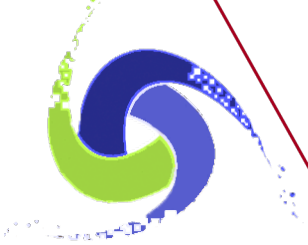


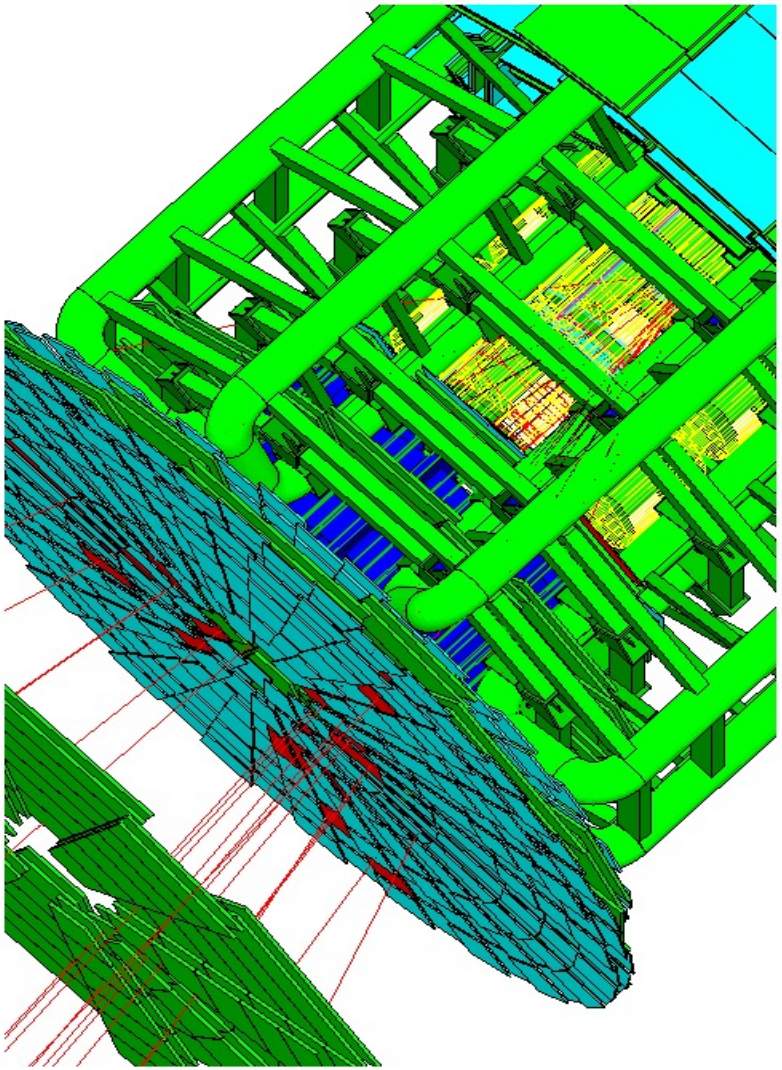
GEANT4
A SIMULATION TOOLKIT
Version 10.7.p02

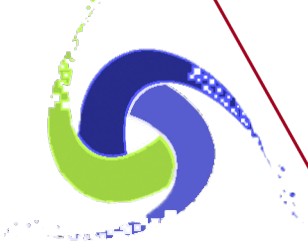
Geant4 Geometry

Makoto Asai
SLAC National Accelerator Laboratory

Contents

- Introduction
- G4VUserDetectorConstruction class
- Material
- Solid and shape
- Logical volume
- Physical volume
  - Placement volume
  - Repeated volume
- Magnetic field
- GDML/CAD interfaces



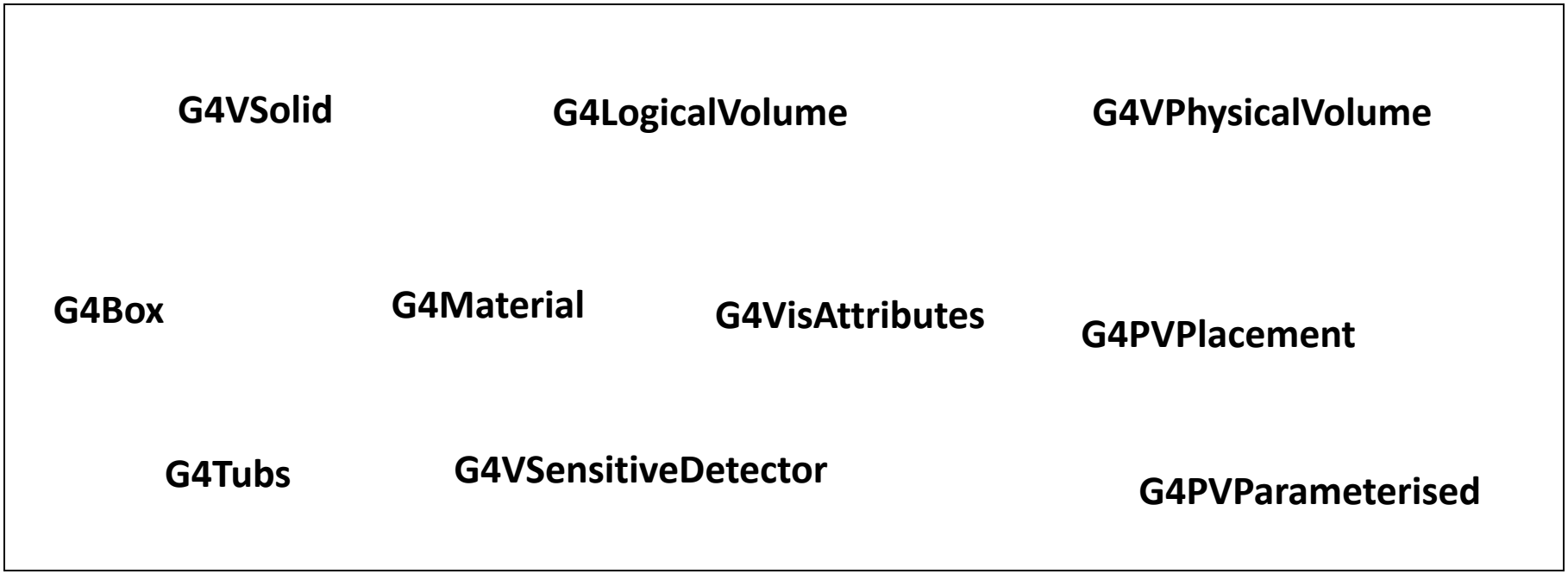


GEANT4
A SIMULATION TOOLKIT
Version 10.7.p02

Introduction

Detector geometry

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

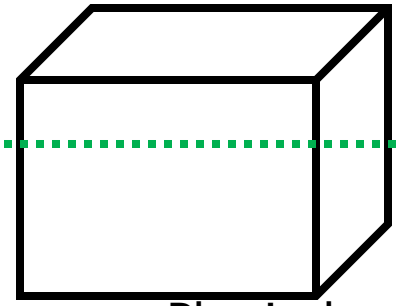


Define detector geometry

- 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);
```

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

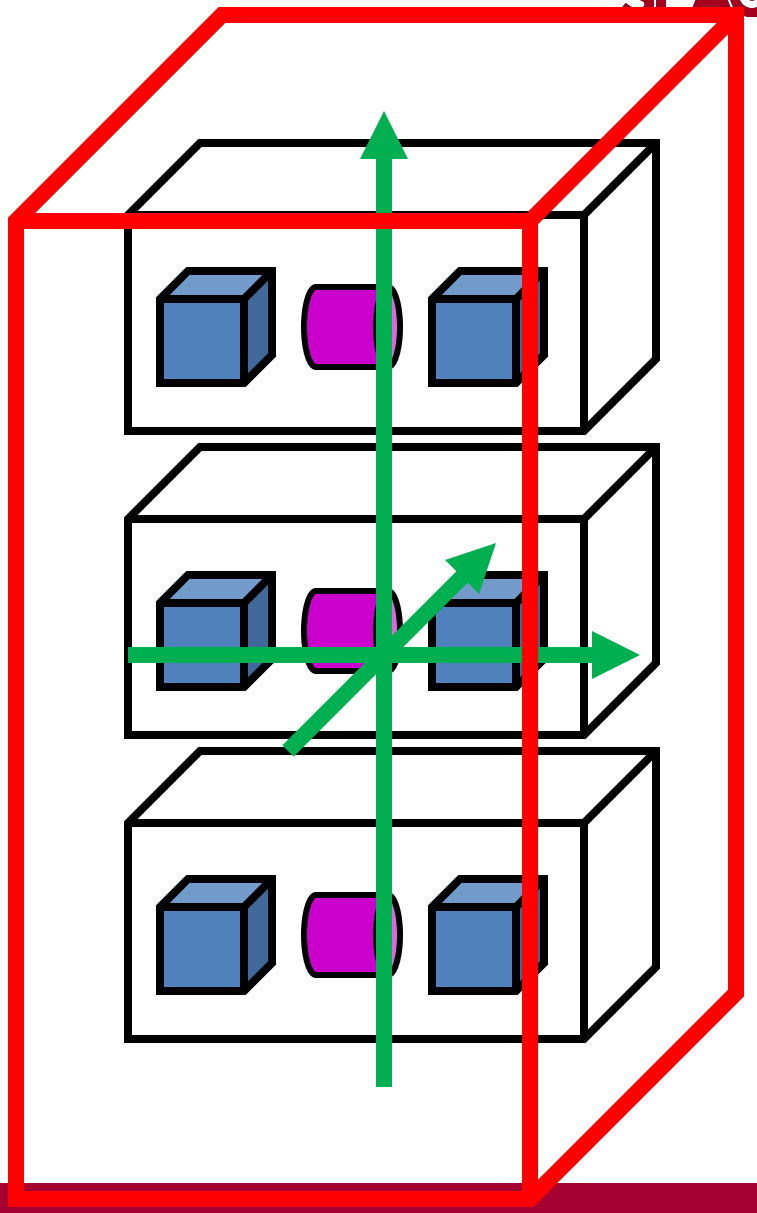


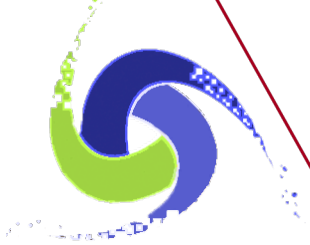
Physical volume :
+ rotation and position

- 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.

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.





GEANT4
A SIMULATION TOOLKIT
Version 10.7.p02

G4VUserDetectorConstruction

User classes

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

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

G4VUserDetectorConstruction

```
class G4VUserDetectorConstruction
{
  public:
    G4VUserDetectorConstruction();
    virtual ~G4VUserDetectorConstruction();
  public:
    virtual G4VPhysicalVolume* Construct() = 0;
    virtual void ConstructSDandField();
  public:
    void RegisterParallelWorld(G4VUserParallelWorld*);
```

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

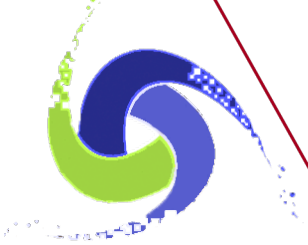
Sensitive detector and field should be instantiated and set to logical volumes in ConstructSDandField() method.

Your detector construction

```
#ifndef MyDetctorConstruction_h
#define MyDetctorConstruction_h 1
#include "G4VUserDetectorConstruction.hh"
class MyDetctorConstruction
      : public G4VUserDetectorConstruction
{
 public:
  G4VUserDetectorConstruction();
  virtual ~G4VUserDetectorConstruction();
  virtual G4VPhysicalVolume* Construct();
  virtual void ConstructSDandField();
 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 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 G4RunManager or G4MTRunManager



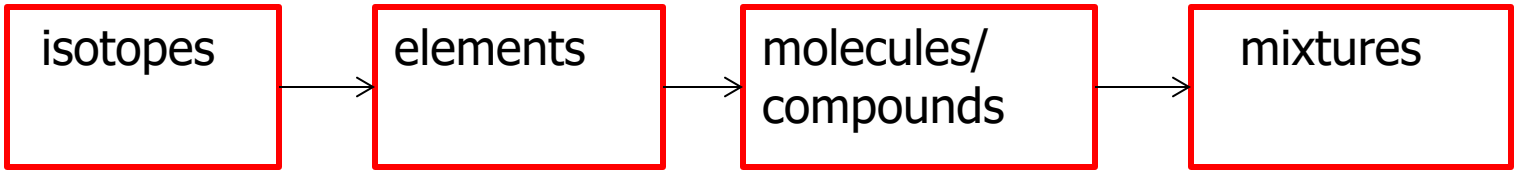
GEANT4
A SIMULATION TOOLKIT
Version 10.7.p02

Material

- Materials in Geant4
- Material definition
- NIST material database

Materials in Geant4

- Geant4 materials (like those in the real world)
  - are made up of isotopes, elements, compounds (or molecules), mixtures of elements and/or compounds



  - can be solid, liquid or gas (sorry, no plasma)
  - may exist under various pressures, temperatures and densities
- Geant4 allows the custom definition of materials
  - starting with elements: use G4Element class
  - compounds or molecules can be built by assigning two or more instances of G4Element to an object of the G4Material class
  - mixtures can be built by assigning two or more compounds or elements to an instance of G4Material
  - a G4Material can also be built from a single G4Element
  - optionally, you may define your own G4Element by assigning to it one or more instances of the class G4Isotope

