Collaboration Using Multiple PDAs Connected to a PC

Brad A. Myers **Herb Stiel** **Robert Gargiulo**

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

The Pebbles project is creating applications to connect multiple Personal Digital Assistants (PDAs) to a main computer such as a PC. We are using 3Com PalmPilots because they are starting to be ubiquitous. We created the "Remote Commander" application to allow users to take turns sending input from their PalmPilots to the PC as if they were using the PC's mouse and keyboard. "PebblesDraw" is a shared whiteboard application we built that allows all of the users to send input simultaneously while sharing the same PC display. We are investigating the use of these applications in various contexts, such as co-located meetings.

Keywords: Personal Digital Assistants (PDAs), PalmPilot, Single Display Groupware, Pebbles, Amulet.

INTRODUCTION

There are certain kinds of meetings, including design reviews, brainstorming sessions, and organization meetings, where a PC is used to display slides or a current plan, and the people in attendance provide input. Many conference and presentation rooms today have built-in facilities for projecting a PC onto a large screen, and various inexpensive technologies are available for rooms that do not. When a PC is used as part of the presentation, often different people will want to take turns controlling the mouse and keyboard. For example, they might want to try out the system under consideration, to investigate different options, or to add their annotations. With standard setups, they will have to awkwardly swap places with the person at the PC.

We observed that at most meetings and talks, attendees do not bring their laptops, probably because they are awkward and slow to set up, the batteries may not last long enough, and there is a social stigma against typing during meetings. Today, however, many people are taking notes on their PalmPilots. A *PalmPilot* is a small "Personal Digital Assistant (PDA)" from 3Com with a 3¼ inch diagonal LCD display screen which is touch sensitive, and a small input area for printing characters using a special alphabet (see Figure 1). PalmPilots have the advantages that they are small, they turn on instantly, the batteries last for weeks, and notes are taken by writing silently with a stylus. Since people have the PalmPilots in their hands anyway, we developed a set of applications to explore how these PalmPilots could be used to allow everyone to provide mouse and keyboard input to the PC without leaving their seats. We call this the "Remote Commander."

Once the PDAs are connected to the PC, many other intriguing applications become available. In addition to the Remote Commander, we also created a shared drawing program, named PebblesDraw, that allows everyone to draw simultaneously. In creating PebblesDraw, we had to solve a number of interesting problems with the standard widgets such as selection handles, menubars and palettes. Since this is a *Single Display Groupware (SDG)* application, all users are sharing the same screen and therefore the same widgets. Palettes, such as those to show the current drawing mode or the current color, normally show the current mode by highlighting one of the items in the palette. This no longer works if there are multiple people with *different* modes. Furthermore, the conventional menubars at the top of the window or screen are difficult to use in multi-user situations for technical reasons, and also because they may pop up on top of a different user's activities. The conventional way to identify different users is by assigning each a different color, but in a drawing program, each user might want to select what color is being used to draw, causing confusion between the color identifying the user, and the color with which the user will draw. We therefore developed a new set of interaction techniques to more effectively support Single Display Groupware.

This research is being performed as part of the *Pebbles* project. Pebbles stands for: **P**almPilots for **E**ntry of **B**oth **B**ytes and **L**ocations from **E**xternal **S**ources.[1] This paper summarizes the PalmPilot's features, and then describes the first two applications: Remote Commander, which allows the PalmPilots to control the PC, and PebblesDraw, which allows multiple people to draw at the same time.

[1] The "bytes" refers to the characters and the "locations" refers to mouse points that are sent from the PalmPilot down to the PC.

Figure 1. The 3Com PalmPilot running the Pebbles Remote Commander application.

RELATED WORK

Most previous CSCW approaches for multiple users in the same room use expensive and special-purpose hardware or specially constructed meeting rooms. For example, the Xerox CoLab [21] and the Univ. of Arizona's electronic meeting room [15] had computers for each person built into special tables. Another previous approach was the Xerox Liveboard [8], which originally supported multiple cursors operating at the same time, but when produced commercially, only supported one person with one cursor at a time. Liveboards proved to be too expensive to be widely used. The Tivoli system [17] supports up to three people using pens simultaneously on the LiveBoard. We believe that much of the software used with these previous technologies can be reproduced using a set of PalmPilots and a single PC, which will be much cheaper and easier to configure.

The Xerox ParcTab [24] project investigated using small hand-held devices continuously connected through infrared (IR) communication. Most of the PARC building was wired with special IR transceivers to make the ParcTabs work. The ParcTab screen was 128x64 pixels and had a few hard-wired buttons and a touch-sensitive screen, which supported the Unistrokes [9] alphabet, which is similar to (and predates) PalmPilot's Graffiti. The ParcTab was used to investigate some aspects of remote cursors, and informal voting about how well the speaker was doing, but due to problems with the IR, they found that multiple people simultaneously using the devices in the same room did not work well.

The PDA-ITV project [20] explored the use of a Newton to control Interactive TV, but used a single real-estate application for shopping for houses, and only dealt with one Newton at a time.

MMM [4] (Multi-Device, Multi-User, Multi-Editor) was one of the first Single Display Groupware (SDG) environments to explore multiple mice on a single display. MMM handled up to three mice. MMM had a single tool palette and color palette, and a separate "home area" to show each user's name, current color, and drawing mode. MMM only supported editing of text and rectangles, and used color to distinguish between the users.

