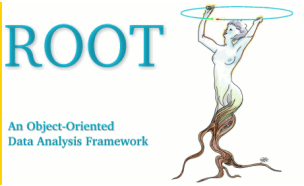
ROOT (2)

ROOT

An Object-Oriented
Data Analysis Framework

http://root.cern.ch/

Τι είναι ROOT



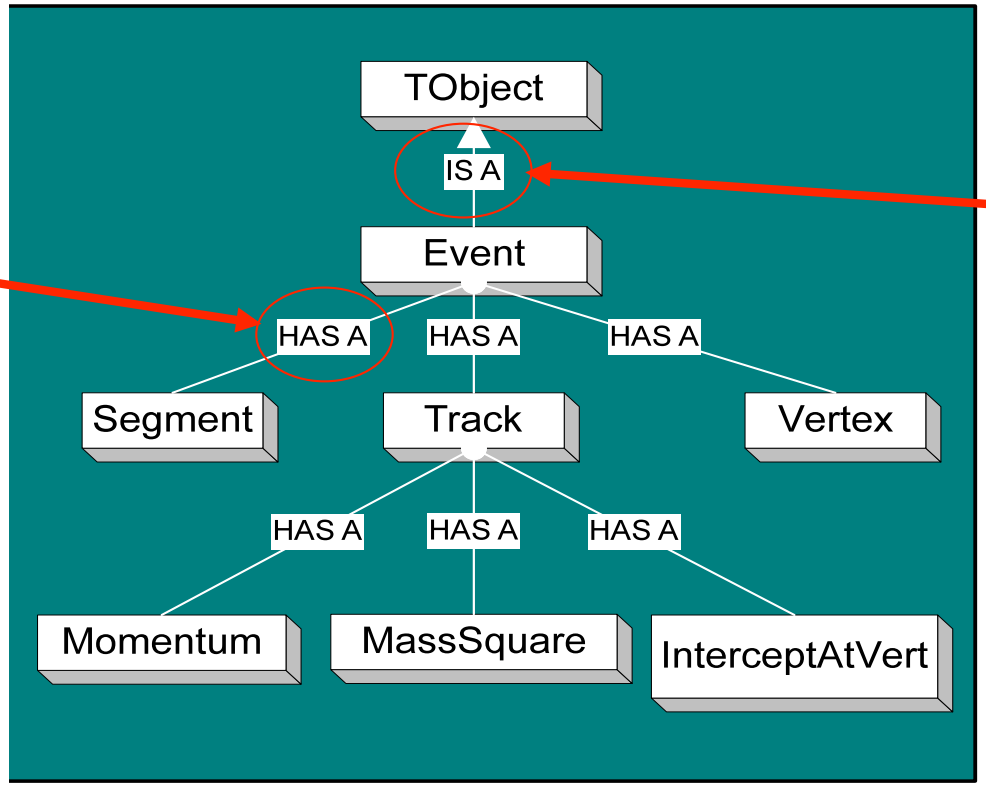
- **Ένα σύνολο αντικειμένων τα οποία μπορούν να χρησιμοποιηθούν για την ανάπτυξη εφαρμογών (Libraries).**

- **Μια διαδραστική (interactive) εφαρμογή μέσω της οποίας ο χρήστης έχει πρόσβαση σε όλες τις τάξεις του ROOT, έχοντας στη διάθεσή του ένα μεταφραστή (Interpreter) γραμμής εντολών.**

- **Ο μεταφραστής είναι ένας C++ interpreter**

Βασικές ιδέες προγρ/σμού ΟΟ

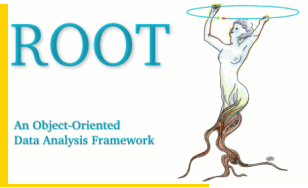
- **Τάξη (Class):** **περιγραφή ενός «πράγματος» στο σύστημα**
- **Αντικείμενο (Object):** **ένα στιγμιότυπο μιας τάξης**
- **Μέθοδος (Methods):** **συνάρτηση μιας τάξης**



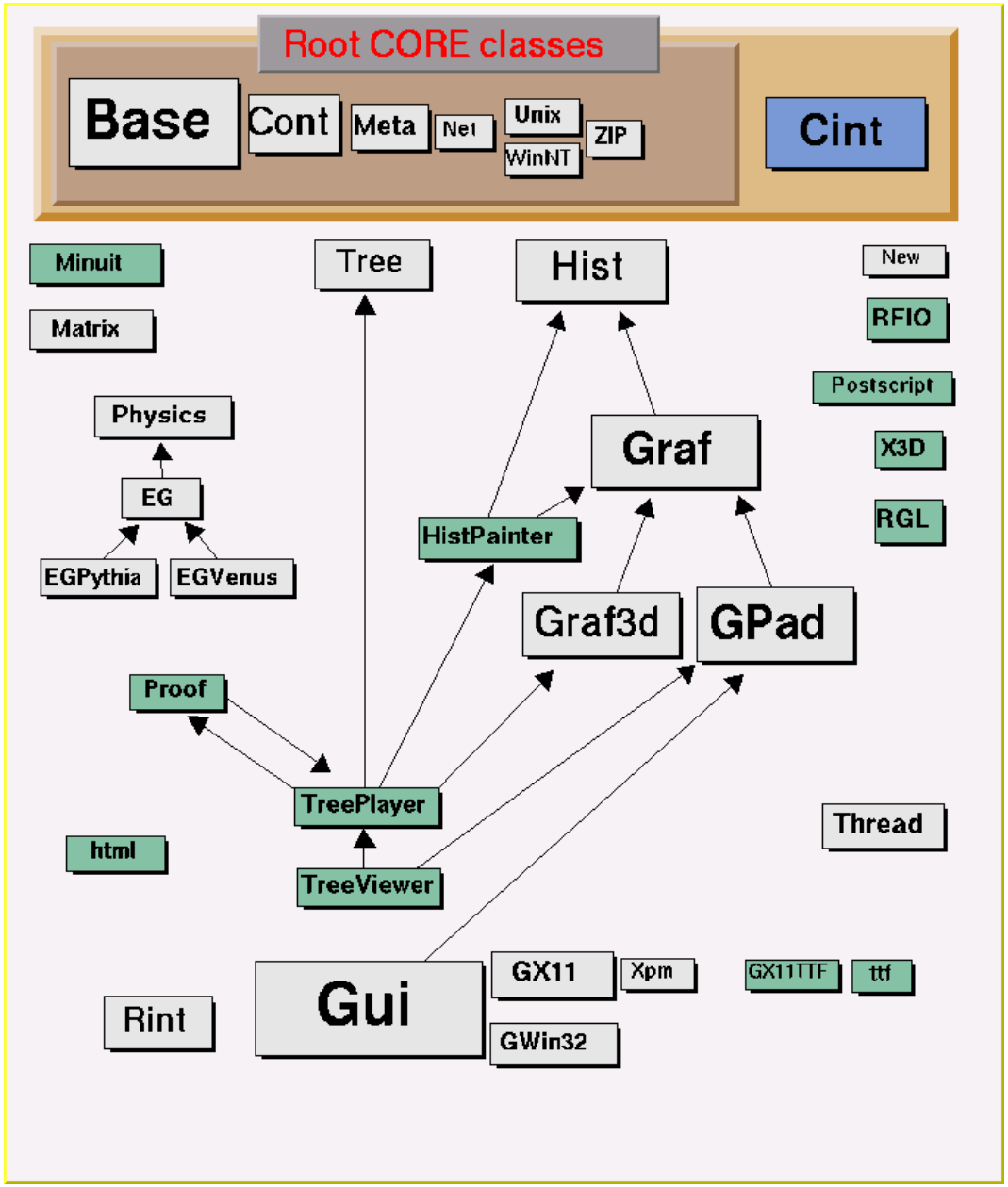
Κληρονομικότητα (Inheritance)

