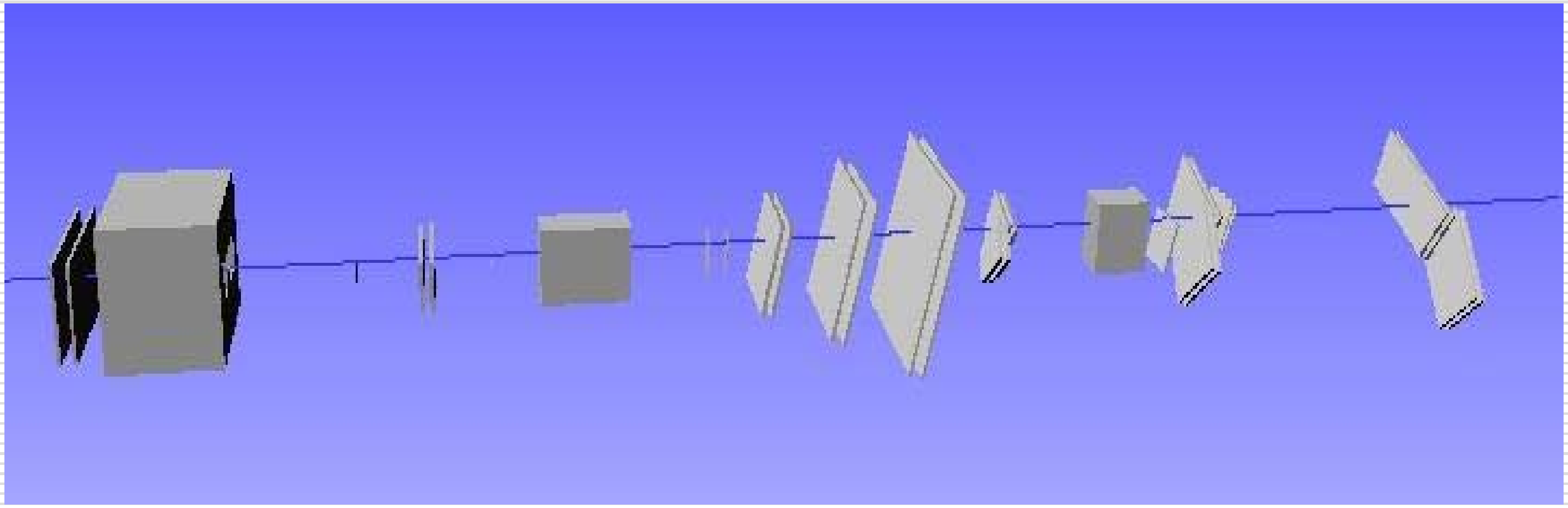
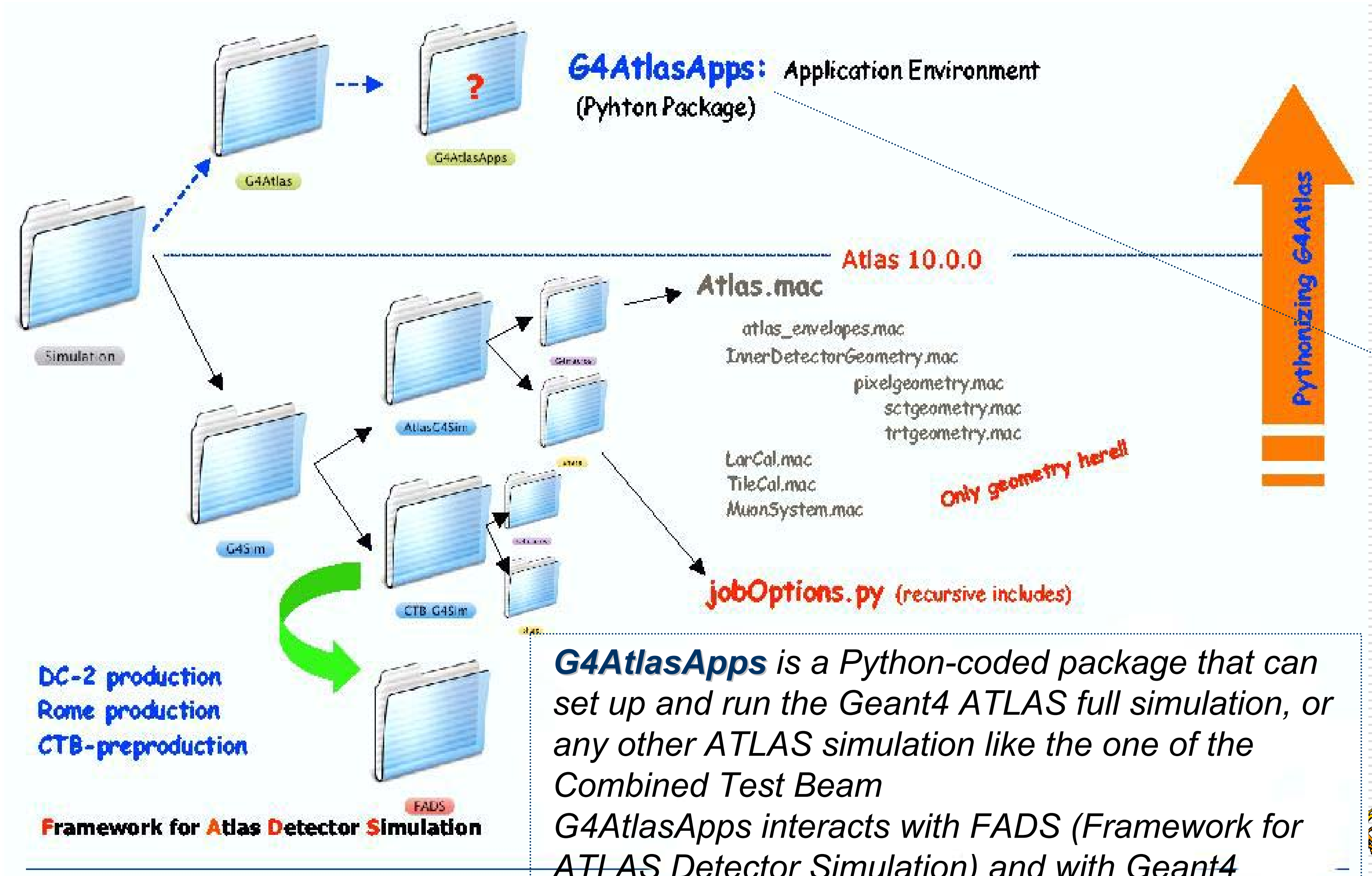
G4Atlas-CTB Simulation



