Generators tutorial

- **We'll be using Pythia 8, WHIZARD and KKMCee to generate events.**
- **Following the FCC-ee tutorial at https://hep-fcc.github.io/fcc-tutorials/master/fast-sim-and-analysis/README.html**

[indico]

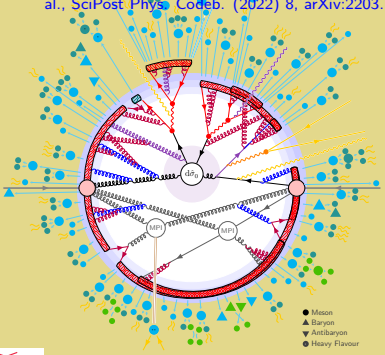
Patrick Koppenburg
[@pkoppenburg.bsky.social] [@pkoppenburg] [patrick.koppenburg@nikhef.nl]

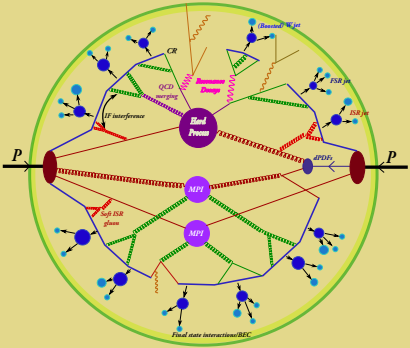
Nikhef

Event generators

Event generators are software libraries that generate simulated high-energy particle physics events. They randomly generate events as those produced in particle accelerators, collider experiments or the early universe. [wikipedia]

Pythia 8 Your do-it-all generator you might know from LHC. [pythia.org] [Bierlich et al., SciPost Phys. Codeb. (2022) 8, arXiv:2203.11601]





Event generators

Event generators are software libraries that generate simulated high-energy particle physics events. They randomly generate events as those produced in particle accelerators, collider experiments or the early universe. [wikipedia]

Pythia 8 Your do-it-all generator you might know from LHC. [pythia.org] [Bierlich et al., SciPost Phys. Codeb. (2022) 8, arXiv:2203.11601]

WHIZARD is a program system designed for the efficient calculation of multi-particle scattering cross sections and simulated event samples. [whizard.hepforge.org] [Kilian, Ohl, Reuter, arXiv:0708.4233]

KKMCee is an adaptation of the KKMC Monte Carlo generator (the latest version of the Koral generators) to the case of e^+e^- colliders. [Jadach, Ward, Was, Yost, Siodmok, Comput. Phys. Commun. 283 (2023) 108556, arXiv:2204.11949]

Herwig is another historical LEP generator providing a different approach to hadronization wrt Pythia. [webpage] [Corcella et al., JHEP 01 (2001) 010, arXiv:hep-ph/0011363]

Event generators

Event generators are software libraries that generate simulated high-energy particle physics events. They randomly generate events as those produced in particle accelerators, collider experiments or the early universe. [wikipedia]

Specialised event generators for luminosity determinations with Bhabha scattering

`BHLUMI` is a Monte Carlo generator of Bhabha events used at LEP for luminosity studies. [Jadach, Comput.Phys.Commun. 70 (1992) 305]

`BabaYaga` is a Monte Carlo generator of two-photons events used at LEP.

[homepage] [Balossini, Carloni Calame, Montagna, Piccinini, NPB 758 (2006) 227-253 , arXiv:hep-ph/0607181]

key4hep

key4hep is plug-and-play and thus so are generators. . . to some extent.

There are two categories

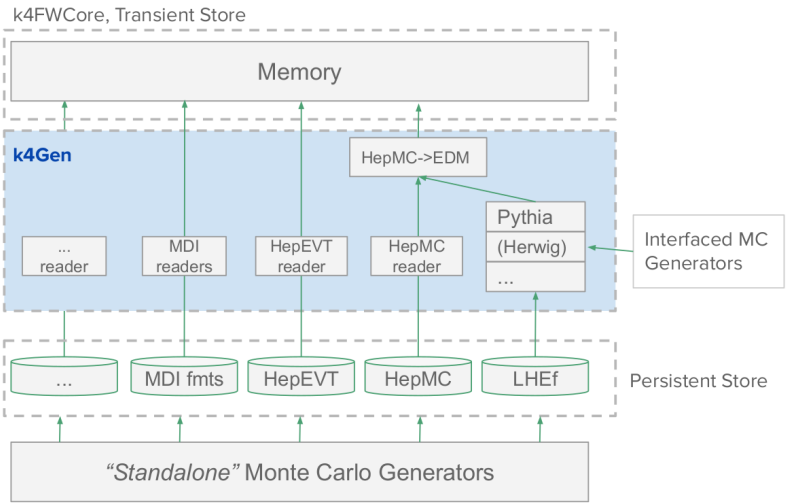
Fully integrated generator: Pythia 8,

Available as stand-alone programs: WHIZARD, KKMCee, Herwig, BHLUMI, BabaYaga

- This means one can get them from the software stack (they are there) but one needs to generate output in HepMC3 or EDM4hep formats and read this is for the next steps.
- Interfacing in progress (or potentially already done)
  - ✗ EvtGen needs work (afaik)

✔ This implies that anything else that can generate compatible output can be used

key4hep


k4FWCore, Transient Store
Memory
k4Gen
HepMC->EDM
Pythia
(Herwig)
...
Interfaced MC Generators
... reader
MDI readers
HepEVT reader
HepMC reader
... MDI fmts HepEVT HepMC LHEf
Persistent Store
"Standalone" Monte Carlo Generators

