Building complex GUIs in Plan 9

Jonas Amoson
jonas.amoson@hv.se

University West
461 86 Trollhättan, Sweden

ABSTRACT

How can non-trivial graphical user interfaces be designed in Plan 9 without them losing their minimalistic style? Different toolkits are discussed, and a proposal for a tabbed toolbar is suggested as a way to add functionality without cluttering the interface and avoiding the use of pop-up dialog boxes. A hypothetical port of the GUI in LyX is used as an example.

1. Introduction

The user interface of Plan 9 sometimes confuses newcomers from other systems, as they are used to grouping user interfaces into being either text-based or graphical, and Plan 9 is neither or both. It is graphical from the ground up, and the mouse has to be used all the time, just to realise that most programs are text based command line tools and that the system is configured by editing configuration files.

The next discovery is that the graphical programs look quite different from their cousins on other popular GUIs on X11 or Windows. They are very minimalistic and do without known graphical components such as dialog boxes and buttons with icons. One reason might be an aspiration for a very clean interface where 'traditional' GUI widgets do not fit in. Another guess is that all the graphical programs written for Plan 9 to this date have managed without much help from menus and buttons.

This paper tries to shed light on the design of graphical interfaces for Plan 9 applications focusing on interactive components such as menus, toolbars and dialog boxes. How do one design an application GUI in Plan 9 that has many functions but still blends well with the minimalistic look of the system? Existing Plan 9 applications has been examined in order to rough out a 'look-and-feel' style guide in comparison with traditional WIMP (Windows, Icon, Mouse, Pointer) systems.

Toolkit libraries are the spine of graphical applications, largely defining their behaviour. The graphics libraries libdraw, libpanel, libframe and libcontrol are presented and discussed in the context of what graphical components they provide a GUI programmer.

To concretise, a hypothetical port of the GUI for the word processor LyX is used as a practical example, and a proposal for a tabbed toolbar is suggested in order to keep the interface clean and to avoid pop-up windows.

Some conclusions are drawn from this work, but the main purpose of the paper is to inspire discussions on how to design GUIs for applications with somewhat more complex GUI needs, without losing the minimalistic style of Plan 9.

2. GUI philosophy

This is an attempt to outline a 'style guide' for graphical programs in Plan 9 by observing the current pool of applications in the distribution and also to some extent in the contrib directory. The first observation is:

> **Keep the graphical interface as minimalistic as possible, with as few functions as possible.**

This embodies the principle of 'The fewer the functions, the easier to use' [1]. If some functionality is supplied by another program, do not include it in a new program also. This rule "one tool for each task" or rule of modularity [2] is easier to achieve on the command line than in a graphical interface, but the *plumbing* mechanism [3] in Plan 9 might be a way to let the user access functionality of other tools, such as Postscript previewer, without including extra code for previewing in the application.

There is no need to make everything configurable. Users will accept a well chosen default, and it makes the program less complex [1].

> **Maximise the usage of the space in the working window for actual content.**

The windowing system has no title bars, and most applications have no menus or toolbars that are visible all the time, but use context menus activated by button 2 and 3 on the mouse.

> **A graphical program should stay within the borders of the window it was started in.**

As a graphical program 'takes over' the screen area of the window from which it was executed, the user does not get surprised by windows popping up on the screen, and resizing is fully controlled by the user [1]. As an exception to this rule some programs, mostly games, automatically change the size of the window to a size that fits the application. The changed size persists when the user quits the game.

Popping up secondary windows (such as dialog boxes) is common in GUIs of other systems, but almost unseen among plan 9 applications. Instead some programs like *acme(1)* split the window into multiple frames, using *tiling* [4, p. 348].

Some programs use scroll-bars if the content of a window (or a frame) doesn't fit, whereas some applications simply show as much as fits, eventually forcing the user to resize the window to be able to access the hidden parts.

> **Icons (small stylistic pictures) are usually not used on buttons or toolbars, instead ordinary text is used.**

Graphical images are not evil, but text occupies less space, and is often easier to understand [5, p. 168]. An example is the editable toolbar in *acme(1)* where the shortcuts to commands can be added just by typing.

