The powerful text widget in the Tk toolkit offers many facilities to writers of Tcl, Perl and Python scripts.

The Tk text widget

All script writers need to deal with textual data at one time or another. One of the most powerful tools for manipulating text in the free software world is the text widget found in the Tk Graphical User Interface (GUI) toolkit. This widget is available to script writers working with Tcl, Perl/Tk and Tkinter in Python, and boasts features and functionality which can solve just about any text related requirement a script writer is likely to encounter.

Key features include multi-line text display and editing, comprehensive text formatting, embedded images, embedded widgets, and a unique, almost boundless mechanism for endowing dynamic behaviour on areas of text.

This article discusses the features of the Tk text widget which make it a rich and dynamic text manipulation utility. Examples will be presented in Tcl/Tk code, but users of other languages shouldn't have much trouble translating the concepts and examples to their favoured environment.

Before we get started, we need to take a quick look at a couple of concepts employed by the widget.

Indices in the text

Each character stored in a text widget can be addressed by an "index". An index is most often defined by two numbers: the line number, and the character position on that line, so the index "10.45" refers to the character on line 10, position 45.

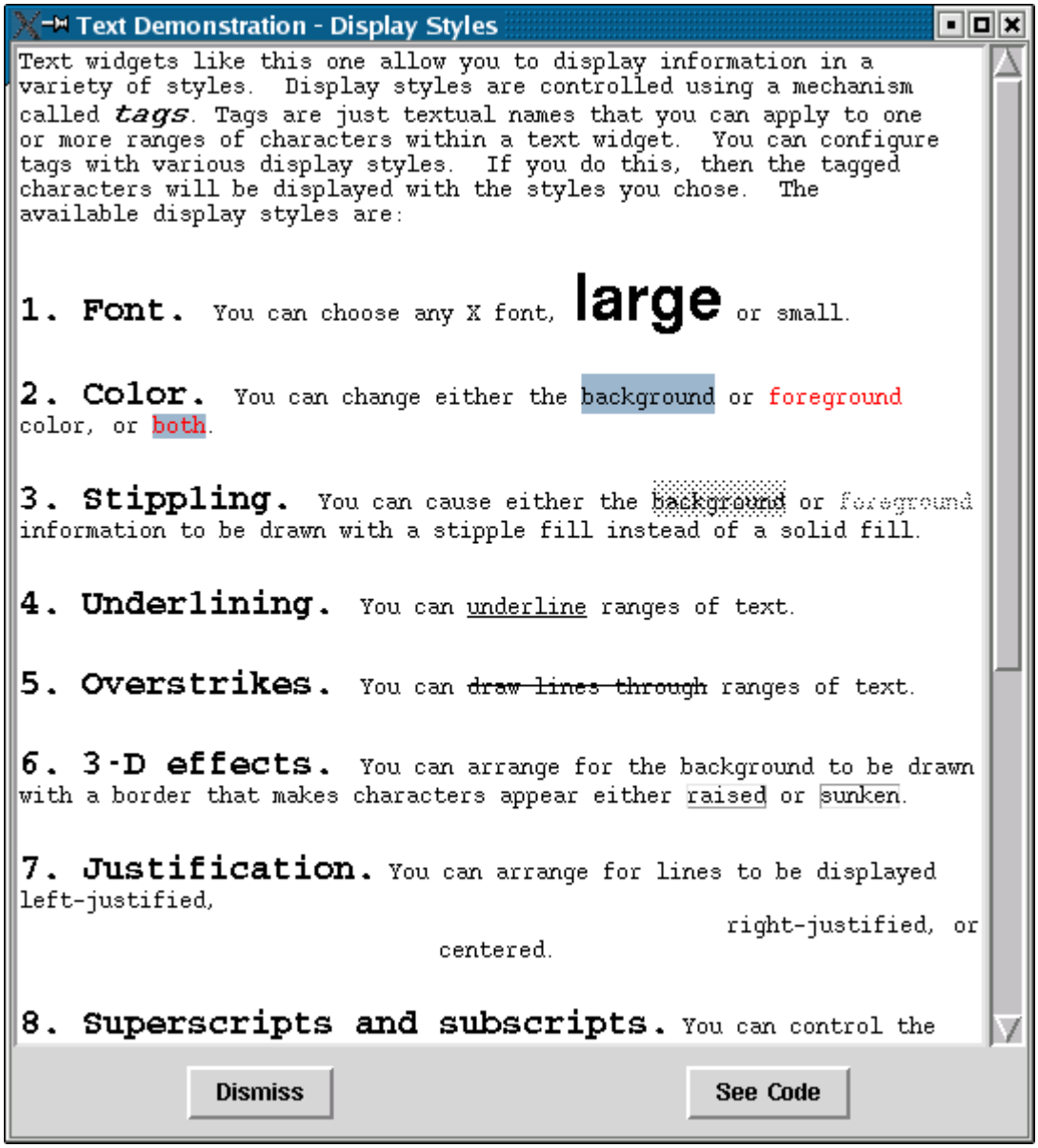
The text widget has built in functionality for index arithmetic – it supports expressions which allow you to find a location, say, 10 lines and 15 characters from a given point. It also handles special indices containing a pixel location (specified as "@x,y"), or containing certain words. For example, "1.end" means the character at the end of the first line, "insert" means the character at the input cursor and "current" means the character under the mouse pointer.

