Getting started with FairShip: simple analysis

Elena Graverini, Thomas Ruf, Annarita Buonaura, Alexander Baranov

October 9, 2015

1 Installation on AFS

Make sure you have an AFS account in the ship-cg computing group. In order to register, you should go to the cern account page: https://account.cern.ch/account There are two possibilities:

1. if you work in parallel for another experiment, you should create a secondary account: https://account.cern.ch/account/Management/NewAccount.aspx, and send Thomas Ruf or Joel Closier the user name for the secondary account.
2. if SHiP is your only experiment, change to computing group ship-cg by asking the administrator of your present computing group to remove you and then send Thomas Ruf or Joel Closier your user name.

Being member of the ship-cg computing group will also help you in the future for getting priorities in executing batch jobs.

Login to LXPLUS with your SHiP account:

```
ssh -X USERNAME@lxplus.cern.ch
```

Prepare your environment:

```
touch shipsetup.sh
echo "source ~/shipsetup.sh" >> ~/.bashrc
echo "export SHIPSOFT=/afs/cern.ch/ship/sw/ShipSoft" >> ~/shipsetup.sh
echo "export FAIRROOTPATH=\${SHIPSOFT}/FairRootInst" >> ~/shipsetup.sh
echo "export SIMPATH=\${SHIPSOFT}/FairSoftInst" >> ~/shipsetup.sh
```

Get the latest FairShip sources:

```
mkdir my_FairShip && cd my_FairShip
source ~/.bashrc
git clone https://github.com/ShipSoft/FairShip.git
```

Compile the sources:

```
mkdir FairShipRun && cd FairShipRun
cmake ../FairShip -DUSE_DIFFERENT_COMPILER=TRUE
make
```

If you followed all the steps above, the compilation should not give any error :) Finish the environment setup (make sure you adapt the paths, if you changed the installation directory):

```
echo "export FAIRSHIP=$HOME/my_FairShip/FairShip" >> ~/shipsetup.sh
echo "export FAIRSHIPRUN=$HOME/my_FairShip/FairShipRun" >> ~/shipsetup.sh
echo "source \${FAIRSHIPRUN}/config.sh" >> ~/shipsetup.sh
echo "export LD_LIBRARY_PATH=\${LD_LIBRARY_PATH}:/opt/rh/python27/root/usr/lib64" >> ~/shipsetup.sh
source ~/.bashrc
```

Run a simulation and event display:

```
cd $FAIRSHIP/macro
    python run_simScript.py -n 100
```

Please be patient, 100 events can take a few minutes if the machine is busy.

```
python -i eventDisplay.py -f ship.10.0.Pythia8-TGeant4.root\
                  -g geofile_full.10.0.Pythia8-TGeant4.root
```

Provided you logged in with the "-X" option, two windows should pop up, containing the event display and a GUI to select what detector elements must be shown.

If you encounter any problem with the process, please write to the ship software mailing list: ship-software@cern.ch

2 Setup

Make sure you have the last FairShip version:

```
cd my_FairShip
cd FairShip
git pull
```

If something was updated, re-compile and re-setup your environment:

```
cd ../FairShipRun
cmake ../FairShip
make
source config.sh
source ~/.bashrc
```

Please work from a custom directory outside FairShip's folder tree, in order to nos mess up the software:

```
cd ..
mkdir my_analysis
cd my_analysis
```

3 Hands-on tutorial

For this tutorial we are going to generate, reconstruct and analyse an ntuple of HNLs decaying to $\pi\mu$.

The python script `run_simScript.py` has several options. It can be used to generate HNLs, muon background, neutrino background, and each one of these use cases can be customized with further options. For now we will just generate a sample of 100 $HNL \to \pi\mu$ events generated in charmed meson decays, with the same couplings used for the studies in the TP:

```
python $FAIRSHIP/macro/run_simScript.py --couplings '4.47e-10,7.15e-9,1.7e-9' -n 100
```

Now let's run the reconstruction:

```
python $FAIRSHIP/macro/ShipReco.py -f ship.10.0.Pythia8-TGeant4.root \
                           -g geofile_full.10.0.Pythia8-TGeant4.root
```

Now let us open the output reconstructed file in pyROOT and see what's inside. We will perform a simple analysis of the reconstructed tree.

```
ipython -i
```

