

CMSSW Tutorial – Part 3

Goals of Part 3

- ❑ *Understand the concept of the Simulation chain.*
 - *GEN, SIM and DIGI tiers.*
- ❑ *Learn how to simulate our own datasets.*
- ❑ *Run the full-fledged simulation.*
- ❑ ***FINAL GOAL: prepare a full-fledged .cc analysis***

Monte Carlo samples

- ❑ *Monte Carlo Event Generators: generation of events.*
 - *Not the Event from the EDM!*
 - *Here, “event” means a set of outgoing particles produced in the interactions between two incoming particles.*
- ❑ *Centrally produced (“official”) samples: can be found with DBS*

http://cmsdbs.cern.ch/DBS2_discovery/

- ❑ *If you don't like them, you can make your own!*

The Simulation Chain

□ *Generation (GEN)*

- *Physical event – only the data of the particles themselves.*
- *Generated by a particle gun and/or Pythia*
- *Relatively fast*

□ *Simulation (SIM)*

- *Transport of the particles through the detector*
- *Simulation of the physical processes - signals in the detector*
- *Generated by GEANT4*
- *Relatively slow*

□ *Digitization (DIGI)*

- *Simulation of the electronics response to the hits in the detector*
- *Relatively fast*

Making our own simulations

- ❑ *Get IOMC.tar.gz from the sampledata directory, and unzip it inside your CMSSW_1_3_3/src directory.*

```
$ cp ~/sampledata/IOMC.tar.gz yourdirectory/CMSSW_1_3_3/src
```

```
$ tar -xzvf IOMC.tar.gz
```

- ❑ *Go into the IOMC/ParticleGuns/test directory.*

- ❑ *Study and then run the .cfg files.*

- ❑ *Go also into IOMC/GeneratorInterface/test.*

- ❑ *Study and then run the .cfg files.*

– Careful with the # of events – can take a long time!

An analysis example

- ❑ *Run `pythiaHZZ4mu+analysis.cfg`.*
 - *Output is at `TestHZZ4muMass.root`.*
 - *Shows histograms for the Higgs mass.*

- ❑ *Look at the `HZZ4muAnalyzer.cc` file. Can you make any sense of it?*
 - *It is well-commented... but can you grasp what each particular part does?*

...

```
for ( HepMC::GenVertex::particles_out_const_iterator
      pout>(*vit)->particles_out_const_begin();
      pout!=(*vit)->particles_out_const_end(); pout++ )
{
  if ( (*pout)->pdg_id() == 25 && (*pout)->status() == 2 )
  {
    if ( (*pout)->end_vertex() != 0 )
    {
      ...
    }
  }
}
```

The full simulation chain

- ❑ *Which tools do we need?*
 - *And in which modules do they reside?*

- ❑ *We need Pythia / ParticleGun for the source (GEN)*
 - *It's in the IOMC subsystem.*

- ❑ *Geant4 for the detector simulation (SIM, DIGI)*
 - *Unpack the SimG4Core subsystem.*

- ❑ *Step by step construction of the .cfg file.*
 - *Using our newfound knowledge of the Configuration File Language*

1st step: the essentials

□ *The Message Logger*

– *Information about the processing*

```
service = MessageLogger
```

```
{
```

```
  untracked vstring destinations = {"cout"}
```

```
  # untracked vstring categories = { "FwkJob" }
```

```
  untracked PSet cout =
```

```
  {
```

```
    untracked PSet default = { untracked int32 limit = 0 } # kill all
```

```
messages in the log
```

```
    # untracked PSet FwkJob = { untracked int32 limit = -1 } # except *all*
```

```
of FwkJob's
```

```
  }
```

```
  #untracked vstring fwkJobReports = {"FrameworkJobReport.xml"}
```

```
}
```

