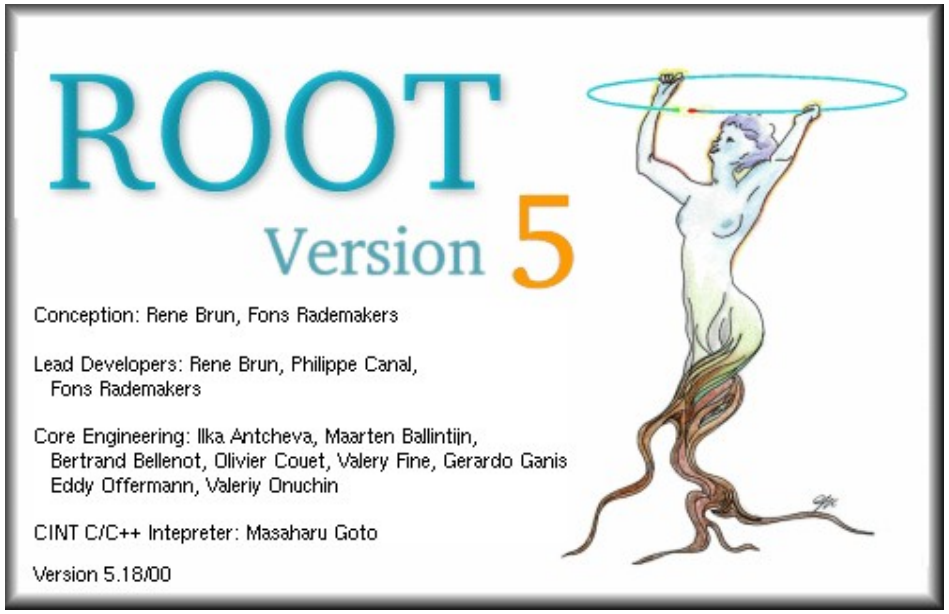
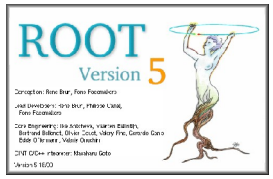
ROOT
A Data Analysis Framework



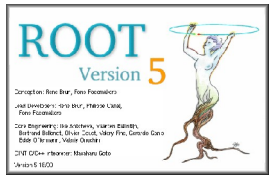
Ph. von Doetinchem
philip.doetinchem@rwth-aachen.de
I. Phys. Inst. B, RWTH Aachen University
April 2009

Overview



- general concepts
- interactive use, macros, compiled scripts
- canvas and style
- histograms and graphs
- fitting
- write and read files
- standalone programs using ROOT

Installation from Source



- download root source: http://root.cern.ch
- Type the build commands:

```
>cd root
>./configure -help
>./configure [<arch>]       [set arch appropriately if no proper
default]
>(g)make                    [or, make -j n for n core machines]
```

- Add bin/ to PATH and lib/ to LD_LIBRARY_PATH. For the sh shell family do:

```
>. bin/thisroot.sh
```

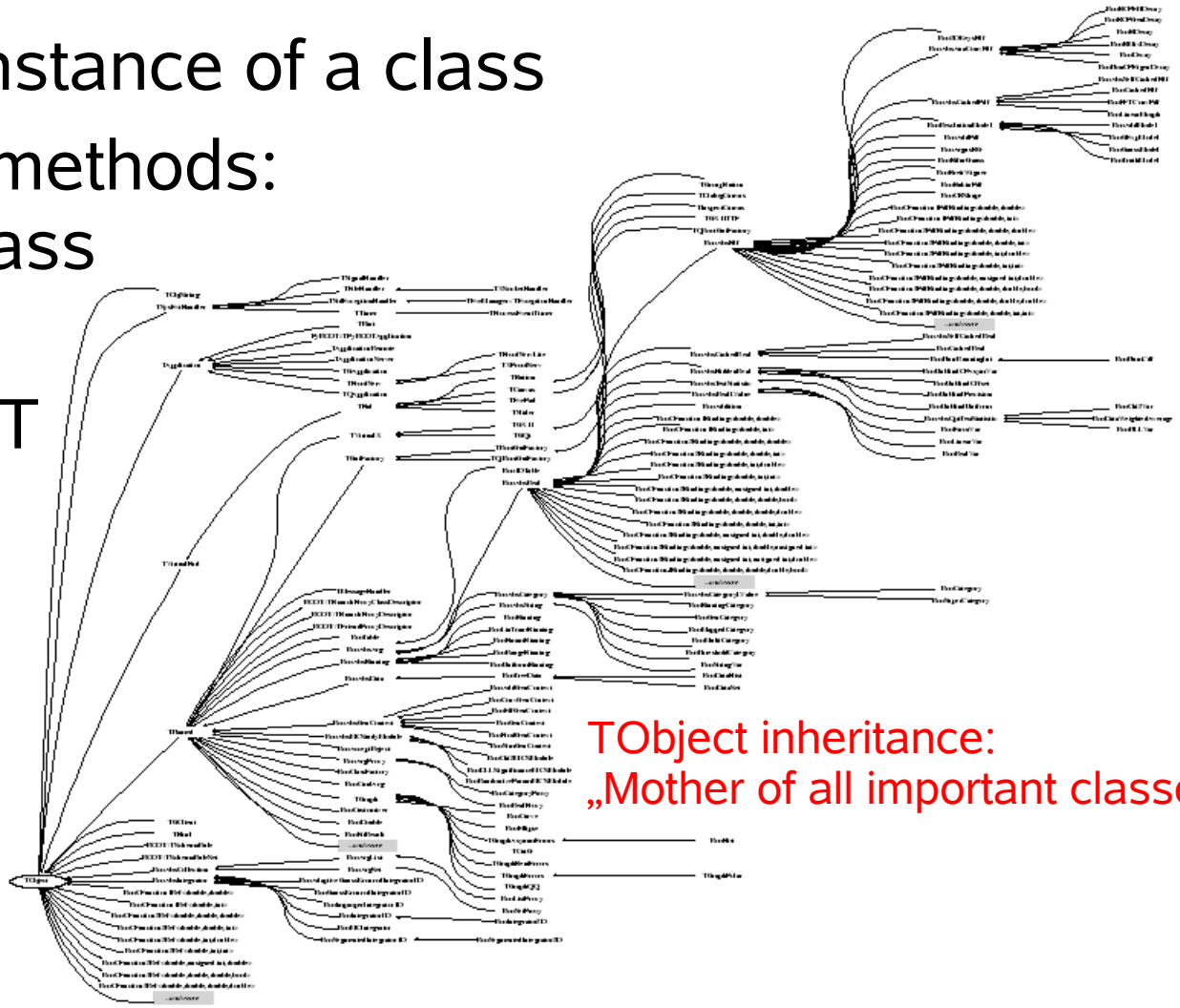
and for the csh shell family do:

```
>source bin/thisroot.csh
```

General Concepts

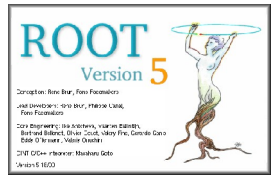
- consists of several 100s of C++ classes
- create objects: instance of a class
- every class has methods: functions for a class
- classes can be used in the ROOT framework

  or in standalone C++ programs



TObject inheritance:
„Mother of all important classes“

The Object Oriented Frame Work



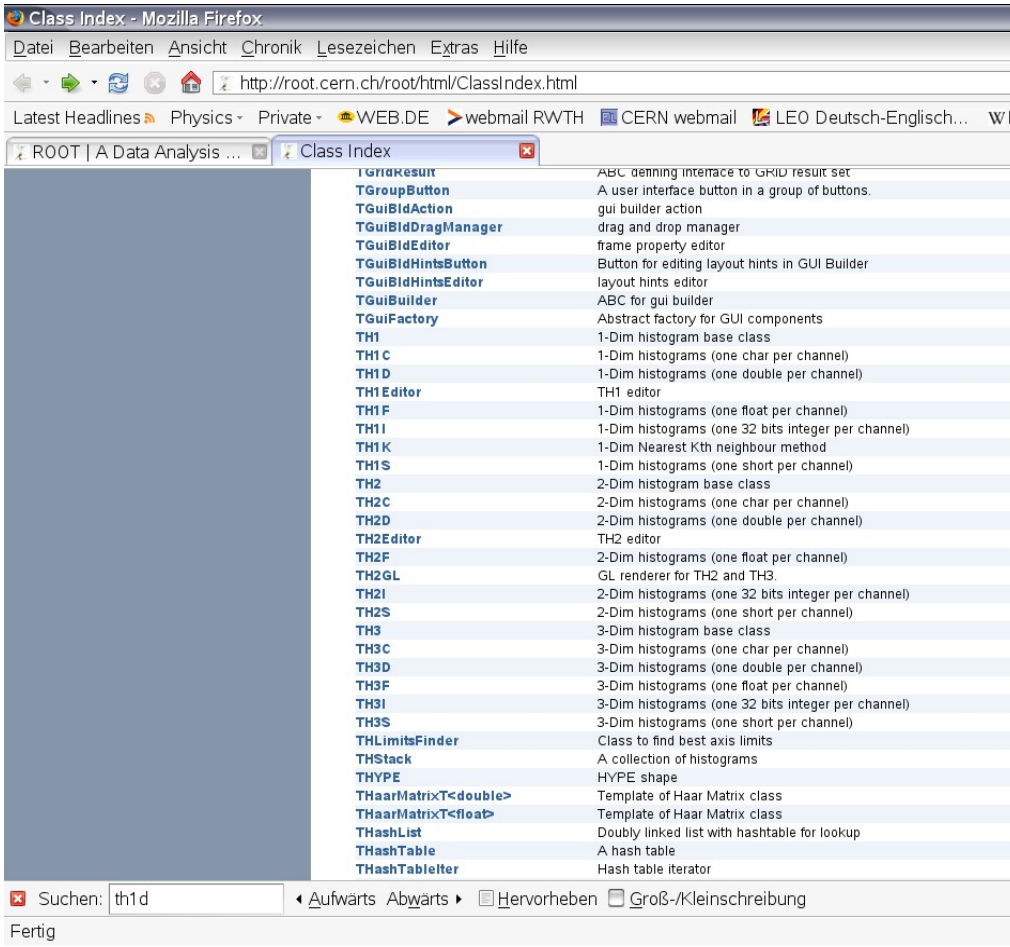
The ROOT framework consists of over 350 C++ classes.

