shapes.

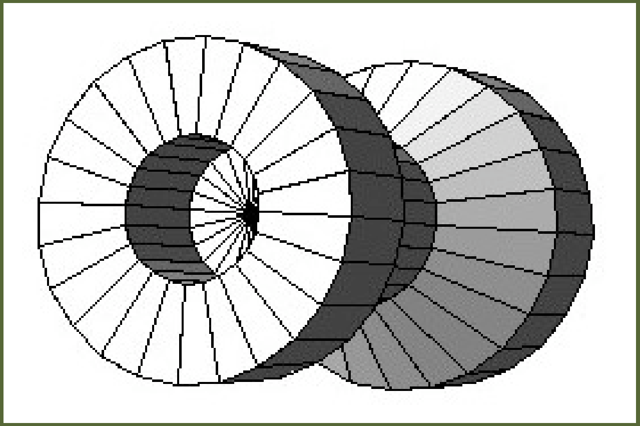
Slide from SLAC Geant4 tutorial course in '06

Specific CSG Solids: G4Polycone

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

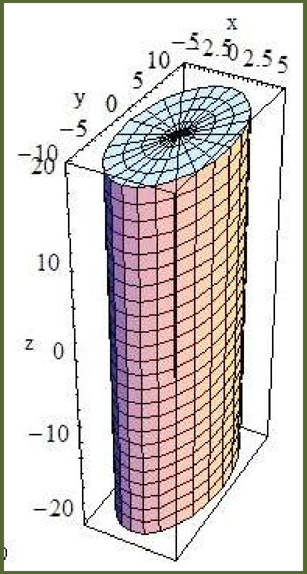


- **numRZ** - numbers of corners in the **r,z** space
- **r, z** - coordinates of corners

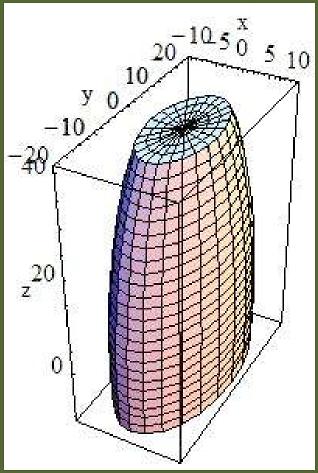


Slide from SLAC Geant4 tutorial course in '06

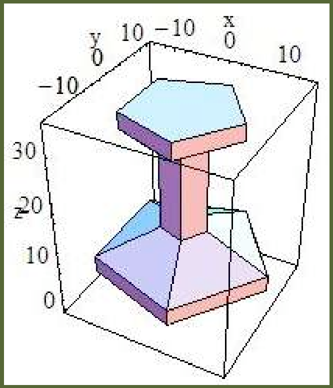
Other Specific CSG solids



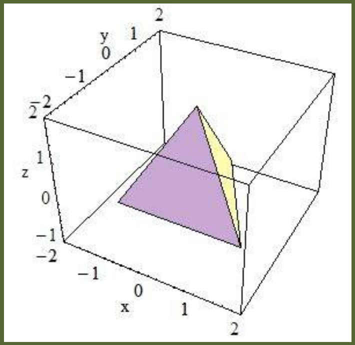
G4EllipticalTube



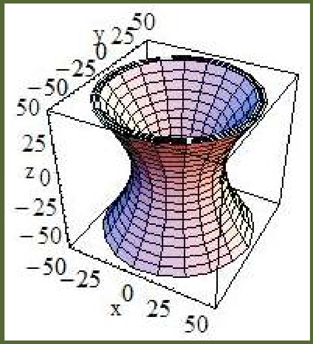
G4Ellipsoid



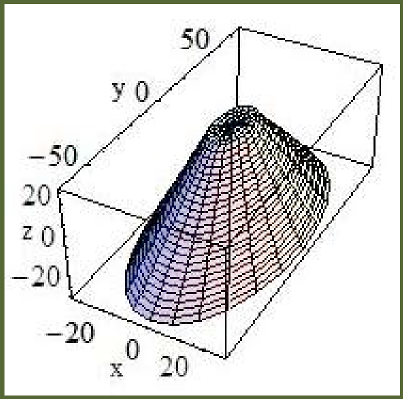
G4Polyhedra



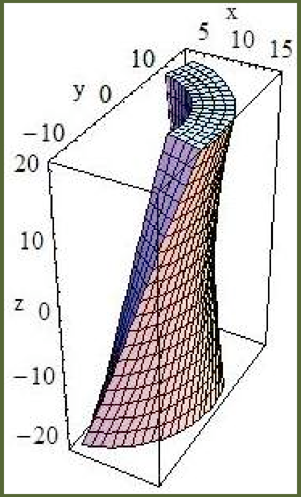
G4Tet
(tetrahedra)