In ~drebuzzi/public/CTBTutorial the file commands_list contains all the command you should type in this tutorial, you can cut&paste from it

Daniela Rebuzzi – Pavia University

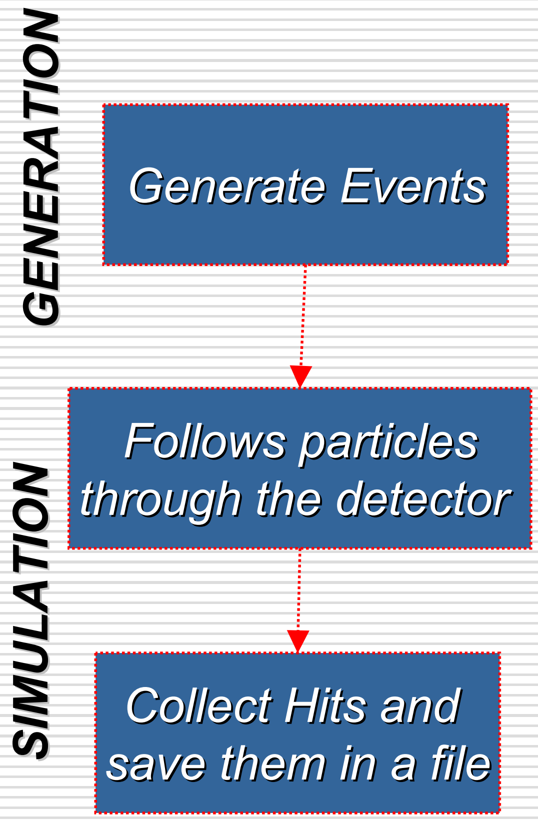
G4Atlas Simulation Roadmap



***G4AtlasApps** is a Python-coded package that can set up and run the Geant4 ATLAS full simulation, or any other ATLAS simulation like the one of the Combined Test Beam*

G4AtlasApps interacts with FADS (Framework for ATLAS Detector Simulation) and with Geant4

G4Atlas Simulation at a glance


GENERATION
Generate Events
SIMULATION
Follows particles through the detector
Collect Hits and save them in a file

- *The detector simulation takes as* ***input*** *the output from generators which consists of list of particles and their properties at specified vertices and follows the particles as they would traverse through the detectors*
- *Each time a particle hits a so-called "sensitive" detector element, a* ***hit*** *is recorded*
- *Hits are the* ***output*** *of the simulation, they are stored in HitCollections (AthenaHitsVector) and saved into a POOL file*

➜*Hits will be the input of the digitization, the simulation of the detector response*

G4Atlas-CTB: HowTo get started

- Create a directory for your CTB simulation

```
$CTBSim> mkdir CTBSim
$CTBSim> cd CTBSim
```

- Set up the correct requirements file

```
$CTBSim> cp ~drebuzzi/public/CTBTutorial/requirements .
$CTBSim> source
   /afs/cern.ch/sw/contrib/CMT/v1r16p20040901/mgr/setup.sh
$CTBSim> cmt config
$CTBSim> source setup.sh -tag=10.0.2,opt
$CTBSim> mkdir 10.0.2
$CTBSim> cd 10.0.2
```

You do not have to check out any package

```
$10.0.2> source /afs/cern.ch/atlas/software/dist/10.0.2/
   Simulation/G4Atlas/G4AtlasApps/G4AtlasApps/00-00-24/cmt/setup.sh
$10.0.2> get_files PDGTABLE.MeV
```

That is all! You can run the CTB full simulation!

```
$10.0.2> athena.py G4AtlasApps/jobOptions.G4Ctb_Sim.py
```

➜This test jobO calls the SingleParticle generator, simulates three electron events and writes out an output POOL file, “ctb_MyOutputFile.root”, which contains the hits

G4Atlas-CTB: Job Configuration

- Retrieve the jobOptions.G4Atlas_Sim.py from the release

```
$10.0.2> get_files -jo G4AtlasApps/jobOptions.G4Ctb_Sim.py
```

