Particle and Processes

Vladimir Ivantchenko (CERN & Princeton University)
Geant4 Advanced Course
Outline

- Geant4 basic interfaces to physics
- Geant4 particles
- Geant4 processes
- Physics Lists
- Ions and exotic particles
- Geant4 cuts
- How to control/modify physics?
Geant4 basic interface to physics

• The interface of Geant4 kernel to physics is abstract
• Base physics abstract classes are following:
  – The `G4ParticleDefinition` objects shared between threads
  – The `G4VProcess` thread local objects
  – The `G4ProcessManager` thread local interface class
• Configuration of physics is prepared in the `G4VUserPhysicsList` mandatory user class
• These interfaces are stable for >20 years allowing users to work with different Geant4 versions and providing a basis for new developments
  – Concrete physics is implemented in physics models and cross section classes
  – Alternative models and cross sections are provided in Geant4 libraries
  – A user may be also a developer of a custom particle, process, physics model, or cross section
GEANT4 PARTICLES
Geant4 Particles

• G4ParticleDefinition is the main object keeping static information about particles
  – Name, mass, charge, quantum numbers, decay table...

• “Stable” particles
  – Leptons: $e^\pm$, $\mu^\pm$, ....
  – Bosons: G4Gamma, G4OpticalPhoton, ....
  – Geantino is a particle without any interaction
  – “Stable” hadrons: $\pi^\pm$, $K^\pm$, ....
  – Light ions: d, t, $^3$He, $^4$He, and anti-ions
  – 12 hyper- and anti-hyper- nuclei are added in Geant4 11.0
  – G4GenericIon is used to define physics for all other ions

• “ShortLived” hadrons normally do not tracked by Geant4 but used internally by hadronic models
  – Quarks, di-quarks, $\rho(770)$, $\omega(783)$...
Split class – case of particle definition

- In Geant4, each particle type has its own dedicated object of G4ParticleDefinition class.
  - Static quantities: mass, charge, lifetime, decay channels, etc.,
    - Are shared by all threads
  - Dedicated object of G4ProcessManager: list of physics processes which this particle undertakes.
    - Physics process object must be thread-local
    - Thread local storage is used (TLS)

```plaintext
G4ParticleDefinition
- G4double mass
- G4double charge
- G4double lifetime
- Decay table
- ...
- G4int particleIndex

G4PartDefSplitter
- Array of TLS pointers of G4ProcessManager
  - TLS pointer
  - TLS pointer
  - TLS pointer

G4ProcessManager
- Proc man*
- Proc man*
- Proc man*
- Proc man*
- Process A*
- Process B*
- Process C*
- Process D*
```

<shared>
<static singleton>
<thread local>
GEANT4 PROCESSES
Geant4 process

- Processes are classified as:
  - Electromagnetic
  - Hadronic
  - Decay
  - Parameterized
  - Transportation
  - ..........

- Any process has process has type and sub-type
  - const G4String& G4VProcess::GetProcessType();
  - G4int G4VProcess::GetProcessSubType();
    - This method is recommended to be used for MC truth
    - The list of sub-types is stable since introduced and only extended with new processes

- Any process may be initialized using virtual methods:
  - G4bool IsApplicable(const G4ParticleDefinition &);
    - Used to check if a process can handle the given particle type
  - void PreparePhysicsTable(const G4ParticleDefinition&);
  - void BuildPhysicsTable(const G4ParticleDefinition&);
    - Used for initialization of internal data of the process before run
From G4Track to processes

- Propagated by the tracking,
- Snapshot of the particle state.

- Momentum, pre-assigned decay…

- The « particle type »:
  - G4Electron,
  - G4PionPlus…

- Holds pointers to processes

- Implementing physics processes

Configured by you, in your "physics list"

Handled by kernel

Geant4 Particles and Processes
Geant4 Physics: Electromagnetic

• **The standard EM part:** provides a complete set of EM interactions (processes) of charged particles and gammas from 1 keV to ~PeV
  – used practically in all kind of Geant4 applications
• **The low energy EM part:** includes special treatments for low energy e-/+, gammas and charged hadrons:
  – more sophisticated approximations valid down to lower energies e.g. more atomic shell structure details
  – some of these models will be valid down to ~10 eV but cannot be used above upper limits, which vary from 1 MeV to few GeV
• **Optical photons:** interactions special only for long wavelength photons
  – processes for reflection/refraction, absorption, wavelength shifting, (special) Rayleigh scattering
  – `G4OpticalPhoton` is the particle type
• **Phonon physics is also implemented within Geant4**
Geant4 Physics: Hadronic

