GEometryANdTracking

Simulation toolkit in C++:

- Variety of geometries → choose your own setup
- Variety of materials → choose your own materials
- Variety of particles → choose particle type energy position direction
- Variety of physics processes → available physics models, cuts

Open source:

- Code
- Manuals
- Examples

http://geant4.web.cern.ch

Transaction on Nuclear Science 53, 270
Geometry

Very complex geometries can be described at three levels:

- **Solid**: shapes and dimensions, boolean operations

- **LogicalVolume**: materials, sensitivity, mother and daughter volumes, visualization etc

- **PhysicalVolume**: position in space and rotation
  
a logical volume can be placed multiple times originating different physical volumes
  
→ single or repeated placements (replicas, parameterizations)
Physics

- electromagnetic interactions for all particles
- inelastic interactions
- elastic scattering
- capture
- decay of unstable particles

plug&play Geant4 physics list
$G4SOURCE/ source/physics_lists/lists

class MyPhysicsList: public G4VUserPhysicsList {
public:
MyPhysicsList();
~MyPhysicsList();
void ConstructParticle();
void ConstructProcess();
void SetCuts();
}

user can implement the methods to define particles processes and cuts (range based) on
generation of secondaries ex. delta rays from ionization, or gamma from bremsstrahlung
Tracking

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
Geant4 example: compile

/home/isapp/geant4/geant/share/Geant4-10.1.3/examples/

- **Source the geant4 script:**
  
  ```
  $ source /home/isapp/geant4/geant/bin/geant4.sh
  ```

- **copy the example source code somewhere $path**

  ```
  $ cp -r /home/isapp/geant4/geant/share/Geant4-10.1.3/examples/basic/B2 $path-to-your-dir
  $ cd $path-to-your-dir
  $ mkdir B2-build
  $ cd B2-build
  $ cmake -DG4eant4_DIR=/home/isapp/geant4/geant/lib /home/isapp/path-to-your-dir/B2
  $ make
  ```
Geant4 example: run

- $ ./exampleB2a
- /gun/particle e-
- /gun/energy 300 MeV
- /run/beamOn 1

here you go!
writing your G4 application
writing your G4 application

Physics:
- use the Geant4 standard provided physic lists:
  \$G4SOURCE/ source/physics_lists/lists
- build/taylor our own models
writing your G4 application

Your detector geometry:
- shapes
- materials
- volumes
- placements
- sensitivity
writing your G4 application

The source of radiation:
- simple gun
- generic source
- external file
writing your G4 application

- G4UserActions
- G4Hit/G4Digi read out and digitization
- G4Visualization
- Analysis

etc etc
G4VRecipe

writing an application you must have:
- a class derived from G4VUserDetectorConstruction
  definition of your detector geometry
- a class derived from G4VUserPhysicsList
  selection of the physics processes
- a class derived from G4VUserPrimaryGeneratorAction
  producing primary events

optional classes inherit from:
- G4UserRunAction
- G4UserEventAction → to be done at the beginning/end of an event
- G4UserTrackingAction
- G4UserStackingAction
- G4UserSteppingAction
example of Main()

{
    // Construct the default run manager
    G4RunManager* runManager = new G4RunManager;

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

    // Set mandatory user action class (Primary Generator)
    runManager->SetUserAction(new MyPrimaryGeneratorAction);

    // Set optional user action classes (e.g. only a few of them)
    MyEventAction* eventAction = new MyEventAction;
    runManager->SetUserAction(eventAction);
    MyRunAction* runAction = new MyRunAction;
    runManager->SetUserAction(runAction);

    delete runManager;
}
Generic Geant Simulation
thanks to Nicola Mori

• Generic implementation of G4 user classes
  – Hits, event actions, run actions,…
  – Persistence on Root files
  – Extensible with plugins
• Speeds-up the development of a G4 Monte Carlo simulation
  – Only geometry has to be created
• Code:
  – https://baltig.infn.it/mori/GGSSoftware
• User’s guide:
  – https://wizard.fi.infn.it/ggs/manual/
Auger SD Simulation

- CORSIKA: primary CR propagated in the Atmosphere
  output → particle distribution at the ground

- Geant4: particles hit Water Cherenkov detectors →
  emission of light & tracking incl. reflections → detection in
  PMTs
  output → photoelectron time distribution

- electronics and PMT: response to single ph.el. to transform
  collected charge into a pulse in V → ADC
From p.el.\( (t) \) to ADCs