It is a python script which defines the simulation configuration

➜ ***Job configuration using DetFlags and SimFlags***

For most of the use cases, the flag mechanism gives enough freedom to configure the job and allows to load the minimal number of libraries and dependencies

- *The **Detector Flags** describe the detectors involved in a simulation job*
- *The **Simulation Flags** are used to configure specifically the simulation*

    ➜*They may carry also a value apart from the boolean status (True/False)*

    - *They appear at the beginning of the jobOption file and they give the possibility to set*

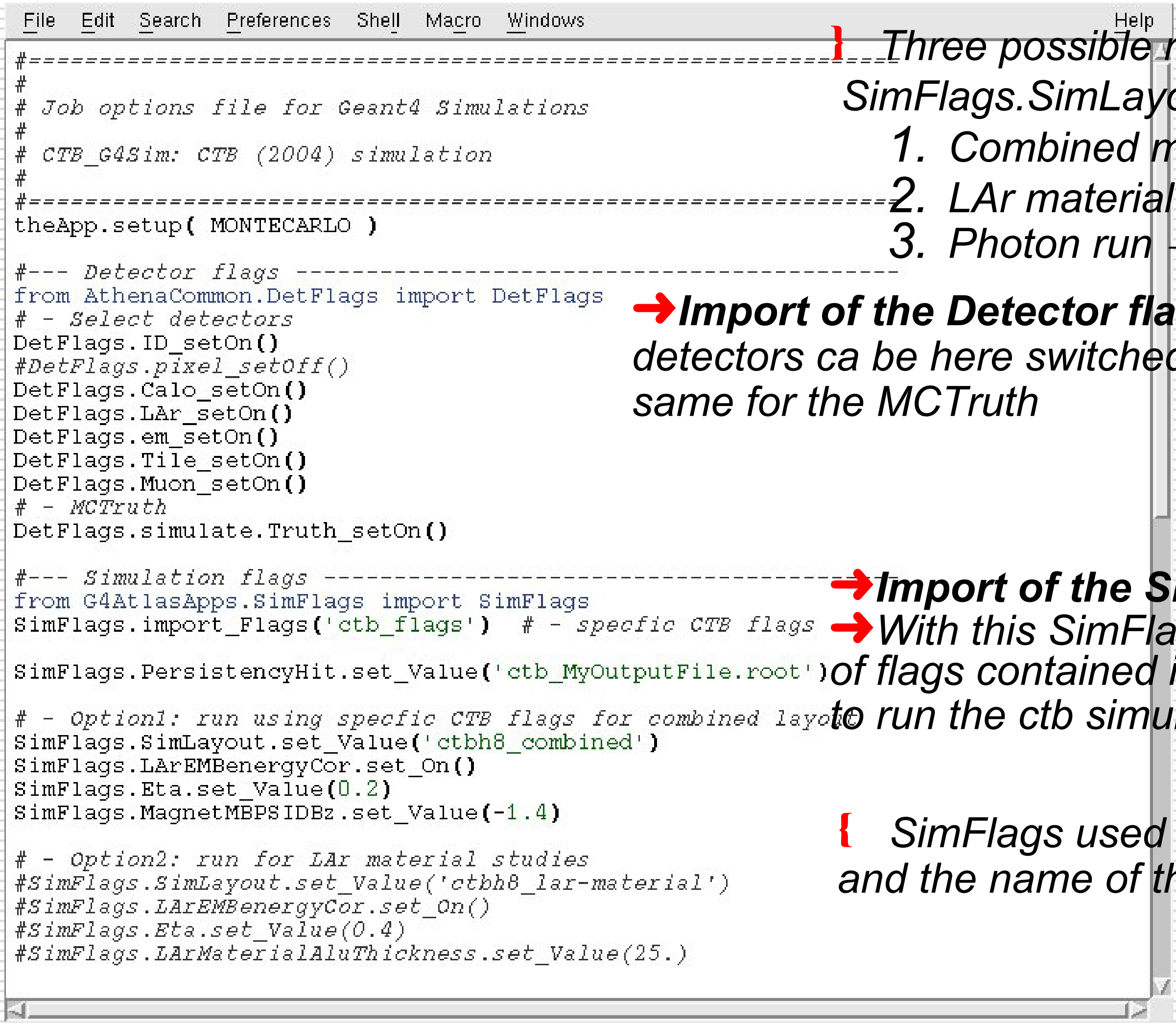
Energy GeneratorPath KinematicsMode ParticlePDG PersistencyDigit PersistencyHit PhysiscList Seeds SimLayout etc. etc. ➜*See the file ctbFlags.out in my public/CTBTutorial*

- *The SimFlags import method present in the jobO*

```
SimFlags.import_Flags('ctb_flags')
```

enriches the initial set of flags with all the flags contained in the ctb_flags.py file (located in G4AtlasApps/python)

