HepMC3 Event Record Library for Monte Carlo Event Generators

http://hepmc.web.cern.ch/hepmc/

Andrii Verbytskyi[1], Andy Buckley[2], David Grellscheid[3], Dmitri Konstantinov[4], Leif Lönnblad[5], James Monk[5,6], Witold Pokorski[4] and Tomasz Przedziński[7]

ACAT 2019, Saas-Fee, Switzerland

March 13, 2019

1 Max-Planck-Institut für Physik 2 University of Glasgow 3 IP3 4 CERN 5 Lund University 6 Niels Bohr Institutet 7 Jagiellonian University in Kraków

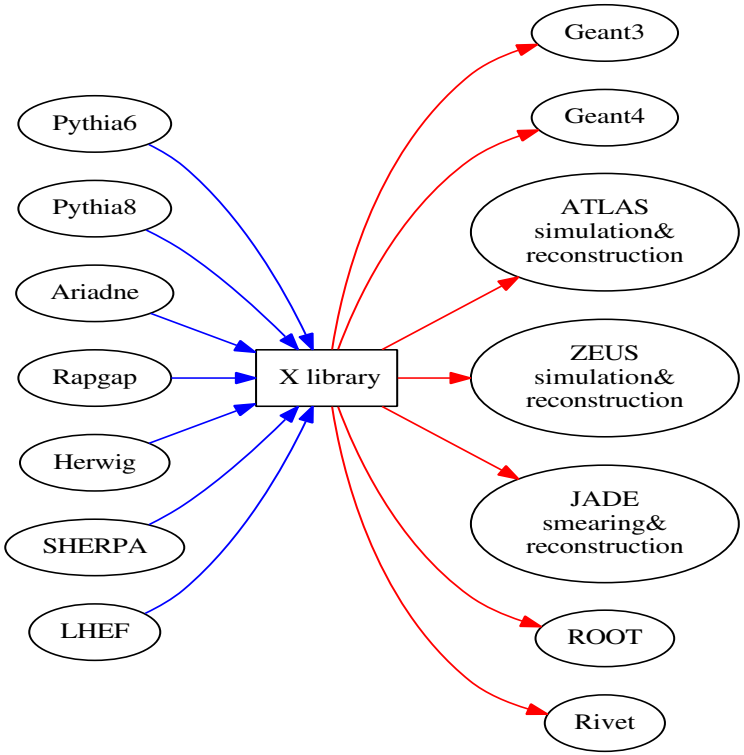
What is HepMC3: Definition and application

- HepMC3 is a library designed to operate with Monte Carlo event records in High Energy physics (HEP).
- Event record contains all physical information on the initial, intermediate and final state particles in the simulated collision: particle flavours, momenta, production coordinates, etc.

$$\Downarrow$$

- The library should be used to store and transfer Monte Carlo event records between different HEP software and/or disk.
- HEP software in focus: Monte Carlo event generators (e.g. Pythia8[1], Herwig7 [2], SHERPA [3]), simulation software (e.g. Geant4 [4], FLUKA [5]), analysis frameworks (ROOT [6], experiment-sepcific), plotting tools (e.g. Rivet [7], HZTOOL [8]) etc.

What is HepMC3: Typical application example



- The typical task is to pass MC generated events betwenn different generators, reconstruction programs and/or analysis frameworks.
- The naive expectation is that a universal translator "X library" should exist for many years, as the task is very common...
- → **No universal translation library existed before 201X!**

Before the HepMC3-3.0.0 the closest match was de-facto a standard HepMC2 [9] and a beta version of HepMC3.

Motivation

So why not HepMC2? Don't fix what is not broken, HepMC2 is working so far!

Fair enough, but:

- Maintenance:
  - HepMC2 is not actively developed for a long time, last release is in **2012**. Many major software projects and repositories should apply patches to the last version.
  - Many small changes would break compatibility but will not allow significant design improvements.
- Features:
  - Many physics-related issues should be updated to meet requirements of **modern physics**, e.g. Heavy Ion information [10], event weight treatment.
  - I/O capabilities should be extended to meet requirements of **modern physicists**, e.g. ROOT, LHEF [11], serialisation of custom information.

HepMC3 is a natural successor of HepMC2. With an idea to keep is good in HepMC2 and add more.

What was added in HepMC3 (vs. HepMC2)

- General features and I/O:
  - Consistent I/O for all popular existing formats of event records.
  - LHEF routines by Leif Lönnblad
  - ROOT interface
  - Simpler event record that allows extensions via attributes
  - Search engine to navigate in the event record
- Programming and interfaces:
  - Multithreading/thread safety
  - CMake instead of autotools should make integration easier
  - "const correctness" for the library
  - Smart pointers everywhere
- Quality assurance:
  - Project in CERN gitlab https://gitlab.cern.ch/hepmc/HepMC3.
  - Continuous integration (64-bit CentOS7 and Ubuntu18).
  - Test engine based on CTest with up to 40 tests. Valgrind included.
  - Ticket system.
- Distribution
  - Closer collaboration with MCEG authors, packagers and (Linux) distributive maintainers.

Attributes in HepMC3

- An attribute is class that holds information on event, particle or vertex and can be represented as a string.
- Every attribute class should inherit from HepMC3::Attribute. The attribute is handled all the time as a string (in memory and in I/O operations) till an explicit request to convert it to an object of some type.

Example of attributes are given below

```
GenCrossSection 2.64422551e+03 2.64422551e+03 -1 -1
```

```
GenPdfInfo 11 -11 9.97420767e-01 9.99999975e-01 9.18812775e+01 1.56824725e+01 2.82148362e+06 0 0
```

```
alphaQCD 0.129844
```

These attributes represent information of classes

```
HepMC3::GenCrossSection
HepMC3::GenPdfInfo
double
```

With attributes one can add arbitrary complex information to event record.

