

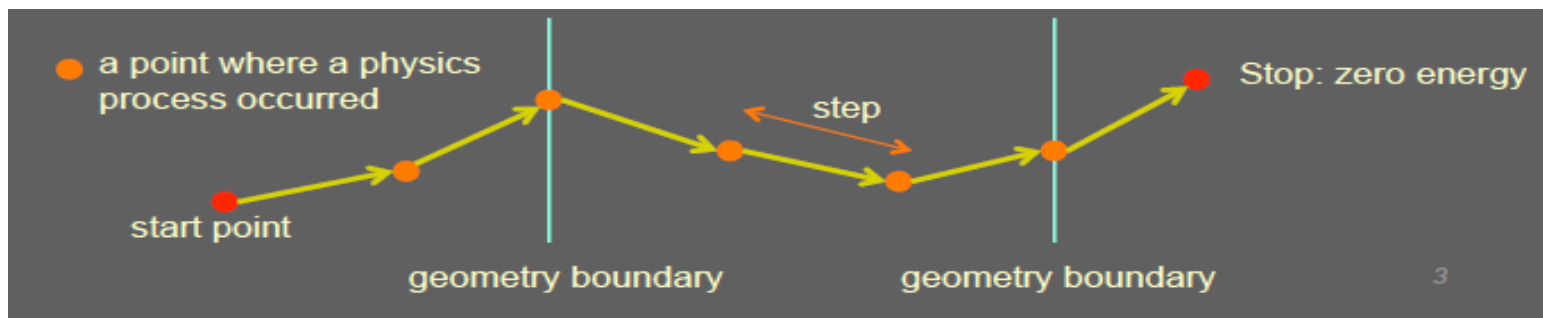
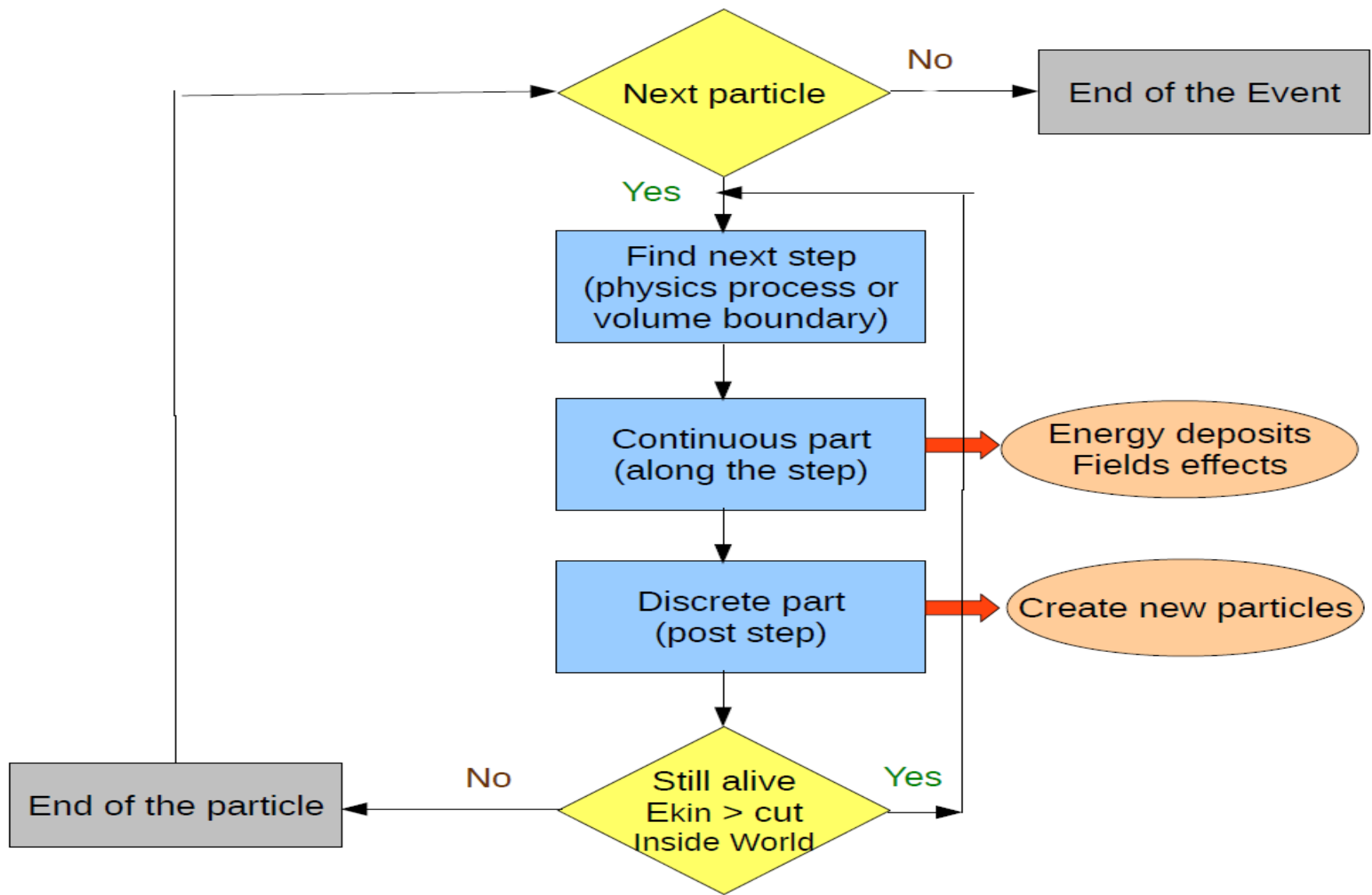
Tracking

