Fermilab, April 20 2001

Herwig++ Monte Carlo

Alberto Ribon

Rutherford Appleton Laboratory

Outline

- Why do we need a new C++ program?
- Use of "Pythia7 core".
- Example: Cluster Hadronization Model.
- A tentative schedule.
- Conclusions.

Limits of the Fortran code

After more than 10 years since the first version of the program, Herwig has been extensively modified, improved, and expanded.
Although quite successful and heavily used in the High Energy community, the code structure ($\sim 30,000$ lines) shows some typical Fortran disadvantages:

- Many parameters/assumptions hardwired in the code;
- Old-style Fortran hard to understand;
- Many patches/kludges stratified during the years;
- Few changes in one place trigger many other changes all over the program;

$\Rightarrow$ very difficult to mantain and improve!

Requirements for the new program

For the next generation of High Energy experiments at LHC (and beyond) we need a Monte Carlo generator that is:

easy to modify and extend
for at least $10 \div 20$ years.

This requires two things:

- modular structure;
- clear separation between interfaces and implementation details;

$\Rightarrow$ Object-Orientation is the natural solution!
C++ is the natural language.

The Physics improvements

- Widths and resonance shapes; BR(m).
- Showering of Susy coloured particles (gluinos, squarks).
- Non-QCD showering.
- Intermediate step between showering and hadronization (heavy quark effective theory).
- Spin - polarization effects in hadronization and decays.
- Improvements of the multiparton and soft physics models.

Other, more technical, improvements are:

- Multi-event-generator: more processes and "parameters sweeping" in the same Run.
- Particle/Antiparticle differences.
- Better choice of the kinematical variables used in the showering (?)

Multi-scale Showering

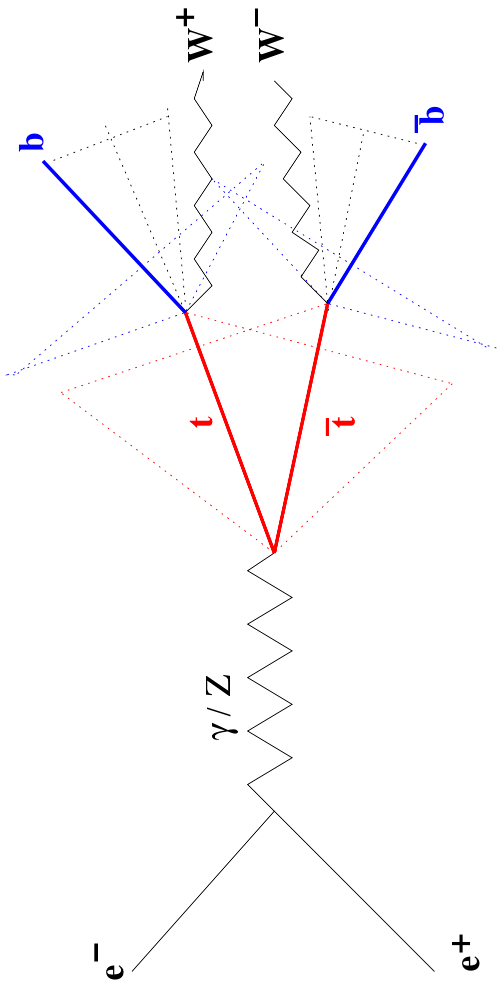
After the hard subprocess, the showering step evolves the partons from the production scale down to a cutoff Q_0, producing new partons in angular order.

Currently in Herwig this is done in one go, assuming:

- $\Gamma \gg Q_0$ decay first, then shower ($\tilde{g}$, $\tilde{q}$).
- $\Gamma \ll Q_0$ shower first, then decay (g , q).

A more correct procedure is to evolve the showering of a coloured particle (with $\Gamma > Q_0$) until Γ (as usual, but vetoing emissions below Γ), then decay it, and then continue the showering of its coloured decay products, first down to Γ (as usual, but vetoing emissions below Γ), and then from Γ to Q_0 with the full partonic final state colour connection.

