Update on Shower Parametrization usage

Joanna Weng (CMS)

Geant4 technical forum, 27 September 2005

Update: Problem solved

Problem seen by CMS/ ATLAS:

fastTrack.GetPrimaryTrack()->GetMomentumDirection()

gives (sometimes) a different direction in:

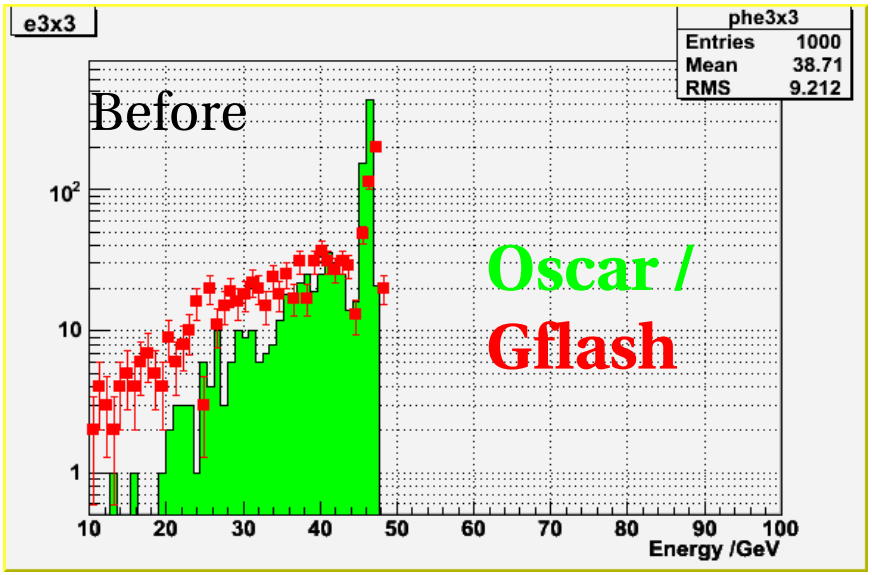
G4bool ModelTrigger(const G4FastTrack &)

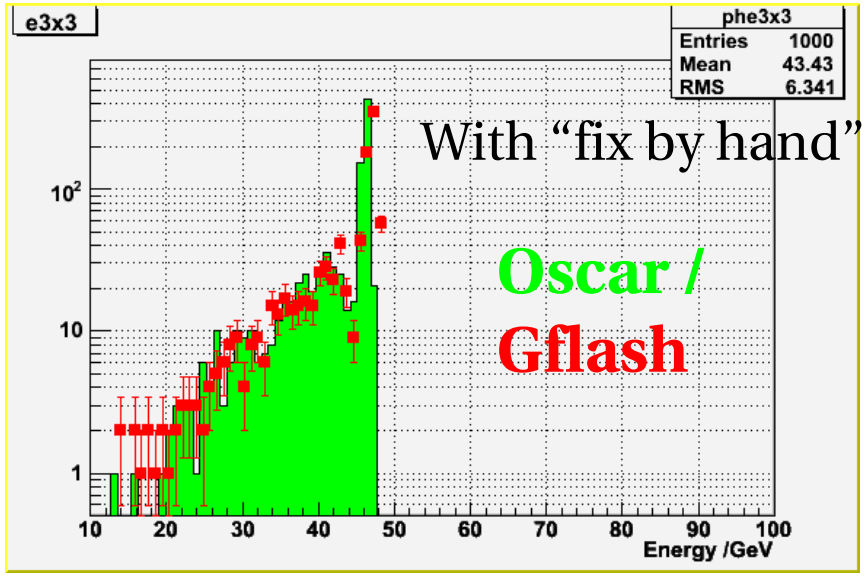
and

void DoIt(const G4FastTrack &, G4FastStep &)

(= particles are rotated)

Effect Large: (30 % of events effected in my test sample)





Update: Problem solved

Traced by CMS/ ATLAS together :-),
finally found by James A.Muller (ATLAS):

- Exclusive call of the parametrization process does not work properly
- Still another process can be called in G4SteppingManager::DefinePhysicalStepLength()

  (in this case G4MultipleScattering52::PostStepDoIt())
  before the parameterization process takes over

-> Reported to Marc / John

GFLASH in G4: Timing

CPU time of full Geant 4.7. simulation and GFLASH shower parametrization for a single electron (Pentium III @ 1Ghz) in an PbWO4 cube:

Electron Energy	Time / event full simulaton	Time / event GFLASH	Speed-up Factor
1 GeV	0.10	0.01	16.5
5 GeV	0.46	0.01	48.6
10 GeV	0.92	0.01	67.3
50 GeV	4.60	0.04	102.9
100 Gev	9.37	0.08	117.1
500 GeV	46.50	0.31	149.2
1000 GeV	91.75	0.57	162.0

Speed-up on simple geometry (cube) impressing

GFLASH in OSCAR: Timing

Speed-Up in OSCAR_5_0_0_pre1

- Single Electrons, flat eta distribution, fixed energy, from centre point :
  (Speed-up about the same for gammas)

Electron Energ	Speed–up Factor
50 GeV	2.0
100 Gev	3.8
500 GeV	11.8
1000 GeV	7.0

- Full LHC events :

Event	Speed–up Factor
H->4 e, Higgs mass 300 GeV	2.0
ADD Gamma (P_T > 1000 GeV)+ Graviton	3.3

(numbers of the same order in ATLAS)

-> Could be faster ... -> is there room for improvement ???

