Designing Mega Widgets in the Tix Library

Ioi K. Lam
Computer Graphics Laboratory
University of Pennsylvania, PA

If the label-text construct can be packaged into a LabelText mega-widget, the Tcl code needed to create the application will be simplified and easier to read (figure 3). Therefore, by using mega-widgets, the amount of application code can be reduced, applications can be developed more rapidly and the resulting programs will be more structural and readable.

```
tixLabelText .lt1 -label Name:
tixLabelText .lt2 -label Age:
tixLabelText .lt3 -label City:
tixLabelText .lt4 -label State:
```

Figure 3: Simpler Code Using Mega Widgets

2 The Tix Library: A Rich Set of Widgets

Although the standard Tk library has many useful widgets, they are far from complete. The Tix library provides most of the commonly needed widgets that are missing from standard TK: FileSelectBox, ComboBox, Control (a.k.a. SpinBox) and an assortment of scroll-able widgets. Tix also includes many more widgets that are generally useful in a wide range of applications: NoteBook, FileEntry, PanedWindow, MDIWindow, etc. With these new widgets, one can introduce new interaction techniques into applications, creating more useful and more intuitive user interfaces.

The design of the Tix widgets puts an emphasis on the needs of the users. With a wide range of widgets from Tix, application designers can choose the most appropriate widgets to match the special needs of their applications and users. For example, TixControl is an entry widget that allows the user to adjust its value using push buttons. TixComboBox allows the user to choose from a listbox or enter an alternative choice into the entry widget. TixNoteBook makes it possible to pack a large interface into a small window using a notebook metaphor.

Figure 4 shows some popular mega-widgets in the Tix library.

The Tix library is implemented using the *Tix Intrinsics* object oriented framework. Figure 5 shows the class hierarchy of the Tix mega-widgets. The implementation of the Tix mega-widgets and the Tix Intrinsics will be discussed in a separate paper.

3 Designing Application Programming Interface to Mega Widgets

A mega-widget set must have a well-designed, consistent, powerful and flexible application programming interface (API) in order to boost the productivity of application programmers. This section describes the four principles for mega-widget API design that the author has summed up during the two year long development of the Tix library.

3.1 Standard Application Programming Interface

It is highly desirable that the application programming interface to the mega-widgets be exactly the same as that of the standard Tk widgets [1], i.e., each mega-widget is an object, maintains its own set of options and can be accessed by "widget commands". With a standard programming interface, mega-widgets can be easily learned and used by application programmers.