Μέλη μιας τάξης (Members)

ROOT's Services/Utilities



- Histogramming and Fitting
- Graphics (2D, 3D)
- I/O to file or socket: specialized for histograms, Ntuples (Trees)
- Collection Classes and Run Time Type Identification
- User Interface
  - GUI: Browsers, Panels, Tree Viewer
  - Command Line interface: C++ interpreter CINT
  - Script Processor (C++ compiled ⇔ C++ interpreted)

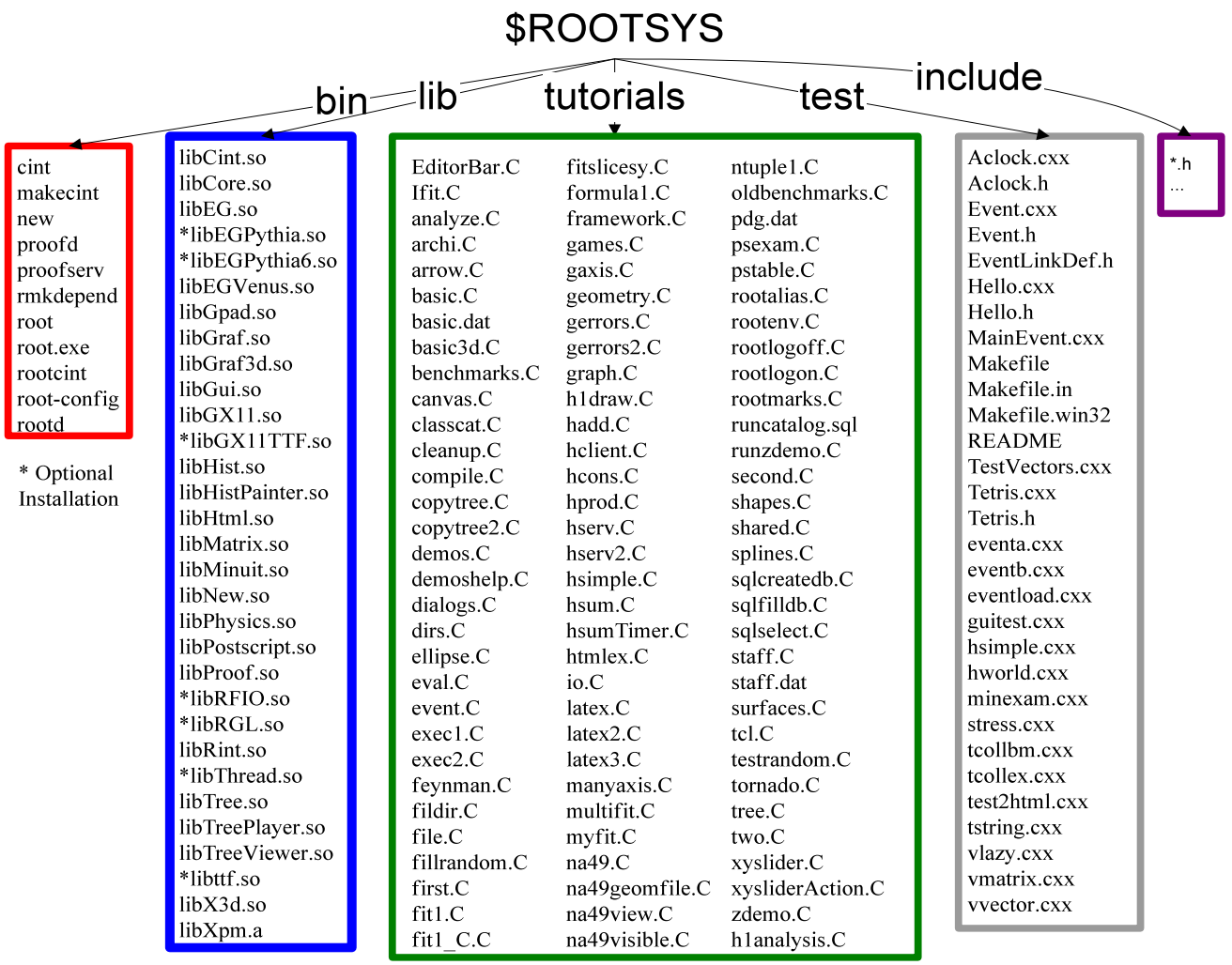


The Libraries

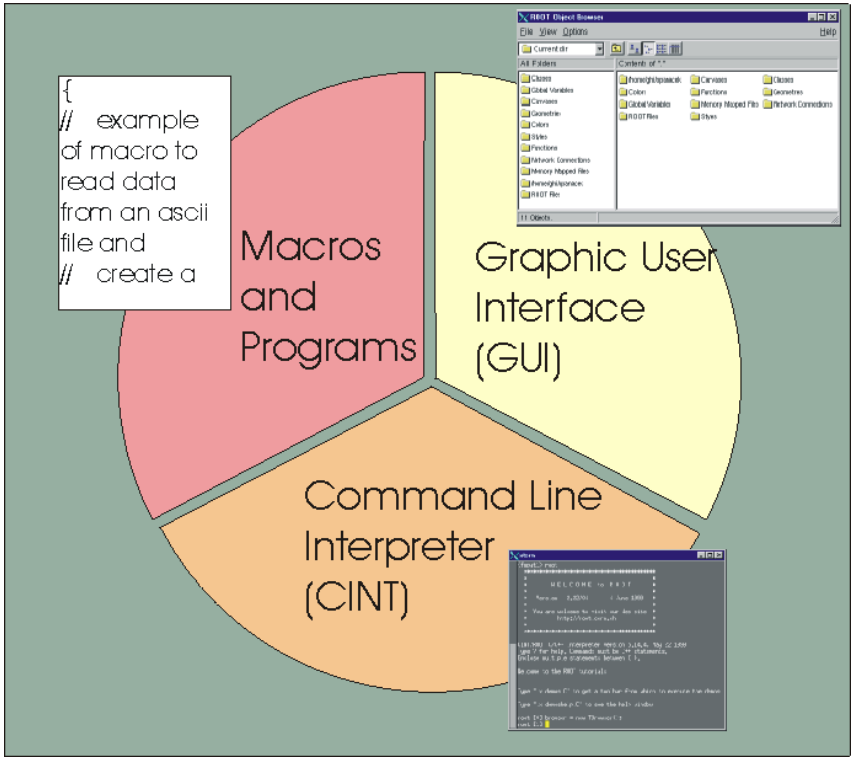
- Over 350 classes

- Core
- CINT
- Libraries loaded at startup: Hist, Tree …
- Libraries loaded when needed: HistPainter, TreePlayer,…
- Special purpose libraries: EG, Physics…

Οργάνωση του ROOT

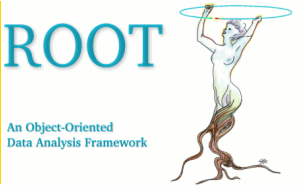


Three User Interfaces



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

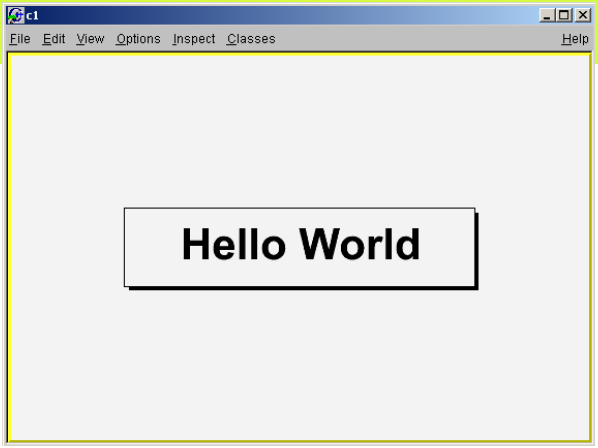
ROOT: C/C++ Interpreter



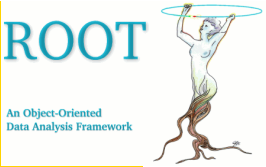

```
$ root
root[0] printf("Hello World\n");
Hello World
root [1]
```

Το ίδιο με λίγα graphics...

```
$ root
root[0] TPaveLabel hello(0.2,0.4,0.8,0.6,"Hello World")
root[1] hello.Draw()
```



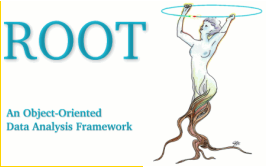
ROOT : C/C++ interpreter



Γραμμή Εντολών

```
root [2] 2+2
(const int)4
root [3] 3+5
(const int)8

root [12] float x = 43.9
root [13] float y = 22.1
root [14] float z = x+y
root [15] x
(float)4.39000015258789060e+001
root [16] y
(float)2.21000003814697270e+001
root [17] z
(float)6.60000000000000000e+001
```

ROOT : C/C++ interpreter



Γραμμή Εντολών

```
root[19] for (int i = 0; i<4; i++) cout << "Hello " << i << endl
Hello 0
Hello 1
Hello 2
Hello 3
