

V2.3 Gaudi v20r3 and later

Using a DST as a big NTuple

Introduction about how to use GaudiPython together with ROOT to have a quick and simple access to data stored on a DST

Changes to previous versions

- ◆ January 2008
 - ▶ Try to make maximum advantage of latest changes in Configurables
 - ▶ Pack basic configuration in a python script, LHCbConfig.py. Makes example scripts cleaner
- ◆ October 2008
 - ▶ Moved example scripts to CVS: Vis/GaudiPythonTutorial, released with Panoramix.
 - ▶ After SetupProject Panoramix: python \$GAUDIPYTHONTUTORIALROOT/python/Ex1.py
- ◆ June 2008
 - ▶ Update for Gaudi v19r9
 - ▶ Added examples: xx_multicore.py
- ◆ March 2008
 - ▶ Update for Gaudi v19r8, using of configurables
 - ▶ New place for example scripts:
/afs/cern.ch/lhcb/group/panoramix/vol1/GaudiPythonTutorial/Gaudi_v19r8
- ◆ September 2007:
 - ▶ Tutorial for Software week
 - ▶ Reference to example scripts on afs at ~truf/public/GaudiPythonTutorial/
- ◆ July 2007:
 - ▶ Added slide about MCMuonInfo
 - ▶ Added list of more advance Python scripts
 - ▶ Add slides about event tag collection
 - ▶ Fix some typos
 - ▶ Add page for using iterators
- ◆ June 2007:
 - ▶ Replace rootSvc with aida2root

Useful links

- ◆ [Python Home Page](#), [Python Tutorial](#), [Python Quick Reference](#)
- ◆ [GaudiPython Twiki](#)

Tutorials & Presentations:

(Caution: many recent improvements, therefore presentations not anymore up-to-date in all areas)

- ◆ [GaudiPython Tutorial](#) (Pere Mato)
- ◆ [Introduction to GaudiPython](#) (Jose A. Hernando)
- ◆ [Bender](#) (Vanya Belyaev)
- ◆ [Interactive analysis with Python, Examples](#) (T.Ruf)
- ◆ [PyRoot](#) (Wim Lavrijsen)

- Python is a scripting language.
- No compilation, ideal for interactive analysis, and fast prototyping.
- Python knows about C++ objects and methods via dictionaries. Full access to fast and debugged C++ algorithms and tools.
- Python allows to work with different applications at the same time: Gaudi + ROOT + OpenScientist + ..., Python acts as glue
- You don't have to read manuals or books before doing useful physics analysis !

Probably the most complicated part

- Software in LHCb is managed using CMT
- CMT sets up the environment for an application
 - ◆ Logical names, path to libraries, include files, ...
- Developers would like to have environment as restricted as possible to avoid interferences
- End-users would like to have an environment with access to everything
- Best solution:
 - ◆ SetupProject Panoramix v16r3
 - ◆ 90% will also work with DaVinci or Brunel environment

- Three ways of running python:
 - ◆ Execute a script:
 - ▶ `python ex0.py`
 - ◆ Execute a script and return to python prompt:
 - ▶ `python -i ex0.py`
 - ▶ Exit python with ctrl-d (Linux) or ctrl-z (Windows)
 - ◆ Entering commands by hand or by cut and paste:
 - ▶ `python`

Example scripts can be found at `$GAUDIPYTHONTUTORIALROOT/python/Ex1.py` or in CVS: `Vis/GaudiPythonTutorial`

For a full list of examples go [here](#)

▶ python

```
>>> import GaudiPython
```

◆ Everything inside GaudiPython will be in the namespace GaudiPython :

```
>>> dir(GaudiPython)
['AppMgr', 'Bindings', 'CallbackStreamBuf', ...]
```

```
>>> dir(GaudiPython.AppMgr())
[ ... , 'addAlgorithm', 'algorithm', 'algorithms', 'config',
'configure', 'createSvc', 'datasvc', 'declSvcType', 'detSvc',
'detsvc', 'evtSel', 'evtSvc', 'evtsel', 'evtsvc', 'execute',
'executeEvent', 'exit', 'finalize', ... ]
```

◆ Also possible:

```
>>> import GaudiPython as gaudi
>>> dir(gaudi)
['AppMgr', 'Bindings', 'CallbackStreamBuf', ...]
```

■ `dir("something")` useful command to quickly see what you could do with "something", or `dir()` to see what is around.

■ Another useful command: `help("something")`

- ◆ With Gaudi v19r9, any LHCb application is configured and started with Python
- ◆ Essentially two steps
 - ▶ Configuring
 - ▶ Running

◆ Configuring

```
>>> from Gaudi.Configuration import *
```

- ▶ # load old option files

```
>>> importOptions('$STDOPTS/LHCbApplication.opts')
```

```
>>> importOptions('$STDOPTS/DstDicts.opts')
```

- ▶ # can also load python options

```
>>> importOptions('myStartup.py')
```

myStartup.py :

```
importOptions('$STDOPTS/LHCbApplication.opts')
importOptions('$STDOPTS/DstDicts.opts')
```

◆ Running

```
>>> appMgr = GaudiPython.AppMgr()
```

```
>>> appMgr.run(1)
```

Reading files: EventSelector and Transient Event Store

- ◆ If not specified in the option file, the input file can be specified using the eventselector:

```
>>> sel = appMgr.evtSel()
```

```
>>> sel.open(['PFN:$AFSROOT/cern.ch/lhcb/group/tracking/vol1/00001378_00000002_5.dst'])  
# [] = list of files or one file
```

- ◆ Get event service for later use to access the transient event store

```
>>> evt = appMgr.evtSvc()
```

- ◆ Read one event

```
>>> appMgr.run(1)
```

- ◆ For better reading of the example scripts, the most basic configuration is moved to LHCbConfig.py

- ◆ It is enough to have in the user script

```
>>> from LHCbConfig import *  
>>> lhcbApp.DataType = 'DC06'  
or  
>>> lhcbApp.DataType = '2008'
```

- ◆ LHCbConfig.py sets up

- ▶ reading of files and unpacking of containers,
- ▶ starts the data on demand and particle property service
- ▶ decoding of raw data banks

```
>>> from LHCbConfig import *
      lhcbApp.DataType = 'DC06'
>>> # configure application manager
      appConf = ApplicationMgr( Outputlevel = INFO )
>>> import GaudiPython
>>> # some additional features:
      import gaudigadgets
>>> # start application manager
      appMgr = GaudiPython.AppMgr()
>>> sel = appMgr.evt sel ()
>>> sel.open(['PFN:$AFSROOT/cern.ch/lhcb/group/tracking/vol1/00001378_00000002_5.dst'])
>>> evt = appMgr.evt svc ()
>>> appMgr.run(1)
>>> evt.dump()
```

◆ Could be executed as:

- ▶ python Ex1.py
 - After script completion, python stops. In this case, not very useful.
- ▶ python -i Ex1.py
 - After script completion, returns to python prompt. Useful for continuing interactive session.
- ▶ python, followed by cut and paste from an editor
- ▶ In batch mode, ROOT needs -b argument otherwise will start graphics:
 - python Ex1.py -b

■ Transient event store

```
>>> evt.dump()
```

```
/Event
/Event/Gen
/Event/MC
/Event/DAQ
/Event/Next
/Event/Prev
/Event/PrevPrev
/Event/Link
/Event/pSim
/Event/pRec
```

◆ Notice:

- ▶ not all data are read, only requested.
- ▶ Some data are packed, pSim, pRec, requires unpacking

```
>>> evt['Rec/Vertex/Primary']
```

```
>>> evt.dump()
```

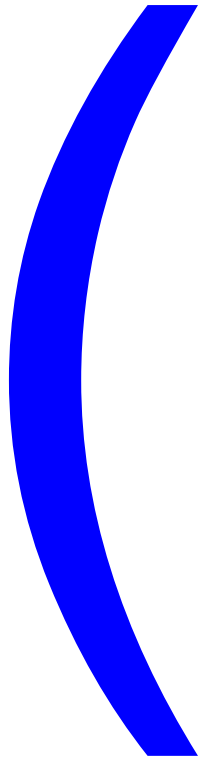
```
...
/Event/Rec
/Event/Rec/Header
/Event/Rec/Status
/Event/Rec/Track
/Event/Rec/Vertex
/Event/Rec/Vertex/Primary
/Event/Rec/Rich
/Event/Rec/Calo
/Event/Rec/ProtoP
...
```

```
>>> evt['Rec/Vertex/Primary'][0]
{ position :      (-0.00195153,0.0830339,-61.8848)
  covMatrix :      [  0.000513381-4.41126e-05  0.00189939
                    -4.41126e-05  0.000114097-0.000206983
                    0.00189939-0.000206983  0.00906074 ]
  chi2 :      0
  nDoF :      25
  extraInfo :
    {{ technique : Unknown }}
```

▶ **dump all leaves:** `>>> evt.dumpAll()`

▶ **forcing reading all below Rec:**

```
>>> allRec = gaudigadgets.nodes(evt, True, 'Rec')
```



More details on Configuration

```
>>> from GaudiConf.Configuration import LHCbApp
```

```
>>> lhcbApp = LHCbApp()
```

See [LHCbConfig.py](#)

◆ Major difference to opts files: help is available

```
>>> help(lhcbApp)
```

```
| Data descriptors defined here:
```

```
| CondDBtag:    Tag for CondDB. Default as set in DDDDBConf for DataType
```

```
| DDDDBtag:    Tag for DDDDB. Default as set in DDDDBConf for DataType
```

```
| DataType:    Data type, can be ['DC06','2008']. Default '2008'
```

```
| EvtMax:      Maximum number of events to process
```

```
| Monitors:    List of monitors to execute
```

```
| Simulation:  Flag to indicate usage of simulation conditions
```

```
| SkipEvents:  Number of events to skip
```

```
| UseOracle:   Flag to enable Oracle CondDB. Default False (use SQLDDDB)
```

◆ DC06 MC setting: `>>> lhcbApp.DataType = 'DC06'`
`>>> lhcbApp.Simulation = False # SIMCONDB didn't exist yet`

◆ 2009 MC setting: `>>> lhcbApp.DataType = '2008'`
`>>> lhcbApp.Simulation = True`

Database Tags, default tag

- ◆ If no tag is specified, the default tag = most recent tag is used
- ◆ No problem for DC06, since by construction all tags are compatible
- ◆ For 2008 MC, until mass production starts, recent tag might not be compatible with the tag used in the simulation
- ◆ For real data, if reconstruction runs from scratch, default tag is the preferred option. If only a partial re-reconstruction is running, tag should be consistent with previous tag used in the reconstruction.
- ◆ The tag should be provided by the Bookkeeping together with list of files. Not in place yet.
- ◆ The application versions and tags used are available for every event:

```
>>> print evt['MC/Header'], evt['MC/DigiHeader'], evt['Rec/Header']
```



Tags, continued

```
>>> evt['MC/Header']
```

```
randomSeeds : [1, 2, 1501426441, 0]
```

```
applicationName : Gauss, applicationVersion : v35r1
```

```
condDBTags : [(DDDB, head-20081002), (SIMCOND, head-20081002)]
```

```
>>> evt['MC/DigiHeader']
```

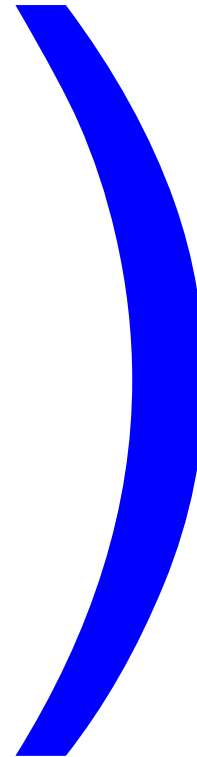
```
applicationName : Boole, applicationVersion : v16r3
```

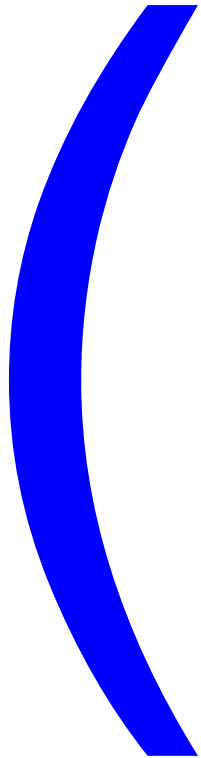
```
condDBTags : [(DDDB, head-20081002), (SIMCOND, head-20081002)]
```

```
>>> evt['Rec/Header']
```

```
applicationName : Brunel, applicationVersion : v34r0
```

```
condDBTags : [(DDDB, head-20081002), (SIMCOND, head-20081002)]
```





And now to something completely different

Getting started: ROOT

python -i RootExa1.py

```
>>> from ROOT import *
```

- ◆ Everything from ROOT will be in the global namespace.
- ◆ Convenient, but not so efficient. Better:

```
>>> from ROOT import TH1F, TBrowser, TCanvas
```

- ◆ Python ROOT and CINT ROOT commands are similar, python is simpler

```
>>> h = TH1F('h_id', ' nr. of tracks', 100, 0., 500.)
```

