コライダーの物理とツール

KEK　野尻

Plan of the lectures

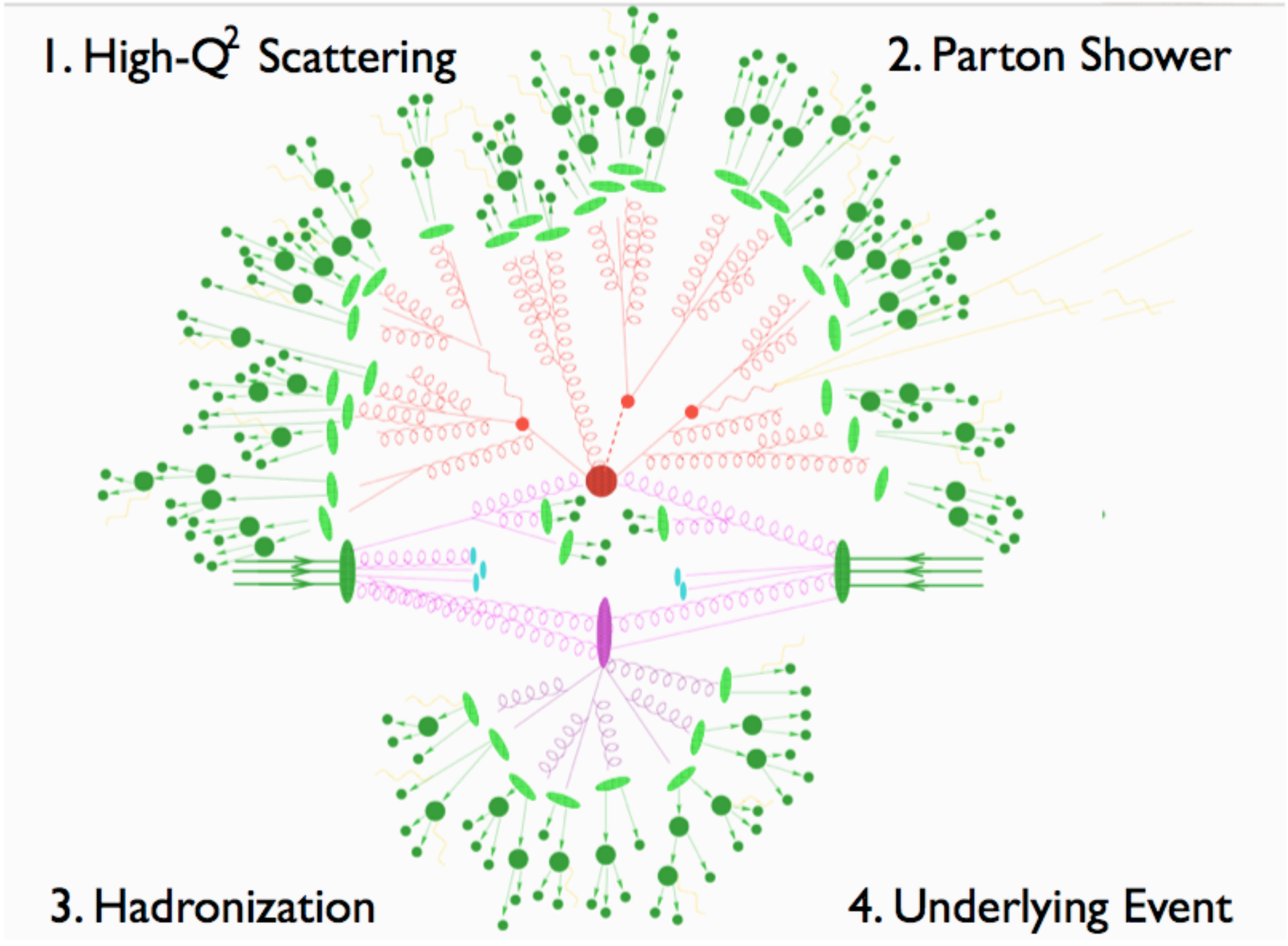
Introduction: The big picture

Infrared Behaviour of QCD

Jet Definitions

Parton Showers

Introduction: The big picture

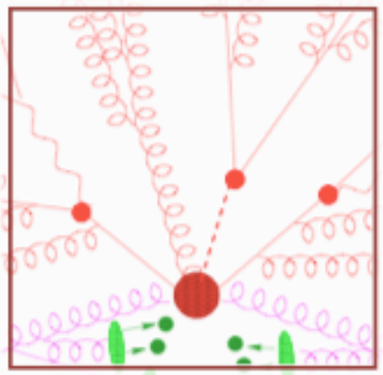


Introduction: The big picture

1. High-Q^2 Scattering

2. Parton Shower

☞ where new physics lies



Feynrules

Madgraph, Calchep

☞ process dependent

☞ first principles description

☞ it can be systematically improved

3. Hadronization

4. Underlying Event

全部やってくれる　pythia herwig++

なぜ数値計算が必要か

- hard process （解析的にできる簡単）
- parton shower + hadron 化 とても大変
- parton distribution function
- NLO …
- detector の影響

Monte Calro 積分しよう

ストレスフリーなコライダーオープンソースの使い方

- 言語の勉強しようとおもわない。ソースを読む。
- まずは example file をコピペ
- 自分で一からかこうなんて２年早い
- 人からもらおう、共同研究をしよう

ストレスフリーな数値計算

- もとのファイルは残しておく
- 一度にたくさん変更しない。一度にたくさん作らない
- **エラーメッセージは常に正しい**ので、エラーが出る原因を解消する

- コードが動いた。まだおかしい。　cout<< 変数<< endl; とすると画面にその値をかき出してくれるので、期待する量がはいっているか確認する

- 変数が定義されているか. 初期化されているか.

- 結果が30分も帰ってこない計算をするのは、論文ができる前の最後の一ヶ月

main01.cc

```
#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;
}
```

ここで必要なお道具を読み込んでる

ここからコード開始

