Multisurface Interaction in the WILD Room

Michel Beaudouin-Lafon, Stéphane Huot, Mathieu Nancel, *Université Paris-Sud*
Wendy Mackay, Emmanuel Pietriga, Romain Primet, Julie Wagner, *INRIA*
Olivier Chapuis, Clément Pillias, *CNRS*
James R. Eagan, *Télécom ParisTech*
Tony Gjerlufsen, Clemens Klokmose, *Aarhus University*

Abstract

The WILD room (wall-sized interaction with large datasets) serves as a testbed for exploring the next generation of interactive systems by distributing interaction across diverse computing devices, enabling multiple users to easily and seamlessly create, share, and manipulate digital content.

Ubiquitous computing offers a vision in which each person owns multiple computers that work together seamlessly, embedded into the fabric of everyday life [1]. Part of this vision has arrived: interactive surfaces are everywhere, from smartphones, tablets, and laptops to large-screen televisions and smart boards; from car navigation systems to fitness monitoring devices. Their integration, however, is hardly seamless: data is often trapped in individual applications or services, and interaction is usually limited to a single device at a time.

As the "The WILD Platform" sidebar describes, the WILD room (wall-sized interaction with large datasets) is a multisurface environment featuring a wall-sized display, a multitouch table, and various mobile devices that we designed to help scientists collaborate on the analysis of large and complex datasets. We combine empirical studies, participatory design, and fundamental research on basic interaction tasks to explore the design and engineering of the next generation of interactive systems. The key to this approach is to distribute interaction, not just content, across a variety of interactive surfaces.

Designing with extreme users

Our research strategy involves designing an extreme environment that pushes the limits of technology—both hardware and software. To ground the design process, we needed extreme users—people whose daily work both inspires and stress-tests the environment. We chose scientists who use a variety of techniques to understand exceptionally large and complex datasets. We invited researchers from the Paris-Saclay campus in astrophysics, particle physics, chemistry, molecular biology, neuroscience, mechanical engineering, and applied mathematics to an initial "show-and-tell" workshop. Scientists from each lab presented specific examples of the challenges they faced at that time, along with their data analysis processes and tools. We discussed the similarities and differences among their approaches, seeking to identify both universal needs and unique opportunities.

For example, a group of microbiologists might arrive in the WILD room with their laptops and analysis tools to study how one molecule docks with another. One might bring up a large molecular model downloaded from the research lab's server, another might add interactive 3D models of related molecules, and others might access online databases, websites, and research articles. They could shift smoothly among different representations of each molecule and transfer them from one interactive display to another, working together in the same room or collaborating with remote colleagues.

We identified four common strategies for managing complex scientific data where the WILD multisurface environment could significantly improve and even completely change work practices:

1. navigation through a single, very large object, such as a simulation of a molecule with tens or hundreds of thousands of atoms or a gigapixel image of deep space containing thousands of galaxies;
2. comparison of a large number of related images, such as pathological brain scans or observations of regions of the sky at different wavelengths;
3. juxtaposition of a variety of heterogeneous forms of data from different sources, such as a mix of research articles, raw data tables, formulas, graphs, photographs, and video clips;
4. communication with remote colleagues about all of the above to facilitate collaborative exploration.

We then used the WILD room as a working laboratory for exploring advanced multisurface interaction techniques.

Sidebar: The WILD platform

The WILD room (**W**all-size **I**nteraction with **L**arge **D**atasets) features a large wall display (top, left) powered by a 16-computer cluster (top, right) and two front-end computers, a motion tracking system (bottom, left), and an interactive table (bottom, right).

The wall display consists of 32 off-the-shelf 30-inch monitors organized in an 8 x 4 grid, for a total resolution of 131 million pixels (20480 x 6400). The high pixel density (about 100 dpi), a defining characteristic of WILD, is rare on wall displays. The monitors are mounted on four movable carts, letting users test different configurations such as the triptych shown in Figure A. Each computer has two graphics cards driving one screen each. Displaying wall-sized images requires distributed software that runs across the cluster.