G4Atlas-CTB: a look at the jobO



```
#================================================================
#
# Job options file for Geant4 Simulations
#
# CTB_G4Sim: CTB (2004) simulation
#
#================================================================
theApp.setup( MONTECARLO )

#--- Detector flags -------------------------------------------
from AthenaCommon.DetFlags import DetFlags
# - Select detectors
DetFlags.ID_setOn()
#DetFlags.pixel_setOff()
DetFlags.Calo_setOn()
DetFlags.LAr_setOn()
DetFlags.em_setOn()
DetFlags.Tile_setOn()
DetFlags.Muon_setOn()
# - MCTruth
DetFlags.simulate.Truth_setOn()

#--- Simulation flags -----------------------------------------
from G4AtlasApps.SimFlags import SimFlags
SimFlags.import_Flags('ctb_flags')  # - specfic CTB flags

SimFlags.PersistencyHit.set_Value('ctb_MyOutputFile.root')

# - Option1: run using specfic CTB flags for combined layout
SimFlags.SimLayout.set_Value('ctbh8_combined')
SimFlags.LArEMBenergyCor.set_On()
SimFlags.Eta.set_Value(0.2)
SimFlags.MagnetMBPSIDBz.set_Value(-1.4)

# - Option2: run for LAr material studies
#SimFlags.SimLayout.set_Value('ctbh8_lar-material')
#SimFlags.LArEMBenergyCor.set_On()
#SimFlags.Eta.set_Value(0.4)
#SimFlags.LArMaterialAluThickness.set_Value(25.)
```

} *Three possible run configuration available with SimFlags.SimLayout*

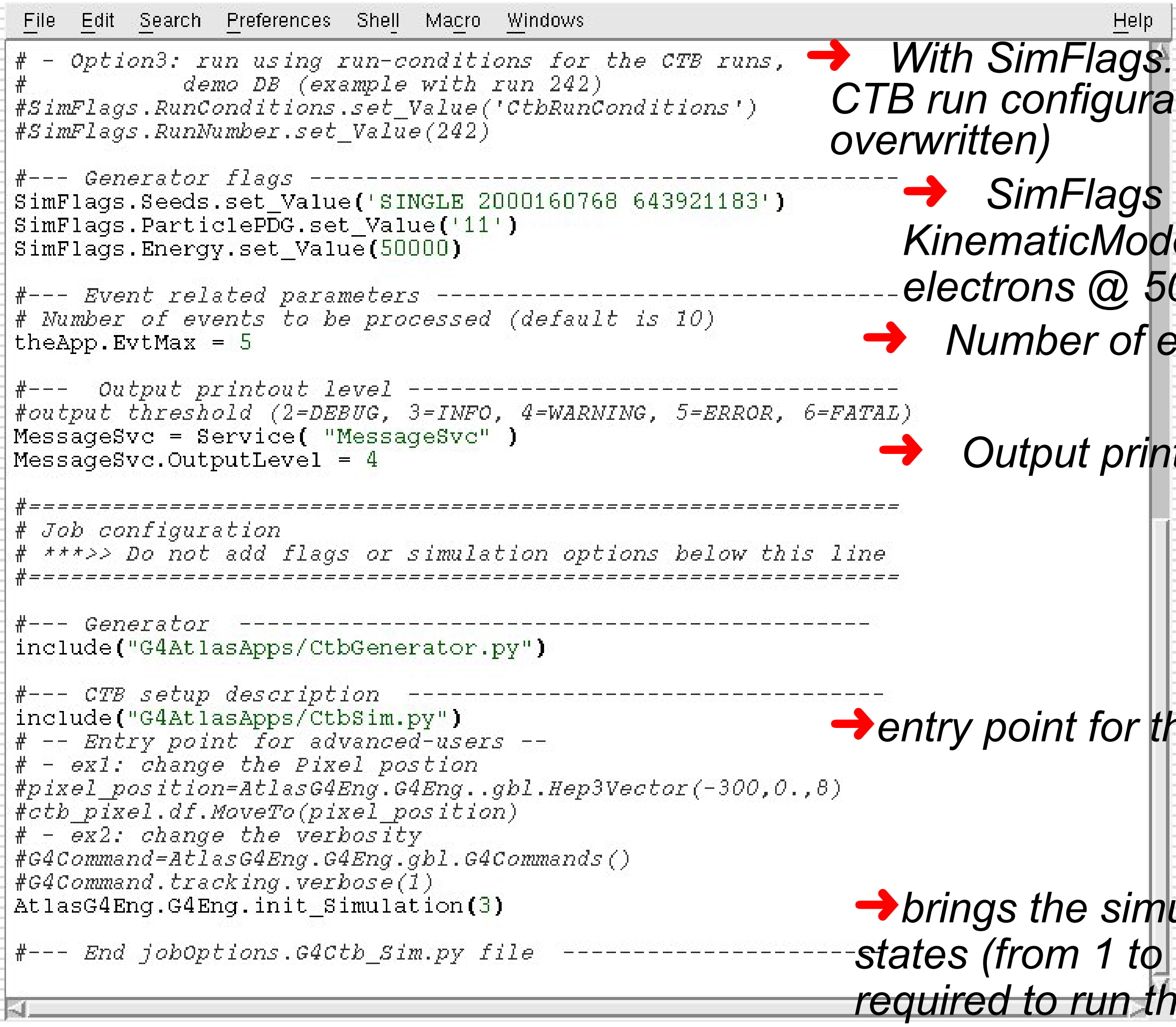
1. *Combined mode → "ctbh8_combined"*
2. *LAr material studies → "ctbh8_lar-material*
3. *Photon run → "ctbh8_photon"*

➜***Import of the Detector flags***
detectors ca be here switched on/off, the same for the MCTruth

➜***Import of the Simulation flags***
➜*With this SimFlags import method, the initial set of flags contained in **ctb_flags.py** (common flags to run the ctb simulation) are imported*

