



This course



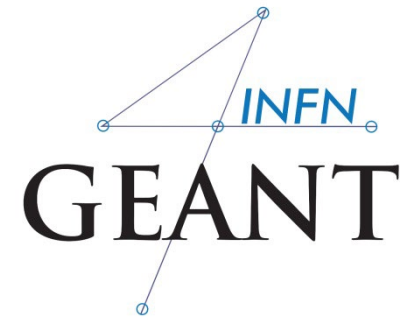
Structure and logistics - 1

- This course is organized in a mixture of **theoretical lectures** and practical **hands-on sessions**
 - The hands-on sessions require **real C++ coding** to build up a **simplified Geant4 application**
 - **Staged approach** in tasks
 - **`http://geant4.lns.infn.it/alghero2022/introduction`**
- A **pre-installed virtual machine** is provided for the hands-on sessions
 - Includes **Geant4 11.0.p01** on a Linux environment
 - You should already have it downloaded and tested
 - Please **let us know** ASAP if you have problems with the VM



Structure and logistics - 2

- You can **try to install** Geant4 on your (Linux/Mac) laptop, if you wish
 - The course is **not meant** to show that, though
- All **lectures (pdf) will be uploaded** on-the-fly on the course indico page
 - <https://agenda.infn.it/e/AlgheroSeminar2022>
 - Please **feel free to ask any question**, either during the **lectures** , during the **exercises** or during the **breaks**
- **Solutions** of the exercises **will be uploaded** after the end of each exercise session



Monte Carlo techniques and

GEANT4 concept

Luciano Pandola

INFN – Laboratori Nazionali del Sud

Geant4 Course, XIX Seminar on Software for Nuclear,
Subnuclear and Applied Physics,
Alghero, June 6th- 10th, 2022

What Monte Carlo (MC) techniques are for?



- Numerical solution of a (**complex**) macroscopic problem, by simulating the microscopic interactions among the components
- Uses random sampling, until convergence is achieved
 - Name after Monte Carlo's casino
- Applications not only in physics and science, but also finances, traffic flow, social studies
 - And not only problems that are intrinsically probabilistic (e.g. numerical integration)

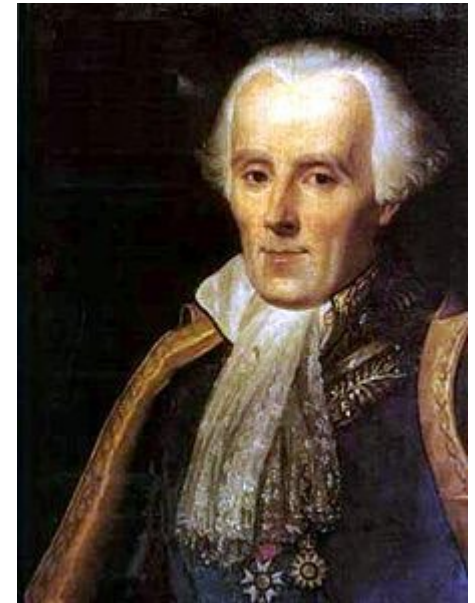


MC in science

- In physics, **elementary laws** are (typically) **known** → MC is used to **predict the outcome** of a (complex) experiment
 - **Exact** calculation from the basic laws is **unpractical**
 - **Optimize** an **experimental setup**, support **data analysis**
- In this course: Monte Carlo for **particle tracking** (interaction of radiation with matter)
- Usually the **Monte Carlo wins** over the exact (mathematical) solution for **complex** problems

A bit of history

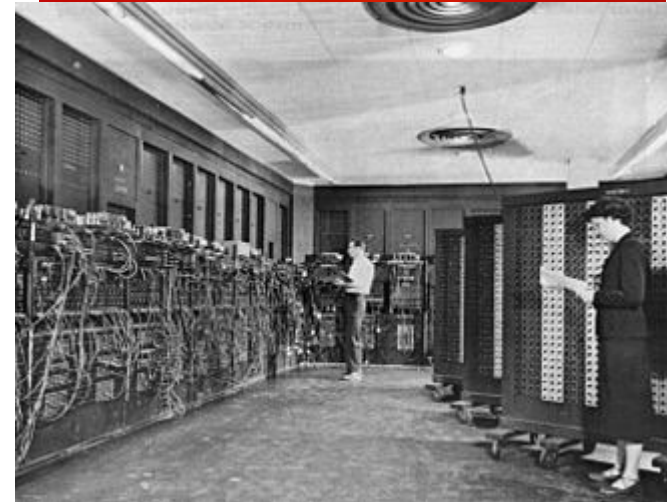
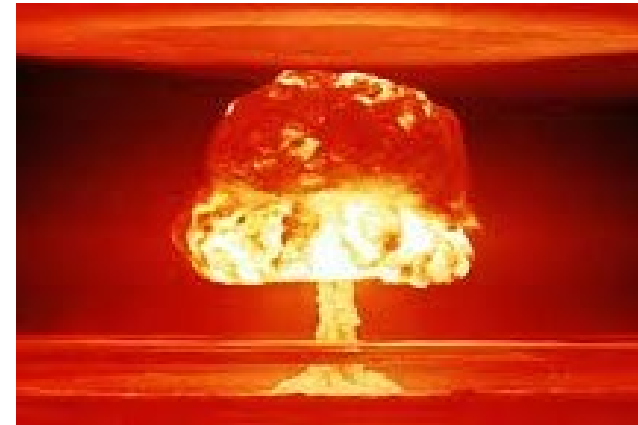
- Very **concept** of Monte Carlo comes in the **XVIII century** (Buffon, 1777, and then Laplace, 1786)
 - Monte Carlo **estimate of π**
- Concept of MC **is much older** than real **computers**
 - one can also implement the **algorithms manually**, with dice (= Random Number Generator)



A bit of history



- Boost in the '50 (Ulam and Von Neumann) for the development of thermonuclear weapons
- Von Neumann invented the name "Monte Carlo" and settled a number of basic theorems
- First (proto)computers available at that time
 - MC mainly **CPU load**, minimal I/O



A bit of history

JOURNAL OF THE AMERICAN STATISTICAL ASSOCIATION

Number 47

SEPTEMBER 1949

Volume 44

THE MONTE CARLO METHOD

NICHOLAS METROPOLIS AND S. ULAM
Los Alamos Laboratory

THE JOURNAL OF CHEMICAL PHYSICS

VOLUME 21, NUMBER 6

JUNE, 1953

Equation of State Calculations by Fast Computing Machines

NICHOLAS METROPOLIS, ARIANNA W. ROSENBLUTH, MARSHALL N. ROSENBLUTH, AND AUGUSTIA H. TELLER,
Los Alamos Scientific Laboratory, Los Alamos, New Mexico

AND

EDWARD TELLER,* *Department of Physics, University of Chicago, Chicago, Illinois*
(Received March 6, 1953)

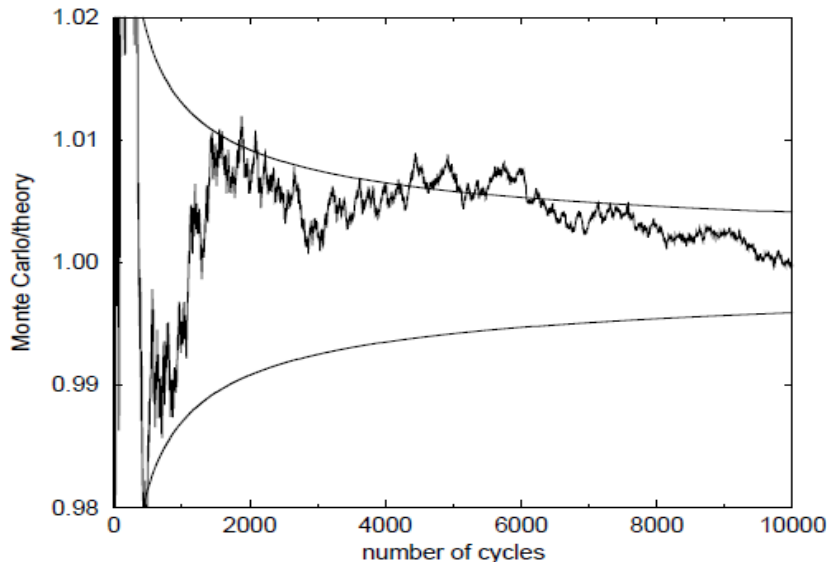
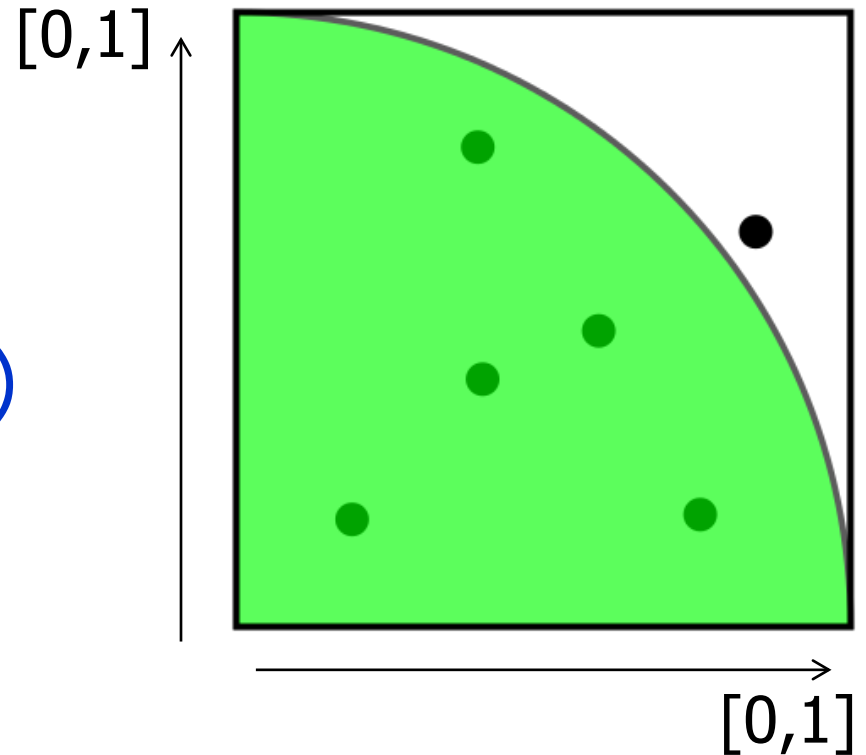


Nick Metropolis enjoying a break in the quantum Monte Carlo conference, Spring 1953.

