



Published on *ROOT* (<http://root.cern.ch/drupal>)

[Home](#) > [Printer-friendly PDF](#) > [Printer-friendly PDF](#)

---

# ROOT Graphics "To-Do" list

[2D](#) <sup>[1]</sup>   [3D](#) <sup>[2]</sup>   [OpenGL](#) <sup>[3]</sup>   [ROOT graphics](#) <sup>[4]</sup>

This page lists the various developments we think are needed for ROOT graphics. They are classified in the following categories:

- **New Functionality, New Classes.**
- **New Functionality in Existing Classes.**
- **Redesign.**
- **Bug Fixes.**
- **3D Graphics OpenGL.**

For each of these categories and list of items is given. We try to list them in the order of priority.

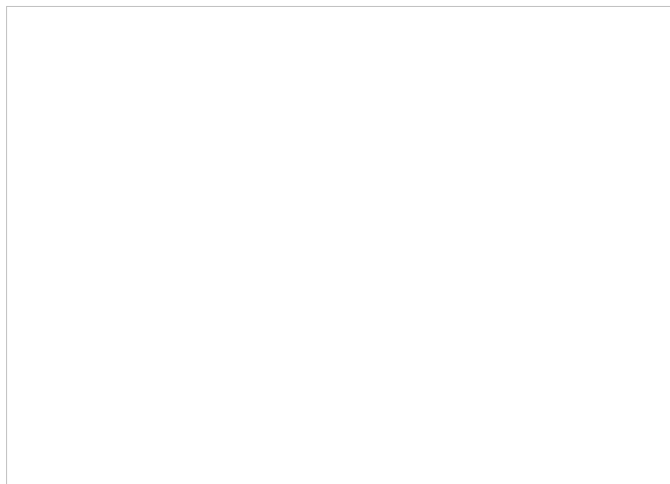
---

## New Functionality, New Classes

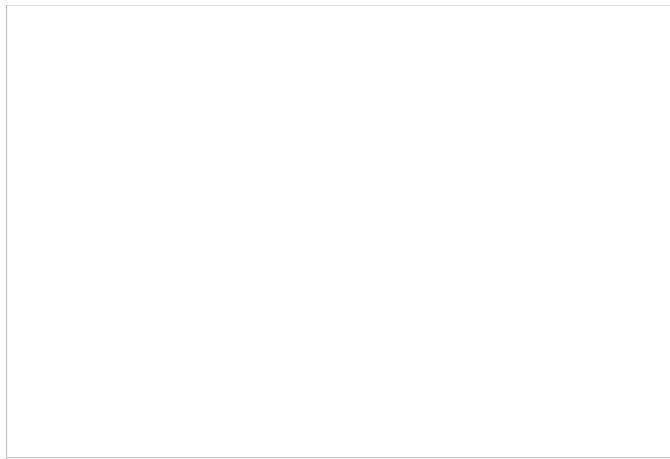
### graphviz:

- A graphical tool allowing to display and interact with graphs has been implemented. The graphs interpretation is done thanks to the [graphviz](#) <sup>[5]</sup> package. Graphs representations are needed in several places in ROOT. A generic tool using the power of the ROOT graphics and GUI would be really useful. *graphviz* was already used in ROOT (via a shell-script interface) to visualize dependencies in the THtml generated documentation. *graphviz* is the standard graph-dependency visualization tool used by open source projects like *kcachegrind*. The new interface class makes use of the existing C-API in this package. This class will be by several ROOT monitoring tools currently under development and also by RooFit and Roostats. The `TGraphStuct`, `TGraphNode` and `TGraphEdge` classes have been implemented. They take advantage of the high quality ROOT graphics to draw graphs and interact with them.

The future developments will be done according to the applications needs. The first application using it will be very likely THTML. The extensions of the current implementation will be done according to what is needed by THTML.



