(1) New PDG Particle Codes
(2) PYTHIA 6.3 Parton Showering*
(3) PYTHIA 8 Progress Report

Torbjörn Sjöstrand
CERN/PH and
Department of Theoretical Physics, Lund University

* reporting for P. Skands, T. Plehn, D. Rainwater
New PDG codes for Nuclei

\[ \text{id} = 10LZZZAAAI \]

For a (hyper)nucleus consisting of \( n_p \) protons, \( n_n \) neutrons and \( n_{\Lambda} \Lambda^0 \)'s:

- \( A = n_p + n_n + n_{\Lambda} \) gives the total baryon number,
- \( Z = n_p \) the total charge, and
- \( L = n_{\Lambda} \) the total number of strange quarks.

\( I \) gives the isomer level, with \( I = 0 \) corresponding to the ground state
  and \( I > 0 \) to excitations.

Examples:

- deuteron \[ 100010020 \]
- \( ^{\alpha} \) \[ 100020040 \]
- \( ^{235}\text{U} \) \[ 100922350 \]

Warning: single hadrons, like \( p, n \) or \( \Lambda^0 \), are not changed.

(has been discussed & circulated, almost “cast in stone”)
New PDG codes for $R$-hadrons

Prompted by split-SUSY interest, but intended more generically for long-lived colour triplets and octets (leptoquarks, extra dimensions, . . .) which hadronize to give $\bar{g}g$, $\bar{g}q\bar{q}$, $\bar{g}qq\bar{q}$, $\bar{q}q$, $\bar{q}qq$

Main principles:
- Put in the 1,000,000 and 2,000,000 normal SUSY series
- Enumerate the flavour content about as for normal mesons/baryons
- Let the squark/gluino flavour be the first one given
- For squark-mesons, use sign $+$ for squarks and $-$ for antisquarks
- Represent gluinos by a 9, like for gluons in glueballs
- The $2s + 1$ digit is based only on spin of the light degrees of freedom (since the heavy spin decouples for $M \to \infty$)

Examples:

<table>
<thead>
<tr>
<th>gluino-hadrons</th>
<th>squark-hadrons</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\bar{g}g$</td>
<td>$\bar{u}_L\bar{s}$</td>
</tr>
<tr>
<td>$\bar{g}ud$</td>
<td>$\bar{u}_R\bar{s}$</td>
</tr>
<tr>
<td>$\bar{g}ud$</td>
<td>$\bar{u}_R^*\bar{s}$</td>
</tr>
<tr>
<td>$\bar{g}uud$</td>
<td>$\bar{c}_Luu$</td>
</tr>
</tbody>
</table>

(informal agreement)
PYTHIA 6.3 Showering in Transverse Momentum

1) Define
\[ p_{\perp \text{evol}}^2 = z(1 - z)Q^2 = z(1 - z)M^2 \] for FSR
\[ p_{\perp \text{evol}}^2 = (1 - z)Q^2 = (1 - z)(-M^2) \] for ISR

2) Evolve all partons downwards in \( p_{\perp \text{evol}} \) from common \( p_{\perp \text{max}} \)
\[ d\mathcal{P}_a = \frac{dp_{\perp \text{evol}}^2}{p_{\perp \text{evol}}^2} \frac{\alpha_S(p_{\perp \text{evol}}^2)}{2\pi} P_{a\rightarrow bc}(z) \, dz \, \exp \left( -\int_{p_{\perp \text{evol}}}^{p_{\perp \text{max}}} \cdots \right) \]
\[ d\mathcal{P}_b = \frac{dp_{\perp \text{evol}}^2}{p_{\perp \text{evol}}^2} \frac{\alpha_S(p_{\perp \text{evol}}^2)}{2\pi} \frac{x'f_a(x', p_{\perp \text{evol}}^2)}{xf_b(x, p_{\perp \text{evol}}^2)} P_{a\rightarrow bc}(z) \, dz \, \exp \left( -\cdots \right) \]

Pick the one with largest \( p_{\perp \text{evol}} \) to undergo branching; also gives \( z \).

3) Kinematics: Derive \( Q^2 = \pm M^2 \) by inversion of 1), but then interpret \( z \) as energy fraction (not lightcone) in “dipole” rest frame, so that Lorentz invariant and matched to matrix elements.
Assume yet unbranched partons on-shell and shuffle \((E, p)\) inside dipole.

