# Physics generators and their availability for FCCSW

FCCSW Workshop October 2019

Oct 02, 2019

Gerardo Ganis for the FCC Software Group
CERN

## Content of the Generators section

- Overview of the processes of interest and availability
- Overview of the ways Generators are interfaced to FCCSW

- Concrete focus on the LEP legacy MC family of generators (S Jadach)

## Why do we need Monte Carlo Generators

- To understand the potential of a detector solution we need to simulate all what is going on
- This means signals and backgrounds
- Backgrounds
    - Unwanted collision products / signals
    - Beam-related backgrounds (SR, ...)
    - Bean unrelated backgrounds (cosmics rays, ...)
- Detectors are complex, analytical calculations are not realistic
- Monte Carlo perform integrations otherwise impossible
- Many programs exists to simulate the most relevant processes
- A list of the ones relevant for FCC-ee provided by S Jadach at the January workshop

# Some Monte Carlo Generators relevant for FCC-ee

- PYTHIA8
  - General purpose for signal generation, hadronization, input handling
- WHIZARD
  - General purpose for signal generation
- MCSANC
  - e+e- -> ffbar, ZH
- BabaYaga
  - e+e- -> gamma gamma
- RacoonWW
  - e+e- -> 4 fermion
- BHLUMI
- KORALW, YFSWW, KKMC
- TAUOLA

$\xrightarrow{\quad}$ Next talk by S Jadach

## Monte Carlo Generators relevant for FCC-ee (2)

- Beam-related backgrounds (no exhaustive list)
    - MDISim: SR, Single beam induced backgrounds
    - SYNC_BKG, SYNRAD+: SR
    - GuineaPig++: IP backgrounds
        - (In)coherent Pairs Creation, γγ to hadrons
    - Pythia8: γγ to hadrons
    - BBBrem+SAD: Radiative Bhabhas
- Cosmic rays background is small and measured with data

SR: Synchrotron Radiation

## Generators repository: GenSer @ LCG stacks

- Generator Service Project in EP-SFT
  Collaborate with the authors of Monte Carlo generators and with the LHC experiments to prepare validated code for both the theoretical and experimental communities at the LHC.
  Tasks include user support, assistance for the development of the new object-oriented generators and maintenance of existing Monte Carlo packages on the LCG-supported platforms.
- Actively used by ATLAS, LHCb, SWAN, and some SME experiments
- Available on CVMFS as part of a give stack

```
$ ls /cvmfs/sft.cern.ch/lcg/releases/LCG_96b/MCGenerators/
FORM chaplin evtgen hepmcanalysis hijing hydjet lhapdf mctester photos professor pythia8 rivet syscalc
vbfnlo agile collier feynhiggs heputils hjets hydjet++ log4cpp  mcutils photos++ prophecy4f qd sherpa tauola
vincia alpgen crmc gosam herwig hoppet jhu looptools njet powheg-box pyquen qgraf starlight tauola++
whizard baurmc dire4pythia8 gosam_contrib herwig++ hto4l jimmy madgraph5amc openloops
powheg-box-v2 pythia6 rapidsim superchic thepeg yoda
```

## Monte Carlo Generators and FCCSW

G A Stewart, FCC week 2019

- Interoperability Level 0 - Common Data Formats
  - Allows interoperability between different programs, running on different hardware
  - Examples: HepMC, Les Houche Events
- Interoperability Level 1 - Callable interfaces
  - API defined by the programming language
  - Can depend on the compiler / language version (e.g. C++)
  - Examples: Boost, FastJet, Pythia8

- Monte Carlo generators are typically standalone codes
  - A noticeable exception is Pythia8, which provides a callable interface
- FCCSW adopts both approaches
  - Level 0: facilitates access to new Generators, if a known common data format is used
  - Level 1: with general purpose products, may extend the functionality available in the framework

## Pythia8 and FCCSW

- Pythia8 integration in FCCSW is an example of level 1 interoperability
- The relevant code is located under Generation/src/components/

  PythiaInterface.h, PythiaInterface.cpp

- PythiaInterface is a GaudiTool and a IHepMCProviderTool
  - Used to create an algorithm to be scheduled in Gaudi
  - Provides a HepMC event for the transient store

