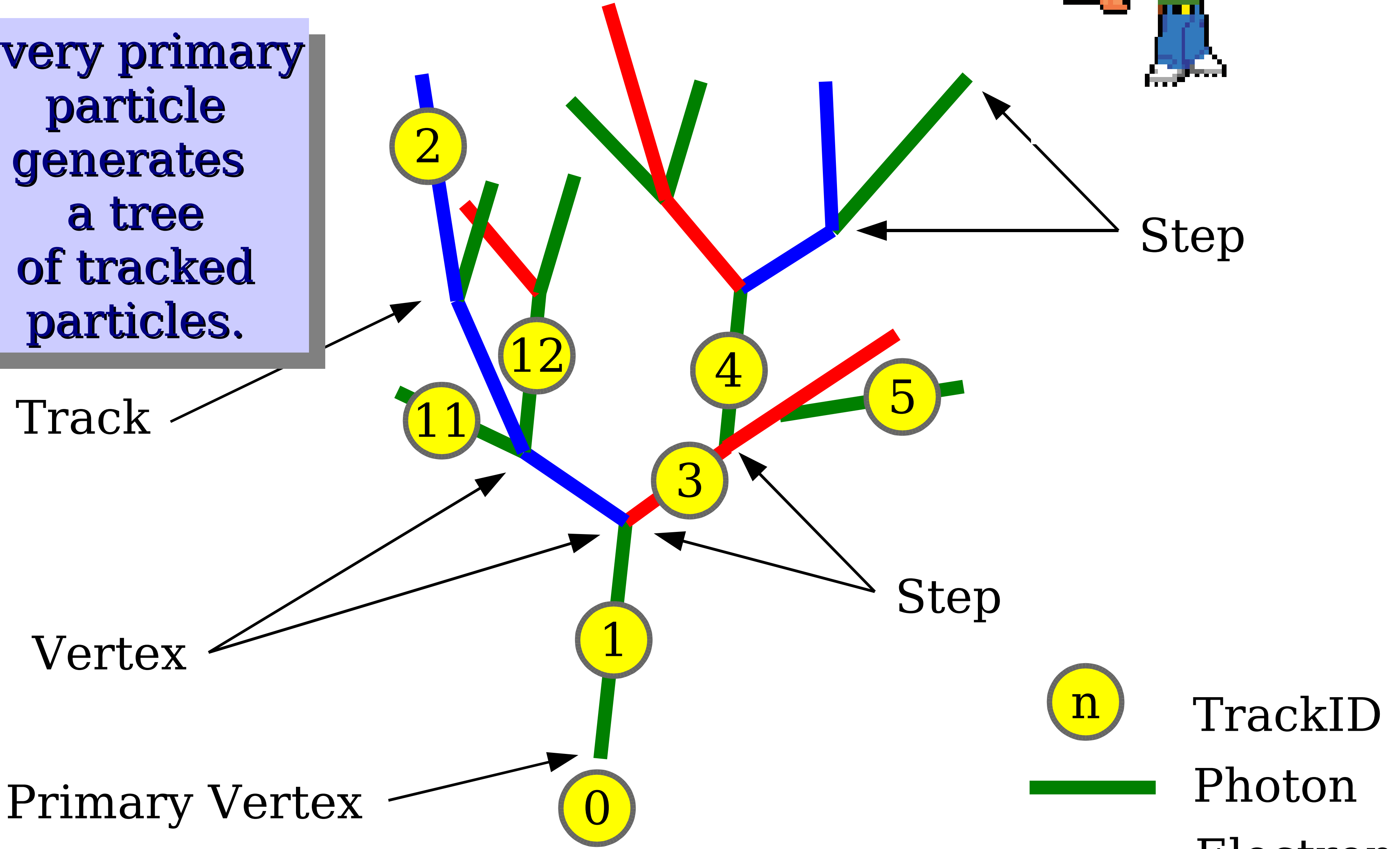
Assume, we have a RunManger,
and all UserInitializations and
-actions are registered to Geant4.

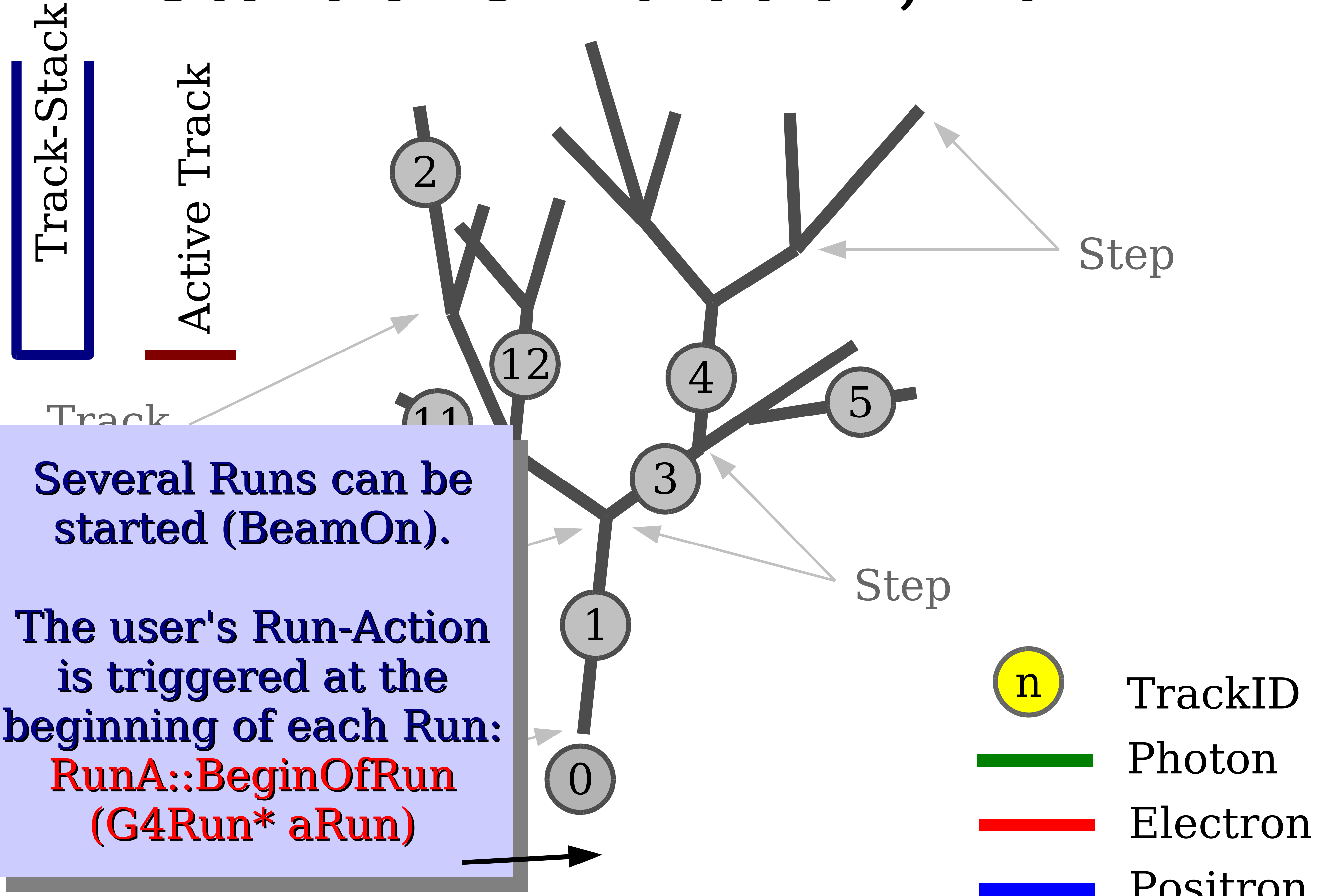
Let's follow the simulation of an electromagnetic shower
to see what gets when and where invoked by G4

And Action!

Every primary particle generates a tree of tracked particles.



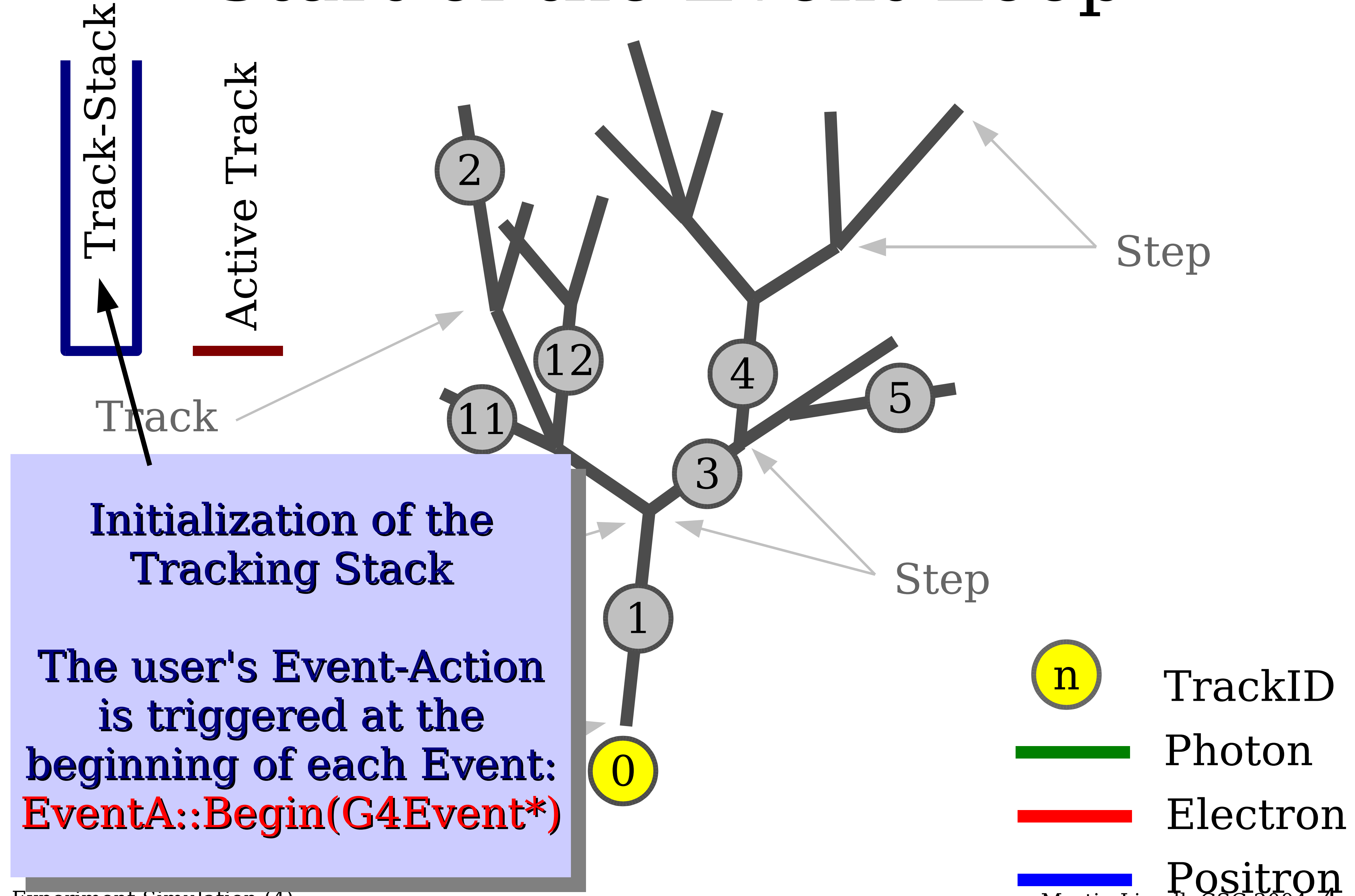
Start of Simulation, Run



Several Runs can be started (BeamOn).

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

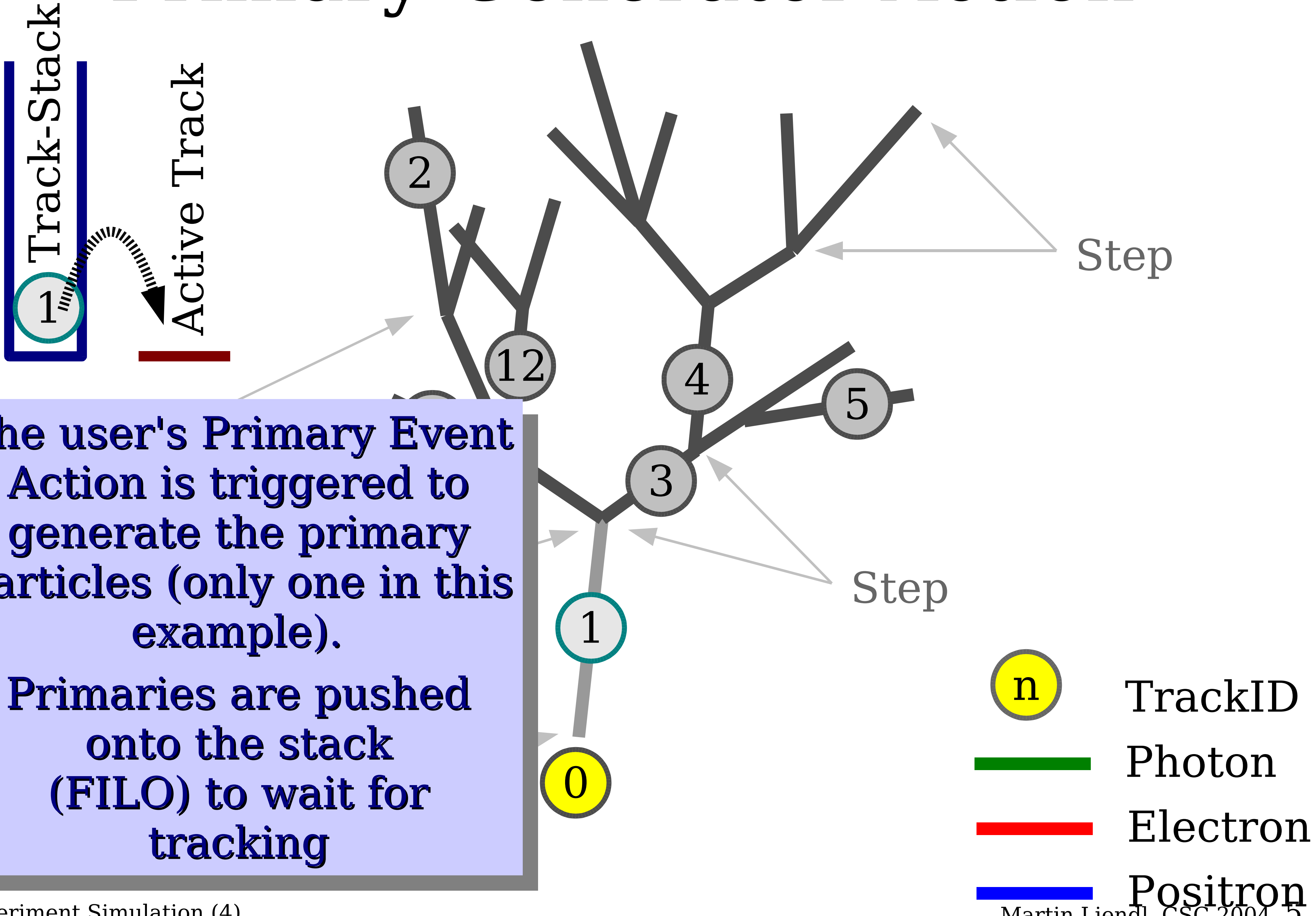
Start of the Event-Loop



Initialization of the Tracking Stack

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

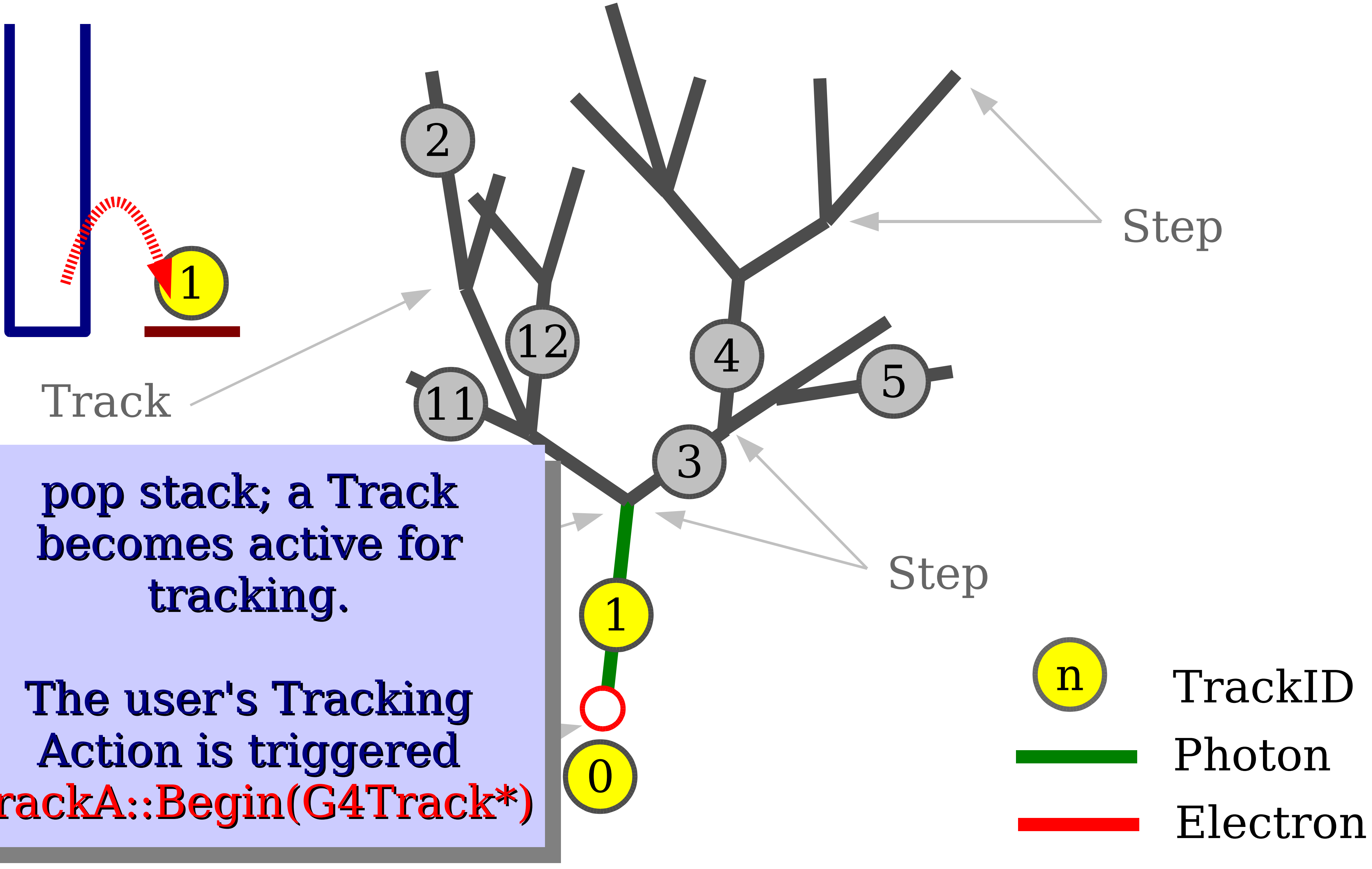
Primary Generator Action



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

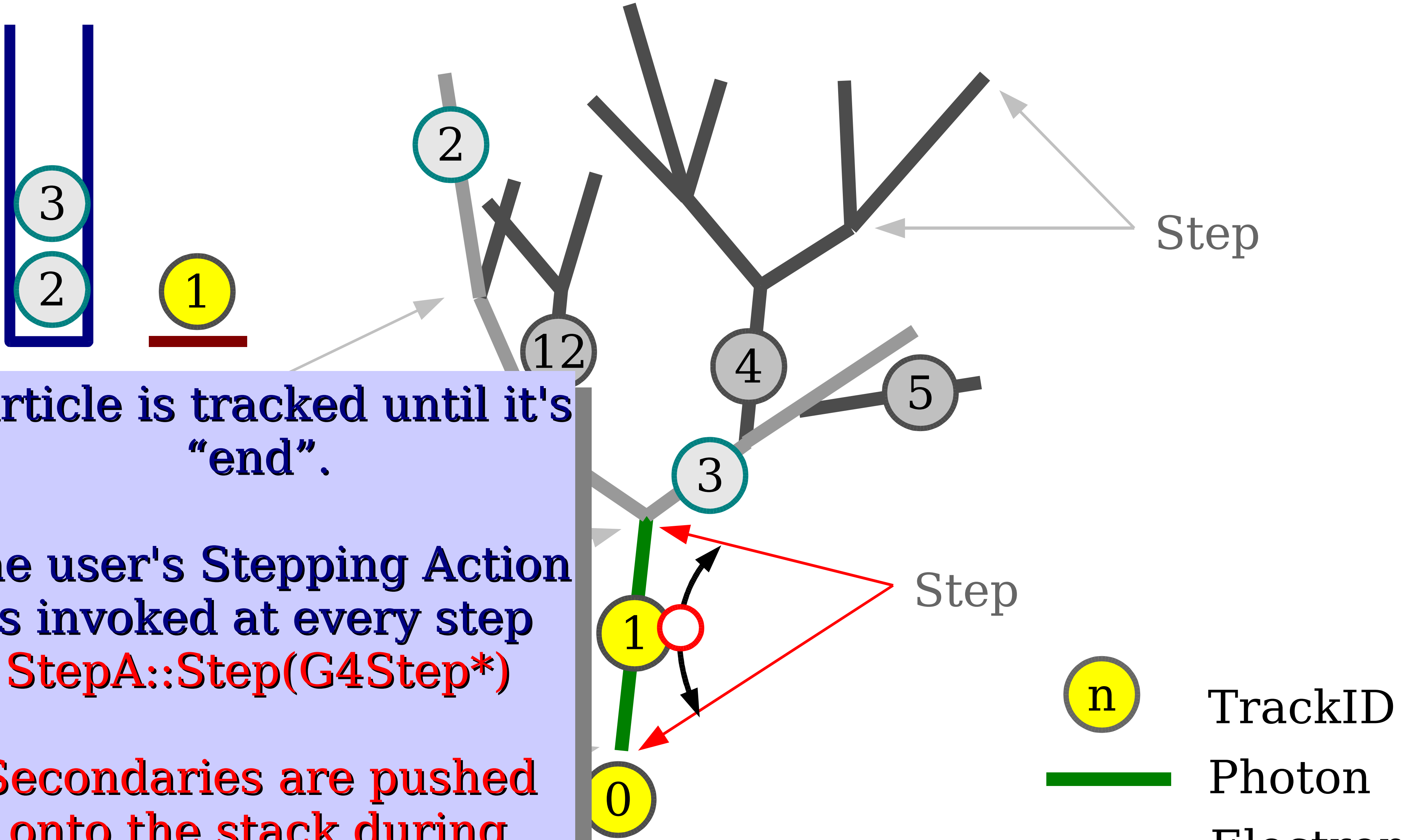
Pop Stack, Tracking (Stepping) until end of Track, generated Secondaries pushed on Stack