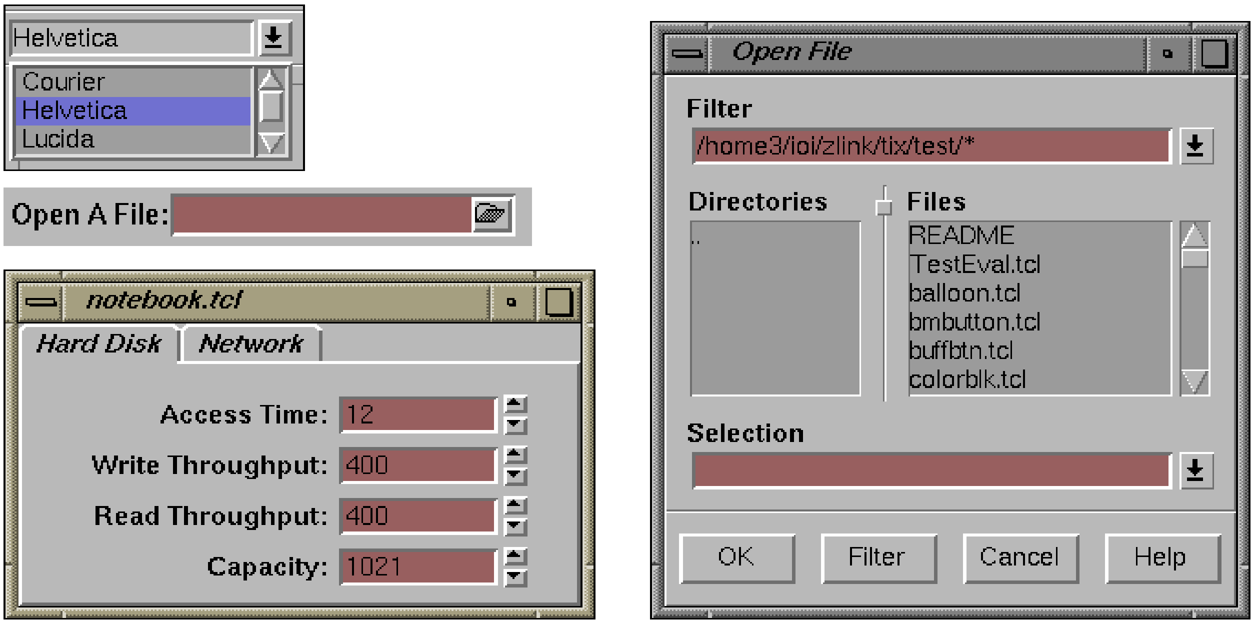

Figure 4: Tix Widgets

Here is a list of the features of the Tix mega-widget API that shows its conformance with the standard Tk programming interface:

- Tix mega-widgets use the same naming convention as Tk widget. For example, `.top.f1.widget1`.
- Tix mega-widget options can be accessed by the `configure` widget command.
- Tix mega-widget options can be specified by the standard X and Tk option database.
- Tix mega-widgets are accessed using the Tk "widget command" mechanism. For example, after the widget `.cbox` of the class TixComboBox is created, it can be accessed as:

```
.cbox config -value "First Choice"
.cbox insert end "Other Choice1"
.cbox insert end "Other Choice2"
```

3.2 Easy Access of User Inputs

The primary purpose of using widgets in an application is to get inputs from the user. Therefore, the API of mega-widgets, as well as normal widgets, should make it easy for the application programmer to access user inputs. Tix makes user inputs available to the application programmer in three ways:

- The `-value` option

  The `-value` option stores the current user input value of a mega-widget. The application can use the "`config`" or "`cget`" widget commands to query or change the current input value.

- The `-variable` option

  The application programmer can instruct a mega-widget to store its `-value` option in a global variable by using the `-variable` option. The

