# The Monte Carlo approach and the Geant4 toolkit

*III Geant4 International and GPU programming school*
*INFN-LNS - Catania (I)*
*9-13 Novembre 2015*

Generals on Monte Carlo

Basic capability of Geant4

Basic structure of the Geant4 components

Generals on Monte Carlo

Basic capability of Geant4

Basic structure of the Geant4 components

At the end of these three days

Installation

Configuration

Generation of particles

Geometry and materials

Physics

Retrieve the information

# Finding the material




- Pablo Cirrone, Giuliana Milluzzo, Luciano Pandola, Giada Petringa, Francesco Romano, Antonella Tramontana*
  INFN-Laboratori Nazionali del Sud - Catania, (I)
  *EBG MedAustron GmbH, Wiener Neustadt (A)


- Official tutorial and school regularly offered: see the geant4 web pages
- Official Geant4 web pages: www.cern.ch/geant4
- The Italian Geant4 group: http://geant4.lngs.infn.it/

# The Monte Carlo method

*A very general introduction*

# Brief history and principles of Monte Carlo



- Comte de Buffon (1777): needle tossing experiment to calculate the π;
- Laplace (1886): random points in a rectangle to calculate π;
- Fermi (1930): random approach to calculate the properties of the newly discovered neutron;
- Manhattan project (40's): simulations during the initial developments of thermonuclear weapons;
- Von Neumann and Ulam coined the term 'Monte Carlo' (1949);
- Exponential growth of the electronic computers (40's-60's);
- Berger (1963): first complete coupled electron-photon transportation code 'ETRAN'.

# Brief history and principles of Monte Carlo



**It is a mathematical approach using a sequence of random numbers to solve a problem**

*"If we are interested in a parameter of, i.e., an equation:*

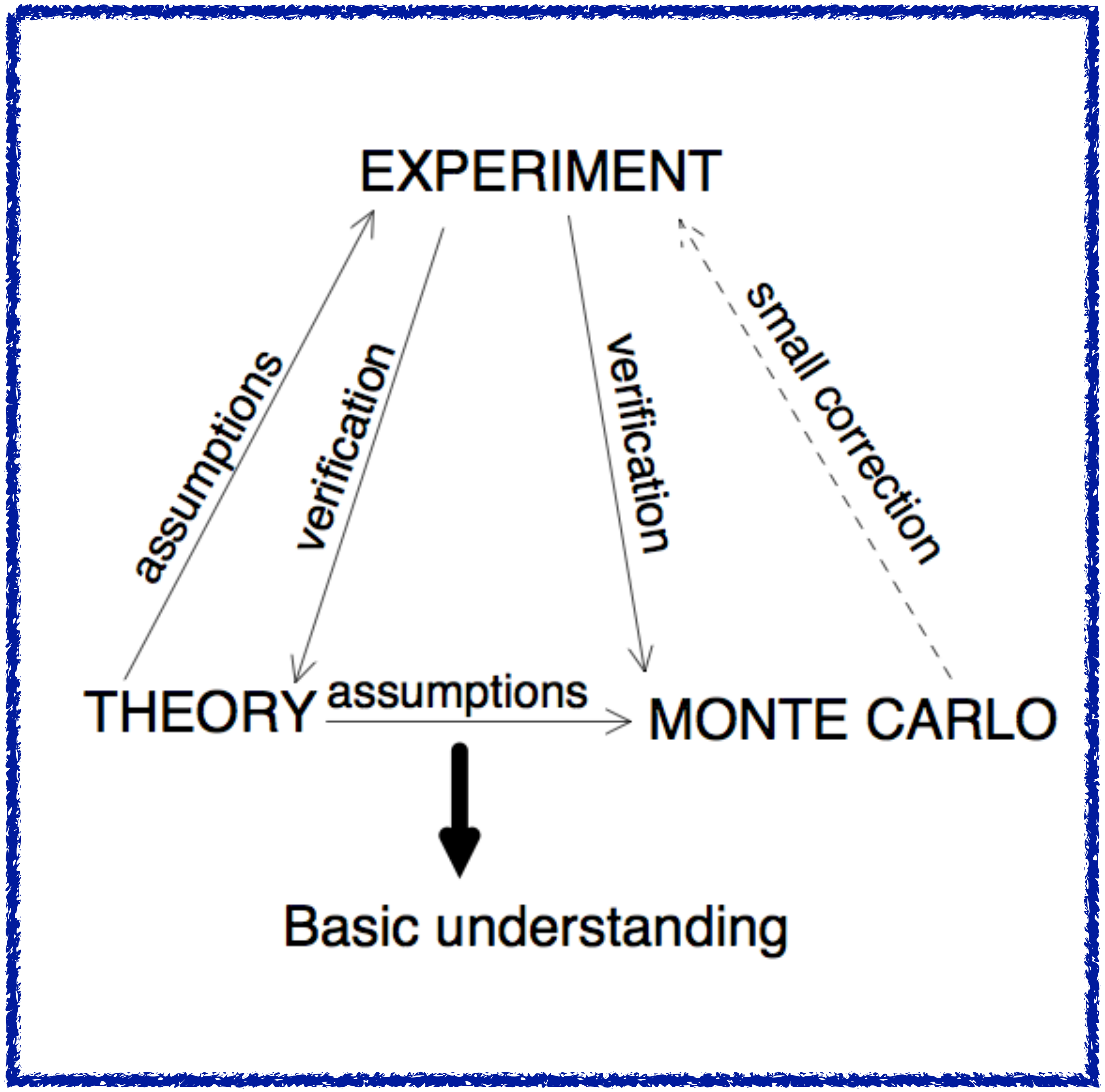
*we must construct a big number of this equations, using different random numbers, and*

*estimate the parameter and its variance"*

**A. F. Bielajew, 2001**

# Brief history and principles of Monte Carlo





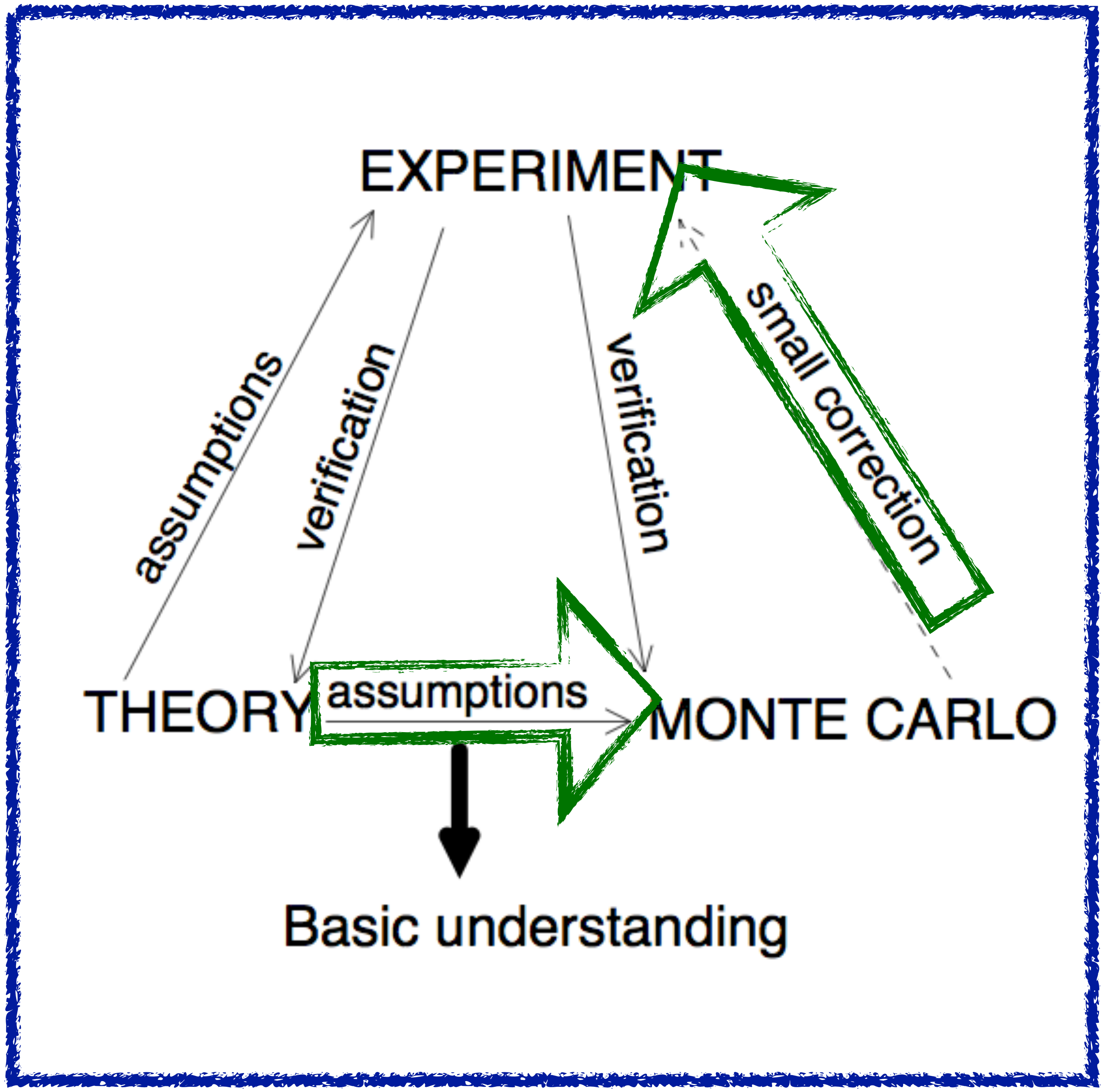

Monte Carlo helps

To verify a theory if physics models are in development

To develop or verify an experiment in the other case

# Brief history and principles of Monte Carlo






In particle transport, if particles interaction models are known, MC can be used to calculate the parameters of the motion equations in a given configuration