If you do not have IPython, just use `python -i`.

```
In [2]: # Import ROOT
        import ROOT as r

In [3]: # Open our reconstructed file and get the tree
        f = r.TFile('ship.10.0.Pythia8-TGeant4_rec.root', 'read')
        f.ls()
        t = f.Get('cbmsim')
        t.Print()
        tb = r.TBrowser()

In [41]: # Look at MCTracks. These are the basic objects representing the simulated particles
         t.GetEntry(1)
         mcTracks = t.MCTrack
         mcTracks[0]
         help(mcTracks[0])
         dir(mcTracks[0])

Help on ShipMCTrack in module __main__ object:

class ShipMCTrack(ROOT.TObject)
 |  Method resolution order:
 |      ShipMCTrack
 |      ROOT.TObject
 |      ROOT.ObjectProxy
 |      __builtin__.object
 |
 |  Methods defined here:
 |
 |  Class(self)
 |      static TClass* ShipMCTrack::Class()
 |
 |  Class_Name(self)
 |      static const char* ShipMCTrack::Class_Name()
 |
 |  Class_Version(self)
 |      static Version_t ShipMCTrack::Class_Version()
 |
 |  DeclFileLine(self)
 |      static int ShipMCTrack::DeclFileLine()
 |
 |  DeclFileName(self)
 |      static const char* ShipMCTrack::DeclFileName()
 |
 |  Dictionary(self)
 |      static void ShipMCTrack::Dictionary()
 |
 |  Get4Momentum(self, TLorentzVector& momentum)
 |      void ShipMCTrack::Get4Momentum(TLorentzVector& momentum)
 |
 |  GetEnergy(self)
 |      Double_t ShipMCTrack::GetEnergy()
 |
 |  GetMass(self)
 |      Double_t ShipMCTrack::GetMass()
 |
```

```
|  GetMomentum(self, TVector3& momentum)
|      void ShipMCTrack::GetMomentum(TVector3& momentum)
|
|  GetMotherId(self)
|      Int_t ShipMCTrack::GetMotherId()
|
|  GetNPoints(self, DetectorId detId)
|      Int_t ShipMCTrack::GetNPoints(DetectorId detId)
|
|  GetP(self)
|      Double_t ShipMCTrack::GetP()
|
|  GetPdgCode(self)
|      Int_t ShipMCTrack::GetPdgCode()
|
|  GetPt(self)
|      Double_t ShipMCTrack::GetPt()
|
|  GetPx(self)
|      Double_t ShipMCTrack::GetPx()
|
|  GetPy(self)
|      Double_t ShipMCTrack::GetPy()
|
|  GetPz(self)
|      Double_t ShipMCTrack::GetPz()
|
|  GetRapidity(self)
|      Double_t ShipMCTrack::GetRapidity()
|
|  GetStartT(self)
|      Double_t ShipMCTrack::GetStartT()
|
|  GetStartVertex(self, TVector3& vertex)
|      void ShipMCTrack::GetStartVertex(TVector3& vertex)
|
|  GetStartX(self)
|      Double_t ShipMCTrack::GetStartX()
|
|  GetStartY(self)
|      Double_t ShipMCTrack::GetStartY()
|
|  GetStartZ(self)
|      Double_t ShipMCTrack::GetStartZ()
|
|  GetWeight(self)
|      Double_t ShipMCTrack::GetWeight()
|
|  ImplFileLine(self)
|      static int ShipMCTrack::ImplFileLine()
|
| [...][PRINTOUT CUT]

Out[41]: ['AbstractMethod',
```

```
 'AppendPad',
 'Browse',
 [...]
 [PRINTOUT CUT]
 [...]
  'Get4Momentum',
 'GetDrawOption',
 'GetDtorOnly',
 'GetEnergy',
 'GetIconName',
 'GetMass',
 'GetMomentum',
 'GetMotherId',
 'GetNPoints',
 'GetName',
 'GetObjectInfo',
 'GetObjectStat',
 'GetOption',
 'GetP',
 'GetPdgCode',
 'GetPt',
 'GetPx',
 'GetPy',
 'GetPz',
 'GetRapidity',
 'GetStartT',
 'GetStartVertex',
 'GetStartX',
 'GetStartY',
 'GetStartZ',
 'GetTitle',
 'GetUniqueID',
 'GetWeight',
 [...]
 [PRINTOUT CUT]
 [...]
 'kOverwrite',
 'kSingleKey',
 'kWriteDelete',
 'kZombie',
 'ls',
 'operator delete',
 'operator delete[]',
 'operator new',
 'operator new[]']
```