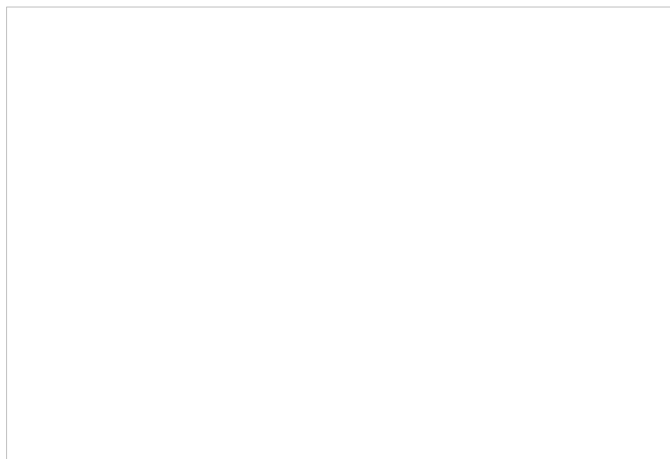
### Honey Comb Histograms:



Now implemented via the TH2Poly class.

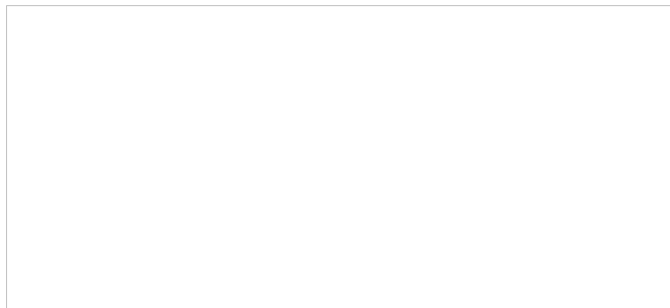
### Time evolving Graphs:

- Tools like the `gapminder` [6] one. The class `TGraphTime` is now implemented. The future work will be around the look and presentation. "gapminder" can be use as example. For instance it will be good to be able to draw the markers with some transparency (not trivial).



### Dendrograms:

- Dendrograms allow to create clusters by calculating the distance (or "similarity") between different entries. It create a first cluster with the 2 entries which are the closest, and so on. See <http://root.cern.ch/phpBB2/viewtopic.php?t=8848> [7]



### New class TLogo:

- Useful to draw a logo. Should inherit from TBox. The constructor would be `TLogo(x1, y1, x2, y2, filename)`. The parameter "filename" is the name of the picture which should be use as logo. It will be displayed in the `TBox` using the `TImage` class.

---

## New Functionality in Existing Classes.

## Vector graphics output (PostScript, PDF, SVG)

- Some not urgent items:
  - SVG: Not complete yet. It seems SVG ROOT files are more and more used, so we may get requests to complete it.
  - PDF: The rendering of bitmap is missing (cell array).

### TGaxis improvements:

- $\pi$ -axis.
- Specify where the " $x10^N$ " appears. Right now, it appears at the end of the axis.
- Discontinuous axis:



- Possibility to align the vertical axis' labels on the left. To avoid the effect shown on the following picture:
- The axis label are always painted with angle equal to 0 (except the alphanumeric labels). This generates two problems:
  1. The options like "V", "U", "D" do not work (labels orientation).
  2. When rotated the axis labels look ugly compare to PAW.
- Possibility to define the label format like in C++. For instance with a function like: `SetLabelsFormat("%5.3g");` If that setting is " " we let it work like now. If it is not, that string will be use to define the axis labels format.
- Log scale in base 2. Only base 10 is available now.
- Possibility to draw labels only on a range of an axis. The "labeled range", by default, would be the full axis but could be define smaller than the full range. May be something like `TGaxis::SetLabeledRange(v1, v2)`
- We had a request to have `fgMaxDigits` for individual axis. Now it is a global value. Does it make sense ?
- Possibility to better place the axis title now it is either right aligned or centered. Left aligned is missing.

