

ISAPP School 2024 · KIT / Bad Liebenzell

# Geant4 Simulations for Rare Event Searches

Holger Kluck · HEPHY · [holger.kluck@oeaw.ac.at](mailto:holger.kluck@oeaw.ac.at)

1/3: Theory of MC Simulations, Programming Environment,  
Basics of Geant4

# Scope

- Q: What will you gain from this lecture?
- A: You will get a basic understanding how to implement and run a Monte Carlo (MC) simulation with the Geant4 code. You should get an understanding what MC background models of rare event searches can do and what their limitations are.

# Schedule

- Today:  
**Theory** of MC simulations, **programming** environment, and basics of **Geant4**
- Friday (20 Sep., 14:00 – 15:45):  
Implement experimental **geometries**, generate **primary particles**, store **data** and analyse it with ROOT
- Wednesday (25 Sep., 16:15 – 18:00):  
Simulation of
  - **intrinsic** backgrounds for **deep underground** experiments searching for Dark Matter
  - **atmospheric** backgrounds for reactor based neutrino experiments at **shallow experimental sites**

# Mode of the lecture

- The lecture will alternate between theory parts (~15min, me talking) and hands-on examples (~20min, you simulating)
- During the hands-on you can also discuss the problem at hand with your fellow student
- If you have questions – don't hesitate to ask them at any time!

What is your previous knowledge?

# Theory\* of MC simulations

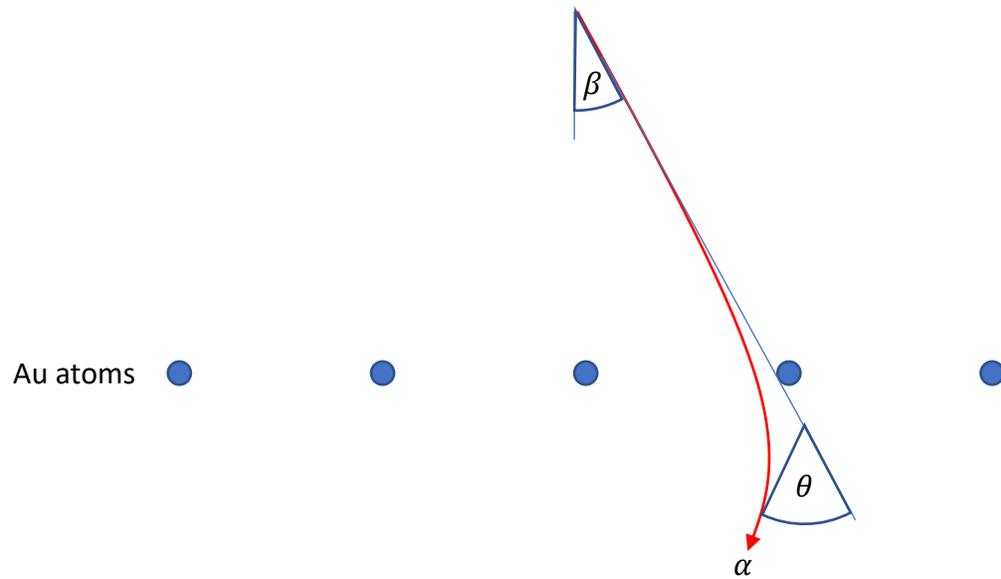
\*To the extent needed to understand an actual simulation and its terminology  
- not more

# Research Objectives

- Treat a MC simulation as a **virtual experiment** and decided before starting it\*:
  - What is the **research objective**? What question should the experiment answer?
  - What is the **observable**? What should be (virtually) measured?
  - What is known about the **boundary conditions** of the experiment? Are positions constrained by the geometry of the apparatus that should be simulated?

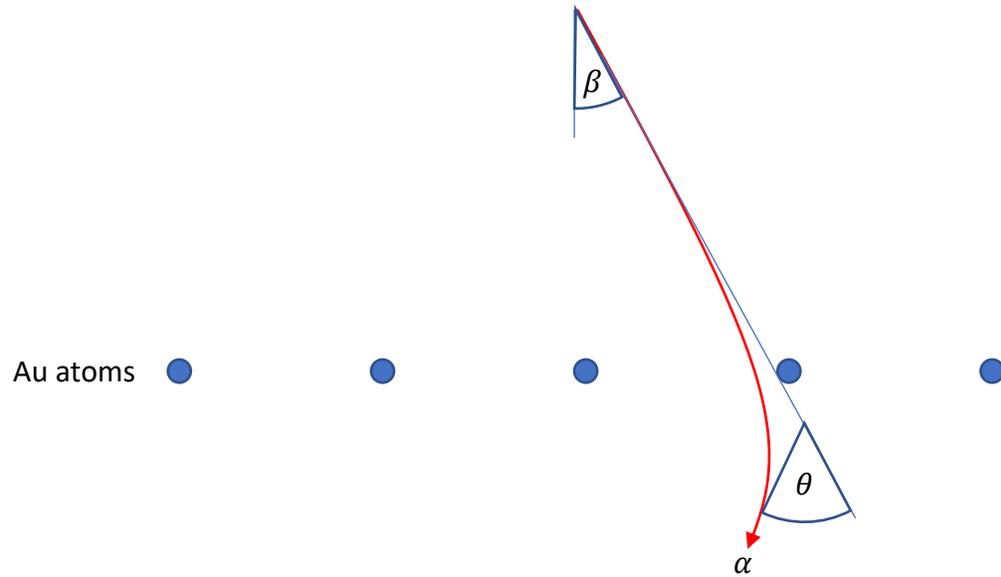
\*At least in a first approximation; as with real experiments, first results may cause refinements or extension of the initial objectives

# Research Objectives



- **Objective:** What is the deflection angle  $\theta$  of an alpha particle of energy  $E$  emitted in direction  $\beta$  at position  $r_0$  after passing through a monoatomic gold layer with atoms at  $r_{\text{Au},i}$ ?

# Input And Output



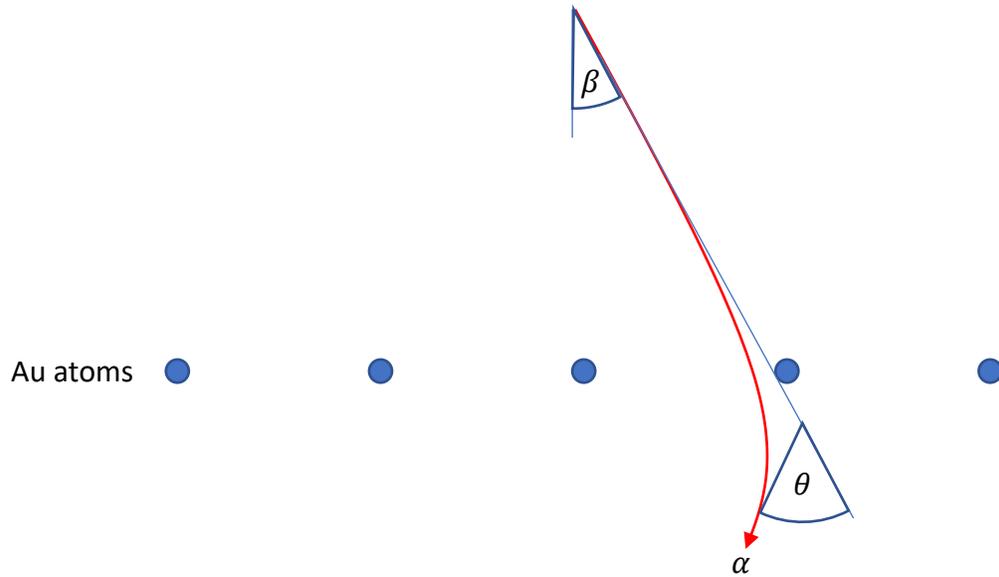
- ***Input:***

- Incident particle: alpha
- Initial energy  $E$
- Initial direction  $\beta$
- Initial position  $r_0$
- Scatterer particle: Au
- Scatterer positions  $r_{\text{Au},i}$

- ***Output:***

- Deflection angle  $\theta$

# Primary Particle, Geometry, Observable



## • **Input:**

- Incident particle: alpha
- Initial energy  $E$
- Initial direction  $\beta$
- Initial position  $r_0$
- Scatterer particle: Au
- Scatterer positions  $r_{\text{Au},i}$

Defines the **primary particle**

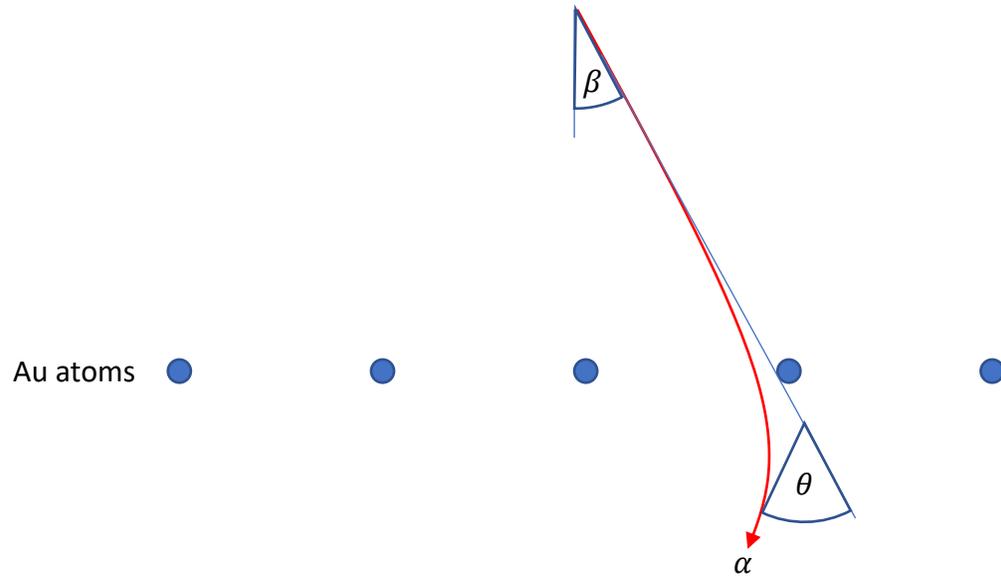
Defines the **geometry** of the experiment

## • **Output:**

- Deflection angle  $\theta$

The **observable** that should be measured

# Physics Model



- **Input:**

- Incident particle: alpha
- Initial energy  $E$
- Initial direction  $\beta$
- Initial position  $r_0$
- Scatterer particle: Au
- Scatterer positions  $r_{\text{Au},i}$

Defines the primary particle

Defines the geometry of the experiment

- **Process: ?**  
(input  $\rightarrow$  output)

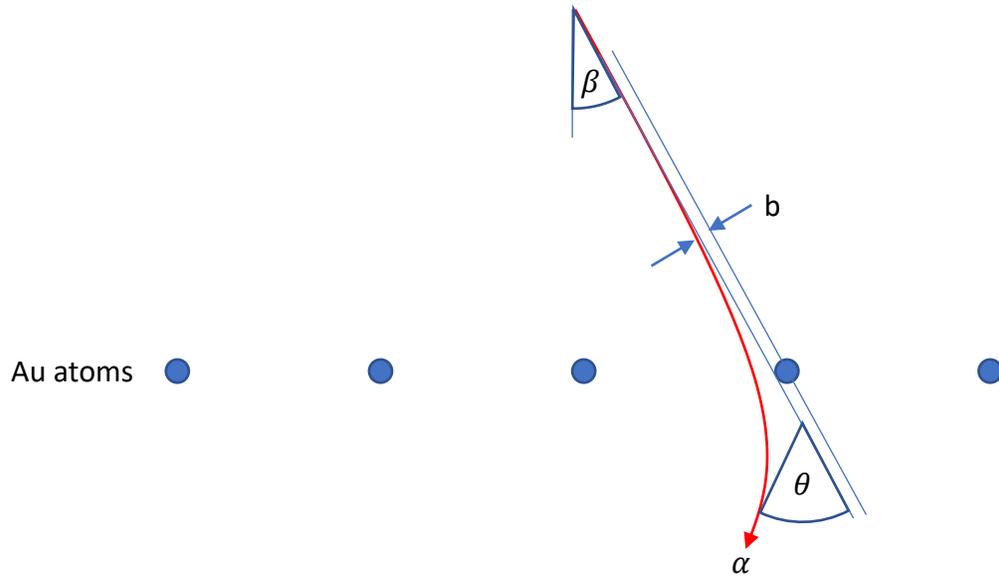
A **model** of the **physical** interactions