3. Toolkits

Libraries of GUI components are often called *toolkits* and are constructed to ease application development, so that the programmer does not have to write lots of low-level graphics code in realising a GUI. Toolkits will also highly influence the look-and-feel of an application, so if all programs use the same toolkit, they will probably look better together, than if they where developed using all different toolkits.

All programs that use the graphical subsystem of plan9 will use libdraw, the basic graphics library described in *draw(2).* Libdraw does not provide much help with widgets, but with the help of *event(2)* it is easy to make popup-menus for mouse button 2 and 3, the most commonly used interaction component of graphical applications in Plan 9.

3.1. libpanel

The panel library was developed by Tom Duff in order to write the Mothra web browser [6]. Panel offers traditional GUI components like pull down menus, input boxes, radio buttons, etc. and has a 3D-look-and-feel†, found in most GUIs today.

Although the look and feel of panel is minimalistic compared to many other 3D toolkits, it does not blend so well with the style of other plan 9 applications. Libpanel is not included in the standard distribution (4th edition) of Plan 9.

3.2. libcontrol

The *control(2)* library developed by Rob Pike and Sape Mullender provides a set of interactive controls (widgets) using the thread library, *thread(2).* Each control has its own thread, and it is possible to send it messages and to receive events from it. Libcontrol is, to my understanding, the preferred toolkit for new applications with complex graphical user interfaces.

3.3. libframe

Another library *frame(2)* provides "frames of editable text in a single font" and is used by applications like *acme(1)* and *rio(1).*

3.4. Toolkit features and use

Table 1 shows the GUI features provided by each of the above described toolkits and also an estimate of how frequently the libraries are used by applications in /sys/src.

Table 1: Toolkit summary.

Component	draw	frame	panel	control
Button			x	x
Context menu	x		x	x
Pulldown menu			x	
Slider		x	x	
Text panel		x	x	x
Number of apps in /sys/src	43	5	-	3

4. Case study - LyX

To practically investigate GUI design options for Plan 9, a port of the graphical interface of LyX, the LaTeX frontend, will be discussed.

> *"LyX is a document processor that encourages an approach to writing based on the structure of your documents (WYSIWYM) and not simply their appearance (WYSIWYG)." [www.lyx.org]*

LyX was chosen because it has a featureful GUI that simplifies the editing task compared to using a markup language and a standard text editor. It is also a good example of a GUI frontend for a text based application.

When porting an application, it is easier to break with the style of the target system, than in a fresh design that are built bottom-up. Considerations should be made to follow the style of the target system, or if needed, carefully extend the style in order to embrace new types of applications.

† 3D in the sense that shadowed borders give the impression of a raised or depressed button.

4.1. GUI description

To get a grip of the functionality that has to be ported, we start with an overview over the current graphical interface of LyX. Figure 1 shows a screenshot of LyX running under Linux, with a document loaded into the editing buffer. The interface consists top-down of:

- Pull down menu-bar (File, Edit, View, Insert, ...)
- Icon toolbar with buttons for commonly used functions
- Writing area
- Status bar

The writing area lets the user edit text in different fonts, but also handles *insets* such as figures, tables and equations. The writing area will not be further discussed in this paper. The toolbar with buttons also features a pull-down menu for selecting paragraph types in the document (standard, section, subsection, quotation, ...).

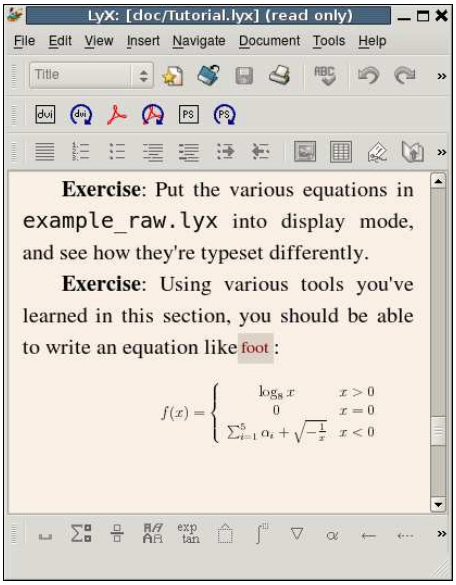

Figure 1: Screenshot of LyX running under Linux.

