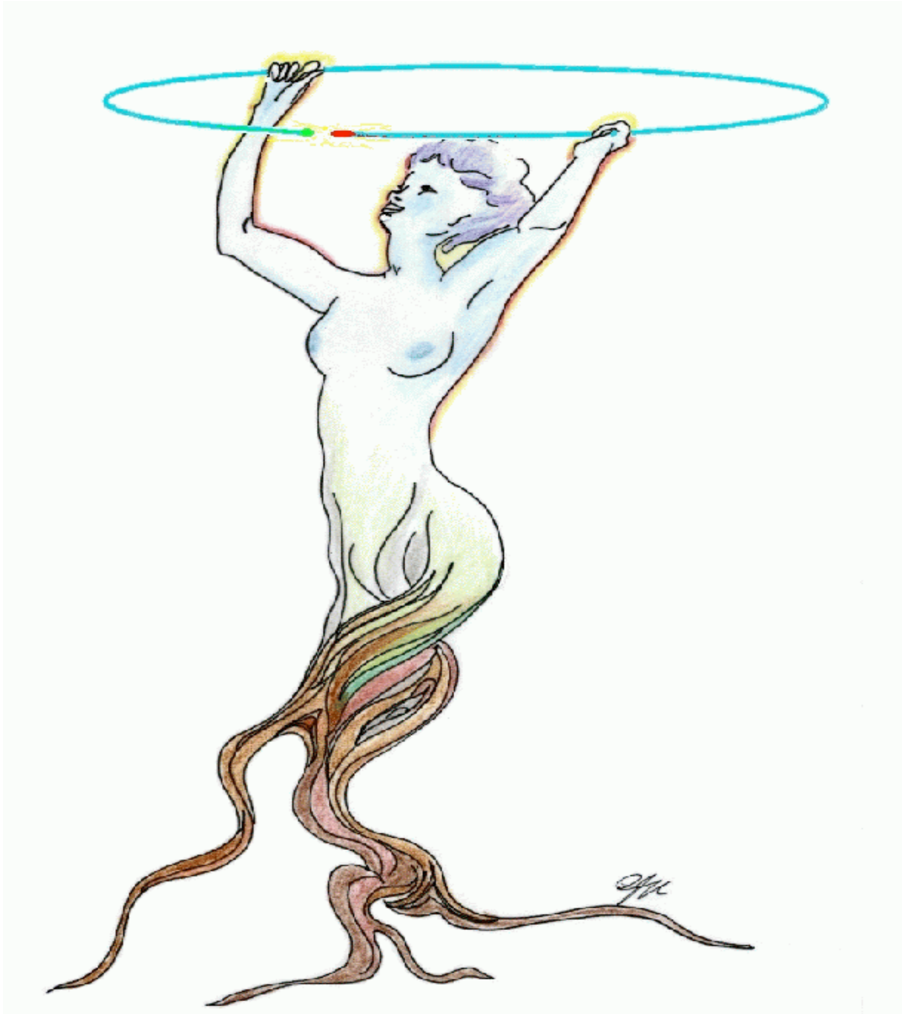
ROOT for beginners

Fourth day

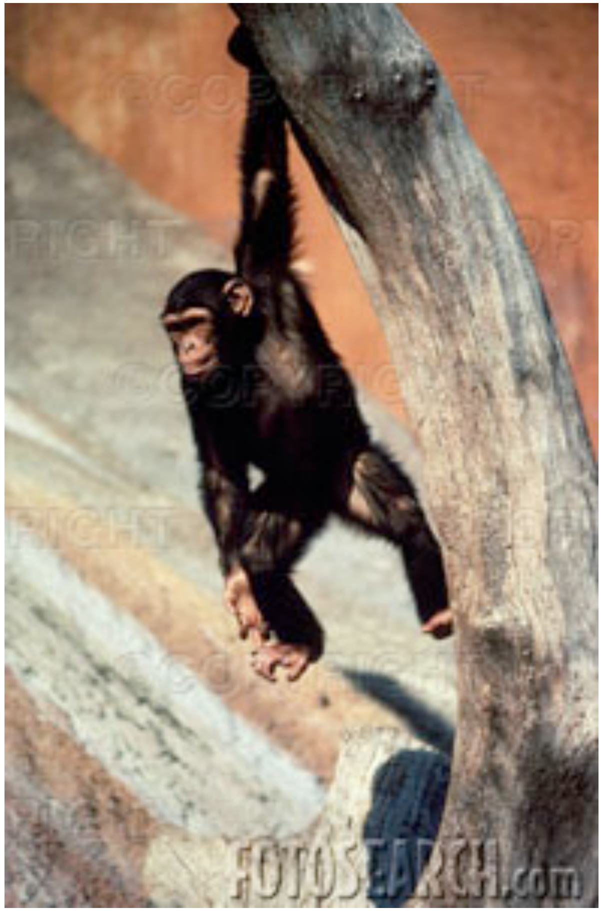
Trees



Let us climb on trees…

Today we will:

- Create a tree
- Fill it
- Read it
- Make analyses
- …



Create a tree

In the shadow of my tree…

- A `TTree` can contain integers, real numbers, *structures*, even *objects*...

tree name · *tree title*

```
TTree *tree=new TTree("MyTree","My 1st tree");
```

tree branches contain the variables (leaves)

```
tree->Branch("My",&super,"branch/F");
```

name of the ***branch***

Name and type of the ***variable***

variable address in the memory

Defining the branches

- Simple variables

```
Int_t mult;
tree->Branch("anInteger", &mult, "Mult/I");
Double_t ToF;
tree->Branch("aDouble", &ToF, "TdV/D");
```

- Fixed size array

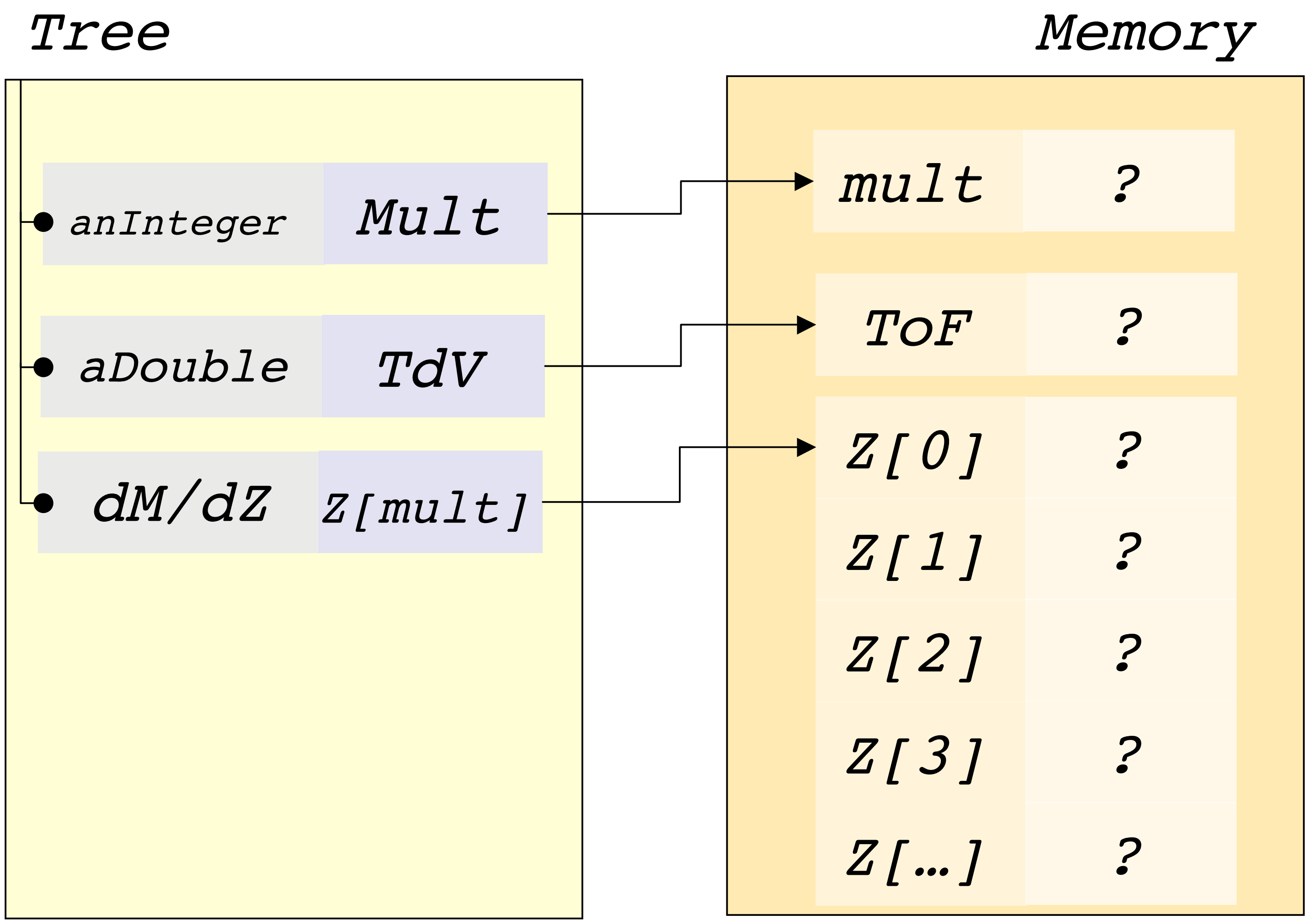
```
Double_t Z[50];
tree->Branch("Z_branch", Z, "Charge[50]/D");
```

Beware!! The array name = the array address !!

- Variable size array

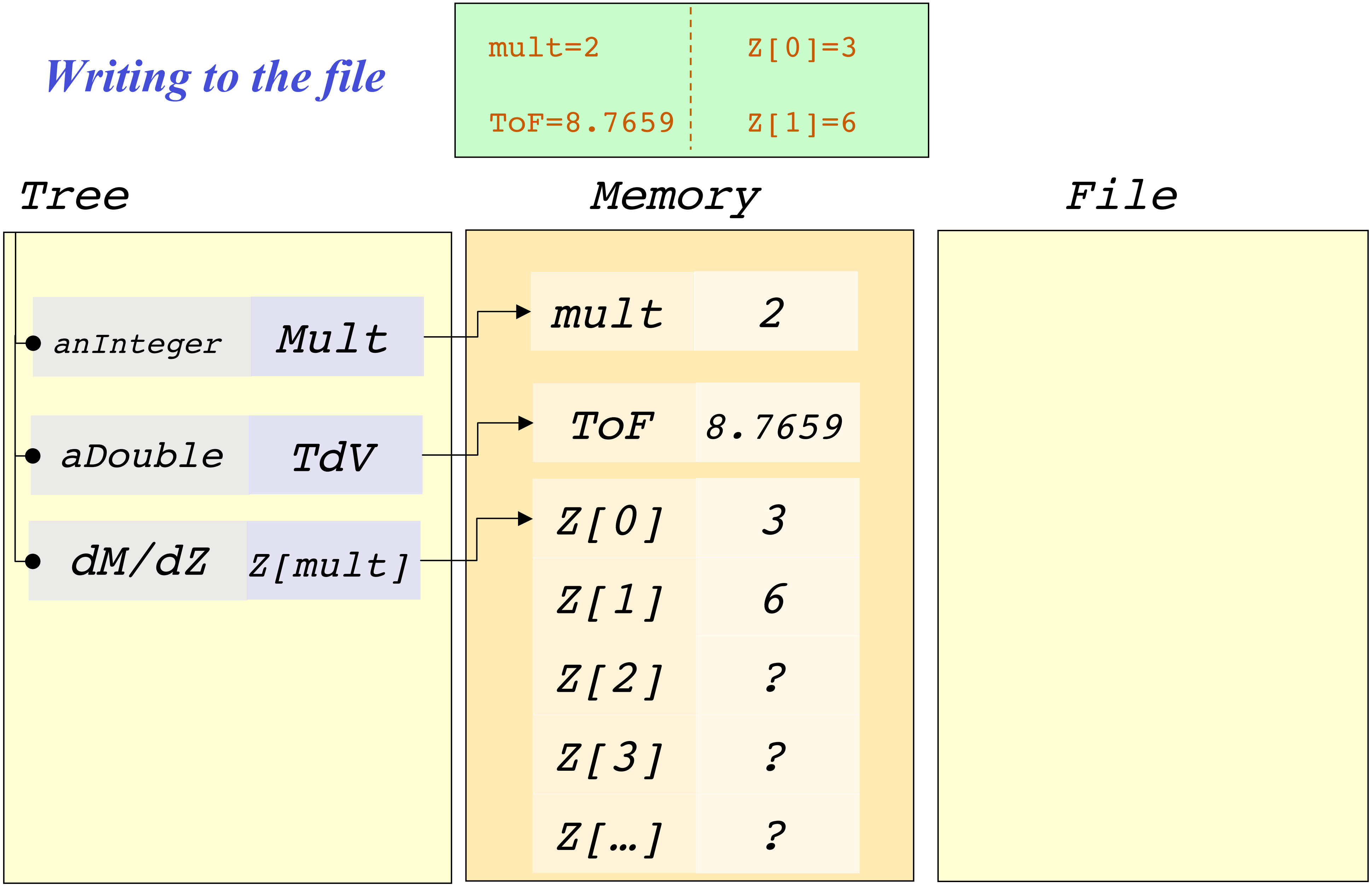
```
tree->Branch("Mult", &mult, "mult/I");
tree->Branch("dM/dZ", Z, "Z[mult]/D");
```

What happens in memory...



