THE PEG, HERWIG++
and ARIADNE

- Introduction
- Overview
- Status
- Current Work
- ARIADNE
THE PEG, HERWIG++ and ARIADNE

- Introduction
- Overview
- Status
- Current Work
- ARIADNE
- Rivet
What is THEPEG

THEPEG provides a general structure for implementing models for event generation.
What is THEPEG

THEPEG provides a general structure for implementing models for event generation.

Both PYTHIA7 and HERWIG++ are built on THEPEG.
What is THEPEG

THEPEG provides a general structure for implementing models for event generation.

Both PYTHIA7 and HERWIG++ are built on THEPEG.

But it is open for anyone...
What is THEPEG

THEPEG provides a general structure for implementing models for event generation.

Both PYTHIA7 and HERWIG++ are built on THEPEG.

But it is open for anyone...
What is THEPEG

THEPEG provides a general structure for implementing models for event generation.

Both PYTHIA7 and HERWIG++ are built on THEPEG.

But it is open for anyone...

Torbjörn Sjöstrand has left THEPEG and is developing PYTHIA8 on his own.
The components of THEPEG

- **Basic infrastructure**: Smart pointers, extended type information, object persistency, Exceptions, Dynamic loading, ... 
- **Kinematics**: Extra utilities on top of CLHEP vectors, eg. 5-vectors, flat n-body decay, ... 
- **Repository**: Manipulation of interfaced objects. Setting of parameters and switches and connecting objects together. 
- **Handler classes**: to inherit from to implement a specific physics model. 
- **Event record**: Used to communicate between handler classes. 
- **Particle data**: particle properties, decay tables, decayers etc...
THEPEG defines a set of abstract Handler classes for hard partonic sub-processes, parton densities, QCD cascades, hadronization, etc...

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.
When implementing models for event generation there is typically a number of parameters and options available (in addition to the parameters of the Standard Model).

THEPEG defines a uniform way of interacting with the handler classes. The sub-classes may define a set of InterfaceBase objects corresponding to parameters, switches or references to objects of other Interfaced classes.

These are then used by the Repository to manipulate the corresponding member variables in the handler classes.
How to use THEPEG

Running THEPEG is separated into two phases.

- **Setup:**
  A setup program is provided to combine different objects implementing physics models together to build up an EventGenerator object. Here the user can also change parameters and switches etc.

  No C++ knowledge is needed for this. Either use simple setup files with commands or click-and-drag using the Java-based GUI.

  The **Repository** already contains a number of ready-built EventGenerators. It is also possible to specify AnalysisHandler object for an EventGenerator.

  In the end the built EventGenerator is saved to a file.
• **Running:**

The saved EventGenerator can be simply read in and run using a special slave program. If AnalysisHandlers have been specified, this is all you have to do.

Alternatively the file with the EventGenerator can be read into any program where it can be used to generate events which can be sent to analysis or to detector simulation.

The ThePEG::Events can, of course, be translated into HepMC::GenEvents or whatever.
The EventGenerator class is the main class administrating an event generation run.

It maintains global information needed by the different models: The ParticleData objects to be used, a StandardModel object with couplings etc, a RandomGenerator, a list of AnalysisHandlers etc.

It also has an EventHandler object to administer the actual process generation.
THEPEG version 1.0α exists and is working. Snapshots of the current development code is available from http://www.thep.lu.se/ThePEG.

PYTHIA7 version 1.0α exists and is working. Snapshots of the current development code is available from http://www.thep.lu.se/Pythia7.

HERWIG++ is also based on THEPEG. Version 2.0β exists and is working. Can be obtained from http://hepforge.cedar.ac.uk/herwig/.
PYTHIA7/ThePEG includes some basic $2 \to 2$ matrix elements, a couple of PDF parameterizations, remnant handling, initial- and final-state parton showers, Lund string fragmentation and particle decays.

HERWIG++ includes a new parton shower algorithm, improved cluster fragmentation, improved hadron decays. Mainly $e^+e^-$, but also Drell-Yan in hadron collisions.
Portability is a bit tricky since THEPEG relies heavily on the ability to dynamically load libraries/modules.

The build process is using the GNU auto-tools to facilitate portability.

Currently THEPEG runs on any platform,
Portability is a bit tricky since THEPEG relies heavily on the ability to dynamically load libraries/modules.

The build process is using the GNU auto-tools to facilitate portability.

Currently THEPEG runs on any platform, as long as it is Linux with gcc version 3 or later.
Current Work

The code documentation has been converted into Doxygen format. Soon to start with reference and user manual also using Doxygen.

The plan is to have many *Howto* examples to which the user community is welcome to contribute.
Development of PYTHIA7 has stopped

Instead Torbjörn Sjöstrand has gone off by himself to build PYTHIA8 which will NOT be based on THEPEG.

Hopefully it will still be possible to call PYTHIA8 modules (?) from within the THEPEG framework.
• **THEPEG:** Documentation

• **THEPEG:** Basic CKKW ME/PS matching facilities

• **THEPEG:** General interface to external ME generators. Only MadGraph so far.

