Tools for Collider Physics

A practical introduction

KEK and IPMU

Nojiri

Plan of the lectures

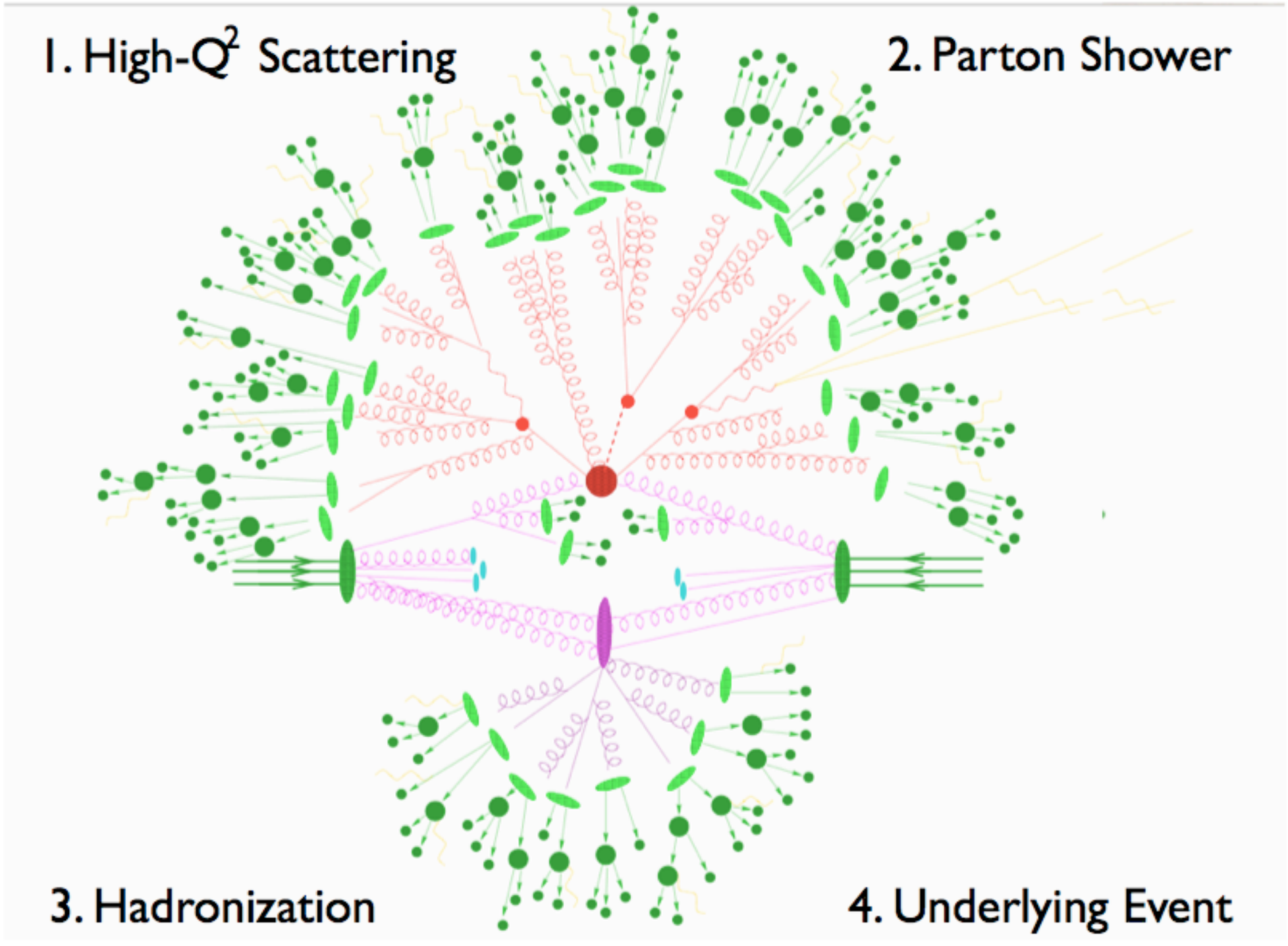
Introduction: The big picture

Infrared Behaviour of QCD

Jet Definitions

Parton Showers

Introduction: The big picture

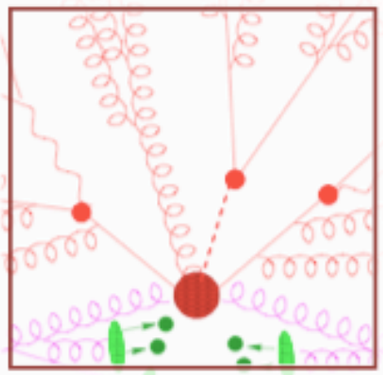


Introduction: The big picture

1. High-Q^2 Scattering

2. Parton Shower

☞ where new physics lies



Parton Level Generator
Feynrules
Madgraph, Calchep

☞ process dependent

☞ first principles description

☞ it can be systematically improved

3. Hadronization

4. Underlying Event

pythia herwig++ take care all steps

Why we need numerical tools

- tree level hard process (easy, maybe analysis)
- Need devoted techniques, experimental data..
  - parton shower + hadronization
  - parton distribution function
  - NLO …
- detector effects

