Whither PYTHIA?

Torbjörn Sjöstrand
Department of Theoretical Physics
Lund University

History
Status
Future
Origin

Q: Why rewrite?
A: Need to clean up!

Q: Why C++?
A: Only game in town!

My original idea:
- simple and robust structure
- throw out-of-date alternatives
- keep current physics $\sim$ unchanged
- many minor improvements
**Lifestyle**

HERWIG camp monolithic (?)

Lund camp pluralistic

**Many programs:**
- JETSET, PYTHIA
- Ariadne, LDCMC
- Fritiof, Luciae
- Lepto (Aroma, Lucifer, PomPyt, ...)
- (Ragpap, Cascade)
- ((HIJING, ...))

**Many physics models:**

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Final-state showers</td>
<td>2 → 3</td>
</tr>
<tr>
<td>Initial-state showers</td>
<td>3 → 4</td>
</tr>
<tr>
<td>String fragmentation</td>
<td>3</td>
</tr>
<tr>
<td>Baryon production</td>
<td>3</td>
</tr>
<tr>
<td>Bose-Einstein</td>
<td>2</td>
</tr>
<tr>
<td>Colour reconnection</td>
<td>3 + not</td>
</tr>
<tr>
<td>Multiple interactions</td>
<td>1 → 2</td>
</tr>
</tbody>
</table>
Evolution

Leif’s idea:
before you worry about physics,
create a generic platform for event generation
= “a language within the language”

HERWIG++ accepted and joined
⇒ basic structure must be physics-neutral

PYTHIA7 ⇐ ThePEG: administration
TheRest: physics
What is ThePEG?

Toolkit for High Energy Physics Event Generation

CLHEP
utilities

ThePEG
basic structure

C++

HERWIG++
physics modules

PYTHIA7
physics modules

Ariadne/LDC
physics modules
What is in TheRest?

- Processes: $QCD \ 2 \rightarrow 2, \ e^+e^- \rightarrow q\bar{q} \ (LL)$
- PDF: GRV 94 series (LL)
- Showers: initial- and final-state (old PYTHIA) (TS)
- Multiple interactions: none
- Beam remnants: Ariadne (LL)
- Fragmentation: simple string (Marc Bertini)
- + simple low-mass corrections (LL)
- Decays: most implemented (LL)

**NOT** useful for physics studies
Mainly simple pieces done ⇒ almost all the hard work remains

- Conversion effort: everything takes longer and costs more
- The physics hurdle is as steep as the C++ learning curve
  ⇒ not convenient to use postdocs
- Need continuity ⇒ as above
- No work for graduate students
A fresh start

- Lower priority than teaching, administration, master’s and graduate students, answering PYTHIA questions, maintaining Fortran code, …
- Leif’s interests are ThePEG, Ariadne, LDC, …
- bad finances in Lund

Solution?: take a sabbatical and work “full-time”!
   (SoFTware development for experiments)

Objectives:
- concentrate on physics, not administration
- pure standard C++, no fancy programming tricks
- independent of ThePEG (or anything else), but
- written to be modular, i.e. easy to interface
- interface to ThePEG later written by Leif(?)
New structure

- internal
- ThePEG
- user

Les Houches Accord (style) Interface

(Remaining resonance decays)

- Initial- and final-state showers
- Multiple interactions and beam remnants
- String fragmentation
- Particle decays

Event record, parameters, data, PDF’s, utilities

Parton-level generators

borderline

Hadron-level generator
Remaining resonance decays (Z, W, H, SUSY, . . .):
- internal or SUSY Les Houches Accord decay tables
- primitive angular correlations

Initial- and final-state showers:
- implement the $p_\perp$-ordered algorithms
- can use existing matrix-element matching code
- (introduce L-CKKW-style mixing, $p_\perp$-ordered)

Multiple interactions and beam remnants:
- based on new scheme under development

String fragmentation:
- reimplement baseline model, minor physics improvements
- low-mass strings
- junction topologies
Particle decays:
  ● update decay tables
  ● (Bose-Einstein; overlaps with fragmentation)

Event record, parameters, data, PDF’s, utilities:
  ● PYTHIA-style event record with LHA colour tags
  ● integrated manual/parameters/data in XML?
  ● LHAPDF parton densities?
  ● simple event analysis (for debug)

Outside scope:
  ● $\gamma p/\gamma^* p/\gamma\gamma/\gamma^*\gamma/\gamma^*\gamma^*$ physics
  ● colour reconnection (WW/ZZ)
  ● SUSY evolution (use SLHA!)
  ● old $e^+e^-$ annihilation machinery ($O(\alpha_S^2)$ ME’s)
  ● independent fragmentation
  ● many out-of-use options
Summary

- Complexity of problem underestimated (C++ & physics)
- “Slave labour” not successful strategy (for me)

⇒ PYTHIA7 (TheRest) nowhere near useful

Tentative schedule:

<table>
<thead>
<tr>
<th>year</th>
<th>time</th>
<th>hadron-level</th>
<th>parton-level</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>fall 2004</td>
<td>begin new assault</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>fall 2005</td>
<td>incomplete draft</td>
<td>LHA-style input</td>
</tr>
<tr>
<td>2</td>
<td>fall 2006</td>
<td>complete, buggy(?)</td>
<td>a few processes</td>
</tr>
<tr>
<td>3</td>
<td>fall 2007</td>
<td>stable, debugged</td>
<td>more processes</td>
</tr>
</tbody>
</table>

…but don’t forget Murphy’s law
Handlers

ThePEG defines a set of abstract Handler classes for hard partonic sub-processes, parton densities, QCD cascades, hadronization, ...

These handler classes interacts with the underlying structure using a special Event Record and a pre-defined set of virtual function definitions.

The procedure to implement e.g. a new hadronization model, is to write a new (C++) class inheriting from the abstract HadronizationHandler base class, implementing the relevant virtual functions.

The structure of the generation process is extremely dynamic:

Besides the standard Handler classes, there is also a general StepHandler class which can do anything and can be inserted anywhere in the generation chain.

In addition, each handler can add steps in the generation chain or redo previous steps depending on the history of each event.
Class Structure of Handlers

- EventHandler
- LuminosityFunction
- CollisionHandler
- StepHandler
- HadronizationHandler
- DecayHandler
- SubProcessHandler
- PartialCollisionHandler
- MEBase
- MIBase
- PartonExtractor
- PhaseSpaceBase
The Particle class provides access to a lot of information. But it only has a pointer to a ParticleData, a Lorentz5Momentum and a pointer to another object carrying the rest of the information (colour, spin etc.) if needed.

Some of this information can be user-defined by creating classes inheriting from e.g. the SpinBase or the completely general EventInfoBase classes. This information can then be accessed through dynamic_casting.
Running ThePEG

The end-user will use a setup program to be able to pick objects corresponding to different physics models to build up an EventGenerator which then can be run interactively or off-line, or as a special slave program e.g. for Geant4.

The setup program is used to choose between a multitude of pre-defined generators, to modify parameters and options of the selected models and, optionally, to specify the analysis to be done on the generated events.

The Repository is the central part of the setup phase. It handles a structured list of all available objects and allows the user to manipulate them.

A flashy Graphical User Interface should be built on top of this Repository. Currently there is only a rudimentary command-line interpreter.

In the end of the run you will get a number of files with statistics and messages. And a \LaTeX-file with references suitable for inclusion in an appendix of a paper.