Kernel III

Makoto Asai (SLAC)

Geant4 Tutorial Course

Geant 4

Contents

- Parallel geometry (New feature of v9.0)
- Moving objects
- Fast simulation (Shower parameterization)
- Tips for simulating huge number of 3D voxels
- Tips for computing performance

Parallel geometry

Note :
New feature with
Geant4 version 9.0
(June 29th, 2007)

Parallel navigation

- In the previous versions, we have already had several ways of utilizing a concept of parallel world. But the usages are quite different to each other.
  - Ghost volume for shower parameterization assigned to G4GlobalFastSimulationManager
  - Readout geometry assigned to G4VSensitiveDetector
  - Importance field geometry for geometry importance biasing assigned to importance biasing process
  - Scoring geometry assigned to scoring process
- We merge all of them into common parallel world scheme.
  - Readout geometry for sensitive detector will be kept for backward compatibility.
  - Other current “parallel world schemes” will become obsolete.

Parallel navigation

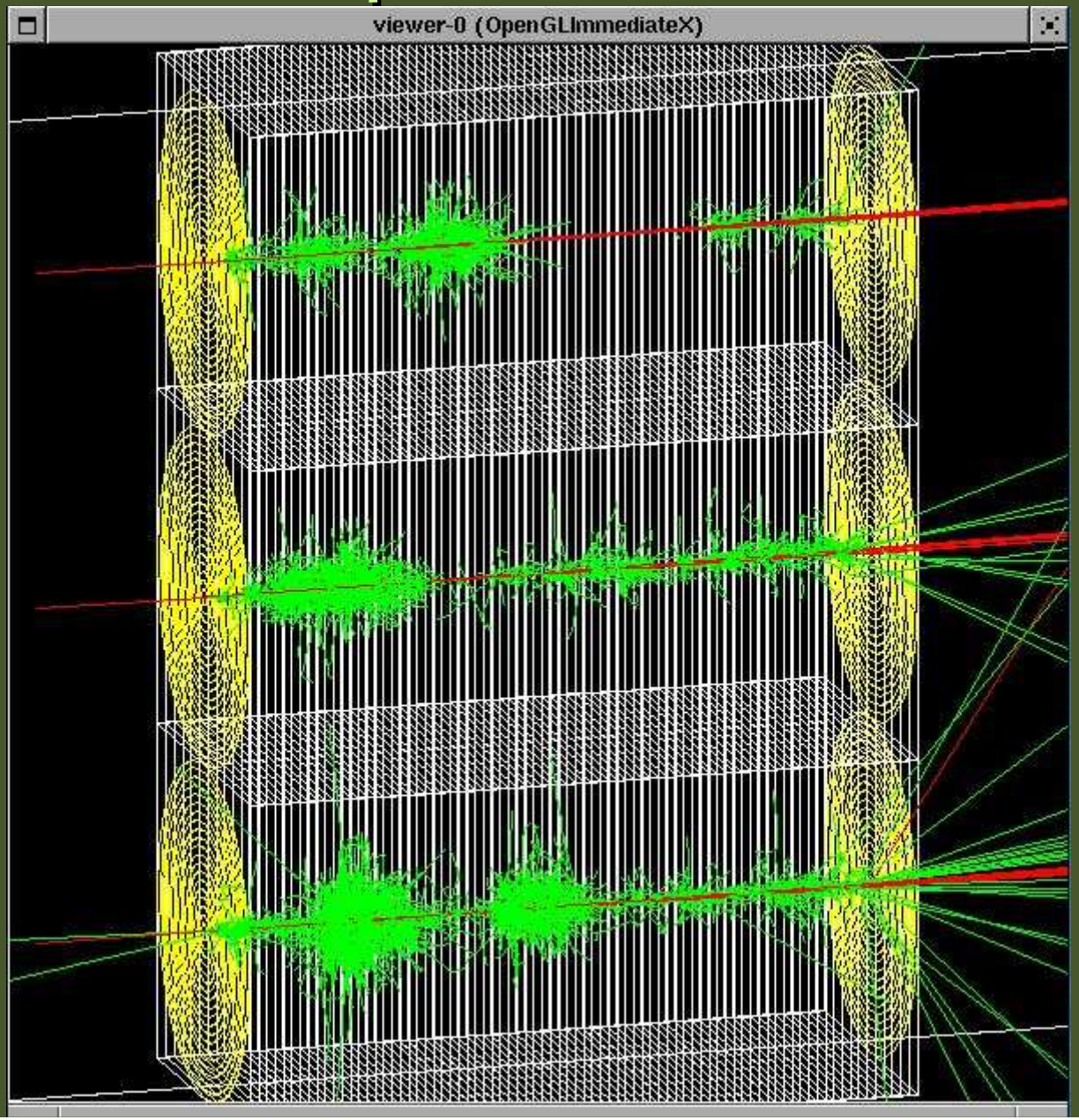
- Occasionally, it is not straightforward to define sensitivity, importance or envelope to be assigned to volumes in the mass geometry.
  - Typically a geometry built machinery by CAD, GDML, DICOM, etc. has this difficulty.
- New parallel navigation functionality allows the user to define more than one worlds simultaneously.
  - New G4Transportation process sees all worlds simultaneously.
  - A step is limited not only by the boundary of the mass geometry but also by the boundaries of parallel geometries.
  - Materials, production thresholds and EM field are used only from the mass geometry.
  - In a parallel world, the user can define volumes in arbitrary manner with sensitivity, regions with shower parameterization, and/or importance field for biasing.
    - Volumes in different worlds may overlap.

