Extending the Windows Desktop Interface With Connected Handheld Computers

Brad A. Myers, Robert C. Miller, Benjamin Bostwick, and Carl Evankovich

Human Computer Interaction Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
bam@cs.cmu.edu
http://www.cs.cmu.edu/~pebbles

ABSTRACT

Increasingly, people will be in situations where there are multiple communicating computing devices that have input / output capabilities. For example, people might carry their handheld computer, such as a Windows CE device or a Palm Pilot, into a room with a desktop or an embedded computer. The handheld computer can communicate with the PC using wired or wireless technologies, and then both computers can be used at the same time. We are investigating many ways in which the handheld computer can be used to *extend* the functions of existing and new applications. For example, the handheld's screen can be used as a customizable input and output device, to provide various controls for desktop applications. The handheld can contain scroll bars, buttons, virtual knobs and menus. It can also display the slide notes or a list of slide titles for a presentation, the list of current tasks and windows, and lists of links for web browsing. The user can tap on these lists on the handheld to control the PC. Information and control can flow fluidly among all the user's devices, so that there is an integrated environment.

1. INTRODUCTION

The age of *ubiquitous computing* [20] is at hand with computing devices of different shapes and sizes appearing in offices, homes, classrooms, and in people's pockets. Many environments contain embedded computers and data projectors, including offices, classrooms (e.g., [1]), meeting rooms, and even homes. One little-studied aspect of these environments is how personal, handheld computers will interoperate with the desktop computers. More and more people are carrying around programmable computers in the form of Personal Digital Assistants (PDAs) such as a Palm Pilot or Windows CE device, and even cell-phones and watches are becoming programmable. We are researching the question of what users can do with their handheld computers in such an environment.[1]

Some researchers have looked at using handheld devices in group settings to support collaborative work, usually with custom applications [1] [6] [13] [15] [17] [18] [19]. However, there has been little study of how portable devices can work in conjunction with the Windows desktop user interface, and with conventional Windows applications. Most of the research and development about handheld computers has focused on how they can be used to *replace* a regular computer for when one is not available. The conventional model for PDAs is that the data is "synchronized" with a PC once a day using the supplied cradle, and otherwise the PDA works independently. This will soon change. CMU has installed a Lucent Wavelan wireless network (which implements the IEEE 802.11 wireless standard) throughout the campus, in a project called "Wireless Andrew" [9]. Many Windows CE handheld computers can be connected to this wireless network using a PCMCIA card (although this is not an ideal solution since 802.11 has a high-demand for power and drains the batteries quickly). This year, the BlueTooth standard for small device wireless radio communication [8] will finally be available, and most PDAs, cell-phones, and other computerized small devices are expected to support it. Therefore, we expect

[1] This paper uses the terms PDAs and handheld computers interchangeably, since most of the ideas in this paper would equally apply to any of these devices. Our research has so far focused on using PDAs such as Palm Pilots and Windows CE devices. We will use "PC" to refer to the "regular" computer, which might be a desktop or laptop computer, or a computer embedded in a room with a large wall-mounted display.

that connecting the PCs and handhelds together will no longer be an occasional event for synchronization. Instead, the devices will frequently be in close, interactive communication. We are studying how the handheld computer's display, input mechanisms, and processor can be used to enhance the desktop computer's application when they are communicating. For example, the handheld can provide extra views of the data on the PC, and there can be buttons on the handheld that control the PC's applications. Thus, rather that trying to repeat the user's desktop and desktop applications on the handheld, we use the handheld to augment and enhance the existing PC. We feel that some of these applications may be sufficiently interesting to warrant buying a PDA (for example, the Slideshow Commander discussed below), whereas others would mainly provide small incremental benefits to people who already own a PDA. This paper presents an overview of the applications we have created for individual use of handhelds and PCs together.

2. OVERVIEW

In the "old days," computers had a variety of input switches and knobs. Today, computers are standardized with a keyboard and a mouse for input, and connecting other input devices can be difficult and expensive. Although today's computers have high-resolution screens and window managers, users can still need extra space to display information. For people who have already purchased a PDA, we want to exploit it to provide the benefits of having an extra input and output device. For example, Figure 1 shows the PDA being used at the same time as the mouse as an extra input and output device for the non-dominant hand. Note that we are not claiming that a PDA is an "ideal" extra input device, or that it is necessarily better than special-purpose extra devices that have been created (e.g., [2] [3]). Rather, we observe that people do not buy such devices, but they do have a PDA, so maybe we can get some of the benefits of the special-purpose devices without accruing the costs.

When the PDA is connected to the PC, we have found that the PDA can be used to *extend* the desktop user interface in various ways. It can serve as a customizable input device, with "soft" buttons, sliders, menus and other controls displayed on the screen. These can be made big enough to operate with a finger, even with the non-dominant hand (as in Figure 1). We have shown that using the non-dominant hand this way is effective for certain interaction techniques, such as scrolling [12]. The PDA can also be used as an output device to provide secondary views. This is useful when the entire PC screen is engaged and unavailable. For example, during a PowerPoint presentation, the PDA can display the notes of the current slide. Another use is to display information that should not be covered by other windows, such as the Windows' task bar, without sacrificing desktop screen real estate. Since the interfaces are on a handheld, they can be carried with the user, and even used with different PCs. Although some of the applications discussed below might work as a separate window on the main PC screen, the advantages of using a separate device are that the user can operate the PDA screen with a finger, it can be used at the same time as a mouse (supporting two-handed input), it can provide additional screen space when the PC's screen is full, and it provides an interface that users can take with them to use with different computers.

