The ROOT System

A Data Access & Analysis Framework

Introduction

Graphics, Histograms, I/O, Trees

3-4-5 February 2004

René Brun CERN/PH

http://root.cern.ch

Plan of course

Tuesday

- 9h00 Introduction with examples (Rene)
- 10h30 break
- 10h50 Graphics (Rene)
- 12h15 break----------
- 14h00 basic framework classes and tools (Fons)
- 15h00 Histograms (Rene)
- 16h00 break
- 16h20 Fitting Exercises(Rene)
- 17h30 end day 1

Wednesday

- 9h00 I/O, Trees introduction (Rene)
- 10h30 break
- 10h50 exercises with histograms I/O and simple Trees (Rene)
- 12h15 break----------
- 14h00 rootcint dictionary (Fons)
- 15h00 Event&Tree (Rene)
- 16h00 break
- 16h20 Exercises (Rene)
- 17h30 end day 2

Thursday

- 9h00 Selectors Remote files, PROOF (Fons)
- 10h30 break
- 10h50 Exercises with selectors(Fons)
- 12h15 break----------
- 14h00 GUI (Fons)
- 15h30 break
- 15h50 Exercises (Fons)
- 17h30 end day 3

Introduction

The ROOT web pages

http://root.cern.ch

- General Information and News
- Download source and binaries
- Howto & tutorials
- User Guide & Reference Guides
- Roottalk Digest & Forum

Mailing lists

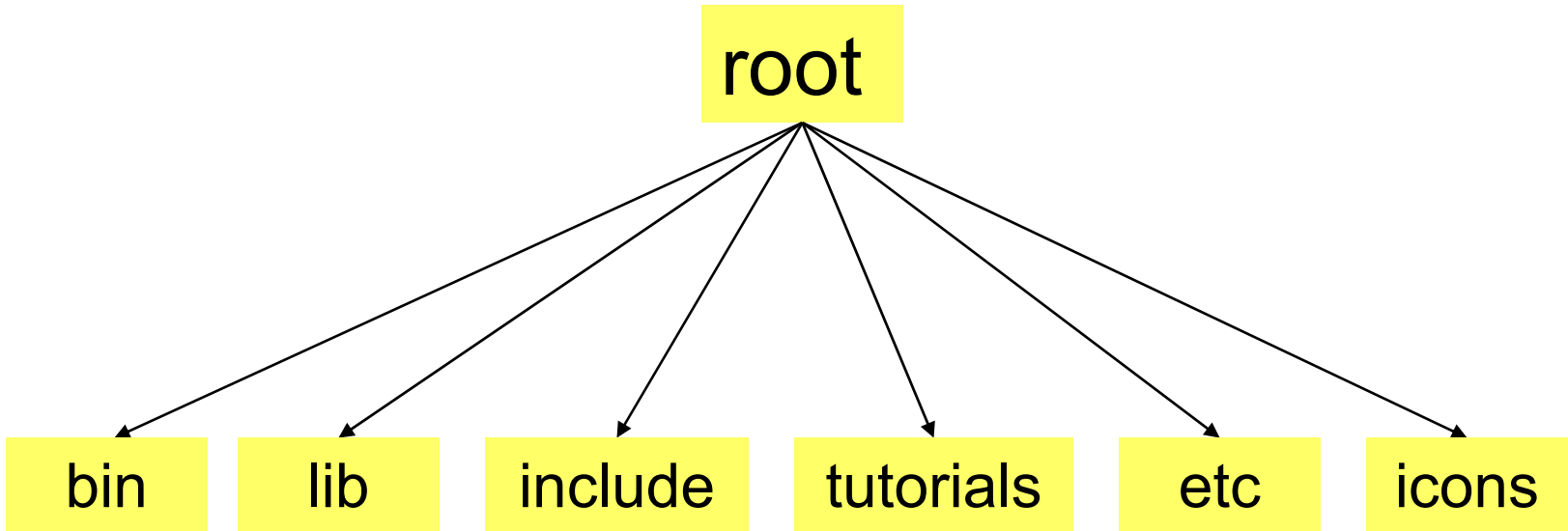
- roottalk@root.cern.ch
  - Use it to ask a question to a wide public or give information of general interest. Read by a few thousand people.
  - See also: http://root.cern.ch/root/roottalk/AboutRootTalk.html
- Root-Forum http://root.cern.ch/phpBB2
  - Same as roottalk with pros &cons
- rootdev@root.cern.ch
  - Use it to report bugs or ask question of non general interests (received by Root developers)

Installing ROOT

- cd $HOME
- with the web browser, download from
  - http://root.cern.ch/root/Version400.html
  - take Intel x86 Linux for Redhat 7.3 and gcc 3.2, version 4.00/01
- echo $ROOTSYS
- echo $PATH
- echo $LD_LIBRARY_PATH
- tar zxvf root_v4.00.01.Linux.RH7.3.gcc32.tar.gz
- cd root/tutorials
- check that Root runs correctly with the small session:
- root

```
root [0]  .x hsum.C
root [1]  .q
```

ROOT directories



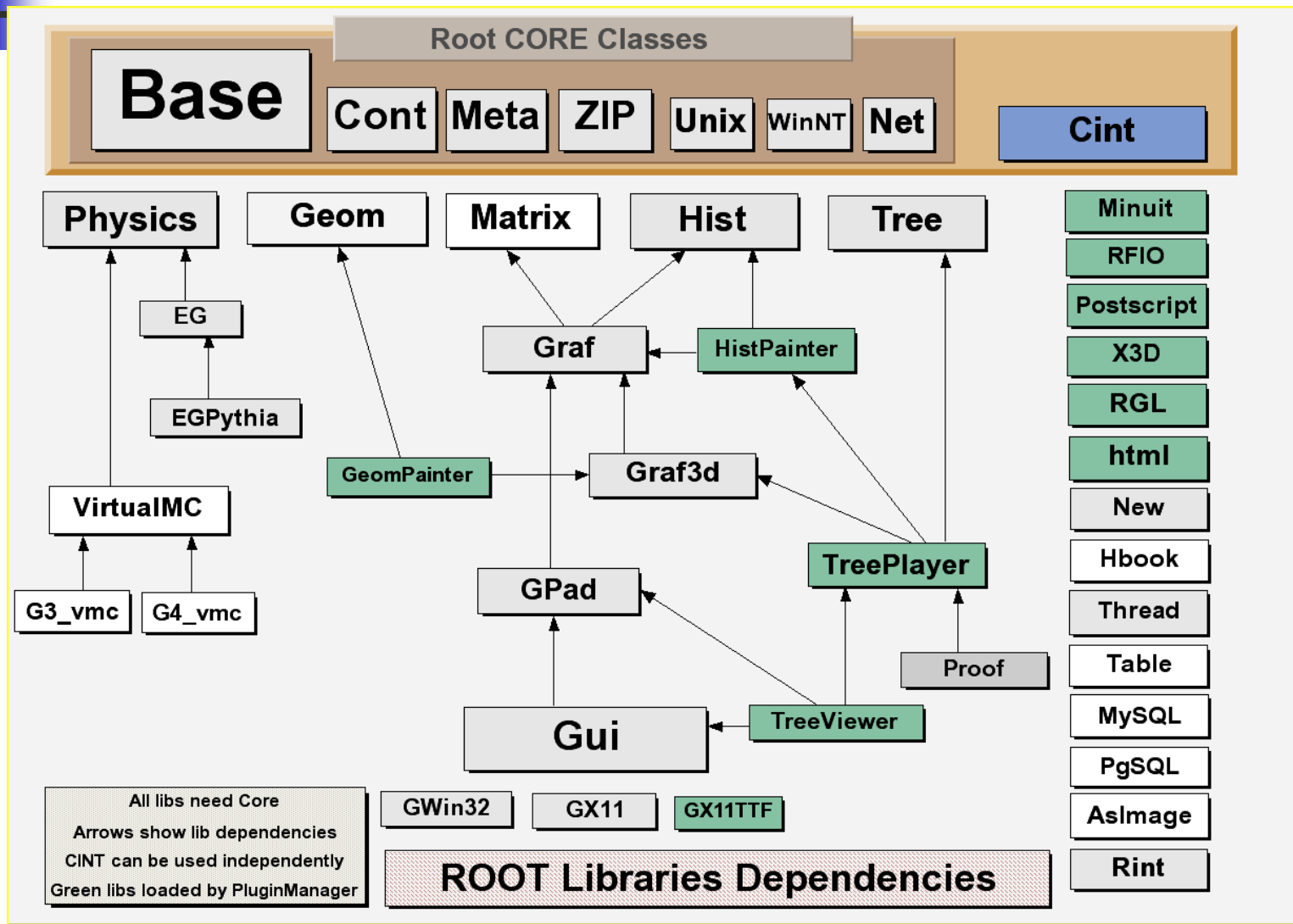
root/bin

```
  14301 cint
 810191 g2root
 801889 g2rootold
1012792 h2root
  20075 hadd
  39133 makecint
   8447 memprobe
 532593 proofd
  15014 proofserv
  24635 rlibmap
  40846 rmkdepend
  33465 root
  17948 rootcint
  14572 root-config
 551416 rootd
  14997 root.exe
  15005 rootn.exe
  56555 rpasswd
  47504 rtconf
  16972 ssh2rpd
```

root/lib

```
253662  libAfterImage.a
 439143 libASImage.so
1485107 libCint.so
6857376 libCore.so
 109262 libDCache.so
 173371 libEGPythia6.so
 155953 libEGPythia.so
 410174 libEG.so
 172332 libEGVenus.so
 414624 libfreetype.a
 103093 libFumili.so
 354436 libGeomPainter.so
3032831 libGeom.so
1094840 libGpad.so
1124799 libGraf3d.so
2190131 libGraf.so
4394725 libGui.so
 313423 libGX11.so
 136738 libGX11TTF.so
1533640 libHbook.so
 430610 libHistPainter.so
2640332 libHist.so
 290454 libHtml.so
```

```
 339895 libKrb5Auth.so
1483241 libMatrix.so
 334854 libMinuit.so
 252002 libMLP.so
 271724 libMySQL.so
  19390 libNew.so
  89756 libPgSQL.so
 380058 libPhysics.so
 210978 libPostscript.so
  49342 libProofGui.so
 668370 libProof.so
 725997 libRFIO.so
  84608 JlibRGL.so
 191666 libRint.so
 124937 libRLDAP.so
41643206 libRoot.a
  49506 libSRPAuth.so
1372791 libTable.so
 315816 libThread.so
 429060 libTreePlayer.so
1337527 libTree.so
 426514 libTreeViewer.so
 283433 libVMC.so
 147669 libX3d.so
```


Root CORE Classes
Base
Cont
Meta
ZIP
Unix
WinNT
Net
Cint
Physics
Geom
Matrix
Hist
Tree
Minuit
RFIO
Postscript
X3D
RGL
html
New
Hbook
Thread
Table
MySQL
PgSQL
AsImage
Rint
EG
EGPythia
Graf
HistPainter
GeomPainter
Graf3d
VirtualMC
G3_vmc
G4_vmc
GPad
TreePlayer
Proof
Gui
TreeViewer
All libs need Core
Arrows show lib dependencies
CINT can be used independently
Green libs loaded by PluginManager
GWin32
GX11
GX11TTF
ROOT Libraries Dependencies

root/tutorials (195)

```
alien.C                 draw2dopt.C             geant3tasks.C           img2pad.C           pclient.C           sqlfilldb.C
analyze.C               DynamicSlice.C          geometry.C              imgconv.C           peaks.C             sqlselect.C
anim.C                  EditorBar.C             geoshapes.C             io.C                PhaseSpace.C        staff.C
approx.C                ellipse.C               gerrors2.C              jets.C              principal.C         surfaces.C
archi.C                 eval.C                  gerrors.C               labels1.C           pserv.C             tasks.C
arrow.C                 event.C                 graph2derrorsfit.C      labels2.C           psexam.C            T.C
basic3d.C               exec1.C                 graph2dfit.C            langaus.C           pstable.C           tcl.C
basic.C                 exec2.C                 graphApply.C            latex2.C            pythiaExample.C     TestAuth.C
basic.dat               exec3.C                 graph.C                 latex3.C            pythiaExample_C.d   testrandom.C
benchmarks.C            FeldmanCousins.C        greyscale.C             latex.C             Quad.cxx            threads.C
bent.C                  feynman.C               guitest.C               LDAPExample.C       Quad.h              timeonaxis2.C
bill.C                  fildir.C                h1analysis.C            limit.C             quantiles.C         timeonaxis.C
bug.C                   file.C                  h1analysis.h            logscales.C         quarks.C            tornado.C
c1.C                    fillrandom.C            h1chain.C               manyaxis.C          Reset               tree0.C
c1.eps                  fillrandom.root         h1draw.C                markerwarning.C     rootalias.C         tree1.C
canvas.C                first.C                 hadd.C                  MDF.C               rootenv.C           tree2a.C
cernbuild.C             FirstContour.C          hadd_old.C              miller.C            rootgeom.C          tree2.C
cernstaff.C             fit1.C                  hbars.C                 mlpHiggs.C          rootlogoff.C        tree3.C
cernstaff.dat           fit1_C.C                hclient.C               mlpHiggs.root       rootlogon.C         tree4.C
classcat.C              fit2a.C                 hcons.C                 motorcycle.C        rootmarks.C         tree.C
cleanup.C               fit2.C                  hksimple.C              motorcycle.dat      rose512.jpg         triangles.C
clonesA_Event.C         fit2d.C                 hlabels1.C              multidimfit.C       rose512.png         tv3.C
clonesA_Event.cxx       fitcont.C               hlabels2.C              multifit.C          rose512.tiff        tvdemo.C
clonesA_Event_cxx.d     fitExclude.C            hprod.C                 multigraph.C        rose512.xpm         two.C
clonesA_Event.h         fithist.C               hserv2.C                MviaS.C             rose_image.C        twoscales.C
compile.C               fitslicesy.C            hserv.C                 myfit.C             runcatalog.sql      waves.C
copytree2.C             FittingDemo.C           hsimple.C               MyTasks.cxx         runzdemo.C          work.C
copytree3.C             FittingDemo_C.d         hsimple.root            na49.C              second.C            WorldMap.C
copytree.C              formula1.C              hstack.C                na49geomfile.C      seism.C             worldmap.xpm
core                    framework.C             hsumanim.C              na49.root           shapesAnim.C        xtruDraw.C
customContextMenu.C     galaxy_image.C          hsum.C                  na49view.C          shapes.C            xtruSamples.C
customTH1Fmenu.C        galaxy.pal.root         hsumTimer.C             na49visible.C       shared.C            xysliderAction.C
CVS                     galaxy.root             htest.C                 ntuple1.C           splines.C           xyslider.C
demos.C                 games.C                 htmldoc                 ntuple1_C.d         spy.C               zdemo.C
demoshelp.C             gaxis.C                 htmlex.C                oldbenchmarks.C     spyserv.C           zones.C
dirs.C                  gbasic.C                Ifit.C                  pad2png.C           sqlcreatedb.C
```

root/etc

Root default settings in etc/system.rootrc

These “public” settings can be redefined in

$HOME/.rootrc and .rootrc

root.exe or root libs

root.exe

```
root > gSystem->Load("libMyclasses")
root > Myclass a;
root > a.DoSomething(..);
```

```
#include "TRint.h"
int main(int argc, char **argv)
{
   TRint theApp("Rint", &argc, argv);

   //enter the event loop...
   theApp.Run();

   return 0;
}
```

myapp.exe

```
#include myclasses.h"
#include "TApplication.h"
int main(int argc, char **argv)
{
   TApplication theApp("MyApp", &argc, argv);
   MyClass a;
   a.DoSomething();
   //enter the event loop...
   theApp.Run();

   return 0;
}
```

Start/Quit

```
(pcbrun2) [224] root
  ****************************************
  *                                      *
  *        W E L C O M E  to  R O O T    *
  *                                      *
  *   Version   4.00/01   31 January 2004   *
  *                                      *
  *  You are welcome to visit our Web site  *
  *          http://root.cern.ch         *
  *                                      *
  ****************************************

Compiled for win32gdk.

C++ Interpreter version 5.15.117, Jan 4 2004
Type ? for help. Commands must be C++ statements.
Enclose multiple statements between { }.

WELCOME to the ROOT demos: Sat Jan 31 23:21:08 2004

root [0] .q
```

rootlogon.C

```
void rootlogon() {
  TDatime date;
  printf("\nWELCOME to the ROOT demos:%s\n\n",date.AsString());
}
```

My first session

root

```
root [0] 344+76.8
(const double)4.20800000000000010e+002
root [1] float x=89.7;
root [2] float y=567.8;
root [3] x+sqrt(y)
(double)1.13528550991510710e+002
root [4] float z = x+2*sqrt(y/6);
root [5] z
(float)1.09155929565429690e+002
root [6] .q
```

root

See file $HOME/.root_hist

root [0] try up and down arrows

My second session

root

```
root [0] .x session2.C
for N=100000, sum= 45908.6
root [1] sum
(double)4.59085828512453370e+004
Root [2] r.Rndm()
(Double_t)8.29029321670533560e-001
root [3] .q
```

unnamed macro
executes in global scope

session2.C

```
{
   int N = 100000;
   TRandom r;
   double sum = 0;
   for (int i=0;i<N;i++) {
      sum += sin(r.Rndm());
   }
   printf("for N=%d, sum= %g\n",N,sum);
}
```

My third session

root

```
root [0] .x session3.C
for N=100000, sum= 45908.6
root [1] sum
Error: Symbol sum is not defined in current scope
*** Interpreter error recovered ***
Root [2] .x session3.C(1000)
for N=1000, sum= 460.311
root [3] .q
```

session3.C

```
void session3 (int N=100000) {
   TRandom r;
   double sum = 0;
   for (int i=0;i<N;i++) {
      sum += sin(r.Rndm());
   }
   printf("for N=%d, sum= %g\n",N,sum);
}
```

Named macro
Normal C++ scope rules

My third session with ACLIC

```
root [0] gROOT->Time();
root [1] .x session4.C(10000000)
for N=10000000, sum= 4.59765e+006
Real time 0:00:06, CP time 6.890
root [2] .x session4.C+(10000000)

for N=10000000, sum= 4.59765e+006
Real time 0:00:09, CP time 1.062
root [3] session4(10000000)
for N=10000000, sum= 4.59765e+006
Real time 0:00:01, CP time 1.052
root [4] .q
```

File session4.C
Automatically compiled and linked by the native compiler.
Must be C++ compliant

session4.C

```
#include "TRandom.h"
void session4 (int N) {
   TRandom r;
   double sum = 0;
   for (int i=0;i<N;i++) {
      sum += sin(r.Rndm());
   }
   printf("for N=%d, sum= %g\n",N,sum);
}
```

Macros with more than one function

```
root [0] .x session5.C >session5.log
root [1] .q
```

```
root [0] .L session5.C
root [1] session5(100); >session5.log
root [2] session5b(3)
sum(0) = 0
sum(1) = 1
sum(2) = 3
root [3] .q
```

.x session5.C
executes the function session5 in session5.C

use gROOT->ProcessLine to execute a macro from a macro or from compiled code

session5.C

```
void session5(int N=100) {
   session5a(N);
   session5b(N);
   gROOT->ProcessLine(".x session4.C+(1000)");
}
void session5a(int N) {
   for (int i=0;i<N;i++) {
      printf("sqrt(%d) = %g\n",i,sqrt(i));
   }
}
void session5b(int N) {
   double sum = 0;
   for (int i=0;i<N;i++) {
      sum += i;
      printf("sum(%d) = %g\n",i,sum);
   }
}
```

Root prompt

- At the Root prompt, you can execute
  - a C/C++ statement
  - a CINT command (start with ".")

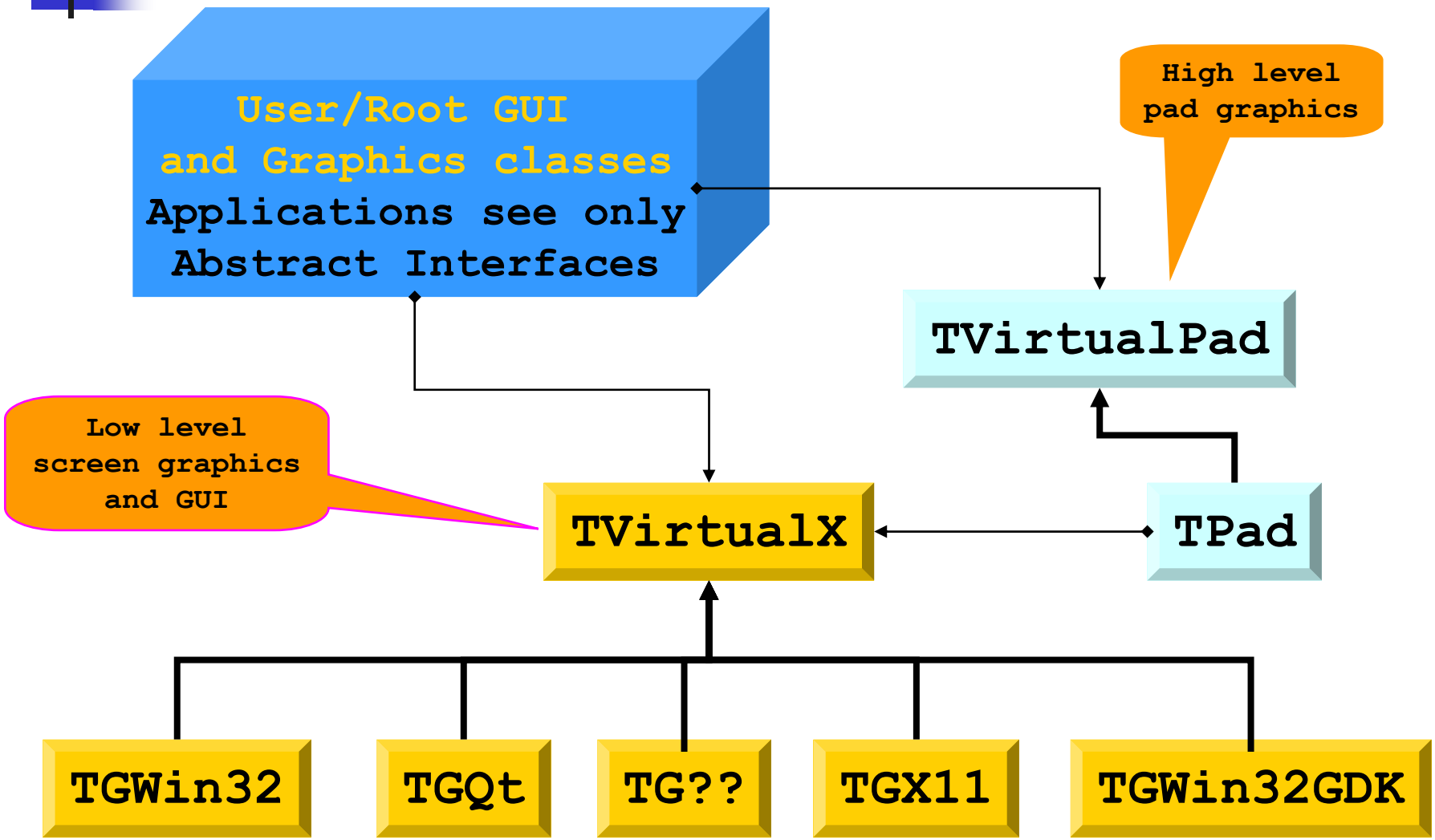
```
.? Show list of CINT commands
.q to quit the application
.x file.C
.L file.C
.T trace mode for macros
.!system_command, eg .!ls
```

Root in batch mode

- root –b –q session5.C
- root –b –q "session5.C(500)"
- root –b
  - root [0] .x session5.C
  - root [1] .q

Graphics

Gui/Graphics interfaces

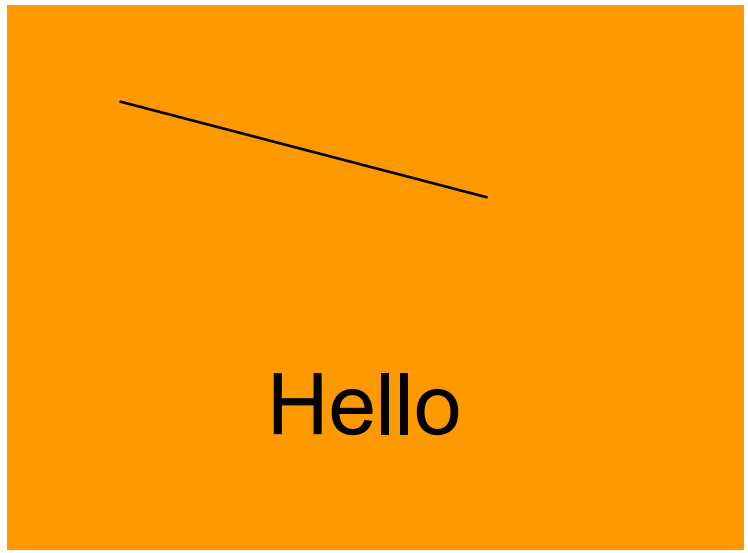


Coordinate Systems

- Pixel coordinates [0,32000] (TVirtualX)
  - (0,0) at top left corner
- NDC (Normalized Device Coordinates)
  - [0,1], (0,0) at bottom left corner
  - System used for annotations
- Normal User coordinates (TVirtualPad)
  - Linear/Log scale in x,y,z
  - Can switch between lin/log scale without calling Draw again.
- TVirtualPad/TPad have functions to convert between systems

TPad: main graphics container

```
Root > TLine line(.1,.9,.6,.6)
Root > line.Draw()
Root > TText text(.5,.2,"Hello")
Root > text.Draw()
```



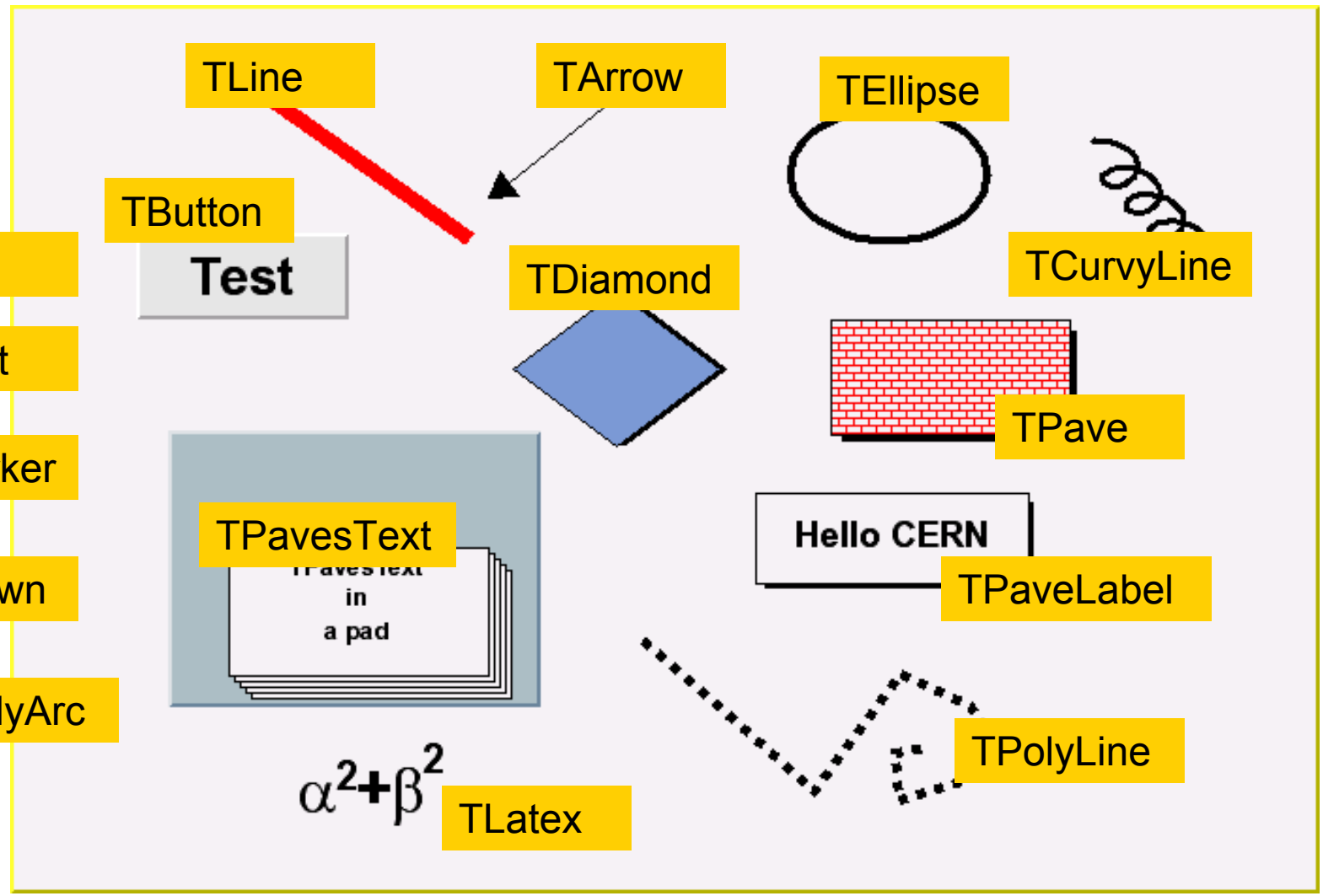
The Draw function adds the object to the list of primitives of the current pad.

If no pad exists, a pad is automatically created with a default range [0,1].

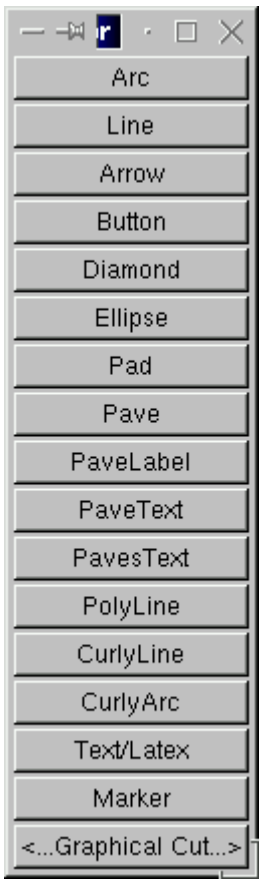
When the pad needs to be drawn or redrawn, the object Paint function is called.

Only objects deriving from TObject may be drawn in a pad
Root Objects or User objects

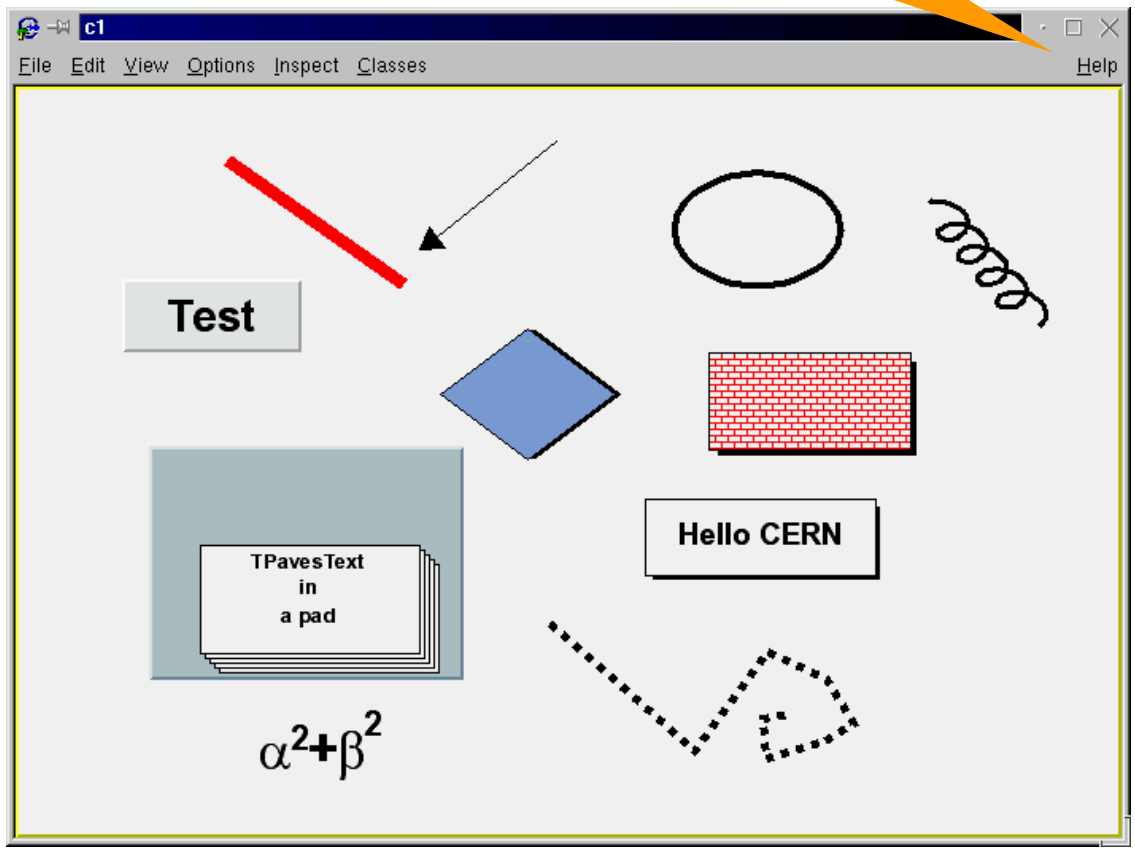
Basic Primitives



The Graphics Editor



See HELP button
to use the graphics editor



Saving a pad/canvas

- A pad/canvas may be saved in many formats using the GUI menu or via TPad::SaveAs
  - canvas.C : a C++ script is automatically generated. The canvas may be generated again via .x canvas.C
  - canvas.ps(eps) Postscript or encapsulated ps
  - canvas.svg : scalable vector graphics
  - canvas.gif
  - canvas.root: keep objects in a compressed and portable format.

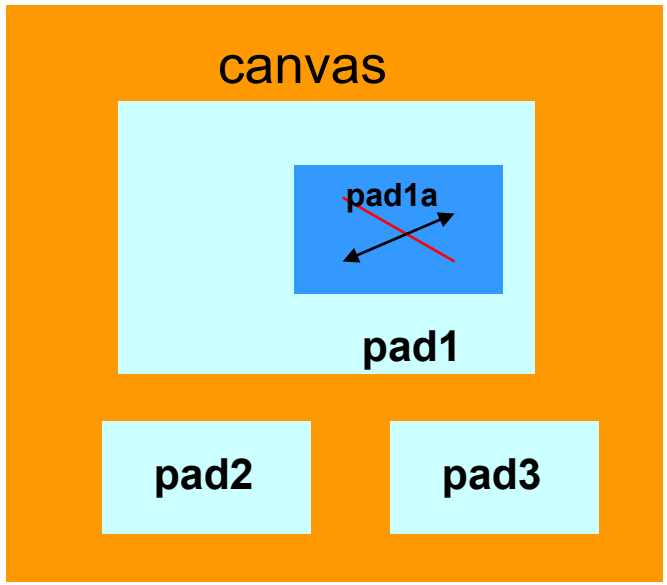
TPad hierarchy

A primitive in the pad can be another pad.

The top level pad is a TCanvas.

A pad has an associate pixmap.

Painting is in the pixmap. When the pad must be redrawn, the pixmap is flushed to the window



The pixmap is flushed when the pad is declared "modified" and TPad::Update called

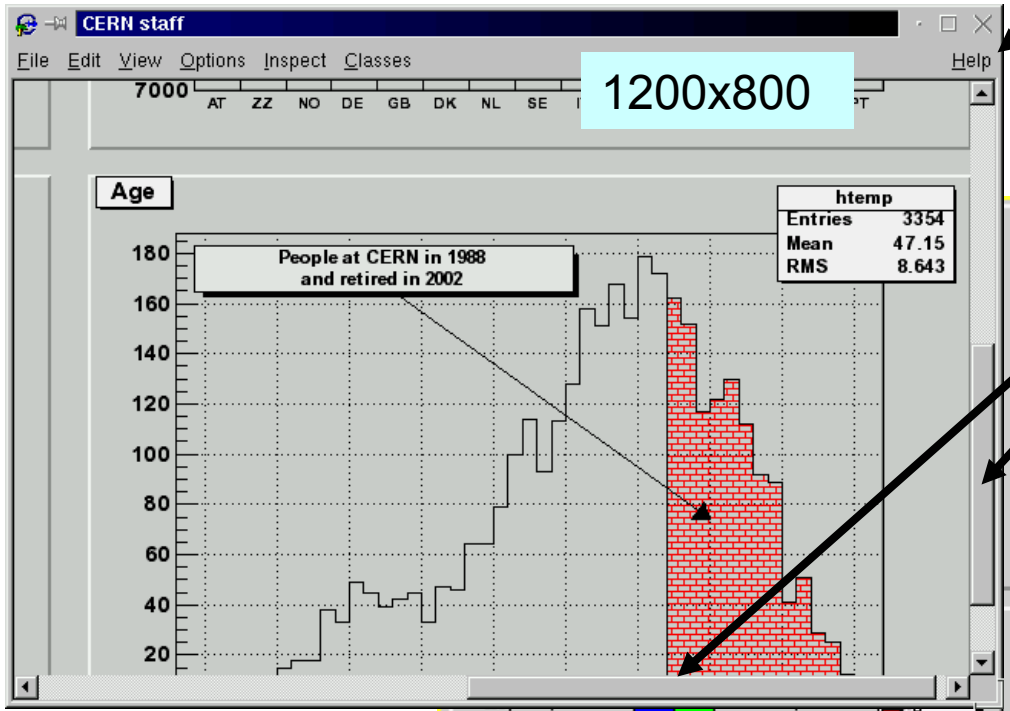
Canvas pixmaps

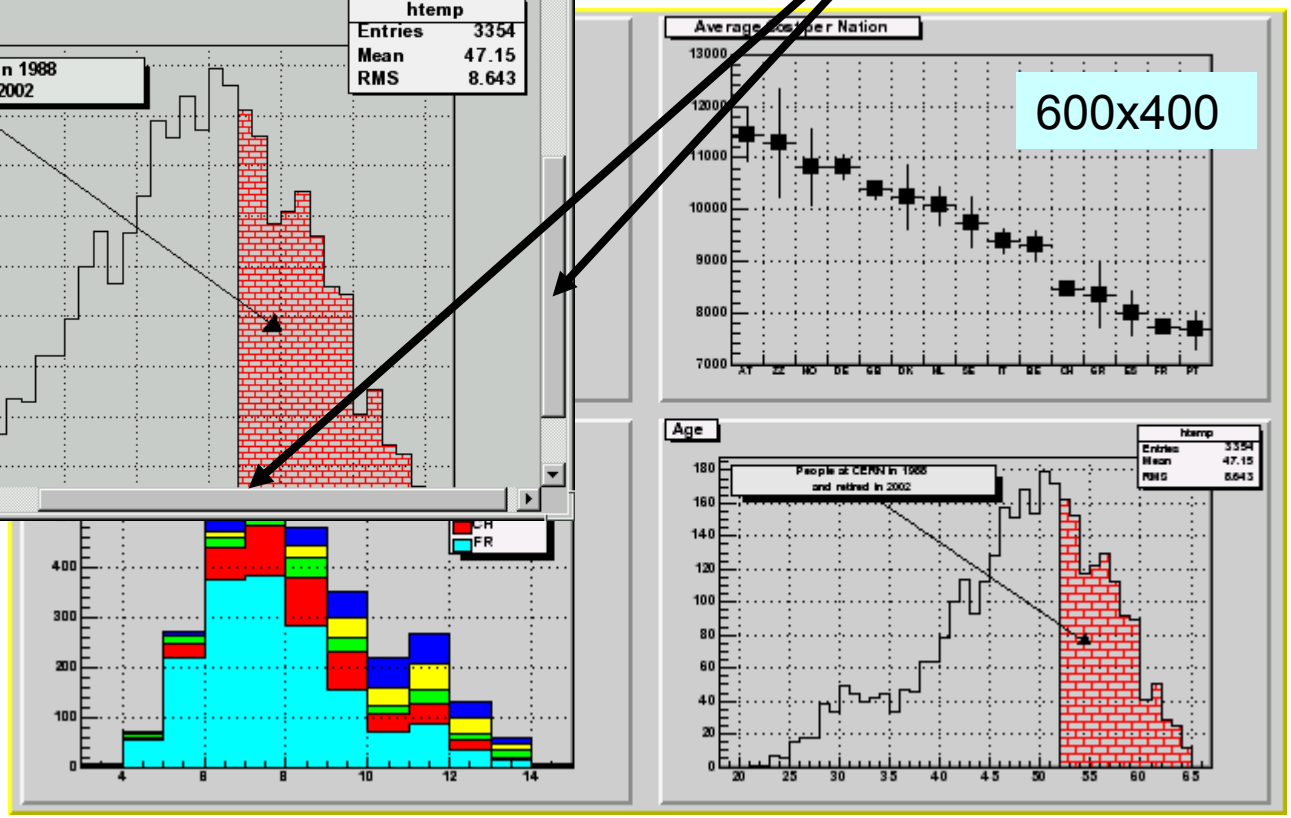
Need Help?

