UBIWISE, A Ubiquitous Wireless Infrastructure Simulation Environment

John J. Barton, Vikram Vijayaraghavan
Mobile and Media Systems Laboratory
HP Laboratories Palo Alto

We describe UbiWise, a simulator for ubiquitous computing. The simulator concentrates on computation and communications devices, either integrated with physical environments or carried by people. It maintains a three dimensional model of a physical environment viewed by users on a desktop computer through two windows. One window uses the Quake III Arena graphics engine to show a first-person view of the physical environment. The other window from a Java program shows a close-up view of devices a user may manipulate using the desktop keyboard and mouse in place of physical controls. These windows attach as clients to a server maintaining the model of the environment. Multiple users can attach to the same server to create interactive ubiquitous computing scenarios. We describe how UbiWise looks to users and how it can be used as a tool for ubiquitous computing.

UBIWISE, A Ubiquitous Wireless Infrastructure Simulation Environment.

John J.Barton, HP Labs,
Vikram Vijayaraghavan, Stanford University.
Palo Alto, CA 94304
Copyright 2002, HP

Abstract. We describe UbiWise, a simulator for ubiquitous computing. The simulator concentrates on computation and communications devices, either integrated with physical environments or carried by people. It maintains a three dimensional model of a physical environment viewed by users on a desktop computer through two windows. One window uses the Quake III Arena graphics engine to show a first-person view of the physical environment. The other window from a Java program shows a close-up view of devices a user may manipulate using the desktop keyboard and mouse in place of physical controls. These windows attach as clients to a server maintaining the model of the environment. Multiple users can attach to the same server to create interactive ubiquitous computing scenarios. We describe how UbiWise looks to users and how it can be used as a tool for ubiquitous computing.

Introduction

This paper describes UbiWise, a ubiquitous wireless infrastructure simulation environment. UbiWise simulates aspects of ubiquitous computing. The simulator concentrates on computation and communications devices, either integrated with physical environments or carried by people. The simulator maintains a three dimensional model of a physical environment viewed by users on a desktop computer through two windows. One window uses the Quake III Arena graphics engine to show a first-person view of the physical environment, including devices embedded in the environment and other users. The other window shows a close-up view of devices a user may manipulate using the desktop keyboard and mouse in place of physical controls. These windows attach as clients to a server maintaining the model of the environment. Multiple users can attach to the same server to create interactive ubiquitous computing scenarios.

Motivation.

UbiWise was born out of a desire to make progress on systems for handheld devices[1] without waiting for the devices themselves. It grew with the realization that this frustrated desire is common.

New hardware drives ubiquitous computing. The concept of ubiquity for computing arose from the potential for inexpensive and tiny computers embedded in or connected to our environment. However, the goals for research in ubiquitous computing lie beyond the hardware itself [2]. The unique aspects of ubiquitous computing arise through combinations of technology available to us--or forced upon us--by ubiquity [3]. From our experience [4] and from observing others, we see a dependency cycle that limits progress in understanding these unique aspects:

- the design of suitable hardware depends upon understanding how the hardware fits in to ubiquitous computing applications,
- ubiquitous computing applications cannot be developed without suitable hardware.

UbiWise provides a framework for researchers to simulate hardware and low-level software so that more work can be done on the other dimensions of ubiquitous computing, leading to more ideas for hardware and more refined requirements for it.

To be a tool for ubiquitous computing the simulator must to tackle a broad spectrum of opportunities and challenges, including the ability to:

- experiment with new sensors, whether handheld or environmental, without building and deploying them;

- aggregate device functions without connecting real devices;
- develop new service and device discovery protocols without implementing these protocols in multiple mobile devices;
- observe how users might react to new devices and services before we can fully realize them;
- explore the integration of handheld devices and Internet services without the huge cost and large teams need to realize this integration.

That is we want a general-purpose simulator, not one limited to a specific device or application. Furthermore we must simulate not just devices but device response to context or proximity to other users or devices.

Obviously we cannot personally implement all of the simulator and these experiments within it. We want other researchers to look at our simulator and decide that they could use it to make progress in their work. Therefore we have tried to build on existing experience [5] and to design the simulator to be useful and interesting to others.

