Simulation of the nuStorm Detector

August 9th, 2021

CERN Summer Student Programme

Hitoshi Baba

The nuSTORM Experiment

- Primary goal:

Form a better understanding of neutrino-nucleus interactions

- Expected to reduce systematic uncertainties for neutrino oscillation experiments

→Contribute to the discovery of leptonic CP-invariance violation

- Potentially make some big discoveries about nucleus structure

- Sterile neutrino search

- Much more to come

The nuSTORM Experiment

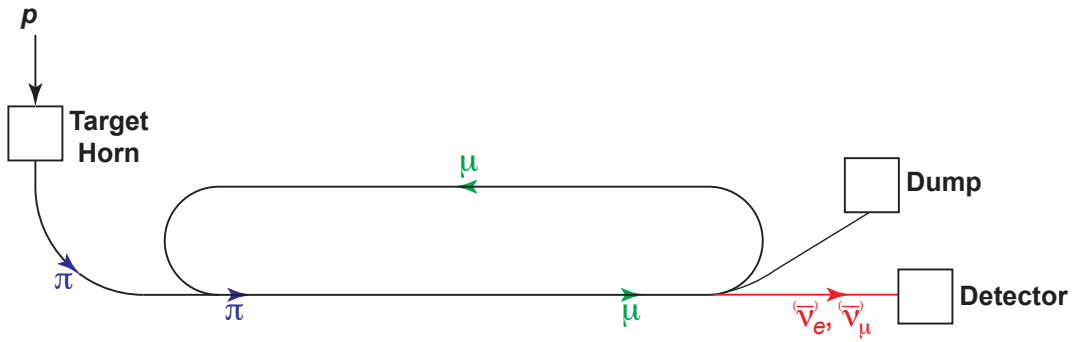


Fig. 1: Schematic of the nuSTORM neutrino-beam facility.

Figure borrowed from the nuSTORM Executive Summary

Neutrino Interactions

- Many different interaction ν A interaction processes including:
  - elastic scattering $\nu + A \rightarrow l + A$
  - Quasi-elastic scattering $\nu + A \rightarrow l + A'$
  - inelastic scattering $\nu + A \rightarrow l +$(a bunch of hadrons)
  - A lot of other processes(e.g., multi-pion resonance-production?)

Dominant at energies <= 2 GeV

Dominant at high energy?

→ Our detector should be capable of tracking each of the particle tracks and energies

The nuSTORM Detector

- Currently some candidates are being considered

→Sampling calorimeters vs. homogeneous calorimeters, solid vs. liquid, etc.

- For my programme, I have simulated a sampling calorimeter using Geant4

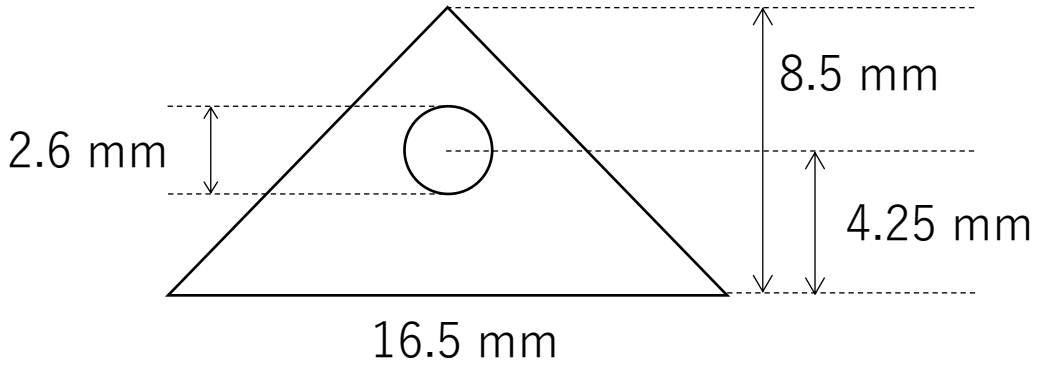
The Codes that I have written

```
source/
├── CMakeLists.txt
├── include
│   ├── stepHit.hh
│   ├── stepSD.hh
│   ├── trianglesActionInitialization.hh
│   ├── trianglesConstants.hh
│   ├── trianglesDetectorConstruction.hh
│   ├── trianglesEventAction.hh
│   ├── trianglesMagneticField.hh
│   ├── trianglesPrimaryGeneratorAction.hh
│   ├── trianglesRunAction.hh
│   ├── trianglesScintHit.hh
│   └── trianglesScintSD.hh
├── init_vis.mac
├── src
│   ├── stepHit.cc
│   ├── stepSD.cc
│   ├── trianglesActionInitialization.cc
│   ├── trianglesDetectorConstruction.cc
│   ├── trianglesEventAction.cc
│   ├── trianglesMagneticField.cc
│   ├── trianglesPrimaryGeneratorAction.cc
│   ├── trianglesRunAction.cc
│   ├── trianglesScintHit.cc
│   └── trianglesScintSD.cc
├── triangles.cc
│
└── vis.mac
```