Witold Pokorski, Alberto Ribon
CERN PH/SFT



Where to Find Information

- Geant4 User Guides
 - geant4.web.cern.ch/geant4/support/userdocuments.shtml
 - User's Guide: For Application Developers
 - Physics Reference Manual
 - ...
- User Support
 - geant4.web.cern.ch/geant4/support/index.shtml
 - Bug reports and fixes
 - ...
- HyperNews Forum
 - hypernews.slac.stanford.edu/HyperNews/geant4/cindex
 - Discussion between users and developers



Classical vs Quantum approach

- In Geant4, a particle that flies through a detector is treated as a **classical particle**, i.e. **not a wave function**, but a point-like object which has a well-defined momentum at each instant:
 - Space-time position (x, y, z, t)
 - Energy-momentum (p_x, p_y, p_z, E)

This is a reasonable **approximation**, given that in most practical situations particles are seen as “**tracks**” in macroscopic detectors

- Geant4 is based on a **semi-classical** approach, because the particles are treated classically, but their interactions - cross sections and final states - take often into account the *results* (not the computation) of **quantum-mechanical** effects

Run, Event,
Particle, Track,
Step, StepPoint
Trajectory, TrajectoryPoint

Run in Geant4

- A **run** is a collection of events
 - Consists of **one event loop**
 - Starts with the ***/run/beamOn*** command
- Within a run, conditions do not change, i.e. the user cannot change:
 - the detector setup
 - the settings of physics processes
- A run in Geant4 is represented by the class **G4Run** or a user-defined class derived from it
- **G4RunManager** is the manager class
- **G4UserRunAction** is the optional user hook

Event in Geant4

- An **event** is the basic unit of simulation in Geant4
- At beginning of processing, primary tracks are generated; these are pushed into a stack
- A track is popped up from the stack one by one and tracked; resulting secondary tracks are pushed into the stack
 - this tracking lasts as long as the stack has a track
- When the stack becomes empty, the event is over
- The class **G4Event** represents an event; it has the following objects at the end of its (successful) processing:
 - List of primary vertices and particles (as input)
 - Hits and Trajectory collections (as output)
- **G4EventManager** is the manager class
- **G4UserEventAction** is the optional user hook