- Pure hadronic interactions for 0 to 100 TeV
  - elastic, inelastic, capture, fission
- Radioactive decay:
  - both at-rest and in-flight
- Photo-nuclear interaction from ~1 MeV up to 100 TeV
- Lepto-nuclear interaction from ~100 MeV up to 100 TeV
  - e+ and e- induced nuclear reactions
  - muon induced nuclear reactions
- Recently introduced processes of neutrino-nuclear interactions
Decay processes includes:
- weak decay (leptonic, semi-leptonic decay, radioactive decay of nuclei)
- electromagnetic decay ($\pi^0$, $\Sigma^0$, etc.)
- strong decay not included by default
  - they are part of hadronic models
  - may be assigned by a user to a particle

Parameterized process:
- assigned to G4LogicalVolume
- instead of step-by-step simulation provides hits in the logical volume and list of particles living the volume
- for example, EM shower generation in a calorimeter based on parameters obtained from detailed simulation of the calorimeter response

Transportation process:
- responsible for propagating a particle through the geometry in electromagnetic or gravitational field
- needs to be assigned to each “stable” particle
Geant4 tracking

- **G4Track** is the object “pushed” step by step by the tracking:
  - Moving by one step is the responsibility of the “stepping”
    - Which is the core engine of the “tracking” machinery

- These moves/steps are defined by physics or by geometry
  - Step length limit is a result of competition of processes
  - Processes involved at a step may change the **G4Track**
  - By default, **G4Transportation** stops track at the volume boundary
    - There are methods how to skip boundaries during tracking
    - Implementation exit, for example, for gamma tracking in ATLAS calorimeter

*Geant4 Particles and Processes*
G4VProcess: 3 kind of actions

- **G4VProcess** is an abstract class defining the common interface of all processes in Geant4:
  - Used by all processes including G4Transportation
  - Defined in `source/processes/management`

- Three kinds of actions:
  - **AtRest** actions:
    - Decay, e$^+$ annihilation …
  - **AlongStep** actions:
    - To describe continuous (inter)actions, occurring along the path of the particle, like ionisation;
  - **PostStep** actions:
    - For describing point-like (inter)actions, like decay in flight
G4VProcess: actions summary

- The virtual «action» methods are following:
  - `AtRestGetPhysicalInteractionLength()`, `AtRestDoIt();`
  - `AlongStepGetPhysicalInteractionLength()`, `AlongStepDoIt();`
  - `PostStepGetPhysicalInteractionLength()`, `PostStepDoIt();`

- Optional run time virtual methods:
  - `StartTracking(G4Track*);`
    - Allowing the process preparation for a new G4Track
  - `EndTracking();`
    - End of given G4Track
A process can implement any combination of the three \textbf{AtRest}, \textbf{AlongStep}, and \textbf{PostStep} actions:

- for example, decay = \texttt{AtRest} + \texttt{PostStep}

If you plan to implement your own process:

- A set on intermediate classes exist implementing various combinations of actions, for example:
  - \texttt{G4VDiscreteProcess}: only \texttt{PostStep} actions
  - \texttt{G4VContinuousDiscreteProcess}: \texttt{AlongStep} + \texttt{PostStep} actions

For EM physics there are extra extensions:

- \texttt{G4VEEmProcess}, \texttt{G4VEnergyLossProcess}, \texttt{G4VMultipleScattering}

For hadronic processes there are extensions:

- \texttt{G4HadronicProcess}
- \texttt{G4HadronElasticProcess}
G4ProcessManager

• It is a Geant4 kernel class
  – A user should not change it

• **G4ProcessManager** maintains three vectors of actions:
  – One for the **AtRest** methods of the particle;
  – One for the **AlongStep** ones;
  – And one for the **PostStep** actions.

• Note, that the ordering of processes provided by/to the **G4ProcessManager** vectors is relevant and used by the stepping
  – There are few critical points you should be aware of
    – Multiple scattering can shift end point of a step
    – Scintillation, Cerenkov and some other processes assuming that a step and energy deposition at the step are defined
    – **G4PhysicsListHelper** class keeps information about process ordering for processes from Geant4 distribution
About process ordering

• The strongest rule for multiple-scattering, transportation, and G4Scintillation

• In your physics list, you should always have, for the ordering of the `AlongGetPhysicalInteractionLength(...)` methods:
  – Transportation last
    • For all particles
  – Multiple scattering second last
    • For charged particles only
      • assuming \( n \) processes
        \[
        \begin{align*}
        [n-2] & \ldots \\
        [n-1] & \text{multiple scattering} \\
        [n] & \text{transportation}
        \end{align*}
        \]

