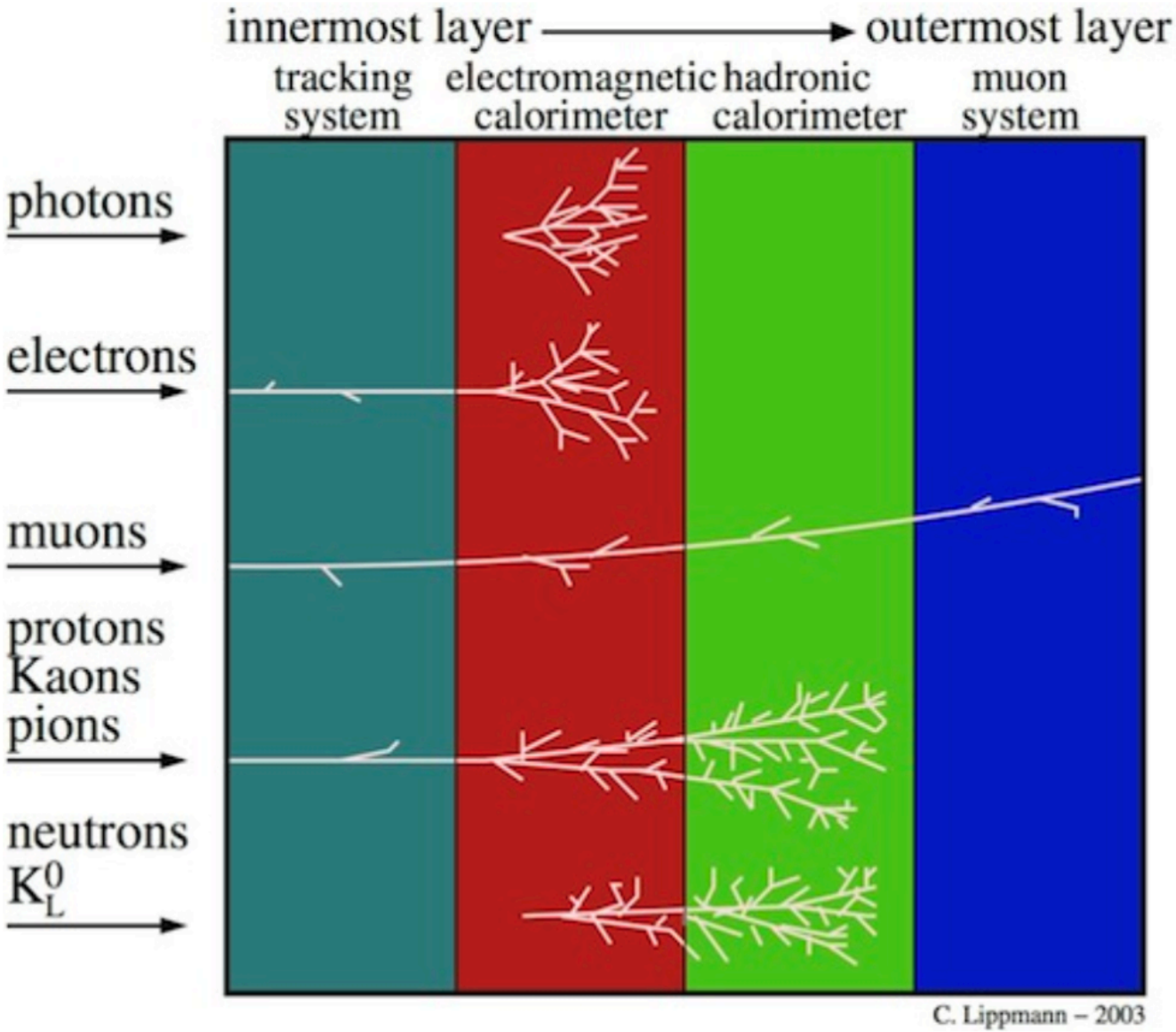
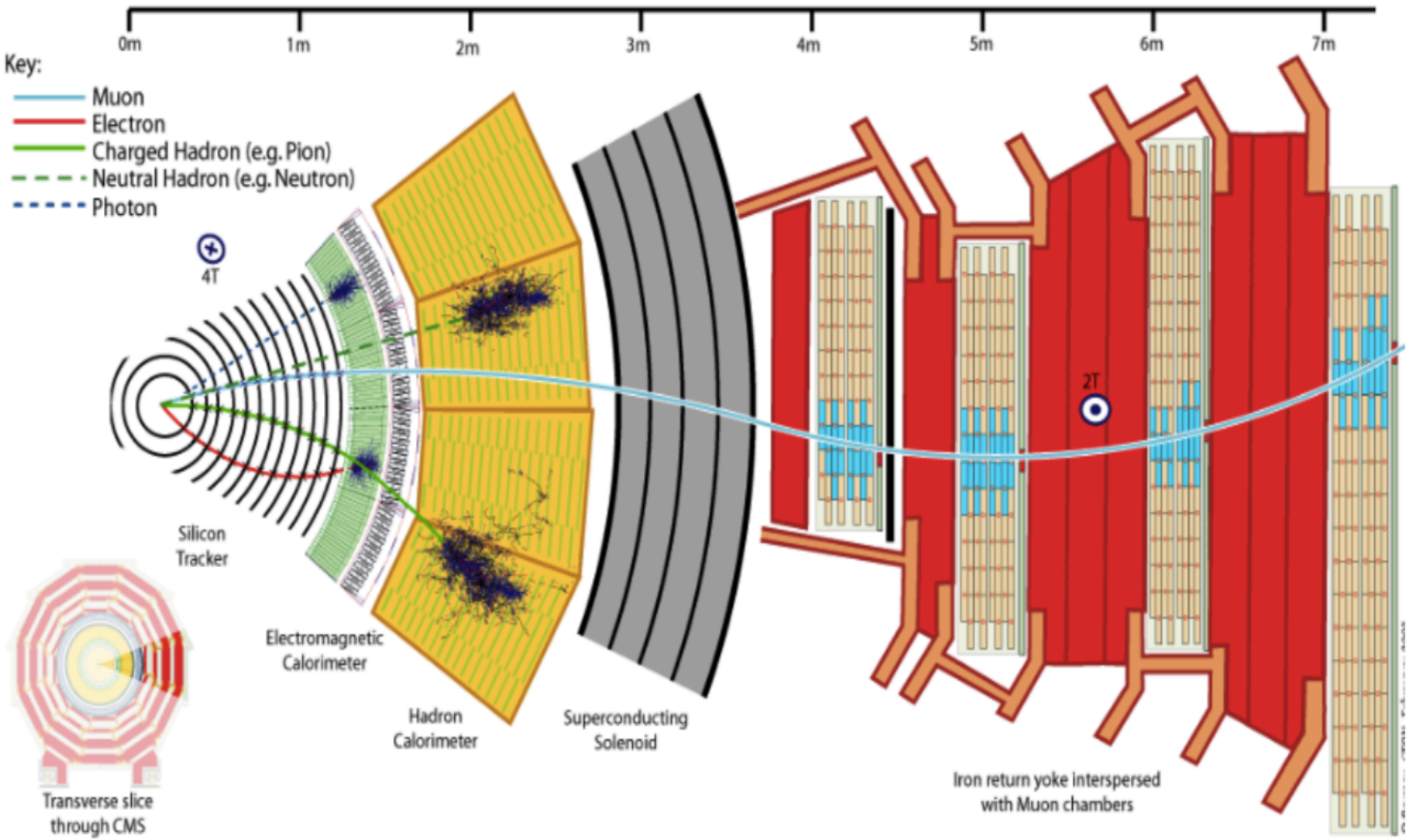
Delphes Framework

