Hello Button
An Introduction to Tcl/Tk

Peter D. Hiscocks, James Gaston
Syscomp Electronic Design Limited
www.syscompdesign.com
phiscock@ee.ryerson.ca
May 16, 2006
Revised April 28, 2007

Abstract

This paper provides a very brief introduction to the Tcl/Tk programming language, with a focus on using Tcl/Tk to create a 'virtual instrument' graphical user interface control panel on a PC platform, under Linux or Windows. We show some simple examples of Tcl/Tk programming and provide pointers to further information.

Contents

1 Introduction

There are many electronic systems where a Graphical User Interface (GUI) is to be used to control some hardware device. For example, the author once had occasion to build a control system for a Solar Telescope telescope. A GUI was to be used to operate the telescope hardware, which consisted of a variety of stepping motors and position sensors.

A useful approach to a system like this is:

- Interface a microprocessor to the hardware.
- Program the microprocessor with a *command interpreter* that actuates the hardware according to simple commands received at the serial port. For example, when the micro receives the string `R103<cr>`, it operates the right ascension motor for 103 counts. (The `<cr>` indicates a carriage return character, which terminates the string.)
- Build a graphical user interface on a desktop or laptop PC.
- Communicate commands from the GUI on the PC to the hardware via a serial communications link.

This arrangement partitions the system into two sections: the microprocessor and the host PC.

It's easy to interface a microprocessor to hardware, but difficult to write the GUI software. On the other hand, a PC is often difficult to interface to hardware[1], but relatively easy to program a GUI. Consequently, it makes sense to partition the system into the microprocessor that talks to the hardware, and the GUI Program, that talks to the operator.

Debugging

This arrangement also simplifies debugging. The hardware interface can be exercised and debugged from a terminal emulator program, which allows low-level exercising and control of the hardware via direct commands to the hardware.

The GUI program can be developed independently, on the PC platform. When it comes time to plug the PC and microprocessor together, you know that the microprocessor is responding to the correct commands, so it is simply a matter of ensuring that the GUI is issuing the correct command strings.

2 GUI Examples

Figure 1 shows two graphical user interfaces: the Syscomp DSO-101 oscilloscope and WGM-101 Waveform generator[2].

These images are built up from *widgets*. Each widget forms some sort of visual entity on the screen - such as a slider, button, text entry form or frame. These widgets are provided by the Tk (tool kit) part of the Tcl/Tk language.

Tcl/Tk runs under Windows, Linux and Mac operating systems. The graphical user interfaces have the same appearance under either operating system, except to the minor extent of adopting the window gadgets - closer, resizer - for that operating system. So the look and feel of the GUI resemble that of other windows on the desktop.

Each widget may be given different attributes, such as colour or border size, and can be associate with some procedure in the associate Tcl program. For example, clicking the left mouse button on a *button widget* in a GUI causes some particular procedure to be called, and that procedure carries out the necessary action. This action might be to change the content of some register in the interface hardware. The action itself, which could be the clicking of a mouse button, actuating a key on the keyboard, or some external signal into the serial port, is known as an *event*.

A Tcl/Tk program consists of

- defining the various widgets
- assembling the widgets into the GUI frames

[1] And will become more diffi cult as the legacy printer and serial ports are eliminated, and bus cards require PCI capability.

[2] The source code for these Tcl/Tk programs is available on the Syscomp web page at `www.syscompdesign.com`. You are welcome to modify these programs and share your work with others.