### Not Urgent:

- A request for log scale along Y axis: need to plot a 2D plot with values on a logarithmic ordinate. The values run from negative to positive values. I would like to plot the ordinate like:

```
+10e3 |
+10e2 |
+10e1 |
+10e0 |-----
-10e1 |
-10e2 |
-10e3 |
-10e4 |
```

I believe this is not yet possible in ROOT, and it would be useful to have. Another axis view is:

```
+10e0 |
+10e1 |
+10e2 |
+10e3 |
+10e4 |-----
```

which would be nice to have too.

- if one chooses time on the horizontal axis, let's say %m/%y, the current implementation makes equidistant bins that result in an x axis like :  
01/01----12/02----01/05 instead of the better: 01/01----01/03---01/05. So, in general, the algorithm of chooses the bins should in case of a time axis not just consider making bins equidistant.

## 3D plots (Lego and surface):

- Possibility do draw the axis labels perpendicular to the axis.

## Astronomers requests:

- **Reverse X and Y axis** [8]. That's not only the axis which should be changed but also the coordinate system inside the pad which should go right to left and up to down. That may implies a deep restructuring of the code.
- **The projections: AITOFF, MERCATOR** etc... are available for some representations but are still missing for COL plots for instance.

## TGraphPolar:

- TGraphPolar logarithmic radial axis [9]

## TLatex new symbols -> See TMathText:

For these characters the mapping between TTF and PostScript is not direct. Sometimes the character is available in PS but not in TTF or vice versa.

- **#ell** Calligraphic "l". This glyph doesn't exist in PostScript. We can emulate with line or polygon drawing.
- **#mathcal{}** Calligraphic font
- **#v{}** (similar to #check but not quite so)
- **#perthousand{}** ‰
- **German umlaut.**
- Possibility to underline a character: **#underline{}**

The new class TMathText (ROOT 6) should address most of these requests.

## TAsimage:

- Pictures containing transparent parts (gif, png) are not transparent when they are displayed in a ROOT pad (TImage::Open).

## Markers:

- **We need more markers type**. The current list is not enough and some are misleading. User defined markers. The list has been completed though.

□ □

## Contour plots:

- **h->Draw("CONT LIST")** draws the contour. There is no way to get the TGraph list without drawing the contours. The option "NOCONT LIST" has been suggested to get **the list of contours without drawing them**.
- CONT4 draws the filled contours correctly but it is not possible to draw a an other graphical object on top of it using the histograms coordinates. In particular **option SAME does not work on top of a contour drawn with CONT4**.
- **Draw the Labels on contours**. Use the macro ContourList.C in \$ROOTSYS/tutorials as starting point.

## Lego and Surface plots:

- **Possibility to draw lego, and surface plots with "holes"** when the value of some bins are unknown or undefined. Can be extended to other representation. May be a convention for the bin content is needed. This can be done using TH2Poly.
- **Make sure that he option SAME works between LEGO and SURF not having the same ranges**. Example:

```
{
  TH2F *h1 = new TH2F("h1", "", 40, 0, 10, 40, 0, 10);
  TH2F *h2 = new TH2F("h2", "", 20, 0, 2, 20, 0, 2);
  for(int i = 0; i < 1000; i++) {
    h1->Fill(gRandom->Gaus(6,1), gRandom->Gaus(6,1));
    h2->Fill(gRandom->Gaus(1,.3), gRandom->Gaus(1,.3));
  }
  TCanvas *can = new TCanvas("can");
  can->Divide(2,1);
  can->cd(1);
  h1->Draw();
  h2->Draw("same");
  can->cd(2);
}
```