With MANIAC: the first
electronic digital
computer

The simplest MC application: numerical estimate of π

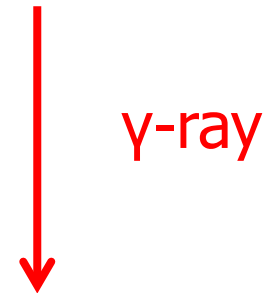
- Shoot N couples (x,y) randomly in $[0,1]$
- Count n : how many couples satisfy $(x^2+y^2 \leq 1)$



- $n/N = \pi/4$ (ratio of areas)
- Convergence as $1/\sqrt{N}$

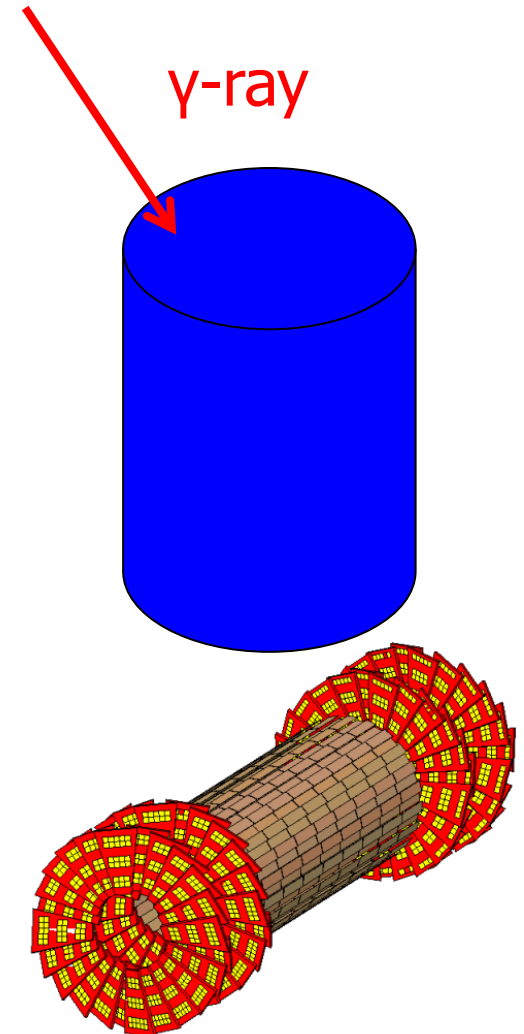
Most common application in particle physics: particle tracking

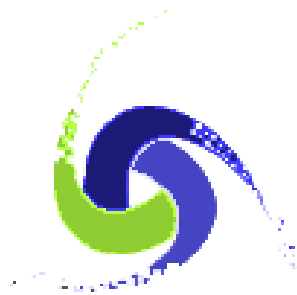
- Problem: track a γ -ray in a semi-infinite detector and determine the energy spectrum deposited
 - Still, a **model case**
- All physics is known from textbook (Compton scattering, photoelectric effect, etc.)
- Yet, the analytical calculation is a **nightmare** (while still possible)



Most common application in particle physics: particle tracking

- Problem v2: track a γ -ray in a finite detector (e.g. a NaI)
 - **Real-life** (simplified) case
- Analytical computation nearly impossible
 - **Monte Carlo** clearly **wins**
- Now make the detector **more complicated**, as in modern physics





GEANT4

A SIMULATION TOOLKIT



S. Agostinelli et al., Nucl. Instr. Meth. A **506** (2003) 250

J. Allison et al., IEEE Trans. Nucl. Sci. **53** (2006) 270

J. Allison et al., Nucl. Instr. Meth. A **835** (2016) 186

What is



- Toolkit for the **Monte Carlo simulation** of the interaction of **particles** with **matter**
 - **physics processes** (EM, hadronic, optical) cover a **comprehensive set** of particles, materials and over a wide energy range
 - offers a complete set of **support functionalities** (tracking, geometry)
 - **Distributed** software production and management: developed by an **international Collaboration**
 - Established in 1998
 - Approximately **100 members**, from Europe, America and Asia
- Written in **C++** language
 - Takes advantage from the **Object Oriented** software technology
- **Open source**

<http://geant4.org>

Overview

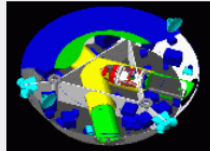
Geant4 is a toolkit for the simulation of the passage of particles through matter. Its areas of application include high energy, nuclear and accelerator physics, as well as studies in medical and space science. The three main reference papers for Geant4 are published in Nuclear Instruments and Methods in Physics Research [A 506 \(2003\) 250-303](#), IEEE Transactions on Nuclear Science [53 No. 1 \(2006\) 270-278](#) and Nuclear Instruments and Methods in Physics Research [A 835 \(2016\) 186-225](#).

Applications



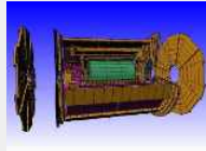
A sampling of applications, technology transfer and other uses of Geant4

User Support



Getting started, guides and information for users and developers

Publications



Validation of Geant4, results from experiments and publications

Collaboration



Who we are: collaborating institutions, [members](#), organization and legal information

Events

[Virtual] [Geant4 Beginners Course @ CERN](#), CERN, Geneva (Switzerland), **9-13 May 2022**.

[Geant4 Course at the 19th Seminar on Software for Nuclear, Sub-nuclear and Applied Physics](#), Porto Conte, Alghero (Italy), **5-10 June 2022**.

27th Geant4 Collaboration Meeting, IRISA Laboratory, Rennes (France), **26-30 September 2022**.

[Past events](#)

News

2022-03-23

[2022 planned developments](#).

2022-03-08

Patch-01 to release 11.0 is available from the [Download area](#).

2021-11-19

Patch-03 to release 10.7 is available from the [Download archive area](#).

<https://geant4.org>

- **Code and documentation** available in the main **web page**
- Regular **tutorial courses** held worldwide



GEANT4 versions and releases

A SIMULATION TOOLKIT

- First release (Geant4 1.0) in December 1998
 - ~Two releases per year since then
 - Major releases (**x.y**) or minor releases (x.**y**) or beta releases
 - Patches regularly issued
- Last version: **Geant4 11.0**
 - Released December 10th, 2021
 - Now patch **11.0.p02** (May 26th, 2022)
 - The VM used for this course has Geant4 **11.0.p01**
- Requires **C++11** since **10.2** (gcc > 4.8.x)
 - **Native** C+11 features in-place (= compilation with old compilers **fails**)



Basic concept of Geant4



Toolkit and User Application

- Geant4 is a **toolkit** (= a collection of tools)
 - i.e. you **cannot** “run” it out of the box
 - You must **write an application**, which **uses** Geant4 tools
- Consequences:
 - There are no such concepts as “Geant4 defaults”
 - You must provide the **necessary information** to configure your simulation
 - You must deliberately **choose** which **Geant4 tools** to use
- Guidance: many **examples** are provided



Basic concepts

- What you **MUST** do:
 - Describe your **experimental set-up**
 - Provide the **primary particles** input to your simulation
 - Decide which **particles** and **physics models** you want to use out of those available in Geant4 and the precision of your simulation (cuts to produce and track secondary particles)
- You **may also want**
 - To interact with Geant4 kernel to **control** your simulation
 - To **visualise** your simulation configuration or results
 - To produce **histograms, tuples** etc. to be further analysed



