PointRight: Experience with Flexible Input Redirection in Interactive Workspaces

Brad Johanson, Greg Hutchins, Terry Winograd
Stanford University
Gates 3B-376, Serra Mall.
Stanford, CA 94305-9035, USA
E-mail: *bjohanso@graphics.stanford.edu*

Maureen Stone
StoneSoup Consulting
191 Pine Lane
Los Altos, CA 94022
E-mail: *stone@stonesc.com*

ABSTRACT
We describe the design of and experience with PointRight, a peerto-peer pointer and keyboard redirection system that operates in multi-machine, multi-user environments. PointRight employs a geometric model for redirecting input across screens driven by multiple independent machines and operating systems. It was created for interactive workspaces that include large, shared displays and individual laptops, but is a general tool that supports many different configurations and modes of use. Although previous systems have provided for re-routing pointer and keyboard control, in this paper we present a more general and flexible system, along with an analysis of the types of re-binding that must be handled by any pointer redirection system This paper describes the system, the ways in which it has been used, and the lessons that have been learned from its use over the last two years.

KEYWORDS: Input redirection, ubiquitous computing, multi-display environments.

INTRODUCTION
PointRight is a pointer and keyboard redirection system that operates in multi-machine, multi-user environments. It employs a geometric model for pointer motion across screens that is similar to conventional multi-headed displays, but redirects input across multiple independent machines and operating systems. We have developed it in conjunction with interactive workspaces that include large, shared displays and individual laptops, but it can be applied to any networked collection of machines and input devices.

Development of PointRight began with the simple goal of making it possible for a single mouse and keyboard to provide input to independent desktops on several large displays in a room. It evolved into a general tool that supports many different configurations and modes of use, which has been in use in a number of settings over the past two years. Users find it a natural and intuitive way to interact with multiple devices and have been very positive in their assessment of its utility and usability. This paper describes the system, the ways in which it has been used, and the lessons that have been learned.


Figure 1. The interactive room (iRoom)

BACKGROUND
The vision of ubiquitous computing [18], has come to fruition in the growing diversity of widely used computer hardware, including PDAs, large displays, wireless networks, and mobile devices of all kinds. But, most of the software has been borrowed from standard desktop operating systems and applications. In most ubiquitous computing environments, the software is based on a one-to-one linking of a user with an input device (mouse and keyboard or stylus) and a single display.

For many computer applications, the display space provided by a single display is not adequate. This has been dealt with in all of the widely used windowing systems (MacOS, Windows. X-Windows) by allowing the user's pointing device to operate in a geometric space that is tiled across the multiple displays managed as a single desktop. Keyboard input follows window focus, as usual.

Although this extension is valuable for individual work, it

does not generalize to the case of multiple independent devices and users in an interactive workspace (see also, [2, 3, 16]), such as the *iRoom* [9], shown in Figure 1. The iRoom has four large, permanently mounted projection displays, three 6' diagonal back-projected touch-sensitive SMART Boards [15], a bottom-projected 5' diagonal table-top display, a custom 7' diagonal high-resolution 12-projector tiled back-projected display and a wireless LAN that supports laptops and PDAs brought into the room. Each projector has two inputs: one fed by a machine that is a permanent part of the iRoom infrastructure, and one connected via a VGA splitter to a cable that can be plugged into a laptop or other external machine. While the iRoom machines by default run MS Windows, laptops regularly run any of Windows, Mac or Linux operating systems.

In this more general situation, the multiple display surfaces are not all controlled by a single machine and there is no master input device to operate across them. There can also be dynamic binding between displays and machines, as when a laptop is projected on a wall display in place of the machine native to the workspace.

Application suites in the iRoom, such as those used for construction management [11], make use of a collection of independent applications running on both shared and individual computers in the room. Their unification into a working environment depends on a general cross-device, cross-platform, multi-user facility that enables each user to control all of the devices with the conceptual simplicity of using multiple monitors tiled in a single geometric space.

To solve this problem, we designed PointRight to allow pointer control from any input device to be re-directed from screen to screen as if all of the screens were a large virtual desktop, despite their being driven by different machines The result is a cursor that the user moves seamlessly across the space of displays as though they were a single surface—when it reaches the edge of a screen, it continues its motion onto the connecting edge of the adjacent screen in that direction. If there is no adjacent screen, it simply stops moving and remains at the edge of the screen. The input of any keyboard associated with the pointing device is directed to whatever machine is currently displaying the cursor.