What was removed from HepMC3 (vs. HepMC2)

- General features and I/O
  - Some duplicated functions.
  - operators for input/output.
  - (custom)iterator-based access.
  - Flow and Polarisation classes are removed and the information should be represented by generic attributes. Event record is simplified!

Particle in standard HepMC2 representation (IO_GenEvent)

```
P 10001 11 0.0e+00 0.0e+00 4.59999e+01 4.60000e+01 5.10999e-04 4 0 0 -1 0
```

Particle in standard HepMC3 representation (Asciiv3)

```
  P 1 0 11 0.0e+00 0.0e+00 4.59999e+01 4.60000e+01 5.10999e-04 4
```

The difference stands for polarization and event flow information → rarely filled and meaningful.

```
                                                                    0 0 -1 0
```

I/O: ASCII formats

```
HepMC::Version 3.0.0
HepMC::Ascii3-START_EVENT_LISTING
W 0
E 0 7 12
U GEV MM
W 1.0000000000000000000000e+00
A 0 GenCrossSection 2.64422551e+03 2.64422551e+03 -1 -1
A 0 GenPdfInfo 11 -11 9.97420767e-01 9.99999975e-01 9.18812775e+01 1.56824725e+01 2.82148362e+06 0 0
A 0 alphaQCD 0.129844
A 0 alphaQED 0.007818181
A 0 event_scale 91.88128
A 0 mpi 0
A 0 signal_process_id 221
A 0 signal_process_vertex 0
P 1 0 11 0.0000000000000000e+00 0.0000000000000000e+00 4.5999999997161737e+01 4.6000000000000007e+01 5.1099999999999995e-04 4
P 2 1 11 0.0000000000000000e+00 0.0000000000000000e+00 4.5881355265109356e+01 4.5881355265109356e+01 0.0000000000000000e+00 61
P 3 1 22 0.0000000000000000e+00 0.0000000000000000e+00 1.1864473489064407e-01 1.1864473489064407e-01 0.0000000000000000e+00 1
P 4 0 -11 0.0000000000000000e+00 0.0000000000000000e+00 -4.5999999997161737e+01 4.6000000000000007e+01 5.1099999999999995e-04 4
P 5 4 -11 0.0000000000000000e+00 0.0000000000000000e+00 -4.5999998855671230e+01 4.5999998855671230e+01 0.0000000000000000e+00 61
P 6 4 22 0.0000000000000000e+00 0.0000000000000000e+00 -1.1443287704082650e-06 1.1443287704082650e-06 0.0000000000000000e+00 1
P 7 2 11 0.0000000000000000e+00 0.0000000000000000e+00 4.5881355265109356e+01 4.5881355265109356e+01 0.0000000000000000e+00 21
```

Listing 8: HepMC3 Asciiv3 (HepMC3 standard)

```
HepMC::Version 3.0.0
HepMC::IO_GenEvent-START_EVENT_LISTING
E 0 0 9.188128e+01 1.298440e-01 7.818181e-03 221 0 7 10001 10004 0 1 1.0000000000000000e+00
N 1 "0"
U GEV MM
C 2.6442255100000002e+03 2.6442255100000002e+03
F 11 -11 9.97420767e-01 9.99999975e-01 9.18812775e+01 1.56824725e+01 2.82148362e+06 0 0
V -1 0 0 0 0 0 1 2 0
P 10001 11 0.0000000000000000e+00 0.0000000000000000e+00 4.5999999997161737e+01 4.6000000000000007e+01 5.1099999999999995e-04 4 0 0 -1 0
P 10002 11 0.0000000000000000e+00 0.0000000000000000e+00 4.5881355265109356e+01 4.5881355265109356e+01 0.0000000000000000e+00 61 0 0 -3 0
P 10003 22 0.0000000000000000e+00 0.0000000000000000e+00 1.1864473489064407e-01 1.1864473489064407e-01 0.0000000000000000e+00 1 0 0 0 0
V -2 0 0 0 0 0 1 2 0
P 10004 -11 0.0000000000000000e+00 0.0000000000000000e+00 -4.5999999997161737e+01 4.6000000000000007e+01 5.1099999999999995e-04 4 0 0 -2 0
P 10005 -11 0.0000000000000000e+00 0.0000000000000000e+00 -4.5999998855671230e+01 4.5999998855671230e+01 0.0000000000000000e+00 61 0 0 -4 0
P 10006 22 0.0000000000000000e+00 0.0000000000000000e+00 -1.1443287704082650e-06 1.1443287704082650e-06 0.0000000000000000e+00 1 0 0 0 0
V -3 0 0 0 0 0 0 1 0
```

Listing 9: IO_GenEvent (HepMC2 standard) as implemented in HepMC3

```
E 0 12
 4 -11 0 0 3 11 0.00000000E+00 0.00000000E+00 -4.60000000E+01 4.60000000E+01 5.11000000E-04
        0.00000000E+00 0.00000000E+00 0.00000000E+00 0.00000000E+00
 4 11 0 0 4 12 0.00000000E+00 0.00000000E+00 4.60000000E+01 4.60000000E+01 5.11000000E-04
        0.00000000E+00 0.00000000E+00 0.00000000E+00 0.00000000E+00
 61 -11 1 1 5 5 0.00000000E+00 0.00000000E+00 -4.59999989E+01 4.59999989E+01 0.00000000E+00
        0.00000000E+00 0.00000000E+00 0.00000000E+00 0.00000000E+00
 61 11 2 2 6 6 0.00000000E+00 0.00000000E+00 4.58813553E+01 4.58813553E+01 0.00000000E+00
        0.00000000E+00 0.00000000E+00 0.00000000E+00 0.00000000E+00
 21 -11 3 3 7 7 0.00000000E+00 0.00000000E+00 -4.59999989E+01 4.59999989E+01 0.00000000E+00
```

We keep compatibility!

