Taking Handheld Devices to the Next Level

As wireless technologies mature, the pressing research questions are no longer about how to connect handhelds to conventional computers, but how they can work together, for example as remote controls that add value to the user's environment.

Brad A. Myers
Jeffrey Nichols
Jacob O. Wobbrock
Carnegie Mellon University

Robert C. Miller
Massachusetts Institute of Technology

Thanks to increased processing power and wireless technologies such as Bluetooth and IEEE 802.11 (Wi-Fi), handheld devices are communicating more frequently with conventional computers in offices, meeting rooms, classrooms, and homes.

The smart homes of the future will have ubiquitous embedded computation, and an increasing number of appliances can already communicate wirelessly. Many common home and office items contain computers—televisions, VCRs, stereo equipment, ovens, thermostats, telephones, camcorders, factory equipment, automobiles, and even some light switches. Unfortunately, many computerized features are more of a hindrance than a convenience because their user interfaces are often too complex to intuitively understand.[1]

In 1997, we and our colleagues in Carnegie Mellon University's Human-Computer Interaction Institute (HCII) launched the Pebbles project to determine whether a handheld device, such as a personal digital assistant (PDA) or cell phone, could serve as a simpler, more effective remote control. As part of the project, which is ongoing and will continue for the foreseeable future, we have been studying the simultaneous use of multiple devices,[2] and we have created more than 30 applications to explore novel ways users can apply handhelds as wireless remote controls in offices, meeting rooms, classrooms, homes, factories, and military command posts.

Office-centered applications include using the PDA instead of a laser pointer, using a PDA to remotely control a PowerPoint presentation, and using a PDA with the nondominant hand to scroll windows on a PC.

In the home, we are exploring how to use the PDA as a customizable, intelligent personal universal controller (PUC) for appliances, creating high-quality control panels on the handheld using a high-level specification of the appliance's capabilities. We are also interested in how PDAs can assist in providing both appliance and computer access for the disabled, including the development of new text entry methods for the motor-impaired user.

As the sidebar "From Wired to Wireless" describes, much research focuses on how handheld devices can *replace* PCs and on one-way communication between a remote and the fixed computer. In contrast, Pebbles focuses on using two-way communication to *augment* computers.

CONTROLLING A PC

In PC-control applications, the handheld device augments and controls a PC as it performs its normal functions. The handheld runs our custom applications, which communicate with the PC through any available wireless or wired connection—802.11, Bluetooth, serial cables, or USB.

On the PC side, a special Pebbles program monitors the communication and interacts with PC applications. In some situations, the program just inserts keystrokes and mouse events into the regular Windows event stream, without any knowledge or modification of the PC application that receives

Published by the IEEE Computer Society
0018-9162/04/$20.00 © 2004 IEEE

From Wired to Wireless

The first commercial handheld devices, such as personal organizers and pen computers, were not designed for communication. Even the original Apple Newton required an extra-cost option to support synchronization with a desktop computer.

The original Palm Pilot introduced the one-button HotSync, which was designed to connect the handheld to the desktop computer only about once a day and then over a wire. Microsoft's PocketPC has a similar mechanism called ActiveSync.

One of the first attempts at wireless connection was the ParcTab,[1] part of the original Xerox PARC ubiquitous computing project. The ParcTab was continuously connected to the network through infrared communication. The project investigated some aspects of remote cursors and informal voting about the quality of a speaker, but involved few other applications.

Only since the rise of 802.11 (Wi-Fi) and Bluetooth radio technologies in the past few years has two-way wireless communication between handhelds and other devices become practical. Since then, many research projects have explored some of the topics that Pebbles addresses. Jun Rekimoto has studied numerous ways to use a PDA along with other computers. One is the M-Pad system,[2] which supports multiple users collaborating with PDAs and a large whiteboard, similar to the multiuser application in Pebbles. In an education setting, Georgia Tech's eClass project[3] has studied the use of handhelds in classrooms.

Numerous commercial and research projects are addressing the concept of a universal remote control device. Some commercial products claim to be universal remote controls, such as the Philips Pronto and the OmniRemote software for the Palm from Pacific Neo-Tek. However, these products are either preprogrammed in the factory for limited features on only a few devices, or the user must laboriously hand-program them button by button.