# Brief history and principles of Monte Carlo

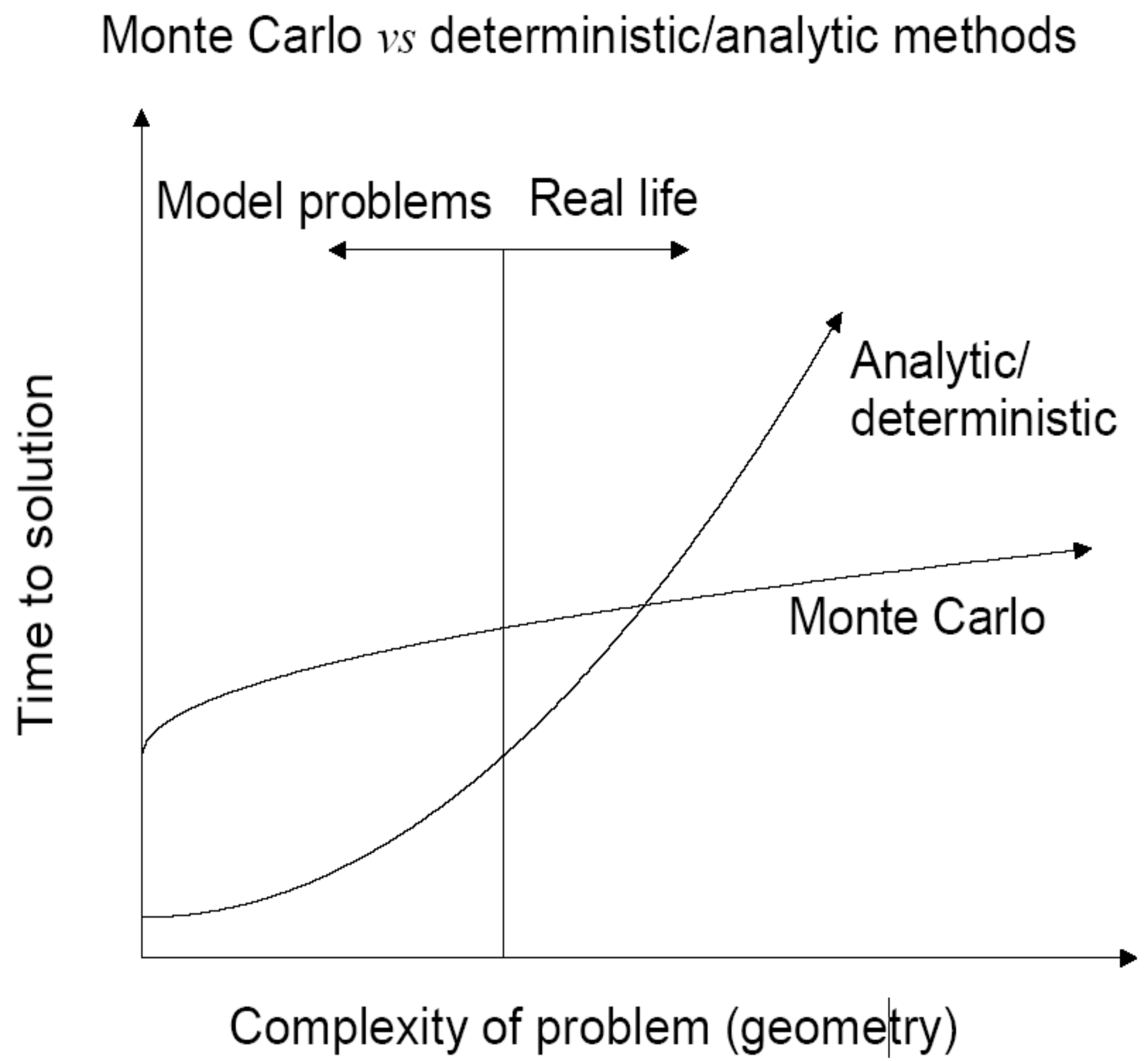


Particles are tracked one-by-one, step-by-step and,
after a reasonable number of particles,
the correct information can be extracted

The Monte Carlo is very time consuming but
........... sometime necessary and
............with many advantages

# Brief history and principles of Monte Carlo






***Plot from Alex F. Bielajew, 2001***

Mathematical proofs exist demonstrating that:

MC is the most efficient way of estimate quantity in 3D when compared to first-order deterministic method

# Is a computer necessary for a Monte Carlo Calculation ?

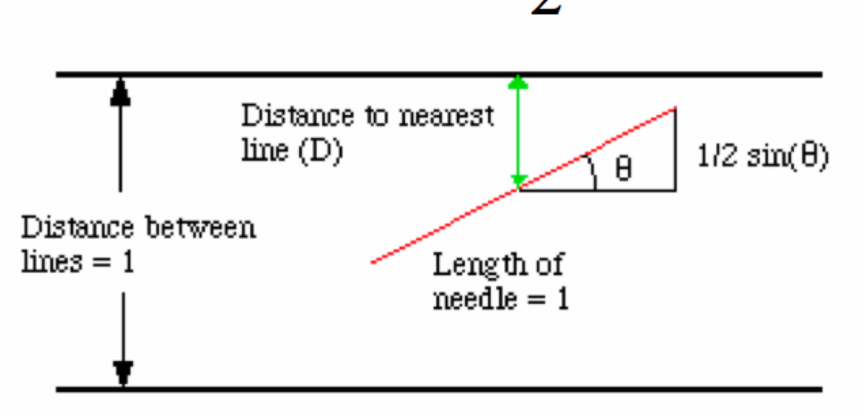
# Brief history and principles of Monte Carlo



Two variables: θ and D

$$0 \leq \vartheta \leq \pi$$

$$0 \leq D \leq \frac{1}{2}$$




Georges Louis Leclerc
Comte de Buffon
(07.09.1707.-16.04.1788.)

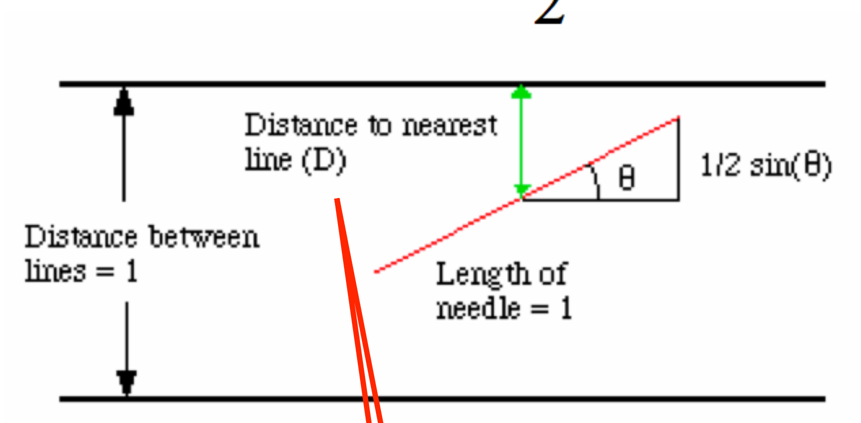
# Brief history and principles of Monte Carlo



Two variables: θ and D

$$0 \le \vartheta \le \pi$$

$$0 \le D \le \frac{1}{2}$$




Georges Louis Leclerc
Comte de Buffon
(07.09.1707.-16.04.1788.)

The needle will hit the line if the closest distance to a line D is:

$$D \le \frac{1}{2}\sin(\vartheta)$$

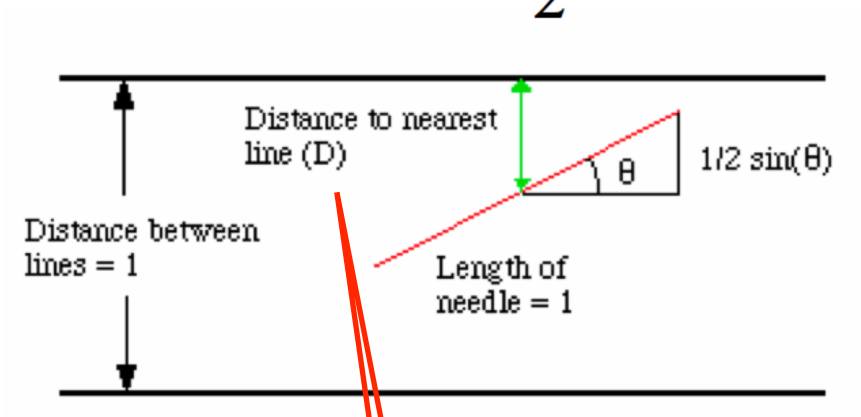
# Brief history and principles of Monte Carlo



Two variables: θ and D

$$0 \le \vartheta \le \pi$$

$$0 \le D \le \frac{1}{2}$$




Georges Louis Leclerc
Comte de Buffon
(07.09.1707.-16.04.1788.)