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

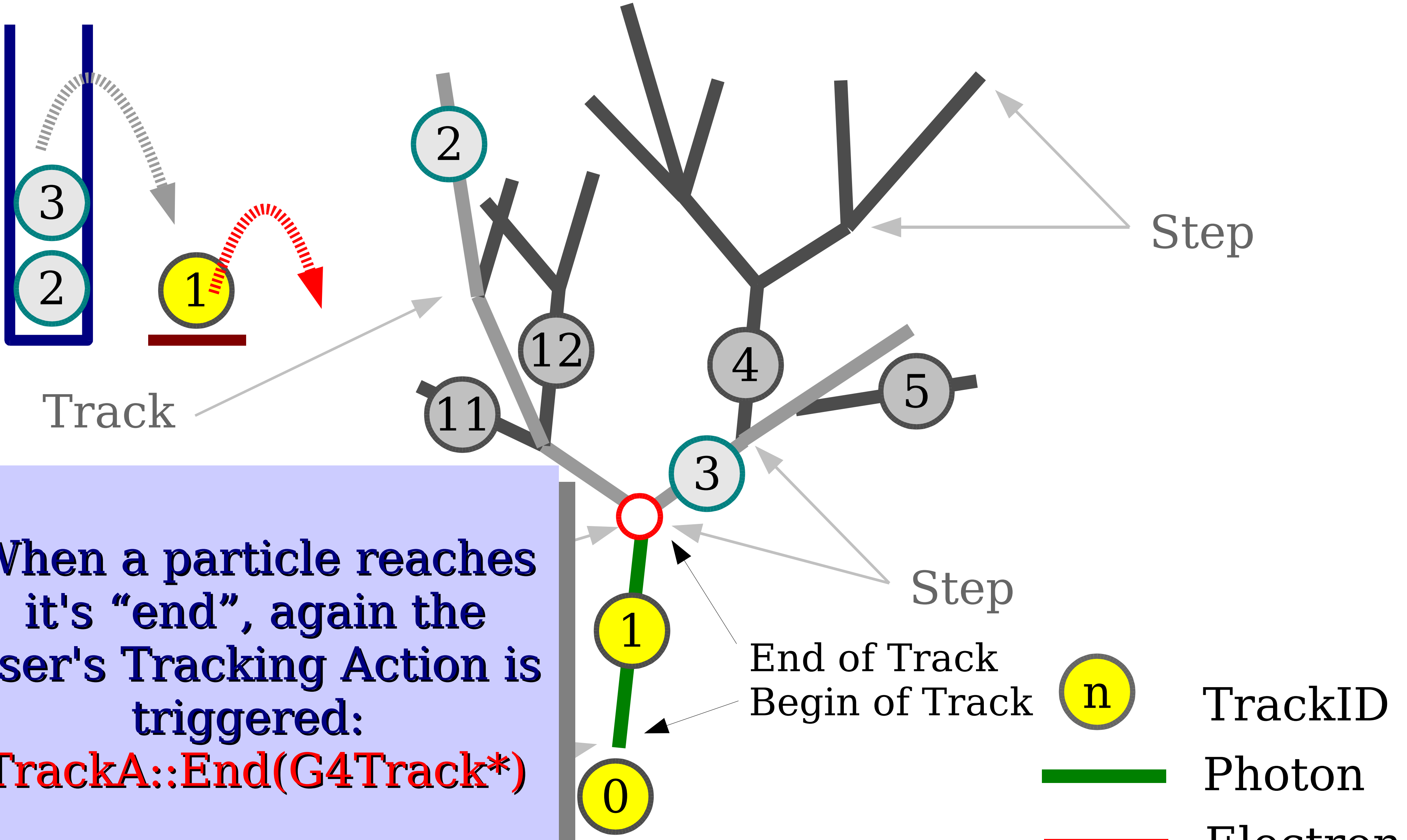


particle is tracked until it's "end".

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

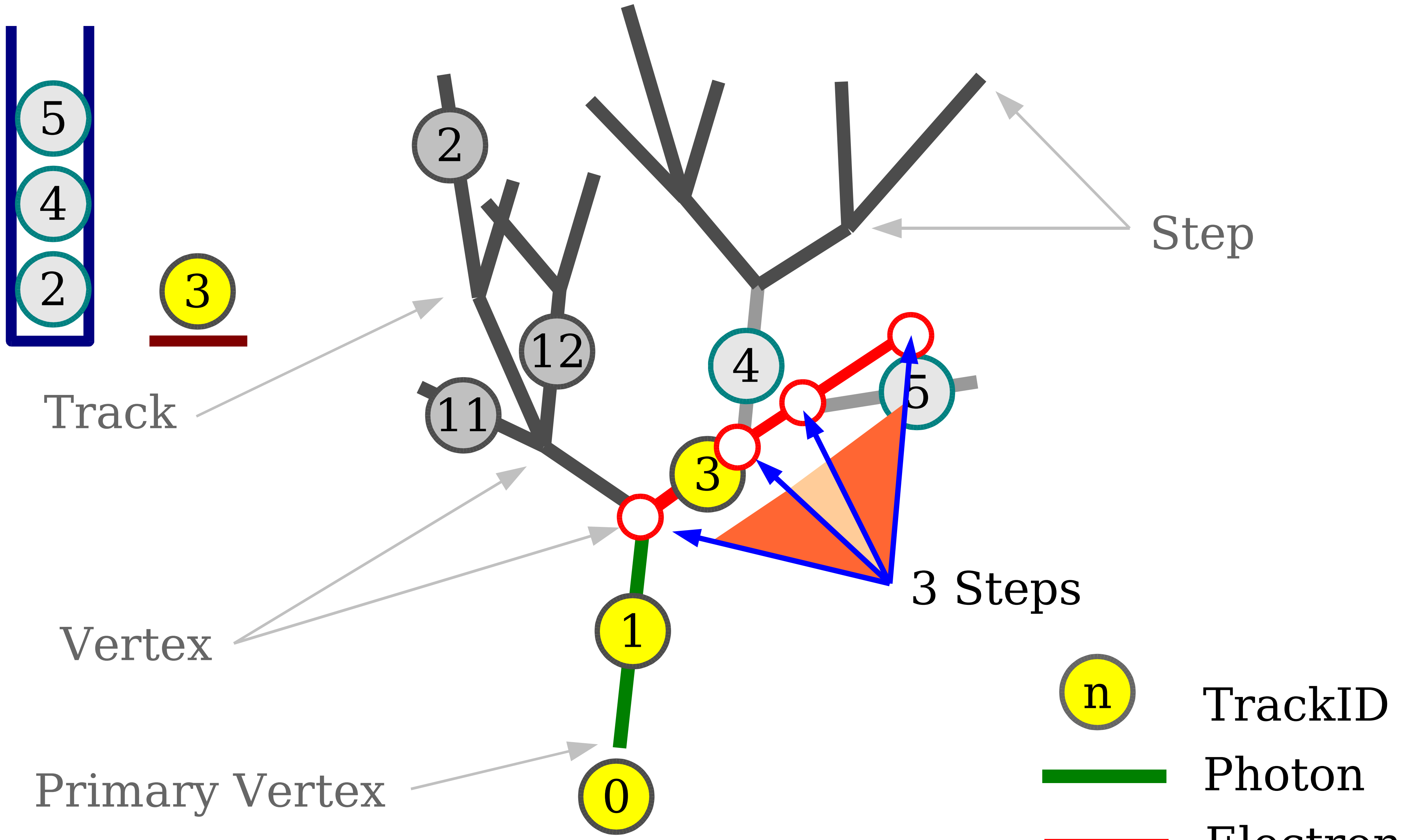
Secondaries are pushed onto the stack during stepping

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

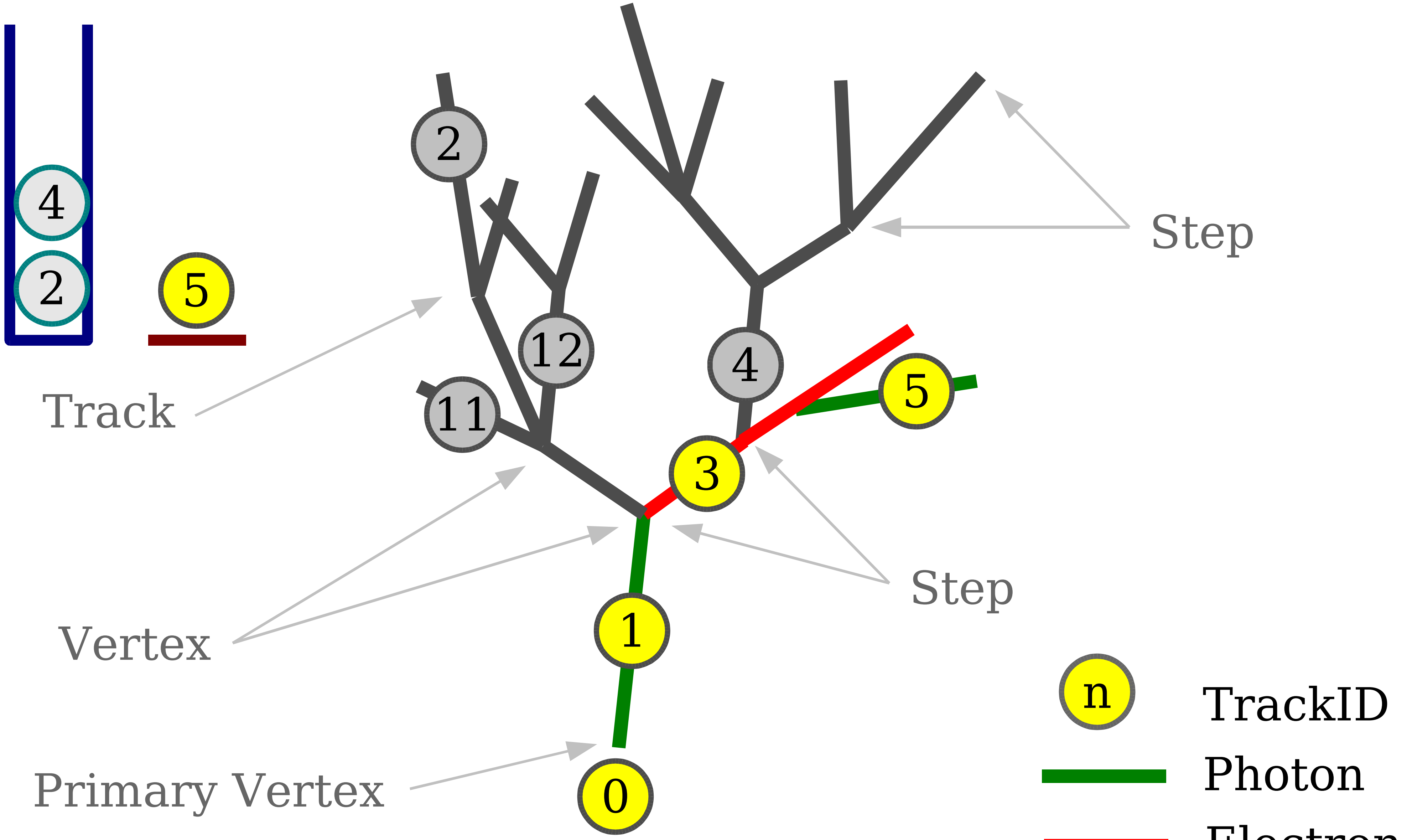


When a particle reaches it's “end”, again the user's Tracking Action is triggered:
TrackA::End(G4Track*)

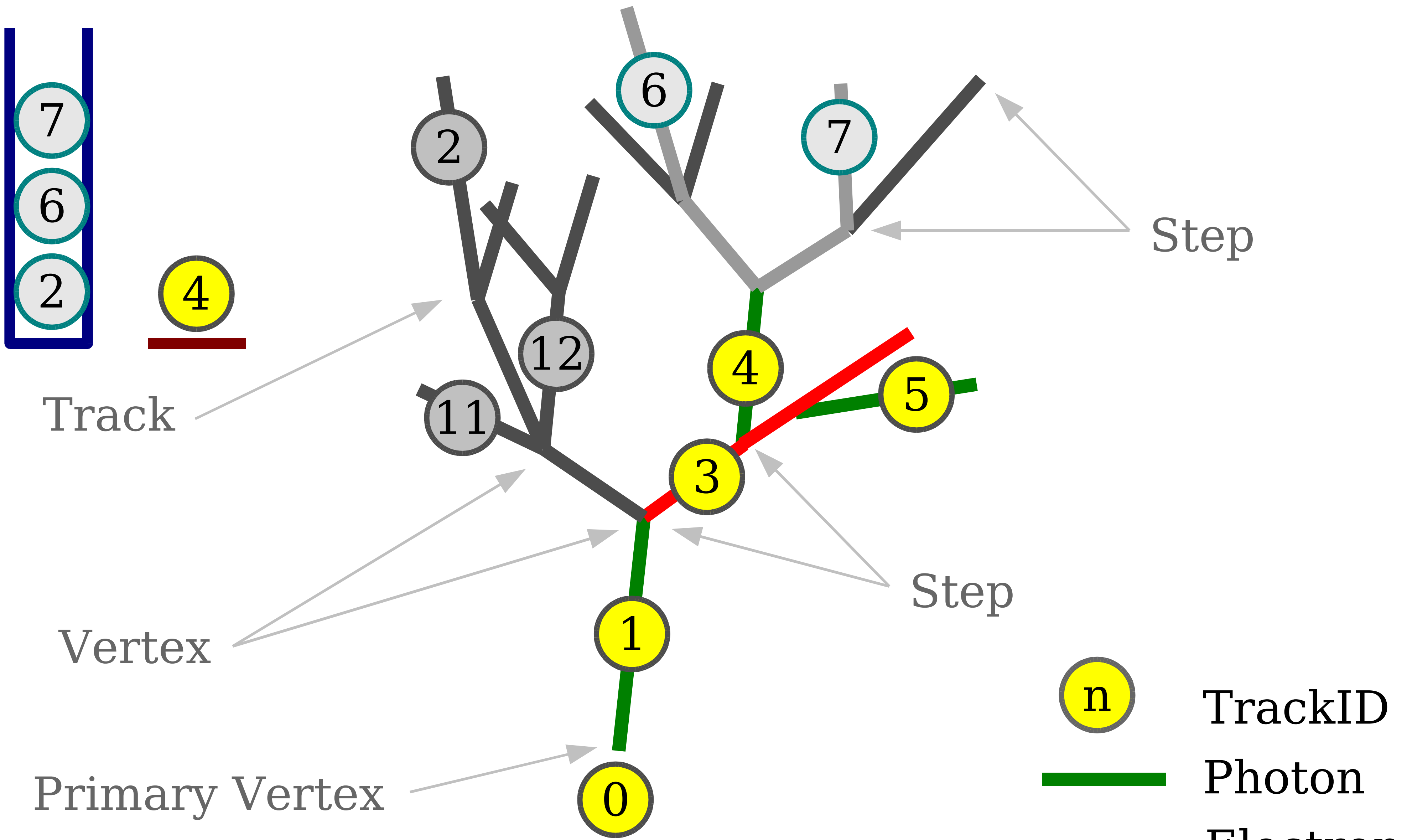
Pop Stack, Tracking (Stepping) until end of Track, generated Secondaries pushed on Stack



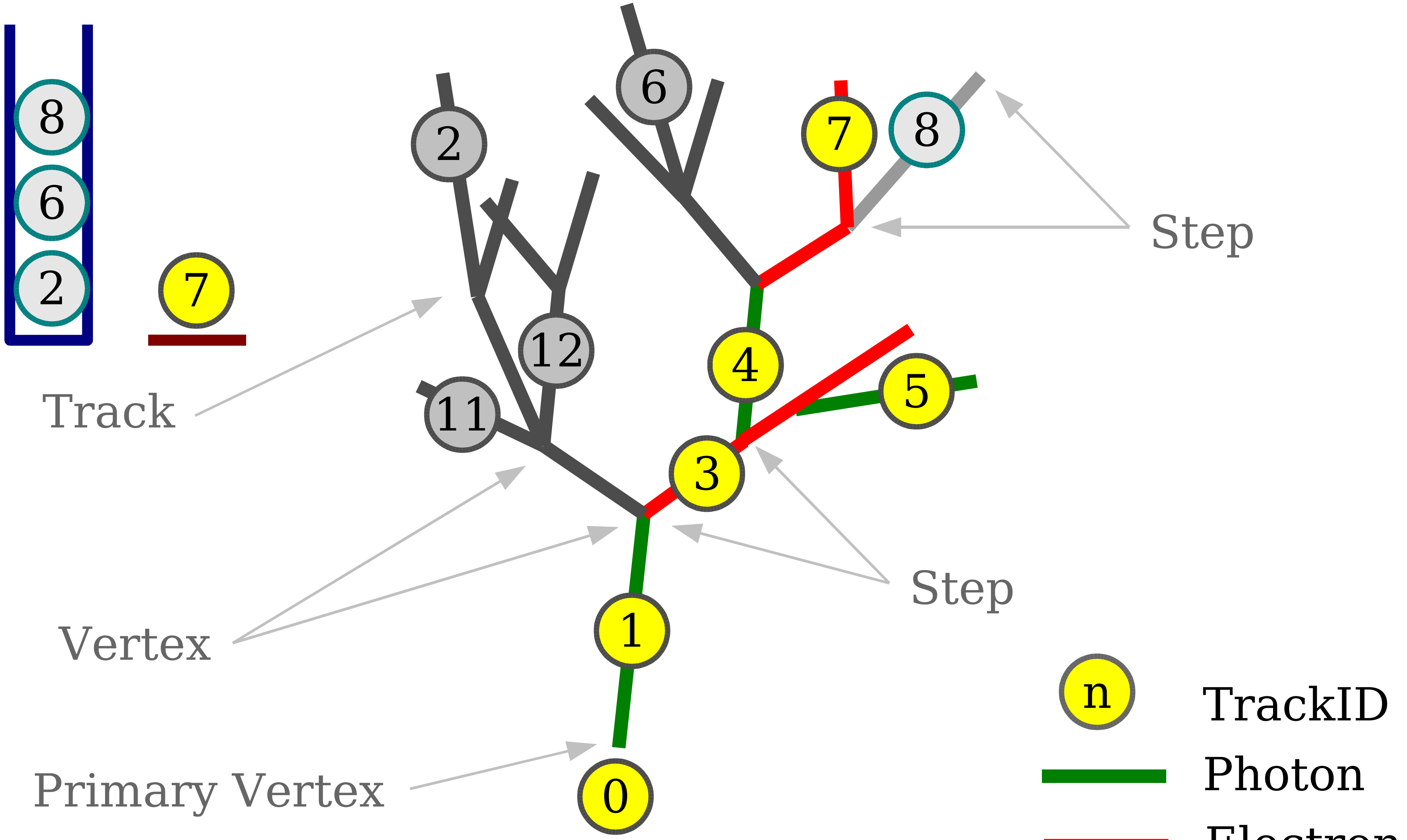
Pop Stack, Tracking (Stepping) until end of Track, generated Secondaries pushed on Stack