Main Geant4 capabilities

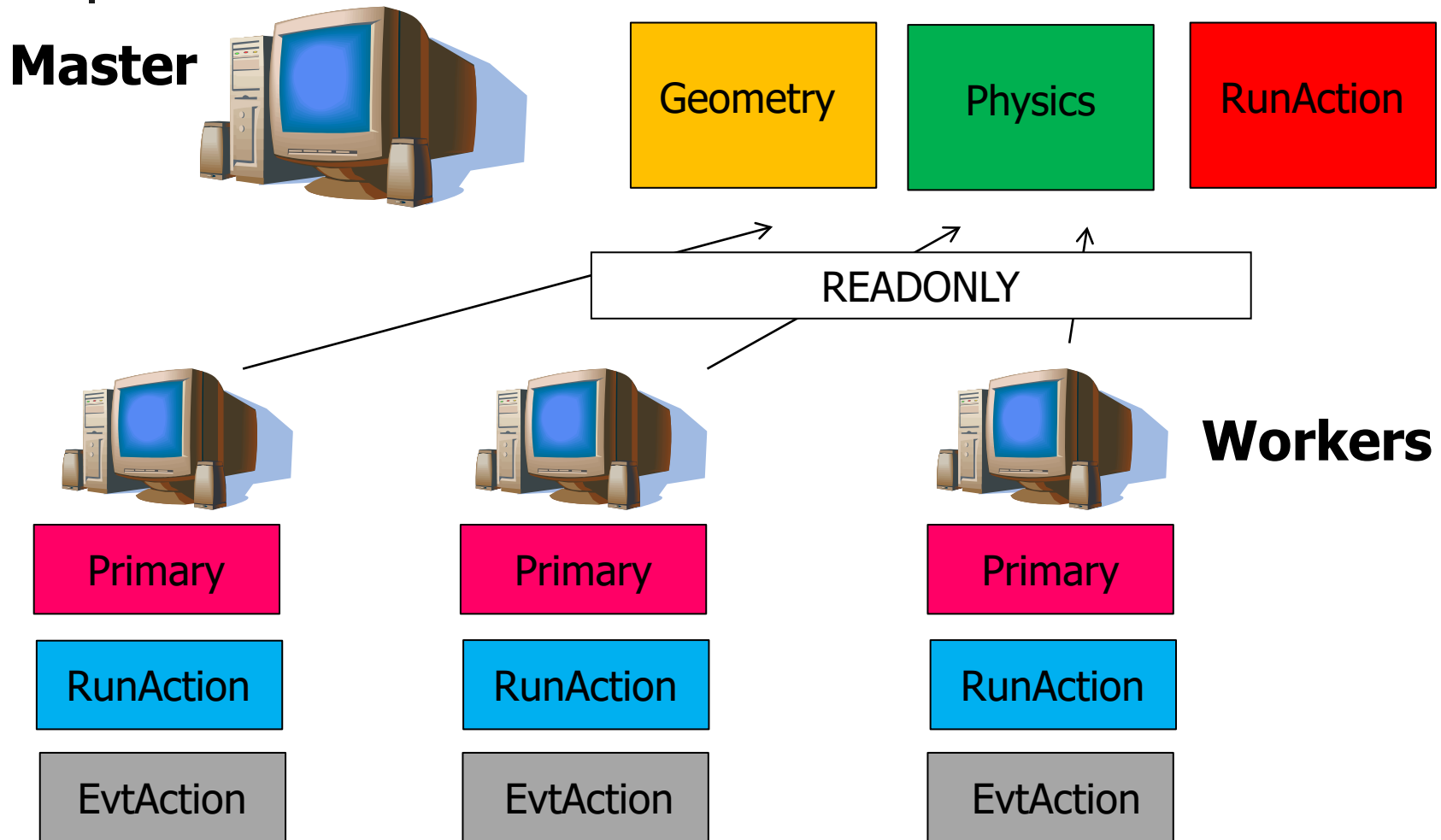
- **Transportation of a particle 'step-by-step'** taking into account all possible interactions with materials and fields
- **The transport ends** if the particle
 - is slowed down to **zero kinetic energy** (and it doesn't have any interaction at rest)
 - **disappears** in some interaction
 - reaches the **end of the simulation volume**
- Geant4 allows the User to **access** the transportation process and **retrieve** the results (**USER ACTIONS**)
 - at the **beginning** and **end** of the **transport**
 - at the **end** of **each step** in transportation
 - if a **particle** reaches a **sensitive detector**
 - Others...



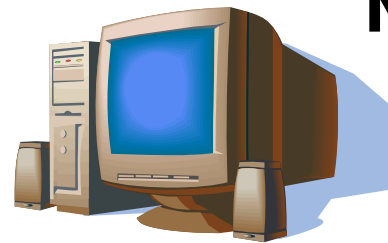
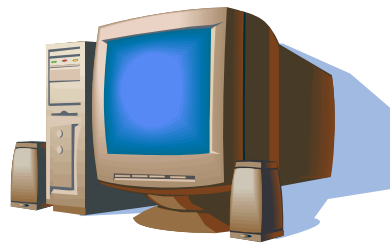
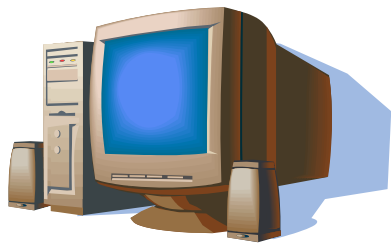
Multi-thread mode

- Geant4 10.0 (released Dec, 2013) supports **multi-thread approach** for multi-core machines
 - Simulation is automatically **split** on an **event-by-event** basis
 - different events are processed by different cores
 - Can fully **profit** of **all cores** available on **modern machines** → substantial **speed-up** of simulations
 - **Unique** copy (master) of **geometry** and **physics**
 - All cores have them as read-only (saves memory)
- **Backwards compatible** with the **sequential** mode
 - The MT programming requires some **care**: need to **avoid conflicts** between threads
 - Some modification and porting required

Concept for multi-thread ...



... vs. parallelisation



Nodes

Geometry

Geometry

Geometry

Physics

Physics

Physics

Primary

Primary

Primary

RunAction

RunAction

RunAction

EvtAction

EvtAction

EvtAction

- Each node hosts a **complete** simulation
- **Many copies** of geometry and physics tables
- More **memory-thrifty**

Interaction with the Geant4 kernel - 1



- Geant4 design provides **tools** for a user application
 - To tell the **kernel** about your simulation **configuration**
 - To **interact** with Geant4 kernel itself
- Geant4 tools for user interaction are **base classes**
 - You create **your own concrete class** derived from the base classes → **interface** to the Geant4 kernel
 - Geant4 kernel handles your own derived classes **transparently** through their base class interface (polymorphism)

Interaction with the Geant4 kernel - 2



Two types of Geant4 base classes:

- **Abstract base classes** for user interaction (classes starting with G4V)
 - User derived concrete classes are **mandatory**
 - User to implement the purely virtual methods
- **Concrete base classes** (with **virtual** dummy default methods) for user interaction
 - User derived classes are **optional**



User Classes

Initialisation classes

Invoked at the initialization

- G4VUserDetectorConstruction
- G4VUserPhysicsList

Global: **only one instance** of them exists in memory, shared by all threads (**readonly**). Managed only by the **master** thread.

Action classes

Invoked during the execution loop

- G4VUserActionInitialization
 - G4VUserPrimaryGeneratorAction
 - G4UserRunAction (*)
 - G4UserEventAction
 - G4UserTrackingAction
 - G4UserStackingAction
 - G4UserSteppingAction

Local: an **instance** of each action class exists **for each thread**.
(*) Two RunAction's allowed: one for master and one for threads



The mandatory user classes

Mandatory classes in ANY Geant4 User Application

- **G4VUserDetectorConstruction**
describe the experimental set-up
- **G4VUserPhysicsList**
select the physics you want to activate
- **G4VUserActionInitialization**
takes care of the user initializations
G4VUserPrimaryGeneratorAction

Will be described in detail in the next lectures
(Tue-Wed)



Optional user classes

- Five **concrete base classes** whose **virtual member functions** the user may override to gain **control** of the simulation at various stages
 - G4User**Run**Action
 - G4User**Event**Action
 - G4User**Tracking**Action
 - G4User**Stacking**Action
 - G4User**Stepping**Action
- ← e.g. actions to be done at the **beginning** and **end** of each event
- Each member function of the base classes has a **dummy implementation** (not purely virtual)
 - Empty implementation: **does nothing**
 - Override only the methods that you need
 - User action classes must be **registered** to the Run Manager via the **G4VUserActionInitialization**



The mandatory user classes



The geometry

- User class which describes the geometry must inherit from **G4VUserDetectorConstruction** and registered in the Run Manager
- Virtual base class: the purely virtual method must be implemented
 - **G4VPhysicalVolume* Construct () = 0;**
 - Must return the pointer to the world volume: all other volumes are contained in it
- Optionally, implement the virtual method
 - **void ConstructSDandField ();**
 - Defines sensitive volumes and EM fields



Select physics processes

- Geant4 **doesn't have** any **default** particles or processes
- Derive your own **concrete class** from the **G4VUserPhysicsList** **abstract base class**
 - define all necessary **particles**
 - define all necessary **processes** and assign them to proper particles
 - define γ/δ production **thresholds** (in terms of range)
- **Pure virtual methods** of **G4VUserPhysicsList**

ConstructParticles()
ConstructProcesses()
SetCuts()



must be **implemented** by the user
in his/her concrete derived class



Action Initialization

- User class must **inherit** from **G4VUserActionInitialization** and registered in the Run Manager
- Implement the **purely virtual** method
 - `void Build() = 0;`
 - Invoked in **sequential mode** and in MT mode by **all workers**
 - Must instantiate **at least** the primary generator
- **Optional** virtual method
 - `void BuildForMaster() ;`
 - Invoked by the **master** in MT mode. Applies **only** to Run Action (all other user actions are thread-local)



Primary generator

- User class must **inherit** from **G4VUserPrimaryGeneratorAction**
 - **Registered** to the **Run Manager** via the **ActionInizialitation** (MT mode)
 - Register directly to the RunManager in seq-mode
- Implement the **purely virtual** method
 - **void GeneratePrimaries (G4Event*)=0 ;**
 - Called by the RunManager during the **event loop**, to generate the primary vertices/particles
- Uses internally a concrete instance of **G4VPrimaryGenerator** (e.g. **G4ParticleGun**) to do the job



The main() program



The main() program - 1

- Geant4 does **not** provide the `main()`
 - Geant4 is a toolkit!
 - The `main()` is part of the **user application**
- In his/her `main()`, the user **must**
 - construct `G4RunManager` (or his/her own derived class)
 - **notify** the `G4RunManager` **mandatory user classes** derived from
 - `G4VUserDetectorConstruction`
 - `G4VUserPhysicsList`
 - `G4VUserActionInitialization` (takes care of Primary)
 - The `G4RunManagerFactory` will pick the Sequential/MT version of the `G4RunManager`