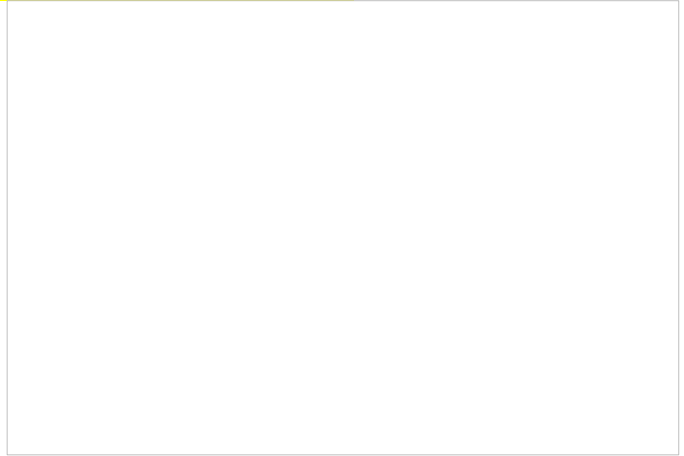
```

    h1->Draw("surf1");
    h2->Draw("surf1 same");
}

```

## TPaletteAxis:

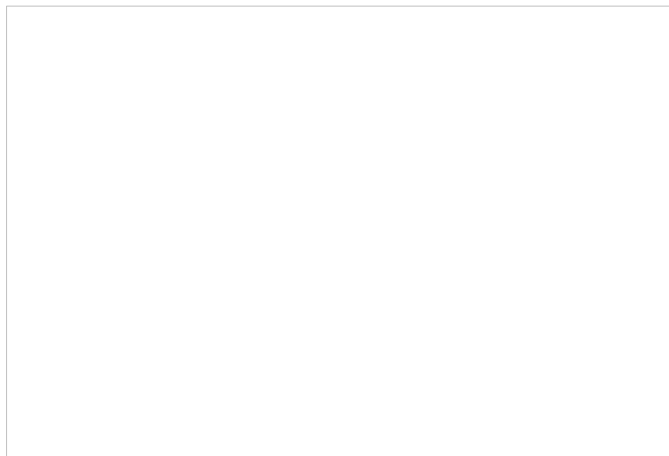
- Get the color of a bin in a 2D histogram drawn with "COLZ". A getter function in TPaletteAxis should be enough.



- Implement the possibility to draw the palette horizontally.
- Call TPaletteAxis in PaintH3: `ntuple.Draw("px:py:random:pz", "", "colz")`

## Histogram plotting:

- With options "c" or "l", drawing is done through the bins centers. One could implement an algorithm extrapolating to the edges.



- Implement a clean drawing for underflows and overflows in TH3 statistics (THistPainter::PaintStat3).
- Implement a way to draw many histograms in one go (all the histograms in a file for instance) by dividing automatically the pad with the needed number of zones.
- Error bars should be visible even if its data point is outside the axis range - at least optionally.

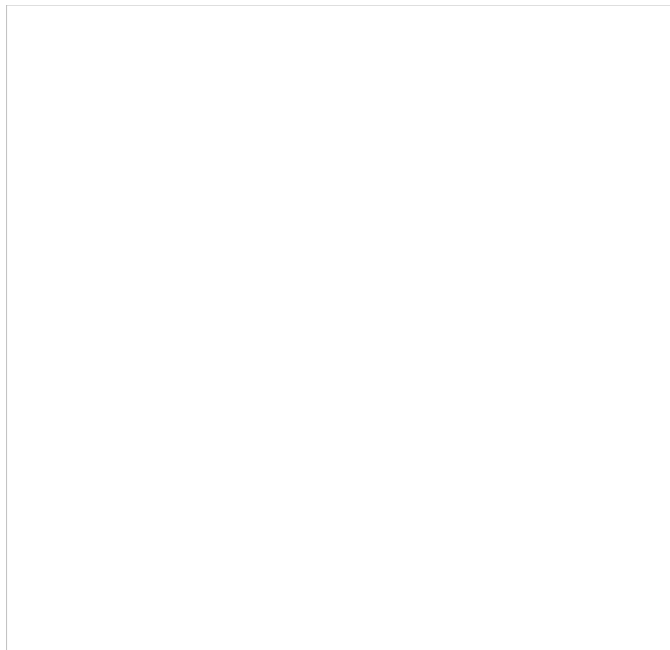
## Plotting Histograms ' overflows:

