# *aa(fit) software spin-off's*

Aart Heijboer, Nikhef

# Data classes / ntuple format

***historical***

- Needed easy-to-use set of classes for hits, tracks for use inside reconstruction and (high level) analysis
- Fully integrated in ROOT → data format for free!
- Proven in real-life analysis (Antares Point Source Search and several others) and reconstruction code (e.g. inside aafit SeaTray module).

***main design ideas:***

- Allow same classes to be used in very wide scope
  - hit-level reconstruction code in compiled c++ (aafit)
  - from within cint/pyroot for fast scripting
  - using TTree::Draw for fast, throw-away analysis

- → do lot's of work work with only a few classes

- Minimize reliance on external tools
  - if ROOT can do what we need, don't reinvent the wheel
  - ROOT is a good thing to learn well for all students to learn well and used everywhere.

# Features of aa-classes

- Only a hand-full of classes to do everything: **Vec, Spe, Trk, Evt, (Cfg)**

- They are kept *as simple as possible*:
  - no inheritance, no templates, no (unnecessary) data hiding
  - short names: helpful when doing TTree::Draw (example later)
    and prevents RSI
  - pragmatic
    example : Each Evt has container for e.g. MC tracks
    it is just empty if there is no such information – same for hits.

- Fully integrated in ROOT → TTree of Evt = file-format

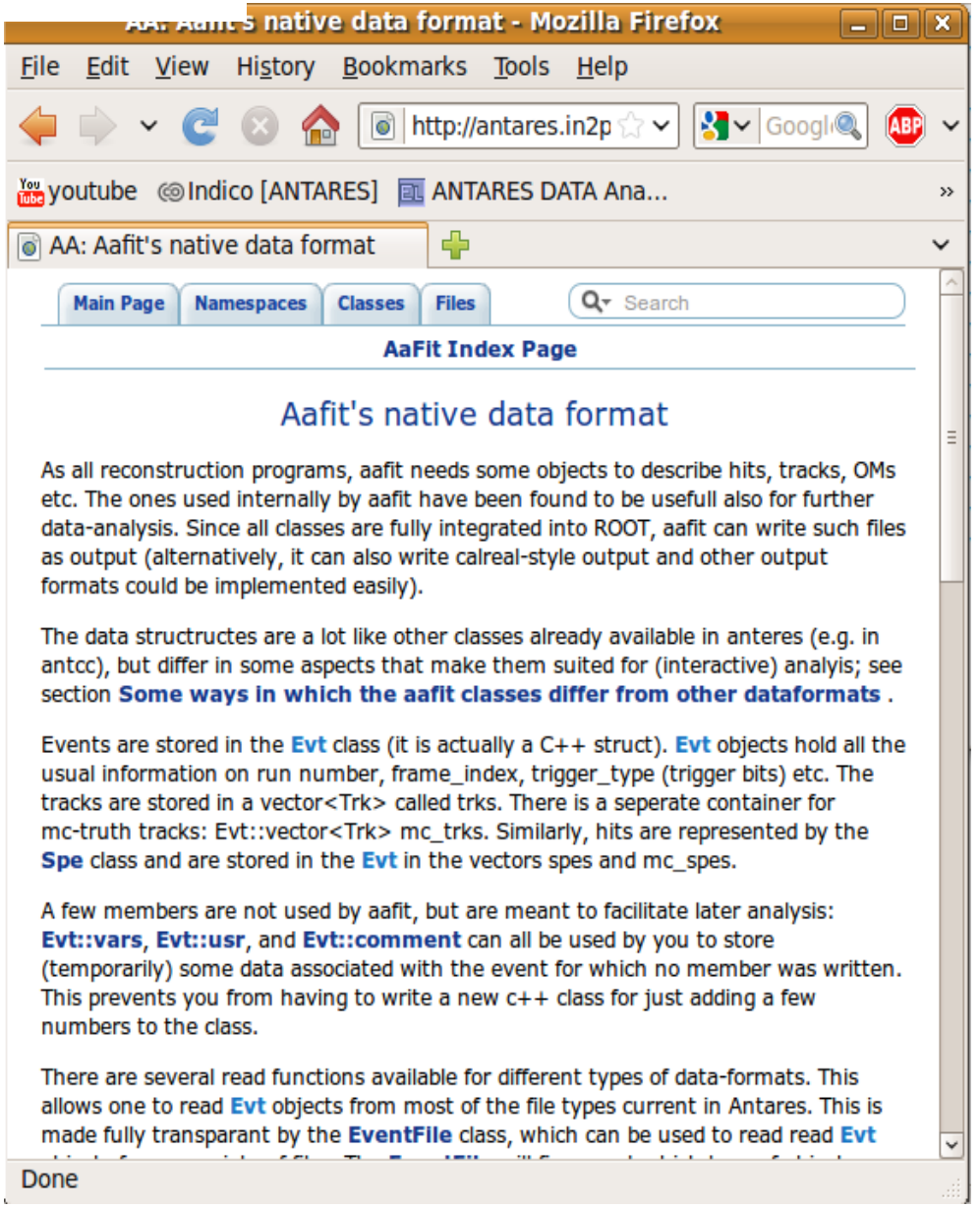
- a few special python methods to work nicely with pyroot