Overview: http://root.cern.ch/root/html/ClassIndex.html

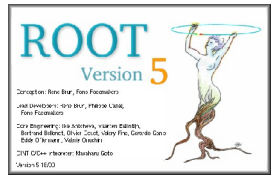
Class Categories:

1. Base Classes
2. Container Classes
3. Histograms and Minimization Classes
4. Tree and Ntuple Classes
5. Linear Algebra, Matrix, and Vector Classes
6. 2D & 3D Graphics Classes

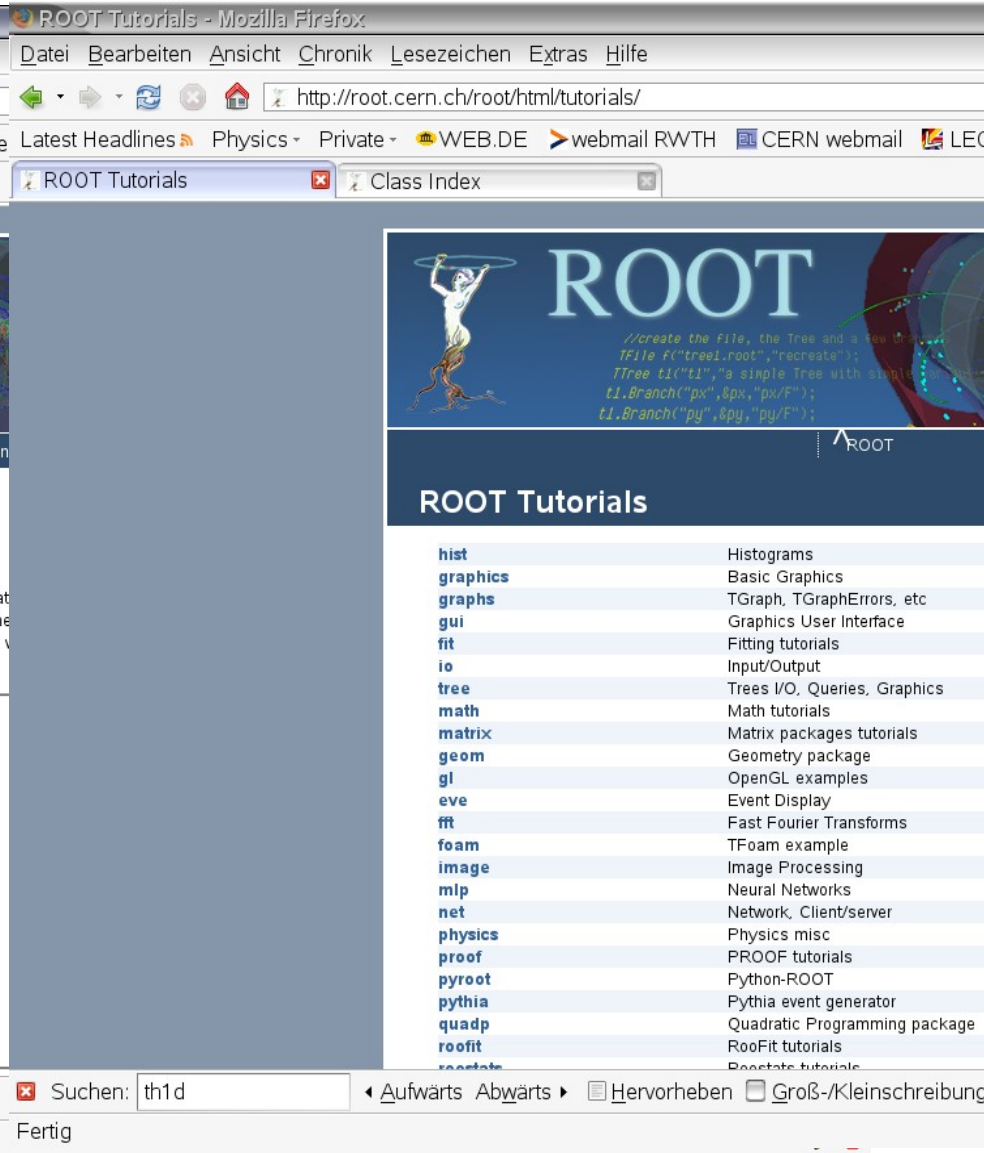
and some more...



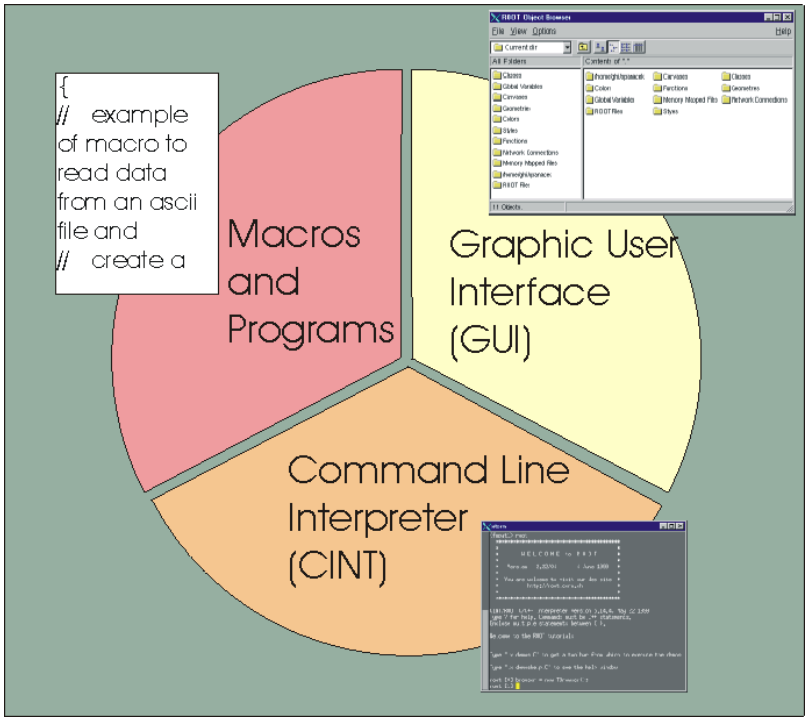
ROOT Homepage: Help



most important page: root.cern.ch

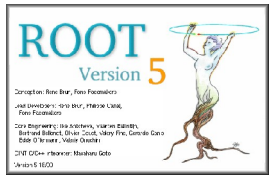


Different Ways to Use ROOT



- GUI windows, buttons, menus
- Root Command line CINT (C++ interpreter)
- Macros, applications, libraries
- standalone programs

Interactive Interpreter



starting and quitting ROOT

```
>root
[0].q
```

options: without splash screen, batch mode

```
>root -l
>root -b
```

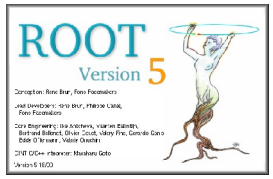
shell commands

```
[0].! ls
```

first commands (reg. C++ commands):

```
[0]cout<<“Hello World“<<endl;
Hello World
[1]int i = 2;
[2]int j = 3;
[3]int k = i+j;
[4]cout<<k<<endl;
5
```

Macros



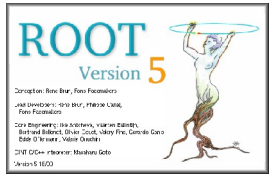
file: macro.cpp

```
{cout<<“Hello
World“<<endl;
int i = 2;
int j = 3;
int k = i+j;
cout<<k<<endl;}
```

```
[0].x macro.cpp
Hello World
5
```