Example: t with $\Gamma = 5\,GeV$

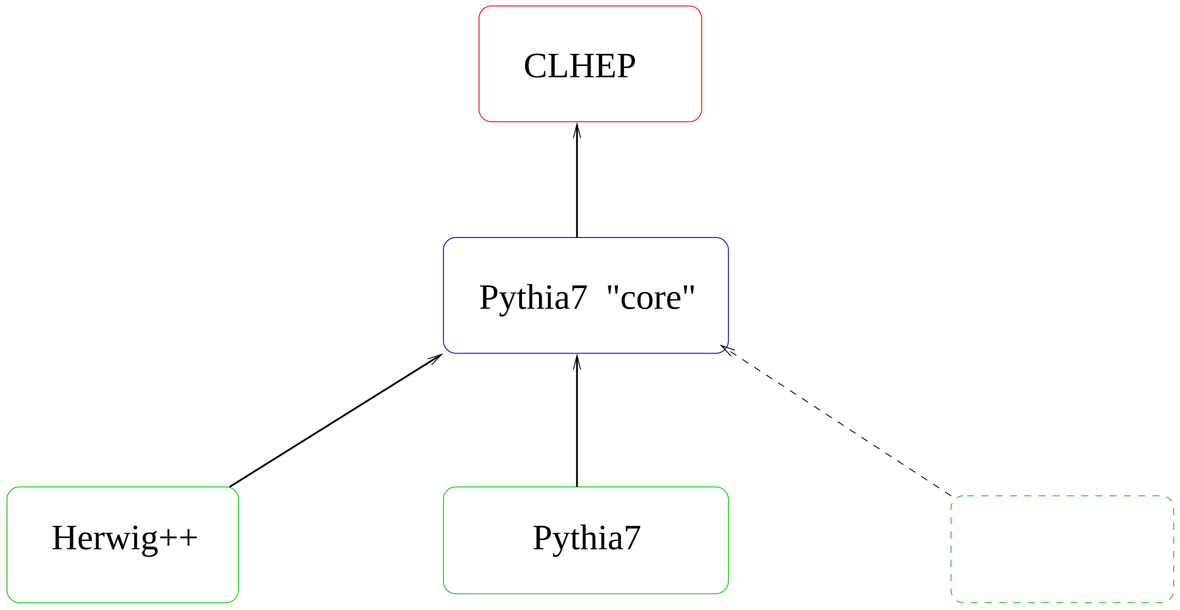


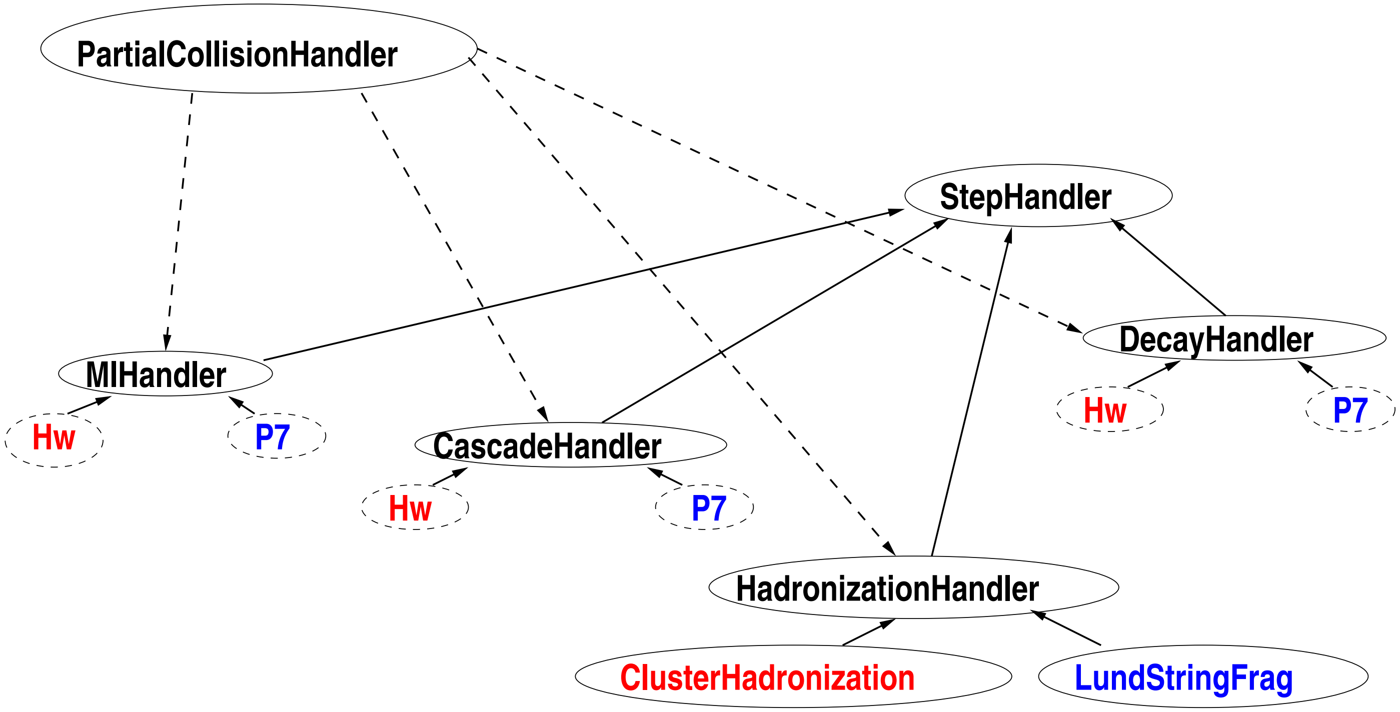
Herwig++

When the Herwig++ project started, one year ago, the basic structure of Pythia7 (that we call "core" or "kernel"), which deals with the user interface, the particles representation, the Event Record, the control flow of operation during the Run, was already implemented.

Furthermore, the design goal of Pythia7 was to provide a general framework (toolkit), independent from any physics models, that can be usable for all Monte Carlo generators.

In short, the administrative part is factorized , the physics part is generator-specific .


CLHEP
Pythia7 "core"
Herwig++
Pythia7


PartialCollisionHandler
StepHandler
MIHandler
DecayHandler
CascadeHandler
HadronizationHandler
Hw
P7
Hw
P7
Hw
P7
ClusterHadronization
LundStringFrag

Advantages of using "Pythia7 core"

For the developers:

- Code reuse: avoid to "reinvent the wheel".
- Sharing (partial or total?) of the subprocess matrix elements.

For the users:

- Same (graphical) user interface for both generators.
- Intermix the two generators:
  Herwig showering + Pythia string fragmentation