- IO provided to many data formats in ANTARES:
  - aafit native output files (trivially)
  - ascii evt files
  - raw data (PhysiscEvent)
  - MC-files from MCEventWriter (McEvent)
  - Calreal-style files (FullEvent)
  - AntDst
  - I3 files (python only for now

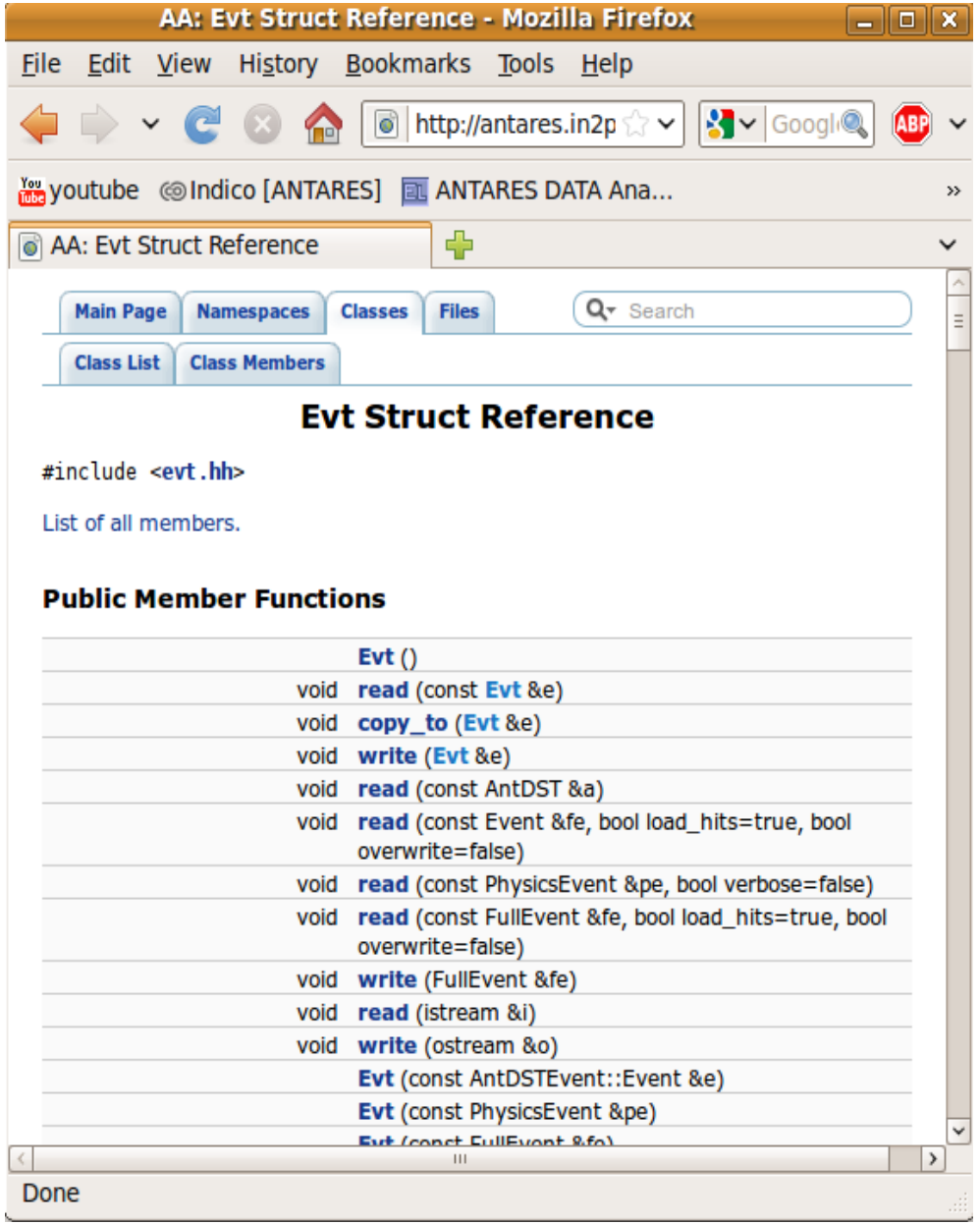
  → clearly not as simple as possible, but all this IO is *vital* to get work done
  (would clean it up for k3mnet)

