The Geant4 simulation toolkit: an introduction

Giada Petringa (LNS-INFN)

XVI Seminar on Software for Nuclear, subnuclear and Applied Physics

G4 School Outline

- Generals on Monte Carlo
- Basic capability of Geant4
- Basic structure of the Geant4 components

At the end of the school

Configuration
Generation of particles
Geometry and materials
Tracking
Physics
Scoring

Finding the material

- Pablo Cirrone, Giada Petringa, Davide Chiappara, Ruhani Khanna
  INFN-Laboratori Nazionali del Sud - Catania, (I)
- Official tutorial and school regularly offered: see the Geant4 web pages
- Official Geant4 web pages:
  www.geant4.org
- The Italian Geant4 group:
  https://web.infn.it/Geant4-INFN/
  https://www.facebook.com/SoftwareandGeant4School/

Structure and Logistics- I

- 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.lngs.infn.it/alghero2019/introduction/index.html
- A pre-installed virtual machine is provided for the hands-on sessions
  - Includes Geant4 10.05.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/event/17240/
- 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

Lesson Outline

- Introduction to the Monte Carlo method
- Introduction to Geant4
- Structure of the Geant4 Kernel
- Structure of a typical Geant4 application

Monte Carlo method

Monte Carlo techinque

- 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 intrisically probabilistic (e.g. numerical integration)

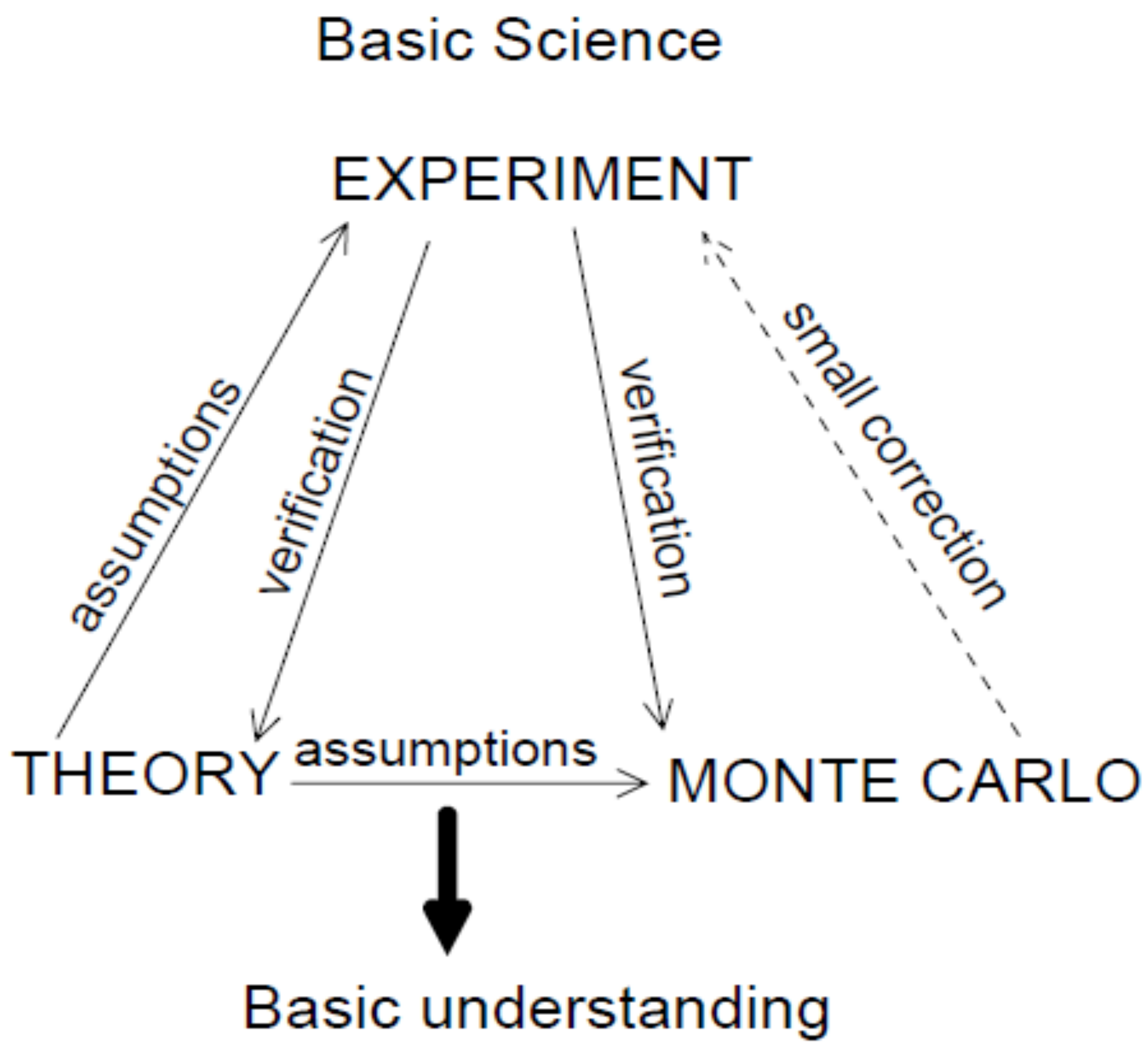
Monte Carlo techinque

- Produce a configuration (or a "final state"), according to some "laws", e.g.
  - People mostly arrive in pairs
  - Audience members prefer an un-obstructed view of the stage
  - Audience members prefer seats in the middle, and close to the front row
  - Only one person can occupy a seat
- Contrarily e.g. to physics, the laws are not known
  - Rather use "working assumptions"
- The math (exact) formulation can be impossible or unpratical => MC is more effective

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
- Can be used to validate/disproof a theory, and/or to provide small corrections to the theory
- In this course: Monte Carlo for particle tracking (interaction of radiation with matter)

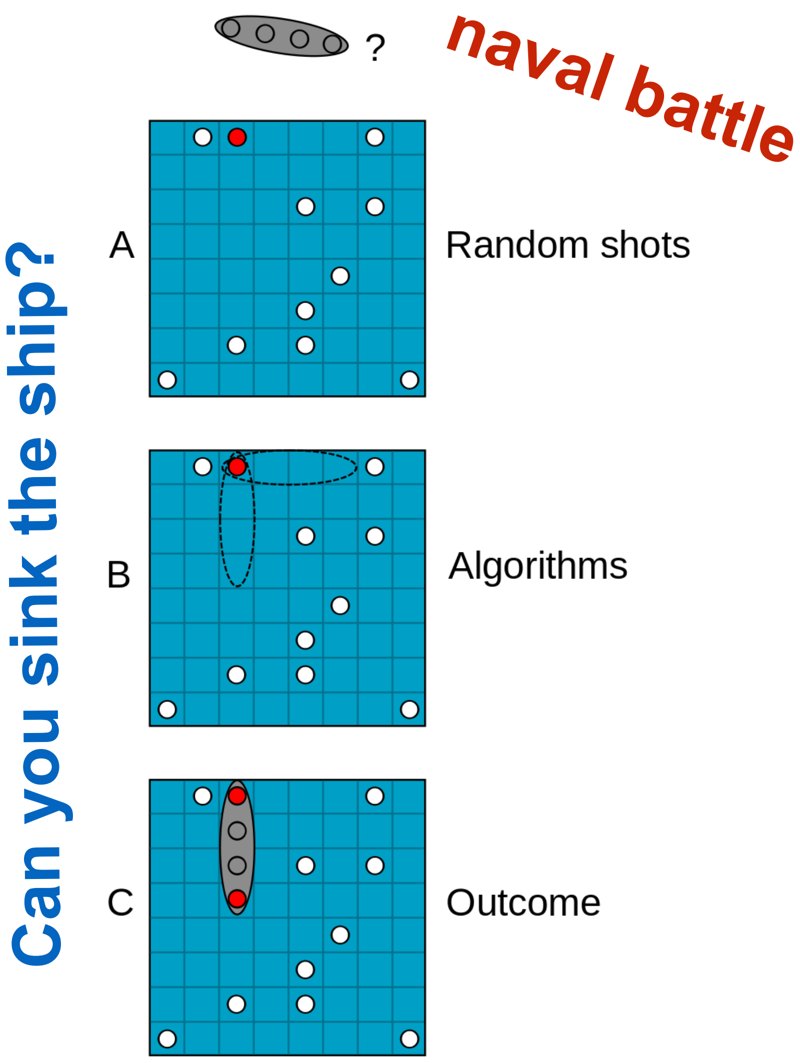
Interplay between theory, simulations and experiments



MC application

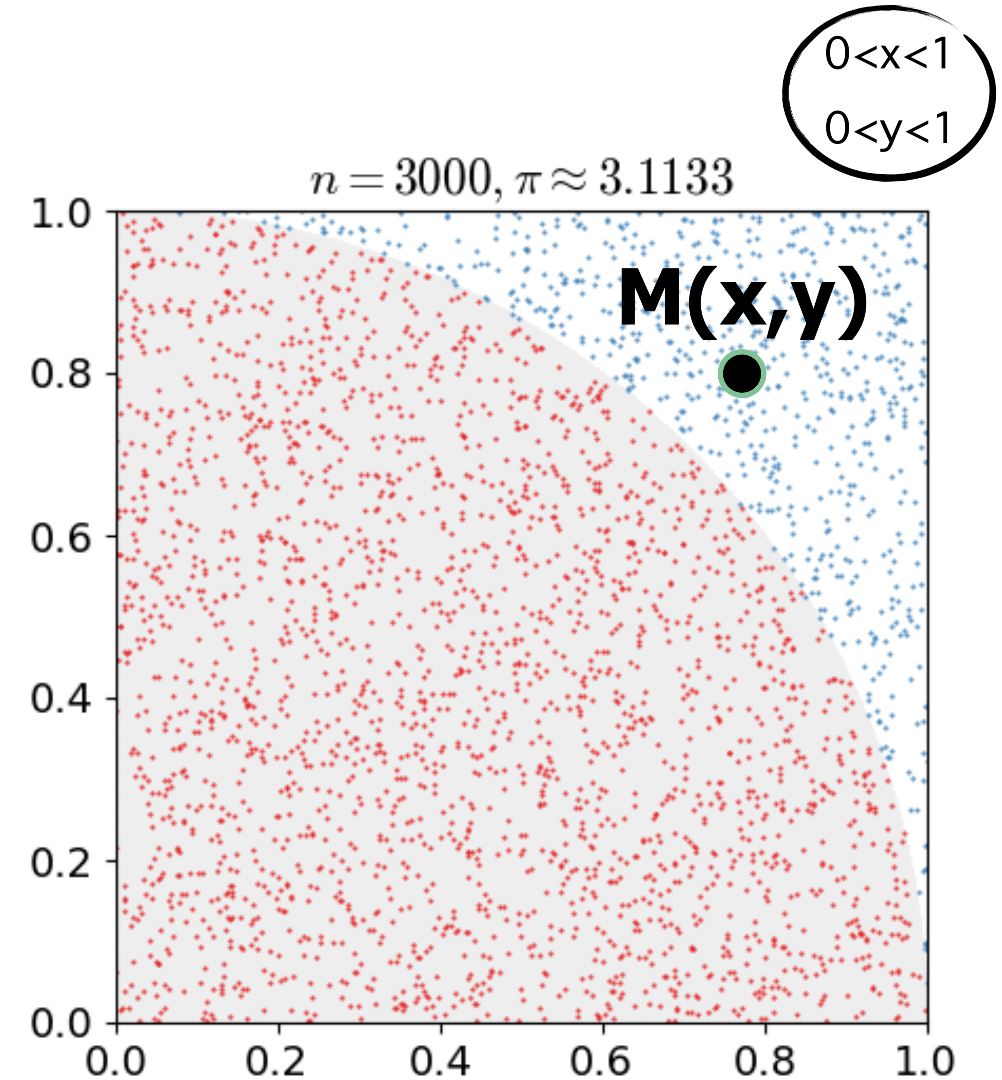
Monte Carlo methods vary, but tend to follow a particular pattern:

1) Define a domain of possible inputs
2) Generate inputs randomly from a probability distribution over the domain
3) Perform a deterministic computation on the inputs
4) Aggregate the results



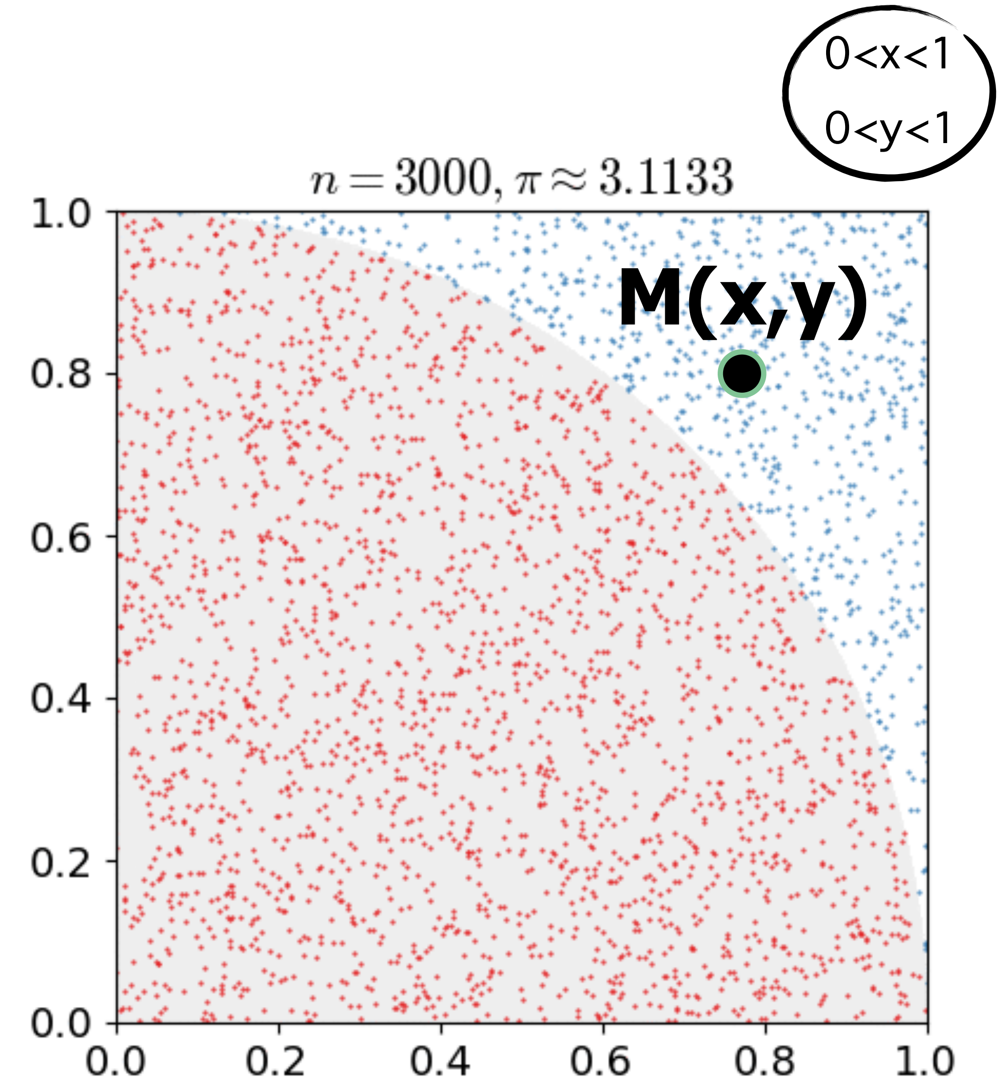
The simplest MC application

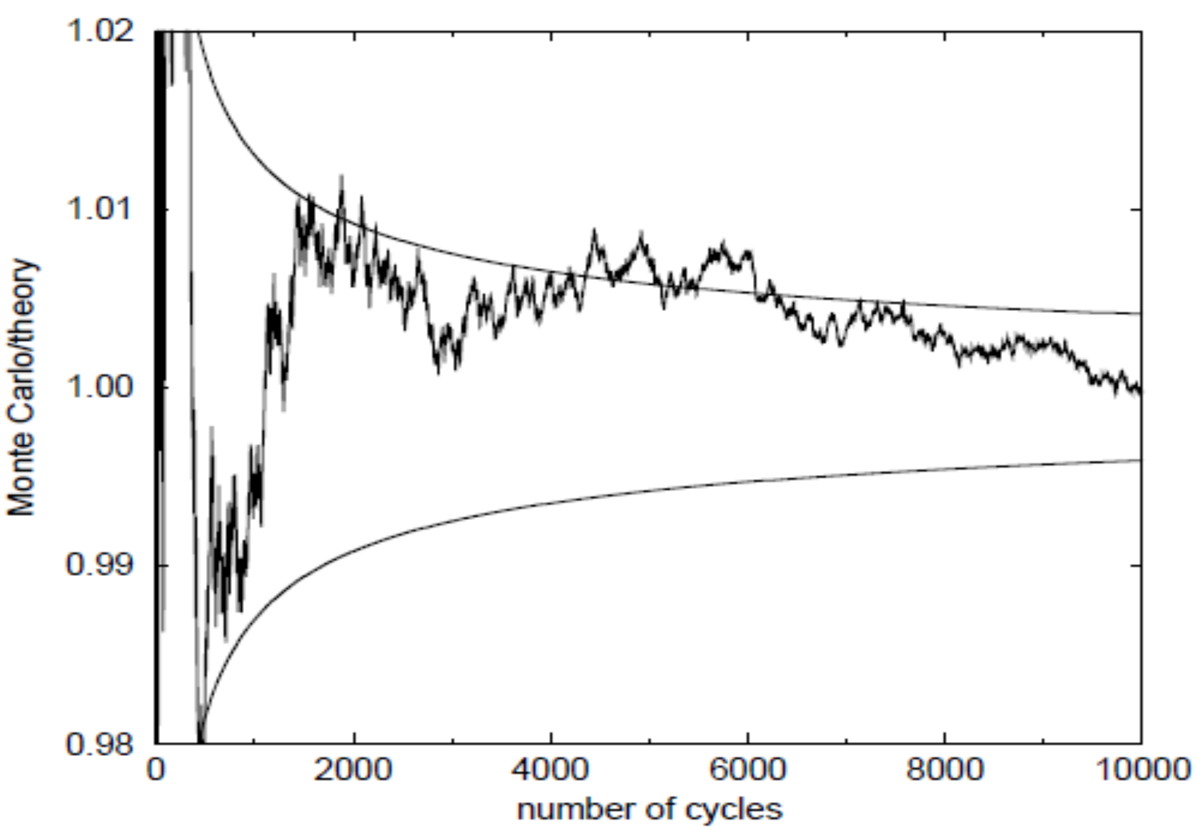
For example, consider a quadrant (circular sector) inscribed in a unit square. Given that the ratio of their areas is π/4, the value of π can be approximated using a Monte Carlo method