Tagging parts of the text

The feature which provides the Tk text widget with much of its power is known as "tagging". Any area of the text, defined as starting at one index and ending at another, can be "tagged" with a logical tag name. Each tag can be assigned certain attributes and properties, and all the areas of text which have been given that tag will immediately take on the assigned properties. A good part of this article will discuss the properties which can be controlled, and the uses this tagging feature can be put to.

Text formatting features

The Tk text widget provides all the facilities required for rich text editing. This includes flexible font handling, adjustable line spacing and margins, word wrap, colour support, cut, copy and paste, an undo stack and the usual array of bold, underline, italic and other formatting options. Figure 1 is a screenshot of one of the demonstration scripts supplied with Tk. It shows some of the formatting options available.



Using tags to control text formatting

All formatting is implemented using the tags mechanism. The script writer defines a tag with the formatting required, then either inserts the required text with that tag, or "applies" that tag to an area of text already in the widget. For example, for an individual text widget named ".t", a title tag might be defined with a large, underlined font, and centre justification:

```
.t tag configure title -font "helvetica 14" \
                       -justify "center" \
                       -underline on
```

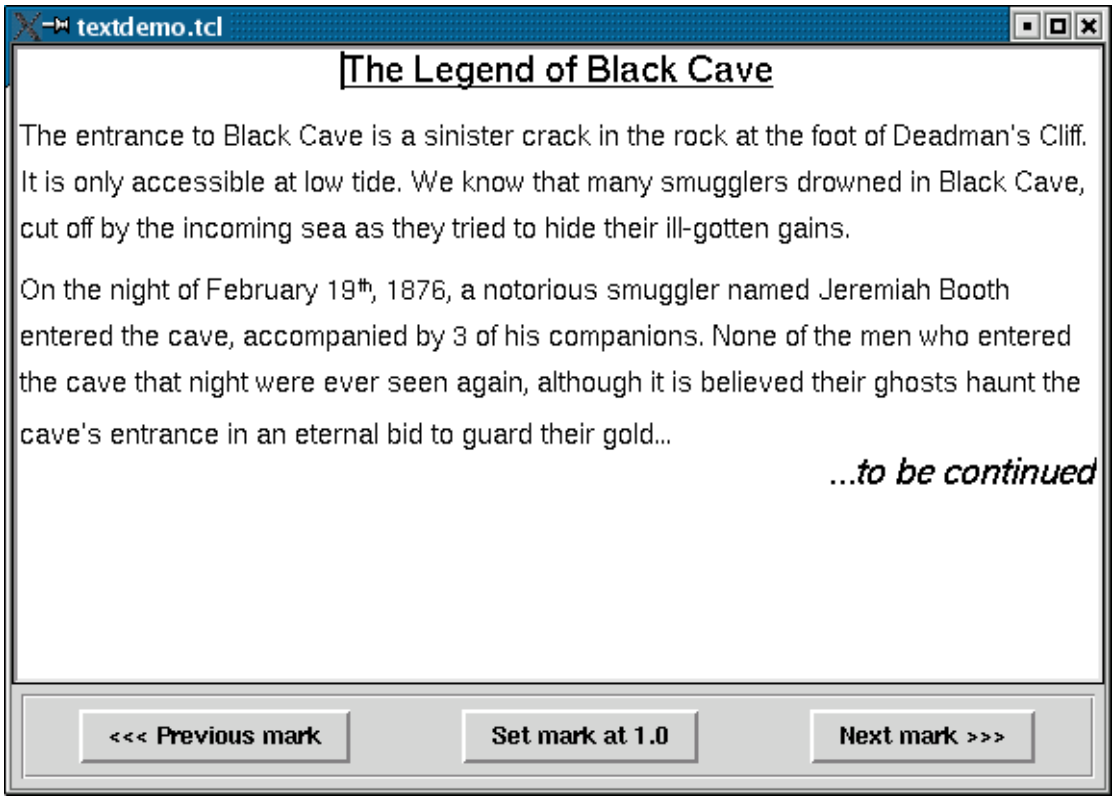
To insert some title text into the widget, this code can be used:

```
.t insert 1.0 "The Legend of Black Cave\n" title
```

This inserts the text at line 1, position 0, with the tag "title" applied. Alternatively, the tag can be applied to text already in the widget using code like:

```
.t tag add title 1.0 1.end
```

This adds the title tag to the text between the given indices – in this case between the start and end of line 1. In a more realistic situation those indices wouldn't be hard coded. They are more likely to be calculated from the location of the selection, for example, or the results of a search. Note the command used to assign a tag to an area of text is "add". This is because an area of text can have several tags assigned to it. For example, our title text might also need to be highlighted as the result of a search, or to indicate its status has changed to urgent.
Figure 2 is a screenshot of a small script which demonstrates formatting via tag manipulation. It also demonstrates the text widget's useful ability to save "marks" – named locations – anywhere in the text.
This script is available for download – see the resources section at the end of this article for the URL.