PointRight allows for arbitrary topologies of rectangular screens, flexible mappings between machines and screens, and multiple simultaneously controlled cursors in a workspace. It interacts with a machine's operating system at the level of mouse inputs, so that PointRight can be used to control all applications. It keeps a map of the room's topology, updating it dynamically to account for changes of mapping (which computer to which display) and the status of each computer (running or not). We have deployed the system in several environments and are currently distributing a version as Open Source [7].

RELATED WORK

The previous systems that are closest to the user's experience of PointRight are single-user systems with multiple monitors. PointRight opens up this functionality to the full array of displays, pointing devices, and keyboards associated with all of the computers in an interactive workspace.

Remote interaction applications, such as *VNC* [14] provide an image of the remote screen, on which the pointer is moved. This differs from the PointRight metaphor of moving the pointer off the edge of one screen onto another. To control multiple displays, the user would need to open one VNC window per machine/display and switch among them.

Systems more similar to PointRight include *x2x* [19] and *x2vnc* [6], which provide for configuration of an X-Windows machine such that mouse and keyboard control are redirected to another X-Windows machine or a machine running a VNC server. These are specific to X-Windows and do not support arbitrary topologies, allow dynamic changes based on machine state, allow multiple machines switched to a single screen, or provide for multiple simultaneous redirections.

The *Mouse Anywhere* capacity of Easy Living [2] allows a single mouse to control any of a number of devices, using the physical proximity of a person to a screen to provide the binding. To redirect the pointer input from screen to screen, the user physically moves from a location near one screen to a location near the other.

The *InfoTable* and *InfoWall* [13] provide a specialized case of the PointRight architecture. Control of the cursor is only from laptops, and both the laptop and the other displays can run only specialized Java applications, which hand off pointing and dragging events via Java RMI. This enables more sophisticated features such as "hyperdragging", but does not provide a solution to generally controlling applications in the workspace. The use of dedicated applications is also the key to the iLand system [16] which builds workspace applications on a uniform SmallTalk software platform, and provides cross display pointer control to applications built on the framework.

A number of researchers have developed systems that provide for multiple people simultaneously using a single application [1]. Software such as *PebblesDraw* [12] coordinates the use of pointers by several users within a single application. The Pebbles *Remote Commander* [12] provides for the case of controlling a single shared display from a collection of PDAs. PointRight input at its most basic is indistinguishable from native mouse or pointer input, and as such, it is restricted by the OS to single cursor control per machine. Extending the PointRight implementation to tag input streams from different users, which are then received by applications that can distinguish them, would allow PointRight to be used for multiple cursors on a single display.

System Design Criteria

The following needs were key to designing PointRight:

A single mouse and keyboard for multiple displays: It is impossible to fully operate a standard OS without a convenient mouse and keyboard. While SMART Technologies provides a software keyboard to supplement the touch panels, it is only useful for trivial amounts of typing. When the room was first set up, there was a wireless mouse and keyboard per projection display. As one would expect, this was both confusing and cluttered. The first goal of PointRight was to reduce this to a single mouse and keyboard for the iRoom.

Controlling any device projected on a touch screen: The touch panels for the SMART boards are connected as input devices to the associated iRoom machines. Projecting a laptop or other computer on the SMART board strongly suggests that the touch panel should work for that machine, not the one that is physically connected to the touch panel. Rewiring and then installing the SMART drivers on a projected laptop is clearly an unsatisfactory solution. PointRight solves this problem.

Using a personal mouse and keyboard on a public screen: Having a single iRoom mouse and keyboard is great for a single user, but for multiple users, it is much more convenient to be able to use any mouse and keyboard at hand. Most often, this is a personal laptop. For example, a key iROS facility, Multibrowse [10], makes it easy for a participant in a meeting to bring up any web page or application on any of the shared screens. Through a simple drag and drop action, a file or URL from the local machine appears on the shared display. Often the next step is to take further actions such as following links on the displayed page. Being able to redirect the laptop's input devices to the public screen via PointRight is much more convenient than switching to the iRoom mouse and keyboard at that point.

These needs led to the following system design criteria:

Multi-input, multi-display

PointRight allows multiple users to use multiple mice and keyboards to interact with multiple displays and machines. The only restriction is that only one cursor can be active on any one machine. This is a result of the decision to make PointRight input masquerade as hardware input on any particular machine, which provides the maximum generality with respect to applications. This could be extended for special applications, as discussed above. As in all pointer redirection systems, keystrokes follow the pointer.

