Nested Parameterization

Makoto Asai (SLAC)
4th Geant4 Space Users’ Workshop
Contents

- Smart voxelization for geometry optimization
- Nested parameterization
- Scorer for nested parameterization
- Moving objects
Geometry optimization
("voxelization")
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.
    - so that each division (voxel) has 2 (default) daughters at most
  - "Smart voxels" are computed at initialization 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
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 voxelisation can be explicitly set
  - Methods `Set/GetSmartless()` from `G4LogicalVolume`
- 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

<table>
<thead>
<tr>
<th>Percent</th>
<th>Memory</th>
<th>Heads</th>
<th>Nodes</th>
<th>Pointers</th>
<th>Total CPU</th>
<th>Volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>91.70</td>
<td>1k</td>
<td>1</td>
<td>50</td>
<td>50</td>
<td>0.00</td>
<td>Calorimeter</td>
</tr>
<tr>
<td>8.30</td>
<td>0k</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>0.00</td>
<td>Layer</td>
</tr>
</tbody>
</table>
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(...)
    ```
  - useful for debugging purposes
Optimization of 3-D parameterized volume

- 3-D parameterized volume has two options of smart voxelization, chosen by EAxis parameter of G4PVParameterised
  - 1-D voxelization (EAxis == kXAxis, kYAxis, ...)
    - Compact memory size but slow
  - 3-D voxelization (EAxis == kUndefined)
    - Fast but huge memory size
- Use replicas for the first and second axes slices and 1-D parameterization for the third axis. Use G4NestedParameterisation to parameterize the material.
  - The material of a voxel should be defined as a function not only of the copy number of the parameterization, but also of the copy numbers of its mother and grand-mother replicas.
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 for CT Scan data (DICOM), Semiconductor, etc.
- 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.
Nested parameterization

- 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 G4VPVPParameterisation 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.
Scorer for
nested parameterization
Keys of G4THitsMap

- All provided primitive scorer classes use `G4THitsMap<G4double>`.
- By default, the copy number is taken from the physical volume to which `G4MultiFunctionalDetector` is assigned.
  - If the physical volume is placed only once, but its (grand-)mother volume is replicated, use the second argument of the constructor of the primitive scorer to indicate the level where the copy number should be taken.
    
    e.g. `G4PSCellFlux(G4Steing name, G4int depth=0)`

- If your indexing scheme is more complicated (e.g. utilizing copy numbers of more than one hierarchies), you can override the virtual method `GetIndex()` provided for all the primitive scorers.
Keys for nested parameterization

- For 3-D nested parameterization volume, the unique ID to be used as the key of G4THitsMap should be the combination of three indices.
  - GetIndex() method of the provided scorer must be overwritten.

```cpp
class MyPSDoseScorer : public G4PSDoseScorer
{
  public:
    virtual G4int GetIndex(G4Step* theStep);
};

G4int MyPSDoseScorer::GetIndex(G4Step* theStep)
{
  G4StepPoint* preStep = thestep->GetPreStepPoint();
  G4TouchableHistory* th
    = (G4TouchableHistory*)(preStep->GetTouchable());
  G4int idx = th->GetReplicaNumber(0) + 1000 * th->GetReplicaNumber(1) + 1000000 * th->GetReplicaNumber(2);
  return idx;
}
```

- parameterized volume
- mother replica
- grand-mother replica
Moving objects
Moving objects

- In some applications, it is essential to simulate the movement of some volumes.
  - E.g. particle therapy simulation
- Geant4 can deal with moving volume
  - In case speed of the moving volume is slow enough compared to speed of elementary particles, so that you can assume the position of moving volume is still within one event.
- Two tips to simulate moving objects:
  1. Use parameterized volume to represent the moving volume.
  2. Do not optimize (voxelize) the mother volume of the moving volume(s).
Moving objects - tip 1

- Use parameterized volume to represent the moving volume.
  - Use event number as a time stamp and calculate position/rotation of the volume as a function of event number.

```cpp
void MyMovingVolumeParameterisation::ComputeTransformation(const G4int copyNo, G4VPhysicalVolume *physVol) const {
  static G4RotationMatrix rMat;
  G4int eID = 0;
  const G4Event* evt = G4RunManager::GetRunManager()->GetCurrentEvent();
  if(evt) eID = evt->GetEventID();
  G4double t = 0.1*s*eID;
  G4double r = rotSpeed*t;
  G4double z = velocity*t+orig;
  while(z>0.*m) {z-=8.*m;}
  rMat.set(HepRotationX(-r));
  physVol->SetTranslation(G4ThreeVector(0.,0.,z));
  physVol->SetRotation(&rMat0);
}
```

Null pointer must be protected. This method is also invoked while geometry is being closed at the beginning of run, i.e. event loop has not yet began. You are responsible not to make the moving volume get out of (protrude from) the mother volume.

Here, event number is converted to time.
(0.1 sec/event)

Position and rotation are set as the function of event number.
Moving objects - tip 2

- Do not optimize (voxelize) the mother volume of the moving volume(s).
  - If moving volume gets out of the original optimized voxel, the navigator gets lost.

motherLogical -> SetSmartless( number_of_daughters_or_larger );

- With this method invocation, the one-and-only optimized voxel has all daughter volumes.
- For the best performance, use hierarchal geometry so that each mother volume has least number of daughters.

- If you are interested in, you can download a sample program
  http://www.slac.stanford.edu/~asai/Rot.tar.gz