What happens in memory…

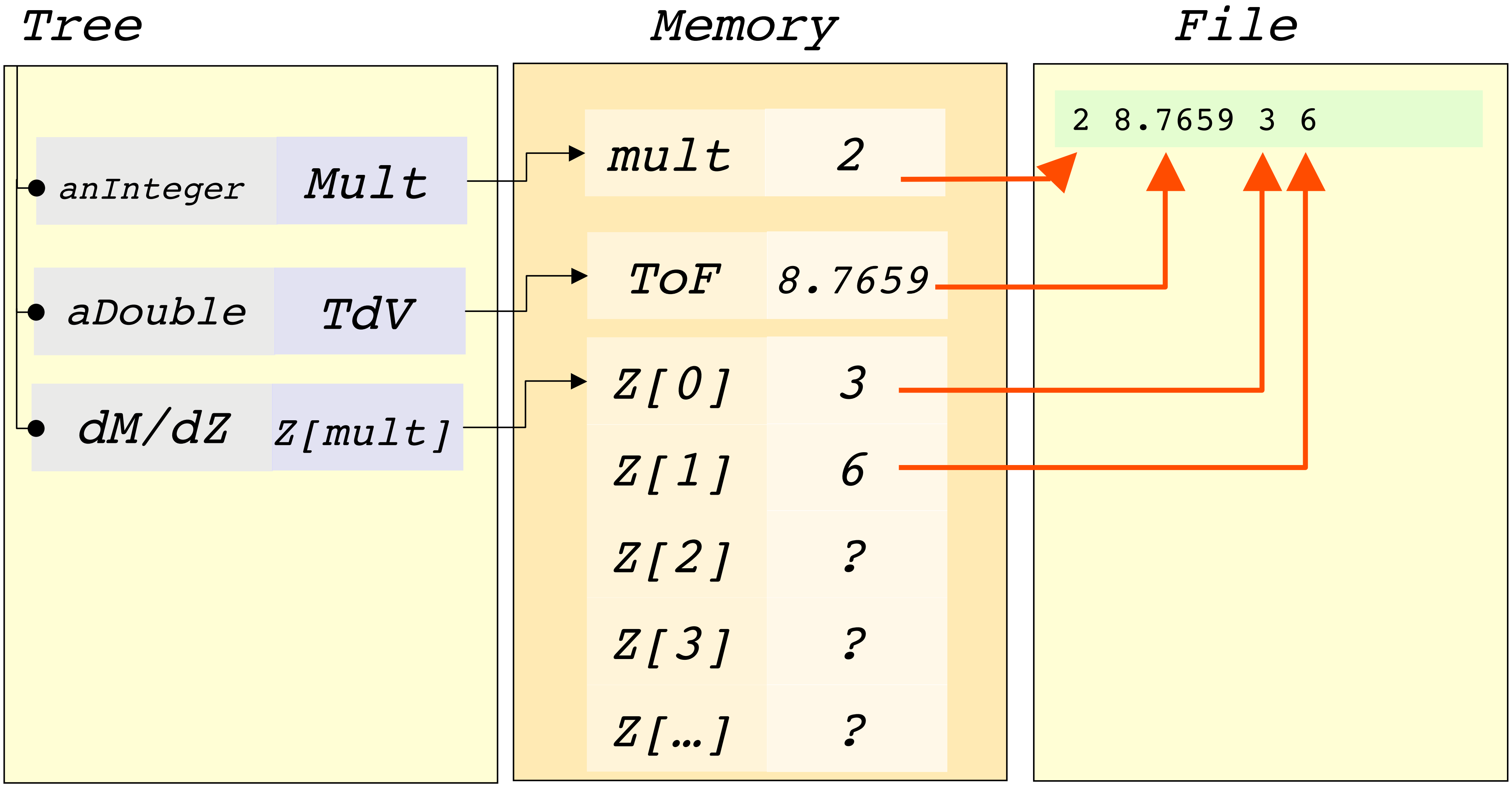
Writing to the file



What happens in memory…

Writing to the file

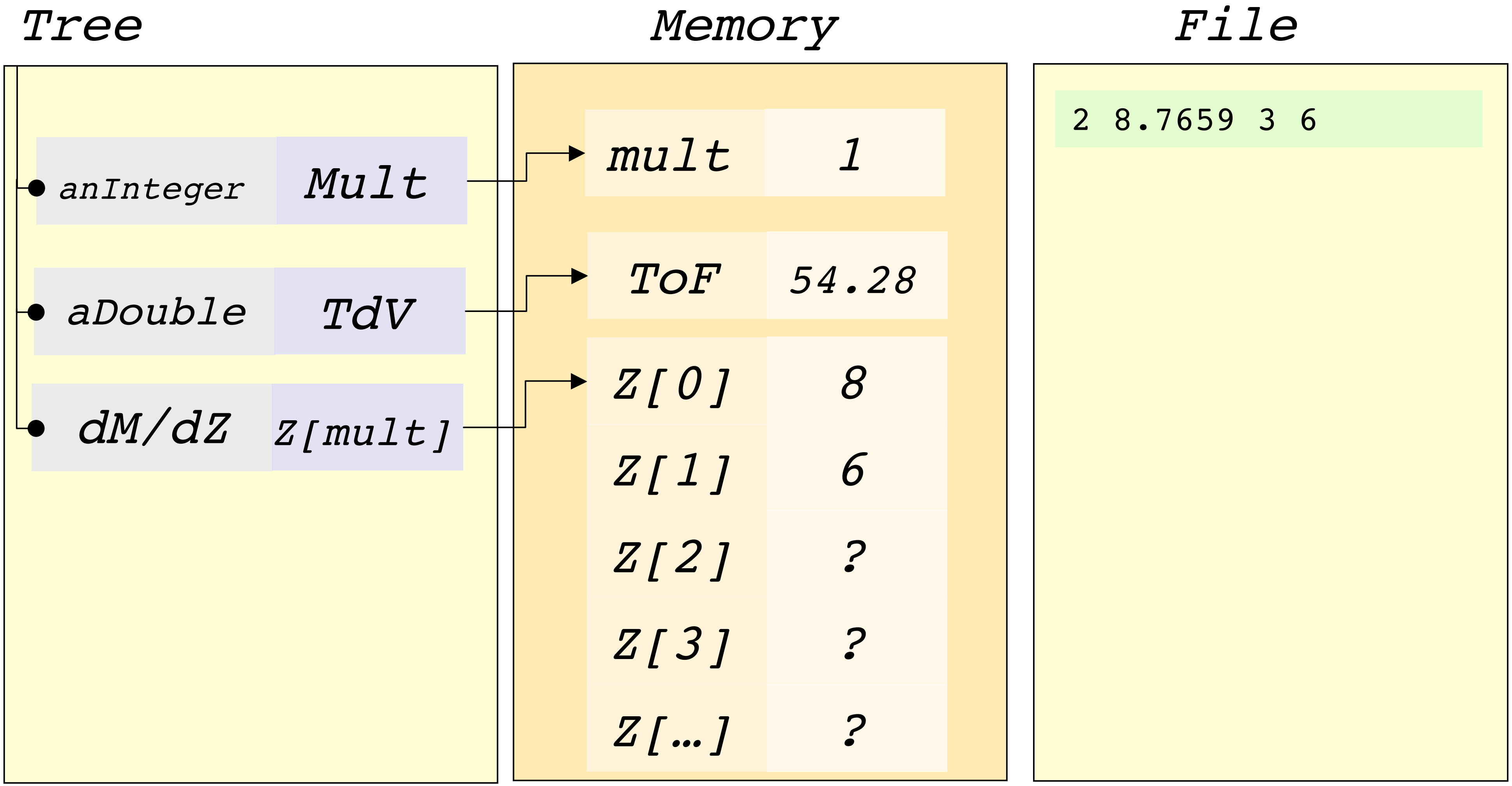
```
tree->Fill()
```



What happens in memory...

Writing to the file

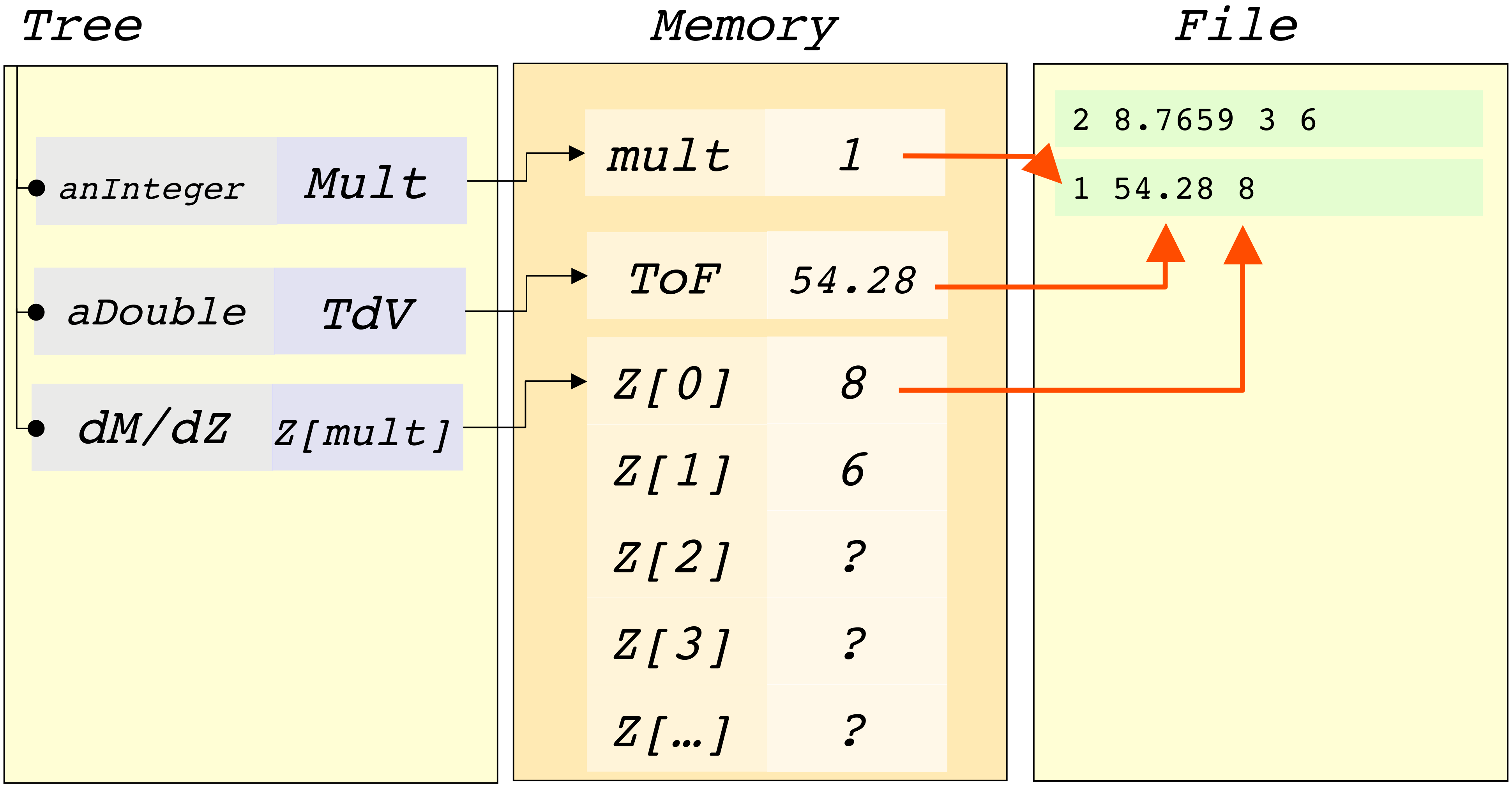
```
mult=1
ToF=54.28
Z[0]=8
```



What happens in memory...

