Montecarlo & GEANT

UNIVERSITÀ DELLA CALABRIA

Salvatore Fazio

Università della Calabria & INFN Cosenza

INFN Gruppo Collegato di Cosenza
Istituto Nazionale di Fisica Nucleare

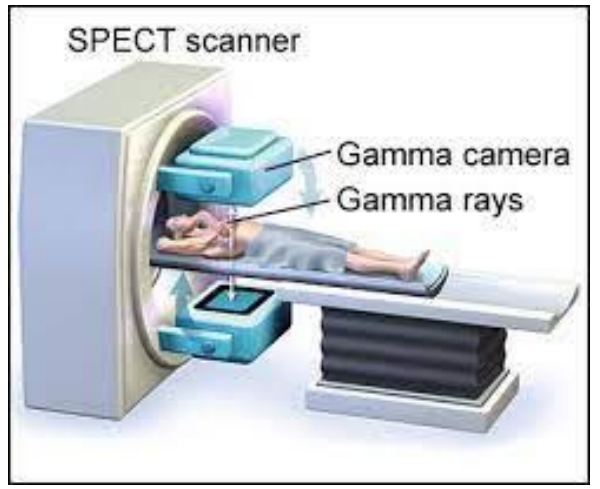
Second European School on the Physics of the Electron-Ion Collider

Benicasim, Valencia - June 2025

- The “GEometry ANd Tracking” toolkit was developed by an International Collaboration established in 1998
  - ~100 members, from Europe, US and Japan
  - GEANT4: open source, written in C++
    - Takes full advantage from an object oriented language (GEANT3 was in Fortran)
  - Official website: http://geant4.org
- References:
  - S. Agostinelli et al., *Geant4, a simulation toolkit*, Nucl. Inst. and Meth. Phys. Res. A, 506 250-303
  - J. Allison et al., *Geant4 developments and applications*, IEEE Transaction on Nuclear Science 53, 270-278
- Now the most used toolkit for detector simulation in particle physics experiments…
  - … but not only for that!
  - GEANT4 has a wide range of applications beyond particle physics

Applications of GEANT4

- Medical physics and health:
  - Hadrontherapy
  - Radiobiology
  - Radio diagnostics
  - and much more…



- Space and radiation science
  - Radio-protection in space missions
  - Shielding for satellites
  - Single event upset and radiation damages to electronics
  - Simulations for nuclear spallation sources
  - Radioactive waste

How to install GEANT4

https://geant4-userdoc.web.cern.ch/UsersGuides/InstallationGuide/html/index.html

- **Build and Install Geant4 from Source**
  - Geant4 uses CMake to configure a build system for compiling and installing the toolkit headers, libraries and support tools from scratch
  - Geant4 System/Software Prerequisites for the operating system and software requirements
  - Instructions on how to build Geant4 on **Unix** or **Windows** platforms are provided here
- **Install Geant4 via a Package Manager**
  - Packages are not maintained by the Geant4 developers, but by helpful members of the community
  - macOS/Linux: Spack, Conda, The Arch Users Repository, …

- **Geant4** is a **toolkit** (= a collection of tools)
  - i.e. you cannot "run" it out of the box
  - You **must write an application**, which **uses** Geant4
- **Consequences:**
  - There are **no** such concepts as **"Geant4 defaults"**
  - You must provide the **necessary information** to configure your simulation
  - You must deliberately **choose** which **Geant4 tools** to use
- **Guidance:** many **examples** are provided
  - Basic Examples: overview of Geant4 tools
  - Advanced Examples: Geant4 tools in real-life applications