Figure 1. A PDA in its cradle on the left of the keyboard, and a person using both the PDA and the mouse simultaneously.

3. APPROACH

In order to support a variety of mobile devices, each running various programs, all attached to a PC running many different applications, we provide a central dispatcher we call PebblesPC (a later section presents the architecture in more detail). Our various PDA applications communicate with PebblesPC using a serial cable, infrared, Lucent's Wavelan radio network, Symbol's Spectrum24 radio network, or BlueTooth [8]. The appropriate PC-side program is run, which interprets the messages from the PDA. Many of the applications discussed below interface with conventional Windows applications, such as Microsoft Office. In other cases, we have created new, custom applications, for example to support multi-user drawing (these are discussed elsewhere [13]).

By interfacing with existing PC applications, we can explore how handheld computers can extend conventional Windows user interfaces, and how the handheld might integrate with the user's existing information environment. The goals of the Pebbles programs include providing mobile remote control of the PC application, moving displays and controls from the PC screen onto the handheld, to free up the PC screen for other tasks, and to enable two-handed operation of applications, for example so the left hand can be used for scrolling. To investigate and demonstrate these issues, we have created a wide range of applications. The following sections discuss some of the applications we have created.

3.1 Slideshow Commander

Often when giving presentations from a computer using slide show programs such as Microsoft's PowerPoint, the speaker will use notes. These are usually printed on paper, which makes them difficult to change. The speaker usually clicks on the mouse or keyboard to move forward and backward through the slides, which means that the speaker must be close to the PC. In a study of PowerPoint presentations [5], a number of problems were identified, including that the speaker often desires to walk away from the presentation computer to be closer to the audience (and some people just like to wander around while talking). It can also be awkward to point to and annotate the slides using a mouse. Further, people often had trouble when trying to navigate to the previous slide or to a particular slide.

The Slideshow Commander application (Figure 2) solves these problems by moving some of the display and control functions to the PDA. On the PC side, it works with Microsoft PowerPoint 97 or 2000 running under Windows. The PDA side works with any Windows CE machine or a Palm Pilot in black-and-white or color. Pressing the physical buttons on the Palm Pilot, using the scroll wheel on the side of a Windows CE palm-size device (Figure 2b), tapping the on-screen buttons with the stylus, or else giving a Graffiti or Jot gesture using a finger or the stylus, causes the slides to advance or go backwards. There are various panels on the PDA that support other tasks during a presentation. The Scribble panel allows the user to point and scribble on a thumbnail picture of the current slide on the PDA (see the top of Figure 2a and 2c), and have the same marks appear on the main screen. For the thumbnail picture to be legible on gray-scale PDAs, we have PowerPoint generate a "black-and-white" picture of the slide, which removes the background.

The Notes panel (Figure 2a on the bottom, and 2d) displays the notes associated with the current slide. The notes page is updated whenever the slide is changed, so it always displays the notes for the current slide. PowerPoint allows each slide to have associated notes that some people use—especially when the slides consist mostly of pictures. Other people put most of their text on the slides as bullet points. The Notes view supports both styles of presentations. The user can choose whether to display the text that appears on the slide itself, the text from the notes associated with that slide, or both.

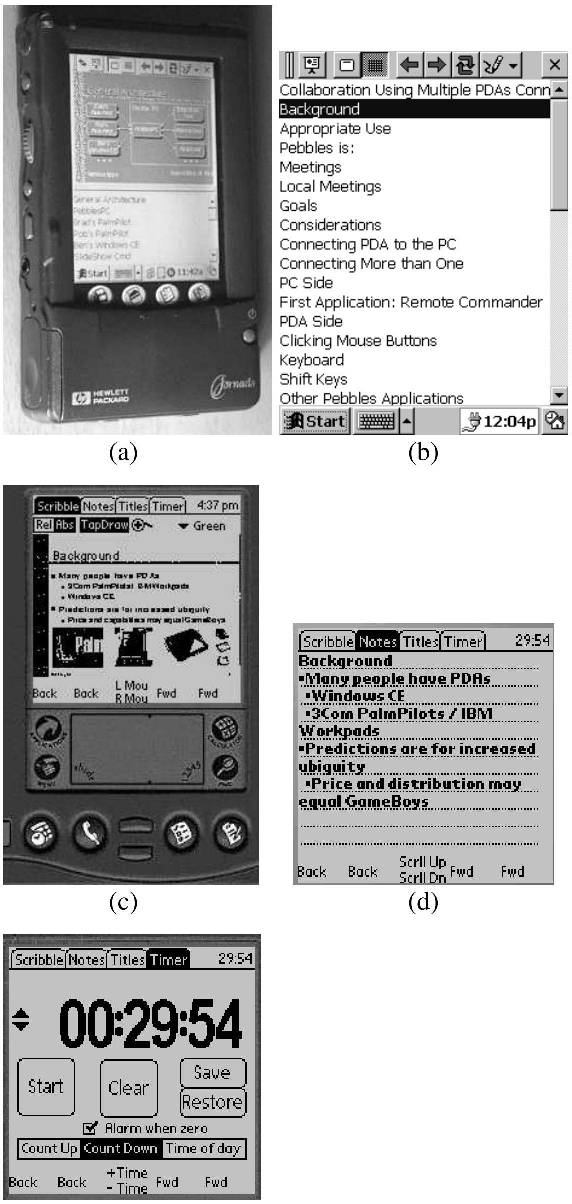
The Titles panel (see Figure 2b) displays the titles of all the slides in the current talk. The currently displayed slide is highlighted, and tapping on a slide name changes the presentation to that slide. This might be useful for people who have a large slide set and want to dynamically choose which slides to use for a given talk. Another use is at the end of the talk, during questions, to jump back to a specific slide under discussion. Finally, the Timer panel (Figure 2e) displays a timer, which can count up or down or display the current time of day. This is useful for timing the talk.

