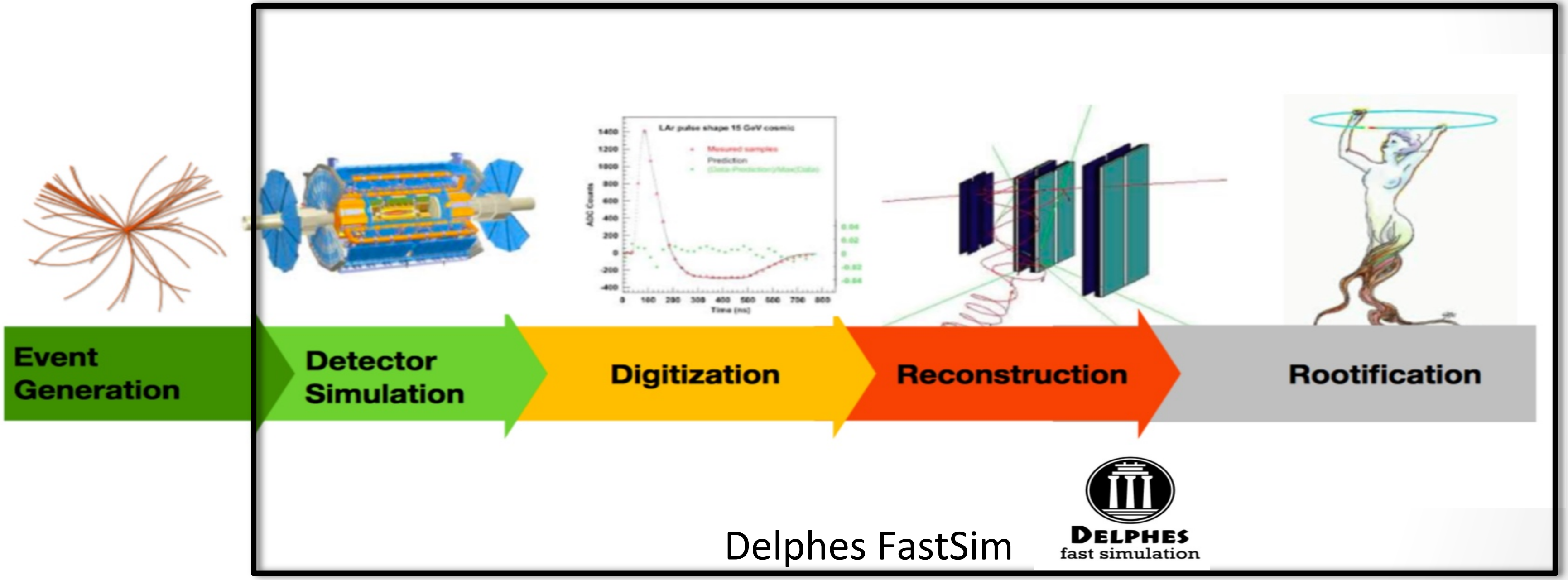
FCC Physics analysis framework

Clement Helsens, CERN-EP

Introduction/motivation

- I will review in this talk the analysis model used at FCC-hh and that is being reworked for a better usage at both FCC-hh and FCC-ee
- FCC-hh analysis framework (event generation to final plots) has been used to produce several billions of events and about 20 analyses
- How should the analysis model look like for FCC-ee?

Monte Carlo Event Generation



FCC-hh(ee) Framework

Fully integrated analysis chain
Links between various steps

1. GridPack producer
   - Makes MG5_aMC@NLO GridPacks, need to understand if other GEN can do it
2. LHE Producer
   - Produce LHE files on condor queues, either from step 1) or on the fly. Need a detailed review of the possibilities offered by other GEN (this has started as we have just seen)
3. FCCSW
   - Runs Pythia8 parton shower+hadronisation and Delphes with FCC detector
     http://fcc-physics-events.web.cern.ch/fcc-physics-events/Delphesevents_fccee_v01.php
     Also need to support later other parton showers
4. Analysis pre-selection
   - Python framework produces flat ROOT trees
5. Analysis final selection
   - Python framework for optimising analysis cut flows and producing
6. Final analyses (custom, but we have limit setting tool for instance)

