Geant4 Masterclass

July 5, 2021

Overview

Jack Bishop

Intro

Intro

About me:
PhD at University of Birmingham, UK
Currently postdoc in Grigory Rogachev's group
Focusing on nuclear structure and astrophysics with TexAT TPC

Your current interests/experience

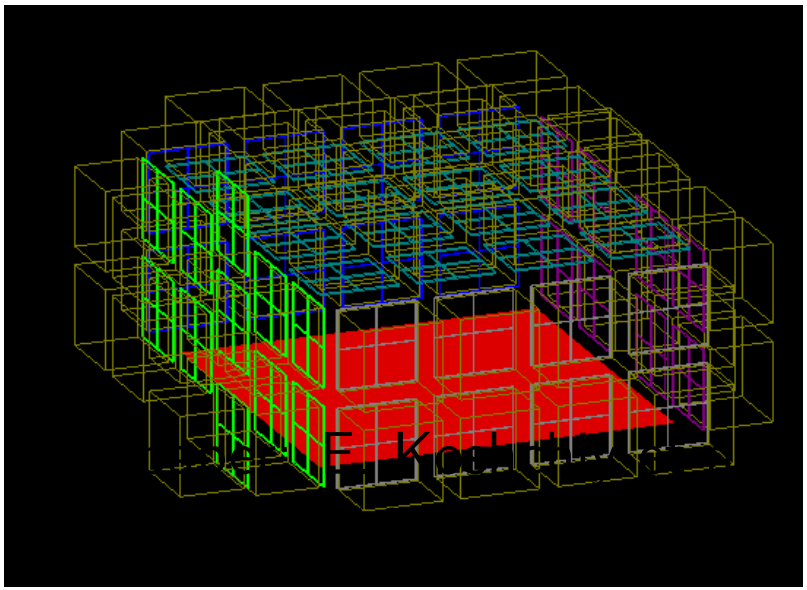
- Current project
- Any prior Geant4 experience?
- Any ROOT experience?

TexAT TPC

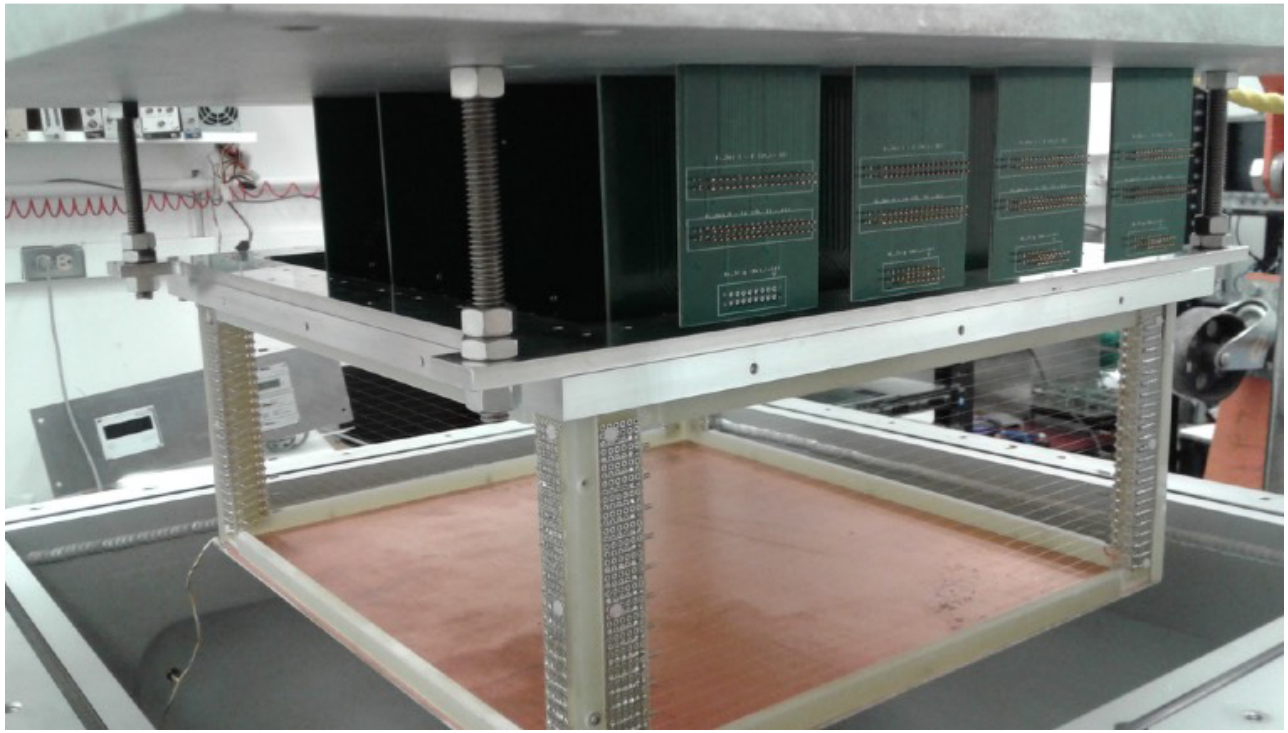
TexAT overview

TexAT TPC - TEXas Active Target Time Projection Chamber

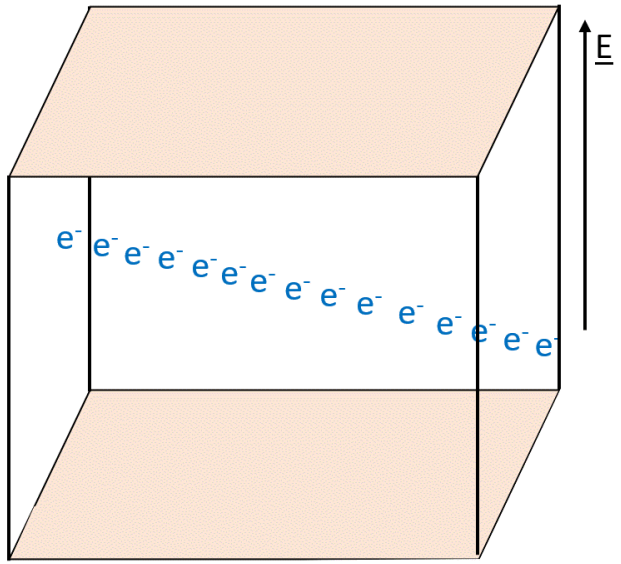
- 224 x 240 x 130 mm sensitive area
- Segmented readout using Micromegas, 1024 channels, pos. res. $\approx$ 1.5 mm in beam direction
- Gas Electron Multipliers (GEMs) provide additional gain. Low dE/dx particle tracks possible
- General Electronics for TPCs (GET) system digitizes waveforms. 512 time buckets at 10 MHz



