Want to be Tickled

by
Lakshmi Sastry
DRAL
A report for AGOCG

1. Introduction

Tcl (Tool Command Language - pronounced "tickle") and its companion Tk, an X Window System based widget toolkit, were developed by John Ousterhout and his team at the University of California at Berkeley in the early 1980s. Tcl and Tk are freely available, even for commercial software development which explains in part its growing popularity. Unlike other conventional graphical user interface development toolkits (GUIDTs) such as OSF/Motif which concentrate primarily on the look and feel of the applications based on them, Tcl and Tk emphasise "connectivity" and "activeness" that distinguishes hypertext and hypermedia applications [Connectivity enables different things to work together - e.g. hyper-links in documents allow browsing back and forth. Activeness of different objects within a hypermedia system supports complex behaviour such as an annotated text being displayed at a particular point of a soundtrack]. Both Tcl and Tk are designed to be programmable and extendable. This well thought out philosophy explains the interesting extensions to and wide ranging applications based on Tcl and Tk.

An overview of Tcl and Tk and a review (particularly how they compare with other toolkits, in particular OSF/Motif) follows:

2. Overview of Tcl

The impetus behind the development of Tcl is the need for a general purpose command language that is reusable across applications which is also usable to communicate across processes and tools based on Tcl. Tcl scripts are similar to that of other shells such as *sh* or *csh*. Only string input are allowed and the Tcl parser converts these to appropriate types just as other shells do. Tcl is developed primarily on and for UNIX.

Tcl can be used interactively by invoking the Tcl application shell *tclsh* or *wish* (the windowing shell that includes the Tk commands as well).Tcl commands. Typical Tcl commands are:

- *expr 2+2* (calculate the result of the expression 2+2; the result is output to the stdout)
- button .b -text "Hello, World!" -command exit
- pack .b (create a button widget as a child of the main window. with text "Hello World" and the command to exit when the user clicks on the button. The pack command causes the widget to appear on the screen).

Tcl commands can be written to a file and these can be executed from the wish shell much the same way as shell scripts (using the source command). The novel feature is that this shell script can be edited on the fly from the wish shell without restarting the program.

2.1 Tcl and your application

Relationship between application and Tcl

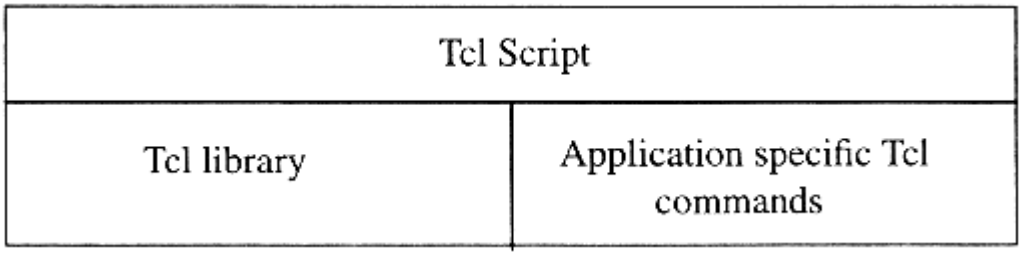

Figure 1
Adopted from Ref.2.

