Kernel III

Makoto Asai (Jefferson Lab)

Geant4 Tutorial Course

Jefferson Lab
U.S. DEPARTMENT OF ENERGY | Office of Science
JSA

Contents

Jefferson Lab
Thomas Jefferson National Accelerator Facility

- Stacking mechanism
- Moving object
- Tips for computing performance

GEANT4
A SIMULATION TOOLKIT

U.S. DEPARTMENT OF ENERGY | Office of Science
JSA

Contents

Jefferson Lab
Thomas Jefferson National Accelerator Facility

- Stacking mechanism
- Moving object
- Tips for computing performance

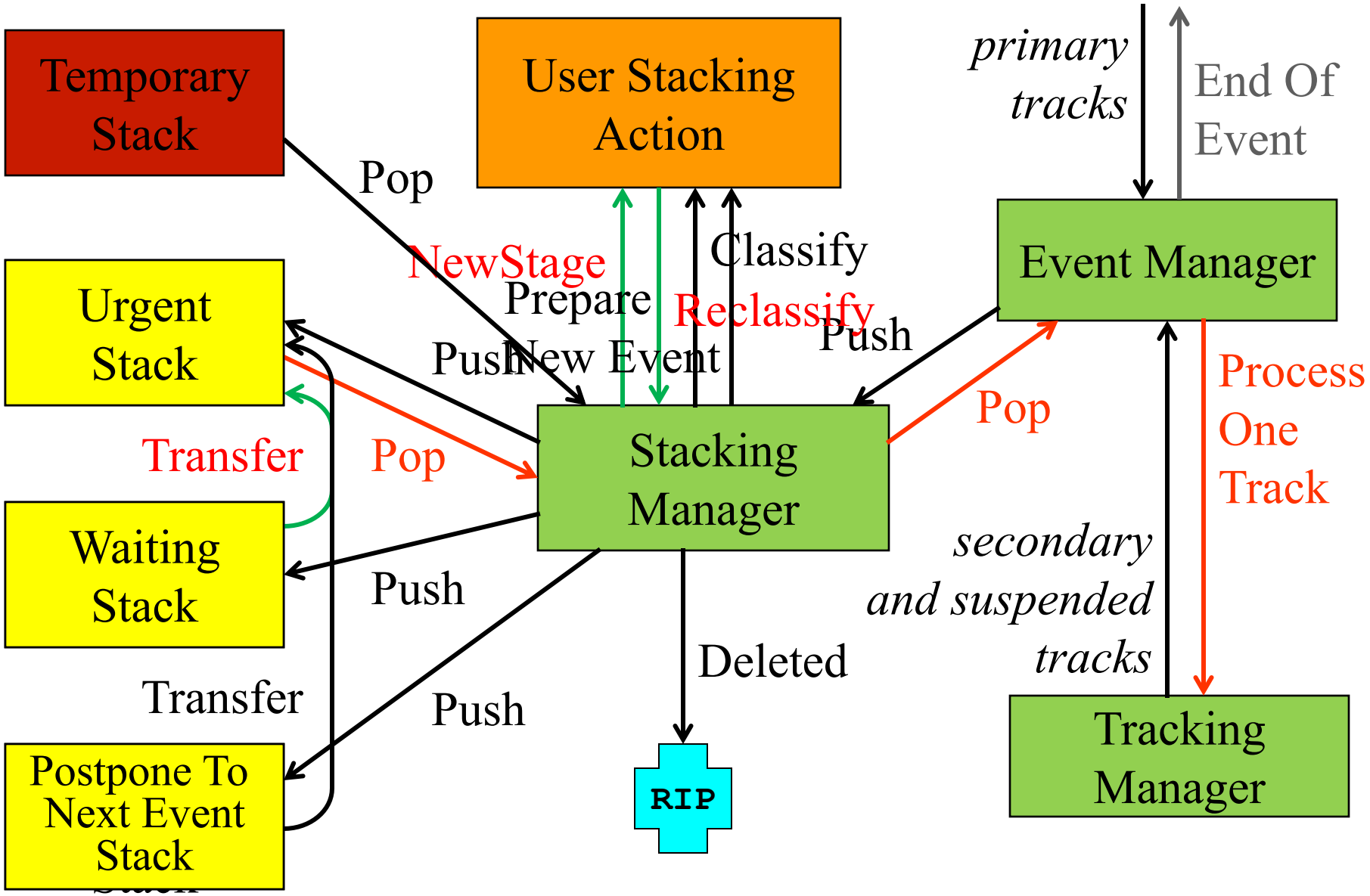
GEANT4
A SIMULATION TOOLKIT

U.S. DEPARTMENT OF ENERGY | Office of Science | JSA

Track stacks in Geant4

- By default, Geant4 has three track stacks.
  - "Urgent", "Waiting" and "PostponeToNextEvent"
  - Each stack is a simple "last-in-first-out" stack.
  - User can arbitrary increase the number of stacks.
- ClassifyNewTrack() method of UserStackingAction decides which stack each newly storing track to be stacked (or to be killed).
  - By default, all tracks go to Urgent stack.
- A Track is popped up only from Urgent stack.
- Once Urgent stack becomes empty, all tracks in Waiting stack are transferred to Urgent stack.
  - And NewStage() method of UsetStackingAction is invoked.
- Utilizing more than one stacks, user can control the priorities of processing tracks without paying the overhead of "scanning the highest priority track".
  - Proper selection/abortion of tracks/events with well designed stack management provides significant efficiency increase of the entire simulation.

Stacking mechanism



G4UserStackingAction

- User has to implement three methods.
- G4ClassificationOfNewTrack ClassifyNewTrack(const G4Track*)
  - Invoked every time a new track is pushed to G4StackManager.
  - Classification