Parallel navigation

- G4VUserParrallelWorld is the new base class where the user implements a parallel world.
  - The world physical volume of the parallel world is provided by G4RunManager as a clone of the mass geometry.
  - All UserParallelWorlds must be registered to UserDetectorConstruction.
  - Each parallel world has its dedicated G4Navigator object, that is automatically assigned when it is constructed.
- Though all worlds will be comprehensively taken care by G4Transportation process for their navigations, each parallel world must have its own process to achieve its purpose.
  - For example, in case the user defines a sensitive detector to a parallel world, a process dedicated to this world is responsible to invoke this detector. G4SteppingManager sees only the detectors in the mass geometry. The user has to have G4ParallelWorldScoringProcess in his physics list.

New exampleN07

- Mass geometry
  - sandwich of rectangular absorbers and scintilators
- Parallel scoring geometry
  - Cylindrical layers



Defining a parallel world

main() (exampleN07.cc)

```
G4VUserDetectorConstruction* geom = new ExN07DetectorConstruction;
G4VUserParallelWorld* parallelGeom
        = new ExN07ParallelWorld("ParallelScoringWorld");
geom->RegisterParallelWorld(parallelGeom);
runManager->SetUserInitialization(geom);
```

- The name defined in the G4VUserParallelWorld constructor is used as the physical volume name of the parallel world, and must be used for G4ParallelWorldScoringProcess (next slide).

void ExN07ParallelWorld::Construct()

```
G4VPhysicalVolume* ghostWorld = GetWorld();
G4LogicalVolume* worldLogical = ghostWorld->GetLogicalVolume();
```

- The world physical volume (“ghostWorld”) is provided as a clone of the world volume of the mass geometry. The user cannot create it.
- You can fill contents regardless of the volumes in the mass geometry.
- Logical volumes in a parallel world needs not to have a material.