Listing 10: HepEVT (Fortran era generators) as implemented in HepMC3

Note absence of unused color flow information in Asciiv3.

I/O: LHEF

- The original Les Houches accord event format (LHEF) is designed for communicating between Matrix element generators and MCEG.
- It agreed upon in 2001 [12](see updates in Refs. [11][13][14]).
- **The routines to handle LHEF is a precious part of HepMC3.**

```
<LesHouchesEvents version="3.0">
  <header>
  <MG5ProcCard>
#*************************************************************
#*                        MadGraph 5                         *
#*                                                           *
#*                *                       *                  *
#*                  *        * *        *                   *
#*                    * * * * 5 * * * *                      *
#*                  *        * *        *                   *
#*                *                       *                  *
#*                                                           *
#*                                                           *
#*         VERSION 2.0.0.beta4           2013-06-22          *
#*                                                           *
#*    The MadGraph Development Team - Please visit us at     *
#*    https://server06.fynu.ucl.ac.be/projects/madgraph      *
#*                                                           *
#*************************************************************
#*                                                           *
```

LHEF is basically XML with extra rules.

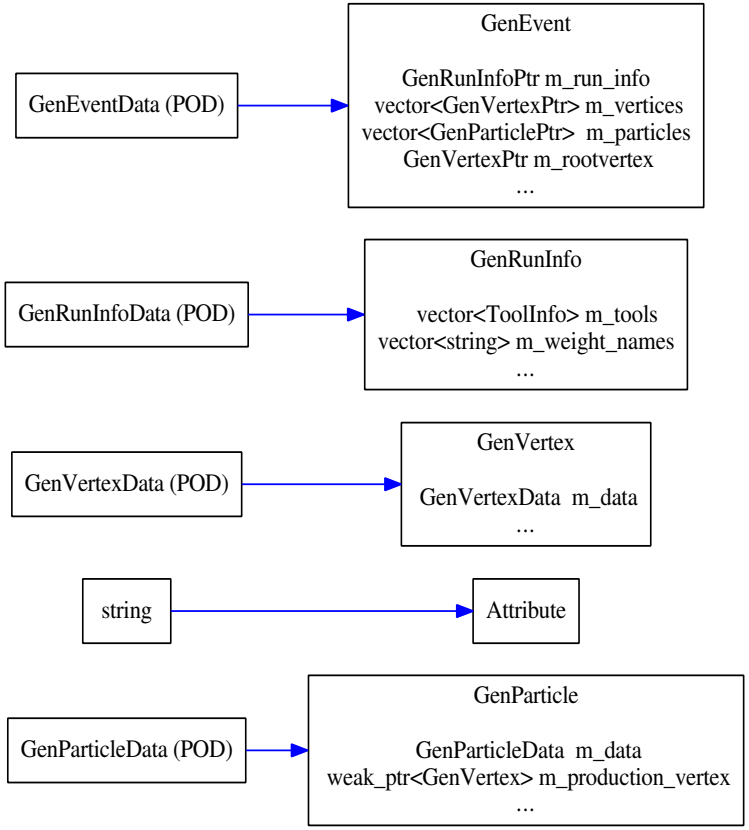
Listing 11: First lines of LHEF header

```
  <event>
  6      66 0.50109093E+02 0.88344569E+02 0.75563862E-02 0.12114027E+00
         2 -1    0    0  502    0 0.00000000E+00 0.00000000E+00 0.62728157E+03 0.62728166E+03 0.33000000E+00 0.0000E+00 0.0000E+00
         5 -1    0    0  501    0 0.00000000E+00 0.00000000E+00 -.44481443E+02 0.44739678E+02 0.48000000E+01 0.0000E+00 0.0000E+00
         6  2    1    2  501    0 -.59350665E+02 0.72244533E+02 0.21037840E+03 0.28804239E+03 0.17311146E+03 0.0000E+00 0.0000E+00
        24  1    3    3    0    0 0.28747403E+02 0.66692858E+02 0.11544955E+03 0.15832494E+03 0.80398000E+02 0.0000E+00 0.0000E+00
         5  1    3    3  501    0 -.88098069E+02 0.55516749E+01 0.94928843E+02 0.12971745E+03 0.48000000E+01 0.0000E+00 0.0000E+00
         1  1    1    2  502    0 0.59350665E+02 -.72244533E+02 0.37242173E+03 0.38397894E+03 0.33000000E+00 0.0000E+00 0.0000E+00
#aMCatNLO 1  6  2  0  0 0.00000000E+00 0.00000000E+00 9  0  0 0.10000000E+01 0.86321681E+00 0.11022720E+01 0.00000000E+00 0.00000000E+00
#aMCatNLO 1  6  2  0  0 0.00000000E+00 0.00000000E+00 9  0  0 0.10000000E+01 0.91292678E+00 0.10493312E+01 0.00000000E+00 0.00000000E+00
#aMCatNLO 1  6  2  0  0 0.00000000E+00 0.00000000E+00 9  0  0 0.10000000E+01 0.91292678E+00 0.10493312E+01 0.00000000E+00 0.00000000E+00
#aMCatNLO 1  6  2  0  0 0.00000000E+00 0.00000000E+00 9  0  0 0.10000000E+01 0.91292678E+00 0.10493312E+01 0.00000000E+00 0.00000000E+00
  <rwgt>
   <wgt id="1001"> 0.50109E+02 </wgt>
   <wgt id="1002"> 0.43255E+02 </wgt>
   <wgt id="1003"> 0.55234E+02 </wgt>
```