The needle will hit the line if the closest distance to a line D is:

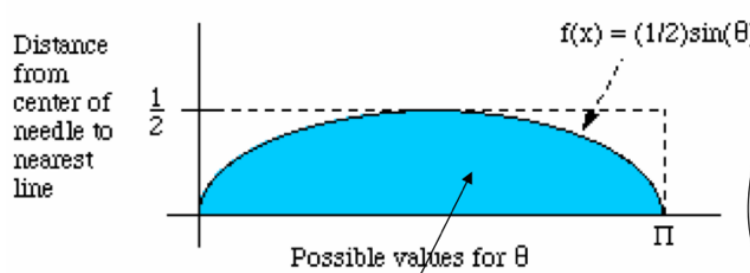
$$D \le \frac{1}{2}\sin(\vartheta)$$

The probability of an hit is the ratio of the blue area ($S_{blue}$) to the entire rectangle R

$$S_{blue} = \int_0^{\pi} \frac{1}{2}\sin(\vartheta) = 1$$

$$R = \frac{1}{2}\cdot\pi$$

$$\frac{S_{blue}}{R} = \frac{2}{\pi}$$



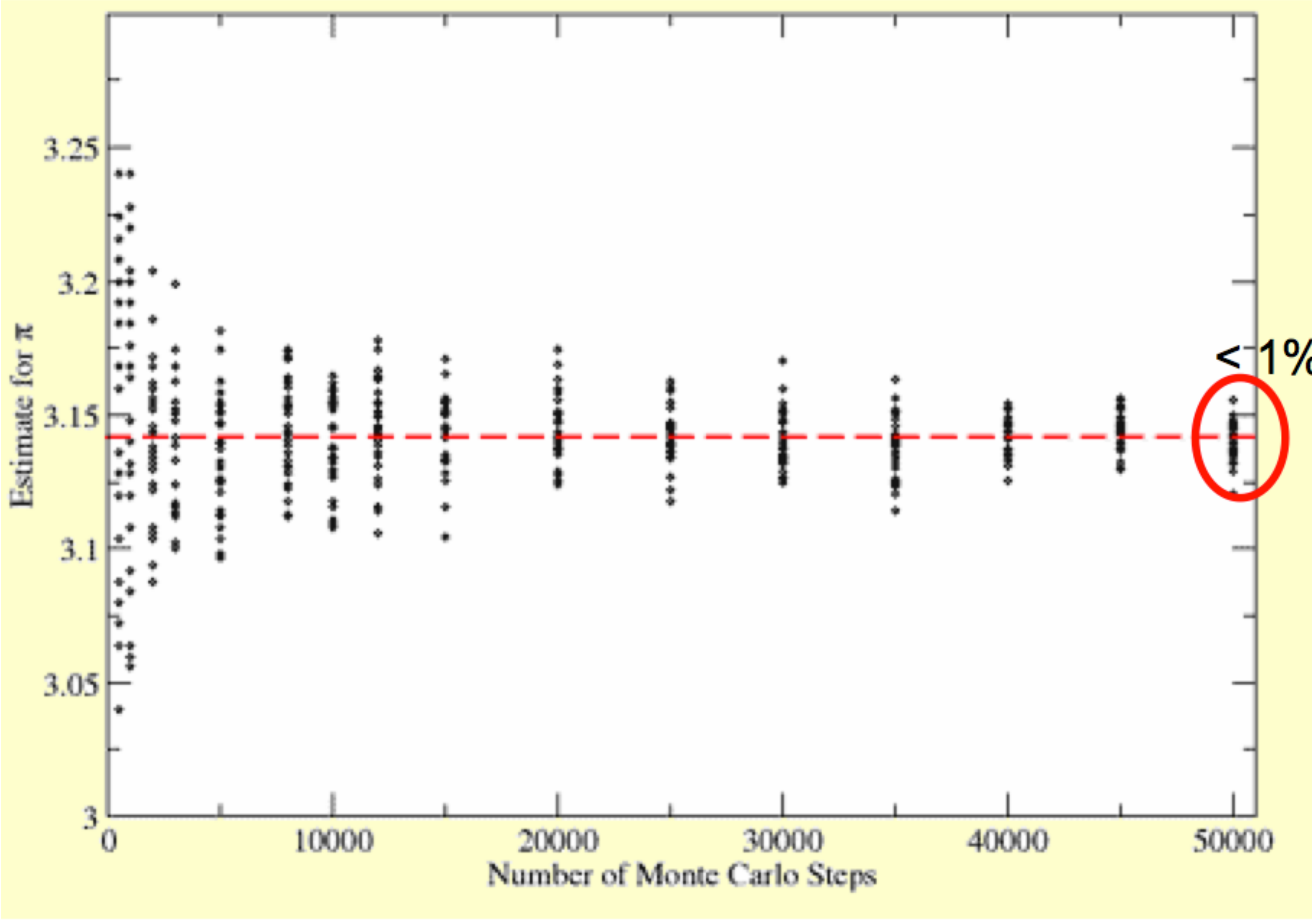

$$D \le \frac{1}{2}\sin(\vartheta)$$

$N_0$ times the needle was shot

N times the needle hit the line

$$\frac{N}{N_0} = \frac{2}{\pi}; \rightarrow \pi = 2\cdot\frac{N_0}{N}$$

Variance reductions is related to the number of shots



# Brief history and principles of Monte Carlo



JOURNAL OF THE AMERICAN STATISTICAL ASSOCIATION

*Number 247* 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

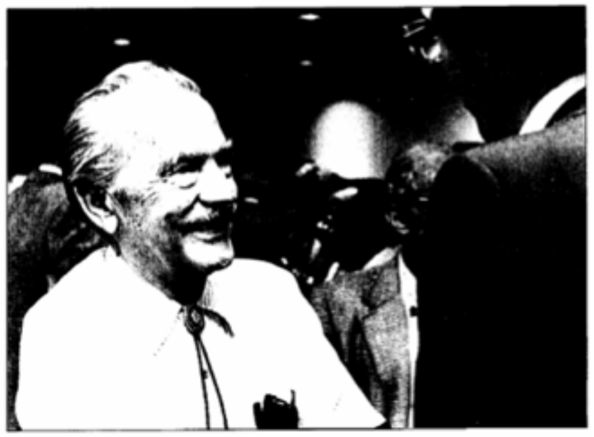
Equation of State Calculations by Fast Computing Machines

NICHOLAS METROPOLIS, ARIANNA W. ROSENBLUTH, MARSHALL N. ROSENBLUTH, AND AUGUSTA 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

## Fermi's work on pion-proton phase shift analysis

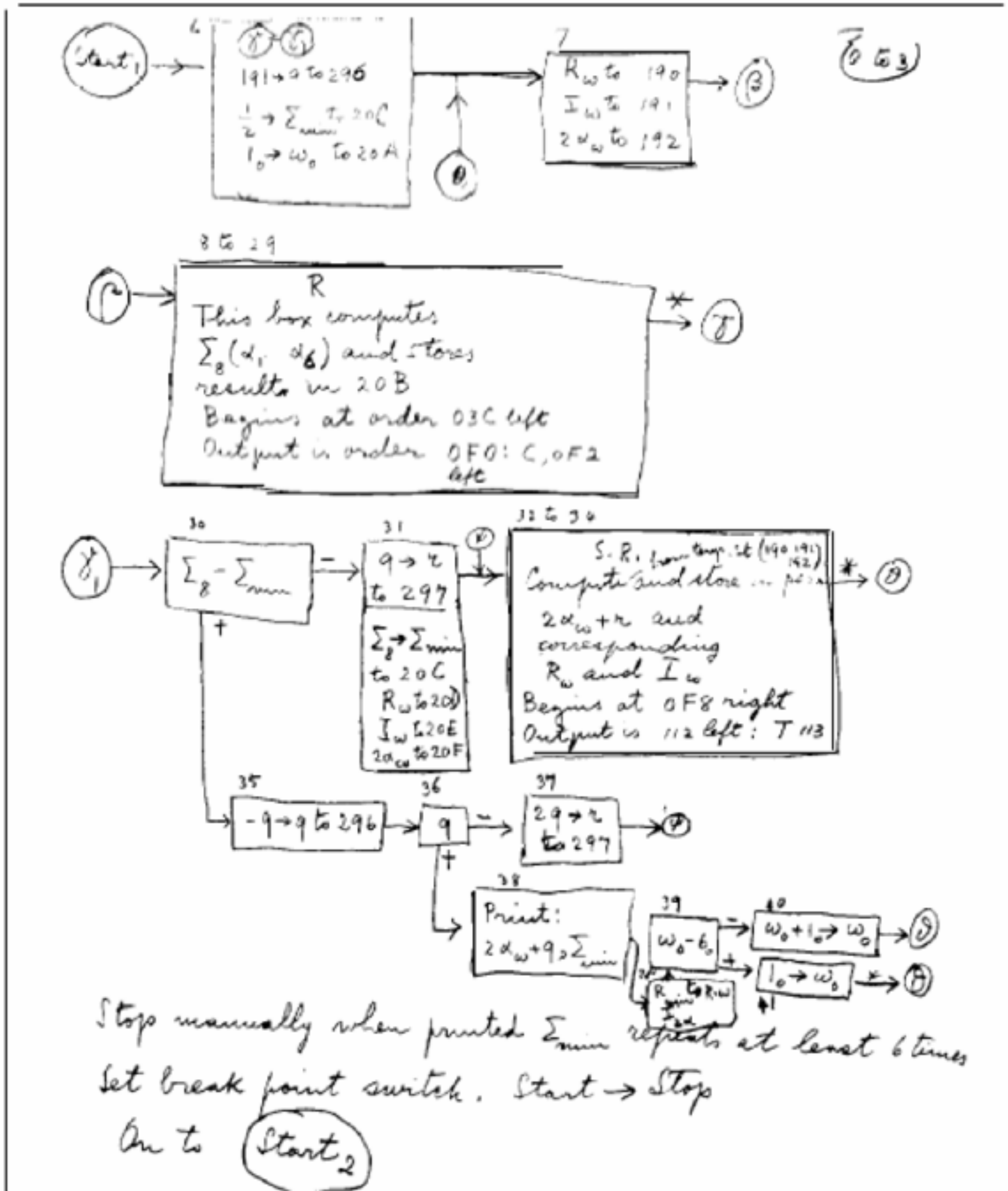
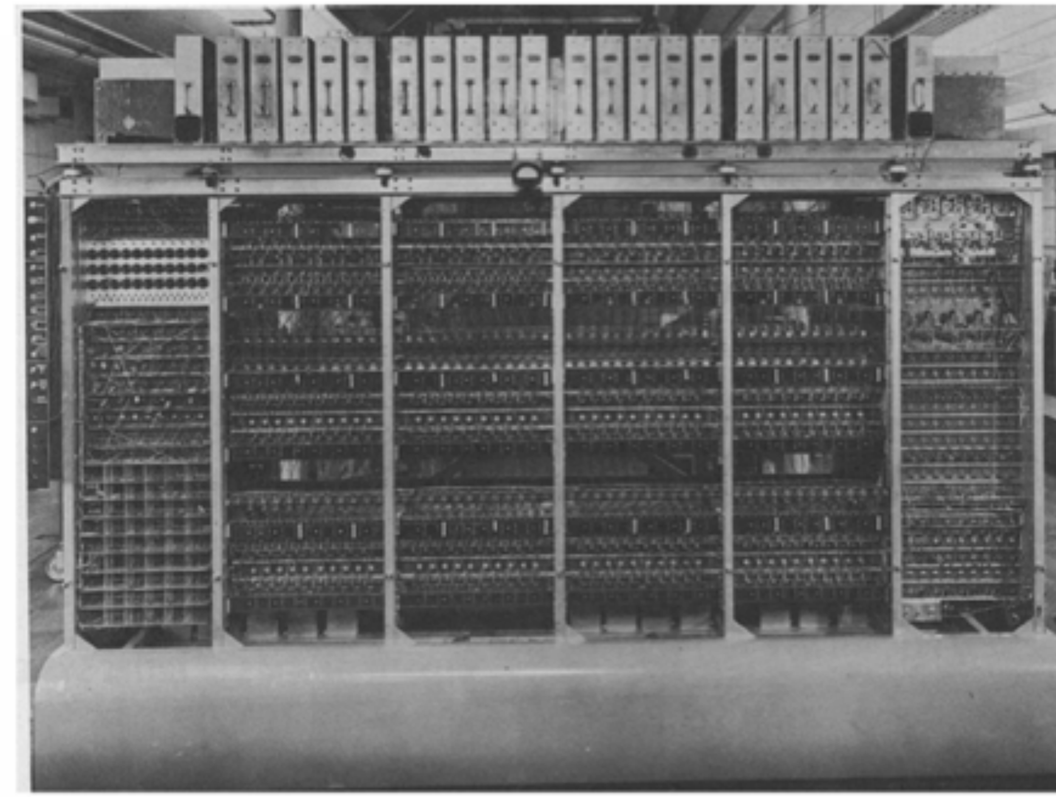