{ *SimFlags used here to set the persistency and the name of the output hit file*

G4Atlas-CTB: a look at the jobO



```
# - Option3: run using run-conditions for the CTB runs,
#            demo DB (example with run 242)
#SimFlags.RunConditions.set_Value('CtbRunConditions')
#SimFlags.RunNumber.set_Value(242)

#--- Generator flags ------------------------------------------
SimFlags.Seeds.set_Value('SINGLE 2000160768 643921183')
SimFlags.ParticlePDG.set_Value('11')
SimFlags.Energy.set_Value(50000)

#--- Event related parameters ---------------------------------
# Number of events to be processed (default is 10)
theApp.EvtMax = 5

#---  Output printout level -----------------------------------
#output threshold (2=DEBUG, 3=INFO, 4=WARNING, 5=ERROR, 6=FATAL)
MessageSvc = Service( "MessageSvc" )
MessageSvc.OutputLevel = 4

#==============================================================
# Job configuration
# ***>> Do not add flags or simulation options below this line
#==============================================================

#--- Generator  -----------------------------------------------
include("G4AtlasApps/CtbGenerator.py")

#--- CTB setup description  -----------------------------------
include("G4AtlasApps/CtbSim.py")
# -- Entry point for advanced-users --
# - ex1: change the Pixel postion
#pixel_position=AtlasG4Eng.G4Eng..gbl.Hep3Vector(-300,0.,8)
#ctb_pixel.df.MoveTo(pixel_position)
# - ex2: change the verbosity
#G4Command=AtlasG4Eng.G4Eng.gbl.G4Commands()
#G4Command.tracking.verbose(1)
AtlasG4Eng.G4Eng.init_Simulation(3)

#--- End jobOptions.G4Ctb_Sim.py file --------------------------
```

→ With SimFlags.RunConditions set a specific CTB run configuration (any other flag will be overwritten)

→ SimFlags used here to Set the KinematicMode to SingleParticle and select electrons @ 50 GeV

→ Number of events to be processed

→ Output printout level

→entry point for the simulation

→brings the simulation to different init states (from 1 to 3), the state 3 is the required to run the simulation

Running in interactive mode

- The athena interactive prompt can be accessed with the following command

```
$CTBSim> athena -i jobOptions.G4Ctb_Sim.py
$athena>
$athena> theApp.run()
$athena> theApp.finalize()
```

➜Running the **local** jobO

➜Starts the simulation application

➜Finalize it

➜apart from the possibility to perform some shell commands (like: ls, pwd, cp ....), the athena prompt offers access to the configuration of simulation job and to the help needed to know about the simulation options & flags

From the athena prompt there are three important objects which can be accessed

```
$athena> DetFlags.Print( )
$athena> SimFlags.print_Flags()
$athena> AtlasG4Eng.G4Eng.Name
```

➜The DetFlags

➜The simulation options and SimFlags

➜For the more advanced use, the Python Geant4 engine ("AtlasG4Eng.G4Eng") is available in the jobOption file and it provides full interactivity and introspection in the current simulation. here its name is retrieved

All DetFlags and SimFlags can be navigated from the athena prompt typing their name followed by . and pressing the <Tab>

➜Flags are auto-documented

G4Atlas-CTB : HowTos

How to visualize the CTB Geometry with VRMLVIEW

```
$CTBSim> export G4VRMLVIEW_VIEWER=/afs/cern-ch/user/d/drebuzzi/
    public/Tutorial/vrmlview
$CTBSim> export LD_LIBRARY_PATH=/afs/cern-ch/user/d/drebuzzi/
    public/Tutorial/$LD_LIBRARY_PATH
```

Check if everything is OK by typing

```
$CTBSim> ~drebuzzi/public/Tutorial/vrmlview
```

a vrmlview window should open

OK, close it and run athena in interactive mode

```
athena> G4command=AtlasG4Eng.G4Eng.gbl.G4Commands()
athena> G4command.vis.open("VRML1FILE")
athena> G4command.vis.drawVolume()
athena> G4command.vis.viewer.flush()
```

➜*Here the AtlasG4Eng.G4Eng is used to create the python interface to G4*

and you will get a ".wrl" file → exit athena and open the file with

```
$CTBSim> ~drebuzzi/public/CTBTutorial/vrmlview g4_00.wrl
```

➜*You better switch off the visibility property of some detector, if you want to run quickly the visualization*

➜*You can customize the volumes to be visualized by changing their vis property → see commands_list for an example*

G4Atlas-CTB: HowTos

How to run the geantino map to check the detector positions

- *Edit the jobOptions.G4Ctb_Sim.py*
- *Select the ParticleGenerator and set "999" as PDGcode for the geantinos*
- *Uncomment these two lines in the jobO*

```
G4Command=AtlasG4Eng.G4Eng.gbl.G4Commands()
G4Command=tracking.verbose(1)
```

- *Run athena with jobOptions.G4Ctb_Sim.py saving the output in the file "geant_map.out" → look at the output file*

How to write hits in POOL

use the SimFlags.PersistencyHits.setValue("nameOfTheOutputFile"), this will

- *set the persistency of the hits to True and*
- *pass to the python interface the chosen name of the hit file*

