DELPHES, a framework for fast simulation of a general purpose LHC detector

S. Ovyn and X. Rouby*
Center for Particle Physics and Phenomenology (CP3)
Université catholique de Louvain
B-1348 Louvain-la-Neuve, Belgium

severine.ovyn@uclouvain.be, xavier.rouby@cern.ch

Knowing whether theoretical predictions are visible and measurable in a high energy experiment is always delicate, due to the complexity of the related detectors, data acquisition chain and software. We introduce here a new framework, DELPHES, for fast simulation of a general purpose experiment. The simulation includes a tracking system, embedded into a magnetic field, calorimetry and a muon system, and possible very forward detectors arranged along the beamline. The framework is interfaced to standard file formats (e.g. Les Houches Event File) and outputs observable analysis data objects, like missing transverse energy and collections of electrons or jets. The simulation of detector response takes into account the detector resolution, and usual reconstruction algorithms for complex objects, like FASTJET. A simplified preselection can also be applied on processed data for trigger emulation. Detection of very forward scattered particles relies on the transport in beamlines with the HECTOR software. Finally, the FROG 2D/3D event display is used for visualisation of the collision final states. An overview of DELPHES is given as well as a few use-cases for illustration.

Keywords: DELPHES, fast simulation, LHC, smearing, trigger, FASTJET, HECTOR, FROG

1 Introduction

Experiments at high energy colliders are very complex systems, in several ways. First, in terms of the various detector subsystems, including tracking, central calorimetry, forward calorimetry, and muon chambers, with different principles, technologies, geometries and

*Now in Physikalisches Institut, Albert-Ludwigs-Universität Freiburg

sensitivities. Then, due to the requirement of a highly effective online selection (i.e. a *trigger*) which is subdivided into several levels for an optimal reduction factor, but based only on partial data. Finally, in terms of the experiment software with different data format (like *raw* or *reconstructed* data), many reconstruction algorithms and particle identification schemes.

This complexity is handled by large collaborations of thousands of people, which restrict the availability of the data, software and documentation to their members. Real data analyses require a full detector simulation, including the various detector inefficiencies, the dead material, the imperfections and the geometrical details. Moreover, detector calibration and alignment are crucial. Such simulation is very complicated, technical and slow. On the other hand, phenomenological studies, looking for the observability of given signals, may require only fast but realistic estimates of the observables.

A new framework, called DELPHES [1], is introduced here, for the fast simulation of a general purpose collider experiment. Using the framework, observables can be estimated for specific signal and background channels, as well as their production and measurement rates, under a set of assumptions. Starting from the output of event generators, the simulation of the detector response takes into account the subdetector resolutions, by smearing the kinematical properties of the visible final particles. Tracks of charged particles and calorimetric towers are then created.

DELPHES includes the most crucial experimental features, like (1) the geometry of both central or forward detectors; (2) lepton isolation; (3) reconstruction of photons, leptons, jets, b-jets, τ-jets and missing transverse energy; (4) trigger emulation and (5) an event display (Fig. 1).

Although this kind of approach yields much realistic results than a simple "parton-level" analysis, a fast simulation comes with some limitations. Detector geometry is idealised, being uniform, symmetric around the beam axis, and having no craks nor dead material. Secondary interactions, multiple scatterings, photon conversion and bremsstrahlung are also neglected.

The simulation package proceeds in two stages. The first part is executed on the generated events. "Particle-level" informations are read from input files and stored in a GEN ROOT tree. Three varieties of input files can currently be used as input in DELPHES. In order to process events from many different generators, the standard Monte Carlo event structure StdHep can be used as an input. Besides, DELPHES can also provide detector response for events read in "Les Houches Event Format" (LHEF) and ROOT files obtained using the **h2root** converter program. This first stage is performed using three C++ classes: `HEPEVTConverter`, `LHEFConverter` and `STDHEPConverter`. Afterwards, DELPHES performs a simple trigger simulation and reconstruct "high-level objects". These informations are organised in classes and each objects are ordered with respect to the transverse momentum. The output of the various C++ classes is stored in the *Analysis* tree. The program is driven by a datacard (data/DataCardDet.dat) which allow a large spectrum of running conditions by modifying basic detector parameters, including calorimeter and tracking coverage and resolution, thresholds or jet algorithm parameters.