The term "Single Display Groupware" was coined by Stewart et. al. [23]. Stewart's KidPad [23] is a SDG environment for kids, where multiple mice are connected to a Unix computer. It uses "local tools" [2] on Pad++, where each tool does exactly one thing (for example, there is a crayon tool). This model is easier for kids than conventional palettes, but it does not have any global state (so, for example, there is no concept of a selection). Therefore, KidPad does not face many of the issues addressed in our Pebbles work. Background studies showed that children often argue and fight when trying to share a single mouse [22], but when using separate mice, cooperated more effectively. Another SDG effort is the COLT toolkit [5], which explores some limited kinds of user interactions and showed that children will stay more focused on their tasks when each child has their own mouse and they simultaneously manipulate the same object together.

Other groups are studying the use of PalmPilots in various settings, where they are *not* connected to a PC. For example, NotePals synchronizes people's independent notes after a meeting [7], and Georgia Tech's "Classroom 2000" project [1] is studying the use of PalmPilots in classrooms.

THE PALMPILOT

The PalmPilot (see http://palmpilot.3com.com/) is a small inexpensive hand-held "Personal Digital Assistant" (see Figure 1) formerly sold by USRobotics (which was bought by 3Com) and also now sold by IBM as the "WorkPad" (see http://www.pc.ibm.com/us/workpad/). The PalmPilot sold over one million units in its first 18 months. They are starting to be ubiquitous, and many people in our academic community are using them to take notes in meetings. Although our research is using PalmPilots because they are easy to program and popular, the research would equally apply to any PDA.

One of the most important reasons the PalmPilot is so popular is that it connects very easily to a PC (and also to a Macintosh or Unix workstation) for synchronization and downloading. Each PalmPilot is shipped with a cradle and wire that connects to a computer's standard serial port. Software supplied with the PalmPilot will synchronize the data with the PC. It is also easy to load new applications into the PalmPilot. Pebbles takes advantage of this easy connection to a PC. PalmPilot are also small enough to easily fit into a pocket, and relatively inexpensive.

The main display of the PalmPilot is a 160x160 pixel LCD panel, with 4 levels of gray (but most applications treat it as monochrome). The screen is touch sensitive, so text and graphics can be selected and drawn. A small stylus which fits into the PalmPilot case is usually used for this, but a finger can also be used for pointing and gestures. Textual characters are entered in the area at the bottom using special gestures called "Graffiti," which is a stylized alphabet that is designed to be easier for the PalmPilot to recognize accurately. Almost all letters and symbols are entered with a single stroke, which approximates the upper or lower case way the letter is drawn. Most people seem to be able to learn the gestures in about 15 minutes. There is also an on-screen keyboard. In Pebbles, we are taking advantage of the fact that people have *already* learned these gestures, and are comfortable with the operation of the PalmPilot since they are already using it for many daily activities.

REMOTE COMMANDER

The first application from our Pebbles project is the *Remote Commander*. This allows strokes on the main display area of the PalmPilot to control the PC's mouse cursor, and for Graffiti input to emulate the PC's keyboard input. The important point is that this works with *all existing* PC applications, without requiring any modifications to the applications themselves. For example, multiple people can use PowerPoint, Word, Excel, Illustrator, etc., from their PalmPilots.

We believe in distributing the results of our research, to help collect useful feedback and aid in technology transfer. The first version of the Remote Commander software was released for general use on February 17 (see http://www.cs.cmu.edu/~pebbles) and was downloaded about 3000 times in the first six weeks. The current version works with Windows 95 or NT, and current versions of the PalmPilot.

In designing this application, we wanted to make sure that it would fit in with the PalmPilot style. Therefore, it needed to be natural and non-intrusive. It should be easy to switch between using the Remote Commander and using other PalmPilot applications, and it should take advantage of the Graffiti gestures that the users had already learned. We also had to deal with the PalmPilot's limitations: a small, grayscale screen, and a fairly slow processor. Furthermore, the users' focus should normally be directed to the PC's screen, so it should be possible to operate the Remote Commander without looking at the PalmPilot, just as the mouse can be used without looking.

In the current version of Remote Commander, we rely on social protocols to control whose turn it is, which is not difficult since everyone is in the same room. Everyone's inputs are mixed together. We decided that a more formal "floor control" mechanism would obstruct the fluid sharing and collaboration that is typical in meetings.

Example Uses

When designing the Remote Commander, we had a number of uses in mind. The first is as discussed above: for multiple people in a meeting to take turns controlling the mouse and keyboard. Remote Commander is designed to support meetings where everyone is in the same room. (Remote participants might can use other CSCW applications, such as Microsoft NetMeeting.) Remote Commander might be used so different people can have a turn controlling an application, to support multiple annotations on a work product, or so everyone can enter their personal data into a shared document. Having a single, shared visual display is important for coordination and collaboration [16], and people will take turns or all work together on these displays.

A second use for Remote Commander is to support small, informal meetings. These arise when two or three people are discussing something that is running on a PC, such as a demo situation, or for small design and debugging discussions.

Another use for the Remote Commander might be any time that a single user would rather have a stylus for input instead of a mouse, for example in situations when freehand drawing or writing is desired. The PalmPilot digitizer can even read the stylus through a sheet of paper, so Remote Commander can support the tracing of small drawings. Of course, if tracing was frequently needed, the user would probably purchase a "real" tablet, but again we are taking advantage of technology the user might already have for a task they might do occasionally.

PalmPilot Module

There are two modules of the Remote Commander — one which runs on the PalmPilot and the other runs on the PC, and these two are connected using serial cables.

Moving the Cursor

Figure 1 shows the Remote Commander application running on the PalmPilot. The large blank upper area is used to represent the mouse. Moving the stylus (or your finger) across the screen of the PalmPilot moves the cursor on the PC. Through experimentation, we arrived at a system which provides good control.

The current design uses relative coordinates. This is analogous to the small touchpad on some laptops, like the Macintosh PowerBook. Movement across the PalmPilot screen is mapped to an incremental movement across the PC's screen, so the actual position where you put down the stylus is not important; just how far it is moved from the initial position.