Pythia というclass の箱(object) pythia を用意

パラメータの読み込み

pythia.readFile(fileName);

でファイルからも読み込める

初期化

イベントを作る

Pythia のメンバーと機能

readString

init　初期化 エネルギー、カット

next　次のイベントを作る

event[]　作られたものが入っている

→ size, isFinal,isCharged

stat 統計情報の打ち出し

C++お作法

iEvent 0, 1… 99まで iEvent を一つづず増やしながら計算をする

```
for (int iEvent = 0; iEvent < 100; ++iEvent) {
    if (!pythia.next()) continue;
```
次のイベントを作って、それが失敗の場合処理をスキップ

event が作れない時は結構ある

```
    // 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;
```
終状態の粒子で、かつ charged なら　nCharged を増やす

```
    mult.fill( nCharged );
  // End of event loop. Statistics. Histogram. Done.
  }
```

実行する

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 情報

```
--------  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
```

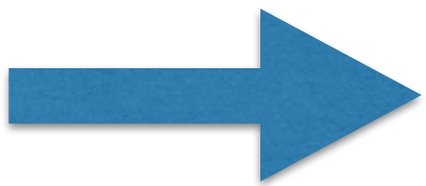
```
*-------  PYTHIA Event and Cross Section Statistics  -------------------------------------------------------------*
|                                                                                                                  |
| Subprocess                                    Code |            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
```

