The Monte Carlo approach and the Geant4 toolkit

IV International Geant4 School
Vinca Institute , Belgrade (SRB)
October 23rd - 28th, 2016

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 this school

Installation
Configuration
Generation of particles
Geometry and materials
Physics
Information retrieving

Finding the material

- Pablo Cirrone, Giuliana Milluzzo, Luciano Pandola, Giada Petringa, Jan Pipek

  INFN-Laboratori Nazionali del Sud - Catania, (I)

- 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/

- https://www.facebook.com/SoftwareandGeant4School/

School organisation

Regular schools officially organised by the Geant4 collaboration

Two editions per years: summer and autumnal school

COMETA

INFN Istituto Nazionale di Fisica Nucleare

uniss UNIVERSITÀ DEGLI STUDI DI SASSARI

ALGHERO
Hotel Porto Conte
04 JUNE 2017 > 10 JUNE 2017

14th SEMINAR ON SOFTWARE FOR NUCLEAR, SUBNUCLEAR AND APPLIED PHYSICS

The Seminar offers lectures to PhD students, Postdoctoral scholars and young Researchers working at Universities or Research Institutes.

The Seminar is organized in didactic units on software developed and used in fundamental and applied physics, theoretical and experimental.

The lectures includes a full official basic course on the Geant4 Monte Carlo simulation toolkit, including theoretical and practical sessions.

In this edition a basic course on GPU programming will be also offered.

In addition, for interested people, a test examination will be performed at the end of the school and a written certificate with grade will be issued.

A limited number of grants are available to cover fee and accommodation expenses.

INFORMATIONS
http://agenda.infn.it/event/AlgheroSeminar2017

Scientific Committee

Tommaso Boccali
Massimo Carpinelli
G.A.Pablo Cirrone
Giacomo Cuttone
Giuliana Milluzzo
Francesca Nicotra
Piernicola Oliva
Luciano Pandola
Giada Petringa
Jan Pipek
Francesco Romano
Valeria Sipala
Arnaldo Stefanini

Generals

- Bus shuttle to reach Vinca and to came back to to the hotel (8:30 am — 6:30 pm)
- Wifi connection
- Coffee breaks
  - 10:30 - 11:00 and 16:00 - 16:30
- Lunches
  - 12:30 - 14:00
- Theoretical lessons followed by exercises
  - Theory + hands-on sessions for each Geant4 module
- SSH connection (if needed)
- Tour and social dinner

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**

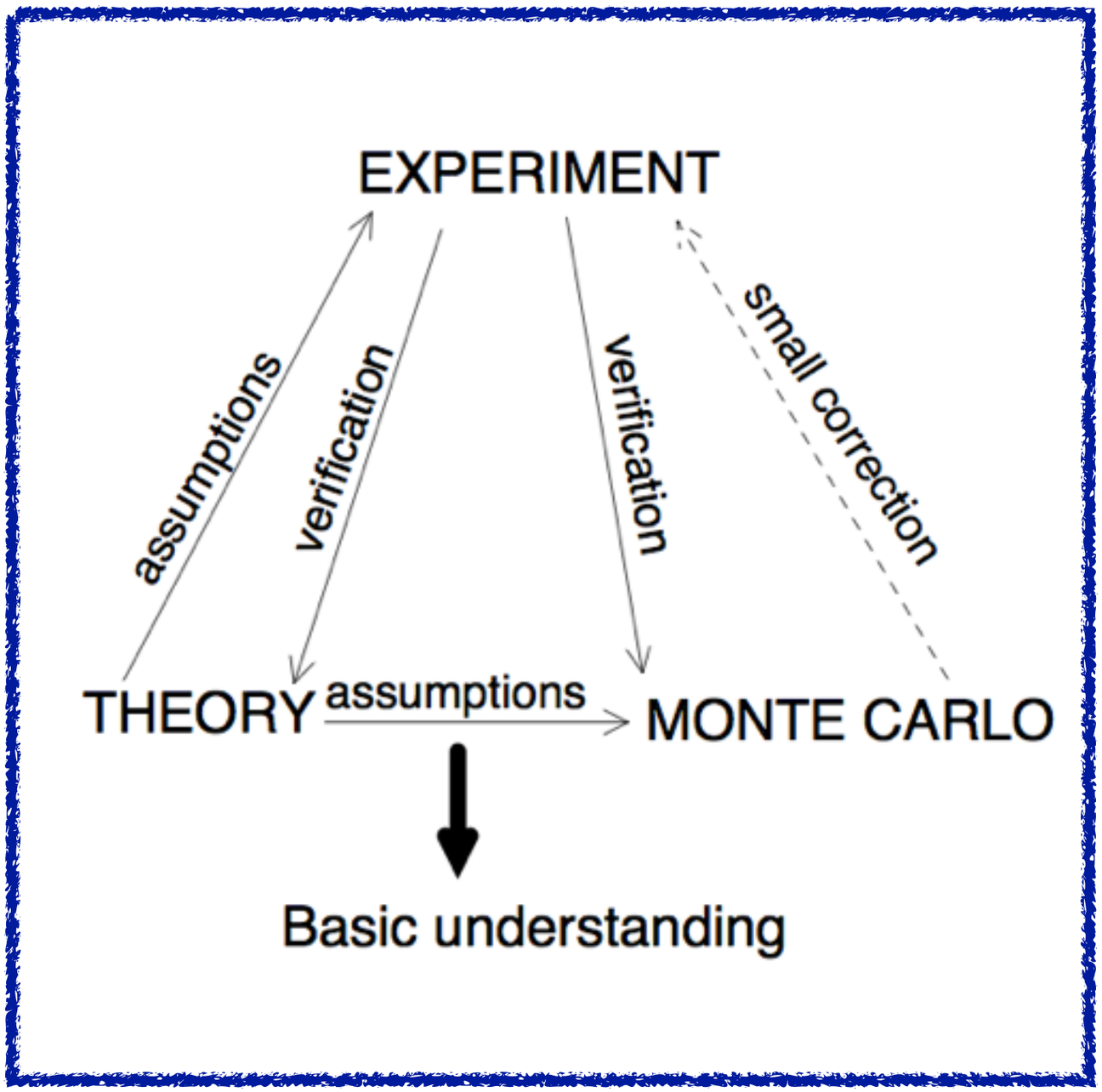
"If we are interested in a parameter of, i.e., an equation:

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

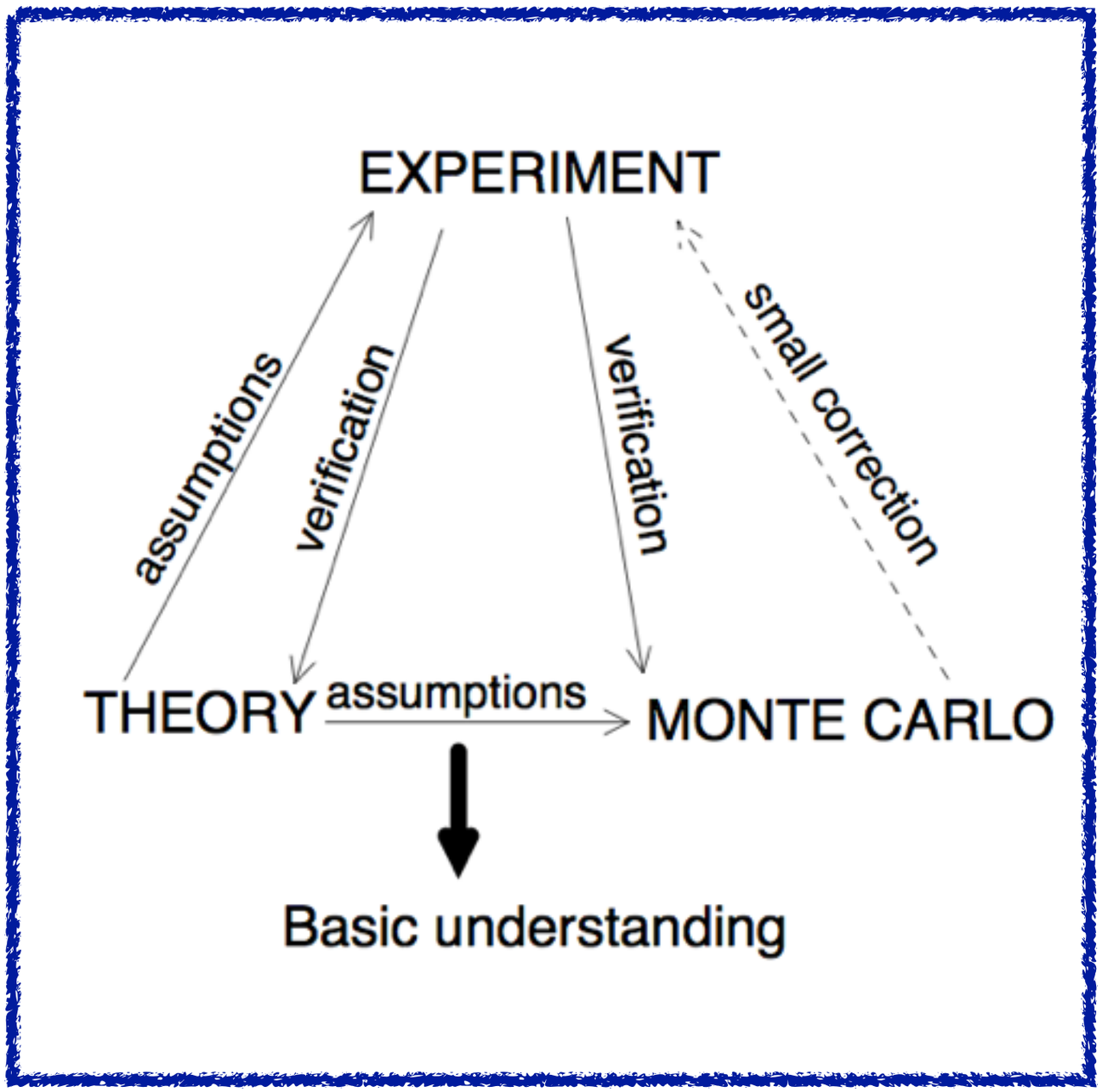


Monte Carlo helps

To verify a theory if physics models are in development

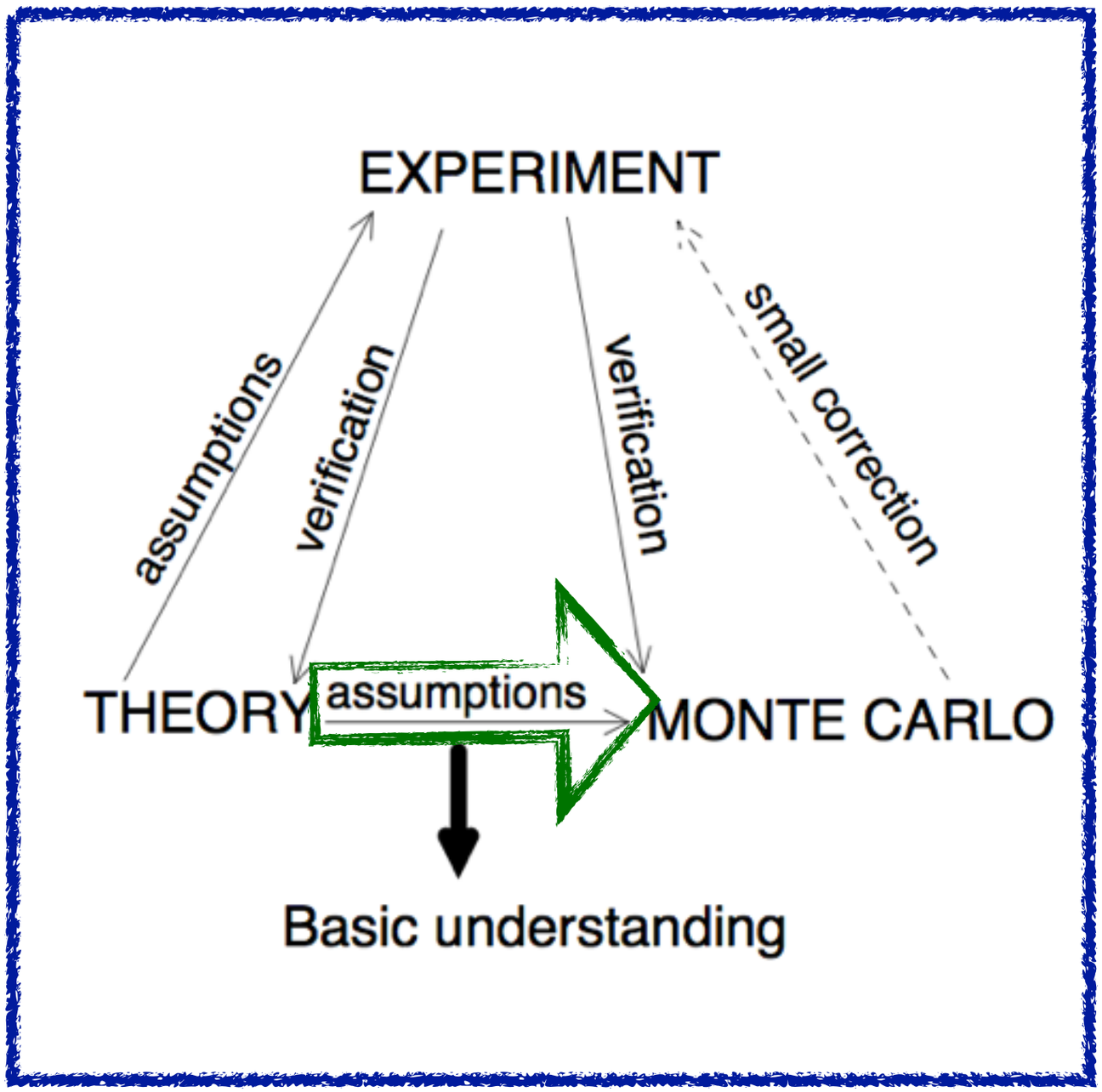
To develop or verify an experiment in the other case

Brief history and principles of Monte Carlo



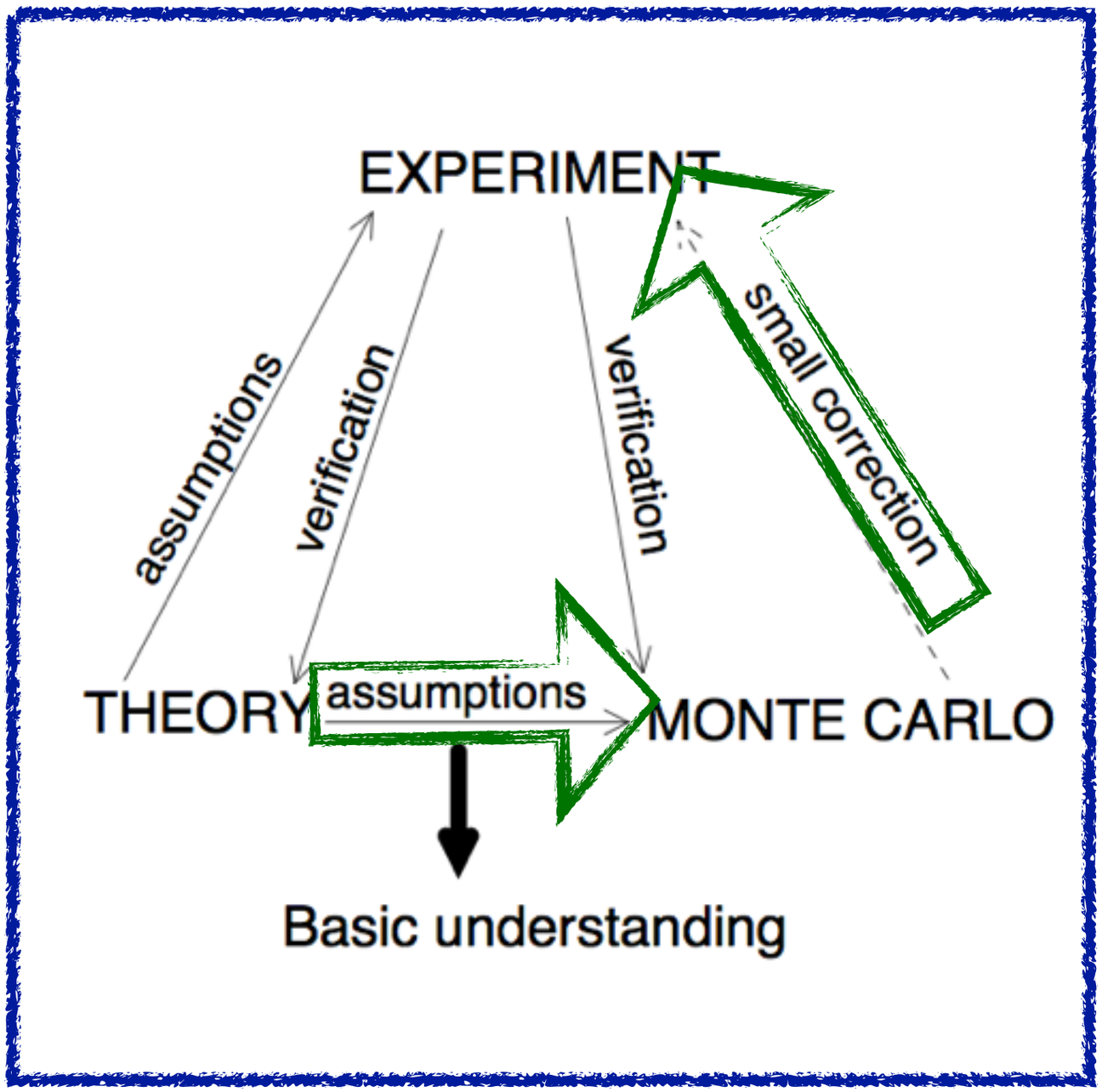
In particle transport, if particles interaction models are known, MC can be used to calculate the parameters of the motion equations in a given configuration

Brief history and principles of Monte Carlo



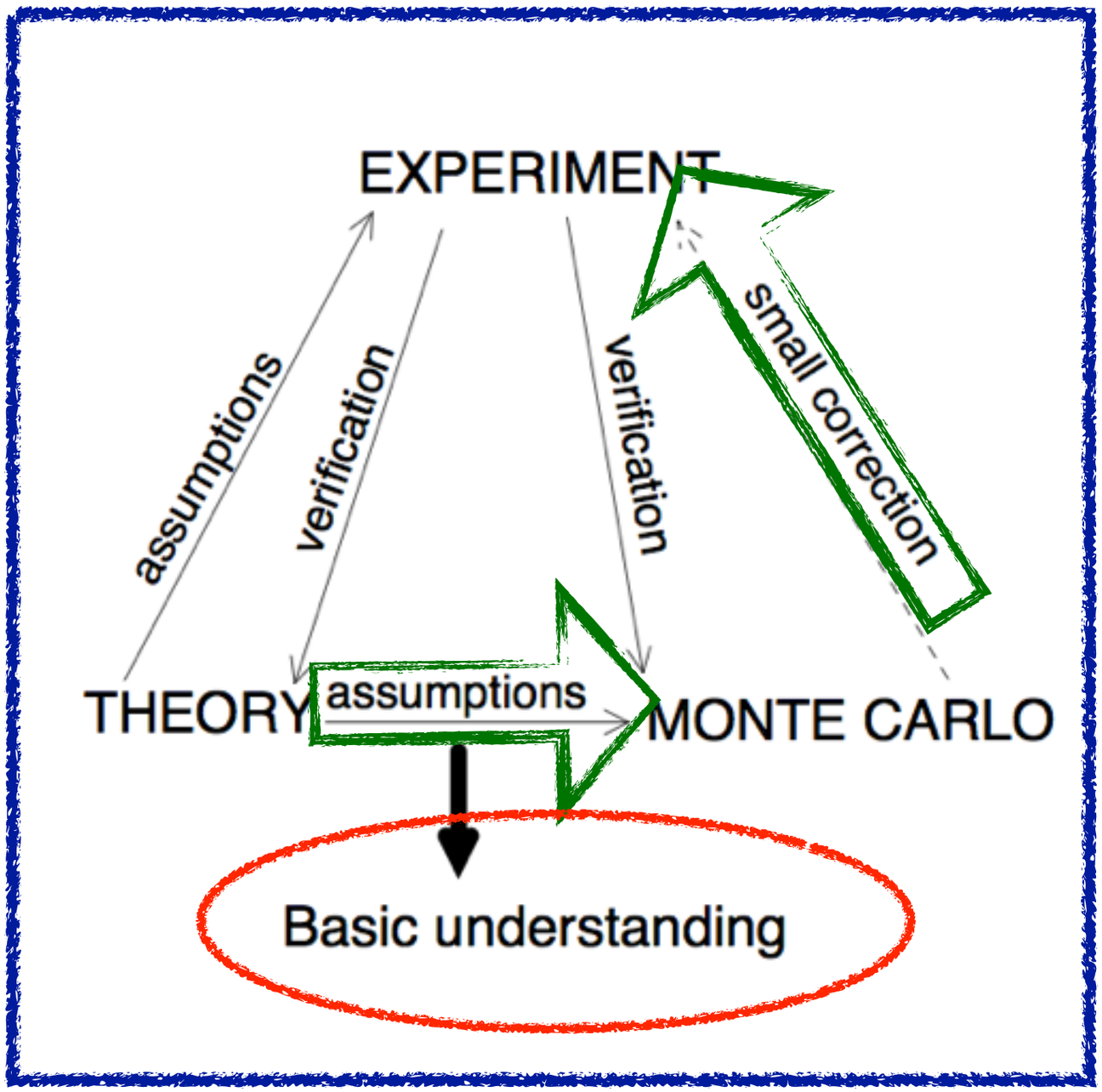
In particle transport, if particles interaction models are known, MC can be used to calculate the parameters of the motion equations in a given configuration