Creates a database of LHE events

Creates a database of FCC events

Use the events in the database to produce analyses templates

Use the events produced at pre-selection to create stacked plots

FCC-hh(ee) Framework

1. GridPack producer
   - Makes MG5_aMC@NLO GridPacks, need to understand if other GEN can do it

2. LHE Producer
   - Produce LHE files on condor queues, either from step 1) or on the fly. Need a detailed review of the possibilities offered by other GEN (this has started as we have just seen)

3. FCCSW
   - Runs Pythia8 parton shower+hadronisation and Delphes with FCC detector
     http://fcc-physics-events.web.cern.ch/fcc-physics-events/Delphesevents_fccee_v01.php
     Also need to support later other parton showers

4. Analysis pre-selection
   - Python framework produces flat ROOT trees

In the tutorial
Tomorrow

5. Analysis final selection
   - Python framework for optimising analysis cut flows and producing

6. Final analyses (custom, but we have limit setting tool for instance)

Creates a database of LHE events

Creates a database of FCC events

Use the events in the database to produce analyses templates

Use the events produced at pre-selection to create stacked plots

2. Generation

- https://github.com/FCC-hh-framework/EventProducer

- Start from Madgrap gridpacks but no problem to use other gridpacks
  - Create LHE files and store them on eos
  - List of available samples and statistics for 100TeV and 27TeV (no LHE yet for FCC-ee)
    - http://fcc-physics-events.web.cern.ch/fcc-physics-events/LHEevents.php
    - http://fcc-physics-events.web.cern.ch/fcc-physics-events/LHEevents_helhc.php

- Can also register to the database your own LHE files, they just need to comply with simple formatting rules

- Fully adapted for FCC-ee if we have gridpacks, but we are investigating the various ee generators to understand what could be done

3. Simulation (FCCSW Delphes)

- https://github.com/FCC-hh-framework/EventProducer

- From the LHE files, create FCC EDM files for a given Delphes parametrisation
- Also possible to directly simulate events with Pythia8 (what we did for FCC-ee tutorial)

- List of available samples and statistics is available for FCC 100/27TeV :
  - http://fcc-physics-events.web.cern.ch/fcc-physics-events/Delphesevents_fcc_v02.php
  - http://fcc-physics-events.web.cern.ch/fcc-physics-events/Delphesevents_helhc_v01.php

- Also started for FCC-ee (90M events, used tomorrow for the tutorial)
  - http://fcc-physics-events.web.cern.ch/fcc-physics-events/Delphesevents_fccee_v01.php

4. Flat trees

- For producing flat trees specific to analysis for the moment still using heppy
  https://github.com/HEP-FCC/FCCAnalyses/

- Within heppy create an analysis directory that contains always the same files:

- analysis.py
  - defines
    - the list of modules to be run
    - the list of samples over which to run
      (the inputs file lists and cross sections etc… centrally defined and supported from step 2.

TreeProducer.py
__init__.py
analysis.py
selection.py

```
selectedComponents = [
                        p8_ee_ZH_ecm240,
                        p8_ee_ZZ_ecm240,
                        p8_ee_WW_ecm240
                        ]
```

```
from heppy.analyzers.Selector import Selector
sel_muons = cfg.Analyzer(
    Selector,
    'sel_muons',
    output = 'sel_muons',
    input_objects = 'muons',
    filter_func = lambda ptc: ptc.pt()>10
)

# select isolated muons
dressed_muons = cfg.Analyzer(
    Selector,
    'dressed_muons',
    output = 'dressed_muons',
    input_objects = 'sel_muons',
    filter_func = lambda ptc: ptc.iso.sumpt/ptc.pt()<0.4
)
```

4. Flat trees

- For producing flat trees specific to analysis for the moment still using heppy
  https://github.com/HEP-FCC/FCCAnalyses/

- Within heppy create an analysis directory that contains always the same files:

- analysis.py
  - defines
    - the list of modules to be run
    - the list of samples over which to run
      (the inputs file lists and cross sections etc… centrally defined and supported from step 2.