**Fig. 4. A subprogram written by Fermi for calculating phase shifts by finding a minimum chi-squared in a fit to the data.**



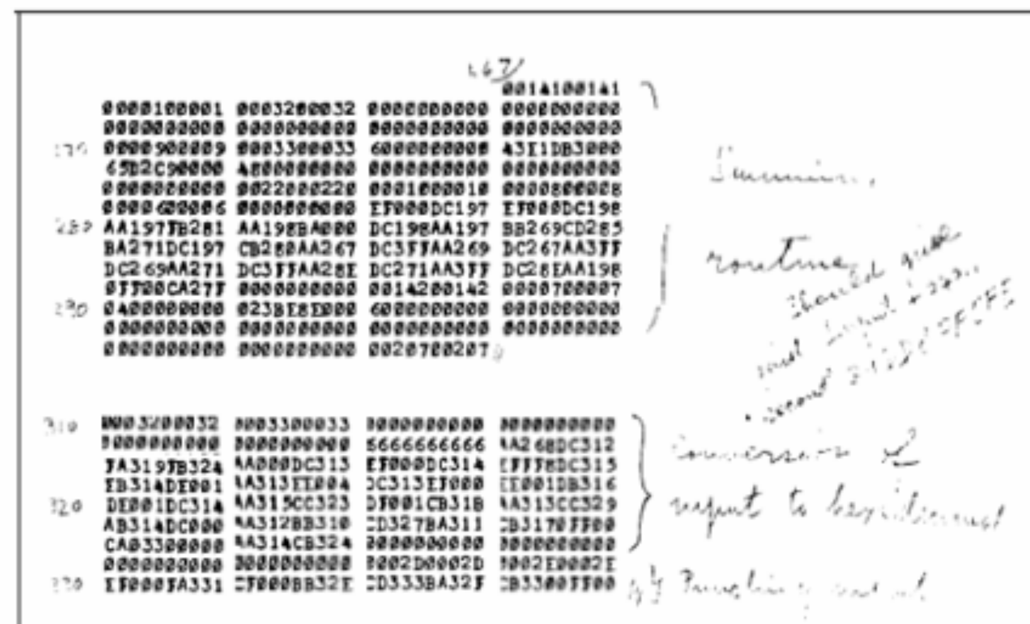



**Fig. 5. A portion of the printout of the program containing the subprograms described in Figs. 3 and 4. The program is written in machine language in hexadecimal numbers.**

LOS ALAMOS SCIENCE Fall 1986

# 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;

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

# 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

........

June 19th, 2015 - Geant4 10.1 patch 02 release ← Current version

We currently provide one public release every year

# News from 10.0 version



Version released on December 13rd, 2013

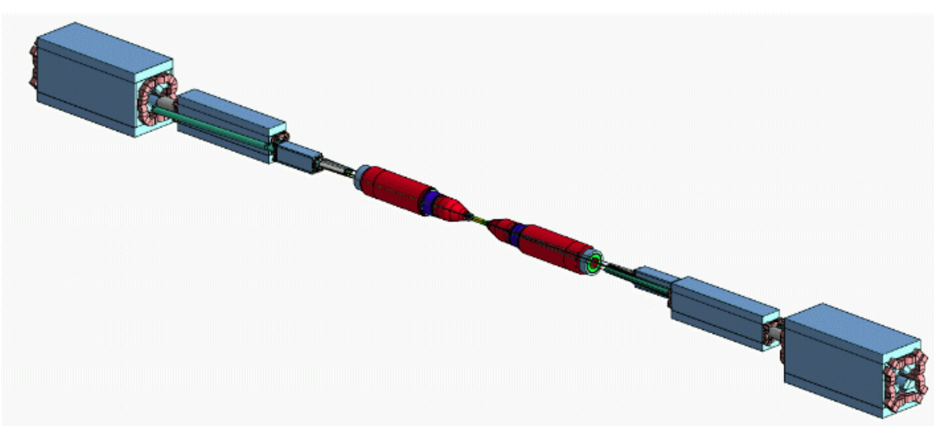
Supports for multi-thread (MT) approach that can be used in multi-cores machines

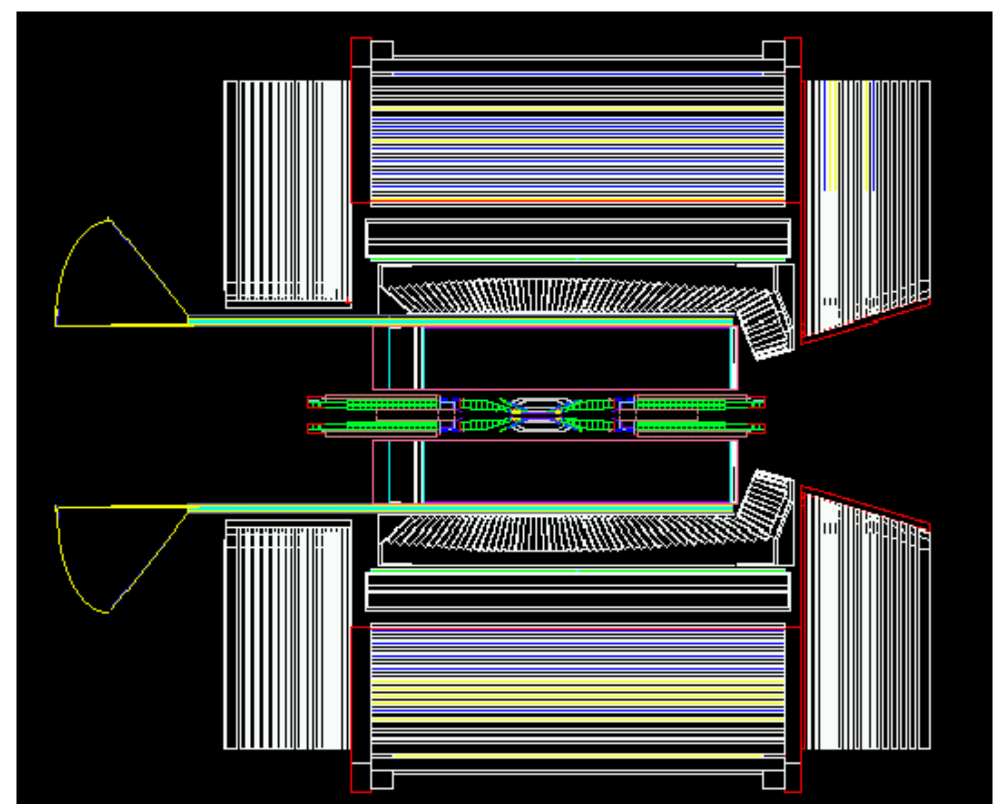
- Simulation is automatically split on a event-by-event basis
  - Different events are processes by different cores
  - Unique copy of Geometry and Physics
  - All the cores have them as read-only
- Backward compatible with the sequential mode
- MT programming requires some cares
  - Need to avoid conflicts between threads
  - Merge information at the end coming from the cores