Basic concepts

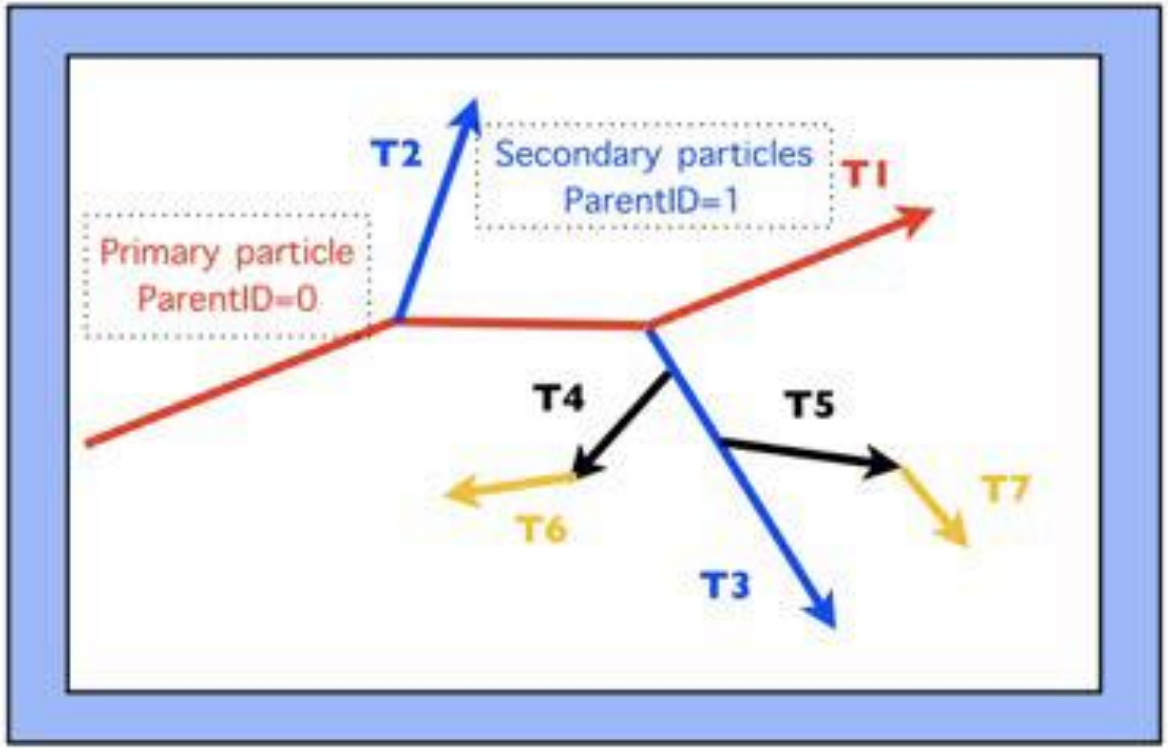
- What you **must do**:
  - Describe your experimental set-up
  - Provide the primary particles input to your simulation
  - Decide which particles and physics models you want to use out of those available in Geant4 and the precision of your simulation (cuts to produce and track secondary particles)
- You **may also want**:
  - To interact with Geant4 kernel to control your simulation
  - To visualise your simulation configuration or results
  - To produce histograms, or ntuples etc. to be further analysed

Main Geant4 capabilities

- Transportation of a particle 'step-by-step' taking into account all possible interactions with materials and fields
- The transport ends if the particle
  - is slowed down to zero kinetic energy (and it doesn't have any interaction at rest)
  - disappears in some interaction
  - reaches the end of the simulation volume
- Geant4 allows the User to access the transportation process and retrieve the results (USER ACTIONS)
  - at the beginning and end of the transport
  - at the end of each step in transportation
  - if a particle reaches a sensitive detector
  - Others…