fast simulation

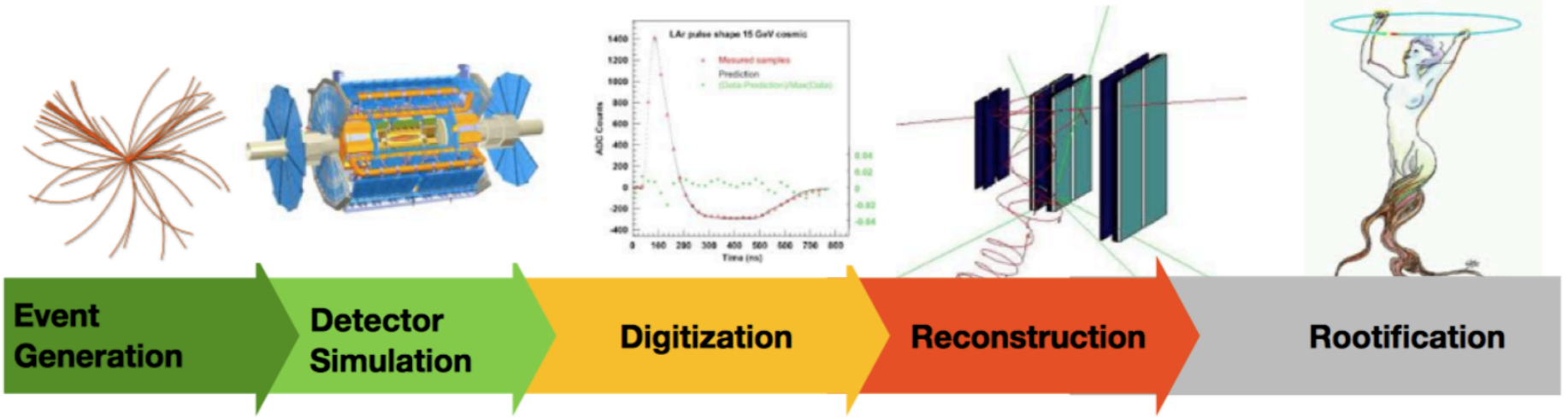
Particle Physics Detector



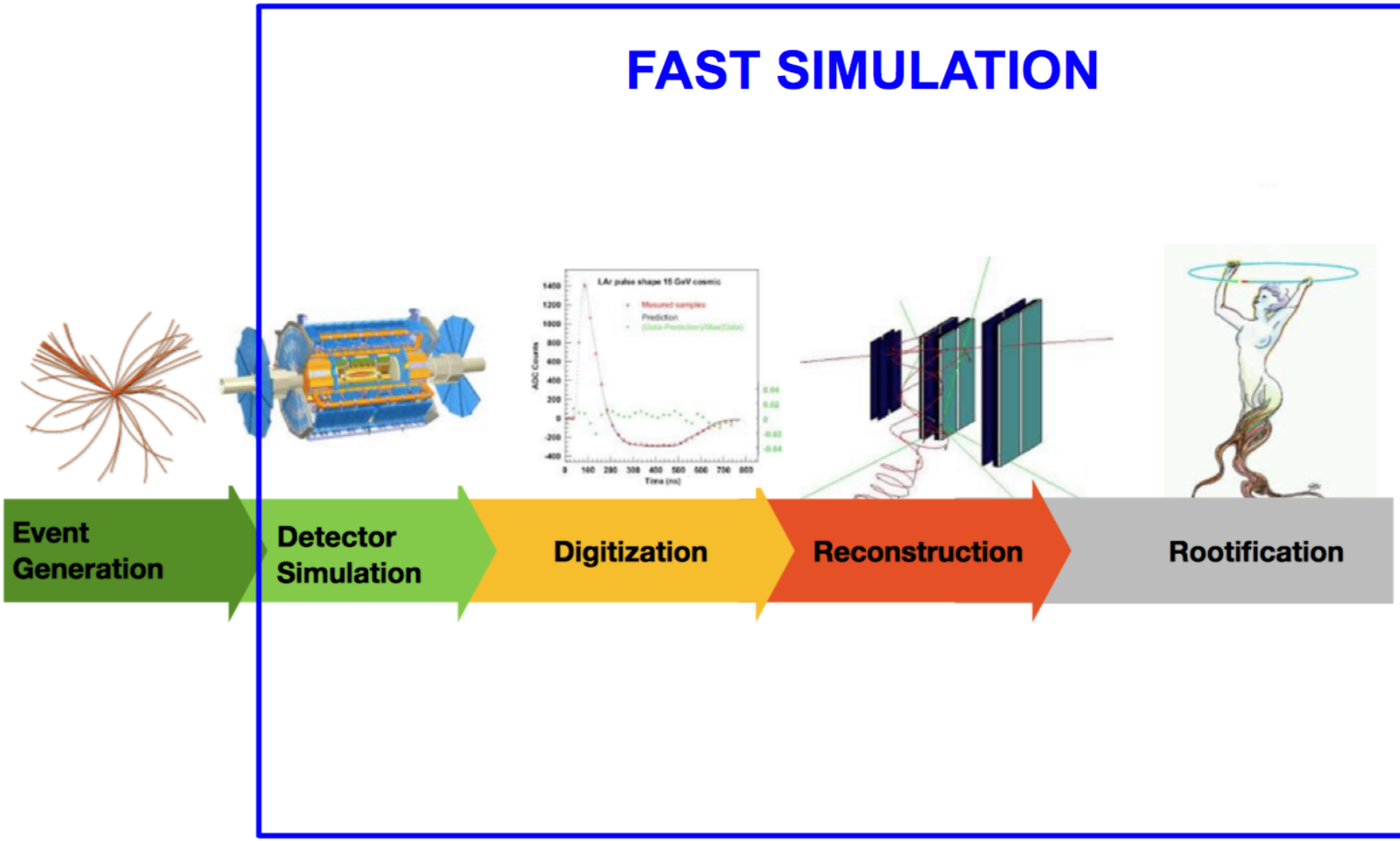
Particle Physics Detector: CMS at LHC



Typical Monte Carlo chain



MC with Fast simulation



Detector simulation

- Full simulation (GEANT):

  - **simulates** particle-matter interaction (including e.m. showering, nuclear int., brehmstrahlung, photon conversions, etc ...) → 100 s /ev

- Experiment Fast simulation (ATLAS, CMS ...):

  - **simplifies** and makes faster simulation and reconstruction → 1 s /ev

- Parametric simulation:

  Delphes, PGS:

  - **parameterize** detector response, reconstruct complex objects → 10 ms /ev

Why Fast Simulation?

- When to use FastSim?

  → test your model with detector simulation
  → more advanced than parton-level studies
  → **sensitive to acceptance and complex observable (Jets,MET)**
  → scan big parameter space (SUSY-like)
