Geometry 4

I.Hrivnacova
IPN, Orsay

Most slides thanks to M. Asai, SLAC

Cours Geant4 @ Paris 2007
4 - 8 June 2007

Contents

- Advanced ways of placement
  - Divisions
  - Assembly volumes
  - Reflected volumes
- Geometry optimization

Contents

Advanced ways of placement

Divisions

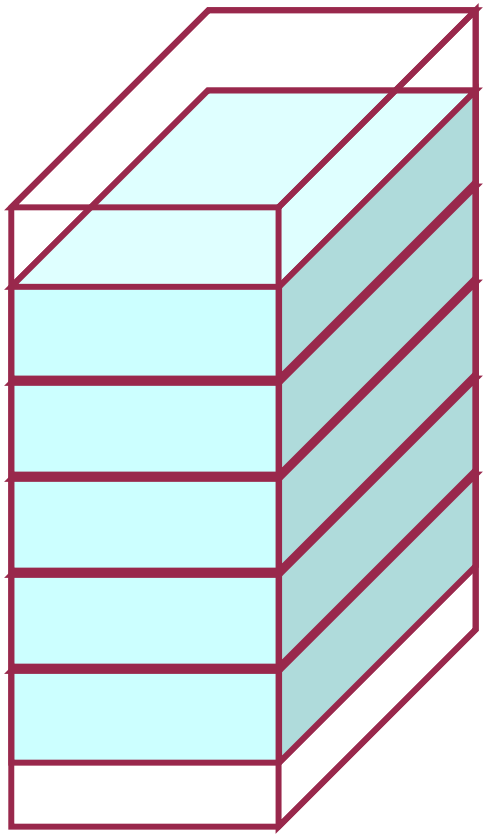
Assembly volumes

Reflected volumes

Geometry optimization

Divisions

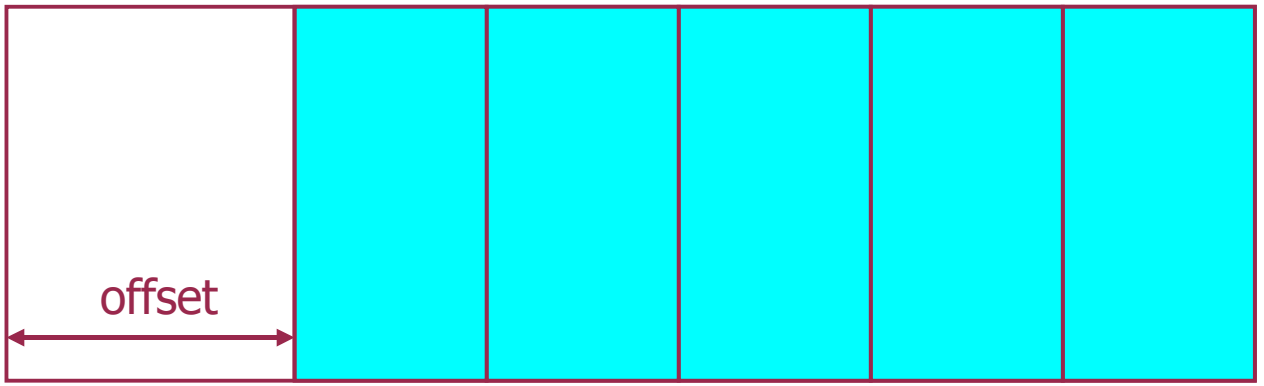
- G4PVDivision is a special kind of G4PVParameterised.
  - G4VPVParameterisation is automatically generated according to the parameters given in G4PVDivision.
- G4PVDivision is similar to G4PVReplica but
  - It currently allows gaps in between mother and daughter volumes
  - We are extending G4PVDivision to allow gaps between daughters, and also gaps on side walls. We plan to release this extension at version 9.0.
- *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 (1)