Writing to the file

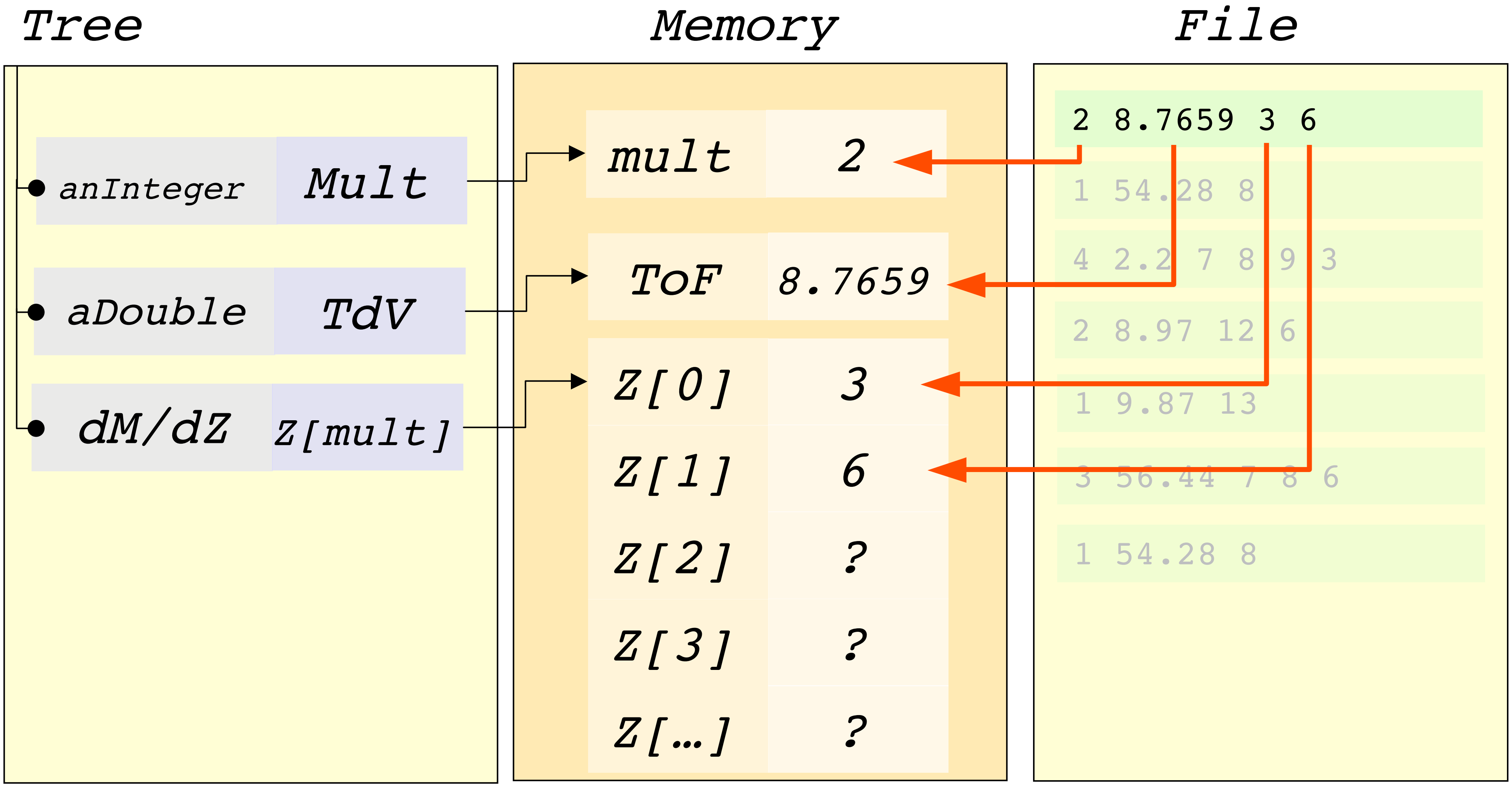
```
tree->Fill()
```



What happens in memory…

Reading the file

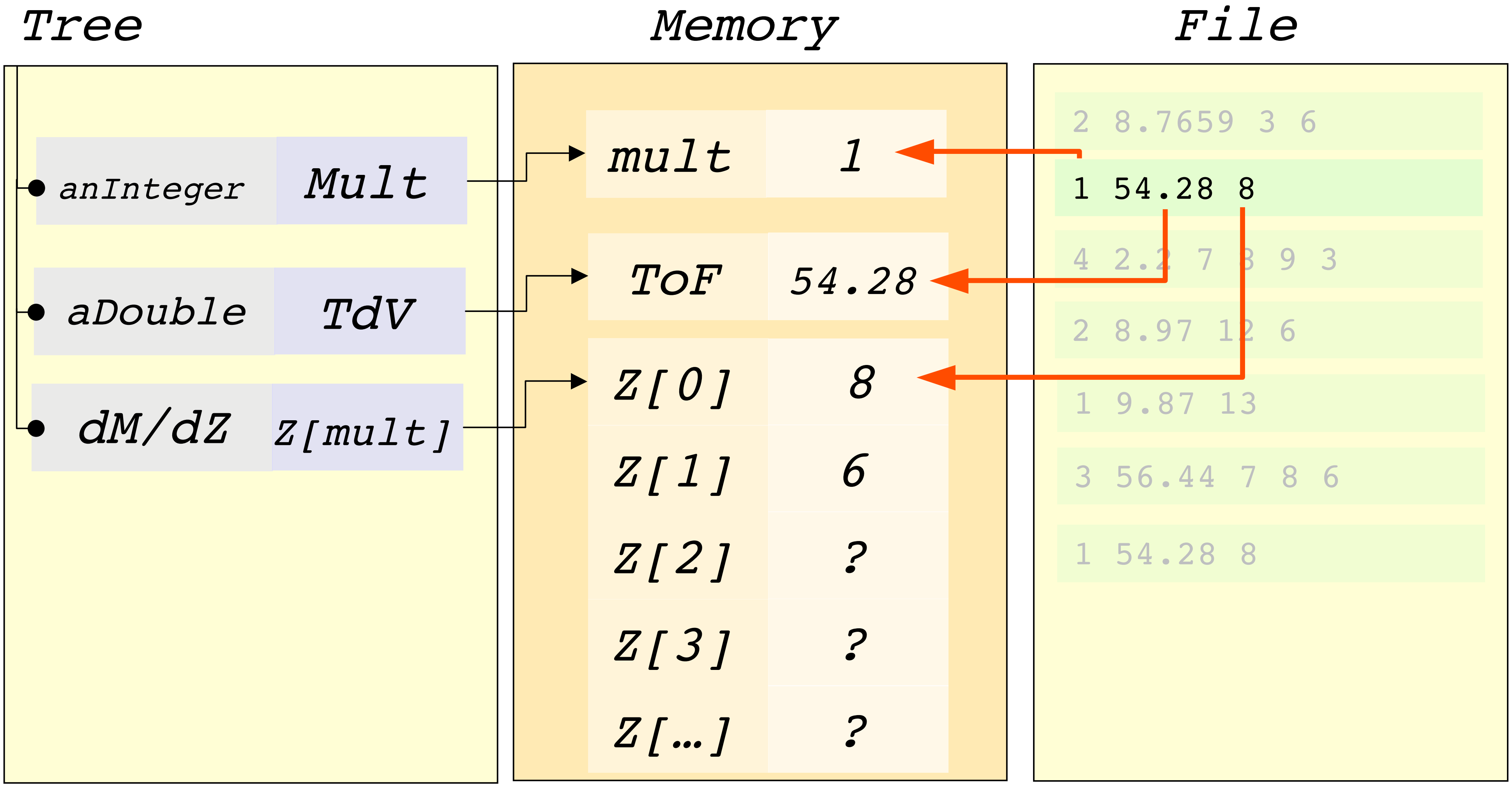
```
tree->GetEntry(0)
```



What happens in memory…

Reading the file

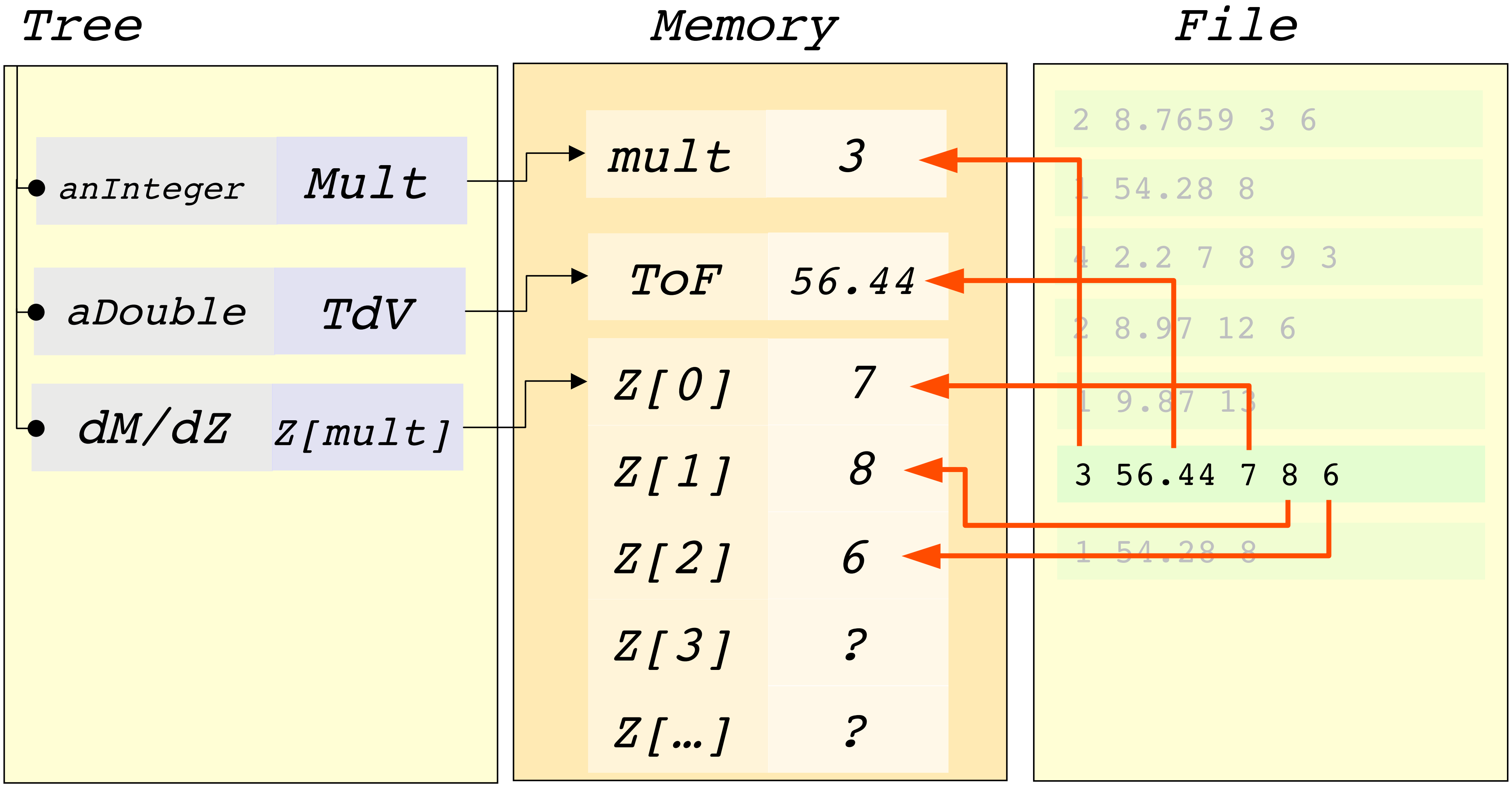
```
tree->GetEntry(1)
```



What happens in memory...

Reading the file

```
tree->GetEntry(5)
```



Example: filling a tree with data

http://caeinfo.in2p3.fr/root/Formation/en/Day4/tree_struc.C
http://caeinfo.in2p3.fr/root/Formation/en/Day4/tree_struc.data

- Have a look at the file **tree_struc.C**
- We will use a *structure** :

**it's not ROOT, it's from C !*

Declaration

```
struct Mon_Event{
   Int_t mult;
   Float_t Z[50];
   Float_t Theta[50];
   Float_t Energie[50];
   };
```

Use

```
Mon_Event event;

event.mult = 0;
event.Z[3] = 2;
file >> event.mult;
```

Reading data in a file

Example: filling a tree with data

- Declaration of the tree

```
TTree *t = new TTree("t", "TTree with a structure");
```

The TTree will be in the general memory (heap)

- Declaration of a branch with an integer and three branches with variable size arrays of single precision real numbers

```
t->Branch("M_part", &event.Mult, "Mult/I");
t->Branch("Z_part", event.Z, "Z[Mult]/F");
t->Branch("Th_part", event.Theta, "Theta[Mult]/F");
t->Branch("E_part", event.Energie, "Energie[Mult]/F");
```

The name of the branch is not necessarily the name of the variable

The arrays have a variable size

With a single branch…

http://caeinfo.in2p3.fr/root/Formation/en/Day4/tree_struc2.C

- Have a look at the file **tree_struc2.C**

- Declaration of a single branch pointing to the structure

the address of the variable ***event*** *of type* ***Mon_Event*** *is given*

arrays have a fixed size

```
t->Branch("bEvent",&event,
           "Mult/I:Z[50]/F:Theta[50]/F:Energie[50]/F");
```

There are many leaves (variables) on this branch

Example: filling a tree with data

- Data will be read in the ASCII file `tree_struc.data`

```
#include "Riostream.h"
...
ifstream file;
file.open("tree_struc.data");
...
file >> event.Mult;
...
for(Int_t i=0;i<event.Mult;i++)
 {
 file >> event.Z[i];
 file >> event.Theta[i];
 file >> event.Energie[i];
 }
t->Fill();
...
file.close();
```

Special ROOT declaration of input/output system of C++ (`#include "Riostream.h"`)

opening the data file (`file.open("tree_struc.data");`)

Reading the data and filling the structure (`file >> event.Mult;` … `file >> event.Energie[i];`)

the data in the structure are transferred to the tree (`t->Fill();`)

Looking at the tree structure

- Run the script and look at the tree !

```
.L tree_struc.C+

MakeTree()
```

```
TFile *f=new
    TFile("tree_struc.root")
TTree *a=(TTree *)f->Get("t")
a->Print()
```

```
******************************************************************************
*Tree    :t         : TTree avec une structure                               *
*Entries :   100000 : Total =        25750346 bytes  File  Size =   16900683 *
*        :          : Tree compression factor =   1.52                       *
******************************************************************************
*Br    0 :M_part    : Mult/I                                                 *
*Entries :   100000 : Total  Size=     401568 bytes  File Size  =      94299 *
*Baskets :       12 : Basket Size=      32000 bytes  Compression=   4.07     *
*............................................................................*
*Br    1 :Z_part    : Z[Mult]/F                                              *
*Entries :   100000 : Total  Size=    8449454 bytes  File Size  =    1840614 *
*Baskets :      276 : Basket Size=      32000 bytes  Compression=   4.58     *
*............................................................................*
*Br    2 :Th_part   : Theta[Mult]/F                                          *
*Entries :   100000 : Total  Size=    8449745 bytes  File Size  =    7396565 *
*Baskets :      276 : Basket Size=      32000 bytes  Compression=   1.14     *
*............................................................................*
*Br    3 :E_part    : Energie[Mult]/F                                        *
*Entries :   100000 : Total  Size=    8449472 bytes  File Size  =    7520599 *
*Baskets :      276 : Basket Size=      32000 bytes  Compression=   1.12     *
*............................................................................*
```

