Is ExclusivelyForced always exclusive?

Anna Zaborowska

GEANT4 Generic Processes and Materials WG
16/09/2019
enum G4ForceCondition {
    InActivated,
    // This PostStepDoIt is inactivated by a user
    Forced,
    // This PostStepDoIt is forced to invoke if particle is not a state of StopAndKill.
    NotForced,
    // This PostStepDoIt is not forced to invoke.
    Conditionally,
    // This PostStepDoIt is forced to invoke only when corresponding
    // AlongStepDoIt limits the Step.
    ExclusivelyForced,
    // Only this PostStepDoIt (or AtRestDoIt) is exclusively forced
    // to invoke - all other DoIt including AlongStepDoIts are ignored.
    StronglyForced
    // This PostStepDoIt is really forced to invoke, anyway.
};
void G4SteppingManager::DefinePhysicalStepLength() {
    for (size_t np=0; np < MAXofPostStepLoops; np++) {
        if (fCondition==ExclusivelyForced) {
            for (size_t nrest=np+1; nrest < MAXofPostStepLoops; nrest++){
                (*fSelectedPostStepDoItVector)[nrest] = InActivated;
            }
            return; // Take note the 'return' at here !!!
        } else {
            if (physIntLength < PhysicalStep ){
                PhysicalStep = physIntLength;
                fStepStatus = fPostStepDoItProc;
                fPostStepDoItProcTriggered = G4int(np);
                fStep->GetPostStepPoint()->SetProcessDefinedStep(fCurrentProcess);
            }
        }
    }
}

void G4SteppingManager::InvokePostStepDoItProcs() {
...
void G4SteppingManager::InvokePostStepDoItProcs() {
    // Exit from PostStepLoop if the track has been killed,
    // but extra treatment for processes with Strongly Forced flag
    if (fTrack->GetTrackStatus() == fStopAndKill) {
        for (size_t np1=np+1; np1 < MAXofPostStepLoops; np1++) {
            G4int Cond2 = (*fSelectedPostStepDoItVector)[MAXofPostStepLoops-np1-1];
            if (Cond2 == StronglyForced) {
                InvokePSDIP(np1);
            }
        }
    }
    ...
}

Even if particle is killed, invoke PostStepDoIt for StronglyForced processes on updated G4Step.
ExclusivelyForced

ExclusivelyForced

```
./track/include/G4StepStatus.hh:  fExclusivelyForcedProc,
./track/include/G4ForceCondition.hh:  ExclusivelyForced,
./tracking/src/G4SteppingVerbose.cc:  (fCondition == ExclusivelyForced) ||
./tracking/src/G4SteppingVerbose.cc:  if(fCondition==ExclusivelyForced){
./tracking/src/G4SteppingVerbose.cc:  G4cout << "ExclusivelyForced" << G4endl;
./tracking/src/G4SteppingManager2.cc:  case ExclusivelyForced:
./tracking/src/G4SteppingManager2.cc:  (*fSelectedPostStepDoItVector)[np] = ExclusivelyForced;
./tracking/src/G4SteppingManager2.cc:  fStepStatus = fExclusivelyForcedProc;
./tracking/src/G4SteppingManager2.cc:  if (fCondition==ExclusivelyForced) {
./tracking/src/G4SteppingManager2.cc: // If the current Step is defined by a 'ExclusivelyForced'
./tracking/src/G4SteppingManager2.cc:  if(fStepStatus == fExclusivelyForcedProc){
./tracking/src/G4SteppingManager2.cc:  ((Cond == Forced) && (fStepStatus != fExclusivelyForcedProc)) ||
./tracking/src/G4SteppingManager2.cc:  ((Cond == ExclusivelyForced) && (fStepStatus == fExclusivelyForcedProc)) ||
./processes/parameterisation/src/G4FastSimulationManagerProcess.cc:  *condition = ExclusivelyForced;
./processes/electromagnetic/dna/management/src/G4ITStepProcessor2.cc: // If the current Step is defined by a 'ExclusivelyForced'
./processes/electromagnetic/dna/management/src/G4ITStepProcessor2.cc:  ((Cond == Forced) && (stepStatus !=
./processes/electromagnetic/dna/management/src/G4ITStepProcessor2.cc:  ((Cond == ExclusivelyForced) && (stepStatus ==
./processes/electromagnetic/dna/management/src/G4ITStepProcessor2.cc:  (fpState->fSelectedPostStepDoItVector)[np] =
./processes/electromagnetic/dna/management/src/G4ITStepProcessor2.cc:  (fpState->fStepStatus == fExclusivelyForcedProc);
./processes/electromagnetic/dna/management/src/G4ITSteppingVerbose.cc:  if(fCondition == Conditionally)  (fCondition ==
./processes/electromagnetic/dna/management/src/G4ITSteppingVerbose.cc:  if(fCondition == "ExclusivelyForced") << G4endl;
```
StronglyForced (1/2)

