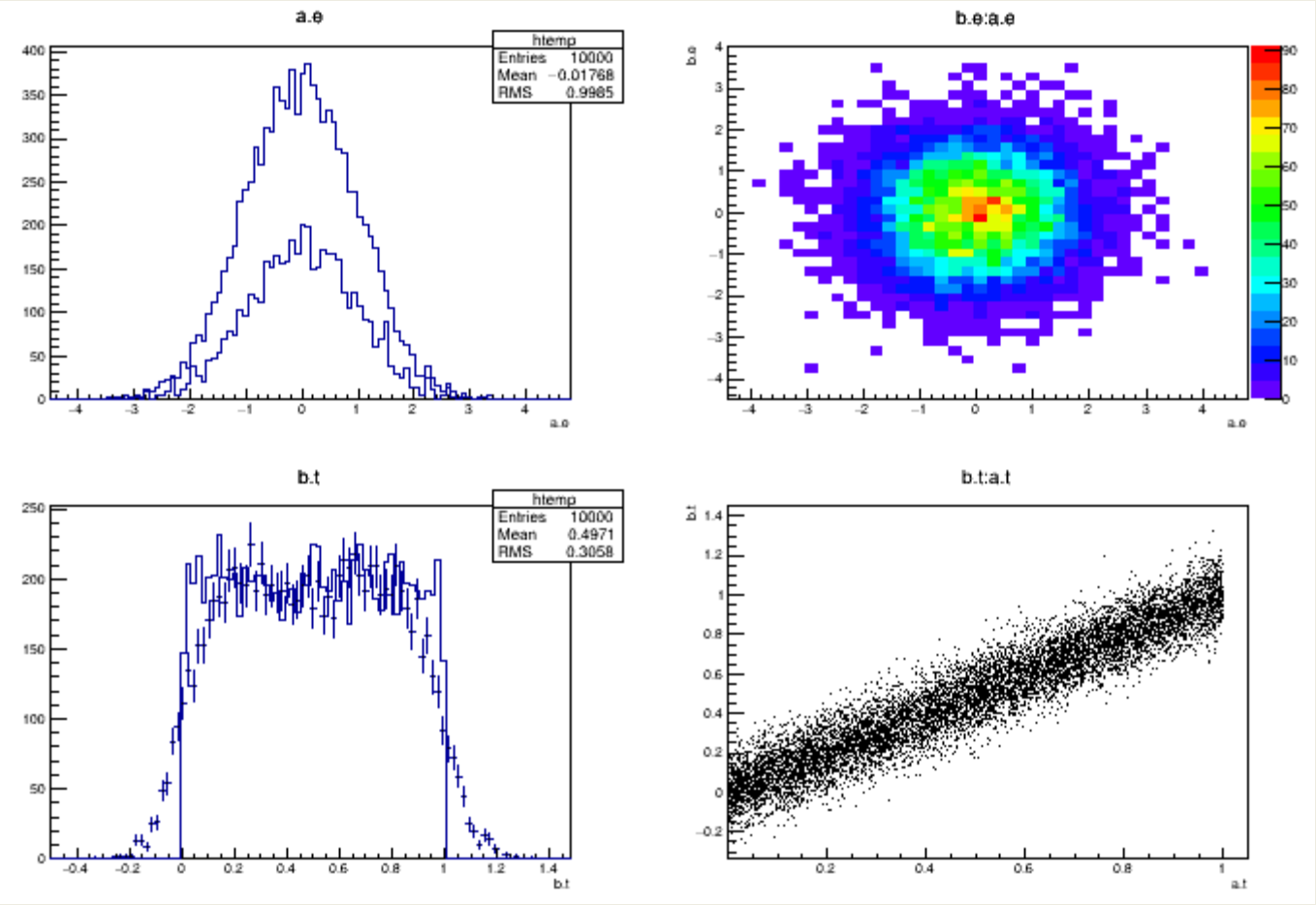
$ROOTSYS/tutorials/tree/tree0.C



siguang@pku.edu.cn

```
// Author: Heiko.Scheit@mpi-hd.mpg.de
//
//  simple Event class example
//
// execute as: .x tree0.C++
//
//  You have to copy it first to a directory where you have write access!
//  Note that .x tree0.C cannot work with this example
//
//
//////////////////////////////
//  Effect of ClassDef() and ClassImp() macros
//==========================================
//
// After running this macro create an instance of Det and Event
//
//   Det d;
//   Event e;
//
// now you can see the effect of the  ClassDef() and ClassImp() macros.
// (for the Det class these commands are commented!)
// For instance 'e' now knows who it is:
//   cout<<e.Class_Name()<<endl;
//
// whereas d does not.
//
// The methods that are added by the ClassDef()/Imp() marcro can be listed with
// .class
//   .class Event
//   .class Det
//////////////////
```

```
#include <TRandom.h>
#include <TTree.h>
#include <TCanvas.h>
#include <TStyle.h>

#include <Riostream.h>

//class Det  : public TObject  {
class Det {  // each detector gives an energy and time signal
public:
  Double_t e; //energy
  Double_t t; //time

//  ClassDef(Det,1)
};

//ClassImp(Det)
```

```
//ClassImp(Det)

//class Event { //TObject is not required by this example
class Event : public TObject {
public:

  Det a; // say there are two detectors (a and b) in the experiment
  Det b;
  ClassDef(Event,1)
};

ClassImp(Event)
```

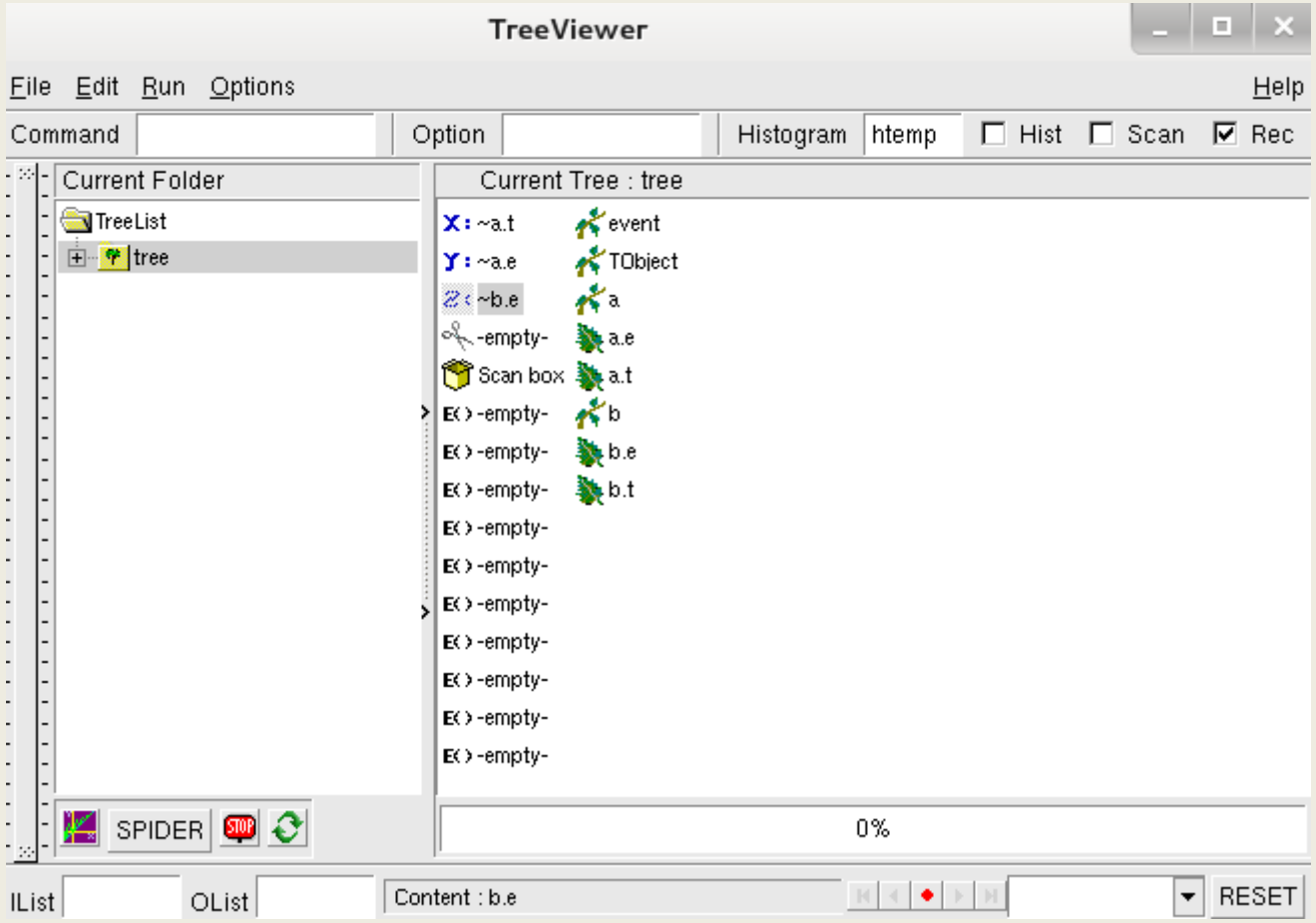
siguang@pku.edu.cn

```
void tree0() {
  // create a TTree
  TTree *tree = new TTree("tree","treelibrated tree");
  Event *e = new Event;

  // create a branch with energy
  tree->Branch("event",&e);

  // fill some events with random numbers
  Int_t nevent=10000;
  for (Int_t iev=0;iev<nevent;iev++) {
    if (iev%1000==0) cout<<"Processing event "<<iev<<"..."<<endl;

    Float_t ea,eb;
    gRandom->Rannor(ea,eb); // the two energies follow a gaus distribution
    e->a.e=ea;
    e->b.e=eb;
    e->a.t=gRandom->Rndm();  // random
    e->b.t=e->a.t + gRandom->Gaus(0.,.1);  // identical to a.t but a gaussian
                                           // 'resolution' was added with sigma .1
    tree->Fill();  // fill the tree with the current event
  }
```

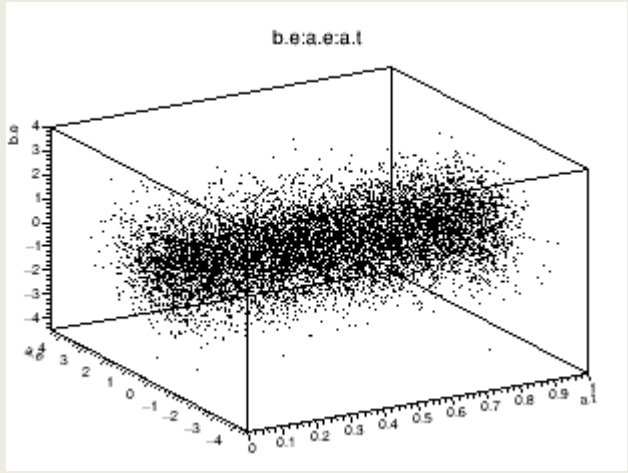
```
 // here you can investigate the structure of your Event class
 tree->StartViewer();

 //gROOT->SetStyle("Plain");   // uncomment to set a different style
 gStyle->SetPalette(1);        // use precomputed color palette 1

 // now draw some tree variables
 TCanvas *c1 = new TCanvas();
 c1->Divide(2,2);
 c1->cd(1);
 tree->Draw("a.e");  //energy of det a
 tree->Draw("a.e","3*(-.2<b.e && b.e<.2)","same");  // same but with condition on energy b; scaled by 3
 c1->cd(2);
 tree->Draw("b.e:a.e","","colz");        // one energy against the other
 c1->cd(3);
 tree->Draw("b.t","","e");    // time of b with errorbars
 tree->Draw("a.t","","same"); // overlay time of detector a
 c1->cd(4);
 tree->Draw("b.t:a.t");       // plot time b again time a
```

```
  cout<<endl;
  cout<<"You can now examine the structure of your tree in the TreeViewer"<<endl;
  cout<<endl;
}
```

TreeViewer的使用



将待显示的变量拖入xyz进行画图



siguang@pku.edu.cn

$ROOTSYS/tutorials/tree/tree1.C

```
#include "TROOT.h"
#include "TFile.h"
#include "TTree.h"
#include "TBrowser.h"
#include "TH2.h"
#include "TRandom.h"

// This example is a variant of hsimple.C but using a TTree instead
// of a TNtuple. It shows :
//   -how to fill a Tree with a few simple variables.
//   -how to read this Tree
//   -how to browse and analyze the Tree via the TBrowser and TTreeViewer
// This example can be run in many different ways:
//  way1:  .x tree1.C    using the CINT interpreter
//  way2:  .x tree1.C++  using the automatic compiler interface
//  way3:  .L tree1.C  or .L tree1.C++
//          tree1()
// One can also run the write and read parts in two separate sessions.
// For example following one of the sessions above, one can start the session:
//   .L tree1.C
//   tree1r();
//
//  Author: Rene Brun
```

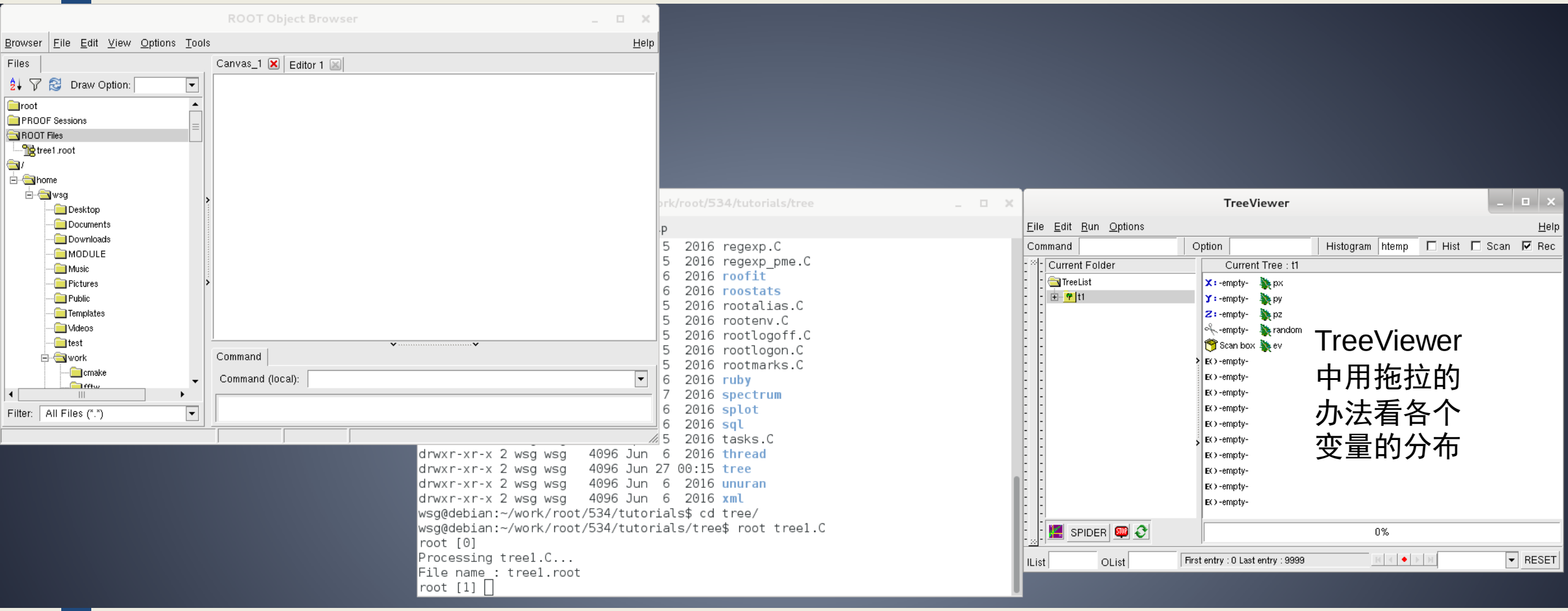
```
void tree1w()
{
   //create a Tree file tree1.root

   //create the file, the Tree and a few branches
   TFile f("tree1.root","recreate");
   TTree t1("t1","a simple Tree with simple variables");
   Float_t px, py, pz;
   Double_t random;
   Int_t ev;
   t1.Branch("px",&px,"px/F");
   t1.Branch("py",&py,"py/F");
   t1.Branch("pz",&pz,"pz/F");
   t1.Branch("random",&random,"random/D");
   t1.Branch("ev",&ev,"ev/I");

   //fill the tree
   for (Int_t i=0;i<10000;i++) {
     gRandom->Rannor(px,py);
     pz = px*px + py*py;
     random = gRandom->Rndm();
     ev = i;
    t1.Fill();
  }

  //save the Tree header. The file will be automatically closed
  //when going out of the function scope
  t1.Write();
}
```

