



LUND
UNIVERSITY

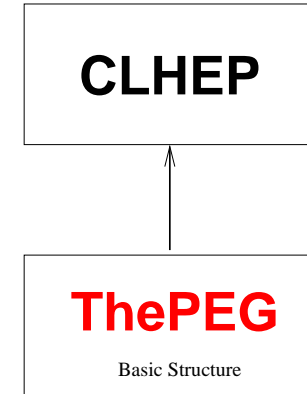
THEPEG, PYTHIA7 and ARIADNE

- Introduction
- Overview
- Status & Future Plans
- ME matching in ARIADNE

Štrbské Pleso
2004.04.15
Leif Lönnblad

What is THEPEG

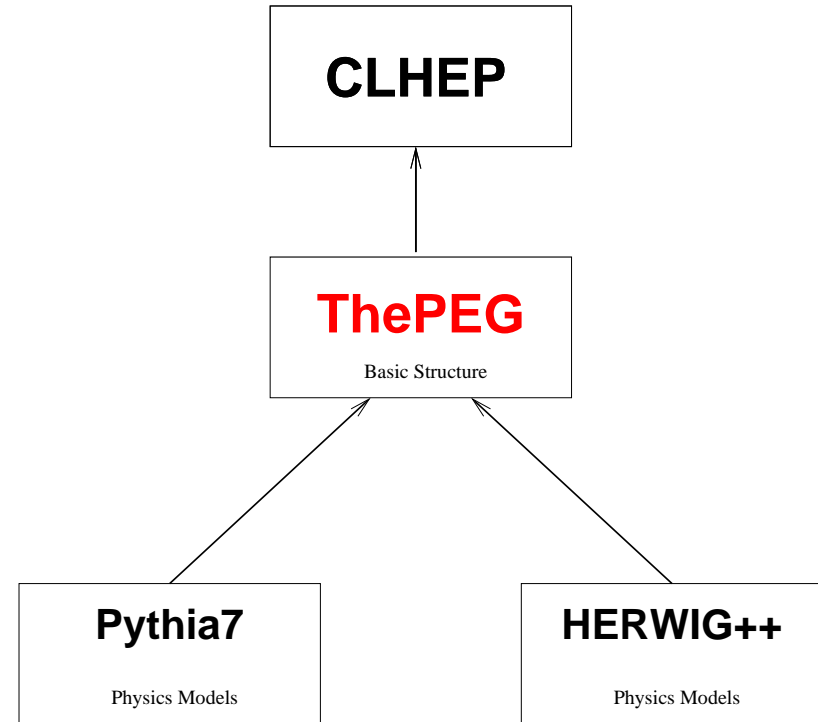
THEPEG is a general and modular C++ framework for implementing event generator models.



What is THEPEG

THEPEG is a general and modular C++ framework for implementing event generator models.

Both PYTHIA7 and HERWIG++ are built on THEPEG.