processes/biasing/importance/ (not used any more)
StronglyForced (2/2)

 processes/electromagnetic/xrays/ - G4ScoreSplittingProcess and G4ParallelWorldProcess
Example
• fast simulation
• SD defined in parallel geometry.

```cpp
g4::G4VModularPhysicsList* physicsList = new FTFP_BERT();
g4::G4String parallelWorldName = "readoutWorld";
auto fastSimulationPhysics = new myG4FastSimulationPhysics();
fastSimulationPhysics->ActivateFastSimulation("e-", parallelWorldName);
physicsList->RegisterPhysics(fastSimulationPhysics);
physicsList->RegisterPhysics(new G4ParallelWorldPhysics(parallelWorldName));
...
```
• fast simulation
• SD defined in parallel geometry.

```
G4VModularPhysicsList* physicsList = new FTFP_BERT();
G4String parallelWorldName = "readoutWorld";
auto fastSimulationPhysics = new myG4FastSimulationPhysics();
fastSimulationPhysics->ActivateFastSimulation("e-", parallelWorldName);
physicsList->RegisterPhysics( fastSimulationPhysics );
physicsList->RegisterPhysics( new G4ParallelWorldPhysics(parallelWorldName) );
...```

Anna Zaborowska
```cpp
void G4FastSimulationHelper::ActivateFastSimulation(G4ProcessManager* pmanager)
{
    pmanager->AddDiscreteProcess(fastSimProcess);
}