Contribution

Our preliminary work on a multi-device simulator (called WISE) was presented at the UbiTools Workshop '02 in Atlanta, Georgia[6]. This tool had 2D views of several devices. At UbiTools we learned of the QuakeSim[5] project of Bylund and Espinoza. They modified the Quake III Arena (Q3A) [7] code to output the position co-ordinates of a player into the Context ToolKit [8]. They demonstrated the potential of this tool in the development of a location-based "virtual note" project called GeoNotes[9]. GeoNotes had a Java program reading the position co-ordinates and providing a user-interface for placing virtual notes.

These successes convinced us to pursue a more complete simulator for ubiquitous computing. We extended the code from Espinoza and Bylund, creating virtual handheld devices that come up like Q3A weapons; this deepened the modification of the Q3A engine to tailor it to ubiquitous computing. Then we coupled the new program into our 2D device-interaction simulator for detailed manipulations. This generalizes the GeoNotes design to create a Java framework for device user interfaces. In addition we have worked on development tools for the simulator.

This paper reports both our progress in implementing the simulator and an analysis for the broad potential for simulation in the complex and emerging field of ubiquitous computing. The UbiWise simulator is intended to be a continuing work in progress and all of the new code is released[10] under the LGPL[11]. Open source collaboration is hoped for in developing this further and especially using it in a variety of ubiquitous computing scenarios.

Describing the Simulator

We describe our simulator from three points of view:
- a user – a person running the simulation;
- a researcher – a person configuring the simulation;
- a developer – a person extending the simulator itself.

Of course most `users` will be `researchers` and many `researchers` will also be `developers`. In addition, "configuring the simulation" can run all the way from minor modifications of some parameters before use to a complex implementation of a new protocol to be tried in the simulator.

To help explain the UbiWise system we will use a simple example scenario. The scenario involves two people, one with a wireless digital camera and one with a network connected PDA. They enter the same room and see three digital picture frames on the wall. The person with the digital camera transfers an image up to one of the frames; the other person copies that image down to their PDA.

The scenario includes discovery, both physical and electronic, device selection, and data transfer between two pairs of devices, two portable and one fixed. There are a large number of ways that this task can be accomplished; we want UbiWise to be able to simulate all of them. This does not mean that UbiWise contains all the devices and protocols, but that we aim to provide a framework sufficient for the addition of the individual elements so that their combination can be explored. We hope that sharing these elements becomes a part of research in ubiquitous computing.

As we describe the simulator from each of the three points of view—user, researcher, and developer---we will relate back to this scenario.

Users view of UbiWise

We start our description with the human participating in a simulated scenario, the *user*. UbiWise is a multi-user client-server simulator. One client represents each user. Each client has two views, a **device-interaction** view mimicking the view close at hand and a **physical-environment** view providing realistic time and space simulation. We start by describing the character of these views.

The **device-interaction view** (see Fig. 1) consists of one window per device that the user can manipulate, all enclosed in a larger window. For our example scenario, the user holding the camera would see the back of the camera and the three picture frames on the wall of the room. This user would not see the PDA of the other user in the device-interaction view. The mouse can be used to push buttons on the camera and buttons (if any) on the picture frames. The enclosing window has menus for adding devices to the scenario and for adding windows that provide information about the devices and their connections.

Figure 1 Device-interaction View for the camera user in our example scenario.

The **physical-environment view** (see Fig. 2) consists of a projection of a three-dimensional model of a room or other environment on to a 2D window. This roughly approximates the viewpoint a person would have in the room. Superimposed on the bottom of the screen is an image of the device in the hands of the

user. This device image helps users connect the physical-environment view to the device-interaction view. In the case of our example, the user would see a room with three digital picture frames. At the bottom of the screen would be a portion of the top of the camera, the same camera they can control in the device-interaction view. If the camera user pivots their viewpoint in the room (using, for example, the mouse) they could see a representation of the other user holding the PDA.

Figure 2 Physical-environment view for the camera user in our example scenario.