Accessing the tree data

- Looking at an "event"

  a->Show(15)

```
======> EVENT:15
 Mult                 = 15
 Z                    = 30,
                       34, 1, 1, 17, 1,
                       8, 2, 1, 1, 2,
                       2, 1, 1, 2
 Theta                = 14.8766,
                       10.048, 59.2787, 164.868, 8.45649, 21.6054,
                       46.5263, 28.4612, 29.1083, 72.3277, 57.2474,
                       32.4265, 16.6426, 6.97173, 9.6734
 Energie              = 983.813,
                       44.1665, 85.591, 29.5007, 655.211, 59.0234,
                       155.18, 134.403, 21.3786, 10.8284, 19.2134,
                       36.4518, 79.2352, 23.5012, 24.5475
```

Using a tree

Accessing the tree data

- Selecting the events and print variables values:

```
a->Scan("Mult:Z[30]:Energie[30]","Mult>30","",1000,0)
```

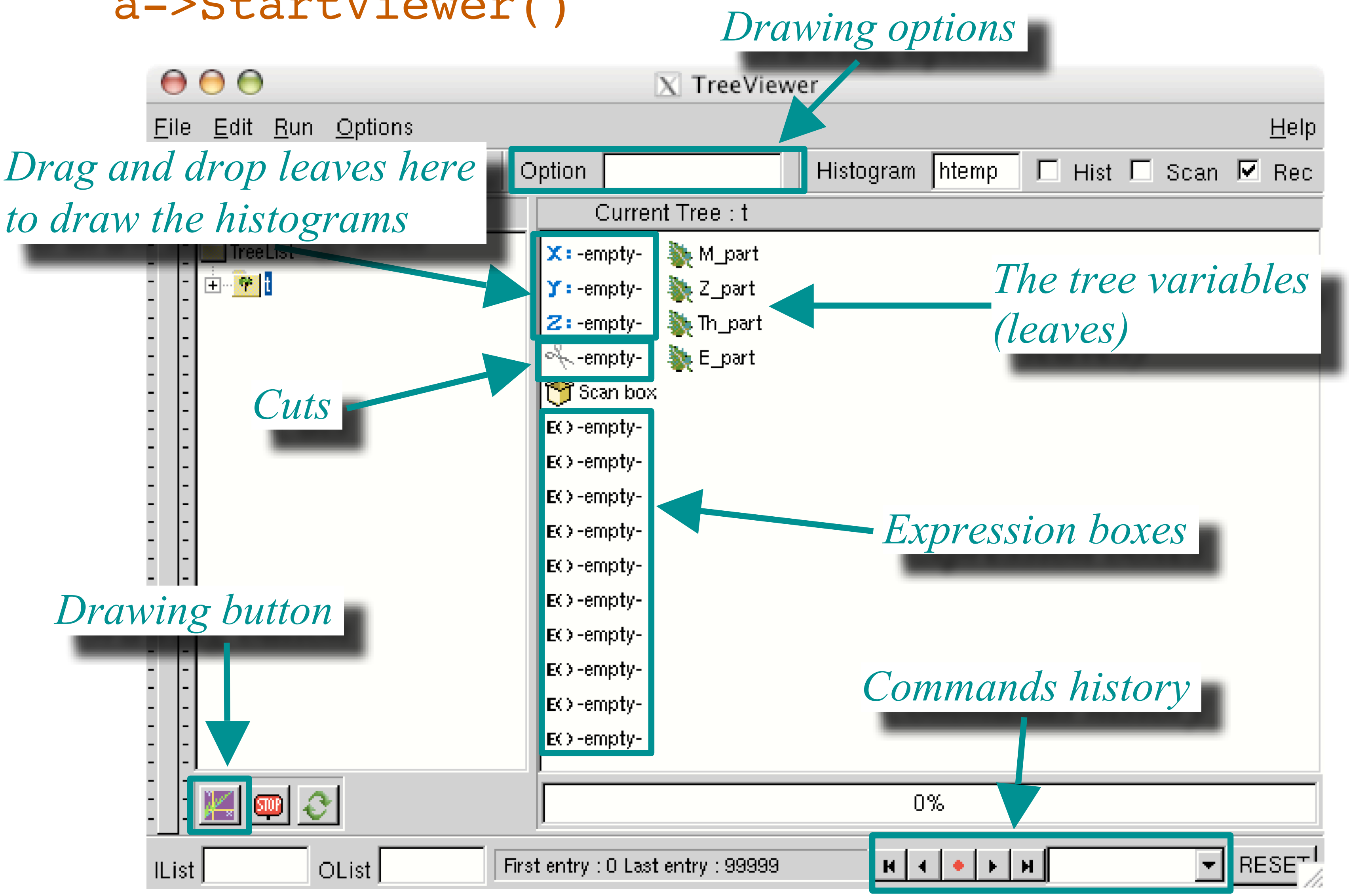
Selection

Event number

```
************************************************
*    Row   *       Mult *      Z[30] * Energie[3 *
************************************************
*       46 *         32 *          2 * 47.778400 *
*       95 *         31 *          2 * 48.006801 *
*      399 *         31 *          1 * 28.520700 *
*      461 *         31 *          2 * 67.939399 *
*      628 *         32 *          2 * 69.046302 *
************************************************
==> 5 selected entries
```

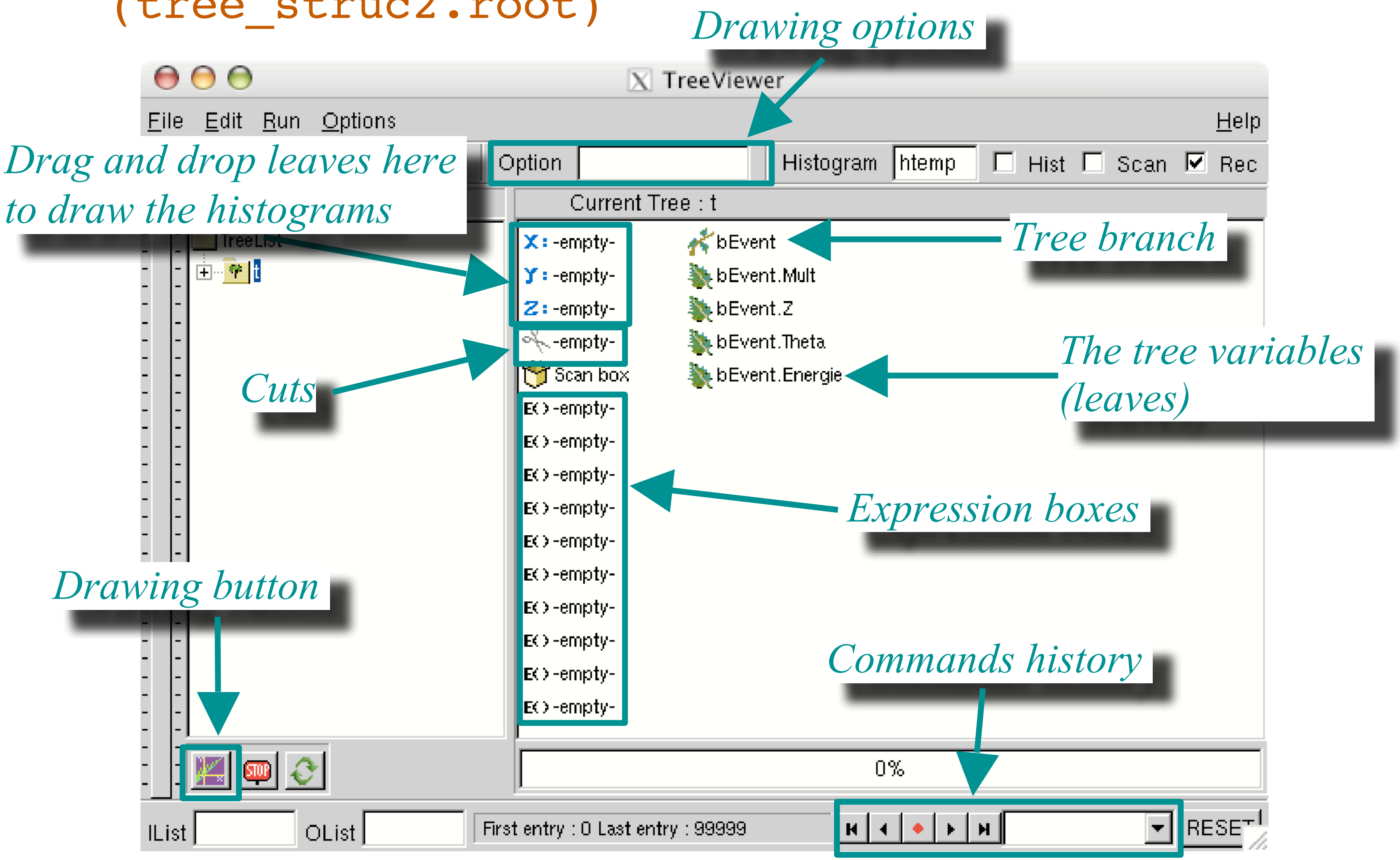
The graphical interface

```
a->StartViewer()
```