# 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.



# Experiments and Monte Carlo



All the experiments have a (more or less) detailed Monte Carlo simulation

Design of the experiment

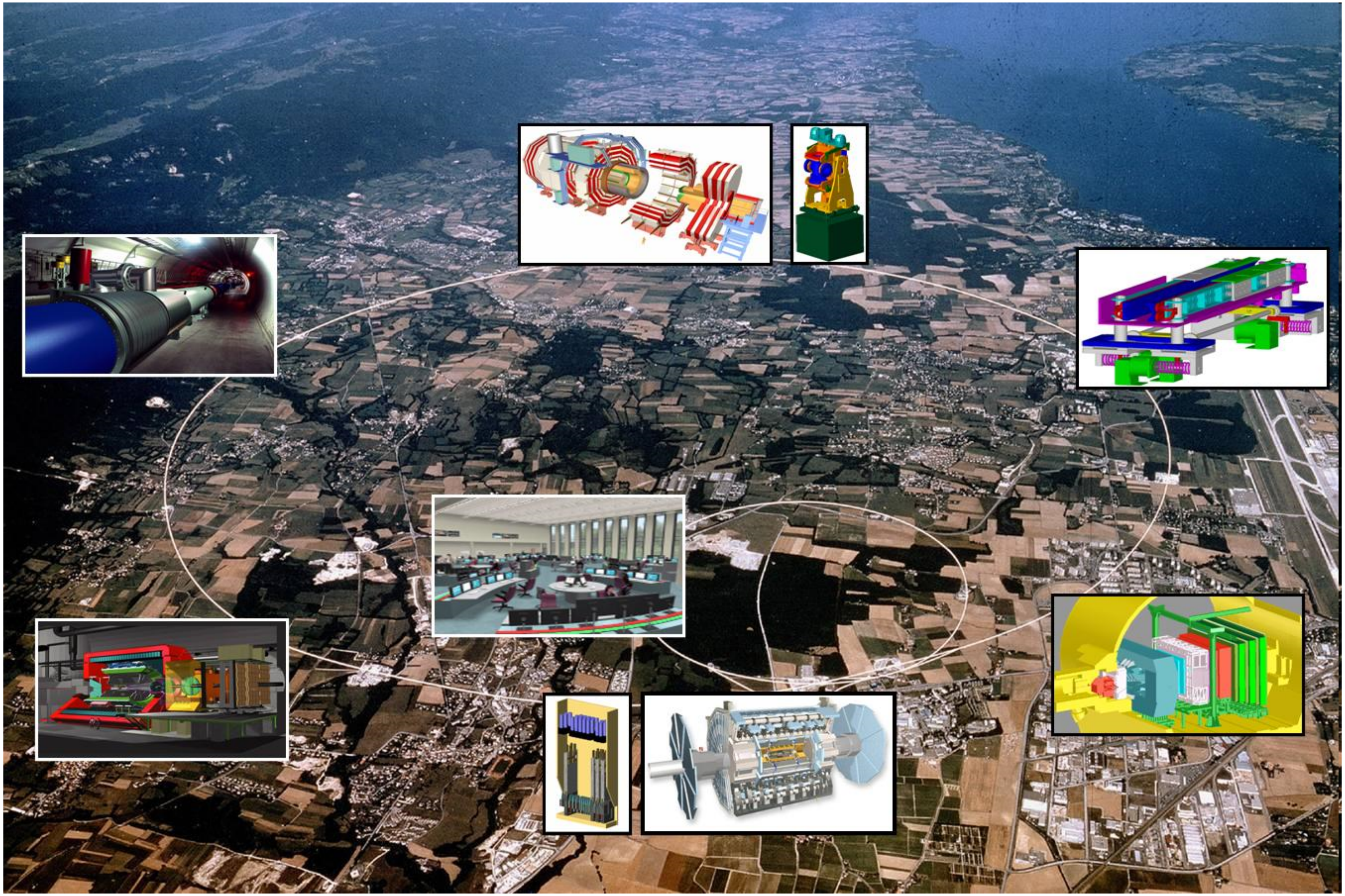
- Background evaluation
- Geometry and detector optimisation to maximise the scientific yield

Running/analysis phase

- Background evaluation, event triggers, efficiency evaluation
- Conversion of relative to absolute yields

