Octave-GUI: a collection of Ideas

Muthiah Annamalai, gnumuthu@users.sf.net

25th september 2005.

[1]

Contents

[1]Note: This is a draft copy of the document, not intended for public circulation, treat like pre-print.

1 Introduction

Octave needs a GUI. This is an absolute necessity, and we as a community, can pool-in the resources, and make a rock-solid cross platform GUI. Octave GUI is needed, because there is a framework that exists today, which we may use to provide, a nice GUI. Were in the year 2005, with stable GUI toolkits, like *GTK,Qt,FLTK,wxWindows* are the production stable, cross platform toolkits, which are candidates, that come up, when we look to build a GUI for Octave.

1.1 Choosing Toolkit

Now we must choose a toolkit for use with a community developed Octave GUI. In this section I will try to show, the advantages of using a cross platform GUI, and evaluate the strengths and weaknesses of each of the toolkits.

A fair state of the various GUI programs can be found here

`http://www.geocities.com/SiliconValley/Vista/7184/guitool.html`

1.1.1 GTK+

GTK+ is a strong toolkit, but has problems from the point of view of a scientific toolkit, that needs many widgets, and as much times, lesser dependencies. GTK+ lacks many widgets, like Rich Text editor, HTML view widget, and list editors/ view editors. Id like to think of GTK+ as a low level API for construction of GUI with lots of control on things, and how they happen. In my opinion, GTK+ must be usable just as it is shipped with, and no need for extra dependencies, like GTK Extra, GtkHTML etc. Maintaining, a fork, or Shipping a separate version with our application will be self defeating. Lack of manpower, version history, dependencies of these packages, etc make this self-maintaining thing really hard. You have to do this to appreciate it. I think a GNOME frontend to Octave will be well suited to be written in GTK+, with MONO bindings, or whatever language you want to write. Good place to start would be YAOG.

Major problem with GTK+ is that it doesnt work off the shelf, for Mac OS X. Again for the same reasons, of man-power, we cannot maintain a separate port for Mac OS X.

`http://www.gtk.org/bindings.html`

1.1.2 Qt

Qt is a rather attractive choice for what I have in mind, for the simple reason, that Qt comes with a lot of nice/rich set of widgets –within– the library itself, and not as external dependencies like in the GTK+ tree. Now Qt has a problem of making you use C++, if that is a problem at all. Now you have to factor in the binary size, loading time, memory consumption etc, to make sure Qt based GUI runs on top. But Qt based GUI is really a nice solution, as long as it works on the MAC OS-X, See [

`http://www.trolltech.com/products/qt/index.html`

] for more on the cross platform capability of Qt. Only problem with Qt, is the C++ hassle, and its bloated size. KDE folks already have a front end called Koctave.

Qt too has its share of Mac OS X problems, as I understand, and licensing issues with Windows platform. So better start from a safer platform.

1.1.3 WxWidgets

I dont know much about them, but they are extremely cross platform, and friendly licensed I understand from their, webpage. This is how, WxWidgets folks talk their marketing pitch,

`http://wxwidgets.org/whychoos.htm`

but again its sacrificing speed & memory to WxWidgets people who, even if they do a good job of emulating platform specific GUI API's over a single method, donot still, provide a complete solution, anyway w.r.t speed etc.

I dont know, is the bottom line, and I dont have enough experience with this toolkit, and cannot judge it really.

C++, memory, and things are against it. But the rendering,and stuff make it really attractive. see

`http://www.flamerobin.org/ screenshots for`

more impressive results with WxWidgets across Mac OS X,GNU/Linux & Windows.

WxWidgets has some 5 GUI designer tools! Damn this, how come I never knew that?

You need to get convinced: *http://wxwidgets.org/fosdem2003/html/talk/*

Note To Vijay **WxWidgets looks really attractive, but Ive put all my money on FLTK already! Atleast 150 hours of work.**

1.1.4 FLTK

This is the darling of all toolkits, AFAIK. Its ugly, or like you say without the bells and whistle of GTK+ or Qt, or WxWidgets, but just raw power with C++/C optional styles, and perfect GCD of features I need and the correct mix of components for development.

Id like to have a HTML widget for help browsing, simple syntax, FLUID UI designer integration, and things like access to FLTK API, from Octave, and all this stuff, with simple clean API, easily learnable from Octave users point of view.