Sorry for the crappy name “triangles” :)

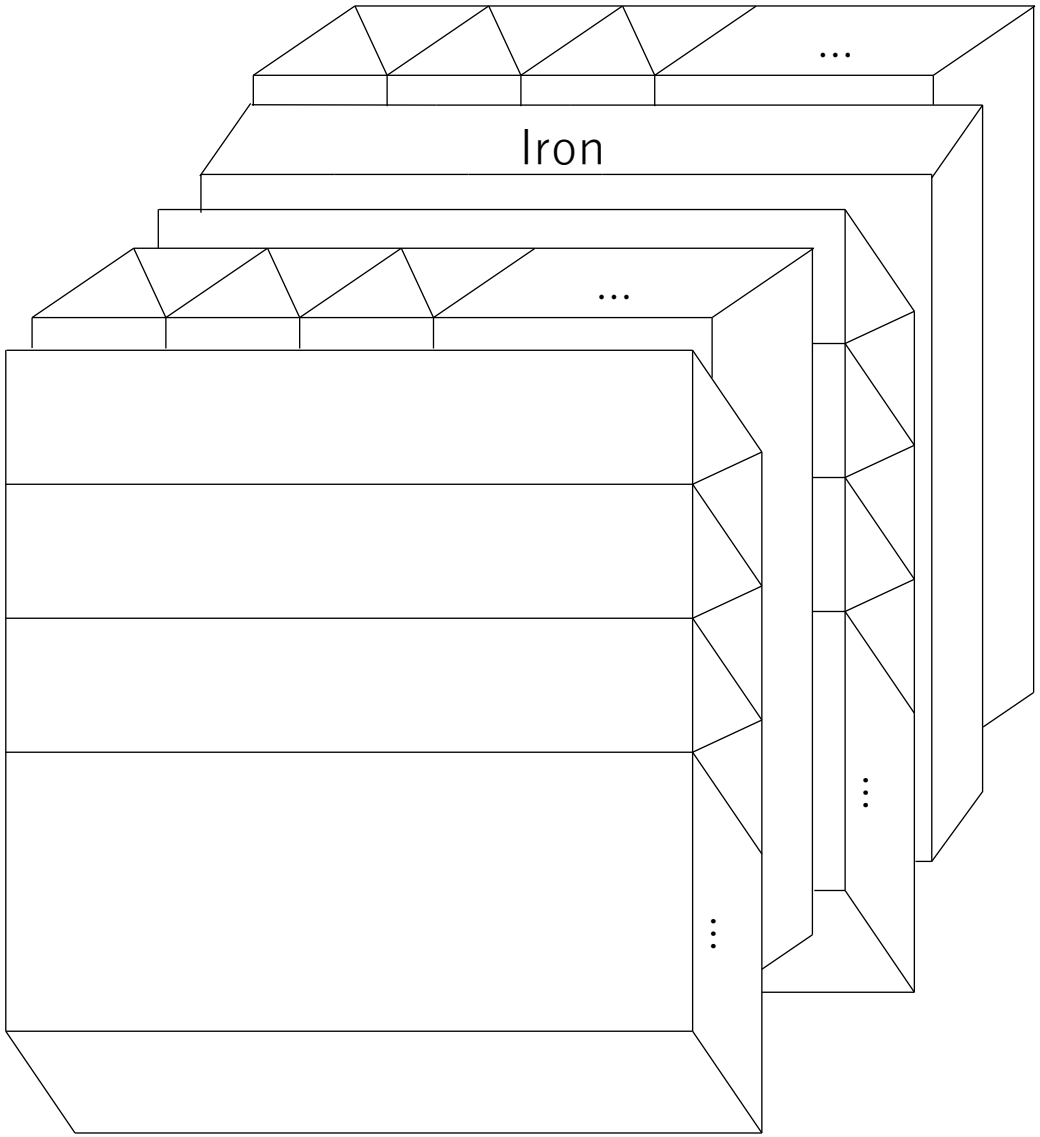
Detector Construction – Basic Idea

Bunch of plastic scintillator strips with triangular cross sections



Cross section of a scintillator strip

Detector Construction – Basic Idea



(X→Y→X→Iron→Y→X→Y→Iron) x 210

480 scintillator strips per detector plane

Detector Construction – trianglesDetectorConstruction class

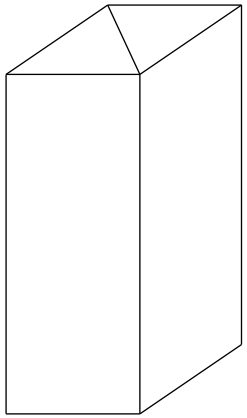
Methods:

- Construct() →Construction of the detectors
- ConstructSDandField() →Adding the magnetic fields and sensitive detectors to the logical volumes

Detector Construction – trianglesDetectorConstruction class

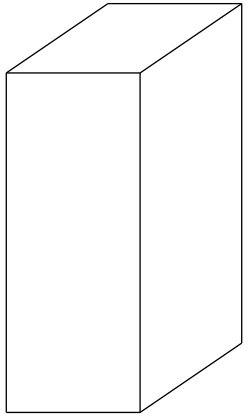
Hierarchy of the logical volumes (the names in parentheses are the names of the Logical Volume)

Placed two scintillator strips…
(Scintillator 1, Scintillator 2)