TexAT overview

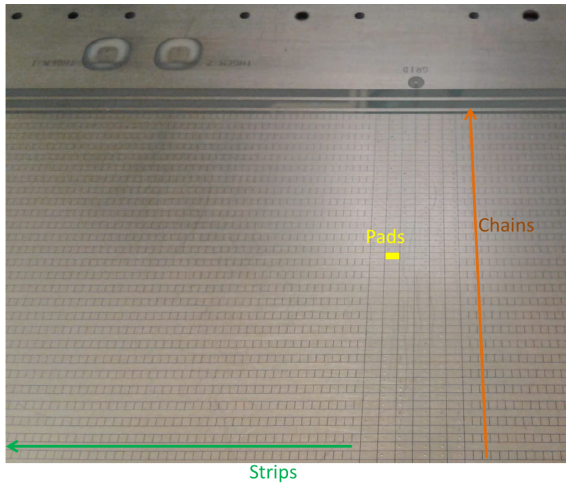


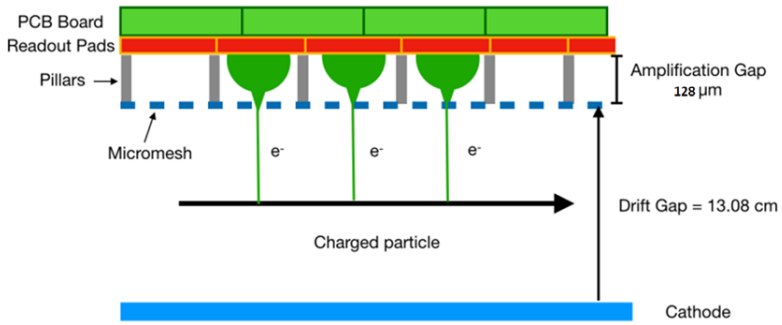
How a TPC works



Micromegas

- Micromegas-based readout
- Amplify and measure electron drift signals
- 128 μm gap
- Central region pads 1.75 x 3.5 mm
- Side regions require multiplexing into ‘strips’ and ‘chains’ parallel and perpendicular to beamline
- THGEMs (1.25 mm thick) or GEMs (128 μm thick)





GEANT4

Basic overview

What is GEANT4?

- GEANT4 is a Monte Carlo simulation suite with built-in physics libraries and cross section data
- We give it a detector geometry (shapes, materials, densities, sensitive areas etc.)
- Fire some particles
- Physics happens
- Extract the result

Start with an example

Fake body composed of different shapes of air, tissue and bone facsimilies
Firing protons and gamma rays into the body to examine the dose generated
GEANT4 example B1 (basic)
$\rightarrow$ Powerpoint

Designing a TPC simulation

Designing a TPC simulation - overview

Kinematics - what physics do we want to put in? Generate some nuclear reactions in our gas
Collect secondary electrons - how do we get the energy loss in the gas out of Geant4?
Drift electrons - model the gas drift to the MM pads
Channel mapping - Generate electronics outputs
Ancillary detectors - Do we want any other detectors alongside the TPC?
https://github.com/jackbishop-phys/bishsim_eTPC

Creating nuclear reactions

Creating nuclear reactions - custom physics classes

We can design our own custom physics classes to include the kinematics of interest to us (e.g. photodissociation)

```
// Physics list
//  G4VModularPhysicsList* physicsList = new QGSP_BERT_HP;
 G4VModularPhysicsList* physicsList = new G4VModularPhysicsList();//Make our own physics list
 G4EmStandardPhysics* emStandard = new G4EmStandardPhysics();
 physicsList->RegisterPhysics(emStandard);
 physicsList->SetVerboseLevel(0);
 runManager->SetUserInitialization(physicsList);
 G4RadioactiveDecayPhysics* radDecayPhysics = new G4RadioactiveDecayPhysics; // radDecayPhysics->ConstructParticle();
 physicsList->RegisterPhysics(radDecayPhysics);
 // Decay
 G4DecayPhysics *decayphys = new G4DecayPhysics();
 physicsList->RegisterPhysics(decayphys);


 // Add in JEB's binary reaction physics
 BinaryReactionPhysics* reactionPhysics = new BinaryReactionPhysics;
 reactionPhysics->SetReactionParams();
 physicsList->RegisterPhysics(reactionPhysics);
 runManager->SetUserInitialization(physicsList);
 G4cout<<"Setting up physics"<<G4endl;
```

Create our own physics list

Add in some standard physics (i.e. EM, radioactivity)

Introduce our own physics for binary reactions via our class for **BinaryReactionPhysics**

Custom Physics Class

The custom physics class is very simple and provides an interface between our 'main' and where the real meat of the physics is included: BinaryReactionProcess

GNU nano 2.3.1 File: src/BinaryReactionPhysics.cc

```
#include "BinaryReactionPhysics.hh"

#include "BinaryReactionProcess.hh"
#include "G4GenericIon.hh"
#include "G4Neutron.hh"
#include "G4Proton.hh"
#include "G4Deuteron.hh"
#include "G4Alpha.hh"
#include "globals.hh"
#include "G4PhysicsListHelper.hh"

// factory
#include "G4PhysicsConstructorFactory.hh"
//
G4_DECLARE_PHYSCONSTR_FACTORY(BinaryReactionPhysics);

BinaryReactionPhysics::BinaryReactionPhysics(G4int)
:  G4VPhysicsConstructor("BinaryReactionPhysics") {
}

BinaryReactionPhysics::BinaryReactionPhysics(const G4String& name)
:  G4VPhysicsConstructor(name) {
    G4cout<<"Creating Binary Reaction Physics"<<G4endl;
}

BinaryReactionPhysics::~BinaryReactionPhysics()
{
}

void BinaryReactionPhysics::ConstructParticle()
{
  G4GenericIon::GenericIon();
  G4Neutron::Neutron();
  G4Proton::Proton();
  G4Alpha::Alpha();
}

void BinaryReactionPhysics::ConstructProcess()
{
  BinaryReactionProcess* reactionProcess = new BinaryReactionProcess();
  reactionProcess->ParseParams(fReactionParams);
  G4PhysicsListHelper::GetPhysicsListHelper()->
    RegisterProcess(reactionProcess, G4GenericIon::GenericIon());
  G4PhysicsListHelper::GetPhysicsListHelper()->
    RegisterProcess(reactionProcess, G4Neutron::Definition());
  G4PhysicsListHelper::GetPhysicsListHelper()->
    RegisterProcess(reactionProcess, G4Proton::Definition());
  G4PhysicsListHelper::GetPhysicsListHelper()->
    RegisterProcess(reactionProcess, G4Deuteron::Definition());
  G4PhysicsListHelper::GetPhysicsListHelper()->
    RegisterProcess(reactionProcess, G4Alpha::Definition());


}
```

Binary Reaction Process

We have now added in our own physics class and told GEANT4 which particles we want to construct the process for
Control the physics process via interfacing with the mfp
mfp = mean free path
Definition: mean distance before an interaction
Geant4 works via evaluating the mfp of a particle and then calculating how large the 'step' will be before this interaction occurs*
$I = I_0 \exp(-x/\ell) \rightarrow x = \ell \ln(\frac{I_0}{I})$
$\frac{I_0}{I}$ from 0 to 1, therefore we can select the distance of the step via this equation and randomly choosing a random uniform dist

Forcing/biasing interactions

We can set the mfp via an evaluation of some external input of cross sections and use the density of the target etc.
This means 99.999% of events simulated will not be what we want $\rightarrow$ **biasing**
a) We can increase the cross sections we are reading in by an artifical factor of 1000 or so, therefore we need to simulate 1000 times fewer events
BUT! If we make this artifical factor too large then our events will be triggered very early inside of the detector, we need to be careful!
b) We can force **every** event to interact via manipulating the mfp parameter
Access the G4Track to get properties of particle, if our reaction triggers then set mfp $= 0$

Biasing example

E.g. $^{12}\mathrm{C}(\alpha,\gamma)$, first, choose an E_{CM} to simulate this event at, at each step of our physics process, if the COM E is less than where we want it to occur, then force the reaction to happen now with mfp=0

We can then make/destroy new particles

For narrow resonances, we need to be careful and ensure the Geant4 step size remains small otherwise we may 'overshoot' the COM E by a little bit (also true for the cross section approach)

This is all done in GetMeanFreePath, we can get the name of the particle being simulated too and ensure we only generate the physics for the particle of interest (and also ensure that the interactions occurs for the particle with the correct TrackID.

