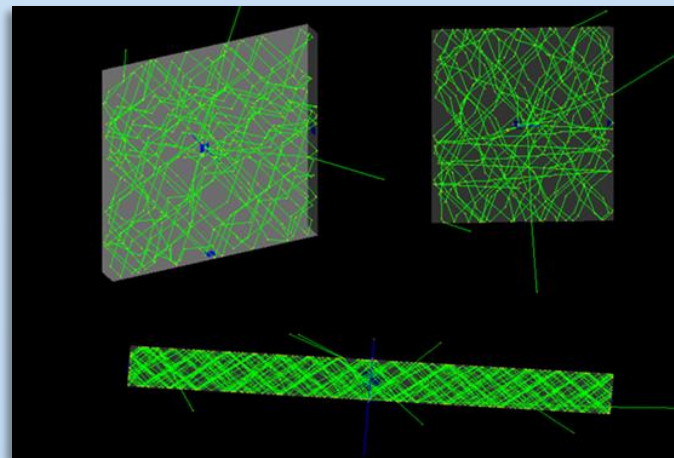


Simulation of optical photon propagation for generic scintillator-based detectors

Lecture 0

Introduction



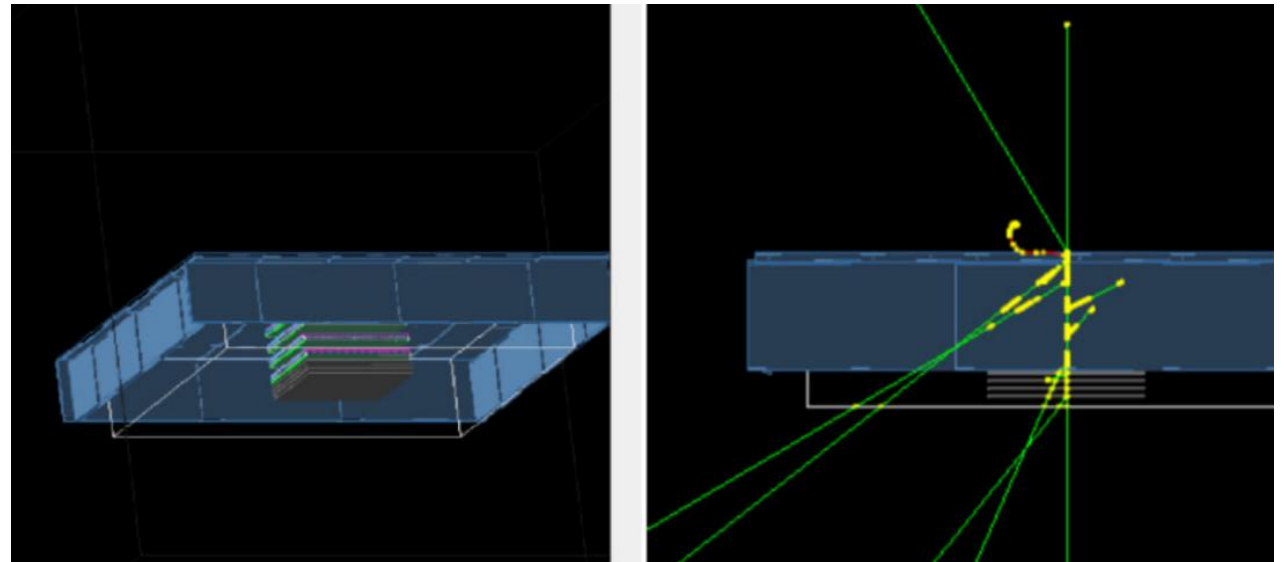
- In the first part of the course we will see a brief introduction of:
 - Scintillator and Photosensors
- The second part of the course provides an overview of Geant4:
 - its capabilities
 - how they can be used in an experimental simulation application (we will focus on the optical simulations)
- Geant4 is a complex and powerful toolkit
 - Impossible to teach (and learn) everything in few days and also our own expertise is not infinite
- This course would provide you with a global vision of Geant4 and a method for “how to use it”
 - Finding **your way** in the complexity of Geant4 is not easy
 - Use the Geant4 User Documentation and the “**BookForApplicationDevelopers**”