Usual algorithms are applied for the reconstruction of complex objects, like the

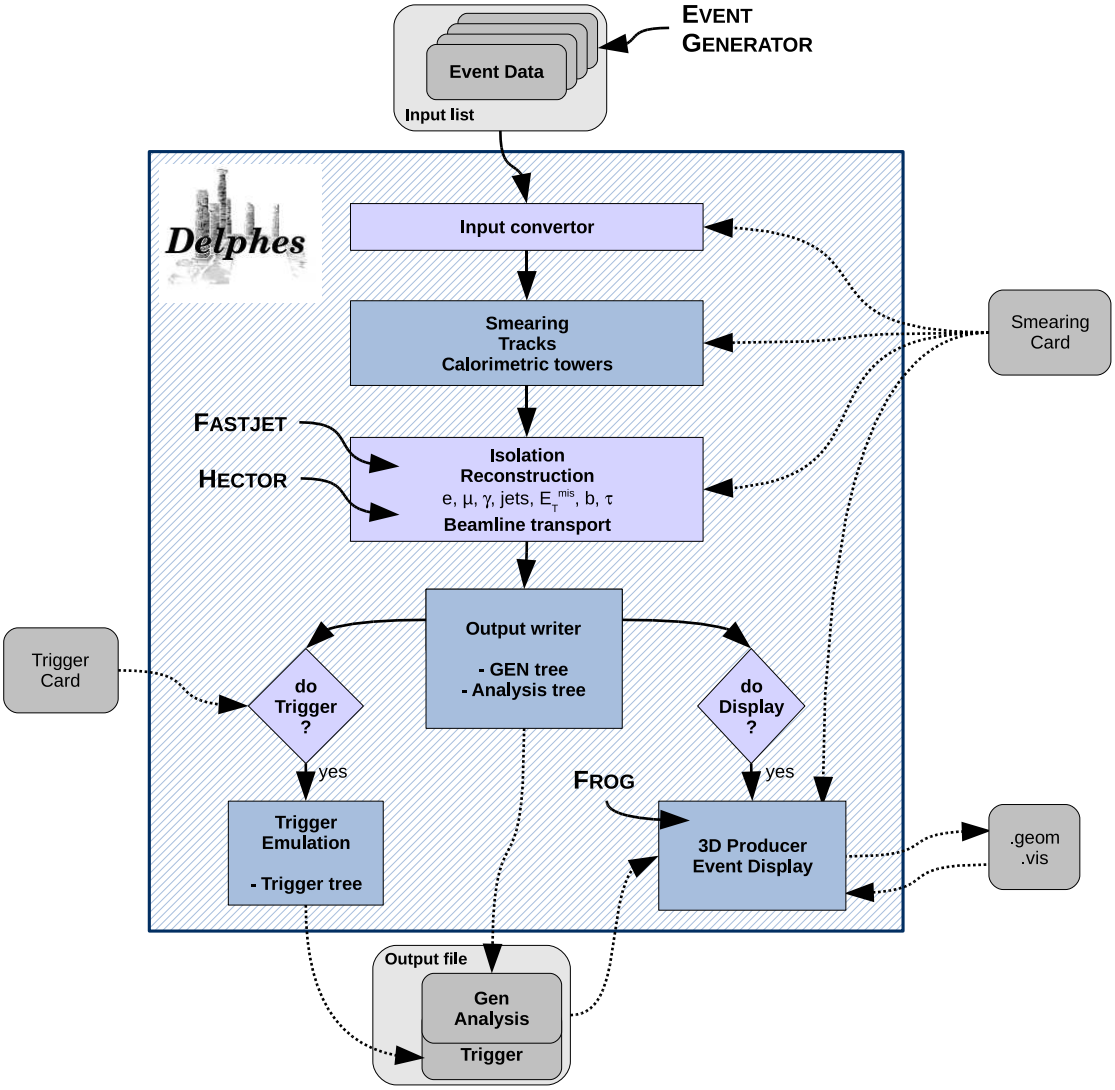

Figure 1: Flow chart describing the principles behind DELPHES.

missing transverse energy or the jets originating from b quarks or τ leptons.