- **Output:**

- Deflection angle  $\theta$

The observable that should be measured

# Ideal Rutherford Scattering



- **Process** (input  $\rightarrow$  output):

Rutherford scattering

$$\cot\left(\frac{\theta}{2}\right) = \frac{4\pi\epsilon_0\mu v^2}{Q_{\text{Au}} \cdot Q_{\alpha}} \cdot b$$

with impact parameter

$$b = b(\beta, r_0, r_{\text{Au}}),$$

projectile velocity

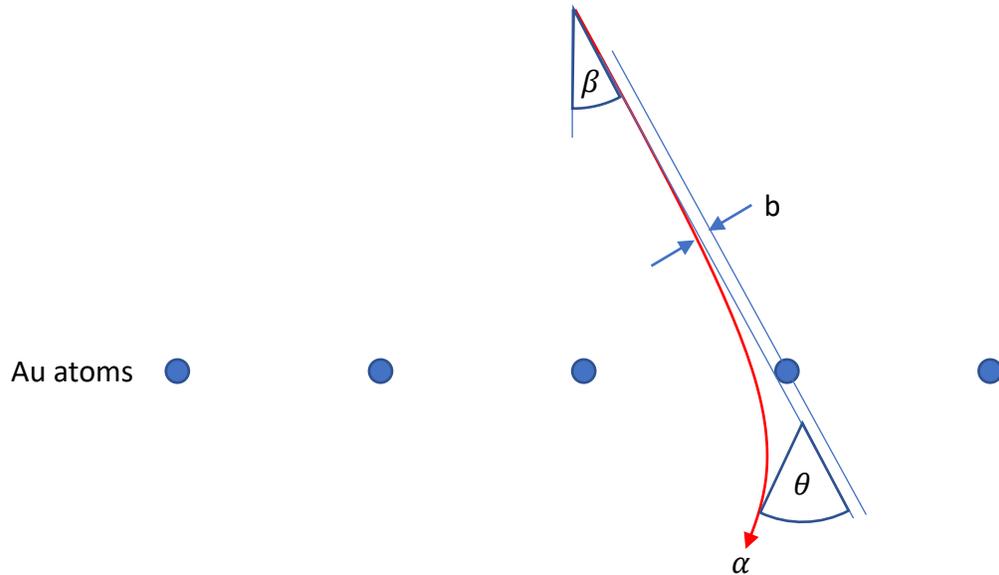
$$v = v(E),$$

and reduced mass

$$\mu = \mu(m_{\text{Au}}, m_{\alpha})$$

computable from inputs

# Ideal Rutherford Scattering

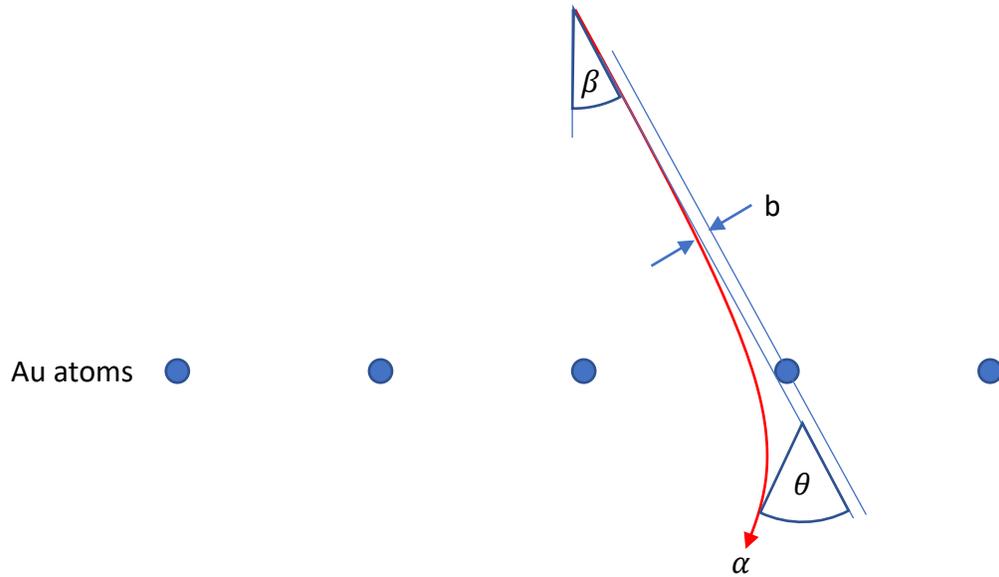


- **Process** (input  $\rightarrow$  output):

$$\theta = \theta(E, \beta, r_0, r_{\text{Au}}),$$

- Rutherford scattering is a classic theory – no randomness is involved
- In an ideal world (perfect preparation of incident particle, perfect knowledge of scatterer), repetitions of the experiment will yield same results

# ~~Ideal~~ Real Rutherford Scattering



- **Process** (input  $\rightarrow$  output):

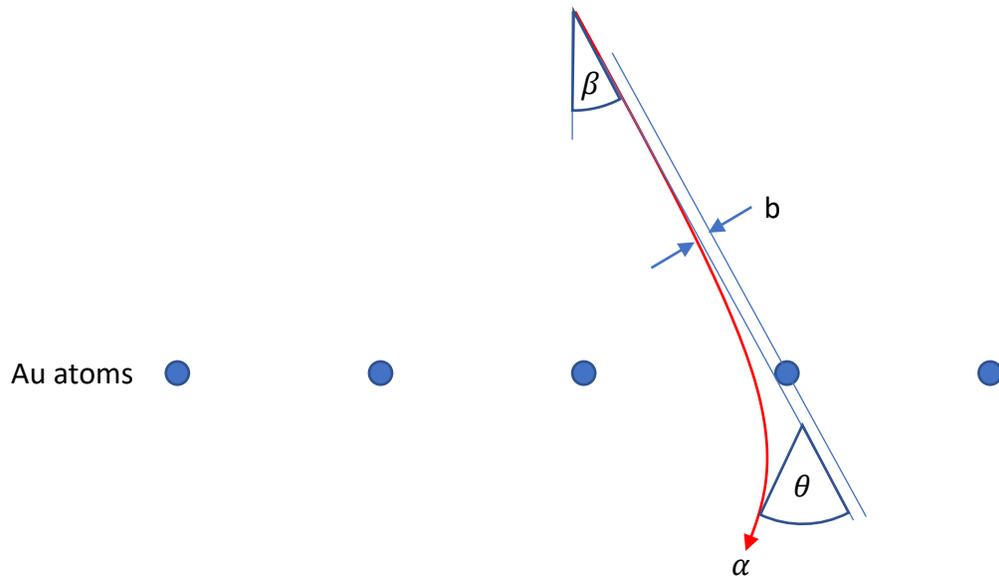
$$\theta = \theta(E, \beta, r_0, r_{\text{Au}}),$$

- In reality, there are **uncertainties**:
  - No particle can be perfectly prepared ( $E, \beta, r_0$ )
  - No perfect knowledge about scatterer ( $b(r_{\text{Au}})$ )

$\rightarrow$  Repetition of the experiment will yield different results

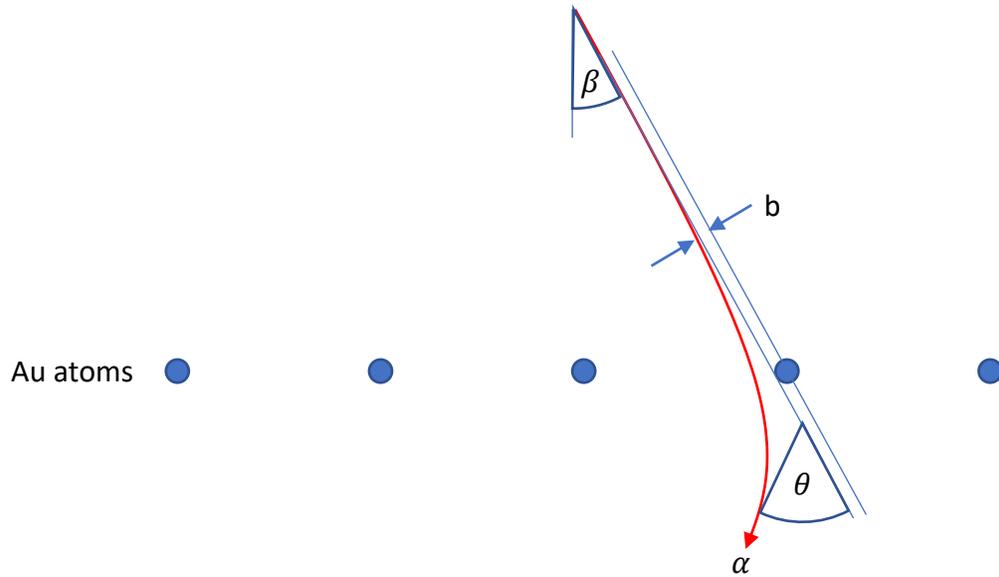
$\rightarrow$  Randomness!