Geant4 web site: <http://cern.ch/geant4>

Detector simulation

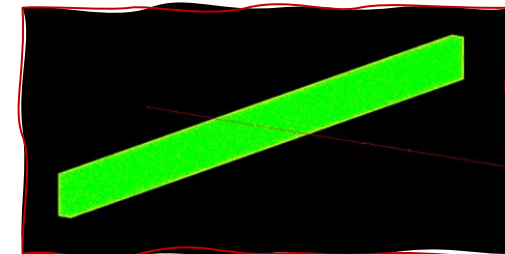
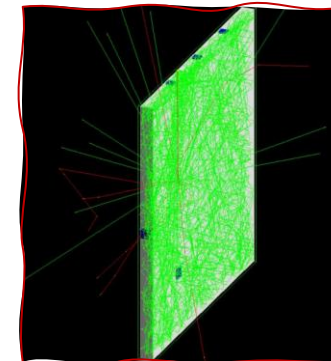
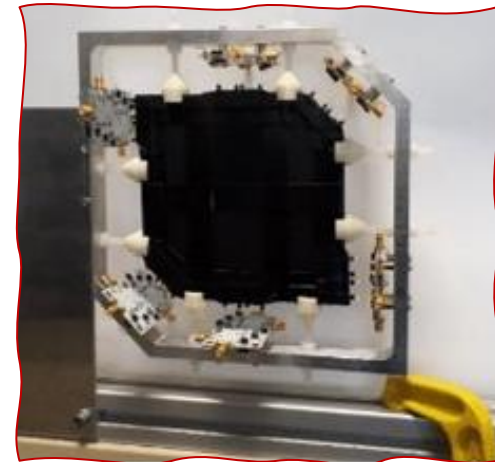
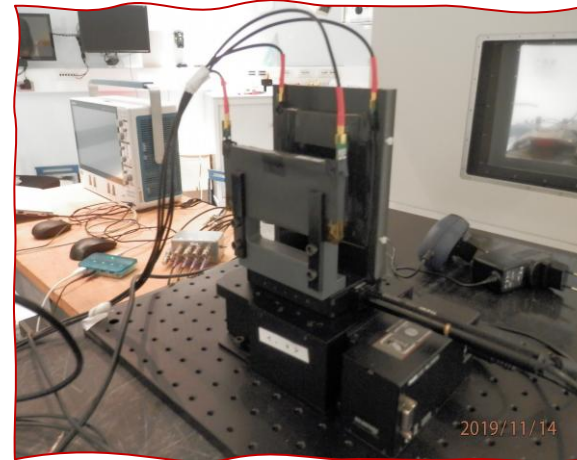
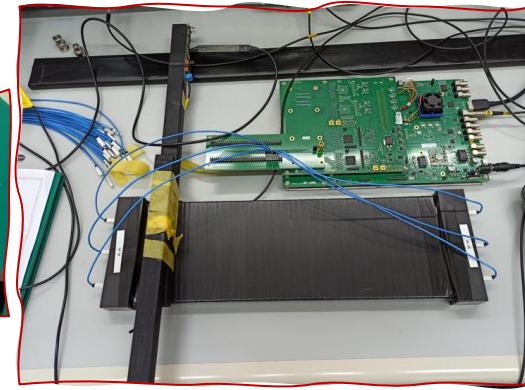
- Simulation is a very useful, essential tool in modern particle physics for:
 - Designing an experiment
 - *Optimization of the geometry*
 - Analysing the data
 - *Interpretation of the experimental results from a huge number of signals*
- The simulation is needed to investigate the technical design:
 - Energy dispersion and resolution
 - Possible leakage, corrections to take into account

Key element of recent experiments to optimize the performance, the size and the cost of each sub-detector