siguang@pku.edu.cn

```
void tree1r()
{
   //read the Tree generated by tree1w and fill two histograms

   //note that we use "new" to create the TFile and TTree objects !
   //because we want to keep these objects alive when we leave this function.
   TFile *f = new TFile("tree1.root");
   TTree *t1 = (TTree*)f->Get("t1");
   Float_t px, py, pz;
   Double_t random;
   Int_t ev;
   t1->SetBranchAddress("px",&px);
   t1->SetBranchAddress("py",&py);
   t1->SetBranchAddress("pz",&pz);
   t1->SetBranchAddress("random",&random);
   t1->SetBranchAddress("ev",&ev);

   //create two histograms
   TH1F *hpx   = new TH1F("hpx","px distribution",100,-3,3);
   TH2F *hpxpy = new TH2F("hpxpy","py vs px",30,-3,3,30,-3,3);
```

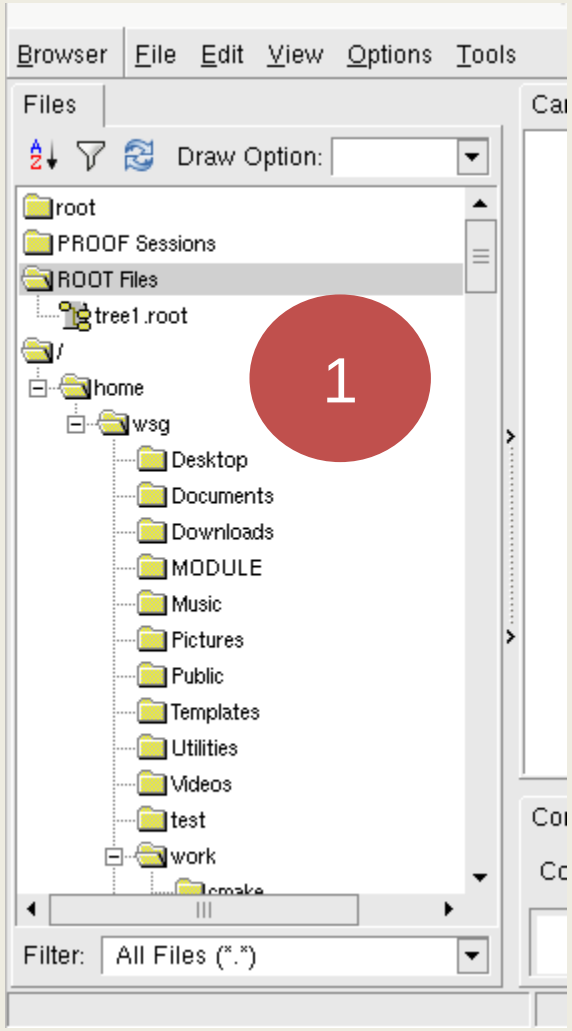
```
   //read all entries and fill the histograms
   Long64_t nentries = t1->GetEntries();
   for (Long64_t i=0;i<nentries;i++) {
     t1->GetEntry(i);
     hpx->Fill(px);
     hpxpy->Fill(px,py);
  }

  //we do not close the file. We want to keep the generated histograms
  //we open a browser and the TreeViewer
  if (gROOT->IsBatch()) return;
  new TBrowser();
  t1->StartViewer();
  // in the browser, click on "ROOT Files", then on "tree1.root".
  //     you can click on the histogram icons in the right panel to draw them.
  // in the TreeViewer, follow the instructions in the Help button.
}

void tree1() {
   tree1w();
   tree1r();
}
```