• **THEPEG:** Spin and Helicity stuff ready (Richardson), but could be expanded to HELAS-like ME generation
• **THEPEG: Documentation**
• **THEPEG: Basic CKKW ME/PS matching facilities**
• **THEPEG: General interface to external ME generators. Only MadGraph so far.**
• **THEPEG: Spin and Helicity stuff ready (Richardson), but could be expanded to HELAS-like ME generation**
• **HERWIG++: Initial state PS (with CKKW)**
• **HERWIG++: SUSY/BSM stuff**
• **HERWIG++: Multiple Interactions à la JIMMY**
• **HERWIG++: All the rest...**
• **THEPEG**: Documentation
• **THEPEG**: Basic CKKW ME/PS matching facilities
• **THEPEG**: General interface to external ME generators. Only MadGraph so far.
• **THEPEG**: Spin and Helicity stuff ready (Richardson), but could be expanded to HELAS-like ME generation
• **HERWIG++**: Initial state PS (with CKKW)
• **HERWIG++**: SUSY/BSM stuff
• **HERWIG++**: Multiple Interactions à la JIMMY
• **HERWIG++**: All the rest…
• **ARIADNE**: Dipole shower with CKKW.
• **ARIADNE**: LDC model with multiple interactions.
<table>
<thead>
<tr>
<th>Standard CKKW</th>
<th>vs. ARIADNE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ktclus to get scales</td>
<td></td>
</tr>
</tbody>
</table>
### CKKW in ARIADNE

<table>
<thead>
<tr>
<th>Standard CKKW</th>
<th>vs. ARIADNE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ktclus to get scales</td>
<td>ARIADNE “backwards” to get scales and intermediate states</td>
</tr>
<tr>
<td>analytic Sudakovs</td>
<td></td>
</tr>
</tbody>
</table>
## CKKW in ARIADNE

<table>
<thead>
<tr>
<th>Standard CKKW</th>
<th>vs. ARIADNE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ktclus to get scales</td>
<td>ARIADNE “backwards” to get scales and intermediate states</td>
</tr>
<tr>
<td>analytic Sudakovs</td>
<td>same Sudakovs as in cascade – truly no-emission probabilities</td>
</tr>
</tbody>
</table>
## CKKW in ARIADNE

<table>
<thead>
<tr>
<th>Standard CKKW</th>
<th>vs. ARIADNE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ktclus to get scales</td>
<td>ARIADNE “backwards” to get scales and intermediate states</td>
</tr>
<tr>
<td>analytic Sudakovs</td>
<td>same Sudakovs as in cascade – truly no-emission probabilities</td>
</tr>
<tr>
<td></td>
<td>Special treatment of highest multiplicity ME</td>
</tr>
</tbody>
</table>
## CKKW in ARIADNE

<table>
<thead>
<tr>
<th>Standard CKKW</th>
<th>vs. ARIADNE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ktclus to get scales</td>
<td>ARIADNE “backwards” to get scales and intermediate states</td>
</tr>
<tr>
<td>analytic Sudakovs</td>
<td>same Sudakovs as in cascade – truly no-emission probabilities</td>
</tr>
<tr>
<td>Also here now</td>
<td>Special treatment of highest multiplicity ME</td>
</tr>
</tbody>
</table>
## CKKW in ARIADNE

<table>
<thead>
<tr>
<th>Standard CKKW</th>
<th>vs. ARIADNE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ktclus to get scales</td>
<td>ARIADNE “backwards” to get scales and intermediate states</td>
</tr>
<tr>
<td>analytic Sudakovs</td>
<td>same Sudakovs as in cascade – truly no-emission probabilities</td>
</tr>
<tr>
<td>Also here now</td>
<td>Special treatment of highest multiplicity ME</td>
</tr>
</tbody>
</table>

Implemented for $e^+e^- \rightarrow$ jets and $pp \rightarrow W +$jets
We do not want analytic Sudakovs, because real ARIADNE Sudakovs resum also some logs of $1/x$. Important for reproducing small-$x$ HERA data.

The dipole model basically only describes emission of gluons. $g \rightarrow q\bar{q}$ put in by hand. Initial-state $q \rightarrow g$ has not been put in. Important for eg. Higgs production at the LHC.

A first THEPEG version of ARIADNE expected this year.
"Robust Independent Validation of Experiment and Theory"

Object oriented C++ replacement for HZTool. Easy comparison (validation) of event generators with published data.

Part of the CEDAR project: http://www.cedar.ac.uk/

Includes jet algorithms and similar tools as does HZTool.
Based on the concept of Projections.

An analysis object takes an HepMC event, applies a number of Projections and fills histograms.

A Projection may use the original events as well as other Projections.

Several different analysis objects/classes are administered by the RivetHandler. If the same Projection is used in several analyses, the actual projection is only done once.
Conclusions