List of MC Generators currently available in key4hep

From spack upstream

form 4.2.1	vbfnlo 2.7.1	collier 1.2.5
crmc 1.6.0	syscalc 1.1.7	gosam 2.0
photos 3.64	thepeg 2.2.1	herwig3 7.2.1
qd 2.3.13	chaplin 1.2	madgraph5amc 2.8.1
evtgen 2.1.0	heputils 1.3.2	openloops 2.1.2
lhapdf 6.3.0	looptools 2.15	pythia8 8.306
mcutils 1.3.5	njet 2.1.1	recola 2.2.3
qgraf 3.4.2	pythia6 6.4.28	yoda 1.9.0
rivet 3.1.4	sherpa 2.2.11	
hepmc 2.06.11	hepmc3 3.2.4	

From key4hep-spack

guinea-pig 1.2.2rc
whizard 3.0.1
KKMCee 5.00.01
BHLUMI 4.04-linuxLHE
Babayaga fcc-1.0.0

Generators tutorial

We learn:

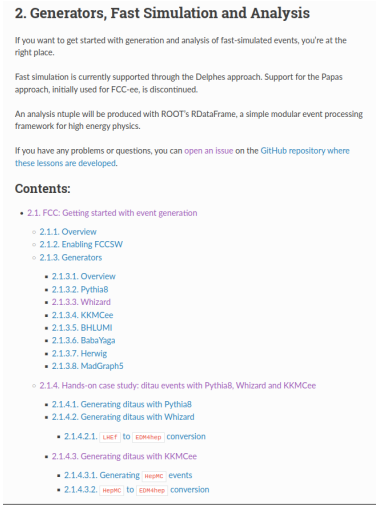
- How to configure and run a MC generator
- How to handle different output files formats
- How to look at the output files
- How to make simple checks

Using

BASH to handle files, running programs, ...

GAUDI adjusting configuration files to needs, ...

ROOT looking at the result (there's also `FCCAnalysis`)



2. Generators, Fast Simulation and Analysis

If you want to get started with generation and analysis of fast-simulated events, you're at the right place.

Fast simulation is currently supported through the Delphes approach. Support for the Papas approach, initially used for FCC-ee, is discontinued.

An analysis ntuple will be produced with ROOT's RDataFrame, a simple modular event processing framework for high energy physics.

If you have any problems or questions, you can open an issue on the GitHub repository where these lessons are developed.

Contents:

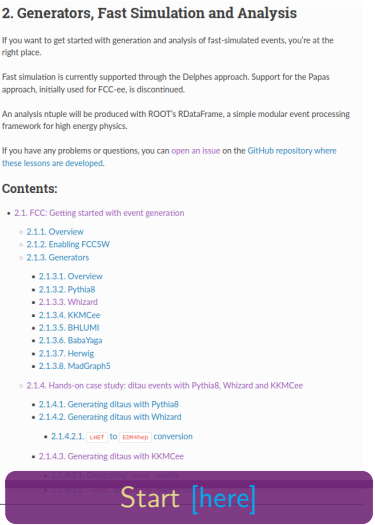
Generators tutorial

- Pythia can do the full generation in one step
- WHIZARD needs Pythia (or something) to do decays
- KKMCee will need a format conversion step

➜ Which are the three use-cases that should cover all generators

Caveats:

1. do not pipe `k4run` into anything. It doesn't like it.
2. There are some minor mistakes in the proposed solutions.



Solutions

Pythia configuration

```
"""
Pythia8, integrated in the FCCSW framework.

Generates according to a pythia .cmd file and saves them in fcc edm format.
"""

import os
from GaudiKernel import SystemOfUnits as units
from Gaudi.Configuration import *

from Configurables import ApplicationMgr
ApplicationMgr().EvtSel = 'NONE'
ApplicationMgr().EvtMax = 2
ApplicationMgr().OutputLevel = INFO
ApplicationMgr().ExtSvc +=["RndmGenSvc"]

#### Data service
from Configurables import k4DataSvc
podioevent = k4DataSvc("EventDataSvc")
ApplicationMgr().ExtSvc += [podioevent]

from Configurables import GaussSmearVertex
smeartool = GaussSmearVertex()
smeartool.xVertexSigma =   0.5*units.mm
smeartool.yVertexSigma =   0.5*units.mm
smeartool.zVertexSigma =  40.0*units.mm
smeartool.tVertexSigma = 180.0*units.picosecond

from Configurables import PythiaInterface
pythia8gentool = PythiaInterface()
### Example of pythia configuration file to generate events
# take from $K4GEN if defined, locally if not
path_to_pythiafile = os.environ.get("K4GEN", "")
pythiafilename = "Pythia_standard.cmd"
pythiafile = os.path.join(path_to_pythiafile, pythiafilename)
# Example of pythia configuration file to read LH event file
#pythiafile="options/Pythia_LHEinput.cmd"
pythia8gentool.pythiacard = pythiafile
pythia8gentool.doEvtGenDecays = False
pythia8gentool.printPythiaStatistics = True
pythia8gentool.pythiaExtraSettings = [""]

from Configurables import GenAlg
pythia8gen = GenAlg("Pythia8")
pythia8gen.SignalProvider = pythia8gentool
pythia8gen.VertexSmearingTool = smeartool
pythia8gen.hepmc.Path = "hepmc"
ApplicationMgr().TopAlg += [pythia8gen]

### Reads an HepMC::GenEvent from the data service and writes a collection of EDM Particle
from Configurables import HepMCToEDMConverter
hepmc_converter = HepMCToEDMConverter()
hepmc_converter.hepmc.Path="hepmc"
hepmc_converter.hepmcStatusList = [] # convert particles with all statuses
hepmc_converter.GenParticles.Path="GenParticles"
ApplicationMgr().TopAlg += [hepmc_converter]

### Filters generated particles
# accept is a list of particle statuses that should be accepted
from Configurables import GenParticleFilter
genfilter = GenParticleFilter("StableParticles")
genfilter.accept = [1]
genfilter.GenParticles.Path = "GenParticles"
genfilter.GenParticlesFiltered.Path = "GenParticlesStable"
ApplicationMgr().TopAlg += [genfilter]

from Configurables import PodioOutput
out = PodioOutput("out")
out.outputCommands = ["keep *"]
ApplicationMgr().TopAlg += [out]
```