Of all the Pebbles applications, the Slideshow Commander is probably the most popular. We have received many positive comments on it, and the free version (which does not have the thumbnail pictures) is in wide scale use. The new version is being released commercially by Synergy Solutions (www.synsolutions.com).

3.2 Scrollers

Research [3] [12] [21] has shown that people can effectively scroll documents using their non-dominant hand instead of using conventional scroll bars. Recently, there has been a profusion of devices to help with scrolling, including the Microsoft IntelliMouse and the IBM ScrollPoint mouse. These mechanisms all use the dominant hand. Pebble's scrolling applications allow the PDA to be used as a scrolling device in either hand. Figure 3 shows some of the scrollers we have created. A user study demonstrated that these could match or beat scrolling using the mouse with conventional scroll bars, and was significantly faster than other scrolling mechanisms such as the scroll wheel built into mice [12]. As part of the same study, we measured how long it takes a person to move from the keyboard to acquire input devices, and found that the penalty for moving both hands (the left hand to the PDA and the right hand to the mouse) is only about 15% slower than moving just the right hand to the mouse and leaving the left hand on the keyboard. The movement time to return to the keyboard was only 13% slower. Thus, using the PDA does not provide a significant penalty, and can increase the speed of scrolling without requiring special-purpose devices.

Although these applications were designed to support scrolling, they can also be used as an extra, parallel input stream from the mouse. For example, in a positioning and resizing task [3], the scroll events can be mapped to change the object's size, which will support two-handed operation. In the future, we will be experimenting with other tasks that require more than two degrees of freedom, such as translation and rotation in 3-D worlds.

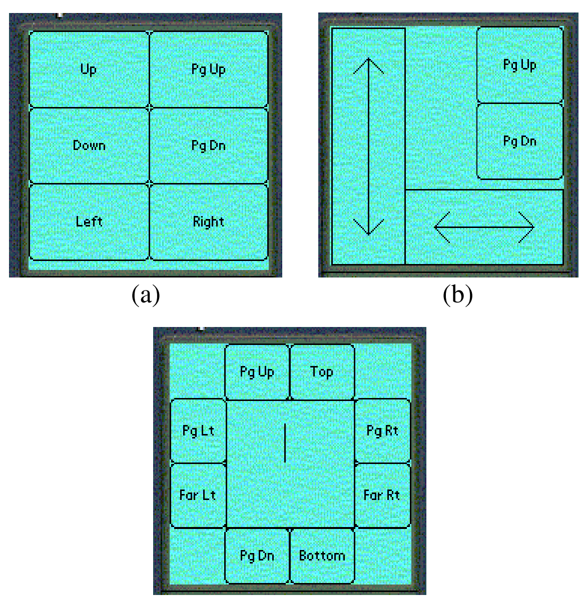


(a) (b) (c) (d) (e)

Figure 2. The Pebbles Slideshow Commander program. (a) The Scribble and Notes panes running in Windows CE on a HP Jornada. (b) The Title panel under Windows CE. (c) The full Palm Pilot, with the "Scribble" panel at the front. The "Notes" (d) and "Timer" (e) panels. Meanwhile, a PC is running PowerPoint and a PDA is in continuous two-way communication with the PC.

3.3 Task and Window Switcher

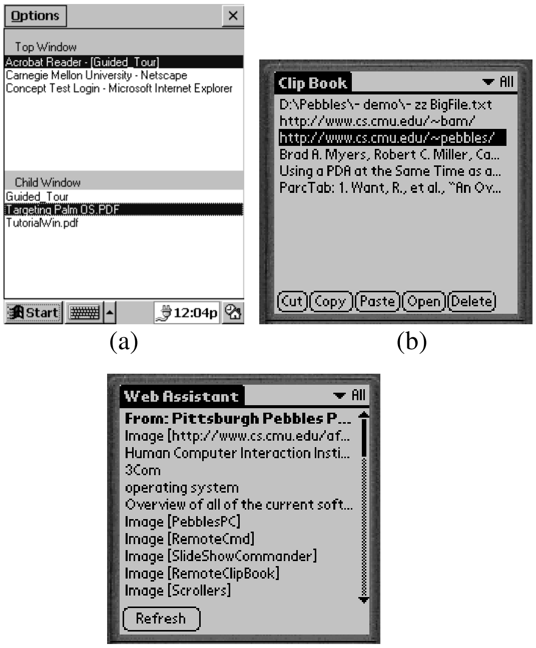
The Windows desktop provides a "Taskbar" at the bottom of the screen to switch among applications. Additionally, many Multi-Document Interface (MDI) applications have a "Windows" menu item to switch among windows in that application. The Switcher application (see Figure 4a) combines both capabilities into a single user interface on the PDA. This eliminates the confusion of which of these very different mechanisms to use to get to a desired window, and provides a consistent, quick, and always-available mechanism that does not use up any valuable real estate on the main screen. The user can tap on an item on the PDA to cause that window to come to the front of the PC. Another advantage is that we can optionally group windows by application, even for multiple instances of the application. Since Switcher provides various ways of organizing the application's windows, this can make it easier to find a window. For example, if there are multiple instances of Notepad running, all the documents from all of them can be combined into one alphabetical list, rather than by the order they were opened.