TreeProducer.py
__init__.py
analysis.py
selection.py

```
sequence = cfg.Sequence( [
    source,

    sel_electrons,
    sel_muons,
    sel_photons,

    dressed_electrons,
    dressed_muons,
    dressed_photons,

    jets_10,
    match_electron_jets,
    jets_noelectron,
    selected_lights,
    selected_bs,
    selected_cs,
    selected_taus,
    zeds,
    recoil,
    selection,
    reco_tree,
    ] )
```

4. Flat trees

- For producing flat trees specific to analysis for the moment still using heppy https://github.com/HEP-FCC/FCCAnalyses/

- Within heppy create an analysis directory that contains always the same files:

- selection.py:
  - defines
    - the list of pre-selections (optional and can also be done in the TreeProducer)

```
class Selection(Analyzer):

    def beginLoop(self, setup):
        super(Selection, self).beginLoop(setup)
        self.counters.addCounter('cut_flow')
        self.counters['cut_flow'].register('All events')
        self.counters['cut_flow'].register('At least one Z -> mu+ mu- candidates')

    def process(self, event):
        self.counters['cut_flow'].inc('All events')

        zeds = event.zeds

        #select events with at least one Z -> mu+ mu- candidates
        if (len(zeds) < 1):
           return False
        self.counters['cut_flow'].inc('At least one Z -> mu+ mu- candidates')
```

4. Flat trees

- For producing flat trees specific to analysis for the moment still using heppy https://github.com/HEP-FCC/FCCAnalyses/

- Within heppy create an analysis directory that contains always the same files:

- TreeProducer.py:
  - defines
    - The variables to be stored in the output file
    - Also can compute high level variables
    - Can also apply some cuts

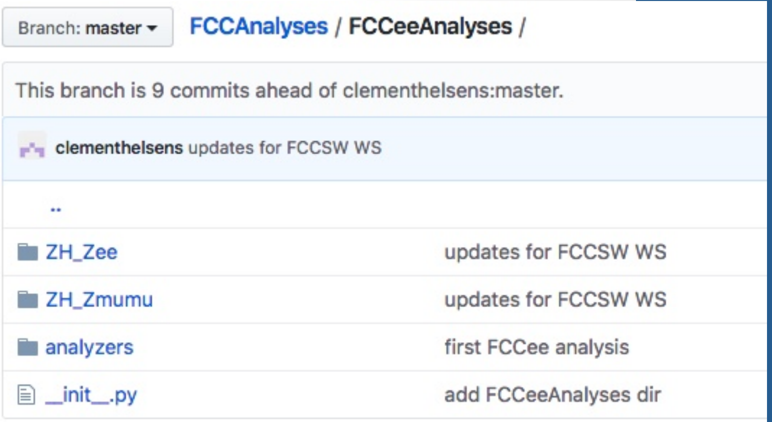
TreeProducer.py
__init__.py
analysis.py
selection.py

```
self.tree.var('nmu', float)
self.tree.var('nele_recoil', float)
self.tree.var('nph', float)
```

```
bookParticle(self.tree, 'recoil')
bookParticle(self.tree, 'zed')
bookParticle(self.tree, 'el1')
bookParticle(self.tree, 'el2')
bookMet(self.tree, 'met')
```

```
fillParticle(self.tree, 'zed', zed)
fillMet(self.tree, 'met', event.met)

fillLepton(self.tree, 'el1', zeds[0].legs[0])
fillLepton(self.tree, 'el2', zeds[0].legs[1])

self.tree.fill('nbjets' , len(event.selected_bs) )
self.tree.fill('ncjets' , len(event.selected_cs) )
self.tree.fill('ntaujets' , len(event.selected_taus) )
self.tree.fill('nljets' , len(event.selected_lights) )
```

4. Flat trees



- Analysis flow is fully reproducible
- Outputs must be stored on the eos FCC
  - /eos/experiments/fcc/ee/analyses/

- Fully adapted for FCC-ee, also have a test example with TDataFrame as backend
- For the tutorial tomorrow we will use the outputs of this step

5. Flat Tree analyser

- https://github.com/HEP-FCC/FlatTreeAnalyzer
  - From the files produced in 4
  - Plots and histograms for final analysis for different selections
  - Templates on github so that we can fully reproduce the results