The motion tracking system uses 10 infrared cameras to detect the position of passive markers attached to different devices, such as the T-shaped tool shown at the lower left in Figure A. The system has very low latency and a precision of less than one millimeter across the room. We typically use it to precisely track each device's position and to support advanced interaction techniques.

The interactive table uses FTIR (frustrated total internal reflection) technology to track up to 32 simultaneous contact points with a 1920 x 1080 resolution. Because it has only half the pixel density of the wall display, we are adding a second table with higher pixel density and a flat screen. Smartphones, PDAs, tablets, and laptops provide additional, personal interactive surfaces. We also use input devices such as gyroscopic and wireless mice and custom devices.

Figure 1: *Using the Wizard of Oz technique to prototype how a tablet can serve as a mobile, physical filter atop a wall-sized image.*

Exploring multi-surface interaction

We employ two complementary strategies for generating and testing ideas: participatory design, which focuses on qualitative understanding and external validity, and controlled experiments, which focus on quantitative evaluations and internal validity.

Participatory design actively involves users throughout the design process. We visited several labs to observe their current research procedures and conducted participatory design workshops in the WILD room with the astrophysicists and neuroanatomists, who face interestingly different analysis challenges.

One of the most effective techniques was the Wizard of Oz, in which scientists acted out ideas for manipulating their data, using paper images, laptops, and other props. A member of the group, identified as the wizard, would operate the WILD wall so that it reacted to the users' actions, creating a compelling shared experience of a possible future. This often sparked additional ideas and provided insights as to which techniques were most worth pursuing. For example, the scientists spontaneously experimented with midair hand gestures and using external props to manage their data. One neuroscientist brought along a 3D physical model of his own brain from an MRI scan. He had the idea of using it to control the orientation of all 64 normal and pathological brains displayed on the wall. He had dreamed of doing this in his lab, where he was limited to using a mouse to compare at most four brain scans on a single screen.

Scientists also explored relationships among mobile and stationary devices. For example, one astrophysicist was examining a large image of the Milky Way galaxy, accompanied by a series of smaller images at different wavelengths. He suddenly grabbed an iPad tablet, held it up to the primary image, and simulated how he would like to treat it as a physical, interactive filter. Figure 1 shows how he envisioned moving the tablet around, maintainng an overview of the whole image while flipping through different filters to focus on specific wavelengths.

Participatory design helped us to delve deeply into the problem space and generate specific innovative ideas. However, we also needed a more systematic approach for characterizing the design space of interaction techniques and making informed choices. For example, the astrophysicists showed us a 400,000-pixel-wide image of the center of the galaxy. While they could see it on WILD much better than in their lab, the image was still 20 times larger than the display capabilities of our wall.

These and other gigapixel images highlighted the need for powerful panning and zooming techniques that could be operated from any location in front of the wall. This suggested midair interaction, using the hands to point to the locus of the zoom within an image and to control its expansion and contraction from there. Based on the participatory design results and our own explorations, we identified three important dimensions, illustrated in Figure 2, that characterize the design space for pan-and-zoom on a wall display.

We ran a controlled experiment to evaluate our hypotheses about which factors increase performance, accuracy, and comfort [2]. Our goal was not necessarily to determine the single "best" technique, but rather to understand the tradeoffs and help users and designers decide which to use under what circumstances.

We found that, in general, two hands are better than one; linear gestures are faster than circular ones, despite the need to "clutch;" and greater guidance (or fewer degrees of freedom) significantly increases performance. Most midair freehand gestures are tiring and inefficient. The only exception is the two-handed linear gestures in free space shown in Figure 2f—an appealing technique that requires no additional device.

These and other experiments, together with the results of the participatory design sessions, have led to an effective set of techniques that we now use routinely in WILD.