  → preliminary tests of new geometries/resolutions (jet substructure)
  → educational purpose (bachelor/master thesis)

- When not to use FastSim?

  → high precision studies
  → very exotic topologies (heavy stable charged particles)

The Delphes project: A bit of history

- Delphes project started back in 2007 at UCL as a side project to allow quick feasibility studies

- Since 2009, its development is **community-based**
  - **ticketing system** for improvement and bug-fixes
    → user proposed patches
  - **Quality control** and **core development** is done at the UCL

- In 2013, **DELPHES 3** was released:
  - modular software
  - new features
  - also included in MadGraph suite (and interfaced with Pythia8)

- **Widely** tested and used by the community (pheno, Snowmass, CMS ECFA efforts, recasting ...)

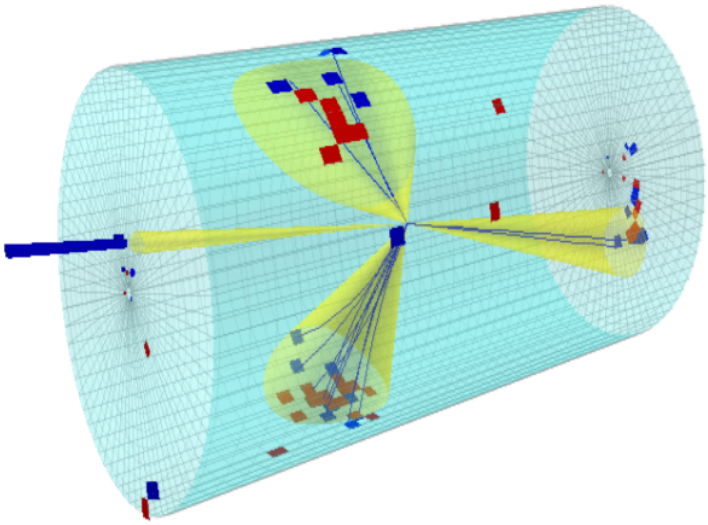
- Website and manual: https://cp3.irmp.ucl.ac.be/projects/delphes