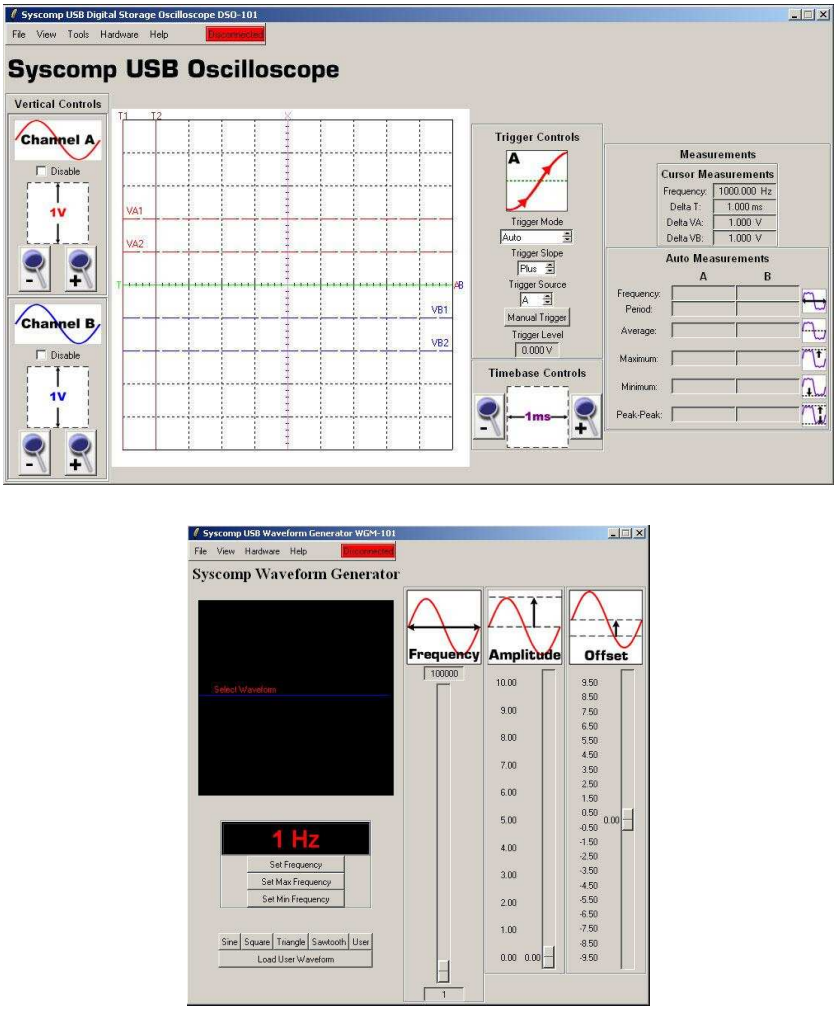

Figure 1: Example Tcl/Tk User Interface

- defining the various procedures to execute events

Notice that it is not necessary to build an *event loop*, as would be required by some languages. The event handler is built into the language interpreter.

3 Installing Tcl/Tk

It is possible to run the Tcl language by itself, without GUI features. We assume here that the main interest is constructing a graphical user interface, so the emphasis will be on using Tk with Tcl, that is, the entire Tcl/Tk language.

3.1 Installing Tcl/Tk Under Linux

The Tcl/Tk language is usually included with a Linux distribution and is part of the default installation. The graphical shell is known as `wish`, for *windowing shell*.

To verify that this wish is installed, type

```
which wish
```

My computer responds with

```
/usr/bin/wish
```

So wish is located in `/usr/bin`. If that is on your path, you may start it simply by typing `wish` at a terminal prompt. There should be a manual entry for wish, which is obtained by typing `man wish`.

You are now ready to construct a Tcl/Tk program, continue with section 4.

3.2 Installing Tcl/Tk Under Windows

The Tcl/Tk language is not part of the Windows operating system, so it must be downloaded and installed. Fortunately, that is straightforward.

Download and install Tcl/Tk for windows from the ActiveState web site:

`www.activestate.com/Products/ActiveTcl`

Follow the installation instructions. At the conclusion of the installation procedure, you should have the Tcl/Tk programming language installed and it will be visible as a new desktop icon such as `WISH84`.

4 A First Tcl/Tk Program

Start `wish`. On a Windows system, double click on the `WISH84` icon on the desktop. On a Linux system, type `wish` at a command prompt.

The `wish` interpreter is illustrated for a Linux system in figure 2. On the right side is a terminal window. The last command executed was `wish`. On the left side is a fragment of the Suse Linux desktop with an empty `wish` window. This window will serve as a container for the Tcl/Tk program output.