Originally, we tried directly mapping the 160x160 pixels of the PalmPilot to the 1024x768 pixels (for example) of the PC, using absolute coordinates. Thus, tapping in the upper left of the PalmPilot screen would move the cursor to the upper left of the PC screen. However, this did not work at all. Since each pixel on the PalmPilot represented 6 pixels on the PC, and since characters are often less than 6 pixels wide, this made it impossible to select individual characters. Furthermore, the positions reported by the PalmPilot digitizer are very jittery, varying by 1 or 2 pixels in all directions when the stylus is kept still, so the cursor jumped all over the PC's screen.

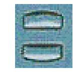
The jittery coordinates coming from the PalmPilot were still a problem when we switched to relative coordinates, making it difficult to position the cursor accurately. Looking at other applications running locally on the PalmPilot, such as various sketching programs, we noticed that the jittery values were not limited to our program, but seem to be a property of the device. Therefore, we added filtering of the positions. After experimenting with various algorithms and parameters, the best behavior resulted from collecting the last 7 points returned by the PalmPilot, and returning the average as the current point. This removes most of the jitter without adding too much lag. This filtering starts over each time the stylus comes in contact with the PalmPilot, and the array of the last 7 points is initialized with the initial point. This allows points to be provided immediately when the stylus comes in contact with the surface. We also added extra acceleration to the PalmPilot output so one swift movement across the PalmPilot screen would move entirely across the PC's screen.

Mouse Buttons

Another issue concerns emulating the mouse buttons. As with a touchpad, the stylus of the PalmPilot must be in contact with the surface in order to move the cursor, so a different signal is needed for the buttons. This is not a problem for applications which run on the PalmPilot itself, since the screen displays the objects that the user wants to point at, and there is no need for a separate cursor.

We provide three different ways to signal pressing a mouse button. First, tapping on the blank area of the PalmPilot screen with the stylus causes the Remote Commander program to send a click event (down press followed by a release) of the left mouse button to the PC. A double-click can easily be sent to the PC by double-tapping. A tap is determined by a press and release at the same place. Since Windows (and now Macintosh system 8) allows menus to stay up when clicked on, this is an easy way to select items in menus and push on-screen buttons. User experience with this tapping is mixed. It appears to take practice to be able to tap without moving, and novices seem to draw little lines when they want to tap. On the other hand, others seem to tap by accident, possibly due to some bounce of the stylus on the surface, or when trying to draw small items. We will continue to investigate adjusting the parameters to see if we can eliminate these problems.

In many cases, it is also necessary to *drag* with the button held down. "Dragging" is when the mouse button is pressed somewhere, then the mouse is moved, and the button is released at a different point. To support dragging, we first added a "Drag" button to the Remote Commander screen (see Figure 1). Pressing on this Drag button starts the dragging by sending a left down event. The user can then move the stylus. Pressing on the Drag button again will end the drag by sending the button-up event. Experience suggests that the Drag button is not a particularly good idea, because people frequently make *mode errors*, where they forget whether the mouse button is pressed or not, and end up drawing a line when they planned to just move the cursor, and vice versa.



We added a third way to signal pressing of the mouse buttons, which proved to be the most successful way to drag. This uses the up and down physical buttons in the center bottom of the PalmPilot (shown in Figure 1 and at right). We use the up button for the left mouse button, and the down button for the right, which is similar to the way many laptops handle with the mouse buttons. It is easy to hold the PalmPilot so that the thumb of the non-dominate hand (e.g. left hand) can work the buttons while the dominate hand (e.g. right hand) draws with the stylus. Experience shows that people do not have much trouble coordinating their two hands for this task. Since this method works well, we did not add any other ways to signal the right mouse button.

Emulating the Keyboard

Writing characters in the Graffiti area of the PalmPilot will send those characters to the PC as if they were typed at the keyboard. Capital letters and punctuation marks work in the normal way for Graffiti. The PalmPilot comes with a built-in on-screen keyboard, which people often use especially for the obscure punctuation symbols for which they may not have memorized the Graffiti gestures. Unfortunately, this keyboard does not have all the special keys found on the PC keyboard, like F1 and ESC, so we had to create our own on-screen keyboard. Just like the regular PalmPilot keyboard, you pop-up the Remote Commander's keyboard by tapping the stylus inside the "abc" area (also labeled "abcde" on some PalmPilots) at the bottom left of the Graffiti area. This brings up the screen shown in Figure 2, on which you can tap the desired keyboard key. To dismiss the keyboard, tap in the "abcde" area again, or tap the "Done" button of the keyboard at the bottom right, or use the Menu command when the keyboard is displayed. The small area above the keyboard is still available to make mouse movements or taps.

The "Shift," "Ctrl" and "Alt" keys on the on-screen keyboard modify the characters hit on the keyboard, and will also modify Graffiti characters and mouse clicks. For example, to do a SHIFT-CLICK of the mouse, you can tap the "Shift" area on the keyboard and then tap the stylus in the upper area.

Because SHIFT, CONTROL and ALT clicking is so common, we wanted to provide more convenient methods for doing this, so we allow the four physical "application buttons" at the bottom of the PalmPilot (see Figure 1) to be used as SHIFT, CONTROL and ALT and FUNCTION. Little labels above the keys show the assignment of the buttons, as shown in Figure 1. Since these buttons normally allow the user to quickly switch to a different PalmPilot application, the use of these buttons as shift keys can be turned off with a Remote Commander menu command, if desired.

← Click in the "abcdef" area to get this keyboard.

Figure 2. The Remote Commander's on-screen keyboard.

