

CSC 2006, Helsinki

Zetor
where traktor drivers are hungry
and cows are nervous...
► Check the menuz

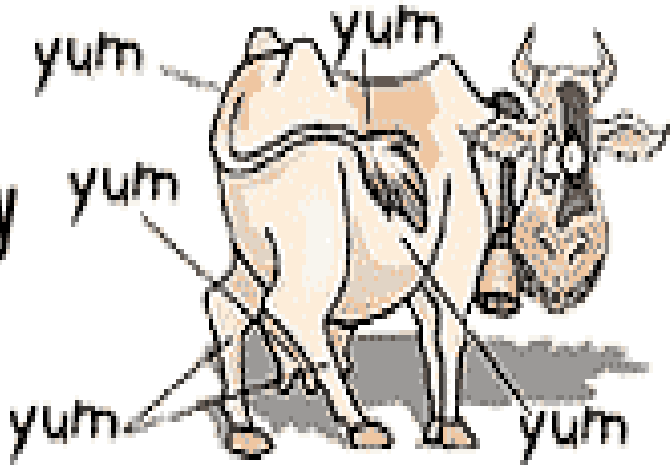


Table of Contents

- 1) How to find a cow
- 2) How to kill it
- 3) How to cook it
- 4) How to eat it




CSC 2006, Helsinki

GEANT4

where students ; are hungry
and lecturers ; are nervous...

► Check the menuz

A collage of images including a man with glasses, a cartoon cow, and the word 'yum' repeated several times.

Ups, unfortunately, we're not in Aatos' programme,
but in Francois' ... ;-)

Experiment Simulation

CERN School of Computing 2006
Helsinki

Lecture 4

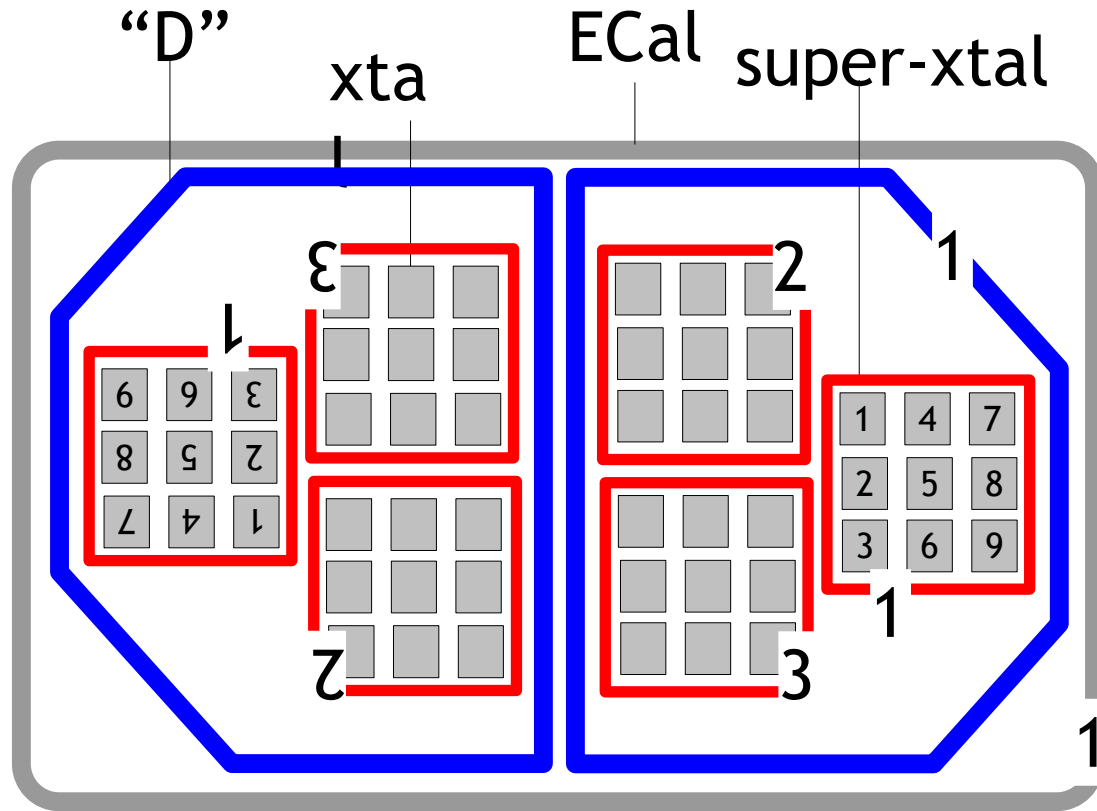
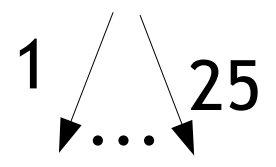
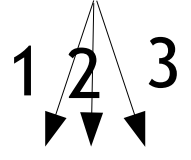
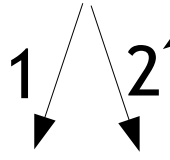
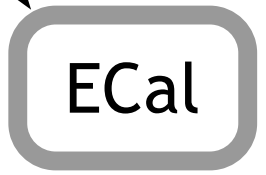
Unfinished Business!

- 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
 - Tracking through a detector description

Graph Structure

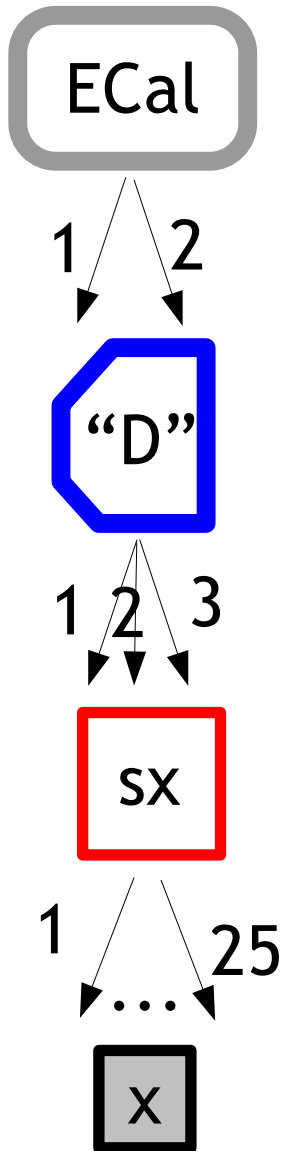
logical volume

physical volume with copy-number



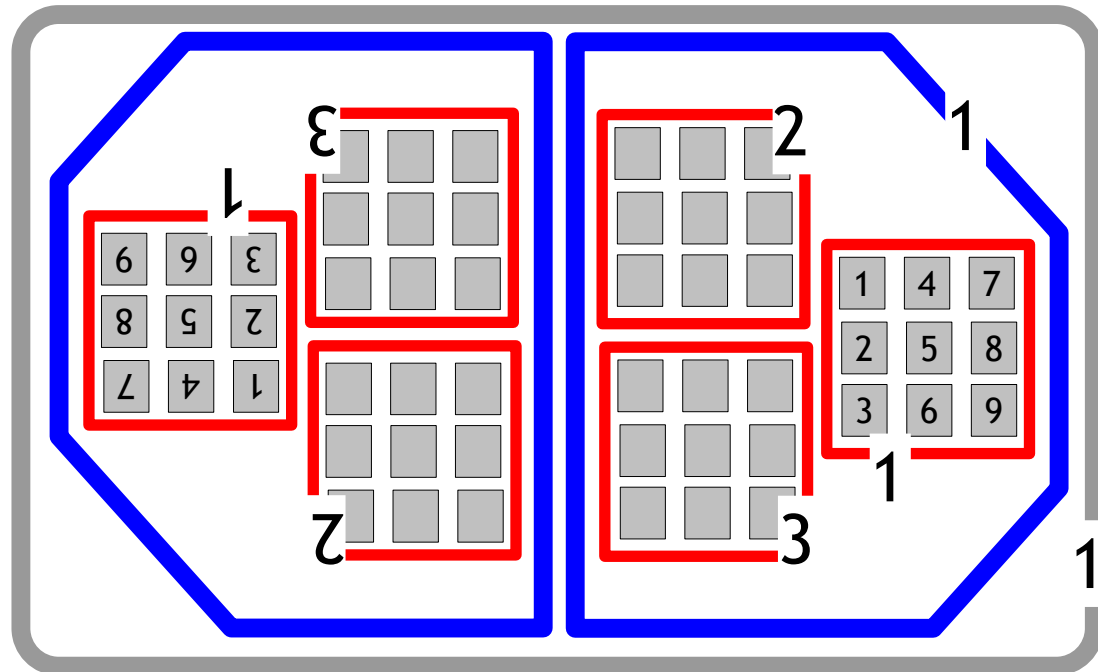
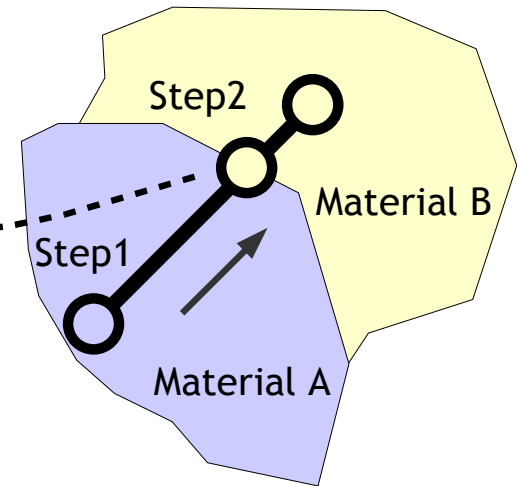
simplified version of an ECal ...

Stepping & Geometry

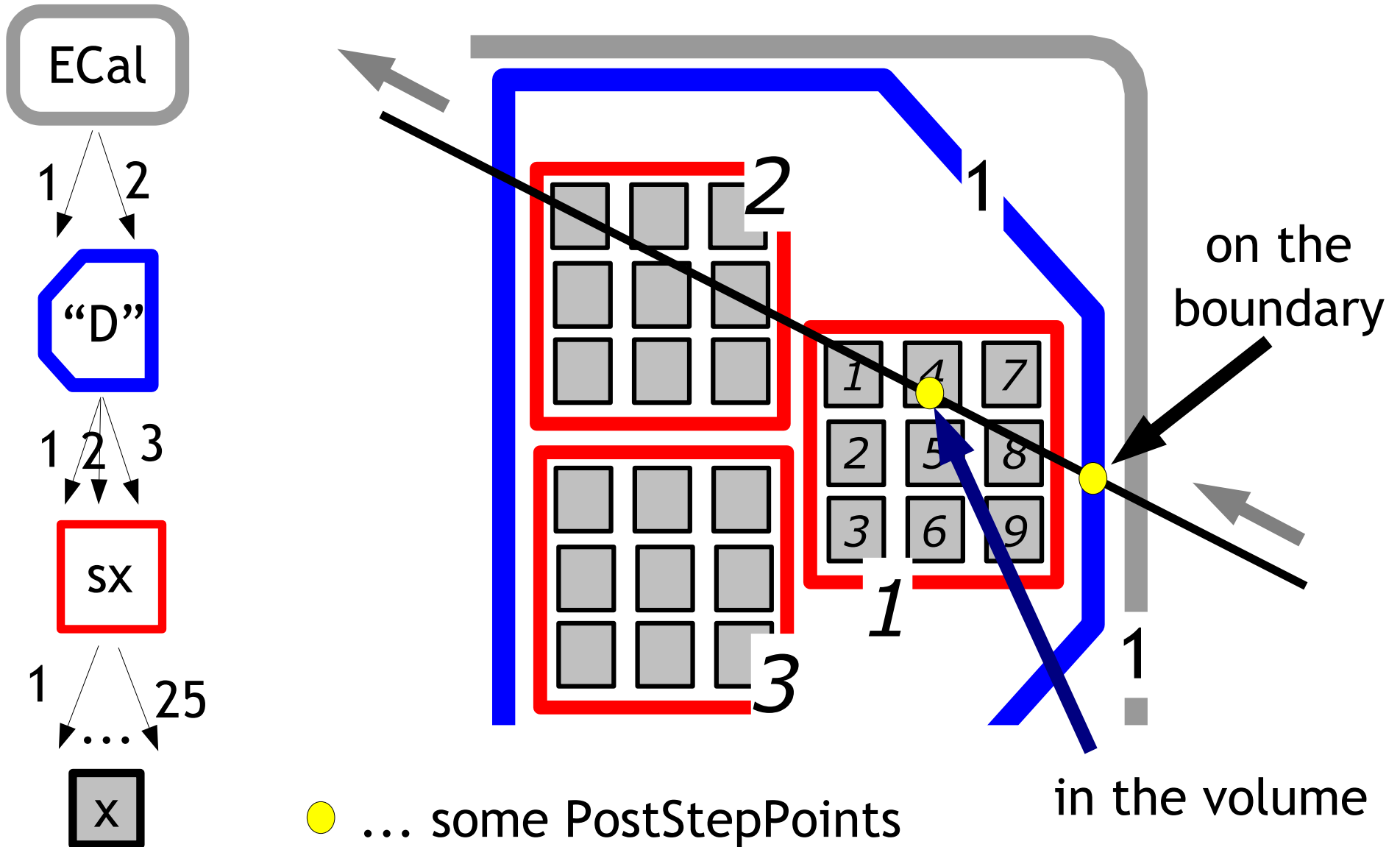


Remember?
 Steps & StepPoints
 determined by boundaries
 and physics processes!