In [6]:
```
# The ROOT PDG database allows quick identification of particles based on their code / name
pdg = r.TDatabasePDG.Instance()
for track in mcTracks:
    print track.GetPdgCode()
```

```
-411
9900015
-211
-13
```

```
11
```

```
In [7]: # The guy with the funny number above is an HNL.
        # The ROOT PDG database does not know about it. But FairShip has a function to add it.
        import pythia8_conf
        pythia8_conf.addHNLtoROOT()
        for track in mcTracks:
            print pdg.GetParticle(track.GetPdgCode()).GetName()
```

```
D-
N2
pi-
mu+
e-
```

```
In [8]: # Now let us have a look at what this event looks like.
        for i,track in enumerate(mcTracks):
            print i, pdg.GetParticle(track.GetPdgCode()).GetName(), track.GetMotherId()
```

```
0 D- -1
1 N2 0
2 pi- 1
3 mu+ 1
4 e- 3
```

```
In [9]: # We want a nicer printout. But careful: the beam does not have a mother!
        for i,track in enumerate(mcTracks):
            if i > 0:
                print( i, pdg.GetParticle(track.GetPdgCode()).GetName(), 'daughter of',
                 pdg.GetParticle(mcTracks[track.GetMotherId()].GetPdgCode()).GetName())
```

```
1 N2 daughter of D-
2 pi- daughter of N2
3 mu+ daughter of N2
4 e- daughter of mu+
```

```
In [10]: # Meet the true (simulated) HNL.
         myHNL = mcTracks[1]
         myHNL.GetPdgCode()
```

```
Out[10]: 9900015
```

```
In [11]: # Meet its daughters.
         for track in mcTracks:
             if track.GetMotherId() == 1:
                 print mcTracks.index(track)

         for i,track in enumerate(mcTracks):
             if track.GetMotherId() == 1:
                 print i, pdg.GetParticle(track.GetPdgCode()).GetName()
```

```
2
3
2 pi-
3 mu+
```

```
In [12]: # We do not feel like typing all that code anymore.
         def name(track):
             return pdg.GetParticle(track.GetPdgCode()).GetName()
         for i,track in enumerate(mcTracks):
             if track.GetMotherId() == 1:
                 print name(track)
```

```
pi-
mu+
```

```
In [13]: # Let's open the geometry file.
         # This file is generated by run_simScript.py along with the data file.
         # It is "unique" to your production. Please always refer to the same geometry you use
         # to generate your file, when analysing it.
         geo = r.TFile('geofile_full.10.0.Pythia8-TGeant4.root','read')
         geo.ls()
         sGeo = geo.FAIRGeom
         fGeo = r.gGeoManager
```

```
In [14]: # FindNode is a useful function. It takes x,y,z coordinates and tells you
         # to what geometry node they correspond.
         # For example, see where these particles originate.
         for track in mcTracks:
             print fGeo.FindNode(track.GetStartX(), track.GetStartY(), track.GetStartZ()).GetName()
```

```
Target_1_1
Target_1_1
cave_1
cave_1
rockD_1
```

```
In [15]: # Meet the basic brick of our geometry.
         fGeo.GetTopVolume()
         help(fGeo.GetTopVolume())
```

```
Help on TGeoVolume in module __main__ object:

class TGeoVolume(ROOT.TNamed, TGeoAtt, ROOT.TAttLine, ROOT.TAttFill, TAtt3D)
 |  Method resolution order:
 |      TGeoVolume
 |      ROOT.TNamed
 |      ROOT.TObject
 |      TGeoAtt
 |      ROOT.TAttLine
 |      ROOT.TAttFill
 |      TAtt3D
 |      ROOT.ObjectProxy
 |      __builtin__.object
 |
 |  Methods defined here:
 |
 |  AddNode(self, TGeoVolume* vol, Int_t copy_no, TGeoMatrix* mat=0, Option_t* option='')
 |      void TGeoVolume::AddNode(TGeoVolume* vol, Int_t copy_no, TGeoMatrix* mat = 0, Option_t* option =
 |
 |  AddNodeOffset(self, TGeoVolume* vol, Int_t copy_no, Double_t offset=0, Option_t* option='')
```

```
|      void TGeoVolume::AddNodeOffset(TGeoVolume* vol, Int_t copy_no, Double_t offset = 0, Option_t* opt
|
|  AddNodeOverlap(self, TGeoVolume* vol, Int_t copy_no, TGeoMatrix* mat=0, Option_t* option='')
|      void TGeoVolume::AddNodeOverlap(TGeoVolume* vol, Int_t copy_no, TGeoMatrix* mat = 0, Option_t* o
|
|  Browse(self, TBrowser* b)
|      void TGeoVolume::Browse(TBrowser* b)
|
|  Capacity(self)
|      Double_t TGeoVolume::Capacity()
|
|  CheckGeometry(self, Int_t nrays=1, Double_t startx=0, Double_t starty=0, Double_t startz=0)
|      void TGeoVolume::CheckGeometry(Int_t nrays = 1, Double_t startx = 0, Double_t starty = 0, Double
|
|  CheckOverlaps(self, Double_t ovlp=0.10000000000000001, Option_t* option='')
|      void TGeoVolume::CheckOverlaps(Double_t ovlp = 0.1, Option_t* option = "")
|
| [...][PRINTOUT CUT]
```

```
In [16]: # First children nodes.
         fGeo.GetTopVolume().GetNodes()
```

```
Out[16]: <ROOT.TObjArray object ("TObjArray") at 0x6e41000>
```

```
In [17]: # Actually we want to see their names...
         for node in fGeo.GetTopVolume().GetNodes():
             print node.GetName()
```

```
MuonShieldArea_1
rockS_1
rockD_1
TargetArea_1
magyoke_1
MCoil_1
DecayVolume_1
MagVolume_1
TimeDet_1
VetoTimeDet_1
Detector2_1
volMagneticSpectrometer_1
volBase_1
volGoliath_1
Veto_5
Tr1_1
Tr2_2
Tr3_3
Tr4_4
Ecal_1
Hcal_1
MuonDetector_1
```

```
In [18]: # Get the z coordinate of the center of a certain node.
         fGeo.GetTopVolume().GetNode('Veto_5')
         fGeo.GetTopVolume().GetNode('Veto_5').GetMatrix()
         fGeo.GetTopVolume().GetNode('Veto_5').GetMatrix().GetTranslation()
         zveto5_center = fGeo.GetTopVolume().GetNode('Veto_5').GetMatrix().GetTranslation()[2]
```

```
In [19]: # Get its half-width
         dzveto5 = fGeo.GetTopVolume().GetNode('Veto_5').GetVolume().GetShape().GetDZ()

In [42]: # Get the end point of this node.
         zveto5_end = zveto5_center + dzveto5
         # Get the starting point of the first tracking station.
         ztr1_start = (fGeo.GetTopVolume().GetNode('Tr1_1').GetMatrix().GetTranslation()[2] -
          fGeo.GetTopVolume().GetNode('Tr1_1').GetVolume().GetShape().GetDZ())

In [21]: # Reconstructed candidates are places in the Particles array.
         # In some events this array can be empty.
         t.Particles
         t.Particles[0]


        ---------------------------------------------------------------------------
    IndexError                                Traceback (most recent call last)

        <ipython-input-21-ea53e38b6172> in <module>()
          1 t.Particles
    ----> 2 t.Particles[0]


        IndexError: index out of range


In [22]: # Let's find the events with reconstructed particles.
         for i,event in enumerate(t):
             try:
                 p0 = event.Particles[0]
                 print 'I found a particle in event', i
             except:
                 continue

I found a particle in event 7
I found a particle in event 9
I found a particle in event 14
I found a particle in event 15
I found a particle in event 26
I found a particle in event 34
I found a particle in event 36
I found a particle in event 37
I found a particle in event 38
I found a particle in event 39
I found a particle in event 41
I found a particle in event 50
I found a particle in event 53
I found a particle in event 60
I found a particle in event 62
I found a particle in event 71
I found a particle in event 85
I found a particle in event 91
I found a particle in event 98
I found a particle in event 99
```

