Experiment Simulation

CERN School of Computing 2005
Saint Malo

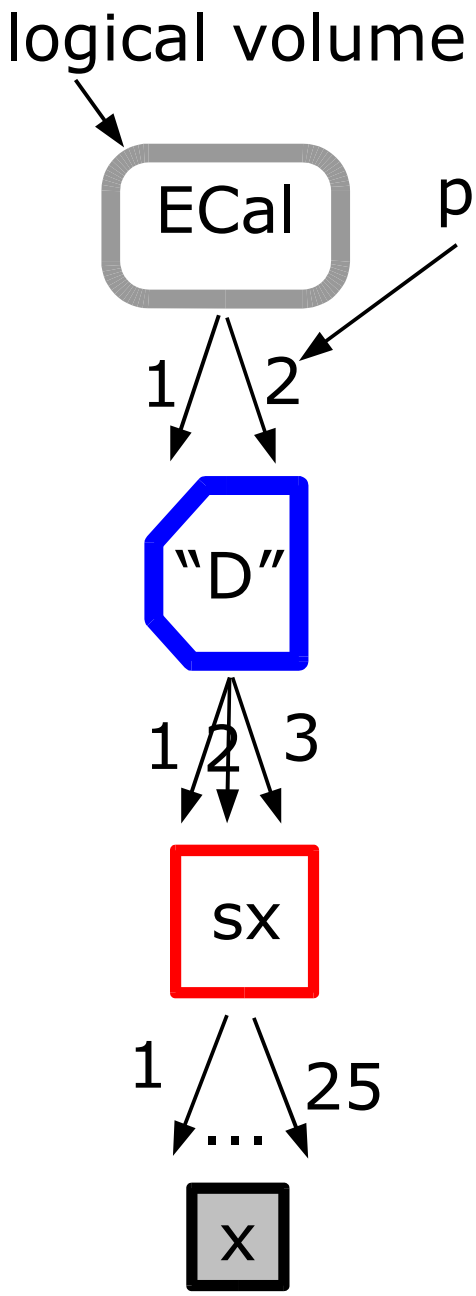
Lecture 4

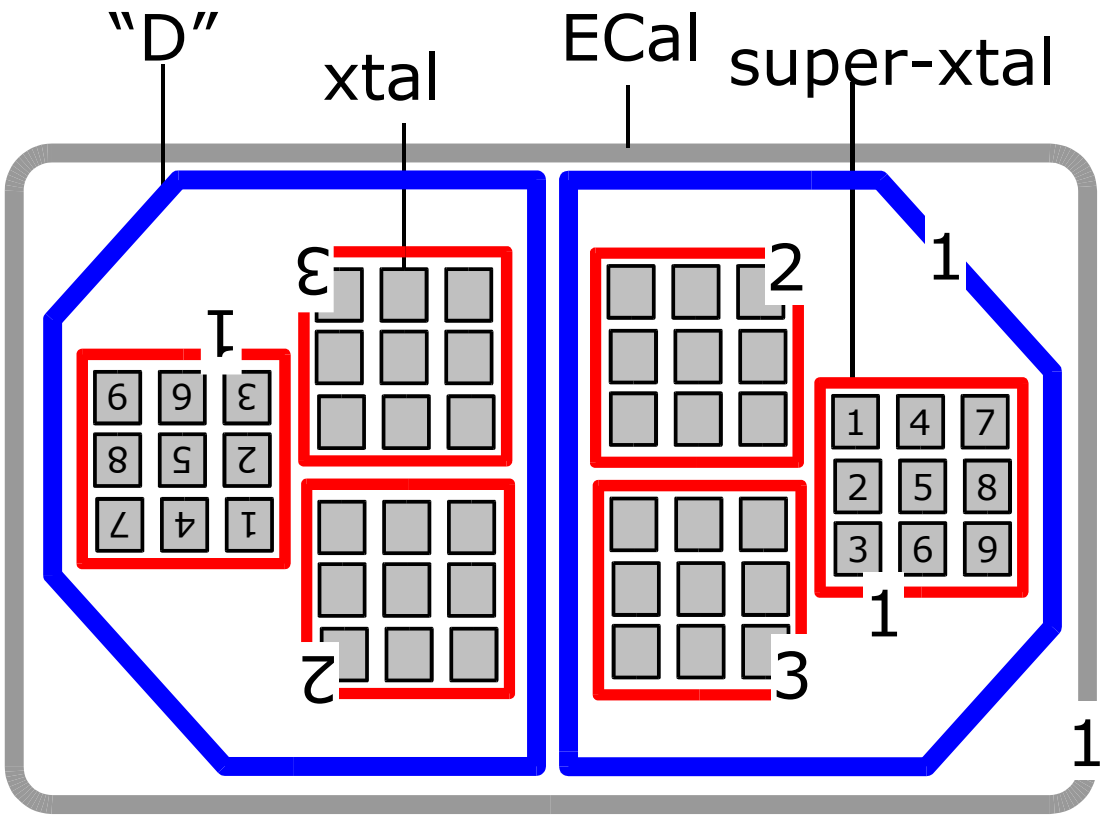
Martin Liendl
SWX Swiss Exchange

Unfinished Business!

- A – Physics Model in GEANT4
  - Stepping: moving in free path lengths
  - Physics Processes
- B – Detector Description in GEANT4
  - Solids/Shape Model
  - Volumes
  - Hierarchy of Volumes
- Combining A + B
  - Stepping through a detector description

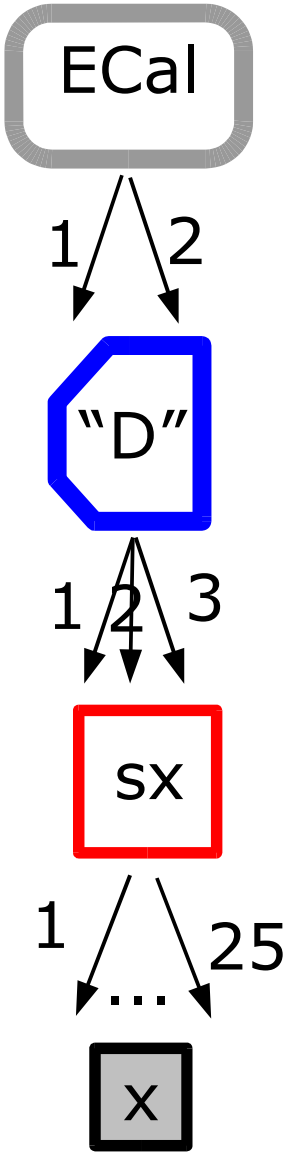
Graph Structure



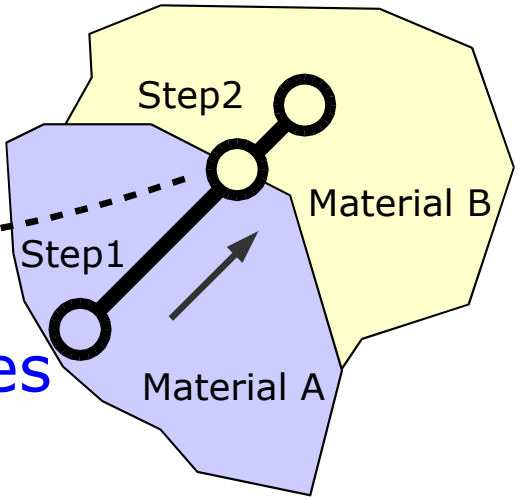


simplified version of an ECal ...

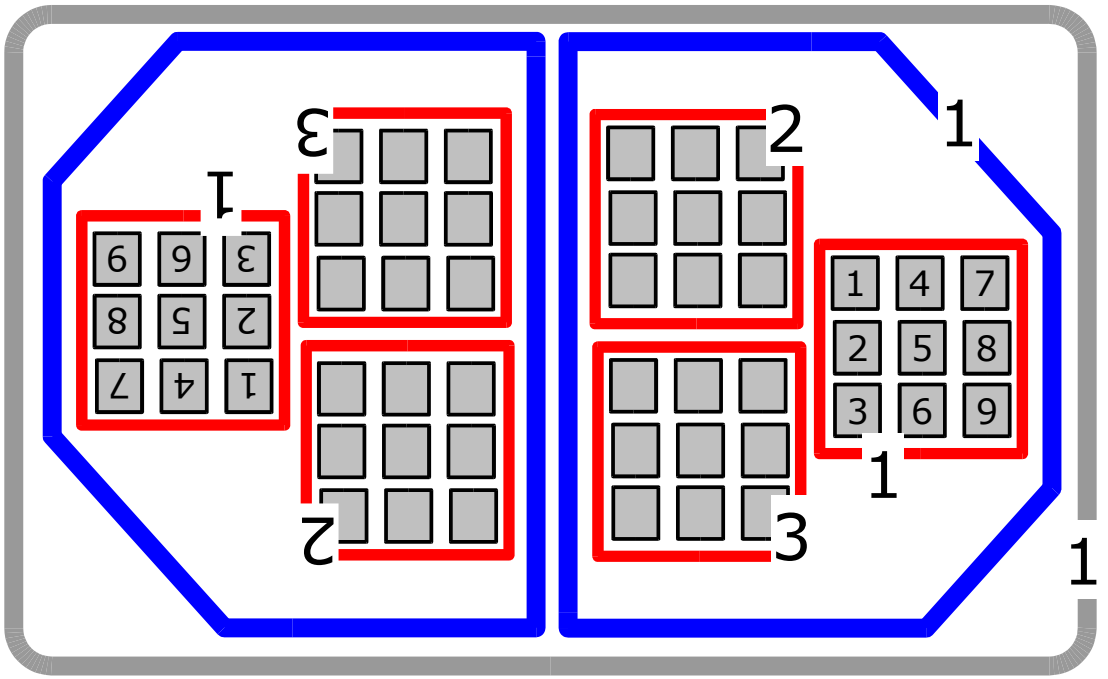
Stepping & Geometry



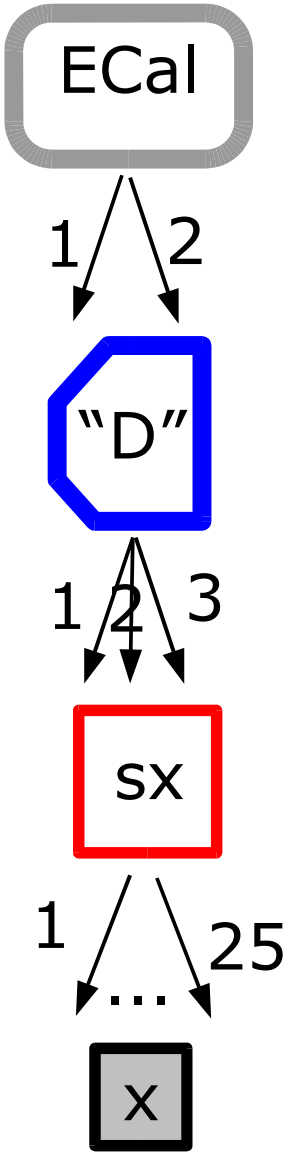
PostStepPoint1
=
PreStepPoint2

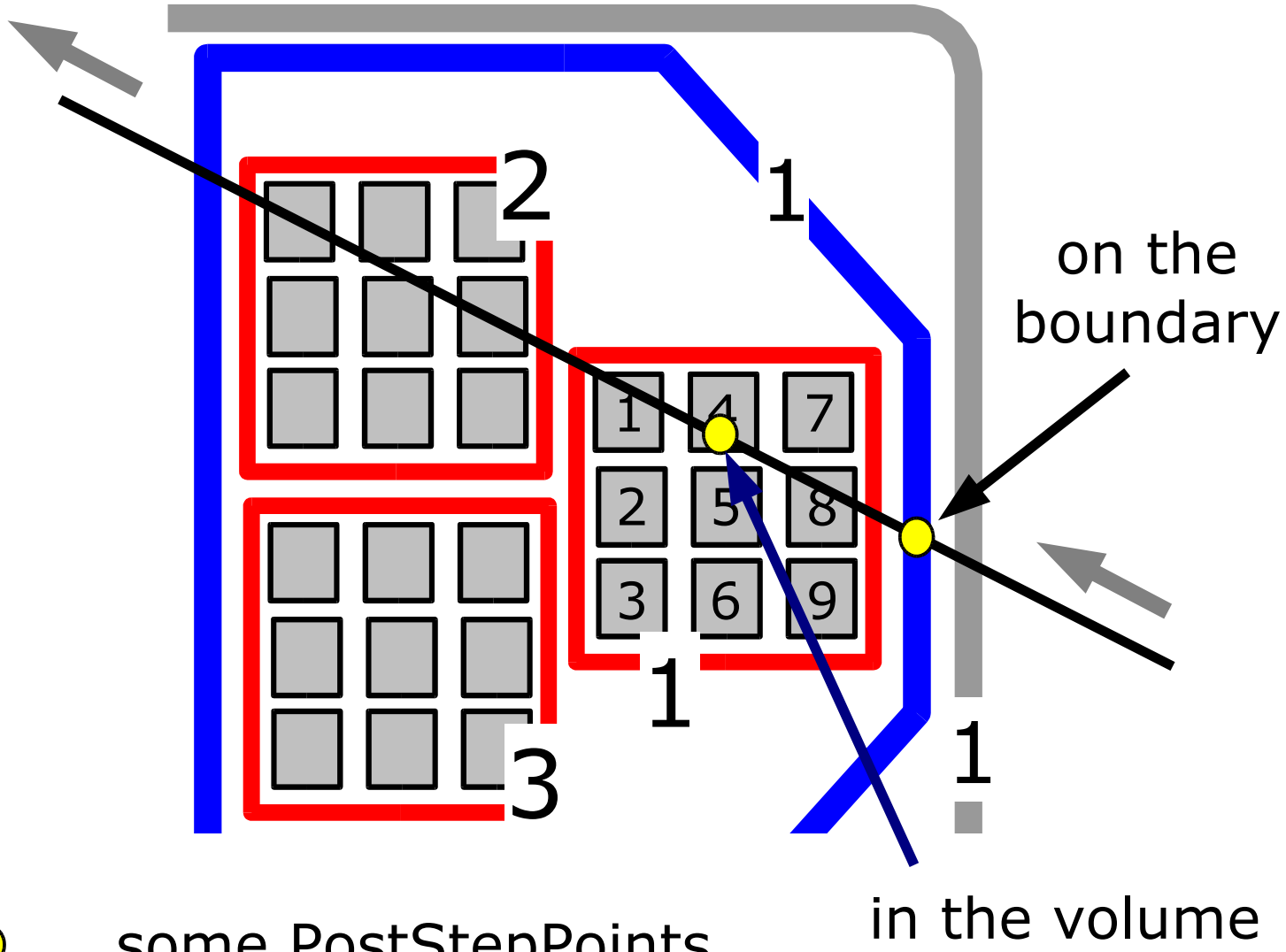


Remember?
Steps & StepPoints
determined by boundaries
and physics processes!

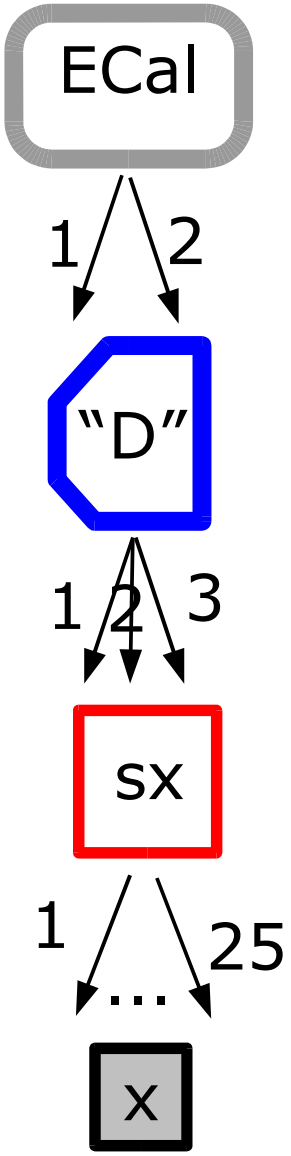


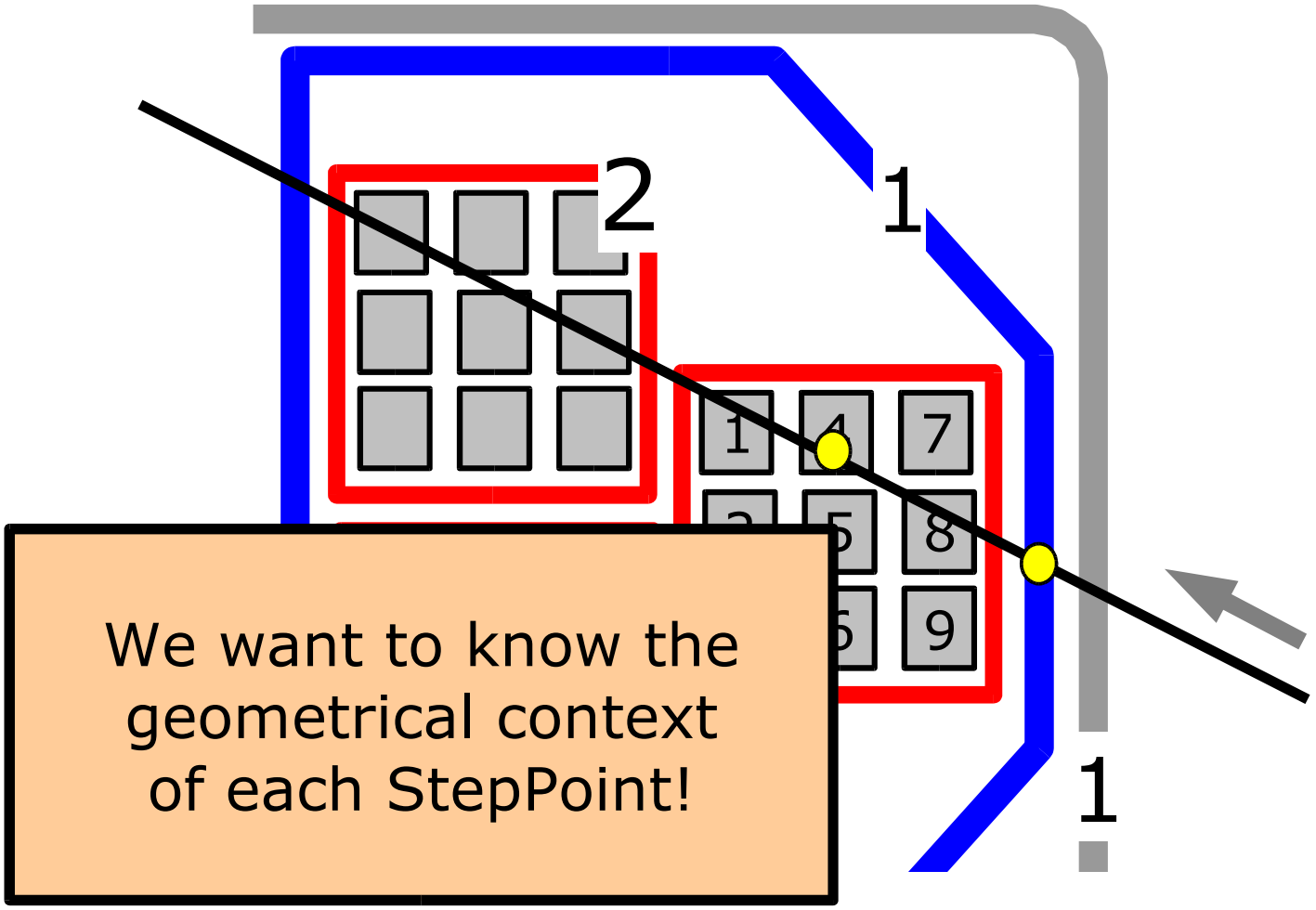
Stepping & Geometry





Stepping & Geometry

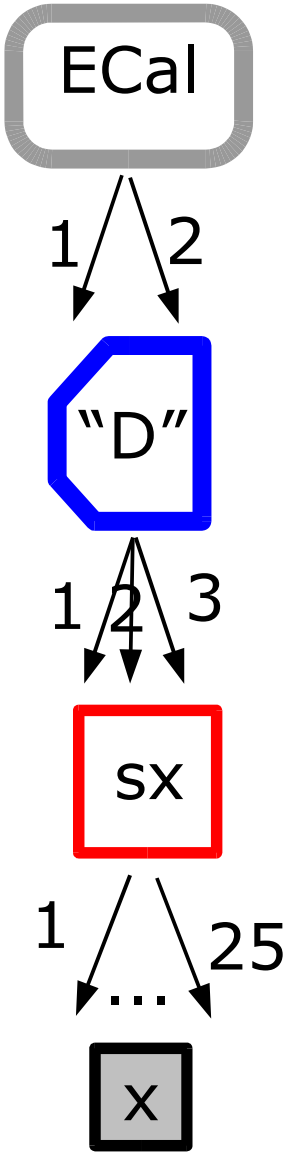


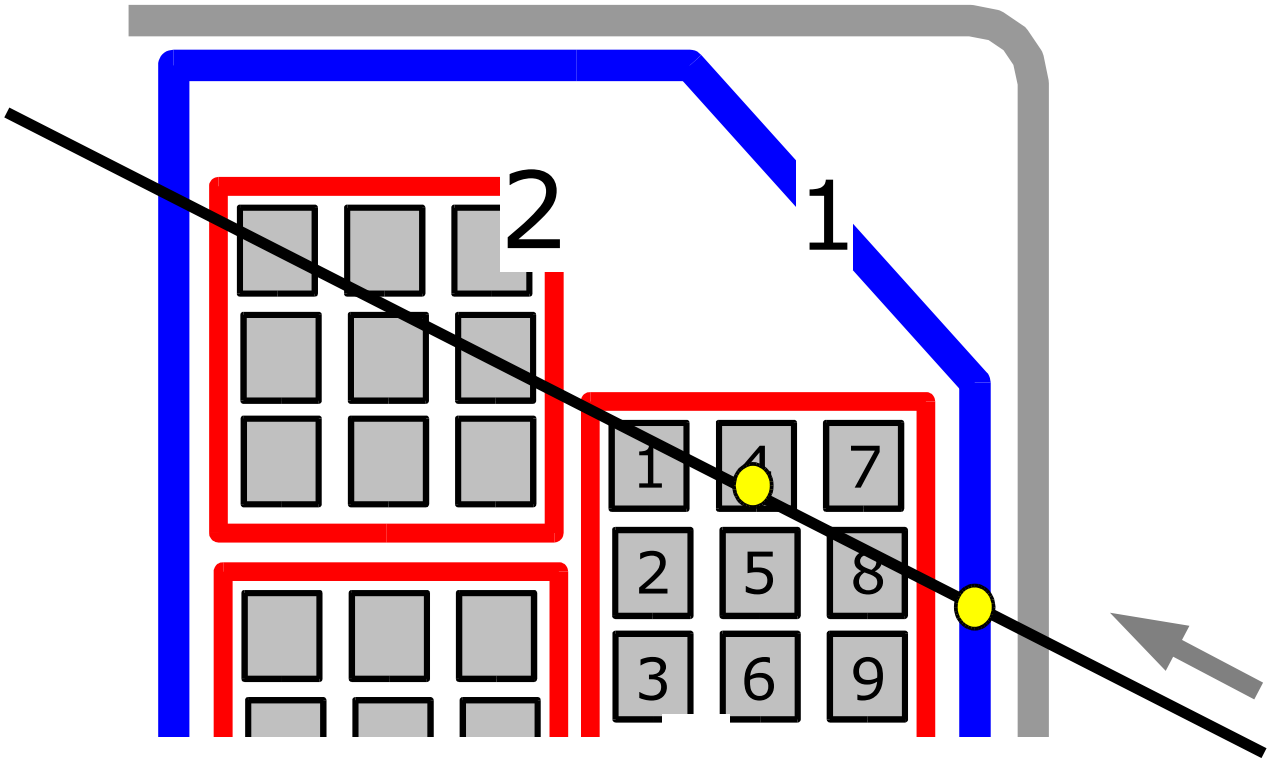


We want to know the geometrical context of each StepPoint!