(a) (b) (c)

Figure 3. Three scrolling interfaces: (a) Buttons that auto-repeat to scroll up and down a line or a page, or left and right. (b) A slider, where dragging a finger or stylus in a rectangle drags the text the same amount. (c) A virtual rate-

controlled joystick, where pressing in the center and moving out scrolls the text in that direction at a rate proportional to the distance.



(a) (b) (c)

Figure 4. (a) Task Switcher displays on the PDA the top-level tasks and the windows for the selected task. (b) Remote Clipboard connects the PC's clipboard and the PDA's. Here is a list of items pasted from the PC, which can be copied back to the PC or to another PDA application. The "Open" button causes the selected file or URL to be opened on the PC. (c) Web Assistant displays the links on the current page. Clicking on a link causes the browser to go to that page. The list of links is only updated when the Refresh button is hit. For images that are links, the "alt" text is displayed in square brackets next to the word "Image" but if there is no "alt" text, then the URL is displayed instead.

3.4 Remote Clipboard

PDAs generally have some sort of synchronization mechanism to keep the information on the PDA consistent with a PC. However, the process is somewhat inconvenient and time consuming. For example, on the Palm Pilot you need to HotSync, which copies all applications' data that have changed. This is inappropriate if you want to just transfer a small amount of data, especially if it is to someone else's PC, to whom one does not want to give all of one's personal data. To provide a quicker way to copy small amounts of data between the PDA and the PC, we developed Remote Clipboard [10], which connects together the clipboards of the PC and the PDA. Whenever the user copies or cuts text on either machine, it can be pasted on the other. Using this familiar interaction technique, users can easily transfer information among many kinds of machines, devices and applications.

File names or URLs can also be pasted onto the PDA, and the Open button on the PDA will cause the PC to open the application associated with the file, or open the web page in the default browser. This allows information on the PC to be copied to the PDA either by value or by reference. If the user copies the information itself and pastes it to the PDA, this corresponds to passing the information "by value." If the user copies the filename or URL of the information, then it is passed "by reference." The Remote Clipboard PDA application (see Figure 4b) provides one place to store the data on the PDA, but the information can be pasted and copied from any PDA application, including the address book, scheduler, MemoPad, etc. In the future, we will investigate transferring richer information than just text. For example, it would be convenient to quickly sketch ideas on the PDA, and paste the pictures onto the PC to use as notes or as templates for more careful drawings.

3.5 Web Assistant

Research [4] and experience show that web browsing often takes on a "hub and spoke" style, where the user frequently returns to a main index page in order to find the next out-link to click on. Examples include the results of a search, and table of contents and maps of web sites and on-line documents. The Web Assistant application aids in these tasks by allowing links from the "hub" page to be copied to the PDA.

Figure 4c shows a view of the list of links on the PDA. Clicking on any link causes the web browser on the PC to switch to the specified page. Note that this does *not* automatically refill the PDA display with the links from the new page—it continues to show the original set of links so the user can easily move from link to link. Pressing the Refresh button refills the PDA page with the current page's links.

Many web pages are filled with irrelevant links. For example, a search results page can easily have more advertisement links than results. Therefore, the user can select a region of the browser text and only copy the links from that region onto the PDA. In the future, we expect to integrate more intelligent parsing technology [11] into the Web Assistant so the useful links can be selected and copied more automatically.

3.6 Shortcutter

The Shortcutter application combines many of the features of the previous utilities to allow users to create custom panels of “shortcut” buttons, sliders, knobs, and pads to control any PC application. The buttons can be big enough to hit with a finger, or tiny so that many will fit on a screen. The Shortcutter can provide customizable interfaces on the PDA even for applications that do not have a customization facility on the PC. Since these are on the PDA, you can take them with you and use them even on other people’s computers. In edit mode, users can draw panels and assign an action to each item in the panel. Switching to run mode, the items will perform their actions.