In [23]:
```
# These are ROOT TParticles. Here are all the methods you can use with them.
for p in t.Particles:
    print p
dir(t.Particles[0])
```

```
<ROOT.TParticle object ("N2") at 0xcb36d80>
```

Out[23]:
```
['AbstractMethod',
 'AppendPad',
 'Beauty',
 'Browse',
 'Charm',
 'Class',
 'ClassName',
 'Class_Name',
 'Class_Version',
 'Clear',
 'Clone',
 'Energy',
 'Eta',
 'Execute',
 'GetCalcMass',
 'GetDaughter',
 'GetDrawOption',
 'GetDtorOnly',
 'GetFirstDaughter',
 'GetFirstMother',
 'GetLastDaughter',
 'GetMass',
 'GetMother',
 'GetNDaughters',
 'GetName',
 'GetPDG',
 'GetPdgCode',
 'GetPolarisation',
 'GetSecondMother',
 'GetWeight',
 'Momentum',
 'P',
 'Phi',
 'ProductionVertex',
 'Pt',
 'Px',
 'Py',
 'Pz',
 'R',
 'Rho',
 'T',
 'Theta',
 'Vx',
 'Vy',
 'Vz',
 'Y',
 [...]
 [PRINTOUT CUT]
```

```
          [...]
          ]
```

In [24]:
```
# Let us take an event with one candidate.
t.GetEntry(98)
p0 = t.Particles[0]
```