Brief history and principles of Monte Carlo



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

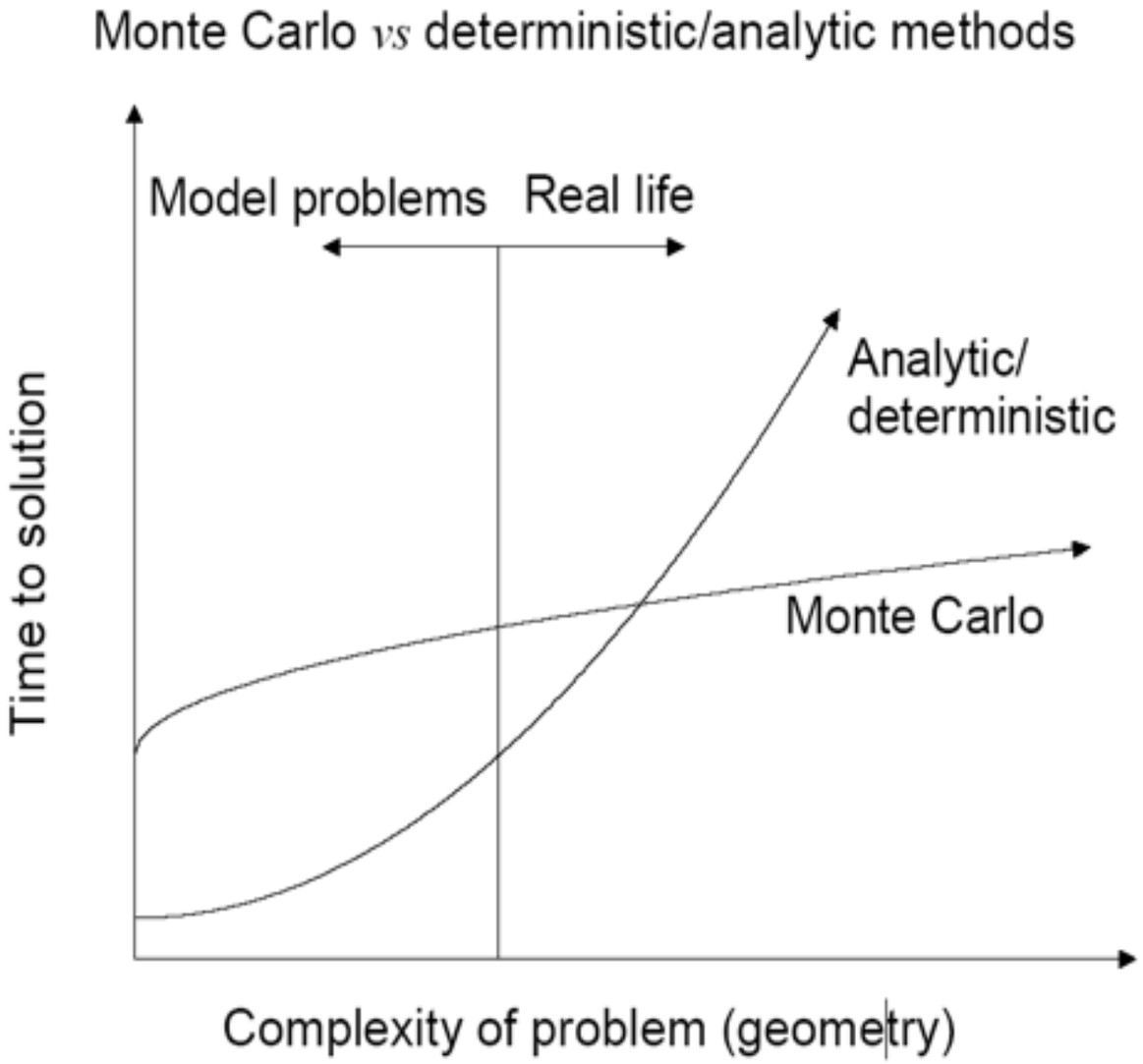


In particle transport, if particles interaction models are known, MC can be used to calculate the parameters of the motion equations in a given configuration

When use the Monte Carlo approach

- Random quantities (e.g. the average value of mm of rain)
- Deterministic problems (definite integral?)

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

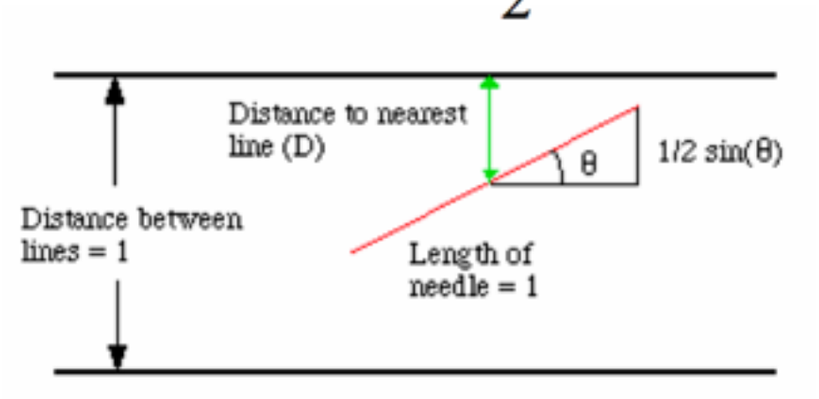
Is a computer necessary for a Monte Carlo Calculation ?

PI derivation using Monte Carlo and shooting a needle

Two variables: θ and D

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

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



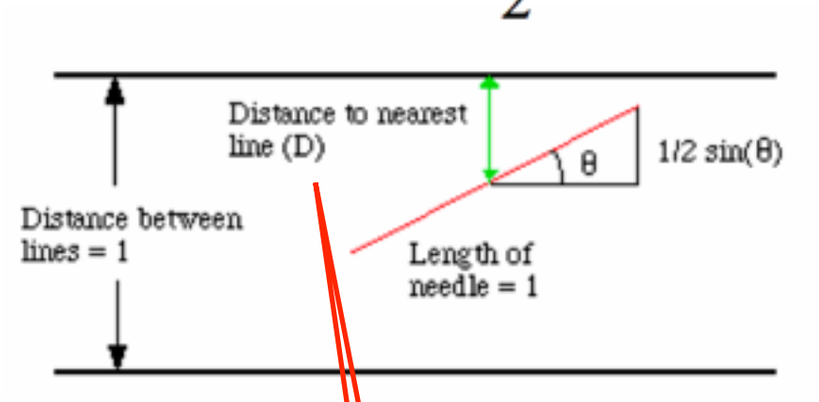
Georges Louis Leclerc
Comte de Buffon
(07.09.1707.-16.04.1788.)

PI derivation using Monte Carlo and shooting a needle

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:

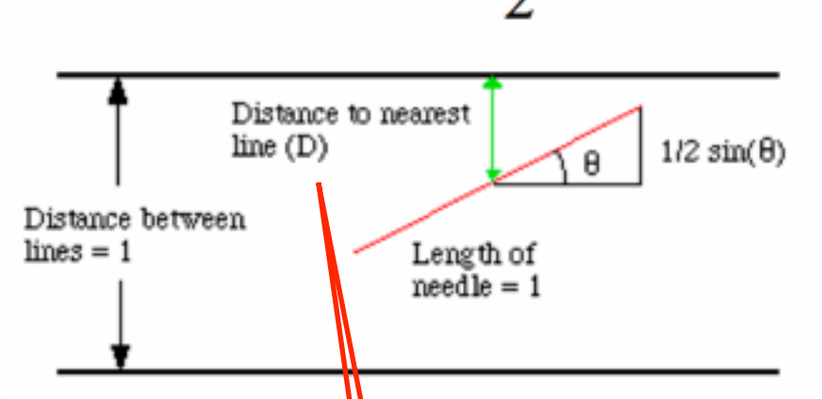
$$D \le \frac{1}{2}\sin(\vartheta)$$

PI derivation using Monte Carlo and shooting a needle

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:

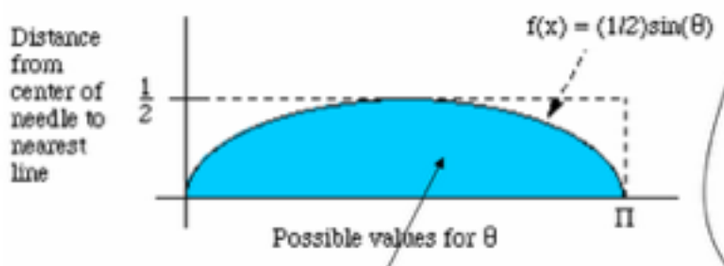
$$D \le \frac{1}{2}\sin(\vartheta)$$

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

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

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

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



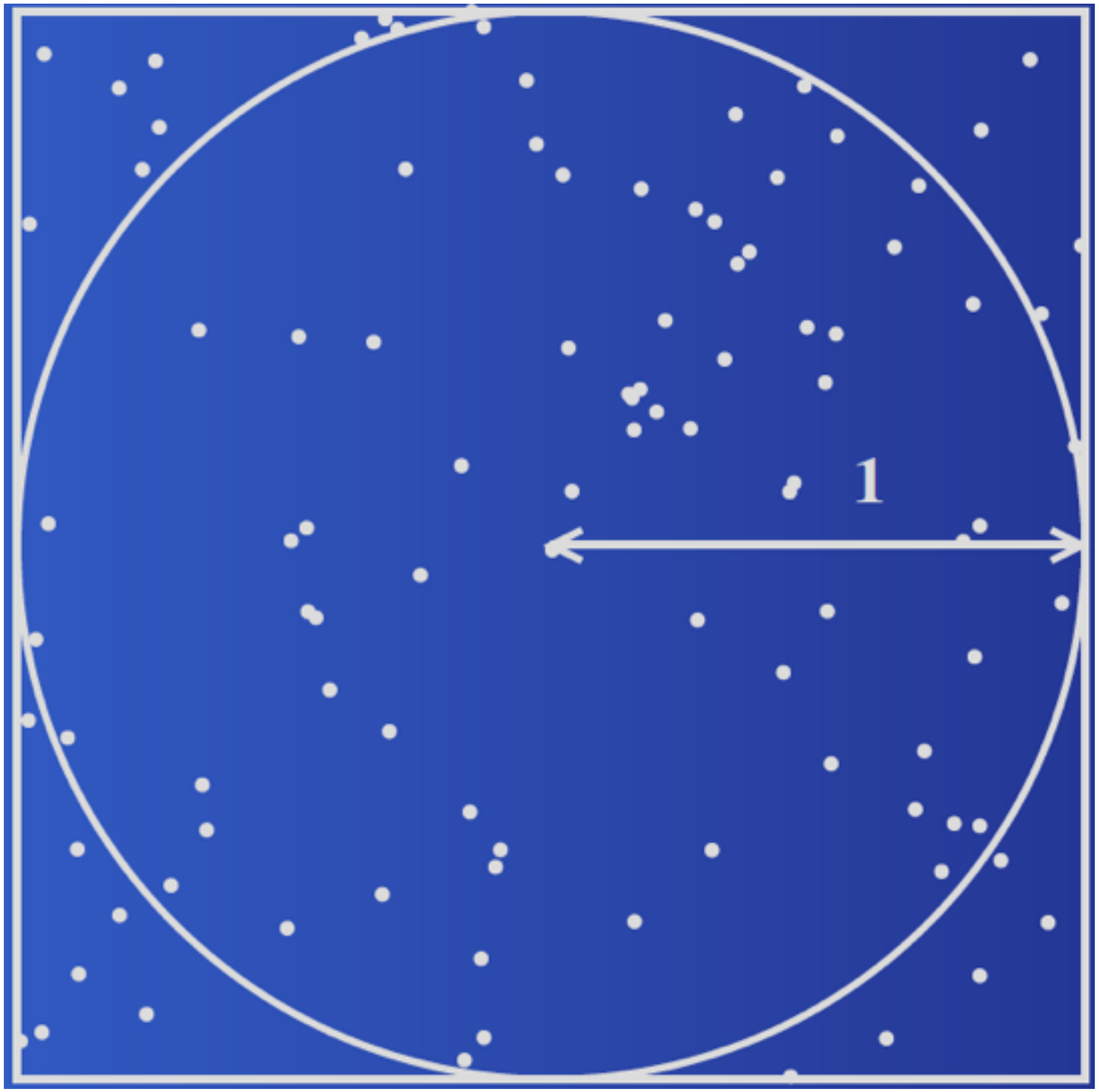
N_0 times the needle was shot

N times the needle hit the line

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

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

PI derivation using Monte Carlo and shooting a small stone



Area of square: As = 4

Area of circle: Ac = π

Fraction p of random points inside the circle:

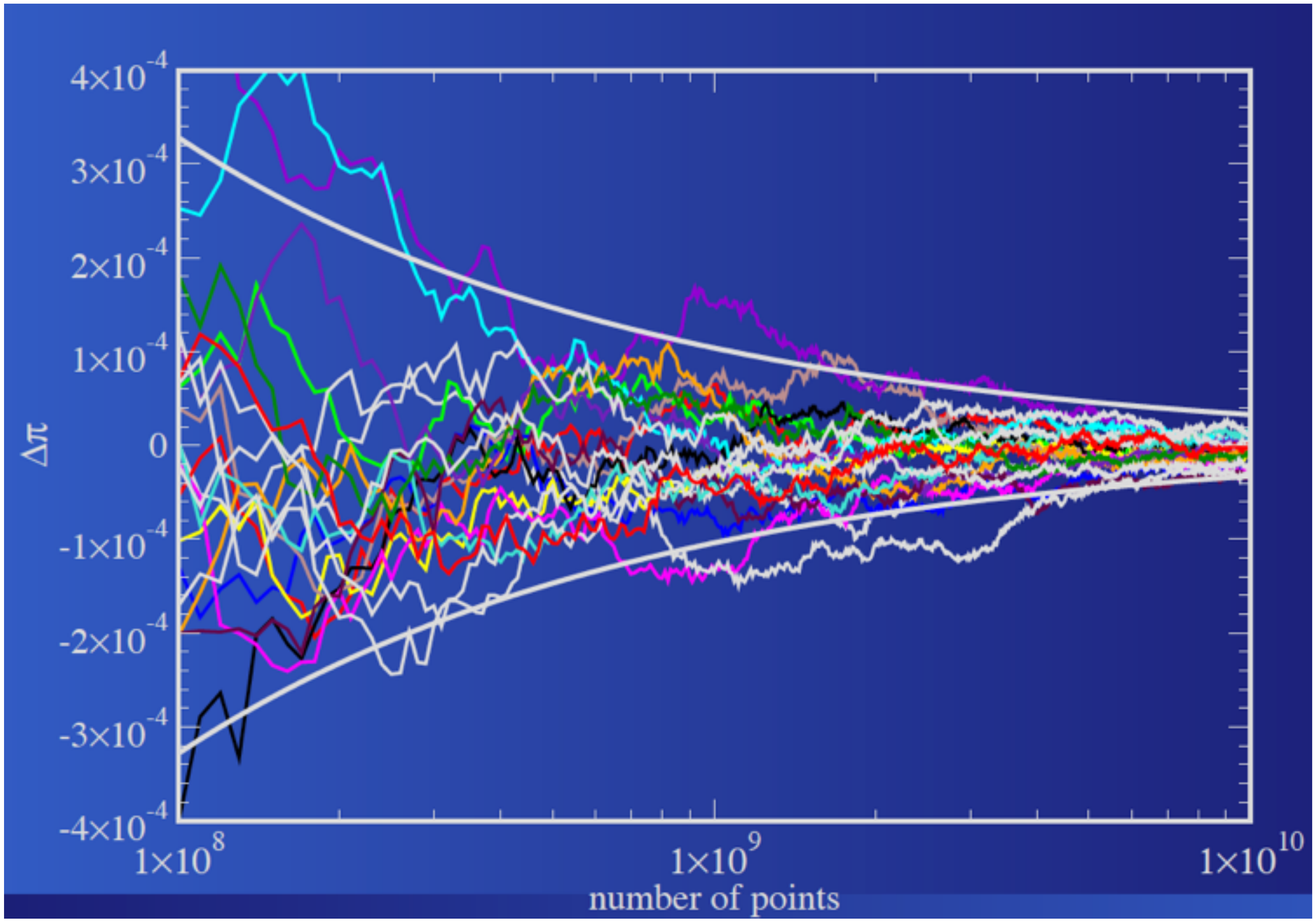
$$p = \frac{A_c}{A_s} = \frac{\pi}{4} = Nc/Ns$$

Random points: N

Random points inside circle: Nc

$$\pi = \frac{4N_c}{N}$$

Variance reduction



Uncertainty and variance reduction

Variance reduction or efficiency increase?

$$\varepsilon = \frac{1}{s^2 T}$$

S^2 variance

T computation time

We must avoid to make the calculation more efficient at the cost of computing results

It the same definition valid in simulation related to radiation treatment?

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

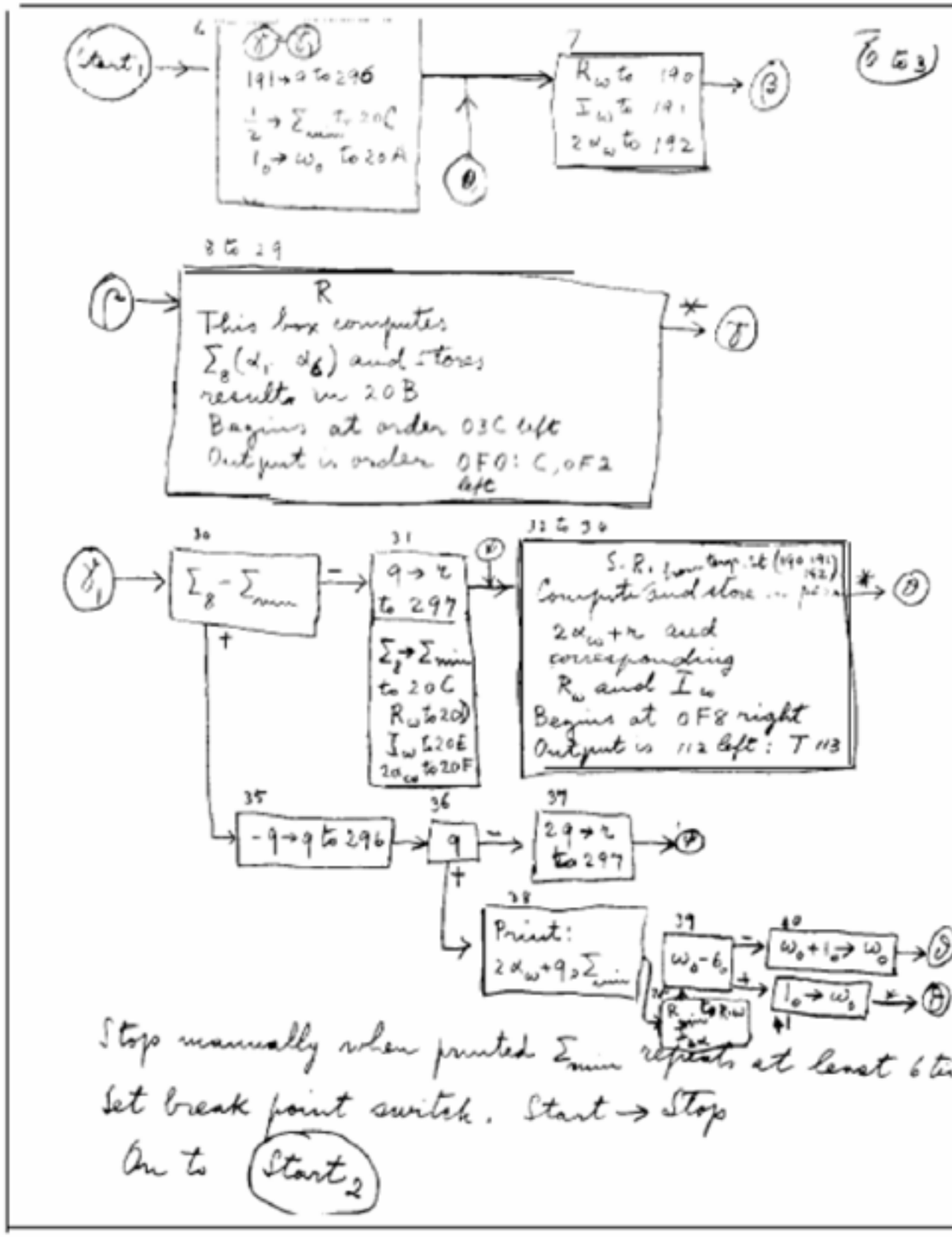
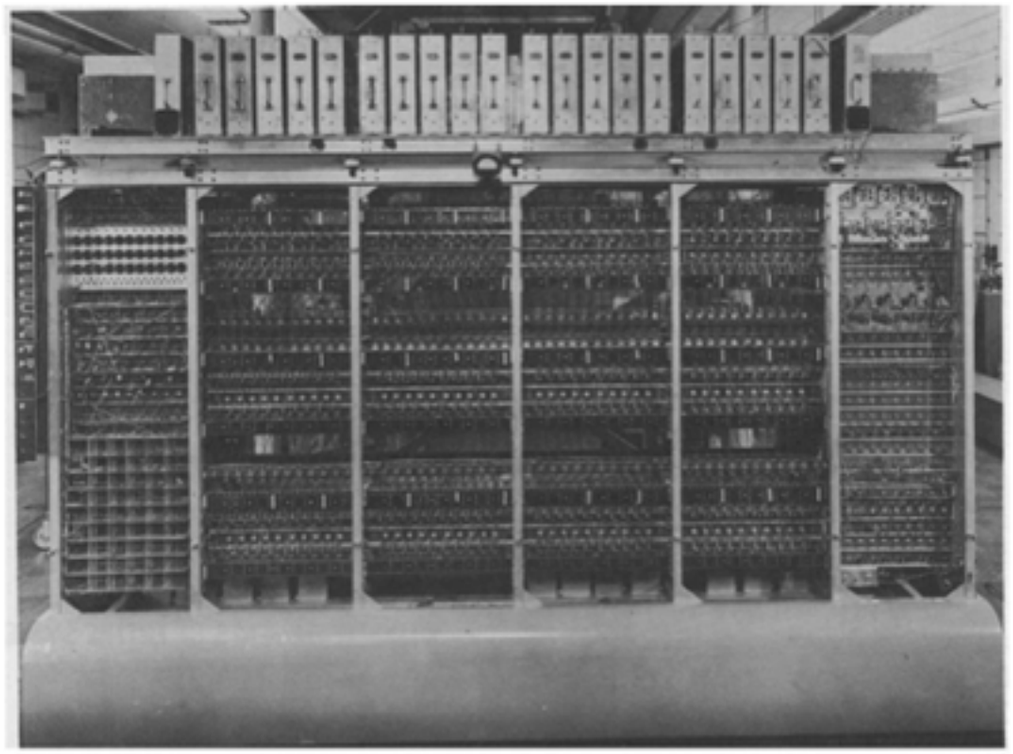