The main() program - 2

- The user **may define** in his/her main()
 - optional user action classes
 - VisManager, (G)UI session
- The user also has to take care of **retrieving** and **saving** the relevant **information** from the simulation (Geant4 will not do that by default)
- Don't forget to **delete** the G4RunManager at the end



An example of main()

```
{  
  ...  
  // Create the run manager (let the RunManagerFactory decide if MT, sequential or other).  
  //The flags from G4RunManagerType are: Default (default), Serial, MT, Tasking, TBB  
  auto* runManager =  
    G4RunManagerFactory::CreateRunManager(G4RunManagerType::Default);  
  
  // Set mandatory user initialization classes  
  MyDetectorConstruction* detector = new MyDetectorConstruction;  
  runManager->SetUserInitialization(detector);  
  MyPhysicsList* physicsList = new MyPhysicsList;  
  runManager->SetUserInitialization(myPhysicsList);  
  
  // Set mandatory user action classes  
  runManager->SetUserAction(new MyActionInitialization);  
  
  // Set optional user action classes  
  runManager->SetUserAction(new MyEventAction());  
  runManager->SetUserAction(new MyRunAction());  
  ...  
}
```

Documentation

- A few **manuals** available in the Geant4 **webpage**
 - **Application developer** manual
 - **Physics** manual
- Other **tools** available
 - **LXR** code repository
 - **User forum**
 - **Bugzilla**
 - **GitHub** code repo

<https://github.com/Geant4>

User Support

Submitted by Anonymous (not verified) on Wed, 06/28/2017

1. [Getting started](#)
2. [Training courses and materials](#)
3. [Source code](#)
 - a. [Download page](#)
 - b. [LXR code browser](#)
 - c. [doxygen documentation](#)
 - d. [GitHub](#)
 - e. [GitLab @ CERN](#)
4. [Frequently Asked Questions \(FAQ\)](#)
5. [Bug reports and fixes](#)
6. [User requirements tracker](#)
7. [User Forum](#)
8. [Documentation](#)
 - a. [Introduction to Geant4 \[pdf \]](#)
 - b. [Installation Guide: \[pdf \]](#)
 - c. [Application Developers \[pdf \]](#)



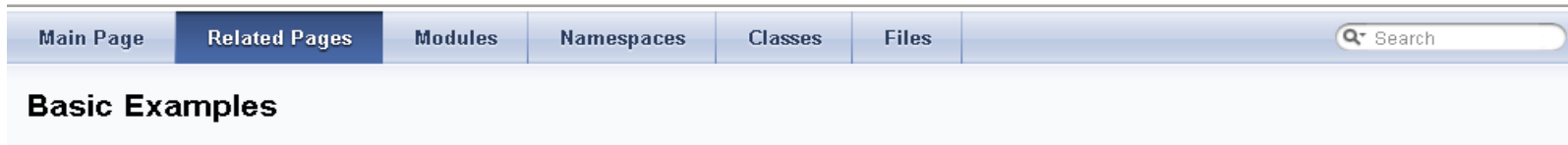
Examples

- Ready-for-the-use Geant4 applications (examples) are distributed with Geant4
 - Very good starting point for new users
- Three suites of examples:
 - **"basic"**: oriented to novice users and covering the most typical use-cases of a Geant4 application with keeping simplicity and ease of use.
 - **"extended"**: covers many specific use cases for actual detector simulation.
 - **"advanced"**: where real-life complete applications for different simulation studies are provided



Examples

- A [webpage with doxygen documentation](#) is available for the basic/extended examples



The screenshot shows a navigation menu with the following items: Main Page, Related Pages (highlighted), Modules, Namespaces, Classes, Files, and a search box labeled "Search". Below the menu, the heading "Basic Examples" is displayed.

The set of basic examples is oriented to "novice" users and covering many basic general use-cases typical of an "application"-oriented kind of development.

- **Example B1**
 - Simple geometry with a few solids
 - Geometry with simple placements ([G4PVPlacement](#))
 - Scoring total dose in a selected volume user action classes
 - Geant4 physics list ([QBBC](#))
- **Example B2**
 - Simplified tracker geometry with global constant magnetic field
 - Geometry with simple placements ([G4PVPlacement](#)) and parameterisation ([G4PVParameterisation](#))
 - Scoring within tracker via G4 sensitive detector and hits
 - Geant4 physics list ([FTFP_BERT](#)) with step limiter
 - Started from novice/[N02](#) example
- **Example B3**
 - Schematic Positron Emitted Tomography system

http://cern.ch/geant4/UserDocumentation/Doxygen/examples_doc/html



Who/why is using Geant4?



Experiments and MC

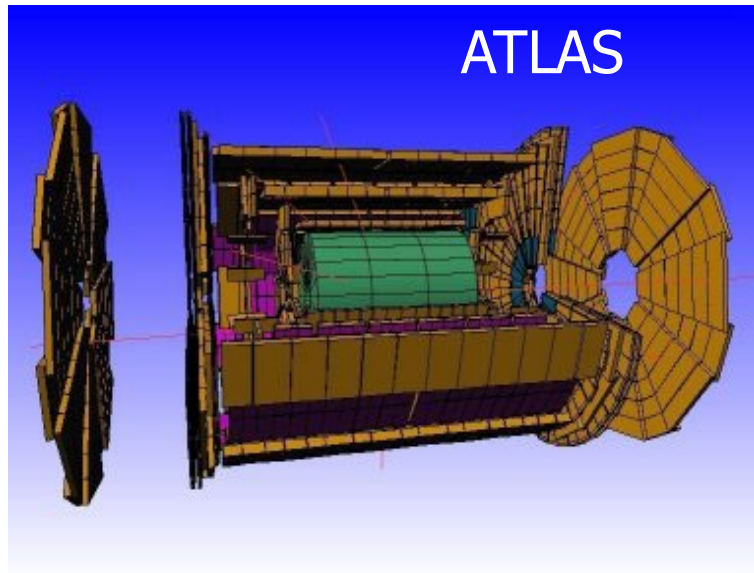
- In my knowledge, **all experiments** have a (more or less detailed) full-scale **Monte Carlo simulation**
- Design phase
 - Evaluation of **background**
 - **Optimization** of setup to maximize **scientific yield**
 - Minimize background, maximize signal efficiency
- Running/analysis phase
 - **Support** of **data analysis** (e.g. provide efficiency for signal, background, coincidences, tagging, ...).
 - Often, Monte Carlo is the only way to convert *relative rates* (events/day) in *absolute yields*



Why Geant4 is a common choice in the market

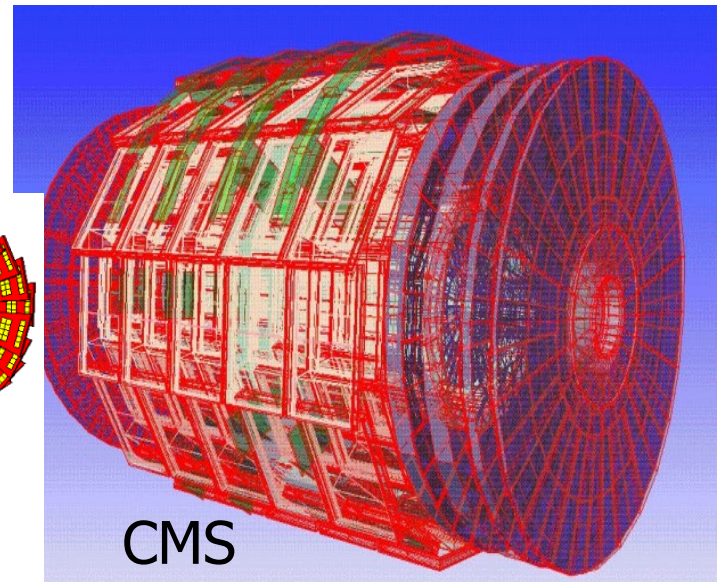
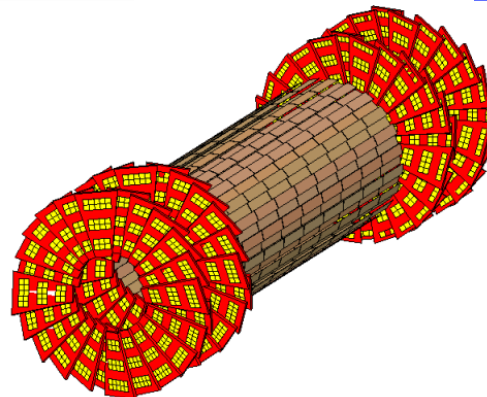
- Open source and object oriented/C++
 - No black box
 - Freely available on all platforms
 - Can be easily extended and customized by using the existing interfaces
 - New processes, new primary generators, interface to ROOT analysis, ...
- Can handle complex geometries
- Regular development, updates, bug fixes and validation
- Good physics, customizable per use-cases
- End-to-end simulation (all particles, including optical photons)

LHC @ CERN



- Benchmark with test-beam data
- Key role for the Higgs searches

- All four big **LHC experiments** have a Geant4 simulation
 - M of volumes
 - Physics at the TeV scale

