



This course



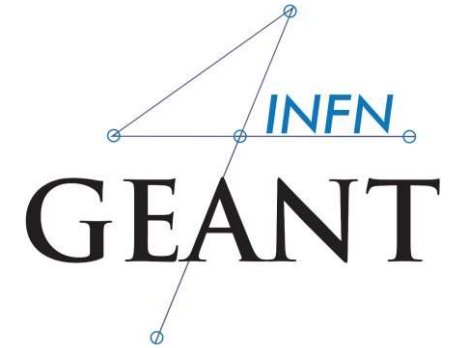
Structure and logistics - 1

- 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/vienna2024/introduction`
- A **pre-installed virtual machine** and **docker container** are provided for the hands-on sessions
 - Includes **Geant4 11.2.p01** on a Linux environment
 - You should already have it downloaded and tested
 - Please **let us know** ASAP if you have problems



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://indico.cern.ch/event/1275551/>
 - 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

at the VIENNA Workshop on Simulations (VIEWS24)

Vienna, April 22nd- 25th, 2024



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

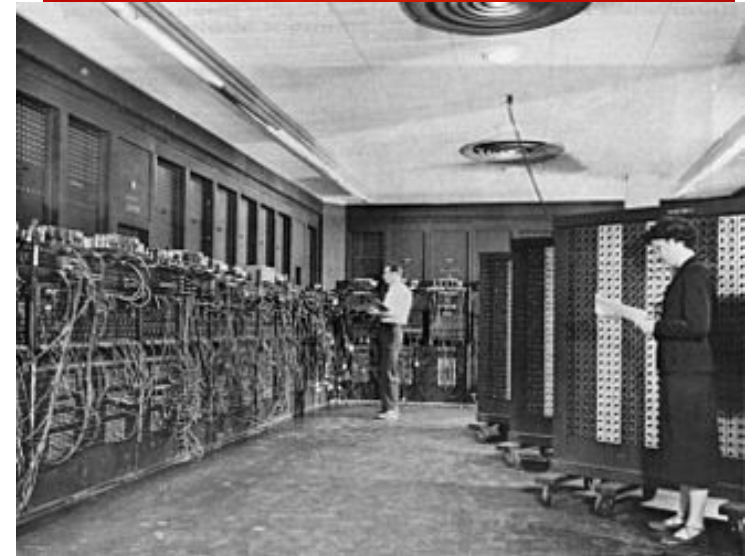
- Very **concept** of Monte Carlo comes in the **XVIII century** (Buffon, 1777, and then Laplace, 1786)
 - Monte Carlo **estimate of π**
- Concept of MC **is much older** than real **computers**
 - one can also implement the **algorithms manually**, with dice (= Random Number Generator)



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 46

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 AUGUSTEA 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)

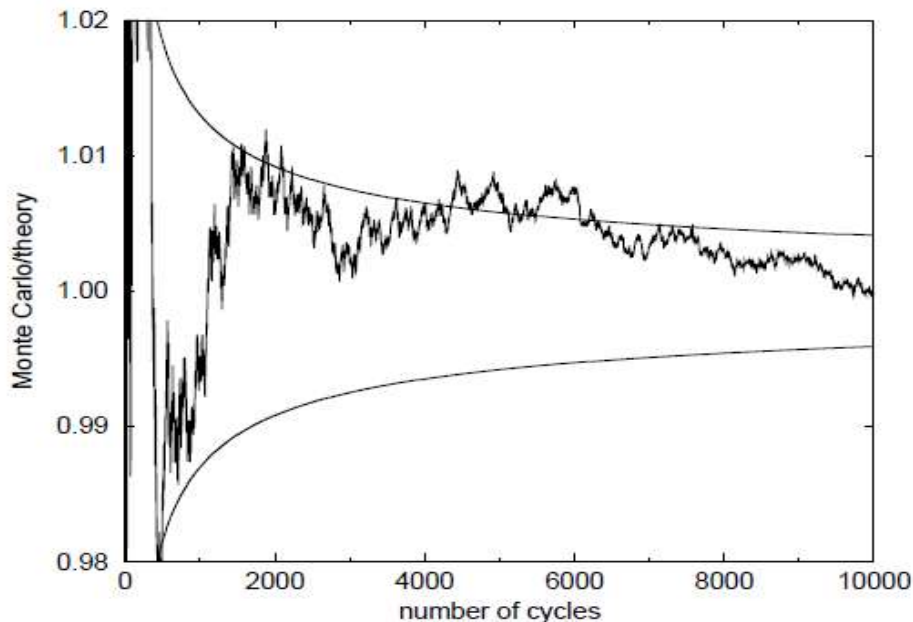
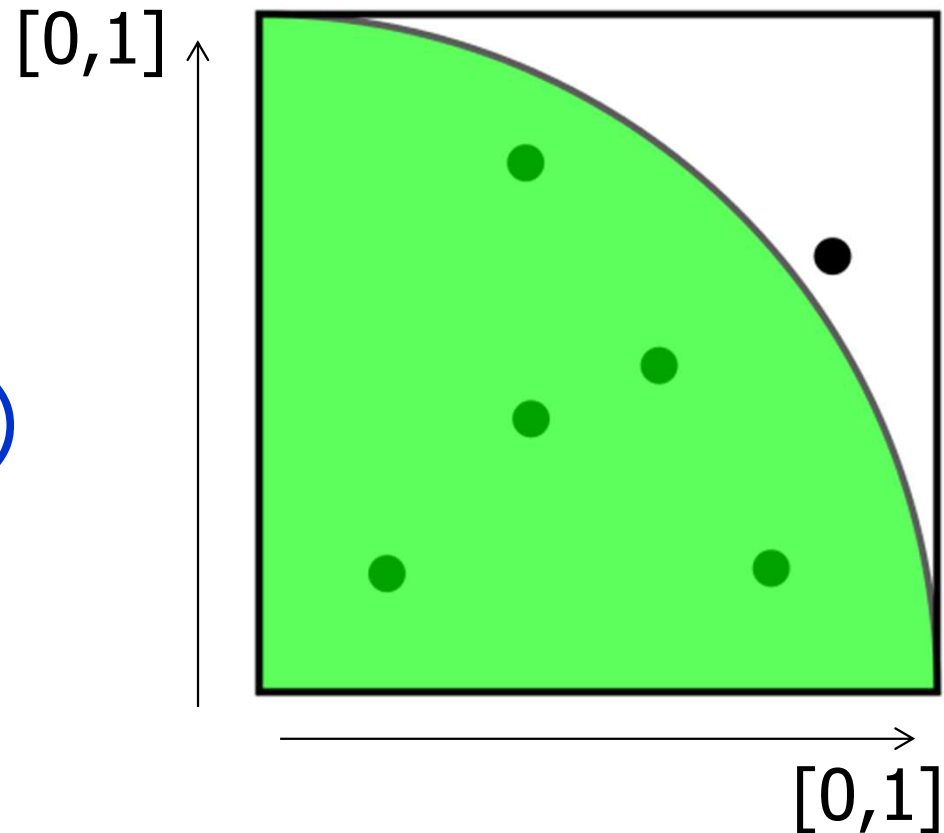