Fig. 4. A subprogram written by Fermi for calculating phase shifts by finding a minimum chi-squared in a fit to the data.



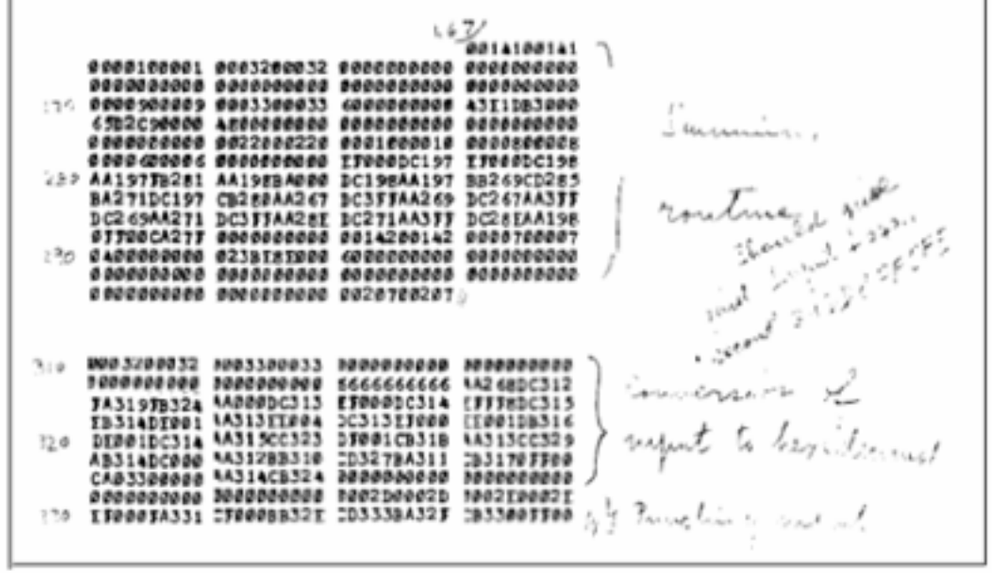


Fig. 5. A portion of the printout of the program containing the subprograms described in Figs. 3 and 4. The program is written in machine language in hexadecimal numbers.

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;

Geant4

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

.....

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 17th, 2016 - Geant4 10.2 patch 02 release ← Current version in the VM

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 17th, 2016 - Geant4 10.2 patch 02 release ← Current version in the VM

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

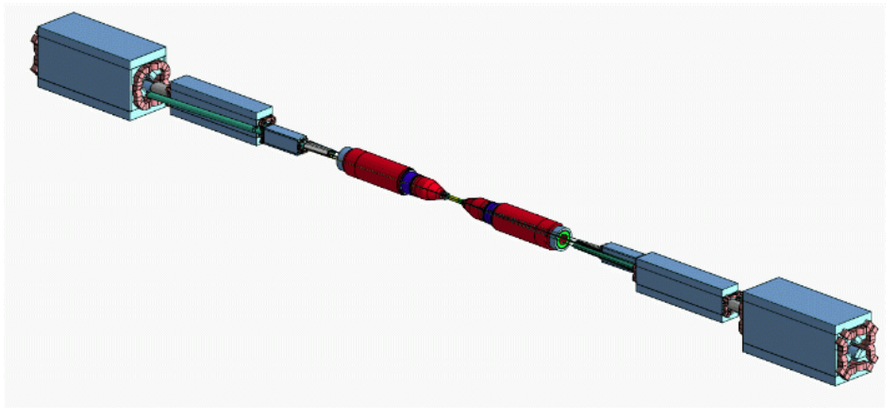
Backward compatible with the sequential mode