void G4FastSimulationHelper::ActivateFastSimulation(G4ProcessManager* pmanager, G4String parallelGeometryName)
{
    pmanager->AddProcess(fastSimProcess);
    pmanager->SetProcessOrdering(fastSimProcess, idxAlongStep, 1);
    pmanager->SetProcessOrdering(fastSimProcess, idxPostStep);
}
```

Default ordering of post step: 1000
void G4ParallelWorldPhysics::ConstructProcess()
{
    G4ParallelWorldProcess* theParallelWorldProcess = new G4ParallelWorldProcess(namePhysics);
    auto myParticleIterator=GetParticleIterator();
    myParticleIterator->reset();
    while( (*myParticleIterator)() ){
        G4ParticleDefinition* particle = myParticleIterator->value();
        G4ProcessManager* pmanager = particle->GetProcessManager();
        pmanager->AddProcess(theParallelWorldProcess);
        if(theParallelWorldProcess->IsAtRestRequired(particle))
            {pmanager->SetProcessOrdering(theParallelWorldProcess, idxAtRest, 9900);}
        pmanager->SetProcessOrderingToSecond(theParallelWorldProcess, idxAlongStep);
        pmanager->SetProcessOrdering(theParallelWorldProcess, idxPostStep, 9900);
    }
}

Ordering of post step: 9900
Example - what will happen?

Ordering:
- Parallel physics registered with ordering 9900 (post step)
- Fast simulation registered with ordering 1000 (post step)
Example - what will happen?

Ordering:
- Parallel physics registered with ordering 9900 (post step)
- Fast simulation registered with ordering 1000 (post step)

Stepping manager:
1. iterates over processes to choose the step–defining process
   **AND** post-step methods to be invoked
Example - what will happen?

Ordering:

- Parallel physics registered with ordering 9900 (post step)
- Fast simulation registered with ordering 1000 (post step)

Stepping manager:

1. iterates over processes to choose the step–defining process
   AND post-step methods to be invoked
   highest $\rightarrow$ lowest ordering
   - ordering 9900: parallel physics (StronglyForced)
   - ordering 1000: fast sim physics (ExclusivelyForced)
   - ordering <1000: ignored (inActivated)
Example - what will happen?

Ordering:
- Parallel physics registered with ordering 9900 (post step)
- Fast simulation registered with ordering 1000 (post step)

Stepping manager:
1. iterates over processes to choose the step–defining process
   **AND** post-step methods to be invoked
   \[\text{highest} \rightarrow \text{lowest ordering}\]
   - ordering 9900: parallel physics (StronglyForced)
   - ordering 1000: fast sim physics (ExclusivelyForced)
   - ordering <1000: ignored (inActivated)

2. invoke selected `PostStepDoIt`
   \[\text{lowest} \rightarrow \text{highest ordering}\]
Example - what will happen?

Ordering:

- Parallel physics registered with ordering 9900 (post step)
- Fast simulation registered with ordering 1000 (post step)

Stepping manager:

1. iterates over processes to choose the step–defining process
   AND post-step methods to be invoked
   \textit{highest} \rightarrow \textit{lowest ordering}
   
   \begin{itemize}
   \item ordering 9900: parallel physics (StronglyForced)
   \item ordering 1000: fast sim physics (ExclusivelyForced)
   \item ordering <1000: ignored (inActivated)
   \end{itemize}

2. invoke selected \texttt{PostStepDoIt}
   \textit{lowest} \rightarrow \textit{highest ordering}
   
   \begin{itemize}
   \item ordering 1000: fast sim physics
   \item ordering 9900: parallel physics
   \end{itemize}
Example - invoke PostStepDoIt (1/2)

*lowest $\rightarrow$ highest ordering*

1. ordering 1000: fast sim physics (if triggered)
Example - invoke PostStepDoIt (1/2)

lowest → highest ordering

1. ordering 1000: fast sim physics (if triggered)
   - take entering electron, $E_0 = 10$ GeV
Example - invoke PostStepDoIt (1/2)

lowest $\rightarrow$ highest ordering

1. ordering 1000: fast sim physics (if triggered)
   - take entering electron, $E_0 = 10$ GeV
   - kill particle on spot (step length $\Delta s = 0$) “depositing” all energy $\Delta E = E_0$
     (G4VParticleChange returned by PostStepDoIt)

```cpp
fastStep.KillPrimaryTrack();
fastStep.SetPrimaryTrackPathLength(0.0);
fastStep.SetTotalEnergyDeposited(fastTrack.GetPrimaryTrack()->GetKineticEnergy());
```

Example - invoke PostStepDoIt (1/2)

lowest $\rightarrow$ highest ordering

1. ordering 1000: fast sim physics (if triggered)
   - take entering electron, $E_0 = 10$ GeV
   - kill particle on spot (step length $\Delta s = 0$) “depositing” all energy $\Delta E = E_0$
     (G4VParticleChange returned by PostStepDoIt)

   ```cpp
   fastStep.KillPrimaryTrack();
   fastStep.SetPrimaryTrackPathLength(0.0);
   fastStep.SetTotalEnergyDeposited(fastTrack.GetPrimaryTrack()->GetKineticEnergy());
   ```

   - do fast sim: e.g. create energy deposits in SD according to some model

     ```cpp
     void GFlashHitMaker::make(GFlashEnergySpot * aSpot, const G4FastTrack * aT) {
       pSensitive = pCurrentVolume->GetLogicalVolume()->GetSensitiveDetector();
       G4VGFlashSensitiveDetector * gflashSensitive =
         dynamic_cast<G4VGFlashSensitiveDetector *> (pSensitive);
       if (gflashSensitive) gflashSensitive->Hit(&theSpot);
     }
     ```
Example - invoke PostStepDoIt (2/2)

```cpp
void G4SteppingManager::InvokePostStepDoItProcs() {
    // Exit from PostStepLoop if the track has been killed,
    // but extra treatment for processes with Strongly Forced flag
    if (fTrack->GetTrackStatus() == fStopAndKill) {
        for (size_t np1 = np1 + 1; np1 < MAXofPostStepLoops; np1++) {
            G4int Cond2 = (*fSelectedPostStepDoItVector)[MAXofPostStepLoops - np1 - 1];
            if (Cond2 == StronglyForced)
                InvokePSDIP(np1);
        }
    }
    ...
}
```
Example - invoke PostStepDoIt (2/2)

```cpp
void G4SteppingManager::InvokePostStepDoItProcs() {
    // Exit from PostStepLoop if the track has been killed,
    // but extra treatment for processes with Strongly Forced flag
    if (fTrack->GetTrackStatus() == fStopAndKill) {
        for (size_t np1 = np + 1; np1 < MAXofPostStepLoops; np1++) {
            G4int Cond2 = (*fSelectedPostStepDoItVector)[MAXofPostStepLoops - np1 - 1];
            if (Cond2 == StronglyForced)
                InvokePSDIP(np1);

    } ...