- Paper: JHEP 02 (2014) 057

The Delphes project: Delphes in a nutshell

- **Delphes** is a **modular framework** that simulates of the response of a multipurpose detector in a **parameterized** fashion

- **Includes**:
  - pile-up
  - charged particle **propagation** in magnetic field
  - electromagnetic and hadronic **calorimeters**
  - **muon** system



- **Provides**:
  - leptons (electrons and muons)
  - photons
  - jets and missing transverse energy (particle-flow)
  - taus and b's

Delphes Framework: hands-on

Setup 1: Delphes Framework

1. Create a tutorial directory

```
$ mkdir DelphesTutorial
$ cd DelphesTutorial
```

2. Get Delphes

```
$ wget http://cp3.irmp.ucl.ac.be/downloads/Delphes-3.3.2.tar.gz
$ tar -zxf Delphes-3.3.2.tar.gz
$ mv Delphes-3.3.2 delphes
$ cd delphes
```

3. Install it

```
$ ./configure
$  make -j 4          <- option for using 4 cpus
```

Setup 2: Delphes Framework

1. Download examples

   Z' (m=2 TeV) to WW events in stdhep format

   ```
   $ wget http://cp3.irmp.ucl.ac.be/downloads/delphes_tuto/pp_zp_ww.hep.gz
   $ gunzip pp_zp_ww.hep.gz
   ```

   Di-jet (pT > 1 TeV) events in stdhep format

   ```
   $ wget http://cp3.irmp.ucl.ac.be/downloads/delphes_tuto/pp_jj.hep.gz
   $ gunzip pp_jj.hep.gz
   ```

2. OK, let's run Delphes with below executables:

   DelphesHepMC -> to be used on HepMC input format (*.hepmc)

   DelphesSTDHEP -> to be used on STDHEP input format (*.hep)

Setup 3: Delphes Framework

1. To run on our your input file, type

   $ ./DelphesSTDHEP **cards/delphes_card_CMS.tcl** your_output_for_pp_zp_ww.root **pp_zp_ww.hep**

   where,

   cards/delphes_card_CMS.tcl : CMS detector card

   pp_zp_ww.hep : STDHEP input physics process data format

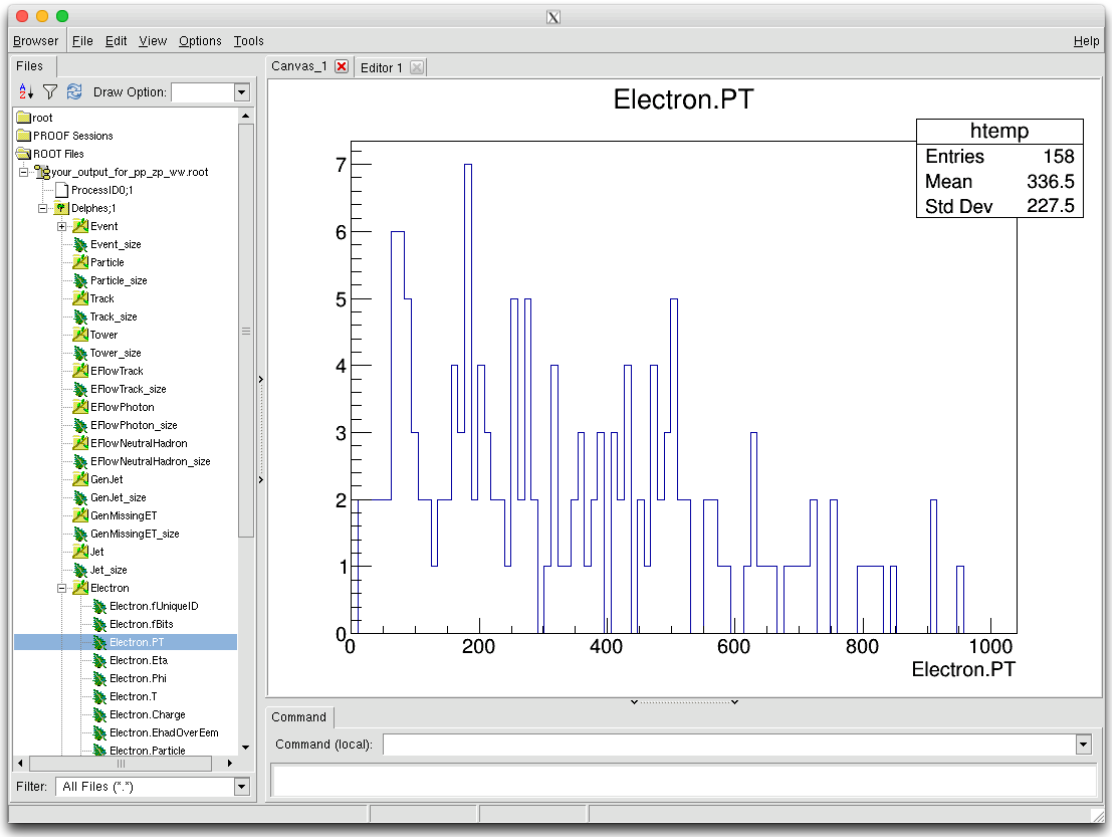
2. Open freshly produced Delphes output with ROOT, and explore it.

   $ root -l your_output_for_pp_zp_ww.root

   root [0] TBrowser t

Setup 3: Delphes Framework

1. In the browser, double click on the "your_output_for_pp_zp_ww.root",
2. then on the "Delphes" tree, play around by double clicking on the various branches/observables.



