Tcl/Tk demo

Brian Toby

Running the shell

- Tcl shell = /usr/bin/tclsh (tclsh.exe)
- Tcl/Tk shell = /usr/bin/wish (wish.exe)
  - As usually distributed requires many extra files

Here: combined distribution as one file (starkit)

- Includes several “extra” packages

Starting “Tcl/Tk”

- Windows: ncnrpack-win.exe
- Linux: ncnrpack-linux
- OS X: ncnrpack-osx
  - Requires X11
  - Aqua version of Tcl/Tk does not support graphics demo

Can be run as

ncnrpack-xxx <script>	**runs commands in script**
ncnrpack-xxx	***opens console***

Tcl stuff

The set command

- `set var1 value`
- `set var2 $var1`
- `set var3 var1`

The set command

- `puts “writes a string: $var1”`
- `puts {w/o substitution: $var1}`

Embedded commands

- `set list [glob *]`

Math

- `set var [expr {pow(2,3)*10}]`

If-then-else

```
if {test1} {
  statement(s)
} else {
   statement(s)
}
```

```
if {var1 == “test”} {
  puts “test done”
  set var1 “”
} else {
  puts “not test”
}
```

looping

```
foreach var {1 2 3 a b c} {
  puts $var
}

for {set I 1} {$I <= 4} {incr I} {
   puts $I
}
```

Tk background

- Master window is named “.”
- Something inside master is a “child”
- Name of children of master will be
  `.<name>`
  – <name> ==> start w/lower case letter
  – .b or .bToby not .BT

Some Tk Commands: making widgets

- Command label
  `label  <child-name> -text ”label text”`
  – Returns <child-name>
- Command button
  `button <child-name> -text ”txt” \`
  `-command ”tcl command”`
  – Returns <child-name>
- N.B. creates child but does not display it

Displaying widgets

- For very quick demos, use pack:
  `label .l -text sample`
  `pack .l`
- Better, use grid
  `grid <child> -column <num> -row <num>`

  `label .l -text sample`
  `grid .l -column 1 -row 1`

Demo1

```
label  .label -text "Welcome to the Tcl/Tk demo"
grid .label -column 1 -row 1
button .b -text "Exit" -command "exit"
grid .b -column 1 -row 2
```

- Type the above into console *or*
- Windows: drag file demo1.txt onto **ncnrpack-win.exe** icon
- *or* Command line:
  `Toby$ ncnrpack-osx demo1.txt`
- *or* In console (windows may need `cd <dir>` to get to right place):
  `source demo1.txt`

Demo 1

Children with children

- Child windows
  `toplevel <name>`
- Frames (containers)
  `frame <name>`
- Child of child is named .parent.child
  ```
  toplevel .win
  button .win.b -text "child button"
  pack .win.b
  ```
- Destroy command deletes widgets & children
  `destroy .win` (deletes .win & .win.b)
  `destroy .` (same as exit)

Demo 2

```
label  .label -text "Welcome to the Tcl/Tk demo"
grid .label -column 1 -row 1
button .b -text "Exit" -command "exit"
grid .b -column 1 -row 2
toplevel .w
pack [label .w.l -text child]
pack [button .w.b -text kill -command "destroy .w"]
```

Creates a 2nd window. Clicking on kill closes the child window; clicking on exit closes both

Demo 2

Demo 3

```
foreach c {1 2 3} {
  foreach r {1 2 3} {
    grid [label .$c$r -text "$c-$r"] -column $c -row $r
  }
}
```

- Creates a table with grid

Demo 3

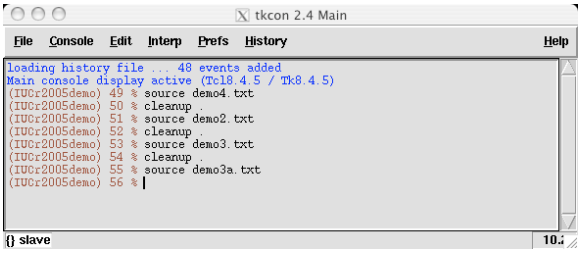
Demo 4
Define a new command

```
# define a routine to delete children of an item
proc cleanup {parent} {
  foreach item [winfo children $parent] {
     if {$item != ".tkcon"} {destroy $item}
  }
}
```

- Define a new command with:
  proc <name> <args> {script}
  - **<name> <arg(s)>** then executes command

Demo 4

- Nothing happens!
  - New command has been defined: cleanup
  - Use as `cleanup .` -- like this:

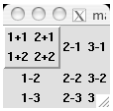


Demo 3a

```
foreach c {1 2 3} {
  foreach r {1 2 3} {
    grid [label .$c$r -text "$c-$r"] -column $c -row $r
  }
}
# replace upper left element with frame
destroy .11
grid [frame .f -relief raised -bd 2] -column 1 -row 1
# fill the frame
foreach c {1 2} {
  foreach r {1 2} {
    grid [label .f.$c$r -text "$c+$r"] -column $c -row $r
  }
}
```