Making an interaction

PostStepDoIt() is called whenever the mfp determines an event should occur. Here, we can once again check the particle name and for each particle interaction/decay we want to occur, we can call a function that removes the previous particle and creates new ones. E.g. TwoBody in bishsim takes current particle and interacts it with the target (Z_T,A_T) to make light product (Z_1,A_1) and heavy product (Z_2,A_2) with excitation energy if we would desire. We can also perform decays (i.e. $^{16}\mathrm{O}^{\star} \rightarrow ^{12}\mathrm{C} + \alpha$.

Making/destroying particles

We then need to create new G4DynamicParticle for the daughter products with their G4ParticleDefinition, angles, energies etc. as we wish to define them from the kinematics we decide.
Then, simply:

```
aParticleChange.AddSecondary(sec1);
aParticleChange.AddSecondary(sec2);
aParticleChange.ProposeEnergy(0.);
aParticleChange.ProposeTrackStatus(fStopAndKill);
return &aParticleChange;
```

Sensitive detectors

Sensitive detectors gives us information after each step
Big picture:
Look at edep, (x,y,z,t) and pass to a collection class
At the end of the event, EventAction collects all energy deposition from collection class
Convert eV $\rightarrow$ electrons from work function (or gamma function with fano factor)
Drift each electron
(x,y) lookup for uvw strip

Event hierachy

Stepping Action - called after each step in our simulation, can extract edep and what caused the energy deposition
Event Action - called after the end of the event, collect all of the (x,y,z,t,E) electrons, perform the drift and send it to a file
Run Action - called after all of our particles have been fired - save the file

Drifting every single electron (thousands) for every event (thousands-millions) will take a very very long time, we can use some precalculations to shorten the computational strain using Garfield.

Garfield++

Basic overview

What is Garfield++?

- Garfield++ is a program we can use to simulate electron drift properties
- Given a certain gas, pressure and electric field strength, what is the average drift property of our electrons
- Avalanche simulations can also be performed

How to use it

Can either write a stand alone compilable code or use Garfroot
My code for calculating basic properties is called IHM - I Hate Mondays
https://github.com/jackbishop-phys/IHM

I Hate Mondays

Designed to take user input and spit out drift velocities
Can also calculate time and spatial resolution

```
[jackbishop@gr-gmaster ~/IHM]$ ./IHM
Heed:
    Database path: /usr/local/garfield/Heed/heed++/database


Gas drift generator
Please select gas mode: 1 for single gas or 2 for mixed gas
1
1 selected
Type in name of gas
Examples: He,CO2,Methane,Isobutane...
CO2
MediumMagboltz::SetComposition:
    CO2
Type in pressure in Torr
50
Type in field cage voltage (across the 13-cm gap)
1000
Electric field is 76.9231 V/cm
 RM48 INITIALIZED:  54217137           0         0


          PROGRAM MAGBOLTZ 2 VERSION 11.9

          MONTE CARLO SOLUTION FOR MIXTURE OF  1 GASES.
     ---------------------------------------------------

       GASES  USED                  PERCENTAGE USED

      CO2  2018 ANISOTROPIC           100.0000


  GAS TEMPERATURE =  20.0 DEGREES CENTIGRADE.
  GAS PRESSURE =    50.0 TORR.

  INTEGRATION FROM 0.0 TO     1.00 EV.  IN 4000 STEPS.
```

I Hate Mondays

Designed to take user input and spit out drift velocities
Can also calculate time and spatial resolution, NB should be in units of $cm^{-1/2}$

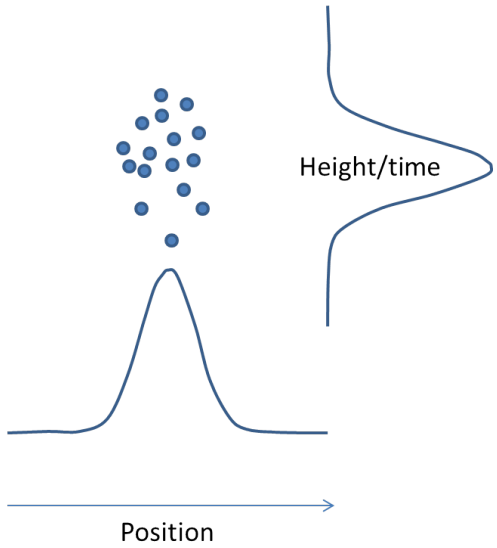
```
      E=      0.938      SPEC= 0.0000D+00
      E=      0.963      SPEC= 0.5000D-07
      E=      0.988      SPEC= 0.0000D+00
Drift velocity: 0.000841494 cm/ns
Drift resolution: 0.0334168 cm^1/2
Time resolution: 0.0309157 cm^1/2
[jackbishop@gr-gmaster ~/IHM]$
```

$$\sigma_x = R_D\sqrt{h} \quad (1)$$

$$\sigma_t = R_T\sqrt{h} \quad (2)$$

$$x = rndm3- > Gaus(x, \sigma_x) \quad (3)$$

$$t = rndm3- > Gaus(t, \sigma_t) \quad (4)$$



ROOT implementation

ROOT implementation

To store all of our simulation output, we want to write the salient outputs to a .root file
Generate TFile
Generate TTree
Define branches of TTree
Fill() for each events
At the end of the sim, save the TFile

Implementing channels

Implementing MM channels

Convert (x,y) to MM chan
For each event and every channel, we get a histogram of electrons and their arrival times
Shape these incident electrons with a gain G, and with a shaping time according to the GET shape

$$F(t) = A \exp(-3(x - t_0)/\sigma_t) * \sin((x - t_0)/\sigma_t) * ((x - t_0)/\sigma_t)^{\alpha},$$

for $t > t_0$, $\alpha \approx 1 - 5$.

Verification

How do we validate the results?

Simple case: p, α-tracks from data
Can we reproduce the key physics results?
What other features in the data do we wish to include? Noise?
Saturation? Pileup?

TexAtSim

Basic overview

What is TexAtSim?

- TexAtSim is the Monte Carlo simulation package for the TexAt detector
- It can simulate the response of the detector to various different beams/gases/decays/interactions
- The geometry/*detection* physics is already laid out in the code, you just have to input the reaction mechanism you need

Source code

- The source code is available on the github repository: `https://github.tamu.edu/rogachev-group/TexAtSim`
- UpdateGeometry branch
- To download it just do:

git clone -b UpdateGeometry
https://github.tamu.edu/rogachev-group/TexAtSim

Installation

To install on gr-gmaster:

1. Ensure your ROOT and GEANT4 paths are set properly (*source /usr/local/root/bin/thisroot.csh*and *source /usr/local/geant4/bin/geant4.csh*)
2. In the TexAtSim directory: *ccmake .*
3. Press *c* to configure
4. If you have no problems, press *c* again, then press *g*
5. You should now have a "Makefile", to make the executable, type *make*
6. When it says "Built target TexAtSim", you are fully installed
7. To execute: *./TexAtSim config/myconfigfile.json*

Physics included

Currently, the code is configured for two-body scattering $a(b, c)d$ where b is part of your detector gas. It can however do any type of decay (with some "small" modification) where the detector gas is not target-based:

- Beta decay
- Fusion
- γ-emission
- 3-body reactions (less trivial)

For this version checkout ''jeb" branch

2-body scattering - configuration file

To define the physics of interest, a configuration file is used:

```
{
  "interactive"     : true,
  "macroName"       : "macros/run.mac",
  "processNumber"   : 0,
  "gasPressure"     : 250.0,
  "gasTemperature"  : 293.15,
  "workFunction"    : 29.0,
  "fanoFactor"      : 0.282,
  "driftVelocity"   : 0.001601,
  "driftTimeRes"    : 0.0285798,
  "driftPosRes"     : 0.0309289,
  "qValue"          : 0.0,
  "target"          : [2,4],
  "lightProduct"    : [2,4],
  "heavyProduct"    : [10,18]
}
```

- Interactive mode?
- Run macro
- Random number seed
- Gas pressure (Torr)
- Gas temperature (K)
- Gas work function (eV)
- Fano factor
- Drift velocity (cm/ns) {Garfield}
- Drift time res.{Garfield}
- Drift pos res. {Garfield}
- Reaction Q-value
- Target nucleus of interest [A,Z]
- Final-state particle (light) [A,Z]
- Final-state particle (heavy) [A,Z]

Setting the gas - new!