The simplest MC application

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

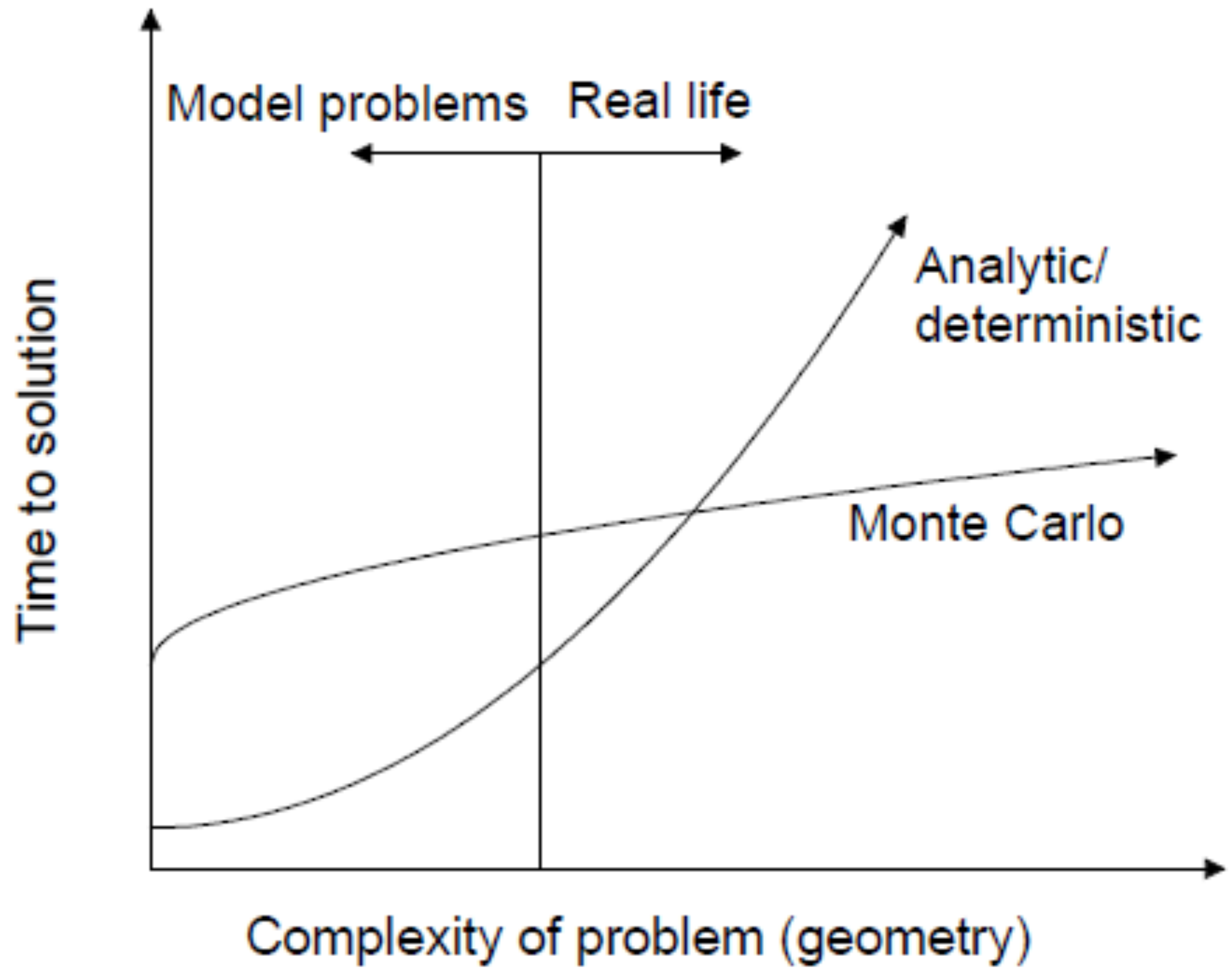




- n/N = r*π/4 (ratio of areas)
- Probability=Asector/Asquare
- Convergence as $1/\sqrt{N}$

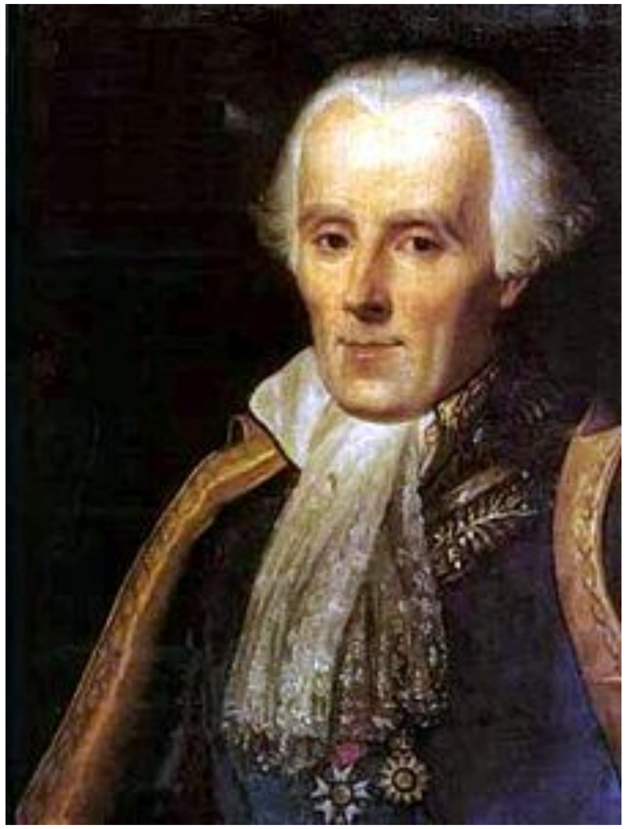
When are MC useful to the math solution?

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



A bit of history

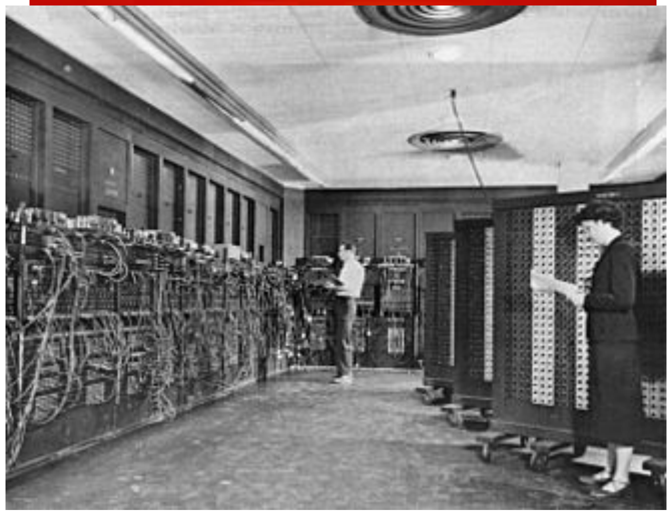
- Very concept of Monte Carlo comes in the XVIII century (Buffon, 1777, and then Laplace, 1786)
  - Monte Carlo estimate of π
- Concept of MC is much older than real computers
  - one can also implement the algorithms manually, with dice (= Random Number Generator)



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



Geant4

Monte Carlo codes and Geant4

MCNP (neutrons mainly)

Penelope (e- and gamma)

PETRA (protons)

EGSnrc (e- and gammas)

PHIT (protons/ions)

FLUKA (any particle)

Geant4

GEometry ANd Traking

Geant4 - a simulation toolkit
Nucl. Inst. and Methods Phys. Res. A, 506:250-303 (2003);

Geant4 developments and applications
Transaction on Nuclear Science 53, 270-278 (2006);

Recent developments in Geant4
Nucl. Inst. and Methods Phys. Res. A, 835:186-225 (2016)

Facts about Geant4

Geant4 started at CHEP 1994 @ San Francisco

"Geant steps into the future", R Brun et al.

"Object oriented analysis and design of a Geant based detector simulator", K Amako et al

Dec '94 - CERN RD44 project starts

Apr '97 - First alpha release

Jul '98 - First beta release

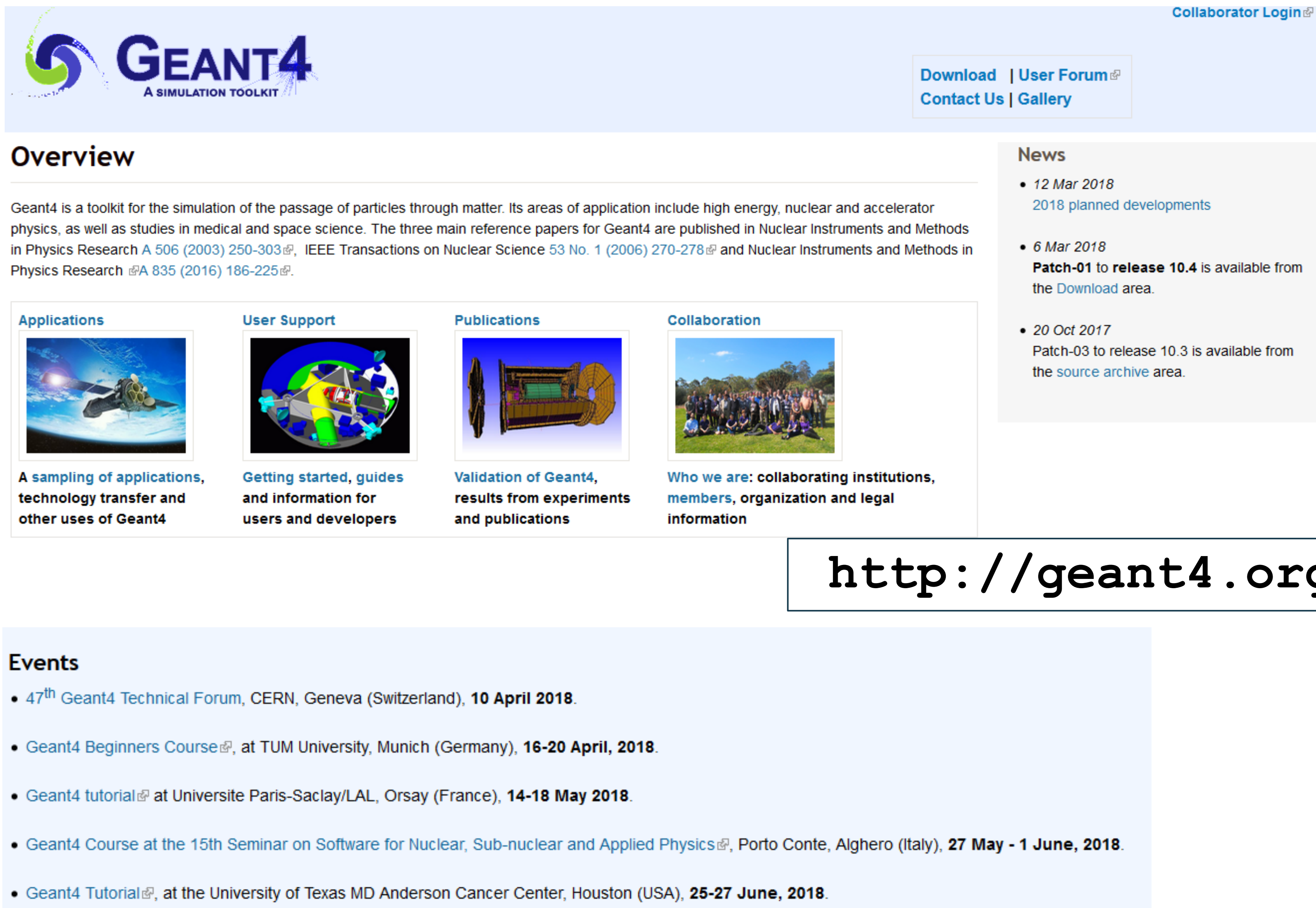
Dec '98 - First Geant4 public release - version 1.0

....

April, 2019 - Geant4 10.5 patch 01 release ← Current version in the VM

We currently provide one public release every year (next release December 9th)

Facts about Geant4



Collaborator Login

GEANT4 A SIMULATION TOOLKIT

Download | User Forum
Contact Us | Gallery

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

News

- *12 Mar 2018*
  2018 planned developments
- *6 Mar 2018*
  Patch-01 to **release 10.4** is available from the Download area.
- *20 Oct 2017*
  Patch-03 to release 10.3 is available from the source archive area.

