Mobile Extensions for SWT - High Level Design

Abstract

There is a need for a standardized UI API fit for mobile devices. The Standard Widget Toolkit (SWT) and the proposed eSWT subset provide a good general API and architecture for embedded devices, but lack mobile specific features. We don't want to bloat SWT with features which are not useful on all platforms/devices. Therefore, herein is proposed mobile specific extensions to the which, when combined with eSWT, meet the UI requirement.

Document Information

Author: Mark Rogalski
Email: Rogalski@us.ibm.com
Phone: 512-838-3512

Change History

0.9.0	7/12/04	First draft
0.9.1	8/27/04	Corrected bullets and indentation
0.9.2	9/16/04	Add widgets moved from eSWT
0.9.3	9/28/04	Incorporate feedback from work group

1. Introduction

The purpose of this document is to define mobile device oriented UI functional specifications which extend the functionality of the Standard Widget Toolkit (SWT). The specifications shall be abstract enough to allow for device dependent implementations such that the full features of a device platform may be exploited.

Refer to ***eSWT User Interface - High Level Design*** for more introduction text about the user interface architecture and about the context it is being applied to.

2. Scope

The API selection is primarily concerned with meeting the function requirements for mobile embedded devices. Many of the mobile device requirements are already fulfilled by eSWT, this document focuses on critical features currently not defined by eSWT. The API is a complementary part to eSWT and may utilize Core eSWT components or native widgets for its implementation.

3. Package Decomposition

This section defines the overall content of the Mobile Extensions for SWT at the widget level.

3.1 Widgets

The following widgets are provided:

- CaptionedControl: A control widget comprised of a text string label and a standard control. The whole widget becomes highlighted when the control portion gains focus. The preferred size of the whole widget is determined by the combined size of the label and the control. Apps can change the label's UI characteristics such as colors and fonts. Labels can have text, an icon, or both. The layout of the CaptionedControl content is implementation dependent.
    - Styles:
        - LEFT, RIGHT, TOP, BOTTOM
    - User interaction: Same action as the control
    - Traversal control: Arrow keys

- ListBox: a list widget that maps icon and text data to a platform specific layout and behavior based on an abstract style. The icon and text data is encapsulated in ListBoxItem objects which allow specification of multiple icons sizes and multiple lines of text. The ListBox layout style determines which icons and text are displayed and where they are displayed. For instance, a two column layout style may use the left column to display a small icon, and the right column to display text line one in a large font and text line two in a small font. The position of a selection mark of an item is implementation-dependent. This widget provides more specialized capabilities than ListView and is specifically formatted for small screens and key only navigation.

  Note: The style of all items in a ListBox must be consistent. That is, applications can not mix different styles, or even different fonts amongst items.
    - Styles:

- SINGLE: single selection mode
- MULTIPLE: multiple selection mode

§ Layout Styles: TBD based on function commonly provided by embedded device native UIs.

§ User interaction: Arrow Keys

§ Traversal control: Arrow keys or Tab key

Photo Album
Birthday Party June 18, 2004
Christmas Season December 26, 2003
Ski Trip January 14, 2004

Figure 1 – ListBox sample: Single column with two lines per entry; lines may be different fonts, colors

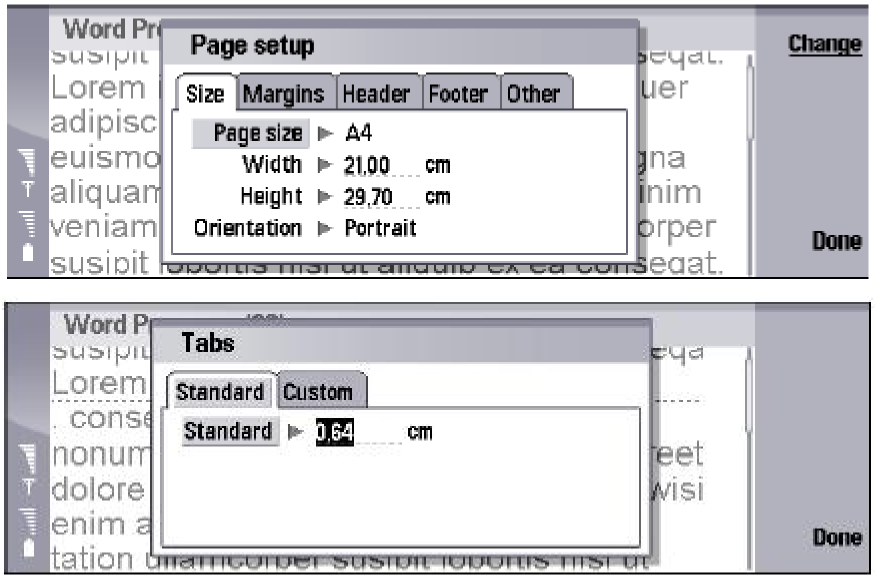
	Photo Album
(image)	**Birthday Party** June 18, 2004
(image)	**Christmas Season** December 26, 2003
(image)	**Ski Trip** January 14, 2004