4) Iterate \( \Rightarrow \) combined sequence \( p_{\perp \text{max}} > p_{\perp 1} > p_{\perp 2} > \cdots > p_{\perp \text{min}} \).
One Objective: Interleaved Multiple Interactions

\[ p_{\perp} \]

\[ p_{\perp\text{max}} \]
\[ p_{\perp1} \]
\[ p'_{\perp1} \]
\[ p_{\perp2} \]
\[ p_{\perp3} \]
\[ p_{\perp23} \]
\[ p_{\perp4} \]
\[ p_{\perp\text{min}} \]

hard int.
mult. int.
ISR
ISR
ISR
ISR
ISR
ISR
ISR
mult int.

interaction number

1 2 3 4
Matrix Elements and Parton Showers

Complementary strengths:
- ME’s good for well separated jets
- PS’s good for structure inside jets

Marriage desirable! But how? Many problems!

Much work ongoing \(\iff\) no established orthodoxy

Three main areas, in ascending order of complication:
  1) Match to lowest-order nontrivial process — merging
  2) Combine leading-order multiparton process — vetoed parton showers
  3) Match to next-to-leading order process — MC@NLO

…but let’s not forget the “original” approach:

0) Improve the shower algorithm itself
(if doable then it gives fast results for “all” processes)
Shower Issues

1) Is $p_\perp^2$ a better evolution variable than $Q^2 = |M^2|$?
   Sudakovs different even if phase space the same;
   evolution in $p_\perp$ favours larger $p_\perp$’s.

2) What is appropriate maximum scale of evolution?
   Conventional wisdom: evolve below “characteristic scale” of process,
   $\sim$ as for PDF scale choice, $\rightarrow f(\hat{s}, \hat{t}, m_i^2)$,
   e.g. $Q_{\text{max}}^2 \approx \hat{s} \approx m_Z^2$ for $s$-channel process like $Z^0$ production,
   or $Q_{\text{max}}^2 \approx m_\perp^2 = m_t^2 + p_\perp^2$ for $t\bar{t}$ production.
   But $Z^0$ experience: $Q_{\text{max}}^2 = s$ surprisingly good,
   i.e. let shower populate whole phase space.
Test of $Z^0$ production at the Tevatron

max scale = $s$

Tune A = old
$Q^2 = |M^2|$ ordering

The others = new
$p_\perp^2$-ordering,
various variants of
MI/ISR/FSR matching
(see TS & P. Skands,
EPJ C39 (2005) 129)

Conclusion:
$p_\perp$ gives improvement,
but details matter
Shower Issues

1) Is $p_\perp^2$ a better evolution variable than $Q^2 = |M^2|$?
   Sudakovs different even if phase space the same;
   evolution in $p_\perp$ favours larger $p_\perp$’s.

2) What is appropriate maximum scale of evolution?
   Conventional wisdom: evolve below “characteristic scale” of process,
   \[ \sim \text{as for PDF scale choice, } \rightarrow f(\hat{s}, \hat{t}, m_i^2), \]
   e.g. $Q_{\text{max}}^2 \approx \hat{s} \approx m_Z^2$ for $s$-channel process like $Z^0$ production,
   or $Q_{\text{max}}^2 \approx m_\perp^2 = m_t^2 + p_\perp^2$ for $t\bar{t}$ production.
   But $Z^0$ experience: $Q_{\text{max}}^2 = s$ surprisingly good,
   i.e. let shower populate whole phase space.

So study $t\bar{t}$ production and compare
- PYTHIA old $Q^2$-ordered, $Q_{\text{max}}^2 = m_\perp^2$
  (PYTHIA default is $Q_{\text{max}}^2 = 4m_\perp^2$)
- PYTHIA old $Q^2$-ordered, $Q_{\text{max}}^2 = s$
- PYTHIA new $p_\perp^2$-ordered, $Q_{\text{max}}^2 = m_\perp^2$
- PYTHIA new $p_\perp^2$-ordered, $Q_{\text{max}}^2 = s$
Current test: $t\bar{t}$ production at the Tevatron

Plots and shower studies by P. Skands
MadGraph ME calculations by T. Plehn & D. Rainwater
(publication in preparation)

Parton-level only, no underlying event, no top decays

$E_{\perp \text{jet}} > 50$ GeV, $\Delta R = 0.4$
Tevatron: ttbar + 2 jets
CTEQ5L, no K-factors

Max Jet $p_T$ ($\text{GeV}$)