- Selection based on variables available in output tree
- Possible to add new variables in the plots/output trees

```
### variable list
variables = {
    "ptz"      :{"name":"zed_pt"    ,"title":"p_{T}^{Z} [GeV]","bin":150,"xmin":0,"xmax":300},
    "mz"       :{"name":"zed_m"     ,"title":"m_{Z} [GeV]","bin":125,"xmin":0,"xmax":250},
    "ptmu_1"   :{"name":"mu1_pt"    ,"title":"p_{T}^{#mu, max} [GeV]","bin":150,"xmin":0,"xmax":150},
    "ptmu_2"   :{"name":"mu2_pt"    ,"title":"p_{T}^{#mu, min} [GeV]","bin":150,"xmin":0,"xmax":150},
    "mrecoil"  :{"name":"recoil_m"  ,"title":"m_{Recoil} [GeV]","bin":100,"xmin":50,"xmax":150},
    "ptrecoil" :{"name":"recoil_m"  ,"title":"p_{T}^{Recoil} [GeV]","bin":125,"xmin":0,"xmax":250},
    "met_pt"   :{"name":"met_pt"    ,"title":"met p_{T} [GeV]","bin":150,"xmin":0,"xmax":150},
    "met_sumet":{"name":"met_sumet" ,"title":"met sum E_{T} [GeV]","bin":125,"xmin":0,"xmax":250},

}
```

5. Flat Tree analyser

- https://github.com/HEP-FCC/FlatTreeAnalyzer
  - From the files produced in 4
  - Plots and histograms for final analysis for different selections
  - Templates on github so that we can fully reproduce the results

- Defines the signal(s) and backgrounds (the name of the dataset is the same as before

```
signal_groups = collections.OrderedDict()
signal_groups['ZH'] = ['p8_ee_ZH_ecm240']

background_groups = collections.OrderedDict()
background_groups['WW'] = ['p8_ee_WW_ecm240']
background_groups['ZZ'] = ['p8_ee_ZZ_ecm240']
```

5. Flat Tree analyser

- https://github.com/HEP-FCC/FlatTreeAnalyzer
  - From the files produced in 4
  - Plots and histograms for final analysis for different selections
  - Templates on github so that we can fully reproduce the results

- Defines the selections based on the variables available in the flat tree

```
# base pre-selections
selbase = 'recoil_m>10.'
selopt  = 'zed_pt<65. && zed_m>70. && zed_m<100. && mu1_pt<75.&& mu2_pt<50. && met_pt<50.'
selbb   = 'nbjets==2'
seltautau   = 'ntaujets==2'
selWhadWhad   = 'nljets+ncjets==4'
selWhadWlep   = 'nljets+ncjets==2 && ((nele==1 &&  nmu_recoil==0) || (nele==0 &&  nmu_recoil==1) )'
selWlepWlep   = '(nele==2 &&  nmu_recoil==0) || (nele==0 &&  nmu_recoil==2) ||  (nele==1 &&  nmu_recoil==1)'
selWW  = selWhadWhad + '||' + selWhadWlep + '||' + selWlepWlep
```

5. Flat Tree analyser

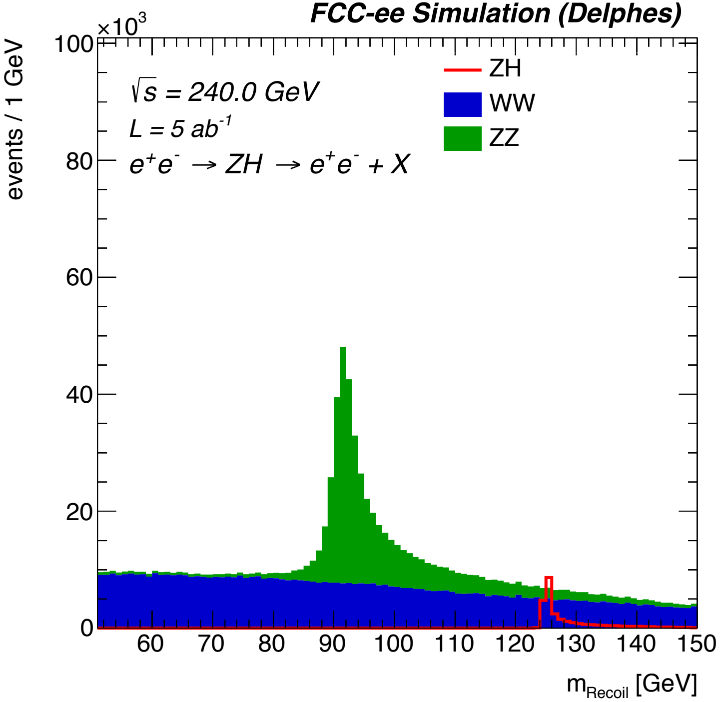
- https://github.com/HEP-FCC/FlatTreeAnalyzer
  - From the files produced in 4
  - Plots and histograms for final analysis for different selections
  - Templates on github so that we can fully reproduce the results