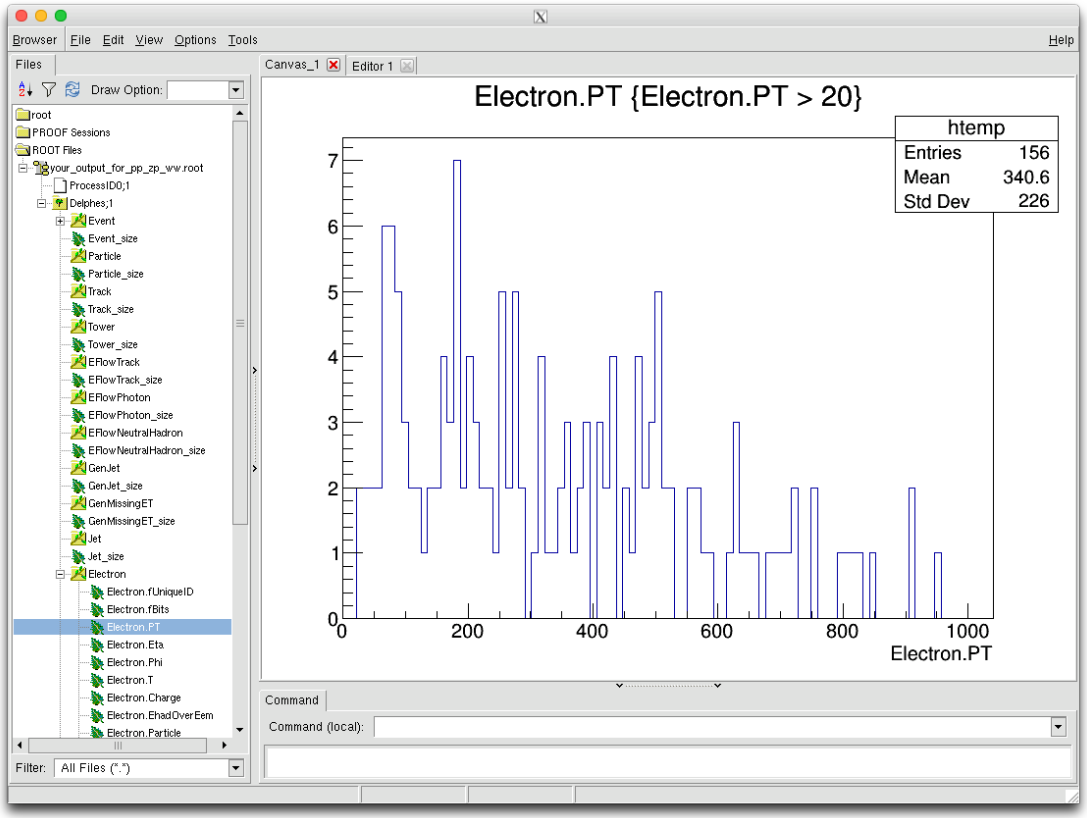
Setup 4: Delphes Framework

1. You can then plot important observable with a simple selection with the following syntax:

```
root [1]  Delphes->Draw("Muon.PT", "Muon.PT > 20");
root [2]  Delphes->Draw("Electron.PT", "Electron.PT > 20");
```