- MadGraph: ttbar + 2 jets
- Pythia 6.3: $p_T^2$ (power)
- Pythia 6.3: $p_T^2$ (wimpy)
- Pythia 6.2: $Q^2$ (power)
- Pythia 6.2: $Q^2$ (wimpy)

Min Jet $p_T$ ($\text{GeV}$)

- MadGraph: ttbar + 2 jets
- Pythia 6.3: $p_T^2$ (power)
- Pythia 6.3: $p_T^2$ (wimpy)
- Pythia 6.2: $Q^2$ (power)
- Pythia 6.2: $Q^2$ (wimpy)
Conclusions:

**Transverse-momentum-ordered showers with maximal starting scale does amazingly well!**

Bodes well for “blind” application of the new showers, either
- to processes not yet studied with more “sophisticated” methods
- to **further** emissions when hardest given by matrix elements
On To C++

Currently HERWIG and PYTHIA are successfully being used, also in new LHC environments, using C++ wrappers

A1: Need to clean up!
Q: Why rewrite?  A2: Fortran 77 is limiting Fortran 90
A3: Young experimentalists will expect C++

PYTHIA7 project \(\Longrightarrow\) ThePEG
Toolkit for High Energy Physics Event Generation
(L. Lönnblad; S. Gieseke, A. Ribon, P. Richardson)

HERWIG++: complete reimplementation
(B.R. Webber; S. Gieseke, A. Ribon, P. Richardson, M. Seymour, P. Stephens, 3 new)

ARIADNE/LDC: to do ISR/FSR showers, multiple interactions
(L. Lönnblad; N. Lavesson)

SHERPA: in C++ from start, partly wrappers to PYTHIA Fortran
(F. Krauss; T. Gleisberg, S. Hoeche, A. Schaelicke, S. Schumann, J. Winter)
PYTHIA8: A fresh start

Problem: PYTHIA7 stalled, no other manpower
Solution?: take a sabbatical and work “full-time”!
(⇒ baseline model, S. Mrenna & P. Skands join later ?)

Tentative schedule:

<table>
<thead>
<tr>
<th>time</th>
<th>date</th>
<th>processes</th>
<th>final states</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1 Sept. 2004</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>1</td>
<td>1 Sept. 2005</td>
<td>LHA-style input</td>
<td>incomplete draft</td>
</tr>
<tr>
<td>2</td>
<td>1 Sept. 2006</td>
<td>a few processes</td>
<td>complete, buggy(?)</td>
</tr>
<tr>
<td>3</td>
<td>1 Sept. 2007</td>
<td>more processes</td>
<td>stable, debugged</td>
</tr>
</tbody>
</table>

...but don’t forget Murphy’s law

Objectives:

- clean up, keep the most recent models
- Les Houches Accord style input central
- independent of ThePEG (or anything else), but
- interface to ThePEG later written by Leif (?)
Current PYTHIA8 structure

The User (≈ Main Program)

Pythia

Event process

Event event

ProcessLevel
- LHAinit
- LHAevnt (PYTHIA 6.3)
  (...??)

PartonLevel
- TimeShower
- SpaceShower
- MultipleInteractions
- BeamRemnants

HadronLevel
- StringFragmentation
- MiniStringFrag...
- ParticleDecays
  (...??)

BeamParticle

Vec4, Random, Settings, ParticleData, StandardModel, ...
## Current PYTHIA8 status

### Existing classes

<table>
<thead>
<tr>
<th>Process</th>
<th>Level</th>
<th>Class</th>
<th>Level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>LHAinit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Level</td>
<td>LHAevnt</td>
<td>**</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(PYTHIA 6.3)</td>
<td>***</td>
</tr>
<tr>
<td></td>
<td>Parton</td>
<td>TimeShower</td>
<td>**</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SpaceShower</td>
<td>**</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MultipleInteractions</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BeamRemnants</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>Hadron</td>
<td>StringFragmentation</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MiniStringFrag...</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ParticleDecays</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>Event</td>
<td>**</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>BeamParticle</td>
<td>**</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Vec4, Random</td>
<td>***</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Settings</td>
<td>**</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ParticleData</td>
<td>*</td>
</tr>
</tbody>
</table>

### Missing classes/topics

- ThePEG input, alternatively
- Cross section administration
- Phase space selection
- Process matrix elements
- Parton density libraries
- Resonance decays
- MI/ISR/FSR interleaving
- colour flow models
- ME/PS matching
- Junction fragmentation
- Popcorn baryons
- updated decay tables
- Bose-Einstein
- event analysis routines

...and much, much more

⇒⇒ Roughly according to three-year plan so far!