    - fUrgent - pushed into Urgent stack
    - fWaiting - pushed into Waiting stack
    - fPostpone - pushed into PostponeToNextEvent stack
    - fKill - killed
- void NewStage()
  - Invoked when Urgent stack becomes empty and all tracks in Waiting stack are transferred to Urgent stack.
  - All tracks which have been transferred from Waiting stack to Urgent stack can be reclassified by invoking stackManager->ReClassify()
- void PrepareNewEvent()
  - Invoked at the beginning of each event for resetting the classification scheme.

Tips of stacking manipulations

- Classify all secondaries as fWaiting until Reclassify() method is invoked.
  - You can simulate all primaries before any secondaries.
- Classify secondary tracks below a certain energy as fWaiting until Reclassify() method is invoked.
  - You can roughly simulate the event before being bothered by low energy EM showers.
- Suspend a track on its fly. Then this track and all of already generated secondaries are pushed to the stack.
  - Given a stack is "last-in-first-out", secondaries are popped out prior to the original suspended track.
  - Quite effective for Cherenkov lights
- Suspend all tracks that are leaving from a region, and classify these suspended tracks as fWaiting until Reclassify() method is invoked.
  - You can simulate all tracks in this region prior to other regions.
  - Note that some back splash tracks may come back into this region later.

Set the track status

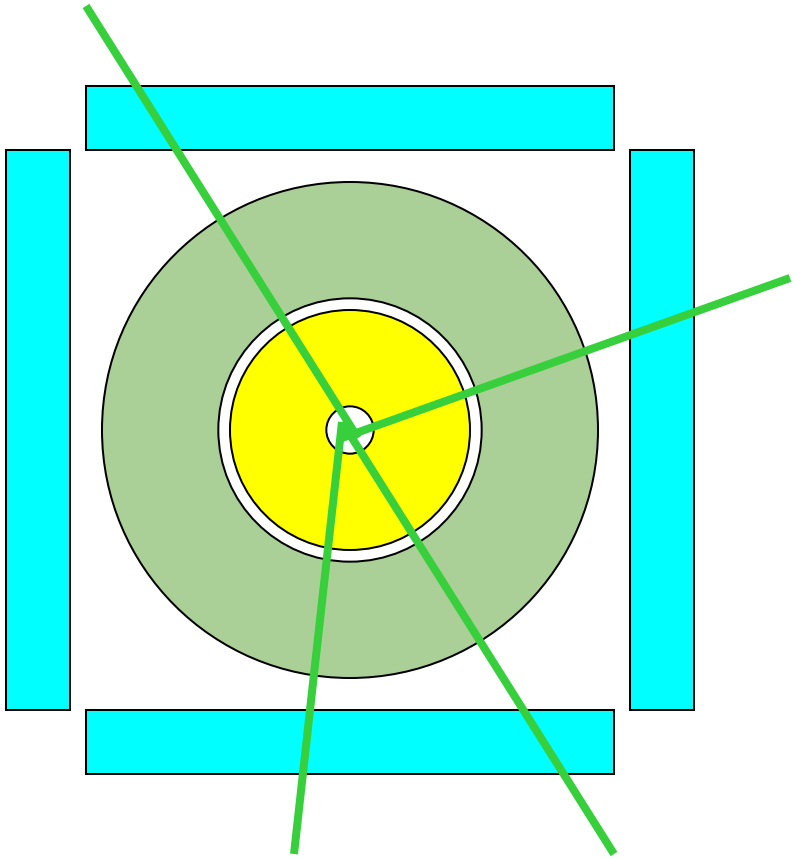
- In UserSteppingAction, user can change the status of a track.

```
void MySteppingAction::UserSteppingAction
                              (const G4Step * theStep)
{
  G4Track* theTrack = theStep->GetTrack();
  if(…) theTrack->SetTrackStatus(fSuspend);
}
```