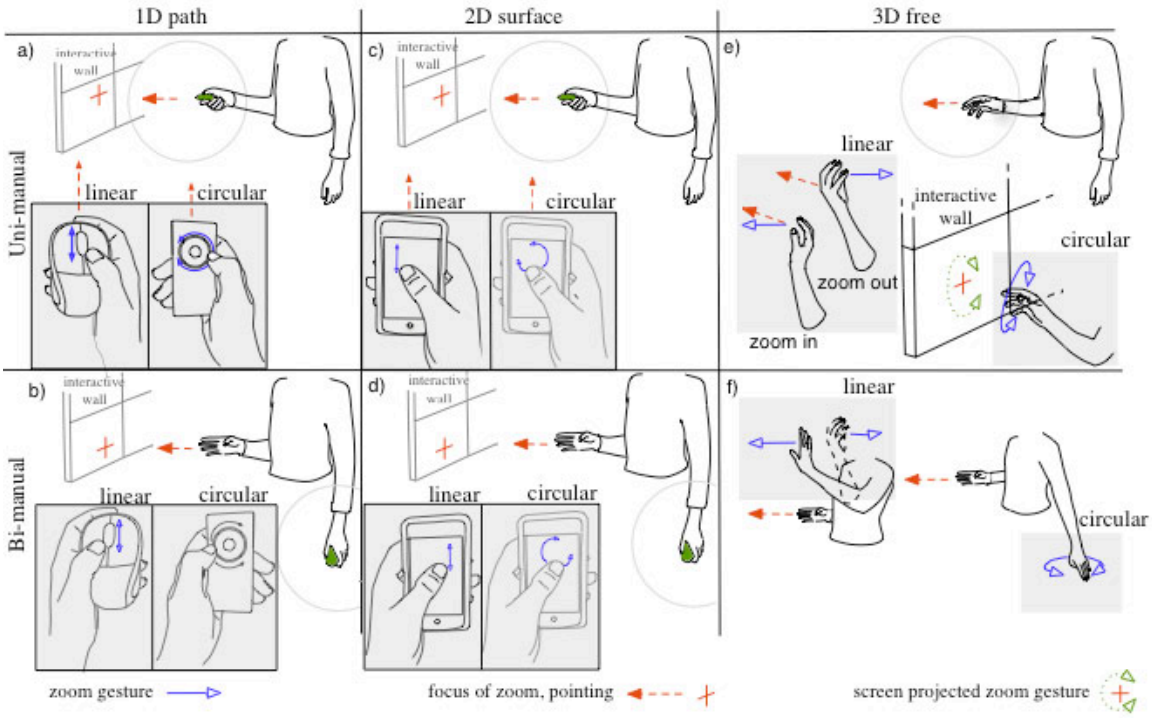

Figure 2: *A design space for midair pan and zoom techniques with three dimensions: interaction with one hand (top row) or both hands (bottom row); gestures that are constrained to one dimension (left column), to a 2D surface (center column), or free in 3D space (right column); linear or circular gestures (insets in each cell). For example, (d) corresponds to using the dominant hand as a laser pointer to indicate the focus point and the nondominant hand to control zooming with linear or circular gestures on a handheld device. In (c), both tasks are carried out with the dominant hand.*

Figure 3: *Interaction instruments. (left) An interaction instrument sorts the 64 displayed brain scans, (center) a brain prop controls the scan orientation, and (right) a digital pen annotates content on the wall. (Source: Photothèque CNRS, Cyril Fresillon.).*

Developing multi-surface applications

Developing software for multisurface environments raises several challenges. First, applications are inherently distributed and the environment is dynamic: 20 to 30 computers are involved in a typical session, including the cluster running the wall, the computers running the table and motion tracking system, the handheld devices, and the users' laptops. Second, input devices can be combined in various ways to interact with the various surfaces, and multiple users must be able to interact in parallel. Finally, content comes from a variety of sources, including static documents brought by users and live windows from legacy applications.

Our goal was to simplify the development of applications in this context without sacrificing the flexibility and openness required by our users. This led to a modular approach that separates user interaction, graphical rendering, and content sources.