In [25]:
```
# Closer look to the ROOT TParticle documentation
help(p0)
```

```
Help on TParticle in module __main__ object:

class TParticle(ROOT.TObject, ROOT.TAttLine, TAtt3D)
 |  Method resolution order:
 |      TParticle
 |      ROOT.TObject
 |      ROOT.TAttLine
 |      TAtt3D
 |      ROOT.ObjectProxy
 |      __builtin__.object
 |
 |  Methods defined here:
 |
 |  Beauty(self)
 |      Int_t TParticle::Beauty()
 |
 |  Charm(self)
 |      Int_t TParticle::Charm()
 |
 | [...][PRINTOUT]
```

In [26]:
```
# The simples analysis ever. Put selections in TTree::Draw.
# Here's how you do it in pyroot.
t.Draw("Particles.fVz", "")
t.Draw("Particles.fVz", "Particles.fVz<0")
t.Draw("Particles.fVz", "Particles.fVz>%s && Particles.fVz<%s"%(zveto5_end, ztr1_start))
```

Out[26]:
```
21L
```

In [27]:
```
# Let's start to do things in loops. E.G.: find the reconstructed vertices.
for event in t:
    for candidate in event.Particles:
        vtx = r.TLorentzVector()
        candidate.ProductionVertex(vtx)
        print "vertex:", vtx.X(), vtx.Y(), vtx.Z(), vtx.T()
```

```
vertex: -34.1353660525 101.253487404 2334.33488573 0.313047184129
vertex: 114.551541144 -143.387887678 2628.57601468 16.3379271866
vertex: 111.361009569 -146.593353991 2296.66518145 8.00779206009
vertex: 126.3541342 -146.079844991 2594.25561827 0.0684620240403
vertex: 126.305301325 -146.089403484 2594.4170677 0.0550978456631
vertex: -8.99156556423 124.39495533 -1691.61605198 0.57347611328
vertex: 60.9643880706 -210.665375627 -675.272949893 0.0550266433574
vertex: -41.7707950578 30.7375435303 1881.05927462 0.273677956344
vertex: -131.901364057 141.854154678 898.954546289 0.50887759781
vertex: 18.7553829227 -39.5128458767 1953.0398594 2.01671603084
```

```
vertex: -35.1481367213 93.1259452578 -1065.51400709 0.459928656441
vertex: 157.952798423 142.024760286 2372.08006999 20.1351966889
vertex: -180.585668633 77.9651632787 1858.63971671 0.421969374874
vertex: -102.246626687 -68.2069633716 -880.910974436 0.827645380593
vertex: 61.9443824766 47.0989463604 -1522.49593534 0.375078319848
vertex: 22.5934831798 0.556577333333 -820.447807401 0.0674950453175
vertex: 46.371858293 -96.1022501404 -439.156267684 0.676162627526
vertex: 3.45703367781 29.2206331138 -2486.57131982 0.0555349184644
vertex: -39.7338228897 -213.302469085 -17.8368662016 0.428616369111
vertex: 19.7920246958 -235.50873585 -1024.93246371 0.0189837563539
vertex: -67.4618473196 -28.7943541674 462.000297503 0.194734080379
vertex: 142.894941973 -59.1565393643 172.127096724 0.553946272779
vertex: 31.5993851039 121.719061476 2064.71341991 0.0781145947893
```

```
In [28]: # The fourth component of the reconstructed vertex is actually the DOCA,
         # that is made persistent this way.
         h_doca = r.TH1F('h_doca', 'h_doca', 100, 0., 103.)
```

```
In [29]: # Retrieve also the reconstructed momentum
         for event in t:
             for candidate in event.Particles:
                 vtx = r.TLorentzVector()
                 candidate.ProductionVertex(vtx)
                 print "vertex:", vtx.X(), vtx.Y(), vtx.Z(), vtx.T()
                 h_doca.Fill(vtx.T())
                 mom = r.TLorentzVector()
                 candidate.Momentum(mom)
                 print "momentum:", mom.X(), mom.Y(), mom.Z(), mom.T()

         # Draw the DOCA
         h_doca.Draw()
```

```
vertex: -34.1353660525 101.253487404 2334.33488573 0.313047184129
momentum: -0.109912174092 0.319334048877 35.4379088445 35.4536268487
vertex: 114.551541144 -143.387887678 2628.57601468 16.3379271866
momentum: -0.43823545764 -0.0404361722514 32.1235885664 32.1911412792
vertex: 111.361009569 -146.593353991 2296.66518145 8.00779206009
momentum: 0.797793083413 -0.123210070506 52.2479618392 52.2747841496
vertex: 126.3541342 -146.079844991 2594.25561827 0.0684620240403
momentum: -0.0641217791079 0.0361776523825 14.1977744222 14.2685860286
vertex: 126.305301325 -146.089403484 2594.4170677 0.0550978456631
momentum: 1.1719183061 -0.0465942909559 34.3203398816 34.3515786624
vertex: -8.99156556423 124.39495533 -1691.61605198 0.57347611328
momentum: -0.0547348069666 0.671682814873 38.7947464921 38.8133698451
vertex: 60.9643880706 -210.665375627 -675.272949893 0.0550266433574
momentum: 0.281964977243 -0.979970327465 38.6481173522 38.6744883466
vertex: -41.7707950578 30.7375435303 1881.05927462 0.273677956344
momentum: -0.0767453674971 0.0566993045415 19.1264527276 19.1526877731
vertex: -131.901364057 141.854154678 898.954546289 0.50887759781
momentum: -0.618579730539 0.656660800484 45.5482151575 45.5679453591
[...][PRINOTOUT CUT]
```

```
In [30]: # For some specific applications (use at your own risk),
         # or for when you do not have a geometry file,
         # you can retrieve geometrical positions using this script from FairShip.
```

```
In [30]: from ShipGeoConfig import ConfigRegistry
         ShipGeo = ConfigRegistry.loadpy("$FAIRSHIP/geometry/geometry_config.py", Yheight = 10.)
         dir(ShipGeo)
```

```
Out[30]: ['Bfield',
          'Chamber1',
          'Chamber2',
          'Chamber3',
          'Chamber4',
          'Chamber5',
          'Chamber6',
          'HcalOption',
          'Muon',
          'MuonFilter0',
          'MuonFilter1',
          'MuonFilter2',
          'MuonStation0',
          'MuonStation1',
          'MuonStation2',
          'MuonStation3',
          'NuTauTarget',
          'TrackStation1',
          'TrackStation2',
          'TrackStation3',
          'TrackStation4',
          'Yheight',
          '__class__',
          '__cmp__',
          '__contains__',
          '__delattr__',
          '__delitem__',
          '__dict__',
          '__doc__',
          [...]
          [PRINTOUT CUT]
          [...]
          'muShield',
          'muShieldDesign',
          'pop',
          'popitem',
          'scintillator',
          'setdefault',
          'straw',
          'strawDesign',
          'strawtubes',
          'target',
          'targetOpt',
          'tauMS',
          'update',
          'values',
          'vetoStation',
          'z']
```

```
In [31]: # Let's do something serious and compute the impact parameter to the target.
         def IPtoTarget(vtx, mom):
```

```
            p = mom.P()
            target = r.TVector3(0., 0., ShipGeo.target.z0)
            delta = 0.
            for i in range(3):
                delta += (target(i) - vtx(i)) * mom(i)/p
            ip = 0.
            for i in range(3):
                ip += (target(i) - vtx(i) - delta*mom(i)/p)**2.
            return r.TMath.Sqrt(ip)
```

