Trying It Out

- Download `pythia8100.tgz` from `http://www.thep.lu.se/∼torbjorn/Pythia.html`
- `tar xvfz pythia8100.tgz` to unzip and expand
- `cd pythia8100` to move to new directory
- `./configure ...` needed for external libraries + debug/shared (see `README`, libraries: HepMC, LHAPDF, PYTHIA 6)
- `make` will compile in ∼ 3 minutes (for archive library, same amount extra for shared)
- The `htmldoc/pythia8100.pdf` file contains A Brief Introduction
- Open `htmldoc/Welcome.html` in a web browser for the full manual
- Install the `phpdoc/` directory on a webserver and open `phpdoc/Welcome.html` in a web browser for an interactive manual
- The `examples` subdirectory contains 30 sample main programs: standalone, link to libraries, semi-internal processes, . . . (`make mainNN` and then `./mainNN.exe > outfile`)
- A `Worksheet` (on the web pages) contains step-by-step instructions and exercises how to write and run a main program

More on settings

Settings are stored in four separate maps (flags/modes/parms/words).
For each setting, need to store

- name: of form `task:property`, e.g. `TimeShower:pTmin`
- default value
- current value
- allowed range: minimum/maximum on/off (not for flags).

Useful commands:

- `pythia.settings.listAll()` : complete list
- `pythia.settings.listChanged()` : only changed ones

```
*-------  PYTHIA Flag + Mode + Parm + Word Settings (changes only)  --------------------------*
|                                                                                             |
| Name                                |            Now |      Default         Min         Max |
|                                     |                |                                      |
| HardQCD:all                         |             on |          off                         |
| Main:eCM                            |      14000.000 |     1960.000    10.00000             |
| Main:numberToList                   |              1 |            2           0             |
| Main:showChangedParticleData        |             on |          off                         |
| Main:timesToShow                    |             20 |           50           0             |
| MultipleInteractions:pTmin          |        3.00000 |      0.20000     0.10000    10.00000 |
| PhaseSpace:pTHatMin                 |       50.00000 |          0.0         0.0             |
| PromptPhoton:all                    |             on |          off                         |
| SpaceShower:pT0Ref                  |        2.00000 |      2.20000     0.50000    10.00000 |
|                                                                                             |
*-------  End PYTHIA Flag + Mode + Parm + Word Settings  -------------------------------------*
```

Hard-process generation

Processes can be switched on with

```
ProcessGroup:ProcessName = on
```

or sometimes

```
ProcessGroup:all = on
```

ProcessGroup	ProcessName
SoftQCD	minBias,elastic, singleDiffractive, doubleDiffractive
HardQCD	gg2gg, gg2qqbar, qg2qg, qq2qq, qqbar2gg, qqbar2qqbarNew, gg2ccbar, qqbar2ccbar, gg2bbbar, qqbar2bbbar
PromptPhoton	qg2qgamma, qqbar2ggamma, gg2ggamma, ffbar2gammagamma, gg2gammagamma
WeakBosonExchange	ff2ff(t:gmZ), ff2ff(t:W)
WeakSingleBoson	ffbar2gmZ, ffbar2W, ffbar2ffbar(s:gm)
WeakDoubleBoson	ffbar2gmZgmZ, ffbar2ZW, ffbar2WW
WeakBosonAndParton	qqbar2gmZg, qg2gmZq, ffbar2gmZgm, fgm2gmZf qqbar2Wg, qg2Wq, ffbar2Wgm, fgm2Wf
Charmonium	gg2QQbar[3S1(1)]g, qg2QQbar[3PJ(8)]q, ...
Bottomonium	gg2QQbar[3S1(1)]g, gg2QQbar[3P2(1)]g, ...