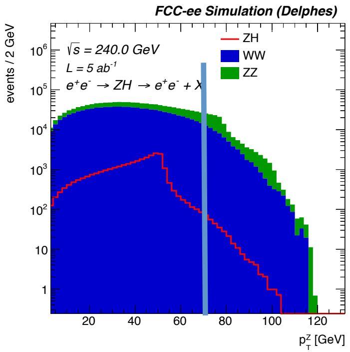
- Add the selections you want to run and run
- Plots and tree will have selN, with N the index of the selection 0,1,2,etc…

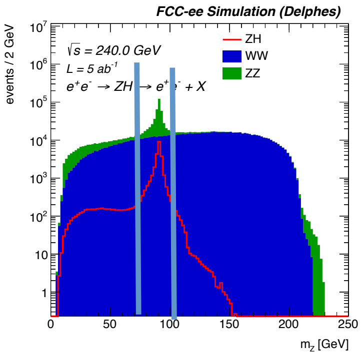
```
# add list of event selections here if needed...

selections = collections.OrderedDict()
selections['ZH'] = []
selections['ZH'].append(selbase)
selections['ZH'].append(selopt)
selections['ZH'].append(selbb)
selections['ZH'].append(seltautau)
selections['ZH'].append(selWhadWhad)
selections['ZH'].append(selWhadWlep)
selections['ZH'].append(selWlepWlep)
selections['ZH'].append(selWW)
```

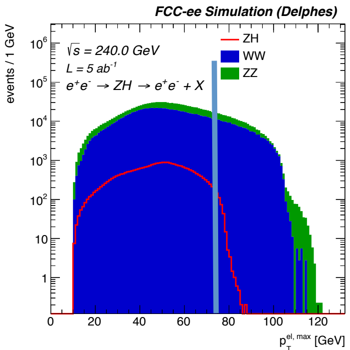
5. Flat Tree analyser

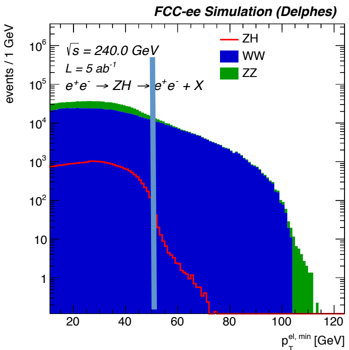
```
# base pre-selections
selbase = 'recoil_m>10.'
selopt  = 'zed_pt<65. && zed_m>70. && zed_m<100. && mu1_pt<75.&& mu2_pt<50. && met_pt<50.'
```