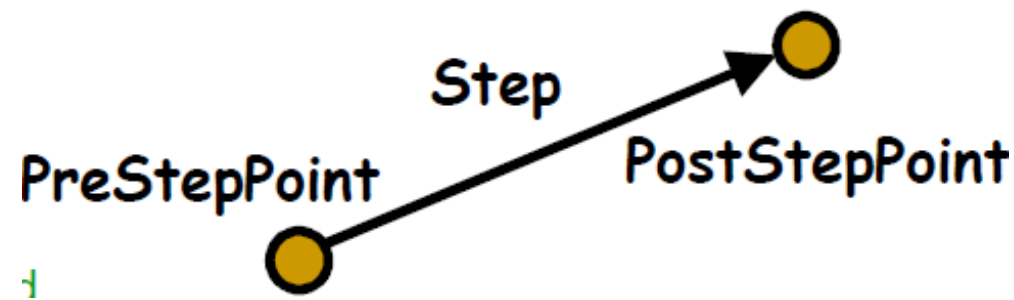
Particle in Geant4

- A **particle** in Geant4 is represented by 3 layers of classes
- **G4Track**
 - Position, geometrical information, etc.
 - This is a class representing a particle to be tracked
- **G4DynamicParticle**
 - "Dynamic" physical properties of a particle: momentum, energy, spin...
 - Each G4Track object has its own G4DynamicParticle object
 - This is a class representing an individual particle
- **G4ParticleDefinition**
 - "Static" properties of a particle: charge, mass, lifetime, etc.
 - **G4ProcessManager** describes the processes involving this particle
 - All G4DynamicParticle objects of the same kind of particle share the same G4ParticleDefinition

Track in Geant4

- A **track** is a snapshot of a particle
 - It has the physical quantities corresponding only to the current instance; it does not record previous quantities
 - Step is a “delta” information to a track; a track is not a collection of steps; instead, a track is updated by steps
- A track object is deleted when
 - it goes out of the world volume
 - it disappears (e.g. decay, inelastic scattering)
 - it goes down to zero kinetic energy and no “AtRest” additional process is required
 - the user decides to kill it artificially
- No track object persists at the end of event
 - For recording tracks, use trajectory class objects
- **G4Track** class represents a track
- **G4TrackingManager** is the manager class
- **G4UserTrackingAction** is the optional user hook

Step in Geant4



- A step has two points and also “delta” information of a particle (energy loss on the step, time-of-flight spent by the step, etc.)
 - A point is represented by the **G4StepPoint** class
- Each point knows the volume (and material). In case a step is limited by a volume boundary, the end point physically stands on the boundary, and it logically belongs to the next volume
 - Because one step knows the materials of two volumes, boundary processes such as transition radiation or refraction can be simulated
- **G4Step** represents a step
- **G4SteppingManager** is the manager class
- **G4UserSteppingAction** is the optional user hook

Trajectory and Trajectory Point in Geant4

- Track does not keep its trace.
No track object persists at the end of an event
- **G4Trajectory** is the class which copies some of the **G4Track** information and persist till the end of an event
- **G4TrajectoryPoint** is the class which copies some of the **G4Step** information and persist till the end of an event
 - G4Trajectory has a vector of G4TrajectoryPoint objects
 - With the command: */tracking/storeTrajectory 1* at the end of event processing, G4Event has a collection of G4Trajectory objects; useful mainly for visualization
- Be careful not to store too many trajectories: memory growth
- G4Trajectory and G4TrajectoryPoint as provided by Geant4 store only the minimum information
 - users can create their own trajectory / trajectory point to store information they need

Tracking

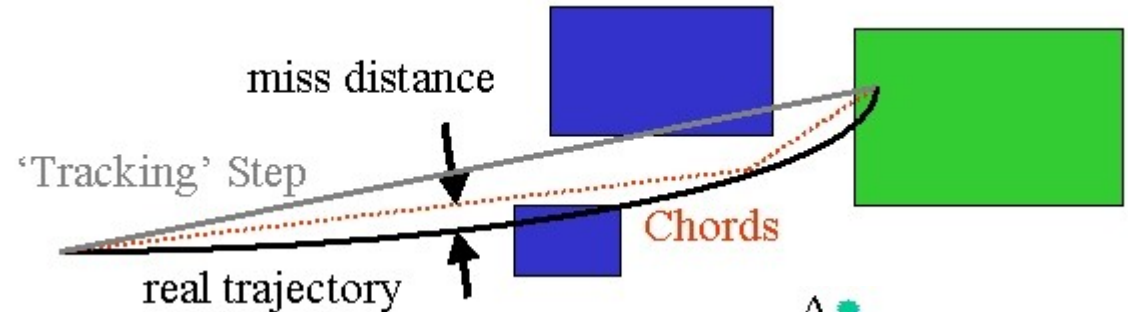
Propagation in a Field (1)

- Geant4 is capable of propagating tracks in a variety of fields:
 - **magnetic**, **electric**, **electromagnetic**, and **gravity** fields
 - **uniform** or **non-uniform** (in space and/or time)
 - with user-defined accuracy (trade-off between accuracy and performance)
- In order to propagate a track inside a field, the **equation of motion** of the particle in the field is integrated
 - In general, this is done using a **Runge-Kutta method** for the integration of ordinary differential equations
 - In ***examples/extended/field/*** you can see some examples of magnetic, electric and gravity fields
 - The user can also create their own type of field, inheriting from **G4VField**, and specifying its associated Equation of Motion, inheriting from the class **G4EqRhs**