どんなプロセスができるのか

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[
    "HardQCD:gg2c
    "HardQCD:qq2q
    "HardQCD:qqba
    "PromptPhoton
    "PromptPhoton
    "WeakBosonExc
    "WeakSingleBo
    "WeakDoubleBo
    "WeakDoubleBo
    "WeakBosonAnd
    "WeakBosonAnd
    "WeakBosonAnd
    "Top:gg2ttbar
    "Top:ffbar2tt
    "HiggsSM:ffba
    "HiggsSM:ffbar2HW", "HiggsSM:ff2Hff(t:ZZ)", "HiggsSM:ff2Hff(t:WW)",
    "HiggsSM:qg2Hq", "HiggsSM:gg2Hg(l:t)", "HiggsSM:qg2Hq(l:t)",
    "HiggsSM:qqbar2Hg(l:t)", "HiggsBSM:allH1", "HiggsBSM:allH2",
    "HiggsBSM:allA3", "HiggsBSM:allH+-", "HiggsBSM:allHpair",
    "NewGaugeBoson:ffbar2gmZZprime", "NewGaugeBoson:ffbar2Wprime",
    "LeftRightSymmmetry:ffbar2ZR", "LeftRightSymmmetry:ffbar2WR",
```

```
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];
}
```

て、

ード

pythia.init の前にいろいろな設定が必要
(例としてW')

- エネルギー　　pythia.readString("Beams:eCM = 8000.");

- プロセス　　pythia.readString(“NewGaugeBoson:ffbar2Wprime=on”);

- 粒子の質量、崩壊モード

```
pythia.readString("34:m0 = 2000.");
pythia.readString("34:onMode = off”); すべてのモードをoff にして、
pythia.readString("34:onIfAny = 24”); pdg コードで24が入っていたら残す
```

- 反応の重心系のエネルギー、ジェットのPT (SM プロセスの場合エネルギーが低イベントはとても数が多い。

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

進んだ設定 I

- **parton distribution function に何を使うか**

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

  LHAPDF を使う場合(PDF パッケージ)

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

- 外部引数の使い方（コードを走らせるときに同時に数字を指定）

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

char は文字

例:何もしないと毎回同じイベントが生成されるので、たくさんイベントが必要な場合は乱数を変える必要がある。例えば毎回乱数の SEED を変えて走らせる場合

```
string iseed= argv[1];
string  dummy="Random:seed = "+ iseed;
//cout<< dummy <<endl;
pythia.readString(dummy);
```

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

演習（休憩中に）

- 欲しいプロセス(new physicsでよい)を決めて、mass が増えると、どの程度 cross section が減るかみる

- 見えない運動量のあるプロセスを選んで、重心系エネルギー を増やすと、どの程度cross section が減るか見る。

- 次回報告 (4~5人）

進んだ設定 イベントの保存／圧縮

- イベントは何度も呼び出して使いたい。
- 後で別の研究にも使いまわしたい。
- いろんなコードに読み込んで、使いたい
- event 生成-> lhe, hepmc -> (測定器シミュレーション）-> root ファイル→再解析
- 歴史　Les Houches workshop 昔はみんな勝手なフォーマットでやっていたが、共通のスタンダートを作って、互いにもっていけるようにした。

pythia8 からhepmc　main41.cc

機能を導入

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

コードの中で