Listing 12: First lines of some LHEF event

More: https://hepmc.web.cern.ch/hepmc/group__LHEF.html

I/O: ROOT Tree format



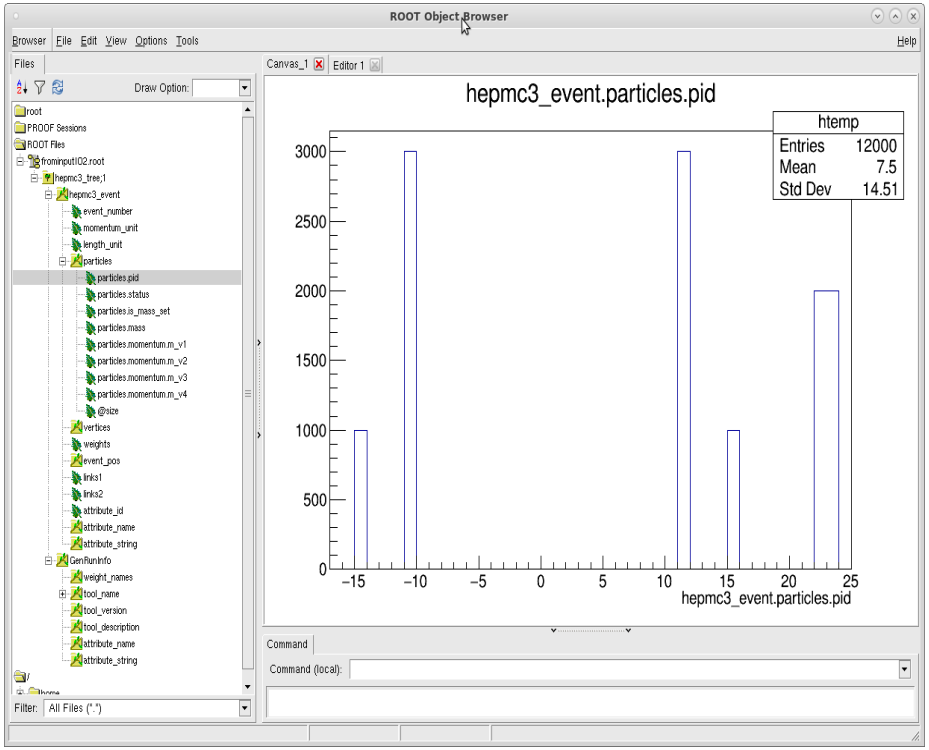


Figure: HepMC3 ROOT file in TBrowser.

HepMC3 uses custom streamers + POD types for ROOT I/O.
HepMC3 ROOT files are readable with standard ROOT, NO externals needed.

I/O: code example

```
#include "HepMC3/GenEvent.h"
#include "HepMC3/WriterAscii.h"
#include "HepMC3/ReaderAsciiHepMC2.h"
int main()
{
    HepMC3::ReaderAsciiHepMC2 inputA("inputIO1.hepmc");
    if(inputA.failed()) return 1;
    HepMC3::WriterAscii       outputA("frominputIO1.hepmc");
    if(outputA.failed()) return 2;
    while( !inputA.failed() )
        {
            HepMC3::GenEvent evt(HepMC3::Units::GEV,HepMC3::Units::MM);
            inputA.read_event(evt);
            if( inputA.failed() )  {printf("End of file reached. Exit.\n"); break;}
            outputA.write_event(evt);
            evt.clear();
        }
    inputA.close();
    outputA.close();
    return 0;
}
```

Listing 13: Reading file in HepMC2 format (IO_GenEvent) and writing in HepMC3 (Asciiv3)

- **Note: Works in the same way for all formats in different combinations.**

Example of event analysis in ROOT (Headers omitted).

```
class SomeAnalysis{
public :
    TChain          *fChain;   //!pointer to the analyzed TTree or TChain
    Int_t            event_number;
    Int_t            momentum_unit;
    TBranch         *b_hepmc3_event_event_number;   //!
    TBranch         *b_hepmc3_event_momentum_unit;   //!
    TBranch         *b_hepmc3_event_length_unit;   //!
    TBranch         *b_hepmc3_event_particles_;   //!
    void Init(TChain *tree) {
        if (!tree) return;
        fChain = tree;
        fChain->SetMakeClass(1);
        fChain->SetBranchAddress("event_number", &event_number, &b_hepmc3_event_event_number);
    SomeAnalysis(const std::string& file) {
        TChain* TempChain= new TChain("hepmc3_tree");
        TempChain->Add(file.c_str());
        Init(TempChain);
    }
};
int main(){
    TH1D H1("H1","Pt of pions or electrons;Events/100MeV;P_{T},GeV",1000,0,100);
    SomeAnalysis* A= new SomeAnalysis("inputIO4.root");
    if (!A->fChain->GetEntries()) return 1;
    for (int entry=0; entry<A->fChain->GetEntries(); entry++){
            A->fChain->GetEntry(entry);
            for (int i=0; i<A->particles_; i++)
                if (A->particles_status[i]==1&&(std::abs(A->particles_pid[i])==211||std::abs(A->particles_pid[i])==11))
                    H1.Fill(std::sqrt(A->particles_momentum_m_v1[i]*A->particles_momentum_m_v1[i]+A->particles_momentum_m_v2[i]*A->particles_momentum_m_v2[i]) );
    }
    delete A;
    H1.Print("All");
    return 0;
}
```

- Reading file in ROOT format without HepMC3.
- Version with HepMC3 is in backups.