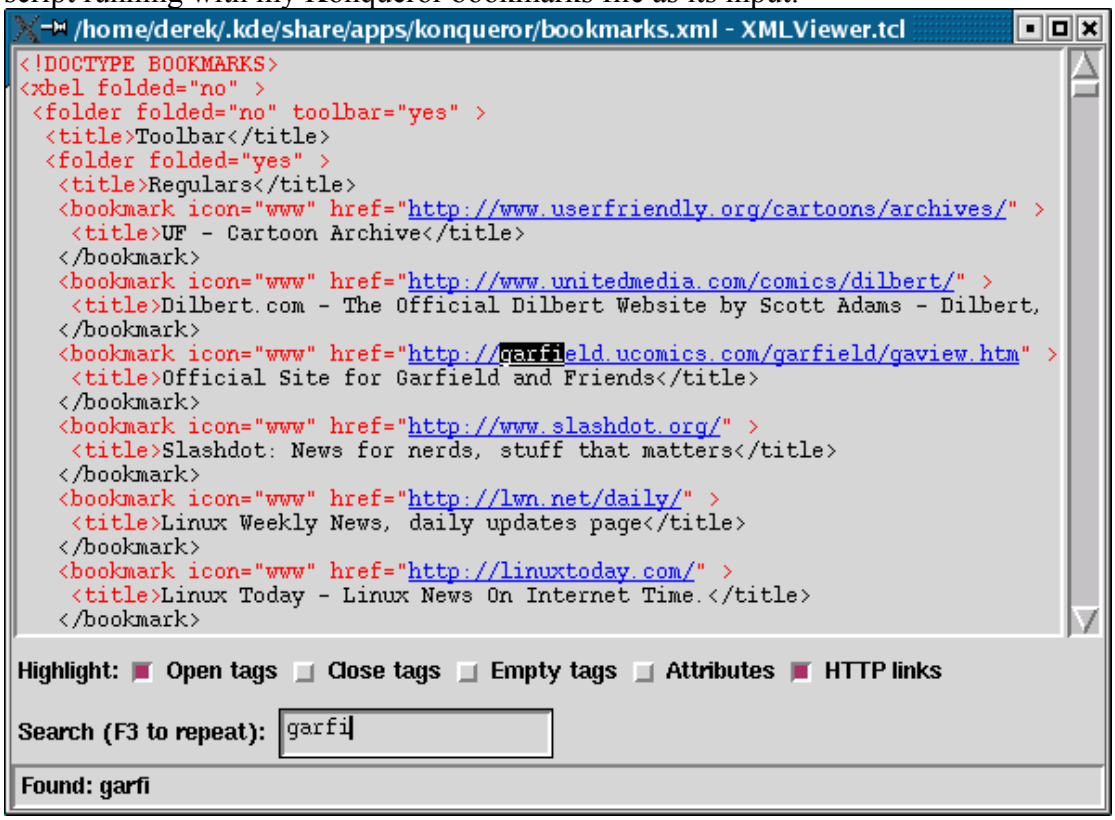
Embedding images and widgets

As well as text formatting, the Tk text widget also supports the embedding of images and other GUI control widgets.
When images are inserted into the text, they "float", so editing of the text around them causes them to move in accordance with the formatting rules configured for that area of text. The same image can be inserted into the widget multiple times, and if an image is dynamically modified elsewhere in the script, its representation in the text widget will be updated immediately.
Other Tk GUI widgets can also be inserted into the text widget. Simple buttons and dropdown lists, as might be found amongst the text of a web page, are just the start. It is possible to embed anything – a drawing canvas, a table or even another text widget. If you have a complicated set of widgets which need to be presented inside some text, the text widget has a mechanism whereby it will only create those widgets when necessary – when the appropriate area is scrolled into view for example.

Advanced features

Text formatting and image and widget presentation is just the top layer of the text widget's box of tricks. Tags provide an extensive array of facilities which can add intriguing dynamic abilities to pieces of text.

In order to make some of the following concepts a little clearer, another small demonstration script is available for download from the Linux Journal FTP site. This script creates a text widget and loads it with the contents of an XML file. Areas of the text are tagged according to their position in the XML structure. Figure 3 shows this script running with my Konqueror bookmarks file as its input.



Searching

The text widget has its own built in search mechanism. A simple search command looks like this:

```
.t search $searchText 1.0 end
```