Materials in Geant4

- Geant4 requires you to set at least one material condition
  - density
- The rest are optional
  - state (default is solid or gas, depending on density)
  - temperature (default = STP temperature = 273.15 K)
  - pressure (default = STP pressure = 100 kPascal = 1 atm)
- Along with normal stuff, you can define some strange things
  - gases far from STP
  - high pressure solids
  - low density liquids
- And there's a shortcut => NIST material database

Definition of Materials

- Let's start with a single-element material:

```
G4double density = 4.506*g/cm3;
G4double a = 47.867*g/mole;
G4Material* ti = new G4Material("pureTitanium", z=22, a, density);
```

- Vacuum is also useful

```
G4NistManager* manager = G4NistManager::Instance();
G4Material* vacuum = manager->FindOrBuildMaterial("G4_Galactic");
```

  - for vacuum, use low density gas rather than density = 0
  - "average" materials (e.g. z = 25.7) are not allowed

Definition of Materials: molecules

- A molecule is made of several elements, with the composition specified by the number of atoms

```
G4double a = 1.01*g/mole;
G4Element* elH = new G4Element(“Hydrogen”, “H”, z=1, a);
a = 16.00*g/mole;
G4Element* elO =  new G4Element(“Oxygen”, “O”, z=8, a);

G4double density = 1.0*g/cm3;
G4int ncomp = 2;
G4Material* H2O = new G4Material(“Water”, density, ncomp);
G4int nAtoms;
H2O->AddElement(elH, nAtoms=2);
H2O->AddElement(elO, nAtoms=1);
```

Definition of Materials: mixtures (alloys)

- A mixture is similar to a molecule, except that materials and elements are combined instead of just elements

```
G4Element* elC = … ;       // define carbon
G4Material* H2O = … ;      // define molecule (previous page)
G4Material* SiO2 = … ;     // define another molecule

G4double density = 0.20*g/cm3;
G4int ncomp = 3;
G4double fracMass;
G4Material* Aerog = new G4Material("Aerogel", density, ncomp);

Aerog->AddMaterial(SiO2, fracMass = 62.5*perCent);
Aerog->AddMaterial(H2O, fracMass = 37.4*perCent);
Aerog->AddElement(elC, fracMass = 0.1*perCent);
```

Using the NIST Material Database

- Most of the materials you will want to define are already done for you
  - also all elements with natural isotopic abundance
  - more than 3000 isotopes defined
- Geant4 has included these pre-defined materials from the NIST database
  - physics.nist.gov/PhysRefData
  - provides the best accuracy for major parameters
    - density
    - isotopic composition of elements
    - elemental composition of materials
    - mean ionization potential
    - chemical bonds
- Documentation provided in README files in each example, and web pages

Pre-defined NIST Materials

- NIST elementary materials
  - up to Z = 98 (Cf)
- NIST compounds and mixtures
  - tissue equivalent plastic, dry air at sea level, many others
- HEP and nuclear materials
  - liquid Ar, $PbWO_4$, CR39, etc.
- Space materials
  - Kevlar, Dacron and so on
- Biochemical materials
  - adipose tissue, cytosine, thymine, etc.
- 315 materials so far

How to Use Pre-defined Materials

- User interfaces to C++ code:

```
G4NistManager* man = G4NistManager::Instance();
G4Element* elm = man->FindOrBuildElement("Chem symbol");
G4Element* elm = man->FindOrBuildElement(G4int Z);

G4Material* mat = man->FindOrBuildMaterial("Name");
G4Material* mat = man->ConstructNewMaterial("Name",
                                const std::vector<G4int>& Z,
                                const std::vector<G4double>& weight,
                                G4double density);

G4double isotopeMass = man->GetMass(G4int Z, G4int N);
```

- Elements built with natural isotope abundance

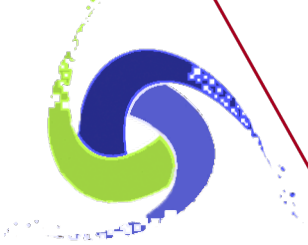
How to Use Pre-defined Materials

- User interfaces to Geant4 command line
  - list all NIST-defined elements:

/material/nist/printElement

  - list all NIST-defined materials:

/material/nist/listMaterials

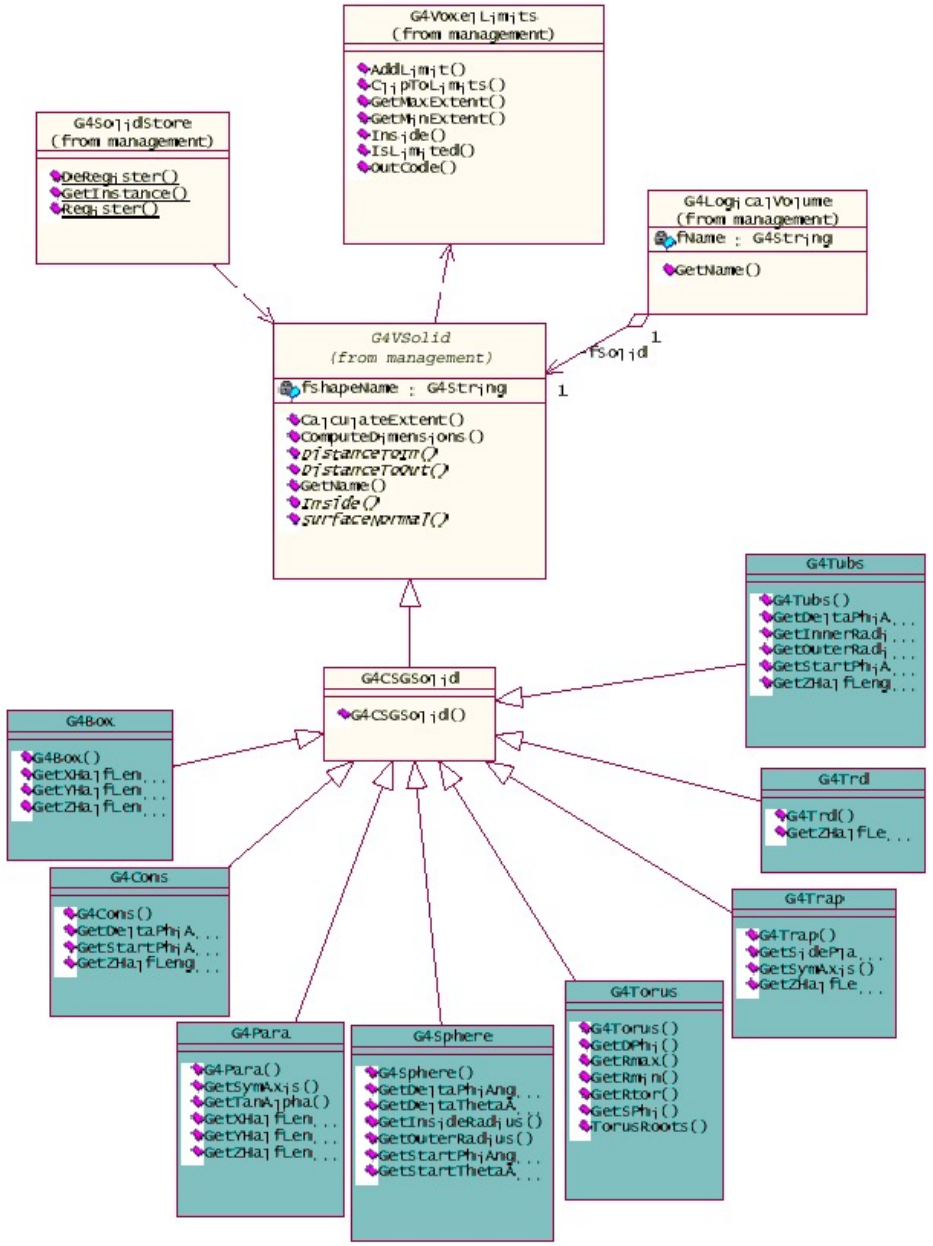


GEANT4
A SIMULATION TOOLKIT
Version 10.7.p02

Solid and shape

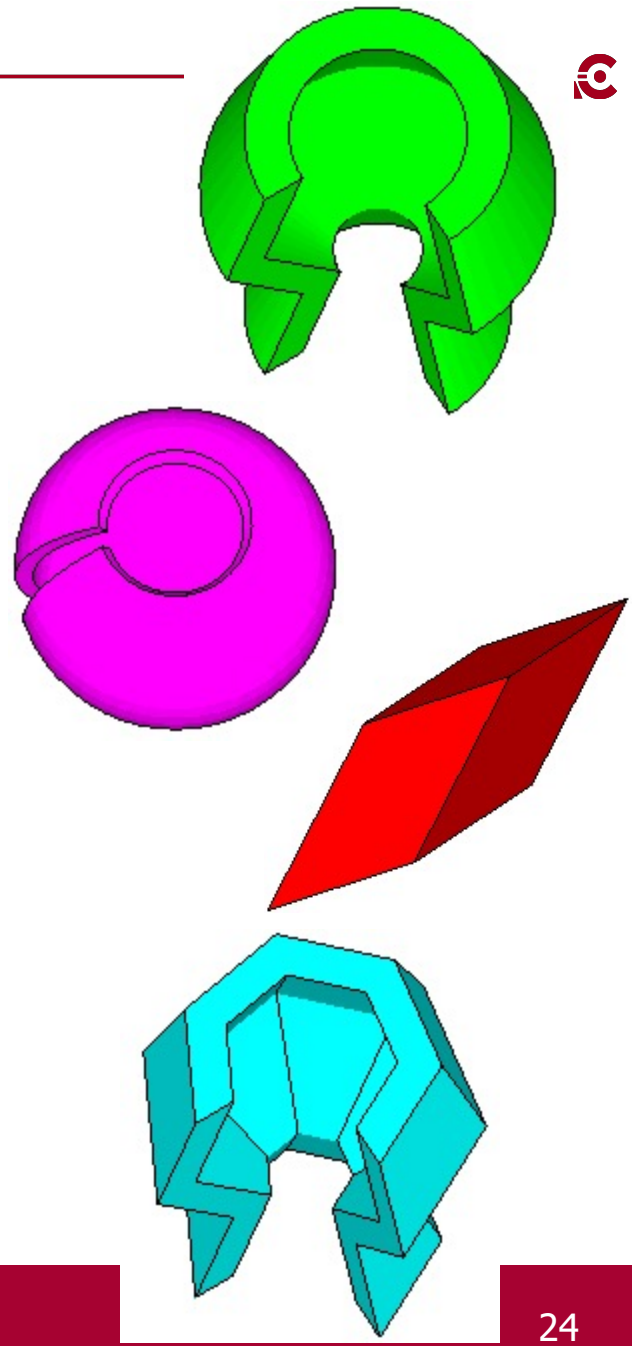
G4VSolid

- 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.



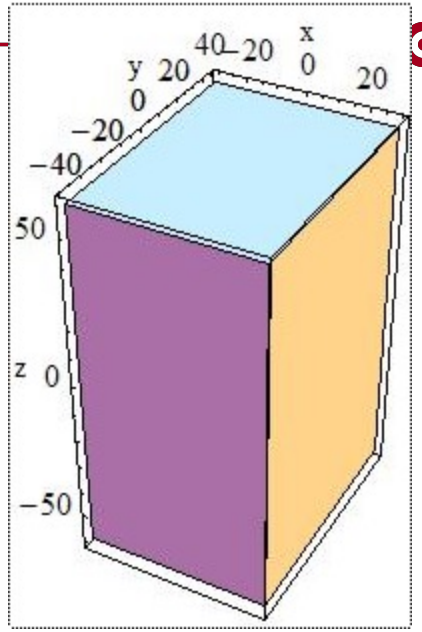
Solids

- 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, …
  - Tessellated solid
    - Solid made by facets
  - Boolean solids
    - G4UnionSolid, G4SubtractionSolid, …
    - G4MultiUnion

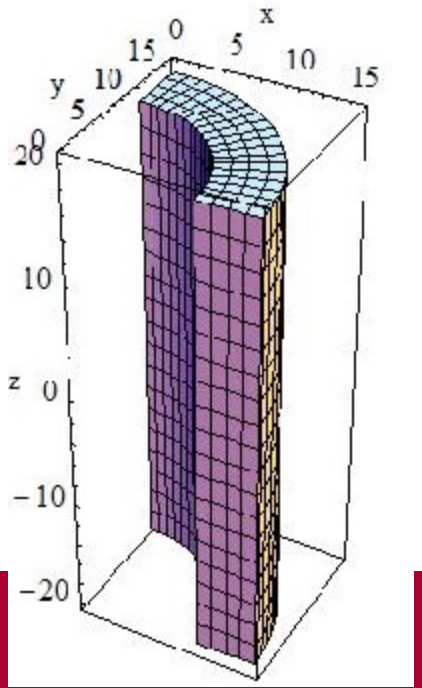


CSG: G4Box, G4Tubs

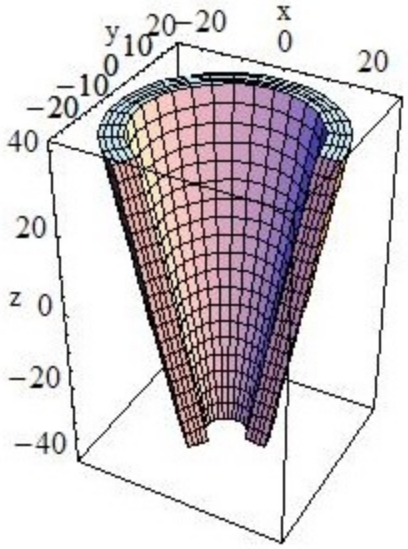
```
G4Box(const G4String &pname,    // name
            G4double half_x,    // X half size
            G4double half_y,    // Y half size
            G4double half_z);   // Z half size
```