The FUNCTION buttons allows Graffiti characters to send all the special PC keyboard keys. We defined a mapping of letters to the function keys so the user never needs to pop up the keyboard. For example, holding down the function button while making the **1** gesture sends F1, using FUNCTION-**e** sends ESCAPE, etc.

Connecting the PalmPilot with the PC

To use the PalmPilot to control the PC, they need to be connected to each other. Our goals for the connection technology include: minimal interference with the use of the PalmPilots, continuous connections while users are writing on the PalmPilots, multiple PalmPilots operating at the same time in the same room, and minimal power consumption.

Ideally, we would use some sort of wireless technology. The most popular short-range wireless technology is infrared (IR), but this does not yet seem to be appropriate for our application. For example, the IR built into the new PalmIII version of the PalmPilot reportedly only transmits for 1 meter, and it must be aimed at the receiver (in the same way that a TV remote control must be aimed at the TV). Only one PalmIII device can be operated in the same area at a time (because they interfere with each other), and the IR will require a lot of power, so you would not want to use it for long periods of time. The designers of the ParcTabs [24] had to go to enormous lengths to make the IR work in those devices, including installing multiple transmitters on each ParcTab, receivers on the ceilings of every room, and an elaborate software protocol to deal with multiple transmissions. They still reported problems, especially with multiple people trying to transmit at the same time. We hope that future IR technologies, like IRDA v2, will resolve these problems for the PalmPilot.

Another wireless possibility is radio. There is a wireless modem for the PalmPilot for about $500, and Bell Atlantic offers wireless Internet service for about $50/month for each PalmPilot. This seems too expensive to use in our Remote Commander.

In the meantime, we are using a wired solution, which has the advantage of being flexible, inexpensive, and easy to set up. It lets us explore the various user interface and systems issues, and is practical for use today.

The Remote Commander currently assumes the devices are connected using the built-in serial ports of the PalmPilot and the PC. The easiest way to do this is to simply use the cradle provided with the PalmPilot. However, using the PalmPilot while it is in the cradle is quite clumsy, so instead, we usually use the HotSync cable sold by 3Com, which does a nice job of staying attached to the PalmPilot without getting in the way. It is also inexpensive.

In order to connect *multiple* PalmPilots to a PC at one time, the PC needs multiple serial ports. For desktop PCs, there are many cards available which provide multiple serial ports. The one we found costs about $400 for 8 ports. For laptops, we found the QSP-100 PCMCIA card [18] which provides 4 ports (see Figure 3) and costs about $500. Most laptops have 2 PCMCIA slots, so this allows up to 8 serial ports using two cards in the laptop. The cards self-configure under Windows95 (they do not currently work on WindowsNT) and use a sequence of COM ports.

Figure 3. Quatech QSP-100 PCMCIA card**.**

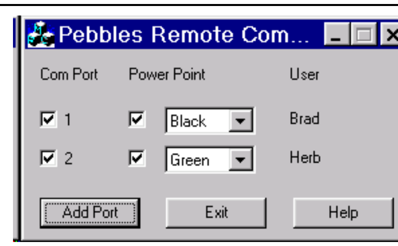


Figure 4. The PC module of the Remote Commander.

PC Module

The PC module of the Remote Commander is fairly simple. It takes the characters and mouse movements reported by the PalmPilot module and inserts them into the Windows input event queue, as if they were regular mouse and keyboard events. The user interface for the PC module is shown in Figure 4. Clicking on the "Add Port" button brings up a dialog box in which a serial port number can be typed, and a user name specified. The "PowerPoint" widgets in the middle are discussed in the next section. The Remote Commander can handle as many users as there are serial ports on the PC. In the future, we may try to make this module more self-configuring, by having the program automatically check for PalmPilots on the PC's serial ports.

PowerPoint version

Of course, Remote Commander can interface with *any* existing Windows application, because it just emulates the regular mouse and keyboard. However, we want to experiment with adapting *existing* applications so they are more amenable to *joint* use by multiple people in the same room.

As an experiment, we added a special feature for using the Remote Commander with Microsoft PowerPoint (versions 7.0 or 97SR-1). In presentation mode in PowerPoint, normally clicking the mouse will change to the next slide, but if you go into "pen" mode, then you can draw on top of the slides in various colors. When the Power Point checkboxes in the middle of the Remote Commander are selected (see Figure 4), then whenever the user drags on the PalmPilot end, Remote Commander will send out codes to set the color of the pen to that user's color. For example, in the case of Figure 4, whenever Brad drew on a slide using the PalmPilot, the ink would be black, and Herb's drawings would be green (see Figure 5). Remote Commander picks unique colors for each user, but users can change their colors by using the menu shown in Figure 4.

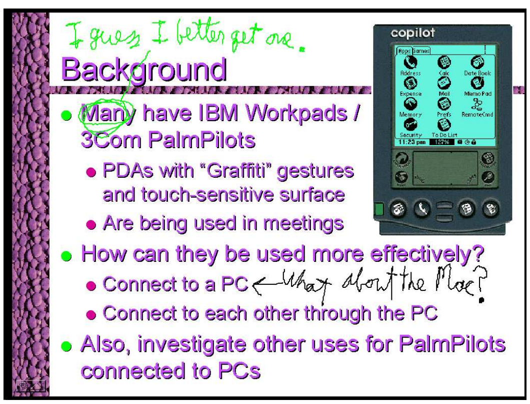


Figure 5. PowerPoint screen showing some annotations entered using Remote Commander from two PalmPilots.

It is important to emphasize that we achieved this without *any* modifications to PowerPoint—Remote Commander just uses the standard commands available from PowerPoint. The color change commands are sent before each mouse drag event (down press) to make sure that the color is set correctly for each user.

