Particles and Processes in Geant4

Vladimir Ivantchenko
*CERN, Geneva, Switzerland & Tomsk State University, Russia*

Geant4 Advanced Course
28 September 2020
Outline

• Geant4 interface to physics
  – Geant4 basic interfaces to physics
  – Geant4 particles
  – Geant4 processes
  – Physics Lists

• Electromagnetic (EM) physics
  – EM physics overview
  – EM physics constructors
  – User interface to EM 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
  - G4GenericIon is used to define physics for all other ions

- “Unstable” 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, life time, decay channels, etc.,
    - To be shared by all threads.
  - Dedicated object of G4ProcessManager: list of physics processes this particular kind of particle undertakes.
    - Physics process object must be thread-local.

<table>
<thead>
<tr>
<th>&lt;shared&gt;</th>
<th>&lt;static singleton&gt;</th>
<th>&lt;thread local&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>G4ParticleDefinition</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- G4double mass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- G4double charge</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- G4double lifetime</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Decay table</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- .....</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- G4int particleIndex</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G4PartDefSplitter</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Array of TLS pointers of G4ProcessManager</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- TLS pointer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- TLS pointer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- TLS pointer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TLS pointer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Proc man*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Proc man*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Proc man*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Proc man*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G4ProcessManager</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Process A*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Process B*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Process C*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Process D*</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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::GetSubType();
    - This method is recommended to be used for MC truth
    - The list of sub-types are only updated 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

- G4Track
  - Propagated by the tracking,
  - Snapshot of the particle state.

- G4DynamicParticle
  - Momentum, pre-assigned decay...

- G4ParticleDefinition
  - The « particle type »:
    - G4Electron,
    - G4PionPlus...

- G4ProcessManager
  - Holds pointers to processes
  - Implementing physics processes

- Process_1
- Process_2
- Process_3

Handled by kernel
Configured by you, in your "physics list"
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 processes will be valid down to below keV but some can be used only up 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
Geant4 Physics: Decay, Parameterized and Transportation

- 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 averaged events

- 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
    - Any process may change the **G4Track**, let’s see how
    - **G4Transportation** stops track at the volume boundary
G4VProcess: 3 kind of actions

- Abstract class defining the common interface of all processes in Geant4:
  - Used by all processes
    - including transportation, etc...
  - 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 \texttt{AtRest}, \texttt{AlongStep} and \texttt{PostStep} actions:
  – 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{G4VDDiscreteProcess}: only \texttt{PostStep} actions
      – \texttt{G4VContinuousDiscreteProcess}: \texttt{AlongStep} + \texttt{PostStep} actions
• **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 step and energy deposition at the step are defined
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
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
Displaying processes and particles

### When your application has started and when the run manager has been initialized, you can:

- Check what particles exist:
  - `/particle/list`

- Check a particle property:
  - `/particle/select e-`
  - `/particle/property/dump`

- Check the physics processes attached and their ordering:
  - `/particle/select e-`
  - `/particle/processes/dump`

- Please type “help” to get the full set of commands for particle category
Cuts definition

• 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
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 store
  - Hadronic cross sections are registered in another thread local store
  - All registrations are done automatically
- All processes, models, and cross section classes should be instantiated via “new”
  - Should not be included by object in any class
ELECTROMAGNETIC (EM) PHYSICS OVERVIEW
Gamma and electron transport

- **Photon processes**
  - $\gamma$ conversion into e+e- pair
  - Compton scattering
  - Photoelectric effect
  - Rayleigh scattering
  - *Gamma-nuclear interaction in* hadronic sub*-library (Bertini cascade)*

- **Electron and positron processes**
  - Ionization
  - Coulomb scattering
  - Bremsstrahlung
  - *Nuclear interaction in* hadronic sub*-library*
  - Positron annihilation

- Suitable for **HEP & many other Geant4 applications** with electron and gamma beams
Geant4 EM sub-libraries

Located in $G4INSTALL/sources/processes/electromagnetic

- **Standard**
  - $\gamma$, e up to 100 TeV
  - hadrons up to 100 TeV
  - ions up to 100 TeV
- **Muons**
  - up to 1 PeV
  - energy loss propagator
- **X-rays**
  - X-ray and optical photon production processes
- **High-energy**
  - processes at high energy (E>10GeV)
  - physics for exotic particles
- **Polarisation**
  - simulation of polarised beams
- **Optical**
  - optical photon interactions
- **Low-energy**
  - Livermore library $\gamma$, e- from 10 eV up to 1 GeV
  - Livermore library based polarized processes
  - PENELOE 2008 code rewrite, $\gamma$, e-, e+ from 250 eV up to 6 GeV
  - hadrons and ions up to 1 GeV
  - atomic de-excitation (fluorescence + Auger)
- **DNA**
  - Geant4 DNA modes and processes
  - Micro-dosimetry models for radiobiology
  - rom 0.025 eV to 10 MeV
  - many of them material specific (water)
  - Chemistry in liquid water
- **Adjoint**
  - sub-library for reverse Monte Carlo simulation from the detector of interest back to source of radiation
- **Utils**: general EM interfaces and helper classes
Software Design of EM Physics