Muon/Electron
-> branch name

PT : variable (leaf) of this branch

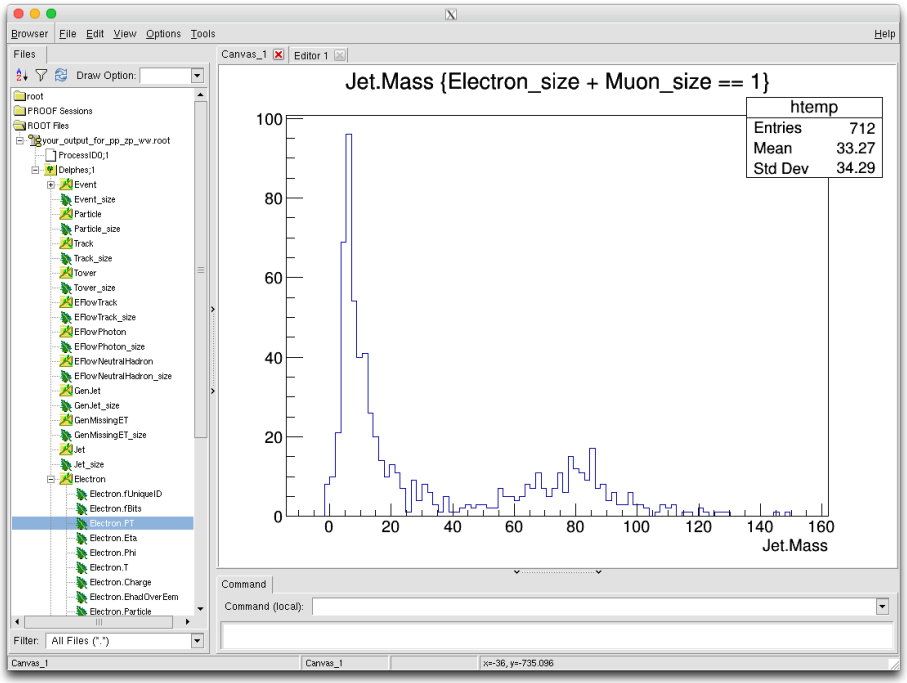


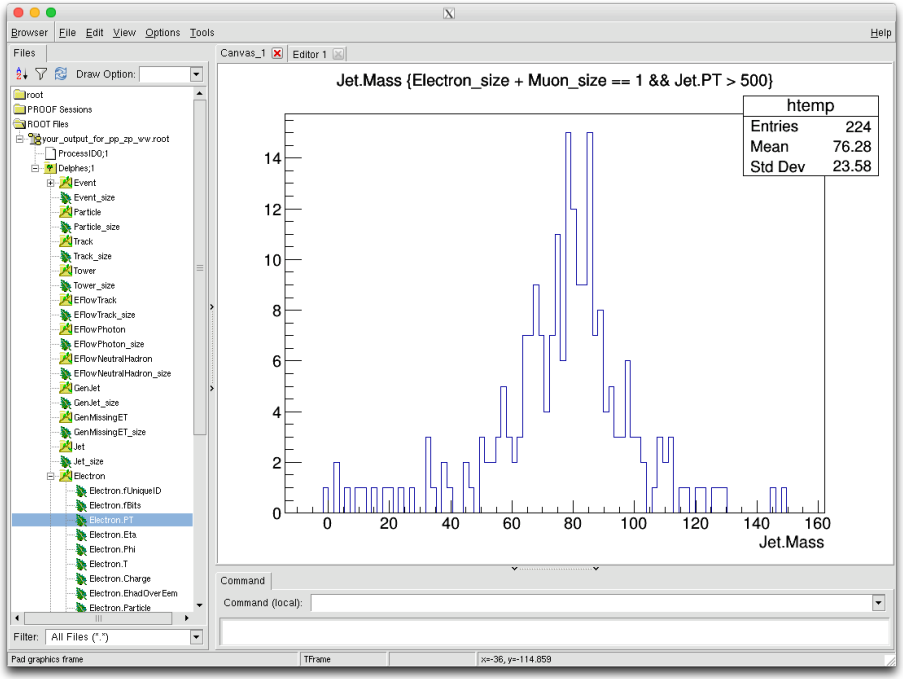
Setup 5: Delphes Framework

1. You can do also something like these:

root [1] Delphes->Draw("Jet.Mass","Electron_size + Muon_size == 1");

root [2] Delphes->Draw("Jet.Mass","Electron_size + Muon_size == 1 && Jet.PT > 500");