Flexible and Dynamic Topology

Interactive workspaces will have a variety of topologies. Not only can screens tile a plane in arbitrary ways, but there may be folds and corners. The system needs to support arbitrary 3D manifolds composed of rectangular screens of different sizes and aspect ratios, providing smooth motion across all of them. For example, in the iRoom setup shown in Figure 2, the left edge of the table display connects to the bottom edge of our front display while the top edge connects to the bottom of the middle SMART Board. Other mappings are possible, such as having each of the three SMART Boards connected to the corresponding third of the adjacent edge of the table screen.

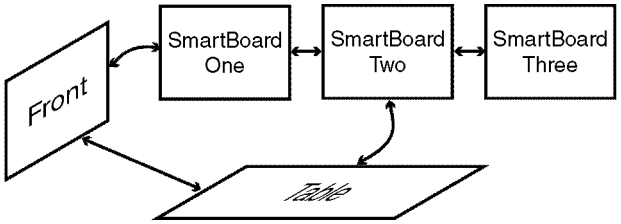

Figure 2. The iRoom screen topology.

The topology can also be dynamic. There can be permanent displays, some of which may switch between displaying several different machines. Some machines may be able to control several different screens simultaneously. Some screens may be on rollers, and laptops may enter and exit the workspace. Therefore, the input redirection system must maintain a dynamic map of the room topology and the mapping between machines and screens.

Different types of input

Some key distinctions constrain the use of different devices in an input redirection system. We identify three different cases and the mappings they support:

Screen-bound: (e.g., SMART Board, tablet computer, eBeam pen) When the pointing device operates directly on the display surface, there is a natural, fixed mapping from the pointing position to a corresponding pixel on the screen. Therefore, only redirection that preserves this mapping will feel natural. This means that a touch panel, for example, only operates naturally on a machine whose display image is visible on the touch surface, either via projection or via a system that mirrors the display pixels, such as VNC[14].

Machine-bound: (e.g., ordinary laptop and desktop). Most common pointing devices, such as mice or trackpads, input relative motion rather than absolute position. This motion is not intrinsically tied to a particular display position. But, the input is conceptually tied to a specific machine by the juxtaposition of the input device with the display. This association is particularly strong for laptops, where the display, keyboard and pointing device share a common package.

The display associated with a machine-bound device is normally private, rather than part of the iRoom topology. The user makes an explicit choice to switch between private vs. room use. The first version of PointRight used a hot-key command, the current one has the user slide the cursor off the top edge of the display to enter the iRoom as

a universal pointer. Because of the strong expectation that machine bound devices operate on their associated display, there needs to be a strong visual reminder (such as a grayed-out screen or a big message box) that input is being remapped to the iRoom.

Free-space:.(e.g., our room mouse and wireless keyboard). Any relative input device can be freely mapped to a workspace. This illusion works particularly well if the actual machine receiving the input is not visible, such as a wireless mouse and keyboard mapped through a machine hidden in the iRoom infrastructure. A free-space device needs an explicit starting screen, which must be active and visible to avoid confusion.

IMPLEMENTATION

The iRoom software is based on a middleware layer called iROS [9], which enables the communication of events and information across all of the machines. The key iROS component is the Event Heap [8], which provides a blackboard-like communication mechanism. PointRight runs on the iROS, and uses the Event Heap for communication, along with direct socket connections as needed for performance. To participate in PointRight, a machine runs a PointRight application and is connected through the local network to the iROS.

The PointRight application consists of a *sender* that redirects mouse and keyboard events from the local input device or devices while the *receiver* accepts remote pointer and keyboard events into the local event stream. Senders use a *space topology description* for the room to direct input to the appropriate display. Receivers are responsible for receiving events and rescaling cursor motions to fit their particular display. Any machine running the PointRight software can operate as either sender or receiver, but not both simultaneously, avoiding problems of recursive redirection, loops, etc. The interaction between a sender and receiver is peer-to-peer with no centralized PointRight server.

The PointRight system currently runs on Windows 9x/NT/2000 and on Mac OS X, and implementation is under way for Linux. The code is available as open source [7], and there are binary installers available for Windows.