- nearly all usual C++ commands can be used in interactive or macro mode
- tab completion
- **BUT:** commands are interpreted and **not** compiled before execution:
  - slower
  - **huge** source of bugs

Compiled Scripts



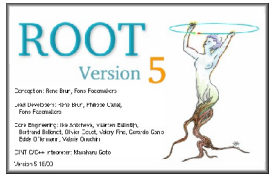
file: compiled.cpp

```
#include<iostream>


void example()
{
cout<<"Hello World"<<endl;
int i = 2;
int j = 3;
int k = i+j;
cout<<k<<endl;
}
```

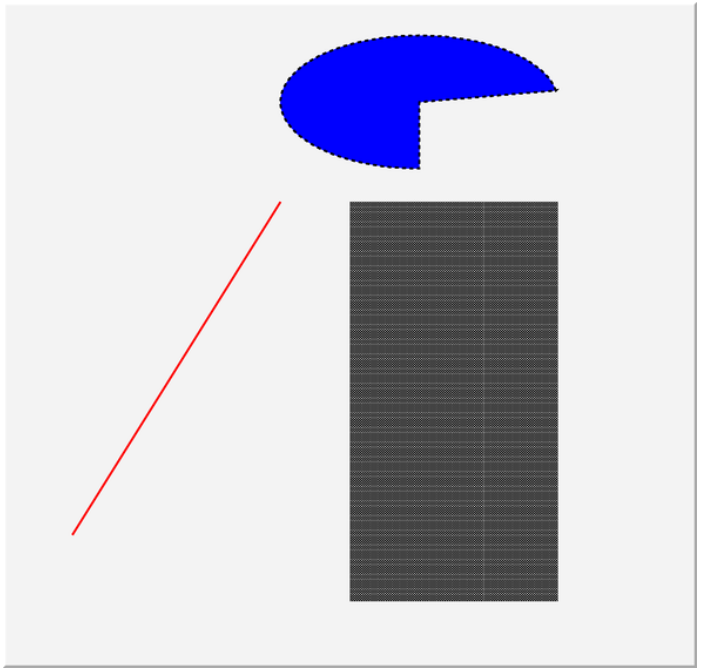
```
[0].L compiled.cpp+
[1]example()
Hello World
5
```

- build up scripts with a lot of different C++ functions using the ROOT classes
- debug with compiler

Draw TCanvas and some Objects

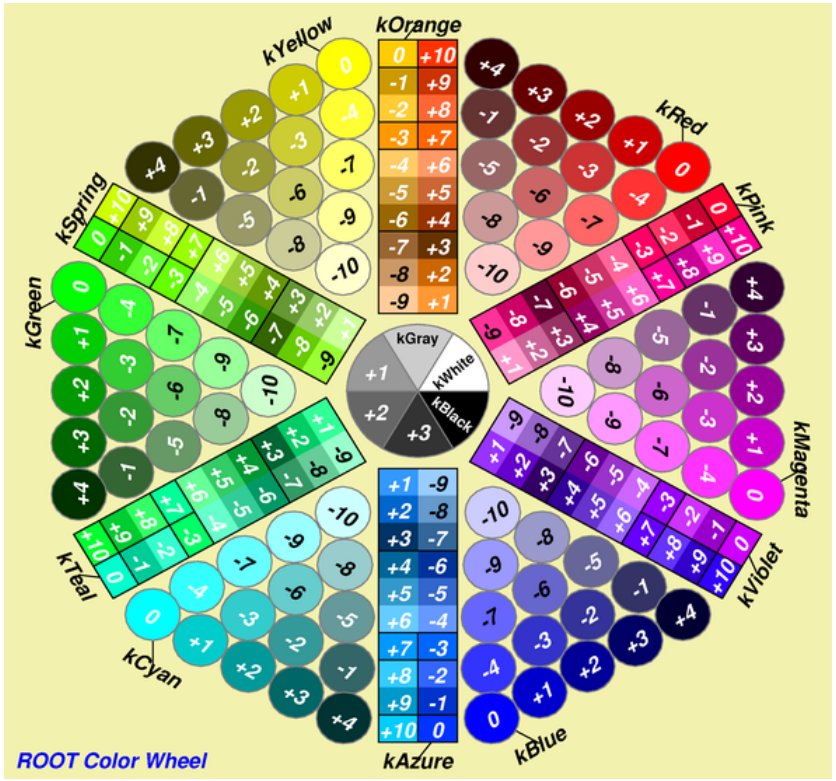



```
#include<iostream>
#include<TCanvas.h>
#include<TLine.h>
#include<TBox.h>
#include<TEllipse.h>
#include<TROOT.h>

void canvas()
{
gROOT->Reset();

TCanvas * c = new TCanvas("c_ref","c_title",
200,10,600,600);

TLine * l = new TLine(0.1,0.2,0.4,0.7);
l->SetLineWidth(2);
l->SetLineColor(2);
l->Draw();

TBox * b = new TBox(0.5, 0.1, 0.8, 0.7);
b->SetFillColor(1);
b->SetFillStyle(3001);
b->Draw();

TEllipse * e = new TEllipse(0.6, 0.85, 0.2, 0.1,10, 270, 0);
e->SetLineStyle(2);
e->SetLineWidth(2);
e->SetFillColor(4);
e->Draw();
}
```



- include classes you like to use
- reset memory
- create canvas
- choosing style, color, width is the same for all the objects one can draw

Color and Style

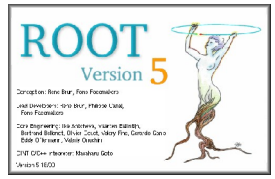


- many nice colors, **but** avoid e.g. yellow and light green!

- set style of canvas and colors at the beginning of your script

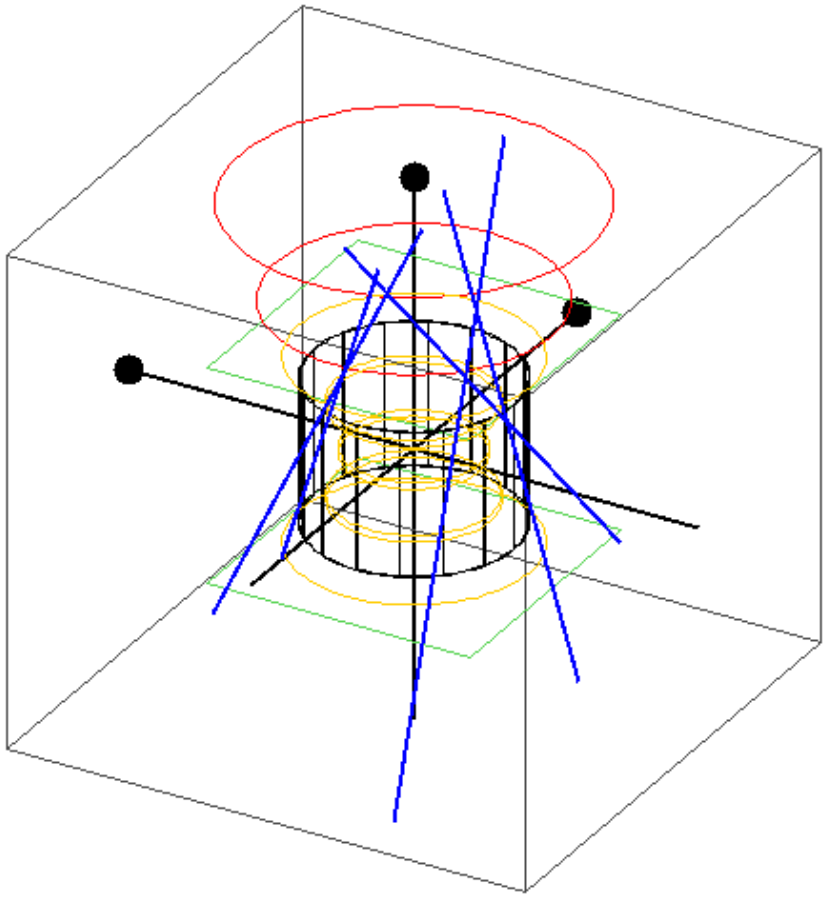
```
TStyle * plain = new TStyle("plain","plain");
plain->SetCanvasBorderMode(0);
plain->SetPadBorderMode(0);
plain->SetPadColor(0);
plain->SetCanvasColor(0);
plain->SetTitleColor(1);
plain->SetStatColor(0);
plain->SetTitleFillColor(0);

gROOT->SetStyle("plain");

int NRGBs = 5;
int NCont = 255;

double stops[] = { 0.00, 0.34, 0.61, 0.84, 1.00 };
double red[]   = { 0.00, 0.00, 0.87, 1.00, 0.51 };
double green[] = { 0.00, 0.81, 1.00, 0.20, 0.00 };
double blue[]  = { 0.51, 1.00, 0.12, 0.00, 0.00 };
TColor::CreateGradientColorTable(NRGBs, stops, red,
green, blue, NCont);
gStyle->SetNumberContours(NCont);
```

