Delphes Hands-on

Pavel Demin, Michele Selvaggi

Université catholique de Louvain
Louvain-la-Neuve, Belgium

Cosmology, Particle Physics and
Phenomenology (CP3)

July 24, 2014

Introduction

- We are presenting the new modular version of Delphes
- Website:

  https://cp3.irmp.ucl.ac.be/projects/delphes
- Documentation for the new modular version:

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

Hands-on session

- Hands-on session outlook:
  - Example 1: Quick start with Delphes
  - Example 2: Simple analysis using TTree::Draw
  - Example 3: Macro-based analysis
  - Example 4: Modifying configuration file
  - Example 5: Adding new module

- Step-by-step instructions can be found on the Indico page of this meeting:
  - Examples 1-4: DelphesTutorial_QuickStart.txt
  - Example 5: DelphesTutorial_NewModule.txt

Example 1: Quick start with Delphes

Installing Delphes from Source

- To successfully build Delphes the following prerequisite packages should be installed:
  - ROOT Data Analysis Framework
  - Tcl scripting language

- Setup ROOT environment variables

```
use root
```

- Get the code:

```
wget http://cp3.irmp.ucl.ac.be/downloads/Delphes-3.1.2.tar.gz

tar -zxf Delphes-3.1.2.tar.gz
```

- Compile:

```
cd Delphes-3.1.2

make
```

- Run:

```
./DelphesSTDHEP examples/delphes_card_CMS.tcl step_1.root
/projet/pth/delphes/Samples/z_ee.hep
```

- Command line parameters:

```
DelphesSTDHEP config_file output_file [input_file(s)]

 config_file - configuration file in Tcl format

 output_file - output file in ROOT format,

 input_file(s) - input file(s) in HepMC format,

 with no input_file, or when input_file is -, read standard input.
```

Running Delphes, cont.

- Running Delphes with HepMC input files:

```
./DelphesHepMC examples/delphes_card_CMS.tcl output.root input.hepmc
```

- Running Delphes with STDHEP (XDR) input files:

```
./DelphesSTDHEP examples/delphes_card_CMS.tcl delphes_output.root input.hep
```

- Running Delphes with LHEF input files:

```
./DelphesLHEF examples/delphes_card_CMS.tcl delphes_output.root input.lhef
```

- Running Delphes with files stored in CASTOR:

```
rfcat /castor/cern.ch/user/d/demine/test.hepmc.gz | gunzip |
./DelphesHepMC examples/delphes_card_CMS.tcl delphes_output.root
```

- Running Delphes with files accessible via HTTP:

```
curl –s http://cp3.irmp.ucl.ac.be/~demin/test.hepmc.gz | gunzip |
./DelphesHepMC examples/delphes_card_CMS.tcl delphes_output.root
```

UCL
Université
catholique
de Louvain

Example 2: Simple analysis using TTree::Draw

ROOT tree: structure

- ROOT tree is a set of branches
- Branch contains a collection of analysis objects for every event:
  - information is stored in TClonesArray, which enables efficient storage an retrieval
  - same C++ classes (Jet, Muon etc.) used for creating the tree and for analyzing the stored data
  - different quantities stored in the tree can be linked by pointers
- ROOT tree description:

  http://cp3.irmp.ucl.ac.be/downloads/RootTreeDescription.html

Example 3: Macro-based analysis

ROOT tree: ExRootTreeReader

- ExRootTreeReader – class simplifying access to ROOT tree data
  - ExRootTreeReader(TTree *) – constructor takes ROOT tree as parameter
  - Long64_t GetEntries() – returns total number of events stored on the tree
  - TClonesArray *UseBranch(branchName, className) – returns pointer to collection of branch elements and specifies branches needed for analysis
  - Bool_t ReadEntry(entry) – loads collections of branch elements with data from specified event