# Randomness



- Assume that the input variables are the components of a **random vector**
$$\vec{X} = (E, \beta, r_0, r_{\text{Au}})$$
with dimension  $n = 4, \dots$
- With all possible realisations  $\vec{x}$  that  $\vec{X}$  can take are given by the **sample space**  $\Omega$ :
$$\vec{x} \in \Omega, \dots$$
- And the **probability** density function (pdf) to realize an actual  $\vec{x}$  is  $P(\vec{x}), \dots$ 
  - In physics, with cross section  $\sigma$ :  $P(\vec{x}) \propto \sigma(\vec{x})$

# Expectation Value

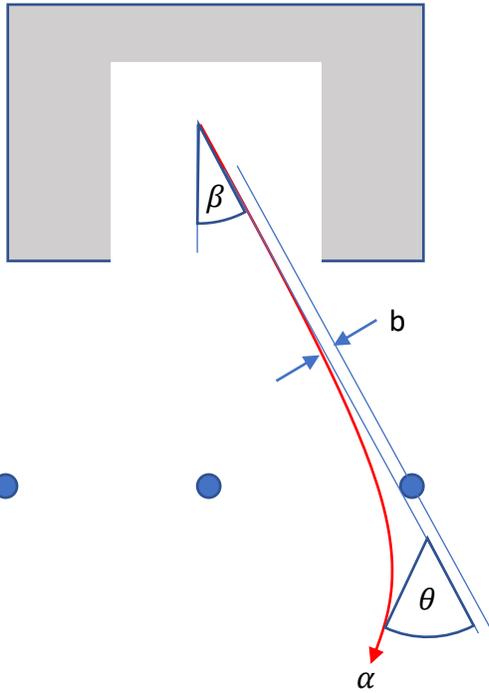


- Then also the output  $\Theta(\vec{X})$  is a random variable with realisation  $\theta$
- And we can use the **expectation value**

$$E[\Theta] = \int_{\vec{x} \in \Omega} P(\vec{x}) \cdot \theta(\vec{x}) d^n x$$

to consider the randomness of the searched for output

# Disadvantages



$$E[\Theta] = \int_{\vec{x} \in \Omega} P(\vec{x}) \cdot \theta(\vec{x}) d^n x$$

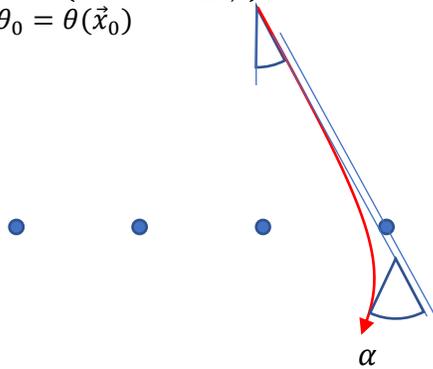
- But it has some potential disadvantages:
  - Already this simple experiment requires a 4-dimensional integration
  - Dimensionality increase rapidly with more realistic modelling of the experiment
  - $\Omega$  (and  $P(\vec{x})$ ) can be very complex, e.g. if  $r_0$  is constrained by some complex source geometry

# Monte Carlo Simulation

$i = 0$

$$\vec{x}_0 = (E_0, \beta_0, r_0, r_{Au,0})$$

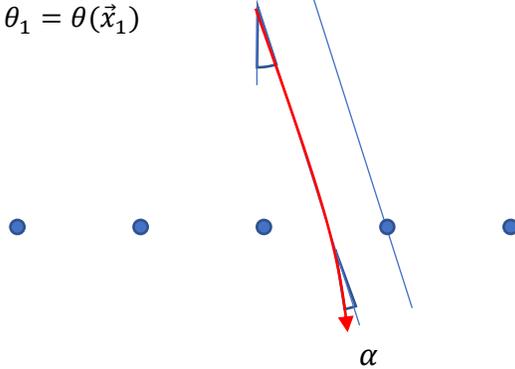
$$\theta_0 = \theta(\vec{x}_0)$$



$i = 1$

$$\vec{x}_1 = (E_0, \beta_1, r_0, r_{Au,0})$$

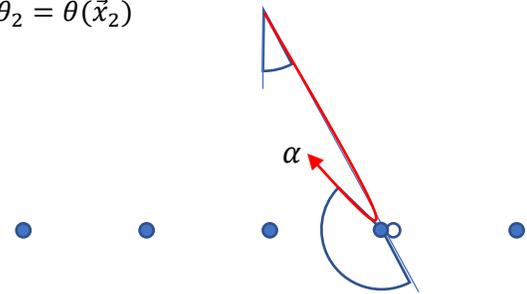
$$\theta_1 = \theta(\vec{x}_1)$$



$i = 2$

$$\vec{x}_2 = (E_0, \beta_0, r_0, r_{Au,2})$$

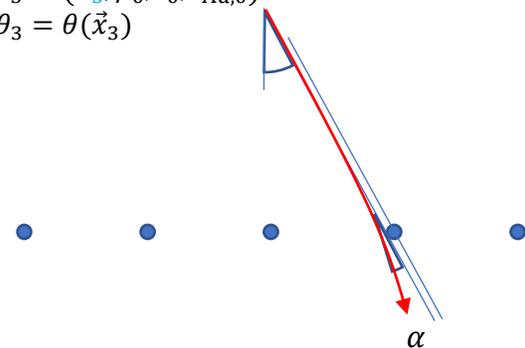
$$\theta_2 = \theta(\vec{x}_2)$$



$i = 3$

$$\vec{x}_3 = (E_3, \beta_0, r_0, r_{Au,0})$$

$$\theta_3 = \theta(\vec{x}_3)$$



**Monte Carlo Simulation:** draw  $N$  samples  $(\vec{x}_i)_{i=0}^N$  from  $\Omega$  and approximate

$$E[\Theta] = \int_{\vec{x} \in \Omega} P(\vec{x}) \cdot \theta(\vec{x}) d^n x$$

with the estimator of the expectation value

$$\hat{E}[\Theta] = \frac{1}{N} \sum_{i=0}^N \theta(\vec{x}_i)$$

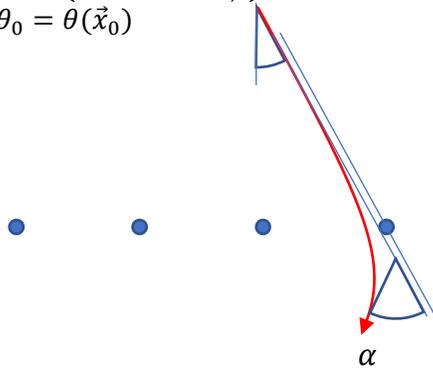
→ Solve the integral via  
**Monte Carlo integration**

# Advantages

$i = 0$

$$\vec{x}_0 = (E_0, \beta_0, r_0, r_{Au,0})$$

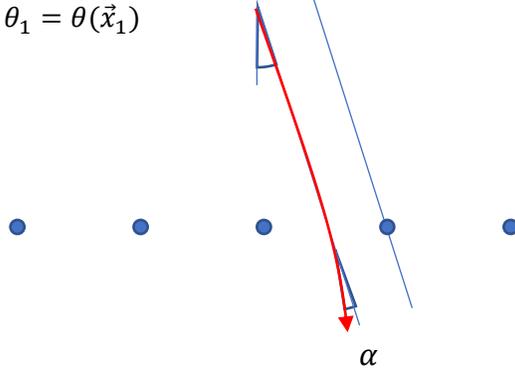
$$\theta_0 = \theta(\vec{x}_0)$$



$i = 1$

$$\vec{x}_1 = (E_0, \beta_1, r_0, r_{Au,0})$$

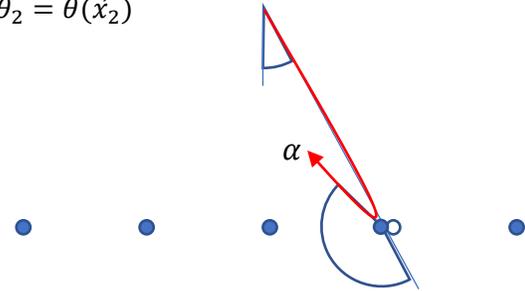
$$\theta_1 = \theta(\vec{x}_1)$$



$i = 2$

$$\vec{x}_2 = (E_0, \beta_0, r_0, r_{Au,2})$$

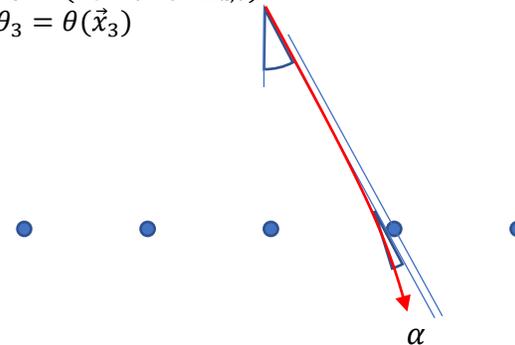
$$\theta_2 = \theta(\vec{x}_2)$$



$i = 3$

$$\vec{x}_3 = (E_3, \beta_0, r_0, r_{Au,0})$$

$$\theta_3 = \theta(\vec{x}_3)$$



- Due to the **Law of Large Numbers**

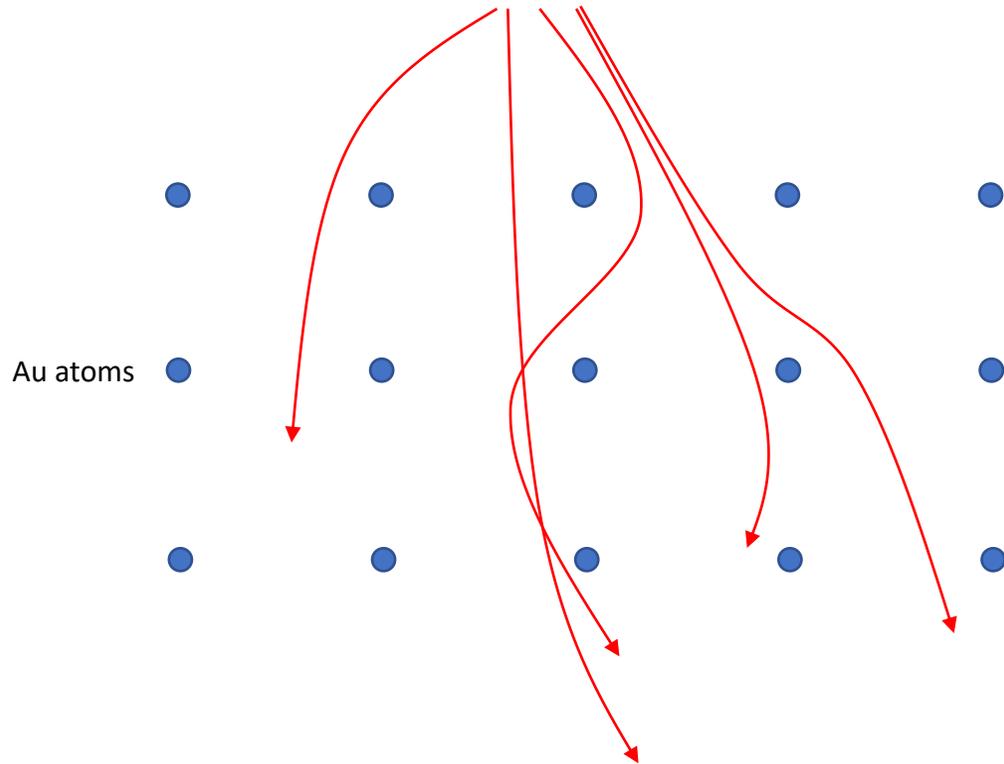
$$\lim_{N \rightarrow \infty} \hat{E}[\Theta] = E[\Theta]$$

accuracy can get arbitrary good

- Compared to numerical integration, e.g. trapezoidal rule, MC integration is **fast**: Improve accuracy for  $d$ -dimensional integral like

- $1/n^{2/d}$  for trapezoidal rule with  $n$  points
- $1/n^{1/2}$  for MC integration with  $n$  samples:  
→ for  $d > 4$ , MC integration is faster

# Samples as Particle Trajectories



As particle physics simulation can be considered virtual experiments, the samples have a clear interpretation:

→ They describe the **trajectory (=track)** a particle using the sampled values as input variables would follow within the given physics model

→ Like in real experiments, one can “measure” more than one observable, e.g. also energy loss  $\Delta E$ :

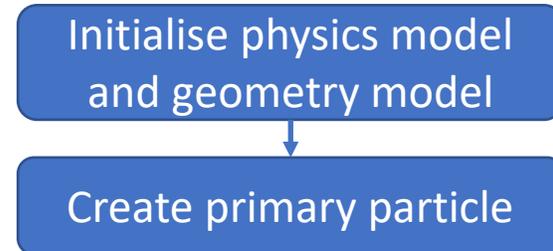
the output is then a tuple  $\theta \rightarrow \vec{\theta} = (\theta, \Delta E)$