G4ParallelWorldScoringProcess

```
void ExN07PhysicsList::ConstructProcess()
{
  AddTransportation();
  ConstructParallelScoring();
  ConstructEM();
}

void ExN07PhysicsList::ConstructParallelScoring()
{
  G4ParallelWorldScoringProcess* theParallelWorldScoringProcess
      = new G4ParallelWorldScoringProcess("ParaWorldScoringProc");
  theParallelWorldScoringProcess->SetParallelWorld("ParallelScoringWorld");
  theParticleIterator->reset();
  while( (*theParticleIterator)() ){
    G4ProcessManager* pmanager = theParticleIterator->value()->GetProcessManager();
    pmanager->AddProcess(theParallelWorldScoringProcess);
    pmanager->SetProcessOrderingToLast(theParallelWorldScoringProcess, idxAtRest);
    pmanager->SetProcessOrdering(theParallelWorldScoringProcess, idxAlongStep, 1);
    pmanager->SetProcessOrderingToLast(theParallelWorldScoringProcess, idxPostStep);
  }
}
```

G4ParallelWorldScoringProcess must be defined after G4Transportation but prior to any EM processes.

Name of the parallel world defined by G4VUserParallelWorld constructor

AlongStep must be 1, while AtRest and PostStep must be last

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.

```
void MyMovingVolumeParameterisation::ComputeTransformation
    (const G4int copyNo, G4VPhysicalVolume *physVol) const
{
  static G4RotationMatrix rMat;
  G4int eID = 0;
  const G4Event* evt = G4RunManager::G
  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(G4ThreeVe
  physVol->SetRotation(&rMat0);
}
```