Scroll bars

1200x800

600x400





Graphics in Batch mode

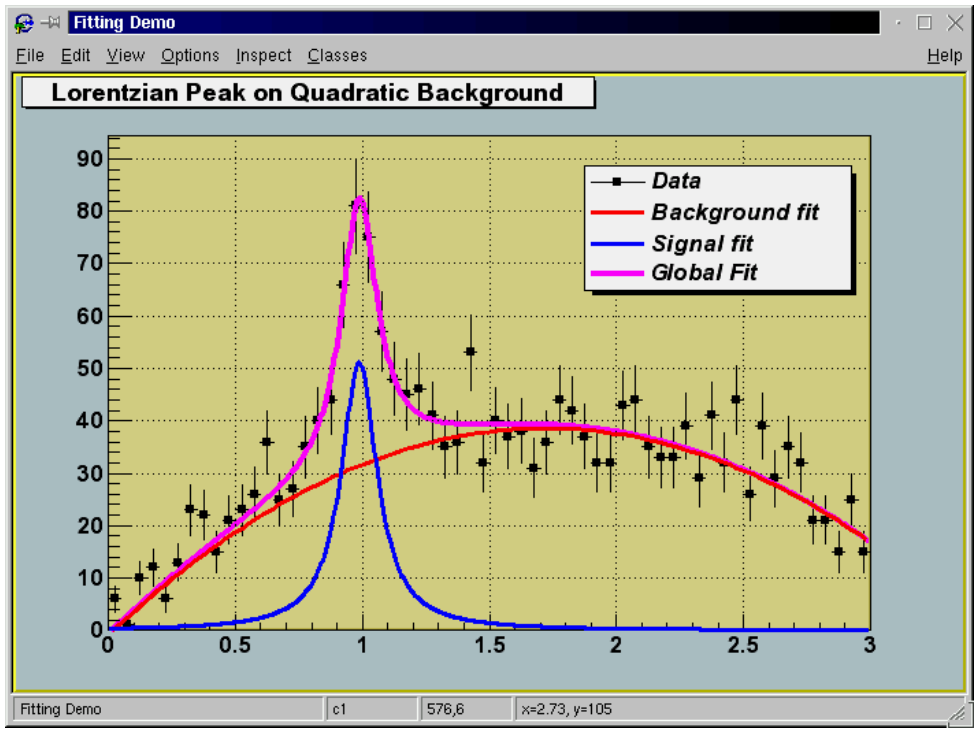
- No changes required to your program
- X11 libs are not loaded
- Can produce output with .ps,.eps,.svg,.root
- example:
  - canvas.SavesAs("file.ps")

Picking- Graphics event loop

TCanvas::HandleInput manages the event loop. When the mouse moves to a pad, TPad::Pick is called.

TPad::Pick loops on all primitives in the pad, calling their function DistancetoPrimitive. The selected object is the one with the smallest distance.

Activate the tool bar option "ShowEventStatus" to understand the behaviour.

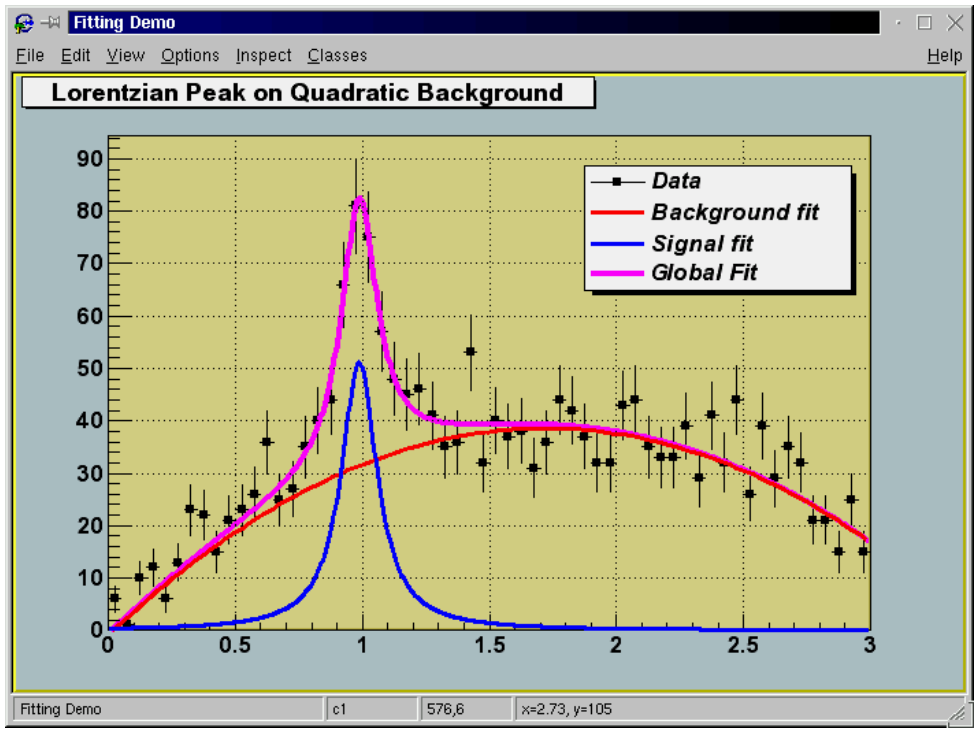


Picking- Graphics event loop

When an object has been selected by DistancetoPrimitive, its ExecuteEvent function is called.

The result of ExecuteEvent is object dependent. The cursor changes shape to suggest possible actions : grow/shrink/move box, rotation, etc.

Pressing the right mouse button on an object shows its context menu.

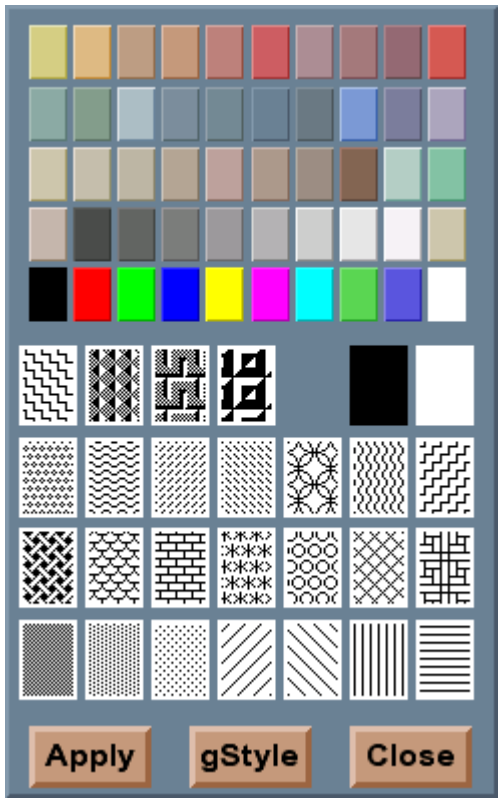


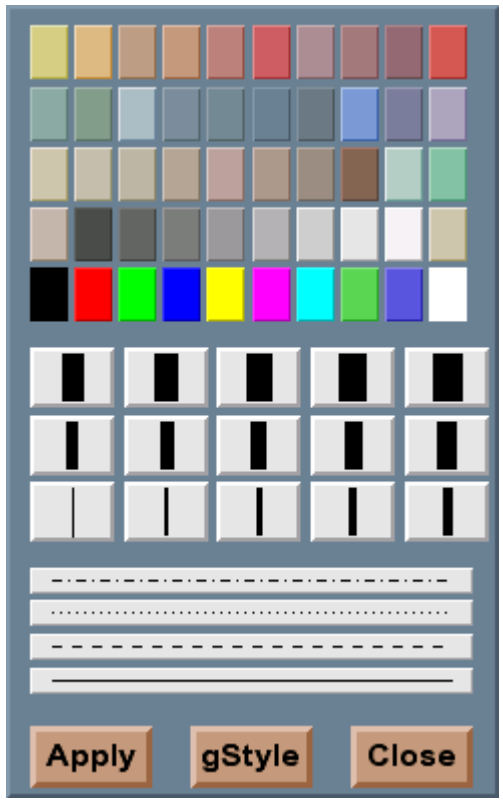
The Graphics Event Loop

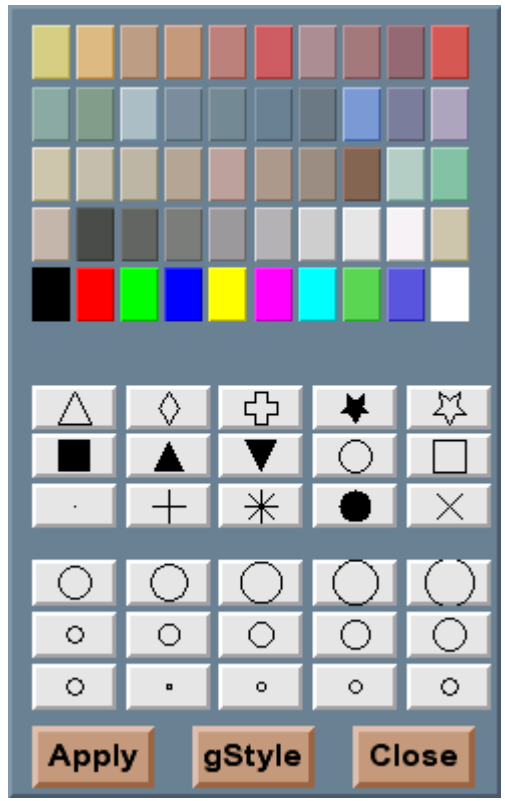
- The ROOT event handler supports:
  - keyboard interrupts
  - system signals
  - X11, Xt, Xm events
  - Sockets interrupts
  - Special messages (shared memory, threads..)
- Foreign systems (eg Inventor, X3d..) can easily be integrated in the ROOT loop.
  - ROOT will dispatch the foreign events using Timers.
- Signals and Slots like in Qt
  - Qt and ROOT can work together

Graphics attributes

Most Root classes derive from very light weight data objects
classes like TAttLine, TAttFill, TAttMarker, TAttText

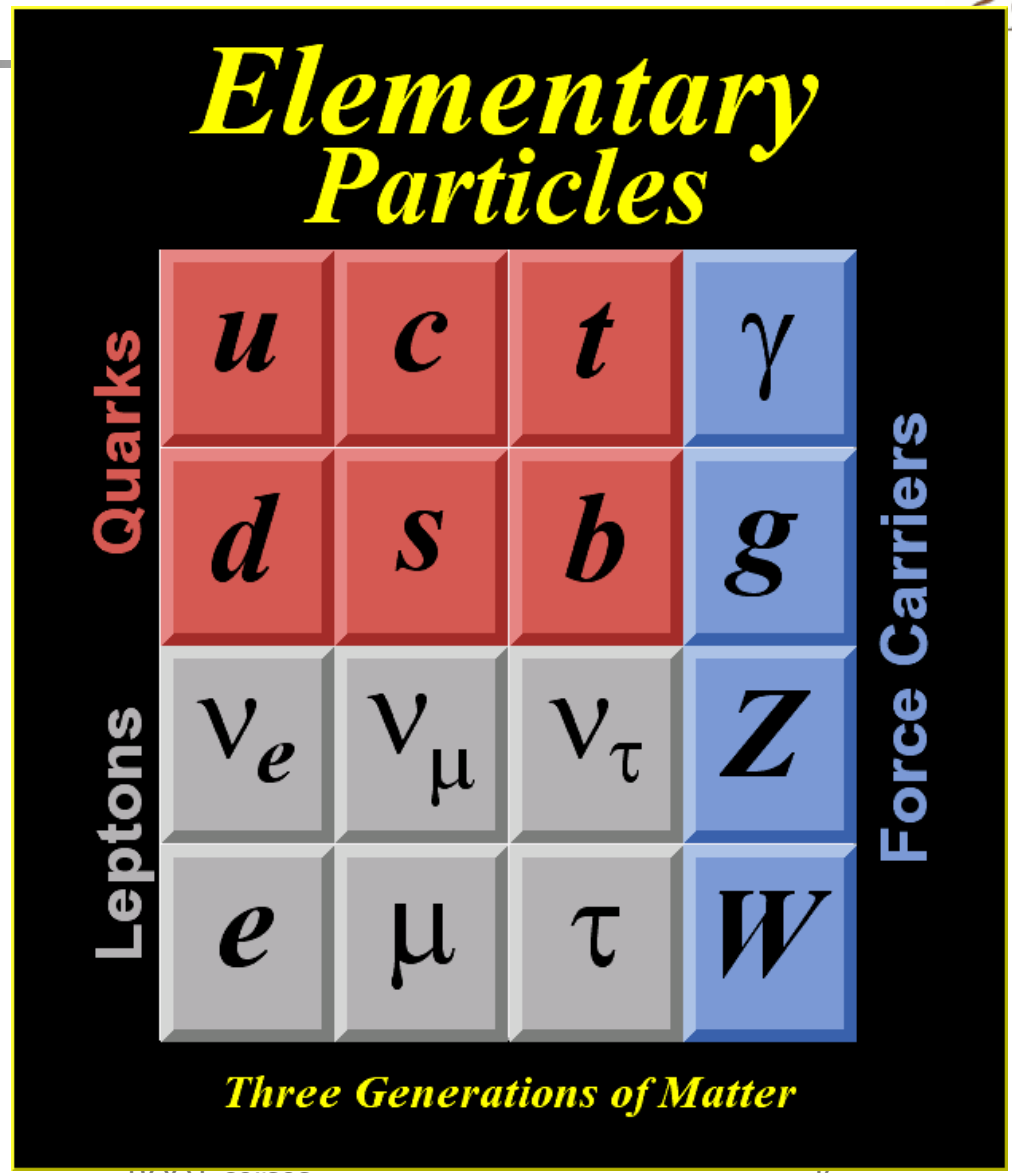






Annotations, Paves

quarks.C



Full LateX support on screen and postscript

latex3.C

Born equation

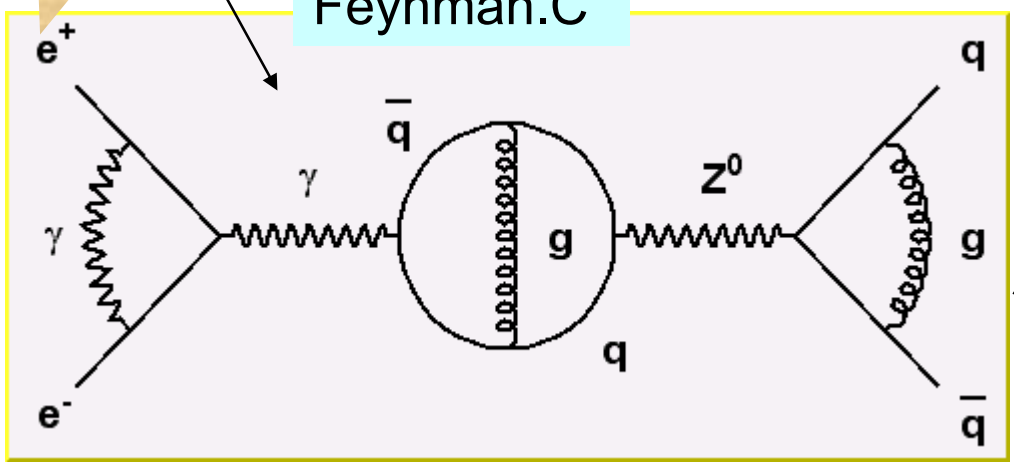
$$\frac{2s}{\pi\alpha^2}\frac{d\sigma}{d\cos\theta}(e^+e^- \to f\bar{f}) = \left|\frac{1}{1-\Delta\alpha}\right|^2 (1+\cos^2\theta)$$

$$+ 4\,\mathrm{Re}\left\{\frac{2}{1-\Delta\alpha}\chi(s)\left[\tilde{g}_v^e\tilde{g}_v^f(1+\cos^2\theta) + 2\,\tilde{g}_a^e\tilde{g}_a^f\cos\theta\right]\right\}$$

$$+ 16\left|\chi(s)\right|^2\left[(\tilde{g}_a^{e\,2} + \tilde{g}_v^{e\,2})(\tilde{g}_a^{f\,2} + \tilde{g}_v^{f\,2})(1+\cos^2\theta) + 8\,\tilde{g}_a^e\tilde{g}_a^f\tilde{g}_v^e\tilde{g}_v^f\cos\theta\right]$$

Formula or diagrams can be edited with the mouse

Feynman.C



TCurlyArc
TCurlyLine
TWavyLine
and other building blocks for Feynmann diagrams

Graphs

TGraphErrors(n,x,y,ex,ey)

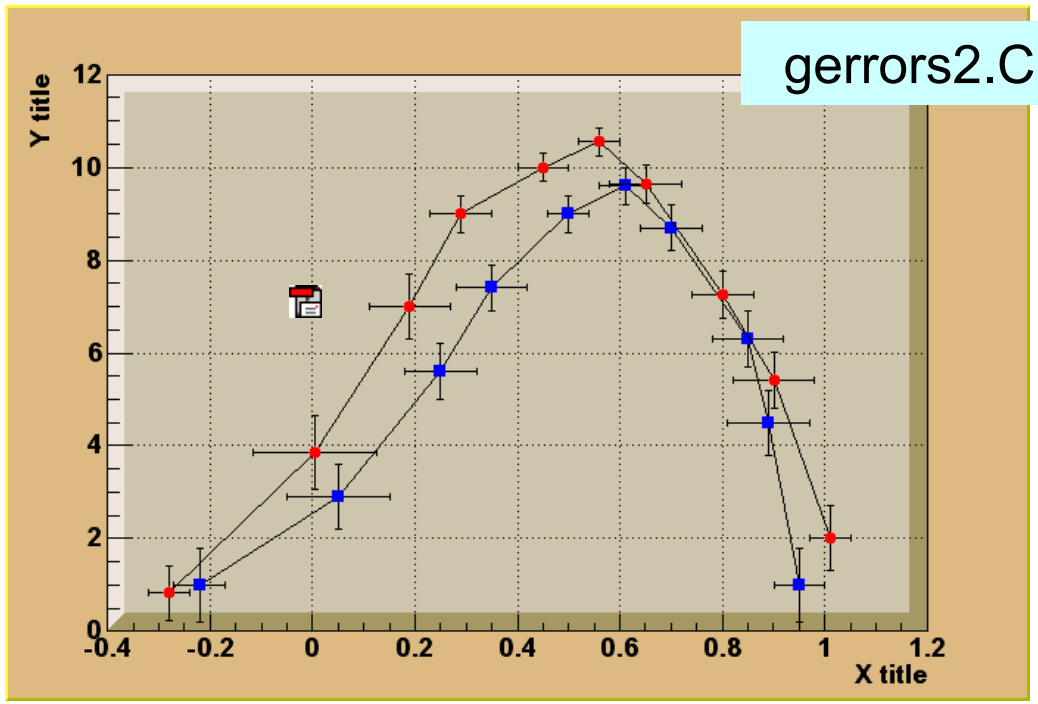
TGraphAsymmErrors(n,x,y,exl,exh,eyl,eyh)

TGraph(n,x,y)

TCutG(n,x,y)

TMultiGraph

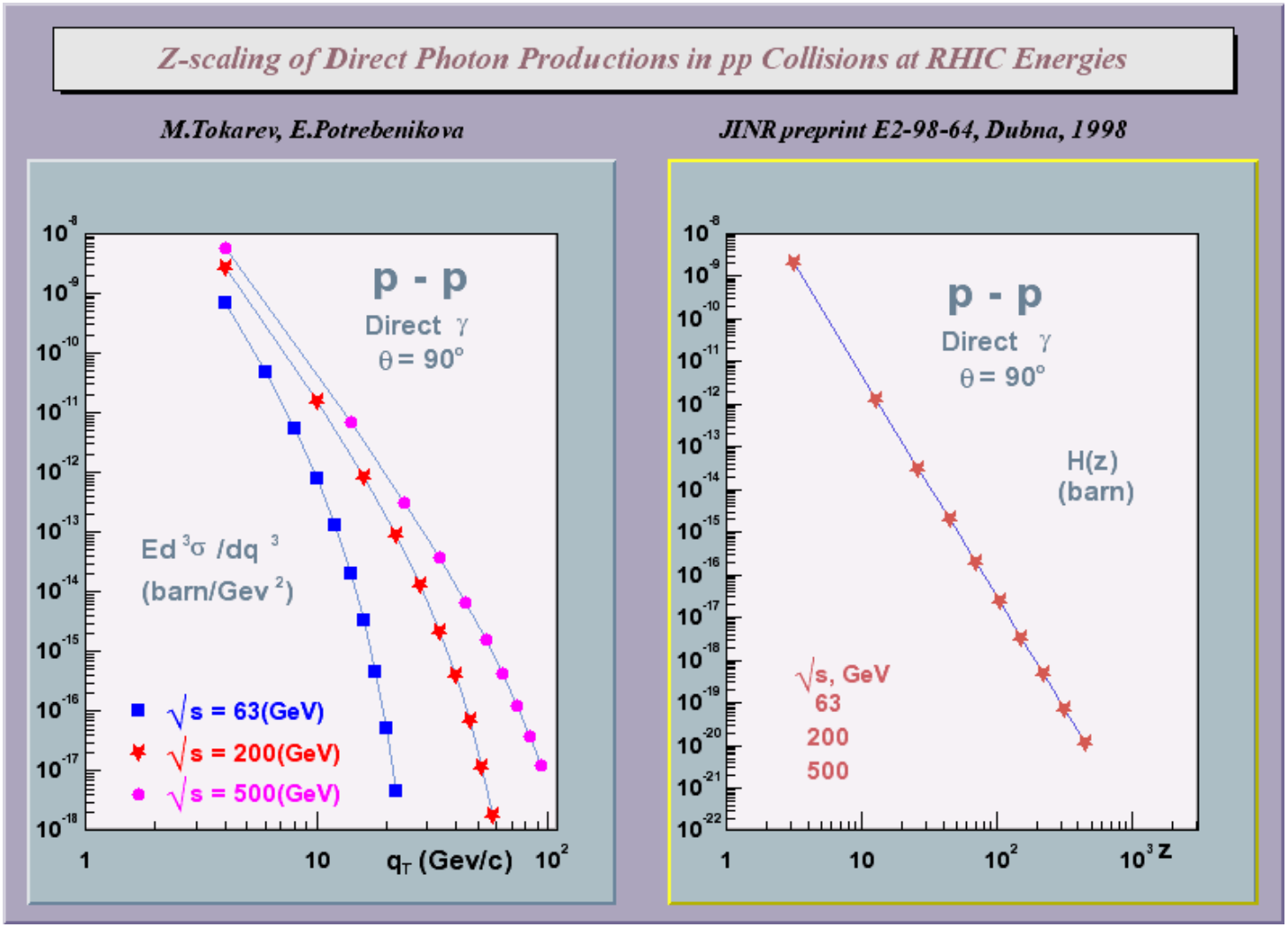
gerrors2.C



TGraphBentErrors(n,x,y,exl,exh,eyl,eyh,exld,exhd,eyld,eyhd)

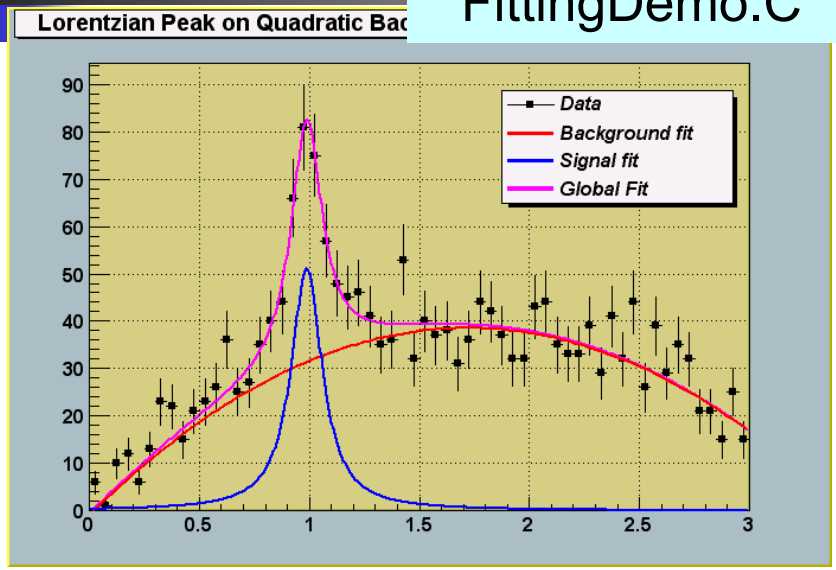
Graphs

zdemo.C

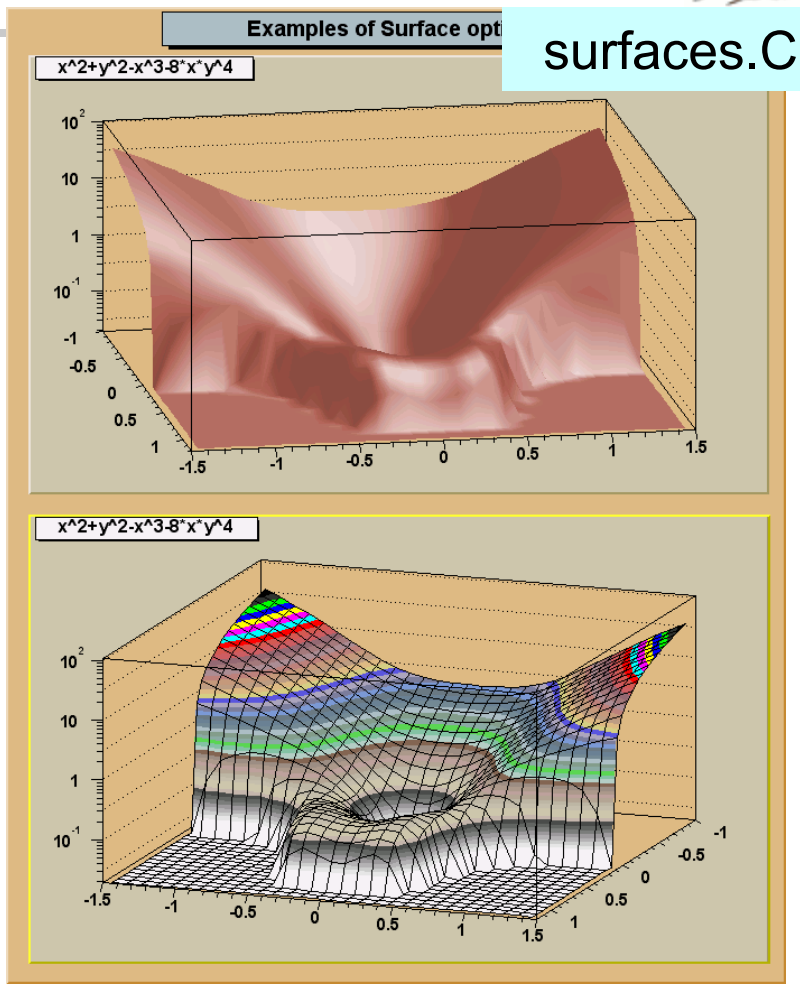


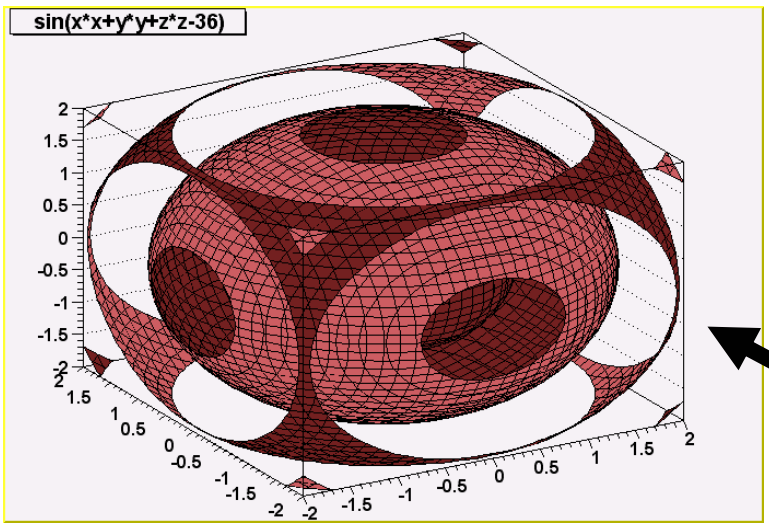
Graphics : 1,2,3-D functions

FittingDemo.C



surfaces.C



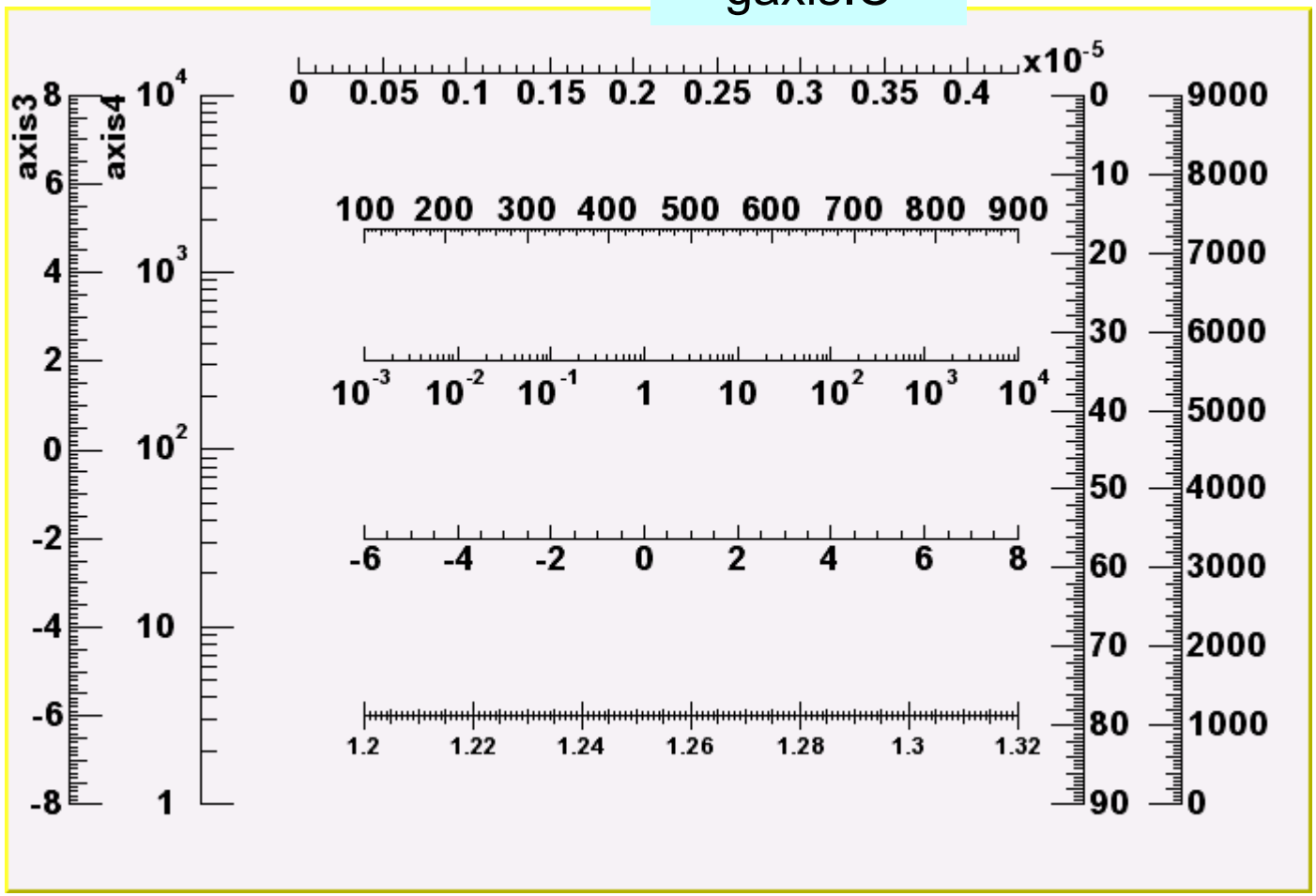


TF3 f3("f3","sin(x*x+y*y+z*z-36)",-2,2,-2,2,-2,2);

f3->Draw();

TGaxis

gaxis.C

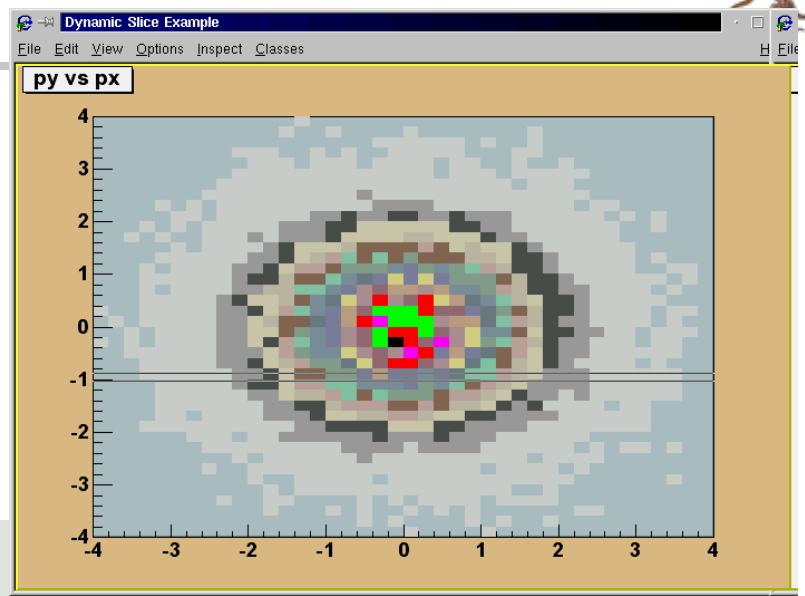


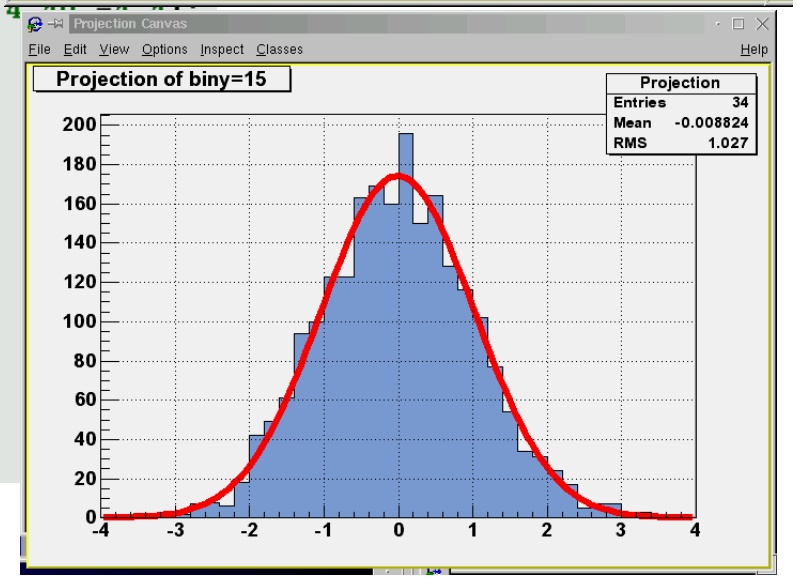
Adding dynamic objects to a pad

A TExec object may be added to the pad. Its Paint function can call any CINT command.

```
{
   //create a 2-d histogram, fill and draw it
   TH2F *hpxpy   = new TH2F("hpxpy","py vs px",40,-4,4
   Double_t px,py;
   for (Int_t i = 0; i < 50000; i++) {
      gRandom->Rannor(px,py);
      hpxpy->Fill(px,py);
   }
   hpxpy->Draw("col");

   //Add a TExec object to the canvas
   c1->AddExec("dynamic","DynamicExec()");
}

void DynamicExec()
{
 ....
}
```

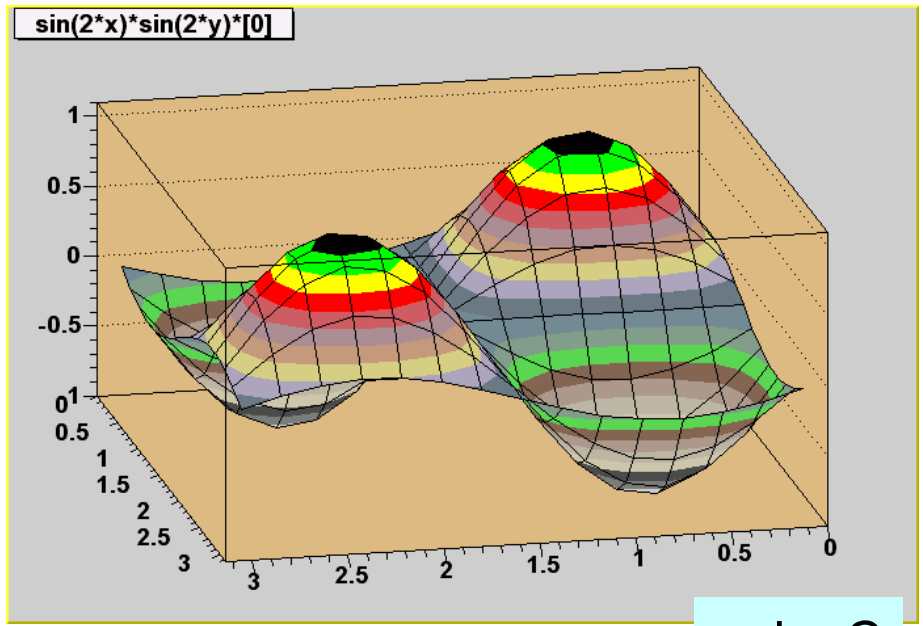
DynamicSlice.C





Using Timers with graphics

```
Double_t pi = TMath::Pi();
TF2 *f2;
Float_t t = 0;
Float_t phi = 30;
void anim() {
   gStyle->SetFrameFillColor(42);
   TCanvas *c1 = new TCanvas("c1");
   c1->SetFillColor(17);
 f2 = new
TF2("f2","sin(2*x)*sin(2*y)*[0]",0,pi,0,pi);
   f2->SetParameter(0,1);
   f2->SetNpx(15);
   f2->SetNpy(15);
   f2->SetMaximum(1);
   f2->SetMinimum(-1);
   f2->Draw("surf1");
   TTimer *timer = new TTimer(20);
   timer->SetCommand("Animate()");
   timer->TurnOn();
}
void Animate() {
   t += 0.05*pi;
   f2->SetParameter(0,TMath::Cos(t));
   phi += 2;
   gPad->SetPhi(phi);
   gPad->Modified();
   gPad->Update();
}
```



anim.C

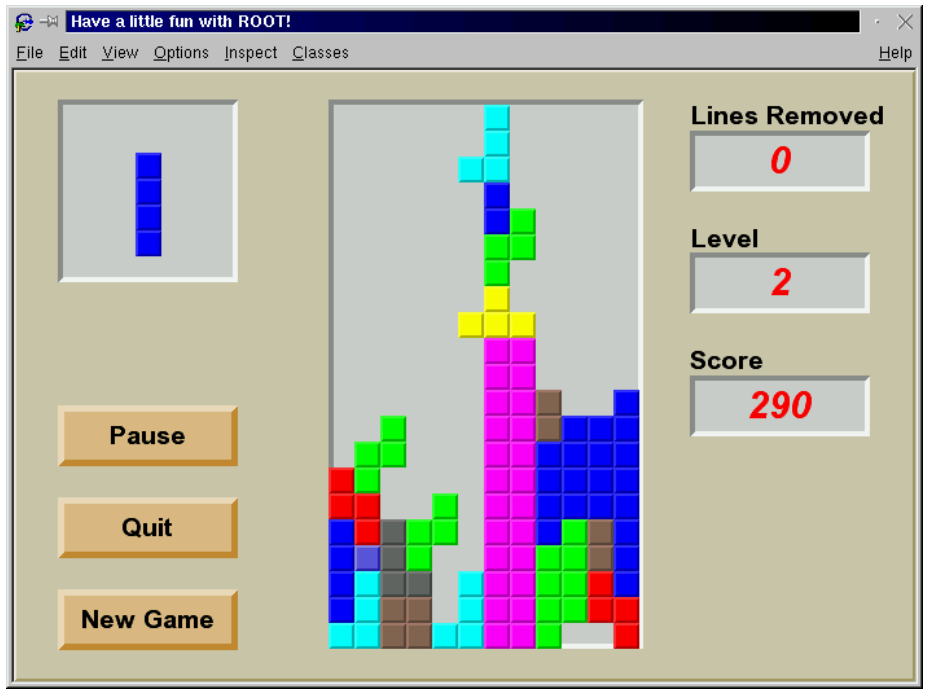
Every 20 milliseconds:

-Modify the function parameter

-Change the view angle

More on Timers with Graphics

```
{
 // run the dancing Hello World
   Hello hello;

   // run the analog clock
   Aclock clock;

   // run the Tetris game
   Tetris tetris;
}
```