Events

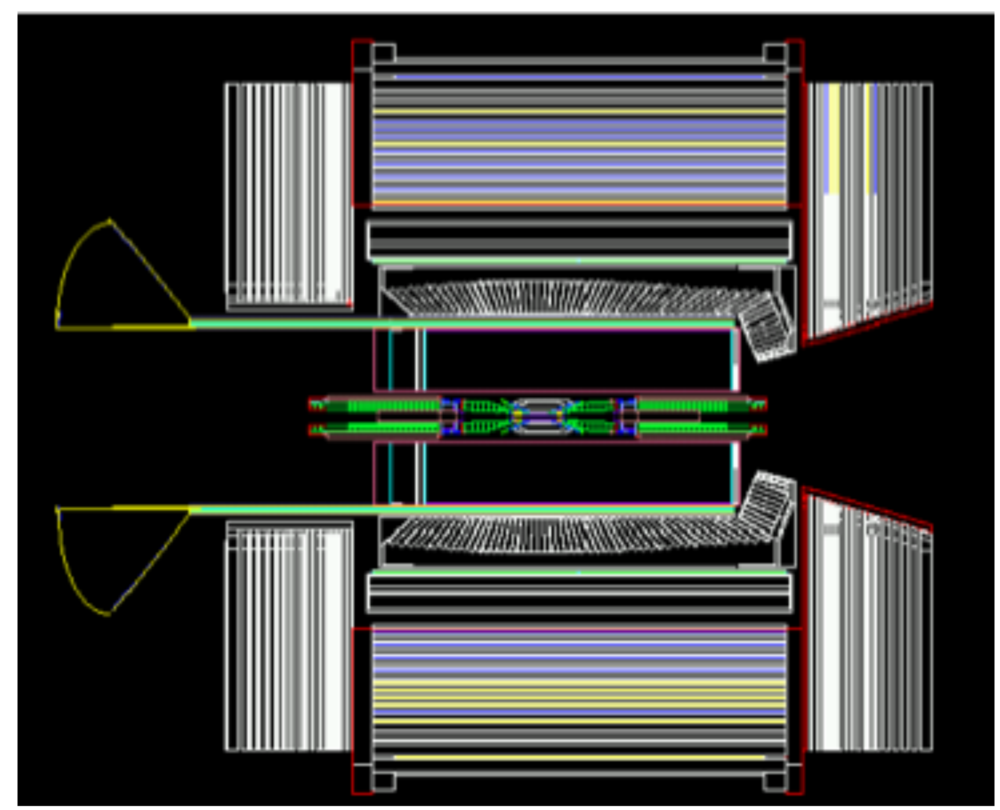
- 47th Geant4 Technical Forum, CERN, Geneva (Switzerland), **10 April 2018**.
- Geant4 Beginners Course, at TUM University, Munich (Germany), **16-20 April, 2018**.
- Geant4 tutorial at Universite Paris-Saclay/LAL, Orsay (France), **14-18 May 2018**.
- Geant4 Course at the 15th Seminar on Software for Nuclear, Sub-nuclear and Applied Physics, Porto Conte, Alghero (Italy), **27 May - 1 June, 2018**.
- Geant4 Tutorial, at the University of Texas MD Anderson Cancer Center, Houston (USA), **25-27 June, 2018**.

`http://geant4.org`

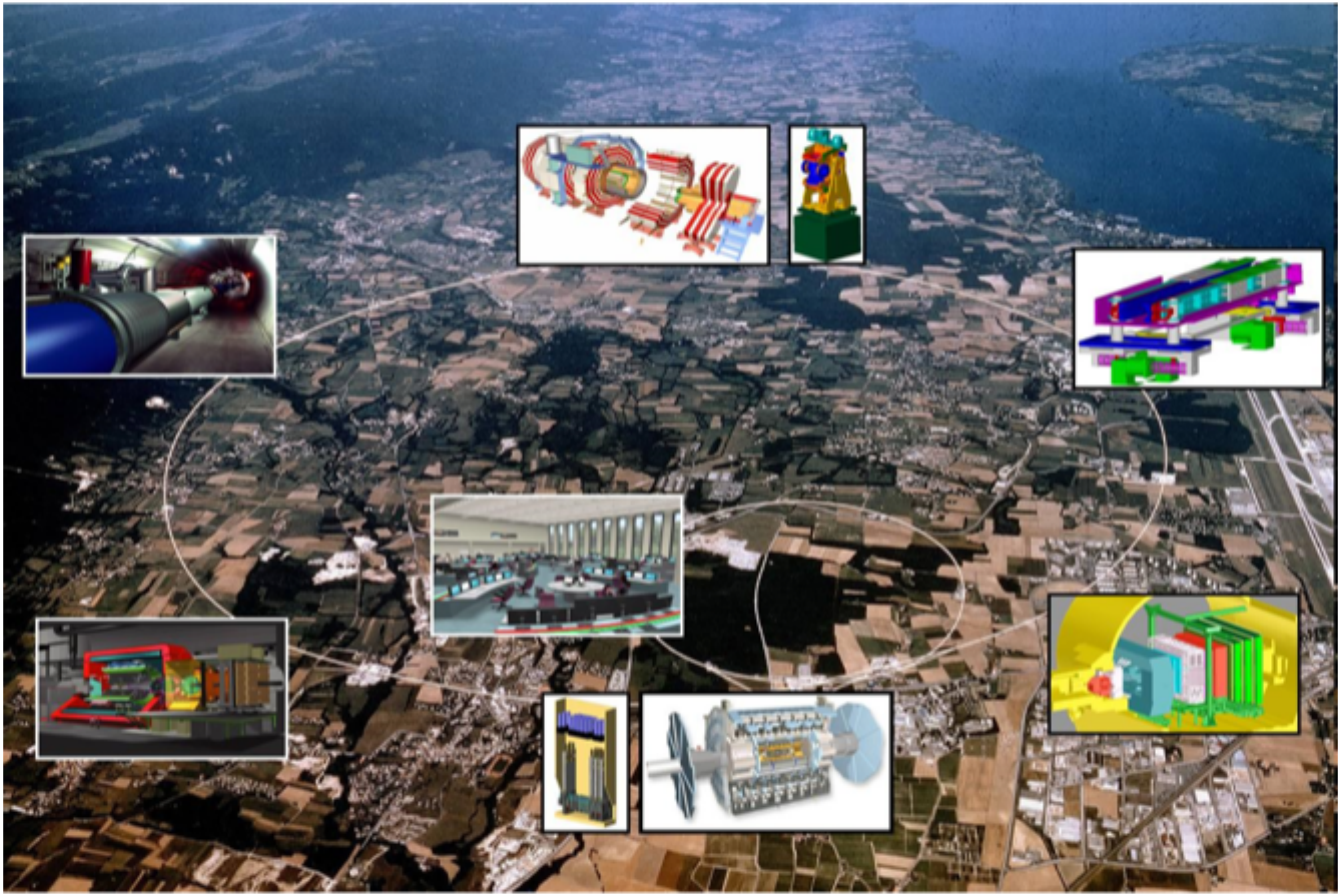
- **Code** and **documentation** available in the main **web page**
- Regular **tutorial courses** held worldwide

Facts about Geant4

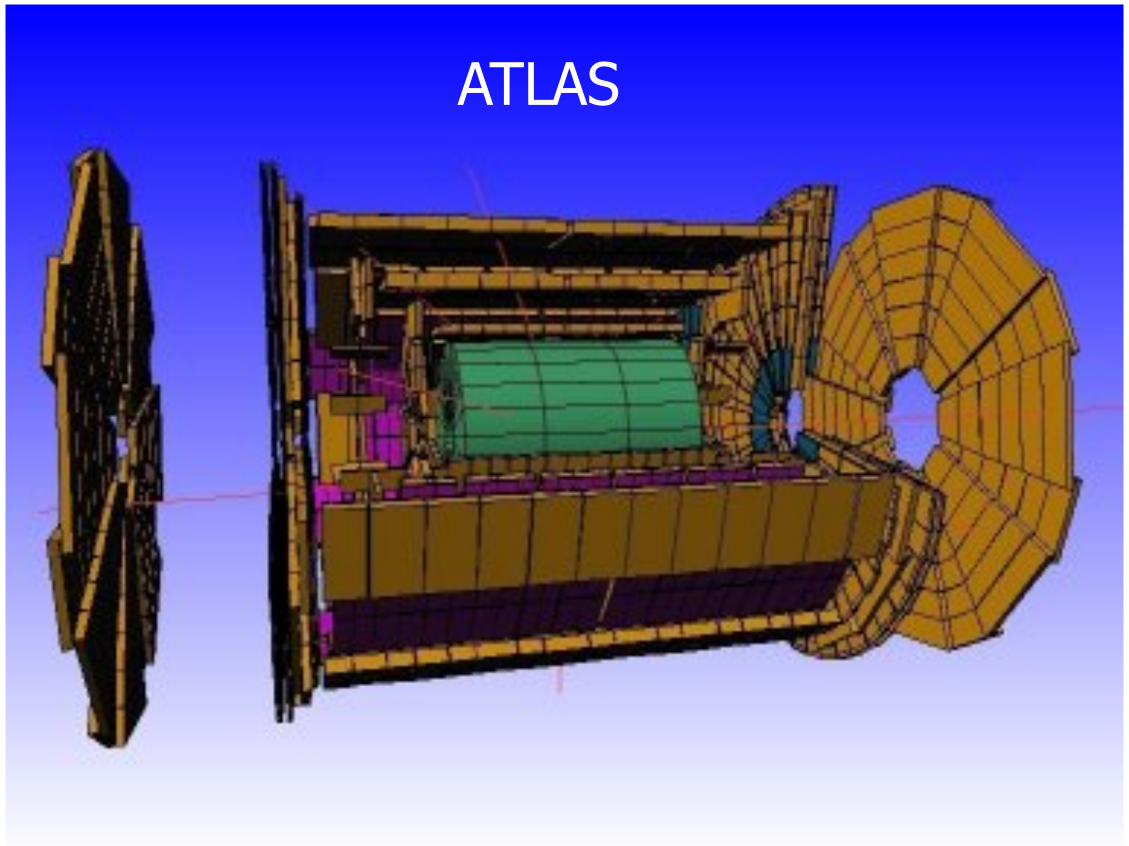
- BaBar is the pioneer HEP experiment in use of OO technology and the first customer of Geant4
  - During the R&D phase of Geant4 a lot of evaluable feedbacks were provided
- BaBar started its simulation production in 2000 and had produced more than 10 bilion events at more than 20 sites in Europe and North America.