... some PostStepPoints

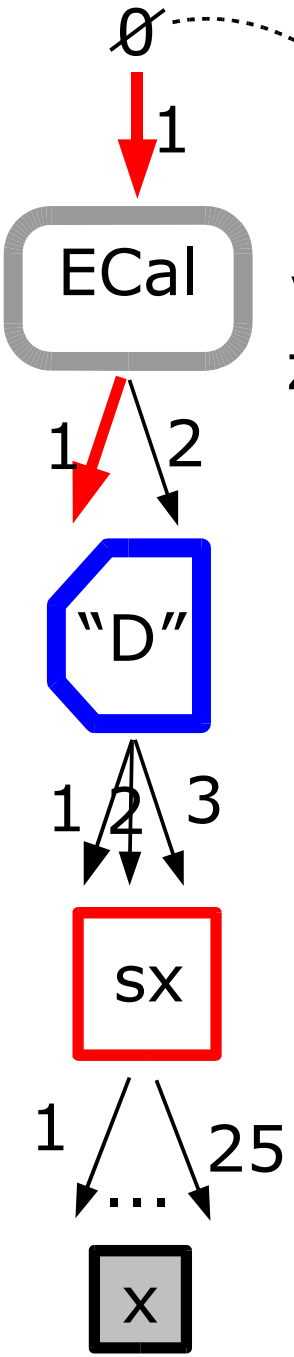
Touchable-History



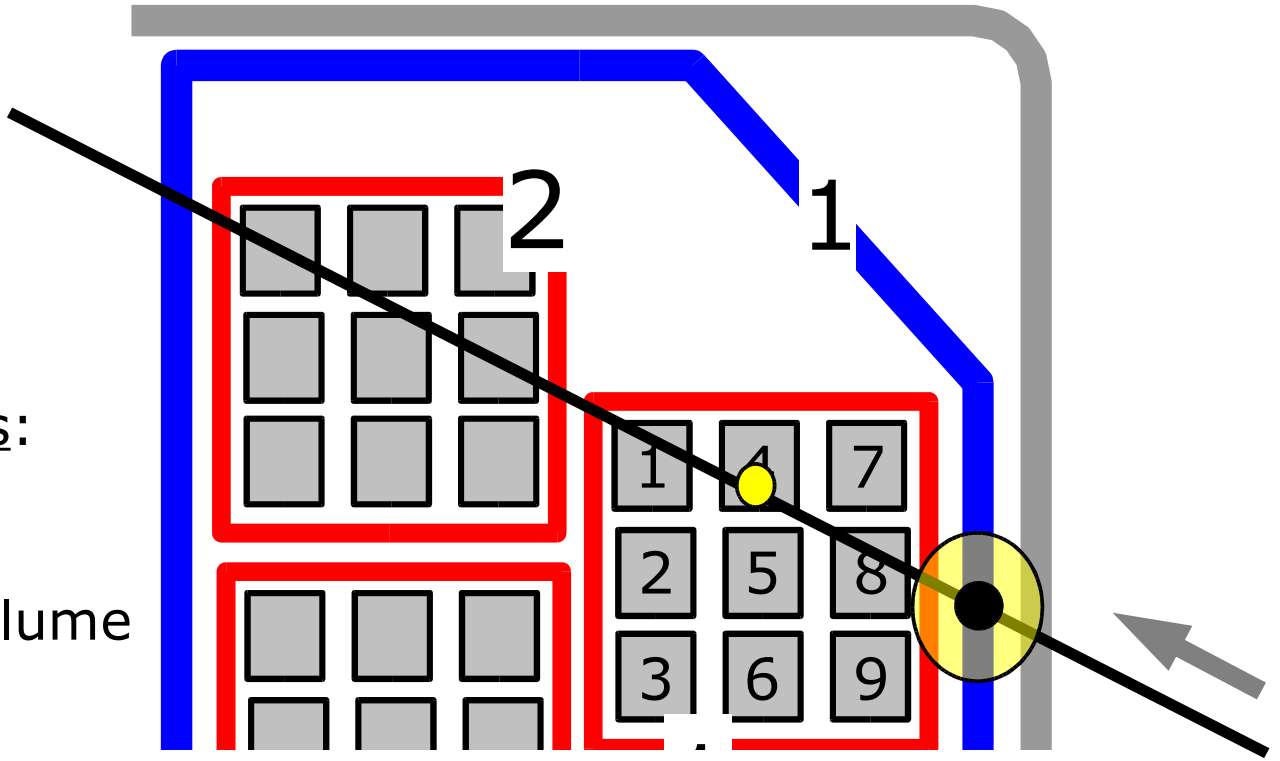


Each G4StepPoint provides access to
a G4TouchableHistory:
a stack of G4VPhysicalVolumes
corresponding to the
geometrical context of the StepPoint.

Touchable-History

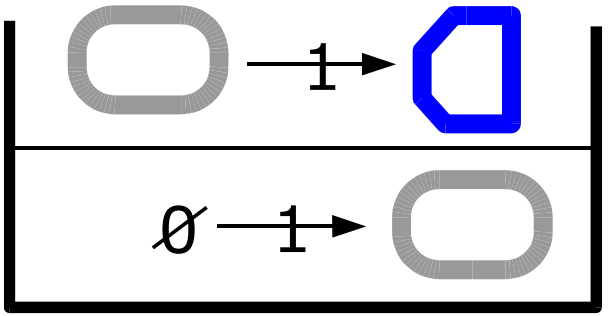


world volume has zero-ptr as "mother"

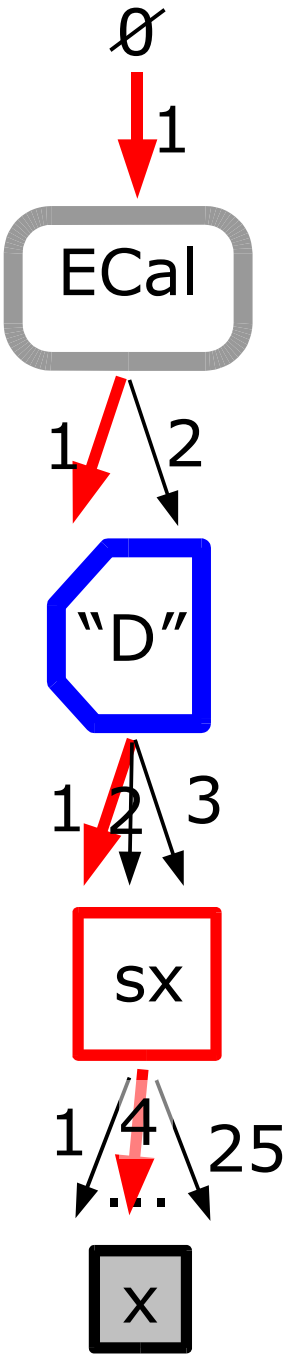


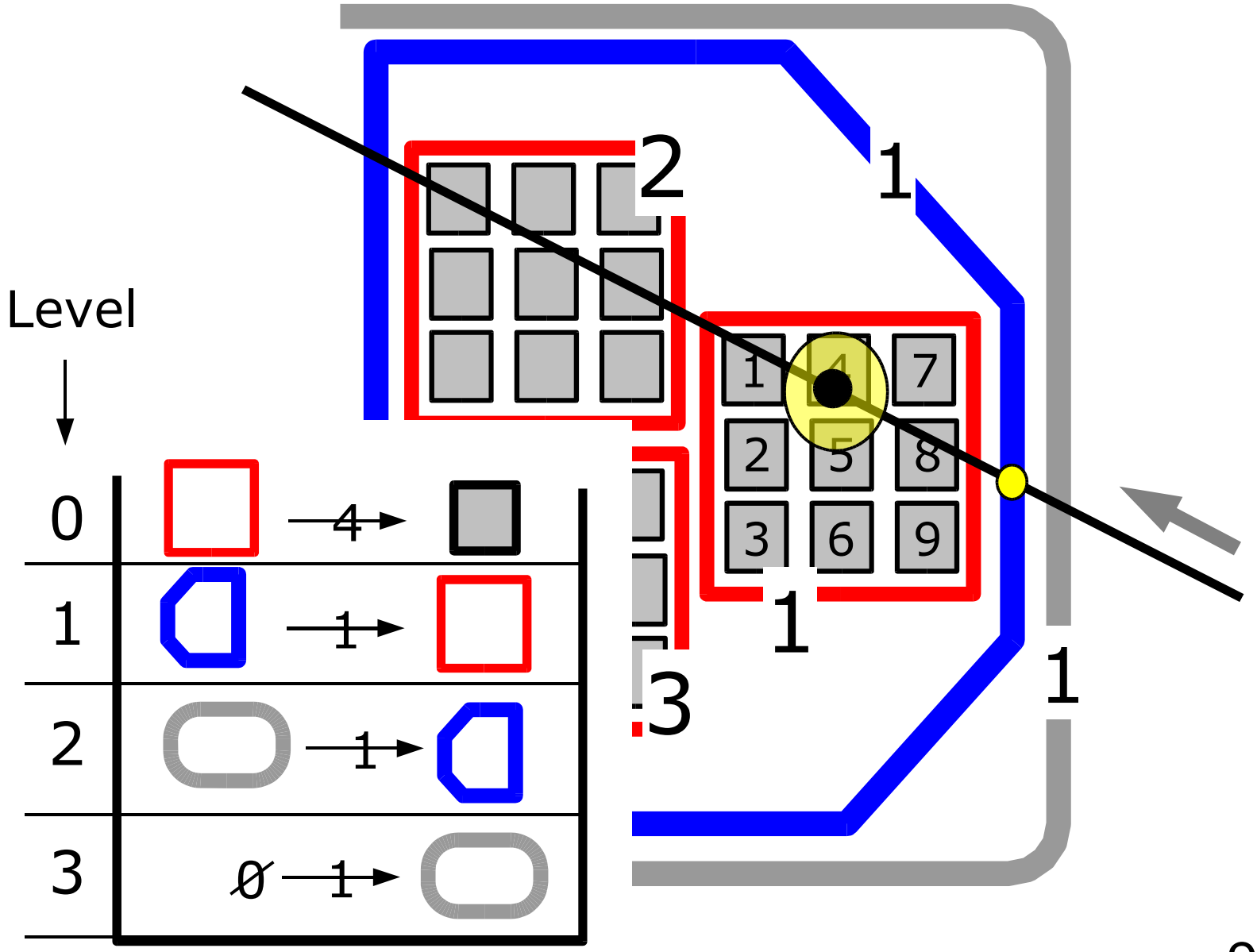
History means: geometrical history back to the root-volume

G4TouchableHistory: stack of physical volumes



Touchable-History



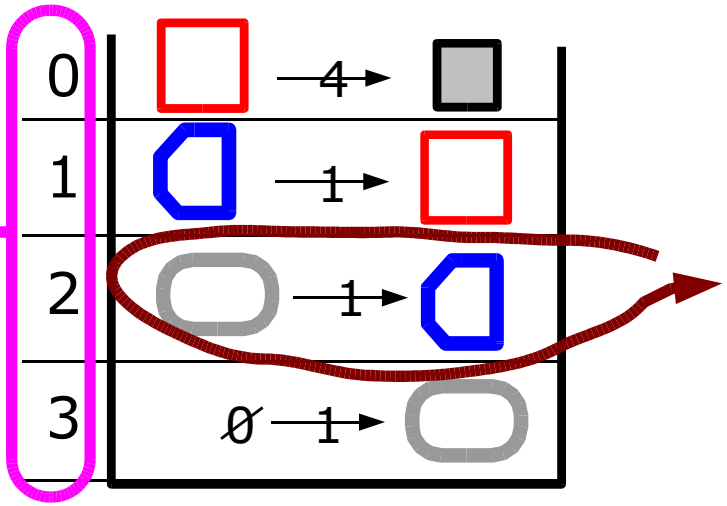


Touchable-History

Edit | Hierarchy | Class | Xref | Include | Retriever | Grep

Name	Class	Type	Parameters
G4VTouchable(md)	G4VTouchable	int	()
~G4VTouchable(md)	G4VTouchable	virtual int	()
GetHistory(md)	G4VTouchable	virtual const G4NavigationHistory *	()
GetHistoryDepth(md)	G4VTouchable	virtual G4int	()
GetReplicaNumber(md)	G4VTouchable	virtual G4int	(G4int)
GetRotation(md)	G4VTouchable	virtual const G4RotationMatrix *	(G4int)
GetSolid(md)	G4VTouchable	virtual G4VSolid *	(G4int)
GetTranslation(md)	G4VTouchable	virtual const G4ThreeVector &	(G4int)
GetVolume(md)	G4VTouchable	virtual G4VPhysicalVolume *	(G4int)

class G4TouchableHistory : public G4VTouchable



instance of G4VPhysicalVolume

Step - StepPoint - Touchable

G4Step * step = ...;

G4StepPoint point = step->GetPostStepPoint()

G4ThreeVector pos = point.GetPosition(); global position

G4VTouchable * touch = point.GetTouchable();

G4VPhysicalVolume * vol = touch->GetPhysicalVolume(3);
....

Touchable can also transform the global position into the relative position w.r.t. the reference frame of the solid in any level of the TouchableHistory ...

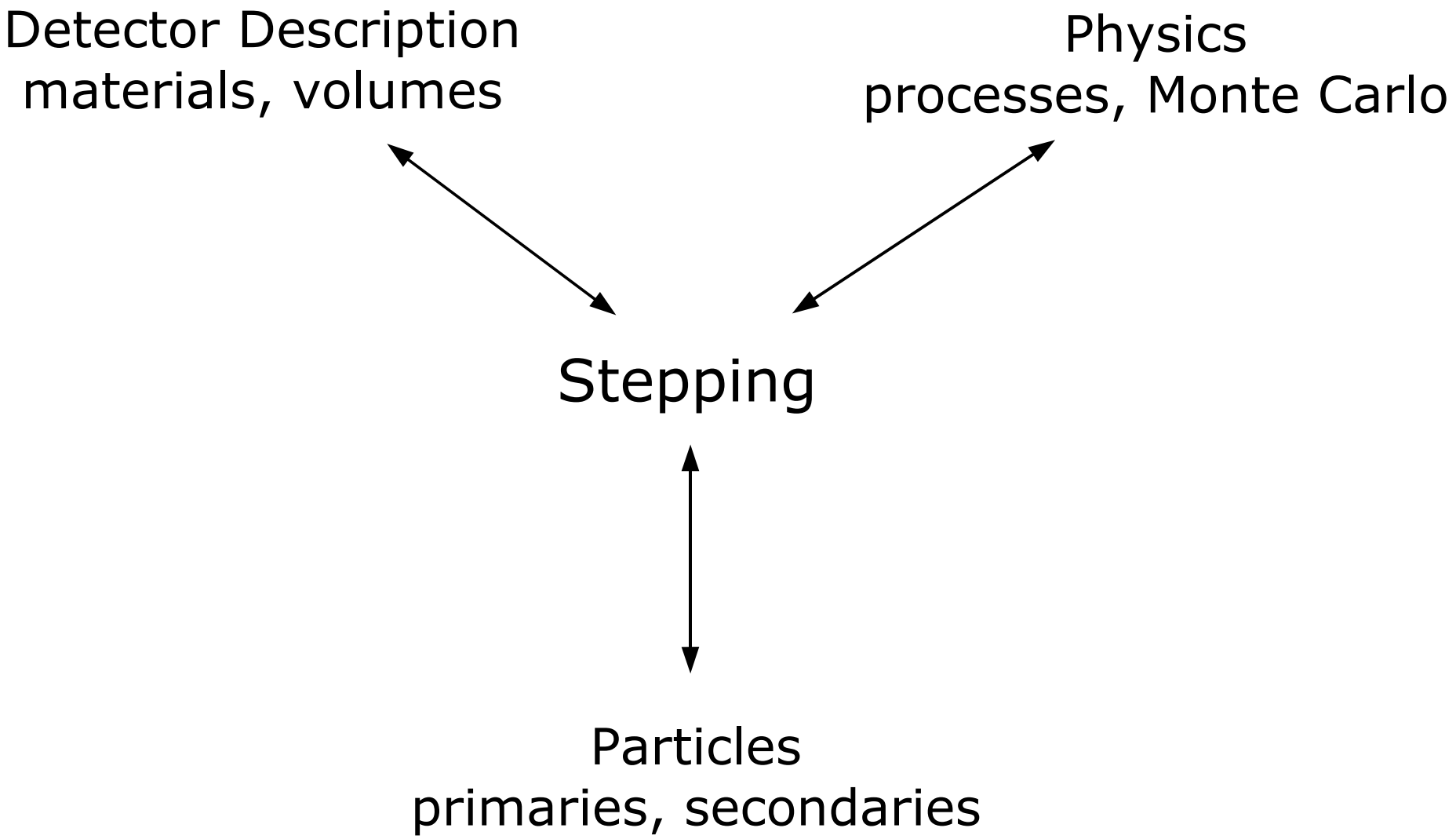
Uff!! ... But hang on!