Nick Metropolis enjoying a break in the quantum Monte Carlo conference, September 1985.

With MANIAC: the first
electronic digital
computer

The simplest MC application: numerical estimate of π

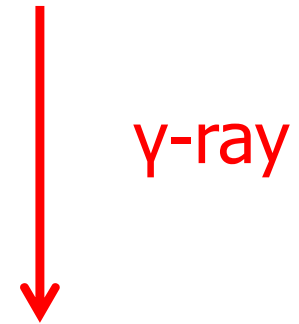
- Shoot N couples (x,y) randomly in $[0,1]$
- Count n : how many couples satisfy $(x^2+y^2 \leq 1)$



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

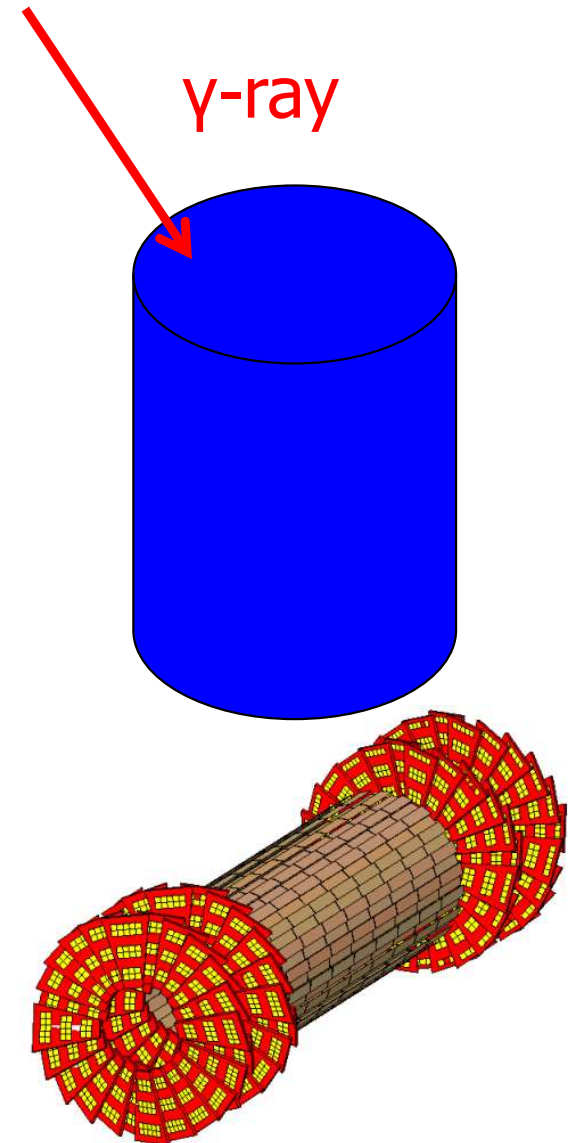
Most common application in particle physics: particle tracking

- 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 complicate**, as in modern physics





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>

Geant4

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.

[Getting started](#)

Get started

Everything you need to get started with Geant4.

[I'm ready to start!](#)

Download

Geant4 source code and installers are available for download, with source code under an [open source license](#).

Latest: [11.2.1](#)

Docs

Documentation for Geant4, along with tutorials and guides, are available online.

[Read documentation](#)

News

[» More](#)

- 16 Feb 2024
[Release 11.2.1](#)
- 08 Dec 2023
[Release 11.2](#)
- 10 Nov 2023
[Release 11.1.3](#)
- 30 Jun 2023



```
template <typename T>
struct G4TaskSingletonEvaluator
{
    using key_type = typename G4Traits::TaskSingletonKeyT::type;
    using data_type = G4TaskSingletonDataT;
};
```

Events

[» More](#)

- 25/3/2024 - 29/3/2024 [Geant4 tutorial at JLAB](#), Jefferson Lab, Newport News (Virginia, USA)
- 25/3/2024 - 26/3/2024 [Geant4-DNA International tutorial](#), Osaka University, Osaka (Japan)
- 27/3/2024 - 29/3/2024 [The 5th Geant4 International User Conference at the Physics - Medicine - Biology frontier](#), Osaka University Nakanoshima Center, Osaka (Japan)
- 11/4/2024 - 11/4/2024 [61st Geant4 Technical Forum](#), CERN, Geneva (Switzerland)
- 15/4/2024 - 19/4/2024 [Geant4 Beginners Course @ CERN](#), CERN, Geneva (Switzerland)
- 22/4/2024 - 27/4/2024 [Geant4 Course at the VIenna Workshop on Simulations 2024 \(VIEWS24\)](#), Campus Akademie, Vienna (Austria)
- 9/6/2024 - 14/6/2024 [Geant4 Course at the XXI Seminar on Software for Nuclear, Sub-nuclear and Applied Physics](#), Porto Conte, Alghero (Italy)

<https://geant4.org>

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



GEANT4 versions and releases

- 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 11.2.p01**
 - Released February 16th, 2024
 - This is the version installed in the VM/docker used for this course
- **Requires C++11** (gcc > 4.8.x)
 - **Native** C+11 features in-place



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...

