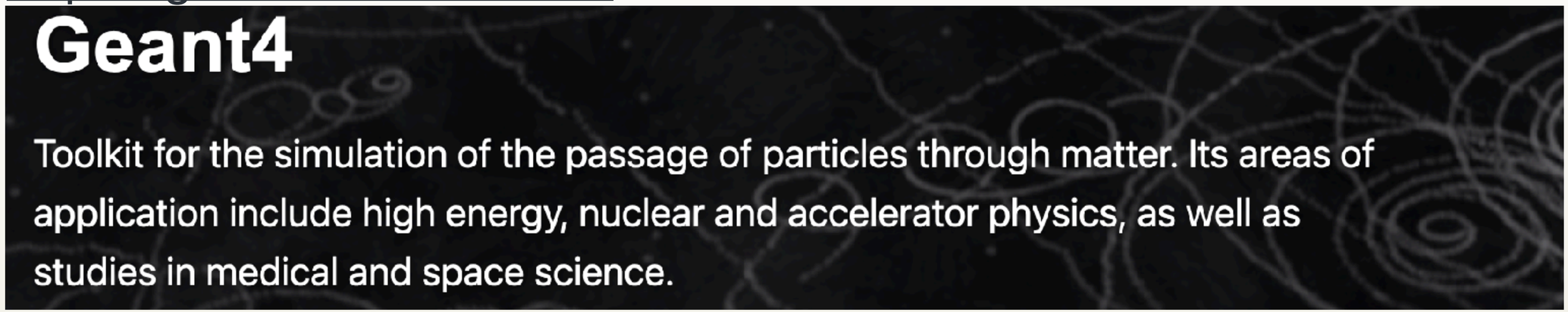
Geant4 simulation

Nam Tran
Boston University

3rd Hardware Camp for Fast and Low-Light Detection
ICISE, Quy Nhon, Vietnam

What is Geant4?