# Workflow of a Simulation

Initialise physics model  
and geometry model

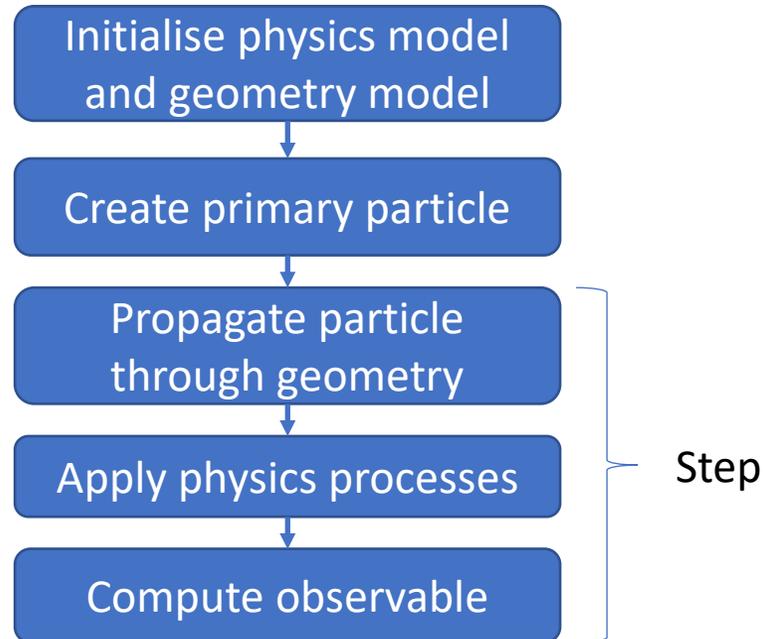
- Setup the virtual experiment:
  - Initialize the geometry model: which materials are placed at which regions or positions?
  - Initialize the physics model: compute material dependent model parameters (based on the geometry model)
  - Decide how many samples  $N$  should be drawn

# Workflow of a Simulation



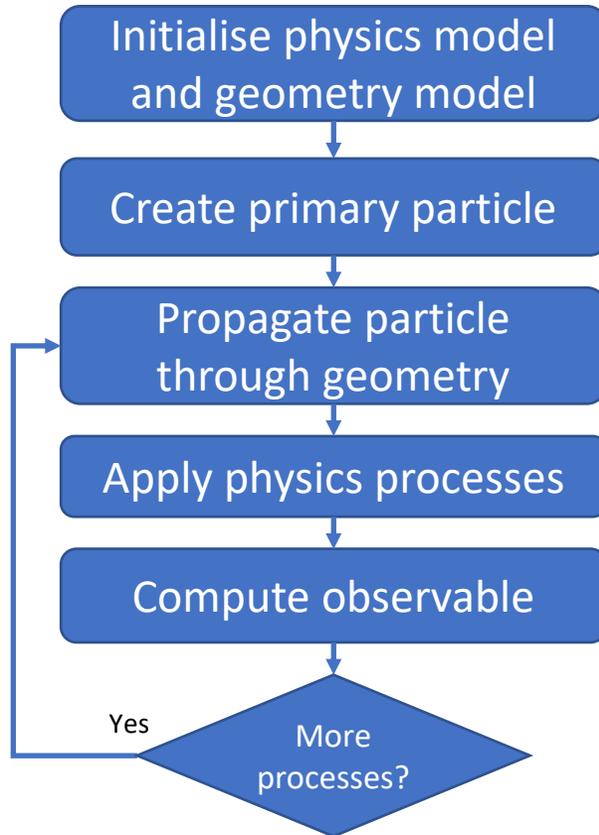
- Use a **primary particle generator** to sample the random variables that define the primary particle
  - Initial direction
  - Initial position (considering constraints from the geometry model, e.g. if primary particles can only be created within a **source** region)
  - Kinetic energy
  - Particle type

# Workflow of a Simulation



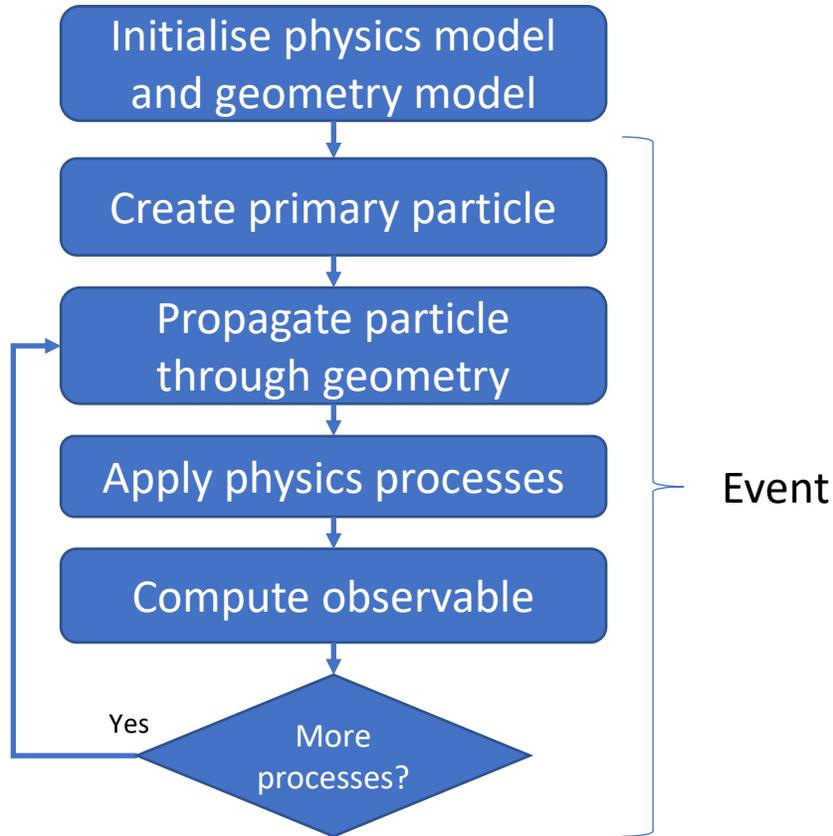
- Start the track of the primary particle
- Based on the physics model, compute the mean free path  $\lambda$ , i.e. average distance between two interactions
- Move the particle along the track by  $\lambda$ , make one **step**
- Compute the interaction, if needed sample input parameters, apply resulting changes on the track, e.g. changing direction due to deflection, reduce kinetic energy due to energy loss
- Update the observable(**s**) (e.g. deflection angle, energy loss) accordingly

# Workflow of a Simulation



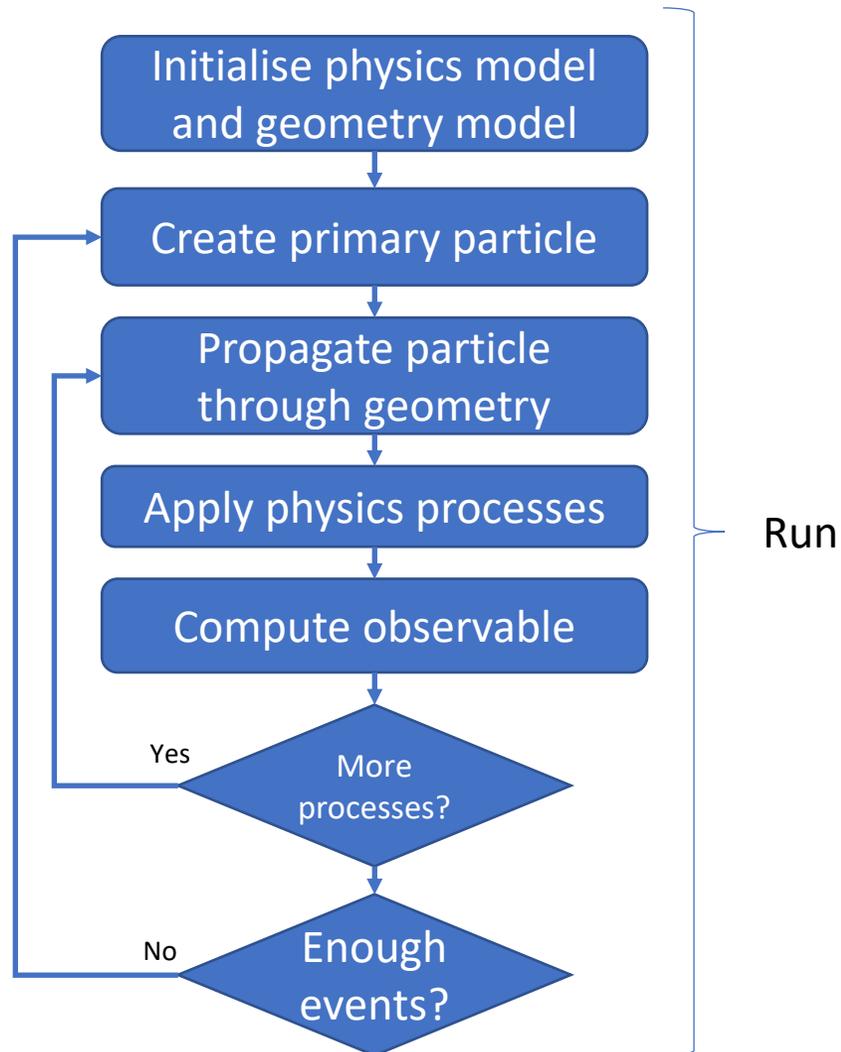
- Are there more processes that can apply to the particle?
- If yes (e.g. multiple scattering in a finite volume), repeat the previous step
- “No” could mean
  - The particle is unstable, and ceased to exist
  - The particle moved out of the finite geometrical model
  - The user deliberately limited the amount of iterations due to time or computing costs
- If no, then the track is finished
  - one sample  $\vec{x}$  from the total sample space  $\Omega$  was drawn
  - observable  $\vec{\theta}(\vec{x})$  was computed

# Workflow of a Simulation



- All steps from creating the primary particle until ending the track are some times referred to as an **event**
- Within one event, one sample from the *total* sample space  $\Omega$  was drawn

# Workflow of a Simulation



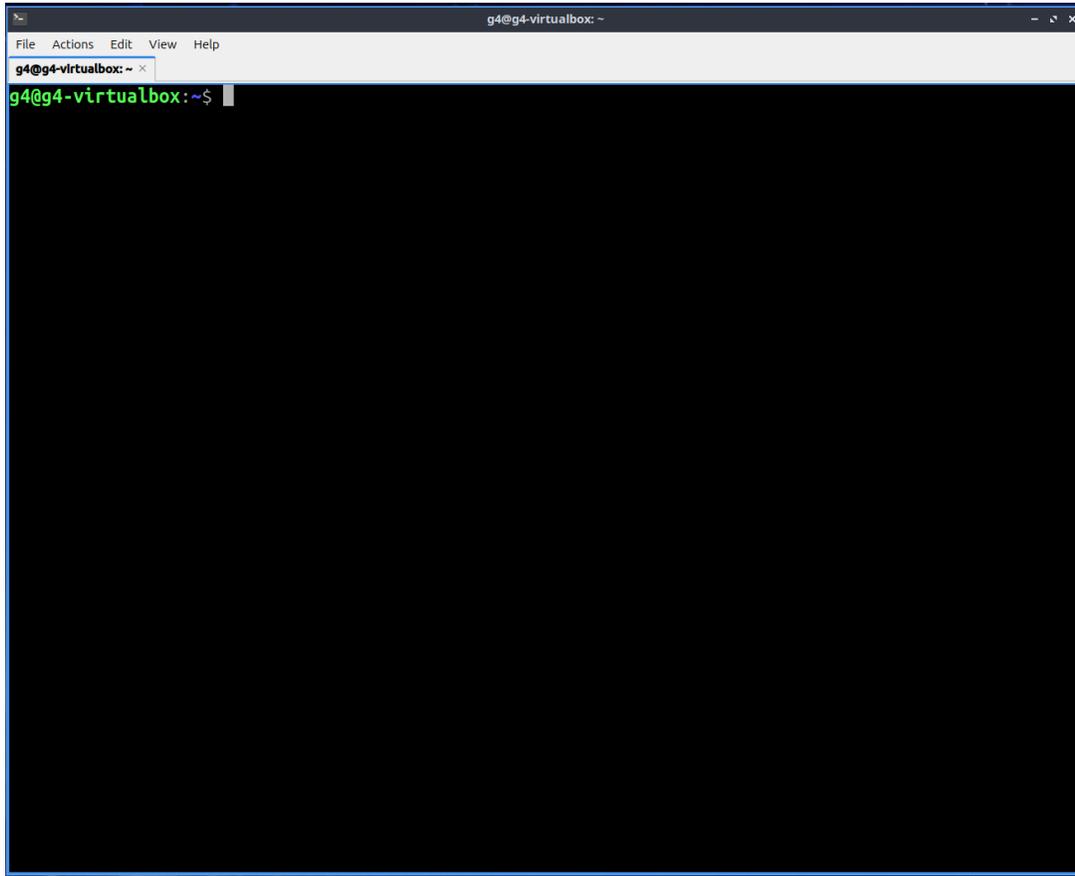
- Draw more samples (=compute more events) until  $N$  is reached
- Sometimes, all steps needed to obtain  $N$  samples is referred to as one **run**

Questions?