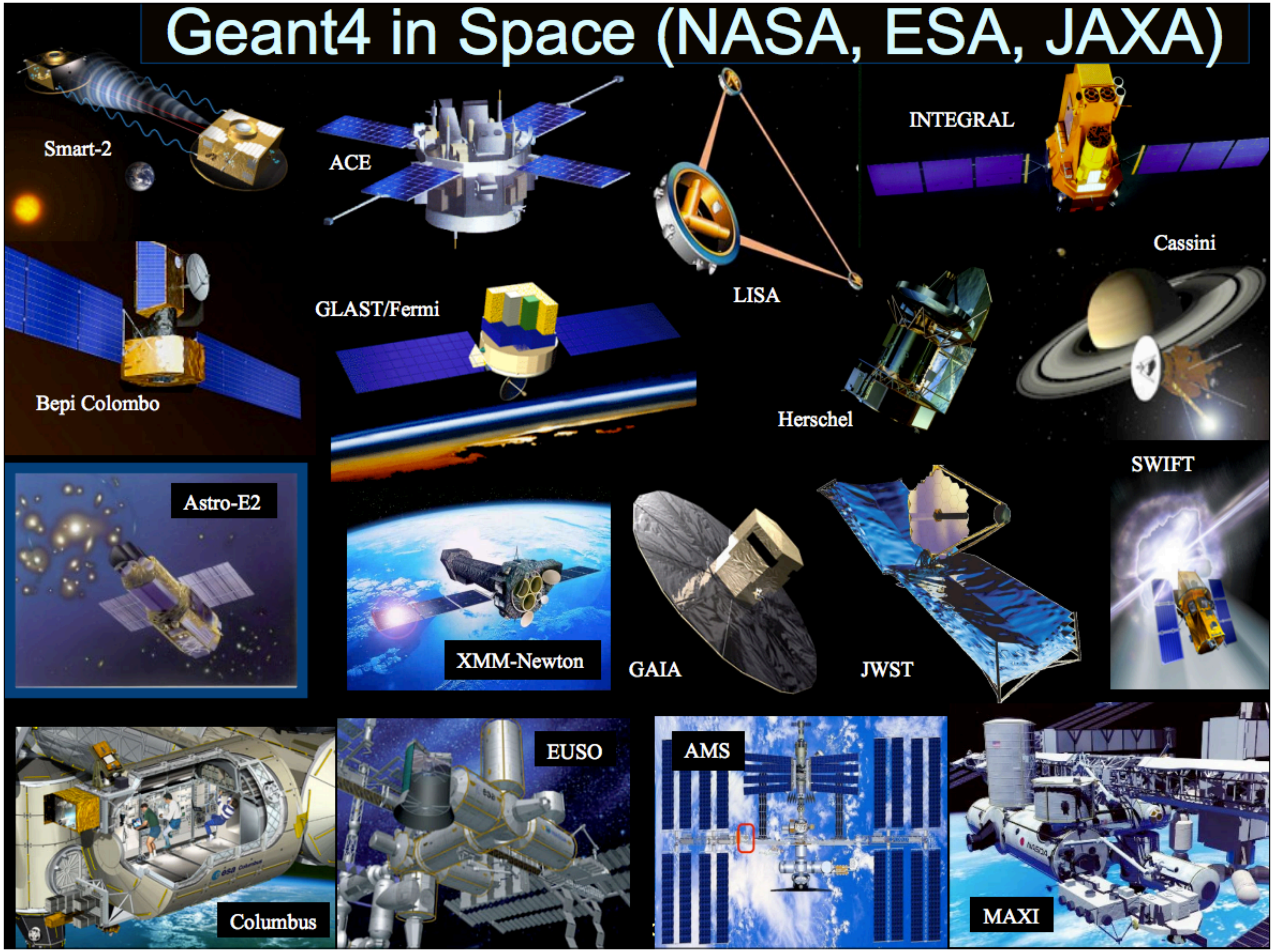
# Facts about Geant4





# Facts about Geant4



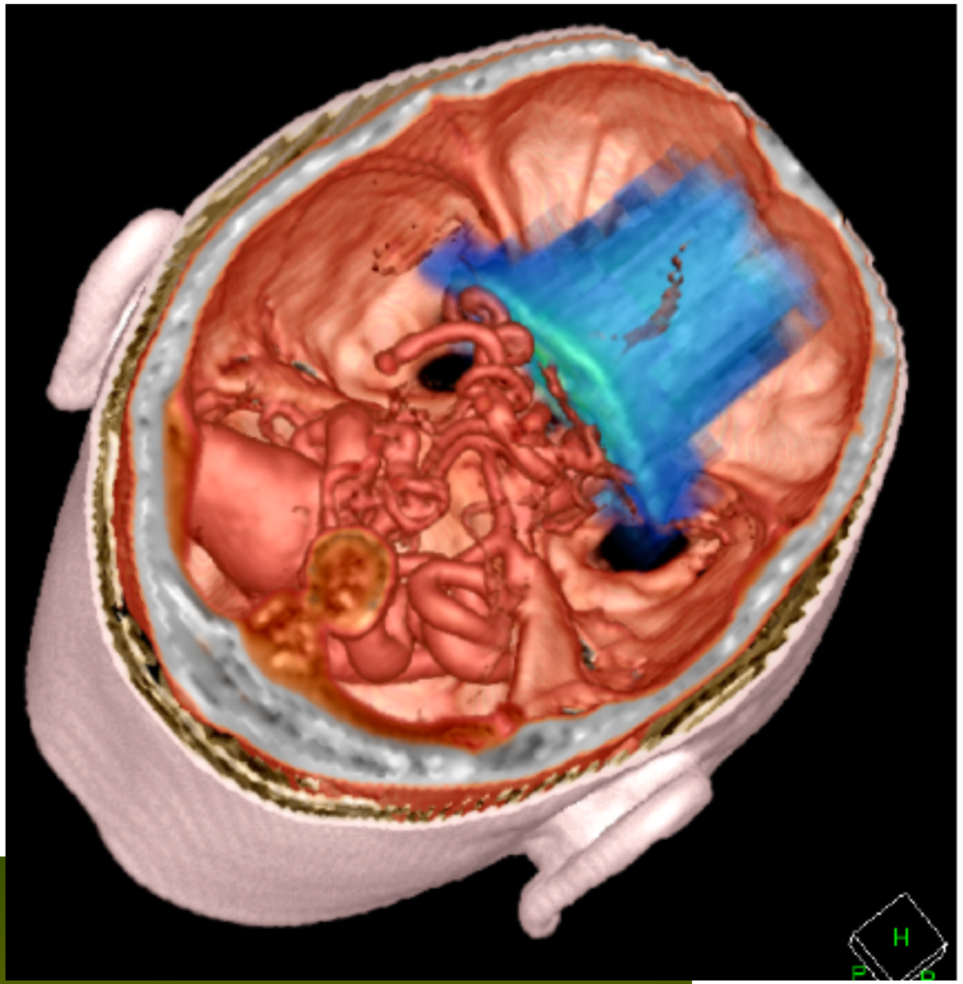


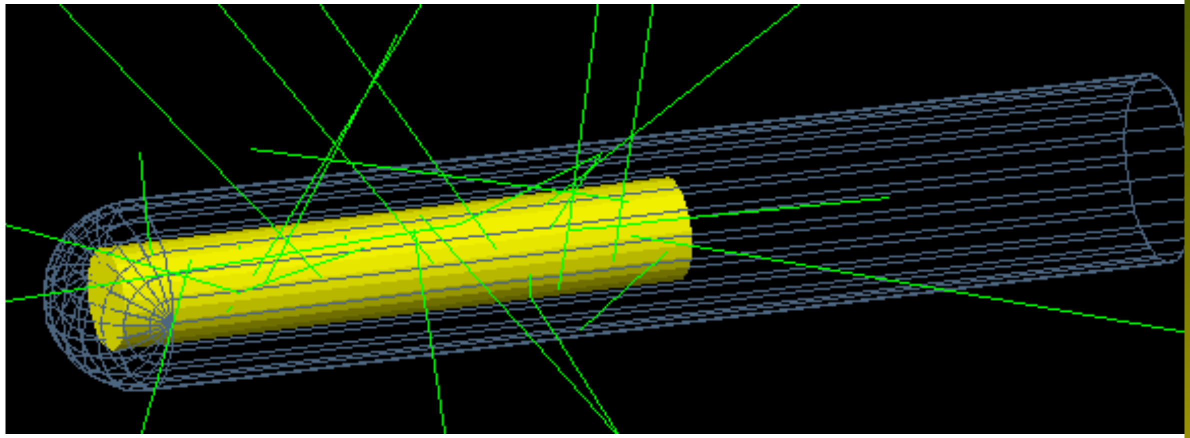
# Facts about Geant4

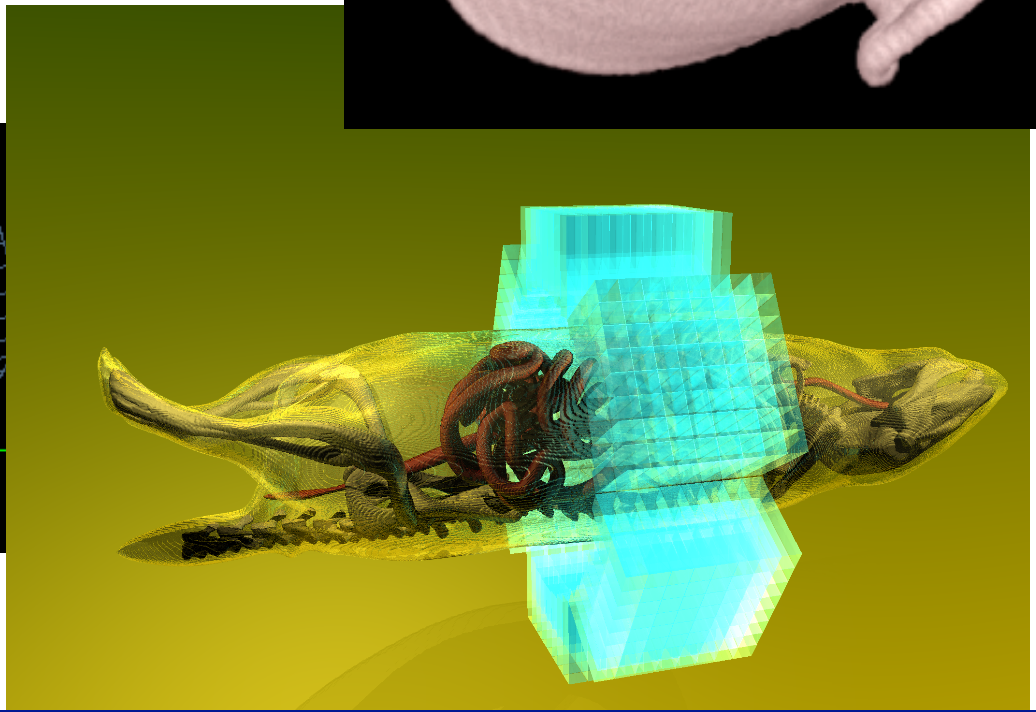


Major use cases

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







# 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:
  - Novice examples: overview of the Geant4 tools
  - Advanced Examples: Geant4 tools in real-life applications

# Minimum software requirements



- C++
  - A basic knowledge is required being Geant4 a collection of C++ libraries
  - It is complex but also no C++ experts can use Geant4
- Object oriented technology (OO)
  - Very basic knowledge
  - Expertise needed for the development of complex applications
- Unix/Linux
  - These are the standard OSs for Geant4 and a basic knowledge is required
  - Principal shell commands
  - How to compile a program
  - How to install from source code

# Why Geant4 is a common choice in the market



- Open Source and Object Oriented/C++
  - No black-box
  - Freely available on all the platforms
  - Can be easily extended and customised using the existing interfaces
    - New processes, new primary generations, interface with other softwares (Ex ROOT, ...)
- Complex geometries can be defined and handled
- Regular releases, validation, bug fixes
- High-quality physics customisable per use-case
- Start-to-end simulation for all particles including optical photons

# 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

# 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 couples of files are necessary (with the Main.cc ons)
  - The PrimaryGeneratorAction (.cc and .hh)
  - The DetectorConstruction (.cc and .hh)
  - The PhysicsList (.cc and .hh)

# Mandatory User's classes



Mandatory classes in ANY Geant4 User Application

- G4VUserDetectorConstruction
  describe the experimental set-up
- G4VUserPhysicsList
  select the physics you want to activate
- G4VUserPrimaryGeneratorAction
  generate primary events