Monte Calro Integration required

Stress free attitude to use open HEP tools

- don’t spend too much time to learn computer languages. Learn from source codes, tutorial file

- You do not have to write big codes from scratch, copy it from your collaborator, and your colleagues.

stress free attitude
for numerical calculations

- Keep original example files
- Do not edit too much lines of the example files at once. Do not generate too much events at once.
- Error messages are always correct. (C++ or clang error messages are very helpful these days.) Read and react them.
- It runs, but results does not seems correct. how to check it.
    - At this level, put in many cout<< val << endl; to see if they behave.
- initialization, definition
- numerial calculations that takes longer than 30 min to get single result is not suited for start up .

main01.cc (pythia8

reading header files=“tools you need”

```
#include "Pythia8/Pythia.h"
using namespace Pythia8;
int main() {
  // Generator. Process selection. LHC initialization. Histogram.
  Pythia pythia;
  pythia.readString("Beams:eCM = 8000.");
  pythia.readString("HardQCD:all = on");
  pythia.readString("PhaseSpace:pTHatMin = 20.");
  pythia.init();
  Hist mult("charged multiplicity", 100, -0.5, 799.5);
  // Begin event loop. Generate event. Skip if error. List first one.
  for (int iEvent = 0; iEvent < 100; ++iEvent) {
    if (!pythia.next()) continue;
    // Find number of all final charged particles and fill histogram.
    int nCharged = 0;
    for (int i = 0; i < pythia.event.size(); ++i)
      if (pythia.event[i].isFinal() && pythia.event[i].isCharged())
        ++nCharged;
    mult.fill( nCharged );
  // End of event loop. Statistics. Histogram. Done.
  }
  pythia.stat();
  cout << mult;
  return 0;
}
```

starting calculation

Make “object” of class called Pythia

reading parameters to set up

pythia.readFile(fileName);

you can also read it file file

initialization

event generation

pythia member function

readString

init() initialization of energy cut..

next() generate next event

event[i] array of particles of an events

→ size() size of the events,

isFinal(final state or not),isCharged

stat() print out statistics

some c++ language

loop over event generation by increasing iEvent from 0 up to 99

```
for (int iEvent = 0; iEvent < 100; ++iEvent) {
   if (!pythia.next()) continue; generate event and skip if it is not successful
   // Find number of all final charged particles and fill histogram.
   int nCharged = 0;
   for (int i = 0; i < pythia.event.size(); ++i)
     if (pythia.event[i].isFinal() && pythia.event[i].isCharged())
       ++nCharged;   if final state and charged increase nCharged by one
   mult.fill( nCharged );
// End of event loop. Statistics. Histogram. Done.
}
```

compile and execute

goto example directory

make main01

./main01 >& test

less test

```
*-------  PYTHIA Process Initialization  ------------------------*
|                                                                |
| We collide p+ with p+ at a CM energy of 8.000e+03 GeV          |
|                                                                |
|----------------------------------------------------------------|
|                                                  |             |
| Subprocess                                  Code |   Estimated |
|                                                  |    max (mb) |
|                                                  |             |
|----------------------------------------------------------------|
|                                                  |             |
| g g -> g g                                   111 |   1.403e+00 |
| g g -> q qbar (uds)                          112 |   1.817e-02 |
| q g -> q g                                   113 |   1.010e+00 |
| q q(bar)' -> q q(bar)'                       114 |   1.100e-01 |
| q qbar -> g g                                115 |   8.378e-04 |
| q qbar -> q' qbar' (uds)                     116 |   3.698e-04 |
| g g -> c cbar                                121 |   5.988e-03 |
| q qbar -> c cbar                             122 |   1.225e-04 |
| g g -> b bbar                                123 |   5.400e-03 |
| q qbar -> b bbar                             124 |   1.160e-04 |
|                                                  |             |
*-------  End PYTHIA Process Initialization ----------------------*
```