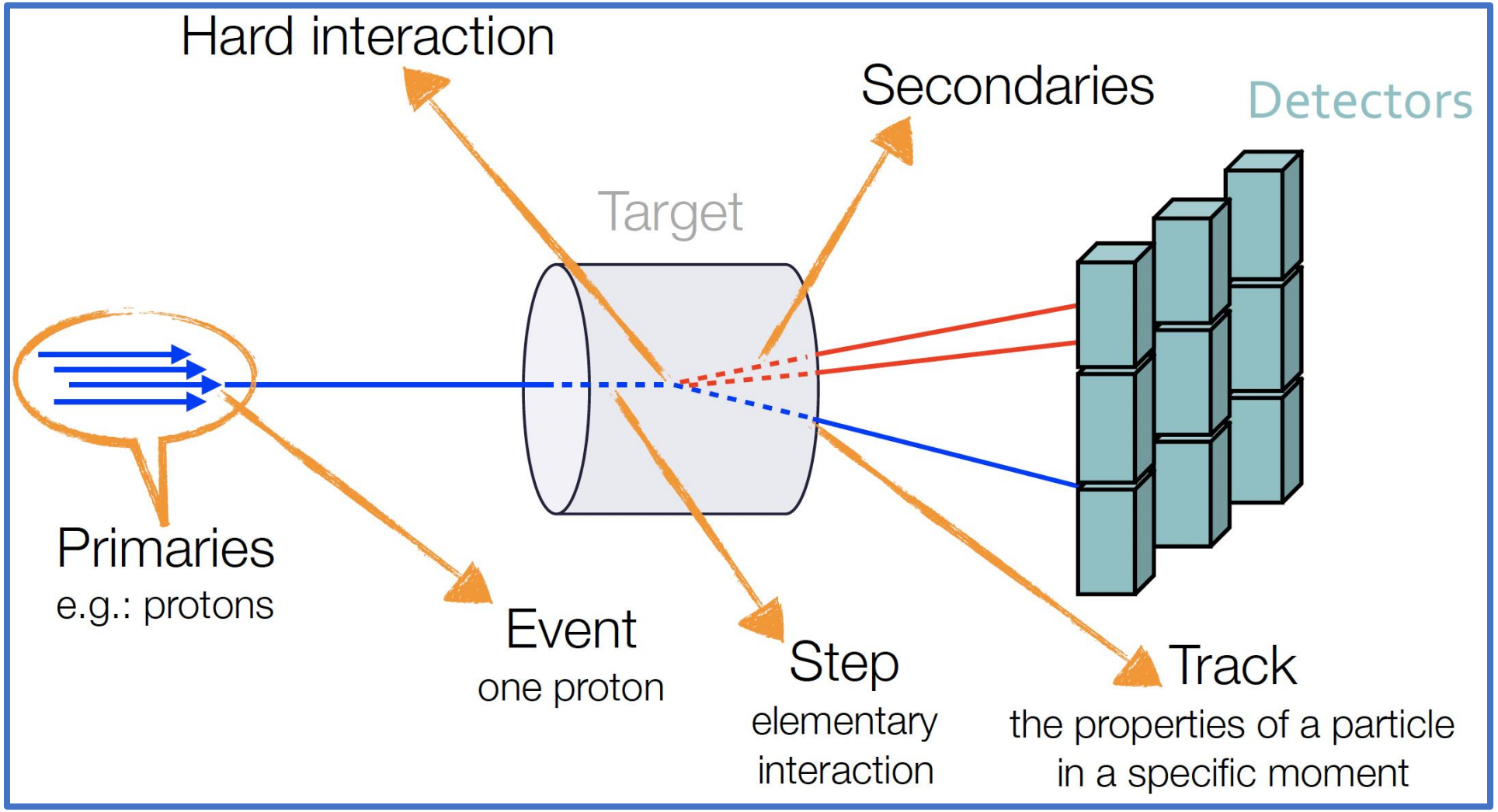
Transport/travel in Geant4

- Step is forced to end at geometry boundaries
  - All **AlongStep** processes **co-occur**
  - The **PostStep** compete, **only one is selected**
- If particle is at rest chose one of the **AtRest** processes
  - The secondaries are saved in the stack
  - To be further tracked with a **last in first out** approach



