The Stanford Interactive Workspaces Project

August 2004

Brad Johanson, Armando Fox, Terry Winograd

Stanford University, Stanford, CA

{fox,winograd}@cs.Stanford.edu, bjohanso@graphics.stanford.edu

1 Introduction

The Stanford Interactive Workspaces Project has created and studied new technologies for integrated multi-person, multi-device collaborative work settings. In addition to our primary testbed, the *iRoom*, we have deployed a number of interactive workspaces at Stanford and at other institutions and evaluated their use in educational settings. The core technologies in these spaces are built around our software infrastructure, *iROS*, which provides a suite of tools for integration and interaction.

Within the broad field of ubiquitous computing we have chosen to focus on co-located collaborative work, emphasizing large shared and walk-up displays using touch or pen interaction.

There are three major aspects to our research:

- Interaction design for shared computer-augmented spaces
- Robust flexible infrastructure for integration
- Empirical studies of collaborative work

This section describes our approach to each and highlights some of the results. More detail on the specific projects is presented in the subsequent sections.

1.1 Fluid Interaction in Ubiquitous Environments

When a person speaks a language fluently, the use of that language is invisible and effortless. The speaker's attention is on what is being said and on the interaction with other people, not on the language itself. *Disfluency*, on the other hand, leads to interruption and reflection. "What does that mean?", "How do I say that?" and "Why didn't they understand me?" Everyday interaction with computers is full of disfluencies, even for experts with long experience and deep knowledge of computing. Tremendous improvements can be achieved by tackling disfluency in the design of the overall user experience. Design concerns go beyond a particular interface or interaction device, to encompass the setting, the user's background, and interaction with other people.

In a ubiquitous computing environment, computing appears in many forms, including interactive displays, handheld mobile devices, pads, wearables, and computers that watch and listen. Without careful design, this will be a Tower of Babel that requires users to operate in not one but several languages, all at the same time. Also, in an interactive environment computers are a part of a dynamic that incorporates people and artifacts, with attention being shared among them.

In designing the interactive aspects of iROS, our focus has been on providing an integrated environment that allows the user's attention to remain focused on the work and people, rather than on the mechanics of interaction. We have started with a general kind of use that we call an "*open participatory meeting*." In this setting, a small group of people (up to a dozen) work together to accomplish a task, usually as part of an ongoing project. People come to the meeting with relevant materials on their laptops or saved on file servers, in a variety of formats for different applications that will be used as part of the meeting. During the meeting there is often a shared focus of attention on a "primary display" with some amount of side work that both draws material from shared displays and brings new material to them. In many cases a facilitator stands at the primary display and is responsible for overall flow of the activities. Examples of such meetings we have supported and observed include our own project group meetings, student project groups in courses, construction management meetings, brainstorming meetings by design firms, collaborative writing courses, and design project group work in a variety of physical settings, including dedicated project spaces and walk-up group facilities.

Our interaction research has included three main components: the development of interaction techniques especially suited for large wall-based or table-based displays shared by a group; the design of *overface* capabilities that are used on top of the standard interfaces to provide access and control to information and interfaces in the room as a whole; and support for prototyping interactions involving new tangible interaction devices.

1.2 Robust Flexible Infrastructure for Integration

We have identified some key characteristics of the required infrastructure for practical ubicomp environments:

Heterogeneity: A variety of different devices will be used in the workspace, chosen for their efficacy in accomplishing specific tasks. In addition to desktop workstations, these include laptops and PDAs with wireless connections used in conjunction with shared displays, as well as physical and tangible technologies specially designed for ubicomp settings. All of these need to interoperate in spite of heterogeneity in software, including legacy applications, since it is infeasible in most cases to write new applications or versions just to take advantage of interactive workspace facilities. One of the main consequences of this heterogeneity is that the software framework must provide cross-platform support—this is not an optional feature. From the HCI perspective, interfaces need to be customized to different sized displays, and possibly different input/output modalities such as speech and voice. The situation is even more complicated in that interfaces will span multiple devices that come and go dynamically in the space.

Dynamism: Interactive workspaces are dynamic on both short and long time-scales. On short time scales, individual devices are turned on and off, wireless devices enter and exit the space, and pieces of equipment may break down for periods of hours or days. On longer timescales, workspaces will incrementally evolve as devices are introduced to facilitate some specific meeting or design task rather than being coherently designed and instantiated once and for all (this is also conjectured to be true for smart homes [19]). The dynamic nature of workspaces means that a software framework must handle applications and devices joining and leaving while minimizing the impact on other entities in the space.

Robustness: For interactive workspaces to become widely deployed they need system stability in the face of change and transient failures. They must "just work" without requiring a full-time system administrator. Users will treat the devices in interactive workspaces as appliances that should not fail in unexplainable ways. Commercial-off-the-shelf (COTS) hardware and software are prone to transient failures and are not designed to be integrated into a heterogeneous environment. Thus, failure needs to be anticipated as a common case, rather than an exception [36]. All of this means that the software framework must ensure that failures in individual applications and devices are non-catastrophic, and must provide for quick recovery, either automatically or by providing a simple set of recovery steps for the users.

Multiplicities: Finally, unlike a standard PC where a single user and set of input and output devices provide interaction with the machine, an interactive workspace by its nature has multiple users, devices and applications all simultaneously active. A software framework must allow group interactions and flexible partitioning of devices into sub-groups.

Our central design philosophy for infrastructure is *design for integration*. There is no such thing as a stable system when we are dealing with rapidly-evolving technologies: this was demonstrably true of the Internet as well as of the personal-computer industry, and there is every reason to believe it will be true in ubiquitous computing for the foreseeable future. As a corollary, we cannot expect people to discard their existing systems and workspaces in favor of newer technology: some degree of backwards compatibility must be retained, and some ability to smoothly and incrementally integrate and migrate to new technology must be provided. Today's exotic new hardware and latest-version software are tomorrow's legacy components. Although efforts such as Jini and UPnP (Universal Plug and Play) are attempting to establish universal standards for devices and interaction, we note that even after several years of discussion, the Jini standard for a "printer" device is still in committee, suggesting that it is trickier than expected to establish a standard even for seemingly simple and well-understood devices.

Consequently, our designs focus not on solving "the problem" as we see it today, but on facilitating *evolutionary* progress into the future. Rather than investigating systems, application suites, and their use just in our specific space, we decided to investigate software techniques that can be used in differently configured interactive workspaces. Our goal is to provide a framework that serves a role similar to the device-driver model, window-manager system, and look-and-feel guidelines for PCs. In other words, we want to create standard abstraction and application design methodologies that apply to any interactive workspace [52].

Our overall system is called iROS, which stands for Interactive Room Operating System. It embodies "design for integration" at all levels: it leverages as many existing subsystems and protocols as possible (Web protocols, Win32 applications, Java); provides specific facilities for allowing the introduction of new functionality gradually and without breaking compatibility with existing functionality; assumes that transient failures will be common in any heterogeneous collection of hardware and software components; and wherever possible emphasizes ease of development or prototyping rather than striving for an optimal but necessarily ephemeral "packaged" solution to a problem.

iROS is a best viewed as a meta-operating system or middleware infrastructure, tying together devices that each have their own low-level operating system. It is a multiple server-based architecture, with each server dedicated to a physically bounded space (typically a room or building). This allows the system to honor the boundary principle [36], which states that ubiquitous computing infrastructure should only allow interaction between devices within the bounds of the local physical space within which a human would assume devices to be collocated. For example, a user might assume that all the devices currently in the conference room could interact, but none of them would be assumed to interact with devices back in his office, unless networking were to be explicitly invoked. We have sought to create a software infrastructure that is associated with a specific physical interactive workspace, and that supports the human-computer interaction needs of applications in that space. To that end, the basic components of our software are charged with making this boundary explicit to applications and devices in the interactive space.

Much of our experimentation has been in a specific instantiation of the space, called the iRoom (see Section 2.2). Other experiments have been in the iLoft in the Stanford Center for Design Research [39], the iLounge at the KTH in Sweden [16], the Teamspace in the Meyer Library at Stanford, and a variety of other sites.

1.3 Applications and Empirical Studies of Collaborative Interaction

Our collaborative technologies have been tested in a variety of modes and settings, ranging from informal observation of our own use of the iRoom to controlled studies of relevant aspects of human perception and the use of particular devices and systems. These studies have some common threads:

- **Emphasize co-location.** There is a long history of research on computer supported cooperative work for distributed access (teleconferencing support). To complement this work, we chose to primarily explore new kinds of support for team meetings in single spaces, taking advantage of the shared physical space for orientation and interaction.
- **Reliance on social conventions.** Many projects have attempted to make an interactive workspace "smart" (usually called an *intelligent environment*) [10, 15]. Rather than have the room figure out what a group is doing and react appropriately, we have chosen to focus on providing the affordances necessary for a group to adjust the environment as they proceed with their task. In other words, we have set our *semantic Rubicon* [36] such that users and social conventions take responsibility for actions, and the system infrastructure is responsible for providing a fluid means to execute those actions.

Some of the experiments are described in more detail below. They have included:

- **Construction Information Workspace:** In our development we have collaborated with other research groups and practitioners to construct "non-toy" applications in design, education, and engineering. One extended development project, in conjunction with the Stanford Center for Integrated Facilities Engineering (CIFE) included studies of users in the construction industry and the design of a Construction Information Workspace [37]. Studies of users in construction planning meetings without computer augmentation showed that they spent almost half of their time simply locating and displaying information. The goal of the prototype system, combining iROS technologies with 4D CAD [38] was to automate the retrieval and cross-correlation of information so that bulk of the human effort could be directed to discussion and planning.