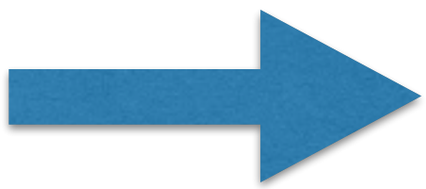
event information and final cross sections

```
--------  PYTHIA Event Listing  (hard process)  -----------------------------------------------------------------------

    no        id   name            status     mothers    daughters     colours      p_x        p_y        p_z        e          m
     0        90   (system)           -11     0     0     0     0     0     0       0.000      0.000      0.000   8000.000   8000.000
     1      2212   (p+)               -12     0     0     3     0     0     0       0.000      0.000   4000.000   4000.000      0.938
     2      2212   (p+)               -12     0     0     4     0     0     0       0.000      0.000  -4000.000   4000.000      0.938
     3        21   (g)                -21     1     0     5     6   102   103       0.000      0.000     20.290     20.290      0.000
     4         4   (c)                -21     2     0     5     6   101     0       0.000      0.000    -24.080     24.080      0.000
     5        21   g                   23     3     4     0     0   101   103      19.321      7.734      5.506     21.528      0.000
     6         4   c                   23     3     4     0     0   102     0     -19.321     -7.734     -9.296     22.843      1.500
                                  Charge sum:  0.667           Momentum sum:       0.000      0.000     -3.790     44.370     44.208

*-------  PYTHIA Event and Cross Section Statistics  -------------------------------------------------------------*
|                                                                                                                  |
| Subprocess                                    Code |            Number of events        |      sigma +- delta |
|                                                    |       Tried   Selected   Accepted  |     (estimated) (mb) |
|                                                    |                                    |                      |
|--------------------------------------------------------------------------------------------------------------|
|                                                    |                                    |                      |
| g g -> g g                                     111 |         344         60         60 |   2.090e-01  1.650e-02 |
| g g -> q qbar (uds)                            112 |           7          1          1 |   3.748e-03  3.748e-03 |
| q g -> q g                                     113 |         256         33         33 |   1.517e-01  1.357e-02 |
| q q(bar)' -> q q(bar)'                         114 |          19          5          5 |   2.106e-02  5.979e-03 |
| q qbar -> g g                                  115 |           0          0          0 |   0.000e+00  0.000e+00 |
| q qbar -> q' qbar' (uds)                       116 |           0          0          0 |   0.000e+00  0.000e+00 |
| g g -> c cbar                                  121 |           3          1          1 |   2.122e-03  2.122e-03 |
| q qbar -> c cbar                               122 |           0          0          0 |   0.000e+00  0.000e+00 |
| g g -> b bbar                                  123 |           1          0          0 |   0.000e+00  0.000e+00 |
| q qbar -> b bbar                               124 |           0          0          0 |   0.000e+00  0.000e+00 |
|                                                    |                                    |                      |
| sum                                            |         630        100        100 |   3.876e-01  2.260e-02 |
|                                                                                                                  |
*-------  End PYTHIA Event and Cross Section Statistics -------------------------------------------------------*

*-------  PYTHIA Error a
```

process implemented in pythia8 (go to their web page to find detail)

Setup Run Tasks

Save Settings
Main-Program Settings
Beam Parameters
Random-Number Seed
PDF Selection
Master Switches
Process Selection
-- QCD
-- Electroweak
-- Onia
-- Top
-- Fourth Generation
-- Higgs
-- SUSY
-- New Gauge Bosons
-- Left-Right Symmetry
-- Leptoquark
-- Compositeness
-- Hidden Valleys
-- Extra Dimensions
A Second Hard Process
Phase Space Cuts
Couplings and Scales
Standard-Model Parameters



New-Gauge-Boson Processes

This page contains the production of new *Z'^0* and *W'^+-* gauge bosons, e.g. with context of a new *U(1)* or *SU(2)* gauge group, and also a (rather speculative) horiz gauge boson *R^0*. Left-right-symmetry scenarios also contain new gauge bosons described separately.

Z'^0

This group only contains one subprocess, with the full *gamma^*/Z^0/Z'^0* interfer structure for couplings to fermion pairs. It is possible to pick only a subset, e.g, or *Z'^0* piece. No higher-order processes are available explicitly, but the ISR showe automatic matching to the *Z'^0* + 1 jet matrix elements, as for the corresponding *gamma^*/Z^0* process.

`flag` **NewGaugeBoson:ffbar2gmZZprime** (`default = off`)
Scattering *f fbar →Z'^0*. Code 3001.