• The uniform coherent approach for all EM packages
  – low energy and high energy models may work together

• A physical interaction or process is described by a process class
  – For example: G4ComptonScattering
    • Assigned to Geant4 particle types in Physics List
  – Three EM base processes:
    • G4VEmProcess
    • G4VEnergyLossProcess
    • G4VMultipleScattering

• A physical process can be simulated according to several models
  – each model being described by a model class
  – Naming scheme: « G4ModelNameProcessNameModel »
    • For example: G4LivermoreComptonModel
  – Models can be assigned to certain energy ranges and G4Regions
  – Inherit from G4VEmModel base class

• Model classes provide the computation of
  – Cross section and stopping power
  – Sample selection of atom in compound
  – Final state (kinematics, production of secondaries...)
The scalability of Geant4 application in the MT mode depends on how effectively data management is performed.

Shared EM physics data:
- Tables for cross sections, stopping powers, and ranges are kept by processes.
- Differential cross section data are kept by models.
- Material properties are in material data classes.
- EM parameters established for Physics Lists in the `G4EmParameters` class.

Tables are filled by Master and have read-only access in runtime.

In this scheme, the number of threads is not limited.
Gamma processes

- Photo-effect is the main process for absorption of low-energy gamma
  - Rayleigh scattering should not be neglected
- At high energy gamma conversion dominates
- Gammas may be absorbed by nuclei due to giant dipole resonance
Photo-electric effect – example of gamma process

In the photo-electric absorption process a **photon is absorbed** by an atom and an **electron is emitted** with an energy:

\[ E_{\text{photoelectron}} = E_\gamma - B_{\text{shell}}(Z_i) \]  

(1)

The atom, left in an excited state with a vacancy in the ionized shell, decays to its ground state through a cascade of radiative and non-radiative transitions with the **emission** of characteristic x-rays and **Auger and Coster-Kronig electrons**.
Electron processes

- At low energies ionisation dominates
- Above critical energy bremsstrahlung is the main process
  - Radiation energy loss exceed ionization energy loss
EM PHYSICS CONSTRUCTORS
A Physics list is the mandatory user class making the general interface between the physics the user needs and the Geant4 kernel.

List of particles: for which EM physics processes are defined
- $\gamma, e^\pm, \mu^\pm, K^\pm, p, \Sigma^\pm, \Xi^-, \Omega^-, \text{anti}(\Sigma^\pm, \Xi^-, \Omega^-)$
- $\tau^\pm, B^\pm, D^\pm, D_s^\pm, \Lambda_c^+, \Sigma_c^+, \Sigma_c^{++}, \Xi_c^+, \text{anti}(\Lambda_c^+, \Sigma_c^+, \Sigma_c^{++}, \Xi_c^+)$
- $d, t, \text{He3, He4, GenericIon, anti}(d, t, \text{He3, He4})$

The G4ProcessManager of each particle maintains a list of processes.

Geant4 provides several configurations of EM physics lists called constructors (G4VPhysicsConstructor) in the physics_lists library of Geant4.

These constructors can be included into a modular Physics list in a user application (G4VModularPhysicsList).
Standard EM Physics Constructors

Geant4 standard EM Physics Constructors for HEP applications
• Description of Coulomb scattering (the same):
  - $e^\pm$: Urban - MSC model below 100 [MeV] and the Wentzel - WVI + Single scattering (mixed simulation) model above 100 [MeV]
  - muon and hadrons: Wentzel - WVI + Single scattering (mixed simulation) model
  - ions: Urban - MSC model
• But different MSC stepping algorithms and/or parameters: speed v.s. accuracy

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Components</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>G4EmStandardPhysics</td>
<td>Default: nothing or _EM0 (QGSP_BERT, FTFP_BERT,…)</td>
<td>for ATLAS and other HEP simulation applications</td>
</tr>
<tr>
<td>G4EmStandardPhysics_option1</td>
<td>Fast: due to simpler MSC step limitation, cuts used by photon processes (FTFP_BERT_EMV)</td>
<td>similar to one used by CMS; good for crystals but not good for sampling calorimeters (i.e. with more detailed geometry)</td>
</tr>
<tr>
<td>G4EmStandardPhysics_option2</td>
<td>Experimental: similar to option1 with updated photoelectric model but no-displacement in MSC (FTFP_BERT_EMX)</td>
<td>similar to one used by LHCb</td>
</tr>
</tbody>
</table>
# EM Physics Constructors for medical applications