# Mandatory User's classes



## 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

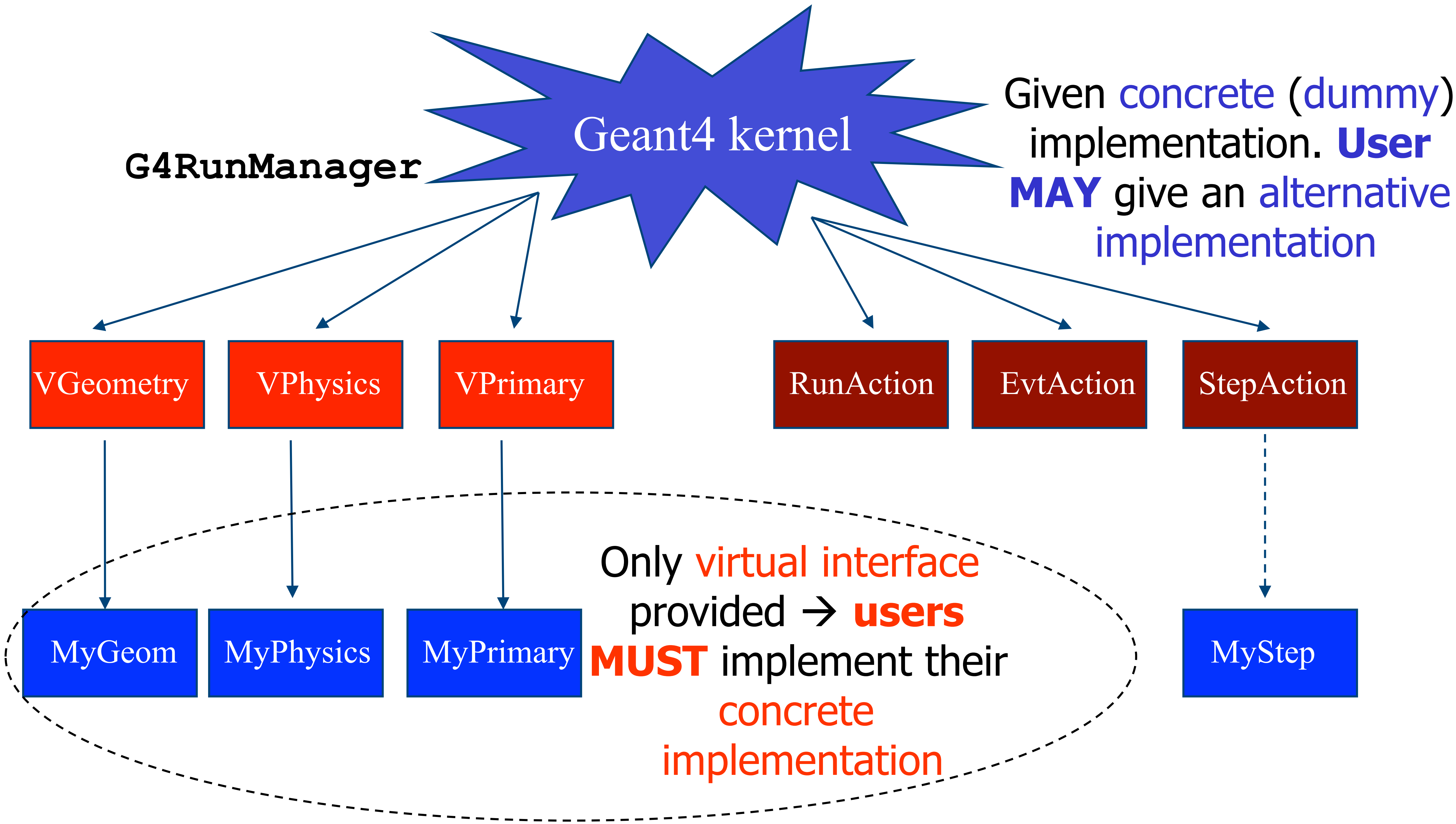
G4VUserPrimaryGeneratorAction
G4UserRunAction (*)
G4UserEventAction
G4UserTrackingAction
G4UserStackingAction
G4UserSteppingAction

Local: an instance of each action class exists **for each thread**.

(*) Two RunAction's allowed: one for master and one for threads

# Geant4 general scheme





# 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);
}
```

# MyActionInitialization() class file



## Register thread-local user actions

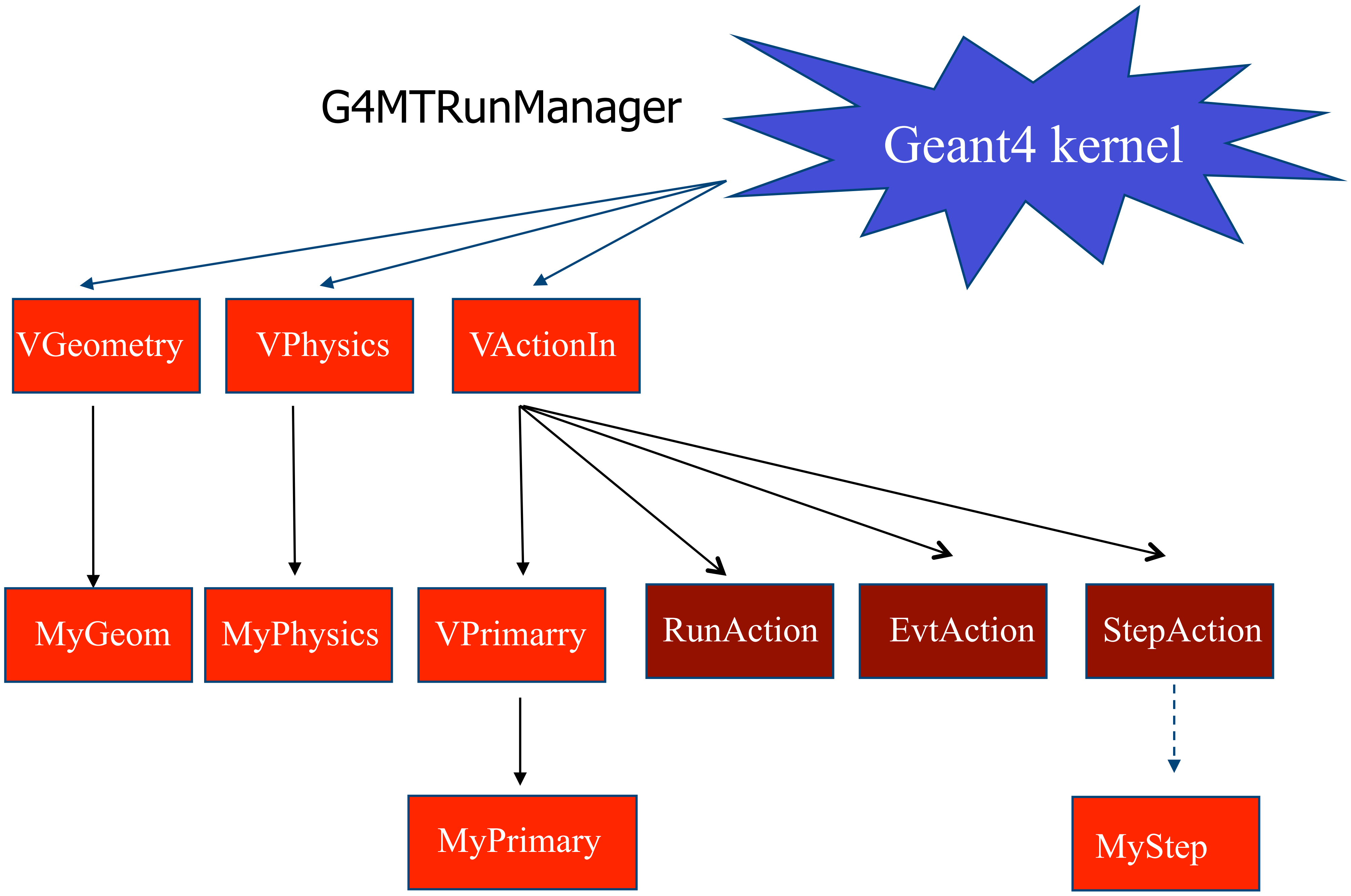
```
void MyActionInitialization::Build() const
{
  //Set mandatory classes
   SetUserAction(new MyPrimaryGeneratorAction());

  // Set optional user action classes
 SetUserAction(new MyEventAction());
 SetUserAction(new MyRunAction());
}
```

## Register RunAction for the Master

```
void MyActionInitialization::BuildForMaster() const
{
  // Set optional user action classes
 SetUserAction(new MyMasterRunAction());
}
```

# Geant4 Concepts with the MultiThreads





# 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 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
- Only a waste of CPU time for LHC, critical for many low-background experiments

# 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()`**

# 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

# Optional Visualisation



- In your main(), taking into account your computer environment, instantiate a `G4VisExecutive` and invoke its `Initialize()` method

- Geant4 provides 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

# 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…

# Installation tips

# Geant4 Software Download

## Geant4 9.5

**released 2 December 2011**

The Geant4 source code is freely available. See the licence conditions.

Please read the **Release Notes** before downloading or using this release.

## Source files

Please choose the archive best suited to your system and archiving tool:

Download GNU or Linux tar format, compressed using gzip ( 27Mbytes, 28458437 bytes ).
*After downloading, gunzip, then unpack using GNU tar.*

Download ZIP format ( 39Mbytes, 40826089 bytes ).
*After downloading, unpack using e.g. WinZip.*

## Data files (*)

For specific, optional physics processes some of the following files are required. The file format is compatible with Unix, GNU, and Windows utilities.

Download Neutron data files with thermal cross sections - version 4.0 ( 381Mbytes, 400001140 bytes ) NEW

Download Neutron data files without thermal cross sections - version 0.2 ( 12Mbytes, 12465281 bytes )

Download Data files for low energy electromagnetic processes - version 6.23 ( 15Mbytes, 15960390 bytes ) NEW

Download Data files for photon evaporation - version 2.2 ( 7.3Mbytes, 7704178 bytes ) NEW

Download Data files for radioactive decay hadronic processes - version 3.4 ( 716Kbytes, 732861 bytes ) NEW

Download Data files for nuclear shell effects in INCL/ABLA hadronic model - version 3.0 ( 54Kbytes, 54909 bytes )

Download Data files for evaluated neutron cross sections on natural composition of elements - version 1.1 ( 1.2Mbytes, 1247160 bytes ) NEW

Download Data files for shell ionisation cross sections - version 1.3 ( 4.1Mbytes, 4293607 bytes ) NEW

Download Data files for measured optical surface reflectance - version 1.0 ( 1.2Mbytes, 1257863 bytes )

## Pre-compiled Libraries

These are compiled with Geant4 default settings and optimization turned on. Please choose according to your system/compiler:

Download compiled using gcc 4.1.2 on Scientific Linux CERN 5 (SLC5, based on Redhat Linux Enterprise 5), 64 bits - ( 32Mbytes, 33212295 bytes )

Download compiled using gcc 4.2.1 on Mac (MacOSX 10.7), 64 bits - ( 31Mbytes, 32039379 bytes )

**Related Links**

- Geant4-MT **prototype**.
- **Previous Releases** of Geant4 (since release 8.3).
- LXR source code browser.
- Installation Guide tutorials for Linux, Mac and Windows.
- **Windows CygWin installation note**.

# Installation tips



You can download the compiled libraries of Geant4 but the compilation in your computer is strongly suggested

Download the source file from the Geant4 web site

Two way to proceed:

- Using cmake via terminal
- Using the GUI version of cmake

# Installation tips



- **cmake** version greater than 2.8.3
- Locate the **source folder**
  Ex: `/home/Username/geant4_10_01_p01`
- Create the **build folder**
  Ex: `/home/Username/geant4_10_01_p01-build`
- Create the **install folder**
  Ex: `/home/Username/geant4_10_01_p01-install`
- `cmake -DCMAKE_INSTALL_PREFIX=/home/Username/geant4_10_01_p01-install/`
- Define and/or activate the **additional features/package you require** using the same cmake interface
- make -jN
- make install

# Installation tips