Sender Implementation

When a sender is started, it accesses the current space topology description to determine the configuration and status of the available machines. Currently it uses a shared, manually edited configuration file that specifies the basic topology of screens, machines, and connections in a text-based attribute-value format. Dynamic state information is maintained about screen state (on/off), machine state (on/off), and which machine is displaying to which display. We are implementing a more flexible representation that will enable dynamic changes to other configuration aspects as well.

By monitoring events on the Event Heap, each sender maintains dynamic data on the state of machines and displays. Events are generated when a screen is switched on or off, a new machine becomes connected to a screen, or when the PointRight application is opened or closed on a machine. Events are posted and retrieved on the Event Heap server machine, and persist for a little more than two minutes. All active machines post events refreshing their current state every two minutes. If the senders do not see an event from a previously active machine or screen, the state of that object in the local database is updated to "off." This soft-state mechanism allows for the graceful handling of crashes, and also provides a simple mechanism for new senders that come up to acquire the current state of machines and screens in the interactive workspace.

There are two basic modes for the sender, one for screen-bound input, and one for normal (machine-bound or free-space) input. For screen-bound devices, once an event is posted that a new machine is being displayed on the screen, the sender begins forwarding the absolute position of the pointing input to the machine currently displaying to it. With relative input, the sender determines a virtual location for the pointer based on its own screen coordinates (extending beyond its visible screen) and uses the geometrical information to convert the absolute position into an appropriate target screen (based on the current state of connections of machines to screens) and a normalized position on that screen. If a machine is not currently running a PointRight receiver, or a display is off, then the sender skips over that display to the next one in the same direction. If no receiver is available in an indicated direction, then the cursor remains at the edge of the screen, just as it does on reaching the edge of the screen in a standard one-display system. Once a target screen is determined, a command to position the cursor on the machine displaying to that screen is sent.

Receiver Implementation

The receiver is configured with the area of the screen for which it is responsible. The sender passes an absolute value in a normalized range for x and y. These normalized values are scaled to the x and y range managed by the receiver, and the pointer on the receiver machine is set to this position. Keyboard events from the sender are inserted into the event queue on the receiving machine. Events coming from multiple senders are put into the queue in whatever order they arrive. Currently they are not tagged with information as to the sender, but that is a planned extension.

Space Topology Description

The space topology description consists of screens, machines, and connections:

Screens: the dimensions of the physical screen, a set of connections to other screens, a set of machines that can display video to the screen, the state of whether the screen is on or off, and which machine is currently driving the screen.

Machines: Machines are computers running the PointRight application and can therefore receive pointer events (they can also send events, but this information does not need to be shared globally). Data includes the rectangular pixel region for which the PointRight application on that machine is responsible. A multi-headed physical machine will therefore have multiple machine objects. There can also be generic machines, which are dynamically linked to specific machines. For example, there is a special machine called "laptop drop" that represents a laptop connected to the iRoom VGA input cable for laptop projection. In the current implementation, this avoids having to edit the static part of the space topology description for each different laptop.

Connection: A connection represents a valid transition between screens that pointers can traverse. They are represented by an edge (top, bottom, left, right) for each of the connected screens, and the region of the edge on each screen through which the pointer can transition. The region is defined by offsets that define the active region on the edge. For example in the configuration shown in Figure 2, 0′-4′ on the top of table connects to 0′-5′ on the bottom of the middle of the display labeled " SmartBoard Two." Note that a left edge can be connected to a top or bottom, and screens flipped relative to each other are handled by reversing the order of the offsets relative to one another (as is the case with the connection between the table and the front screen in Figure 2). There can be multiple connections to a single screen edge as long as the edge regions for the connections do not overlap.

Communication

A set of PointRight senders and receivers communicate over an IP network with access to a shared Event Heap server. The original implementation of the Event Heap had too much latency to track cursor motion at adequate rates, so a simple socket-based protocol was implemented to send these events directly from the sender to the current receiver. The Event Heap is used to communicate the changes of pointer location to a new screen/machine that lead to opening and closing these sockets (sockets are only opened if there is no existing connection to the new target machine). Keystrokes are communicated using the socket as well.

The current implementation of the Event Heap is efficient enough that we are exploring using it for all PointRight communication. Initial explorations with the Mac OS X implementation suggest that it will easily support a half dozen users on a wireless network (more on a wired network). Above this, the network, not the Event Heap, causes unsatisfactory delays. Events are much larger than the data passed over the sockets, so this can be solved in any way that simply reduces the total traffic on the network.