`mode` **Zprime:gmZmode** (`default = 0; minimum = 0; maximum = 6`)
Choice of full *gamma^*/Z^0/Z'^0* structure or not in the above process. Note that, *Z'^0* part switched off, this process is reduced to what already exists among elect processes, so those options are here only for crosschecks.
`option` **0** : full *gamma^*/Z^0/Z'^0* structure, with interference included.
`option` **1** : only pure *gamma^** contribution.
`option` **2** : only pure *Z^0* contribution.
`option` **3** : only pure *Z'^0* contribution.
`option` **4** : only the *gamma^*/Z^0* contribution, including interference.
`option` **5** : only the *gamma^*/Z'^0* contribution, including interference.
`option` **6** : only the *Z^0/Z'^0* contribution, including interference.
Note: irrespective of the option used, the particle produced will always be assign for *Z'^0*, and open decay channels is purely dictated by what is set for the *Z'^0*.

main14.cc (SUSY はmain24.cc)

```
// Subruns   0 -   5 : QCD jets
//           6 -  10 : prompt photons.
//          11 -  12 : t-channel gamma/Z/W exchange.
//          13 -  23 : gamma*/Z^0/W^+-, singly, in pairs or with parton
//          24 -  25 : onia.
//          26 -  30 : top.
//          31 -  40 : Standard Model Higgs.
//          41 -  45 : MSSM Higgses (trivial couplings).
//          46 -  47 : Z' and W'
//          48 -  51 : Left-right-symmetric scenario.
//          52 -  52 : Leptoquark.
//          53 -  55 : Excited fermions (compositeness).
//          56 -  56 : excited Graviton (RS extra dimensions).
```

```
string processes[61] = { "HardQCD:gg2gg", "HardQCD:gg2qqbar",
    "HardQCD:gg2ccbar",  "HardQCD:gg2bbbar","HardQCD:qg2qg" ,
    "HardQCD:qq2qq", "HardQCD:qqbar2gg", "HardQCD:qqbar2qqbarNew",
    "HardQCD:qqbar2ccbar", "HardQCD:qqbar2bbbar", "PromptPhoton:qg2qgamma",
    "PromptPhoton:qqbar2ggamma", "PromptPhoton:gg2ggamma",
    "PromptPhoton:ffbar2gamma
    "WeakBosonExchange:ff2ff(
    "WeakSingleBoson:ffbar2gm
    "WeakDoubleBoson:ffbar2gm
    "WeakDoubleBoson:ffbar2WW
    "WeakBosonAndParton:qg2gm
    "WeakBosonAndParton:qqbar
    "WeakBosonAndParton:ffbar
    "Top:gg2ttbar",  "Top:qqb
    "Top:ffbar2ttbar(s:gmZ)",
    "HiggsSM:ffbar2H", "Higgs
    "HiggsSM:ffbar2HW", "Higg
    "HiggsSM:qg2Hq", "HiggsSM
    "HiggsSM:qqbar2Hg(l:t)",
    "HiggsBSM:allA3", "HiggsB
    "NewGaugeBoson:ffbar2gmZZ
    "LeftRightSymmmetry:ffbar
```

main14.cc allows to calculate many processes

```
if (iProc > iFirst) for (int i = iBeg[iProc - 1]; i < iBeg[iProc]; ++i)
  pythia.readString( processes[i] + " = off" );
for (int i = iBeg[iProc]; i < iBeg[iProc + 1]; ++i) {
  pythia.readString( processes[i] + " = on" );
  if (i > iBeg[iProc]) cout << " + ";
  cout << processes[i];
}
```

setup before pythia.init()
(example of W' and so on)

- beam energy pythia.readString("Beams:eCM = 8000.");
- process pythia.readString(“NewGaugeBoson:ffbar2Wprime=on”);
- particle mass, decay mode

```
pythia.readString("34:m0 = 2000.");
pythia.readString("34:onMode = off”); switch off decay
pythia.readString("34:onIfAny = 24”); then is final state contain pdg code 24
namely, W
```

- energy of hard interaction, minimum PT. Important to reduce the number of events to be generated.

```
pythia.readString("PhaseSpace:mHatMin = 1300.");
pythia.readString("PhaseSpace:pTHatMin = 500.");
```

advanced setup I

- **choice of parton distribution function**

  - pythia.readString(“PDF:pSet=7”) CTEQ 6L NLO

  using LHAPDF(PDF package)

```
pythia.readString(“PDF:pSet = LHAPDF6:CT10");
pythia.readString("Random:setSeed = on");
```

- external input (specify numbers when you run the code.)

```
int main(int argc, char **argv) {…  char mean character

example:If you do not specify anything, code produce same events. If you
want to make huge number of events, you have to restart your code with
different rundum number seed. Write as follows
  string iseed= argv[1];
  string  dummy="Random:seed = "+ iseed;    // This makes longer “string”
  //cout<< dummy <<endl;
  pythia.readString(dummy); //read the seed setup

final state radiation : pythia.readString(“Tune:pp=11"); //using Tunes
```

exercise

- Choose your process of new physics, change the particle mass, see how cross section reduces with mass.

- Choose the process with missing momentum, (those with Z, W, top …) see how cross section reduces with collision energy cutoff.

advanced set up. save events/compress events.