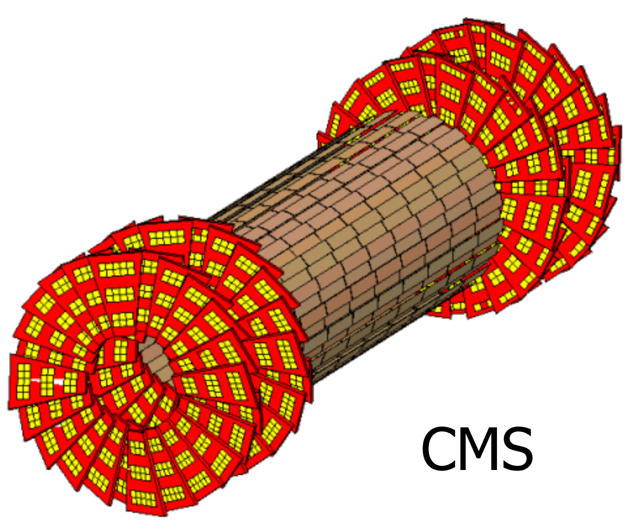
Facts about Geant4

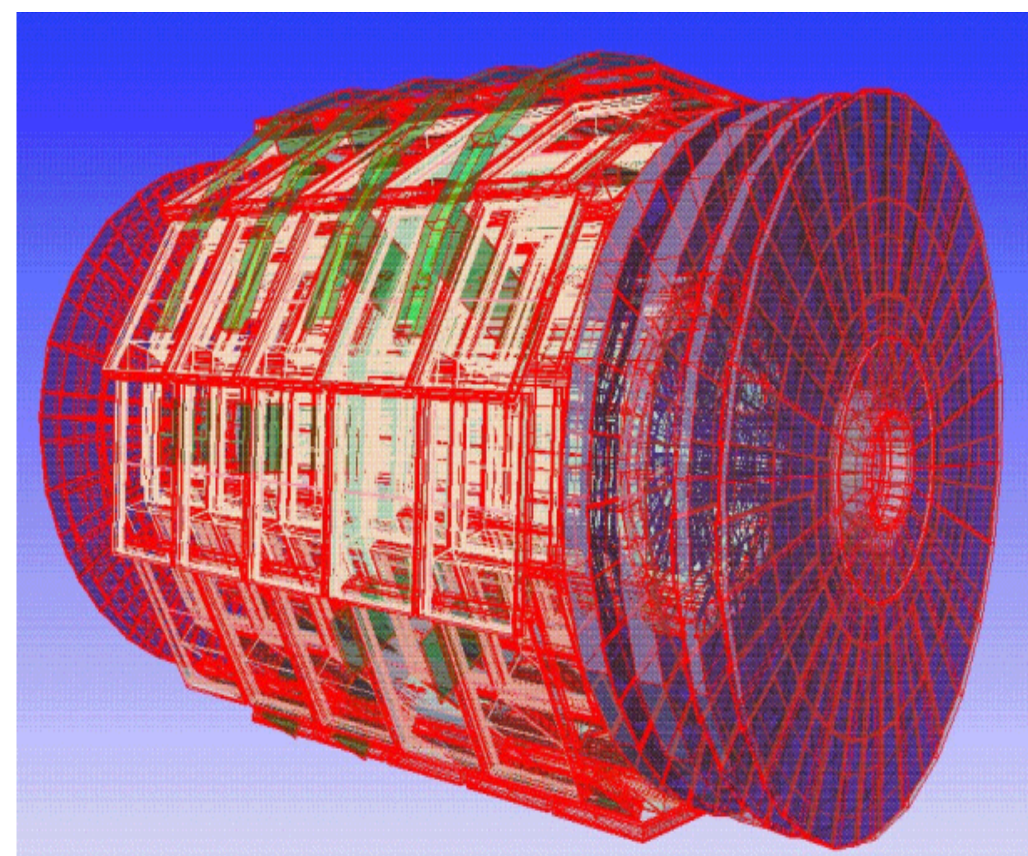


LHC @CERN



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

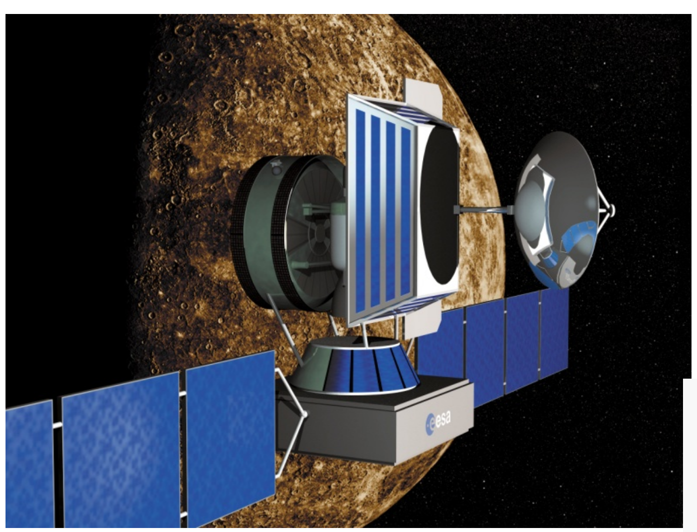




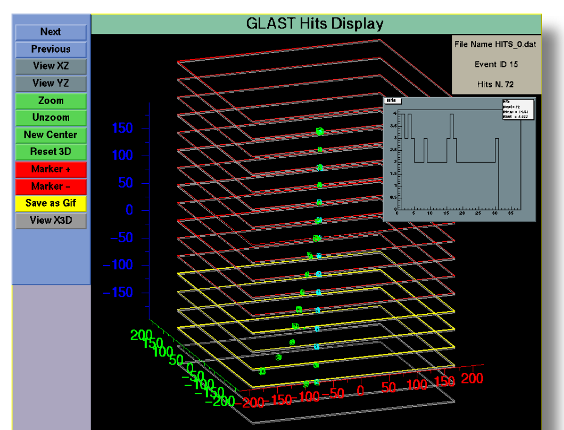
- Benchmark with test-beam data
- Key role for the Higgs searches

Space application

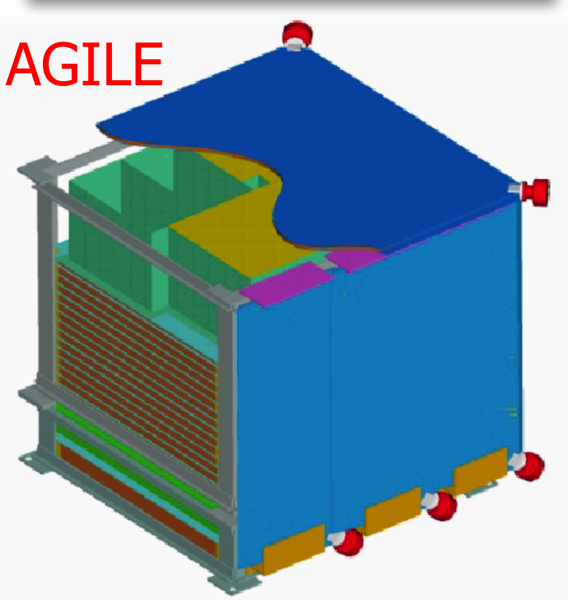
- Satellites (γ astrophysics, planetary sciences)
- Funding from ESA

