Appendix A

Extra notes on Tcl/Tk

This section includes some extra material related to the use of Tcl/Tk for developing GUI applications. In particular - constructing menu items, using the Tk Canvas and structured data items. There are pointers to some supplied reference material. Note the following points related to trying out Tcl/Tk:

- If you are using **cygwin-b20**, the wish interpreter is called **cygwish80.exe**. This file is found in the directory **/cygnus/cygwin-b20/H-i586-cygwin32/cygwish80.exe**. Make a copy of this file in the same directory, and call it **wish8.0.exe** for compatibility with UNIX Tcl/Tk scripts.
- In the first line of your tcl files, you should put **#!wish8.0**
- If you download the file **~cs3283/ftp/demos.tar** and extract it into **/cygnus**, you will have a series of Tcl/Tk widget examples in **/cygnus/Demos**. Change into the directory **/cygnus/Demos**, and type **./widget**.
- There is a Tcl/Tk tutor, and many learn-to-program-Tcl/Tk documents available at many sites on the Internet - if you continue to have trouble, you may wish to try them.

There is no substitute for just trying to program - set yourself a small goal, and discover how to do it in Tcl/Tk.

A.1 Tcl/Tk menus

The menu strategy is fairly simple -

1. Make up a frame for the menu
2. Add in the top level menu items
3. For each top level item, add in the drop-menu items
4. For each nested item, add in any cascaded menus.
5. Remember to pack it...

As an example, the following code creates a fairly conventional application with menus, a help dialog, and cascaded menu items.

CODE LISTING **Menus.tcl**

```
#!/usr/bin/wish

frame .mbar -relief raised -bd 2
pack .mbar -side top -fill x

frame .dummy -width 10c -height 100
pack .dummy

menubutton .mbar.file -text File -underline 0 -menu .mbar.file.menu
menu .mbar.file.menu -tearoff 0
.mbar.file.menu add command -label "New..." -command "newcommand"
.mbar.file.menu add command -label "Open..." -command "opencommand"
.mbar.file.menu add separator
.mbar.file.menu add command -label Quit -command exit
pack .mbar.file  -side left

menubutton .mbar.edit -text Edit -underline 0 -menu .mbar.edit.menu
menu .mbar.edit.menu -tearoff 1
.mbar.edit.menu add command -label "Undo..." -command "undocommand"
.mbar.edit.menu add separator
.mbar.edit.menu add cascade -label Preferences -menu .mbar.edit.menu.prefs
menu .mbar.edit.menu.prefs -tearoff 0
.mbar.edit.menu.prefs add command -label "Load default" -command "defaultprefs"
.mbar.edit.menu.prefs add command -label "Revert" -command "revertprefs"
pack .mbar.edit  -side left

menubutton .mbar.help -text Help -underline 0 -menu .mbar.help.menu
menu .mbar.help.menu -tearoff 0
.mbar.help.menu add command -label "About ThisApp..." -command "aboutcommand"
pack .mbar.help  -side right

proc aboutcommand {} {
    tk_dialog .win {About this program} "Hugh wrote it!"  {} 0 OK
}
```

A.2 The Tk canvas

The Tk canvas widget allows you to draw items on a pane of the application. Items may be tagged when created, and then these tagged items may be bound to events, which may be used to manipulate the items at a later stage.

This process is described in detail in Robert Biddle's "Using the Tk Canvas Facility", a copy of which is found at **~cs3283/ftp/CS-TR-94-5.pdf**.

Note also the use of dynamically created variable names (**node$nodes**).

Tutorial 5 - questions for week 5 (Feb 6, 2002)

All these questions relate to practical programming concerns in Tcl/Tk.

1. How can you create a cascade menu item where the cacaded items are radiobuttons (i.e. - only can select one at a time).
2. Tcl only supports a simple idea of variable scope. How can you create global variables? How can you create local variables? How can you refer to global variables within a proc?
3. How would you create structured type variables in Tcl? (i.e. a variable with various sub-components).
4. How do you change the name on the surrounding window decoration for an application?
5. How can you create a file-selection dialog box when you click on an 'open' menu item? How does your appliaction know which file was selected? How does your application know when no file is selected?
6. How can you change the cursor (to -say- a watch), when it moves over a particular item embedded in a canvas?

Further study

- TclTk widgets:
  http://www.comp.nus.edu.sg/~cs3283/ftp/CS-TR-94-5.pdf,
  http://www.comp.nus.edu.sg/~cs3283/ftp/demos.tar.

A.3 Assignment 3 - Implementation

I have moved assignment 4 here, as we have not yet got to Java/Swing.