- save events to use later
- want to interface the events to the other tools
- event generation -> lhe, hepmc files(common formats -> (detector simulation?) -> root files? →more analysis
- History: MC authors to adopt same format so that different tools can cooperate starting from Les Houches workshop

pythia8 -> save to .hepmc files　main41.cc

need new function written in HepMC2.h

```
#include "Pythia8/Pythia.h"
#include “Pythia8Plugins/HepMC2.h"
/ Therefore large event samples may be impractical.


int main() {

  // Interface for conversion from Pythia8::Event to HepMC event.
  HepMC::Pythia8ToHepMC ToHepMC;

  // Specify file where HepMC events will be stored.
  HepMC::IO_GenEvent ascii_io("hepmcout41.dat", std::ios::out);

  for (int iEvent = 0; iEvent < 100; ++iEvent) {
    if (!pythia.next()) continue;
(途中省略)
    // Construct new empty HepMC event and fill it.
    // Units will be as chosen for HepMC build; but can be changed
    // by arguments, e.g. GenEvt( HepMC::Units::GEV, HepMC::Units::MM)
  HepMC::GenEvent* hepmcevt = new HepMC::GenEvent();
    ToHepMC.fill_next_event( pythia, hepmcevt );

    // Write the HepMC event to file. Done with it.
    ascii_io << hepmcevt;
    delete hepmcevt;
  // End of event loop. Statistics. Histogram.
  }
```

tool to convert pythia events to HepMC

name of the file

memory of HepMC event

copy and convert pythia events to hemp

write to hepmcout41.dat file

kill the memory

exercise

- write the events to hepmc file.
- look into the hepmc and guess what is written

C++

Understand “Class”

“Jet”

energy momentum <-Lorentz transformation , mass 、 pT, eta, phi

number of charged track

particle momentum in the Jet

cluster sequence

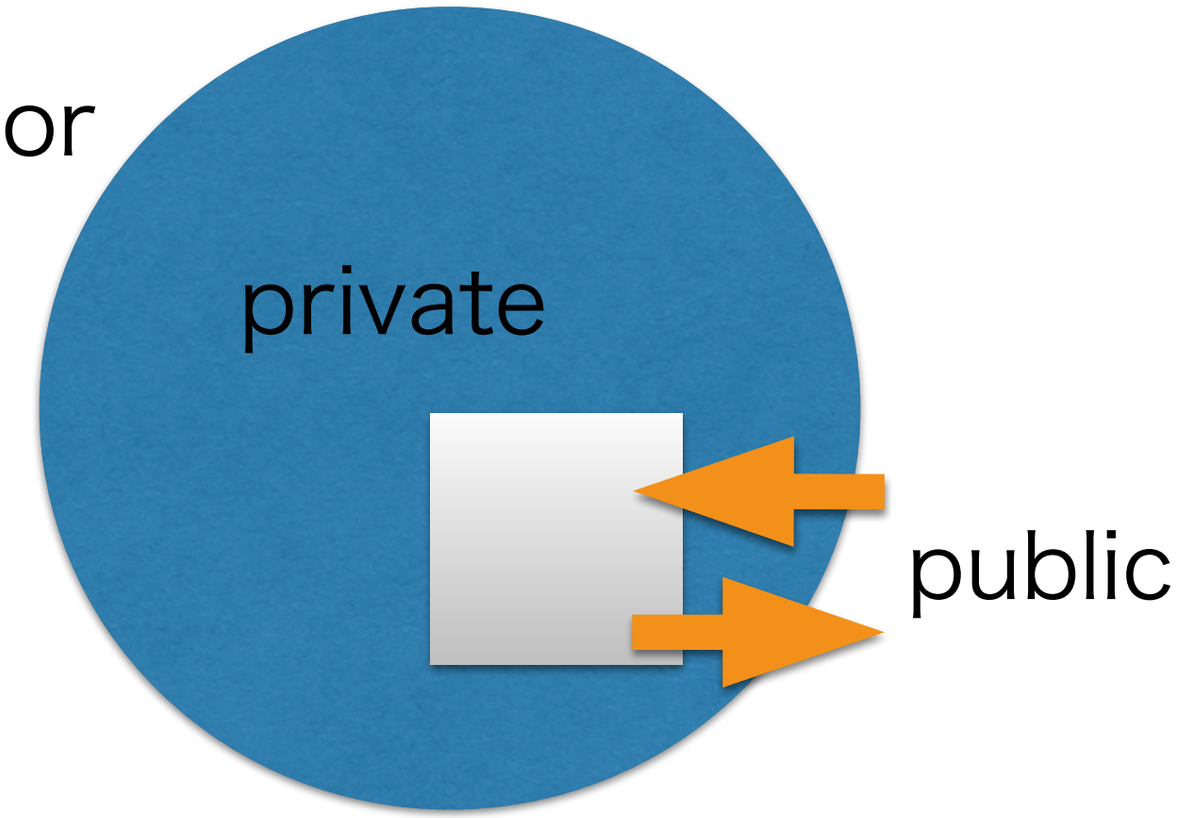
Class allows you to define “Jet” with all of necessary features. and most importantly, creating new jet

```
Jet j1;
```

creat memory on your computer with all necessary features defined, and when you create another jet Jet j2, j1 and j2 have same feature, but they are completely separated in the memory space. you are safe!