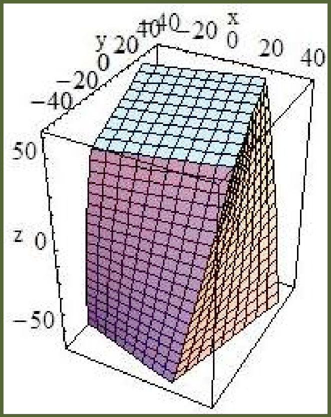
G4Hype



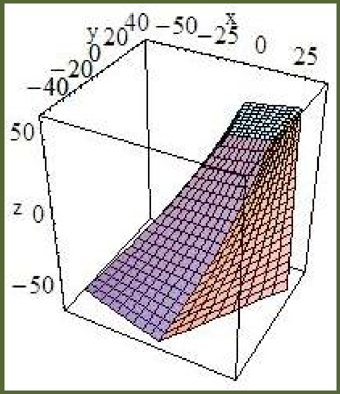
G4EllipticalCone



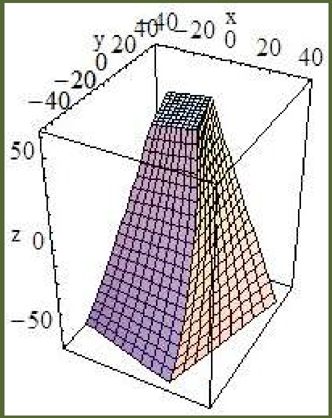
G4TwistedTubs



G4TwistedBox



G4TwistedTrap



G4TwistedTrd

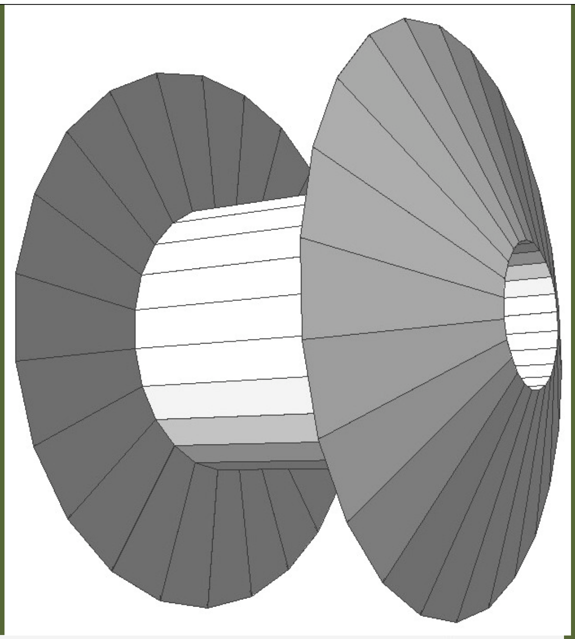
Consult to
Section 4.1.2 of Geant4 Application Deve
for all available shapes.

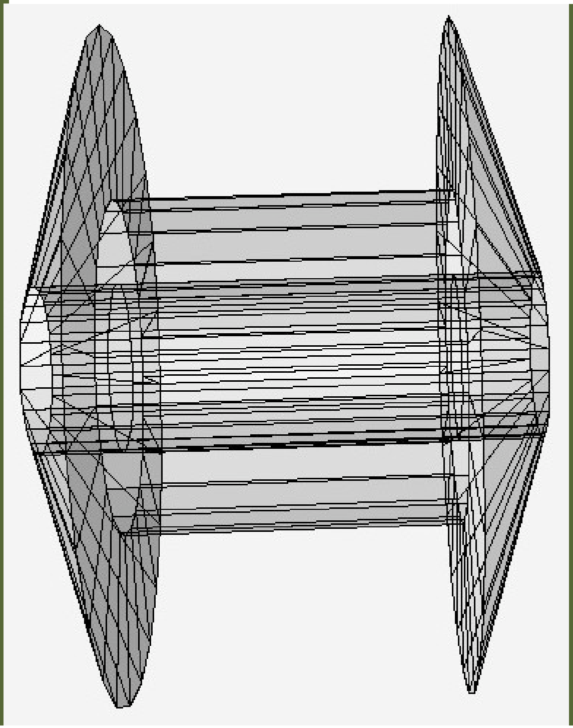
Slide from SLAC Geant4 tutorial course in '06

BREP Solids

- BREP = Boundary REPresented Solid
- Listing all its surfaces specifies a solid
  - e.g. 6 planes for a cube
- Surfaces can be
  - planar, 2nd 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