In [32]:
```
# Let's look at the IPs in order to define a selection.
for event in t:
        for candidate in event.Particles:
                vtx = r.TLorentzVector()
                candidate.ProductionVertex(vtx)
                print "vertex:", vtx.X(), vtx.Y(), vtx.Z(), vtx.T()
                mom = r.TLorentzVector()
                candidate.Momentum(mom)
                print "momentum:", mom.X(), mom.Y(), mom.Z(), mom.T()
                print 'IP:', IPtoTarget(vtx, mom)
```

```
vertex: -34.1353660525 101.253487404 2334.33488573 0.313047184129
momentum: -0.109912174092 0.319334048877 35.4379088445 35.4536268487
IP: 0.82301980716
vertex: 114.551541144 -143.387887678 2628.57601468 16.3379271866
momentum: -0.43823545764 -0.0404361722514 32.1235885664 32.1911412792
IP: 301.127242483
vertex: 111.361009569 -146.593353991 2296.66518145 8.00779206009
momentum: 0.797793083413 -0.123210070506 52.2479618392 52.2747841496
IP: 134.282928964
vertex: 126.3541342 -146.079844991 2594.25561827 0.0684620240403
momentum: -0.0641217791079 0.0361776523825 14.1977744222 14.2685860286
IP: 250.197820407
vertex: 126.305301325 -146.089403484 2594.4170677 0.0550978456631
momentum: 1.1719183061 -0.0465942909559 34.3203398816 34.3515786624
IP: 297.12729042
vertex: -8.99156556423 124.39495533 -1691.61605198 0.57347611328
momentum: -0.0547348069666 0.671682814873 38.7947464921 38.8133698451
IP: 1.49957861078
vertex: 60.9643880706 -210.665375627 -675.272949893 0.0550266433574
momentum: 0.281964977243 -0.979970327465 38.6481173522 38.6744883466
IP: 1.62235947922
vertex: -41.7707950578 30.7375435303 1881.05927462 0.273677956344
momentum: -0.0767453674971 0.0566993045415 19.1264527276 19.1526877731
IP: 2.06053608453
vertex: -131.901364057 141.854154678 898.954546289 0.50887759781
momentum: -0.618579730539 0.656660800484 45.5482151575 45.5679453591
IP: 1.55481921548
[...][PRINTOUT CUT]
```