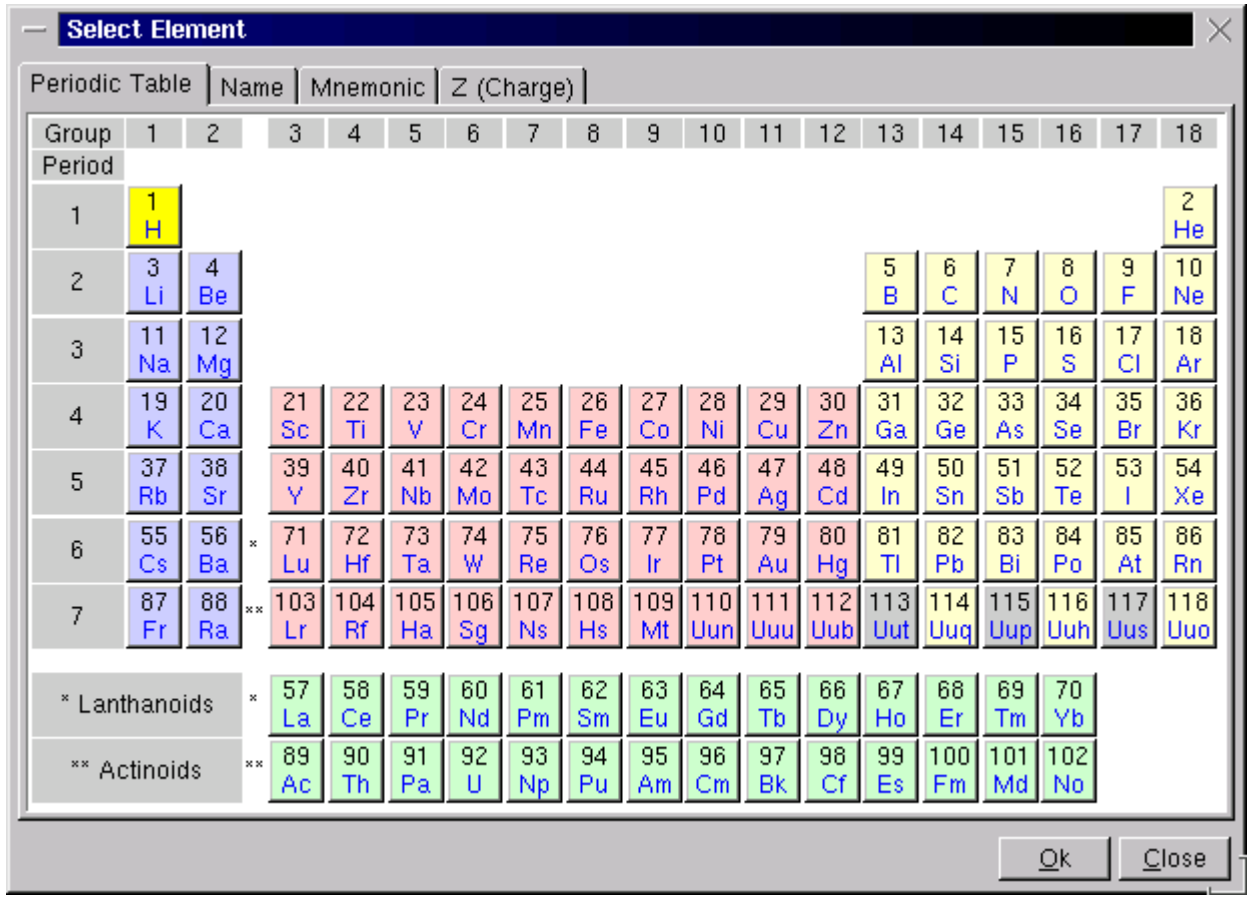


The 3 canvases run in parallel

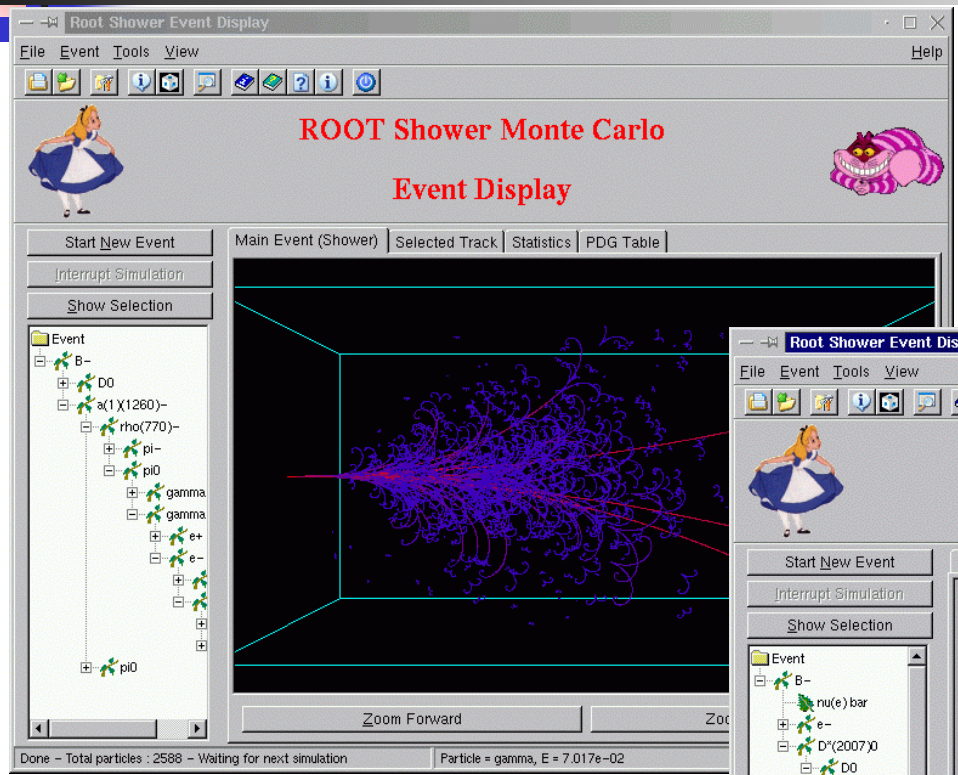
games.C

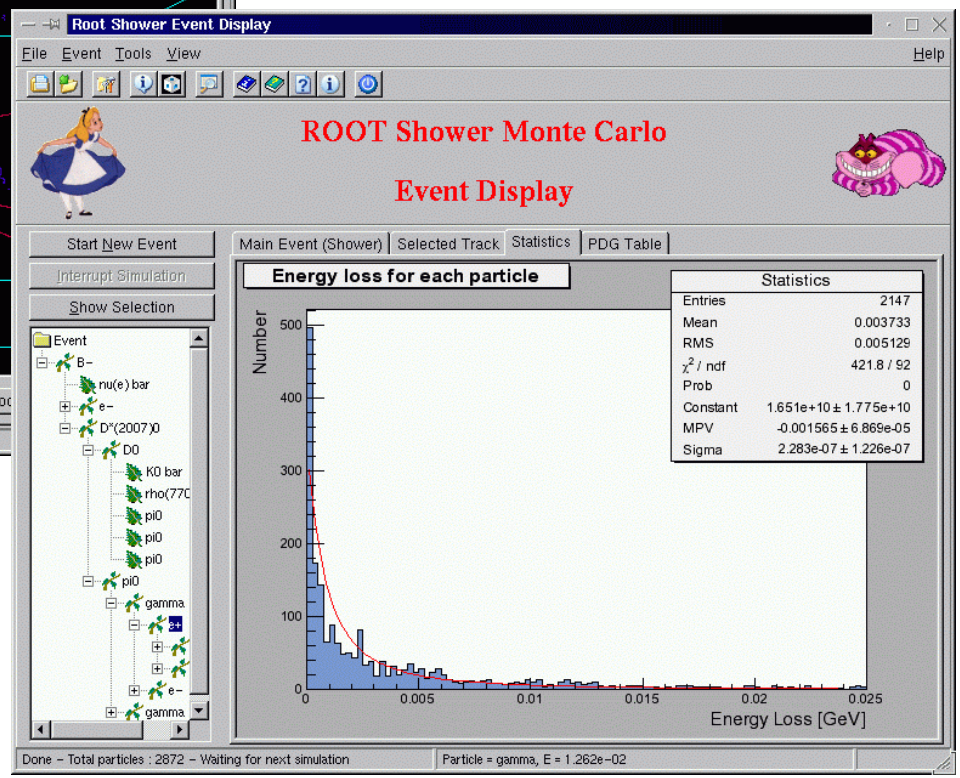
GUI User example

Example of GUI based on ROOT tools
Each element is clickable



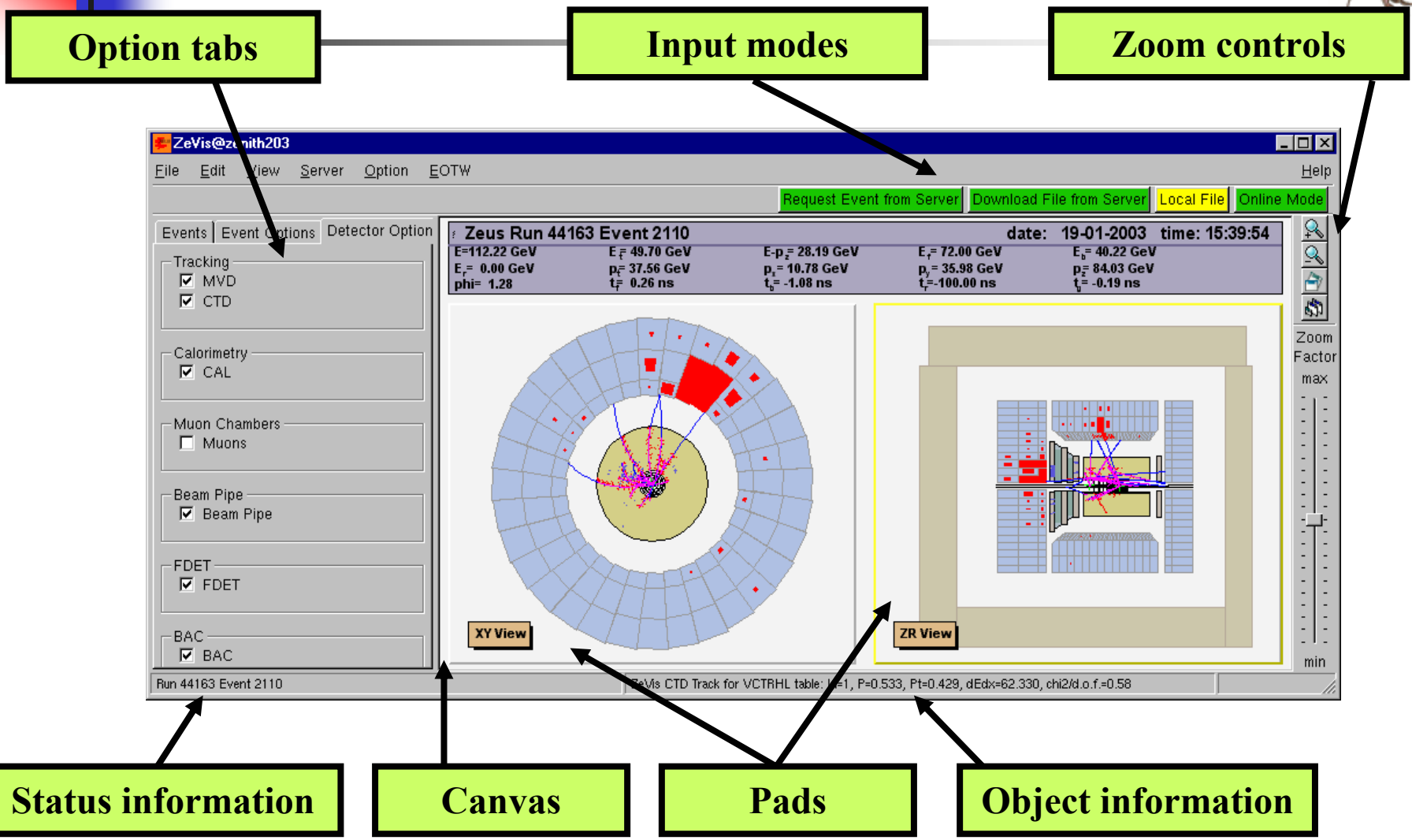
GUI Examples



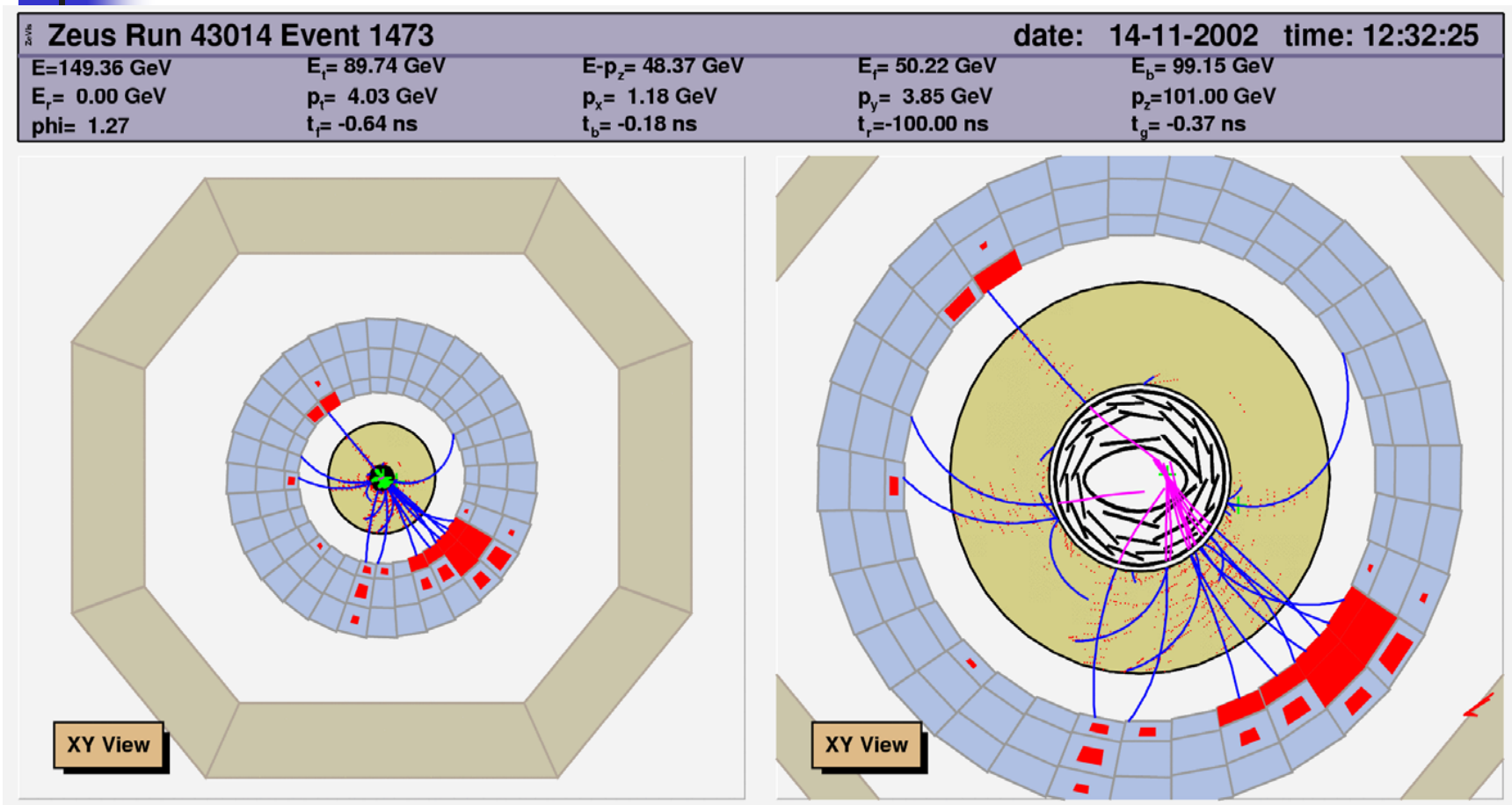


Graphical User Interface

Ex: ZEUS event display



Fish-eye View



TASImage classes

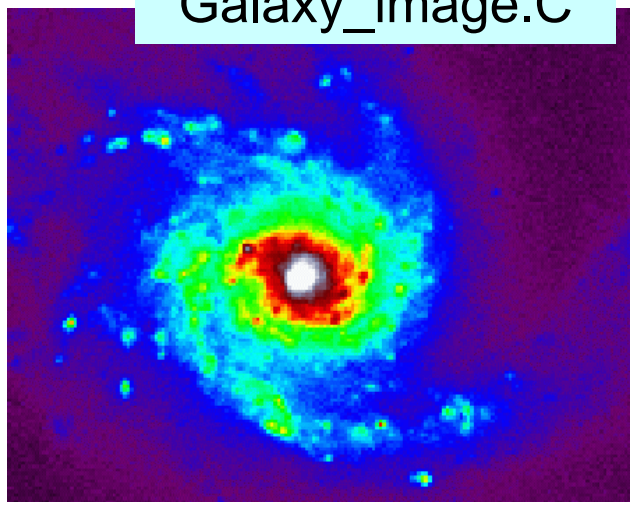
New set of image processing classes. The TImage class is the abstract image base class and TASImage is the concrete implementation using the libAfterImage imaging library of Sasha Vasko <sasha@aftercode.net>.

A large part of the development was done by Reiner Rohlfs from the ISDC based on a set of astrophysics user requirements.

The image class allows for the reading and writing of images in different formats, several image manipulations (scaling, tiling, merging, etc.) and displaying in pads. The size of the image on the screen does not depend on the original size of the image but on the size of the pad.Therefore it is very easy to resize the image on the screen by resizing the pad.

Rose_image.C

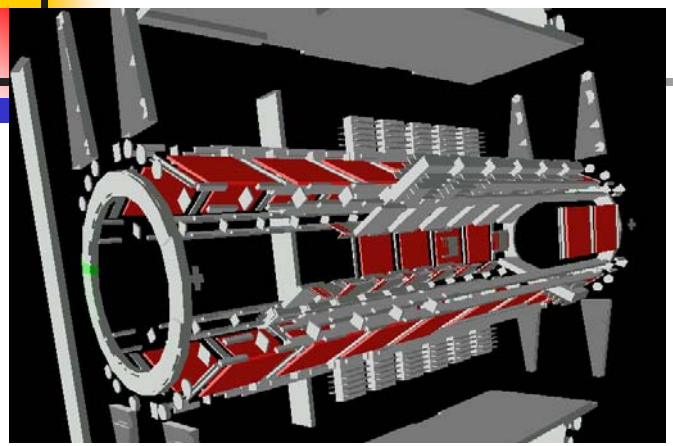
Galaxy_image.C

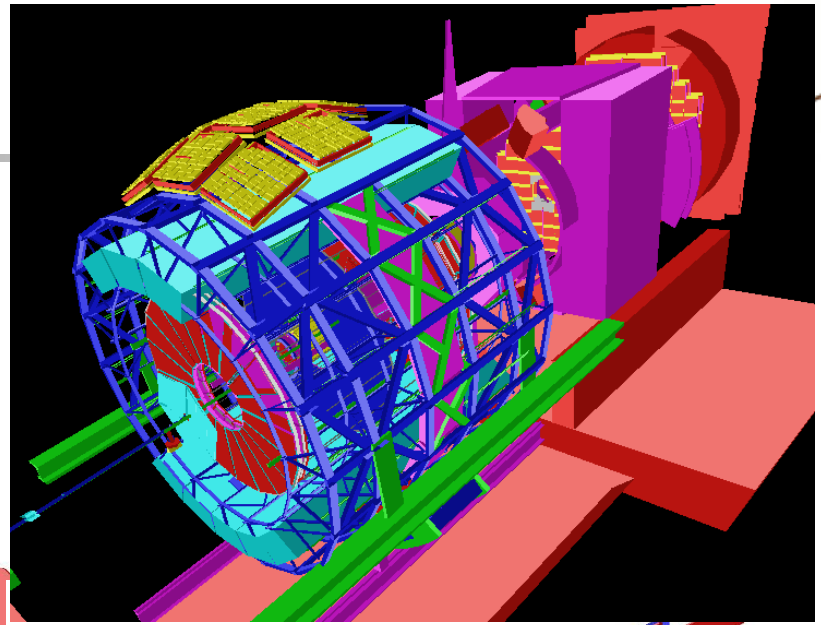


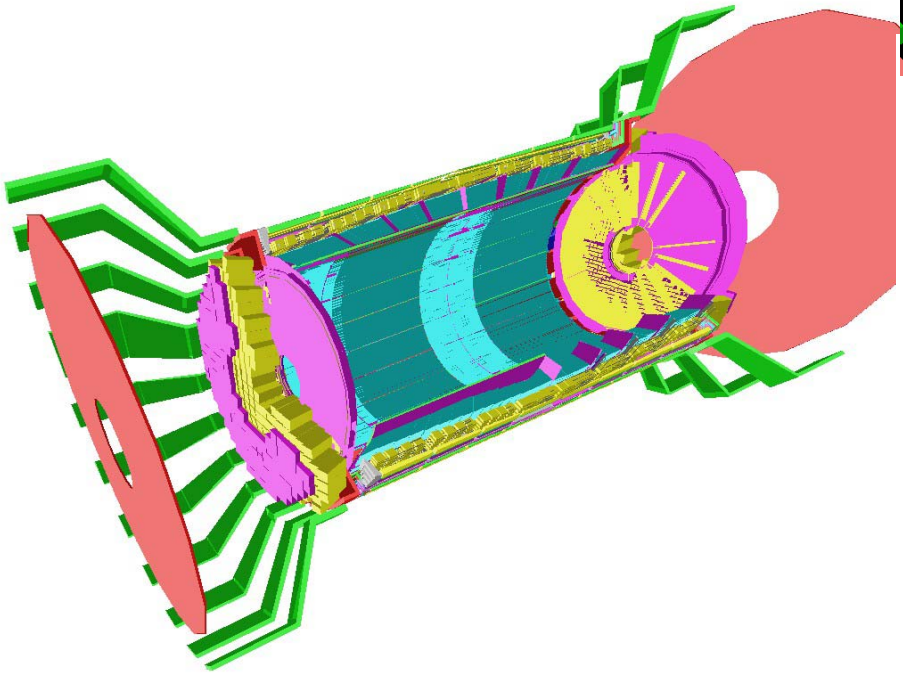
3-D Graphics

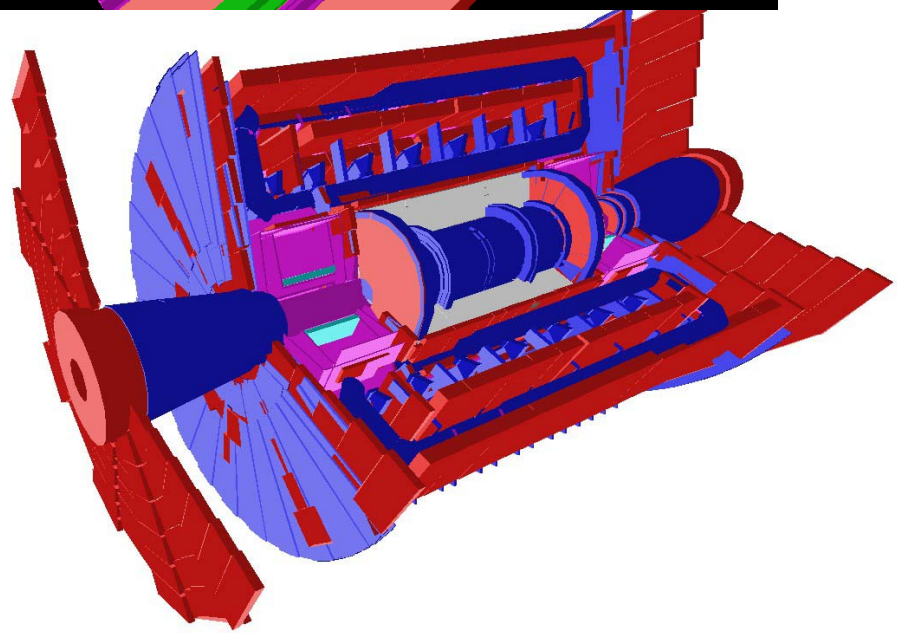
- Basic primitives
  - TPolyLine3D, TPolyMarker3D, THelix, TMarker3DBox,TAxis3D
- Geant primitives
  - Support for all Geant3 volumes + a few new volume types
  - TBRIK,TCONE,TCONS,TCTUB,TELTU,TGTRA,THYPE,TPARA,TPCON, TPGON,TSPHE,TTUBE,TTUBS,TTRAP,TTRD1,TTRD2,TXTRU
- Geometry package
- Rendering with:
  - TPad
  - X3D (very fast. Unix only. Good on networks)
  - OpenGL
  - OpenInventor (new addition in 3.01)

Some detectors in ROOT geometry



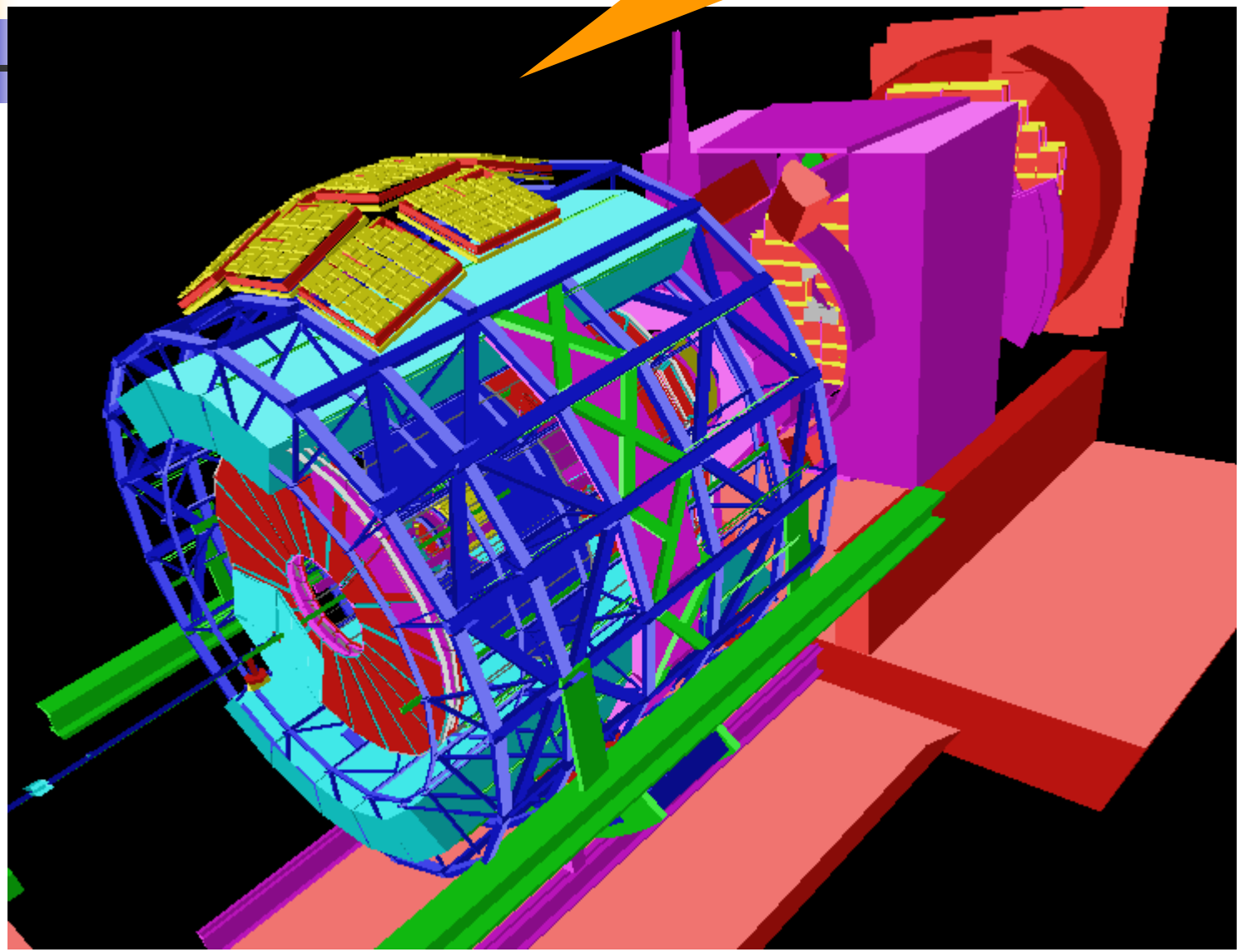


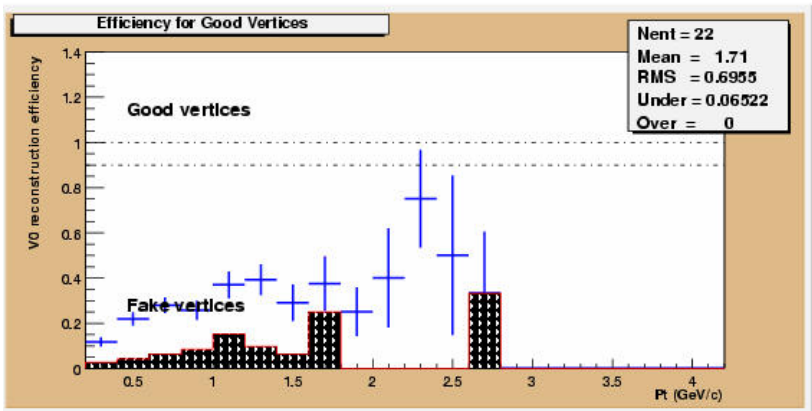


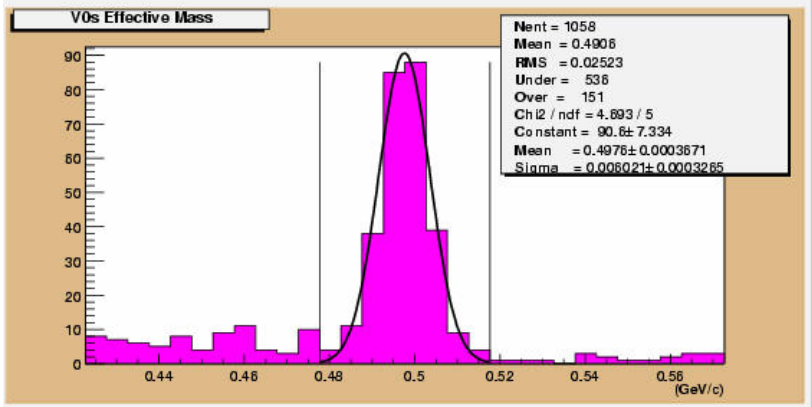


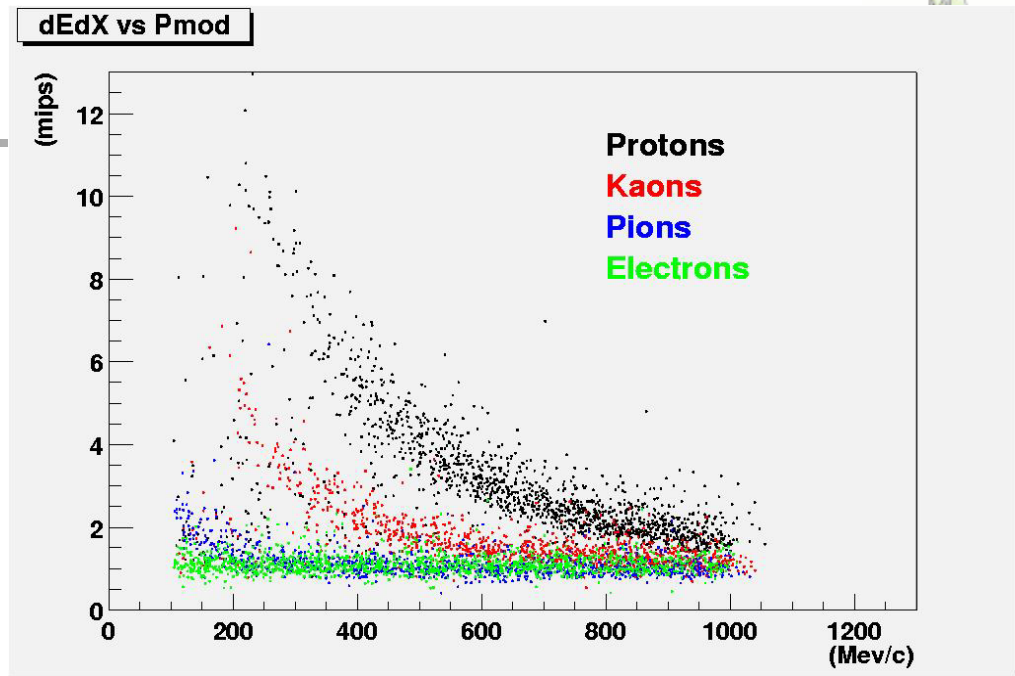
Alice

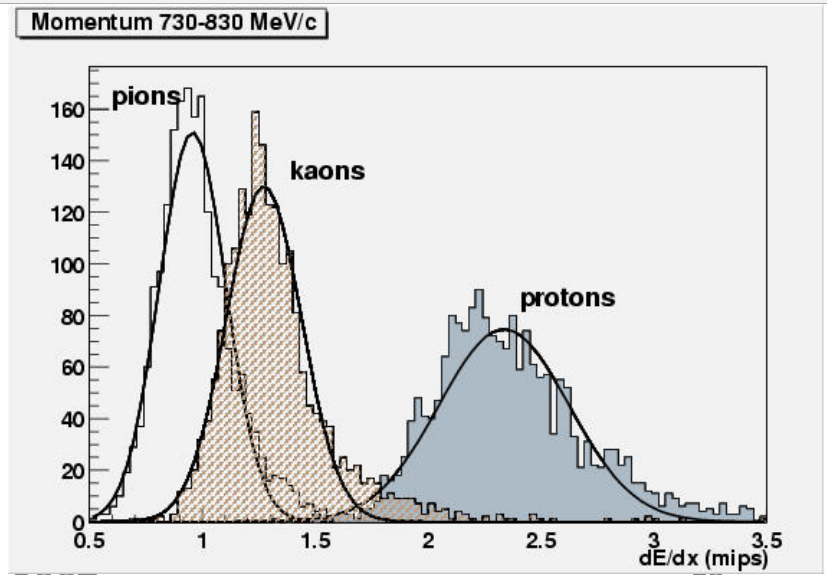
3 million nodes




Efficiency for Good Vertices
Nent = 22
Mean = 1.71
RMS = 0.6955
Under = 0.06522
Over = 0
Good vertices
Fake vertices
V0 reconstruction efficiency
Pt (GeV/c)

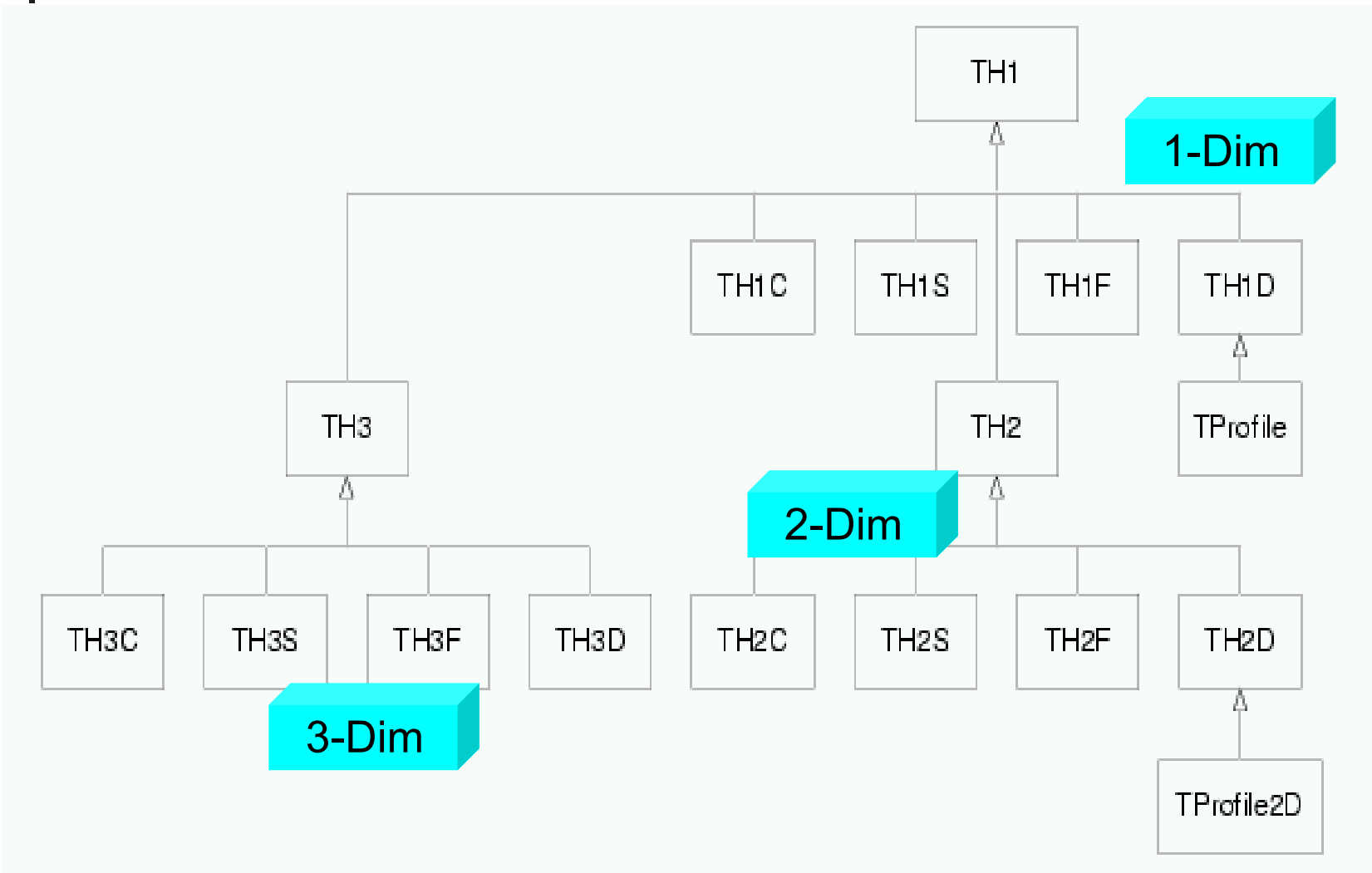

V0s Effective Mass
Nent = 1058
Mean = 0.4906
RMS = 0.02523
Under = 536
Over = 151
Chi2 / ndf = 4.693 / 5
Constant = 90.6± 7.334
Mean = 0.4976± 0.0003671
Sigma = 0.006021± 0.0003265
(GeV/c)


dEdX vs Pmod
(mips)
Protons
Kaons
Pions
Electrons
(Mev/c)


Momentum 730-830 MeV/c
pions
kaons
protons
dE/dx (mips)

The Histogram Package

The Histogram Classes



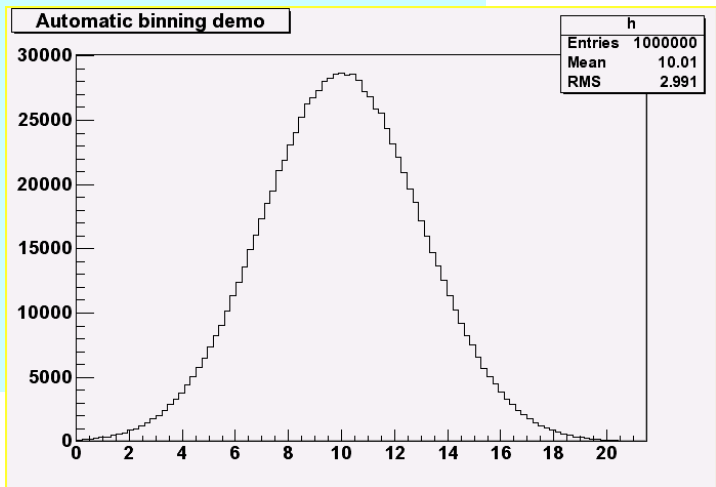
Filling Histograms

- An histogram is typically filled with statements like:
  - ***h1->Fill(x);***
  - ***h1->Fill(x,w); //fill with weight***
  - ***h2->Fill(x,y)***
  - ***h2->Fill(x,y,w)***
  - ***h3->Fill(x,y,z)***
  - ***h3->Fill(x,y,z,w)***
  - ***h1->Fill(string)***
- The Fill functions return the bin number for 1-D histograms or global bin number for 2-D and 3-D histograms.
- If TH1::Sumw2 has been called before filling, the sum of squares of weights is also stored.
- One can also increment directly a bin number via TH1::AddBinContent or replace the existing content via TH1::SetBinContent.
- To access the bin content of a given bin, do:
  Double_t binContent = h->GetBinContent(bin);

Automatic binning

```
#include "TH1.h"
#include "TF1.h"

void demoauto() {
   TF1 *f1 = new TF1("f1","gaus",0,30);
   f1->SetParameters(1,10,3);
   TH1F *h = new TH1F("h","Automatic binning demo",100,0,0);
   for (Int_t i=0;i<1000000;i++) {
      h->Fill(f1->GetRandom());
   }
   h->Draw();
}
```