The latest version of TexAtSim on UpdateGeometry branch (April 24th 2019) allows you to set the gas (including a mixture of two gases) from the configuration file:

```
{
  "interactive"    : false,
  "macroName"      : "macros/run.mac",
  "processNumber"  : 0,
  "gasName1"       : "G4_He",
  "gasName2"       : "G4_CARBON_DIOXIDE",
  "gasFrac1"       : 5.0,
  "gasFrac2"       : 95.0,
  "gasPressure"    : 50.0,
  "gasTemperature" : 293.15,
  "workFunction"   : 29.0,
  "fanoFactor"     : 0.282,
  "driftVelocity"  : 0.001601,
  "driftTimeRes"   : 0.0285798,
  "driftPosRes"    : 0.0309289,
  "qValue"         : 0.0,
  "target"         : [2,4],
  "lightProduct"   : [2,4],
  "heavyProduct"   : [10,20]
}
```

Mixture of He and CO2 here (5 % He)
Check the terminal output to ensure you have selected the gas correctly. For a pure gas just set either gasFrac1 or gasFrac2 to 0.

Simulating resonances - new!

The latest version of TexAtSim on UpdateGeometry branch (May 7th 2019) allows you to set a resonance to simulate from the configuration file

By default, the old method of selecting a scattering energy for the interaction to take place at will happen (selectionmode=1) **but** to simulate a resonance:

"selectionmode" : 2

"Eresonance" : [Energy of the resonance in COM (MeV),total width of the resonance (MeV),L of the state]

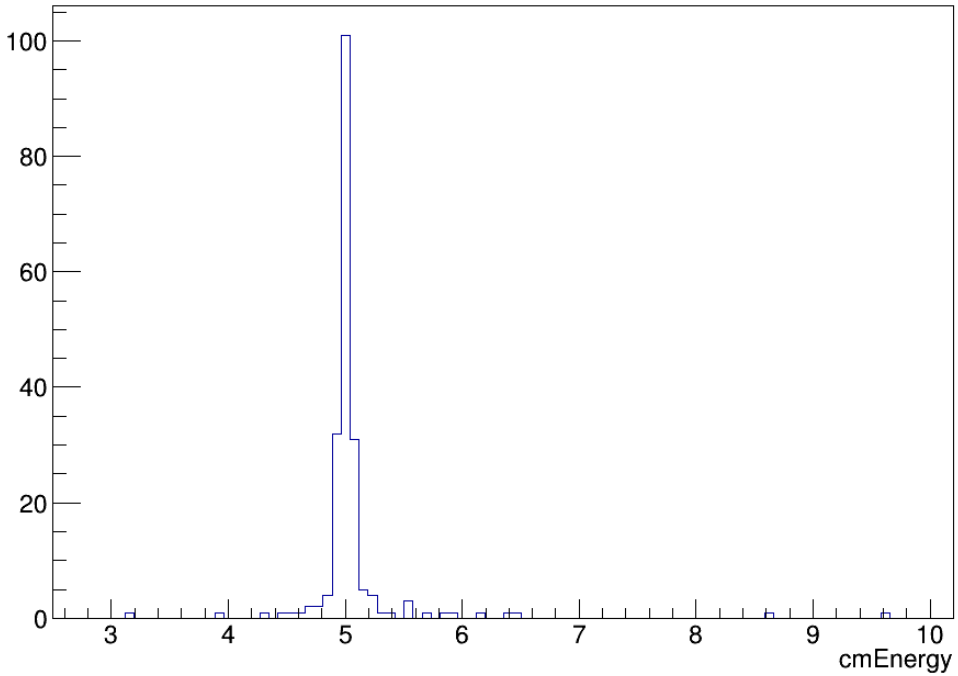
L of the state then gives the appropriate Legendre polynomial distribution for the resonance in the COM. Set to 0 for isotropic in COM (changed from the incorrect isotropy in lab before).

Simulating resonances - new!

Example:

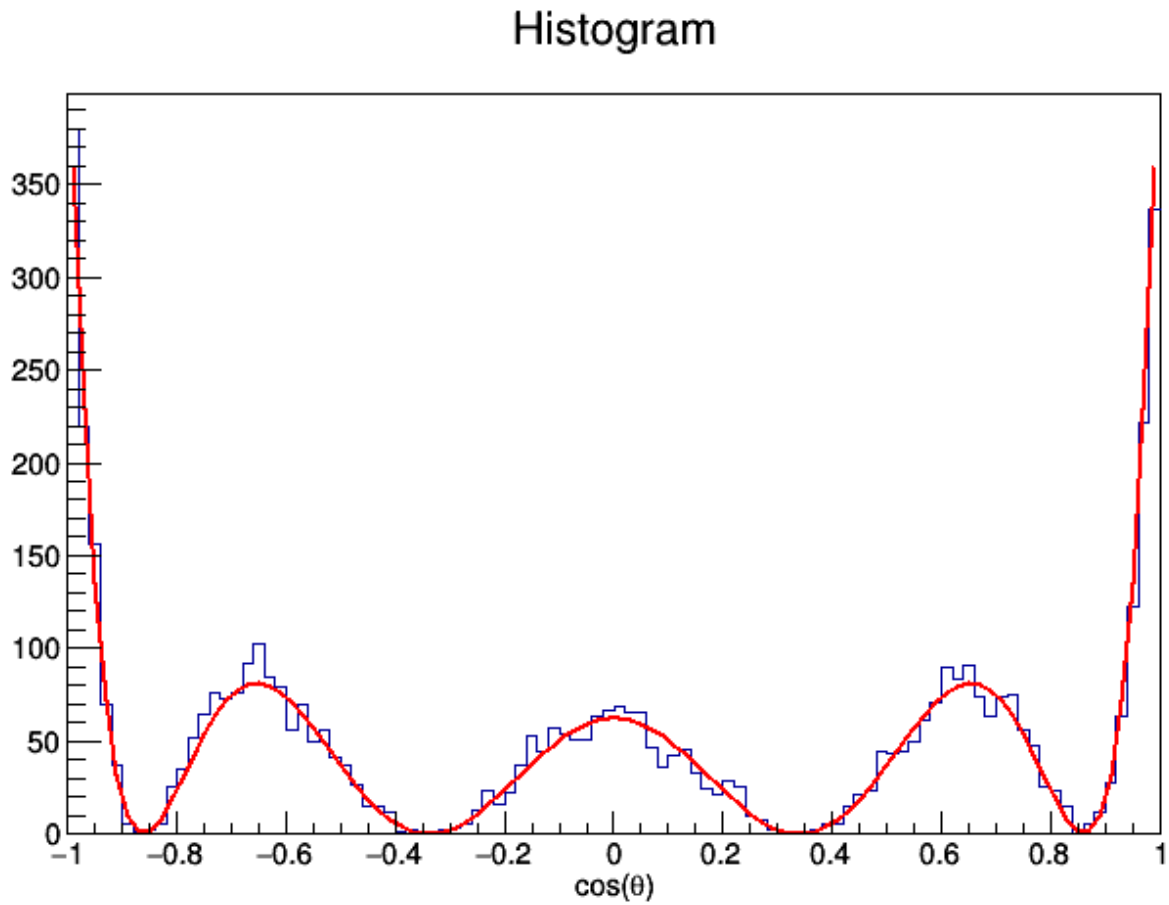
```
"lightProduct"    : [2,4],
"heavyProduct"    : [10,20],
"selectionmode"   : 2,
"Eresonance"      : [5.0,0.1,0]
}
```

The terminal output will verify these parameters, check the cmEnergy branch in the ROOT tree to check it is sampling correctly.