A simplified preselection can also be applied on processed data for trigger emulation. All detectors are assumed to be symmetric with respect to the beam axis

2 Central detector simulation

The overall layout of the general purpose detector simulated by DELPHES is shown in figure 2. A central tracking system surrounded by an electromagnetic and a hadron calorimeters. A forward calorimeter ensures a larger geometric coverage for the mea-

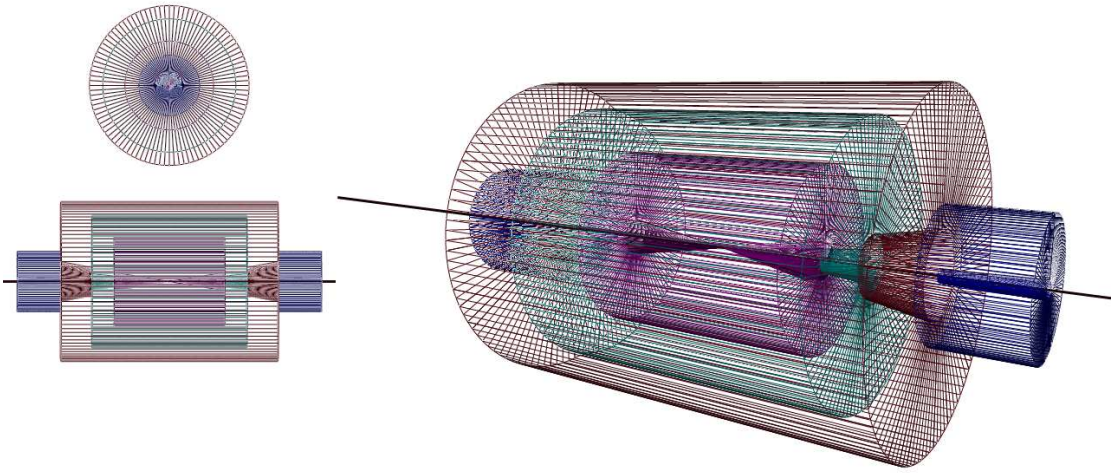

Figure 2: Layout of the generic detector geometry assumed in DELPHES. The innermost layer, close to the interaction point, is a ce ntral tracking system (pink). It is surrounded by a central calorimeter volume (green) with both electromagnetic and hadronic sections. Th e outer layer of the central system (red) consist of a muon system. In addition, two end-cap calorimeters (blue) extend the pseudorapidity coverage of the central detector. The actual detector granularity and extension is defined in the user-configuration card. The detector i s assumed to be strictly symmetric around the beam axis (black line). Additional forward detectors are not depicted.

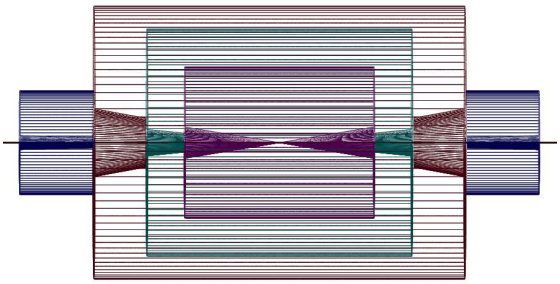

Figure 3: Profile of the layout assumed in DELPHES. The extension of the various subdetectors, as defined in Tab. 2 , are clearly visible. Same colour codes as for Fig. 2 are applied. Additional forward detectors are not depicted.

surement of the missing transverse energy. The fast simulation of the detector response takes into account geometrical acceptance of sub-detectors and their finite resolution.

No smearing is applied on particle direction. (???)

Before starting to loop over events, the `RESOLution` class loads all sub-detector

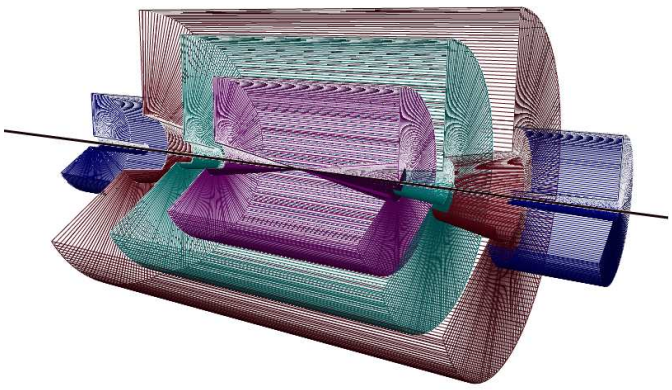

Figure 4: Layout of the generic detector geometry assumed in DELPHES. Open 3D-view of the detector with solid volumes. Same colour codes as for Fig. 2 are applied. Additional forward detectors are not depicted.

resolutions and coverage from the detector parameter file. If no such file is provided, predifined values are used. The coverage of the various sub-systems used in the default configuration are summarized in table 2.