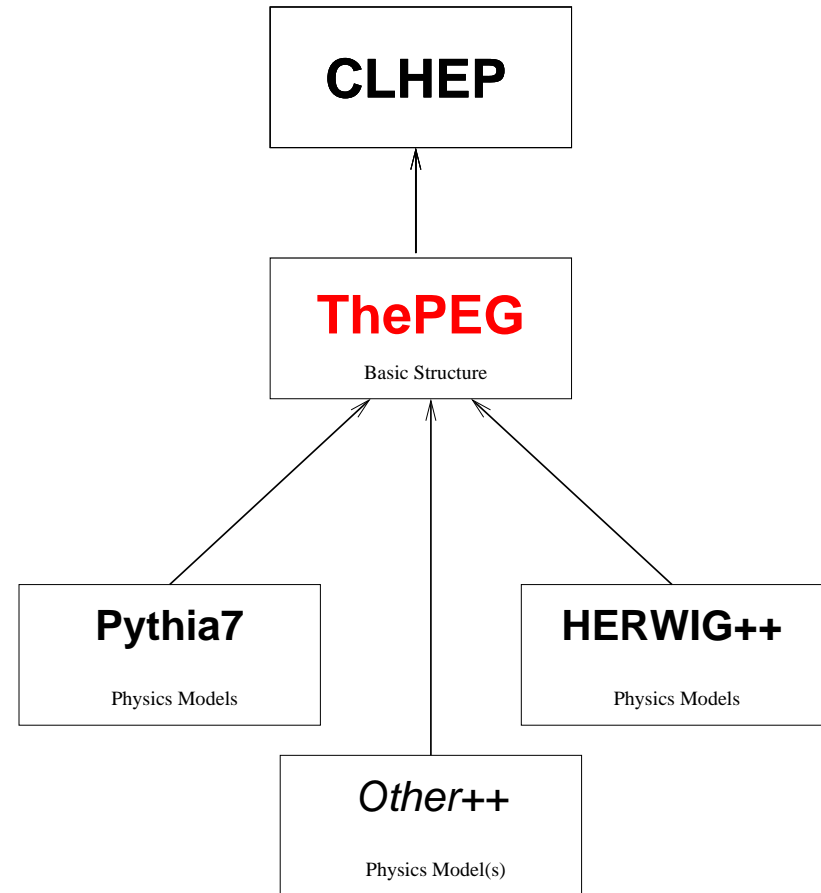


What is THEPEG

THEPEG is a general and modular C++ framework for implementing event generator models.

Both PYTHIA7 and HERWIG++ are built on THEPEG.

But it is open for anyone. . .



The components of THEPEG

- **Basic infrastructure:** Smart pointers, extended type information, object persistency, Exceptions, Dynamic loading, ...
- **Kinematics:** Extra utilities on top of CLHEP vectors, 5-vectors, flat n-body decay, ... should be moved to CLHEP.
- **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.



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. In the future we would like a nice GUI so that the user can just click-and-drag.

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 the file with the EventGenerator can be read into any program which can then use it to generate events ^a which can be sent to analysis or to detector simulation.

^aThePEG::Events which can be translated into HepMC::GenEvents



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 generation.



Status

THEPEG version 1.0 α exists and is working. Snapshots of the current development code is available from <http://www.thep.lu.se/ThePEG>.

PYTHIA7 is now based on THEPEG. Version 1.0 α 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 1.0 exists and is working. Can be obtained from <http://www.hep.phy.cam.ac.uk/theory/Herwig++/>.



PYTHIA7/THEPEG includes some basic $2 \rightarrow 2$ 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. Mainly e^+e^- .



Future Plans

- THEPEG: Documentation
- THEPEG: Java GUI
- THEPEG: Spin and Helicity stuff, almost ready



Future Plans

- THEPEG: Documentation
- THEPEG: Java GUI
- THEPEG: Spin and Helicity stuff, almost ready
- PYTHIA7: Rework fragmentation to include junction strings.
- PYTHIA7: Multiple Interactions.
- PYTHIA7: All the rest...



Future Plans

- THEPEG: Documentation
- THEPEG: Java GUI
- THEPEG: Spin and Helicity stuff, almost ready
- PYTHIA7: Rework fragmentation to include junction strings.
- PYTHIA7: Multiple Interactions.
- PYTHIA7: All the rest...
- HERWIG++: Initial state PS
- HERWIG++: SUSY/BSM stuff
- HERWIG++: All the rest...



Future Plans

- THEPEG: Documentation
- THEPEG: Java GUI
- THEPEG: Spin and Helicity stuff, almost ready
- PYTHIA7: Rework fragmentation to include junction strings.
- PYTHIA7: Multiple Interactions.
- PYTHIA7: All the rest...
- HERWIG++: Initial state PS
- HERWIG++: SUSY/BSM stuff
- HERWIG++: All the rest...
- ARIADNE: Dipole shower with ME matching.
- ARIADNE: LDC model with multiple interactions.



Manpower

- THEPEG: L.L.
- PYTHIA7: L.L., Torbjörn Sjöstrand
- HERWIG++: Stefan Gieseke, Alberto Ribon, Peter Richardson, Mike Seymour, Phil Stephens, Bryan Webber.
- ARIADNE: L.L.