```
[  4%] Building CXX object source/global/CMakeFiles/G4global.dir/management/src/G4PhysicsVectorCache.cc.o
[  4%] Building CXX object source/global/CMakeFiles/G4global.dir/management/src/G4Physics2DVector.cc.o
[  4%] Building CXX object source/global/CMakeFiles/G4global.dir/management/src/G4Physics2DVectorCache.cc.o
[  4%] Building CXX object source/global/CMakeFiles/G4global.dir/management/src/G4Pow.cc.o
[  4%] Building CXX object source/global/CMakeFiles/G4global.dir/management/src/G4SliceTimer.cc.o
[  4%] Building CXX object source/global/CMakeFiles/G4global.dir/management/src/G4StateManager.cc.o
[  4%] Building CXX object source/global/CMakeFiles/G4global.dir/management/src/G4Timer.cc.o
[  4%] Building CXX object source/global/CMakeFiles/G4global.dir/management/src/G4UnitsTable.cc.o
[  4%] Building CXX object source/global/CMakeFiles/G4global.dir/management/src/G4VExceptionHandler.cc.o
[  4%] Building CXX object source/global/CMakeFiles/G4global.dir/management/src/G4VNotifier.cc.o
[  4%] Building CXX object source/global/CMakeFiles/G4global.dir/management/src/G4VStateDependent.cc.o
[  4%] Building CXX object source/global/CMakeFiles/G4global.dir/management/src/G4coutDestination.cc.o
[  4%] Building CXX object source/global/CMakeFiles/G4global.dir/management/src/G4ios.cc.o
Linking CXX shared library ../../outputs/library/Darwin-g++/libG4global.dylib
[  4%] Built target G4global
Scanning dependencies of target G4analysis
Scanning dependencies of target G4intercoms
[  4%] Building CXX object source/intercoms/CMakeFiles/G4intercoms.dir/src/G4UIaliasList.cc.o
[  4%] Building CXX object source/analysis/CMakeFiles/G4analysis.dir/src/G4AnalysisVerbose.cc.o
[  4%] Building CXX object source/intercoms/CMakeFiles/G4intercoms.dir/src/G4UIbatch.cc.o
[  4%] Building CXX object source/analysis/CMakeFiles/G4analysis.dir/src/G4CsvAnalysisManager.cc.o
[  4%] Building CXX object source/intercoms/CMakeFiles/G4intercoms.dir/src/G4UIcmdWith3Vector.cc.o
[  4%] Building CXX object source/intercoms/CMakeFiles/G4intercoms.dir/src/G4UIcmdWith3VectorAndUnit.cc.o
[  4%] Building CXX object source/analysis/CMakeFiles/G4analysis.dir/src/G4VAnalysisManager.cc.o
[  5%] Building CXX object source/analysis/CMakeFiles/G4analysis.dir/src/G4RootAnalysisManager.cc.o
[  5%] Building CXX object source/intercoms/CMakeFiles/G4intercoms.dir/src/G4UIcmdWithABool.cc.o
[  5%] Building CXX object source/intercoms/CMakeFiles/G4intercoms.dir/src/G4UIcmdWithADouble.cc.o
[  5%] Building CXX object source/intercoms/CMakeFiles/G4intercoms.dir/src/G4UIcmdWithADoubleAndUnit.cc.o
[  5%] Building CXX object source/intercoms/CMakeFiles/G4intercoms.dir/src/G4UIcmdWithAString.cc.o
[  5%] Building CXX object source/intercoms/CMakeFiles/G4intercoms.dir/src/G4UIcmdWithAnInteger.cc.o
[  5%] Building CXX object source/intercoms/CMakeFiles/G4intercoms.dir/src/G4UIcmdWithoutParameter.cc.o
[  5%] Building CXX object source/intercoms/CMakeFiles/G4intercoms.dir/src/G4UIcommand.cc.o
```

# Installation tips



If GEANT4_INSTALL_DATA is ON
the additional external data libraries are automatically downloaded

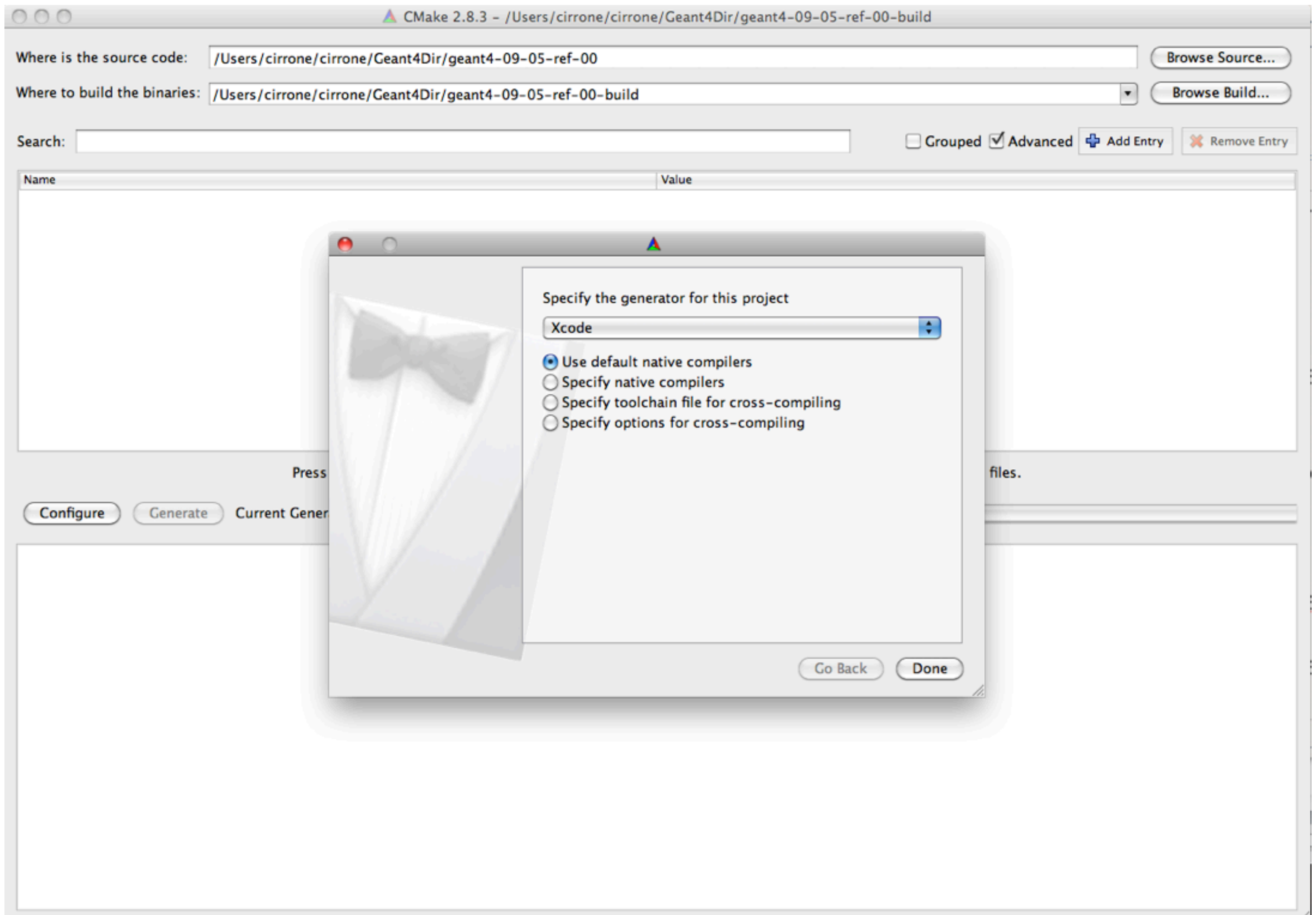
If GEANT4_INSTALL_EXAMPLES is ON
Examples are installed

If GEANT4_USE_SYSTEM_CLHEP is ON
external CLHEP are searched

See documentation for details for the complete variables list and explanation

# Installation tips






A friendly way to do the same things (on Windows and Mac)

# The end

# Main Geant4 capabilities



Transportation of a particle ‘step-by-step’ taking into account all the possible interactions with materials and fields

The transport ends if the particle

- reaches a zero kinetic energy
- disappears in some interaction
- reaches the end of the simulation volume

# An example of Multi Threads main()



```
{
 …
 // Construct the default run manager
 G4MTRunManager* runManager = new G4MTRunManager;

 // 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);
 …
}
```