2. ordering 9900: parallel world (StronglyForced)
```
Example - invoke PostStepDoIt (2/2)

```cpp
void G4SteppingManager::InvokePostStepDoItProcs() {
    // Exit from PostStepLoop if the track has been killed,
    // but extra treatment for processes with Strongly Forced flag
    if (fTrack->GetTrackStatus() == fStopAndKill) {
        for (size_t np1 = np + 1; np1 < MAXofPostStepLoops; np1++) {
            G4int Cond2 = (*fSelectedPostStepDoItVector)[MAXofPostStepLoops-np1-1];
            if (Cond2 == StronglyForced) InvokePSDIP(np1);
        }
    }
}
```

2. ordering 9900: parallel world (StronglyForced)

- create deposit according to an updated by fast sim G4Step ($\Delta E = E_0 = 10$ GeV, $\Delta s = 0$)

```cpp
G4VParticleChange* G4ParallelWorldProcess::PostStepDoIt(const G4Track& track, const G4Step& step) {
    G4VSensitiveDetector* sd = fGhostPreStepPoint->GetSensitiveDetector();
    if (sd)
        sd->Hit(fGhostStep);
    ...
}
```
Total deposited energy for $E = 10$ GeV electrons $\rightarrow E_{\text{dep}} = 20$ GeV
Example - current fix

Register fast sim physics with ordering higher than any other (9999)

```
@@ -49,5 +49,7 @@
  
  // -- is an Along+PostStep process, and ordering matters:
  pmanager->AddProcess(fastSimProcess);
  pmanager->SetProcessOrdering(fastSimProcess, idAlongStep, 1);
- pmanager->SetProcessOrdering(fastSimProcess, idPostStep);
+ // registered as the process with highest order so it is invoked as the first one
+ // and it is exclusively forced (and the only one executed)
+ pmanager->SetProcessOrderingToLast(fastSimProcess, idPostStep);

  }
```
Example - current fix

Register fast sim physics with ordering higher than any other (9999)

```cpp
@@ -49,5 +49,7 @@
 void G4FastSimulationHelper::ActivateFastSimulation(G4ProcessManager* pmanager,
 