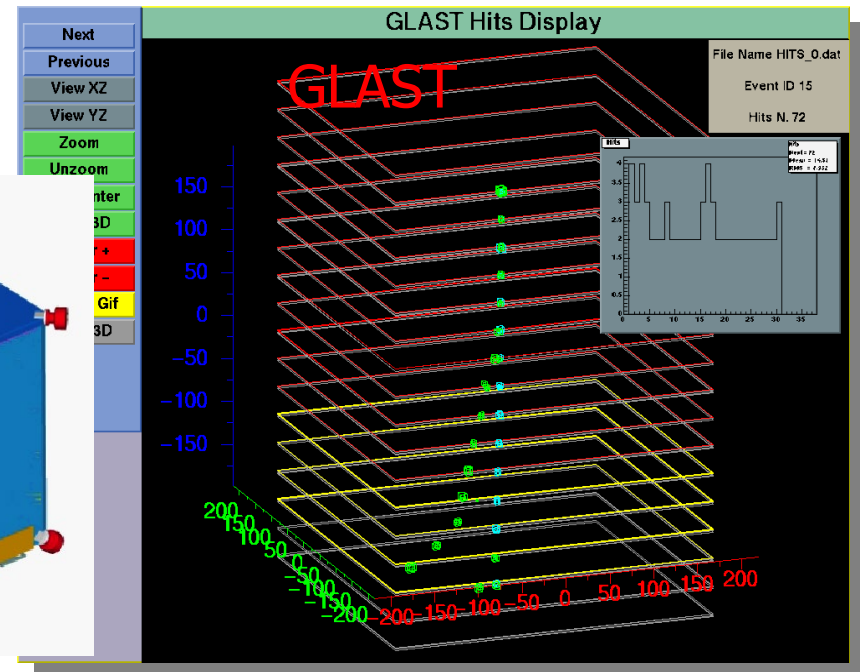
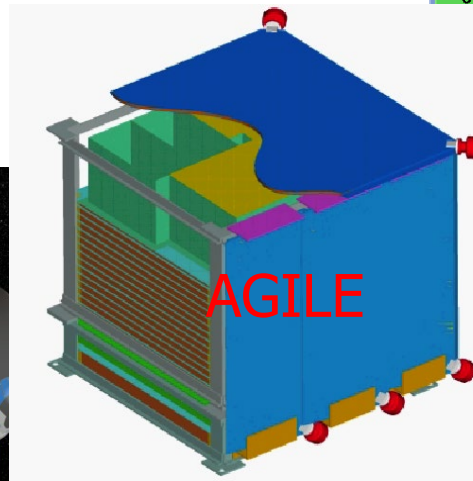
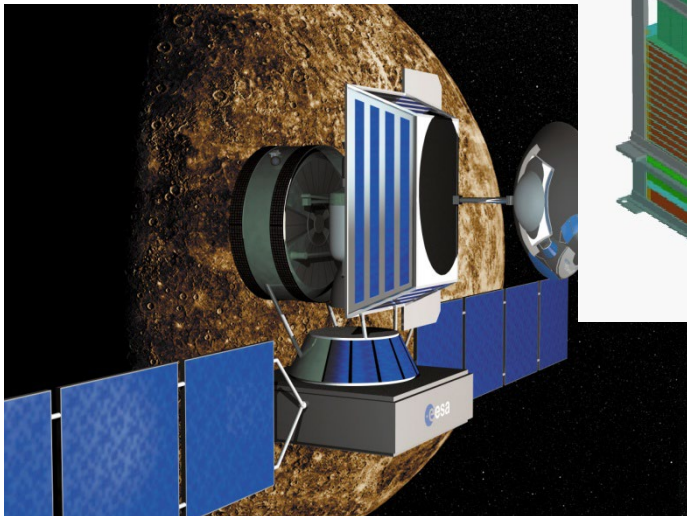


Space applications

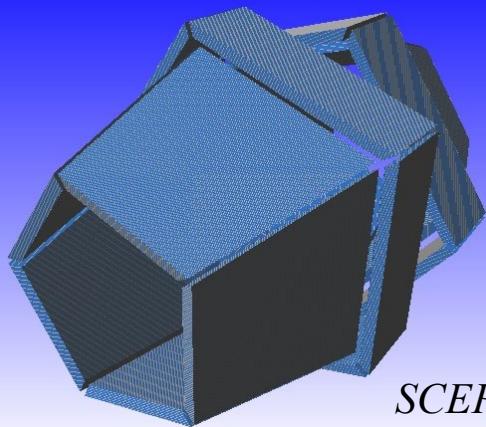
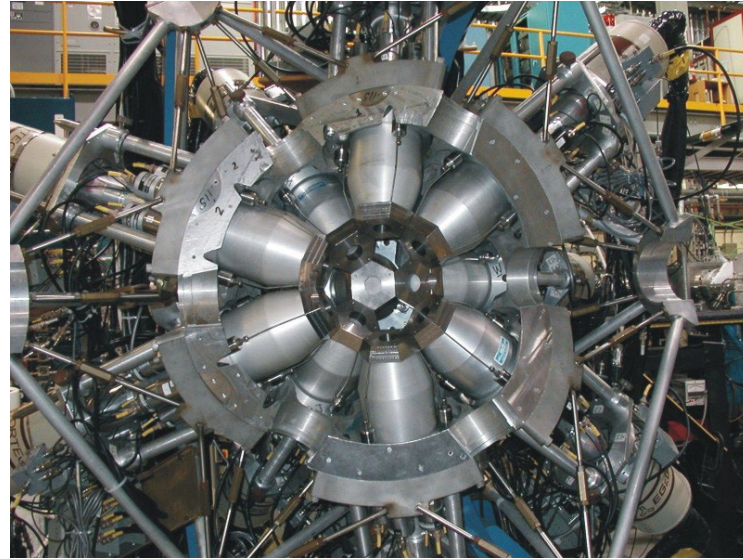
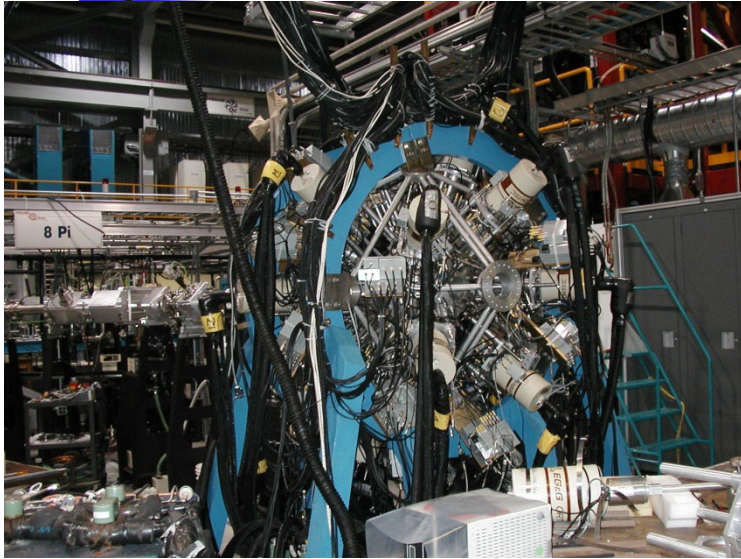
- **Satellites** (γ astrophysics, planetary sciences)
- **Funding from ESA**

Typical telescope:

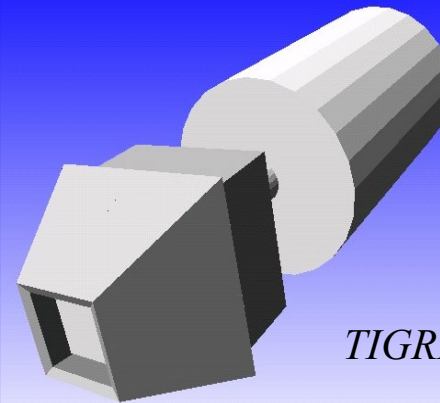
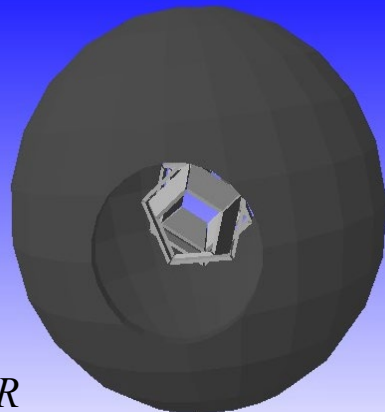
Tracker
Calorimeter
Anticoincidence