Tracking	`CEN_max_tracker`	2.5
Calorimeters	`CEN_max_calo_cen`	3.0
	`CEN_max_calo_fwd`	5.0
Muon	`CEN_max_mu`	2.4

2.1 Simulation of calorimeters response

The energy of all particle considered as stable in the generator particle list are smeared according to a resolution depending which sub-calorimeter is assumed to be used for the energy measurement. For particles with a short lifetime such as the K_s, the fraction of electromagnetic or hadronic energy is determined according to its decay products. The response of the each sub-calorimeter is parametrized as a function of the energy

$$\frac{\sigma}{E} = \frac{S}{\sqrt{E}} \oplus \frac{N}{E} \oplus C, \tag{1}$$

where S is the stochastic term, N the noise and C the constant term.

The response of the detector is applied to the electromagnetic and the hadronic particles through the `SmearElectron`and `SmearHadron` functions. The 4-momentum p^μ are smeared with a parametrisation directly derived from the detector techinal designs.

In the default parametrisation, the calorimeter is assumed to cover the pseudorapidity range $|\eta| < 3$ and consists in an electromagnetic and an hadronic part. Coverage between pseudorapidities of 3.0 and 5.0 is provided by a forward calorimeter. The response of this calorimeter can be different for electrons and hadrons. The default values of the stochastic, noisy and constant terms as well as the "Card flag" names used in the configuration file are given in table 2.1.

Resolution Term		Card flag	Value
Central ECAL	S	`ELG_Scen`	0.05
	N	`ELG_Ncen`	0.25
	C	`ELG_Ccen`	0.0055
Forward ECAL	S	`ELG_Sfwd`	2.084
	N	`ELG_Nfwd`	0.0
	C	`ELG_Cfwd`	0.107
Central HCAL	S	`HAD_Shcal`	1.5
	N	`HAD_Nhcal`	0.
	C	`HAD_Chcal`	0.05
Forward HCAL	S	`HAD_Shf`	2.7
	N	`HAD_Nhf`	0.
	C	`HAD_Chf`	0.13

The energy of electron and photon particles found in the particle list are smeared using the ECAL resolution terms. Charged and neutral final state hadrons interact with the ECAL, HCAL and the forward calorimeter. Some long-living particles, such as the K_s, possessing lifetime $c\tau$ smaller than 10 mma are considering as stable particles although they decay in the calorimeters. The energy smearing of such particles is performed using the expected fraction of the energy, determined according to their decay products, that whould be deposited into the ECAL (E_{ecal}) and into the HCAL (E_{hcal}). Defining F as the fraction of the energy leading to a HCAL deposit, the two energy values are given by

$$E_{hcal} = E \times F \text{ and } E_{ecal} = E \times (1 - F), \tag{2}$$

where $0 \leq F \leq 1$. The electromagnetic part is handled as the electrons, while the resolution terms used for the hadronic part are `HAD_Shcal`, `HAD_Nhcal` and `HAD_Chcal`. The resulting final energy given after the application of the smearing is then $E = E_{hcal}+E_{ecal}$.

2.2 Muon smearing

Muons candidates are searched The smearing ot the muon 4-momentum p^μ is given by a Gaussian smearing of the p_T function `SmearMuon`. Only the p_T is smeared, but neither η nor ϕ.

2.3 Tracks reconstruction

All stable charged particles lying inside the fiducial volume of the tracking coverage provide a track. The reconstructio efficiency is manageable in the input datacard through the `TRACKING_EFF` term. By default, a track is assumed to be reconstructed with 90% probability.

2.4 Calorimetric towers

All undecayed particles, except muons and neutrinos are producing a calorimetric tower. The same particles enter in the calculation of the missing transverse energy. *what is used is the particle smeared momentum, not the calorimetric towers!*

2.5 Isolated lepton reconstruction

Photon and electron candidates are reconstructed if they fall into the acceptance of the tracking system and have a transverse momentum above the `ELEC_pt` value (10 GeV by default). Muons candidates are searched

Lepton isolation demands that there is no other charged particles with $p_T > 2$ GeV within a cone of $\Delta R < 0.5$ around the lepton.