root[23]

root[23] 1+sqrt(9)
(const double)4.0000000000000000e+00
```

ROOT : C/C++ interpreter

Μπορείτε να δώσετε από το πληκτρολόγιο περισσότερες εντολές για να εκτελεστούν μαζί κλείνοντας τες σε { }.

```
root[] {
end with '}'> Int_t j = 0;
end with '}'> for (Int_t i = 0; i < 3; i++)
end with '}'> {
end with '}'> j= j + i;
end with '}'> cout << "i = " << i << ", j = " << j << endl;
end with '}'> }
end with '}'> }
i = 0, j = 0
i = 1, j = 1
i = 2, j = 3
```

ROOT : συμβάσεις

Identifier	σύμβαση	παράδειγμα
Classes	αρχίζει με T	THashTable
Non-class types	τελειώνει με _t	Simple_t, Int_t
Enumeration types	αρχίζει με E	EColorLevel
Data members	αρχίζει με f	fViewList
Member functions	αρχίζει με κεφαλαίο	Draw()
Static variables	αρχίζει με g	gSystem
Static data members	αρχίζει με fg	fgTokenClient
Locals and parameters	αρχίζει με μικρό γράμμα	seed, thePad
Constants	αρχίζει με k	kInitialSize, kRed
Template arguments	αρχίζει με A	AType
Getters and setters	αρχίζει με Get, Set, ή Is (boolean)	SetLast(), Get-First(), IsDone()

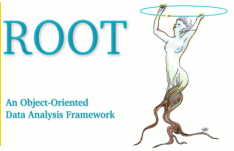
ROOT : τύποι δεδομένων

Βασικοί τύποι δεδομένων (Machine Independent)

Χρησιμοποιώντας αυτούς τους τύπους δεδομένων είμαστε βέβαιοι για τη φορητότητα του κώδικας μας.

· Char_t Signed	1 byte
· UChar_t Unsigned	1 byte
· Short_t Signed short integer	2 bytes
· UShort_t Unsigned short integer	2 bytes
· Int_t Signed integer	4 bytes
· UInt_t Unsigned integer	4 bytes
· Long_t Signed long integer	8 bytes
· ULong_t Unsigned long integer	8 bytes
· Float_t Float	4 bytes
· Double_t Float	8 bytes
· Bool_t Boolean	(kFALSE, kTRUE)

Βοήθεια με TAB!