- **Collaborative writing**: Students in a series of courses offered by the Stanford Program in Writing and Rhetoric used an iROS installation combining individual laptops, small-group shared plasma displays, and a classroom-shared projection display. They could move writing samples from one to the other and jointly edit and work on them while having discussions.
- **Walk-up collaboration**: In the TeamSpace project [47], a simple subset of iROS facilities was provided for casual use by students in a public study space. Studies showed that they were able to easily and quickly master the technologies and adapt them to their group practices.
- **Design project capture and recall**: WorkspaceNavigator [35] consists of a suite of tools built on iROS for the capture and review of activity in ongoing design projects. It was deployed in design project spaces in Mechanical Engineering and used over the course of several months. One interface, intended for the design team members, used timelines and visual overviews of the workspace to help index and access both snapshots and online activity records. Another interface, for use in design research, provided visualizations of group activity based on sensors and monitors in the workspace.
- **Remote design team drawing sharing**: An experiment called GroupBoard [39], studied the effects of shared vs. individual display visibility in the work of tele-distributed design engineering teams
- **Sharing and communication in tabletop interaction**: We have conducted studies both on the bottom-projected iTable and the top-projected multi-touch Diamond Touch table [18], in which users cooperatively perform tasks that involve the organization and arrangement of visual materials. In one experiment on sorting photographs [21], behaviors on electronic vs. manual tables were compared. In another [40], the tabletop interaction has been augmented by audio in both private and shared modes, leading to different kinds of cooperative behavior.

After giving a brief history of the project, in subsequent sections we provide details of our attempts to enable fluid interaction as well as the software abstractions that support the required mechanisms. We then describe in more detail some of the empirical studies mentioned above.

2 History of the project

2.1 Interactive Mural

The initial impetus for the Interactive Workspaces project in 1999 was research done jointly with the Stanford Graphics Laboratory on large (white-board sized) high-resolution displays. The graphics group was developing software to display large high-resolution images on tiled displays using multiple projectors [27, 48]. We designed and built an initial 4-projector prototype display, called the *Interactive Mural*. It was followed by one using 8 projectors for a resolution of approximately 4000 by 1500 pixels. To support interactive applications we experimented with several different kinds of pointing devices and their integration into the OpenGL-based display software. Candidate devices included gyroscopic mouse, ultrasonic pen, and laser pointer tracking [14, 54]. We standardized on the use of an ultrasonic (eBeam™) pen[1]. Several experiments were done on the use of this device to support different interactive techniques, including the FlowMenu [25] and a multi-pane Geometer's Workbench for interactive mathematics visualization [26]. As part of a joint interactive art project with the Art Department, we installed a 10' x 12' pressure-sensitive floor in front of the mural. Each of 120 1' x 1' "footsels" returned a binary value, based on a pressure threshold. The input was accurate enough to indicate when someone was standing with at least one foot within the area.

The final version of the Interactive Mural was installed in the iRoom and used for the development of the PostBrainstorm interface[24]. 12 projectors achieve an overall resolution of about 70 dpi over a 6' diagonal screen. A custom-built mounting system (Figure 1) was used to align the projectors with sub-pixel resolution [48]. The 9 megapixels provided a whiteboard-like space with approximately the same resolution as a standard 21" monitor. This allows both fine work at the surface and the ability to stand at a distance to get the overall picture. The mural was powered by a 32 PC rendering cluster and a custom software system called WireGL [27], which allows it to appear to applications as a single large OpenGL device. It uses an eBeam ultrasonic pen[1], augmented with a button that provides two distinct modes (used for drawing and commands).

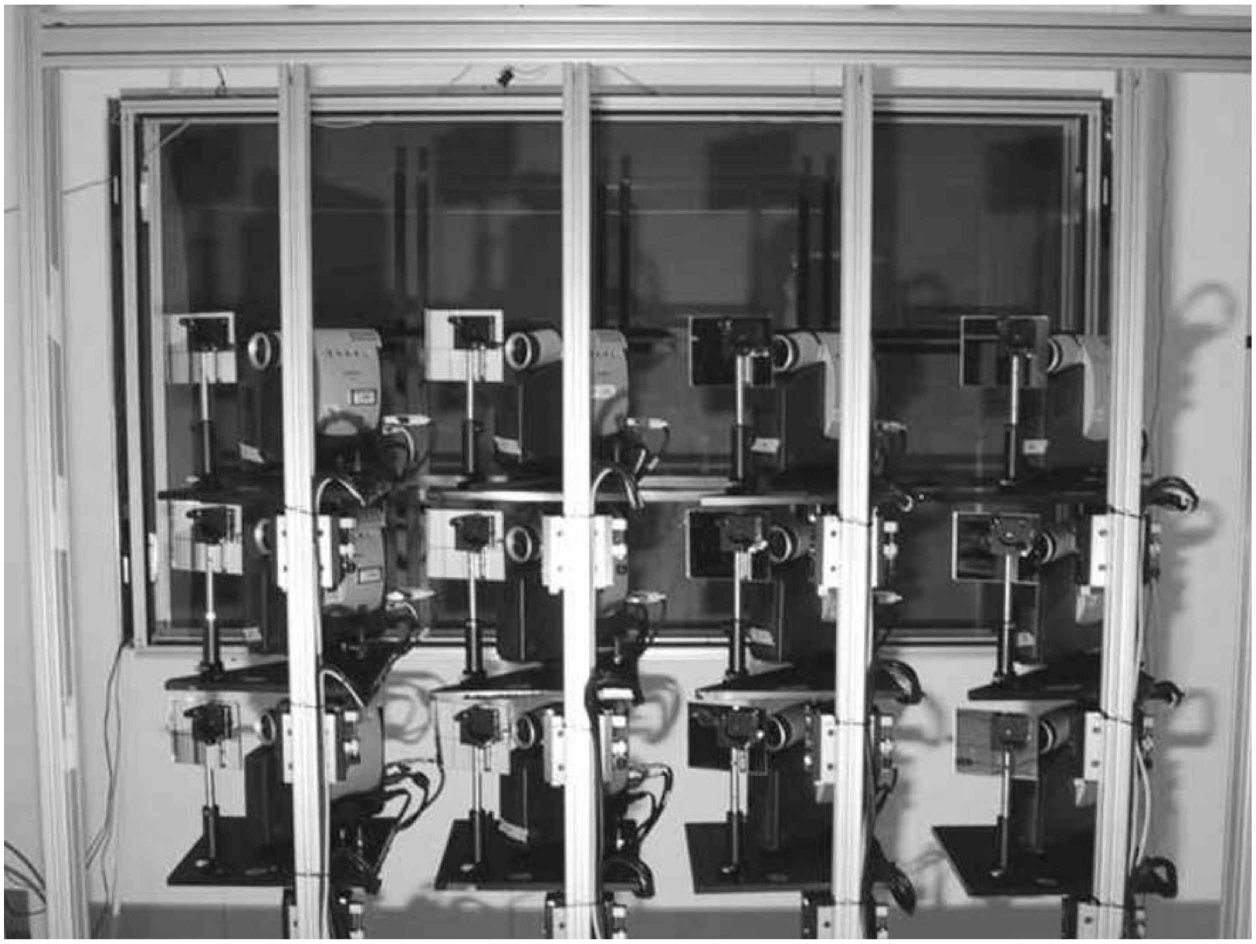


Figure 1- View behind Interactive Mural, showing custom mounting hardware and projectors

2.2 iRoom Version 1

Although the free-standing first version of the mural was useful for experimenting with interaction techniques, its middle-of-the-lab setting meant that it could not be used practically for long periods of time and offered little integration with other devices, such as workstations, laptops, and PDAs, through which people got their work done. To enhance utility in realistic uses, we began to investigate the design of rooms containing one or more large displays with the ability to integrate portable devices and to create applications integrating the use of multiple devices in the space.

In Summer 1999, we designed and constructed a conference-room-like setting called the Interactive Room, or *iRoom*. Our initial iRoom was very similar to the second and final version of the iRoom, which is discussed in the next section and appears in Figure 2. It contained three 5' diagonal SMART Board™ touch-screens along one wall and a custom-designed bottom-projection table, called the *iTable.* which uses a projector and mirror under the raised floor to project the image on the table surface. No barrier is used to prevent legs from obstructing the light path, which makes the iTable look more like a standard conference room table. The initial iRoom also had a front-projected full-wall display and a wireless network that enabled laptops in the room to communicate.

In a student project course on "Interactive Workplaces" five student teams wrote applications for the iRoom, including an application to do movie storyboarding, which integrated Adobe Premiere; an image sorter with the ability to scan hand drawn sketches, and a Presentation Application, which allowed all three screens to be used as part of a coordinated PowerPoint application. One of the projects also created a directional sound API for the iRoom and installed a 4.1 speaker surround-sound system. With Barehands [45] we also experimented with augmenting the touch-screen interaction on the wall displays so that hand posture (as viewed by a behind-screen camera) could provide an additional input channel.

2.3 iRoom Version 2

In Summer of 2000, we rebuilt both the physical environment and the software infrastructure. The 12-projector version of the Interactive Mural (Section 2.1) was integrated as the front of the iRoom. This required a reconfiguration of the entire workspace. During the remodel we introduced more compact light-folding optics for the projectors on the SMART Boards, and did a better job of running the over one-half mile of cables for the room. Based on a plan arrived at with the help of designers from IDEO, a developer lab adjacent to the room was added along with a sign-in area that holds mobile devices and a dedicated machine that can be used for room control. The room itself, with the Interactive Mural, can be seen in Figure 2.

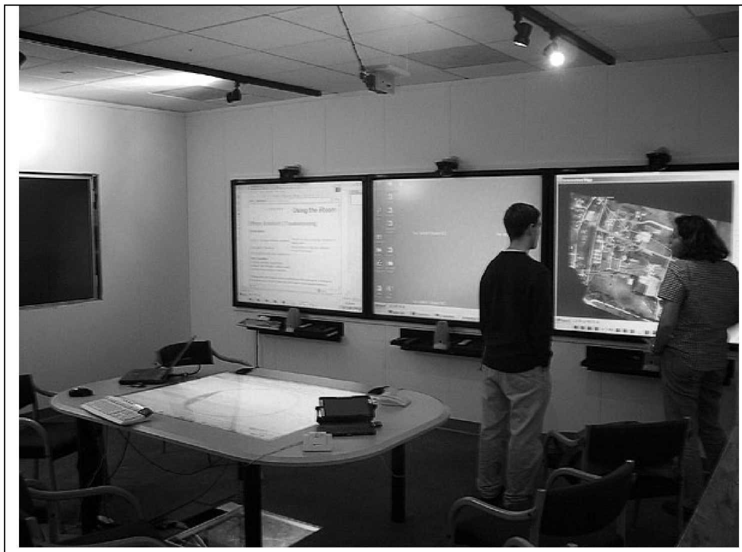


Figure 2 - A View of the Interactive Room v2 (iRoom). The hi-res Mural, visible at far left, is mounted in a wall that was used as a passive projection surface in our original iRoom. The orientation of the space was also rotated 90 degrees relative to the original iRoom, requiring compact light-folding optics for the three SMART Board rear-projected displays.