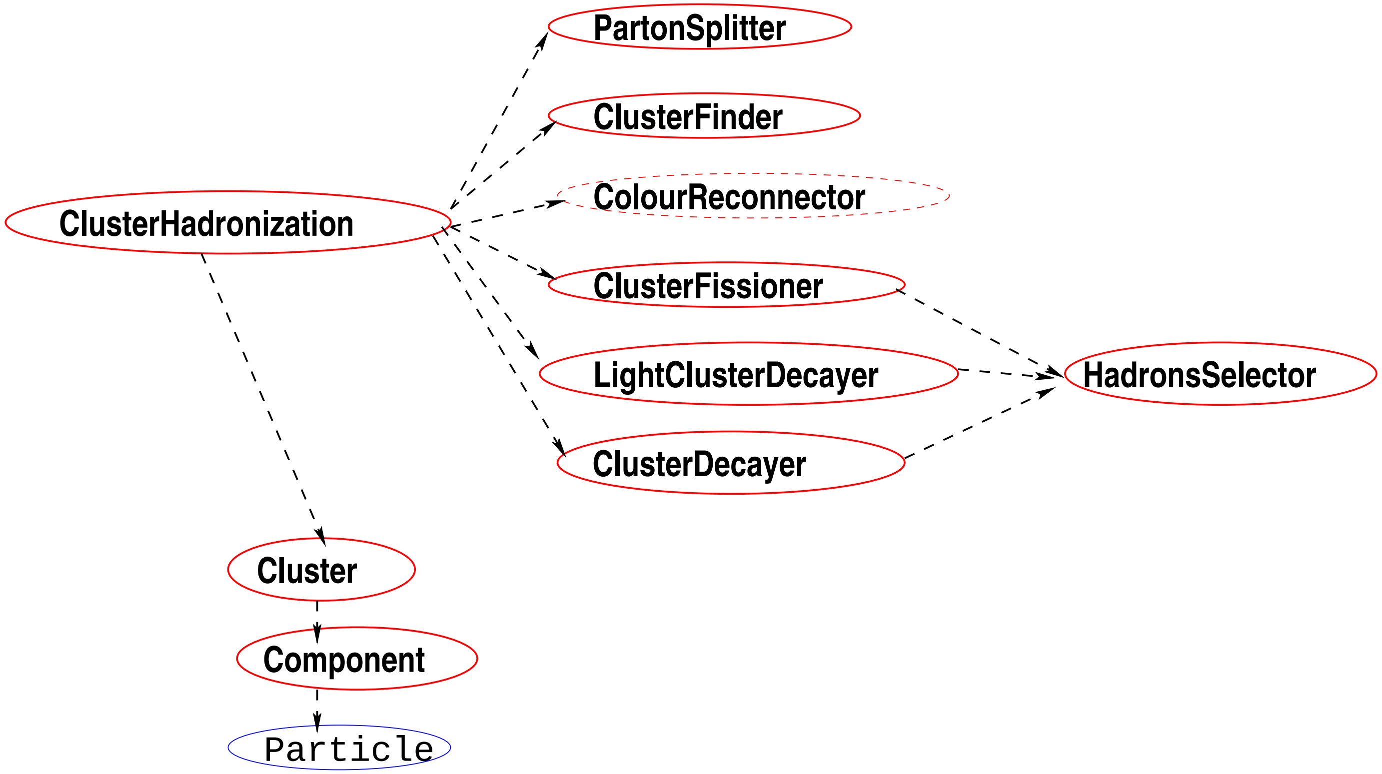
Possible disadvantages

- Inherit specific Pythia7' physics model assumptions that do not match with Herwig++' ones.
- Bugs would spread in both generators rather than compare them to debug each other.
- (general possibilities of large C++ program) large executable size and/or slow performance.

$\Rightarrow$ We will be wary!

Example: Cluster Hadronization

- Non-perturbative gluon splitting $g \to q\bar{q}$.
- Cluster formation
  (usually 2 components, but 3 also should be allowed $\chi^0 \to udd$) .
- Heavy clusters are split $C \to C_1 C_2$.
- Light clusters "decay" into single hadrons
  $C \to h_{lightest} + \text{reshuffling}$.
- Normal clusters decay into two hadrons
  $C \to h_1 h_2$
  (but polarized decays and/or three-body-decays could be considered as well)
  with BRs derived from phase-space.


PartonSplitter
ClusterFinder
ColourReconnector
ClusterHadronization
ClusterFissioner
LightClusterDecayer
HadronsSelector
ClusterDecayer
Cluster
Component
Particle

A tentative schedule

Assuming two people working full time from now, a neither too conservative nor too optimistic schedule for Herwig++ is the following:

- e^+e^- with simple SM subprocesses in one year;
- $p\bar{p}$ with simple SM subprocesses in two years;
- beta version with lepton-lepton, lepton-hadron, hadron-hadron full set of subprocesses in three years;

$\Rightarrow$ it should fit perfectly for LHC experiments, but perhaps also part of the final analyses of Run II experiments at Tevatron could use it!

Conclusions

Herwig++ will be the next generation Monte Carlo generator which replaces Herwig, by including all present features but also with many physics improvements. It will provide the same, easy to use, (graphical) user interface as Pythia7, and with the possibility to intermix the two generators. For the developers, the program will be flexible enough to allow modifications, improvements, and extensions over a long period. Alas, the beta version of the full Herwig++ will take about another three years to be ready!