First public “proof-of-concept” version by GENSER July meeting (!?)
Event generation structure

1) Initialization step
   • select process(es) to study
   • modify physics parameters
   • set kinematics constraints
   • modify generator settings
   • initialize generator
   • book histograms

```
#include "Pythia.h"
using namespace Pythia8;
Pythia pythia;
pythia.readLine("command");
pythia.readFile("command.file");
pythia.init(idBeamA,idBeamB,eCM);
```

2) Generation loop
   • generate one event at a time
   • analyze it (or store for later)
   • add results to histograms
   • print a few events

```
pythia.next();
pythia.process.list();
pythia.event.list();
int id = pythia.event[i].id();
```

3) Finishing step
   • print deduced cross-sections
   • print/save histograms etc.

```
pythia.statistics();
pythia.settings.listChanged();
```
Sample input cards

! This file contains commands to be read in for a Pythia8 run.
! Lines not beginning with a letter are comments.

! 1) Settings that could be used in a main program, if desired.
Main:idBeamA = 2212 ! first beam, p = 2212, pbar = -2212
Main:idBeamB = 2212 ! second beam, p = 2212, pbar = -2212
Main:eCM = 14000. ! CM energy of collision
Main:numberOfEvents = 1000 ! number of events to generate
Main:numberToPrint = 2 ! number of events to print
Main:numberToShow = 50 ! show how far along run is
Main:showChangedSettings = on ! print changed flags/modes/parameters
Main:showAllSettings = off ! print all flags/modes/parameters

! 2) Settings for the hard-process generation.
! Based on an interface to the Fortran Pythia6 program.
#Pythia6:msel = 1 ! QCD production
#Pythia6:ckin(3) = 100. ! pTmin cut
Pythia6:msel = 6 ! t tbar production

! 3) Settings for the event generation process in the Pythia8 library.
#PartonLevel:MI = off ! no multiple interactions
#PartonLevel:ISR = off ! no initial-state radiation
PartonLevel:FSR = off ! no final-state radiation
#HadronLevel:Hadronize = off ! no hadronization
SpaceShower:pT0 = 2.0 ! dampening of pT -> 0 divergence
MultipleInteractions:pTmin = 3.0 ! lower pT cutoff for interactions
Sample output from run

-------- Pythia Flag + Mode + Parameter Settings (changes only) ---------

<table>
<thead>
<tr>
<th>Kind</th>
<th>Name</th>
<th>Now</th>
<th>Default</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>double</td>
<td>Main:eCM</td>
<td>1.40e+04</td>
<td>2000.0000</td>
<td>0.0000</td>
<td>1.00e+05</td>
</tr>
<tr>
<td>double</td>
<td>MultipleInteractions:pTmin</td>
<td>3.0000</td>
<td>2.0000</td>
<td>0.5000</td>
<td>10.0000</td>
</tr>
<tr>
<td>bool</td>
<td>PartonLevel:FSR</td>
<td>off</td>
<td>on</td>
<td></td>
<td></td>
</tr>
<tr>
<td>double</td>
<td>SpaceShower:pT0</td>
<td>2.0000</td>
<td>0.5000</td>
<td>0.0000</td>
<td>10.0000</td>
</tr>
</tbody>
</table>

-------- End Pythia Flag + Mode + Parameter Settings  ------------

-------- Pythia Event Listing (hard process) -------------------------------

<table>
<thead>
<tr>
<th>no</th>
<th>id</th>
<th>name</th>
<th>status</th>
<th>mothers</th>
<th>daughters</th>
<th>colours</th>
<th>p_x</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>90</td>
<td>(system)</td>
<td>-11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>1</td>
<td>2212</td>
<td>(p+)</td>
<td>-12</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0.000</td>
</tr>
<tr>
<td>2</td>
<td>2212</td>
<td>(p+)</td>
<td>-12</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0.000</td>
</tr>
<tr>
<td>3</td>
<td>21</td>
<td>(g)</td>
<td>-21</td>
<td>1</td>
<td>0</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>21</td>
<td>(g)</td>
<td>-21</td>
<td>2</td>
<td>0</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>(tbar)</td>
<td>-22</td>
<td>3</td>
<td>4</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>(t)</td>
<td>-22</td>
<td>3</td>
<td>4</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>24</td>
<td>(W-)</td>
<td>-22</td>
<td>5</td>
<td>0</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>bbar</td>
<td>23</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>24</td>
<td>(W+)</td>
<td>-22</td>
<td>6</td>
<td>0</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>b</td>
<td>23</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>103</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>e-</td>
<td>23</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>-12</td>
<td>nu_ebar</td>
<td>23</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>-1</td>
<td>dbar</td>
<td>23</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>u</td>
<td>23</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>104</td>
</tr>
</tbody>
</table>