Ideas for more speed up: killing

Number of Parameterisation method calls:

Energy	IsApplicable()		ModelTrigger()		DoIt()	
10GeV	7400	200	48000	1100	4	4
100GeV	120000	1500	470000	8500	22	22

Elisabetta Barberio/Tom Atkinson/ Anthony Waugh

Original scheme / **new scheme**

-> IsApplicable() and ModelTrigger() called very often for low energetic particles ...(I checked: ~ same situation in CMS)

ATLAS idea: kill low energetic $\mathbf{e^+}$ / $\mathbf{e^-}$ / **photons** with 0.0 GeV < E < 0.1GeV (= **new scheme**)

Ideas for more speed up: killing

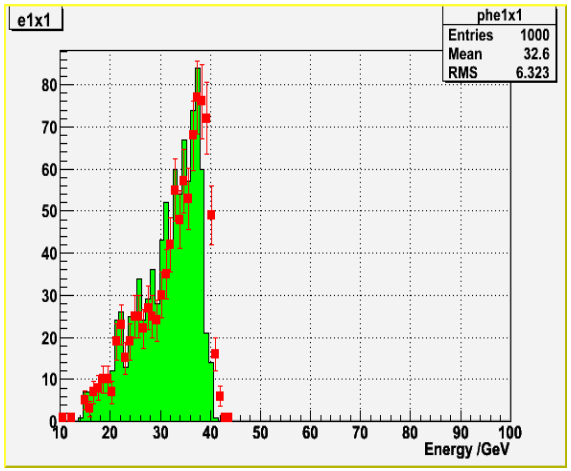
- Tried the same in CMS /OSCAR in simple scenario for different killing energies:
particles with $E < E_{kill}$ is killed and energy is deposited in the current place.
- Test setup as before, $E_{electron}$ = 50 GeV (slide 5):

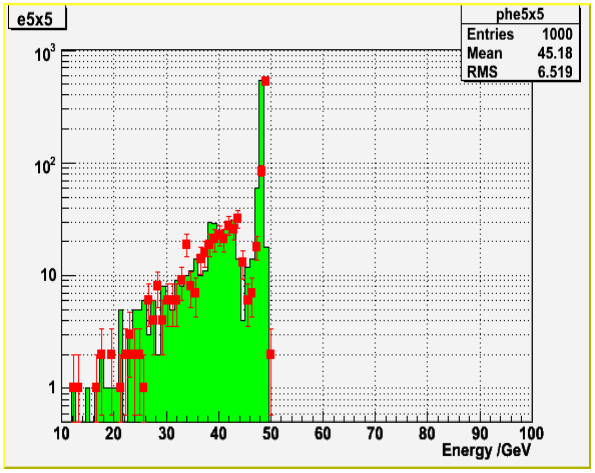
E_{kill}	Speed-Up
No kill	3.3
0.01 GeV	4.7
0.03 GeV	5.8
0.05 GeV	8.1
0.1 GeV	10.8

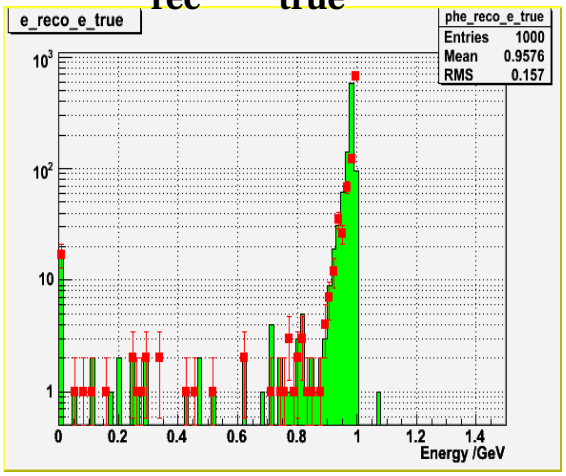
-> Approach saves a lot of time ;-)
-> But it has also impact on energy deposit in ECAL crystals :-(

Ideas for more speed up: killing

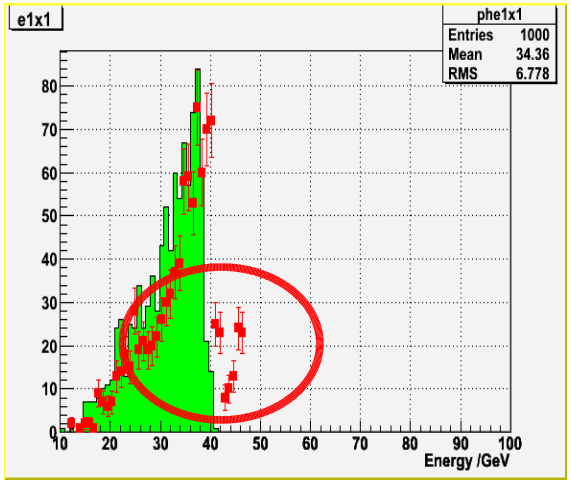
No kill : central crystal **3x3 matrix** **E_{rec} / E_{true}**