# Programming Environment

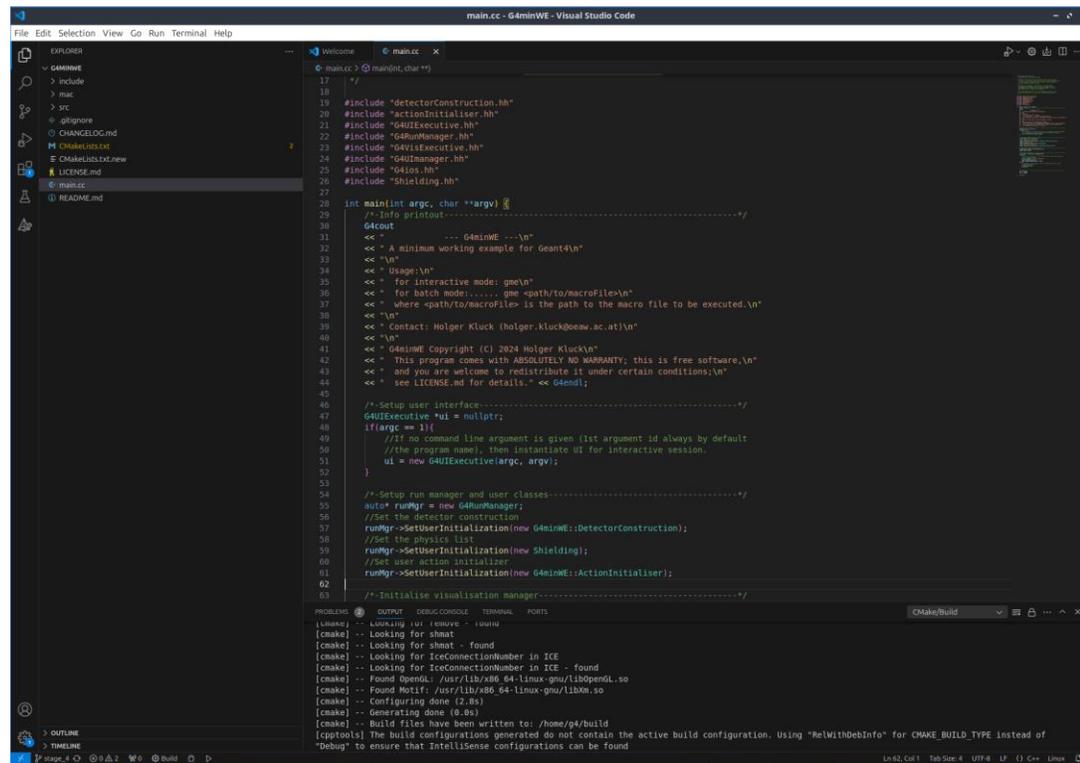
Linux; Terminal; Visual Studio Code; Git; CMake; Geant4

# Programming Environment



- For the hand-ons examples, we will use the MC simulation framework “Geant4”
- It is best to run it under Linux
- You will interact with Linux mostly via text commands, entered in a **terminal** window – it’s a “Text User Interface” (TUI)

# Linux



```
17 //
18
19 #include "detectorConstruction.hh"
20 #include "actionInitialiser.hh"
21 #include "G4MIME.hh"
22 #include "G4MIMEManager.hh"
23 #include "G4MIMEUserInterface.hh"
24 #include "G4MIMEPhysicsList.hh"
25 #include "G4MIMEActionInitialiser.hh"
26 #include "Shielding.hh"
27
28
29 int main(int argc, char **argv) {
30     /*-Info printout-----*/
31     G4cout
32     << " --- G4MIME ---\n"
33     << " * A minimum working example for Geant4\n"
34     << " * Usage:\n"
35     << " * for interactive mode: gme\n"
36     << " * for batch mode:.... gme -path/to/macroFile\n"
37     << " * where <path/to/macroFile> is the path to the macro file to be executed.\n"
38     << "\n"
39     << " * Contact: Holger Kluck (holger.kluck@oeaw.ac.at)\n"
40     << "\n"
41     << " * G4MIME Copyright (C) 2014 Holger Kluck\n"
42     << " * This program comes with ABSOLUTELY NO WARRANTY; this is free software,\n"
43     << " * and you are welcome to redistribute it under certain conditions.\n"
44     << " * see LICENSE.md for details." << G4endl;
45
46     /*-Setup user interface-----*/
47     G4MIMEUserInterface *ui = nullptr;
48     if(argc == 1){
49         //If no command line argument is given (1st argument id always by default
50         //the program name), then instantiate UI for interactive session.
51         ui = new G4MIMEUserInterface(argc, argv);
52     }
53
54     /*-Setup run manager and user classes-----*/
55     auto runMgr = new G4MIMEManager;
56     //Set the detector construction
57     runMgr->SetUserInitialization(new G4MIME::DetectorConstruction);
58     //Set the physics list
59     runMgr->SetUserInitialization(new Shielding);
60     //Set user action initializer
61     runMgr->SetUserInitialization(new G4MIME::ActionInitialiser);
62
63     /*-Initialise visualization manager-----*/
64 }
```

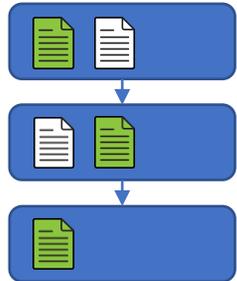
• For the actual programming, there are also **integrated development environments (IDE)** which provide many benefits:

- syntax highlighting
- code completion
- etc.

• We will use **Microsoft Visual Studio Code (VSC)**  
<https://code.visualstudio.com/>

- Other common IDEs are, e.g.
- eclipse  
<https://projects.eclipse.org/projects/tools.cdt>
  - CLion  
<https://www.jetbrains.com/de-de/clion/>

# Git



3. commit: change first file
2. commit: add another file
1. commit: add a new file

- Git is a code repository, it allows a user to track the changes made to a set of file over time
- We will use it for the files with the source code for the hands-on examples: **~/G4minWE**
- Via so-called **commits** the user can ask Git to make snapshots of the files within the repository
- Each commit is identified by its **hash**
- One can go to other commits (e.g. earlier ones) without losing the current state via the **checkout** command

# Git

```
g4@g4-virtualbox: ~/G4minWE
File Actions Edit View Help
g4@g4-virtualbox: ~/G4minWE
g4@g4-virtualbox:~$ cd ./G4minWE/
g4@g4-virtualbox:~/G4minWE$ git branch -a
main
stage_0
* stage_4
remotes/origin/HEAD -> origin/main
remotes/origin/main
remotes/origin/stage_0
remotes/origin/stage_1
remotes/origin/stage_2
remotes/origin/stage_3
remotes/origin/stage_4
remotes/origin/stage_5
g4@g4-virtualbox:~/G4minWE$
```

- The hands-on examples follow a bottom-up approach: each example is an extension of the previous one
- The examples-repository provides branches that contain the “extra code” needed to go from one examples to the next: stage\_0, stage\_1, etc.
- Branches with prefix “remotes” are not yet copied from remote repository

# Git

```
g4@g4-virtualbox: ~/G4minWE
File Actions Edit View Help
g4@g4-virtualbox: ~/G4minWE x
g4@g4-virtualbox:~$ cd ./G4minWE/
g4@g4-virtualbox:~/G4minWE$ git branch -a
main
stage_0
* stage_4
remotes/origin/HEAD -> origin/main
remotes/origin/main
remotes/origin/stage_0
remotes/origin/stage_1
remotes/origin/stage_2
remotes/origin/stage_3
remotes/origin/stage_4
remotes/origin/stage_5
g4@g4-virtualbox:~/G4minWE$ git checkout stage_0
Switched to branch 'stage_0'
Your branch is up-to-date with 'origin/stage_0'.
g4@g4-virtualbox:~/G4minWE$
```

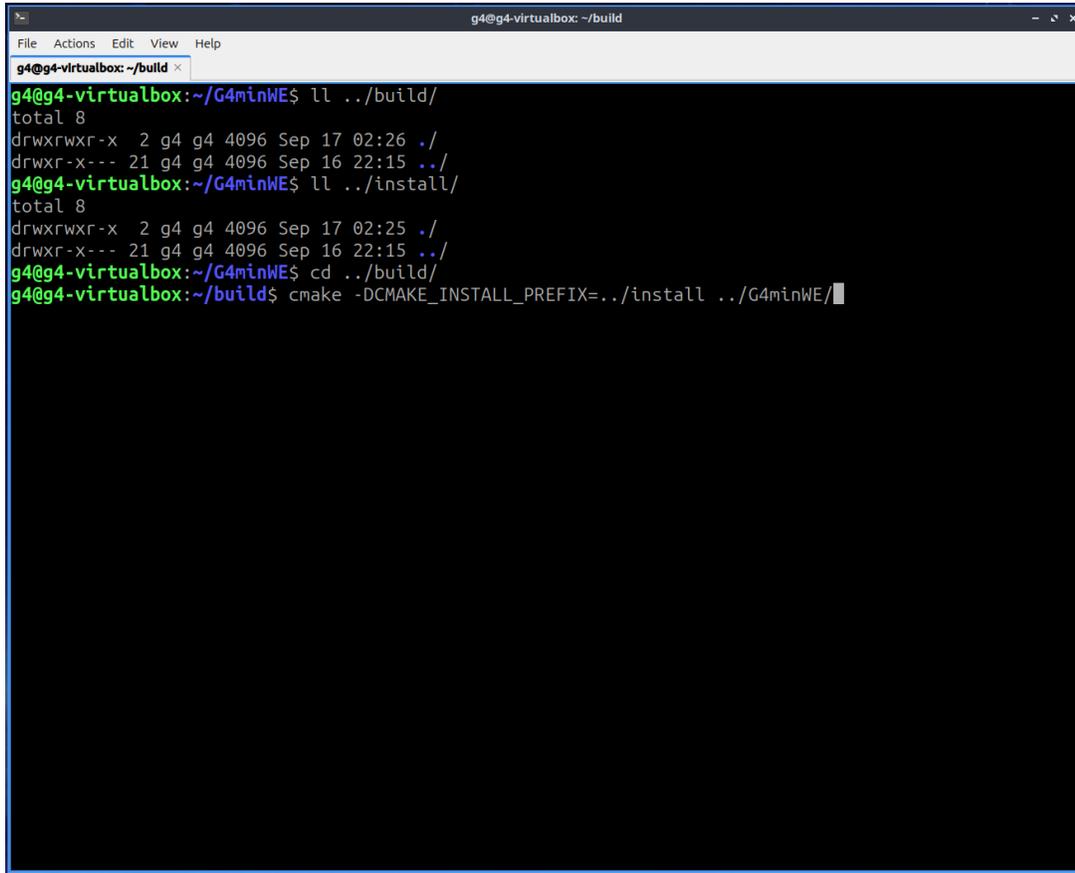
- Use **checkout** command to change to a branch
- We will use “stage\_0” for the very first hands-on
- For latter hands-on we will use “stage\_4”