Figure 2 – ListBox sample: Double column with image in column 1 and text in column 2

	Mom
Mobile (icon)	**1-716-675-0024**
Office (icon)	**1-512-838-3512**
Fax (icon)	**1-512-838-3703**

Figure 3 – ListBox sample: Double column with image and text in column 1 and text in column 2

- MultiPageDialog: a special dialog contains one or mutilple pages. Each page acts as a container for controls. The reason for defining such a Dialog, rather than using TabFolder, is because the choices, i.e. buttons in a dialog, are often positioned in a platform-dependent way. The MultiPage Dialog example (see Figure 4), for instance, uses softkeys to make selections, instead of using buttons in the dialog. The label texts used for selection softkeys or buttons must be customizable by applications. On small displays, the dialog may display full screen without title bar or border.



Page size ▶ A4
Width ▶ 21,00 cm
Height ▶ 29,70 cm
Orientation ▶ Portrait

Standard ▶ 0,64 cm

Figure 4 - MultiPage Dialog example

- RichTextBox: This widget can read and edit rich text. It allows entry of alphanumeric characters in multiple languages. It also enables text formatting, different font types, text colors, and some page layout information. It also allows the use of hyperlinks. This widget allows exchange of documents among different kinds of word processors. Example:

  - Styles:
    - TOOL_BAR: show text control pane
  - User interaction: Arrow Keys
  - Traversal control: Arrow keys or Tab key.

- SoftkeyMenu: menu which displays labels for configurable function softkeys. The location, look, and feel of this type of menu are device dependent. SoftkeyMenuItems in this type of menu are mapped on a best-fit basis to hardware buttons which have a configurable meaning. The location and number of such softkeys is device dependent. Thus, the SoftkeyMenu widget coordinates menu item to button assignment via a method which allows the application to remain device independent. The order of the menu items within the menu determines the priority in assigning items to limited softkeys. If the application provides a normal application menu widget as a constructor parameter for the SoftkeyMenu, SoftkeyMenuItems which are not mapped to softkeys are added to the provided menu.

  The application may also query the number of softkeys available to determine whether an alternate input paradigm may be desired.

  SoftkeyMenu supports sub-menus. In this case, SoftkeyMenuItem with CASCADE style is used as the parent constructor argument.

  - Styles: None
    - User interaction: None
    - Traversal control: Activated by device-specific key(s)

- SoftkeyMenuItem: represents a menu item which may be mapped to a hardware button.. SoftkeyMenuItems use SoftkeyMenu as their parental container. SOFTKEY_XXX menu item styles are hints which allow the SoftkeyMenu widget to better map menu items to softkeys which are commonly used for the same semantic purpose. A hint style is not required. Hints take precedence over any order of menu item within a menu. Different implementations may position the SoftkeyMenuItems according to their own style guidelines. Implementations may also restrict use of sub-menus. If the SoftkeyMenu does not allow sub-menus, SoftkeyMenuItems with CASCADE style are automatically appended to the application menu bar if provided. Otherwise, sub-menus will be ignored.