An observation while using this feature in early versions of Remote Commander was that, not surprisingly, users often made *mode errors* where they forgot to turn the PowerPoint mode off when switching applications. Remote Commander would then send the PowerPoint change-color commands to the current application, which would do various incorrect actions. For example, it was not possible to un-check the PowerPoint mode in the Remote Commander PC module because the color change commands would cause the module to quit! Therefore, we added more checking to try to detect whether the front-most window is really a PowerPoint presentation and only if so, does Remote Commander send the color-change commands.

In the future, we would like to investigate special versions of Remote Commander for other applications. For example, Microsoft Word allows the color of the text to be changed. Although Word lacks built-in keyboard commands to change the color, it would be easy to build special styles or macros to change the color, assign keyboard commands to these, and then have the Remote Commander execute these commands before each keystroke. Then, each user's input would appear in a different color. Similar techniques could be used for Excel, Paint, and many other tools. This is a very versatile capability that would allow gesturing and annotation on top of many different applications, without the need to change the applications themselves. Eventually, it might be better and more robust to use OLE integration to control the applications, instead of sending special text as commands.

Experiences with Remote Commander

When we use the Remote Commander in meetings, we find that people initially "fiddle around" moving the mouse even when they are not planning to do anything, which interferes with other people's attempts at real work. The result is that someone will tell another to stop sending input. This problem seems to disappear with practice, but if we wanted to address it, we would probably *not* go to a full, formal floor control mechanism, because that seems too inhibiting.

The feedback from the users who have downloaded the Remote Commander has been quite positive, and we have received many nice compliments.

Some substantive suggestions from users have been to support key repeat when the user holds the stylus over an on-screen keyboard key (which we plan to do), and to add the acceleration for cursor movement (already done). Another good idea is to support the "Paste" operation on the PalmPilot module, so text typed into other PalmPilot applications, like Memo, can be easily entered into the PC. Ideas from users we probably will *not* pursue include the ability to support turning the PalmPilot upside down or sideways, so the physical buttons would be in a more natural position (but Graffiti would not work in these orientations), and to have the cursor continue moving when the stylus is pegged at the side of the digitizer (but there is a dead area around the digitizer inside the case, so it would be hard for users to stay over the edge of the digitizer). We will continue to gather feedback and ideas from the users and our own experience with Remote Commander.

PEBBLESDRAW

In addition to Remote Commander, we built another application, called PebblesDraw, to explore allowing *all* users to have their *own* cursors. PebblesDraw is a multi-cursor drawing program that is analogous to other "shared white-board" applications, with one important difference: here all the users share the *same* display. Thus, PebblesDraw qualifies as Single Display Groupware.

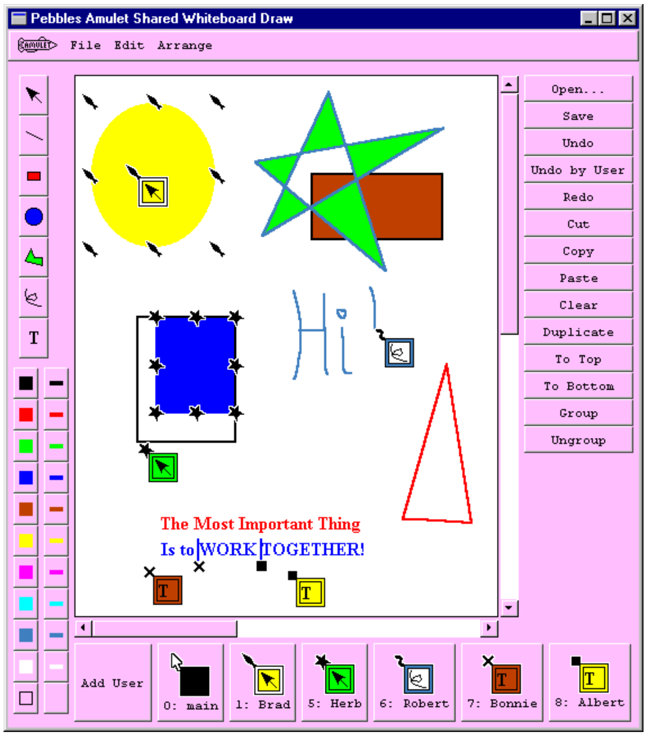


Figure 6. First version of PebblesDraw**.**

The initial design for PebblesDraw is shown in Figure 6. Clicking on the "Add User" button at the bottom allows the name, serial port number and the shape for that user to be entered. All active users are shown along the bottom of the window. At the left are the conventional drawing and color palettes. At the right is a button panel that contains the most common commands.

In creating this application, we discovered a number of important issues not addressed by previous SDG applications. In particular, the standard widgets, like selection handles, palettes, and pull-down menus, do not work for multiple users. Previous SDG systems had little state associated with each user. For example, in KidPad [23] there are no selections. Tivoli [17] supports only a few pen widths. MMM [4] only dealt with up to 4 users. We wanted to support more of the choices available in typical drawing programs so a more general mechanism was required. However, we wanted the interaction style to be similar to conventional direct manipulation applications, so it would be familiar.

The problem with conventional selection handles is that they do not show which user has the object selected. One goal for PebblesDraw is that each user's actions are independent. Each user therefore must have their own separate selection handles. Most previous SDG applications, such as MMM [4], assign each user a color. Since PebblesDraw allows each user to select a color that will next be drawn, we thought using color-coding would be confusing. In fact, the developer of KidPad [23] reported that using color as feedback was not a good idea because kids had a lot of trouble. COLT [5] also used a color to identify people, and reported running into trouble with one color-blind user.