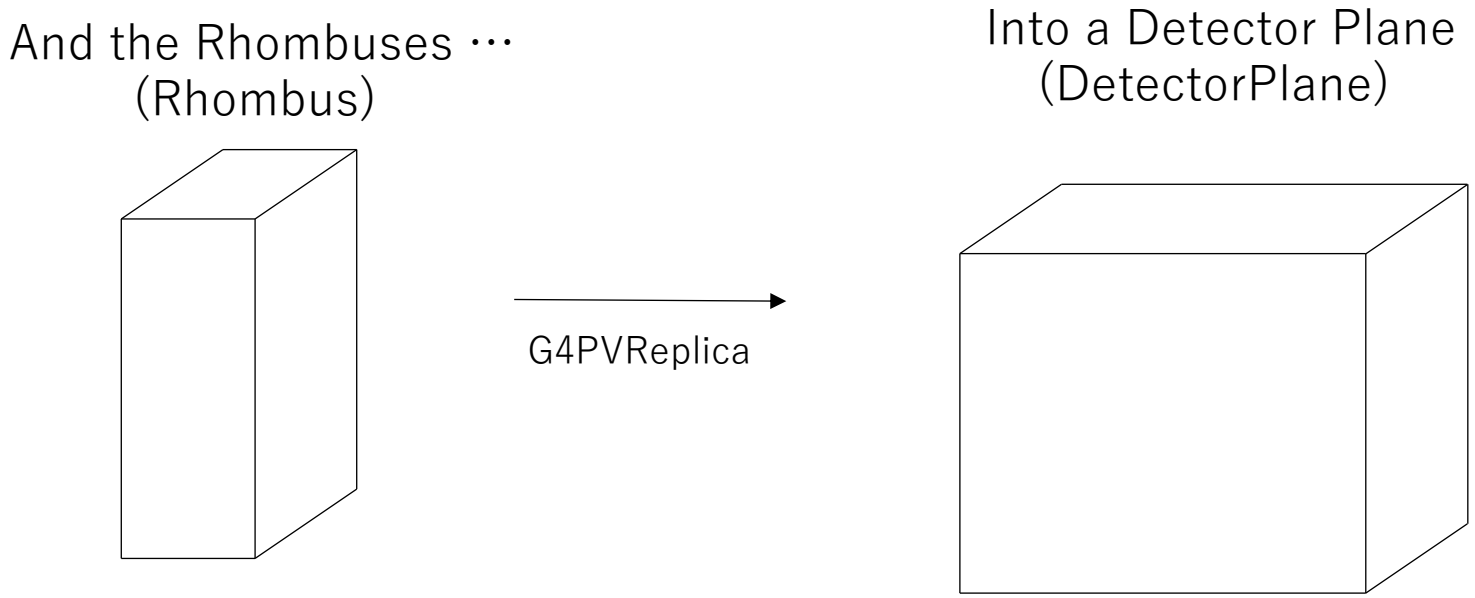
G4PVPlacement

Into a Rhombus
(Rhombus)



Detector Construction – trianglesDetectorConstruction class

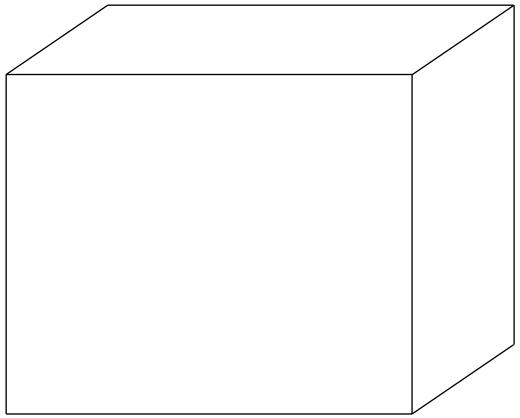
Hierarchy of the logical volumes (the names in parentheses are the names of the Logical Volume)



Detector Construction – trianglesDetectorConstruction class

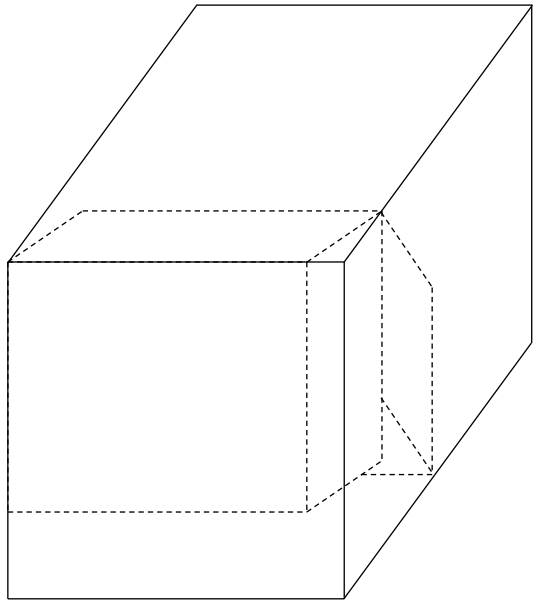
Hierarchy of the logical volumes (the names in parentheses are the names of the Logical Volume)

Prepared six of the Detector Planes
+
Two iron plates



G4PVPlacement

And placed them inside a module (Module)



The detector plane is a trapezoid, but the module is a box.

Detector Construction – trianglesDetectorConstruction class

Hierarchy of the logical volumes (the names in parentheses are the names of the Logical Volume)

Finally placed the modules into a logical volume called WholeDetector using G4PVReplica and placed the WholeDetector in the World.

Detector Construction – trianglesDetectorConstruction class

Sensitive detectors:
Scintillator1, Scintillator2.