Member function

- define class vector
- public
  - access the information vec.PT() get pt of the vector
  - define vector vec.init(px,py,pz,E)
  - copy vec1=vec
  - operation vec1+ vec2
- private completely internal



Root class TLorenzVector

Public Types

enum	{ **kX** =0, **kY** =1, **kZ** =2, **kT** =3, **kNUM_COORDINATES** =4, **kSIZE** =kNUM_COORDINATES }
typedef **Double_t**	**Scalar**

▸ **Public Types inherited from TObject**

Public Member Functions

	TLorentzVector ()
	TLorentzVector (**Double_t x**, **Double_t y**, **Double_t** z, **Double_t** t)
	TLorentzVector (const **Double_t** *carray)
	TLorentzVector (const **Float_t** *carray)
	TLorentzVector (const **TVector3** &vector3, **Double_t** t)
	TLorentzVector (const **TLorentzVector** &lorentzvector)
virtual	**~TLorentzVector** ()
Double_t	**X** () const
Double_t	**Y** () const
Double_t	**Z** () const
Double_t	**T** () const
Double_t	**DeltaPhi** (const **TLorentzVector** &
Double_t	**DeltaR** (const **TLorentzVector** &) c
Double_t	**DrEtaPhi** (const **TLorentzVector** &
TVector2	**EtaPhiVector** ()
Double_t	**Angle** (const **TVector3** &**v**) const
Double_t	**Mag2** () const

more on C++

- Pythia8 and HepMC are “class”
- Pythia8 can generate events and it also contain another classes
- you can use objects in the different class by writing Phythia8::name HepMC:: name
- HepMC objects are defined in HepMC2.h. pay attention to the header file when you need the code
- namespace Pythia8; allows you to omit Pythia8::
- object with IO probably Input output tools.
- object defined with * is an address without *, it is the contents at the address . member can be accessed by “object -> member_name” for the object defined by the address, and if it is not the address access by “object.member_name”
- if you create by new, then delete to avoid memory leak.

inside include/Pythia8/Pythia.h

```
#include “Pythia8/Analysis.h" //more  Pythia8 object is defined in .h files
#include "Pythia8/Basics.h"
#include "Pythia8/BeamParticle.h"
#include "Pythia8/BeamShape.h"
#include "Pythia8/ColourReconnection.h"
#include “Pythia8/Event.h”
(途中省略
namespace Pythia8 { //start to define the namespace
class Pythia {  //definition of Pythia class
public:  can be accessed
  // Constructor. (See Pythia.cc file.)
  Pythia(string xmlDir = "../share/Pythia8/xmldoc", bool printBanner = true);

  // Destructor. (See Pythia.cc file.)
  ~Pythia();
  // Read in one update for a setting or particle data from a single line.
  bool readString(string, bool warn = true);
途中省略
  // The event record for the parton-level central process.
  Event           process;
  Event           event;
(途中省略)
private: (Pythia class internal

  // Copy and = constructors are made private so they cannot be used.
  Pythia(const Pythia&);
  Pythia& operator=(const Pythia&);

} // end namespace Pythia
```

problem on compiling check sheet

- where did you define the object. which header file is relevant?
- where is the header file
- what is liked when you “make”

```
g++ main41.cc ../lib/libpythia8.a -o main41 -I./ -I../include -O2 -ansi -pedantic -W -
Wall -Wshadow -fPIC -Wl,-rpath ../lib -ldl\
    -L./ -Wl,-rpath ./ -lHepMC
```