# CMake



- To manage the compilation of the simulation code, we will use CMake:  
<https://cmake.org/>
  - Depending on the provide CMakeLists.txt file CMake will determine the dependencies between the different parts of the code and generate a **build script** – called **configuring** the project
  - Depending on this build script, it will then call the **compiler** to compile the source code to object files and call the **linker** to link the object files together to the executable – called **building** the project
  - As actual compiler we will use the GNU Compiler Collection – but thanks to CMake we do not interact directly with it

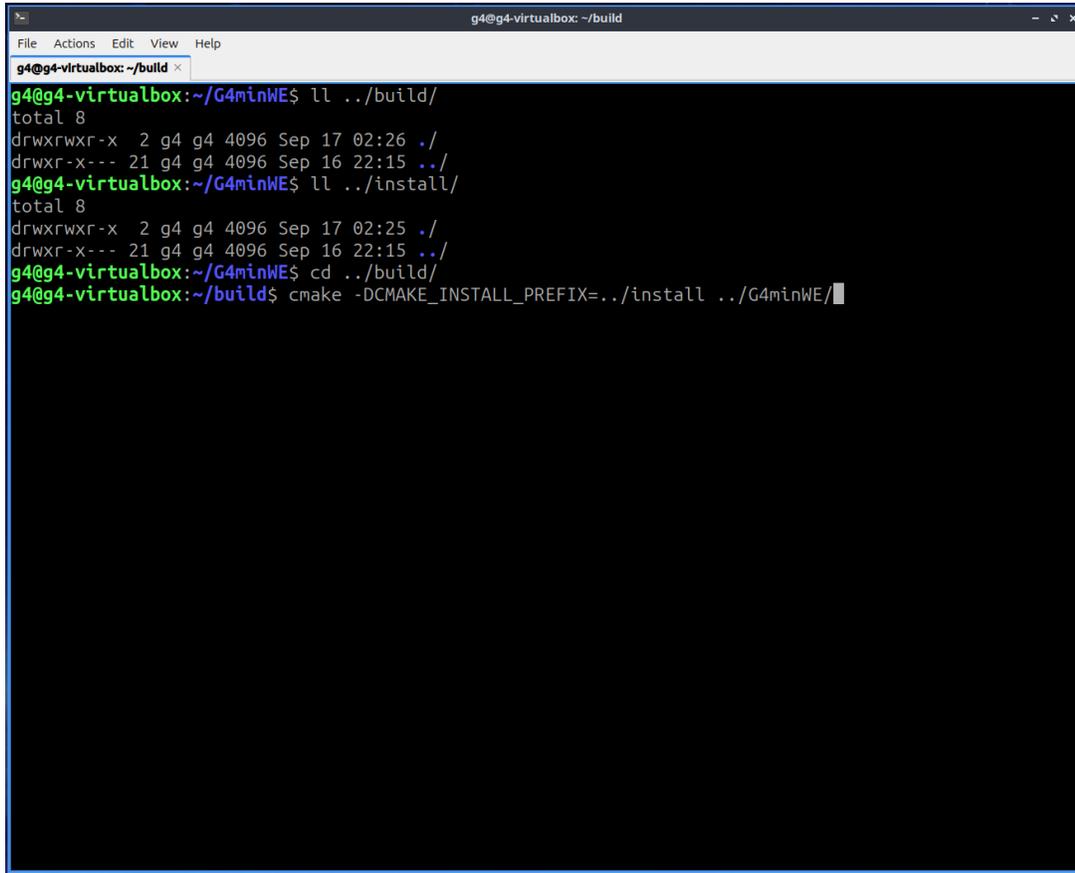
# CMake



```
g4@g4-virtualbox: ~/build
File Actions Edit View Help
g4@g4-virtualbox: ~/build x
g4@g4-virtualbox:~/G4minWE$ ll ../build/
total 8
drwxrwxr-x  2 g4 g4 4096 Sep 17 02:26 ./
drwxr-x--- 21 g4 g4 4096 Sep 16 22:15 ../
g4@g4-virtualbox:~/G4minWE$ ll ../install/
total 8
drwxrwxr-x  2 g4 g4 4096 Sep 17 02:25 ./
drwxr-x--- 21 g4 g4 4096 Sep 16 22:15 ../
g4@g4-virtualbox:~/G4minWE$ cd ../build/
g4@g4-virtualbox:~/build$ cmake -DCMAKE_INSTALL_PREFIX=../install ../G4minWE/
```

- CMake uses 3 directories:
  - One that contains the **source code** of the program to be build, e.g. the local copy of a repository
  - The **build directory** where Cmake creates the build script, runs the compiler, etc.
  - The **install directory** where the compiled executable will be copied to
- This way, build artefacts (=temporary files needed during building) will not “pollute” the source files and after installing one can simply delete the build directory with all its temporary files

# CMake

A terminal window titled 'g4@g4-virtualbox: ~/build' showing a series of commands and their outputs. The user navigates to the 'build' directory, lists its contents, then navigates to the 'install' directory and lists its contents. Finally, the user runs the 'cmake' command with the option '-DCMAKE\_INSTALL\_PREFIX=../install' in the 'build' directory.

```
g4@g4-virtualbox:~/G4minWE$ ll ../build/
total 8
drwxrwxr-x  2 g4 g4 4096 Sep 17 02:26 ./
drwxr-x--- 21 g4 g4 4096 Sep 16 22:15 ../
g4@g4-virtualbox:~/G4minWE$ ll ../install/
total 8
drwxrwxr-x  2 g4 g4 4096 Sep 17 02:25 ./
drwxr-x--- 21 g4 g4 4096 Sep 16 22:15 ../
g4@g4-virtualbox:~/G4minWE$ cd ../build/
g4@g4-virtualbox:~/build$ cmake -DCMAKE_INSTALL_PREFIX=../install ../G4minWE/
```

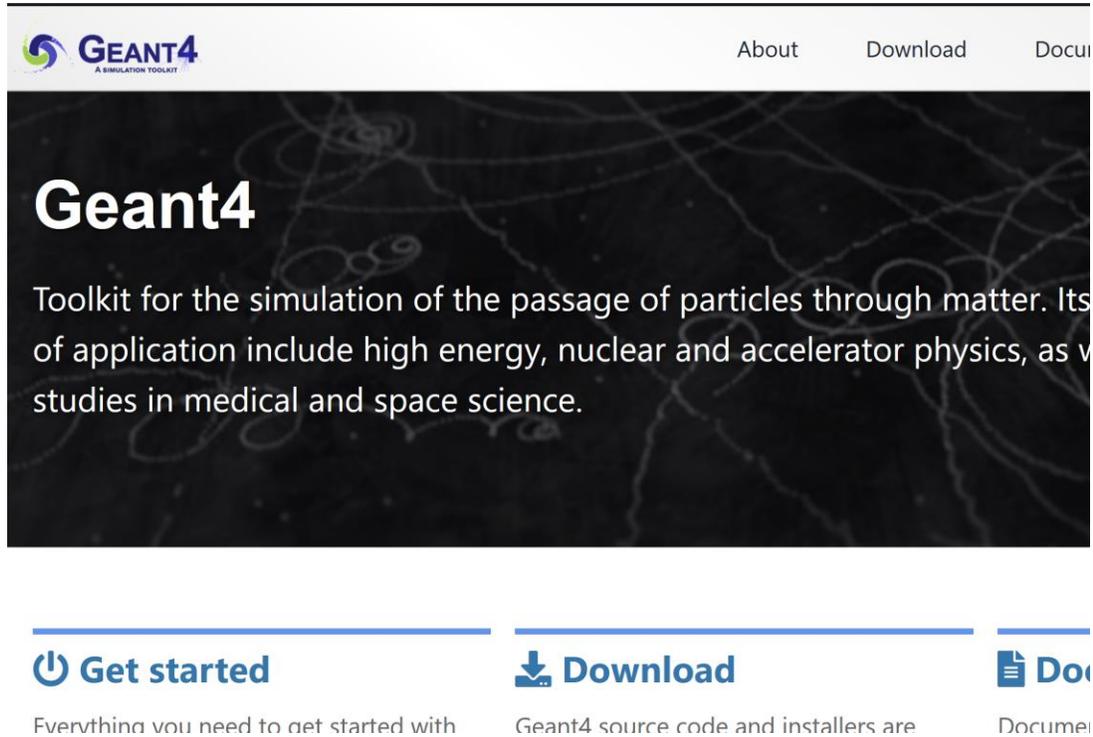
- The **cmake** command configure a project
  - One can specify the install directory via the option `-DCMAKE_INSTALL_PREFIX`
  - The argument to cmake is the source directory
  - It's necessary if one adds or removes source code files from a project
  
- In the pre-installed VSC, you can configure your project by pressing the [F8] key

# CMake

```
g4@g4-virtualbox: ~/build
File Actions Edit View Help
g4@g4-virtualbox: ~/build x
-- Found ZLIB: /usr/lib/x86_64-linux-gnu/libz.so (found suitable version "1.3", minimum required is "1.3")
-- Found XercesC: /usr/lib/x86_64-linux-gnu/libxerces-c.so (found suitable version "3.2.4", minimum required is "3.2.4")
-- Found Freetype: /usr/lib/x86_64-linux-gnu/libfreetype.so (found suitable version "2.13.2", minimum required is "2.13.2")
-- Found X11: /usr/include
-- Looking for XOpenDisplay in /usr/lib/x86_64-linux-gnu/libX11.so;/usr/lib/x86_64-linux-gnu/libXext.so
-- Looking for XOpenDisplay in /usr/lib/x86_64-linux-gnu/libX11.so;/usr/lib/x86_64-linux-gnu/libXext.so - found
-- Looking for gethostbyname
-- Looking for gethostbyname - found
-- Looking for connect
-- Looking for connect - found
-- Looking for remove
-- Looking for remove - found
-- Looking for shmat
-- Looking for shmat - found
-- Looking for IceConnectionNumber in ICE
-- Looking for IceConnectionNumber in ICE - found
-- Found OpenGL: /usr/lib/x86_64-linux-gnu/libOpenGL.so
-- Found Motif: /usr/lib/x86_64-linux-gnu/libXm.so
-- Configuring done (1.1s)
-- Generating done (0.0s)
-- Build files have been written to: /home/g4/build
g4@g4-virtualbox:~/build$ ll
total 76
drwxrwxr-x 3 g4 g4 4096 Sep 17 02:27 ./
drwxr-x--- 21 g4 g4 4096 Sep 16 22:15 ../
-rw-rw-r-- 1 g4 g4 51034 Sep 17 02:27 CMakeCache.txt
drwxrwxr-x 6 g4 g4 4096 Sep 17 02:27 CMakeFiles/
-rw-rw-r-- 1 g4 g4 3339 Sep 17 02:27 cmake_install.cmake
-rw-rw-r-- 1 g4 g4 7032 Sep 17 02:27 Makefile
g4@g4-virtualbox:~/build$ cmake --build . --target install -j2
```

- Once the build script is generate, **cmake --build** start the actual building
  - The **--target install** option tells cmake to copy the built files to the install directory
  - If one has more CPU cores available, one can assign *n* of them to the build process via the **-jn** option
- In the pre-installed VSC, you can compile your project by pressing the [F7] key
- And install it by pressing the [F9] key

# Geant4



The screenshot shows the Geant4 website homepage. At the top left is the Geant4 logo with the tagline "A SIMULATION TOOLKIT". To the right are navigation links for "About", "Download", and "Docu". The main content area has a dark background with a particle track visualization. The heading "Geant4" is in large white font, followed by a paragraph: "Toolkit for the simulation of the passage of particles through matter. Its of application include high energy, nuclear and accelerator physics, as v studies in medical and space science." At the bottom, there are three columns of navigation links: "Get started" (with a power icon), "Download" (with a download icon), and "Docu" (with a document icon). Below each link is a short description: "Everything you need to get started with", "Geant4 source code and installers are", and "Documen".