- A – Physics Model in GEANT4
  - Stepping: moving in free path lengths
  - Physics Processes
- B – Detector Description in GEANT4
  - Solids/Shape Model
  - Volumes
  - Hierarchy of Volumes
- Combining A + B
  - Stepping through a detector description

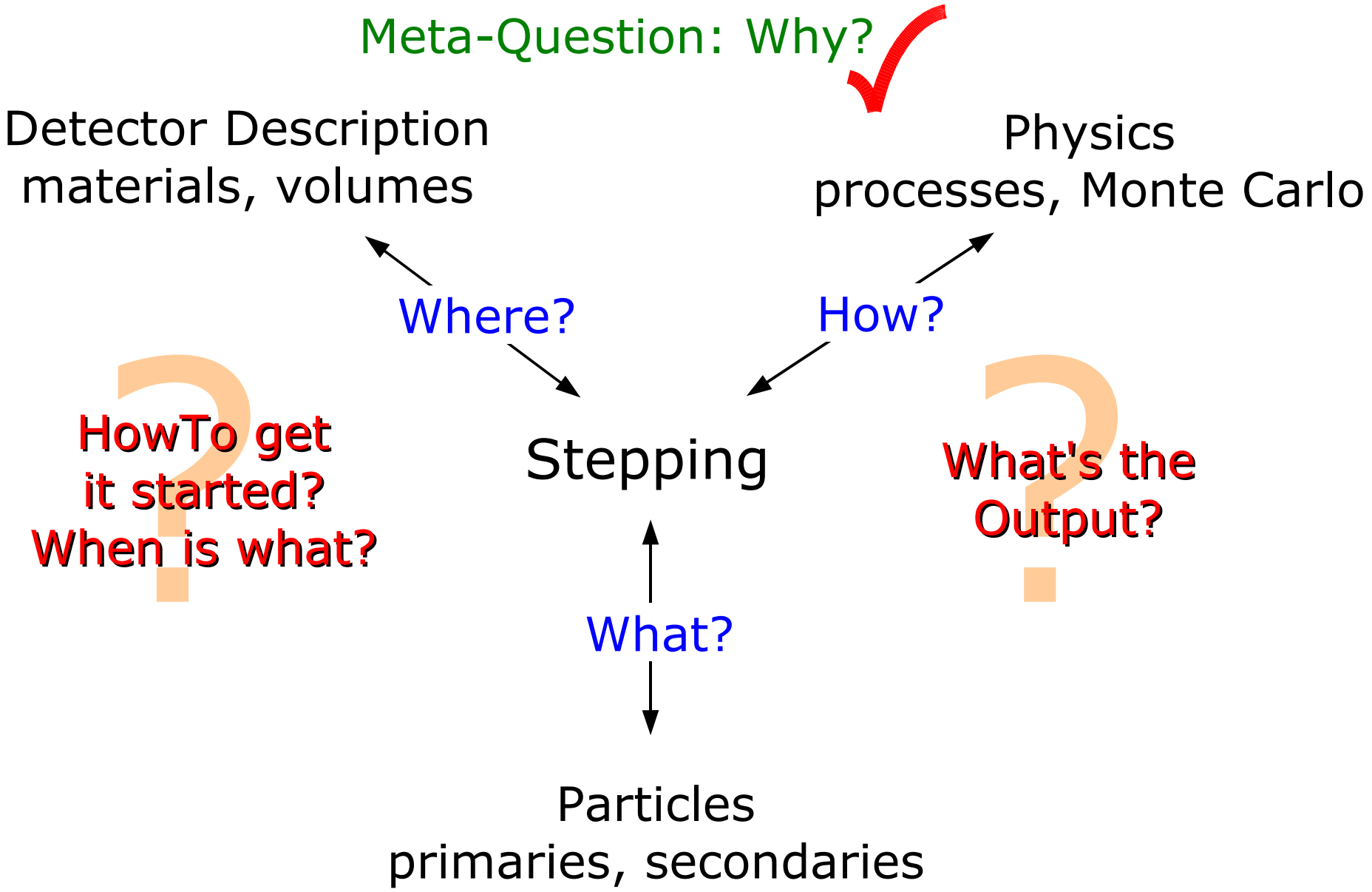
Final Part - Overview

- Putting it all together!
  - Structure of a GEANT4 simulation
  - User Initialization Hooks
  - User Action Hooks
- Extracting Information
  - Sensitive Detectors & Hits
- Boosting the Next Level
  - handover to Tony and Mark
- Summary
- Exercises

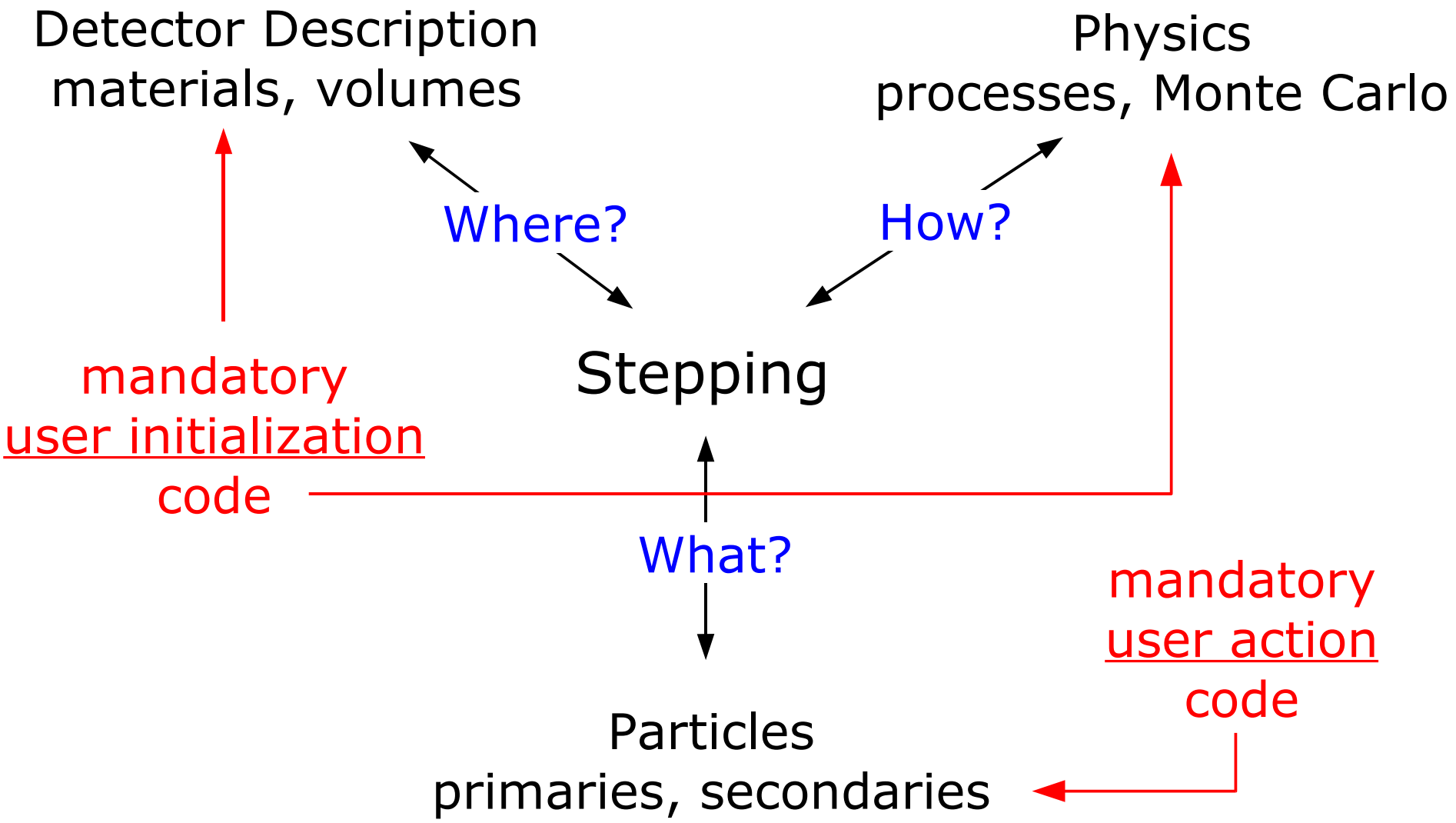
The Missing Link(s)



The Missing Link(s)



The Missing Link(s)



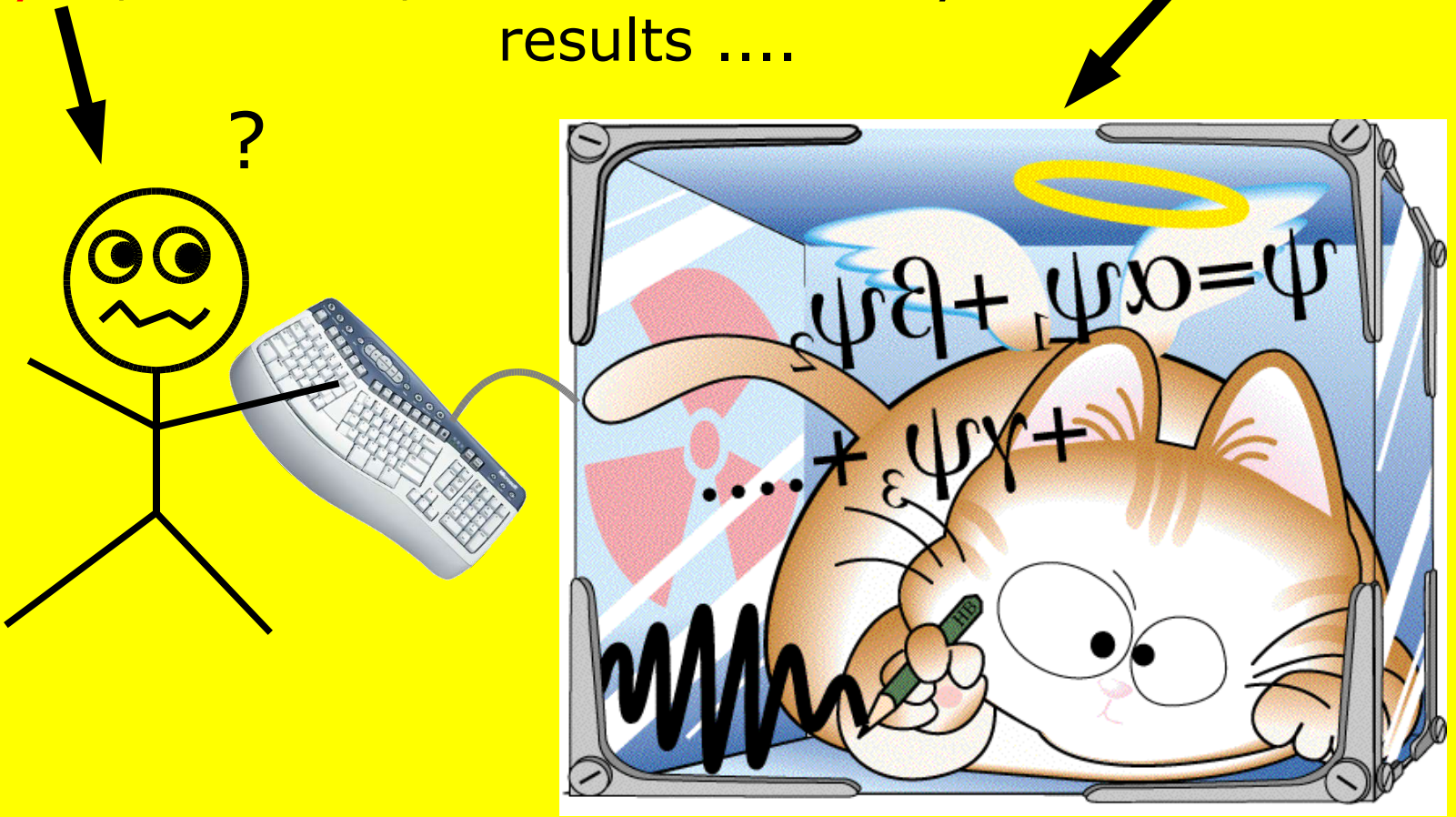
Mandatory User Code

- class G4VUserDetectorConstruction
  - virtual G4VPhysicalVolume * Construct() = 0;
  - overload to implement the detector description
- class G4VUserPhysicsList
  - virtual void ConstructParticles() = 0;
    virtual void ConstructProcesses() = 0;
  - overload for particles & physics processes selection
- class G4VUserPrimaryGenerator
  - virtual void GeneratePrimaries(class G4Event *) = 0;
  - overload to generate primary particles

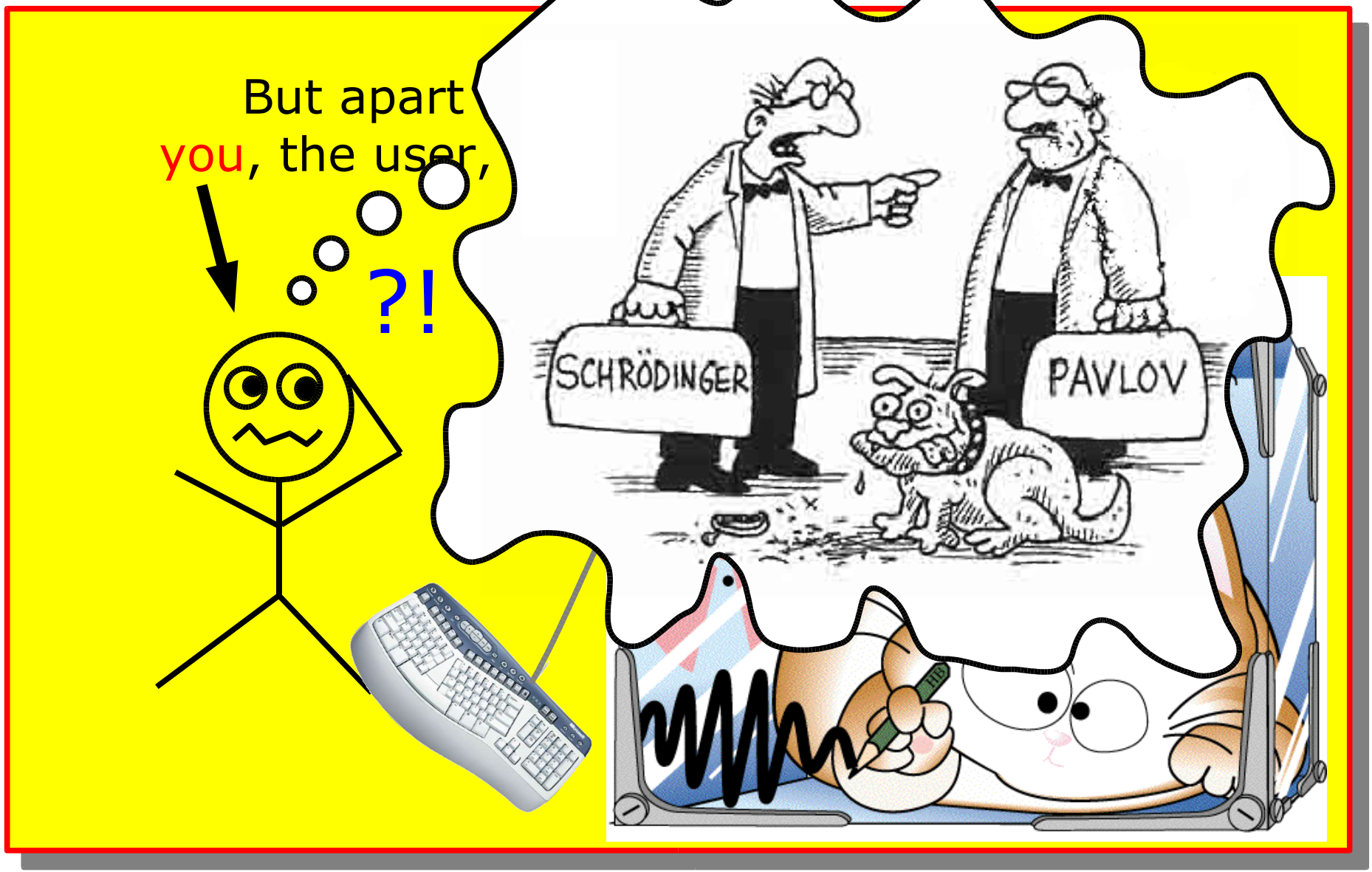
Once implementations of these three base classes are provided, a Geant4 based simulation can run!

Mandatory User Code

But apart from CPU processing time,
you, the user, don't observe any simulation
results ....

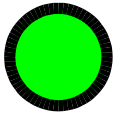


Mandatory User Code



Initialization & Action Hooks

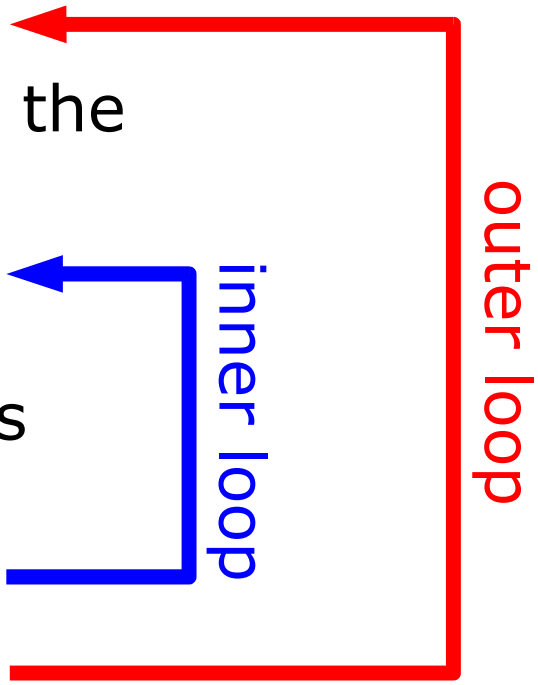
- Geant4 foresees user hooks to instrument the simulation:
  - User-Hooks: sub-classes of G4VUserXXX or G4UserYYY
  - polymorphic methods are called by the G4-kernel
- Mandatory User Actions
  - Detector description (G4VUserDetectorDescription)
  - Physics List – which particles, which processes (G4VUserPhysicsList)
  - Primary event generation (G4VUserPrimaryGenerator)
- Optional User Actions
  - Run-Action: beginning and end of a Run
  - Event-Action: beginning and end of an Event
  - Track: beginning and end of the track of one particle
  - Step: after each G4Step
  - Hooks for hit and digitization processing



Task Breakdown

User-Action & Initialization-Code invoked during various stages during a simulation run

- Initialization
  - detector description
  - physics processes selection and configuration
- Run
  - contains several Events under the same simulation conditions
- Event
  - generation of primary particles
  - tracking of all particles



Task Breakdown

- In
  - –
  - –
- Ru
  - –
- Ev
  - –
  - –

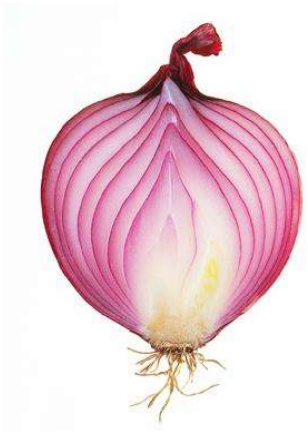
Within these two main nested loops,
Run & Event,
Geant4 offers several user-hooks (call-backs)
that can be implemented by the
simulation user to extract simulation information
during tracking or to influence / steer
the simulation:

Run-, Event-, Track-, Step-Actions,
Primary Particle - Action

(not treated in these lectures:
Stack-Action)

G4 Russian Onion

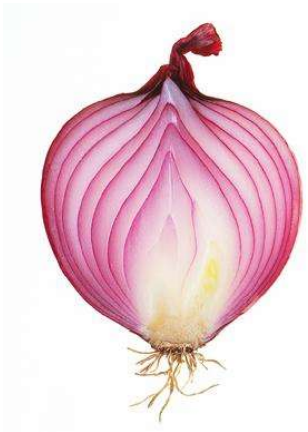
Simulation
1
n
Run
1
e
Event
1
t
Track
1
s
Step



run loop
event loop
tracking
stepping

Subclassing Actions

Simulation
1
n
Run
1
e
Event
1
t
Track
1
s
Step



G4VUserDetectorConstruction

G4VUserPhysicsList

class MyDetector
: public G4VUserDetectorConstruction

class MyPhysics
: public G4VUserPhysicsList

before a Run starts:
- process user-parameters
  (not fully treated in these lectures)
- create materials, geometry, physics processes

run loop

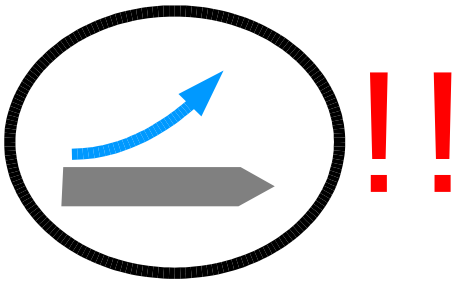
Physics Initialization

Remember?

- bool G4VProcess::IsApplicable(const G4ParticleDefinition &)

- Static instances of subclasses of G4ParticleDefinition describe the particle types. The user has to instantiate these type, that should be used in the simulation!

All this is done in the PhysicsList initialization hook!



Physics Initialization

Each particle type has a process manager!
Processes are simply registered to the appropriate process manager:

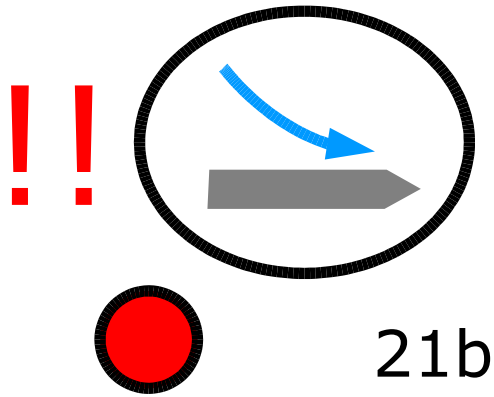
```
// Get the process manager for gamma
G4ParticleDefinition* particle = G4Gamma::GammaDefinition();
G4ProcessManager* pmanager = particle->GetProcessManager();

// Construct processes for gamma
G4PhotoElectricEffect * thePhotoElectricEffect = new G4PhotoElectricEffect();
G4ComptonScattering * theComptonScattering = new G4ComptonScattering();
G4GammaConversion* theGammaConversion = new G4GammaConversion();

// Register processes to gamma's process manager
pmanager->AddDiscreteProcess(thePhotoElectricEffect);
pmanager->AddDiscreteProcess(theComptonScattering);
pmanager->AddDiscreteProcess(theGammaConversion);
```

Physics Initialization

There are many details not shown before, like:

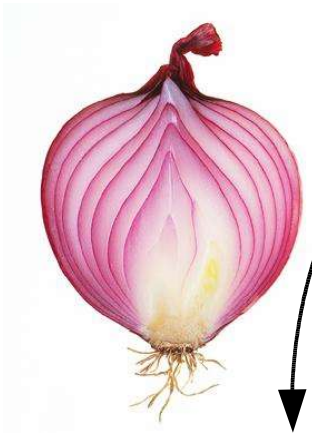
- which particles to include for a “proper” simulation?
- which processes to include for a “proper” simulation?
- specifying the order of process invocation within the process manager of one particle type
- setting cut values, i.e. values tuning the range of validity of a process

Geant4 comes with a set of pre-configured, general purpose physics lists!



