TrackG4PS

A simulation of the CMS PS modules with cosmic rays

Piero Viscone

University of Pisa

Tools and libraries used

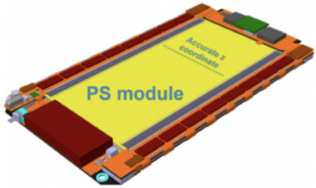
Geant4	Simulation toolkit
ROOT	Data analysis framework
Cmake	Makefile generator
CCache	Compiler cache
Catch2	Unit-testing
cppcheck	Static analyzer
Doxygen	Docs generator
github-actions	Docs deployment
github-pages	Docs host
cppitertools	python-like iterators

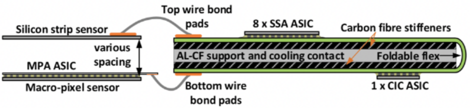
Lack of CI/CD due to the size of the dependencies.

Possible workarounds:

- Self-hosted runner: Security issues.
- Static linking: Not resolve the problem at all.
- Docker container: Exceed the guaranteed space for a free account.

PS modules





PS (and 2S) modules will be the modules of the outer tracker of CMS during the Run4.

They are made of two sensors: a pixel module and a strip module. These are 300 μm thick and spaced apart by a variable-length (1 to 4 mm).

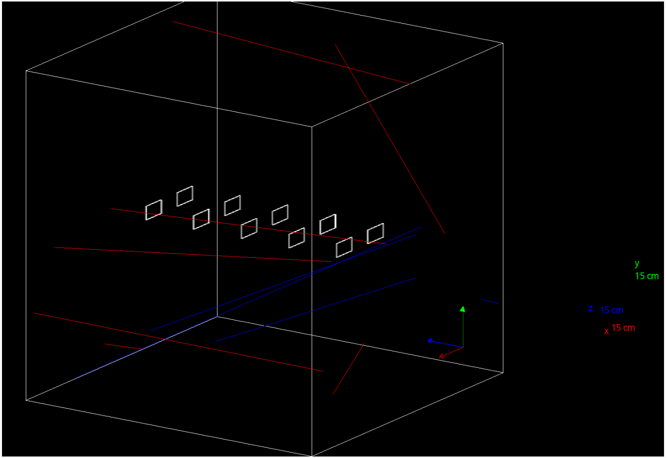
Pixel module

- 960×32 pixel
- Pixel dimension $100\mu m \times 1.5mm$

Strips module

- 2 column of 960 strips
- Strip dimension $100\mu m \times 2.5cm$

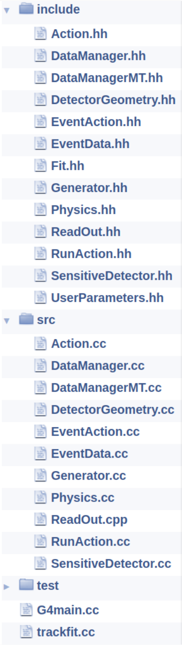
Detector setup



These modules are currently being tested in a cryostat with cosmic rays in a grid of 2 columns and 5 layers.

Simulation

Simulation



Geant provides a lot of abstract classes.
The user has to create concrete classes inheriting from these and defining their virtual methods.

DetectorGeometry Defines the geometry of the detector

EventAction Defines what to do at the start/end of an event

Generator Manage the particle gun

Physics Defines the physical processes involved

RunAction Defines what to do at the start/end of the run

SensitiveDetector Defines what to do at each step in the sensitive volumes

---

Other classes was also implemented

ReadOut Manage the ReadOut

EventData The class object to fill with the data

DataManager & DataManagerMT Manage the output files and data

All simulation parameters are grouped into namespaces in UserParameters.hh and can be changed to try different configurations.

Physics

The Physics module defined was G4EMStandardPhysics.
It defines:

- Compton Scattering, Photoelectric effect and pair production for photons.
- Positron annihilation
- Ionization, Bremsstrahlung, and multiple scattering for e and μ
- Pair production by μ

Cut in range for e^+, e^- set to avoid the production of low energy delta rays:

- 10cm in Air
- 0.1mm in Silicon

Particle Gun

The **particle gun** is generated in a random position on the face of the world box.
Muons generated according the cosmic rays:

- Charge ratio $\mu^+/\mu^- = 1.3$
- Angular distribution $\cos^2(\theta)$
- Energy distribution $\frac{dN_\mu}{dE_\mu d\Omega} \approx \frac{0.14 E_\mu^{-2.7}}{cm^2\ s\ sr\ GeV} \left(\frac{1}{1+\frac{1.1 E_\mu \cos\theta}{115 GeV}} + \frac{0.054}{1+\frac{1.1 E_\mu \cos\theta}{850 GeV}} \right)$ between 1 GeV and 1 TeV

All parameters are generated at the beginning of each event with the method GetRandom of ROOT::TF1.