Absent other constraints, we might have preferred a single view for users of the simulator. However, this would make extending the simulator much more difficult. As we discuss in the next section, the device-interaction view uses Java's Swing toolkit to simplify graphical programming. Implementing device-interaction using the graphical toolkit available in the physical environment simulator would be quite complex. Conversely implementing a full 3D physical model in Java would be a complete project in itself. Since our aim is to provide a tool for research, ease of extension weighs equally with convenience of use.

Researcher's View of UbiWise

In this section we look at some aspects of the simulator behind two windows seen by users. Then we explore some of the avenues open to researchers using UbiWise. Again, we refer to "researchers" here to avoid confusing designers of a scenario with "users" operating the controls of the program, even when these may be the same person.

Inside the Device-Interaction View.

The device-interaction view handles close-up views of devices and simulation of their `virtual' properties like network connections and digital data. This view is a Java program we call **"WISE"**, where

WISE stands for Wireless Infrastructure Simulation Environment. (Our original aim was to concentrate on issue of users interacting with wireless devices).

In WISE, a device is simulated by an image representing a control surface of a physical device and some Java code to handle mouse-button clicks representing user manipulation of the device. The programming model is intentionally conventional and straightforward. As shown in figure 3,
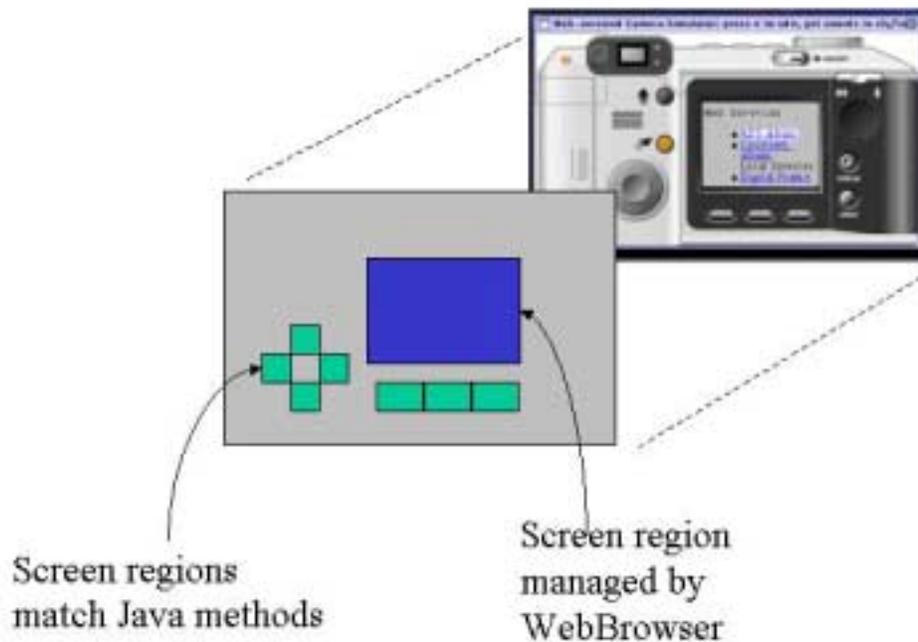
Figure 3 Device- Interaction programming model. The rectangle in front of the camera image indicates regions on the image that are connected to actions in software.

regions on the device image map into methods using Java's Swing Toolkit[12]. A simple screen geometry measurement tool to aids region selection.

Screen regions can also be handled by more sophisticated user-interface code imported from outside the WISE toolkit. In our own work in ubiquitous computing we are extending web-clients for non-desktop applications. Therefore we adapted Java's HTML classes to create a `WebBrowser` class and we used Java's JPEG facilities to create an `ImageBrowser`. These classes offer methods to change the view within their region; these methods can be called when users click on regions of the device image that simulate buttons described above. We have also experimented with the integration of the X-smiles[13] XML browser in a screen region.

The overall effect of WISE is to use a personal computer to simulate interaction with another computer—the digital camera in our example—with different controls and outputs. Thus far we have used only mouse clicks to simulate user fingers and PC screens to simulate device screens. Other input and output modalities could be added, limited only by the capability of the personal computer hosting the simulator.

