Geometry II

Makoto Asai (SLAC)

Geant4 Tutorial Course
Contents

- Various ways of placement
  - Simple placement volume
  - Parameterized volume
  - Replicated volume
  - Nested-parameterization volume
  - Divided volume
  - Reflected volume
  - Assembly volume

- Detail is given in later talk.
Define 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

```cpp
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 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
Physical volume - 1

- **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 visualization attributes can be parameterized by the copy numbers of ancestors.
Physical volume - 2

- **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**  
  utility for a pair of Placement volumes
  - generating placements of a volume and its reflected volume
  - Useful typically for end-cap calorimeter

- **G4AssemblyVolume**  
  utility for a set of Placement volumes
  - Position a group of volumes
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.
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.
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/](http://erpc1.naruto-u.ac.jp/~geant4/)
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

Geometry II - M.Asai (SLAC) 15
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);
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
};
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 = material11;
        physVol->GetLogicalVolume()->SetVisAttributes( att1 );
    }
    ...
    return mat;
}
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
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 
    
    $$ -width \cdot (nReplicas-1) \cdot 0.5 + n \cdot width $$
  - Offset shall not be used

- **Radial axis** - $kRaxis$
  - Center of $n$-th daughter is given as 
    
    $$ width \cdot (n+0.5) + offset $$
  - Offset must be the inner radius of the mother

- **Phi axis** - $kPhi$
  - Center of $n$-th daughter is given as 
    
    $$ width \cdot (n+0.5) + offset $$
  - 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);
Nested parameterization
Nested parameterization

- Suppose your geometry has three-dimensional regular reputation of same shape and size of volumes without gap between volumes. And material of such volumes are changing according to the position.
  - E.g. voxels made by CT Scan data (DICOM)
  - Instead of direct three-dimensional parameterized volume, use replicas for the first and second axes sequentially, and then use one-dimensional parameterization along the third axis.

- It requires much less memory for geometry optimization and gives much faster navigation for ultra-large number of voxels.
Given geometry is defined as two sequential replicas and then one-dimensional parameterization,

- Material of a voxel must be parameterized not only by the copy number of the voxel, but also by the copy numbers of ancestors.
- Material is indexed by three indices.
- `G4VNestedParameterisation` is a special parameterization class derived from `G4VPVParameterisation` base class.
  - `ComputeMaterial()` method of `G4VNestedParameterisation` has a touchable object of the parent physical volume, in addition to the copy number of the voxel.
    - Index of first axis = `theTouchable->GetCopyNumber(1);`
    - Index of second axis = `theTouchable->GetCopyNumber(0);`
    - Index of third axis = copy number
**G4VNestedParameterisation**

- G4VNestedParameterisation is derived from G4VPVParameterization.
- G4VNestedParameterisation class has three **pure virtual** methods you have to implement,
  - in addition to ComputeTransformation() method, which is mandatory for all G4VPVParameterization classes.

```cpp
virtual G4Material* ComputeMaterial(G4VPhysicalVolume *currentVol, const G4int repNo, const G4VTouchable *parentTouch=0)=0;
```
- Return a material pointer w.r.t. copy numbers of itself and ancestors.
- Must cope with parentTouch=0 for navigator's sake. Typically, return a default material if parentTouch=0.

```cpp
virtual G4int GetNumberOfMaterials() const=0;
```
- Return total number of materials which may appear as the return value of ComputeMaterial() method.

```cpp
virtual G4Material* GetMaterial(G4int idx) const=0;
```
- Return idx-th material.
- “idx” is not a copy number. idx = [0, nMaterial-1]
G4VNestedParameterisation

- G4VNestedParameterisation is a kind of G4VPVParameterization.
  - It can be used as an argument of G4PVParameterised.
  - All other arguments of G4PVParameterised are unaffected.
- Nested parameterization of placement volume is **not** supported.
  - All levels used as indices of material must be **repeated volume**.
    There cannot be a level of placement volume in between.
Divided volume
G4PVDivision

- G4PVDivision is a special kind of G4PVParameterised.
  - G4PVParameterisation is automatically generated according to the parameters given in G4PVDivision.
- G4PVDivision is similar to G4PVReplica but
  - It currently allows gaps in between mother and daughter volumes
  - We are extending G4PVDivision to allow gaps between daughters, and also gaps on side walls. We plan to release this extension in near future.
- Shape of all daughter volumes must be same shape as the mother volume.
  - G4VSolid (to be assigned to the daughter logical volume) must be the same type, but different object.
- Replication must be aligned along one axis.
- If your geometry does not have gaps, use G4Replica.
  - For identical geometry, navigation of G4Replica is faster.
G4PVDivision(const G4String& pName,
    G4LogicalVolume* pDaughterLogical,
    G4LogicalVolume* pMotherLogical,
    const EAxis pAxis,
    const G4int nDivisions, // number of division is given
    const G4double offset);

- The size (width) of the daughter volume is calculated as
  \[(\text{size of mother}) - \text{offset}) / \text{nDivisions}\]
The number of daughter volumes is calculated as

\[
\text{int}\left( \frac{\text{(size of mother) - offset}}{\text{width}} \right)
\]

As many daughters as width and offset allow
G4PVDivision - 3

G4PVDivision(const G4String& pName,
             G4LogicalVolume* pDaughterLogical,
             G4LogicalVolume* pMotherLogical,
             const EAxis pAxis,
             const G4int nDivisions,
             const G4double width,    // both number of division and width are given
             const G4double offset);

- \textit{nDivisions} daughters of \textit{width} thickness