

Tcl is a script based programming language similar to shell, awk, and perl scripts – that works well with numerical expressions.

Tk is a X like windowing toolkit similar to Xt and Motif

Tkinter is the "TK interface" for other languages: Perl, Python, Ruby ...

Tcl/Tk ("*tickle*") is an alternative method for developing X applications.

There are ports of tcl/tk to MS Windows and Mac OS...

```
$ wish < myFile
```

when wish is not in /usr/local/bin or appropriate PATH set for OS

J.K. Ousterhout, **Tcl and the Tk Toolkit**, Addison Wesley, 1994.

OpenSource distributions of Tcl/Tk

```
http://www.activestate.com/Products/ActiveTcl/
```

```
http://tcl.sourceforge.net/
```

```
#!/usr/local/bin/wish -f
button .b -text "Hello, world!" -command exit
pack .b
```

The above script will create a window with a labelled button. Clicking on the button will execute the exit command.

wish is a tcl Window Interpreter SHell

b is a widget instance of the button widget class, a child of (.) toplevel (assumed)

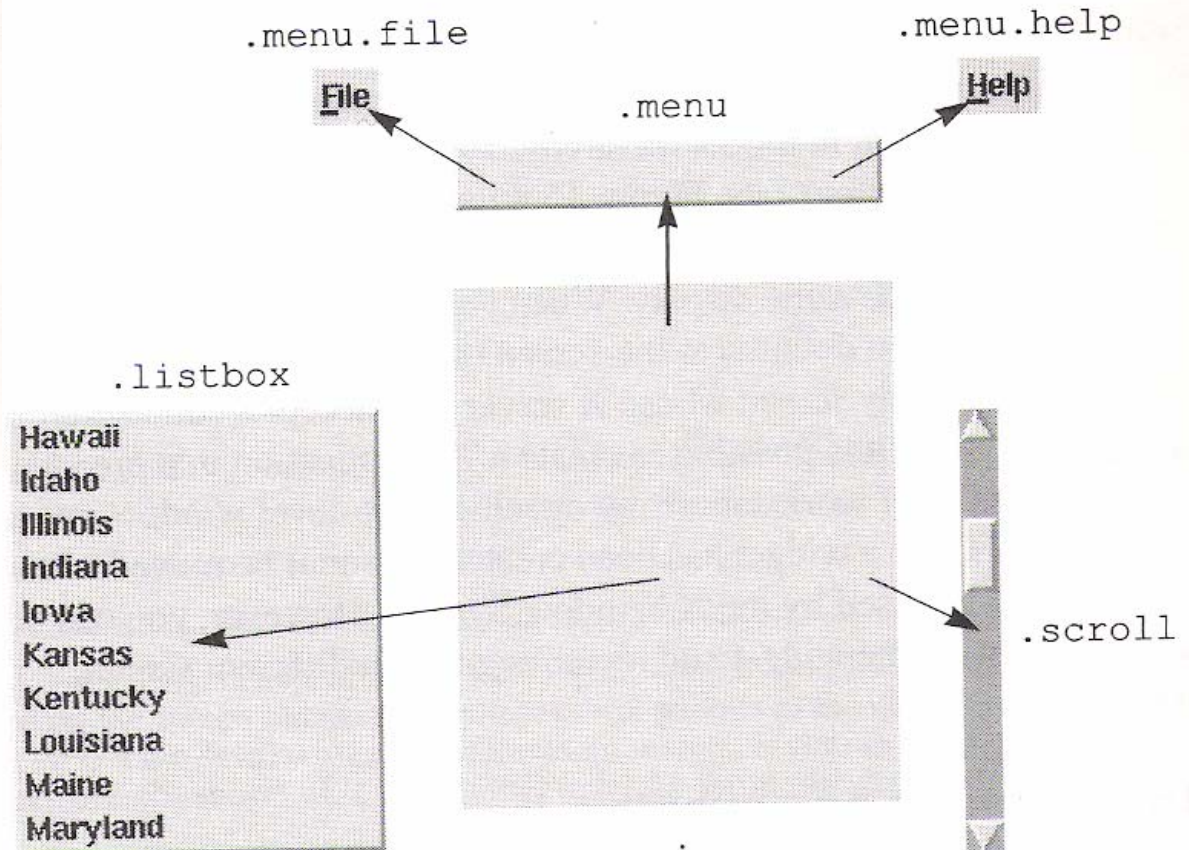
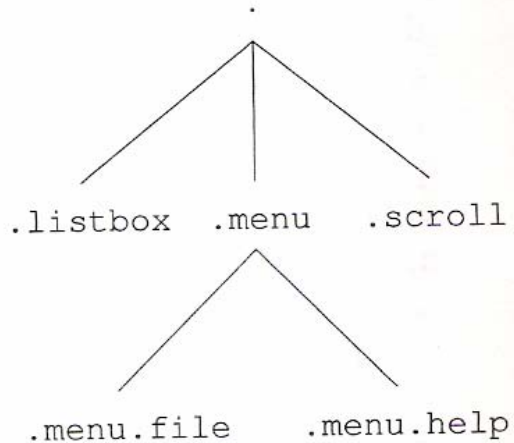
pack "manages/maps" .b as a child of toplevel

See *<yourTclPath>\Demos\TK8.4\widgets.tcl*

Widget names and hierarchy

All widgets in an application exist in a hierarchy from "."