- If a track is killed in UserSteppingAction or in the UserStackingAction, physics quantities of the track (energy, charge, etc.) are not conserved but completely lost.

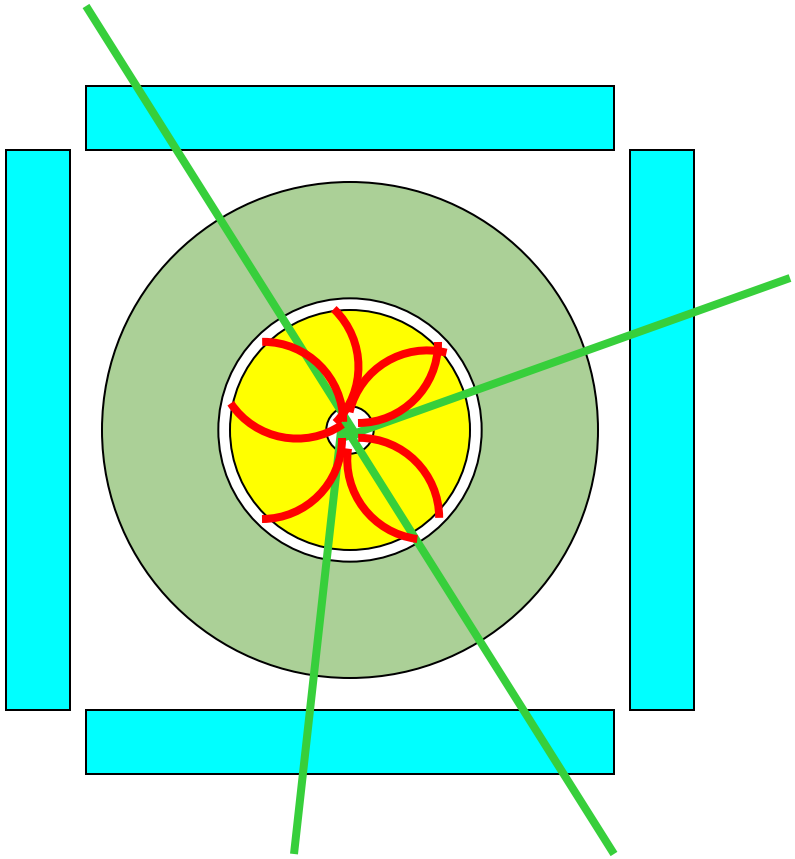
RE05StackingAction

- RE05 has simplified collider detector geometry and event samples of Higgs decays into four muons.
- Stage 0
  - Only primary muons are pushed into Urgent stack and all other primaries and secondaries are pushed into Waiting stack.
  - All of four muons are tracked without being bothered by EM showers caused by delta-rays.
  - Once Urgent stack becomes empty (i.e. end of stage 0), number of hits in muon counters are examined.
  - Proceed to next stage only if sufficient number of muons passed through muon counters. Otherwise the event is aborted.