G4Atlas-CTB: HowTos

How to check the energy loss in the different detectors

- *Edit the jobOptions.G4Ctb_Sim.py*
- *Select the ParticleGenerator KinematicsMode and set the PDGcode for the selected particle (say, mu- = 13)*
- *Uncomment these two lines in the jobO*

```
G4Command=AtlasG4Eng.G4Eng.gbl.G4Commands()
G4Command=tracking.verbose(1)
```

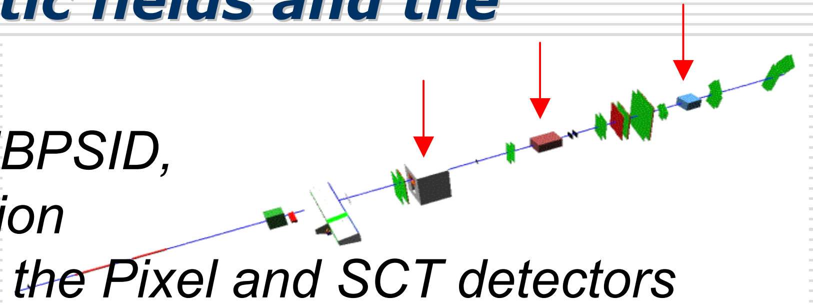
- *Run athena with jobOptions.G4Ctb_Sim.py saving the output in the file "enloss.out" → look at the output file*
- *On it you can perform a little analysis, as an example*
  - ➜*Copy from drebuzzi/public/CTBTutorial the awk script "exEnLoss.exe" and run it on the "enloss.out" file with*

```
awk -f exEnLoss.exe enloss.out > enloss.dat
```

  - ➜*Create an ROOT/HBOOK histogram reading enloss.dat (copy an example from my public/CTBTutorial, enHisto.C)*

G4Atlas-CTB: HowTos

How to switch on/off the magnetic fields and the magnets



- *There are three main magnets called MBPSID, MBPL and MBPS2 along the beam direction*
    1. *MBPSID, the first in the line, hosts the Pixel and SCT detectors*
    2. *MBPL and MBPS2 magnets are in the Muon region*
- *Two additional magnets, MBPL13 and MBPL12, are placed upstream MBPSID for the photon runs*

- *By default, only MBPSID is positioned and ON, with a -1.4 T field*

- *Magnets can be handled using the SimFlags*
    - *Add the magnets to the setup*

      `SimFlags.MagnetMBPLBy.set_Value(2.0)`
    - *Position a magnet without switching on its field*

      `SimFlags.MagnetMBPLBy.set_Value(0.0)`

G4Atlas-CTB: HowTos

How to use the ParticleGenerator

G4Ctb is using the ParticleGenerator package, which generates single particles and puts them into HepMC standard format

- *ParticleGenerator can be handled using the SimFlags. What is possible to select, at the moment*

```
SimFlags.Energy.set_Value(100000.0)
SimFlags.ParticlePDG.set_Value(13)
```

How to set the run configuration

Something is useful to associate the simulation options and flags to a given run number, real or fictitious
This way, the setting can be retrieved (from a database or local data) and set in one single python line

```
SimFlags.RunConditions.set_Value('MyRunCondition')
SimFlags.RunNumber.set_Value(242)
```

➜Defines the run conditions storage as 'MyRunConditions' and asks for run number 242

G4Atlas-CTB: HowTos

How to check/retrieve hits saved in a POOL file

• *A monitoring algorithm, ReadMuonSimHits, located in the repository under MuonSpectrometer/MuonDigitization/MuonDigiExample, can be used to check whether a POOL hit file contains Muon hits*

•*This algorithm simply dumps hits out if not-empty Muon XXXSimHitCollections are found on StoreGate*

• *To use it, copy the ReadMuonSimHitsOptions.py from MuonDigiExample/share or from my ~drebuzzi/public/CTBTutorial*

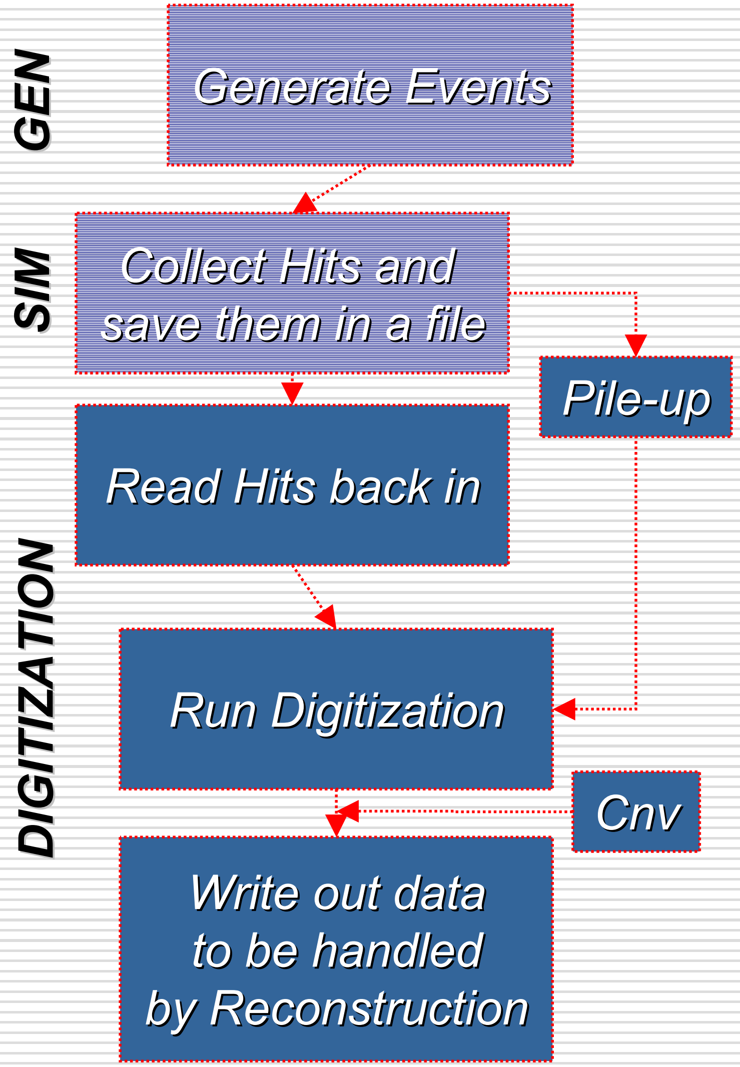
```
$CTBSim> cp ~drebuzzi/public/CTBTutorial/ReadMuonSimHitsOptions.py .
```