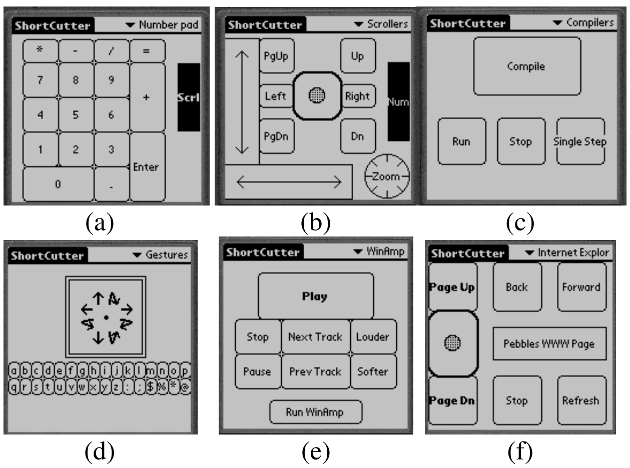


Figure 5. Panels created with Shortcutter. (a) A numeric keypad, (b) a collection of scrollers and a knob, (c) buttons for controlling any of a set of compilers, (d) a gesture pad and two rows of small buttons, (e) a controller for the WinAmp PC program, and (f) a panel for browsing in Internet Explorer.

We have defined several kinds of widgets that can be added to panels:

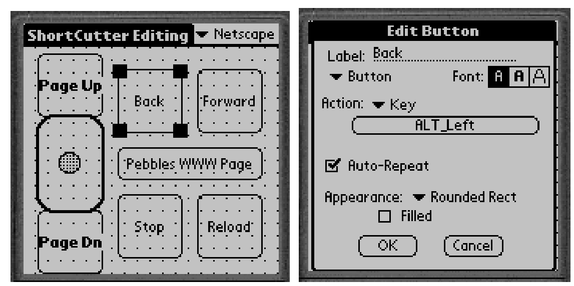
- **Buttons:** The user can assign a label, and use various shapes and patterns. Buttons can be specified to auto-repeat if the user presses and holds on them, or to only execute once per tap. Pressing a button invokes one of the actions listed below.
- **Scrolling Widget:** The vertical and horizontal sliders and the rate-control joystick scroller (see Figure 3) are also available in the Shortcutter (see Figure 5b which includes two sliders, a rate-control joystick in the center, and some scroll buttons). Scrolling is implemented on the PC by sending WM_HSCROLL and WM_VSCROLL messages to the foreground window.
- **Mouse Pad:** This supplies a rectangular area within which the user drags the stylus to perform mouse movements. It works like a trackpad on some laptops. Mouse events are inserted in the Windows event queue with the mouse_event API.
- **Gesture Pad:** This item allows users to give gestures such as those used in previous studies [2]. Figure 5d shows the gesture pad with icons for the nine supported gestures: tap (dot), strokes in four directions, and back-and-forth strokes in four directions. Each of these gestures can be assigned a different action. In the future, we might support more elaborate gestures or even user-trainable gestures.
- **Knobs:** For knob widgets, the user can specify one action to be generated when the user makes a clockwise gesture and another for counter-clockwise. These actions are repeatedly sent as the gesture continues, to give the effect of “turning” the knob. For example, a keyboard key that increments a value (such as the “+” key in Quicken, or a string like “+1=“ for a calculator) might be assigned to one direction and decrementing to the other direction. Since this is a general facility, any action can be used. Figure 5b shows a knob for zooming.

For many of these widgets, the user can choose any of the following actions:

- **Key:** Send any PC keyboard key to the PC as if the user had typed it, including function and other special keys and modifiers such as Control and Alt. This action is implemented by inserting keystrokes in the Windows event queue with the keybd_event API. This action makes it easy to create a panel like a numeric keypad (Figure 5a) that might be useful with laptops that do not have one. Anything that can be invoked using keyboard keys (menu items, etc.) can be easily assigned to a Shortcutter button.
- **String:** Send a string to the PC as if the user had typed it on the keyboard. This action is implemented by inserting each character of the string as if it were a keystroke. This might be useful for creating buttons that serve as abbreviations, or as input to the PC during a macro.
- **Open File/URL**: To tell the PC to open a file in its application, or to go to a specified page in a browser. This action is implemented by passing filename or URL to the Windows ShellExecute API.
- **Run Application:** Causes an application to be run, given an executable filename and optional command-line arguments. The user can specify whether to run a new instance of the application (using the WinExec API) or switch to the application if it is already running. The user can specify the executable filename of an application, possibly by finding the executable for the application, or a shortcut to it possibly in the Start Menu or on the desktop. An-

other, often more convenient way is to ask Shortcutter to determine the executable filename of a running application. On Windows platforms that support the Toolhelp32 API (Windows 95 and Windows 2000), Shortcutter finds the executable by calling CreateToolhelp32Snapshot to get information about the running process. On other platforms (specifically NT 4), Shortcutter injects itself in the address space of the running process with a window hook in order to call the GetModuleFilename API. Shortcutter uses this same technique to switch to a running application, by enumerating windows and searching for a window created by the given executable filename.