It is important to realize that no communication over a general, shared network can guarantee the pointer performance provided by a local machine. In modern workstations, both the hardware and the operating system have been tuned to give latency-free pointer input the highest priority. Raw network speed is some compensation, but is not sufficient in itself to guarantee performance for applications such as PointRight.

CONFIGURATIONS

The value of PointRight is its generality, which enables it to be used in a variety of hardware and user configurations. It has been deployed in a number of different settings, summarized below and shown in Figure 3.

The original iRoom

The primary development of PointRight has been in the iRoom, described above and shown in Figure 1 and in Figure 3(a). It is used primarily as a laboratory for our ubiquitous computing research, and for small meetings and presentations.

PointRight has been in daily use for two years, and has become one of the key features of the iRoom. It has been used in all of the ways that were described in the earlier description of needs, and meetings are disrupted when it is not available. Fortunately, it has been relatively robust, and on the rare occasions when it fails, restarting takes only a minute or two. We were initially concerned with whether the non-planar mapping of the left side of the table to the bottom of the front screen (as shown in Figure 2) would confuse users. We found, however, that they used it without hesitation or confusion.

Alternate iRooms

In addition to conducting our own project and course meetings in the iRoom, applications have been developed for other uses, such as construction management [11] and project-based education. We have installed several iRoom facilities for these projects, with different configurations of devices. One uses three front-projected touch panels, as shown in Figure 3(f).

Users have the choice of using the touch screen directly, thereby shadowing the screen, or using PointRight with a free-space device, so they don't have to stand in front of the screen. The overall experience is that the bulk of "driving" the display is done from a PointRight sender away from the screen, but participants (not just the presenter) will often step to the screen for a moment to perform a simple action such as scrolling or following a link.

Another room, the prototype for a project learning setting, has two front projection displays and no touch panels (Figure 3(d)). PointRight on the wireless mouse and keyboard or on laptops is the primary input mode for that room.

Figure 3. PointRight configurations, clockwise from the top right. (a) The iRoom, with its three rear-projected SMART boards and table. The free-space wireless mouse and keyboard, along with a laptop, are shown on the table. (b) The Writing Laboratory, with its big front display and one of its 5 plasma displays. (c) Close-up of the plasma display and its three laptops. (d) Project learning lab, two projectors pointed at a foam core screen, controlled by a wireless mouse and keyboard. (e) A desktop iRoom. (f) The iRoom used for construction management, with its three front-projected, touch sensitive SMART boards. The free space wireless keyboard and mouse are placed behind the projectors to eliminate shadowing.

IRoom2Go

Few environments provide the dedicated high-capacity facilities of an iRoom. We have found that PointRight is equally valuable in "low tech" environments using small amounts of standard equipment. The smallest configuration is a pair of computers (laptops or desktops) one of which runs the Event Heap server (which supports a variety of iROS capacities in addition to PointRight) and one or both of which run a PointRight sender or receiver. We have found it useful in a development environment, where a developer can use a single mouse and keyboard to interact with several machines without using a monitor switcher, as shown in Figure 3(e).

Writing Laboratory

PointRight was used to create an environment for group critiquing of student writing. Laptops are provided for individual students and 5 large plasma panel displays are available for shared use by groups of three laptops as shown in Figure 5(b) and (c). A single projected display is at the front of the room. The PointRight configuration is set up so that students within a group can simply move the cursor off the top of their laptop screens onto the bottom of the shared screen, which displays a document being jointly discussed. Moving off the top of the group screen, the cursor goes onto the projected room screen.

Ordinary word processing software is used on the shared screens, with no special treatment for multiple pointers. Since the students are engaged in face-to-face discussion, ordinary social protocols are quite appropriate and have proved adequate to avoid problems of concurrent action. This negotiation is invisible in the same sense that the movement of pointer from screen to screen is invisible. It does not interpose any explicit mechanisms, but simply meets natural expectations.

To create a mode where each student's cursor moves directly from the laptop to the room screen instead of to a group plasma panel, the plasma panels can be turned off and the dynamic mapping automatically routes across them to the next available target, which is the room screen. When we have a dynamic topology database it will also be possible to accomplish this by changing the connections between screens.