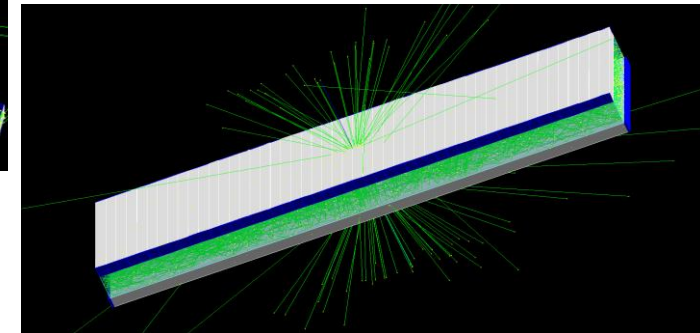
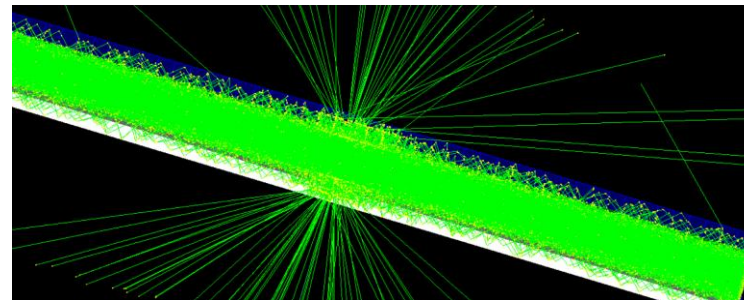
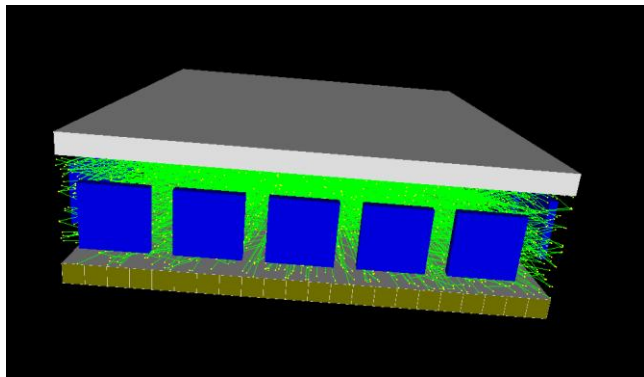
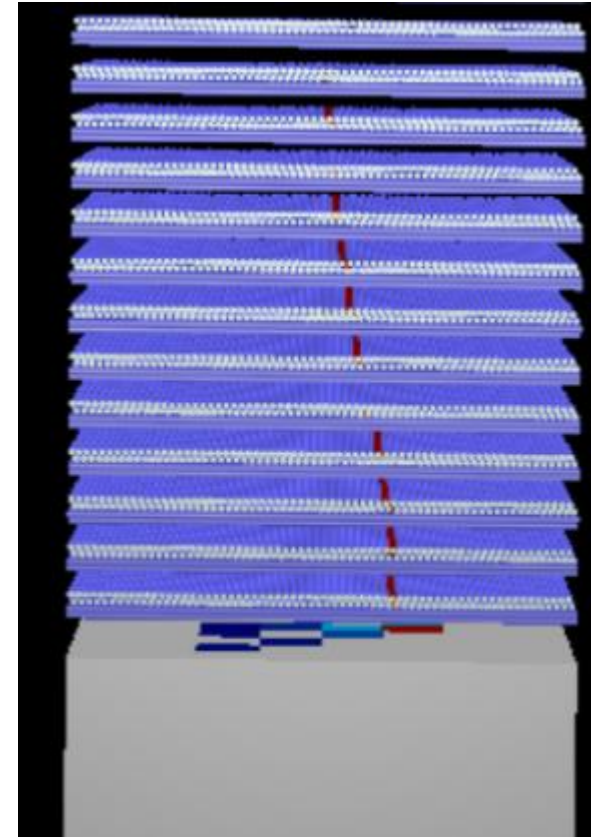
Built prototypes under testing

- The **High Energy cosmic-Radiation Detection** (HERD) facility is an international future space mission.
- We built several tile and bar prototypes for the HERD Plastic Scintillator Detector (PSD), and we are testing them in different configurations:
 - Test with protons and ions @ CERN
 - Test with CR in lab
 - Test with radioactive sources in lab
- We have also developed full customizable simulation tool based on GEANT4 that tracks every optical photon generated by scintillation in tiles/bars
 - The simulation purpose is to study the best tile/bar geometry and the best design for the SiPM-based readout system