ME matching in ARIADNE

Parton shower generators are not good at describing more than one or two hard jets. If we want more we need to use Matrix Element generators. But we still need parton showers to be able to use hadronization models to get proper jets.

How do we combine ME and PS?



ME matching in ARIADNE

Parton shower generators are not good at describing more than one or two hard jets. If we want more we need to use Matrix Element generators. But we still need parton showers to be able to use hadronization models to get proper jets.

How do we combine ME and PS?

- Make a clean cut in phase space between ME and PS. No double counting and no under counting.
- PS uses Sudakovs to get exclusive states (MLLA resummation of virtual corrections or no-emission probabilities)
- PS has a definite order in the cascade, ME's does not.



A general fixed (second) order calculation

$$O_{+0\text{jet}} = C_{0,0} + C_{0,1}\alpha_s + C_{0,2}\alpha_s^2$$

$$O_{+1\text{jet}} = C_{1,1}\alpha_s + C_{1,2}\alpha_s^2$$

$$O_{+2\text{jet}} = C_{2,2}\alpha_s^2$$

But all the coefficients are divergent in the soft and collinear limit, so we need a cutoff.

When we add PS, we must not add radiation above this cutoff and also not leave out any phase space below it.

But if you add a PS below the cutoff to an N-jet state from an ME generator, the PS assumes there are no other emissions above.



Parton shower generators do things to all orders, summing up all virtual corrections to leading log into Sudakov form factors.

$$\begin{aligned}O_{+0\text{jet}} &= C_{0,0}^{\text{PS}} \Delta_{S0} \\O_{+1\text{jet}} &= C_{1,1}^{\text{PS}} \alpha_s \Delta_{S1} \\O_{+2\text{jet}} &= C_{2,2}^{\text{PS}} \alpha_s^2 \Delta_{S2} \\&\dots\end{aligned}$$

$O_{+1\text{jet}} = C_{1,1}^{\text{PS}} \alpha_s \Delta_{S1}$ is the cross section for to producing **one additional jet** and **nothing else**. The Sudakov form factor is a no-emission probability.



Parton shower generators do things to all orders, summing up all virtual corrections to leading log into Sudakov form factors.

$$\begin{aligned}
 O_{+0\text{jet}} &= C_{0,0}^{\text{PS}} \Delta_{S0} = C_{0,0}^{\text{PS}} + C_{0,1}^{\text{PS}} \alpha_s + C_{0,2}^{\text{PS}} \alpha_s^2 + \dots \\
 O_{+1\text{jet}} &= C_{1,1}^{\text{PS}} \alpha_s \Delta_{S1} = C_{1,1}^{\text{PS}} \alpha_s + C_{1,2}^{\text{PS}} \alpha_s^2 + C_{1,3}^{\text{PS}} \alpha_s^3 + \dots \\
 O_{+2\text{jet}} &= C_{2,2}^{\text{PS}} \alpha_s^2 \Delta_{S2} = C_{2,2}^{\text{PS}} \alpha_s^2 + C_{2,3}^{\text{PS}} \alpha_s^3 + C_{2,4}^{\text{PS}} \alpha_s^4 + \dots \\
 &\dots
 \end{aligned}$$

$O_{+1\text{jet}} = C_{1,1}^{\text{PS}} \alpha_s \Delta_{S1}$ is the cross section for to producing **one additional jet** and **nothing else**. The Sudakov form factor is a no-emission probability.

Also these coefficients are divergent. But when summed to all orders the result is finite.



The CKKW strategy

$$O_{+0\text{jet}} = C_{0,0}^{\text{ME}} \Delta_{S0}$$

$$O_{+1\text{jet}} = C_{1,1}^{\text{ME}} \alpha_s \Delta_{S1}$$

$$O_{+2\text{jet}} = C_{2,2}^{\text{ME}} \alpha_s^2 \Delta_{S2}$$

...