Most of the hardware in the room is standard commercial equipment, the major exceptions being the Interactive Mural and the iTable (the physical table, not the projected computer display). All displays other than the Interactive Mural are driven by standard Windows 2000 PCs. The room also has 802.11b wireless support for laptops and hand-helds.

For experiments with telecollaboration, (Virtual Auditorium [13]) and meeting capture (WorkspaceNavigator [35]) there are three video cameras, one above each board, plus others with a general room view. Additional devices such as the wireless receiver for iStuff devices and the barcode scanner for RedBoard have also been deployed in the room.

From the time it was completed in 2001, we have been using the iRoom as an everyday workspace for our own research groups, for a number of collaborations with application development groups, for courses and student projects, and for structured experiments.

2.4 The Proliferation of Interactive Workspaces

Since the building of iRoom v2, Interactive Workspaces technology has been deployed at many more locations around the Stanford campus including a prototype construction trailer setup at CIFE, a future classroom prototype at the Stanford Learning Lab, a classroom used for joint critiquing in the Program on Writing and Rhetoric, and a number of classroom spaces in a new Stanford building for innovation in education, Wallenberg Hall. We have deployed interactive workspaces with the WorkspaceNavigator system in two project-based engineering design courses, in which each project team has a dedicated project space in the engineering building. The TeamSpace system [47], a portable, robust iROS installation, has been tested in a common access area of the undergraduate library and is being deployed to a number of other settings. It is based in part on joint work with the Hewlett-Packard Laboratories on a "meeting machine" [8].

Beginning in 2001, we collaborated in the *iSpaces Project* with research groups at KTH in Stockholm, under the sponsorship of the Wallenberg Global Learning Network. During the subsequent three years we cooperated on a number of projects, including installing iROS-based workspaces at several locations in Sweden and jointly conducting usability studies. We have provided open source versions of our software to the research community and it has been deployed in a number of additional sites in Sweden, Switzerland, Finland, and the U.S. The iROS system is available as open source for download and installation with Windows 2000 and Mac OS X installers for both servers and clients [2]. A startup company, Tidebreak [3], has been created to explore commercial versions of iROS-like software.

3 Details of Fluid Interaction

3.1 Interaction with shared displays

The initial motivation for the iRoom was to take advantage of interaction with large high-resolution displays, such as the Interactive Mural [22, 26, 54]. Through a series of prototypes we have explored means for interacting directly with a wall-mounted display (rather than controlling it remotely from a laptop or other machine). The attention of a presenter or facilitator in a meeting is focused on the contents of the board and on the other participants. Any use of a keyboard is a distraction, so we have designed methods for direct interaction with a pen-like device and with direct touch on the board. The primary experiment in this area was the PostBrainstorm system, which used a large high-resolution (9 megapixel) display to facilitate design brainstorming sessions [22]. It was tested in actual use by a team of professional designers from IDEO and introduced a number of interaction innovations, including menu selection (FlowMenu [23, 25]), spatial information management (ZoomScape [24]), and integration of handwriting, sketching, and 3D manipulation using a pen as the only input device.

Using a bottom-projected table (see Figure 3), we developed an experimental interface for organizing images, such as photographs, and experimented with different visualizations and affordances for common actions, such as creating piles and browsing collections [21]. In other experiments, using a DiamondTouch [18] table, a top-projected display with a touch-sensitive surface which accepts simultaneous input from up to four people, we have explored a number of issues about the tradeoffs between shared and private information [40].

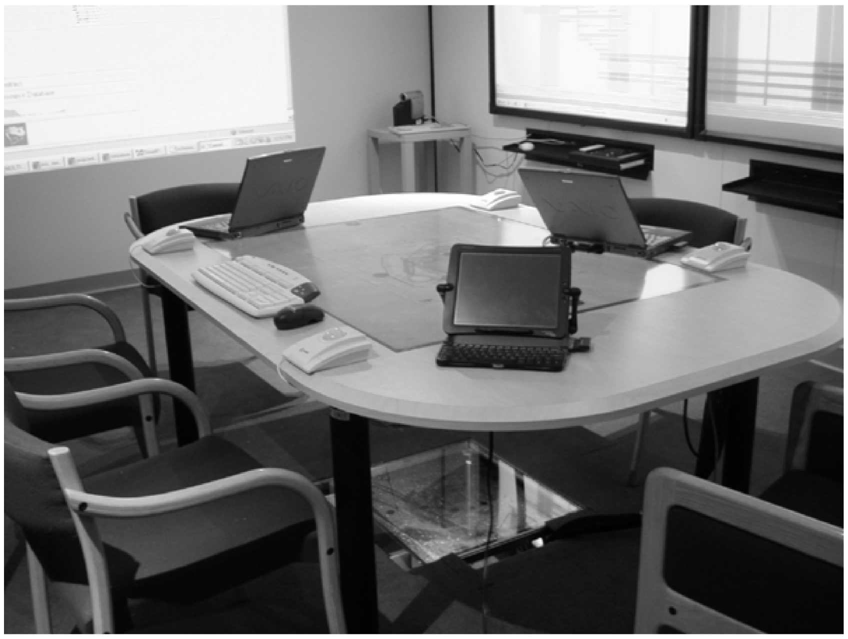


Figure 3 - The iTable, a bottom projected table used to experiment with table top UIs

Although most of our work has focused on co-located collaboration, one experiment has explored the potentials for tele-presentation of lectures using technologies that provide a high degree of presence with relatively low bandwidth and computation. The Stanford Video Auditorium [13] (see Figure 9) allows an instructor to see dozens of students on a tiled wall-sized display and establish eye contact with any student, with telephone-quality audio and television-quality video.

3.2 Overface

In providing software for an open interactive environment, we faced a fundamental tradeoff in interaction design. On the one hand, we are committed to accepting the diversity of devices, operating systems, and legacy applications that people bring to the space. On the other hand, we need to keep the interface simple, or people will not use it. Spaces such as the iRoom are intended to be used intermittently by people with little explicit training, who are engaged in a variety of interactions with other people during the meeting. This is very different from a "heads-down" desktop application that receives long training and full attention focus for many hours at a span.

The overface needs to be uniform across devices and provide several core functions from any of the devices in the room, including PDAs, laptops, and any of the wall-mounted or table-top displays:

- Moving information of all kinds from anywhere onto any of the display surfaces
- Controlling applications running on any of the display surfaces, from any device.
- Controlling the environment (lights, projectors, display sources, etc.)

A number of other projects have proposed mechanisms for information mobility [9, 44, 49, 51] and for portability of control [41]. As part of our iROS environment [42] we have developed a suite of related applications, including Multibrowse, the room applet, PointRight, and InterfaceCrafter, which are described further in Section 4.2.

3.3 PostDesktop devices

A key aspect of emerging ubiquitous computing environments is the variety of input devices and modalities that work together. In an interactive workspace it can be confusing to have physical input devices associated with specific machines. We are all accustomed to thinking of output devices, such as printers, as networked resources accessible from any machine, and this should apply to inputs as well. In addition to the pen-based and touch-based interfaces described in the previous sections, we have experimented with a variety of physical and tangible devices.

In order to facilitate prototyping of these interactions, we developed a protocol and set of sample devices, such as sliders, orientation sensors, audio output, and even a stuffed "*iDog*" as part of a project called *iStuff* [6]. These devices can post events on a shared EventHeap (see Section 4) for any program to use, and can therefore be easily adapted to different functions. The iStuff infrastructure is a first step towards general human-centered interaction architecture [53], in which the architecture is centered around people and the physical devices they encounter, instead of being tied to the implementation of processes, drivers, and the like. iStuff devices were tested in a number of experimental projects, including *iPong* (a multi-screen version of the original Pong video game), *iClub* (an interactive disco application), and a collection of simple one-button devices that could be associated with arbitrary actions through a web-form interface. For example, a push on a particular button can bring up a set of pre-designated applications on multiple devices in the room, to set up a meeting context.

In our experiments with sketch-based brainstorming, we discovered that designers wanted to be able to draw sketches independently and then “post” them when finished to a digital group surface. In an initial experiment we gave each of them a tablet computer (Vadem Clio) on which they could enter drawings directly. They found that compared to their standard sketching practices with pens and pencils on paper, the feel of drawing on the tablet was highly disagreeable. We decided instead to make it easy to bring hand-sketched material to the shared surface, getting rid of the interaction overhead of standard flatbed scanners. To enter visual material of any kind (including snapshots and physical objects), the material is placed on the table, and a pair of retro-reflective L-shaped crop marks is positioned to indicate the cropping boundaries. A command to our *FlowScan* software causes the ceiling-mounted digital camera to take a picture and transmit a JPEG image to the indicated display. Compared to ordinary scanning, the resolution is low (100 dpi) and there is no control of contrast, color, etc. However, the intended use for bringing materials into a meeting is well served and the simplicity of the interface makes it much more usable than one that requires distracting interactions.

In addition to the optical scanner, we have introduced other devices such as a bar code scanner to implement a system for personal information in shared spaces, similar to IBM's BlueBoard [46]. When the barcode scanner posts an event due to a user scanning a personal barcode, the application checks a table of codes registered to individual iRoom users, and if there is a match, it posts a portal to the user’s personal information space on one of the large electronic white-boards. In work with iROS in the iLounge at KTH, Swedish researchers developed *Magic Bowl*, a tangible user interface for controlling the interactive workspace [17]. The Magic Bowl makes it possible to quickly start the interactive workspace with a personalized configuration, using tangible RFID-based tokens to represent different configurations associated with particular groups and users.