Subclassing Actions

Simulation
1
n
Run
1
e
Event
1
t
Track
1
s
Step



G4Run
RecordEvent(G4Event*)
int GetRunID()
.. administration of
hit collection tables ..

class MyRun
: public G4Run

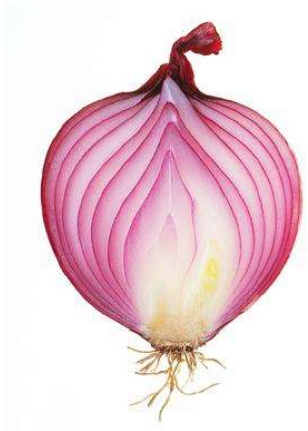
G4UserRunAction
BeginOfRunAction(G4Run*)
EndOfRunAction(G4Run*)

class MyRunAction
: public G4UserRunAction

run loop

Subclassing Actions

Simulation
1
n
Run
1
e
Event
1
t
Track
1
s
Step



G4Event

int GetEventID()
.. access to hit-collections ..
.. access to digi-collections ..

G4UserEventAction

BeginOfEventAction(G4Event*)
EndOfEventAction(G4Event*)

class MyEventAction
: public G4UserEventAction

run loop event loop

Subclassing

Simulation
1
n
Run
1
e
Event
1
t
Track
1
s
Step

G4VUserPrimary-GeneratorAction

GeneratePrimaries (G4Event *)

class MyGenerator : public G4VUserPrimary....

G4Event

int GetEventID()
.. access to hit-collections ..
.. access to digi-collections ..

G4UserEventAction

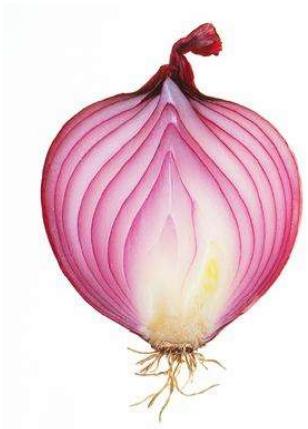
BeginOfEventAction(G4Event*)
EndOfEventAction(G4Event*)

class MyEventAction
: public G4UserEventAction

run loop event loop

Subclassing Actions

Simulation
1
n
Run
1
e
Event
1
t
Track
1
s
Step



G4Track

int GetTrackID()
.. access track vertex ..
.. access current kinematics ..

G4UserTrackingAction

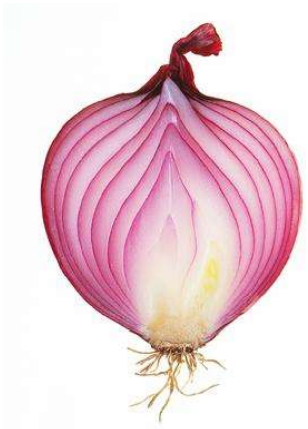
PreUserTrackingAction(G4Track*)
PostUserTrackingAction(G4Track*)

class MyTrackingAction
: public G4UserTrackingAction

run loop event loop tracking

Subclassing Actions

Simulation
1
n
Run
1
e
Event
1
t
Track
1
s
Step



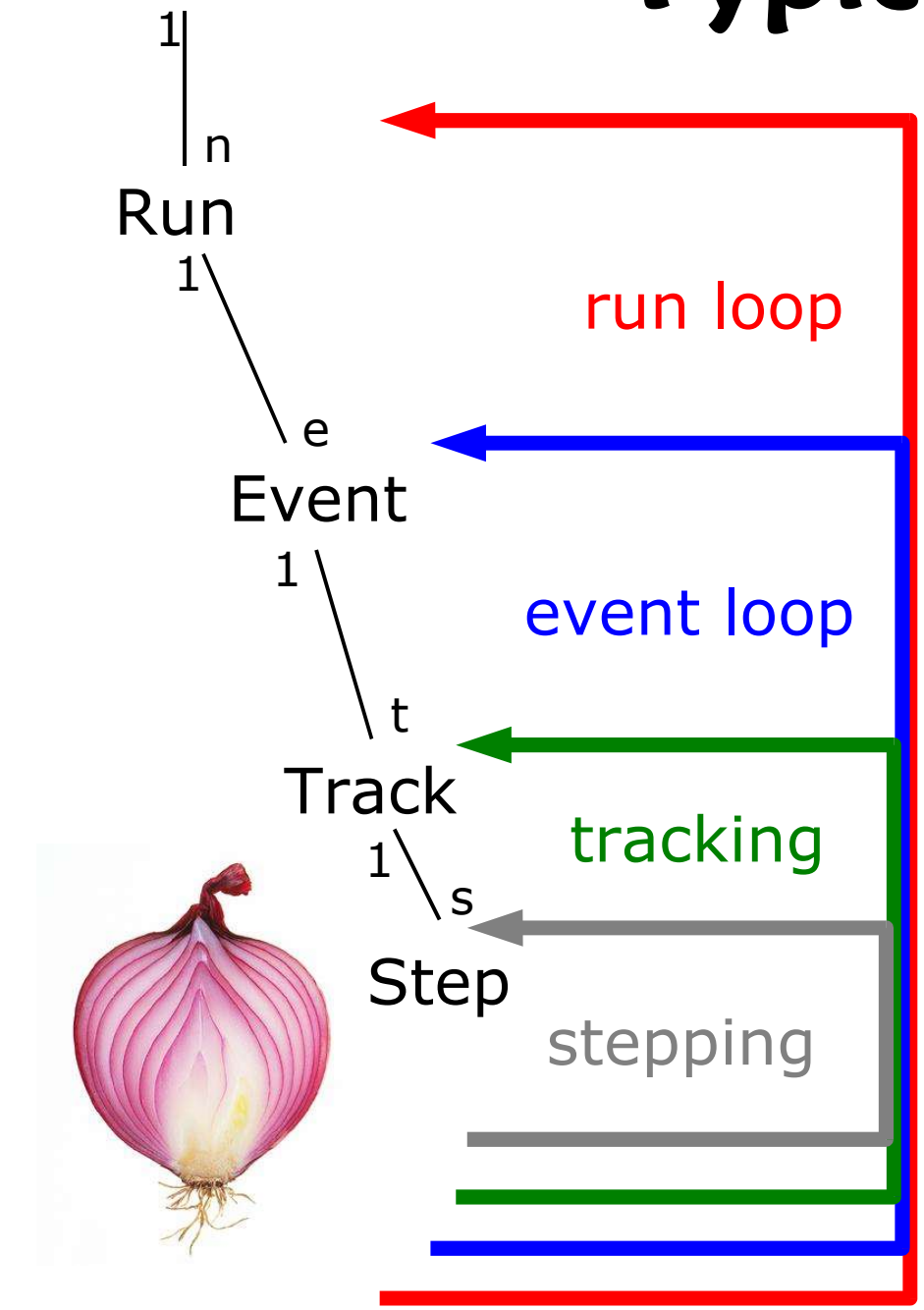
G4Step
.. pre- and post-step-points, delta information ..
.. kinematics information ..

G4UserSteppingAction
UserSteppingAction(G4Step*)

class MyStepping : public G4UserSteppingAction

run loop event loop tracking stepping

Typical Usage



Persistency related tasks;
extraction of meta-data;
pile-up; digitization

Persistency related tasks;
hit analysis;
digitization

collecting "MC truth";
influencing order of simulation

collecting "MC truth";
debugging

Putting them together

- Singleton G4RunManager is the central registry for the mandatory and optional user extensions to Geant4:
  - SetUserInitialization(..*)
    - detector description: G4VUserDetectorConstruction [m]
    - physics selection: G4VUserPhysicsList [m]
  - SetUserAction(..*)
    - primary generator: G4VUserPrimaryGeneratorAction [m]
    - run: G4VUserRunAction [o]
    - event: G4VUserEventAction [o]
    - track: G4VUserTrackingAction [o]
    - step: G4VUserSteppingAction [o]
  - BeamOn(G4int n)
    - start the simulation of n events.

[m] mandatory
[o] optional

*(G4Run- & other Managers)

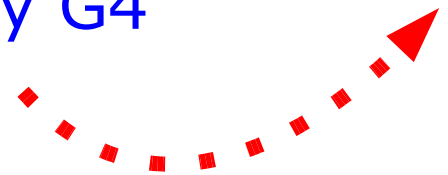
- G4RunManger registers actions to other managers
  - Event, Tracking, SteppingManagers, ..
  - many are singletons, i.e. kind of registries,
  - Managers are responsible for the steering of a specific simulation-layer
  - usually not visible to the user
- G4RunManager::beamOn(G4int n)
  - triggers a Run
  - n Events are simulated by invoking the G4EventManger
    - G4EventManager invokes G4TrackingManager
      - G4TrackingManager invokes G4SteppingManager
        - ...
- Also Managers for geometry, visualization, user-interface, ... (no time in this lectures ...)

Or: Do it yourself!

- BUT: you don't have to use G4RunManger
  - you can re-write it completely to fit your simulation requirements
  - then you have to care about how your actions are registered to Geant4
  - then you have to control the Run- & Event-Loops!
    - i.e. implement the counterpart of BeamOn(G4int n)
    - invoke your PrimaryGenerator
  - CMS does this!
- Many technical details
  - all described in the Geant4 manuals
  - but not time to show today ...

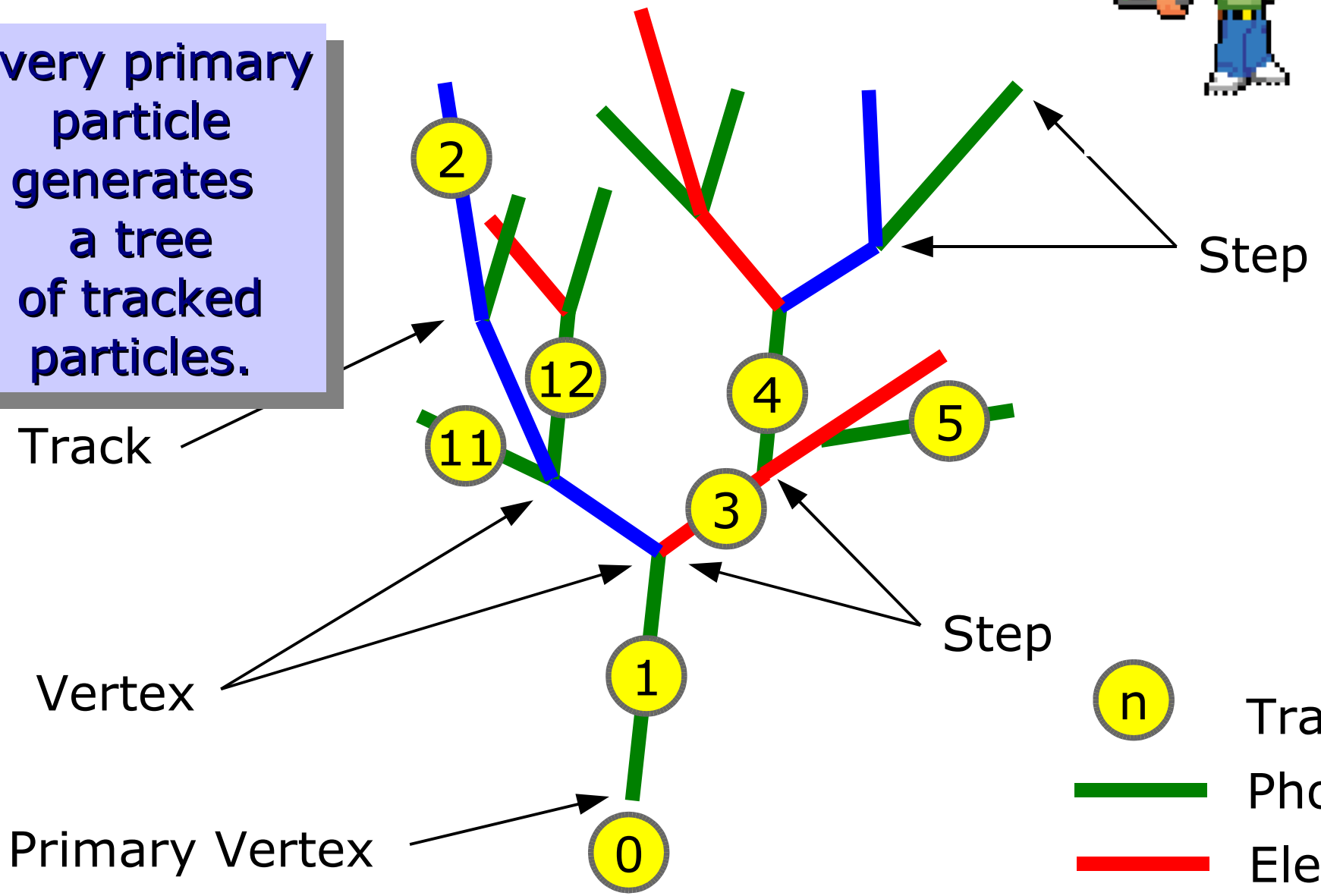
Assume, we have a RunManger,
and all UserInitializations and
-actions are registered to Geant4.

Let's follow the simulation of an
electromagnetic shower
to see when where what gets invoked by G4

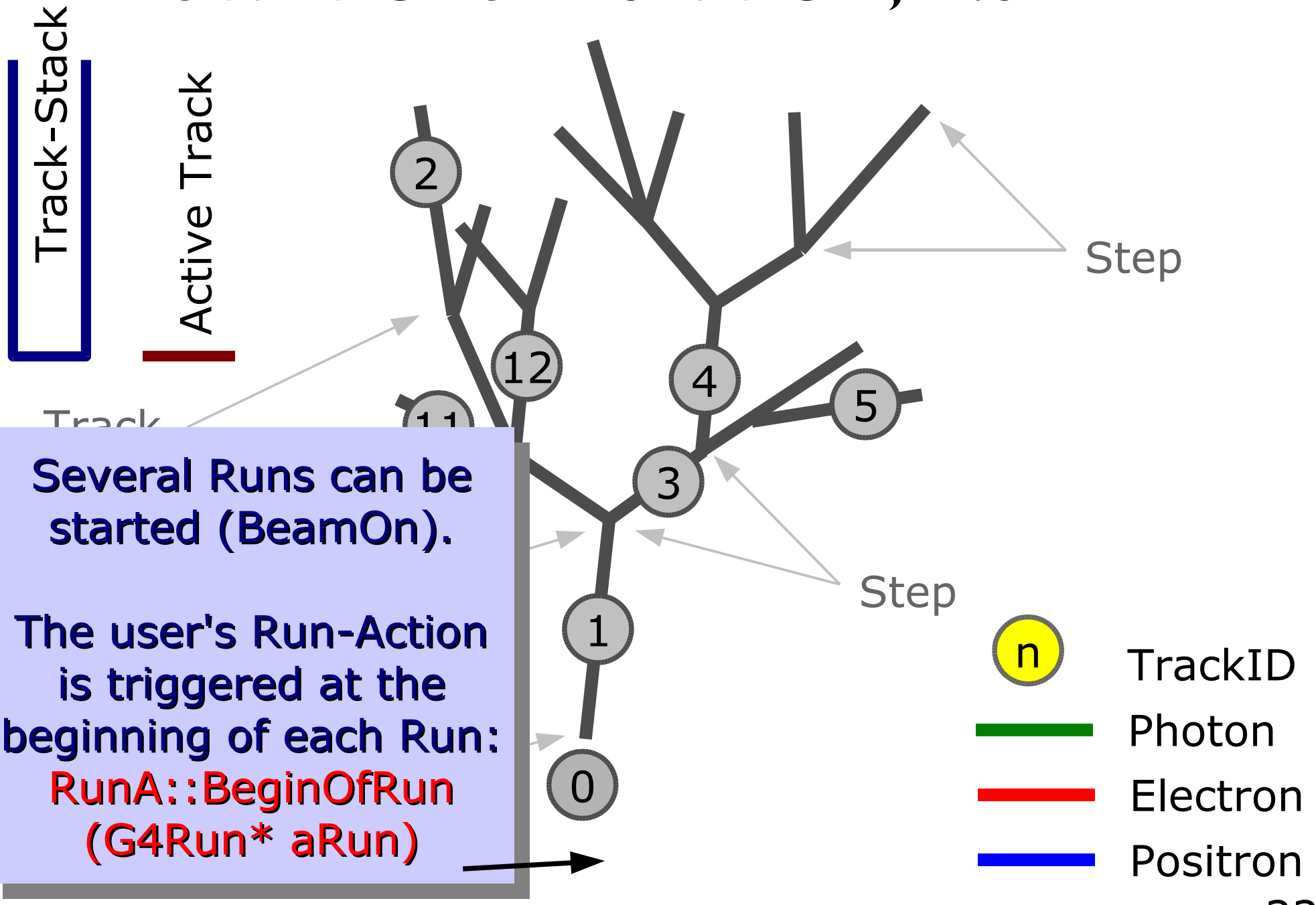


And Action!

Every primary particle generates a tree of tracked particles.



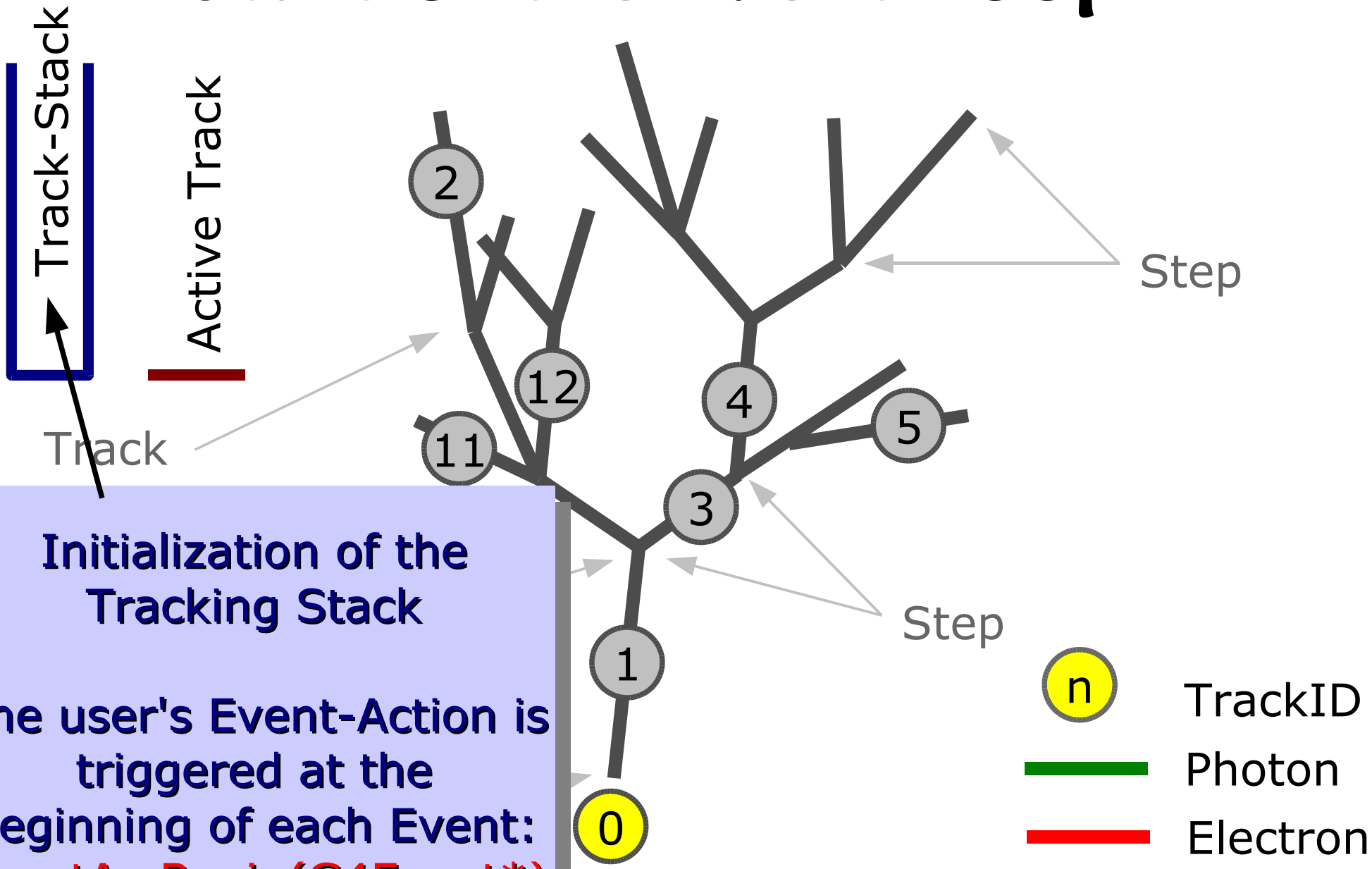
Start of Simulation, Run



Several Runs can be started (BeamOn).

