Geant4 Atomic Deexcitation (new) interface

Switch It On

V. Ivantchenko, A. Mantero
News!

- Deexcitation AlongStep
- Deexcitation per Region
- Deexcitation in EmStd
- Deexcitation for all ionisation processes
- Electron (and ions) ionisation cross sections models can be choosen
- Deexcitation *used* in DNA
- Manageable from UI

PIXE simulation in Geant4, X-Ray Spec. 40 (2011) 135-140
The Old Developer Style

- Instanciate an AtomicDeexcitation object in the model
- Go to SampleSecondaries in the model (or, if VERY OLD Style go to PostStepDolt in process)
- Find the vacancy (Id) in the Electronic Structure of an atom (Z)
- Pass (Id, Z) to AtomiDeexcitation
- Get the secondaries as the output
The Old Developer Style

• Instanciate an AtomicDeexxcitation object in the model

• Go to SampleSecondaries in the model (or, if VERY OLD Style go to PostStepDolt in process)

• Find the vacancy (Id) in the Electronic Structure of an atom (Z)

• Pass (Id, Z) to AtomiDeexxcitation

• Get the secondaries as the output

This means cut & paste of the same code for each model (or process)
The new Developer Style
What to do if you want to use deexcitation

• In the model & in the process
  • If the process derives from G4VEnergyLossProcess
    - For the continuous part
      • Do nothing
    - For the discrete part
      • Sample Deexcitation in model
  • If the process DOES NOT derive from G4VEnergyLossProcess
    - For the continuous part
      • Sample Deexcitation
    - For the discrete part
      • Sample Deexcitation in model
The new Developer Style
What to do if you want to use deexcitation

- How to Sample Deexcitation
  - In Continuous Process
    - Get the “deexcitation thing” form G4LossTableManager
    - Pass &ParticleChange, step, eLoss, coupleIndex to the deexcitation thing whenever Deexcitation happens
  - In Discrete Model
    - Mark “deexcitation on” in the constructor
    - Get the “deexcitation thing” from EnergyLoss things
    - Find a vacancy (Id) in the Electronic Structure of an atom (Z)
    - Pass (secVect, Id, Z, region) to AtomiDeexcitation
Up and Working

- Processes (Continuous - realized by std, Livermore and penelope models)
  - G4alphalonisation
  - G4eBremsstrahlung
  - G4elonisation
  - G4ionlonisation
Up and Working

• Models (Discrete)
  • G4PhotoelectricEffect
    - G4PEEffectFluoModel
    - G4KleinNishinaModel
    - G4PenelopePhotoElectricModel
    - G4LivermorePhotoElectricModel
  • G4ComptonScattering
    - G4LivermoreComptonModel
    - G4PenelopeComptonModel
The Old User Style

- In PhysicsList
  - `G4PhotoElectricEffect* thePhotoElectricEffect = new G4PhotoElectricEffect();`
  - `theLivermorePhotoElectricModel = new G4LivermorePhotoElectricModel();`
  - `TheLivermorePhotoElectricModel ->ActivateAuger(true);`

- In UI
  - No UI commands
The new User style

- In PhysicsList: use G4EmOptions
  - `emOptions.SetFluo(true);` // To activate deexcitation processes and fluorescence
  - `emOptions.SetAuger(true);` // To activate Auger effect if deexcitation is activated
  - `emOptions.SetPIXE(true);` // To activate Particle Induced X-Ray Emission (PIXE)

- Use pre-built PhysicsLists
  - Everything is already done
The new User style

- Via UI:
  - `/run/initialize`
  - `/process/em/fluorescence true`
    - (activate Deexcitation THIS IS THE DEFAULT)
  - `/process/em/auger true`
    - (activate Auger production TOO)
  - `/process/em/pixe true`
    - (activate Deexcitation form hadrons and electrons TOO)
  - `/process/em/deexcitation region true true true`
    - (activate deexcitation for given region)
  - `/process/em/pixeXSmodel analytical or empirical`
    - (choose between ECPSSR or Paul/Orlic cross section for PIXE)

Further infos at: https://twiki.cern.ch/twiki/bin/view/Geant4/LoweAtomicDeexcitation
Perfect? No.

- Discrete generation depends on continuous things
- Strictly bound to “LowEnergy” Deexcitation
- Logic is not user-friendly
  - It is cpu-friendly!
- Deexcitation-per-process has been lost
Perfect? No.

- Discrete generation depends on continuous things
- Strictly bound to “LowEnergy” Deexcitation
- Logic is not user-friendly
- CPU-friendly!

Deexcitation has been GREATLY improved:
- Deexcitation-per-process has been lost
  - Design
  - Performance
  - Customization
  - Coding
  - Availability
  - User Interface