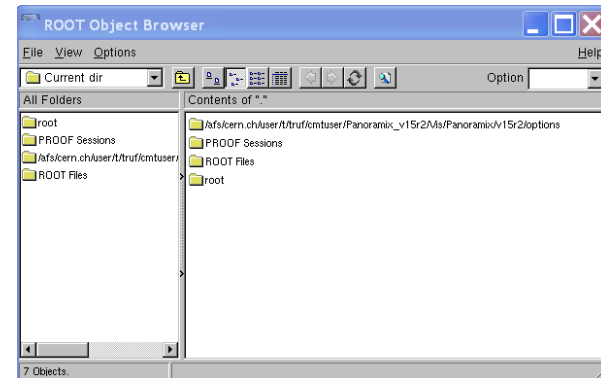
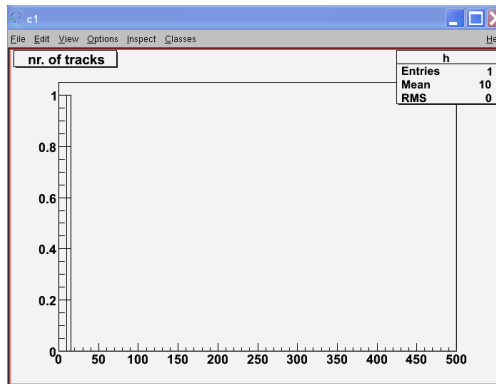
```
>>> dir(h)
```

```
['=', 'AbstractMethod', 'Add', 'AddAt', 'AddBinContent', 'AddDirectory', ... , 'Fill',
 'FillBuffer', 'FillN', 'FillRandom', 'FindBin', 'FindObject', 'Fit', 'FitPanel', ...
```

```
>>> h.Fill(10)
```

```
>>> h.Draw()
```

```
>>> b=TBrowser()
```



Command completion:

h.Dr and Tab

Hbook inspired, filling by identifier:

```
fh = gROOT.FindObjectAny
hstore = [] # you are the owner
# booking
for n in range(21) :
    hstore.append(TH1F('sensor'+str(n), ' ADC counts',100,0.,255.))
...
# filling
rc = fh('sensor'+str(n)).Fill(adc)
...
# saving
f=TFile('histos.root','recreate')
for h in hstore :
    h.Write()
f.Close()
# retrieving
f=TFile('histos.root')
fh = f.FindObjectAny
# drawing
fh('sensor'+str(10)).Draw()
```

Access to doxygen

Module with some helper methods

```
>>> from gaudiwidgets import doxygen
```

```
▶ h = TH1F('h', ...
```

```
>>> cl = GaudiPython.gbl.LHCb.VeloCluster()
```

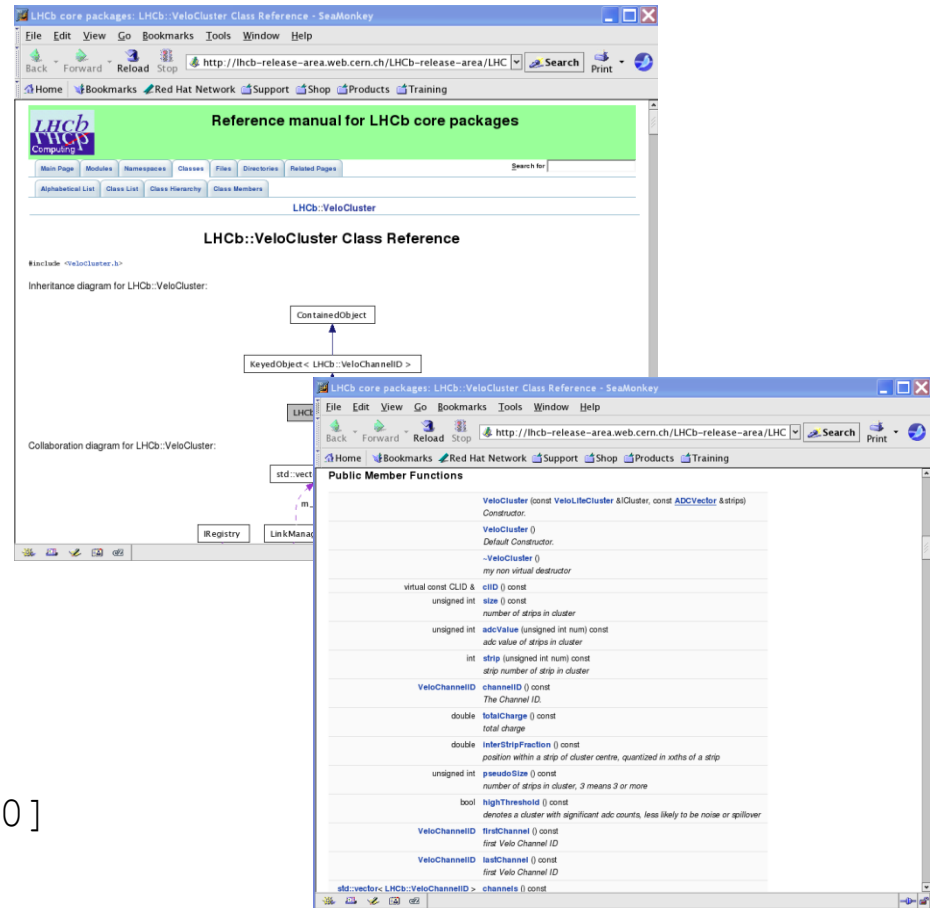
```
>>> doxygen(cl)
```

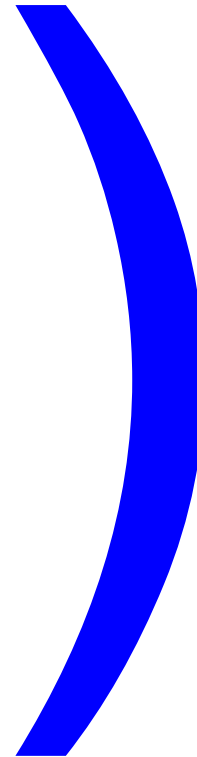
```
>>> doxygen(h)
```



```
>>> pv = evt['Rec/Vertex/Primary'][0]
```

```
>>> doxygen(pv)
```





```
>>> h = TH1F('h', ' nr. of tracks', 100, 0., 500.)
```

■ Containers

```
>>> tc = evt['Rec/Track/Best']
```

```
>>> result = h.Fill(tc.size())
```

■ List of objects

```
>>> for t in tc :
    print t
    result = hp.Fill(t.p())
```

```
>>> dir(tc[3])
```

```
[..., 'addInfo', 'addToAncestors', 'addToLhcbIDs', 'addToMeasurements', 'addToNodes',
'addToStates', 'ancestors', 'charge', 'checkFitHistory', 'checkFitStatus', 'checkFlag',
'checkHistory', 'checkPatRecStatus', 'checkType', 'chi2', 'chi2PerDoF', ..., 'momentum',
'nDoF', 'nLHCbIDs', 'nMeasurements', 'nMeasurementsRemoved', 'nStates', 'nodes', 'p', ...,
'stateAt', 'states', 'type', 'typeBits', 'typeMask']
```

```
# root has to come first, otherwise set to batch mode by Gaudi
▶ from ROOT import TH1F
# get the basic configuration from here
▶ from LHCBConfig import *
▶ lhcbApp.DataType = "DC06"
▶ ApplicationMgr( OutputLevel = INFO )
▶ import GaudiPython
▶ appMgr = GaudiPython.AppMgr()
▶ sel = appMgr.evtsel()
▶ sel.open(['PFN:$AFSROOT/cern.ch/lhcb/group/tracking/vol1/00001378_00000002_5.dst'])
▶ evt = appMgr.evtsvc()
▶ h = TH1F('h',' nr. of tracks',100,0.,500.)
▶ hp = TH1F('hp',' momentum of tracks',100,0.,50000.)
▶ for n in range(100) :
▶     appMgr.run(1)
▶     tc = evt['Rec/Track/Best']
▶     result = h.Fill(tc.size())
▶     for t in tc :
▶         result = hp.Fill(t.p())
▶ h.Draw()
▶ hp.draw()
```

Unpacking MC data

```
▶ appConf = ApplicationMgr( OutputLevel = INFO, AppName = 'Ex3' )
  appConf.TopAlg += ["UnpackMCParticle", "UnpackMCVertex"])
```

◆ ApplicationMgr is the configurable of the Application manager.

◆ What to configure: `help(appConf)` or `appConf.properties().keys()`

```
▶ appMgr = GaudiPython.AppMgr()
▶ sel     = appMgr.evtsel()
▶ sel.open(['PFN:$AFSROOT/cern.ch/lhcb/group/tracking/vol1/00001378_00000002_5.dst'])
▶ evt = appMgr.evtsvc()
▶ appMgr.run(1)
▶ evt.dump()
```

...

`/Event/MC/Particles`

`/Event/MC/Vertices`

...

More elegant: `python -i Ex3b.py`

```
▶ appConf.ExtSvc += ['DataOnDemandSvc']
▶ DataOnDemandSvc().Algorithms +=
  ["DATA='/Event/MC/Particles' TYPE='UnpackMCParticle'"]
▶ DataOnDemandSvc().Algorithms +=
  ["DATA='/Event/MC/Vertices' TYPE='UnpackMCVertex'"]
▶ By default in LHCbConf.py
```

- ◆ Done by data on demand service.
 - ◆ Configured in LHCbConf.py and \$STDOPTS/DecodeRawEvent.py
- ```

▶ # get the basic configuration from here
▶ from LHCbConfig import *
▶ lhcbApp.DataType = "DC06"
▶ appConf = ApplicationMgr(OutputLevel = INFO, AppName = 'Ex3c')
▶ import GaudiPython
▶ appMgr = GaudiPython.AppMgr()
▶ sel = appMgr.evtsel()
▶ sel.open(['PFN:$AFSROOT/cern.ch/lhcb/group/tracking/vol1/00001378_00000002_5.dst'])
▶ evt = appMgr.evtsvc()
▶ appMgr.run(1)
▶ veloClusters = evt['Raw/Velo/Clusters']
▶ print 'size of VeloCluster container:', veloClusters.size()
▶ ItClusters = evt['Raw/IT/Clusters']
▶ print 'size of ItCluster container:', ItClusters.size()
▶ Muoncoords = evt['Raw/Muon/Coords']
▶ print 'size of Muon container:', Muoncoords.size()

```

- ▶ `MCParticles = evt['MC/Particles']`
- ▶ `#first particle`  
`mcp = MCParticles.containedObjects()[0]`  
`# particle with key 0`  
`mcp = MCParticles[0]`
- ▶ `veloClusters = evt['Raw/Velo/Clusters']`

(The key for `VeloClusters` is a `VeloChannelID` object, therefore direct access by “index” , `evt['Raw/Velo/Clusters'][0]` , does not work.)

- ▶ `aVeloCluster = veloClusters.containedObjects()[0]`
- ▶ `doxygen(aVeloCluster)`

# Another example: rewind()

python -i Ex3d.py

```

▶ from gaudigadgets import panorewind
▶ hpt = TH1F('hpt','pt of bees',100,0.,10000.)
▶ for n in range(100) :
▶ appMgr.run(1)
▶ mc = evt['MC/Particles']
▶ for mcp in mc :
▶ if mcp.particleID().hasBottom() :
▶ result = hpt.Fill(mcp.pt())
▶ hpt.Draw() # plot Root histogram
▶ panorewind() # start again from first event

▶ hpt2 = TH1F('hpt2','pt of charm',100,0.,10000.)
▶ for n in range(100) :
▶ appMgr.run(1)
▶ mc = evt['MC/Particles']
▶ for mcp in mc :
▶ if mcp.particleID().hasCharm() :
▶ result = hpt2.Fill(mcp.pt())
▶ hpt2.SetLineColor(2)
▶ hpt2.Draw('same')

```

- ◆ This worked in the past with `sel.rewind()`. After changes to Gaudi state machine, requires some tricks.

