Structure of GeantV Physics : a Proposal

M. Novak, A. Ribon

CERN PH/SFT

GeantV meeting, FNAL, 6 October 2015

Goal

- Create the basic infrastructure to fit the whole physics of GeantV
  - Decay
  - Electromagnetic physics
  - Hadronic physics
  - FastSim
  - Biasing
  - etc.

“Minimalist” Strategy

- Keep a small coupling between kernel and physics
  - ideally as it is right now
  - keep transportation in the kernel, outside the physics
- Re-use as much as possible the design of Geant4 physics
  - to avoid to reinvent the wheel
  - to benefit from more than 20 years of successful development in G4
  - but trying to streamlining it by reducing the inheritance depth
  - boosting its CPU performance by replacing, whenever possible, virtual function polymorphism with static template polymorphism
  - and offering thin interfaces for direct calls to cross sections and final-state models
- Focus on scalar case (i.e. single-track) but keep in mind also the extension to track vectorization

Applications

- Physics list with only one single process
  - e.g. Compton
- Physics list with more processes for only one single particle
  - e.g. gamma
- Physics list for electrons
  - Multiple scattering
  - Continuous-discrete processes: ionization and bremsstrahlung
- Physics list with tabulated physics, i.e. equivalent to the current one, but within the new infrastructure
  - Modular, i.e. can easily replace one process, or all processes associated to one particle type (e.g. gamma), with detailed one(s)
- Physics list for fast simulation
- ...

Main Ideas

- As in G4, any physics process can have 3 “components” :
  - Along-the-step, i.e. continuous part of the in-flight process
  - Post-step, i.e. discrete part of the in-flight process
  - At-rest, i.e. interaction when the particle stops
    - But we plan to call it directly from along-the-step and post-step, to save one useless step...
- As in G4, as a general rule, discrete and at-rest processes compete, whereas continuous processes collaborate
- As in G4, each continuous process can propose a step limitation; but for discrete processes, the proposed length takes into account the cross sections of all discrete processes of the particle (as in GV TabPhys)
  - Equivalent to G4, and with the same number of random calls
  - More adapted for simplicity, locality and hopefully track vectorization
  - Individual cross sections of the discrete processes are then used to sample the target and the process
  - Similar approach also for the lifetime of at-rest processes

Kernel – Physics Interface

- Two main kernel classes
  - *GeantPropagator* and *WorkloadManager*
- One physics interface class
  - *PhysicsProcess* : to be renamed as ***GeantPhysicsInterface***
- Start : initialization
  - *GeantPropagator::PropagatorGeom*

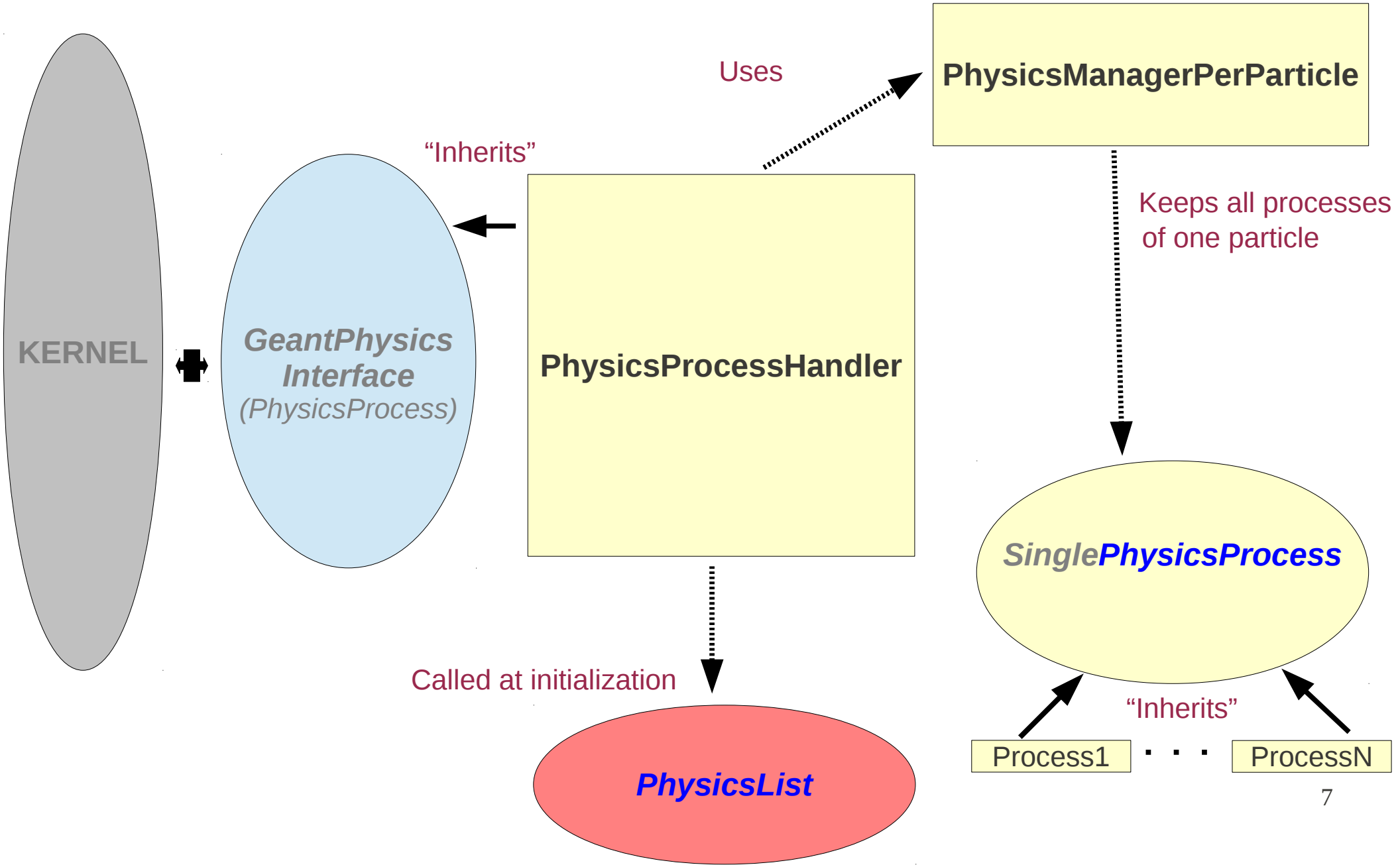
    calls *PhysicsProcess::Initialize*
- Event processing
  - *WorkloadManager::TransportTracks*

