Notes on tutorial #3 (for week 6)
(Feb 13, 2004)

26th February 2004

Q1: Write a small Tcl/Tk student-data-entry application which is a *form* for entering name, matriculation number and one of 10 (fixed) courses, selected using a suitable widget. Your application should have a submit button, which will append the information in a suitable format to a file[1], and then allow you to enter the next record and so on.

Answer: *Something like the following:*

CODE LISTING **tut3q1** Page 1/1

```
#!/usr/bin/wish -f

frame .name
label .name.label -text "Name:" -width 13 -anchor w
entry .name.entry -width 40 -textvariable userName
pack .name.label .name.entry -side left -expand 1
frame .matric
label .matric.label -text "Matric:" -width 13 -anchor w
entry .matric.entry -width 40 -textvariable matricID
pack .matric.label .matric.entry -side left -expand 1
pack .name .matric -side top
frame .left
frame .right
pack .left .right -side left -expand yes  -pady .5c -padx .5c

foreach i {cs2101 cs2102 cs2103 cs2104 cs2105} {
    radiobutton .left.b$i -text "$i" -variable course \
            -relief flat -value $i
    pack .left.b$i  -side top -pady 2 -anchor w
}
foreach i {cs2106 cs2107 cs2108 cs2109 cs2110} {
    radiobutton .right.b$i -text "$i" -variable course \
            -relief flat -value $i
    pack .right.b$i  -side top -pady 2 -anchor w
}

button .b -text "Enter data into File"  -command {processit}
pack .b -side bottom

proc processit {} {
   global userName
   global matricID
   global course
   set f [open "dbase.db" "a"]
   puts $f  "set info(\$count) {{$userName} {$matricID} {$course}}; incr count"
   close $f
   set userName {}
   set matricID {}
   set course {}
}
```

[1] Note that you might want to consider the use of an active file - Instead of putting lines like
"Name","Add1,Add2,Add3",cs3213
you might instead use lines like
set info($current) {{Name} {{Add1} {Add2} {Add3}} {cs3213}
which could then be sourced.

Q2: *Formally* specify your file format for question Q2 above.

Answer: *The BNF meta-description language may be used:*

```
<dbase>      ::= { <line> }
<line>       ::= <preamble> <name> <matric> <course> <postscript>
<preamble>   ::= "set info($count) {"
<name>       ::= "{" <string> "}"
<matric>     ::= "{" <string> "}"
<course>     ::= "{" <string> "}"
<postscript> ::= "; incr count\n"
<string>     ::= ....
```

An example of a file in this format is:

```
set info($count) {{dfdfsdf} {sfsdfsdf} {cs2107}}; incr count
set info($count) {{dfsdfsd} {sdfsdfsd} {cs2107}}; incr count
set info($count) {{sfsdf} {sdfsdfsd} {cs2107}}; incr count
```

Q3: Write another small Tcl/Tk application which will read in files from your student-data-entry application, allow you to (repeatedly) select a particular course, and then display the students on that particular course.

Answer: *Something like this:*

CODE LISTING **tut3q3** Page 1/1

```
#!/usr/bin/wish -f


frame .left
frame .right
pack .left .right -side left -expand yes  -pady .5c -padx .5c

foreach i {cs2101 cs2102 cs2103 cs2104 cs2105} {
    radiobutton .left.b$i -text "$i" -variable course \
            -relief flat -value $i
    pack .left.b$i  -side top -pady 2 -anchor w
}
foreach i {cs2106 cs2107 cs2108 cs2109 cs2110} {
    radiobutton .right.b$i -text "$i" -variable course \
            -relief flat -value $i
    pack .right.b$i  -side top -pady 2 -anchor w
}

button .b -text "Display the students"  -command {processit}
pack .b -side bottom
text .log -width 60 -height 10 -relief raised
pack .log
set count 0;
source dbase.db

proc processit {} {
   global course
   global info
   global count
   set i 0
   .log delete 0.0 end
   while {$i!=$count} {
      if {$course==[lindex $info($i) 2]} {
         puts stdout [lindex $info($i) 1]\ [lindex $info($i) 0]
         .log insert end [lindex $info($i) 1]\ [lindex $info($i) 0]
         .log insert end \n
      }
      set i [expr $i+1]
    }
}
```

Q4: Write a small Tcl/Tk program using a *canvas* that will put a new circle on the canvas when you click the right mouse button, and that each circle can be grabbed and dragged around using the left mouse button.

Answer: *The following code does it.*

CODE LISTING **tut3q4** Page 1/1

```
#!/usr/bin/wish -f

canvas .net -width 400 -height 200; pack .net
bind .net <ButtonPress-3> {makenode %x %y}
set nodes 0

proc makenode {x y} {
   global nodes
   incr nodes
   set x1 [expr "$x-10"]; set y1 [expr "$y-10"]
   set x2 [expr "$x+10"]; set y2 [expr "$y+10"]
   .net create oval $x1 $y1 $x2 $y2 -tag node$nodes -fill red
   .net bind node$nodes <ButtonPress-1> "beginmove %x %y"
   .net bind node$nodes <B1-Motion> "domove node$nodes %x %y"
}
proc beginmove {x y} {
        global oldx oldy
        set oldx $x; set oldy $y
}

proc domove {item x y} {
        global oldx oldy
        .net move $item [expr "$x - $oldx"] [expr "$y - $oldy"]
        set oldx $x; set oldy $y
}
```