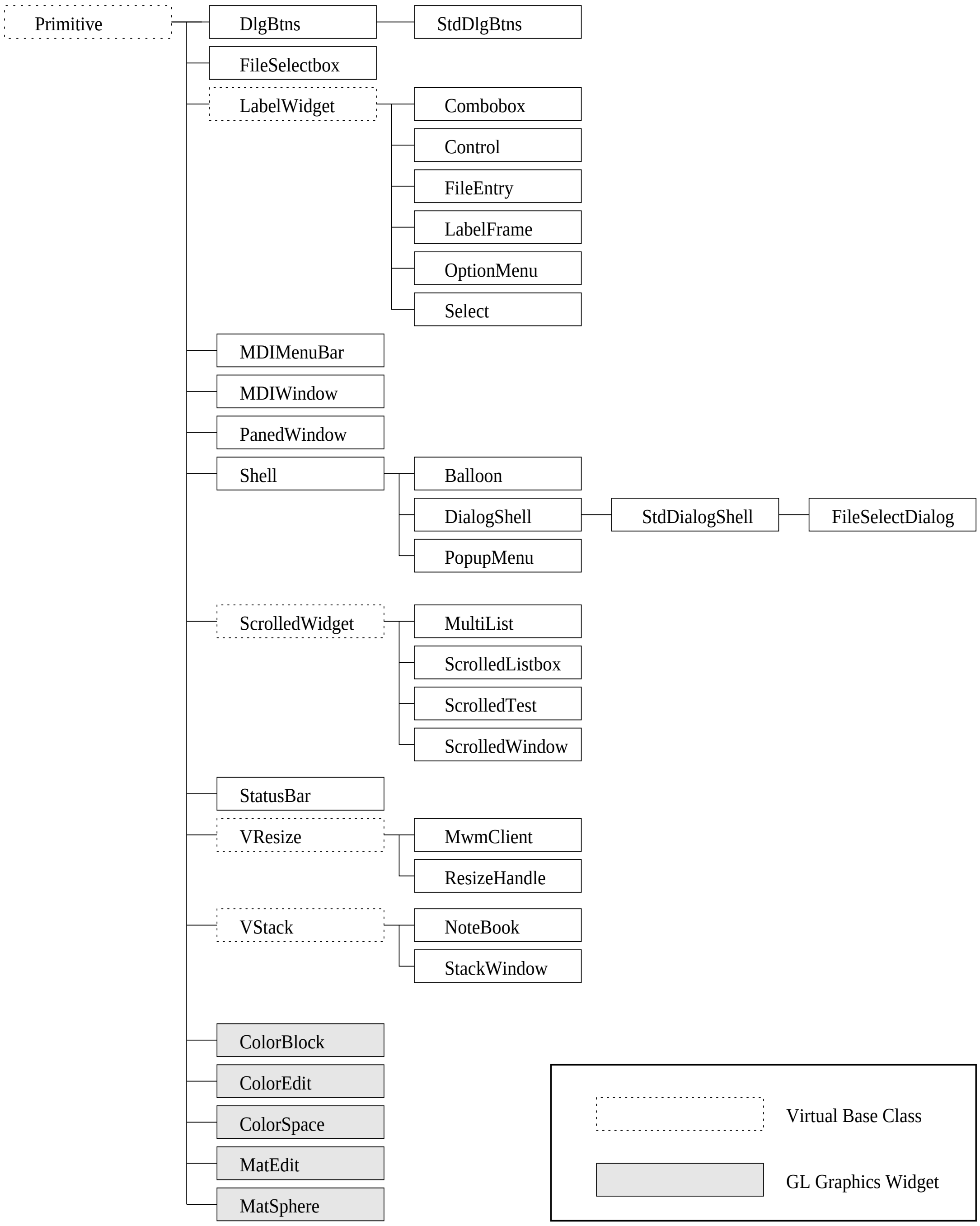

Figure 5: The Tix Mega Widget Class Hierarchy

-variable option can also be used to communicate between several widgets and synchronize their **-value** options.

- The **-command**, **-browsecmd** and **-validatecmd** options

  The **-command** option can be used to specify a Tcl "callback" command to be called when the **-value** option of a mega-widget changes. The **-browsecmd** option can be use to monitor the user's browsing actions in a mega-widget that does not necessarily change its **-value**. The **-validatecmd** option can be used to validate a user's input before it is stored in the **-value** option.

Most Tix mega-widgets, when appropriate, support the **-value**, **-variable**, **-command**, **-browsecmd** and **-validatecmd** options. When used together, these options provide an elegant, consistent and powerful mechanism to access user inputs and respond to user actions.

In contrast, the methods to access user inputs in the standard Tk widgets seem awkward and inconsistent. For example, the Tk **entry** widget does not support the **-value** option and must be accessed by the "**get** " and "**insert**" widget commands. The Tk **listbox** widget does not support **-browsecmd** or **-command** options and must be accessed by event bindings.

The Tk "**bind**" command is difficult to use at best. Once set, the event bindings are difficult to modify and remove, which makes "**bind**" one of the most feared commands for novice Tk users. Moreover, subwidget bindings is a particularly dirty issue in mega-widget programming and so far has not be resolved by Tcl/Tk researchers.

By supporting the **-command**, **-browsecmd** and **-validatecmd** options, the Tix mega-widgets obviate the needs of bindings and thus provides an easy to use, higher-level API for accessing user inputs and responding to events.