This will return the index of the first occurrence of the string in $searchText between the start of the text (index "1.0") and the end (index "end"). Implementing a "find next" feature is as simple as changing the starting position for the search to the location the last find returned.

Searches can be configured to go forwards or backwards, be case insensitive, or be made to look for regular expressions as opposed to fixed strings.

Tag and Bind

All of the above demonstrates that the Tk text widget is a flexible solution to most text formatting and editing requirements. However, there's more power to be explored yet, and this is where the programmer needs to start employing some imagination! This extra power is opened up with the text widget's "bind" subcommand

Binding Events

The bind subcommand associates a Tcl script with the occurrence of an X event within a text widget, often, but not necessarily, over a tagged area of text. A simple example might be a mouse over – when the X system reports that the mouse pointer has been placed over a piece of tagged text, we might want to change the mouse pointer shape. Code to bind such an event looks like this:

```
.t tag bind mouseover <Enter> {.t configure -cursor center_ptr}
.t tag bind mouseover <Leave> {.t configure -cursor xterm}
```

The X <Enter> event is received when the mouse pointer enters a certain screen location, and the <Leave> event is received when it leaves it. In this case the "screen location" is any area of text in the .t widget which is tagged with the "mouseover" tag. In this simple example the script run in each case is just a one liner which reconfigures the text widget with a new mouse pointer shape. In a more advanced example perhaps some bubble help would appear, or an acronym would be expanded, or maybe more details could be presented from a database lookup. The only limitation is what can be achieved in the fairly limited timeframe. Given modern hardware and fast scripting languages like Tcl, Perl or Python, a surprisingly large amount of code can be executed within a mouse movement.