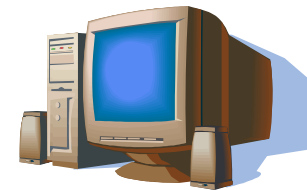
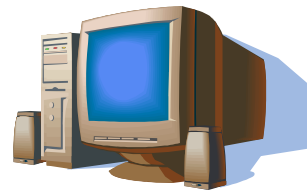
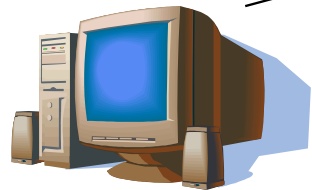
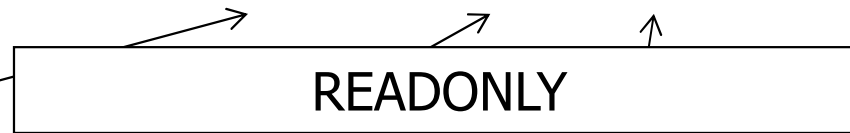
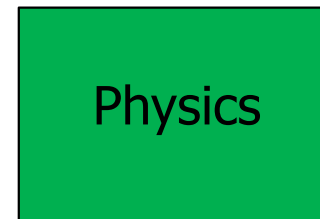
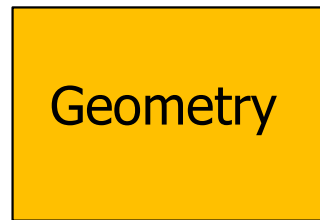
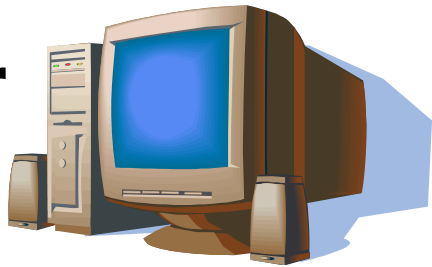


Multi-thread mode

- Geant4 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 ...

Master

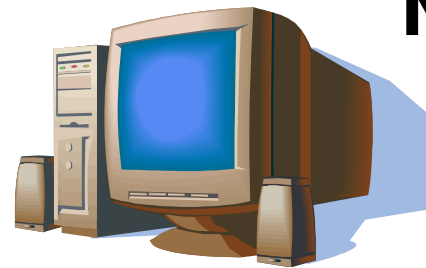
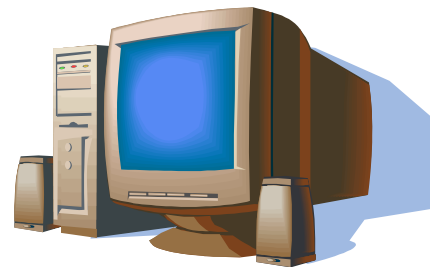
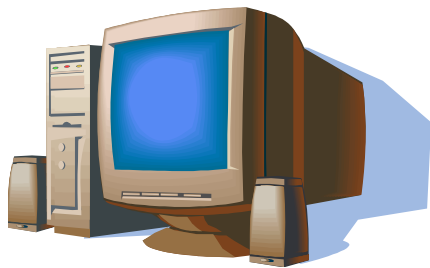


Workers



... vs. parallelisation

Nodes



Geometry

Physics

Primary

RunAction

EvtAction

Geometry

Physics

Primary

RunAction

EvtAction

Geometry

Physics

Primary

RunAction

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
(Mon-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



Physics Lists

- Geant4 **doesn't have** any **default** particles or processes
- *Partially true*: there is **no default**, but there are a set of **"ready-for-use" physics lists** released with Geant4, **tailored** to different **use cases**. Mix and match:
 - Different sets of **hadronic models** (depending on the energy scale and modeling of the interactions)
 - Different options for **neutron** tracking
 - Do we need (CPU-intensive) description of thermal neutrons, neutron capture, etc?
 - Different options for **EM physics**
 - Do you need (CPU-intensive) precise description at the low-energy scale (< 1 MeV)? E.g. fluorescence, Doppler effects in the Compton scattering, Auger emission, Rayleigh diffusion
 - Only a waste of CPU time for LHC, critical for many low-background experiments



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 **ActionInitialization** (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::Serial);  
  
    // 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>

On this page

[Introduction to Geant4](#)

[Installation Guide](#)

User guides

[For Application Developers](#)

[For Toolkit Developers](#)

[Physics Reference Manual](#)

[Physics List Guide](#)

Examples

[Frequently Asked Questions](#)

[Geant4 source code](#)