Magnetic Field:
1.5 T, inside the iron plates, perpendicular to the beam line.
The trianglesDetectorConstruction class assigns the magnetic field to a logical volume.
The definition of the magnetic field is done in the trianglesMagneticField class

Detector Construction – Considerations

- It may be easier/better to use G4PVParametrization instead of the way I have constructed the detector geometry.

→ This way there would be no need to make 2 different scintillator strips

- For the material of the components, I have used G4_Fe and G4_PLASITC_SC_VINYLTOLUENE, from the Geant4 database/NIST Compounds. If we want to think of other materials with specific properties, we will have to code it on our own.

Details of the materials can be found in https://geant4-userdoc.web.cern.ch/UsersGuides/ForApplicationDeveloper/html/Appendix/materialNames.html

- More complex geometry?

Sensitive Detectors – Hits classes and SD classes

trianglesScintHit class, trianglesScintSD class, stepHit class and stepSD class

The Hit class defines (declares) the variables to store information (e.g., energy, position, etc.)
The SD class defines when to create those hits, and what to put in those variables.

“trianglesScint” records the energy deposited to each of the scintillator strips throughout an event.
→Works as a calorimeter

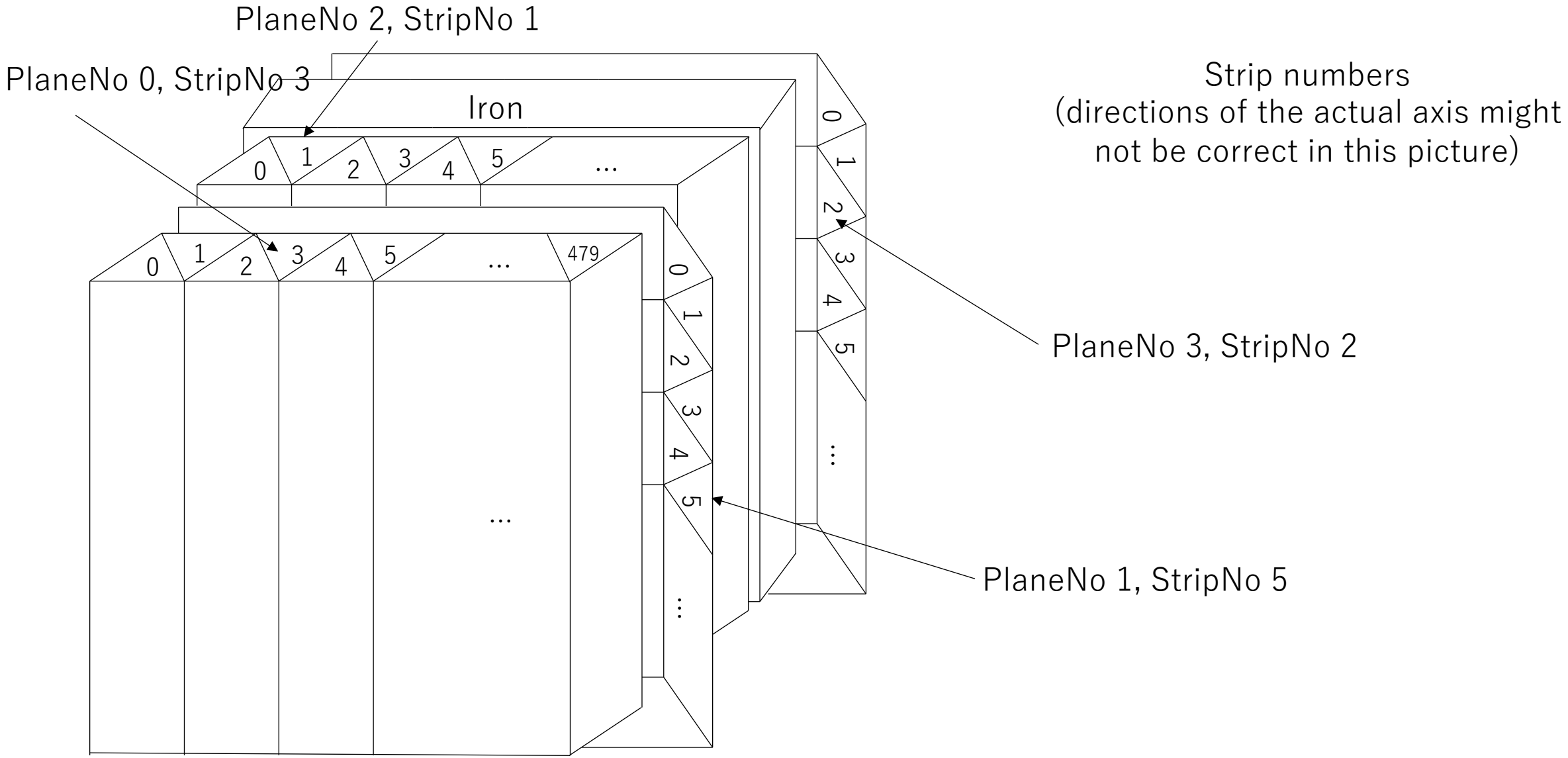
“step” records energy, position, momentum, and strip No. in which the step took place.
→Works as a tracking detector