[B]

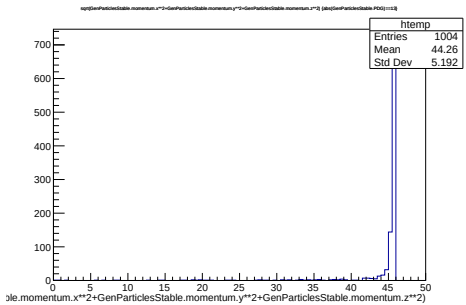
Pythia configuration

$Z \to \mu^+\mu^-$

```
! This file contains commands to be read in for a Pythia8 run.
! Lines not beginning with a letter or digit are comments.
! Names are case-insensitive  -  but spellings-sensitive!
! The settings here are illustrative, not always physics-motivated.

! 1) Settings used in the main program.
Random:setSeed = on
Main:timesAllowErrors = 5          ! how many aborts before run stops
Stat:showProcessLevel = on

! 2) Settings related to output in init(), next() and stat().
Init:showChangedSettings = on      ! list changed settings
Init:showChangedParticleData = off ! list changed particle data
Next:numberCount = 100             ! print message every n events
Next:numberShowInfo = 1            ! print event information n times
Next:numberShowProcess = 1         ! print process record n times
Next:numberShowEvent = 0           ! print event record n times

! 3) Beam parameter settings. Values below agree with default ones.
Beams:idA = 11                     ! first beam, e = 2212, pbar = -2212
Beams:idB = -11                    ! second beam, e = 2212, pbar = -2212

! 4) Hard process : Z->qqbar at Ecm=91 GeV
Beams:eCM = 91.188  ! CM energy of collision

WeakSingleBoson:ffbar2gmZ = on
23:onMode = off
23:onIfAny = 13
22:onMode = off
22:onIfAny = 13
```

[B]

```
k4run pythia.py -n 500
--out.filename p8_mumu_500.e4h.root
--Pythia8.PythiaInterface.pythiacard
p8_ee_Zmumu_ecm91.cmd
```



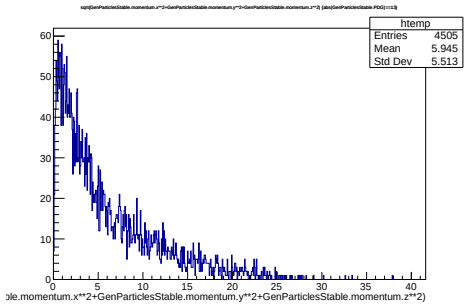
μ momentum

Pythia configuration

$Z \rightarrow b\bar{b}$

```
! This file contains commands to be read in for a Pythia8 run.
! Lines not beginning with a letter or digit are comments.
! Names are case-insensitive  -  but spellings-sensitive!
! The settings here are illustrative, not always physics-motivated.

! 1) Settings used in the main program.
Random:setSeed = on
Main:timesAllowErrors = 5          ! how many aborts before run stops
Stat:showProcessLevel = on

! 2) Settings related to output in init(), next() and stat().
Init:showChangedSettings = on      ! list changed settings
Init:showChangedParticleData = off ! list changed particle data
Next:numberCount = 100             ! print message every n events
Next:numberShowInfo = 1            ! print event information n times
Next:numberShowProcess = 1         ! print process record n times
Next:numberShowEvent = 0           ! print event record n times

! 3) Beam parameter settings. Values below agree with default ones.
Beams:idA = 11                     ! first beam, e = 2212, pbar = -2212
Beams:idB = -11                    ! second beam, e = 2212, pbar = -2212

! 4) Hard process : Z->qqbar at Ecm=91 GeV
Beams:eCM = 91.188  ! CM energy of collision

WeakSingleBoson:ffbar2gmZ = on
23:onMode = off
23:onIfAny = 5
22:onMode = off
22:onIfAny = 5
```

```
k4run pythia.py -n 10000
--out.filename p8_bb_10000.e4h.root
--Pythia8.PythiaInterface.pythiacard
p8_ee_Zbb_ecm91.cmd
```