The user's Run-Action is triggered at the beginning of each Run:
RunA::BeginOfRun (G4Run* aRun)

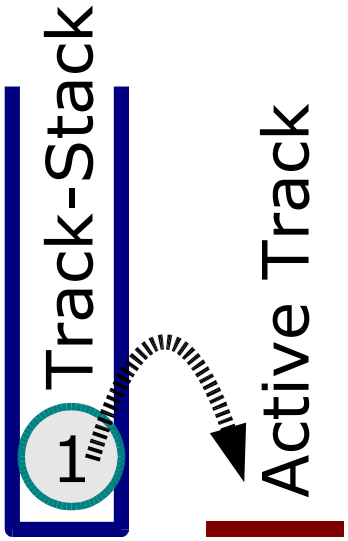
Start of the Event-Loop

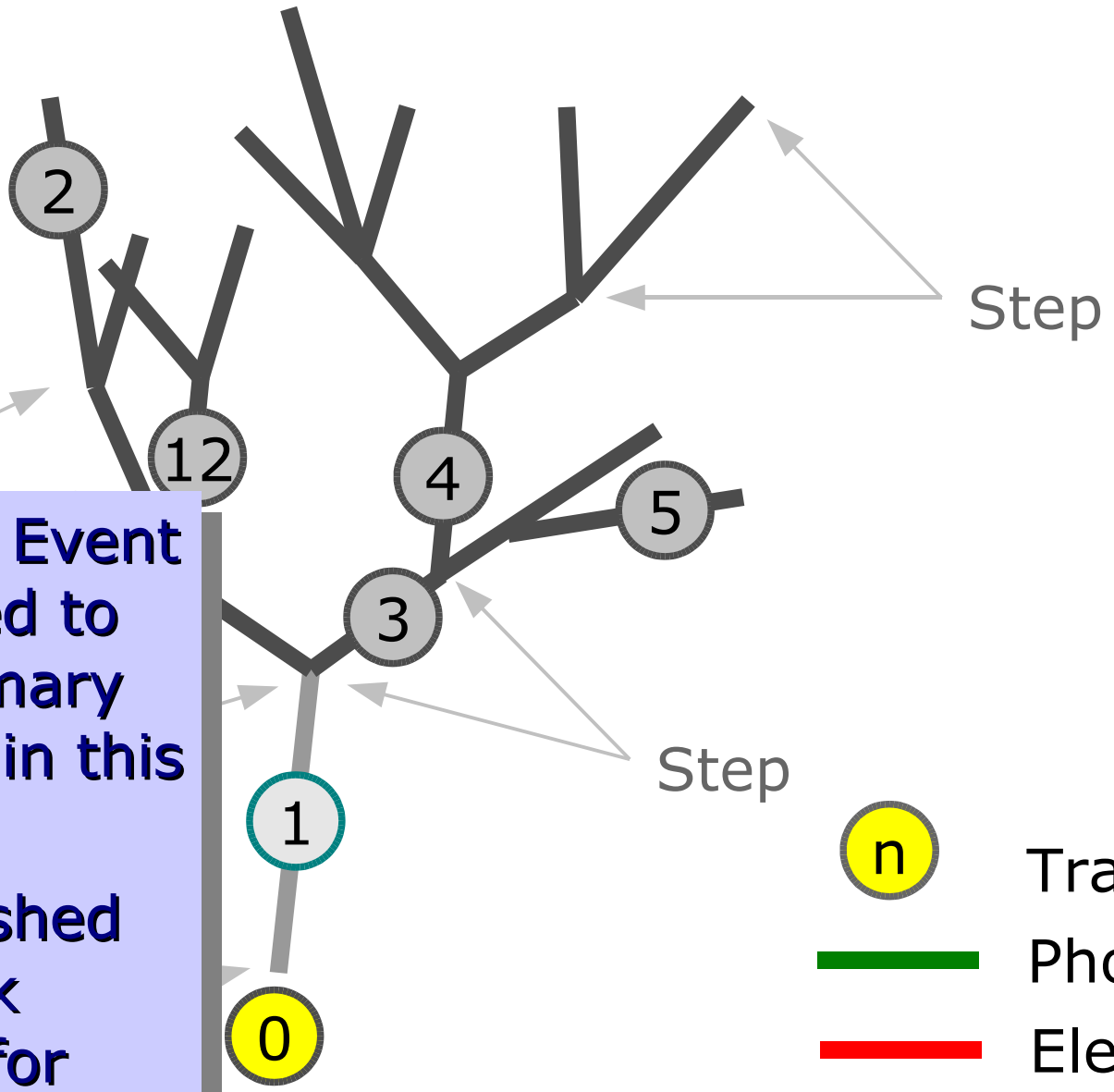


Initialization of the Tracking Stack

The user's Event-Action is triggered at the beginning of each Event:
EventA::Begin(G4Event*)

Primary Generator Action

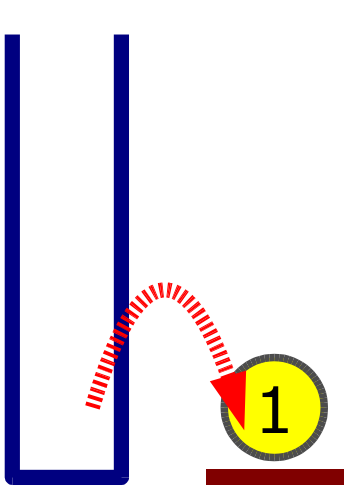




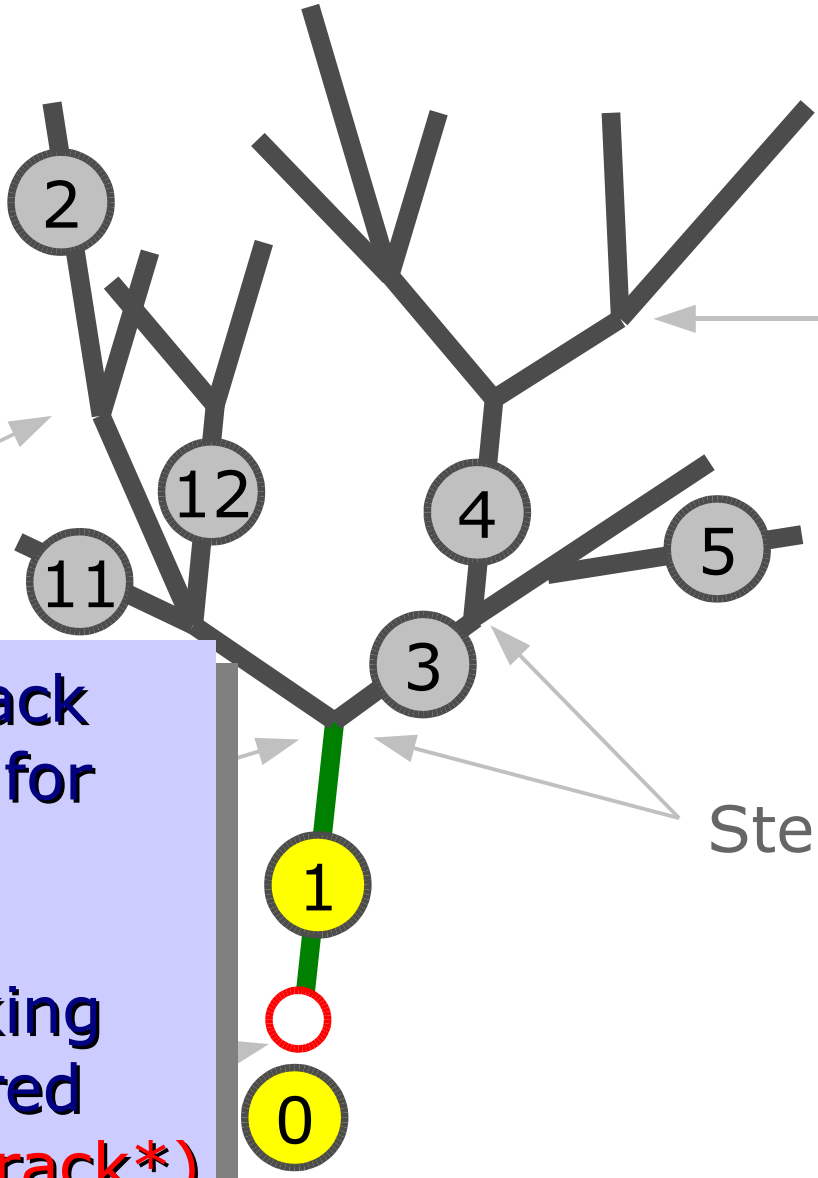
The user's Primary Event Action is triggered to generate the primary particles (only one in this example).

Primaries are pushed onto the stack (FILO) to wait for tracking

Pop Stack, Tracking (Stepping) until end of Track, generated Secondaries pushed on Stack



Track

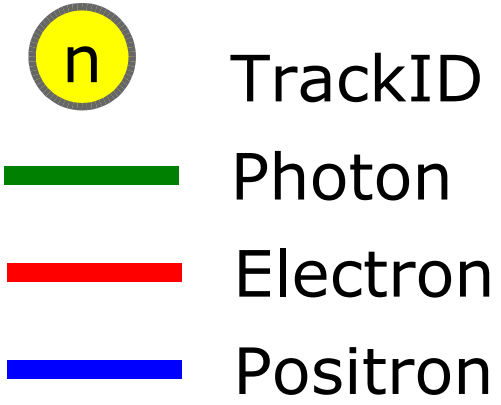


Step

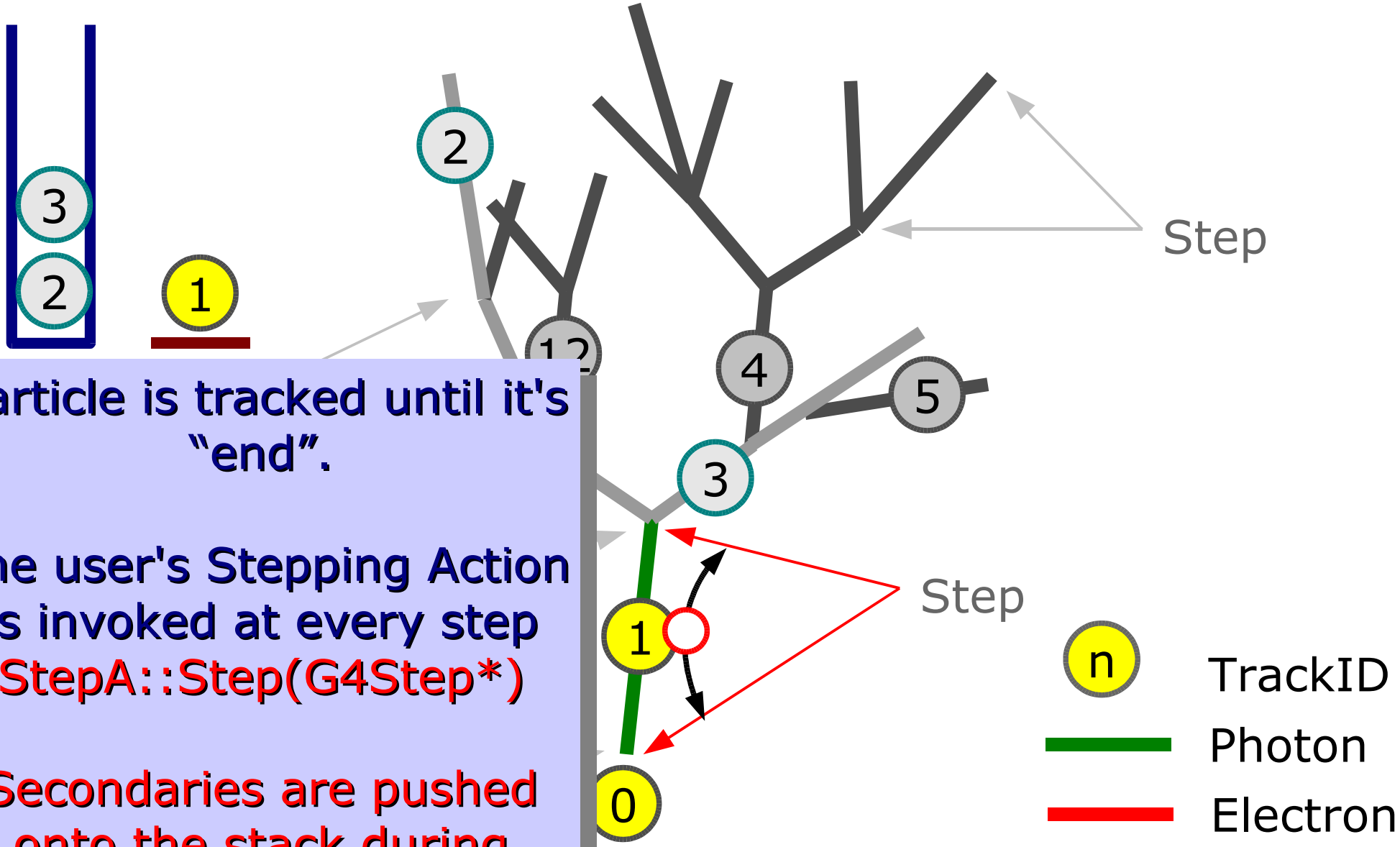
Step

pop stack; a Track becomes active for tracking.

The user's Tracking Action is triggered
TrackA::Begin(G4Track*)



Pop Stack, Tracking (Stepping) until end of Track, generated Secondaries pushed on Stack

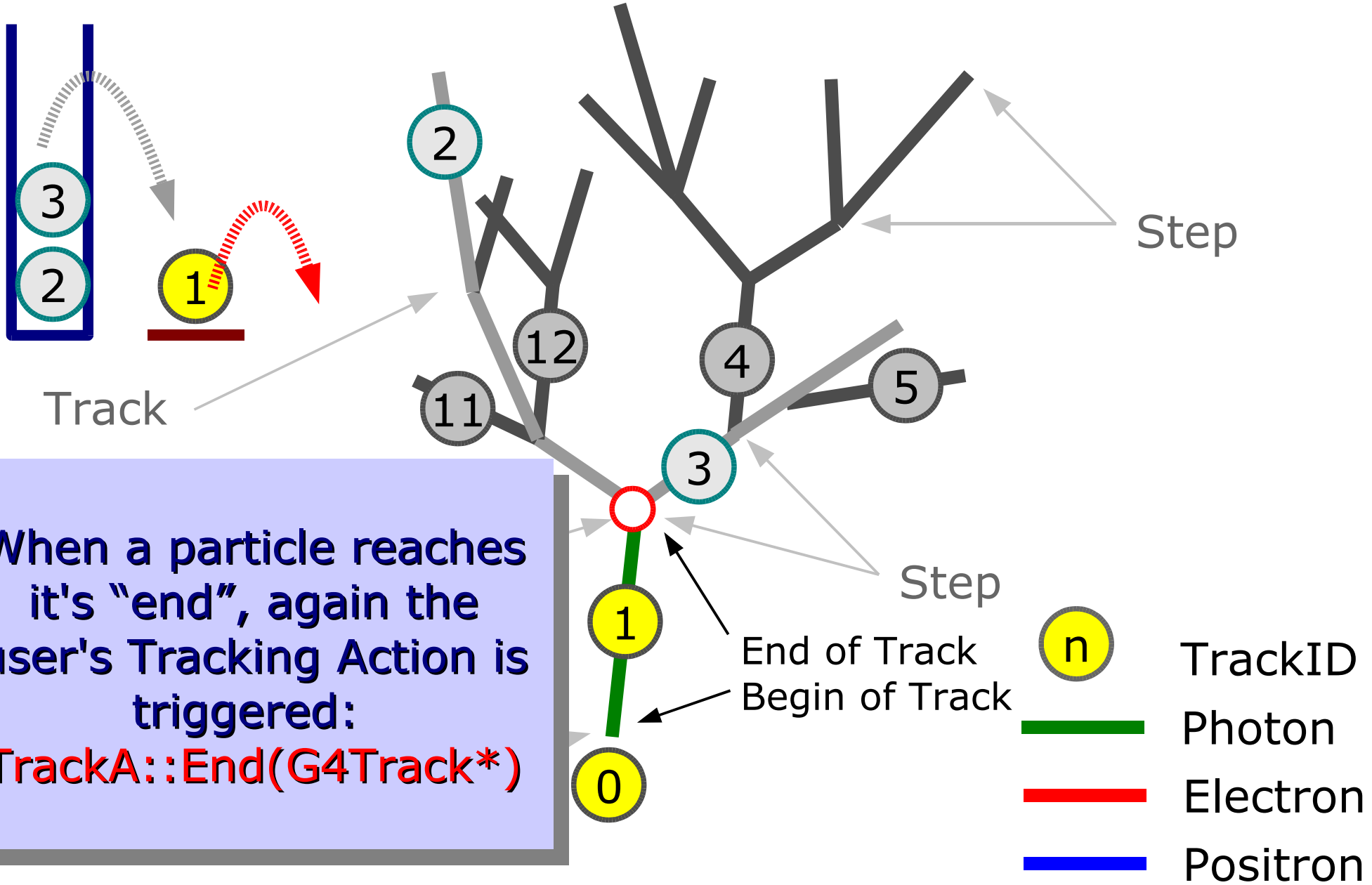


particle is tracked until it's "end".

The user's Stepping Action is invoked at every step
StepA::Step(G4Step*)

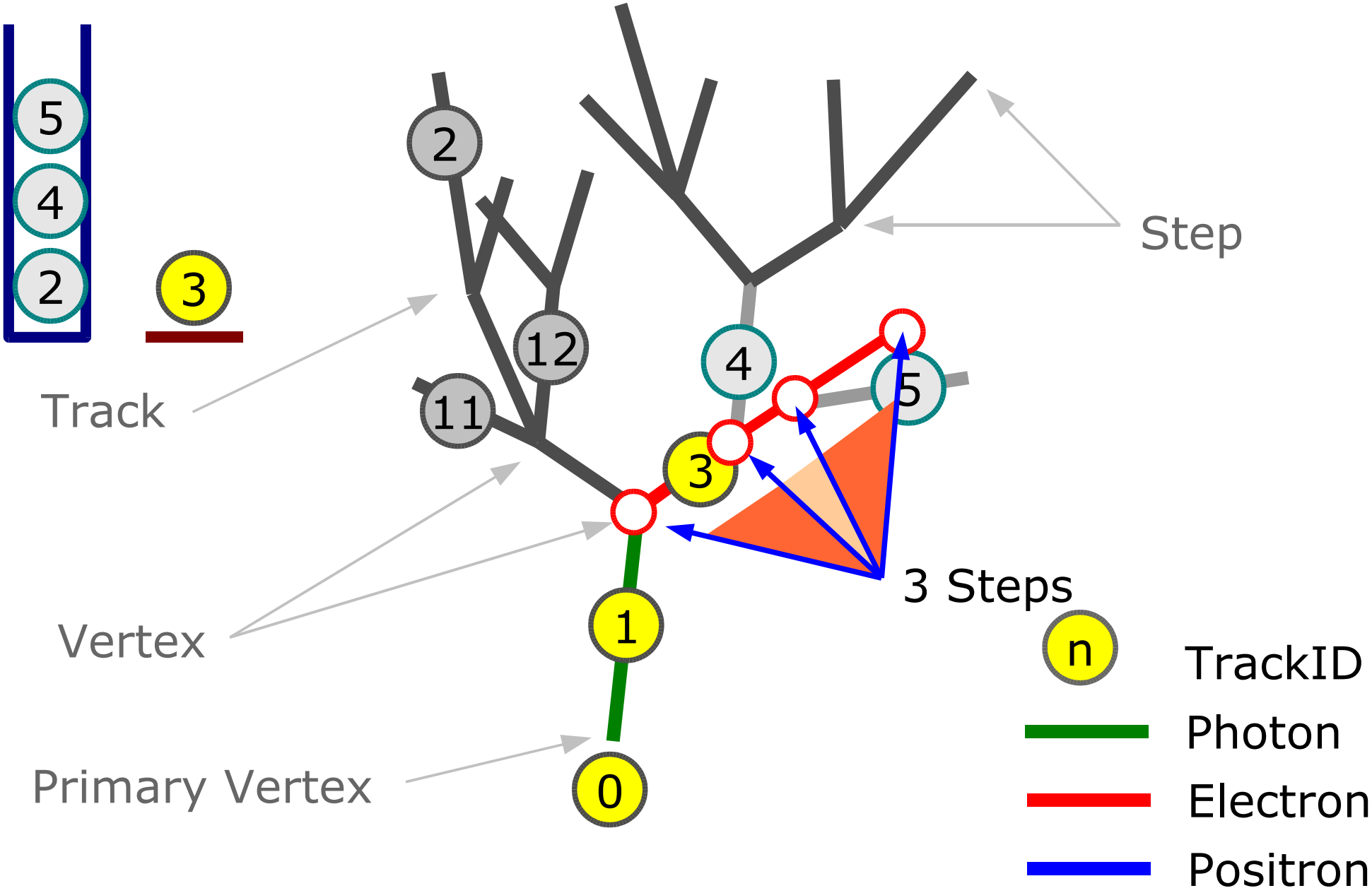
Secondaries are pushed onto the stack during stepping

Pop Stack, Tracking (Stepping) until end of Track, generated Secondaries pushed on Stack