Χρησιμοποιώντας το πλήκτρο TAB έχουμε βοήθεια …

- `root [0] b = new TB <TAB>`
- `root [1] b = new TBrow<TAB>`
- `root [2] b = new TBrowser(<TAB>`

Βρίσκουμε τη λίστα των μεθόδων (συναρτήσεων)
Βρίσκουμε τη λίστα των παραμέτρων

ROOT : C/C++ interpreter


```
root [0] TLine l
root [1] l.Print()
TLine X1=0.000000 Y1=0.000000 X2=0.000000 Y2=0.000000
root [2] l.SetX1(10)
root [3] l.SetY1(11)
root [4] l.Print()
TLine X1=10.000000 Y1=11.000000 X2=0.000000 Y2=0.000000

root [5] .g
...
...
0x4038f080 class TLine l , size=40
0x0 protected: Coord_t fX1 //X of 1st point
0x0 protected: Coord_t fY1 //Y of 1st point
0x0 protected: Coord_t fX2 //X of 2nd point
0x0 protected: Coord_t fY2 //Y of 2nd point
0x0 private: static class TClass* fgIsA
```

Τέλος εντολής ; δεν απαιτείται.
Οι εντολές του interpreter ξεκινούν με . (dot).

.g εκτυπώνει όλα τα αντικείμενα που είναι ορισμένα

ROOT : C/C++ interpreter

```
root [6] .class TLine
==========================================
class TLine //A line segment
size=0x28
List of base class----------------------
0x0 public: TObject //Basic ROOT object
0xc public: TAttLine //Line attributes
List of member variable------------------
Defined in TLine
0x0 protected: Coord_t fX1 //X of 1st point
0x0 protected: Coord_t fY1 //Y of 1st point
0x0 protected: Coord_t fX2 //X of 2nd point
0x0 protected: Coord_t fY2 //Y of 2nd point
0x0 private: static class TClass* fgIsA
List of member function-------------------------
Defined in TLine
filename line:size busy function type and name
(compiled) 0:0 0 public: class TLine TLine(void);
(compiled) 0:0 0 public: Coord_t GetX1(void);
(compiled) 0:0 0 public: Coord_t GetX2(void);
...
...
(compiled) 0:0 public: virtual void SetX1(Coord_t x1);
(compiled) 0:0 public: virtual void SetY2(Coord_t y2);
(compiled) 0:0 0 public: void ~TLine(void);
```

.class

Παρουσιάζει τον ορισμό της τάξης (class definition)

Χρησιμοποιείται για γρήγορη βοήθεια

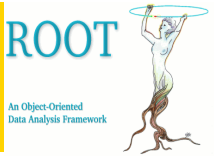
ROOT : C/C++ interpreter


```
root [7] l.Print(); > test.log
root [8] l.Dump(); >> test.log
root [9] ?
root [9] .! ls
```

> ή >>	Αλλαγή κατεύθυνσης (απαιτείται ; πριν από το >)
?	Βοήθεια
.!	Εντολές προς το shell

Για περισσότερες λεπτομέρειες:
http://root.cern.ch/root/CintInterpreter.html

ROOT script files (macros) 1



Τα αρχεία script του ROOT περιέχουν κώδικα C++. Αυτός μπορεί να είναι

- Σειρά απλών εντολών (Un-Named script)
- Ορισμός σύνθετων τάξεων και συναρτήσεων (Named script)

Un-Named script

Ένα αρχείο με μια σειρά από εντολές ανάμεσα σε { }, αποτελεί για τη ROOT "μη επώνυμο" αρχείο script και μπορεί να εκτελεστεί δίνοντας

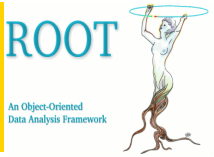
```
root[] .x filename.ext
```

script1.c

```
{
   cout << " Hello" << endl;
   float x = 3.;
   float y = 5.;
   int   i = 101;
   cout <<" x = "<<x<<" y = "<<y<<" i =
"<<i<< endl;
}
```

```
root[] .which script1.c
```

ROOT script files (macros) 2



Αντιγράψτε το script1.c σε script2.c και προσθέστε τη συνάρτηση my_script().

```
root[] .L script2.c
root[]  my_script2()
```

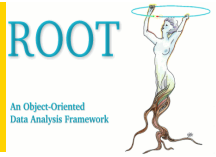
Η πρώτη εντολή φορτώνει το αρχείο και η δεύτερη εκτελεί τη συνάρτηση `my_script2()`

script2.c

```
int my_script()
{
cout << " Hello" << endl;
   float x = 3.;
   float y = 5.;
   int   i = 101;
   cout <<" x = "<<x<<" y = "<<y<<" i =
"<<i<< endl;
}
```

```
root[] .func
```

ROOT script files (macros) 3



Named script

Τα named script περιέχουν ορισμούς συναρτήσεων εκ των οποίων η μια τουλάχιστον έχει το ίδιο όνομα με το αρχείο.

Δίνοντας π.χ.

```
root[] .x sript3.C
```

εκτελείται η συνάρτηση με όνομα script3( )

script3.c

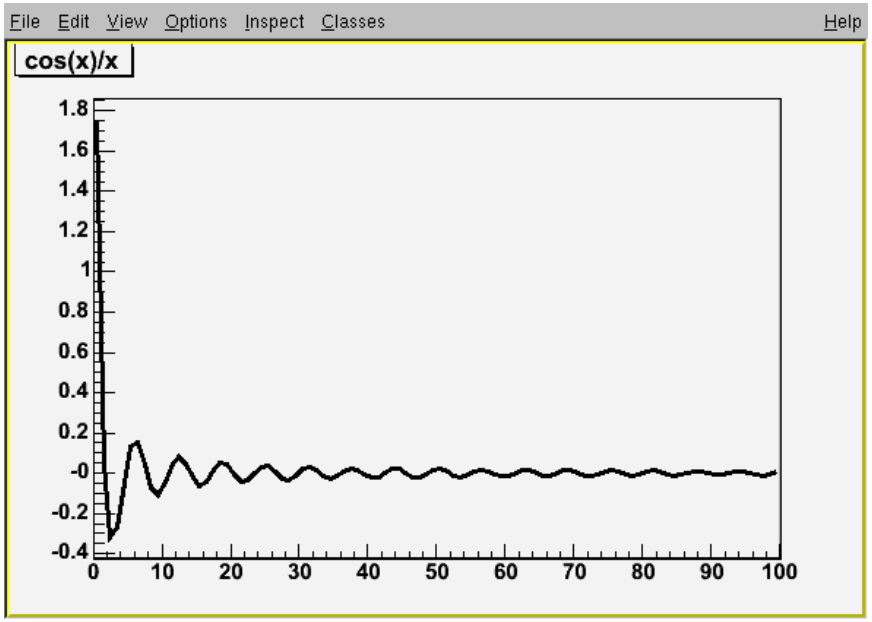
```
int script3(int j = 10)
{
   cout << " Hello" << endl;
   float x = 3.;
   float y = 5.;
   int   i = j;
   cout <<" x = "<<x<<" y = "<<y<<" i =
"<<i<<endl;
   return 0;
}
```

```
root[] .x script3.c(5)

root[] script3(15)
```