ProcessGroup	ProcessName
Top	gg2ttbar, qqbar2ttbar, qq2tq(t:W), ffbar2ttbar(s:gmZ), ffbar2tqbar(s:W)
FourthBottom	gg2bPrimebPrimebar, qq2bPrimeq(t:W) , ...
FourthTop	qqbar2tPrimetPrimebar, fbar2tPrimeqbar(s:W), ...
FourthPair	ffbar2tPrimebPrimebar(s:W), fbar2tauPrimenuPrimebar(s:W)
HiggsSM	ffbar2H, gg2H, ffbar2HZ, ff2Hff(t:WW), ...
HiggsBSM	h, H and A as above, charged Higgs, pairs
SUSY	qqbar2chi0chi0 (SUSY barely begun)
NewGaugeBoson	ffbar2gmZZprime, ffbar2Wprime, ffbar2R0
LeftRightSymmmetry	ffbar2ZR, ffbar2WR, ffbar2HLHL, ...
LeptoQuark	ql2LQ, qg2LQl, gg2LQLQbar, qqbar2LQLQbar
ExcitedFermion	dg2dStar, qq2uStarq, qqbar2muStarmu, ...
ExtraDimensionsG*	gg2G*, qqbar2G*, ...

Can also use (and sometimes mix with)

- Les Houches Event Files
- Les Houches Accord-style runtime C++ interface
- Les Houches Accord runtime Fortran 77 interface (and that way runtime link to PYTHIA 6.4)
- semi-internal matrix elements and resonances (external matrix elements, internal phase space)

More on particle data

The static `ParticleDataTable` class contains info by PDG `id` code:

- `name(id), hasAnti(id)`
- `spinType(id), chargeType(id), charge(id), colType(id)`
- `m0(id), mWidth(id), mMin(id), mMax(id), tau0(id), ...`

plus a vector of `DecayChannel`s with

- `onMode(), bRatio(), meMode(), multiplicity(), product(i)`

User modifies by methods, `readString("...")` and `readFile("filename")` with commands `id:property = value` or `id:channel:property = value`.

Some special commands:

```
id:all = name antiName spinType chargeType colType m0 mWidth mMin mMax tau0
id:new = name antiName spinType chargeType colType m0 mWidth mMin mMax tau0
id:channel:all = onMode bRatio meMode products
id:oneChannel = onMode bRatio meMode products
id:addChannel = onMode bRatio meMode products
id:onMode = onMode
id:onIfAny = products    and   id:offIfAny = products
id:onIfAll = products    and   id:offIfAll = products
id:onIfMatch = products  and   id:offIfMatch = products
```

Useful commands:

- `pythia.particleData.listAll()` : complete list
- `pythia.particleData.listChanged()` : only changed ones
- `pythia.particleData.list(id)` : only one (or `vector<int>`)

```
 --------  PYTHIA Particle Data Table (changed only)  --------------------------
----------------------------------------------------

      id   name            antiName         spn chg col      m0        mWidth
   mMin       mMax       tau0    res dec ext vis wid
             no onMode    bRatio   meMode     products

     111  pi0                                 1   0   0    0.13498    0.00000
 0.00000    0.00000  2.51000e-05   0   0   0   1   0
              0     1   0.9879900    0       22       22
              1     1   0.0119800   11       22       11      -11
              2     1   0.0000300   13       11      -11       11      -11

     223  omega                               3   0   0    0.78259    0.00849
 0.10000    0.00000  0.00000e+00   0   1   0   1   0
              0     1   0.8924000    1      211     -211      111
              1     1   0.0892800    0       22      111
              2     1   0.0170000    3      211     -211
              3     1   0.0004900    0      221       22
              4     1   0.0000700    0      111      111       22
              5     1   0.0005900    0      111       11      -11
              6     1   0.0001000    0      111       13      -13
              7     1   0.0000700    0       11      -11

 --------  End PYTHIA Particle Data Table  -------------------------------------
----------------------------------------------------
```

Other event information

You can use `pythia.info.method()` to extract one-of-a-kind information, such as:

- `idA(), idB(), eCM()` : incoming beams and cm energy.
- `name(), code()` : the name and code of the subprocess.
- `id1(), id2(), x1(), x2()` : the identities and x fractions of the two partons coming in to the hard subprocess.
- `pdf1(), pdf2(), Q2Fac()` : parton densities $x\,f(x, Q^2)$ evaluated for the two incoming partons, and the associated Q^2 scale.
- `alphaS(), alphaEM(), Q2Ren()` : α_s, α_{em} and their Q^2 scale.
- `mHat(), sHat(), tHat(), uHat()` : the invariant mass of the hard subprocess and the Mandelstam variables.
- `pTHat(), thetaHat(), phiHat()` : transverse momentum and polar and azimuthal scattering angles of the hard subprocess.
- `bMI(), nMI()` : impact parameter (rescaled) and number of multiple interactions.
- `list()` : list some information on output.
- `sigmaGen(), sigmaErr()` : the process-summed estimated cross section and its estimated statistical error, in mb.