Figure 2: Wish Window under Linux

Back to figure 2 and the terminal window on the right. The normal prompt is:

```
phiscock@panther:~>
```

Once you have started `wish`, that terminal window becomes a console for entering commands directly into the `wish` interpreter. If you've ever used `BASIC`, you'll be familiar with this: `Tcl/Tk` has a *command* mode in which it can executed instructions directly as they are typed in. It also has a *program* mode, in which it will execute a text file containing a program. The *Tcl/Tk console* mode is indicated by a change in prompt:

```
phiscock@panther:~> wish
%
```

The `%` sign indicates that the *Tcl/Tk console* is active and waiting for commands. Click the left mouse in the console window to move focus there, so you can type commands into the window.

To bail out of this mode, simply type `exit` at the prompt.

As a simple Tcl/Tk command, try this at the Tcl/Tk console prompt:

```
% clock seconds
```

and you should get something like:

```
1147753154
```

which is the current time, formatted as seconds. (It is possible to reformat this string into a more readable date.)

4.1 Hello Button

Now we are ready for our first Graphical User Interface program. Type in the following line at the Tcl/Tk console prompt:

```
button .b -text "Hello" -command {exit}
```

(You can see the Tcl/Tk commands in the terminal window on the right side of figure 3.)

What we have done is created a button widget named `b` that is a child of the top window known as `.` (dot). (It's a Unix tradition to call the current top-level (root) directory `.`). This button contains the text label `Hello`. When the button is actuated by a left mouse button click, it will execute the command `exit`.

The Tcl/Tk console responds with:

```
.b
```

which confirms that the button has been created correctly. Now we need to `pack` the button into the root window:

```
pack .b
```

The button appears as shown in figure 3. The button is surrounded by a fragment of the root window that is somewhat larger than the button is – because the root window must carry a label and the usual window tools.

Left-click on the button and it disappears - it carries out the `exit` command and executes itself.

That's it! You have constructed a graphical user interface.

If you have used some other window construction method, you will notice that this is very simple and straightforward:

- The button widget assumes reasonable defaults, so we didn't need to specify the colour, the size, the width of the borders, the size of the text and so on.
- The Tcl/Tk interpreter provides immediate feedback on the actions of the program. There is no need to go through a compile or link stage.
- It is not necessary to construct the *event loop* that handles button clicks and other actions. It is and integral part of the language.

4.2 Executing a Text File

Direct entry of commands is a useful way to test Tcl/Tk commands. However, it's usually much more convenient to create program with a text file that is then executed by the Tcl/Tk interpreter.

First, create a text file containing the Tcl/Tk program: we'll assume that it is called `hello-button.tcl`.

Figure 3: Hello Button

Executing a Text File: Linux

Change to the directory where `hello-button.tcl` is located.
Start the `wish` interpreter by direct entry of the `wish` command.
Into the console window enter the command:

```
source hello-button.tcl
```

The program should run.
You can terminate the program by typing the usual `<ctrl>c` command.

Executing a Text File: Windows

Change to the directory where `hello-button.tcl` is located.
Double-click on the file, and it should run.
If it does not run, you may need to set up the *file association* so that the operating system associates files ending in `.tcl` with the `WISH` program. To do that:

1. Right-click on `hello-button.tcl`
2. Check that `Opens With` shows `WISH84` (or whatever your `WISH` interpreter is named). If this is not the case, then:
3. Select `Change`. This brings up the `Open With..` dialogue.
4. Select `Browse`. This brings up a file selector dialogue.
5. Now find the `WISH84` icon on your desktop. Right click on the `WISH84` icon, select `Properties` to determine the location of the `WISH84` program.
6. Copy that location into the previous `Browse` dialogue.

Now, double-clicking (left button) on the `hello-button.tcl` file should cause it to run.
Fortunately, you should only have to do this once.

5 Questions and Answers

1. *I'd like to start learning the Tcl/Tk language. What do I need to purchase to get started, and where should I get it?*

   You don't it need to purchase anything. You do need two things: a Tcl/Tk interpreter and a text editor (see section 7.3 below). The Tcl/Tk interpreter comes with your operating system (Linux) or is a free download from Activestate (`www.activestate.com`). Any text editor will do, though it's helpful to get one that has keyword highlighting for the Tcl/Tk language.