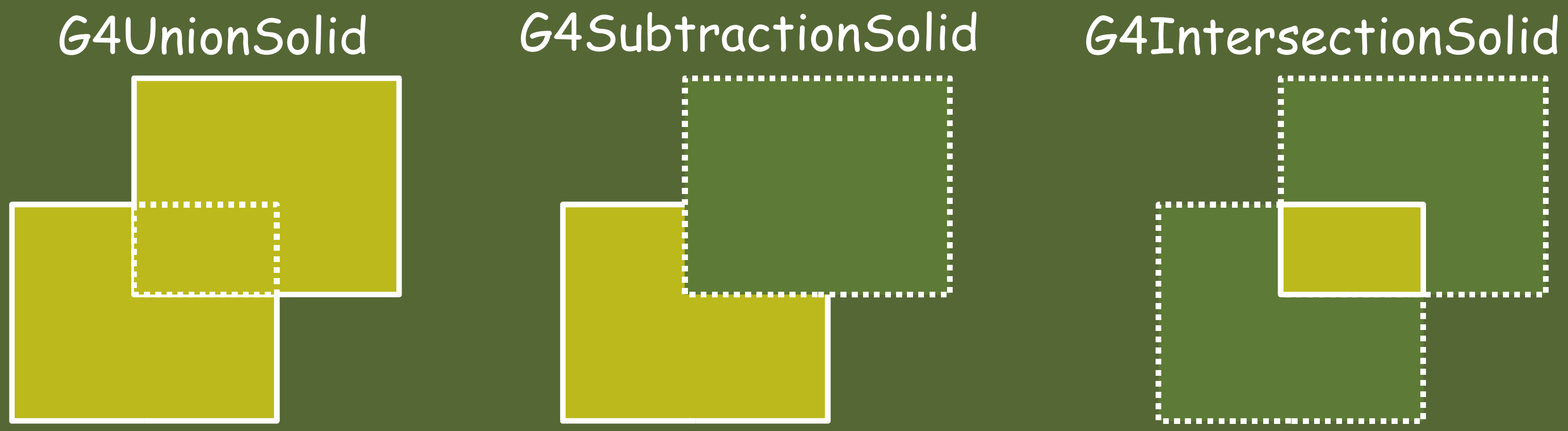




Slide from SLAC Geant4 tutorial course in '06

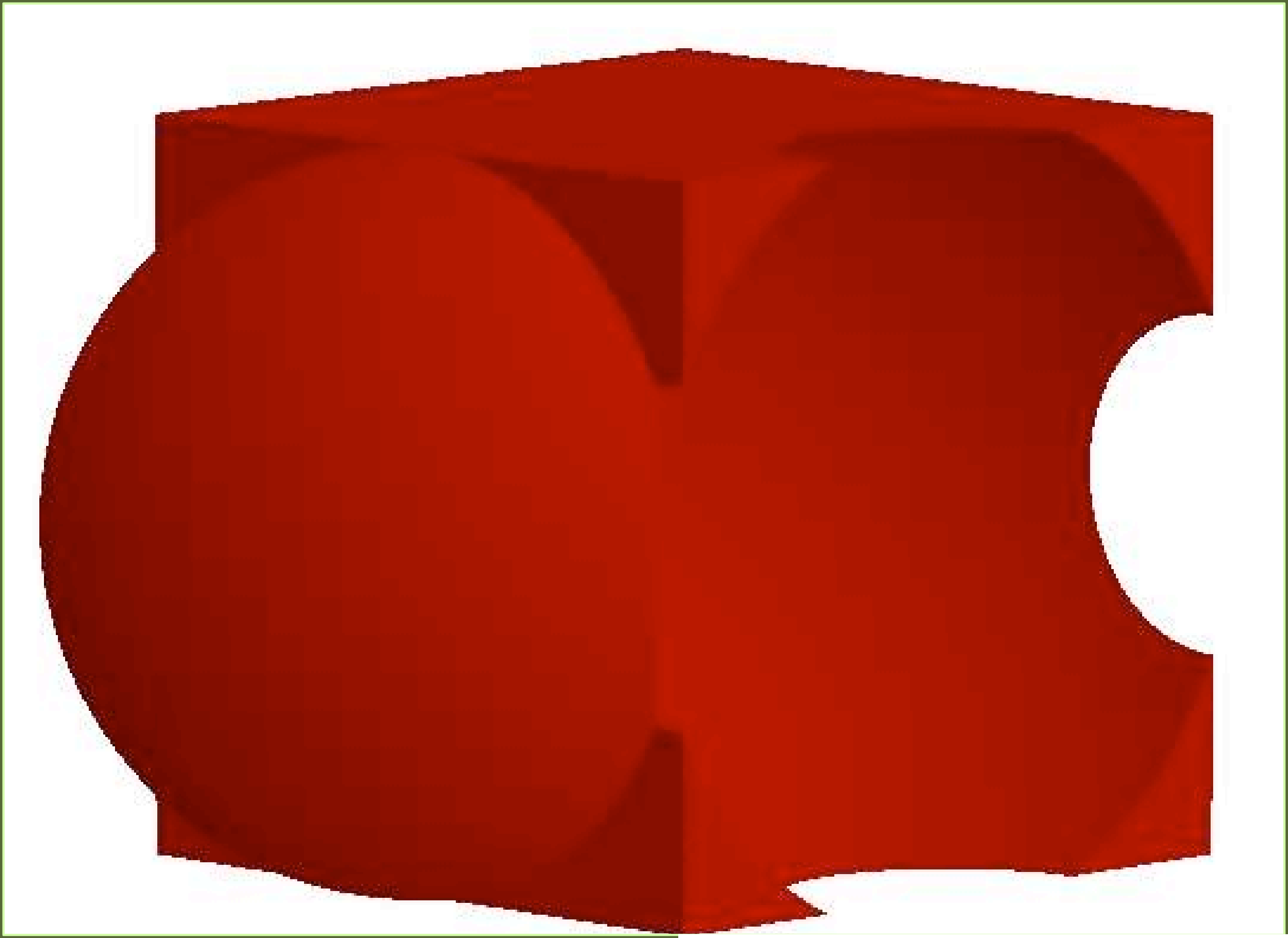
Boolean Solids

- 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



Slide from SLAC Geant4 tutorial course in '06

Boolean solid



Slide from SLAC Geant4 tutorial course in '06

Boolean Solids - example

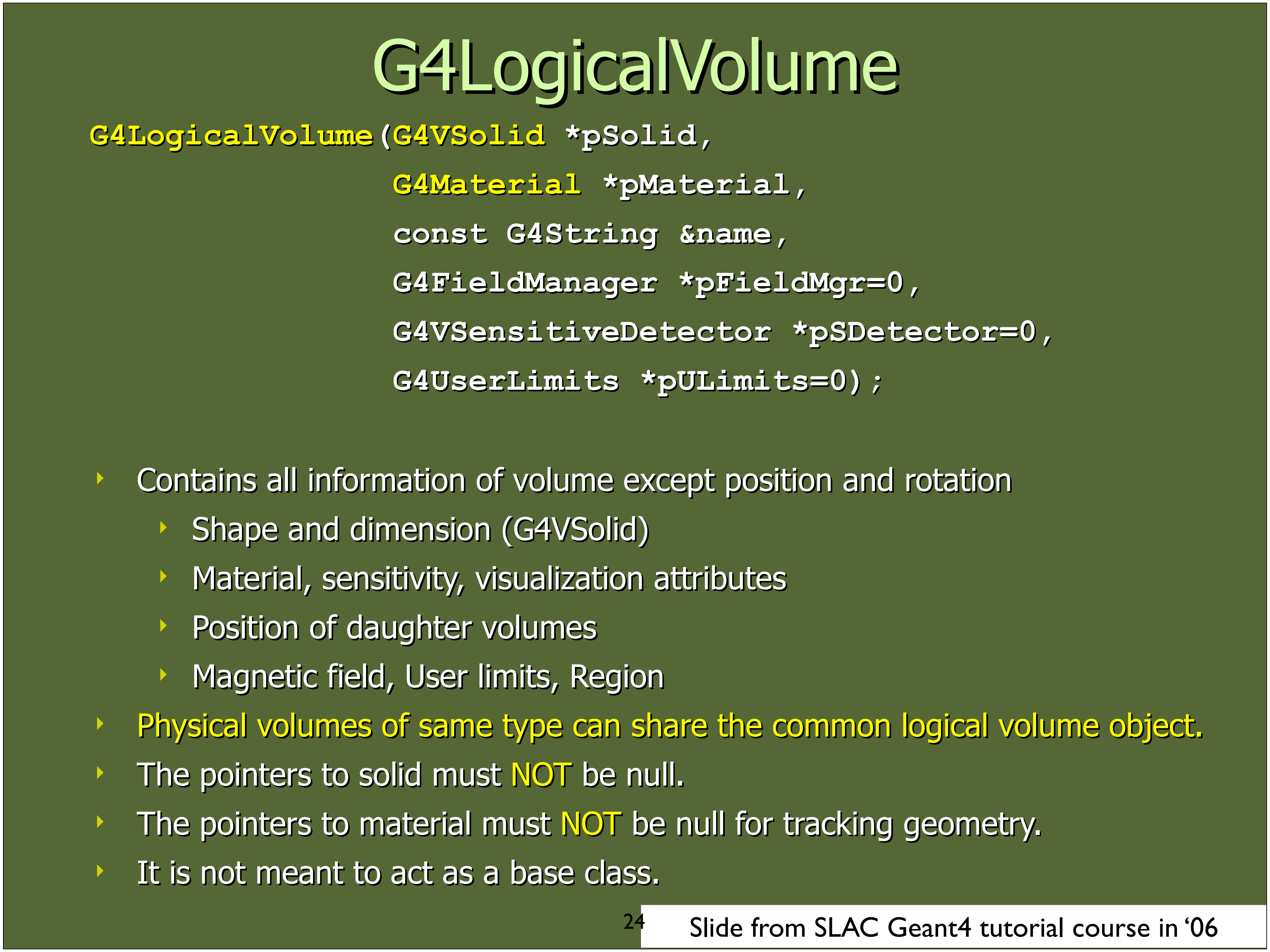
```
G4VSolid* box = new G4Box(“Box",50*cm,60*cm,40*cm);
G4VSolid* cylinder
 = new G4Tubs(“Cylinder”,0.,50.*cm,50.*cm,0.,2*M_PI*rad);
G4VSolid* union
 = new G4UnionSolid("Box+Cylinder", box, cylinder);
G4VSolid* subtract
 = new G4SubtractionSolid("Box-Cylinder", box, cylinder,
       0, G4ThreeVector(30.*cm,0.,0.));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

Slide from SLAC Geant4 tutorial course in ‘06

G4LogicalVolume

Slide from SLAC Geant4 tutorial course in ‘06



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.

Slide from SLAC Geant4 tutorial course in '06

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);
```

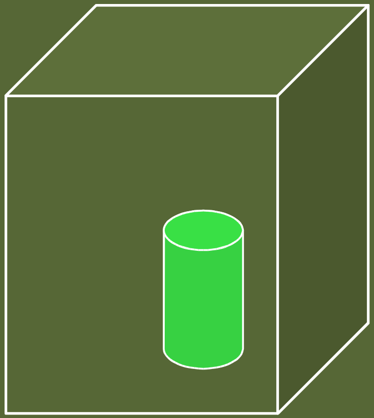
  - The computation may require a considerable amount of time, depending on the complexity of the geometry.
  - The return value is cached and reused until ***forced***=true.
  - Daughter volumes will be neglected if ***propagate***=false.