In this assignment, you may work in a group of up to 4 people - or you may do it by yourself. The assignment is worth 30% of your assignment grade (10.5% of your final mark), and is due at 5:00 p.m. on Friday, 8th March, 2002 - please deliver to Hugh's room. The assignment is to be done using Tcl/Tk or Perl/Tk or C/Tk.

Task:

Your task is to implement and document a user interface for a new software analysis tool, which I will attempt to describe below. Note that this is a real task, and parts of this tool have already been prototyped. However I am looking for better ideas :)

The interface is intended to manage the overall process of an analysis of program source code. Beginning with some initial document, the developer uses (external) program analysis tools[1] to transform an existing document into a new document. This document in turn may be further transformed using the same, or different analysis tools.

The user of your application will use a range of different analysis tools, in different orders, and some analysis sequences will be useful, and others may not. However all transformations are to be recorded.

The first prototype of the user interface is shown in Figure A.1, however, your version need not look at all like this - my expectation is that you should be able to come up with better interfaces - I am a little worried about putting this one up, because It may *limit* you in your development (which I do not want to do). The interface maintains a library of documents. The activities performed here are:

- Entering new documents into the library from external source code.
- Transforming existing documents using a range of analysis programs.

The process of applying analysis programs to transform documents in the library leads to a library structure which is a directed graph without loops. This may be represented by a hierarchy diagram, and this sort of diagram is used as a basis for the interface.

The idea here is that each dot represents a document, with the directed arrows representing the application of an analysis program to a document. Operations on documents are done by selecting the document in the GUI interface, and then selecting actions (in this case from a menu). Once the external analysis program has created a new document, it is displayed as a dot in the interface, with its directed arrows.

[1] These tools have already been developed in a separate project, but may be run as commands using the **exec** Tcl facility. For example **prove1 <filein.txt >fileout.txt**.

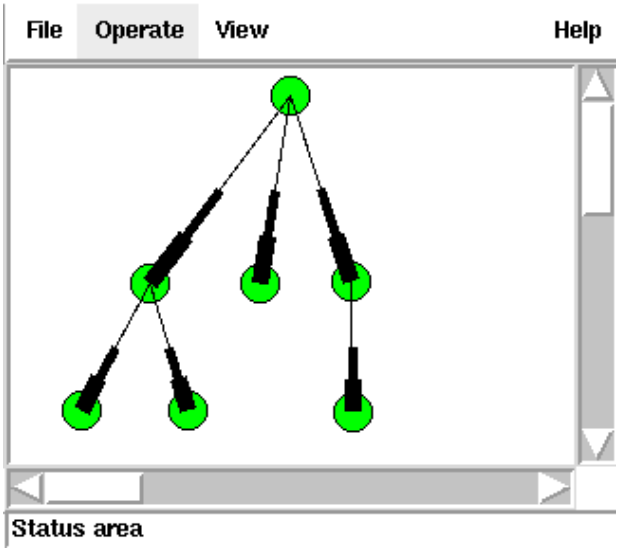

Figure A.1: Document management tool

The graphical view displayedshould be easy to navigate, and display a complete history of the program analysis.

Notes:

The following points form part of the functional specification of the user interface.

1. At any time you should be able to save and restore the state of the development (i.e. all the things visible, and the state of the library).
2. The program should be safe on failure - that is - if an external transformation tool dies for some reason, it will not affect the stability of your library and display.
3. The general flow of operation of the interface is you select (one or more) documents, and then select a single analysis tool, which is passed the selected documents, and returns a new document. The new display should reflect the new document's position in the library.
4. The document names are managed by your tool, which might name the documents according to some internal strategy (doc1, doc2, doc3 :). However - the tool should also maintain descriptions associated with each document, that may be entered by the user, and displayed later. (Perhaps something like - if you right-click on a document, you get the description).
5. The transforms between documents also may have attached descriptions, editable and displayable in a similar manner.
6. The descriptions should be time stamped.
7. The transforms should be time stamped.

Tips:

I'm just guessing here, but

1. You probably have to maintain a configuration file (or files) for the library - containing the state of the display, and the names and descriptions of the elements in the display.
2. Rather than hard coding which transform tools are to be used, you might read a configuration file containing the tools and which menu/button items they are to be associated with.
3. You may dummy up tools, (tool1, tool2 tool3 and so on), by just using a simple program like cat or copy, to make an exact duplicate of the document.

Deliverables:

- A title page containing your names and matriculation numbers.
- A ten to twenty page document containing
  - A brief summary of the overall design constraints
  - An overview of the interface design
  - A user manual for the interface
- A disk containing the code, with a (small) README file to explain how I am supposed to run your software.

Note that this assignment requires you to implement the application, not just to design one.

Assessment:

The assessment is as follows:

Documentation	25%
Code style/quality	25%
Operation of the interface	50%