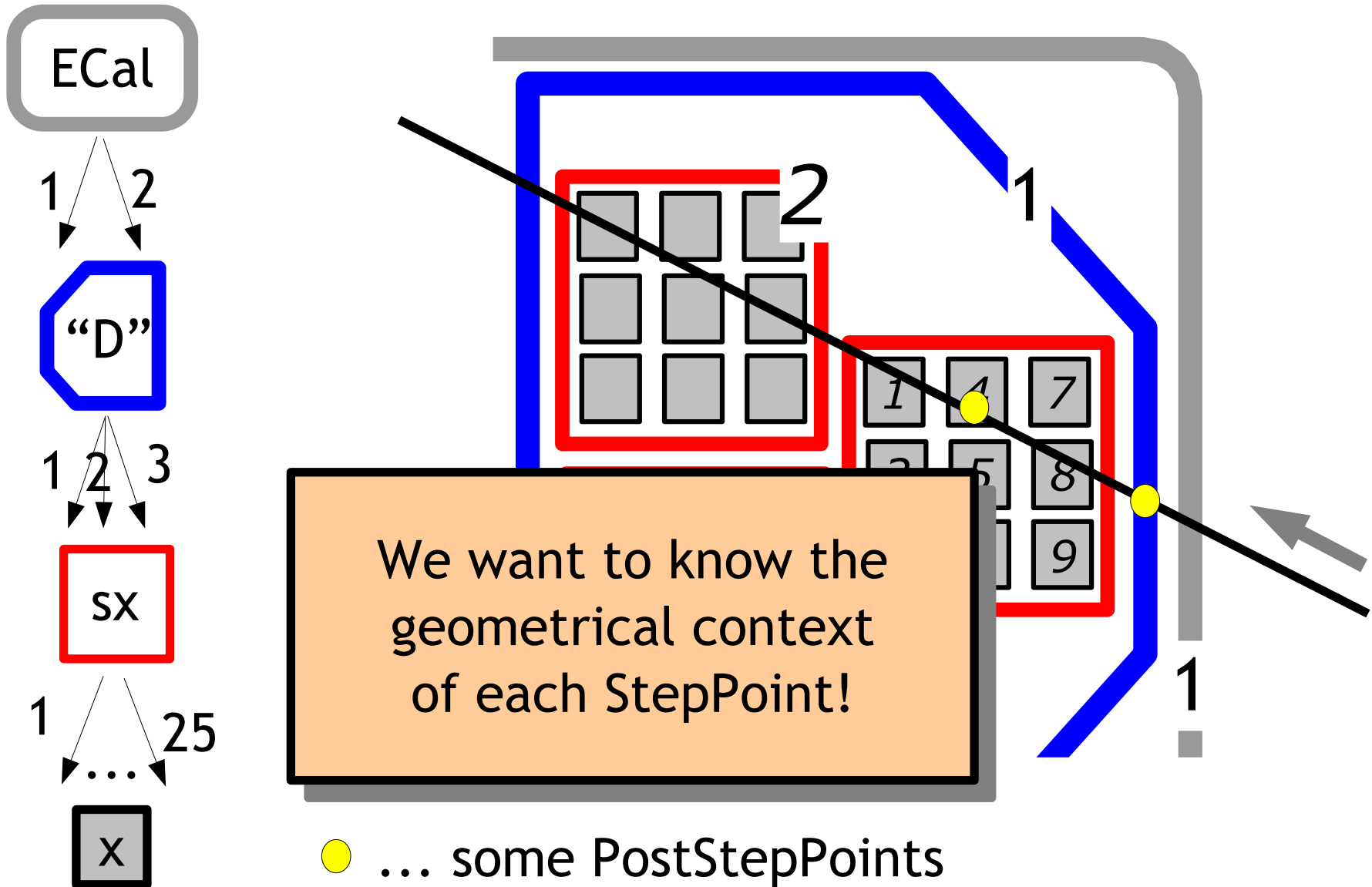
PostStepPoint1
 =
 PreStepPoint2



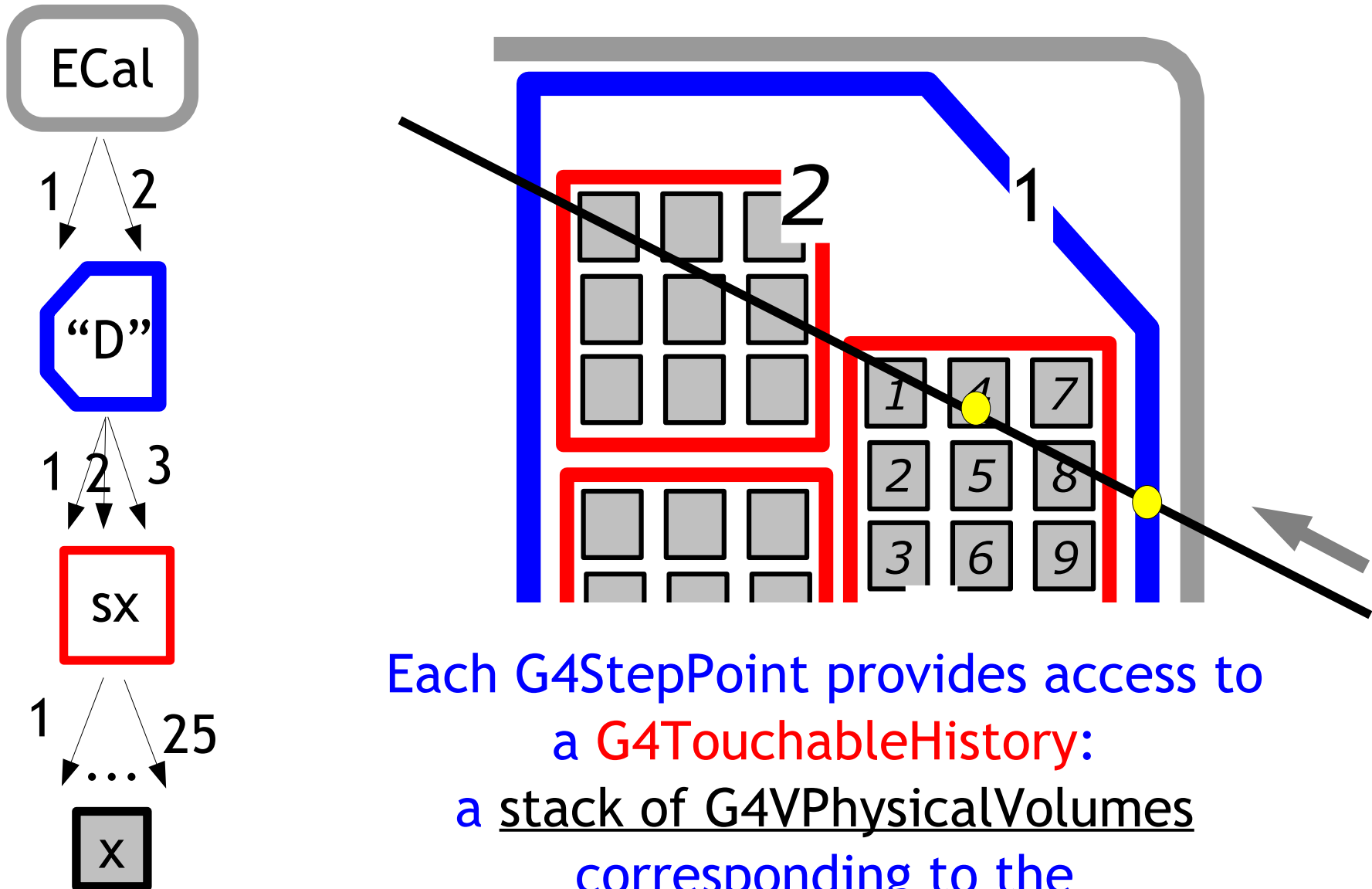
Stepping & Geometry



Stepping & Geometry

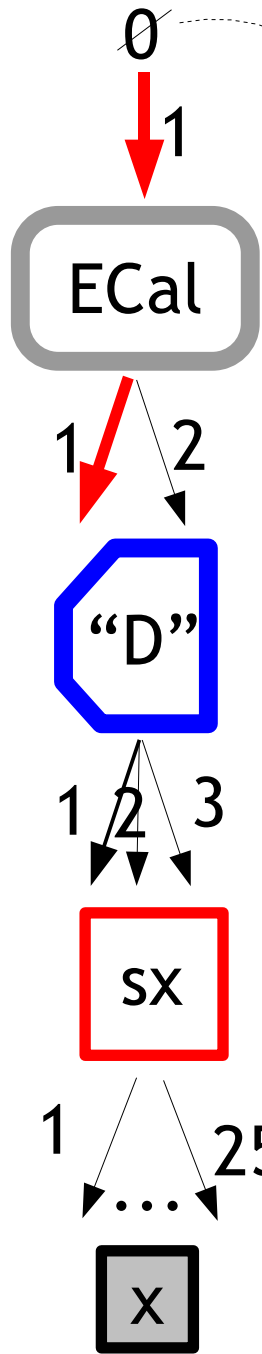


Touchable-History



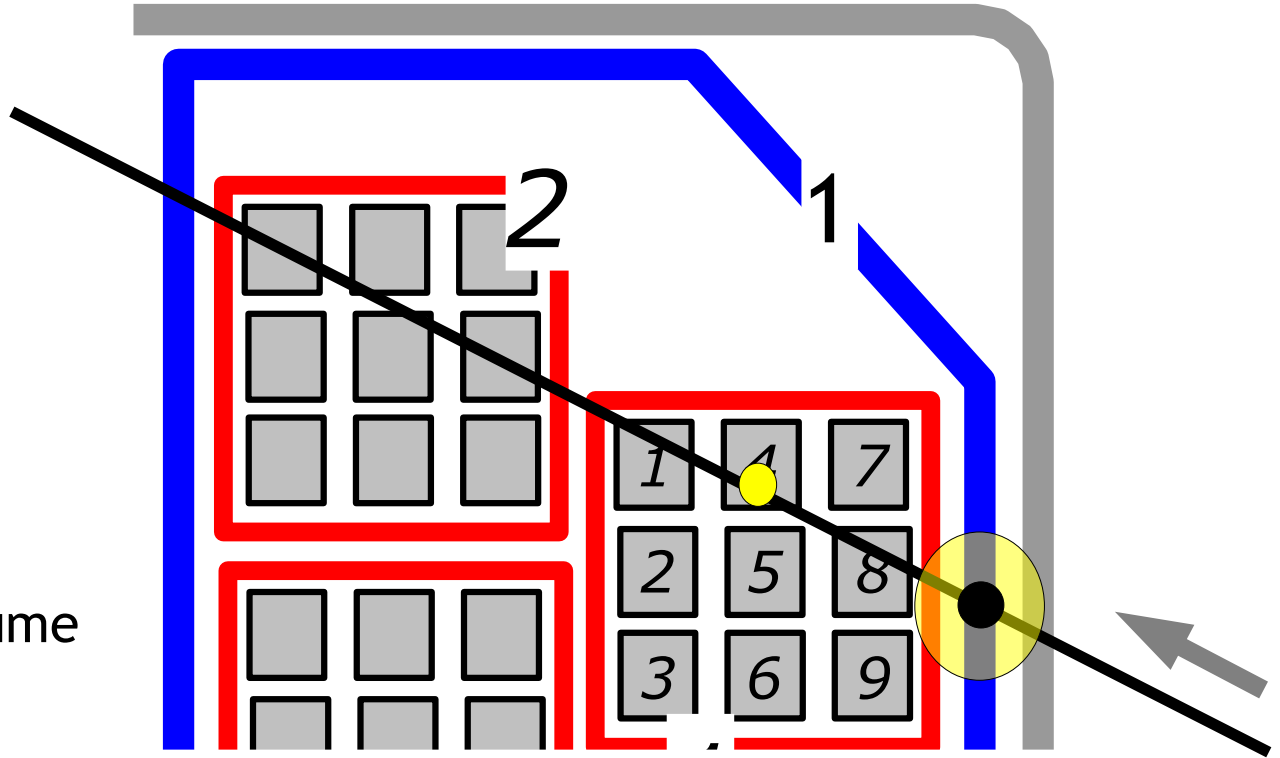
Each G4StepPoint provides access to
a **G4TouchableHistory**:
a stack of G4VPhysicalVolumes
corresponding to the
geometrical context of the StepPoint.

Touchable-History



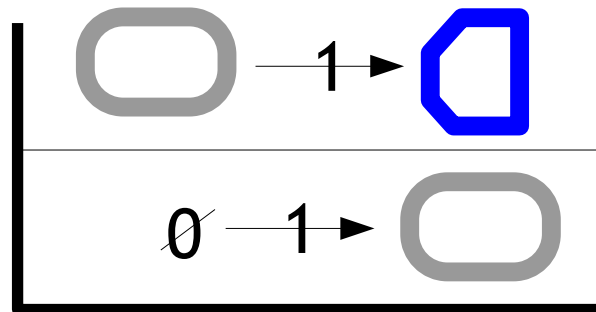
world
volume has
zero-ptr as
"mother"

History means:
geometrical
history back
to the root-volume

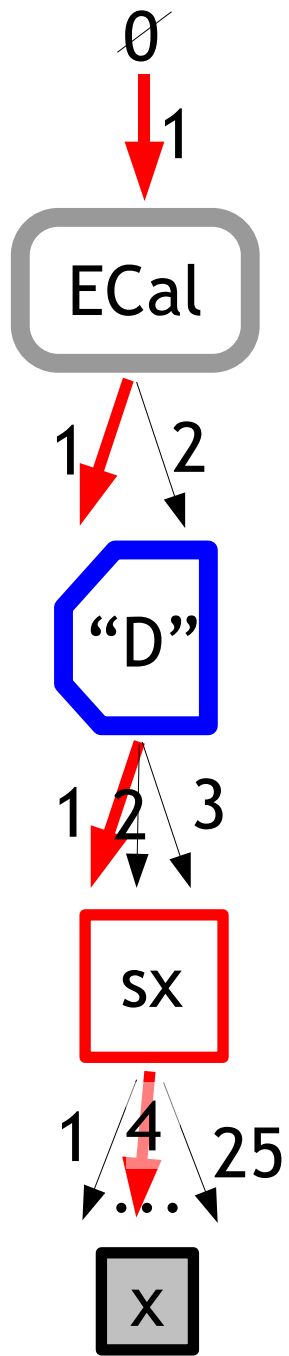


G4TouchableHistory: stack of physical volumes

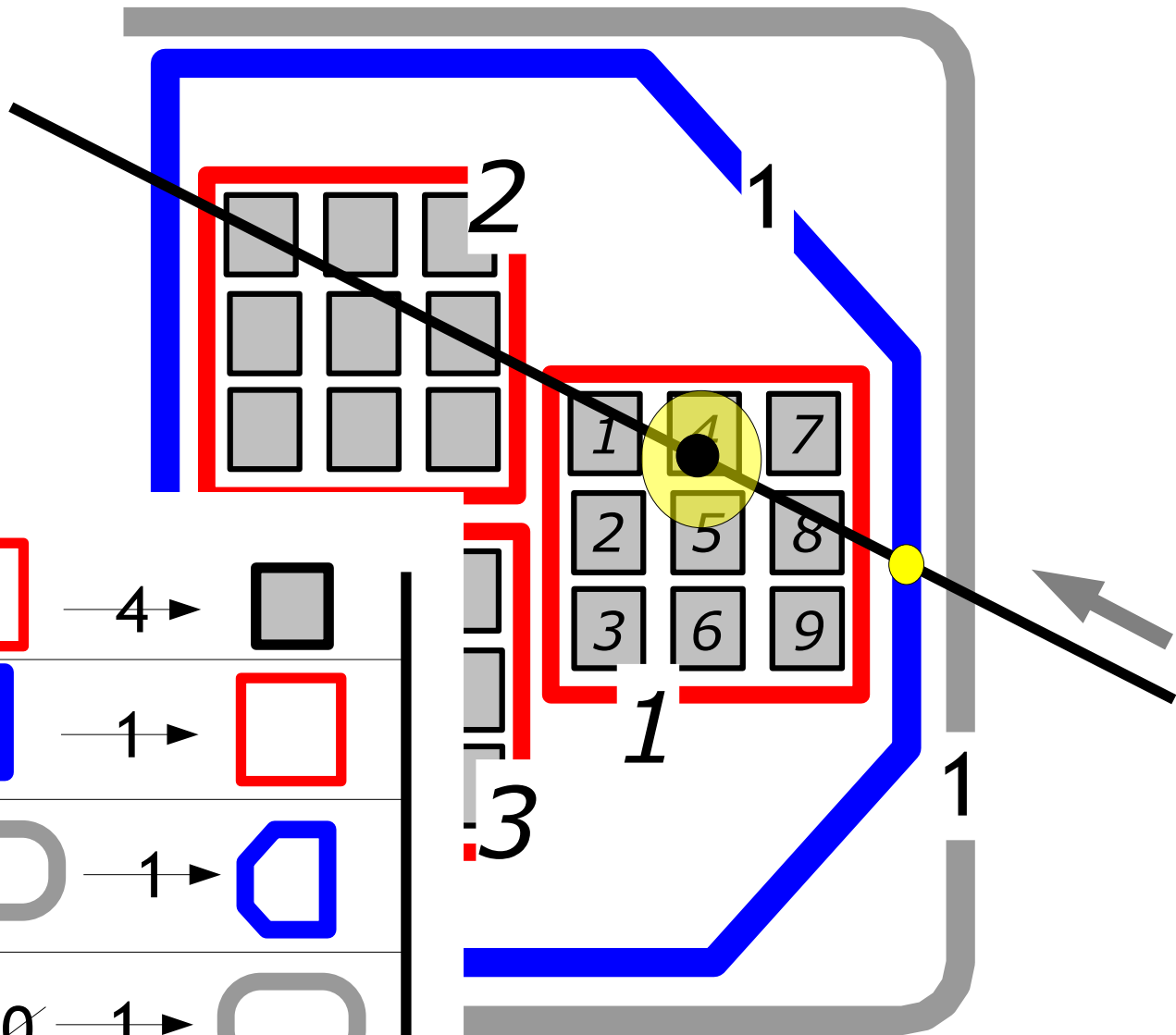
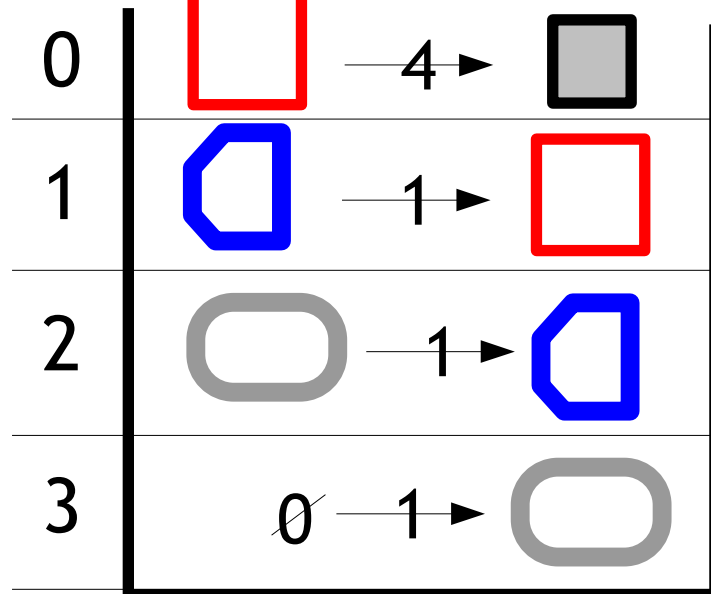
a stack:



Touchable-History



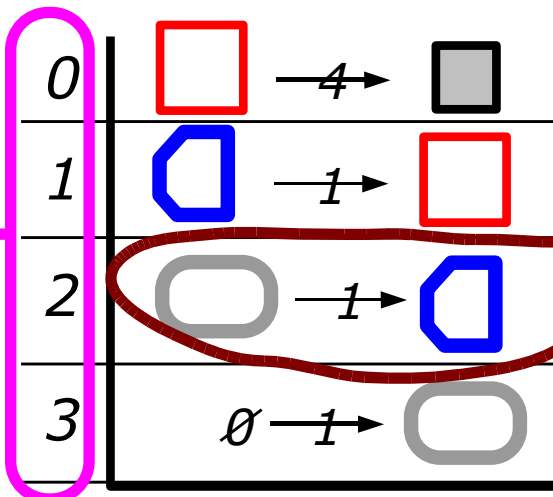
Level



Touchable-History

Name	Class	Type	Parameters
<u>G4VTouchable (md)</u>	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 = (from a G4 callback - see later);
```

```
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 ...

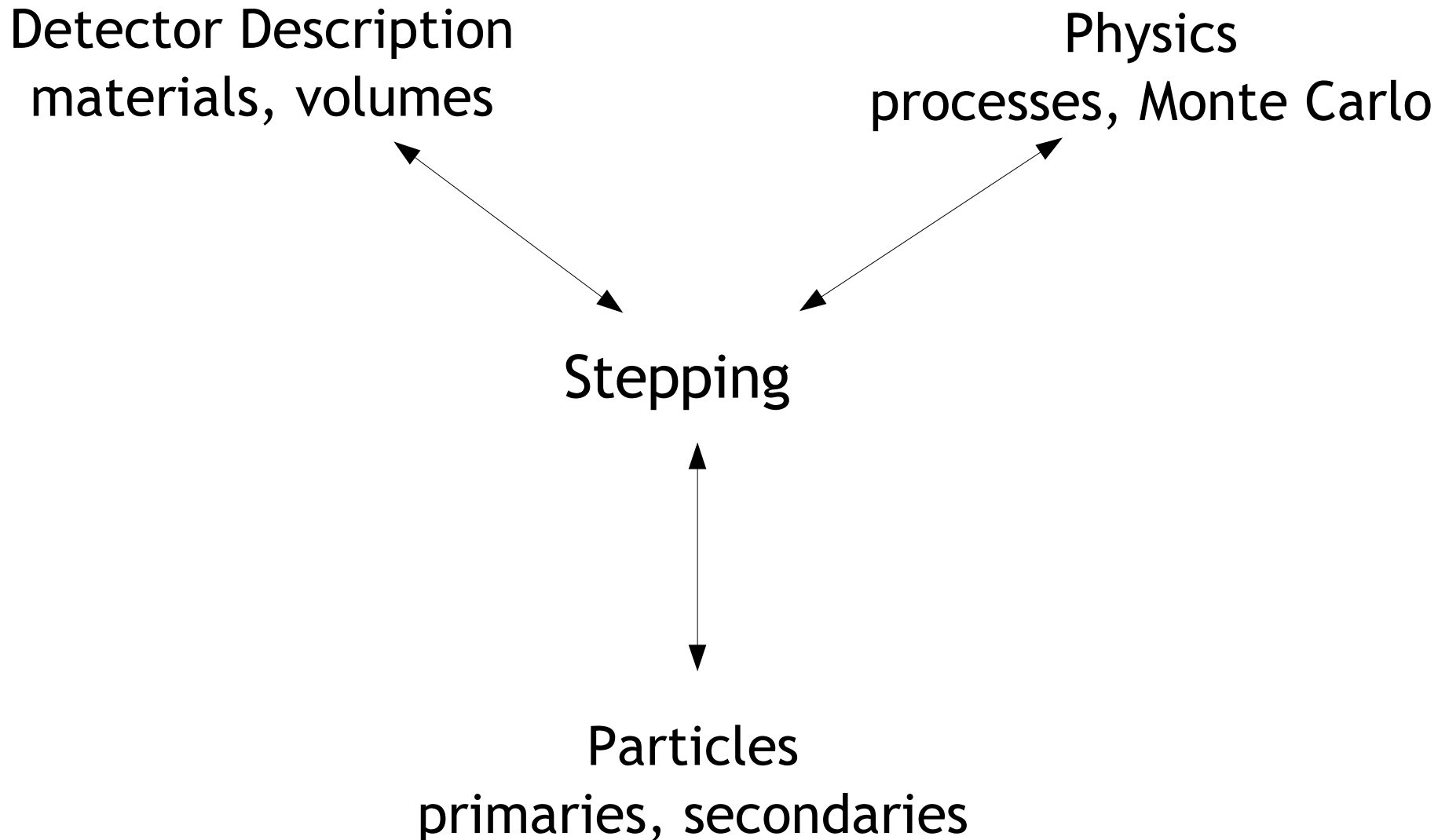
Uff!! ... But hang on!