2.6 Very forward detectors simulation

Some subdetectors have the ability to measure the time of flight of the particle. This correspond to the delay after which the particle is observed in the detector, after the bunch crossing. The time of flight measurement of ZDC and FP420 detector is implemented here. For the ZDC, the formula is simply

$$t_2 = t_1 + \frac{1}{v} \times \left(\frac{s - z}{\cos\theta}\right), \tag{3}$$

where t_2 is the time of flight, t_1 is the true time coordinate of the vertex from which the particle originates, v the particle velocity, s is the ZDC distance to the interaction point, z is the longitudinal coordinate of the vertex from which the particle comes from, *theta* is the particle emission angle. This assumes that the neutral particle observed in the ZDC is highly relativistic, i.e. travelling at the speed of light c. We also assume that $\cos\theta = 1$, i.e. $\theta \approx 0$ or equivalently η is large. As an example, $\eta = 5$ leads to $\theta = 0.013$ and $1 - \cos\theta < 10^{-4}$. The formula then reduces to

$$t_2 = \frac{1}{c} \times (s - z) \tag{4}$$

NB : for the moment, only neutrons and photons are assumed to be able to reach the ZDC. All other particles are neglected

To fix the ideas, if the ZDC is located at $s = 140$ m, neglecting z and θ, and assuming that $v = c$, one gets $t = 0.47$ μs.

3 "High-level" objects reconstruction

3.1 Jet reconstruction

Jets are reconstructed using a cone algorithm with $R = 0.7$ and make only use of the smeared particle momenta. The reconstructed jets are required to have a transverse momentum above 20 GeV and $|\eta| < 3.0$. A jet is tagged as b-jets if its direction lies in the acceptance of the tracker, $|\eta| < 0.5$, and if it is associated to a parent b-quark. A b-tagging efficiency of 40% is assumed if the jet has a parent b quark. For c-jets and light/gluon jets, a fake b-tagging efficiency of 10% and 1% respectively is assumed.

3.2 b-tagging

The simulation of the b-tagging is based on the detector efficiencies assumed (1) for the tagging of a b-jet and (2) for the mis-identification of other jets as b-jets. This relies on the TAGGING_B, MISTAGGING_C and MISTAGGING_L constants, for (respectively) the efficiency of tagging of a b-jet, the efficiency of mistagging a c-jet as a b-jet, and the efficiency of mistatting a light jet (u,d,s,g) as a b-jet. The (mis)tagging relies on the particle ID of the most energetic particle within a cone around the observed (eta,phi) region, with a radius CONERADIUS.

3.3 Tau identification

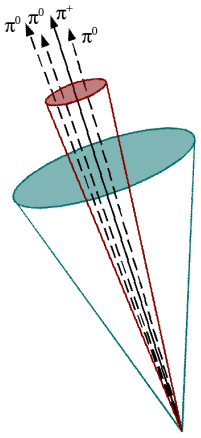

Figure 5: detectorAng.eps

Jets originating from τ-decay are identified using an identification procedure consistent with the one applied in a full detector simulation. The tagging rely on two tau properties. First, in roughly 75% of the time, the hadronic τ-decay products contain only one charged hadron and a number of π^0. Second, the particles arisen from the τ-lepton produce narrow jets in the calorimeter.

Electromagnetic collimation

To use the narrowness of the τ-jet, the *electromagnetic collimation* (C_τ^{em}) is defined as the sum of the energy in a cone with $\Delta R =$`TAU_energy_scone` around the jet axis divided by the energy of the reconstructed jet. The energy in the small cone is calculated using the towers objects. To be taken into account a calorimeter tower should have a transverse energy above a given threshold `JET_M_seed`. A large fraction of the jet energy, denominated here with `TAU_energy_frac` is expected in this small cone. The quantity is represented in figure 6 for the default values (see table 3.3).