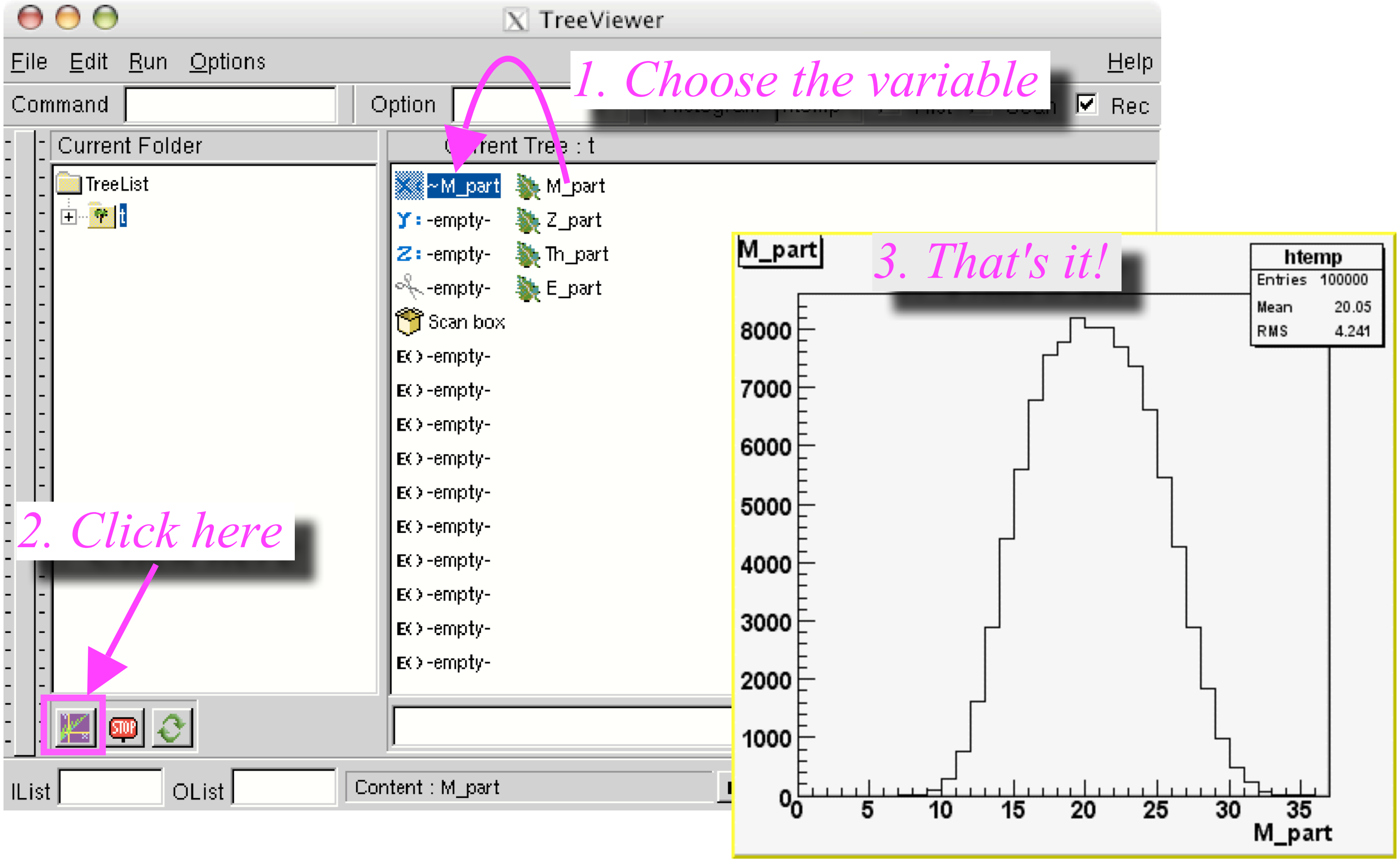


For the single branch tree

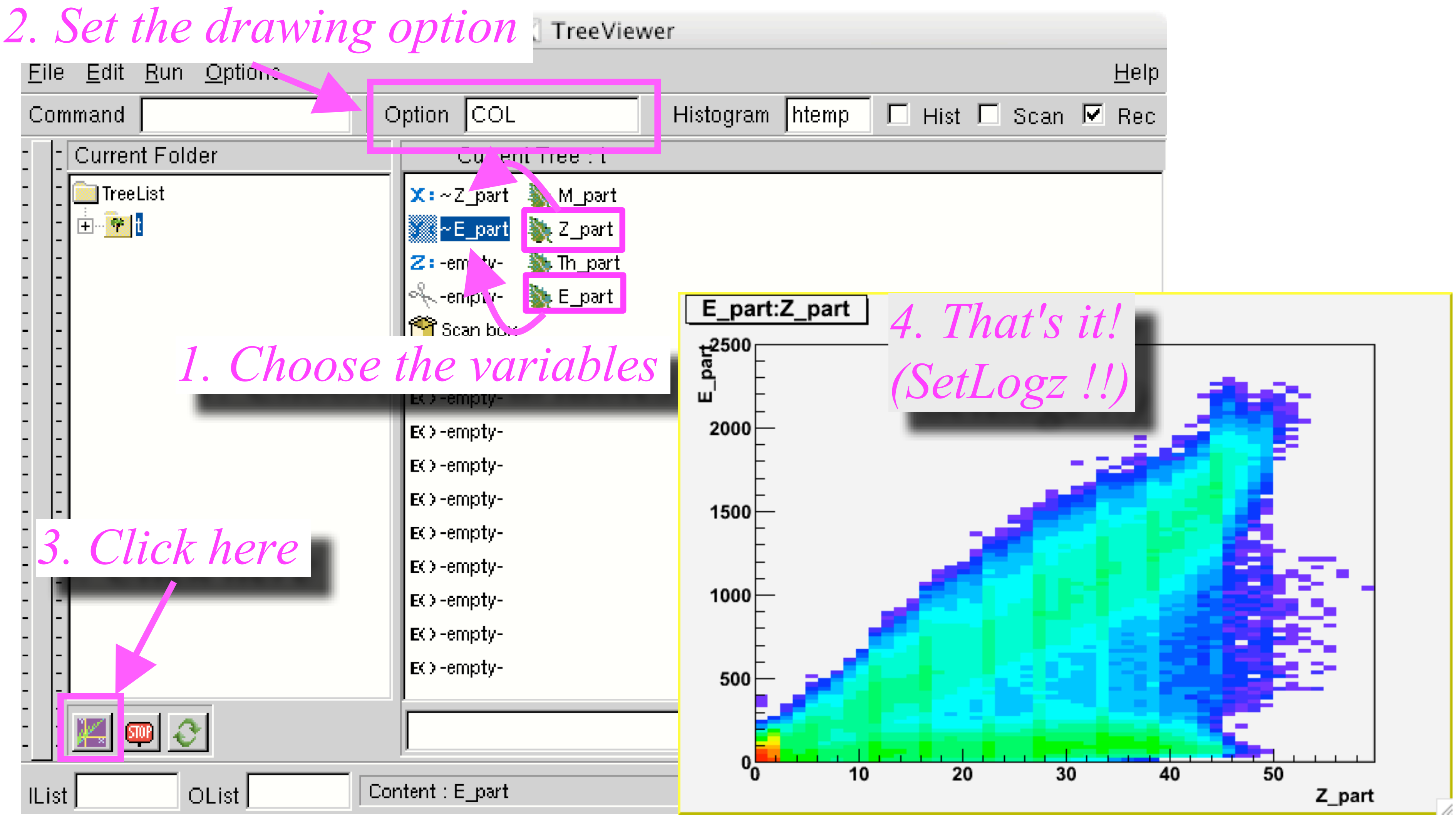
(tree_struc2.root)



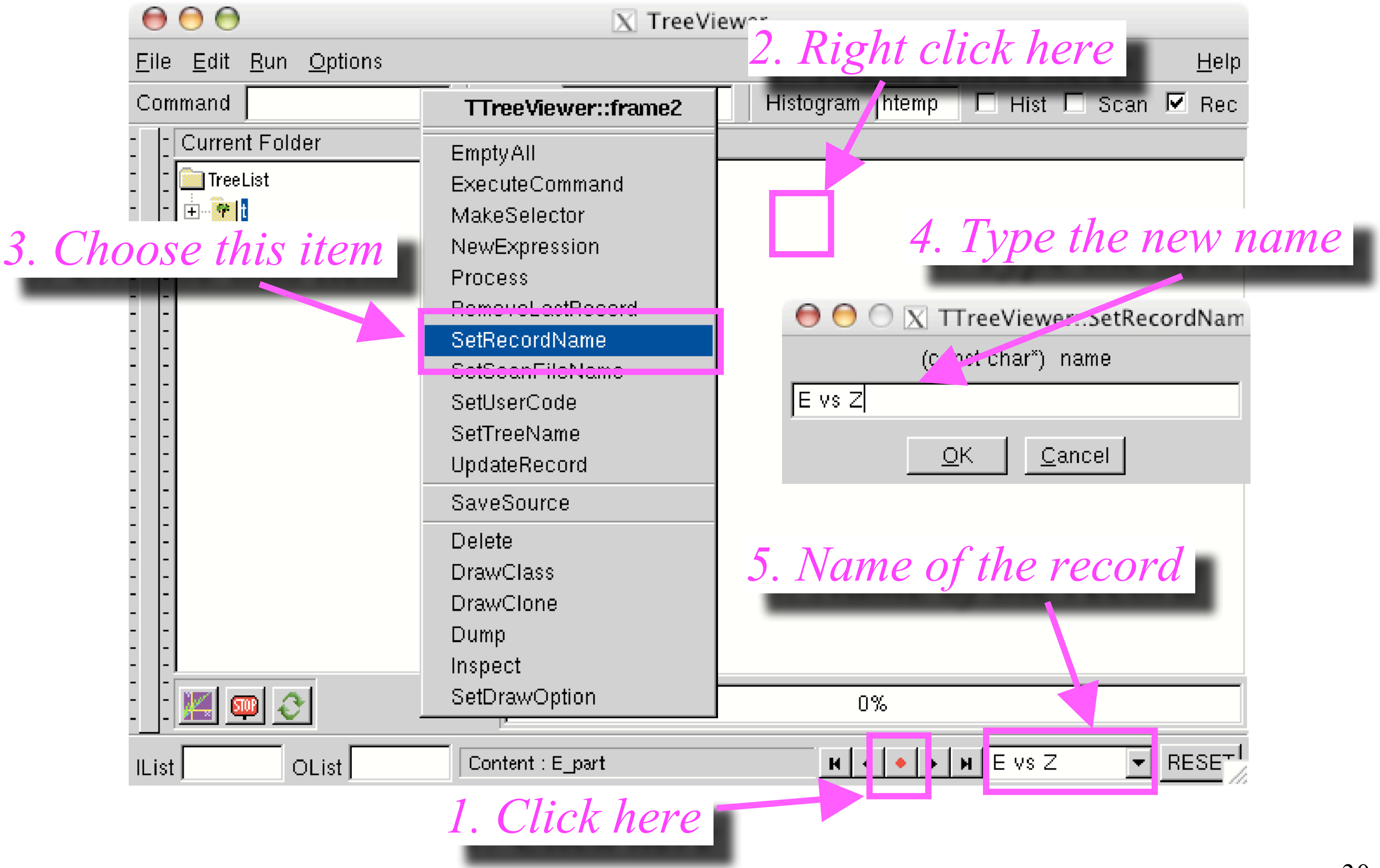
Plotting a 1D histogram



Plotting a 2D histogram



Recording the current display

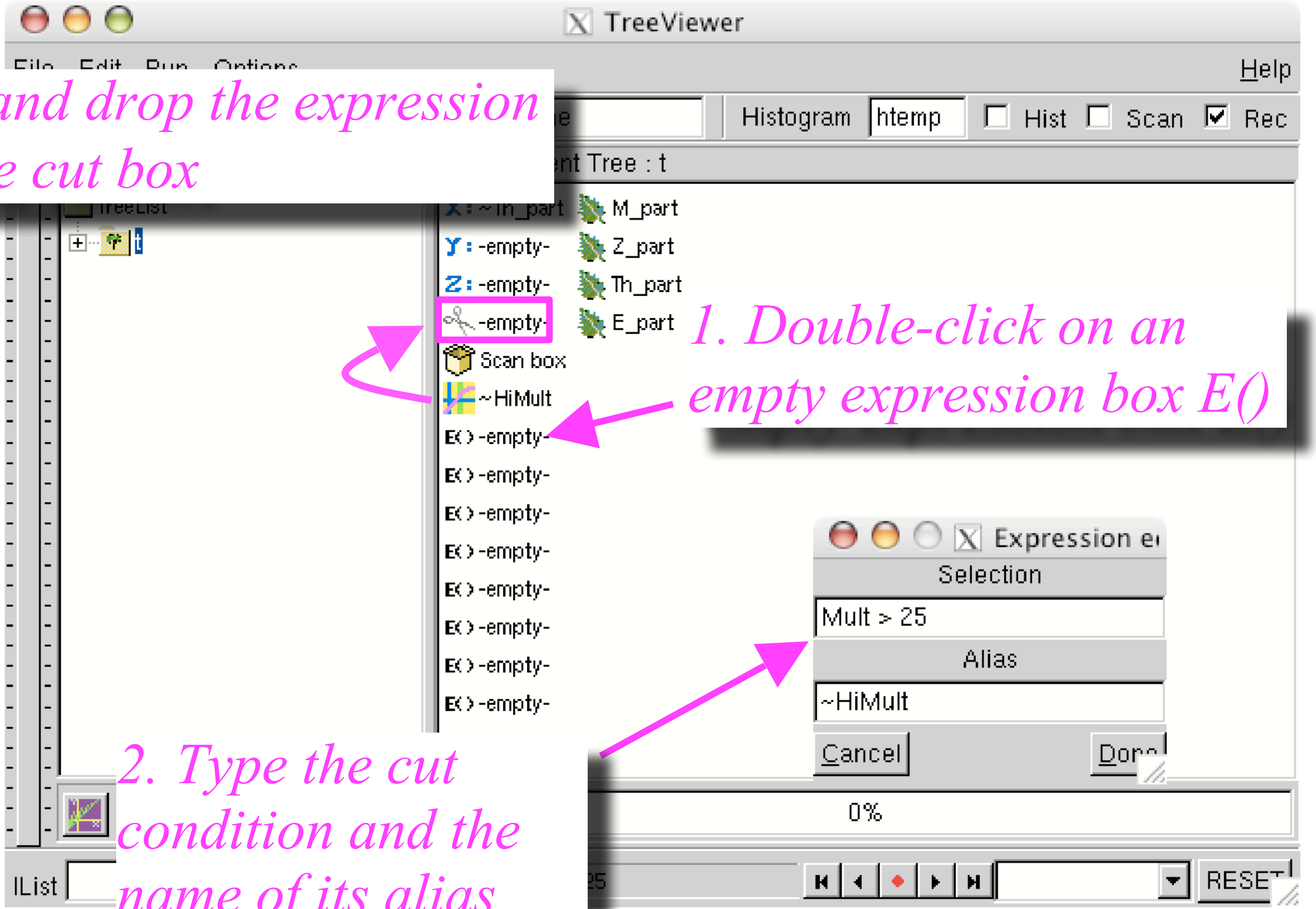


Using cuts

3. Drag and drop the expression box in the cut box

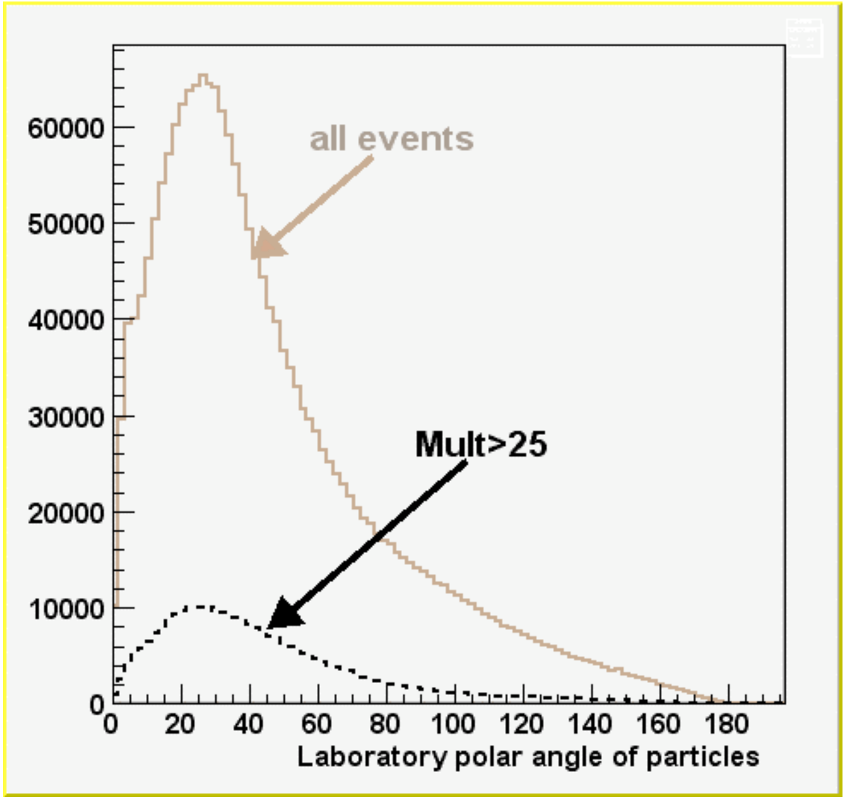
1. Double-click on an empty expression box E()