Example 4: Modifying configuration file

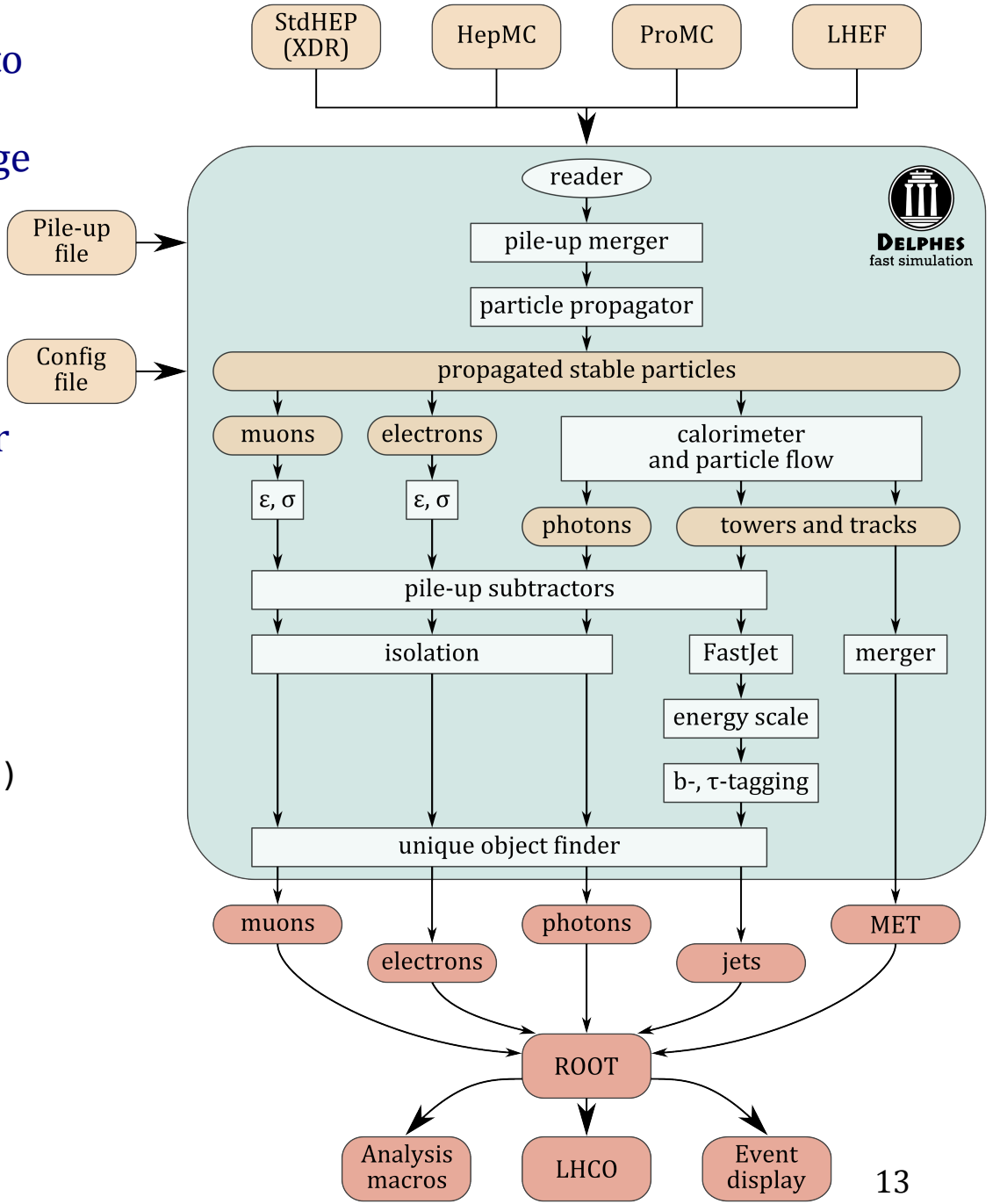
Modular system

- The modular system allows the user to configure and schedule modules via a configuration file, add modules, change data flow, alter output information

- Modules communicate entirely via collections of universal objects (**TObjArray** of **Candidate** four-vector like objects).