Sensitive Detectors – Calorimeter Part

trianglesScintHit class, trianglesScintSD class

- A Hit stores strip no., and energy deposited to the strip throughout the event.
- A Hit is created at the beginning of the Event
- Deposited energy is added every step

Sensitive Detectors – Calorimeter Part



Sensitive Detectors – Tracking Detector Part

stepHit class, stepSD class

- A Hit stores strip no., energy deposited, momentum and time that the step took place in.
- A Hit is created every time a step happens in a scintillator strip.

Recording the results – Event Action and Run Action

trianglesEventAction class and trianglesRunAction class

- At the beginning of a run, the analysis files are created (Run Action)
- There the ntuple columns are created
- The ntuple is filled at the end of the Event (Event Action)
- For the calorimeter part, only the hits which had a non-zero energy deposit were recorded.
- For the tracking detector part, all the hits created were recorded.

Recording the results – The ntuple

- Each row stands for one event.
- Some columns (e.g., event no., position/momentum/energy of the primary particle) have one variable.
- Others (e.g., strip no., position/momentum/energy deposit of each step, etc.) are expressed as a long vector.

Some Other Components

triangles.cc	Main part of the application
trianglesActionInitialization	Things related to the initialization. Creates the primary particle and initializes the Event/Run actions.
trianglesPrimaryGeneratorAction	Specifies the details of the primary particle (Particle type, energy, momentum, etc.)
trianglesConstants.hh	Some constants (e.g., number of strips per plane, number of modules)
trianglesMagneticField	Details (e.g., strength and direction) of the magnetic field.

Some Other Components

vis.mac	Visualization settings for running the application
init_vis.mac	Macro for initializing the application. This macro tells the application to initialize the kernel and execute vis.mac
run_100eve.mac	Macro for executing a run with 100 events.

When you start running the application by typing in “./triangles” in your terminal, init_vis.mac will be executed automatically (this is specified in triangles.cc).

How to Run – generating the executable

1. Create a build directory (outside the source directory), and cd into it.
2. Run “cmake (path to source directory)”

```
$ cmake ../source/
```

3. Then run “make”

If all goes well, this should produce an executable file named “./triangles”