3.3 Subwidget Access

Subwidget access is a difficult problem for mega-widget designers. For the sake of flexibility, the application programmer should be able to access subwidgets to a limited degree, e.g., changing the text of button subwidgets or setting the size of listbox subwidgets. However, application programmers should not be allowed to access subwidget arbitrarily.

The most undesirable scenario is to allow application programmers to access subwidgets by their absolute pathnames. For example, if the application knows the pathname of the button subwidget in a ComboBox **.cbox** is **.cbox.btn** and access it as such in his/her code, it would be a nightmare when the next version of the mega-widget package changes the button's pathname to **.cbox.f.btn**.

Tix provides a well-defined mechanism with a convenient syntax for accessing subwidgets. All Tix mega-widgets support the "**subwidget**" command. The subwidgets in a Tix mega-widget are accessed by the **subwidget** command with their "public names".

For example, the public name of the button subwidget in the ComboBox **.cbox** may be **button**. The command

```
.cbox subwidget button
```

returns the pathname of the button subwidget. The command

```
.cbox subwidget button config \
    -bg red -fg blue
```

changes the foreground and background options of the button subwidget. The subwidget command can also be cascaded. For example, if the FileSelectBox widget

`.fbox` contains the ComboBox subwidget `combo`, which in turn contains the button subwidget `button`, the following command can be issued:

```
.fbox subwidget combo subwidget \
    button config -bg black
```

3.3.1 Subwidget Options

Some other mega-widget packages (for example, [incr Tk]) provide special "subwidget options" for configuring subwidgets. For example, a `-buttonfg` option may be used to set the foreground color of a button subwidget.

However, subwidget options are often confusing. For example, when a button subwidget's name is "`down`", one may never be sure whether the option to configure its foreground is `-buttonfg` or `-downfg` or something else. Moreover, when the number of subwidgets are large, the number of subwidget options will explode and will be very difficult for the mega-widget programmer to maintain.

The Tix library completely do away with subwidget options. This significantly reduces the number of options needed for each Tix mega-widget. With Tix, the subwidgets' options can be configured using their public names by either the X/Tk option database or by the `-options` option. Here are a few examples:

```
option add \
    *TixComboBox*button.foreground \
    black
option add \
    *TixComboBox*entry.foreground \
    black
```

or

```
TixComboBox .cbox -options {
    button.foreground black
    entry.foreground black
}
```

3.4 Encapsulating Commonly Needed Features

A mega-widget package must pay extra attentions to the needs of the application programmers. Since a mega-widget is a pre-packaged component intended to make the application programmer's life easier, it should perform all necessary routines tasks to reduce the workload of the application programmer.

For example, one routine task is to put labels next to input widgets to describe their functions. Therefore, many Tix mega-widgets, including ComboBox, Control, FileEntry and OptionMenu, support the `-label` and `-labelside` options so that the application programmer can create what he/she wants with only a single line of code:

```
tixComboBox .cbox -label "Fonts:" \
    -labelside left
```

4 Summary

The Tix mega-widgets are higher-level GUI elements that simplify the application development process. With many new widgets, the Tix library also enriches Tcl/Tk applications by introducing new interaction techniques. The design of the API to the Tix widgets has followed the four principles stated in this paper, which makes the Tix widgets powerful and easy to program. These principles should also prove valuable in the design of other mega-widget packages.

References

[1] M. J. McLennan, "[incr Tk]: Building Extensible Widgets with [incr Tcl],", *Proceedings of the Tcl/Tk Workshop*, 1994

[2] M. J. McLennan, “[incr Tcl]: Object-Oriented Programming in Tcl,”, *Proceedings of the Tcl/Tk Workshop*, University of California at Berkeley, June 10-11, 1993

[3] I. K. Lam, *An Introduction to the Tix Widget Set and Widget Programming in Tcl*, `http://www.cis.upenn.edu/~ioi/tix/doc/tix-1.0/paper.html`, 1993