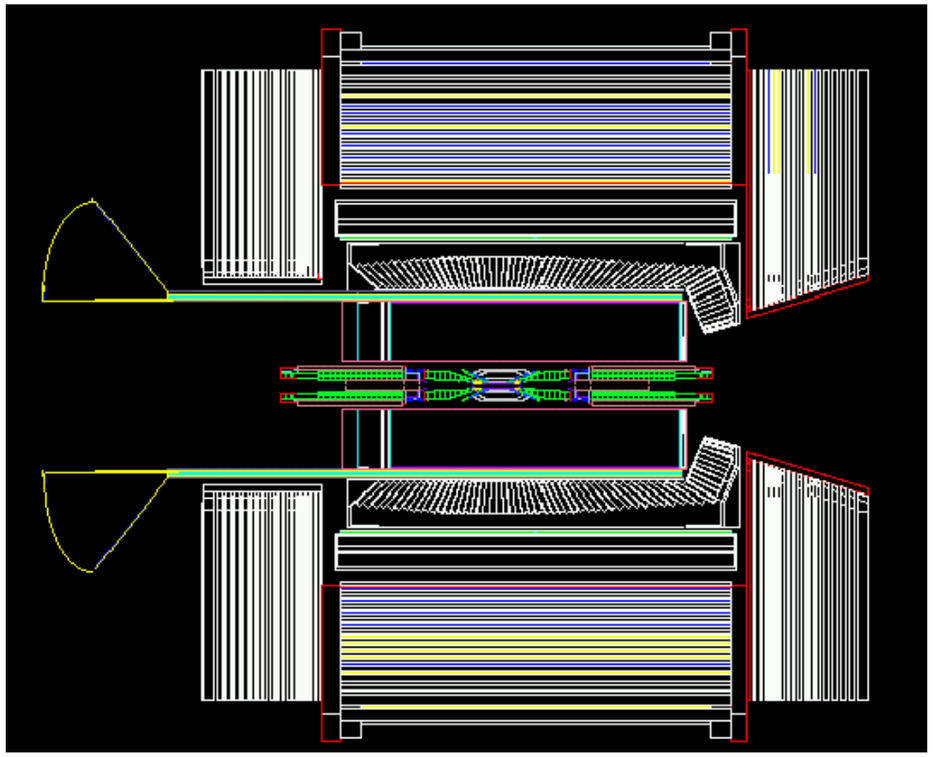
MT programming requires some cares

- 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

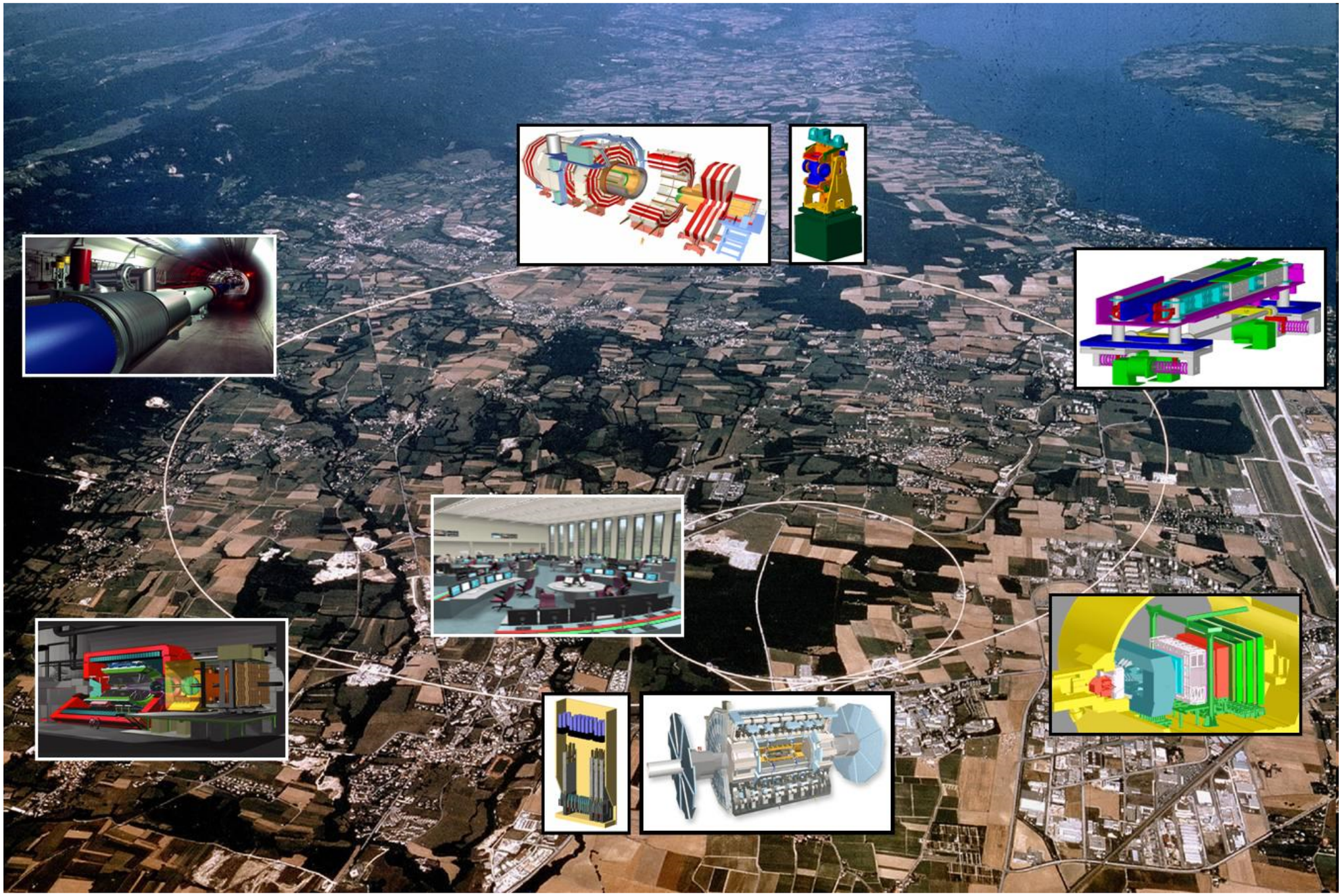
Design of the experiment

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

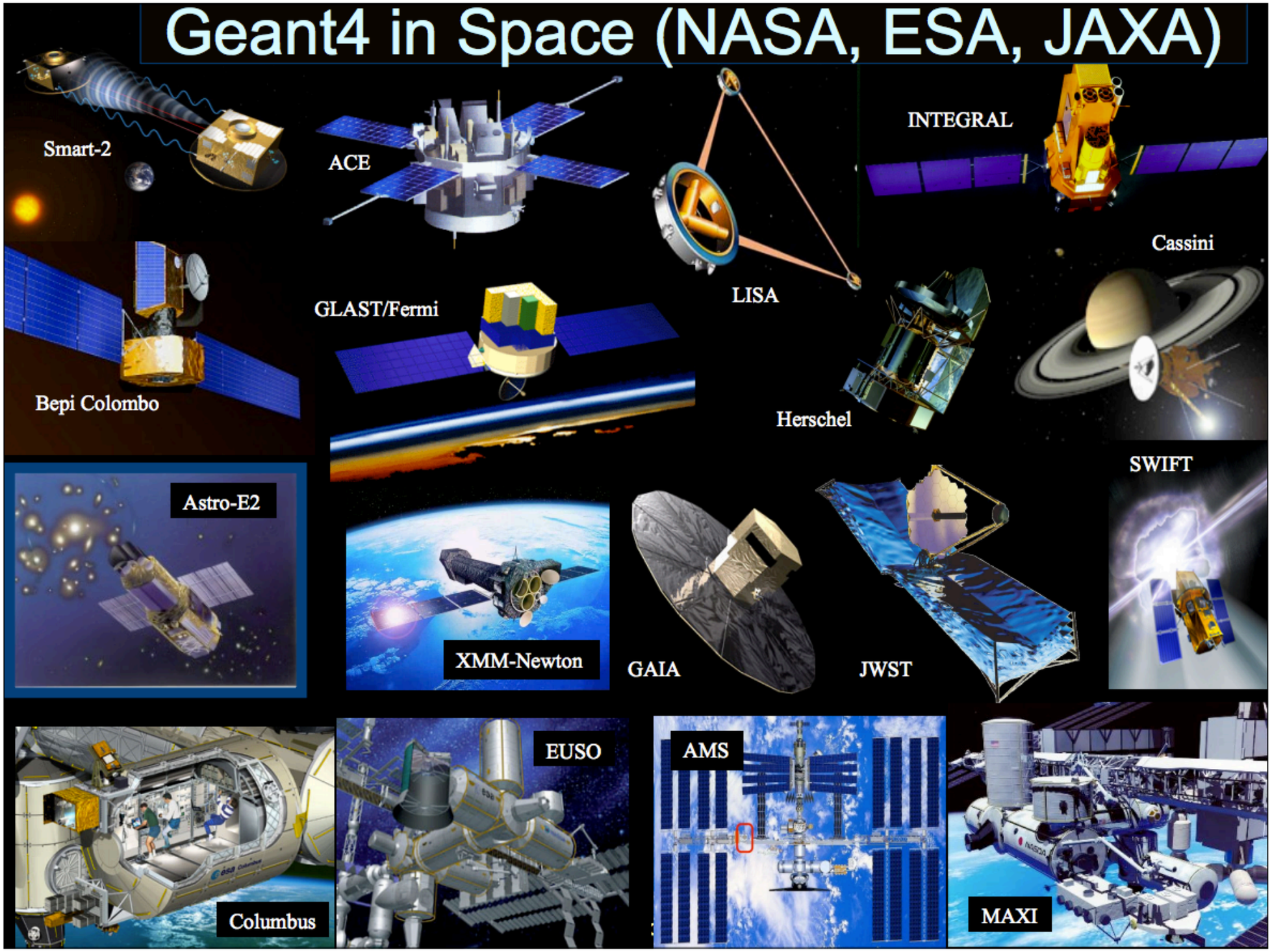
Running/analysis phase

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

Facts about Geant4



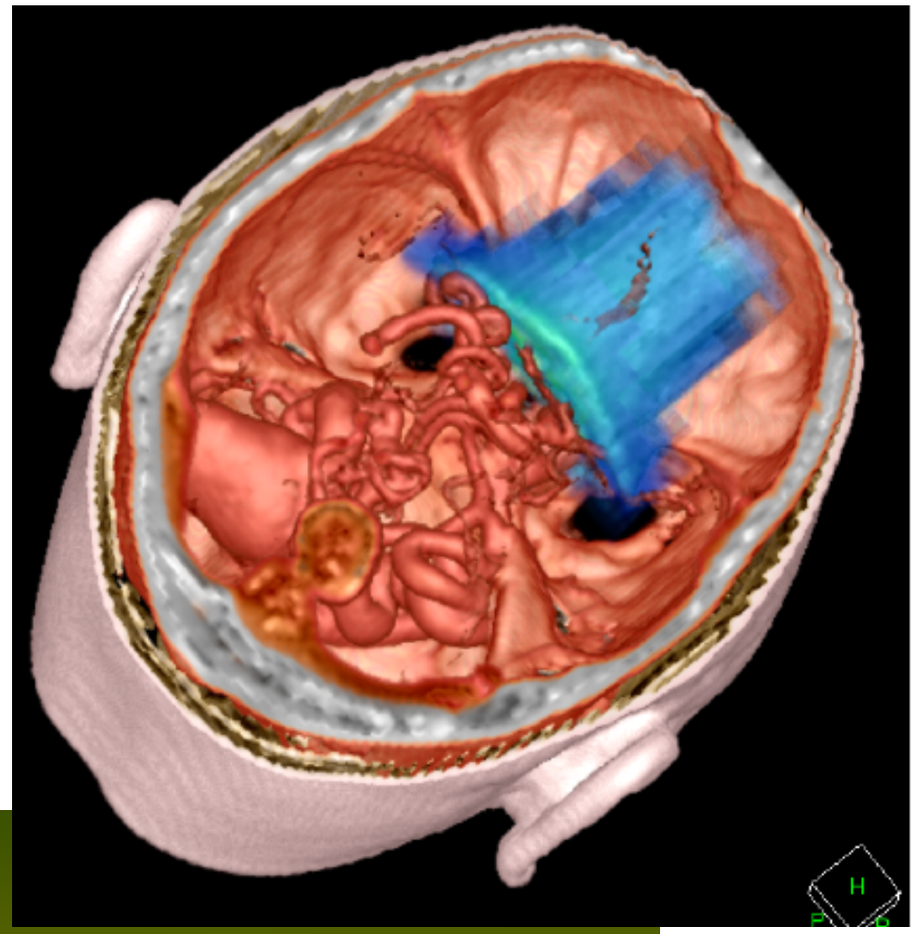
Facts about Geant4

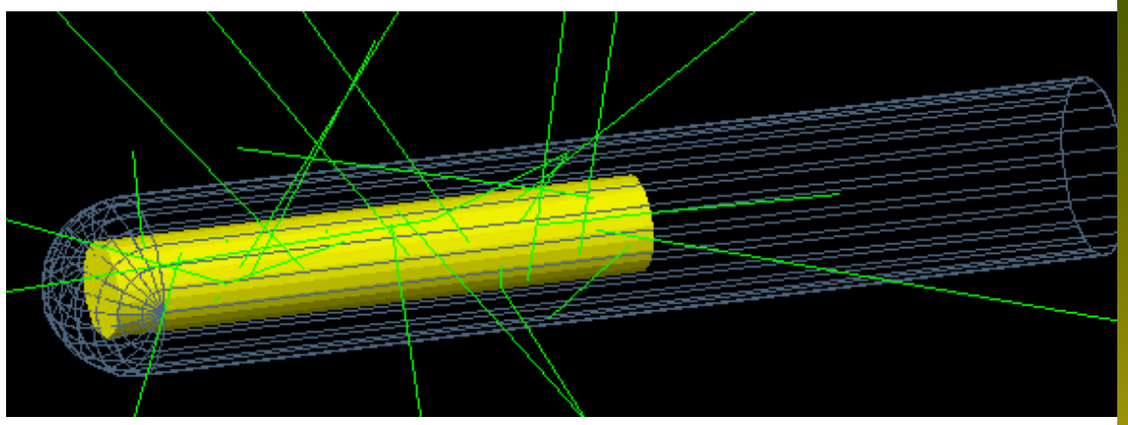


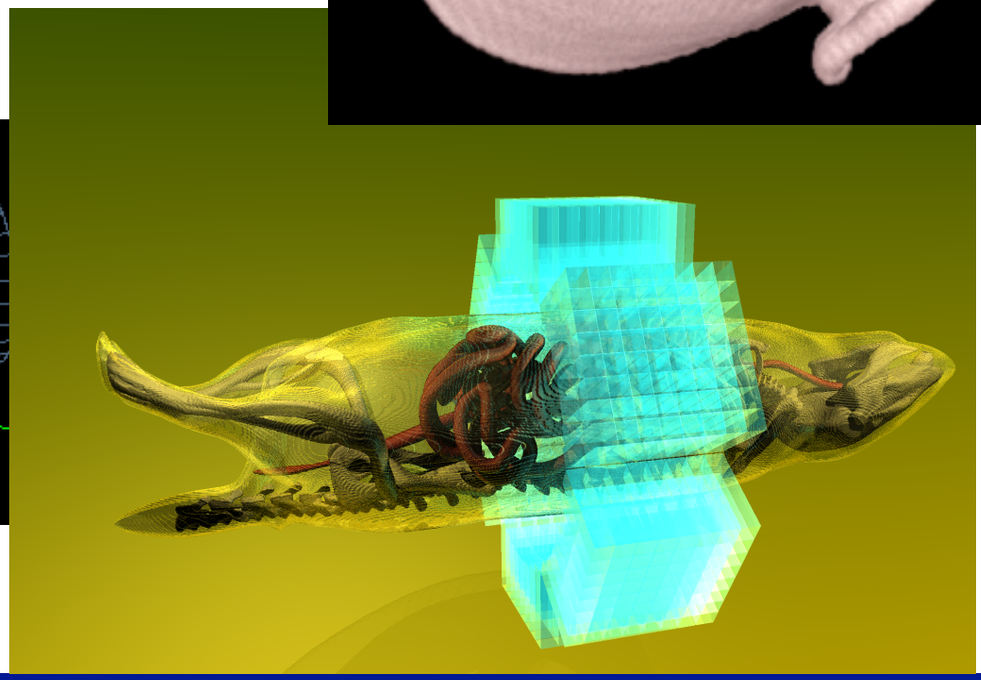
Facts about Geant4

Major use cases

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







Basic concepts and Geant4 capabilities

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 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++ (c11 compliant)