TPolyline3D

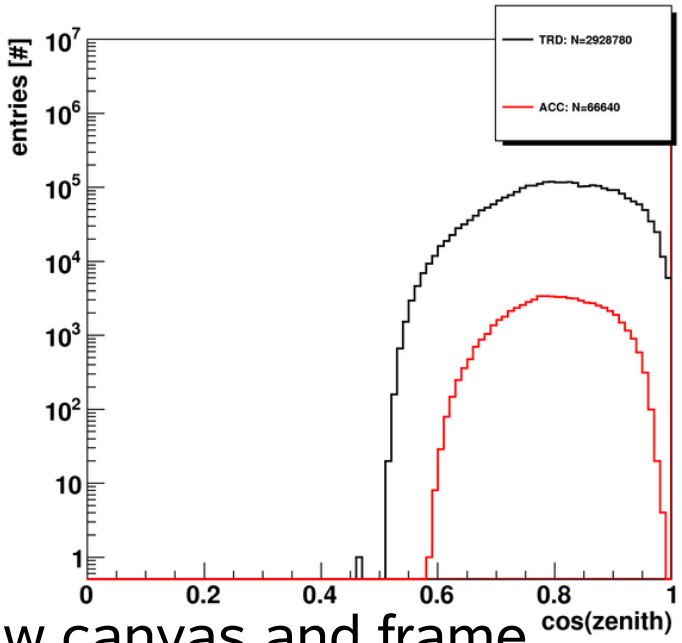


Illustrate even complex 3D objects...

```
TView *view = Tview::CreateView(1);
view->SetRange( -1.5, -1.5,-1.5, 1.5, 1.5, 1.5);


TPolyLine3D *cylinder_outer_bottom = new
TPolyLine3D();
double height = 0.8;
double radius_outer = 0.55;
for(int i = 1; i <= 100; i++)
     {
     double alpha = 2*TMath::Pi()/100.0*i;
cylinder_outer_bottom->SetPoint(i-1,
radius_outer*cos(alpha),radius_outer*sin(alpha),-0
.5*height);
     }
cylinder_outer_bottom->Draw();
```



TH1 Histograms



- draw canvas and frame

```
TCanvas * c = new TCanvas("c","theta",
200,10,600,600);
c->SetLeftMargin(0.12);
c->SetRightMargin(0.05);

TH1F * frame = c->DrawFrame(0,0.5,1, 1e7);

frame->SetTitle("");
frame->GetYaxis()->SetTitle("entries [#]");
frame->GetXaxis()->SetTitle("cos(zenith)");
frame->GetXaxis()->SetNdivisions(505);
frame->GetYaxis()->SetTitleOffset(1.5);
```

- create and fill histogram

```
TH1D * h_trd = new TH1D("trd", "trd",
100,0,1);

double value;
h_trd->Fill(value);
```

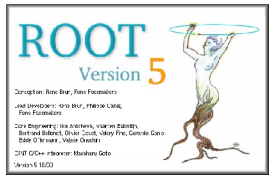
- draw histogram

```
h_trd->SetLineWidth(2);
h_trd->SetLineColor(1);
h_trd->Draw("same");

h_acc->SetLineWidth(2);
h_acc->SetLineColor(2);
h_acc->Draw("same");

gPad->SetLogy(1);
```

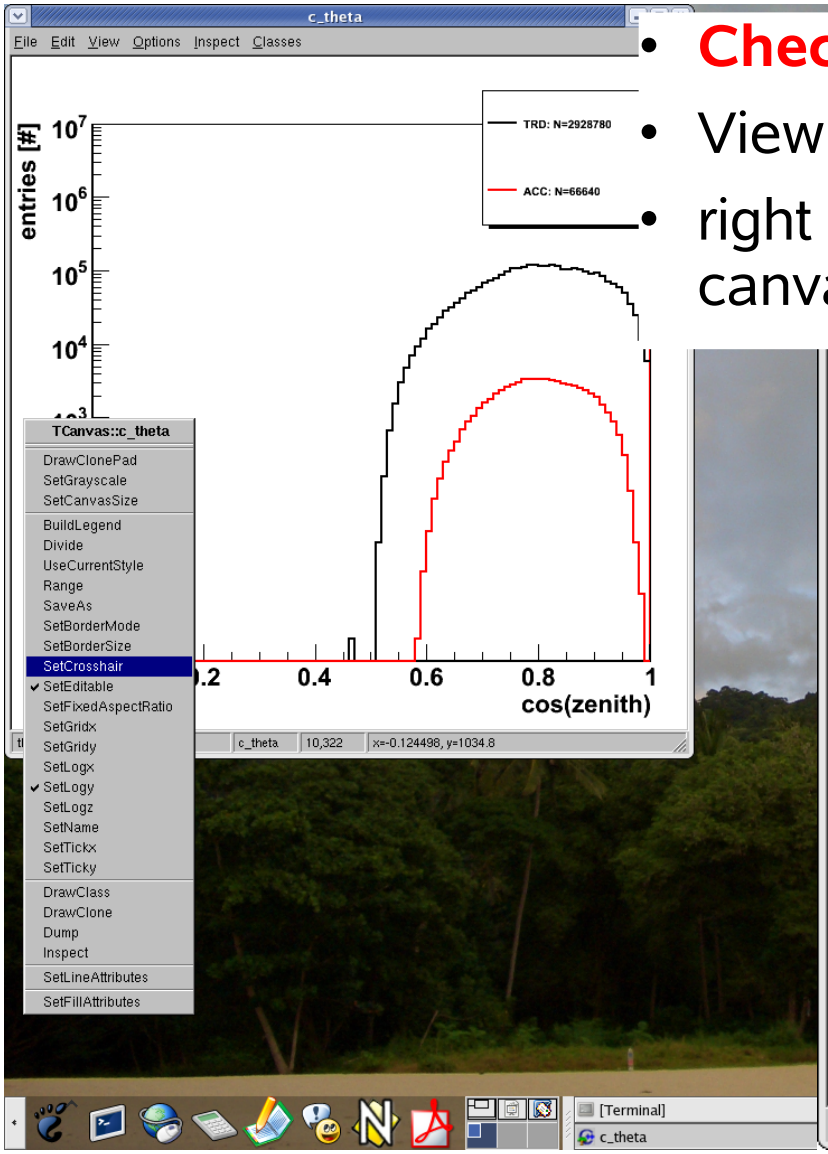
- draw legend

```
TLegend* leg = new
TLegend(0.7,0.75,0.95,0.95);
leg->SetFillColor(0);
leg->AddEntry(h_trd,“TRD“,"L");
leg->AddEntry(h_acc_theta,“ACC“,"L");

leg->Draw();
```

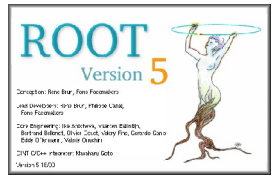
Some interactive Stuff



- **Check values:**
- View → Event Statusbar
- right click on the right side of the canvas: check SetCrosshair



Important TH1 Methods




```
int      Fill(double x, double w)
void     AddBinContent(int bin, double w)
void     SetBinContent(int bin, double content)
void     SetBinError(int bin, double error)

void     FillRandom(const char* fname, int
ntimes = 5000)
void     FillRandom(TH1* h, int ntimes = 5000)
double   GetRandom()

void     TH1::Add(const TH1* h1, double c1 = 1)
void     Multiply(TF1* h1, double c1 = 1)
void     Divide(TF1* f1, double c1 = 1)

double   GetEntries()
double   Integral(Option_t* option = "")
void     Scale(double c1 = 1, Option_t* option =
"")

double   GetMean(int axis = 1)
double   GetRMS(int axis = 1)

double   GetBinCenter(int bin)
double   GetBinContent(int bin)
double   GetBinError(int bin)
double   GetBinLowEdge(int bin)
double   GetBinWidth(int bin)
```

```
intGetMaximumBin()
intGetMinimumBin()

TObject*TObject::Clone(const char* newname =
"")
void     Reset(Option_t* option = "")

TH1*     Rebin(int ngroup = 2, char* newname =
"", double* xbins = 0)
void     Smooth(int ntimes = 1, Option_t*
option = "")

int Fit(TF1* f1, Option* option = "", Option*
goption = "", double xmin = 0, double xmax =
0)
```