Just like in a particle experiment

RUN in Geant4

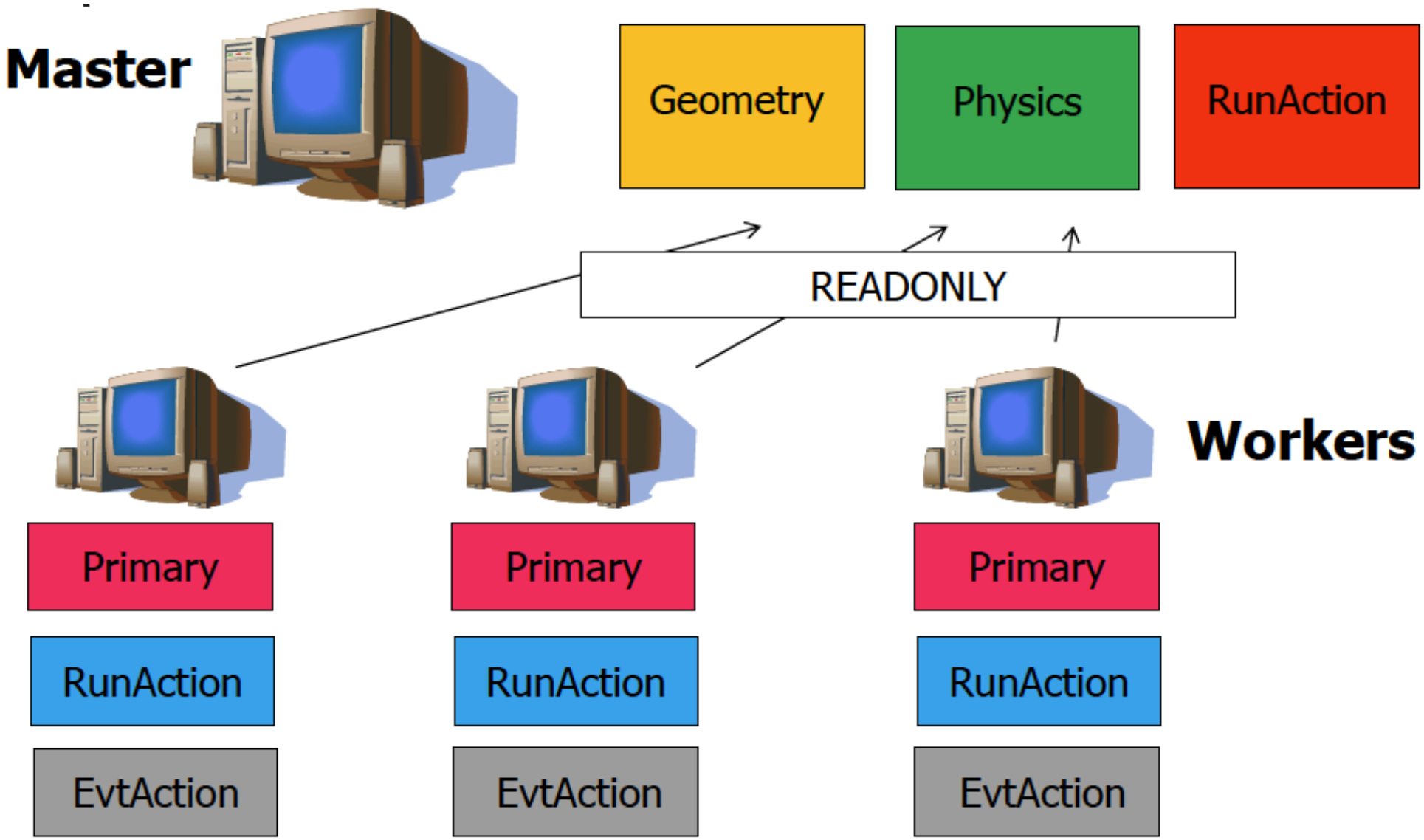


Multi-thread mode

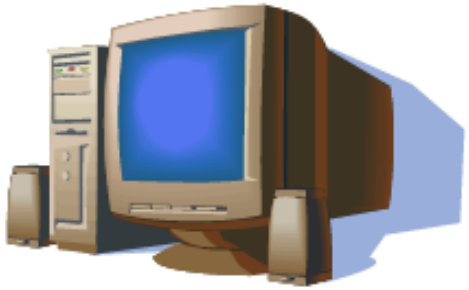
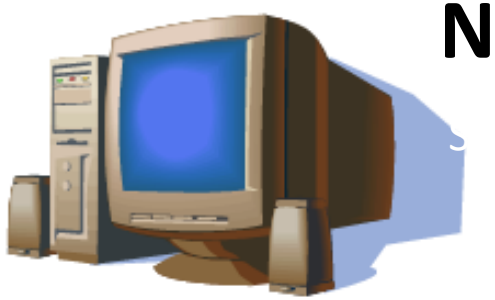
- Geant4 supports **multi-thread approach** for multicore machines
  - Simulation is automatically **split** on an **event-by-event** basis
  - Different events are processed by different cores
  - Can fully profit of all cores available on modern machines

    ⇨ substantial speed-up of simulations
  - Unique copy (master) of geometry and physics
    - All cores have them as read-only (saves memory)
- Backwards compatible with the sequential mode
  - The Multi-thread (MT) programming requires some care: need to avoid conflicts between threads
  - Some modification and porting required

