
# Introduction

Mihaly Novak (CERN, EP-SFT)

Getting Started with **`Geant4`** at CERN, Geneva (Switzerland)

- What is **`Geant4`**?
- What is our goal during this course?
- Documentation and installation
- The main: user initialisation and mandatory actions
- Our step-by-step plan

Introduction

# WHAT IS GEANT4?

- **`Geant4` is a toolkit:**
  - for simulating the passage of particles through matter
  - toolkit i.e. there here is no main program
  - provides all the necessary **components** needed **to describe and to solve** particle transport **simulation** problems
  - **problem definitions/description:** geometry, particles, physics, etc.
  - **problem solution**: step-by-step particle transport computation
  - while providing **interaction point**s for the user
- **Toolkit vs program?**
  - as a toolkit, **`Geant4`** does **not** provide **a** main **program**
  - each simulation problem requires different configuration (geometry, scoring, particles, physics, etc..) that the user needs to define
  - **`Geant4`**, as a **toolkit**, provides all the necessary **components** in form of **interfaces** (called actions in **`Geant4`** terminology but see soon)

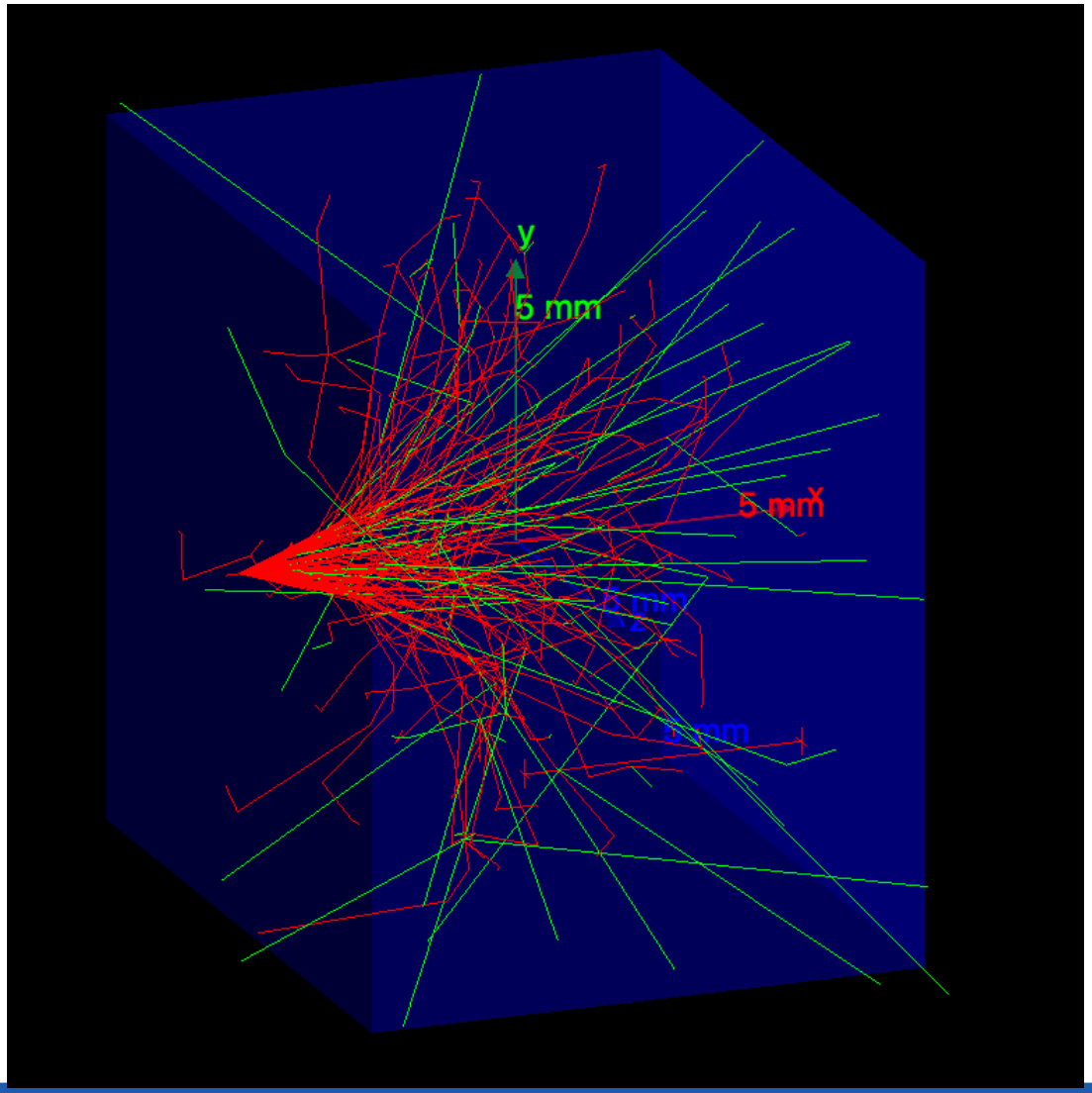
# WHAT IS OUR GOAL DURING THIS COURSE?