Figure 6:

τ selection using tracks

Figure 7:

The tracking isolation for the τ identification requires that the number of tracks associated to a particle with $p_T >$ `TAU_track_pt` is one and only one in a cone with $\Delta R =$ `TAU_track_scone`. This cone should be entirely included in the tracker to be taken into account. This procedure selects taus decaying hadronically with a typical efficiency of 60%. Moreover, the minimal p_T of the τ-jet is required to be `TAUJET_pt`(default value: 10 GeV).

Tau definition	Card flag	Value
$\Delta R^{for\ em}$	`TAU_energy_scone`	0.15
min E_T^{tower}	`JET_M_seed`	1.0 GeV
C_τ^{em}	`TAU_energy_frac`	0.95.
$\Delta R^{for\ tracks}$	`TAU_track_scone`	0.4
min p_T^{tracks}	`PTAU_track_pt`	2 GeV

3.4 Transverse missing energy

4 Trigger emulation

5 Validation

6 Visualisation

As an illustration, an associated photoproduction of a W boson and a t quark is shown in Fig. 8. This corresponds to a $pp \to Wt\ +\ p\ +\ X$ process, where the Wt couple is induced by an incoming photon emitted by one interacting proton. This leading proton survives from the photon emission and subsequently from the pp interaction, and is present in the final state. The experimental signature is a lack of hadronic activity in one forward hemisphere, where the surviving proton escapes. The t quark decays into a W and a b. Both W bosons decay into leptons ($W \to \mu\nu_\mu$ and $W \to \tau\nu_\tau$).

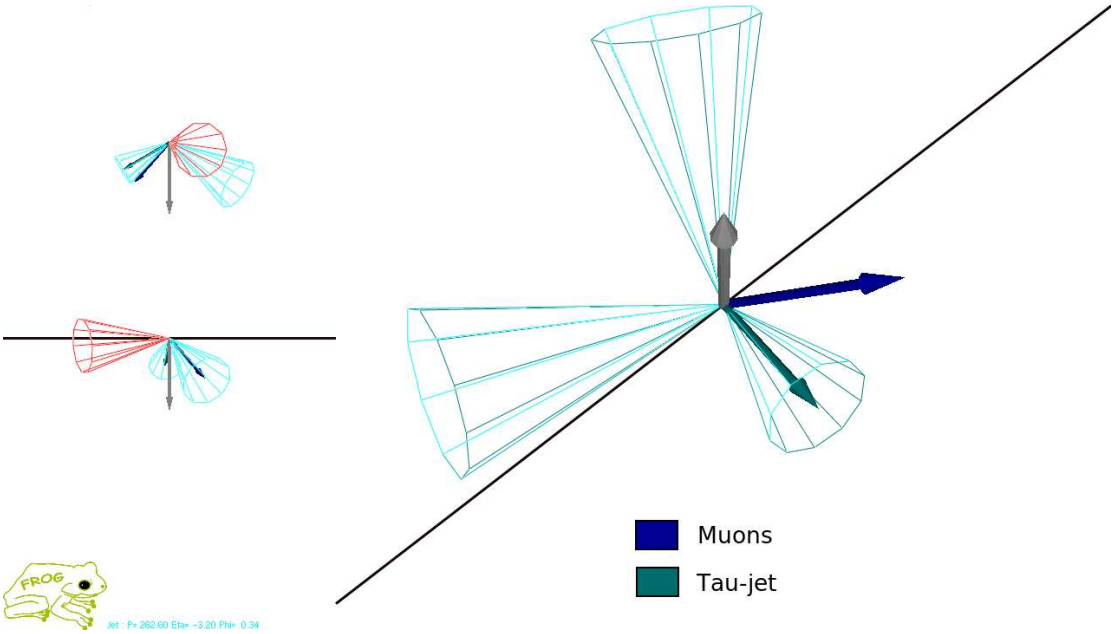

Figure 8: Example of $pp(\gamma p \to Wt)pY$ event. One W boson decays into a μ ν_μ pair and the second one into a τ ν_τ pair. The surviving proton leaves a forward hemisphere with no hadronic activity. The isolated muon is shown as the blue vector. The τ-jet is the cone around the green vector, while the reconstructed missing energy is shown in gray. One jet is visible in one forward region, along the beamline axis, opposite to the direction of the escaping proton.