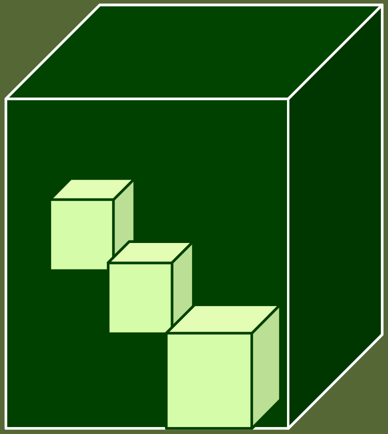
Slide from SLAC Geant4 tutorial course in ‘06

Physical volume

Physical Volumes

- 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
    - repetition w.r.t. 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

Physical Volume

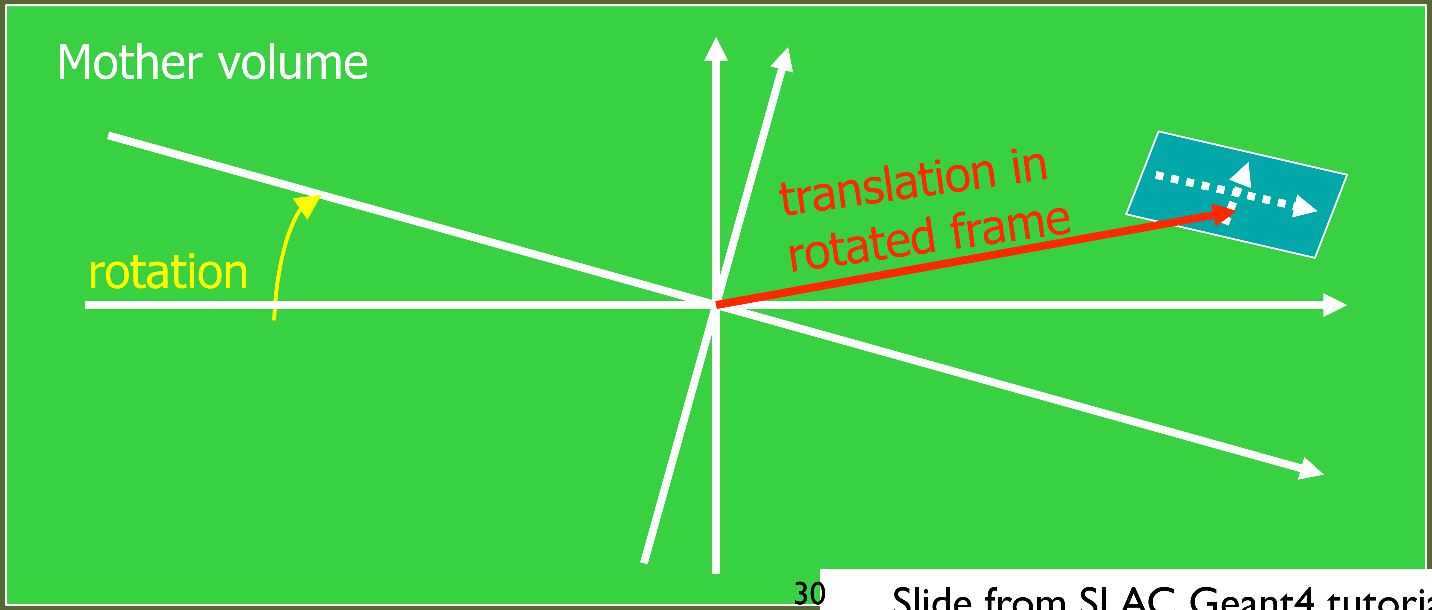
- G4PVPlacement (One Placement --> One placement volume)
  - Volume positioned in the mother volume
- G4PVReplica (One Replica --> Many repeated volumes)
  - Daughters of the same shape aligned along one axis
  - Daughters fill the mother volume without any gap in between
- G4PVDivision (One Division --> Many repeated volumes)
  - Similar to G4PVReplica but G4PVDivision allows gap (offset) along the axis of division.
- G4PVParameterised (One Parameterised --> Many repeated volumes)
  - Shape, dimensions, material, sensitivity, vis attributes, position, and rotation of a repeated volume are parameterised as a function of the copy number (Implement G4VPVParameterisation class)

G4PVPlacement

G4PVPlacement

```
G4PVPlacement(G4RotationMatrix* pRot,   // rotation of mother frame
              const G4ThreeVector &tlate, // position in rotated frame
              G4LogicalVolume *pDaughterLogical,
              const G4String &pName,
              G4LogicalVolume *pMotherLogical,
              G4bool pMany, // ‘true’ is not supported yet…
              G4int pCopyNo, // unique arbitrary integer
              G4bool pSurfChk=false); // optional boundary check
```

- Single volume positioned relatively to the mother volume.