No limits



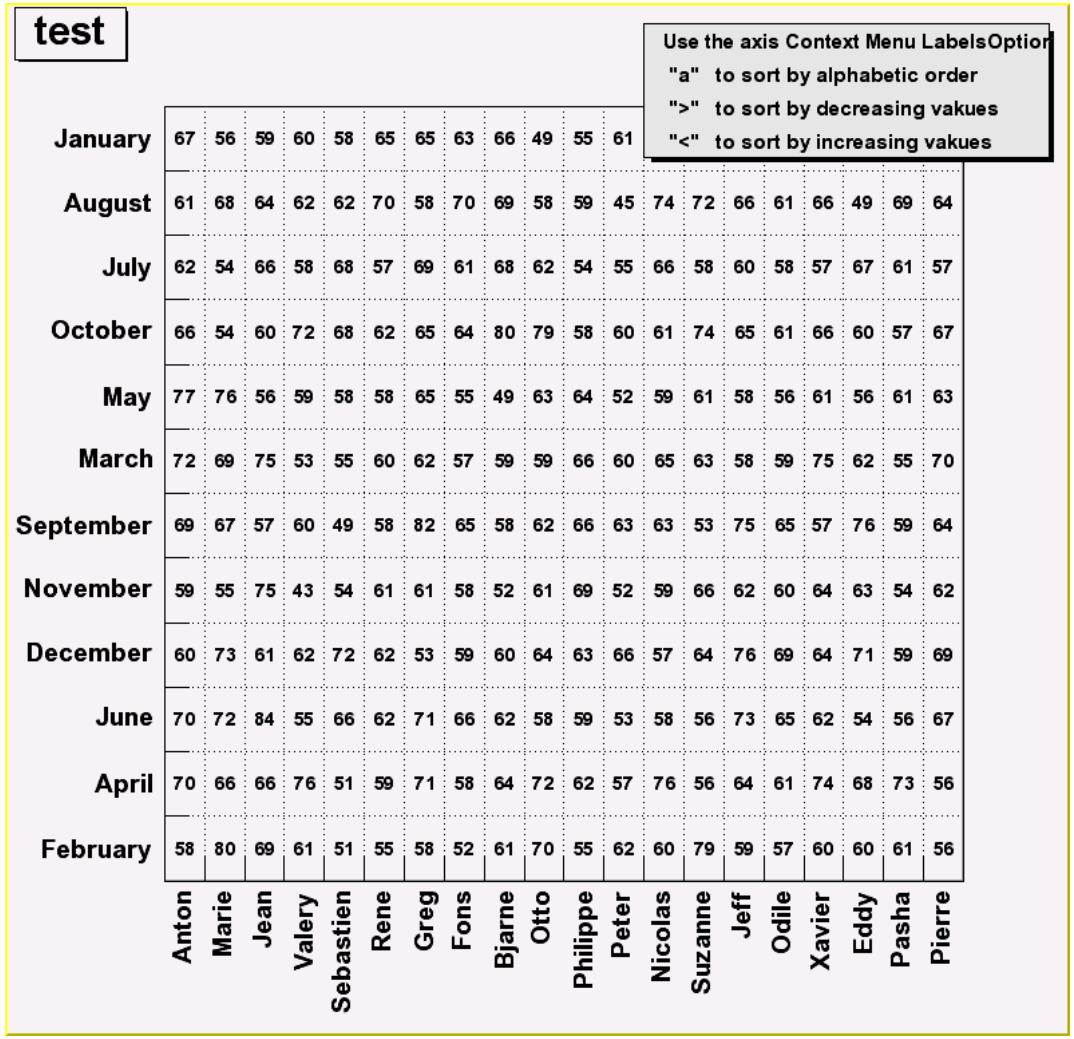
Filling with strings

See tutorials

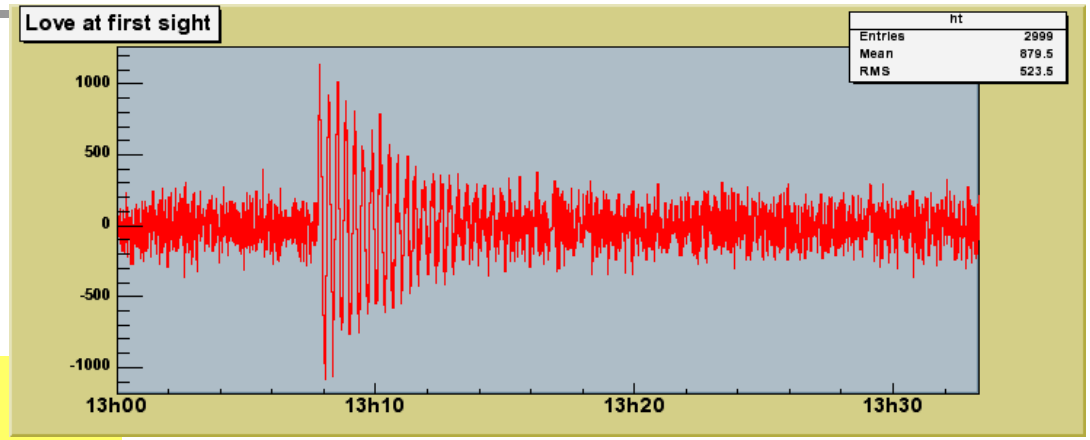
-hlabels1.C

-hlabels2.C

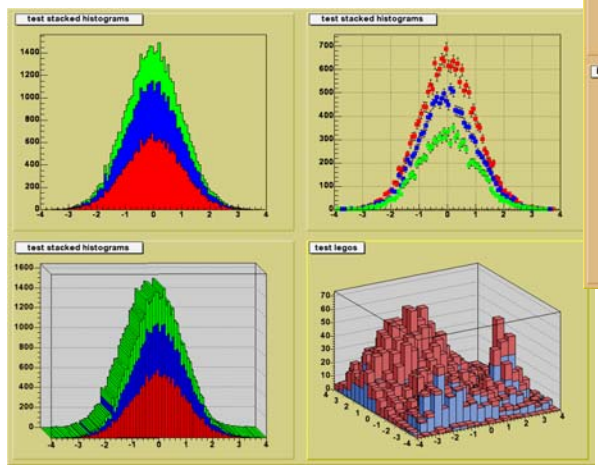
-cernstaff.C

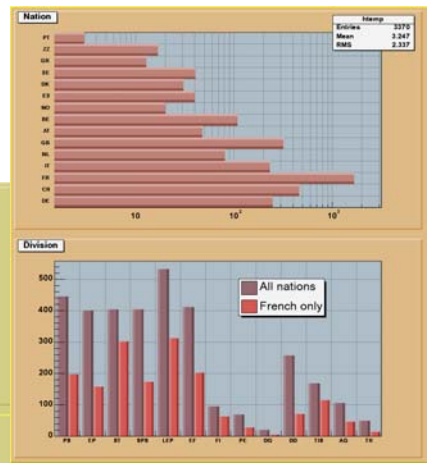


Histograms



- New class THStack
- Long list of new functions in TH1
- Plenty of new drawing options
- Filling with string variables
- TH1::Merge(TCollection *list)

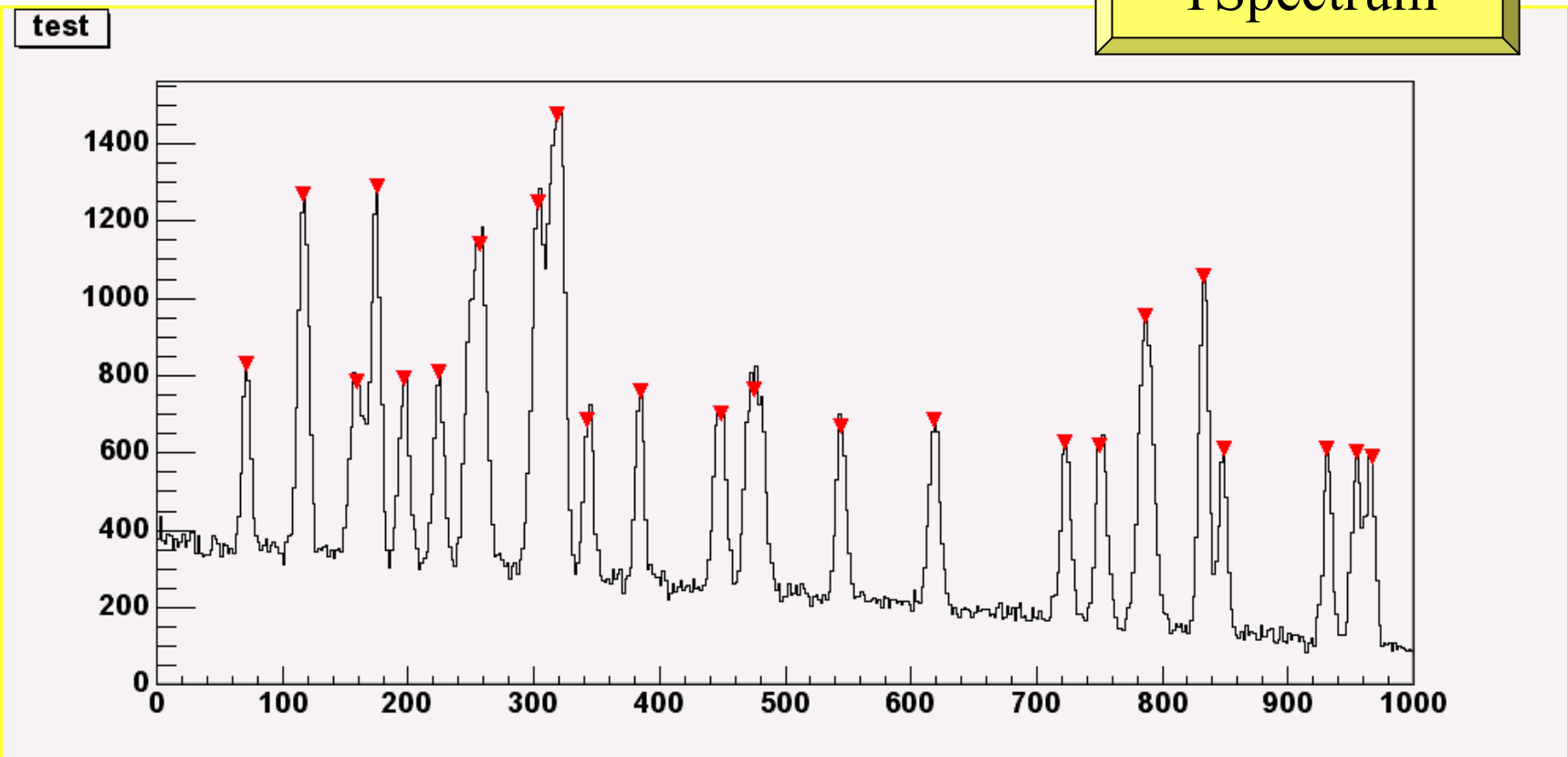




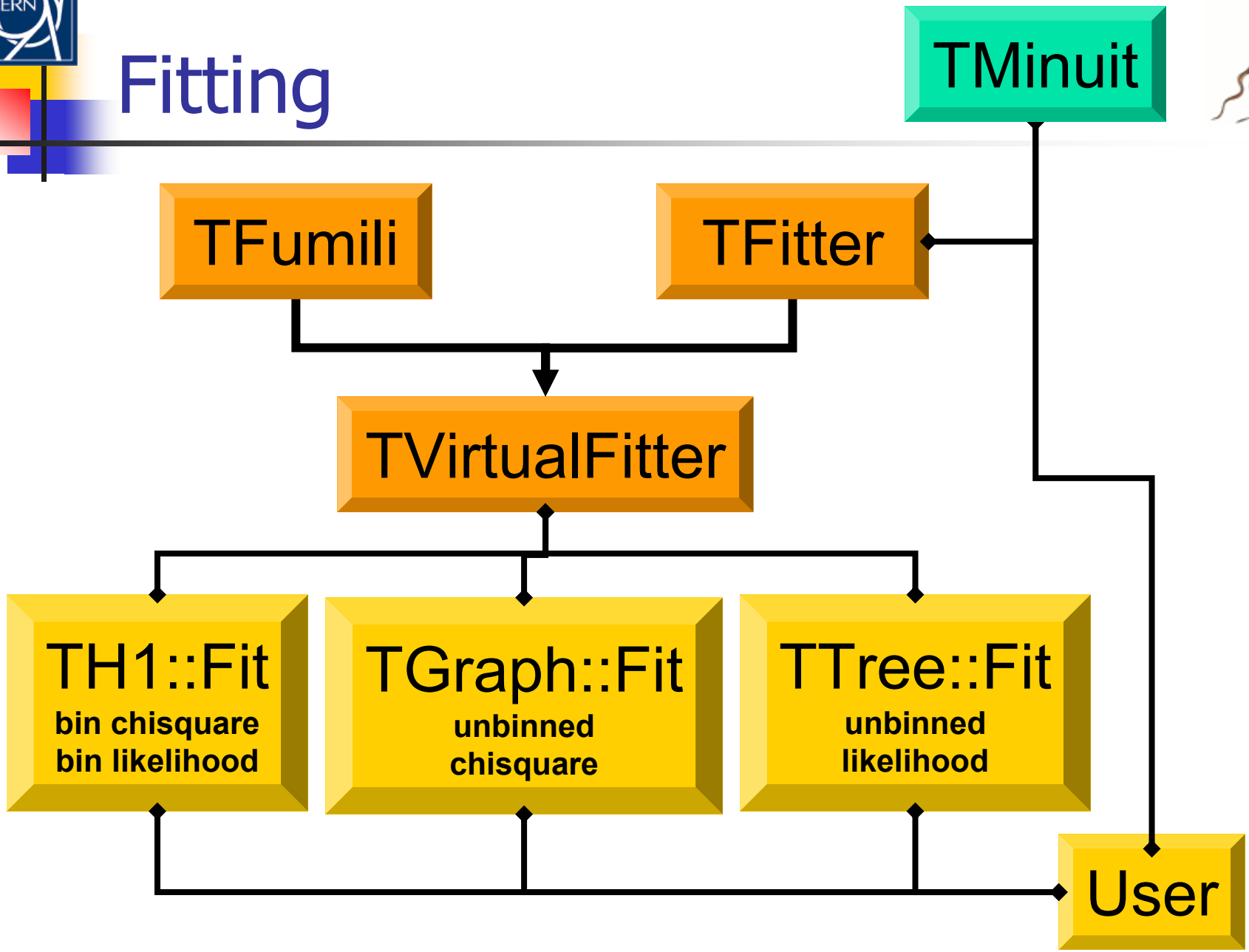
Peak Finder + Deconvolutions

by Miroslav Morach

TSpectrum



Fitting

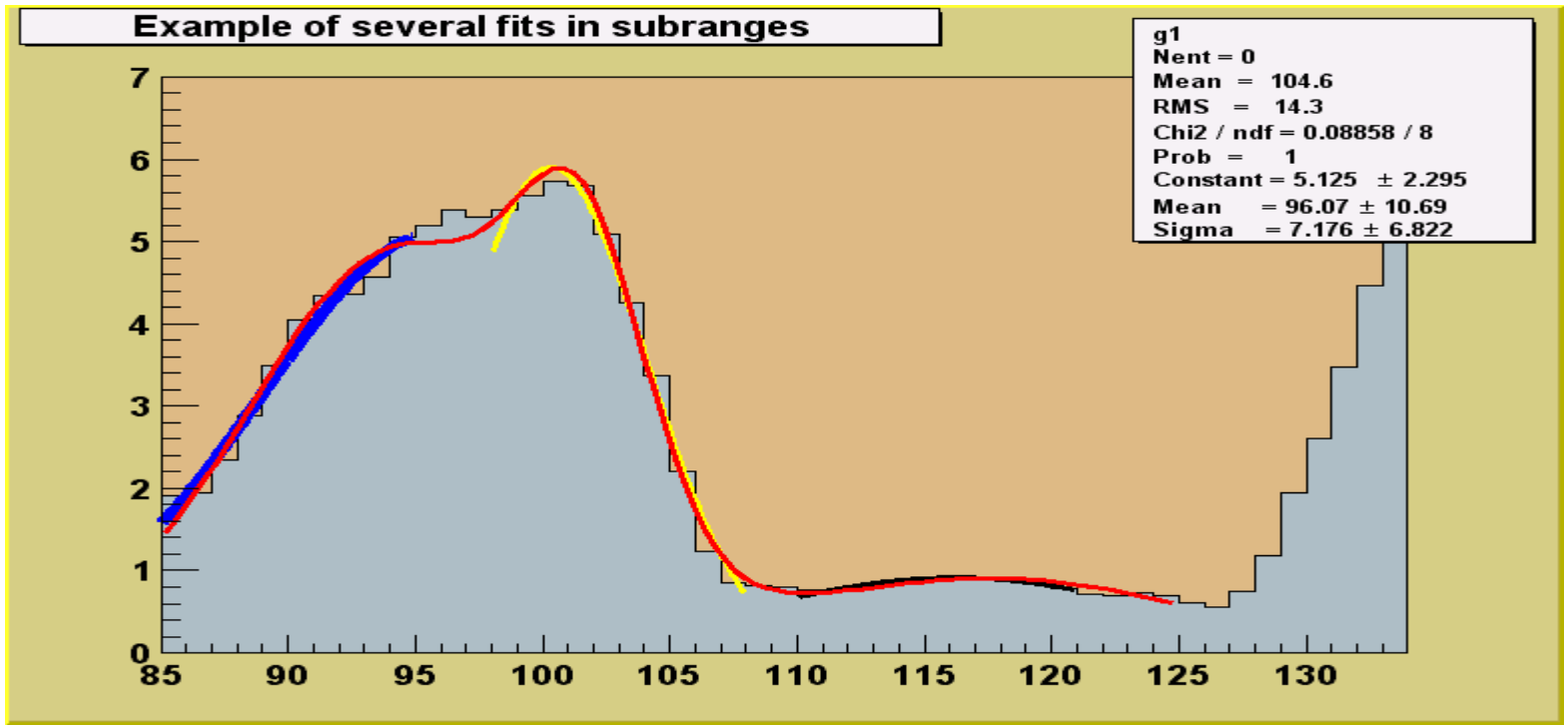

TMinuit
TFumili
TFitter
TVirtualFitter
TH1::Fit
bin chisquare
bin likelihood
TGraph::Fit
unbinned
chisquare
TTree::Fit
unbinned
likelihood
User

Fitting histograms

- Histograms (1-D,2-D,3-D and Profiles) can be fitted with a user specified function via TH1::Fit. Two Fitting algorithms are supported: Chisquare method and Log Likelihood
- The user functions may be of the following types:
  - standard functions: gaus, landau, expo, poln
  - combination of standard functions; poln + gaus
  - A C++ interpreted function or a C++ precompiled function
- An option is provided to compute the integral of the function bin by bin instead of simply compute the function value at the center of the bin.
- When an histogram is fitted, the resulting function with its parameters is added to the list of functions of this histogram. If the histogram is made persistent, the list of associated functions is also persistent.
- One can retrieve the function/fit parameters with calls such as:
  - ***Double_t chi2 = myfunc->GetChisquare();***
  - ***Double_t par0 = myfunc->GetParameter(0); //value of 1st parameter***
  - ***Double_t err0 = myfunc->GetParError(0); //error on first parameter***

Associated functions

- One or more object (typically a TF1*) can be added to the list of functions associated to each histogram.
- When TF1::Fit is invoked, the fitted function is added to this list.
- Given an histogram h, one can retrieve an associated function with:
  - ***TF1 *myfunc = h->GetFunction("myfunc");***



Operations on histograms

- Many types of operations are supported on histograms or between histograms
  - Addition of an histogram to the current histogram
  - Additions of two histograms with coefficients and storage into the current histogram
  - Multiplications and Divisions are supported in the same way as additions.
- The Add, Divide and Multiply functions also exist to add,divide or multiply an histogram by a function. If an histogram has associated error bars (TH1::Sumw2 has been called), the resulting error bars are also computed assuming independent histograms.
- In case of divisions, Binomial errors are also supported.
- The standard operators +, -, *, / are supported.

Random Numbers and Histograms

- TH1::FillRandom can be used to randomly fill an histogram using
  - the contents of an existing TF1 analytic function
  - another histogram (for all dimensions).

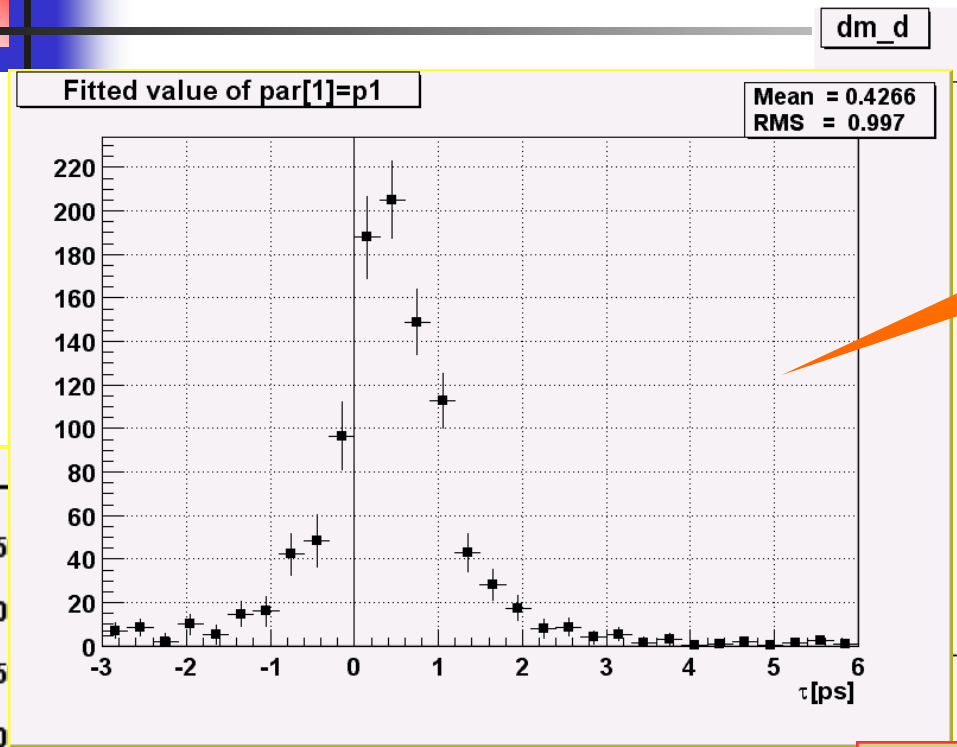
- For example the following two statements create and fill an histogram 10000 times with a default gaussian distribution of mean 0 and sigma 1:
  - ***TH1F h1("h1","histo from a gaussian",100,-3,3);***
  - ***h1.FillRandom("gaus",10000);***

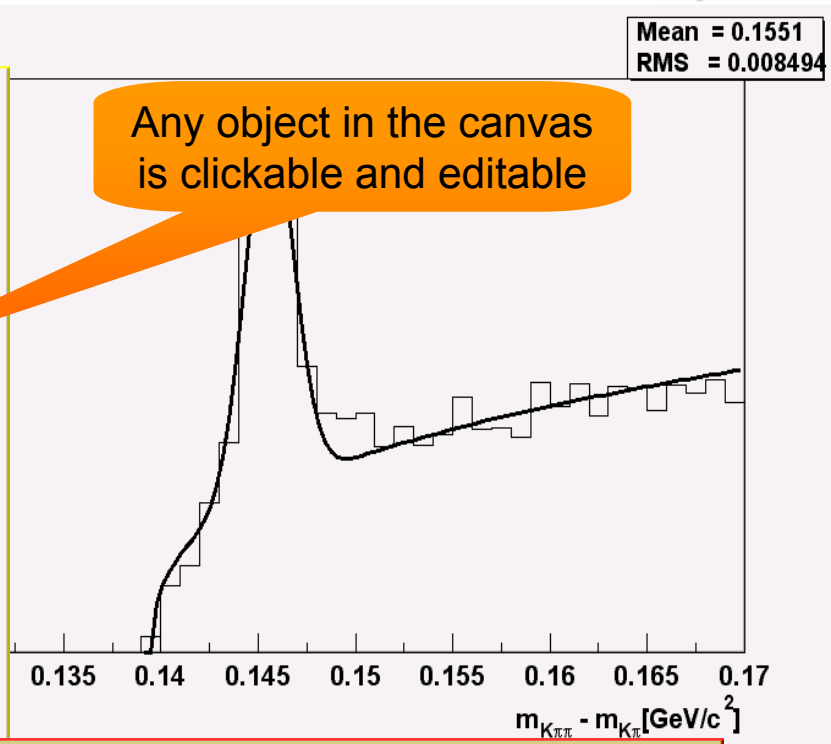
- TH1::GetRandom can be used to return a random number distributed according the contents of an histogram.

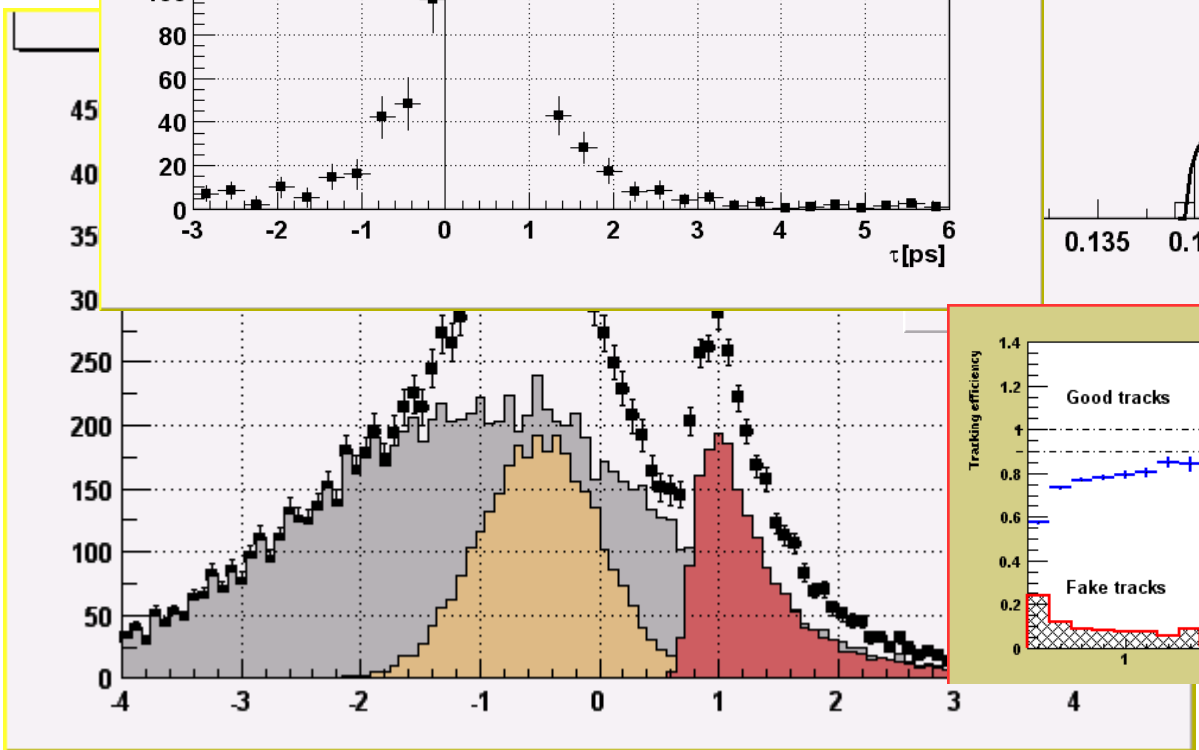
Drawing Histograms

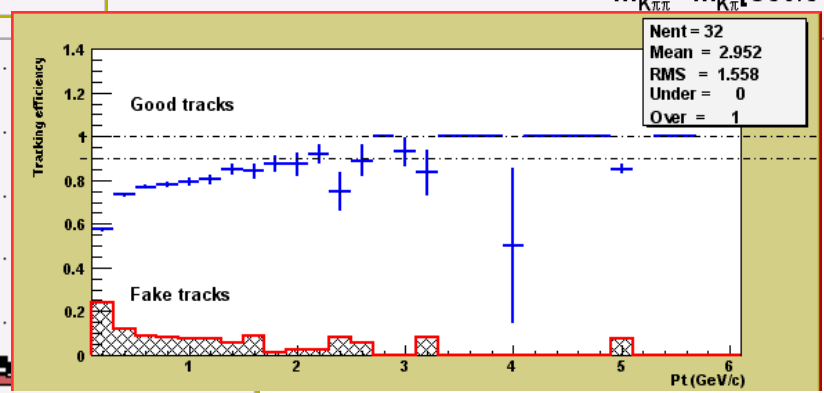
- When you call the Draw method of a histogram for the first time (TH1::Draw), it creates a THistPainter object and saves a pointer to painter as a data member of the histogram.
- The THistPainter class specializes in the drawing of histograms. It is separate from the histogram so that one can have histograms without the graphics overhead, for example in a batch program. The choice to give each histogram have its own painter rather than a central singleton painter, allows two histograms to be drawn in two threads without overwriting the painter's values.
- When a displayed histogram is filled again you do not have to call the Draw method again. The image is refreshed the next time the pad is updated.
- The same histogram can be drawn with different graphics options in different pads.
- When a displayed histogram is deleted, its image is automatically removed from the pad.

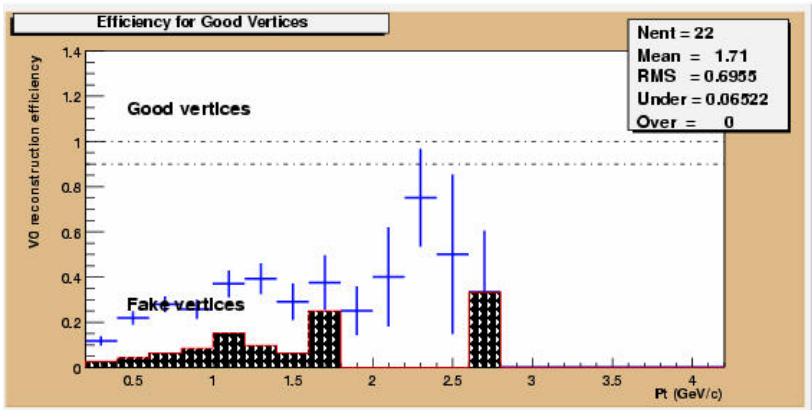
1-D drawing Options

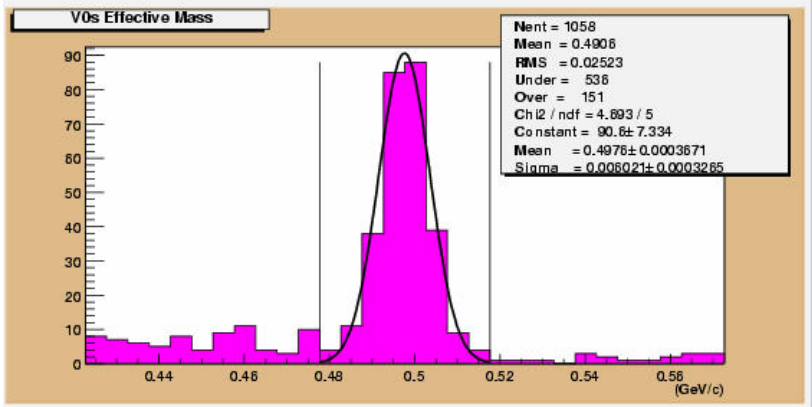


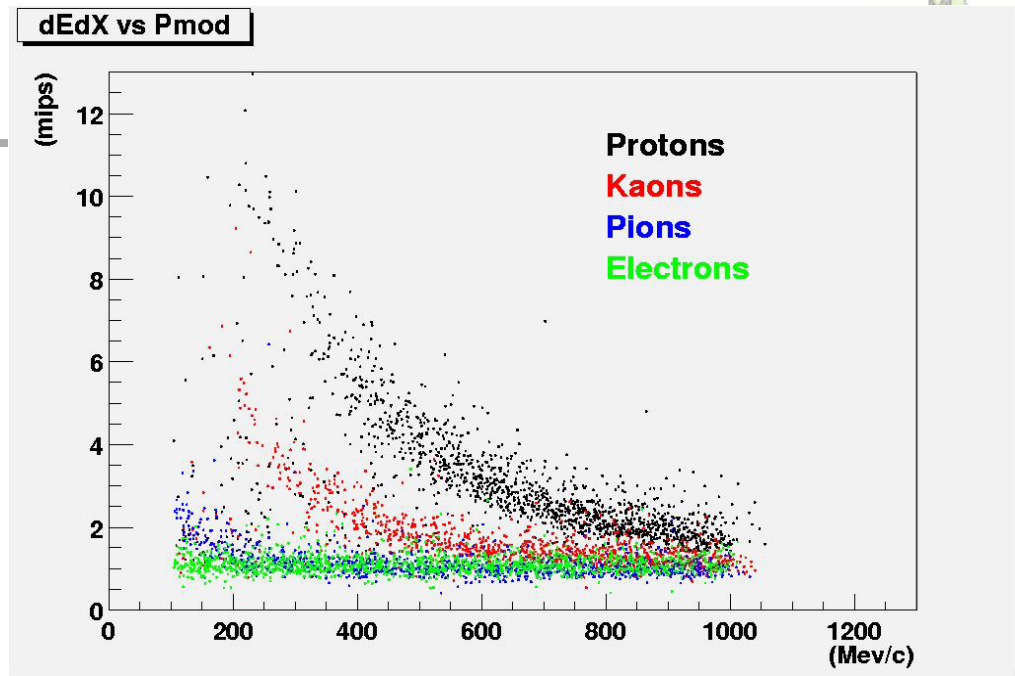


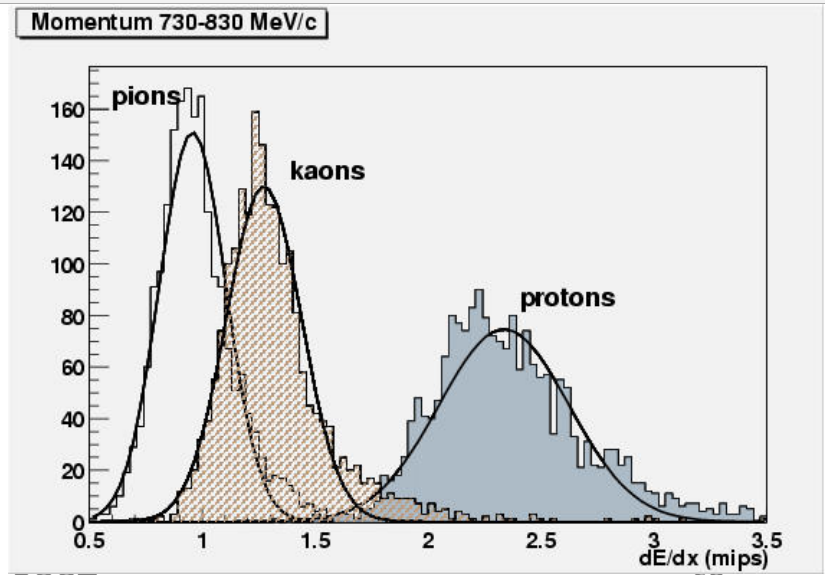




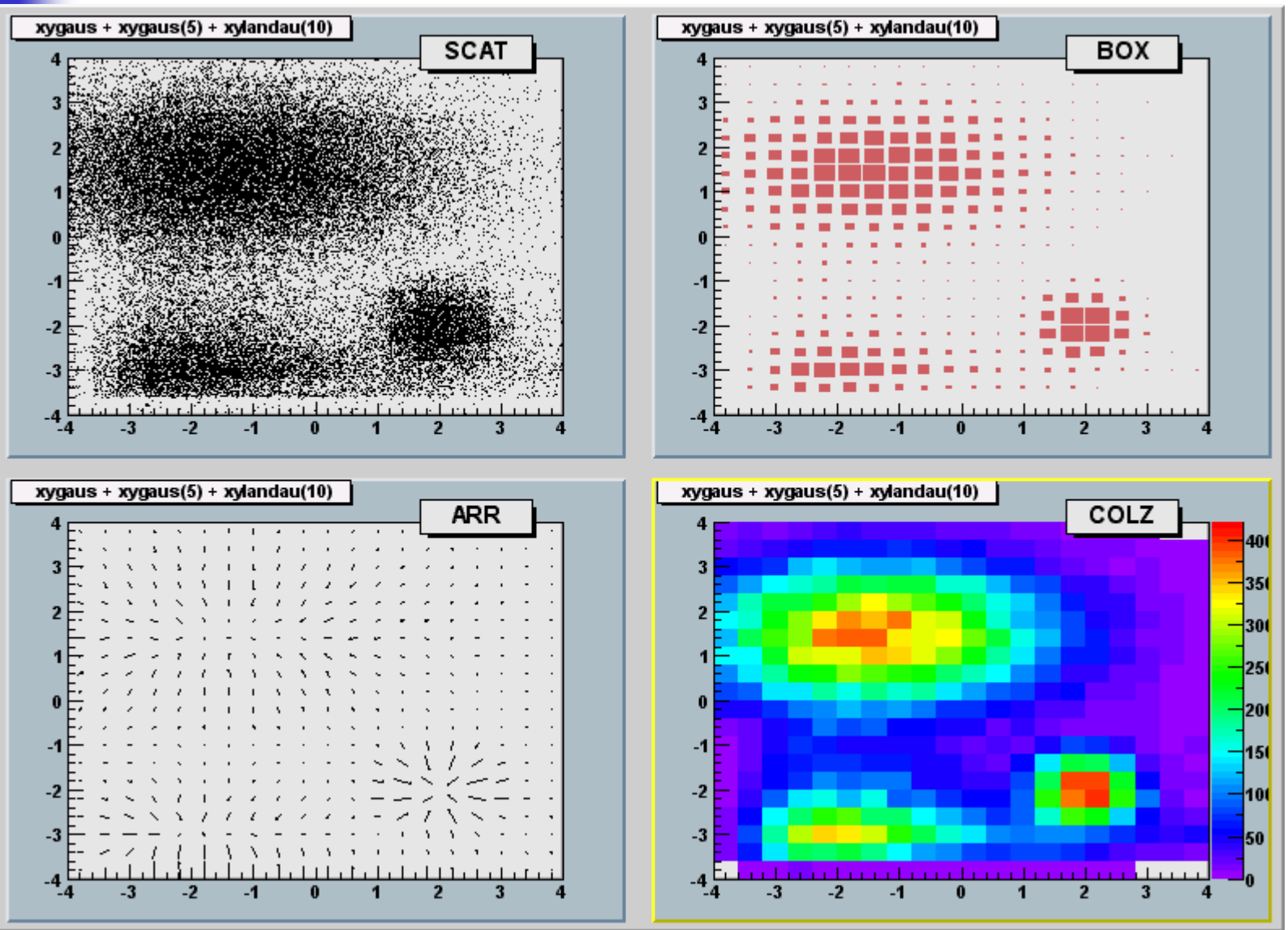

Efficiency for Good Vertices
Nent = 22
Mean = 1.71
RMS = 0.6955
Under = 0.06522
Over = 0
Good vertices
Fake vertices
V0 reconstruction efficiency
Pt (GeV/c)


V0s Effective Mass
Nent = 1058
Mean = 0.4906
RMS = 0.02523
Under = 536
Over = 151
Chi2 / ndf = 4.693 / 5
Constant = 90.6± 7.334
Mean = 0.4976± 0.0003671
Sigma = 0.006021± 0.0003265
(GeV/c)

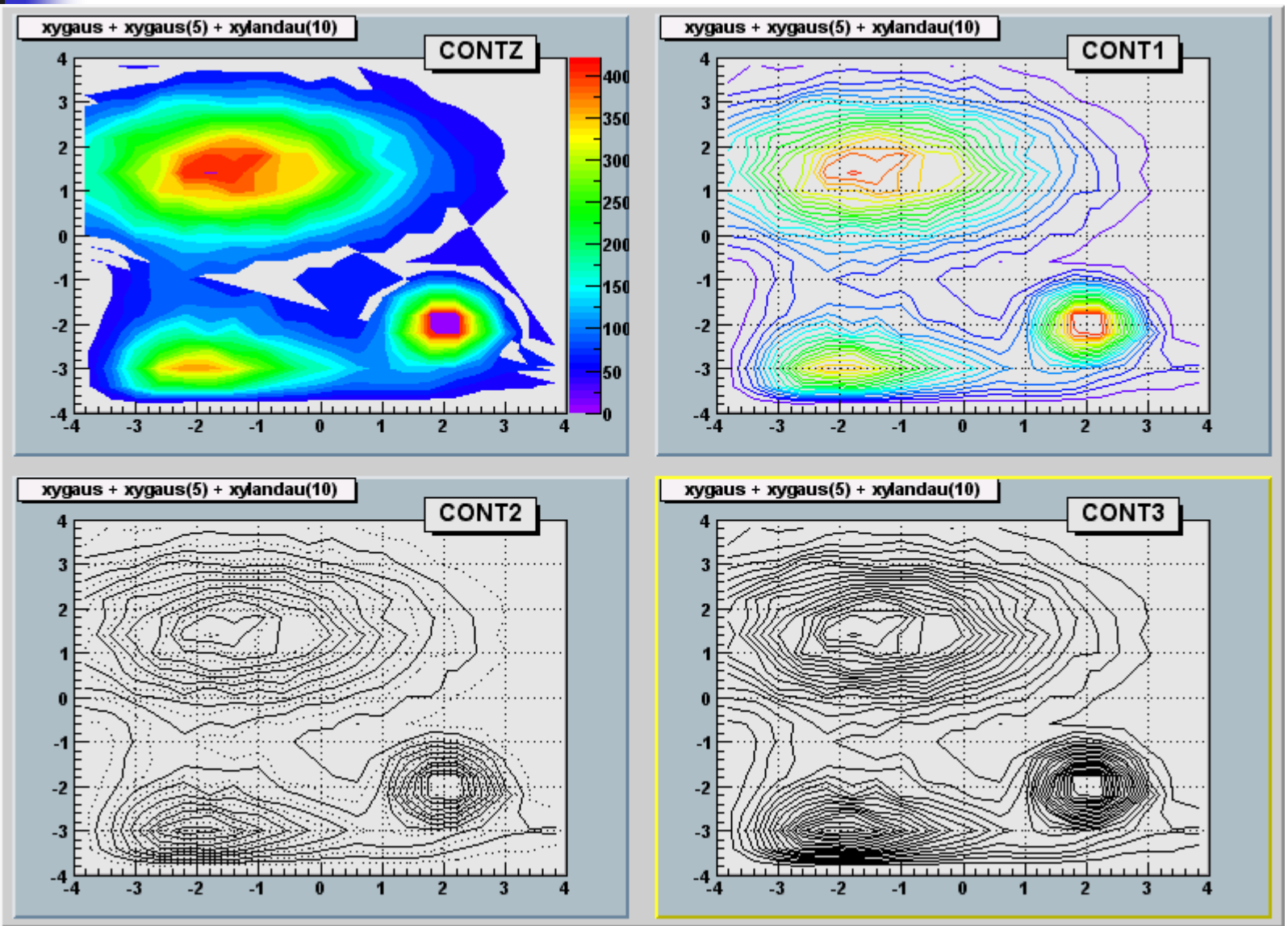

dEdX vs Pmod
(mips)
Protons
Kaons
Pions
Electrons
(Mev/c)


Momentum 730-830 MeV/c
pions
kaons
protons
dE/dx (mips)