Using the WISE WebBrowser class makes our simulated device in to a web client using the host machines TCP/IP stack as a crude simulator for a wireless connection. Simulated devices can also act as servers. Again reflecting our own approach to ubiquitous computing, we provide simple adapters to create simulated devices containing web servers. The Web server we use is a re-release of the Nexus HTTP Server[10] written in Java. However this choice is not fundamental to the simulator and other protocols can

be substituted. Over time we hope that researchers with expertise in wireless protocols will help us improve the simulation of the wireless connection.

WISE also provides for debugging, tracing the devices, controls, and classes attached to them. Moreover features like record/playback and an internal-data browser make it easy to test the virtual device in the simulator.

Referring back to our example, the WISE implementation for the image-transfer scenario has three device images: 1)a camera image from the development team behind HP's C618 digital camera, 2)a PDA image downloaded from an HP product description page on the web, and 3)a photograph of a digital picture frame from the Cooltown project[4] demo room. For each image we created a Java class derived from 'WiseDevice'; each button on the camera and PDA has a Java method in the corresponding class. Fig. 1 has a screen shot of WISE for the camera user showing one instance of the camera and three instances of the digital picture frame. (The screen shot that shows the PDA users' view appears as part of Fig. 4).

Inside the Physical-Environment View.

While the device-interaction view represents devices much like they would appear in users hands, the physical-environment view displays a view of the users' environment. This view is a C program we call *UbiSim*, a modification of the Quake III Arena (Q3A) 1.27g source code [7] that includes the location-reporting code from the QuakeSim project[5]. The UbiSim view of the client shows a "first-person" view familiar from combat action computer games (see Fig. 2). On a large screen and with appropriate graphical and sound effects the view is adequate to give a sense of reality; we expect that the client will be more commonly used within a window on a desktop when contemplation is more important than immersion.

The Q3A system was designed to support adaptation and has been use to create other games. The source available for modification roughly approximates the "controller" part of a model-view-controller architecture [14]. Output from graphical design tools like GtkRadiant[15] initialize the 'model' which is then maintained by the Q3A engine along with the `view'. The `controller' code concerns the interaction of entities in the model. This code is built into a dynamic linked library and loaded by the game engine. This is the part that researchers need to change to simulate new wireless media effects. The game engine is currently not available in source, but copies of the binary on CDROM can be purchased and are inexpensive. Versions are available for Windows, Macintosh, and Linux. The engine runs on 3D graphics cards with OpenGL drivers.

For our example scenario, the MilkShape[16] editor was used to create models for the camera, PDA, and the digital picture frame. Using the GtkRadiant editor we create a simple room with three digital picture frames on one wall. Each picture frame could be given a network connectivity option. For example, the frames could have wireless LAN connections. The initial position for the user and the devices available to the user can also be specified in the editor. Once the simulator is running, the camera user would see a low resolution image of the camera in the lower part of the UbiSim window (see Fig. 2); the PDA user would see similar image of a PDA. Both users could move in the shared place, see each other and see the three picture frames on the wall.

How Researchers Might Use UbiWise.

With this rough sketch of the two programs we now describe some of the potential of the simulator as a tool for research. We have tried some preliminary examples along these lines, but for the most part we are speculating rather than report work completed. The descriptions are intended to convey the scope of the simulator and the kinds of work needed use it.

Devices. At the user-interface level, researchers change device decoration, button placement, or semantics to alter the affordance of a device as a component of ubiquitous computing. The simulated devices can mimic existing devices of course, but they can also be devices first invented in simulation. For example we have experimented with a tiny web browser on the back of a digital camera as a means for users to select destinations for images.

For such device experiments, researchers need 2D images of devices and simple Java code for the user interface. Images may be drawn on paper and scanned in, modified from the image of an existing device, or created digitally, for examples. The same images can be used in "model editors [16]" to create 3D visual representations of devices for the physical-environment view. This representations or 'models' correspond

to the weapons or items in the Q3A game. The graphical 3D editors appear to create virtual devices by, for example, tracing the outline of objects from images of them.