GUI (Graphical User Interface)

```
root[] TF1 f1("func1","cos(x)/x",0,100)
root[] f1.Draw()
```

Αντικείμενο Συνάρτηση μιας μεταβλητής
TF1 αντικείμενο: συνάρτηση 1D



Ποια η τιμή της συνάρτησης για x=3;

Ποια η τιμή της παραγώγου για x=3;

Ποιο το ορισμένο ολοκλήρωμα στο [1,20]

```
root [ ] f1.Eval(3)
(Double_t)(-3.2999749866815120e-001)
root [ ] f1.Derivative(3)
(const Double_t)6.2959170441155926e-002
root [ ] f1.Integral(1,20)
(Double_t)(-2.9298410205561481e-001)
root [ ]
```

GUI Αντικείμενα του ROOT

Canvas
Βασικό παράθυρο γραφικών. Μπορεί να δημιουργηθεί και αυτόματα με μια εντολή σχεδίασης γραφικών αντικειμένων (γραμμές, κείμενο, ιστόγραμμα, συνάρτηση).

Pad
Σε ένα canvas μπορούν να ορισθούν τμήματα (**pads**) όπου μπορούν να σχεδιασθούν γραφικά αντικείμενα.

Ένα γραφικό αντικείμενο μπορείτε, με το ποντίκι, να το μεταφέρετε , να του αλλάξετε μέγεθος, ή να το περιστρέψετε.

canvas.C

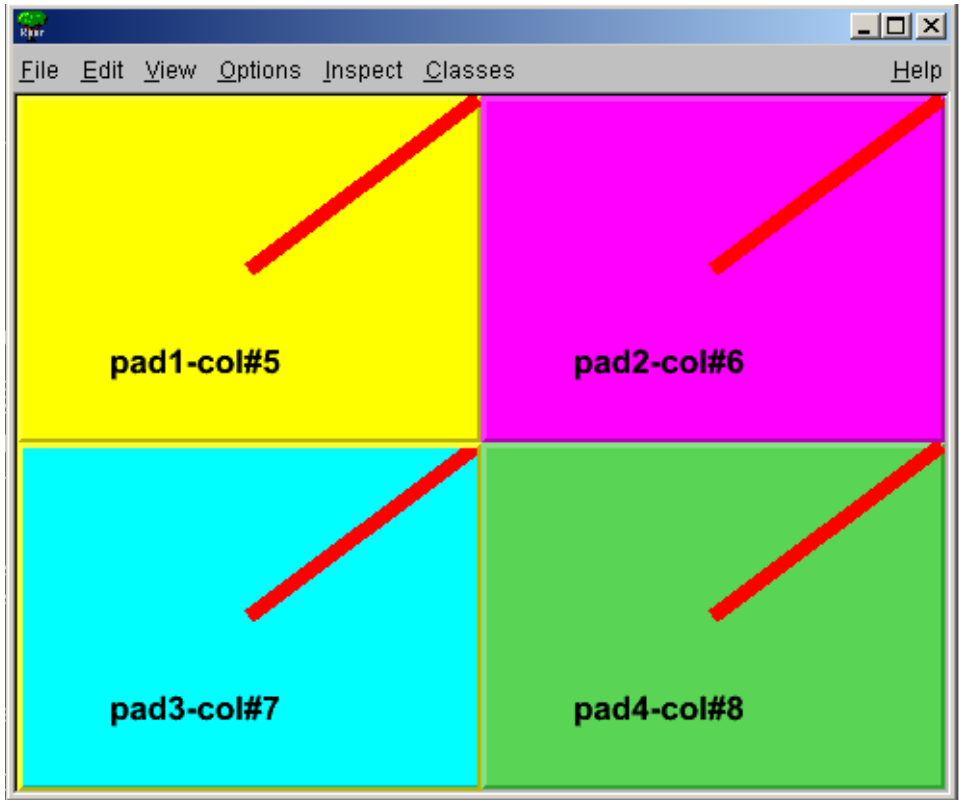
```
{
  gROOT->Reset();

  /*** create a Canvas ***/
  c1 = new TCanvas("c1"," ",200,10,600,480);
  /*** Create a Pad ***/
  TPad pad1("pad1"," ",0.25,0.25,0.75,0.75);
  pad1.SetFillColor(11);
  pad1.Draw();
  pad1.cd();
  /*** Write some text ***/
  TText t1(0.5,0.5,"TEXT");
  t1->SetTextSize(0.1);
  t1.Draw();
  /*** Draw a line ***/
  TLine line1(0.05,0.05,0.95,0.95);
  line1.SetLineWidth(8);
  line1.SetLineColor(2);
  line1.Draw();

}
```

Canvas - Pad

Μπορούμε να έχουμε πολλά pads σε ένα canvas

TPad pad1("pad1"," ",0,0.5,0.5,1);

