Vahe Karamian
Project Dazzle Me
CS 408
Dr. Lazlo

Project Dazzle Me

My project is concerned with sml_tk library. sml_tk is a Standard ML (SML) package that provides a portable, typed and abstract interface to the user interface description and command language Tcl/Tk. Tcl/Tk consists of two components, which in principle are independent: the interface toolkit Tk offering a highly portable interface to various operating systems and their graphical display engine (such as Windows, MacOS and UNIX with its X Window System) and the command language Tcl, a weakly typed Lisp-like scripting language. The package sml_tk has been designed to combine the advantages of the Tk toolkit with those of SML.

Part of the sml_tk package is a toolkit library of reusable interface building components, including

? a collection of frequently used windows to display error messages, warnings or prompt for user input,
? a file browser ("sml_tk Explorer"),
? a generic drag&drop-canvas,
? a generic tree-navigation-canvas, and
? a generic user interface GenGUI.

The core of sml_tk implements the following entities:

? Windows (the basic building block of a graphical user interface)
? Widgets (graphical objects with a particular appearance and behaviour like Buttons, Menus or Textfields);
? Annotations (of a text, like hyperlinks, inserted pictures or buttons); and
? Canvas Items (items on a drawing area, a so-called canvas, such as lines, points, small icons, text).

To give a flavour of programming in sml_tk, we sketch a small example: suppose we want to construct a small window consisting of a text entry field labelled "name:" left from this field, and a quit button. After clicking on the text field, new text can be entered and line-edited. After finishing by hitting the Return key, the entered text should rename the window. When the quit button is pressed (i.e. clicked on with the mouse) the window should be closed, and the example terminates.

```
val mainID = newWinId()
val entID = newWidgetId()
```
The other two widgets can remain anonymous, since we need not refer to them explicitly.

```
(* Define the widgets: *)

val lab = Label{widId=newWidgetId(), packings=[Side Left]
            configs=[Text "name:"], bindings=[]}
val input = let fun endInput _ = changeTitle mainID (mkTitle(readTextAll entID))
        in Entry{widId=entID, packings=[], configs=[Width 20],
                     bindings=[BindEv(KeyPress "Return",endInput)]}
            end
val quit = let fun stop _ = closeWindow mainID
           in Button{widId=newWidgetId(), packings=[Side Bottom],
                     configs=[Text "Quit", Command stop],
                     bindings=[]}
           end
```

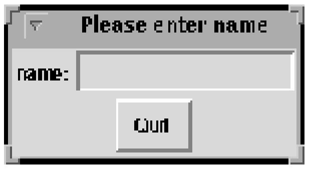
The layout is achieved as follows: the two widgets label and input should be placed side by side on top, with the widget quit at the bottom. To put two widgets side by side, we put them in one widget called a frame:

```
val topblock = Frame{widId=newWidgetId(), widgets=Pack [lab, input],
                          packings=[Side Top], configs=[], bindings=[]}
```

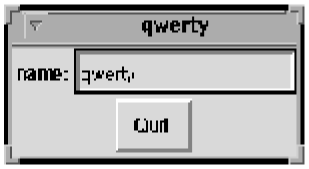
We are now ready to create the main window:

```
val enterwin = {winId = mainID,
                config = [WinTitle "Please enter name"],
                widgets = Pack [topblock, quit],
                bindings = [],
                init = noAction}
```

The following window appears on the screen:



We can now click on the entry field, and type in some text, finishing with <Return>, and we get:



"Pressing" the quit button (i.e. moving the cursor over it and pressing a mouse button) closes the window and terminates the startTcl command.

Reference:
http://www.informatik.uni-bremen.de/~cxl/sml_tk/#main