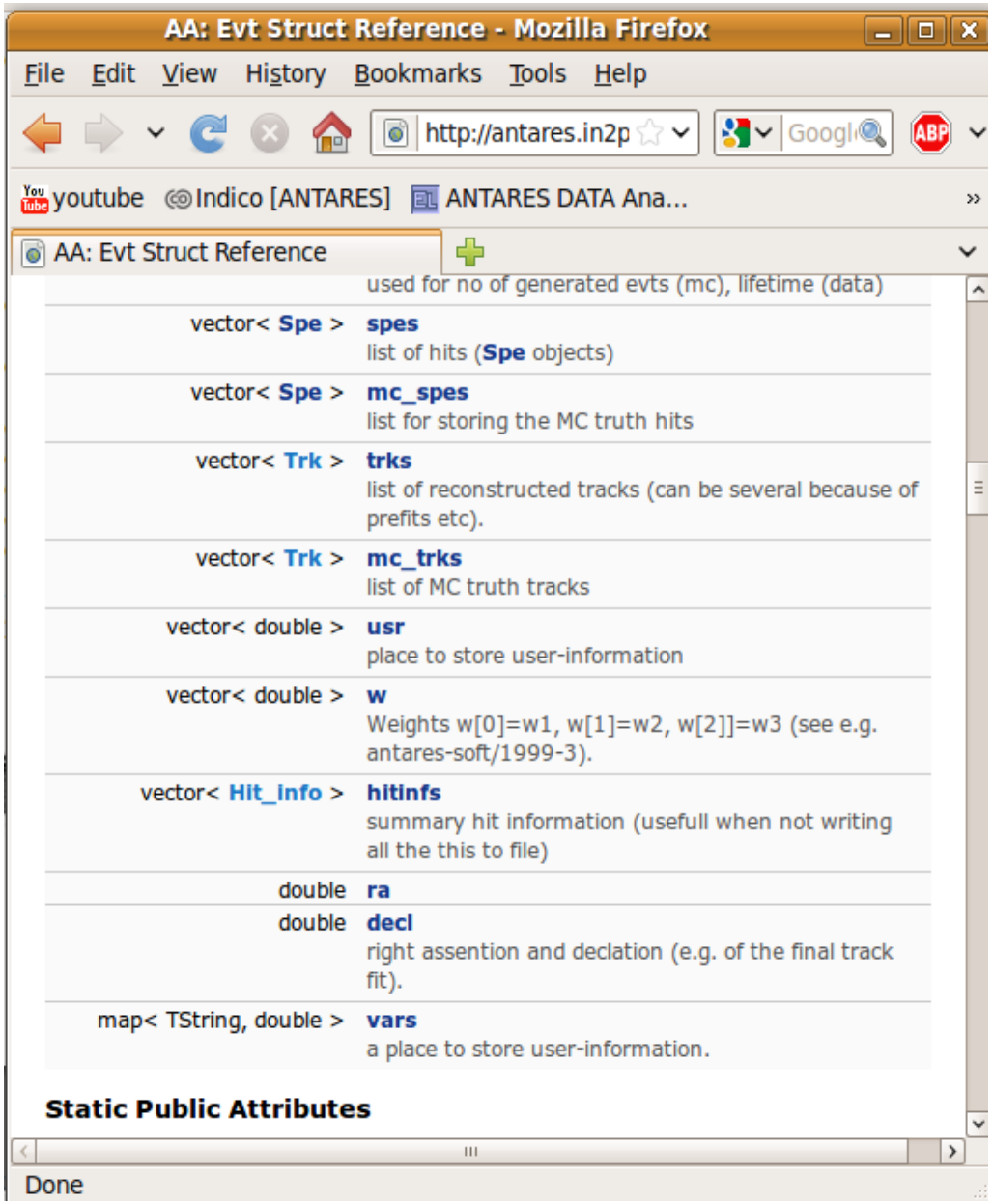
# Features of aa-classes - documentation