- Creates a table inside a table with grid & frame

Demo 3a

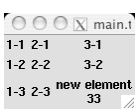


Demo 3b

```
foreach c {1 2 3} {
  foreach r {1 2 3} {
    grid [label .$c$r -text "$c-$r"] -column $c -row $r
  }
}
# change an element on the fly to show it can be done
.33 configure -text "new element\n33"
```

- Changes an existing label

Demo 3b

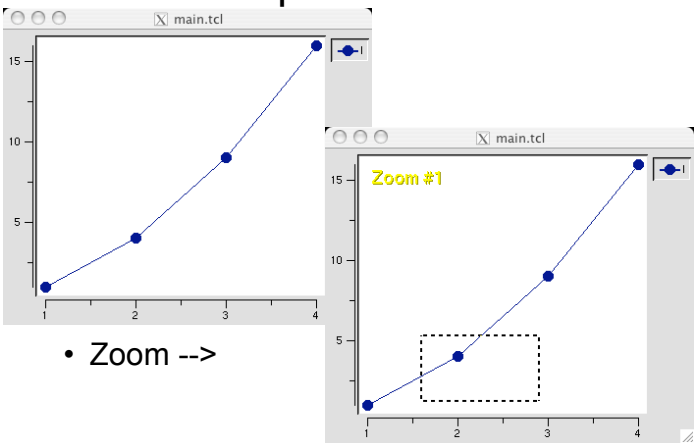


Graphics demo

```
package require BLT
pack [blt::graph .g]
.g element create l -xdata {1 2 3 4} -ydata {1 2 9 16}
# oops
.g element configure l -ydata {1 4 9 16}
# allow zooming
Blt_ZoomStack .g
```

- Shows how to use BLT for plotting

Graphics Demo



- Zoom -->

Running external programs

- Many ways -- this one is platform independent:

```
set fp [open x.txt w]
puts $fp "input"
close $fp
exec program < x.txt > x.out
set fp [open x.out r]
while {[gets $fp line] >= 0} {
  #do something with line
}
```

Running external programs

- Even better, have program write tcl code as output, then replace while with:

```
set lines [read $fp]
close $fp
eval $lines
```

- To suppress errors in output

```
catch {eval $lines}
```

Running interactive programs: Win9x, -ME

```
# this creates a DOS box to run a program in
 proc forknewterm {title command "wait 1" "scrollbar 1"} {
global env expgui
set pwd [file nativename [pwd]]

# check the .EXP path -- can DOS use it?
if {[string first // [pwd]] != -1} {
    MyMessageBox -parent . -title "Invalid Path" \
        -message {Error -- Use "Map network drive" to
access this directory with a letter (e.g. F:) GSAS can't
directly access a network drive} \
        -icon error -type ok -default ok \
        -helplink "expgui_Win_readme.html NetPath"
    return
}
```

```
if {[info command winutils::shell] == ""} {
    MyMessageBox -parent . -title "Setup error" \
     -message {Error -- "WINTILS not found. Can't do
anything!"} \
     -icon error -type darn -default darn \
     -helplink "expgui_Win_readme.html Winexec"
    return
}
# loop over multiple commands
foreach cmd $command {
    # replace the forward slashes with backward
    regsub -all / $cmd \\ cmd
     winutils::shell [file join $dir gsastcl.bat] $cmd
    }
}
}
```

Win-NT/-2000/-XP

```
proc forknewterm {title command} {
  global env expgui
      # loop over commands
      foreach cmd $command {
      # replace the forward slashes with backward
      regsub -all / $cmd \\ cmd
      exec $env(COMSPEC) /c \
            "start [file join $dir gsastcl.bat] $cmd"
      }
}
```

Unix/OS X

```
proc forknewterm {title command "wait 1" "scrollbar 1"} {
  global env expgui
  set termopts {}
  if $scrollbar {
      append termopts " -sb"
  } else {
      append termopts " +sb"
  }
  if {$wait} {
      set suffix {}
  } else {
      set suffix {&}
  }
  catch {eval exec xterm $termopts -title [list $title] \
       -e /bin/sh -c [list $command] $suffix} errmsg
  if $expgui(debug) {puts "xterm result = $errmsg"}
}
```

- How can you tell where you are?

```
if {$tcl_platform(platform) == "windows" && \
    $tcl_platform(os) == "Windows 95"} {
# win-95…
} elseif {$tcl_platform(platform) == "windows"} {
# win-XP…
} else {
# the rest
}
```

- See file expgui/gsascmds.tcl for more complete versions of the commands and for gsascmds.bat