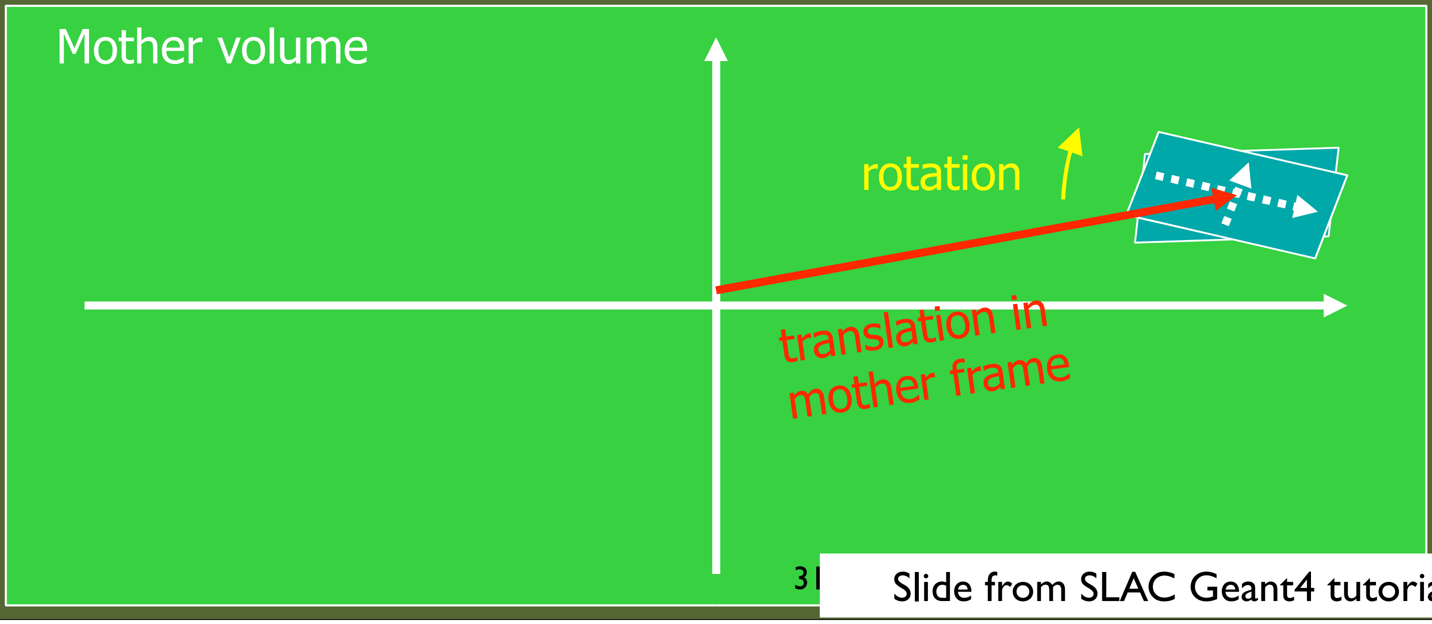


Slide from SLAC Geant4 tutorial course in ‘06

Alternative G4PVPlacement

```
G4PVPlacement(
  G4Transform3D(G4RotationMatrix &pRot, // rotation of daughter frame
             const G4ThreeVector &tlate), // position in mother frame
  G4LogicalVolume *pDaughterLogical,
  const G4String &pName,
  G4LogicalVolume *pMotherLogical,
  G4bool pMany, // 'true' is not supported yet…
  G4int pCopyNo, // unique arbitrary integer
  G4bool pSurfChk=false); // optional boundary check
```

- Single volume positioned relatively to the mother volume.



Slide from SLAC Geant4 tutorial course in '06

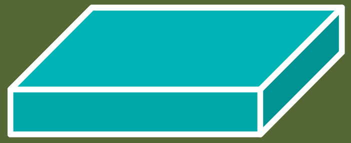
GGE (Graphical Geometry Editor)

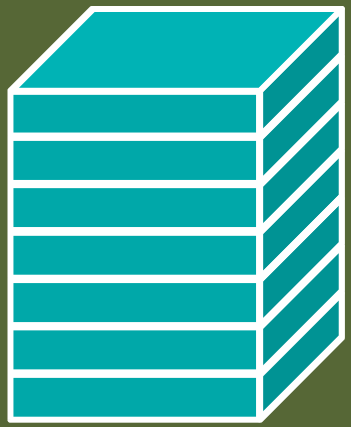
- Implemented in JAVA, GGE is a graphical geometry editor compliant to Geant4. It allows to:
  - Describe a detector geometry including:
    - materials, solids, logical volumes, placements
  - Graphically visualize the geometry using a Geant4 supported visualization system
  - Store persistently the detector description
  - Generate the C++ code according to the Geant4 specifications
- GGE is a part of MOMO. MOMO can be downloaded from Web as a separate tool:
  - **`http://erpc1.naruto-u.ac.jp/~geant4/`**

Replicated volume

Replicated Volumes

- The mother volume is completely filled with replicas, all of which are the same size (width) and shape.
- Replication may occur along:
  - Cartesian axes (X, Y, Z) – slices are considered perpendicular to the axis of replication
    - Coordinate system at the center of each replica
  - Radial axis (Rho) – cons/tubs sections centered on the origin and un-rotated
    - Coordinate system same as the mother
  - Phi axis (Phi) – phi sections or wedges, of cons/tubs form
    - Coordinate system rotated such as that the X axis bisects the angle made by each wedge


a daughter logical volume to be replicated


mother volume

G4PVReplica

```
G4PVReplica(const G4String &pName,
            G4LogicalVolume *pLogical,
            G4LogicalVolume *pMother,
            const EAxis pAxis,
            const G4int nReplicas,
            const G4double width,
            const G4double offset=0.);
```

- `offset` may be used only for tube/cone segment
- Features and restrictions:
  - Replicas can be placed inside other replicas
  - Normal placement volumes can be placed inside replicas, assuming no intersection/overlaps with the mother volume or with other replicas
  - No volume can be placed inside a radial replication
  - Parameterised volumes cannot be placed inside a replica