```
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.
  }
```

pythia のevent を HepMC に

書き出し

generator のイベントをいれる箱

pythia のイベントコピー

書き込み

後始末

問題

- 自分の作りたいイベントをhepmc file にかきなさい。

- hepmc ファイルを眺めて何がかいてあるか想像しなさい。

C++お作法

クラスとは何か。

例えば我々が物理的実態としてジェットを表すときに

運動量 <-ローレンツ変換性　演算(重さ、pT, eta, phi)

中にある粒子数

中にある粒子の分布(個々の運動量)

電荷を持つ粒子数…

などいろいろな量と、その変換性、ジェット相互の演算などが一体となっている。

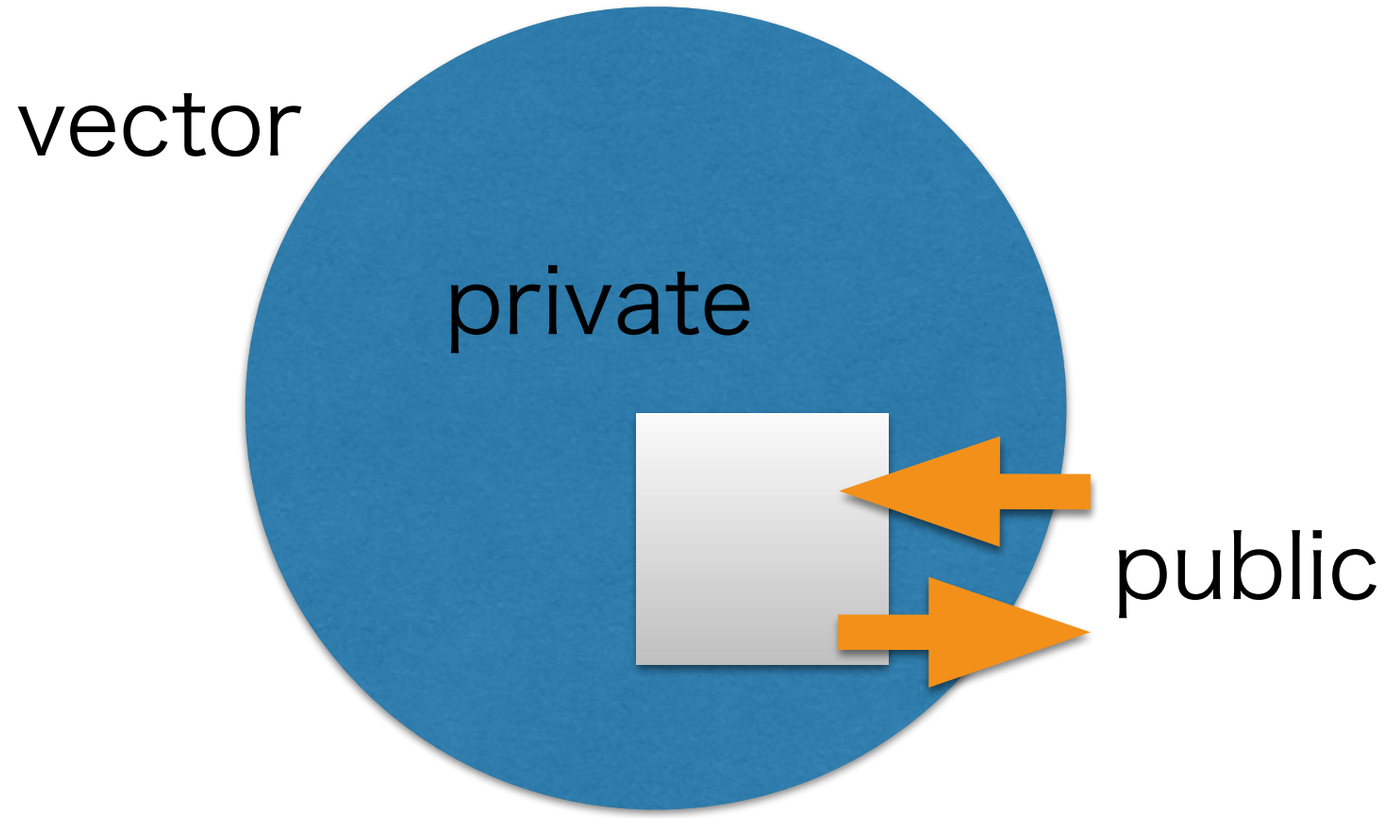
ジェットのクラス　Jet(Delpehs で定義) はこれらが一つのまとまりとして定義されている。そしてジェットを作る