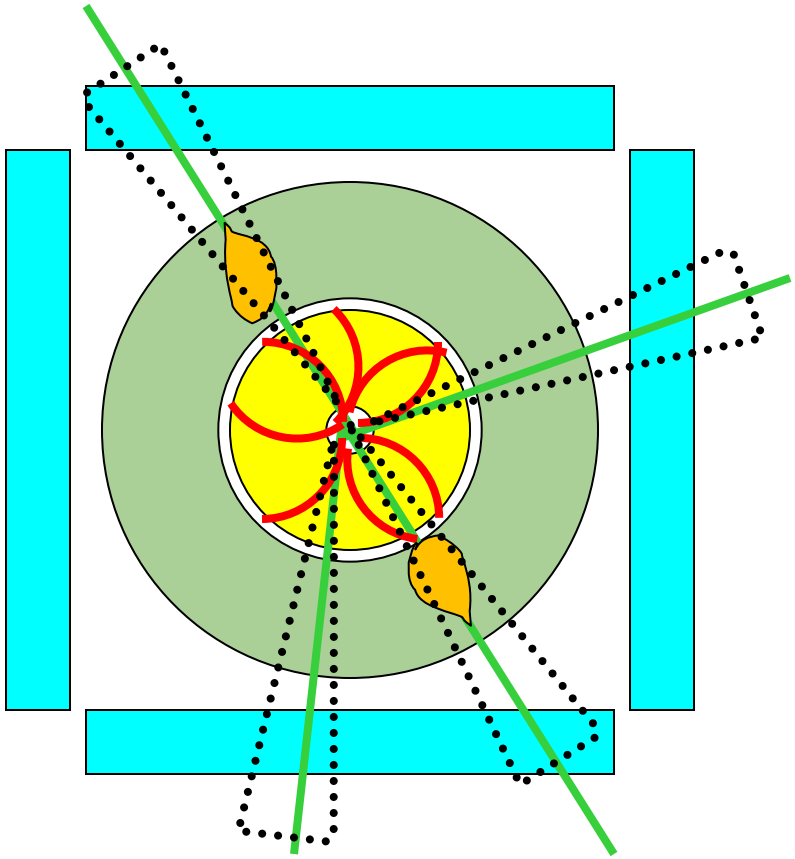
RE05StackingAction

- Stage 1
  - Only primary charged particles are pushed into Urgent stack and all other primaries and secondaries are pushed into Waiting stack.
  - All of primary charged particles are tracked until they reach to the surface of calorimeter. Tracks reached to the calorimeter surface are suspended and pushed back to Waiting stack.
  - All charged primaries are tracked in the tracking region without being bothered by the showers in calorimeter.
  - At the end of stage 1, isolation of muon tracks is examined.



RE05StackingAction

- Stage 2
  - Only tracks in "region of interest" are pushed into Urgent stack and all other tracks are killed.
  - Showers are calculated only inside of "region of interest".



Tips of stacking manipulations

- Classify all secondaries as fWaiting until Reclassify() method is invoked.
  - You can simulate all primaries before any secondaries.
- Classify tracks below a certain energy as fWaiting until Reclassify() method is invoked.
  - You can roughly simulate the event before being bothered by low energy EM showers.
- Suspend a track on its fly. Then this track and all of already generated secondaries are pushed to the stack.
  - Given a stack is "last-in-first-out", secondaries are popped out prior to the original suspended track.
  - Quite effective for Cherenkov / scintillation lights
- Suspend all tracks that are leaving from a region, and classify these suspended tracks as fWaiting until Reclassify() method is invoked.
  - You can simulate all tracks in this region prior to other regions.
  - Note that some back-splash tracks may come back into this region later.