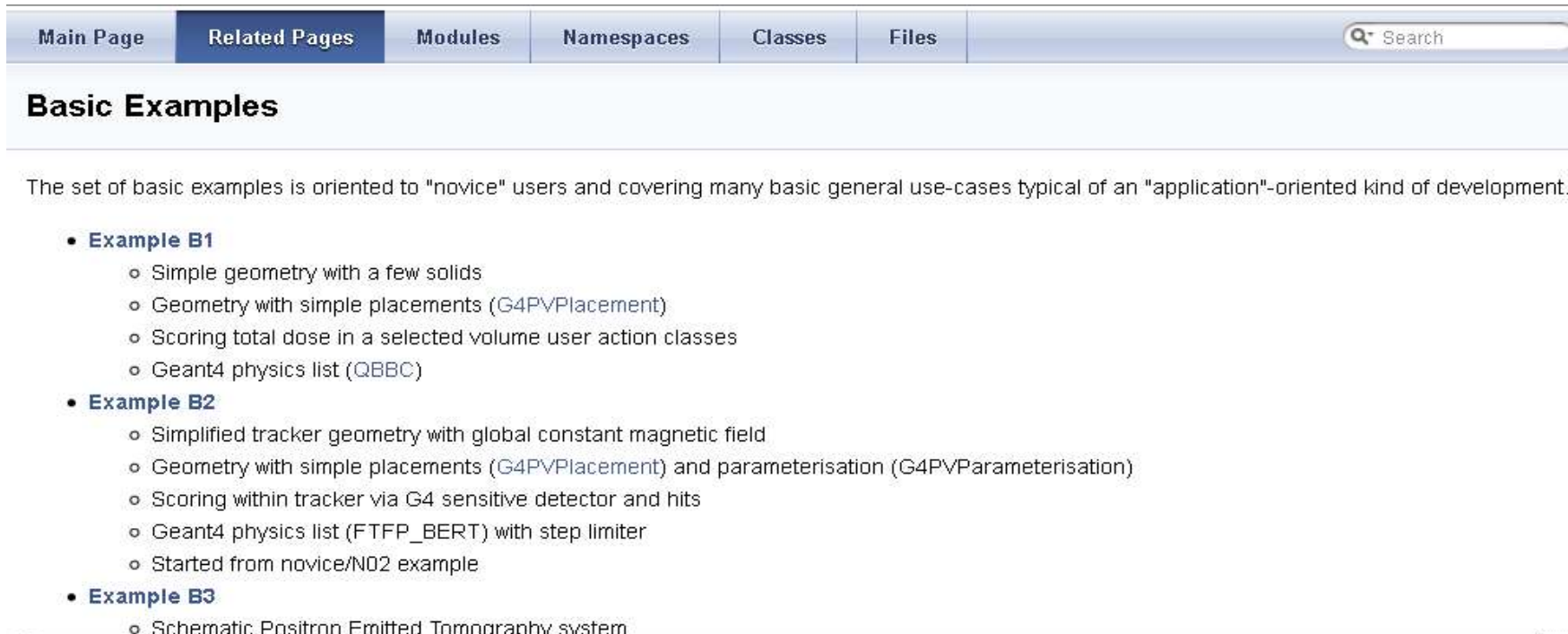
Examples

- 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 web browser interface with a navigation menu at the top. The menu includes 'Main Page', 'Related Pages', 'Modules', 'Namespaces', 'Classes', and 'Files'. A search bar is located on the right side of the menu. Below the menu, the page title is 'Basic Examples'. The main content area contains a paragraph and a list of examples:

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

https://geant4-userdoc.web.cern.ch/Doxygen/examples_doc/html/index.html



Who/why is using Geant4?



Experiments and MC

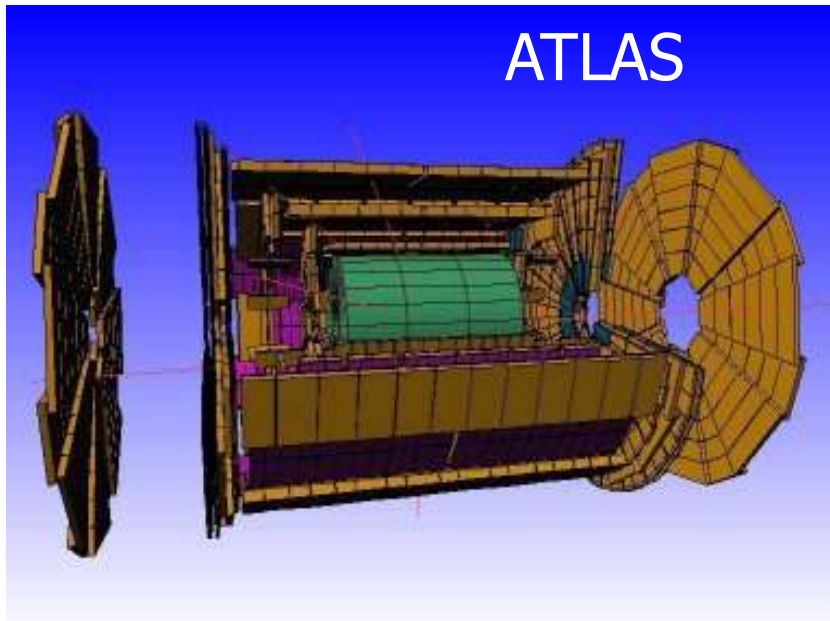
- In my knowledge, **all experiments** have a (more or less detailed) full-scale **Monte Carlo simulation**
- Design phase
 - Evaluation of **background**
 - **Optimization** of setup to maximize **scientific yield**
 - Minimize background, maximize signal efficiency
- Running/analysis phase
 - **Support** 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