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

Geant4 basics

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)

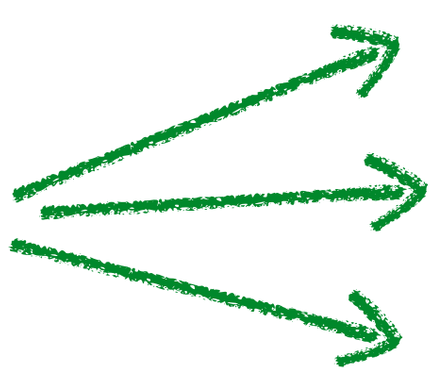
The PrimaryGeneratorAction (.cc and .hh)

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

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()

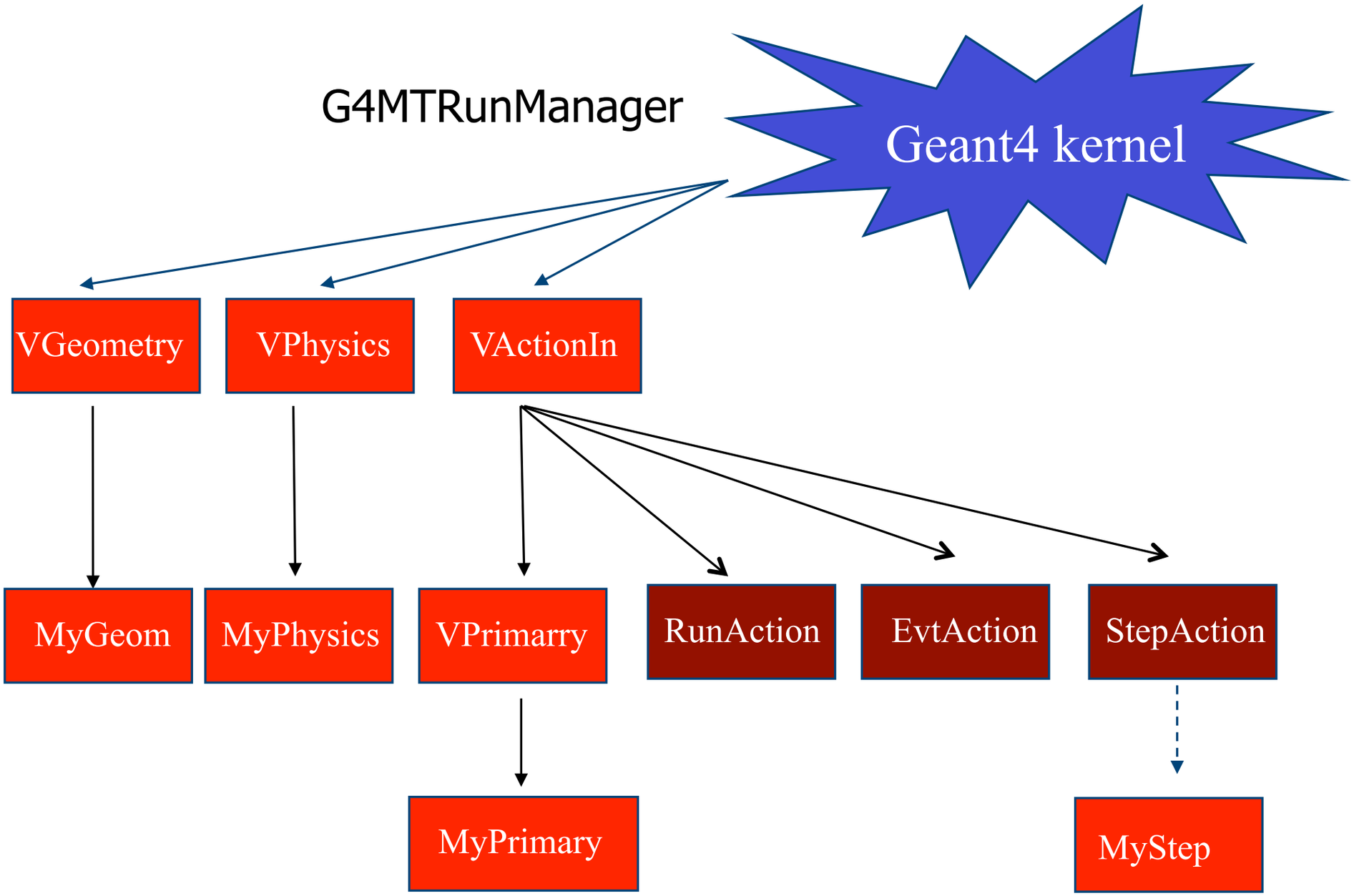
The user MAY define in his/her main():

- 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

Geant4 Concepts with the MultiThreads



Main()


```
66      auto runManager = new RunManager();
67
68      #ifdef G4VIS_USE
69          G4VisManager* visManager = new G4VisExecutive("quiet");
70          visManager->SetVerboseLevel(0);     // Default, you can always override this using macro commands
71          visManager->Initialize();
72      #endif
73
74      runManager->SetUserInitialization(new PhysicsList());
75
76      // Task 1: See that we instantiate the detector construction here
77      runManager->SetUserInitialization(new DetectorConstruction());
78      runManager->SetUserInitialization(new ActionInitialization());
79
80      #ifdef G4UI_USE
81          G4UIExecutive* ui;
82          if (interactive)
83          {
84              ui = new G4UIExecutive(argc, argv);
85          }
86      #endif
87
88      G4UImanager* UImanager = G4UImanager::GetUIpointer();
89
90      // Task 4b.1: You need to access the scoring manager here (or above)
91
92      for (auto macro : macros)
93      {
94          G4String command = "/control/execute ";
95          UImanager->ApplyCommand(command + macro);
96      }
97
98  #ifdef G4UI_USE
99      if (interactive)
100       {
101           UImanager->ApplyCommand("/control/execute macros/ui.mac");
102           ui->SessionStart();
103           delete ui;
104       }
105 #endif
106
107     delete runManager;
108
109     std::cout << "Application successfully ended.\nBye :-)" << std::endl;
110
111     return 0;
112 }
```

Main()

RunManager initialisation

```
66  auto runManager = new RunManager();
67
68  #ifdef G4VIS_USE
69      G4VisManager* visManager = new G4VisExecutive("quiet");
70      visManager->SetVerboseLevel(0);     // Default, you can always override this using macro commands
71      visManager->Initialize();
72  #endif
73
74  runManager->SetUserInitialization(new PhysicsList());
75
76  // Task 1: See that we instantiate the detector construction here
77  runManager->SetUserInitialization(new DetectorConstruction());
78  runManager->SetUserInitialization(new ActionInitialization());
79
80  #ifdef G4UI_USE
81      G4UIExecutive* ui;
82      if (interactive)
83      {
84          ui = new G4UIExecutive(argc, argv);
85      }
86  #endif
87
88      G4UImanager* UImanager = G4UImanager::GetUIpointer();
89
90      // Task 4b.1: You need to access the scoring manager here (or above)
91
92      for (auto macro : macros)
93      {
94          G4String command = "/control/execute ";
95          UImanager->ApplyCommand(command + macro);
96      }
97
98  #ifdef G4UI_USE
99      if (interactive)
100       {
101           UImanager->ApplyCommand("/control/execute macros/ui.mac");
102           ui->SessionStart();
103           delete ui;
104       }
105 #endif
106
107     delete runManager;
108
109     std::cout << "Application successfully ended.\nBye :-)" << std::endl;
110
111     return 0;
112 }
```

Main()


```
66  auto runManager = new RunManager();
67
68  #ifdef G4VIS_USE
69      G4VisManager* visManager = new G4VisExecutive("quiet");
70      visManager->SetVerboseLevel(0);    // Default, you can always override this using macro commands
71      visManager->Initialize();
72  #endif
73
74  runManager->SetUserInitialization(new PhysicsList());
75
76  // Task 1: See that we instantiate the detector construction here
77  runManager->SetUserInitialization(new DetectorConstruction());
78  runManager->SetUserInitialization(new ActionInitialization());
79
80  #ifdef G4UI_USE
81      G4UIExecutive* ui;
82      if (interactive)
83      {
84          ui = new G4UIExecutive(argc, argv);
85      }
86  #endif
87
88      G4UImanager* UImanager = G4UImanager::GetUIpointer();
89
90      // Task 4b.1: You need to access the scoring manager here (or above)
91
92      for (auto macro : macros)
93      {
94          G4String command = "/control/execute ";
95          UImanager->ApplyCommand(command + macro);
96      }
97
98  #ifdef G4UI_USE
99      if (interactive)
100       {
101           UImanager->ApplyCommand("/control/execute macros/ui.mac");
102           ui->SessionStart();
103           delete ui;
104       }
105 #endif
106
107     delete runManager;
108
109     std::cout << "Application successfully ended.\nBye :-)" << std::endl;
110
111     return 0;
112 }
```

RunManager initialisation

Visualisation initialisation

Main()


```
66  auto runManager = new RunManager();
67
68  #ifdef G4VIS_USE
69      G4VisManager* visManager = new G4VisExecutive("quiet");
70      visManager->SetVerboseLevel(0);    // Default, you can always override this using macro commands
71      visManager->Initialize();
72  #endif
73
74  runManager->SetUserInitialization(new PhysicsList());
75
76  // Task 1: See that we instantiate the detector construction here
77  runManager->SetUserInitialization(new DetectorConstruction());
78  runManager->SetUserInitialization(new ActionInitialization());
79
80  #ifdef G4UI_USE
81      G4UIExecutive* ui;
82      if (interactive)
83      {
84          ui = new G4UIExecutive(argc, argv);
85      }
86  #endif
87
88      G4UImanager* UImanager = G4UImanager::GetUIpointer();
89
90      // Task 4b.1: You need to access the scoring manager here (or above)
91
92      for (auto macro : macros)
93      {
94          G4String command = "/control/execute ";
95          UImanager->ApplyCommand(command + macro);
96      }
97
98  #ifdef G4UI_USE
99      if (interactive)
100       {
101           UImanager->ApplyCommand("/control/execute macros/ui.mac");
102           ui->SessionStart();
103           delete ui;
104       }
105 #endif
106
107     delete runManager;
108
109     std::cout << "Application successfully ended.\nBye :-)" << std::endl;
110
111     return 0;
112 }
```