- 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
- Summary
- Exercises

The Missing Link(s)



The Missing Link(s)

Meta-Question: Why? ✓

Detector Description
materials, volumes

Physics
processes, Monte Carlo

Where?

How?

HowTo get
it started?
When is what?

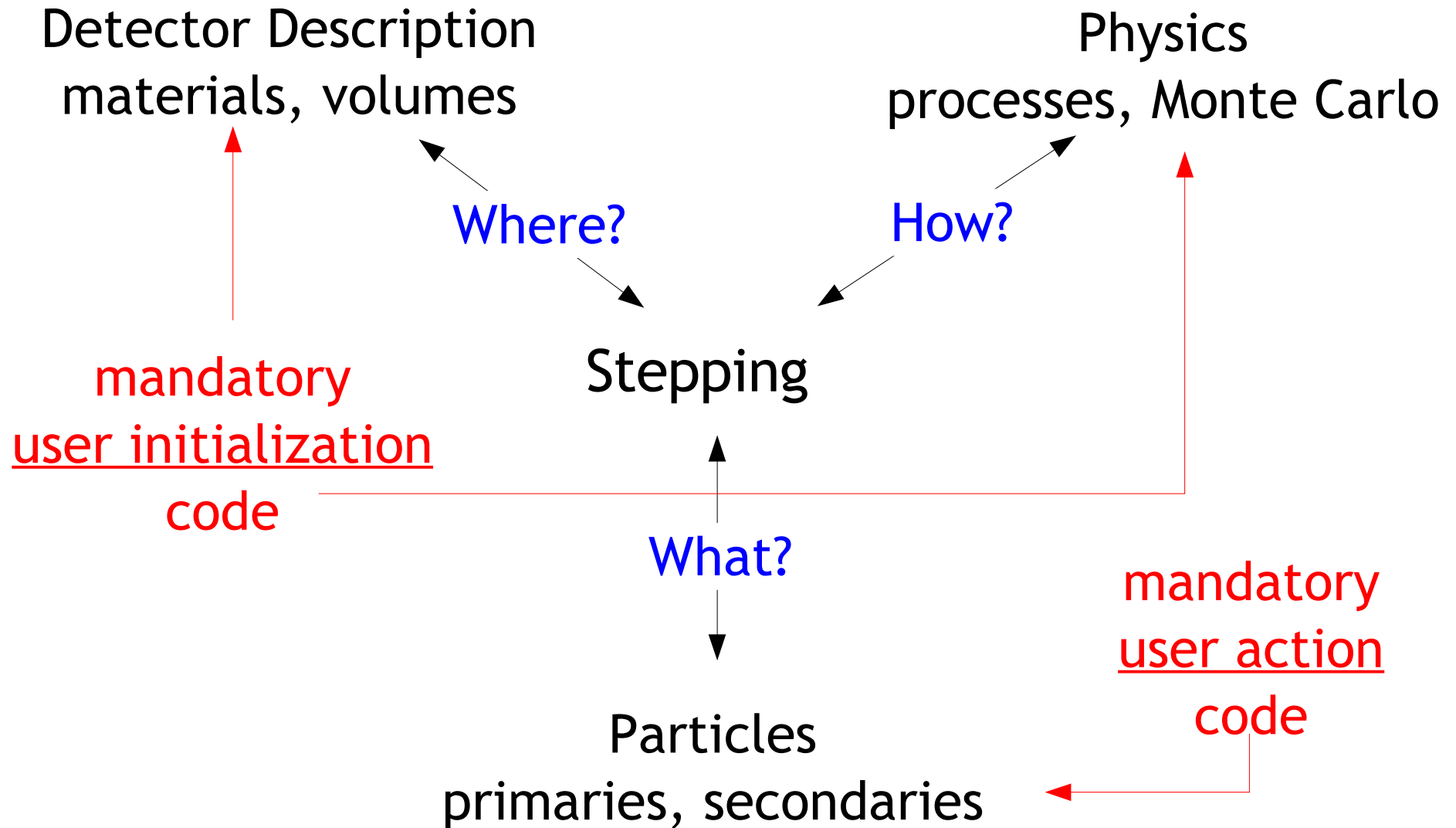
Stepping

What's the
Output?

What?

Particles
primaries, secondaries

User initialization/action code



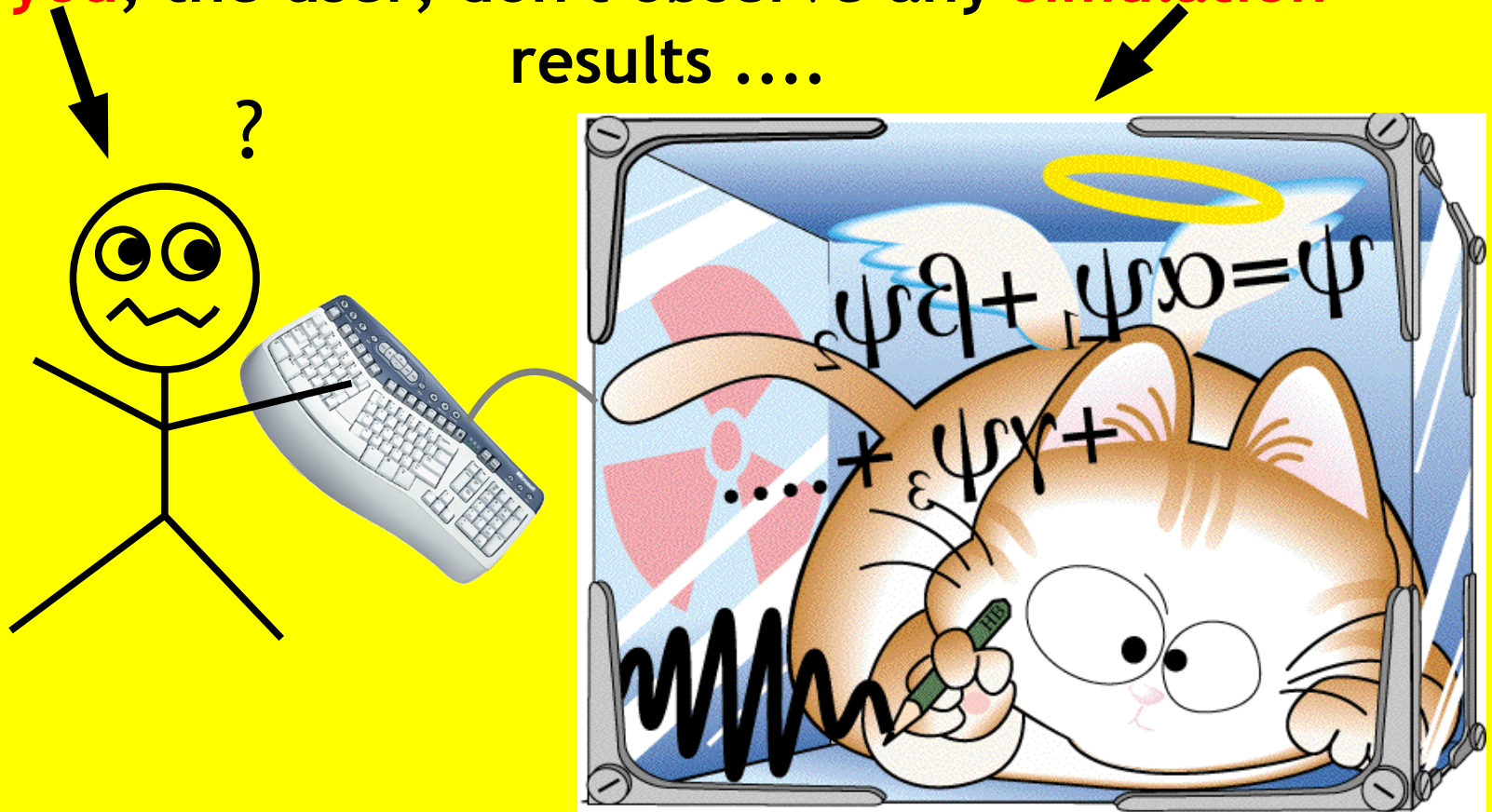
Mandatory User Code

- 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!

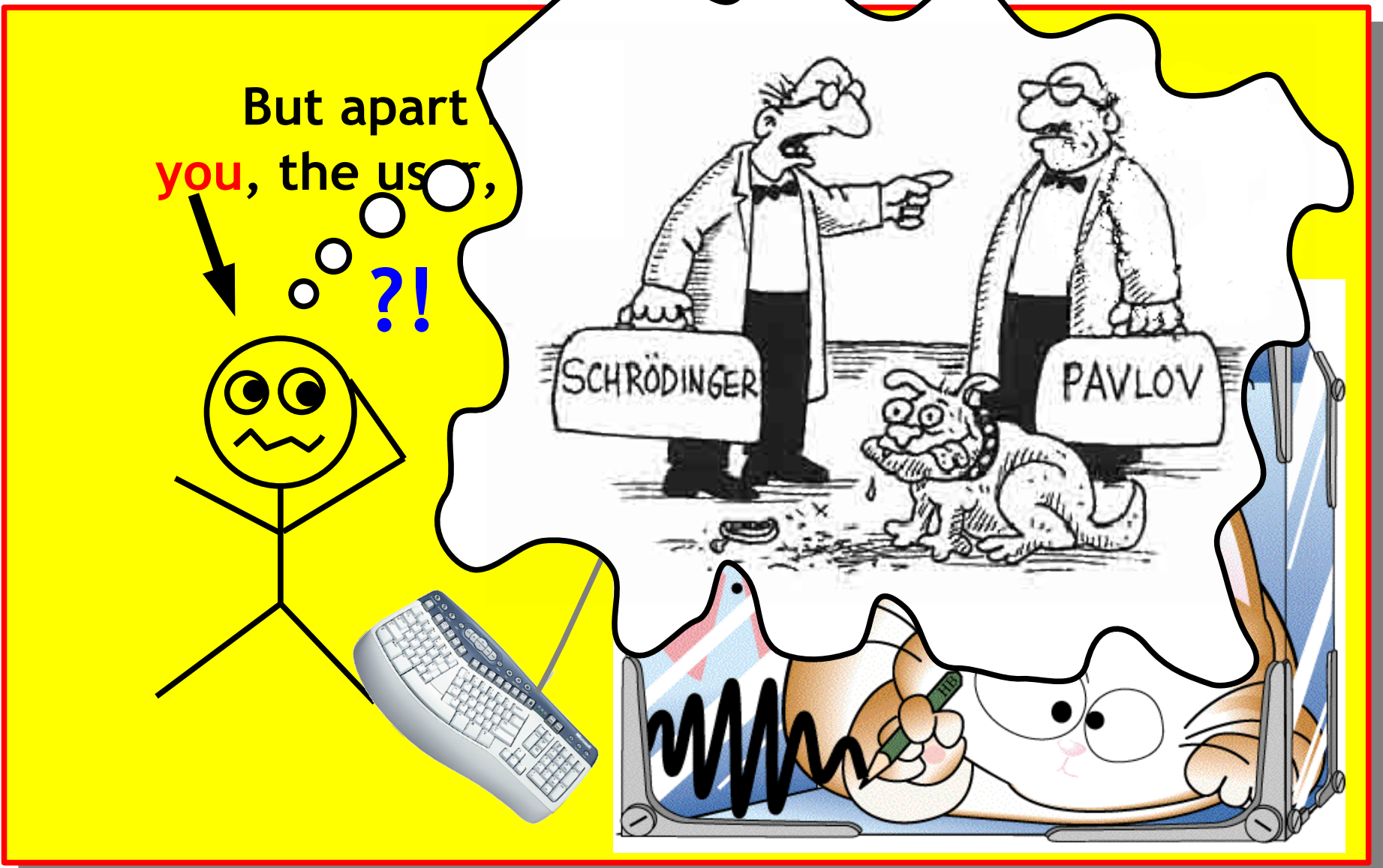
Mandatory User Code

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



a GEANTic Schrödinger cat

Mandatory User Code



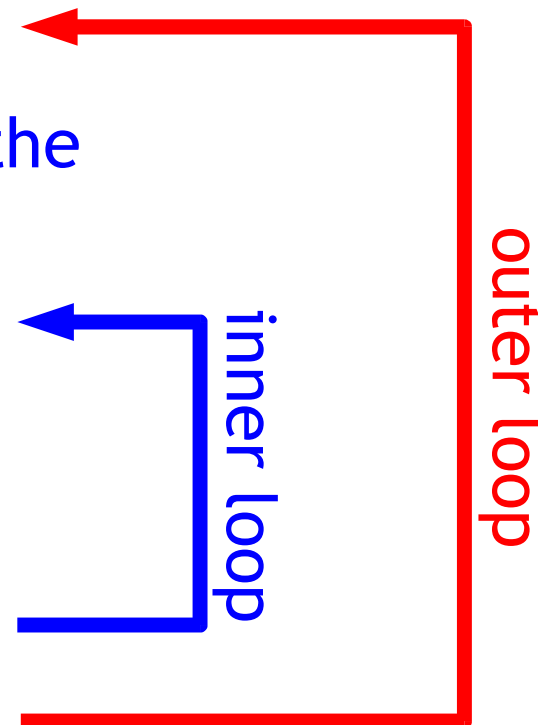
Initialization & Action Hooks

- 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

- Int

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:

- Ru

- Ev

Run-, Event-, Track-, Step-Actions,

(not treated in these lectures:

Stack-Action)

G4 Russian Onion



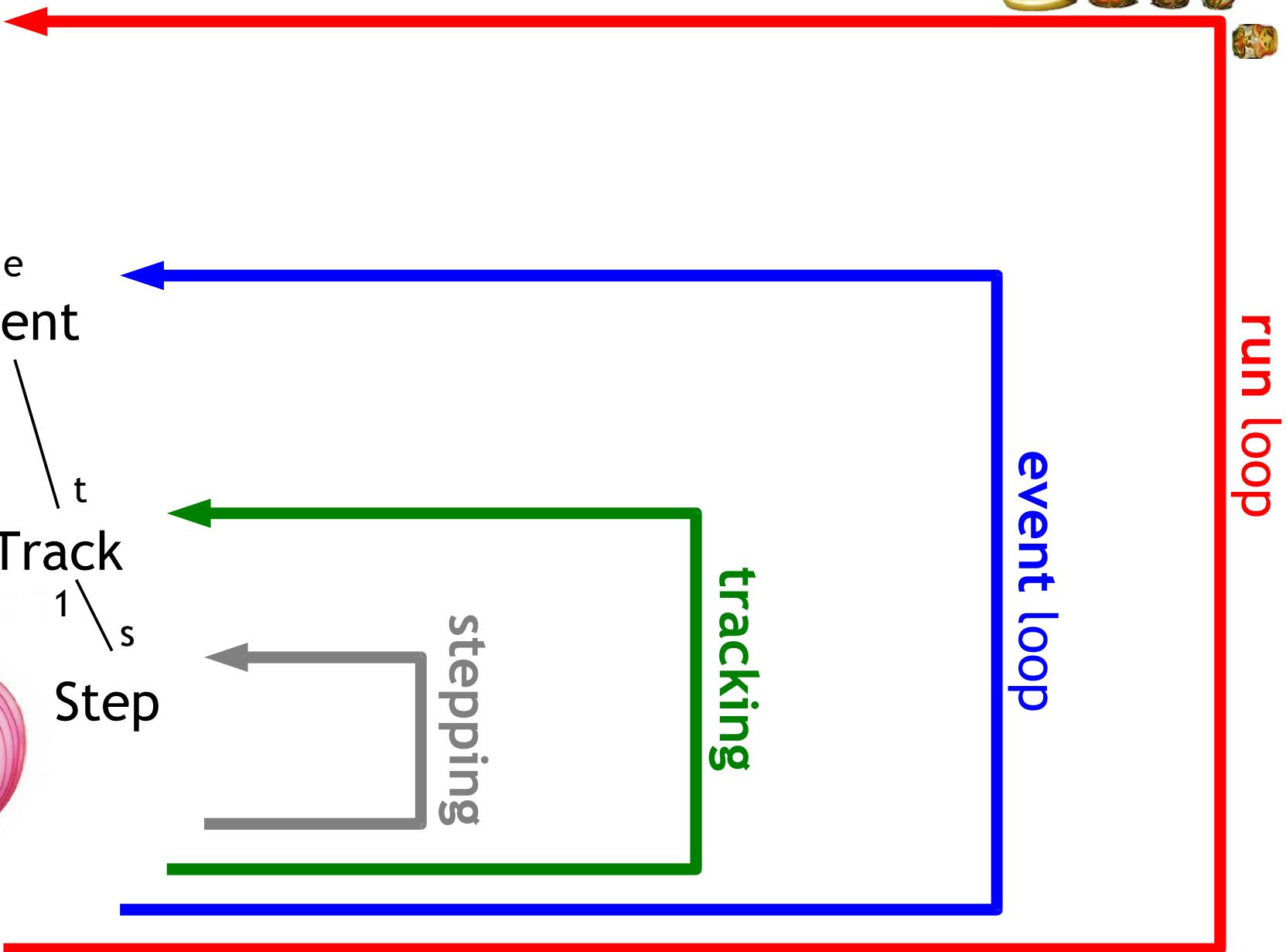
Simulation

1
|
n
Run

e
Event

1
t
Track

1
s
Step



Subclassing Actions



Simulation

1
|
n
Run
1

e
Event

1
t
Track

1
s
Step



G4VUserDetectorConstruction

```
class MyDetector  
: public G4VUserDetectorConstruction
```

before a Run starts:

- process user-parameters (UI interaction, not in lectures)
- create materials, geometry, physics processes

run loop

Subclassing Actions



Simulation



e
Event

t
Track

s
Step



G4VUserPhysicsList

```
class MyPhysics
: public G4VUserPhysicsList
```

before a Run starts:
- define the particles and physics processes including their parameters (cuts)

run loop

G4VUserPhysicsList class

Edit Hierarchy Class Xref Include Retriever G

Name	Class	Type	Parameter
ConstructParticle(md)		virtual void	()
ConstructProcess(md)		virtual void	()
~G4VUserPhysicsList(md)		virtual int	()
SetCuts(md)		virtual void	()

Instantiate the singletons
G4Electron, G4Positron, ..