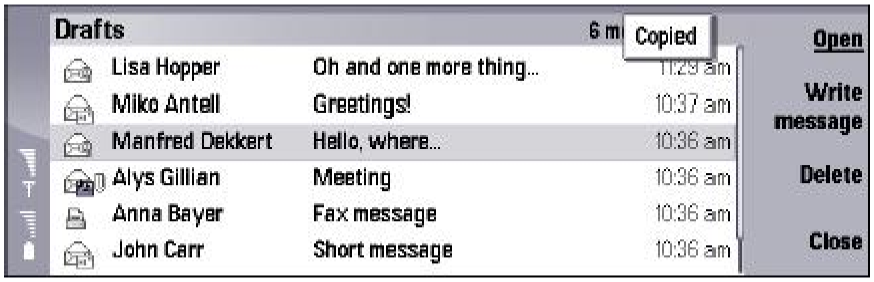
  - Styles:
    - CASCADE (for softkey sub-menus)
    - SOFTKEY_BACK
    - SOFTKEY _CANCEL
    - SOFTKEY _EXIT
    - SOFTKEY _HELP

  - SOFTKEY _OK
  - SOFTKEY _NEXT
  - SOFTKEY_OPTIONS
  - SOFTKEY _STOP
- User interaction: Arrow keys to navigate; Softkeys to select
- Traversal control: Arrow keys

3.2 Windows

- DynamicTrimShell: Top level window which is managed by the window manager. Used for application's main window. A special property of this widget over a normal Shell widget is that it supports a method to dynamically change Shell trim styles. It also supports a status area trim which can display text or small icons.
  - Styles**:** CLOSE, NORMAL_TRIM, REDUCED_TRIM, NO_TRIM, TITLE, NORMAL_STATUS, NARROW STATUS

- TaskTip: A non-modal, non-focusable message used to relay information about a task or the progress of some long-term processing. The difference between TaskTip and ToolTip is that TaskTip is a initiated by an application, whereas, a ToolTip displays automatically in response to some user input (such as focusing a widget.) TaskTip is not a *Control* that can be used inside a *Composite*. The look and location of the tip is implementation dependent. Applications can give placement hints via min, max and position values, but the final location of the tip is implementation dependent. Those values are ignored for style INDETERMINATE. If the platform does not explicitly support this feature, a TimedMessageBox is utilized. The lifetime of the tip is implementation dependent.
  - Styles:
    - INFORMATION: display a text message only.
    - INDETERMINATE: a progress icon is used to indicate some time-consuming tasks are going on. Implementations on different platforms may determine the way to display the icon and the text, for example, the text can be inside the progress icon, or can be completely invisible.



Lisa Hopper	Oh and one more thing...	11:23 am
Miko Antell	Greetings!	10:37 am
Manfred Dekkert	Hello, where...	10:36 am
Alys Gillian	Meeting	10:36 am
Anna Bayer	Fax message	10:36 am
John Carr	Short message	10:36 am

Figure 5. TaskTip example – INFORMATION style

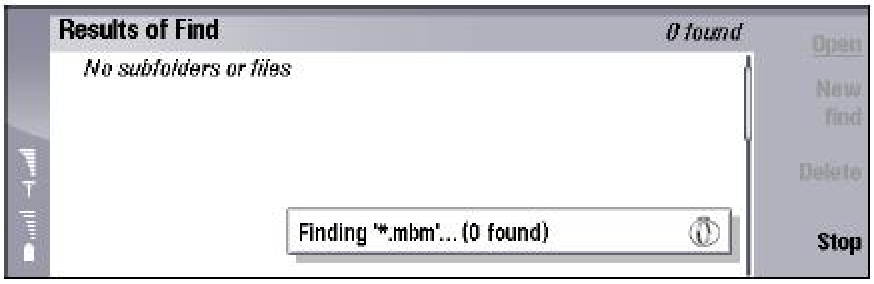


Figure 6. TaskTip example – INDETERMINATE style

- TimedMessageBox: a dialog used to inform the user of limited information using a standard style. Several buttons for user response can be specified. Caller provided button text is used to label softkeys where feasible or to display buttons within the dialog. The dialog returns which button was selected by the user. The caller may also set an arbitrary image to be used in place of one of the standard images. A TimedMessageBox is capable of closing itself automatically after a certain period of time. The timeout may be specified by the caller or will default to a platform-dependent timeout value. The caller may also specify an infinite timeout value.
  - Semantic styles:
    - ICON_NONE
    - ICON_ERROR
    - ICON_INFORMATION
    - ICON_QUESTION
    - ICON_WARNING
    - ICON_WORKING
  - User interaction: softkeys or select key
  - Traversal control: None.