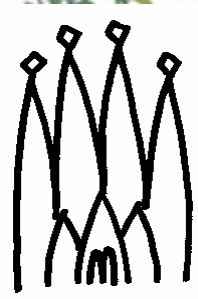


Python-based Physics Analysis Environment for LHCb

Vanya Belyaev, LAPP/Annecy

Goals

(User friendly) Environment for development of physics analysis code

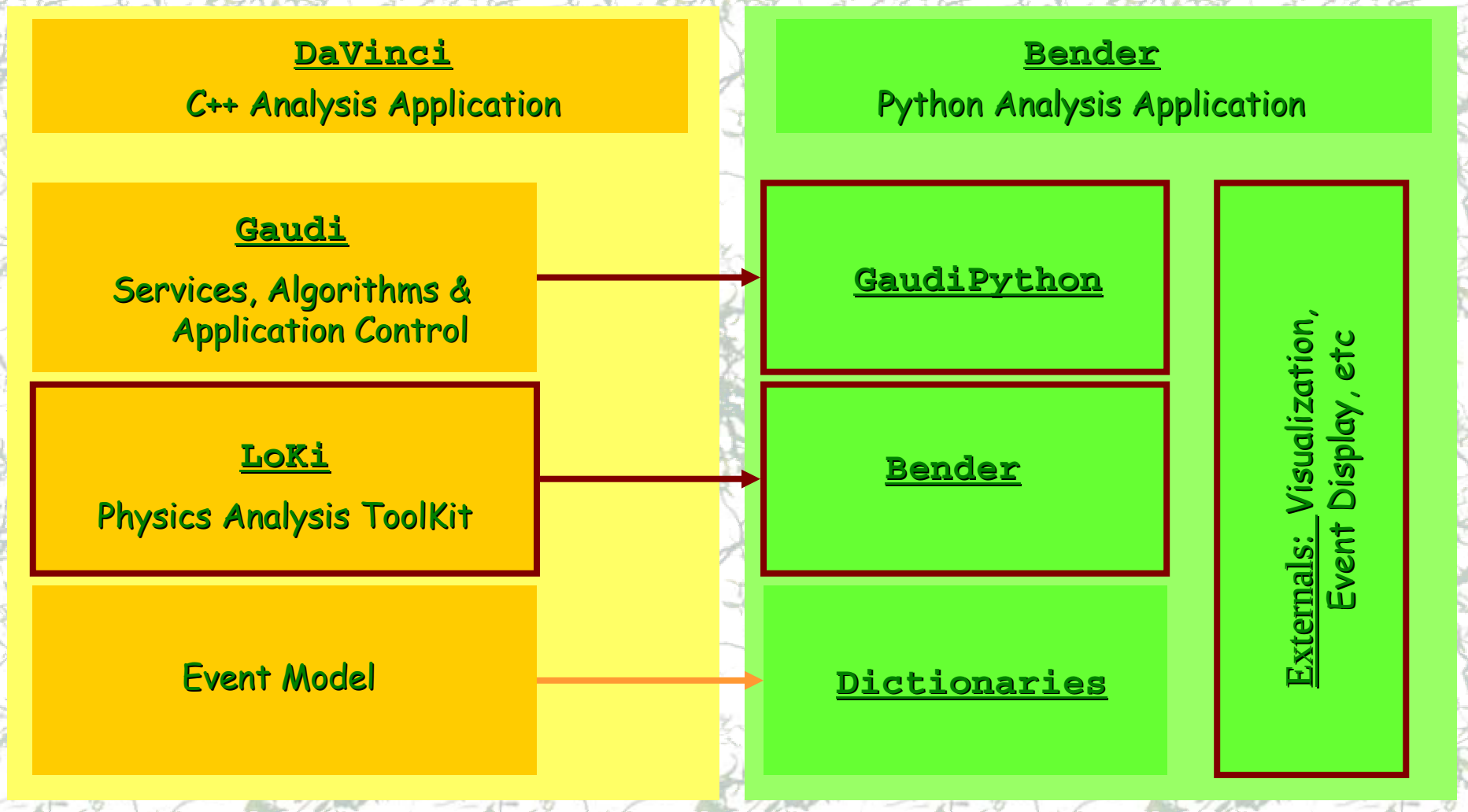
- "Easy-to-understand", "readable"
  - Physics-driven semantics
  - 1-1 matching with physical description
  - all technical details are masked from user
- Compact
  - ~1 page of code per "typical" analysis algorithm
- Interactive
  - (Re)Define cuts/algorithms on-flight

  \+ Visualization
- Complete
  - not *ONLY* development
- RAD
  - no compilation

Why Python-based ?