Some research systems have also investigated the use of handhelds as remotes. The Stanford ICrafter[4] is a framework for distributing appliance interfaces to many controlling devices. The XWeb[5] project is working to separate the functionality of the appliance from the display device. Pebbles differs from this research because it is the first project to focus on the ubiquitous remote control of PC applications and the automatic generation of high-quality control panels.

References

1. R. Want et al., "An Overview of the ParcTab Ubiquitous Computing Experiment," *IEEE Personal Comm.*, Dec. 1995, pp. 28-43.
2. J. Rekimoto, "A Multiple Device Approach for Supporting Whiteboard-Based Interactions," *Proc. ACM Conf. Human Factors in Computing Systems* (CHI 98), ACM Press, 1998, pp. 344-351.
3. G.D. Abowd et al., "Investigating the Capture, Integration and Access Problem of Ubiquitous Computing in an Educational Setting," *Proc. ACM Conf. Human Factors in Computing Systems* (CHI 98), ACM Press, 1998, pp. 440-447.
4. S.R. Ponnekanti et al., "ICrafter: A Service Framework for Ubiquitous Computing Environments," *Proc. Int'l Conf. Ubiquitous Computing*, Springer-Verlag, 2001, pp. 56-75.
5. D.R. Olsen Jr. et al., "Cross-Modal Interaction Using XWeb," *Proc. ACM Siggraph Symp. User-Interface Software and Technology*, ACM Press, 2000, pp. 191-200.

the events. In other cases, a special plug-in connects through the standard Windows COM interface so that the PC can retrieve data from the application.

In a business setting

To understand the handheld's potential usefulness, we applied the contextual inquiry technique[3]—in which experimenters observe people in their normal working context and analyze their actions—to study how people interact with the computer during meetings. We found that often someone far from the PC's keyboard and mouse would want to take control, such as in a collaborative design session, where each participant might want a turn to test or edit an evolving prototype. At that point, the participant must elect to go to the keyboard or stay seated and try to tell the person already there what to do with the mouse or keyboard.

From these observations, we created Remote Commander, an application that lets people control the PC from their seats using their handhelds. Figure 1 shows possible handheld displays. All keyboard and mouse functions are available, and either a full-screen (Figure 1a) or zoomed-in (Figure 1b) picture of the PC's screen can appear on the handheld.

The handheld's user can control the PC's mouse in one of three modes, through manipulation of

- the real cursor, in which case people must take turns, since a PC has only one real cursor;
- a simulated cursor that supports "scribbling" on the screen (Figure 1b), in which case each person has a separate cursor with a separate color, but the cursors appear to float above all real applications; or
- an independent cursor inside an application customized to support multiple cursors.

We used the third mode to explore the interaction techniques required when people share a display. Figure 1c shows the PebblesDraw shared drawing program, in which each person's cursor and selection handles appear on the PC's screen with a different shape. Each user's handheld controls that user's cursor.

We later discovered that Remote Commander was also useful for people with motor impairments such as muscular dystrophy or cerebral palsy, who often retain their fine motor control even after losing their gross motor skills. Consequently, although they