Null pointer must be protected.
This method is also invoked while
at

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

You are responsible not to make the moving volume get out of (protrude from) t

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 );
```

  - 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

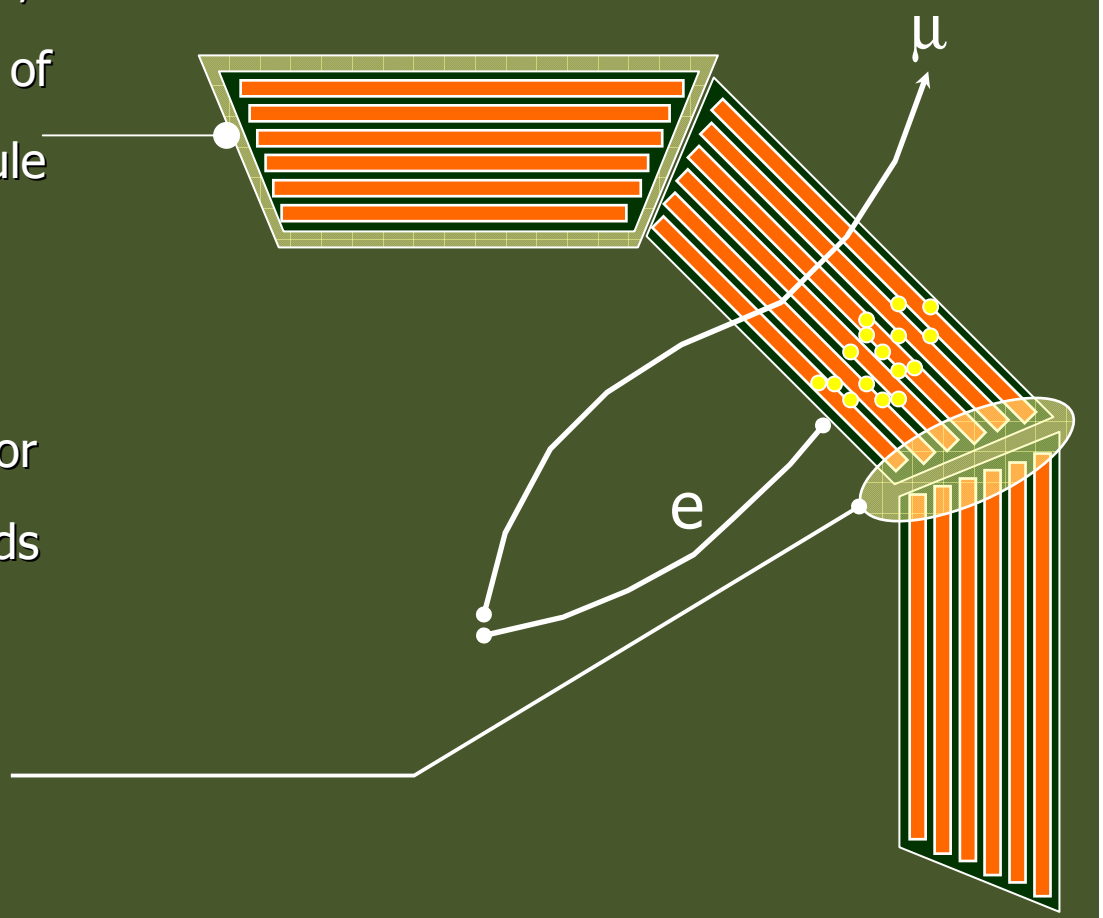
Fast simulation
(shower parameterization)

Fast simulation - Generalities

- Fast Simulation, also called as shower parameterization, is a shortcut to the "ordinary" tracking.
- Fast Simulation allows you to take over the tracking and implement your own "fast" physics and detector response.
- The classical use case of fast simulation is the shower parameterization where the typical several thousand steps per GeV computed by the tracking are replaced by a few ten of energy deposits per GeV.
- Parameterizations are generally experiment dependent. Geant4 provides a convenient framework.

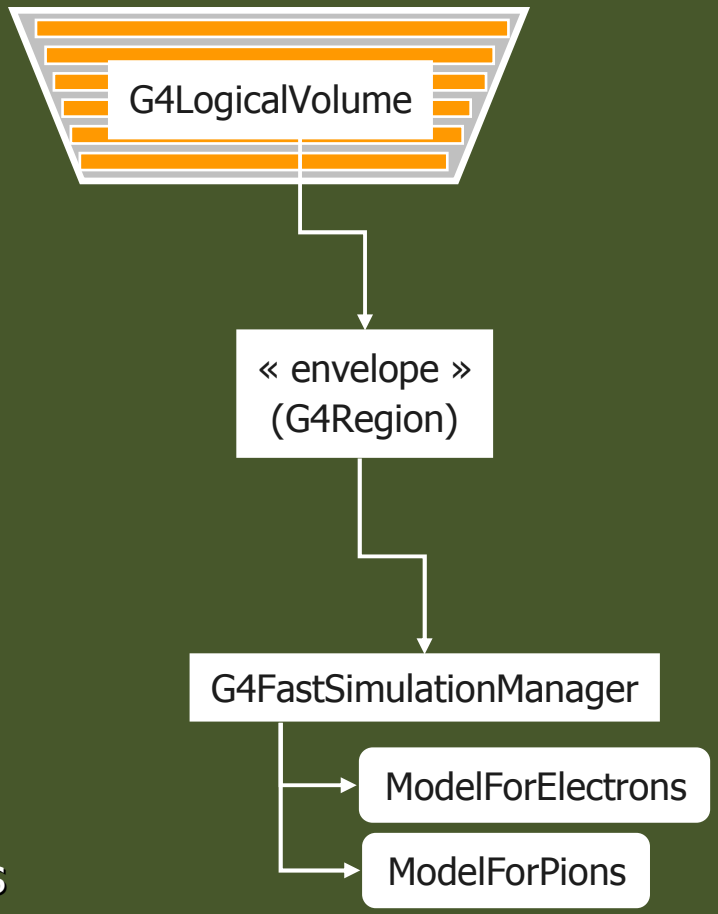
Parameterization features

- Parameterizations take place in an *envelope*. An envelope is a region, that is typically a mother volume of a sub-system or of a major module of such a sub-system.
- Parameterizations are often dependent to particle types and/or may be applied only to some kinds of particles.
- They are often not applied in complicated regions.



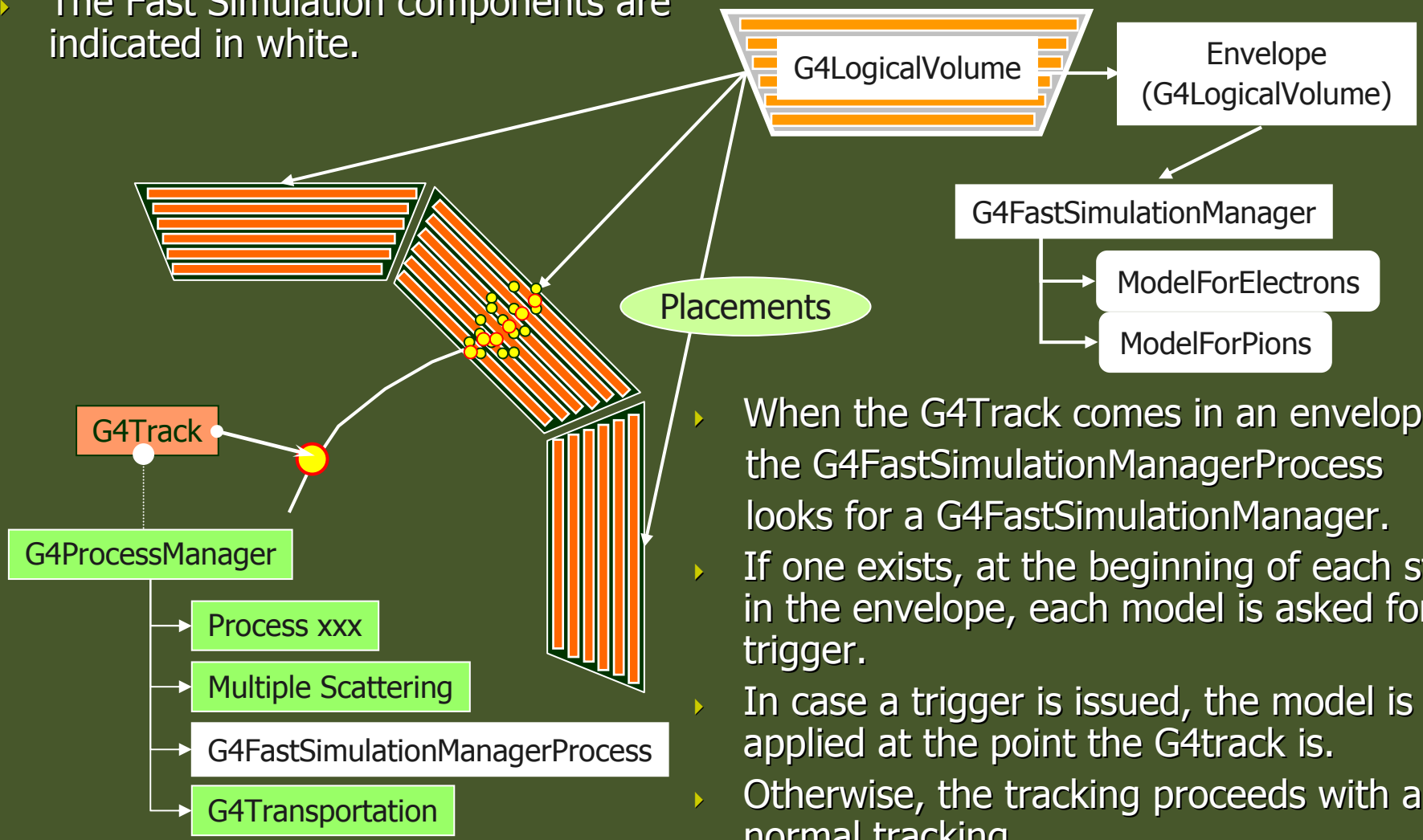
Models and envelope

- Concrete models are bound to the envelope through a G4FastSimulationManager object.
- This allows several models to be bound to one envelope.
- The envelope is simply a G4Region which has G4FastSimulationManager.
- All [grand[…]]daughter volumes will be sensitive to the parameterizations.
- A model may returns back to the "ordinary" tracking the new state of G4Track after parameterization (alive/killed, new position, new momentum, etc.) and eventually adds secondaries (e.g. punch-through) created by the parameterization.



Fast Simulation

- The Fast Simulation components are indicated in white.



- When the G4Track comes in an envelope, the G4FastSimulationManagerProcess looks for a G4FastSimulationManager.