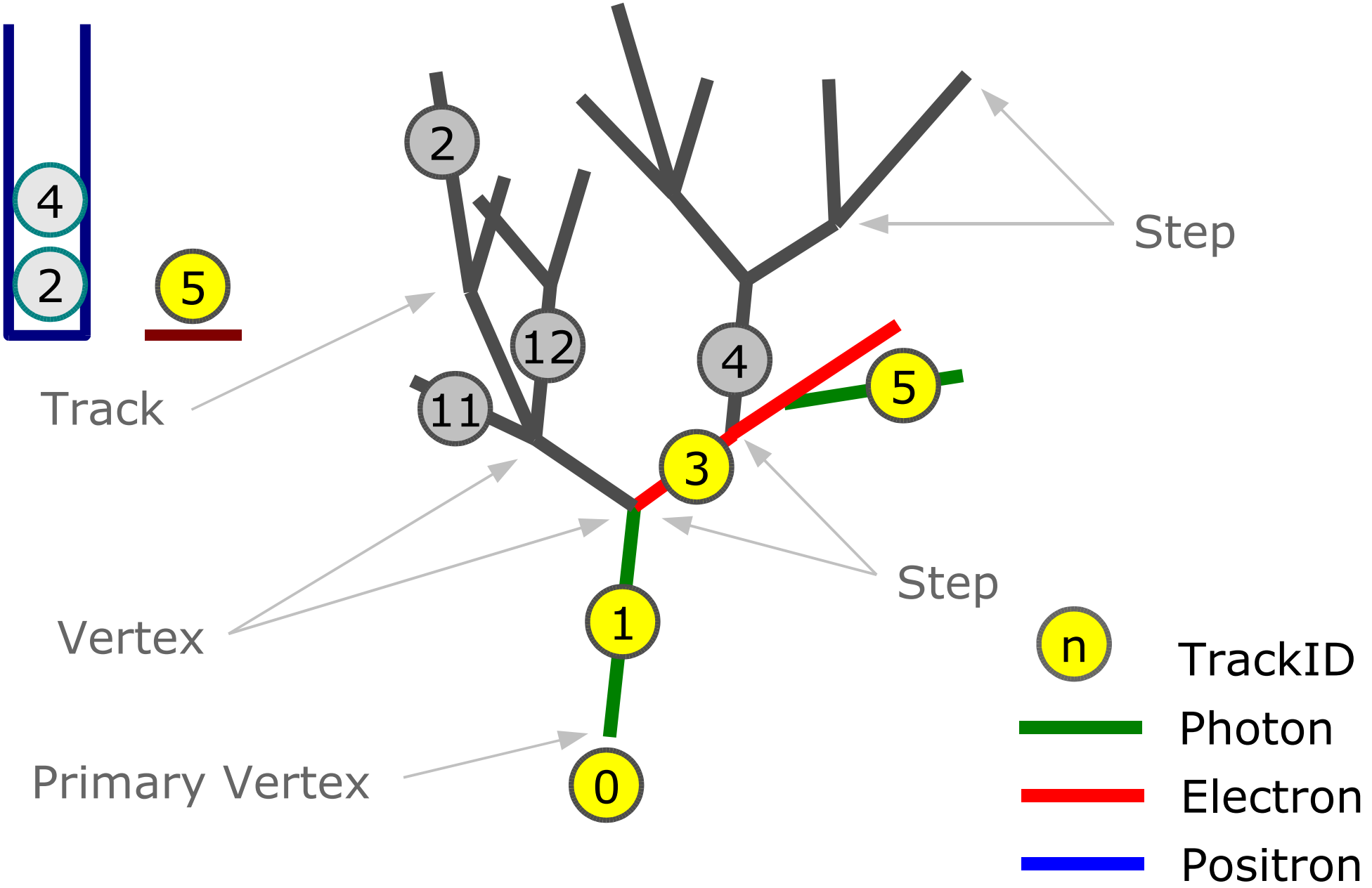


When a particle reaches it's "end", again the user's Tracking Action is triggered:
TrackA::End(G4Track*)

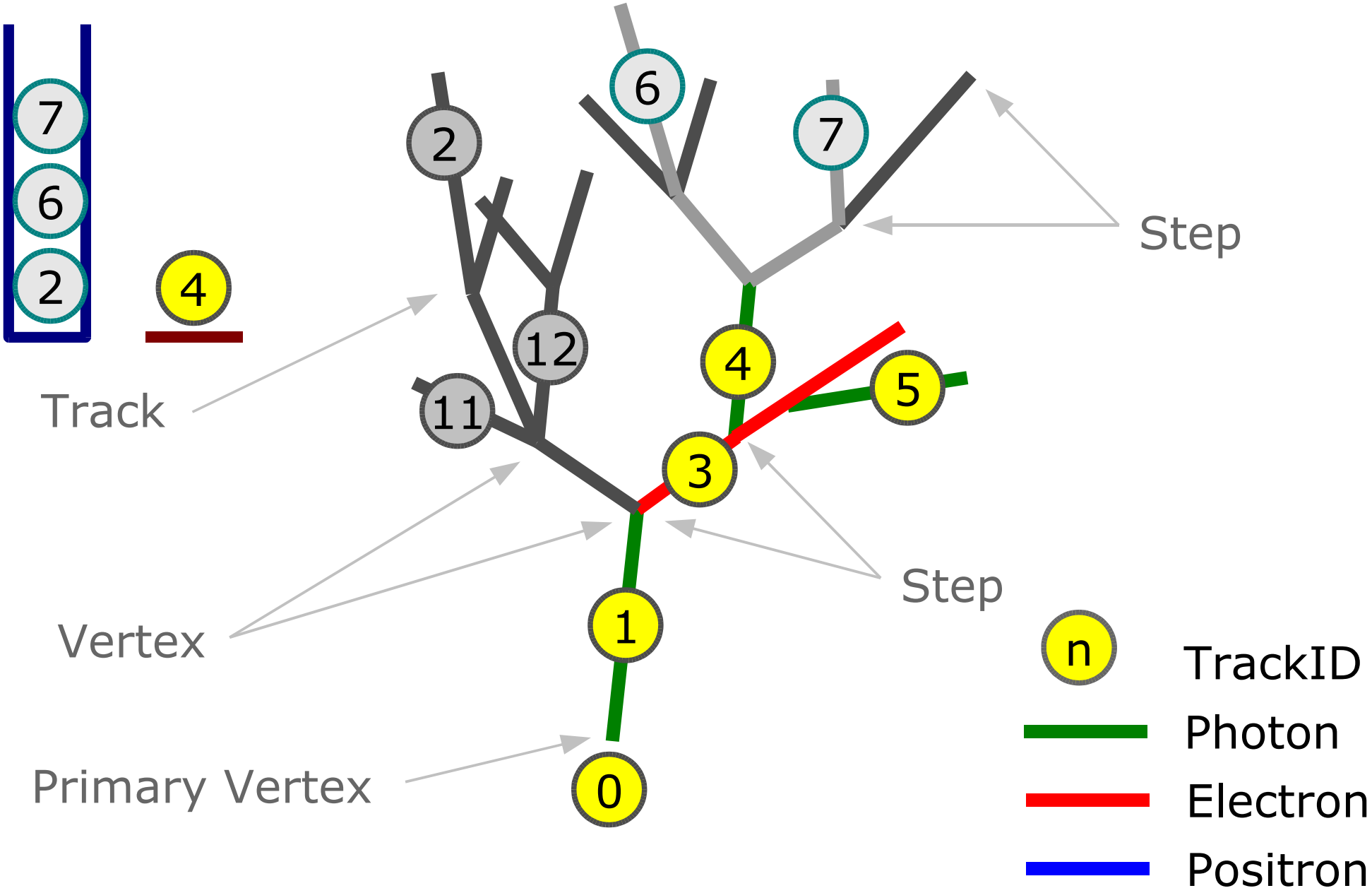
Pop Stack, Tracking (Stepping) until end of Track, generated Secondaries pushed on Stack



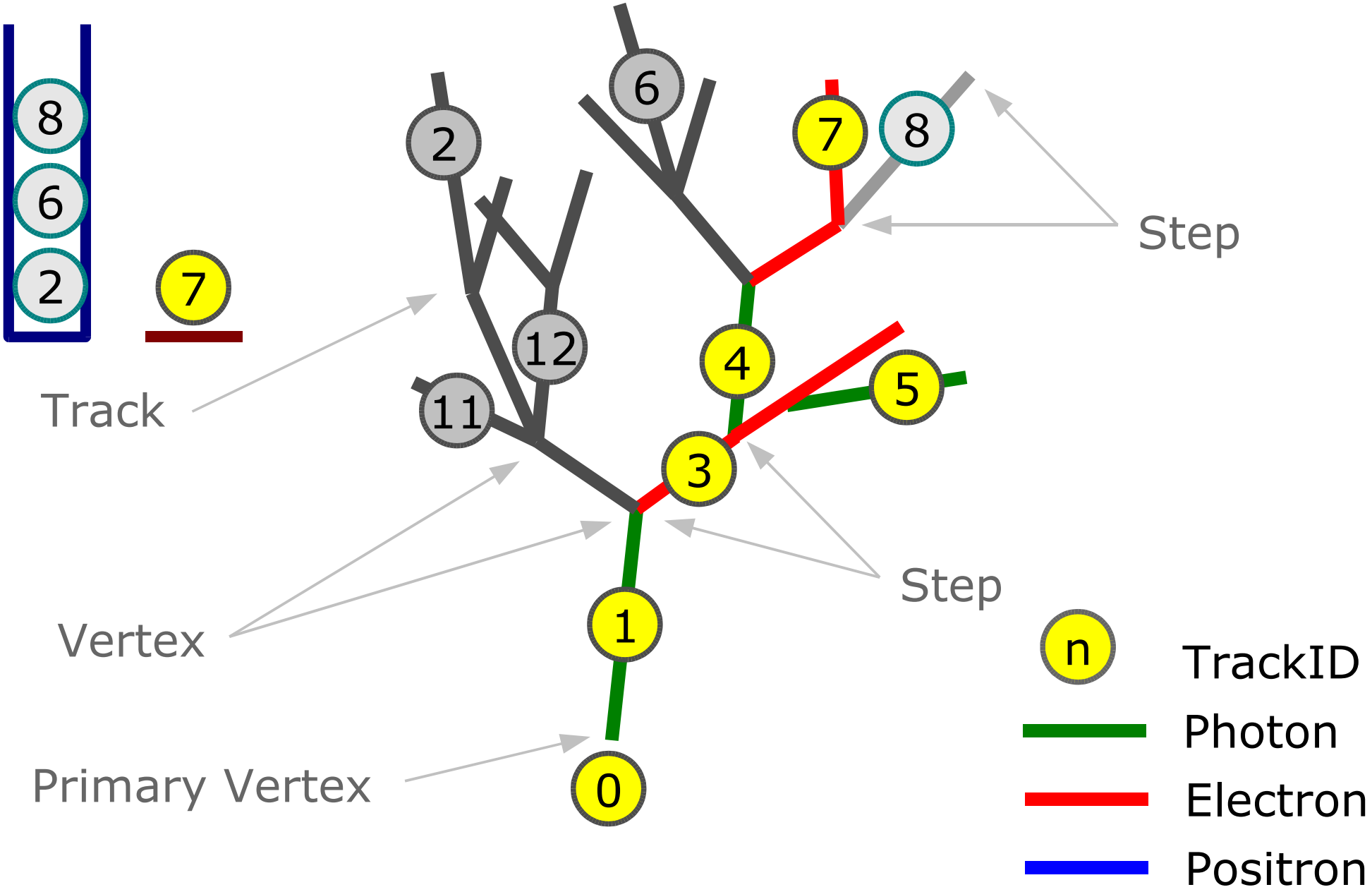
Pop Stack, Tracking (Stepping) until end of Track, generated Secondaries pushed on Stack



Pop Stack, Tracking (Stepping) until end of Track, generated Secondaries pushed on Stack

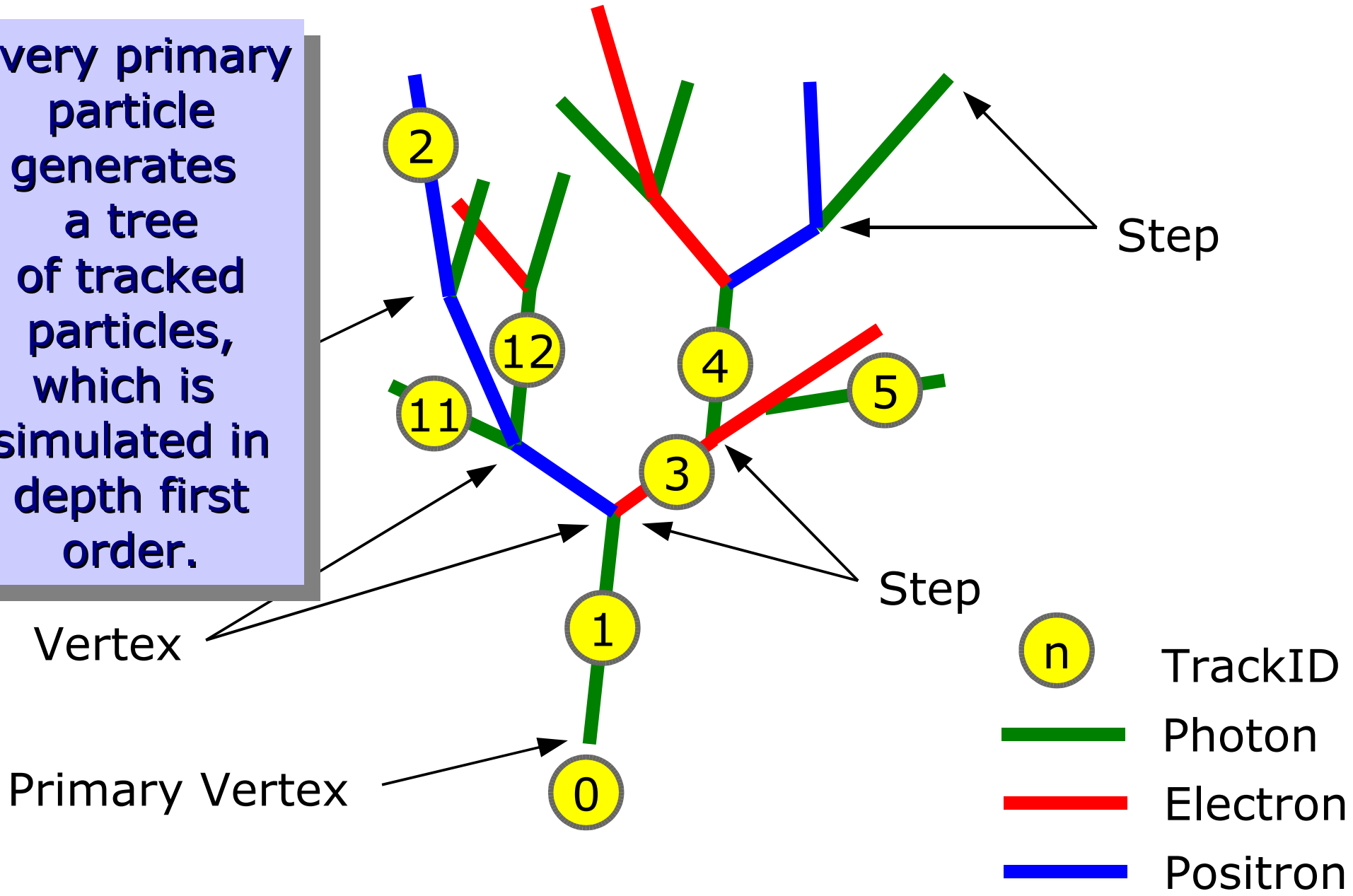


“Depth First” Tree Traversal



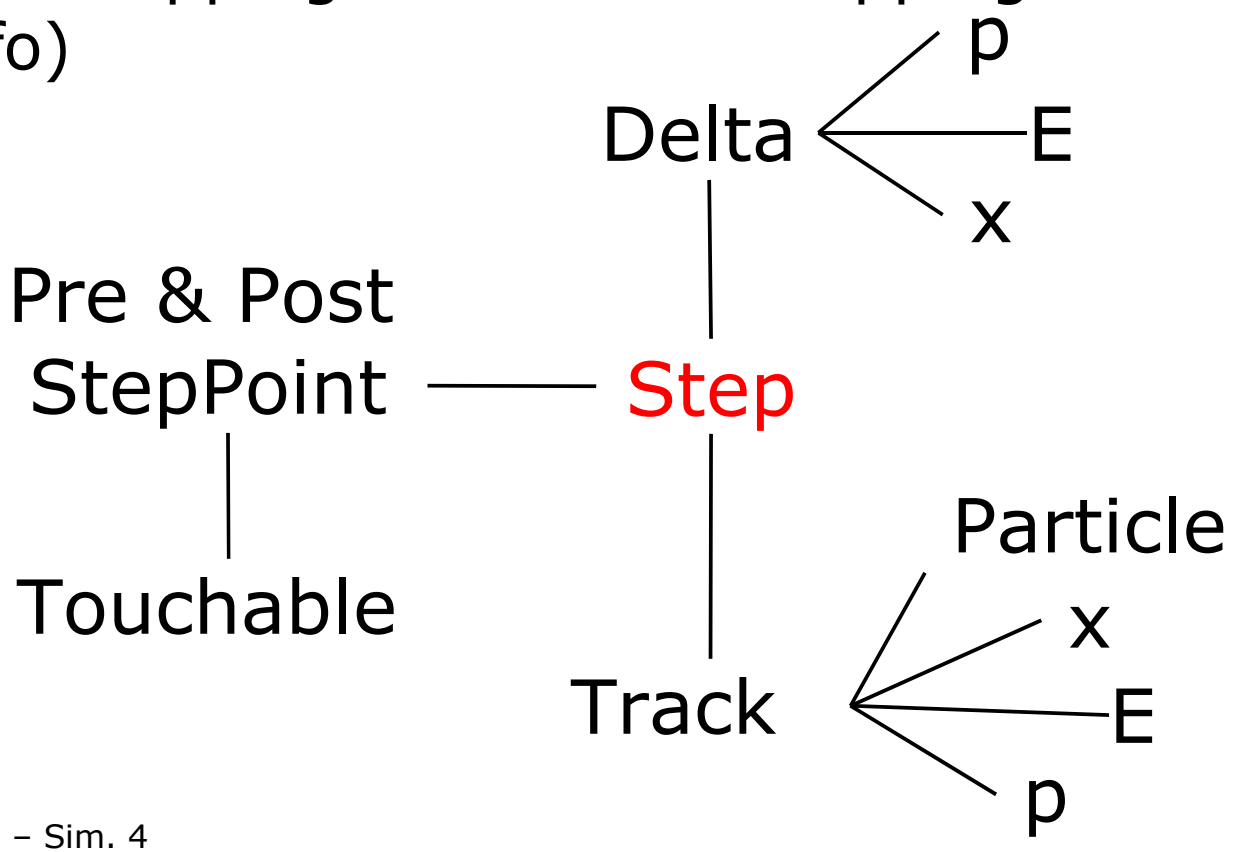
Track/Step (once more)

Every primary particle generates a tree of tracked particles, which is simulated in depth first order.