In [33]:
```
h_ip = r.TH1F('h_ip', 'h_ip', 100, 0., 2000.)
ips = []
for event in t:
        for candidate in event.Particles:
                vtx = r.TLorentzVector()
```

```
                        candidate.ProductionVertex(vtx)
                        print "vertex:", vtx.X(), vtx.Y(), vtx.Z(), vtx.T()
                        mom = r.TLorentzVector()
                        candidate.Momentum(mom)
                        print "momentum:", mom.X(), mom.Y(), mom.Z(), mom.T()
                        print 'IP:', IPtoTarget(vtx, mom)
                        h_ip.Fill(IPtoTarget(vtx, mom))
                        ips.append(IPtoTarget(vtx, mom))
        ips
        h_ip.Draw()
```

```
vertex: -34.1353660525 101.253487404 2334.33488573 0.313047184129
momentum: -0.109912174092 0.319334048877 35.4379088445 35.4536268487
IP: 0.82301980716
vertex: 114.551541144 -143.387887678 2628.57601468 16.3379271866
momentum: -0.43823545764 -0.0404361722514 32.1235885664 32.1911412792
IP: 301.127242483
vertex: 111.361009569 -146.593353991 2296.66518145 8.00779206009
momentum: 0.797793083413 -0.123210070506 52.2479618392 52.2747841496
IP: 134.282928964
[...][PRINTOUT CUT]
```

In [34]:
```
# Let's prepare a selection based on the fiducial volume.
def in_vessel(vtx):
    x, y, z = vtx.X(), vtx.Y(), vtx.Z()
    if z < zveto5_end: return False
    if z > ShipGeo.TrackStation1.z - 5.: return False
    if (x/245.)**2 + (y/495.)**2 > 1: return False
    return True

total = 0
selected = 0
for event in t:
    for candidate in event.Particles:
        total += 1
        vtx = r.TLorentzVector()
        mom = r.TLorentzVector()
        candidate.ProductionVertex(vtx)
        candidate.Momentum(mom)
        if in_vessel(vtx) and IPtoTarget(vtx, mom) < 250.:
            selected += 1
print total, selected
```

```
23 19
```