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

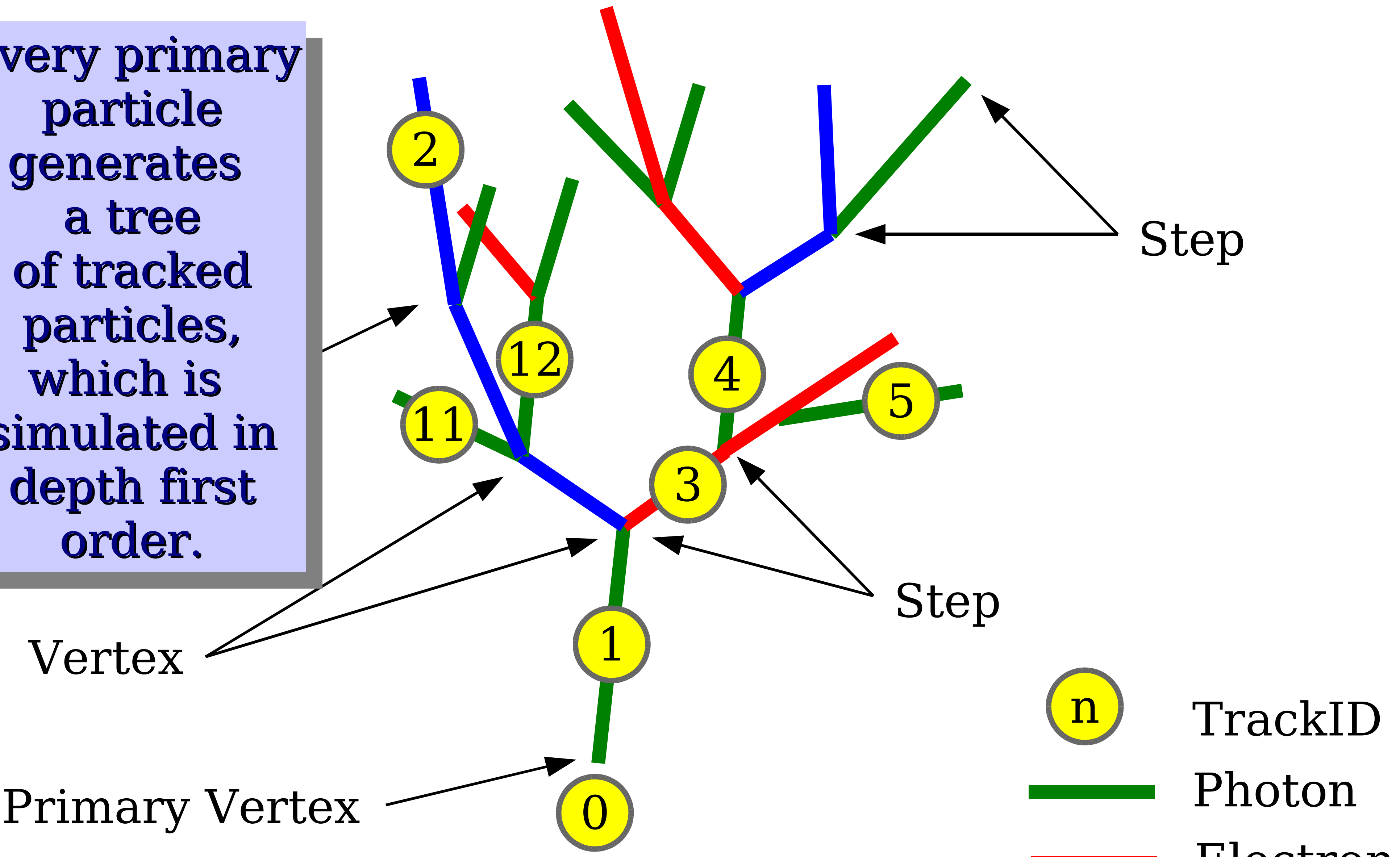


“Depth First” Tree Traversal



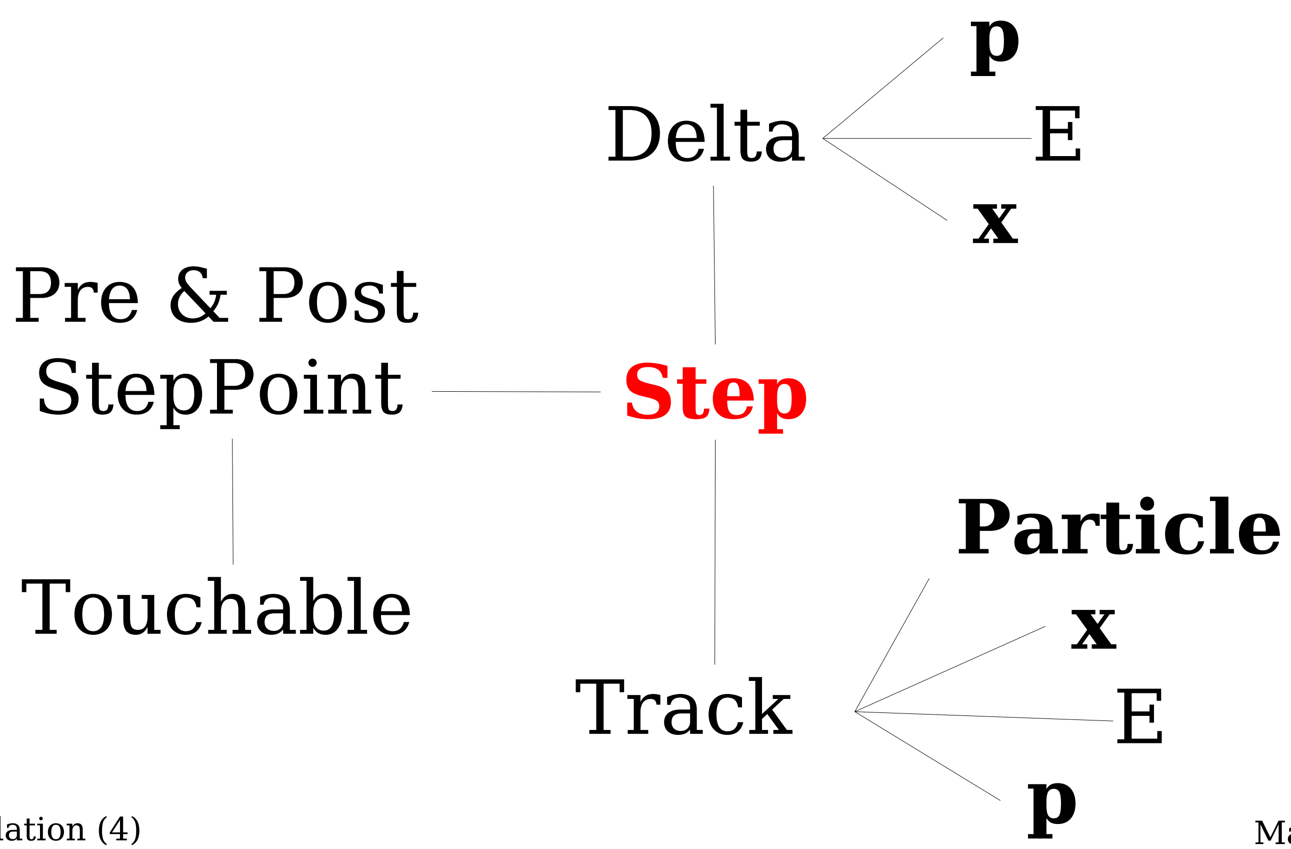
Track/Step (once more)

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



Stepping Action

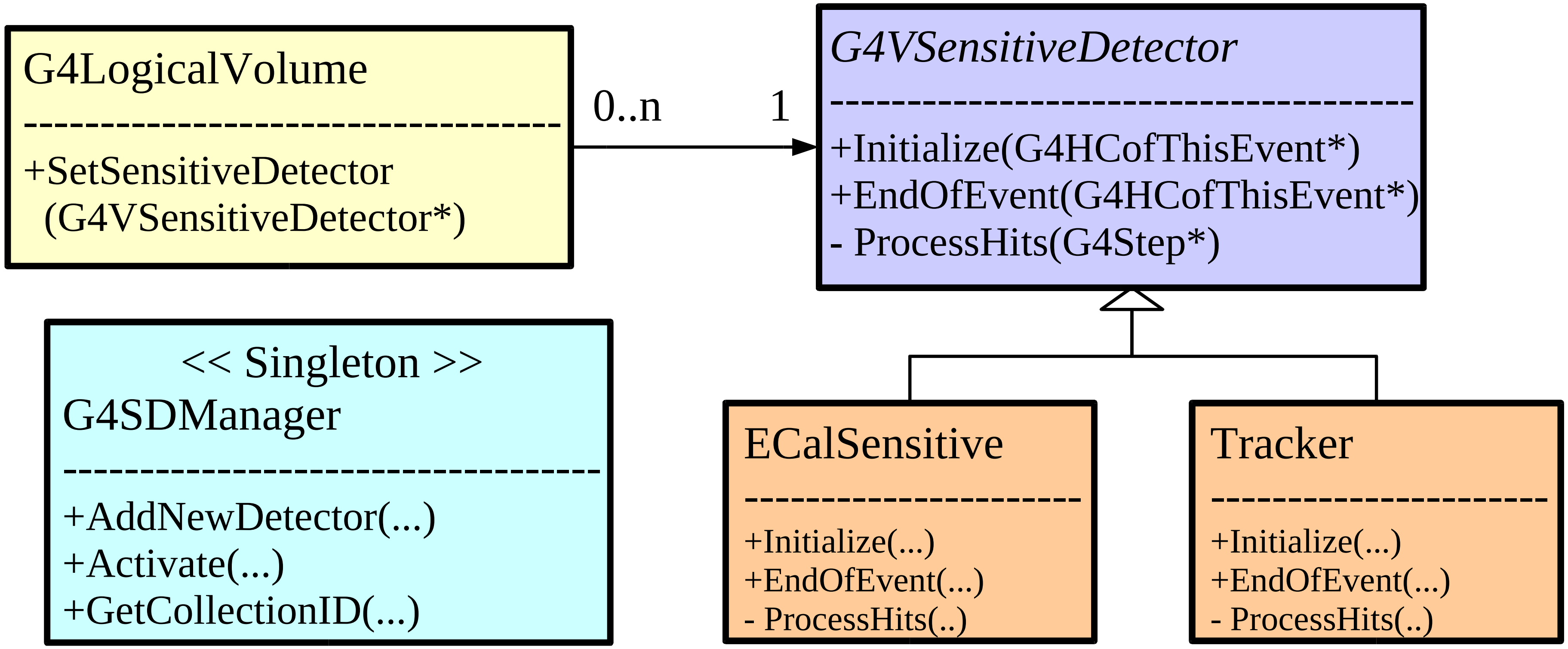
- G4VSteppingAction: finest granularity of obtaining information
  - register a sub-class to the Geant4 kernel
  - at every step: G4VSteppingAction::UserSteppingAction(G4Step * info)



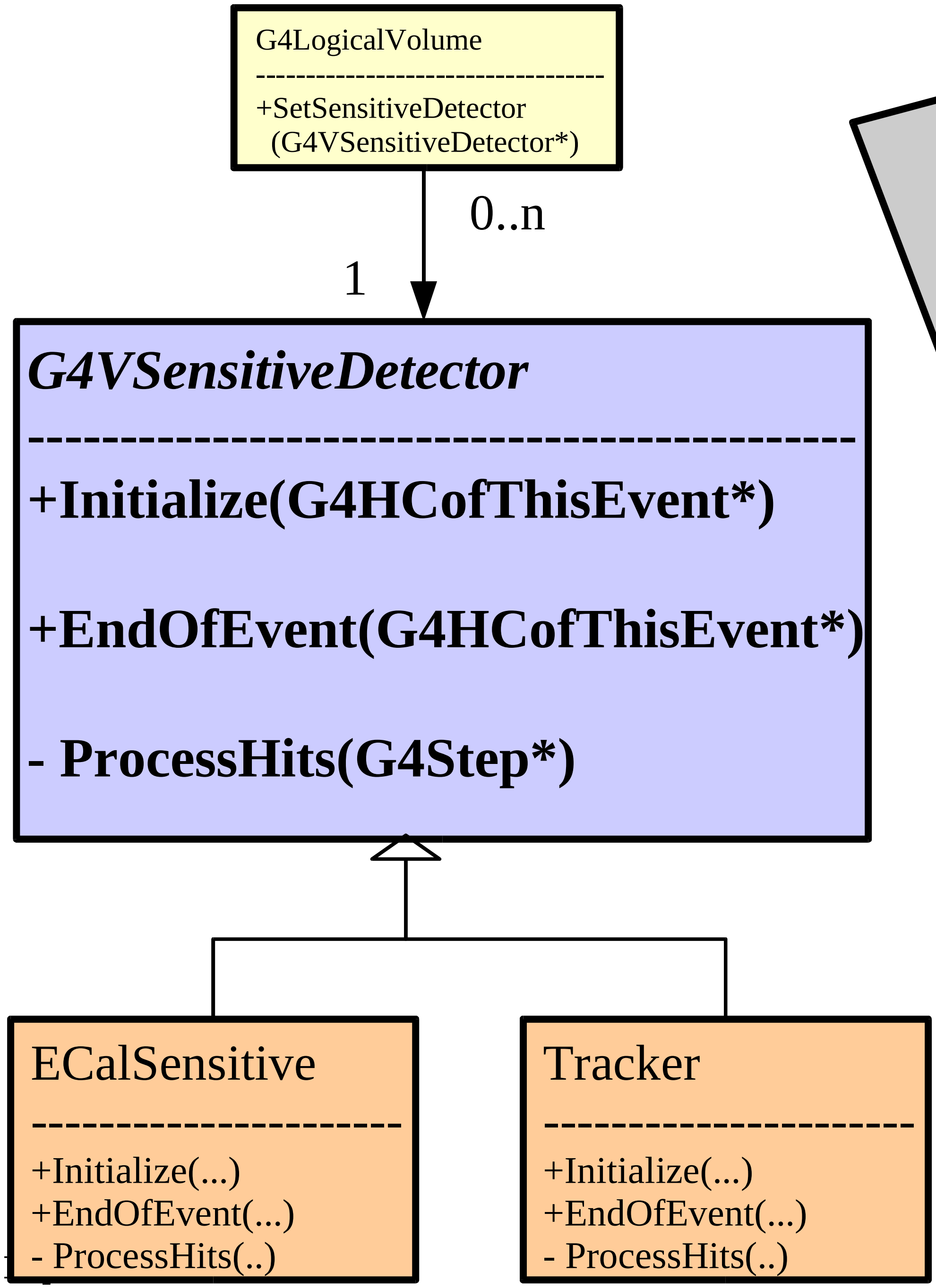
- The User-Actions shown before are generic:
  - invoked every time a Geant4 event occurs
  - independently invoked of the particularities of your experimental setup
- Want to have specific actions for specific detector parts:
  - Tracker: sensitive elements measure positions of single tracks
  - Calorimeter: 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

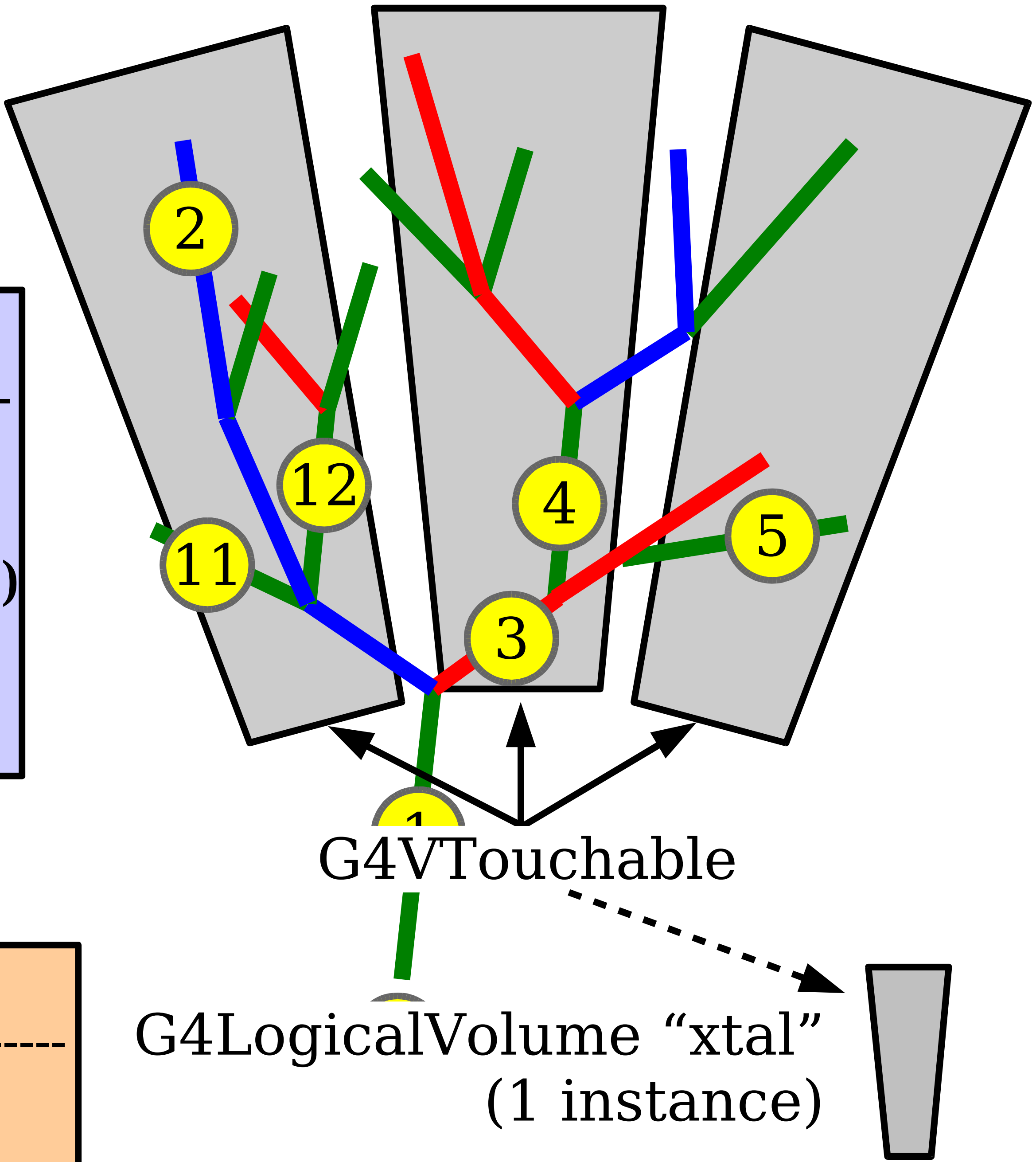
Sensitive Detector

- 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:

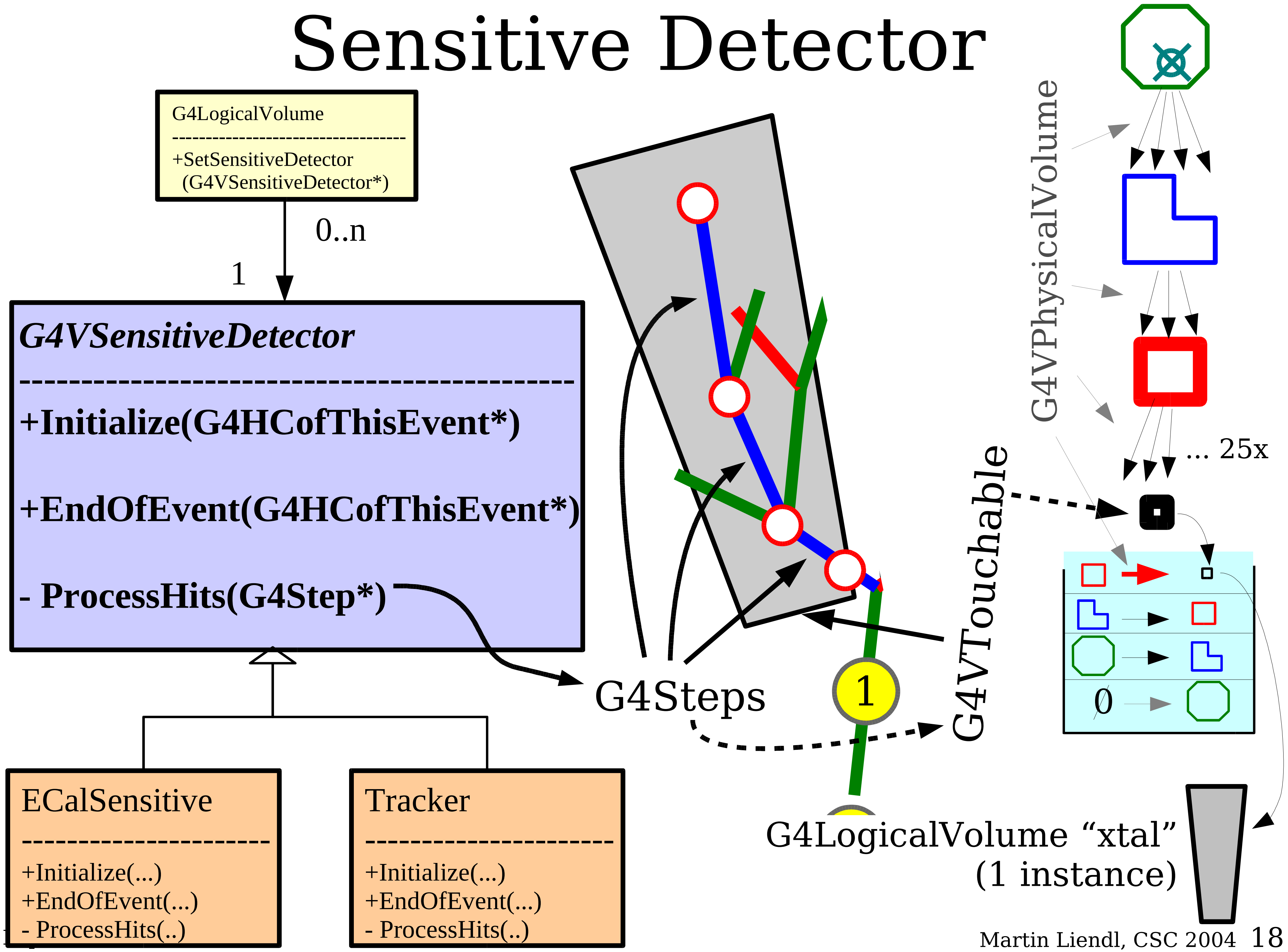


Sensitive Detector & Geometry



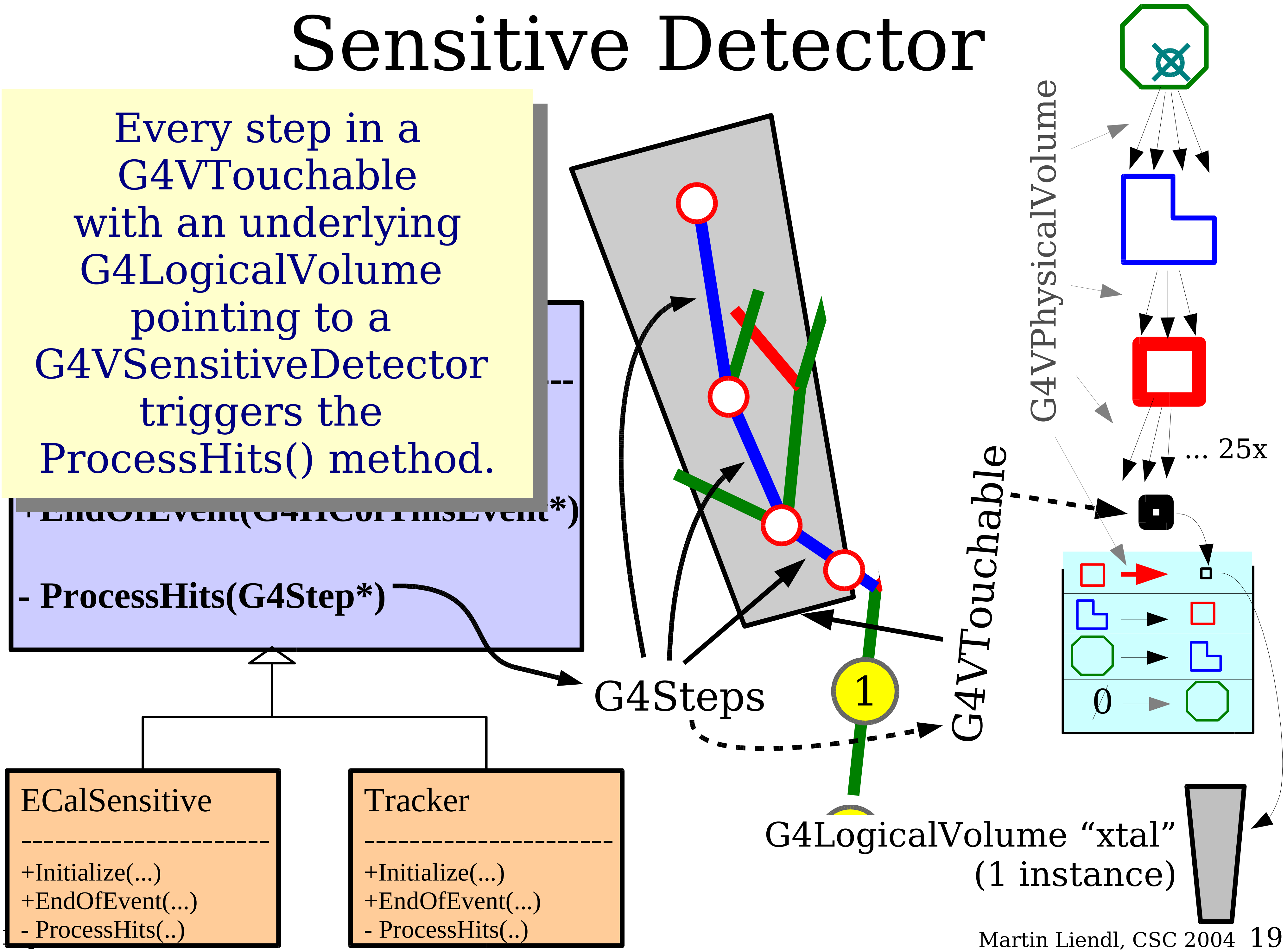


Sensitive Detector

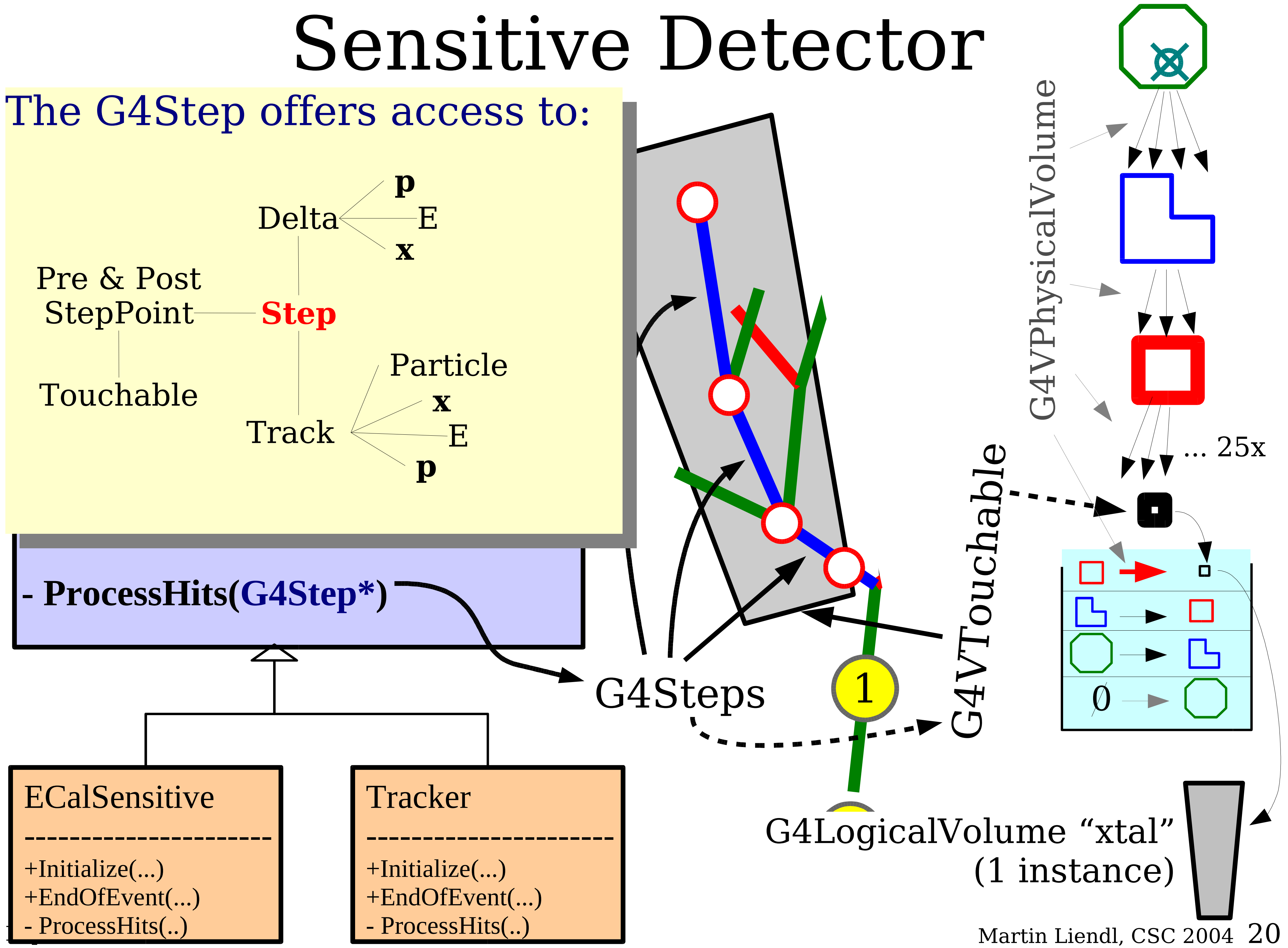


Sensitive Detector

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

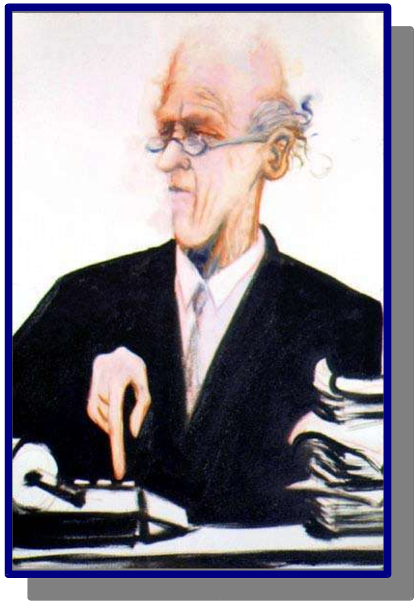


Sensitive Detector



THE Simulation FAQ

Q: What to do?
A: Tedious bookkeeping!



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

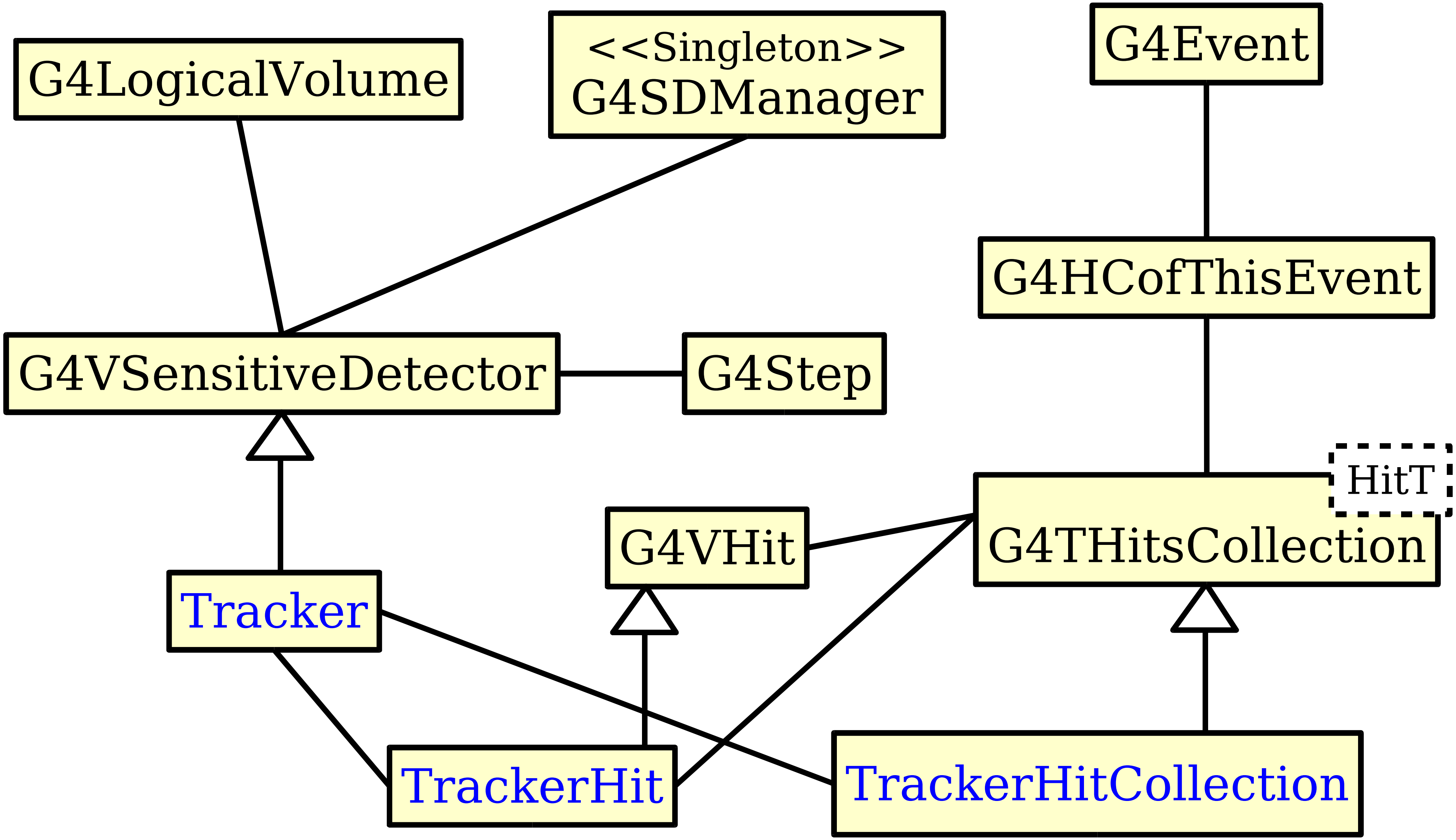
The User Actions and Sensitive Detector classes are the means for filtering & bookkeeping the huge amount of simulation information available during tracking.

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

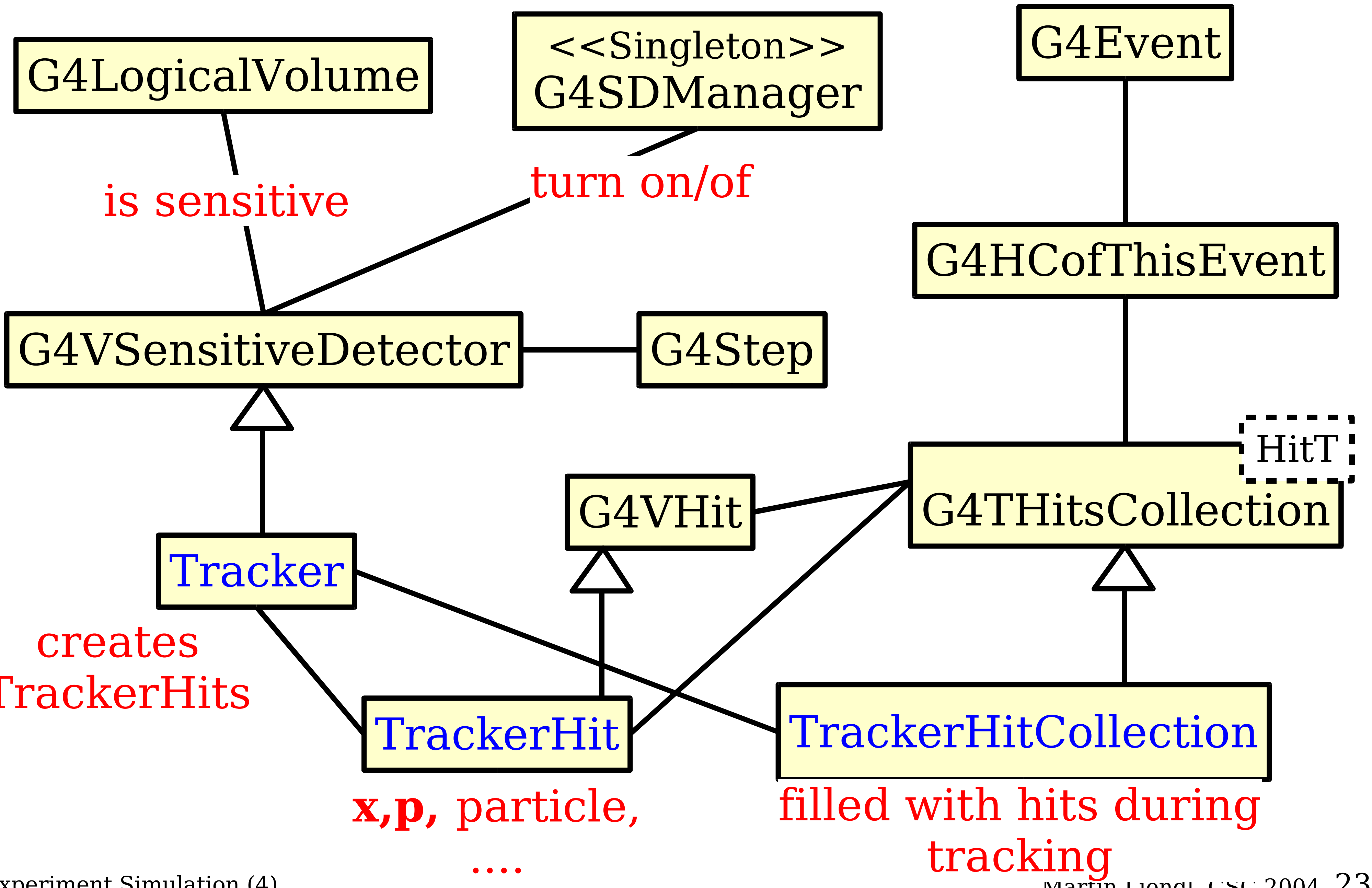
Geant4 helps in providing interfaces for Hits and Hit Collections.