Hyperlinks

All X events can be utilised by the bind command, including mouse clicks. An obvious use for this is a hyperlink. Setting up the binding is as simple as before:

```
.t tag bind hyperlink <Button-1> {clickLink %x %y}
```

Now, whenever mouse button 1 is clicked over any text tagged with the "hyperlink" tag, the Tcl procedure "clickLink" is called. The %x and %y are replaced by the bind mechanism with the pixel coordinates of where the mouse click happened. This allows us to find the clicked piece of text. The clickLink procedure, which assumes the clicked text contains a URL, looks like this:

```
proc clickLink { xpos ypos } {

    set i [.t index @$xpos,$ypos]
    set range [.t tag prevrange hyperlink $i]
    set url [eval .t get $range]
    catch { exec mozilla $url & }
}
```

This code finds the index of the character clicked on using the "@x,y" syntax of the text widget's index subcommand. It then uses that index to find the range of text covered by that occurrence of the hyperlink tag. It then gets that piece of text – the URL – and hands it over to, in this case, Mozilla. The catch command prevents the script from stopping if there's a problem starting Mozilla. I've omitted error handling for simplicity - obviously in a real example a problem with Mozilla would need to be handled smoothly. Witness the power of this tag-and-bind approach – hyperlinks implemented in less than 10 lines of code!

Hyperlinks are a fairly obvious use for this functionality, so let's consider some alternatives. Imagine an email client where the user clicks on a subject line in one text widget and sees the text of the email in another. Imagine a programmer's editor where you can click on a keyword or function and see a summary of the syntax it requires.

How about a database front end, where the user hovers the mouse pointer over a table name and sees the structure of that table? These sorts of features are simple to implement with the tag and bind approach.

Caveats

There are very few problems with the Tk text widget. When a tool has been around for as long as this one, it gets honed closer to perfection with each release.
There is one significant problem with the widget as it stands, however. In order to get the full benefits of margin handling and the like, text needs to be inserted without newline characters. That is, each paragraph needs to be one long string with a newline at the end. This allows the word wrapping to work properly. However, the widget's default keyboard navigation behaviour is that pressing the up or down arrow key moves up or down a real line, rather than a displayed line. The effect is to skip up or down a paragraph at a time instead of a line, and so the only way to move to the middle of a paragraph is to hold the right arrow key and allow the cursor to work its way across and downwards.
Given the quality of thought and implementation which has clearly gone into the rest of the widget it's hard to see how this behaviour has been tolerated for so long. The problem has finally been addressed for Tk 8.5, which, as of this writing, is in Alpha release. In the meantime the Tcler's Wiki has a workaround which installs more sensible behaviour – see the resources section.

Conclusion

The Tk text widget is a remarkably powerful tool available to script writers who work with Tcl, Python and Perl. In its basic form it allows multi-line text entry, with rich text formatting and undo facilities where appropriate. Its more powerful features provide a framework for implementing all sorts of interesting facilities, including image handling, embedded GUI controls, mouse overs and hyperlinks.
If your application involves the manipulation or presentation of textual information, the abilities of Tk's text widget should not be overlooked.

Resources

Tcl/Tk headquarters:
http://www.tcl.tk

The Tcl/Tk text widget man page:
http://www.tcl.tk/man/tcl8.4/TkCmd/text.htm

The Tkinter text widget reference page:
http://www.pythonware.com/library/tkinter/introduction/text.htm

Demonstration scripts used in this article:
ftp://ftp.scc.com/somewhere1.tar.gz

The Wiki text widget page, with lots of uses and examples:
http://mini.net/tcl/text

Wiki workaround for strange cursor behaviour:
http://mini.net/tcl/3082