ReadOut

Geant provides G4VReadOutGeometry: the documentation about this class is inexistent, so a simple ReadOut class was made from scratch.

To transform the position of the hit to the position of the channel hit.

1. find the position of the closest module to the hit on a given axis
2. translates the hit position in the frame of the module
3. calculate the position of the center of the channel that was hit in the module frame
4. translates back to the world frame

This is applied to the 3 axes separately.

Trigger

The trigger implemented is very simple:

- **Pulse height discrimination:** Rejects all hits with released energy below a given threshold (40 keV).
  The main scope of the pulse height discrimination is to limit the presence of delta rays' hits in the data.
- **Coincidence**: Accept only events with hits in at least two different module layers.

DataManager

Geant provides the class G4VAnalysisManager:

- Inconvenient to manage trees
- Inconvenient to fill them with custom object

Solution: use directly **ROOT methods** and a **custom data manager** to share the ROOT objects among different classes.

The **DataManager** class is a singleton that contains a TFile, a TTree, and a custom event object.

In this way, we can obtain the same TFile, TTree, or Event object when we need it in the virtual methods of different classes and use the ROOT methods directly to fill the tree and manage the file.

Event object

The data are stored in a root file with a single branch containing a custom Event object.

```
Event
├─ eventID
├─ detectorData
│  ├─ TrackID
│  ├─ ParticleID (0 μ⁻,1 μ⁺,2 e⁻,3 e⁺,4 γ)
│  ├─ EnergyDeposited (keV)
│  ├─ posX (position of the channel hit in mm)
│  ├─ posY
│  ├─ posZ
│  └─ Layer (odds are pixels, evens are strips)
└─ truthBeamData
   ├─ posX (position of the particle gun)
   ├─ posY
   ├─ posZ
   ├─ phi (polar angle)
   ├─ theta (azimuthal angle)
   ├─ energy (GeV)
   └─ particleID (0 μ⁻, 1 μ⁺)
```

- eventID is obtained in MyEventAction::EndOfEventAction virtual method
- The detector data is obtained in MySensitiveDetector::ProcessHits virtual method
- The beam data is obteined in MyPrimaryGenerator::GeneratePrimaries virtual method

Thread safety

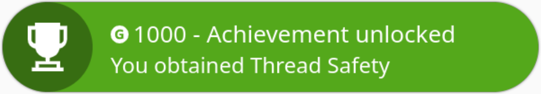
Geant strategy: Each thread works on a different file and different events.

Problem: DataManager is a singleton. All threads get the same TFile and Event object

The **DataManagerMT** singleton class is a friend class of DataManager.

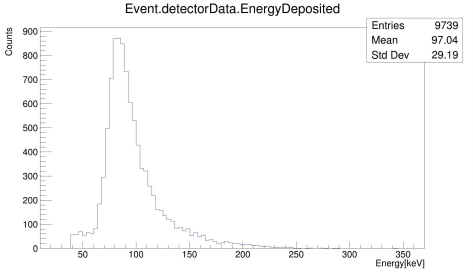
It accesses the private constructor of DataManager to create multiple instances of DataManager, one each thread

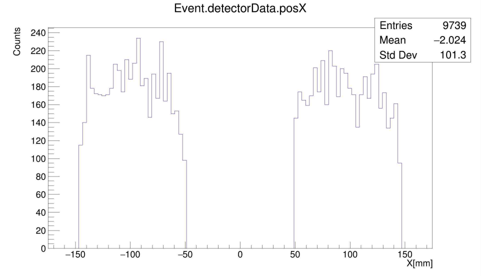
At the beginning of the run, the master thread creates the DataManagerMT instance; then, each worker thread can obtain its own DataManager object



Results

1819 events accepted by the trigger over 500.000 particles generated by the particle gun.





