Geant4:
Electromagnetic Physics 4
V.Ivanchenko, BINP & CERN

• Process interface for developers
• Conception of interaction lengths
  • Interaction with G4 kernel
  • Examples
G4VProcess interface overview

- All processes inherit from base class G4VProcess
- Standard interface: a process provides *Interaction Lengths*, *StepLimits*, and *DoIt* methods
- Processes active *AlongStep*, *PostStep*, *AtRest*

- There are 7 types of processes with predefined interfaces:
  - ContinuousProcess
  - ContinuousDiscreteProcess
  - DiscreteProcess
  - RestContinuousDiscreteProcess
  - RestContinuousProcess
  - RestDiscreteProcess
  - RestProcess
G4VProcess interface

- **G4VProcess** defines 6 **pure virtual** methods:
  - `AtRestGetPhysicalInteractionLength(....)`
  - `AtRestDoIt(....)`
  - `AlongStepGetPhysicalInteractionLength(....)`
  - `AlongStepDoIt(....)`
  - `PostStepGetPhysicalInteractionLength(....)`
  - `PostStepDoIt(....)`

- There are also other **virtual** methods:
  - `IsApplicable( const G4ParticleDefinition&)`
  - `BuildPhysicsTable( const G4ParticleDefinition&)`
  - ....

- **G4VProcess** defined in `source/processes/management`
Number of interaction length

- **G4VProcess** and derived classes implement a scheme based on the «number of interaction length» to define the time/space point of the interaction and to choose the process of interaction
  - assumed for the processes dealing with the exponential law
- At the beginning of the tracking the process is given a sampled value «number of interaction length» $N_{\text{int}}$
- At the beginning of each step
  - The (concrete) process evaluates the current «mean free path» $l_{\text{free}}$, given the current material;
  - The «true path length» the process allows to the particle before the interaction occurs is then: $N_{\text{int}} \times l_{\text{free}}$
  - This value is returned by `GetPhysicalInteractionLength`;
Number of interaction length

- Then the step occurs with the actual step length $L_{\text{step}}$ value;
- At the beginning of the new step:
  - If the process has limited the previous step (i.e., its interaction occurred), it gets a new $N_{\text{int}}$ value;
  - Otherwise, the process converts back $L_{\text{step}}$ into a number of «consumed» interaction length, which is subtracted to its $N_{\text{int}}$ amount;
- Please review for example G4VDDiscreteProcess;
  - Note that all related methods are virtual, allowing to redefine them, if needed.
Example: G4VDiscreteProcess

```cpp
inline G4double G4VDiscreteProcess::PostStepGetPhysicalInteractionLength(
    const G4Track& track,
    G4double previousStepSize,
    G4ForceCondition* condition) {
    if ( (previousStepSize <=0.0) || (theNumberOfInteractionLengthLeft<=0.0)) {
        // beginning of tracking (or just after DoIt of this process)
        ResetNumberOfInteractionLengthLeft();
    } else {
        //subtract NumberOfInteractionLengthLeft
        SubtractNumberOfInteractionLengthLeft(previousStepSize);
        if (theNumberOfInteractionLengthLeft<perMillion) {
            theNumberOfInteractionLengthLeft=0.;
        }
    }
    //get mean free path
    _currentInteractionLength = GetMeanFreePath(track, previousStepSize, condition);
    G4double value = theNumberOfInteractionLengthLeft*currentInteractionLength;
    return value;
}
```
Some advices

• Do not overwrite GPIL virtual functions if it is possible!
• Try to use one of 7 predefined interfaces
• In these interfaces the accurate control on interaction length is provided for you
Interfaces to be used

• protected:
• // For all processes
• virtual G4double GetMeanFreePath(
  const G4Track& track,
  G4double previousStepSize,
  G4ForceCondition* condition
) = 0;

• // For continuous processes
• virtual G4double GetContinuousStepLimit(
  const G4Track& track,
  G4double previousStepSize,
  G4double currentMinimumStep,
  G4ForceCondition* condition
) = 0;
When process is active?

- All Continuous processes are invocated at each step of the particle
- If Discrete or Rest process limits the step then it is invocated
- To help to activate Rest or Discrete process at each step one should use G4ForceCondition
G4ForceConditions

- **GetPhysicalInteractionLength** methods involve `G4ForceCondition` & `G4GPIILSelection`;

- These are two enumerations:
  - They define signals, that processes send to the stepping, to require the treatment they wish from the stepping;
  - Involve ± «delicate» aspects;
  - Defined in `source/track`;
G4ForceConditions

- **G4ForceCondition** (AtRest and PostStep) defines requests for treatment of the DoIt methods.
- It can take the values:
  - **NotForced**: Usual case 😊: the DoIt method is invoked if the related GetPhysicalInteractionLength has limited the step
  - **Forced**: The related DoIt is applied if the particle is not killed
  - **Conditionally**: The PostStepDoIt is applied if the AlongStep has limited the step
  - **ExclusivelyForced**: Only the PostStepDoIt of the process is applied: all other AlongStep and PostStep are ignored
  - **StronglyForced**: The related DoIt is applied in any
G4GPISelection

• More delicate…

• **G4GPISelection** (*AlongStep*) defines requests for the treatment of the `GetPhysicalInteractionLength` methods.