- **Scrolling:** Causes the foreground window to scroll horizontally or vertically, by lines or pages. This action is implemented by sending WM_HSCROLL and WM_VSCROLL messages.
- **Switch Panel:** The user can create a button that goes to a particular panel. This does not send anything to the PC. The black buttons on the right of Figures 5a and 5b switch between these two panels.
- **Mouse Button**: This is for sending mouse button events, such as left or right buttons down and double-clicking. Parameters include the modifier keys, if any (to support events like SHIFT-CLICK). Mouse buttons are simulated by calling the Windows mouse_event API.
- **Recorded Event:** This action is created by recording the next WM_COMMAND message sent by a menu item or toolbar button, using a global windows hook. When the action is invoked later, it replays the message to the current foreground window. This allows items on the PDA to perform actions that may not have a keyboard equivalent in the application, such as changing modes in a drawing program. Some menus and toolbars cannot be recorded in this fashion, however, either because they do not use WM_COMMAND messages, or because the messages must be sent to a window other than the main application window.
- **Macro:** This allows a sequence of the above actions to be associated with an item. Note that these macros can operate across multiple PC applications. Macros can invoke other macros, which supports procedural abstraction.
- **Wait for User:** This pops up a window with a button that waits for the user to tap OK. It is only really useful in macros. A string can be displayed on the PDA to instruct the user what is expected. A cancel button aborts the operation of the macro.
- **Application-Specific:** An item can have multiple actions associated with it, where each action is specific to a different PC application. Then, if the button is pressed, Shortcutter checks which PC application is in the foreground (using the techniques described above under Run Application), and chooses the appropriate action. This can provide a uniform, virtual interface to a set of applications that perform the same function but have disparate interfaces. An example use for this is that we use a variety of programming environments that unfortunately have different key assignments for actions and are not customizable. We created a panel of application-specific buttons that send the appropriate key for the current environment (see Figure 5c).



(a) (b)

Figure 6. (a) The editing mode of Shortcutter with the Back button selected. (b) Setting the properties of the Back button so when you click on the Back button at run-time, it sends to the PC the keystroke Alt-left-arrow.

The panels are constructed on the PDA by switching into "Edit Mode" (see Figure 6a). This supplies a conventional direct-manipulation editor, where buttons can be simply drawn, and their labels and properties assigned. The property sheet for a button is shown in Figure 6b. For the more complicated actions, other forms are used.

To make Shortcutter even more useful we allow the hardware (physical) buttons on the PDA to be mapped to any action. Panels and items can be stored on the PC and reused, and useful ones can be easily shared. Users can design buttons that are large enough to be hit with a finger (as in Figures 4 and 5), or very small so many items can fit on a panel and can be invoked using the stylus (as in the bottom of Figure 5d). The result is a very flexible and useful application with which users can create many interesting personalized shortcuts that might make their everyday use of a PC more effective.

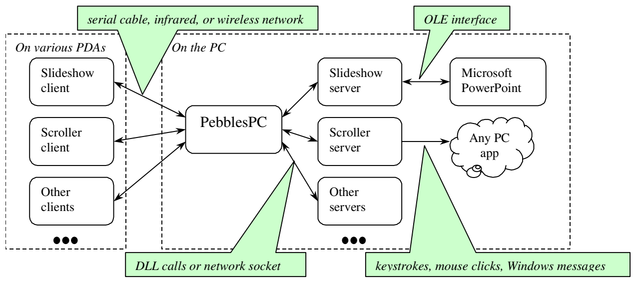


Figure 7. The Pebbles architecture.

4. ARCHITECTURE

The general architecture of Pebbles is shown in Figure 7. The main components are client programs running on (one or more) PDAs, server programs running on the PC, and PebblesPC, a PC program that mediates between clients and servers. These components communicate using a flexible message protocol we have designed.

4.1 Clients

Client programs run on handheld devices. Most of our applications run on both Palm and Windows CE devices. Multiple handhelds can be connected to the same PC, enabling not only multi-user applications but also the single-user, multi-device applications described in this paper. We generally assume that a handheld device can run only one program at a time. (This assumption is always true on the Palm, but on Windows CE a program can continue to run in the background, although you cannot see it.) Thus we make no allowances for multiplexing a serial connection between multiple simultaneously active client programs. The user is free to switch among client programs at any time, however.

4.2 Servers

Server programs run on the Windows PC. Our architecture supports two kinds of servers. The first kind uses plugins, which are dynamic link libraries (DLLs) loaded into PebblesPC's address space. Each plugin runs in its own thread to avoid blocking PebblesPC, and each plugin thread has a private Windows message queue. Windows messages are used to communicate between the plugin and PebblesPC. The second kind of server is a separate process, running either on the same PC or a remote host, and communicating with PebblesPC through a network socket.

Servers perform their operation in various ways, with various levels of application independence. For example, the Slideshow Commander server interacts directly with PowerPoint through OLE Automation. This kind of server clearly requires significant knowledge of the application being controlled. At the other extreme, the Scroller server simulates scrolling by inserting keystrokes and Windows messages into the standard Windows event stream. This kind of server need not know anything about the Windows applications that eventually receive the input events.

4.3 PebblesPC

PebblesPC acts as both a naming service and a message router. A server makes itself available to clients by connecting to PebblesPC and registering its name. For plugin servers, this happens automatically when PebblesPC loads the plugin's DLL, and the server's name is derived from the DLL filename. Clients connect to a server by first connecting to PebblesPC and requesting a server name (such as "Slideshow Commander"). If a server by that name is available, then PebblesPC makes a virtual connection between the client and the server, routing messages back and forth. PebblesPC allows clients and servers to connect through heterogeneous I/O interfaces, including serial ports, infrared, network sockets, and Windows message passing. PebblesPC handles the low-level details of each interface.