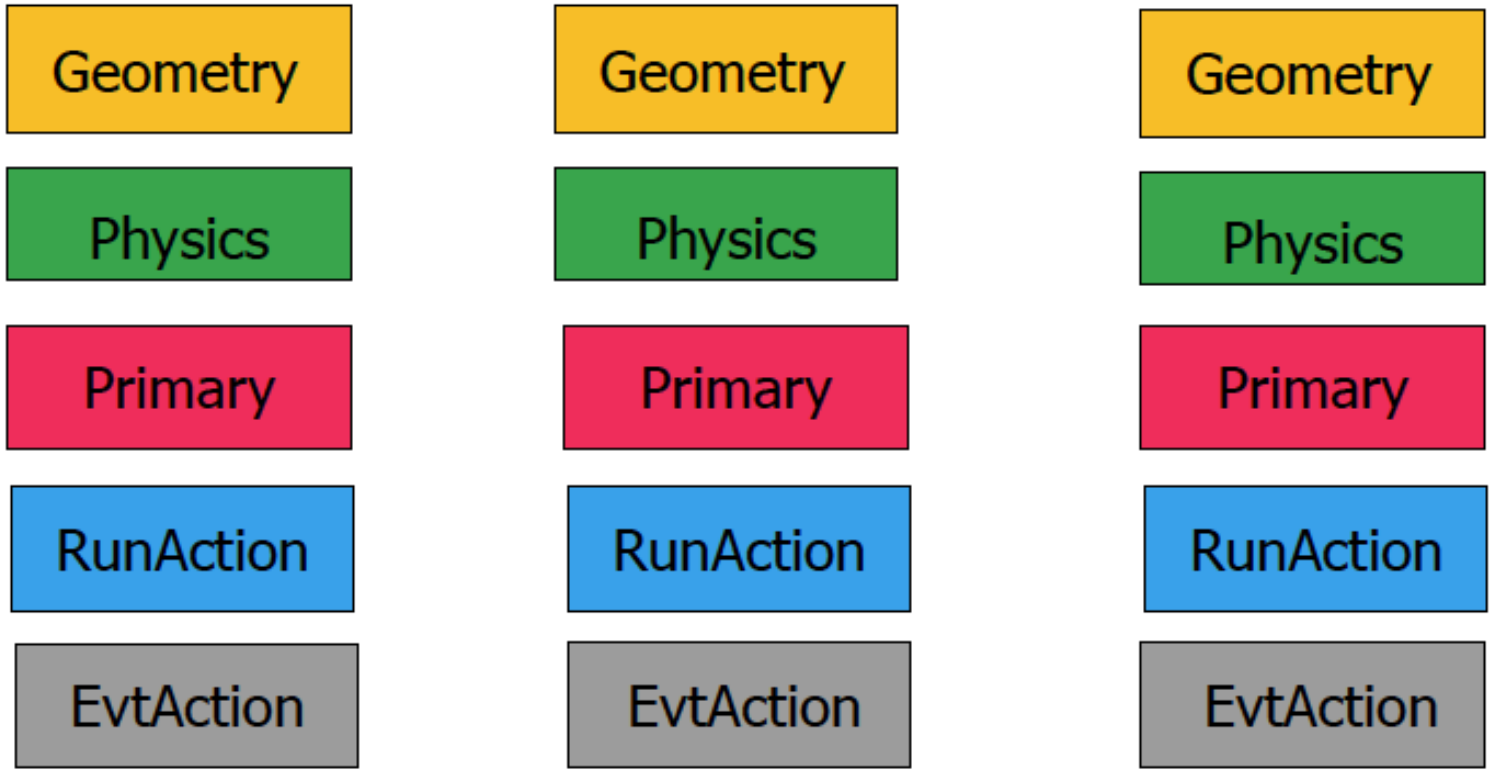
Concept of multi-thread…



… vs parallelization

Nodes



- Each node hosts a complete simulation
- Many copies of geometry and physics tables
- More memory-thirsty

Interaction with the Geant4 kernel

- Geant4 design provides **tools** for a **user application**
  - To tell the kernel about your simulation configuration
  - To interact with Geant4 kernel itself
- Geant4 tools for user interaction are **base classes**
  - You create **your own class** derived from base classes ⇨ interface to the Geant4 kernel
  - Geant4 kernel handles your own derived classes transparently through their base class interface (polymorphism)

Two types of Geant4 base classes:

- Abstract base classes for user interaction (classes starting with **G4V**)
  - User derived concrete classes are mandatory
  - User to implement the purely virtual methods
- Concrete base classes (with virtual dummy default methods) for user interaction
  - User derived classes are optional

User classes

Initialization classes

[Invoked at the initialization]

- G4VUserDetectorConstruction
- G4VUserPhysicsList

Global:

- **only one instance** of them exists in memory, shared by all threads (**readonly**)
- Managed only by the **master** thread

Action classes

[Invoked during the execution loop]

- G4VUserActionInitialization
  - G4VUserPrimaryGeneratorAction
  - G4UserRunAction (*)
  - G4UserEventAction
  - G4UserTrackingAction
  - G4UserStackingAction
  - G4UserSteppingAction

Local:

- **an instance** of each action class exists **for each thread**

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

Mandatory user classes

Mandatory classes
in ANY
Geant4 User Application

- G4VUserDetectorConstruction
  - describe the experimental set-up
- G4VUserPhysicsList
  - select the physics you want to activate
- G4VUserActionInitialization
  - takes care of the user initializations
  - G4VUserPrimaryGeneratorAction

Optional user classes

- Five base classes whose virtual member functions the user may override to gain control of the simulation at various stages
  - G4User**Run**Action
  - G4User**Event**Action
  - G4User**Tracking**Action
  - G4User**Stacking**Action
  - G4User**Stepping**Action

e.g. actions to be done at the beginning and end of each event

- Each member function of the base classes has a dummy implementation (not purely virtual)
  - Empty implementation: does nothing
  - Override only the methods that you need
- User action classes must be **registered** to the Run Manager via the G4VUserActionInizialization

MANDATORY USER CLASSES

The geometry

- User class which describes the geometry must inherit from G4VUserDetectorConstruction and registered in the Run Manager
- Virtual base class: the purely virtual method must be implemented
  - **G4VPhysicalVolume*** Construct() = 0;
    - Must return the pointer to the world volume: all other volumes are contained in it
- Optionally, implement the virtual method
  - **void** ConstructSDandField();
    - Defines sensitive volumes and EM fields