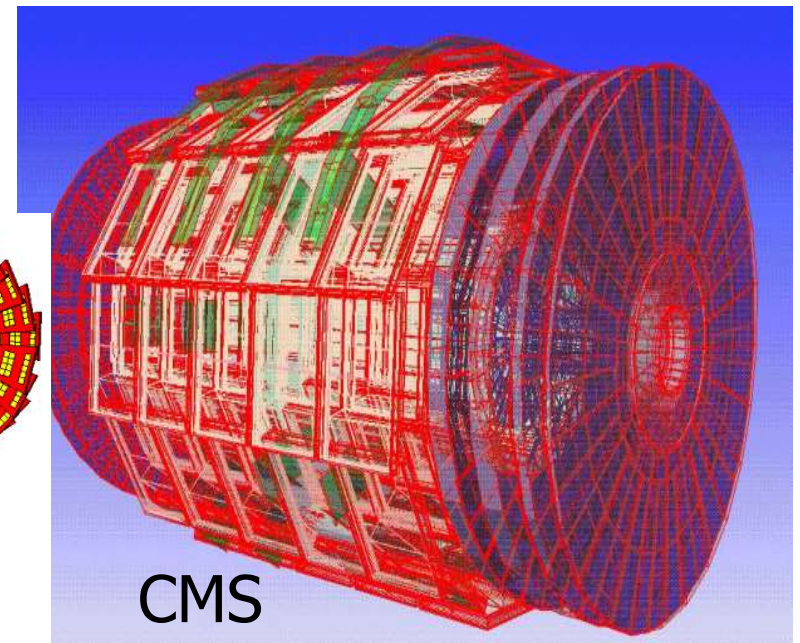
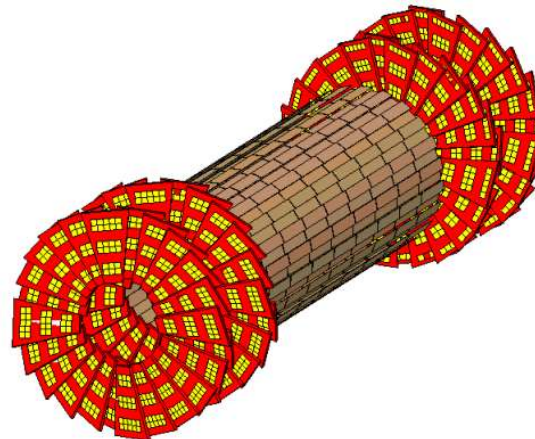
- Open source and object oriented/C++
 - No black box
 - Freely available on all platforms
 - Can be easily extended and customized by using the existing interfaces
 - New processes, new primary generators, interface to ROOT analysis, ...
- Can handle complex geometries
- Regular development, updates,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

LHC @ CERN



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

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

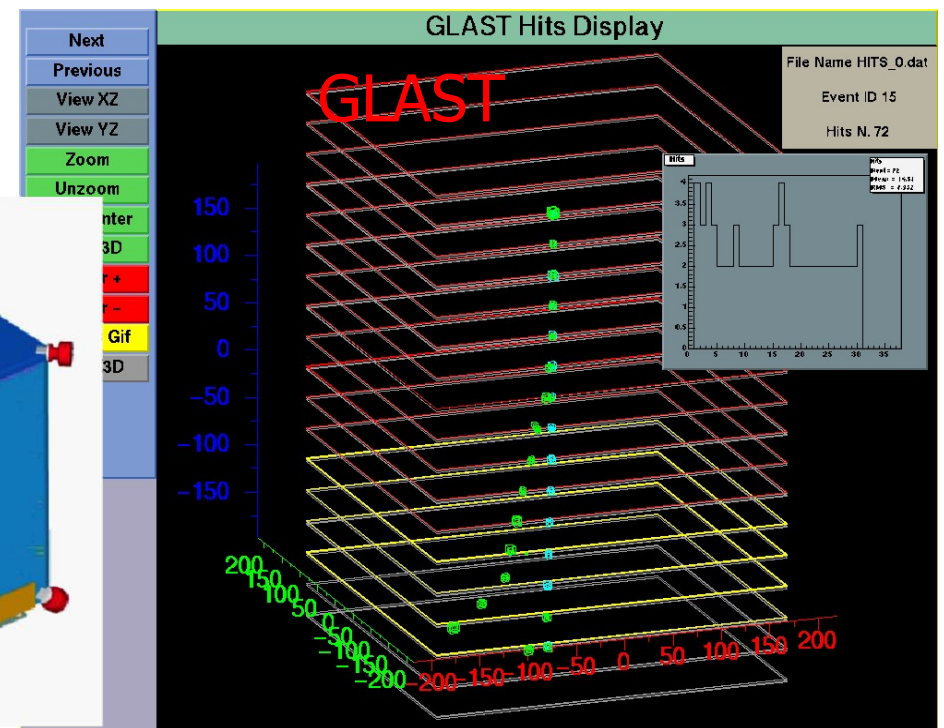
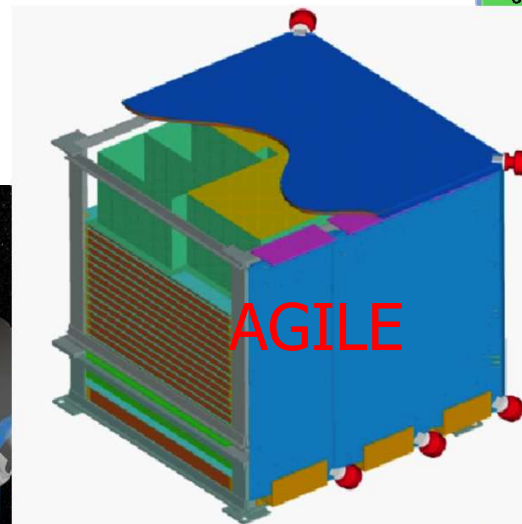
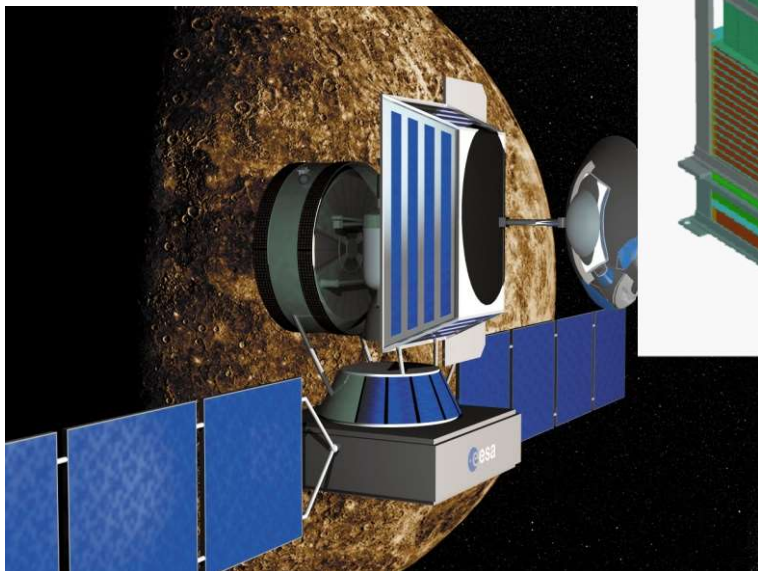