Simulating resonances - new!

With L=4:



Better exit channel parameters - new!

If you want your particles to be in an excited state following the reaction then you can add "excitation" : [Ex1,Ex2] (in MeV). By default these will be set to the ground state so omission of this parameter is not an issue.

Macros

Macros give *GEANT4* a set of commands that start your simulation with the parameters you require.
Example:

- */run/verbose 0* → Increase for info on terminal
- */event/verbose 0* → Increase for info on terminal
- */tracking/verbose 0* → Increase for info on terminal
- */gun/particle ion* → We're firing an ion
- */gun/ion 10 18* → Z=10, A=18
- */gun/energy 20 MeV* → Beam energy = 20 MeV
- */run/beamOn 10* → Fire 10 ions

There are also macros that define the visualization of events:

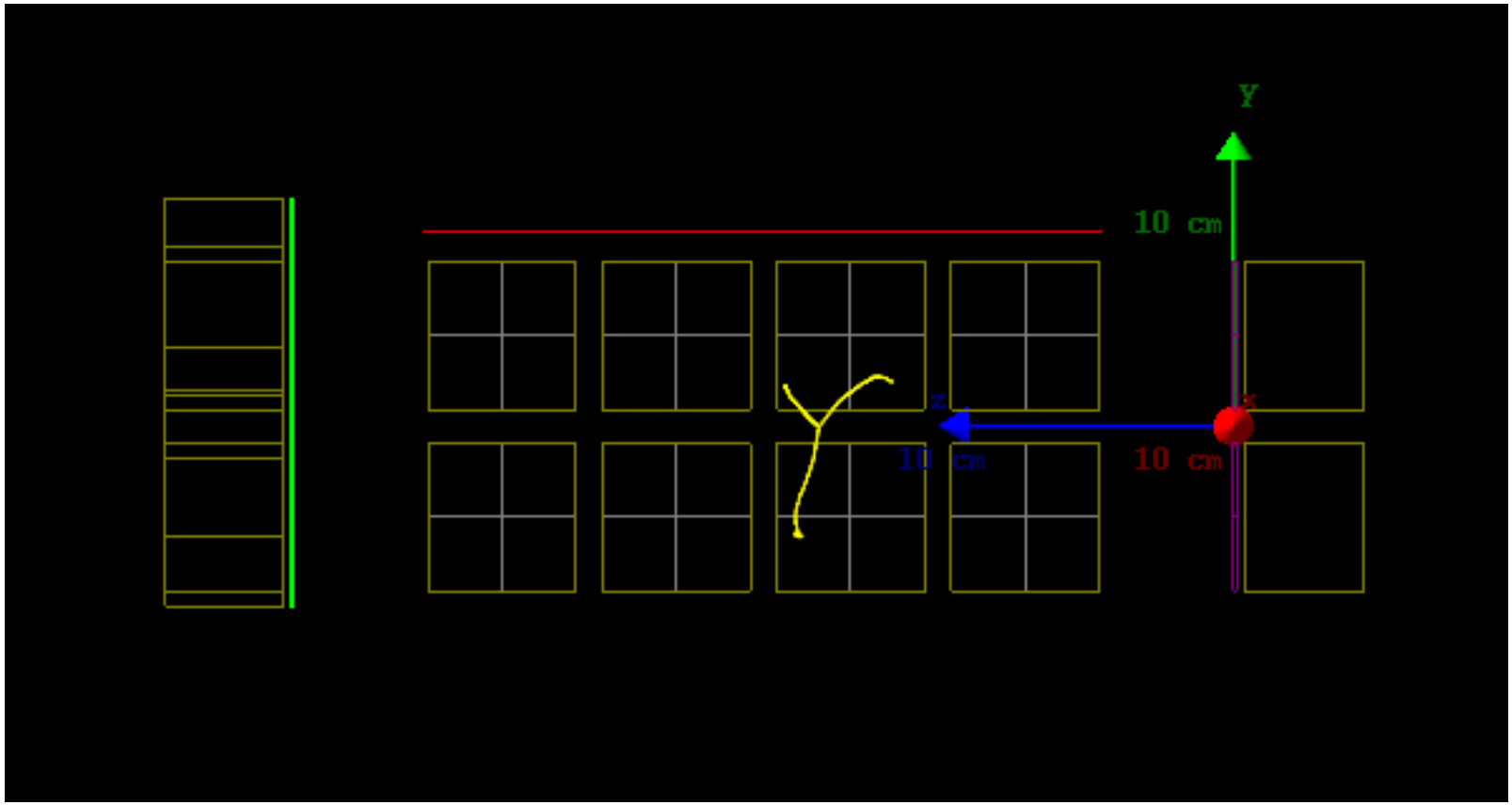
Visualization

If *interactive* is set to *true* in your configuration file then you will see a visualization of the events.

Visualization

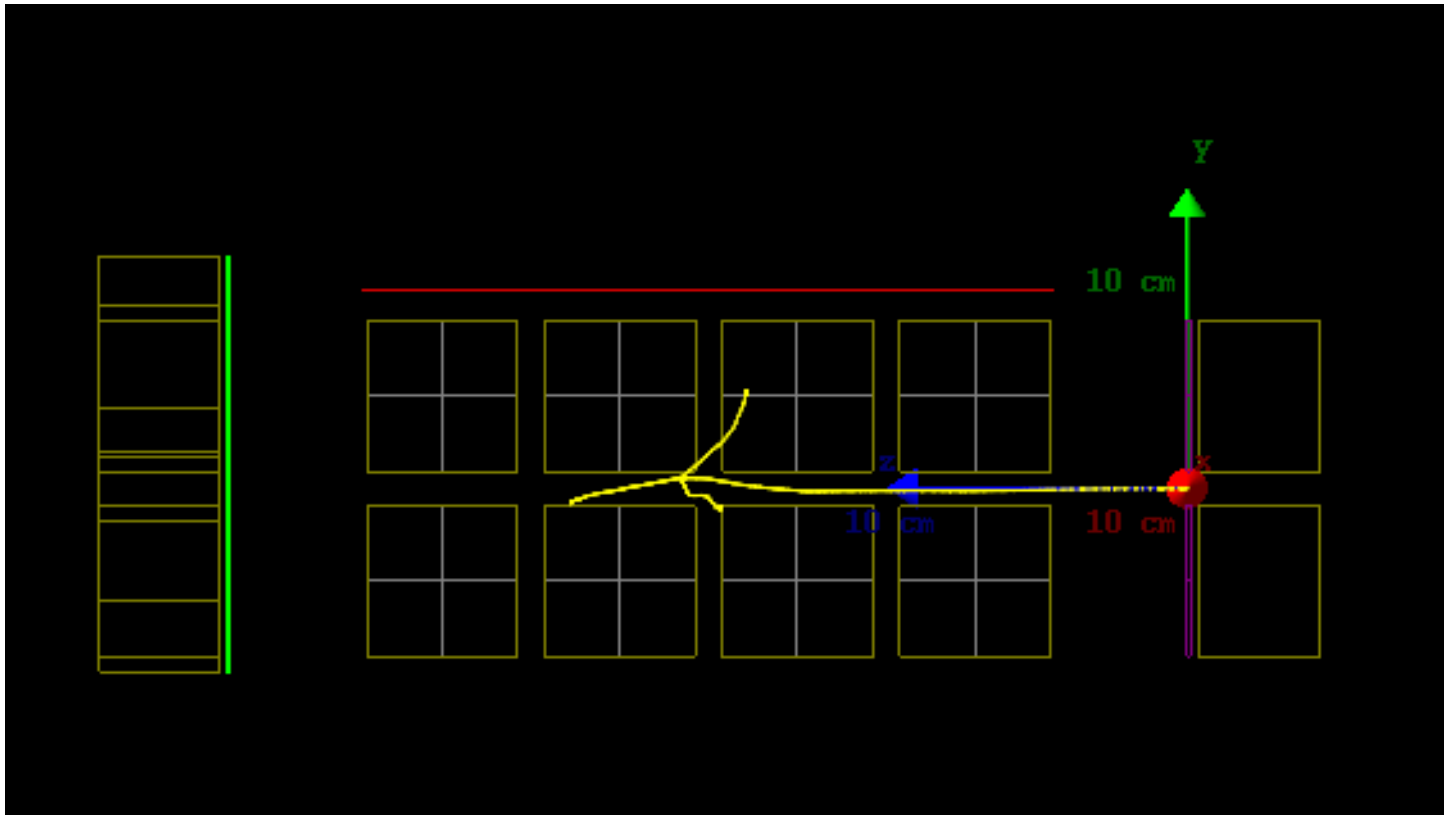
If *interactive* is set to *true* in your configuration file then you will see a visualization of the events.