## http://antares.in2p3.fr/users/heijboer/aadox/

# Features of aa-classes - documentation



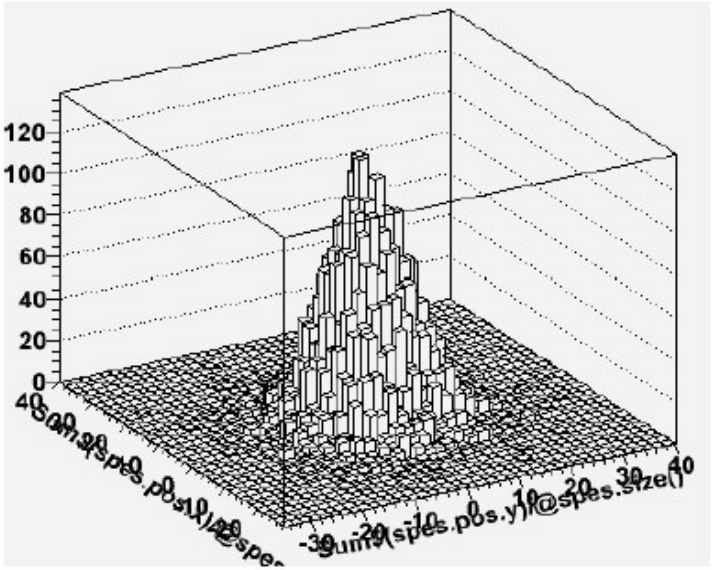
# Example

```
//c++

Vec cg( Evt& evt )
{
  Vec r;

  for (vector<Spe>::iterator ihit = evt.spes.begin() ;
       ihit!= evt.spes.end() ; ihit++)
    {
      r += (*ihit).pos;
    }
  return r/spe.size();
}

// Python / PyRoot

def cg ( evt ):
  r = Vec()
  for h in evt.spes:          # loop over std::vector<Spe>
          r += h.pos          # calls Vec::__add__()
  return r / spes.size()


// nicer Python

def cg (evt):
  return sum ( h.pos for h in evt.spes ) / spes.size()


// TTree
f = TFile("file.aav0r9.withhits.root") // dont need to lood libs
E.Draw("Sum$(spes.pos.x)/@spes.size():Sum$(spes.pos.y)/@spes.size()",
       "","lego")
```

Use same idea/code in both c++ and python.

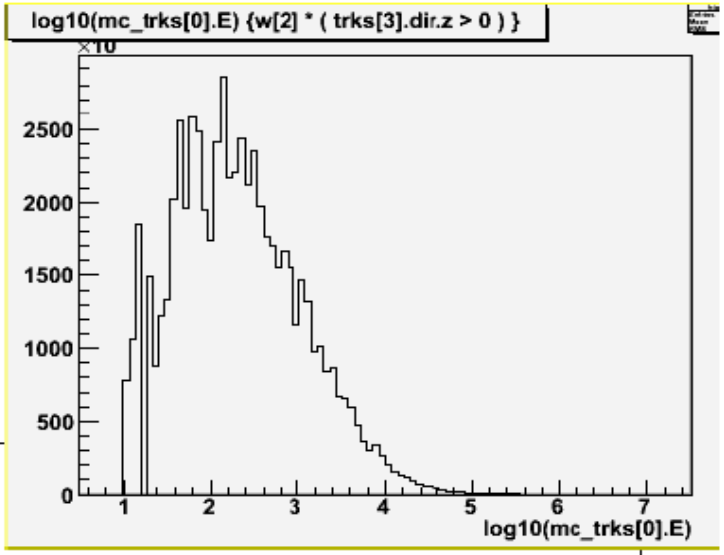
Looping over events is made very easy (not shown here)

TTree::Draw can be very powerful: useful to learn (use everywhere in HEP)