Contents

Jefferson Lab
Thomas Jefferson National Accelerator Facility

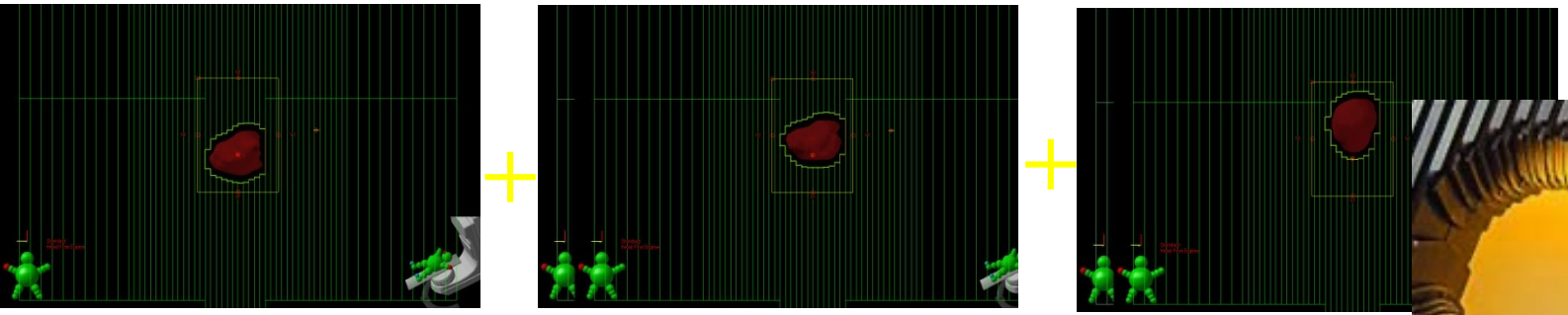
- Stacking mechanism
- Moving object
- Tips for computing performance