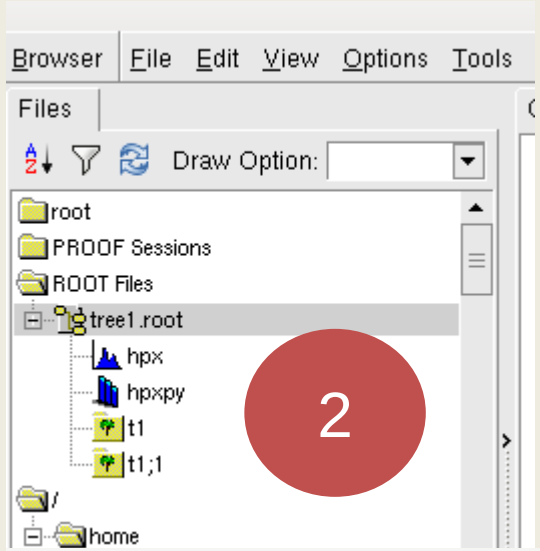
运行后分别弹出TBrowser和TreeViewer

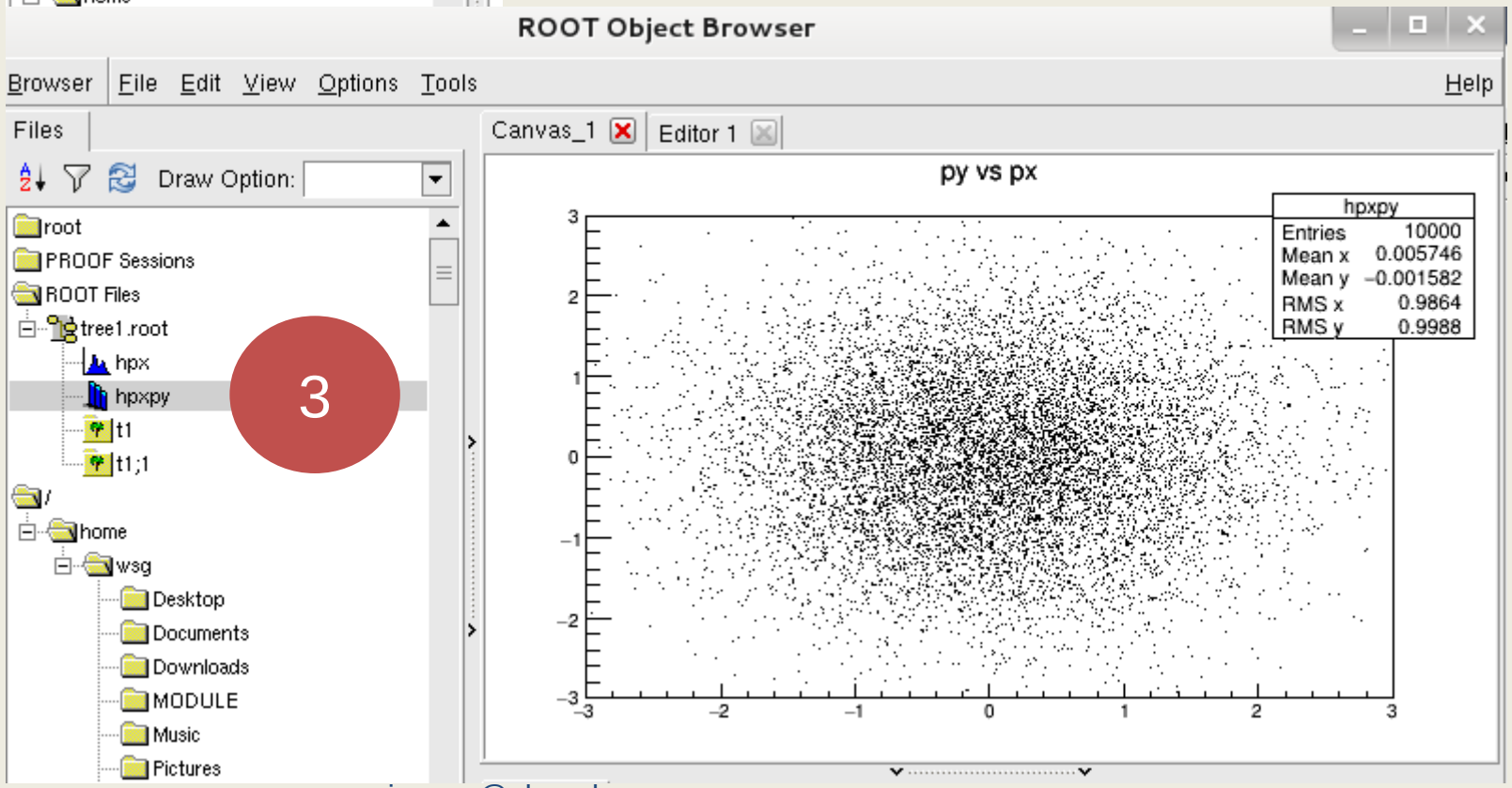


TreeViewer中用拖拉的办法看各个变量的分布

new TBrowser()结果：

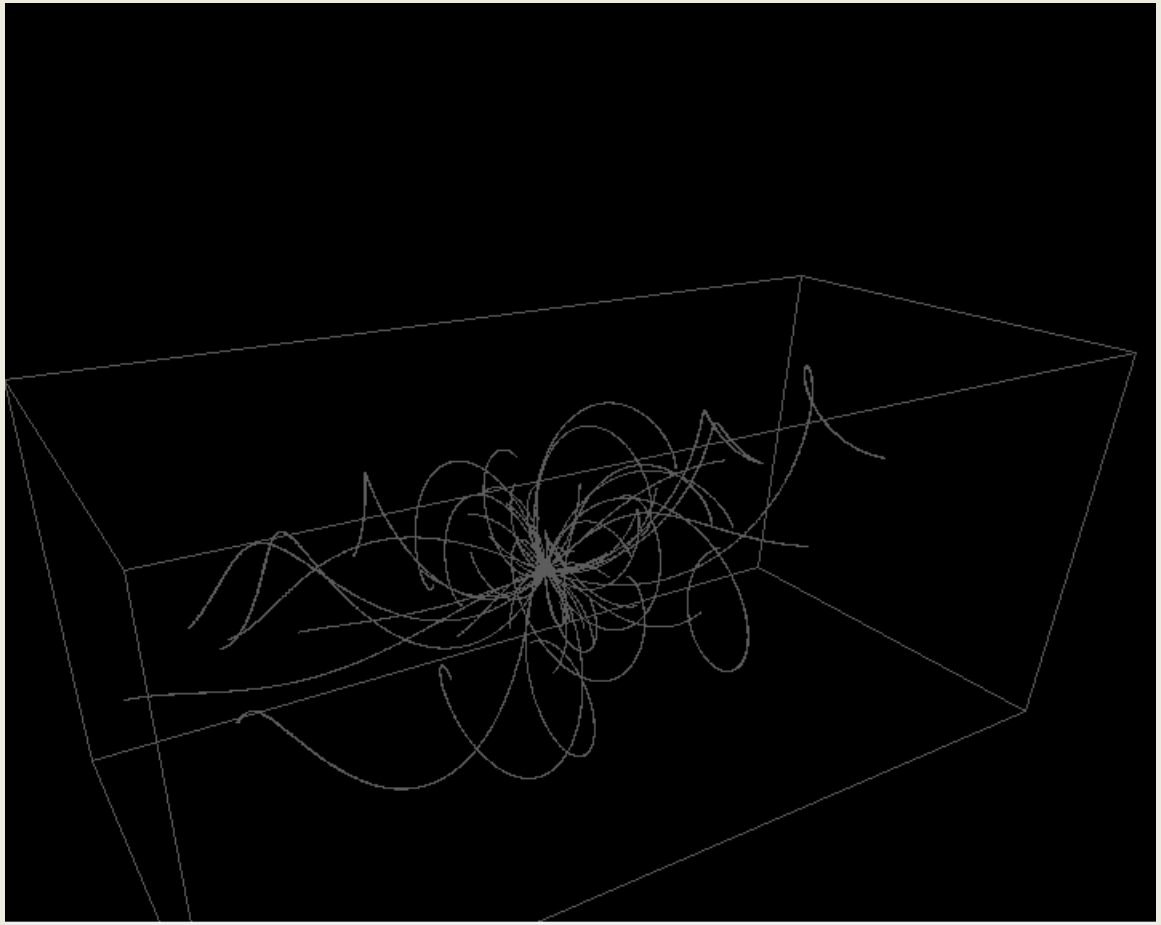


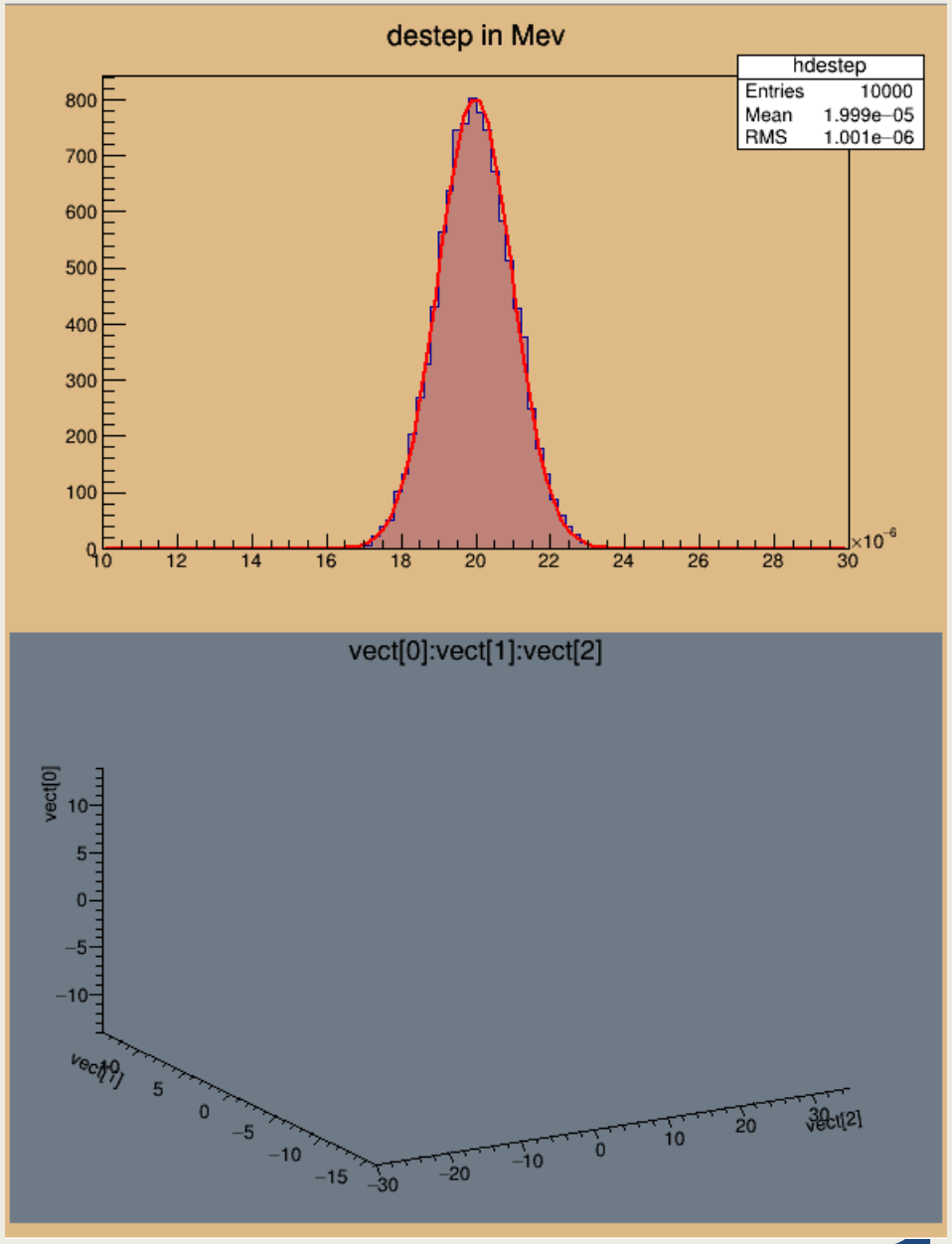




1→2 → 3依次展开

$ROOTSYS/tutorials/tree/tree2.C





siguang@pku.edu.cn

```
#include "TFile.h"
#include "TTree.h"
#include "TBrowser.h"
#include "TH2.h"
#include "TRandom.h"
#include "TCanvas.h"
#include "TMath.h"
#include "TROOT.h"

// This example illustrates how to make a Tree from variables or arrays
// in a C struct. Use of C structs is strongly discouraged and one should
// use classes instead. However support for C structs is important for
// legacy applications written in C or Fortran.
//    see tree2a.C for the same example using a class instead of a C-struct.
//
// In this example, we are mapping a C struct to one of the Geant3
// common blocks /gctrak/. In the real life, this common will be filled
// by Geant3 at each step and only the Tree Fill function should be called.
// The example emulates the Geant3 step routines.
//
// to run the example, do:
// .x tree2.C   to execute with the CINT interpreter
// .x tree2.C++ to execute with native compiler
//
//  Author: Rene Brun
```

```
const Int_t MAXMEC = 30;
//       PARAMETER (MAXMEC=30)
//       COMMON/GCTRAK/VECT(7),GETOT,GEKIN,VOUT(7),NMEC,LMEC(MAXMEC)
//      + ,NAMEC(MAXMEC),NSTEP ,PID,DESTEP,DESTEL,SAFETY,SLENG
//      + ,STEP  ,SNEXT ,SFIELD,TOFG  ,GEKRAT,UPWGHT
typedef struct {
  Float_t  vect[7];
  Float_t  getot;
  Float_t  gekin;
  Float_t  vout[7];
  Int_t    nmec;
  Int_t    lmec[MAXMEC];
  Int_t    namec[MAXMEC];
  Int_t    nstep;
  Int_t    pid;
  Float_t  destep;
  Float_t  destel;
  Float_t  safety;
  Float_t  sleng;
  Float_t  step;
  Float_t  snext;
  Float_t  sfield;
  Float_t  tofg;
  Float_t  gekrat;
  Float_t  upwght;
} Gctrak_t;
```

siguang@pku.edu.cn

```
void helixStep(Float_t step, Float_t *vect, Float_t *vout)
{
  // extrapolate track in constant field
   Float_t field = 20;       //magnetic field in kilogauss
   enum Evect {kX,kY,kZ,kPX,kPY,kPZ,kPP};
   vout[kPP] = vect[kPP];
   Float_t h4    = field*2.99792e-4;
   Float_t rho   = -h4/vect[kPP];
   Float_t tet   = rho*step;
   Float_t tsint = tet*tet/6;
   Float_t sintt = 1 - tsint;
   Float_t sint  = tet*sintt;
   Float_t cos1t = tet/2;
   Float_t f1 = step*sintt;
   Float_t f2 = step*cos1t;
   Float_t f3 = step*tsint*vect[kPZ];
   Float_t f4 = -tet*cos1t;
   Float_t f5 = sint;
   Float_t f6 = tet*cos1t*vect[kPZ];
   vout[kX]   = vect[kX]  + (f1*vect[kPX] - f2*vect[kPY]);
   vout[kY]   = vect[kY]  + (f1*vect[kPY] + f2*vect[kPX]);
   vout[kZ]   = vect[kZ]  + (f1*vect[kPZ] + f3);
   vout[kPX]  = vect[kPX] + (f4*vect[kPX] - f5*vect[kPY]);
   vout[kPY]  = vect[kPY] + (f4*vect[kPY] + f5*vect[kPX]);
   vout[kPZ]  = vect[kPZ] + (f4*vect[kPZ] + f6);
}
```

siguang@pku.edu.cn

```
void tree2w()
{
   //create a Tree file tree2.root

   //create the file, the Tree and a few branches with
   //a subset of gctrak
   TFile f("tree2.root","recreate");
   TTree t2("t2","a Tree with data from a fake Geant3");
   Gctrak_t gstep;
   t2.Branch("vect",gstep.vect,"vect[7]/F");
   t2.Branch("getot",&gstep.getot,"getot/F");
   t2.Branch("gekin",&gstep.gekin,"gekin/F");
   t2.Branch("nmec",&gstep.nmec,"nmec/I");
   t2.Branch("lmec",gstep.lmec,"lmec[nmec]/I");
   t2.Branch("destep",&gstep.destep,"destep/F");
   t2.Branch("pid",&gstep.pid,"pid/I");

   //Initialize particle parameters at first point
   Float_t px,py,pz,p,charge=0;
   Float_t vout[7];
   Float_t mass  = 0.137;
   Bool_t newParticle = kTRUE;
   gstep.step    = 0.1;
   gstep.destep  = 0;
   gstep.nmec    = 0;
   gstep.pid     = 0;
```

siguang@pku.edu.cn

```
//transport particles
for (Int_t i=0;i<10000;i++) {
   //generate a new particle if necessary
   if (newParticle) {
      px = gRandom->Gaus(0,.02);
      py = gRandom->Gaus(0,.02);
      pz = gRandom->Gaus(0,.02);
      p  = TMath::Sqrt(px*px+py*py+pz*pz);
      charge = 1; if (gRandom->Rndm() < 0.5) charge = -1;
      gstep.pid    += 1;
      gstep.vect[0] = 0;
      gstep.vect[1] = 0;
      gstep.vect[2] = 0;
      gstep.vect[3] = px/p;
      gstep.vect[4] = py/p;
      gstep.vect[5] = pz/p;
      gstep.vect[6] = p*charge;
      gstep.getot   = TMath::Sqrt(p*p + mass*mass);
      gstep.gekin   = gstep.getot - mass;
      newParticle = kFALSE;
   }

   // fill the Tree with current step parameters
   t2.Fill();

   //transport particle in magnetic field
   helixStep(gstep.step, gstep.vect, vout); //make one step
```

siguang@pku.edu.cn

```
      //apply energy loss
      gstep.destep = gstep.step*gRandom->Gaus(0.0002,0.00001);
      gstep.gekin -= gstep.destep;
      gstep.getot  = gstep.gekin + mass;
      gstep.vect[6] = charge*TMath::Sqrt(gstep.getot*gstep.getot - mass*mass);
      gstep.vect[0] = vout[0];
      gstep.vect[1] = vout[1];
      gstep.vect[2] = vout[2];
      gstep.vect[3] = vout[3];
      gstep.vect[4] = vout[4];
      gstep.vect[5] = vout[5];
      gstep.nmec    = (Int_t)(5*gRandom->Rndm());
      for (Int_t l=0;l<gstep.nmec;l++) gstep.lmec[l] = l;
      if (gstep.gekin < 0.001)              newParticle = kTRUE;
      if (TMath::Abs(gstep.vect[2]) > 30) newParticle = kTRUE;
   }

   //save the Tree header. The file will be automatically closed
   //when going out of the function scope
   t2.Write();
}
```

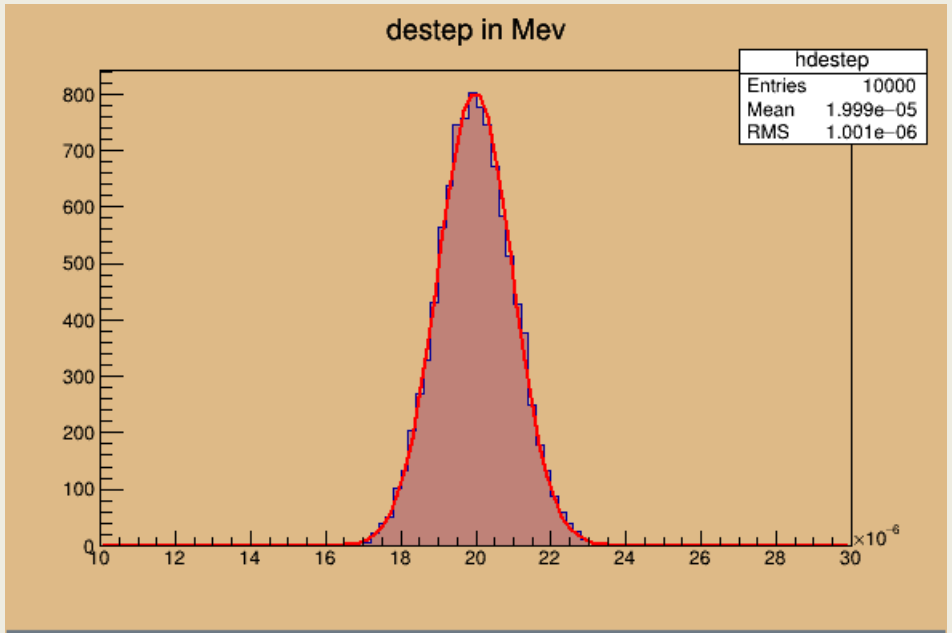
siguang@pku.edu.cn

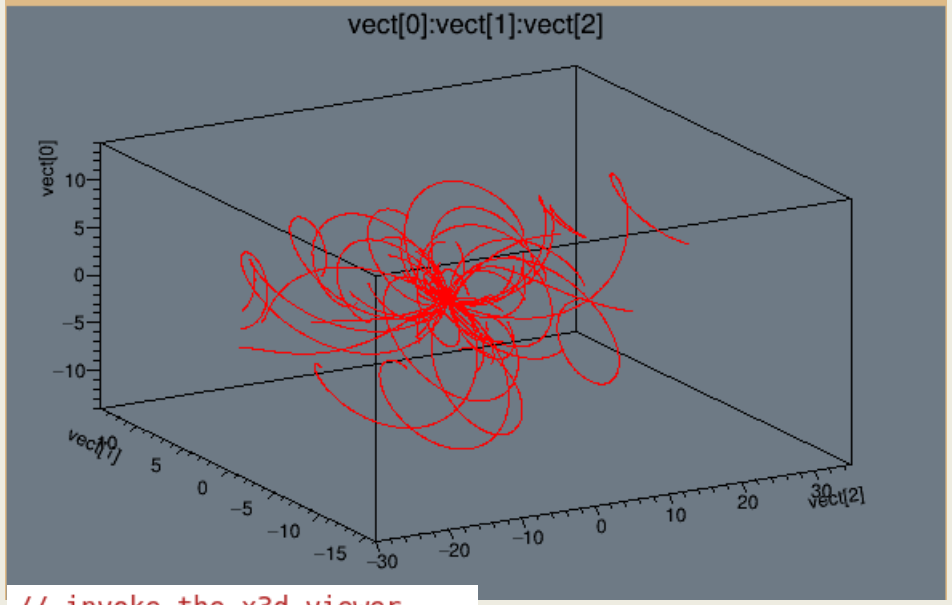
```
void tree2r()
{
   //read the Tree generated by tree2w and fill one histogram
   //we are only interested by the destep branch.

   //note that we use "new" to create the TFile and TTree objects !
   //because we want to keep these objects alive when we leave
   //this function.
   TFile *f = new TFile("tree2.root");
   TTree *t2 = (TTree*)f->Get("t2");
   static Float_t destep;
   TBranch *b_destep = t2->GetBranch("destep");
   b_destep->SetAddress(&destep);

   //create one histogram
   TH1F *hdestep   = new TH1F("hdestep","destep in Mev",100,1e-5,3e-5);

   //read only the destep branch for all entries
   Long64_t nentries = t2->GetEntries();
   for (Long64_t i=0;i<nentries;i++) {
      b_destep->GetEntry(i);
      hdestep->Fill(destep);
   }
```

可用：t2->SetBranchAddress(“destep”,&destep);
代替左边两句命令

可用：t2->GetEntry(i)

```
   //we do not close the file.
   //We want to keep the generated histograms
   //We fill a 3-d scatter plot with the particle step coordinates
   TCanvas *c1 = new TCanvas("c1","c1",600,800);
   c1->SetFillColor(42);
   c1->Divide(1,2);
   c1->cd(1);
   hdestep->SetFillColor(45);
   hdestep->Fit("gaus");
   c1->cd(2);
   gPad->SetFillColor(37);
   t2->SetMarkerColor(kRed);
   t2->Draw("vect[0]:vect[1]:vect[2]");
   if (gROOT->IsBatch()) return;

   // invoke the x3d viewer
   gPad->GetViewer3D("x3d");
}

void tree2() {
   tree2w();
   tree2r();
}
```






```
// invoke the x3d viewer
// gPad->GetViewer3D("x3d");
```

siguang@pku.edu.cn

$ROOTSYS/tutorials/tree/tree2a.C

```
class Gctrak : public TObject {
public:
  Float_t  vect[7];
  Float_t  getot;
  Float_t  gekin;
  Float_t  vout[7];    //! not persistent
  Int_t    nmec;
  Int_t   *lmec;       //[nmec]
  Int_t   *namec;      //[nmec]
  Int_t    nstep;      //! not persistent
  Int_t    pid;
  Float_t  destep;
  Float_t  destel;     //! not persistent
  Float_t  safety;     //! not persistent
  Float_t  sleng;      //! not persistent
  Float_t  step;       //! not persistent
  Float_t  snext;      //! not persistent
  Float_t  sfield;     //! not persistent
  Float_t  tofg;       //! not persistent
  Float_t  gekrat;     //! not persistent
  Float_t  upwght;     //! not persistent

  Gctrak() {lmec=0; namec=0;}

  ClassDef(Gctrak,1)
};
```