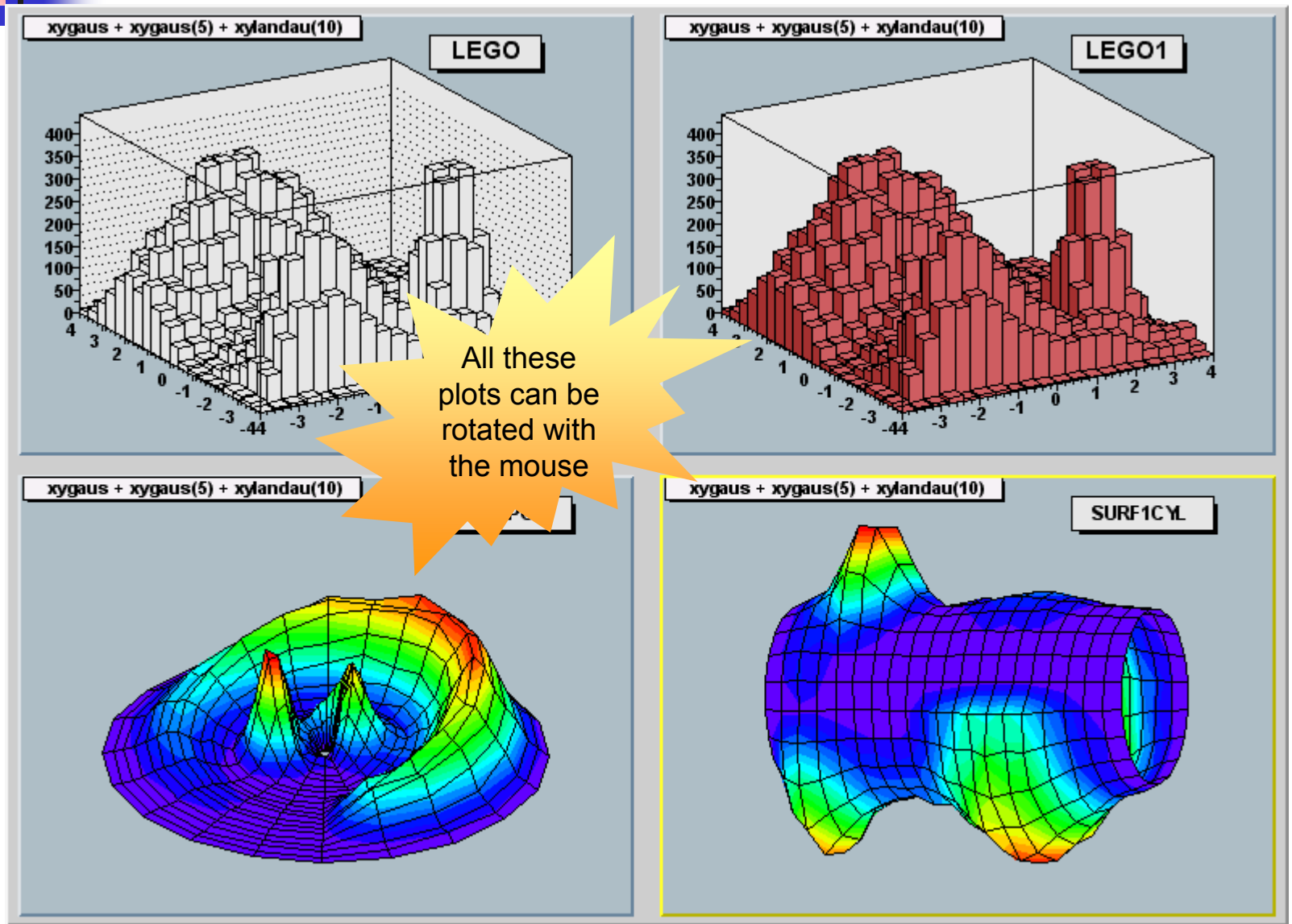
2-D drawing options



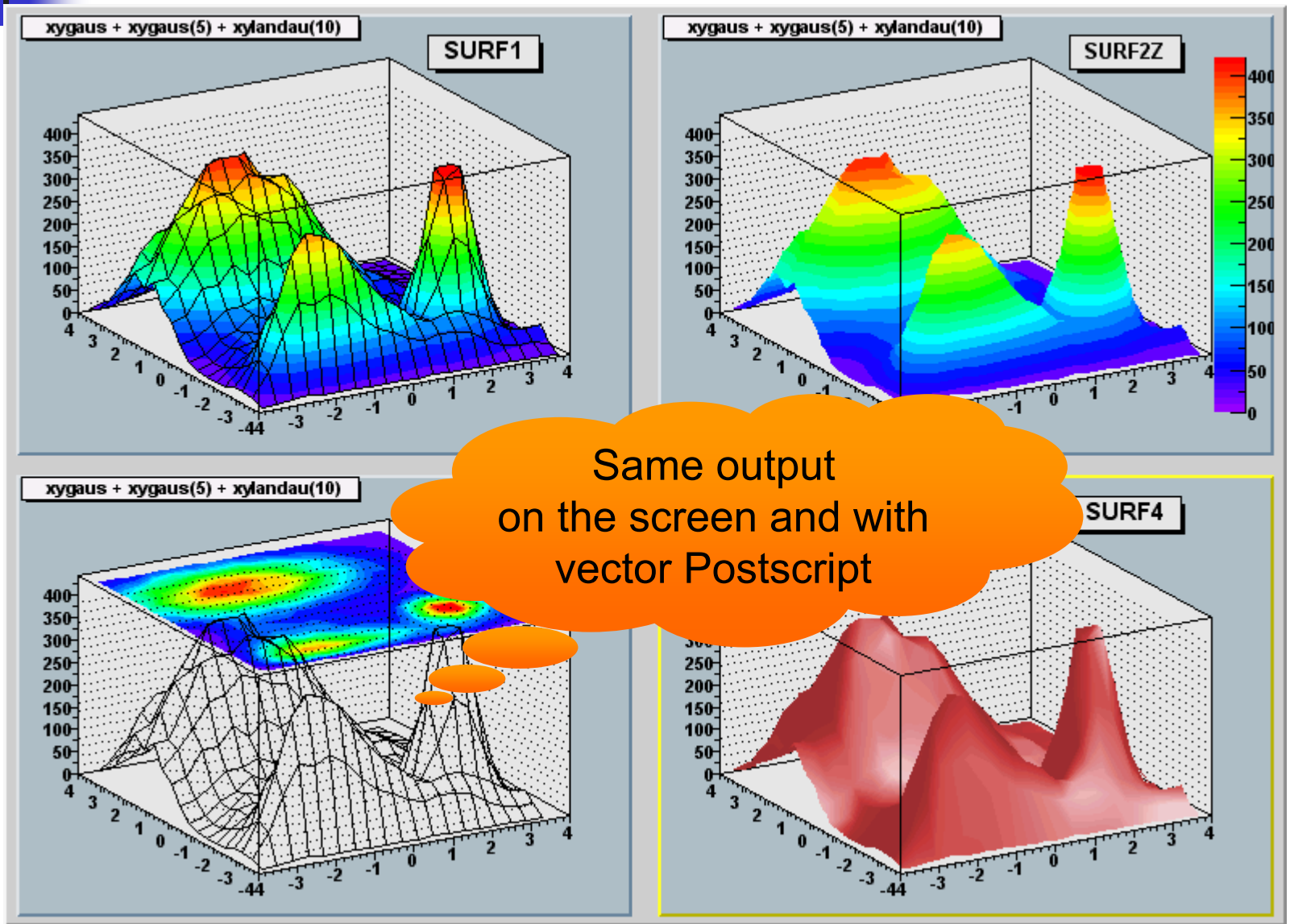
2-D drawing options



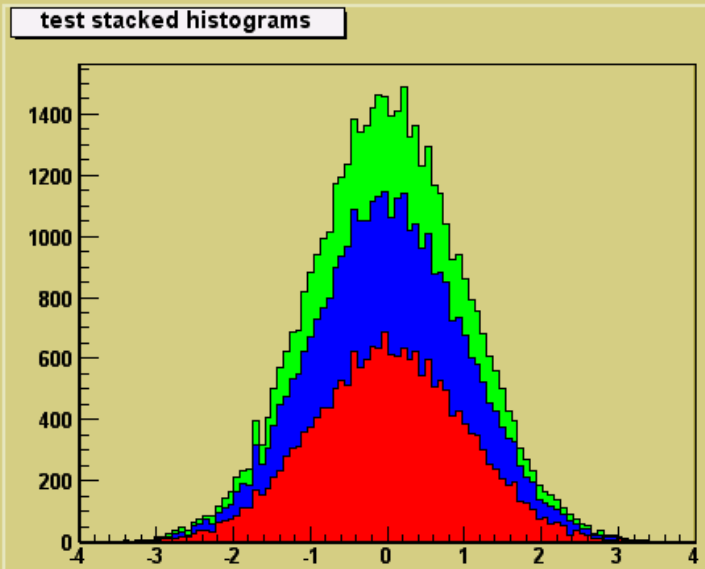
2-D drawing Options

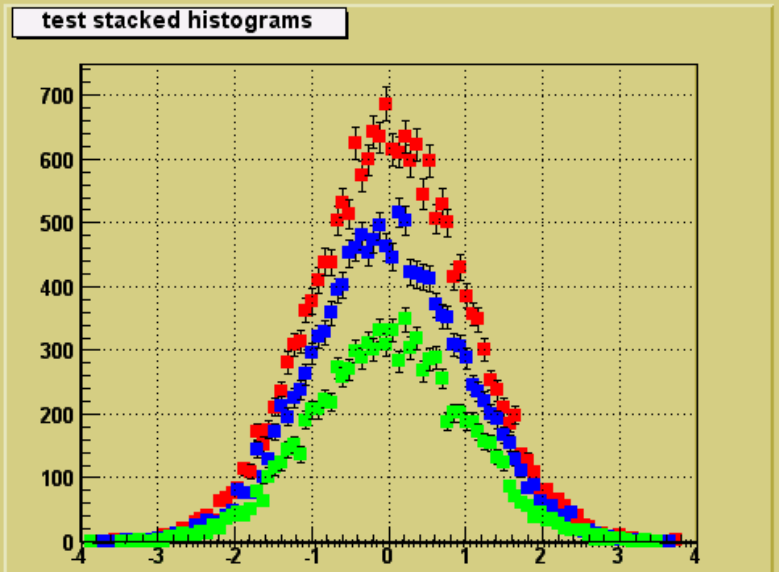


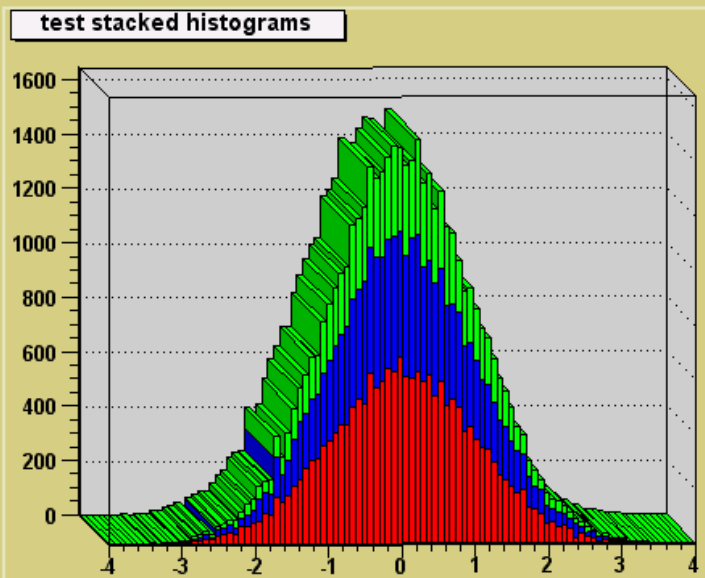
2-D drawing Options

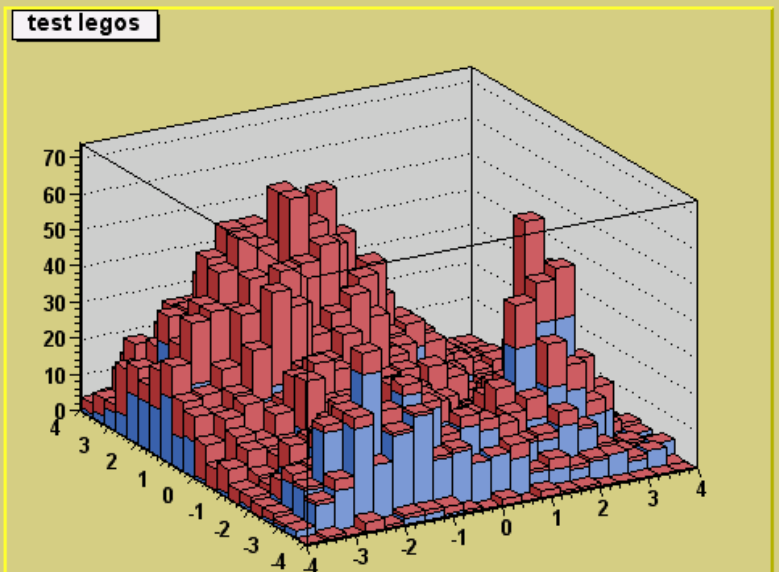


THStack examples

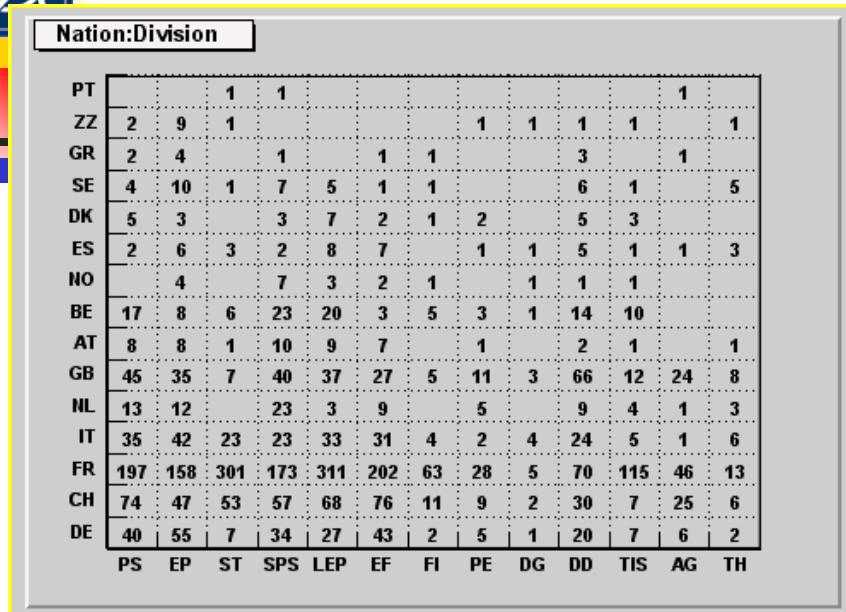


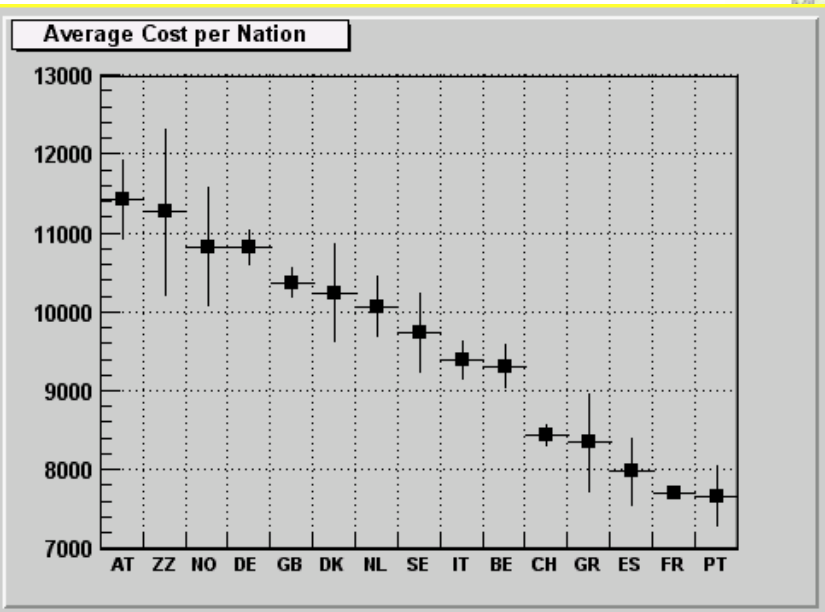


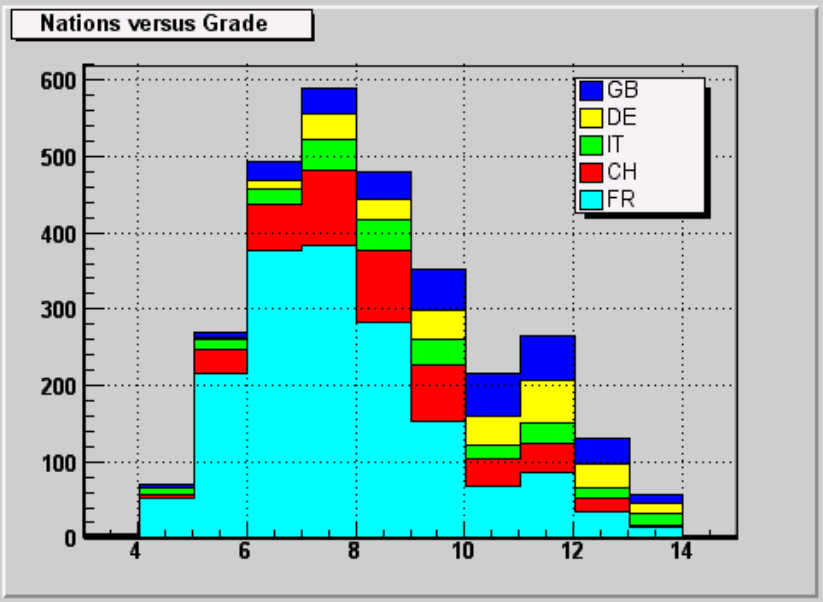


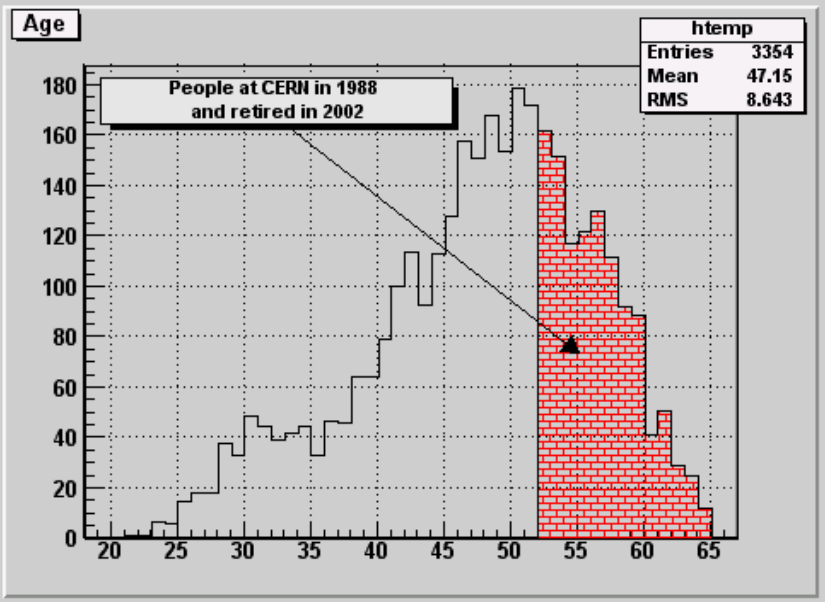


Filling with string variables









Math Libs & Statistics

In ROOT today

TVector2,3

TLorentzRotation

TLorentzVector

TRandom,2,3

TMatrix

TMath

TFeldmanCousins

TPrincipal

TMultidimFit

TConfidenceLevel

TFractionFitter

Math Libs

Statistics

Can generate random numbers

from basic distributions; gaus, poisson, etc

from parametric analytic functions 1,2,3-d

from histograms, 1,2,3-d

Event

Matrix package maintained by E. Offermann (Rentec)

A collection of many algorithms

CERNLIB, Numerical Recipes in C/C++

Many algorithms classes

developed by a huge user community

See recent FNAL meeting

and effort organized within ACAT

Would like to see an interface

to GSL

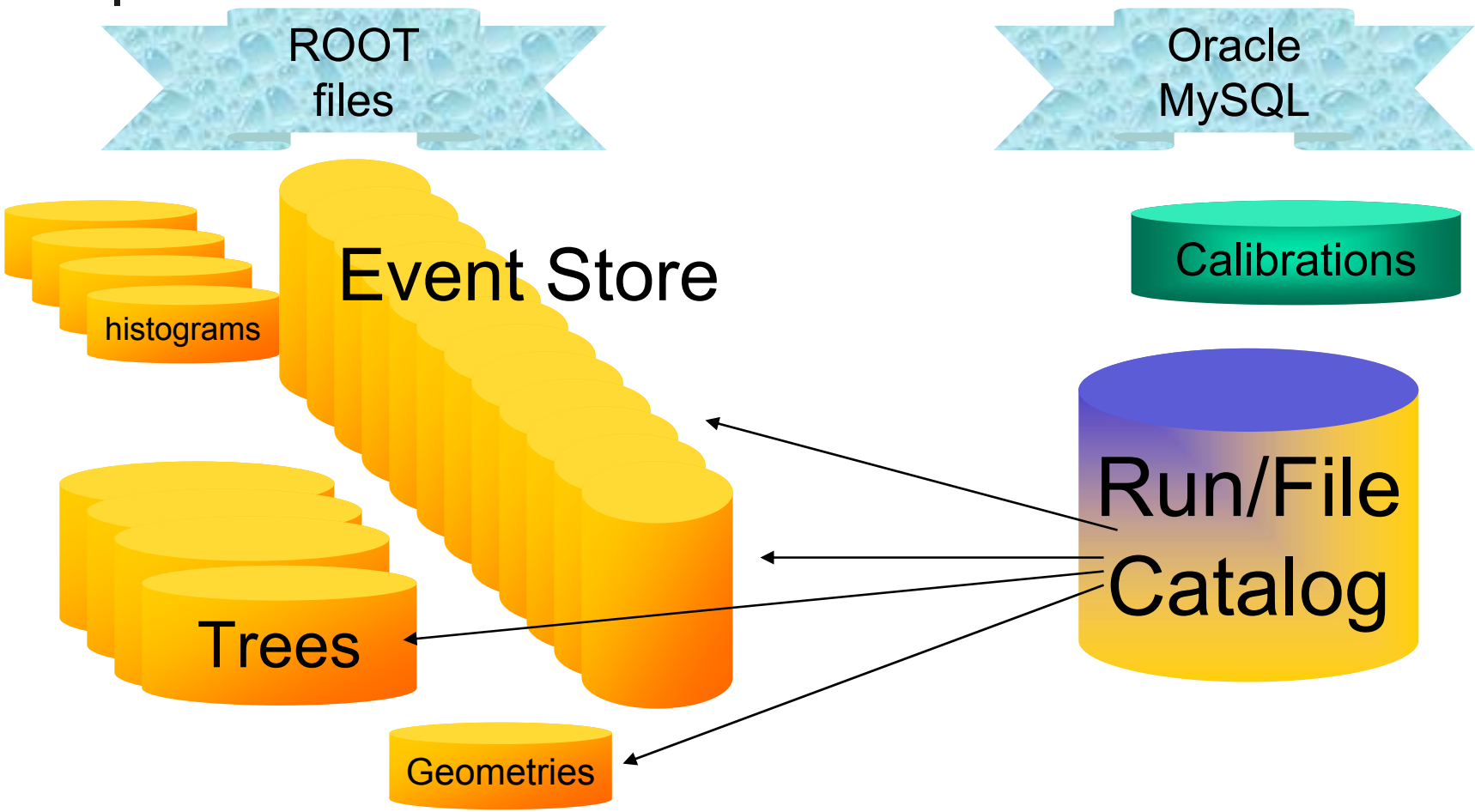
to Numerical Recipes in C++

Collaboration with

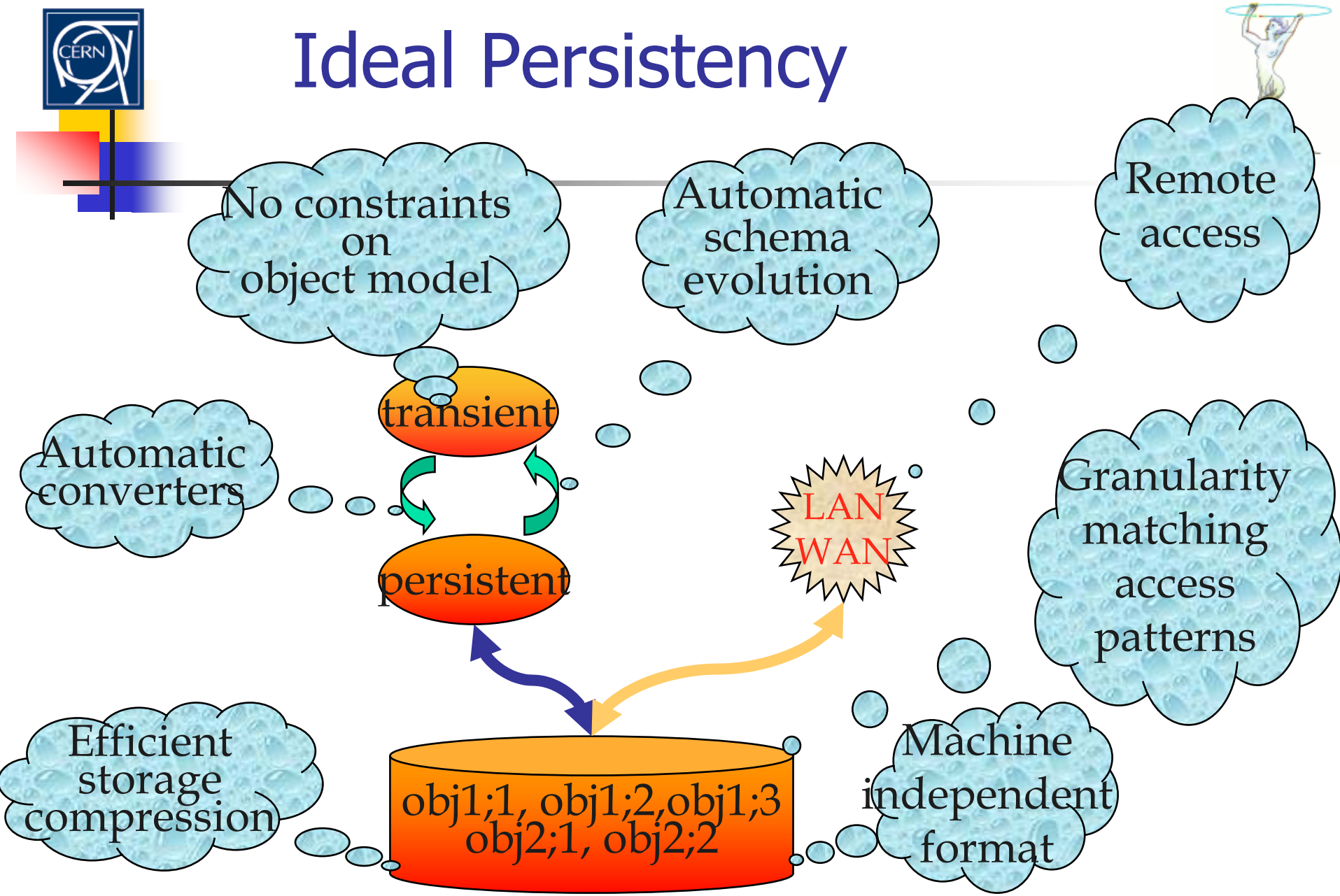
Fred James, Louis Lyons, Sherry Towers

Input/Output

ROOT + RDBMS Model



Ideal Persistency



Object Persistency (in a nutshell)

- Two I/O modes supported (Keys and Trees).
- Key access: simple object streaming mode.
  - A ROOT file is like a Unix directory tree
  - Very convenient for objects like histograms, geometries, mag.field, calibrations
- Trees
  - A generalization of ntuples to objects
  - Designed for storing events
  - split and no split modes
  - query processor
- Chains: Collections of files containing Trees
- ROOT files are self-describing
- Interfaces with RDBMS also available
- Access to remote files (RFIO, DCACHE, GRID)

ROOT I/O : An Example

Program Writing

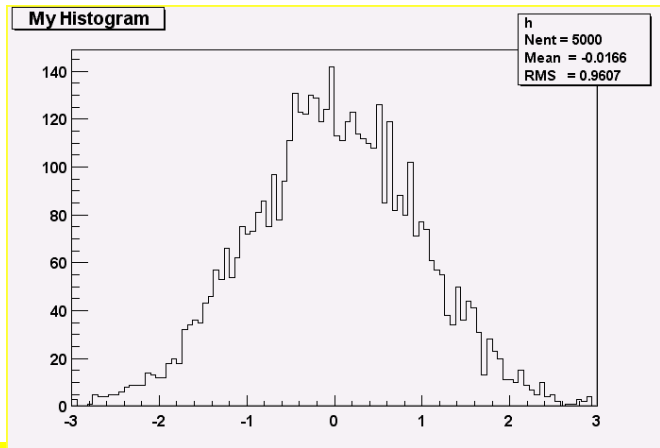
demoh.C

```
TFile f("example.root","new");
TH1F h("h","My histogram",100,-3,3);
h.FillRandom("gaus",5000);
h.Write();
```

Program Reading

demohr.C

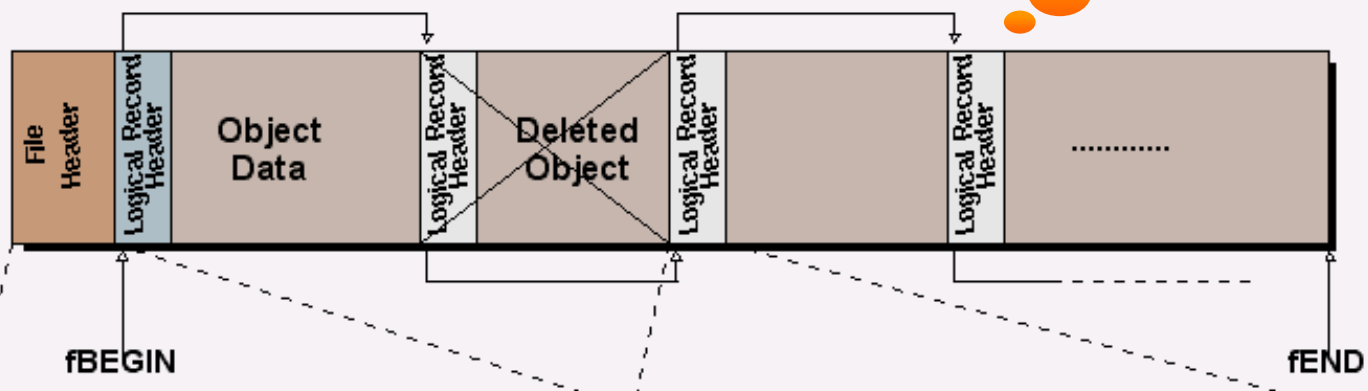
```
TFile f("example.root");
TH1F *h = (TH1F*)f.Get("h");
h->Draw();
f.Map();
```




```
20010831/171903  At:64      N=90      TFile
20010831/171941  At:154     N=453     TH1F          CX =  2.09
20010831/171946  At:607     N=2364    StreamerInfo  CX =  3.25
20010831/171946  At:2971    N=96      KeysList
20010831/171946  At:3067    N=56      FreeSegments
20010831/171946  At:3123    N=1       END
```

All what you need to known to navigate in a ROOT file

ROOT File description

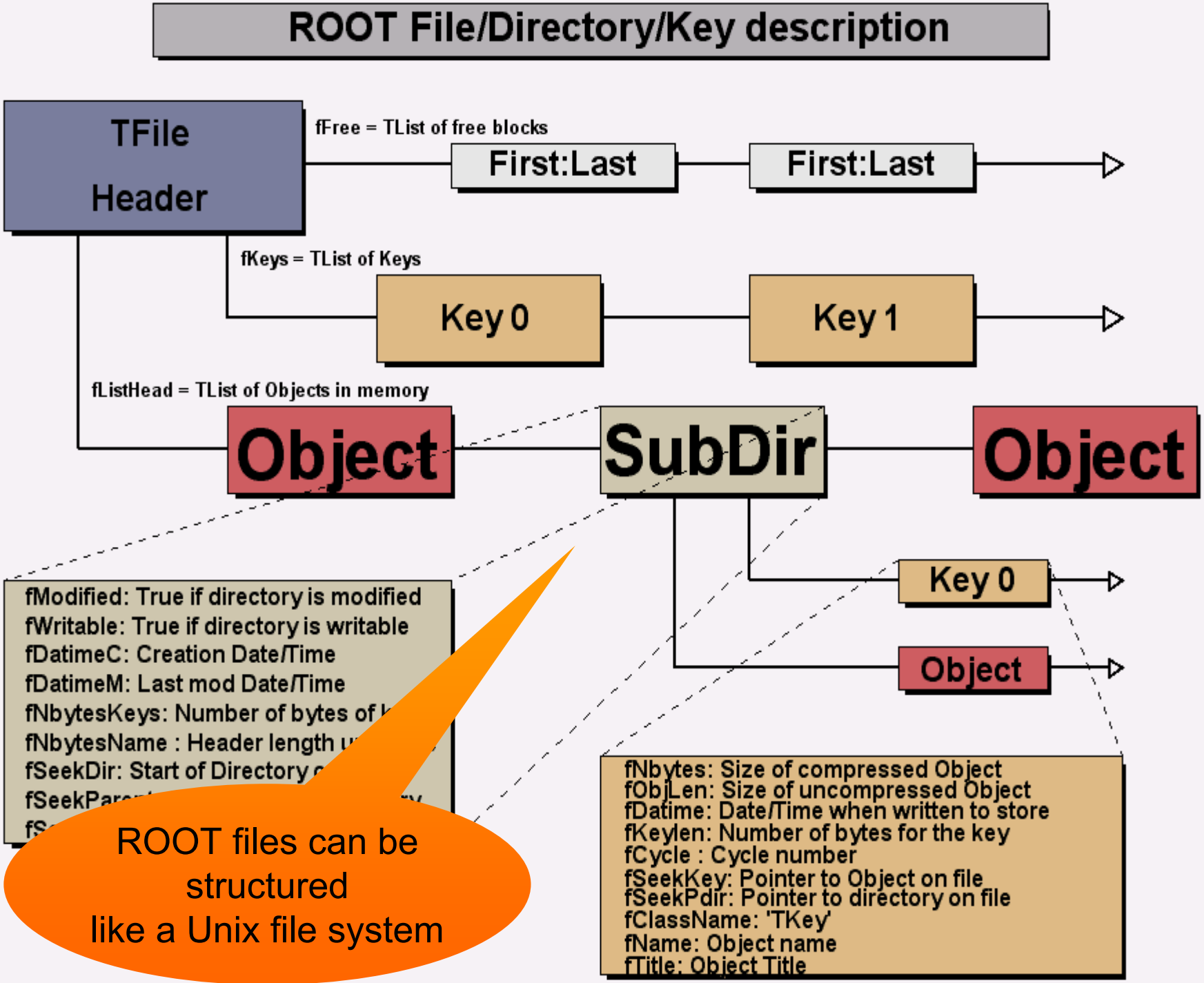


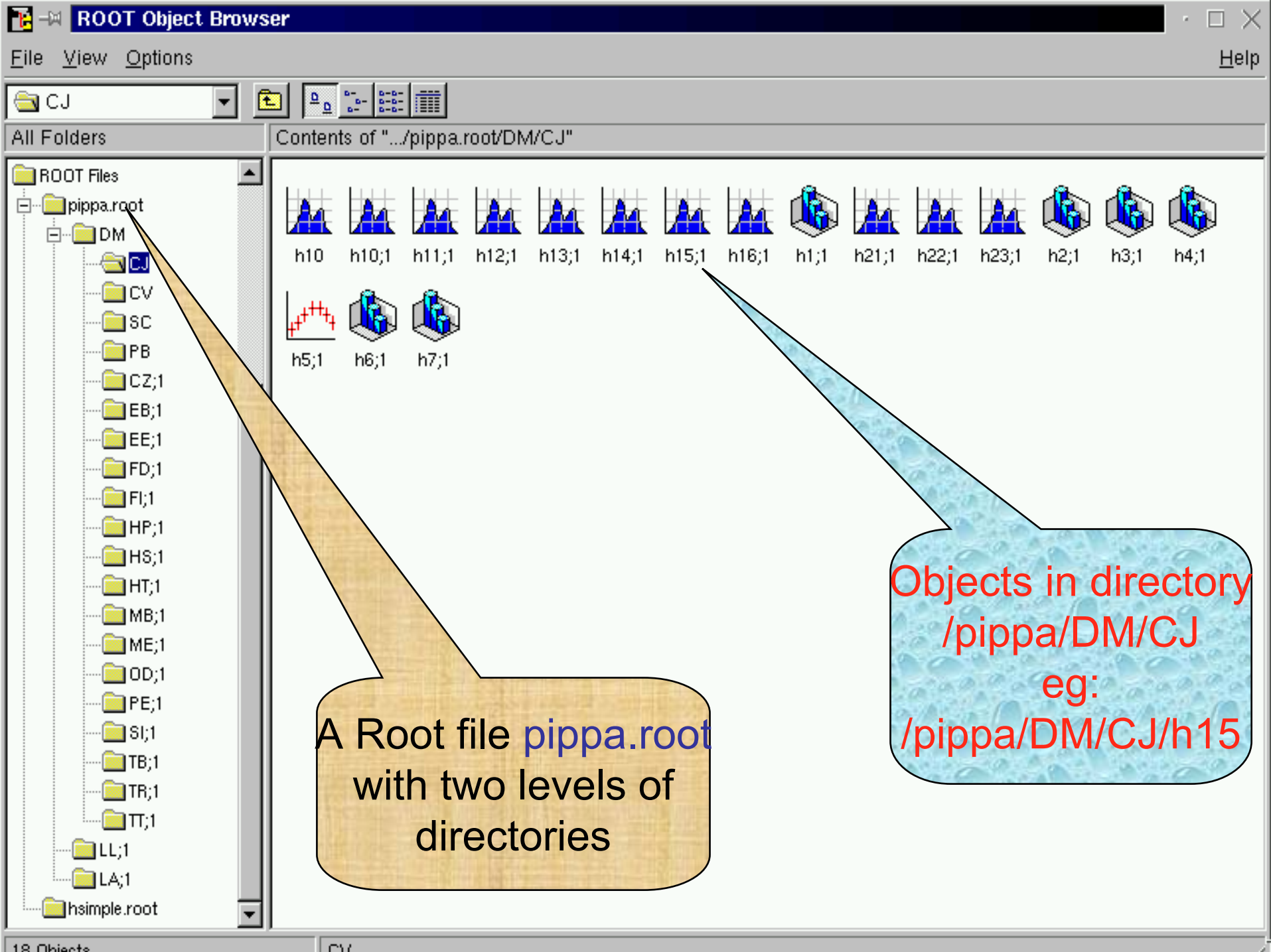
File Header

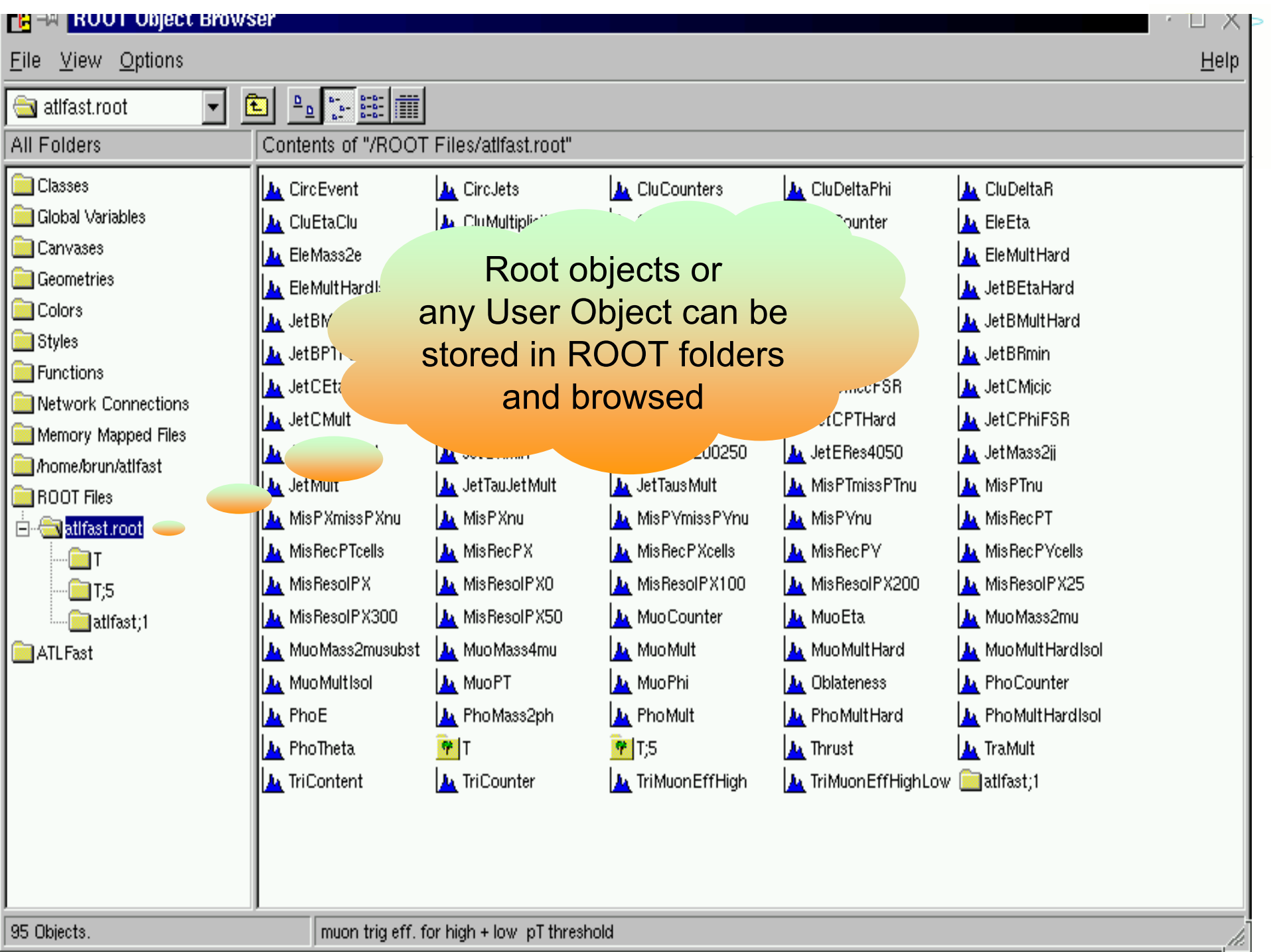
"root": Root File Identifier
fVersion: File version identifier
fBEGIN: Pointer to first data record
fEND: Pointer to first free word at EOF
fSeekFree: Pointer to FREE data record
fNbytesFree: Number of bytes in FREE
fNfree: Number of free data records
fNbytesName: Number of bytes in name/title
fUnits: Number of bytes for pointers
fCompress: Compression level