Stepping Action

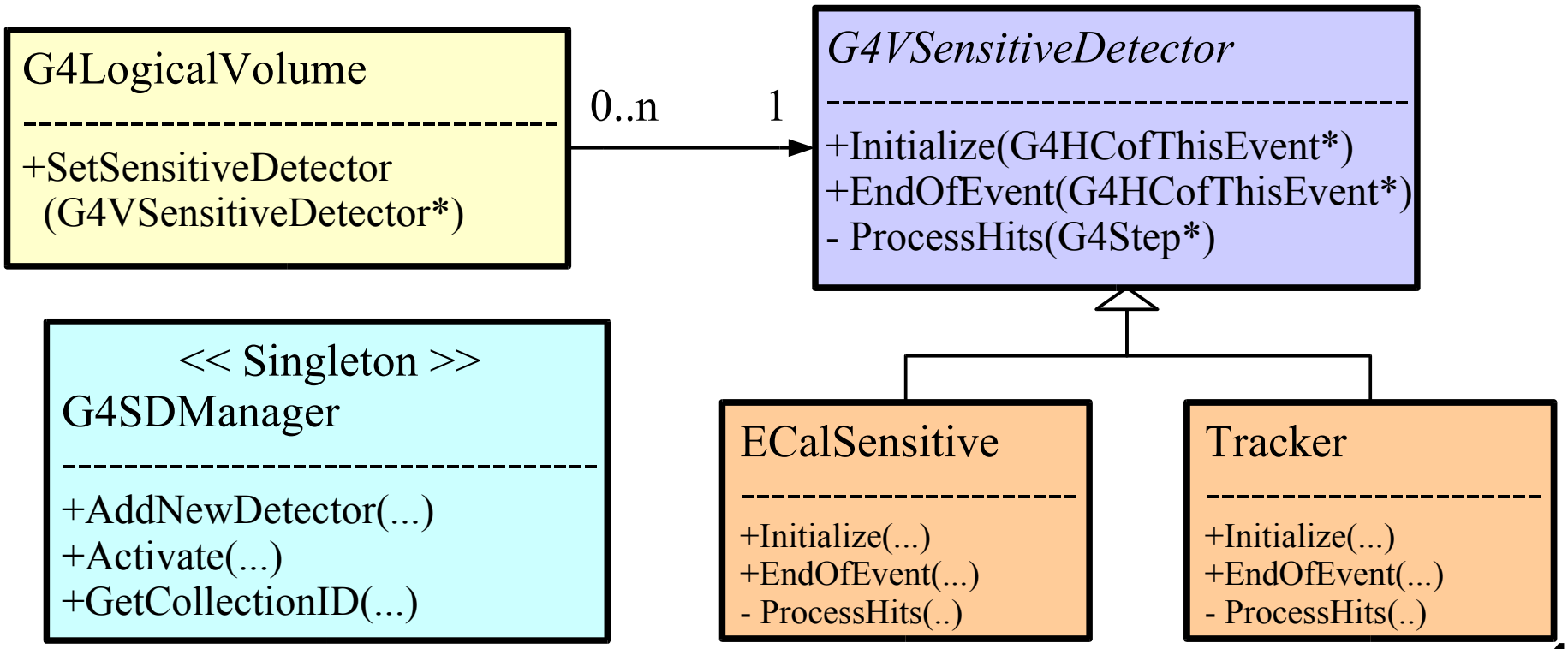
- G4VSteppingAction: finest granularity of obtaining information
  - register a sub-class to the Geant4 kernel
  - at every step: G4VSteppingAction::UserSteppingAction(G4Step * info)



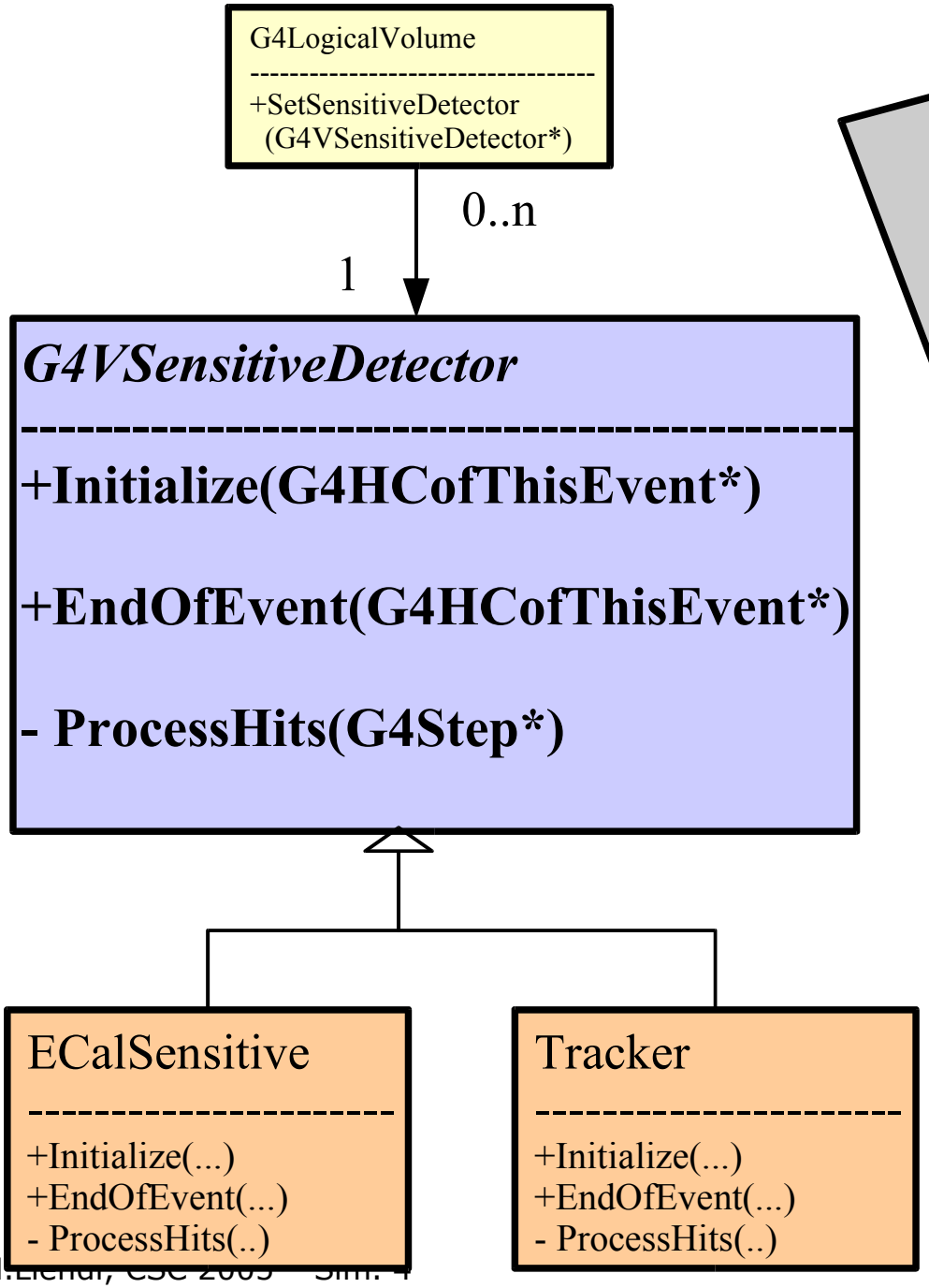
- The User-Actions shown before are generic:
  - invoked every time a Geant4 event occurs
  - independently invoked of the particularities of your experimental setup
- Want to have specific actions for specific detector parts:
  - Tracker: sensitive elements measure positions of single tracks
  - Calorimeter: sensitive elements measure integral information – energy deposit – not individual tracks
- Geant4 offers "G4SensitiveDetector"
  - specialized user-actions for different parts
- Geant4 supports you in bookkeeping simulation results
  - G4VHit interface, Hit collections
  - Digi interface, Digi collections

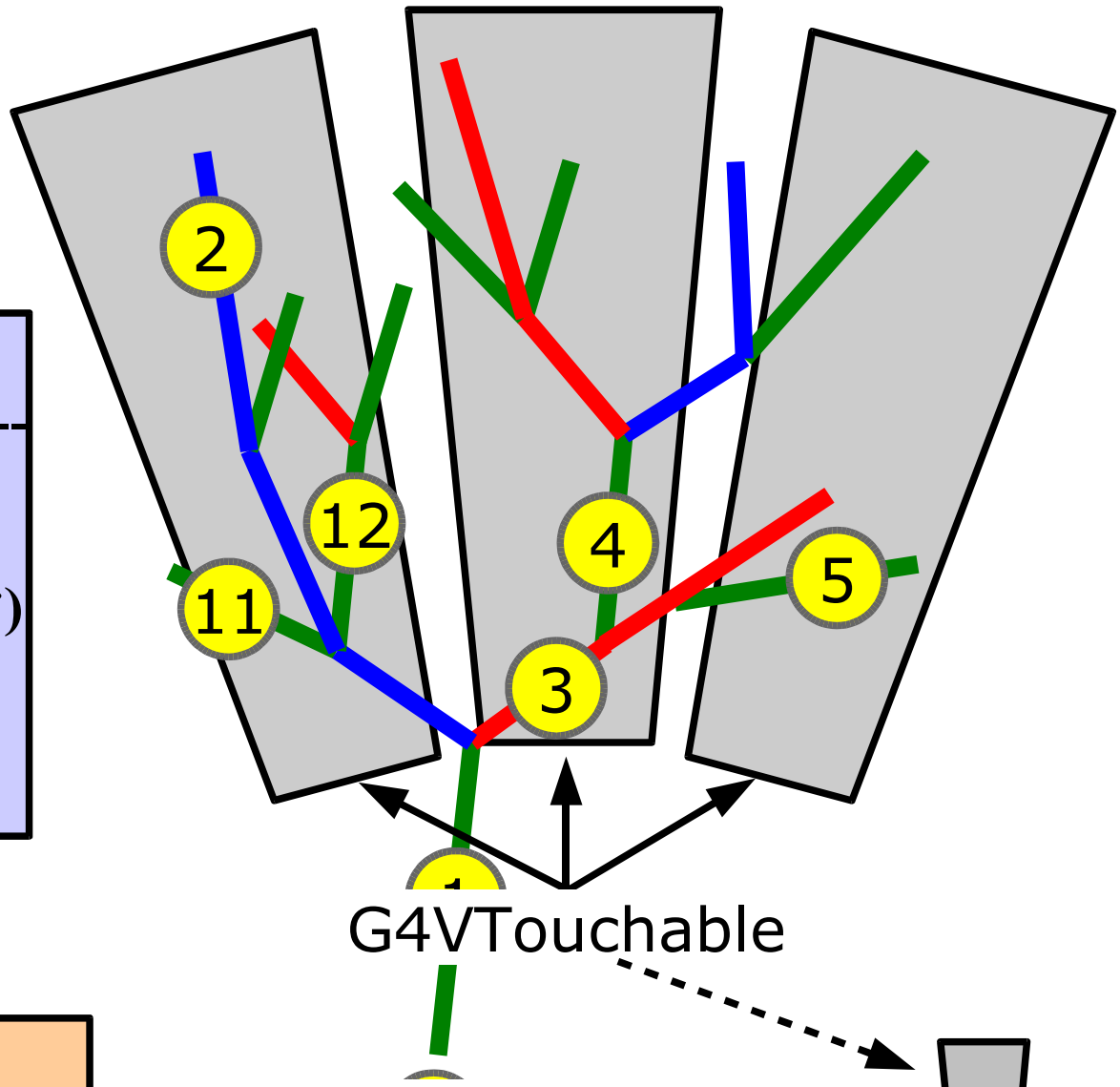
Sensitive Detector

- Main motivation: to get information from the parts of the detector that will do the actual measurement (e.g. silicon pixel detectors in the tracker, crystals in the ECal, drifttubes in the Muon-system, ...)
  - these parts of the detector are called "Sensitive Detector"
  - Sensitiveness is handled via logical volumes:

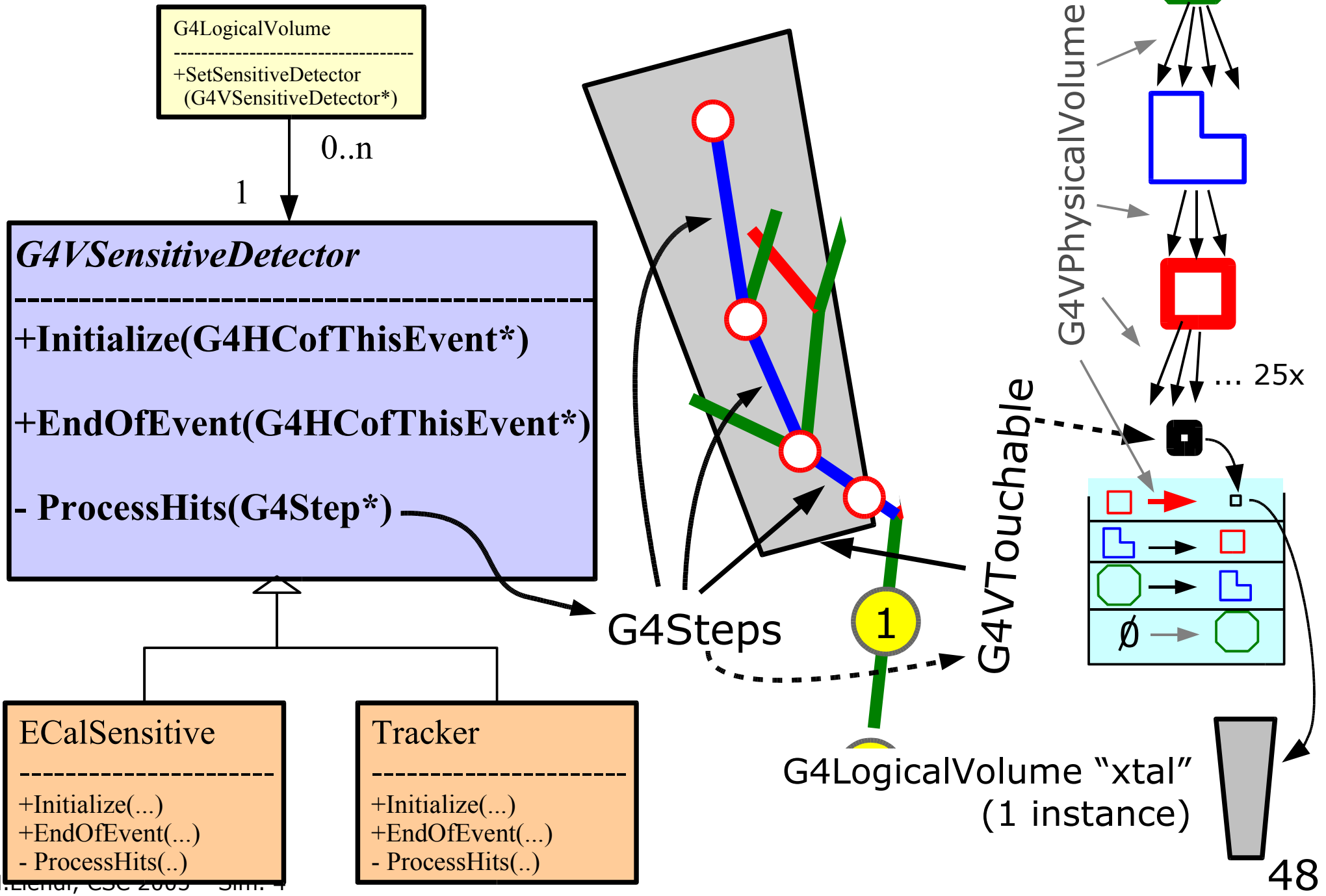


Sensitive Detector & Geometry



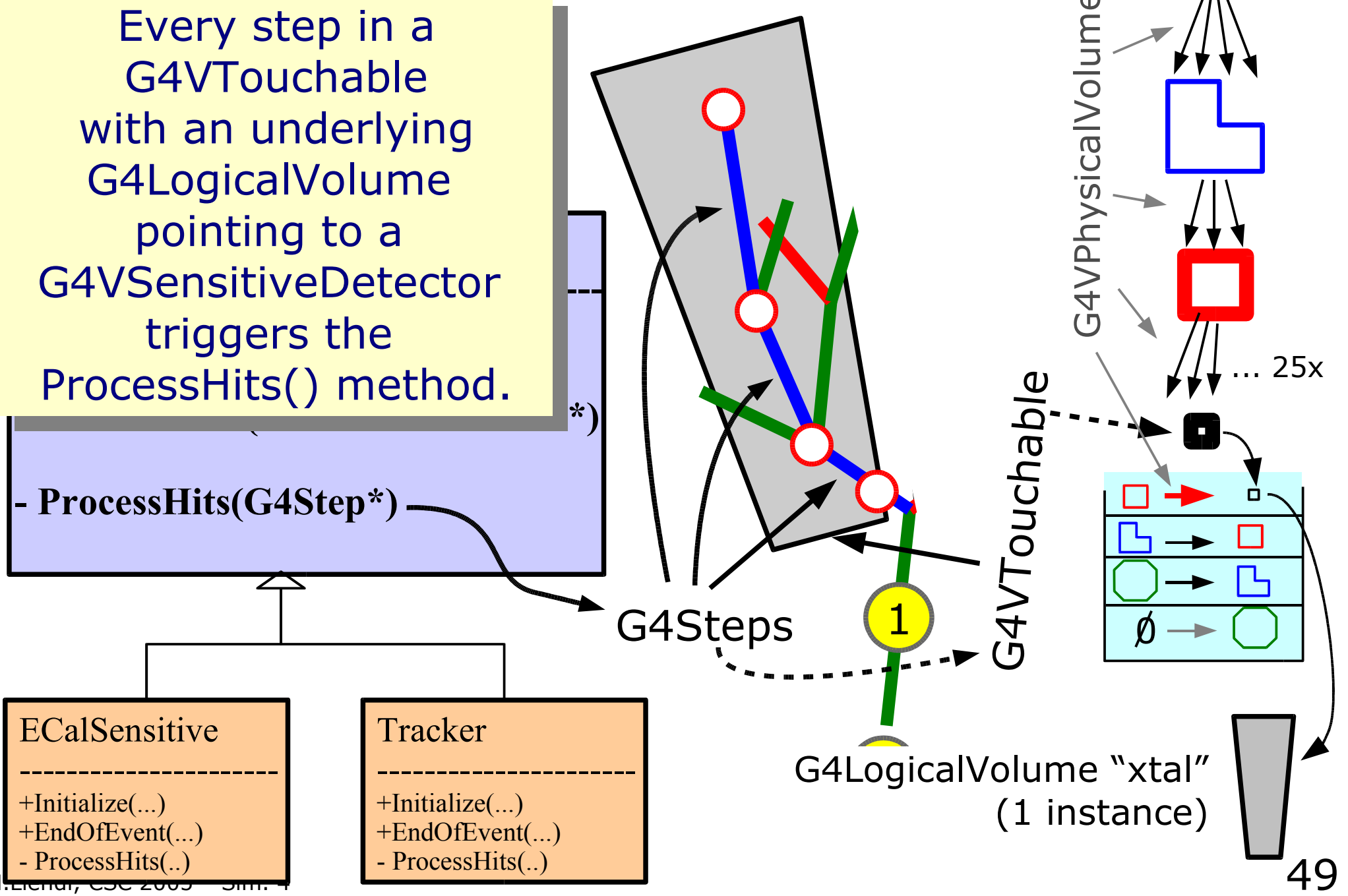


Sensitive Detector



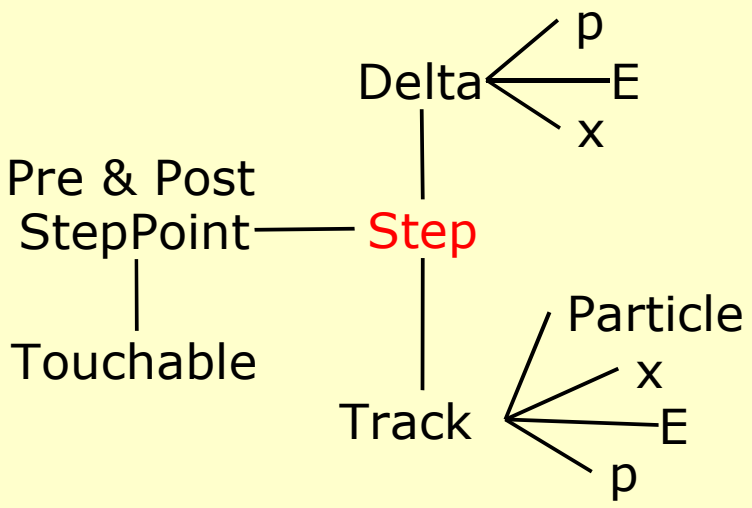
Sensitive Detector

Every step in a G4VTouchable with an underlying G4LogicalVolume pointing to a G4VSensitiveDetector triggers the ProcessHits() method.