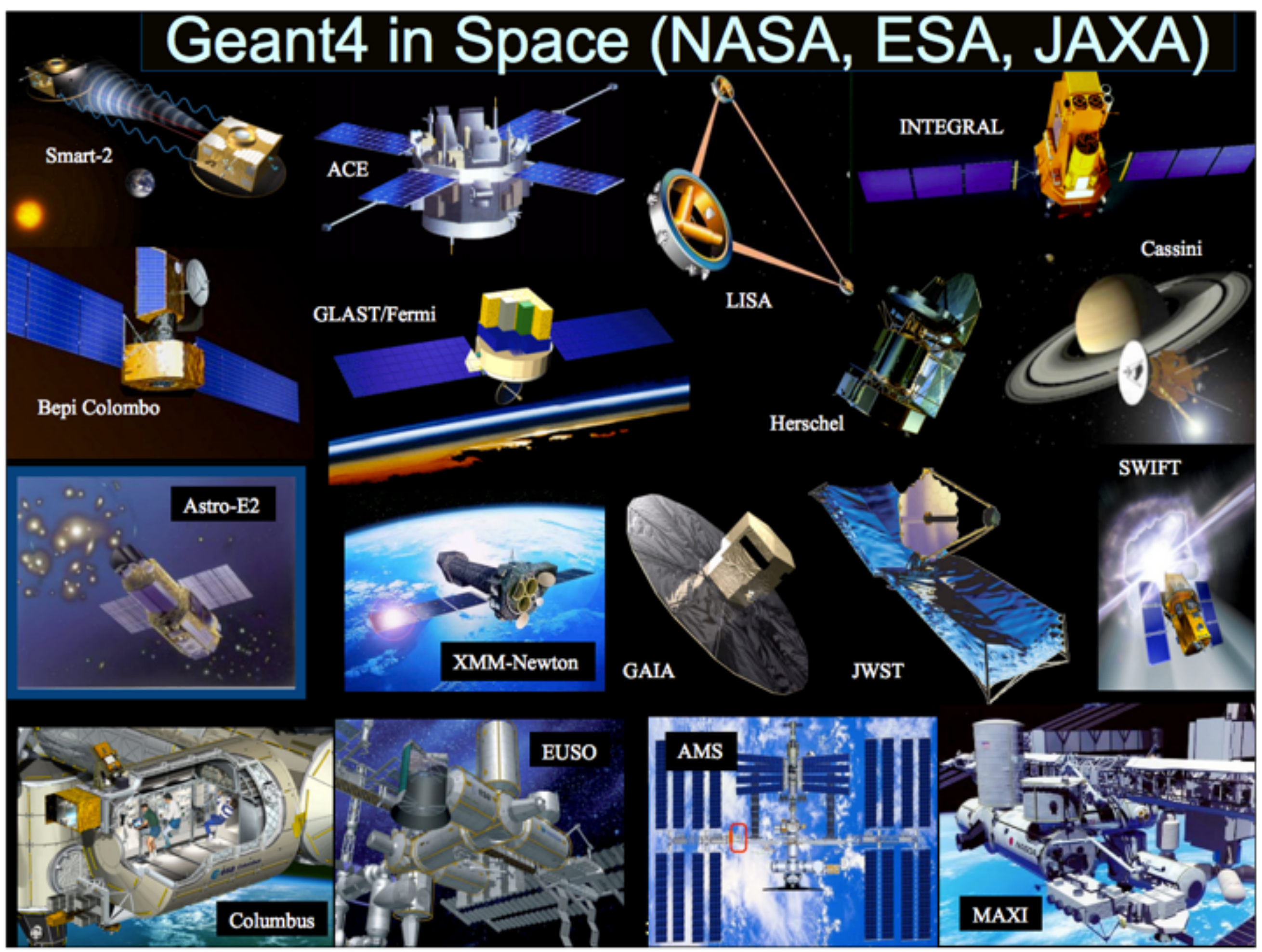


GLAST



AGILE



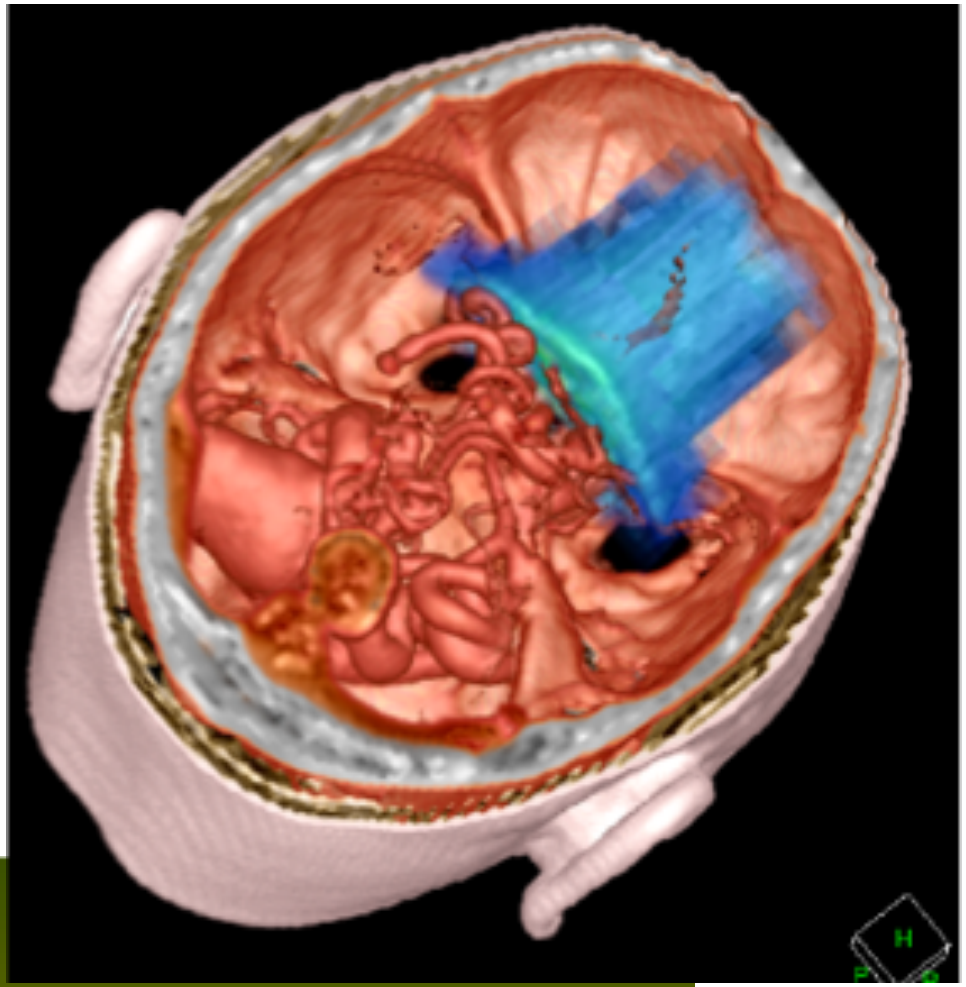


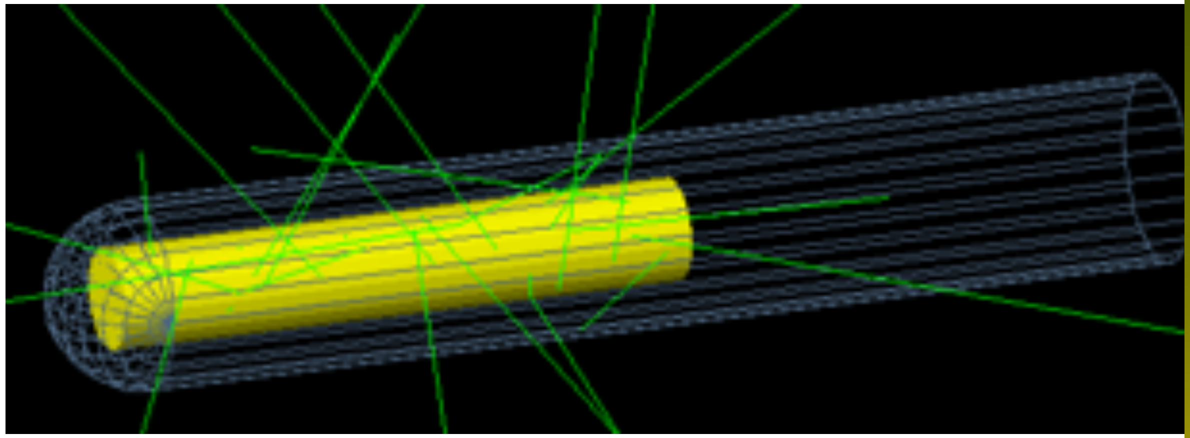
<u>Typical telescope:</u>
Tracker
Calorimeter
Anticoincidence

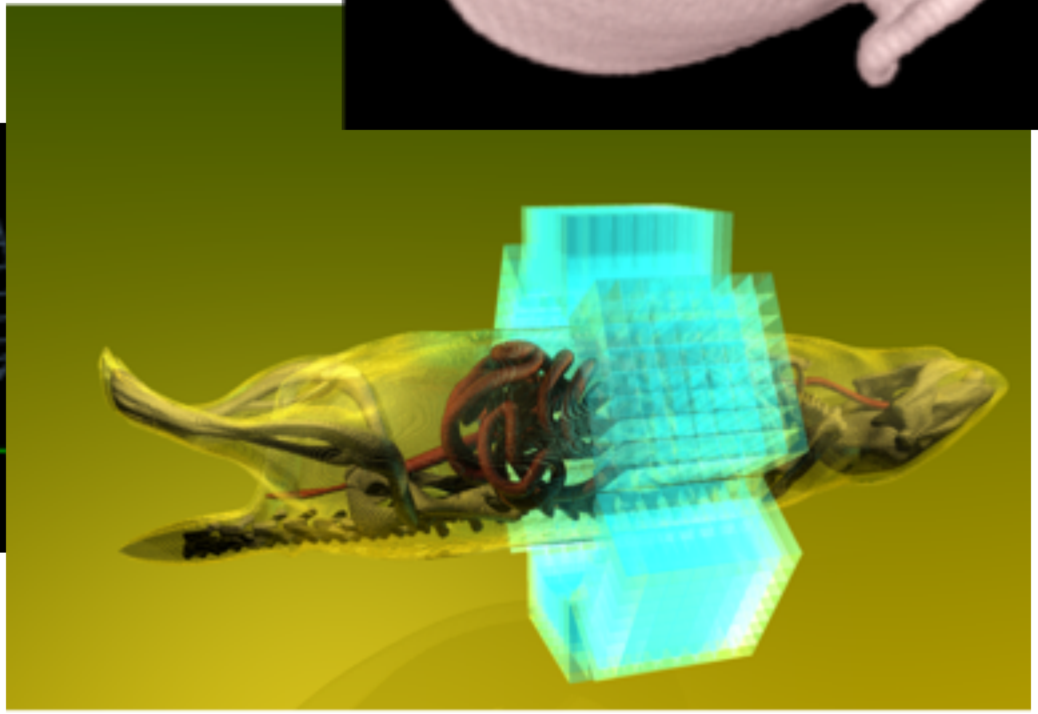
Facts about Geant4

Major use cases

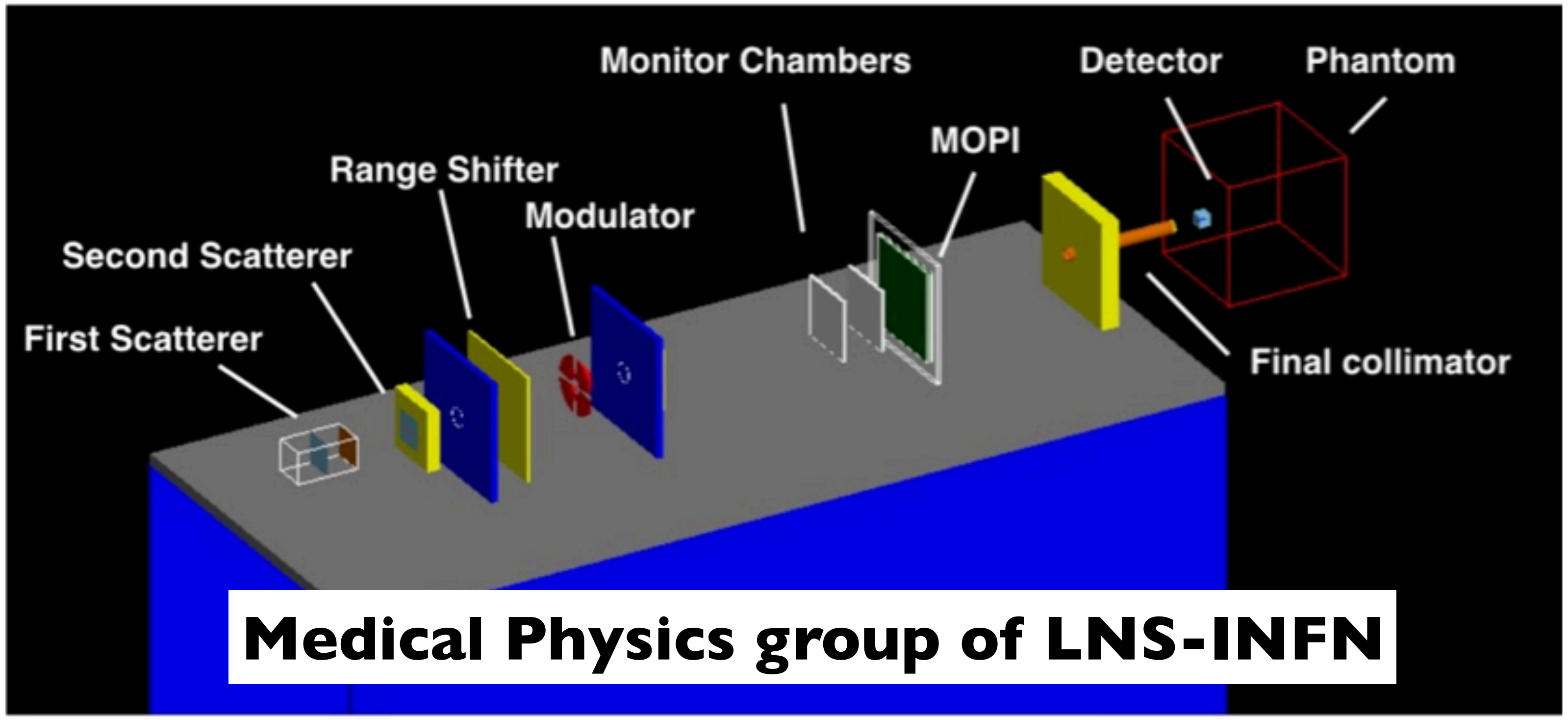
- Beam therapy
- Brachytherapy
- Imaging
- Irradiation study
- Nuclear medicine and radioisotopes
- Biological damage