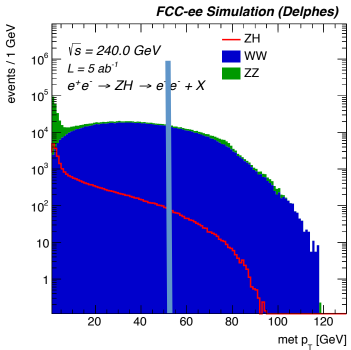






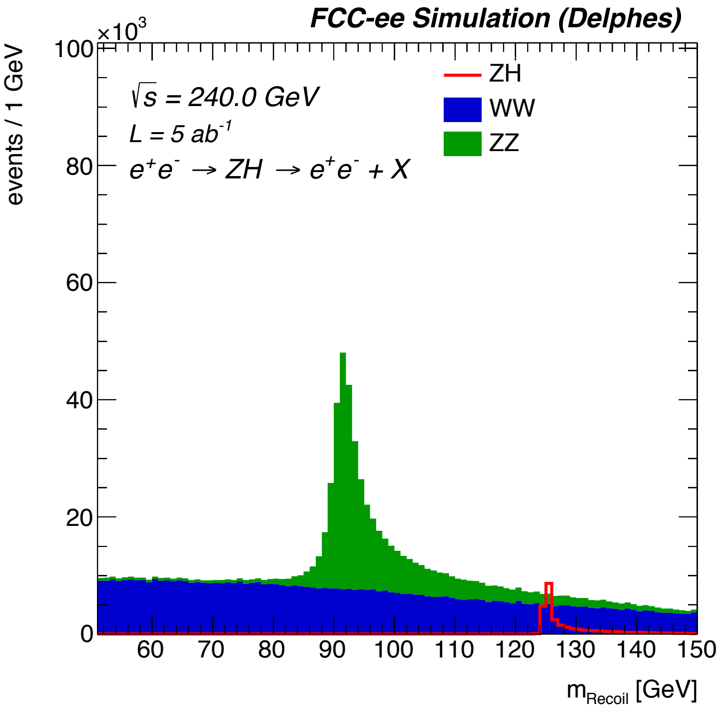


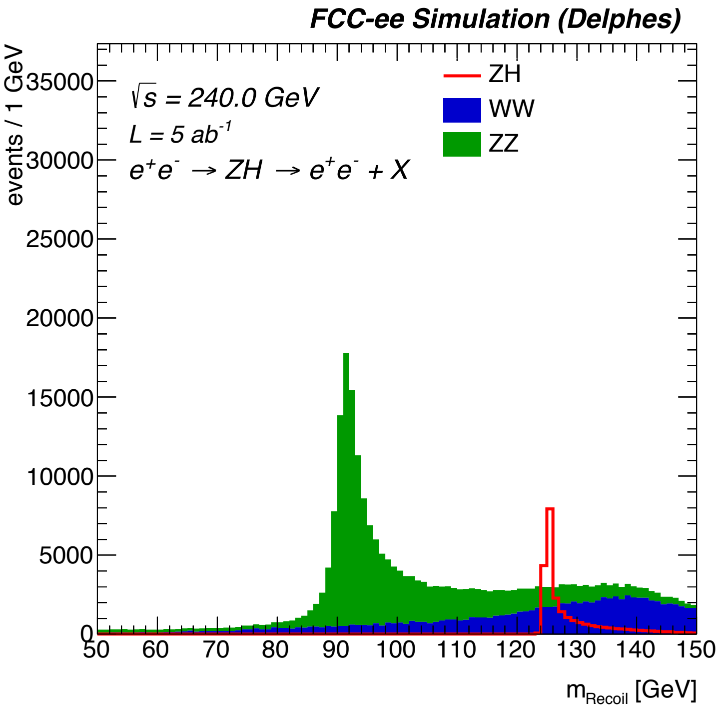




5. Flat Tree analyser

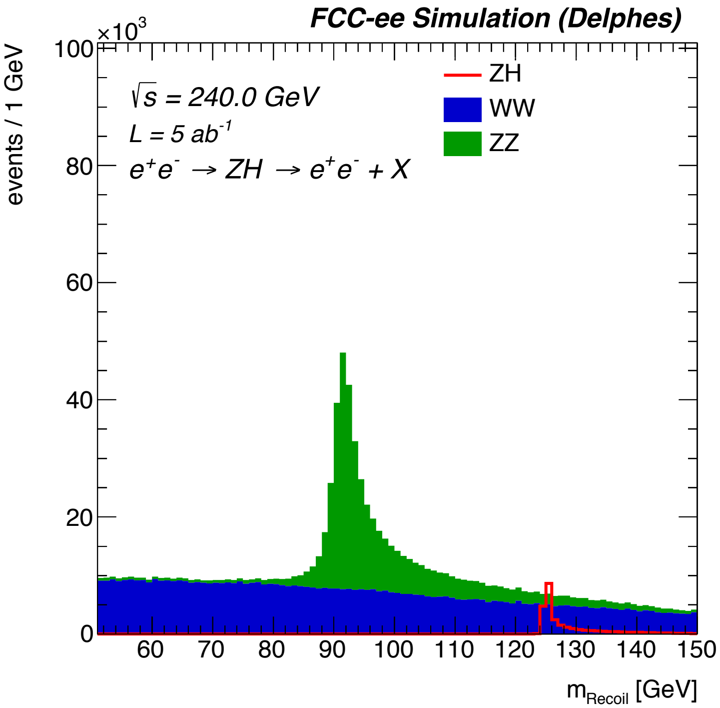
```
# base pre-selections
selbase = 'recoil_m>10.'
selopt  = 'zed_pt<65. && zed_m>70. && zed_m<100. && mu1_pt<75.&& mu2_pt<50. && met_pt<50.'
```