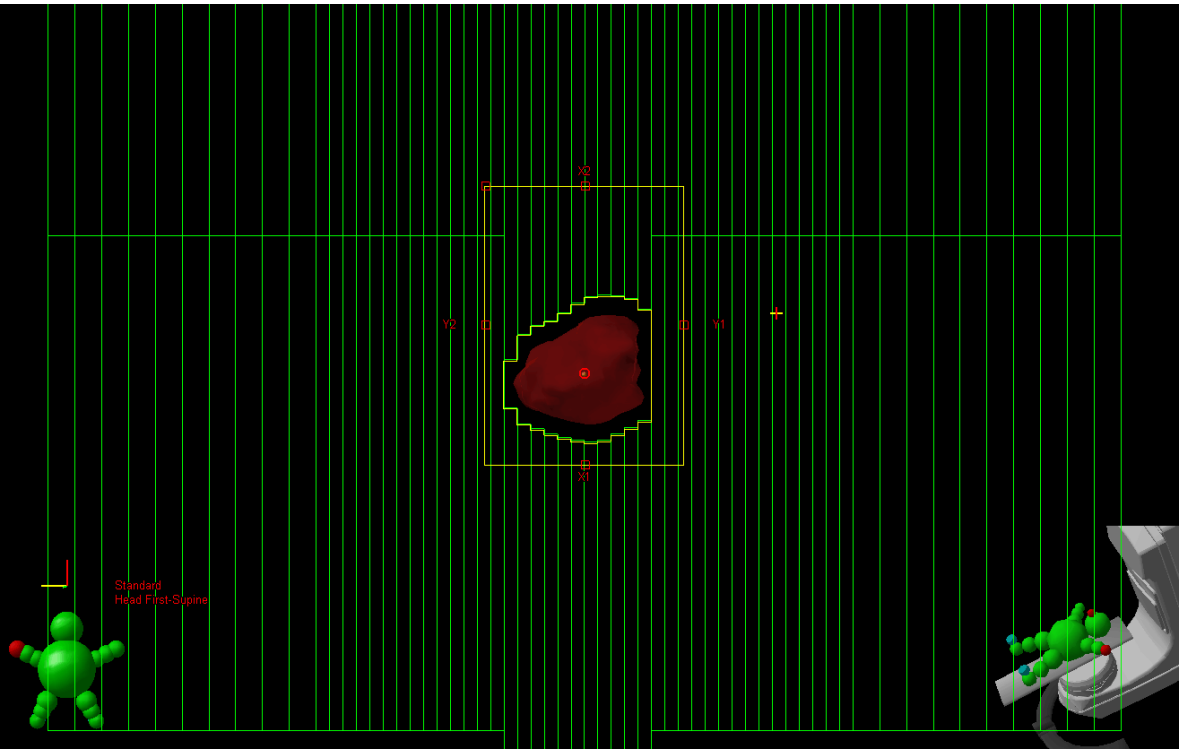
GEANT4
A SIMULATION TOOLKIT

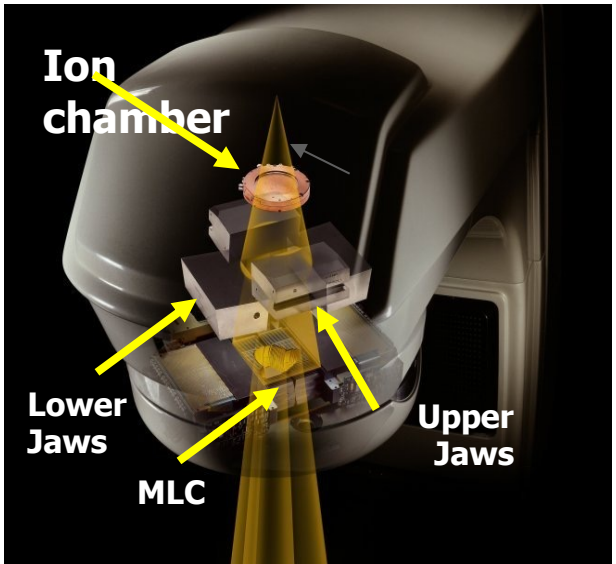
U.S. DEPARTMENT OF ENERGY | Office of Science
JSA

4D RT Treatment Plan

Source: Lei Xing, Stanford University

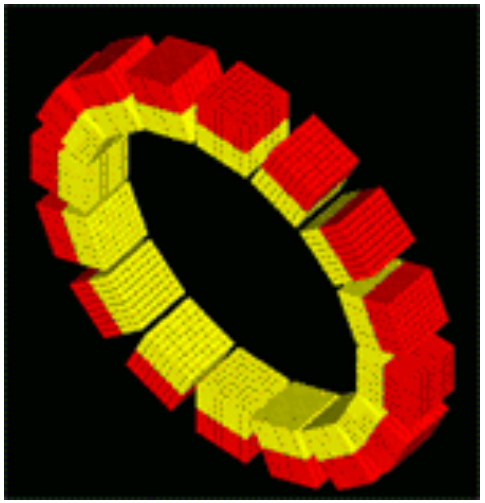






Moving objects

- In some applications, it is essential to simulate the movement of some volumes.
  - E.g. particle therapy simulation
- Geant4 can deal with moving volume
  - 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 constant 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
{
  G4RotationMatrix rMat;
  G4int eID = 0;
  const G4Event* evt =
      G4RunManager::GetRunManager()->GetCurren
  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(CLHEP::HepRotationX(-r));
  physVol->SetTranslation(G4ThreeVector(0.,0.,
  physVol->SetRotation(&rMat);
```

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

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

You are responsible not to make the moving volume get out of (protr e.

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.

Contents

Jefferson Lab
Thomas Jefferson National Accelerator Facility

- Stacking mechanism
- Moving object
- Tips for computing performance

GEANT4
A SIMULATION TOOLKIT

U.S. DEPARTMENT OF ENERGY | Office of Science

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.

Tips for relatively low-energy applications

- Do not store the random number engine status for each event.
- Do not seed worker threads for each event.

  /run/eventModulo ! 1
  - /run/eventModulo <*N*> <*seedOnce*>
  - <*N*> sets the event modulo for dispatching events to worker threads, i.e. each worker thread is ordered to simulate N events and then comes back to G4MTRunManager for next set.
    - N If it is set to zero (default value), N is roughly given by this.
      N = int(sqrt(number_of_events / number_of_threads))
    - The value N may affect on the computing performance if N is too small compared to the total number of events.
  - <*seedOnce*> specifies how frequently each worker thread is seeded by the random number sequence centrally managed by the master G4MTRunManager.
    - If seedOnce is set to 0 (default), seeds that are centrally managed by G4MTRunManager are set for every event of every worker thread. This option guarantees event reproducibility regardless of number of threads.
    - If seedOnce is set to 1, seeds are set only once for the first event of each run of each worker thread. Event reproducibility is not guaranteed depending on number of worker threads or how busy these threads are. On the other hand, this option offers better computing performance for applications with relatively small primary particle energy and large number of events.

Tips for medical applications - 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 where 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.

Tips for medical applications - 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.