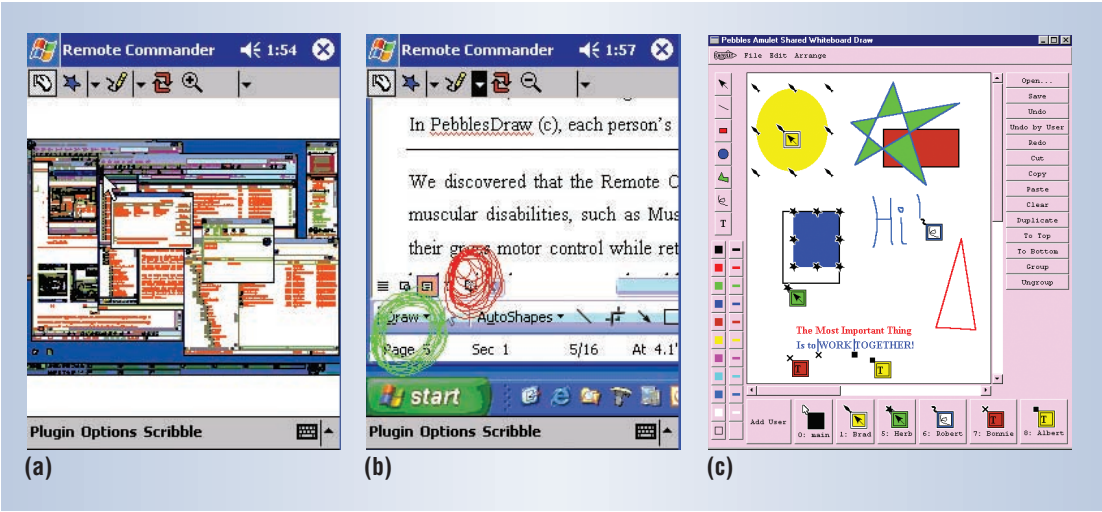

Figure 1. Remote Commander. This application supports all keyboard and mouse functions and displays screen images on the handheld, either (a) showing the full screen or (b) zoomed in so the pixel ratio is one-to-one. (c) The application also permits shared drawing through PebblesDraw, where each person's cursor and selection handles have a different shape.

might not be able to operate a regular keyboard or mouse, they can still use a handheld's tiny onscreen keyboard and mouse control as their PC interface. Remote Commander also provides intelligent word prediction and completion, which can increase entry rates for people with slow motor functions.

At present, we are looking for additional ways to use handhelds to augment and control PCs and appliances for the disabled. One promising technique for people with motor impairment is EdgeWrite, which makes text input more accurate by using a square hole to provide extra stability for a stylus, finger, or joystick as the user is tracing letter forms.[4] EdgeWrite's core concept is to use physical edges to provide this stability so that motor-impaired users can more accurately enter text in handheld devices and desktop PCs.

In our studies, we also found that, in a meeting or presentation, the most natural way to refer to something on a big screen is to point to it, and most participants prefer to use a laser pointer to point to things at a distance. However, using an input technique in which a camera tracks the laser pointer dot has numerous problems. For example, people do not know exactly where the dot will appear when the beam is turned on. They also cannot hold the beam steady, and the beam tends to flick away as the user turns it off.

As an alternative, we created Semantic Snarfing, an interaction technique that copies the content in the vicinity of the laser pointer's dot, transforms that content into a format that users can edit on their handheld, and then returns the edited content to the screen.

Another application, SlideShow Commander, alleviates the difficulties presenters encounter when trying to control a PC while giving a PowerPoint presentation. As Figure 2 shows, a handheld running SlideShow Commander displays the current slide's image and notes, the list of slide titles, and the time. Users can easily change slides, preview other slides without changing the audience view, draw pictures on the current slide, tap on embedded links and hot spots, or switch to and from demonstrations on the PC. SlideShow Commander has been quite popular and has been licensed for commercial sale (www.pebbles.hcii.cmu.edu/slideshow/).

In military technology

The Command Post of the Future (CPoF) project investigated how military commanders and their staff might use advanced technology. One of CPoF's visions was to integrate large screen displays, multimodal input, and laser pointing to extend the staff's personal handheld devices to operate with large wall maps and status displays.

Our part of CPoF, funded by the US Defense Advanced Research Projects Agency from 1998 to 2002, was to enable personal handhelds to display the details of selected public information—a technique we called the "private drill-down of public information."[5] The user would specify the area or items of interest using direct pointing, the Remote Commander mouse emulator, or the Semantic Snarfing laser pointer technique, and the handheld would then display the details of selected items. The user could edit or explore the items on the handheld and then make any required reports or changes public by merging the data with the main display.

We created a prototype of this concept and studied its effectiveness with a map-based task. The results showed that the concept had promise but that tuning the application's user interface on the handhelds so that it matched the effectiveness of the big-screen version would require significant work. An additional issue is the level of coupling between the handheld and the big screen: During private exploration, should the handheld's display still reflect data changes in the main display? When using a handheld as a remote control for the main display, should scrolling and other view changes on one display affect the other? These issues require further study.