• Why?
  – Processes return a « true path length »;
  – The multiple scattering folds up this length into a **shorter** « geometrical » path length;
  – Based on this new length, the transportation can geometrically limits the step.
PHYSICS LISTS
Physics Lists

• Physics List is an object that is responsible to:
  – specify all the particles that will be used in the simulation application
  – together with the list of physics processes assigned to each individual particles

• One out of the 3 mandatory objects that the user needs to provide to the G4RunManager in case of all Geant4 applications:
  – it provides the information when, how and what set of physics needs to be invoked

• Provides a very flexible way to set up the physics environment:
  – the user can chose and specify the particles that they want to be used
  – the user can chose the physics (processes) to assign to each particle

• Geant4 distribution includes the “physics_list” sub-library with many components and many predefined “reference” Physics Lists
  – Simulation results between different group of users may be compared
Modular Physics Lists

• Current recommendation to use Physics List via inheritance from G4VModularPhysicsList which derives from G4VUserPhysicsList

• Main public methods:
  – G4VModularPhysicsList::RegisterPhysics(G4VPhysicsConstructor*)
    • Addition of physics constructor
  – G4VModularPhysicsList::ReplacePhysics(G4VPhysicsConstructor*)
    • Replacement of the same type of physics constructor

• Constructor types:
  – Electromagnetic, EM extra (lepton-nuclear)
  – Decay, Radioactive Decay
  – Hadron elastic, hadron inelastic
  – Ion elastic and inelastic
  – Stopping of negatively charged particles
  – Step limiters (tracking cuts)
  – Optical
  – User may add custom constructor

• Physics List and its components are unique objects, which called in each thread two methods
  – G4VPhysicsConstructor::ConstructParticle()
  – G4VPhysicsConstructor::ConstructProcess()
  – Only const class members are allowed
Cuts definition

• In past cuts were defined in SetCuts() method of physics list
  – After migration to the MT mode, we recommend not doing this
  – Cuts may be defined via UI commands
  – Details on Geant4 cuts will be described below

• Using UI interface Geant4 kernel change cuts and try to count number of steps in the same run
  – /run/setCut 0.01 mm
  – /run/beamOn 100

• Define cuts only for electrons
  – /run/setCutForAGivenParticle e- 10 um
  – /run/setCutForRegion GasDetector 0.1 mm
  – /run/dumpCouples

• How to change low-energy limit of production threshold
  – /cuts/setLowEdge 0.1 keV
  – /cuts/setHighEdge 5 GeV
  – The highEdge limit cannot be above 10 GeV
    – Until now there was no need to increase this limit
Instantiation and ownership of physics objects

- **G4PhysicsListHelper** provides correct ordering for all processes from Geant4 libraries
  - `helper->RegisterProcess(G4VProcess*, G4ParticleDefinition*)`
- Custom process should be instantiated with defined ordering
  - `G4ParticleDefinition* particle;`
  - `G4ProcessManager* man = particle->GetProcessManager();`
  - `man->AddDiscreteProcess(G4VDiscreteProcess*); // added to the end`
  - `man->AddProcess(G4VProcess*, idxAtRest, idxAlongStep, idxPostStep);`
- Ownership of classes is not belonging to the Physics List class
  - `G4ParticleDefinition` classes are static shared between threads
  - `G4VProcess` classes are registered in process thread local store
  - Model classes for EM and hadronic physics are also registered in thread local stores
  - Hadronic cross sections are registered in another thread local store
  - All registrations and destructions are done automatically
- All processes, models, and cross section classes should be instantiated via “new”
  - Allowing sharing of processes/models between particles
  - Should not be included by object in any class
    - Does not guaranteed correct destruction order at different platforms
IONS AND EXOTIC PARTICLES
Geant4 Approach for Ions

- Light ions are individual Geant4 particles:
  - G4Deuteron
  - G4Triton
  - G4He3
  - G4Alpha

- Generic ion serves all other ions:
  - G4GenericIon - only one particle
  - Not a real particle (charge = +1, mass = Mp)
  - Serving for any kind of ion with Z>2
  - All concrete ions peak up processes and cross sections of the G4GenericIon
    - Scaling relations are used in run time

- Ion names
  - “C12” means that the carbon ion is in the ground state
  - “Co60[58.590]” is the first excitation state of Co60
  - Extra information about atomic shell may be filled to any ion
Exotic particles

