iRemote: a Platform for Controlling Public Displays with Mobile Phones

Olivier Liechti and Kenji Mase
ATR Media Integration and Communication Research Laboratories
Seika-cho, Soraku-gun, Kyoto 619-0288 Japan
{olivier,mase}@mic.atr.co.jp

INTRODUCTION

In his vision of ubiquitous computing [4], Mark Weiser described devices of three different scales and form factors: "tabs", "pads" and "boards". Today, devices belonging to these three categories are already very common. However, there is still very little support for *interoperability* between them. It is possible to synchronize data between a PDA and a desktop computer, but this is a rather limited function. More interesting is the pick-and-drop interaction technique [3], which makes the flow of information across devices easier. Other approaches have been described in [1, 2], focusing on the interaction between PDAs and electronic whiteboards in collaborative work settings.

The design of the iRemote platform was motivated by this lack of interoperability, with a special focus on two classes of devices. Firstly, Web-enabled mobile phones (categorized as "tabs"), which are personal, mobile and already widely adopted. Secondly, public displays (categorized as "boards"), which exist in various forms and scales. Examples include street displays (used mainly for advertisement), panels used in sport stadiums and concert halls, projection displays used at conferences and in schools, electronic whiteboards, and others.

The basic goal of iRemote is to make the increasing interactive surface accessible to people, and to explore a new generation of collaborative applications. By collaborative applications, we do not restrict our scope to collaborative work, but also consider entertainment and "communityware" applications. A simple example is to allow people to make digital graffiti on street displays, by entering commands on their mobile phones. Another example is an interactive game played simultaneously by a very large number of spectators in a sport stadium. In the remaining sections, we first describe the architecture of the system and review its components. We then describe a simple application developed on top of the platform and report on its deployment and evaluation.

ARCHITECTURE

The architecture of iRemote, illustrated in Figure 1, includes i) Web-enabled mobile phones, ii) a publicly available HTTP server, iii) a display controller and iv) an actual display hardware component.

The iRemote personal terminal

No custom hardware is required on the client side: any i–mode phone can be used to access the system. Furthermore, it would be easy to develop interfaces for other wireless WWW platforms, such as WAP. The user interface provided on the iRemote terminal is exclusively embedded within the micro-browser. In other words, it currently consists of HTML pages, where lists of hyperlinks simulate menus and are used to trigger actions. User input is possible via checkboxes, radio buttons, push buttons and text-input areas. As a simple example of user interaction, consider a hyperlink labeled "draw circle". When the user selects this hyperlink, a circle is immediately drawn on the public display. Another example is a HTML form, where the user can enter a text message. When a "send" button is selected, the message appears on the public display.

In order to initiate a session between a terminal and a public display, the only operation required is to enter the URL associated with the public display. Of course, this URL has to be communicated to the user by some mechanism. For instance, the URL can be shown directly on the display. Some applications may require more subtle mechanisms, for instance if access is restricted to a given user population.

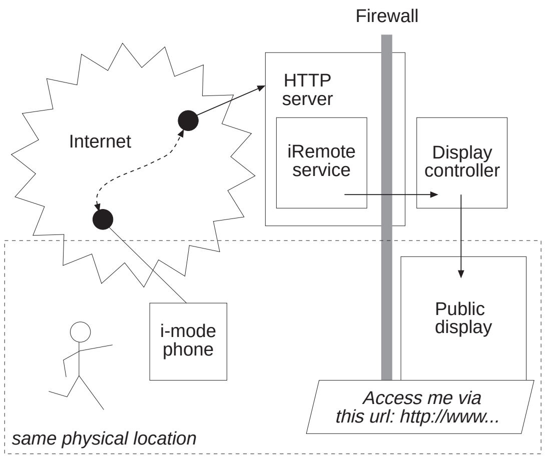

Figure 1. Architecture of the iRemote platform

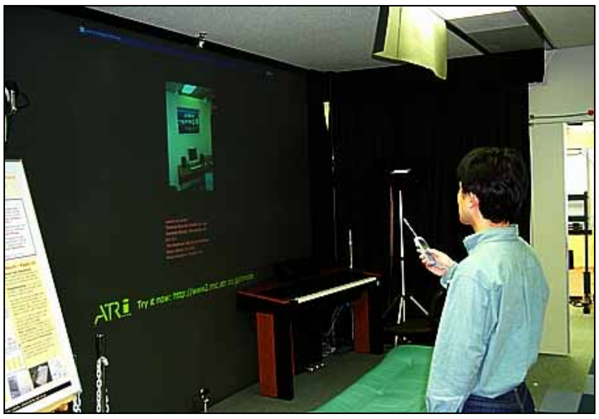

Figure 2. A user controls the flow of information

The iRemote HTTP server

The role of the HTTP server is i) to generate the user interface for the i-mode clients and ii) to interpret user actions and to forward commands to the display controller. Obviously, the HTTP server must be accessible to i-mode clients, which

i-リモコン:携帯電話による公衆ディスプレイの制御機構
オリビエ・リヒティ，間瀬健二，ATR知能映像通信研究所

means that it must be on the public Internet. Because the iRemote display controller is likely to be protected by a firewall, a bridging mechanism is necessary. Essentially, the software running on the HTTP server consists of CGI scripts that dynamically generate HTML pages. A number of constraints are imposed by the use of i-mode. For example, the fact that cookies cannot be used requires alternative mechanisms for implementing sessions over HTTP.