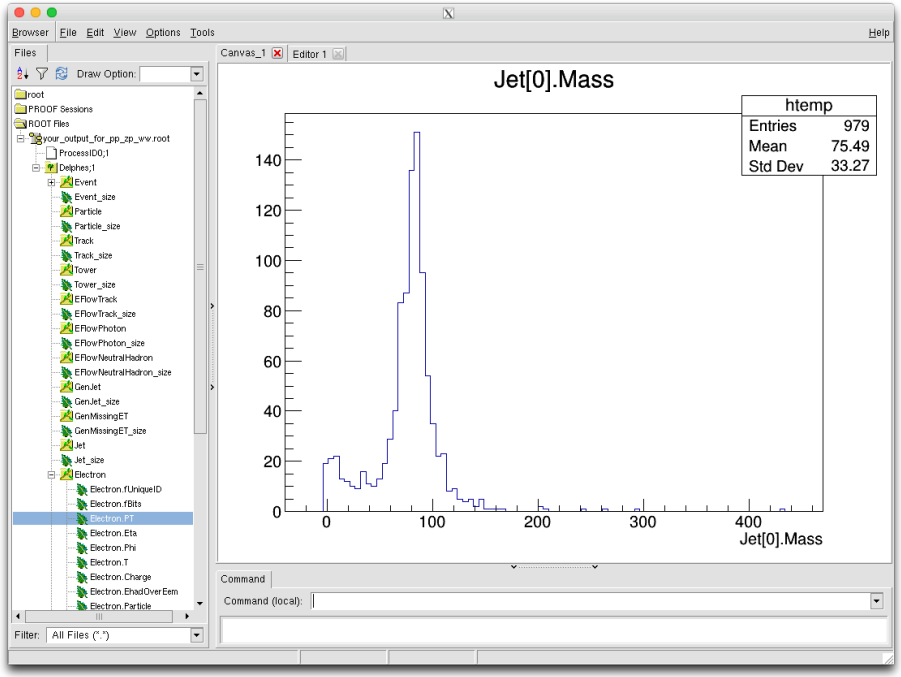


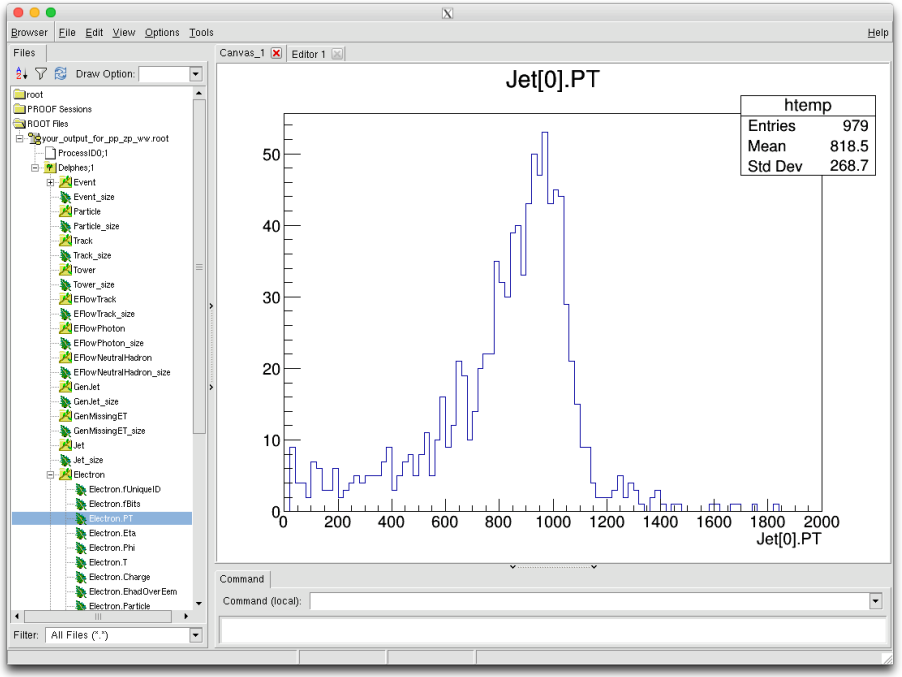
Setup 6: Delphes Framework

1. Objects are already ordered in PT, you can then plot the leading jet observables in this way:

```
root [1]  Delphes->Draw("Jet[0].Mass");
root [2]  Delphes->Draw("Jet[0].PT");
```

Enjoy!





Configuration file in Delphes

The delphes card can be schematically divided in three parts:

1. The ExecutionPath is where the simulation/reconstruction sequence of modules is defined
2. The list of modules configurations.
3. The TreeWriter, where the user defines which objects he stores in the output tree.

You can find an explanation for most Delphes modules here:

https://cp3.irmp.ucl.ac.be/projects/delphes/wiki/WorkBook/Modules

Configuration file in Delphes

1. Open the card with vi editor and try to make sense of it.

cards/delphes_card_CMS.tcl

2. Then configure the FastJetFinder module by switching on the options for jet substructure:

```
#############
  Jet finder
#############

module FastJetFinder FastJetFinder {
  set InputArray EFlowMerger/eflow

  set OutputArray jets

  # algorithm: 1 CDFJetClu, 2 MidPoint, 3 SIScone, 4 kt, 5 Cambridge/Aachen, 6 antikt
  set JetAlgorithm 5
  set ParameterR 0.8

  set ComputeNsubjettiness 1
  set Beta 1.0
  set AxisMode 4

  set ComputeTrimming 1
  set RTrim 0.2
  set PtFracTrim 0.05

  set ComputePruning 1
  set ZcutPrun 0.1
  set RcutPrun 0.5
  set RPrun 0.8

  set ComputeSoftDrop 1
  set BetaSoftDrop 0.0
  set SymmetryCutSoftDrop 0.1
  set R0SoftDrop 0.8

  set JetPTMin 20.0
}
```

Configuration file in Delphes

1. Then, run Delphes with the newly modified card on both the dijet and Z' samples:

   $ ./DelphesSTDHEP cards/delphes_card_CMS.tcl your_output_for_pp_zp_ww_js.root pp_zp_ww.hep

   $ ./DelphesSTDHEP cards/delphes_card_CMS.tcl your_output_for_pp_jj_js.root pp_jj.hep

2. Now download and run this simple event selection macro.

3. It will plot two crucial jet observables that can help in discriminating between a w-jet and a light jet: the jet mass, and the N-subjettiness ratio tau2/tau1.

   $ wget https://cp3.irmp.ucl.ac.be/projects/delphes/raw-attachment/wiki/WorkBook/Tutorials/Mc4Bsm/macro.C --directory-prefix=examples

   $ root examples/macro.C'("your_outpu_pp_jj_js.root","out_pp_zp_ww_js.root")'

Configuration file in Delphes

1. Edit the configuration file to improve the calorimeter resolution, both energy and angular resolutions (by a factor 20) :