1.2 Conclusion / choice

I think FLTK is already put in place, and will make things easy in making a Octave-GUI, as FLTK itself is small, slick and easy to use. But wxWidgets are a dream for Octave, and Id give lots of time to make that happen. But first getting Octave-FLTK-1.0 rolled out, and integrating FLUID [FLTK UI Designer] into the Octave-GUI will make this Octave really attractive for people to work on.

I think Id give 150+ hours to port wxWidgets to Octave, and in the meanwhile Id have a comparative advantage of moving Octave-FLTK-1.0 to completion & stable release, and do the above mentioned tasks, to fullness, and release Octave-GUI. This will probably bring credibility to warrant porting of wxWidgets to Octave, and anyone can then write a GUI for themselves, to compete with my Octave-GUI.

I got for FLTK, even though Im crushed about WxWidgets, as it seems to follow *Alan Kay's* rule:

simple things must be simple, difficult things possible

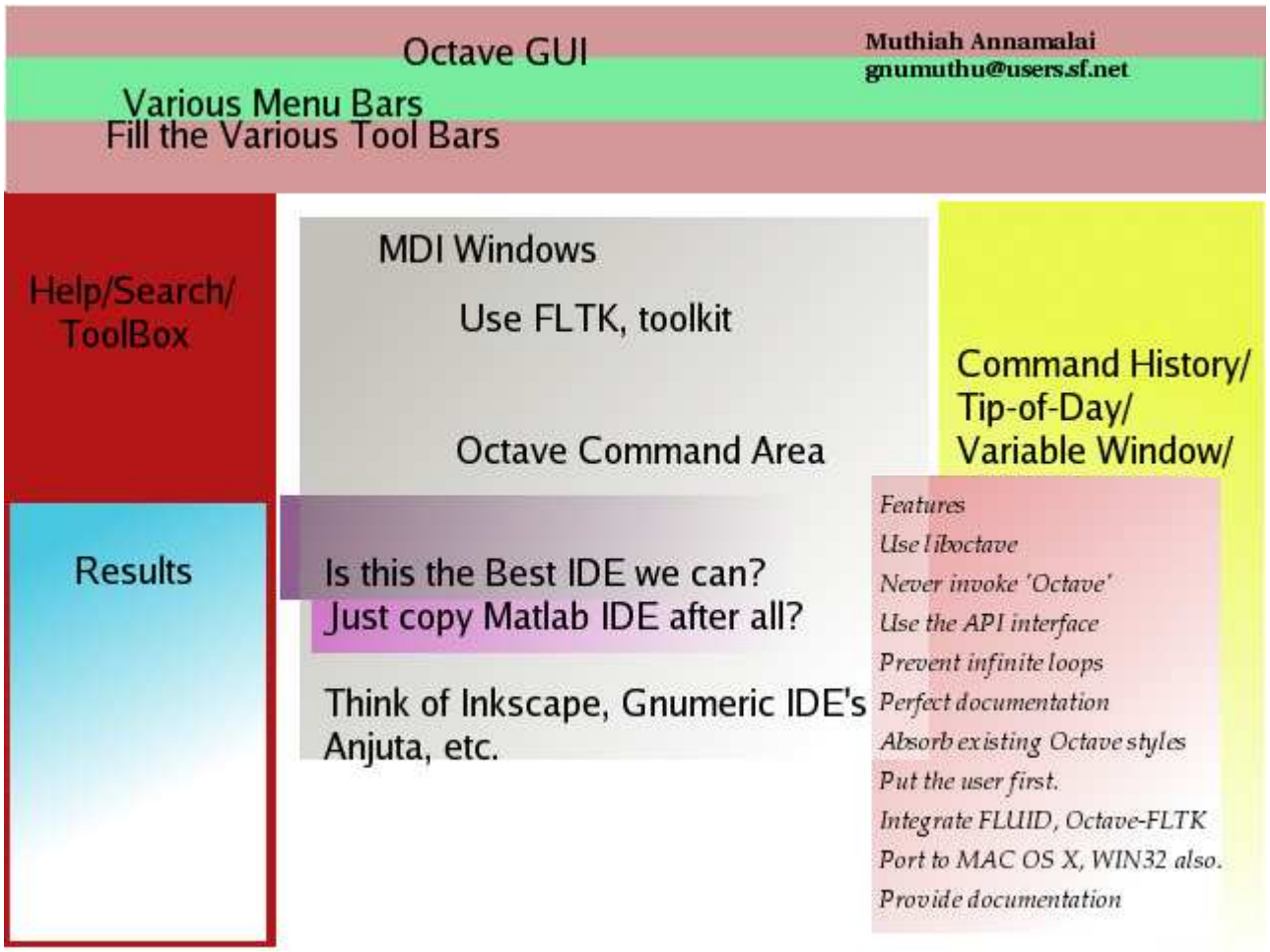

Figure 1: Octave-GUI Mockup

[2]

2 GUI design

A simple Mockup.