□ *The Random Number Generator*

- *All Monte Carlo modules use it.*
- *IMPORTANT: one different seed for each module.*
- *The modules' labels: from their .cfi files.*

```
service = RandomNumberGeneratorService
{
  untracked uint32 sourceSeed = 135799753
  PSet moduleSeeds =
  {
    untracked uint32 VtxSmeared = 123456789
    untracked uint32 g4SimHits = 9876
    untracked uint32 siPixelDigis = 2345
    untracked uint32 siStripDigis = 3456
    # untracked uint32 ecalUnsuppressedDigis = 4567
    # untracked uint32 hcalDigis = 5678
    # untracked uint32 muonCSCDigis = 6789
    # untracked uint32 muonDTDigis = 7890
    # untracked uint32 muonRPCDigis = 8901
  }
}
```

2nd step: the source (GEN)

- ❑ *Remember, there must be exactly ONE unnamed source.*

```
include "IOMC/GeneratorInterface/data/PythiaHZZ4mu.cfi"  
replace PythiaSource.maxEvents = 5  
replace PythiaSource.pythiaHepMCVerbosity = false
```

- ❑ *Inside the .cfi, you can find the source itself:*

```
source = PythiaSource  
{  
  untracked int32 maxEvents = 10  
  # to printout pythia event record (call pylist)  
  untracked int32 pythiaPylistVerbosity = 0  
  # to printout HepMC::GenEvent record (HepMC::GenEvent::print())  
  untracked bool pythiaHepMCVerbosity = false  
  untracked int32 maxEventsToPrint = 0  
  
  PSet PythiaParameters =  
  {
```

- Notice that a string of parameters is passed on to Pythia itself afterwards – in the appropriate format.

```
vstring parameterSets =
```

```
{  
  "pythiaHZZmumumumu"  
}
```

```
vstring pythiaHZZmumumumu =
```

```
{  
  "PMAS(25,1)=190.0      ! mass of Higgs",  
  "MSEL=0                ! (D=1) to select between full user control (0, then  
use                        MSUB) and some preprogrammed alternative.  
  "MSTJ(11)=3           ! Choice of the fragmentation function",  
  "MSTJ(41)=1           ! Switch off Pythia QED bremsstrahlung",  
  "MSTP(51)=7           ! structure function chosen",  
  "MSTP(61)=0           ! no initial-state showers",  
  "MSTP(71)=0           ! no final-state showers",  
  "MSTP(81)=0           ! no multiple interactions",  
  "MSTP(111)=0          ! no hadronization",  
  ...  
}
```

□ *Vertex smearing module*

– *Not all collisions happen at from the (0,0,0) nominal vertex.*

```
include "IOMC/EventVertexGenerators/data/VtxSmearedGauss.cfi"
```

□ *Inside the .cfi file, you will find:*

```
#  
# this module takes input in the units of *cm* !!!  
#  
module VtxSmeared = VertexGenerator  
{  
  string type =  
"IOMC/EventVertexGenerators/GaussianEventVertexGenerator"  
  double MeanX = 0.  
  double MeanY = 0.  
  double MeanZ = 0.  
  double SigmaX = 0.0015  
  double SigmaY = 0.0015  
  double SigmaZ = 5.3  
}
```

3rd step: SIM data

- *Call GEANT4 to do this job.*

```
include "SimG4Core/Application/data/SIM-DIGI.cff"
```

- *A .cff file is a configuration file fragment – similar to the .cfi file. It must also be included in a .cfg file.*

- *Inside the .cff, you will find:*

```
#Geometry
```

```
#
```

```
# include "Geometry/CMSCommonData/data/cmsSimIdealGeometryXML.cfi"
```

```
include "Geometry/CMSCommonData/data/cmsIdealGeometryXML.cfi"
```

```
es_module = TrackerGeometricDetESModule {}
```

```
# Magnetic Field
```

```
#
```

```
include "MagneticField/Engine/data/volumeBasedMagneticField.cfi"
```

– *For GEANT4 to work, it must know the detector geometry and the magnetic field configuration.*