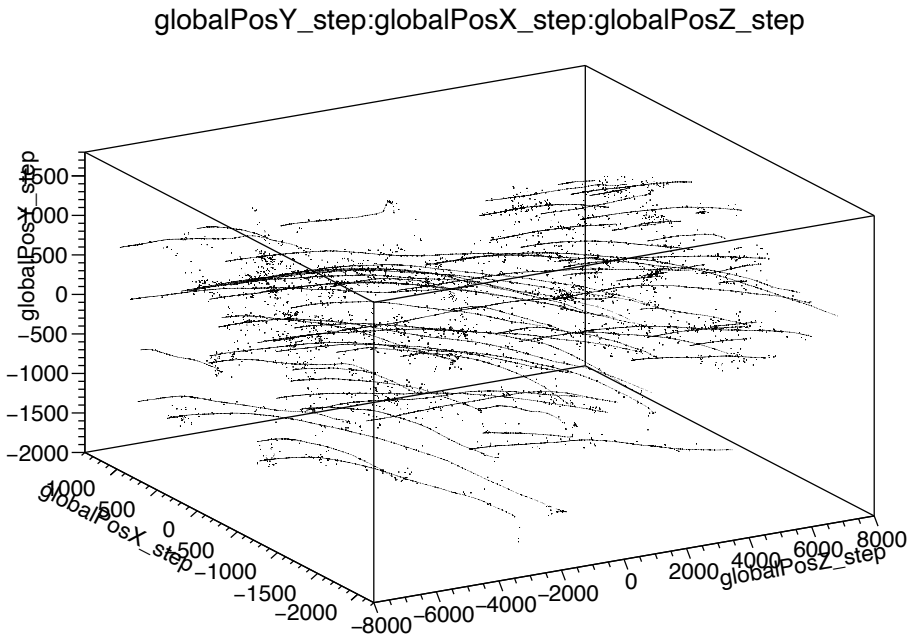
How to Run – running a simulation

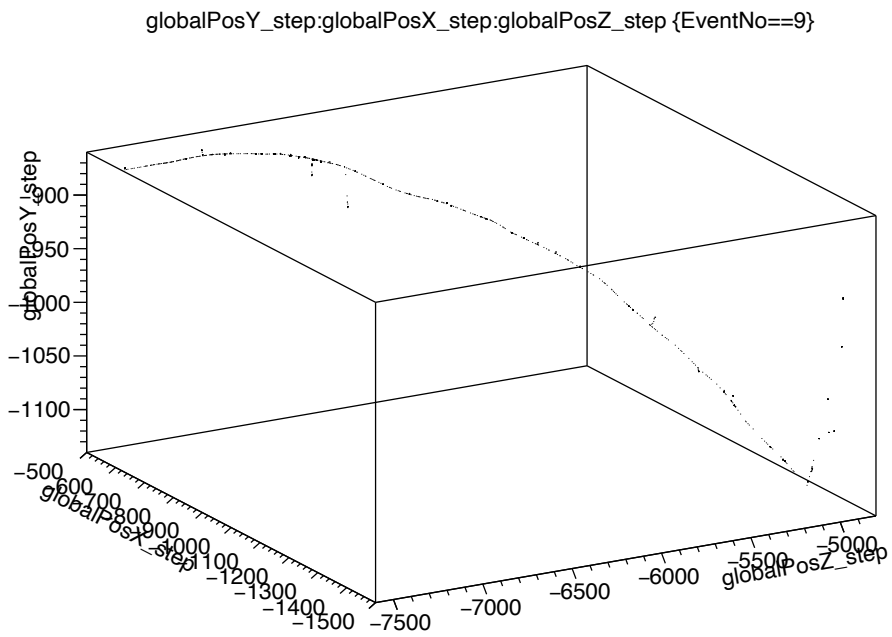
By executing “./triangles” without specifying anything, you should be able to run the simulation in interactive mode.

There you can create visual images, create runs with multiple events, etc.

Or, you can create a macro like “run_100eve.mac” in advance and execute the application by specifying the macro like “./triangles run_100eve.mac”

Some Nice Pictures





Left: Muon track from when I shot 100 μ^+s with random energy/position.
Right: One of the events

This picture uses additional information that was recorded for reference, so this wouldn't be possible when we actually build the detector.
I didn't have time to make any physically interesting observations/analyses…

Work that still need to be done

- Random number generator needs some adjusting (probably not that difficult)

- Support for multithreading needed

(Currently, ntuple merging is not available because of this. This means that if we do more than two runs during a session, we will be able to get data only from the last run.)

- Memory optimization needed?

(If we try to simulate a large number of events the simulation always stops somewhere from 10 ~ 100 events.)