Propagation in a Field (2)

The curved path, in a tracking step, is broken up into linear chord segments

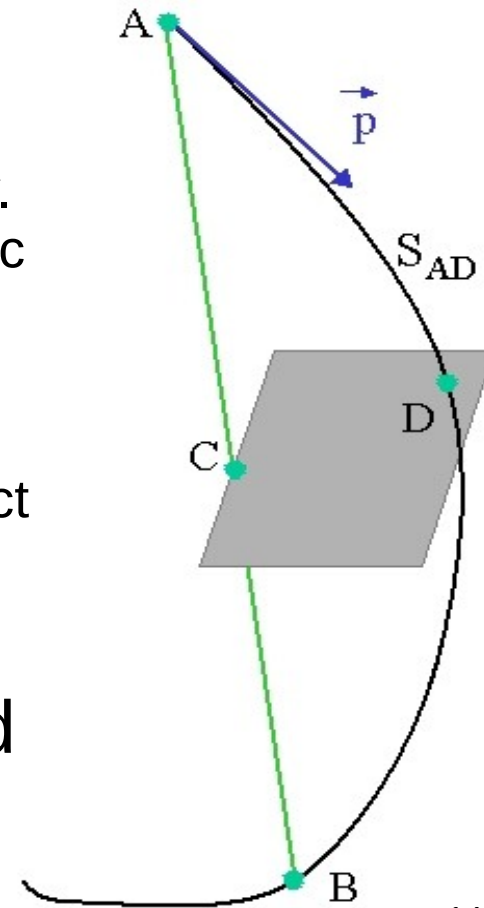
miss distance: maximum estimated distance between curve and chord



delta intersection: maximum estimated distance (CD) between the curve and chord intersection on a volume boundary. This is an important parameter, related to the potential systematic errors in the momentum of reconstructed tracks.

delta one step: maximum estimated distance between the endpoint of an 'ordinary' integration step, which does not intersect a volume boundary (i.e. a physics step), and the curve endpoint

These are some of the parameters of the Field Manager, which the user can tune to optimise between accuracy and performance



Propagation in a Field (3)

- Choosing a **field** :
 - Uniform fields: **G4UniformMagField**, **G4UniformElectricField**, **G4UniformGravityField**
 - Non-uniform fields: concrete classes derived from:
G4MagneticField, **G4ElectricField**,
G4ElectroMagneticField, **G4Field** ;
must define the method: *void GetFieldValue(...)*
- Choosing a **stepper** :
 - Runge-Kutta integration is used to compute the motion in a general field. There are many general steppers from which to choose, of low and high order, and specialized steppers for pure magnetic fields
 - General: **G4DormandPrince745** (default), **G4ClassicalRK4** ,
G4SimpleRunge, **G4SimpleHeum**, **G4CashKarpRKF45**
 - Specialized for pure magnetic fields: **G4NystromRK4**,
G4HelixImplicitEuler, **G4HelixExplicitEuler**,
G4HelixSimpleRunge, etc.

Propagation in a Field (4)

- Example of how to create a global magnetic field

```
G4UniformMagField* magField =  
    new G4UniformMagField( G4ThreeVector( 0.0, 0.0, 4.0*Tesla ) );  
G4FieldManager* fieldMgr =  
    G4TransportationManager::GetTransportationManager()->GetFieldManager();  
fieldMgr->SetDetectorField( magField );  
fieldMgr->CreateChordFinder( magField ); // with default parameters
```

- Example of how to create a local magnetic field

```
logicVolume->SetFieldManager( localFieldManager, true );
```

- For more examples see: [*examples/extended/field/*](#)
 - Tracking in magnetic field
 - Tracking in electric field
 - Tracking in overlapping fields (electric and magnetic)
 - Tracking in gravity field

Production Cuts (1)