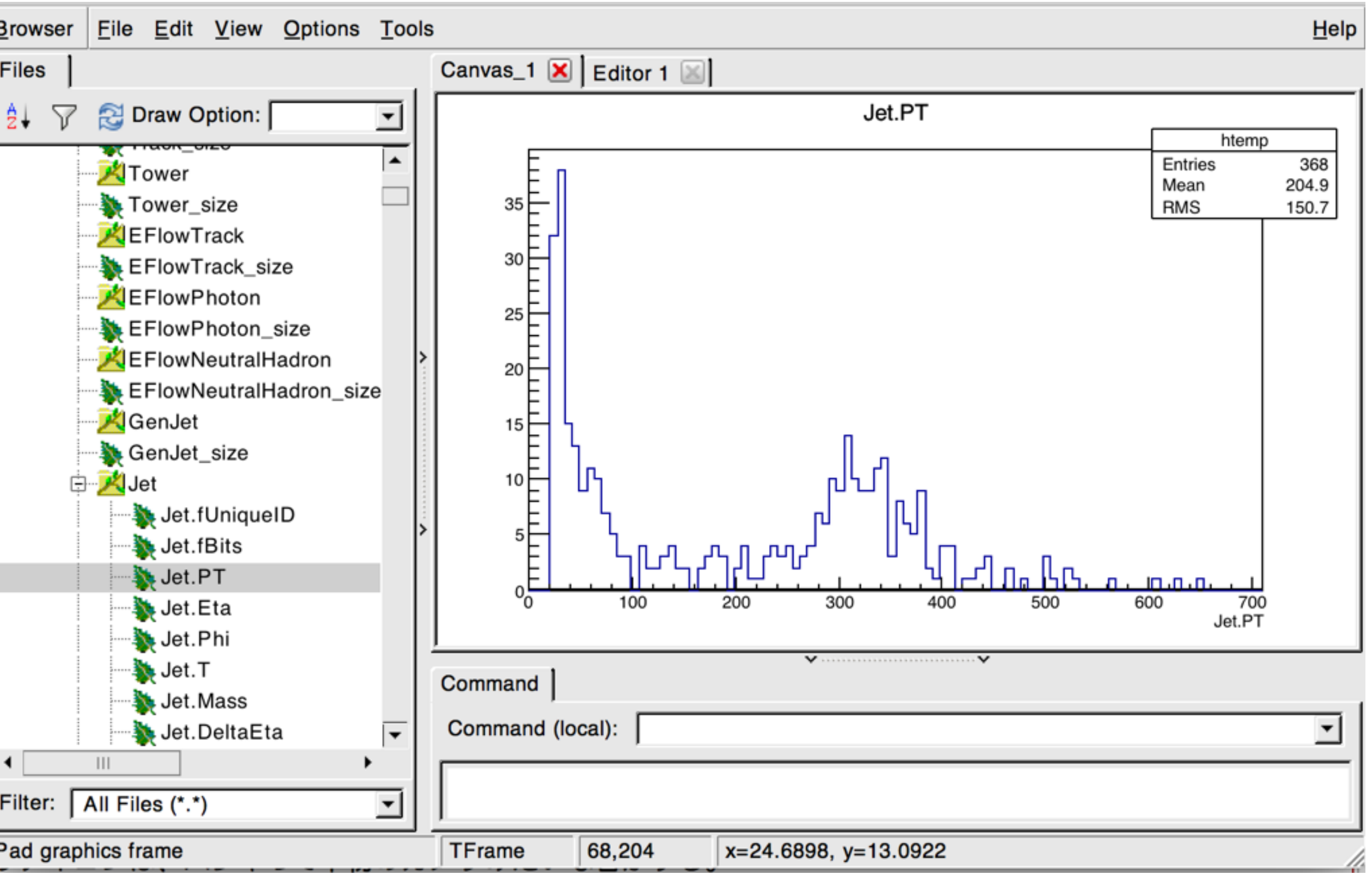
(-lHepMC pathのあるところに libHepMC.a libHepMCdylib … といったファイルがあることを仮定している。)

- if you are not sure what is happening, go to the directory with name “ include” , then grep function_name *.*

delphes3
(detector simulation)

```
$ ./DelphesHepMC
 Usage: DelphesHepMC 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.
$ ./DelphesHepMC cards/delphes_card_ATLAS.tcl test.root
hepmcout41.dat
create  test.root file

$ root  test.root
$ TBrowser b;
you will see new screen、click left tab test.root, then click
tab Delphes  which is a root “tree”
```

PT distribution of JETs of a QCD sample with PT cut > 300 GeV



Delphes tcl file read modules

```
set ExecutionPath{
  ParticlePropagator
  ChargedHadronTrackingEfficiency
  ElectronTrackingEfficiency
  MuonTrackingEfficiency
  ChargedHadronMomentumSmearing
  ElectronMomentumSmearing
  MuonMomentumSmearing
  TrackMerger
  Calorimeter
  EFlowMerger
  PhotonEfficiency
  PhotonIsolation
  ElectronFilter
  ElectronEfficiency
  ElectronIsolation
  MuonEfficiency
  MuonIsolation
  MissingET
  NeutrinoFilter
  GenJetFinder
  FastJetFinder
  JetEnergyScale
  JetFlavorAssociation
  BTagging
  TauTagging
  UniqueObjectFinder
  ScalarHT
  TreeWriter
}
```

