Stopping Process Concerns

Most existing stopping processes outside of hadronic framework

Inherit directly from **G4VRestProcess**

If hadronic models are used, get instantiated directly, not necessarily with expected configuration

New **G4HadronStoppingProcess** subclass of **G4HadronicProcess** provides full interface to framework

Only Bertini-based absorption model currently using

G4VProcess Classification

Based on how tracked particle loses energy/interacts with medium

Continuous

Particle transfers energy to medium all along trajectory

Discrete

Particle interacts with specific atom/nucleus in medium

Rest

Particle interacts at zero kinetic energy, possibly with nearby atom/nucleus in medium

Pairwise subclasses (e.g., **G4VRestDiscreteProcess**) allow for models with broader applicability

Hadronic Process Categories

Only discrete interactions, classified by consequences to target and projectile

Inelastic – *In flight*

Projectile interacts with target (usually nucleus), absorbed (killed), multiple interaction secondaries
(*Includes neutron-induced fission*)

Elastic – *In flight*

Projectile interacts with target, redirected, target becomes secondary (or target fragments)

Absorption – *At rest*

Projectile interacts with nucleus, killed, multiple interaction secondaries.

Radioactive Decay – *In flight or at rest*

Projectile disappears (killed), decay products as secondaries

G4HadronicProcess inherits from **G4VDiscreteProcess**, not consistent with absorption process

Inheritance Options?

Ideally, different hadronic process types would directly match G4VProcess subclasses

- *In flight* $\Longrightarrow$ **G4VDiscreteProcess**
- *At rest* $\Longrightarrow$ **G4VRestProcess**
- *Radiactive decay* $\Longrightarrow$ **G4VRestDiscreteProcess**

This either introduces three separate hadronic-process base classes without a common interface class

or

requires multiple inheritance, with virtual inheritance to deal with the consequent “diamond pattern”

Configuration Flags

G4VProcess subclasses are identified at runtime via flags, set by subclass constructors (base class sets all *true*)

> `G4bool enableAtRestDoIt` : **G4V∗Rest∗Process**
>
> `G4bool enableAlongStepDoIt` : **G4V∗Continuous∗Process**
>
> `G4bool enablePostStepDoIt` : **G4V∗Discrete∗Process**

G4HadronStoppingProcess sets `enableAtRestDoIt=true;`, overriding default from **G4VDiscreteProcess**

Defines non-trivial `AtRestDoIt` to handle interface to models

Rationalize Interface?

G4HadronicProcess could inherit from **G4VRestDiscreteProcess**

Define base `AtRestDoIt` as call-through to `PostStepDoIt`

G4HadronStoppingProcess sets `enablePostStepDoIt=false`

Keeps existing `AtRestDoIt` implementation

Unnecessary complication: Requires same kind of flag setting as current situation, no particular benefit

Other Concerns

G4RadioactiveDecay also standalone, inherits from **G4VRestDiscreteProcess**

In directory `hadronic/models`, implemented as top-level *Process*

G4HadronStoppingProcess base class in `hadronic/processes/stopping`

Ought to be `hadronic/processes/management`

Legacy stopping processes should be removed, including all usage in examples

Can this be done for 9.6? Or replace with non-functional error messages and remove in GEANT4 X?