- Python is a language with the special emphasize for fast prototyping and development
- Scripting and interactivity combined in a natural way
- Easy integration with the third party software
- Availability of external packages
  - visualization, statistical analysis
    - ROOT, HippoDraw, Panoramix, PyX, GNUplot, PAIDA
  - Event display
    - Panoramix
  - Bookkeeping data base
  - Interface to GRID
    - GANGA
  - and many others

LHCb Analysis Applications



GaudiPython

The generic package for **Gaudi & Python** bindings

- Access to major **Gaudi** Components
  - Services, algorithms and tools
- Application Configuration
  - Algorithm schedule
  - Configuration of all components
  - "Dynamic" reconfiguration is possible

The technique

- **LCG dictionaries** for **C++/Python** binding

LoKi

User friendly C++ Physics Analysis ToolKit

- Set of high level analysis utilities
- Physics oriented semantics
  - Inspired by **KAL** from the genius H. Albrecht and **GPATTERN** by T. Glebe
- Compact Code
- Concept of locality
- Technicalities are "hidden" from end-user
- The kernel has low coupling to event model
- Template/Inline/Efficient

LoKi


```
LOKI_ALGORITHM( MyAlg )
{

  select( "K-"  ,
          "K-" == ID && PT > 1 * GeV );
  select( "pi+" ,
          "pi+"== ID && P  > 3 * GeV );

  Cut dmass = ABSDM("D0") < 30 * MeV ;
  for( Loop D0 = loop("K- pi+","D0") ;
                           D0 ; ++D0 )
   {
     if ( VCHI2(D0) > 4 && dmass( D0 ) )
                  { D0->save("D0") ; }
   }

  for ( Loop Dst = loop( "D0 pi+");
                       Dst ; ++Dst )
   {
     double dm = M(Dst) - M1(Dst) ;
     plot ( dm , "DM for D*+" , 130 , 180 ) ;
   }

  return StatusCode::SUCCESS ;
};
```

- Plenty of useful "functors"
  - `P, PT, IP,VCHI2,Q,….`
- Selection/filtering of particles
- Multi-particle looping
- Kinematical/Topology fits
- Intuitive interface to histograms and `N-Tuples`
  - Book-and-fill on demand
- Easy matching for MC-truth information

Bender = LoKi + Python + …

- LCG dictionaries for C++/Python binding
  - A bit of raw Boost also

Only Boost.Python for v1

- >95% of LoKi's C++ functionality is available in Python
  - Non-trivial due to heavy templated nature of LoKi d
    - set of wrappers is required
  - Situation improves with PyLCG evolution
- The mixture of C++ and Python is possible
  - C++ algorithm with Python cuts ("LoKiHybrid")
- The bulk of actual computations in C++
  - Minimal Python-related penalty
- The conversion between existing Python Bender's and C++ LoKi' s algorithms is simple in both directions:
  - Semantics is very similar

Bender v*

- Develop own `Python` algorithm
  - Inheritance from the base class

```
class MyAlg(Algo)
```

- Configure own algorithm (if needed)

```
myAlg.OutputLevel = 5
```

- Configure the rest of job (if needed)
  - Reuse of standard configuration *.opts files for Gaudi

```
bender.config( files =
  ['DaVinciCommon.opts','DaVinciReco.opts'] )
hsvc.OutputFile = "myhistos.hbook"
```

- execute:

```
gaudi.run(500)
```

Configuration: CMT + GaudiPython

- Job (self)-configuration/environment: **cmt.py**

```
import cmt
cmt.project( 'Bender' , 'v4r0' )
cmt.use ( package = 'Ex/BenderExample')
cmt.setup()
```

- Application configuration: **GaudiPython**

```
gaudi.TopAlg += [ "MyAlgorithm" ]
gaudi.ExtSvc += [ "MyService" ]
gaudi.run( 100 )
```