Use tree-level ME generator with some cutoff. Make a jet reconstruction to find a sequence of ordered emissions. Reweight with the Sudakov form factors (and running α_s) and add a (vetoed) parton shower below the cutoff.

The dependence on the cutoff disappears to NNLL. But it is still visible and some tuning is needed.



ARIADNE is special.

- The emissions are ordered in p_{\perp} .
- All partons are always on-shell in the cascade.



ARIADNE is special.

- The emissions are ordered in p_{\perp} . The *inverse* of ARIADNE is a good jet algorithm (DICLUS).
- All partons are always on-shell in the cascade.



ARIADNE is special.

- The emissions are ordered in p_{\perp} . The *inverse* of ARIADNE is a good jet algorithm (DICLUS). Instead of reconstructing intermediate scales with the k_{\perp} -algorithm we reconstruct a likely reconstructed shower history with scales and intermediate states.
- All partons are always on-shell in the cascade.



ARIADNE is special.

- The emissions are ordered in p_{\perp} . The *inverse* of ARIADNE is a good jet algorithm (DCLUS). Instead of reconstructing intermediate scales with the k_{\perp} -algorithm we reconstruct a likely reconstructed shower history with scales and intermediate states.
- All partons are always on-shell in the cascade. The cascade can be started/stopped at any intermediate state.



ARIADNE is special.

- The emissions are ordered in p_{\perp} . The *inverse* of ARIADNE is a good jet algorithm (DCLUS). Instead of reconstructing intermediate scales with the k_{\perp} -algorithm we reconstruct a likely reconstructed shower history with scales and intermediate states.
- All partons are always on-shell in the cascade. The cascade can be started/stopped at any intermediate state. We can make trial emissions from any reconstructed intermediate state and by checking where it ends up we can get the no-emission probability for a phase space region which gives exactly the same Sudakov as ARIADNE.



ARIADNE is special.

- The emissions are ordered in p_{\perp} . The *inverse* of ARIADNE is a good jet algorithm (DCLUS). Instead of reconstructing intermediate scales with the k_{\perp} -algorithm we reconstruct a likely reconstructed shower history with scales and intermediate states.
- All partons are always on-shell in the cascade. The cascade can be started/stopped at any intermediate state. We can make trial emissions from any reconstructed intermediate state and by checking where it ends up we can get the no-emission probability for a phase space region which gives exactly the same Sudakov as ARIADNE.

No cutoff dependence in e^+e^-



Incoming hadrons are even more special in ARIADNE

All gluon radiation is treated as final state emissions from colour dipoles between partons formed in the hard sub-process and between the partons and the hadron remnants.

Coherence effects suppresses radiation from remnant dipoles due to the extendedness of the remnants.

Initial-state $g \rightarrow q\bar{q}$ has to be added by hand as in standard backward evolution PS.

ARIADNE does not generate standard DGLAP evolution (small- x effects are included)



ME generators sample the PDF's at the cutoff scale. ARIADNE uses the scale of the leading order sub-process. Reweighting needed.

$$\frac{x f_i(x, Q^2)}{x f_j(x', E_{\text{cut}})}$$



ME generators sample the PDF's at the cutoff scale. ARIADNE uses the scale of the leading order sub-process. Reweighting needed.

$$\frac{x f_i(x, Q^2)}{x f_j(x', E_{\text{cut}})}$$

In addition to the Sudakov and α_s reweighting, each initial-state (or remnant) emission is reweighted by