Distributed interaction

Our concept of ubiquitous instrumental interaction separates interaction from the rest of the application [3]. An interaction instrument mediates interaction between a user and the objects of interest. For example, users can designate objects with a pointing instrument, move them with a drag-and-drop instrument, and change their color with a color selection instrument. Instruments are independent of the objects they operate on: they need only know that the object implements a given protocol, such as selecting, changing position, or setting a color. Multiple instruments can be used in parallel. Instruments can also be embodied in portable devices—for example, a smartphone used as a laser pointer. In this case, the instrument runs on the device and interacts with objects located on other surfaces.

We have created generic instruments for selecting, moving, organizing, and annotating objects, as well as more specific ones, such as the brain prop shown in Figure 3, which is used to control the orientation of brain scans on the wall. These interaction instruments have proven very flexible since they can be customized to the users' needs without modifying the application: instruments discover which objects they can interact with based on the protocols that the objects implement.

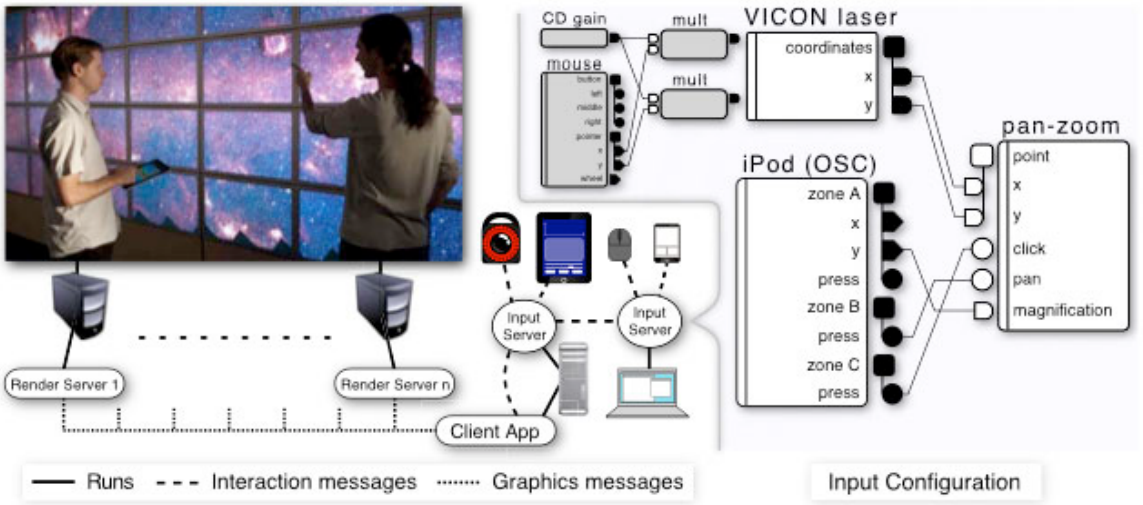

Figure 4: *jBricks and the WILD Input Server. (left) A jBricks application manages a scene of 2D objects laid out on an infinite canvas. On the cluster, render servers replicate the scene and display only the objects that lie in their viewing frustum. (right) A configuration of the WILD Input Server for a virtual device combining a VICON position-tracking component and an iPod handheld device. The configuration can be tested outside the WILD room by replacing the VICON component with those in gray and using a mouse for position input. The pan-zoom component on the right sends high-level events to the application.*

At a lower level, input in a multisurface environment can come from a variety of sources, including standard devices such as mice and keyboards, multitouch devices such as interactive tables and tablets, and systems such as motion trackers. Rather than sending this raw input directly to applications or instruments, we have created an intermediate layer called the WILD Input Server [4].

The WILD Input Server uses the ICon visual editor [5] to create and edit input configurations. Figure 4 shows how a configuration transforms low-level input from physical devices into higher-level events sent to client applications. The WILD Input Server supports standard protocols such as USB-HID, OSC, TUIO, and VRPN as well as devices such as LiveScribe interactive pens or the VICON motion tracker. The server sends events to applications through various protocols (primarily OSC; http://opensoundcontrol.org) or plug-ins. Applications can also remotely control the server to start, stop, or change a configuration or to load a plug-in.