- https://geant4.web.cern.ch/

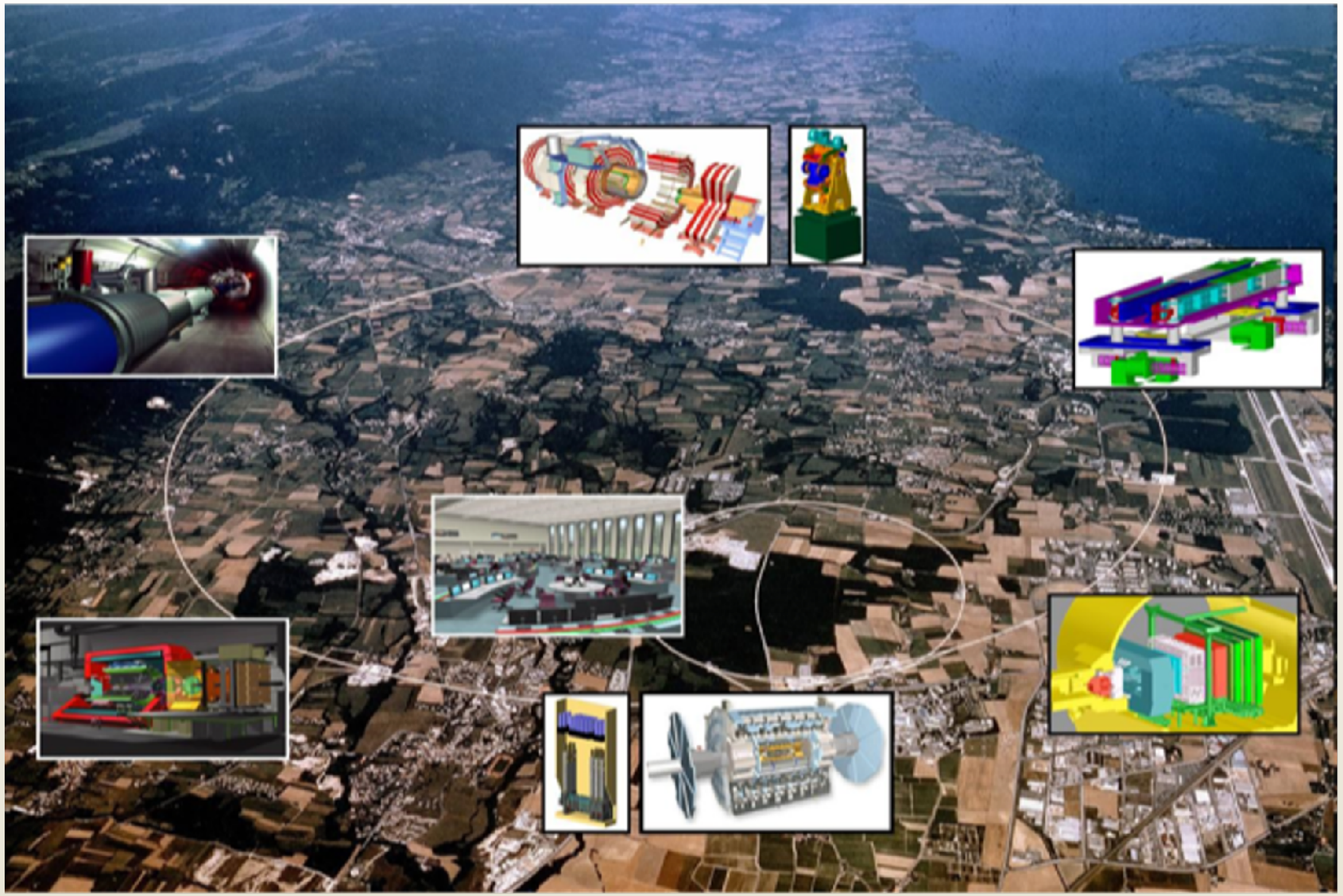


- A full set of libraries written in C++
  - Allow simulating user's detectors
  - Automatically transports the particles shot into the detector by simulating the particle interactions in matter based on the Monte Carlo method
- Initially developed for the simulation of next generation HEP detectors (ATLAS, Alice, CMS, LHCb…)
  - used widely today for the simulation of the current generation detectors and also in the space and medical physics