2. Type the cut condition and the name of its alias

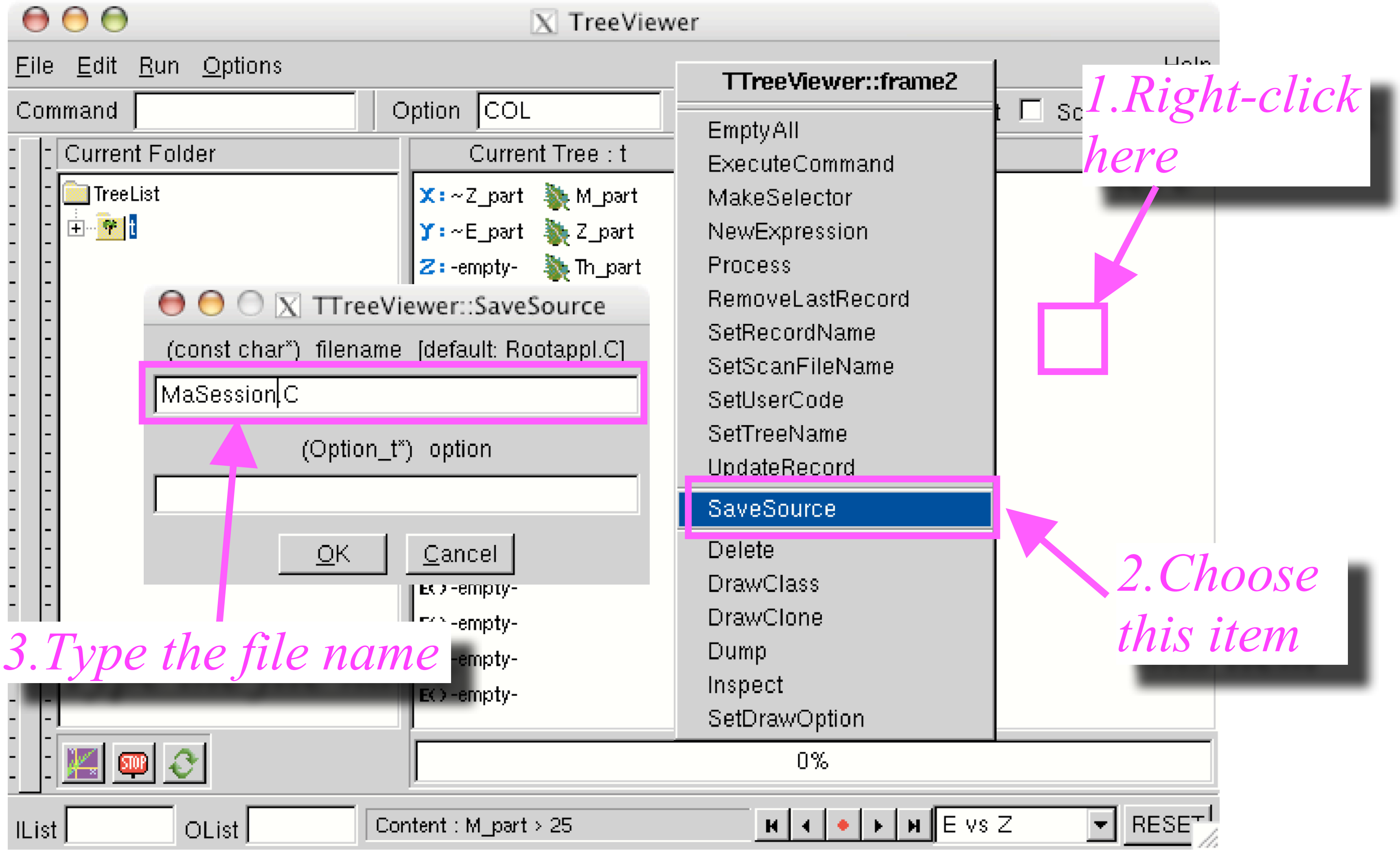


Using cuts (2)

- Drag and drop the **`Th_part`** variable on the *x* axis
- Drag and drop the cut in the "scissors box"
- Double-click on the "scissors box" to disable the cut selection (red line)
- Draw the histogram **without the cut selection**
- Enable the cut selection
- Type `"same"` in the drawing option field
- Draw the histogram **with the cut selection**
- Record the display
- Perfect the presentation of the figure !

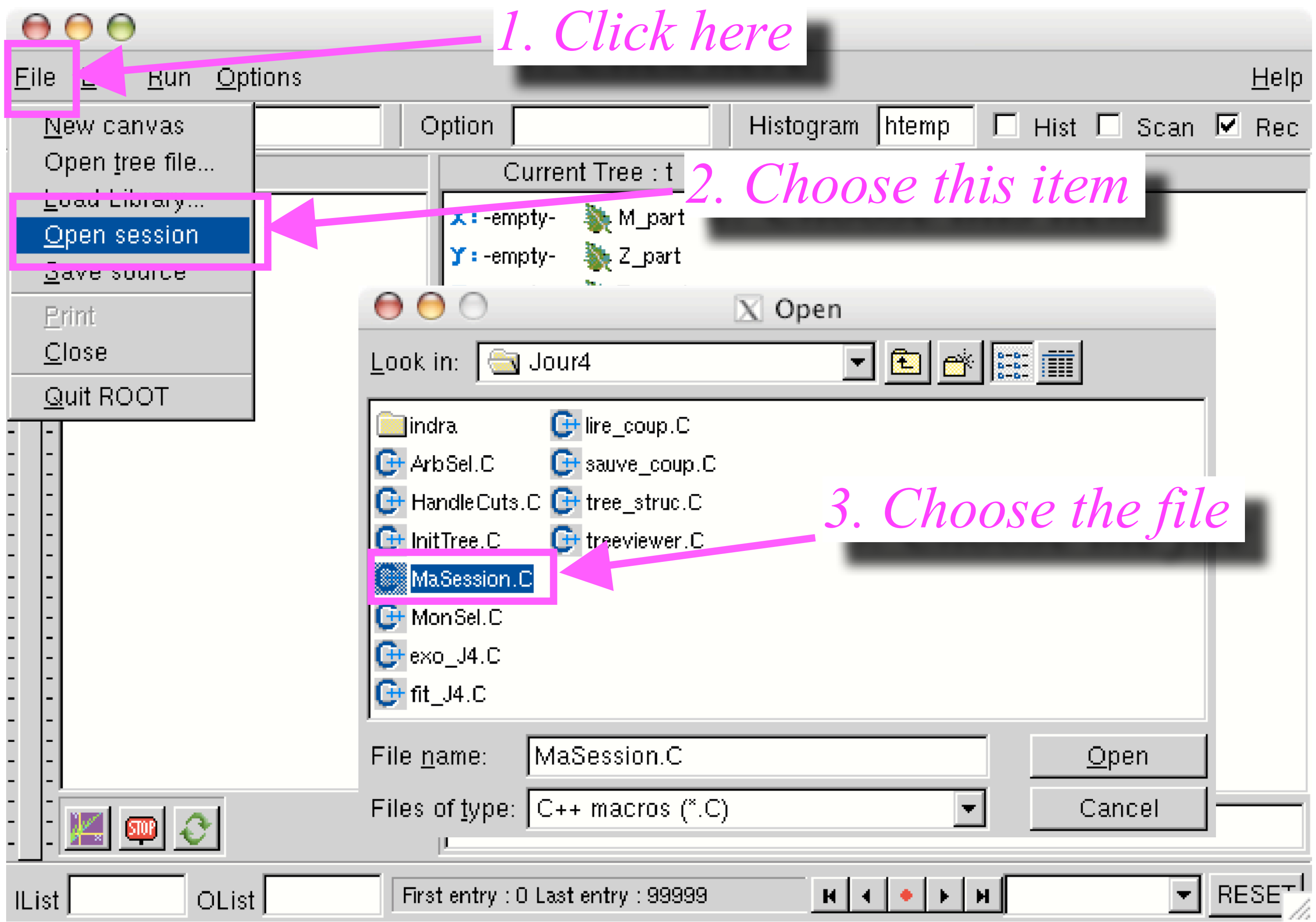


Save it...

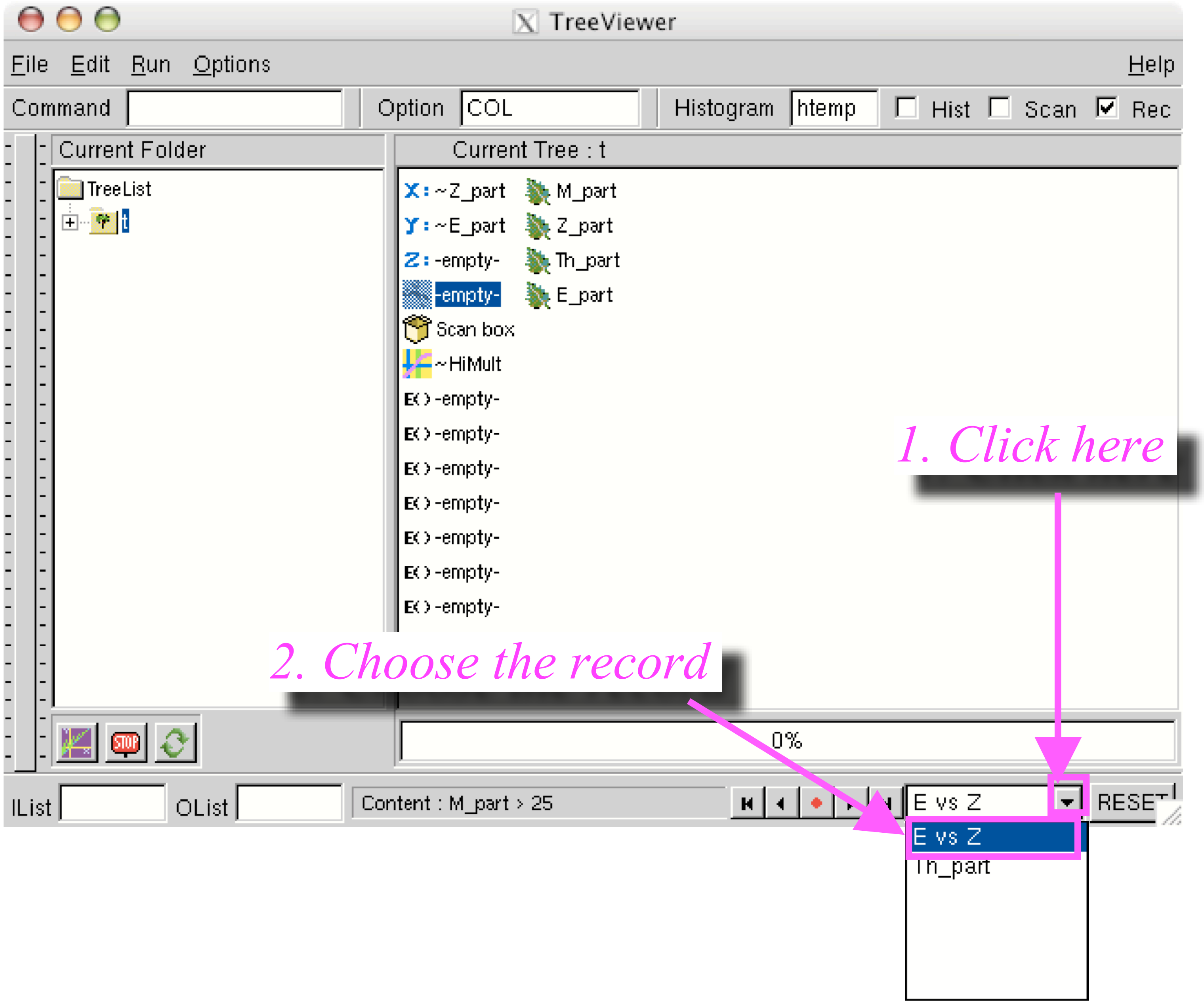


TreeViewer
File
Edit
Run
Options
Command
Option
COL
Current Folder
Current Tree : t
TreeList
X: ~Z_part
Y: ~E_part
Z: -empty-
M_part
Z_part
Th_part
TTreeViewer::frame2
EmptyAll
ExecuteCommand
MakeSelector
NewExpression
Process
RemoveLastRecord
SetRecordName
SetScanFileName
SetUserCode
SetTreeName
UpdateRecord
SaveSource
Delete
DrawClass
DrawClone
Dump
Inspect
SetDrawOption
TTreeViewer::SaveSource
(const char*) filename [default: Rootappl.C]
MaSession.C
(Option_t*) option
OK
Cancel
0%
IList
OList
Content : M_part > 25
E vs Z
1.Right-click here
2.Choose this item
3.Type the file name