Δίνοντας `pad1->cd( )`
γίνεται το pad1 τρέχον



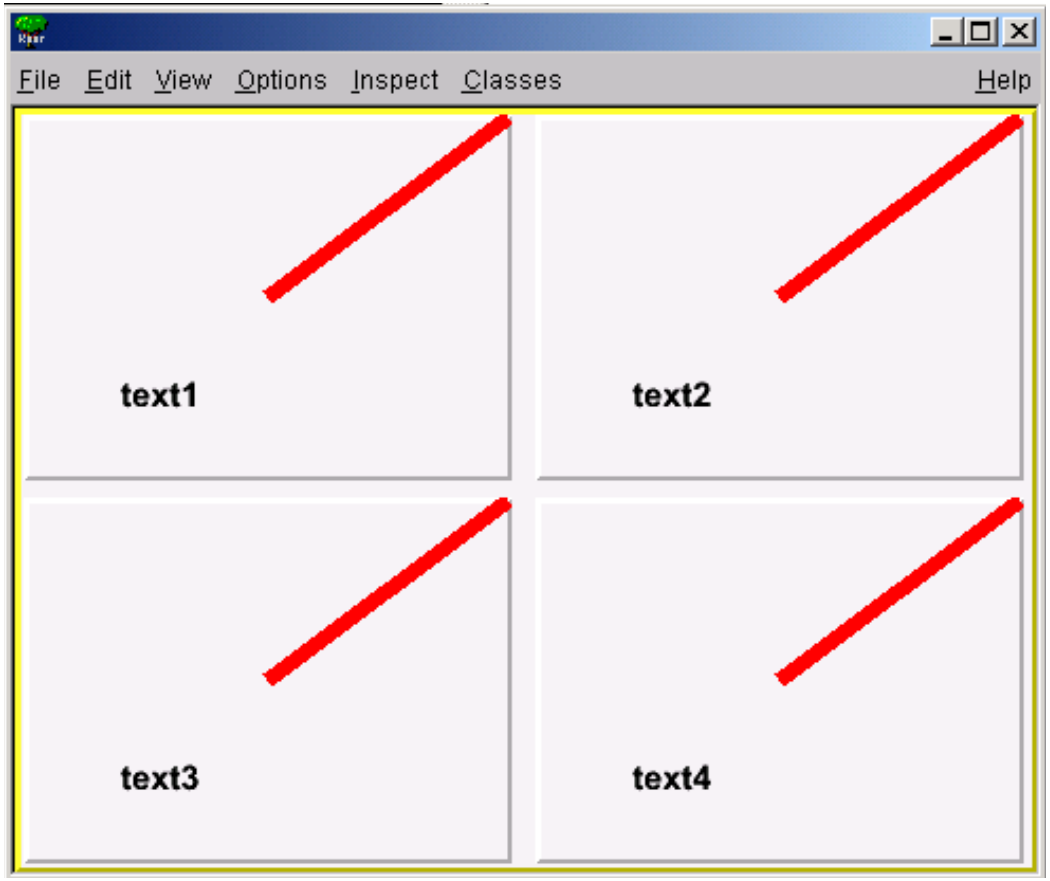
Ξεκ νώντας από το αρχε ο canvas.C προσθέστε τ ς κατάλληλες εντολές ώστε το αποτέλεσμα να ε να όπως παραπλεύρως

4 pads με δ αφορετ κό χρώμα, κε μενο κα μ α γραμμή.

Canvas

Μπορούμε να έχουμε πολλά ***τμήματα*** σε ένα canvas με όνομα ct διαιρώντας το με την εντολή `ct->Divide(2,2)`

Δίνοντας `ct->cd(1)`
γίνεται το τμήμα 1 τρέχον



Ξεκ νώντας από το αρχε ο canvas.C προσθέστε τ ς κατάλληλες εντολές ώστε το αποτέλεσμα να ε να όπως παραπλεύρως

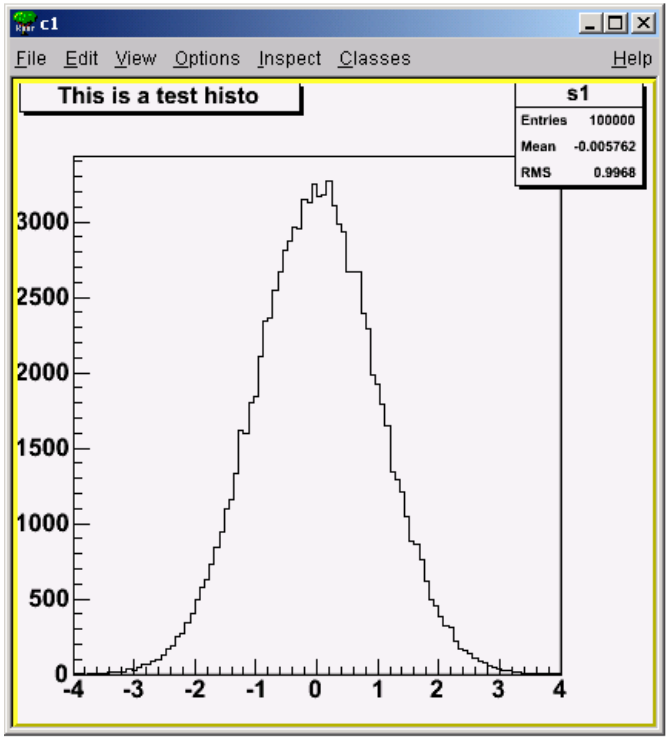
1canvas χωρ σμένο σε 4 μέρη με κε μενο κα μ α γραμμή στο καθένα.

Ιστογράμματα (Histograms)

Ιστόγραμμα είναι μια δομή δεδομένων που περιέχει κανάλια 1,2 ή 3 διαστάσεων.

Διαδικασία:

- Δημιουργία
- Γέμισμα
- Σχεδίαση



histo1.c

```
{
   gROOT->Reset();

  s1      = new TH1F("s1","test histo",100,-4,4);
  float   x;
  for ( Int_t i=0; i<100000; i++) {
     x = gRandom->Gaus(0,1);
     s1->Fill(x);
  }
  s1->Draw();
}
```

Ιστογράμματα (Histograms)

Δημιουργία ιστογράμματος (class constructors).

```
TH1F *hfix = new TH1F("hfix","hf title",nbins,xlow,xup);
```

όπου **nbins** ο αριθμός των bins
xlow το κάτω όριο του πρώτου bin
xup το πάνω όριο του τελευταίου bin

```
TH1F("hfix", "hf title", 3, 0, 3)
```

Ορίζει ένα ιστόγραμμα με 3 bins από το 0 έως το3
(bin 1 = 0-1, bin 2 = 1-2, bin 3 = 2-3).

Ιστόγραμμα μεταβλητού bin:

```
TH1F *hvar = new TH1F("hvar","hvar title",nbins,xbins);
```

Όπου xbins είναι ένας πίνακας από nbins+1 floats που περιέχει τα κάτω όρια των πρώτων nbins bins και το πάνω όριο του τελευταίου bin.