Logical Record Header (TKEY)

fNbytes: Length of compressed object
fVersion: Key version identifier
fObjLen: Length of uncompressed object
fDatime: Date/Time when written to store
fKeylen: Number of bytes for the key
fCycle : Cycle number
fSeekKey: Pointer to object on file
fSeekPdir: Pointer to directory on file
fClassName: class name of the object
fName: name of the object
fTitle: title of the object


ROOT File/Directory/Key description
TFile
Header
fFree = TList of free blocks
First:Last
First:Last
fKeys = TList of Keys
Key 0
Key 1
fListHead = TList of Objects in memory
Object
SubDir
Object
Key 0
Object
fModified: True if directory is modified
fWritable: True if directory is writable
fDatimeC: Creation Date/Time
fDatimeM: Last mod Date/Time
fNbytesKeys: Number of bytes of
fNbytesName : Header length
fSeekDir: Start of Directory
fNbytes: Size of compressed Object
fObjLen: Size of uncompressed Object
fDatime: Date/Time when written to store
fKeylen: Number of bytes for the key
fCycle : Cycle number
fSeekKey: Pointer to Object on file
fSeekPdir: Pointer to directory on file
fClassName: 'TKey'
fName: Object name
fTitle: Object Title
ROOT files can be
structured
like a Unix file system


ROOT Object Browser
File View Options Help
CJ
All Folders
Contents of ".../pippa.root/DM/CJ"
ROOT Files
pippa.root
DM
CJ
CV
SC
PB
CZ;1
EB;1
EE;1
FD;1
FI;1
HP;1
HS;1
HT;1
MB;1
ME;1
OD;1
PE;1
SI;1
TB;1
TR;1
TT;1
LL;1
LA;1
hsimple.root
h10
h10;1
h11;1
h12;1
h13;1
h14;1
h15;1
h16;1
h1;1
h21;1
h22;1
h23;1
h2;1
h3;1
h4;1
h5;1
h6;1
h7;1
A Root file pippa.root with two levels of directories
Objects in directory /pippa/DM/CJ eg: /pippa/DM/CJ/h15


ROOT Object Browser
File View Options
Help
atlfast.root
All Folders
Contents of "/ROOT Files/atlfast.root"
Classes
Global Variables
Canvases
Geometries
Colors
Styles
Functions
Network Connections
Memory Mapped Files
/home/brun/atlfast
ROOT Files
atlfast.root
T
T;5
atlfast;1
ATLFast
CircEvent
CircJets
CluCounters
CluDeltaPhi
CluDeltaR
CluEtaClu
EleEta
EleMass2e
EleMultHard
JetBEtaHard
JetBMultHard
JetBRmin
JetCMjcjc
JetCPhiFSR
JetCMult
JetERes4050
JetMass2jj
JetMult
JetTauJetMult
JetTausMult
MisPTmissPTnu
MisPTnu
MisPXmissPXnu
MisPXnu
MisPYmissPYnu
MisPYnu
MisRecPT
MisRecPTcells
MisRecPX
MisRecPXcells
MisRecPY
MisRecPYcells
MisResolPX
MisResolPX0
MisResolPX100
MisResolPX200
MisResolPX25
MisResolPX300
MisResolPX50
MuoCounter
MuoEta
MuoMass2mu
MuoMass2musubst
MuoMass4mu
MuoMult
MuoMultHard
MuoMultHardIsol
MuoMultIsol
MuoPT
MuoPhi
Oblateness
PhoCounter
PhoE
PhoMass2ph
PhoMult
PhoMultHard
PhoMultHardIsol
PhoTheta
T
T;5
Thrust
TraMult
TriContent
TriCounter
TriMuonEffHigh
TriMuonEffHighLow
atlfast;1
Root objects or
any User Object can be
stored in ROOT folders
and browsed
95 Objects.
muon trig eff. for high + low pT threshold

LAN/WAN files

- Files and Directories
  - a directory holds a list of named objects
  - a file may have a hierarchy of directories (a la Unix)
  - ROOT files are machine independent
  - built-in compression
- Support for local, LAN and WAN files
  - TFile f1("myfile.root")
  - TFile f2("http://pcbrun.cern.ch/Renefile.root")
  - TFile f3("root://cdfsga.fnal.gov/bigfile.root")
  - TFile f4("rfio://alice/run678.root")

Local file

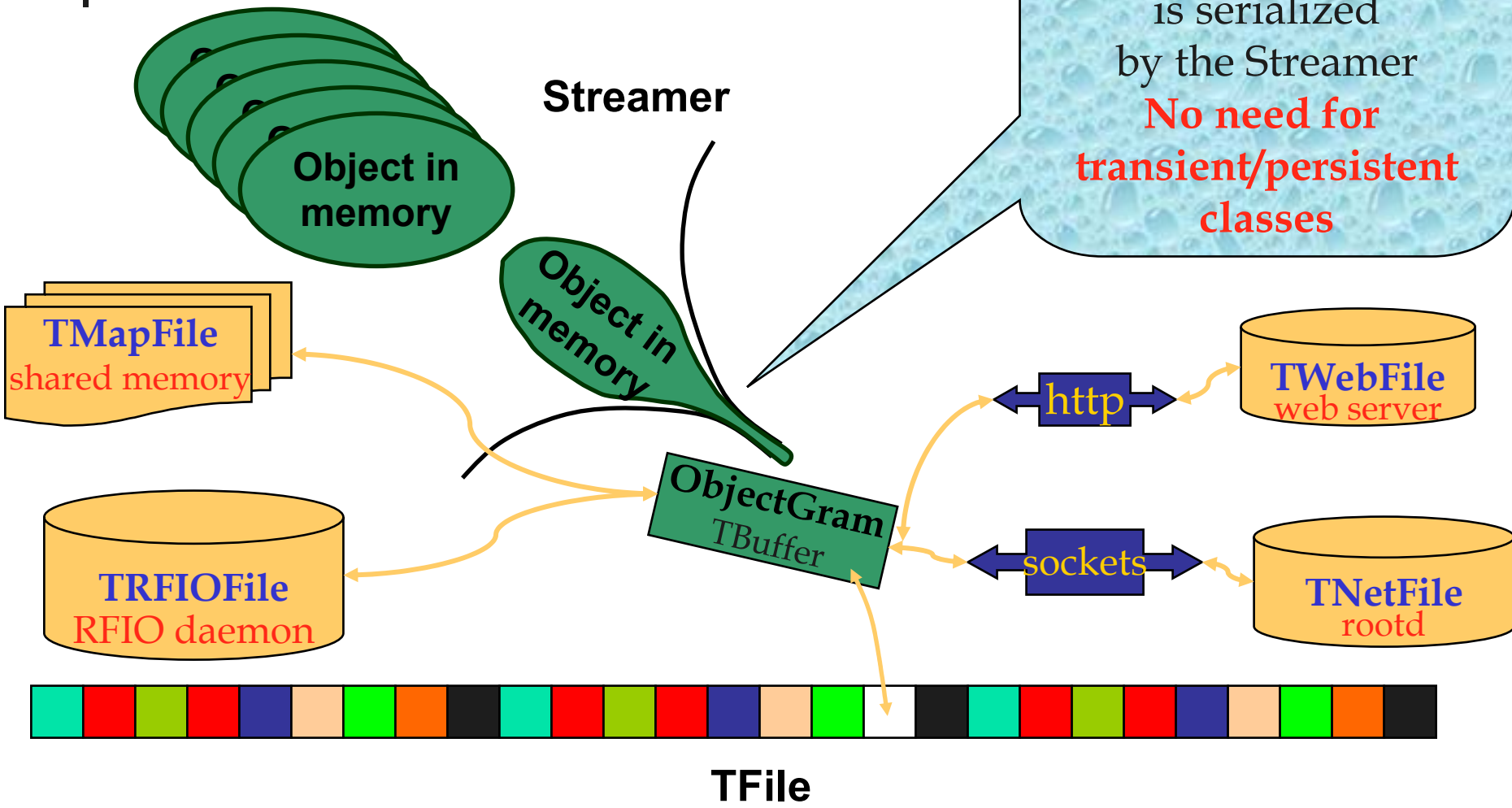
Remote file access via a Web server

Remote file access via the ROOT daemon

Access to a file on a mass store hpps, castor, via RFIO

Streaming Objects

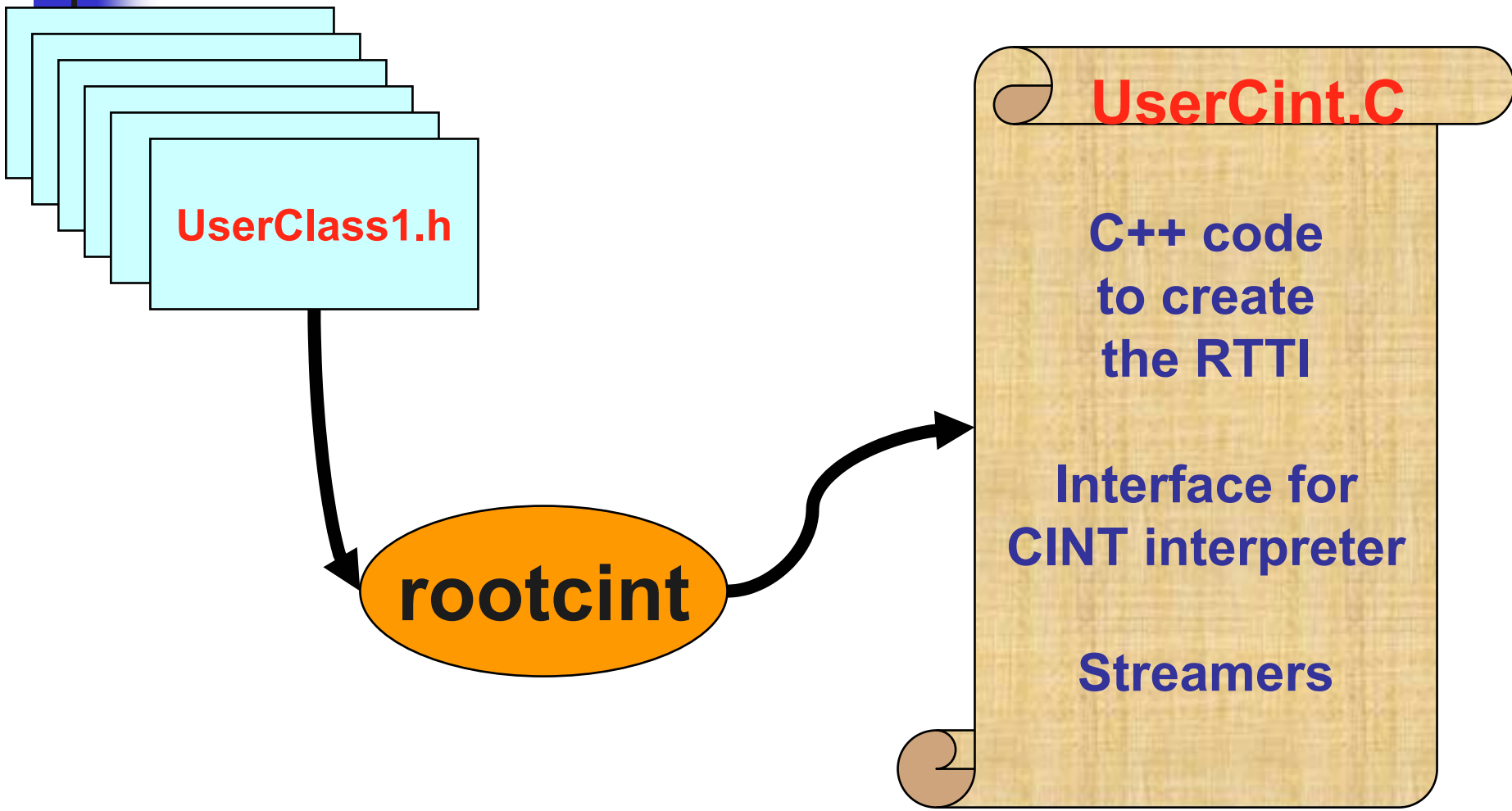
ROOT I/O -- *Sequential/Flat*



The CINT RTTI

- The Run Time Type Information provided by CINT (rootcint) is the brain of Root. rootcint can be used to parse user classes and considerably extend the power of Root.
- RTTI is used by the I/O services
- By definition the interpreter is based on it.
- The GUI object context sensitive menus also.
- Also Browsers, Inspectors and html generator
- also Root utilities to draw class diagrams
- rootcint can be used to parse user classes such that user class functions can be called interactively and code for I/O generated automatically.

Rootcint Preprocessor



Old Streamers in 0.90 (1996)

Evolution illustrated with the ROOT class TAxis

TAxis.h

```
class TAxis : public
TNamed,
public TAttAxis {

private:
   Int_t         fNbins;
   Float_t       fXmin;
   Float_t       fXmax;
   TArrayF       fXbins;
```

rootcint

```
TBuffer b;
object.Streamer(b);
```

Dict.cxx

```
void TAxis::Streamer(TBuffer &b)
{
   if (b.IsReading()) {
      Version_t v = b.ReadVersion();
      TNamed::Streamer(b);
      TAttAxis::Streamer(b);
      b >> fNbins;
      b >> fXmin;
      b >> fXmax;
      fXbins.Streamer(b);
   } else {
      b.WriteVersion(TAxis::IsA());
      TNamed::Streamer(b);
      TAttAxis::Streamer(b);
      b << fNbins;
      b << fXmin;
      b << fXmax;
      fXbins.Streamer(b);
   }
}
```

Old Streamers in 2.25 (1999)

```
class TAxis : public TNamed,
public TAttAxis {

private:
   Int_t         fNbins;
   Float_t       fXmin;
   Float_t       fXmax;
   TArrayF       fXbins;
   Int_t         fFirst;
   Int_t         fLast;
   TString       fTimeFormat;
   Bool_t        fTimeDisplay;
```

rootcint

```
void TAxis::Streamer(TBuffer &b) {
   UInt_t R__s, R__c;
   if (b.IsReading()) {
      Version_t v = b.ReadVersion(&R__s, &R__c);
      TNamed::Streamer(b);
      TAttAxis::Streamer(b);
      b >> fNbins;
      b >> fXmin;
      b >> fXmax;
      fXbins.Streamer(b);
      if (v > 2) {
         b >> fFirst;
         b >> fLast;
      }
      if (v > 3) {
         b >> fTimeDisplay;
         fTimeFormat.Streamer(b);
      } else {
         SetTimeFormat();
      }
      b.CheckByteCount(R__s, R__c, TAxis::IsA());
   } else {
      R__c = b.WriteVersion(TAxis::IsA(), kTRUE);
      TNamed::Streamer(b);
      TAttAxis::Streamer(b);
      b << fNbins;
      b << fXmin;
      b << fXmax;
      fXbins.Streamer(b);
      b << fFirst;
      b << fLast;
      b << fTimeDisplay;
      fTimeFormat.Streamer(b);
      b.SetByteCount(R__c, kTRUE);
   }
}
```

Problems with Old Streamers

- Experience in several large experiments has shown that a system based only on automatic code generation with no support for schema evolution is not a long term solution. A huge maintenance problem.
- In a system with several hundred (thousand) classes and as many users, it is difficult to maintain coherent shared libs to support all possible combinations when accessing collections of old data sets.
- A few attempts (eg in STAR) to support automatic schema evolution seen as a progress, but not sufficient.
- We have seen a rapidly growing request for reading data sets without having the original classes.
- Backward compatibility (reading an old data set with new classes) is a must. Forward compatibility (reading a new data set with old classes) also a must.

The ROOT solution

- Minimize reliance on generated code.
- Exploit the powerful CINT Object Dictionary
- Make the process as automatic as possible and as simple as possible.
- Be as efficient as with the generated code.
- Self-describing data sets.
- Come with a solution that does not prevent the move to another language in the future.
- Back compatibility with the original system.
  - Like upgrading the engine in a running car

Implementing all these features was a non trivial exercise and a lot of work

Thanks to our huge users base for providing many use cases and testing

New Streamers in 3.00

```
class TAxis : public TNamed,
public TAttAxis {

private:
   Int_t         fNbins;
   Double_t      fXmin;
   Double_t      fXmax;
   TArrayD       fXbins;
   Char_t       *fXlabels;     //!
   Int_t         fFirst;
   Int_t         fLast;
   TString       fTimeFormat;
   Bool_t        fTimeDisplay;
   TObject      *fParent;       //!
```

```
void TAxis::Streamer(TBuffer &b)
{
   // Stream an object of class TAxis.

   if (b.IsReading())
      TAxis::Class()->ReadBuffer(b, this);
   else
      TAxis::Class()->WriteBuffer(b,this);
}
```

rootcint

Self-Describing file

```
Root > TFile f("demo1.root");
Root > f.ShowStreamerInfo();

StreamerInfo for class: PSimHit, version=1
  BASE          TObject          offset=  0 type=66 Basic ROOT object
  Local3DPoint  theEntryPoint    offset=  0 type=62 position
  Local3DPoint  theExitPoint     offset=  0 type=62
  float         thePabs          offset=  0 type= 5 momentum
  float         theTof           offset=  0 type= 5 Time Of Flight
  float         theEnergyLoss    offset=  0 type= 5 Energy loss
  int           theParticleType  offset=  0 type= 3
  int           theDetUnitId     offset=  0 type= 3
  unsigned int  theTrackId       offset=  0 type=13

StreamerInfo for class: Point3DBase<float,LocalTag>, version=1
  BASE          PV3DBase<float,PointTag,LocalTag> offset=  0 type= 0

StreamerInfo for class: PV3DBase<float,PointTag,LocalTag>, version=1
  Basic3DVector<float>theVector       offset=  0 type=62

StreamerInfo for class: Basic3DVector<float>, version=1
  float         theX             offset=  0 type= 5
  float         theY             offset=  0 type= 5
  float         theZ             offset=  0 type= 5
```

Self-describing files

- Dictionary for persistent classes written to the file when closing the file.
- ROOT files can be read by foreign readers (eg JavaRoot (Tony Johnson)
- Support for Backward and Forward compatibility
- Files created in 2003 must be readable in 2015
- Classes (data objects) for all objects in a file can be regenerated via TFile::MakeProject

```
Root >TFile f("demo.root");
Root > f.MakeProject("dir","*","new++");
```

Showing classes in a file TFile::ShowStreamerInfo

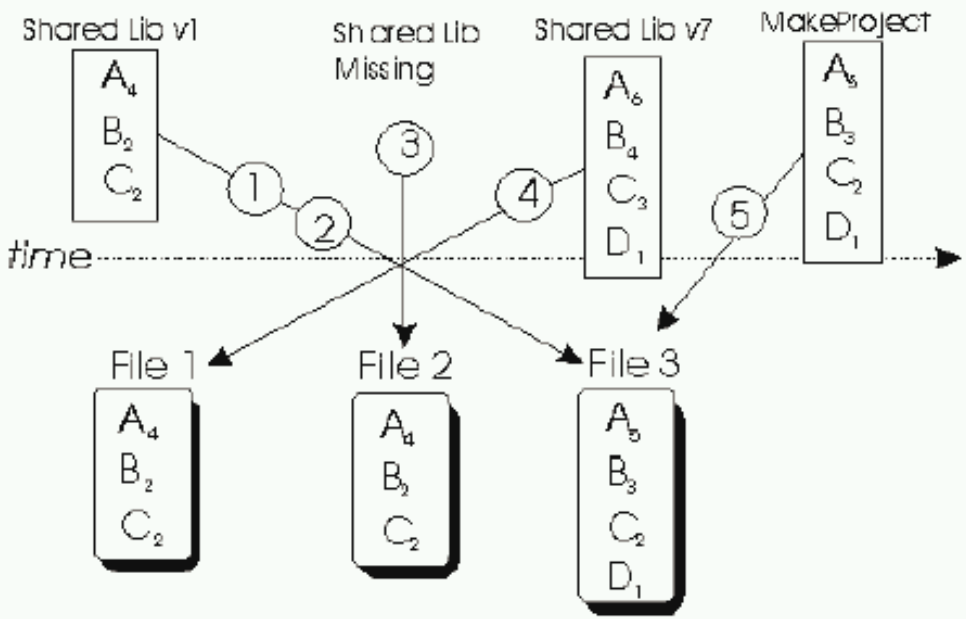
```
Root > f.ShowStreamerInfo()
```

```
StreamerInfo for class: ATLFMuon, version=1
  BASE          TObject         offset=  0 type=66 Basic ROOT object
  BASE          TAtt3D          offset=  0 type= 0 3D attributes
  Int_t         m_KFcode        offset=  0 type= 3 Muon KF-code
  Int_t         m_MCParticle    offset=  0 type= 3 Muon position in MCParticles list
  Int_t         m_KFmother      offset=  0 type= 3 Muon mother KF-code
  Int_t         m_UseFlag       offset=  0 type= 3 Muon energy usage flag (0 for used in clusters)
  Int_t         m_Isolated      offset=  0 type= 3 Muon isolation (1 for isolated)
  Float_t       m_Eta           offset=  0 type= 5 Eta coordinate
  Float_t       m_Phi           offset=  0 type= 5 Phi coordinate
  Float_t       m_PT            offset=  0 type= 5 Transverse energy
  Int_t         m_Trigger       offset=  0 type= 3 Result of trigger

StreamerInfo for class: ATLFElectron, version=1
  BASE          TObject         offset=  0 type=66 Basic ROOT object
  BASE          TAtt3D          offset=  0 type= 0 3D attributes
  Int_t         m_KFcode        offset=  0 type= 3 Electron KF-code
  Int_t         m_MCParticle    offset=  0 type= 3 Electron position in MCParticles list
  Int_t         m_KFmother      offset=  0 type= 3 Electron mother KF-code
  Float_t       m_Eta           offset=  0 type= 5 Eta coordinate
  Float_t       m_Phi           offset=  0 type= 5 Phi coordinate
  Float_t       m_PT            offset=  0 type= 5 Transverse energy

StreamerInfo for class: ATLFPhoton, version=1
  BASE          TObject         offset=  0 type=66 Basic ROOT object
  BASE          TAtt3D          offset=  0 type= 0 3D attributes
  Int_t         m_KFcode        offset=  0 type= 3 Photon KF-code
  Int_t         m_MCParticle    offset=  0 type= 3 Photon position in MCParticles list
  Int_t         m_KFmother      offset=  0 type= 3 Photon mother KF-code
  Float_t       m_Eta           offset=  0 type= 5 Eta coordinate
  Float_t       m_Phi           offset=  0 type= 5 Phi coordinate
  Float_t       m_PT            offset=  0 type= 5 Transverse energy

StreamerInfo for class: ATLFJet, version=1
  BASE          TObject         offset=  0 type=66 Basic ROOT object
  BASE          TAtt3D          offset=  0 type= 0 3D attributes
  Int_t         m_KFcode        offset=  0 type= 3 Jet KF-code
  Int_t         m_Ncells        offset=  0 type= 3 Number of cells used for reconstruction
  Int_t         m_Nparticles    offset=  0 type= 3 Number of particles assigned to jet
  Int_t         m_Part          offset=  0 type= 3 Position in MCParticle list of matching b-quark/c-qu
  Float_t       m_Eta0          offset=  0 type= 5 Eta position of initiator cell
  Float_t       m_Phi0          offset=  0 type= 5 Phi position of initiator cell
  Float_t       m_Eta           offset=  0 type= 5 Eta  of jet bary-center
  Float_t       m_Phi           offset=  0 type= 5 Phi  of jet bary-center
  Float_t       m_PT            offset=  0 type= 5 Transverse momentum of jet
```

ROOT conventions/extentions

```
class TGeoNode : public TNamed, public TGeoAtt
{
protected:
   TGeoVolume     *fVolume;     // volume associated with this
   TGeoVolume     *fMother;     // mother volume
   Int_t           fNumber;     // copy number
   Int_t           fNovlp;      // number of overlaps
   Int_t          *fOverlaps;   //[fNovlp] list of indices for overlapping


class TGeoManager : public TNamed
{
private :
   Double_t              fStep;             //! step to be done from current point
   Double_t              fSafety;           //! safety radius from current point
   Double_t              fPhimin;           // lowest range for phi cut
   Double_t              fPhimax;           // highest range for phi cut
   TGeoNodeCache        *fCache;            //! cache for physical nodes
   TVirtualGeoPainter   *fPainter;          //! current painter
   TList                *fMatrices;         //-> list of local transformations
   TObjArray            *fNodes;            //-> current branch of nodes
   TGeoVolume           *fMasterVolume;     // master volume
   TGeoVolume           *fCurrentVolume;    //! current volume
   TGeoNode             *fCurrentNode;      //! current node
```

Automatic Schema Evolution



1) An old version of a shared library and a file with new class definitions. This can be the case when someone has not updated the library and is reading a new file.

2) Reading a file with a shared library that is missing a class definition (i.e. missing class D).

3) Reading a file without any class definitions. This can be the case where the class definition is lost, or unavailable.

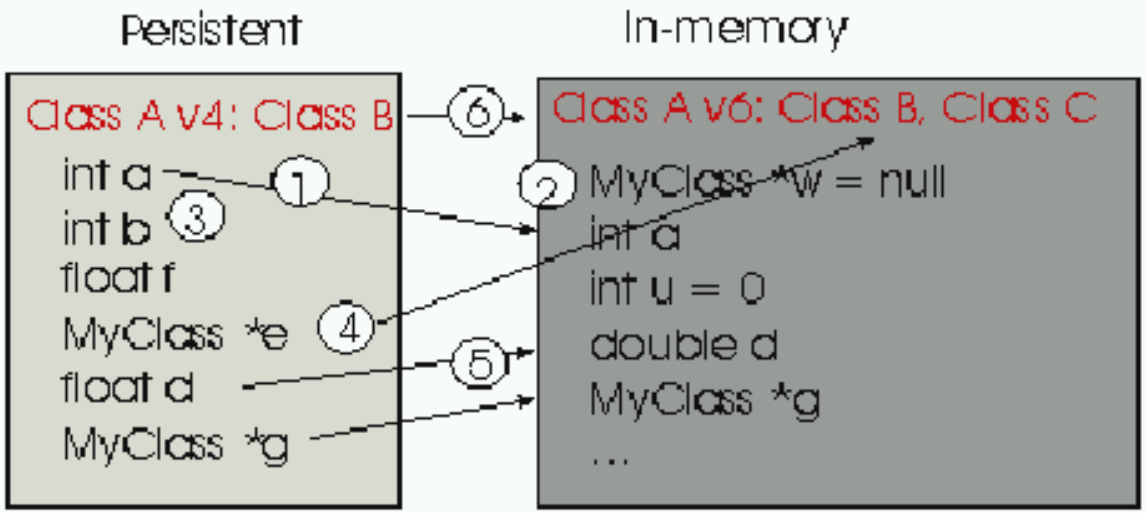
4) The current version of a shared library and an old file with old class versions (backward compatibility). This is often the case when reading old data.

5) Reading a file with a shared library built with `MakeProject`. This is the case when someone has already read the data without a shared library and has used ROOT's `MakeProject` feature to reconstruct the class definitions and shared library (`MakeProject` is explained in detail later on).

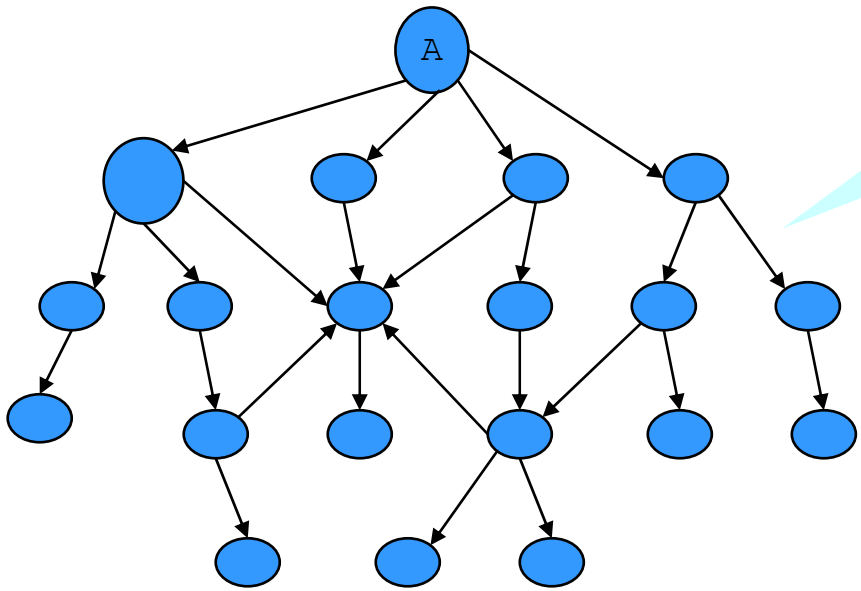
Auto Schema Evolution (2)

In case of a mismatch between the in-memory version and the persistent version of a class, ROOT maps the persistent one to the one in memory. This allows you to change the class definition at will, for example:

1) Change the order of data members in the class.
2) Add new data members. By default the value of the missing member will be 0 or in case of an object it will be set to null.
3) Remove data members.
4) Move a data member to a base class or vice –versa.
5) Change the type of a member if it is a simple type or a pointer to a simple type. If a loss of precision occurs, a warning is given.
6) Add or remove a base class



Normal Streaming mode
References using C++ pointers



Only one copy of each object in the graph saved to buffer

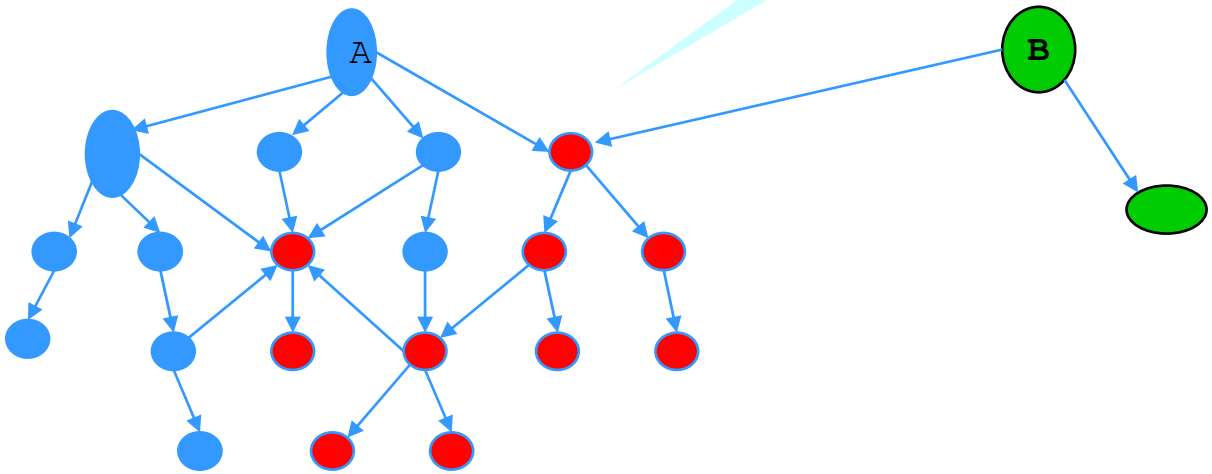
```
TBuffer b;
A.Streamer(b)
```

Normal Streaming mode

References using C++ pointers

```
TBuffer b1;
A.Streamer(b1)
TBuffer b2;
B.Streamer(b2)
```

Objects in red
are in b1 and b2

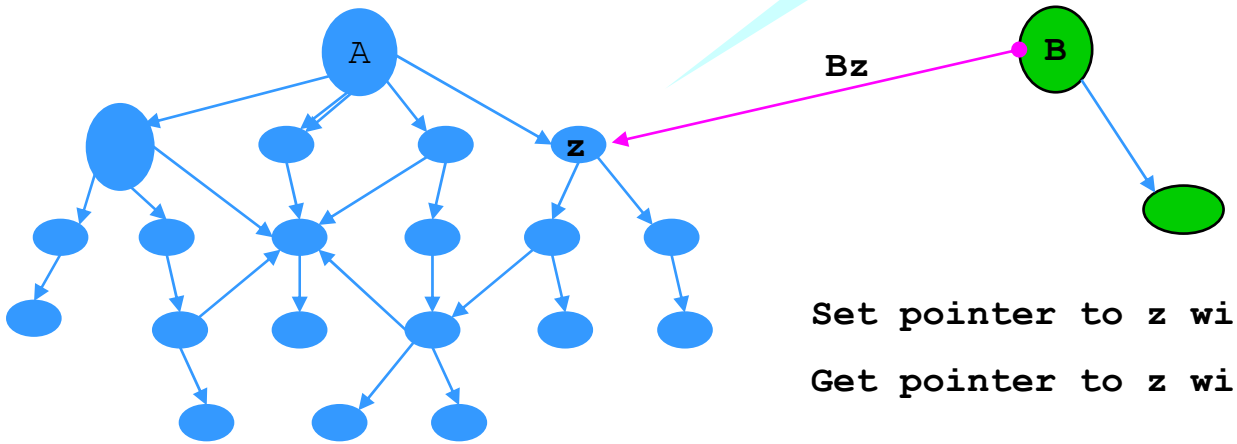


Normal Streaming mode

References using TRef pointers

```
TBuffer b1;
A.Streamer(b1)
TBuffer b2;
B.Streamer(b2)
```

Objects in blue are only in b1



C++ pointer

TRef

Set pointer to z with: `TRef Bz = z;`

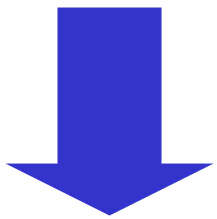
Get pointer to z with: `z = Bz.GetObject()`

TRef example: Event.h

```
class Event : public TObject {

private:
   char            fType[20];          //event type
   char           *fEventName;         //run+event number in character format
   int             fNtrack;            //Number of tracks
   int             fNseg;              //Number of track segments
   int             fNvertex;
   int             fMeasures[10];
   float           fMatrix[4][4];
   float          *fClosestDistance;   //[fNvertex]
   EventHeader     fEvtHdr;
   TClonesArray   *fTracks;            //->array with all tracks
   TRefArray      *fHighPt;            //array of High Pt tracks only
   TRefArray      *fMuons;             //array of Muon tracks only
   TRef            fLastTrack;         //reference pointer to last track
   TRef            fWebHistogram;      //EXEC:GetWebHistogram
   TH1F           *fH;                 //->


public:
   ...
   TH1F          *GetHistogram() const {return fH;}
   TH1F          *GetWebHistogram(Bool_t reload=kFALSE)  const {
                    return (TH1F*)fWebHistogram.GetObject(reload);}
```

TRef, TRefArray

- Designed as light weight entities
- Assume large number of TRefs per event
- Very fast dereferencing (direct access tables)
- Cannot (not designed for) find an object in a file



TLongRef, TUUID classes proposed for references with load on demand

TRef/TRefArray goals

- TRef is perfect for referencing objects like hits, clusters, tracks that may be > 10000.
- You would not like to have the size of a TRef bigger than the size of its referenced object !
- A TRef occupies in average 2.5 bytes in the file
- There is no point in providing load on demand for one single hit, cluster or track.

Setting a TRef pointer

- Assuming obj = pointer to a TObject*
- TRef ref = obj;
- TRef is itself a TObject

```
Class TObject {
unsigned int fBits;
unsigned int fUniqueID;
```

```
Class TRef : public TObject {
TProcessID *fPID; //!pointer to process id
```

- If the obj::kIsReferenced bit is not yet set, the obj::fUniqueID is set to the CurrentNumber+1 and obj::kIsReferenced is set to 1.
- Its fUniqueID is set to obj::fUniqueID
- CurrentNumber is managed by TProcessID
- fObjs[CurrentNumber] is set to obj in fPID

Writing Referenced objects

- A referenced object is written by obj->Streamer
- This Streamer at some point calls its TObject::Streamer
- In TObject::Streamer, if the kIsReferenced bit is set in fBits, the following additional info is also written:
- pid (4 bytes) = TProcessID of current process
- Average overhead in memory = 0 bytes
- Average overhead on disk (no comp) = 2 bytes
- Average overhead on disk (comp) = 2.4 bytes (0.4)

A Referenced object may be written multiple times in the same file as the TRef or in other files

Writing TRefs to a buffer

- A TRef is written by TRef::Streamer
- Writes uid(4 bytes) + pid(4 bytes)
- uid = object unique identifier
  - uid = ref::fUniqueID = obj::fUniqueID = current object nr
- pid = Process identifier
  - Each process has a unique pid (TProcessID)
  - A file contains the pids of all processes that have written objects to it.