Dialog widgets are toplevel widgets -- but are also in the hierarchy.



Each widget occupies a window.

Each widget has a **class** (use) and options

frame (group widgets) relief, borderwidth, color, foreground,
background (all widgets have these options)

toplevel (interface with window manager) screen

label (display text, bitmaps) text, font, bitmap

button, checkbutton, radiobutton (respond to button events)
text, variable or value (radiobutton sets),
command (callback procedure)

message (display multiline strings) width, aspect, justify, text

listbox (collection of strings -- entries)

scrolling (mouse 2 drag moves entries like scrollbar),
commands (entry insert, delete, get (retrieve))

scrollbar (controls view in other widgets)

yscroll, xscroll, command, orient (default vertical)

scale (edit variable values via slider)

length, from, to, tickinterval, command

entry (text edit) commands (insert, delete) index, icursor, textvariable, width

menubutton (buttons on menubar, menus are added to menubuttons).

menu (menubar selections, entries that can be added) entries are:

command, checkbutton, radiobutton, cascade, separator.

menus also have post. (with dialogs can be pop ups).

underline (keyboard accel), label, text,color, accelerator

canvas (drawing area) create (circles, rectangles, lines, text, bitmaps, widgets), tags and identifiers (ways to manipulate canvas items with events)

text (text field display and edit multiline text) insert, delete ...

Tcl/Tk uses a "master/slave" terminology to describe parent/child window management.

pack is the packer command (do layout mgmt).

Options are:

- after or before widget

- expand boolean (1, default is false 0) slave absorbs extra space

- fill describes how to expand (x, y, both, default is none)

- in determines who master is (default is widgets parent)

- master must be parent or descendant of parent.

- ipadx, ipady specifies distance for internal padding inside slave (0)

- padx, pady specifies external padding (outside slave inside container)

- side specifies attachment (top (default), bottom, left or right)

Frames are used for hierarchical packing layouts.

Events can be bound to widgets.

Key, KeyPress, KeyRelease, Button, ButtonPress, ButtonRelease, Enter, Leave, Motion,

Example button event bindings for .b

```
bind .b <Enter> {.b config -state active}
bind .b <Leave>  {.b config -state normal}
```

Event Patterns <modifier of event>

Modifiers: Control, Shift, Lock, Any, Meta, Button1..5, B1..5, Double, Triple

<Any-B1-Motion> any button1 motion

<KeyPress-a> pressing key a

Actions bound to event

{a proc_name and args, or set of tcl statements}

Script substitution enables event specific information to be returned to application. there are 30 substitutions ..

%x, %y	x and y coordinates of event
%W	path name of event window
%K	keysym from event (KeyPress...)
%A	8 bit iso character value of KeyPress
%%	substitute the % character

```
bind .aCanvas <ButtonPress> {  
    set xpoint %x  
    set ypoint %y }
```

update is a command that enables tcl scripts to escape from a procedure, process all waiting events, and then return to script.

tkwait command can be used to force an application to wait for a response from a window before continuing (modality)

```
tkwait window .dlg
```

Shell script like syntax *cmd arg arg arg ...*

Commands separated by newlines, semicolons
command continue next line with \

Commands return strings

Substitution Rules

Variables: use \$ to substitute value (disabled with \ and {...})

```
set a $b
```

Command results: use [...]

```
set a [expr $b + 2]
```

Compound statements {...}

```
if {$a < 0} {  
    puts "a is negative"  
    puts "\t a's value is $a" }
```

Tcl is:

String based: easy access from C
programs and data interchangeable

Built in commands implement:

Variables, associative arrays, lists, strings

Array, list and string functions

Math expression and functions (basic ones)

Branching and Looping structures

Procedures

Access to unix files i/o and commands (exec)

Control Structures:

if elseif case

for foreach

while

break and continue

eval

proc command defines procedure:

```
proc procName args { body of proc }
```

```
proc fac x {  
  if $x == 1 {return 1}  
  return [expr {  
    $x * [fac [expr $x - 1]]  
  } ]  
}
```

```
fac 4                      returns 24
```

Procedures look like build in commands.

Arguments can have defaults:

```
proc decr {x {y 1}  
  {expr $x - $y}
```

Variable number of arguments:

```
proc foo {a b args} {...} // where args is a list of arguments
```

exec command will fork child process

pipes and redirection

File commands

open, close, seek, tell, cd, pwd, gets, puts, read, flush, source, eof, file

Example variable and command substitutions:

<code>set b 66</code>	<code>66</code>
<code>set a \$b</code>	<code>66</code>
<code>set a "b is \$b"</code>	<code>b is 66</code>
<code>set a \$b.3</code>	<code>66.3</code>
<code>set a \${b}4</code>	<code>664</code>
<code>set a \$b4</code>	<code>no such variable</code>
<code>set a {\$b}</code>	<code>\$b</code>
<code>set b 8</code>	<code>8</code>
<code>set a [expr \$b + 2]</code>	<code>10</code>
<code>set a xy[expr \$b - 3]zz</code>	<code>xy5zz</code>
<code>set a "b-3 is [expr \$b - 3]</code>	<code>b-3 is 5</code>
<code>set a {[expr \$b + 2]}</code>	<code>[expr \$b + 2]</code>