4.2. User interaction

Some of the pull down menus in LyX open dialog boxes (or pop-up windows) with further options. Figure 2 shows the box for controlling the layout of paragraphs. Settings for the document as whole (meta-data) is configured in a dialog box with a tree graph with sections for each types of configuration.

Most of the pop-up windows in LyX are 'stay on top' but *modeless*†, enabling the user with a large screen to keep the windows open while editing the document. Other

† Modeless as opposed to modal pop-up windows that locks the main window until the user has closed the dialog box [4, p.355]. Modeless windows that are open all the time are often called *tool boxes*.

menu items and menu buttons take direct action in the document, some by inserting a formula box in the document, that is then further filled out by the user.

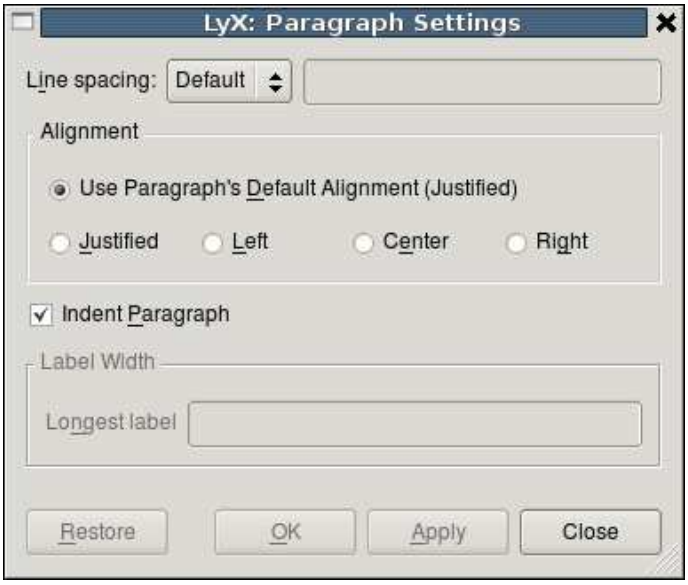

Figure 2: Paragraph settings in LyX.

4.3. Choice of toolkit

There are many possible routes to take in translating a GUI to the Plan 9 environment. One obvious way would be to create an interface as identical as possible to its original form, but the risk is then high that it will feel ill-fitted in its new environment.

For a translation function-by-function, the Panel library would probably be a first choice, as it has the widest support for traditional GUI concepts like pull-down menus and other 3D widgets, see table 1. But as mentioned above, using the Control library would make the application fit better with the overall style of Plan 9, and the discussion that follows assumes a port based on libcontrol.

4.4. Menus

The graphical interface in LyX is full of menus and buttons, whereas the typical plan 9 program is not, so how to proceed? Some functionality can be fulfilled by other tools, and can therefore be left out; this includes Postscript and PDF-export, dialog boxes for opening and saving files as well as printing.

Menu items that cannot easily be omitted, without compromising the usefulness of LyX compared to hand editing the LaTeX code, include setting headings as well as insertion of figures and tables.

Context menus accessed by pressing mouse button 2 and 3 while in the writing area could be used for some functionality, but putting too many menu items, or even sub-level items, in them is discouraged [1]. Many users of Plan 9 have also got used to the mouse chording mechanism in *acme(1)* and would probably wish for similar functionality in other applications for heavy text editing, reserving the mouse buttons for that purpose.

4.5. The toolbar

A toolbar can be seen as a portion of the graphical interface reserved for commands and buttons, in contrast to the main working area, in our case the writing area in LyX. The text editor *sam(1)* has a window for entering written commands and many frame based applications such as *acme(1)* and the web browser *abaco* have toolbars with text buttons.

The toolbar is accessed using the mouse, which would be in line with Plan 9 style, as the system is quite mouse intensive compared to other graphical systems, especially as keyboard shortcuts are not used much. One drawback with the toolbar is that it easily gets messy if it is filled with different buttons and options, another negative aspect is that it wastes available screen space. A proposed toolbar solution for a port of the LyX GUI will be discussed later.