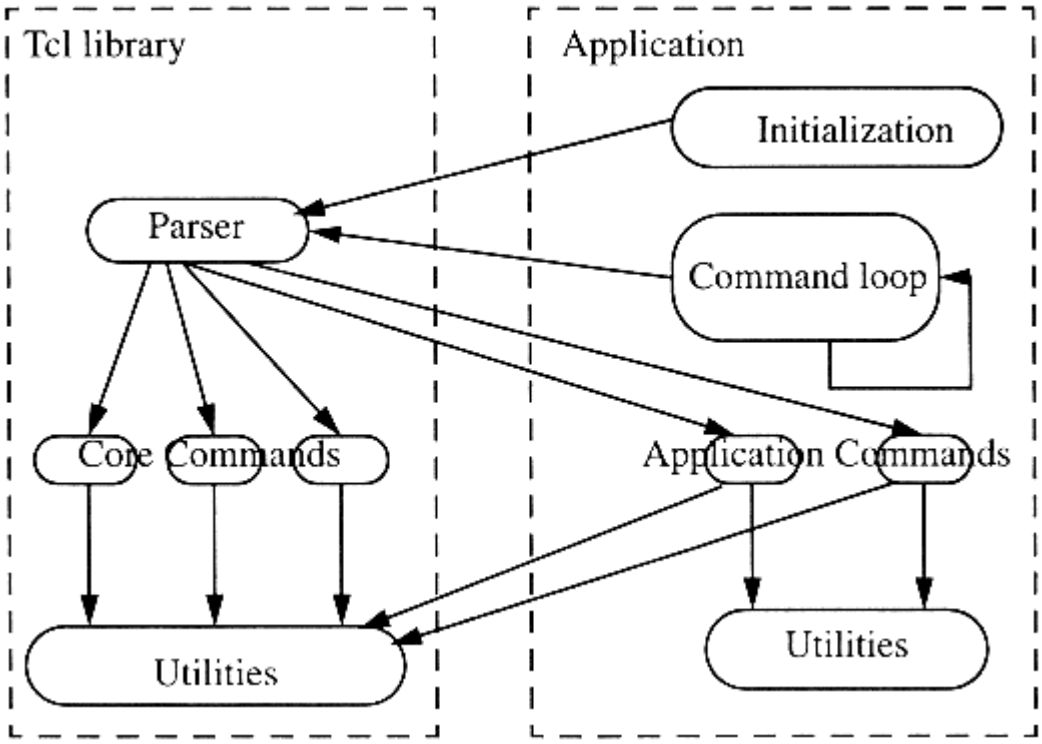

Figure 2
Adopted from Ref.1.

Tcl is an interpreted language. Its parser is implemented as a C library package. The parser is application callable (to parse the built-in Tcl commands that are used by the application) and the C library that implements this parser is linked to the application. Tcl is designed to be extendable. Its incremental approach allows each new application based on Tcl to use some of existing Tcl commands but also add a few new commands specific to itself. There are built-in Tcl commands that allow the user to write his own C or C++ routine, register it as a new Tcl command using built-in Tcl command. All application specific commands are written, compiled and registered this way.

Tcl can be used in one of two ways. There is a simple standard main program as part of the Tcl C library. This can be used by the application to handle the all required initialization, creation of the application context for the parser (more than one Tcl application may be running at a time), enter the event loop, maintain the interface and execute all commands (see Figure 1).

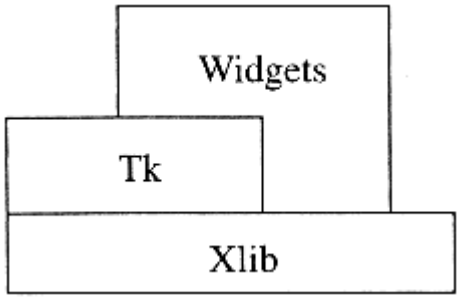

Figure 3
Xlib, Tk and Widgets (courtesy: John Ousterhout

Alternatively the application programmer can choose to write his own main program (since he wants the main program to perform other complex tasks (Figure 2)). In the latter case where an interactive window based application provides its own event loop, it is the applications job to keep the user interface active by using built-in command.

3. Overview of Tk

Tk, developed during the later half of 1980s, has the influence of Apple's HyperCard system. It is based on Xlib and provides a set of about fifteen classes of Motif like widgets. The widgets range from button, labels to inherently sophisticated text and canvas widgets. Tk widgets are created and manipulated using Tcl scripts. Tk widgets also directly access Xlib to provide certain functionality such as drawing (Figure3.). Like Tcl, Tk is also implemented as a C library and can be incorporated into C and C++ applications.

Tk's development period approximates that of other GUIDTs such as OpenLook and Motif. Their development philosophies probably can be compared to the evolution of different branches of early man in that the tools such as Motif concentrated on the look and feel aspect while Tk's development concentrated in providing a powerful interprocess communication facility, connectivity and activeness. Tk supports a natural mechanism (through its built-in ***send***

command) for applications to pass requests (in the form of Tcl scripts) and data between them at run-time. The advantage of this message passing is fully realised, for instance, a Tcl based spreadsheet program can issue display request to a charting application rather than include display routines within itself.

4. Tcl, Tk and the Known Devils

4.1 Tcl, Tk, C and C++

Tcl is very easy to learn, fully programmable and provides features such as variables, associative arrays (arrays whose elements can be addressed by name rather than just indices), lists, control structures such as *if, while* etc. There are only about half a dozen rules on syntax and there is a very rich set of built-in commands and utilities. Since Tcl is an interpreted language it is far quicker to debug and modify the code than a compiled language such as C. By the same token it is slow. Efficiency may prove crucial for applications that perform intensive numerical computations or manipulate large arrays of data. For such applications these functionality must be implemented in C or FORTRAN [FORTRAN routines require a C wrapper for them to be registered with TCl].

Familiarity in C (or C++) is a pre-requisite for Tcl and Tk. All new application specific commands and widgets are required to be written in C (or C++). In the case of widgets, Tcl and Tk developers recommend the programmers to develop application specific widgets to implement sophisticated functionality if required. Since Tk is based on Xlib, this requires the user not only to program in C but to be proficient in Xlib, the lowest application programming interface to X.

Tcl cannot be used in isolation since Tcl does not provide access to low-level facilities such as network sockets. Tcl does not provide complex data structures and in fact the scripts have or impose very little structured programming support. Managing very large applications with hundreds and thousands of lines of Tcl scripts may prove inefficient if not impossible [3].

The above points are raised merely to warn and in no way diminishes the ease, and the simplicity of writing Tcl scripts and interprocess communication that Tcl supports. In fact, the very first encounter with Tcl and Tk is pleasantly surprising. They are, in my experience of both commercial and public domain software packages, the only package that come with well-configured installation scripts that sets up its path by inspecting the installer's load library path. It is a novice's dream come true !