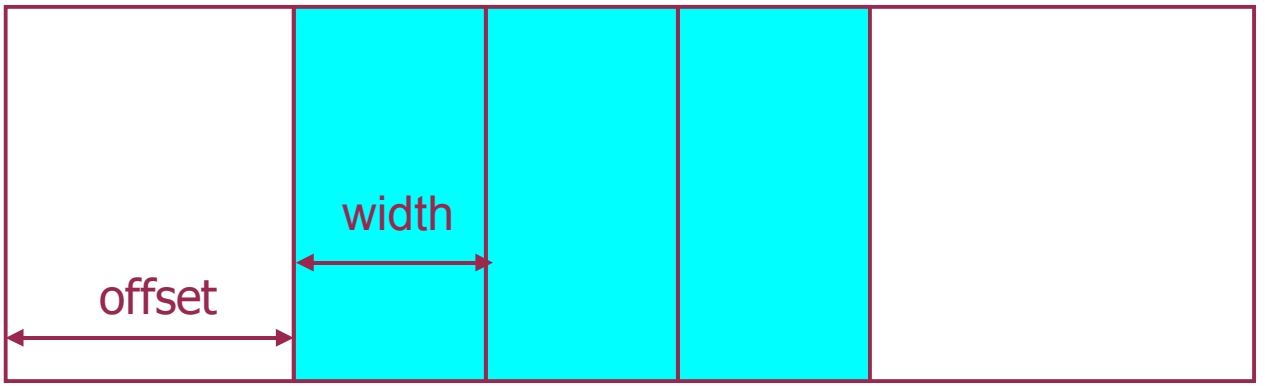
- ```
  G4PVDivision(const G4String& name,
               G4LogicalVolume* daughterLogical,
               G4LogicalVolume* motherLogical,
               const EAxis axis,
               const G4int nofDivisions,  // number of division is given
               const G4double offset = 0.);
  ```
- The size (width) of the daughter volume is calculated as:

  `( (size of mother) - offset ) / nDivisions`


  ```

G4PVDivision (2)

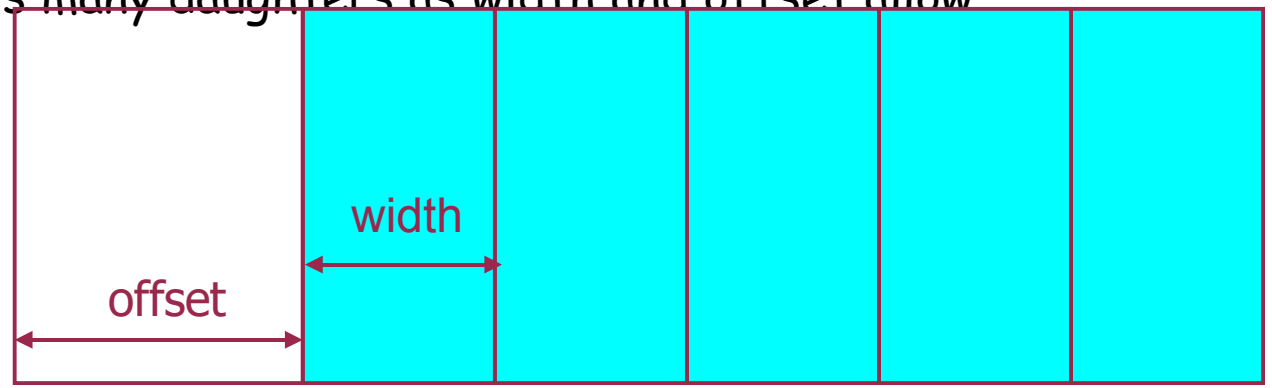
- ```
  G4PVDivision(const G4String& name,
               G4LogicalVolume* daughterLogical,
               G4LogicalVolume* motherLogical,
               const EAxis axis,
               const G4int nofDivisions,
               const G4double width, // both number of division and width
                                     // are given
               const G4double offset = 0.);
  ```
- ***nofDivisions*** daughters of ***width*** thickness


  ```

G4PVDivision (3)

- ```
  G4PVDivision(const G4String& name,
               G4LogicalVolume* daughterLogical,
               G4LogicalVolume* motherLogical,
               const EAxis axis,
               const G4double width, // width of daughter volume is given
               const G4double offset = 0.);
  ```
- The number of daughter volumes is calculated as