- The following macro is a working example but it could be a general option. In particular it should work for THStack too.

```

void overflow() {
    c1 = new TCanvas("c1", "Overflow", 200, 10, 700, 700);
    hpx = new TH1F(...);
    [...] PaintOverflow(hpx);
}

```



```
void PaintOverflow(TH1 *h) {
    // This function paint the histogram h with an extra bin for overflows
    char* name = h->GetName();
    char* title = h->GetTitle();
    Int_t nx = h->GetNbinsX()+1;
    Double_t x1 = h->GetBinLowEdge(1);
    Double_t bw = h->GetBinWidth(nx);
    Double_t x2 = h->GetBinLowEdge(nx)+bw;
    TH1F *htmp = new TH1F(name, title, nx, x1, x2);
    for (Int_t i=1; i<=nx; i++) {
        htmp->Fill(htmp->GetBinCenter(i), h->GetBinContent(i));
    }
    htmp->Fill(x1-1, h->GetBinContent(0));
    htmp->SetEntries(h->GetEntries());
    htmp->Draw();
    TText *t = new TText(x2-bw/2, h->GetBinContent(nx), "Overflow");
    t->SetTextAngle(90);
    t->SetTextAlign(12);
    t->SetTextSize(0.03);
    t->Draw();
}
```

## Variable bin width histograms:

- If the bin width is 5 and there is 1 entry, the bin height should be 0.2, not 1. One can do:

```
for (int i=1; i<nbins; i++) {
    SetBinContent(histo->GetBinContent(i)/histo->GetBinWidth(i));
}
```

but a draw option would make it easier.

## TMarker3DBox:

- **TMarker3DBox in TPad (the none GL version) must be drawn with colors.** Now it is wire frame only. Like the COL option but in 3D. Also OpenGL has a role to play here.

## Paint 3D triangles:

- Implement `Paint3DTriangles` method in TPad.

## TSpline2D:

- Like we have TGraph2D and TGraph, I would be nice to have **TSpline2D** (TSpline already exist).

## Temporary pads' resizing:

- Sometimes it would be useful to scale up a single pad in a canvas temporarily. Especially if one needs a lot of pads in parallel to get an overview over many different signals this would be very nice to have a closer look at some

graphs. An other way to do that would be enlarge the Pad when moving the mouse over.

## Candle plots:

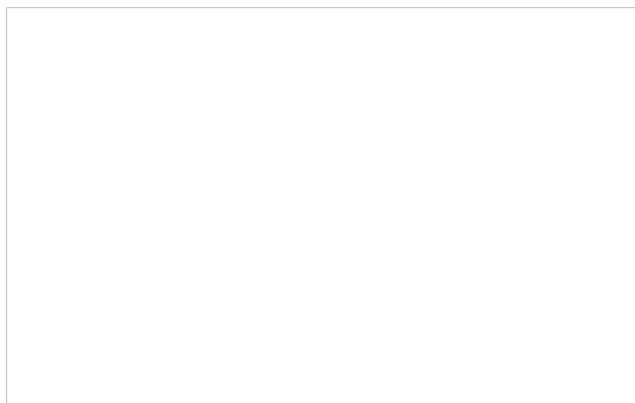
- Possibility to plot only **one data set (one variable) as a candle plot**. Now the minimum is 2 (see [here](#) <sup>[10]</sup>).

## TParallelCoord:

- **Give the possibility to change attributes of the various text displayed by the TParallelCoord**. Basically the axis labels and the axis titles. As there is two type of text it might be not enough to just inherit from TAttText. Special setters might be required.