Custom classes

HepMC3 I/O is easily extendable for different purposes

- See **30 LOCs** to write events for ZEUS detector simulation.
- Writing events into dot format is given in git repository. To be used with GraphViz:

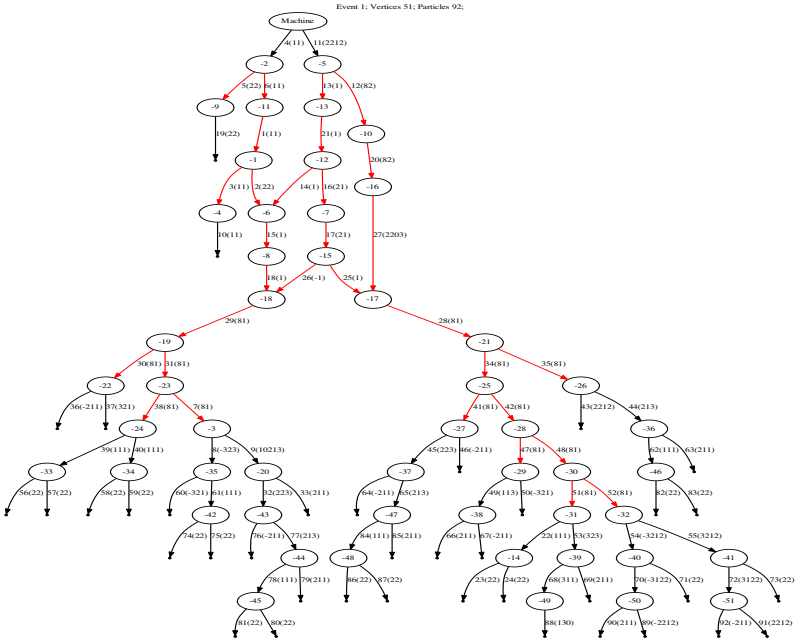


Figure: e^-p collision in Herwig 7.1.4

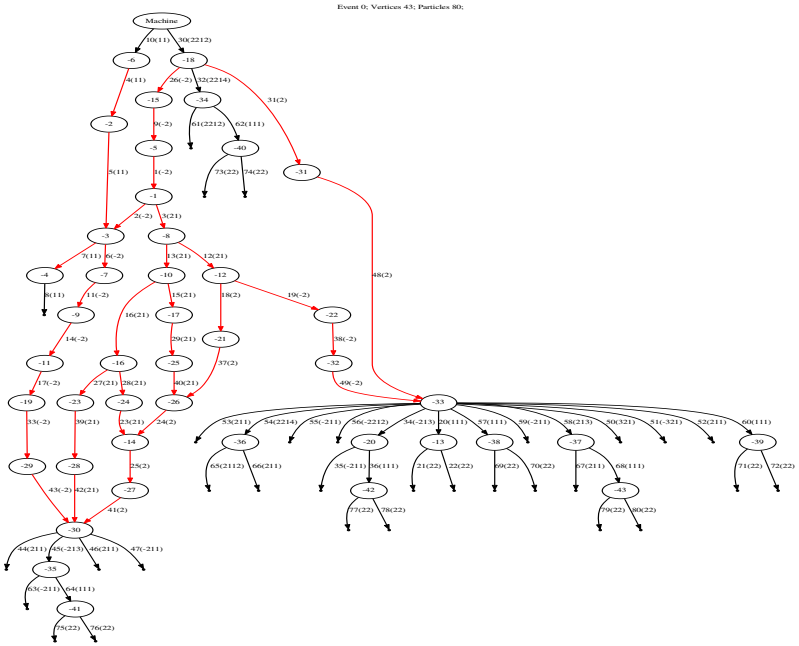


Figure: e^-p collision in Pythia 8.2.40

Format conversion

```
[username@hostname1 outputs]$ bin/convert_example.exe -h
convert_example 3.1
Convert between different file formats of Monte Carlo event record.
Example:
         convert_example -i hepmc2 -o treeroot input.hepmc output.root

Usage: convert_example [OPTIONS]... [inputfile outputfile]...

  -h, --help                      Print help and exit
  -i, --input-format=STRING       Input format  (possible values="hepmc2",
                                    "hepmc3", "hpe", "root", "treeroot",
                                    "lhef") (mandatory)
  -o, --output-format=STRING      Output format  (possible values="hepmc2",
                                    "hepmc3", "hpe", "root", "treeroot",
                                    "treerootopal", "hpezeus", "dump", "dot")
                                    (mandatory)
  -e, --extensions=STRING         Extensions, in a form extension=value, could be
                                    passed to readers/writers
      --events-limit=LONG         Limit of events to read from input
                                    (default=`100000000')
      --first-event-number=LONG   Lowest allowed event number
                                    (default=`-100000000')
      --last-event-number=LONG    Highest allowed event number
                                    (default=`100000000')
      --print-every-events-parsed=LONG
                                  Frequency of parsing information printouts
                                    (default=`100')

[username@hostname1 outputs]$
```

- convert_example.exe is provided in examples.
- Can be recompiller with custom classes.
- Note: "hpezeus", "dot", "treerootopal" are custom classes.