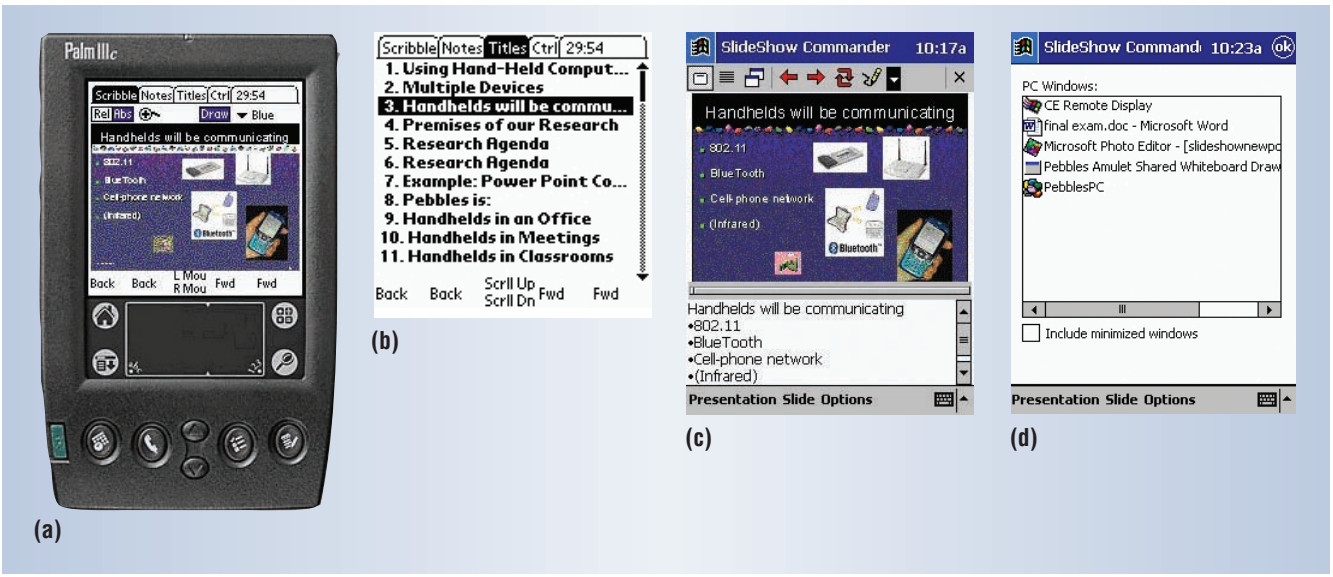

Figure 2. SlideShow Commander. With the application, a handheld can control a PC running PowerPoint. (a) Thumbnail of the slide, (b) list of titles, (c) notes for the slide, and (d) list of other running applications. In (a) and (b), the application is running on a Palm; in (c) and (d), on a PocketPC.

In education

Because many students are bringing their handhelds to classrooms, we wondered if these devices could improve both the instructor and student experience in large lecture halls.

Many instructors frequently stop their lectures to ask the class a question. If the instructor asks a multiple-choice question, students could use handhelds to answer, and the instructor could easily keep track of who is answering and get a bar graph of the results. The process would help keep students thinking about the material, and the instructor could more easily evaluate the students' level of understanding during a lecture.

Starting in 2000, we explored this technique during two second-semester chemistry classes, with about 100 students each. Surveys showed that students preferred using handhelds over other alternatives, such as raising their hands.[6]

Another classroom idea is to adapt SlideShow Commander for use as a note-taking tool. Instructors can save their annotations as public notes, while student annotations would be private notes. Georgia Tech's eClass project has shown that similar features are useful.

In general-purpose computer use

Other applications we developed were aimed more at the individual than the group and proved suitable for a range of tasks. For example, people often put their handheld in a cradle while at their desk to recharge it and synchronize its data with their PC. Our studies showed that the user could place the PDA beside the keyboard and use it with the nondominant hand for various activities. In this manner, right-handed users could efficiently use their left hand on the PDA to scroll windows on the PC while clicking the mouse with their right hand. Our studies showed that using this two-handed technique was faster than conventional scrolling for some tasks, such as scrolling a long Web page to find hyperlinks.