For all these reasons, we assign each user a *shape*. The model was a game like Monopoly where each person picks a shape as their piece. The same shape is used as the user's cursor, and to show the user's selection, to show the text editing position, and to mark the user's commands in the undo history. We selected a set of shapes that could be easily distinguished when they are very small (11x11 pixels). We also needed each shape to have a pointy part at the upper left, so it would be clear how to use them to point at objects. The complete set of the current icons is shown in Figure 7.

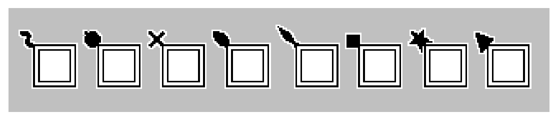


Figure 7. The current set of cursors for PebblesDraw.

When a user executes an operation, it will only operate on that user's selected objects. Thus, if Brad does a cut, only the yellow oval will be affected. Currently, multiple people can select the same object at the same time. The cursor shapes are designed so users can always see that the object is multiply selected, although it can be difficult to tell by which users. The operations do reasonable things if two people manipulate on the same object at the same time. In the future, we might want to disallow multiple people selecting the same object if this proves too confusing. Alternatively, we might make is easier to see which users have the object selected. For example, since there happen to be eight handles around an object and we support up to eight shapes, an obvious idea is to divide the handle positions among all the users who have this object selected. However, this might confuse users into thinking that they can only change the object's size from the handles that have their shape. Further studies of these issues are planned.

The problem with the standard design for palettes is that the currently selected tool is displayed in the palette itself (see Figure 8), which does not work if different users can have different modes. If one user is drawing a red circle at the same time that another is typing blue text, how would that be shown? Most CSCW applications are for multiple machines and assume that each user can see their own private copy of the palettes on their own separate displays, so this is not an issue. The MMM palettes did not show any state and showed each user's current modes only in the home areas. The Tivoli project [17] mentioned this problem with palettes, but apparently provided no feedback as to the users' modes.

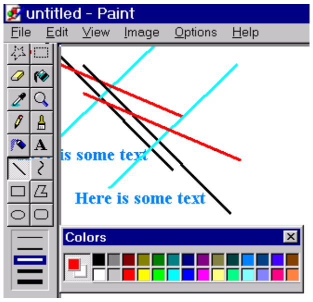


Figure 8. As a single-user application, Microsoft Paint can show the current tool (line tool), the current line style (medium) and the current colors (red line and white fill) in the palettes.

To solve this problem, we provide the feedback for all the user's modes both in that user's cursor as well as in the picture for that user at the bottom of the window. The bottom row corresponds to the home areas in MMM, but we feel that having the modes also in the user's cursor will be less confusing and will require less eye movement. The item for "0" is labeled "main" in Figure 6, and is used to show the modes for the regular mouse and keyboard. Clicking on a item in a palette (at the left of Figure 6) copies that mode into the user's cursor and home area. Notice that the palettes do not show a current mode. Instead, the current tool is shown in the center of the box attached to the cursor, the line color is shown in the outline of the box, and the fill color is shown inside the box. In Figure 6, Herb is using the star cursor. He is currently in select mode, and is growing the blue rectangle. Brad is also in select mode, and has the yellow oval selected. Robert is drawing a freehand shape. Both Bonnie and Albert are editing the second row of text. Each user's text input cursor has that user's icon hanging from it, so users can simultaneously edit text and still know where they are working.

Another set of problems relate to drop-down and pop-up menus. The first problem is that the menu for one user might pop up on top of where a different user is working, especially for the large menus typical in today's applications. The second problem with pop-ups is more technical: each pop-up is implemented as a separate window, and the Windows window manager deals with directing the mouse input to the various windows as they pop-up and disappear. Since in this shared cursor application we are not using the real cursor for all the users (there is only one, after all), it is difficult to direct the PalmPilot input to the correct window, especially since different users might have different pop-ups in different parts of the screen. For these reasons, we decided to try to minimize the use of pop-ups, so the button panel on the right allows the most common operations to be performed without using a pop-up menu (see Figure 6).

Another issue is graying out of illegal items. Since only one user at a time can use the drop-down menus, it makes sense for the items in those menus to gray out as appropriate for that user. For example, if Bonnie has nothing selected, then if she would use the drop-down menus, the commands that require a selection would be grayed out. However, the button panel of commands (at the right of Figure 6) is always visible. Therefore, it does *not* work for items to be grayed out, because some commands will be valid for one user but invalid for another user. Therefore, we had to modify all the operations to make sure that they did something reasonable, like beep or display an error message, if they were invoked when they were not valid for the current user. The previous implementation of these commands in Amulet assumed that since they would be grayed out, they could never be invoked when not valid.

Gestural input is a quick and easy way to give commands without using a tool palette. With a gesture, the *path* of the input device is used to interpret the meaning. Like Tivoli [17], PebblesDraw supports *gestural* input. For example, in PebblesDraw, making an "L" shape while pressing the right mouse button (or the bottom PalmPilot arrow button) creates a new rectangle no matter what the current selected mode is. Making a circular shape with the right mouse button down creates a circle. Other gestures support entering lines and text, deleting objects, and undo. The advantages of gestures are that the user does not have to keep going back and forth to the palette to change tools, and the single gesture defines both the operation to perform, and the parameters such as how big the object should be. The gesture recognizer is trainable, so new gestures can be easily added [10].