$$\Theta_{\text{sup}}(p_{\perp}, z)/z$$

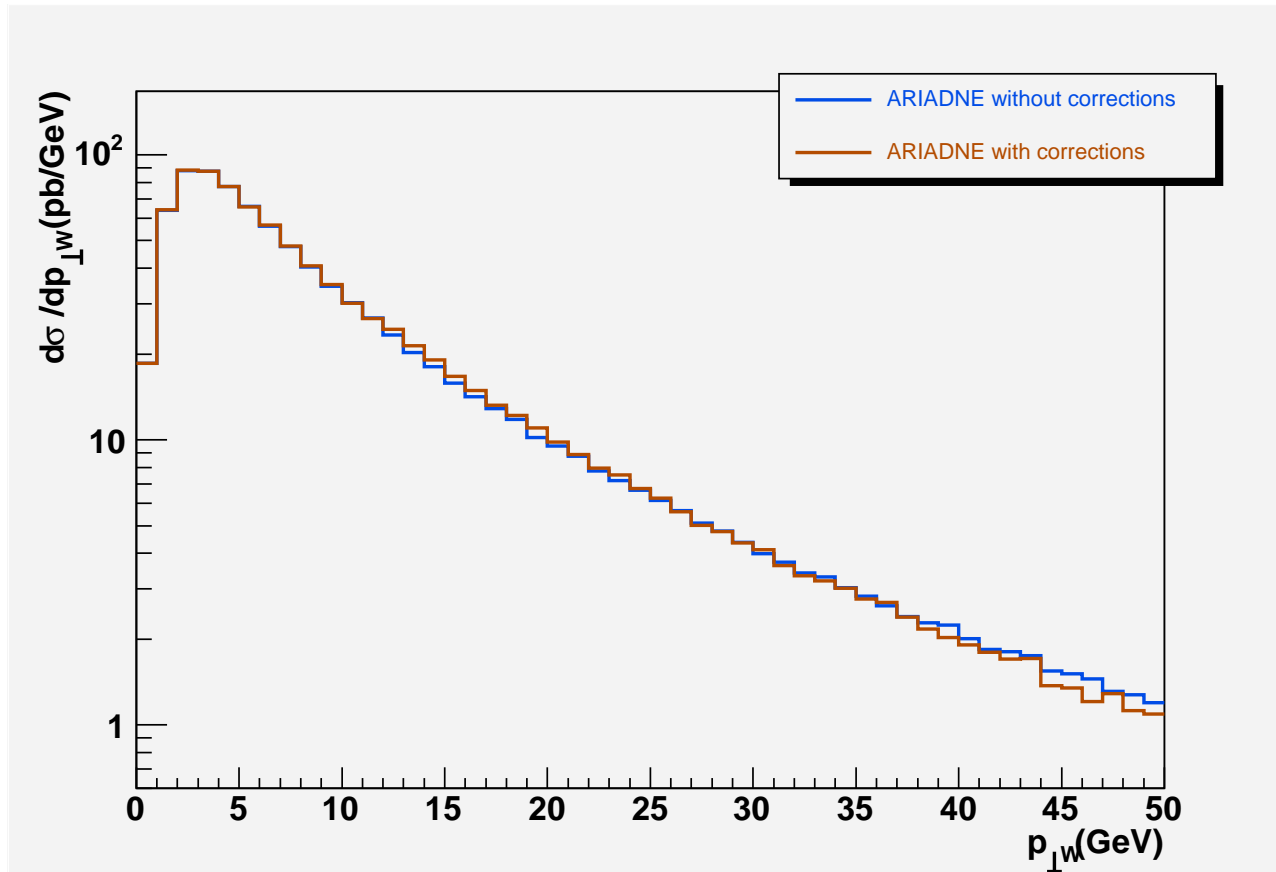
for gluon emission and

$$\frac{x f_g(x/z, p_{\perp}^2)}{x f_q(x, p_{\perp}^2)}$$

for $g \rightarrow q\bar{q}$.