- Any module can access TObjArrays produced by other modules using **ImportArray** method:

```
ImportArray("ModuleName/arrayName")
```



Configuration file

- Delphes' configuration file is based on Tcl scripting language
- Basic commands:
  - set value of a scalar parameter:

```
set ParamName value
```

  - append a list of values to a vector (list) parameter

```
add ParamName value1 value2 value3 ...
```

  - activate a module:

```
module ModuleClass ModuleName ModuleConfigurationBody
```

  - define order of execution of various modules:

```
set ExecutionPath ListOfModuleNames

set ExecutionPath {
  ParticlePropagator
  TrackEfficiency
  TrackMomentumSmearing
  ...
  TreeWriter
}
```

Module configuration example

```
module Efficiency ChargedHadronTrackingEfficiency {
  set InputArray ParticlePropagator/chargedHadrons
  set OutputArray chargedHadrons

  # add EfficiencyFormula {efficiency formula as a function of eta and pt}

  # tracking efficiency formula for charged hadrons
  set EfficiencyFormula {
                                                  (pt <= 0.1)   * (0.00) +
                      (abs(eta) <= 1.5) * (pt > 0.1   && pt <= 1.0)   * (0.70) +
                      (abs(eta) <= 1.5) * (pt > 1.0)                  * (0.95) +
    (abs(eta) > 1.5 && abs(eta) <= 2.5) * (pt > 0.1   && pt <= 1.0)   * (0.60) +
    (abs(eta) > 1.5 && abs(eta) <= 2.5) * (pt > 1.0)                  * (0.85) +
    (abs(eta) > 2.5)                                                  * (0.00)
  }
}
```

All efficiency, resolution, etc formulas can be configured as function of E, p_T, η, φ using ROOT's TFormula syntax: http://root.cern.ch/root/html/TFormula.html

TreeWriter module

All the output ROOT tree branches are configured in the TreeWriter module

```
module TreeWriter TreeWriter {
# add Branch InputArray BranchName BranchClass
  add Branch Delphes/allParticles Particle GenParticle

  add Branch TrackMerger/tracks Track Track
  add Branch Calorimeter/towers Tower Tower

  add Branch Calorimeter/eflowTracks EFlowTrack Track
  add Branch Calorimeter/eflowPhotons EFlowPhoton Tower
  add Branch Calorimeter/eflowNeutralHadrons EFlowNeutralHadron Tower

  add Branch GenJetFinder/jets GenJet Jet
  add Branch UniqueObjectFinder/jets Jet Jet
  add Branch UniqueObjectFinder/electrons Electron Electron
  add Branch UniqueObjectFinder/photons Photon Photon
  add Branch UniqueObjectFinder/muons Muon Muon
  add Branch MissingET/momentum MissingET MissingET
  add Branch ScalarHT/energy ScalarHT ScalarHT
}
```

Changing the electron energy smearing

- Now we are going to modify the Delphes configuration file

- and replace resolution formula for electrons with:

```
# resolution formula for electrons
set ResolutionFormula {
                    (abs(eta) <= 3.0) * sqrt(energy^2*0.007^2 + energy*0.07^2 + 0.35^2) +
  (abs(eta) > 3.0 && abs(eta) <= 5.0) * sqrt(energy^2*0.107^2 + energy*2.08^2)
}
```

UCL
Université
catholique
de Louvain

Example 5: Adding new module

Adding new module

- Steps needed for adding new module:
  - copy template/example module to the new one:

```
cp modules/ExampleModule.h modules/NewModule.h
cp modules/ExampleModule.cc modules/NewModule.cc
```

  - edit new module:

```
vi modules/NewModule.h
vi modules/NewModule.cc
```

  - add new module to modules/ModulesLinkDef.h:

```
#include "modules/NewModule.h"
...
#pragma link C++ class NewModule+;
```

  - regenerate Makefile:

```
./configure
```

  - rebuild Delphes:

```
make –j 4
```

- Exercise guidelines can be found in DelphesTutorial_NewModule.txt