Search engine

The main class to perform selection is HepMC3::Selector.

```
ConstSelectorPtr status = std::make_shared<SelectorWrapper<int> >([](ConstParticlePtr p)->int{return p->status();});
ConstSelectorPtr pt = std::make_shared<SelectorWrapper<double> >([](ConstParticlePtr p)->double{return p->momentum().pt();});
```

One can then use the Selector to construct Filter functions that evaluate on particles, e.g.

```
Filter is_stable = (*status) == 1;
bool stable = is_stable(p);
bool beam = (*status == 4)(p);
```

Selector contains a few standard Selectors already defined, e.g.

```
ConstGenParticlePtr p;
(Selector::STATUS == 1)(p);
(Selector::PT > 15.)(p);
(abs(Selector::RAPIDITY) < 2.5)(p);
```

One can also combined them e.g.

```
Filter myCuts = (Selector::PT > 15.) && (*abs(Selector::RAPIDITY) < 2.5) || (Selector::PT > 100.);
bool passCuts = myCuts(p);
```

Advantages of C++11 in action

Heavy Ion

- GenHeavyIon is a standard attribute to hold information in Heavy Ion collisions.
- Significantly expanded in comparison to HepMC2.
- Can be extended even more, e.g. to comply to Lisbon Accord [10] or needs of Heavy Ion experiments.

```
int    Ncoll_hard;                        /// the number of hard nucleon-nucleon collisions.
int    Npart_proj;                        /// the number of participating nucleons in the projectile.
int    Npart_targ;                        /// the number of participating nucleons in the target.
int    Ncoll;                             /// the number of inelastic nucleon-nucleon collisions.
int    N_Nwounded_collisions;             /// Collisions with a diffractively excited target nucleon.
int    Nwounded_N_collisions;             /// Collisions with a diffractively excited projectile nucleon.
int    Nwounded_Nwounded_collisions;      /// Non-diffractive or doubly diffractive collisions.
double impact_parameter;                  /// The impact parameter.
double event_plane_angle;                 /// The event plane angle.
double sigma_inel_NN;                     /// The assumed inelastic nucleon-nucleon cross section
double centrality;                        /// The centrality.
double user_cent_estimate;                /// A user defined centrality estimator.
int Nspec_proj_n;                         /// The number of spectator neutrons in the projectile
int Nspec_targ_n;                         /// The number of spectator neutrons in the target
int Nspec_proj_p;                         /// The number of spectator protons in the projectile
int Nspec_targ_p;                         /// The number of spectator protons in the target
map<int,double> participant_plane_angles; /// Participant plane angles
map<int,double> eccentricities;           /// Eccentricities
```

Current usage

Code	Type	Implementation
SHERPA-MC	MCEG	3.1/3.0 in SHERPA-MC master/2.2.5
JetScape	MCEG	3.0 in JetScape 1.0
ThePEG	MCEG Toolkit	3.1 in ThePEG master
Herwig7	MCEG	3.1 via ThePEG
Pythia8	MCEG	3.1/3.0 in HepMC3
Pythia6	MCEG	3.1 in HepMC3 examples
Tauola	MCEG	3.1/3.0 in HepMC3
Photos	MCEG	3.1/3.0 in HepMC3
WHIZARD	MCEG	3.0?
EvtGen	MCEG	in touch with authors
GeantV	Simulation	3.0
pyhepmc-ng/scikit-hep	Utility	3.1/3.0 in scikit-hep master
MC-ANALYSER	Analysis/Ploting	HepMC3
Rivet	Analysis/Ploting	work in progress

Note: HepMC3 3.1.0 and HepMC2 can co-exist in one installation → painless migration from HePMC2. +We will help you to find out how to implement HepMC3 support → hepmc-dev@cern.ch.

Requirements

Minimal:

- Modern Linux, OSX or Windows system
- C++11 compatible compiler
- CMake 2.8.X

Recommended=Minimal+

- CMake> 3.3.4
- ROOT6

Full=Recommended+

- doxygen, latex, graphviz for documentation
- Pythia8, HepMC2, Photos [15], Tauola [16], MC-TESTER [17] for tests
- Fortran77 compiler for examples

Contact developers in case something is not working.
Hopefully, there will be no need to know the requirements.

Adoption by major distributives

OS	Tested	Repository	HepMC3 versions	Credits
MacOS	+	homebrew-hep	3.1	Enrico Bothmann
ArchLinux	-	Standard(AUR)	3.1/3.0	Frank Siegert
Debian9 (+Ubuntu18, ...)	+	Standard	(3.1 In progress)	Mo Zhou
Fedora28,29,30,31 (+CentOS7,RHEL7, ...)	+	Standard	3.1	Mattias Ellert
openSUSE42.3 (+SUSE, ...)	+	Multiple unofficial	3.0	
Windows10	+	N/A	3.1	

One should be able to do

```
[username@fedora ~]$ dnf install HepMC3
or
[username@centos ~]$ yum install HepMC3
or
username-macbook% brew install HepMC3
```

without knowing the requirements. ~~Ignorance is Strength.~~

We want to save your time and ensure painless installation.

Summary

- **HepMC3 version 3.1 was released.**
- **Moving towards full replacement of HepMC2.**
- Collection suggestions from LHC experiments, MCEG authors and interested individuals.

Visit http://hepmc.web.cern.ch/hepmc/

- Thanks a lot for the feedback we got recently! The fixes will enter the version 3.1.1 soon.
- Special thanks to people who packaged HepMC3 for distributives: Enrico Bothmann, Mattias Ellert, Frank Siegert and Mo Zhou.

Further developments

- Under discussion/implementation:
  - Python bindings
  - Module with extra attributes, e.g. to hold information on TMDs, Heavy Ion observables, DVCS observables, polarisation of beams, etc.
- Random ideas:
  - Visualisation module, e.g. like VisualHepMC. Largely covered by WriterDOT from examples.
  - PDG information module and utils, e.g. like in HepMC<2.05. Either trivial (masses and widths), does not exist (branching fractions) or depends on used software (particle codes).
- Your suggestions are very welcome!