Multi-board integration

Our iRoom includes a machine with multi-display output that can be used to display a single Windows desktop com-

bining the three adjacent SMART Boards. Although it can be controlled through PointRight using a free-space or machine-bound mouse, the natural interaction is to use the touch screens, each of which now has a display that maps to one third of the desktop on the single machine. Although this is a somewhat specialized situation, the fact that the generalized PointRight mechanism can be used for it is indicative of the flexibility of the approach.

LESSONS

In working with these various configurations, we have learned a number of lessons about what makes a cross-machine input redirection mechanism effective.

Understanding input mapping: Our initial implementations had a single way of treating input and display mapping. The distinctions between different kinds of input led to distinguishing the functionality that works for screen-bound and other input devices. While this distinction is invisible to users, it allows them to interact with displays in a more intuitive way.

Detailed model of machines, displays, and connections. To support the flexible configuration and reconfiguration of an iRoom, we needed a detailed model of the correspondence between computers, displays, devices, and regions. One aspect of this was refining the geometric model so that users can go across gaps and around corners without confusion. We have learned that mappings that violate 2-dimensional constraints (e.g., mappings from the side of the table display to the bottom of the front display) do not create difficulty or confusion to the user. The perceptual/motor mapping to the 3-dimensional space leads to use that seems "obvious".

Robust dynamic data. The soft state mechanism makes it possible for machines to react to system changes without interruption. In particular, if some machine goes down, the system adapts to the change within a couple of minutes without the need for explicit intervention. Unfortunately in today's computing environment, this needs to be planned for as a routine event

Include personal machines. PointRight can easily be installed as a sender on a personal laptop. When a machine starts up it simply needs to run the application to become a sender. This makes it realistic to use individual laptops for PointRight input, even when they are not dedicated for use in the workspace. This use has turned out to be one of the most effective uses of PointRight.

FUTURE WORK

We are continuing to develop and extend PointRight in several directions:

Dynamic topology update. Currently, data about the interactive space is partitioned into two components: static and dynamic. Information about the size and location of displays, projectors, machines, etc. is stored in a file that is accessed by the senders when they start up. Information about the current state of each of these (whether projectors are on or off, what machine currently is feeding the VGA cable, etc.) is maintained dynamically through the soft state mechanism of the Event Heap. We are developing ways to interactively update the static information without having to edit a configuration file. Some of this can be done automatically (e.g., detecting new machines that are running a PointRight receiver). In the long run, we can imagine a context-aware environment in which sensors report the location, identity, and orientation of each device, possibly extending the vision-based techniques used to locate laptops on the InfoTable [13] and to track people in Easy Living [2].

Further integration of the visual space. PointRight provides a unified multi-display space for pointing. But, the illusion of a continuous desktop breaks down if the user trys to drag a window or icon across a display boundary. We would like to find a method that elegantly extends PointRight to moving information around the iRoom, while maintaining our focus on general, heterogeneous applications and operating systems.

Flexible, concurrent input. PointRight already enables multiple users to point at the same screen—they just can't do it simultaneously. Having multiple users active on the same screen requires multiple cursors (or the equivalent), plus OS and application support for multiple inputs. Using the latest PointRight implementation, which uses only the Event Heap for communication, we are exploring various flexible input architectures.

In other work in our interactive workspace, we have developed a large high-resolution interactive display based on a modeless interaction style that does not require a single input focus [5]. It was developed using a pen-based technology that supports only a single physical input device, but the underlying structure could be used to allow concurrent operation by multiple PointRight senders, and we hope to extend it in that way.

CONCLUSIONS

PointRight was originally designed to provide common keyboard and mouse control for the collection of machines in the iRoom. It has evolved to a general pointer redirection mechanism, and has been deployed in several other settings with configurations substantially different from the iRoom.

Unlike the other systems we know of, it was designed to provide for heterogeneous operating systems and applications, combinations of fixed and mobile devices, and flexible layout topologies. It supports dynamic environments, multiple machines per screen, and multiple screens per machine, and allows for direct interaction input devices such as touch screens. The system requires only that an Event Heap be running in the interactive workspace, and that each machine that is going to participate, either as a source or

target of pointer events, install a small application. Users need no special training or explanation, beyond a simple introduction to the idea. They find using PointRight to be intuitive and convenient.

PointRight is a part of the iROS software that we have made publicly available [7], and we expect to develop it further in response to feedback from a larger community of users. The iROS software is part of the Stanford Interactive Workspaces project (iwork.stanford.edu).