- If one exists, at the beginning of each step in the envelope, each model is asked for a trigger.
- In case a trigger is issued, the model is applied at the point the G4track is.
- Otherwise, the tracking proceeds with a normal tracking.

G4FastSimulationManagerProcess

- The G4FastSimulationManagerProcess is a process providing the interface between the tracking and the fast simulation.
- It has to be set to the particles to be parameterized:
  - The process ordering must be the following:

    [n-3] …

    [n-2] Multiple Scattering

    [n-1] G4FastSimulationManagerProcess

    [n] G4Transportation
  - It can be set as a discrete process or it must be set as a continuous & discrete process if using ghost volumes.

Most efficient way of simulating DICOM-like 3D voxels with material parameterization

Options you can take - 1

- There is no silver bullet. You can try some/all of these options combined.
- Huge number of cells
  - If 3D parameterized volume with material parameterization is used,
    - Compact memory size but slow in case of 1D optimization
    - Fast but huge memory size in case of 3D optimization
  - Use replica for the first and second axes slices and 1-dimensional parameterization for the third axis. Use G4NestedParameterisation to parameterize the material.
- Material map
  - Though number of materials appear is quite limited, each cell must at least have a pointer to a material. I.e. you have to have a huge material map which has entries of the number of cells.
  - Split your whole voxel geometry into reasonable number of regions, and assign a dedicated stack to each region. For example 5*5*5 = 125 regions.
  - Load material map (from your file on the disk) only for one region. If a track reaches to the boundary of the region you are currently simulating, suspend the track.
  - Simulate all the tracks in one region. Once a region becomes empty, load material map for another region and simulate all tracks in that region.
  - Note that some tracks may come back to a region you have already simulated.

Options you can take - 2