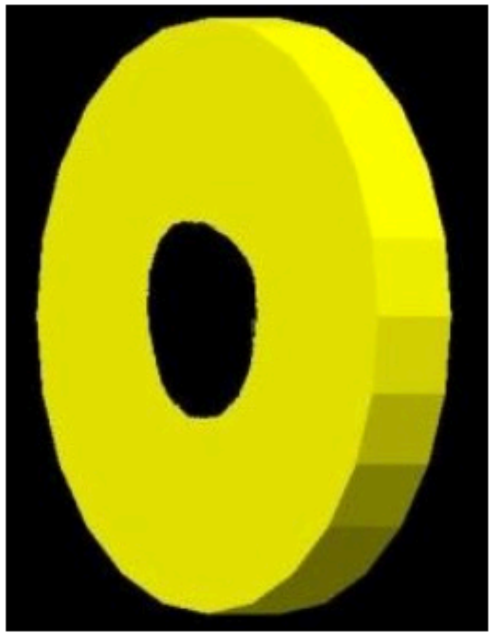




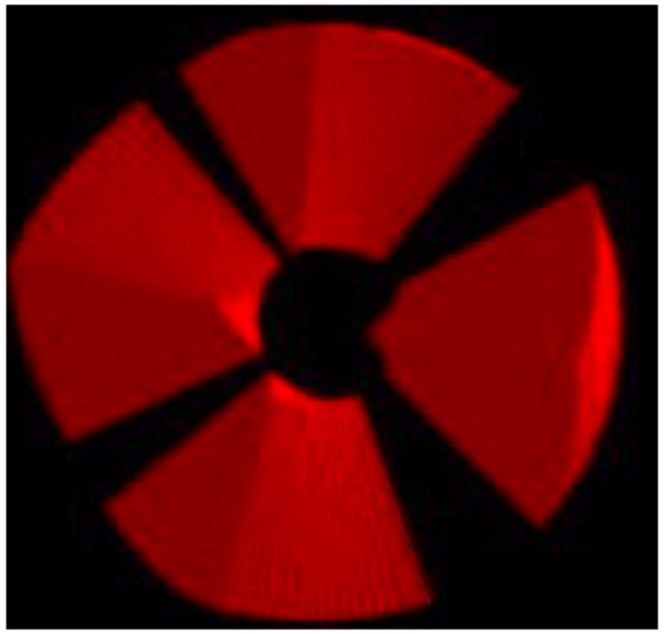
Medical Physics



- **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`



Final Collimator



Modulator

Basic concepts and Geant4 capabilities

Geant4 overview

C++ language

Object Oriented

Open Source

Once per year released

It is a toolkit, i.e. a collection of tools the User can use for his/her simulation

Consequences:

- There are not such concepts as "Geant4 defaults"
- You must provide the necessary the necessary information to configure your simulation
- You must choose the Geant4 tool to use

Guidance: many examples are provided:

- Basic examples: overview of the Geant4 tools
- Extended examples: showing specific Geant4 functionalities
- Advanced Examples: Geant4 tools in real-life applications

Geant4 basics

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 the Geant4 kernel to control your simulation

To visualise your simulation set-up and particles

To produce histograms, tuples, etc. to be further analysed

Multi-threading

Geant4 supports multi-thread (MT) approach that can be used in multi-cores machines

Simulation is automatically split on a event-by-event basis

- Different events are processes by different cores
- Unique copy of Geometry and Physics
  - All cores have them as read-only

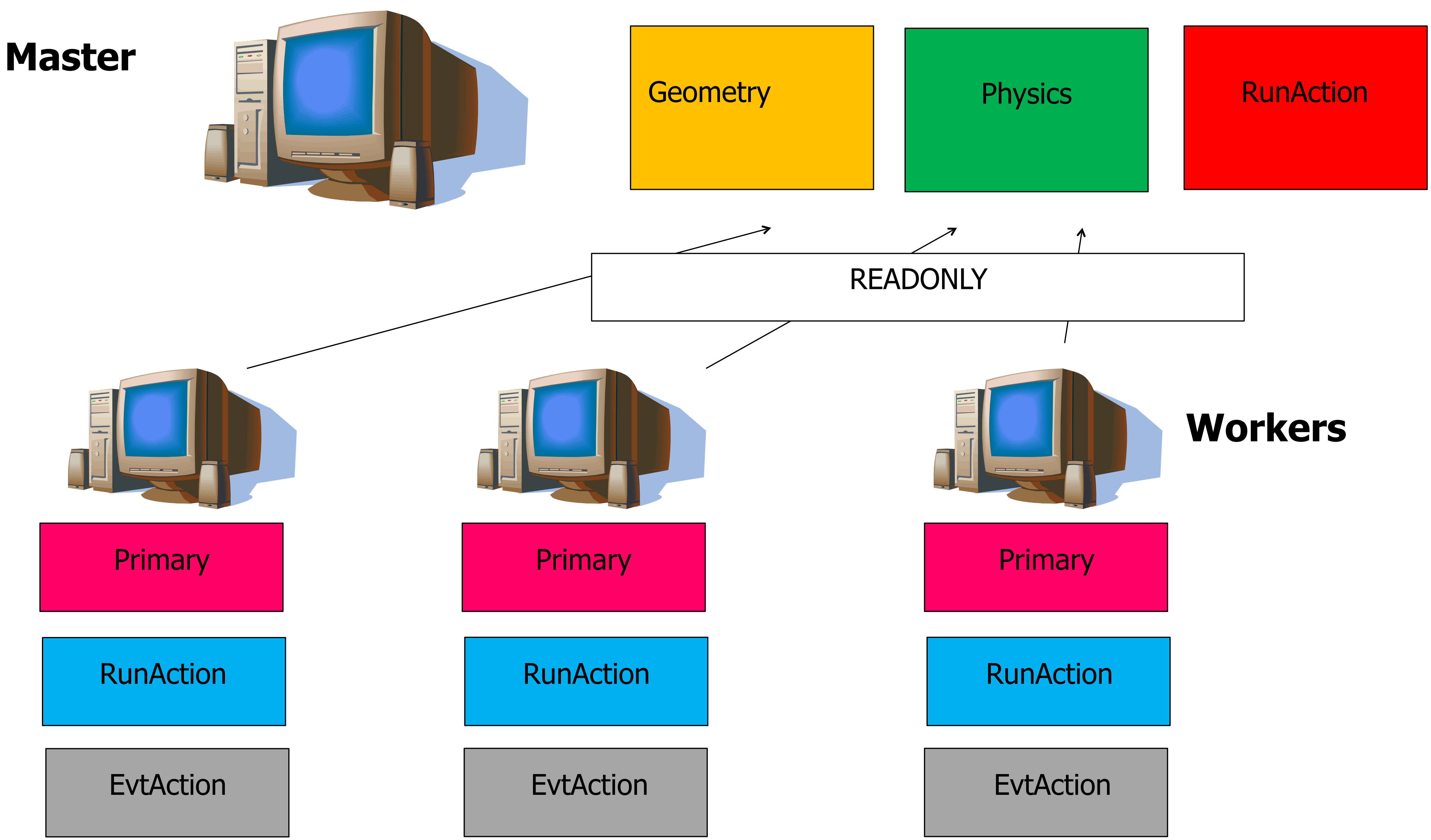
Functionality in place since Geant4 10.0

- Backward compatible with the sequential mode

MT programming requires some cares

- Need to avoid conflicts between threads

Multi-threading



Files composing a Geant4 application

Main() file

Sources files (*.cc)

usually included in the **src/** folder

Header files (*.hh)

usually included in the **include/** files

Three classes are necessary (with the Main.cc ons), each typically in a dedicated header/source pair of files

The PrimaryGeneratorAction (.cc and .hh)

The DetectorConstruction (.cc and .hh)

The PhysicsList (.cc and .hh)

Mandatory User's classes – v1

Initialisation classes

Invoked at the initialisation

G4VUserDetectorConstruction
G4VUserPhysicsList

Action classes

Invoked during the execution loop

G4VUserPrimaryGeneratorAction

G4UserRunAction
G4UserEventAction
G4UserTrackingAction
G4UserStackingAction
G4UserSteppingAction

Mandatory User's classes – v2 (MT)

Initialisation classes

Invoked at the initialisation

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 main() file

The main()

- Geant4 does not provide a main() file
  - Geant4 is a toolkit!
  - The main() is part of the User application
- In his/her main(), the user must:
  - Construct the **G4RunManager**
  - Notify the **G4RunManager** the mandatory user classes derived from:
    - ✓ **runManager -> SetUserInitialization (new MyApplicationDetectorConstruction)**

The main()