Hoyle decay event



Visualization options

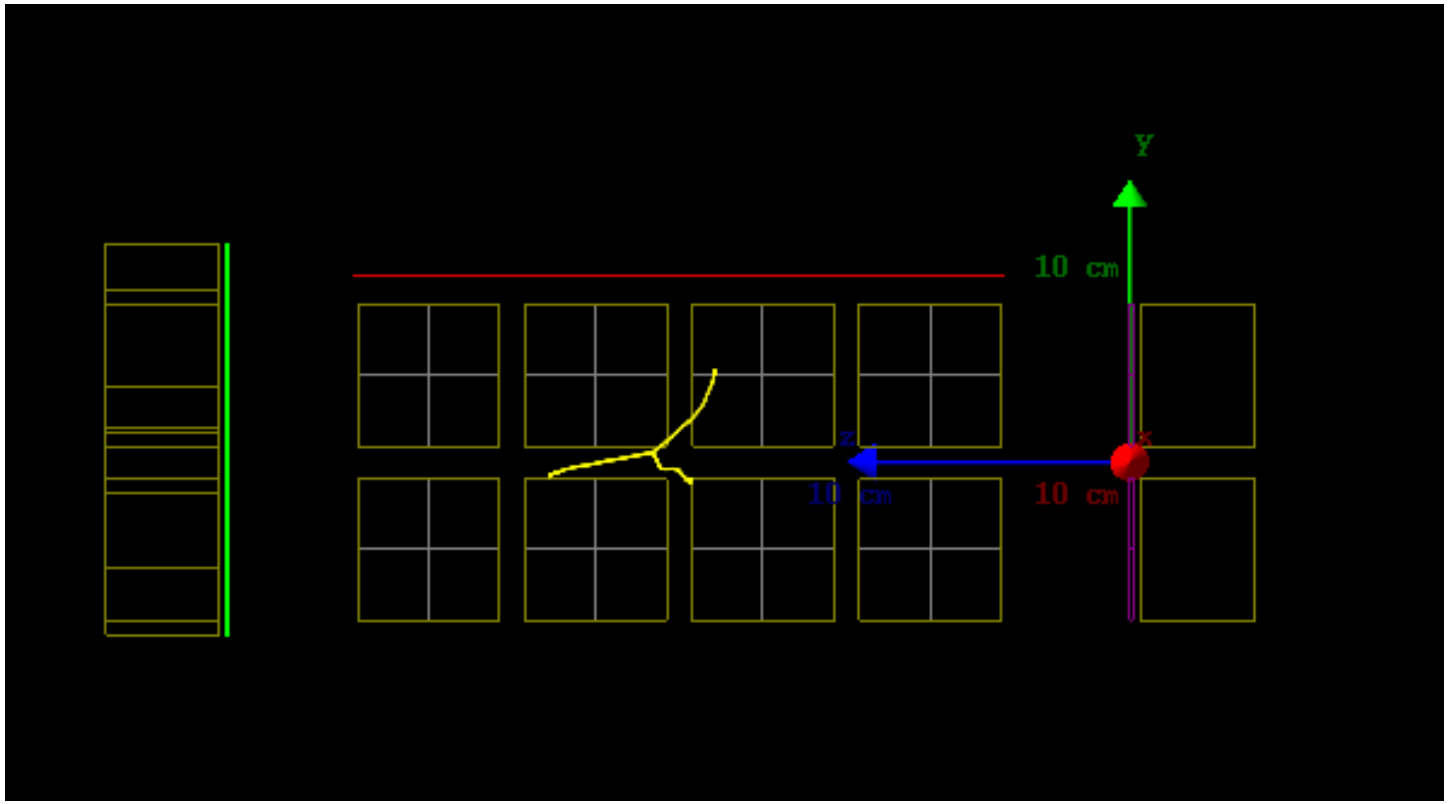
When debugging it may be useful to filter which particles are drawn:
Include all particles