- **Production cuts** for secondaries can be specified as **range cuts**, which are converted (at initialization) into **energy thresholds** (material-dependent) for secondary **gammas, electrons, positrons** and **protons**
- For **electrons** and **gammas**, production cuts are absolutely needed in, respectively, **ionization** and **bremsstrahlung** processes to avoid the **infrared singularity**
 - $\sigma_{\text{brems}} \sim 1/E_{\gamma}$; $\sigma_{\text{ionization}} \sim 1/(T_e * T_e)$
- For **positrons**, the production cut is almost always ignored
 - i.e. positrons are always produced in e+e- pair-production, regardless of their energy (range) (because, in matter, they annihilate (even at rest) and always produce a pair of gammas that can fly...)
 - except for very high production cuts on gamma (greater than the electron mass, which is the minimum energy of each of the two gammas generated by a positron-electron annihilation...)
 - Use-case is for underground experiments, stopping positron in the mountain above

Production Cuts (2)

- For **protons**, it has the following meaning:
if a hadron scatters elastically on a nucleus (of the detector material), this (recoiling) nucleus becomes a new G4Track (i.e. a particle to be transported by Geant4) only if its kinetic energy is above the value:

$$(100 * keV) * proton_production_cut_in_mm$$

- If all these cases, whenever a secondary particle is not produced because it is below the production cut, then its kinetic energy contributes to the so-called “**continuous energy deposition**” or “**along the step**”
 - As for the concept of “step”, also “continuous energy deposition” or “along the step” does not correspond to anything physically
 - But it is a convenient artifact to speed up the simulation

Which Processes are Using Cuts?

- Energy threshold for *gammas* is used in **bremsstrahlung**
- Energy threshold for *electrons* is used in **ionisation** and **e+e- pair-production** processes
- Energy threshold for *positrons* is used in the **e+e- pair-production** process
- Energy thresholds for *gammas* and *electrons* can be used in **all electromagnetic discrete processes** if the “**ApplyCuts**” option is activated
 - Photoelectric effect, Compton, etc.
- Energy threshold for *protons* are used in processes of **elastic hadron-nucleus scattering**
 - defining the kinetic energy threshold of nuclear recoil

How to Set Production Cuts?

- A range cut value is set by default to **0.7 mm** in Geant4 reference physics lists
- This can be changed via UI (User Interface) command, e.g.:

/run/setCut 0.05 mm

- You can set a different value for each particle type, e.g.:

/run/setCutForAGivenParticle e- 0.05 mm

/run/setCutForAGivenParticle e+ 0.01 mm

/run/setCutForAGivenParticle gamma 1.0 cm

/run/setCutForAGivenParticle proton 0.2 mm

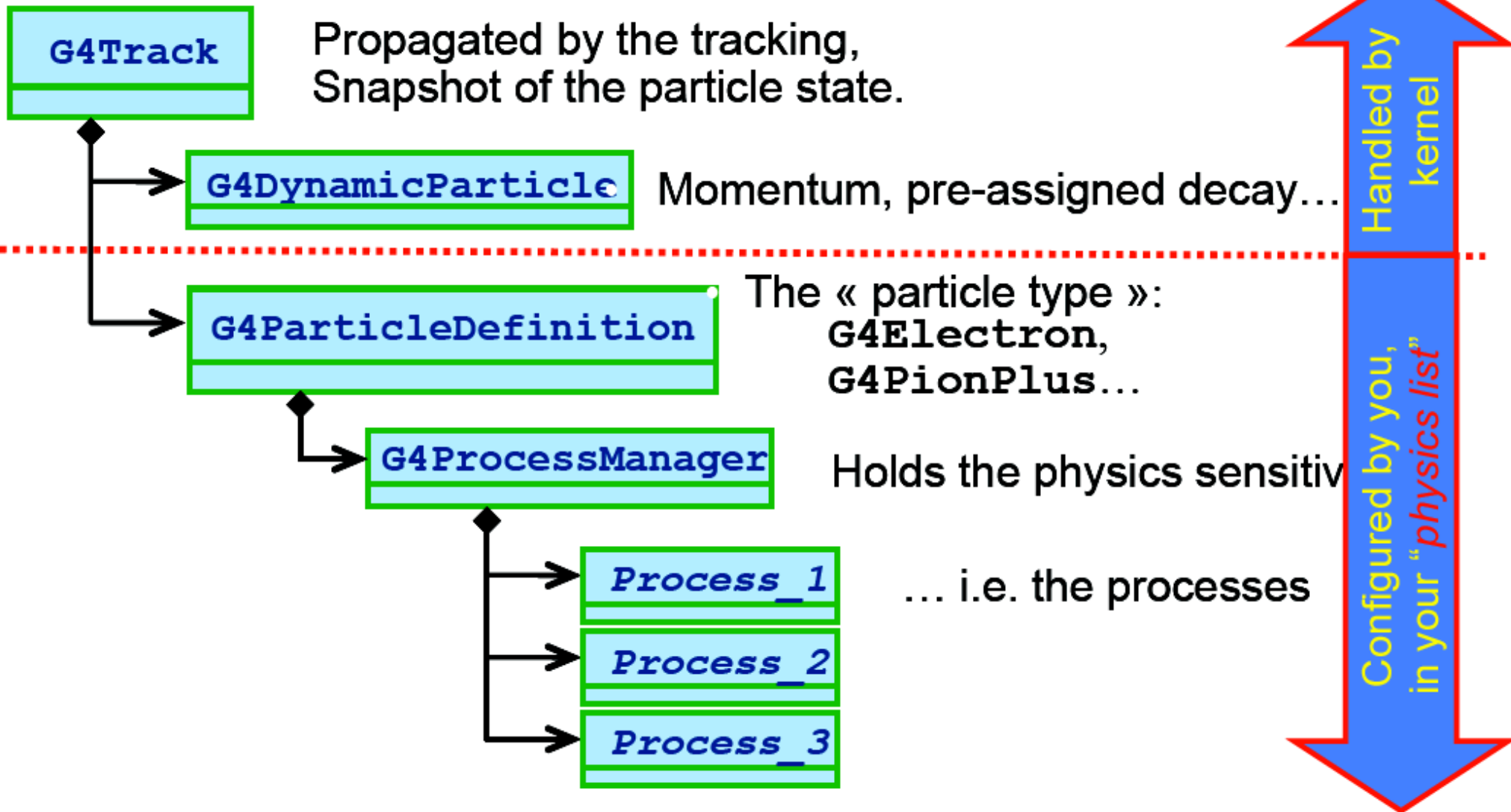
- Production cuts can be set globally or per-region
- For a complex detectors, the optimization of the range cuts per region is crucial for the CPU performance of the simulation

Special Tracking Cuts

- By default in Geant4, there are **only production cuts**, and **not tracking cuts**: the produced particles are tracked down to zero kinetic energy (range)
 - The treatment is reliable down to **~1 keV** : below it is approximated
- For optimization reasons, a user may limit the tracking of particular particle types in specified volumes
- The approach is as follows: special user cuts are registered in the **G4UserLimits** class, which is associated with the logical volume class. The current default list is:
 - max allowed step size
 - max total track length
 - max total time of flight
 - min kinetic energy
 - min remaining range
- For an example, see: ***examples/basic/B2***

Processes

Track and Processes



Processes: 3 kinds of Actions

- Abstract class **G4VProcess** defines the common interface of all processes in Geant4
 - including Transportation
- Defines three kinds of actions:
 - **AtRest** actions
 - Decay at rest, e+ annihilation at rest, nuclear capture at rest, etc.
 - **AlongStep** actions
 - “Continuous” energy deposition (e.g. production below threshold); fields effect
 - **PostStep** actions
 - In-flight decays and interactions
- G4ProcessManager has 3 vectors of actions (per particle-type):
 - one for **AtRest** actions: these processes **compete**
 - one for **AlongStep** actions: these processes **cooperate**
 - one for **PostStep** actions: these processes **compete**



G4VProcess: action methods

- A process will implement any combination of the **3 actions**:

- **AtRest**
- **AlongStep**
- **PostStep**

e.g. decay : AtRest + PostStep

- Each action defines **2 methods**:

- **GetPhysicalInteractionLength()**

- Used to **limit the step size**

- because the process triggers an interaction, a decay, geometry boundary, a user's limit, etc.
- The cross section for a in-flight physics process, or the mean lifetime for an at-rest process is used

- **Dolt()**

- Implements the **actual action** to be applied to the track

- typically the generation of the final state

Ordering of the Processes

- Process ordering, in general, is not critical...
- ... except for multiple-scattering and transportation
- Assuming n processes, the ordering of the *AlongStepGetPhysicalInteractionLength* should be:

[$n-2$] ...

[$n-1$] multiple scattering (before last)

[n] transportation (last)

- Why?

- Processes return a “true path length”
- The multiple scattering virtually folds up this true path length into a shorter “geometrical path length”
- Based on this new length, the transportation can geometrically limit the step