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

Overview of G4 Hits & Collections



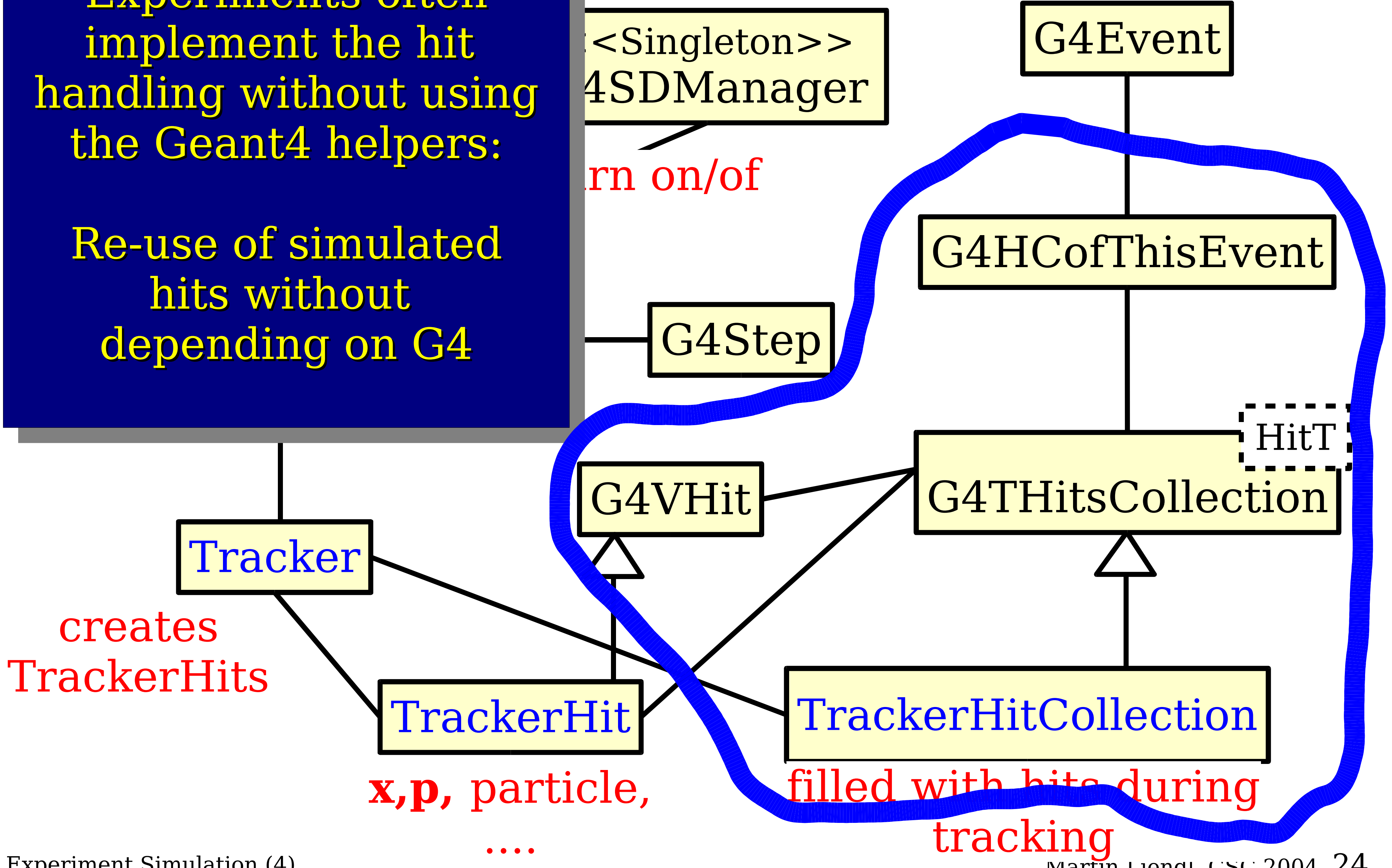
Overview of G4 Hits & Collections



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
G4VDigitizerModule	G4VSensitiveDetector
G4VDi	
G4TDi	>
G4DCo	

Differen

G4VDigiti me!

(G4VSensiti

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

CMS does this!

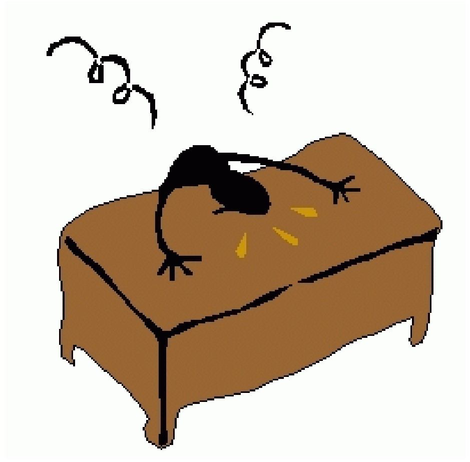
G4VDigitizerModule::Digitize(..) is never called by the G4!

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

G4RunManager::SetNumberOfEventsToBeStored(G4int n):

keep n consecutive events & their hit-collections in memory

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)

That's it (not yet) ...

See: http://www.cern.ch/geant4

Some of the issues below will be addressed in the following slides about how it's done at CMS.

- 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)

PART VII

(the final chapter)

CMS Experiment Simulation:

Selected Topics

Experiment simulation at CMS

- **Compact Muon Solenoid**
  - compact, because most of the hadronic calorimeter, all of the electromagnetic calorimeter, and all of the tracker are inside a superconducting magnet solenoid producing a field of maximal 4 Tesla

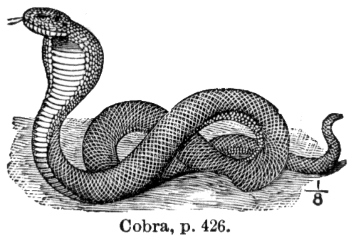
CMS Topics

Will present some selected topics related to simulation

- ➔ CMS software for Simulation and Reconstruction
- Detector Description
- Visualization
- Simulation example
- Framework
- User Interface for batch processing
- CMS Computing Infrastructure

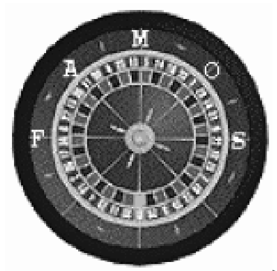
AAA(Acronyms, Animals, Anything)rghhh!

- **COBRA**
  - handling of persistency via POOL (data, metadata)
  - common definitions: hits, geometry, …
  - definition of the "event loop", dispatch of information to user code

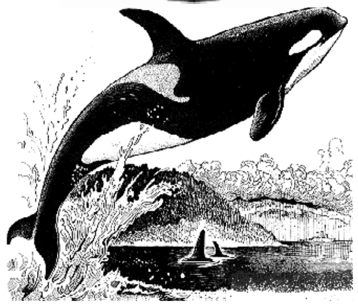


- **OSCAR**
  - "full" experiment simulation based on Geant4

- **FAMOS**
  - libraries for fast simulation
  - used stand alone, or in OSCAR, or in ORCA



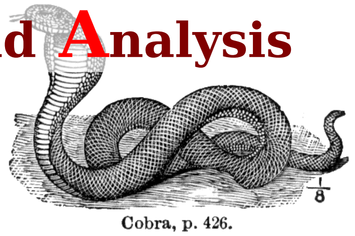
- **ORCA**
  - pile-up simulation
  - detector response simulation
  - event reconstruction



- **IGUANA**
  - Visualization and Analysis framework

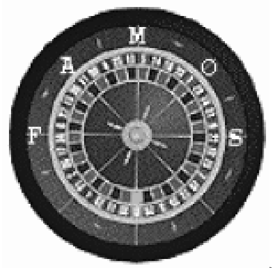
AAA (in development since ~1997) **rghhh!**

- **Coherent Object-oriented Base for Reconstruction and Analysis**
  - handling of persistency via POOL (data, metadata)
  - common definitions: hits, geometry, ...
  - definition of the "event loop", dispatch of information to user code

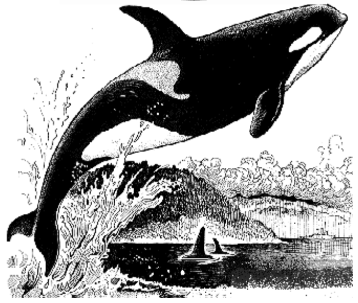


- **Object-oriented Simulation for CMS Analysis and Reconstruction**
  - "full" experiment simulation based on Geant4

- **FAast MOnte Carlo Simulation**
  - libraries for fast simulation
  - used stand alone, or in OSCAR, or in ORCA

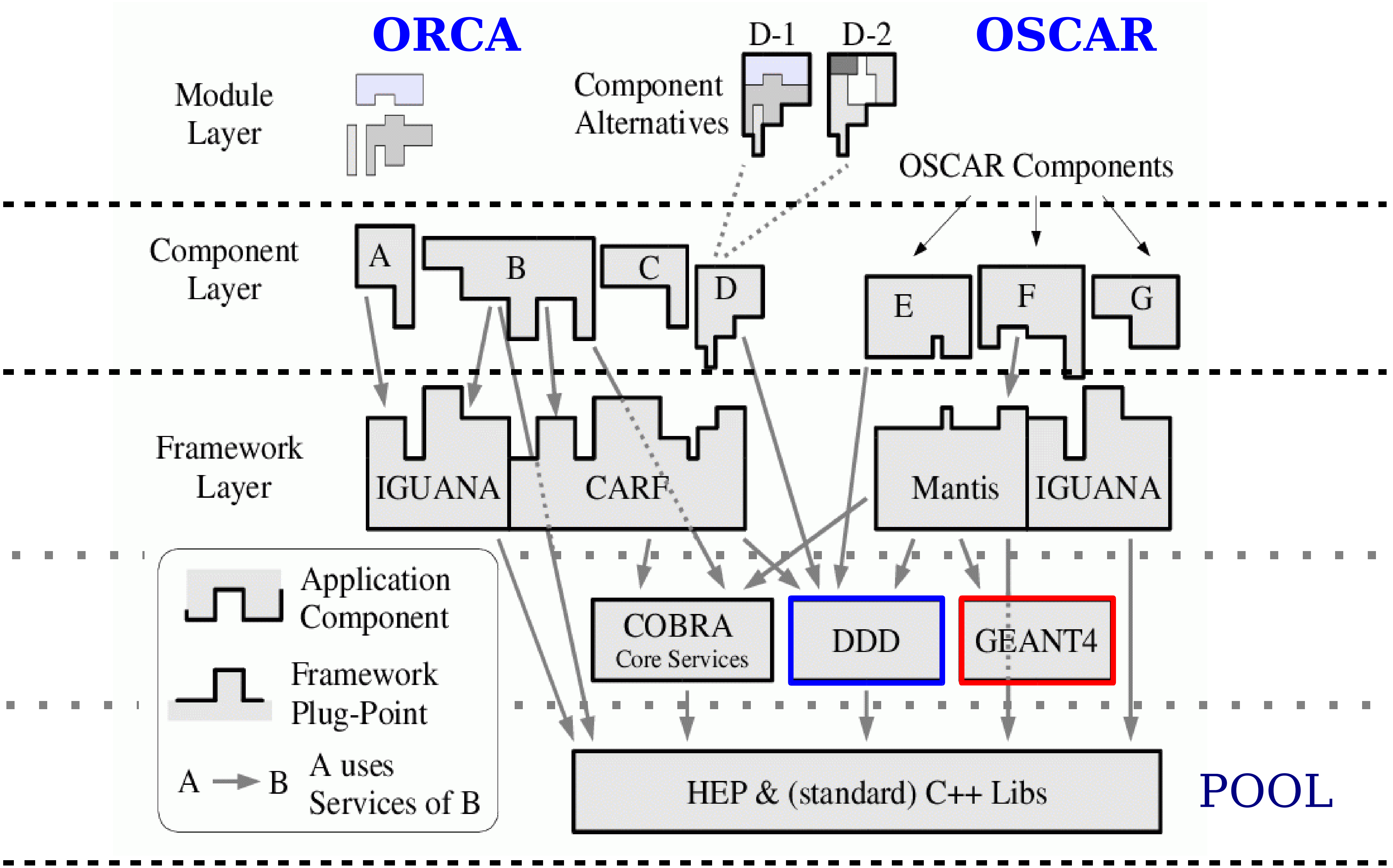


- **Object-oriented Reconstruction for CMS Analysis**
  - pile-up simulation
  - detector response simulation
  - event reconstruction

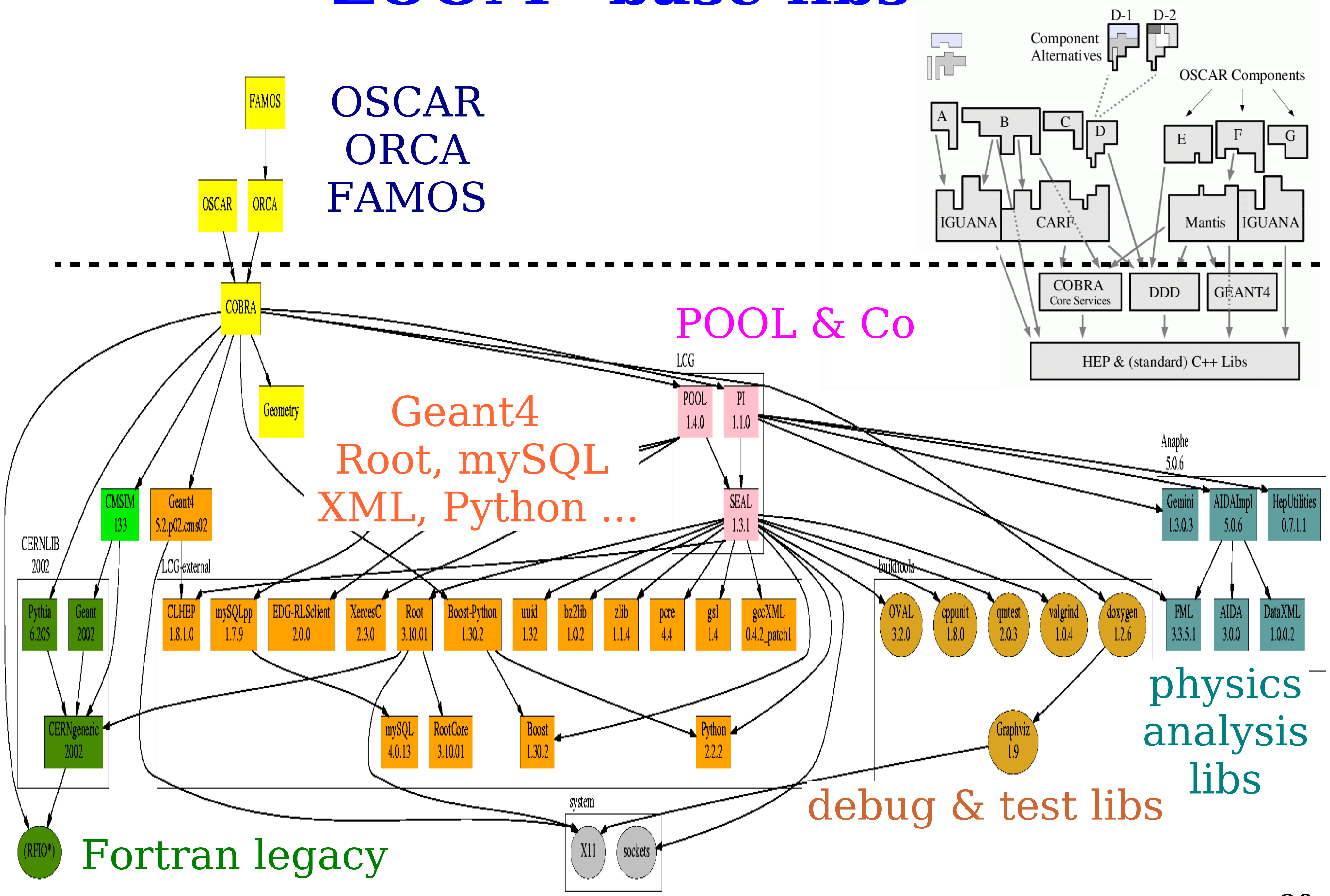


- **Interactive Graphics for User ANAlysis**
  - Visualization and Analysis framework

Layers


ORCA
OSCAR
Module Layer
Component Alternatives
D-1
D-2
OSCAR Components
Component Layer
A
B
C
D
E
F
G
Framework Layer
IGUANA
CARF
Mantis
IGUANA
Application Component
Framework Plug-Point
A → B A uses Services of B
COBRA Core Services
DDD
GEANT4
HEP & (standard) C++ Libs
POOL

ZOOM "base libs"



OSCAR
ORCA
FAMOS

POOL & Co

Geant4
Root, mySQL
XML, Python ...

debug & test libs

physics
analysis
libs

Fortran legacy

- ✓ CMS software for Simulation and Reconstruction
- ➔ Detector Description
- Visualization
- Simulation example
- Framework
- User Interface for batch processing
- CMS Computing Infrastructure

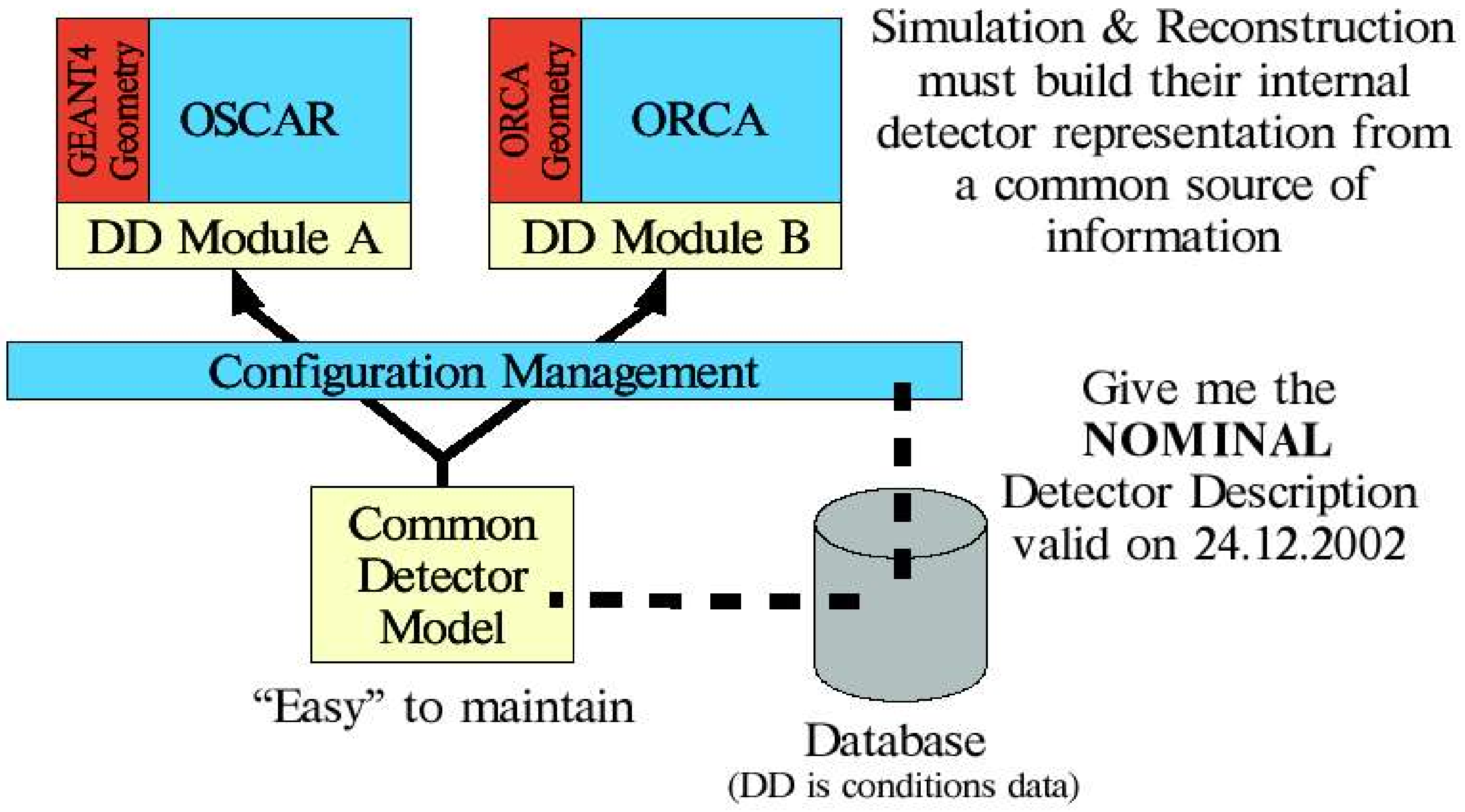
Detector Description

- Realistic simulation requires a very accurate detector model
  - all sensitive parts and all non-sensitive parts of the detector have to be described accurately to the simulation
  - describes an ideal detector (“blue print”) as designed (computing time!!)
  - Geant4 constraints (overlaps!); Geant4 practice: “ghost volumes” for grouping, ...
- Real event data will come from the real detector
  wow, what a statement ;-)
  - only from the sensitive parts; the information about “dead” material is not directly contained in event data
  - reconstruction software is bound to data and therefore interested in the accurate **conditions** of the detector element
    - time dependent conditions, stored in the Conditions DB
    - deviation of nominal positions, calibration factors, ...