Track Reconstruction

Tracking algorithm

Simple toy model: **Linear fit into the XZ and YZ projections**

$$z = m_x x + x_0$$

$$z = m_y y + y_0$$

1. Sort the hits along the z-axis
2. Compute the initial parameters of the linear function considering only the first and the last point.
3. Perform the fit exploiting the TGraphError class. (Error bars set to the pitch of strips/pixels divided by $\sqrt{12}$)

Reconstructed events

Reconstructed events are saved in an NTuple.

```
fitResults
 |__evID (Id of the reconstructed event
 |__nHit (num of hits)
 |__x0
 |__x0_err
 |__mx
 |__mx_err
 |__chi2zx (reduced chi2 of the ZX fit)
 |__y0
 |__y0_err
 |__my
 |__my_err
 |__chi2zy
```

Despite the trigger, there are still events affected by delta rays

Results I

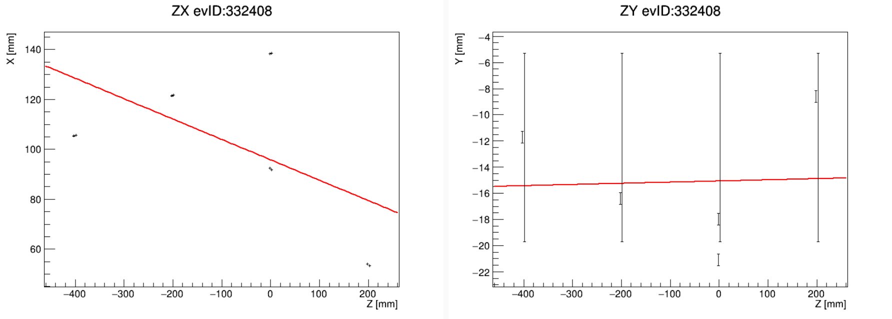


Figure 1: Event affected by the presence of delta rays' hits

Results II

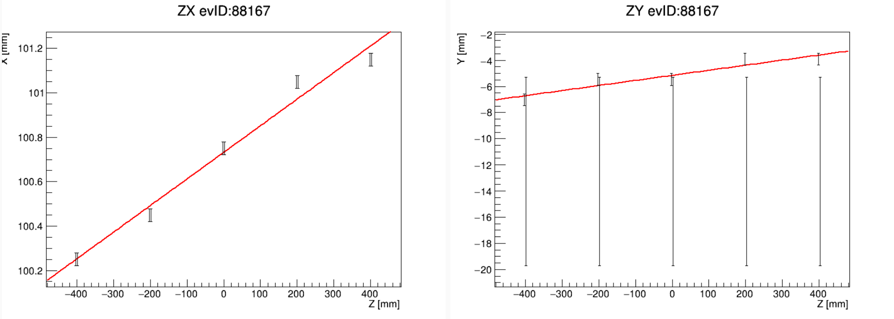


Figure 2: Well fitted event. Multiple scattering is clearly visible

Results III

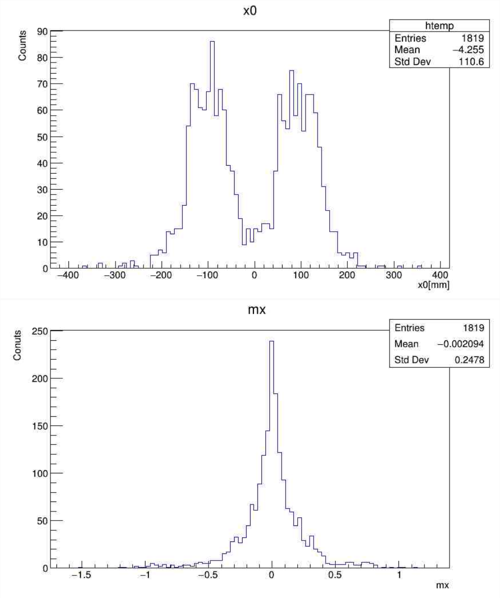


Figure 3: Best fit parameters in the XZ projection

[2, 1]

Geant4.
Geant4 User Documentation.
https://geant4.web.cern.ch/support/user_documentation.

A. L. Rosa.
The Upgrade of the CMS Tracker at HL-LHC.
https://doi.org/10.7566/JPSCP.34.010006, 2021.