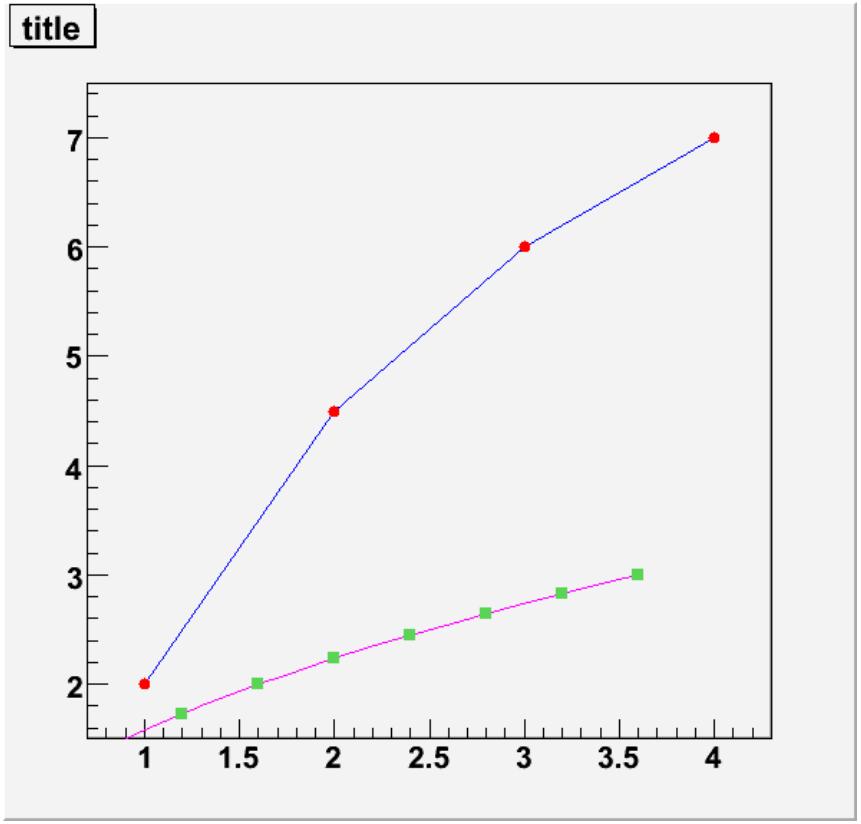
Important and often used TH1 methods, for sure **NOT** complete!

http://root.cern.ch/root/html/TH1D.html

TGraph

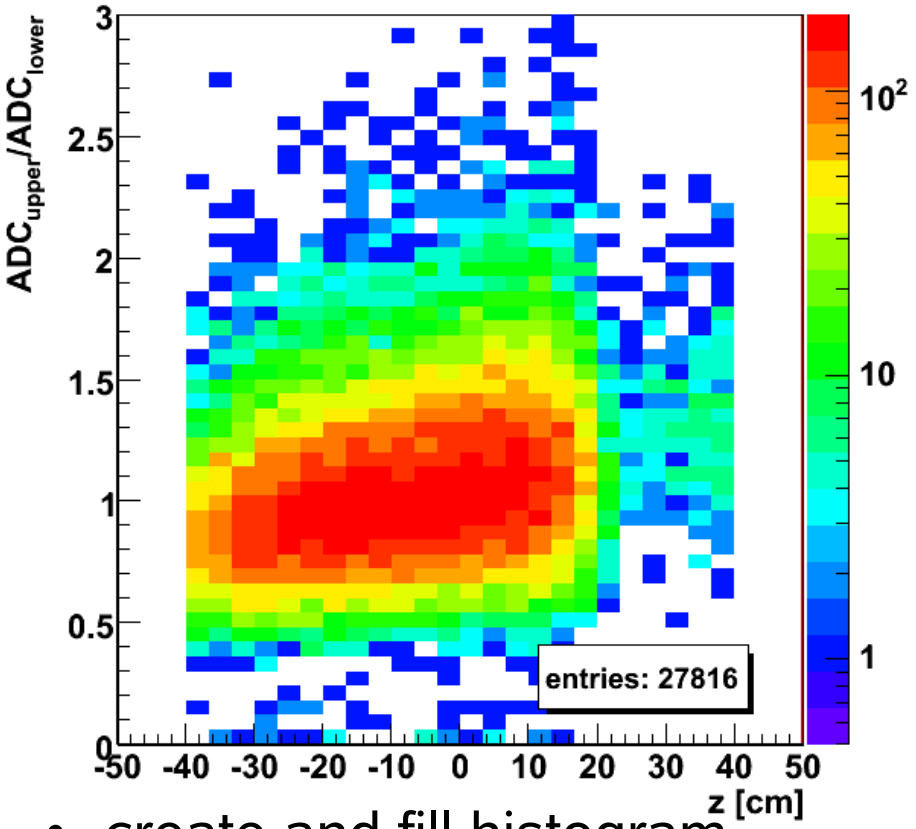
- different ways of creating graphs

```
int n = 4;
double x[] = {1,2,3,4};
double y[] = {2,4.5,6,7};

TGraph * g1 = new TGraph(n,x,y);

g1->SetTitle("title");

g1->SetMarkerStyle(20);
g1->SetMarkerColor(2);
g1->SetLineColor(4);
g1->Draw("apl");

TGraph * g2 = new TGraph();
for(double i = 0; i < 10; i++) g2->SetPoint(int(i),i*0.4,sqrt(i));

g2->SetMarkerStyle(21);
g2->SetMarkerColor(8);
g2->SetLineColor(6);
g2->Draw("pc");
```



- marker styles

TH2

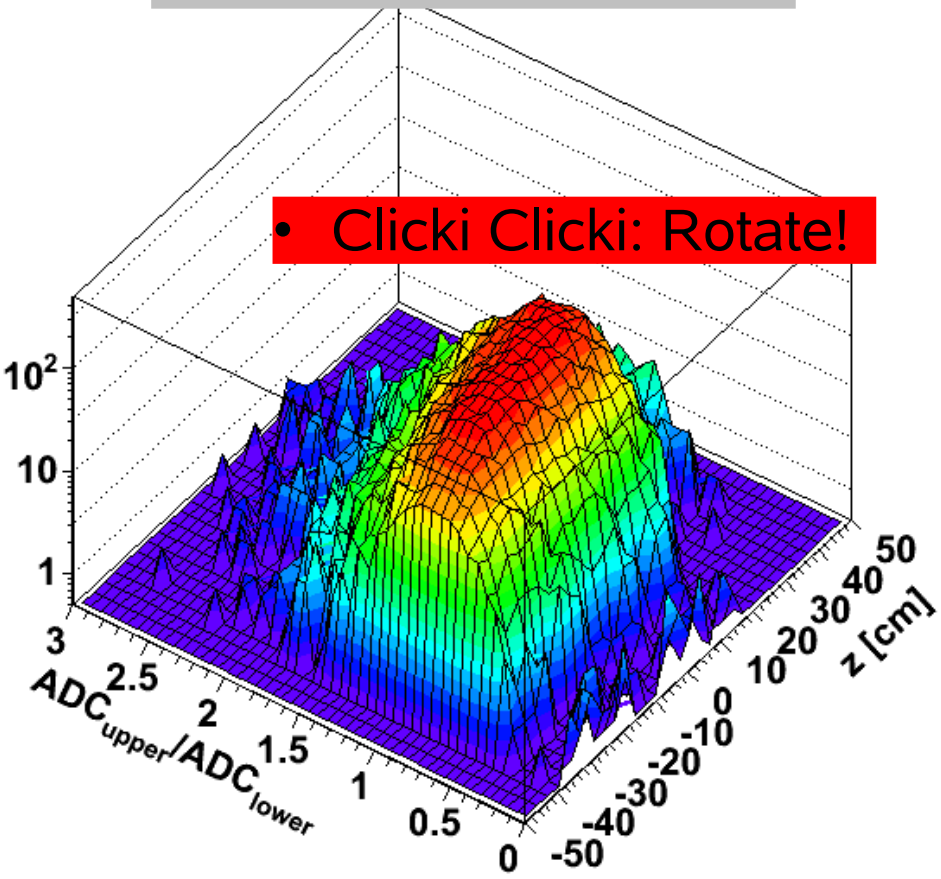


- create and fill histogram

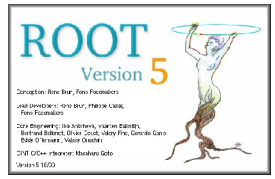
```
TH2D * h_frac = new TH2D("frac", "",
30,-50,50, 50,0,3);
double x, y;
h_frac->Fill(x,y);
h_frac->Draw("colz");
```