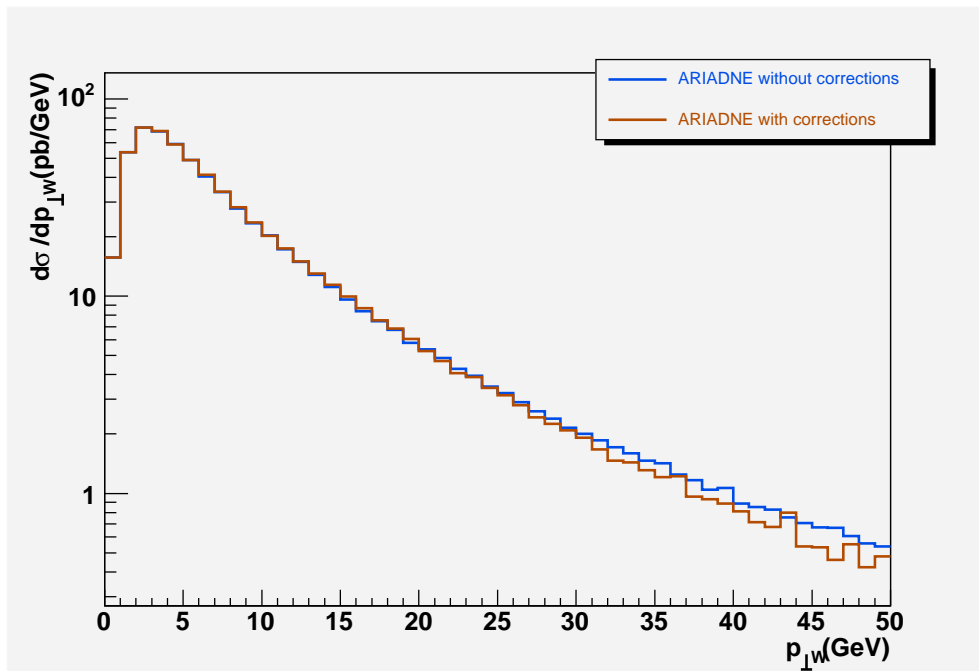
W production @ Tevatron



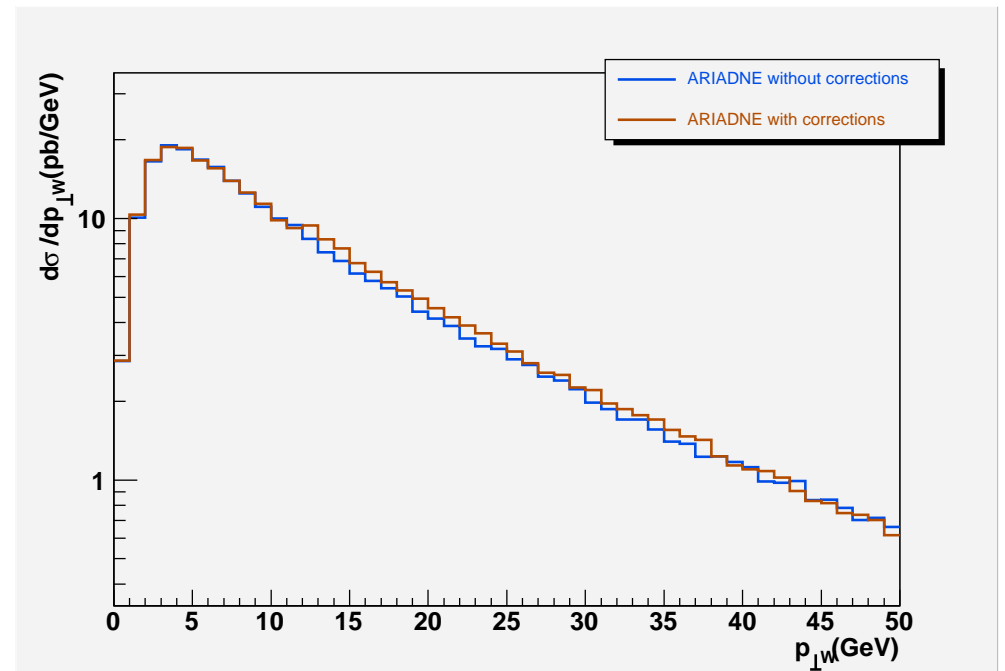
(preliminary)

Only first order correction to check consistency of procedure with standard ARIADNE where the correction is included 'exactly'.





$$q\bar{q}' \rightarrow Wg$$



$$qq \rightarrow Wq'$$



Conclusions

- THEPEG will become the standard generator platform for LHC.
- PYTHIA7/HERWIG++ is coming along slowly but surely.
- ME/PS matching is maturing.
- W-production is almost there.
- DIS is on its way.