  `int( ( (size of mother) - offset ) / width )`
  - As many daughters as width and offset allow


  ```

G4PVDivision
Supported Cases (1)

G4PVDivision currently supports following shapes / axes.

- CSG solids:
  - G4Box : **kXAxis, kYAxis, kZAxis**
  - G4Tubs : **kRho, kPhi, kZAxis**
  - G4Cons : **kRho, kPhi, kZAxis**
  - G4Trd : **kXAxis, kYAxis, kZAxis**
  - G4Para : **kXAxis, kYAxis, kZAxis**

Division
Supported Cases (2)

- Specific solids:
  - G4Polycone : **kRho, kPhi, kZAxis**
    - **kZAxis** - the number of divisions has to be the same as solid sections, (i.e. **numZPlanes-1**), the width will not be taken into account.
  - G4Polyhedra : **kRho, kPhi, kZAxis**
    - **kPhi** - the number of divisions has to be the same as solid sides, (i.e. **numSides**), the width will not be taken into account.
    - **kZAxis** - the number of divisions has to be the same as solid sections, (i.e. **numZPlanes-1**), the width will not be taken into account.
- In the case of division along **kRho** of G4Cons, G4Polycone, G4Polyhedra, if width is provided, it is taken as the width at the -Z radius; the width at other radii will be scaled to this one.

Contents

Advanced ways of placement

Divisions

Assembly volumes

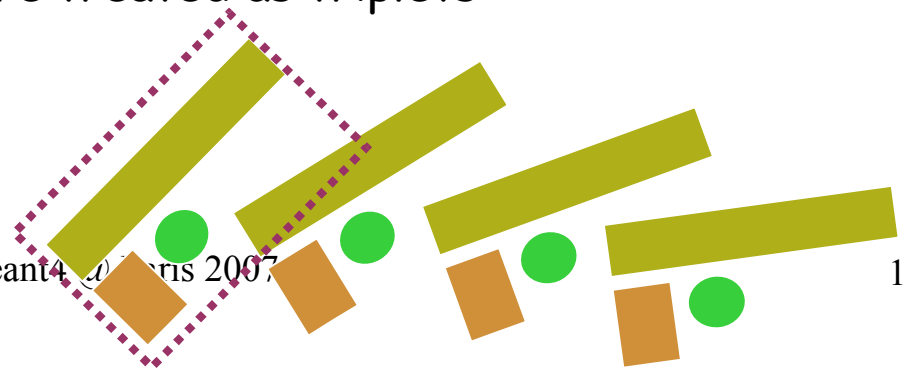
Reflected volumes

Geometry optimization

Assembly Volumes

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. envelope
  - Rotation w.r.t. envelope



Assembly Volumes

G4Assembly Volume

- Helper class to combine daughter logical volumes in arbitrary way
- **G4AssemblyVolume::AddPlacedVolume (**

```
G4LogicalVolume* volume,
G4ThreeVector& translation, G4RotationMatrix* rotation );
```

  - Adds a volume in the assembly with a given placement
- **G4AssemblyVolume::AddPlacedAssembly (**