Ομοίως
2-D ιστογράμματα
3-D ιστογράμματα

Υπάρχουν 5 διαφορετικοί τύποι για κάθε ακρίβεια
π.χ. TH2C, TH2S, TH2I, TH2F, TH2D
για char, short, int, float και double)

Ιστογράμματα (Histograms)

ROOT
An Object-Oriented Data Analysis Framework

Γέμισμα ιστογράμματος

- **Fill(x)** αυξάνει κατά 1 το bin που αντιστοιχεί στο x.
- **Fill(x,w)** αυξάνει κατά w το bin που αντιστοιχεί στο x.
- **Fill(x,y)** αυξάνει κατά 1 το κελί (cell) που αντιστοιχεί στα x και y.
- **Fill(x,y,w)** αυξάνει κατά w το κελί (cell) που αντιστοιχεί στα x και y.
- **Fill(x,y,z)** .... αντίστοιχα για 3-D ιστογράμματα.

Ιστογράμματα (“Option”)

Εκτύπωση Ιστογράμματος
με τη συνάρτηση Draw(" option")

Όπου " option" *υποστηρίζονται οι παρακάτω:*

· **"SAME"**	**Ζωγραφίζεται στην προηγούμενη εικόνα**
· **"CYL"**	**Κυλινδρικές συντεταγμένες**
· **"POL"**	**Πολικές** συντεταγμένες
· **"SPH"**	**Σφαιρικές** συντεταγμένες
· **"PSR"**	**Ψευδοραπίντιτι – φ (PseudoRapidity/Phi)** συντεταγμένες
· **"LEGO"**	
· **"LEGO1"**	
· **"LEGO2"**	**Ιστογράμματα τύπου “LEGO”**
· **"SURF"**	
· **"SURF1"**	
· **"SURF2"**	
· **"SURF3"**	
· **"SURF4"**	**3-D Ιστογράμματα τύπου επιφάνειας**

Ιστογράμματα (“Option”)

Για 1-D ιστογράμματα υποστηρίζονται οι επιλογές :

- "A" Μην ζωγραφίζεις αξονες
- "B" Ραβδόγραμμα (Bar chart)
- "C" Smooth curve
- "E" Error bars
- "E0" Error bars συμπεριλαμβάνοντας τα bins με περιεχόμενο 0
- "E1" Error bars με κάθετες γραμμές
- "E2" Error bars με παραλληλόγραμμα
- "E3" Γεμάτη περιοχή (Fill area)
- "E4" Όπως E3 αλλά με μια καμπύλη (smoothed filled area)
- "L" Γραμμή δια των σημείων
- "P" Ζωγραφίζει το τρέχων σημάδι (marker) σε κάθε bin
- "*" * σε κάθε bin

Ιστογράμματα (“Option”)

Για 2-D ιστογράμματα υποστηρίζονται οι επιλογές :

· "ARR"	Βέλος που δείχνει την κλίση ανάμεσα σε γειτονικά κελιά
· "BOX"	Box για κάθε κελί επιφάνειας ανάλογης του περιεχομένου
· "COL"	Box με χρωματική κλίμακα
· "CONT"	Ισοϋψείς καμπύλες (contour plot)
· "CONT0"	
· "CONT1"	
· "CONT2"	
· "CONT3"	
· "FB"	Με LEGO ή SURFACE, αποκρύπτοντας το Front-Box
· "BB"	Με LEGO ή SURFACE, αποκρύπτοντας το Back-Box
· "SCAT"	Scatter-plot (default)

Ενα Ιστόγραμμα αφού εμφανιστεί μια φορά μπορούν να γίνουν αλλαγές χρησιμοποιώντας το ποντίκι και το DrawPanel.

Ιστογράμματα (Histograms)

Πράξεις με ιστογράμματα

- THdp& operator=(THdp&)
- THdp operator +(THdp&, THdp&)
- THdp operator -(THdp&, THdp&)
- THdp operator *(THdp&, THdp&)
- THdp operator /(THdp&, THdp&)
- THdp operator *(THdp&, Float_t)

Όπου dp είναι η διάσταση και η ακρίβεια.

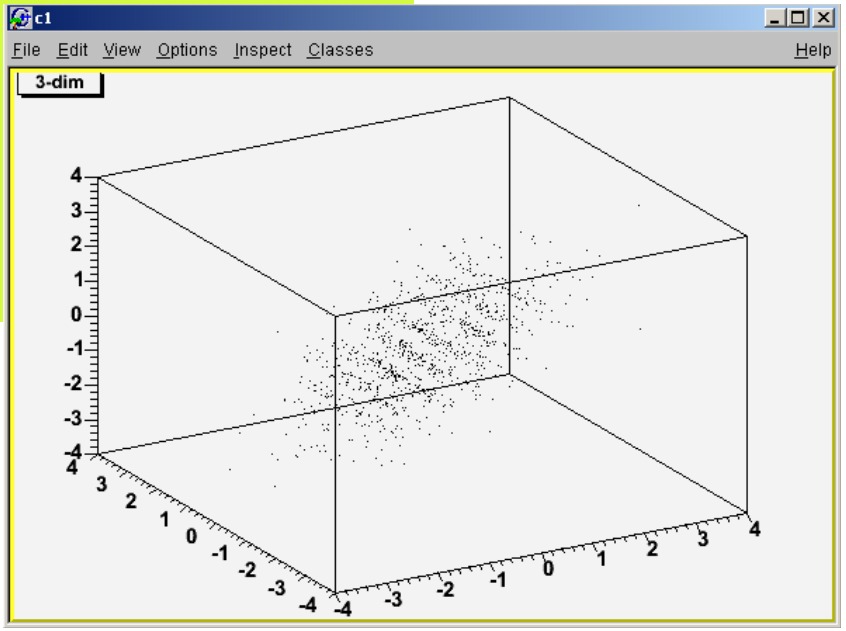
```
TH1F   h = h1*2.5 + h2*8;
TH1F hf;
hf = h/h1;
```

Ιστογράμματα (Histograms)