Select physics processes

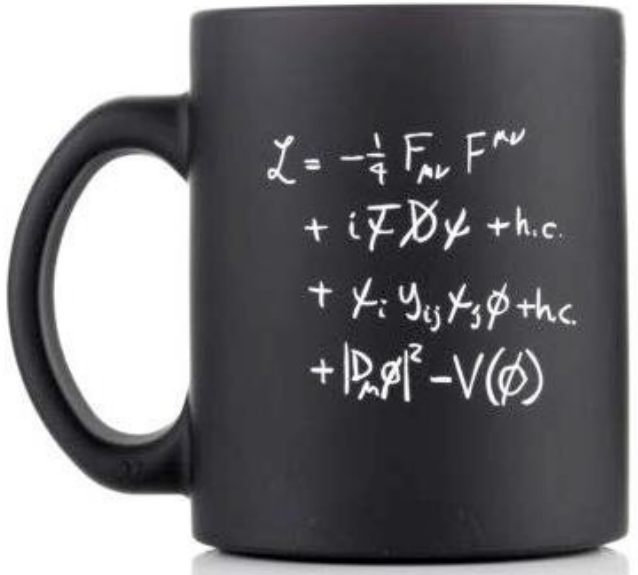
- Geant4 doesn't have any default particles or processes
- Derive your own concrete class from the **G4VUserPhysicsList** abstract base class
  - define all necessary particles
  - define all necessary processes and assign them to proper particles
  - define production thresholds (in terms of range)
- Pure virtual methods of **G4VUserPhysicsList**

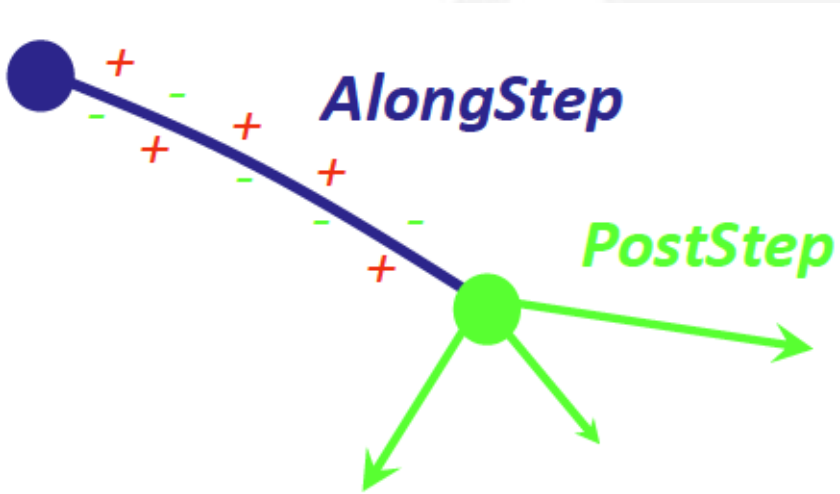
ConstructParticles()
ConstructProcesses()
SetCuts()

must be implemented by the user
in his/her concrete derived class

Physics processes – an overview

- All physics processes derive from an abstract class
- It defines the common interface of all processes in Geant4
- Three kind of “actions”:
  - **AlongStep:** all the soft interactions
  - **PostStep:** all the hard interactions
  - **AtRest:** decays, e^+e^- annihilation





γ model inventory

- Many models available for each process
- Differ for energy range, precision and CPU speed
- Final state generators

Model	E_{min}	E_{max}	CPU
G4LivermoreRayleighModel	100 eV	10 PeV	1.2
G4PenelopeRayleighModel	100 eV	10 GeV	0.9
G4KleinNishinaCompton	100 eV	10 TeV	1.4
G4KleinNishinaModel	100 eV	10 TeV	1.9
G4LivermoreComptonModel	100 eV	10 TeV	2.8
G4PenelopeComptonModel	10 keV	10 GeV	3.6
G4LowEPComptonModel	100 eV	20 MeV	3.9
G4BetheHeitlerModel	1.02 MeV	100 GeV	2.0
G4PairProductionRelModel	10 MeV	10 PeV	1.9
G4LivermoreGammaConversionModel	1.02 MeV	100 GeV	2.1
G4PenelopeGammaConversionModel	1.02 MeV	10 GeV	2.2
G4PEEFluoModel	1 keV	10 PeV	1
G4LivermorePhotoElectricModel	10 eV	10 PeV	1.1
G4PenelopePhotoElectricModel	10 eV	10 GeV	2.9

Electromagnetic models

- The same physics processes can be described by different models
- For instance: **Compton scattering** can be described by
  - **G4KleinNishinaCompton**
  - **G4LivermoreComptonModel** (low-energy, based on the Livermore database)
  - **G4PenelopeComptonModel** (low-energy, based on the Penelope analytical model)
  - **G4LivermorePolarizedComptonModel** (low-energy, Livermore database with polarization)
  - **G4PolarizedComptonModel** (Klein-Nishina with polarization)
  - **G4LowEPComptonModel** (full relativistic 3D simulation)
- Different models can be combined, so that the appropriate one is used in each given energy range ("*à performance*" optimization)