• It can take the values:
  - **CandidateForSelection**: *usual case*: the process will be « declared » to have limited the step if it returns the smallest length;
  - **NotCandidateForSelection**: the process will not be « declared » to have limited the step, even if it returns the smallest length;

• In practice, only the multiple-scattering makes use of the « **NotCandidateForSelection** » signal up to now
Examples of processes

- G4hIonisation – notForced ContinuousDiscrete
- G4Decay – notForced RestDiscrete
- G4Cherenkov – Continuous
- G4Scintillation – Forced RestDiscrete
- G4MuonMinusCaptureAtRest – Rest
- G4ProtonInelasticProcess – notForced Discrete
Dolt signature

- virtual G4VParticleChange* AtRestDolt(
  const G4Track& track,
  const G4Step& step
) = 0;

- virtual G4VParticleChange* AlongStepDolt(
  const G4Track& track,
  const G4Step& step
) = 0;

- virtual G4VParticleChange* PostStepDolt(
  const G4Track& track,
  const G4Step& step
) = 0;
Dolt signature

- All **Dolt** methods have the same signature:
  - They receive `const G4Track` and `G4Step`
    - It is **not allowed** to change directly the track, nor the step
  - They return a `G4VParticleChange`:
    - This `G4VParticleChange` returns the changes of the track to the stepping
      - Not the « delta »!
    - It is assumed to **create of secondary G4Track**
    - Need to be familiar with, to implement a process 😊;
G4VParticleChange

- **G4VParticleChange** is defined in `source/track`
- It defines the virtual methods:
  - `virtual G4Step* UpdateStepForAtRest(G4Step*)`;
  - `virtual G4Step* UpdateStepForAlongStep(G4Step*)`;
  - `virtual G4Step* UpdateStepForPostStep(G4Step*)`;
- Which are used to communicate the changes to be applied on the primary;
  - They return the `G4Step` after having updated it;
- Each concrete **G4VParticleChange** should modify only the necessary members of the `G4Step`;
  - Can be relevant if your **G4VParticleChange** is often used;
G4VParticleChange

• To create **secondaries by the process**, the following methods have to be used:
  – `void SetNumberOfSecondaries(G4int);`
    • To declare the maximum number of secondaries which will be created by the process;
  – `void AddSecondary(G4Track* aSecondary);`
    • Which has to be called for each secondary created;

• **G4VParticleChange** has a method `Initialize(const G4Track&)` which is used to initialize the members which will be changed by the process
G4TrackStatus

- **G4TrackStatus** defines the possible status a track can undertake;
- It is needed when writing a process:
  
  - `fAlive,`  // Continue the tracking
  - `fStopButAlive,`  // Invoke active rest physics processes
  - `and`  // and kill the current track afterward
  - `fStopAndKill,`  // Kill the current track
  - `fKillTrackAndSecondaries,`  // Kill the current track and also
  - `// associated secondaries.`
  - `fSuspend,`  // Suspend the current track
  - `fPostponeToNextEvent`  // Postpones the tracking of the current
  - `// track to the next event.`
Example with G4GammaConversion

• Example with G4GammaConversion, which uses a particle change defined in the base class G4VDiscreteProcess;

```cpp
G4VParticleChange* G4GammaConversion::PostStepDoIt(
    const G4Track& aTrack,
    const G4Step& aStep)
{
    aParticleChange.Initialize(aTrack);
    //Does the physics…
    aParticleChange.SetNumberOfSecondaries(2);
    //…
    G4double localEnergyDeposit = 0.;

    if (ElectKineEnergy > fminimalEnergy)
    { //…
        // create G4DynamicParticle object for the particle1
        G4DynamicParticle* aParticle1 = new G4DynamicParticle(
            G4Electron::Electron(), ElectDirection, ElectKineEnergy);
        aParticleChange.AddSecondary(aParticle1);
    } else { localEnergyDeposit += ElectKineEnergy;}
```
// the e+ is always created (even with Ekine=0) for further annihilation.

// ...
if (PositKineEnergy < fminimalEnergy)
{ localEnergyDeposit += PositKineEnergy; PositKineEnergy = 0.;}

// ...
// create G4DynamicParticle object for the particle2
G4DynamicParticle* aParticle2 = new G4DynamicParticle(
    G4Positron::Positron(), PositDirection, PositKineEnergy);
aParticleChange.AddSecondary(aParticle2);

aParticleChange.SetLocalEnergyDeposit(localEnergyDeposit);

// // Kill the incident photon
//
aParticleChange.SetEnergyChange( 0. );
aParticleChange.SetStatusChange( fStopAndKill );
return G4VDiscreteProcess::PostStepDoIt( aTrack, aStep );
Some remarks

• It is not necessary to know Geant4 kernel in order to implement a new process
• One have to follow the described interfaces
• Having several implementation for given process is normal for Geant4
Conclusion remarks

- The toolkit provides a wide choice of processes, so try to use existing processes
- User can substitute any of existing processes
- It is assumed that Geant4 user is at the same time a developer
- Geant4 team will appreciate an efforts of any user to implement his/her own process or model, if it is correct from physics point of view