- Use the transparency provided by OpenGL to display the ||-Coord. Some prototype have been done. It is very promising:

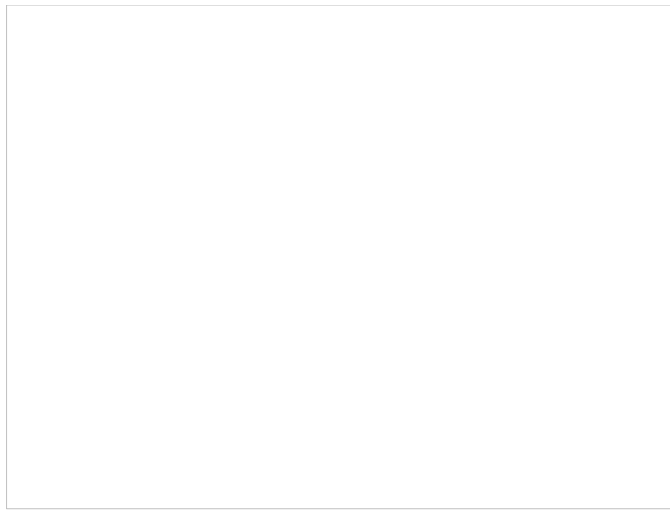


## TTF fonts:

- **Access TTF fonts by name.** `TAttText::SetTextFont(const char *fontname)`, `TTF::SetTextFont(const char *fontname)` already exist. Having this functionality in `TAttText` would allow to access extra fonts in `$ROOTSYS/fonts` like: `BlackChancery.ttf`, `verdana.ttf` etc ... or any ttf file users may want to put in `$ROOTSYS/fonts`.
- **Extend the list of available fonts (support for any TTF fonts like in TASImage).** The new fonts should be available for TTF, X11 fonts, SVG and PostScript/PDF fonts.

## Transparency:

- **The fill area transparency is available for pads only and on screen only** (see [here](#) <sup>[11]</sup>), it would be good to also have it for any fill area to allow the following kind of plots:



---

## Redesign

### Normalized Device Coordinates (NDC):

- What is called "NDC space" in ROOT is not a true NDC space. **NDC means *Normalized Device coordinates***. A such space provides a uniform access to the *device coordinates* which are device dependent (pixel on screen, dots or inches on PostScript, etc ...). What is called **NDC space in ROOT is a kind of normalized space which varies when the canvas ratio changes**. For some drawing, this creates weird side effects. For example in TArrow, some intermediate transformations are needed because of that, otherwise the arrow head is distorted. The NDC space is a nice concept well known in 2D graphics (GKS) but has been wrongly implemented in ROOT.

### Pad division:

- **The current implementation of the pad division causes several troubles** and questions, specially when people want to create pads next to each other with no margins and common axis. The way `TPad::Divide` is implemented makes it hard to get nice plots. The two main criticisms are:
  1. The various sub pads do not have the same surfaces (some macro doing it better have been implemented).
  2. Some axis labels are erased or overlapp.

### TStyle:

- **Replace the three TAttAxis members of TStyle by TAxis objects**, such that all TAxis options could be supported without adding new functions in TStyle. For example the title centering. It would be nice to center the axis titles with a global TStyle function.

### Plots layout:

- **Defining a standard page layout is often not easy**. The user should often change parameters like the margins to make sure his plots look good. For instance if one of his plot has long axis labels along the Y axis, he will have to increase the left margin to make sure they are visible and if his next plot has short label he will have to decrease the left margin again otherwise his plot will really look ugly. In a such case an other approach would be to define the left margin automatically to make sure the axis labels always fit.

### THistPainter, TGraphPainter internals:

- Replace **gCurrentHist, Hoption and Hparam by class members**
- **Revisit the way TGraph is painted**. It is really needed to use an intermediate histogram ? May be that's too late to change that now.