## ■ More object oriented approach:

```

appConf.HistogramPersistency = "ROOT"
HistogramPersistencySvc.OutputFile = "Ex3e.root"
import GaudiPython
class MyAlg(GaudiPython.PyAlgorithm):
 def execute(self):
 evh = evt['Rec/Header']
 mcps = evt['MC/Particles']
 print 'event # = ', evh.evtNumber()
 h1.fill(mcps.size())
 return True

...
hist = appMgr.histsvc()
h1 = hist.book('h1', '# of MCParticles', 40, 0, 5000)
appMgr.addAlgorithm(MyAlg())
appMgr.run(10)

```

Books AIDA histogram

In case somebody really wants HBOOK output:

```

appConf.HistogramPersistency = "HBOOK"
HistogramPersistencySvc.OutputFile = "Ex3e.hbook"

```

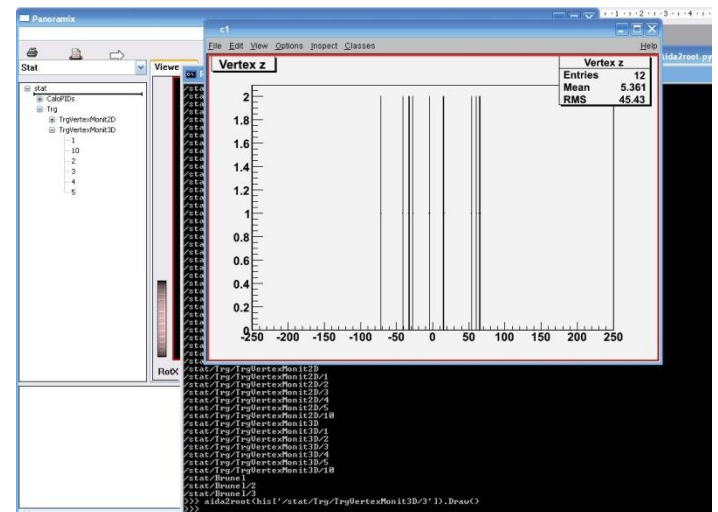
```
python -i Ex3e.py
```

- Histogram is “inside” Gaudi defined by AIDA interface

```
>>> hist.dump()
/stat
/stat/h1
>>> hist['h1']
Histogram 1D "# of MCParticles" 40 bins[0.000000,5000.000000]
```

- Get access to histogram

```
>>> aida2root = GaudiPython.gbl.Gaudi.Utils.Aida2ROOT.aida2root
>>> rh1 = aida2root(hist['h1'])
>>> rh1.Draw()
>>> rh1.Fit('gaus')
```



- Setting up environment:      SetupProject Panoramix v16r3

- Be able to read sim, digi, dst, rdst, mdf files.

```

▶ from LHCbConfig import *
▶ lhcbApp.DataType = "DC06"
 appConf = ApplicationMgr(OutputLevel = INFO, AppName = 'myJob')
▶ import GaudiPython
 appMgr = GaudiPython.AppMgr()
 sel = appMgr.evtsel()
 sel.open(['PFN:$AFSROOT/cern.ch/lhcb/group/tracking/vol1/00001378_00000002_5.dst'])
 evt = appMgr.evtsvc()
 appMgr.run(1)

```

- Inspection, help, documentation:

```

▶ aTrack = evt['Rec/Track/Best'][0]
▶ dir(aTrack), help(aTrack), doxygen(aTrack)

```

- Read whole event and dump leaves

```

▶ evt.dumpAll() # requires import gaudigadgets

```

- Book, fill and draw histograms

```

▶ from ROOT import *
▶ hpt = TH1F('hpt','pt of bees',100,0.,10000.)
▶ hpt.Fill(pt)
▶ hpt.Draw()

```

# Access to Geometry, Detector Elements, Conditions

```
python -i Ex4.py
```

```
▶ from LHCbConfig import *
▶ lhcbApp.DataType = "2008"
 lhcbApp.Simulation = False # default, uses LHCBCONDB
 # lhcbApp.Simulation = True # uses SIMCONDB, didn't existed for DC06
```

## ■ Velo example

```
◆ velo = det['/dd/Structure/LHCb/BeforeMagnetRegion/Velo']
```

```
>>> rSensor_10 = velo.rSensor(10)
```

```
>>> rSensor_10.rMax(0)
```

```
42.0
```

```
>>> rSensor_10.rMin(0)
```

```
8.169999999999999
```

```
>>> rSensor_10.rOfStrip(145)
```

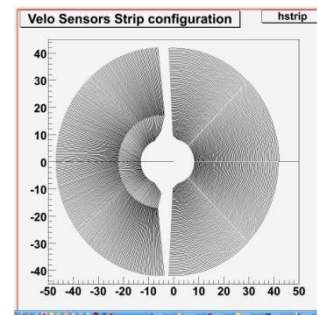
```
14.82897361256984
```

```
>>> ac = velo.geometry.alignmentCondition
```

```
>>> print ac.printParams()
```

```
(St6vectorIdSaIdEE) dPosXYZ = 0 0 0
```

```
(St6vectorIdSaIdEE) dRotXYZ = 0 0 0
```



## ■ More sophisticate example:

```
◆ python -i Velodet06.py
```

[http://lhcb-reconstruction.web.cern.ch/lhcb-reconstruction/Software/python\\_for\\_cyclists.htm#Det example](http://lhcb-reconstruction.web.cern.ch/lhcb-reconstruction/Software/python_for_cyclists.htm#Det%20example)

# Creating Objects, Examples

```
python -i Ex5.py
```

## ■ Useful abbreviation for classes:

- ▶ `XYZPoint` = `GaudiPython.gbl.ROOT.Math.XYZPoint`
- ▶ `XYZVector` = `GaudiPython.gbl.ROOT.Math.XYZVector`

## ■ Instantiation of the objects:

- ▶ `aPoint = XYZPoint(0.,0.,10.)`
- ▶ `aVector = XYZVector(1.,1.,5.)`
- ▶ `print aVector.x(), aVector.y(), aVector.z()`

You can also do in existing python session:

```
from Ex5 import *
```

In case you changed the script later on

```
Import sys
reload(sys.modules['Ex5'])
```

## ■ Principle

- ▶ `tsvc = appMgr.toolsvc()`
- ▶ `aTool = tsvc.create('name_of_tool', interface='Interface_of_tool')`

## ■ Ex6.py:

- ▶ `velopotool = tsvc.create('VeloClusterPosition', interface='IVeloClusterPosition')`
- ▶ `aCluster = evt['Raw/Velo/Clusters'].containedObjects()[0]`

(The key for VeloClusters is a VeloChannelID object, therefore direct access by “index”, `evt['Raw/Velo/Clusters'][0]`, does not work.)

```
>>> clusInfo = velopotool.position(aCluster)
>>> clusInfo.fractionalError
0.17767
```

## ■ Ex6b.py:

◆ Calculating MS of RF shield: `python -i checkGEANT4_MS.py`  
[http://lhcb-reconstruction.web.cern.ch/lhcb-reconstruction/Python/CheckGeant4/some\\_geant4\\_checks.htm](http://lhcb-reconstruction.web.cern.ch/lhcb-reconstruction/Python/CheckGeant4/some_geant4_checks.htm)

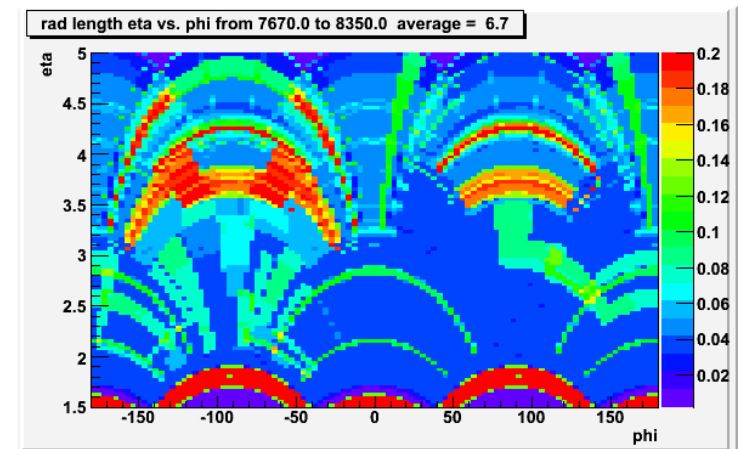
```

▶ extrap = appMgr.toolsvc().create('TrackMasterExtrapolator',
 interface='ITrackExtrapolator')
▶ State = GaudiPython.gbl.LHCb.State
▶ s_origin = State()
 setState(...)
 | void State(const Gaudi::TrackVector& state)
 | void State(double x, double y, double z, double tx, double ty, double qOverP)
▶ # create a state ot origin with slow tx=0.1, ty=-0.1 and p = 5000 MeV/c
▶ s_origin.setState(0.,0.,0.,0.1,-0.1,1./5000.)
▶ s_extrap = s_origin.clone()
▶ # extrapolate to z=12m
▶ result = extrap.propagate(s_extrap,12000.)
▶ print 'extrapolated state: x,y,z = %8.2f, %8.2f, %8.2f'%(s_extrap.x(),s_extrap.y(),s_extrap.z())
▶ print 'extrapolated state: tx,ty,p = %8.2f, %8.2f, %8.2f'%(s_extrap.tx(),s_extrap.ty(),s_extrap.p())
>>> extrapolated state: x,y,z = 3048.075, -1213.064, 12000.000
>>> extrapolated state: tx,ty,p = 0.378, -0.100, 4971.968

```

## ■ Ex7.py:

- ▶ `appConf.ExtSvc += ['TransportSvc']`
- ▶ `appMgr = GaudiPython.AppMgr()`
- ▶ `appMgr.initialize()`
- ▶ `tranSvc = appMgr.service('TransportSvc', 'ITransportSvc')`
- ▶ `a = XYZPoint(0,0,0)`
- ▶ `b = XYZPoint(500.,350.,2000.)`
- ▶ `radlength = tranSvc.distanceInRadUnits(a,b)`
- ▶ See also: `python -i rad_map06.py` and [http://lhcb-reconstruction.web.cern.ch/lhcb-reconstruction/Python/Radiation\\_and\\_Bfield\\_maps.htm](http://lhcb-reconstruction.web.cern.ch/lhcb-reconstruction/Python/Radiation_and_Bfield_maps.htm)



## ■ Ex7b.py:

```
ParticlePropertySvc setup in LHCbConf.py
>>> appMgr.initialize()
>>> partSvc = appMgr.ppSvc()
>>> partSvc.find(ParticleID(521)).mass()
5279.0
>>> partSvc.find(ParticleID(521)).lifetime()
0.001671
>>> partSvc.find(ParticleID(521)).name()
'B+'
>>> print partSvc.find(ParticleID(521)) †
B+ PDG: 521, Q: +1, mass: 5.2791 GeV, ctau: 491.06 um
```

† only available with `import PartProp.decorators`, done in `LHCbConfig.py`

## ■ Ex7c.py:

```

▶ appConf.ExtSvc += ['MagneticFieldSvc']
▶ magSvc = appMgr.service('MagneticFieldSvc','IMagneticFieldSvc')
▶ a = XYZPoint(0.,0.,5000.)
▶ v = XYZVector()
▶ magSvc.fieldVector(a,v)
>>> v.y()
-0.00103710293

▶ bintegrator =
 appMgr.toolsvc().create('BIntegrator',interface='IBIntegrator')
▶ Bdl = XYZVector()
▶ def Double(d) : return d + 0.0 # this is a little trick
▶ zCenter = Double(0.)
▶ b = XYZPoint(0.,0.,2500.)
▶ result = bintegrator.calculateBdlAndCenter(a,b,0.,0.,zCenter,Bdl)
>>> Bdl.x()
-0.0020624343736251353

```

```

▶ from LinkerInstances.eventassoc import *
▶ MCParticle = GaudiPython.gbl.LHCb.MCParticle
▶ Track = GaudiPython.gbl.LHCb.Track
▶ ... Following for every event
▶ #####MC relation
▶ ltrack2part = linkedTo(MCParticle,Track,'Rec/Track/Best')
▶ lpart2track = linkedFrom(Track,MCParticle,'Rec/Track/Best')
▶ t = evt['Rec/Track/Best'][0]
▶ for mcp in ltrack2part.range(t) :
 print mcp
 ▶ { momentum : (923.68,-113.08,3783.46,3898.72)
 particleID : { pid : 211
▶ for tt in lpart2track.range(mcp) :
 print tt.key()
 ▶ 0
 15 #MCParticle is reconstructed twice

```

Quite useful for studying data sizes of L1 boards, see also [Raw Data sizes](#)

## Examples:

```

▶ from gaudigadgets import getEnumNames
▶ ...
▶ Enums = getEnumNames('LHCb::RawBank')
▶ BankType = Enums['BankType']
▶ f = open('raw_buffer.txt','w')
▶ rb = evt['DAQ/RawEvent']
▶ f.write(' banktype sourceID word hex string \n')
▶ i=0
▶ for k in BankType.keys() :
▶ b = BankType[k]
▶ nTells = rb.banks(k).size()
▶ for m in range(nTells) :
▶ size = int((rb.banks(k)[m].size()) / 4 + 0.5)
▶ for l in range(size) :
▶ word = rb.banks(k)[m].data()[l]
▶ t2 = (word >> 16) & 0x0000ffff
▶ t1 = word & 0x0000ffff
▶ if word < 0:
▶ word = t2 * 2 ** 16 + t1
▶ f.write('%15s %6d %8x %10s '
▶ '%(b+' + str(rb.banks(k)[m].sourceID()), i, word, tobin(word)))
▶ f.write('\n')
▶ i += 1
▶ f.close()
▶ webbrowser.open(os.getcwd() + '/raw_buffer.txt')

```

| banktype | sourceID | word | hex      | string                           |
|----------|----------|------|----------|----------------------------------|
| L0Calo   | 0        | 0    | 0019c52c | 00000000000110011100010100101100 |
| L0Calo   | 0        | 1    | 00144d03 | 00000000000101000100110100000011 |
| L0Calo   | 0        | 2    | 00289609 | 00000000001010001001011000001001 |
| L0Calo   | 0        | 3    | 002bd213 | 0000000000101011101001000010011  |
| L0Calo   | 0        | 4    | 01137410 | 00000001000100110111010000010000 |
| ...      |          |      |          |                                  |
| Velo     | 7        | 212  | 95e29564 | 10010101111000101001010101100100 |
| Velo     | 7        | 213  | ded09a8b | 11011110110100001001101010001011 |
| Velo     | 7        | 214  | a158a10e | 10100001010110001010000100001110 |
| Velo     | 7        | 215  | afc42cf0 | 1010111110001000010110011110000  |
| Velo     | 8        | 224  | 00000015 | 0000000000000000000000000010101  |
| Velo     | 8        | 225  | 806d8053 | 10000000011011011000000001010011 |
| Velo     | 8        | 226  | 862bc57f | 1000011000101011100010101111111  |
| Velo     | 8        | 227  | 8b0b86ac | 10001011000010111000011010101100 |
| Velo     | 8        | 228  | 9183d136 | 10010001100000111101000100110110 |
| ...      |          |      |          |                                  |
| Rich     | 3        | 1653 | 31142e20 | 00110001000101000010111000100000 |
| Rich     | 3        | 1654 | 44803508 | 01000100100000000011010100001000 |
| Rich     | 3        | 1655 | 00004810 | 00000000000000000010010000001000 |
| Rich     | 3        | 1656 | c0011401 | 11000000000000010001010000000001 |
| Rich     | 3        | 1657 | 00002040 | 00000000000000000001000000100000 |
| Rich     | 3        | 1658 | c0024005 | 1100000000000100100000000000101  |
| Rich     | 3        | 1659 | 51024101 | 01010001000000100100000100000000 |