效果与tree2.C同；
用class保存；
需编译运行：
root tree2a.C++
//! 的作用是该变量不被保持在root文件中

写：

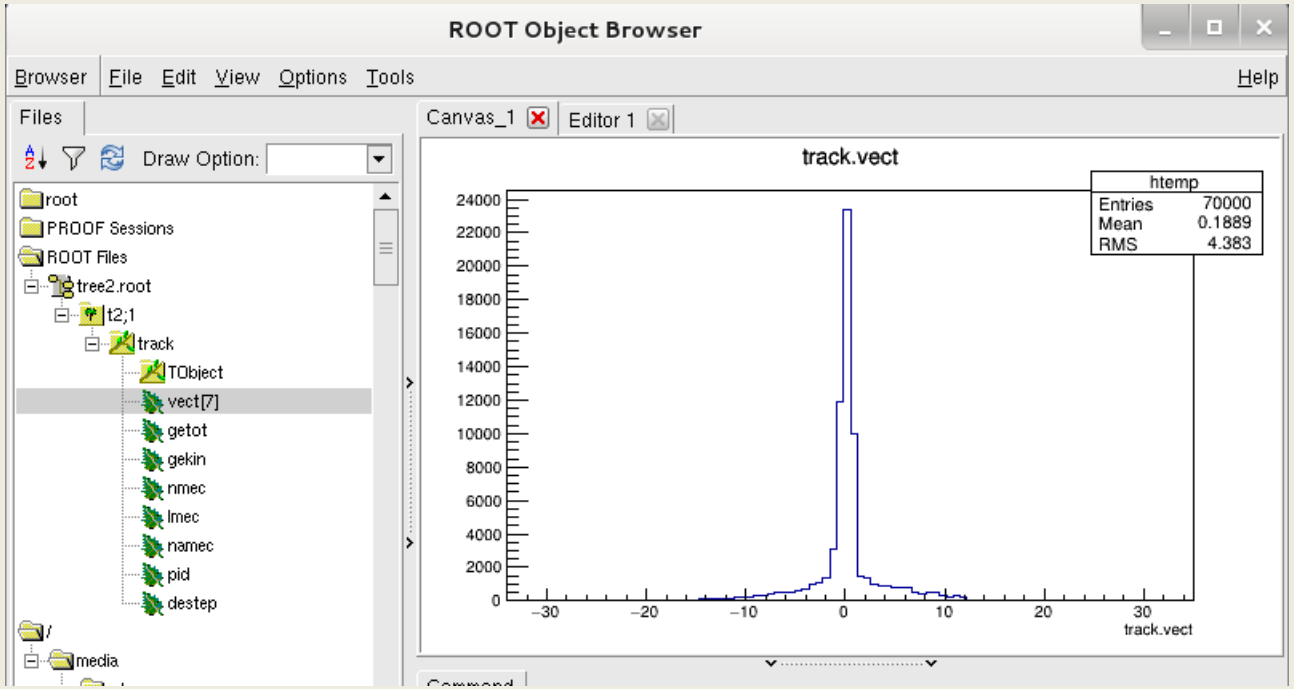
```
//create the file, the Tree and a few branches with
//a subset of gctrak
TFile f("tree2.root","recreate");
TTree t2("t2","a Tree with data from a fake Geant3");
Gctrak *gstep = new Gctrak;
t2.Branch("track",&gstep,8000,1);
```

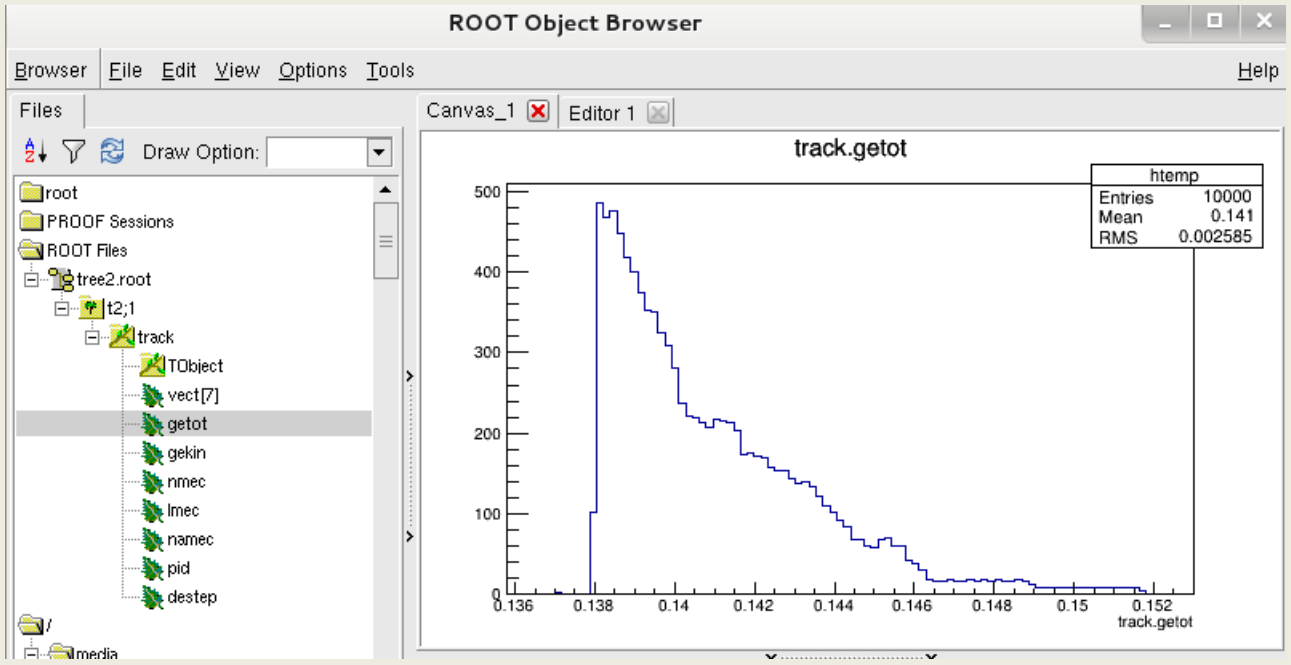
Branch(branchname, &p_object, bufsize, splitlevel)

读：

```
TFile *f = new TFile("tree2.root");
TTree *t2 = (TTree*)f->Get("t2");
Gctrak *gstep = 0;
t2->SetBranchAddress("track",&gstep);
TBranch *b_destep = t2->GetBranch("destep");
```

siguang@pku.edu.cn






```
class Gctrak : public TObject {
public:
  Float_t  vect[7];
  Float_t  getot;
  Float_t  gekin;
  Float_t  vout[7];    //! not persistent
  Int_t    nmec;
  Int_t   *lmec;       //[nmec]
  Int_t   *namec;      //[nmec]
  Int_t    nstep;      //! not persistent
  Int_t    pid;
  Float_t  destep;
  Float_t  destel;     //! not persistent
  Float_t  safety;     //! not persistent
  Float_t  sleng;      //! not persistent
  Float_t  step;       //! not persistent
  Float_t  snext;      //! not persistent
  Float_t  sfield;     //! not persistent
  Float_t  tofg;       //! not persistent
  Float_t  gekrat;     //! not persistent
  Float_t  upwght;     //! not persistent

  Gctrak() {lmec=0; namec=0;}

  ClassDef(Gctrak,1)
};
```

siguang@pku.edu.cn

$ROOTSYS/tutorials/tree/tree3.C

展示：保存长度可变化的数组；友Tree的使用

siguang@pku.edu.cn

```
#include "TFile.h"
#include "TTree.h"
#include "TRandom.h"
#include "TCanvas.h"

void tree3w() {
// Example of a Tree where branches are variable length arrays
// A second Tree is created and filled in parallel.
// Run this script with
//    .x tree3.C
// In the function treer, the first Tree is open.
// The second Tree is declared friend of the first tree.
// TTree::Draw is called with variables from both Trees.
//
//  Author: Rene Brun

   const Int_t kMaxTrack = 500;
   Int_t ntrack;
   Int_t stat[kMaxTrack];
   Int_t sign[kMaxTrack];
   Float_t px[kMaxTrack];
   Float_t py[kMaxTrack];
   Float_t pz[kMaxTrack];
   Float_t pt[kMaxTrack];
   Float_t zv[kMaxTrack];
   Float_t chi2[kMaxTrack];
   Double_t sumstat;
```

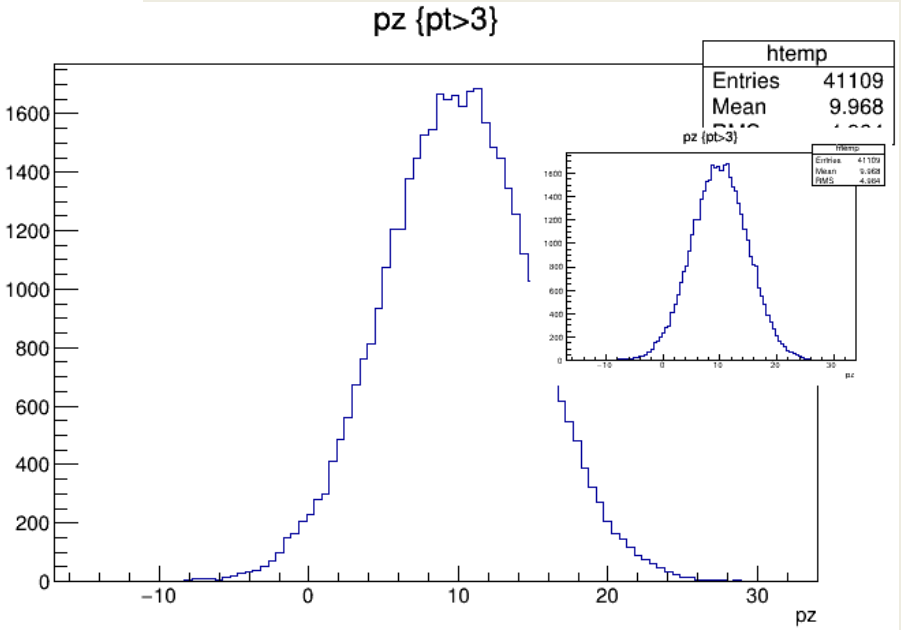
```
TFile f("tree3.root","recreate");
TTree *t3 = new TTree("t3","Reconst ntuple");
t3->Branch("ntrack",&ntrack,"ntrack/I");
t3->Branch("stat",stat,"stat[ntrack]/I");
t3->Branch("sign",sign,"sign[ntrack]/I");
t3->Branch("px",px,"px[ntrack]/F");
t3->Branch("py",py,"py[ntrack]/F");
t3->Branch("pz",pz,"pz[ntrack]/F");
t3->Branch("zv",zv,"zv[ntrack]/F");
t3->Branch("chi2",chi2,"chi2[ntrack]/F");

TFile fr("tree3f.root","recreate");
TTree *t3f = new TTree("t3f","a friend Tree");
t3f->Branch("ntrack",&ntrack,"ntrack/I");
t3f->Branch("sumstat",&sumstat,"sumstat/D");
t3f->Branch("pt",pt,"pt[ntrack]/F");
```

```
   for (Int_t i=0;i<1000;i++) {
      Int_t nt = gRandom->Rndm()*(kMaxTrack-1);
      ntrack = nt;
      sumstat = 0;
      for (Int_t n=0;n<nt;n++) {
         stat[n] = n%3;
         sign[n] = i%2;
         px[n]   = gRandom->Gaus(0,1);
         py[n]   = gRandom->Gaus(0,2);
         pz[n]   = gRandom->Gaus(10,5);
         zv[n]   = gRandom->Gaus(100,2);
         chi2[n] = gRandom->Gaus(0,.01);
         sumstat += chi2[n];
         pt[n]   = TMath::Sqrt(px[n]*px[n] + py[n]*py[n]);
      }
      t3->Fill();
      t3f->Fill();
   }
   t3->Print();
   f.cd();
   t3->Write();
   fr.cd();
   t3f->Write();
}
```

siguang@pku.edu.cn

```
void tree3r()
{
   TFile *f = new TFile("tree3.root");
   TTree *t3 = (TTree*)f->Get("t3");
   t3->AddFriend("t3f","tree3f.root");
   t3->Draw("pz","pt>3");
}

void tree3r2()
{
   TPad *p = new TPad("p","p",0.6, 0.4, 0.98, 0.8);
   p->Draw(); p->cd();
   TFile *f1 = new TFile("tree3.root");
   TFile *f2 = new TFile("tree3f.root");
   TTree *t3 = (TTree*)f1->Get("t3");
   t3->AddFriend("t3f",f2);
   t3->Draw("pz","pt>3");
}

void tree3()
{
   tree3w();
   tree3r();
   tree3r2();
}
```

注意：pz是t3的但pt是t3f的Branch！但因两Tree是好友，可干涉内政！



siguang@pku.edu.cn

$ROOTSYS/tutorials/tree/tree4.C

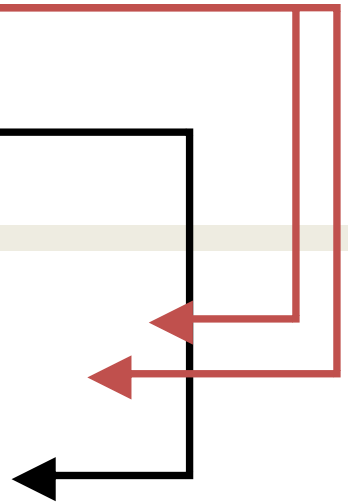
运行前：
在$ROOTSYS/test 目录下运行make Event生成libEvent.so

```
wsg@debian:~/work/root/534/test$ make Event
c++  -O2 -Wall -fPIC -pthread -m64 -I/home/wsg/work/root/534/include -c Event.cx
x
Generating dictionary EventDict.cxx...
rootcint -f EventDict.cxx -c Event.h EventLinkDef.h
c++  -O2 -Wall -fPIC -pthread -m64 -I/home/wsg/work/root/534/include -c EventDic
t.cxx
c++ -shared -O2 -m64 Event.o EventDict.o -o  libEvent.so
libEvent.so done
c++  -O2 -Wall -fPIC -pthread -m64 -I/home/wsg/work/root/534/include -c MainEven
t.cxx
c++ -O2 -m64 MainEvent.o Event.o EventDict.o -L/home/wsg/work/root/534/lib -lCor
e -lCint -lRIO -lNet -lHist -lGraf -lGraf3d -lGpad -lTree -lRint -lPostscript -l
Matrix -lPhysics -lMathCore -lThread -pthread -lm -ldl -rdynamic  -o Event
Event done
wsg@debian:~/work/root/534/test$ ll libEvent.so
-rwxr-xr-x 1 wsg wsg 149920 Jul  3 19:59 libEvent.so
wsg@debian:~/work/root/534/test$
```

siguang@pku.edu.cn

```
#include "TFile.h"
#include "TTree.h"
#include "TBrowser.h"
#include "TH2.h"
#include "TRandom.h"
#include "TClassTable.h"
#include "TSystem.h"
#include "TROOT.h"
#if defined(__CINT__) && !defined(__MAKECINT__)
#include "../test/libEvent.so"
#else
#include "../test/Event.h"
#endif
```

```
// This example can be run in many different ways:
//  way1:   .x tree4.C    using the CINT interpreter
//  way2:   .L tree4.C
//           tree4()
//  way3:   .L ../test/libEvent.so
//          .x tree4.C++   using ACLIC
// One can also run the write and read parts in two separate sessions.
// For example following one of the sessions above, one can start the session:
//   .L tree4.C
//   tree4r();
```