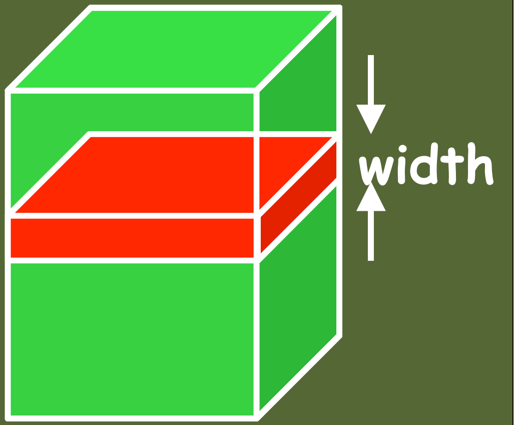
Replica - axis, width, offset

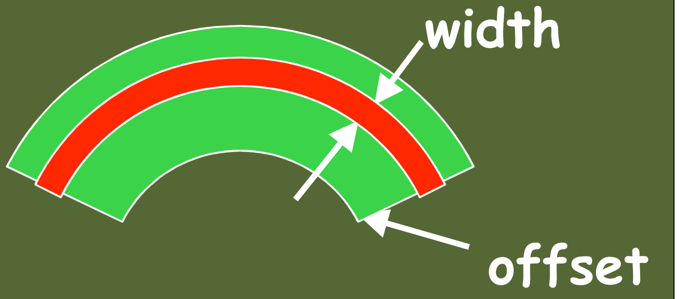
- Cartesian axes - **`kXaxis, kYaxis, kZaxis`**
  - Center of n-th daughter is given as

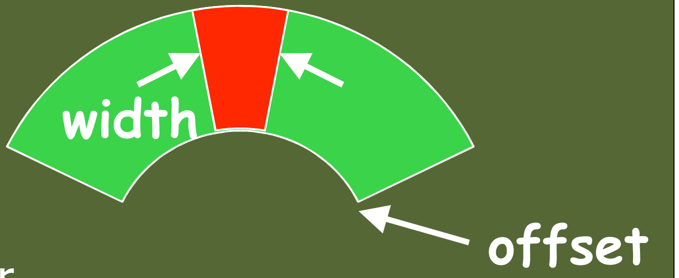
    `-width*(nReplicas-1)*0.5+n*width`
  - Offset shall not be used
- Radial axis - **`kRaxis`**
  - Center of n-th daughter is given as

    `width*(n+0.5)+offset`
  - Offset must be the inner radius of the mother
- Phi axis - **`kPhi`**
  - Center of n-th daughter is given as

    `width*(n+0.5)+offset`
  - Offset must be the starting angle of the mother



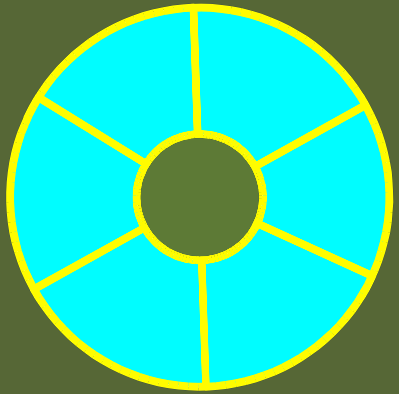


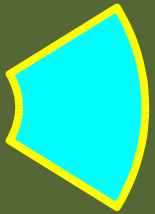


Slide from SLAC Geant4 tutorial course in '06

G4PVReplica : example

```
G4double tube_dPhi = 2.* M_PI * rad;
G4VSolid* tube =
   new G4Tubs("tube",20*cm,50*cm,30*cm,0.,tube_dPhi);
G4LogicalVolume * tube_log =
   new G4LogicalVolume(tube, Air, "tubeL", 0, 0, 0);
G4VPhysicalVolume* tube_phys =
   new G4PVPlacement(0,G4ThreeVector(-200.*cm,0.,0.),
            "tubeP", tube_log, world_phys, false, 0);
G4double divided_tube_dPhi = tube_dPhi/6.;
G4VSolid* div_tube =
   new G4Tubs("div_tube", 20*cm, 50*cm, 30*cm,
        -divided_tube_dPhi/2., divided_tube_dPhi);
G4LogicalVolume* div_tube_log =
   new G4LogicalVolume(div_tube,Pb,"div_tubeL",0,0,0);
G4VPhysicalVolume* div_tube_phys =
   new G4PVReplica("div_tube_phys", div_tube_log,
   tube_log, kPhi, 6, divided_tube_dPhi);
```





Parameterized volume

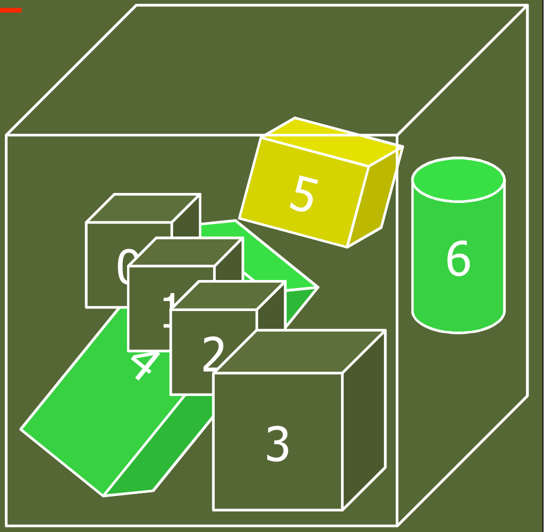
G4PVParameterised

```
G4PVParameterised(const G4String& pName,
                  G4LogicalVolume* pLogical,
                  G4LogicalVolume* pMother,
                  const EAxis pAxis,
                  const G4int nReplicas,
                  G4VPVParameterisation *pParam
                  G4bool pSurfChk=false);
```

- Replicates the volume **`nReplicas`** times using the parameterization **`pParam`**, within the mother volume **`pMother`**
- **`pAxis`** is a suggestion to the navigator along which Cartesian axis replication of parameterized volumes dominates.
  - kXAxis, kYAxis, kZAxis : one-dimensional optimization
  - kUndefined : three-dimensional optimization