We can change this by modifying *macros/vis.mac*

Visualization options

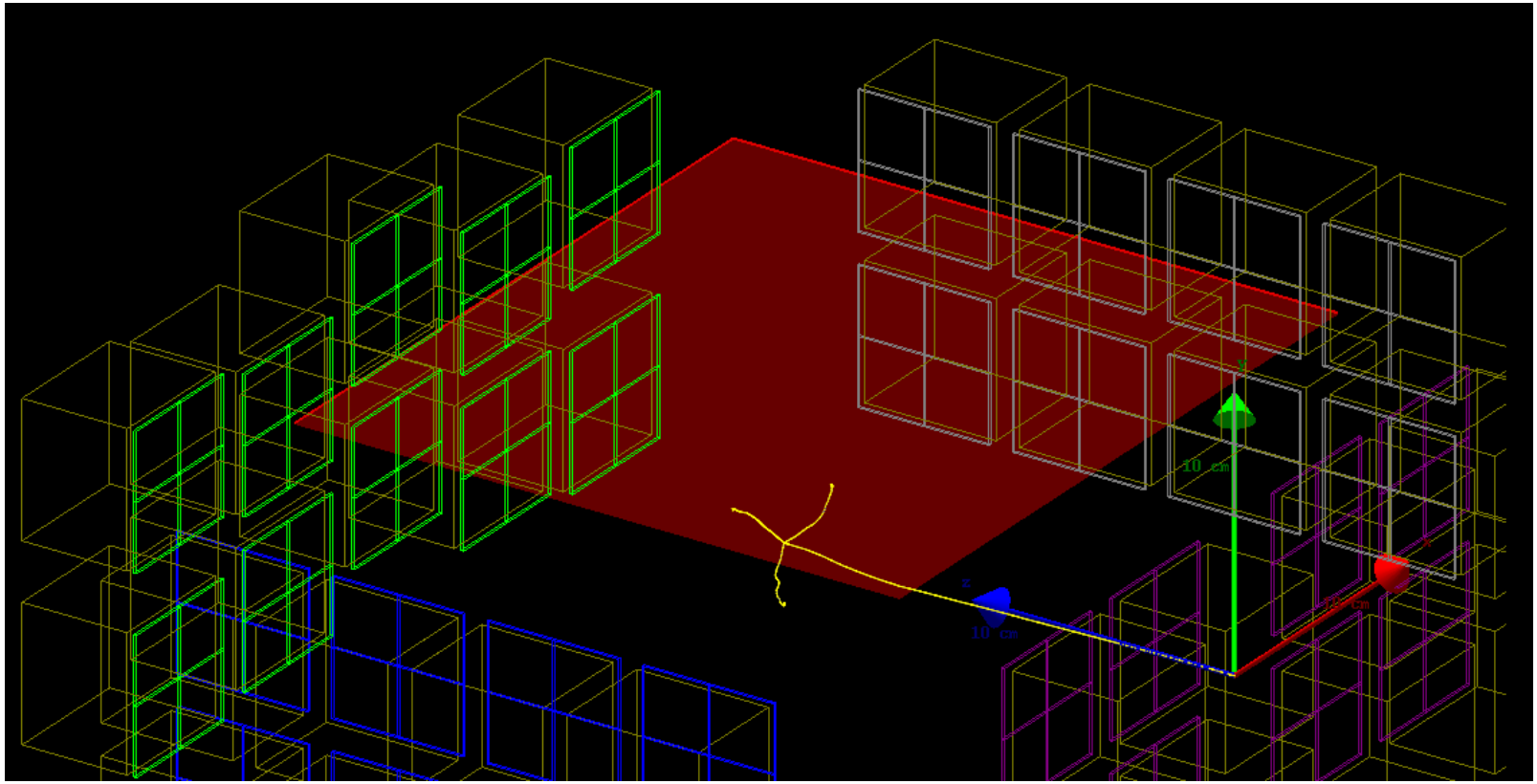
When debugging it may be useful to filter which particles are drawn:
Include only alpha-particles



/vis/modeling/trajectories/create/particleFilter
/vis/modeling/trajectories/create/particleFilter-0/add alpha

Visualization options

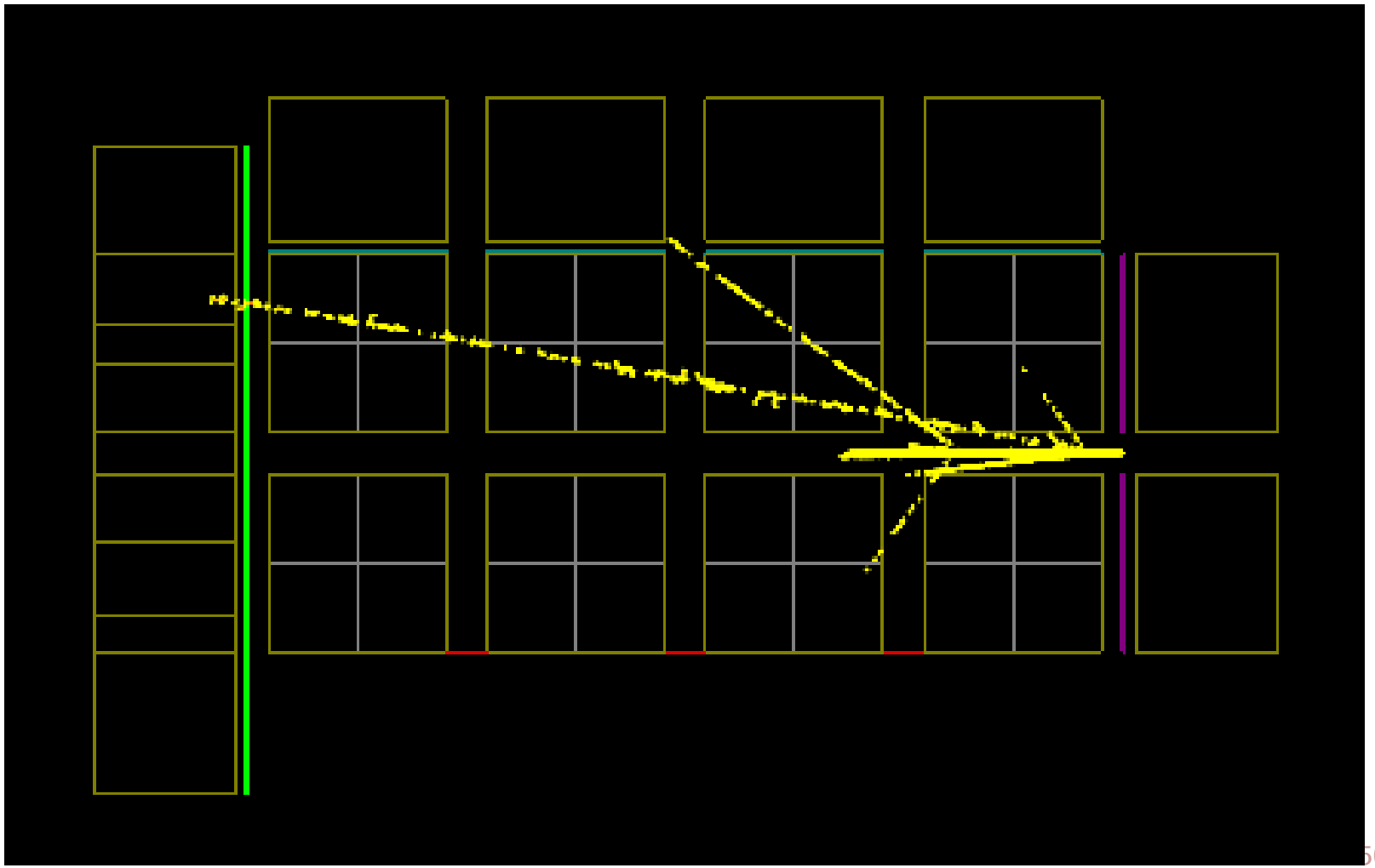
When debugging it may be useful to filter which particles are drawn:
Change the view direction/zoom



/vis/viewer/set/viewpointThetaPhi 60. 30.
/vis/viewer/zoomTo 2.

Outputs

We perform these simulations but what is the output we get from them? In the MM, this comes from the electrons:



Outputs

We perform these simulations but what is the output we get from them? In the MM, this comes from the electrons:

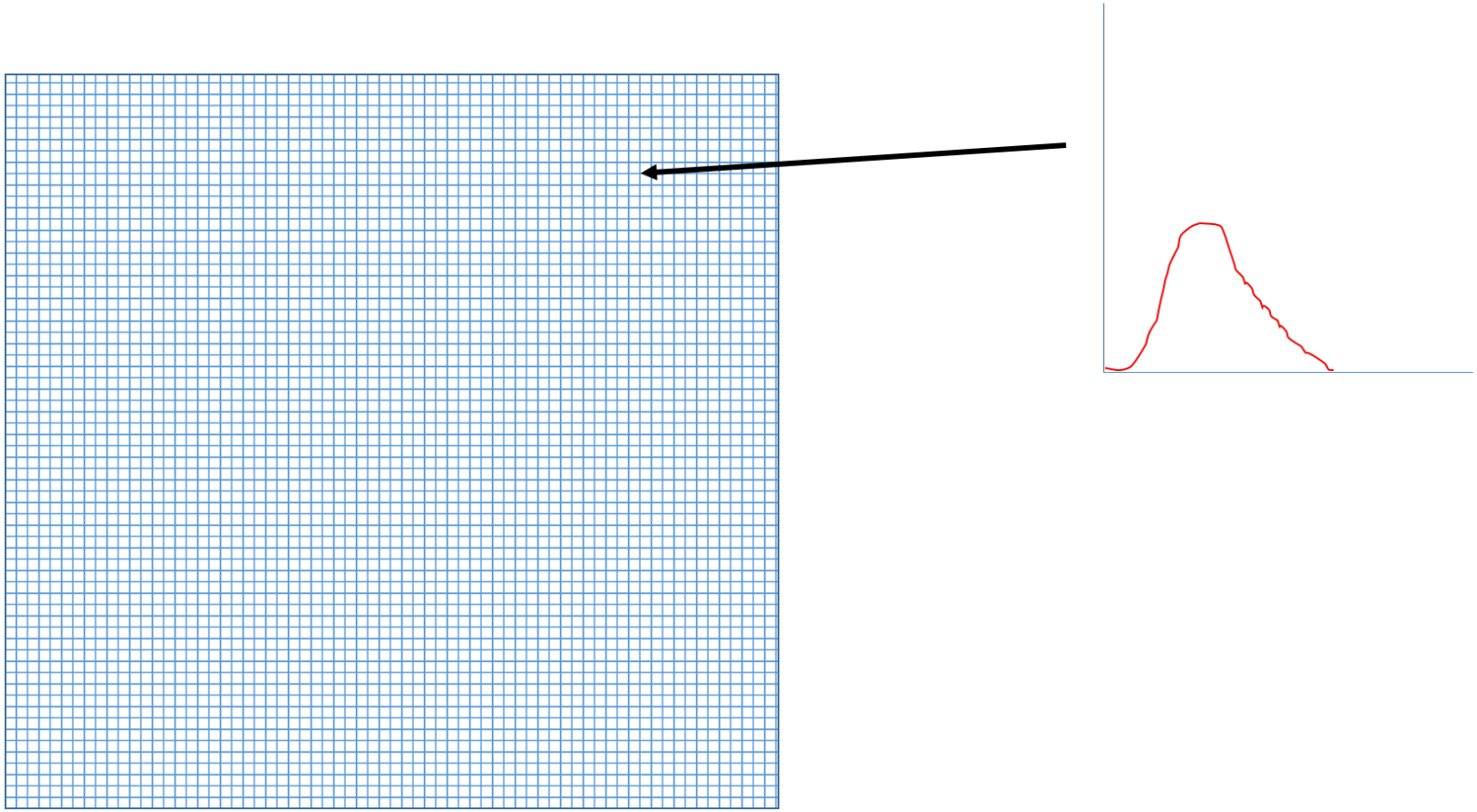
For each electron created along the track, it is drifted according to the drift velocity, position and time resolution provided in the config file.

```
double x1,z1,t1;
//Get endpoint of electron drift, translate it to a bin/pixel number
G4double height = 5.575+y;
G4double driftTime = height/fDriftVelocity;
t1 = t+fRandom3->Gaus(driftTime,driftTime*fDriftTimeRes/sqrt(height));
x1 = fRandom3->Gaus(x,height*fDriftPosRes/sqrt(height));
z1 = fRandom3->Gaus(z-4.175,height*fDriftPosRes/sqrt(height));
int bin = pixMap->FindBin(-x1,z1);
//check if pixel has been hit already, if not create a new timing histogram in map
if(timingHists.find(bin) == timingHists.end()) {
  timingHists[bin] = new TH1F(Form("t_%d",bin),Form("t_%d",bin),2048,0,10240);
  timingHists[bin]->SetDirectory(0);
}
timingHists[bin]->Fill(t1);
```

Outputs

We perform these simulations but what is the output we get from them? In the MM, this comes from the electrons:

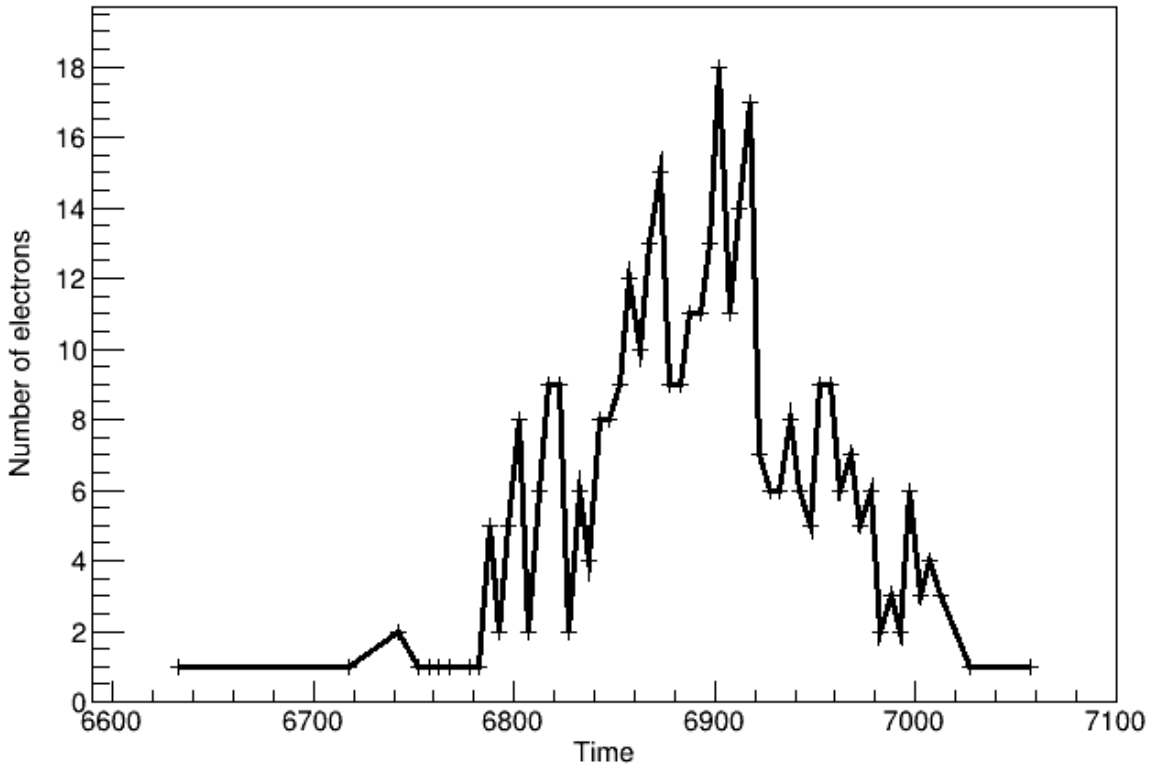
For each pseudo-pixel there is a histogram of electron hits.



Outputs

We perform these simulations but what is the output we get from them? In the MM, this comes from the electrons:

E.g.



(small amplitude pseudo-pixel chosen to show discreteness)

Additional information

Si/CsI are simpler: each hit has a X, Y, Z, E, T, detector number, quad number.

We also have the "truth" level interaction points/energies in our ROOT tree if we want to compare later...
This can answer questions such as:

- How does our reconstructed track vertex compare with the true vertex location?
- What is our position resolution in X, Y and Z?
- What is our efficiency as a function of vertex location?

New In new version (April 24th 2019), the ROOT tree gives you the light and heavy recoil E,θ and ϕ in lab and CM

Temporary page!

LaTeX was unable to guess the total number of pages correctly. As there was some unprocessed data that should have been added to the final page this extra page has been added to receive it.
If you rerun the document (without altering it) this surplus page will go away, because LaTeX now knows how many pages to expect for this document.