## ■ Calorimeter example:

```

▶ from LHCbConfig import *
▶ lhcbApp.DataType = "2008"

▶ appConf = ApplicationMgr(OutputLevel = INFO)
▶ EventSelector(Input = ["DATA=
'castor:/castor/cern.ch/grid/lhcb/data/2008/RAW/LHCb/PHYSICS/24080/024080_0000054568.raw'
SVC='LHCb::MDFSelector'"])

▶ import GaudiPython
▶ appMgr = GaudiPython.AppMgr()
▶ evt = appMgr.evtsvc()
▶ appMgr.run(1)
▶ print '+++++', evt['Raw/Hcal/Digits'].size()

```

## ■ Run PatSeeding: `python -i Ex10b.py`

# Hitmaps, Landau Distribution

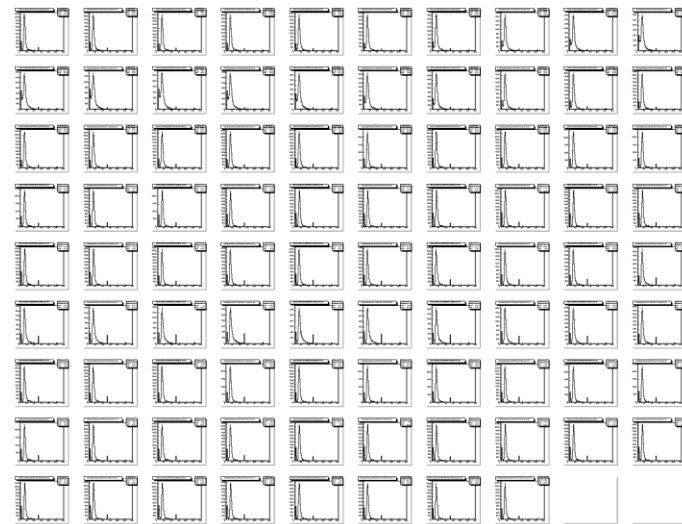
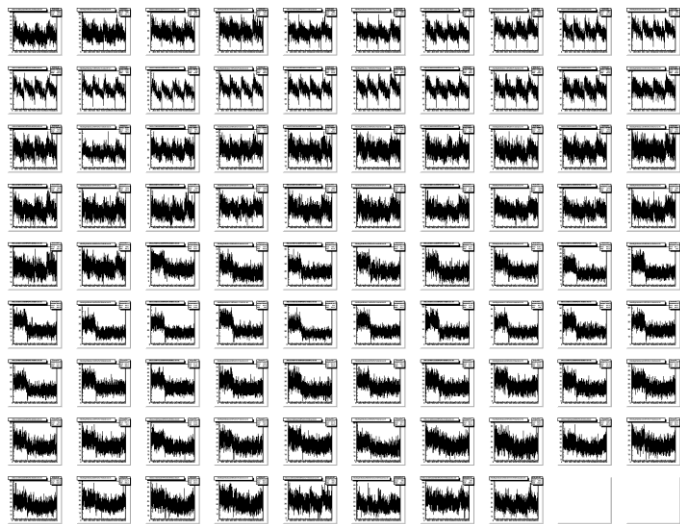
```

▶ class myAlg(GaudiPython.PyAlgorithm):
▶ def execute(self):
▶ vc = evt['Raw/Velo/Clusters']
▶ for cl in vc :
▶ sensorNr = cl.channelID().sensor()
▶ h = hlist[sensorNr]
▶ hl = hlandau[sensorNr]
▶ for i in range(cl.size()) :
▶ success = h.Fill(cl.strip(i))
▶ success = hl.Fill(cl.totalCharge())
▶ return True

```

python -i VeloCluster.py

See also [http://lhcb-reconstruction.web.cern.ch/lhcb-reconstruction/Software/python\\_for\\_cyclists.htm](http://lhcb-reconstruction.web.cern.ch/lhcb-reconstruction/Software/python_for_cyclists.htm)



# Another colourful Example

## ■ Origin of secondaries,

`python -i readSIM_MCHits.py`

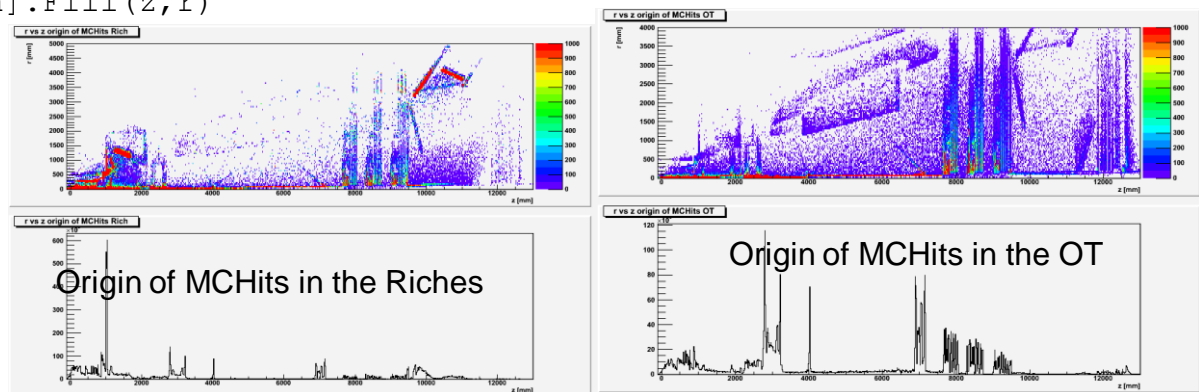
see <http://lhcb-reconstruction.web.cern.ch/lhcb-reconstruction/Python/PlotsOfsecondaries.htm>

```

▶ hlist = {}
▶ hlist['MC/IT/Hits'] = TH2F('horigin_IT', 'r vs z origin of MCHits IT'
,1000,-100.,13000. ,100,0.,2000.)
▶ hlist['MC/TT/Hits'] = TH2F('horigin_TT', 'r vs z origin of MCHits TT' ,
500,-100., 4000. ,100,0.,2000.)
...
▶ for h in hlist :
▶ mch = evt[h]
▶ for ahit in mch :
▶ pos = ahit.mcParticle().originVertex().position()
▶ r = pos.rho()
▶ z = pos.z()
▶ if r>3. :
▶ result = hlist[h].Fill(z,r)

```

That is all.  
The rest is ROOT presentation.



## Ex11.py:

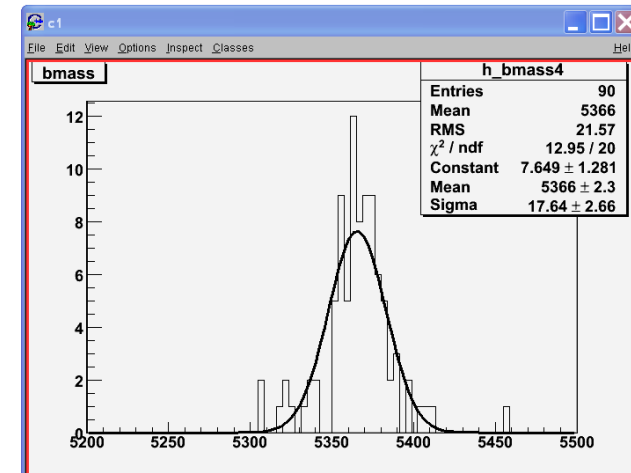
```

▶ from LHCbConfig import *
▶ lhcbApp.DataType = "2008"
▶ lhcbApp.Simulation = True
▶ importOptions('$PANORAMIXROOT/options/Panoramix_DaVinci.py')
▶ importOptions('$CCBARROOT/options/DoDC06SelBs2Jpsi2MuMuPhi2KK_lifetime_unbiased.opts')
▶ appConf = ApplicationMgr(OutputLevel = INFO, AppName = 'Ex11')
▶ ...
▶ sel.open(['PFN:castor:/castor/cern.ch/user/t/truf/MC2008/00003402_121314_s.dst'])
▶ ...
▶ h_bmass = TH1F('h_bmass','Mass of B candidate',100,5200.,5500.)
▶ while 0 < 1:
▶ appMgr.run(1)
▶ # check if there are still valid events
▶ if evt['Rec/Header'] == None : break
▶ cont = evt['Phys/DC06selBs2Jpsi2MuMu_Phi2KK/Particles']
▶ if cont != None :
▶ for b in cont :
▶ success = h_bmass.Fill(b.momentum().mass())
▶ h_bmass.Draw()

```

All you need for a mass plot

Starting again: >>> gaudigadgets.panorewind()

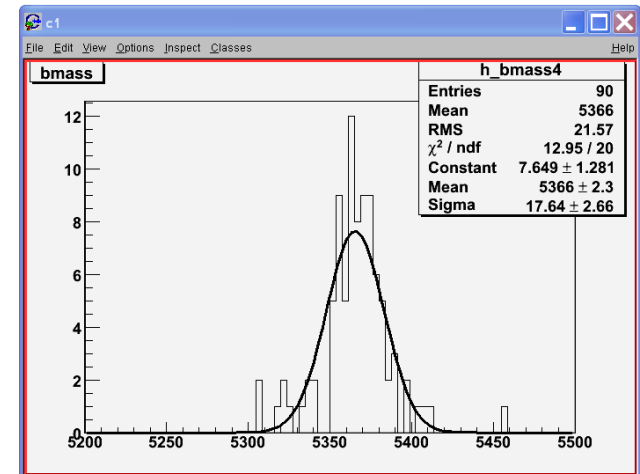


## Ex17b.py:

```

▶ appConf.ExtSvc += ['TagCollectionSvc/EvtTupleSvc']
▶ import GaudiKernel.SystemOfUnits as units
▶ file = 'PFN:castor:/castor/cern.ch/user/l/lshchuts/stripping/SETC_newDST_v31/1.root'
▶ EventSelector(Input = ["COLLECTION=\ 'TagCreator/1\ ' DATAFILE='%s' TYP='POOL_ROOT'
 SEL='(PreselHeavyDimuon>0)'"%file])
▶ FileCatalog().Catalogs = ["xmlcatalog_file:newDST_v31.xml"]
▶
▶
▶ h_bmass = TH1F('h_bmass','Mass of B candidate',100,5200.,5500.)
▶ def my_loop():
▶ while 0<1 :
▶ appMgr.run(1)
▶ if evt['Rec/Header'] == None : break
▶ mybees = evt['Phys/Phys/PreselHeavyDimuon/Particles']
▶ if mybees != None :
▶ for b in mybees :
▶ rc=h_bmass.Fill(b.measuredMass()/units.GeV)
▶ my_loop()
▶ h_bmass.Draw()

```



# Creating a microDST

## ► Basic configuration of the script:

- selection\_name = 'DC06selBs2JpsiPhi\_unbiased'
- saveMC = True
- sequence = 'Seq'+selection\_name

python microDSTEx0.py

## ► Configure microDST algorithms:

- from Configurables import CopyRecHeader, CopyMCParticles, CopyParticles, CopyPrimaryVertices, CopyParticle2PVLINK, CopyRelatedMCParticles, CopyFlavourTag, CopyODIN, ParticleCloner, MCParticleCloner
- copyParticles = CopyParticles()
- copyParticles.InputLocation = "Phys/"+selection\_name+"/Particles"

# Copy related MCParticles and also associations between

- copyMC = CopyRelatedMCParticles()
- copyMC.InputLocation = copyParticles.InputLocation
- copyMC.ICloneMCParticle = "MCParticleCloner"
- copyMC.addTool(MCParticleCloner,name="MCParticleCloner")
- copyMC.MCParticleCloner.ICloneMCVertex = "MCVertexCloner "

# add to sequence

- seq = GaudiSequencer(sequence)
- seq.Members += [copyParticles,copyMC,CopyPrimaryVertices(),CopyRecHeader(),CopyODIN()]

## ► Define outputstream

- microDSTStream = OutputStream('MicroDSTStream')
- microDSTStream.ItemList = ["/Event/microDST#99"]
- fname = "DATAFILE='PFN:'+output\_name+' TYP='POOL\_ROOTTREE' OPT='REC'"
- microDSTStream.Output = fname
- ApplicationMgr().OutStream.append(microDSTStream)