## Combined Geant4 EM Physics Constructors

- The primary goal is more the physics accuracy over the speed
- Combination of standard and low-energy EM models for more accurate physics description
- More accurate models for $e^\pm$ MSC (Goudsmit-Saunderson(GS)) and more accurate stepping algorithms (compared to HEP)
- Stronger continuous step limitation due to ionisation (as others given per particle groups)
- Recommended for more accuracy sensitive applications: medical (hadron/ion therapy), space

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Components</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>G4EmStandardPhysics_option3</td>
<td>Urban MSC model for all particles</td>
<td>proton/ion therapy</td>
</tr>
<tr>
<td>G4EmStandardPhysics_option4</td>
<td>most accurate combination of models (particle type and energy); GS MSC model with Mott correction and error-free stepping for $e^\pm$</td>
<td>the ultimate goal is to have the most accurate EM physics description</td>
</tr>
<tr>
<td>G4EmLivermorePhysics</td>
<td>Livermore models for $e^-$, $\gamma$ below 1 GeV and standard above; same GS MSC for $e^\pm$ as in option4</td>
<td>accurate Livermore based low energy $e^-$ and $\gamma$ transport</td>
</tr>
<tr>
<td>G4EmPenelopePhysics</td>
<td>PENELOPE models for $e^\pm$, $\gamma$ below 1 GeV and standard above; same GS MSC for $e^\pm$ as in option4</td>
<td>accurate PENELOPE based low energy $e^-$, $e^+$ and $\gamma$ transport</td>
</tr>
</tbody>
</table>
# EM Physics Constructors for testing of new models

Experimental Geant4 EM Physics Constructors

- Supposed to be used only by the developers for validations and model developments
- The main difference is in the description of the Coulomb scattering (GS, WVI, SS)

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Components</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>G4EmStandardPhysicsGS</td>
<td>standard EM physics and the GS MSC model for $e^\pm$ with HEP settings</td>
<td>may be considered as an alternative to EM0 i.e. for HEP</td>
</tr>
<tr>
<td>G4EmStandardPhysicsWVI</td>
<td>WentzelWVI + Single Scattering mixed simulation model for Coulomb scattering</td>
<td>high and intermediate energy applications</td>
</tr>
<tr>
<td>G4EmStandardPhysicsSS</td>
<td>single scattering (SS) model description of the Coulomb scattering</td>
<td>validation and verification of the MSC and mixed simulation models</td>
</tr>
<tr>
<td>G4EmLowEPPhysics</td>
<td>Monarsh University Compton scattering model, 5D gamma conversion model, WVI-LE model</td>
<td>testing some low energy models</td>
</tr>
<tr>
<td>G4EmLivermorePolarized</td>
<td>polarized gamma models</td>
<td>a (polarized) extension of the Livermore physics models</td>
</tr>
</tbody>
</table>
USER INTERFACE TO EM PHYSICS
EM parameters

- EM parameters of any EM physics list may be modified at initialization of Geant4 using C++ interface to the G4EmParameter class or via UI commands.

- Example of interfaces of G4EmParameters:
  - SetMuHadLateralDisplacement()
  - SetMscMuHadRangeFactor()
  - SetMscMuHadStepLimitType()

- Corresponding UI commands:
  - /process/msc/MuHadLateralDisplacement
  - /process/msc/RangeFactorMuHad
  - /process/msc/StepLimitMuHad

- Some other UI commands:
  - /process/em/deexcitationIgnoreCut true
  - /process/eLoss/UseAngularGenerator true
  - /process/em/lowestElectronEnergy 50 eV
  - /process/em/lowestMuHadEnergy 100 keV
  - ....
User Interfaces and Helper Classes

• Geant4 UI commands to define cuts and other EM parameters
• G4EmCalculator
  – easy access to cross sections and stopping powers (TestEm0)
• G4EmParameters
  – C++ interface to EM options alternative to UI commands
• G4EmSaturation
  – Birks effect (recombination effects)
• G4ElectronIonPair
  – sampling of ionisation clusters in gaseous or silicon detectors
• G4EmConfigurator
  – add models per energy range and geometry region
• G4NIELCalculator
  – Helper class allowing computation of NIEL at a step, which should be added in user stepping actions or sensitive detector (TestEm1)
How to extract Physics?

- Possible to retrieve Physics quantities using a **G4EmCalculator** object
- Physics List should be **initialized**
- Example for retrieving the total cross section of a process with name **procName**, for particle and material **matName**

```cpp
#include "G4EmCalculator.hh"
...
G4EmCalculator emCalculator;

G4Material* material =
    G4NistManager::Instance()->FindOrBuildMaterial(matName);
G4double density = material->GetDensity();
G4double massSigma = emCalculator.ComputeCrossSectionPerVolume(
    energy, particle, procName, material)/density;
G4cout << G4BestUnit(massSigma, "Surface/Mass") << G4endl;
```

- A good example: **$G4INSTALL/examples/extended/electromagnetic/TestEm0**
  Look in particular at the **RunAction.cc** class
THANK YOU