μ momentum

$Z \to \tau^+\tau^-$ with Pythia

```
import os
from GaudiKernel import SystemOfUnits as units
from Gaudi.Configuration import *

from Configurables import ApplicationMgr
ApplicationMgr().EvtSel = 'NONE'
ApplicationMgr().EvtMax = 2
ApplicationMgr().OutputLevel = INFO
ApplicationMgr().ExtSvc +=["RndmGenSvc"]

#### Data service
from Configurables import k4DataSvc
podioevent = k4DataSvc("EventDataSvc")
ApplicationMgr().ExtSvc += [podioevent]

from Configurables import GaussSmearVertex
smeartool = GaussSmearVertex()
smeartool.xVertexSigma =   0.5*units.mm
smeartool.yVertexSigma =   0.5*units.mm
smeartool.zVertexSigma =  40.0*units.mm
smeartool.tVertexSigma = 180.0*units.picosecond

from Configurables import PythiaInterface
pythia8gentool = PythiaInterface()
### Example of pythia configuration file to generate events
# take from $K4GEN if defined, locally if not
path_to_pythiafile = os.environ.get("K4GEN", "")
pythiafilename = "Pythia_standard.cmd"
pythiafile = os.path.join(path_to_pythiafile, pythiafilename)
# Example of pythia configuration file to read LH event file
#pythiafile="options/Pythia_LHEinput.cmd"
pythia8gentool.pythiacard = pythiafile
pythia8gentool.doEvtGenDecays = False
pythia8gentool.printPythiaStatistics = True
pythia8gentool.pythiaExtraSettings = [""]
```

```
from Configurables import GenAlg
pythia8gen = GenAlg("Pythia8")
pythia8gen.SignalProvider = pythia8gentool
pythia8gen.VertexSmearingTool = smeartool
pythia8gen.hepmc.Path = "hepmc"
ApplicationMgr().TopAlg += [pythia8gen]

### Reads an HepMC::GenEvent from the data service and writes a collection of
from Configurables import HepMCToEDMConverter
hepmc_converter = HepMCToEDMConverter()
hepmc_converter.hepmc.Path="hepmc"
hepmc_converter.hepmcStatusList = [] # convert particles with all statuses
hepmc_converter.GenParticles.Path="GenParticles"
ApplicationMgr().TopAlg += [hepmc_converter]

### Filters generated particles
# accept is a list of particle statuses that should be accepted
from Configurables import GenParticleFilter
genfilter = GenParticleFilter("StableParticles")
genfilter.accept = [1]
genfilter.GenParticles.Path = "GenParticles"
genfilter.GenParticlesFiltered.Path = "GenParticlesStable"
ApplicationMgr().TopAlg += [genfilter]

from Configurables import PodioOutput
out = PodioOutput("out")
out.outputCommands = ["keep *"]
ApplicationMgr().TopAlg += [out]
```

```
k4run config/pythia.py -n 10000 --out.filename
gen/p8_tautau_ecm91/events_1.root
--Pythia8.PythiaInterface.pythiacard
cards/p8_ee_Ztautau_mumu_ecm91.cmd
```

$Z \to \tau^+\tau^-$ WITH WHIZARD

```
# Z_tautau.sin
# Example of simple generation of ditau events at Z

model = SM

# Processes
process ztautau = e1, E1 => e3, E3
beams = e1, E1 => isr

compile

# Center of mass energy
sqrts = 91.188 GeV
# sqrts = 91.2 GeV

# Calculate
integrate (ztautau)

# Set the number of events. Alternatively we can set the luminosity in femtobarn
# luminosity = 1
n_events = 10000

# Define title etc. as global variables, that will be used by PLOT
$description =
  "A WHIZARD 3.0 Example.
   Z -> tautau @ 91.2 events for FCC ee."
$y_label = "$N_{\textrm{events}}$"

# simulate (ztautau) { $sample = "z_tautau" sample_format = hepmc }
# simulate (ztautau) { $sample = "z_tautau" sample_format = stdhep }
simulate (ztautau) { $sample = "z_tautau" sample_format = lhef }

?ps_taudec_active = true
```

```
whizard ../Z_tautau.sin
```

```
!
! This file contains commands to be read in for a Pythia8 run.
! Lines not beginning with a letter or digit are comments.
! Names are case-insensitive  -  but spellings-sensitive!
! Adjusted from Pythia example: main42.cmnd

! 1) Settings that will be used in a main program.
Main:numberOfEvents = 100          ! number of events to generate
Main:timesAllowErrors = 10         ! abort run after this many flawed events

! 2) Settings related to output in init(), next() and stat() functions.
Init:showChangedSettings = on      ! list changed settings
Init:showAllSettings = off         ! list all settings
Init:showChangedParticleData = on  ! list changed particle data
Init:showAllParticleData = off     ! list all particle data
Next:numberCount = 1000            ! print message every n events
Next:numberShowLHA = 10            ! print LHA information n times
Next:numberShowInfo = 10           ! print event information n times
Next:numberShowProcess = 10        ! print process record n times
Next:numberShowEvent = 10          ! print event record n times
Stat:showPartonLevel = off         ! additional statistics on MPI
Stat:showProcessLevel = off        ! additional statistics on MPI

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

! Tau decays to mu nu_tau nu_mu (from pythia8)
15:onMode = off
15:onIfAny = 14
```

```
k4run ../config/pythia.py -n 10000 --out.filename
../gen/wz_tautau_ecm91/events_1.root
--Pythia8.PythiaInterface.pythiacard
../cards/p8_lhereader.cmd | tee
wz_ee_Ztautau_mumu_ecm91.log
```