# Creating a microDST, cont.

## ► Execution:

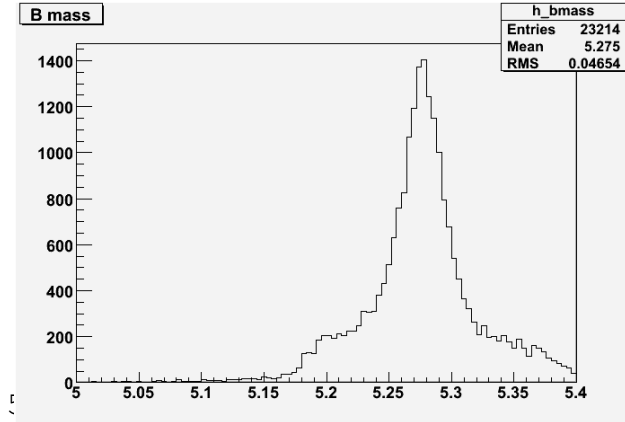
- # keep control of event loop
- appMgr.algorithm('MicroDSTStream').Enable = False
- Nevents = 0
- while 1>0 :
- appMgr.run(1)
- if evt['DAQ/ODIN']==None : break
- if evt['Phys/DC06selBs2JpsiPhi\_unbiased/Particles'] != None :
- if evt['Phys/DC06selBs2JpsiPhi\_unbiased/Particles'].size() > 0 :
- # copy selected event to output file
- appMgr.algorithm('MicroDSTStream').execute()
- Nevents += 1
- print "Number of selected events:", Nevents

## microDSTEx1.py:

```

▶ from LHCbConfig import *
▶ lhcbApp.DataType = "DC06"
▶ appConf = ApplicationMgr(OutputLevel = INFO)
▶ EventSelector().PrintFreq = 10000
▶ import GaudiPython
▶ import GaudiKernel.SystemOfUnits as units
▶ locationRoot = '/Event/microDST/'
▶ h_bmass = TH1F('h_bmass','Mass of B candidate',100,5200.,5
▶ def my_loop():
▶ while 0<1 :
▶ appMgr.run(1)
▶ if evt[locationRoot+'Rec/Header'] == None : break
▶ mybees = evt[locationRoot+'Phys/DC06selBs2JpsiPhi_unbiased/Particles']
▶ if mybees != None :
▶ for b in mybees :
▶ rc=h_bmass.Fill(b.measuredMass()/units.GeV)
▶ import time
▶ print time.clock()
▶ my_loop()
▶ print time.clock()

```



All you need for a mass plot

16k events, 6-8 sec on lxplus



# Copying Events

## ◆ Write a few events to a separate file

```

▶ importOptions(...)
▶ InputCopyStream().Output = "DATAFILE='PFN:myEvents.dst' TYP='POOL_ROOTTREE' OPT='REC' "
▶ appConf = ApplicationMgr(
 OutputLevel = INFO, AppName = 'Ex13', OutStream = [InputCopyStream()])
▶ import GaudiPython
▶ appMgr = GaudiPython.AppMgr()
▶ sel = appMgr.evtsel()
▶ sel.open(['PFN:$AFSROOT/cern.ch/lhcb/group/tracking/vol1/Bsjpsiphi_00001620_00000004_5.dst'])
▶ evt = appMgr.evtsvc()
▶ appMgr.algorithm('InputCopyStream').Enable = False

▶ while 1>0 :
▶ appMgr.run(1)
▶ if evt['Rec/Header'] == None : break
▶ cont = evt['Phys/DC06selBs2Jpsi2MuMu_Phi2KK/Particles']
▶ if cont != None :
▶ out = False
▶ for b in cont :
▶ if b.pt() > 10000. : out = True
▶ if out : appMgr.algorithm('InputCopyStream').execute()

```

# L0 Yes, Raw data only

python -i Ex13b.py

```

▶ ...
▶ importOptions('$L0DUOPTS/L0DUFromRaw.opts')
▶ from Configurables import EventNodeKiller
▶ rawwriter = OutputStream('RawWriter', Preload = False,
 ItemList = ["/Event#1", "/Event/DAQ#1", "/Event/DAQ/RawEvent#1", "/Event/DAQ/ODIN#1"],
 Output = "DATAFILE='PFN:L0yes.raw' TYP='POOL_ROOTTREE' OPT='REC' ")
▶ appConf = ApplicationMgr(OutputLevel = INFO, AppName = 'Ex13b')
▶ appConf.TopAlg = ['EventNodeKiller/EventNodeKiller', 'GaudiSequencer/L0FromRaw']
▶ appConf.OutStream = [rawwriter]
▶ EventNodeKiller().Nodes =
 ["Rec", "Trig", "MC", "Raw", "Gen", "Link", "pSim", "Prev", "PrevPrev", "Next"]
▶ EventSelector().PrintFreq = 50

▶ import GaudiPython
▶ appMgr = GaudiPython.AppMgr()
▶ sel = appMgr.evtsel()
▶ sel.open(['PFN:D:/LHCb_data/Bsjpsiphi_00001620_00000004_5.dst'])
▶ evt = appMgr.evtsvc()
▶ appMgr.algorithm('RawWriter').Enable = False # stop automatic execution of RawWriter

```

# L0 Yes, Raw data only cont.

python -i Ex13b.py

```
▶ while 1>0 :
▶ appMgr.run(1)
▶ # check L0
▶ L0dir = evt['Trig/L0/L0DU']
▶ if L0dir == None : break # probably end of input
▶ if L0dir.decision() > 0 :
▶ rc = appMgr.algorithm('RawWriter').execute() # output event
```

Verify that file makes sense:

python -i Ex13c.py

```
▶ import gaudigadgets
▶ sel.open(['PFN:L0yes.raw'])
▶ evt = appMgr.evtsvc()
▶ appMgr.run(1)
▶ evt.dumpAll()
```

```

▶ ...
▶ # Default options to rerun L0 and run Hlt
▶ importOptions ('$L0DURROOT/options/ReplaceL0BanksWithEmulated.opts')
▶ from HltConf.Configuration import *
▶ hltconf = HltConf()
▶ hltconf.oldStyle = False
▶ hltconf.hltType = 'PA+LU+VE+MU+HA+PH+EL'
▶
▶ appMgr = GaudiPython.AppMgr(o)
▶ sel = appMgr.evtsel()
▶ sel.open(['PFN:$AFSROOT/cern.ch/lhcb/group/tracking/vol1/00001378_00000002_5.dst'])
▶ evt = appMgr.evtsvc()
▶ hist = appMgr.histSvc()
▶ appMgr.run(1)

▶ import hltexamples
▶ # nice summary printouts per event
▶ hltexamples.l0()
▶ hltexamples.l0_values()
▶ hltexamples.l0_candidates()
▶ hltexamples.hlt_decisions()
▶ hltexamples.hlt_selection_report()

```

▶ evt.dump()

```

/Event/Rec/Trig
/Event/Rec/Trig/Vertex3D
/Event/Hlt
/Event/Hlt/Track
/Event/Hlt/Track/Forward
/Event/Hlt/Track/MuonForward
/Event/Hlt/Track/MuonVelo
/Event/Hlt/Track/Velo
/Event/Hlt/Track/VeloTT
/Event/Hlt/Vertex
/Event/Hlt/Vertex/Bank
/Event/Hlt/Vertex/DiMuonVer
/Event/Hlt/Vertex/PV2D
/Event/Hlt/Vertex/PreDiMuonVer
/Event/Hlt/Summary
/Event/Trig
/Event/Trig/L0
/Event/Trig/L0/L0DU
/Event/Raw
/Event/Raw/Velo
/Event/Raw/Velo/LiteClusters
/Event/Raw/TT
/Event/Raw/TT/LiteClusters

```

▶ histo\_dump(hist)

```

/stat/Hlt/1 Title: time
/stat/Hlt/2 Title: time0
/stat/Hlt/3 Title: rate
/stat/HltAllAlleys/1 Title: t
/stat/HltAllAlleys/2 Title: t
/stat/HltAllAlleys/3 Title: r
/stat/HltMuonAlley/1 Title: t
/stat/HltMuonAlley/2 Title: t
/stat/HltMuonAlley/3 Title: r
/stat/RealMuonAlley/1 Title:
/stat/RealMuonAlley/2 Title:
/stat/RealMuonAlley/3 Title:
/stat/HltMuonTConfL0SingleAll
/stat/HltMuonTConfL0SingleAll
/stat/HltMuonTConfL0SingleAll
/stat/HltMuonTConfL0DimuonAlley/1 Title: time
/stat/HltMuonTConfL0DimuonAlley/2 Title: time0
/stat/HltMuonTConfL0DimuonAlley/3 Title: rate
/stat/HltHadAlley/1 Title: time
/stat/HltHadAlley/2 Title: time0
/stat/HltHadAlley/3 Title: rat
/stat/HltHadAlley/4 Title: LOH
/stat/HltHadAlley/5 Title: Hlt

```

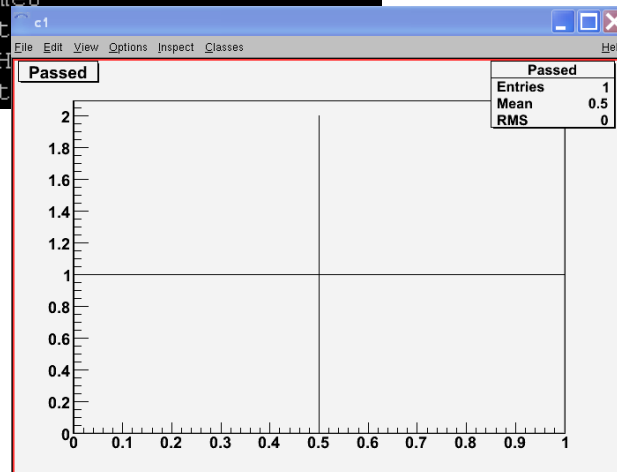
From gaudigadgets

```

def histo_dump(hsvc, node=None) :
 if not node :
 root = hsvc.retrieveObject('')
 if root : node = root.registry()
 else : return
 if node.object() :
 if hsvc.leaves(node).size() > 0 :
 for l in hsvc.leaves(node) :
 histo_dump(hsvc,l)
 else :
 print node.identifier(),'Title:',
 hsvc[node.identifier()].title()

```

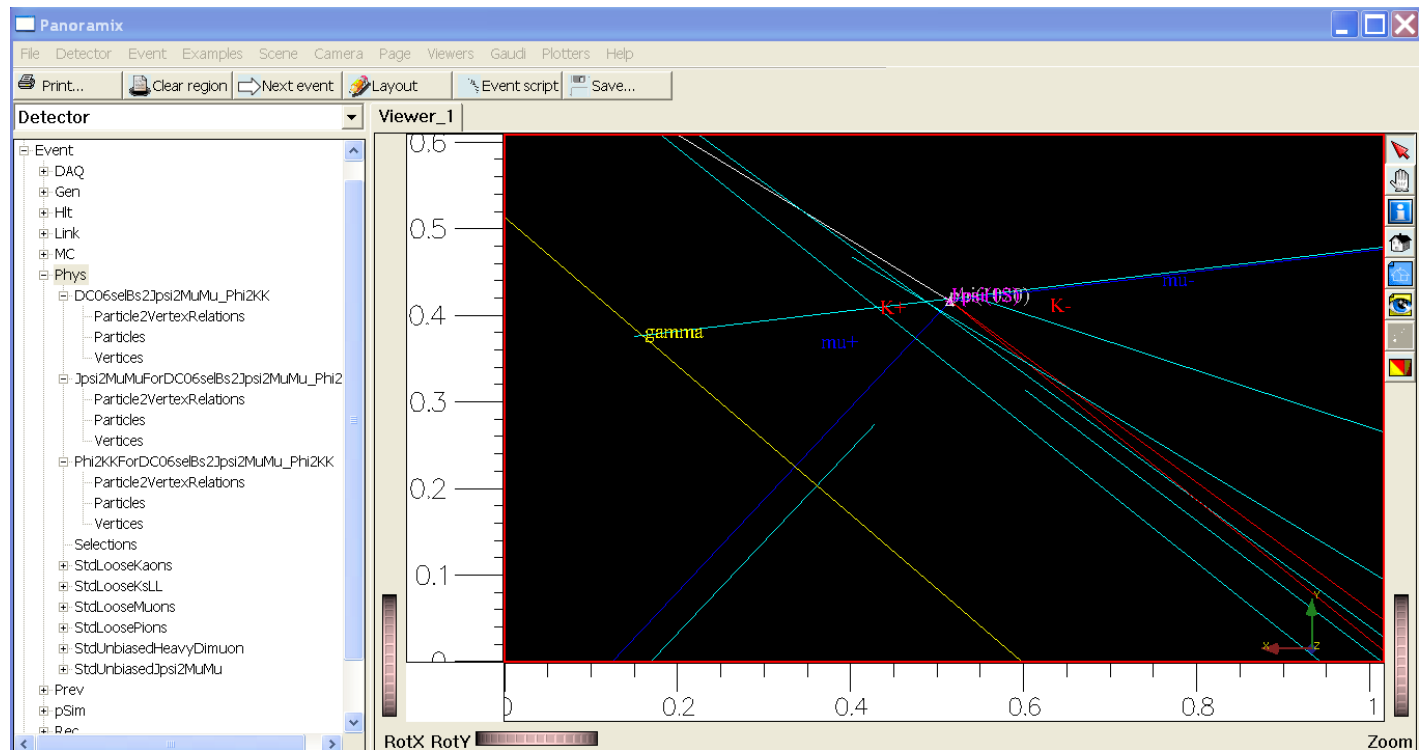
- ▶ appMgr.run(200)
- ▶ aida2root = GaudiPython.gbl.Gaudi.Utils.Aida2ROOT.aida2root
- ▶ aida2root(hist['HltCaloTrigger/1']).Draw()
- ▶ aida2root(hist['HltHadRecoIPVelo/3']).Draw()



python -i Ex15.py

```
▶ ...
▶ importOptions('$PANORAMIXROOT/options/PanoramixVis.py')
```

```
▶ ...
>>> from panoramixmodule import *
>>> toui()
```



