



**GEANT4**  
A SIMULATION TOOLKIT

# Random Numbers

---

**Geant4 PHENIICS & IN2P3 Tutorial,**  
16 – 20 May 2022,  
Orsay

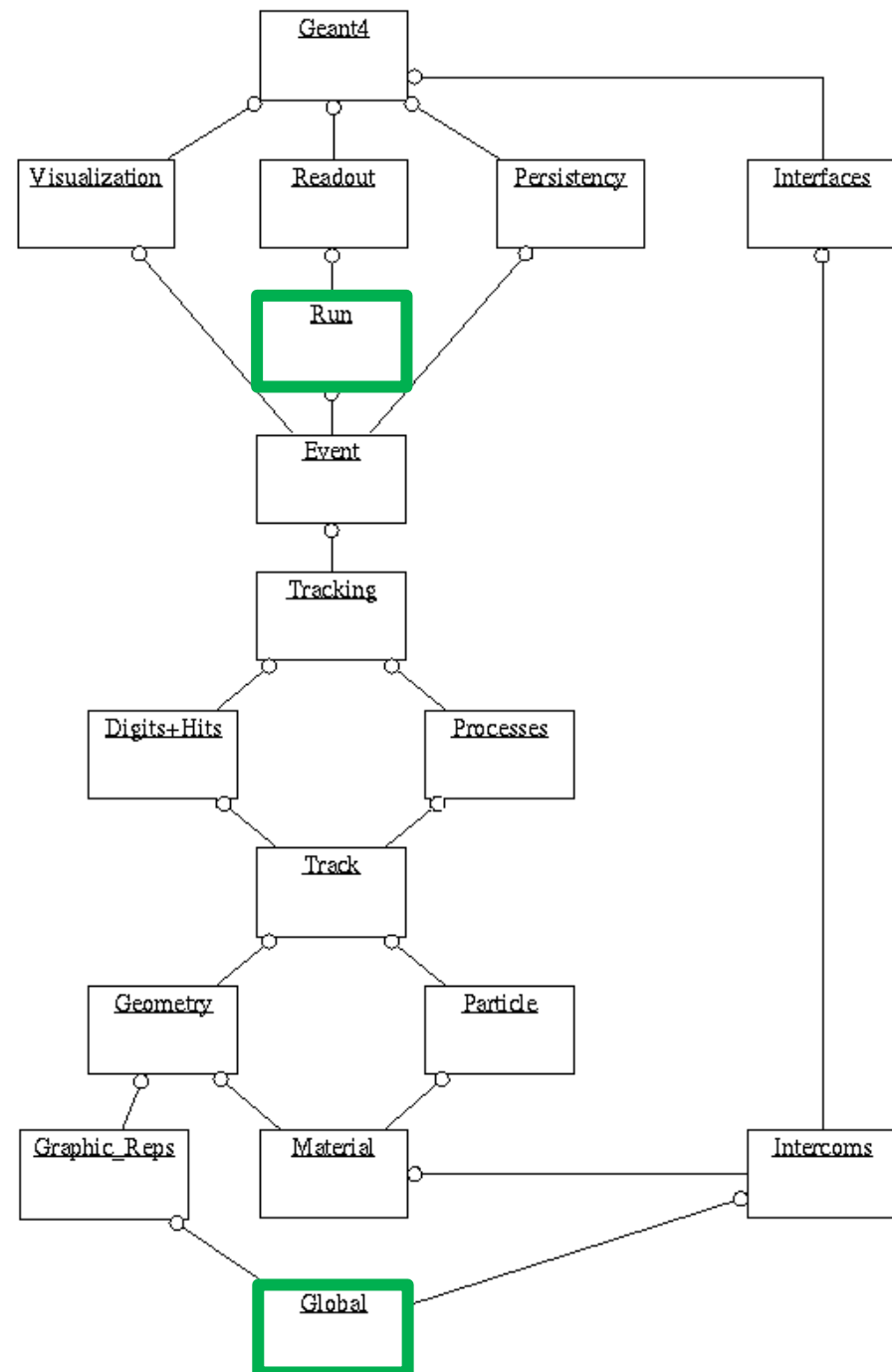
Marc Verderi  
LLR, Ecole polytechnique



# Where will we look in the toolkit ?

Main categories and directories involved:

- Global:
  - `geant4/source/global/HEPRandom`
- Run
  - `geant4/source/run`



# Introduction

- Monte Carlo simulation relies on random number generators.
  - To mimic the random nature of processes
- A random number generator is actually a “pseudo-random” generator
  - As it uses a deterministic algorithm
- Several generation techniques exist, with various quality vs. speed characteristics
- A random number generator is often called an “engine”.
- A random number generator uses a “seed” as starting point
  - A seed is an integer or a set of integers
  - And from a same seed, an engine generates always a same sequence of random numbers
- A random number generator has also a “status”:
  - Starting from a given seed, and after N numbers generation, the engine is in some state, that can be saved as a status
  - This status can be reloaded later on, and the engine will continue its generation as if it hasn’t been interrupted
- An engine generates uniform random numbers in the  $]0, 1[$  range.
  - Other distributions (exponential, gaussian, ...) can be obtained from flat distribution(s) with proper transformations.
- Geant4 offers the ability to chose the engine, set the seed, save/reload the status.
  - It is explained here the interest for this and how to do.