```
G4AssemblyVolume* volume,
G4ThreeVector& translation, G4RotationMatrix* rotation );
```

  - The daughter of the assembly can be also an assembly of volumes

Assembly Volumes
G4Assembly Volume

- Imprints of the assembly volume are made inside a mother logical volume through:

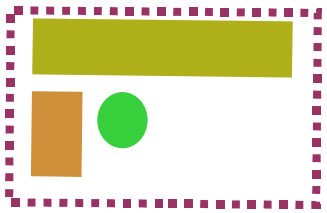
```
G4AssemblyVolume::MakeImprint (
      G4LogicalVolume* motherVolume,
      G4ThreeVector& translation, G4RotationMatrix* rotation );
```

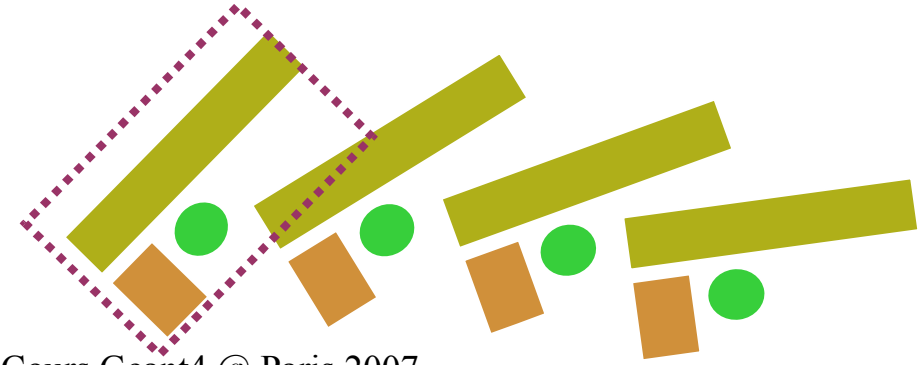
  - Each physical volume name is generated automatically
    - Format: **av_WWW_impr_XXX_YYY_ZZZ**
      - **WWW** - assembly volume instance number
      - **XXX** - assembly volume imprint number
      - **YYY** - name of the placed logical volume in the assembly
      - **ZZZ** - index of the associated logical volume
  - Generated physical volumes (and related transformations) are automatically managed (creation and destruction)

Assembly Volumes
Example

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

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





Contents

Advanced ways of placement

Divisions

Assembly volumes

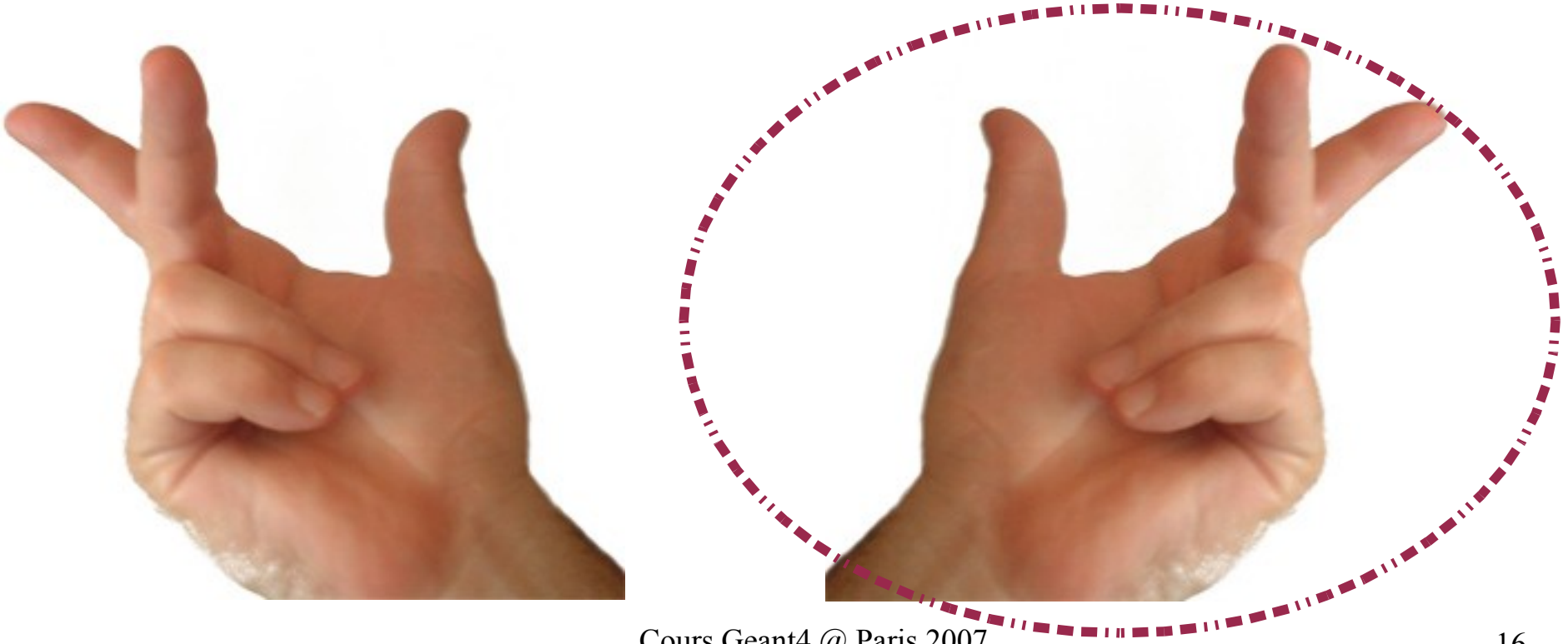
Reflected volumes

Geometry optimization

Reflected Volumes

Reflecting Solids

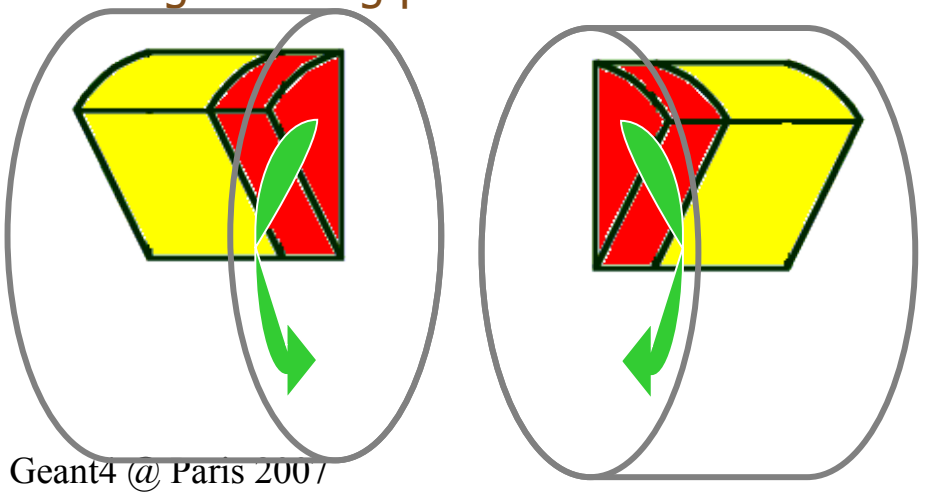
- Let's take as an example a human hand
  - In a mirror the right hand becomes a left hand
  - But we cannot make a left hand from the right one by a simple 180 degree rotation or a translation
- The hand in a mirror is not the same 'solid' as the hand before



Reflected Volumes

Reflecting Solids