4.4 Message Protocol

Clients and servers communicate using an asynchronous message protocol designed to be simple, lightweight, and easy to implement. Low overhead is vital because many Pebbles applications use the handheld as a pointing device, which sends frequent update messages over a low-bandwidth channel (such as a serial port). Each message consists of a 1-byte command field (indicating the type of message), a 2-byte length field (extensible to 4 bytes if necessary), and a data field. Several command values are reserved for PebblesPC functions, such as registering client and server names, requesting a connection to a server, and closing a connection. Messages with other command values are passed unchanged between client and server, allowing the command to have an arbitrary meaning. For example, the Slideshow Commander application uses various command codes for requesting slides, changing the current slide, and sending slide titles, text, and thumbnail images. These messages can be sent directly through the serial cable or infrared, or else the messages can be sent over a TCP stream for network connections.

We have developed libraries for the Palm Pilot, Windows CE and Windows operating systems that implement the Pebbles protocol for various I/O interfaces, including serial, infrared, network sockets, and

Windows messages. This makes creating new Pebbles applications relatively easy.

5. RELATED WORK

The Xerox ParcTab [19] project investigated using small handheld devices at the same time as other computers. The ParcTab was used to investigate some aspects of remote cursors and informal voting about how well the speaker was doing. These applications are more closely related to the groupware Pebbles applications discussed in our previous paper [13]. One application allowed ParcTab to be scripted in Tcl/Tk to create applications that performed such functions as moving forward and backwards in a Tcl/Tk presentation manager, controlling the stylus to move the PCs cursor, remotely controlling a camera, creating postit notes on a PC, controlling Mosaic, and switching among X11 windows on Unix [15].

Rekimoto has studied a number of ways to use a PDA along with other computers. For example, Pick-And-Drop [16] and HyperDragging [18] are new ways to move information among different devices, which are related to (but different from) our cut-and-paste multi-machine model. The M-Pad system [17] supports multiple users collaborating with PDAs and a large whiteboard, which is similar to our multi-user application [13]. The large body of work on multi-modal user interfaces (e.g., [14]) typically combines direct manipulation with gestures and speech, but does not typically discuss the use of a hand-held device with a PC.

There have been many investigations of using both hands at the same time to control a computer. For example, an early study [3] showed that people could effectively scroll and change objects' size with the left hand while positioning with the right hand, and that many people operated both devices at the same time. Another study investigated how accurately gestures could be drawn with the non-dominant hand on a small touchpad mounted on top of a mouse [2]. Cross-application macros, such as provided by Shortcutter have been available in other applications. However, Shortcutter adds several new kinds of actions to the script, and the ability to keep the scripts on a PDA that can be easily transported.

Other groups are studying the use of Palm Pilots in various settings, where they are *not* connected to a PC. For example, NotePals synchronizes people's independent notes after a meeting [6], and Georgia Tech's "Classroom 2000" project [1] is studying the use of handheld computers in classrooms. For PDAs connected to a PC, our previous paper [13] discusses the use of multiple PDAs connected to a PC to support meetings. After the release of an earlier version our of Slideshow Commander for the presenter, a different group created a system with similar capabilities aimed at helping the audience members who have a PDA [7]. The current work shows that a PDA can be equally useful for a single person as an extension to the Windows user interface for desktop applications, and discusses the various ways we have extended the PC desktop user interface using the PDA.

6. STATUS AND FUTURE WORK

All of the applications discussed here are available for downloading off the World-Wide Web at http://www.cs.cmu.edu/~pebbles. These applications have been downloaded over 15,000 times. The newest version of Slideshow Commander is available commercially from Synergy Solutions (www.synsolutions.com). Third parties are also picking up on our architecture. For example, a commercial company, Iron Creek Software, used our architecture to build a Palm interface to the popular WinAmp PC program for playing MP3 and other digital music. Figure 5e shows a Shortcutter WinAmp controller, but the Iron Creek version also supports downloading and rearranging play-lists on the PDA.

Current work in our project is focusing on two areas: mixing private and public information, and classroom use. In many public meetings, it is useful for individuals to privately get on their handheld more details about publicly displayed information on a wall display. Alternatively, users might have additional details on their handheld that they want to combine with the public record. We are exploring how to make this information flow be fluid and natural. Another important area of work will be on transitioning some of these applications to the classroom. For example, the Slideshow Commander might broadcast to all the audience's computers the thumbnail picture and the notes of the current slide, to facilitate notetaking and understanding.

In general, we will continue to explore the many ways that PDAs can be used at the same time as desktop computers to enhance the user's effectiveness. Surprisingly, this is an area that has received very little study. As the communication mechanisms improve, and handheld computers become more capable and ubiquitous, it will be increasingly important to consider how the users' mobile devices can interoperate with the stationary computers in the environment. The research described here presents a number of ways in which the conventional Windows user interface can be extended using mobile devices.

ACKNOWLEDGMENTS

For help with this paper, we would like to thank Andrew Faulring, Brad Vander Zanden, Karen Cross, Rich McDaniel, Bernita Myers and the referees.