Space applications

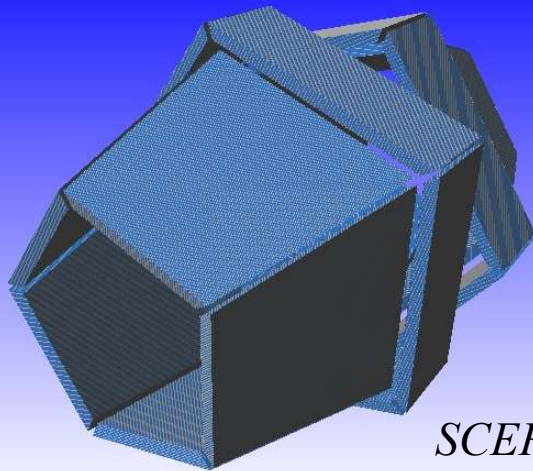
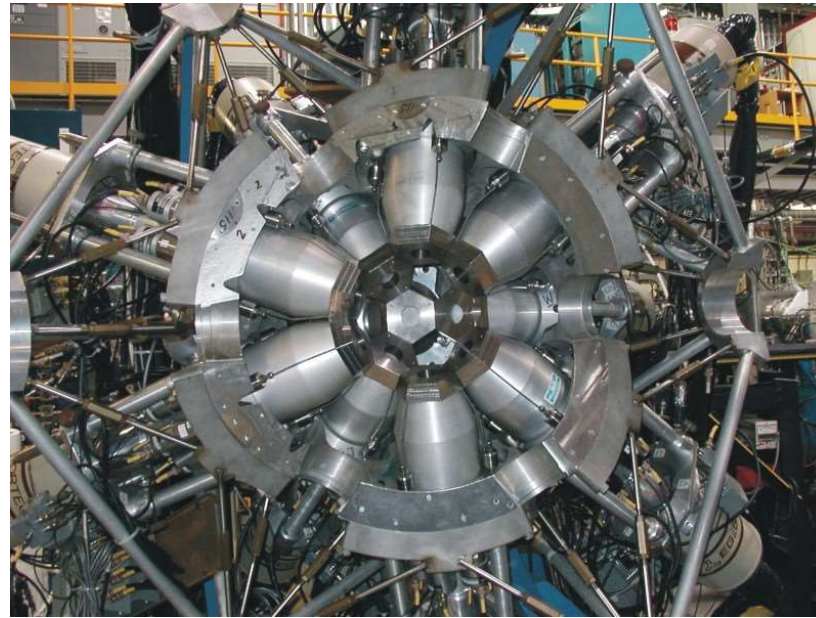
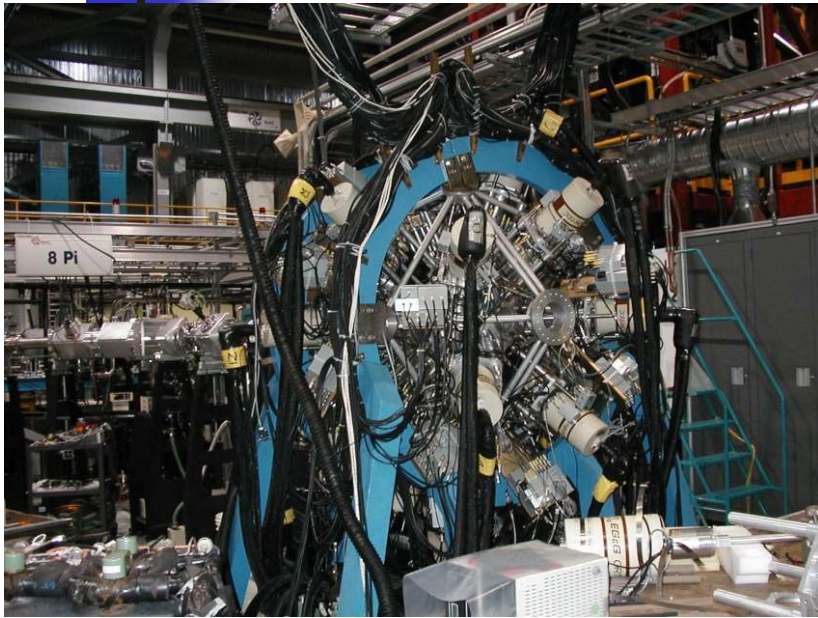
- Satellites (γ astrophysics, planetary sciences)
- Funding from ESA

Typical telescope:

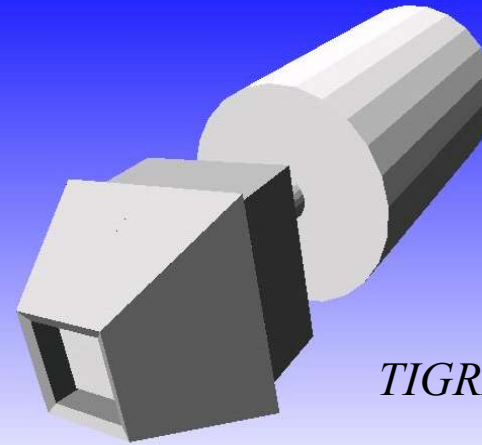
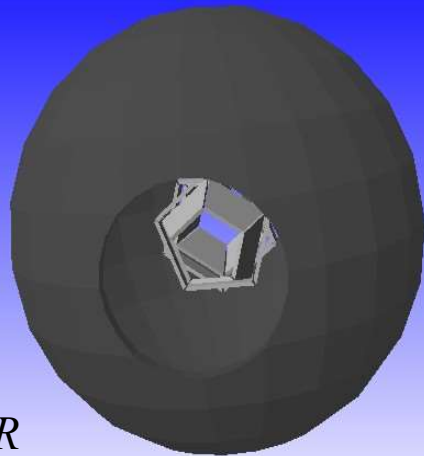
Tracker
Calorimeter
Anticoincidence