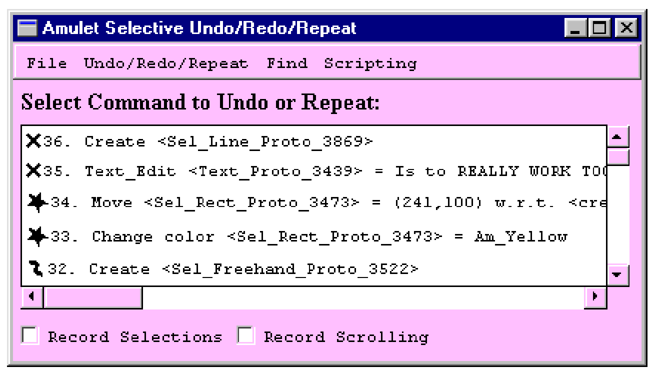


Figure 9. Undo dialog box [13] for PebblesDraw where each command is marked with the shape for the user who performed it.

The undo dialog box for Amulet was augmented to annotate each command with the shape for the user who executed it (see Figure 9). The normal Undo command undoes the last executed command no matter who executed it. The Undo-by-User command undoes the last command of the user who executes this undo command. This takes advantage of Amulet's selective undo mechanism [13] which can undo any previous operation if it still makes sense. For example, in Figure 9, if Herb performs undo-by-user, it will skip over Bonnie's commands and undo the Move of a rectangle (command # 34). If that rectangle had been deleted by a different user, then Herb's attempt to do undo-by-user would just beep, since his last command could not be undone. Of course, regular undo is always possible. Multi-user undo has been previously studied in various systems, such as GINA [3].

The implementation architecture that supports PebblesDraw is described in a separate paper [12]. Briefly, the part of the Remote Commander that accepts input from the serial ports and then converts it into events is linked in with the Amulet toolkit [14]. Instead of inserting these events into the regular Windows input stream, as for the regular Remote Commander, Amulet was augmented to directly accept multiple, parallel streams of input. The Amulet behavior objects (called "interactors") and widgets were augmented to accept a *user-id* parameter. If the user-id is a particular user's id, then this widget is reserved for use by only that user. Each user in PebblesDraw has a separate selection handles widget using this mechanism. Special values for the user-id slot include *"anyone,"* which means everyone can use this widget, even simultaneously, and *"one-at-a-time,"* which means that whoever starts using this widget gets to finish their interaction and other users cannot use it until the first user is finished. *One-at-a-time* is the default, and the buttons, palettes, scroll bars and menus are all marked as *one-at-a-time*.

We envision PebblesDraw as being used to allow multiple users to create or annotate drawings at the same time. To make it more useful as an annotation tool, images can be read in and annotations added on top. The annotations can be saved, read and further edited. For example, Figure 10 shows PebblesDraw being used by 3 people to annotate a plan on a map.

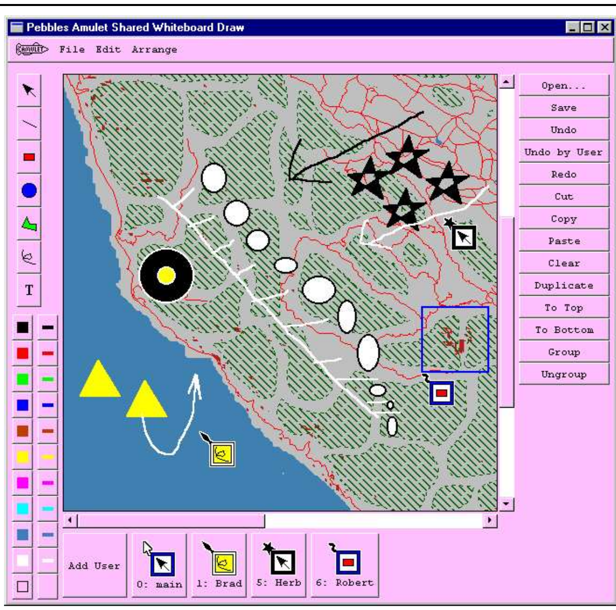


Figure 10. Multiple users can concurrently be annotating and setting up the map.

STATUS AND FUTURE WORK

The first version of the Remote Commander was released for general use, as mentioned above, and the second version along with PebblesDraw will be released shortly. One important area for further research is to more formally evaluate the Pebbles applications.

We have many exciting ideas for future work for other applications which might benefit from multiple PDAs connected to a PC. The next program we plan to write is a Chat program, so that one PalmPilot user can write a note that will be seen on one or more other PalmPilots. This might be useful for passing side notes in a meeting. Another application might support voting, which could either be formal secret or role-call votes, or the informal "how is the speaker doing" style of votes that were done for the ParcTab [24].

Another planned project is joint work with Alex Waibel to use his NPen++ handwriting recognizer [11] with the PalmPilot. NPen++ is much more accurate than previous attempts (like the Apple Newton), but is currently too large to run on the PalmPilot. Therefore, we will use the Remote Commander mechanism so the recognizer can run on the PC and the writing can be performed on the PalmPilot, and the recognized words can then be sent back to the PalmPilot for display.

Some new mice, like the "IntelliMouse" from Microsoft and the "ScrollPoint" mouse from IBM, have an extra input device between the two buttons which can be used for scrolling. In earlier work, we found that it is more effective to use the *other hand* for scrolling, since then the user will naturally operate both input devices at the same time [6]. We will explore using the PalmPilot as the other input device, so it can be used with the other hand for scrolling.

Instead of having the users' current modes shown in the cursor as in PebblesDraw, they might be shown on each person's PalmPilot screen. Then, the user could easily see and choose the desired mode. The interesting research questions in this area are how to make it easy for application writers to specify the palettes for the PalmPilot, and how to augment the communication path to support the high-level *semantic* input from the PalmPilots. The PalmPilot can also be used as the user's "private space," if they want to compose content before sharing it with the group. Another idea is to use the PalmPilot to hold items temporarily, like the Pick-and-Drop pens [19] that make it easy to transfer information from one computer to another.