$Z \to \tau^+\tau^-$ WITH WHIZARD

```
# Z_tautau.sin
# Example of simple generation of ditau events at Z

model = SM

# Processes
process ztautau = e1, E1 => e3, E3
beams = e1, E1 => isr

compile

# Center of mass energy
sqrts = 91.188 GeV
# sqrts = 91.2 GeV

# Calculate
integrate (ztautau)

# Set the number of [illegible] alternatively [illegible] in fautobarn
# luminosity = 1
n_events = 10000

# Define title etc. as global variables, that will be used by PLOT
$description =
  "A WHIZARD 3.0 Example.
   Z -> tautau @ 91.2 events for FCC ee."
$y_label = "$N_{\textrm{events}}$"

# simulate (ztautau) { $sample = "z_tautau" sample_format = hepmc }
# simulate (ztautau) { $sample = "z_tautau" sample_format = stdhep }
simulate (ztautau) { $sample = "z_tautau" sample_format = lhef }

?ps_taudec_active = true
```

Make sure it's the right number of events.
Or PYTHIA will issue an error.

```
whizard ../Z_tautau.sin
```

```
!
! This file contains commands to be read in for a Pythia8 run.
! Lines not beginning with a letter or digit are comments.
! Names are case-insensitive  - but spellings-sensitive!
! Adjusted from Pythia example: main42.cmnd

! 1) Settings that will be used in a main program.
Main:numberOfEvents = 100          ! number of events to generate
Main:timesAllowErrors = 10         ! abort run after this many flawed events

! 2) Settings related to output in init(), next() and stat() functions.
Init:showChangedSettings = on      ! list changed settings
Init:showAllSettings = off         ! list all settings
Init:s[illegible]                  [illegible]cle data
Init:s[illegible]                  cle data
Next:numberCount = [illegible]     ! print message every n events
Next:numberShow[illegible]         information n times
Next:numberShowInfo = 10           ! print event information n times
Next:numberShowProcess = 10        ! print process record n times
Next:numberShowEvent = 10          ! print event record n times
Stat:showPartonLevel = on          ! additional statistics on MPI
Stat:showProcessLevel = on         ! additional statistics on MPI

! 4) Read-in Les Houches event file - alternative beam and process selection.
Beams:frameType = 4                ! read info from a LHEF
Beams:LHEF = z_tautau.lhe          ! the LHEF to read from

! Tau decays to mu nu_tau nu_mu (from pythia8)
15:onMode = off
15:onIfAny = 14
```

Beware of the location of the input file.

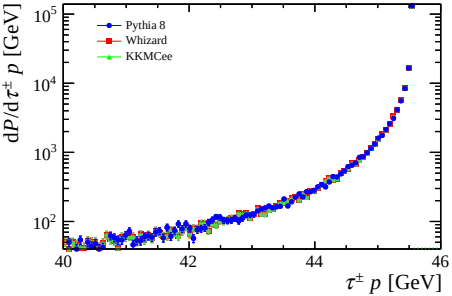
```
k4run ../config/pythia.py -n 10000 --out.filename
../gen/wz_tautau_ecm91/events_1.root
--Pythia8.PythiaInterface.pythiacard
../cards/p8_lhereader.cmd | tee
wz_ee_Ztautau_mumu_ecm91.log
```

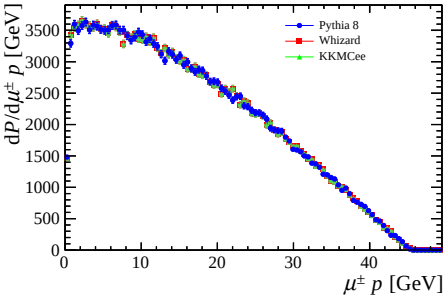
$Z \to \tau^+\tau^-$ WITH KKMCee

```
KKMCee -f Tau -e 91.2 -n 10000 -o
kk_ee_Zautau_mumu_10000.hepmc -t 2002

k4run config/hepmc2edm.py -n 10000
--GenAlg.HepMCFileReader.Filename
kk/kk_tautau_10000.hepmc --out.filename
gen/kk_tautau_ecm91/events_1.root
```

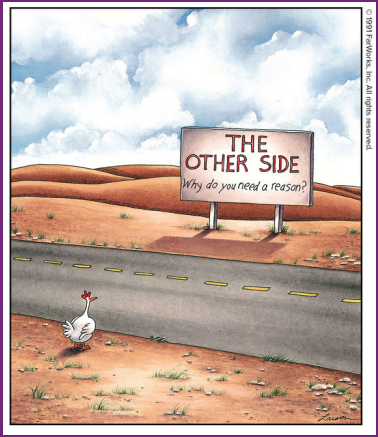
```
from Gaudi.Configuration import *
from GaudiKernel import SystemOfUnits as units

from Configurables import ApplicationMgr
ApplicationMgr(EvtSel='NONE',EvtMax=1,OutputLevel=INFO)

from Configurables import k4DataSvc
podioevent = k4DataSvc("EventDataSvc")
ApplicationMgr().ExtSvc += [podioevent]

from Configurables import HepMCFileReader
hepmcreader = HepMCFileReader()

from Configurables import GenAlg
reader = GenAlg()
reader.SignalProvider = hepmcreader
reader.hepmc.Path = "hepmc"
ApplicationMgr().TopAlg += [reader]

from Configurables import HepMCToEDMConverter
hepmc_converter = HepMCToEDMConverter()
hepmc_converter.hepmc.Path="hepmc"
hepmc_converter.GenParticles.Path = "GenParticles"
ApplicationMgr().TopAlg += [hepmc_converter]

### Filters generated particles
# accept is a list of particle statuses that should be accepted
from Configurables import GenParticleFilter
genfilter = GenParticleFilter("StableParticles")
genfilter.accept = [1]
genfilter.GenParticles.Path = "GenParticles"
genfilter.GenParticlesFiltered.Path = "GenParticlesStable"
ApplicationMgr().TopAlg += [genfilter]

from Configurables import PodioOutput
out = PodioOutput("out", filename = "hepmc2edm_output.root")
out.outputCommands = ["keep *"]
ApplicationMgr().TopAlg += [out]
```