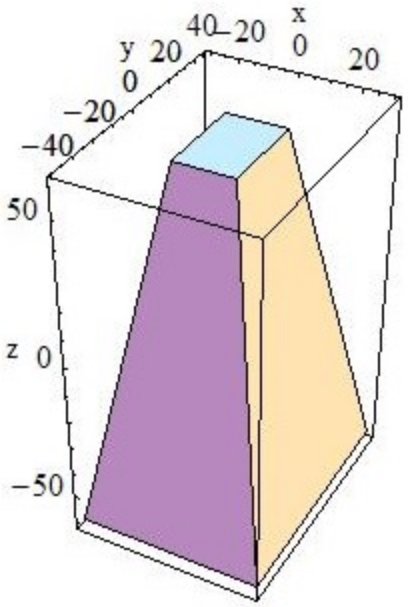

```
G4Tubs(const G4String &pname,   // name
             G4double  pRmin,   // inner radius
             G4double  pRmax,   // outer radius
             G4double  pDz,     // Z half length
             G4double  pSphi,   // starting Phi
             G4double  pDphi); // segment angle
```



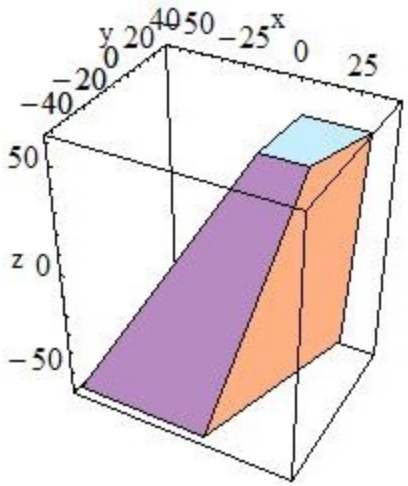
Other CSG solids



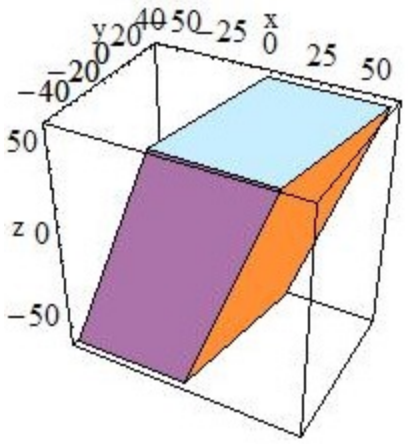
G4Cons



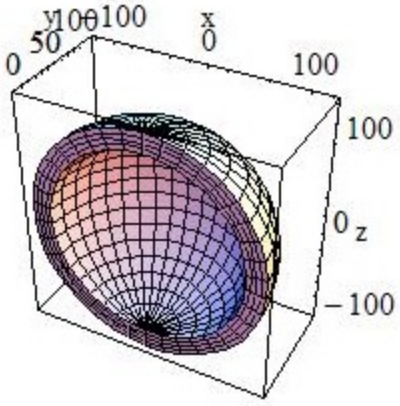
G4Trd



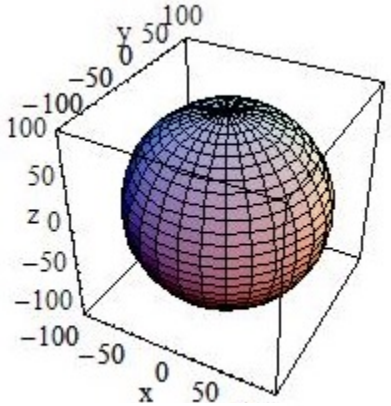
G4Trap



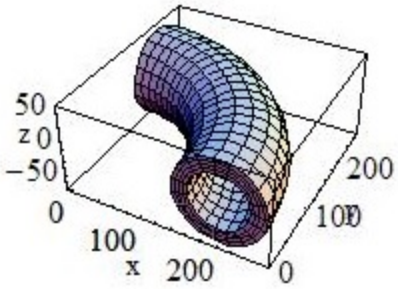
G4Para
(parallelepiped)



G4Sphere



G4Orb
(full solid sphere)

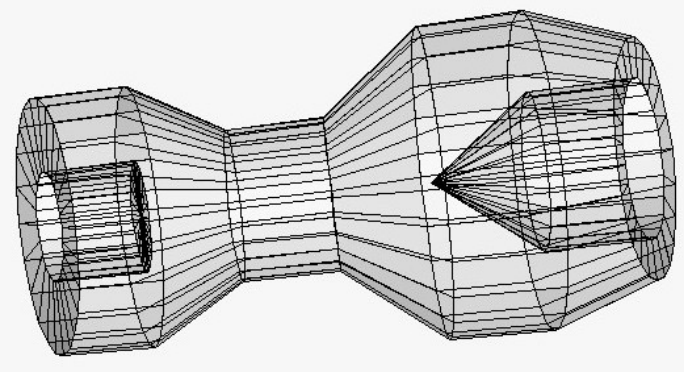


G4Torus

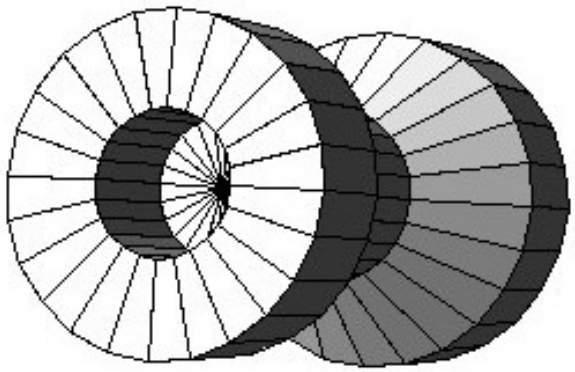
Consult to Section 4.1.2 of Geant4 Application Developers Guide for all available shapes.

Specific CSG Solids: G4Polycone

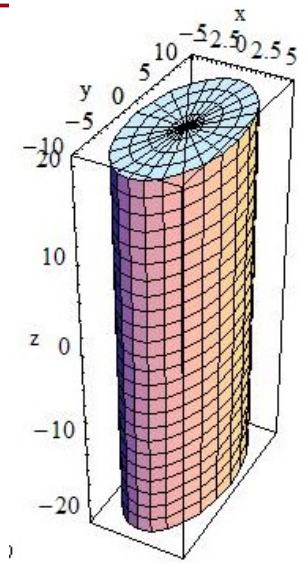
```
G4Polycone(const G4String& pName,
           G4double phiStart,
           G4double phiTotal,
           G4int numRZ,
           const G4double r[],
           const G4double z[]);
```



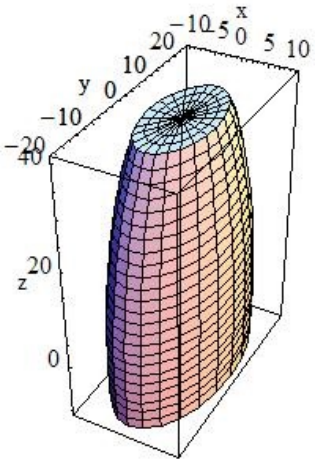
- **numRZ** - numbers of corners in the **r,z** space
- **r, z** - coordinates of corners



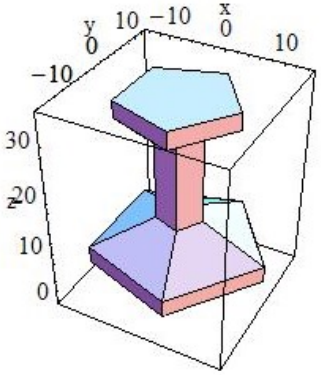
Other Specific CSG solids



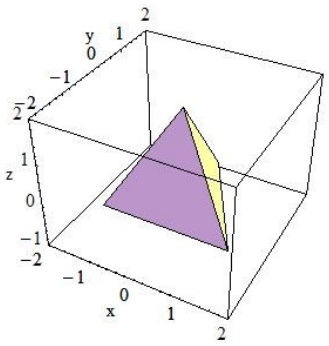
G4EllipticalTube



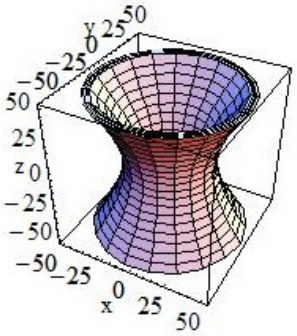
G4Ellipsoid



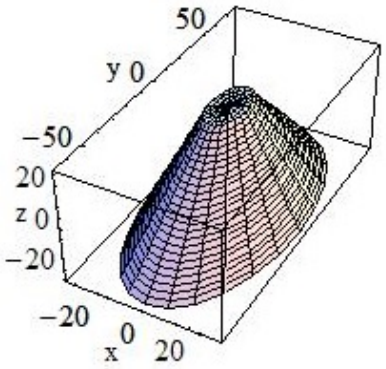
G4Polyhedra



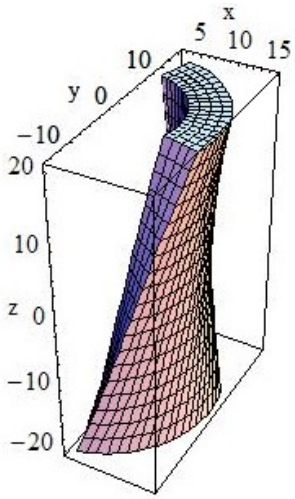
G4Tet
(tetrahedra)