remove non accepted particles

smearing energy and momentum

put together track and calorimeter info

photon tag と isolation check

electron efficiency isolation check

muon efficiency and isolation

invisible momentum

making jet

flavor tag

putting final objects

create root file

delphes module

modules are defined in /modules tcl read modules and define the input and output of the module.

```
nojiri$ ls *.h
AngularSmearing.h                IdentificationMap.h              PileUpMerger.h
BTagging.h                       ImpactParameterSmearing.h        PileUpMergerPythia8.h
Calorimeter.h                    Isolation.h                      Pythia8LinkDef.h
Cloner.h                         JetFakeParticle.h                RunPUPPI.h
ConstituentFilter.h              JetFlavorAssociation.h           SimpleCalorimeter.h
Delphes.h                        JetPileUpSubtractor.h            StatusPidFilter.h
Efficiency.h                     LeptonDressing.h
TaggingParticlesSkimmer.h
EnergyScale.h                    Merger.h                         TauTagging.h
EnergySmearing.h                 ModulesLinkDef.h                 TimeSmearing.h
ExampleModule.h                  MomentumSmearing.h               TrackCountingBTagging.h
FastJetFinder.h                  ParticlePropagator.h             TrackPileUpSubtractor.h
FastJetGridMedianEstimator.h     PdgCodeFilter.h                  TreeWriter.h
FastJetLinkDef.h                 PhotonConversions.h              UniqueObjectFinder.h
Hector.h                         PileUpJetID.h                    Weighter.h
```

read .tcl file （cont)

GenJetFinder has structure of FastJetFinder

for particle level jet

```
module FastJetFinder GenJetFinder {
  set InputArray NeutrinoFilter/filteredParticles

  set OutputArray jets

  # algorithm: 1 CDFJetClu, 2 MidPoint, 3 SIScone, 4 kt, 5
Cambridge/Aachen, 6 antikt
  set JetAlgorithm 6
  set ParameterR 0.5

  set JetPTMin 20.0
}
```

input output

parameter needed

FastJetFinder

For smeared jet

```
module FastJetFinder FastJetFinder {
#  set InputArray Calorimeter/towers
  set InputArray EFlowMerger/eflow

  set OutputArray jets

  # algorithm: 1 CDFJetClu, 2 MidPoint, 3 SIScone, 4 kt, 5
Cambridge/Aachen, 6 antikt
  set JetAlgorithm 6
  set ParameterR 0.5

  set JetPTMin 20.0
}
```

different input and output

tcl file defines the structure of the root file as well.

- input -> jet of FastJetFinder
- -> jet of JetEnergyScale -> add flavor info -> jet of UniqueObjectFinder
- TreeWrite define the name of branches written in Delphes Tree of the Delphes root file.

```
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
}
```

play with the root file

- **Example directory の下の　Example1.C**

```
 say $root -l examples/Example1.C’(“delphes_output.root")'
 (interpret Example1.C as C++ file . cint という)
```

```
void Example1(const char *inputFile)
{
 gSystem->Load("libDelphes");
  // Create chain of root trees

 TChain chain("Delphes");
chain.Add(inputFile);

ExRootTreeReader *treeReader = new ExRootTreeReader(&chain);
Long64_t numberOfEntries = treeReader->GetEntries();

// Get pointers to branches used in this analysis
TClonesArray *branchJet = treeReader->UseBranch("Jet");
TClonesArray *branchElectron = treeReader->UseBranch(“Electron");

// Book histograms
TH1 *histJetPT = new TH1F("jet_pt", "jet P_{T}", 100, 0.0, 100.0);
TH1 *histMass = new TH1F("mass", "M_{inv}(e_{1}, e_{2})", 100, 40.0, 140.0);
```

read libDelphes

use tree Delphes in the root file

specify the name of root file in input file

get branch to
the memory branchJet

define Histgram

```
// Loop over all events
  for(Int_t entry = 0; entry < numberOfEntries; ++entry)
  {
    // Load selected branches with data from specified event
    treeReader->ReadEntry(entry);

    // If event contains at least 1 jet
    if(branchJet->GetEntries() > 0)
    {
      // Take first jet
      Jet *jet = (Jet*) branchJet->At(0);

      // Plot jet transverse momentum
      histJetPT->Fill(jet->PT);

      // Print jet transverse momentum
      cout << "Jet pt: "<<jet->PT << endl;
}
```

loop over events

read events :

if the number of the jet is not zero

*jet is the memory where highest pT jet (at At(0)) is assigned

図を表示

```
 // Show resulting histograms
  histJetPT->Draw();
  histMass->Draw();
```

More exercise next week