□ *You will also find the GEANT4 module itself (it is named after its old name, OSCAR):*

```
module g4SimHits = OscarProducer  
{  
  bool NonBeamEvent = false  
  untracked int32 G4EventManagerVerbosity = 0  
  untracked int32 G4StackManagerVerbosity = 0  
  untracked int32 G4TrackingManagerVerbosity = 0  
  bool UseMagneticField = true  
  ...
```

□ *This takes care of the SIM part. Now for the DIGI part.*

4th step: DIGI data

□ *Each subsystem in CMS is digitized separately*

Digitization of the Tracker

Tracker geometry - required for Pixel and SiStrip Digitizers

include "Geometry/TrackerGeometryBuilder/data/trackerGeometry.cfi"

Pixel's digitization

include "SimTracker/SiPixelDigitizer/data/PixelDigi.cfi"

SiStrip's digitization

include "SimTracker/SiStripDigitizer/data/SiStripDigi.cfi"

- *Extra geometry configuration for the tracker.*
- *Extra modules to do the digitization of the silicon pixels and strips separately.*
- *We're doing only Tracker digitization – but the .cff has configurations for digitizing all subsystems.*

The MixingModule

Pile-up – results from different physics interactions appearing in the event.

- High concentration of particles in a bunch*
- Electronic signal spill-over from previous bunch crossings.*

□ *MixingModule can simulate this for us.*

```
module mix = MixingModule
{
  int32 bunchspace = 25
}
```

□ *MixingModule is mandatory if you want to digitize ECAL, HCAL, and Muon subsystems.*

5th step: the sequences

- *It is not unusual to see included files including files of their own.*
 - *You have to follow them all to understand what is being done.*
- *Scattered among the .cfi and .cff files, the following sequences are defined:*

sequence trDigi = { siPixelDigis & siStripDigis }

sequence calDigi = { ecalUnsuppressedDigis & hcalDigis }

sequence muonDigi = { muonCSCDigis & muonDTDigis & muonRPCDigis }

sequence doAllDigi = { mix, trDigi & calDigi & muonDigi }

sequence doG4SimHitsDigi = { g4SimHits, doAllDigi }

- *We care only for the Tracker part as of now.*

6th step: output and path

- *We will output to a ROOT file:*

```
# Output module
#
module OUTPUT = PoolOutputModule
{
  untracked string fileName = "HZZ4mu.root"
}
```

- *The path and endpath:*

```
path p1 = { VtxSmeared, doG4SimHitsDigi }
```

```
endpath outputPath = { GEN-SIM-DIGI }
```

- *Try it and see if it runs!*

And what next?

- ❑ *The HZZ4muAnalyzer.cc CHEATS!*

- *It ask specifically for particles to be Higgs, for instance!*

- ```
if ((*pout)->pdg_id() == 25
```

- *This line means “if this particle is a Higgs...”*

- ❑ *It uses only GEN data – impossible in real event.*

- ❑ **FINAL GOAL:** *build a .cc file that does analysis the way we are going to do it the LHC!*

- ❑ *Physicists will work mostly with RECO / AOD data – if you're making simulations, they're the next tiers. Try to learn more about those!*

# Resources

❑ *The CMSSW Offline Workbook:*

<https://twiki.cern.ch/twiki/bin/view/CMS/WorkBook>

– *All of this Tutorial has been based on the Workbook*

❑ *The CMSSW Documentation:*

[http://cmsdoc.cern.ch/Releases/CMSSW/latest\\_nightly/doc/html/index.html](http://cmsdoc.cern.ch/Releases/CMSSW/latest_nightly/doc/html/index.html)

– *Advanced documentation*

❑ *The CMSSW CVS Server:*

<http://cmssw.cvs.cern.ch/cgi-bin/cmssw.cgi/CMSSW/>

– *Files that make up CMSSW – accessible with CVS*



*Thank you!*