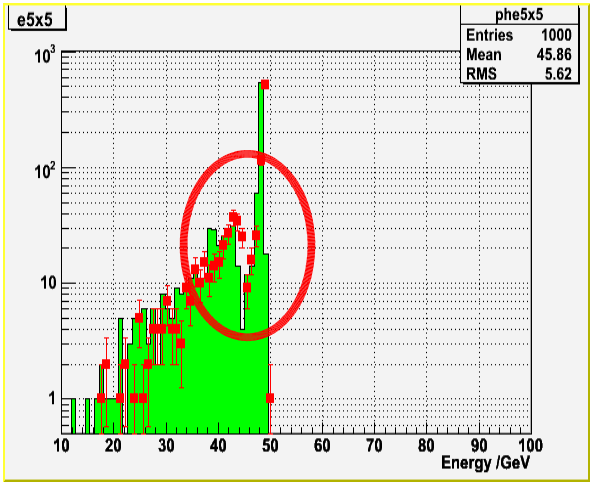


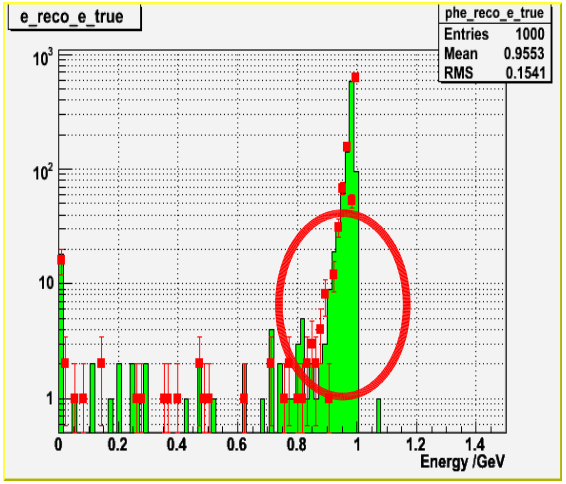




E_{kill} = 0.1 GeV:







Ideas for more speed up: k killing

- Not negligible Impact on energy deposit in CMS case with simple killing approach

Another idea:

- A lot of time spend in calling n times IsApplicable() and ModelTrigger() for low energetic particles
- They are never parameterized

-> Maybe introduce a mechanism that IsApplicable() and ModelTrigger() are NEVER called below a certain energy threshold ?

-> Any other ideas to Speed Up full events very welcome !

Conclusion

- Bug found & reported
- GFLASH performs great on simple geometries and for single electrons also on complex geometries
- Full events on complex geometries, where there is always full tracking & parameterization, are still speed-up up to 5 times ...
- More speed-up possible ??? (My wish :-))

  -> Ideas welcome !

Last remark :

It is hard to find the right envelope in the CMS geometry, that
a.) is a G4LogicalVolume
b.) contains all the sensitive material (= ECAL crystals)

-> I look forward to test the new region based code for the shower parameterization

Reminder: G4VFastSimulationModel

Override the three following methods to create your model:

- **G4bool IsApplicable(const G4ParticleDefinition *)**
  - Must return 'true' for if your model applies to the given particle.
- **G4bool ModelTrigger(const G4FastTrack &)**
  - Must return 'true' if you want your model to take over the tracking at the current point.
  - The trigger decision is helped using the ***G4FastTrack*** which aggregates the G4Track and envelope related informations.
- **void DoIt(const G4FastTrack &, G4FastStep &)**
  - This is your parameterization properly said, where you will develop your parameterized shower for example. DoIt() is invoked if ModelTrigger() has previously returned 'true'.
  - To tell the tracking that you moved/killed/… the primary G4Track and/or produced secondaries during your parameterization you have to fill the ***G4FastStep*** accordingly.

Backup: Problem solved

Details:

if trigger decides to parameterize it sets:
(*fSelectedPostStepDoItVector)[4]=ExclusivelyForced
and returns, leaving the (*fSelectedPostStepDoItVector)[5] still set to ExclusivelyForced.
Thus InvokePostStepDoIt runs whatever happens to be process 5 for the e- (in this case multiple scattering).
A simple solution :

```
// For ExclusivelyForced, reset rest of array and return
    if (fCondition==ExclusivelyForced) {
      for(size_t nrest=np+1; nrest < MAXofPostStepLoops; nrest++){
         (*fSelectedPostStepDoItVector)[nrest] = InActivated;
      }
      return;  // Take note the 'return' at here !!!
    }
```