For each particle, instantiate
the processes according to
which the particle should be
created, tracked, destroyed

G4VUserDetectorConstruction class Filter

Edit Hierarchy Class Xref Include Retriever Grep

Name	Class	Type	Parameter
Construct(md)		virtual G4VPhysicalVolume *	()
G4VUserDetectorConstruction(md)		int	()
~G4VUserDetectorConstruction(md)		virtual int	()

Create materials, logical and physical volumes;
add volume specifics like sensitive detectors (see later)

Subclassing Actions



Simulation



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

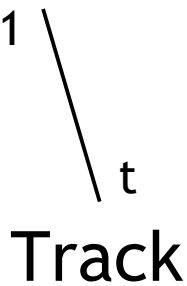
Subclassing Actions



Simulation



e
Event



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

Simulation

Subclassing

G4UserPrimary-GeneratorAction

GeneratePrimaries
(G4Event *)

```
class MyGenerator : public  
G4UserPrimary....
```

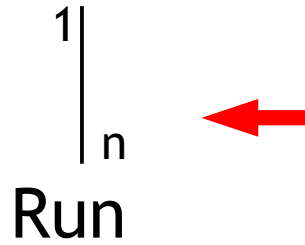
G4Event

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

G4UserEventAction

```
BeginOfEventAction(G4Event*)  
EndOfEventAction(G4Event*)
```

```
class MyEventAction  
: public G4UserEventAction
```



Event

Track

Step

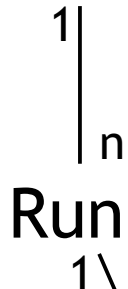


run loop event loop

Subclassing Actions



Simulation



e
Event

1
t
Track

1
s
Step



G4Track

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

G4UserTrackingAction

```
PreUserTrackingAction(G4Track*)
PostUserTrackingAction(G4Track*)
```

class MyTrackingAction

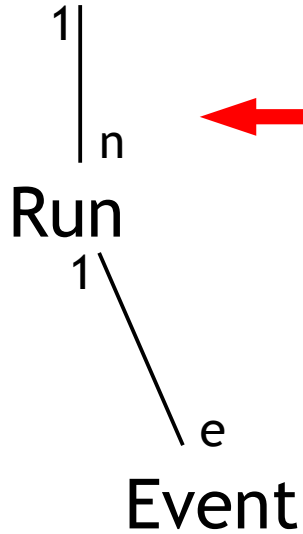
```
: public G4UserTrackingAction
```

run loop event loop tracking

Subclassing Actions



Simulation



Run

Event

Track

Step



```

classDiagram
    class G4Step {
        .. pre- and post-step-points, delta information ..
        .. kinematics information ..
    }
    class G4UserSteppingAction {
        UserSteppingAction(G4Step*)
    }
    class MyStepping {
        public G4UserSteppingAction
    }
    G4Step --> G4UserSteppingAction
    G4UserSteppingAction --> MyStepping
  
```

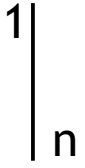
run loop event loop

stepping tracking

Typical Usage



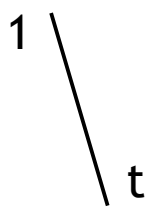
Simulation



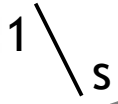
Run



Event



Track



Step

run loop

event loop

tracking

stepping

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)

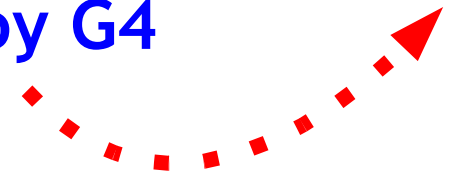
- G4RunManger registers actions to other managers
 - Event, Tracking, SteppingManagers, ..
 - many are singletons, i.e. kind of registries,
 - Managers are responsible for the steering of a specific simulation-layer
 - usually not visible to the user
- G4RunManager::beamOn(G4int n)
 - triggers a Run
 - n Events are simulated by invoking the G4EventManager
 - 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 collaboration does this, for example
- **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**

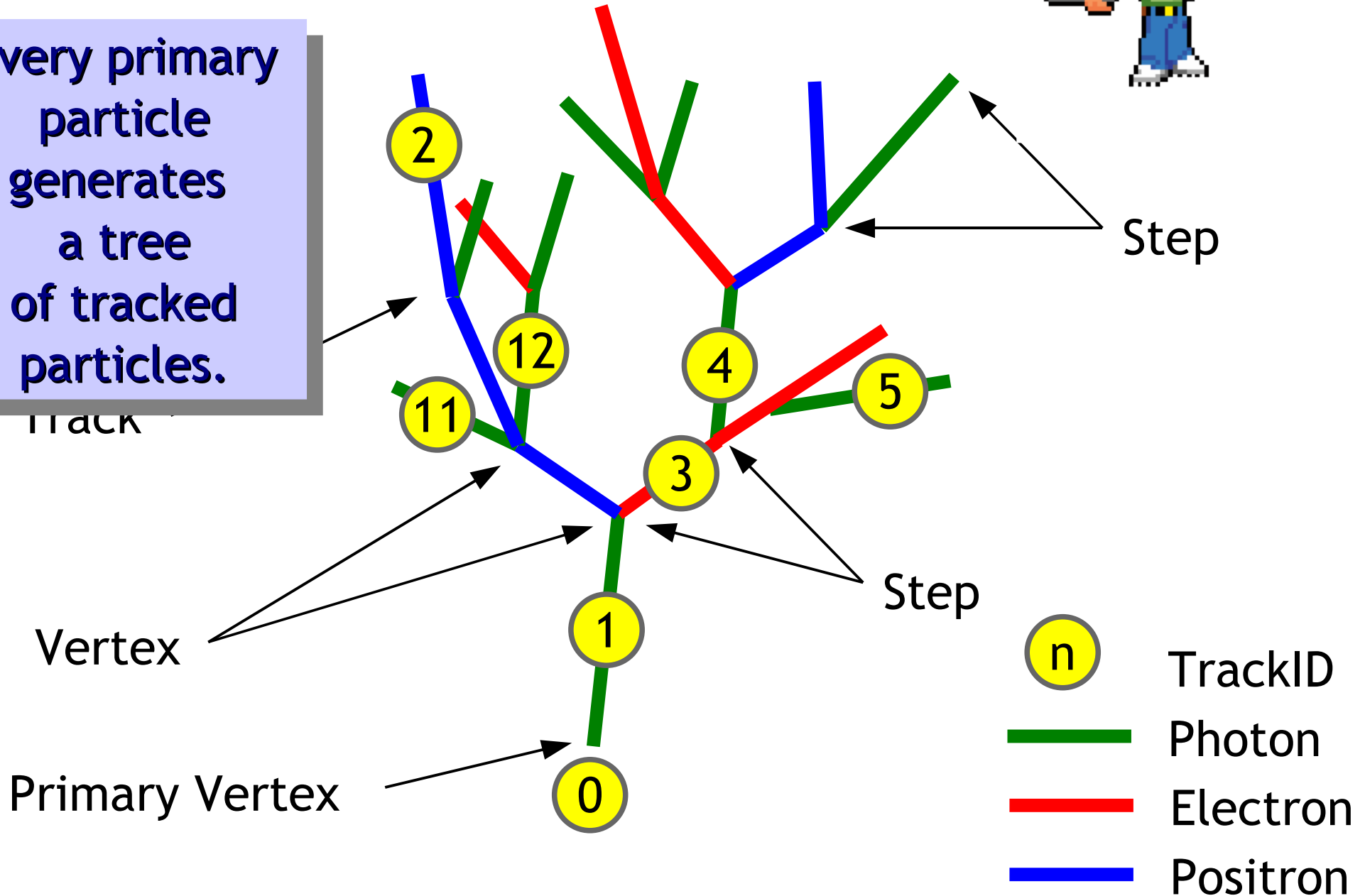


And Action!

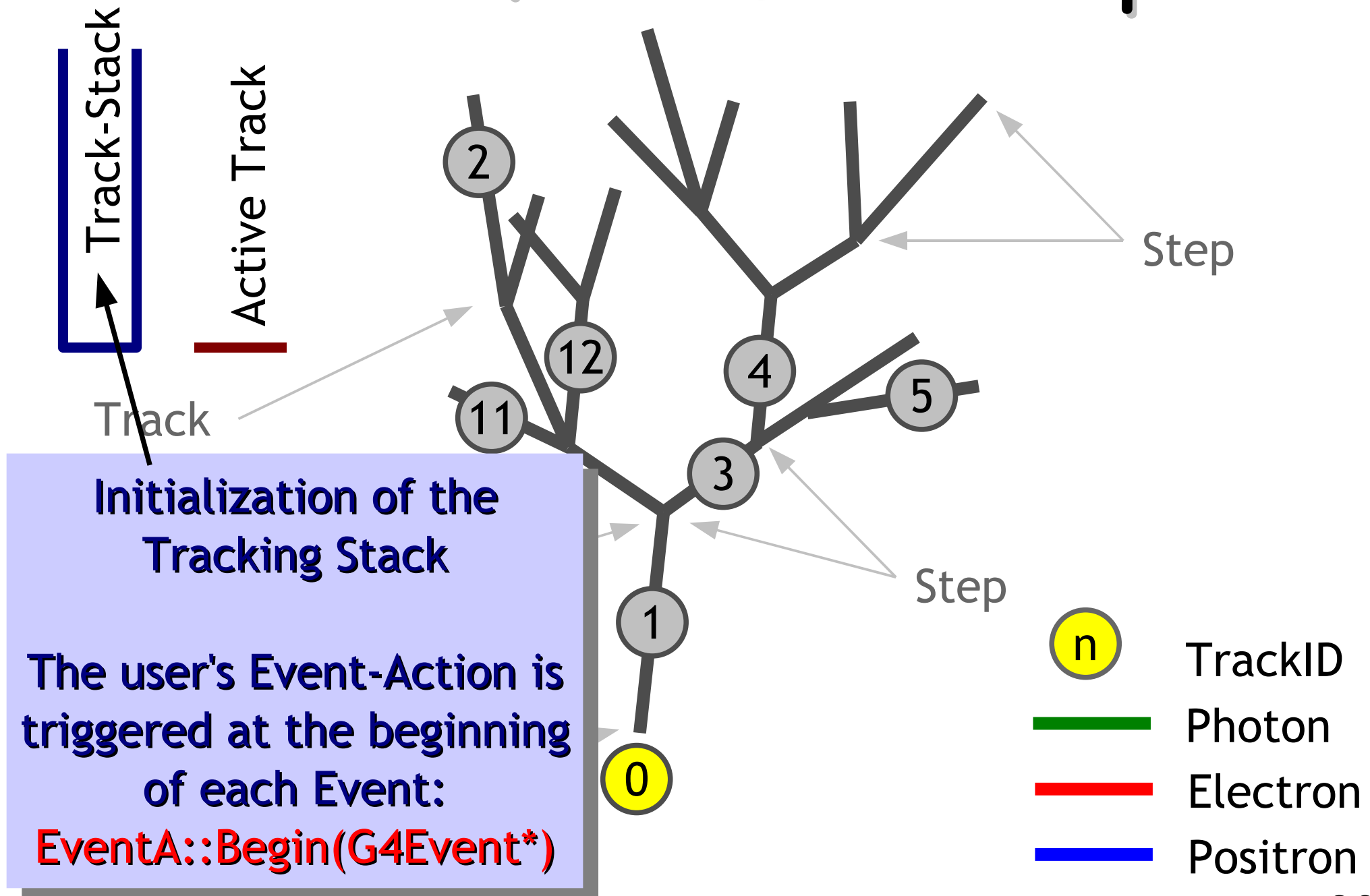


Every primary particle generates a tree of tracked particles.

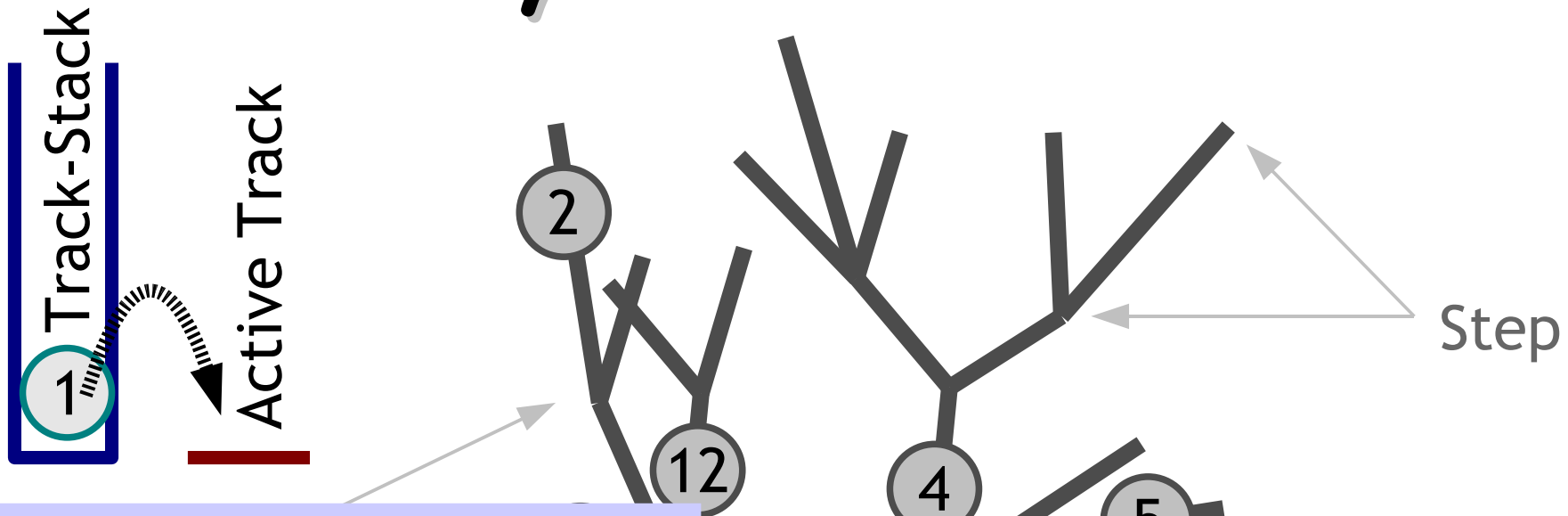
Track



Start of the Event-Loop

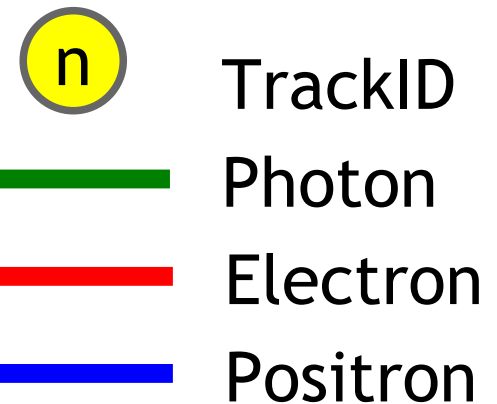


Primary Generator Action

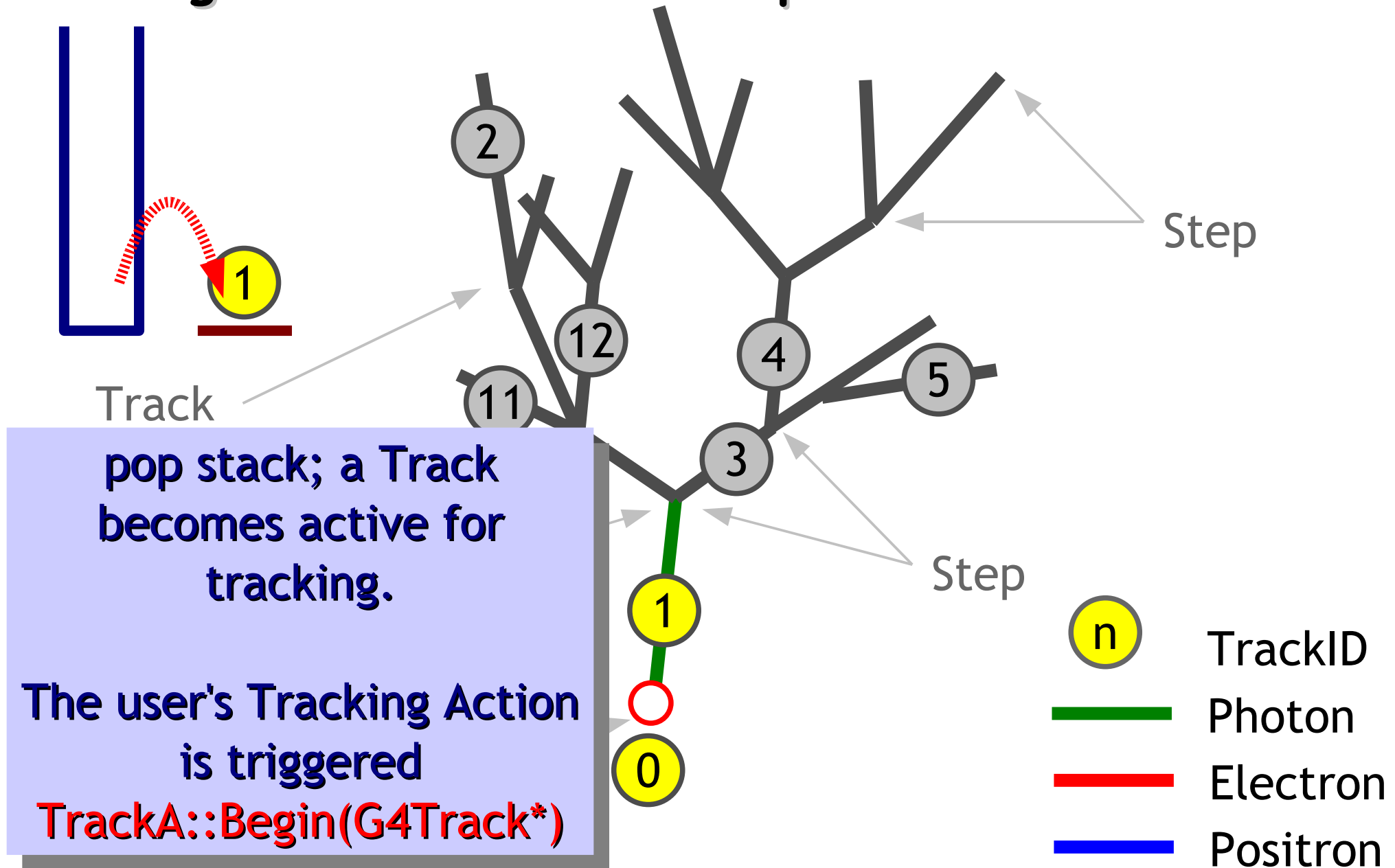


The user's Primary Event Action is triggered to generate the primary particles (only one in this example).

