G4HadronicProcess class: proposal to improve

V.Ivanchenko

14th Geant4 User and Collaboration Workshop
Catania (Italy) 19-23 October 2009

Introduction

- G4HadronicProcess class is a base class for hadronic processes
- Cleaned up for 9.2 :
  - Removed inline constructor and destructor from base and derived classes
  - Added PreparePhysicsTable method
  - Removed unnecessary and strange methods
  - G4HadronicProcessStore and register/deregister mechanism
  - More natural sampling of isotope
  - Removed call to one environment variable
- There are still strange methods and lines of strange code
  - We are limited by our rule – no change of user interface

Exceptions: status

- Currently there are 18 Exceptions inside G4HadronProcess class and 3 different implementations!
  - It seems to be too much
- G4HadronicException class is based on standard c++ try/cauch solution
- G4HadReentrentException is checked in parallel to G4HadronicException
  - Used only for kaons in LHEP and for RPG
- G4HadronicWhiteBoard is a singleton class, which keep strings – information from G4Track is used to produced strings at each call PostStepDoIt and printout in the case of G4HadronicException
  - Seems to be an overhead
- In the case of hadronic exception not always clear what happens because the printout is cryptic and incomplete

Exceptions: proposal

- Remove G4HadReentrentException and G4HadronicWhiteBoard classes (at least, not use these classes)
  - all model developers provide migration to G4HadronicException
- Clean up the code removing unnecessary exception
  - The most of checks should be done at initialisation and not at run time
- In the case of an exception
  - provide printout of clear message what happens
  - use standard << operator of G4Track for detail output of primary particle
  - Always cout material, Z and A
- Proposal is easy to implement for 9.3 if we agree

G4HadronicProcess: a list of strange things (extra overhead for nothing)to be removed from PostStepDoIt

- If(!ModellingState && !getenv("BypassAllSafetyChecks"))
  - ModellingState = 0,1 meanless variable
- If(result->GetStatusChange()==isAlive && thePro.GetDefinition() != aTrack.GetDefinition())
  - Against any logic
- If(getenv("HadronicDoitLogging"))
  - Use verbosityLevel instead
- G4double originalEnergy = aParticle->GetKineticEnergy();
  G4double kineticEnergy = originalEnergy;
  G4double e = aTrack.GetKineticEnergy()
- G4Nancheck – to be removed
- If(isoIsEnable || isoIsOnAnyway)
- If(e<5*GeV)
- It is easy to implement for 9.3 if we will agree

Proposal for initialisation of models

- Add to G4HadronicModel a method
  BuildPhysicsTable(const G4ParticleDefinition*)
- Propagate this method to all models assigned to a given process
- Before 9.3 it can be done as an addition – the change will require a coherent tags for few hadronic subdirectories
  - At the beginning an empty default implementation will be used
  - Each model author will be able to make initialisation at appropriate time
- Propose a general rule for all models to perform initialisation only inside the BuildPhysicsTable:
  - check on environment variables
  - instantiation std vectors and other objects needed for the model
  - definition of options and flags
- Introduce G4HadronicMessenger which will provide UI commands for hadronic processes/models as we use for EM physics
  - verbosity level
  - de-excitation options

Isotope production in G4HadronicProcess

- Not appropriate, because majority of models naturally produced residual isotopes
- Other models LHEP, RPG should include this mechanism or be discarded
- When is an optimal moment to remove isotope production?
  - If we decide to remove now it is very easy to do for 9.3
  - Alternatively it is possible to move this mechanism to the level of base parameterized models – more work but doable

Cross section biasing in G4HadronicProcess

- There is a mechanism of cross section biasing
- The correctness and quality are not clear
- There are questions:
  - can hadronic cross section be biased ignoring existence of EM processes?
  - Are there use-cases when current method is correct?
- Possible strategies:
  1. For the time being keep infrastructure for cross section biasing but not activate it
     - members of the class and methods
     - Comment out PostStepDoIt
  2. Remove all this infrastructure taking into account a fact that a proper place for the biasing is the Wrapper process
  3. Do nothing
- Any of this strategies easy to implement!