Event analysis

Four-vectors in a class `Vec4`, with overloaded operators.

A small package for one-dimensional histograms:

- Book with `Hist name( title, numberOfBins, xMin, xMax);`
  or `Hist name; name.book(title, numberOfBins, xMin, xMax);`
- Fill with `name.fill( xValue, weight);` with default `weight = 1`
- Print with `cout << name;`
- Overloaded operators for addition, multiplication, . . .

Sphericity analysis (similarly thrust):

- Instantiate with `Sphericity sph( power, select);`
- Analyze with `sph.analyze( event);`
- Info with `sph.sph(), sph.EigenVector(i), sph.list(), ...`

Cone jet finder a la UA1 (PYCELL) (similarly Lund/JADE/Durham):

- Instantiate with `CellJet cellJet( etaMax, nEta, nPhi, select, smear, resolution, upperCut, threshold);`
- Analyze with `cellJet.analyze(event, eTjetMin, coneRadius, eTseed);`
- Info with `cellJet.size(), cellJet.eT(i), cellJet.list(), ...`

Link to other program

PYTHIA is standalone, but several possibilities to link to it.

Posibilities similar to PYTHIA 6.4:

- Input from Les Houches Accord & Les Houches Event Files
- Output to HepMC event format (more robust than PYTHIA 6!?)
- SUSY Les Houches Accord (input file with masses, couplings, . . .)
- Link to external decays, e.g. for τ and B.
- Link to LHAPDF version 5.3.0 or later, or to your own PDF.

New possibilities, based on derived classes and pointers to them:

- Semi-internal process: write derived matrix-element class,
  ```
  SigmaProcess* mySigma = new MySigma();
  pythia.setSigmaPtr( mySigma);
  ```
  and let PYTHIA do phase space integration, process mixing, . . .
- Semi-internal resonance in same style: calculate partial widths
- Link to external random-number generator.
- Link to external shower, e.g. VINCIA for FSR.
- User hooks: veto events early on or reweight cross section.

Sample Main Programs

- `main01.cc`: charged multiplicity distribution
- `main02.cc`: Z^0 $p_\perp$ spectrum
- `main03.cc & main03.cmnd`: single-particle analysis in jet events
- `main04.cc & main04.cmnd`: tests of event properties
- `main05.cc`: cone-jet analysis of LHC events
- `main06.cc & main06.cmnd`: study elastic/diffractive events
- `main07.cc & main07.cmnd`: study minimum-bias events
- `main08.cc & main08.cmnd`: combine results of subruns in $p_\perp$ bins
- `main09.cc`: LEP events with sphericity/thrust/jetfinder analysis
- `main10.cc`: use UserHooks to interact with generation process
- `main11.cc`: set two hard interactions in the same event
- `main12.cc & ttbar.lhe`: input from a Les Houches Event File
- `main13.cc & ttbar.lhe & ttbar2.lhe`: input from two Les Houches Event Files; mix with internal processes
- `main14.cc`: **compare** several cross sections with PYTHIA 6.4 values
- `main15.cc`: redo B decays several times for each event

- `main16.cc`: user analysis class; command-line input file
- `main17.cc`: Pythia wrapper class; command-line input file
- `main21.cc`: input of parton configurations for hadronization only
- `main22.cc & main22.cmnd & main22.spc`: SUSY with SLHA input
- `main23.cc`: link an external decay handler
- `main24.cc`: link an external random number generator
- `main25.cc`: link an external process for internal use
- `main26.cc`: link an external resonance and process for internal use
- `main31.cc & main31.cmnd`: simple output to HepMC event file
- `main32.cc & main32.cmnd`: streamlined production to HepMC; command-line input and output files
- `main41.cc`: test shapes of PDF's in LHAPDF
- `main42.cc`: compare event properties for different LHAPDF PDF's
- `main51.cc`: runtime LHA link to PYTHIA 6.4
- `main52.cc & main52.ccmnd & main52.fcmnd`: ditto with input files
- `main53.f`: (Fortran!) have PYTHIA 6.4 generate an LHEF
- `main54.cc & main54.cmnd`: input from PYTHIA 6.4 and output to HepMC