The iRemote display controller

The display controller is the software component that drives the display hardware. Hence, it depends on the technology chosen for an actual deployment (e.g. some displays may simply consist of a matrix of LEDs). Currently, we have only used wall-projected displays driven by personal computers – hence the problem is not different from controlling a desktop monitor.

The current version of the display controller is implemented in Java. More than a single application, it is an API that integrates a collection of drawing primitives and animations techniques (e.g. scrolling a text, fading an image in/out). Application designers can use this API to implement their own systems. Additionally, the display controller may support some of the application logic. In the application described in this paper, it is for example responsible for fetching multimedia content in a backend database.

The iRemote public display

As we already mentioned, the choice for one particular hardware technology depends on the situation. Whether the screen is located indoors or outdoors, its size and its intended use are some of the factors to consider. Support for a custom technology needs to be integrated to the iRemote platform.

FIRST APPLICATION: FUNCTIONS AND EVALUATION

The architecture described in the previous section is very general and was designed to support a range of interactive applications. We already mentioned messaging and gaming – other examples include electronic bulletin boards (situated in real-world contexts), interactive art installations and novel advertising techniques. To illustrate the use of the platform and gain a better understanding of related issues, we designed a first application and deployed it at the occasion of the ATR Open House exhibition, held in November 2000.

Collaborative browsing

The first function of the application allows the users to control the flow of information projected on the display (see Figure 2). The user interface supporting this task consists of a hierarchical menu, where users can successively select categories (e.g. "display news feeds", "display photographs") and sub-categories (e.g. "international news", "Japanese news", "photographs taken in Japan"). Hence, users do not request for a very specific piece of content to be displayed on the screen. Rather, they only indicate a category. The application then selects some multimedia content that fits the category and displays it. There is therefore an element of surprise for the user, which is quite interesting. A collection of software agents has been implemented to automatically extract content from WWW sites and feed it into the database.

This function was very popular with users, probably because the experience of controlling a large interactive surface is empowering and enjoyable. But beyond this direct benefit, a more interesting outcome of the system use resides in the social interactions that it promotes. Indeed, at many occasions we observed that people standing in front of the display started to talk *about* the projected information. The information provided a context for the starting a conversation and was a catalyst for informal communication.

Synchronous and asynchronous messaging

The first function does not support the creation of content by users, but only the display of already existing content. On the other hand, the second function allows users to create text messages and to feed them in the system.

The user interface supporting this task is very simple. Users first click on a hyperlink labeled "Send a message", which displays an HTML form. In a text area, they enter a message using the keys of their phone. They can also identify themselves, by entering a name in a field, or decide to remain anonymous. Finally, a "send" button can be clicked, which transmits the message first to the iRemote HTTP server, then to the iRemote display controller.

When a new message is sent, it is immediately displayed on the screen. This supports synchronous communication between the people currently looking at the display. For instance, imagine teenagers in Shibuya sending each other (love) messages using a similar system. Also, messages are stored in a database and periodically re-displayed on the screen. This supports asynchronous communication within a geographically bound community: it is possible to see messages from people who were "there" in the past.

In the first deployment of the system, this function was not very successful, for several reasons. One of them relates to a common user-interface problem, namely the difficulty to enter text on a mobile phone. Although many visitors had an i-mode phone, very few were able to use the text-entry system efficiently. Another factor was that because of the nature of the event, and the affluence of visitors, people did not spend a very long time in front of the display.

CONCLUSION

The implementation and first trial of the prototype was quite successful. The system was robust, there was little latency between user requests and feedback on the display. As a result, users seemed to enjoy the experience a lot. However, the experience raised a number of issues, which will be the focus of our future work.

REFERENCES

1. Greenberg, S., M. Boyle, and J. Laberge, *PDAs and Shared Public Displays: Making Personal Information Public, and Public Information Private*. Personal Technologies, 1999. **1&2**(3): p. 54-64.
2. Myers, B.A., H. Stiel, and R. Gargiulo. *Collaboration Using Multiple PDAs Connected to a PC*. in proceedings of the ACM Conference on Computer supported cooperative work (CSCW'98). November 1998, Seattle, WA. ACM.
3. Rekimoto, J. *Pick-and-Drop: A Direct Manipulation Technique for Multiple Computer Environments*. in proceedings of the ACM Symposium on User Inteface Software and Technology (UIST'97), Banff, Alberta, Canada. ACM Press.
4. Weiser, M., *The Computer for the 21st Century*, in *Scientific American*. 1991. p. 94-104.