A lot of different styles exist!
Similar commands as for TH1

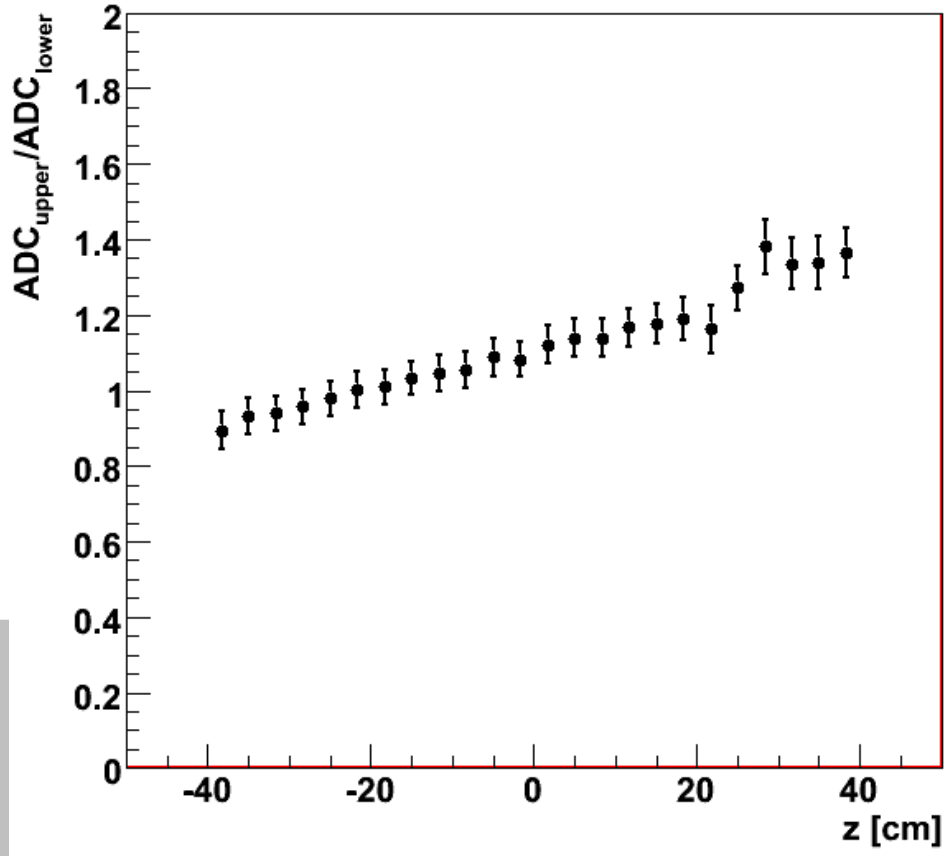
```
h_acc_z_frac->Draw("surf1");
```



TProfile and TGraphErrors

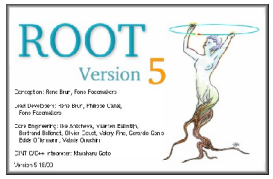


- extract profile in y direction from TH2 histogram
- take mean and RMS of profile as input for TGraphErrors
- plot graph with error bars




```
TGraphErrors * g_frac_mean = new TGraphErrors();
int ctr = 0;
for(int i = 1; i <= h_frac->GetNbinsX(); i++)
     {
     TProfile temp_frac = *h_frac->ProfileY(" ",i,i);

     if(temp_frac.GetEntries() > 0)
          {
          g_frac_mean->SetPoint(ctr,h_frac->GetBinCenter(i),temp_frac.GetMean());
          g_frac_mean->SetPointError(ctr,0,temp_frac.GetRMS()/sqrt(temp_frac.GetEntries()));

          ctr++;
          }
     }
```

TF1 and Fit




```
TGraphErrors * g_frac_mean = new TgraphErrors();

TF1 * lin = new TF1("lin","[0]*x+[1]",-1000,1000);

g_frac_mean->Fit("lin","0");
lin->SetLineWidth(2);
lin->SetLineColor(2);
lin->Draw("same");

double p0 = lin->GetParameter(0);
double p1 = lin->GetParameter(1);
```

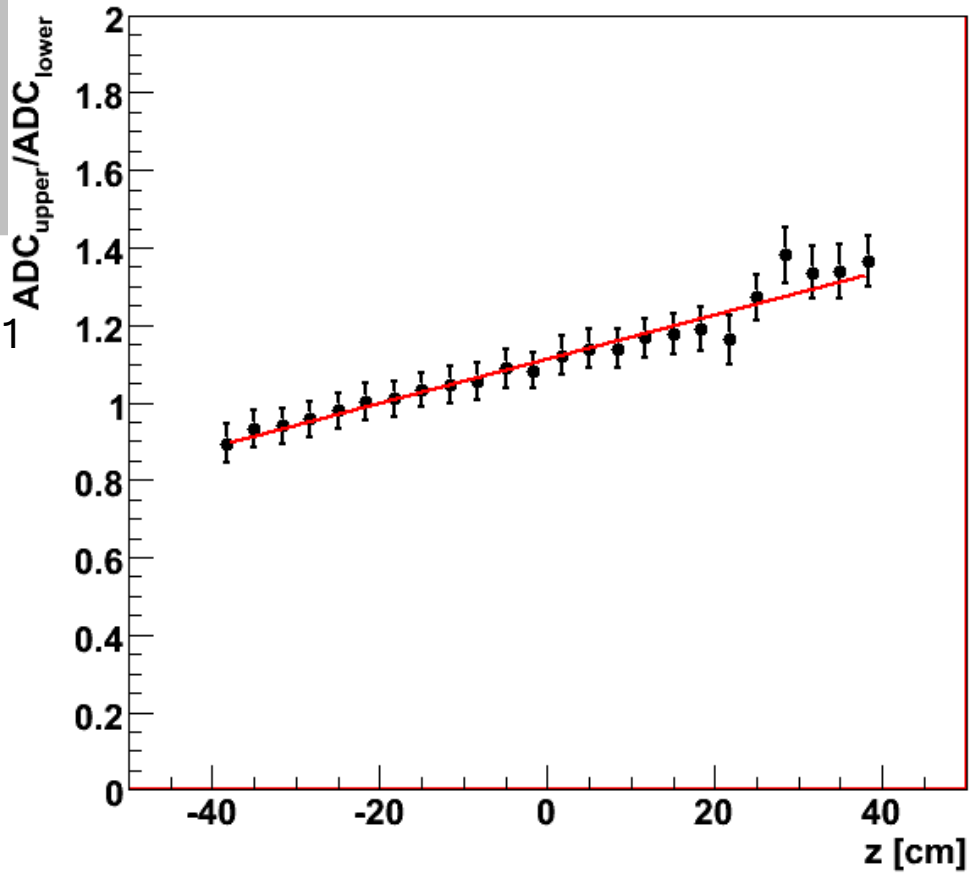
- fit histograms and graphs with arbitrary TF1 functions
- important options:
  "Q" Quiet mode (minimum printing)

  "V" Verbose mode

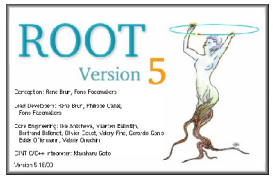
  "R" Use the Range specified in the function range

  "N" Do not store the function, do not draw

  "0" Do not plot the result of the fit.



TMath & TRandom



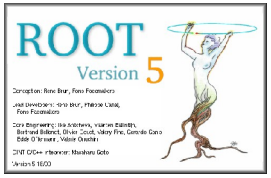
- TRandom: periodicity = 10^9 (fast)
  TRandom1: periodicity = 10^{171} (slow)
  TRandom2: periodicity = 10^{26} (fast)
  TRandom3: periodicity = 10^{6000} (fast)

```
TRandom3 * ran = new TRandom3();

ran->SetSeed(0); //0 = computer clock

ran->Uniform(1,10);
ran->Gaus(2,5);
```

- TMath: very useful namespace of mathematical tools, usage: (http://root.cern.ch/root/html/TMath.html)

```
int dof;
double chi2;
TMath::Prob(chi2,dof);
```

TText, TLatex, TPaveText




```
TText t;
t.SetTextSize(0.1);
t.SetTextAngle(20);
t.SetTextColor(4);
t.DrawText(0.1,0.2,"test 1 2 3");

TPaveText *p_acc_z_adc_norm_highest_bottom = new
TPaveText(0.6,0.2,0.9,0.6,"NDC");
p_acc_z_adc_norm_highest_bottom->AddText("bla");
p_acc_z_adc_norm_highest_bottom->AddText("bupp");
p_acc_z_adc_norm_highest_bottom->SetFillColor(0);
p_acc_z_adc_norm_highest_bottom->Draw();

TLatex latex;
latex.SetTextSize(0.08);
latex.SetTextAlign(13);
latex.DrawLatex(0.2,0.8,"x = #frac{y+z/2}
{y^{2}+1}");
```

- write text on your canvas!