Event analysis listing of ROOT HepMC3 tree, with HepMC3

```
#include "HepMC3/GenEvent.h"
#include "HepMC3/GenParticle.h"
#include "HepMC3/ReaderRootTree.h"
#include <TH1D.h>
int main()
{

    TH1D H2("H2","Pt of pions or electrons;Events/100MeV;P_{T},GeV",1000,0,100);
    HepMC3::ReaderRootTree inputA("inputI04.root");
    if(inputA.failed()) return 10002;
    while( !inputA.failed() )
        {
            HepMC3::GenEvent evt(HepMC3::Units::GEV,HepMC3::Units::MM);
            inputA.read_event(evt);
            if( inputA.failed() )  {printf("End of file reached. Exit.\n"); break;}
            for(auto p: evt.particles() )
            if ( std::abs(p->status()) == 1 && (std::abs(p->pdg_id()) == 211||std::abs(p->pdg_id
      ()) == 11) )
                H2.Fill( p->momentum().perp());
            evt.clear();
        }
    inputA.close();
    H2.Print("All");
    return 0;
}
```

Listing 22: Reading file in ROOT format with HepMC3

MC events fro ZEUS simulation

One of options for MC simulations in ZEUS was to use ASCII input very similar to HEPEVT format. The implementation of format in HepMC3 is listed below.

```
#include "HepMC3/WriterHEPEVT.h"
namespace HepMC3 {
class WriterHEPEVTZEUS : public  WriterHEPEVT {
public:
    WriterHEPEVTZEUS(const std::string &filename);
    void write_hepevt_event_header();
    void write_hepevt_particle( int index, bool iflong );
};
}
```

Listing 23: Header

```
#include "WriterHEPEVTZEUS.h"
#include "HepMC3/HEPEVT_Wrapper.h"
namespace HepMC3{
WriterHEPEVTZEUS::WriterHEPEVTZEUS(const std::string &filename):WriterHEPEVT(filename) {}
void WriterHEPEVTZEUS::write_hepevt_event_header()
{    fprintf(m_file," E % 12i% 12i% 12i\n",HEPEVT_Wrapper::event_number(),0,HEPEVT_Wrapper::
      number_entries());}
  void WriterHEPEVTZEUS::write_hepevt_particle( int index, bool /*iflong*/ ){
    fprintf(m_file,"% 12i% 8i",HEPEVT_Wrapper::status(index), HEPEVT_Wrapper::id(index));
    fprintf(m_file,"% 8i% 8i",HEPEVT_Wrapper::first_parent(index),HEPEVT_Wrapper::last_parent(
      index));
    fprintf(m_file,"% 8i% 8i",HEPEVT_Wrapper::first_child(index),HEPEVT_Wrapper::last_child(
      index));
    fprintf(m_file,      "% 19.11E% 19.11E% 19.11E% 19.11E% 19.11E\n",HEPEVT_Wrapper::px(index),
      HEPEVT_Wrapper::py(index),HEPEVT_Wrapper::pz(index),HEPEVT_Wrapper::e(index),
      HEPEVT_Wrapper::m(index));
    fprintf(m_file, "%-52s% 19.11E% 19.11E% 19.11E% 19.11E% 19.11E\n"," ",HEPEVT_Wrapper::x(
      index),HEPEVT_Wrapper::y(index),HEPEVT_Wrapper::z(index),HEPEVT_Wrapper::t(index),0.0);
}
}// namespace HepMC3
```

Listing 24: Source

Installation on CentOS7

```
[root@hostname1 username]# yum install HepMC3 HepMC3-devel  HepMC3-search HepMC3-search-devel  HepMC3-rootIO HepMC3-rootIO-devel HepMC3-interfaces-devel HepMC3-doc
Resolving Dependencies
Skipping filters plugin, no data
--> Running transaction check
---> Package HepMC3.x86_64 0:3.1.0-3.el7 will be installed
---> Package HepMC3-devel.x86_64 0:3.1.0-3.el7 will be installed
---> Package HepMC3-doc.noarch 0:3.1.0-3.el7 will be installed
---> Package HepMC3-interfaces-devel.noarch 0:3.1.0-3.el7 will be installed
---> Package HepMC3-rootIO.x86_64 0:3.1.0-3.el7 will be installed
---> Package HepMC3-rootIO-devel.x86_64 0:3.1.0-3.el7 will be installed
---> Package HepMC3-search.x86_64 0:3.1.0-3.el7 will be installed
---> Package HepMC3-search-devel.x86_64 0:3.1.0-3.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

======================================================================================================
 Package                                 Arch                Version             Repository      Size
======================================================================================================
Installing:
 HepMC3                                  x86_64              3.1.0-3.el7         epel-testing   218 k
 HepMC3-devel                            x86_64              3.1.0-3.el7         epel-testing    52 k
 HepMC3-doc                              noarch              3.1.0-3.el7         epel-testing    43 M
 HepMC3-interfaces-devel                 noarch              3.1.0-3.el7         epel-testing    21 k
 HepMC3-rootIO                           x86_64              3.1.0-3.el7         epel-testing    43 k
 HepMC3-rootIO-devel                     x86_64              3.1.0-3.el7         epel-testing   5.5 k
 HepMC3-search                           x86_64              3.1.0-3.el7         epel-testing    29 k
 HepMC3-search-devel                     x86_64              3.1.0-3.el7         epel-testing    10 k

Transaction Summary
======================================================================================================
Install  8 Packages

Total download size: 44 M
Installed size: 85 M
Is this ok [y/d/N]: y
Downloading packages:
HepMC3-doc-3.1.0-3.el7.noarch.rpm                             |  43 MB  00:00:01
(1/7): HepMC3-3.1.0-3.el7.x86_64.rpm                          | 218 kB  00:00:00
(2/7): HepMC3-devel-3.1.0-3.el7.x86_64.rpm                    |  52 kB  00:00:00
(3/7): HepMC3-interfaces-devel-3.1.0-3.el7.noarch.rpm         |  21 kB  00:00:00
(4/7): HepMC3-rootIO-3.1.0-3.el7.x86_64.rpm                   |  43 kB  00:00:00
(5/7): HepMC3-rootIO-devel-3.1.0-3.el7.x86_64.rpm             | 5.5 kB  00:00:00
(6/7): HepMC3-search-3.1.0-3.el7.x86_64.rpm                   |  29 kB  00:00:00
(7/7): HepMC3-search-devel-3.1.0-3.el7.x86_64.rpm             |  10 kB  00:00:00
------------------------------------------------------------------------------------------------------
Total                                                874 kB/s | 378 kB  00:00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : HepMC3-3.1.0-3.el7.x86_64                    1/8
  Installing : HepMC3-devel-3.1.0-3.el7.x86_64              2/8
  Installing : HepMC3-search-3.1.0-3.el7.x86_64             3/8
  Installing : HepMC3-rootIO-3.1.0-3.el7.x86_64             4/8
  Installing : HepMC3-rootIO-devel-3.1.0-3.el7.x86_64       5/8
  Installing : HepMC3-search-devel-3.1.0-3.el7.x86_64       6/8
  Installing : HepMC3-interfaces-devel-3.1.0-3.el7.noarch   7/8
  Installing : HepMC3-doc-3.1.0-3.el7.noarch                8/8
  Verifying  : HepMC3-devel-3.1.0-3.el7.x86_64              1/8
  Verifying  : HepMC3-3.1.0-3.el7.x86_64                    2/8
  Verifying  : HepMC3-search-devel-3.1.0-3.el7.x86_64       3/8
  Verifying  : HepMC3-search-3.1.0-3.el7.x86_64             4/8
  Verifying  : HepMC3-doc-3.1.0-3.el7.noarch                5/8
  Verifying  : HepMC3-rootIO-devel-3.1.0-3.el7.x86_64       6/8
  Verifying  : HepMC3-rootIO-3.1.0-3.el7.x86_64             7/8
  Verifying  : HepMC3-interfaces-devel-3.1.0-3.el7.noarch   8/8

Installed:
  HepMC3.x86_64 0:3.1.0-3.el7         HepMC3-devel.x86_64 0:3.1.0-3.el7         HepMC3-doc.noarch 0:3.1.0-3.el7  HepMC3-interfaces-devel.noarch 0:3.1.0-3.el7  HepMC3-rootIO.x86_64 0:3.1.0-3.el7  HepMC3-rootIO-devel.x86_64 0:3.1.0-3.el7
  HepMC3-search.x86_64 0:3.1.0-3.el7  HepMC3-search-devel.x86_64 0:3.1.0-3.el7

Complete!
New leaves:
  HepMC3-doc.noarch
  HepMC3-interfaces-devel.noarch
  HepMC3-rootIO-devel.x86_64
  HepMC3-search-devel.x86_64
```