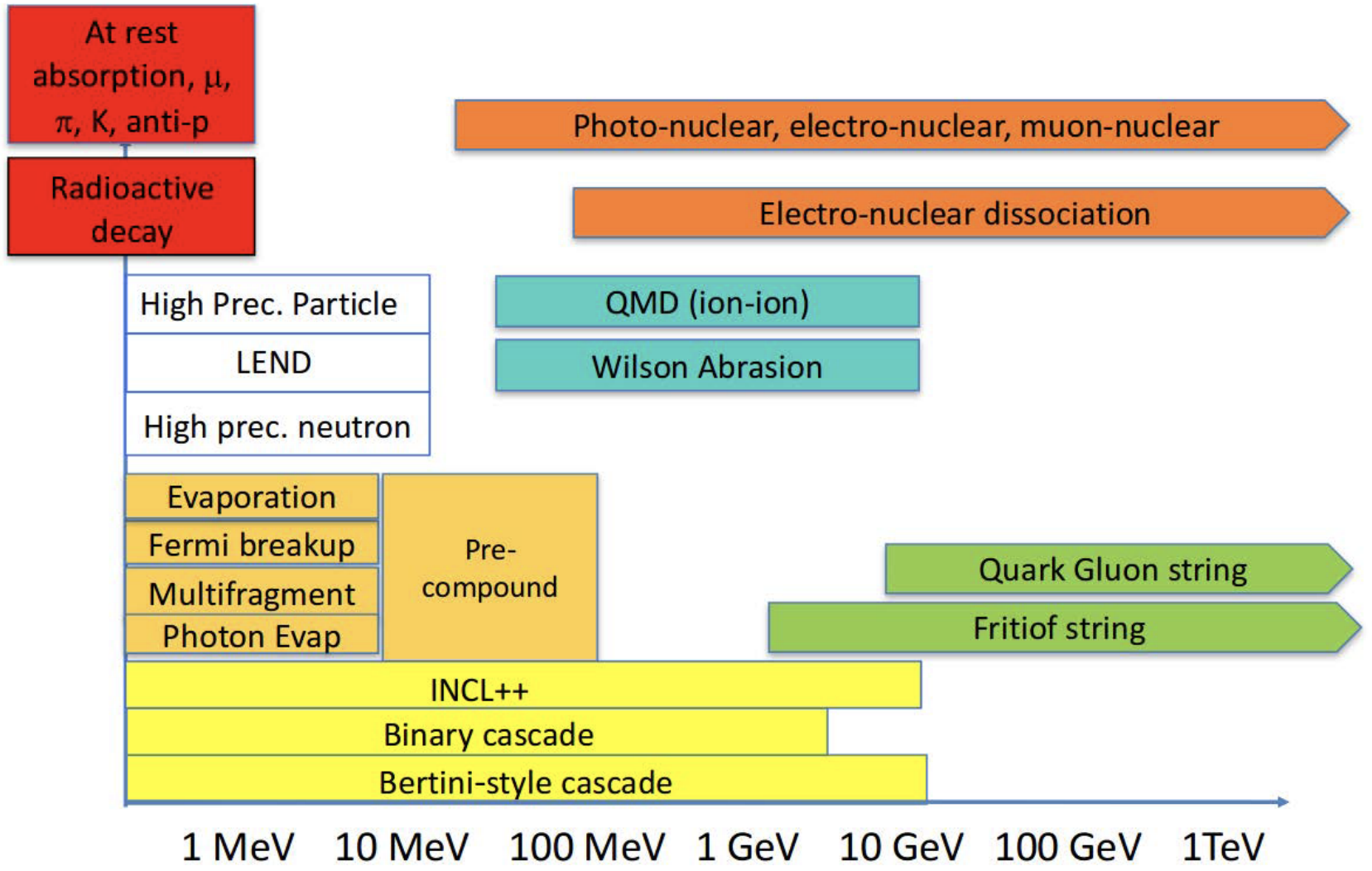
Hadronic processes

- **At rest**
  - Stopped muon, pion, kaon, anti-proton
  - Radioactive decay
  - Particle decay (decay-in-flight is PostStep)
- **Elastic**
  - Same process to handle all long-lived hadrons (multiple models available)
- **Inelastic**
  - Different processes for each hadron (possibly with multiple models vs. energy)
  - Photo-nuclear, electro-nuclear, mu-nuclear
- **Capture**
  - Pion- and kaon- in flight, neutron
- **Fission**

Challenges with hadronic processes

- **Three energy regimes**
  - < 100 MeV
  - resonance and cascade region (100 MeV - 10 GeV)
  - > 20 GeV (QCD strings)
- **Within each regime** there are **several models,** many are phenomenological models
- **Two families** of builders for the **high-energy part**
  - **QGS**, or list based on a model that use the Quark Gluon String model for high energy hadronic interactions of protons, neutrons, pions and kaons
  - **FTF**, based on the FTF (FRITIOF like string model) for protons, neutrons, pions and kaons
- Three families for the **cascade energy range**
  - **BIC**, binary cascade
  - **BERT**, Bertini cascade
  - **INCLXX**, Liege Intranuclear cascade model