Next generation of gamma-ray MeV telescopes

- The new generation gamma-ray experiments will cover the MeV range, which is still poorly explored
- Detectors need to operate both in the Compton and pair conversion regimes
- Ongoing: Optical simulation activities
 - *Prototypes of Compton-pair calorimeter imaging and tracking systems based on fiber tracker detectors coupled with scintillator crystals*



μ^+ @ 10 GeV perpendicular to the Tile
*Scintillation Yield reduced 1/100

- General G4 introduction
 - How to run the example B1
 - Build and run the code
 - *Run a simple simulation using the User Interface*
 - *Run a simple simulation using a macro*
- Make your own project
 - Build your project:
 - *Main & Physics list*
 - *Geometry*
 - *Write a macro*
 - Build your simulation: data management tools:
 - *Sensitive Detector*
 - *Hit collection*
 - *Run Action*
 - *Event Action*
 - Make your simulation (final exercise)

General G4 introduction

• Some information about G4 (GEometry ANd Traking)

- C++ Language
- Open Source
- It is a toolkit

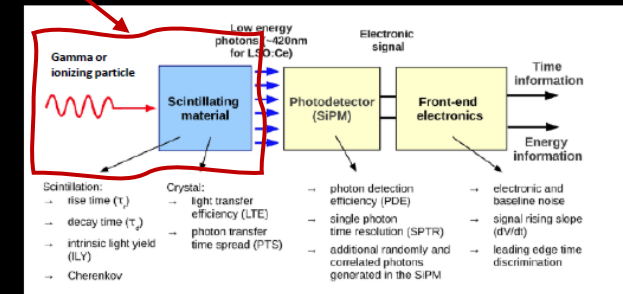
• Composing a G4 application:

• General Classes:

- *Main() file*
- *Sources files (*.cc)*
 - *Usually included in the src/ folder*
- *Header files (*.hh)*
 - *Usually included in the include/ folder*
- *Three classes are mandatory*
 - *PrimaryGenerationAction (.cc and .hh)*
 - *DetectorConstruction (.cc and .hh)*
 - *PhysicsList (.cc and .hh)*

• Detector composed of three building blocks

- Scintillating material
 - Absorption of radiation and conversion into UV or visible photon
- Photodetector (SiPM)
 - Detection of UV or visible photons
- Front-end electronics
 - Readout of electric signal



You must provide the necessary information to configure your simulation

- GEANT4 (**GE**ometry **ANd** **T**raking)
 - Transports a particle step-by-step, taking into account interactions with materials (and external electromagnetic fields), until the particle
 - *loses all its kinetic energy,*
 - *disappears due to an interaction,*
 - *reaches the boundary of the simulation volume*
 - Provides information for the user
 - *at the beginning and end of transport*
 - *at the end of each step in transport*
 - *at the time when the particle is in a sensitive volume of the setup*
 - *etc.*
 - You specify the set-up of an experimental system
 - *The software system automatically transports the particle you shoot into the defined system by simulating their interactions in matter based on the Monte Carlo method*
 - The heart of the simulation: the **Monte Carlo method**
 - *A method to search for solutions to a mathematical problem using statistical sampling with random numbers*

- Geant4 is a toolkit
 - You cannot “run” Geant4 out of the box
 - You must write an application, which uses Geant4 tools
 - There is no a concept of “Geant4 defaults”
 - You must provide the necessary information to configure your simulation
 - **many examples are distributed with Geant4**
- What a user must do:
 - Describe the experimental set-up
 - Input primary particles into the simulation
 - Decide which particles and physics models you wants to use out of those available in Geant4
 - *and the desired precision of the simulation (cuts to produce secondary particles to be further tracked)*

- Composing a G4 application:

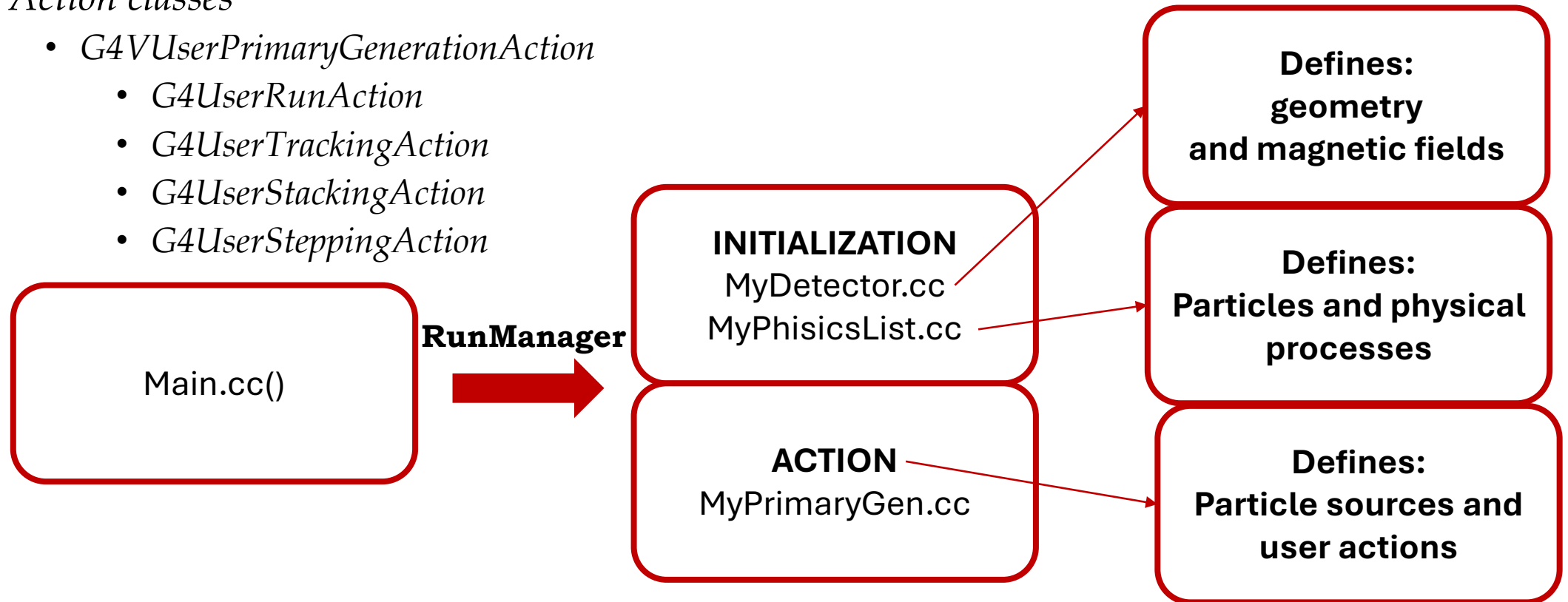
- General Classes:

- Initialization classes

- G4VUserDetectorConstrucion*
- G4VUserPhysicsList*

- Action classes

- G4VUserPrimaryGenerationAction*
- G4UserRunAction*
- G4UserTrackingAction*
- G4UserStackingAction*
- G4UserSteppingAction*



- What is necessary to built a complete simulation:
 - Detector Construction:
 - *Build your geometry, define your materials and eventually assign optical bulk and surface properties*
 - Physics List:
 - *“Activate” the physical processes that can occur during your simulation*
 - Hit collection:
 - *Define your “hit container”; i.e. which are the information that you want store during your simulation*
 - Sensitive Detector:
 - *Define your definition of hit; i.e. when you want to store the information*
 - Action classes:
 - G4UserRunAction:
 - *Define what happen at the begin and at the end of the **complete simulation***
 - *Create an output file with its structure (Beginning)*
 - *Close the output file (end)*
 - G4UserEventAction:
 - *Define what happen at the begin and at the end of **each event***
 - *Retrieve the information of each hit*
 - *Fill the output*