Connectivity. At the media layer we can study how the transmission characteristics of new connectivity solutions interact with our use of devices in a physical world. For most wireless technologies, like Bluetooth [17] or IRDA[18], the proximity of devices to each other or to access points plays a critical role in the success or failure of communications. The physical characteristics of the media and the arrangement of devices then becomes part of the user interface for ubiquitous computing.

For media layer experiments, researchers work in the physical-environment simulator, a C-language 3D graphics program designed to be responsive to users. Fortunately, this coding requires little expertise in 3D graphics or multiplayer game dynamics; it is almost entirely code that manipulates ubiquitous computing entities like "users", the devices they carry or encounter, and a space of interaction. In this code we can get notified of many spatial events, such as when a user comes within radio range of a fixed device, when a user picks or sets down up a handheld device, and so on. Sensors, networks of sensors, and location sensing technologies can all be simulated and various aspects of the technologies can be studied one at a time.

Protocols. At the protocol layer we can simulate variable latency, intermittent connection, and limited bandwidth to help engineers and designers learn how to help users with these challenges of real systems. On top of basic connectivity or networks, we can simulate or develop device discovery and registration protocols for ubiquitous computing.

Researchers working on protocols can work as they do now, except that they may choose to relax design constraints like memory footprint, execution speed, and impact on battery life. Conversely, certain design constraints can be increased in the simulator. They may also simulate parts of overall protocols or replace them with existing solutions available on desktop machines. UbiWise acts as an environment for experimenting with the role of protocols in ubiquitous computing; it doesn't support protocols design in a specific way.

Scenarios. By scenario we mean a simulated place with certain devices embedded and a set of users interacting. In terms of the Q3A game, a scenario corresponds to a "level" or a "map". At the scenario level, we consider how devices can be used together or how a device can interact with embedded systems. Or we could explore the role of sensors or networks of sensors interacting with users.

As one kind of example, we should be able to reproduce both the original GeoNotes[9] work and the Cooltown [4] scenarios. That would allow us to begin seeing how set of assumptions by different groups might lead to serendipitous advantages or simply confuse users.

To create scenarios, researchers use a "level editor" like GtkRadiant [15]. This is 3D graphical editor focused on maps of rooms, buildings, or campuses of buildings. By replacing a configuration file used for combat level design with one for UbiWise scenario design, researchers can position devices like digital picture frames in a simulated environment with a graphical tool.

In addition to positioning entities in space, the level editor can be used to attach connectivity options to devices. We use this to, for example, cause a digital picture frame entity (a 3D model) to have the property of a display device (an entity the refreshes one rectangular region with a new JPEG image) using HTTP (has a WebServer). Because this attachment happens in the level editor rather than by programming, new combinations can be tried quickly and easily. This allows researchers to build up a toolkit of simulated devices, media simulations, and plugable protocols.

User Studies. In the similar way that storyboards and mockups help us learn how users interact with technology, the user experience in ubiquitous computing can be studied with the simulator. For example, we can study scalability within the simulator by having, say, 20 users enter a room and access its ubiquitous computing facilities simultaneously. While we cannot say how successful this approach might be for users outside of the research community, much can be learned from experiments on each other.

Mixing in reality. One of our first applications of UbiWise was in a mixed simulator/real mode. By using a physical digital camera posting to a simulated digital picture frame or by using a simulated camera to post to a physical picture frame, we have a portable tool for interoperations testing in ubiquitous computing development [5]. UbiWise has debugging and trace classes that are difficult and costly to create for physical devices.

For the most part these categories are independent: we can simulate, say, device discovery when we are trying to develop data-transfer protocols for a wireless camera or we can simulate data-transfer while we work on discovery. That is, of course, exactly the value of a simulator.

Developers View of UbiWise

We have described the simulator from the user and researcher perspectives. A third perspective would be that of a person who wanted to extend or enhance the quality of the simulation itself. Some activities we discussed in the preceding section already have this quality. For example, the addition of new sensors or wireless media materially improves the simulator for all other researchers. Therefore we focus here on an overview of the client/server design.