4 Details of Software Infrastructure

4.1 Sharing Resources: iROS Meta-Operating System

For any real world system to support the modalities and characteristics described in the beginning of this chapter, systems infrastructure and human interface issues must be looked at together—the two are inextricably tied. The system structure and API need to reflect the way that applications written on top of it will be used.

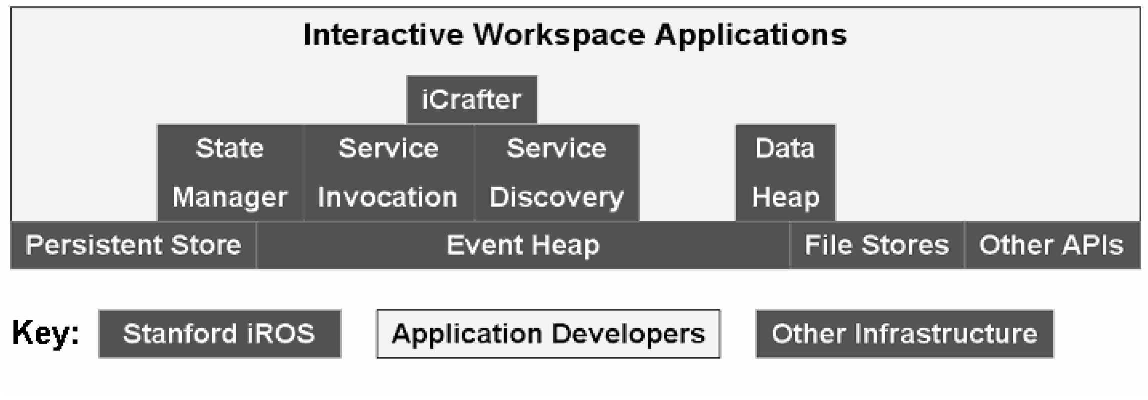


Figure 4 - iROS Component Structure

Figure 4 shows how the major iROS components fit together. The only component common to all iROS programs is the *EventHeap*, which is the underlying communication infrastructure for applications within an interactive workspace.

4.1.1 EventHeap

Given the heterogeneity in interactive workspaces and the likelihood of failure in individual devices and applications, it is important that the underlying coordination mechanism decouple applications from one another as much as possible. This encourages applications to be written to be less dependent on one another, thereby making the overall system less brittle and more stable. The EventHeap coordination infrastructure for iROS [29, 31, 34] expands on the tuplespace model [11] to provide inter-application coordination. The basic operations are *put,* which posts an event, and *get,* which queries for the existence of an event based on a template that specifies required fields and constraints on their values. Event subscription is also provided, which allows for publish/subscribe semantics: clients can receive a callback when events matching a particular template are posted to the EH. Various libraries and other software components allow EH clients to be written in Java, C/C++, Visual Basic, Perl, Python, and other languages; servlets allow Web-based clients to post events to the EH as well. The EH itself is written in Java and currently implemented as a centralized server process; elsewhere we have discussed why there appears to be limited benefit to a distributed implementation [31]. Initial versions of the EventHeap were built on top of TSpaces [56], but we have moved to our own code base for better performance and cross platform support.

EH semantics differ from the original tuplespace semantics in a few important ways. First, every tuple (unordered set of attribute-value pairs) carries a required *type* field as one of the attribute-value pairs; the presence and interpretation of any remaining attributes are application-specific and determined by the type. Second, events from the same source are sequenced, so that applications can get and handle events one at a time in submission order, much as they would with an event queue. All applications see the same ordering of events from a particular source, but there is no guaranteed ordering of events from different sources. Third, there is built-in support for routing or receiving events from specific clients, applications, devices, people or groups, via event attributes that are automatically filled in by the EH client libraries rather than having to be explicitly set by the client application. Finally, events automatically expire and are removed from the EH after their expiration time has passed; expiration times are set by the entity posting the event.

In keeping with the Boundary Principle[36], a single EH is the locus of interaction for a single ubiquitous computing environment, and a service or device can participate in that environment if and only if it can communicate with that environment's EH. In keeping with the Volatility Principle [36], automatic event expiration sidesteps the resource-reclamation problems that might arise if intended event recipients are crashed, hung, etc.; combining expiration with announce-listen beacons for stateful services allows any component of an interactive workspace, including the EH itself, to be recovered by simply restarting it. Of course, there is no guarantee that restarting something will fix the problem, but given frequent transient bugs in off-the-shelf applications, resource issues such as memory leaks, temporary network failures, etc., restarting can be surprisingly effective, and our design ensures that it is safe to try [20]. All of this is discussed in greater depth in [29, 30].

4.1.2 The DataHeap.

The EH is designed for the exchange of small control messages and by design its contents do not survive crashes. We need a separate facility for persistent state storage and large objects. The DataHeap allows naming and storage of data by attribute-value pairs, including the representation or format of the data (e.g. GIF vs. JPEG image). When a client requests data and indicates which formats it can consume, the DataHeap automatically transforms the data to the suitable format if possible. This makes integration of new media types into existing applications modular, when adapters can be provided.

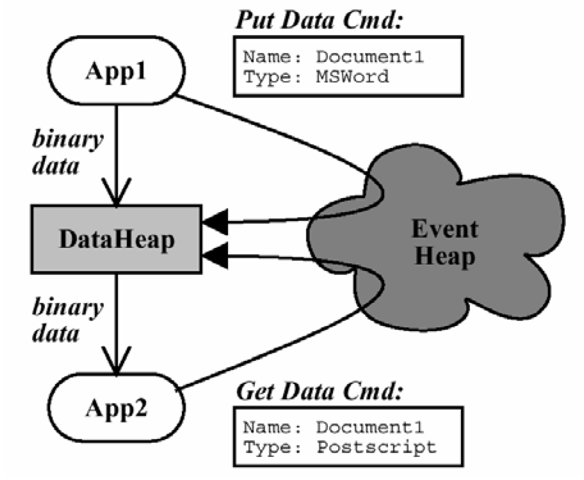


Figure 5 - The DataHeap

4.1.3 The iROS Manager.

The *iROS Manager* user interface provides several functions including selectively restarting groups of applications. One of the iROS Manager components facilitates software packaging and distribution for iSpace applications and works with an iSpace *Dependability Manager* to keep an iSpace running even when some of the component applications fail. iROS Manager encapsulates dependency information among iSpace applications, which results in "single-click" installation for administrators and "near-zero administration" for dealing with the day-to-day operational issues that inevitably arise in installations of many heterogeneous components.

4.1.4 iROS Design Principles

Some common principles run throughout the iROS system:

Decoupling to Make System More Flexible: Applications do not communicate directly with one another, but use indirection through the EventHeap, which improves fault isolation. iROS systems decouple applications referentially: Rather than routing events based on unique identities of devices, processes, or sockets, they use attributes for posting and retrieval (as in [4]). In addition, the EventHeap and DataHeap also decouple applications temporally: programs need not be running at the same time to communicate. Applications that transiently fail can retrieve non-expired events they missed while they were down.

Application Ensembles: Applications written in the iROS system are not a special-platform monolithic system (as in Gaia OS [12] or BEACH [50]) but are bound together into a dynamically changing ensemble as users launch and use them. The EventHeap facilitates this by allowing applications with common event types to coordinate regardless of the machine on which they are running. Using the DataHeap, applications that use different formats can exchange data.

Modular Restartability: In our design, failure is treated as a common case, so when something breaks it can simply be restarted. Clients automatically reconnect when started, so the EventHeap server, interface manager and DataHeap server can all be restarted in a room full of running devices and applications without interfering with their function. The net result of this is that any subset of machines in the workspace can be restarted in any order. Any important state that might be lost during this process is either stored in persistent form in the DataHeap, or is beaconed as soft-state which is regenerated as clients come back up.

Simplicity: Rather than trying to provide a large API that handles many different situations, we provide simple APIs with minimal client-side overhead. Wherever possible we rely on existing technologies like the Web, since HTML serves as a baseline presentation an UI layer. This makes the code base small for impoverished devices (our current Java JAR library for all of the iROS core functionality is less than 200KB), and also simplifies the task of porting the client interface to new devices. This simplicity comes at the expense of more sophisticated functionality, such as atomic transactions, total ordering, etc. Our experience suggests that these features should be provided at a higher level for those applications that need them, but that many of the smaller and simpler (yet very useful) applications and behaviors we and our collaborators have built so far have not required them.

4.2 Supporting Overface: Common end-user tools

Each of the iROS installations has had its own specific interfaces and applications. Within these, there is a small common core of facilities:

4.2.1 Multibrowse

Multibrowse [33] allows users to flexibly move and open web content and application documents across the devices in a workspace. For web pages there is an Internet Explorer plug-in that submits web page display requests and a daemon program that displays the appropriate web page on the requested device upon reception of the requests. For application files, the daemon can use the DataHeap or other shared server to copy the file to the target device and then send commands to open it. We have experimented with a variety of tools for specifying the destination of a MultiBrowse, including drop-down lists and room maps. See the discussion of InterfaceCrafter below for more detail.

4.2.2 PointRight

When we began conducting meetings in the prototype iRoom, it became clear that users wanted mixed control of the large displays. At some moments, a person standing at the display needs to control it directly, using touch, pen, etc. At other moments, a person sitting elsewhere in the room wants to perform an operation on that display, which may be as simple as bringing a window to the front, or typing a URL into a browser. The flow is disrupted by asking the person at the board to do it, or by getting up and walking to the board. A number of previous systems have dealt with multi-user control of a shared device, often providing sophisticated floor-control mechanisms to manage conflicts. In keeping with our “keep it simple” philosophy, we created a mechanism called PointRight [32], which provides the key functionality without being intrusive. PointRight provides an intuitive model for moving control among displays in an interactive workspace and has proven to be one of the most useful iROS applications.

With PointRight, any machine’s pointing device can become a “super pointer” whose field of operation includes all of the display surfaces in the room, as well as the machine it is on. Rather than require configuration by the user, we take advantage of the spatial visibility of the room.