Detector Description

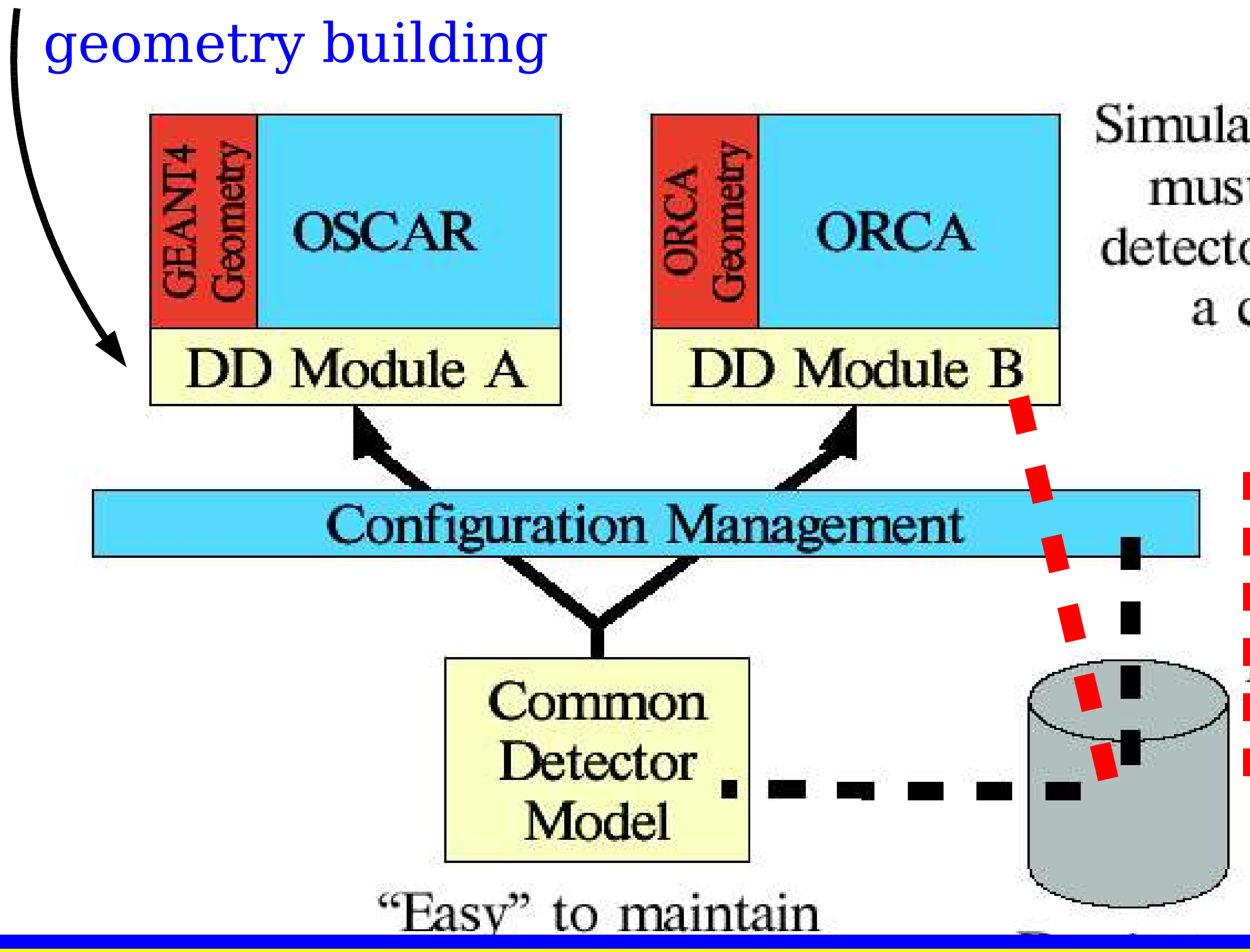
- Deviation of nominal positions means
  - have to treat each part **individually** at the instance level
  - graph structure (sim) -> **tree** structure (reco)
- Simulation and reconstruction models have to be consistent!
  - when all conditions are “1 factors”, models should describe the same nominal positions
- CMS employs a common source for detector description, the **Detector Description Database**
  - has it's own, **light object model** for geometry
  - description format is an **XML** language
  - C++ Algorithm can be user-defined additionally to populate instances of the object model

Detector Description


GEANT4 Geometry
OSCAR
DD Module A
ORCA Geometry
ORCA
DD Module B
Simulation & Reconstruction must build their internal detector representation from a common source of information
Configuration Management
Give me the NOMINAL Detector Description valid on 24.12.2002
Common Detector Model
"Easy" to maintain
Database
(DD is conditions data)

Detector Description

automatized Geant4
geometry building



Simulation & Reconstruction must build their internal detector representation from a common source of information

Conditions DB also stores alignment and calibration data!

One common model to avoid hard-coded values for geometry or magnetic field descriptions in the applications!!!

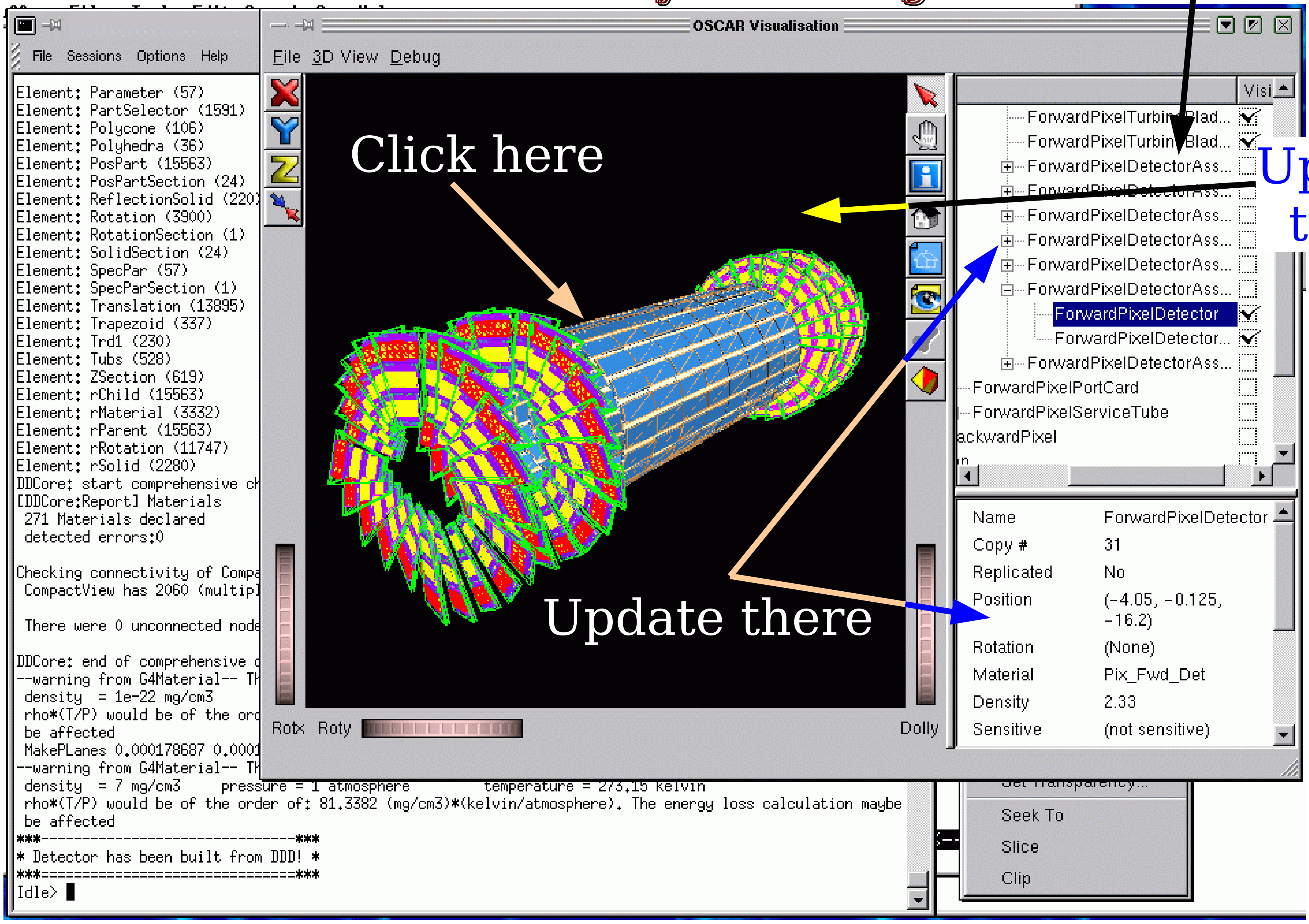
- ✔ CMS software for Simulation and Reconstruction
- ✔ Detector Description
- ➔ Visualization
- • Simulation example
- • Framework
- • User Interface for batch processing
- • CMS Computing Infrastructure

Visualization

- Geant4 supports quite some visualizers / visualization formats:
  OpenGL, VRML, DAWN, …
  - but no graphical interactivity!
    (no graphical browsing of geometry, hits; picking of volumes on the screen, …)
  - event if one builds a nice GUI directly on top of Geant4, one has little re-use gain for non Geant4 based applications
- CMS wants to have the same “look & feel” for its applications, fitting the requirements of the collaboration
  - **IGUANA - Interactive Graphics for User ANAlysis**

Visualization

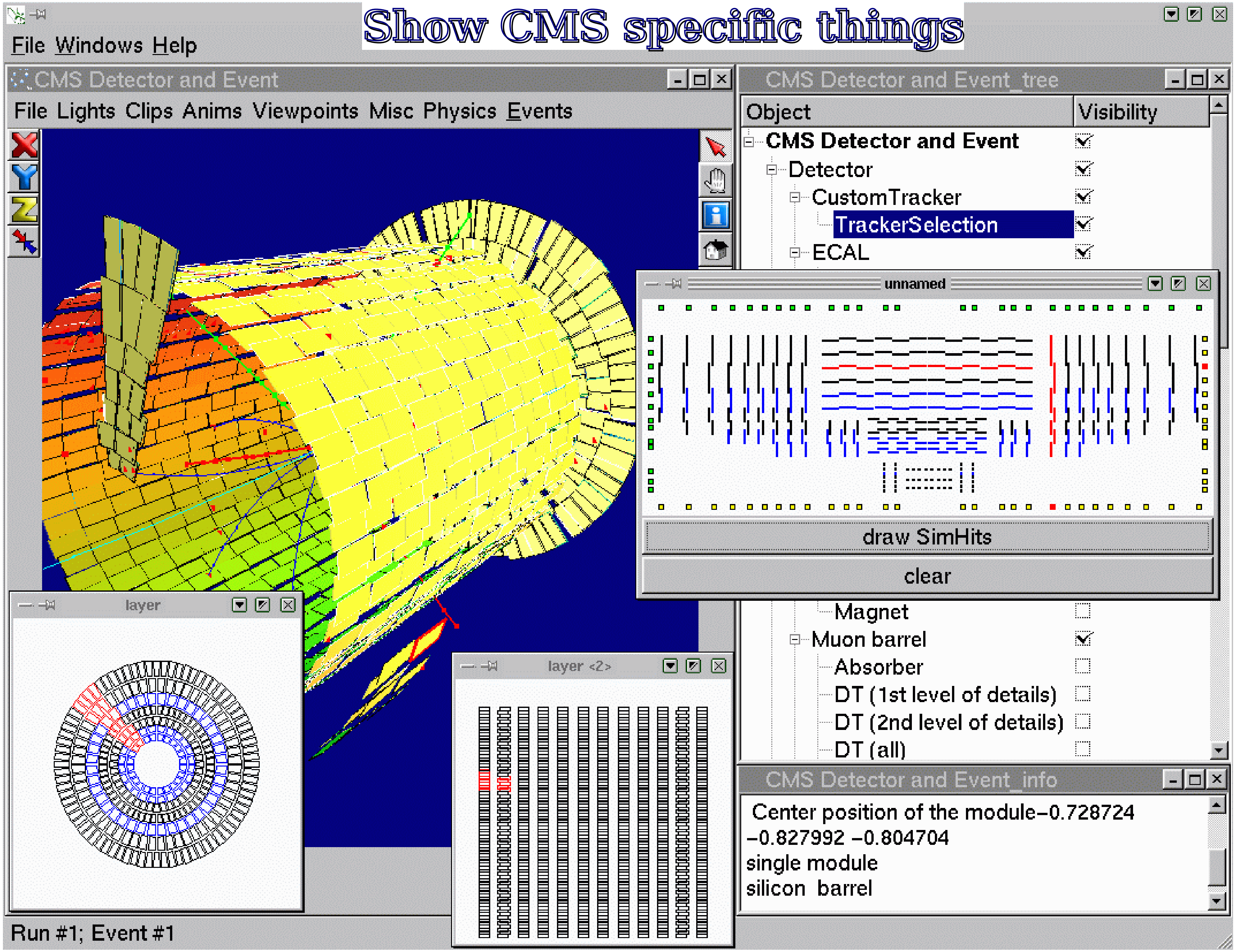
Geometry browsing


Click here
Update there
Click here
Update there
OSCAR Visualisation
File Sessions Options Help
File 3D View Debug
Visi
ForwardPixelTurbinBlad...
ForwardPixelTurbinBlad...
ForwardPixelDetectorAss...
ForwardPixelDetectorAss...
ForwardPixelDetectorAss...
ForwardPixelDetectorAss...
ForwardPixelDetectorAss...
ForwardPixelDetectorAss...
ForwardPixelDetector
ForwardPixelDetector...
ForwardPixelDetectorAss...
ForwardPixelPortCard
ForwardPixelServiceTube
ackwardPixel
Rotx Roty
Dolly
Seek To
Slice
Clip

```
Element: Parameter (57)
Element: PartSelector (1591)
Element: Polycone (106)
Element: Polyhedra (36)
Element: PosPart (15563)
Element: PosPartSection (24)
Element: ReflectionSolid (220)
Element: Rotation (3900)
Element: RotationSection (1)
Element: SolidSection (24)
Element: SpecPar (57)
Element: SpecParSection (1)
Element: Translation (13895)
Element: Trapezoid (337)
Element: Trd1 (230)
Element: Tubs (528)
Element: ZSection (619)
Element: rChild (15563)
Element: rMaterial (3332)
Element: rParent (15563)
Element: rRotation (11747)
Element: rSolid (2280)
DDCore: start comprehensive ch
[DDCore:Report] Materials
 271 Materials declared
 detected errors:0

Checking connectivity of Compa
 CompactView has 2060 (multipl

 There were 0 unconnected node

DDCore: end of comprehensive c
--warning from G4Material-- Th
 density  = 1e-22 mg/cm3
 rho*(T/P) would be of the ord
 be affected
 MakePLanes 0.000178687 0.0001
--warning from G4Material-- Th
 density  = 7 mg/cm3      pressure = 1 atmosphere        temperature = 273.15 kelvin
 rho*(T/P) would be of the order of: 81.3382 (mg/cm3)*(kelvin/atmosphere). The energy loss calculation maybe
 be affected
***------------------------------***
* Detector has been built from DDD! *
***==============================***
Idle>
```

Name	ForwardPixelDetector
Copy #	31
Replicated	No
Position	(-4.05, -0.125, -16.2)
Rotation	(None)
Material	Pix_Fwd_Det
Density	2.33
Sensitive	(not sensitive)

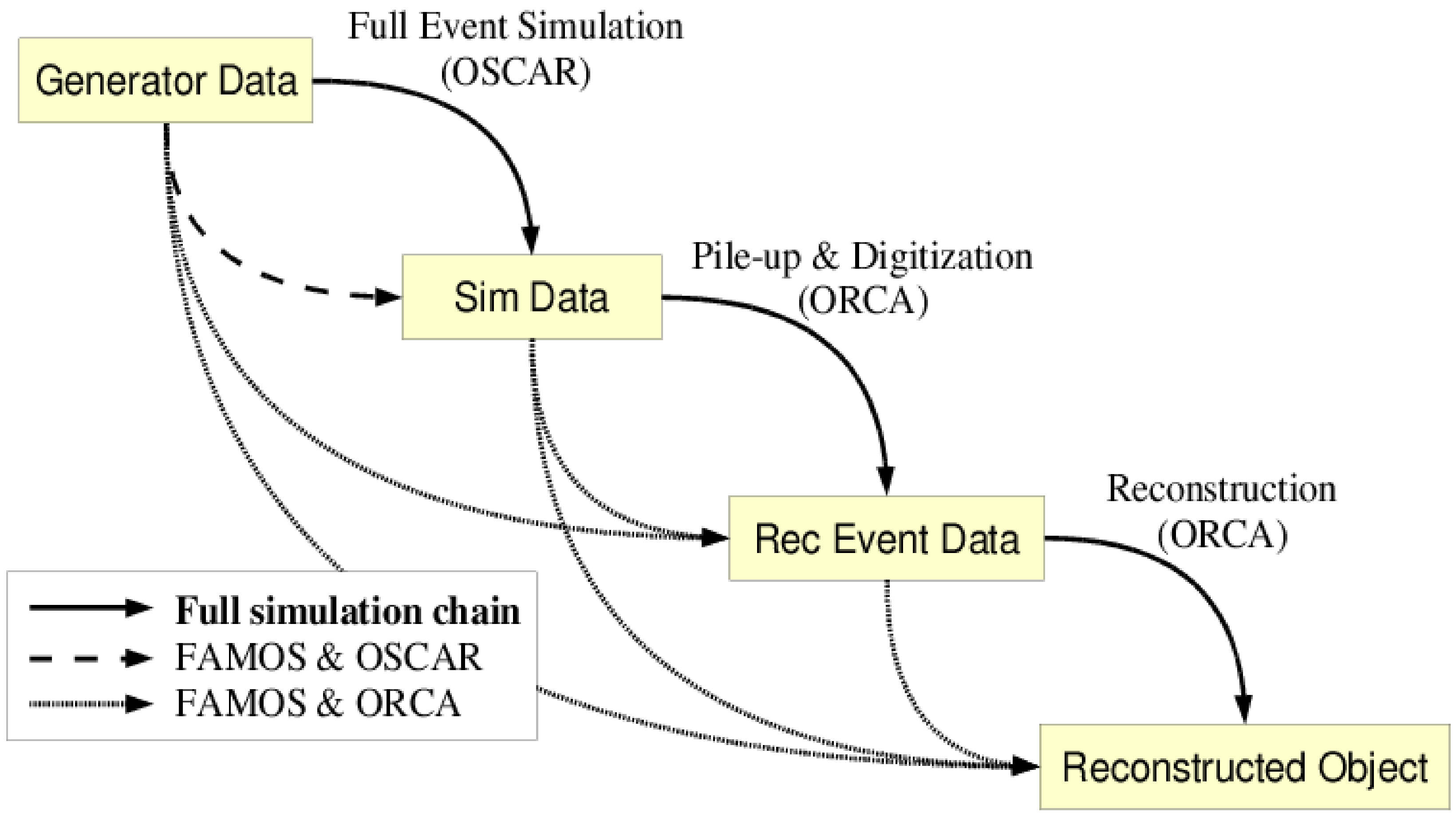
Show CMS specific things



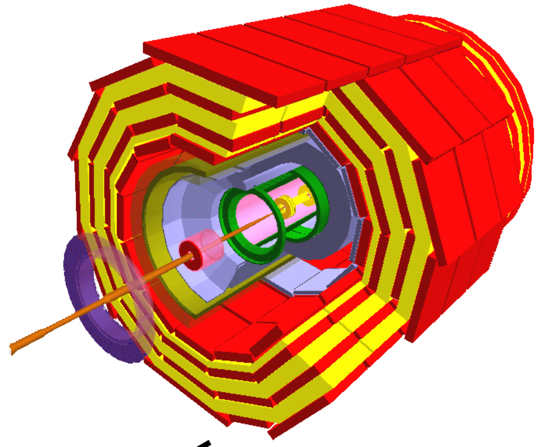
Center position of the module–0.728724
–0.827992 –0.804704
single module
silicon barrel