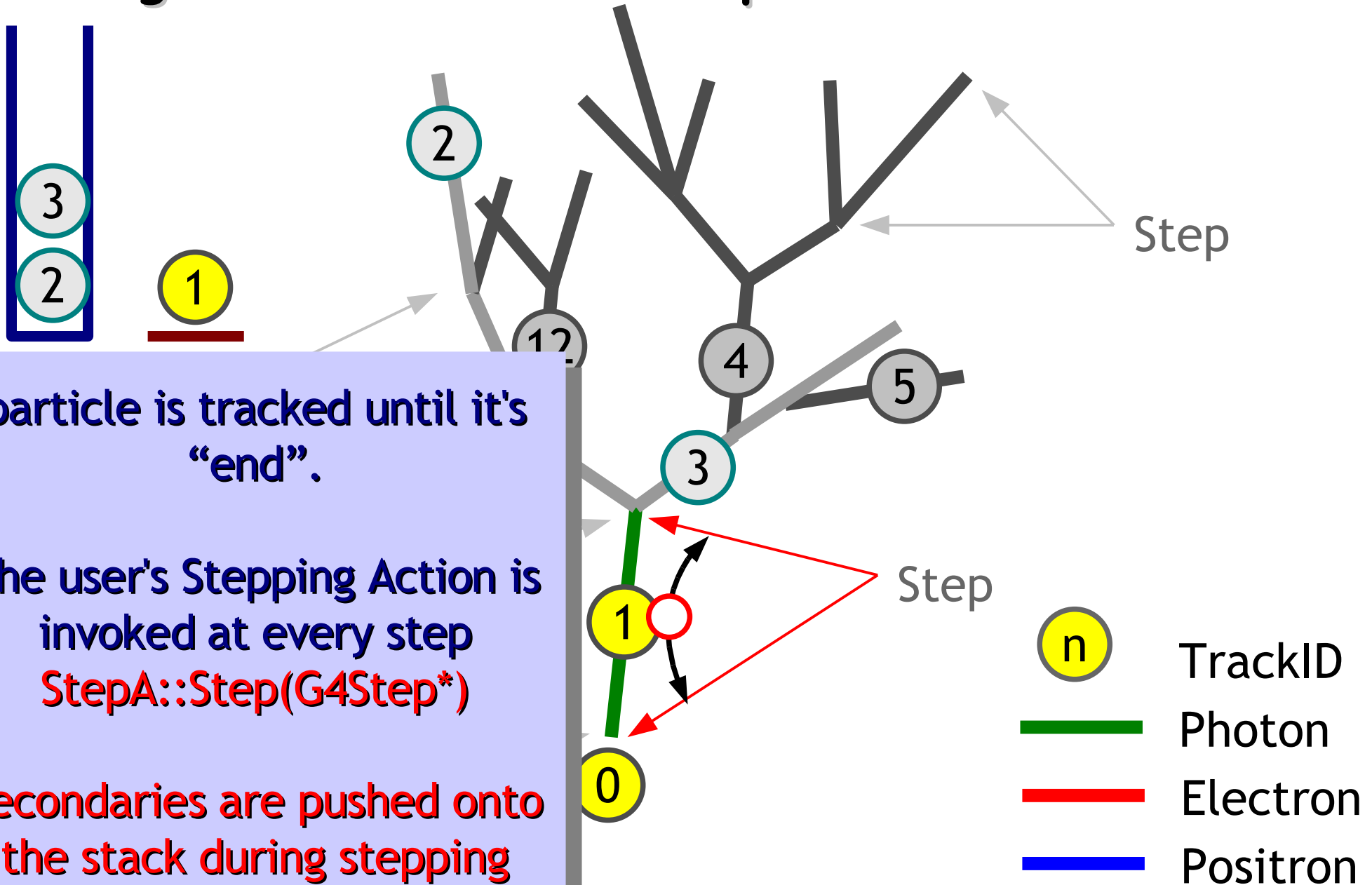
Primaries are pushed onto the stack (FILO) to wait for tracking



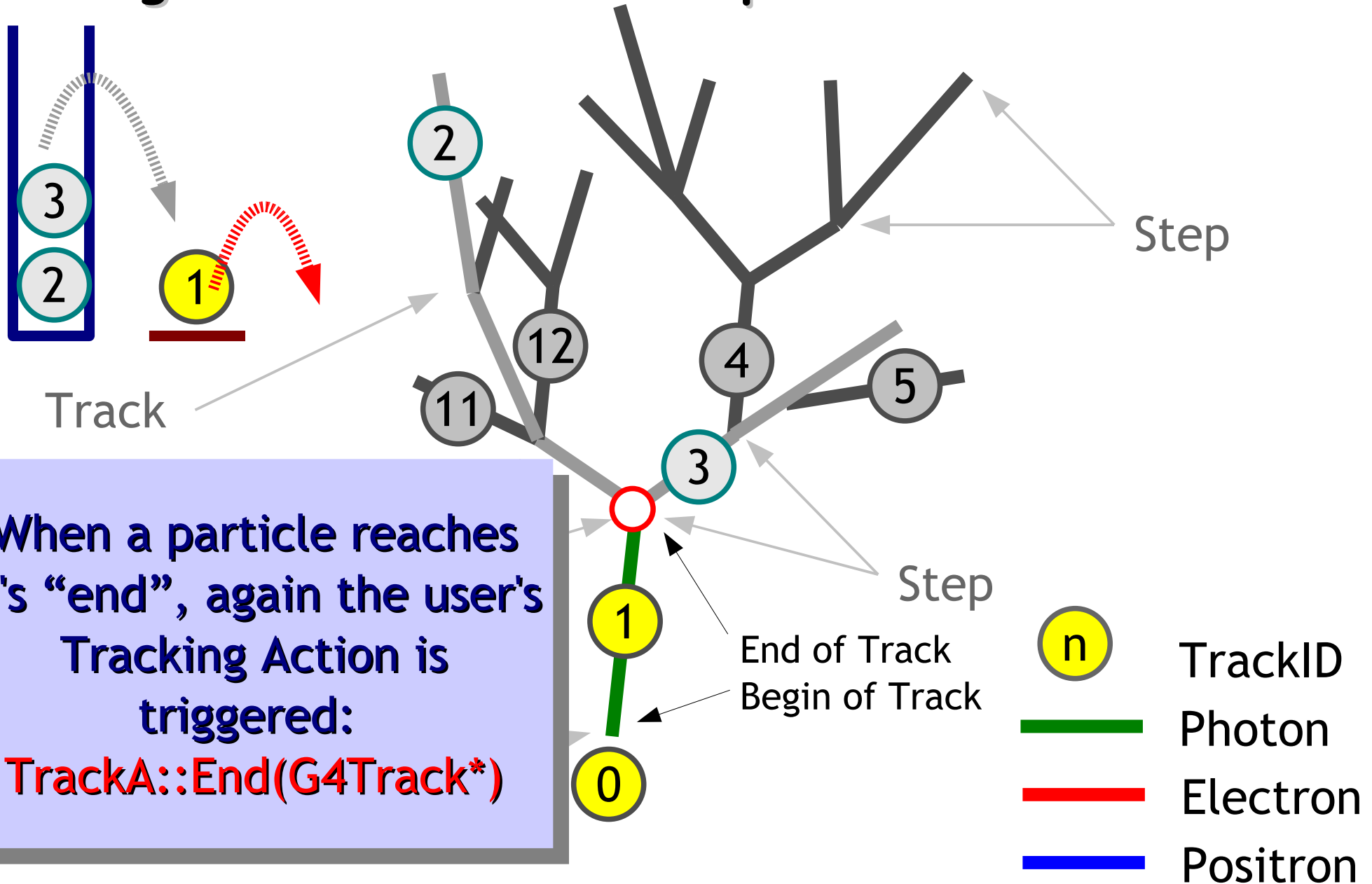
Pop Stack, Tracking (Stepping) until end of Track, generated Secondaries pushed on Stack



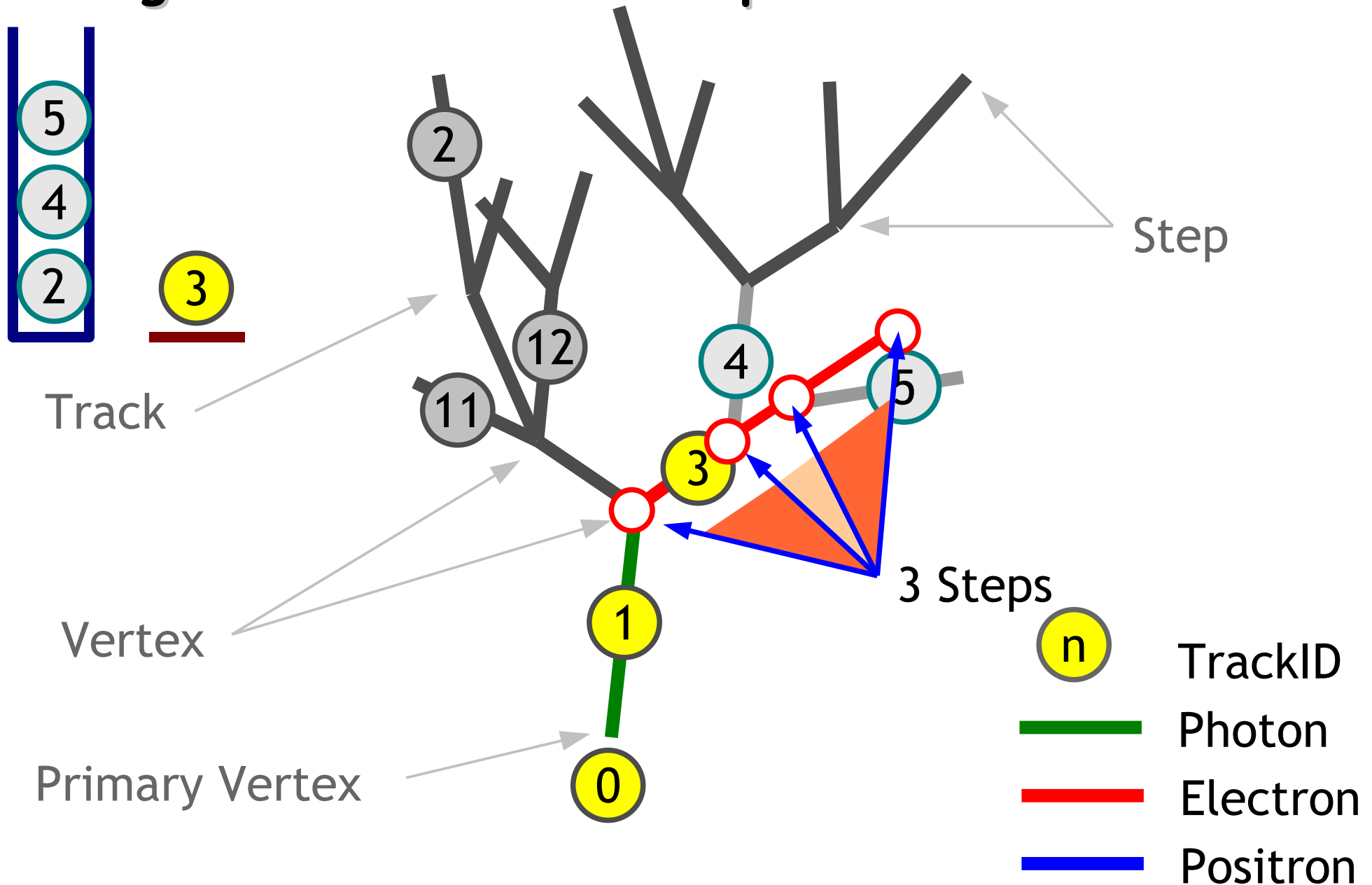
Pop Stack, Tracking (Stepping) until end of Track, generated Secondaries pushed on Stack



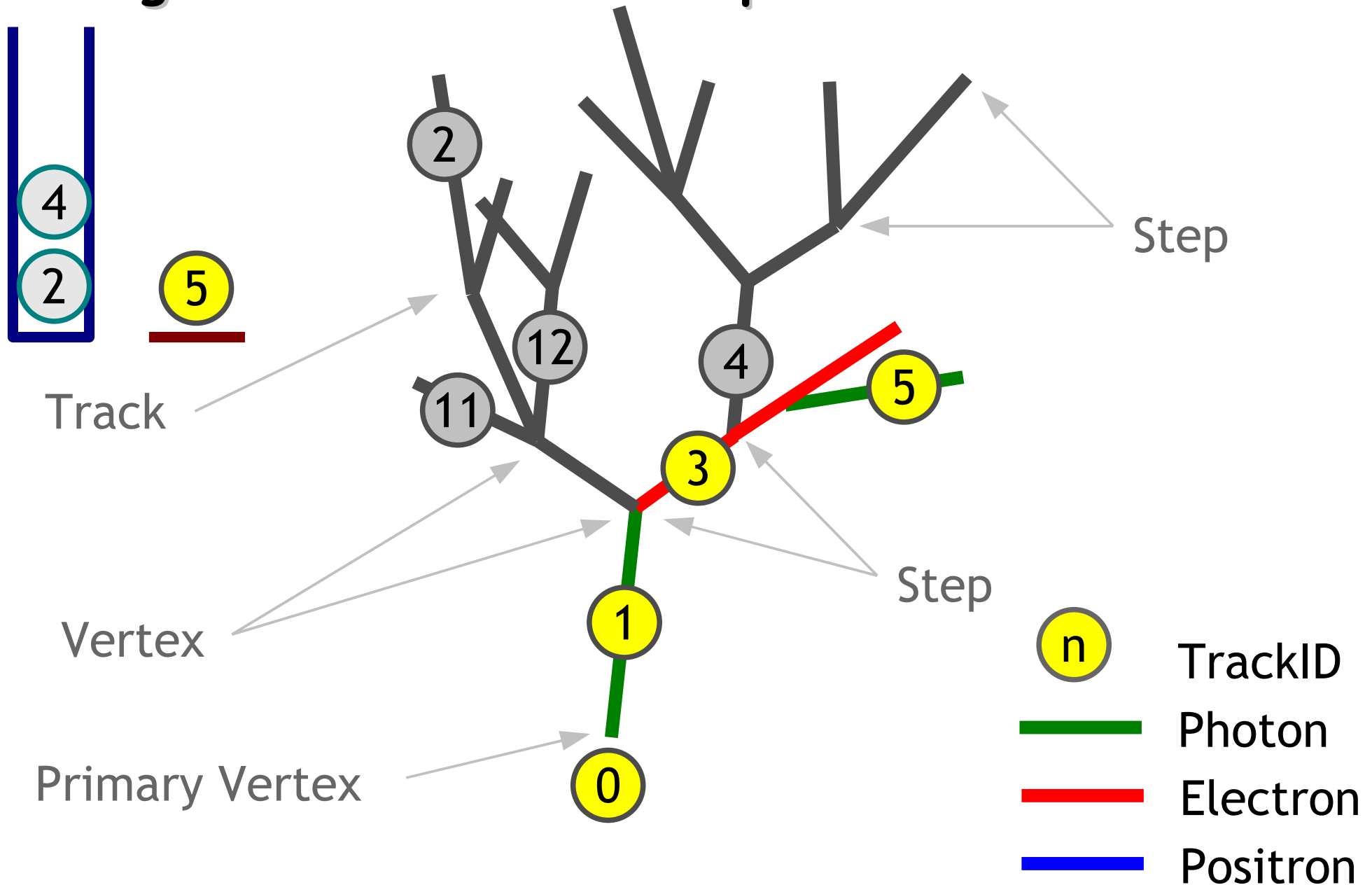
Pop Stack, Tracking (Stepping) until end of Track, generated Secondaries pushed on Stack