The user MAY define in his/her main():

- Optional user action classes
- VisManager, (G)UI session

The User has also to take care of retrieve and save the relevant information from the simulation (Geant4 will not do that by default)

Do not forget to delete the **G4RunManager** at the end

An example of sequential main()


```
{
 // Construct the default run manager
 G4RunManager* runManager = new G4RunManager;

 // 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 MyPrimaryGeneratorAction);

 // Set optional user action classes
 MyEventAction* eventAction = new MyEventAction();
 runManager -> SetUserAction(eventAction);
 MyRunAction* runAction = new MyRunAction();
 runManager -> SetUserAction(runAction);
}
```

Methods for Users classes

G4UserRunAction

```
BeginOfRunAction(const G4Run*) // book histos

EndOfRunAction(const G4Run*)   // store histos
```

G4UserEventAction

```
BeginOfEventAction(const G4Event*) //initialize event

EndOfEventAction (const G4Event*) // analyze event
```

G4UserTrackingAction

//decide to store/not store a given track

```
PreUserTrackingAction(const G4Track*)

PostUserTrackingAction(const G4Track*)
```

Methods for Users classes

G4UserSteppingAction

```
UserSteppingAction(const G4Step*)
```
//kill, suspend, pospone the track, draw the step, …

G4UserStackingAction

```
PrepareNewEvent()
```
//reset priority control

```
ClassifyNewTrack(const G4Track*)
```
// Invoked when a new track is registered (e.g. kill, pospone)

```
NewStage()
```
// Invoked when the Urgent stack becomes empty (re-classify, abort event)

Selection of physics processes and optional capabilities

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 particles production threshold (in terms of range)

Methods of **`G4VUserPhysicsList`**:

- **`ContructParticles()`**
- **`ConstructProcesses()`**
- **`SetCuts()`**

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 modelling 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

Optional (G)UI

- In your main(), taking into account your computer environment, instantiate a **`G4UISession`** provided by Geant4 and invoke its **`SessionStart()`** method:

  - **`mysession -> SessionStart();`**

- Geant4 provides:

  - G4UIterminal;
  - csh or tcsh like shell
  - G4UIBatch
  - Bach job with macro files

Interfaces to various graphics drivers
Dawn
Wired
RayTracer
OpenGL
OpenInventor
VRML

Summary: general recipe for novice Users

A general recipe

- **Design your application** .... requires preliminary thinking (what is supposed to do?)
- Create your derived **mandatory user classes**
  - MyDetectorConstruction
  - MyPhysicsList
  - MyPrimaryGeneratorAction
- Create **optional derived user action classes**
  - MyUserRunAction, MyUserEventAction
- **Create your main()** file
  - Instantiate G4RunManager
  - Notify the RunManager of your mandatory and optional user classes
  - Optionally initialise your favourite User Interface and Visualisation

Experienced users may do much more,

but the conceptual process is still the same…

Thank you

User Interface

Three ways of steering the simulation

- hard-coded application
  - no user interaction
  - everything specified in the C++ source
  - re-compile needed to apply changes
- batch session
  - commands in external macro file
- interactive session
  - real-time command input by user
  - textual, graphical, (network-based)

Select the way of control

main.cc

```
int main(int argc, char** argv) {
    G4RunManager* runManager = new G4RunManager;
    runManager->SetUserInitialization(new MyDetectorConstruction());

    // Physics list
    G4VModularPhysicsList* physicsList = new MyPhysicsList;
    physicsList->SetVerboseLevel(1);
    runManager->SetUserInitialization(physicsList);

    // User actions initialization
    runManager->SetUserInitialization(new MyActionInitialization());
```

Insert the control code here!

```
    delete runManager;
}
```

Hard-coded C++


```
    // ...
    // User actions initialization
    runManager->SetUserInitialization(new MyActionInitialization());

    runManager->Initialize();
    runManager->BeamOn(!000);


    // ...
    delete runManager;
}
```

- You must initialize and start the run by issuing "beam on".
- Even the number of events has to be specified!

Batch session


```
    // ...
    // User actions initialization
    runManager->SetUserInitialization(new MyActionInitialization());

    G4UImanager* UImanager = G4UImanager::GetUIpointer();
    G4String command = "/control/execute ";
    G4String fileName = argv[1];
    UImanager->ApplyCommand(command + fileName);

    // ...
    delete runManager;
}
```

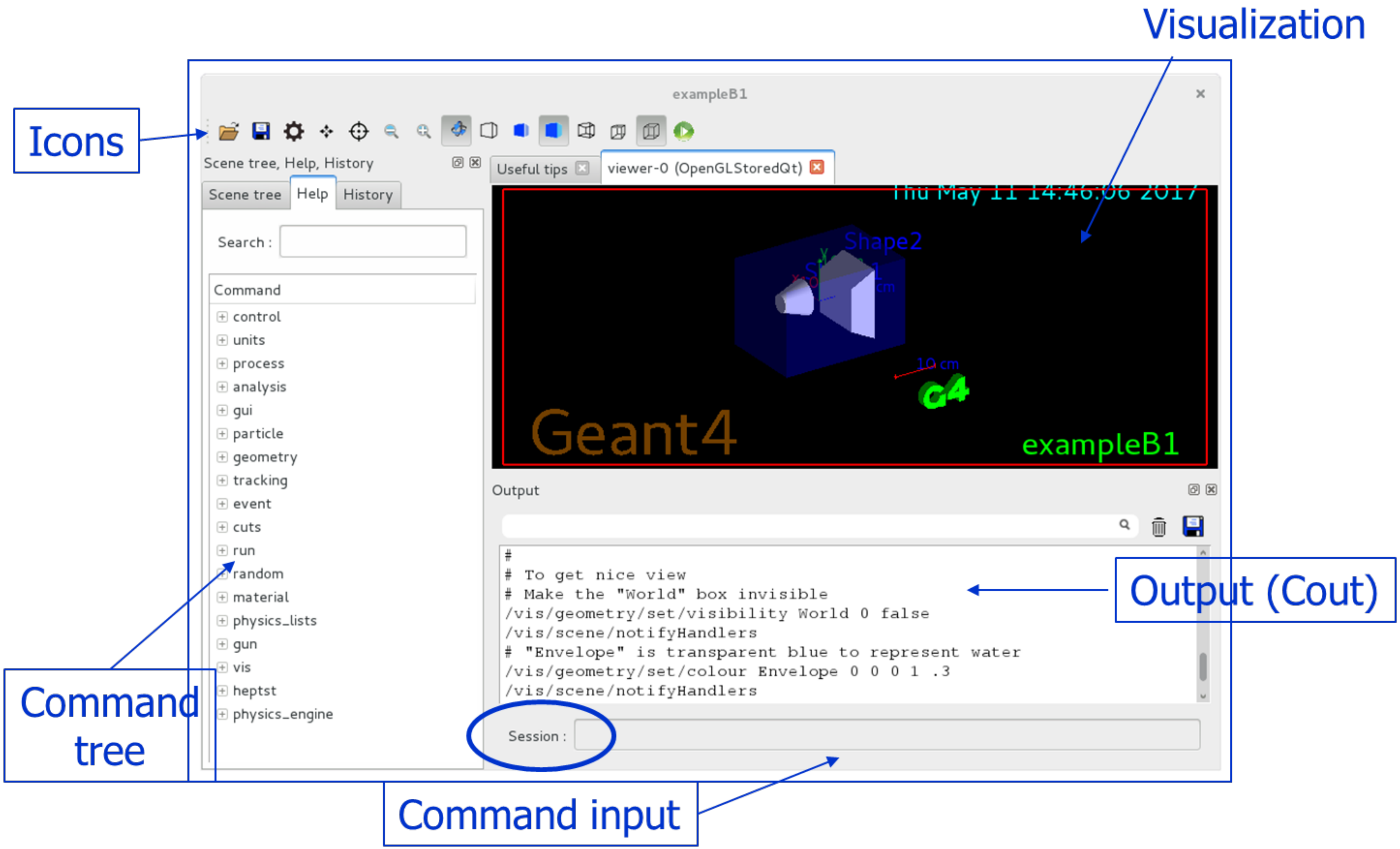
- This example gets the file name of the macro from the command-line argument:

```
./myApplication my-macro.mac
```

Interactive sessions

- Many different session types, inheriting from **`G4UIsession`** class:
  - command-line based (dumb terminal)
  - graphical
  - special
  - your own? ☺
- **`G4UIExecutive`** class enabling to select the appropriate session at runtime, based on the environment variables (recommended)

G4UIQt session



Universal batch/interactive approach


```
int main(int argc, char** argv) {
    // ...
    if (argc == 1) {
        // Interactive mode
        G4UIExecutive* ui = new G4UIExecutive(argc, argv);
        ui->SessionStart();
        delete ui;
} else {
        // Batch mode
        G4UImanager* UImanager = G4UImanager::GetUIpointer();
        G4String command = "/control/execute ";
        G4String fileName = argv[1];
        UImanager->ApplyCommand(command + fileName);
}
    // ...
```

No argument (→ `if (argc == 1)`)

One argument (or more) (→ `else`)

- Mode selected based on application argument:
  - **No** argument = **interactive** mode
  - **One** argument = **batch** mode

Example UI commands

- **`/run/verbose 1`** – sets how much output the run manager will print (similar for other classes)
- **`/run/initialize`** – initializes the run (constructing the geometry, physics and preparing the user actions)
- **`/run/beamOn 100`** – starts a run with 100 events
- **`/control/execute macroName`** – run the commands in a macro file
- A complete list of built-in commands is available in the Geant4 Application Developers Guide, Chapter 7.1 (http://geant4.cern.ch/G4UsersDocuments/UsersGuides/ForApplicationDeveloper/html/Control/commands.html)

Visualization

- To enable visualization, instantiate a **G4VisExecutive** and invoke its Initialize() method
- Geant4 provides interfaces to various graphics (and "graphics") drivers:
  - **OpenGL (+Qt)**
  - HepRApp
  - Dawn
  - Wired
  - RayTracer
  - OpenInventor
  - VRML
  - ....

```
#include <G4VisExecutive.hh>
// ...

int main(int argc, char** argv) {
    // ...
    G4VisManager* visManager = new G4VisExecutive;
    visManager->Initialize();
    G4UIExecutive* ui = new G4UIExecutive(argc, argv);
    ui->SessionStart();
    delete ui;
    delete visManager;
    delete runManager;
}
```

Thank you