```
Jet j1;
```

という操作は、メモリーの上に「Jet の性質が定義されて、かつ初期化されている場所」を確保する。さらに　Jet j2; と定義すると j1 とj2 は同じ性質をもつ物理量を持っているが、メモリー上で混ざらない（安全）

メンバー関数

- 例えば vector というクラスがあったとする
- public な機能
  - 取り出し
  - vec.PT()　vector の持ってる運動量にアクセスする。
  - 定義
  - vec.init(px,py,pz,E)
  - コピー vec1=vec
  - 演算　vec1+ vec2
- private は外に出さない



Root のクラス　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

C++のお作法

- Pythia8 と HepMC 別のクラス
- Pythia8 クラスは、イベントの生成自体を扱える機能を持ったクラスで、さらにいくつものクラスから構築されている。（次のスライド）
- 名前空間があるので、　Phythia8:: 名前 HepMC:: 名前と区別できる。
- HepMC:: の中のものは HepMC2.h の中に定義されてるはず。(機能は header なしにこない)
- namespace Pythia8;　とやると Pythia8:: は省略できる　(名前ダブってると怒られる, と思う）HepMC2 のものを使いたい時は明示的に指定する。
- IO 多分 Input output のツール
- * つきはアドレス　 *なしは実体.　* つきで作ったものの メンバー は　名前->obj で
- new で作ったものは用がすんだら delete すること。

include/Pythia8/Pythia.hの中身

```
#include “Pythia8/Analysis.h" このなかにも お名前空間の追加がある。
#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 { お名前空間の定義が始まる
class Pythia {  Pythia クラスの定義
public:  外から使えるもの
  // 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 の中でだけ情報のやりとりに使う

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

} // end namespace Pythia
```

定義されてないとかライブラリが同じgcc にないとか怒られたら

- header file がコードの最初にあるか
- header file はどこに置かれているか
- make するとき何がlink されているか

```
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。)

- どこにあるかわからなくなったら　include directory のしたにいって　grep “機能名” * *.* などとすればいい。

delphes3
(測定器シミュレーション)

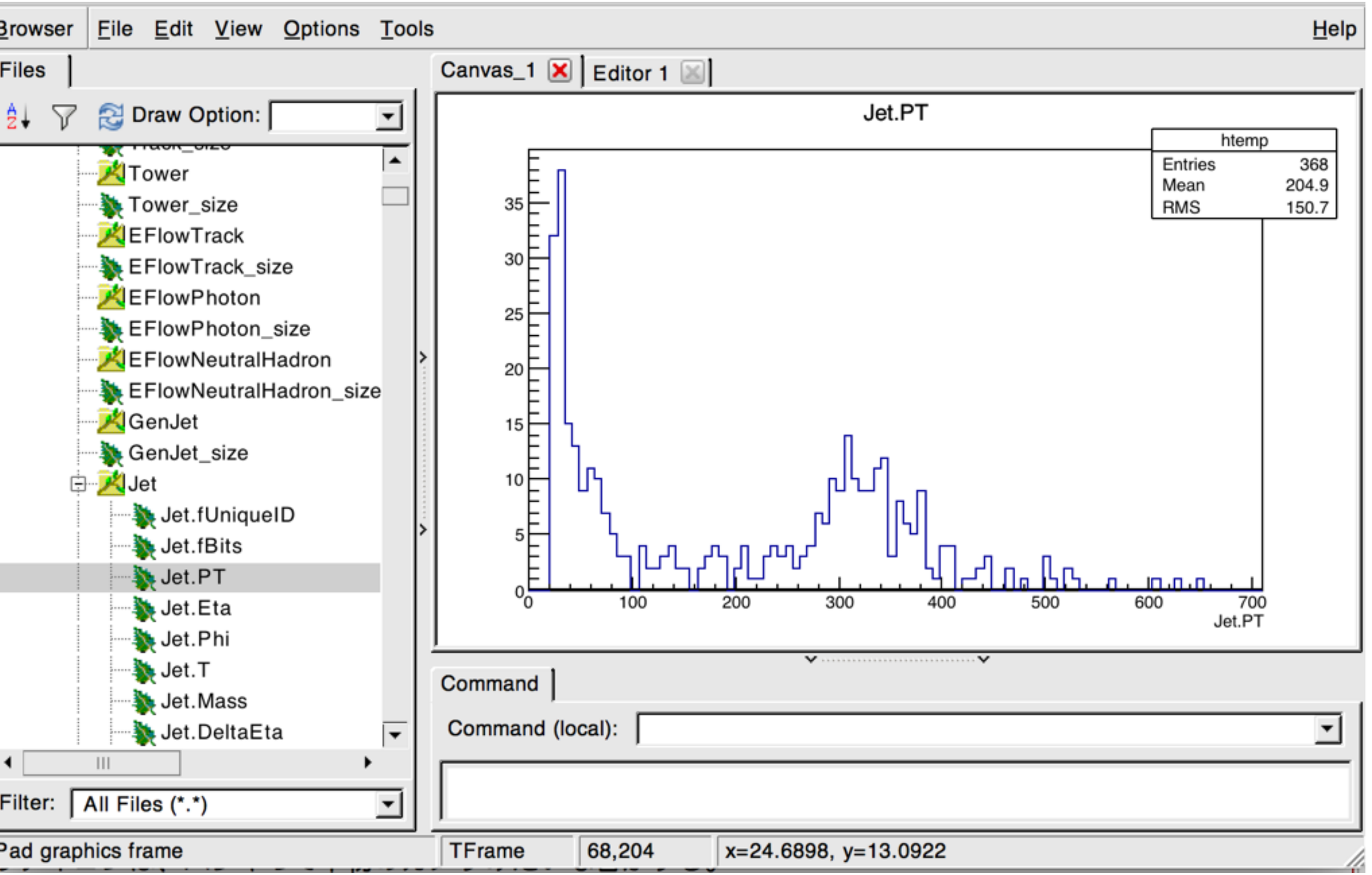
```
$ ./DelphesHepMC
 Usage: DelphesHepMC config_file output_file [input_file(s)]
 config_file - configuration file in Tcl format,
 output_file - output file in ROOT format,
 input_file(s) - input file(s) in HepMC format,
 with no input_file, or when input_file is -, read standard
input.
$ ./DelphesHepMC cards/delphes_card_ATLAS.tcl test.root
hepmcout41.dat
```