Parameterized Physical Volumes

- User should implement a class derived from G4VPVParameterisation abstract base class and define following as a function of copy number
  - where it is positioned (transformation, rotation)
- Optional:
  - the size of the solid (dimensions)
  - the type of the solid, material, sensitivity, vis attributes
- All daughters must be fully contained in the mother.
- Daughters should not overlap to each other.
- Limitations:
  - Applies to simple CSG solids only
  - Granddaughter volumes allowed only for special cases
  - Consider parameterised volumes as "leaf" volumes
- Typical use-cases
  - Complex detectors
    - with large repetition of volumes, regular or irregular
  - Medical applications
    - the material in animal tissue is measured as cubes with varying material



Slide from SLAC Geant4 tutorial course in '06

G4PVParameterized : example

```
G4VSolid* solidChamber =
   new G4Box("chamber", 100*cm, 100*cm, 10*cm);
G4LogicalVolume* logicChamber =
   new G4LogicalVolume
   (solidChamber, ChamberMater, "Chamber", 0, 0, 0);
G4VPVParameterisation* chamberParam =
   new ChamberParameterisation();
G4VPhysicalVolume* physChamber =
   new G4PVParameterised("Chamber", logicChamber,
        logicMother, kZAxis, NbOfChambers, chamberParam);
```

Need to implement this

Slide from SLAC Geant4 tutorial course in ‘06

G4VPVParameterisation : example

```
class ChamberParameterisation : public G4VPVParameterisation
{
  public:
    ChamberParameterisation();
    virtual ~ChamberParameterisation();
1.  virtual void ComputeTransformation // position, rotation
      (const G4int copyNo, G4VPhysicalVolume* physVol) const;
2.  virtual void ComputeDimensions // size
      (G4Box& trackerLayer, const G4int copyNo,
          const G4VPhysicalVolume* physVol) const;
3.  virtual G4VSolid* ComputeSolid // shape
      (const G4int copyNo, G4VPhysicalVolume* physVol);
4.  virtual G4Material* ComputeMaterial // material, sensitivity, visAtt
      (const G4int copyNo, G4VPhysicalVolume* physVol,
          const G4VTouchable *parentTouch=0);
        // G4VTouchable should not be used for ordinary parameterization
};
```

G4VPVParameterisation : example

1.

```
void ChamberParameterisation::ComputeTransformation
(const G4int copyNo, G4VPhysicalVolume* physVol) const
{
  G4double Xposition = … // w.r.t. copyNo
  G4ThreeVector origin(Xposition,Yposition,Zposition);
  physVol->SetTranslation(origin);
  physVol->SetRotation(0);
}
```

2.

```
void ChamberParameterisation::ComputeDimensions
(G4Box& trackerChamber, const G4int copyNo,
 const G4VPhysicalVolume* physVol) const
{
  G4double XhalfLength = … // w.r.t. copyNo
  trackerChamber.SetXHalfLength(XhalfLength);
  trackerChamber.SetYHalfLength(YhalfLength);
  trackerChamber.SetZHalfLength(ZHalfLength);
}
```

Slide from SLAC Geant4 tutorial course in ‘06

G4VPVParameterisation : example

3.

```
G4VSolid* ChamberParameterisation::ComputeSolid
     (const G4int copyNo, G4VPhysicalVolume* physVol)
{
  G4VSolid* solid;
  if(copyNo == …) solid = myBox;
  else if(copyNo == …) solid = myTubs;
  …
  return solid;
}
```

4.

```
G4Material* ComputeMaterial // material, sensitivity, visAtt
     (const G4int copyNo, G4VPhysicalVolume* physVol,
         const G4VTouchable *parentTouch=0);
{
  G4Material* mat;
  if(copyNo == …)
  {
    mat = material1;
    physVol->GetLogicalVolume()->SetVisAttributes( att1 );
  }
  …
  return mat;
}
```

Slide from SLAC Geant4 tutorial course in ‘06

Moving Objects in GEANT4

- GEANT4 can deal with moving volume
- Assumption: The speed of moving volume is slow enough compared to the speed of particles, so that the moving volume can be treated as "stationary" within one event.
- Use parameterized volume to represent the moving volume.
  - Use event number as a time stamp and calculate position and rotation of the volume as a function of event number.
- Moving volume should not goes out of its mother volume
- For more detailed info about moving volume in GEANT4, see http://geant4.slac.stanford.edu/SLACTutorial07/Kernel3.ppt

This slide based on SLAC Geant4 tutorial slides

EM Field

- To define a global uniform magnetic field:

  Use an object of the G4UniformMagField class

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

  To define a non-uniform magnetic field, create your own class derived from G4MagneticField and implement GetFieldValue method.

- To define a global uniform electric field:

```
G4ElectricField* fEMfield =
new G4UniformElectricField( G4ThreeVector(0., 100000.*kilovolt/cm, 0.) );
```

This slide based on a SLAC Geant4 tutorial slide

Magnetic field

- Tell Geant4 to use your field

1. Find the global Field Manager

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

2. Set the field for this FieldManager,

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

1. and create a Chord Finder.

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

- /example/novice/N04/ExN04 is a good starting point

Summary

- To define a geometry, G4VUserDetectorConstruction needs to be implemented (one of the three mandatory classes)
- Basic procedures for defining a physical volume (simplest case):
    1. Define solids/shapes
    2. Define its logical volume
    3. Place volume in a mother logical volume
- Many pre-defined shapes are available in GEANT4
- Electric and Magnetic Field can be defined.