- TRef memory size = 16 bytes

- TRef size on disk (no comp) = 8 bytes

- TRef size on disk (comp) = 2.4 bytes

Reading Referenced objects

- A referenced object is read by obj->Streamer
- This Streamer at some point calls its TObject::Streamer
- In TObject::Streamer, if the kIsReferenced bit is set in fBits, the following additional info is also read:
- pid (4 bytes) = TProcessID of current process
- In TProcessID::fObjs fObjs[fUniqueID] = obj
- When obj is deleted, fObjs[fUniqueID] = 0;

Reading TRefs from a buffer

- A TRef object is read by TRef::Streamer
- The pair uid,pid is read
- the fUniqueID of TRef is set to uid
- The transient pointer fPID is set to the TProcessID corresponding to pid
- The bit 1 of fBits is set

A TRef use case

Session 1

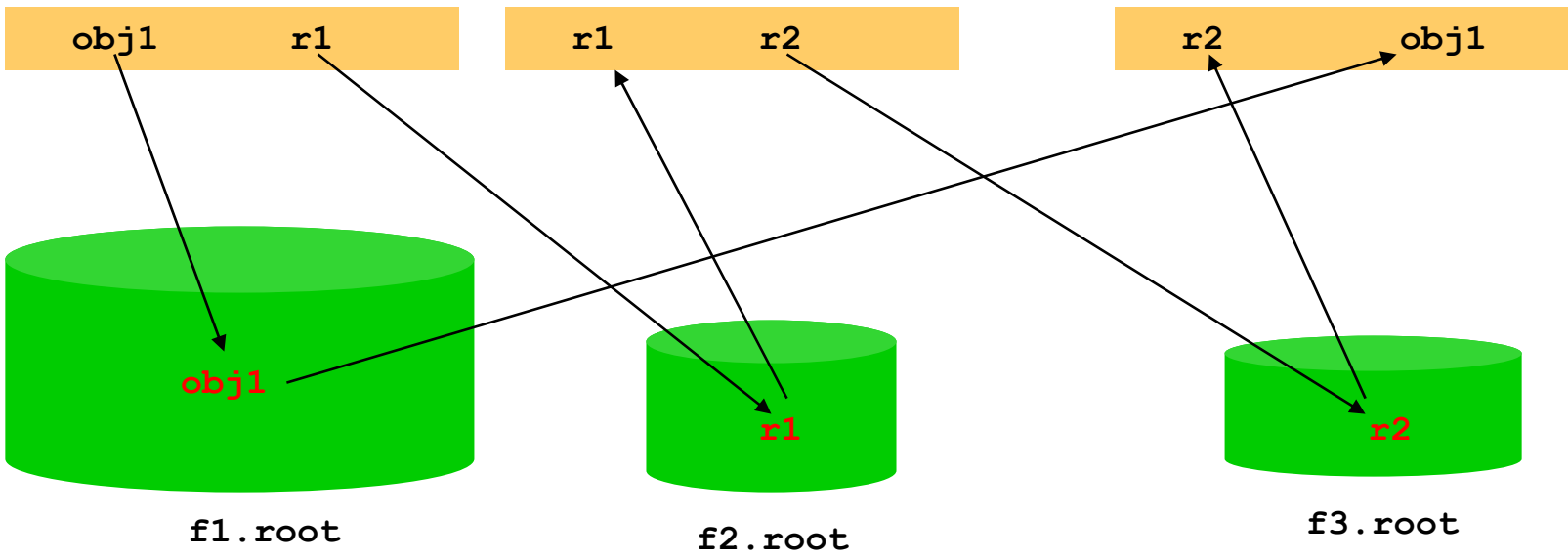
```
Myclass *obj1;
TRef r1 = obj1;
TFile f1("f1.root","new");
obj1->Write("obj1");
TFile f2("f2.root","new");
r1.Write("r1");
```

Session 2

```
TFile f2("f2.root");
TRef *r1;
r1 = (TRef*)f2.Get("r1");
TFile f3("f3.root","new");
TRef r2 = *r1;
r2.Write("r2")
```

Session 3

```
TFile f3("f3.root");
TRef *r2;
r2 = (TRef*)f3.Get("r2");
r2->GetObject();    = 0
TFile f1("f1.root");
Myclass *obj1;
obj1 = (Myclass*)f1.Get("obj1");
r2->GetObject();   --> obj1
```



The ROOT System

A Data Access & Analysis Framework

Trees

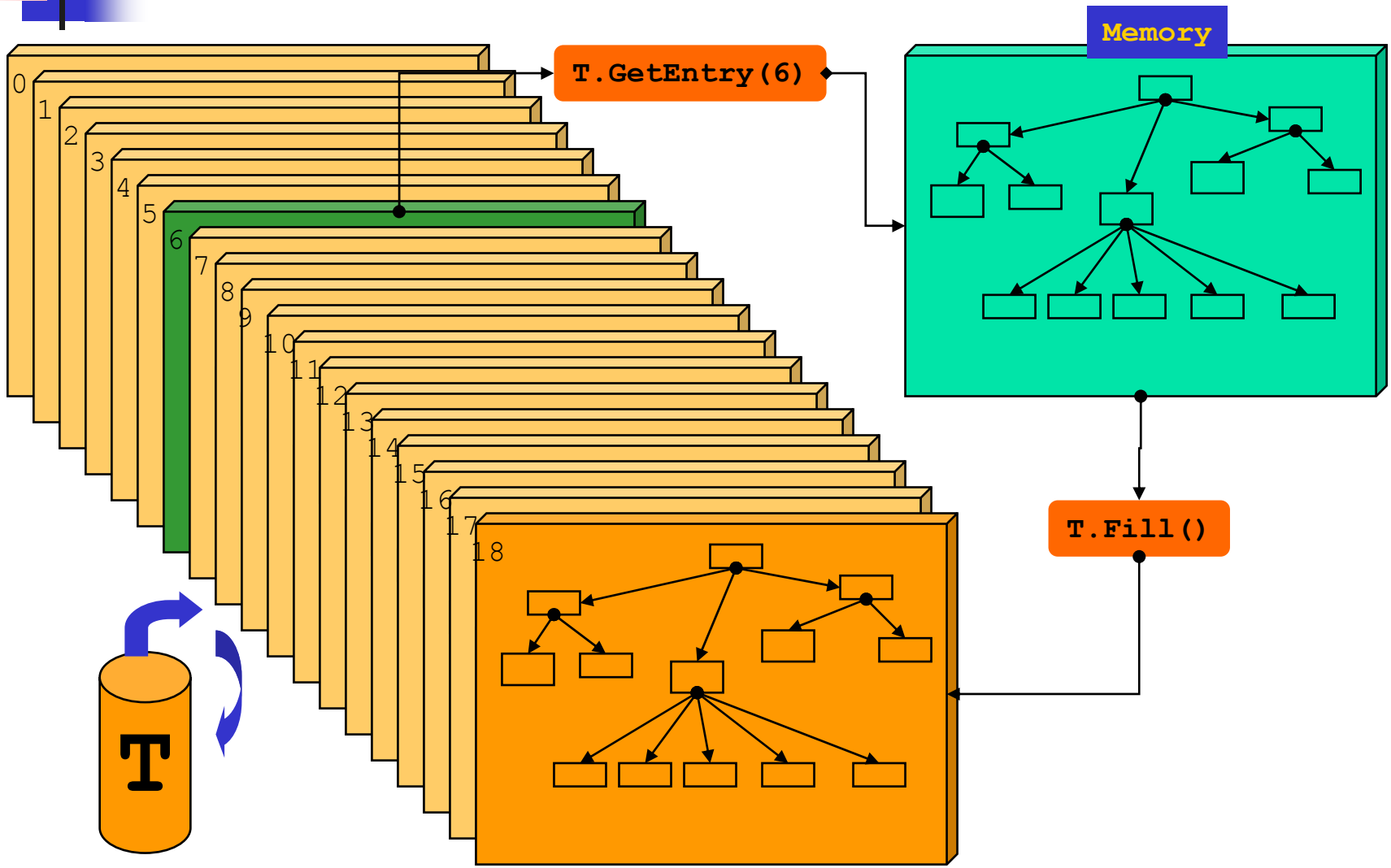
4-5-6 February 2003

René Brun/EP

http://root.cern.ch

Memory <--> Tree

Each Node is a branch in the Tree

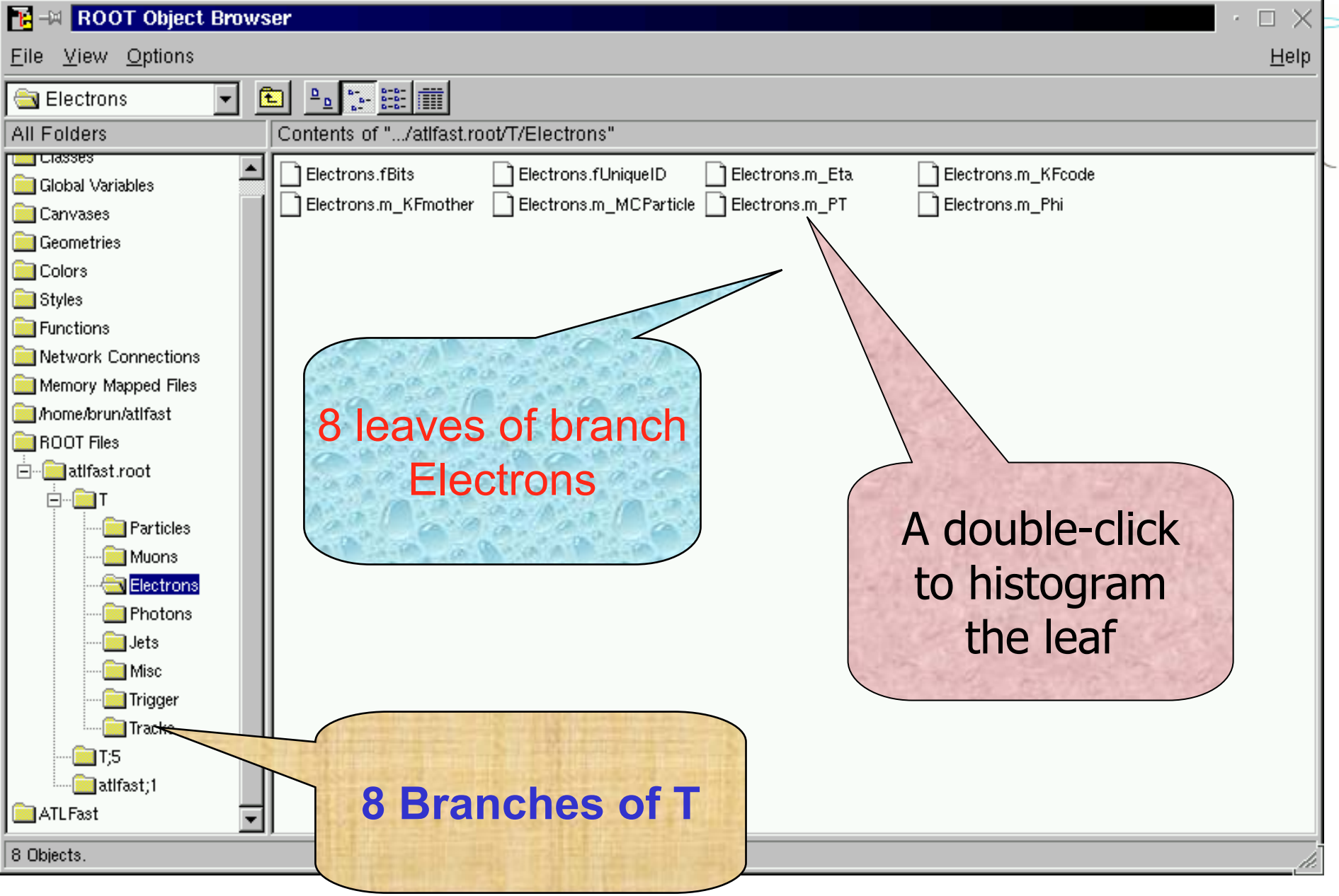


Tree Creation Example

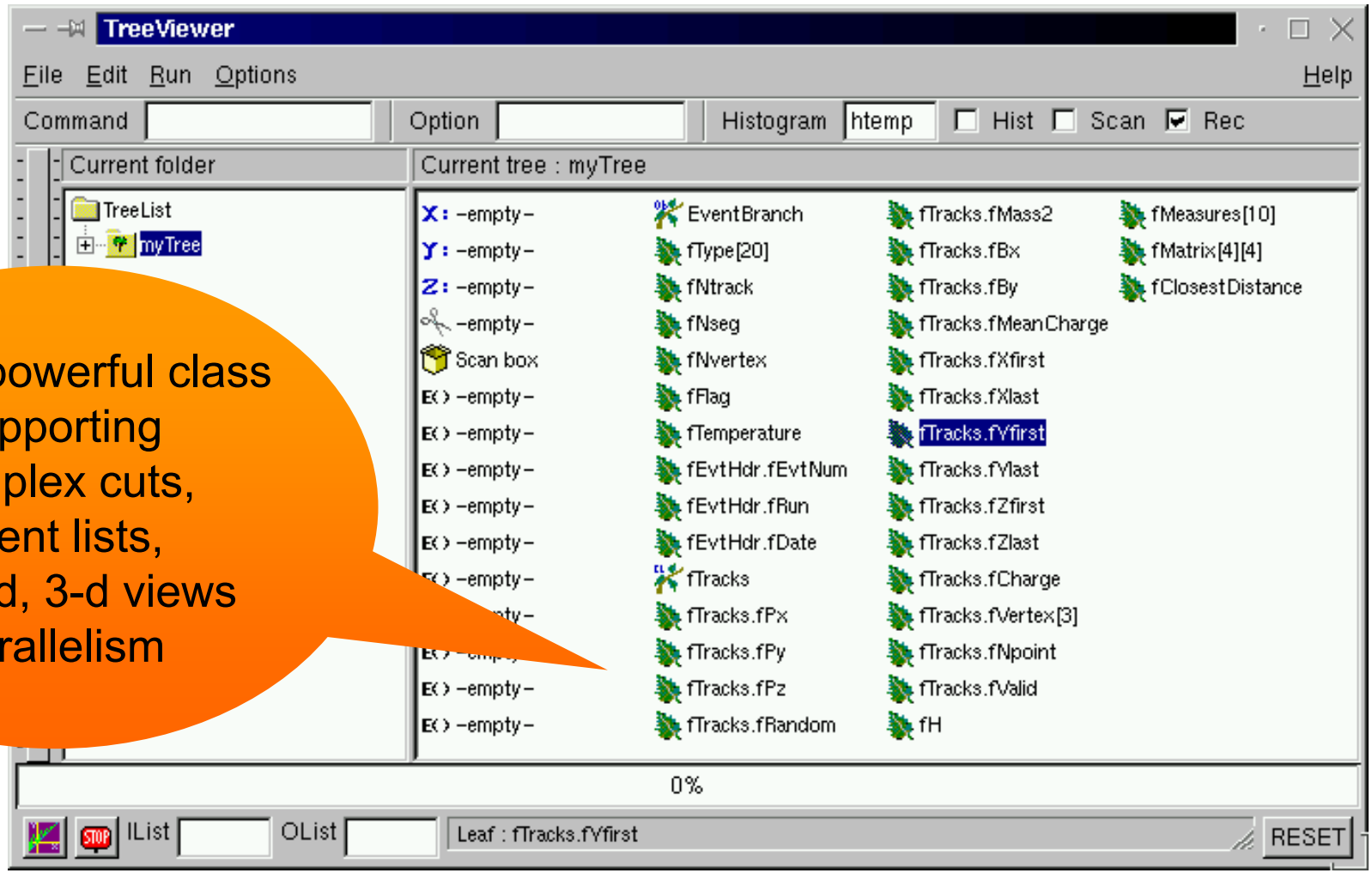
```
class TEvent: public TObject
   THeader          *fHeader;     //Event Header object
   TObjArray        *fVertex;     //List of vertice
   TClonesArray     *fTracks;     //List of track
   TTOF             *fTOF;        //Time of Fl
   TCalor           *fCalor;      //Calorime
```

A few lines of code to create a Tree for structures that may be very complex

```
main()
TEvent *event;
TFile dst("demo.root","NEW");
TTree tree("T","Example of Tree");
Int_t split = 1;  // or split=0
tree.Branch("event", "TEvent", &event, split);

for (int ev =0; ev<10000;ev++) {
   event = new TEvent(ev);
   tree.Fill();
   delete event;
}
dst.Close();
```

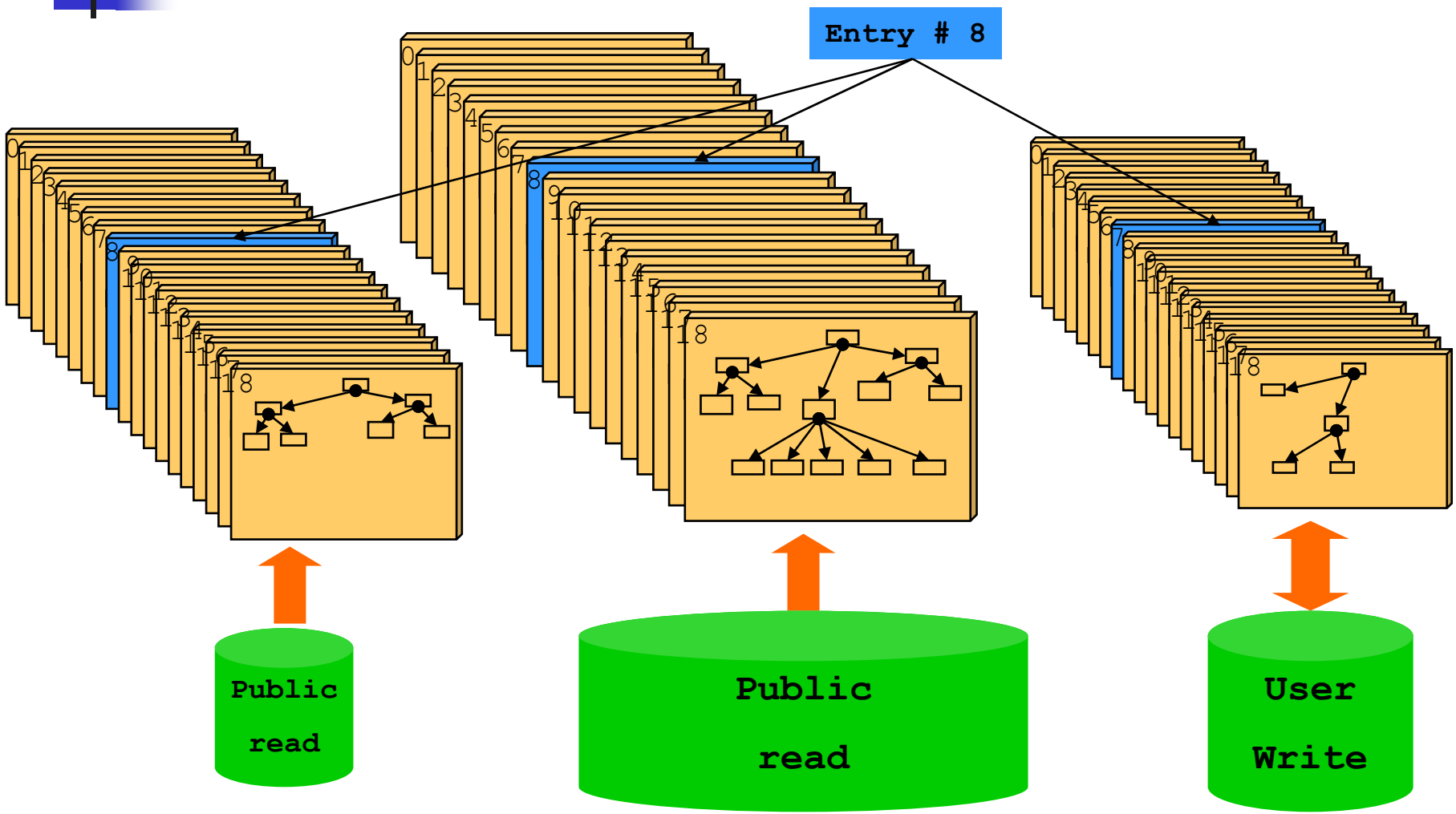

ROOT Object Browser
File View Options Help
Electrons
All Folders
Contents of ".../atlfast.root/T/Electrons"
Global Variables
Canvases
Geometries
Colors
Styles
Functions
Network Connections
Memory Mapped Files
/home/brun/atlfast
ROOT Files
atlfast.root
T
Particles
Muons
Electrons
Photons
Jets
Misc
Trigger
Tracks
T;5
atlfast;1
ATLFast
Electrons.fBits
Electrons.fUniqueID
Electrons.m_Eta
Electrons.m_KFcode
Electrons.m_KFmother
Electrons.m_MCParticle
Electrons.m_PT
Electrons.m_Phi
8 Objects.
8 leaves of branch Electrons
A double-click to histogram the leaf
8 Branches of T

The Tree Viewer & Analyzer

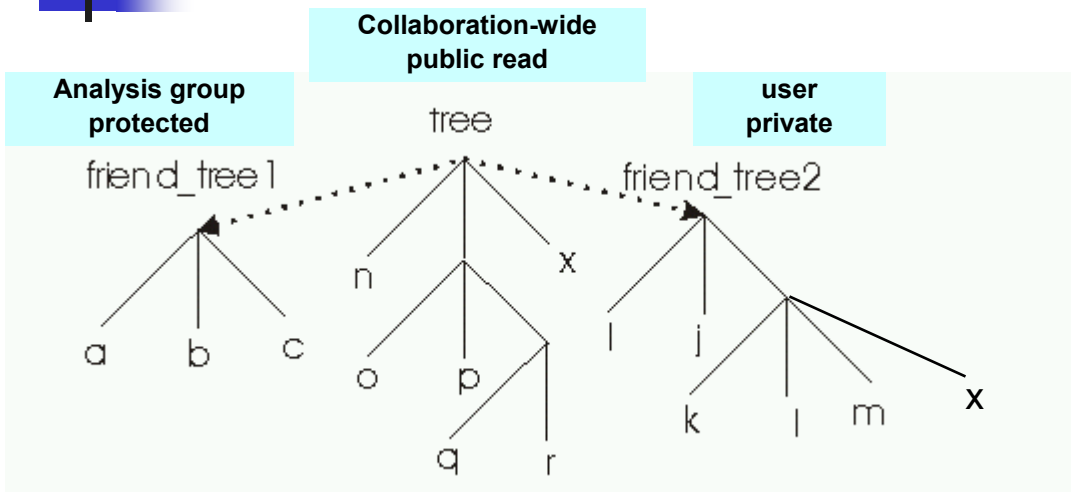


A very powerful class supporting complex cuts, event lists, 1-d,2-d, 3-d views parallelism

Tree Friends



Tree Friends



Processing time independent of the number of friends unlike table joins in RDBMS

```
Root > TFile f1("tree1.root");
Root > tree.AddFriend("tree2","tree2.root")
Root > tree.AddFriend("tree3","tree3.root");
Root > tree.Draw("x:a","k<c");
Root > tree.Draw("x:tree2.x","sqrt(p)<b");
```

Chains of Trees

- A TChain is a collection of Trees.
- Same semantics for TChains and TTrees
  - root > .x h1chain.C
  - root > chain.Process("h1analysis.C")

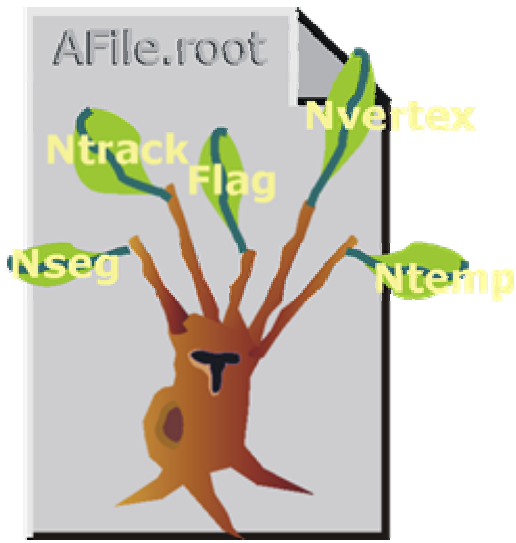
```
{
  //creates a TChain to be used by the h1analysis.C class
  //the symbol H1 must point to a directory where the H1 data sets
  //have been installed

  TChain chain("h42");
    chain.Add("$H1/dstarmb.root");
    chain.Add("$H1/dstarp1a.root");
    chain.Add("$H1/dstarp1b.root");
    chain.Add("$H1/dstarp2.root");
}
```

Ntuples and Trees

- Ntuples
  - support PAW-like ntuples and functions
  - PAW ntuples/histograms can be imported
- Trees
  - Extension of Ntuples for Objects
  - Collection of branches (branch has its own buffer)
  - Can input partial Event
  - Can have several Trees in parallel
- Chains = collections of Trees

Why Trees ?

- Any object deriving from TObject can be written to a file with an associated key with object.Write()
- However each key has an overhead in the directory structure in memory (about 60 bytes). Object.Write is very convenient for objects like histograms, detector objects, calibrations, but not for event objects.



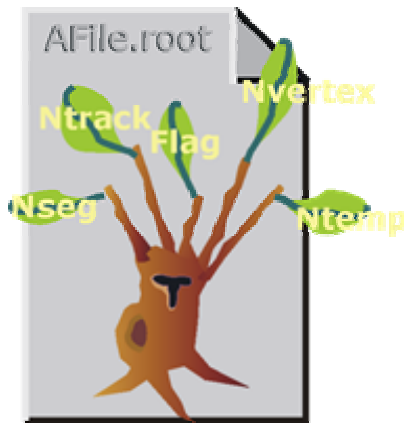
Why Trees ?

- Trees have been designed to support very large collections of objects. The overhead in memory is in general less than 4 bytes per entry.
- Trees allow direct and random access to any entry (sequential access is the best)
- Trees have branches and leaves. One can read a subset of all branches. This can speed-up considerably the data analysis processes.

Adding a Branch

Many Branch constructors Only a few shown here

- Branch name
- Class name
- Address of the pointer to the Object (descendant of TObject)
- Buffer size (default = 32,000)
- Split level (default = 1)

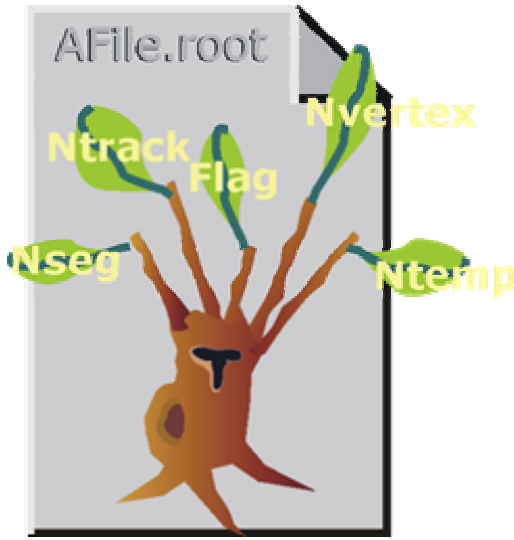



```
Event *event = new  Event();
myTree->Branch("eBranch","Event",&event,64000,1);
```

Splitting a Branch

Setting the split level (default = 1)

Split level = 0



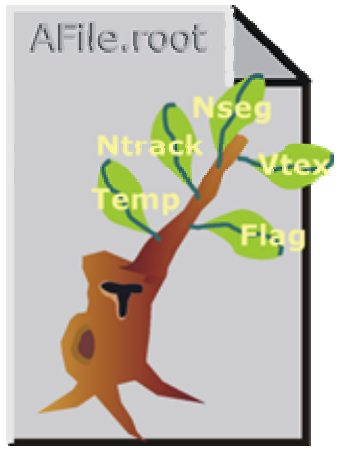
Split level = 1

Example:

```
tree->Branch("EvBr","Event",&ev,64000,0);
```

Adding Branches with a List of Variables

- Branch name
- Address: the address of the first item of a structure.
- Leaflist: all variable names and types
- Order the variables according to their size



Example

```
TBranch *b = tree->Branch ("Ev_Branch",&event,
      "ntrack/I:nseg:nvtex:flag/i:temp/F");
```

Why Trees ?

- PAW ntuples are a special case of Trees.
- Trees are designed to work with complex event objects.
- High level functions like TTree::Draw loop on all entries with selection expressions.
- Trees can be browsed via TBrowser
- Trees can be analized via TTreeViewer

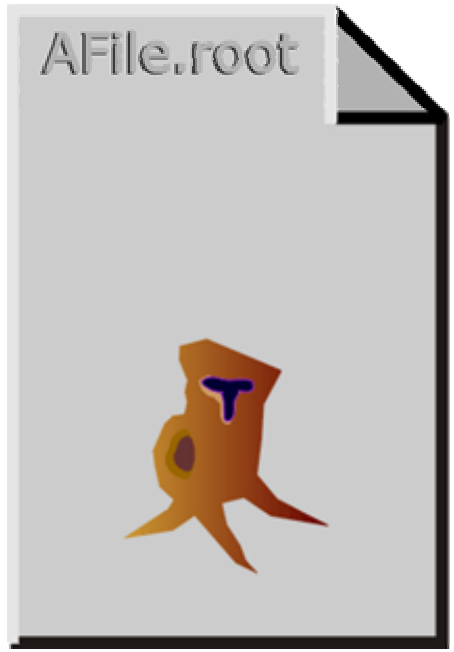
The PROOF system is designed to process chains of Trees in parallel in a GRID environment

Create a TTree Object

A tree is a list of branches.

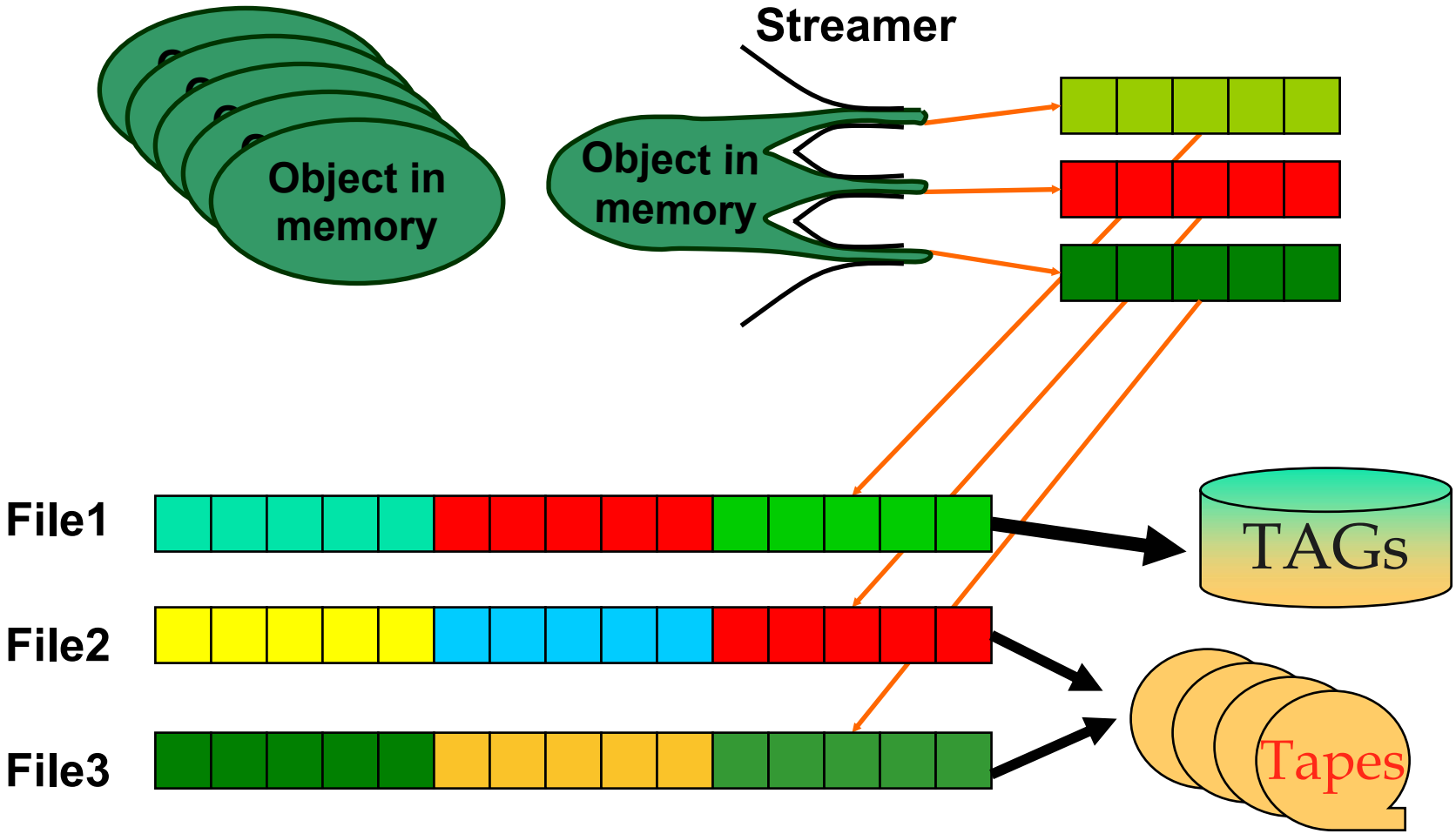
The TTree Constructor:

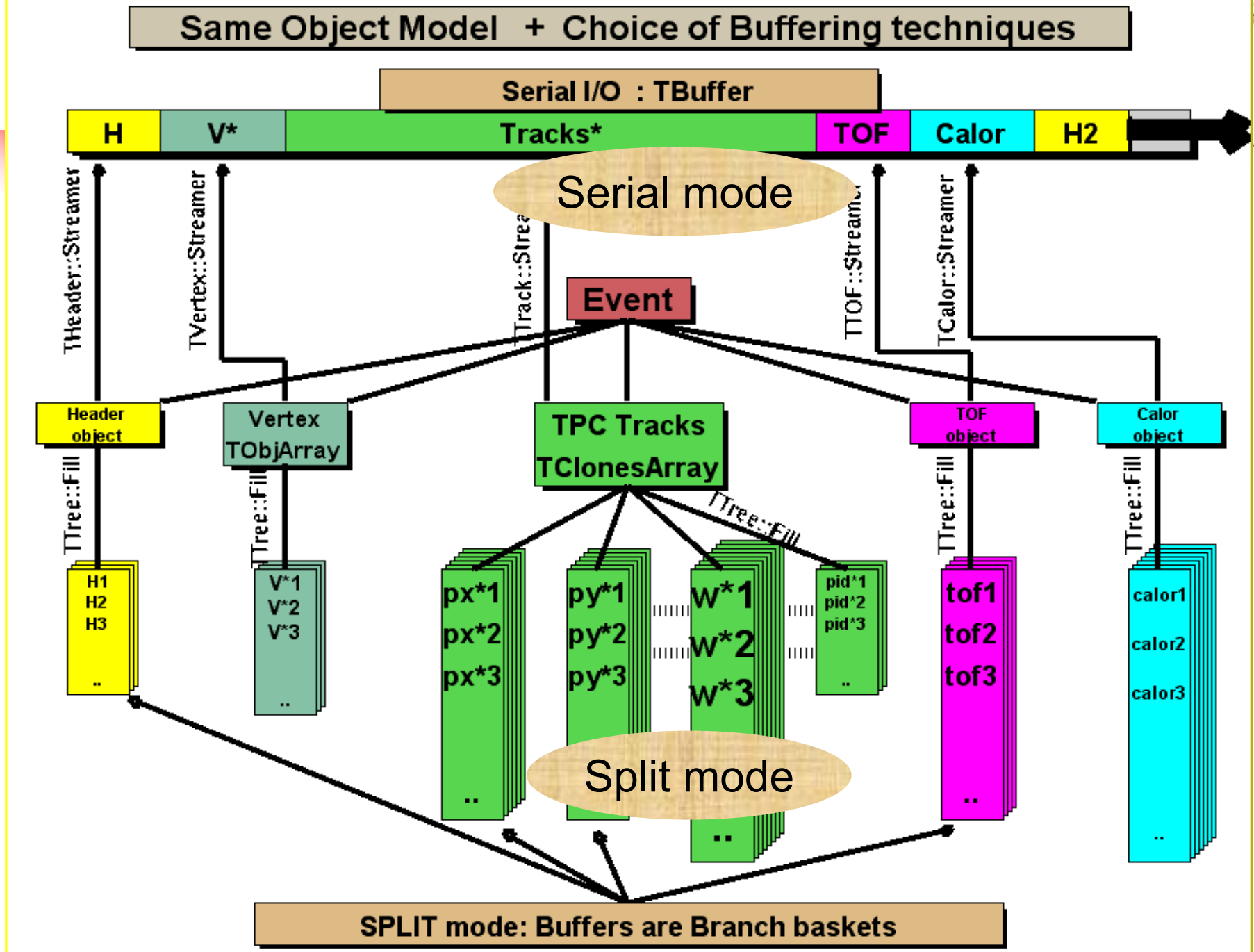
- Tree Name (e.g. "myTree")
- Tree Title

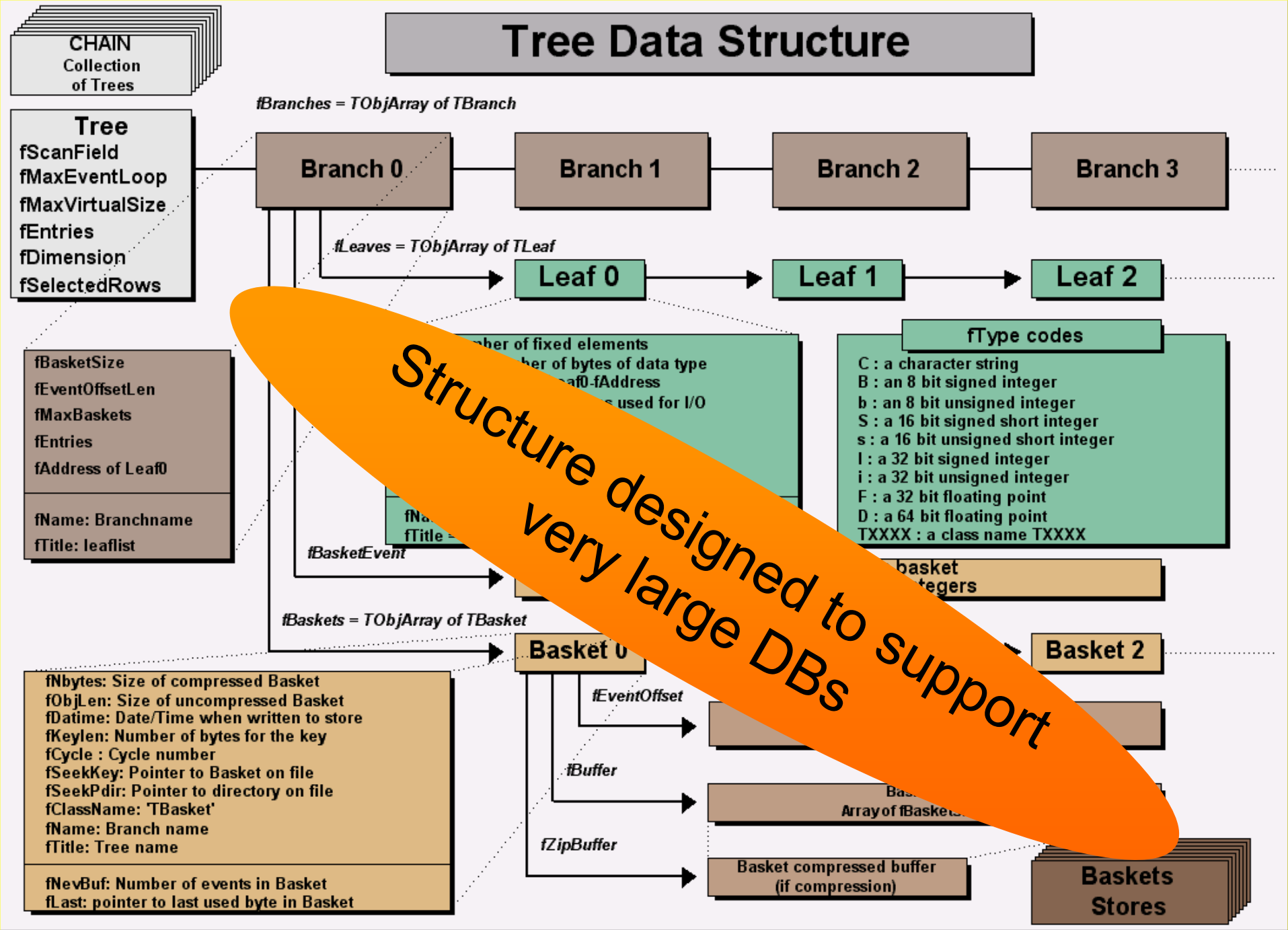



```
TTree *tree = new TTree("T","A ROOT tree");
```

ROOT I/O - *Split - multifile*




Same Object Model + Choice of Buffering techniques
Serial I/O : TBuffer
H
V*
Tracks*
TOF
Calor
H2
Serial mode
THeader::Streamer
TVertex::Streamer
TTrack::Strea
TTOF::Streamer
TCalor::Streamer
Event
Header object
Vertex TObjArray
TPC Tracks TClonesArray
TOF object
Calor object
TTree::Fill
H1
H2
H3
V*1
V*2
V*3
px*1
px*2
px*3
py*1
py*2
py*3
w*1
w*2
w*3
pid*1
pid*2
pid*3
tof1
tof2
tof3
calor1
calor2
calor3
Split mode
SPLIT mode: Buffers are Branch baskets
/user/brun/root/slides/chep97/iomode.C


Tree Data Structure
CHAIN
Collection
of Trees
Tree
fScanField
fMaxEventLoop
fMaxVirtualSize
fEntries
fDimension
fSelectedRows
fBranches = TObjArray of TBranch
Branch 0
Branch 1
Branch 2
Branch 3
fLeaves = TObjArray of TLeaf
Leaf 0
Leaf 1
Leaf 2
fBasketSize
fEventOffsetLen
fMaxBaskets
fEntries
fAddress of Leaf0
fName: Branchname
fTitle: leaflist
fType codes
C : a character string
B : an 8 bit signed integer
b : an 8 bit unsigned integer
S : a 16 bit signed short integer
s : a 16 bit unsigned short integer
I : a 32 bit signed integer
i : a 32 bit unsigned integer
F : a 32 bit floating point
D : a 64 bit floating point
TXXXX : a class name TXXXX
fBasketEvent
fBaskets = TObjArray of TBasket
Basket 2
fNbytes: Size of compressed Basket
fObjLen: Size of uncompressed Basket
fDatime: Date/Time when written to store
fKeylen: Number of bytes for the key
fCycle : Cycle number
fSeekKey: Pointer to Basket on file
fSeekPdir: Pointer to directory on file
fClassName: 'TBasket'
fName: Branch name
fTitle: Tree name
fNevBuf: Number of events in Basket
fLast: pointer to last used byte in Basket
fEventOffset
fBuffer
fZipBuffer
Basket compressed buffer
(if compression)
Baskets
Stores
Structure designed to support
very large DBs

The Event class

```
class Event : public TObject {

private:
   char            fType[20];          //event type
   Int_t           fNtrack;            //Number of tracks
   Int_t           fNseg;              //Number of track segments
   Int_t           fNvertex;
   UInt_t          fFlag;
   Float_t         fTemperature;
   Int_t           fMeasures[10];
   Float_t         fMatrix[4][4];
   Float_t        *fClosestDistance;   //[fNvertex]
   EventHeader     fEvtHdr;
   TClonesArray   *fTracks;            //->array with all tracks
   TRefArray      *fHighPt;            //array of High Pt tracks only
   TRefArray      *fMuons;             //array of Muon tracks only
   TRef            fLastTrack;         //reference pointer to last track
   TH1F           *fH;                 //->
```

```
class EventHeader {

private:
   Int_t    fEvtNum;
   Int_t    fRun;
   Int_t    fDate;
```

See $ROOTSYS/test/Event.h

The Track class

```
class Track : public TObject {

private:
   Float_t      fPx;           //X component of the momentum
   Float_t      fPy;           //Y component of the momentum
   Float_t      fPz;           //Z component of the momentum
   Float_t      fRandom;       //A random track quantity
   Float_t      fMass2;        //The mass square of this particle
   Float_t      fBx;           //X intercept at the vertex
   Float_t      fBy;           //Y intercept at the vertex
   Float_t      fMeanCharge;   //Mean charge deposition of all hits
   Float_t      fXfirst;       //X coordinate of the first point
   Float_t      fXlast;        //X coordinate of the last point
   Float_t      fYfirst;       //Y coordinate of the first point
   Float_t      fYlast;        //Y coordinate of the last point
   Float_t      fZfirst;       //Z coordinate of the first point
   Float_t      fZlast;        //Z coordinate of the last point
   Float_t      fCharge;       //Charge of this track
   Float_t      fVertex[3];    //Track vertex position
   Int_t        fNpoint;       //Number of points for this track
   Short_t      fValid;        //Validity criterion
```

Event Builder

```
void Event::Build(Int_t ev, Int_ntrack, Float_t ptmin) {

  Clear();
  ……..
  for (Int_t t = 0; t < ntrack; t++) AddTrack(random,ptmin);
}

Track *Event::AddTrack(Float_t random, Float_t ptmin)
{
   // Add a new track to the list of tracks for this event.
   // To avoid calling the very time consuming operator new for each track,
   // the standard but not well know C++ operator "new with placement"
   // is called. If tracks[i] is 0, a new Track object will be created
   // otherwise the previous Track[i] will be overwritten.

   TClonesArray &tracks = *fTracks;
   Track *track = new(tracks[fNtrack++]) Track(random);
   //Save reference to last Track in the collection of Tracks
   fLastTrack = track;
   //Save reference in fHighPt if track is a high Pt track
   if (track->GetPt() > ptmin)    fHighPt->Add(track);
   //Save reference in fMuons if track is a muon candidate
   if (track->GetMass2() < 0.11) fMuons->Add(track);
   return track;
}
```

Tree example Event (write)

```
void demoe(int nevents) {
   //load shared lib with the Event class
   gSystem->Load("$ROOTSYS/test/libEvent");

   //create a new ROOT file
   TFile f("demoe.root","new");

   //Create a ROOT Tree with one single top level branch
   int split = 99;  //try also split=1 and split=0
   int bufsize = 16000;
   Event *event = new Event;
   TTree T("T","Event demo tree");
   T.Branch("event","Event",&event,bufsize,split);

   //Build Event in a loop and fill the Tree
   for (int i=0;i<nevents;i++) {
      event->Build(i);
      T.Fill();
   }

   T.Print();  //Print Tree statistics
   T.Write();  //Write Tree header to the file
}
```

All the examples can be executed with CINT or the compiler

```
root > .x demoe.C
root > .x demoe.C++
```

Tree example Event (read 1)

```
void demoer() {
   //load shared lib with the Event class
   gSystem->Load("$ROOTSYS/test/libEvent");

   //connect ROOT file
   TFile *f = new TFile("demoe.root");

   //Read Tree header and set top branch address
   Event *event = 0;
   TTree *T = (TTree*)f->Get("T");
   T->SetBranchAddress("event",&event);

   //Loop on events and fill an histogram
   TH1F *h = new TH1F("hntrack","Number of tracks",100,580,620);
   int nevents = (int)T->GetEntries();
   for (int i=0;i<nevents;i++) {
      T->GetEntry(i);
      h->Fill(event->GetNtrack());
   }

   h->Draw();
}
```

Rebuild the full event in memory

Tree example Event (read 2)

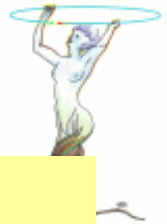
```
void demoer2() {
   //load shared lib with the Event class
   gSystem->Load("$ROOTSYS/test/libEvent");

   //connect ROOT file
   TFile *f = new TFile("demoe.root");

   //Read Tree header and set top branch address
   Event *event = 0;
   TTree *T = (TTree*)f->Get("T");
   T->SetBranchAddress("event",&event);
   Tbranch *bntrack = T->GetBranch("fNtrack");

   //Loop on events and fill an histogram
   TH1F *h = new TH1F("hntrack","Number of tracks",100,580,620);
   int nevents = (int)T->GetEntries();
   for (int i=0;i<nevents;i++) {
      bntrack->GetEntry(i);
      h->Fill(event->GetNtrack());
   }

   h->Draw();
}
```

Read only one branch Much faster !