- <http://pyprocessing.berlios.de/>
  - ◆ `processing` is a package which supports the spawning of processes using the API of the standard library's threading module. It runs on both Unix and Windows.
- See also presentation: Multiprocessors and GaudiPython by Pere Mato <http://indico.cern.ch/conferenceDisplay.py?confId=25001>
- Future version of GaudiPython should contain a "use processing v\* LCG\_Interfaces", until then use:
  - ◆ `setenv PYTHONPATH /afs/cern.ch/sw/lcg/external/processing/0.51/${CMTCONFIG}/lib/python2.5/site-packages:${PYTHONPATH}`
  - ◆ Or: source `multicore_setup`
- Also useful, `pickleRoot.py`: allows to return root objects

# How to use it

- ◆ GaudiPython configuration step

- ▶ `import GaudiPython`
- ▶ `from pickleROOT import *`
- ▶ `from processing import Pool`

- ◆ Encapsulate execution step in a function

- ▶ `def processFile(file) :`  
     `appMgr = GaudiPython.AppMgr()`  
     `...`  
     `return something (should be pickleable)`

- ▶ `#-----Use up to 8 processes`

- ▶ `pool = Pool(8)`

**Files = list of input files**

- ▶ `result = pool.map_async(processFile, files)`
- ▶ `for r in result.get(timeout=10000) : print r`

## ■ Ex2b\_multicore.py

- ◆ Loop over 16 input files, run DaVinci selection, fill histogram with B mass
- ◆ For each input file, histograms are returned and added up in the main program
- ◆ On Ixbuild, 8 core machine, real time:
 

|                                   | Ixplus                     |
|-----------------------------------|----------------------------|
| ▶ Single process: 258.7 sec       | 264.0 sec                  |
| ▶ 2 processes: 116.1 sec          | 147.2 sec                  |
| ▶ 4 processes: 95.5 sec           | 92.4 sec                   |
| ▶ 8 processes: 94.9 sec           | Configuration dominated !! |
| ▶ 8 processes (32 files): 115 sec | 4 processes: 165.1 sec     |

## ■ Ex3\_multicore.py

- ◆ Loop over n input files, run full reconstruction, fill histogram with track fit chi2
- ◆ On lxbuild, 8 core machine, real time
  - ▶ Single process: 505 sec, 1 file or ~1 s / event
  - ▶ 8 processes: 983 sec, 8 files or ~0.25 s / event

```

lxbuild111.cern.ch - PuTTY
top - 15:58:56 up 20 days, 3:55, 6 users, load average: 8.46, 6.30, 3.15
Tasks: 168 total, 9 running, 159 sleeping, 0 stopped, 0 zombie
Cpu(s): 94.6% us, 0.3% sy, 0.0% ni, 4.8% id, 0.0% wa, 0.0% hi, 0.2% si
Mem: 16414756k total, 11696504k used, 4718252k free, 1227480k buffers
Swap: 8385880k total, 0k used, 8385880k free, 6280468k cached

 PID USER PR NI %CPU TIME+ %MEM VIRT RES SHR S COMMAND
17226 truf 16 0 97 6:31.42 2.6 806m 409m 73m R python
17227 truf 16 0 96 6:27.48 2.6 811m 414m 73m S python
17221 truf 18 0 96 6:36.37 2.4 783m 387m 73m R python
17222 truf 17 0 96 6:29.19 2.6 814m 418m 73m R python
17225 truf 17 0 95 6:23.48 2.6 810m 413m 73m R python
17228 truf 16 0 94 6:29.93 2.4 788m 391m 73m R python
17223 truf 17 0 93 6:34.95 2.7 831m 435m 73m R python
17224 truf 16 0 93 6:34.13 2.6 811m 415m 73m R python
 7796 truf 15 0 1 0:03.15 0.0 40748 2528 1748 S sshd
 1 root 16 0 0 0:01.82 0.0 4756 616 520 S init
 2 root RT 0 0 0:00.42 0.0 0 0 0 S migration/0
 3 root 34 19 0 0:00.47 0.0 0 0 0 S ksoftirqd/0
 4 root RT 0 0 0:00.38 0.0 0 0 0 S migration/1
 5 root 34 19 0 0:00.66 0.0 0 0 0 S ksoftirqd/1
 6 root RT 0 0 0:00.30 0.0 0 0 0 S migration/2
 7 root 34 19 0 0:00.51 0.0 0 0 0 S ksoftirqd/2
 8 root RT 0 0 0:00.27 0.0 0 0 0 S migration/3

```

662x415

105x50

- GaudiPython: access to Gaudi and LHCb software
- ROOT: access to PyROOT
- os, sys : access to operating system
- math: basic math from Python
- array: like python lists, restricted to basic values: characters, integers, floats
- time: access to system clocks
- LinkerInstances.eventassoc: access to LHCb linker classes
- gaudigadgets: some additional features
- panoramixmodule: access to LHCb Event Display

- For more sophisticated analysis, kinematics, combinatorics, etc., look at Bender package  
== mirror of LoKi in Python
- See other Tutorial

### ■ Fill size of container

- ◆ Rec/Track/Best
  - ▶ 57 kB and 14 ms / event
- ◆ Raw/Velo/Cluster
  - ▶ 34 kB and 21 ms / event
- ◆ MC/Particles
  - ▶ 89.4 kB and 45 ms / event

I/O, same for running C++

~ 4 MB / s

~ 2 MB / s, created from Raw

~ 2 MB / s, additional unpacking

### ■ Loop over container, fill one variable

- ◆ Rec/Track/Best
  - ▶ 115 objects and 4.5 ms / event
- ◆ Raw/Velo/Cluster
  - ▶ 1243 objects and 28 ms / event
- ◆ MC/Particles
  - ▶ 2428 objects and 83 ms / event

~30  $\mu$ s / object

### ■ Simple N-tuple

- ◆ MCParticles: mom, oVx, eVX, pid
  - ▶ 2.5 kB, 2428 objects, 0.5ms / event

### ■ Fill size of container

- ◆ Rec/Track/Best
  - ▶ (57 kB) and 4.9 ms / event
- ◆ Raw/Velo/Cluster
  - ▶ (34 kB) and 3.3 ms / event
- ◆ MC/Particles
  - ▶ (89 kB) and 12.7 ms / event

I/O, same for running C++

~ 12 MB / s

~ 10 MB / s, created from Raw

~ 7 MB / s, additional unpacking

### ■ Loop over container, fill one variable

- ◆ Rec/Track/Best
  - ▶ 116 objects and 1 ms / event
- ◆ Raw/Velo/Cluster
  - ▶ 1287 objects and 14 ms / event
- ◆ MC/Particles
  - ▶ 2542 objects and 34 ms / event

~10  $\mu$ s / object

### ■ Simple N-tuple

- ◆ MCParticles: mom, oVx, eVX, pid
  - ▶ 2.5 kB, 2542 objects, 0.4ms / event

- Fill size of container
  - ◆ Phys/.../Particles
    - ▶ (0.5 kB) and 0.36 ms / event
  - ◆ MC/Particles
    - ▶ (0.6 kB) and 0.33 ms / event
  - ◆ Rec/Track/Best
    - ▶ (4.6 kB) and 0.67 ms / event
  
- Loop over container, fill histogram
  - ◆ Phys/.../Particles
    - ▶ 1 object and 0.01 ms / event
    - ▶ Filling 4 histograms with 4 variables, no difference
  - ◆ MC/Particles
    - ▶ 9.7 objects and 0.13 ms / event
  - ◆ Rec/Track/Best
    - ▶ 6 objects and 0.03 ms / event
  
- Example: Bs2JpsiPhi  $\mu$ DST
  - ◆ 8.8 kB/event  $\Rightarrow 10^6$  events = 8.8 GB
  - ◆ one loop over  $10^6$  events: ~7 minutes
  - ◆ MC part: 1.1 kB/event, Tracks: 4.6 kB/event

I/O, same for running C++

~ 1.4 MB / s

~ 1.8 MB / s

~ 6.9 MB / s

~ 6-13  $\mu$ s / object

# List of more advanced example scripts

1. [Ks reconstruction study](http://lhcb-reconstruction.web.cern.ch/lhcb-reconstruction/Python/KsShorts/) `python -i KsCheck.py`
2. [Checking Measurements](http://lhcb-reconstruction.web.cern.ch/lhcb-reconstruction/Python/Tracking/Scripts/checkMeasurements.py)  
`python -i checkMeasurements_pg.py`  
`python -i checkMeasurements_ST.py`
3. [Geant4 multiple scattering](#) `python -i checkGeant4_MS.py`
4. [Make IP and P resolution plots](http://lhcb-reconstruction.web.cern.ch/lhcb-reconstruction/Python/Tracking/Scripts/IPandPresol_plot.py) `python -i IPandPresol_plot.py`
5. [Check Velo Position Tool](http://lhcb-reconstruction.web.cern.ch/lhcb-reconstruction/Python/Tracking/Scripts/checkVeloPoTool.py)
6. [Get events on the grid via Event and Run number](http://lhcb-reconstruction.web.cern.ch/lhcb-reconstruction/Python/Grid.htm)
7. Other inspirations at `$PANORAMIXROOT/Examples/Python`

# List of more advanced example scripts, cont.

## 7. Fake B reconstruction

`python -i fakeBreco.py`

Reconstructs all b-particles using links to MCTruth. Provides mass resolution plots

`python -i fakeJpsireco.py`

## 8. Fake $J/\psi$ reconstruction, as above

## 9. Hlt Velo Position A and C side

`python -i HltVeloPos.py`

## 10. Make ROOT histograms and convert to HBOOK

`python -i Ex11b.py and Ex11c.py`

## 11. Execute a list of sample scripts and search for new warnings and errors in the log files

`python runall.py > log.txt`

- Other stuff, random collection

- If you want to know which Algorithms are running with which parameters:

- ▶ `>>> appMgr.algorithms()`  
`['ProtoPRecalibration', 'MakeSomeResonances', 'SeqDC06selBs2Jpsi2MuMu_Phi2KK', 'UnpackMCParticle', 'UnpackMCVertex', 'ChargedProtoCombineDLLsAlg', ... , 'DC06selBs2Jpsi2MuMu_Phi2KK']`
- ▶ `>>> anAlg = appMgr.algorithm('Phi2KKForDC06selBs2Jpsi2MuMu_Phi2KK')`
- ▶ `>>> for x in anAlg.properties() :`  
`... print x,getattr(anAlg,x)`
- ▶ `EvtColDir Phi2KKForDC06selBs2Jpsi2MuMu_Phi2KK`
- ▶ `... MotherFilterName MotherFilter ...`
- ▶ `anAlg = appMgr.algorithm('Phi2KKForDC06selBs2Jpsi2MuMu_Phi2KK.MotherFilter ')`
- ▶ `>>> for x in anAlg.properties() :`  
`... print x,getattr(anAlg,x)`
- ▶ `Selections ['B_s0 : VtxFilterCriterion/bsVertex']`
- ▶ `anAlg = appMgr.algorithm('Phi2KKForDC06selBs2Jpsi2MuMu_Phi2KK.MotherFilter.bsVertex')`
- ▶ `>>> for x in anAlg.properties() :`  
`... print x,getattr(anAlg,x)`
- ▶ `MaxChi2 22.5`

- Alternative: Inspect the configurable of an algorithm

## ■ Booking:

- ▶ `hstore = {}`
- ▶ For every histogram, function:  
`hstore([aUniqueKey]) = TH1F(aUniqueKey, 'title', 100, 1., 0.)`

## ■ Filling:

- ▶ `hstore([aUniqueKey]).Fill(x)`

## ■ Retrieval from ROOT file:

- ▶ `from ROOT import gROOT, TFile, TROOT`
- ▶ `def TROOT_getitem(self, item) :`
- ▶ `return self.FindObject(item)`
- ▶ `TROOT.__getitem__ = TROOT_getitem`
- ▶ `hstore = gROOT._gROOT`
- ▶ `file = TFile(....)`
- ▶ `hstore[aUniqueKey].Draw()`

- ▶ `a=3.1415; b='text'`
- ▶ `print 'a=',a, 'b=',b` : `a= 3.1415 b= text`
- ▶ `print 'a=%6.2f b=%s'%(a,b)` : `a= 3.14 b=text`
- ▶ `print 'a=%(X)03d'%{'X':7}` : `a=007`

◆ Define a helper class:

```
class irange(object) :
 def __init__(self, b, e) :
 self.begin, self.end = b, e
 def __iter__(self):
 it = self.begin
 while it != self.end :
 yield it.__deref__()
 it.__postinc__(1)
```

gaudigadets.py

◆ Then :

```
from gaudigadgets import irange as irange
GenEvtContainer = evt['Gen/HepMCEvents']
for genEvt in GenEvtContainer:
 pGenEvt = genEvt.pGenEvt()
 for particle in irange(pGenEvt.particles_begin(), pGenEvt.particles_end()):
 if particle.production_vertex() :
 vtx = particle.production_vertex()
 print "PDG ID %i production vertex z=%4.2F part out: %3i'"
 %(particle.pdg_id(), vtx.position().z(), vtx.particles_out_size())
 else:
 print "PDG ID %i " %(particle.pdg_id())
```