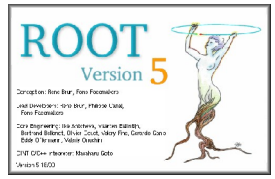
$$x = \frac{y+z/2}{y^2+1}$$

test 1 2 3

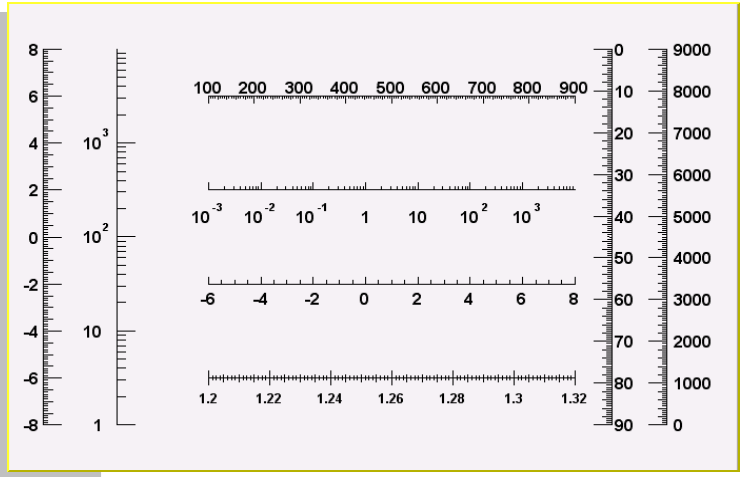
bla

bupp

Axis

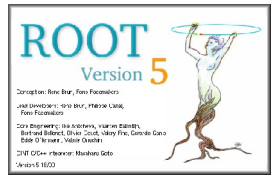



```
TGaxis *axis1 = new TGaxis(-4.5,-0.2,5.5,-0.2,-6,8,510,"");
axis1->SetName("axis1");
axis1->Draw();

TGaxis *axis2 = new TGaxis(-4.5,0.2,5.5,0.2,0.001,10000,510,"G");
axis2->SetName("axis2");
axis2->Draw();

TGaxis *axis3 = new TGaxis(-9,-0.8,-9,0.8,-8,8,50510,"");
axis3->SetName("axis3");
axis3->Draw();

TGaxis *axis4 = new TGaxis(-7,-0.8,-7,0.8,1,10000,50510,"G");
axis4->SetName("axis4");
axis4->Draw();

TGaxis *axis5 = new TGaxis(-4.5,-0.6,5.5,-0.6,1.2,1.32,80506,"-+");
axis5->SetName("axis5");
axis5->SetLabelSize(0.03);
axis5->SetTextFont(72);
axis5->SetLabelOffset(0.025);

axis5->Draw();

TGaxis *axis6 = new TGaxis(-4.5,0.6,5.5,0.6,100,900,50510,"-");
axis6->SetName("axis6");
axis6->Draw();

TGaxis *axis7 = new TGaxis(8,-0.8,8,0.8,0,9000,50510,"+L");
axis7->SetName("axis7");
axis7->SetLabelOffset(0.01);
axis7->Draw();

TGaxis *axis8 = new TGaxis(6.5,0.8,6.499,-0.8,0,90,50510,"-");
axis8->SetName("axis8");
axis8->Draw();
```

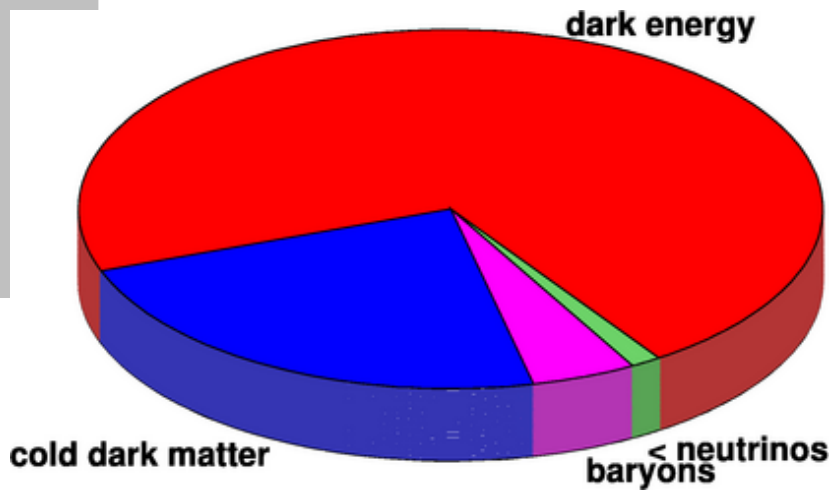


- various options for axis
- operations on TGaxis also apply for regular axis of histograms/graphs

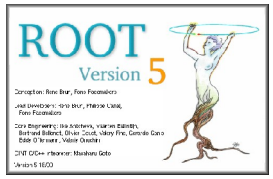
TPie




```
float vals[] = {0.721,0.233,0.0462,0.0133};
int colors[] = {2,4,6,8};
char label[4][200] ={"dark energy", "cold dark
matter", "baryons", "< neutrinos"};
int nvals = 4;

TCanvas *cpie = new TCanvas("cpie","TPie
test",700,700);

TPie *pie = new
TPie("pie","",nvals,vals,colors);

pie->SetAngularOffset(304);
pie->SetTextSize(0.04);
pie->GetSlice(0)->SetTitle("dark energy");
pie->GetSlice(1)->SetTitle("cold dark
matter");
pie->GetSlice(2)->SetTitle("baryons");
pie->GetSlice(3)->SetTitle("< neutrinos");
pie->Draw("3d <");
```



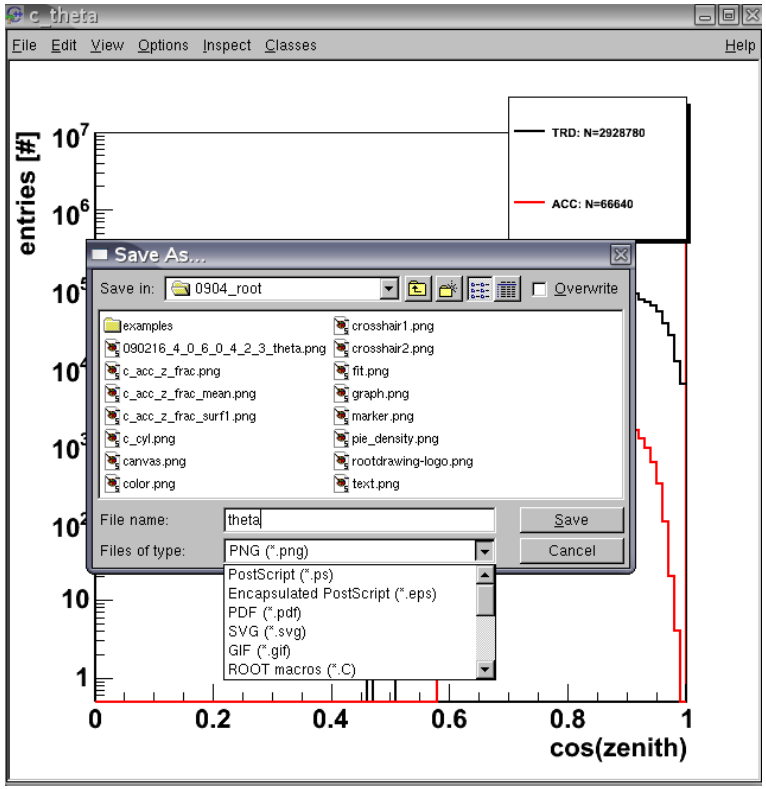
Read ASCII file




```
#include<fstream>
#include<iostream>
#include<TH2D.h>

void ascii(char *filename)
{
ifstream in(filename);
if(in.bad())
    {
    cout<<" "<<filename<<" does not exist!"<<endl;
    exit(-1);
    }

TH2D * h_frac = new TH2D("frac", "", 30,-50,50, 50,0,3);

double x, y;
while(1)
    {
    in>>x>>y;

    if(in.eof()) break;
    else
        {
        cout<<x<<" "<<y<<endl;
        h_frac->Fill(x,y);
        }
    }
in.close();
}
```

- read ascii file with two columns

Save TCanvas

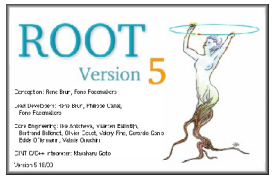


- save files interactively (Windows style): File->SaveAs:
  - ps, eps, pdf, jpg, png, gif...
- save files directly from program:

```
TCanvas * c = new TCanvas("c","",
200,10,600,600);
c->SaveAs("filename.eps");
c->SaveAs("filename.root");
```