Bibliography I

[1] T. Sjöstrand, S. Mrenna and P.Z. Skands,
A brief introduction to PYTHIA 8.1.
Comput. Phys. Commun. 178, 852 (2008).
arXiv:0710.3820.

[2] J. Bellm et al.,
Herwig 7.0/Herwig++ 3.0 release note.
Eur. Phys. J. C76, 196 (2016).
arXiv:1512.01178.

[3] T. Gleisberg et al.,
Event generation with SHERPA 1.1.
JHEP 02, 007 (2009).
arXiv:0811.4622.

[4] GEANT4, S. Agostinelli et al.,
GEANT4: A Simulation toolkit.
Nucl. Instrum. Meth. A506, 250 (2003).

[5] A. Ferrari et al.,
FLUKA: A multi-particle transport code (Program version 2005).
(2005).

[6] I. Antcheva et al.,
ROOT: A C++ framework for petabyte data storage, statistical analysis and visualization.
Comput. Phys. Commun. 182, 1384 (2011).

[7] A. Buckley et al.,
Rivet user manual.
Comput. Phys. Commun. 184, 2803 (2013).
arXiv:1003.0694.

Bibliography II

[8] J. Bromley et al.,
HZTOOL: A package for Monte Carlo-data comparison at HERA (version 1.0),
Future physics at HERA. Proceedings, Workshop, Hamburg, Germany, September 25, 1995-May 31, 1996. Vol. 1, 2.
(1995).

[9] M. Dobbs and J.B. Hansen,
The HepMC C++ Monte Carlo event record for High Energy Physics.
Comput. Phys. Commun. 134, 41 (2001).

[10] J.G. Milhano et al.,
Lisbon Accord: a standard format for heavy-ion event generators.
(2017).

[11] J. Alwall et al.,
A Standard format for Les Houches event files.
Comput. Phys. Commun. 176, 300 (2007).
arXiv:hep-ph/0609017.

[12] E. Boos et al.,
Generic user process interface for event generators.
(2001).
arXiv:hep-ph/0109068.

[13] J.M. Butterworth et al.,
THE TOOLS AND MONTE CARLO WORKING GROUP Summary Report from the Les Houches 2009 Workshop on TeV Colliders.
(2010).
arXiv:1003.1643.

Bibliography III

[14] J.R. Andersen et al.,
Les Houches 2013: Physics at TeV Colliders: Standard Model Working Group Report.
(2014).
arXiv:1405.1067.

[15] E. Barberio and Z. Was,
PHOTOS: A Universal Monte Carlo for QED radiative corrections. Version 2.0.
Comput. Phys. Commun. 79, 291 (1994).

[16] S. Jadach, J.H. Kühn and Z. Was,
TAUOLA: a library of Monte Carlo programs to simulate decays of polarized tau leptons.
Comput. Phys. Commun. 64, 275 (1990).

[17] P. Golonka,T. Pierzchala and Z. Was,
MC-TESTER: A Universal tool for comparisons of Monte Carlo predictions for particle decays in high-energy physics.
Comput. Phys. Commun. 157, 39 (2004).
arXiv:hep-ph/0210252.