Another general-purpose application is ShortCutter, which lets users create custom panels of shortcut buttons, sliders, knobs, and pads to control PC applications. As Figure 3 shows, with the direct manipulation editor, users can design panels and then assign PC actions to the buttons and other widgets. Users can also create panels to control media players, design control panels for servers without monitors, and start and control their favorite applications.

CONTROLLING APPLIANCES

With ShortCutter, users can draw panels of controls, but individually laying out and programming each control can be tedious. A more efficient alternative is to have the handheld automatically create the control panels. Toward that end, we created the personal universal controller (PUC) to remotely control not just PC applications, but also computerized appliances in general.[7] By "appliance" we mean any electrical or electronic equipment.

The PUC introduces an intermediary graphical or speech interface that lets the handheld engage in two-way communication with appliances. It first downloads a specification of the appliance's functions and then automatically creates interfaces for controlling that appliance. Each appliance's specification includes a high-level description of every function, a hierarchical function grouping, and dependency information that relates the availability of each function to the appliance's state. The specification does not include any information about panel layout or interface design.

The PUC offers many advantages. The first is that *the PUC can make multiple interfaces be consistent*. Many people have trouble setting the clock on their VCR—not because clock setting is difficult, since the same people can easily set their own

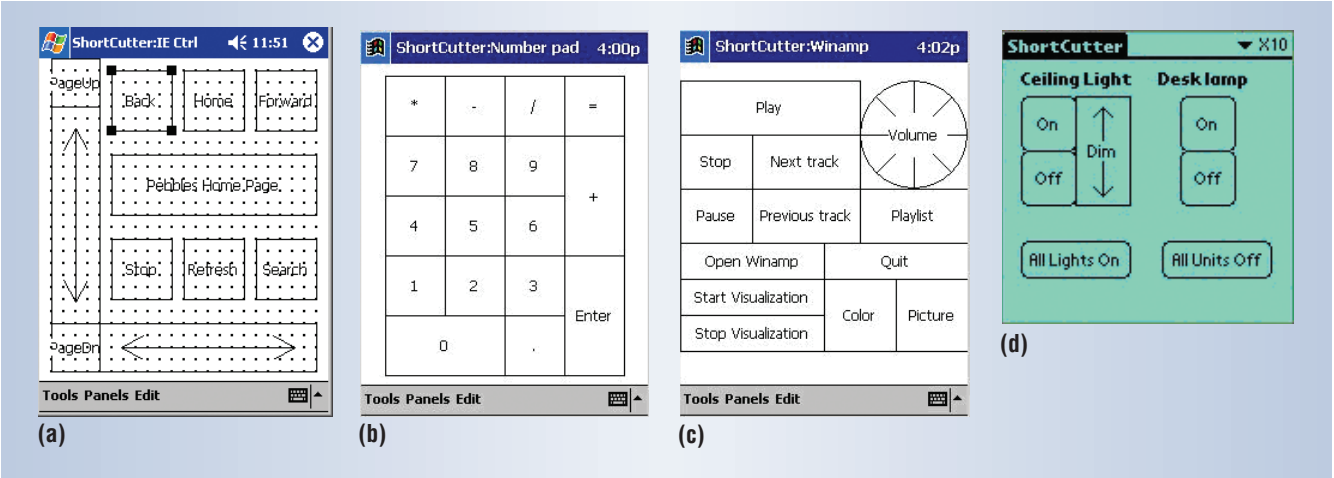

Figure 3. ShortCutter. Users can create custom panels of shortcut buttons and edit the panel by direct manipulation. Actions include (a) scrolling, (b) sending a keystroke, (c) invoking or controlling an application, or (d) sending a command to control an external device, such as a light switch. In (a) through (c), ShortCutter is running on a PocketPC; in (d), on a Palm. Shortcutter is in edit mode in (a) and in run mode in (b) through (d).

alarm clocks, but because each appliance has its own idiosyncratic way of performing functions. The PUC eliminates this problem by providing the same interface for the same functions across all appliances.