# What is our goal during this course?

- **Goal:**
  - introduce all the **mandatory** and some of the important **optional components** that needs to/can be utilised when writing a **simulation application** based on the **`Geant4`** simulation toolkit
  - show source of information that can be useful when writing such an application (documentation, **`Geant4`** source code, etc..)
  - become familiar with the *best practice* when developing your own application
- **How?**
  - we will write a **`Geant4`** simulation application together from scratch
  - the application:
    - **setup**: a simple box target with configurable thickness and material hit by a configurable particle source (see the next slide)
    - **goal**: collect information regarding the energy deposit in the target
  - we will do **mainly coding instead of lectures** with short explanations
  - more and more functionality will be added when time goes
  - both the agenda and the final state of our application is flexible

# What is our goal during this course?

## 4 [MeV] electrons in Silicon (1 [cm])



Introduction

# DOCUMENTATION

- **Documentation:**
  - all documentation can be found at the Geant4 webpage under the **User Support** menu
  - the **Book For Application Developers** will be our main source of documentation in the next 3 days
  - it is important that all of **you have a working version of** the **`Geant4`** toolkit available on your machine (the VM is the preferred one) before we go any further:
    - we will have a look to the **Installation Guide** now together
    - build/try one of the example applications that Geant4 provides (`/examples/basic/B1`)
    - inspect the installation directory structure (and understand the role of `-DGeant4_DIR` **`Geant4`** `cmake` variable with a simple example)
    - let's do it by ourself… :-)

Introduction

# THE MAIN: USER INITIALISATIONS AND MANDATORY ACTIONS

# The main: user initialisations and mandatory actions

- As mentioned before, **`Geant4`** do not provide a main program:
  - there are components of a simulation, like the **geometry**, **physics** settings and the **primary** particle **generation** that are changing from problem to problem
  - therefore, the **user needs to provide these** settings **in the `main` method** of their **`Geant4`** application

# The main: user initialisations and mandatory actions (`G4RunManager`)

- As mentioned before, **`Geant4`** do not provide a main program:
  - there are components of a simulation, like the **geometry**, **physics** settings and the **primary** particle **generation** that are changing from problem to problem
  - therefore, the **user needs to provide these** settings **in the `main` method** of their **`Geant4`** application

1. Create a **`G4RunManager`** object **(mandatory)**:
   - the **only** mandatory **manager object** that user the needs to create: all others (**`G4EventManager`**, **`G4SteppingManger`**, etc.) are created and deleted automatically
   - the **`G4RunManager`** is responsible to **control the** flow of a **run**, the top level simulation unit
   - this includes **initialisation of the run** i.e. building, **setting up the** simulation **environment**
   - all problem specific information need to be given to the **`G4RunManager`** by the user through the interfaces provided by the **`Geant4`** toolkit (we will see them one by one):
     - **`G4VUserDetectorConstruction`**(mandatory): how the geometry should be constructed, built
     - **`G4VUserPhyscsList`**(mandatory): all the particles and their physics interactions to be simulated
     - **`G4VUserActionInitialization`** (mandatory):
       - ✳ **`G4VUserPrimaryGeneratorAction`** (mandatory): how the primary particle(s) in an event should be produced
       - ✳ additional, **optional user actions** (**`G4UserRunAction`**, **`G4UserEventAction`**, **`G4UserSteppingAction`**, etc..)

- As mentioned before, **`Geant4`** do not provide a main program:
  - there are components of a simulation, like the **geometry**, **physics** settings and the **primary** particle **generation** that are changing from problem to problem
  - therefore, the **user needs to provide these** settings **in the `main` method** of their **`Geant4`** application

1. Create a **`G4RunManager`** object **(mandatory)**:
   - the **only** mandatory **manager object** that user needs to create: all others (**`G4EventManager`**,

**See more when we write the `main` method of our own application!**

should be produced

✳ additional, **optional user actions** (**`G4UserRunAction`**, **`G4UserEventAction`**, **`G4UserSteppingAction`**, etc..)