- Not discovered particles are not part of Geant4 particle library
- To search exotics users should introduce non-existing particles in the user code
  - Such particles should be instantiated in `ConstructParticle()` method of one of custom G4VPhysicsConstructor, which is user responsibility
  - User should take care attaching processes to exotic particles in `ConstructProcess()` method
- Geant4 offers two extended examples
  - `$G4INSTALL/examples/extended/exoticphysics/monopole`
  - `$G4INSTALL/examples/extended/exoticphysics/dmparticle`
  - These examples demonstrate different variants of addition of extra particles and interactions
- In the monopole example additional classes are available for tracking of the magnetic monopole in magnetic field
  - `G4MonopoleTransportation`, `G4MonopoleEquation`
How tracks are created and killed

• G4Track can be created
  – By G4VUserPrimaryGeneratorAction
  – By any G4VProcess

• Geant4 particle is tracked until it is killed by one of Geant4 processes:
  – Transport out of the world volume
  – Inelastic interaction
  – Decay
  – Tracking low energy cut in the ionization process
  – G4NeutronKiller or G4UserLimits
    – If during tracking kinetic energy become zero and there is no processes AtRest
      the particle is killed by the stepping manager

• Any particle may be also killed by user action classes

• Geant4 introduced conception of “cut in range”
  – Physically this means required spatial accuracy of simulation
  – At initialization for each material a production threshold for kinetic energy of secondary
    particles is computed
    • This means different production thresholds for different materials
  – This is the main difference between Geant4 and other simulation tools, which
    implement only tracking cuts and cuts per volume
Cut and production thresholds for energy loss processes

- User defines cut in range expressed in units of length
- Using this range Geant4 kernel compute production threshold $T_{cut}$ for each material during initialization
- For a typical process (G4hIonisation, G4eIonisation, ...), the production threshold $T_{cut}$ subdivides the continuous and discrete parts of energy loss:
  - Mean rate of energy lost due to soft energy transfers
  - Cross section for discrete $\delta$-electron production
  - above $T_{cut}$
  - Both energy loss and cross sections are restricted

- At each step energy deposition is sampled by a fluctuation model using the computed mean energy loss
- Optionally, energy loss may be modified:
  - for the sampling of fluorescence and Auger–electrons emission
- 4-momentum balance is provided in all cases
What particles have cut in range?

- Cuts in range are defined for
  - Gamma
  - Electron
  - Positron
  - Proton

- Cut for proton is used for all hadrons and ions by elastic scattering processes
  - It is a cut on recoil ion kinetic energy

- By default, cut in range is defined globally
  - It is possible to have different cut in range for particle type
  - It is possible to define specific cut in range per G4Region
  - It is possible to set proton cut in range to zero
  - It is not possible to set other cuts below lowEdge limit
Which processes use cut in range?

• It is not mandatory to use cuts
  – They are needed to secure CPU performance of simulation
• Energy thresholds (derived from cut in range) are used
  – for gamma are used in Bremsstrahlung
  – for electrons are used in ionisation and e+e- pair production processes
  – for positrons is used in the e+e- pair production process
  – for gamma and electrons are used optionally (“ApplyCuts” options) in some discrete processes
    • Photoelectric effect, Compton, gamma conversion
• Production threshold for gamma and e+e- obtained from range cut cannot be whatever
  – The default low energy limit is 1 keV
  – The default high energy limit is 10 GeV
  – May be changed via UI command:
    • /cuts/setLowEdge 100 keV
• Energy threshold for protons are used to define the threshold for kinetic energy of a nuclear recoil
  – EM single scattering process
  – Hadron elastic scattering
Tracking cuts

• Additionally, to cut in range it is possible to use various tracking cuts
  – Unwanted particles may be killed after the step if corresponded flag is proposed
• In the default physics configurations two types of tracking cuts are applied:
  – Low-energy thresholds for charged particles by ionization 1 keV
  – Time cut for neutron transport 10000 ns
• Tracking cuts values are customizable and can be changed via UI commands
• User may easily setup extra tracking cut or step limiter
  – The best is to add an extra custom G4VProcess
  – **G4NeutronKiller** is the example
HOW TO CONTROL/MODIFY PHYSICS?
Information on Geant4 initialisation

• By default, detailed information on Geant4 physics configuration is printed
  – Tables of parameters
    • EM physics
    • Nuclear de-excitation module
    • Radioactive decay
    • Particle HP
  – EM processes and models parameters
    • For selected particles
  – Hadronic processes, models, and cross sections
    • For selected particles
  – Printout may be disabled via UI command
    • /process/had/verbose 0
    • /process/em/verbose 0

• Set of physics parameters and physics constructors may be modified before
  initialization of physics
  – Initialization of physics is triggered by the /run/initilize UI command
  – Between runs limited number of parameters may be changed
THANK YOU