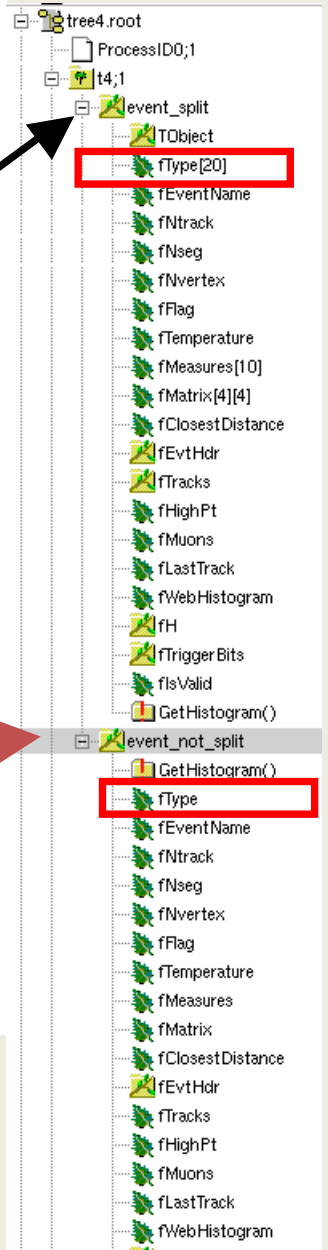


```
// This example writes a tree with objects of the class Event.
// It is a simplified version of $ROOTSYS/test/MainEvent.cxx to
// write the tree, and $ROOTSYS/test/eventb.C
// It shows:
//   -how to fill a Tree with an event class containing these
//    data members:
//     char            fType[20];
//     Int_t           fNtrack;
//     Int_t           fNseg;
//     Int_t           fNvertex;
//     UInt_t          fFlag;
//     Float_t         fTemperature;
//     EventHeader     fEvtHdr;
//     TClonesArray   *fTracks;            //->
//     TH1F           *fH;                 //->
//     Int_t           fMeasures[10];
//     Float_t         fMatrix[4][4];
//     Float_t        *fClosestDistance;   //[fNvertex]
//
//   -the difference in splitting or not splitting a branch
//   -how to read selected branches of the tree,
//    and print the first entry with less than 587 tracks.
//   -how to browse and analyze the Tree via the TBrowser and TTreeViewer
```

//->假设指针有效，避免指向同类循环检查

//[fNvertex]给定数组长度，fNvertex必须先定义的整数

```
void tree4w()
{

  //create a Tree file tree4.root
  TFile f("tree4.root","RECREATE");

  // Create a ROOT Tree
  TTree t4("t4","A Tree with Events");

  // Create a pointer to an Event object
  Event *event = new Event();

  // Create two branches, split one.
  t4.Branch("event_split", &event,16000,99);
  t4.Branch("event_not_split", &event,16000,0);

  // a local variable for the event type
  char etype[20];
```



siguang@pku.edu.cn

```
// a local variable for the event type
char etype[20];

// Fill the tree
for (Int_t ev = 0; ev <100; ev++) {
  Float_t sigmat, sigmas;
  gRandom->Rannor(sigmat,sigmas);
  Int_t ntrack   = Int_t(600 + 600 *sigmat/120.);
  Float_t random = gRandom->Rndm(1);
  sprintf(etype,"type%d",ev%5);
  event->SetType(etype);
  event->SetHeader(ev, 200, 960312, random);
  event->SetNseg(Int_t(10*ntrack+20*sigmas));
  event->SetNvertex(Int_t(1+20*gRandom->Rndm()));
  event->SetFlag(UInt_t(random+0.5));
  event->SetTemperature(random+20.);

  for(UChar_t m = 0; m < 10; m++) {
    event->SetMeasure(m, Int_t(gRandom->Gaus(m,m+1)));
  }

  // fill the matrix
  for(UChar_t i0 = 0; i0 < 4; i0++) {
    for(UChar_t i1 = 0; i1 < 4; i1++) {
      event->SetMatrix(i0,i1,gRandom->Gaus(i0*i1,1));
    }
  }
```

```
      //  Create and fill the Track objects
      for (Int_t t = 0; t < ntrack; t++) event->AddTrack(random);

      // Fill the tree
      t4.Fill();

      // Clear the event before reloading it
      event->Clear();
   }

   // Write the file header
   f.Write();

   // Print the tree contents
   t4.Print();
}
```

t4.Print()的结果

```
wsg@debian:~/work/root/534/tutorials/tree$ root tree4.C
root [0]
Processing tree4.C...
*******************************************************************************
*Tree    :t4        : A Tree with Events                                     *
*Entries :      100 : Total =        15234773 bytes  File  Size =    7001656 *
*        :          : Tree compression factor =   2.17                       *
*******************************************************************************
*Branch  :event_split                                                        *
*Entries :      100 : BranchElement (see below)                              *
*.............................................................................*
*Br    0 :fUniqueID : UInt_t                                                 *
*Entries :      100 : Total  Size=        976 bytes  File Size  =         97 *
*Baskets :        1 : Basket Size=      16000 bytes  Compression=   4.89     *
*.............................................................................*
*Br    1 :fBits     : UInt_t                                                 *
*Entries :      100 : Total  Size=       1364 bytes  File Size  =        285 *
*Baskets :        1 : Basket Size=      16000 bytes  Compression=   3.08     *
*.............................................................................*
*Br    2 :fType[20] : Char_t                                                 *
*Entries :      100 : Total  Size=       2980 bytes  File Size  =        357 *
*Baskets :        1 : Basket Size=      16000 bytes  Compression=   6.95     *
*.............................................................................*
*Br    3 :fEventName : char*                                                 *
*Entries :      100 : Total  Size=       1389 bytes  File Size  =        288 *
*Baskets :        1 : Basket Size=      16000 bytes  Compression=   3.07     *
*.............................................................................*
*Br    4 :fNtrack   : Int_t                                                  *
*Entries :      100 : Total  Size=        966 bytes  File Size  =        234 *
*Baskets :        1 : Basket Size=      16000 bytes  Compression=   2.02     *
*.............................................................................*
```

```
void tree4r()
{
  // check to see if the event class is in the dictionary
  // if it is not load the definition in libEvent.so
  if (!TClassTable::GetDict("Event")) {
    gSystem->Load("$ROOTSYS/test/libEvent");
  }

  // read the tree generated with tree4w

  //note that we use "new" to create the TFile and TTree objects !
  //because we want to keep these objects alive when we leave this function.
  TFile *f = new TFile("tree4.root");
  TTree *t4 = (TTree*)f->Get("t4");

  // create a pointer to an event object. This will be used
  // to read the branch values.
  Event *event = new Event();

  // get two branches and set the branch address
  TBranch *bntrack = t4->GetBranch("fNtrack");
  TBranch *branch  = t4->GetBranch("event_split");
  branch->SetAddress(&event);

  Long64_t nevent = t4->GetEntries();
  Int_t nselected = 0;
  Int_t nb = 0;
```

```
  for (Long64_t i=0;i<nevent;i++) {
    //read branch "fNtrack"only
    bntrack->GetEntry(i);

    //reject events with more than 587 tracks
    if (event->GetNtrack() > 587)continue;

    //read complete accepted event in memory
    nb += t4->GetEntry(i);
    nselected++;

    //print the first accepted event
    if (nselected == 1) t4->Show();

    //clear tracks array
    event->Clear();
  }

  if (gROOT->IsBatch()) return;
  new TBrowser();
  t4->StartViewer();
}
```

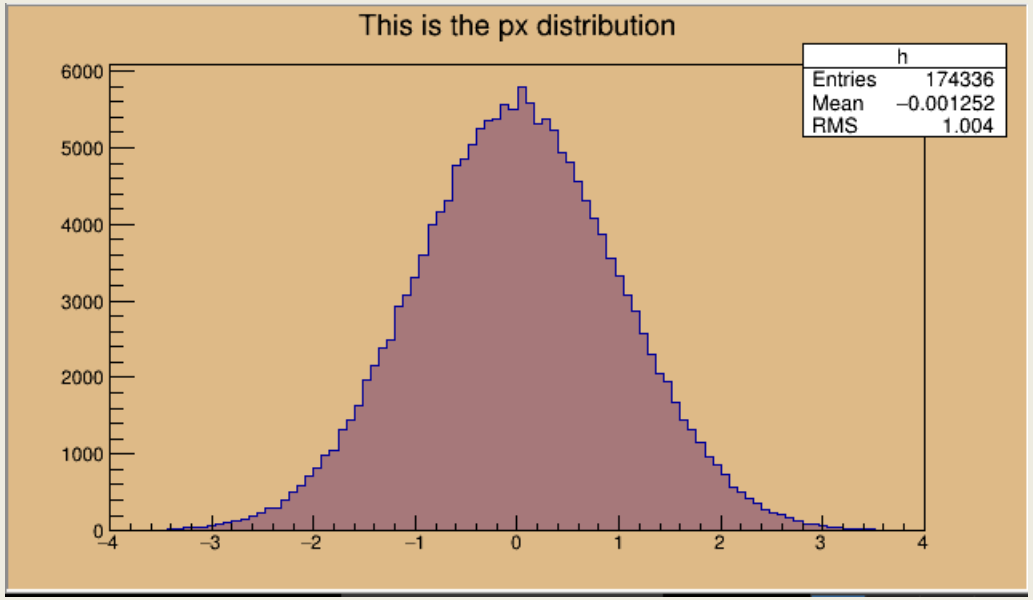
```
======> EVENT:67
 event_split      = (Event*)0x2143280
 fUniqueID        = 0
 fBits            = 50331648
 fType[20]        = t , y , p , e , 2 ,  ,  ,  ,  ,  ,  ,  ,  ,  ,  ,  ,  ,  ,

 fEventName       =
 fNtrack          = 584
 fNseg            = 5853
 fNvertex         = 8
 fFlag            = 1
 fTemperature     = 20.6322
 fMeasures[10]    = 1 , 1 , -3 , -6 , 9 , 2 , 12 , 0 , 5 , 23

 fMatrix[4][4]    = 0.970497 , 2.31131 , -0.247461 , 0 , 0.549269 ,
                    0.762471 , 3.84119 , 0 , -1.40909 , 1.82048 ,
                    3.13009 , 0 , 0 , 0 , 0 ,
                    0

 fClosestDistance = 1.03125
0.390625 , 2.25 , 0.546875 , 0.128906 ,
                    1.65625 , 0.585938 , -0.625 ,
 fEvtHdr.fEvtNum = 67
 fEvtHdr.fRun     = 200
 fEvtHdr.fDate    = 960312
 fTracks          = 584
 fTracks.fUniqueID = 40197, 40198, 40199, 40200, 40201, 40202, 40203, 40204,
 fTracks.fBits    = 50331672, 50331672, 50331672, 50331672, 50331672, 50331672
331672
 fTracks.fPx      = -0.203794, -0.967152, 1.828120, -0.506898, 1.468140, 1.030
16135, -0.100120
```

siguang@pku.edu.cn

```
void tree4() {
   Event::Reset(); // Allow for re-run this script by cleaning static variables.
   tree4w();
   Event::Reset(); // Allow for re-run this script by cleaning static variables.
   tree4r();
}
```

$ROOTSYS/tutorials/tree/hvector.C

展示将vector<float>存在TTree中



siguang@pku.edu.cn

```
#include <vector>

#include "TFile.h"
#include "TTree.h"
#include "TCanvas.h"
#include "TFrame.h"
#include "TH1F.h"
#include "TBenchmark.h"
#include "TRandom.h"
#include "TSystem.h"

#ifdef __MAKECINT__
#pragma link C++ class vector<float>+;
#endif
```

对于用
1、root hvector.C++进行编译运行是必须的，进行预处理产生字典用于数据的存储。所产生的信息写入ACLiC产生的字典文件中。
2、root hvector.C 进行解释运行这句被自动屏蔽

siguang@pku.edu.cn

```
void write()
{

   TFile *f = TFile::Open("hvector.root","RECREATE");

   if (!f) { return; }

   // Create one histograms
   TH1F *hpx = new TH1F("hpx","This is the px distribution",100,-4,4);
   hpx->SetFillColor(48);

   std::vector<float> vpx;
   std::vector<float> vpy;
   std::vector<float> vpz;
   std::vector<float> vrand;

   // Create a TTree
   TTree *t = new TTree("tvec","Tree with vectors");
   t->Branch("vpx",&vpx);
   t->Branch("vpy",&vpy);
   t->Branch("vpz",&vpz);
   t->Branch("vrand",&vrand);
```

siguang@pku.edu.cn

```
// Create a new canvas.
 TCanvas *c1 = new TCanvas("c1","Dynamic Filling Example",200,10,700,500);
 c1->SetFillColor(42);
 c1->GetFrame()->SetFillColor(21);
 c1->GetFrame()->SetBorderSize(6);
 c1->GetFrame()->SetBorderMode(-1);

 gRandom->SetSeed();
 const Int_t kUPDATE = 1000;
 for (Int_t i = 0; i < 25000; i++) {
    Int_t npx = (Int_t)(gRandom->Rndm(1)*15);

    vpx.clear();
    vpy.clear();
    vpz.clear();
    vrand.clear();

    for (Int_t j = 0; j < npx; ++j) {

       Float_t px,py,pz;
       gRandom->Rannor(px,py);
       pz = px*px + py*py;
       Float_t random = gRandom->Rndm(1);

       hpx->Fill(px);

       vpx.push_back(px);
       vpy.push_back(py);
```

```
            vpz.push_back(pz);
            vrand.push_back(random);

        }
        if (i && (i%kUPDATE) == 0) {
            if (i == kUPDATE) hpx->Draw();
            c1->Modified();
            c1->Update();
            if (gSystem->ProcessEvents())
               break;
        }
        t->Fill();
    }
    f->Write();

    delete f;
}
```

```
void read()
{

   TFile *f = TFile::Open("hvector.root","READ");

   if (!f) { return; }

   TTree *t; f->GetObject("tvec",t);

   std::vector<float> *vpx = 0;

  // Create a new canvas.
   TCanvas *c1 = new TCanvas("c1","Dynamic Filling Example",200,10,700,500);
   c1->SetFillColor(42);
   c1->GetFrame()->SetFillColor(21);
   c1->GetFrame()->SetBorderSize(6);
   c1->GetFrame()->SetBorderMode(-1);

   const Int_t kUPDATE = 1000;

   TBranch *bvpx = 0;
   t->SetBranchAddress("vpx",&vpx,&bvpx);


   // Create one histograms
   TH1F *h = new TH1F("h","This is the px distribution",100,-4,4);
   h->SetFillColor(48);
```

```
   for (Int_t i = 0; i < 25000; i++) {

      Long64_t tentry = t->LoadTree(i);
      bvpx->GetEntry(tentry);

      for (UInt_t j = 0; j < vpx->size(); ++j) {

         h->Fill(vpx->at(j));

      }
      if (i && (i%kUPDATE) == 0) {
         if (i == kUPDATE) h->Draw();
         c1->Modified();
         c1->Update();
         if (gSystem->ProcessEvents())
            break;
      }
   }

   // Since we passed the address of a local variable we need
   // to remove it.
   t->ResetBranchAddresses();
}
```