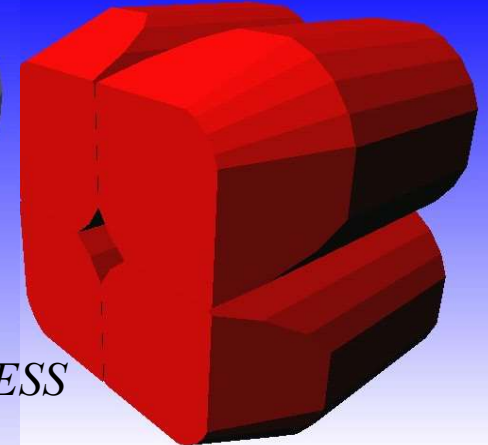
Nuclear spectroscopy



SCEPTAR



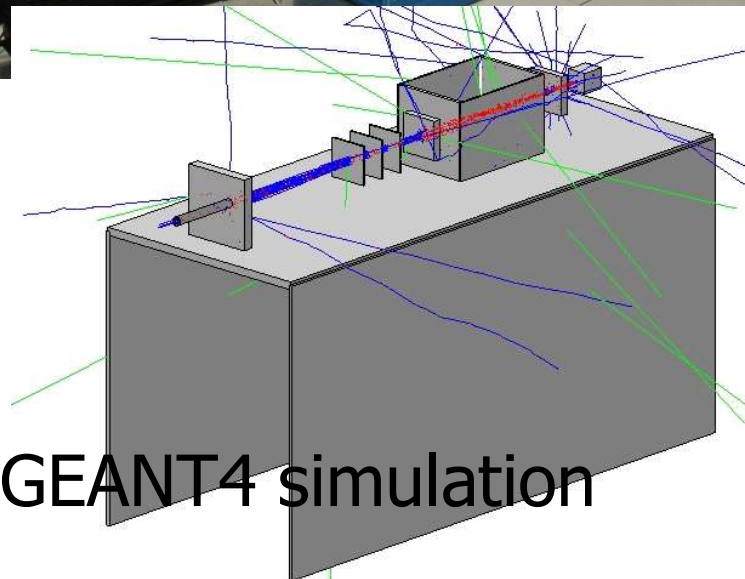
TIGRESS



Medical applications



Proton-therapy beam line



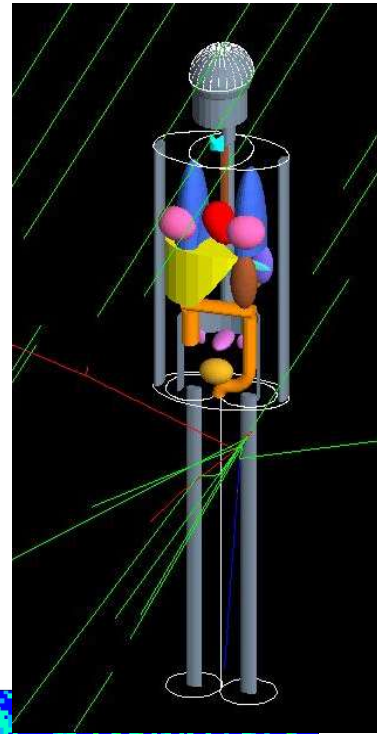
GEANT4 simulation

- **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**

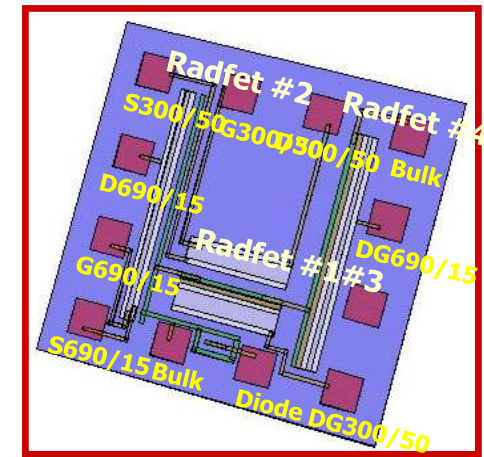
Dosimetry with Geant4



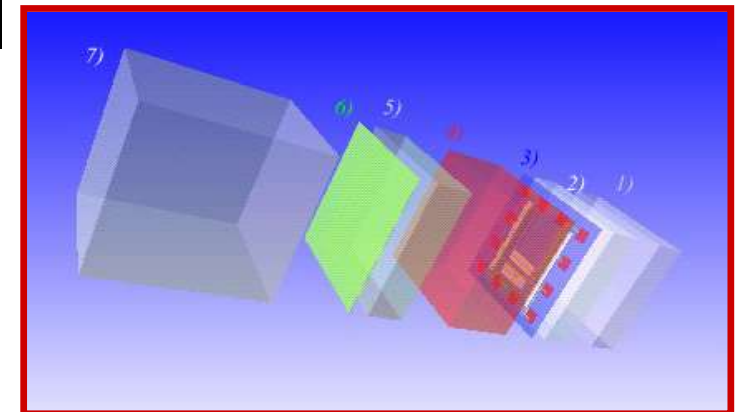
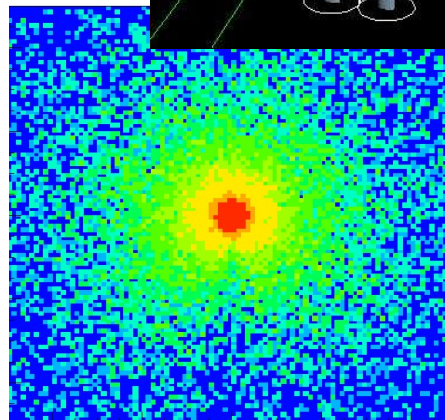
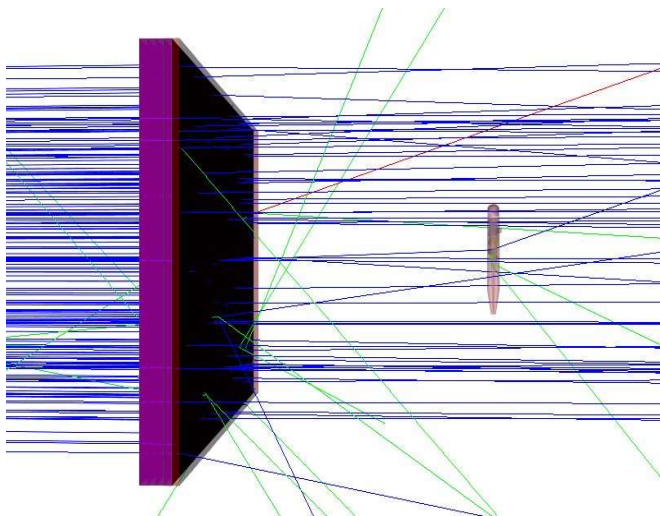
Space science