Tree example Event (read 3)

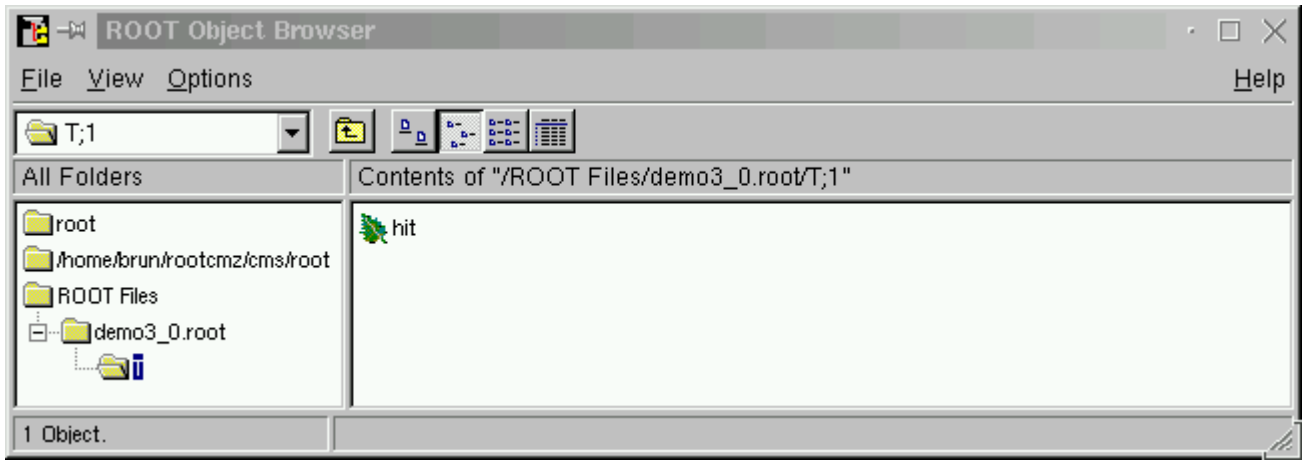
```
void demoer3() {
   //load shared lib with the Event class
   gSystem->Load("$ROOTSYS/test/libEvent");

   //connect ROOT file
   TFile *f = new TFile("demoe.root");

   //Read Tree header
   TTree *T = (TTree*)f->Get("T");

   //Histogram number of tracks via the TreePlayer
   T->Draw("event->GetNtrack()");
}
```

Writing CMS PSimHit in a Tree



```
void demo3() {
   //create a new ROOT file
   TFile f("demo3.root","recreate");

   //Create a ROOT Tree with one single top level branch
   int split = 99;  //you can try split=1 and split=0
   int bufsize = 16000;
   PSimHit *hit = 0;
   TTree T("T","CMS demo tree");
   T.Branch("hit","PSimHit",&hit,bufsize,split);

   //Create hits in a loop and fill the Tree
   TRandom r;
   for (int i=0;i<50000;i++) {
      delete hit;
      Local3DPoint pentry(r.Gaus(0,1), r.Gaus(0,1), r.Gaus(0,10));
      Local3DPoint pexit (r.Gaus(0,3), r.Gaus(0,3), r.Gaus(50,20));
      float pabs = 100*r.Rndm();
      float tof  = r.Gaus(1e-6,1e-8);
      float eloss= r.Landau(1e-3,1e-7);
      int ptype  = i%2;
      int detId  = i%20;
      int trackId= i%100;
      hit = new PSimHit(pentry,pexit,pabs,tof,eloss,ptype,detId,trackId);

      T.Fill();
   }

   T.Print();   //Print Tree statistics
   T.Write();   //Write Tree header to the file
}
```

Browsing the PSimHit Tree
split = 0



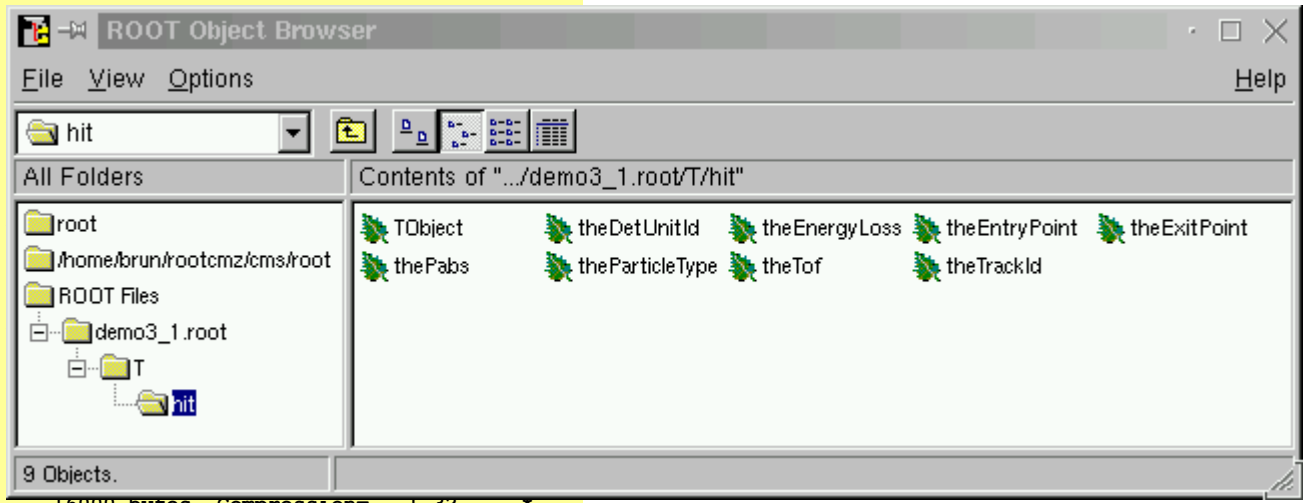

```
*Tree    :T         : CMS demo tree                                          *
*Entries :    50000 : Total =         4703775 bytes  File  Size =    2207143 *
*        :          : Tree compression factor =   2.13                       *
*****************************************************************************
*Br    0 :hit       :                                                        *
*Entries :    50000 : Total  Size=    4703775 bytes  File Size  =    2207143 *
*Baskets :      295 : Basket Size=      16000 bytes  Compression=   2.13     *
*.............................................................................*
```

1 branch only

Browsing the PSimHit Tree

split = 1

```
*******************************************************************************
*Tree    :T         : CMS demo tree                                          *
*Entries :    50000 : Total =         5258415 bytes  File  Size =    2021907 *
*        :          : Tree compression factor =   2.60                       *
*******************************************************************************
*Branch  :hit                                                                 *
*Entries :    50000 : BranchElement (see below)                               *
*.....................................
*Br    0 :TObject   :
*Entries :    50000 : Total  Size=
*Baskets :       56 : Basket Size=
*.....................................
*Br    1 :theEntryPoint :
*Entries :    50000 : Total  Size=
*Baskets :      119 : Basket Size=
*.....................................
*Br    2 :theExitPoint :
*Entries :    50000 : Total  Size=
*Baskets :      119 : Basket Size=
*.....................................
*Br    3 :thePabs   :
*Entries :    50000 : Total  Size=
*Baskets :       12 : Basket Size=
*.....................................
*Br    4 :theTof    :
*Entries :    50000 : Total  Size=
*Baskets :       12 : Basket Size=      16000 bytes  Compression=   1.32     *
*.....................................................................*
*Br    5 :theEnergyLoss :                                                     *
*Entries :    50000 : Total  Size=     191964 bytes  File Size  =     122761 *
*Baskets :       12 : Basket Size=      16000 bytes  Compression=   1.56     *
*.....................................................................*
*Br    6 :theParticleType :                                                   *
*Entries :    50000 : Total  Size=     191988 bytes  File Size  =       1860 *
*Baskets :       12 : Basket Size=      16000 bytes  Compression= 103.22     *
*.....................................................................*
*Br    7 :theDetUnitId :                                                      *
*Entries :    50000 : Total  Size=     191952 bytes  File Size  =       2298 *
*Baskets :       12 : Basket Size=      16000 bytes  Compression=  83.53     *
*.....................................................................*
*Br    8 :theTrackId :                                                        *
*Entries :    50000 : Total  Size=     191976 bytes  File Size  =       4179 *
*Baskets :       12 : Basket Size=      16000 bytes  Compression=  45.94     *
*.....................................................................*
```

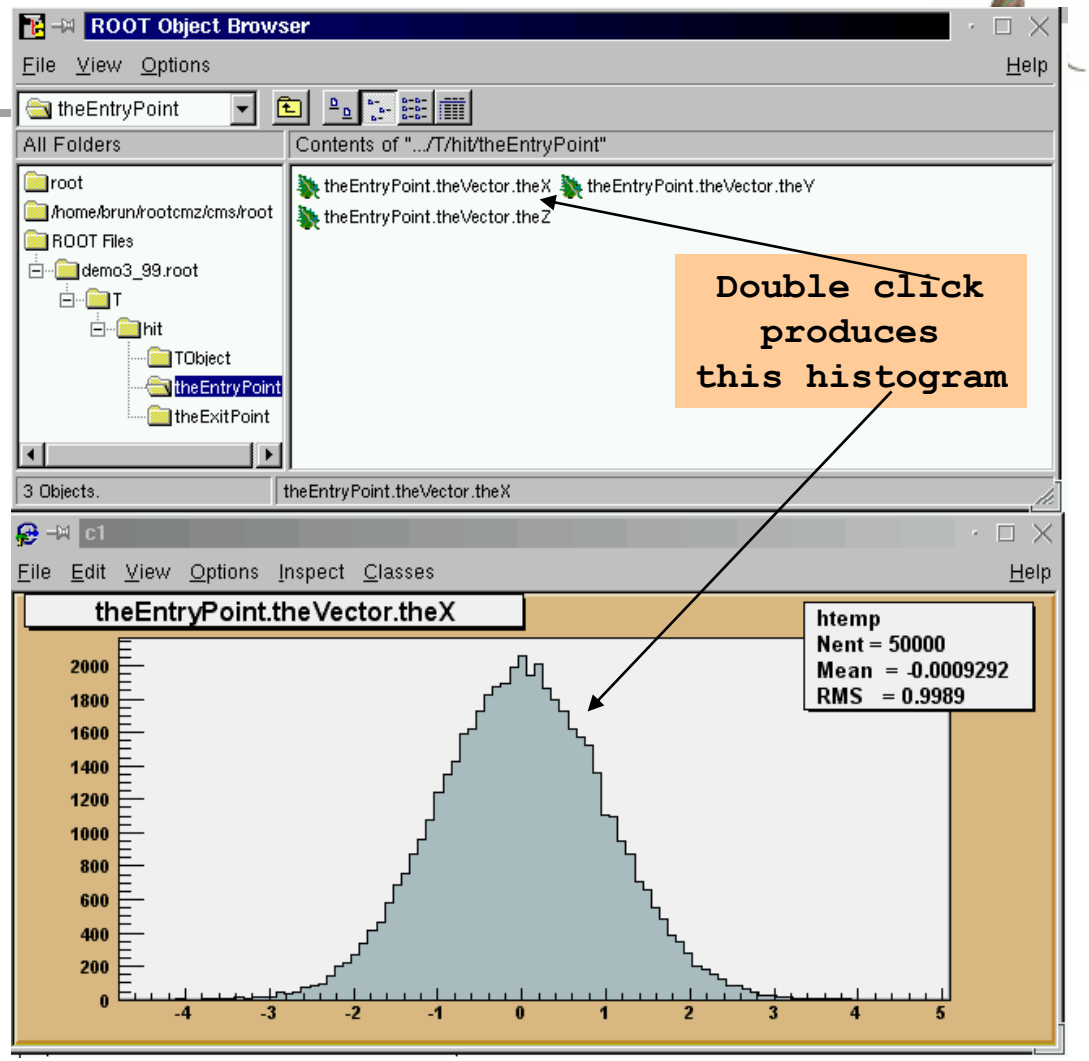


9 branches

Browsing the PSimHit Tree

split = 99

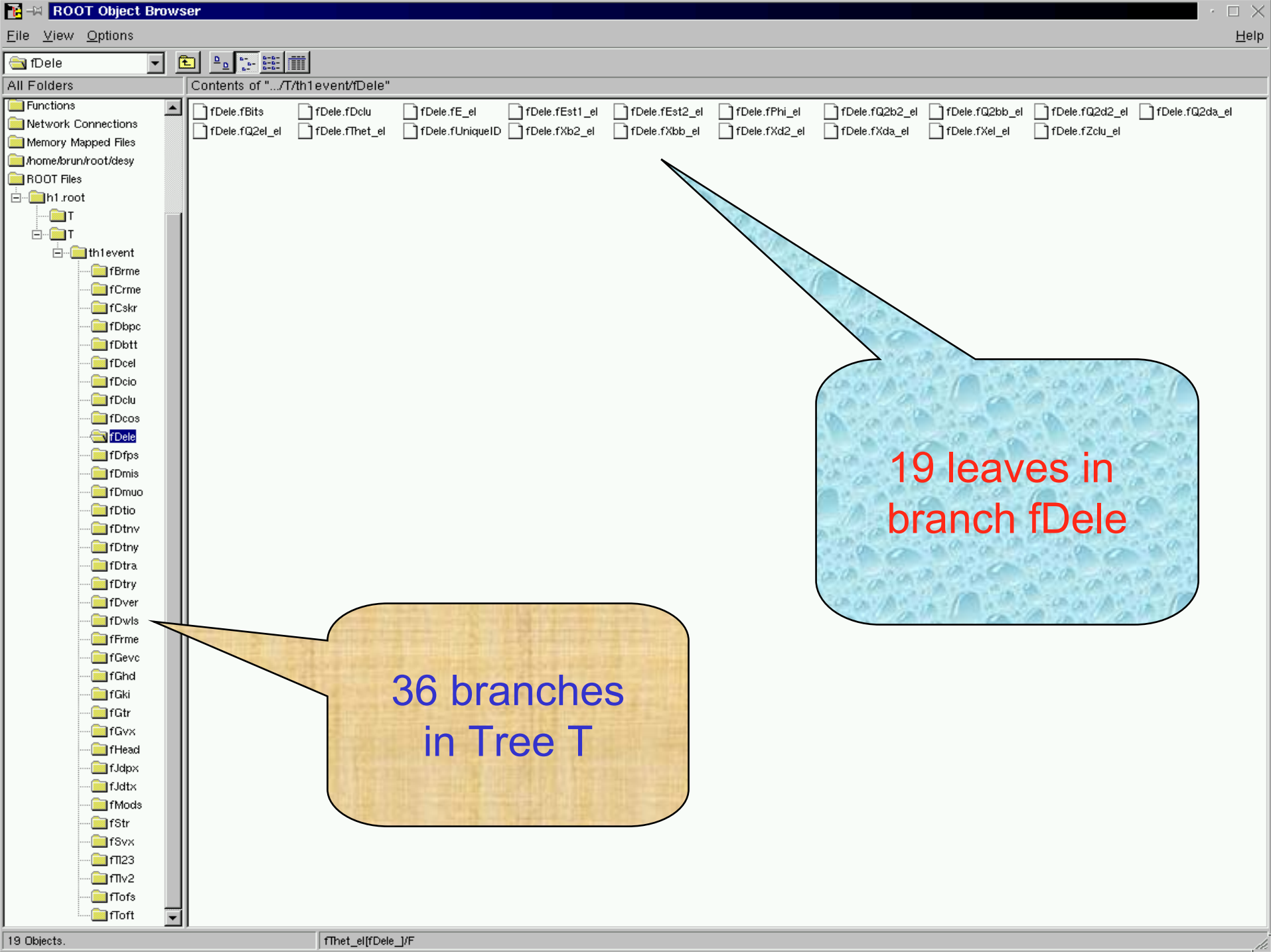
```
*******************************************************************************
*Tree    :T         : CMS demo tree                                          *
*Entries :    50000 : Total =         2687592 bytes  File  Size =    1509041 *
*        :          : Tree compression factor =   1.78                       *
*******************************************************************************
*Branch  :hit                                                                 *
*Entries :    50000 : BranchElement (see below)                               *
*.............................................................................*
*Br    0 :fUniqueID :                                                         *
*Entries :    50000 : Total  Size=     191964 bytes  File Size  =       1272 *
*Baskets :       12 : Basket Size=      16000 bytes  Compression= 150.92     *
*.............................................................................*
*Br    1 :fBits     :                                                         *
*Entries :    50000 : Total  Size=     191964 bytes  File Size  =       1260 *
*Baskets :       12 : Basket Size=      16000 bytes  Compression= 152.35     *
*.............................................................................*
*Br    2 :theEntryPoint :                                                     *
*Entries :    50000 : Total  Size=          0 bytes  File Size  =          0 *
*Baskets :        0 : Basket Size=      16000 bytes  Compression=   1.00     *
*.............................................................................*
*Br    3 :theEntryPoint.theVector.theX :                                      *
*Entries :    50000 : Total  Size=     191952 bytes  File Size  =     177959 *
*Baskets :       12 : Basket Size=      16000 bytes  Compression=   1.08     *
*.............................................................................*
*Br    4 :theEntryPoint.theVector.theY :                                      *
*Entries :    50000 : Total  Size=     191952 bytes  File Size  =     177934 *
*Baskets :       12 : Basket Size=      16000 bytes  Compression=   1.08     *
*.............................................................................*
*Br    5 :theEntryPoint.theVector.theZ :                                      *
*Entries :    50000 : Total  Size=     191952 bytes  File Size  =     178312 *
*Baskets :       12 : Basket Size=      16000 bytes  Compression=   1.08     *
*.............................................................................*
*Br    6 :theExitPoint :                                                      *
*Entries :    50000 : Total  Size=          0 bytes  File Size  =          0 *
*Baskets :        0 : Basket Size=      16000 bytes  Compression=   1.00     *
*.............................................................................*
*Br    7 :theExitPoint.theVector.theX :                                       *
*Entries :    50000 : Total  Size=     191988 bytes  File Size  =     178060 *
*Baskets :       12 : Basket Size=      16000 bytes  Compression=   1.08     *
*.............................................................................*
*Br    8 :theExitPoint.theVector.theY :                                       *
*Entries :    50000 : Total  Size=     191988 bytes  File Size  =     178072 *
*Baskets :       12 : Basket Size=      16000 bytes  Compression=   1.08     *
*.............................................................................*
*Br    9 :theExitPoint.theVector.theZ :                                       *
*Entries :    50000 : Total  Size=     191988 bytes  File Size  =     168655 *
*Baskets :       12 : Basket Size=      16000 bytes  Compression=   1.14     *
*.............................................................................*
*Br   10 :thePabs   :                                                        *
*Entries :    50000 : Total  Size=     191988 bytes  File Size  =     170871 *
*Baskets :       12 : Basket Size=      16000 bytes  Compression=   1.12     *
*.............................................................................*
*Br   11 :theTof    :                                                        *
*Entries :    50000 : Total  Size=     191976 bytes  File Size  =     145548 *
*Baskets :       12 : Basket Size=      16000 bytes  Compression=   1.32     *
*.............................................................................*
*Br   12 :theEnergyLoss :                                                    *
*Entries :    50000 : Total  Size=     191964 bytes  File Size  =     122761 *
*Baskets :       12 : Basket Size=      16000 bytes  Compression=   1.56     *
*.............................................................................*
*Br   13 :theParticleType :                                                  *
*Entries :    50000 : Total  Size=     191988 bytes  File Size  =       1860 *
*Baskets :       12 : Basket Size=      16000 bytes  Compression= 103.22     *
*.............................................................................*
*Br   14 :theDetUnitId :                                                     *
*Entries :    50000 : Total  Size=     191952 bytes  File Size  =       2298 *
*Baskets :       12 : Basket Size=      16000 bytes  Compression=  83.53     *

*.............................................................................*
*Br   15 :theTrackId :                                                       *
*Entries :    50000 : Total  Size=     191976 bytes  File Size  =       4179 *
*Baskets :       12 : Basket Size=      16000 bytes  Compression=  45.94     *
*.............................................................................*
```

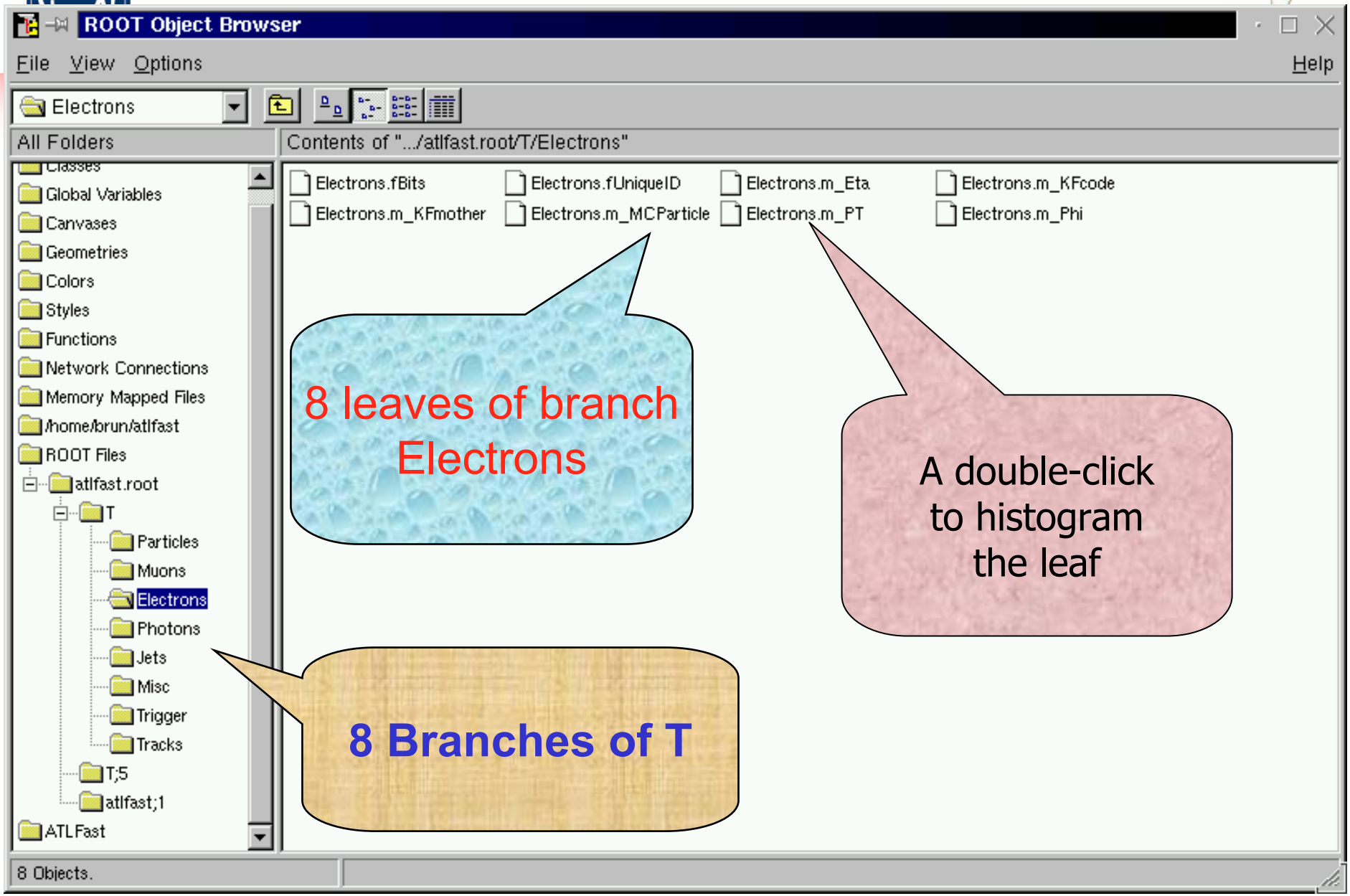


16 branches

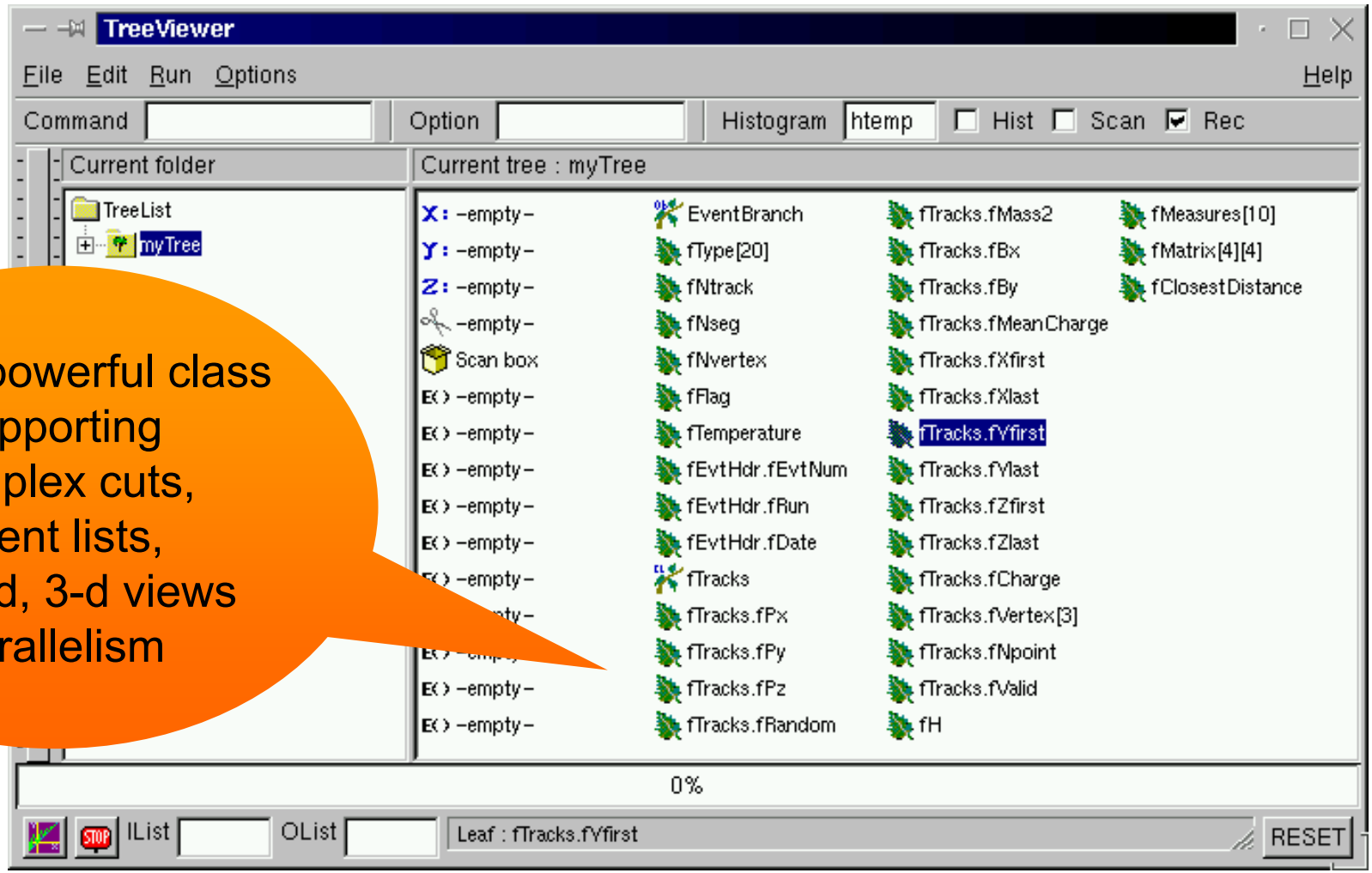
Collections of Hits

- A more realistic Tree will have
  - A collection of Detectors
  - Each detector one or more collection of hits


ROOT Object Browser
File View Options Help
fDele
All Folders
Contents of ".../T/th1event/fDele"
Functions
Network Connections
Memory Mapped Files
/home/brun/root/desy
ROOT Files
h1.root
T
T
th1event
fBrme
fCrme
fCskr
fDbpc
fDbtt
fDcel
fDcio
fDclu
fDcos
fDele
fDfps
fDmis
fDmuo
fDtio
fDtnv
fDtny
fDtra
fDtry
fDver
fDwls
fFrme
fGevc
fGhd
fGki
fGtr
fGvx
fHead
fJdpx
fJdtx
fMods
fStr
fSvx
fTl23
fTlv2
fTofs
fToft
fDele.fBits
fDele.fDclu
fDele.fE_el
fDele.fEst1_el
fDele.fEst2_el
fDele.fPhi_el
fDele.fQ2b2_el
fDele.fQ2bb_el
fDele.fQ2d2_el
fDele.fQ2da_el
fDele.fQ2el_el
fDele.fThet_el
fDele.fUniqueID
fDele.fXb2_el
fDele.fXbb_el
fDele.fXd2_el
fDele.fXda_el
fDele.fXel_el
fDele.fZclu_el
19 leaves in branch fDele
36 branches in Tree T
19 Objects.
fThet_el[fDele_]/F

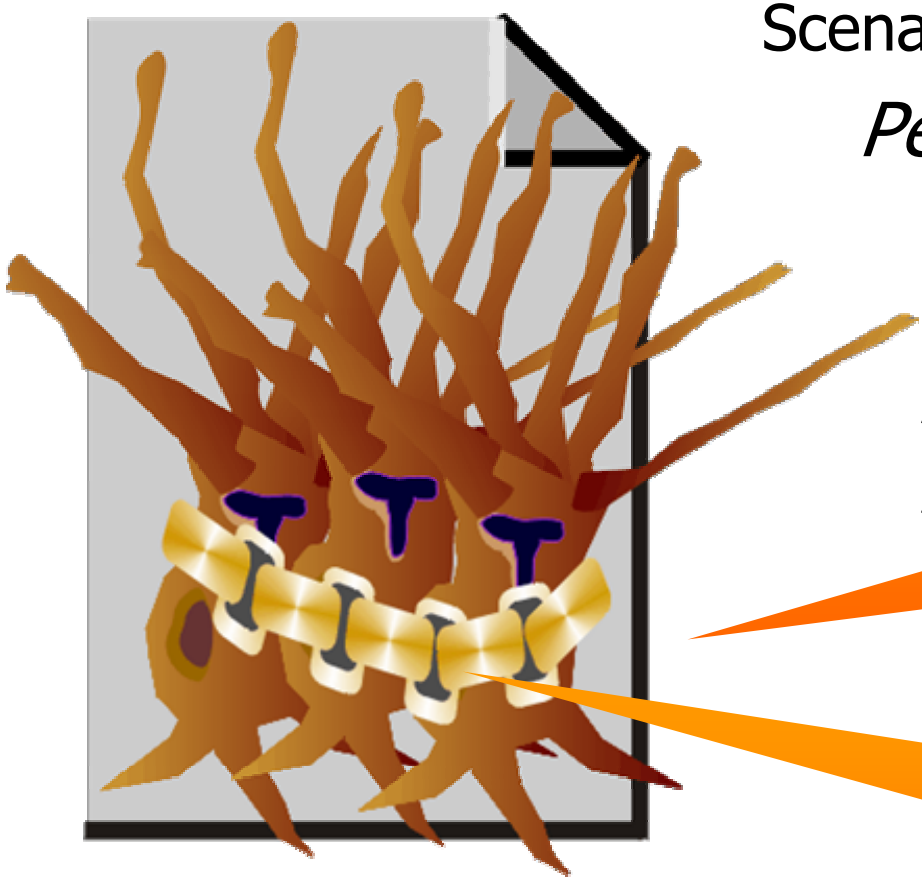

ROOT Object Browser
File View Options Help
Electrons
All Folders
Contents of ".../atlfast.root/T/Electrons"
Global Variables
Canvases
Geometries
Colors
Styles
Functions
Network Connections
Memory Mapped Files
/home/brun/atlfast
ROOT Files
atlfast.root
T
Particles
Muons
Electrons
Photons
Jets
Misc
Trigger
Tracks
T;5
atlfast;1
ATLFast
Electrons.fBits
Electrons.fUniqueID
Electrons.m_Eta
Electrons.m_KFcode
Electrons.m_KFmother
Electrons.m_MCParticle
Electrons.m_PT
Electrons.m_Phi
8 leaves of branch Electrons
A double-click to histogram the leaf
8 Branches of T
8 Objects.

The Tree Viewer & Analyzer



A very powerful class supporting complex cuts, event lists, 1-d,2-d, 3-d views parallelism

Chains



Scenario:

Perform an analysis using multiple ROOT files. All files are of the same structu… the san…

Chains
are collections of
chains or files

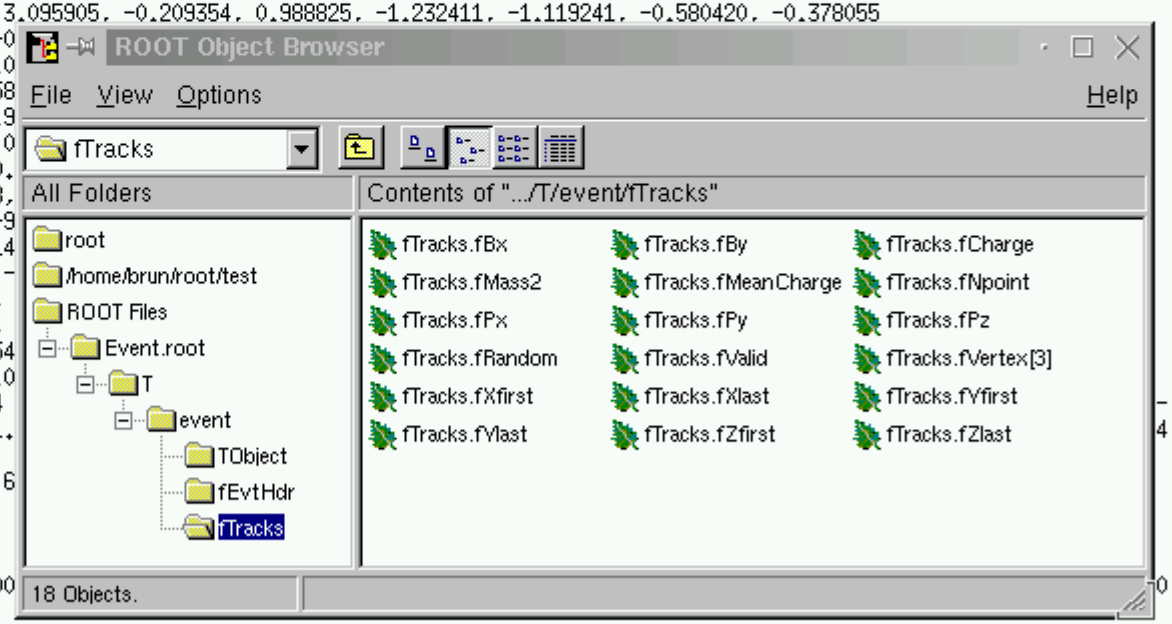
Chains can be built
automatically by quering
the run/file catalog

The "No Shared Library" case

- There are many applications for which it does not make sense to read data without the code of the corresponding classes.
- In true OO, you want to exploit Data Hiding and rely on the functional interface.
- However, there are also cases where the functional interface is not necessary (PAW ntuples).
- It is nice to be able to browse any type of file without any code. May be you cannot do much, but it gives some confidence that you can always read your data sets.
- We have seen a religious debate on this subject.
- Our conclusion was that we had to support these two modes of operation.
- Support for the "No Shared Lib case" is non trivial

read/query Trees without the classes

```
root [0] TFile f("Event.root")
Warning in <TClass::TClass>: no dictionary for class Event is available
Warning in <TClass::TClass>: no dictionary for class EventHeader is available
Warning in <TClass::TClass>: no dictionary for class Track is available
root [1] T.Show(45)
======> EVENT:45
 fUniqueID       = 0
 fBits           = 50331648
 fType[20]       = 116 121 112 101 48 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 fNtrack         = 609
 fNseg           = 6066
 fNvertex        = 7
 fFlag           = 1
 fTemperature    = 20.529432
 fEvtHdr.fEvtNum = 45
 fEvtHdr.fRun    = 200
 fEvtHdr.fDate   = 960312
 fTracks         = 609
 fTracks.fPx     = -0.077692, -2.625842, 1.130895, 3.095905, -0.209354, 0.988825, -1.232411, -1.119241, -0.580420, -0.378055
 fTracks.fPy     = 0.332117, 0.482258, -0.789722, -0
 fTracks.fPz     = 0.341083, 2.669760, 1.379341, 3.0
 fTracks.fRandom = 529.431580, 529.431580, 529.43158
 fTracks.fMass2  = 8.900000, 8.900000, 8.900000, 8.9
 fTracks.fBx     = -0.042621, 0.069631, -0.141984, 0
 fTracks.fBy     = 0.001728, 0.116303, 0.046655, -0.
 fTracks.fMeanCharge = 0.001209, 0.001738, 0.006828,
 fTracks.fXfirst = 2.092409, 0.549266, -1.027804, -9
 fTracks.fXlast  = 5.749741, 15.938519, 0.260784, 14
 fTracks.fYfirst = 10.398095, 2.455857, 16.579025, -
 fTracks.fYlast  = -7.106707, 13.617641, 12.946399,
 fTracks.fZfirst = 56.846382, 39.277451, 47.035370,
 fTracks.fZlast  = 211.758606, 213.753479, 217.62054
 fTracks.fCharge = 0.000000, 0.000000, 1.000000, 0.0
 fTracks.fVertex[3] = -0.181014 0.105711 -20.070364
0.034267 0.096083 -3.769124 , -0.119004 0.038680 -1.
79879 , -0.165064 0.044276 -9.960069
 fTracks.fNpoint = 64, 60, 66, 61, 62, 61, 62, 64, 6
 fTracks.fValid  = 1, 0, 1, 0, 1, 1, 0, 1, 0, 1
 fH              = (TH1F*)8a93ed0
 fMeasures[10]   = -2 0 5 2 5 1 14 13 9 11
 fMatrix[4][4]   = -0.625079 0.530725 -0.786688 0.00
.000000 0.000000
 fClosestDistance = 1.441706 1.708780 -0.655489 1.539576 0.804130 0.048241 -0.594909
root [2] new TBrowser
(class TBrowser*)0x88b7460
root [3]
```



TFile::MakeProject

Generate the classes
header files
Compile them
make a shared lib
link the shared lib

```
root [4] f.MakeProject("xx","*","recreate++")
MakeProject has generated 21 classes in xx
xx/MAKE file has been generated
Shared lib xx/xx.so has been generated
Shared lib xx/xx.so has been dynamically linked
root [5] ATLFElectron e
root [6] e.Dump()
m_KFcode                11          Electron KF-code
m_MCParticle            6           Electron position in MCParticles list
m_KFmother              0           Electron mother KF-code
m_Eta                   0           Eta coordinate
m_Phi                   0           Phi coordinate
m_PT                    0           Transverse energy
fUniqueID               0           object unique identifier
fBits                   50331648    bit field status word
```

TFile::MakeProject

```
//////////////////////////////////////////////////////////
//   This class has been generated by TFile::MakeProject
//     (Mon May 28 19:34:37 2001 by ROOT version 3.01/03)
//      from the StreamerInfo in file atlfast.root
//////////////////////////////////////////////////////////


#ifndef ATLFElectron_h
#define ATLFElectron_h

#include "TObject.h"
#include "TAtt3D.h"

class ATLFElectron : public TObject , public TAtt3D {

public:
   Int_t        m_KFcode;       //Electron KF-code
   Int_t        m_MCParticle;   //Electron position in MCParticles
   Int_t        m_KFmother;     //Electron mother KF-code
   Float_t      m_Eta;          //Eta coordinate
   Float_t      m_Phi;          //Phi coordinate
   Float_t      m_PT;           //Transverse energy

   ATLFElectron() {;}
   virtual ~ATLFElectron() {;}

   ClassDef(ATLFElectron,1) //
};

   ClassImp(ATLFElectron)
#endif
```

All necessary header files are included

Comments preserved

Can do I/O Inspect Browse,etc