    calls GeantPropagator::ProposeStep
    *calls PhysicsProcess::**ComputeIntLen***
    *calls PhysicsProcess::**Eloss***

    *calls PhysicsProcess::**PostStepTypeOfIntrActSampling***

    *calls PhysicsProcess::**PostStepFinalStateSampling***

New Classes


KERNEL
GeantPhysics Interface
(PhysicsProcess)
PhysicsProcessHandler
Uses
PhysicsManagerPerParticle
"Inherits"
Keeps all processes of one particle
SinglePhysicsProcess
"Inherits"
Process1
ProcessN
Called at initialization
PhysicsList

PhysicsProcessHandler

- Similarly to the Geant4 *G4SteppingManager* concept, this class decides and calls the appropriate methods to simulate the physics of any given particle happening in one step
- This class "derives" from the class *GeantPhysicsInterface* and implements the following methods by calling appropriately the *PhysicsManagerPerParticle* object of the particle-type being propagated, and its associated *SinglePhysicsProcess* objects:
  - *Initialize(...)* gets production thresholds cuts, the material-cut couple table and tracking cuts; then calls *PhysicsList::Initialize*
  - *ComputeIntLen(...)* proposed length from the physics: the minimum between the single value proposed by all the discrete processes combined (draw from an exponential whose lambda is given by *PhysicsManagerPerParticle::GetTotalLambdaTable*) and the ones proposed by each continuous process *SinglePhysicsProcess::AlongStepLimitationLength*

PhysicsProcessHandler (*cont.*)

- *Eloss(...)* : to be renamed *AlongStepAction* , calls all the *SinglePhysicsProcess::AlongStepDoIt* methods
- *PostStepAction(...)* :
  - Selects the winner discrete process by comparing their respective *SinglePhysicsProcess::InverseLambda* values
  - Creates eventually vectors of light tracks (i.e. *LightTrack_v* objects)
  - Samples the target (Z, A) by calling *SinglePhysicsProcess::SampleTarget*
  - Calls the winner discrete process (and all “Forced” discrete processes) *SinglePhysicsProcess::PostStepDoIt*
- *AtRestAction(...)* :
  - Selects the winner at-rest process by comparing their *SinglePhysicsProcess::AverageLifetime* values
  - Calls the winner at rest-process (and all “Forced” at-rest processes) *SinglePhysicsProcess::AtRestDoIt* (which if needed - e.g. nuclear capture, but not for decays - samples the target (Z, A))
- *ApplyMsc(...)* : should be removed: it is not and will not be used

PhysicsManagerPerParticle

- Similarly to the Geant4 *G4ProcessManager* concept, this class keeps the list of (single) physics processes associated to one type of particle. Main methods:
  - *AddProcess(...)* (used at initialization)
  - *BuildTotalLambdaTables(...)* (used at initialization)
  - *GetParticleType(...)*
  - *GetTotalLambdaTable(...)*
  - *GetListAlongStepProcesses(...)*
  - *GetListPostStepCandidateProcesses(...)*
  - *GetListPostStepForcedProcesses(...)*
  - *GetListAtRestCandidateProcesses(...)*
  - *GetListAtRestForcedProcesses(...)*