Everything is not lost…



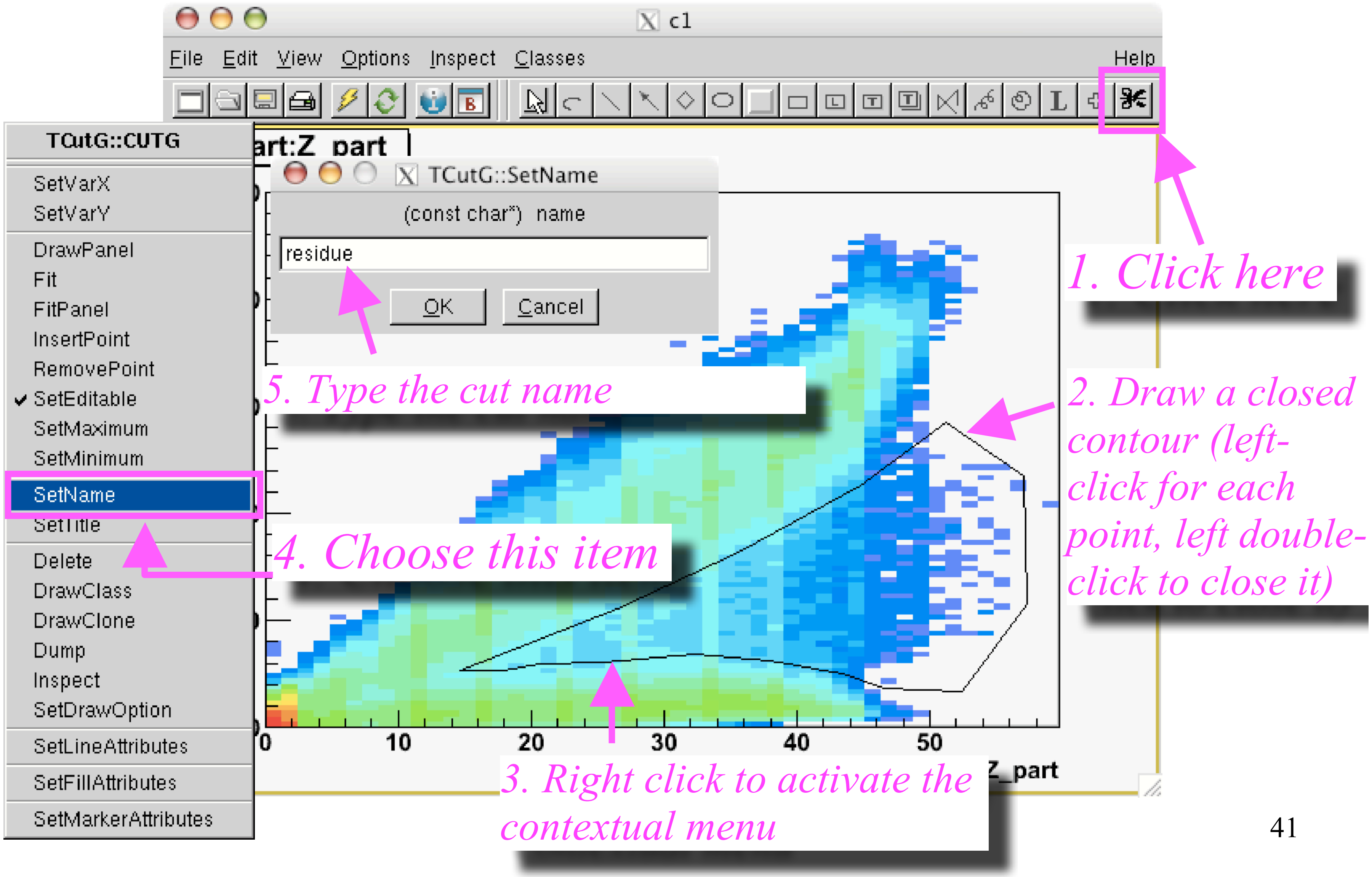
Recalling a recorded display



It's guillotine time: the cut machine

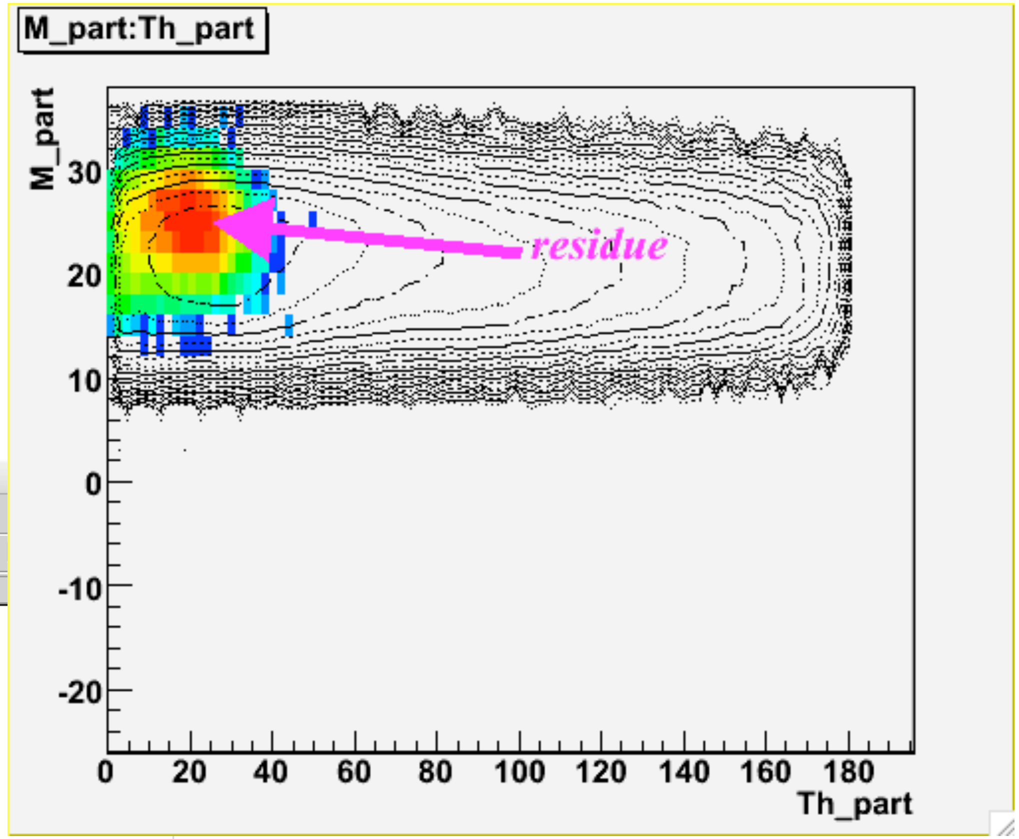
Craphical cuts

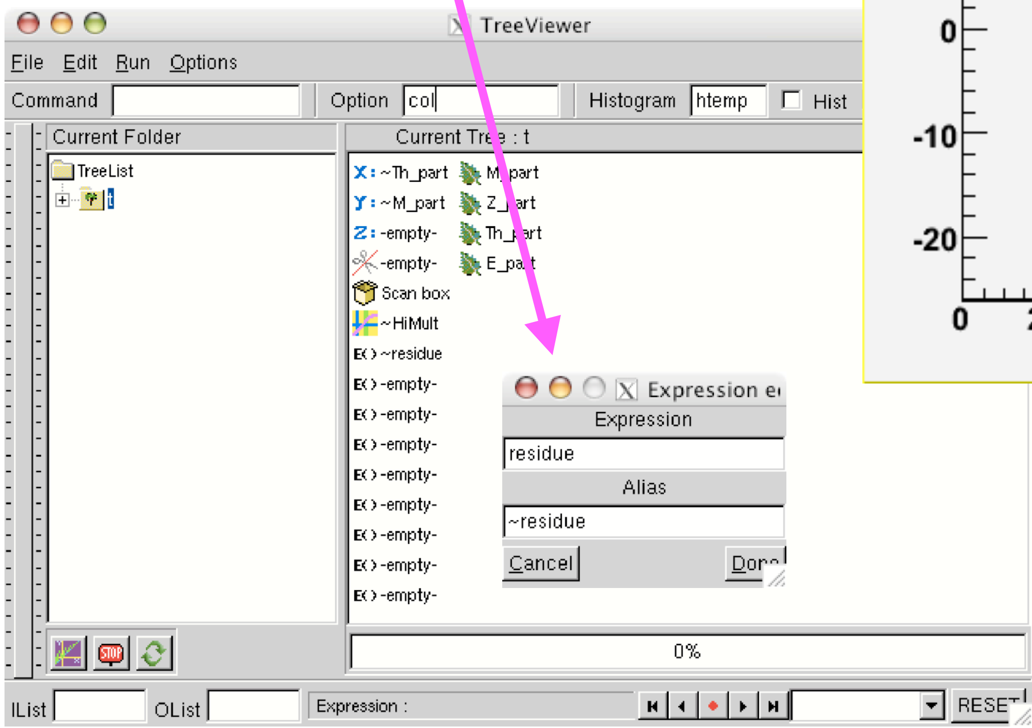
- Open the tool-bar (Canvas menu View->Toolbar)



Using the cut

- When the name of the graphical cut is given to an expression box, this cut can be used to select events...





Mind your fingers: let's mix our cuts

```
a->Draw("Z_part","M_part>30","")
```

- But also…

```
TCut cut1("M_part > 30")
a->Draw("Z_part",cut1,"")
```

- Or…

```
TCut cut2("E_part < 200")
a->Draw("Z_part",cut1 && cut2,"")
```

AND for C++

- For the graphical cuts

```
a->Draw("Z_part",cut1 || "residue" ,"")
```

OR for C++

The Swiss knife…

Variable combinations

- Variables can be combined to define new ones.
- Examples:

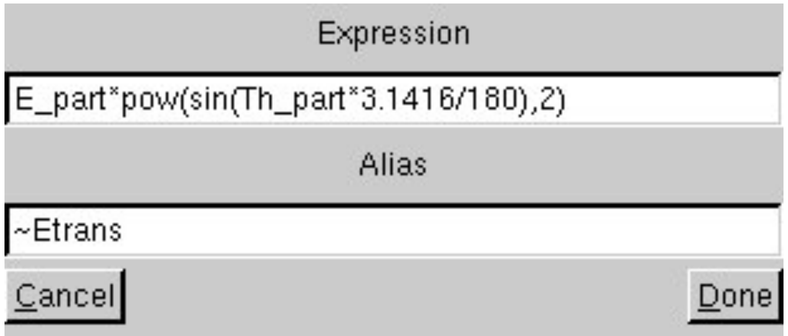
  Draw the parallel velocity component V_z

```
a->Draw("sqrt(E_part/(931.5*Z_part))*cos(Th_part*3.1416/180.)")
```

  Draw the transverse energy as a function of Z

```
a->Draw("E_part*pow(sin(Th_part*3.1416/180.),2):Z_part","","box")
```

- The new variables can be defined in the expression boxes of the TreeViewer



Alias, poor Yorick…

- Pseudo variables (alias) can be defined

  Examples:

  velocity modulus:

```
a->SetAlias("V","sqrt(E_part/(931.5*Z_part))*30")
```

  cosine of the θ angle:

```
a->SetAlias("cost","cos(Th_part*3.1416/180.)")
```

  V_z velocity component

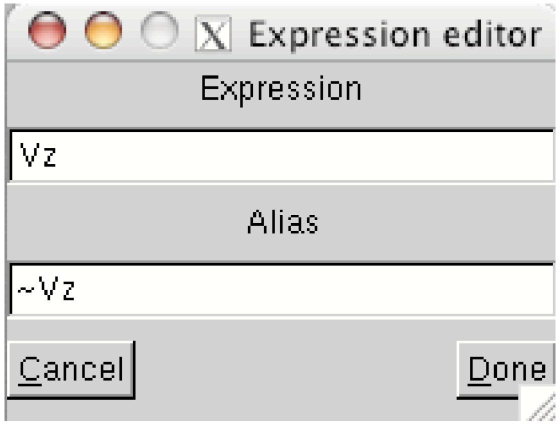
```
a->SetAlias("Vz","V*cost")
```

  Use:

```
a->Draw("Z_part:Vz","Vz>-10","col")
```

- They can be used in the TreeViewer

  BEWARE: an alias from the TreeViewer can not be used with the draw command **`a->Draw()`**



Summing everything…

- Macro-commands can be used with arrays in trees:

Examples:

Sum of products Z*Vz:

```
a->Draw("Sum$(Z*Vz)")
```

Alias Mimf

```
a->SetAlias("Mimf","Sum$(Z>2)")
```

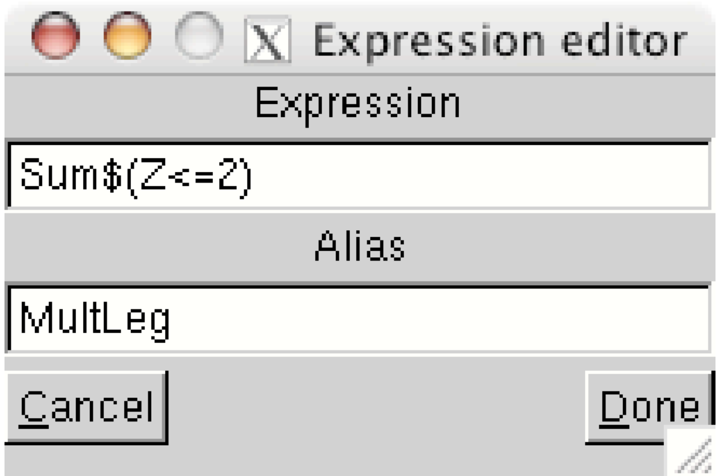
Z	6	1	4	2	Sum$(Z>2)
Z>2	1	0	1	0	2

Alias Transverse Energy of light particles

```
a->SetAlias("Et12","Sum$(E*(1-cost*cost)*(Z<=2))")
```

Use:

```
a->Draw("Mimf:Et12","Sum$(Z>2)>3","col")
a->Draw("Mimf:Et12","Mimf>3","col")
```



- These macro-commands can be used in the TreeViewer
- Have a look at other macro-commands at

http://root.cern.ch/root/html/TTree.html#TTree:Draw

Strings

- Character strings can be passed as arguments of Draw, Scan, SetAlias, GetAlias.