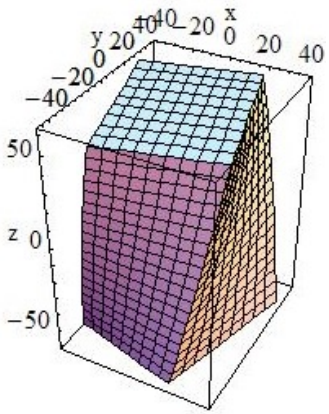
G4Hype



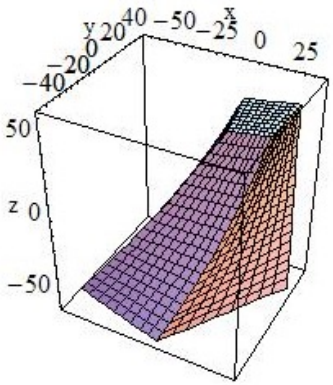
G4EllipticalCone



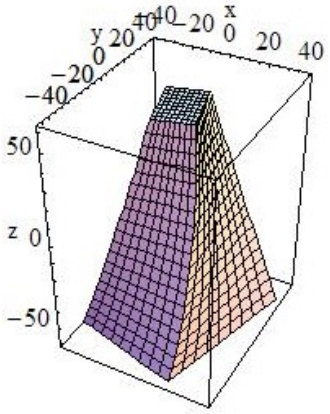
G4TwistedTubs



G4TwistedBox



G4TwistedTrap

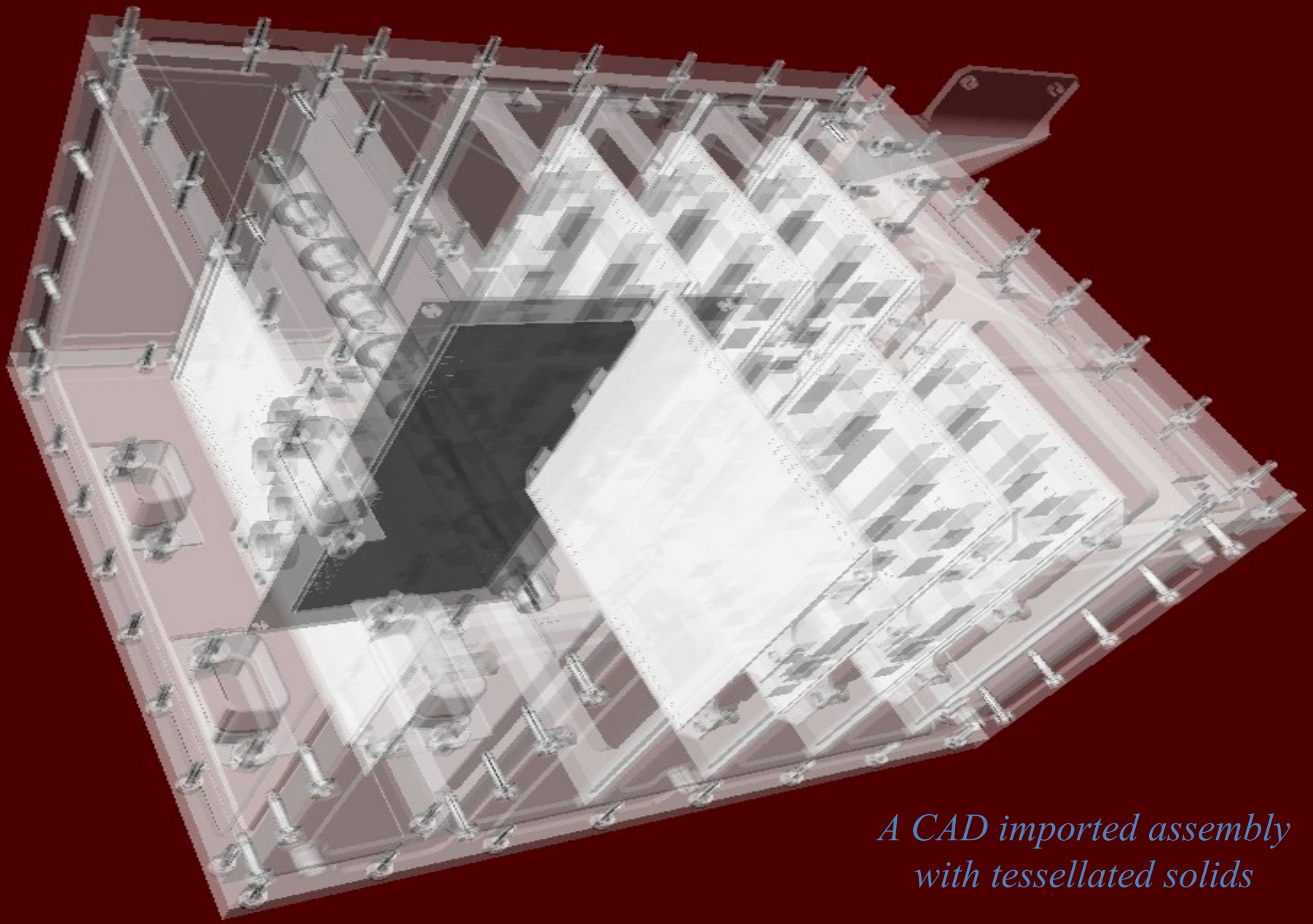


G4TwistedTrd

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

Tessellated solids

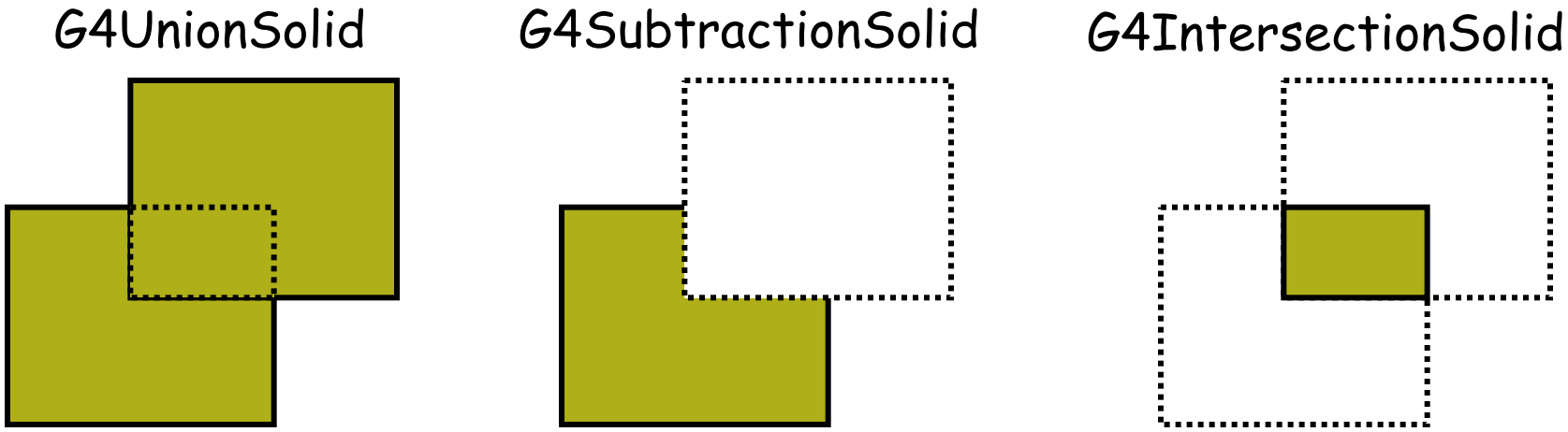
- **`G4TessellatedSolid`** (since 8.1)
  - Generic solid defined by a number of facets (**`G4VFacet`**)
    - Facets can be triangular (**`G4TriangularFacet`**) or quadrangular (**`G4QuadrangularFacet`**)
  - Constructs especially important for conversion of complex geometrical shapes imported from CAD 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



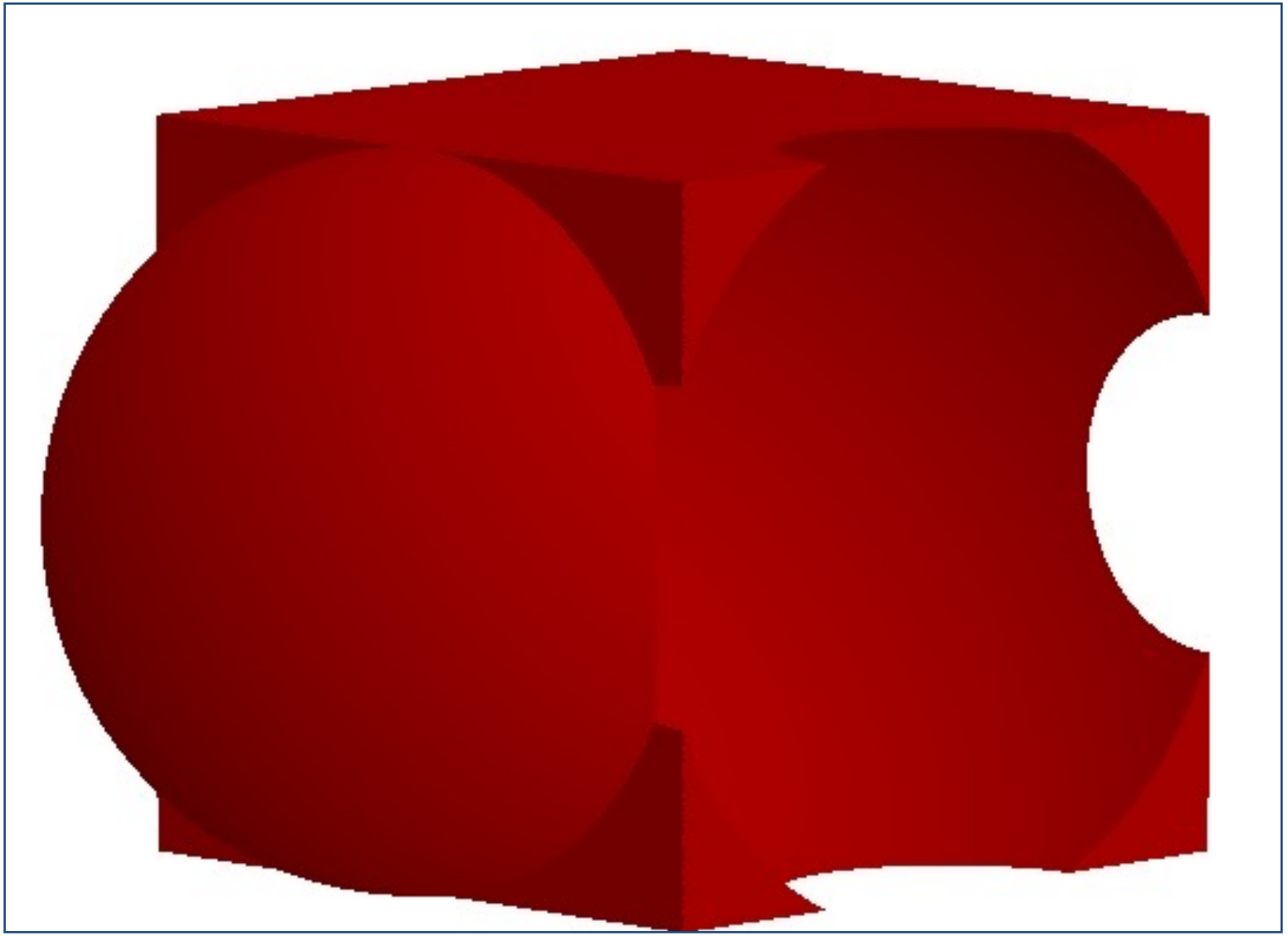
A CAD imported assembly with tessellated solids

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.



Boolean solid

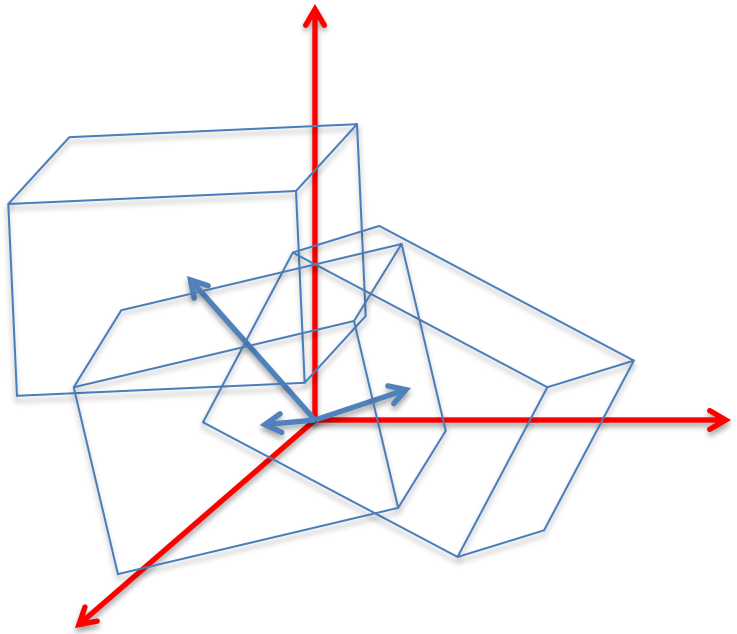


G4MultiUnion

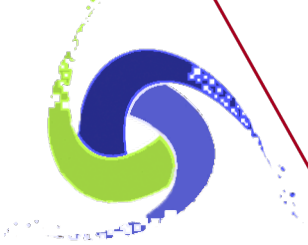
```
G4MultiUnion* munion_solid = new G4MultiUnion("UnitedBoxes");

for( int i=0 ; i < nNode ; i++)
{
  G4Box* aBox = new G4Box(…);
  G4ThreeVector pos = G4ThreeVector(…);
  G4RotationMatrix rot = G4ThreeVector(…);
  G4Transform3D tr = G4Transform3D(rot, pos);
  munion_solid -> AddNode( *aBox, tr );
}

munion_solid -> Voxelize();
```



Note : G4MultiUnion is a solid. Use it to create a logical volume.



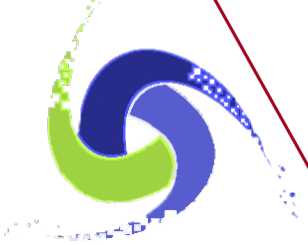
GEANT4
A SIMULATION TOOLKIT
Version 10.7.p02

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.