UbiWise is a multi-client/multi-server system. As we have described above, each user interacts with two programs. The server has two corresponding parts, a WISE server for the device-interaction clients and a UbiSim server for the 3D physical-interaction clients.

The WISE server talks with the WISE clients; they form one client/server subsystem. The WISE client/server path is used for data transfer between simulated devices. The UbiSim server talks with the UbiSim clients; they form a second client/server subsystem. The UbiSim client server/path synchronizes the physical models. Separate paths help to insure that data transfer does slow or block physical synchrony.

As illustrated in Fig. 4, the two subsystems connect at the server side. The WISE server sends data-update events to the UbiSim server and the UbiSim server sends device-update events back.
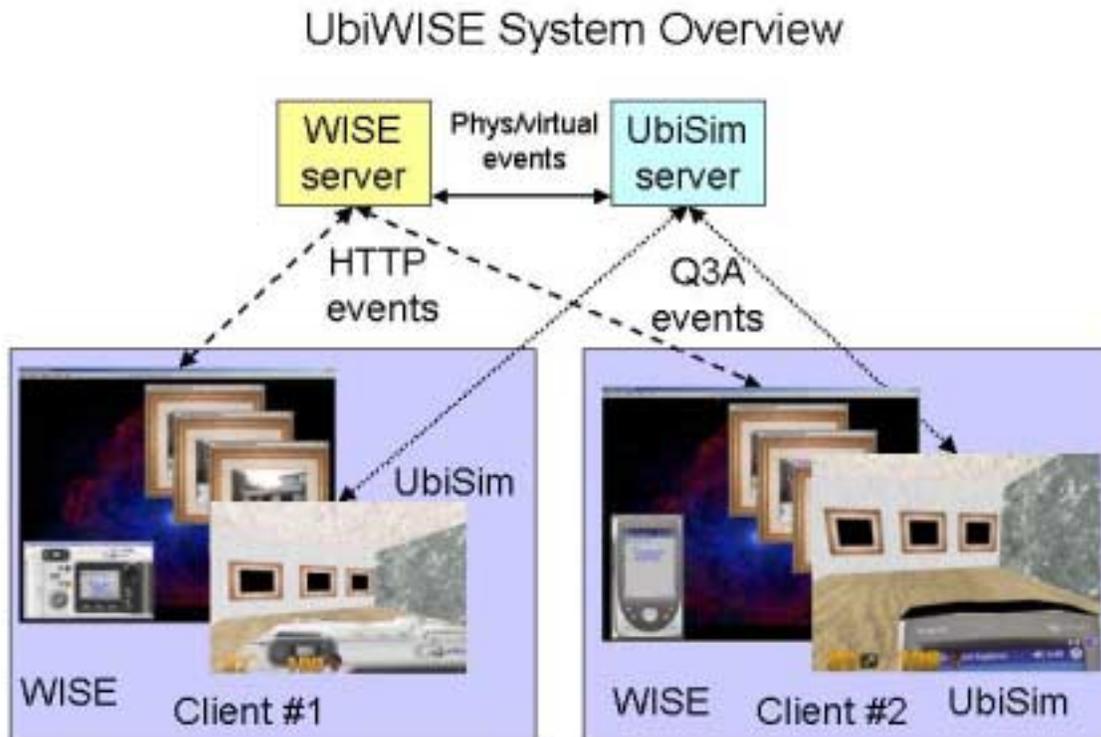
Figure 4 The connections between clients and servers in the Ubiwise simulator. Two user views are shown, each of which has two windows. Each window represents a client program connected to a server. The pairs of similar windows are connected by events sent to their corresponding server. The two subsystems communicate via messages between the servers.

Consider the same example scenario discussed above. When the scenario starts, UbiSim server creates three picture frames for its model of the room. The UbiSim server sends 3 *device creation* events to the WISE server. When the camera user connects, a fourth device creation event is sent to the WISE server, marked as belonging to that user. The WISE server transmits these 4 device names to the WISE client when it connects, giving the view of Fig.1 and Fig. 2.