In [35]:
```
# Let's introduce weights.
# This is the recipe to compute the HNL acceptance.
# We produce a lot of HNLs, and we use the weights to model the exponential
# distribution of their decay position.
total = 0
selected = 0
wtotal = 0.
wselected = 0.
for event in t:
    for candidate in event.Particles:
```

```
                total += 1
                wtotal += event.MCTrack[1].GetWeight()
                vtx = r.TLorentzVector()
                mom = r.TLorentzVector()
                candidate.ProductionVertex(vtx)
                candidate.Momentum(mom)
                if in_vessel(vtx) and IPtoTarget(vtx, mom) < 250.:
                    selected += 1
                    wselected += event.MCTrack[1].GetWeight()
        print total, selected
        print wtotal/t.GetEntries(), wselected/t.GetEntries()
```

```
23 19
7.51578242671e-06 6.76745660712e-06
```

```
In [36]: # But wait, did we match the reconstructed candidate to the true HNL?
         def match2MC(event, candidate):
             d1, d2 = candidate.GetDaughter(0), candidate.GetDaughter(1)
             # the fitTrack2MC leave is an array of the same length as FitTracks.
             # it contains the index of the corresponding MCTrack.
             d1_mc, d2_mc = event.fitTrack2MC[d1], event.fitTrack2MC[d2]
             for d in [d1_mc, d2_mc]:
                 mum = event.MCTrack[d].GetMotherId()
                 if not pdg.GetParticle(event.MCTrack[mum].GetPdgCode()).GetName() == 'N2':
                     return False
             return True
```

```
In [37]: total = 0
         selected = 0
         wtotal = 0.
         wselected = 0.
         for event in t:
             for candidate in event.Particles:
                 if not match2MC(event, candidate): continue
                 total += 1
                 wtotal += event.MCTrack[1].GetWeight()
                 vtx = r.TLorentzVector()
                 mom = r.TLorentzVector()
                 candidate.ProductionVertex(vtx)
                 candidate.Momentum(mom)
                 if in_vessel(vtx) and IPtoTarget(vtx, mom) < 250.:
                     selected += 1
                     wselected += event.MCTrack[1].GetWeight()
         print total, selected
         print wtotal/t.GetEntries(), wselected/t.GetEntries()
```

```
17 16
5.4142179124e-06 5.14913724146e-06
```

```
In [38]: # By the way: reconstructed tracks are here.
         t.FitTracks[0]
```

```
Out[38]: <ROOT.genfit::Track object ("genfit::Track") at 0x1092f310>
```

```
In [39]: # How to quickly implement vetoes.
         import shipVeto
```

```
        veto = shipVeto.Task()
        total = 0
        selected = 0
        wtotal = 0.
        wselected = 0.
        for event in t:
            # veto decisions have weights.
            # They represent the probability, given a certain veto efficiency,
            # that an event passes a VETO decision.
            sbt, sbtw, sbtn = veto.SBT_decision(event)
            print 'SBT: veto decision ', sbt, 'with weight', sbtw
            uvt, uvtw, uvtn = veto.UVT_decision(event)
            print 'UVT: veto decision ', uvt, 'with weight', uvtw
            svt, svtw, svtn = veto.SVT_decision(event)
            print 'SVT: veto decision ', svt, 'with weight', svtw
            # If decision is True, event is vetoed!
            if sbt or uvt or svt: continue
            for candidate in event.Particles:
                if not match2MC(event, candidate): continue
                total += 1
                wtotal += event.MCTrack[1].GetWeight()
                vtx = r.TLorentzVector()
                mom = r.TLorentzVector()
                candidate.ProductionVertex(vtx)
                candidate.Momentum(mom)
                if in_vessel(vtx) and IPtoTarget(vtx, mom) < 250.:
                    selected += 1
                    wselected += event.MCTrack[1].GetWeight()
        print total, selected
        print wtotal/t.GetEntries(), wselected/t.GetEntries()
```

```
SBT: veto decision  True with weight 0.01
UVT: veto decision  False with weight 1.0
SVT: veto decision  True with weight 1.86264514923e-67
SBT: veto decision  False with weight 1.0
UVT: veto decision  False with weight 1.0
SVT: veto decision  False with weight 1.0
SBT: veto decision  False with weight 1.0
UVT: veto decision  False with weight 1.0
SVT: veto decision  False with weight 1.0
SBT: veto decision  True with weight 0.0001
UVT: veto decision  False with weight 1.0
SVT: veto decision  False with weight 1.0
SBT: veto decision  True with weight 0.01
UVT: veto decision  False with weight 1.0
SVT: veto decision  False with weight 1.0
SBT: veto decision  True with weight 1e-14
UVT: veto decision  False with weight 1.0
SVT: veto decision  False with weight 1.0
SBT: veto decision  False with weight 1.0
UVT: veto decision  False with weight 1.0
SVT: veto decision  False with weight 1.0
SBT: veto decision  False with weight 1.0
UVT: veto decision  False with weight 1.0
```

```
SVT: veto decision  False with weight 1.0
SBT: veto decision  True with weight 1e-10
UVT: veto decision  False with weight 1.0
SVT: veto decision  False with weight 1.0
SBT: veto decision  True with weight 0.01
UVT: veto decision  False with weight 1.0
SVT: veto decision  False with weight 1.0
SBT: veto decision  False with weight 1.0
UVT: veto decision  False with weight 1.0
SVT: veto decision  False with weight 1.0
SBT: veto decision  False with weight 1.0
UVT: veto decision  False with weight 1.0
SVT: veto decision  False with weight 1.0
SBT: veto decision  True with weight 1e-06
UVT: veto decision  False with weight 1.0
SVT: veto decision  True with weight 2.44140625e-28
SBT: veto decision  True with weight 1e-18
UVT: veto decision  True with weight 1e-06
SVT: veto decision  True with weight 3.46944695195e-134
SBT: veto decision  False with weight 1.0
UVT: veto decision  False with weight 1.0
SVT: veto decision  False with weight 1.0
SBT: veto decision  False with weight 1.0
UVT: veto decision  False with weight 1.0
SVT: veto decision  False with weight 1.0
SBT: veto decision  True with weight 0.01
[...][PRINTOUT CUT]
14 14
4.72911772704e-06 4.72911772704e-06

In [40]: print wtotal/t.GetEntries(), wselected/t.GetEntries()

4.72911772704e-06 4.72911772704e-06
```