When a device runs PointRight, the edges of its screen are associated with the edges of other displays. The user simply continues moving the cursor off the edge of the local screen, and it moves onto one of the other screens. The keyboard is mapped to whatever screen the cursor is currently on, providing full application control. The system tracks projector on/off state, machine-display connections, and controllable machines through the posting of events to the EventHeap by the room controller system and PointRight clients. A beaconing mechanism that takes advantage of the automatic expiration of events from the EventHeap insures that machines that go down or are removed from the workspace are also removed from the list of valid control targets.

The PointRight mechanism does not do anything special about simultaneous access. Pointer input is simply fed into the target machine's event queue as absolute cursor positioning events. This means that when two users are moving the pointer on the same screen, the effect is the same as it would be for two absolute physical pointing devices attached to the machine driving that screen. In normal Windows operation, a ghost cursor appears for each user as the system time interleaves the cursor display. Although this would be problematic in a distributed setting, we have found that it causes little problem in a physical shared setting, where people directly see the effects of their actions and can negotiate with others for control in the rare cases when that is necessary. In addition to laptops that are brought to meetings, the iRoom has a dedicated wireless keyboard and mouse running PointRight, which can be used as a general keyboard and pointer interaction device for all of the surfaces.

4.2.3 Room Controller

One obvious advantage of working in a room-based environment is that people share a common model of where devices are positioned, which they can use as a convenient way of identifying them. Our "room controller" (see Figure 6) uses a small map of the room to indicate the lights, projectors, and display surfaces. These can be controlled by simple toggles and menus associated with the objects in the map. The room controller can be used to switch video inputs to projectors as well as to turn lights and projectors on or off.

Initial versions of this controller were built as standard GUI applications, which could only run on some systems. We broadened their availability to a wider range of devices by providing them as web pages (using forms) and as web applets (using Java). Our later research generalized the process further with InterfaceCrafter (see Section 4.3.1), which provides for automated generation of interfaces for any set of services, on any device that can support one of a variety of interface languages (Java Swing, HTML, WABA, etc.).

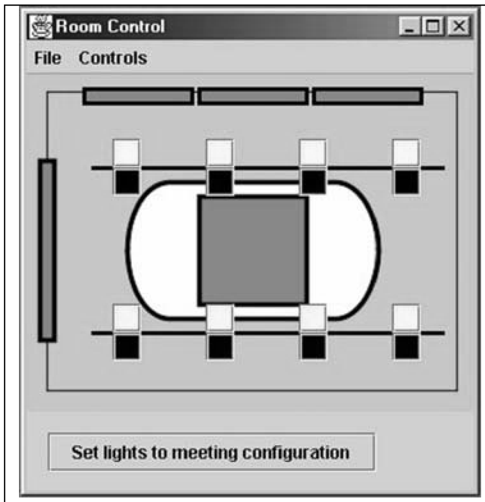


Figure 6 –Java Swing Room Control UI generated by Interface Crafter. The grey rectangles are surfaces onto which information can be dragged. The yellow/black rectangles are light switches corresponding to the array of track lights on the ceiling.

In addition to providing environment control, the same interface serves as a convenient way of moving information onto displays. The user drags an information object (URL or file icon), onto the appropriate region in the map to indicate the display on which it should appear. The MultiBrowse mechanism [33] is used in conjunction with the DataHeap to initiate an application on the target device that displays the indicated object. The room control system stores the geometric arrangement of screens and lights in the room in a configuration file.

4.3 Supporting PostDesktop: Interface development tools

In addition to the standard facilities we want to be able to provide end-user interaction mechanisms that are not specific to the devices in the room. We have several developed tools to provide for the mapping from input modalities to effective action.

4.3.1 InterfaceCrafter

Rather than hardcode an interface such as the Room Controller for a specific device, InterfaceCrafter [57] provides a way to specify the room geometry and automatically generate a visual controller tailored to the controlling device (e.g., a laptop or a PDA). For example, the interface of Figure 6 is provided to any device supporting a Java Swing UI. If an interface is requested for a standard web browser, e.g., a laptop without iROS client software, a simplified controller using only HTML forms is provided instead.

InterfaceCrafter exploits beacon events for service advertisement: a service's beacon contains an XML-based description of the methods callable in that service, and separate "interface generators" can represent those methods into appropriate human interfaces in a variety of formats (Java Swing, HTML, VoiceXML, etc.). Generators can be service-specific, device-specific, both, or neither; InterfaceCrafter attempts to select the most-specific generator in any given scenario. We have developed tools that can automatically create generic but functional InterfaceCrafter beacons from any Win32 application using COM, and from many Java applications using method introspection; the resulting automatically generated user interfaces are not elegant, but they are functional and suitable for rapid prototyping, and they can be replaced by more customized versions as needed.

4.3.2 iStuff and PatchPanel

The iStuff [6] project includes a toolkit of physical devices and a flexible software infrastructure to simplify the prototyping of novel interaction techniques. The toolkit leverages the iROS infrastructure by adding fields to the protocols that are specific to device input and coordination. The toolkit allows developers to rapidly integrate inexpensive "generic" physical interactors such as buttons and dials into applications. We have experimented with a variety of simple wireless devices, as shown in Figure 7.

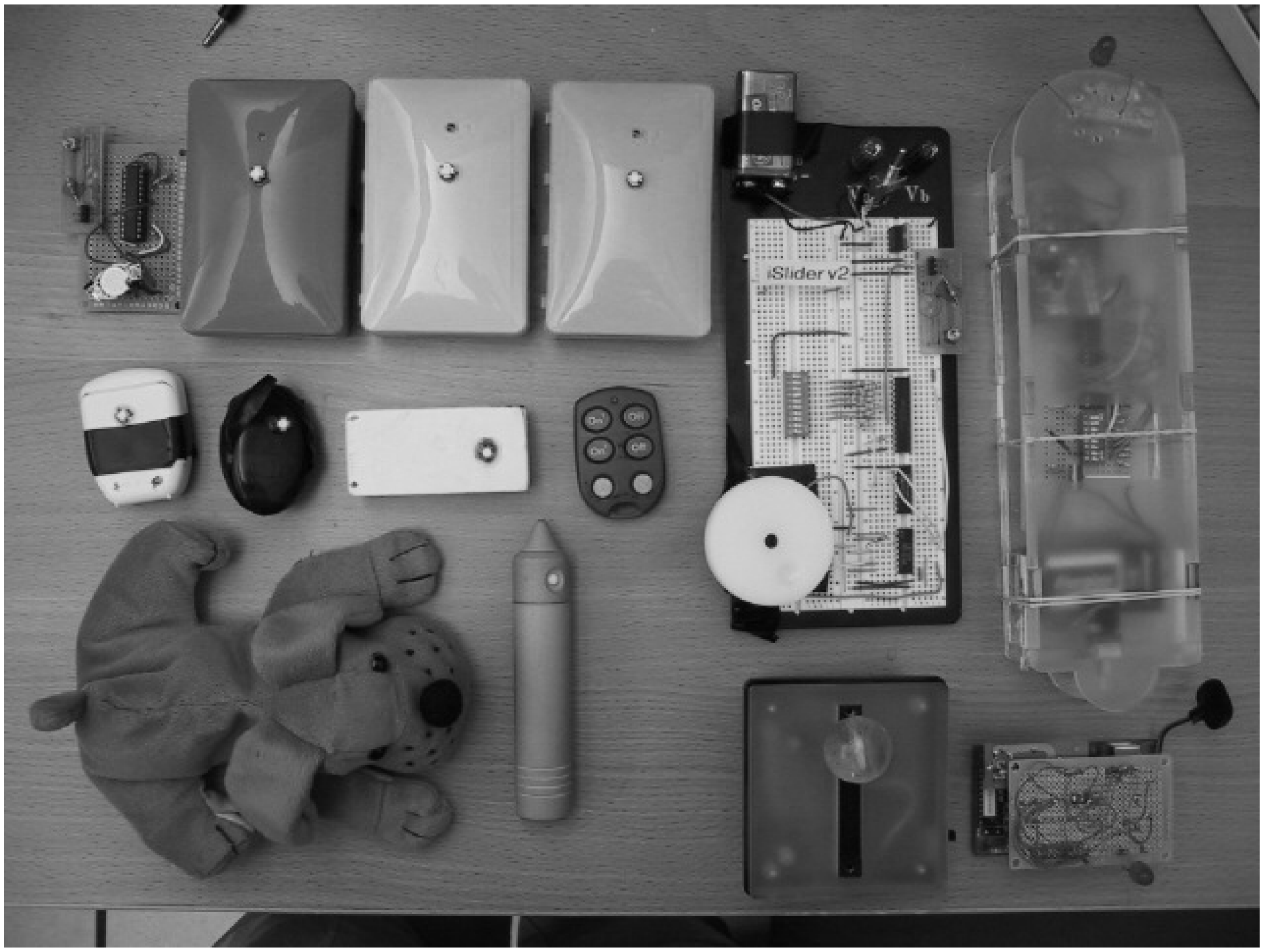


Figure 7 - iStuff Devices

The *PatchPanel* [7] simplifies the dynamic mapping of devices to iSpace behaviors, enabling them to be composed into more sophisticated or even multimodal controllers, and to be programmed to trigger "macro" sequences of actions. It takes advantage of the *spatial decoupling* provided by the EH. Specifically, since all events must pass through the EH on their way from a sender to one or more receivers, we can intercept those events and rewrite them, copy them for logging, etc. The PatchPanel subscribes to *all* EH events, allowing users to set up rules indicating how new events should be created in response to events observed being placed into the EventHeap. It supports simple translations, such as changing the event type and copying across field contents, as well as more complex stateful filtering operations, such as creating a new event with average values over several primitive events. Although each device can only output (or input) a fixed repertoire of events, PatchPanel translation can be used to connect the 33devices to existing behaviors or to combine devices to create multimodal UI's.

5 Applications and Empirical Studies

Space does not allow for details of all of the experiments, so we will present some notable examples and results.