とやると test.rootというファイルができる。

```
$ root  test.root
$ TBrowser b;
```

とやると画面が立ち上がるので、左の test.root -> Delphes というタブをクリック

PTcut を300GeV にしたQCD



ここで自分が何をしたか考える　tcl ファイルの中身

```
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
}
```

detector に確率的にかからない分を
乱数で落とす

エネルギー運動量のsmearing を追加

トラック、カロリメータ情報まとめ

photon tag と isolation check

電子効率と isolation check

muon 効率とisolation

見えない運動量

ジェットこねこね

flavor tag

整理整頓

root file に書き出し

delphes module

/modules　にあるツールをカードに書いてある順番に　input とoutput を指定して繋いでいく構造

```
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
```

.tcl file を読む（続き）

GenJetFinder は FastJetFinder の構造をもつ

```
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 の選択
output の選択

基本的な設定をかく

FastJetFinder もある。

```
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
}
```

別のinput の選択
output の選択

root への書き出しを決める／jet の定義　+　書き出し

- input から　FastJetFinder の jet
- 　-> JetEnergyScale のjet -> flavor の情報の追加-> UniqueObjectFinderのJet になる.
- root へのoutput　ここに root file に書き出すものの名前が設定されている
- 書き出し

```
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
}
```

できた root file で遊ぶ

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

動かし方 $root -l examples/Example1.C’(“delphes_output.root")'

(こうすると あたかもC++ のようにコマンドを解釈して計算をしてくれる. 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);
```

ライブラリ読み込み

root file の中にある Delphes の tree　を利用

使うファイルを指定 (ファイルをいくつも付け足せる)

branch の割り付け

Histgram の定義

```
// 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;
}
```

event でloop する

イベント読み込み: ここで branchJet 等の中身が書きかわる

jet の数が 0 でないとして

highest pT jet　をとる

図を表示

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

もう少しましなexample をメールで回します。