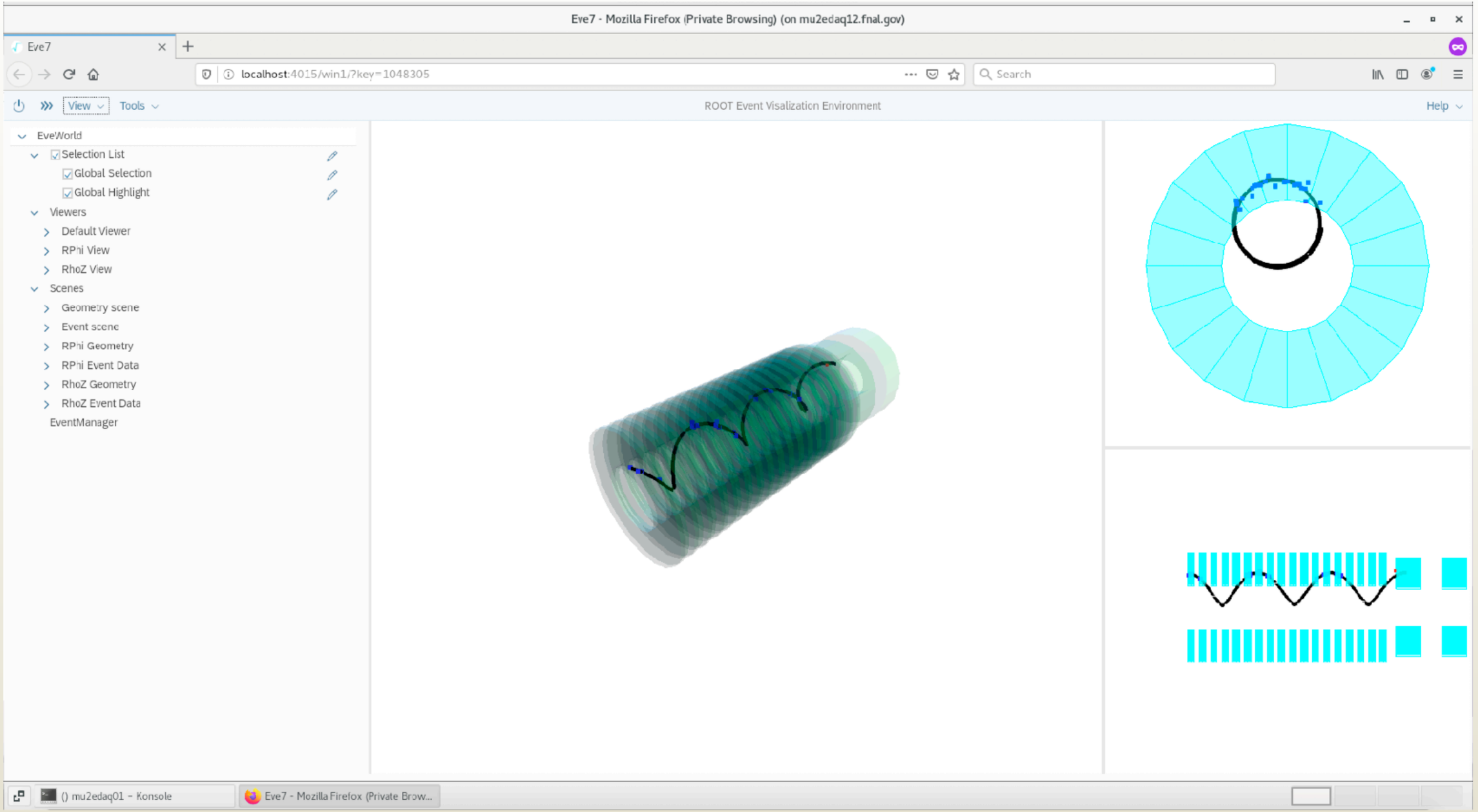
Monte Carlo method

- Mathematical approach using a sequence of random numbers to solve a problem
- In Geant4:
  - if particles interaction models are known, MC can be used to calculate the parameters of the motion equations in a given configuration
  - particles are tracked one-by-one, step-by-step
  - and, after a reasonable number of particles, the correct information can be extracted