# Example

Let's *Draw the energy spectrum of upgoing-reconstructed atmpheric neutrinos*




```
// aa ntuple file

E.Draw("log10(mc_trks[0].E)","w[2] * ( trks[3].dir.z > 0 ) ")

// antdst

AntData.Draw("log10(fMCEvent.fParticles.fEnergy)",
             "(fMCEvent.fWeight_W3) * (fMCEvent.fParticles.
              fParticleType==-14) * (fRecEvents.fRecParticles.
              fZenith > 3.1415/2)")
```

aa ntuple designed with this in mind:
- short names
- very simple class structure

```
// other example draw reconstruction error
E.Draw("acos(mc_trks[0].dir.x*trks[3].dir.x+
       mc_trks[0].dir.y*trks[3].dir.y+mc_trks[0].dir.z*trks[3].dir.z)")
```

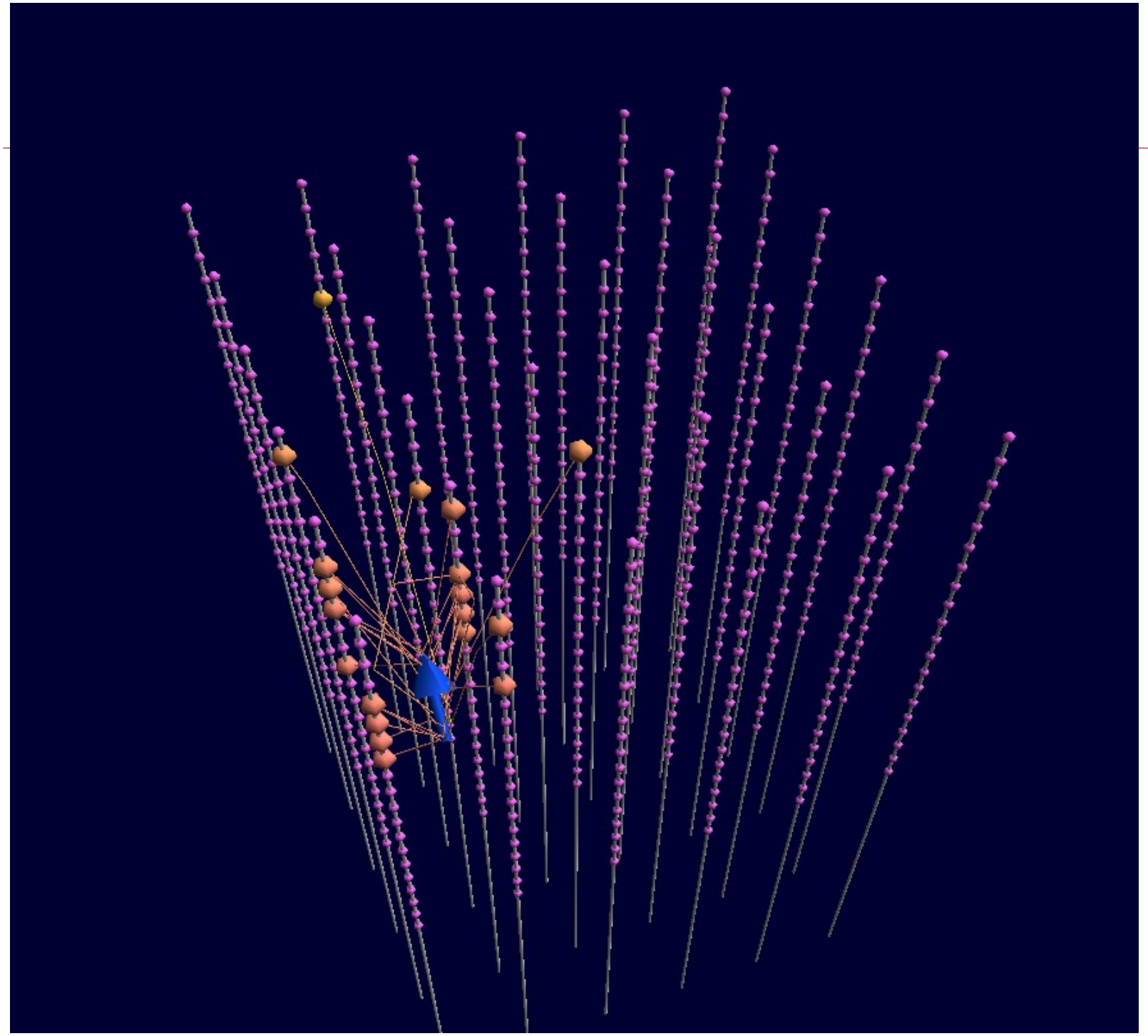
# Other codes available

- **Event Display**
  - OpenGL via ROOT (fast, also on remote server)
  - difference: ~no GUI, but fully controlled from user analysis code
  - has been very useful for development & debugging
    (really easy if you can quickly see what your script is doing!)
  - can be developed further to polished product for KM3NeT (if manpower)

- **geant4 + photon tracking code (aasim)**
  - fast photon tracking code for full photon propagation
  - interfaced with geant4
  - has been useful in some niches, like ORCA and water-properties studies
  - probably not for HE astrophysics.