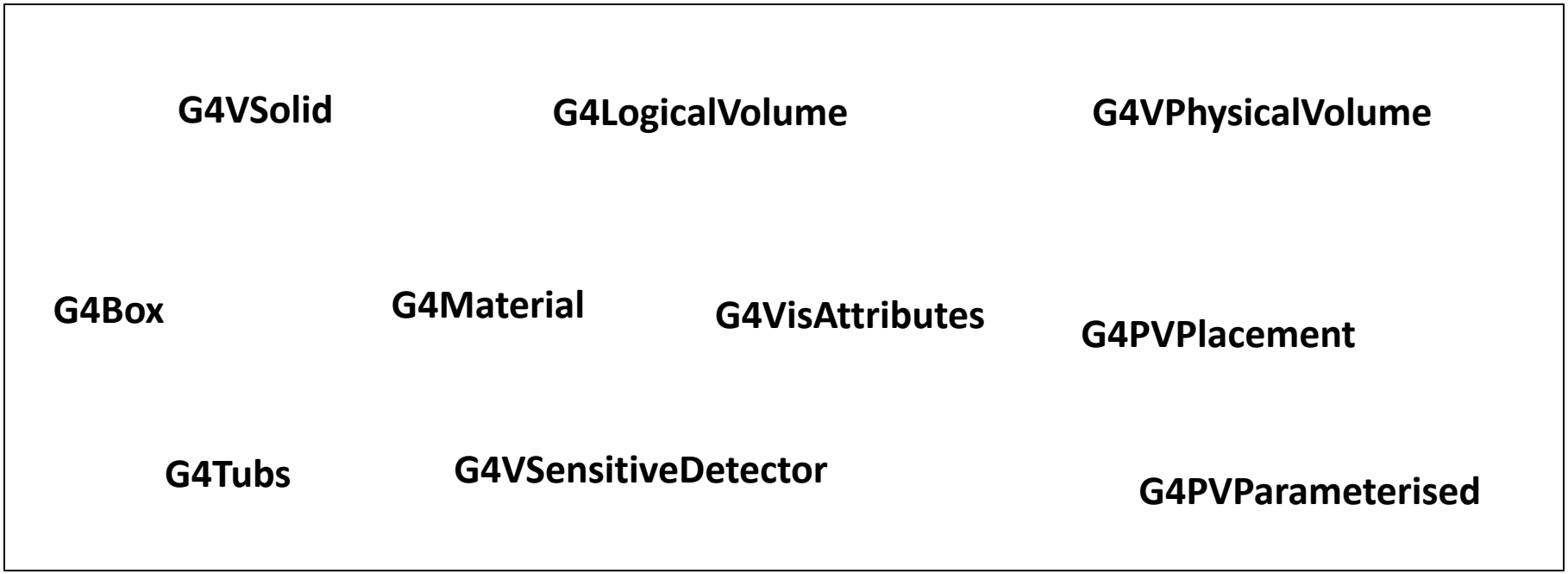


GEANT4
A SIMULATION TOOLKIT
Version 10.7.p02

Physical volume

Detector geometry

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



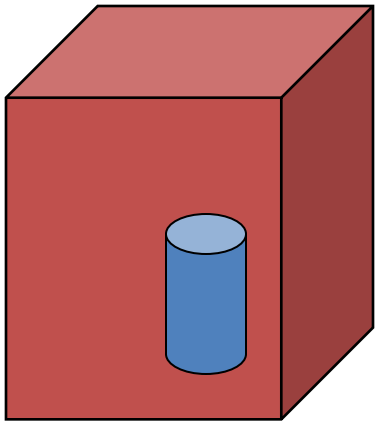
Define detector geometry

- Basic strategy

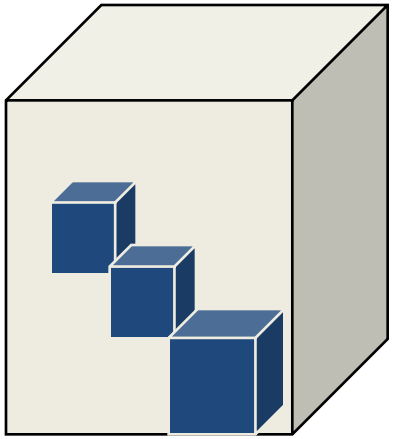
```
G4VSolid* pBoxSolid =
  new G4Box("aBoxSolid", 1.*m, 2.*m, 3.*m);
G4LogicalVolume* pBoxLog =
  new G4LogicalVolume( pBoxSolid, pBoxMaterial,
                       "aBoxLog", 0, 0, 0);
G4VPhysicalVolume* aBoxPhys =
  new G4PVPlacement( pRotation,
    G4ThreeVector(posX, posY, posZ), pBoxLog,
   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

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

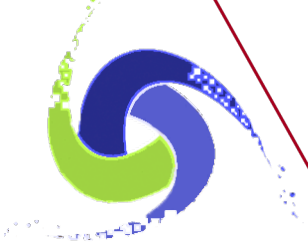
- G4PVPlacement 1 Placement = One Placement Volume
  - A volume instance positioned once in its mother volume
- G4PVParameterised 1 Parameterized = Many Repeated Volumes
  - Parameterized by the copy number
    - Shape, size, material, sensitivity, vis attributes, position and rotation can be parameterized by the copy number.
    - You have to implement a concrete class of G4VPVParameterisation.
  - Reduction of memory consumption
  - Currently: parameterization can be used only for volumes that either

    a) have no further daughters, or

    b) are identical in size & shape (so that grand-daughters are safely fit inside).
  - By implementing G4PVNestedParameterisation instead of G4VPVParameterisation, material, sensitivity and vis attributes can be parameterized by the copy numbers of ancestors.

Physical volume

- G4PVReplica 1 Replica = Many Repeated Volumes
  - Daughters of same shape are aligned along one axis
  - Daughters fill the mother completely without gap in between.
- G4PVDivision 1 Division = Many Repeated Volumes
  - Daughters of same shape are aligned along one axis and fill the mother.
  - There can be gaps between mother wall and outmost daughters.
  - No gap in between daughters.
- G4ReflectionFactory 1 Placement = a pair of Placement volumes
  - generating placements of a volume and its reflected volume
  - Useful typically for end-cap calorimeter
- G4AssemblyVolume 1 Placement = a set of Placement volumes
  - Position a group of volumes



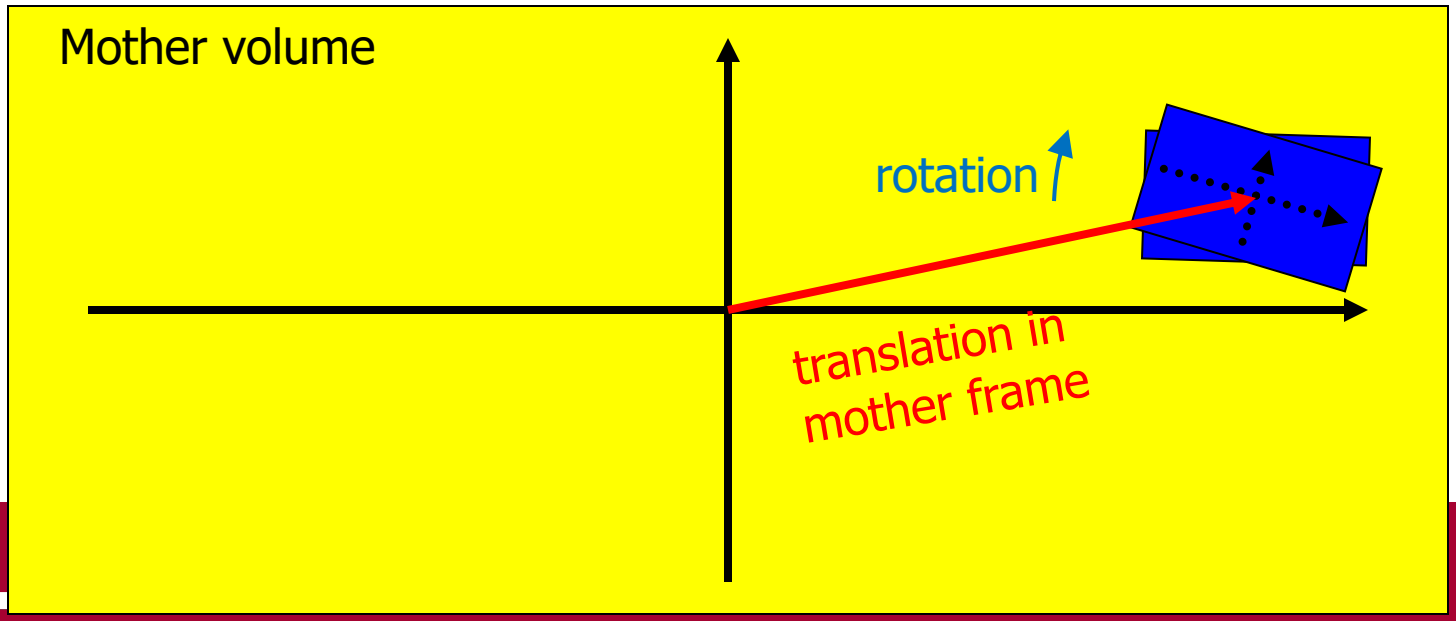
GEANT4
A SIMULATION TOOLKIT
Version 10.7.p02

G4PVPlacement

G4PVPlacement

```
G4PVPlacement(
  G4Transform3D(G4RotationMatrix &pRot, // rotation of daughter volume
                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.

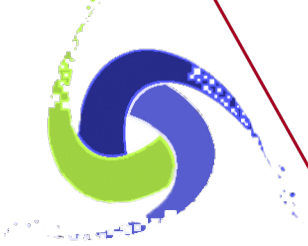


Alternative G4PVPlacement

```
G4PVPlacement(G4RotationMatrix* pRot,   // rotation of mother frame
              const G4ThreeVector &tlate, // position in mother frame
              G4LogicalVolume *pDaughterLogical,
              const G4String &pName,
              G4LogicalVolume *pMotherLogical,
              G4bool pMany, // 'true' is not supported yet…
```

Note:

- This G4PVPlacement is identical to the previous one if there is no rotation.
  - 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.
  - This is for power-users, though.



GEANT4
A SIMULATION TOOLKIT
Version 10.7.p02

Parameterized volume

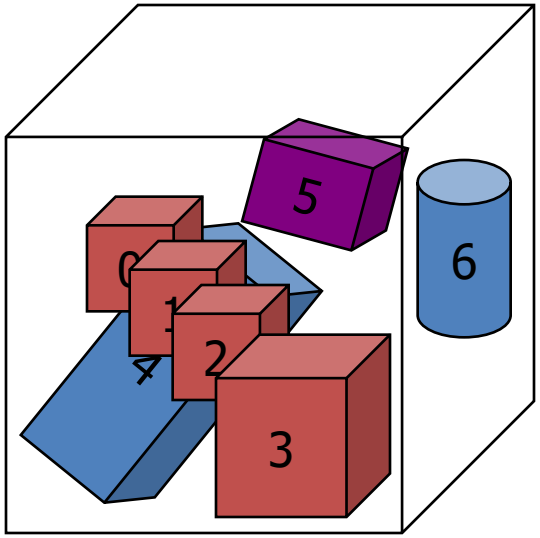
G4PVParameterised

```
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



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);
```

G4VPVParameterisation : example

```
class ChamberParameterisation : public G4VPVParameterisation
{
  public:
   ChamberParameterisation();
   virtual ~ChamberParameterisation();
   virtual void ComputeTransformation // position, rotation
     (const G4int copyNo, G4VPhysicalVolume* physVol) const;
   virtual void ComputeDimensions // size
     (G4Box& trackerLayer, const G4int copyNo,
         const G4VPhysicalVolume* physVol) const;
   virtual G4VSolid* ComputeSolid // shape
     (const G4int copyNo, G4VPhysicalVolume* physVol);
   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

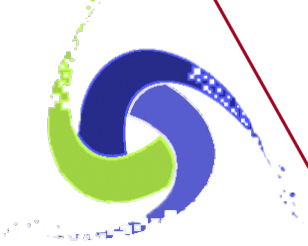
```
void ChamberParameterisation::ComputeTransformation
(const G4int copyNo, G4VPhysicalVolume* physVol) const
{
  G4double Xposition = … // w.r.t. copyNo
  G4ThreeVector origin(Xposition,Yposition,Zposition);
  physVol->SetTranslation(origin);
  physVol->SetRotation(0);
}

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);
}
```

G4VPVParameterisation : example

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

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;
}
```

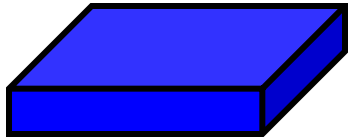


GEANT4
A SIMULATION TOOLKIT
Version 10.7.p02

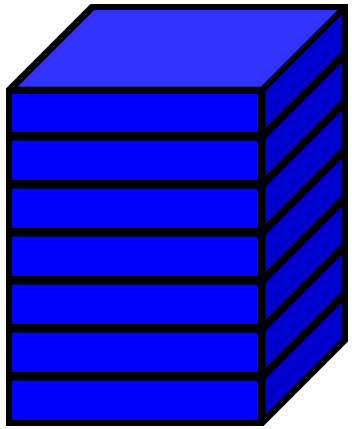
Replicated volume

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

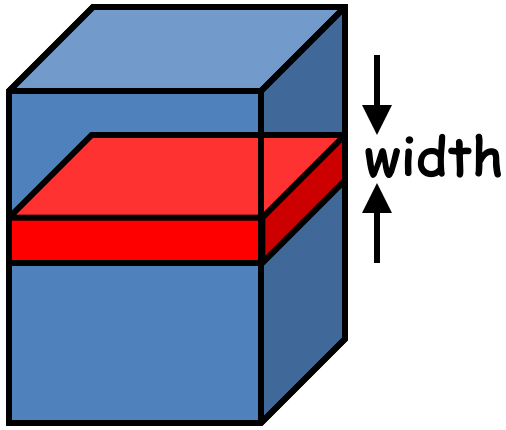
Replica - axis, width, offset

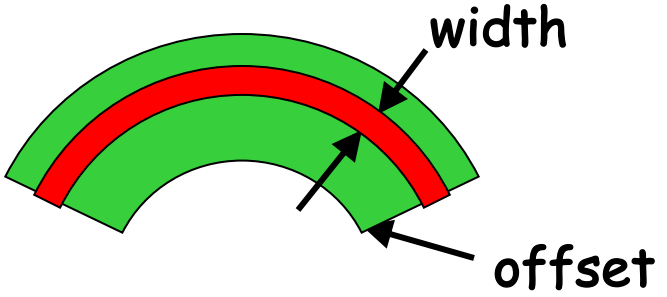
- Cartesian axes - **`kXaxis, kYaxis, kZaxis`**
  - Center of n-th daughter is given as