Geant4 in CERN's experiments

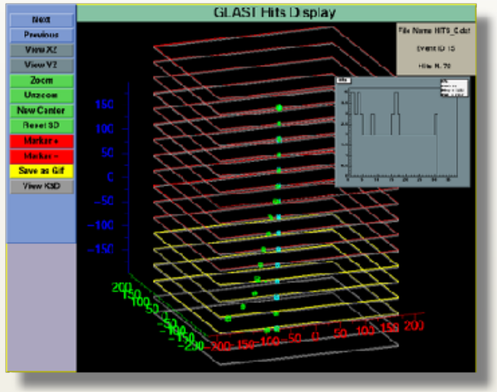


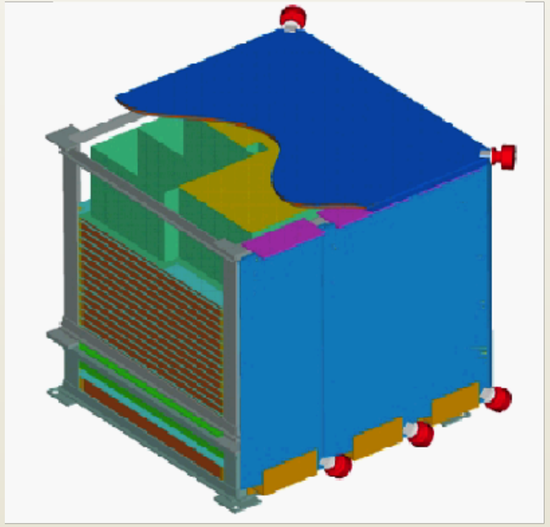
Geant4 in Mu2e experiment

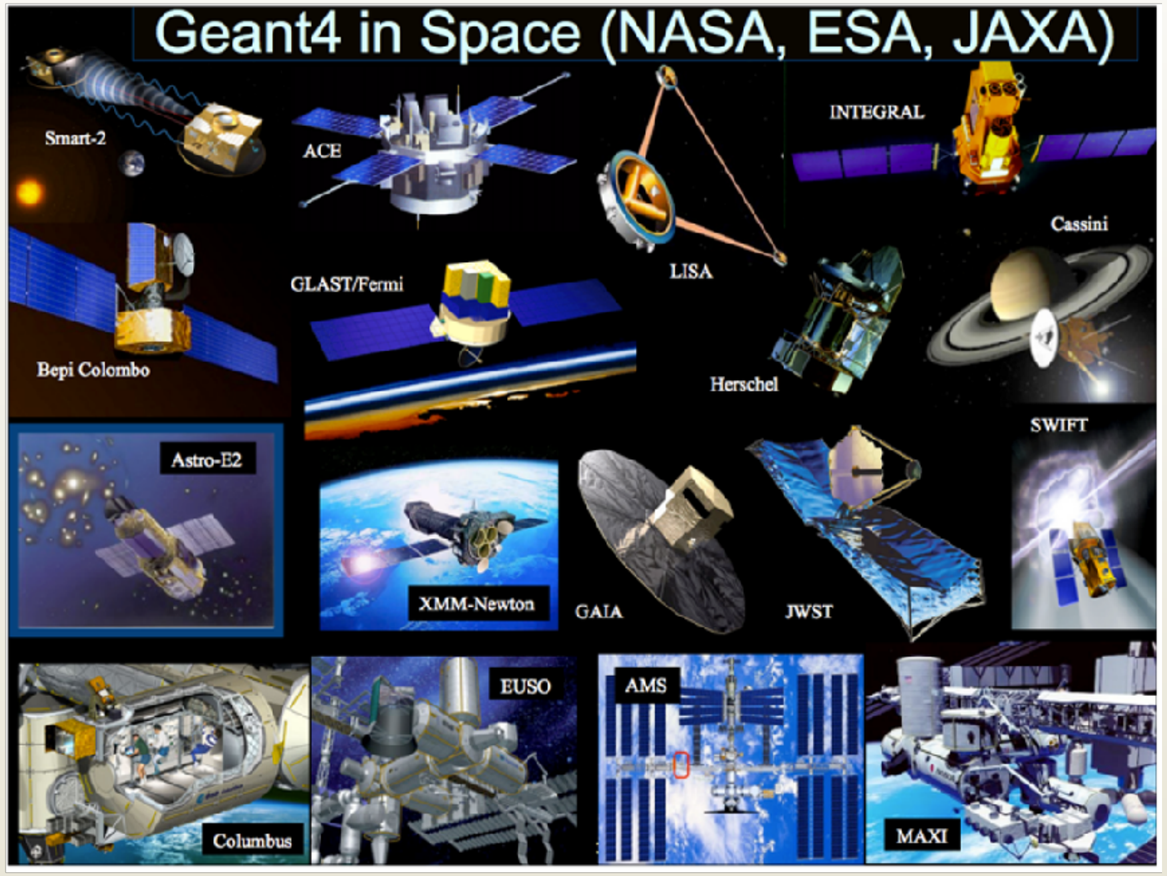


Geant4 in space applications

- Satellites (γ astrophysics, planetary sciences)
- Typical telescope:
  - Tracker
  - Calorimeter
  - Anticoincidence