I just read through all the mails, weve exchanged in the Octave mailing lists, over the last year [May-June 2004-05], regarding a GUI for the Octave project, and I just come to once conclusion: we must go for an FLTK based GUI frontend, or else I cannot find/think of a simpler solution. FLTK is really light weight, and easy to learn and use from the developer point of

[2]Note: This is a draft copy of the document, not intended for public circulation, treat like pre-print.

view, and comes with a few good widgets, apart from the default widgets, like the HTML-browser, rich text editor, dials, etc.

So I have an idea, which is fairly obvious, as we have to make the maximum of Octave being free software. I think we must make a GUI for Octave, which gives similar GUI functionality/interface, like Matlab, Maple, Inkscape, Gnumeric projects.

Various features of the GUI will have to be

2.1 Developers POV(point of view).

```
1: GUI must be based on the liboctave API.

2: GUI cannot invoke the Octave interpreter for running the commands,
   all at the same time, using pipes etc.. Running the interpreter,
   within the GUI, as a passive interface will be a good deal.

3: Use some type of threading, so that you can save the GUI/user
   from crashing applications. User experience must be our priority.

4: Use various free-software components, and copy their source trees,
   into your own project, with a all patches against, their source
   trees, if they'd like it, or just have this new-modified source
   package, that makes life smart \& simple..

   eg: Integrate FLUID within this Octave-GUI, so that you can just invoke
   it using processCreate [win32] or fork; Basically most importantly, make
   editing callbacks in Octave, and generating the stub code in Octave.

   I think its rational, that we must integrate, FLUID with Octave, so
   that you can get a nice \& smooth IDE, work for you.

5: Reuse, Reuse, Reuse. Copy and make use of the free-software advantage:
   Copy freely, acknowledge,

6: Make a nice UI. Let all your fears rest.
```

2.2 useers POV Features list

```
1: [See the mockup diagram ]octave-gui.png[
2: Help for each function/builtin commands,variables.
3: Syntax hilighting Editor, with script editing and running features.
```

4: Access to various flags in plotting, computation, library access,
   PATH settings, preferences etc.
5: Help with Octave-C++ API, header functions, mkoctfile invocations,
   installation etc.
6: Access to command history, tip-of-the day, variable edit/save windows
7: Autocompletion widgets, with some scintillia like component, for making
  syntax hilighting user entered commands, and interpreter results.
8: Various menus, and toolbars for common functions found in almost all
   applications:

   File.
     open/close/save/load/run/print/statistics/print-preview
     import/export/save-as/exit/most-recent files/Launch FLUID

   Edit.
     copy/paste/query|find-replace/undo/redo/variables/workspace/

   View,
     full screen, variables-in-spread-sheet,

   Insert
     function/date/time/file/

   Tools
     preferences/plugin-browser/path-editor/

   Help
     function-documentation/function-search/Octave-GUI tutorial/
     Octave-tutorial/Octave general mailing lists/ CODA documents/
     Demos/ Example Code/ FLUID-guide/Octave-GUI Manual
     {for search, use Lucene package/ Indexer or something similar}
     About, Credits.

9: A graph viewer that integrates, gnuplot within Octave.
   This seems a long shot away. Need we integrate, Imlib.

10: Make using Octave, browsing Octave code \& help functions easy.

```
11: Help suggestions, tips, and a robust application.
```

3 Proposal

Octave-GTK fell short of its aims, because, it cannot probably in its present form really be suitable for crossplatform GUI development, and not make the feature rich GUI environment we need, on small code-print and memory foot-print.

So we must build a GUI. We must integrate FLTK. We must make FLUID work with Octave.

3.1 Plan of Action

1. Contact gnumuthu@users.sf.net, Muthiah Annamalai
2. Download Octave-FLTK-0.6.1 from http://octave-gtk.sf.net/octave-fltk/
3. Send new code.
4. Send Examples.

[3]

[3]Note: This is a draft copy of the document, not intended for public circulation, treat like pre-print.