- ◆ Details about Event Tag Collections can be found [here](https://twiki.cern.ch/twiki/bin/view/LHCb/DC06StrippingHowTo):  
<https://twiki.cern.ch/twiki/bin/view/LHCb/DC06StrippingHowTo>
- ◆ `etc = GaudiPython.AppMgr().evtcolsvc()`
- ◆ `appMgr.initialize()`
- ◆ `etc.dump()`  

```

/NTUPLES
/NTUPLES/EventSelector.DataStreamTool_1
/NTUPLES/EventSelector.DataStreamTool_1/TagCreator
/NTUPLES/EventSelector.DataStreamTool_1/TagCreator/1

```
- ◆ `tupl=etc['EventSelector.DataStreamTool_1/TagCreator/1']`  

```

<ROOT.NTuple::ColumnWiseTuple object at 0x3efb180>

```
- ◆ `dir(tupl)` does not yet yield any useful methods, DEAD END. Wait for feedback from experts. Meanwhile use ROOT.
- ◆ `tupl = gROOT.FindObjectAny('<local>_TagCreator_1')`  

```

<ROOT.TTree object ("<local>_TagCreator_1") at 0x3e97600>

```

# Event Tag Collections II

```
◆ lvs = tupl.GetListOfLeaves()
◆ for n in lvs : print n.GetName()
```

```
Address
token
event
run
entry
nPrim
nChargedProto
nNeutralProto
PreselB2DiMuon
PreselBd2KstarMuMu
PreselBd2Kstaree
PreselBs2MuMu
PreselBu2LLK
PreselBs2PhiEtac
PreselHeavyDimuon
PreselJpsi2ee
PreselUnbiasedJpsi2ee
...
```

```

▶ DataOnDemandSvc().AlgMap['/Event/Link/Raw/Muon/Coords'] =
 'MuonCoord2MCParticleLink'
▶ ...
▶ MCMuonDigitInfo = GaudiPython.gbl.LHCb.MCMuonDigitInfo
▶ MuonDigit = GaudiPython.gbl.LHCb.MuonDigit
▶ MCParticle = GaudiPython.gbl.LHCb.MCParticle
▶ MuonCoord = GaudiPython.gbl.LHCb.MuonCoord
▶ ...
▶ appMgr.run(1)
▶ lcoord2part = linkedTo(MCParticle, MuonCoord, 'Raw/Muon/Coords')
▶ l_mcpart = {}
▶ muonCoords = evt['Raw/Muon/Coords']
▶ # take first muonCoord as an example
▶ obj = muonCoords.containedObjects()[0]
▶ # find all mcparticles linked with this object
▶ l_mcpart[obj]=[]
▶ for mcp in lcoord2part.range(obj) :
▶ l_mcpart[obj].append(mcp)
▶ # print info about muon tile and its related MCParticles
▶ print l_mcpart

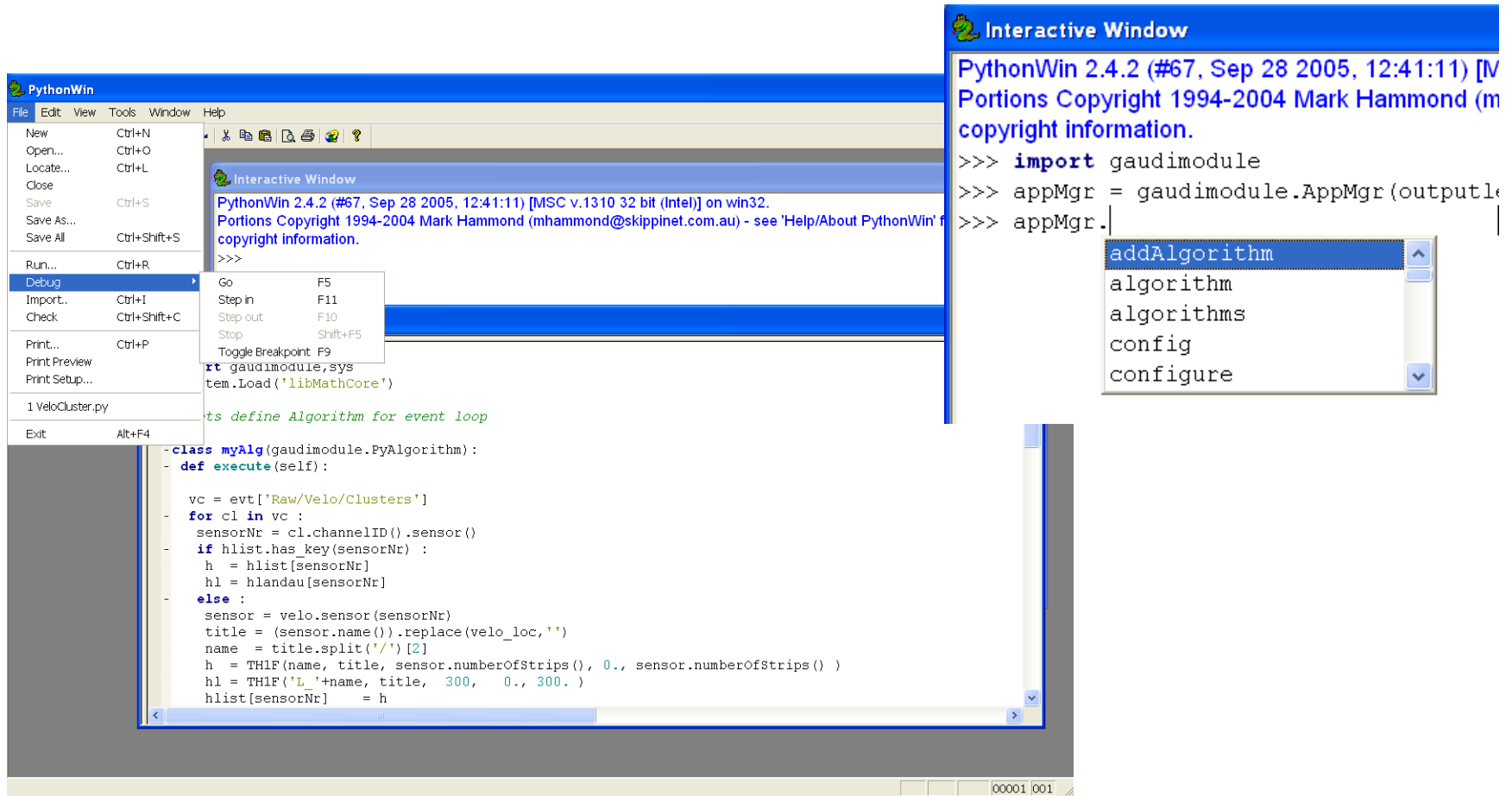
```

```
def muondigithistory(mc):
 if mc.isGeantHit(): print 'IsGeant'
 if mc.isXTalkHit(): print 'IsXtalk'
 if mc.isBackgroundHit(): print 'IsBackg'
 if mc.isChamberNoiseHit(): print 'IsChamberNoise'
 if mc.isElNoiseHit(): print 'ElNoiseHit'
 if mc.isFlatSpilloverHit(): print 'FlatSpillover'
 if mc.isMachineBkgHit(): print 'MachineBkg`

 ▶ # Vector of digit
 ▶ digits = obj.digitTile()
 ▶ MCInfo = evt['MC/Muon/DigitsInfo']
 ▶ mci = MCMuonDigitInfo()
 ▶ for d in digits :
 ▶ info = MCInfo.link(d.key())
 ▶ mci.setDigitInfo(info)
 ▶ # print more info about MC history
 ▶ muondigithistory(mci)
```

# AOB

- Pythonwin: Editor, debugger, command completion

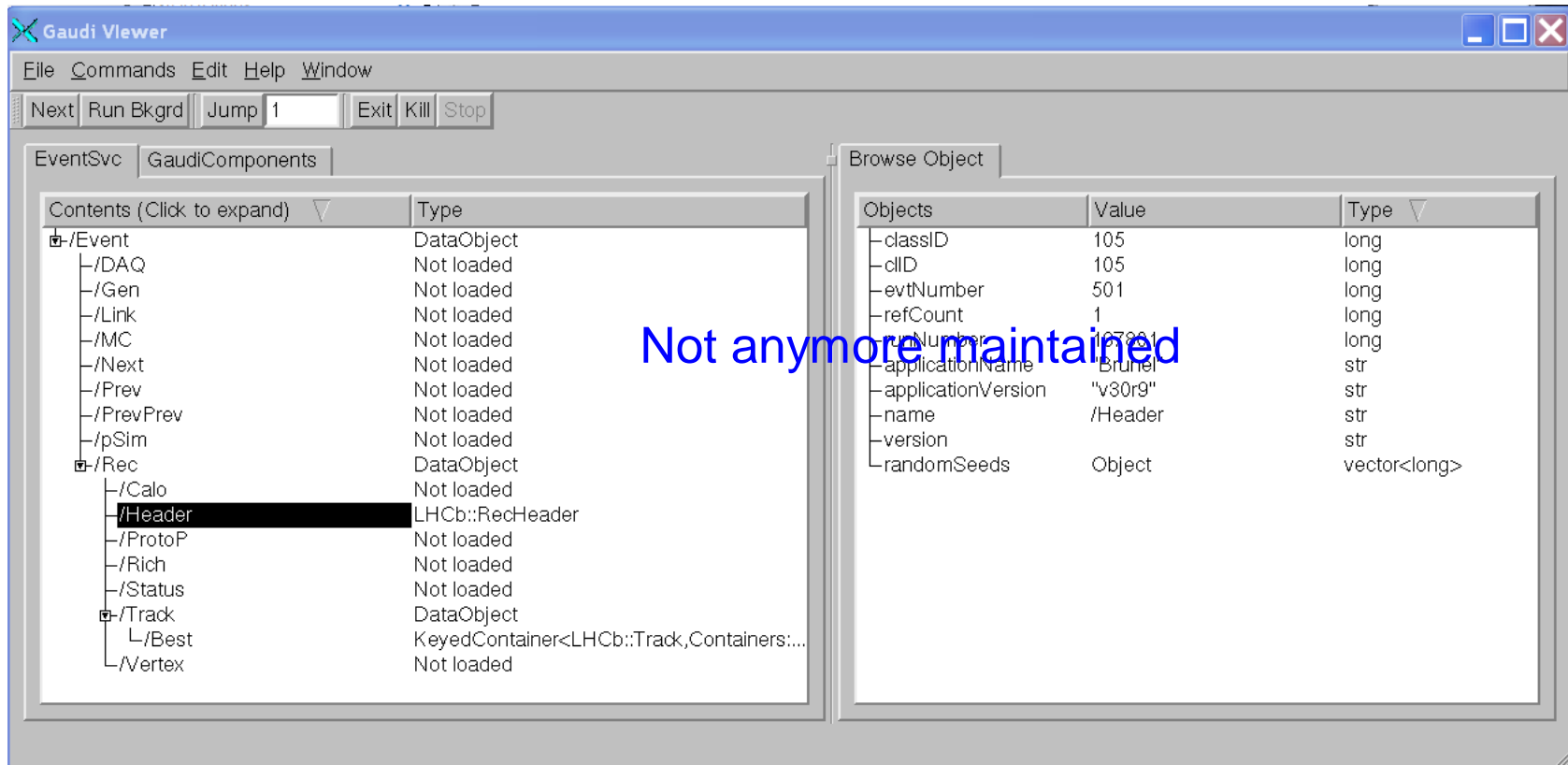


The screenshot displays the PythonWin application interface. On the left, a menu is open with options like 'New', 'Open...', 'Save', and 'Debug'. The main editor window shows Python code for a class named `myAlg` that inherits from `gaudimodule.PyAlgorithm`. The code includes a `def execute(self):` method that processes sensor data. A 'Debug' menu is also visible, showing options like 'Go', 'Step in', and 'Stop'. On the right, an 'Interactive Window' is open, showing the Python prompt and the execution of `import gaudimodule` and `appMgr = gaudimodule.AppMgr(output)`. A dropdown menu is shown below the prompt, listing completion options: `addAlgorithm`, `algorithm`, `algorithms`, `config`, and `configure`.