也可用h->Fill((*vpx)[j])代替

```
void hvector()
{
   gBenchmark->Start("hvector");

   write();
   read();

   gBenchmark->Show("hvector");
}
```

```
wsg@debian:~/work/root/534/tutorials/tree/tmp$ root hvector.C
root [0]
Processing hvector.C...
Warning in <TCanvas::Constructor>: Deleting canvas with same name: c1
hvector   : Real Time =   1.03 seconds Cpu Time =   0.72 seconds
root [1]
```

```
wsg@debian:~/work/root/534/tutorials/tree/tmp$ root hvector.C++
root [0]
Processing hvector.C++...
Info in <TUnixSystem::ACLiC>: creating shared library /home/wsg/work/
root/534/tutorials/tree/tmp/./hvector_C.so
Note: Link requested for already precompiled class vector<float,alloc
ator<float> > (ignore this message) :0:
Warning in <TCanvas::Constructor>: Deleting canvas with same name: c1
hvector   : Real Time =   0.85 seconds Cpu Time =   0.54 seconds
root [1]
```

利用TTree自动生成分析框架

王思广

北京大学物理学院

siguang@pku.edu.cn

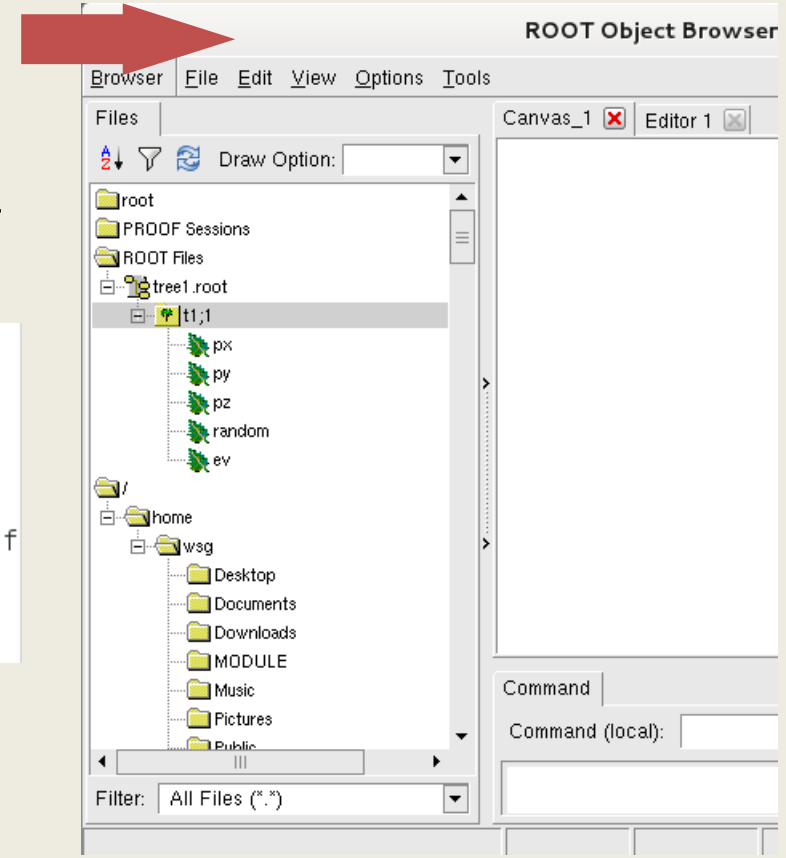
生成框架的三种方法

- t1->MakeClass(“TReadTree”)
- t1->MakeSelector(“Sel”)
- t1->MakeCode(“ReadTree.C”)

利用TTree的MakeClass生成分析框架

1)运行$ROOTSYS/tutorials/tree/tree1.C 生成tree1.root文件，运行方式：root tree1.C
2)运行 $root tree1.root
3)打开new TBrowser, 双击鼠标左键展开tree1.root的t1(查看在root文件中TTree的名字）
4) 在root的命令输入界面中执行：t1->MakeClass("TReadTree")
其中红色字体是任意给的要生成的Class的名字。这样就会在当前目录下产生TReadTree.h及TReadTree.C两个文件。

```
wsg@debian:~/work/root/534/tutorials/tree$ root tree1.root
root [0]
Attaching file tree1.root as _file0...
root [1] new TBrowser
(class TBrowser*)0x11603f0
root [2] t1->MakeClass("TReadTree")
Info in <TTreePlayer::MakeClass>: Files: TReadTree.h and TReadTree.C generated f
rom TTree: t1
(Int_t)0
root [3]
```



TReadTree.h的结构

```
class TReadTree {
public :
   TTree          *fChain;   //!pointer to the analyzed TTree or TChain
   Int_t           fCurrent; //!current Tree number in a TChain

   // Declaration of leaf types
   Float_t         px;
   Float_t         py;
   Float_t         pz;
   Double_t        random;
   Int_t           ev;

   // List of branches
   TBranch        *b_px;   //!
   TBranch        *b_py;   //!
   TBranch        *b_pz;   //!
   TBranch        *b_random;   //!
   TBranch        *b_ev;   //!

   TReadTree(TTree *tree=0);
   virtual ~TReadTree();
   virtual Int_t    Cut(Long64_t entry);
   virtual Int_t    GetEntry(Long64_t entry);
   virtual Long64_t LoadTree(Long64_t entry);
   virtual void     Init(TTree *tree);
   virtual void     Loop();
   virtual Bool_t   Notify();
   virtual void     Show(Long64_t entry = -1);
};
```

循环读内容

siguang@pku.edu.cn

```
TReadTree::TReadTree(TTree *tree) : fChain(0)
{
// if parameter tree is not specified (or zero), connect the file
// used to generate this class and read the Tree.
   if (tree == 0) {
      TFile *f = (TFile*)gROOT->GetListOfFiles()->FindObject("tree1.root");
      if (!f || !f->IsOpen()) {
         f = new TFile("tree1.root");
      }
      f->GetObject("t1",tree);

   }
   Init(tree);
}
```

待分析的root文件
如果也可改为其它的
同结构的待分析文件

TReadTree.C

```
void TReadTree::Loop()
{
//   In a ROOT session, you can do:
//      Root > .L TReadTree.C
//      Root > TReadTree t
//      Root > t.GetEntry(12); // Fill t data members with entry number 12
//      Root > t.Show();       // Show values of entry 12
//      Root > t.Show(16);     // Read and show values of entry 16
//      Root > t.Loop();       // Loop on all entries
//

//     This is the loop skeleton where:
//    jentry is the global entry number in the chain
//    ientry is the entry number in the current Tree
//  Note that the argument to GetEntry must be:
//    jentry for TChain::GetEntry
//    ientry for TTree::GetEntry and TBranch::GetEntry
//
//       To read only selected branches, Insert statements like:
// METHOD1:
//    fChain->SetBranchStatus("*",0);  // disable all branches
//    fChain->SetBranchStatus("branchname",1);  // activate branchname
// METHOD2: replace line
//    fChain->GetEntry(jentry);       //read all branches
//by  b_branchname->GetEntry(ientry); //read only this branch
   if (fChain == 0) return;

   Long64_t nentries = fChain->GetEntriesFast();

   Long64_t nbytes = 0, nb = 0;
   for (Long64_t jentry=0; jentry<nentries;jentry++) {
      Long64_t ientry = LoadTree(jentry);
      if (ientry < 0) break;
      nb = fChain->GetEntry(jentry);   nbytes += nb;
      // if (Cut(ientry) < 0) continue;
   }
}
```

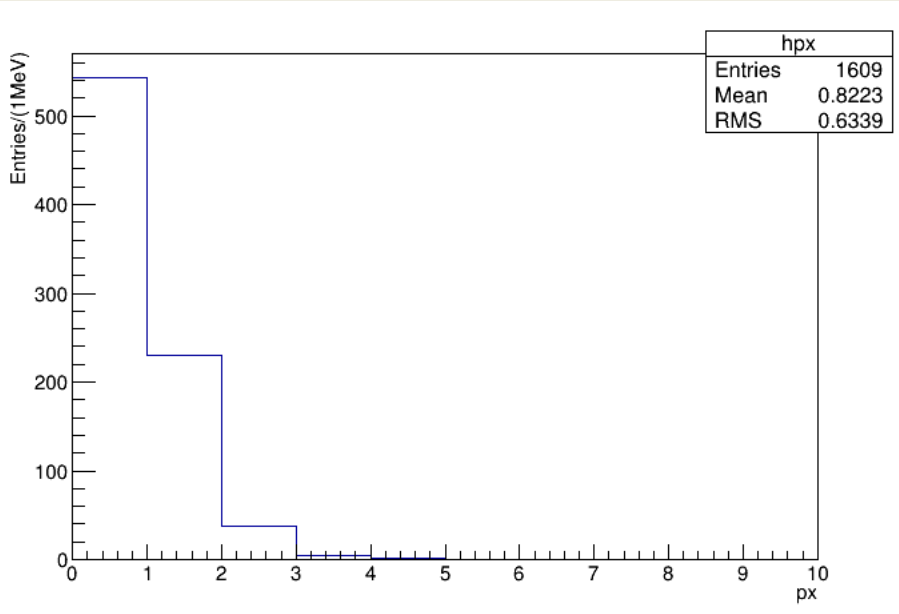
分析代码在循环中

siguang@pku.edu.cn

修改的例子

```
TH1F *hpx = new TH1F("hpx",";px;Entries/(1MeV)",10,0,10); //added
Long64_t nentries = fChain->GetEntriesFast();

Long64_t nbytes = 0, nb = 0;
for (Long64_t jentry=0; jentry<nentries;jentry++) {
   Long64_t ientry = LoadTree(jentry);
   if (ientry < 0) break;
   nb = fChain->GetEntry(jentry);   nbytes += nb;
   // if (Cut(ientry) < 0) continue;
   if(!(py>1)) continue; //added
   if(!(pz>1)) continue; //added
   hpx->Fill(px); //added
}
hpx->Draw(); //added
```

```
wsg@debian:~/work/root/534/tutorials/tree$ root
root [0] .L TReadTree.C
root [1] TReadTree t
root [2] t.Loop()
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
root [3]
```



利用TTree的MakeSelector生成分析框架

- 与MakeClass相似，执行 t1->MakeSelector(“Sel”)将生成Sel.h及Sel.C文件
- 打开Sel.C文件，在Process过程中加分析代码（见右侧红线标识）
- 对于待分析的tree1.root执行如下(t1为tree的名字）：

  >root tree1.root

  []t1->Process(“Sel.C”)

```
Bool_t Sel::Process(Long64_t entry)
{
   // The Process() function is called for each entry in the tree (or possibly
   // keyed object in the case of PROOF) to be processed. The entry argument
   // specifies which entry in the currently loaded tree is to be processed.
   // It can be passed to either Sel::GetEntry() or TBranch::GetEntry()
   // to read either all or the required parts of the data. When processing
   // keyed objects with PROOF, the object is already loaded and is available
   // via the fObject pointer.
   //
   // This function should contain the "body" of the analysis. It can contain
   // simple or elaborate selection criteria, run algorithms on the data
   // of the event and typically fill histograms.
   //
   // The processing can be stopped by calling Abort().
   //
   // Use fStatus to set the return value of TTree::Process().
   //
   // The return value is currently not used.
  fChain->GetTree()->GetEntry(entry); //added by Siguang
  if(px>0) printf("%f\n",px);          //added by Siguang
   return kTRUE;
}
```

siguang@pku.edu.cn

利用TTree的MakeCode生成分析框架

MakeCode生成一个C文件：
t1->MakeCode(“ReadTree.C”)

```
wsg@debian:~/work/root/534/tutorials/tree$ root tree1.root
root [0]
Attaching file tree1.root as _file0...
root [1] t1->MakeCode("ReadTree.C")
Warning in <TTree::MakeCode>: MakeCode is obsolete. Use MakeClass or MakeSelecto
r instead
Macro: ReadTree.C generated from Tree: t1
(Int_t)0
root [2]
```

所生成的ReadTree.C文件并不能直接运行，需要对代码进行稍微改变：

```
wsg@debian:~/work/root/534/tutorials/tree$ root ReadTree.C
root [0]
Processing ReadTree.C...
Error: Symbol tree is not defined in current scope  ReadTree.C:16:
*** Interpreter error recovered ***
root [1]
```

MakeCode生成的ReadTree.C

```
{
//////////////////////////////////////////////////////////
//   This file has been automatically generated
//     (Tue Jul 18 18:45:38 2017 by ROOT version5.34/36)
//   from TTree t1/a simple Tree with simple variables
//   found on file: tree1.root
//////////////////////////////////////////////////////////


//Reset ROOT and connect tree file
   gROOT->Reset();
   TFile *f = (TFile*)gROOT->GetListOfFiles()->FindObject("tree1.root");
   if (!f) {
      f = new TFile("tree1.root");
   }
    f->GetObject("t1",tree);

//Declaration of leaves types
   Float_t         px;
   Float_t         py;
   Float_t         pz;
   Double_t        random;
   Int_t           ev;

   // Set branch addresses.
   t1->SetBranchAddress("px",&px);
   t1->SetBranchAddress("py",&py);
   t1->SetBranchAddress("pz",&pz);
   t1->SetBranchAddress("random",&random);
   t1->SetBranchAddress("ev",&ev);
   Long64_t nentries = t1->GetEntries();

   Long64_t nbytes = 0;
//   for (Long64_t i=0; i<nentries;i++) {
//      nbytes += t1->GetEntry(i);
//   }
}
```