- ✔ CMS software for Simulation and Reconstruction
- ✔ Detector Description
- ✔ Visualization
- ➔ Simulation example
- Framework
- User Interface for batch processing
- CMS Computing Infrastructure

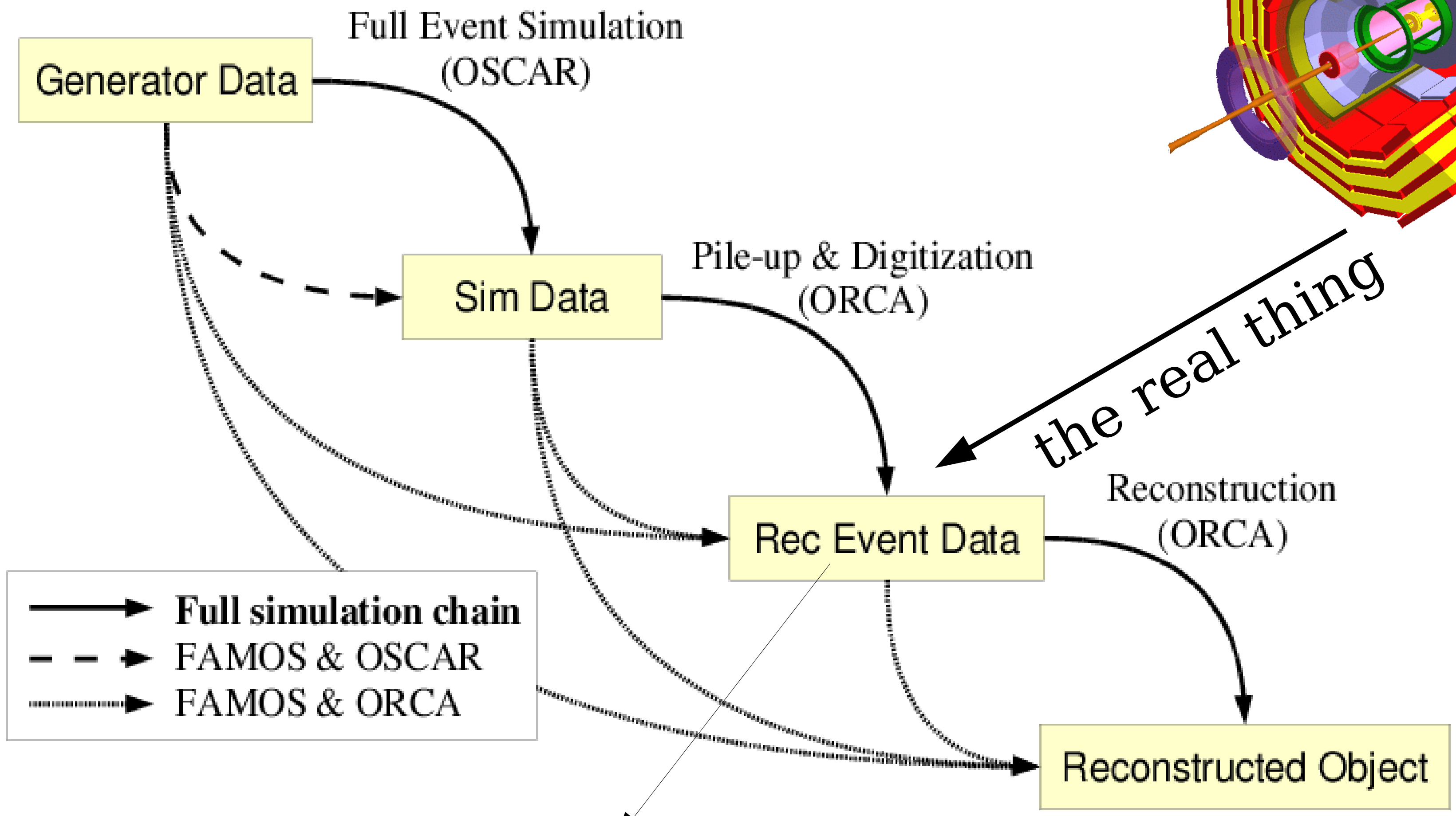
Simulation Plug&Play



Simulation Plug&Play

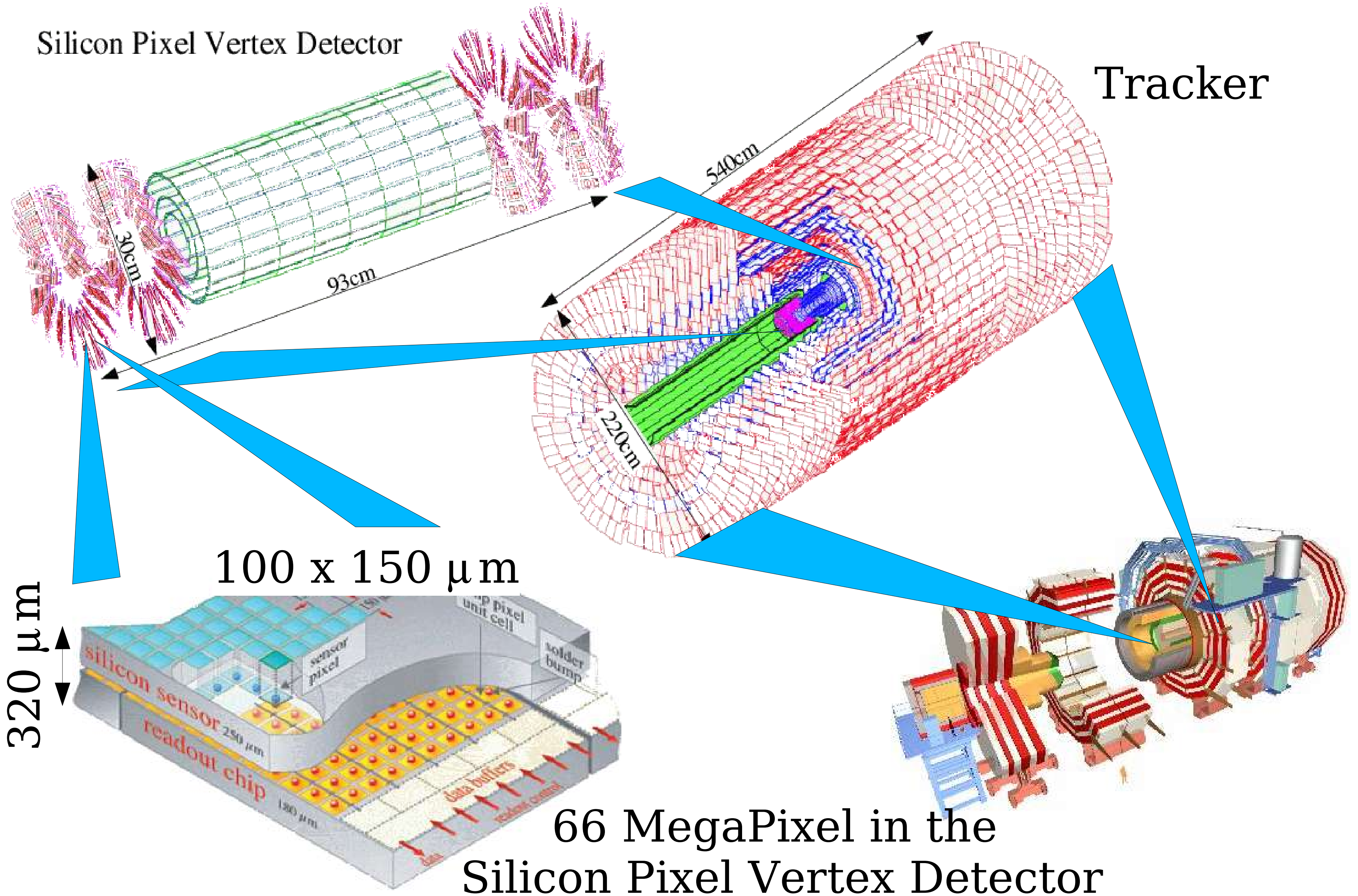


the real thing



Digitized data & reconstructed hits

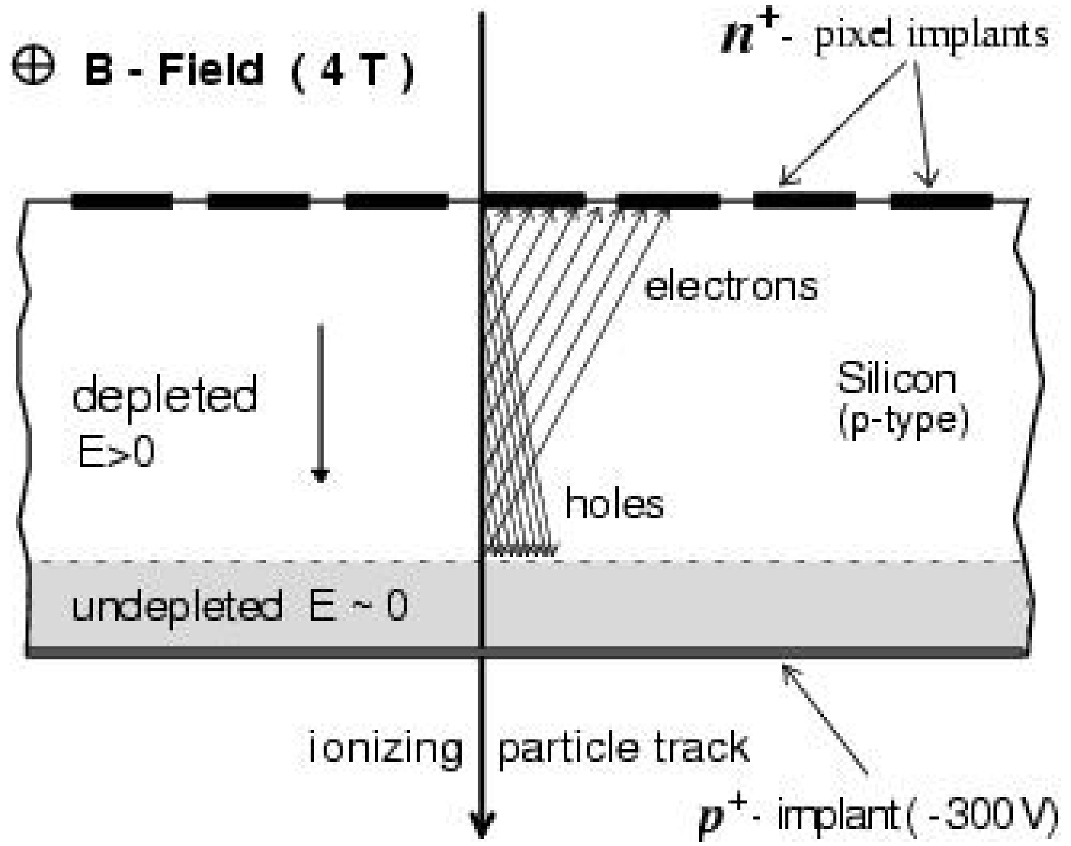
Example: CMS Pixel Detector



66 MegaPixel in the
Silicon Pixel Vertex Detector

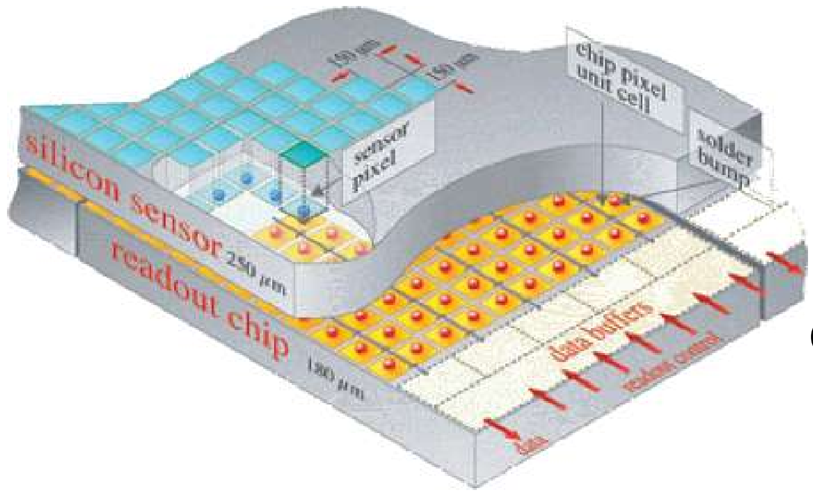
Measurement Process

Ionizing particles create e-,hole pairs in the silicon.
Drifting of (e-,hole) pairs due to electrostatic field
Deflection due to magnetostatic field (Lorentz angle ..)



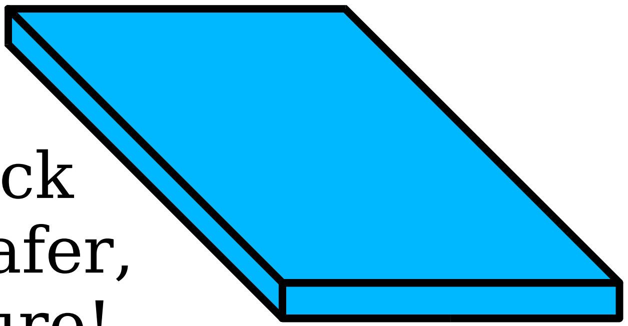
Collected charge of e- triggers an electric signal that is converted to ADC counts.

Geant4 Simulation



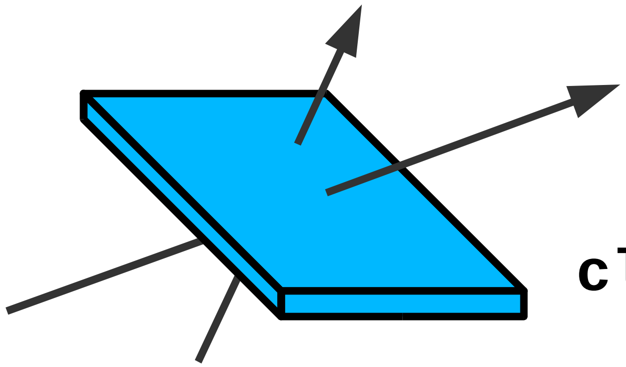
simplified!

one single block
of silicon per wafer,
no pixel structure!



```
<LogicalPart name="PXBD">
  <Box name="PXBD" dx="8.1*mm" dy="150*mum" dz="3.225*cm"/>
  <rMaterial name="Silicon"/>
</LogicalPart>

<ElementaryMaterial name="Silicon" density="2.33*g/cm3"
                   symbol=" " atomicWeight="28.09*g/mole"
                   atomicNumber="14"/>

<SpecPar name="ROUHitsPixelBarrel">
  <PartSelector path="//PXBD" />
  <Parameter name="SensitiveDetector"
             value="TkAccumulatingSensitiveDetector" />
</SpecPar>
```

SimHit in OSCAR (Geant4)



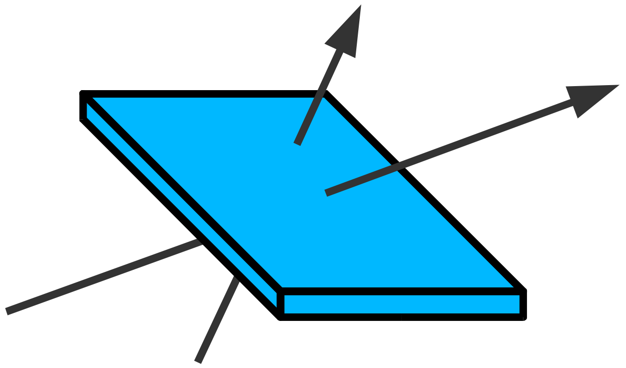
red: directly from Geant4
blue: indirectly from Geant4

```
class SensitiveDetector :
   public G4VSensitiveDetector;

class BeginOfEvent { G4Event* evt_;
   ... // CMS specific stuff};

class PSimHit;

class TkAccumulatingSensitiveDetector :
   public SensitiveDetector,
   private Observer<const BeginOfEvent *>
{  bool ProcessHits(...);
   void EndOfEvent(...);
   void upDate(BeginOfEvent *);
   PSimHit hit_;
};
```

Note:
upDate(..) is called from the COBRA framework. COBRA implements an Event-UserAction which converts a G4Event into a CMS BeginOfEvent and EndOfEvent object.

pure G4 would be:
Initialize(...)

SimHit in OSCAR (Geant4)



`class PSimHit;`

PSimHit does not depend on Geant4

It's stored via POOL (PSimHit = PersistentSimHit).

TkAccumulatingSensitiveDetector creates PSimHits

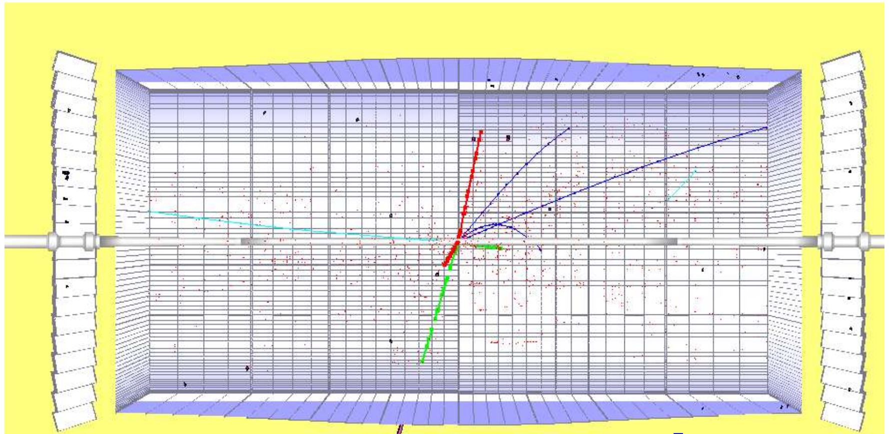
Following information is extracted from Geant4 to a PSimHit:

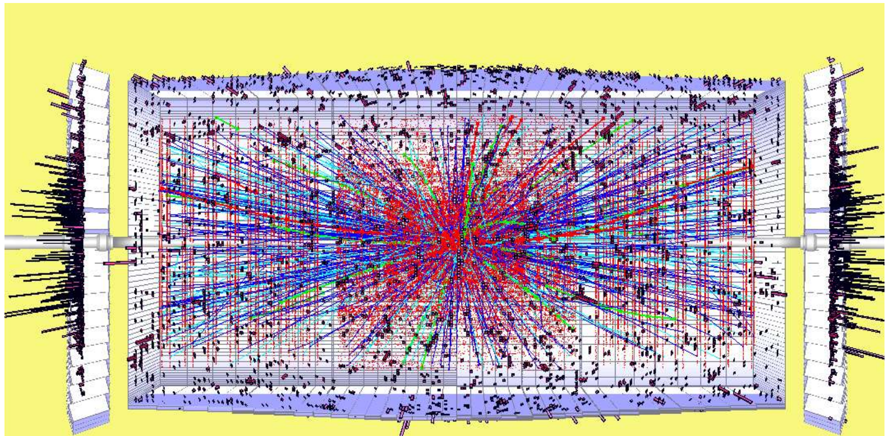
- unit ID (ID=func(G4VTouchable)) = ID of the silicon wafer not the individual pixel!!
- local entry and exit points of the track (from G4Track or G4Step)
- time of flight since primary interaction (from G4Track)
- energy loss (from G4Step)
- azimuth & polar angle at entry point (G4VTouchable)
- particle type, process type that created the track (G4Track)
- momentum information (G4Track)

No (explicit) information about individual pixel stored!