$Z \to \tau^+\tau^-$ PLOTS





Plots of `sqrt(GenParticles.momentum.x**2 + GenParticles.momentum.y**2 + GenParticles.momentum.z**2)` with `abs(GenParticles.PDG)==15 && GenParticles.generatorStatus==2` (or 13)


THE
OTHER SIDE
Why do you need a reason?
© 1997 FarWorks, Inc. All rights reserved.

Backup

MC generators

FCC Software HandsOn tutorial

October 20, 2022
G Ganis, CERN-EP

MC Generators

See also W. Kilian's, ECFA kick-off, June 2021

Needs

- High energy ($\sqrt{s}$ > hZ threshold) e+e- generators
- Generators at Z peak, WW
- Heavy Flavour decays, including taus

Examples of heavily used codes for LC

- Whizard, MadGraph5_aMC, PhysSim, Pythia6, ...

Areas of work (not exhaustive)

- Recovery of LEP generators, but still state of art for Z peak, WW
  - KKMC family, BHLUMI, BHWIDE, Babayaga, … interfacing work ~~in progress~~ — mostly done
- Hadronization "tunes" for e+e- (Pythia, Herwig, …) — partly done
  - Eg. cannot import Pythia6 tune (from LEP) to Pythia8
- Interfaces with up-to-date decay codes (EvtGen, …) — work needed

Monte Carlo Generators in key4hep

- A Monte Carlo generator is a package
- Key4hep includes already many generators as packages
  - Initial list derived from LCG stacks, so sort of LHC oriented
  - But several e+e- additions available: Whizard, KKMCee, BabaYaga, BHLUMI, …
    - Including wrappers for better user experience
- What does it mean “adding a generator to key4hep”?
  - Required information for inclusion in the package manager
    - Source location, minimal documentation on how to build and required dependencies, default configuration files, tests, …
  - Key4hep infrastructure will
    - Build in shared installation mode
    - Run built-in tests, if any
    - Install in distributed shared file system

List of MC Generators currently available in key4hep

From spack upstream

form 4.2.1	vbfnlo 2.7.1	collier 1.2.5
crmc 1.6.0	syscalc 1.1.7	gosam 2.0
photos 3.64	thepeg 2.2.1	herwig3 7.2.1
qd 2.3.13	chaplin 1.2	madgraph5amc 2.8.1
evtgen 2.1.0	heputils 1.3.2	openloops 2.1.2
lhapdf 6.3.0	looptools 2.15	pythia8 8.306
mcutils 1.3.5	njet 2.1.1	recola 2.2.3
qgraf 3.4.2	pythia6 6.4.28	yoda 1.9.0
rivet 3.1.4	sherpa 2.2.11	
hepmc 2.06.11	hepmc3 3.2.4	

From key4hep-spack

guinea-pig 1.2.2rc
whizard 3.0.1
KKMCee 5.00.01
BHLUMI 4.04-linuxLHE
Babayaga fcc-1.0.0

Levels of interoperability

- Level 0 - *Common Data Formats*
  - Maximal interoperability, even on different hardware
- Level 1 - *Callable Interfaces*
  - Defined for one or more programming languages
  - Implementation quality of interfaced components important
  - Required to define plugins
- Level 2 - *Introspection Capabilities*
  - Software elements to facilitate the interaction of objects in a generic manner such as Dictionaries and Scripting interfaces
  - Language bindings, e.g. PyROOT
- Level 3 - *Component Level*
  - Software components are part of a *common framework*, optimal interplay
  - Common configuration, log and error reporting, plug-in management, ...

Common Data Formats

Recommended

- HEPMC
  - Generic framework for MC generator event record encoding and manipulation
  - Version 3 complete re-write of previous version and generally adopted
  - Provide tools for managing other formats, e.g. LHEf files
- EDM4hep
  - Common multipurpose event data model
  - Contains relevant data structures

Accepted

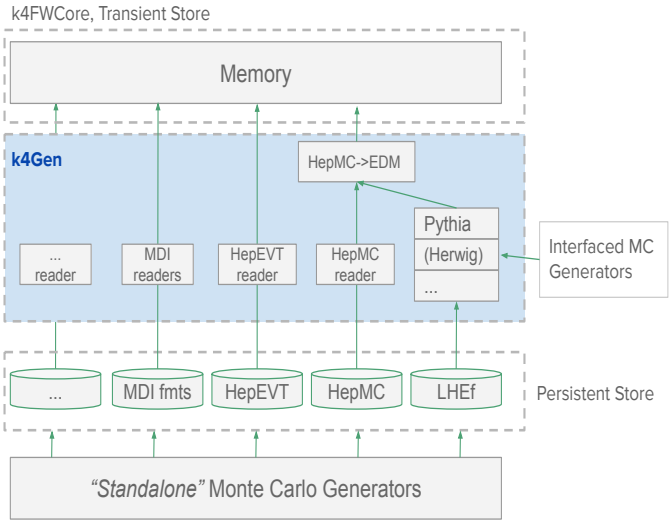
- LHEf
  - Format agreed for LHC, generally OK for e+e-
  - Some issues for FCC (see FCC presentation)
- HepEVT
  - Format used by old fortran generators
- Potentially any format completely documented