SinglePhysicsProcess

- If we adopt the name *GeantPhysicsInterface* we can rename this class as *PhysicsProcess*
- Similarly to the Geant4 *G4VProcess* concept, this "base class" specifies one single physics process for one particle type
- Main methods
  - *bool isDiscrete, isContinuous, isAtRest;*
  - *enum ForcedCondition { InActivated, NotForced, ForcedAndCandidate, ForcedNotCandidate, ExclusivelyForced };*
  - *BuildPhysicsTables(...)* : called at initialization to build the physics tables
  - *IsApplicable(LightTrack(_v))*
  - *GetAtomicCrossSection(...)* : method useful only for direct call from users, not used in the event simulation

SinglePhysicsProcess *(cont.)*

- *InverseLambda(LightTrack(_v))* : the "macroscopic cross section" (i.e. number of atoms per unit of volume multiplied by the atomic cross section) of the discrete part of the physics process (this method is used only for the sampling of the type of interaction)
- *AlongStepLimitationLength(LightTrack(_v))* : returns the step-length limitation of the continuous part of the process
- *AverageLifetime(LightTrack(_v))* : returns the mean lifetime of the at-rest part of the physics process
- *void* SampleTarget(*LightTrack(_v)*) : samples the target (Z, A) for the discrete interaction
- *LightTrack_v* AlongStepDoIt(*LightTrack_(v)*) : does the continuous part of the physics process
- *LightTrack_v* PostStepDoIt(*LightTrack_(v)*) : does the discrete part of the physics process
- *LightTrack_v* AtRestDoIt(*LightTrack_(v)*) : does the at-rest part of the physics process (including sampling of the target (Z, A) if needed)

PhysicsList

- Similarly to Geant4, this “base class” specifies all the single physics processes associated to each particle
- We assume that all the GeantV particles are always present in the simulation, but by default they have no physics processes associated to them (i.e. they will be transported without interactions, like geantinos)
- Main methods:
  - ***Initialize(…)*** : construct all single physics processes and creates all needed physics tables

GeantPropagator::PropagatorGeom
calls GeantPhysicsInterface::Initialize
calls PhysicsList::Initialize
calls PhysicsManagerPerParticle::PhysicsManagerPerParticle
PhysicsProcess::PhysicsProcess
PhysicsProcess::PhysicsTables
PhysicsManagerPerParticle::AddProcess
PhysicsManagerPerParticle::BuildTotalLambdaTables

LightTrack (& LightTrack_**v**)

- Light version of *GeantTrack* : minimum needed by the physics
  - *int* : particle type (better particle-internal code than pdg)
  - *int* : index in the GeantTrack vector (secondaries will get the index of the parent)
  - *int* : material-cut couple index
  - *int* : track status (e.g. alive, killed, etc.)
  - *int* : index of the selected process (in the list of active discrete or at-rest processes kept by the PhysicsManagerPerParticle; negative if winner is continuous)
  - *int* : atomic number (Z) of the selected target
  - *int* : number of nucleons (A = Z + N) of the selected target
  - *double* : u_x (x-component of the unit direction)
  - *double* : u_y (y-component of the unit direction)
  - *double* : u_z (z-component of the unit direction)
  - *double* : kinetic energy (internal unit: GeV)
  - *double* : dynamic mass (internal unit: GeV)
  - *double* : weight (for biasing)
- To be placed in *VecGeom/VecCore/*
  - To allow the physics model library to depend only on this core library (not on GeantV lib)

Physical Units

- For performance reasons (i.e. avoid unnecessary multiplications), better to stick to one set of internal units in the implementation of physics models
- Different units might be eventually offered to user applications, but not to the physics part of GeantV
- Any reasonable choice would be acceptable.
  We propose:

  (**GeV** , **cm** , **sec** , g/cm^3 , T , K , …)

  Motivations:
  - The same as Pythia(8), Herwig(++), Fluka, Geant3
  - The fact that they are different from Geant4 (MeV, mm, nsec) forces us not to cut-and-paste code from Geant4

Directory Structure

- *physics/* (at the top level)
  - *data/* (independent)
  - *models/* (depends on VecCore library (and data/))
    - *decay/*
    - *electromagnetic/*
    - *hadronic/cross_sections/*
      - *final_states/*
    - *fastSim/*
  - *management/* (depends on models & GeantV libraries to start with, one single management library)
    - *processes/*
      - *electromagnetic/*
      - *decay/*
      - *hadronic/*
      - *fastSim/*
    - *physicsList/* (depends on processes/)

To-Do

- Need a singleton table with all the material-cut couples
- Need an internal particle code
  - from *0* to *Nmax* , to use as a vector index (useful in various situations...)
- Most needed physics tables are the lambda tables
  - Think about a general physics table in GeantV (look to the ones in G4)
    - Min, max, number of equally-spaced bins, either log or lin
    - Linear or higher-order interpolation
- Add more specifications about the new physics classes and start implementing them (iterative process...)
- Changes to *PhysicsProcess*
  - Rename the class and some of its methods
  - Remove the method *ApplyMsc*
  - Use static template polymorphism if possible

More information and updates in:

/afs/cern.ch/user/r/ribon/public/physics_design_draft_0.txt
physics_design_draft_1.txt
physics_design_draft_2.txt