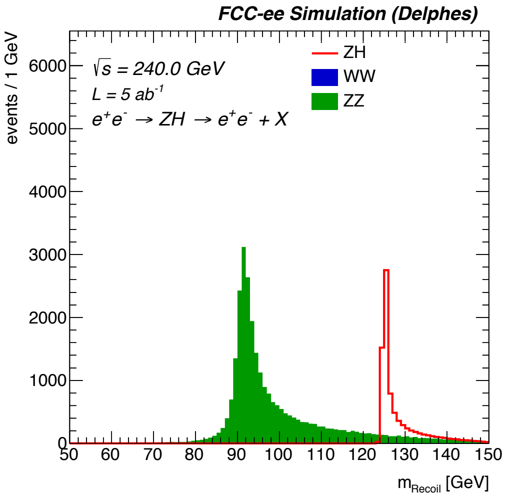




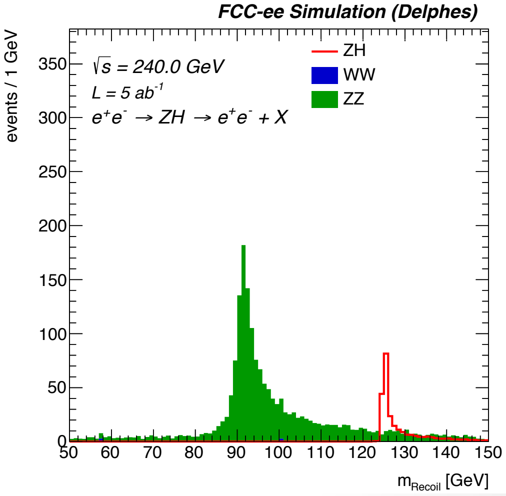
5. Flat Tree analyser

```
# base pre-selections
selbase = 'recoil_m>10.'

selbb    = 'nbjets==2'
seltautau    = 'ntaujets==2'
```





2-bjets



2-taus

FCC-ee events

- Generated 30M each of ZZ, WW and ZH with Pythia8 and processed them through IDEA Delphes card in FCCSW

Home | About | Contact | 100TeV FCC Physics | 27TeV HE-LHC Physics | Full Simulation FCChh | FCCee Physics | Stat

Delphes FCCee Physic events v0.1

Search for names..

NO	NAME	NEVENTS	NWEIGHTS	NFILES	NBAD	NEOS	SIZE (GB)	OUTPUT PATH	MAIN PROCESS	FINAL STATES	CROSS SECTION (PB)
1	p8_ee_ZH_ecm240	29,850,000	0	2985	0	2985	125.57	/eos/experiment/fcc/ee/generation/DelphesEvents/fcc_v01/p8_ee_ZH_ecm240/	ZH ecm=240GeV	inclusive decays	0.201037
2	p8_ee_ZZ_ecm240	29,880,000	0	2988	0	2988	96.24	/eos/experiment/fcc/ee/generation/DelphesEvents/fcc_v01/p8_ee_ZZ_ecm240/	ZZ ecm=240GeV	inclusive decays	1.35899
3	p8_ee_WW_ecm240	29,630,000	0	2963	0	2963	91.49	/eos/experiment/fcc/ee/generation/DelphesEvents/fcc_v01/p8_ee_WW_ecm240/	WW ecm=240GeV	inclusive decays	16.4385
4	total	89,360,000	0.0	8,936	0.0	8,936.0	313.30				

Conclusions

- We provided a **simple, highly modular framework** for performing fast detector simulation

- **Integrated** in MG5 suite and in the FCCSW framework and can be used for FCC and HE-LHC studies

- Can be used and configured for:
  - quick **phenomenological** studies
  - as an **alternative for full-simulation** if accurately tuned

- Reproducibility exists, but could be improved

- Already ~25 analyses using this workflow