- Geant4 does not provide a standard main() function
- The main() is part of the user application
- In the main() the user **must**
 - instantiate G4RunManager(or his/her own derived class)
 - notify the G4RunManager mandatory user classes derived from
 - *G4VUserDetectorConstruction*
 - *G4VUserPhysicsList*
 - *G4VUserPrimaryGeneratorAction*
- The user may also instantiate in the main() function
 - optional user action classes
 - visualisation manager, UI session

main()

```
{
...
// Instantiate the default run manager
G4RunManager* runManager = new G4RunManager;

// Instantiate mandatory user initialization classes and notify
runManager
MyDetectorConstruction* detector = new MyDetectorConstruction;
runManager->SetUserInitialization(detector);
MyPhysicsList* physicsList = new MyPhysicsList;
runManager->SetUserInitialization(myPhysicsList);

// Instantiate mandatory user action classes and notify runManager
runManager->SetUserAction(new MyPrimaryGeneratorAction);

// Instantiate optional user action classes and notify runManager
MyEventAction* eventAction = new MyEventAction();
runManager->SetUserAction(eventAction);
MyRunAction* runAction = new MyRunAction();
runManager->SetUserAction(runAction);
...
}
```

A **run** is a collection of **events** that share the *same detector conditions*

- Detector and physics settings are frozen in a run

An **event** initially contains the *primary particles*; they are pushed into a stack and further processed



Example B1

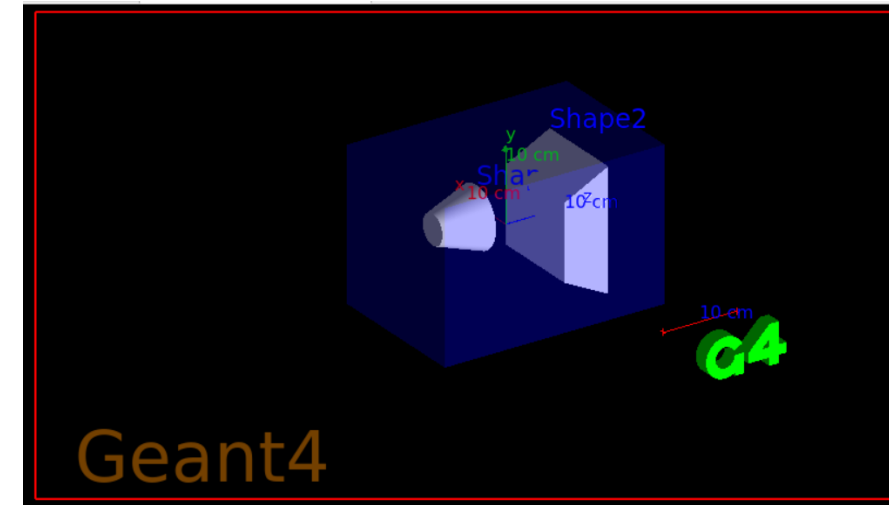
This example demonstrates a very simple application where an energy deposit is accounted in user actions and a dose in a selected volume is calculated.