 49  49 // -- is an Along+PostStep process, and ordering matters:
 50  50 pmanager->AddProcess(fastSimProcess);
 51  51 pmanager->SetProcessOrdering(fastSimProcess, idxAlongStep, 1);
 52 - pmanager->SetProcessOrdering(fastSimProcess, idxPostStep);
 53 + // registered as the process with highest order so it is invoked as the first one
 54 + // and it is exclusively forced (and the only one executed)
 55 + pmanager->SetProcessOrderingToLast(fastSimProcess, idxPostStep);
 56 }
```

Is it enough?

- this way neither the Exclusively Forced is exclusive
  ```
  // to invoke - all other DoIt including AlongStepDoIt's are ignored.
  ```

- nor Strongly Forced is ...
  ```
  // ... really forced to invoke, anyway.
  ```
For discussion

If not enough... What can be done beyond the current fix (random ideas I came up with):

1. Comments and documentation should be updated
2. Warning may be issued by `G4ProcessManager::CheckOrderingParameters`
3. ExclusivelyForced disactivates not only processes below but also above (ordering-agnostic) for
   ```cpp
   size_t nrest=np+1; nrest < MAXofPostStepLoops; nrest++)
   (*fSelectedPostStepDoItVector)[nrest] = InActivated;
   ```
4. ExclusivelyForced gets renamed to more appropriate `DefineStep/IgnoreBelow/IgnoreLowerOrdering/DefineStepAndIgnoreRest` ...
5. Maybe not a problem of ExclusivelyForced but of StronglyForced? Rethinking needed?
   // really forced to invoke, anyway.
   not always true ... (ordering...)
6. maybe fix 3. + 1. (add except for ExclusivelyForced processes)
For discussion

If not enough... What can be done beyond the current fix (random ideas I came up with):

1. Comments and documentation should be updated
For discussion

If not enough... What can be done beyond the current fix (random ideas I came up with):

1. Comments and documentation should be updated

2. Warning may be issued by G4ProcessManager::CheckOrderingParameters

ExclusivelyForced gets renamed to more appropriate DefineStep/IgnoreBelow/IgnoreLowerOrdering/DefineStepAndIgnoreRest

Maybe not a problem of ExclusivelyForced but of StronglyForced? Rethinking needed?

// really forced to invoke, anyway. not always true ... (ordering...)?
For discussion

If not enough... What can be done beyond the current fix (random ideas I came up with):

1. Comments and documentation should be updated

2. Warning may be issued by `G4ProcessManager::CheckOrderingParameters`

3. `ExclusivelyForced` disactivates not only processes below but also above (ordering-agnostic)

4. Maybe not a problem of `ExclusivelyForced` but of `StronglyForced`? Rethinking needed?

5. // really forced to invoke, anyway. not always true ... (ordering...)
For discussion

If not enough... What can be done beyond the current fix (random ideas I came up with):

👍 1. Comments and documentation should be updated

❓ 2. Warning may be issued by `G4ProcessManager::CheckOrderingParameters`

⚠️ 3. `ExclusivelyForced` disactivates not only processes below but also above (ordering-agnostic)

---

```cpp
for(size_t nrest=np+1; nrest < MAXofPostStepLoops; nrest++){
    (*fSelectedPostStepDoItVector)[nrest] = InActivated;
}
```
For discussion

If not enough... What can be done beyond the current fix (random ideas I came up with):

1. Comments and documentation should be updated

2. Warning may be issued by `G4ProcessManager::CheckOrderingParameters`

3. `ExclusivelyForced` disactivates not only processes below but also above (ordering-agnostic)

   ```
   for(size_t nrest=np+1; nrest < MAXofPostStepLoops; nrest++){
       (*fSelectedPostStepDoItVector)[nrest] = InActivated;
   }
   ```

4. `ExclusivelyForced` gets renamed to more appropriate
   `DefineStep/IgnoreBelow/IgnoreLowerOrdering/DefineStepAndIgnoreRest/...`
   
   that new condition + highest ordering = exclusive

   

Anna Zaborowska

16/09/2019
For discussion

If not enough... What can be done beyond the current fix (random ideas I came up with):

1. Comments and documentation should be updated

2. Warning may be issued by `G4ProcessManager::CheckOrderingParameters`

3. `ExclusivelyForced` disactivates not only processes below but also above (ordering-agnostic)

   ```cpp
   for(size_t nrest=np+1; nrest < MAXofPostStepLoops; nrest++){
       (*fSelectedPostStepDoItVector)[nrest] = InActivated;
   }
   ```

4. `ExclusivelyForced` gets renamed to more appropriate
   
   DefineStep/IgnoreBelow/IgnoreLowerOrdering/DefineStepAndIgnoreRest/...

   that new condition + highest ordering = exclusive

5. Maybe not a problem of `ExclusivelyForced` but of `StronglyForced`? Rethinking needed?
   
   // really forced to invoke, anyway.

   not always true ... (ordering...)
For discussion

If not enough... What can be done beyond the current fix (random ideas I came up with):

👍 1. Comments and documentation should be updated

❓ 2. Warning may be issued by `G4ProcessManager::CheckOrderingParameters`

⚠️ 3. `ExclusivelyForced` disactivates not only processes below but also above (ordering-agnostic)