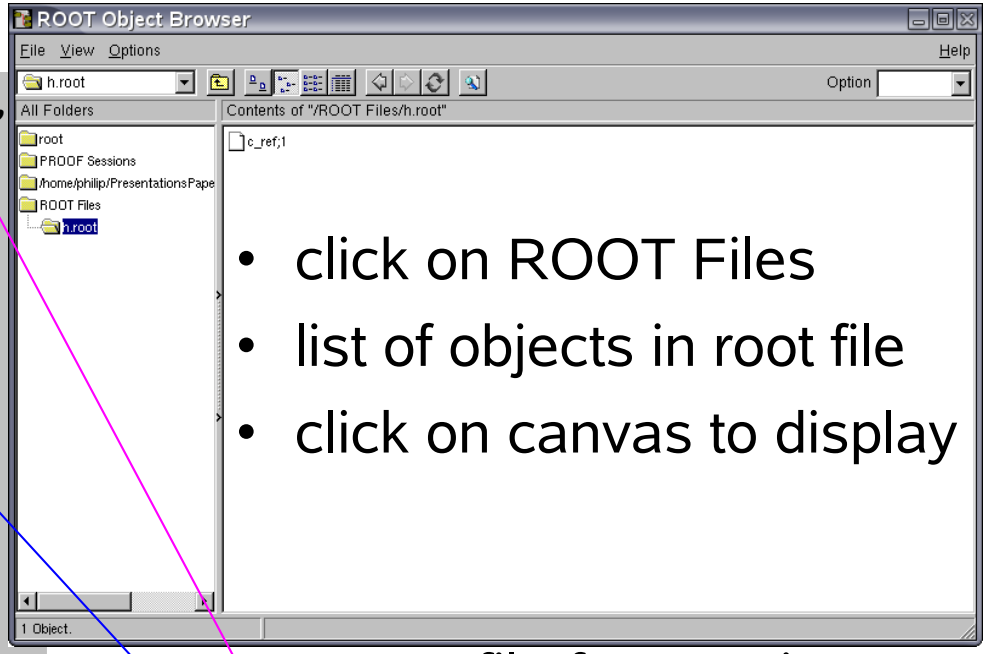
- **RECOMMENDATION**
  save canvas in eps (pdf) and root format in final nice looking layout:
  - eps for your thesis
  - root for later editing without running your analysis again
  - use reasonable directory/file structure

Open saved TCanvas/TBrowser



- save TCanvas with TH1D

```
TCanvas * c = new Tcanvas("c_ref",
"c_title", 200,10,600,600);

TH1D * h = new TH1D("histo", "",
100,0,1);

TF1 * f1 = new TF1("f1","Gaus(x,
[0],[1])",-1000,1000);
f1->SetParameters(0.5,0.1);

h->FillRandom("f1", 5000);

h->Draw();

c->SaveAs("h.eps");
c->SaveAs("h.root");
```



- click on ROOT Files
- list of objects in root file
- click on canvas to display

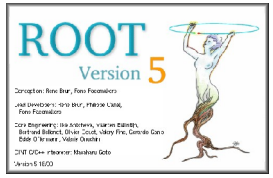
- open the root file interactively

```
>root -l h.root
[0]
Attaching file h.root as _file0...
[1]new TBrowser
```

- open root file from script

```
TFile * file = new TFile("h.root");

TCanvas * c1 = (TCanvas*)file
->Get("c_ref");

TH1D * h1 = (TH1D*)c1
->GetListOfPrimitives()
->FindObject("histo");

file->Close();
```

Save and load TTree



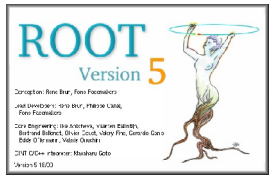

```
void save_tree()
{
TFile * f = new TFile("tree.root", "RECREATE");

TTree * t = new TTree("data","");

double x, y;
int event;

t->Branch("x", &x, "x/D");
t->Branch("y", &y, "y/D");
t->Branch("event", &event, "event/I");

TRandom3 * ran = new TRandom3();
ran->SetSeed(0);

for(int i = 0; i < 1000; i++)
     {
     x = ran->Gaus(0.5,3);
     y = ran->Exp(1);
     event = i;

     t->Fill();
     }

f->Write();
f->Close();
}
```

- save variables and all kind of inherited TObjects in root files

```
void open_tree()
{
TFile * f = new TFile("tree.root");

TTree * t = (TTree*)f->Get("data");

double x, y;
int event;

t->SetBranchAddress("x", &x);
t->SetBranchAddress("y", &y);
t->SetBranchAddress("event", &event);

for(Long64_t i = 0; i < t->GetEntriesFast(); i++)
     {
     t->GetEntry(i);
     cout<<event<<" "<<x<<" "<<y<<endl;
     }

f->Close();
}
```

```
TH1D * h = new TH1D("h","",100,0,1);
h->Write("h");
```

Standalone Programs



- C++ program:

```
#include<TApplication.h>
#include<TCanvas.h>
#include<TH1D.h>
#include<TF1.h>
#include<TStyle.h>
#include<TROOT.h>

int main(int argc, char * argv[])
{
int n = atoi(argv[1]);

TApplication a("a", &argc, argv);

gROOT->Reset();

TStyle * plain = new TStyle("plain","plain");
plain->SetCanvasBorderMode(0);
plain->SetPadBorderMode(0);
plain->SetPadColor(0);
plain->SetCanvasColor(0);
plain->SetTitleColor(1);
plain->SetStatColor(0);
plain->SetTitleFillColor(0);

gROOT->SetStyle("plain");

TCanvas * c = new Tcanvas("c_ref","c_title",
 200,10,600,600);
```

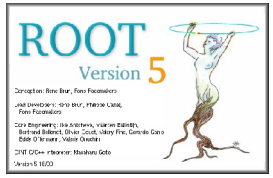
```
TH1D * h = new TH1D("histo", "", 100,0,1);

TF1 * f1 = new TF1("f1","Gaus(x,[0],[1])",-1000,1000);
f1->SetParameters(0.5,0.1);

h->FillRandom("f1", n);

h->Draw();

a.Run();

return 0;
}
```

- Makefile:

```
ROOTLIBS    = $(shell root-config --libs)

all:        standalone

standalone: standalone.o
            g++ -o standalone standalone.o $(ROOTLIBS)

.cpp.o:     $<.cpp
            g++ -I${ROOTSYS}/include -c $<

clean:
            @rm -f *.o
```

```
>make
>standalone 10000
```

Create new Class



- header file: class.hpp

```
#ifndef ROOTCLASS
#define ROOTCLASS

#include "TObject.h"

class root_class : public TObject
{
public:

root_class();
~root_class();

double get_var1();
int get_var2();

void set_var1(double var1);
void set_var2(int var2);

private:

double its_var1;
int its_var2;

ClassDef(root_class,1)
};

#endif
```

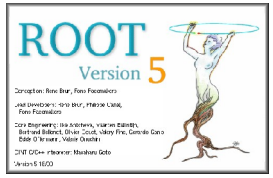
- function definitions: class.cpp

```
#include "class.hpp"

root_class::root_class()
{
its_var1 = 5.3;
its_var2 = 3;
}

root_class::~root_class(){;}

double root_class::get_var1(){return its_var1;}

int root_class::get_var2(){return its_var2;}

void root_class::set_var1(double var1){its_var1 =
var1;}

void root_class::set_var2(int var2){its_var2 = var2;}
```

- link new class to root: linkdef.hpp

```
#ifdef __CINT__

#pragma link C++ class root_class+;

#endif
```

Save and read new Class



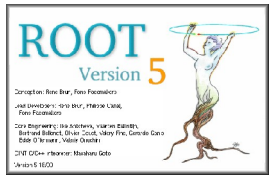
- Makefile

```
ROOTLIBS     = $(shell root-config --libs)

all:         class

class:       main_class.o class.o dict.o
(ROOTLIBS)   g++ -o class main_class.o class.o dict.o $

dict.cpp:    class.hpp class.cpp
             rootcint -f dict.cpp -c class.hpp linkdef.hpp


.cpp.o:      $<.cpp
             g++ -I${ROOTSYS}/include -c $<

clean:
             @rm -f *.o dict.h dict.cpp
```

- excerpt of main file: main_class.cpp

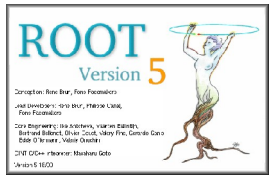
```
root_class * c1 = new root_class();

TFile * f1 = new TFile("root_class.root", "RECREATE");
TTree * t1 = new TTree("class1","");
t1->Branch("c1", &c1);

for(int i = 0; i < 1000; i++)
      {
      c1->set_var1(sqrt(double(i)));
      c1->set_var2(i-3);
      t1->Fill();
      }

f1->Write();
f1->Close();

root_class * c2 = new root_class();
TChain * t2 = new TChain("class1");
t2->SetBranchAddress("c1",&c2);
t2->Add("root_class.root");

for(Long64_t i = 0; i < t2->GetEntriesFast(); i++)
      {
      t2->GetEntry(i);
      cout<<c2->get_var1()<<" "<<c2->get_var2()<<endl;
      }
```

Conclusion



- most important web page: http://root.cern.ch with class descriptions, HowTo's, tutorials, help,...
- general concepts: canvas, histograms, graphs, fits
- IO file handling
- standalone programs
- this presentation and example files: http://accms04.physik.rwth-aachen.de/~doetinchem/0904_root_course.zip

Exercise



1. Setup compiled ROOT scripts
2. Try to fill histograms randomly and try different styles.
3. Try to draw graphs with and without errors.
4. Try to fit also complicated functions (not only linear and Gaussian...)
5. Save your files and open them again.
6. Try to save and read root files with trees (new classes and variables) and histograms