Another advantage is that *interfaces can reflect a user's individual preferences or needs*. The user will be able to have larger buttons that are easier to press or fewer options to make the interface less cluttered. For disabled users, the PUC could generate an interface with suitable properties for that disability, such as a Braille interface for a blind person or an interface with only a few buttons for someone with a cognitive impairment. To further investigate these issues, we are collaborating with the standardization efforts of the International Committee for Information Technology Standards' Technical Committee V2—Information Technology Access Interfaces (www.ncits.org/tc_home/v2.htm), which is developing a protocol to help disabled people use everyday appliances.

The *generator customizes the interface to the handheld's platform*. We have created automatic user interface generators for PocketPCs, cell phones, and TabletPC-sized displays. As Figure 4 shows, in each case, the generator creates an interface that takes into account the platform's properties. The SmartPhone in Figure 4b does not have a touch screen, so it requires different interaction techniques.

The PUC can even generate interfaces for different modalities. Using the same high-level specification, it can automatically generate a speech interface, including the grammar, language model, and pronunciation dictionary.

Finally, a key advantage of automatic generation is that *one panel can combine the functions from multiple appliances*. We are working to automatically provide buttons that will invoke a series of functions for high-level tasks. If the task is to play a DVD, for example, the user should be able to press one button that would turn on the TV and switch it to input-3, turn on the stereo and switch it to auxiliary, turn off the cable receiver, turn on the DVD, and then play.

Appliance specification language

As a first step toward determining the PUC's viability and to see what information the high-level specification language should include, we hand-designed control panels for two appliances: a stereo system and a telephone/digital answering machine. We then gave them to several participants in an experiment to compare the ease of performing certain tasks with our PDA interface relative to using the manufacturer's interfaces for the stereo and answering machine.

Even though they were familiar with the appliances' interfaces, when the participants used our PDA interfaces, they completed simple tasks (such as changing the volume) and complex tasks (such as programming the playing order of CD tracks) in about half the time and with half the errors compared with using the manufacturers' interfaces.

Encouraged by these results, we developed an XML-based high-level specification language that contains all the information the PUC needs to create these interfaces automatically. The language, which we have fully documented, includes an XML schema for validating specifications and uses state variables and commands to describe the appliance's functions. Each state variable uses one of the built-in base types, such as Boolean, string, enumerated, integer, fixed point, or floating point.

The specification also provides labels for all variables and commands, including pronunciation information for speech interfaces and labels for specific variable values, such as "off" for the false value of the Boolean power variable.

A unique aspect of our language is that each label can have multiple strings, which means the interface generator can present the most detailed label for the allowed screen space. The specification also includes organization information that groups all commands and variables hierarchically, which aids in automatically generating well-structured interfaces.

Figure 4. Automatically generated interfaces. (a) An interface for a stereo on a PocketPC and (b) and (c) on a Microsoft SmartPhone cell phone. In (a) and (c), a Smart Template renders the play control buttons.

An important innovation in the PUC specification language is that it includes dependency information, which specifies when to enable each variable and command in terms of the other variables. In a PC application's user interface, the controls become grayed out when they are not usable, but this is impossible for a regular remote for two reasons. First, a regular remote has physical buttons, which cannot be disabled. Second, the remote has no way to know the appliance's state, so it cannot know when it would be appropriate to disable a button. However, because the PUC gets updates from the appliance about all state variables, providing dependency information in the specification language lets the PUC gray out the onscreen controls appropriately.

The PUC's user interface generators also use the dependency information to structure the graphical interfaces and interpret ambiguous or abbreviated phrases that the user utters to a speech interface. The generator, for example, compares the dependencies of the state variables and commands to determine if they are ever available at the same time. If two variables are mutually exclusive, the generator can place them on independent panels, since the user will never need them together. Further, the variables that control what is available are usually important mode choices, and the generator should assign them special widgets, such as the panel selector. None of this information is available from hierarchical groupings, which is what other systems have relied on for their automatic generation.