5.1 CIFE Suite: Example of Dynamic Application Coordination

In addition to manual control, we use the EventHeap to link applications for cross-application control. Through submission of events to the EventHeap, interface actions within one application can trigger actions within another running on any of the machines in the workspace. This has been employed in a suite of applications developed by The Center for Integrated Facility Engineering (CIFE) [37] for use in construction management meetings. The software provides a set of viewers that run on the various displays in the workspace:

- A construction site map displayed on the iTable, which allows the selection of various view points in the construction site and then emits an appropriate view change event. Users can click on regions of the map to control the point-of-view (POV) of 3D CAD models of the construction site being displayed by other applications in the workspace. Since maps are relatively orientation independent, users can interact with the map regardless of where they are sitting at the table
- A "4D" viewer that shows a time-sequence 3D model of projected state of the construction site for any date during construction. It responds to events that change the view, select objects and zones, and change the date for the current model view.
- A web based viewer that displays tables of construction plan information. This viewer also emits zone and date selection events when table information is selected and listens for the same events to highlight information in the table.

All of the applications are essentially standalone, and communicate through the EventHeap. The 4D viewer was originally designed for use on a single PC and was modified to use the EventHeap by adding around 100 lines of code. Since the viewers use common event types, the various components of the suite retain their ability to coordinate while being displayed on any screen in a workspace. Since the components are loosely coupled, the absence or disappearance of an event source or event sink does not affect any of the application components currently in use. The overall effect for the user is that clicking on an element in any one of the applications produces a corresponding change in the other applications running in the workspace.

5.2 PostBrainstorm

The PostBrainstorm interface on the interactive mural provides a high-resolution display with the ability to intermix direct marking, control of images and 3D renderings, and arbitrary desktop applications. The details are given in other papers [22, 24, 25] and will only be briefly described here. The key design goal was to provide “fluid interaction,” which does not require focused attention of the user, who is typically focused on person-to-person interactions in a meeting. This goal led to the development of several new mechanisms:

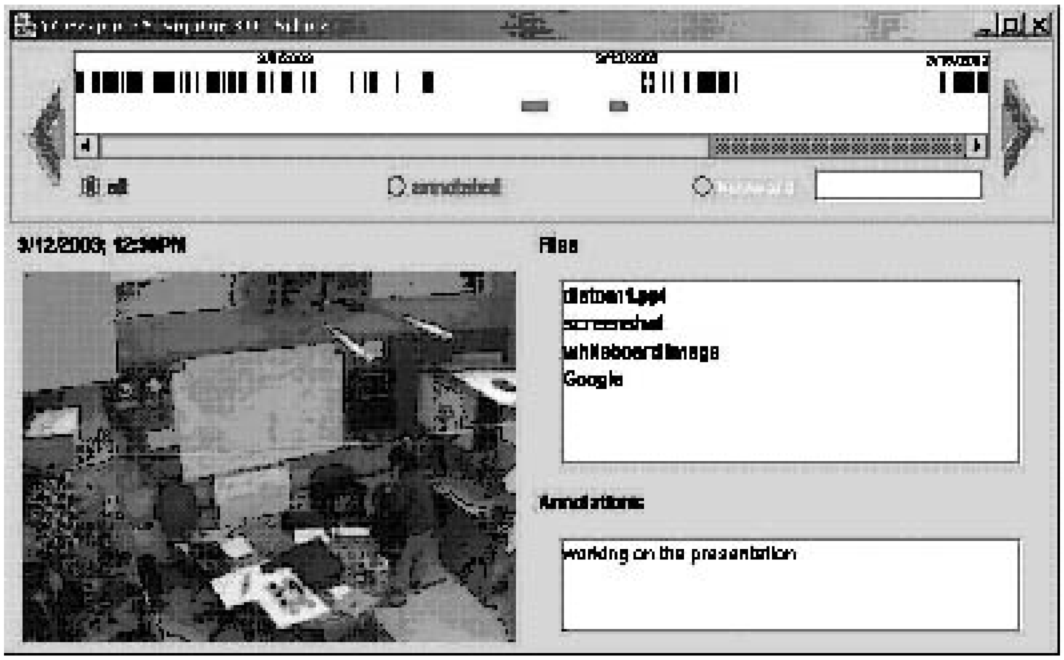
- **FlowMenu:** a contextual pop-up menu system that combines the choice of an action with parameter specification in a single pen stroke. This makes it possible to avoid interface modes, in which the first part of an action leaves the system in an altered state, which affects the interpretation of subsequent actions. Modes lead to confusion and errors for a user who is not paying full attention (see [43] for discussion). Because the menu is radial rather than linear, multi-level operations can be learned as a single motion path or gesture, so an experienced user does not even need to look at the menu to select an action [23].
- **ZoomScape**: a configurable “warping” of the screen space so that the visible scale of an object is implicitly controlled by where it is moved. The object retains its geometry while being scaled as a whole. In our default configuration, the top quarter of the screen is a reduction area, in which objects are one-quarter size. An object can be moved out of the primary attention area and reduced all in one pen stroke, with a smooth size transition as it goes through a boundary area. This provides a simple mechanism for screen real-estate management without requiring explicit commands to change size, iconify, etc.
- **Typed Drag-and-Drop**: Handwriting on the screen is recognized by a background process, retaining the digital ink and annotating it with the interpreted characters. Through FlowMenu commands, a sheet of writing can be specified to have a desired semantics (e.g., the name and value of a property to be associated with an object) and then dragged onto the target object to have the intended effect. This provides a crossover between simple board interaction (hand drawn text) and application-specific GUI interactions.

The overall system was tested in actual use by several groups of industrial designers from two local design firms (IDEO and SpeckDesign). Their overall evaluation of the facility was quite positive [22] and provided us with a number of specific areas for improvement.

In addition to experimenting with these facilities on the high-resolution interactive mural, we have ported them to standard Windows systems, and have made use of them on the SmartBoard screens.

5.3 Workspace Navigator

The WorkspaceNavigator is a suite of iROS-based tools to support the capture, recall and reuse of material and ideas generated by a group doing semi-structured work in an interactive workspace. Our focus is on the capture of digital information, including screenshots, files, and URLs. This automatically captured information can be augmented by user-supplied annotations both during and after a session. All of the information is stored as a sequence of time slices, integrated by an overview image of the physical space at the time of the capture. The overview image provides spatial cues for accessing the captured information, and is treated as an image map with links to the other information captured during that time slice. We have conducted two user studies of the WorkspaceNavigator tools and found that capturing coordinated slices of digital information is useful for recall and summarization activities, and that coordinating access through the visual metaphor of the overview image is understandable and effective [28, 35].



3/12/2003; 12:30PM

distant.ppt
screenshot
whiteboard image
Google

working on the presentation

Figure 8 - Snapshot-based browser for Workspace Navigator

5.4 Virtual Auditorium

The Virtual Auditorium [13] is a system for teleconferenced remote teaching, with a central instructor node and up to a few dozen student nodes at other locations. The instructor node in the initial prototype uses the three large wall-display computers in the iRoom, the interactive table, and the cameras mounted above the wall displays. Student nodes are workstations with simple video cameras. All nodes are connected by high-speed computer networks.

The conceptual usage model of the Virtual Auditorium is that all participants can be seen and heard with minimal latency at all times. The instructor can see the remote students on the display wall at roughly life size, shown in a grid of positions shown in Figure 9.

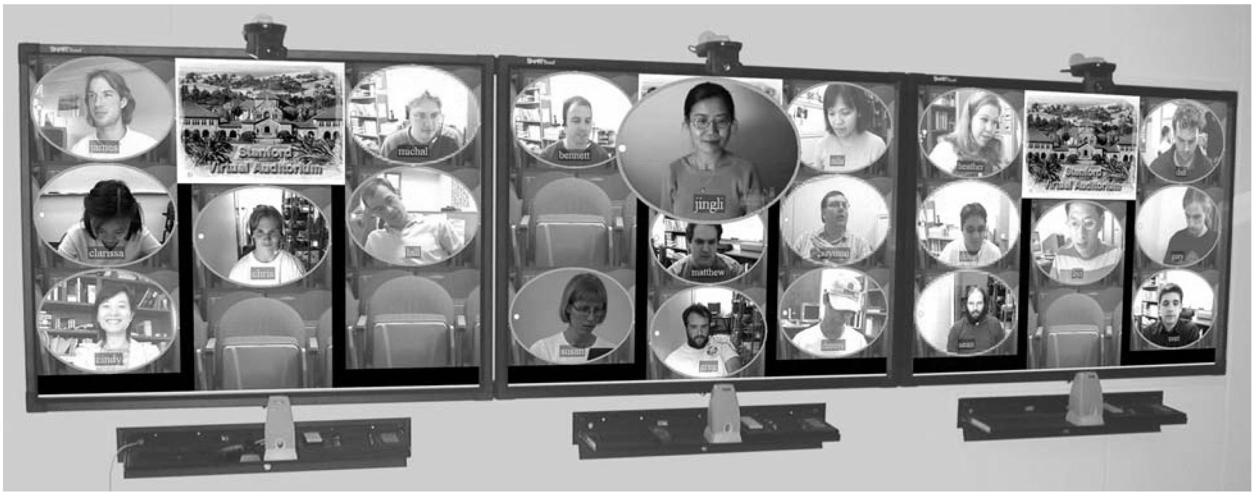


Figure 9 - Virtual Auditorium Display Wall

The instructor can alter the seating arrangement by dragging a student's video to any empty seat using PointRight. Each of the three sections of the display wall has a speaker, enabling the instructor to locate the general direction of the speaking student from the location of the loudspeaker. Each student's instantaneous audio volume is displayed next to his name to enhance the visual signal of lip movement; thus, allowing the instructor to easily locate the speaking student.

The design was particularly optimized to deal with issues of eye contact, which remains one of the key problems in remote video applications. In situations with more than a pair of communicating speakers, eye contact is a fundamental cue as to where a speaker's attention and communication is directed. In a teleconferencing system, gazing at the picture of the recipient does not generally cause the recipient to see an image of the speaker looking at him or her (which would require looking into the camera, instead). Also, with a single camera, every recipient sees the same view, eliminating the cue of who is being addressed. In the Virtual Auditorium, the instructor can establish eye contact with any one student, a group of students, or the entire class using a technique called directed gaze.