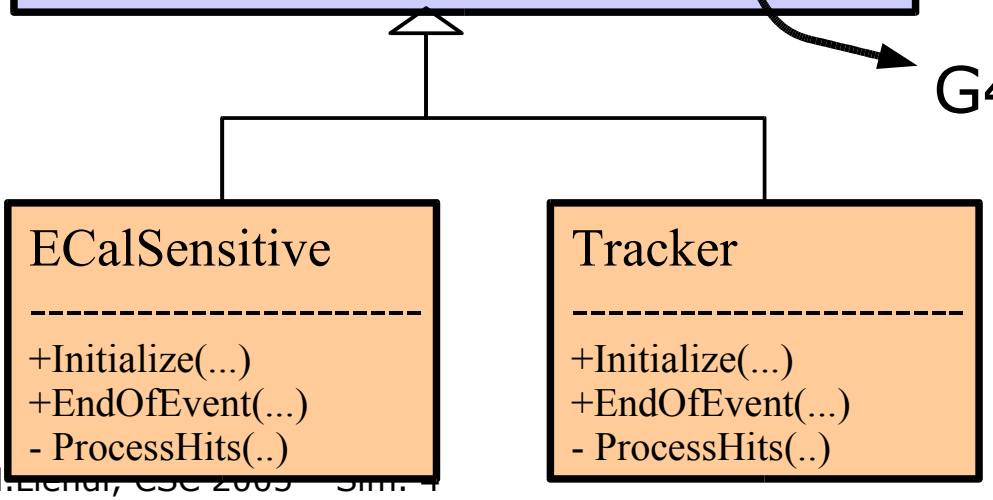


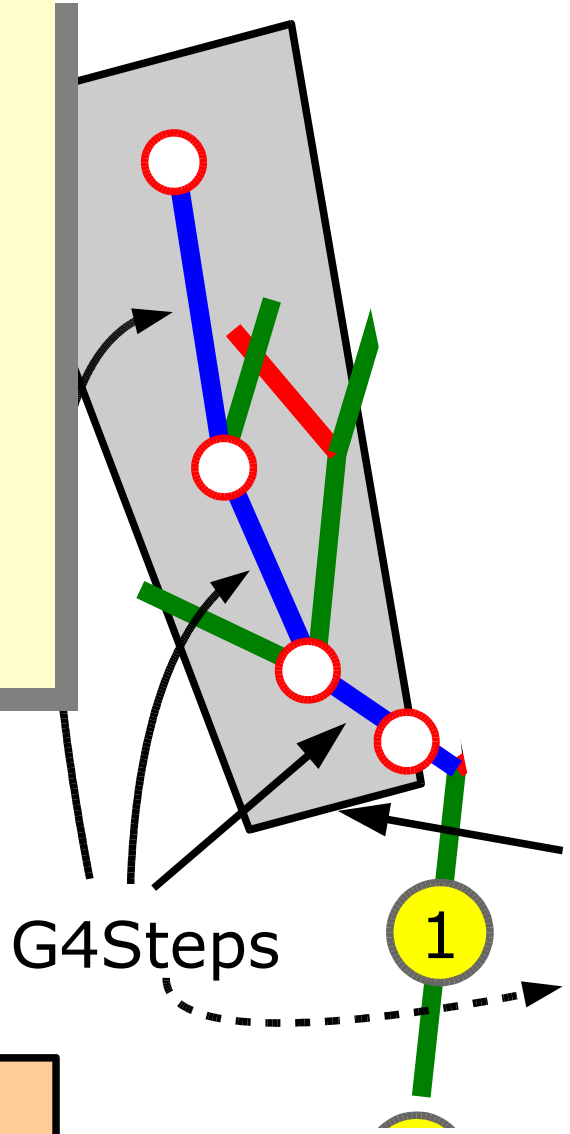
Sensitive Detector

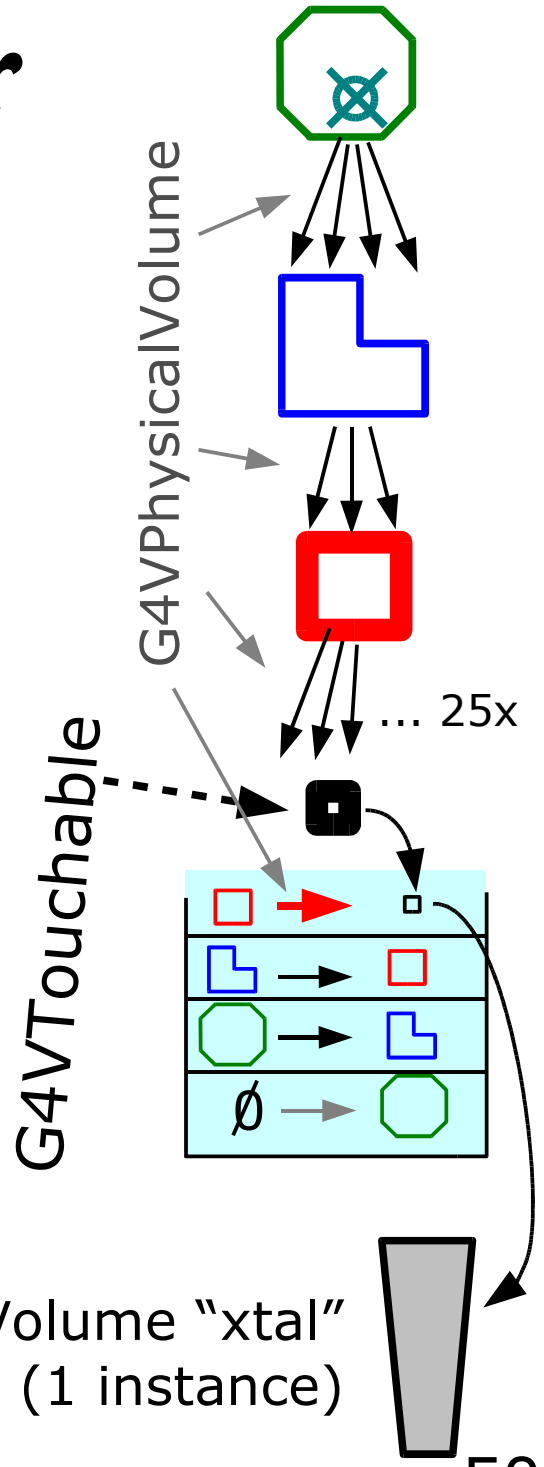
The G4Step offers access to:



- ProcessHits(G4Step*)



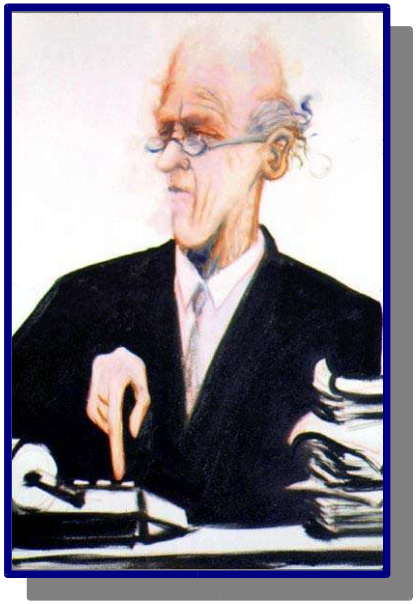




THE Simulation FAQ

Q: What to do?
A: Tedious bookkeeping!

The User Actions and Sensitive Detector classes are the means for filtering & bookkeeping the huge amount of simulation information available during tracking.



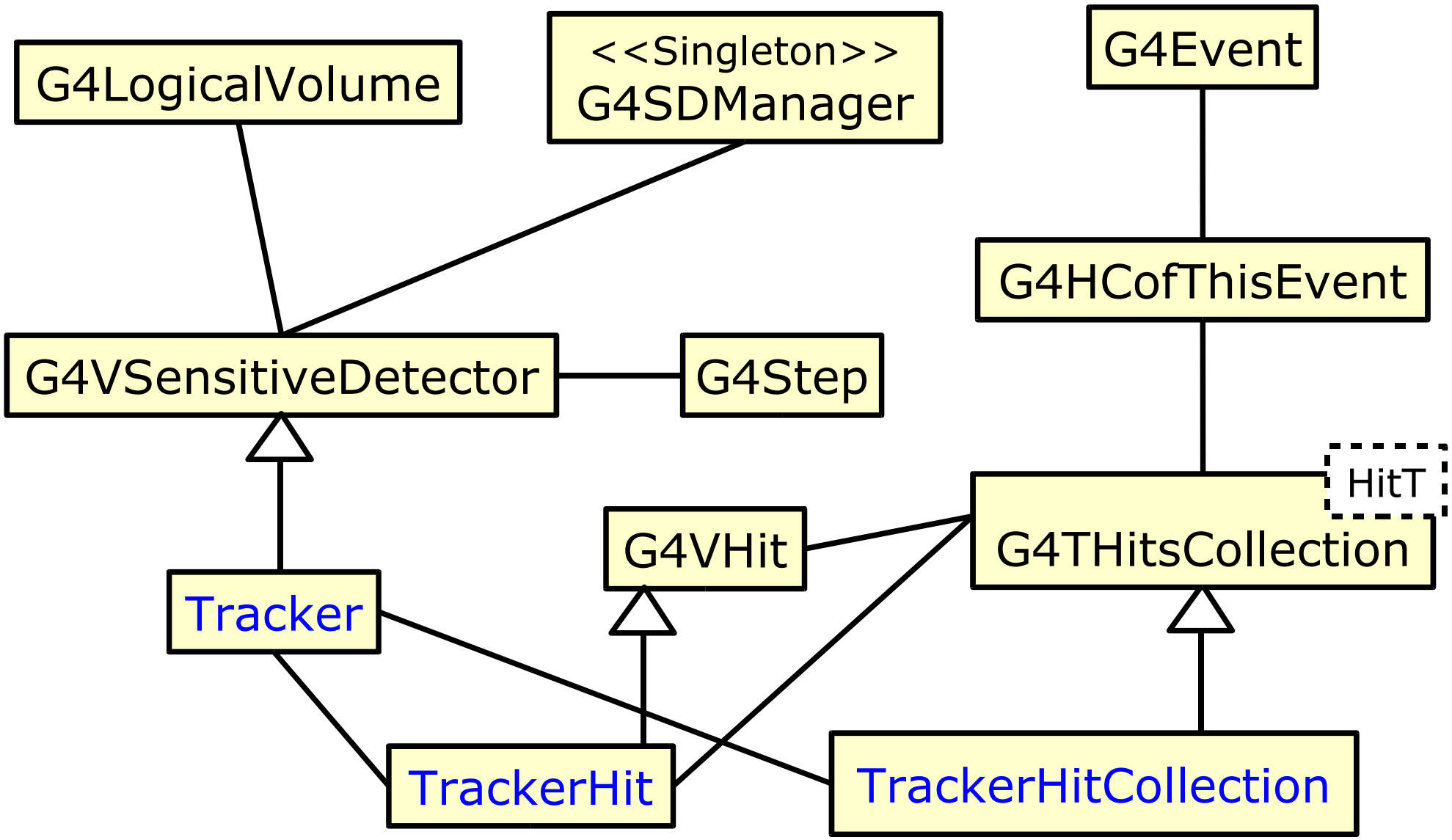
Robert D. Williams
"The Accountant"
oil on canvas, 1997

Sensitive Detectors create Simulated Hits, which typically contain data as the real detector elements ideally measure.

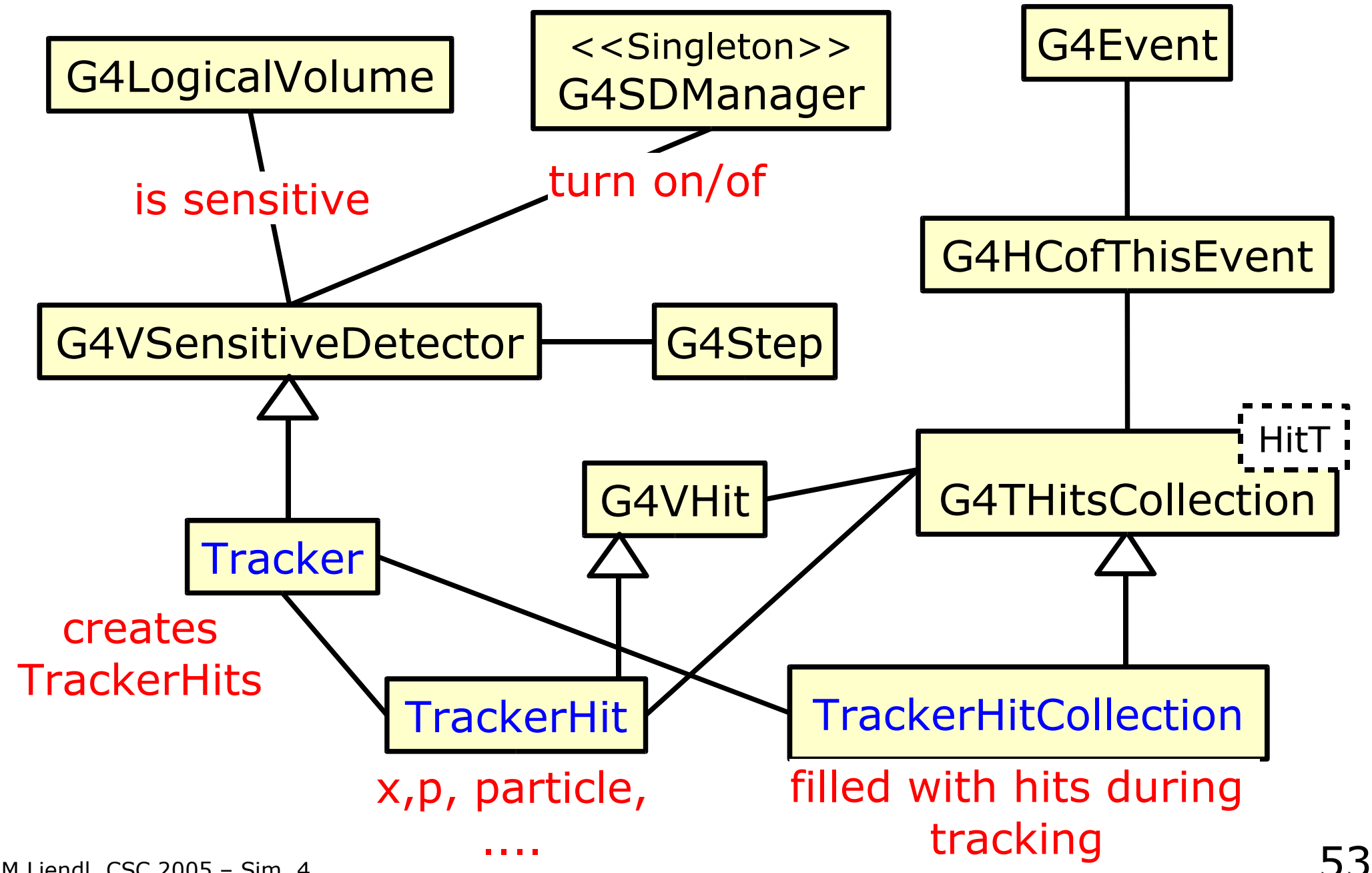
Geant4 helps in providing interfaces for Hits and Hit Collections.

However, many G4 based experiment simulations implement their own machineries for collecting & storing Hits.

Overview of G4 Hits & Collections



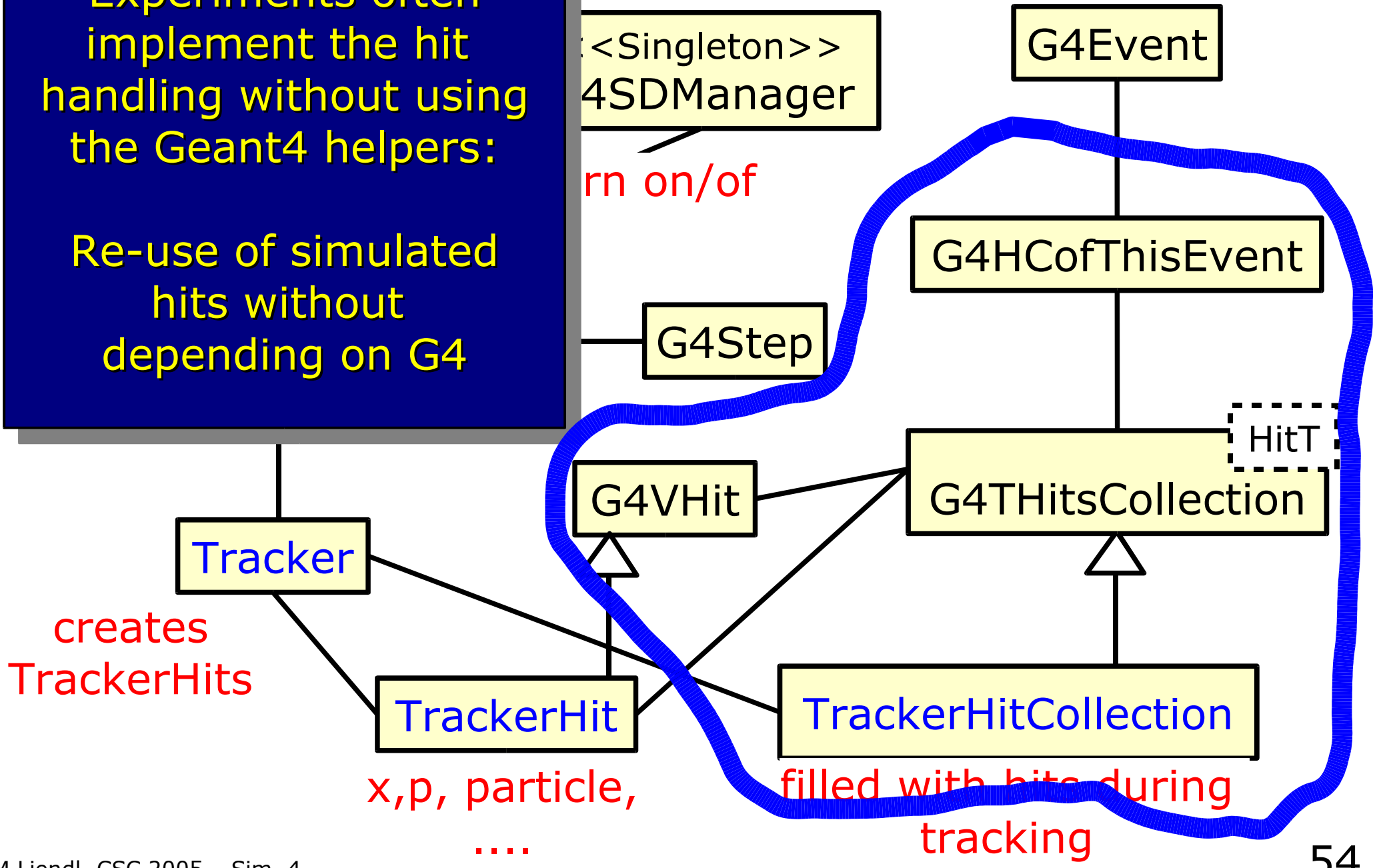
Overview of G4 Hits & Collections



Overview of G4 Hits & Collections

Experiments often implement the hit handling without using the Geant4 helpers:

Re-use of simulated hits without depending on G4



A small remark ...

Objects of G4Step, G4Track, G4Event, G4Run, G4VHit, G4THitCollection, ...
are transient, even within one simulation execution.
They are deleted by Geant4 as soon
as they are not needed anymore for the subsequent simulation.

You have to use the User-Actions to keep
data for longer than they are required by Geant4.

Usually, you want to store your simulation
results somewhere on disk ...

Digitization

= process of converting simulated physics information of sensitive detector parts into simulated electronic response

= the stuff that is expected to come out of the real detector

e.g.:
- convert deposited energy into ADC counts
- add electronic noise to measurement signal
- take latencies caused by singal processing into account

Digitization is highly detector specific!
That's why Geant4 can't offer too much support for it.

Digitization

Similar book-keeping tools as for hits:

G4DigiManager	G4SDManager
G4VDigitizerModule	G4VSensitiveDetector
G4VDigi	G4VHit
G4TDigiCollection<T>	G4THitCollection<T>
G4DCofThisEvent	G4DCofThisEvent

Differences w.r.t. to hit processing:

G4VDigitizerModule is not associated with any volume!
(G4VSensitiveDetector is associated with G4LogicalVolume)

G4VDigitizerModule::Digitize(..) is never called by the G4!
(G4VSensitiveDetector::ProcessHits(..) called by the G4 kernel)

G4RunManager::SetNumberOfEventsToBeStored(G4int n):

keep n consecutive events & their hit-collections in memory

Digitization

Similar book-keeping tools as for hits:

G4DigiManager

G4SDManager

G4VDi

G4VDi

G4TDi

G4DCo

Differenc

G4VDigiti

(G4VSensiti

Very often experiments don't use Geant4 Digitization classes at all and implement their own Digitization machineries!

CMS does this!

G4VDigitizerModule::Digitize(..) is never called by the G4!
(G4VSensitiveDetector::ProcessHits(..) called by the G4 kernel)

G4RunManager::SetNumberOfEventsToBeStored(G4int n):

keep n consecutive events & their hit-collections in memory

Summary -4-

- Stepping through a detector description
- Instrumenting a G4 simulation
  - mandatory initialization:
    - physics list
    - detector description
  - mandatory action hooks:
    - primary particle generation
  - optional action hooks:
    - run, event, track, stepping, stacking actions
- Hits & Sensitive Detectors
- Digitization

That's it!

That's it ...

... what I've wanted to tell you about Geant4 and the principles of Monte Carlo simulation

Several "secrets" remain as an "exercise for the reader" (G4-documentation, special G4 training weeks, ...)

- advanced geometry (parameterizations, assemblies)
- advanced tracking (parallel read-out geometries, event biasing & scoring)
- fast simulation support (user takes over tracking responsibility under user defined conditions)
- user interface support (G4 macros, interactive simulation, user commands)
- visualization (many supported visualizers, many options)

Some answers already now - by Tony & Mark

- user interface support (G4 macros, interactive simulation, user commands)
  running G4 through JAVA ANALYSIS STUDIO
- visualization (many supported visualizers, many options)
  WIRED PLUGIN
- Detector Description without coding
  GDML – Geometry Description Markup Language
  an XML for describing G4 compatible geometries
- Hit´s & Co
  Analyzing simulation output – tools for physics studies (histograms, ...)

:-)

(small print: the summary is yet to come)

Summary

the "needle vs. balloon end of lecture indicator"

- Why do we need simulation?
  - detector studies, "unmeasured" physics
- Types of simulation
  - event generators:
    - detector independent, simulates "physics"
  - experiment simulation
    - passage of particles through matter - >>Geant4<<
    - detector response (digitization)
- Basic Ingredients (and how to use them in Geant4)
  - quantities, units
  - materials & their static character in Geant4
  - geometry
    - basic requirements (shapes, materials, external fields)

Summary

- Details of Geant4 geometries
  - G4LogicalVolume, G4VSolid, G4Material
  - geometrical aspects (inside, outside, ..) of solids
  - hierarchies - G4VPhysicalVolume & the geometry graph
  - flexible & extensible model
- Principles of tracking of particles in Geant4
  - Geometry context during stepping: G4VTouchable
    - stack of G4VPhysicalVolumes at every StepPoint
  - geometrical aspects of tracking
    - steps between volume boundaries
    - particularities in external (magnetic) fields

Summary

- Side track: origins of the Monte Carlo method
  - tracking & sampling principles haven't really changed
- Principles of physics interactions in bulk material
  - mean free path length
  - tracking with physics, concurrent processes
  - basics of the Geant4 physics model: processes & particles
- Basics of a Geant4 simulation
  - mandatory user-actions:
    - detector description, physics list, primary particles
  - Geant4 event-loop & optional user-actions
    - run, event, track, step and their user-actions
  - Sensitive detectors, & hits; Digitization & digis

Want to know more?

- http://www.cern.ch/geant4
  - FAQs, user forum, problem reporting, talks & papers, the code itself (sources and some supported binaries)
  - Collections of Tutorial
  - Read the Fine Manuals: User manuals
    - for application developers, for Geant4 extensions
    - physics manual, SW reference manual
- Visit one of the excellent Geant4 tutorials (typical 3-5 days)
  - USA: http://geant4.slac.stanford.edu/
  - Europe: http://www.cern.ch/geant4


a real shower!


Thank You!
a real shower!