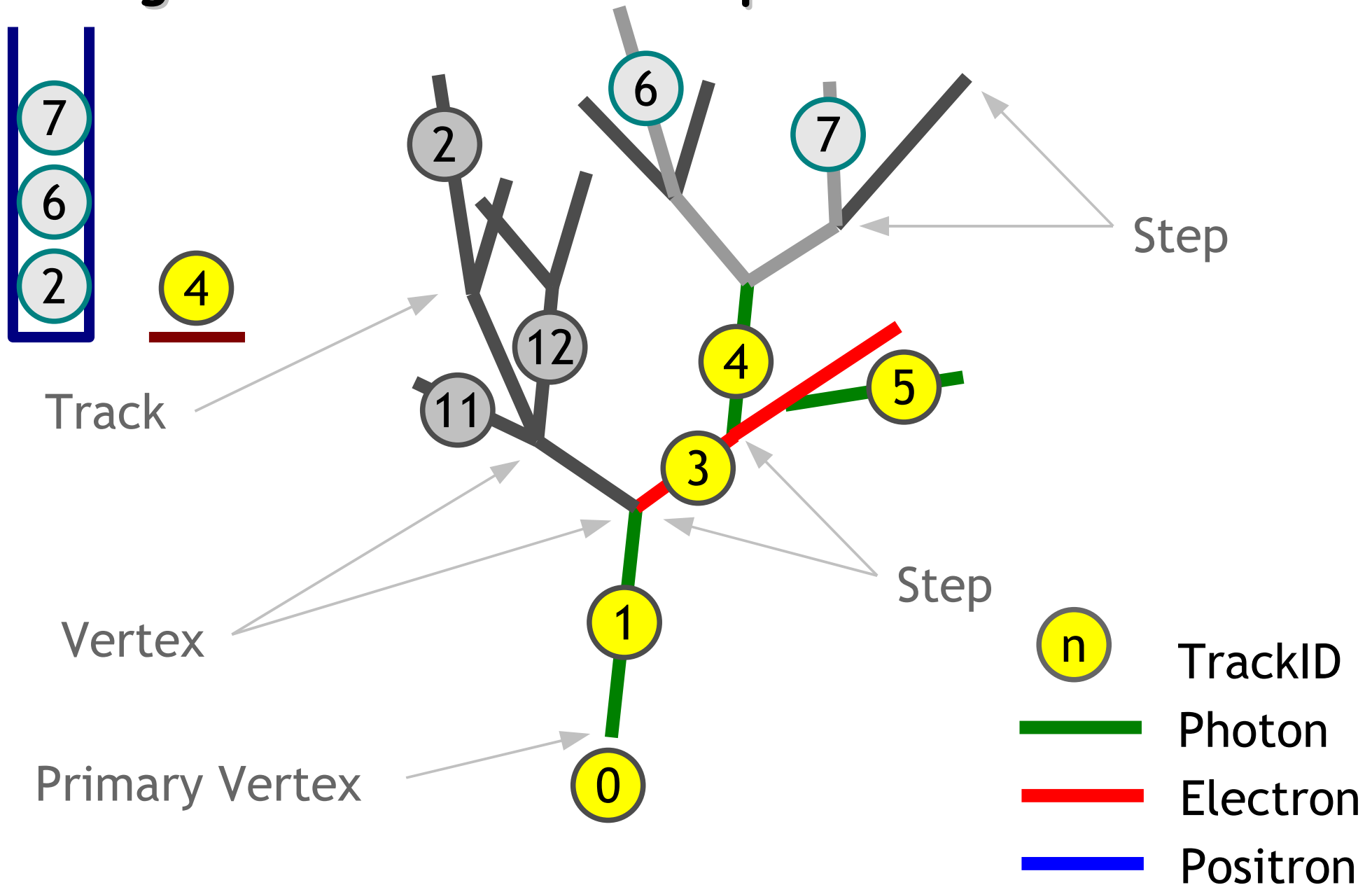
Pop Stack, Tracking (Stepping) until end of Track, generated Secondaries pushed on Stack



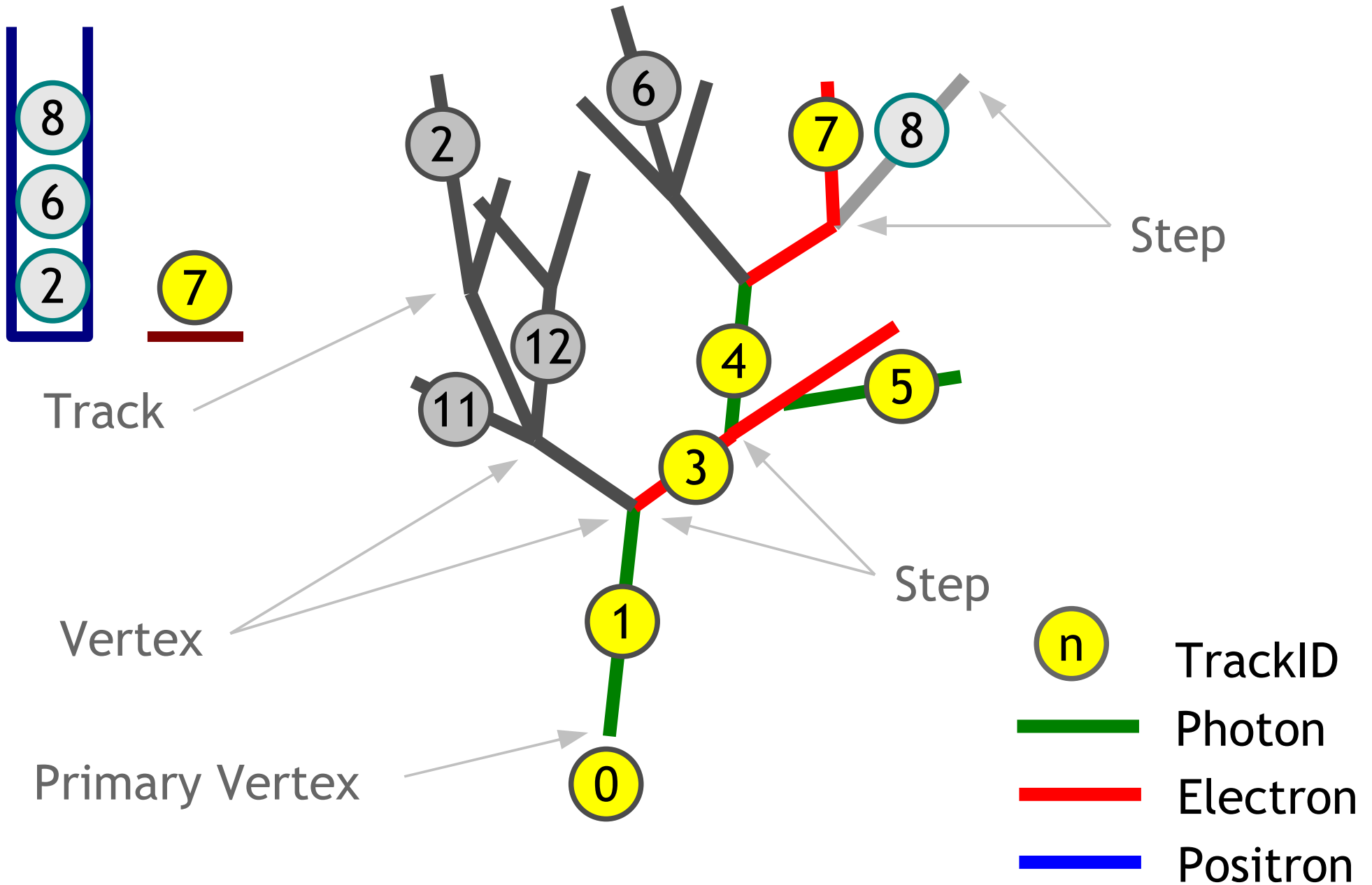
Pop Stack, Tracking (Stepping) until end of Track, generated Secondaries pushed on Stack



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



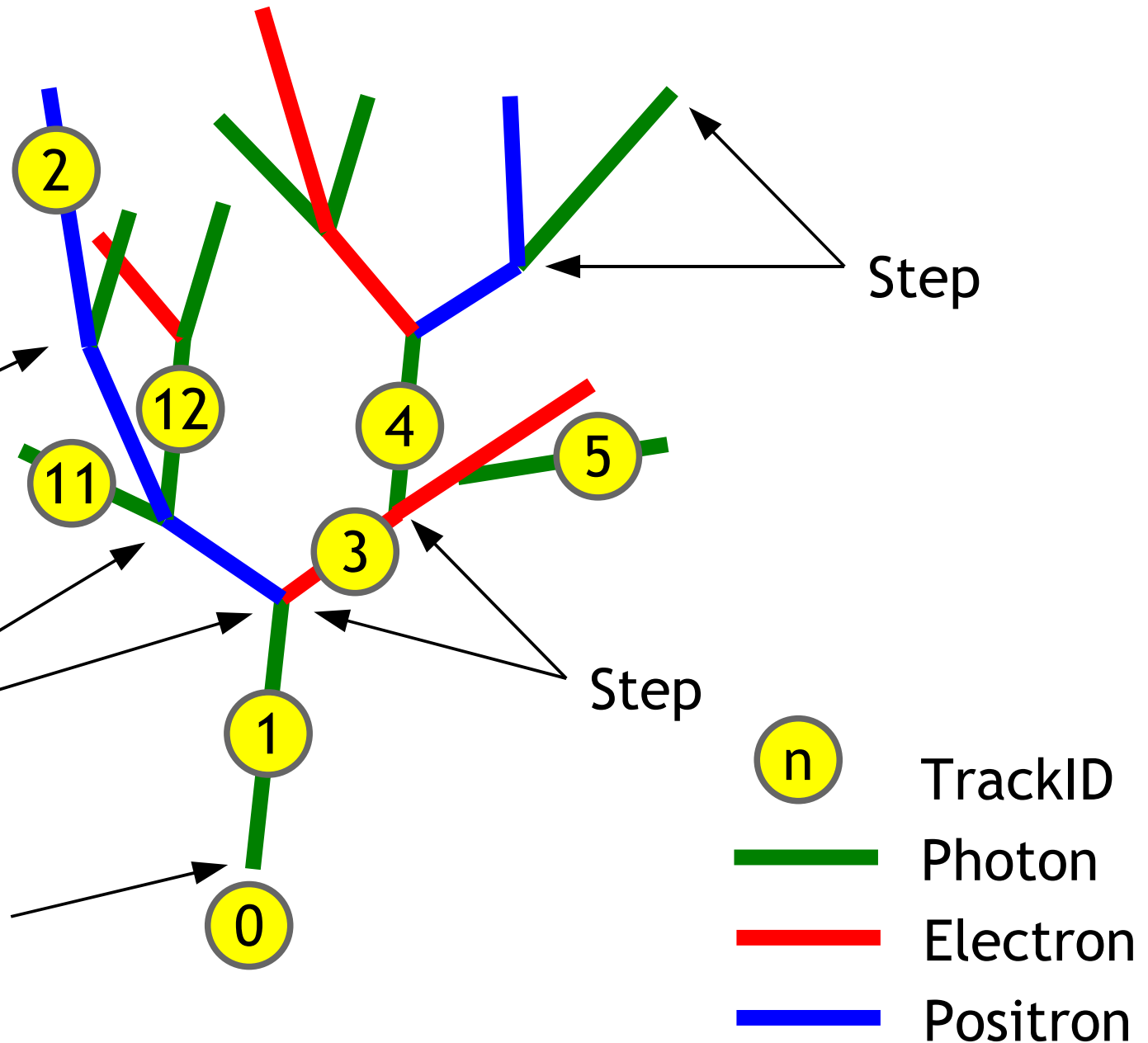
"Depth First" Tree Traversal



Track/Step (once more)

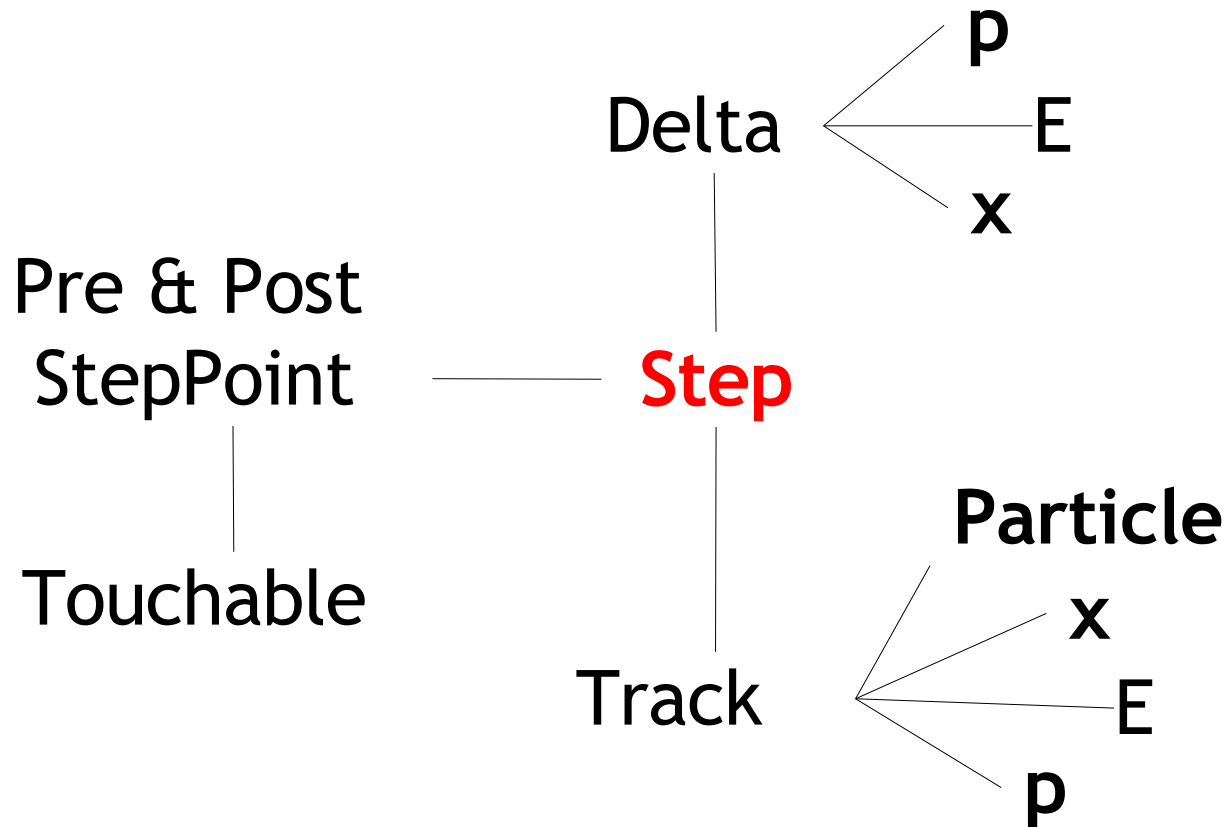
Every primary particle generates a tree of tracked particles, which is simulated in depth first order.

Primary Vertex



Stepping Action

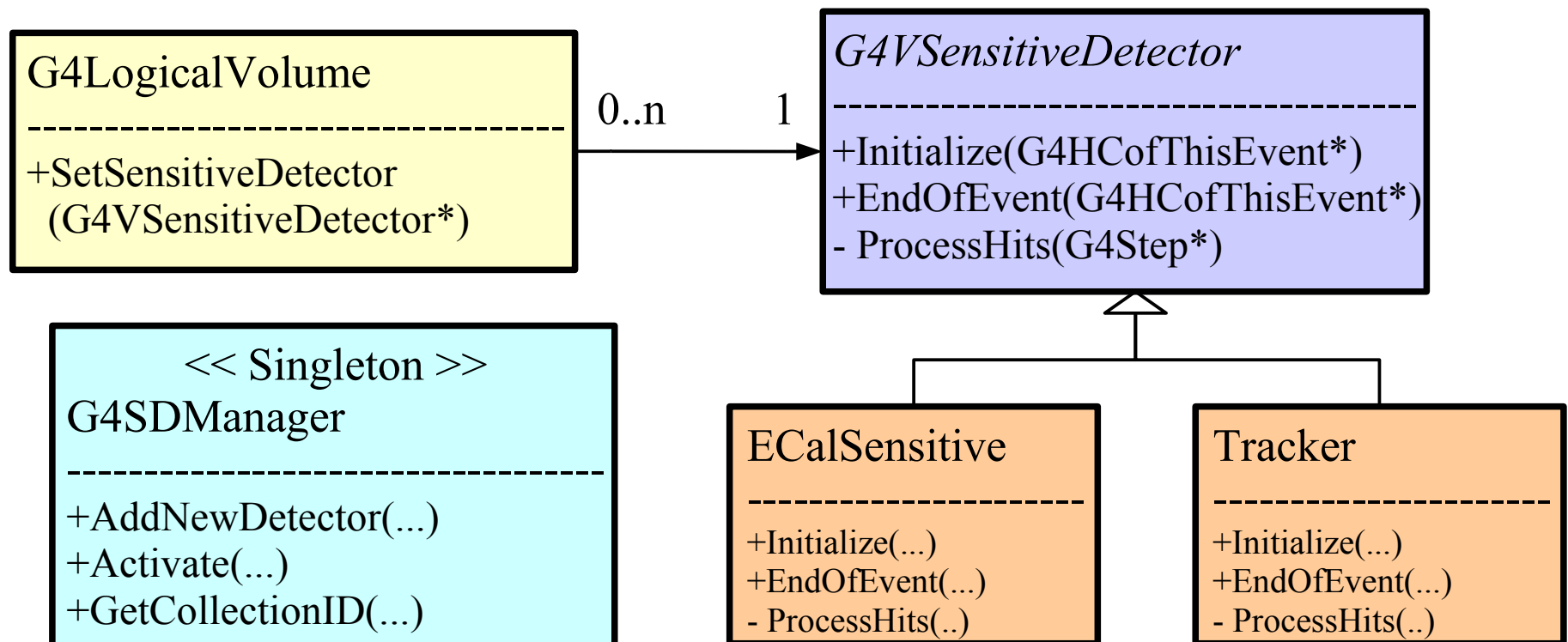
- G4VSteppingAction: finest granularity of obtaining information
 - register a sub-class to the Geant4 kernel
 - at every step:
G4VSteppingAction::UserSteppingAction(G4Step * info)