Sum: -0.000

-------- End Pythia Event Listing----------------------------------------
<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>status</th>
<th>mothers</th>
<th>daughters</th>
<th>colours</th>
<th>p_x</th>
<th>p_y</th>
<th>p_z</th>
<th>e</th>
<th>m</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>90 (system)</td>
<td>-11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>14000.000</td>
<td>14000.000</td>
</tr>
<tr>
<td>1</td>
<td>2212 (p+)</td>
<td>-12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>7000.000</td>
<td>7000.000</td>
</tr>
<tr>
<td>2</td>
<td>2212 (p+)</td>
<td>-12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>-7000.000</td>
<td>7000.000</td>
</tr>
<tr>
<td>3</td>
<td>21 (g)</td>
<td>-21</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>53.792</td>
<td>53.792</td>
</tr>
<tr>
<td>4</td>
<td>21 (g)</td>
<td>-21</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>-829.022</td>
<td>829.022</td>
</tr>
<tr>
<td>5</td>
<td>-6 (tbar)</td>
<td>-22</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>9</td>
<td>0</td>
<td>102</td>
<td>-107.572</td>
<td>-45.614</td>
</tr>
<tr>
<td>6</td>
<td>6 (t)</td>
<td>-22</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>10</td>
<td>0</td>
<td>103</td>
<td>0.000</td>
<td>-5.011</td>
</tr>
<tr>
<td>7</td>
<td>21 (g)</td>
<td>-41</td>
<td>12</td>
<td>12</td>
<td>11</td>
<td>3</td>
<td>105</td>
<td>102</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>8</td>
<td>21 (g)</td>
<td>-42</td>
<td>13</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>103</td>
<td>101</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>9</td>
<td>-6 (tbar)</td>
<td>-44</td>
<td>5</td>
<td>5</td>
<td>14</td>
<td>14</td>
<td>0</td>
<td>102</td>
<td>-127.853</td>
<td>-17.612</td>
</tr>
<tr>
<td>10</td>
<td>6 (t)</td>
<td>-44</td>
<td>6</td>
<td>6</td>
<td>15</td>
<td>15</td>
<td>103</td>
<td>0</td>
<td>90.752</td>
<td>68.837</td>
</tr>
<tr>
<td>11</td>
<td>21 (g)</td>
<td>-43</td>
<td>7</td>
<td>0</td>
<td>16</td>
<td>16</td>
<td>105</td>
<td>101</td>
<td>37.101</td>
<td>-51.226</td>
</tr>
</tbody>
</table>

Sum: -0.000 -0.000 -0.000 14000.000 14000.000

--- End Pythia Event Listing ---
Sample run with Les Houches input

#include "Pythia.h"
using namespace Pythia8;

int main() {

    int nPrint = 2; // Number of events to print.
    Pythia pythia; // Generator.
    pythia.readLine("PartonLevel:MI = off"); // No multiple interactions.
    pythia.readLine("SpaceShower:pTmin = 1.0"); // Change pTmin cutoff of ISR.
    LHAinitPythia6 lhaInit("sample.init"); // Les Houches initialization object.
    LHAevntPythia6 lhaEvnt("sample.evnt"); // Les Houches event object.
    pythia.init(&lhaInit, &lhaEvnt); // Initialize with pointers.
    cout << lhaInit; // List initialization information.
    Hist nFinal("final particle multiplicity",100,-0.5,499.5); // Histogram.

    int iEvent = 0; // Begin event loop
    while (pythia.next()) { // Generate event until none left.
        if (iEvent++ < nPrint) { // List first few events.
            cout << lhaEvnt; // List Les Houches input event.
            pythia.process.list(); // List Pythia hard-process event.
            pythia.event.list(); // List Pythia complete event.
        }
    }
    int nFin = 0; // Sum up final multiplicity
    for (int i = 0; i < pythia.event.size(); ++i)
        if (pythia.event[i].remains()) nFin++;
    nFinal.fill(nFin); // Fill histogram.
}

    cout << nFinal; // Print histogram.
    return 0; // Done.
}