1- GEOMETRY DEFINITION

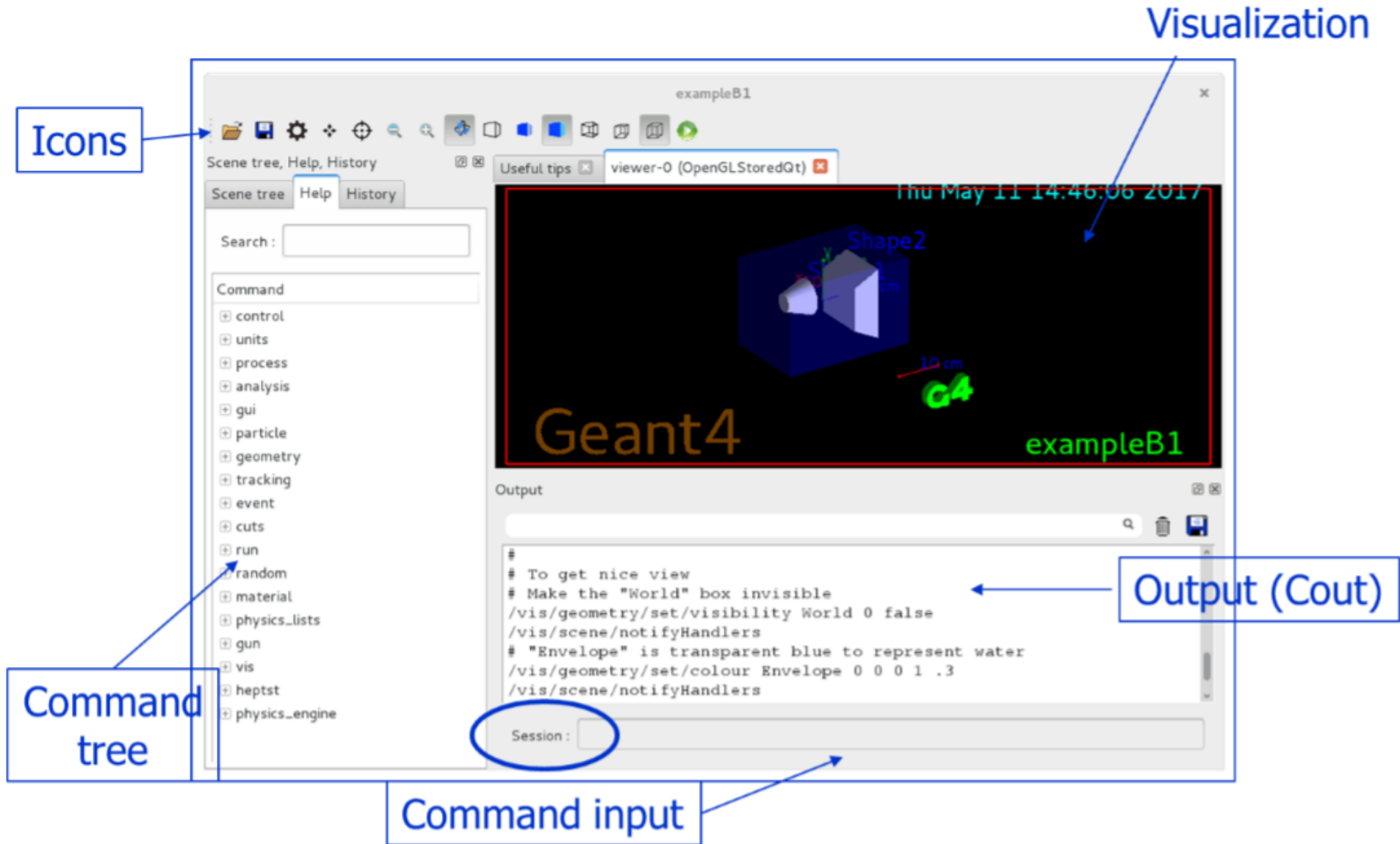
The geometry is constructed in the B1DetectorConstruction class. The setup consists of a an envelope of box shape containing two volumes: a spherical cone and a trapezoid.

In this example we use some common materials materials for medical applications. The envelope is made of water and the two inner volumes are made from tissue and bone materials.

The materials are created with the help of the G4NistManager class, which allows to build a material from the NIST database using their names. All available materials can be found in the Geant4 User's Guide for Application Developers, Appendix 10: Geant4 Materials Database.



User interface (Qt and OpenGL)



Explore Geant4 and B1 example

• How to compile a simulation

- Create a directory ProjectName_build and compile your source code inside the directory

- make:

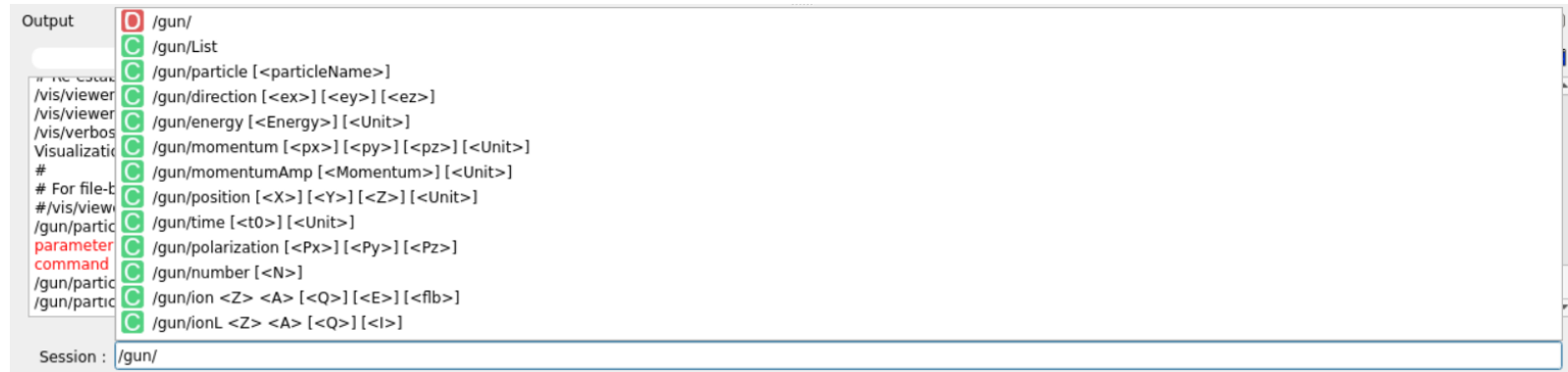
- cmake /Path\to\Project_source (Example: `cmake /home/dserini/geant4/B1`)
- Compile your source code with the command make
- Example: `>> cd $HOME/B1_build`
`>> cmake /home/dserini/geant4/B1`
`>> ./exampleB1`