- 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

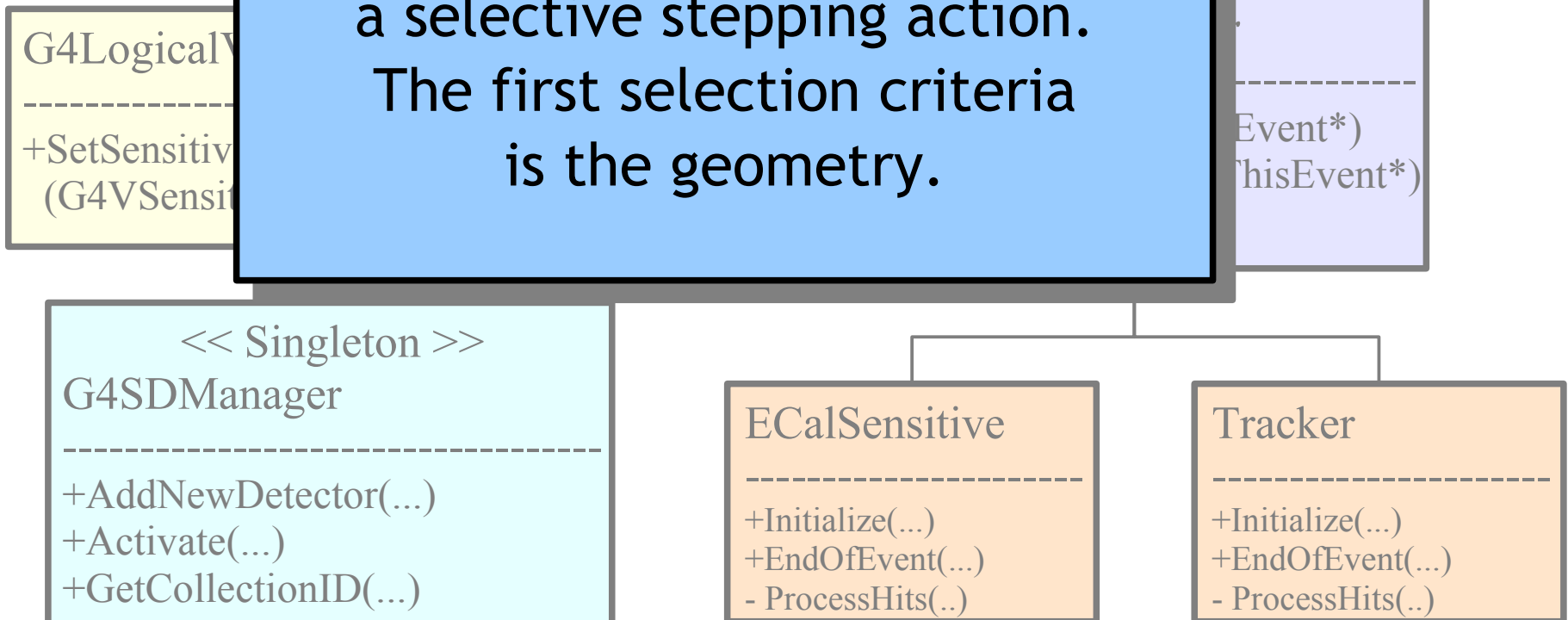
- Main motivation: to get information from the parts of the detector that will do the actual measurement (e.g. silicon pixel detectors in the tracker, crystals in the ECal, drifttubes in the Muon-system, ...)
 - these parts of the detector are called “Sensitive Detector”
 - Sensitiveness is handled via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



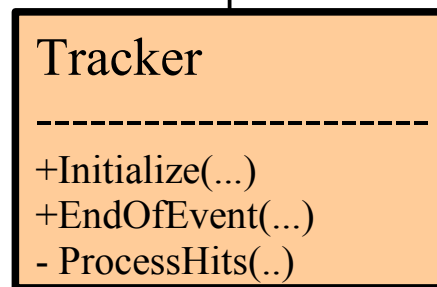
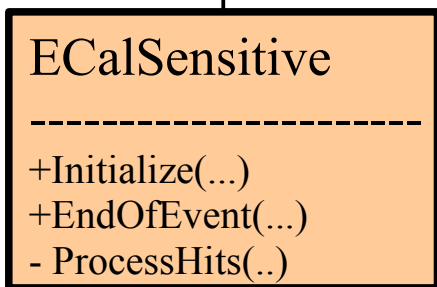
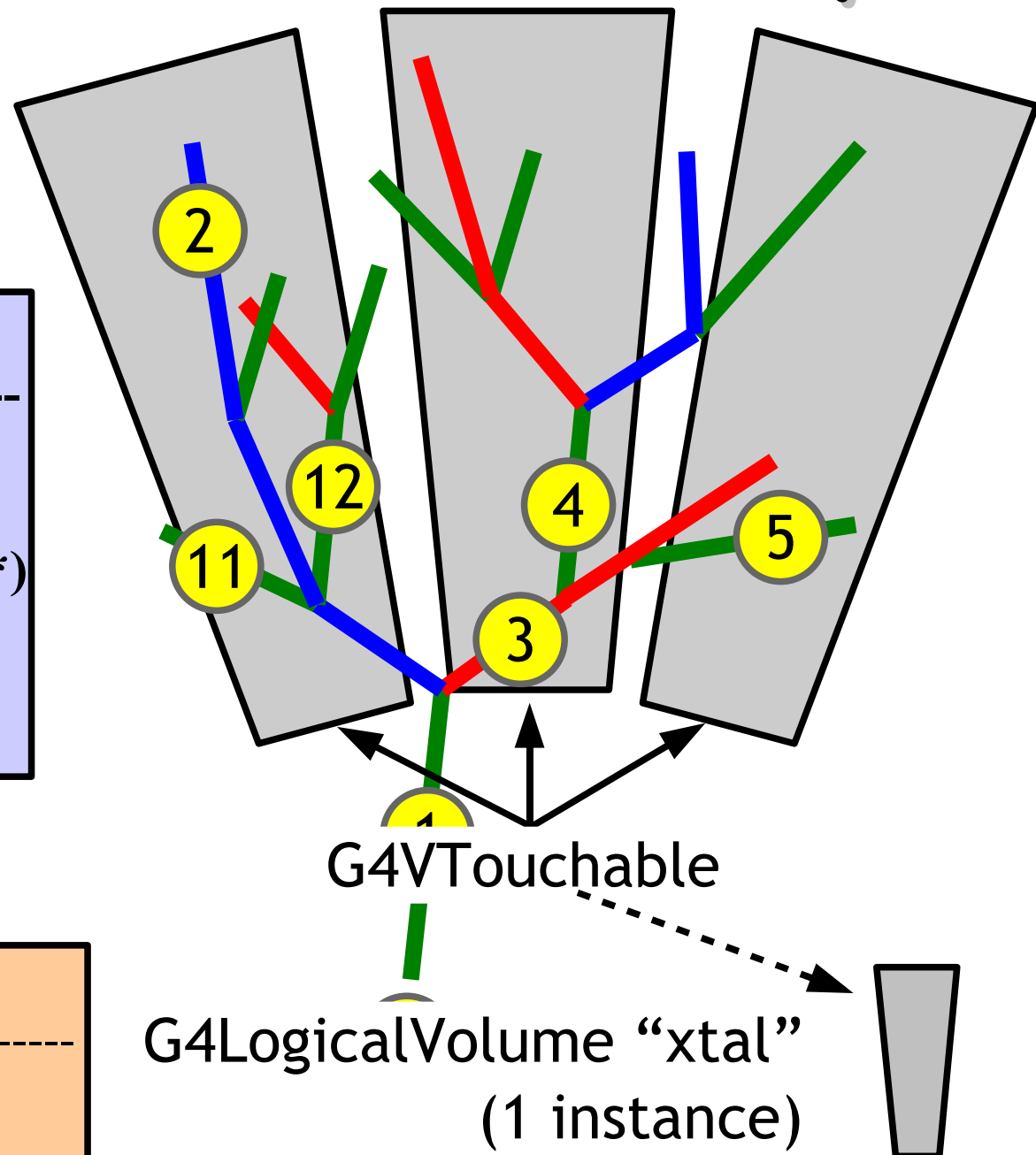
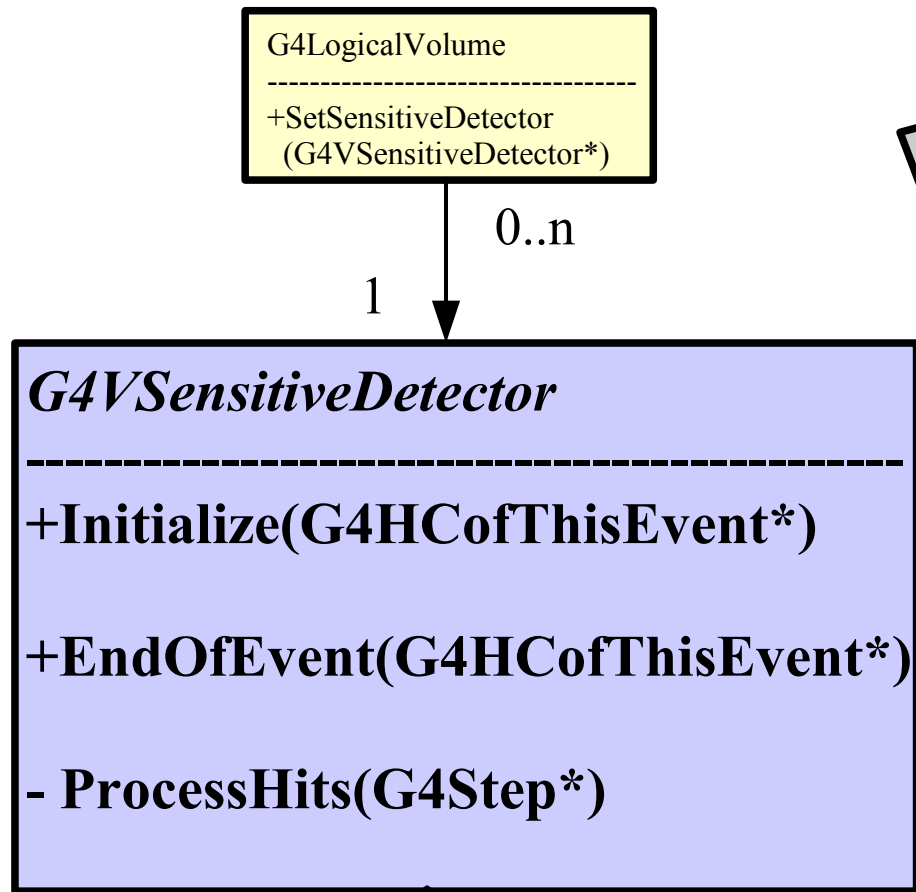
Sensitive Detector

- 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”
- Sensitive

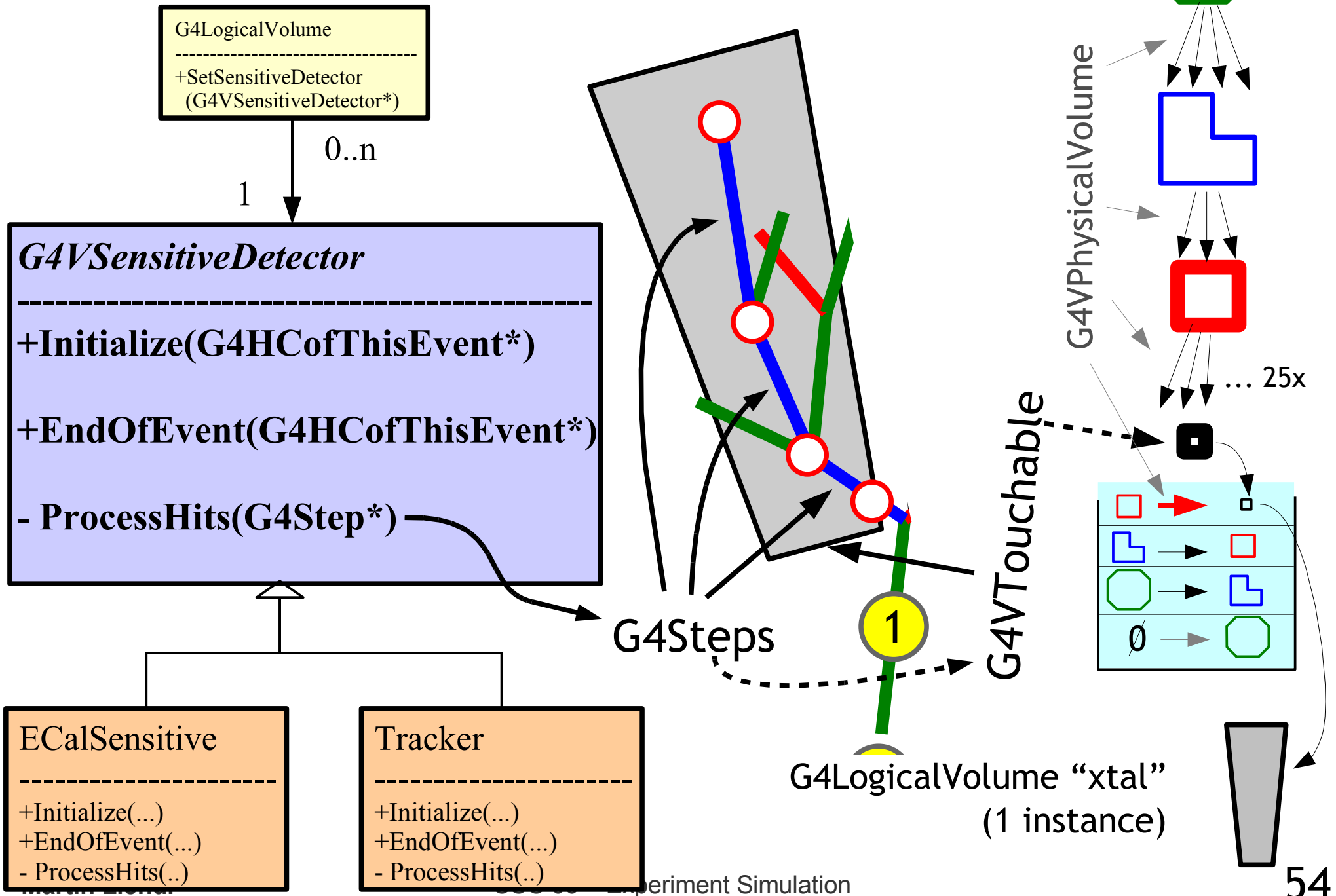
A sensitive detector is basically a selective stepping action. The first selection criteria is the geometry.



Sensitive Detector & Geometry



Sensitive Detector



Sensitive Detector

Every step in a G4Touchable with an underlying G4LogicalVolume pointing to a G4VSensitiveDetector triggers the ProcessHits() method.

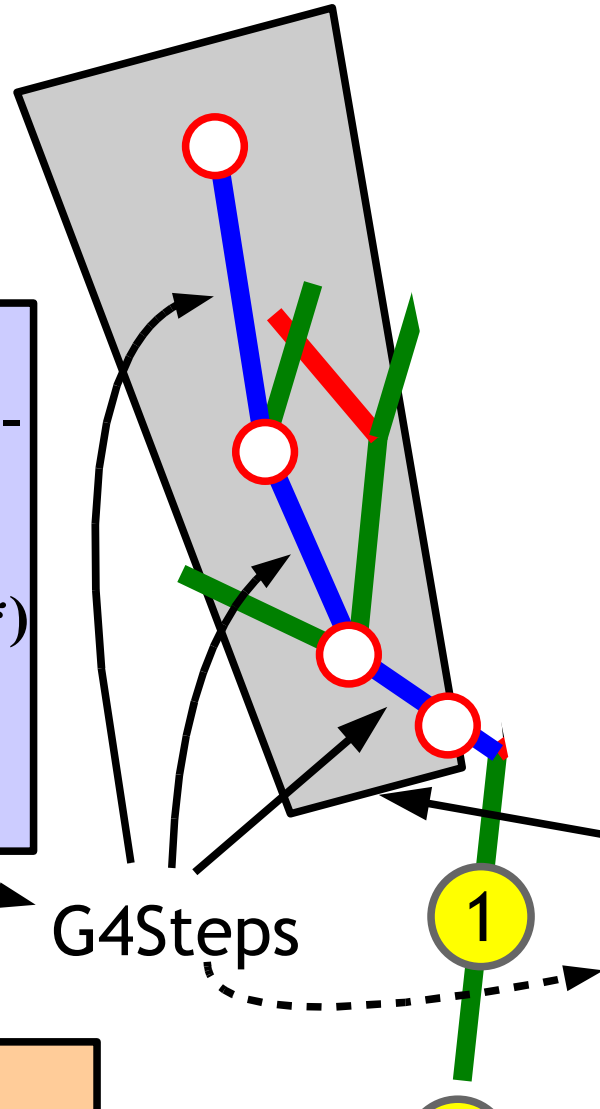
- ProcessHits(G4Step*)

ECalSensitive

 +Initialize(...)
 +EndOfEvent(...)
 - ProcessHits(..)

Tracker

 +Initialize(...)
 +EndOfEvent(...)
 - ProcessHits(..)