Another common problem for automatic interface generators is that their interface designs do not conform to the domain-specific design conventions users are accustomed to. Solving this problem is particularly important for the PUC because remote-control interfaces frequently use design conventions. A telephone's user interface, for example, should include a standard number pad layout, and a media player should use the standard icons for the play and stop functions.

To address this issue, we developed Smart Templates, which augment the PUC specification language's primitive type information with high-level semantic information.[8] If an interface generator understands a Smart Template, it can apply the appropriate design convention. If it does not recognize a Smart Template, it can still render the appliance's functions because each template is based on the language's primitive elements. Figure 4 shows interfaces generated using our media controls template.

The PUC's two-way communication language lets the appliance and controller device synchronize their state variables at all times.

Appliance adaptors

To control the actual appliances using the interfaces our generators create, the PUC provides appliance adaptors—also called proxies or bridges—which are software and hardware modules that translate between the communication protocols found on many appliances and our PUC protocol. We have created adaptors for many standard and proprietary protocols:

- X-10 for controlling lights;
- AV/C for controlling devices on IEEE 1394 (FireWire) cables, such as a Sony Camcorder;
- HAVi protocol for IEEE 1394 communication, such as with a high-end Mitsubishi HDTV VCR;
- Universal Plug and Play (UPnP), a new standard that devices are just beginning to use;

- custom hardware to connect to a stereo;
- COM interfaces to connect to PC applications such as Windows Media Player; and
- custom software to connect to an ARQ MP3 player, WinAmp, Lutron lighting systems, and a simulated elevator.

We are also looking at using handhelds to control most of a General Motors car's nondriving functions, which can have many customizable settings that users sometimes find confusing. The climate control system on a high-end car can have multiple zones with interlocking modes that determine when to enable and disable various controls. Some cars have navigation systems that also serve as the car's radio, and these can have dozens of settings. We believe that using PUC-generated interfaces can make controlling these functions much easier.

MULTIMACHINE USER INTERFACES

In creating and using our experimental applications, we have made several observations. Unlike conventional user-interface designs, we are dealing with multimachine user interfaces, in which the user is interacting with multiple devices, all of which have displays and controls. Consequently, we must consider which parts of the interface should move to the handheld and which should stay on the original device. For PC applications, any significant text entry probably should stay on the PC, whereas tapping on buttons and reviewing small amounts of status information is easy to do on a handheld.

Even more than with desktop applications, the handheld's user interface design must take into account the context of use—public or private, user sitting or standing. If the user is standing or mobile—as in the CPoF application or SlideShow Commander—providing more functions on the handheld is helpful.

Many open questions remain about how handheld devices can control PCs and computerized appliances. We have implemented most applications for controlling a PC for both PocketPCs and Palms and are now conducting further research that focuses on how people with various disabilities can use handhelds effectively.

The PUC part of the research is continuing on a number of fronts, and the PUC specification language documentation is a free download at www.pebbles.hcii.cmu.edu/puc. We are continuing work on controlling multiple appliances at once, and we are committed to further user testing of the PUC concept, particularly in comparing PUC-generated interfaces with the appliance manufacturers' interfaces.

Our goal in all future research remains to guide the design of interfaces that will improve the usability of common devices and thus make the user's life easier. ■

Acknowledgments

In addition to the authors, many individuals contributed to Pebbles research: Htet Htet Aung, Rishi Bhatnagar, Ben Bostwick, Franklin Chen, Yu Shan Chuang, Karen Cross, Carl Evankovich, Ivan Gonzalez, Marc Khadpe, Dave Kong, Chun-Kwok Lee, Joonhwan Lee, Jennifer Li, Yuhua Li, Leo Lie, Jack Lin, Kevin Litwack, A. Chris Long, Colin McCabe, Choon Hong Peck, Mathilde Pignol, Suporn Pongnumkul, Rajesh Seenichamy, Pegeen Shen, Herbert Stiel, Jeff Stylos, Marsha Tjandra, Claire Tokar, Adrienne Warmack, Jerry Yang, Sunny Yang, and Brian Yeung.