## Pythia8Interface

```
class PythiaInterface : public GaudiTool, virtual public IHepMCProviderTool {
public:
  /// Constructor.
  PythiaInterface(const std::string& type, const std::string& name,
                                             const IInterface* parent);
  virtual StatusCode initialize();
  virtual StatusCode finalize();
  virtual StatusCode getNextEvent(HepMC::GenEvent& theEvent);
private:
  /// Pythia8 engine
  std::unique_ptr<Pythia8::Pythia> m_pythiaSignal;
  /// Name of Pythia configuration file with Pythia simulation settings & input LHE file
(if required)
  Gaudi::Property<std::string> m_parfile{this, "Filename",
    "Generation/data/Pythia_minbias_pp_100TeV.cmd", "Name of the Pythia cmd file"};
```

## Pythia8Interface implementation

```
StatusCode PythiaInterface::initialize() {

// Initialize pythia
   m_pythiaSignal = std::unique_ptr<Pythia8::Pythia>(new Pythia8::Pythia(xmlpath));

// Read Pythia configuration files
   m_pythiaSignal->readFile(m_parfile.value().c_str());

(... other initializations …)

// Overall initialization
   m_pythiaSignal->init();

}
```

## Pythia8 behavior determined by config file

- Few useful examples available under Generation/data
    - Pythia_minbias_pp_100TeV.cmd
    - Pythia_standard.cmd
        - pp -> HardQCD
    - Pythia_ttbar.cmd
    - Pythia_pp_h_4l.cmd
    - Pythia_pp_zgzg_4l.cmd
    - ee_Z_ddbar.cmd
        - e+e- -> q qbar @ 91 GeV

## How to run Pythia8 in FCCSW

- Pythia is run via the GAUDI infrastructure and the steering **`fccrun`** script
- Example: generate ttbar events and process them with Delphes

```
$ fccrun FCCSW/Sim/SimDelphesInterface/options/PythiaDelphes_config.py \
        --Filename FCCSW/Generation/data/Pythia_ttbar.cmd \
        --DelphesCard FCCSW/Sim/SimDelphesInterface/data/FCChh_DelphesCard_Baseline_v01.tcl
```

## Pythia8 as Les Houches Event (LHE) reader

- Pythia8 offers an interface with the LHE files
- The Beams: identifier card can point to a LHE file
- FCCSW has a special Pythia confguration file for that

Generation/data/Pythia_LHEinput.cmd

```
! 4) Read-in Les Houches Event file - alternative beam and process selection.
Beams:frameType = 4                          ! read info from a LHEF
Beams:LHEF = Generation/data/events.lhe      ! the LHEF to read from
```

- PythiaInterface can be used for any code creating LHE files!

## Whizard

WHIZARD is a program system designed for the efficient calculation of multi-particle scattering cross sections and simulated event samples. (...) Tree-level matrix elements are generated automatically for arbitrary partonic processes by using the Optimized Matrix Element Generator O'Mega. (...)
WHIZARD supports the Standard Model and a huge number of BSM models. Model extensions or completely different models can be added. There are also interfaces to FeynRules and SARAH

- Used in LC studies, validated by ATLAS and CMS
- Version 2.8.1 (latest) available in the LCG stack[1]

[1]: with some trick ... (see hands-on)

## Generating LHE files with Whizard

- One nice thing of Whizard is that it can generate output in many formats
  - Ascii, HepMC, HepEvt, LCIO, LHE, StdHEP, ...
- Whizard configuration files use SINDARIN language and have extesion .sin
- Examples of configuration files useful for FCCee can be found here (dimuon production)

```
...
simulate (zmumu) { $sample = "z_mumu" sample_format = lhef }
```

## Processing Whizard LHE with Pythia8

- Just use the Pythia_LHEinput.cmd configuration file
  - With the proper path to the file ...

```
$ fccrun FCCSW/Sim/SimDelphesInterface/options/PythiaDelphes_config.py \
        --Filename FCCSW/Generation/data/Pythia_LHEinput.cmd \
        --DelphesCard FCCSW/Sim/SimDelphesInterface/data/FCChh_DelphesCard_Baseline_v01.tcl
```

## Importance of Common Data Formats

- Large effort made before LHC to find formats covering the needs
- Several format available
- Wrappers can be provided for missing standard formats
    - E.g. StdHEP
- Key point for interoperability

## Summary

- Generators can interfaced in two ways
- Pythia8 good example of Level 1 interoperability
- Whizard interfaced via LHE
- Using common shared formats greatly facilitates interoperation