Hadronic model inventory



Physics lists

- A **"physics list"** is a class which collects all the particles, physics processes and production thresholds needed for your application
- It tells the run manager how and when to invoke physics
- It is a very flexible way to build a physics environment
  - user can pick the particles he wants
  - user can pick the physics to assign to each particle
- Nevertheless, user must have a good understanding of the physics required
- Omission of particles or physics could cause errors or poor simulation
- Geant4 provides several packaged "production physics lists" which are routinely validated and updated with each release
  - These lists should be considered only as starting points which you may need to validate or modify for your application

Initialization & Primary Generator

- User class must inherit from `G4VUserActionInitialization` and registered in the Run Manager
- User class must inherit from `G4VUserPrimaryGeneratorAction`
  - Registered to the Run Manager via the ActionInizialitation
- Implement the purely virtual method
  - `void GeneratePrimaries(G4Event*)=0;`
  - Called by the RunManager during the event loop, to generate the primary vertices/particles
- Internally uses a concrete instance of `G4VPrimaryGenerator` (for example `G4ParticleGun`) to do the job

THE MAIN() PROGRAM

Main program

- Geant4 does **not** provide the **`main()`**
  - Geant4 is a toolkit!
  - The `main()` is part of the user application
- In his/her `main()`, the user must:
  - construct `G4RunManager`
  - notify the `G4RunManager` mandatory user classes derived from:
    - `G4VUserDetectorConstruction`
    - `G4VUserPhysicsList`
    - `G4VUserActionInitialization` (takes care of Primary)
  - The `G4RunManagerFactory` will pick the Sequential/MT version of the `G4RunManager`

Main program

- The user may define in his/her `main()`
  - optional user action classes
  - VisManager, (G)UI session
- The user also must take care of retrieving and saving the relevant information from the simulation
  - Geant4 will not do that by default
- Do not forget to delete the `G4RunManager` at the end

Let's see an example of `main()`

Example of `main()`

```
{
…
// Create the run manager (let the RunManagerFactory decide if MT,
sequential or other).
//The flags from G4RunManagerType: Serial, MT
auto* runManager =
G4RunManagerFactory::CreateRunManager(G4RunManagerType::Serial);

// Set mandatory user initialization classes
MyDetectorConstruction* detector = new MyDetectorConstruction;
runManager->SetUserInitialization(detector);

MyPhysicsList* physicsList = new MyPhysicsList;
runManager->SetUserInitialization(myPhysicsList);

// Set mandatory user action classes
runManager->SetUserAction(new MyActionInitialization);
…
}
```

Documentation

http://geant4.org

- A few manuals available in the Geant4 webpage
  - Application developer manual
  - Physics manual
- Other tools available
  - LXR code repository
  - User forum
  - Bugzilla
  - GitHub code repo

https://github.com/Geant4

Examples

- Ready-for-the-use Geant4 applications (*examples*) are distributed with Geant4
  - Very good starting point for new users
- Three suites of examples:
  - **"basic":** oriented to novice users and covering the most typical use-cases of a Geant4 application with keeping simplicity and ease of use
  - **"extended":** covers many specific use cases for actual detector simulation
  - **"advanced":** where real-life complete applications for different simulation studies are provided
- A webpage with doxygen documentation is available for the basic/extended examples

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

Main Page | Related Pages | Topics | Namespaces | Classes | Files

The set of basic examples is oriented to "novice" users and covering many basic general use-cases typical of an "application"-oriented kind of development.

- **Example B1**
  - Simple geometry with a few solids
  - Geometry with simple placements (G4PVPlacement)
  - Scoring total dose in a selected volume user action classes
  - Using G4Accumulable for automatic merging of scored values in multi-threading mode
  - Geant4 physics list (QBBC)
- **Example B2**
  - Simplified tracker geometry with global constant magnetic field
  - Geometry with simple placements (G4PVPlacement) and parameterisation (G4PVParameterisation)
  - Scoring within tracker via G4 sensitive detector and hits
  - Geant4 physics list (FTFP_BERT) with step limiter
  - Started from novice/N02 example
- **Example B3**
  - Schematic Positron Emitted Tomography system
  - Geometry with simple placements with rotation (G4PVPlacement)
  - Radioactive source
  - Scoring within Crystals via G4 scorers
  - Using G4Accumulable for automatic merging of scored values in multi-threading mode (a) and G4StatAnalysis for accumulating statistics (b)
  - Modular physics list built via builders provided in Geant4
- **Example B4**
  - Simplified calorimeter with layers of two materials
  - Geometry with replica (G4PVReplica)
  - Scoring within layers in four ways: via user actions (a), via user own object (b), via G4 sensitive detector and hits (c) and via scorers (d)
  - Geant4 physics list (FTFP_BERT)
  - Histograms (1D) and ntuple saved in the output file
  - Started from novice/N03 example
- **Example B5**
  - A double-arm spectrometer with wire chambers, hodoscopes and calorimeters with a local constant magnetic field

Basic Examples

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