Managing interoperability through Gaudi

- Components tailored for specific tasks
  - Tools for inputting and managing MC outputs
- What is required
  - Readers for data formats
    - Currently available: HepEVT, HepMC, LHEf, ...
  - Level-1 interfaces for MC
    - Only currently available Pythia8
    - Place where to put an interface to Herwig

The repository is k4Gen, and includes also modules to perform actions on the MC events, such as vertex smearing, particle filters, or basic MC tools, such particle guns

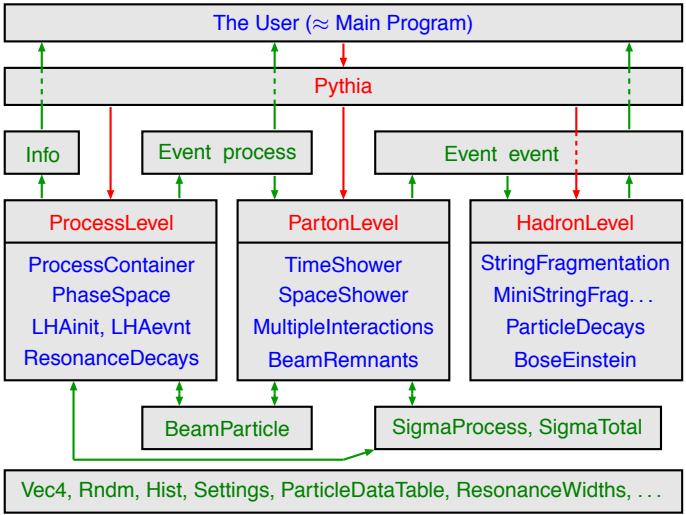
Managing interoperability



Hands-on

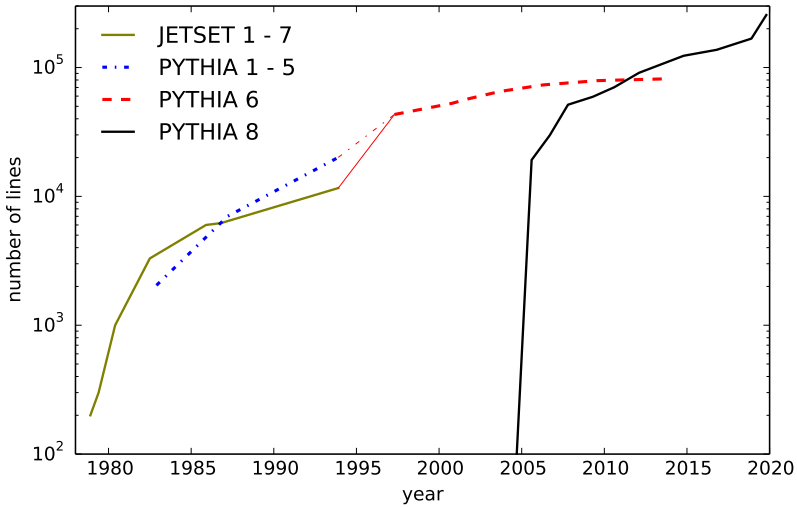
- Main page (added Whizard for LHEf)
- Steps
  - Generating ditaus with KKMCee
    - HepMC to EDM4hep conversion
  - Generating ditaus with Pythia8
  - Generating ditaus with Whizard
    - LHEf to EDM4hep conversion
  - Looking at the produced files: the MCParticle class
    - Creating flat ntuples with FCCAnalyses
    - Comparing distributions

Pythia

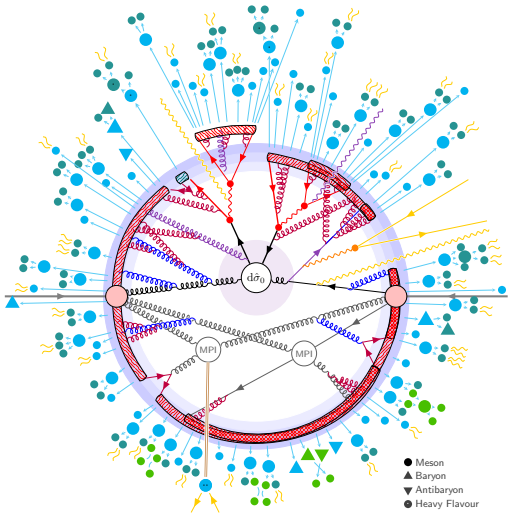

The User (≈ Main Program)
Pythia
Info
Event process
Event event
ProcessLevel
ProcessContainer
PhaseSpace
LHAinit, LHAevnt
ResonanceDecays
PartonLevel
TimeShower
SpaceShower
MultipleInteractions
BeamRemnants
HadronLevel
StringFragmentation
MiniStringFrag...
ParticleDecays
BoseEinstein
BeamParticle
SigmaProcess, SigmaTotal
Vec4, Rndm, Hist, Settings, ParticleDataTable, ResonanceWidths, ...