Now suppose the camera user sends a picture to a picture frame. The image must be seen on all four clients of Fig. 4. The image starts in the WISE client of the camera user. The WISE client/server path is used to transfer the new image first to the WISE server and then down to the other WISE client. Next the WISE server sends its counterpart, the UbiSim server, a *data-update* message for the corresponding picture frame. The UbiSim server broadcasts this to the UbiSim clients. The update message has the filename of the image previously sent along the WISE path. In this way we get all four clients updated but only send a short filename across the time-sensitive UbiSim pathway.

Discussion

UbiWise seeks to subsume the hard work of developing new hardware and software to allow us to concentrate on the connections among new devices and on the uses we can make of those connections. We claim that we are on a path to a 'ubiquitous computing simulator'. To support this claim we shall examine the extent to which we help researchers to develop systems that combine spontaneous interaction and physical-virtual connection, the characteristics cited by Kindberg & Fox[3] as distinguishing ubiquitous computing.

To simulate the challenges created by *spontaneous interaction*, UbiWise allows diverse experiments with device interaction. Many different kinds of devices can be created and their combinations tried out. The interaction of multiple users on various (possibly conflicting!) tasks, interacting with mobile and fixed devices can be simulated. A variety of computation and communications infrastructures designed to help with spontaneous interaction can be plugged into the simulator and tested. Realistic simulation of these mechanisms can include communications latency and disconnection.

To simulate *physical-virtual connection*, the simulator provides a realistic simulation of the physical world at time scale of human interaction. We have the ability to simulate mechanisms that ubiquitous computing research has developed to sense physical events, such as RFID tags, ultrasound beacons, or infrared signals. Conversely we can simulate changes in the physical world caused by events in the virtual world, such as the update of a picture frame with a camera image.

UbiWise allows both of these aspects of ubiquity to be explored together. No simulator will supplant experimentation with physical hardware in the hands of appropriately selected human users. However, we hope that UbiWise will help us understand the issues of ubiquity that inform the design of hardware and help us develop the supporting software and infrastructure.

The distinguishing characteristics of ubiquitous computing lead us in different directions than simulators for human-computer interface (HCI) experiments, computer-computer networking simulator[19], or simulators for specific devices. For our work, the human-computer interface must be challenged by spontaneous interaction. One certain way to achieve this uses other humans working on other computers, all within an environment that may effect the operations of the machines. Network simulation, specifically the work in ad-hoc networks, deals with some kinds of spontaneous interaction but the physical environment is usually enters these simulators as a problem to be solved rather than a source of data for a system. Simulators for devices are excellent tools for interface design but they are not aimed at understanding the role of the device in a larger system. Our goal is to focus on interaction among devices and between devices and their environment, areas not handled by other kinds of simulators.

As with all simulators we must confront a fundamental design tradeoff. The simulator must have adequate "fidelity". That is, the devices created in the simulator, the user interaction, and the environment provided must be close enough to reality to validate ideas. Otherwise the simulator will be ineffective. Balancing against fidelity is our desire for "simplicity": in the simulator it must take dramatically less time and human energy to create devices, protocols, environmental interaction scenarios than it would take with physical devices and environments. Fidelity opposes simplicity in simulations: making the simulator simpler costs fidelity and vice versa.

The UbiWise simulator attempts to balance fidelity and simplicity in a simulator for ubiquitous computing. If we accept that simplicity calls for a desktop PC based simulator, the combination of a state-of-the-art graphics engine like Q3A and a full screen resolution image for devices gives sufficient fidelity. Working with a time-sensitive physical simulator with 3D graphics written C language weighs against simplicity, but the Q3A code has proven reasonably straightforward and we have been able to limit its role to aspects of physical simulation. Moreover, the architecture and tool-set that comes with Q3A mitigates its complexity by providing a commercial off-the-shelf solution for ubiquitous computing scenario

development. By placing device-interface development and device protocol development in WISE under Java, the extensive Java tool-set for computer communications can be leveraged into our simulations. Thus we claim that we have made a reasonable balance.