- Indexing organs
  - If you are accumulating, e.g. energy deposition, just for each organ rather than for individual voxels, you may overwrite GetIndex() method of G4PSEnergyDeposit scorer to return the organ index rather than the copy number of a voxel. Then the scorer creates a map of organ index and energy deposition. Thus reduces the size of map significantly.
- Event biasing
  - In particular, geometrical importance biasing, and secondary particle splitting must be good options to take.
  - You must validate results of your biasing options with full simulation.
- Shower parameterization
  - In stead of having a full EM shower, you may want to consider the shower parameterization in particular for the core part of the shower.
- Dedicated navigator
  - Given the geometry is perfectly regular, you may want to consider implementing a dedicated navigator that is absolutely simple-minded to handle just regular pattern of boxes of same size, thus quite fast.
  - Dedicated navigator for voxel geometry is under development.
- Parallelization
  - Allocate good number of CPUs…

Tips for computing performance

Some tips to consider

- We are making our best effort to improve the speed of Geant4 toolkit. But, since it is a toolkit, a user may also make the simulation unnecessarily slow.
- For general applications
  - Check methods which are invoked frequently, e.g. UserSteppingAction(), ProcessHits(), ComputeTransformation(), GetField() etc.
  - In such methods, avoid string manipulation, file access or cout, unnecessary object instantiation or deletion, or unnecessary massive polynomial calculation such as sin(), cos(), log(), exp().
- For relatively complex geometry or high energy applications
  - Kill unnecessary secondary particles as soon as possible.
  - Utilize G4Region for regional cut-offs, user limits.
  - For geometry, consider replica rather than parameterized volume as much as possible. Also consider nested parameterization.
  - Do not keep too many trajectories.
- For relatively simple geometry or low energy applications
  - Do not store the random number engine status for each event.