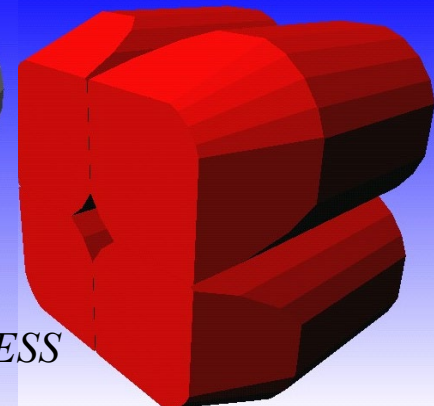
Nuclear spectroscopy



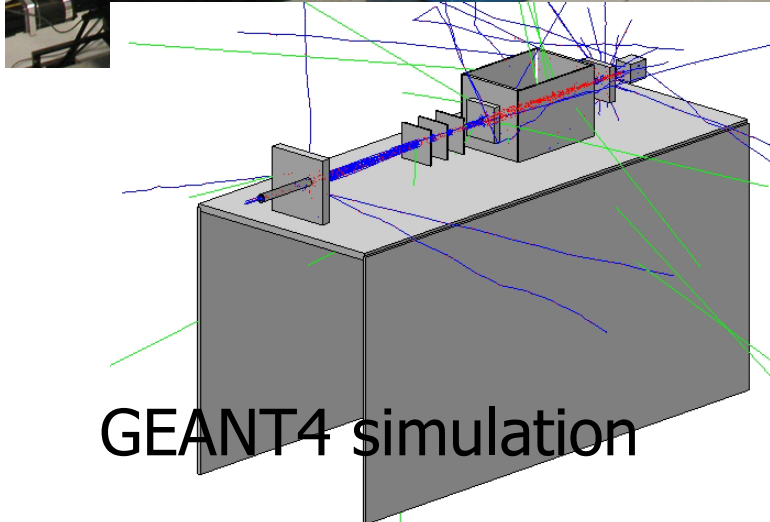
SCEPTAR



TIGRESS



Medical applications



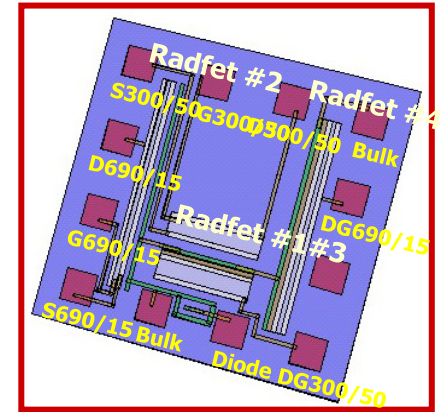
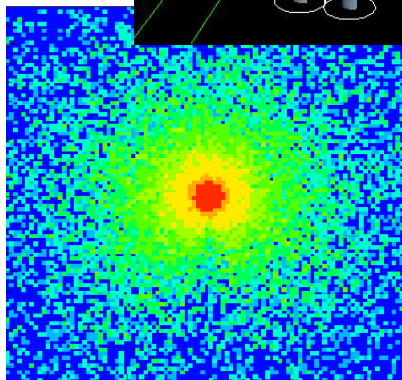
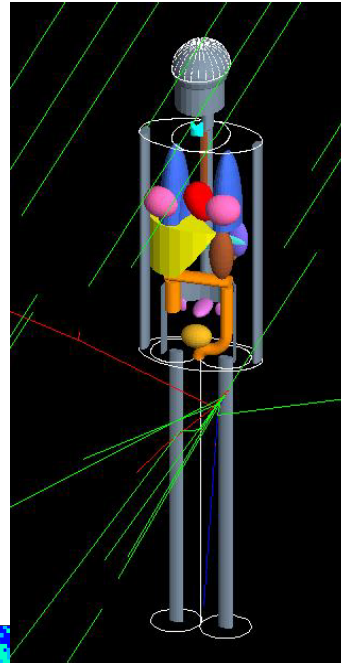
- **Treatment planning** for **hadrontherapy** and proton-therapy systems
 - Goal: deliver dose to the tumor while **sparing** the **healthy tissues**
 - Alternative to **less-precise** (and commercial) TP software
- **Medical imaging**
- **Radiation fields** from medical accelerators and **devices**
 - **medical_linac**
 - **gamma-knife**
 - **brachytherapy**

Dosimetry with Geant4

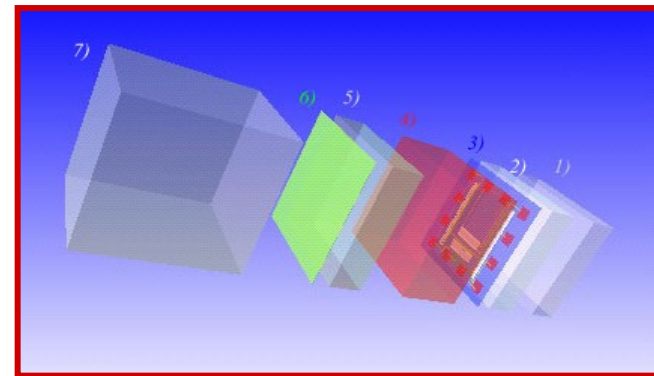
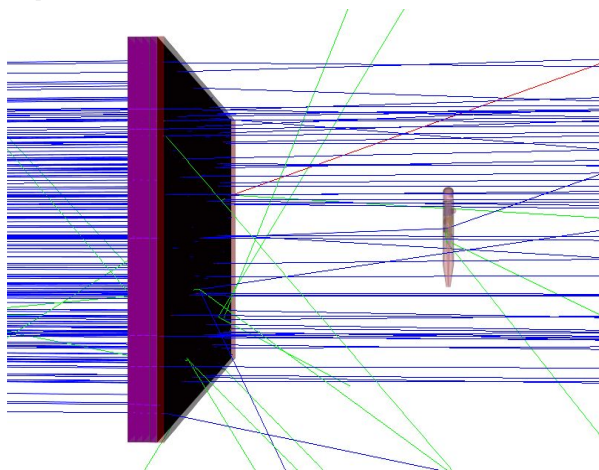


Space science

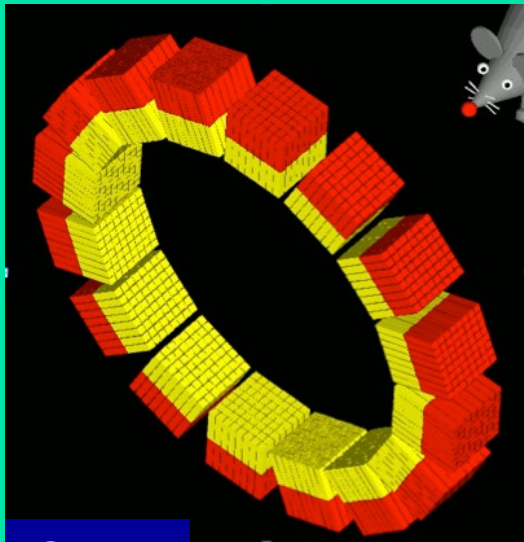
Radiotherapy



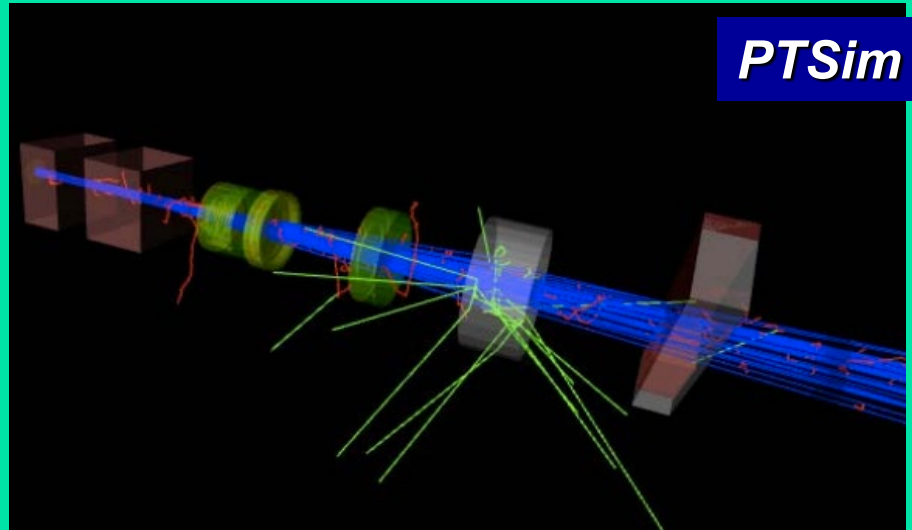
Effects on electronics components



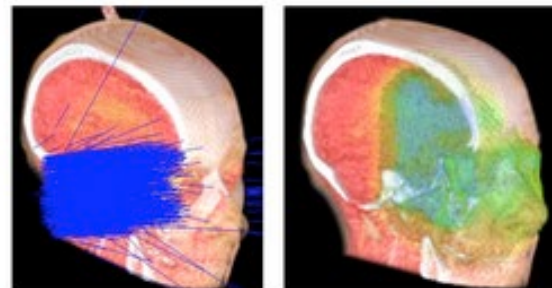
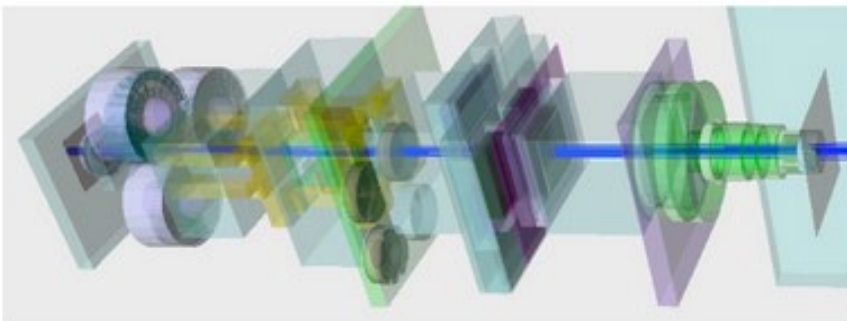
Geant4-based frameworks in the medical physics