4.2 Tcl, Tk and Motif

Much acclaim is given, deservedly too, for the Tk text and canvas widgets. The Tk text widget provides inherent hyperlink capabilities. The canvas widget has the ability to [4]:

- recognise its content (drawn primitives) as individual objects which can be grouped together to form more complex objects, have one or more tags associated with individual and grouped objects, direct manipulation of these objects (through the ***select*** command)
- provide an insertion cursor so that labels can be inserted interactively.

- generate encapsulated PostScript of its contents which could be used in other documents or made hard copies from.

The above capability points to a graphical application development support similar to that gained by using PHIGS with Motif, albeit with reduced level of functionality that PHIGS supports (no hierarchical relationships, lighting, shading, HLHSR etc). The advantage of using Tcl and Tk is of course their higher level programming interface.

Motif and higher level GUIDTs based on Motif have been supporting hyperlinks in text widgets for quite a while now. Robust and competitively priced interactive GUI development tools for Motif are available which reduce much of the programming complexity of Motif [5]. Availability of the free PHIGS and PEXlib enables the programmer to have access to 3D graphics capability from within Motif based applications.

In comparison with Tk, Motif provides a versatile geometry management and drag and drop mechanism. Motif provides finer control and details over the user interface objects and a variety of callbacks. More importantly perhaps, since Motif is based on the X Toolkit Intrinsic, user defined widgets can be created at the very productive Xt level. The data structures that are manipulated are at this level too. Higher level tools such as XFaceMaker make the creation of new widgets even more easier by making this process interactive[5]. These tools provide the ability to include even animation to the user defined objects. In comparison Tk's binding scripts to user interaction with objects appear limited.

In conclusion, for those who are Motif and C++ programmers, the only compelling reason for using Tcl and Tk is its capability for interprocess communication.

5. Information on Extensions and Applications

Tcl and Tk are available for almost all versions of UNIX. Porting to DOS and Mac are under way and a version is available for VMS. Since it is developed on and primarily for UNIX there are UNIX based Tcl commands such as exec may not available on VMS.

XF is a public domain graphical user interface development tool for Tcl and Tk. XF requires Tcl and Tk programming ability as a pre-requisite. It is distributed along with Tcl and Tk.

Tcl-DP is an extension of Tcl commands to support the development of distributed programs with applications communicating via tcl scripts.

It is beyond the scope of this review to assess each and every extension and application based on Tcl and Tk. So an index of these together with "how to get it" is appended to this article. The index is available on-line along with the Tcl and Tk package.

comp.lang.tcl is the news bulletin for Tcl and Tk related information exchange.

6. Conclusion

To summarise, the following is probably major points holds true about Tcl and Tk:

- The scripting language, even though less clear than C, is at a very productive higher

level. So it is far easier to use, particularly in teaching, than Motif.

- Tcl and Tk is perhaps most useful for those applications that need their own extensive set of application specific objects and methods and/or require other application libraries (graphics packages). In such circumstances, Tcl and Tk acts as a powerful glue and at the same time provides standard user interface objects such as buttons, labels and scrollbars for more mundane tasks.

- For very large scale applications that require efficiency, access to low-level facilities or manipulation complex data structures Tcl and Tk is not ideal.

- Tcl and Tk should be considered as an additional (non exclusive) powerful tool alongside C++ and Motif for writing interesting interactive applications in any specialisation. Much of the hypertext capabilities, a far superior geometry management and finer details and control of interface are robustly supported by Motif.

Together Tcl and Tk support an application development far beyond any other typical graphical user interface development tool. Hence Tcl and Tk is classified as an application development environment. While the power and ease of use of Tcl and Tk is obvious, it has to be remembered that the language and its syntax are non-standard. The shape of things to come once CDE (Common Desktop Environment) becomes more prevalent is unknown at present but the combined commercial momentum cannot be under estimated. This should alert the potential user to questions on the maintenance of legacy code.

So, to answer the question "Do you want to be tickled?" Definitely there are enough reasons to want to be tickled on the right occasions. But beware that intensive tickling can be harmful!

7. Where to Get Tcl and Tk

Latest versions of Tcl and Tk and other extensions mentioned above can be down-loaded from src.doc.ic.ac.uk:pub/packages. The up to date version of Tcl is 7.4 and Tk is 4.0

8. References

1. Tcl and the Tk Toolkit, John K. Ousterhout, Addison-Wesley, 1994

2. Practical Programming in Tcl and Tk, To be published by Prentice Hall.

3. Tickled Pink, Kevin Richard and Eric F. Johnson, Unix Review, March 1994.

4. Hypergraphics and Hypertext in Tk, John K. Ousterhout, The X Resource, Issue Five, 1993.

5. Graphical User Interface Development Tools - Report on the Review and Workshops on the Current State of the Art, ed. Lakshmi Sastry and J R Gallop. Enquiries to mvj@inf.rl.ac.uk.