- **point-source pseudo-experiment classes and cluster fitting code.**
  - you can make point source analyses with them

- **ORCA**
  code for oscillation fit, oscillation probabilities and pseudo-experiments

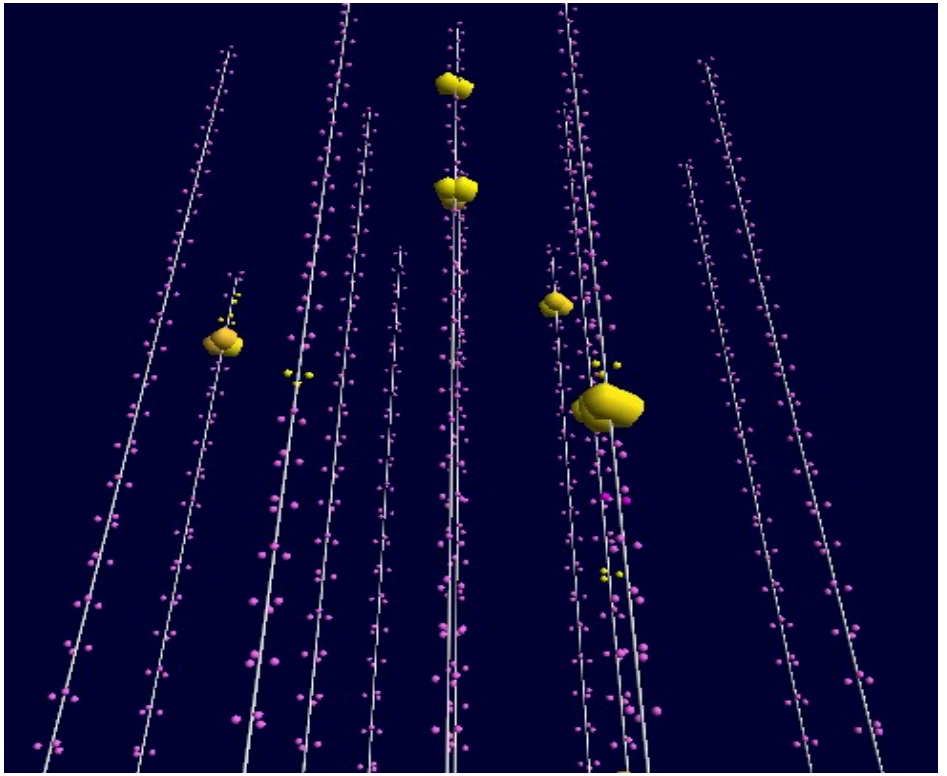


event display
of aasim event
(for orca)

# Example: display events from some antdst file

```
# - load event display libary, load detector

if gSystem.Load('../g/gui.so') < 0   : sys.exit()
display = A3D()

cfg = Cfg("r12_c00_s01.det")
display.set( cfg )


# use helper EventFile class to read antdst transparently

input_file = "/sps/km3net/mc/rbr/v0.1/seatray_v0.1.1/out/MC_025800_-14_6776_151.RECO_dst.root"
f = EventFile( input_file )

while f.next():

    # show only hits with amplitude > 10
    for h in f.evt.spes :
        if h.a < 10 : h.a = 0

    # display the event
    display.set( f.evt )

print 'bye'
```

control display directly from event loop : (more useful than clicking through a file with events)



# Conclusions

- ***Event classes for use in data-analysis***
  - mature & well tested in practice
  - modification for KM3NeT could be easy
  - **propose to make them an official KM3NeT tool set + ntuple format**

- ***Photon tracking (+Geant 4)***
  - “experimental” software.
  - useful for specialized jobs
  - no immediate effort to use for HE astronomy
  - could release version if interest (e.g. for orca)

- ***Event display***
  - new approach: driven from user analysis script
  - very useful for developers
    - propose to maintain simple version with ntuple format
  - full version with complete GUI needs manpower (not me)

- for the rest: just ask..