GATE



PTSim



TOPAS



Backup



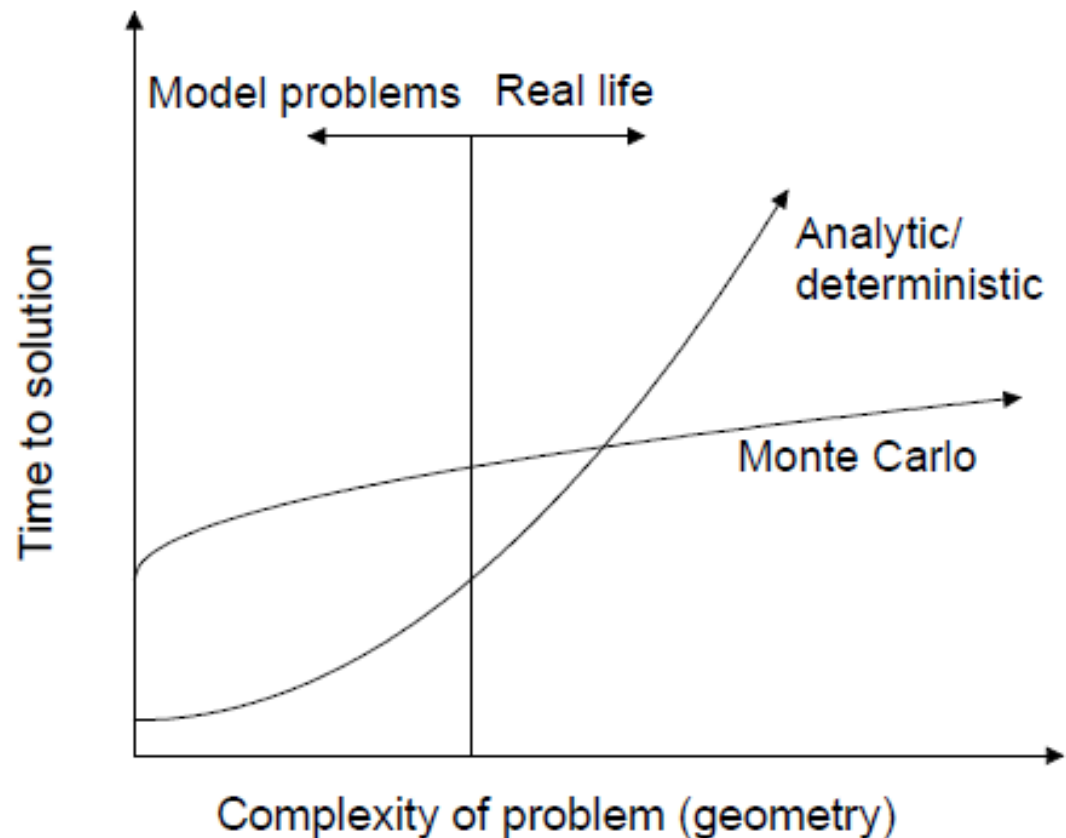
Sequential vs. MT

- Since Geant4 10.7, the **G4RunManagerFactory** makes the choice of the **appropriate run manager** in the main():
 - **G4RunManager** for sequential
 - **G4MTRunManager** for multi-thread
 - User code is exactly the same
- User can **set the choice**, via **G4RunManagerType**

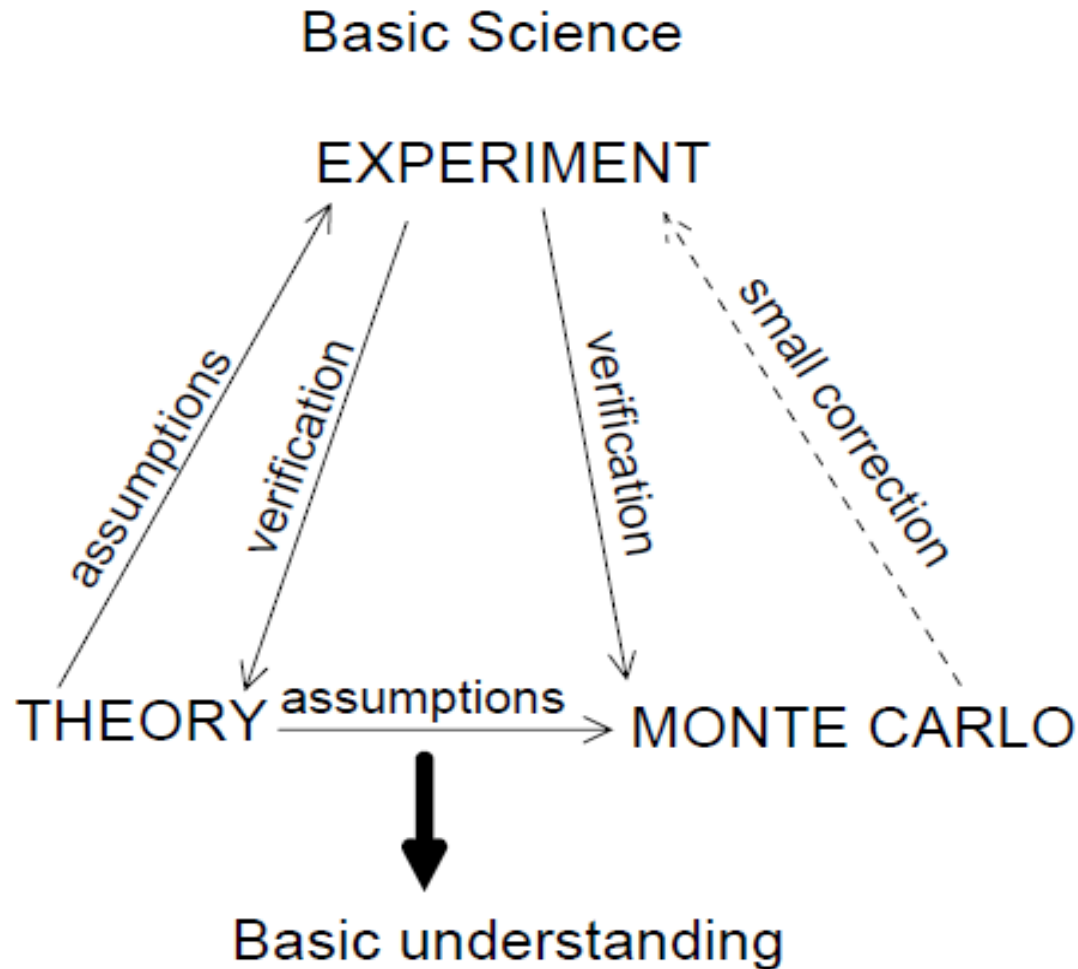
```
// Create the run manager (let the RunManagerFactory
// decide if MT, sequential or other).
// The flags from G4RunManagerType are: Default (default),
// Serial, MT, Tasking, TBB
auto* runManager =
    G4RunManagerFactory::CreateRunManager (
        G4RunManagerType::Default);
```

When are MC useful wrt to the math exact solution?


- Usually the **Monte Carlo** wins over the exact (mathematical) solution for **complex** problems



Interplay between theory, simulation and experiments



How to cook up the laws of physics into a tracking algorithm



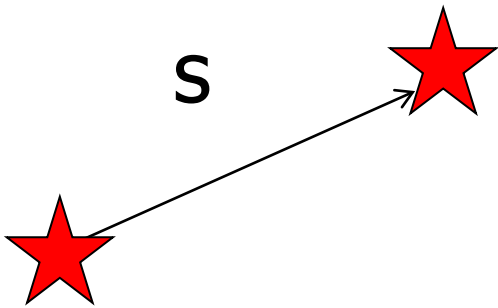
(Bird's eye view)



Particle tracking

- Distance s between two subsequent interactions distributed as

$$p(s) = \mu e^{-\mu s}$$



- μ is a property of the medium (homogeneous) and of the physics
- μ is proportional to the **total cross section** and depends on the **density** of the material

$$\mu = N\sigma = N \sum_i \sigma_i = \sum_i \mu_i$$

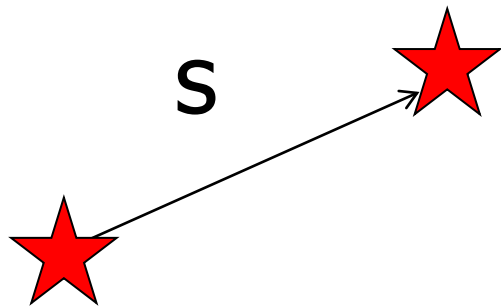
- All competing processes contribute with their own μ_i
- Each process takes place with probability $\mu_i/\mu \rightarrow$ i.e. proportionally to the **partial cross sections**



Particle tracking

- μ is proportional to the **total cross section** and depends on the **density** of the material

$$\mu = N\sigma = N \sum_i \sigma_i = \sum_i \mu_i$$



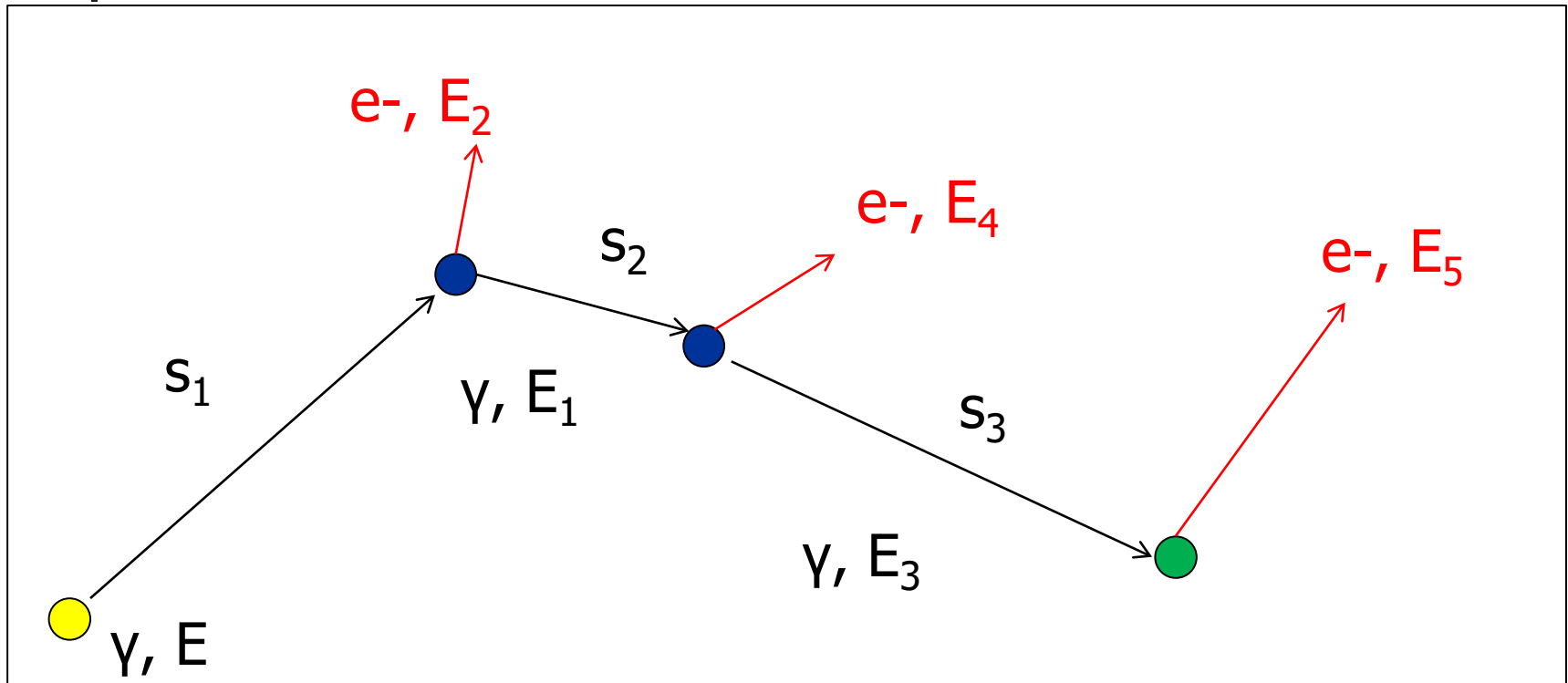
- All competing processes contribute with their own μ_i
- Each process takes place with probability $\mu_i/\mu \rightarrow$ i.e. proportionally to the **partial cross sections**