You can almost see the seed as a “tag” of a random number series.

# Why controlling the random number generator ?

- There are very common needs for controlling the engine
- **Setting the seed(s):**
  - This allows to run jobs in parallels that will be statistically independent
    - instead of running 10 000 000 events in one “/run/beamOn 10 000 000” command
    - launch 10 jobs in parallel, each having a different seed and issue for each “/run/beamOn 1 000 000”.
    - If you have 10 CPU available, you get your simulation done 10 times faster.
  - Note that with today’s computers, if you have several cores on your machine, you get a simulation speed-up just for free, running as many applications as the number of cores.
  - ***Setting up the seed is heavily used in simulation production (and is the key of this).***
- Saving the random number status
  - Particularly useful in case of crash...
    - in what case you need to investigate the cause of the crash.
  - If a crash happens after 1M events, you would wish to skip the first 999 999 events to go directly to the one that crashed, simulating the *\*same\** event than the one that crashed.
  - Saving the random number status allow this :
    - you save the status after 999 999 events, and reload it to run the problematic event to investigate it.
  - But note that saving the status is time consuming, and is something you would not do by default.
- The choice of the engine relies on speed vs accuracy considerations
  - and is a more technical choice in principle
  - though if this is easy to change the engine by another one in practice.

# Random number generators in Geant4

- Geant4 uses the HEPRandom module of CLHEP library:
  - with documentation from [https://geant4.web.cern.ch/support/user\\_documentation](https://geant4.web.cern.ch/support/user_documentation) → User's Guide: For Application Developers → The HEPRandom module in CLHEP
- HepRandomEngine is the abstract interface for random generators in CLHEP.
  - All engines are of type HepRandomEngine
- A static instance exists, allowing the engine to be shared by all random number consumers.
  
- Geant4 uses this static instance of HepRandom.
- And this static instance holds a random number engine
  - This engine can be changed with
  - `G4Random::setTheEngine(CLHEP::HepRandomEngine*);`
  - **Don't use** `CLHEP::HepRandom::setTheEngine(CLHEP::HepRandomEngine*)` ! **Not thread safe !**
  - But a default engine exists, which is  

`HepJamesRandom`
  - Above default made that you did not need to care about random numbers up to now.
- (Some) other existing engines in CLHEP are:
  - `DRand48Engine`, `RandEngine`, `RanluxEngine` –which allows controlling the quality of the engine by “luxury” levels- , `RanecuEngine`
- Changing the engine is the only things that requires (by default) C++ coding.
  - All other actions to control the random number generator can be done interactively.

# Controlling/piloting the random number engine interactively

- Beyond engine choice, control of the engine can be done interactively
- Commands can be obtained simply with the help menu
- `/random/setSeeds int [int [int [...]]]`
  - Set the seed(s), the number of ints, depends on the engine
- `/random/setDirectoryName [dirName]`
  - Set or create the directory in which to save the engine status
- `/random/setSavingFlag [value]`
  - Turn on(off) status change at each start of run and event
  - Status are saved in `currentRun.rndm` (for run) and `currentEvent.rndm` (event)
- `/random/saveThisRun( or saveThisEvent)`
  - copy `currentRun.rndm` to `runXXX.rndm` (or `currentEvent.rndm` to `runXXXevtYYY.rndm`)
- `/random/resetEngineFrom [fileName]`
  - Restore the engine status from the given file name
  - The directory where file is stored had to be previously set by `/random/setDirectoryName`

# Controlling/piloting the random number engine with C++

- Previous interactive commands are exploiting existing C++ methods that provide the functionalities.
- Common methods used:
  - Set the seed(s):
    - `G4Random::setSeed(long seed, int);`
    - `G4Random::setSeeds(const long * seeds, int);`
  - Save the engine status into a file:
    - `G4Random::saveEngineStatus(const char filename[] = "Config.conf");`
  - Restore the engine status from a file:
    - `G4Random::restoreEngineStatus(const char filename[] = "Config.conf");`
  - Display the engine status:
    - `G4Random::showEngineStatus();`
- More can be found in [documentation](#) and in base class header file:
  - `geant4/source/externals/clhep/include/CLHEP/Random/RandomEngine.h`
- You may need some of the above methods in today's exercise.
  - Note : you may build you file names with `"std::ostringstream fileName;"` and to get the `"char *"` do `"fileName.str().c_str();"`

# Summary

- In a first approach, you don't need to care about random number configuration in Geant4
  - A default configuration exists
- For more serious needs, you will likely need to
  - Set the seed(s)
    - To run multiple jobs, statistically independent
  - Chose your favorite random number engine
  - Save / restore the random number engine status
    - For example in case of a crash happening rarely and you want to investigate
- These can be done with
  - C++ coding
  - Or more simply, interactive commands
    - For setting the seed(s) and saving/restoring the random number engine state