Developers can easily create and modify configurations by assembling components such as filters, adapters, and flow controllers, even during a prototyping session. Configurations typically define virtual devices that aggregate input from multiple sources. For example, the application sees a multitouch handheld device whose 3D position is provided by the motion tracking system as a single device.

Our implementation of the pan-and-zoom techniques from Figure 2 illustrate the flexibility of this approach. We developed the techniques outside the WILD room, substituting a mouse or a Wiimote for the motion tracking system, and created a set of virtual devices that we could modify and fine-tune in the WILD room, without relaunching the application.

Distributed rendering

Displaying graphics in a multisurface environment is challenging because users want to organize their data onto a virtual canvas that spans multiple surfaces. Depending on the configuration and the task at hand, different surfaces display either the same part or different parts of the canvas. Tiled displays require particularly high performance to create the illusion of a single, continuous surface with no tearing.

Existing cluster-based systems for distributed rendering do not fit our requirements. For example, Equalizer and CGLX require adapting or rewriting applications using OpenGL, while SAGE [6] uses pixel streaming and therefore cannot take full advantage of ultra-high resolution wall displays. Our approach uses replication: each machine driving a display runs a replica of the complete application or a rendering client that holds a copy of the scene. Each replica knows which part of the scene to display; a master application synchronizes changes to the scene and the viewing camera.

We created two frameworks to develop multisurface applications based on this model. The first, jBricks [4], is based on a 2D scene graph that describes the canvas's content and a set of reactions that describe how to respond to user actions, similar to traditional user-interface toolkits. Scene graph objects include geometric shapes, text, images, and Java Swing widgets laid out on an infinite canvas and observed through one or more cameras.

jBricks uses a replicated approach to render the scene graph on a cluster-driven tiled display. The toolkit supports smooth real-time panning and zooming of very large information spaces, including gigapixel images, as well as interactive visual effects such as magnifying lenses. By making distribution transparent to the application, jBricks greatly lowers the barrier to developing multisurface applications.

Our second framework, Shared Substance, takes a different approach by making distribution explicit [7]. A Shared Substance application is a collection of processes called environments that run on different machines. The application discovers environments dynamically, and they can appear and disappear at any time. Each environment contains a hierarchical data structure that it can share, in whole or in part, with other environments.

An environment accesses a shared subtree either by replicating it and accessing the local copy or by mounting it and accessing the original through remote procedure calls. Environments can use facets to dynamically add functionality to a shared subtree. For example, Figure 5 shows how our Substance Canvas application uses facets to display the canvas, modify its content, and support interaction. Shared Substance provides great flexibility and makes it possible to create applications that dynamically adapt to their use context and are reconfigurable at runtime.

Distributed content sources

In a multisurface environment, users need to juxtapose content from multiple sources, as if the various surfaces were extensions of their laptops. Sources include passive documents such as PDF files and images, active documents such as webpages, and live applications such as data analysis and visualization programs. The challenge lies in integrating such heterogeneous sources into a unified environment.

We began with simple but effective solutions based on conventional tools: a user can e-mail a document to WILD to display it on the wall or "print to the wall" by sending a document to a printer queue that WILD monitors. Users can also fill out a simple Web form or use a bookmarklet to display webpages on the wall.