Future work

We have already examined a number of applications for the simulator. A number of additions to the simulator would make it more widely applicable. Many ubiquitous computing scenarios involve speech: we need a simple mechanism for leveraging the PC's audio input and output for simulations. This would then be a basis for adding an `AudioBrowser` to complement our `ImageBrowser` and `WebBrowser` components. This would also lead us towards reasonable simulation of multimodal devices. Linking multiple UbiWise servers should be possible providing a means for scaling up the simulation worlds or alternatively this mechanism could be used to provide a series of related scenarios to illustrate or contrast approaches to ubiquitous computing.

Acknowledgements

References

[1]   Andrew C.   Huang, Benjamin C. Ling, John J. Barton, Armando Fox, "Making Computers Disapper: Appliance Data Services" ACM SIGMOBILE Seventh Annual International Conference on Mobile Computing and Networking (Mobicom) 2001, Rome Italy.

[2]   Mark Weiser, "Hot Topics: Ubiquitous Computing" *IEEE Computer*, October 1993.

[3]   Tim Kindberg, Armando Fox "System Software for Ubiquitous Computing", IEEE Pervasive Computing v1, p71.

[4]   Tim Kindberg, John Barton, Jeff Morgan, Gene Becker, Ilja Bedner, Debbie Caswell, Phillipe Debaty, Gita Gopal, Marcos Frid, Venky Krishnan, Howard Morris, Celine Pering, John Schettino, Bill Serra, and M. Spasojevic. "People, Places, Things: Web Presence for the Real World", MONET Vol. 7, No. 5 (October 2002). See also Cooltown Project, HPLabs : http://cooltown.hpl.hp.com

[5]   Markus Bylund and Fredrik Espinoza, "Testing and demonstrating context-aware services with Quake III Arena"**,** Communications of the ACM, ACM Press, New York, NY, USA, 2002, Pages: 46 - 48

[6]   Vikram Vijayraghavan and John J. Barton. "WISE - A Simulator Toolkit for Ubiquitous Computing Scenarios" UbiTools-'01 workshop, and http://www.hpl.hp.com/personal/John_Barton/Publications/WISE_Ubitools_3.pdf

[7]   Id Software, "Quake III Arena", http://www.idsoftware.com/

[8]   Anind K. Dey and Gregory D. Abowd "The Context Toolkit: Aiding the Development of Context-Aware Applications**"** In the *Workshop on Software Engineering for Wearable and Pervasive Computing,* Limerick, Ireland, June 6, 2000. See also http://www.cc.gatech.edu/fce/contexttoolkit/

[9]   Espinoza, F., Persson, P., Sandin, A., Nyström, H., Cacciatore. E. and Bylund, M., "GeoNotes: Social and Navigational Aspects of Location-Based Information Systems", in Abowd, Brumitt & Shafer (eds.) Ubicomp 2001: Ubiquitous Computing, International Conference Atlanta, Georgia, September 30 - October 2, Berlin: Springer, 2001, p. 2-17.

[10] Ubiwise is available as on http://devnet.hp.com/projects as three packages under projects "wise", "ubisim", and "nexus".

[11] "LGPL Lesser GNU Public License", http://www.gnu.org

[12] Sun Microsystems, Java Swing Toolkit, http://java.sun.com/products/jfc/tsc/index.html

[13] "X-Smiles, An Open XML-Browser for Exotic Devices", http://www.x-smiles.org/

[14] Glenn E. Krasner and Stephen T. Pope, "A cookbook for using the model view controller user interface paradigm in Smalltalk-80". Journal of Object-Orientated Programming, 1(3):26-49, August/September 1988

[15] Robert A. Duffy and others, "QERadiant" http://www.qeradiant.com/

[16] Mete Ciragan "MilkShape 3D" http://www.swissquake.ch/chumbalum-soft/

[17] Bluetooth protocol specification : http://www.ietf.org

[18] The Infrared Data Association, IRDA, http://www.irda.org/standards/standards.asp

[19] See for example, Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, John Heidemann, Ahmed Helmy, Polly Huang, Steven McCanne, Kannan Varadhan, Ya Xu, and Haobo Yu. "**Advances in Network Simulation."** IEEE

Computer, 33 (5), pp. 59-67, May, 2000. See also other papers cited on The Network Simulator - ns-2 http://www.isi.edu/nsnam/ns/