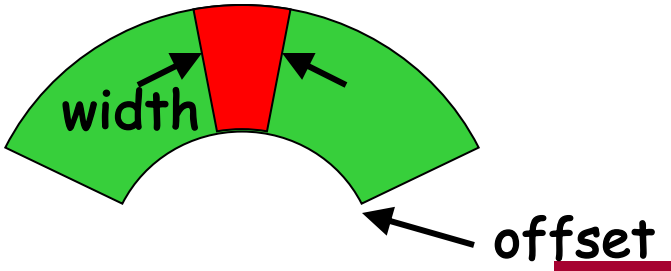
    `-width*(nReplicas-1)*0.5+n*width`
  - Offset shall not be used
- Radial axis - **`kRaxis`**
  - Center of n-th daughter is given as

    `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

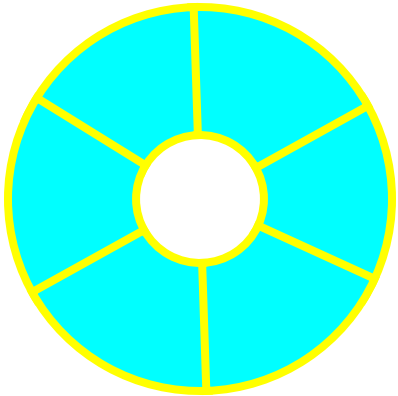


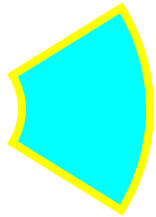


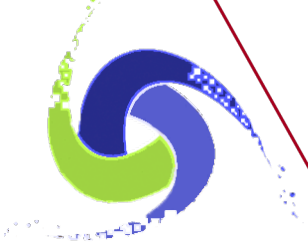


G4PVReplica : example

```
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);
```





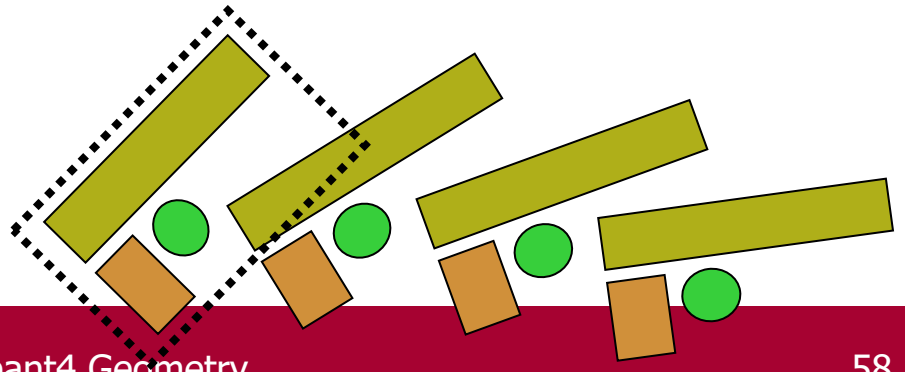


GEANT4
A SIMULATION TOOLKIT
Version 10.7.p02

Assembly volume

Grouping volumes

- To represent a regular pattern of positioned volumes, composing a more or less complex structure
  - structures which are hard to describe with simple replicas or parameterised volumes
  - structures which may consist of different shapes
  - Too densely positioned to utilize a mother volume
- Assembly volume
  - acts as an *envelope* for its daughter volumes
  - its role is over once its logical volume has been placed
  - daughter physical volumes become independent copies in the final structure
- Participating daughter logical volumes are treated as triplets
  - logical volume
  - translation w.r.t. envelop
  - rotation w.r.t. envelop



G4AssemblyVolume

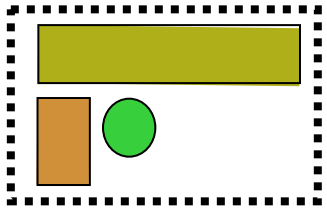
```
G4AssemblyVolume::AddPlacedVolume
                    ( G4LogicalVolume* volume,
                      G4ThreeVector& translation,
                      G4RotationMatrix* rotation );
```

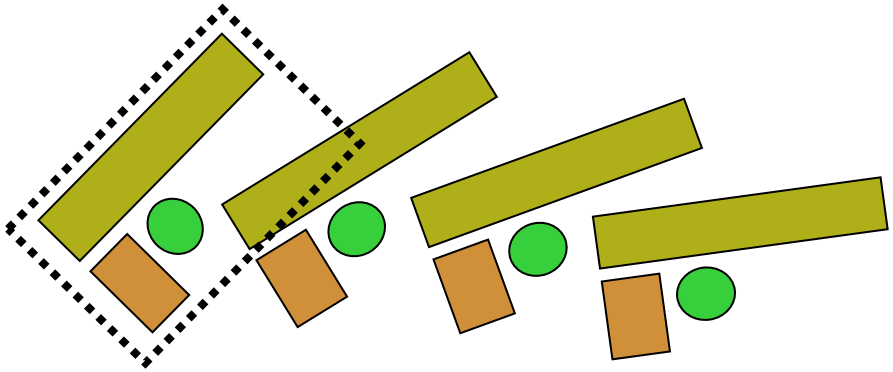
- Helper class to combine daughter logical volumes in arbitrary way
  - Imprints of the assembly volume are made inside a mother logical volume through `G4AssemblyVolume::MakeImprint(…)`
  - Each physical volume name is generated automatically
    - Format: **av**_WWW_**impr**_XXX_YYY_ZZZ
      - **WWW** – assembly volume instance number
      - **XXX** – assembly volume imprint number
      - **YYY** – name of the placed logical volume in the assembly
      - **ZZZ** – index of the associated logical volume
  - Generated physical volumes (and related transformations) are automatically managed (creation and destruction)

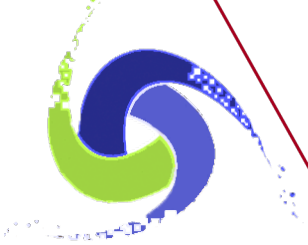
G4AssemblyVolume : example

```
G4AssemblyVolume* assembly = new G4AssemblyVolume();
G4RotationMatrix Ra;
G4ThreeVector Ta;
Ta.setX(…); Ta.setY(…); Ta.setZ(…);
assembly->AddPlacedVolume( plateLV, Ta, Ra );
  … // repeat placement for each daughter

for( unsigned int i = 0; i < layers; i++ ) {
  G4RotationMatrix Rm(…);
  G4ThreeVector Tm(…);
  assembly->MakeImprint( worldLV, Tm, Rm );
}
```







GEANT4
A SIMULATION TOOLKIT
Version 10.7.p02

Defining a magnetic field

Magnetic field (1)

- Create your Magnetic field class. It must be instantiated in ConstructSDandField() method of your DetectorConstruction
  - Uniform field :
    - Use an object of the G4UniformMagField class

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

  - Non-uniform field :
    - Create your own concrete class derived from G4MagneticField and implement **`GetFieldValue`** method.

```
void MyField::GetFieldValue(
        const double Point[4], double *field) const
```

    - Point[0..2] are position in global coordinate system, Point[3] is time
    - field[0..2] are returning magnetic field

Global and local fields

- One field manager is associated with the 'world' and it is set in G4TransportationManager
- Other volumes can override this
  - An alternative field manager can be associated with any logical volume
    - The field must accept position in global coordinates and return field in global coordinates
  - By default this is propagated to all its daughter volumes

```
G4FieldManager* localFieldMgr
      = new G4FieldManager(magField);
logVolume->setFieldManager(localFieldMgr, true);
```

where '**true**' makes it push the field to all the volumes it contains, unless a daughter has its own field manager.

- Customizing the field propagation classes
  - Choosing an appropriate stepper for your field
  - Setting precision parameters

Magnetic field (2)

```
MyField* fMyField = new MyField();

G4Fieldmanager* fFieldMgr = new G4FieldManager();

fFieldMgr->SetDetectorField(fMyField);

fFieldMgr->CreateChordFinder(fMyField);

G4bool forceToAllDaughters = true;

fMagneticLogical->SetFieldManager(fFieldMgr,
                                  forceToAllDaughters);
```

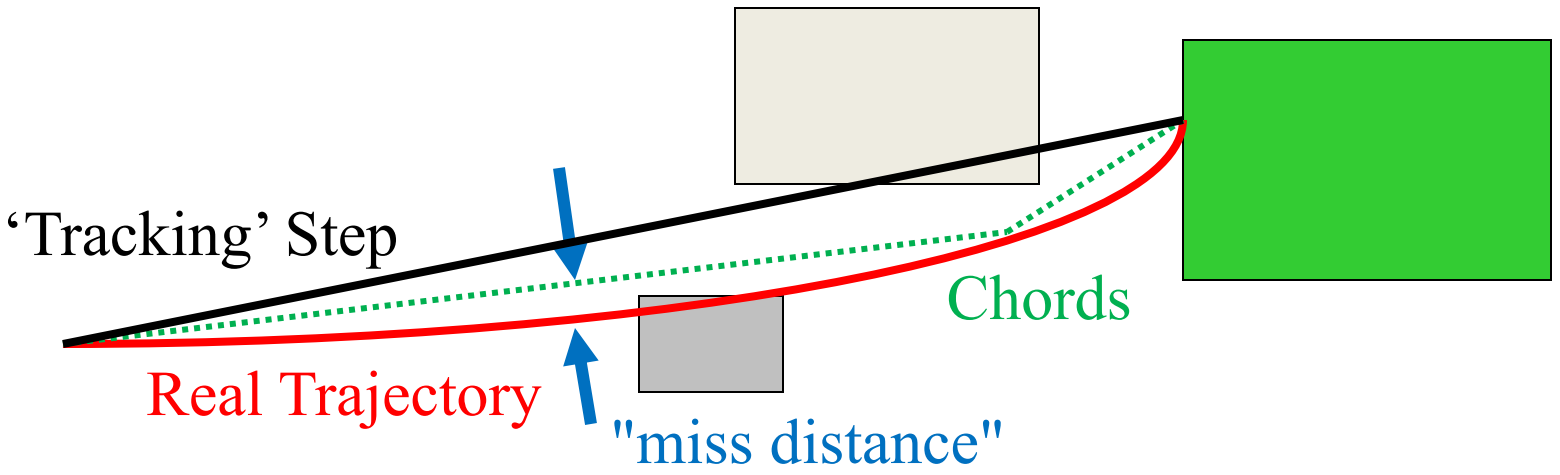
- /example/basic/B5 is a good starting point

Field integration

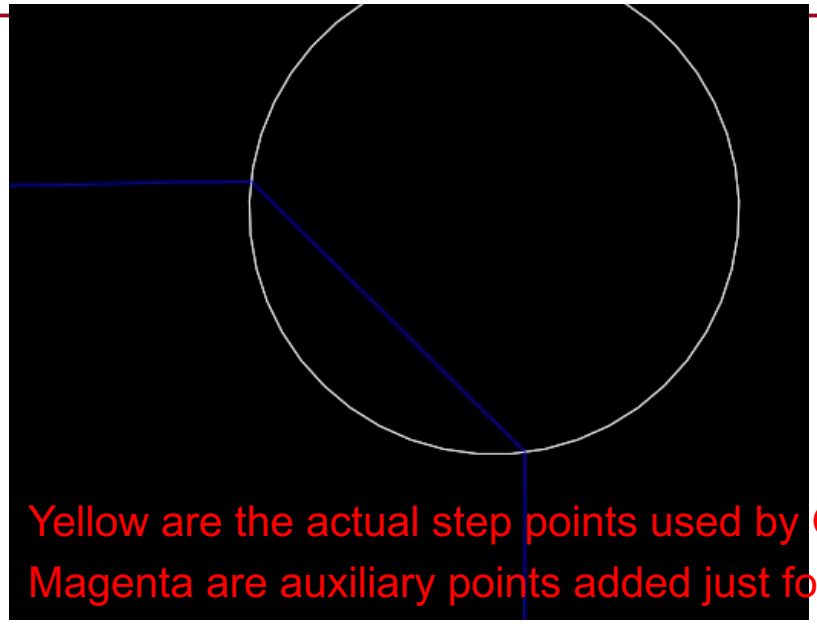
- In order to propagate a particle inside a field (e.g. magnetic, electric or both), we solve the equation of motion of the particle in the field.
- We use a Runge-Kutta method for the integration of the ordinary differential equations of motion.
  - Several Runge-Kutta 'steppers' are available.
- In specific cases other solvers can also be used:
  - In a uniform field, using the analytical solution.
  - In a smooth but varying field, with RK+helix.
- Using the method to calculate the track's motion in a field, Geant4 breaks up this curved path into linear chord segments.
  - We determine the chord segments so that they closely approximate the curved path.