### Histogram plotting:

- **Option POL combined with option COL** should be revisited, completed and documented when ready. Right now the status is still: <http://root.cern.ch/viewvc?view=rev&revision=10690> <sup>[12]</sup>

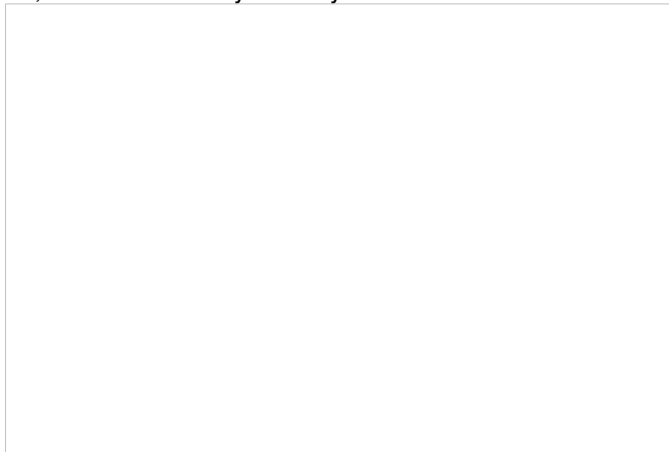
---

## Bug Fixes



## Histogram plotting:

- Scatter plot with negative content draw far too much points. The painting is very slow.
- The fit function drawing is sometimes misleading, like in the following example. The function's behavior around  $x=0$  and at  $x>3.8$  is incorrect. In both cases the function should continue as a straight line and be clipped where the function value leaves the frame, and not show any line beyond that  $x$  value.



---

## 3D Graphics OpenGL

### Opengl:

- Finalize 2D pad drawn with GL. PS (batch rendering) rendering (using `TPainter3DAlgorithms`) of the new surfaces without open GL.
- Finalize/integration 5D plots and density estimator (package in Fortran).
- Implement TGraph2D and stack of lego.
- In PostScript the axes are not very accurate (on legos and surfaces). Real 3D axis will fix that.
- Interface the Pad-GL developments with the 3-D GL viewer and the EVE packages in view to insert 1-d, 2-d, 3-d histograms or any 2-D graphic in a GL scene.
- If we have 2D histograms where the bins have a shape defined by a graph, the bins can have any shapes. Such bins should be easy to draw with the scatter option and with the color option but it would be also nice to draw in 3D (like legos). The base shape of a bin will be defined by a curved and the elevation proportional to the bin content. Such plots are often shown for instance with the map of a country. Each bin is a part of the country and a shape which correspond to boundaries, like for instance the states in the USA. **This is done in TH2POLY**
- 
- Try some "i-Tunes like features" to manage/display multiple canvases.
- Experiment with shaders and anti-aliasing.
- Implement the option SAME for the "GLBOX" option in order to display several data set on the same plot. See the suggestion [here](#) [13].

---

### Some requests difficult to implement because of historical reasons:

- When drawing a graph, you have to indicate "A" to draw the axes, and no option to not. This is the opposite of histogram drawing, where you give no option to draw a fresh axis and "SAME" to draw on top of an old histogram.

© 1995-2013 The ROOT Team

---

Source URL: <http://root.cern.ch/drupal/content/root-graphics-do-list>

#### Links:

- [1] <http://root.cern.ch/drupal/category/package-context/2d>
- [2] <http://root.cern.ch/drupal/category/package-context/3d>
- [3] <http://root.cern.ch/drupal/category/package-context/opengl>
- [4] <http://root.cern.ch/drupal/category/package-context/root-graphics>
- [5] <http://www.graphviz.org/>
- [6] <http://www.gapminder.org/>
- [7] <http://root.cern.ch/phpBB2/viewtopic.php?t=8848>
- [8] <https://savannah.cern.ch/bugs/?38747>
- [9] <https://savannah.cern.ch/bugs/?47987>
- [10] <http://root.cern.ch/phpBB2/viewtopic.php?p=35732#35732>
- [11] <http://root.cern.ch/root/html/TAttFill.html#F2>

- [12] <http://root.cern.ch/viewvc?view=rev&revision=10690>  
[13] <http://root.cern.ch/phpBB2/viewtopic.php?t=8534>