这个语句没有预先定义t1指针，需要改为：
TTree *t1 = (TTree *)f->Get(“t1”);

siguang@pku.edu.cn

- 产生Ntuple：
  $ROOTSYS/tutorials/hsimple.C

- 读取Ntuple：

$ROOTSYS/tutorials/tree/ntuple1.C

siguang@pku.edu.cn

$ROOTSYS/tutorials/hsimple.C

```
TFile *hsimple(Int_t get=0)
{
//  This program creates :
//    - a one dimensional histogram
//    - a two dimensional histogram
//    - a profile histogram
//    - a memory-resident ntuple
//
//  These objects are filled with some random numbers and saved on a file.
//  If get=1 the macro returns a pointer to the TFile of "hsimple.root"
//           if this file exists, otherwise it is created.
//  The file "hsimple.root" is created in $ROOTSYS/tutorials if the caller has
//  write access to this directory, otherwise the file is created in $PWD

   TString filename = "hsimple.root";
   TString dir = gSystem->UnixPathName(gInterpreter->GetCurrentMacroName());
   dir.ReplaceAll("hsimple.C","");
   dir.ReplaceAll("/./","/");
   TFile *hfile = 0;
```

```
if (get) {
   // if the argument get =1 return the file "hsimple.root"
   // if the file does not exist, it is created
   TString fullPath = dir+"hsimple.root";
   if (!gSystem->AccessPathName(fullPath,kFileExists)) {
      hfile = TFile::Open(fullPath); //in $ROOTSYS/tutorials
      if (hfile) return hfile;
   }
   //otherwise try $PWD/hsimple.root
   if (!gSystem->AccessPathName("hsimple.root",kFileExists)) {
      hfile = TFile::Open("hsimple.root"); //in current dir
      if (hfile) return hfile;
   }
}
//no hsimple.root file found. Must generate it !
//generate hsimple.root in current directory if we have write access
if (gSystem->AccessPathName(".",kWritePermission)) {
   printf("you must run the script in a directory with write access\n");
   return 0;
}
hfile = (TFile*)gROOT->FindObject(filename); if (hfile) hfile->Close();
hfile = new TFile(filename,"RECREATE","Demo ROOT file with histograms");
```

```
// Create some histograms, a profile histogram and an ntuple
TH1F *hpx = new TH1F("hpx","This is the px distribution",100,-4,4);
hpx->SetFillColor(48);
TH2F *hpxpy = new TH2F("hpxpy","py vs px",40,-4,4,40,-4,4);
TProfile *hprof = new TProfile("hprof","Profile of pz versus px",100,-4,4,0,20);
TNtuple *ntuple = new TNtuple("ntuple","Demo ntuple","px:py:pz:random:i");

gBenchmark->Start("hsimple");

// Create a new canvas.
TCanvas *c1 = new TCanvas("c1","Dynamic Filling Example",200,10,700,500);
c1->SetFillColor(42);
c1->GetFrame()->SetFillColor(21);
c1->GetFrame()->SetBorderSize(6);
c1->GetFrame()->SetBorderMode(-1);
```

siguang@pku.edu.cn

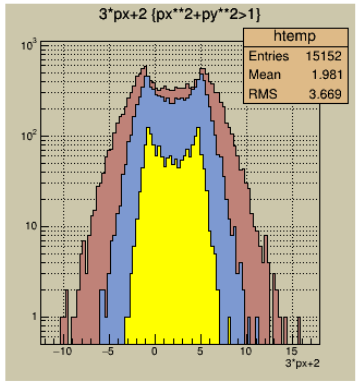
```
// Fill histograms randomly
TRandom3 random;
Float_t px, py, pz;
const Int_t kUPDATE = 1000;
for (Int_t i = 0; i < 25000; i++) {
   random.Rannor(px,py);
   pz = px*px + py*py;
   Float_t rnd = random.Rndm(1);
   hpx->Fill(px);
   hpxpy->Fill(px,py);
   hprof->Fill(px,pz);
   ntuple->Fill(px,py,pz,rnd,i);
   if (i && (i%kUPDATE) == 0) {
      if (i == kUPDATE) hpx->Draw();
      c1->Modified();
      c1->Update();
      if (gSystem->ProcessEvents())
         break;
   }
}
gBenchmark->Show("hsimple");
```

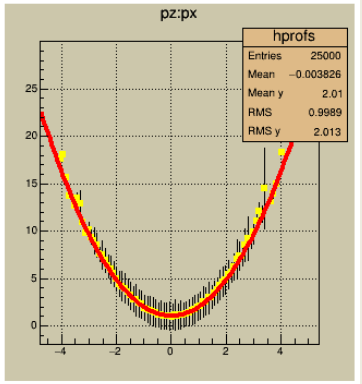
```
root [3] ntuple->Print()
******************************************************************************
*Tree    :ntuple    : Demo ntuple                                            *
*Entries :    25000 : Total =          664510 bytes  File  Size =     400860 *
*        :          : Tree compression factor =   1.25                       *
******************************************************************************
*Br    0 :px        : Float_t                                                *
*Entries :    25000 : Total  Size=     132830 bytes  File Size  =      93173 *
*Baskets :        4 : Basket Size=      32000 bytes  Compression=   1.08     *
*.............................................................................*
*Br    1 :py        : Float_t                                                *
*Entries :    25000 : Total  Size=     132830 bytes  File Size  =      93195 *
*Baskets :        4 : Basket Size=      32000 bytes  Compression=   1.08     *
*.............................................................................*
*Br    2 :pz        : Float_t                                                *
*Entries :    25000 : Total  Size=     132830 bytes  File Size  =      91577 *
*Baskets :        4 : Basket Size=      32000 bytes  Compression=   1.10     *
*.............................................................................*
*Br    3 :random    : Float_t                                                *
*Entries :    25000 : Total  Size=     132866 bytes  File Size  =      90472 *
*Baskets :        4 : Basket Size=      32000 bytes  Compression=   1.11     *
*.............................................................................*
*Br    4 :i         : Float_t                                                *
*Entries :    25000 : Total  Size=     132820 bytes  File Size  =      31643 *
*Baskets :        4 : Basket Size=      32000 bytes  Compression=   3.17     *
*.............................................................................*
root [4]
```

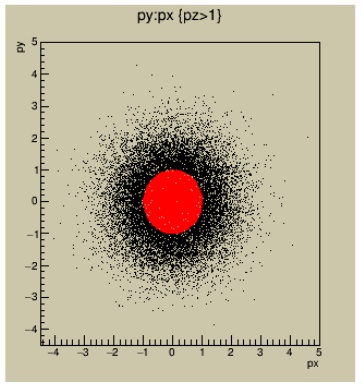
```
   // Save all objects in this file
   hpx->SetFillColor(0);
   hfile->Write();
   hpx->SetFillColor(48);
   c1->Modified();
   return hfile;

// Note that the file is automatically close when application terminates
// or when the file destructor is called.
}
```

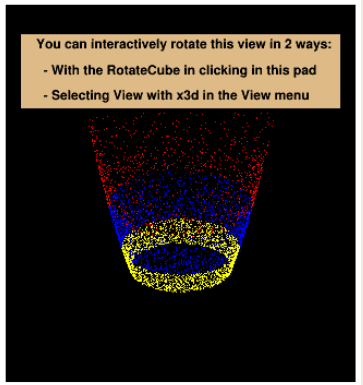
$ROOTSYS/tutorials/tree/ntuple1.C

```
void ntuple1() {
  //Small tree analysis script
  // To see the output of this macro, click begin_html <a href="gif/ntuple1.gif">here</a> end_html
  //Author:: Rene Brun

  //just in case this script is executed multiple times
  delete gROOT->GetListOfFiles()->FindObject("hsimple.root");
  delete gROOT->GetListOfCanvases()->FindObject("c1");

  gBenchmark->Start("ntuple1");
  //
  // Connect ROOT histogram/ntuple demonstration file
  // generated by example $ROOTSYS/tutorials/hsimple.C.
  TFile *f1 = TFile::Open("hsimple.root");
  if (!f1) return;
  //
  // Create a canvas, with 4 pads
  //
  TCanvas *c1 = new TCanvas("c1","The Ntuple canvas",200,10,700,780);
  TPad *pad1 = new TPad("pad1","This is pad1",0.02,0.52,0.48,0.98,21);
  TPad *pad2 = new TPad("pad2","This is pad2",0.52,0.52,0.98,0.98,21);
  TPad *pad3 = new TPad("pad3","This is pad3",0.02,0.02,0.48,0.48,21);
  TPad *pad4 = new TPad("pad4","This is pad4",0.52,0.02,0.98,0.48,1);
  pad1->Draw();
  pad2->Draw();
  pad3->Draw();
  pad4->Draw();
```









```
// Change default style for the statistics box
gStyle->SetStatW(0.30);
gStyle->SetStatH(0.20);
gStyle->SetStatColor(42);
//
// Display a function of one ntuple column imposing a condition
// on another column.
pad1->cd();
pad1->SetGrid();
pad1->SetLogy();
pad1->GetFrame()->SetFillColor(15);
TNtuple *ntuple = (TNtuple*)f1->Get("ntuple");
ntuple->SetLineColor(1);
ntuple->SetFillStyle(1001);
ntuple->SetFillColor(45);
ntuple->Draw("3*px+2","px**2+py**2>1");
ntuple->SetFillColor(38);
ntuple->Draw("2*px+2","pz>2","same");
ntuple->SetFillColor(5);
ntuple->Draw("1.3*px+2","(px^2+py^2>4) && py>0","same");
pad1->RedrawAxis();
```

siguang@pku.edu.cn

```
// Display the profile of two columns
// The profile histogram produced is saved in the current directory with
// the name hprofs
pad2->cd();
pad2->SetGrid();
pad2->GetFrame()->SetFillColor(32);
ntuple->Draw("pz:px>>hprofs","","goffprofs");
TProfile *hprofs = (TProfile*)gDirectory->Get("hprofs");
hprofs->SetMarkerColor(5);
hprofs->SetMarkerSize(0.7);
hprofs->SetMarkerStyle(21);
hprofs->Fit("pol2");
// Get pointer to fitted function and modify its attributes
TF1 *fpol2 = hprofs->GetFunction("pol2");
fpol2->SetLineWidth(4);
fpol2->SetLineColor(2);
//
// Display a scatter plot of two columns with a selection.
// Superimpose the result of another cut with a different marker color
pad3->cd();
pad3->GetFrame()->SetFillColor(38);
pad3->GetFrame()->SetBorderSize(8);
ntuple->SetMarkerColor(1);
ntuple->Draw("py:px","pz>1");
ntuple->SetMarkerColor(2);
ntuple->Draw("py:px","pz<1","same");
```

```
  // Display a 3-D scatter plot of 3 columns. Superimpose a different selection.
  pad4->cd();
  ntuple->Draw("pz:py:px","(pz<10 && pz>6)+(pz<4 && pz>3)");
  ntuple->SetMarkerColor(4);
  ntuple->Draw("pz:py:px","pz<6 && pz>4","same");
  ntuple->SetMarkerColor(5);
  ntuple->Draw("pz:py:px","pz<4 && pz>3","same");
  TPaveText *l4 = new TPaveText(-0.9,0.5,0.9,0.95);
  l4->SetFillColor(42);
  l4->SetTextAlign(12);
  l4->AddText("You can interactively rotate this view in 2 ways:");
  l4->AddText("  - With the RotateCube in clicking in this pad");
  l4->AddText("  - Selecting View with x3d in the View menu");
  l4->Draw();
  //
  c1->cd();
  c1->Update();
  gStyle->SetStatColor(19);
  gBenchmark->Show("ntuple1");
}
```