This research is supported by grants from DARPA, Microsoft, Symbol, HP, IBM and Palm. This research was performed in part in connection with Contract number DAAD17-99-C-0061 with the U.S. Army Research Laboratory. The views and conclusions contained in this document are those of the authors and should not be interpreted as presenting the official policies or position, either expressed or implied, of the U.S. Army Research Laboratory or the U.S. Government unless so designated by other authorized documents. Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

REFERENCES

1. Abowd, G.D., *et al.* "Investigating the capture, integration and access problem of ubiquitous computing in an educational setting," in *Proceedings SIGCHI'98: Human Factors in Computing Systems.* 1998. Los Angeles, CA: pp. 440-447.

2. Balakrishnan, R. and Patel, P. "The PadMouse: Facilitating Selection and Spatial Positioning for the Non-Dominant Hand," in *Proceedings SIGCHI'98: Human Factors in Computing Systems.* 1998. Los Angeles, CA: pp. 9-16.

3. Buxton, W. and Myers, B. "A Study in Two-Handed Input," in *Proceedings SIGCHI'86: Human Factors in Computing Systems.* 1986. Boston, MA: pp. 321-326.

4. Card, S.K., Robertson, G.G., and York, W. "The WebBook and the Web-Forager: An Information Workspace for the World-Wide Web," in *Proceedings CHI'96: Human Factors in Computing Systems.* 1996. Vancouver, BC, Canada: pp. 111-117.

5. Cross, K. and Warmack, A. "Contextual Inquiry: Quantification and Use in Videotaped Analysis," in *Adjunct Proceedings CHI'2000: Human Factors in Computing Systems.* 2000. The Hague, The Netherlands: pp. To appear.

6. Davis, R.C., *et al.* "NotePals: Lightweight Note Sharing by the Group, for the Group," in *Proceedings, CHI'99: Human Factors in Computing Systems.* 1999. Pittsburgh, PA: ACM. pp. 338-345.

7. Dey, A.K., *et al.* "The Conference Assistant: Combining Context-Awareness with Wearable Computing," in *Proceedings of the 3rd International Symposium on Wearable Computers.* 1999. San Francisco, CA: pp. To Appear.

8. Haartsen, J., *et al.*, "Bluetooth: Vision, Goals, and Architecture." *ACM Mobile Computing and Communications Review*, 1998. **2**(4): pp. 38-45. Oct. www.bluetooth.com.

9. Hills, A., "Wireless Andrew." *IEEE Spectrum*, 1999. **36**(6)June.

10. Miller, R.C. and Myers, B. "Synchronizing Clipboards of Multiple Computers," in *Proceedings UIST'99: ACM SIGGRAPH Symposium on User Interface Software and Technology.* 1999. Asheville, NC: pp. 65-66.

11. Miller, R.C. and Myers, B.A. "Lightweight Structured Text Processing," in *Usenix Annual Technical Conference.* 1999. Monterey, California: pp. 131-144.

12. Myers, B.A., Lie, K.P.L., and Yang, B.-C.J. "Two-Handed Input Using a PDA And a Mouse," in *Proceedings CHI'2000: Human Factors in Computing Systems.* 2000. The Hague, The Netherlands: pp. To Appear.

13. Myers, B.A., Stiel, H., and Gargiulo, R. "Collaboration Using Multiple PDAs Connected to a PC," in *Proceedings CSCW'98: ACM Conference on Computer-Supported Cooperative Work.* 1998. Seattle, WA: pp. 285-294.

14. Oviatt, S. and Cohen, P., "Multimodal Interfaces That Process What Comes Naturally." *Communications of the ACM*, 2000. **43**(3): pp. 45-53. March.

15. Petersen, K. "Tcl/Tk for a Digital Personal Assistant," in *Also Xerox PARC CSL Technical Report: CSL-94-16, Dec. 1994.: Proc. of the USENIX Symp. on Very High Level Languages (VHLL).* 1994. Santa Fe, New Mexico: pp. 41--56.

16. Rekimoto, J. "Pick-and-Drop: A Direct Manipulation Technique for Multiple Computer Environments," in *Proceedings UIST'97: ACM SIGGRAPH Symposium on User Interface Software and Technology.* 1997. Banff, Alberta, Canada: pp. 31-39.

17. Rekimoto, J. "A Multiple Device Approach for Supporting Whiteboard-based Interactions," in *Proceedings SIGCHI'98: Human Factors in Computing Systems.* 1998. Los Angeles, CA: pp. 344-351.

18. Rekimoto, J. and Saitoh, M. "Augmented Surfaces: A Spatially Continuous Work Space for Hybrid Computing Environments," in *Proceedings SIGCHI'99: Human Factors in Computing Systems.* 1999. Pittsburgh, PA: pp. 378-385.

19. Want, R., *et al.*, "An Overview of the ParcTab Ubiquitous Computing Experiment." *IEEE Personal Communications*, 1995. : pp. 28-43. December. Also appears as Xerox PARC Technical Report CSL-95-1, March, 1995.

20. Weiser, M., "Some Computer Science Issues in Ubiquitous Computing." *CACM*, 1993. **36**(7): pp. 74-83. July.

21. Zhai, S., Smith, B.A., and Selker, T. "Improving Browsing Performance: A Study of Four Input Devices for Scrolling and Pointing," in *Proceedings of Interact97: The Sixth IFIP Conference on Human-Computer Interaction.* 1997. Sydney, Australia: pp. 286-292.