- **MT note:** **`G4MTRunManager`** object needs to be created in case of **`Geant4 MT`**

2. Create **YourDetectorConstruction** object and register it in your **G4RunManager** object **(mandatory)**:

- the **G4VUserDetectorConstruction** interface is provided by the **Geant4** toolkit to **describe** the **geometrical setup**, including all volumes with their shape, position and **material definition**
- its **G4VUserDetectorConstruction::Construct()** interface method (pure virtual) is invoked by the **G4RunManager** at initialisation
- **derive your own detector description**, e.g. **YourDetectorConstruction** class from this base class and implement the **Construct()** interface method:
  - create all materials will need to use in your geometry
  - describe your detector geometry by creating and positioning all volumes
  - return the pointer to the root of your geometry hierarchy i.e. the pointer to your "World" **G4VPhysicalVolume**
- create **YourDetectorConstruction** object and register it in your **G4RunManager** object by using the `G4RunManager:SetUserinitialization` method (see this in the source!)
- **MT note:** the **Construct()** interface method is **invoked only by the Master Thread** in case of **Geant4 MT** (i.e. only one detector object), while the other `ConstructSDandField()` interface method is **invoked by each Worker Threads** (i.e. thread local objects created)

2. Create `YourDetectorConstruction` object and register it in your `G4RunManager` object **(mandatory)**:
   - the `G4VUserDetectorConstruction` interface is provided by the **`Geant4`** toolkit to **describe** the **geometrical setup**, including all volumes with their shape, position and

**See more at the Detector Construction lecture!**

**We will write together the `DetectorConstruction`.**

   - **MT note:** the `Construct()` interface method is **invoked only by the Master Thread** in case of **`Geant4 MT`** (i.e. only one detector object), while the other `ConstructSDandField()` interface method is **invoked by each Worker Threads** (i.e. thread local objects created)

3. Create YourPhysicsList object and register it in your G4RunManager object **(mandatory)**:

- the G4VUserPhysicsList interface is provided by the **Geant4** toolkit to **describe** the **physics setup**, including **definition of** all **particles** and their physics interactions, **processes**
- its G4VUserPhysicsList::ConstructParticle() and ::ConstructProcess() interface methods (pure virtual) are invoked by the **G4RunManager** (actually by the **G4RunManagerKernel** and process construction is invoked indirectly) at initialisation
- **derive your own physics list**, e.g. YourPhysicsList class from this base class and implement the ConstructParticle() and ConstructProcess() interface methods:
  - create all particles in the ConstructParticle() method
  - create all processes and sign them to particles in the ConstructProcess() method
- create YourPhysicsList object and register it in your G4RunManager object by using the `G4RunManager:SetUserinitialization` method (see this in the source!)
- constructing physics list as described above is **recommended only for advanced users**!
- **Geant4** provides possibilities with different level of granularity to build up or obtain even complete pre-defined physics list

3. Create `YourPhysicsList` object and register it in your `G4RunManager` object **(mandatory)**:
   - the `G4VUserPhysicsList` interface is provided by the `Geant4` toolkit to **describe** the **physics setup**, including **definition of** all **particles** and their physics interactions, **processes**

**See more at the Physics List lecture!**

**We will use one of the pre-defined physics list.**

   - `Geant4` provides possibilities with different level of granularity to build up or obtain even complete pre-defined physics list

4. Create **YourPrimaryGeneratorAction** object **(mandatory**, see next slide how to register**)**:

- the **G4VUserPrimaryGeneratorAction** interface is provided by the **Geant4** toolkit to describe how the primary particle(s) in an event should be produced
- its **G4VUserPrimaryGeneratorAction::GeneratePrimaries()** interface method (pure virtual) is invoked by the **G4RunManager** during the event-loop (in its `G4RunManager::GenerateEvent()` method)
- **derive your own primary generator action,** e.g. **YourPrimaryGeneratorAction** class from this base class and implement the **GeneratePrimaries()** interface method:
  - describe how the primary particle(s) in an event should be produced
  - we will use a **G4ParticleGun** object, provided by the **Geant4** toolkit, to generate primary particles: one particle per event with defined kinematics
- note:
  - the **Detector-Construction** and the **Physics-List** need to be **created directly in the main** program and **registered directly in the G4RunManager** object
  - all **User-Actions** needs to be **created** and **registered** in the **User-Action-Initialisation** (**including the** only **mandatory Primary-Generator-Action** as well as **all** other, **optional User-Actions**)
  - see more on this and on the **G4VUserActionInitialization** in the next slide