Of course, we want to explore many other applications of these ideas to new domains. The CSCW literature contains many interesting programs designed for multiple computers, such as "Electronic Brainstorming" and "Structured Idea Generation Process" from Univ. of Arizona [15] and Xerox PARC's Cognoter [21]. We want to see which of these will be effective if used with PalmPilots and a single PC display.

CONCLUSIONS

We have developed applications to investigate many interesting uses for multiple PDAs connected to a PC. The Remote Commander application allows multiple co-located users to take turns controlling the mouse and keyboard of a PC without leaving their seat. The PebblesDraw application demonstrates interaction techniques that allow multiple users to work on the same display simultaneously. These tools can be useful for formal and informal meetings, and even for a single user. Using PDAs in this way takes advantage of technology that people already have and have already learned, and does not require changes in work habits. We are excited about the potential of this approach and the positive response we have received from initial users.

ACKNOWLEDGMENTS

For help with this paper, we would like to thank Lauren Bricker, Andrew Faulring, Richard McDaniel, Robert Miller, Bernita Myers, Jason Stewart, and Bernhard Suhm.

This research was partially sponsored by NCCOSC under Contract No. N66001-94-C-6037, Arpa Order No. B326. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government.

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: To appear.

2. Bederson, B., *et al.* "Local Tools: An Alternative to Tool Palettes," in *Proceedings UIST'96: ACM SIGGRAPH Symposium on User Interface Software and Technology.* 1996. Seattle, WA: pp. 169-170.

3. Berlage, T. and Genau, A. "A Framework for Shared Applications with a Replicated Architecture," in *Proceedings UIST'93: ACM SIGGRAPH Symposium on User Interface Software and Technology.* 1993. Atlanta, GA: pp. 249-257.

4. Bier, E.A. and Freeman, S. "MMM: A User Interface Architecture for Shared Editors on a Single Screen," in *Proceedings UIST'91: ACM SIGGRAPH Symposium on User Interface Software and Technology.* 1991. Hilton Head, SC: pp. 79-86.

5. Bricker, L., *Cooperatively Controlled Objects in Support of Collaboration.* PhD Thesis, Department of Computer Science and Engineering University of Washington, 1998, Seattle, WA.

6. 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.

7. Davis, R.C., *et al.*, *NotePals: Lightweight Note Taking by the Group, for the Group.* CS Division, EECS Department, UC Berkeley, Report #CSD-98-997, 1998, Berkeley, CA.

8. Elrod, S., *et al.* "LiveBoard: A Large Interactive Display Supporting Group Meetings, Presentations and Remote Collaboration," in *Proceedings SIGCHI'92: Human Factors in Computing Systems.* 1992. Monterey, CA: pp. 599-607.

9. Goldberg, D. and Richardson, C. "Touch Typing with a Stylus," in *Proceedings INTERCHI'93: Human Factors in Computing Systems.* 1993. Amsterdam, The Netherlands: pp. 80-87.

10. Landay, J.A. and Myers, B.A. "Extending an Existing User Interface Toolkit to Support Gesture Recognition," in *Adjunct Proceedings INTERCHI'93: Human Factors in Computing Systems.* 1993. Amsterdam, The Netherlands: pp. 91-92.

11. Manke, S., Finke, M., and Waibel, A. "NPen++: A Writer Independent, Large Vocabulary On-Line Cursive Handwriting Recognition System," in *Proceedings of the International Conference on Document Analysis and Recognition.* 1995. Montreal, Canada: IEEE Computer Society.

12. Myers, B.A., "An Implementation Architecture to Support Single-Display Groupware," 1998. Submitted for Publication.

13. Myers, B.A. and Kosbie, D. "Reusable Hierarchical Command Objects," in *Proceedings CHI'96: Human Factors in Computing Systems.* 1996. Vancouver, BC, Canada: pp. 260-267.

14. Myers, B.A., *et al.*, "The Amulet Environment: New Models for Effective User Interface Software Development." *IEEE Transactions on Software Engineering*, 1997. **23**(6): 347-365.

15. Nunamaker, e.a., "Electronic Meeting Systems to Support Group Work." *CACM*, 1991. **34**(7): pp. 40-61.

16. Olson, J. and Rocco, E. "A Room of Your Own," in *Adjunct Proceedings of SIGCHI'98: Human Factors in Computer Systems.* 1998. Los Angeles, CA: To appear.

17. Pederson, E., *et al.* "Tivoli: An Electronic Whiteboard for Informal Workgroup Meetings," in *Proceedings INTERCHI'93: Human Factors in Computing Systems.* 1993. Amsterdam, The Netherlands: pp. 391-398.

18. Quatech, "QSP-100 Four Channel Asynchronous RS-232 PCMCIA Adapter," 1997. Quatech, Inc., 662 Wolf Ledges Parkway, Akron, OH 44311. (330) 434-3154: http://204.210.192.4/public/qsp100.htm.

19. 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.

20. Robertson, S., *et al.* "Dual Device User Interface Design: PDAs and Interactive Television," in *Proceedings CHI'96: Human Factors in Computing Systems.* 1996. Vancouver, BC, Canada: pp. 79-86.

21. Stefik, e.a., "Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings." *Communications of the ACM*, 1987. **30**(1): pp. 32-47.

22. Stewart, J., *et al.* "When Two Hands Are Better Than One: Enhancing Collaboration Using Single Display Groupware," in *Adjunct Proceedings of SIGCHI'98: Human Factors in Computer Systems.* 1998. Los Angeles, CA: pp. To appear.

23. Stewart, J.E. "Single Display Groupware," in *SIGCHI'97 Adjunct Proceedings: Human Factors in Computer Systems, Extended Abstracts.* 1997. Atlanta, GA: pp. 71-72.

24. 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.