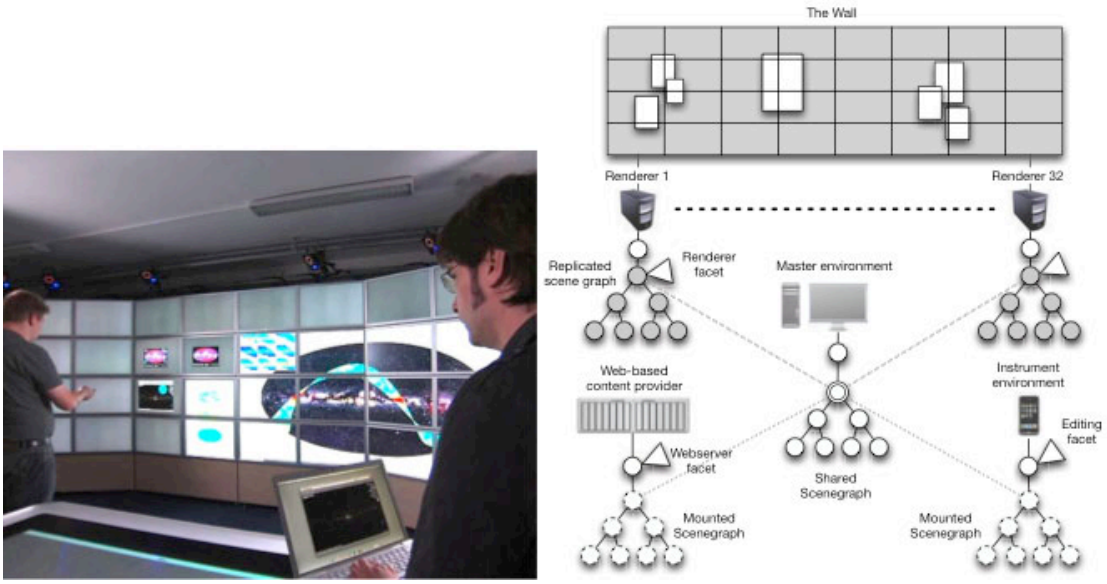

Figure 5: *Substance Canvas application. (left) Two users share content between the wall, the table and a laptop. (right) A master environment shares a scene graph representing a canvas. Rendering environments replicate the scene graph to add local rendering capabilities, while interaction instruments mount the scene graph to add editing functions. Content providers then mount the scene graph to modify its content, for example, through a webservice. (Source: INRIA.)*

Even so, scientists must be able to use existing applications. Since porting them to our frameworks is not practical, at least in the short term, both jBricks and Shared Substance support the display of live applications running on a different computer, typically a user's laptop. For Linux, we use Metisse [8] to send pixel-based representations of the windows. For Mac OS, we use Scotty [9] to send vector-based representations of the windows, resulting in smooth scaling when displayed on the wall. In both cases, the scientists can use an instrument that simulates a mouse to interact with the teleported applications.

An alternative with better performance is to run the legacy application on the WILD cluster itself. Using Shared Substance, we wrapped the BrainVISA 3D visualization application (http://brainvisa.info) into an environment that shares the address of the scan being displayed and the position of the virtual camera controlling its orientation. Figure 3 shows the cluster running 64 such environments, each displaying a different brain scan. The table runs an instrument for organizing the brain scans, while the brain prop controls the orientation of a master camera, which is shared by the 64 environments that display the individual brain scans.

The resulting application was created in a few days, providing neuroanatomists with a unique tool to study the brain. We used a similar approach with the PyMol molecule viewer. We can display a single molecule on the full wall by having each replica display its part. Rotating it in real time shows no visible tearing.

By distributed content, rendering, and interaction we have created a modular architecture that simplifies the development of multisurface applications while supporting flexible interaction as well as legacy content and applications. Even without optimization, performance is good: users can interact with full-wall images in real-time with little perceivable lag. The ability to change configurations and components on the fly during a design session makes these tools an excellent platform for rapid prototyping.

Sidebar: Recommended Reading

Researchers have long been interested in room-scale interaction. An early project was the Stanford iRoom [10], an infrastructure that enabled the devices in a room to communicate with each other. Lucia Terrenghi and colleagues provided a comprehensive taxonomy of different scales of multisurface environments [11], from wristwatches and phones to the side of a building. These environments support users interacting in isolation or simultaneously, in parallel or collaboratively.

At the room-sized scale of this spectrum, much work has focused on creating large high-resolution displays such as wall-sized tiled displays and CAVEs. These projects often focus on high-performance distributed rendering and data-sharing rather than on interaction. Tao Ni and colleagues surveyed the technologies and application for such environments and emphasized the need for better interaction techniques [12]. Our work addresses these issues by introducing concepts and techniques for distributed, multisurface interaction [3, 4, 7].