- Run the executable: `./exampleB1`

• Run a simulation using the Geant4 GUI:

- G4ParticleGun (GUN) main commands (http://www.hep.ph.ic.ac.uk/~yoshiu/COMET/comet_g4HTMLdoc/_gun_.html)

```
/gun/particle proton
/gun/position 0 0 1 cm
/gun/direction 0 0 1
/gun/energy 1 GeV
/run/beamOn 1
```



The screenshot shows the Geant4 GUI with the G4ParticleGun (GUN) commands and their corresponding parameters. The commands are listed on the left, and the parameters are listed on the right. The parameters are color-coded: green for standard parameters, red for parameters that are not yet defined, and blue for parameters that are already defined.

Command	Parameter
/gun/	/gun/
/gun/List	/gun/List
/gun/particle [<particleName>]	/gun/particle [<particleName>]
/gun/direction [<ex>] [<ey>] [<ez>]	/gun/direction [<ex>] [<ey>] [<ez>]
/gun/energy [<Energy>] [<Unit>]	/gun/energy [<Energy>] [<Unit>]
/gun/momentum [<px>] [<py>] [<pz>] [<Unit>]	/gun/momentum [<px>] [<py>] [<pz>] [<Unit>]
/gun/momentumAmp [<Momentum>] [<Unit>]	/gun/momentumAmp [<Momentum>] [<Unit>]
/gun/position [<X>] [<Y>] [<Z>] [<Unit>]	/gun/position [<X>] [<Y>] [<Z>] [<Unit>]
/gun/time [<t0>] [<Unit>]	/gun/time [<t0>] [<Unit>]
/gun/polarization [<Px>] [<Py>] [<Pz>]	/gun/polarization [<Px>] [<Py>] [<Pz>]
/gun/number [<N>]	/gun/number [<N>]
/gun/ion <Z> <A> [<Q>] [<E>] [<flb>]	/gun/ion <Z> <A> [<Q>] [<E>] [<flb>]
/gun/ionL <Z> <A> [<Q>] [<I>]	/gun/ionL <Z> <A> [<Q>] [<I>]

Session : /gun/

- G4GeneralParticleSource (GPS) main commands (<http://geant4-userdoc.web.cern.ch/geant4-userdoc/UsersGuides/ForApplicationDeveloper/html/GettingStarted/generalParticleSource.html>):

- Some UI commands:
 - /run/verbose 1
 - *Sets how much output the run manager will print*
 - /run/initialize
 - *Initializes the run*
 - /run/beamOn 100
 - *Starts a run with 100 events*
 - /control/execute macroName.mac
 - *Run the commands in a macro file*
- A complete list of built-in command is available in the Application Developers Guide, Chapter 7

- General G4 introduction
 - How to run the example B1
 - Build and run the code
 - *Run a simple simulation using the User Interface*
 - *Run a simple simulation using a macro*
- Make your own project
 - Build your basic project:
 - *main & physics list*
 - *geometry*
 - *write a macro*
 - Build your simulation: data management tools:
 - *Sensitive Detector*
 - *Hit collection*
 - *Run Action*
 - *Event Action*
 - Make your simulation (final exercise)