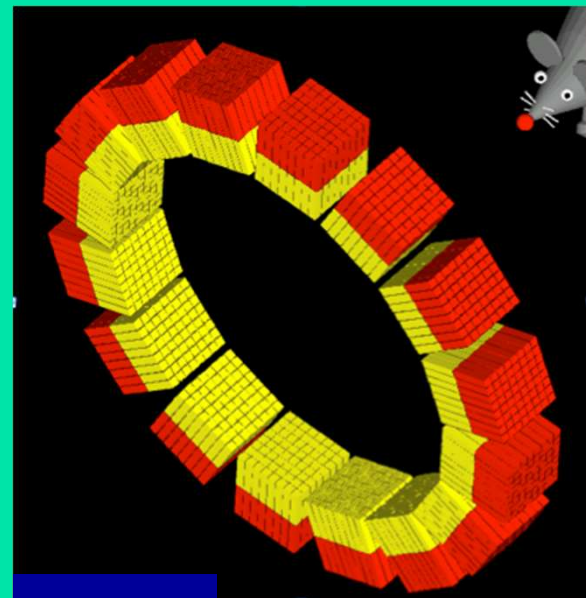
Radiotherapy



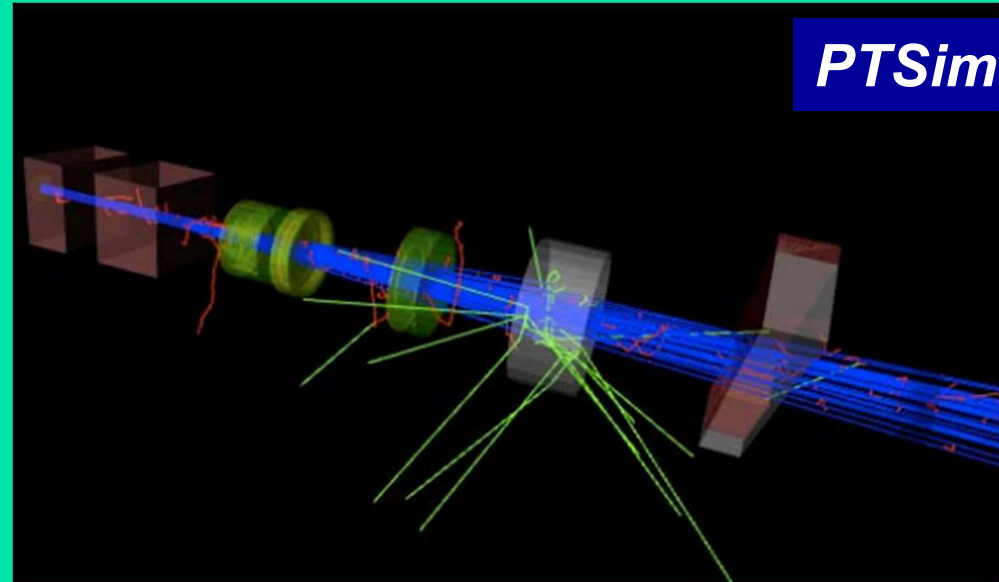
Effects on electronics components



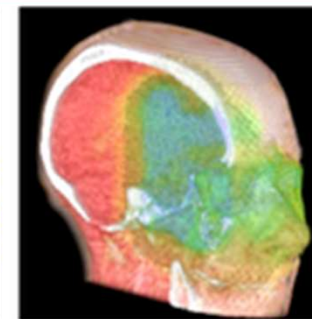
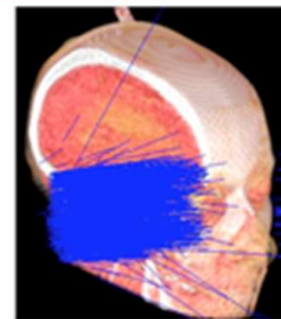
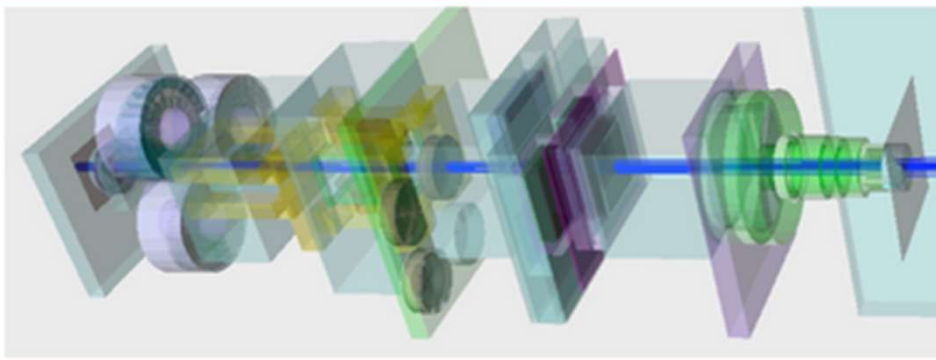
Geant4-based frameworks in the medical physics



GATE



PTSim



TOPAS