- Hierarchies of volumes based on CSG or specific solids can be reflected by means of the reflection factory and reflected solid classes
- G4ReflectedSolid (derived from G4VSolid)
  - Utility class representing a solid shifted from its original reference frame to a new mirror symmetric one
    - The right hand -> the left hand
  - Once the reflection (G4Reflect[X/Y/Z]3D) is applied to solid, the new solid can be placed with a transformation made by a translation and rotation only
- G4ReflectionFactory
  - Singleton object using G4ReflectedSolid for generating placements of reflected volumes
- Reflections are limited to simple CSG solids and specific solids



Reflected Volumes

Reflection Factory

- When reflecting hierarchies of volume, the reflection factory creates for each solid and logical volume its reflected counterpart
- When placing a volume (or volume tree) in a geometry containing already reflected volumes, it is important to use constantly G4ReflectionFactory, as it guarantees that the placement will occur also in a reflected counterpart of the mother logical volume
- Reflection factory methods for placements:
  - `G4PhysicalVolumesPair G4ReflectionFactory::Place (..);`
  - `G4PhysicalVolumesPair G4ReflectionFactory::Replicate (..);`
  - `G4PhysicalVolumesPair G4ReflectionFactory::Divide (..);`
  - Reflection of generic parameterised volumes is not possible yet.
- All return a pair of physical volumes, the second being a placement in the reflected mother, if the mother volume has its reflected counterpart:
  - `G4PhysicalVolumesPair is std::pair<G4VPhysicalVolume*,G4VPhysicalVolume*>`

Reflecting hierarchies of volumes (1)