Finally, the fluidity of using the PointRight system is difficult to convey without seeing it in action. A video of the system is available in streaming RealVideo format at http://graphics.stanford.edu/papers/pointright-uist2002/..

ACKNOWLEDGMENTS

The Interactive Workspaces project is the result of efforts by many students. Thanks to Bryn Forbes, and Rito Trevino in particular, who helped figure out how to tap into the Windows and Linux mouse and keyboard event systems, and to Susan Shepard for her help making the video and keeping the iRoom stable enough to develop in. Aren Sandersen and Brian Luehrs have done the most recent implementation and maintenance on the Windows version of PointRight, Jeff Raymakers and Robert Brydon created the OS X version. The work described here was supported by DoE grant B504665, The Wallenberg Gobal Learning Network, and by donations of equipment and software from Intel Corp., InFocus, IBM Corp. and Microsoft Corp.

REFERENCES

1. Bier, Eric A., Steve Freeman, Ken Pier, MMM: The multi-device multi-user multi-editor, *CHI92*

2. Brumitt, B., Meyers, B., Krumm, J., Kern, A., and Shafer, S, EasyLiving: Technologies for Intelligent Environments". *Handheld and Ubiquitous Computing*, September 2000.

3. Coen, Michael. Design Principles for Intelligent Enviroments**.** Proceedings of AAAI'98. Madison, WI, 1998

4. Fox, A., Johanson, B., Hanrahan, P., Winograd, T., PDAs in Interactive Workspaces, *Computer Graphics and Animation*, May, 2000.

5. Guimbretière, François, Maureen Stone, Terry Winograd, Fluid Interaction with High-resolution Wall-size Displays, *UIST 2001*.

6. Hubinette, Fredrik, x2vnc, http://fredrik.hubbe.net/x2vnc.html.

7. iROS open source, http://iros.sourceforge.net

8. Johanson, B. and Fox, A., "The Event Heap: A Coordination Infrastructure for Interactive Workspaces," To appear in *Proc. of the 4th IEEE Workshop on Mobile Computer Systems and Applications (WMCSA-2002),* Callicoon, New York, USA, June, 2002.

9. Johanson, Brad, Armando Fox, and Terry Winograd, The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms, *IEEE Pervasive Computing Magazine 1(2),* April-June 2002.

10. Johanson, B., Ponnekanti, S., Sengupta, C., Fox, A., "Multibrowsing: Moving Web Content Across Multiple Displays," *Proceedings of Ubicomp 2001*, September 30-October 2, 2001.

11. Fischer, Martin; Stone Maureen; Liston, Kathleen; Kunz, John; Singhal, Vibha (2002). "Multi-stakeholder collaboration: The CIFE iRoom." Proceedings CIB W78 Conference 2002: Distributing Knowledge in Building, Aarhus School of Architecture and Centre for Integrated Design, Aahus, Denmark, pp. 6-13.

12. Myers, Brad A., Herb Stiel, and Robert Gargiulo. "Collaboration Using Multiple PDAs Connected to a PC." Proceedings CSCW'98: *ACM Conference on Computer-Supported Cooperative Work*, November 14-18, 1998, Seattle, WA. pp. 285-294.

13. Rekimoto, J. "Augmented Surfaces: A Spatially Continuous Work Space for Hybrid Computing Environments." *CHI'99*, pp. 378-385.

14. Richardson, Tristan, Quentin Stafford-Fraser, Kenneth R. Wood & Andy Hopper, "Virtual Network Computing", *IEEE Internet Computing*, Vol.2 No.1, Jan/Feb 1998 pp33-38.

15. Smart Technologies SMART Board, http://www.smarttech.com/ SmartBoard/.

16. Streitz, Norbert A., Jörg Geißler, Torsten Holmer, Shin'ichi Konomi, Christian Müller-Tomfelde, Wolfgang Reischl, Petra Rexroth, Peter Seitz, Ralf Steinmetz, i-LAND: An interactive Landscape for Creativity and Innovation, *CHI99*

17. Tandler, Peter: Software Infrastructure for Ubiquitous Computing Environments: Supporting Synchronous Collaboration with Heterogeneous Devices. In: *Proceedings of UbiComp2001: Ubiquitous Computing*. Heidelberg: Springer LNCS 2201, 2001, pp. 96-115.

18. Weiser, M., The computer for the twenty-first century. *Scientific American*, 94-100, September 1991.

19. x2x, http://ftp.digital.com/pub/DEC/SRC/x2x/