  Examples:

  We want to define alias names "`NewVarX`" as follows:

  "`variableX`-(maximum of the histogram named `HistoX_mono`)"
  for X ranging from 1 to 10

```
Char_t nomAlias[80];
for(Int_t i=1;i<=10;i++)
 {
 sprintf(nomAlias,"NewVar%d",i);
 TString var("variable");
 var+=i;
 TH1 *h=(TH1 *)gROOT->FindObject(Form("Histo%d_mono",i));
 Double_t y=h->GetMaximum();
 a->SetAlias(nomAlias,Form("%s-%f",var.Data(),y));
 }
a->GetListOfAliases()->ls();
```

Projection to a histogram

Creation of the histogram:

```
TH1F *h1=new TH1F("DistZ",
         "Distribution de charge",100,-0.5,99.5)
```

- Projection!

the ***name*** *of the histogram is necessary!*

```
a->Draw("Z_part >> DistZ","M_part>30")
```

ou

```
a->Project("DistZ","Z_part","M_part>30")
```

- Cumulative projection !

```
a->Draw("Z_part >>+DistZ","M_part<=30")
```

or a `"+"` sign before the histogram name in the TreeViewer

The event lists

- These lists can save time if a complex and time consuming cut is applied frequently: only the index numbers of the events corresponding to this cut are recorded in the list!

```
a->Draw(">> listem","M_part>30","")
```

- To use the event list:

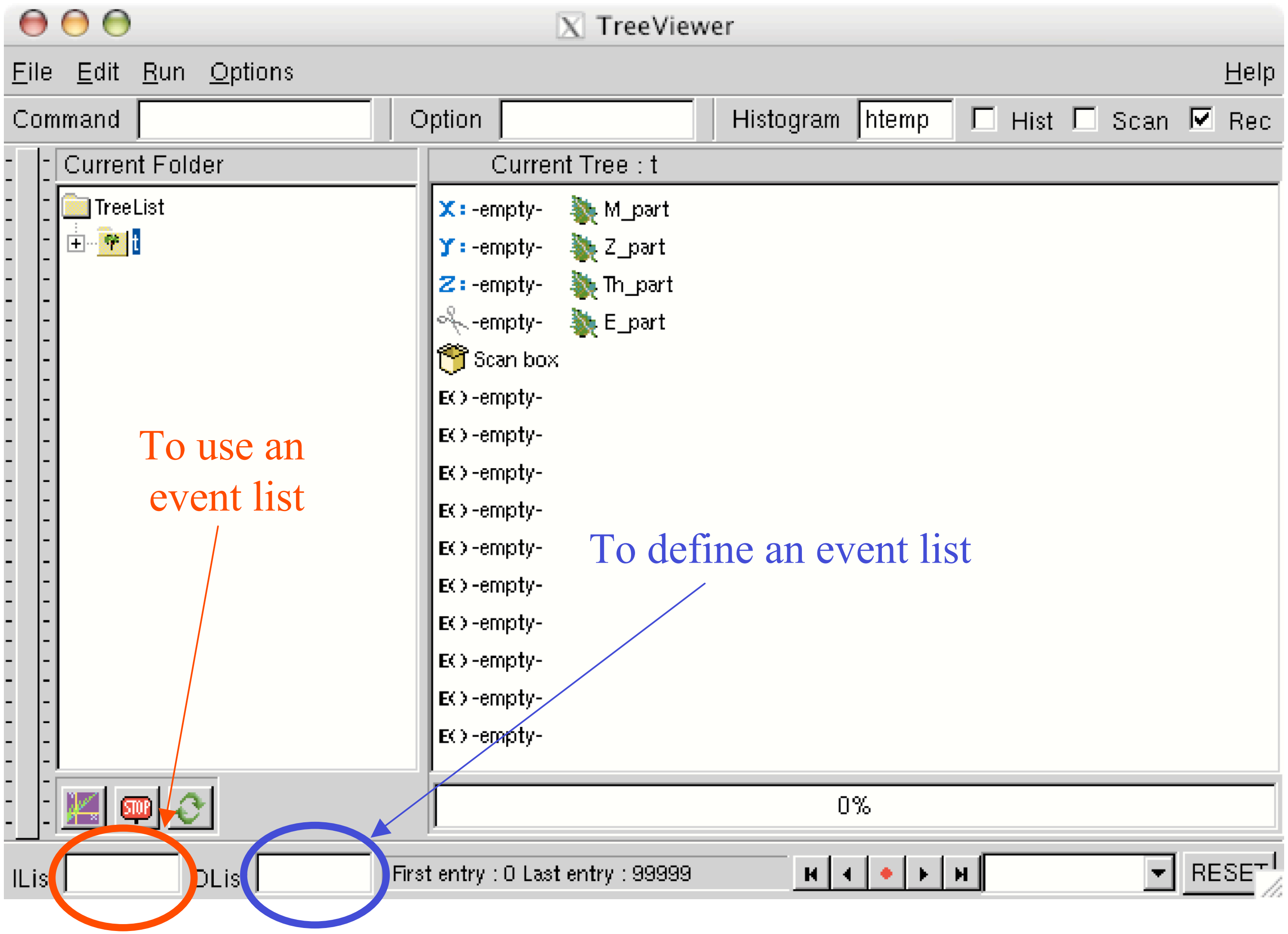
```
TEventList *lm=(TEventList *)gROOT->FindObject("listem")
lm->Print("all")
a->SetEventList(lm)
a->Draw("Z_part")
a->Draw("E_part")
```

← *Printout of all event numbers in the list*

- To remove it from the tree:

```
a->SetEventList(0)
```

The event list in the TreeViewer



Exercise

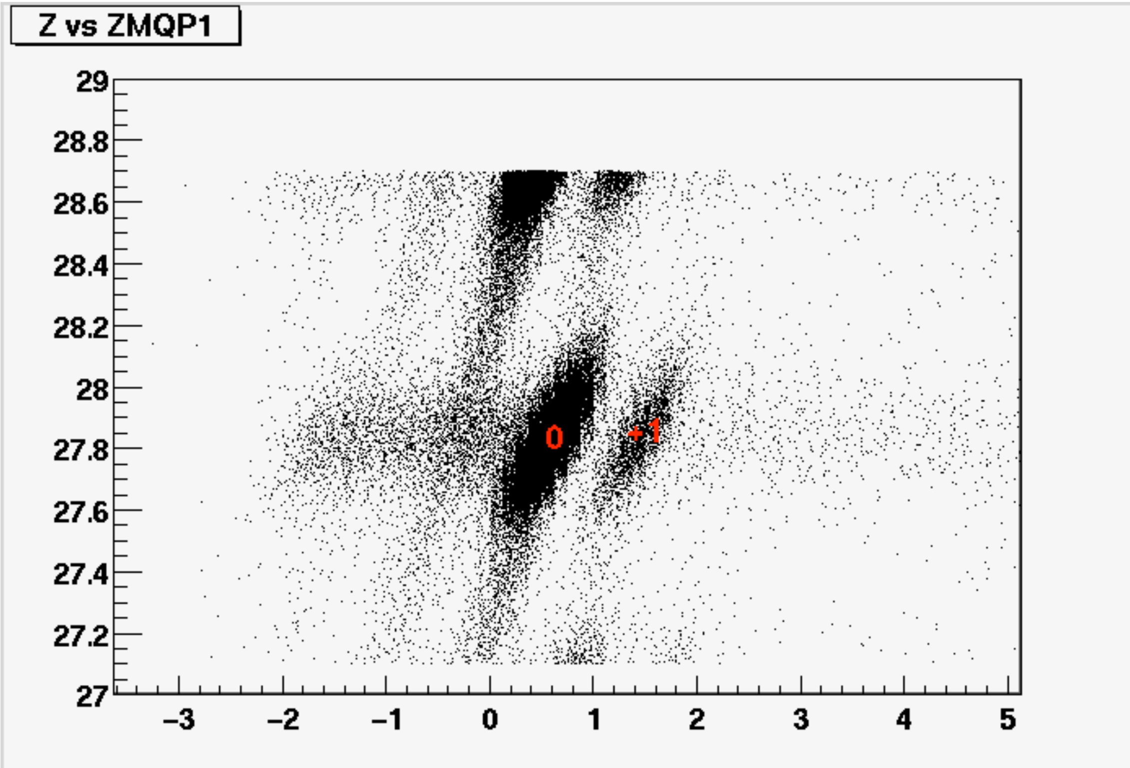
- You will analyse data from a LISE* experiment whose goal is to show the differences between the γ energy spectra for two nickel isotopes. The data are stored in a TTree in the file **`r50_69ni.root`**.
- You will proceed step by step:
  1. Selection of the correct charge state
  2. Selection of the two Ni isotopes.
  3. Calibrate time spectra to build a cumulative histogram.
  4. Building the γ energy spectra for both isotopes.

`http://caeinfo.in2p3.fr/root/Formation/en/Day4/r50_69ni.root`

**Thanks to M.Sawicka, F.De Oliveira and J.M.Daugas!*

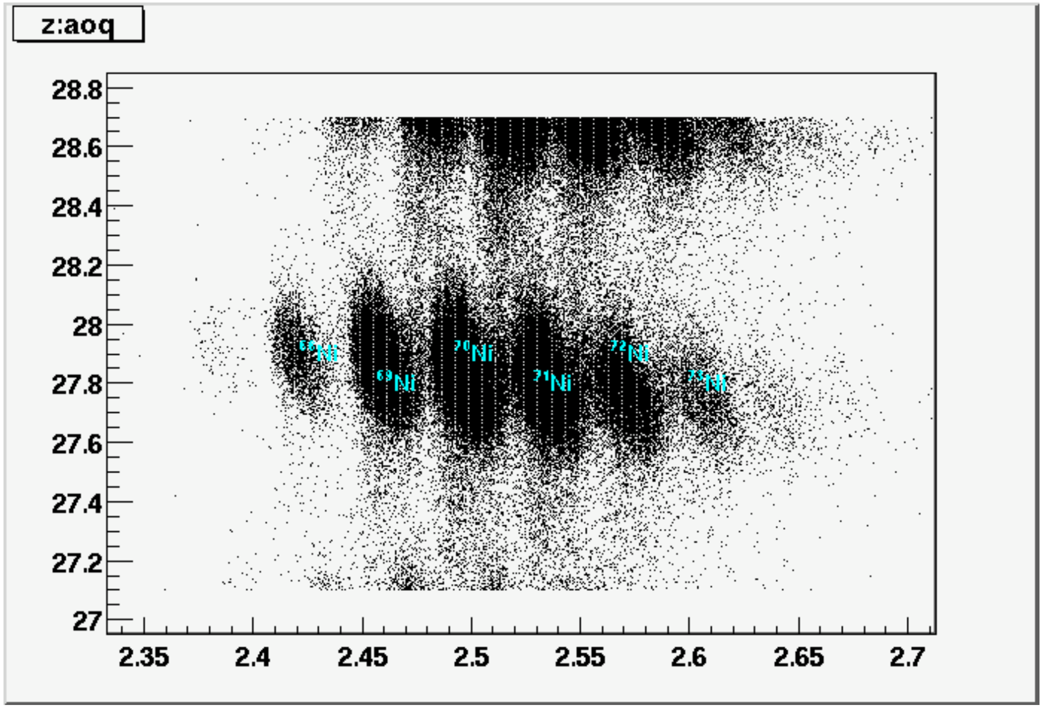
Exercise: Step 1

- Selection of the charge state
  - Build the histogram `z` versus `zmqp1`.
  - Build a graphical cut named CUTEC around the accumulation of data centred at (0.5,27.8)



Exercise: Step 2

- Selection of the Ni isotopes:
  - Build the histogram **z** versus **aoq**
  - Build a graphical cut named CUTNI69 around the area centred at (2.45,27.9)
  - Build a graphical cut named CUTNI70 around the area centred at (2.5,27.9)



Saving the cuts

```
#include "TFile.h"
#include "TCUTG.h"

void SaveCuts(void)
{
TFile *fcoup=new TFile("coupures.root","recreate");
fcoup->cd();
gROOT->FindObject("CUTEC")->Write();
gROOT->FindObject("CUTNI69")->Write();
gROOT->FindObject("CUTNI70")->Write();
fcoup->Close();
}
```

http://caeinfo.in2p3.fr/root/Formation/en/Day4/HandleCut.C

To retrieve the cuts

```
void LoadCuts(void)
{
TFile *fcoup=new TFile("coupures.root");
TCutG *CUTEC=(TCUTG *)fcoup->Get("CUTEC");
TCutG *CUTNI69=(TCUTG *)fcoup->Get("CUTNI69");
TCutG *CUTNI70=(TCUTG *)fcoup->Get("CUTNI70");
fcoup->Close();
}
```

Use:

```
root[0] .L HandleCuts.C+
root[1] SaveCuts()
root[2] LoadCuts()
```

`root[1] SaveCuts()` *to save them*

`root[2] LoadCuts()` *to load them*

Exercise: Step 3

- Calibrating the « long » time spectra.
  - Build the histogram of **`tg1lo`** for values of **`tg1lo`** lower than 3000
  - Locate the abscissa `T1M` of the spectrum's maximum
  - Build the alias named `RTG1LO` = **`tg1lo`** -`T1M`
  - Repeat the same procedure for the 5 other variables **`tgxlo`** for **`x`** ranging from 2 to 6.

Exercise: Step 4

- **Build the following histograms for the charge state 0**
  - Cumulative histogram of spectra **Egxc** for **x** ranging from 1 to 6
  - Same histogram for ^{69}Ni alone
  - Same histogram for ^{70}Ni alone
  - Superimpose these histograms
  - Conclusions?
- **Build the following histograms for the charge state 0**
  - Cumulative histogram of spectra **Egxc** vs RTGXLO for **x** ranging from 1 to 6 for the ^{70}Ni.
  - Make the projections of this histogram on the time axis for the two most intense energy peaks ($E_\gamma \approx 183$ keV et $E_\gamma \approx 447$ keV)
  - Extract from these projections the half-life of these two γ peaks (using a fit function being the sum of a constant and an exponential)
  - Conclusions?