Particle tracking: basic recipe

- Divide the trajectory of the particle in "**steps**"
 - **Straight free-flight tracks** between consecutive physics interactions
 - Steps can also be **limited** by **geometry boundaries**
- Decide the **step length** s , by **sampling** according to $p(s) = \mu e^{-\mu s}$, with the **proper** μ (material+physics)
- Decide **which interaction** takes place at the end of the step, according to μ_i/μ
- Produce the **final state** according to the **physics** of the interaction ($d^2\sigma/d\Omega dE$)
 - Update **direction** of the primary particle
 - **Store** somewhere the possible **secondary particles**, to be tracked later on

Particle tracking: basic recipe



- Follow **all secondaries**, until **absorbed** or **leave** volume
- Notice: μ depends **on energy** (cross sections do!)



Well, not so easy

- This basic recipe works fine for γ -rays and other **neutral** particles (e.g. neutrons)
- Not so well for e^\pm : the **cross section** (ionization & bremsstrahlung) **is very high**, so the steps between two consecutive interactions are very small
 - **CPU intensive**: viable for **low energies** and **thin** material
- Even worse: in each interaction only **a small fraction of energy is lost**, and the **angular displacement** is small
 - A **lot of time** is spent to simulate interactions having **small effect**
 - The interactions of γ are "**catastrophics**": large change in energy/direction

Solution: the mixed Monte Carlo

- Simulate **explicitly** (i.e. force step) interactions **only if** energy loss (or change of direction) is **above threshold W_0**
 - **Detailed** simulation
 - "**hard**" interaction (like γ interactions)
- The effect of **all sub-threshold interactions** is described **statistically** (= cumulatively)
 - **Condensed** simulation
 - "**soft**" interactions
- **Hard interactions** occur **much less frequently** than soft interactions
 - Fully detailed simulation **restored** for **$W_0=0$**



The mixed Monte Carlo

- Has some **technical tricks**:
 - since **energy** is **lost** along the step due to **soft interactions**, the sampled step s cannot be too long ($s < s_{\max}$)

- Parameter μ_h between **hard collisions**

$$\mu_h = N \int_{W_0}^E \frac{d\sigma}{dW}(E) dW$$

- Has $\mu_h \ll \mu$ because the **differential cross section** is strongly peaked at low W (= soft secondaries)
- Much **longer step length**



The mixed Monte Carlo

- Stopping power due to **soft** collisions (dE/dx)

$$S_s = N \int_0^{W_0} W \frac{d\sigma}{dW}(E) dW$$

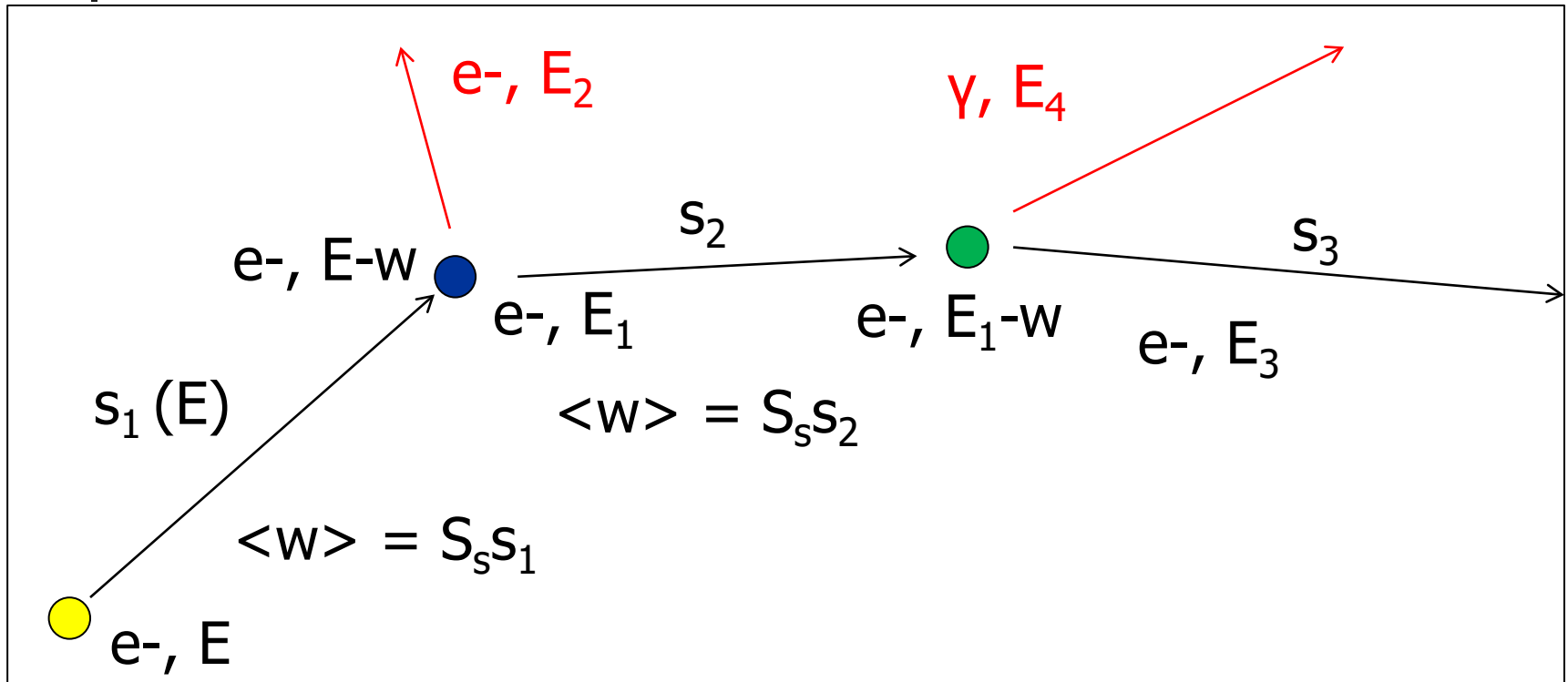
- Average energy lost along the step: $\langle w \rangle = sS_s$
 - Must be $\langle w \rangle \ll E$
- Fluctuations around the average value $\langle w \rangle$ have to be taken into account
 - Appropriate **random sampling** of w with mean value $\langle w \rangle$ and variance (**straggling**)



Extended recipe

1. Decide the **step length** s , by sampling according to $p(s) = \mu_h e^{-\mu_h s}$, with the **proper** μ_h
2. Calculate the **cumulative effect** of the **soft** interactions along the step: sample the **energy loss** w , with $\langle w \rangle = sS_s$, and the **displacement**
3. Update **energy and direction** of the primary particle at the end of the step $E \rightarrow E-w$
4. Decide which **interaction takes place** at the end of the step, according to $\mu_{i,h}/\mu_h$
5. Produce the **final state** according to the **physics** of the interaction ($d^2\sigma/d\Omega dE$)

Particle tracking: mixed recipe

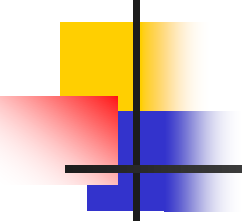


- Follow **all secondaries**, until absorbed or leave volume



Geometry

- **Geometry** also **enters** into the tracking
 - A **step** can **never cross** a geometry **boundary**
 - **Always stop** the step when there is a **boundary**, then **re-start** in the new medium
- **Navigation** in the geometry can be **CPU-intensive**
 - One must know to **which volume** each **point (x,y,z)** **belongs to**, and **how far** (and in which direction) is the **closest boundary**
- **Trajectories** can be affected also by **EM fields**, for charged particles



...luckily enough, somebody else already implemented the tracking algorithms for us (and much more)

General recipe for novice users

Experienced users may do **much more**, but the conceptual process is still the same...

- **Design** your application... requires some preliminar **thinking** (what is it supposed to do?)
- Create your **derived mandatory** user classes
 - My**DetectorConstruction**
 - My**PhysicsList**
 - My**ActionInitialization** (must register My**PrimaryGenerator**)
- Create **optionally** your **derived user action classes**
 - My**UserRunAction**, My**UserEventAction**, ...
- Create your **main()**
 - Instantiate **G4RunManager** or your own derived MyRunManager
 - **Notify** the RunManager of your mandatory and optional user classes
 - Optionally initialize your favourite User Interface and Visualization
- That's all!