4. Create `YourPrimaryGeneratorAction` object **(mandatory**, see next slide how to register**)**:
   - the `G4VUserPrimaryGeneratorAction` interface is provided by the **`Geant4`** toolkit to describe how the primary particle(s) in an event should be produced
   - its `G4VUserPrimaryGeneratorAction::GeneratePrimaries()` interface method (pure virtual) is invoked by the **`G4RunManager`** during the event-loop (in its `G4RunManager::GenerateEvent()` method)

**We will write together the `PrimaryGeneratorAction`.**

   - note:
     - the **Detector-Construction** and the **Physics-List** need to be **created directly in the main** program and **registered directly in the `G4RunManager`** object
     - all **User-Actions** needs to be **created** and **registered** in the **User-Action-Initialisation** (**including the** only **mandatory Primary-Generator-Action** as well as **all** other, **optional User-Actions**)
     - see more on this and on the `G4VUserActionInitialization` in the next slide

5. Create **YourActionInitialization** object and register it in your **G4RunManager** object **(mandatory)**:

- the **G4VUserActionInitialization** interface is provided by the **Geant4** toolkit to **create** and **register:**
  - **the only one mandatory G4VUserPrimaryGeneratorAction user action**
  - all other **optional user actions** (**G4UserRunAction**, **G4UserEventAction**, etc..)
- its **G4VUserActionInitialization::Build()** interface method (pure virtual) is invoked by the **G4RunManager** at initialisation
- **derive your own action initialisation**, e.g. **YourActionInitialization** class from this base class and implement the **Build()** interface methods:
  - **create** an object from **YourPrimaryGeneratorAction** (see next slide) and **register** by calling the corresponding `G4VUserActionInitialization::SetUserAction()` base class method
  - **create all** additional, **optional user action** objects and **register them** similarly
- **MT note:** we will get back to this before adding optional user actions (see OptionalUserActionsAndMT)

5. Create `YourActionInitialization` object and register it in your `G4RunManager` object **(mandatory)**:

- the `G4VUserActionInitialization` interface is provided by the **`Geant4`** toolkit to **create** and **register:**
  - **the only one mandatory `G4VUserPrimaryGeneratorAction` user action**
  - all other **optional user actions** (`G4UserRunAction`, `G4UserEventAction`, etc..)
- its `G4VUserActionInitialization::Build()` interface method (pure virtual) is invoked by the `G4RunManager` at initialisation
- **derive your own action initialisation**, e.g. `YourActionInitialization` class from this base class and implement the `Build()` interface methods:
  - **create** an object from `YourPrimaryGeneratorAction` (see next slide) and

# We will write together the `ActionInitializaton.`

# OUR STEP BY STEP PLAN

I. **Intermediate application:** the bare minimum that is needed to run the simulation:

- the bare minimum that is needed to run the simulation (execute, without collecting data!)
- implement all the mandatory components mentioned before:
  - **`YourDetectorConstruction`**:
    - ✳a simple **box** (shape) as the detector/**target** filled with with **silicon** as material
    - ✳placed in a **box** (shape) "**world**" volume filled with **low density hydrogen gas**
  - **`YourPhysicsList`: we will use** one of the **pre-defined**, ready-to-use **physics list provided by** the **`Geant4`** toolkit (therefore no need to write any user physics list class like in our case)
  - **`YourPrimaryGeneratorAction`**:
    - ✳a simple **particle gun (`G4ParticleGun`):** generates a **single primary** particle **per event** with pre-**defined** particle **type** and **kinematics** pointing toward to our target
  - **`YourActionInitialization`:**
    - ✳implement the construction and registration of our **`YourPrimaryGeneratorAction`** object
- develop the `main` method of the application and execute the simulation
- inspect the results

II. **Intermediate application:** more control over the execution

- more control over the execution, but still no simulation data collected
- add **functionality to** the `main` method of our application to be able to **run the application**:
  - in **interactive** or **batch mode**
  - with or without **visualisation**
  - write the corresponding macro files
- become familiar with all possible modes of executions, understand their advantage and find your most comfortable one

## III. **Final application:** extension with optional User-Actions

- extension needed to collect some data during the simulation
- add **optional User Actions** (run-, event-, stepping-actions):
  - simulate mean value and sigma of the energy deposited in the detector/target per event
- become conformable with all of these optional user actions that provide access to the simulation workflow, data, states to the application developer (and eventually to the user)

IV. **Final application:** add more flexibility regarding the configuration

- define and add **User Interface (UI) commands** to the detector construction to be able to **configure the target material** and **thickness**
- become familiar with developing your own UI commands to your **`Geant4`** application that can increase significantly the flexibility of your application