5.5 Teamspace

A typical Stanford undergraduate student spends only 15-18 hours a week in the classroom. Far more time is spent in technology spaces such as those in libraries and residences, where learning, research, and course project collaboration take place, primarily using student-owned laptops. The Teamspace packages key iRoom functionality in a simple "zero-configuration" download for students and a near-zero-administration server installation designed for "walk-up" use by small student groups. Open casual public workspaces such as Teamspace require security and access control that does not require pre-registration and yet prevents unwanted incursion or snooping. The key element is physical co-location: people who are in the space should have access, while those who are not currently there should not. We address this through line-of-sight login and session expiration. A person wishing to log in to Teamspace must enter a server-generated password that appears on the group display, thereby restricting login to people who can physically see the group display. Once a session ends or times out, users who wish to continue must login again with a new password to continue the session. These simple mechanisms combined with social protocols appear sufficient for adequate access control.

We conducted user studies on fifty undergraduate students representing a wide range of technical and non-technical majors. Details are in [47]. In one study, groups were arbitrarily formed and given a specific group task. We found that users tended to work relatively independently, collaborating at the beginning to divide the work and at the end to compile the work. During these moments of collaboration, they tended to multibrowse documents to the shared display (but generally not from the shared display to their own screens), with a single individual using PointRight to control the main display. Often, the choice of this individual was determined by pointer contention early in the session. In the second study, existing student teams working on real group projects used Teamspace for their collaboration. These participants, sharing a common external purpose and already familiar with one another, were more apt to collaborate, not only on the project at hand but also in learning the Teamspace technology. They tended to multibrowse throughout the session both to the public screen and to one another's laptops, and felt more comfortable vying for the group screen pointer, resulting in increased collaboration on the group display and more frequent switching between private and public workspaces.

6 Future Directions

6.1 Interaction

Much of the initial work emphasized wall-mounted touch-screen displays. We have more recently been incorporating table-based displays, including ones that are able to handle multiple touches and distinguish users, and have added audio modalities. We are exploring the ways in which the boundaries between shared and private information and activity in cooperative tasks are affected by the choice of technologies.

Also, we have begun an investigation from a theoretical perspective of how users move information and control in an interactive environment [55]. We are exploring a variety of mechanisms from selected existing real-world and research systems and have built an experimental platform in Java that we call the iWall, for experimenting with alternative ways to move objects across multiple displays. Our research interests lie in the area of fluency: which methods of moving data do users find easier, simpler, or more natural than others, and why?

The cognitive theories most generally applied to HCI deal with task-related measures (speed, error, etc.). In an interactive environment we need situation-related measures, such as the degree to which computer use interferes with other activities. This can be across people (my action interferes with yours) or within task (e.g., having to figure out how to move a visual image interferes with my work in using it, or in explaining it to you). Some modes of interaction may be significantly better or worse in this dimension, and we want to understand what determines a user's fluency, or lack thereof, with an interface.

6.2 Infrastructure

While our system tries to minimize the amount of time required to integrate a device into an interactive workspace, there is still overhead in configuring room geometries and specifying which servers to use. We plan to make it simpler to create and extend a workspace and to move portable devices between them. Users should only have to plug in a device or bring it into a physical space in order for it to become a part of the corresponding software infrastructure. User configuration should be simple and prompted by the space—for example the user might be requested to specify where in the room the device is located. The logical extension of this is to allow ad hoc interactive workspaces to form wherever a group of devices are gathered. A group of laptop computers brought together by field engineers could, for example, automatically be used as an iRoom-to-go.

So far, we have focused primarily on co-located collaboration. As we continue, we want to support project teams in remotely located interlinked workspaces, facilitating coordination between desired applications while insuring that workspace-specific events remain in the appropriate location. For example, sending an event to turn on all lights should have effects only in the environment where it was generated. One of the design decisions in the EventHeap was to assume workspaces with fixed infrastructure, with the EventHeap server running on a permanent machine in the workspace. Given the ubiquity of laptops and other portable devices, however, it is quite possible to create ad hoc interactive workspaces.

EventSpaces is our next generation coordination system which will be a peer-to-peer system with mechanisms for defining workspaces and allowing devices to join and remove themselves. It will support bridging between different workspaces, and breaking off sub-groups within a single workspace. In the end we hope that the system will be sufficiently powerful to work both for ad hoc situations and in workspaces with fixed infrastructure, such as those currently supported by the EventHeap.

7 Final Words

As with all systems being built in relatively new domains, and particularly with systems that involve user interaction, it is difficult to come up with a quantitative measure of success. We have had a number of experimental uses, including:

- Design brainstorming sessions by professional designers
- Construction of class projects built on the iROS system
- Training sessions for secondary school principals
- Construction management experiments
- Group interactions in several Stanford courses, including Latin, English, Japanese and Archaeology
- Project groups from an interaction design course
- Casual team use in a public space
- and, of course, our own weekly group meetings.

The overall results have been positive, with many suggestions for further development and improvement. We have provided open source versions of our software to the research community and it has been deployed in a number of institutions. The iROS system is available for download and installation on Windows 2000 and Mac OS X for both servers and clients.

Comments from developers who have appreciated how easy it is to develop applications with our framework are also encouraging. Finally, the adoption and spread of our technology to other research groups (discussed earlier in the chapter) also suggests that our system is meeting the needs of the growing community of developers for interactive workspaces.

Acknowledgments

The work described here owes a great deal to many people, both at Stanford and at our partner institutions. In addition to all of our excellent students, we want to especially thank Pat Hanrahan, Maureen Stone, John Barton, Carl Gustaf Jansson, Bjorn Pehrson, and Brian Luehrs. The research was supported by grants from Hewlett-Packard, IBM, Intel, Mitsubishi Electronics Research Lab, the Stanford Learning Lab, and most substantially by the Wallenberg Global Learning Network.

iSpaces Systems Glossary

eBeam is a commercially available ultrasonic pen input device (from EFI, Inc.), which is used for the Interactive Mural and for other surfaces in the iRoom

EventHeap is a piece of software that is the basic system “glue” that enables ease of integration, ease of development, robustness

EventSpaces is a generalization of the EventHeap to provide flexible use of multiple spaces, including ad hoc spaces and spaces working together at a distance.

FlowMenu is an integrated interface mechanism for menu selection and parameter entry, designed for use on large direct-contact boards such as the Interactive Mural.

FlowScan is a system using iROS that allows user of the iRoom to enter pictures into the information space through an overhead camera with minimal interaction.

Interactive Mural is a large high-resolution wall-mounted display, of which several versions were developed in our research.

InterfaceCrafter is a general mechanism for producing interfaces based on service descriptions that are tailored to the devices on which the interface is to be used.

iRoom is the prototype room in the Computer Science Department at Stanford, in which we have done our primary experiments in interactive workspaces.

iROS is the umbrella term for the iRoom Operating System - the software that integrates all parts of our interactive workspaces.

iStuff is a collection of wireless input/output devices using the EventHeap to communicate easily with programs of all kinds

iTable is a table with bottom-projected computer display, used for experiments with interaction on horizontal surfaces.

iWall is an infrastructure that supports display and movement of information on multiple screens for experimenting with different affordances

MultiBrowse is an iROS application that allows any computer in a workspace to bring up materials on another computer by remote control

Overface is a general term for the collection of interaction mechanisms that we provide on top of the regular interfaces to the devices in the workspace

PatchPanel is a software component for integrating actions of devices that use the EventHeap

PointRight is an iROS application that lets any of the displays in a room be controlled from any laptop or pointing device

PostBrainstorm is an application of the Interactive Mural with a number of innovative interaction mechanisms designed to facilitate graphic brainstorming

Redboard is a distributed access system that makes it easy for a person or group to bring into a workspace environment materials from their online files, and to transmit materials to others.

SmartBoard is a commercial touch-screen device for large displays (from Smart Technologies), which is used on many of our experimental display surfaces.

SmartPresenter is an iROS application that allows lecture presentations to make flexible use of multiple screens.

TeamSpace is a simplified version of iROS with a minimal equipment configuration, intended for walk-up use.

Virtual Auditorium is a system for remote education that uses interactive workspace mechanisms for the instructor node, allowing management of eye contact and attention.

WorkspaceNavigator is a collection of multi-modal capture components that can be reused and deployed to reflect the needs of a variety of instructional situations

ZoomScape is a mechanism for managing the size and position of materials on a large screen, enabling its use as for activities such as brainstorming.

References

1. *The eBeam System*, . 2000, Luidia, Inc.: Foster City, CA http://www.luidia.com/.
2. *iROS Meta-Operating System*, . 2001-2004, Interactive Workspaces Group, Stanford University: Stanford, CA http://iros.sourceforge.net.
3. *Tidebreak, Inc.*, http://www.tidebreak.com.
4. Adjie-Winoto, W., *et al.*, *The design and implementation of an intentional naming system.* Oper. Syst. Rev. (USA), Operating Systems Review, 1999. **33**: p. 186-201.
5. Andersson, H., *et al.*, *iSecurity*. CSD 2003 Class Report, . 2003: Swedish Royal Institute of Technology (KTH) and Stanford University. 76. http://csd.ssvl.kth.se/~csd2003-team16/FinalReport/final_report.pdf.
6. Ballagas, R., *et al. iStuff: A Physical User Interface Toolkit for Ubiquitous Computing Environments*. in *CHI 2003: Human Factors in Computing Systems*. 2003. Fort Lauderdale, FL, USA: Association for Computing Machinery: p.
7. Ballagas, R., A. Szybalski, and A. Fox. *The PatchPanel: Enabling Control-Flow Interoperability in Ubicomp Environments*. in *2nd IEEE International Conference on Pervasive Computing and Communications (PerCom 2004)*. 2004. Orlando, FL, USA: IEEE: p.
8. Barton, J.J., *et al. The MeetingMachine: interactive workspace support for nomadic users.* in *Fifth IEEE Workshop on Mobile Computing Systems and Applications*. 2003. Monterey, CA, USA: Los Alamitos, CA, USA : IEEE Comput. Soc, 2003: p. p.2-12.
9. Bolt, R.A. *'Put-that-there': voice and gesture at the graphics interface*. in *SIGGRAPH 1980 Seventh Annual Conference on Computer Graphics and Interactive Techniques*. 1980. Seattle, WA, USA: Usa : 1980: p.
10. Brumitt, B., *et al. EasyLiving: technologies for intelligent environments*. in *Handheld and Ubiquitous Computing Second International Symposium HUC 2000*. 2000. Bristol, UK: Berlin, Germany : Springer-Verlag, 2000: p. 12-29.
11. Carriero, N. and D. Gelernter, *Linda in context (parallel programming).* Communications of the ACM, 1989. **32**(4): p. 444-58.