Conclusion

Realizing the vision of ubiquitous computing requires creating interaction architectures and paradigms that harness the power of combining devices and services into integrated environments. Today's smartphones, tablets, multitouch tables, and wall displays bring little more than the sum of their parts. In contrast, the WILD room's multisurface interaction paradigm illustrates how interaction, not just content, can be distributed across multiple devices.

The scientists we have worked with are eager to use WILD for their daily work. By involving them in the design process, we have been able to focus on their real needs and identify the real technological challenges. We have learned the following lessons in the process:

- decouple tools from one another and use simple protocols to facilitate their integration;
- focus on interaction rather than rendering, and assume that hardware will provide sufficient performance;
- leverage existing tools when possible, but also develop from scratch when needed; and
- explore alternative designs to gain deeper understanding of their respective advantages and disadvantages.

However, this is just the beginning. We must work with additional user groups to gain new insights and expand the scope of multisurface interaction, extend our interaction vocabulary to match the richness of desktop interfaces, and scale our software architectures to test them with other applications.

One important requirement not currently addressed by WILD and unanimously requested by our users is support for collaboration among remote colleagues. While the multisurface interaction paradigm naturally scales to remote groups, additional technology is needed to support face-to-face communication. The WILD room is now part of Digiscope (http://digiscope.fr), a larger project that will create a network of interactive visualization rooms specifically designed to address these issues.

In the long run, platforms such as WILD will become increasingly affordable. Wall-sized displays will combine high-definition and multitouch surfaces without borders, and motion tracking will become more reliable, without the need for markers. These advances will reduce the constraints on users and support a wider range of multisurface interactions.

We anticipate that this technology will become prevalent in the workplace, first in meeting rooms and design studios, then in offices, and later in the home, offering families new ways to play, study, communicate, and enjoy entertainment. Only then will multisurface interaction become truly integrated into the fabric of our everyday lives.

Acknowledgments

We thank our partner laboratories, in particular IAS (astrophysics), LAL (particle physics), IGM (biology) and Neurospin (neuroscience) for their participation. WILD is supported by a Région Île-de-France/Digiteo grant and by Université Paris-Sud, INRIA, CNRS, ANR and the INRIA-Microsoft joint laboratory.

References

1. Mark Weiser. The computer for the 21st century. *Scientific American,* 265(3):94–104, 1991.
2. Mathieu Nancel, Julie Wagner, Emmanuel Pietriga, Olivier Chapuis , and Wendy Mackay. Mid-air pan-and-zoom on wall-sized displays. In *Proc. Human Factors in Computing Systems,* CHI '11, 177–186. ACM, 2011.
3. Clemens Klokmose and Michel Beaudouin-Lafon. VIGO: Instrumental interaction in multi-surface environments. In *Proc. Human Factors in Computing Systems,* CHI '09, 869–878. ACM, 2009.
4. Emmanuel Pietriga, Stéphane Huot, Mathieu Nancel, and Romain Primet. Rapid development of user interfaces on cluster-driven wall displays with jBricks. In *Proc. Engineering Interactive Computing Systems,* EICS '11, 185–190. ACM, 2011.
5. Pierre Dragicevic and Jean-Daniel Fekete. Support for input adaptability in the ICon toolkit. In *Proc. Multimodal Interfaces,* ICMI '04, 212–219. ACM, 2004.
6. Byungil Jeong, Jason Leigh, Andrew Johnson, Luc Renambot, Maxine Brown, Ratko Jagodic, Sungwon Nam, and Hyejung Hur. Ultrascale collaborative visualization using a display-rich global cyberinfrastructure. *IEEE Computer Graphics and Applications,* 30(3):71–83, 2010.
7. Tony Gjerlufsen, Clemens Nylandsted Klokmose, James Eagan, Clément Pillias, and Michel Beaudouin-Lafon. Shared Substance: developing flexible multi-surface applications. In *Proc. Human Factors in Computing Systems,* CHI '11, 3383–3392. ACM, 2011.
8. Olivier Chapuis and Nicolas Roussel. Metisse is not a 3D desktop! In *Proc. User Interface Software and Technology,* UIST '05, 13–22. ACM, 2005.
9. James R. Eagan, Michel Beaudouin-Lafon and Wendy E. Mackay. Cracking the cocoa nut: user interface programming at runtime. In *Proc. User Interface Software and Technology,* UIST '11, 225–234. ACM, 2011.
10. Jan Borchers, Meredith Ringel, Joshua Tyler, and Armando Fox. Stanford interactive workspaces: a framework for physical and graphical user interface prototyping. *IEEE Wireless Communications,* 9(6):64–69, December 2002.
11. Lucia Terrenghi, Aaron Quigley, and Alan Dix. A taxonomy for and analysis of multi-person-display ecosystems. *Personal and Ubiquitous Computing,* 13:583–598, November 2009.
12. Tao Ni, Greg S. Schmidt, Oliver G. Staadt, Mark A. Livingston, Robert Ball, and Richard May . A survey of large high-resolution display technologies, techniques, and applications. In *Proc. Virtual Reality Conference,* VR '06, 223–236. IEEEMarch 2006.