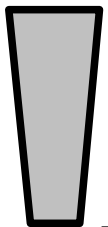
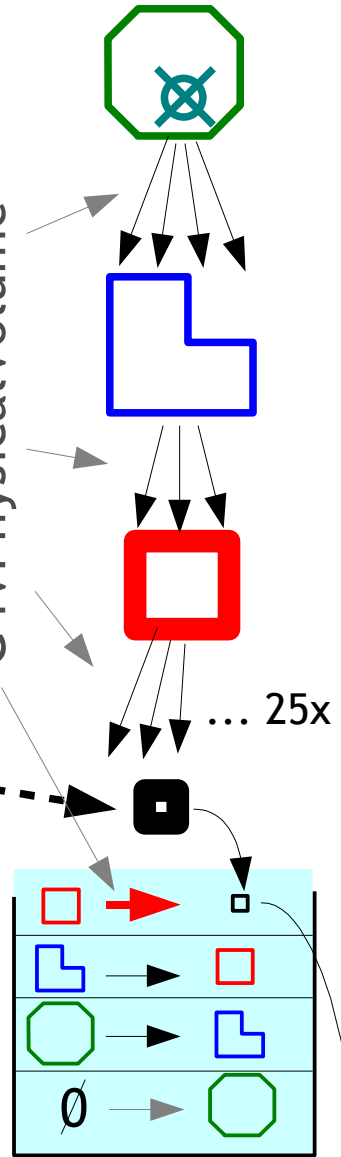


G4Steps

G4Touchable

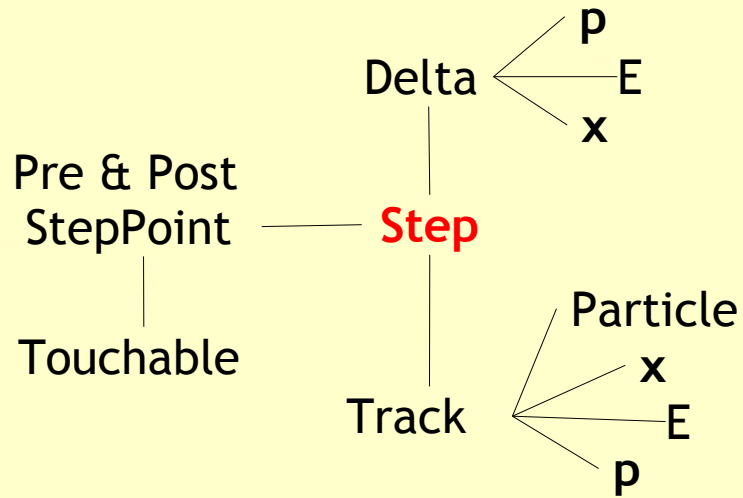
G4LogicalVolume "xtal"
 (1 instance)

G4PhysicalVolume



Sensitive Detector

The G4Step offers access to:



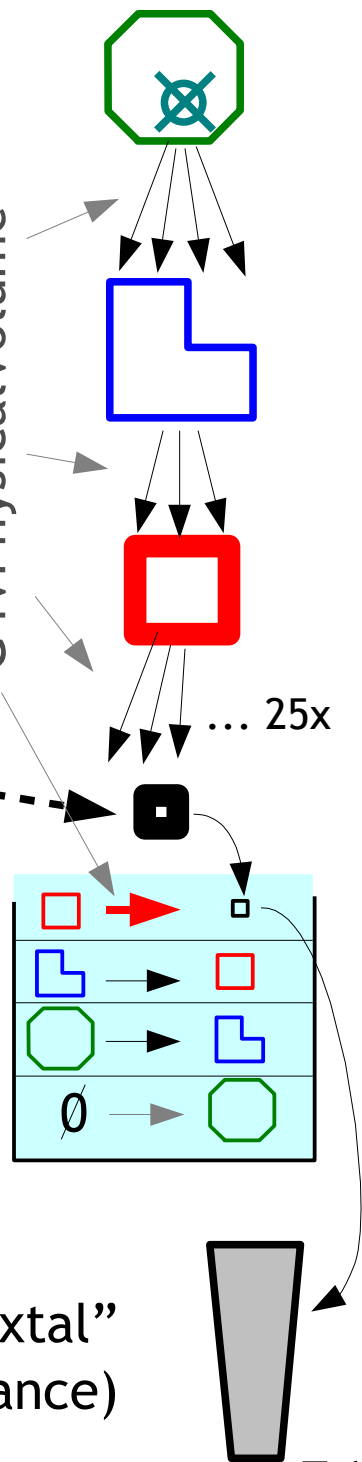
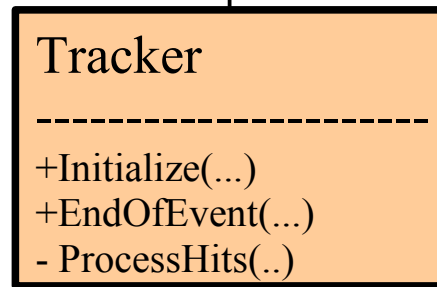
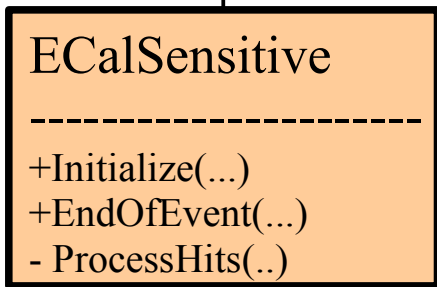
- ProcessHits(G4Step*)

G4Steps

G4Touchable

G4PhysicalVolume

G4LogicalVolume "xtal"
(1 instance)

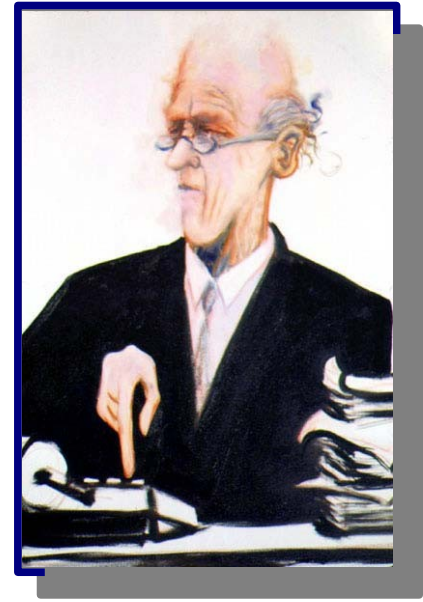


THE Simulation FAQ

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.



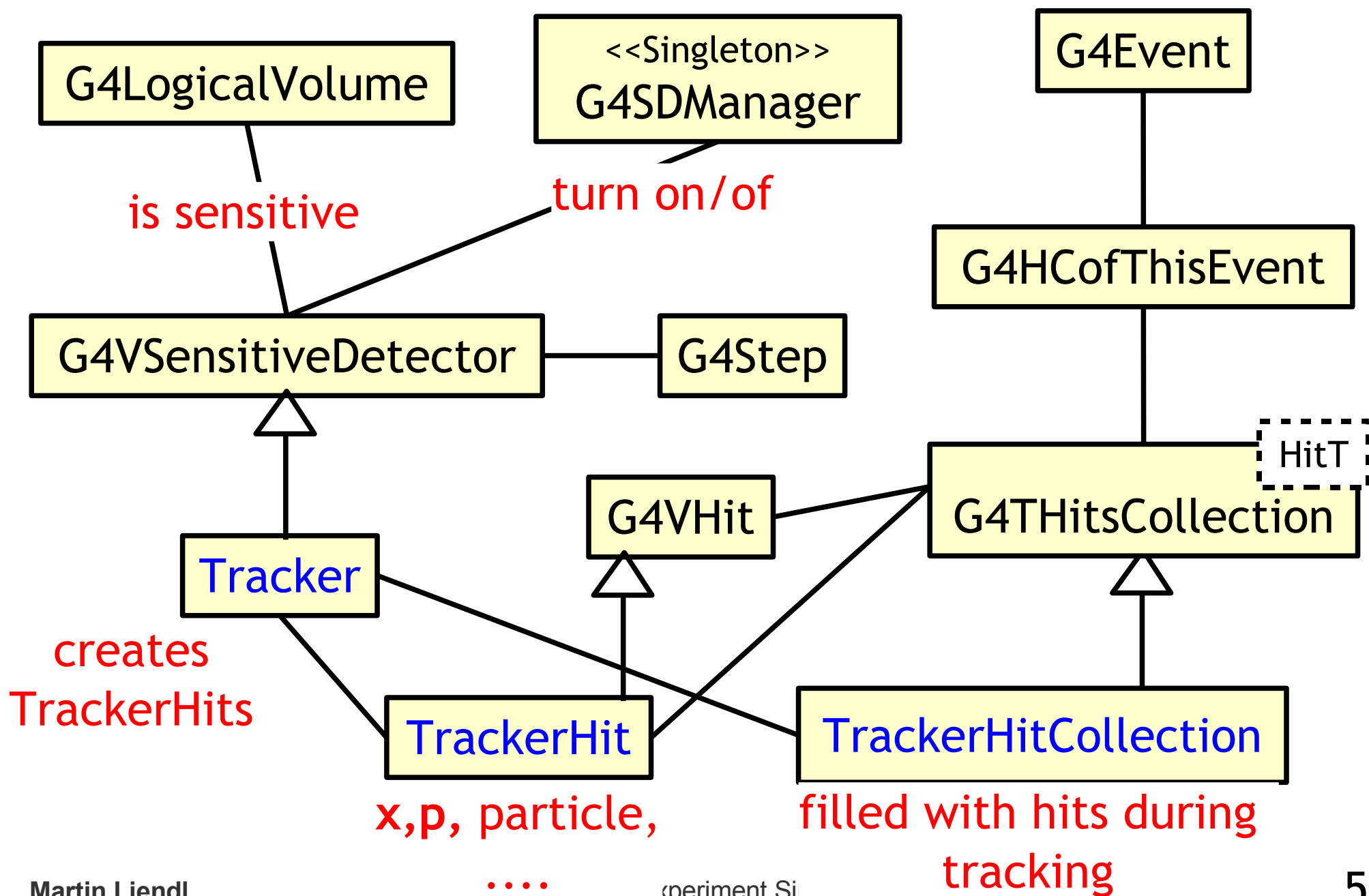
Robert D. Williams
“The Accountant”
oil on canvas, 1997

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.

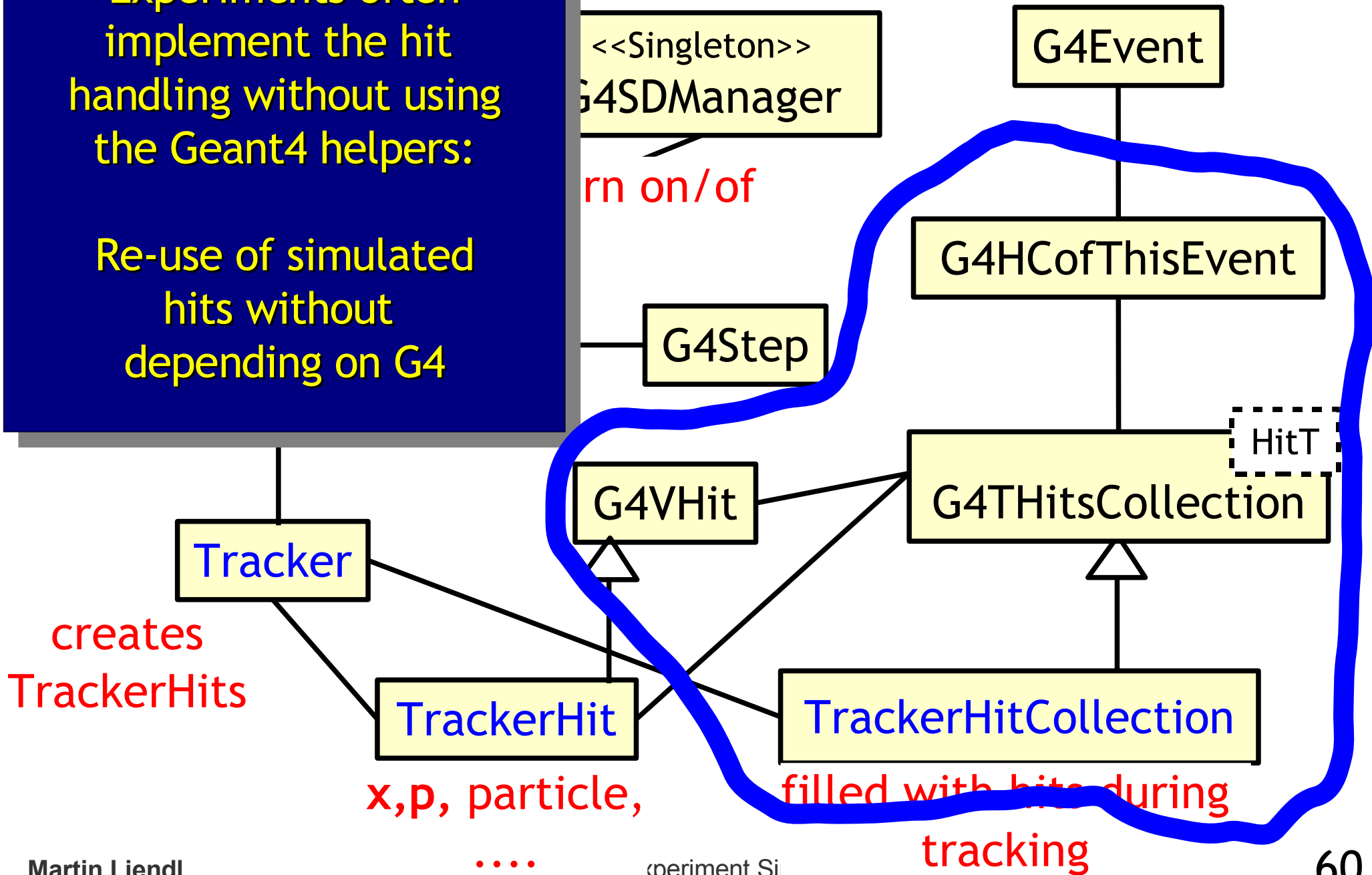
Overview of G4 Hits & Collections



Geant4 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 or reused by Geant4 as soon as they are not needed any more 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 signal 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

G4VDigitizerModule

G4VDigi

G4TDigiCollection<T>

G4DCofThisEvent

G4SDManager

G4VSensitiveDetector

G4VHit

G4THitCollection<T>

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

G4VDigitizerModule

G4VSensitiveDetector

G4VDigi

G4TDigi

G4DCo

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

e.g. the CMS collaboration does this!

Difference

G4VDigitiz

(G4VSensitive

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

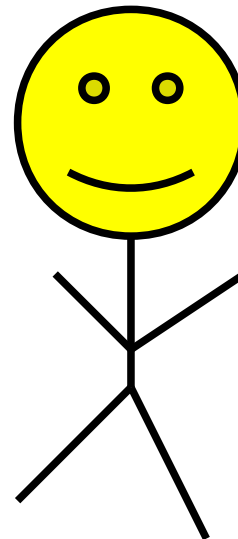
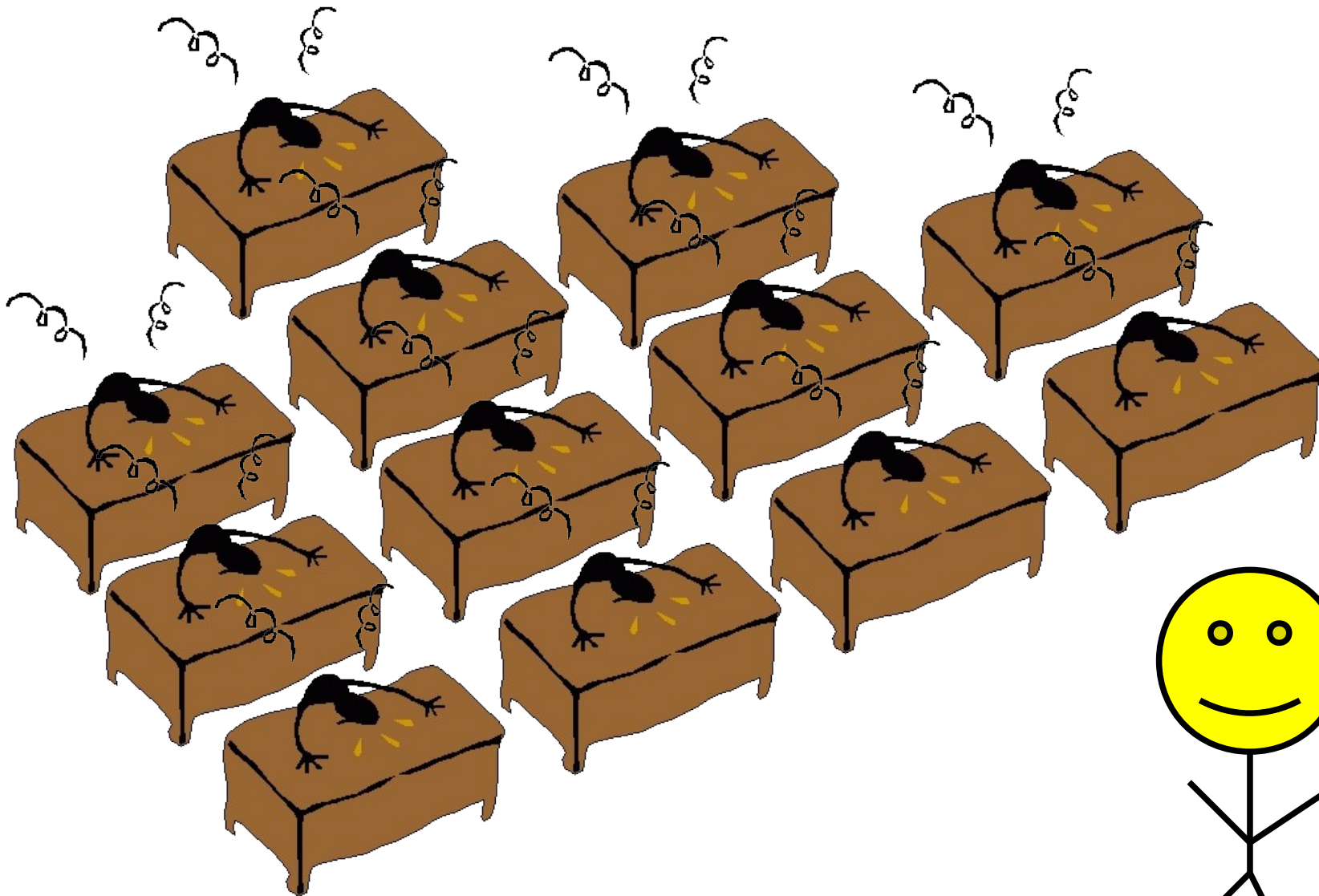
(G4VSensitiveDetector::ProcessHits(..) called by the G4 kernel)

G4RunManager::SetNumberOfEventsToBeStored(G4int n):

keep n consecutive events & their hit-collections in memory

- 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!



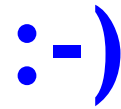
That's it ...

... 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)

Questions??



(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



- **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>

Thank you!



Enjoy CSC 06
and the real
excursion(s)!