ROOT
An Object-Oriented Data Analysis Framework

histo3.c

```
{
  hx      = new TH1F("hx","1-dim",100,-4,4);
  hxy     = new TH2F("hxy","2-dim",40,-4,4,40,-4,4);
  hxyz    = new TH3F("hxyz","3-dim",20,-4,4,20,-4,4,20,-4,4);
  gRandom->SetSeed();
  float x, y, z;
  for (int i=0; i<1000; i++) {
     gRandom->Rannor(x,y);
     z = x;
     hx->Fill(x);
     hxy->Fill(x,y);
     hxyz->Fill(x,y,z);
  }
//    hx->Draw();
//  hxy->Draw();
   hxyz->Draw();
}
```

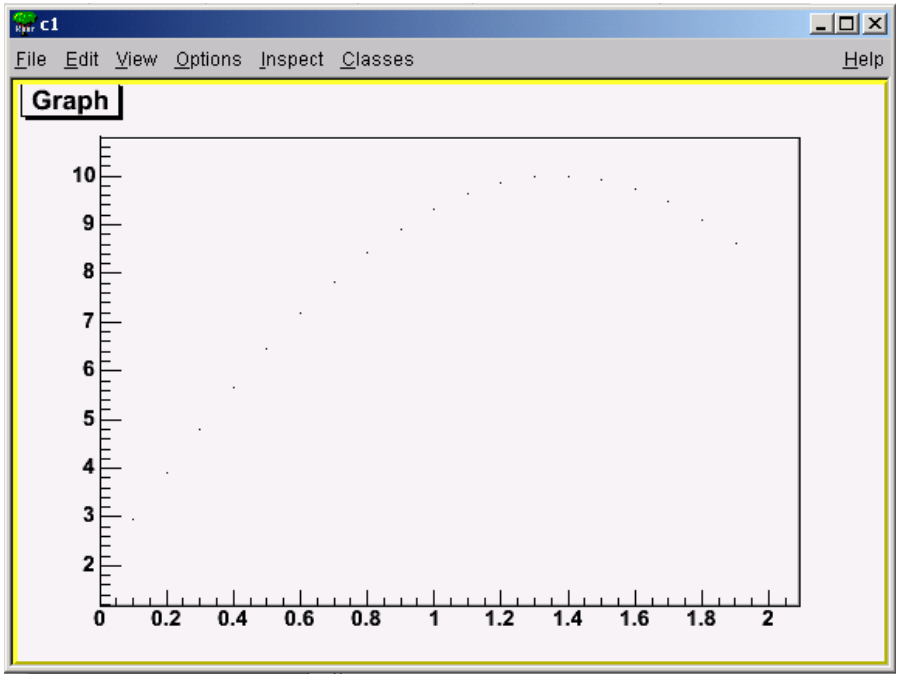


Γραφικές παραστάσεις (Graphs)

Γραφική παράσταση x-y χωρίς σφάλματα

root [0] .x graph1.C

Μπορείτε να αλλάξετε τη θέση σημείου με το ποντίκι



graph1.C

```
{
 gROOT->Reset();

   const Int_t n = 20;
   Double_t x[n], y[n];

   for (Int_t i=0;i<n;i++) {
     x[i] = i*0.1;
     y[i] = 10*sin(x[i]+0.2);
     printf(" i %i %f %f \n",i,x[i],y[i]);
   }

   gr = new TGraph(n,x,y);
   gr->Draw("AP");
}
```

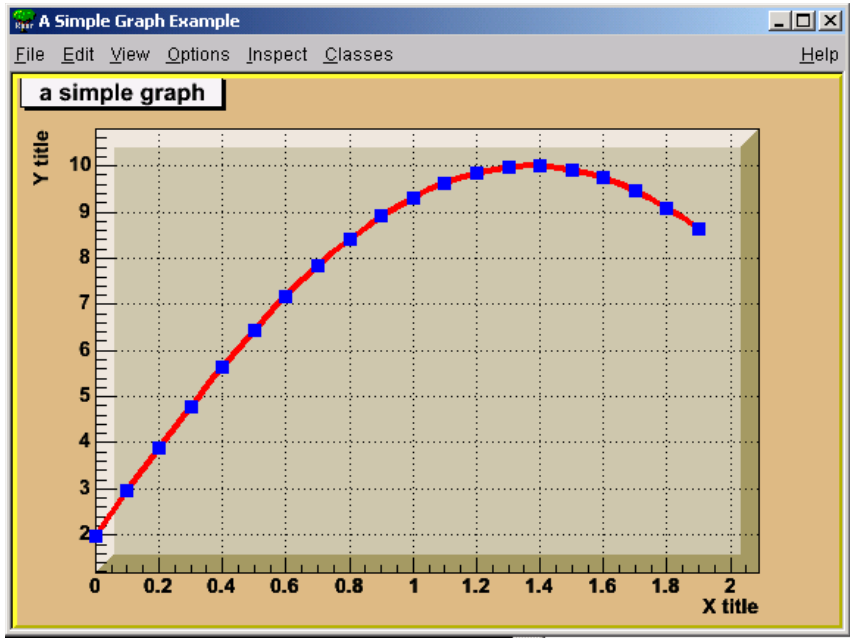
1	black
2	red
3	green
4	blue
5	yellow
6	pink
7	light blue
8	white

Γραφικές παραστάσεις (Graphs)

root [0] .x graphbeauty.C

Το προηγούμενο παράδειγμα ....λίγο πιο όμορφο...

Πολλά μπορούν να γίνουν με το ποντίκι



graphbeauty.C

```
{
   gROOT->Reset();
   c1 = new TCanvas("c1","A Simple Graph
Example",200,10,700,500);
   c1->SetFillColor(42);
   c1->SetGrid();
   const Int_t n = 20;
   Double_t x[n], y[n];

   for (Int_t i=0;i<n;i++) {
     x[i] = i*0.1;
     y[i] = 10*sin(x[i]+0.2);
     printf(" i %i %f %f \n",i,x[i],y[i]);
   }
   gr = new TGraph(n,x,y);
   gr->SetLineColor(2);
   gr->SetLineWidth(4);
   gr->SetMarkerColor(4);
   gr->SetMarkerStyle(21);
   gr->SetTitle("a simple graph");
   gr->GetXaxis()->SetTitle("X title");
   gr->GetYaxis()->SetTitle("Y title");
   gr->Draw("ACP");
   // TCanvas::Update() draws the frame,
   //after which one can change it
   c1->Update();
   c1->GetFrame()->SetFillColor(21);
   c1->GetFrame()->SetBorderSize(12);
   c1->Modified();
}
```