12. Cerqueira, R., *et al. Gaia: A Development Infrastructure for Active Spaces*. in *Ubitools Workshop at Ubicomp 2001*. 2001. Atlanta, GA: p.

13. Chen, M. *Design of a Virtual Auditorium*. in *Ninth ACM International Conference on Multimedia*. 2001. Ottawa, Canada: ACM Press: New York, NY, USA: p. 19-28.

14. Chen, X.C. and J. Davis, *LumiPoint: Multi-User Laser-Based Interaction on Large Tiled Displays.* Displays, 2002. **22**(1).

15. Coen, M.H., *et al. Meeting the Computational Needs of Intelligent Environments: The Metaglue System*. in *MANSE99 : 1st International Workshop Managing Interactions in Smart Environments*. 1999. Dublin, Ireland: p.

16. Croné, M. *Persistence in Interactive Workspaces*. in *Collaboration with Interactive Walls and Tables Workshop at UbiComp 2002*. 2002. Goteborg, Sweden: p.

17. Croné, M., *et al. Magic Bowl: a Tangible User Interface for Configuration of Interactive Environments*. in *6th International Conference on the Design of Cooperative Systems*. 2004. French Riviera, France: p.

18. Dietz, P. and D. Leigh. *DiamondTouch: a multi-user touch technology*. in *UIST'01: ACM Symposium on User Interface Software and Technology*. 2001. Orlando FL USA: New York, NY, USA : ACM, 2001: p. 219-26.

19. Edwards, W.K. and R. Grinter. *At Home with Ubiquitous Computing: Seven Challenges*. in *Ubicomp 2001*. 2001. Atlanta, GA, USA: p. 256-272.

20. Fox, A. and D.A.-S.U. Patterson, Stanford, CA 94305 USA AD - Stanford Univ, Stanford, CA 94305 USA, *Self-repairing computers.* Scientific American, 2003. **288**(6): p. 54-61.

21. Grant, K., *et al.*, *Beyond the Shoe Box: Foundations for Flexibly Organizing Photographs on a Computer*. Digital Libraries Report, 2002-45. 2002, Stanford, CA: Stanford University. .

22. Guimbretière, F., *Fluid Interaction for High Resolution Wall-size Displays*. Ph.D. Dissertation, Computer Science. 2002, Stanford, CA, USA: Stanford University. 140.

23. Guimbretière, F., A. Martin, and T. Winograd, *Benefits of Merging Command Selection and Direct Manipulation.* HCI Journal (in Press), 2005.

24. Guimbretière, F., M. Stone, and T. Winograd, *Fluid Interaction with High-resolution Wall-Size Displays.* UIST (User Interface Software and Technology): Proceedings of the ACM Symposium, 2001: p. 21-30.

25. Guimbretière, F. and T. Winograd, *FlowMenu: Combining command, text, and data entry.* UIST (User Interface Software and Technology): Proceedings of the ACM Symposium, 2000: p. 213-216.

26. Guimbretière, F., T. Winograd, and S.X. Wei, *The Geometer's Workbench: An Experiment in Interacting with a Large, Hgh Resolution Display*. Interactivity Lab Technical Report, . 2000, Stanford, CA, USA: Stanford University. 7. http://graphics.stanford.edu/~francois/Papers/UIST2000/geometerworkbench.pdf.

27. Humphreys, G., *et al.*, *WireGL: A scalable graphics system for clusters.* Proceedings of the ACM SIGGRAPH Conference on Computer Graphics, 2001: p. 129-140.

28. Ionescu, A., M. Stone, and T. Winograd, *WorkspaceNavigator: Capture, Recall and Reuse using Spatial Cues in an Interactive Workspace*. Technical Report, TR2002-04. 2002, Stanford, CA, USA: Stanford University. 8. http://www.stanford.edu/~arna/persist/wkspcNav-286.pdf Verified: 11/2002).

29. Johanson, B., *Application Coordination Infrastructure for Ubiquitous Computing Rooms*. Ph.D. Dissertation, Electrical Engineering. 2003, Stanford, CA, USA: Stanford University. 231.

30. Johanson, B. and A. Fox. *The EventHeap: a coordination infrastructure for interactive workspaces*. in *Fourth IEEE Workshop on Mobile Computing Systems and Applications*. 2002. Callicoon, NY, USA: Los Alamitos, CA, USA : IEEE Comput. Soc, 2002: p. 83-93.

31. Johanson, B. and A. Fox, *Extending tuplespaces for coordination in interactive workspaces.* Journal of Systems and Software, 2004. **69**(3): p. 243-266.

32. Johanson, B., *et al. PointRight: Experience with Flexible Input Redirection in Interactive Workspaces*. in *ACM Symposium on User Interface Software and Technology (UIST-2002)*. 2002. Paris, France: p. 227-234.

33. Johanson, B., *et al. Multibrowsing: Moving Web Content across Multiple Displays*. in *Ubicomp 2001*. 2001. Atlanta, GA, USA: p. 256-272.

34. Johanson, B., T. Winograd, and A. Fox, *Interactive Workspaces.* Computer, 2003. **36**(4): p. 99-101.

35. Ju, W., *et al. Where the Wild Things Work: Capturing Physical Design Spaces*. in *Conference on Computer-Supported Cooperative Work*. 2004. Chicago, IL, USA: in press

36. Kindberg, T. and A. Fox, *System Software for Ubiquitous Computing*, in *IEEE Pervasive Computing*. 2002. p. 70-81

37. Liston, K., M. Fischer, and T. Winograd, *Focused Sharing of Information for Multi-disciplinary Decision Making by Project Teams.* ITcon, 2001. **6**: p. 69-81.

38. McKinney, K., *et al. Interactive 4D-CAD*. in *Third Congress on Computing in Civil Engineering*. 1996. Anaheim, CA, USA: p. 383-389.

39. Milne, A. and T. Winograd. *The iLoft Project: A Technologically Advanced Collaborative Design Workspace as Research Instrument*. in *14th Annual International Conference on Engineering Design (ICED'03)*. 2003. Stockholm, Sweden: p.

40. Morris, M.R., D. Morris, and T. Winograd. *Individual Audio Channels with Single Display Groupware: Effects on Communication and Task Strategy*. in *Conference on Computer-Supported Cooperative Work.* 2004. Chicago, IL, USA: p.

41. Myers, B., *Using Handhelds and PCs Together*, in *Communications of the ACM*. 2001. p. 34-41

42. Ponnekanti, S., *et al. Portability, Extensibility and Robustness in iROS*. in *1st IEEE International Conference on Pervasive Computing and Communications (PerCom 2003).* 2003. Dallas-Fort Worth, Texas, USA: IEEE: p. 11-19.

43. Raskin, J., *The humane interface : new directions for designing interactive systems*. 2000, Reading, Mass.: Addison Wesley. xix, 233.

44. Rekimoto, J. *Pick-and-drop: a direct manipulation technique for multiple computer environments*. in *Tenth Annual Symposium on User Interface Software and Technology.* 1997. Banff Alta. Canada: New York, NY, USA : ACM, 1997: p. 31-9.

45. Ringel, M., *et al. Barehands: implement-free interaction with a wall mounted display*. in *CHI '01 extended abstracts on Human factors in computer systems*. 2001. Seattle, Washington: ACM Press: p. 367-368.

46. Russell, D. and R. Gossweiler. *On the Design of Personal & Communal Large Information Scale Appliances*. in *Ubicomp 2001.* 2001. Atlanta, GA, USA: p. 354-361.

47. Shih, C.C., *et al. Teamspace: A Simple, Low-Cost and Self-Sufficient Workspace for Small-Group Collaborative Computing*. in *Submitted to the Conference on Computer-Supported Cooperative Work*. 2004. Chicago, IL, USA: p.

48. Stone, M.C., *Color and brightness appearance issues in tiled displays.* IEEE Computer Graphics and Applications, 2001. **21**(5): p. 58-66.

49. Streitz, N., *et al. i-LAND: An interactive Landscape for Creativity and Innovation*. in *ACM Conference on Human Factors in Computing Systems (CHI'99).* 1999. Pittsburgh, PA, USA: ACM Press, New York, NY, USA: p. 120-127.

50. Tandler, P., *The BEACH Application Model and Software Framework for Synchronous Collaboration in Ubiquitous Computing Environments.* To appear in Journal of Systems and Software, 2003(Special Issue on Application Models and Programming Tools for Ubiquitous Computing).

51. Ullmer, B., H. Ishii, and D. Glas. *mediaBlocks: physical containers, transports, and controls for online media*. in *SIGGRAPH 98: 25th International Conference on Computer Graphics and Interactive Techniques*. 1998. Orlando, FL, USA: New York, NY, USA : ACM, 1998: p. 379-86.

52. Winograd, T., *Architectures for context.* Human-Computer Interaction, 2001. **16**(2/4): p. 401-19.

53. Winograd, T., *Interaction Spaces for 21st Century Computing*, in *HCI in the New Millennium*, J. Carroll, Editor. 2001, Addison Wesley.

54. Winograd, T. and F. Guimbretiére. *Visual Instruments for an Interactive Mural*. in *CHI '99 extended abstracts on Human factors in computer systems*. 1999. Pittsburgh, PA, USA: ACM Press: p. 234-235.

55. Winograd, T. and B. Lee. *Cognitive factors in multi-device interaction.* in *HCI Consortium*. 2004. Winter Park, CO, USA.

56. Wyckoff, P., *et al.*, *T spaces.* IBM Systems Journal, 1998. **37**(3): p. 454-74.

57. Ponnekanti, S., et al. ICrafter: A Service Framework for Ubiquitous Computing Environments. UBICOMP 2001, Atlanta, Georgia, USA. 2001.