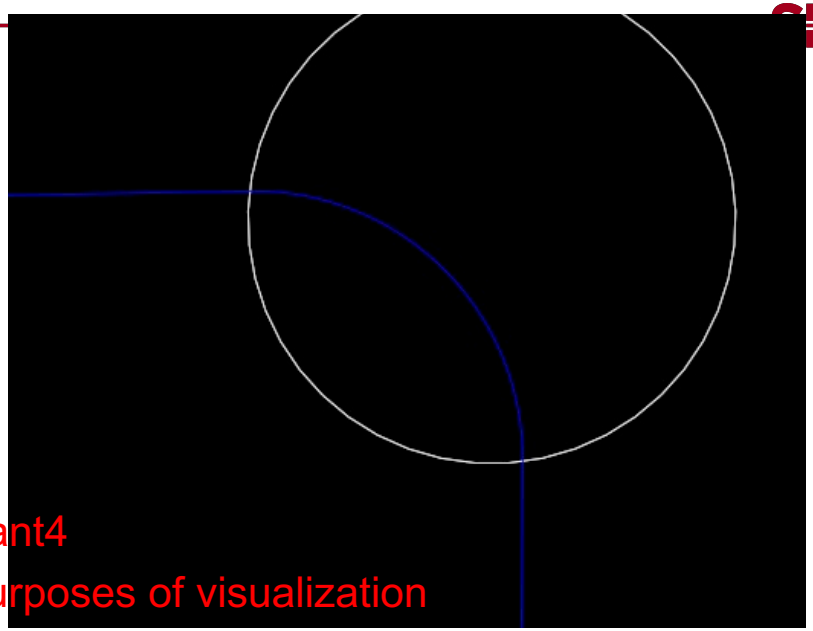
Tracking in field

- We use the chords to interrogate the G4Navigator, to see whether the track has crossed a volume boundary.
- One physics/tracking step can create several chords.
  - In some cases, one step consists of several helix turns.
- User can set the accuracy of the volume intersection,
  - By setting a parameter called the “miss distance”
    - It is a measure of the error in whether the approximate track intersects a volume.
    - It is quite expensive in CPU performance to set too small “miss distance”.



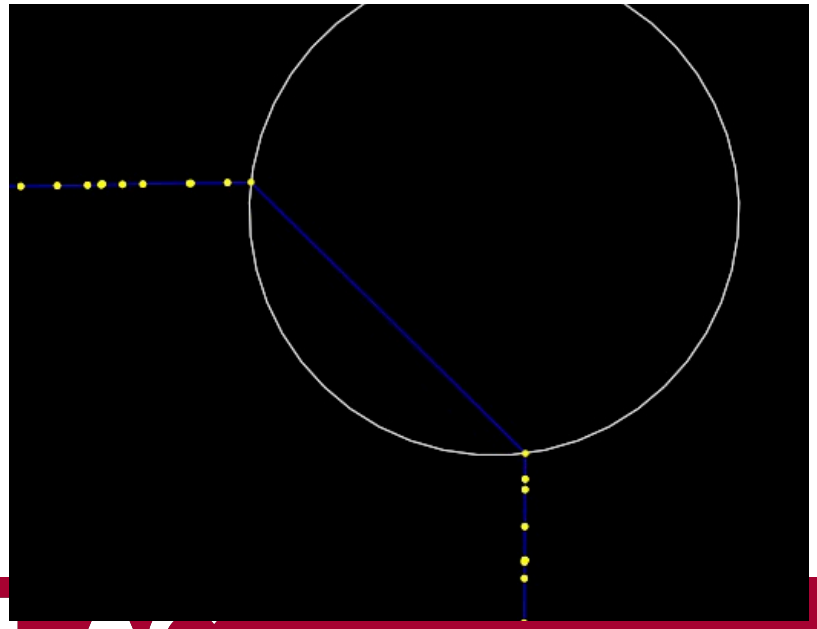
Regular versus Smooth Trajectory

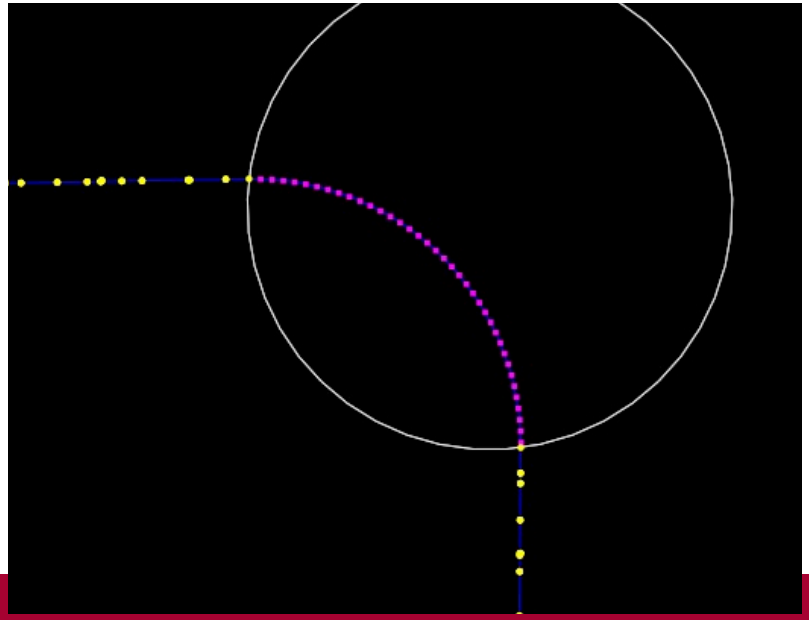




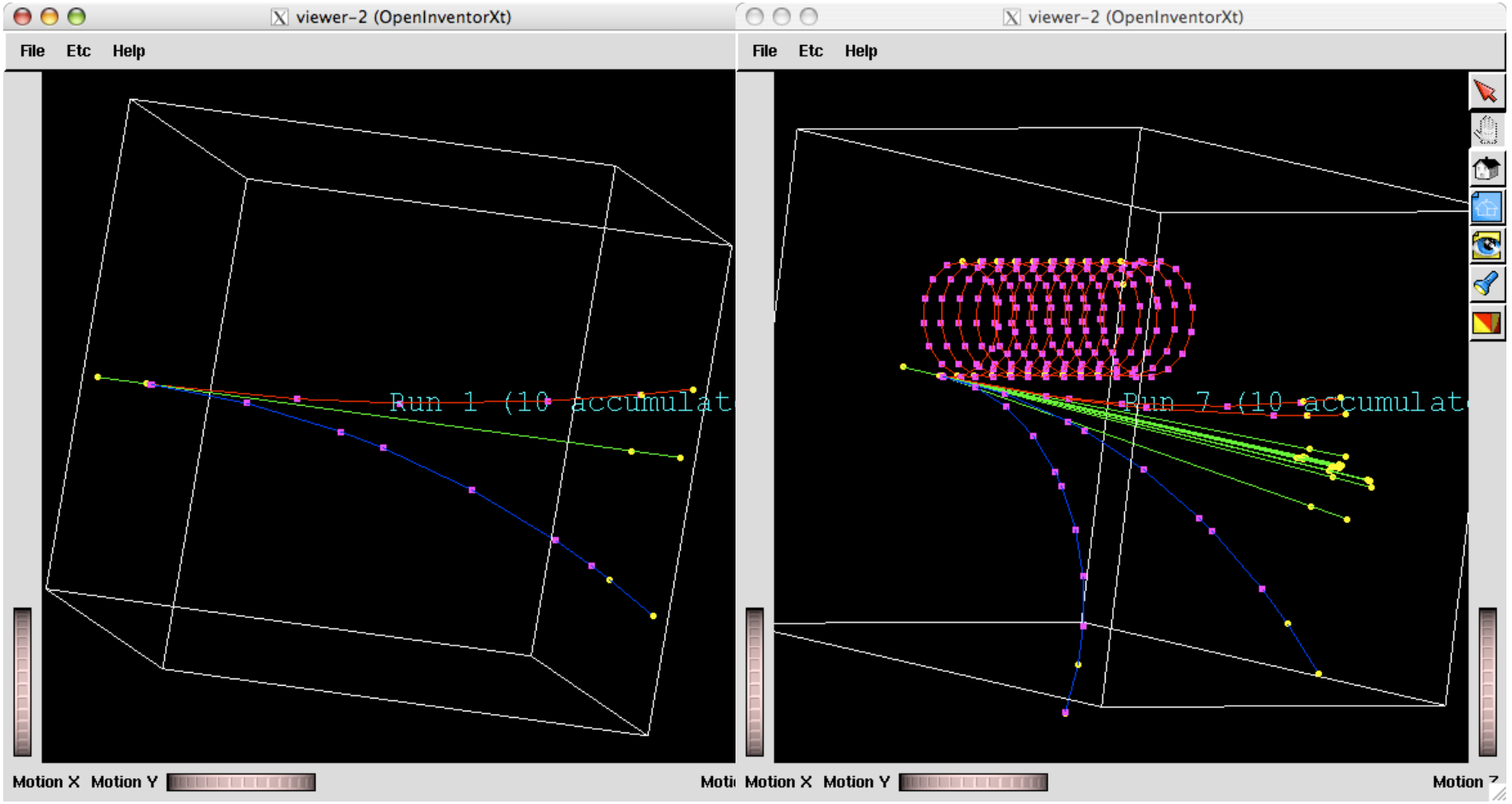
Yellow are the actual step points used by Geant4

Magenta are auxiliary points added just for purposes of visualization

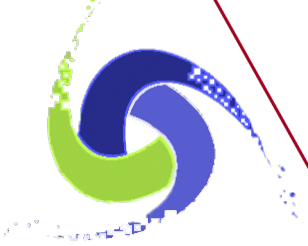




Smooth Trajectory Makes Big Difference for Trajectories that Loop in a Magnetic Field



- Yellow dots are the actual step points used by Geant4
- Magenta dots are auxiliary points added just for purposes of visualization

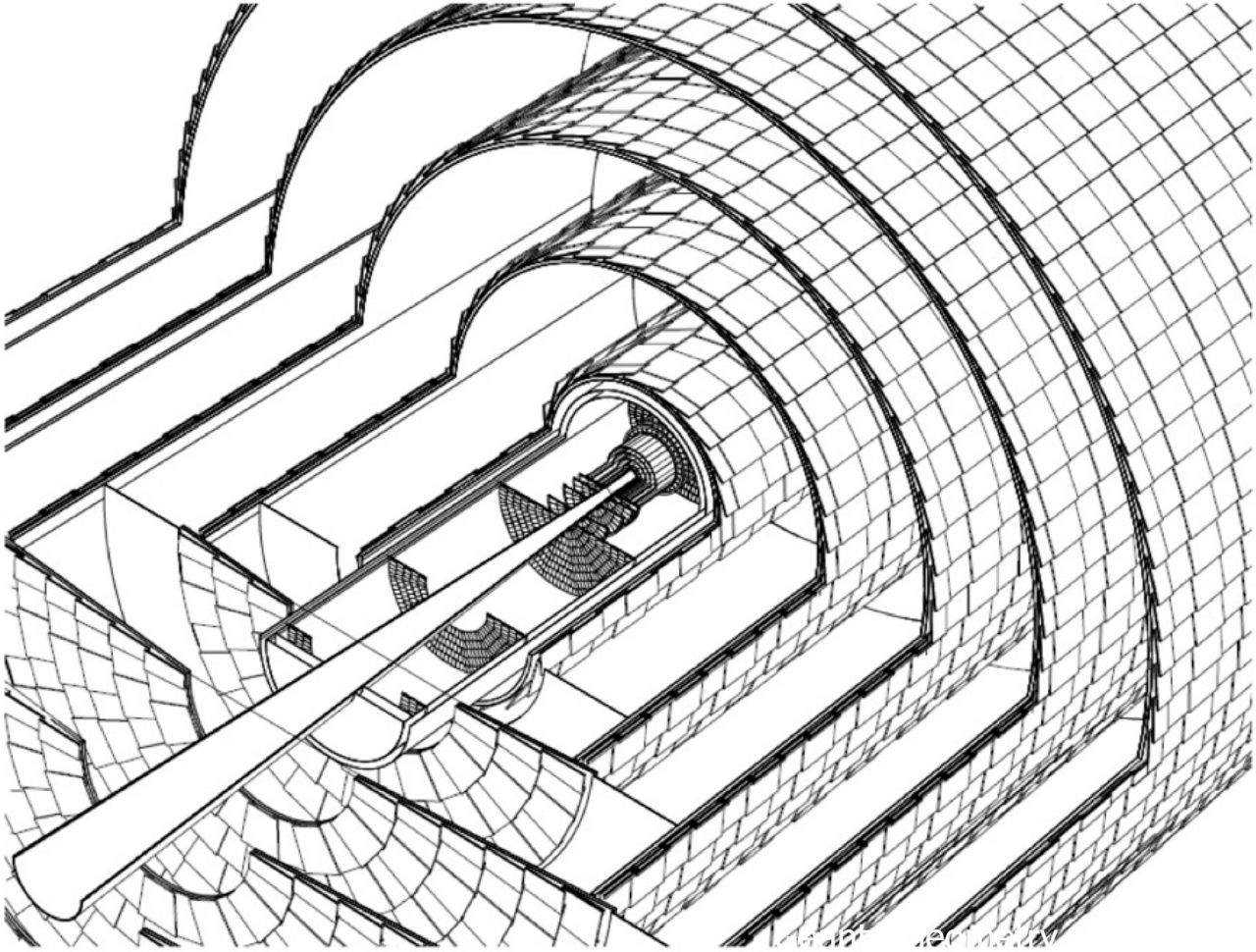


GEANT4
A SIMULATION TOOLKIT
Version 10.7.p02

GDML/CAD interfaces

GDML and CAD Interfaces

- Up to now, the course has shown how to define materials and volumes from C++.
- This part of slides shows some alternate ways to define geometry at runtime by providing a file-based detector description.