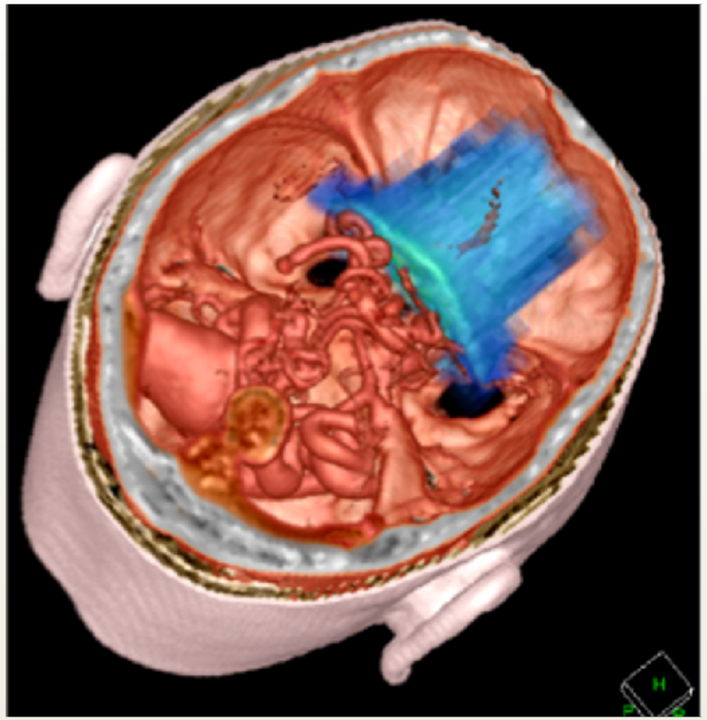


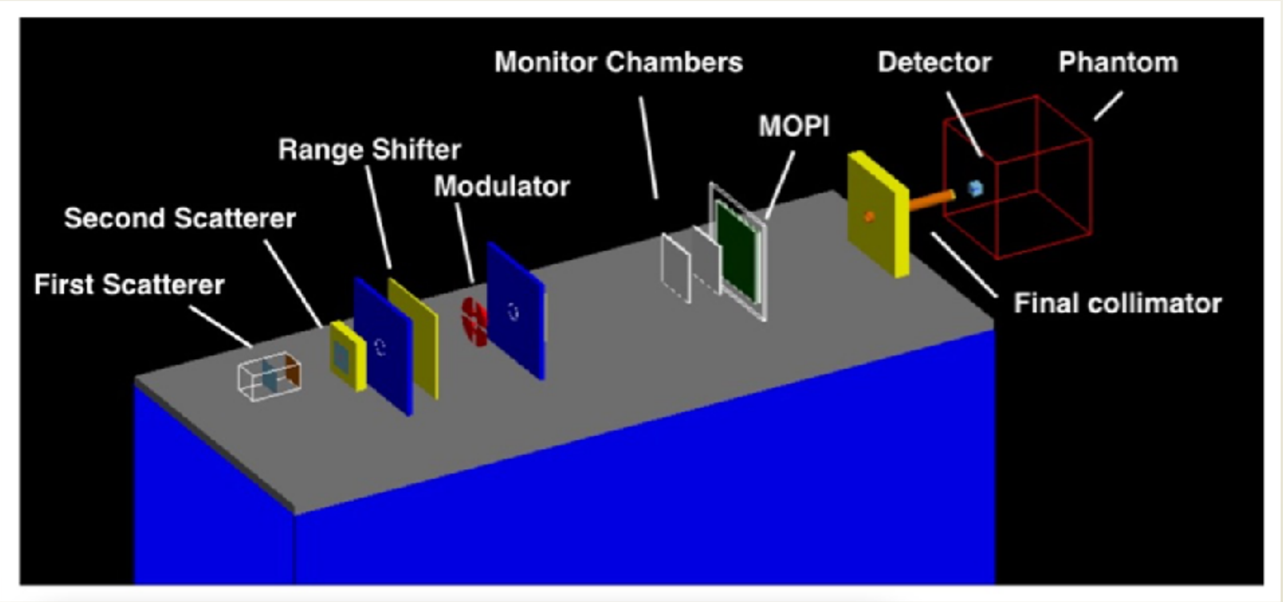


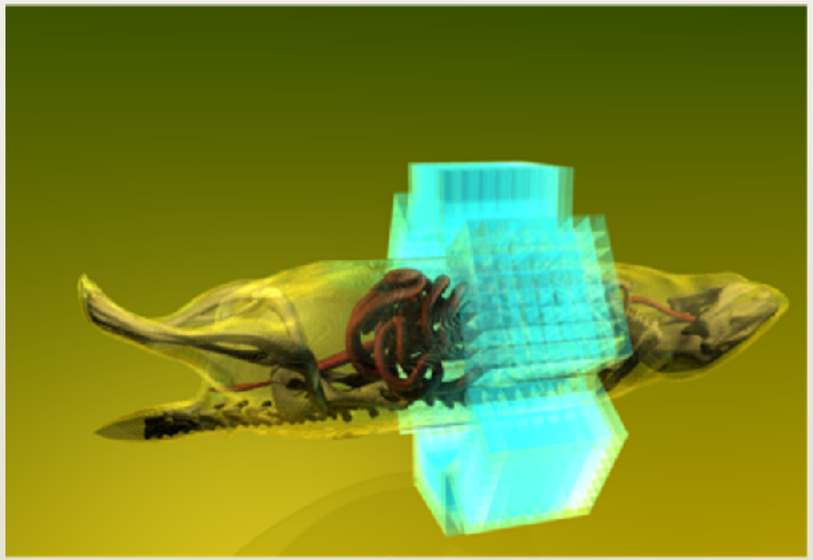


Geant4 in medical applications

- Major use cases:
  - Beam therapy
  - Brachytherapy
  - Imaging
  - Irradiation study
  - Nuclear medicine and radioisotopes
  - Biological damage







Why do we simulate?