[B]

Pythia

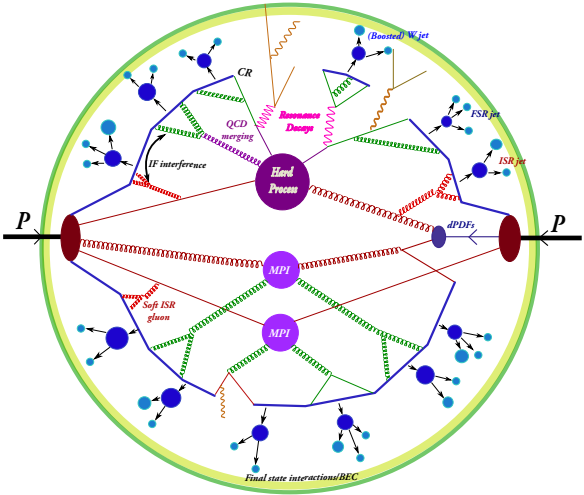

JETSET 1 - 7
PYTHIA 1 - 5
PYTHIA 6
PYTHIA 8
number of lines
year
10^2
10^3
10^4
10^5
1980
1985
1990
1995
2000
2005
2010
2015
2020

Pythia



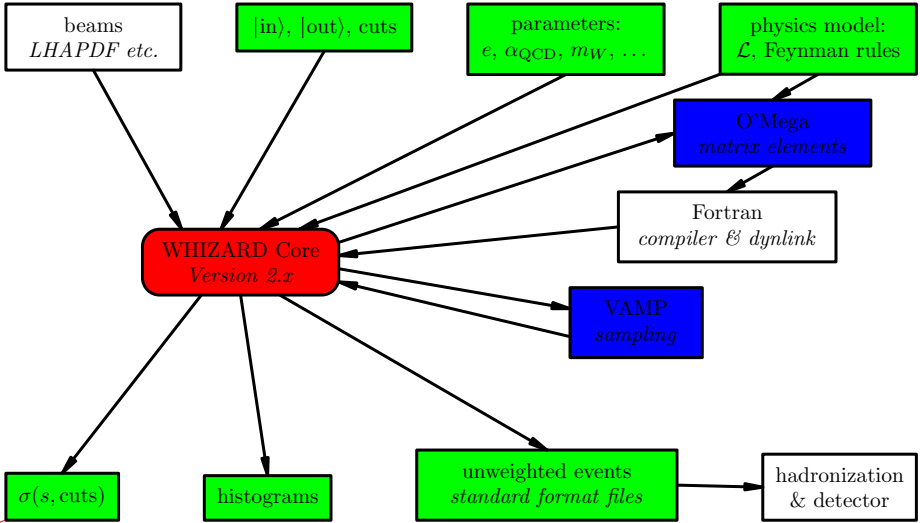
[B]

Pythia

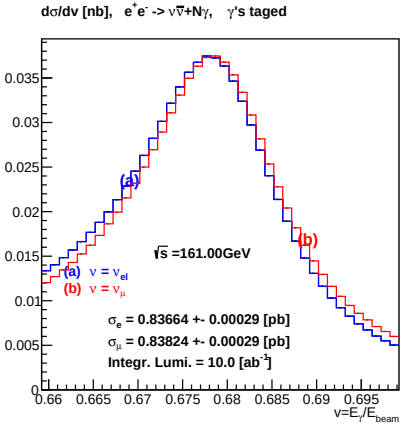

(Boosted) W jet
CR
Resonance Decays
FSR jet
QCD merging
ISR jet
IF interference
Hard Process
P
P
dPDFs
MPI
Soft ISR gluon
MPI
Final state interactions/BEC

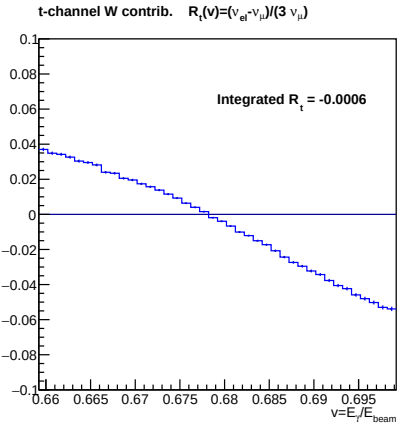
[B]

WHIZARD


beams
LHAPDF etc.
|in⟩, |out⟩, cuts
parameters:
e, α_{QCD}, m_W, ...
physics model:
$\mathcal{L}$, Feynman rules
O'Mega
matrix elements
Fortran
compiler & dynlink
WHIZARD Core
Version 2.x
VAMP
sampling
$\sigma(s, \text{cuts})$
histograms
unweighted events
standard format files
hadronization
& detector

KKMCee





$e^+e^- \to$ 2 fermions plus photons

[Bhabha, Proc.Roy.Soc.Lond.A 154 (1936) 195]

Bhabha scattering

To leading order, the spin-averaged differential cross section for $e^+e^- \to e^+e^-$ is

$$\frac{\mathrm{d}\sigma}{\mathrm{d}(\cos\theta)} = \frac{\pi\alpha^2}{s}\left(u^2\left(\frac{1}{s}+\frac{1}{t}\right)^2 + \left(\frac{t}{s}\right)^2 + \left(\frac{s}{t}\right)^2\right)$$

where the Mandelstam variables are

$$s = (k+p)^2 = (k'+p')^2 \simeq \quad 2k\cdot p \simeq \quad 2k'\cdot p'$$
$$s = (k-k')^2 = (p-p')^2 \simeq -2k\cdot k' \simeq -2p\cdot p'$$
$$s = (k-p')^2 = (p-k')^2 \simeq -2k\cdot p' \simeq -2k'\cdot p$$

and approximations are due to the relativistic limit [wikipedia]