- Geant4 is freely available from CERN: <https://geant4.web.cern.ch/>
- Most current version is 11.2.1, we will use 10.6.3
- Manuals: <https://geant4.web.cern.ch/docs/> especially the Book For Application Developer (BAD)
- API documentation: <https://geant4.kek.jp/Reference/>  
<https://geant4.kek.jp/LXR/>

# Hands-on

- Change to the source directory under your home directory:
- Checkout the „stage\_0“ branch:
- Change to the „build“ directory, configure and build the program via the command line:
- Change to the “install” directory and run G4minWE

# Hands-on

- Change to the source directory under your home directory:

```
cd ~/G4minWE
```

- Checkout the „stage\_0“ branch:

```
git checkout stage_0
```

- Change to the „build“ directory, configure and build the program via the command line:

```
cd ../build
```

```
cmake -DCMAKE_INSTALL_PREFIX=../install ../G4minWE
```

```
cmake --build . --target install -j2
```

- Change to the “install” directory and run G4minWE

```
cd ../install
```

```
./bin/G4minWE
```

# Basics of Geant4

Basic Structure; Visualisation; Macro Files

# Geant4

```
18
19 #include "detectorConstruction.hh"
20
21 #include "G4UIExecutive.hh"
22 #include "G4RunManager.hh"
23 #include "G4VisExecutive.hh"
24 #include "G4UImanager.hh"
25 #include "G4ios.hh"
26 #include "Shielding.hh"
27
28 int main(int argc, char **argv) {
29     /*-Info printout-----
30     G4cout
31     << "          --- G4minWE ---\n"
32     << " A minimum working example for Geant4\n"
33     << "\n"
34     << " Usage:\n"
35     << "   for interactive mode: gme\n"
36     << "   for batch mode:..... gme <path/to/macroFile>\n"
```

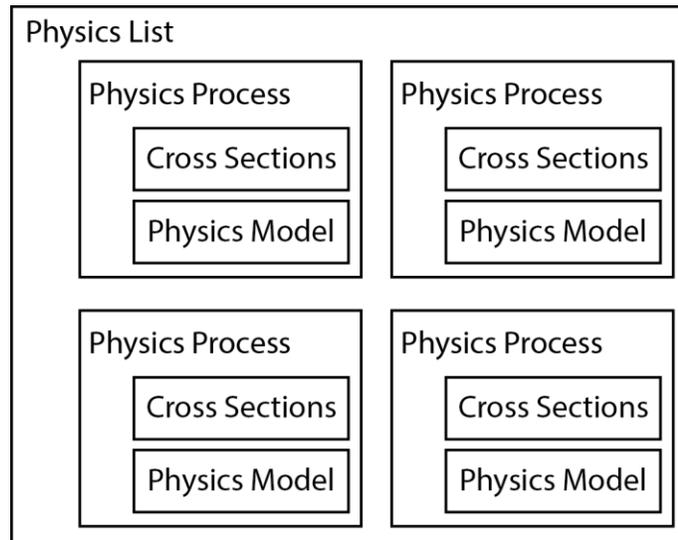
- The user interacts with the Geant4 framework via the `main()` function
  - Geometry model, physics list, primary particle generation are specified in **classes the user derived from Geant4 base classes**
  - Some features are provided as ready-to-use, e.g. **visualisation**
  - In the main function, instances of these user defined classes are passed to the **manager classes** provided by Geant4

# Physics List

```
58 //Set the physics list
59 runMgr->SetUserInitialization(new Shielding);
```

- The physics list has to be
  - Instantiated in the `main()` function
  - Registered to the `G4RunManger`
  - And must not be deleted
- Geant4 provides several pre-defined physics lists tuned for several use cases, see [Guide for Physics Lists](#)
- In our examples, we use `Shielding`

# Physics List



- Geant4 offers the users flexibility which kind of physics to apply in the simulation via **physics lists** [BAD §6.2.2]
  - List of **physics processes** that are applicable for a **particle**
  - A physics process is a combination of **physics model** and **cross sections**
  - Physics models give the **final state** of the reaction products, including any **secondary particles**

# Visualisation

```
61     /*-Initialise visualisation manager-----  
62     G4VisManager* visMgr = new G4VisExecutive;  
63     visMgr->Initialize();  
64
```

- Geant4 can **visualize** the implemented geometry (and the particle interaction happening within)
- To enable visualisation, the **visualisation manager** has to be instantiated in the main function
- Depending on the way Geant4 as installed, several **visualisation drivers** are available [BAD, §8.1.2]
- User can configure it via **macro files**

# Macro Files

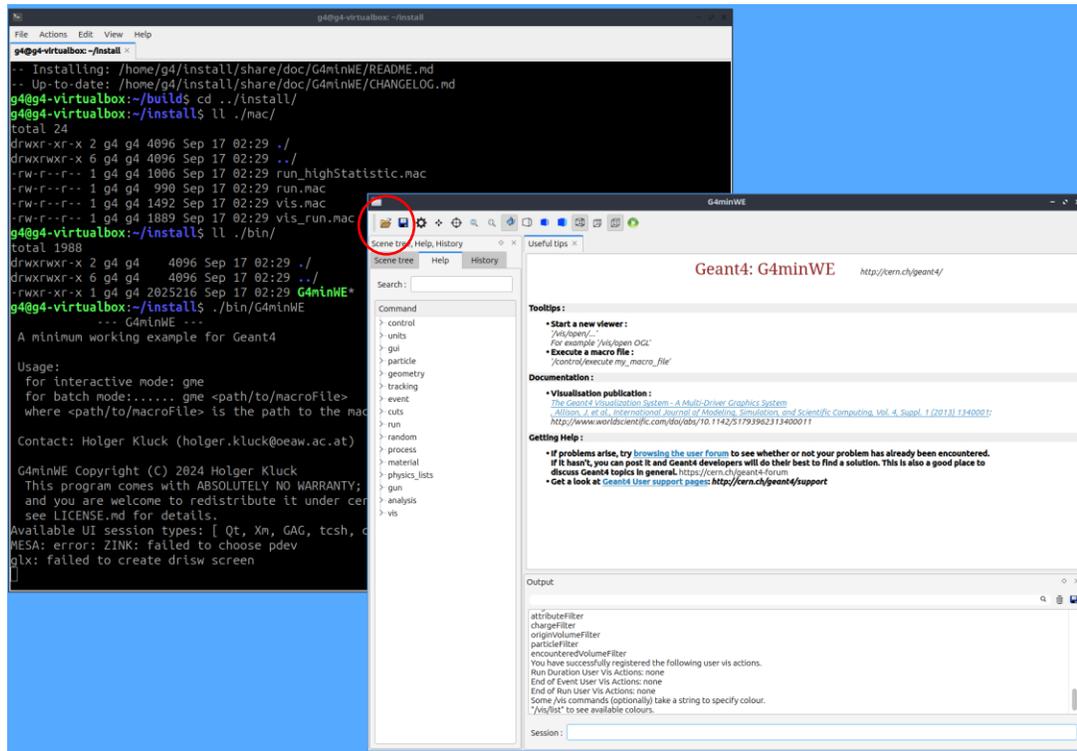
- Geant4 can be controlled via **macro files** (file extension: mac)

# Macro Files

```
g4@g4-virtualbox: ~/install
File Actions Edit View Help
g4@g4-virtualbox: ~/install x
Install the project...
-- Install configuration: "RelWithDebInfo"
-- Installing: /home/g4/install/bin/G4minWE
-- Set non-toolchain portion of runtime path of "/home/g4/install/bin/G4minWE" to ""
-- Installing: /home/g4/install/include
-- Installing: /home/g4/install/include/sensitiveDetector.hh
-- Installing: /home/g4/install/include/eventAction.hh
-- Installing: /home/g4/install/include/hit.hh
-- Installing: /home/g4/install/include/detectorConstruction.hh
-- Installing: /home/g4/install/include/primaryParticleAction.hh
-- Installing: /home/g4/install/include/runAction.hh
-- Installing: /home/g4/install/include/actionInitialiser.hh
-- Installing: /home/g4/install/mac
-- Installing: /home/g4/install/mac/vis.mac
-- Installing: /home/g4/install/mac/run.mac
-- Installing: /home/g4/install/mac/run_highStatistic.mac
-- Installing: /home/g4/install/mac/vis_run.mac
-- Up-to-date: /home/g4/install/share/doc/G4minWE/LICENSE.md
-- Installing: /home/g4/install/share/doc/G4minWE/README.md
-- Up-to-date: /home/g4/install/share/doc/G4minWE/CHANGELOG.md
g4@g4-virtualbox:~/build$ cd ../install/
g4@g4-virtualbox:~/install$ ll ./mac/
total 24
drwxr-xr-x 2 g4 g4 4096 Sep 17 02:29 ./
drwxrwxr-x 6 g4 g4 4096 Sep 17 02:29 ../
-rw-r--r-- 1 g4 g4 1006 Sep 17 02:29 run_highStatistic.mac
-rw-r--r-- 1 g4 g4 990 Sep 17 02:29 run.mac
-rw-r--r-- 1 g4 g4 1492 Sep 17 02:29 vis.mac
-rw-r--r-- 1 g4 g4 1889 Sep 17 02:29 vis_run.mac
g4@g4-virtualbox:~/install$ ll ./bin/
total 1988
drwxrwxr-x 2 g4 g4 4096 Sep 17 02:29 ./
drwxrwxr-x 6 g4 g4 4096 Sep 17 02:29 ../
-rwxr-xr-x 1 g4 g4 2025216 Sep 17 02:29 G4minWE*
g4@g4-virtualbox:~/install$
```

- Geant4 can be controlled via **macro files**
- Pass a macro file either on the command line for **batch mode**