The Pebbles research project has been funded by grants from DARPA, Microsoft, NSF, General Motors, the NEC Foundation, and the Pittsburgh Digital Greenhouse, along with equipment grants from Hewlett-Packard, Lucent Technologies, Mitsubishi Electric, Microsoft, Palm Computing, Symbol Technologies, IBM, SMART Technologies, VividLogic, Synergy Solutions, TDK, Lutron, Lantronix, AT Sciences, Nokia, Synaptics, and Handango.

Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the US government.

References

1. M.D. Brouwer-Janse et al., "Interfaces for Consumer Products: How to Camouflage the Computer," *Proc. ACM Conf. Human Factors in Computing Systems* (CHI 92), AMC Press, 1992, pp. 287-290.
2. B.A. Myers, "Using Handheld Devices and PCs Together," *Comm. ACM*, Nov. 2001, pp. 34-41.
3. H. Beyer and K. Holtzblatt, *Contextual Design: Defining Custom-Centered Systems*, Morgan Kaufmann, 1998.
4. J.O. Wobbrock, B.A. Myers, and J. Kembel, "EdgeWrite: A Stylus-Based Text Entry Method Designed for High Accuracy and Stability of Motion," *Proc. ACM Symp. User-Interface Software and Technology* (UIST 03), ACM Press, 2003, pp. 61-70.

5. B.A. Myers et al, "Fleximodal and Multimachine User Interfaces," *Proc. IEEE 4th Int'l Conf. Multimodal Interfaces*, IEEE Press, 2002, pp. 343-348.
6. F. Chen, B. Myers, and D. Yaron, "Using Handheld Devices for Tests in Classes," tech. report CMU-CS-00-152, School of Computer Science, Carnegie Mellon Univ., 2000; www.cs.cmu.edu/~pebbles/papers/CMU-CS-00-152.pdf.
7. J. Nichols et al., "Generating Remote-Control Interfaces for Complex Appliances," *Proc. ACM Symp. User-Interface Software and Technology* (UIST02), ACM Press, 2002, pp. 161-170.
8. J. Nichols, B.A. Myers, and K. Litwack, "Improving Automatic Interface Generation with Smart Templates," *Proc. 9th Int'l Conf. Intelligent User Interfaces* (IUI 04), Sheridan Printing, 2004, pp. 286-288.

Brad A. Myers *is a professor in the Human-Computer Interaction Institute in Carnegie Mellon University's School of Computer Science. His research interests include user interfaces, handheld computers, programming by example, programming languages for kids, visual programming, interaction techniques, and programming environments. Myers received a PhD in computer science from the University of Toronto. He is a member of the ACM and a senior member of the IEEE. He was awarded membership in the CHI Academy in 2004. Contact him at bam@cs.cmu.edu; www.cs.cmu.edu/~bam.*

Jeffrey Nichols *is a fifth-year doctoral student in the Human-Computer Interaction Institute in Carnegie Mellon University's School of Computer Science. He is the lead student researcher on the Personal Universal Controller Project, exploring how handheld computers can improve the usability of household and office appliances. Nichols received a BS in computer engineering from the University of Washington. Contact him at jeffreyn@ cs.cmu.edu; www.cs.cmu.edu/~jeffreyn.*

Jacob O. Wobbrock *is a fourth-year doctoral student in the Human-Computer Interaction Institute in Carnegie Mellon University's School of Computer Science. He worked extensively on Remote Commander and is the creator of the EdgeWrite input technique (www.edgewrite.com). Wobbrock's research interests include text entry methods, universal design, computer access, and handheld devices. He received an MS in computer science from Stanford University. Contact him at jrock@cs.cmu.edu; www.cs.cmu.edu/~jrock.*

Robert C. Miller *is an assistant professor of electrical engineering and computer science at the Massachusetts Institute of Technology and a member of the MIT Computer Science and Artificial Intelligence Laboratory. His research interests are intelligent text processing, end-user Web automation, user interfaces for software development, and usable security. Miller received a PhD in computer science from Carnegie Mellon University. Contact him at rcm@mit.edu; www.mit.edu/~rcm.*