```
G4PhysicalVolumesPair G4ReflectionFactory::Place (
  const G4Transform3D& transform3D, // the transformation
  const G4String&  name,            // the name
  G4LogicalVolume* LV,              // the logical volume
  G4LogicalVolume* motherLV,        // the mother volume
  G4bool           noBool,          // currently unused
  G4int            copyNo           // optional copy number );
```

Used for normal (simple) placements:

1) Performs the transformation decomposition
2) Generates a new reflected solid and logical volume, or retrieves it from a map if the reflected object is already created
3) Transforms all daughters and places them in the given mother
4) If motherLV has its reflected counterpart, create a second placement of this volume in the reflected mother

Reflecting hierarchies of volumes (2)

For replicated volumes:

```
G4PhysicalVolumesPair G4ReflectionFactory::Replicate (
   const G4String&  name,       // the actual name
   G4LogicalVolume* LV,         // the logical volume
   G4LogicalVolume* motherLV,   // the mother volume
   Eaxis            axis,       // axis of replication
   G4int            replicaNo, // number of replicas
   G4int            width,      // width of single replica
   G4int            offset=0    // optional mother offset );
```

1) Creates replicas in the given mother volume
2) If motherLV has its reflected counterpart, create a second placement of this volume in this reflected mother

Reflecting hierarchies of volumes (3)

For divided volumes:

- First it is necessary to explicitly instantiate a concrete division factory -before- applying the actual reflection:
  `G4PVDivisionFactory::GetInstance();`
- `G4PhysicalVolumesPair G4ReflectionFactory::Divide (`

```
const G4String&  name,         // the actual name
G4LogicalVolume* LV,           // the logical volume
G4LogicalVolume* motherLV,     // the mother volume
Eaxis            axis          // axis of division
G4int            nofDivisions // number of division
G4int            width,        // width of single division
G4int            offset=0      // optional mother offset
```

1) This creates division in the given mother volume
2) If motherLV has its reflected counterpart, create a second placement of this volume in this reflected mother

Contents

Advanced ways of placement

Divisions

Assembly volumes

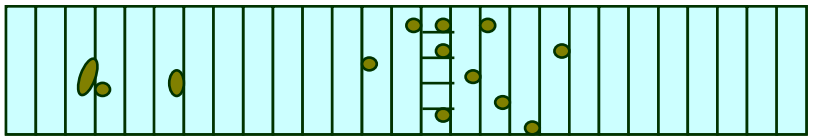
Reflected volumes

Geometry optimization

Geometry Optimisation

Smart Voxelization

- In case of Geant 3.21, the user had to carefully implement his/her geometry to maximize the performance of geometrical navigation.
- While in Geant4, user's geometry is automatically optimized to most suitable to the navigation. - "Voxelization"
  - For each mother volume, one-dimensional virtual division is performed.
  - Subdivisions (slices) containing same volumes are gathered into one.
  - Additional division again using second and/or third Cartesian axes, if needed.
- *"Smart voxels"* are computed at initialisation time
  - When the detector geometry is *closed*
  - Does not require large memory or computing resources
  - At tracking time, searching is done in a hierarchy of virtual divisions



Geometry Optimisation

Detector description tuning

- Some geometry topologies may require 'special' tuning for ideal and efficient optimisation
  - for example: a dense nucleus of volumes included in very large mother volume
- Granularity of voxelization can be explicitly set to logical volume via
  - its method: `SetSmartless(G4double);`
- Critical regions for optimisation can be detected
  - Helper class `G4SmartVoxelStat` for monitoring time spent in detector geometry optimisation
    - Automatically activated if `/run/verbose` greater than 1

```
 Percent      Memory      Heads     Nodes    Pointers     Total CPU     Volume
-------      ------     -----     -----   --------     ---------     -----------
  91.70          1k         1        50         50          0.00     Calorimeter
   8.30          0k         1         3          4          0.00     Layer
```

Visualising voxel structure

- The computed voxel structure can be visualized with the final detector geometry
  - Helper class `G4DrawVoxels`
  - Visualize voxels given a logical volume

    `G4DrawVoxels::DrawVoxels(const G4LogicalVolume*)`
  - Allows setting of visualization attributes for voxels

    `G4DrawVoxels::SetVoxelsVisAttributes(…)`