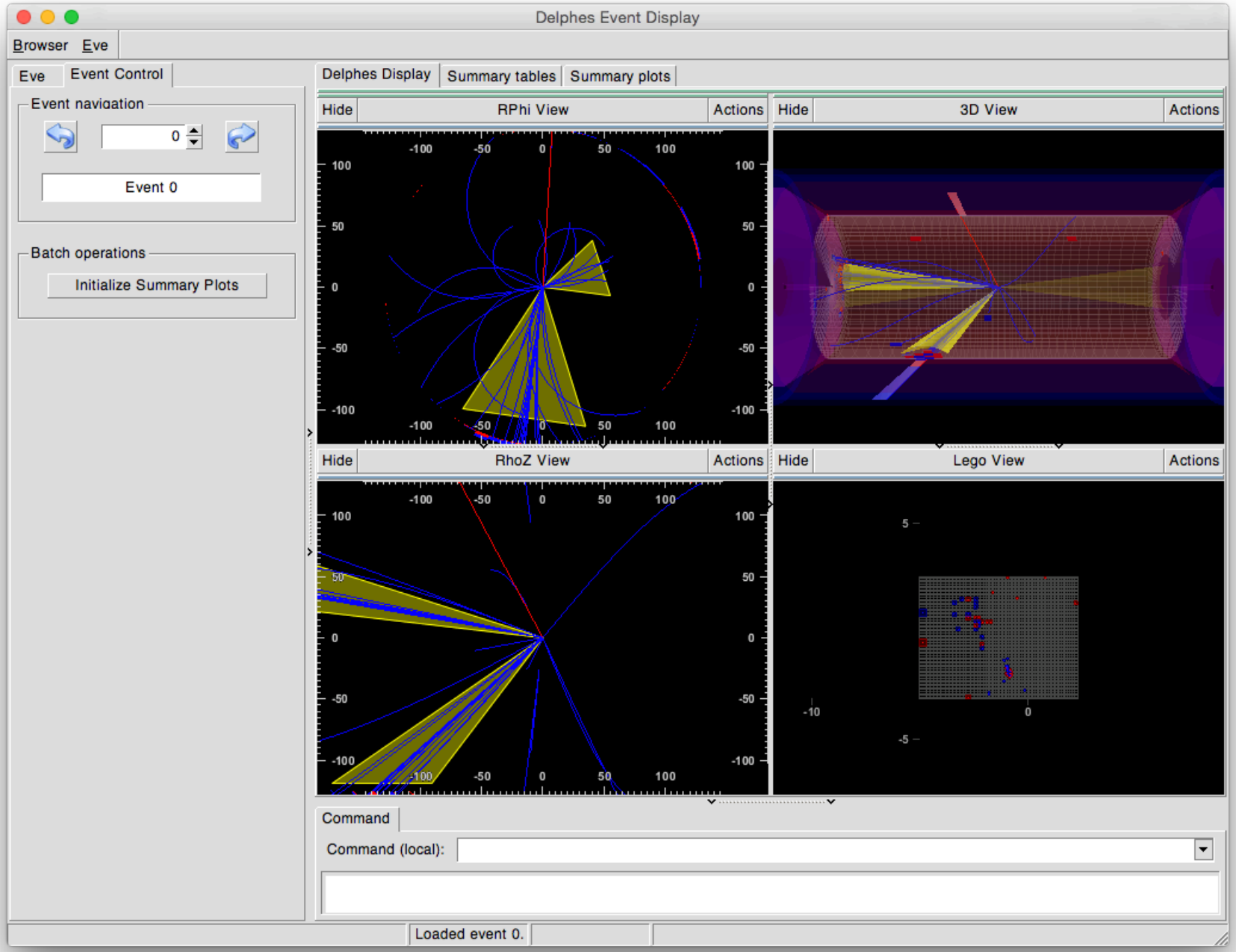
```
#############
# Calorimeter
#############

...


# switching to 0.005 x 0.005 (eta,phi) calorimeter cell size for this tutorial

set PhiBins {}
  for {set i -360} {$i <= 360} {incr i} {
    add PhiBins [expr {$i * $pi/360.0}]
  }

  # 0.01 unit in eta up to eta = 2.5
  for {set i -1000} {$i <= 1000} {incr i} {
    set eta [expr {$i * 0.005}]
    add EtaPhiBins $eta $PhiBins
  }

 ...

# improve the energy resolution by a factor 0.05

  set ECalResolutionFormula {                  (abs(eta) <= 1.5) *0.05*(1+0.64*eta^2) * sqrt(energy^2*0.008^2 + energy*0.11^
                             (abs(eta) > 1.5 && abs(eta) <= 2.5) *0.05* (2.16 + 5.6*(abs(eta)-2)^2) * sqrt(energy^2*0.008^2 +
                             (abs(eta) > 2.5 && abs(eta) <= 5.0) *0.05* sqrt(energy^2*0.107^2 + energy*2.08^2)}

  set HCalResolutionFormula {                  (abs(eta) <= 3.0) *0.05* sqrt(energy^2*0.050^2 + energy*1.50^2) +
                             (abs(eta) > 3.0 && abs(eta) <= 5.0) *0.05* sqrt(energy^2*0.130^2 + energy*2.70^2)}
```

Configuration file in Delphes

1. Re-run Delphes simulation and macro.C with the new configuration and appreciate the effect of improved resolution on the final distributions:

```
$ ./DelphesSTDHEP cards/delphes_card_CMS_nc.tcl your_output_for_pp_zp_ww_js_new.root pp_zp_ww.hep
$ ./DelphesSTDHEP cards/delphes_card_CMS_nc.tcl your_output_for_pp_jj_js_new.root pp_jj.hep
$ root examples/macro.C'("your_output_for_pp_jj_js_new.root","your_output_for_pp_zp_ww_js_new.root")
```

Event display



Madgraph : High Energy Physics event generator

- MadGraph5_aMC@NLO is a framework that aims at providing all the elements necessary for Standard Model (SM) and Beyond SM (BSM) phenomenology, such as
  - the computations of cross sections,
  - the generation of hard events and their matching with event generators,
  - the use of a variety of tools relevant to event manipulation and analysis.
- Processes can be simulated to Leading-Order (LO) accuracy for any user-defined Lagrangian, and the Next-to-Leading-Order (NLO) accuracy in the case of QCD corrections to SM processes. Matrix elements at the tree- and one-loop-level can also be obtained.
- Website: http://madgraph.physics.illinois.edu
  - You need to register, then can get access and download.

PYTHIA : Parton shower simulation

- PYTHIA is a program for the generation of high-energy physics events.
- i.e. for the description of collisions at high energies between elementary particles such as e+, e-, p and pbar in various combinations.
- It contains theory and models for a number of physics aspects,
  - including hard and soft interactions,
  - parton distributions, initial- and final-state parton showers,
  - multiparton interactions,
  - fragmentation and decay.
- It is largely based on original research, but also borrows many formulae and other knowledge from the literature.

Final Project

1. Install analysis framework:

   Madgraph + Pythia + Delphes

1. Generate high energy physics events for signal and background with the Madgraph and Pythia
2. Produce simulation data with Delphes based on CMS detector card.
3. Do data analysis based on what you learned in our class.
   I. Probability distributions
   II. Monte Carlo Method
   III. Fitting Method
   IV. Statistical Method
   V. Etc.

Final Project

1. You should make your small group (for 3 people)
2. We will have presentation from each group every week.
3. You have to give a talk for your status on your project during 4~5 weeks.
   - Each student will have a chance to present at least.
4. Final presentation is expected on Nov. 28 or Dec. 5.