4.6. Dialog boxes

Dialog boxes are quite common in traditional graphical interfaces found on Unix, Mac OS or Windows, and they are used mostly when the application needs to ask the user something more in detail, like where to save a file and in what format. Another use for dialog boxes is for setting preferences, especially in systems where text-based configuration files are avoided at all cost. Dialog boxes have the advantage, compared to toolbars, of not occupying screen area when they are not activated.

As dialog boxes are uncommon in graphical programs designed for Plan 9 it might be desirable to do without them in a port of the LyX GUI. One way could be to sub-frame the working window whenever a dialogue has to be made with the user, the frame is again removed as soon as the user is finished with it. The good thing about this solution is that the application stays within its allocated window area, but the main drawback is that the editing window has to shrink its size temporarily, which may seem even more annoying to the user than the opening of secondary pop-up windows. Another suggestion is to reserve a fixed sub-frame for dialogue purposes, avoiding pop-ups and shrinking frames.

4.7. Dynamic dialog toolbar

If space is to be withheld for a toolbar or for user dialogues, the use of it ought to be as efficient as possible. One idea is to use a menu-tabbed toolbar much like the 'Ribbon' in Microsoft Office 2007 [7]. A crude mock-up of how it could render is shown in figure 3.

The idea with the tabbed toolbar is that the user first selects one of the tabs, say *paragraph settings* which causes the toolbar to be filled with controls associated with paragraphs. Reaching a function now requires an extra click on the tab, but only if the user wasn't already editing the paragraph settings for some other paragraph in the document.

5. Conclusions

An outspoken goal of Plan 9 is to keep the GUI simple and clean. This can be accomplished by reducing functionality or dividing the tasks into multiple programs, thereby eliminating the need for more complex UI structures such as dialog boxes and button toolbars. When this cannot be done without losing too much of the usefulness of having a GUI for the application, new directions have to be found.

For GUIs with lots of functionality, as in the word processor LyX, a dynamic dialog toolbar is proposed as an efficient use of screen space in an attempt to reduce the need for annoying pop-up windows.

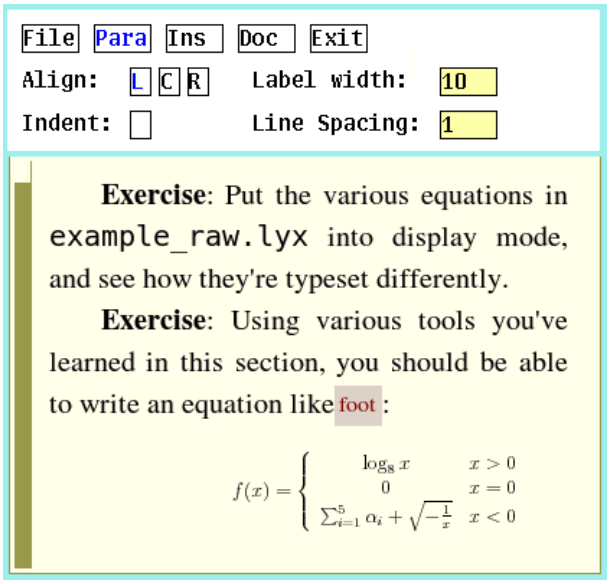

Figure 3: Mock-up of LyX with a tabbed toolbar.

6. References

[1] Rob Pike, *Window Systems Should Be Transparent,* Computing Systems, Vol. 1, 3, pp. 279--296, Summer 1988.

[2] Eric S. Raymond, *The Art of Unix Programming,* Addison-Wesley, 2004.

[3] Rob Pike, *Plumbing and Other Utilities,* Proceedings of the 2000 USENIX Technical Conference, pp. 159--170, San Diego, 2000.

[4] Wilbert Galitz, *The essential guide to User interface design,* Wiley Computer Publishing, 2002.

[5] Jeff Raskin, *The humane interface, new directions for designing interactive systems,* Addison-Wesley, 2000.

[6] Tom Duff, *A Quick Introduction to the Panel Library* http://plan9.bell-labs.com/who/rsc/panel.pdf.

[7] Microsoft, *The Office 2007 Ribbon overview,* http://office.microsoft.com.