Pile-up / Digitization

- Detector digitization has to take into account multiple LHC bunch crossings (one crossing every 25 ns!)
  - in each bunch crossing there are ~17 collision events!
  - Pileup: @10^{34} = 17 minimum bias events/crossing
    - Calorimetry needs -5 to +3 crossings
    - Muon DT ought to have +- 10 crossings
    - Tracker loopers can persist for many crossings
- Study at different luminosities infers different pileup
- Impossible to simulate every event as described above

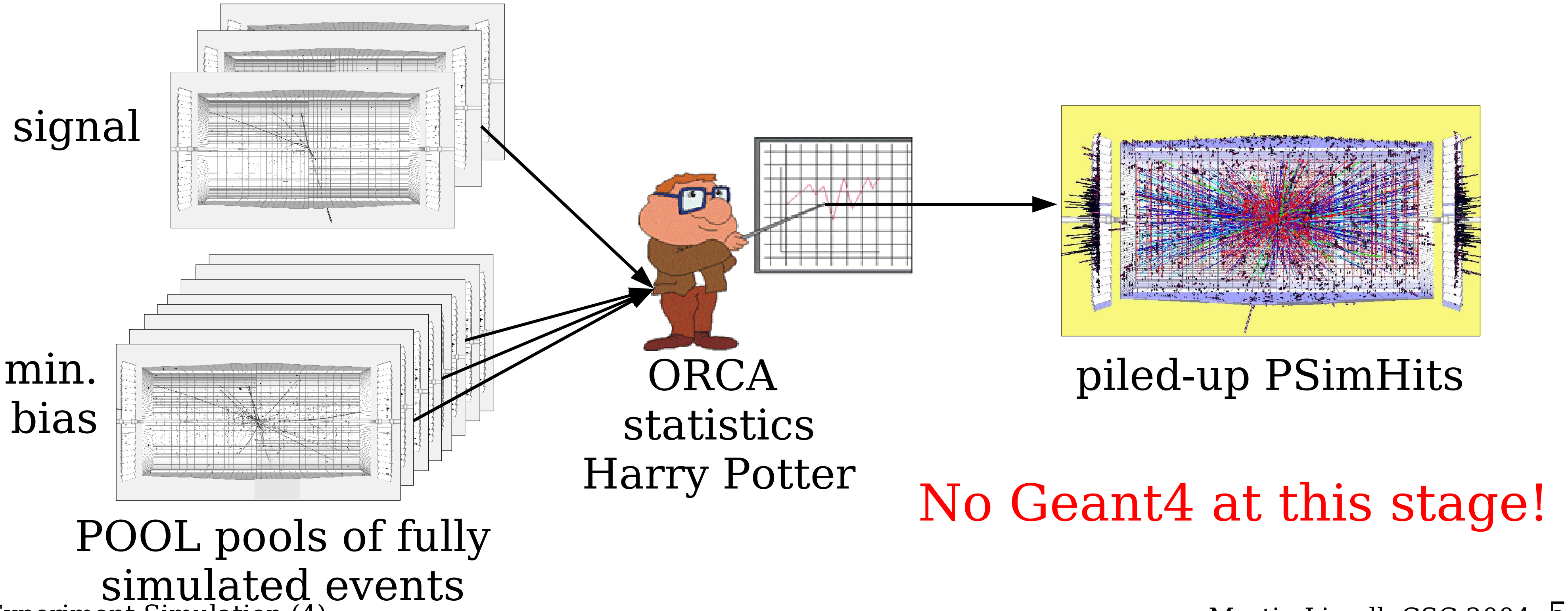

one signal event


+minimum bias of one bunch crossing

Pile-up / Digitization

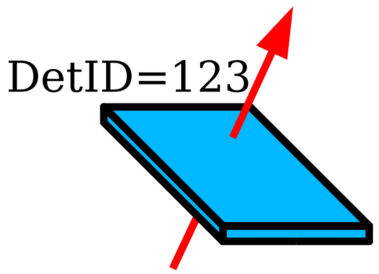
Strategy:

- fully Geant4-simulate a pool of different minimum bias event (done with OSCAR)
- fully Geant4-simulate a pool of different signal events (done with OSCAR)
- statistically mix signal & minimal bias PSimHits to represent pile-up events (done with ORCA, no Geant4)

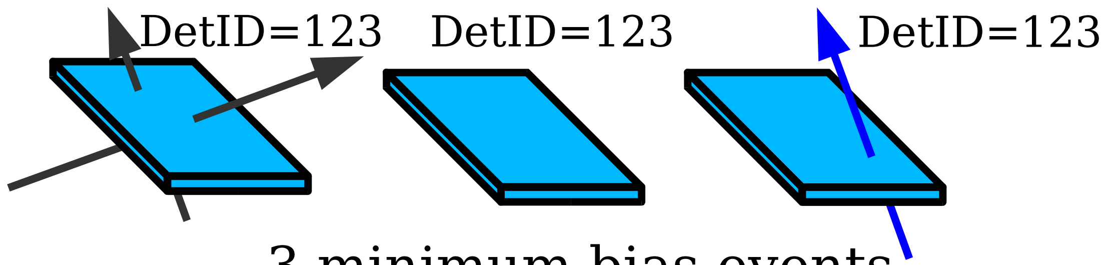

signal
min. bias
POOL pools of fully simulated events
ORCA statistics Harry Potter
piled-up PSimHits

No Geant4 at this stage!

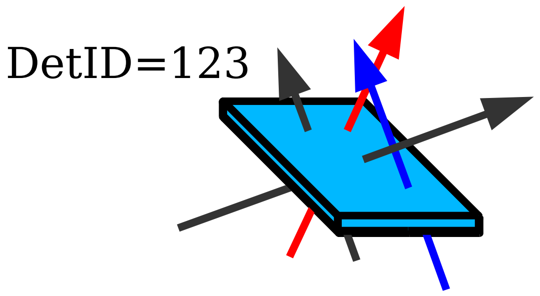
Pile-up & Digitization for Pixel Detector



signal event
in one DetUnit

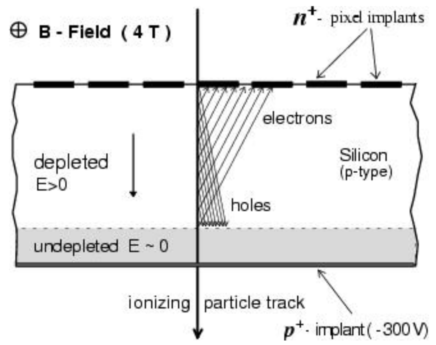


3 minimum bias events
for the same DetUnit



piled-up: DetUnit 123 contains 4 PSimHits

Foreach PSimHit in DetUnit do:



calculate the
charge collection
including the
Lorentz drift

Then create PixelDigis by converting the collected charge into ADC counts for each affected pixel (not forgetting about noise, dead pixels, latencies, individual gains, ..)

Pile-up & Digitization for Pixel Detector

```
class PSimHit { public:
  ParticleType particle() const;
  LocalPoint entryPoint() const;
  LocalPoint exitPoint() const;
  int detID() const;
  ... };
```

OSCAR (Geant4)

$\downarrow$

ORCA

```
class DetUnit { public:
  int id() const;
  const vector<PixelDigi>& digis() const;
  ... };

class PixelDigi { public:
 int row() const; // row of the pixel
 int column() const; // column of the pixel
 int adc() const; // adc counts of the pixel
 ... };
```

Summary: Pixel-Simulation

1) Event Generation (Pythia,..; min.bias, signal)
2) PSimHit (OSCAR), pools of min. bias & signal
3) Pile-up mixing (ORCA)
4) DetUnit has many PixelDigis (ORCA)

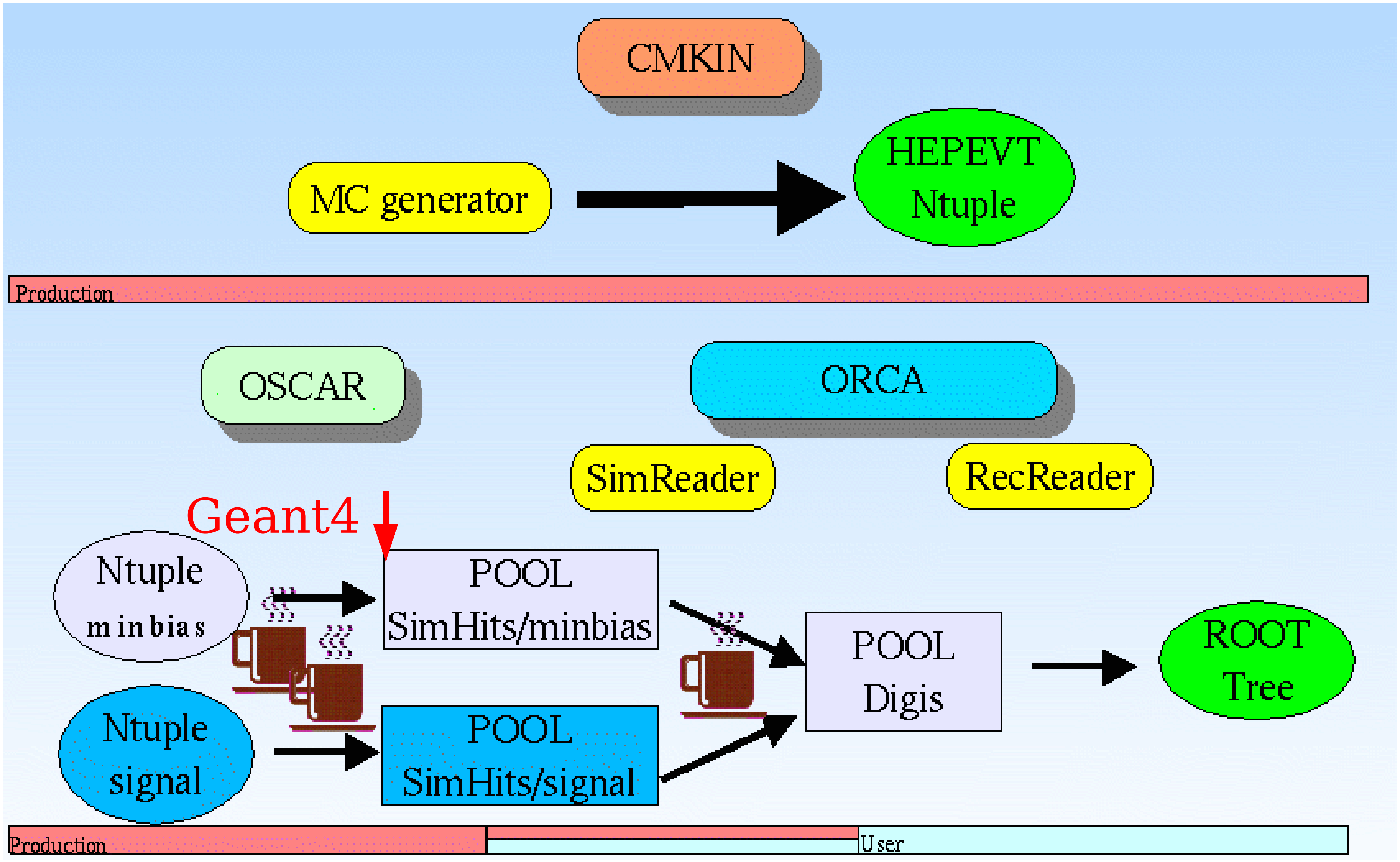
Within the 66 MegaPixel, the Reconstruction
now must find PixelDigi-clusters,
and from them reconstruct the
particle impact information

RECO(many,many PixelDigis) -> many RecHits

RecHits are "statistical", i.e. they have errors assigned.

Check with PSimHit the quality of RecHits
(or check higher level reconstructed objects - RecTracks)

Simulation cycle

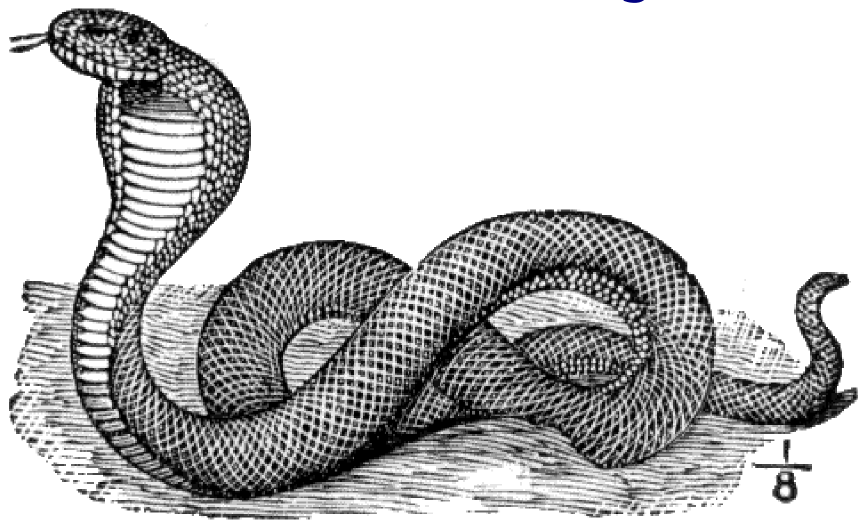


Most of the simulation work is done by CMS production teams in world wide distributed production centers

- ✔ CMS software for Simulation and Reconstruction
- ✔ Detector Description
- ✔ Visualization
- ✔ Simulation example
- ➔ Framework
- • User Interface for batch processing
- • CMS Computing Infrastructure

Why an own framework?

- Geant4 has all what is needed to build a simulation application:
  - initialization callbacks for setting the geometry, physics
  - user actions for extracting hit information, performing pile-up and digitization calculations
  - extensible command-line user-interface
  - several graphics system for visualization
  - Why don't we build our experiment simulation using Geant4 as “framework”



Cobra, p. 426.

Why an own framework?

- Decoupling of Geant4 based simulation, fast simulation, real data in a transparent way for reconstruction & analysis
  - want define the "event loop" to be independent of simulation libraries
- CMS framework is a notification/observer framework
  - registration of "call-backs" easier & more flexible compared to Geant4 ...
  - in simulation: often you want more (meta-data) than only a Geant4 object like G4Event* "dispatched" to your user code
  - easier to integrate new application modules at runtime
    - loading shared libs on demand
    - integration to scripting environments (especially for analysis)
- Services used by all applications are implemented in one place
  - persistency service, metadata services
  - common configuration management, user interface (setting parameters in the same way for all framework based applications)

- ✔ CMS software for Simulation and Reconstruction
- ✔ Detector Description
- ✔ Visualization
- ✔ Simulation example
- ✔ Framework
- ➔ User Interface for batch processing
- • CMS Computing Infrastructure

User-Interface (batch)

Geant4 provides an extensible (pseudo) macro language to pass arguments to a simulation.
e.g.: **`/run/beamOn(20)`** … simulate 20 event, built-in cmd.
`/Tracker/cuts/gamma 100 MeV` … user cmd.

You need to write quite some G4-based code to add a new command (messenger classes, …)

CMS wants to have the same interface for all its applications. And, it should be easy to add new parameters!

Code:

default

```
SimpleConfigurable<bool> noise(true,"PixelDigitizer:AddNoisyPixels");
 bool addNoisyPixels = noise;      // Add noisy pixels
```

Configuration file:

```
PixelDigitizer:Paramter1 = 'foo'
PixelDigitizer:AddNoisyPixels = false
ECal:Parameter2 = 3.141592
```

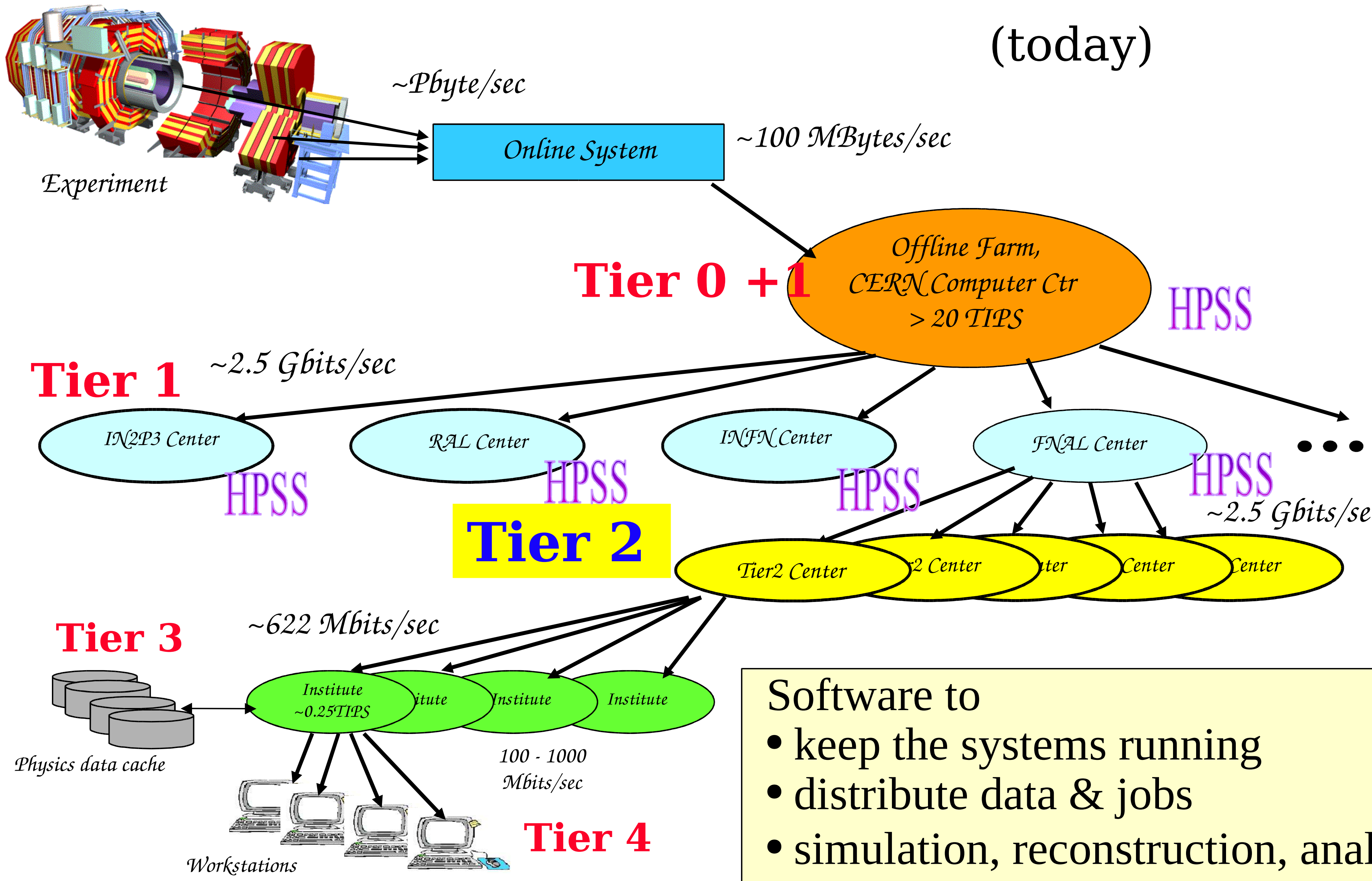
- ✔ CMS software for Simulation and Reconstruction
- ✔ Detector Description
- ✔ Visualization
- ✔ Simulation example
- ✔ Framework
- ✔ User Interface for batch processing
- ➔ CMS Computing Infrastructure

“Production” in CMS

- Production is the process to produce large set of simulated data to study
  - the detector performance
  - issues related to data management (“data challenges”) in a world-wide collaboration
    - storing, retrieving, navigating data
    - distributing computing jobs to computing centers
    - testbed for Grid tools
  - test, benchmark, and verify reconstruction algorithms
  - ...

CMS Computing Model

(today)



Software to
- keep the systems running
- distribute data & jobs
- simulation, reconstruction, analysis