```cpp
for(size_t nrest=np+1; nrest < MAXofPostStepLoops; nrest++){
    (*fSelectedPostStepDoItVector)[nrest] = InActivated;
}
```

⚠️ 4. `ExclusivelyForced` gets renamed to more appropriate

`DefineStep/IgnoreBelow/IgnoreLowerOrdering/DefineStepAndIgnoreRest/...`

that new condition + highest ordering = exclusive

⚠️ 5. Maybe not a problem of `ExclusivelyForced` but of `StronglyForced`? Rethinking needed?

    // really forced to invoke, anyway.

    not always true ... (ordering...)

❓ 6. maybe fix 3. + 1. (add except for `ExclusivelyForced` processes)
Additional slides
class G4VProcess {
  ...
  virtual G4double PostStepGetPhysicalInteractionLength(
    const G4Track& track,
    G4double previousStepSize,
    G4ForceCondition* condition) = 0;

  // Returns the Step-size (actual length) which is allowed
  // by "this" process. (for AtRestGetPhysicalInteractionLength,
  // return value is Step-time) The NumberOfInteractionLengthLeft is
  // recalculated by using previousStepSize and the Step-size is
  // calucaled accoding to the resultant NumberOfInteractionLengthLeft.
  // using NumberOfInteractionLengthLeft, which is recalculated at
  // arguments

  // G4ForceCondition* condition:
  // the flag indicates DoIt of this process is forced
  // to be called
  // Forced: Corresponding DoIt is forced
  // NotForced: Corresponding DoIt is called
  // if the Step size of this Step is determined
  // by this process
  // !! AlongStepDoIt is always called !!
}

!! AlongStepDoIt is always called !!
class G4VProcess {
    ...
    virtual G4double PostStepGetPhysicalInteractionLength(
        const G4Track& track,
        G4double previousStepSize,
        G4ForceCondition* condition
    ) = 0;

    inline G4double G4VProcess::PostStepGPIL(
        const G4Track& track,
        G4double previousStepSize,
        G4ForceCondition* condition
    )
    {
        G4double value
        = PostStepGetPhysicalInteractionLength(track, previousStepSize, condition);
        return thePILfactor*value;
    }
}
void G4SteppingManager::DefinePhysicalStepLength() {
...
for(size_t np=0; np < MAXofPostStepLoops; np++){
    fCurrentProcess = (*fPostStepGetPhysIntVector)(np);
    if (fCurrentProcess== 0) {
        (*fSelectedPostStepDoItVector)[np] = InActivated;
        continue;
    } // NULL means the process is inactivated by a user on fly.
    physIntLength = fCurrentProcess->
        PostStepGPIL( *fTrack, fPreviousStepSize, &fCondition );
...
}...
}...
void G4SteppingManager::DefinePhysicalStepLength() {

    for (size_t np=0; np < MAXofPostStepLoops; np++) {

        switch (fCondition) {
            case ExclusivelyForced:
                (*fSelectedPostStepDoItVector)[np] = ExclusivelyForced;
                fStepStatus = fExclusivelyForcedProc;
                fStep->GetPostStepPoint()->SetProcessDefinedStep(fCurrentProcess);
                break;
            case Conditionally:
                // (*fSelectedPostStepDoItVector)[np] = Conditionally;
                G4Exception("G4SteppingManager::DefinePhysicalStepLength()", "Tracking1001",
                            FatalException, "This feature no more supported");
                break;
            case Forced:
                (*fSelectedPostStepDoItVector)[np] = Forced;
                break;
            case StronglyForced:
                (*fSelectedPostStepDoItVector)[np] = StronglyForced;
                break;
            default:
                (*fSelectedPostStepDoItVector)[np] = InActivated;
                break;
        }
    }
}
void G4SteppingManager::DefinePhysicalStepLength() {
    for(size_t np=0; np < MAXofPostStepLoops; np++) {
        switch (fCondition) {
        case ExclusivelyForced:
            (*fSelectedPostStepDoItVector)[np] = ExclusivelyForced;
            fStepStatus = fExclusivelyForcedProc;
            fStep->GetPostStepPoint()->SetProcessDefinedStep(fCurrentProcess);
            break;
        case Conditionally:
            // (*fSelectedPostStepDoItVector)[np] = Conditionally;
            G4Exception("G4SteppingManager::DefinePhysicalStepLength()", "Tracking1001",
                        FatalException, "This feature no more supported");
            break;
        case Forced:
            (*fSelectedPostStepDoItVector)[np] = Forced;
            break;
        case StronglyForced:
            (*fSelectedPostStepDoItVector)[np] = StronglyForced;
            break;
        default:
            (*fSelectedPostStepDoItVector)[np] = InActivated;
            break;
        }
    }
}