7 Conclusion and perspectives

A User manual

The available code is a tar file which comes with everything you need to run the DELPHES package. Nevertheless in order to visualise the events with the FROG program, you need to install libraries as explained in *href="http://projects.hepforge.org/frog/*

A.1 Getting started

In order to run DELPHES on your system, first download is sources and compile it:

```
me@mylaptop:~$ wget http://www.fynu.ucl.ac.be/users/s.ovyn/files/Delphes_V_*.*.tar
me@mylaptop:~$ tar -xvf Delphes_V_*.*. tar
me@mylaptop:~$ cd Delphes_V_*.*
me@mylaptop:~$ ./genMakefile.tcl >; Makefile
me@mylaptop:~$ make
```

A.2 Running Delphes on your events

A.2.1 Setting the run configuration

The program is driven by two datacards (default cards are data/DataCardDet.dat and data/trigger.dat) which allow a large spectrum of running conditions. The run card

Contains all needed information to run DELPHES

- The following parameters are available: detector parameters, including calorimeter and tracking coverage and resolution, transverse energy thresholds allowed for reconstructed objects, jet algorithm to use as well as jet parameters.
- Four flags, `FLAG_bfield`, `FLAG_vfd`, `FLAG_trigger` and `FLAG_frog` should be assigned to decide if the magnetic field propagation, the very forward detectors acceptance, the trigger selection and the preparation for FROG display respectively are running by DELPHES.
- An example (the default detector card) can be found in `files/DataCardDet.dat`

The trigger card
Contains the definition of all trigger bits

- Cuts can be applied on the transverse momentum of electrons, muons, jets, tau-jets, photons and transverse missing energy.
- Be careful that the following structured should be used:
  1. One trigger bit per line, the first entry in the line is the name of the trigger bit

2. If the trigger bit uses the presence of multiple identical objects, their transverse momentum thresholds must be defined in decreasing order
3. The different object requirements must be separated by a `&&` flag
4. Example of a trigger bit line:

```
DoubleElec  >> ELEC1_PT: '20' && ELEC2_PT: '10'
```

- An example (the default trigger card) can be found ¡a href="files/trigger.dat" title="Home"¿here¡/a¿¡/li¿

A.2.2 Running the code

Create the above cards (data/mydetector.dat and data/mytrigger.dat) Create a text file containing the list of input files that will be used by DELPHES (with extension *.lhe, *.root or *.hep) To run the code, type the following

```
me@mylaptop:~$ ./Delphes inputlist.list OutputRootFileName.root data/mydetector.dat dat
```

A.3 Running an analysis on your Delphes events

Two examples of codes running on the output root file of DELPHES are coming with the package

1. The `Examples/Analysis_Ex.cpp` code shows how to access the available reconstructed objects and the trigger information The two following arguments are required: a text file containing the input DELPHES root files to run, and the name of the output root file. To run the code:

```
./Analysis_Ex input_file.list output_file.root
```

2. The `Examples/Trigger_Only.cpp` code permits to run the trigger selection separately from the general detector simulation on output DELPHES root files. An input DELPHES root file is mandatory as argument. The new tree containing the trigger information will be added in these file. The trigger datacard is also necessary. To run the code:

```
./Trigger_Only input_file.root data/trigger.dat
```

A.4 Running the Frog event display

- If the `FLAG_frog` was switched on, two files were created during the run of DELPHES: `DelphesToFrog.vis` and `DelphesToFrog.geom`. They contain all the needed information to run frog.
- To display the events and the geometry, you first need to compile FROG. Go to the `Utilities/FROG` and type `make`.
- Go back into the main directory and type `./Utilities/FROG/frog`.

References

[1] DELPHES, hepforge:

[2] FAST-JET,

[3] FROG,

Attention : in SmearUtil::NumTracks, the function arguments 'Eta' and 'Phi' have been switched. Previously, 'Phi' was before 'Eta', now 'Eta' comes in front. This is for consistency with the other functions in SmearUtil. Check your routines, when using NumTracks !

In the list of input files, all files should have the same type

Attention : in SmearUtil::RESOLution::BJets, the maximal energy was looked in CONERADIUS/2 instead of CONERADIUS. This bug has been removed.

Attention : for the tau-jet identification : CONERADIUS /2 was used instead of CONERADIUS !

in other words, the effect related to the particle showers that would happen in the calorimeters are not taken into account. We took the hypothesis that stable particles interacting electromagneticaly deposit their energies in the ECAL calorimeter and that the hadrons just interact with the HCAL