Silicon Pixel & Microstrip Tracker for Collider Detector
N. Graf, J. McCormick, LCDD, SLAC

GDML : Geometrical Description Modeling Language

- An XML-based language designed as an application independent persistent format for describing the geometries of detectors.
  - Implements “geometry trees” which correspond to the hierarchy of volumes a detector geometry can be composed of
  - Allows materials to be defined and solids to be positioned
- Because it is pure XML, GDML can be used universally
  - Not just for Geant4
  - Can be format for interchanging geometries among different applications.
  - Can be used to translate CAD geometries to Geant4
- XML is simple
  - Rigid set of rules, self-describing data validated against schema
- XML is extensible
  - Easy to add custom features, data types
- XML is Interoperable to OS’s, languages, applications
- XML has hierarchical structure
  - Appropriate for Object-Oriented programming
  - Detector/sub-detector relationships

GDML structure

- Contains numerical values of constants, positions, rotations and scales that will be used later on in the geometry construction.
  - Uses CLHEP expressions
- Constants

  <constant name="length" value="6.25"/>
- Variables

  <variable name="x" value="6"/>
  - Once defined, can be used anywhere later, e.g.

    <variable name="y" value="x/2"/>

    <box name="my_box" x="x" y="y" z="x+y"/>
- Positions

  <position name="P1" x="25.0" y="50.0" z="75.0" unit="cm"/>
- Rotations

  <rotation name="RotateZ" z="30" unit="deg"/>
- Matrices

  <matrix name="m" coldim="3" values=" 0.4 9 126
  8.5 7 21
  34.6 7 9"/>

GDML structure

- Simple Elements

```
<element Z="8" formula="O" name="Oxygen" >
    <atom value="16" />
</element>
```

- Material by number of atoms (“molecule”)

```
<material name="Water" formula="H2O">
    <D value="1.0" />
    <composite n="2" ref="Hydrogen" />
    <composite n="1" ref="Oxygen" />
</material>
```

- Material as a fractional mixture of elements or materials, (“compound”):

```
<material formula="air" name="Air" >
    <D value="0.00129" />
    <fraction n="0.7" ref="Nitrogen" />
    <fraction n="0.3" ref="Oxygen" />
</material>
```

Importing GDML file

- GDML files can be directly imported into Geant4 geometry, using the GDML plug-in facility:

  #include “G4GDMLParser.hh”

- Generally you will want to put the following lines into your DetectorConstruction class:

  G4GDMLParser parser;

  parser.Read(“geometryFile.gdml”);

  G4VphysicalVolume* W = parser.GetWorldVolume();

- To include the Geant4 module for GDML,
  - Install the XercesC parser (version 2.8.0 or 3.0.0)

    http://xerces.apache.org/xerces-c/download.cgi
  - Set appropriate environment variables when G4 libraries are built
- Examples available in:

  $G4INSTALL/examples/extended/persistency/gdml

Importing CAD geometry

- Users with 3D engineering drawings may want to incorporate these into their Geant4 simulation as directly as possible
- Difficulties include:
  - Proprietary, undocumented or changing CAD formats
  - Usually no connection between geometry and materials
  - Mismatch in level of detail required to machine a part and that required to transport particles in that part
- CAD is never as easy as you might think (if the geometry is complex enough to require CAD in the first place)
- CADMesh is a direct CAD model import interface for GEANT4 optionally leveraging VCGLIB, and ASSIMP by default. Currently it supports the import of triangular facet surface meshes defined in formats such as STL and PLY. A G4TessellatedSolid object is returned and can be included in a standard user detector constructor.
  - https://code.google.com/p/cadmesh/
- One output format most CAD programs do support is STEP
  - Not a complete solution, in particular does not contain material information
  - There are movements under way to get new formats that contain additional information, but none yet widely adopted.

CADMesh

- CADMesh is a direct CAD model import interface for GEANT4 optionally leveraging VCGLIB, and ASSIMP by default. Currently it supports the import of triangular facet surface meshes defined in formats such as STL and PLY. A G4TessellatedSolid is returned and can be included in a standard user detector construction.
  - https://code.google.com/p/cadmesh/
  - http://arxiv.org/pdf/1105.0963.pdf

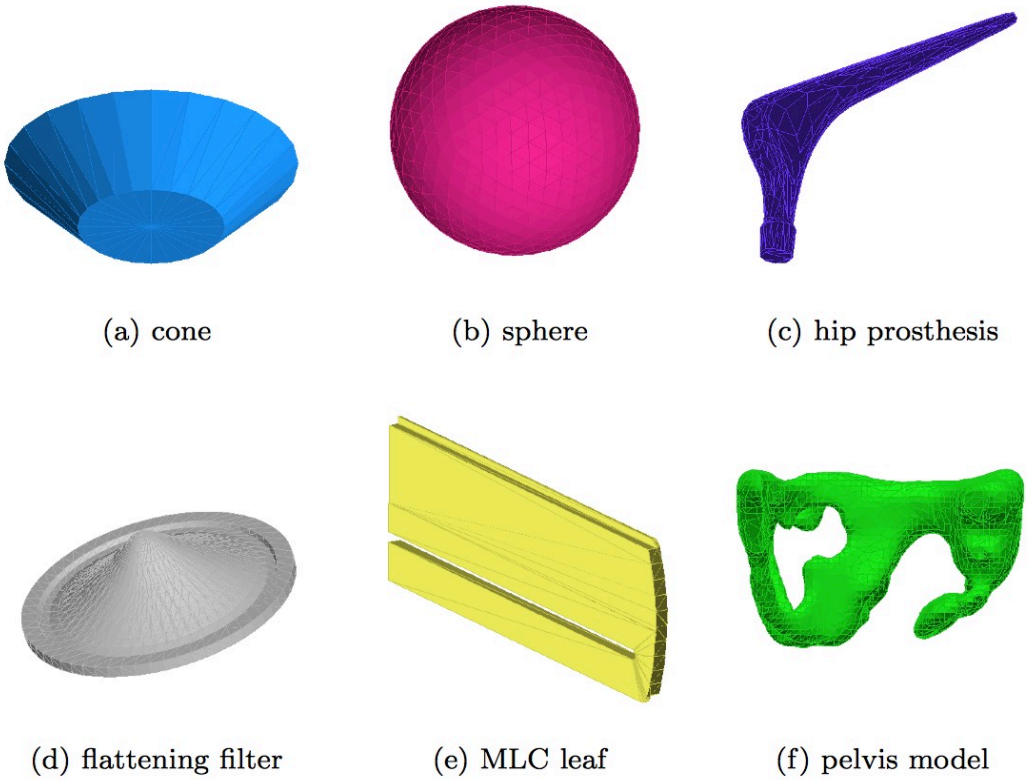
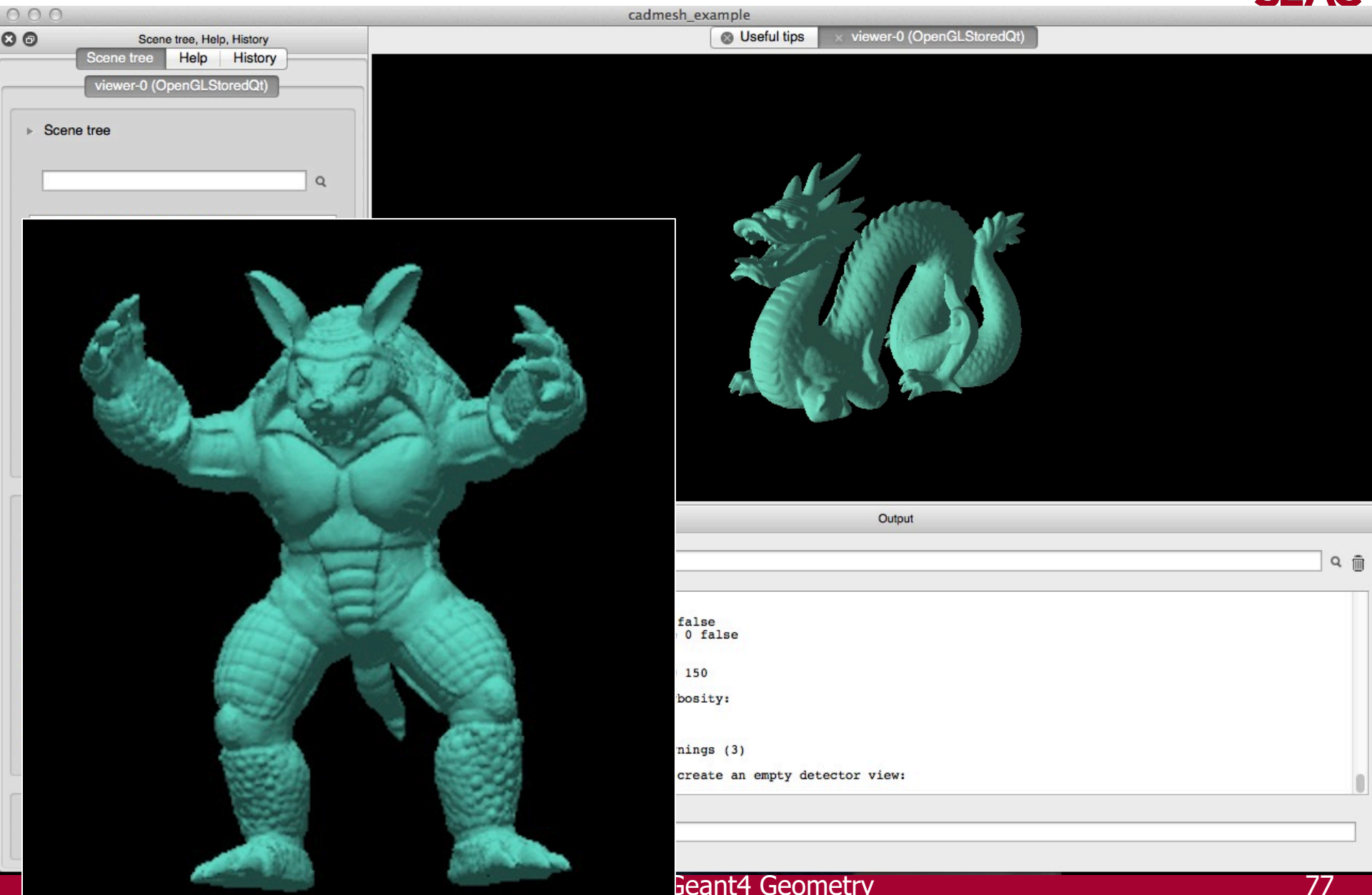


Fig. 3 Six test geometries loaded directly into GEANT4 using the proposed CAD interface and visualised using the GEANT4 OpenGL viewer.

CADMesh


cadmesh_example
Scene tree, Help, History
Scene tree
Help
History
viewer-0 (OpenGLStoredQt)
Scene tree
Useful tips
viewer-0 (OpenGLStoredQt)
Output
false
0 false
150
bosity:
nings (3)
create an empty detector view:

Loading CADMesh solid

- CADMesh provides a G4TessellatedSolid object (a kind of G4VSolid).

```
#include “CADMesh.hh”
G4VPhysicalVolume* MyDetectorConstruction::Construct()
{
  …
  CADMesh cadMesh;
  G4VSolid* aSolid = cadMesh.LoadMesh( file_name, file_type );
  G4LogicalVolume* aLV = new G4LoficalVolume( aSolid, material, “name” );
  G4VPhysicalVolume* aPV = new G4PVPlacement( 0,
                  G4ThreeVector(x,y,z), aLV, “name”, motheLV, 0, 0 );
  …
```

Importing CAD geometries: more solutions …

- InStep: supporting import/export of various formats including STL, STEP and GDML
  https://www.solveering.com/InStep/instep.aspx
- SALOME: can import STEP BREP/IGES/STEP/ACIS, mesh it, export to STL (then STL2GDML)
  http://www.salome-platform.org
- ESABASE2: space environment analysis CAD, basic modules are free for academic non-commercial use. Exports to GDML shapes or complete geometry. Imports STEP.
  https://esabase2.net
- Blender GDML exporter: GDML plug-in for Blender tool
  http://projects.blender.org/tracker/index.php?func=detail&aid=30578&group_id=153&atid=467
- FASTRAD: 3D tool for radiation shielding analysis, exports meshes to GDML
  http://www.fastrad.net
- STEP Solutions: commercial, exports meshes GDML
  http://www.steptools.com/products/stdev/
- Cogenda TCAD: for the case of 3D meshes. Module Gds2Mesh exports to GDML
  http://www.cogenda.com/article/products#VTCAD
- SW2GDML: converts SolidWorks descriptions (using its API) to real primitives in GDML, including material
  https://github.com/cvuosalo/SW2GDMLconverter
- CadMC: tool to convert FreeCAD geometry to Geant4 (tessellated and CGS shapes)
  http://polar.psi.ch/cadmc/
- McCAD tool: 'integrated approach' by KIT group using half-space solids extensions to Geant4
  http://indico.cern.ch/event/400576/contributions/1841503/attachments/802327/1099598/CERN_Visit_YQIU_V0.2.pdf
- EDGE: CAD with direct editing/importing/exporting GDML with material definition
  http://www.space-suite.com/
- MRADSIM: CAD with direct editing/importing/exporting GDML with material definition
  https://www.mradsim.com/mradsim-converter/