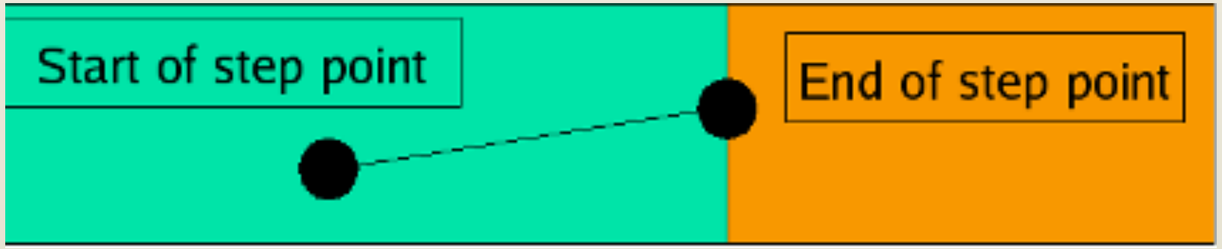
- Simulation plays a fundamental role in various domains and phases of an experimental physics project :
  - design of the experimental set-up,
  - evaluation and definition of the potential physics output of the project,
  - evaluation of potential risks to the project,
  - assessment of the performance of the experiment,
  - development, test and optimization of reconstruction and physics analysis software,
  - contribution to the calculation and validation of physics results…

Basic terminologies

- A run
  - the largest unit of simulation in Geant4
  - is a collection of events which are produced under identical conditions
    - within a run, the user cannot change the detector or apparatus geometry, nor the physics process settings
- An event:
  - at the start of processing: contains primary particles (from a generator, a particle gun, ...), which are pushed onto a stack.
  - during the processing, each particle is popped from the stack and tracked. When the stack is empty, processing of the event is over
  - at the end of processing, has the following objects: list of primary vertices and particles (the input), hits collections, trajectory collections (optional), digitization collections (optional)

Basic terminologies

- A track
  - a snapshot of a particle within its environment as the particle moves.
  - has a lifetime
    - created by a generator or a physics process (e.g. decay)
    - deleted when it leaves the World mother volume, disappears (particle decays or is absorbed), goes to zero energy and no “at rest” process is defined or the user kills it.
  - Is not persistent (no track object survives the end of an event)
- A step
  - is the basic unit of simulation
  - has two points (pre-step, post-step)
  - contains the incremental particle information (energy loss, elapsed time, etc.). Each point contains volume and material information



Geant4 as a toolkit

- Consequences:
  - There are not such concepts as “Geant4 defaults”
  - You must provide the necessary the necessary information to configure your simulation
  - You must choose the Geant4 tool to use
- There are lots of guidances from many examples:
  - Basic examples: overview of the Geant4 tools
  - Extended examples: showing specific Geant4 functionalities
  - Advanced examples: Geant4 tools in real-life applications

What to do in order to have a running Geant4 application

- 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 the Geant4 kernel to control your simulation
  - To visualise your simulation set-up and particles
  - To produce histograms, tuples, etc. to be further analysed

Files composing a Geant4 application

- `main()` file
- Sources files (`*.cc`)
  - usually included in the `src/` folder
- Header files (`*.hh`)
  - usually included in the `include/` folder
- Three classes are must-have, each typically in a dedicated header/source pair of files
  - The `PrimaryGeneratorAction` (`.cc` and `.hh`)
  - The `DetectorConstruction` (`.cc` and `.hh`)
  - The `PhysicsList` (`.cc` and `.hh`)

The `main()`

- Geant4 does not provide a `main()` file
  - Geant4 is a toolkit!
  - The `main()` is part of the User application
- In his/her `main()`, the user must:
  - Construct the `G4RunManager`
  - Notify the `G4RunManager` the mandatory user classes derived from:
    - `runManager -> SetUserInitialization (new MyApplicationDetectorConstruction)`
- The user MAY define in his/her `main()`:
  - Optional user action classes
  - VisManager, (G)UI session
- The user has also to take care of retrieve and save the relevant information from the simulation (Geant4 will not do that by default)
- Remember to delete the `G4RunManager` at the end