RunManager initialisation

Visualisation initialisation

User initialisation

Main()


```
66  auto runManager = new RunManager();
67
68  #ifdef G4VIS_USE
69      G4VisManager* visManager = new G4VisExecutive("quiet");
70      visManager->SetVerboseLevel(0);     // Default, you can always override this using macro commands
71      visManager->Initialize();
72  #endif
73
74  runManager->SetUserInitialization(new PhysicsList());
75
76  // Task 1: See that we instantiate the detector construction here
77  runManager->SetUserInitialization(new DetectorConstruction());
78  runManager->SetUserInitialization(new ActionInitialization());
79
80  #ifdef G4UI_USE
81      G4UIExecutive* ui;
82      if (interactive)
83      {
84          ui = new G4UIExecutive(argc, argv);
85      }
86  #endif
87
88      G4UImanager* UImanager = G4UImanager::GetUIpointer();
89
90      // Task 4b.1: You need to access the scoring manager here (or above)
91
92      for (auto macro : macros)
93      {
94          G4String command = "/control/execute ";
95          UImanager->ApplyCommand(command + macro);
96      }
97
98  #ifdef G4UI_USE
99      if (interactive)
100       {
101           UImanager->ApplyCommand("/control/execute macros/ui.mac");
102           ui->SessionStart();
103           delete ui;
104       }
105 #endif
106
107     delete runManager;
108
109     std::cout << "Application successfully ended.\nBye :-)" << std::endl;
110
111     return 0;
112 }
```

RunManager initialisation

Visualisation initialisation

User initialisation

User interface
(simulation run)

Main()


```
66  auto runManager = new RunManager();
67
68  #ifdef G4VIS_USE
69      G4VisManager* visManager = new G4VisExecutive("quiet");
70      visManager->SetVerboseLevel(0);     // Default, you can always override this using macro commands
71      visManager->Initialize();
72  #endif
73
74  runManager->SetUserInitialization(new PhysicsList());
75
76  // Task 1: See that we instantiate the detector construction here
77  runManager->SetUserInitialization(new DetectorConstruction());
78  runManager->SetUserInitialization(new ActionInitialization());
79
80  #ifdef G4UI_USE
81      G4UIExecutive* ui;
82      if (interactive)
83      {
84          ui = new G4UIExecutive(argc, argv);
85      }
86  #endif
87
88      G4UImanager* UImanager = G4UImanager::GetUIpointer();
89
90      // Task 4b.1: You need to access the scoring manager here (or above)
91
92      for (auto macro : macros)
93      {
94          G4String command = "/control/execute ";
95          UImanager->ApplyCommand(command + macro);
96      }
97
98  #ifdef G4UI_USE
99      if (interactive)
100       {
101           UImanager->ApplyCommand("/control/execute macros/ui.mac");
102           ui->SessionStart();
103           delete ui;
104       }
105 #endif
106
107     delete runManager;
108
109     std::cout << "Application successfully ended.\nBye :-)" << std::endl;
110
111     return 0;
112 }
```

RunManager initialisation

Visualisation initialisation

User initialisation

User interface
(simulation run)

Delete RunManager

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());
  }
```

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
  - ....

User Interface (how to run a simulation)

Run a simulation

- A Geant4 simulation can be steered in three ways
  - Everything is hard-coded in the C++ source (you need to recompile at any change)
  - Batch session (via an ASCII macro file)
  - In interactive mode (the session bar you will see also in our exercises)

Scene tree | Help | History

viewer-0 (OpenGLStoredQt)

Scene tree

Scene tree : viewer-0 (OpenGLStoredQt)

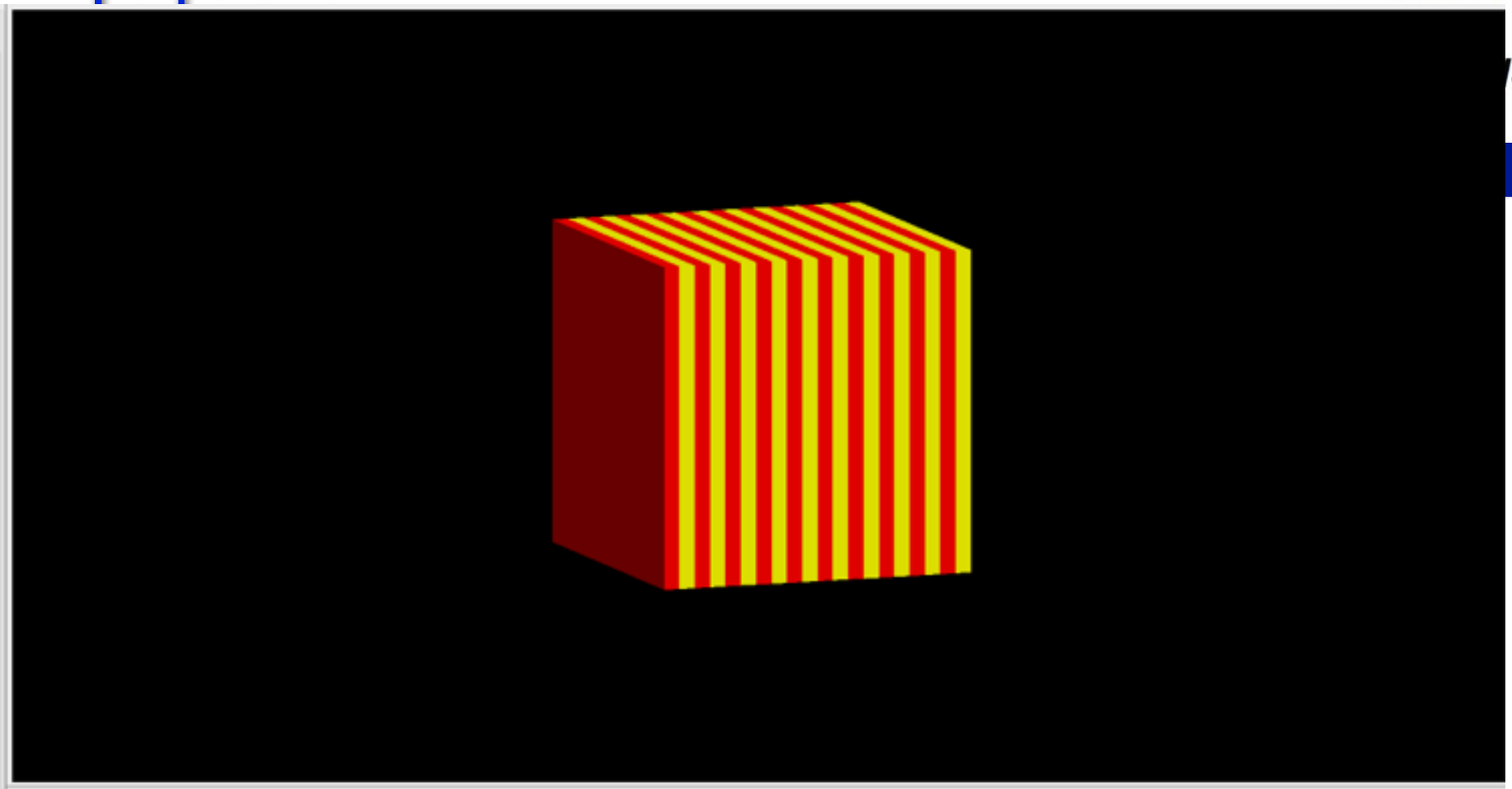
- Touchables
  - world [0]
    - absorber0 [0]
    - scintillator0 [0]
    - absorber1 [1]
    - scintillator1 [1]
    - absorber2 [2]
    - scintillator2 [2]
    - absorber3 [3]

Show all — Hide all

Viewer properties

Property	Value
autoRefresh	True
auxiliaryEdge	False
background	0 0 0 1
culling	1
cutawayMode	union
defaultColour	1 1 1 1
defaultTextColour	0 0 1 1
edge	False
explodeFactor	1 1 mm
globalLineWidthScale	1
globalMarkerScale	1

Picking informations — Picking mode active



Output

```
/tracking/storeTrajectory 2
```

Session :

Run a simulation

- Setting up the batch-mode (read commands from a macro file) in the main()

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

takes the first argument after the executable as the macro name and runs it

- Your executable can be run as:
  myExecutable mymacro.mac
- To execute a macro interactively
  /control/execute mymacro.mac

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

Geant 4

Home > User Support > Download

Geant4 Software Download

Geant4 10.2

first released 4 December 2015 (patch-01, released 26 February 2016)

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

Please read the **Release Notes** before downloading or using this release.
The patch below contains bug fixes to release 10.2, we suggest you to download and apply the latest patch for release 10.2 (see the additional notes for patch-01). or download the complete source with the patch applied; in any case, it is required to apply a full rebuild of the libraries.

Source files

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

Download GNU or Linux tar format, compressed using gzip (30.7Mb, 32207829 bytes)
After downloading, gunzip, then unpack using GNU tar.

Download ZIP format (43.8Mb, 45930086 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 G4NDL4.5, Neutron data files with thermal cross-sections - version 4.5 (402.2Mb, 421710294 bytes)

Download G4EMLOW6.48, data files for low energy electromagnetic processes - version 6.48 (24.2Mb, 25371256 bytes) NEW

Download G4PhotonEvaporation3.2, data files for photon evaporation - version 3.2 (8.9Mb, 9300395 bytes) NEW

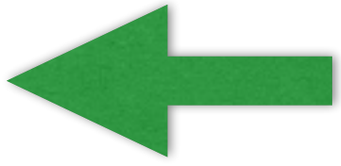
Download G4RadioactiveDecay4.3.1, data files for radioactive decay hadronic processes - version 4.3.1 (7.6Mb, 7956480 bytes) NEW

Download G4SAIDDATA1.1, data files from evaluated cross-sections in SAID data-base - version 1.1 (25.2kb, 25800 bytes)

Download G4NEUTRONXS1.4, data files for evaluated neutron cross-sections on natural composition of elements - version 1.4 (2.1Mb, 2249001 bytes)

Run the installation script in any new terminal:

source /usr/local/geant4/geant4.10.02.p02-install/bin/geant4.sh



The End