### ◆ Gaudi Viewer (Radu Stoica)

► [Presentation](http://indico.cern.ch/getFile.py/access?contribId=10&resId=1&materialId=slides&confId=3130)

```
>>> python Gaudi.py -d -f
$AFSROOT/cern.ch/lhcb/group/tracking/vol1/00001378_00000002_5.dst
```



The screenshot shows the Gaudi Viewer application window. The main area displays a tree view of event components under 'EventSvc' and 'GaudiComponents'. The tree is expanded to show the '/Rec' component, which contains several sub-components like '/Calo', '/Header', '/ProtoP', '/Rich', '/Status', '/Track', and '/Vertex'. The '/Header' component is highlighted. To the right, a 'Browse Object' window is open, displaying a table of object attributes and their values.

| Objects             | Value   | Type         |
|---------------------|---------|--------------|
| -classID            | 105     | long         |
| -clID               | 105     | long         |
| -evtNumber          | 501     | long         |
| -refCount           | 1       | long         |
| -runNumber          | 107801  | long         |
| -applicationName    | Brunel  | str          |
| -applicationVersion | "v30r9" | str          |
| -name               | /Header | str          |
| -version            |         | str          |
| -randomSeeds        | Object  | vector<long> |

Not anymore maintained

# Gaudi Viewer

Gaudi Viewer

File Commands Edit Help Window

Next Run Bkgrd Jump 1 Exit Kill Stop

EventSvc GaudiComponents

| Contents (Click to expand) | Type                                      |
|----------------------------|-------------------------------------------|
| [-] /Event                 | DataObject                                |
| [-] /DAQ                   | Not loaded                                |
| [-] /Gen                   | Not loaded                                |
| [-] /Link                  | Not loaded                                |
| [-] /MC                    | Not loaded                                |
| [-] /Next                  | Not loaded                                |
| [-] /Prev                  | Not loaded                                |
| [-] /PrevPrev              | Not loaded                                |
| [-] /pSim                  | Not loaded                                |
| [+] /Rec                   | DataObject                                |
| [-] /Calo                  | Not loaded                                |
| [-] /Header                | LHCb::RecHeader                           |
| [-] /ProtoP                | Not loaded                                |
| [-] /Rich                  | Not loaded                                |
| [-] /Status                | Not loaded                                |
| [+] /Track                 | DataObject                                |
| [-] /Best                  | KeyedContainer<LHCb::Track,Containers:... |
| [-] /Vertex                | Not loaded                                |

Browse Object

| Objects               | Value          | Type                  |
|-----------------------|----------------|-----------------------|
| [-] Contained Objects |                |                       |
| [-] 00                | Object         | LHCb::Track           |
| [-] chi2PerDoF        | 0.849472019366 | float                 |
| [-] extralInfo        | Object         | GaudiUtils::Vector... |
| [-] fitHistory        | 2              | int                   |
| [-] fitStatus         | 1              | int                   |
| [-] flag              | 0              | int                   |
| [-] history           | 4              | int                   |
| [-] nDoF              | 8              | int                   |
| [-] patRecStatus      | 1              | int                   |
| [-] type              | 4              | int                   |
| [-] flags             | 268521540      | long                  |
| [-] specific          | 1              | long                  |
| [-] parent            | Object         | Not loaded            |
| [+] /hcbIDs           | Object         | vector<LHCb::LH...    |
| [-] Contained ...     |                |                       |
| [-] 00                | Object         | LHCb::LHCbID          |
| [-] 01                | Object         | LHCb::LHCbID          |
| [-] 02                | Object         | LHCb::LHCbID          |

Gaudi Viewer

File Commands Edit Help Window

Next Run Bkgrd Jump 1 Exit Kill Stop

EventSvc GaudiComponents

| Contents (Click to expand)  | Type     |
|-----------------------------|----------|
| [+] ApplicationMgr          | AppMgr   |
| [-] Algorithms              |          |
| [-] PyAlgorithms            |          |
| [+] Services                |          |
| [-] AppMgrRunnable          | iService |
| [-] EventDataSvc            | iService |
| [-] EventLoopMgr            | iService |
| [-] EventPersistenceSvc     | iService |
| [-] EventSelector           | iService |
| [-] HistogramDataSvc        | iService |
| [-] HistogramPersistenceSvc | iService |
| [-] IncidentSvc             | iService |
| [-] JobOptionsSvc           | iService |
| [-] MessageSvc              | iService |
| [-] PoolDbCacheSvc          | iService |
| [-] PoolRootEvtCnvSvc       | iService |
| [-] PoolRootKeyEvtCnvSvc    | iService |
| [-] PoolRootTreeEvtCnvSvc   | iService |
| [-] ToolSvc                 | iService |

Browse Object

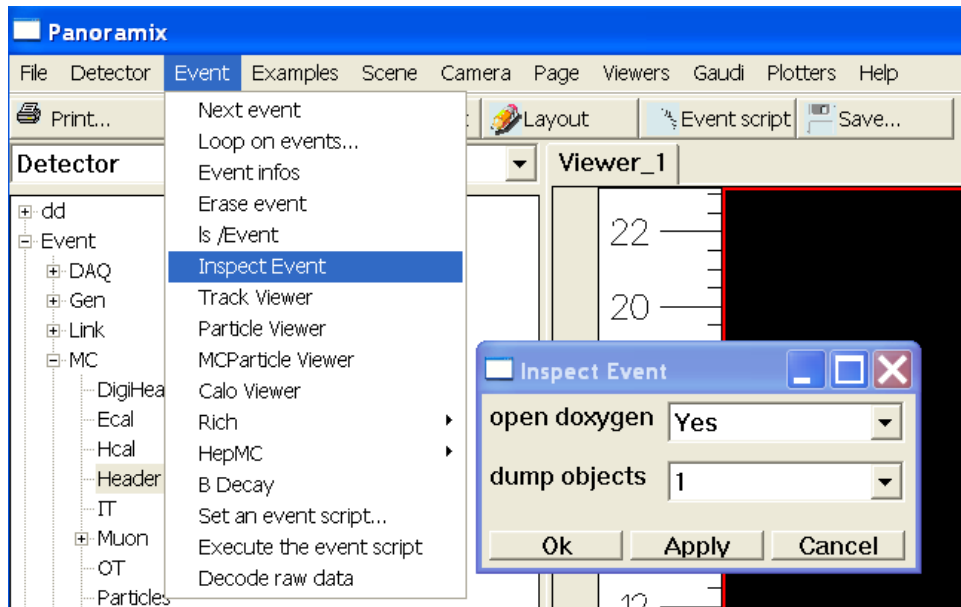
| Objects             | Value  | Type          |
|---------------------|--------|---------------|
| [-] CONFIGURED      | 1      | int           |
| [-] FINALIZED       | 2      | int           |
| [-] INITIALIZED     | 3      | int           |
| [-] OFFLINE         | 0      | int           |
| [-] State           | 3      | int           |
| [-] AuditFinalize   | Object | PropertyEntry |
| [-] AuditInitialize | Object | PropertyEntry |
| [-] AuditServices   | Object | PropertyEntry |
| [-] EvtMax          | Object | PropertyEntry |
| [-] FirstEvent      | Object | PropertyEntry |
| [-] Input           | Object | PropertyEntry |
| [-] OutputLevel     | Object | PropertyEntry |
| [-] PrintFreq       | Object | PropertyEntry |
| [-] StreamManager   | Object | PropertyEntry |

# Feicim Browser

- Zsolt Lazar, Ronan McNulty
  - ◆ [Ideas presented at Core Software meeting, 28 March 2007](#)
  - ◆ To facilitate start-up of newcomers

## ■ Three steps

1. Select container in Event Tree
2. Start Data Browser under Event Menu
  - Select doxygen class information
  - Select how many objects to dump
    - Information dumped depends on the implementation of the fillstream method by the class author



# Full list of Examples

| Script                  | Short Description                                                      |
|-------------------------|------------------------------------------------------------------------|
| AccessBookkeeping.py    | access to production info, requiring prodid                            |
| checkGeant4_MS.py       | compare Geant4 MCHits with LHCb extrapolator                           |
| checkMeasurements_pg.py | detailed check of Velo Cluster resolutions                             |
| checkMeasurements_ST.py | detailed check of ST Cluster resolutions                               |
| createPrivateDict.py    | create a private dictionary for simple classes:<br>int, double, string |
| database_localcopy.py   | make a local copy of database                                          |
| decodePUS.py            | python decoding of the PUS detector                                    |
| dst_sizes.py            | measure I/O, only works on Windows                                     |
| Ex0.py                  | simplest script                                                        |
| Ex10b.py                | reading mdf file, running PatSeeding                                   |
| Ex10.py                 | reading mdf file, calorimeter                                          |
| Ex11b.py                | Reconstruct B, plot B mass, convert to HBOOK                           |
| Ex11c.py                | Reconstruct B, plot B mass, convert to HBOOK, DC06                     |
| Ex11d.py                | Reconstruct B, plot B mass, lines for proto2mc link                    |

# Full list of Examples, cont.

| Script   | Short Description                                                           |
|----------|-----------------------------------------------------------------------------|
| Ex11.py  | reconstruct B, plot B mass                                                  |
| Ex12b.py | Decay tree, increase output level via python opts                           |
| Ex12.py  | Decay tree                                                                  |
| Ex13b.py | copy only Raw banks for L0 triggered events                                 |
| Ex13c.py | Read back L0 triggered events                                               |
| Ex13d.py | Copy events with B candidates                                               |
| Ex13.py  | Make one to one copy of input DST using InputCopyStream                     |
| Ex14.py  | Running Hlt, example printouts                                              |
| Ex15.py  | visualize a track                                                           |
| Ex16.py  | read HepMCEvents and dump info                                              |
| Ex17b.py | read ETC, Event Tag Collection. make mass plot                              |
| Ex17.py  | read ETC, Event Tag Collection                                              |
| Ex1b.py  | read DST, print header                                                      |
| Ex1.py   | read DST,unpack all packed containers read DST,unpack all packed containers |

# Full list of Examples, cont.

| Script            | Short Description                                         |
|-------------------|-----------------------------------------------------------|
| Ex_multicore.py   | examples for running script on multicore machines         |
| Ex1_multicore.py  |                                                           |
| Ex2b_multicore.py |                                                           |
| Ex2c_multicore.py |                                                           |
| Ex2_multicore.py  |                                                           |
| Ex2_singlecore.py |                                                           |
| Ex3_multicore.py  |                                                           |
| Ex3_singlecore.py |                                                           |
| Ex2.py            | read DST, fill ROOT histogram                             |
| Ex3b.py           | DataOnDemand example                                      |
| Ex3c.py           | print size of containers, access to doxygen documentation |
| Ex3d.py           | rewind file and fill another histogram                    |
| Ex3e.py           | direct access to Aida histograms                          |
| Ex3.py            | add algorithms to TopAlg                                  |

# Full list of Examples, cont.

| Script       | Short Description                             |
|--------------|-----------------------------------------------|
| Ex4b.py      | measure time to access different det elements |
| Ex4.py       | access Velo detector element                  |
| Ex5.py       | create objects, XYZPoint and Vector           |
| Ex6b.py      | use TrackMasterExtrapolator                   |
| Ex6.py       | use Velo position tool                        |
| Ex7b.py      | use ParticlePropertyService, ppSVC            |
| Ex7c.py      | use Magnetic Field Service                    |
| Ex7.py       | use Transport Service                         |
| Ex8.py       | use linker tables to access MC truth          |
| Ex9.py       | dump raw bank in HEX and ASCII format         |
| Ex_Castor.py | access to castor information                  |
| Ex_mdf.py    | open an MDF file and dump an event            |
| ExMuonEff.py | check muon id efficiency as function of pt    |
| Ex_TisTos.py | run Hlt1+Hlt2 and check for signal Tis Tos    |

# Full list of Examples, cont.

| Script              | Short Description                                                       |
|---------------------|-------------------------------------------------------------------------|
| fakeBreco.py        | reconstruct B decays by following links from MC to reconstructed tracks |
| fakeJpsireco.py     | reconstruct J/psi by following links from MC to reconstructed tracks    |
| FullMonty.py        | run Gauss/Boole/Brunel in one go                                        |
| FullSequence.py     | run reconstruction and phys sequence in one go                          |
| gaudigadgets.py     | additional useful gaudipython methods                                   |
| HLTandOffline.py    | compare Hlt tracking and offline tracking                               |
| hltexamples.py      | hltexample module                                                       |
| HltVeloPos.py       | plot primary vertex positions of Velo A and C side                      |
| IPandPresol_plot.py | determine and plot IP and momentum resolution and pulls                 |
| KsCheck.py          | investigate Kshort reconstruction                                       |
| L0DUextract.py      | write L0 triggered events with additional user info                     |
| L0DUreadtest.py     | read back L0 triggered events                                           |
| LHCbConfig.py       | module for basic LHCb software configuration                            |
| microDSTEx0.py      | write a microDST                                                        |

# Full list of Examples, cont.

| Script                   | Short Description                                         |
|--------------------------|-----------------------------------------------------------|
| microDSTEx1.py           | read a microDST                                           |
| myBrunel.py              | main script for Brunel                                    |
| myGauss.py               | main script for Gauss                                     |
| myBoole.py               | main script for Boole                                     |
| ParallelMonty.py         | run Gauss/Boole/Brunel in one go parallel on n processors |
| pickleROOT.py            | not needed anymore, now included in ROOT                  |
| PrimVX.py                | compare vertices reconstructed in different Velo halves   |
| protonDecay_multicore.py | look for strange elastic events                           |
| rad_map06.py             | produce map of radiation length at various positions      |
| rawbuffer_sizes_2008.py  | determine rawbuffer sizes                                 |
| ReadLogFiles_2008.py     | read logfiles on the internet                             |
| ReadLogFiles.py          | plot position of MCHits                                   |
| readSIM_MCHits.py        | direct access to Aida histograms                          |
| Reco.py                  | run track reconstruction                                  |

# Full list of Examples, cont.

| Script               | Short Description                                          |
|----------------------|------------------------------------------------------------|
| replace.py           | replace string A with B in files of current directory      |
| RootExa1.py          | create, fill and plot a root histogram                     |
| RootExa2.py          | simulate Landau distributions, save file and read back     |
| runall.py            | run a list of selected scripts and examine log files       |
| run_Ex10b.py         | re-edits Ex10b.py for different TAE slots and runs it      |
| selv19r9.py          | write out selected events                                  |
| TGhost.py            | print info about ghosts                                    |
| VeloAlignment_0.py   | velo misalginment scenario, input config file              |
| VeloCluster.py       | make hitmaps and Landau distributions for all Velo sensors |
| VeloDet06.py         | access Velo detector element, plot velo sensor geometry    |
| velopostool_check.py | check Velo position tool                                   |