and edit it to set the input file name to be re-read

• *Run it with*

```
$CTBSim> athena.py ReadMuonSimHitsOptions.py
```

Atlas-CTB Digitization


GEN
SIM
DIGITIZATION
Generate Events
Collect Hits and save them in a file
Pile-up
Read Hits back in
Run Digitization
Cnv
Write out data to be handled by Reconstruction

- The **digitization** takes as ***input*** the output from the simulation (HitCollections) and applies the effect resulting from the detector and its electronics
- The ***output*** are the so-called digits which are subsequently converted into bytestream or RDOs (Raw Data Objects = object copy of the bytestream content) which are the persistified objects
- Bytestreams or RDOs (and Digits) are the final product of the simulation chain and are fed into the reconstruction

➜ Separation between event and detector data → connection via identifier objects

G4Atlas-CTB: HowTos

How to run the digitization and write digits in POOL

The jobOptions.CTB_G4Dig.py in the share directory reads the POOL file with hits and produces another POOL file with digits

- *to run the digitization you should type from your run dir*

```
$CTBSim> athena.py CTB_G4Sim/jobOptions.CTB_G4Dig.py
```

the default jobO creates the digit POOL file ctb_MyOutputFileDig.root reading from the POOL hits file ctb_MyOutputFile.root

➜ *You can modify these names in the jobOptions.CTB_G4Dig.py*

- *if you have plans to modify the jobO, retrieve it with*

```
$CTBSim> get_files -jo jobOptions.CTB_G4Dig.py
$CTBSim> athena.py jobOptions.CTB_G4Dig.py
```

G4Atlas-CTB: HowTos

How to digitize an already generated hit file on castor

• *if you want to read an already generated hit file on castor → edit the jobOptions.CTB_G4Dig.py and modify the InputCollection name*

```
EventSelector.InputCollection =
"rfio:/castor/cern.ch/atlas/ctb/test/monte_carlo/simulation/muon-
/preprod_g4sim.CTB_G4Sim_mu-_180_GeV_eta_0.2_Mag_0.v1.900.00001.root"
```

and copy the corresponding PoolFileCatalog in your run dir

```
$CTBSim> rfcp
/castor/cern.ch/atlas/ctb/test/monte_carlo/simulation/muon-/
PoolFileCatalog.preprod_g4sim.CTB_G4Sim_mu_180_GeV_eta_0.2_Mag_0
.v1.900.00001.xml .
```

➜ *The PoolFileCatalog is not necessary running with releases > = 10.0.1*

the list of the available hit files already generated can be gotten by typing

```
$CTBSim> rfdir /castor/cern.ch/atlas/ctb/test/monte_carlo/simulation
```

or at the following web page (not yet updated)

http://atlas.web.cern.ch/Atlas/GROUPS/SOFTWARE/OO/testbeam/simulation CTB/productionMC.html

G4Atlas-CTB: Digitization Options

```
# - Select detectors
DetFlags.ID_setOff()
DetFlags.Calo_setOff()
DetFlags.em_setOff()
DetFlags.Tile_setOff()
DetFlags.Muon_setOn()
DetFlags.Truth_setOff()
#DetFlags.LVL1_setOn()

# - Switch off tasks
DetFlags.pileup.all_setOff()
DetFlags.simulate.all_setOff()
DetFlags.makeRIO.all_setOff()
DetFlags.writeBS.all_setOff()
DetFlags.readRDOBS.all_setOff()
DetFlags.readRIOBS.all_setOff()
DetFlags.readRIOPool.all_setOff()
DetFlags.writeRIOPool.all_setOff()
#DetFlags.writeRDOPool.all_setOff()
```

```
RPC_Digitizer.WindowLowerOffset = -1000
RPC_Digitizer.WindowUpperOffset = 1000
RPC_Digitizer.CTB2004 = TRUE;
MDT_Digitizer.OffsetTDC = 0.
MDT_Digitizer.BunchCountOffset = -200
MDT_Digitizer.UseTof = FALSE
MDT_Digitizer.UseProp = FALSE
```

Properties of the Muon Digitization, optimized for the testbeam setup

The digitization can be customized by changing the DetFlags in jobOptions.CTB_G4Dig.py

Muon Digitization - CTB

Two digitization tools available:

1. ***MDT_Response_DigiTool*** *(default): detailed simulation of MDT response including cluster size fluctuations, diffusion and the ADC response (slewing correction)*

You can tune many parameters, e.g. the electronic threshold

```
ToolSvc = Service( "ToolSvc" )
ToolSvc.MDT_Response_DigiTool.Threshold = 20
```

2. ***RT_Relation_DigiTool****: transforms r→t + smearing in time using an external rt relations (from Garfield, from data)*

If you want to select this → add the following line and uncommented it

```
# Uncomment the following line if you want to use MDT digitization
# with external RT relation (default is Nikhef algorithm digi.)
#MDT_Digitizer.DigitizationTool =   "RT_Relation_DigiTool"
```

➜ *the default rt relation which is taken is the ArCO2.rt file in MuonSpectrometer/ MuonDigitization/MDT_Digitization/share*

Be careful because the t→r conversion in the reconstruction must be selected consequently!

Muon Digitization - CTB

How to set a customized rt relation

If you want to change the rt relation, using your customized one

1. check that you have both the rt and the tr file, in the standard format "à la Calib"

2. select the RT_Relation_DigiTool in the jobOptions.CTB_G4 Sim.py, as explained in the previous slide

3. check out the MDT_Digitization

```
$CTBSim> cmt co MuonSpectrometer/MuonDigitization/MDT_Digitization
```

4. in the share directory, replace the file ArCO2.rt with your rt file (r, t, resolution on r), for example

```
$CTBSim> cp ~drebuzzi/public/CTBTutorial/ArCO2_ludo.rt/
   MuonSpectrometer/MuonDigitization/MDT_Digitization/MDT_Digitizat
   ion-*/share
```

5. compile the package

6. be sure to pass to the reconstruction the same tr relation → e.g.

```
MuonTBCalibrationSvc.RT_InputFiles = [ "/afs/cern.ch/user/d/
drebuzzi/public/CTBTutorial/ArCO2_ludo.tr" ]
```

G4Atlas-CTB: HowTos

How to retrieve digits saved in a POOL file

• *A monitoring algorithm, ReadMuonDigits, located in the repository under MuonSpectrometer/MuonDigitization/MuonDigiExample, can be used to check whether a POOL digit file contains Muon hits*

• *This algorithm simply dumps hits out if not-empty Muon DigitCollections are found on StoreGate*

• *To use it, copy the ReadMuonDigitsFromDigitFileOptions.py from MuonDigiExample/share or from my ~drebuzzi/public/Tutorial*

```
$CTBSim> cp ~drebuzzi/public/CTBTutorial/ReadMuonDigitsFrom
DigitFileOptions.py .
```

and edit it to set the input file name to be re-read

• *Run it with*

```
$CTBSim> athena.py ReadMuonSimDigitsFromDigitFileOptions.py
```

G4Atlas-CTB: Exercise 1

1. create a sample of 10 mu- events @ 250GeV, commenting out all the detectors, but the muons
2. re-read the hits using the jobOptions ReadMuonSimHitOptions.py $\rightarrow$ from the run dir
3. run the digitization on the generated hit file, using the default digitization tool
4. re-read the digits with the jobOptions ReadMuonDigitFromDigitFileOptions.py
5. (reconstruct events with RecExTB)

G4Atlas-CTB: Exercise 2

1. run the digitization and the reconstruction on the fly (using the jobOptions.CTB.G4Dig+Rec.py from my public/CTBTutorial) on the hit file simul.10.0.0.muon_only.b.01.root

```
$CTBSim> cp drebuzzi/public/CTBTutorial/simul.10.0.0.muon.only
   .b.01.root .
$CTBSim> cp ~drebuzzi/public/CTBTutorial/jobOptions.CTB_G4Dig+Rec.py .
$CTBSim> source /afs/cern.ch/atlas/software/dist/10.0.2/AtlasRelease/
   AtlasRelease-10-00-02/cmt/setup.sh
```

2. analyze the reconstruction root file (ntuple.root) and check the sagitta and the residuals for the barrel chambers

3. (compare the results with the ones obtained on the 250GeV data sample analyzed before)

Useful Links and Documentation

- ***Geant4 ATLAS Application***

http://atlas-sw.cern.ch/cgi-bin/viewcvs atlas.cgi/offline/Simulation/ G4Atlas/G4AtlasApps/doc/index.html

- ***G4 Simulation web page***

http://atlas.web.cern.ch/Atlas/GROUPS/SOFTWARE/OO/simulation/geant4/

- ***CTB_G4Sim progress by tag***

http://mgallas.home.cern.ch/mgallas/ctb_atlas/ctb_progress.html

- ***Status of Simulation production***

http://atlas.web.cern.ch/Atlas/GROUPS/SOFTWARE/OO/testbeam/simulationCTB/productionMC.html

- ***Main CTB web page***

http://atlas.web.cern.ch/Atlas/GROUPS/SOFTWARE/OO/testbeam/testbeam.html

- ***The 2004 Atlas Muon Test Beam in H8***

http://atlas/tb/muon.web.cern.ch/atlas-tb-muon

- ***"ATLAS Barrel Combined Run in 2004 Test Beam Setup and its evolution"****, B.Di Girolamo, M.Gallas and T.Koffas, EDMS Note: ATC-TT-IN-0001*

Acknowledgements

Many thanks! *to*

Manuel - Daniele