   On the other hand, Activestate do sell an IDE (Interactive Development Environment) that facilitates Tcl/Tk code development. This includes a compiler, which is necessary if you want to prevent others from reading your code. The IDE is useful for those who do a lot of code development and need to have the absolute best working environment.

2. *Where can I find some examples of Tcl/Tk code?*

   There are *many* examples on the web. Google `Tcl/Tk Examples` or `Tcl/Tk Tutorial` and you'll find them. One excellent source is the Tcl Wiki at `http://wiki.tcl.tk`.

   Of course, you are welcome to read the source code for the Syscomp instruments. It's available from Sourceforge under OIP: *Open Instrumentation Project*.

3. *Where can I find a command reference?*

   If you are operating a Linux system, the commands are available using 'man'. for example, `man after` gives a synopsis of the Tcl `after` command. Be warned, this information (like all `man` entries) extremely terse and devoid of examples.

   If you are operating a Windows system or would like more information with examples, the Tcl Wiki has some pointers to online references. For example, see
   `http://aspn.activestate.com/ASPN/docs/ActiveTcl/8.4/tcl/TclCmd/contents.htm`

6 Continuing On

Now you have the basics, you can begin to learn the language and write your own Tcl/Tk programs. A huge amount of information is available on the Web and in the books listed below. In general, a Tcl/Tk program may be loaded into a text editor and then examined, so it is possible to learn by reading and running examples that are available from these sources.

7 Resources

7.1 Books

The definitive text reference is [1]. Other books, such as [3] and [5] are useful alternate sources for information and examples. Other books are listed in the *References*.

7.2 Web Pages

A useful introduction to the Tcl/Tk language is at `http://www.tcl.tk/`.

The web page at `http://www.tclscripting.com/resources.html` points to a number of resources on the web.

A Tcl/Tk Wiki is at `http://wiki.tcl.tk`

The Tcl/Tk developers have worked very hard to make the Linux/Unix version of Tcl/Tk equivalent to the Windows version. However, there are some unavoidable differences. The FAQ at the URL given below is somewhat outdated, but lists many of the problem areas.

`http://www.faqs.org/faqs/tcl-faq/tk/windows/`

A high level view of the Tcl/Tk scripting language, by John K. Ousterhout, its inventor.
`http://home.pacbell.net/ouster/scripting.html`

7.3 Editors

You may have a favourite editor for creation of the Tcl/Tk files. One of us (Hiscocks) uses the `joe` editor under Linux, which is a functional general purpose editor. Mainly, that's a consequence of habit. However, the `joe` editor does have a nice `help` screen that can be invoked easily, and that helps the learning curve.

The other author (Gaston) uses the `Scite` text editor, which is probably a better choice. It can collapse sections of the code and it can highlight Tcl/Tk keywords. Highlighting can be very helpful when trying to reverse engineer code written by someone else. Scite also runs under Windows and Linux, so its a good choice for cross-platform development.
`http://scintilla.sourceforge.net/SciTE.html`

Under Windows, something as simple as `Wordpad` or `Notepad` will work, but they lack many of the creature comforts of a full programming editor.

Other editors such as `vi` and `emacs`, will work satisfactorily and are even, surprisingly, preferred by some users. `;)`.

References

[1] *Practical Programming in Tcl and Tk, 3rd Edition*
Brent B. Welch
Prentice Hall, 2000

[2] *Tcl/Tk for Programmers*
J.A.Zimmer
IEEE Computer Society, 1998

[3] *Graphical Applications with Tcl and Tk*
Eric Foster-Johnson
M&T Books, 1997

[4] *Effective Tcl/Tk Programming*
Mark Harrison and Michael McLennan
Addison-Wesley, 1998

[5] *Tcl/Tk, Second Edition : A Developer's Guide*
Clif Flynt
Morgan Kaufmann, 2003

[6] *Tcl/Tk Tools*
Mark Harrison
O'Reilly, 1997

[7] *Teach Yourself Tcl/Tk in 24 Hours*
Venkat Sastry, Lakshmi Sastry
Howard Sams, 2000

[8] *Tcl/Tk for Dummies*
Tim Webster
IDG Books, 1997