满足筛选条件下的TTree复制

王思广

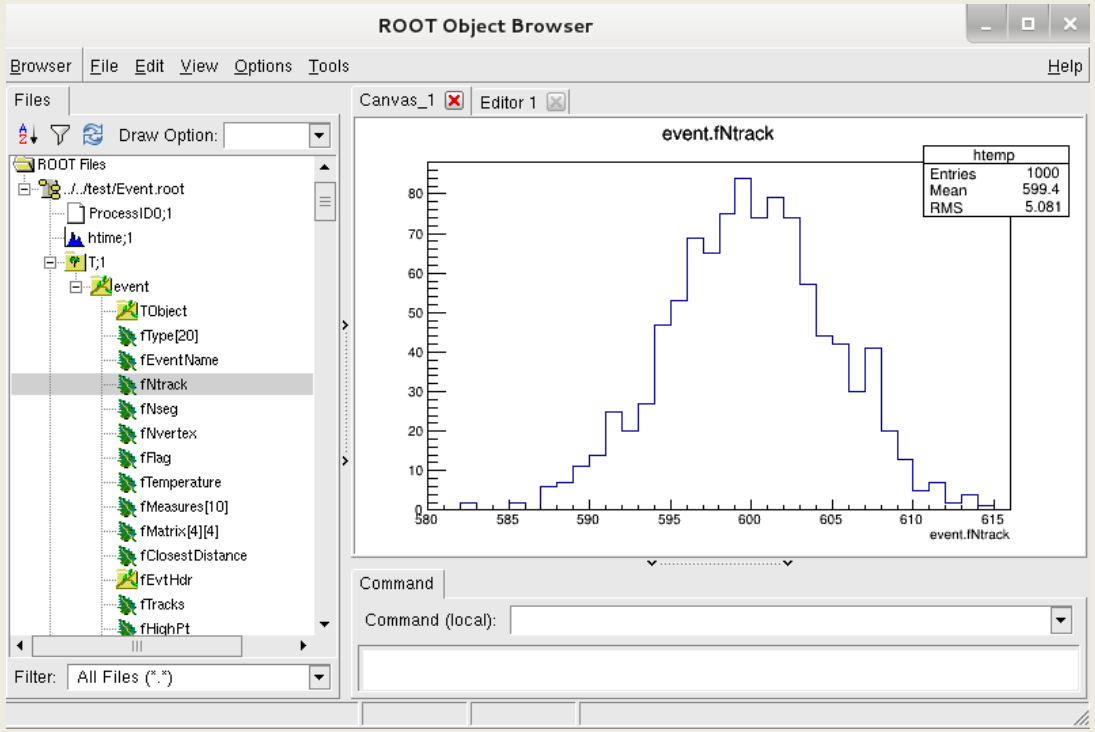
北京大学物理学院

siguang@pku.edu.cn

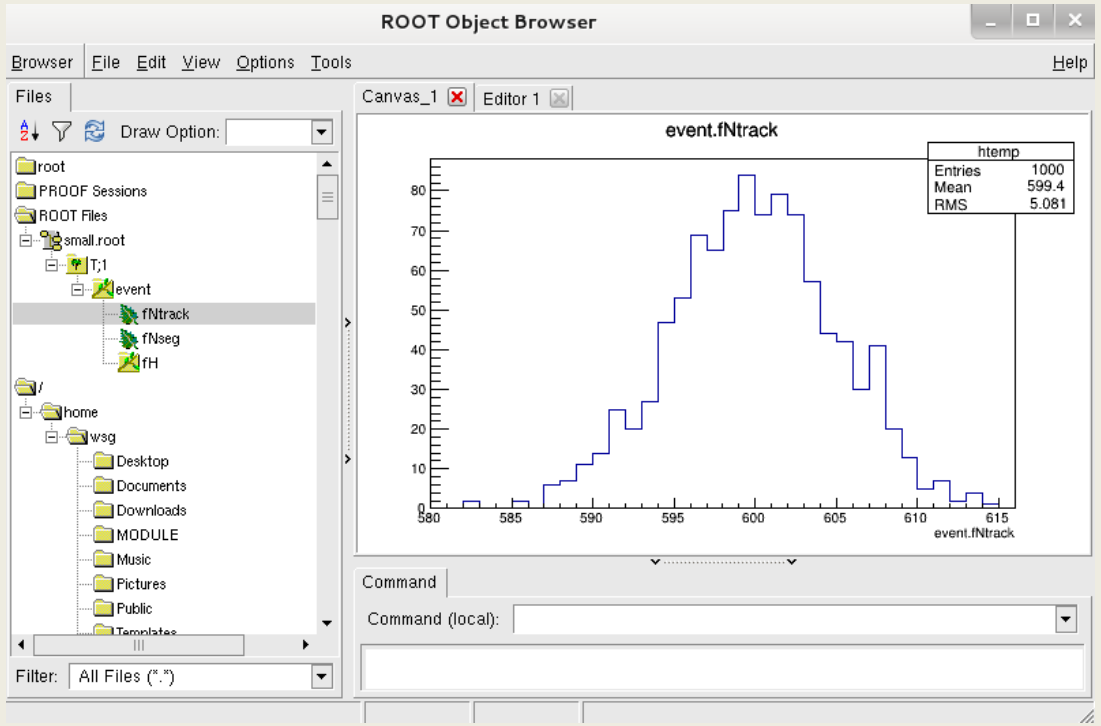
部分Branch复制：

例子：$ROOTSYS/tutorials/tree/copytree.C

该例子所使用的输入数据Event.root生成代码：用$ROOTSYS/test/Event
>./Event 1000 1 1 1



输入Event.root, 文件大小：36Mb



生成的small.root, 文件大小：92kb

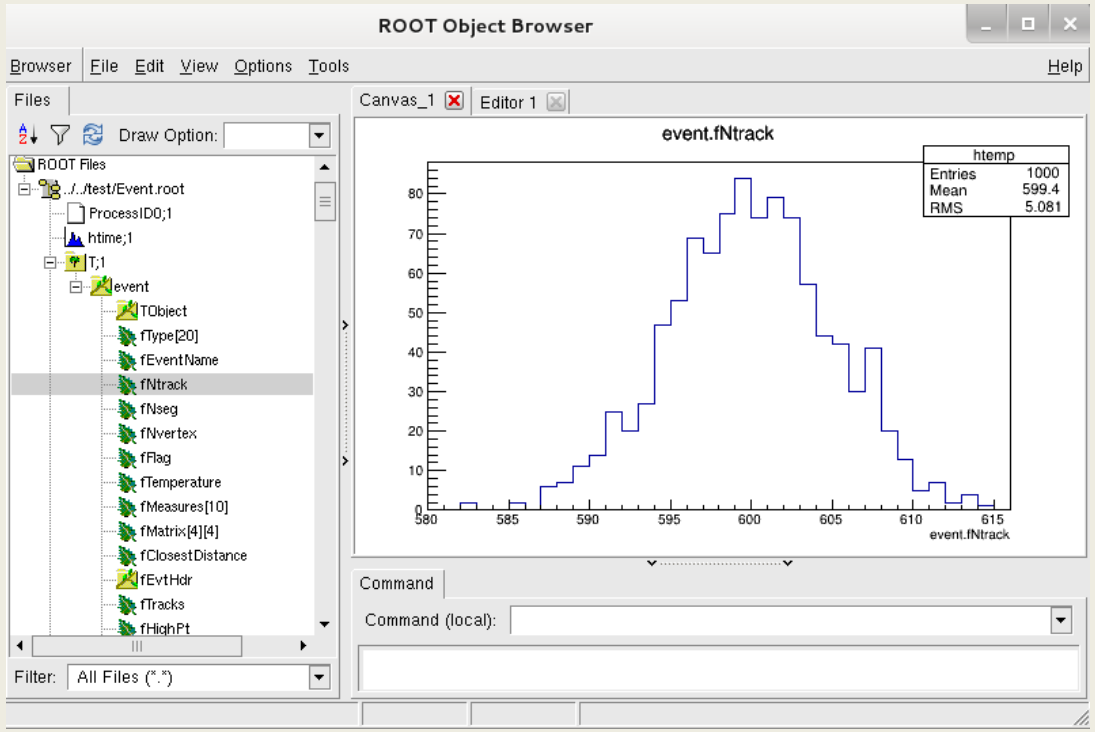
$ROOTSYS/tutorials/tree/copytree.C

```
void copytree() {
// Example of Root macro to copy a subset of a Tree to a new Tree
// The input file has been generated by the program in $ROOTSYS/test/Event
// with   Event 1000 1 1 1
//Author: Rene Brun

  gSystem->Load("$ROOTSYS/test/libEvent");

  //Get old file, old tree and set top branch address
  TFile *oldfile = new TFile("$ROOTSYS/test/Event.root");
  TTree *oldtree = (TTree*)oldfile->Get("T");
  Event *event  = new Event();
  oldtree->SetBranchAddress("event",&event);
  oldtree->SetBranchStatus("*",0);
  oldtree->SetBranchStatus("event",1);
  oldtree->SetBranchStatus("fNtrack",1);
  oldtree->SetBranchStatus("fNseg",1);
  oldtree->SetBranchStatus("fH",1);

  //Create a new file + a clone of old tree in new file
  TFile *newfile = new TFile("small.root","recreate");
  TTree *newtree = oldtree->CloneTree();

  newtree->Print();
  newfile->Write();
  delete oldfile;
  delete newfile;
}
```

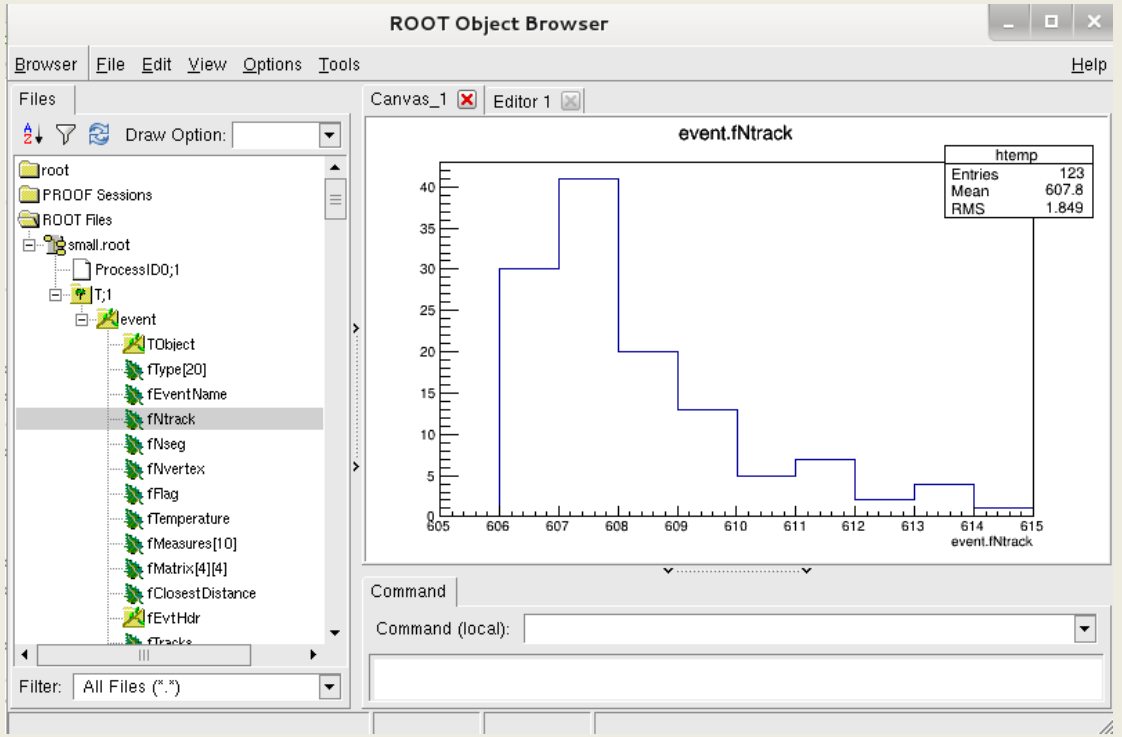
部分数据复制：

例子：$ROOTSYS/tutorials/tree/copytree3.C

该例子所使用的输入数据Event.root生成代码：用$ROOTSYS/test/Event
>./Event 1000 1 1 1



输入Event.root, 文件大小：36Mb



生成的small.root, 文件大小：4.6Mb，TTree的结构一样

$ROOTSYS/tutorials/tree/copytree3.C

```
void copytree3() {
// Example of Root macro to copy a subset of a Tree to a new Tree
// Only selected entries are copied to the new Tree.
// The input file has been generated by the program in $ROOTSYS/test/Event
// with   Event 1000 1 99 1
//Author: Rene Brun

  gSystem->Load("$ROOTSYS/test/libEvent");

  //Get old file, old tree and set top branch address
  TFile *oldfile = new TFile("$ROOTSYS/test/Event.root");
  TTree *oldtree = (TTree*)oldfile->Get("T");
  Long64_t nentries = oldtree->GetEntries();
  Event *event  = 0;
  oldtree->SetBranchAddress("event",&event);

  //Create a new file + a clone of old tree in new file
  TFile *newfile = new TFile("small.root","recreate");
  TTree *newtree = oldtree->CloneTree(0);

  for (Long64_t i=0;i<nentries; i++) {
    oldtree->GetEntry(i);
    if (event->GetNtrack() > 605) newtree->Fill();
    event->Clear();
  }
  newtree->Print();
  newtree->AutoSave();
  delete oldfile;
  delete newfile;
}
```

“0”代表不进行数据复制，仅仅数据结构的复制

TClonesArray及TGenPhaseSpace的演示

王思广

北京大学物理学院

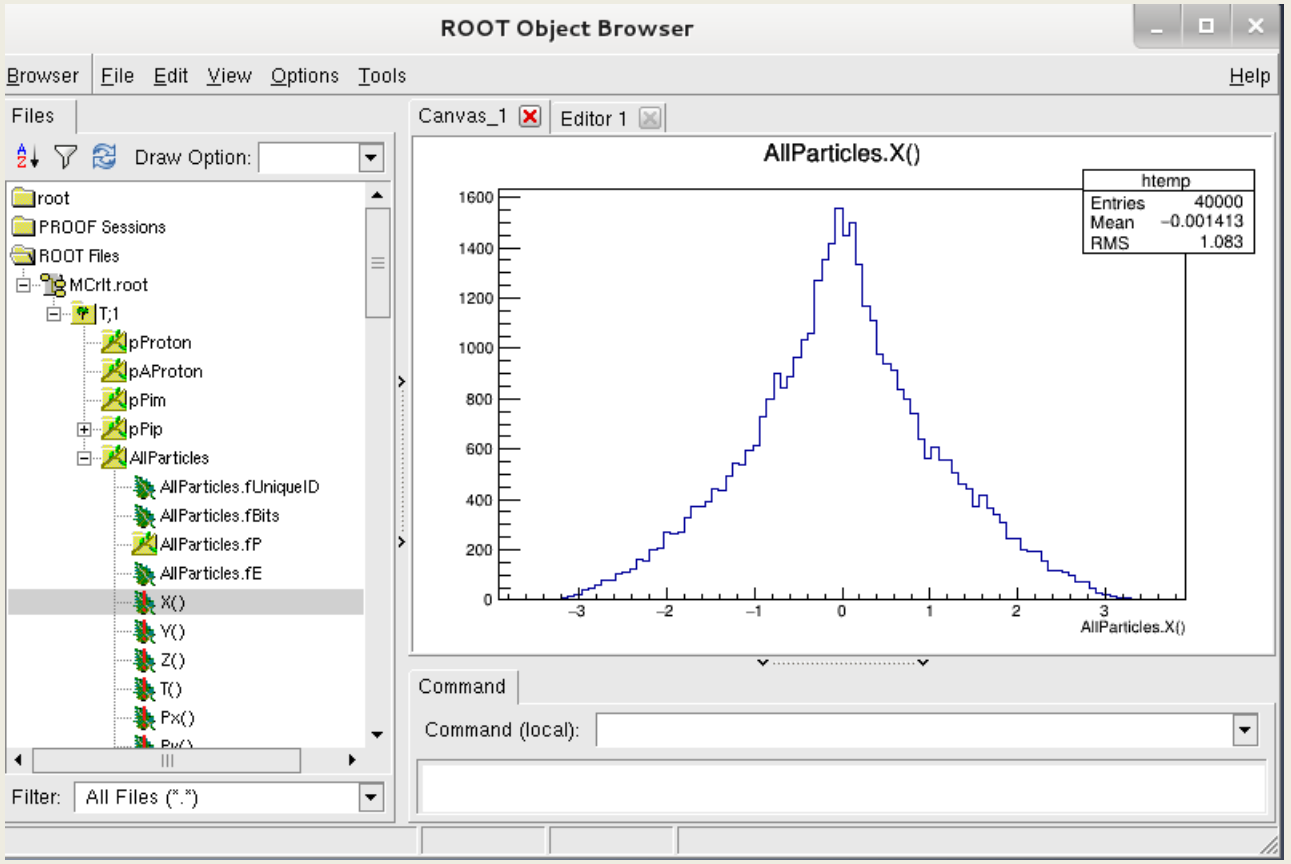
siguang@pku.edu.cn

TClonesArray 可以将同一类型的继承于TObject的类叠加在一起存储到TTree中；
TGenPhaseSpace可以模拟产生次级粒子的四动量（用TLorentzVector表示）

```
#include <TGenPhaseSpace.h>
#include <TROOT.h>
#include <TSystem.h>
#include <TFile.h>
#include <TTree.h>
#include <TClonesArray.h>

void GenPhaseSpaceClArray() {

  if (!gROOT->GetClass("TGenPhaseSpace")) gSystem->Load("libPhysics");
  Int_t nSplit = 1;
  TLorentzVector target(0.0, 0.0, 0.0, 0.938); //px py pz  Energy
  TLorentzVector beam(0.0, 0.0, 27.6,27.6);    //
  TLorentzVector W = beam + target;

  //(Momentum, Energy units are Gev/C, GeV)
  Double_t masses[4] = { 0.938, 0.139, 0.938,0.139} ;// e + Proton -> Proton + Pi^- + Anti-Proton + Pi^+

  TGenPhaseSpace event;

  event.SetDecay(W, 4, masses);
  TFile *fout = new TFile("MCrlt.root","recreate");
  TTree *T = new TTree("T","MC Results");
  TLorentzVector *pProton = 0;
  TLorentzVector *pPim   = 0;
  TLorentzVector *pAProton = 0;
  TLorentzVector *pPip   = 0;

  TClonesArray *AllParticles = new TClonesArray("TLorentzVector",1000);
```

```
T->Branch("pProton",&pProton,32000,nSplit);
T->Branch("pAProton",&pAProton,32000,nSplit);
T->Branch("pPim",&pPim,32000,nSplit);
T->Branch("pPip",&pPip,32000,nSplit);
T->Branch("AllParticles",&AllParticles,32000,nSplit);

if(nSplit>0) AllParticles->BypassStreamer(kTRUE);
else AllParticles->BypassStreamer(kFALSE);
/*  When the kBypassStreamer bit is set, the automatically                */
/*  generated Streamer can call directly TClass::WriteBuffer.             */
/*  Bypassing the Streamer improves the performance when writing/reading */
/*  the objects in the TClonesArray.                                      */

for (Int_t n=0;n<10000;n++) {
  event.Generate();
  pProton = event.GetDecay(0);
  pPim   = event.GetDecay(1);
  pAProton = event.GetDecay(2);
  pPip   = event.GetDecay(3);
  TLorentzVector *pNewPart1 = new TLorentzVector(*pProton);
  TLorentzVector *pNewPart2 = new TLorentzVector(*pAProton);
  TLorentzVector *pNewPart3 = new TLorentzVector();
  *pNewPart3 = *pPip;
  TLorentzVector *pNewPart4 = new TLorentzVector();
  pNewPart4->SetPxPyPzE(pPim->Px(),pPim->Py(),pPim->Pz(),pPim->E());

  new ((*AllParticles)[0]) TLorentzVector(*pNewPart1);
  new ((*AllParticles)[1]) TLorentzVector(*pNewPart2);
  new ((*AllParticles)[2]) TLorentzVector(*pNewPart3);
  new ((*AllParticles)[3]) TLorentzVector(*pNewPart3);
  T->Fill();
```

```
    AllParticles->SetOwner(kTRUE); // with this to enable delete objects
    AllParticles->Clear("C");
  }
  fout->cd();
  T->Write();
  fout->Close();
  delete fout;
}
```



siguang@pku.edu.cn