Physics analysis

- Functions and cuts

```
cut = (ID =='D+') & (P > 5*GeV ) & (PT > 2*GeV )
```

- Selection of particles

```
k = self.select( tag = 'K+' ,
        cuts = ( 'K+' == ID ) & ( PT > 0.5 * GeV )
```

- Looping over combinations

```
for phi in self.loop(formula='K+ K-',pid='phi(1020)'):
    m = M( phi ) / GeV
    p = P( phi ) / GeV
```

- Saving/retrieve of interesting combinations,
- Vertex/Mass-Vertex/Direction/Lifetime fits

Histos & N-Tuples

(Pre)booking is **OPTIONAL**

- **Histograms**

```
h1 = self.plot ( title = " phi mass "   ,
                 value = M( phi )       ,
                 low=1000, high=1050    )
```

- **N-Tuples**

```
tup = self.nTuple( title="Phi NTuple")
tup.column ( name= "ID", value= ID(phi) )
tup.column ( name= "p" , value= P (phi) )
tup.column ( name= "pt", value= PT(phi) )
tup.write()
```

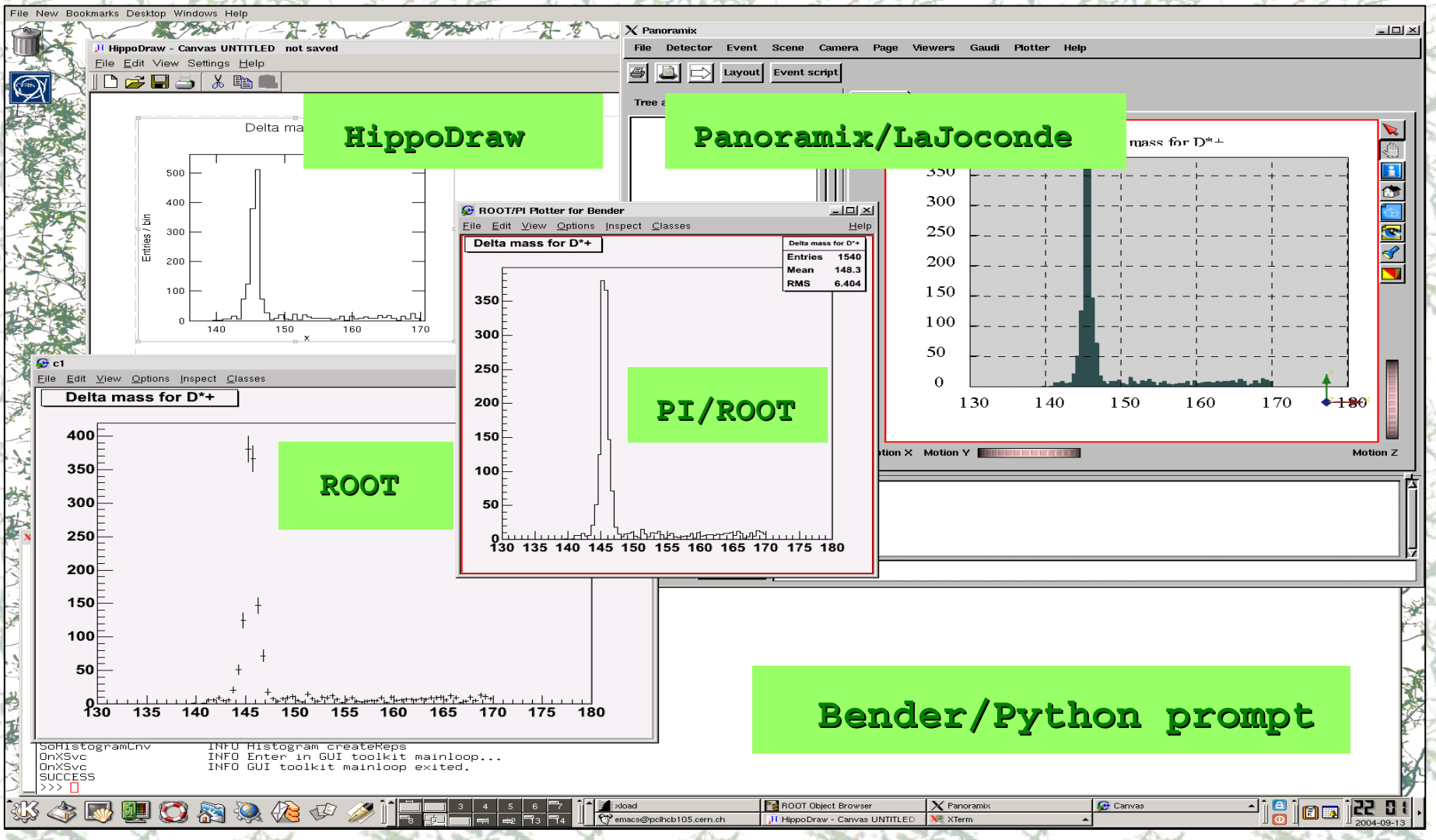
Histo visualization

- The histogram visualization can be done through
  - ROOT
    - native ROOT through Python prompt : PyROOT
    - rootPlotter from PI through AIDA pointer : PiROOT
  - Panoramix
    - Directly through AIDA pointer
  - HippoDraw
    - hippoPlotter from PI through AIDA pointer
- Few lines "common interface" for trivial plotting exist
- The interactive analysis of Gaudi N-Tuples is possible in Bender with ROOT persistency and ROOT module directly
  - Prototype for HippoDraw

Histo visualization

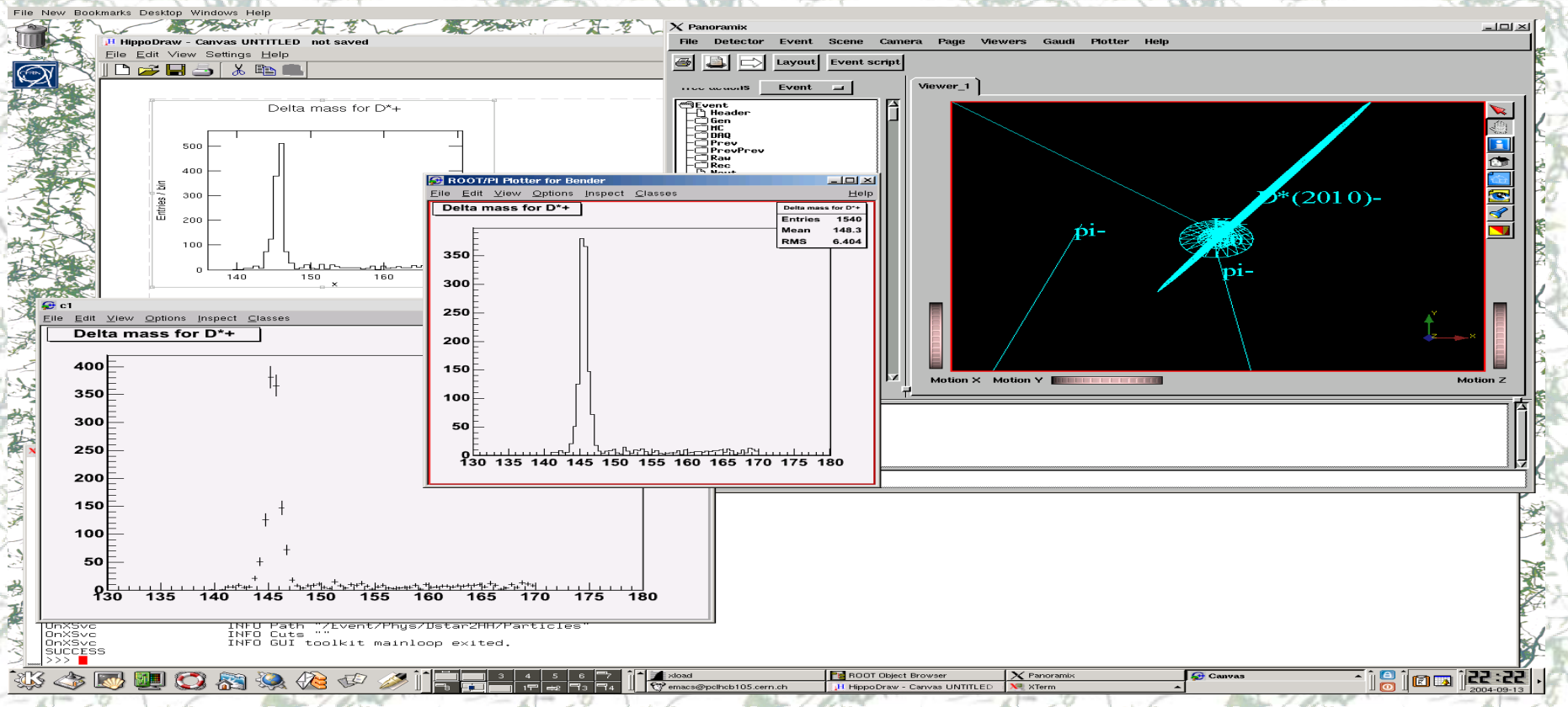

```
# get the histogram (AIDA pointer)
h1 = histoSvc['MyAlg/1']


from benderPiHippo import plotter
plotter.plot ( h1 )

from benderPiRoot import plotter
plotter.plot ( h1 )

from benderPyROOT import plotter
plotter.plot ( h1 )
```