Example of a main() file

```
int main(int argc, char **argv) {
  // Construct the default run manager
  G4RunManager *runManager = new G4RunManager;

  // Set mandatory initialization classes
  // Detector construction
  runManager->SetUserInitialization(new DetectorConstruction());
  // Physics list
  G4VModularPhysicsList *physicsList = new QBBC;
  physicsList->SetVerboseLevel(0);
  runManager->SetUserInitialization(physicsList);

  // User action initialization
  runManager->SetUserInitialization(new ActionInitialization());



  // Job termination
  delete runManager;
}
```

DetectorConstruction

```
E  32 #include "DetectorConstruction.hh"   ■ 'DetectorConstruction.hh' file not found
   31
   30 #include "G4Box.hh"
   29 #include "G4LogicalVolume.hh"
   28 #include "G4MultiFunctionalDetector.hh"
   27 #include "G4NistManager.hh"
   26 #include "G4PSDoseDeposit.hh"
   25 #include "G4PSEnergyDeposit.hh"
   24 #include "G4PVPlacement.hh"
   23 #include "G4PhysicalConstants.hh"
   22 #include "G4RotationMatrix.hh"
   21 #include "G4RunManager.hh"
   20 #include "G4SDManager.hh"
   19 #include "G4SystemOfUnits.hh"
   18 #include "G4Transform3D.hh"
   17 #include "G4Tubs.hh"
   16 #include "G4VPrimitiveScorer.hh"
   15 #include "G4VisAttributes.hh"
   14
   13 //....oooOOOOOooo.......oooOOOOOooo.......oooOOOOOooo.......oooOOOOOooo.....
   12
E  11 DetectorConstruction::DetectorConstruction()      ■ Use of undeclared identifier 'DetectorConstruction'
   10     : G4VUserDetectorConstruction(), fCheckOverlaps(true) {
    9   DefineMaterials();
    8 }
    7
E   6 DetectorConstruction::~DetectorConstruction() {}      ■ Use of undeclared identifier 'DetectorConstruction'
    5
E   4 void DetectorConstruction::DefineMaterials() {}      ■ Use of undeclared identifier 'DetectorConstruction'
    3
E   2 +-- 58 lines: G4VPhysicalVolume *DetectorConstruction::Construct() {·····
  121
E  90 void DetectorConstruction::ConstructSDandField() {      ■ Use of undeclared identifier 'DetectorConstruction'
    1   G4SDManager::GetSDMpointer()->SetVerboseLevel(0);
    2
    3   G4MultiFunctionalDetector *scSD = new G4MultiFunctionalDetector("scSD");
    4   G4SDManager::GetSDMpointer()->AddNewDetector(scSD);
    5   G4VPrimitiveScorer *primitiv1 = new G4PSEnergyDeposit("edep");
    6   scSD->RegisterPrimitive(primitiv1);
    7   SetSensitiveDetector("logicSc", scSD);
    8 }
```

Physics list

- Geant4 doesn’t have any default particles or processes
  - Partially true: there is no default, but there are a set of "ready-for-use" physics lists released with Geant4, tailored to different use cases
  - Different sets of hadronic models (depending on the energy scale and modelling of the interactions)
- Different options for neutron tracking
  - Do we need (CPU-intensive) description of thermal neutrons, neutron capture, etc?
- Different options for electromagnetic physics
  - Do you need (CPU-intensive) precise description at the low-energy scale (< 1 MeV)?
    - E.g. fluorescence, Doppler effects in the Compton scattering, Auger emission, Rayleigh diffusion

Primary generator action

- For each event, the user must define all details of initial particle
  - derive a concrete class from the `G4VUserPrimaryGeneratorAction` abstract base class
- Several ways to do this:
  - G4ParticleGun
  - G4HEPEvtInterface
  - G4GeneralParticleSource

Particle Gun	General Particle Source	HEP event interface
Simple and native	Powerful	Doesn't give place of primary particle
Shoots one track at a time	Controlled by UI commands	Interaction point must be set by user
Easy to handle	Capability of shooting particles from a surface of a volume and of randomising kinetic energy, position, direction, following (complicated) user specified distribution	

Summary

- Geant4 is a powerful and widely used tool in particle physics!
- Remember that 3 required pieces needed to run a Geant4 simulation:
  - Detector construction
  - Physics list
  - Primary generator