# Macro Files



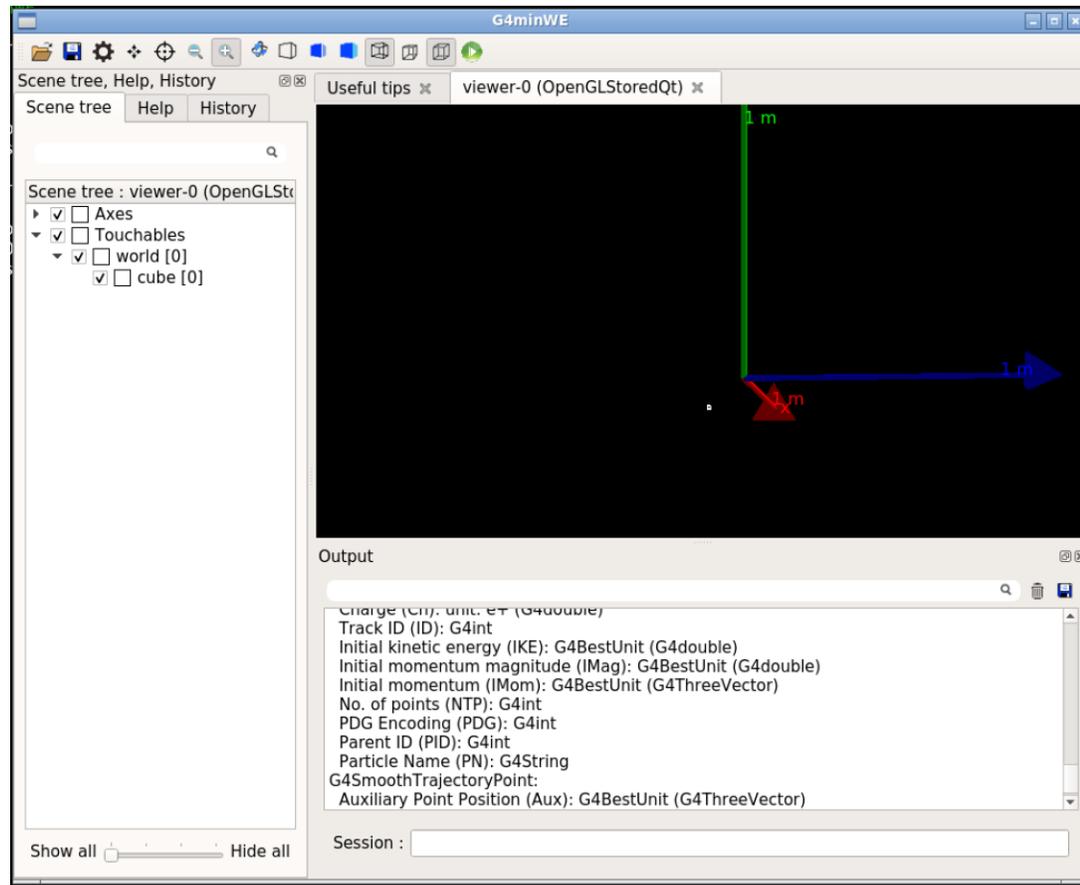
- Geant4 can be controlled via **macro files**
- Pass a macro file either on the command line for **batch mode**
- Or select it in an **interactive GUI** (via the „open file“ icon)
- If you want to simulate large numbers of events, use batch mode; use GUI only for test or debugging purposes

# Visualisation

```
17 #Initialse Geant4
18 /run/initialize
19
20 #Use OpenGL visualiser with 600 pixel x 600 pixel window size
21 /vis/open OGL 600x600-0+0
22
23 #Or create a HepRepXML file containing the visualisation
24 #view it with JAS3
25 #/vis/open HepRepXML
26
27 #Visualiser should report errors
28 /vis/verbose errors
29
30 #Draw the geometry
31 /vis/drawVolume
32
33 #View on the scene from top
34 /vis/viewer/set/viewpointVector -1 0 0
35
36 #Draw the scene as wireframe
37 /vis/viewer/set/style wireframe
38 /vis/viewer/set/auxiliaryEdge true
39 #Increase number of sampling points for circles
40 /vis/viewer/set/lineSegmentsPerCircle 100
41
42 #Add a axes cross which length of each axes of 1 m
43 /vis/scene/add/axes 0 0 0 1 m
44
45 #For file-based drivers, use this to create an empty detector view:
46 #/vis/viewer/flush
47
```

- Control the visualisation settings via macro file commands
  - Before the geometry can be visualised, Geant4 need to be initialised
  - Select the visualization driver
    - OGL for interactive visualisation
    - HepRepXML / **JAS3** for offline use
  - Draw the geometry
  - Configure the visualisation style, add axes cross, orient the point of view, etc.  
see [list of all commands](#)

# Interactive Visualisation



- In G4minWE, by default **OpenGL** is used a visualisation driver
  - It is interactive: one can pan and rotate the scene via mouse
  - Zoom in and out
  - Switch on/off individual volumes via the scene tree
  - Macro file vis.mac from stage\_1 of G4minWE onwards adds an axes cross to the small PMMA (=Acrylic glass) cube defined in DetectorConstruction

# Offline Visualisation

```
17 #Initialse Geant4
18 /run/initialize
19
20 #Use OpenGL visualiser with 600 pixel x 600 pixel window size
21 // #/vis/open OGL 600x600-0+0
22
23 #Or create a HepRepXML file containing the visualisation
24 #view it with JAS3
25 // #/vis/open HepRepXML
26
27 #Visualiser should report errors
28 /vis/verbose errors
29
30 #Draw the geometry
31 /vis/drawVolume
32
33 #View on the scene from top
34 /vis/viewer/set/viewpointVector -1 0 0
35
36 #Draw the scene as wireframe
37 /vis/viewer/set/style wireframe
38 /vis/viewer/set/auxiliaryEdge true
39 #Increase number of sampling points for circles
40 /vis/viewer/set/lineSegmentsPerCircle 100
41
42 #Add a axes cross which length of each axes of 1 m
43 /vis/scene/add/axes 0 0 0 1 m
44
45 #For file-based drivers, use this to create an empty detector view:
46 // #/vis/viewer/flush
47
```

- One can also create an **HepRepXML** file that contains a description of the geometry  
→ adapt vis.mac as shown on the screen shot

# Offline Visualisation

```
g4@g4-virtualbox: ~/install
File Actions Edit View Help
g4@g4-virtualbox: ~/install x
g4@g4-virtualbox:~/install$ ./bin/G4minWE ./mac/vis.mac
--- G4minWE ---
A minimum working example for Geant4

Usage:
for interactive mode: gme
for batch mode:..... gme <path/to/macroFile>
where <path/to/macroFile> is the path to the macro file to be executed.

Contact: Holger Kluck (holger.kluck@oeaw.ac.at)

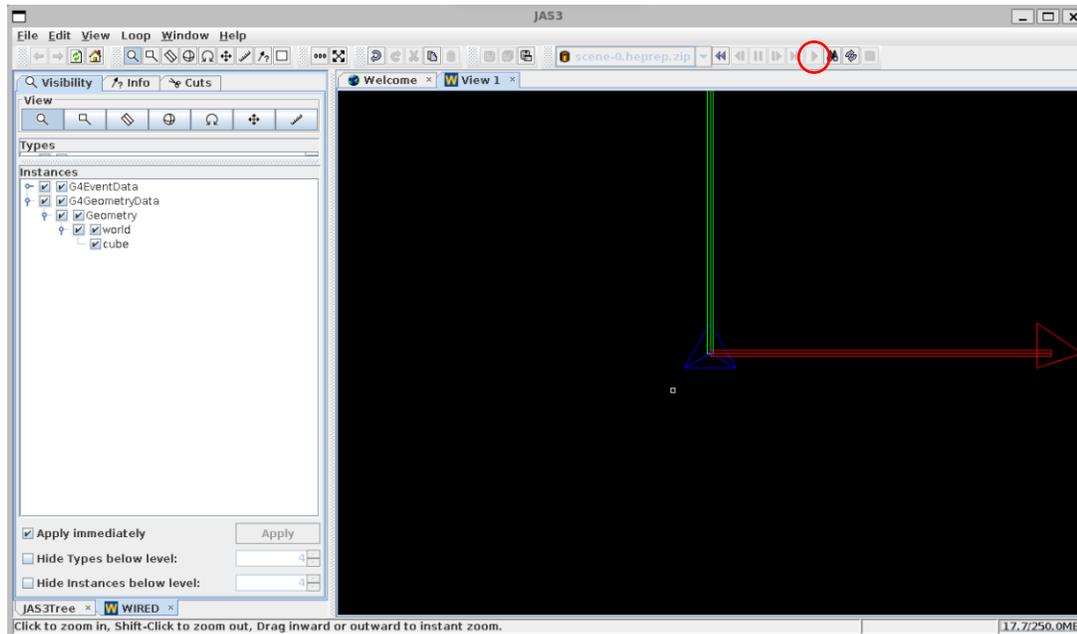
G4minWE Copyright (C) 2024 Holger Kluck
This program comes with ABSOLUTELY NO WARRANTY; this is free software,
and you are welcome to redistribute it under certain conditions;
see LICENSE.md for details.

*****
Geant4 version Name: geant4-10-06-patch-03 (6-November-2020)
Copyright : Geant4 Collaboration
References : NIM A 506 (2003), 250-303
            : IEEE-TNS 53 (2006), 270-278
```

```
Parent ID (PID): G4int
Particle Name (PN): G4String
G4SmoothTrajectoryPoint:
Auxiliary Point Position (Aux): G4BestUnit (G4ThreeVector)
Step Position (Pos): G4BestUnit (G4ThreeVector)
Graphics systems deleted.
Visualization Manager deleting...
g4@g4-virtualbox:~/install$ ll
total 28
drwxrwxr-x 6 g4 g4 4096 Sep 17 02:33 ./
drwxr-x--- 21 g4 g4 4096 Sep 16 22:15 ../
drwxrwxr-x 2 g4 g4 4096 Sep 17 02:29 bin/
drwxr-xr-x 2 g4 g4 4096 Sep 17 02:29 include/
drwxr-xr-x 2 g4 g4 4096 Sep 17 02:33 mac/
-rw-rw-r-- 1 g4 g4 2685 Sep 17 02:33 scene-0.heprep.zip
drwxrwxr-x 3 g4 g4 4096 Sep 17 02:28 share/
g4@g4-virtualbox:~/install$ jas3 scene-0.heprep.zip
```

- The **JAS3** tool can visualise the geometry described in a HepRepXML file
- The default name of the HepRepXML file is **scene-0.heprep.zip**

# Offline Visualisation



- `jas3 ./scene-0.heprep.zip`
- Open a Wire4 view via:  
“File > New > Wired4 View”
- If there is no „Wired4 View“ go to “View > Plugin Manager > Available > common” select „WIRE4“ and click „Install selected plugins“
- Click the „play“ button to start visualisation

# Hands-on

- Change to the source directory and checkout the „stage\_4“ branch:
- Configure, build and install the code via VSC:
- Change to the install directory and run `./mac/vis.mac` via the GUI
- Use VSC to activate the JAS3 visualisation in `./mac/vis.mac`, install it
- Run `./mac/vis.mac` in batch mode and open the output file in JAS3

# Hands-on

- **Change to the source directory and checkout the „stage\_4“ branch:**  
`cd ~/G4minWE`  
`git checkout stage_4`
- **Configure, build and install the code via VSC:**  
In VSC, press the [F8], [F7], [F9] keys
- **Change to the install directory and run ./mac/vis.mac via the GUI**  
`cd ../install`  
`./bin/G4minWE`  
In the GUI click “File open” icon, select ./mac/vis.mac
- **Use VSC to activate the JAS3 visualisation in ./mac/vis.mac, install it**  
Comment line 21, uncomment lines 25,46, press [F9]
- **Run ./mac/vis.mac in batch mode and open the output file in JAS3**  
`./bin/G4minWE ./mac/vis.mac`  
`jas3 scene-0.heprep.zip`

# Take Home Messages

- Simulations can be regarded as virtual experiments
- A background simulation depends crucially on its model assumptions
- Each simulated event is one drawn sample from the sample space – more samples results in a more precise result
- Geant4 is a free and widely used software framework to implement a MC simulation – the scope of the simulation is the responsibility of its developers
- Unfortunately, some tools are needed (e.g. Linux, IDEs, C++, etc.) to create a MC simulation – like real experiments depends on tools