About the authors

Michel Beaudouin-Lafon *is a professor of computer science at Université Paris-Sud and a senior member of Institut Universitaire de France. His research interests include interaction techniques and paradigms, collaborative systems, and engineering of interactive systems. He received a PhD in computer science from Université Paris-Sud. Contact him at mbl@lri.fr.*

Olivier Chapuis *is a research scientist at CNRS. His research interests include windowing systems, pointing, multiscale interfaces, and interaction techniques. He received a PhD in mathematics from Université Paris VII Diderot. Contact him at olivier.chapuis@lri.fr.*

James R. Eagan *is an assistant professor at Télécom ParisTech. His research interests include information visualization and making software more malleable for end-users and programmers. He received a PhD in computer science from the Georgia Institute of Technology. Contact him at james.eagan@telecom-paristech.fr.*

Tony Gjerlufsen *received a PhD in computer science from Aarhus University. His research interests include software architecture, human-computer interaction, philosophy of computer science, and ubiquitous computing. Contact him at tony@cs.au.dk.*

Stéphane Huot *is an associate professor at Université Paris-Sud, on leave at INRIA. His research interests include interaction techniques, input devices and methods, and engineering of interactive systems. He received a PhD in computer science from Université de Nantes. Contact him at stephane.huot@lri.fr.*

Clemens Klokmose *is a postdoctoral fellow at Aarhus University. His research interests include human-computer interaction and multisurface environments. He received a PhD in computer science from Aarhus University. Contact him at clemens@cs.au.dk.*

Wendy Mackay *is a principle research scientist at INRIA and heads the INSITU lab. Her research interests include coadaptive systems, interactive paper, mediated communication, and participatory design. She received a PhD from the Massachusetts Institute of Technology. Contact her at wendy.mackay@lri.fr.*

Mathieu Nancel *is pursuing a PhD at Université Paris-Sud. His research interests include interaction techniques, visualization platforms, and user performance modeling. He received an MSc and an engineering degree in computer science from Université Paris-Sud. Contact him at mathieu.nancel@lri.fr.*

Emmanuel Pietriga *is a research scientist at INRIA. His research interests include interaction techniques, information visualization, the Semantic Web, and the engineering of interactive systems. He received a PhD in computer science from Institut National Polytechnique de Grenoble. Contact him at emmanuel.pietriga@inria.fr.*

Clément Pillias *is an engineer at CNRS. He received an MSc in computer science from Université Paris 6. His research interests include interaction techniques, gestural interfaces, collaborative interaction, and engineering of interactive systems. Contact him at clement.pillias@lri.fr.*

Romain Primet *is a research engineer at INRIA. He received an MSc in computer science from Université de Nice. Contact him at romain.primet@inria.fr.*

Julie Wagner *is pursuing a PhD at INRIA. Her research interests include embodied and tangible interaction with large surfaces. She received an MSc in computer science from RWTH Aachen University. Contact her at julie.wagner@lri.fr.*