Everything can be combined



Event Display: Panoramix/LaJoconde

The integrated analysis and visualization of statistical and event data is possible



Hybrid (C++/Python) solution

"Hybrid" for HLT

- The hybrid(C++/Python) solution has been developed on explicit request of the most clever physicists:

  the analysis code in C++

  the actual cuts in Python

  No Python-related CPU penalty!!!

*.opts:

```
MyAlg.Filters = {"HybridFilter/Cut1"} ;
Cut1.Code =
" ( MINTREE( PT , 'pi+' == ABSID  ) > 0.5 * GeV ) &
  ( abs( LV01 )          < 0.9       ) &
  ( abs( DMASS("D0") ) < 20 * MeV )   " ;
```

"(1) *The minimal transverse momentum for all daughter π^+ and π^- must be in excess of 500 MeV and (2) the absolute value of cosine of the decay angle should not exceed 0.9 and (3) the invariant mass should be within 20 MeV/c^2 from the nominal mass of D^0 "*

Result


```
from bendermodule import *
class Dstar(Algo):
    def analyse( self) :
        self.select ( tag='K-', cuts=('K-' ==ID)&(PT>1*GeV) )
        self.select ( tag='pi+',cuts=('pi+'==ID)&(P >3*GeV) )
        dmass = ABSDM("D0") < 30 * MeV

        for D0 in self.loop ( formula='K- pi+' , pid='D0' ) :
            If ( VCHI2(D0) < 4 ) &  dmass( D0 ) : D0.save('D0')

        tup = self.nTuple ( title = "D*+ N-Tuple " )

        for Dst in self.loop ( formula='D0 pi+' ,
                               pid='D*(2010)+' ) :
            dm = M(Dst)-M1(Dst)
            h1 = self.plot( title = "Delta mass for D*+" ,
               value = dm , low=130 , high=170 )
            tup.column( name = 'M'    , value = M(Dst)  / GeV )
            tup.column( name = 'DM'   , value = dm      / GeV )
            tup.column( name = 'p'    , value = P (Dst) / GeV )
            tup.column( name = 'pt'   , value = PT(Dst) / GeV )
            tup.write ()
        return SUCCESS
```

Comeback to C++ is trivial!

Summary

- **Bender** is a part of officially released LHCb software
- Outcome from **Bender** to **Gaudi (Python)** is expected
- Adequate functionality to perform physics analysis
  - Even more is expected
- **Bender** is a friendly guy:
  - **ROOT, HippoDraw, Panoramix**
  - (**GANGA,DIRAC,…**)?
- Physicists are using it for their studies
  - Some of them (>50%???) use it for completely different purposes
    - **HLT** development, Track reconstruction tests, etc…
    - Originally not expected at all
- Mailing list: **lhcb-bender@cern.ch**

LoKi

Bender

- **Loki** is a god of wit and mischief in Norse mythology
- **Lo**ops & **Ki**nematics

Ostap Suleiman Berta Maria **Bender**-bei

- The cult-hero of books by *I.Il'f & E.Petrov: "The 12 chairs" ,"The golden calf"*
- The title: "The great schemer"
- Attractive & brilliant cheater

Essential for successful and good physics analysis